SPRINGER
REFERENCE

Christodoulos A. Floudas
Panos M. Pardalos
Editors

Encyclopedia of Optimization

2nd Edition

Springer

Encyclopedia of Optimization

Second Edition

C. A. Floudas and P. M. Pardalos (Eds.)

Encyclopedia of Optimization

Second Edition

With 613 Figures and 247 Tables

Christodoulos A. Floudas
Department of Chemical Engineering
Princeton University
Princeton, NJ 08544-5263
USA
floudas@titan.princeton.edu

Panos M. Pardalos
Center for Applied Optimization
Department of Industrial and Systems Engineering
University of Florida
Gainesville, FL 32611-6595
USA
pardalos@cao.ise.ufl.edu

Library of Congress Control Number: 2008927531

ISBN: 978-0-387-74759-0

The print publication is available under ISBN: 978-0-387-74758-3
The print and electronic bundle is available under ISBN: 978-0-387-74760-6

© 2009 Springer Science+Buisiness Media, LLC.

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC., 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.
The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

springer.com

Printed on acid free paper SPIN: 11680840 2109letex – 5 4 3 2 1 0

Preface to the Second Edition

Optimization may be regarded as the cornerstone of many areas of applied mathematics, computer science, engineering, and a number of other scientific disciplines. Among other things, optimization plays a key role in finding feasible solutions to real-life problems, from mathematical programming to operations research, economics, management science, business, medicine, life science, and artificial intelligence, to mention only several.

Optimization entails engaging in an action to find the best solution. As a flourishing research activity, it has led to theoretical and computational advances, new technologies and new methods in developing more optimal designs of different systems, efficiency, and robustness, in minimizing the costs of operations in a process, and maximizing the profits of a company.

The first edition of the encyclopedia of optimization was well received by the scientific community and has been an invaluable source of scientific information for researchers, practitioners, and students.

Given the enormous yearly increases in this field since the appearance of the first edition, additional optimization knowledge has been added to this second edition. As before, entries are arranged in alphabetical order; the style of the entries has been retained to emphasize the expository and survey-type nature of the articles. Also many older entries have been updated and revised in light of new developments. Finally, several improvements have been made in the format to allow for more links to appropriate internet cites and electronic availability.

Acknowledgments

We wish to thank all invited contributors for their excellent efforts in writing their article in an expository way so that it is accessible to most scientists. The editors want to also take this opportunity to express gratitude to all the contributors, their research groups, their families for their support and encouragement, their research sponsors, and to Princeton University, and the University of Florida.

Many thanks go especially to Kerstin Kindler at Springer for her efficiency, terrific organization skills, and friendly spirit through the entire project. We would like also to thank the advisory board for their support and suggestions. Finally, we would like to thank the editors of Springer, Ann Kostant and Elizabeth Loew, for their thoughtful guidance, assistance and support during the planning and preparation of the second edition.

C.A. Floudas and P.M. Pardalos
Editors

About the Editors

Christodoulos A. Floudas

Christodoulos A. Floudas is the Stephen C. Macaleer '63 Professor in Engineering and Applied Science and Professor of Chemical Engineering at Princeton University. Concurrent Faculty positions include Center for Quantitative Biology at the Lewis-Sigler Institute, Program in Applied and Computational Mathematics, and Department of Operations Research and Financial Engineering. He has held Visiting Professor positions at Imperial College, Swiss Federal Institute of Technology, ETH, University of Vienna, and the Chemical Process Engineering Research Institute (CPERI), Thessaloniki, Greece.

Dr. Floudas obtained his Ph. D. from Carnegie Mellon University, and today is a world authority in mathematical modeling and optimization of complex systems. His research interests lie at the interface of chemical engineering, applied mathematics, and operations research with principal areas of focus including chemical process synthesis and design, process control and operations, discrete-continuous nonlinear optimization, local and global optimization, and computational chemistry and molecular biology.

He has received numerous awards for teaching and research including the NSF Presidential Young Investigator Award, the Engineering Council Teaching Award, the Bodossaki Foundation Award in Applied Sciences, the Best Paper Award in Computers and Chemical Engineering, Aspen Tech Excellence in Teaching Award, the 2001 AIChE Professional Progress Award for Outstanding Progress in Chemical Engineering, the 2006 AIChE Computing in Chemical Engineering Award, and the 2007 Graduate Mentoring Award. Dr. Floudas has served on the editorial boards of *Industrial Engineering Chemistry Research*, *Journal of Global Optimization*, *Computers and Chemical Engineering*, and various book series. He has authored 2 graduate textbooks, has co-edited several volumes, has published over 200 articles, and has delivered over 300 invited lectures and seminars.

Panos M. Pardalos

Panos M. Pardalos is Distinguished Professor of Industrial and Systems Engineering at the University of Florida and the director of the Center for Applied Optimization. He is also an affiliated faculty member of the Computer Science Department, the Hellenic Studies Center, and the Biomedical Engineering Program. Pardalos has held visiting appointments at Princeton University, DIMACS Center, Institute of Mathematics and Applications, Fields Institute, AT & T Labs Research, Trier University, Linkoping Institute of Technology, and universities in Greece.

Dr. Pardalos obtained his Ph. D. from the University of Minnesota, and today is a world leading expert in global and combinatorial optimization. His primary research interests include network design problems, optimization in telecommunications, e-commerce, data mining, biomedical applications, and massive computing. He has been an invited lecturer at several universities and research institutes around the world and has organized several international conferences.

He has received numerous awards including University of Florida Research Foundation Professor, UF Doctoral Dissertation Advisor/Mentoring Award, Foreign Member of the Royal Academy of Doctors (Spain), Foreign Member Lithuanian Academy of Sciences, Foreign Member of the Ukrainian Academy of Sciences, Foreign Member of the Petrovskaya Academy of Sciences and Arts (Russia), and Honorary Member of the Mongolian Academy of Sciences. He has also received an Honorary Doctorate degree from Lobachevski University, he is a fellow of AAAS, a fellow of INFORMS, and in 2001 he was awarded the Greek National Award and Gold Medal for Operations Research. Panos Pardalos is the editor-in-chief of the *Journal of Global Optimization*, *Journal of Optimization Letters*, and *Computational Management Science*. Dr. Pardalos is also the managing editor of several book series, and a member of the editorial board of various international journals. He has authored 8 books, has edited several volumes, has written numerous articles, and has also developed several well known software packages.

Editorial Board Members

Dimitri P. Bertsekas
McAfee Professor of Engineering
Massachusetts Institute of Technology
Cambridge, MA, USA

John R. Birge
Jerry W. and Carol Lee Levin
Professor of Operations Management
The University of Chicago Graduate School
of Business
Chicago, IL, USA

Jonathan M. Borwein
FRSC, Canada Research Chair
Computer Science Department
Dalhousie University
Halifax, NS, Canada

Vladimir F. Demyanov
Applied Mathematics Department
St. Petersburg State University
St. Petersburg, Russia

Fred Glover
University of Colorado
Boulder, CO, USA

Olvi L. Mangasarian
Computer Sciences Department
University of Wisconsin
Madison, WI, USA

Robert R. Meyer
Computer Sciences Department
University of Wisconsin
Madison, WI, USA

Boris Mordukhovich
Department of Mathematics
Wayne State University
Detroit, MI, USA

Walter Murray
Department of Management Science and Engineering
Stanford University
Stanford, CA, USA

George L. Nemhauser
Chandler Chaired Professor
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA, USA

James B. Orlin
Edward Pennell Brooks Professor
of Operations Research
MIT Sloan School of Management
Cambridge, MA, USA

J. Ben Rosen
Computer Science Department
UCSD and University of Minnesota
La Jolla, CA, and Minneapolis, MN, USA

Robert B. Schnabel
Computer Science Department
University of Colorado
Boulder, CO, USA

Hanif D. Sherali
W. Thomas Rice Chaired Professor of Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA, USA

Ralph E. Steuer
Department of Banking & Finance
Terry College of Business
University of Georgia
Athens, GA, USA

TAMÁS TERLAKY
Canada Research Chair in Optimization
Director, McMaster School of Computational
Engineering and Science
McMaster University
Hamilton, ON, Canada

J.-PH. VIAL
Operations Management
University of Geneva
Geneva, Switzerland

HOANG TUY
Department of Optimization and Control
Institute of Mathematics, VAST
Hanoi, Vietnam

HENRY WOLKOWICZ
Faculty of Mathematics
University of Waterloo
Waterloo, ON, Canada

List of Contributors

ACKERMANN, JUERGEN
DLR Oberpfaffenhofen
Wessling, Germany

ADJIMAN, CLAIRE S.
Imperial College
London, UK

AGGOUN, ABDERRAHMANE
KLS-OPTIM
Villebon sur Yvette, France

AHMED, SHABBIR
University of Illinois
Urbana-Champaign, IL, USA

AHUJA, RAVINDRA K.
University of Florida
Gainesville, FL, USA

AIT SAHLIA, FARID
University of Florida
Gainesville, FL, USA

ALEVRAS, DIMITRIS
IBM Corporation
West Chester, PA, USA

ALEXANDROV, NATALIA M.
NASA Langley Res. Center
Hampton, VA, USA

ALIZAMIR, SAED
University of Florida
Gainesville, FL, USA

ALKAYA, DILEK
Carnegie Mellon University
Pittsburgh, PA, USA

ALVES, MARIA JOÃO
University Coimbra and INESC
Coimbra, Portugal

ANASTASSIOU, GEORGE A.
University of Memphis
Memphis, TN, USA

ANDERSEN, ERLING D.
Odense University
Odense M, Denmark

ANDREI, NECULAI
Center for Advanced Modeling and Optimization
and
Academy of Romanian Scientists
Bucharest, Romania

ANDRICIOAEI, IOAN
Boston University
Boston, MA, USA

ANDROULAKIS, IOANNIS P.
Rutgers University
Piscataway, NJ, USA

ANSTREICHER, KURT M.
University of Iowa
Iowa City, IA, USA

ARABIE, P.
Rutgers University
Newark, DE, USA

AURENHAMMER, FRANZ
Graz University of Technology
Graz, Austria

AUSTIN-RODRIGUEZ, JENNIFER
Louisiana State University
Baton Rouge, LA, USA

BAGIROV, ADIL
University of Ballarat
Victoria, Australia

Balas, Egon
Carnegie Mellon University
Pittsburgh, PA, USA

Balasundaram, B.
Texas A&M University
College Station, TX, USA

Banerjee, Ipsita
Rutgers University
Piscataway, NJ, USA

Bao, Gang
Michigan State University
East Lansing, MI, USA

Bard, Jonathan F.
University of Texas
Austin, TX, USA

Bardow, André
RWTH Aachen University
Aachen, Germany

Barnette, Gregory
University of Florida
Gainsville, FL, USA

Barnhart, Cynthia
Massachusetts Institute of Technology
Cambridge, MA, USA

Battiti, Roberto
Universitá Trento
Povo (Trento), Italy

Beasley, John E.
The Management School, Imperial College
London, England

Becker, Otwin
University of Heidelberg
Heidelberg, Germany

Behnke, Henning
Institute Math. TU Clausthal
Clausthal, Germany

Beliakov, Gleb
Deakin University
Victoria, Australia

Bendsøe, Martin P.
Technical University of Denmark
Lyngby, Denmark

Benhamou, Frédéric
Université de Nantes
Nantes, France

Benson, Harold P.
University of Florida
Gainesville, FL, USA

Ben-Tal, Aharon
Technion – Israel Institute of Technology
Haifa, Israel

Bertsekas, Dimitri P.
Massachusetts Institute of Technology
Cambridge, MA, USA

Biegler, L. T.
Carnegie Mellon University
Pittsburgh, PA, USA

Billups, Stephen C.
University of Colorado, Denver
Denver, CO, USA

Birge, John R.
Northwestern University
Evanston, IL, USA

Birgin, Ernesto G.
University of São Paulo
São Paulo, Brazil

Bischof, Christian H.
RWTH Aachen University
Aachen, Germany

Björck, Åke
Linköping University
Linköping, Sweden

Board, John L.G.
London School of Economics and Political Sci.
London, UK

Bomze, Immanuel M.
University of Vienna
Wien, Austria

Borchers, Brian
New Mexico Technology
Socorro, NM, USA

Borgwardt, Karl Heinz
University of Augsburg
Augsburg, Germany

Bouyssou, Denis
LAMSADE University Paris Dauphine
Paris, France

Boyd, Stephen
Stanford University
Stanford, CA, USA

Brandeau, Margaret
Stanford University
Stanford, CA, USA

Brännlund, Ulf
Kungliga Tekniska Högskolan
Stockholm, Sweden

Brezinski, Claude
Université Sci. et Techn. Lille Flandres–Artois
Lille, France

Brimberg, Jack
Royal Military College of Canada
Kingston, ON, Canada

Brualdi, Richard A.
University of Wisconsin
Madison, WI, USA

Brucker, Peter
Universität Osnabrück
Osnabrück, Germany

Bruglieri, Maurizio
University of Camerino
Camerino, Italy

Burkard, Rainer E.
Technical University of Graz
Graz, Austria

Busygin, Stanislav
University of Florida
Gainesville, FL, USA

Butenko, S.
Texas A&M University
College Station, TX, USA

Calvete, Herminia I.
Universidad de Zaragoza
Zaragoza, Spain

Calvin, J. M.
New Jersey Institute of Technology
Newark, DE, USA

Cambini, Alberto
University of Pisa
Pisa, Italy

Capar, Ismail
Texas A&M University
College Station, TX, USA

Carlson, Dean A.
University of Toledo
Toledo, OH, USA

Caron, Richard J.
University of Windsor
Windsor, ON, Canada

Cervantes, Arturo
Mellon University
Pittsburgh, PA, USA

Cha, Meeyoung
Korean Advanced Institute of Science and Technology
Daejeon, Korea

Chang, ByungMan
Seoul National University of Technology
Seoul, Korea

Chaovalitwongse, Paveena
University of Florida
Gainesville, FL, USA

Chaovalitwongse, W. Art
Rutgers University
Piscataway, NJ, USA

Chardaire, Pierre
University of East Anglia
Norwich, UK

CHEN, ANTHONY
Utah State University
Logan, UT, USA

CHEN, JIANER
Texas A&M University
College Station, TX, USA

CHEN, QING
Louisiana State University
Baton Rouge, LA, USA

CHRISTIANSEN, MARIELLE
Norwegian University of Science and Technology
Trondheim, Norway

CIESLIK, DIETMAR
University of Greifswald
Greifswald, Germany

CIFARELLI, C.
Università di Roma "La Sapienza"
Roma, Italy

CIRIC, AMY
University of Cincinnati
Cincinnati, OH, USA

CLÍMACO, JOÃO
University Coimbra and INESC
Coimbra, Portugal

COMBETTES, PATRICK L.
University Pierre et Marie Curie
Paris, France
and
New York University
New York, NY, USA

COMMANDER, CLAYTON W.
University of Florida
Gainesville, FL, USA

CONEJEROS, RAÚL
Cambridge University
Cambridge, UK

CONSTANTIN, ZOPOUNIDIS
Technical University of Crete
Chania, Greece

CORLISS, GEORGE F.
Marquette University
Milwaukee, WI, USA

COTTLE, RICHARD W.
Stanford University
Stanford, CA, USA

CRAVEN, B. D.
University of Melbourne
Melbourne, VIC, Australia

CSENDES, TIBOR
University of Szeged
Szeged, Hungary

CUCINIELLO, SALVATORE
Italian Research Council
Napoli, Italy

DADUNA, JOACHIM R.
Fachhochschule für Wirtschaft (FHW) Berlin
Berlin, Germany

DANNINGER-UCHIDA, GABRIELE E.
University of Vienna
Vienna, Austria

DASCI, ABDULLAH
York University
Toronto, ON, Canada

DE ANGELIS, PASQUALE L.
Naval Institute
Naples, Italy

DE LEONE, R.
University degli Studi di Camerino
Camerino, Italy

DEMPE, STEPHAN
Freiberg University of Mining and Technology
Freiberg, Germany

DEMYANOV, VLADIMIR F.
St. Petersburg State University
St. Petersburg, Russia

DENG, NAIYANG
China Agricultural University
Beijing, China

DENG, XIAOTIE
City University of Hong Kong
Kowloon, China

DIDERICH, CLAUDE G.
Swiss Federal Institute of Technology
Lausanne, Switzerland

DI GIACOMO, LAURA
Università di Roma "La Sapienza"
Roma, Italy

DIMAGGIO JR., PETER A.
Princeton University
Princeton, NJ, USA

DIXON, LAURENCE
University of Hertfordshire
Hatfield, England

DOUMPOS, MICHAEL
Financial Engineering Lab. Techn. University Crete
Chania, Greece

DREZNER, TAMMY
California State University
Fullerton, CA, USA

DUA, PINKY
Imperial College
London, UK
and
GlaxoSmithKline Research & Development Limited
Harlow, UK

DUA, VIVEK
Imperial College
London, UK

DU, DING-ZHU
University of Texas at Dallas
Richardson, TX, USA

DUNN, JOSEPH C.
North Carolina State University
Raleigh, NC, USA

DUPAČOVÁ, JITKA
Charles University
Prague, Czech Republic

DYE, SHANE
University of Canterbury
Christchurch, New Zealand

EDIRISINGHE, CHANAKA
University of Tennessee
Knoxville, TN, USA

EDMONSON, WILLIAM
Hampton University
Hampton, VA, USA

EGLESE, RICHARD
Lancaster University
Lancaster, UK

GOUNARIS, CHRYSANTHOS E.
Princeton University
Princeton, NJ, USA

EKSIOGLU, BURAK
Mississippi State University
Mississippi State, MS, USA

EKSIOGLU, SANDRA DUNI
University of Florida
Gainesville, FL, USA

ELHEDHLI, SAMIR
McGill University
Montréal, QC, Canada

EMMONS, HAMILTON
Case Western Reserve University
Cleveland, OH, USA

ENGE, ANDREAS
University of Augsburg
Augsburg, Germany

ERLEBACH, THOMAS
University of Leicester
Leicester, UK

ERMOLIEV, YURI
International Institute for Applied Systems Analysis
Laxenburg, Austria

ESCUDERO, LAUREANO F.
M. Hernández University
Elche, Spain

Esposito, William R.
Princeton University
Princeton, NJ, USA

Evstigneev, Igor
Russian Acadamy of Science
Moscow, Russia

Fábián, Csaba I.
Eötvös Loránd University
Budapest, Hungary

Fagerholt, Kjetil
Norwegian University of Science and Technology
and
Norwegian Marine Technology Research Institute (MARINTEK)
Trondheim, Norway

Faísca, Nuno P.
Imperial College
London, UK

Fang, Shu-Cherng
North Carolina State University
Raleigh, NC, USA

Fang, Yuguang
University of Florida
Gainesville, FL, USA

Fan, Ya-Ju
Rutgers University
Piscataway, NJ, USA

Feminiano, Davide
Italian Research Council
Napoli, Italy

Feng, F. J.
University of Sussex
Sussex, England

Ferreira, Afonso
INRIA Sophia Antipolis
Sophia–Antipolis, France

Festa, Paola
Universitá Salerno
Baronissi, Italy

Fiacco, A. V.
George Washington University
Washington, DC, USA

Fischer, Herbert
Technical University of Munich
München, Germany

Flåm, Sjur Didrik
University of Bergen
Bergen, Norway

Floudas, Christodoulos A.
Princeton University
Princeton, NJ, USA

Forsgren, Anders
Royal Institute of Technology (KTH)
Stockholm, Sweden

Foulds, L. R.
University Waikato
Waikato, New Zealand

Frauendorfer, Karl
University of St. Gallen
St. Gallen, Switzerland

Frenk, Hans
Erasmus University
Rotterdam, The Netherlands

Fu, Chang-Jui
National Tsing Hua University
Hsinchu, Taiwan

Fung, Ho Ki
Princeton University
Princeton, NJ, USA

Fürer, Martin
Pennsylvania State University
University Park, PA, USA

Gaganis, Chrysovalantis
Technical University of Crete
Chania, Greece

Galé, Carmen
Universidad de Zaragoza
Zaragoza, Spain

GAO, DAVID YANG
Virginia Polytechnic Institute and State University
Blacksburg, VA, USA

GARCÍA-PALOMARES, UBALDO M.
University of Simón Bolívar
Caracas, Venezuela

GEHRLEIN, WILLIAM V.
University of Delaware
Newark, DE, USA

GENGLER, MARC
Université Méditerrannée
Marseille, France

GEROGIORGIS, DIMITRIOS I.
Imperial College
London, UK

GEUNES, JOSEPH
University of Florida
Gainesville, FL, USA

GIACOMO, LAURA DI
Università di Roma "La Sapienza"
Roma, Italy

GIANNESSI, FRANCO
University of Pisa
Pisa, Italy

GOELEVEN, DANIEL
I.R.E.M.I.A. University de la Réunion
Saint-Denis, France

GOFFIN, JEAN-LOUIS
McGill University
Montréal, QC, Canada

GOUNARIS, CHRYSANTHOS E.
Princeton University
Princeton, NJ, USA

GRAMA, ANANTH
Purdue University
West Lafayette, IN, USA

GRIEWANK, ANDREAS
Technical University of Dresden
Dresden, Germany

GRIPPO, LUIGI
Università di Roma "La Sapienza"
Roma, Italy

GROSSMANN, IGNACIO E.
Carnegie Mellon University
Pittsburgh, PA, USA

GRUNDEL, DON
671 ARSS/SYEA
Eglin AFB, FL, USA

GUARRACINO, MARIO R.
Italian Research Council
Napoli, Italy

GUDDAT, JÜRGEN
Humboldt University
Berlin, Germany

GUERRA, FRANCISCO
University of the Americas
Cholula, Mexico

GÜMÜŞ, ZEYNEP H.
University of Cincinnati
Cincinnati, OH, USA
and
Cornell University
New York, NY, USA

GUPTA, KAPIL
Georgia Institute of Technology
Atlanta, GA, USA

GÜRSOY, KORHAN
University of Cincinnati
Cincinnati, OH, USA

GUSTAFSON, SVEN-ÅKE
Stavanger University
Stavanger, Norway

GUTIN, GREGORY
University of London
Egham, UK

HADDAD, CAROLINE N.
State University of New York
Geneseo, NY, USA

HADJISAVVAS, NICOLAS
University of the Aegean
Hermoupolis, Greece

HAFTKA, RAPHAEL T.
University of Florida
Gainesville, FL, USA

HAMACHER, HORST W.
Universität Kaiserslautern
Kaiserslautern, Germany

HAN, CHI-GEUN
Kyung Hee University
Seoul, Korea

HANSEN, PIERRE
GERAD and HEC Montréal
Montréal, QC, Canada

HANSMANN, ULRICH H.E.
Michigan Technological University
Houghton, MI, USA

HARDING, S. T.
Princeton University
Princeton, NJ, USA

HARHAMMER, PETER G.
Technical University of Vienna
Vienna, Austria

HARJUNKOSKI, IIRO
Åbo Akademi University
Turku, Finland

HASLINGER, JAROSLAV
Charles University
Prague, Czech Republic

HATZINAKOS, DIMITRIOS
University of Toronto
Toronto, ON, Canada

HAUPTMAN, HERBERT A.
Hauptman–Woodward Medical Research Institute Inc.
Buffalo, NY, USA

HAURIE, ALAIN B.
University of Geneva
Geneva, Switzerland

HEARN, DONALD W.
University of Florida
Gainesville, FL, USA

HEINKENSCHLOSS, MATTHIAS
Rice University
Houston, TX, USA

HERTZ, DAVID
RAFAEL Department 82
Haifa, Israel

HETTICH, RAINER
University of Trier
Trier, Germany

HE, XIAOZHENG
University of Minnesota
Minneapolis, MN, USA

HICKS, ILLYA V.
Rice University
Houston, TX, USA

HIGLE, JULIA L.
University of Arizona
Tucson, AZ, USA

HOFFMAN, KARLA
George Mason University
Fairfax, VA, USA

HOLDER, ALLEN
University Colorado
Denver, CO, USA

HOLMBERG, KAJ
Linköping Institute of Technology
Linköping, Sweden

HOOKER, J. N.
Carnegie Mellon University
Pittsburgh, PA, USA

HOVLAND, PAUL D.
Argonne National Lab.
Argonne, IL, USA

HUANG, HONG-XUAN
Tsinghua University
Beijing, P. R. China

HUANG, XIAOXIA
University of Florida
Gainesville, FL, USA

HUBERT, L. J.
University of Illinois
Champaign, IL, USA

HUNT III, H. B.
University of Albany
New York, NY, USA

HÜRLIMANN, TONY
University of Fribourg
Fribourg, Switzerland

HURSON, CHRISTIAN
University Rouen, CREGO
Mont Saint Aignan, France

IERAPETRITOU, MARIANTHI
Rutgers University
Piscataway, NJ, USA

IRI, MASAO
Chuo University
Tokyo, Japan

ISAC, GEORGE
Royal Military College of Canada
Kingston, ON, Canada

İZBIRAK, GÖ KHAN
Eastern Mediterranean University
Mersin-10, Turkey

JACOBSEN, STEPHEN E.
University of California
Los Angeles, CA, USA

JAHN, JOHANNES
University of Erlangen–Nürnberg
Erlangen, Germany

JANAK, STACY L.
Princeton University
Princeton, NJ, USA

JANSEN, KLAUS
Univerisität Kiel
Kiel, Germany

JANSSON, C.
Techn. Universität Hamburg-Harburg
Hamburg, Germany

JEFFCOAT, DAVID
AFRL/RWGN
Eglin AFB, FL, USA

JEYAKUMAR, V.
University of New South Wales
Sydney, NSW, Australia

JHA, KRISHNA C.
GTEC
Gainesville, FL, USA

JHONES, ALINA RUIZ
University of Havana San Lázaro y L
Ciudad Habana, Cuba

JIA, ZHENYA
Rutgers University
Piscataway, NJ, USA

JONES, DONALD R.
General Motors Corp.
Warren, MI, USA

JONGEN, HUBERTUS T.
RWTH Aachen University
Aachen, Germany

JUDSON, RICHARD S.
Genaissance Pharmaceuticals
New Haven, CT, USA

KAKLAMANIS, CHRISTOS
University of Patras
Patras, Greece

KALLRATH, JOSEF
BASF Aktiengesellschaft
Ludwigshafen, Germany
and
University of Florida
Gainesville, FL, USA

KALMAN, DAN
American University
Washington, DC, USA

KAMMERDINER, ALLA R.
University of Florida
Gainesville, FL, USA

KAPLAN, ALEXANDER
University of Trier
Trier, Germany

KAPLAN, UĞUR
Koç University
Istanbul, Turkey

KASAP, SUAT
University of Oklahoma
Norman, OK, USA

KAS, PÉTER
Eastern Mediterranean University
Mersin-10, Turkey

KATOH, NAOKI
Kyoto University
Kyoto, Japan

KEARFOTT, R. BAKER
University of Louisiana at Lafayette
Lafayette, LA, USA

KELLEY, C. T.
North Carolina State University
Raleigh, NC, USA

KENNINGTON, JEFFERY L.
Southern Methodist University
Dallas, TX, USA

KESAVAN, H. K.
University of Waterloo
Waterloo, ON, Canada

KHACHIYAN, LEONID
Rutgers University
Piscataway, NJ, USA

KIM, DUKWON
University of Florida
Gainesville, FL, USA

KISIALIOU, MIKALAI
University of Minnesota
Minneapolis, MN, USA

KLAFSZKY, EMIL
Technical University
Budapest, Hungary

KLATTE, DIETHARD
University of Zurich
Zurich, Switzerland

KLEPEIS, JOHN L.
Princeton University
Princeton, NJ, USA

KNIGHT, DOYLE
Rutgers University
New Brunswick, NJ, USA

KOBLER, DANIEL
Swiss Federal Institute of Technology
Lausanne, Switzerland

KOHOUT, LADISLAV J.
Florida State University
Tallahassee, FL, USA

KOMÁROMI, ÉVA
Budapest University of Economic Sciences
Budapest, Hungary

KONNOV, IGOR V.
Kazan University
Kazan, Russia

KORHONEN, PEKKA
Internat. Institute Applied Systems Analysis
Laxenburg, Austria
and
Helsinki School Economics and Business Adm.
Helsinki, Finland

KOROTKICH, VICTOR
Central Queensland University
Mackay, QLD, Australia

KORTANEK, K. O.
University of Iowa
Iowa City, IA, USA

KOSMIDIS, VASSILEIOS D.
Imperial College
London, UK

Kostreva, Michael M.
Clemson University
Clemson, SC, USA

Kouramas, K.I.
Imperial College
London, UK

Krabs, W.
University Darmstadt
Darmstadt, Germany

Krarup, Jakob
DIKU Universitetsparken 1
Copenhagen, Denmark

Krishnan, Niranjan
Massachusetts Institute of Technology
Cambridge, MA, USA

Krokhmal, Pavlo A.
University of Iowa
Iowa City, IA, USA

Kruger, Alexander Y.
University of Ballarat
Ballarat, VIC, Australia

Kubota, Koichi
Chuo University
Tokyo, Japan

Kuehrer, Martin
Siemens AG (NYSE: SI)
Wien, Austria

Kumar, Arvind
GTEC
Gainesville, FL, USA

Kumar, Vipin
University of Minnesota
Minneapolis, MN, USA

Kundakcioglu, O. Erhun
University of Florida
Gainesville, FL, USA

Kuno, Takahito
University of Tsukuba
Ibaraki, Japan

Kyparisis, George J.
Florida International University
Miami, FL, USA

Kyriaki, Kosmidou
Athens University Economics and Business
Athens, Greek

Lamar, Bruce W.
The MITRE Corp.
Bedford, MA, USA

Lancaster, Laura C.
Clemson University
Clemson, SC, USA

Laporte, Gilbert
HEC Montréal
Montréal, QC, Canada

Laurent, Monique
CWI
Amsterdam, The Netherlands

Lavor, Carlile
State University of Campinas (IMECC-UNICAMP)
Campinas, Brazil

Lawphongpanich, Siriphong
Naval Postgraduate School
Monterey, CA, USA

Leclerc, Anthony P.
College of Charleston
Charleston, SC, USA

Ledzewicz, Urszula
Southern Illinois University at Edwardsville
Edwardsville, IL, USA

Lee, Eva K.
Georgia Institute of Technology
Atlanta, GA, USA

Lee, Wen
University of Florida
Gainesville, FL, USA

Leopold-Wildburger, Ulrike
Karl-Franzens University of Graz
Graz, Austria

LEPP, RIHO
Tallinn Technical University
Tallinn, Estonia

LETCHFORD, ADAM
Lancaster University
Lancaster, UK

LEWIS, KAREN R.
Southern Methodist University
Dallas, TX, USA

LEYFFER, S.
University of Dundee
Dundee, UK

LIANG, ZHE
Rutgers University
Piscataway, NJ, USA

LIANG, ZHIAN
Shanghai University of Finance and Economics
Shanghai, P.R. China

LIBERTI, LEO
LIX
Palaiseau, France

LI, GUANGYE
Silicon Graphics, Inc.
Houston, TX, USA

LI, HAN-LIN
National Chiao Tung University
Hsinchu, Taiwan

LIM, GINO J.
University of Houston
Houston, TX, USA

LINDBERG, P. O.
Linköping University
Linköping, Sweden

LIN, YOUDONG
University of Notre Dame
Notre Dame, IN, USA

LISSER, ABDEL
France Telecom
Issy les Moulineaux, France

LIU, WENBIN
University of Kent
Canterbury, England

LIWO, ADAM
Cornell University
Ithaca, NY, USA

L. JANAK, STACY
Princeton University
Princeton, NJ, USA

LOCKHART BOGLE, IAN DAVID
University College London
London, UK

LOUVEAUX, FRANCOIS
University of Namur
Namur, Belgium

LOWE, TIMOTHY J.
University of Iowa
Iowa City, IA, USA

LU, BING
University of Minnesota
Minneapolis, MN, USA

LUCIA, ANGELO
University of Rhode Island
Kingston, RI, USA

LUO, ZHI-QUAN
University of Minnesota
Minneapolis, MN, USA

LUUS, REIN
University of Toronto
Toronto, ON, Canada

MAAREN, HANS VAN
Delft University of Technology
Delft, The Netherlands

MACULAN, NELSON
Federal University of Rio de Janeiro (COPPE-UFRJ)
Rio de Janeiro, Brazil

MAGNANTI, THOMAS L.
Massachusetts Institute of Technology
Cambridge, MA, USA

MAIER, HELMUT
University of Ulm
Ulm, Germany

MÄKELÄ, MARKO M.
University of Jyväskylä
Jyväskylä, Finland

MÁLYUSZ, LEVENTE
Technical University
Budapest, Hungary

MAMMADOV (MAMEDOV), MUSA
University of Ballarat
Ballarat, VIC, Australia

MAPONI, PIERLUIGI
University of Camerino
Camerino, Italy

MARANAS, COSTAS D.
Pennsylvania State University
University Park, PA, USA

MARAVELIAS, CHRISTOS
University of Wisconsin – Madison
Madison, WI, USA

MARCOTTE, PATRICE
University of Montréal
Montréal, QC, Canada

MARINAKI, MAGDALENE
Technical University of Crete
Chania, Greece

MARINAKIS, YANNIS
Technical University of Crete
Chania, Greece

MARINO, MARINA
University of Naples 'Federico II' and CPS
Naples, Italy

MARQUARDT, WOLFGANG
RWTH Aachen University
Aachen, Germany

MARTEIN, LAURA
University of Pisa
Pisa, Italy

MARTI, KURT
University of Munich
Neubiberg, Germany

MARTÍNEZ, J. M.
University of Campinas
Campinas, Brazil

MATOS, ANA C.
Université Sci. et Techn. Lille Flandres–Artois
Lille, France

MAVRIDOU, THELMA D.
University of Florida
Gainesville, FL, USA

MAVROCORDATOS, P.
Algotheque and Université Paris 6
Paris, France

MCALLISTER, S. R.
Princeton University
Princeton, NJ, USA

MCDONALD, CONOR M.
E.I. DuPont de Nemours & Co.
Wilmington, DE, USA

MEDVEDEV, VLADIMIR G.
Byelorussian State University
Minsk, Republic Belarus

MEULMAN, J.
Leiden University
Leiden, The Netherlands

MIETTINEN, MARKKU
University of Jyväskylä
Jyväskylä, Finland

MINOUX, MICHEL
University of Paris
Paris, France

MISSEN, RONALD W.
University of Toronto
Toronto, ON, Canada

MISTAKIDIS, EURIPIDIS
University of Thessaly
Volos, Greece

MITCHELL, JOHN E.
Math. Sci. Rensselaer Polytechnic Institute
Troy, NY, USA

MLADENOVIĆ, NENAD
Brunel University
Uxbridge, UK

MOCKUS, JONAS
Institute Math. and Informatics
Vilnius, Lithuania

MOHEBI, HOSSEIN
University of Kerman
Kerman, Iran

MONDAINI, RUBEM P.
Federal University of Rio de Janeiro, Centre of Technology/COPPE
Rio de Janeiro, Brazil

MONGEAU, MARCEL
University of Paul Sabatier
Toulouse, France

MÖNNIGMANN, MARTIN
Technische Universität Braunschweig
Braunschweig, Germany

MOON, SUE B.
Korean Advanced Institute of Science and Technology
Daejeon, Korea

MOORE, RAMON E.
Worthington, OH, USA

MORTON, DAVID P.
University of Texas at Austin
Austin, TX, USA

MOTREANU, DUMITRU
University of Alexandru Ioan Cuza
Iasi, Romania

MULVEY, JOHN M.
Princeton University
Princeton, NJ, USA

MURLI, ALMERICO
University of Naples Federico II
and
Center for Research on Parallel Computing and Supercomputers of the CNR (CPS-CNR)
Napoli, Italy

MUROTA, KAZUO
Res. Institute Math. Sci. Kyoto University
Kyoto, Japan

MURPHEY, ROBERT
US Air Force Research Labor.
Eglin AFB, FL, USA

MURRAY, WALTER
Stanford University
Stanford, CA, USA

MURTY, KATTA G.
University of Michigan
Ann Arbor, MI, USA

MUTZEL, PETRA
University of Wien
Wien, Austria

NAGURNEY, ANNA
University of Massachusetts
Amherst, MA, USA

NAHAPETYAN, ARTYOM G.
University of Florida
Gainesville, FL, USA

NAKAYAMA, HIROTAKA
Konan University
Kobe, Japan

NAZARETH, J. L.
Washington State University
Pullman, WA, USA
and
University of Washington
Seattle, WA, USA

NEMIROVSKI, ARKADI
Technion: Israel Institute Technology
Technion-City, Haifa, Israel

NGO, HUANG
State University of New York at Buffalo
Buffalo, NY, USA

NICKEL, STEFAN
Universität Kaiserslautern
Kaiserslautern, Germany

NIELSEN, SØREN S.
University of Copenhagen
Copenhagen, Denmark

NIÑO-MORA, JOSÉ
Universidad Carlos III de Madrid
Getafe, Spain

NOOR, MUHAMMAD ASLAM
Dalhousie University in Halifax
Halifax, NS, Canada

NOWACK, DIETER
Humboldt University
Berlin, Germany

OKAMOTO, YUKO
Graduate University of Adv. Studies
Okazaki, Japan

OLAFSSON, SIGURDUR
Iowa State University
Ames, IA, USA

OLSON, DAVID L.
University of Nebraska
Lincoln, NE, USA

ONN, SHMUEL
Technion – Israel Institute of Technology
Haifa, Israel

ORLIK, PETER
University of Wisconsin
Madison, WI, USA

ORLIN, JAMES B.
Massachusetts Institute of Technology
Cambridge, MA, USA

OULTON, R.F.
University of California at Berkeley
Berkeley, CA, USA

PACHTER, RUTH
Air Force Research Laboratory Materials &
Manufacturing Directorate
Wright–Patterson AFB, USA

PADBERG, MANFRED
New York University
New York, NY, USA

PALAGI, LAURA
Universitá di Roma "La Sapienza"
Roma, Italy

PANAGOULI, OLYMPIA
Aristotle University
Thessaloniki, Greece

PANICUCCI, BARBARA
University of Pisa
Pisa, Italy

PAPAGEORGIOU, LAZAROS G.
UCL (University College London)
London, UK

PAPAJORGJI, PETRAQ
University of Florida
Gainesville, FL, USA

PAPALEXANDRI, KATERINA P.
bp Upstream Technology
Middlesex, UK

PAPARRIZOS, KONSTANTINOS
University of Macedonia
Thessaloniki, Greece

PAPPALARDO, MASSIMO
University of Pisa
Pisa, Italy

PARDALOS, PANOS M.
University of Florida
Gainesville, FL, USA

PARPAS, PANOS
Imperial College
London, UK

PASIOURAS, FOTIOS
University of Bath
Bath, UK

PATRIKSSON, MICHAEL
Chalmers University of Technology
Göteborg, Sweden

PATRIZI, GIACOMO
Università di Roma "La Sapienza"
Roma, Italy

PELILLO, MARCELLO
Università Ca' Foscari di Venezia
Venice, Italy

PERSIANO, GIUSEPPE
Università di Salerno
Fisciano, Italy

PFLUG, GEORG
University of Vienna
Vienna, Austria

PHILLIPS, ANDREW T.
University of Wisconsin–Eau Claire
Eau Claire, WI, USA

PIAO TAN, MENG
Princeton University
Princeton, NJ, USA

PICKENHAIN, SABINE
Brandenburg Technical University Cottbus
Cottbus, Germany

PĬNAR, MUSTAFA Ç.
Bilkent University
Ankara, Turkey

PINTÉR, JÁNOS D.
Pintér Consulting Services, Inc.,
and Dalhousie University
Halifax, NS, Canada

PISTIKOPOULOS, EFSTRATIOS N.
Imperial College
London, UK

PITSOULIS, LEONIDAS
Princeton University
Princeton, NJ, USA

POLYAKOVA, LYUDMILA N.
St. Petersburg State University
St. Petersburg, Russia

POPOVA, ELMIRA
University of Texas at Austin
Austin, TX, USA

PÖRN, RAY
Åbo Akademi University
Turku, Finland

POTVIN, JEAN-YVES
University of Montréal
Montréal, QC, Canada

POURBAIX, DIMITRI
Royal Observatory of Belgium
Brussels, Belgium

PRÉKOPA, ANDRÁS
RUTCOR, Rutgers Center for Operations Research
Piscataway, NJ, USA

PROKOPYEV, OLEG
University of Pittsburgh
Pittsburgh, PA, USA

QI, LIQUN
University of New South Wales
Sydney, NSW, Australia

QUERIDO, TANIA
University of Florida
Gainesville, FL, USA

QUEYRANNE, MAURICE
University of British Columbia
Vancouver, BC, Canada

RADZIK, TOMASZ
King's College London
London, UK

RAGLE, MICHELLE A.
University of Florida
Gainesville, FL, USA

RAI, SANATAN
Case Western Reserve University
Cleveland, OH, USA

RAJGARIA, R.
Princeton University
Princeton, NJ, USA

RALL, L. B.
University of Wisconsin–Madison
Madison, WI, USA

RALPH, DANIEL
University of Melbourne
Melbourne, VIC, Australia

RAPCSÁK, TAMÁS
Hungarian Academy of Sciences
Budapest, Hungary

RASSIAS, THEMISTOCLES M.
University Athens Zografou Campus
Athens, Greece

RATSCHEK, H.
Universität Düsseldorf
Düsseldorf, Germany

RAYDAN, MARCOS
Universidad Central de Venezuela
Caracas, Venezuela

REBENNACK, STEFFEN
University of Florida
Gainesville, FL, USA

RECCHIONI, MARIA CRISTINA
University of Ancona
Ancona, Italy

REEMTSEN, REMBERT
Brandenburg Technical University Cottbus
Cottbus, Germany

REINEFELD, ALEXANDER
ZIB Berlin
Berlin, Germany

RESENDE, MAURICIO G.C.
AT&T Labs Res.
Florham Park, NJ, USA

RIBEIRO, CELSO C.
Catholic University Rio de Janeiro
Rio de Janeiro, Brazil

RIPOLL, DANIEL R.
Cornell University
Ithaca, NY, USA

R. KAMMERDINER, ALLA
University of Florida
Gainesville, FL, USA

ROKNE, J.
University of Calgary
Calgary, AB, Canada

ROMA, MASSIMO
Universitá di Roma "La Sapienza"
Roma, Italy

ROMEIJN, H. EDWIN
University of Florida
Gainsville, FL, USA

ROOS, KEES
Delft University of Technology
AJ Delft, The Netherlands

RUBINOV, A. M.
School Inform. Techn. and Math. Sci. University
Ballarat
Ballarat, VIC, Australia

RUBIN, PAUL A.
Michigan State University
East Lansing, MI, USA

RUBIO, J. E.
University of Leeds
Leeds, UK

RÜCKMANN, JAN-J.
Ilmenau University of Technology
Ilmenau, Germany

RUSTEM, BERÇ
Imperial College
London, UK

SAFONOV, MICHAEL G.
University Southern California
Los Angeles, CA, USA

SAGASTIZÁBAL, CLAUDIA
IMPA
Jardim Botânico, Brazil

SAHINIDIS, NIKOLAOS V.
University of Illinois
Urbana-Champaign, IL, USA

SAHIN, KEMAL
University of Cincinnati
Cincinnati, OH, USA

SAKIZLIS, V.
Bechtel Co. Ltd.
London, UK

SAMARAS, NIKOLAOS
University of Macedonia
Thessaloniki, Greece

SANCHEZ, SALVADOR NIETO
Louisiana State University
Baton Rouge, LA, USA

SARAIVA, PEDRO M.
Imperial College
London, UK

SAVARD, GILLES
École Polytechnique
Montréal, QC, Canada

SAVELSBERGH, MARTIN W.P.
Georgia Institute of Technology
Atlanta, GA, USA

SAYIN, SERPIL
Koç University
İstanbul, Turkey

SAYYADY, FATEMEH
North Carolina State University
Raleigh, NC, USA

SCHAIBLE, SIEGFRIED
University of California
Riverside, CA, USA

SCHÄTTLER, HEINZ
Washington University
St. Louis, MO, USA

SCHERAGA, HAROLD A.
Cornell University
Ithaca, NY, USA

SCHOEN, FABIO
Universita Firenze
Firenze, Italy

SCHULTZ, RÜDIGER
Gerhard-Mercator University
Duisburg, Germany

SCHÜRLE, MICHAEL
University of St. Gallen
St. Gallen, Switzerland

SCHWEIGER, CARL A.
Princeton University
Princeton, NJ, USA

SEN, SUVRAJEET
University of Arizona
Tucson, AZ, USA

SHAIKH, AMAN
AT&T Labs – Research
Florham Park, NJ, USA

SHAIK, MUNAWAR A.
Princeton University
Princeton, NJ, USA

SHALLOWAY, DAVID
Cornell University
Ithaca, NY, USA

SHAPIRO, ALEXANDER
Georgia Institute of Technology Atlanta
Atlanta, GA, USA

SHERALI, HANIF D.
Virginia Polytechnic Institute and State University
Blacksburg, VA, USA

SHETTY, BALA
Texas A&M University
College Station, TX, USA

SHI, LEYUAN
University of Wisconsin
Madison, WI, USA

SIM, MELVYN
NUS Business School Office
Singapore

SIMONE, VALENTINA DE
University Naples 'Federico II' and CPS
Naples, Italy

SIMONS, STEPHEN
University of California
Santa Barbara, CA, USA

SIRLANTZIS, K.
University of Kent
Canterbury, England

Siskos, Yannis
Technical University of Crete
Chania, Greece

Sivazlian, B. D.
University of Florida
Gainesville, FL, USA

Slowinski, Roman
Poznań University Technology
Poznań, Poland

Smith, Alexander Barton
Carnegie Mellon University
Pittsburgh, PA, USA

Smith, J. MacGregor
University of Massachusetts, Amherst
Amherst, MA, USA

Smith, William R.
University of Guelph
Guelph, ON, Canada

So, Anthony Man-Cho
The Chinese University of Hong Kong
Hong Kong, China

Sobieszczanski-Sobieski, Jaroslaw
NASA Langley Research Center
Hampton, VA, USA

Sokolowski, Jan
Université Henri Poincaré
Nancy, France

Solodov, Michael V.
Institute Mat. Pura e Apl.
Rio de Janeiro, Brazil

Spedicato, Emilio
University of Bergamo
Bergamo, Italy

Spieksma, Frits
Maastricht University
Maastricht, The Netherlands

Stadtherr, Mark A.
University of Notre Dame
Notre Dame, IN, USA

Stavroulakis, Georgios E.
Carolo Wilhelmina Technical University
Braunschweig, Germany

Stein, Oliver
University of Karlsruhe
Karlsruhe, Germany

Still, Claus
Åbo Akademi University
Åbo, Finland

Still, Georg
University of Twente
Enschede, The Netherlands

Straub, John
Boston University
Boston, MA, USA

Strongin, Roman G.
Nizhni Novgorod State University
Nizhni Novgorod, Russia

Sugihara, Kokichi
Graduate School of Engineering, University of Tokyo
Tokyo, Japan

Sun, Defeng
University of New South Wales
Sydney, NSW, Australia

Sutcliffe, Charles M.S.
University of Southampton
Southampton, UK

Svensson, Lars
Royal Institute of Technology (KTH)
Stockholm, Sweden

Szántai, Tamás
Technical University
Budapest, Hungary

Tan, Meng Piao
Princeton University
Princeton, NJ, USA

Tawarmalani, Mohit
University of Illinois
Urbana-Champaign, IL, USA

TEBOULLE, MARC
Tel-Aviv University
Ramat-Aviv, Tel-Aviv, Israel

TEGHEM, JACQUES
Polytechnique Mons
Mons, Belgium

TERLAKY, TAMÁS
McMaster University
Hamilton, ON, Canada

TESFATSION, LEIGH
Iowa State University
Ames, IA, USA

THENGVALL, BENJAMIN
CALEB Technologies Corp.
Austin, TX, USA

THOAI, NGUYEN V.
University of Trier
Trier, Germany

THOMAS, REKHA R.
University of Washington
Seattle, WA, USA

TICHATSCHKE, RAINER
University of Trier
Trier, Germany

TIND, JØRGEN
University of Copenhagen
Copenhagen, Denmark

TITS, ANDRÉ L.
University of Maryland
College Park, MD, USA

TORALDO, GERARDO
University of Naples 'Federico II' and CPS
Naples, Italy

TORVIK, VETLE I.
Louisiana State University
Baton Rouge, LA, USA

TRAFALIS, THEODORE B.
University of Oklahoma
Norman, OK, USA

TRAUB, J. F.
Columbia University
New York, NY, USA

TRIANTAPHYLLOU, EVANGELOS
Louisiana State University
Baton Rouge, LA, USA

TRLIFAJOVÁ, KATEŘINA
Charles University
Prague, Czech Republic

TSAI, JUNG-FA
National Taipei University of Technology
Taipei, Taiwan

TSAO, JACOB H.-S.
San José State University
San José, CA, USA

TSENG, PAUL
University of Washington
Seattle, WA, USA

TSIPLIDIS, KONSTANTINOS
University of Macedonia
Thessaloniki, Greece

TUNÇEL, LEVENT
University of Waterloo
Waterloo, ON, Canada

TÜRKAY, METIN
Koç University
Istanbul, Turkey

TUY, HOANG
Vietnamese Academy of Science and Technology
Hanoi, Vietnam

UBHAYA, VASANT A.
North Dakota State University
Fargo, ND, USA

URYASEV, S.
University of Florida
Gainesville, FL, USA

VAIDYANATHAN, BALACHANDRAN
University of Florida
Gainesville, FL, USA

VAIRAKTARAKIS, GEORGE
Case Western Reserve University
Cleveland, OH, USA

VANCE, PAMELA H.
Emory University
Atlanta, GA, USA

VANDENBERGHE, LIEVEN
University of California
Los Angeles, CA, USA

VAN DEN HEEVER, SUSARA
Carnegie Mellon University
Pittsburgh, PA, USA

VASANTHARAJAN, SRIRAM
Mobil Technology Company
Dallas, TX, USA

VASSILIADIS, VASSILIOS S.
Cambridge University
Cambridge, UK

VAVASIS, STEPHEN A.
Cornell University
Ithaca, NY, USA

VAZACOPOULOS, ALKIS
Dash Optimization
Englewood Cliffs, NJ, USA

VEMULAPATI, UDAYA BHASKAR
University of Central Florida
Orlando, FL, USA

VERTER, VEDAT
McGill University
Montréal, QC, Canada

VIAL, JEAN-PHILIPPE
University of Genève
Geneva, Switzerland

VICENTE, LUIS N.
University of Coimbra
Coimbra, Portugal

VINCKE, PH.
Université Libre de Bruxelles, Gestion
Brussels, Belgium

VISWESWARAN, VISWANATHAN
SCA Technologies, LLC
Pittsburgh, PA, USA

VLADIMIROU, HERCULES
University of Cyprus
Nicosia, Cyprus

VOPĚNKA, PETR
Charles University
Prague, Czech Republic

VOSS, STEFAN
University of Hamburg
Hamburg, Germany

VURAL, ARIF VOLKAN
Mississippi State University
Mississippi State, MS, USA

WALLACE, STEIN W.
Norwegian University Sci. and Techn.
Trondheim, Norway

WALTERS, JAMES B.
Marquette University
Milwaukee, WI, USA

WANG, YANJUN
Shanghai University of Finance and Economics
Shanghai, China

WANG, ZHIQIANG
Air Force Research Laboratory Materials &
Manufacturing Directorate
Wright-Patterson AFB, OH, USA

WATSON, LAYNE T.
Virginia Polytechnic Institute and State University
Blacksburg, VA, USA

WEI, JAMES
Princeton University
Princeton, NJ, USA

WERSCHULZ, A. G.
Fordham University
and
Columbia University
New York, NY, USA

WESOLOWSKY, GEORGE O.
McMaster University
Hamilton, ON, Canada

WESTERLUND, TAPIO
Åbo Akademi University
Åbo, Finland

WOLKOWICZ, HENRY
University of Waterloo
Waterloo, ON, Canada

WOOD, GRAHAM
Massey University
Palmerston North, New Zealand

WU, SHAO-PO
Stanford University
Stanford, CA, USA

WU, TSUNG-LIN
Georgia Institute of Technology
Atlanta, GA, USA

WU, WEILI
University of Minnesota
Minneapolis, MN, USA

WYNTER, LAURA
University of Versailles
Versailles-Cedex, France

XANTHOPOULOS, PETROS
University of Florida
Gainesville, FL, USA

XIA, ZUNQUAN
Dalian University of Technology
Dalian, China

XIE, WEI
American Airlines Operations Research and
Decision Support Group
Fort Worth, TX, USA

XI, SHAOLIN
Beijing Polytechnic University
Beijing, China

XU, CHENGXIAN
Xian Jiaotong University
Xian, China

XUE, JUE
City University of Hong Kong
Kowloon, Hong Kong

XU, YINFENG
Xian Jiaotong University
Xian, China

YAJIMA, YASUTOSHI
Tokyo Institute of Technology
Tokyo, Japan

YANG, ERIC
Rutgers University
Piscataway, NJ, USA

YATES, JENNIFER
AT&T Labs – Research
Florham Park, NJ, USA

YAVUZ, MESUT
University of Florida
Gainesvile, FL, USA

YE, YINYU
University of Iowa
Iowa City, IA, USA

YU, GANG
University of Texas at Austin
Austin, TX, USA

YÜKSEKTEPE, FADIME ÜNEY
Koç University
Istanbul, Turkey

ZABINSKY, ZELDA B.
University of Washington
Seattle, WA, USA

ZAMORA, JUAN M.
Universidad Autónoma Metropolitana-Iztapalapa
Mexico City, Mexico

ZELIKOVSKY, ALEXANDER
Georgia State University
Atlanta, GA, USA

ZENIOS, STAVROS A.
University of Cyprus
Nicosia, Cyprus

ZHANG, JIANZHONG
City University of Hong Kong
Kowloon Tong, Hong Kong

ZHANG, LIWEI
Dalian University of Technology
Dalian, China

ZHANG, QINGHONG
University of Iowa
Iowa City, IA, USA

ZHU, YUSHAN
Tsinghua University
Beijing, China

ZIEMBA, WILLIAM T.
University of British Columbia
Vancouver, BC, Canada

ŽILINSKAS, ANTANAS
Institute of Mathematics and Informatics
Vilnius, Lithuania

ŽILINSKAS, JULIUS
Institute of Mathematics and Informatics
Vilnius, Lithuania

ZIRILLI, FRANCESCO
Università di Roma "La Sapienza"
Roma, Italy

ZISSOPOULOS, DIMITRIOS
Technical Institute of West Macedonia
Kozani, Greece

ZLOBEC, SANJO
McGill University
West Montréal, QC, Canada

ZOCHOWSKI, ANTONI
Systems Research Institute of the Polish Academy of Sciences
Warsaw, Poland

ZOPOUNIDIS, CONSTANTIN
Technical University of Crete
Chania, Greece

ZOWE, JOCHEM
University of Nürnberg–Erlangen
Erlangen, Germany

A

ABS Algorithms for Linear Equations and Linear Least Squares

EMILIO SPEDICATO
Department Math., University Bergamo,
Bergamo, Italy

MSC2000: 65K05, 65K10

Article Outline

Keywords
Synonyms
The Scaled ABS Class: General Properties
Subclasses of the ABS Class
The Implicit LU Algorithm
and the Huang Algorithm
Other ABS Linear Solvers
ABS Methods for Linear Least Squares
See also
References

Keywords

Linear algebraic equations; Linear least squares; ABS methods; Abaffian matrices; Huang algorithm; Implicit LU algorithm; Implicit LX algorithm

Synonyms

Abaffi–Broyden–Spedicato algorithms for linear equations and linear least squares

The Scaled ABS Class: General Properties

ABS methods were introduced by [1], in a paper dealing originally only with solving linear equations via what is now called the *basic* or *unscaled ABS class*. The basic ABS class was later generalized to the so-called *scaled ABS class* and subsequently applied to linear least squares, nonlinear equations and optimization problems, see [2]. Preliminary work has also been initiated concerning *Diophantine equations*, with possible extensions to combinatorial optimization, and the eigenvalue problem. There are presently (1998) over 350 papers in the ABS field, see [11]. In this contribution we will review the basic properties and results of ABS methods for solving linear determined or underdetermined systems and overdetermined linear systems in the least squares sense.

Let us consider the linear determined or underdetermined system, where rank(A) is arbitrary

$$Ax = b, \quad x \in \mathbb{R}^n, b \in \mathbb{R}^m, \quad m \leq n, \tag{1}$$

or

$$a_i^\top x - b_i = 0, \quad i = 1, \ldots, m, \tag{2}$$

where

$$A = \begin{pmatrix} a_1^\top \\ \vdots \\ a_m^\top \end{pmatrix}. \tag{3}$$

The steps of the scaled ABS class algorithms are as follows:

A) Let $x_1 \in \mathbf{R}^n$ be arbitrary, $H_1 \in \mathbf{R}^{n,n}$ be nonsingular arbitrary, v_1 be an arbitrary nonzero vector in $\mathbf{R}^m$; set $i = 1$.

B) Compute the residual $r_i = Ax_i - b$. If $r_i = 0$, stop (x_i solves the problem); else compute $s_i = H_i A^\top v_i$. If $s_i \neq 0$, then go to C). If $s_i = 0$ and $\tau = v_i^\top r_i = 0$, then set $x_{i+1} = x_i$, $H_{i+1} = H_i$ and go to F), else stop (the system has no solution).

C) Compute the search vector p_i by

$$p_i = H_i^\top z_i, \tag{4}$$

where $z_i \in \mathbf{R}^n$ is arbitrary save for the condition

$$v_i^\top A H_i^\top z_i \neq 0. \tag{5}$$

D) Update the estimate of the solution by

$$x_{i+1} = x_i - \alpha_i p_i, \tag{6}$$

where the stepsize α_i is given by

$$\alpha_i = \frac{v_i^\top r_i}{r_i^\top A p_i}. \tag{7}$$

E) Update the matrix H_i by

$$H_{i+1} = H_i - \frac{H_i A^\top v_i w_i^\top H_i}{w_i^\top H_i A^\top v_i}, \tag{8}$$

where $w_i \in \mathbf{R}^n$ is arbitrary save for the condition

$$w_i^\top H_i A^\top v_i \neq 0. \tag{9}$$

F) If $i = m$, then stop (x_{m+1} solves the system), else define v_{i+1} as an arbitrary vector in $\mathbf{R}^m$ but linearly independent from $v_1, \ldots, v_i$, increment i by one and go to B).

The matrices H_i appearing in step E) are generalizations of (*oblique*) *projection matrices*. They probably first appeared in [16]. They have been named *Abaffians* since the first international conference on ABS methods (Luoyang, China, 1991) and this name will be used here.

The above recursion defines a class of algorithms, each particular method being determined by the choice of the parameters H_1, v_i, z_i, w_i. The *basic ABS class* is obtained by taking $v_i = e_i$, e_i being the ith unitary vector in $\mathbf{R}^m$. The parameters w_i, z_i, H_1 have been introduced respectively by J. Abaffy, C.G. Broyden and E. Spedicato, whose initials are referred to in the name of the class. It is possible to show that the scaled ABS class is a complete realization of the so-called *Petrov–Galerkin iteration* for solving a linear system (but the principle can be applied to more general problems), where the iteration has the form $x_{i+1} = x_i - \alpha_i p_i$ with α_i, p_i chosen so that the orthogonality relation $r_{i+1}^\top v_j = 0$, $j = 1, \ldots, i$, holds, the vectors v_j being arbitrary linearly independent. It appears that all deterministic algorithms in the literature having finite termination on a linear system are members of the scaled ABS class (this statement has been recently shown to be true also for the *quasi-Newton methods*, which are known to have under some conditions termination in at most $2n$ steps: the iterate of index $2i - 1$ generated by Broyden's iteration corresponds to the ith iterate of a certain algorithm in the ABS class).

Referring [2] for proofs, we give some of the general properties of methods of the scaled ABS class, assuming, for simplicity, that A has full rank.

- Define $V_i = (v_1, \ldots, v_i)$, $W_i = (w_1, \ldots, w_i)$. Then $H_{i+1} A^\top V_i = 0$, $H_{i+1}^\top W_i = 0$, meaning that vectors $A^\top v_j$, w_j, $j = 1, \ldots, i$, span the null spaces of H_{i+1} and its transpose, respectively.
- The vectors $H_i A^\top v_i$, $H_i^\top w_i$ are nonzero if and only if a_i, w_i are linearly independent from $a_1, \ldots, a_{i-1}$, $w_1, \ldots, w_{i-1}$, respectively.
- Define $P_i = (p_1, \ldots, p_i)$. Then the implicit factorization $V_i^\top A_i^\top P_i = L_i$ holds, where L_i is nonsingular lower triangular. From this relation, if $m = n$, one obtains the following semi-explicit factorization of the inverse, with $P = P_n$, $V = V_n$, $L = L_n$

$$A^{-1} = P L^{-1} V^\top. \tag{10}$$

For several choices of the matrix V the matrix L is diagonal, hence formula (10) gives a fully explicit factorization of the inverse as a byproduct of the ABS solution of a linear system, a property that does not hold for the classical solvers. It can also be shown that all possible factorizations of the form (10) can be obtained by proper parameter choices in the scaled ABS class, another completeness result.

- Define S_i and R_i by $S_i = (s_1, \ldots, s_i)$, $R_i = (r_1, \ldots, r_i)$, where $s_i = H_i A^\top v_i$, $r_i = H_i^\top w_i$. Then the Abaffian can be written in the form $H_{i+1} = H_1 - S_i R_i^\top$ and the vectors s_i, r_i can be built via a *Gram–Schmidt type iterations* involving the previous vectors (the search vector p_i can be built in a similar way). This representation of the Abaffian in terms of $2i$ vectors is computationally convenient when the number of equations is much less than the number of variables. Notice that there is also a representation in terms of $n - i$ vectors.

- A compact formula of the Abaffian in terms of the parameter matrices is the following

$$H_{i+1} = H_1 - H_1 A^\top V_i (W_i^\top H_1 A^\top V_i)^{-1} W_i^\top H_1. \tag{11}$$

  Letting $V = V_m$, $W = W_m$, one can show that the parameter matrices H_1, V, W are admissible (i. e. are such that condition (9) is satisfied) if and only if the matrix $Q = V^\top A H_1^\top W$ is *strongly nonsingular* (i. e. is LU factorizable). Notice that this condition can always be satisfied by suitable exchanges of the columns of V or W, equivalent to a row or a column pivoting on the matrix Q. If Q is strongly nonsingular and we take, as is done in all algorithms insofar considered, $z_i = w_i$, then condition (5) is also satisfied.

It can be shown that the *scaled ABS class* corresponds to applying (implicitly) the unscaled ABS algorithm to the scaled (or preconditioned) system $V^\top Ax = V^\top b$, where V is an arbitrary nonsingular matrix of order m. Therefore we see that the scaled ABS class is also complete with respect to all possible left preconditioning matrices, which in the ABS context are defined implicitly and dynamically (only the ith column of V is needed at the ith iteration, and it can also be a function of the previous column choices).

Subclasses of the ABS Class

In [1], nine subclasses are considered of the scaled ABS class. Here we quote three important subclasses.

- The *conjugate direction subclass*. This class is well defined under the condition (sufficient but not necessary) that A is symmetric and positive definite. It contains the *implicit Choleski algorithm*, the *Hestenes–Stiefel* and the *Lanczos algorithms*. This class generates all possible algorithms whose search directions are A-conjugate. The vector x_{i+1} minimizes the *energy* or *A-weighted Euclidean norm* of the error over $x_1 + \mathrm{Span}(p_1, \ldots, p_i)$. If $x_1 = 0$, then the solution is approached monotonically from below in the energy norm.
- The *orthogonally scaled subclass*. This class is well defined if A has full column rank and remains well defined even if m is greater than n. It contains the ABS formulation of the QR algorithm (the so-called *implicit QR algorithm*), of the *GMRES* and of the *conjugate residual algorithms*. The scaling vectors are orthogonal and the search vectors are $AA^\top$-conjugate. The vector x_{i+1} minimizes the Euclidean norm of the residual over $x_1 + \mathrm{Span}(p_1, \ldots, p_i)$. In general, the methods in this class can be applied to overdetermined systems to obtain the solution in the least squares sense.
- The *optimally scaled subclass*. This class is obtained by the choice $v_i = A^{-\top} p_i$. The inverse of $A^\top$ disappears in the actual formulas, if we make the change of variables $z_i = A^\top u_i$, u_i being now the parameter that defines the search vector. For $u_i = e_i$ the *Huang method* is obtained and for $u_i = r_i$ a method equivalent to *Craig's conjugate gradient type algorithm*. From the general implicit factorization relation one obtains $P^\top P = D$ or $V^\top AA^\top V = D$, a relation which was shown in [5] to characterize the optimal choice of the parameters in the general Petrov–Galerkin process in terms of minimizing the effect of a single error in x_i on the final computed solution. Such a property is therefore satisfied by the Huang (and the Craig) algorithm, but not, for instance, by the implicit LU or the implicit QR algorithms. A. Galantai [8] has shown that the condition characterizing the optimal choice of the scaling parameters in terms of minimizing the final residual Euclidean norm is $V^\top V = D$, a condition satisfied by the implicit QR algorithm, the GMRES method, the implicit LU algorithm and again by the Huang algorithm, which therefore satisfies both conditions). The methods in the optimally stable subclass have the property that x_{i+1} minimizes the Euclidean norm of the error over $x_1 + \mathrm{Span}(p_1, \ldots, p_i)$.

The Implicit LU Algorithm and the Huang Algorithm

Specific algorithms of the scaled ABS class are obtained by choosing the available parameters. The *implicit LU algorithm* is given by the choices $H_1 = I$, $z_i = w_i = v_i = e_i$. We quote the following properties of the implicit LU algorithm.

a) The algorithm is well defined if and only if A is *regular* (i. e. all principal submatrices are nonsingular). Otherwise column pivoting has to be performed (or, if $m = n$, equations pivoting).

b) The Abaffian H_{i+1} has the following structure, with $K_i \in \mathbf{R}^{n-i,\,i}$:

$$H_{i+1} = \begin{pmatrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ K_i & I_{n-i} \end{pmatrix}. \tag{12}$$

c) Only the first i components of p_i can be nonzero and the ith component is one. Hence the matrix P_i is unit upper triangular, so that the implicit factorization $A = LP^{-1}$ is of the LU type, with units on the diagonal, justifying the name.

d) Only K_i has to be updated. The algorithm requires $nm^2 - 2m^3/3$ multiplications plus lower order terms, hence, for $m = n$, $n^3/3$ multiplications plus lower order terms. This is the same overhead required by the *classical LU factorization* or *Gaussian elimination* (which are two essentially equivalent processes).

e) The main storage requirement is the storage of K_i, whose maximum value is $n^2/4$. This is two times less than the storage needed by Gaussian elimination and four times less than the storage needed by the LU factorization algorithm (assuming that A is not overwritten). Hence the implicit LU algorithm is computationally better than the classical Gaussian elimination or LU algorithm, having the same overhead but less memory cost.

The implicit LU algorithm, implemented in the case $m = n$ with row pivoting, has been shown in experiments of M. Bertocchi and Spedicato [3] to be numerically stable and in experiments of E. Bodon [4] on the vector processor Alliant FX 80 with 8 processors to be about twice faster than the *LAPACK* implementation of the classical LU algorithm.

The *Huang algorithm* is obtained by the parameter choices $H_1 = I$, $z_i = w_i = a_i$, $v_i = e_i$. A mathematically equivalent, but numerically more stable, formulation of this algorithm is the so-called *modified Huang algorithm* where the search vectors and the Abaffians are given by formulas $p_i = H_i(H_i a_i)$ and $H_{i+1} = H_i - p_i p_i^\top / p_i^\top p_i$. Some properties of this algorithm follow.

- The search vectors are orthogonal and are the same vectors obtained by applying the classical Gram–Schmidt orthogonalization procedure to the rows of A. The modified Huang algorithm is related, but is not numerically identical, with the *Daniel–Gragg–Kaufmann–Stewart reorthogonalized Gram–Schmidt algorithm* [6].
- If x_1 is the zero vector, then the vector x_{i+1} is the solution with least Euclidean norm of the first i equations and the solution x^+ of least Euclidean norm of the whole system is approached monotonically and from below by the sequence x_i. L. Zhang [17] has shown that the Huang algorithm can be applied, via the *Goldfarb–Idnani active set strategy* [9], to systems of linear inequalities. The process in a finite number of steps either finds the solution with least Euclidean norm or determines that the system has no solution.
- While the error growth in the Huang algorithm is governed by the square of the number $\eta_i = \| a_i \| / \| H_i a_i \|$, which is certainly large for some i if A is ill conditioned, the error growth depends only on η_i if p_i or H_i are defined as in the modified Huang algorithm and, at first order, there is no error growth for the modified Huang algorithm.
- Numerical experiments, see [15], have shown that the modified Huang algorithm is very stable, giving usually better accuracy in the computed solution than both the implicit LU algorithm and the classical LU factorization method.

The *implicit LX algorithm* is defined by the choices $H_1 = I$, $v_i = e_i$, $z_i = w_i = e_{k_i}$, where k_i is an integer, $1 \le k_i \le n$, such that

$$e_{k_i}^\top H_i a_i \neq 0. \tag{13}$$

Notice that by a general property of the ABS class for A with full rank there is at least one index k_i such that (13) is satisfied. For stability reasons it may be recommended to select k_i such that $\eta_i = |e_{k_i}^\top H_i a_i|$ is maximized.

The following properties are valid for the implicit LX algorithm. Let N be the set of integers from 1 to n, $N = (1, \ldots, n)$. Let B_i be the set of indexes $k_1, \ldots, k_i$ chosen for the parameters of the implicit LX algorithm up to the step i. Let N_i be the set $N \setminus B_i$. Then:

- The index k_i is selected in the set N_{i-1}.
- The rows of H_{i+1} of index $k \in B_i$ are null rows.
- The vector p_i has $n - i$ zero components; its k_ith component is equal to one.
- If $x_1 = 0$, then x_{i+1} is a basic type solution of the first i equations, whose nonzero components may lie

only in the positions corresponding to the indices $k \in B_i$.

- The columns of H_{i+1} of index $k \in N_i$ are the unit vectors e_k, while the columns of H_{i+1} of index $k \in B_i$ have zero components in the jth position, with $j \in B_i$, implying that only $i(n-i)$ elements of such columns have to be computed.
- At the ith step $i(n-i)$ multiplications are needed to compute $H_i a_i$ and $i(n-i)$ to update the nontrivial part of H_i. Hence the total number of multiplications is the same as for the implicit LU algorithm (i. e. $n^3/3$), but no pivoting is necessary, reflecting the fact that no condition is required on the matrix A.
- The storage requirement is the same as for the implicit LU algorithm, i. e. at most $n^2/4$. Hence the implicit LX algorithm shares the same storage advantage of the implicit LU algorithm over the classical LU algorithm, with the additional advantage of not requiring pivoting.
- Numerical experiments by K. Mirnia [10] have shown that the implicit LX method gives usually better accuracy, in terms of error in the computed solution, than the implicit LU algorithm and often even than the modified Huang algorithm. In terms of size of the final residual, its accuracy is comparable to that of the LU algorithm as implemented (with row pivoting) in the MATLAB or LAPACK libraries, but it is better again in terms of error in the solution.

Other ABS Linear Solvers

ABS reformulations have been obtained for most algorithms proposed in the literature. The availability of several formulations of the linear algebra of the ABS process allows alternative formulations of each method, with possibly different values of overhead, storage and different properties of numerical stability, vectorization and parallelization. The reprojection technique, already seen in the case of the modified Huang algorithm and based upon the identities $H_i q = H_i(H_i q)$, $H_i^\top = H_i^\top (H_i^\top q)$, valid for any vector q if $H_1 = I$, remarkably improves the stability of the algorithm. The ABS versions of the *Hestenes–Stiefel* and the *Craig algorithms* for instance are very stable under the above reprojection. The *implicit QR algorithm*, defined by the choices $H_1 = I$, $v_i = Ap_i$, $z_i = w_i = e_i$ can be implemented in a very stable way using the reprojection in both the definition of the search vector and the scaling vector. It should also be noticed that the classical iterative refinement procedure, which amounts to a Newton iteration on the system $Ax - b = 0$ using the approximate factors of A, can be reformulated in the ABS context using the previously defined search vectors p_i. Experiments of Mirnia [11] have shown that ABS refinement works excellently.

For problems with special structure ABS methods can often be implemented taking into account the effect of the structure on the Abaffian matrix, which often tends to reflect the structure of the matrix A. For instance, if A has a banded structure, the same is true for the Abaffian matrix generated by the implicit LU, the implicit QR and the Huang algorithm, albeit the band size is increased. If A is SPD and has a *ND* structure, the same is true for the Abaffian matrix. In this case the implementation of the implicit LU algorithm has much less storage cost, for large n, than the cost required by an implementation of the Choleski algorithm. For matrices having the Kuhn–Tucker structure (KT structure) large classes of ABS methods have been devised, see ► ABS algorithms for optimization. For matrices with general sparsity patterns little is presently known about minimizing the fill-in in the Abaffian matrix. Careful use of BLAS4 routines can however substantially reduce the number of operations and make the ABS implementation competitive with a sparse implementation of say the LU factorization (e. g. by the code MA28) for values of n not too big.

It is possible to implement the ABS process also in block form, where several equations, instead of just one, are dealt with at each step. The block formulation does not deteriorate the numerical accuracy and can lead to reduction of overhead on special problems or to faster implementations on vector or parallel computers.

Finally infinite iterative methods can be obtained by the finite ABS methods via two approaches. The first one consists in restarting the iteration after $k < m$ steps, so that the storage will be of order $2kn$ if the representation of the Abaffian in terms of $2i$ vectors is used. The second approach consists in using only a limited number of terms in the Gram–Schmidt type processes that are alternative formulations of the ABS procedure. For both cases convergence at a linear rate has been established using the technique developed in [7]. The infinite

iteration methods obtained by these approaches define a very large class of methods, that contains not only all *Krylov space type methods* of the literature, but also non-Krylov type methods as the *Gauss–Seidel*, the *De La Garza* and the *Kackmartz methods*, with their generalizations.

ABS Methods for Linear Least Squares

There are several ways of using ABS methods for solving in the least squares sense an overdetermined linear system without forming the normal equations of Gauss, which are usually avoided on the account of their higher conditioning. One possibility is to compute explicitly the factors associated with the implicit factorization and then use them in the standard way. From results of [14] the obtained methods work well, giving usually better results than the methods using the QR factorization computed in the standard way. A second possibility is to use the representation of the *Moore–Penrose pseudo-inverse* that is provided explicitly by the ABS technique described in [13]. Again this approach has given very good numerical results. A third possibility is based upon the equivalence of the normal system $A^\top Ax = A^\top b$ with the extended system in the variables $x \in \mathbf{R}^n$, $y \in \mathbf{R}^m$, given by the two subsystems $Ax = y$, $A^\top y = A^\top b$. The first of the subsystems is overdetermined but must be solvable. Hence y must lie in the range of $A^\top$, which means that y must be the solution of least Euclidean norm of the second underdetermined subsystem. Such a solution is computed by the Huang algorithm. Then the ABS algorithm, applied to the first subsystem, in step B) recognizes and eliminates the $m - k$ dependent equations, where k is the rank of A. If $k < n$ there are infinite solutions and the one of least Euclidean norm is obtained by using again the Huang algorithm on the first subsystem.

Finally a large class of ABS methods can be applied directly to an overdetermined system stopping after n iterations in a least squares solution. The class is obtained by defining $V = AU$, where U is an arbitrary nonsingular matrix in $\mathbf{R}^n$. Indeed at the point x_{n+1} the satisfied Petrov–Galerkin condition is just equivalent to the normal equations of Gauss. If $U = P$ then the orthogonally scaled class is obtained, implying, as already stated in section 2, that the methods of this class can be applied to solve linear least squares (but a suitable modification has to be made for the *GMRES* method). A version of the implicit QR algorithm, with reprojection on both the search vector and the scaling vector, tested in [12], has outperformed other ABS algorithms for linear least squares methods as well as methods in the *LINPACK* and *NAG library* based upon the classical QR factorization via the Householder matrices.

See also

- ▶ ABS Algorithms for Optimization
- ▶ Cholesky Factorization
- ▶ Gauss–Newton Method: Least Squares, Relation to Newton's Method
- ▶ Generalized Total Least Squares
- ▶ Interval Linear Systems
- ▶ Large Scale Trust Region Problems
- ▶ Large Scale Unconstrained Optimization
- ▶ Least Squares Orthogonal Polynomials
- ▶ Least Squares Problems
- ▶ Linear Programming
- ▶ Nonlinear Least Squares: Newton-type Methods
- ▶ Nonlinear Least Squares Problems
- ▶ Nonlinear Least Squares: Trust Region Methods
- ▶ Orthogonal Triangularization
- ▶ Overdetermined Systems of Linear Equations
- ▶ QR Factorization
- ▶ Solving Large Scale and Sparse Semidefinite Programs
- ▶ Symmetric Systems of Linear Equations

References

1. Abaffy J, Broyden CG, Spedicato E (1984) A class of direct methods for linear systems. Numerische Math, 45:361–376
2. Abaffy J, Spedicato E (1989) ABS projection algorithms: Mathematical techniques for linear and nonlinear equations. Horwood, Westergate
3. Bertocchi M, Spedicato E (1989) Performance of the implicit Gauss–Choleski algorithm of the ABS class on the IBM 3090 VF. In: Proc. 10th Symp. Algorithms, Strbske Pleso, pp 30–40
4. Bodon E (1993) Numerical experiments on the ABS algorithms for linear systems of equations. Report DMSIA Univ Bergamo 93(17)
5. Broyden CG (1985) On the numerical stability of Huang's and related methods. JOTA 47:401–412
6. Daniel J, Gragg WB, Kaufman L, Stewart GW (1976) Reorthogonalized and stable algorithms for updating

the Gram–Schmidt QR factorization. Math Comput 30: 772–795
7. Dennis J, Turner K (1987) Generalized conjugate directions. Linear Alg & Its Appl 88/89:187–209
8. Galantai A (1991) Analysis of error propagation in the ABS class. Ann Inst Statist Math 43:597–603
9. Goldfarb D, Idnani A (1983) A numerically stable dual method for solving strictly convex quadratic programming. Math Program 27:1–33
10. Mirnia K (1996) Numerical experiments with iterative refinement of solutions of linear equations by ABS methods. Report DMSIA Univ Bergamo 32/96
11. Nicolai S, Spedicato E (1997) A bibliography of the ABS methods. OMS 8:171–183
12. Spedicato E, Bodon E (1989) Solving linear least squares by orthogonal factorization and pseudoinverse computation via the modified Huang algorithm in the ABS class. Computing 42:195–205
13. Spedicato E, Bodon E (1992) Numerical behaviour of the implicit QR algorithm in the ABS class for linear least squares. Ricerca Oper 22:43–55
14. Spedicato E, Bodon E (1993) Solution of linear least squares via the ABS algorithm. Math Program 58:111–136
15. Spedicato E, Vespucci MT (1993) Variations on the Gram-Schmidt and the Huang algorithms for linear systems: A numerical study. Appl Math 2:81–100
16. Wedderburn JHM (1934) Lectures on matrices. Colloq Publ Amer Math Soc
17. Zhang L (1995) An algorithm for the least Euclidean norm solution of a linear system of inequalities via the Huang ABS algorithm and the Goldfarb–Idnani strategy. Report DMSIA Univ Bergamo 95/2

ABS Algorithms for Optimization

Emilio Spedicato[1], Zunquan Xia[2], Liwei Zhang[2]
[1] Department Math., University Bergamo, Bergamo, Italy
[2] Department Applied Math., Dalian University Technol., Dalian, China

MSC2000: 65K05, 65K10

Article Outline

Keywords
A Class of ABS Projection Methods for Unconstrained Optimization
Applications to Quasi-Newton Methods
ABS Methods for Kuhn–Tucker Equations
Reformulation of the Simplex Method via the Implicit LX Algorithm
ABS Unification of Feasible Direction Methods for Minimization with Linear Constraints
See also
References

Keywords

Linear equations; Optimization; ABS methods; Quasi-Newton methods; Linear programming; Feasible direction methods; KT equations; Interior point methods

The *scaled ABS* (Abaffy–Broyden–Spedicato) class of algorithms, see [1] and ► ABS algorithms for linear equations and linear least squares, is a very general process for solving linear equations, realizing the so-called *Petrov–Galerkin approach*. In addition to solving general determined or underdetermined linear systems $Ax = b$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $m \leq n$, $\text{rank}(A) \leq m$, $A = [a_1, \ldots a_m]^\mathsf{T}$, ABS methods can also solve linear least squares problems and nonlinear algebraic equations. In this article we will consider applications of ABS methods to optimization problems. We will consider only the so-called *basic ABS class*, defined by the following procedure for solving $Ax = b$:

A) Let $x_1 \in \mathbb{R}^n$ be arbitrary, $H_1 \in \mathbb{R}^{n,n}$ be nonsingular arbitrary, set $i = 1$.
B) Compute $s_i = H_i\, a_i$. IF $s_i \neq 0$, go to C).
IF $s_i = 0$ and $\tau = a_i^\top x_i - b_i = 0$, THEN set $x_{i+1} = x_i$, $H_{i+1} = H_i$ and go to F), ELSE stop, the system has no solution.
C) Compute the search vector p_i by $p_i = H_i^\top\, z_i$, where $z_i \in \mathbb{R}^n$ is arbitrary save for the condition $a_i^\top\, H_i^\top\, z_i \neq 0$.
D) Update the estimate of the solution by $x_{i+1} = x_i - \alpha_i p_i$, where the stepsize α_i is given by $\alpha_i = (a_i^\top p_i - b_i)/a_i^\top p_i$.
E) Update the matrix H_i by $H_{i+1} = H_i - H_i a_i w_i^\top H_i / w_i^\top H_i a_i$, where $w_i \in \mathbb{R}^n$ is arbitrary save for the condition $w_i^\top H_i a_i \neq 0$.
F) IF $i = m$, THEN stop; x_{m+1} solves the system, ELSE increment i by one and go to B).

Among the properties of the ABS class the following is fundamental in the applications to optimization. Let

$m < n$ and, for simplicity, assume that rank$(A) = m$. Then the linear variety containing all solutions of the underdetermined system $Ax = b$ is represented by the vectors x of the form

$$x = x_{m+1} + H_{m+1}^{\top} q, \tag{1}$$

where $q \in \mathbb{R}^n$ is arbitrary. In the following the matrices generated by the ABS process will be called *Abaffians*. It is recalled that the matrix H_{i+1} can be represented in terms of either $2i$ vectors or of $n - i$ vectors, which is also true for the representation of the search vector p_i. The first representation is computationally convenient for systems where the number of equations is small (less than $n/2$), while the second one is suitable for problems where m is close to n. In the applications to optimization, the first case corresponds to problems with few constraints (many degrees of freedom), the second case to problems with many constraints (few degrees of freedom).

Among the algorithms of the basic ABS class, the following are particularly important.

a) The *implicit LU algorithm* is given by the choices $H_1 = I$, $z_i = w_i = e_i$, where e_i is the ith unit vector in $\mathbb{R}^n$. This algorithm is well defined if and only if A is regular (otherwise pivoting of the columns has to be performed, or of the equations, if $m = n$). Due to the special structure of the Abaffian induced by the parameter choices (the first i rows of H_{i+1} are identically zero, while the last $n - i$ columns are unit vectors) the maximum storage is $n^2/4$, hence 4 times less than for the classical LU factorization or twice less than for *Gaussian elimination*; the number of multiplications is $nm^2 - 2m^3/3$, hence, for $m = n$, $n^3/3$, i. e. the same as for Gaussian elimination or the LU factorization algorithm.

b) The *Huang algorithm* is obtained by the parameter choices $H_1 = I$, $z_i = w_i = a_i$. A mathematically equivalent, but numerically more stable, formulation of this algorithm is the so-called *modified Huang algorithm* where the search vectors and the Abaffians are given by formulas $p_i = H_i(H_i a_i)$ and $H_{i+1} = H_i - p_i p_i^{\top}/p_i^{\top} p_i$. The search vectors are orthogonal and are equal to the vectors obtained by applying the classical *Gram–Schmidt orthogonalization* procedure to the rows of A. If x_1 is the zero vector, then the vector x_{i+1} is the solution of least Euclidean norm of the first i equations and the solution x^+ of least Euclidean norm of the whole system is approached monotonically and from below by the sequence x_i.

c) The *implicit LX algorithm*, where 'L' refers to the lower triangular left factor while 'X' refers to the right factor, which is a matrix obtainable after row permutation of an upper triangular matrix, considered by Z. Xia, is defined by the choices $H_1 = I$, $z_i = w_i = e_{k_i}$ where k_i is an integer, $1 \le k_i \le n$, such that

$$e_{k_i}^{\top} H_i a_i \neq 0. \tag{2}$$

If A has full rank, from a property of the basic ABS class the vector $H_i a_i$ is nonzero, hence there is at least one index k_i such that (2) is satisfied. The implicit LX algorithm has the same overhead as the implicit LU algorithm, hence the same as Gaussian elimination, and the same storage requirement, i. e. less than Gaussian elimination or the LU factorization algorithm. It has the additional advantage of not requiring any condition on the matrix A, hence pivoting is not necessary. The structure of the Abaffian matrix is somewhat more complicated than for the implicit LU algorithm, the zero rows of H_{i+1} being now in the positions $k_1, \ldots, k_i$ and the columns that are unit vectors being in the positions that do not correspond to the already chosen indices k_i.

The vector p_i has $n - i$ zero components and its k_ith component is equal to one. It follows that if $x_1 = 0$, then x_{i+1} is a basic type solution of the first i equations, whose nonzero components correspond to the chosen indices k_i.

In this paper we will present the following applications of ABS methods to optimization problems. In Section 2 we describe a class of ABS related methods for the unconstrained optimization problem. In Section 3 we show how ABS methods provide the general solution of the quasi-Newton equation, also with sparsity and symmetry and we discuss how SPD solutions can be obtained. In Section 4 we present several special ABS methods for solving the Kuhn–Tucker equations. In Section 5 we consider the application of the implicit LX algorithm to the linear programming (LP) problem. In Section 6 we present ABS approaches to the general linearly constrained optimization problem, which unify linear and nonlinear problems.

A Class of ABS Projection Methods for Unconstrained Optimization

ABS methods can be applied directly to solve *unconstrained optimization* problems via the iteration $x_{i+1} = x_i - \alpha_i H_i^\top z_i$, where H_i is reset after n or less steps and z_i is chosen so that the descent condition holds, i. e. $g_i^\top H_i^\top z_i > 0$, with g_i the gradient of the function at x_i. If the function to be minimized is quadratic, one can identify the matrix A in the Abaffian update formula with the Hessian of the quadratic function. Defining a perturbed point x' by $x' = x_i - \beta\ v_i$ one has on quadratic functions $g' = g - \beta\ Av_i$, hence the update of the Abaffian takes the form $H_{i+1} = H_i - H_i y_i w_i^\top H_i / w_i^\top H_i y_i$, where $y_i = g' - g_i$. The above defined class has termination on quadratic functions and local superlinear (n-step Q-quadratic) rate of convergence on general functions. It is a special case of a class of projection methods developed in [7]. Almost no numerical results are available about the performance of the methods in this class.

Applications to Quasi-Newton Methods

ABS methods have been used to provide the general solution of the quasi-Newton equation, also with the additional conditions of symmetry, sparsity and positive definiteness. While the general solution of only the quasi-Newton equation was already known from [2], the explicit formulas obtained for the sparse symmetric case are new, and so is the way of constructing sparse SPD updates.

Let us consider the quasi-Newton equation defining the new approximation to a Jacobian or a Hessian, in the transpose form

$$d^\top B' = y^\top , \tag{3}$$

where $d = x' - x$, $y = g' - g$. We observe that (3) can be seen as a set of n linear underdetermined systems, each one having just one equation and differing only in the right-hand side. Hence the general solution can be obtained by one step of the ABS method. It can be written in the following way

$$B' = B - \frac{s(B^\top d - y)^\top}{d^\top s} + \left(I - \frac{sd^\top}{d^\top s}\right) Q , \tag{4}$$

where $Q \in \mathbb{R}^{n,\,n}$ is arbitrary and $s \in \mathbb{R}^n$ is arbitrary subject to $s^\intercal d \neq 0$. Formula (4), derived in [9], is equivalent to the formula in [2].

Now the conditions that some elements of B' should be zero, or have constant value or that B' should be symmetric can be written as the additional linear constraints, where b'_i is the ith column of B'

$$(b'_i)^\top e_k = \eta_{ij} , \tag{5}$$

where $\eta_{ij} = 0$ implies sparsity, $\eta_{ij} = \text{const}$ implies that some elements do not change their value and $\eta_{ij} = \eta_{ji}$ implies symmetry. The ABS algorithm can deal with these extra conditions, see [11], giving the solution in explicit form, columnwise in presence of symmetry. By adding the additional condition that the diagonal elements be sufficiently large, it is possible to obtain formulas where B' is quasi positive definite or quasi diagonally dominant, in the sense that the principal submatrix of order $n - 1$ is positive definite or diagonally dominant. It is not possible in general to force B' to be SPD, since SPD solutions may not exist, which is reflected in the fact that no additional conditions can be put on the last diagonal element, since the last column is fully determined by the $n - 1$ symmetry conditions and the quasi-Newton equation. This result can however be exploited to provide SPD approximations by imbedding the original minimization problem of n variables in a problem of $n + 1$ variables, whose solution with respect to the first n variables is the original solution (just set, for instance, $f(x') = f(x) + x_{n+1}^2$). This imbedding modifies the quasi-Newton equation so that SPD solutions exist.

ABS Methods for Kuhn–Tucker Equations

The *Kuhn–Tucker equations* (KT equations), which should more appropriately be named *Kantorovich–Karush–Kuhn–Tucker equations* (KKKT equations), are a special linear system, obtained by writing the optimality conditions of the problem of minimizing a quadratic function with Hessian G subject to the linear equality constraint $Cx = b$. They are the system $Ax = b$, where A is a symmetric indefinite matrix of the following form, with $G \in \mathbb{R}^{n,\,n}$, $C \in \mathbb{R}^{m,\,n}$

$$A = \begin{pmatrix} G & C^\top \\ C & 0 \end{pmatrix} . \tag{6}$$

If G is nonsingular, then A is nonsingular if and only if $CG^{-1}C^\intercal$ is nonsingular. Usually G is nonsingular, symmetric and positive definite, but this assumption, required by several classical solvers, is not necessary for the ABS solvers.

ABS classes for solving the KT problem can be derived in several ways. Observe that system (6) is equivalent to the two subsystems

$$Gp + C^\top z = g, \tag{7}$$
$$Cp = c, \tag{8}$$

where $x = (p^\intercal, z^\intercal)^\intercal$ and $b = (g^\intercal, C^\intercal)^\intercal$. The general solution of subsystem (8) has the form, see (1)

$$p = p_{m+1} + H_{m+1}^\top q, \tag{9}$$

with q arbitrary. The parameter choices made to construct p_{m+1} and H_{m+1} are arbitrary and define therefore a class of algorithms.

Since the KT equations have a unique solution, there must be a choice of q in (9) which makes p be the unique n-dimensional subvector defined by the first n components of the solution x. Notice that since H_{m+1} is singular, q is not uniquely defined (but would be uniquely defined if one takes the representation of the Abaffian in terms of $n - m$ vectors).

By multiplying equation (7) on the left by H_{m+1} and using the ABS property $H_{m+1}\ C^\intercal = 0$, we obtain the equation

$$H_{m+1}Gp = H_{m+1}g, \tag{10}$$

which does not contain z. Now there are two possibilities to determine p:

A1) Consider the system formed by equations (8) and (10). Such a system is solvable but overdetermined. Since $\text{rank}(H_{m+1}) = n - m$, m equations are recognized as dependent and are eliminated in step B) of any ABS algorithm applied to this system.

A2) In equation (10) substitute p with the expression of the general solution (9) obtaining

$$H_{m+1}GH_{m+1}^\top q = H_{m+1}g - H_{m+1}Gp_{m+1}. \tag{11}$$

The above system can be solved by any ABS method for a particular solution q, m equations being again removed at step B) of the ABS algorithm as linearly dependent.

Once p is determined, there are two approaches to determine z, namely:

B1) Solve by any ABS method the overdetermined compatible system

$$C^\top z = g - Gp \tag{12}$$

by removing at step B) of the ABS algorithm the $n - m$ dependent equations.

B2) Let $P = (p_1, \ldots p_m)$ be the matrix whose columns are the search vectors generated on the system $Cp = c$. Now $CP = L$, with L nonsingular lower diagonal. Multiplying equation (12) on the left by $P^\intercal$ we obtain a triangular system, defining z uniquely

$$L^\top z = P^\top g - P^\top Gp. \tag{13}$$

Extensive numerical testing has evaluated the accuracy of the above considered ABS algorithms for KT equations for certain choices of the ABS parameters (corresponding to the implicit LU algorithm with row pivoting and the modified Huang algorithm). The methods have been tested against classical methods, in particular the method of Aasen and methods using the QR factorization. The experiments have shown that some ABS methods are the most accurate, in both residual and solution error; moreover some ABS algorithms are cheaper in storage and in overhead, up to one order, especially for the case when m is close to n.

In many interior point methods the main computational cost is to compute the solution for a sequence of KT problems where only G, which is diagonal, changes. In such a case the ABS methods, which initially work on the matrix C, which is unchanged, are advantaged, particularly when m is large, where the dominant cubic term decreases with m and disappears for $m = n$, so that the overhead is dominated by second order terms. Again numerical experiments show that some ABS methods are more accurate than the classical ones. For details see [8].

Reformulation of the Simplex Method via the Implicit LX Algorithm

The implicit LX algorithm has a natural application to a reformulation of the simplex method for the LP prob-

lem in standard form, i. e. the problem

$$\begin{cases} \min & c^\top x \\ \text{s.t.} & Ax = b \\ & x \geq 0\,. \end{cases}$$

The applicability of the implicit LX method is a consequence of the fact that the iterate x_{i+1} generated by the method, started from the zero vector, is a basic type vector, with a unit component in the position k_i, non identically zero components corresponding to indices $j \in B_i$, where B_i is the set of indices of the unit vectors chosen as the z_i, w_i parameters, i. e. the set $B_i = (k_1, \ldots, k_i)$, while the components of x_{i+1} of indices in the set $N_i = N/B_i$ are identically zero, where $N = (1, \ldots n)$. Therefore, if the nonzero components are nonnegative, the point defines a vertex of the polytope containing the feasible points defined by the constraints of the LP problem.

In the simplex method one moves from a vertex to another one, according to some rules and usually reducing at each step the value of the function $c^\intercal x$. The direction along which one moves from a vertex to another one is an edge direction of the polytope and is determined by solving a linear system, whose coefficient matrix A_B, the *basic matrix*, is defined by m linearly independent columns of the matrix A, called the *basic columns*. Usually such a system is solved by the LU factorization method or occasionally by the *QR method*, see [5]. The new vertex is associated to a new basic matrix A_B', which is obtained by substituting one of the columns in A_B by a column of the matrix A_N, which comprises the columns of A that do not belong to A_B. The most efficient algorithm for solving the modified system, after the column interchange, is the *Forrest–Goldfarb method* [6], requiring m^2 multiplications. Notice that the classical simplex method requires m^2 storage for the matrix A_B plus mn storage for the matrix A, which must be kept in general to provide the columns for the exchange.

The application of the implicit LX method to the simplex method, developed in [4,10,13,17] exploits the fact that in the implicit LX algorithm the interchange of a jth column in A_B with a kth column in A_N corresponds to the interchange of a previously chosen parameter vector $z_j = w_j = e_j$ with a new parameter $z_k = w_k = e_k$. This operation is a special case of the perturbation of the Abaffian after a change in the parameters and can be done using a general formula of [15], without explicit use of the kth column in A_N. Moreover since all quantities which are needed for the construction of the search direction (the edge direction) and for the interchange criteria can as well be implemented without explicit use of the columns of A, it follows that the ABS approach needs only the storage of the matrix H_{m+1}, which, in the case of the implicit LX algorithm, has a cost of at most $n^2/4$. Therefore for values of m close to n the storage required by the ABS formulation is about 8 times less than for the classical simplex method.

Here we give the basic formulas of the simplex method in the classical and in the ABS formulation. The column in A_N substituting an old column in A_B is often taken as the column with minimal relative cost. In terms of the ABS formulation this is equivalent to minimize with respect to $i \in N_m$ the scalar $\eta_i = c^\intercal H^\intercal e_i$. Let N^* be the index chosen in this way. The column in A_B to be exchanged is usually chosen with the criterion of the maximum displacement along an edge which keeps the basic variables nonnegative. Define $\omega_i = x^\intercal e_i / e_i^\top H^\intercal e_{N^*}$, where x is the current basic feasible solution. Then the above criterion is equivalent to minimize ω_i with respect the set of indices $i \in B_m$ such that

$$e_i^\top H^\top e_{N^*} > 0. \tag{14}$$

Notice that $H^\intercal e_{N^*} \neq 0$ and that an index i such that (14) is satisfied always exists, unless x is a solution of the LP problem.

The update of the Abaffian after the interchange of the unit vectors, which corresponds to the update of the LU factors after the interchange of the basic with the nonbasic column, is given by the following formula

$$H' = H - (He_{B^*} - e_{B^*}) \frac{e_{N^*}^\top H}{e_{N^*}^\top H e_{B^*}}. \tag{15}$$

The search direction d, which in the classical formulation is obtained by solving the system $A_B d = -Ae_{N^*}$, is given by $d = H_{m+1}^\top e_{N^*}$, hence at no cost. Finally, the relative cost vector r, classically given by $r = c - A^\intercal A_B^{-1} c_B$, where c_B consists of the components of c with indices corresponding to those of the basic columns, is simply given by $r = H_{m+1} c$.

Let us now consider the computational cost of update (15). Since $H e_{B^*}$ has at most $n - m$ nonzero components, while $H^\intercal e_{N^*}$ has at most m, no more than $m(n - m)$ multiplications are required. The update is most expensive for $m = n/2$ and gets cheaper the smaller m is or the closer it is to n. In the dual steepest edge Forrest–Goldfarb method [6] the overhead for replacing a column is m^2, hence formula (15) is faster for $m > n/2$ and is recommended on overhead considerations for m sufficiently large. However we notice that ABS updates having a $O(m^2)$ cost can also be obtained by using the representation of the Abaffian in terms of $2m$ vectors. No computational experience has been obtained till now on the new ABS formulation of the simplex method.

Finally, a generalization of the *simplex method*, based upon the use of the Huang algorithm started with a suitable singular matrix, has been developed in [16]. In this formulation the solution is approached by points lying on a face of the polytope. Whenever the point hits a vertex the remaining iterates move among vertices and the method is reduced to the simplex method.

ABS Unification of Feasible Direction Methods for Minimization with Linear Constraints

ABS algorithms can be used to provide a unification of feasible point methods for nonlinear minimization with linear constraints, including as a special case the LP problem. Let us first consider the problem with only linear equality constraints:

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & f(x) \\ \text{s.t.} & Ax = b \\ & A \in \mathbb{R}^{m,n}, \quad m \le n, \\ & \operatorname{rank}(A) = m. \end{cases}$$

Let x_1 be a feasible starting point; then for an iteration procedure of the form $x_{i+1} = x_i - \alpha_i d_i$, the search direction will generate feasible points if and only if

$$Ad_i = 0. \tag{16}$$

Solving the underdetermined system (16) for d_i by the ABS algorithm, the solution can be written in the following form, taking, without loss of generality, the zero vector as a special solution

$$d_i = H_{m+1}^\top q, \tag{17}$$

where the matrix H_{m+1} depends on the arbitrary choice of the parameters H_1, w_i and v_i used in solving (16) and $q \in \mathbb{R}^n$ is arbitrary. Hence the general feasible direction iteration has the form

$$x_{i+1} = x_i - \alpha_i H_{m+1}^\top q. \tag{18}$$

The search direction is a descent direction if and only if $d^\intercal \nabla f(x) = q^\intercal H_{m+1} \nabla f(x) > 0$. Such a condition can always be satisfied by choice of q unless $H_{m+1} \nabla f(x) = 0$, which implies, from the null space structure of H_{m+1}, that $\nabla f(x) = A^\intercal \lambda$ for some λ, hence that x_{i+1} is a KT point and λ is the vector of the Lagrange multipliers. When x_{i+1} is not a KT point, it is immediate to see that the search direction is a descent directions if we select q as $q = WH_{m+1} \nabla f(x)$, where W is a symmetric and positive definite matrix.

Particular well-known algorithms from the literature are obtained by the following choices of q, with $W = I$:

- The *Wolfe reduced gradient method*. Here, H_{m+1} is constructed by the implicit LU (or the implicit LX) algorithm.
- The *Rosen gradient projection method*. Here, H_{m+1} is built using the Huang algorithm.
- The *Goldfarb–Idnani method*. Here, H_{m+1} is built via the modification of the Huang algorithm where H_1 is a symmetric positive definite matrix approximating the inverse Hessian of $f(x)$.

If there are inequalities two approaches are possible:

A) The *active set* approach. In this approach the set of linear equality constraints is modified at every iteration by adding and/or dropping some of the linear inequality constraints. Adding or deleting a single constraint can be done, for every ABS algorithm, in order two operations, see [15]. In the ABS reformulation of the Goldfarb–Idnani method, the initial matrix is related to a quasi-Newton approximation of the Hessian and an efficient update of the Abaffian after a change in the initial matrix is discussed in [14].

B) The *standard form* approach. In this approach, by introducing slack variables, the problem with both types of linear constraints is written in the equivalent form

$$\begin{cases} \min & f(x) \\ \text{s.t.} & Ax = b \\ & x \geq 0. \end{cases}$$

The following general iteration, started with x_1 a feasible point, generates a sequence of feasible points for the problem in standard form

$$x_{i+1} = x_i - \alpha_i \beta_i H_{m+1} \nabla f(x), \tag{19}$$

where the parameter α_i can be chosen by a line search along the vector $H_{m+1} \nabla f(x)$, while the relaxation parameter $\beta_i > 0$ is selected to avoid that the new point has some negative components.

If $f(x)$ is nonlinear, then H_{m+1} can be determined once and for all at the first step, since $\nabla f(x)$ generally changes from iteration to iteration, therefore modifying the search direction. If, however, $f(x) = c^\intercal x$ is linear (we have then the LP problem) to modify the search direction we need to change H_{m+1}. As observed before, the simplex method is obtained by constructing H_{m+1} with the implicit LX algorithm, every step of the method corresponding to a change of the parameters e_{k_i}. It can be shown, see [13], that the *method of Karmarkar* (equivalent to an earlier *method of Evtushenko* [3]), corresponds to using the generalized Huang algorithm, with initial matrix $H_1 = \text{Diag}(x_i)$ changing from iteration to iteration. Another method, faster than Karmarkar's and having superlinear against linear rate of convergence and $O(\sqrt{n})$ against $O(n)$ complexity, again first proposed by Y. Evtushenko, is obtained by the generalized Huang algorithm with initial matrix $H_1 = \text{Diag}(x_i^2)$.

See also

- ▶ ABS Algorithms for Linear Equations and Linear Least Squares
- ▶ Gauss–Newton Method: Least Squares, Relation to Newton's Method
- ▶ Generalized Total Least Squares
- ▶ Least Squares Orthogonal Polynomials
- ▶ Least Squares Problems
- ▶ Nonlinear Least Squares: Newton-type Methods
- ▶ Nonlinear Least Squares Problems
- ▶ Nonlinear Least Squares: Trust Region Methods

References

1. Abaffy J, Spedicato E (1989) ABS projection algorithms: Mathematical techniques for linear and nonlinear equations. Horwood, Westergate
2. Adachi N (1971) On variable metric algorithms. JOTA 7:391–409
3. Evtushenko Y (1974) Two numerical methods of solving nonlinear programming problems. Soviet Dokl Akad Nauk 251:420–423
4. Feng E, Wang XM, Wang XL (1997) On the application of the ABS algorithm to linear programming and linear complementarity. Optim Methods Softw 8:133–142
5. Fletcher R (1997) Dense factors of sparse matrices. In: Buhmann MD, Iserles A (eds) Approximation Theory and Optimization. Cambridge Univ. Press, Cambridge, pp 145–166
6. Forrest JH, Goldfarb D (1992) Steepest edge simplex algorithms for linear programming. Math Program 57:341–374
7. Psenichny BN, Danilin YM (1978) Numerical methods in extremal problems. MIR, Moscow
8. Spedicato E, Chen Z, Bodon E (1996) ABS methods for KT equations. In: Di Pillo G, Giannessi F (eds) Nonlinear Optimization and Applications. Plenum, New York, pp 345–359
9. Spedicato E, Xia Z (1992) Finding general solutions of the quasi-Newton equation in the ABS approach. Optim Methods Softw 1:273–281
10. Spedicato E, Xia Z, Zhang L (1995) Reformulation of the simplex algorithm via the ABS algorithm. Preprint Univ Bergamo
11. Spedicato E, Zhao J (1992) Explicit general solution of the quasi-Newton equation with sparsity and symmetry. Optim Methods Softw 2:311–319
12. Xia Z (1995) ABS generalization and formulation of the interior point method. Preprint Univ Bergamo
13. Xia Z (1995) ABS reformulation of some versions of the simplex method for linear programming. Report DMSIA Univ Bergamo 10/95
14. Xia Z, Liu Y, Zhang L (1992) Application of a representation of ABS updating matrices to linearly constrained optimization. Northeast Oper Res 7:1–9
15. Zhang L (1995) Updating of Abaffian matrices under perturbation in W and A. Report DMSIA Univ Bergamo 95/16
16. Zhang L (1997) On the ABS algorithm with singular initial matrix and its application to linear programming. Optim Methods Softw 8:143–156
17. Zhang L, Xia ZH (1995) Application of the implicit LX algorithm to the simplex method. Report DMSIA Univ Bergamo 9/95

Adaptive Convexification in Semi-Infinite Optimization

Oliver Stein
School of Economics and Business Engineering, University of Karlsruhe, Karlsruhe, Germany

MSC2000: 90C34, 90C33, 90C26, 65K05

Article Outline

Synonyms
Introduction
Feasibility in Semi-Infinite Optimization
Convex Lower Level Problems
The αBB Method
Formulation
αBB for the Lower Level
The MPCC Reformulation
Method
Refinement Step
The Algorithm
A Consistent Initial Approximation
A Certificate for Global Optimality
Conclusions
See also
References

Synonyms

ACA

Introduction

The adaptive convexification algorithm is a method to solve semi-infinite optimization problems via a sequence of *feasible iterates*. Its main idea [6] is to adaptively construct convex relaxations of the lower level problem, replace the relaxed lower level problems equivalently by their Karush–Kuhn–Tucker conditions, and solve the resulting mathematical programs with complementarity constraints. The convex relaxations are constructed with ideas from the αBB method of global optimization.

Feasibility in Semi-Infinite Optimization

In a (standard) semi-infinite optimization problem a finite-dimensional decision variable is subject to infinitely many inequality constraints. For adaptive convexification one assumes the form

$$SIP\colon \min_{x\in X} f(x) \quad \text{subject to} \quad g(x,y)\le 0, \quad \text{for all } y\in[0,1]$$

with objective function $f\in C^2(\mathbb{R}^n,\mathbb{R})$, constraint function $g\in C^2(\mathbb{R}^n\times\mathbb{R},\mathbb{R})$, a box constraint set $X=[x^\ell,x^u]\subset\mathbb{R}^n$ with $x^\ell<x^u\in\mathbb{R}^n$, and the set of infinitely many indices $Y=[0,1]$. Adaptive convexification easily generalizes to problems with additional inequality and equality constraints, a finite number of semi-infinite constraints as well as higher-dimensional box index sets [6]. Reviews on semi-infinite programming are given in [8,13], and [9,14,15] overview the existing numerical methods.

Classical numerical methods for *SIP* suffer from the drawback that their approximations of the feasible set $X\cap M$ with

$$M = \{x\in\mathbb{R}^n \mid g(x,y)\le 0 \text{ for all } y\in[0,1]\}$$

may contain infeasible points. In fact, discretization and exchange methods approximate M by finitely many inequalities corresponding to finitely many indices in $Y=[0,1]$, yielding an outer approximation of M, and reduction based methods solve the Karush–Kuhn–Tucker system of *SIP* by a Newton-SQP approach. As a consequence, the iterates of these methods are not necessarily feasible for *SIP*, but only their limit might be. On the other hand, a first method producing feasible iterates for *SIP* was presented in the articles [3,4], where a branch-and-bound framework for the global solution of *SIP* generates convergent sequences of lower and upper bounds for the globally optimal value.

In fact, checking feasibility of a given point $\bar{x}\in\mathbb{R}^n$ is the crucial problem in semi-infinite optimization. Clearly we have $\bar{x}\in M$ if and only if $\varphi(\bar{x})\le 0$ holds with the function

$$\varphi\colon \mathbb{R}^n\to\mathbb{R},\ x\mapsto \max_{y\in[0,1]} g(x,y)\,.$$

The latter function is the optimal value function of the so-called lower level problem of *SIP*,

$$Q(x)\colon \max_{y\in\mathbb{R}} g(x,y) \quad \text{subject to} \quad 0\le y\le 1\,.$$

The difficulty lies in the fact that $\varphi(\bar{x})$ is the *globally* optimal value of $Q(\bar{x})$ which might be hard to determine numerically. In fact, standard NLP solvers can

only be expected to produce a *local* maximizer y_{loc} of $Q(\bar{x})$ which is not necessarily a global maximizer y_{glob}. Even if $g(\bar{x}, y_{\text{loc}}) \le 0$ is satisfied, $\bar{x}$ might be infeasible since $g(\bar{x}, y_{\text{loc}}) \le 0 < \varphi(\bar{x}) = g(\bar{x}, y_{\text{glob}})$ may hold.

Convex Lower Level Problems

Assume for a moment that $Q(x)$ is a convex optimization problem for all $x \in X$, that is, $g(x, \cdot)$ is concave on $Y = [0, 1]$ for these x. An approach developed for so-called generalized semi-infinite programs from [18,19] then takes advantage of the fact that the solution set of a differentiable convex lower level problem satisfying a constraint qualification is characterized by its first order optimality condition. In fact, *SIP* and the Stackelberg game

$$SG\colon \min_{x,y} f(x) \quad \text{subject to} \quad g(x, y) \le 0, \quad \text{and } y \text{ solves } Q(x)$$

are equivalent problems, and the restriction 'y solves $Q(x)$' in *SG* can be equivalently replaced by its Karush–Kuhn–Tucker condition. For this reformulation we use that the Lagrange function of $Q(x)$,

$$\mathcal{L}(x, y, \gamma_\ell, \gamma_u) \;=\; g(x, y) + \gamma_\ell\, y + \gamma_u(1 - y),$$

satisfies

$$\nabla_y \mathcal{L}(x, y, \gamma_\ell, \gamma_u) \;=\; \nabla_y g(x, y) + \gamma_\ell - \gamma_u$$

and obtain that the Stackelberg game is equivalent to the following mathematical program with complementarity constraints:

$$\begin{aligned} MPCC\colon \min_{x,y,\gamma_\ell,\gamma_u} \; & f(x) \text{ subject to } g(x, y) \le 0 \\ & \nabla_y g(x, y) + \gamma_\ell - \gamma_u = 0 \\ & \gamma_\ell\, y = 0 \\ & \gamma_u(1 - y) = 0 \\ & \gamma_\ell,\ \gamma_u \ge 0 \\ & y,\ 1 - y \ge 0\,. \end{aligned}$$

Overviews of solution methods for *MPCC* are given in [10,11,17]. One approach to solve *MPCC* is the reformulation of the complementarity constraints by a so-called NCP function, that is, a function $\phi\colon \mathbb{R}^2 \to \mathbb{R}$ with

$$\phi(a, b) \;=\; 0 \quad \text{if and only if} \quad a \ge 0\,, \quad b \ge 0, \quad a\, b = 0\,.$$

For numerical purposes one can regularize these nondifferentiable NCP functions. Although MPCC does not necessarily have to be solved via the NCP function formulation, in the following we will use NCP functions to keep the notation concise. In fact, *MPCC* can be equivalently rewritten as the nonsmooth problem

$$\begin{aligned} P\colon \min_{x,y,\gamma_\ell,\gamma_u} \; & f(x) \quad \text{subject to} \quad g(x, y) \le 0 \\ & \nabla_y g(x, y) + \gamma_\ell - \gamma_u = 0 \\ & \phi(\gamma_\ell, y) = 0 \\ & \phi(\gamma_u, 1 - y) = 0\,. \end{aligned}$$

The αBB Method

In αBB, a convex underestimator of a nonconvex function is constructed by decomposing it into a sum of nonconvex terms of special type (e. g., linear, bilinear, trilinear, fractional, fractional trilinear, convex, univariate concave) and nonconvex terms of arbitrary type. The first type is then replaced by its convex envelope or very tight convex underestimators which are already known. A complete list of the tight convex underestimators of the above special type nonconvex terms is provided in [5].

For the ease of presentation, here we will treat all terms as arbitrarily nonconvex. For these terms, αBB constructs convex underestimators by adding a quadratic relaxation function ψ. With the obvious modification we use this approach to construct a concave overestimator for a nonconcave function $g\colon [y^\ell, y^u] \to \mathbb{R}$ being C^2 on an open neighborhood of $[y^\ell, y^u]$. With

$$\psi(y; \alpha, y^\ell, y^u) \;=\; \frac{\alpha}{2}(y - y^\ell)(y^u - y) \tag{1}$$

we put

$$\tilde{g}(y; \alpha, y^\ell, y^u) \;=\; g(y) + \psi(y; \alpha, y^\ell, y^u)\,.$$

In the sequel we will suppress the dependence of $\tilde{g}$ on y^ℓ, y^u. For $\alpha \ge 0$ the function $\tilde{g}$ clearly is an overestimator of g on $[y^\ell, y^u]$, and it coincides with g at the endpoints y^ℓ, y^u of the domain. Moreover, $\tilde{g}$ is twice continuously differentiable with second derivative

$$\nabla_y^2 \tilde{g}(y; \alpha) \;=\; \nabla^2 g(y) - \alpha$$

on $[y^\ell, y^u]$. Consequently $\tilde{g}$ is concave on $[y^\ell, y^u]$ for

$$\alpha \geq \max_{y\in[y^\ell,y^u]} \nabla^2 g(y) \tag{2}$$

(cf. also [1,2]). The computation of α thus involves a global optimization problem itself. Note, however, that one may use *any* upper bound for the right-hand side in (2). Such upper bounds can be provided by interval methods (see, e. g., [5,7,12]). An α satisfying (2) is called *convexification parameter*.

Combining these facts shows that for

$$\alpha \geq \max\left(0, \max_{y\in[y^\ell,y^u]} \nabla^2 g(y)\right)$$

the function $\tilde{g}(y;\alpha)$ is a concave overestimator of g on $[y^\ell, y^u]$.

Formulation

For $N \in \mathcal{N}$ let $0 = \eta^0 < \eta^1 < \ldots < \eta^{N-1} < \eta^N = 1$ define a subdivision of $Y = [0,1]$, that is, with $K = \{1,\ldots,N\}$ and

$$Y^k = [\eta^{k-1}, \eta^k], \quad k \in K,$$

we have

$$Y = \bigcup_{k\in K} Y_k .$$

A trivial but very useful observation is that the single semi-infinite constraint

$$g(x,y) \leq 0 \quad \text{for all} \quad y \in Y$$

is equivalent to the finitely many semi-infinite constraints

$$g(x,y) \leq 0 \quad \text{for all} \quad y \in Y^k,\ k \in K .$$

Given a subdivision, one can construct concave overestimators for each of these finitely many semi-infinite constraints, solve the corresponding optimization problem, and adaptively refine the subdivision.

The following lemma formulates the obvious fact that replacing g by overestimators on each subdivision node Y^k results in an approximation of M by *feasible* points.

Lemma 1 *For each $k \in K$ let $g^k\colon X \times Y^k \to \mathbb{R}$, and let $\bar{x} \in X$ be given such that for all $k \in K$ and all $y \in Y^k$ we have $g(\bar{x},y) \leq g^k(\bar{x},y)$. Then the constraints*

$$g^k(\bar{x},y) \leq 0 \quad \textit{for all} \quad y \in Y^k,\ k \in K,$$

entail $\bar{x} \in M$.

αBB for the Lower Level

For the construction of these overestimators one uses ideas of the αBB method. In fact, for each $k \in K$ we put

$$g^k\colon X\times Y^k \to \mathbb{R}, (x,y) \mapsto g(x,y)+\psi(y;\alpha_k,\eta^{k-1},\eta^k) \tag{3}$$

with the quadratic relaxation function ψ from (1) and

$$\alpha_k > \max\left(0, \max_{(x,y)\in X\times Y^k} \nabla_y^2 g(x,y)\right). \tag{4}$$

Note that the latter condition on α_k is uniform in x. We emphasize that with the single bound

$$\bar{\alpha} > \max\left(0, \max_{(x,y)\in X\times Y} \nabla_y^2 g(x,y)\right) \tag{5}$$

the choices $\alpha_k := \bar{\alpha}$ satisfy (4) for all $k \in K$. Moreover, the α_k can always be chosen such that $\alpha_k \leq \bar{\alpha}$, $k \in K$.

The following properties of g^k are easily verified.

Lemma 2 ([6]) *For each $k \in K$ let g^k be given by (3). Then the following holds:*
(i) For all $(x,y) \in X \times Y^k$ we have $g(x,y) \leq g^k(x,y)$.
(ii) For all $x \in X$, the function $g^k(x,\cdot)$ is concave on Y^k.

Now consider the following approximation of the feasible set M, where $E = \{\eta^k \mid k \in K\}$ denotes the set of subdivision points, and α the vector of convexification parameters:

$$M_{\alpha BB}(E,\alpha) = \{\, x \in \mathbb{R}^n \mid g^k(x,y) \leq 0\,, \text{ for all } y \in Y^k,\ k \in K\,\}.$$

By Lemma 1 and Lemma 2(i) we have $M_{\alpha BB}(E,\alpha) \subset M$. This means that any solution concept for

$$SIP_{\alpha BB}(E,\alpha)\colon \min_{x\in X} f(x) \quad \text{subject to} \quad x \in M_{\alpha BB}(E,\alpha),$$

be it global solutions, local solutions or stationary points, will at least lead to feasible points of *SIP* (provided that $SIP_{\alpha BB}(E,\alpha)$ is consistent).

The problem $SIP_{\alpha BB}(E,\alpha)$ has finitely many lower level problems $Q^k(x)$, $k \in K$, with

$$Q^k(x)\colon \max_{y\in\mathbb{R}} g^k(x,y) \quad \text{subject to} \quad \eta^{k-1} \le y \le \eta^k\,.$$

Since the inequality (4) is strict, the convex problem $Q^k(x)$ has a unique solution $y^k(x)$ for each $k \in K$ and $x \in X$. Recall that $y \in Y^k$ is called active for the constraint $\max_{y\in Y^k} g^k(x,y) \le 0$ at $\bar{x}$ if $g^k(\bar{x},y) = 0$ holds. By the uniqueness of the global solution of $Q^k(\bar{x})$ there exists *at most one* active index for each $k \in K$, namely $y^k(\bar{x})$. Thus, one can consider the finite active index sets

$$\begin{aligned} K_0(\bar{x}) &= \{\, k \in K \mid g^k(\bar{x}, y^k(\bar{x})) = 0 \,\},\\ Y_0^{\alpha BB}(\bar{x}) &= \{\, y^k(\bar{x}) \mid k \in K_0(\bar{x}) \,\}\,. \end{aligned}$$

The MPCC Reformulation

Following the ideas to treat convex lower level problems, y^k solves $Q^k(x)$ if and only if $(x, y^k, \gamma_\ell^k, \gamma_u^k)$ solves the system

$$\begin{aligned} \nabla_y g^k(x,y) + \gamma_\ell - \gamma_u &= 0\\ \phi(\gamma_\ell, y - \eta^{k-1}) &= 0\\ \phi(\gamma_u, \eta^k - y) &= 0 \end{aligned}$$

with some γ_ℓ^k, γ_u^k, and ϕ denoting some NCP function. With

$$\begin{aligned} w &:= (x, y^k, \gamma_\ell^k, \gamma_u^k, k \in K)\\ F(w) &:= f(x)\\ G^k(w;E,\alpha) &:= g(x,y^k) + \frac{\alpha_k}{2}(y^k - \eta^{k-1})(\eta^k - y^k)\\ H^k(w;E,\alpha) &:= \begin{pmatrix} \nabla_y g(x,y^k) + \alpha_k\left(\frac{\eta^{k-1}+\eta^k}{2} - y^k\right) + \gamma_\ell^k - \gamma_u^k\\ \phi(\gamma_\ell^k, y^k - \eta^{k-1})\\ \phi(\gamma_u^k, \eta^k - y^k) \end{pmatrix} \end{aligned}$$

one can thus replace $SIP_{\alpha BB}(E,\alpha)$ equivalently by the nonsmooth problem

$$\begin{aligned} P(E,\alpha)\colon \min_w\ & F(w) \quad \text{subject to}\\ & G^k(w;E,\alpha) \le 0,\\ & H^k(w;E,\alpha) = 0, \quad k \in K\,. \end{aligned}$$

The latter problem can be solved to local optimality by MPCC algorithms [10,11,17]. For a local minimizer $\bar{w}$ of $P(E,\alpha)$ the subvector $\bar{x}$ of $\bar{w}$ is a local minimizer and, hence, a stationary point of $SIP_{\alpha BB}(E,\alpha)$.

Method

The main idea of the adaptive convexification algorithm is to compute a stationary point $\bar{x}$ of $SIP_{\alpha BB}(E,\alpha)$ by the approach from the previous section, and terminate if $\bar{x}$ is also stationary for *SIP* within given tolerances. If $\bar{x}$ is not stationary it refines the subdivision E in the spirit of exchange methods [8,15] by adding the active indices $Y_0^{\alpha BB}(\bar{x})$ to E, and constructs a refined problem $SIP_{\alpha BB}(E \cup Y_0^{\alpha BB}(\bar{x}), \tilde{\alpha})$ by the following procedure. Note that, in view of Carathéodory's theorem, the number of elements of $Y_0^{\alpha BB}(\bar{x})$ may be bounded by $n+1$.

Refinement Step

For any $\tilde{\eta} \in Y_0^{\alpha BB}(\bar{x})$, let $k \in K$ be the index with $\tilde{\eta} \in [\eta^{k-1}, \eta^k]$. Put $Y^{k,1} = [\eta^{k-1}, \tilde{\eta}]$, $Y^{k,2} = [\tilde{\eta}, \eta^k]$, let $\alpha_{k,1}$ and $\alpha_{k,2}$ be the corresponding convexification parameters, put

$$\begin{aligned} g^{k,1}(x,y) &= g(x,y) + \frac{\alpha_{k,1}}{2}(y - \eta^{k-1})(\tilde{\eta} - y),\\ g^{k,2}(x,y) &= g(x,y) + \frac{\alpha_{k,2}}{2}(y - \tilde{\eta})(\eta^k - y), \end{aligned}$$

and define $M_{\alpha BB}(E \cup \{\tilde{\eta}\}, \tilde{\alpha})$ by replacing the constraint

$$g^k(x,y) \le 0, \quad \text{for all } y \in Y^k$$

in $M_{\alpha BB}(E,\alpha)$ by the two new constraints

$$g^{k,i}(x,y) \le 0, \quad \text{for all} \quad y \in Y^{k,i}, \quad i = 1,2\,,$$

and by replacing the entry α_k of α by the two new entries $\alpha_{k,i}$, $i = 1,2$.

The Algorithm

The point $\bar{x}$ is stationary for $SIP_{\alpha BB}(E,\alpha)$ (in the sense of Fritz John) if $\bar{x} \in M_{\alpha BB}(E,\alpha)$ and if there exist $y^k \in Y_0^{\alpha BB}(\bar{x})$, $1 \le k \le n+1$, and $(\kappa,\lambda) \in S^{n+1}$ (the $(n+1)-$dimensional standard simplex) with

$$\begin{aligned} \kappa \nabla f(\bar{x}) + \sum_{k=1}^{n+1} \lambda_k \nabla_x g(\bar{x}, y^k) &= 0\\ \lambda_k \cdot g^k(\bar{x}, y^k) &= 0,\ 1 \le k \le n+1\,. \end{aligned}$$

For the adaptive convexification algorithm the notions of *active index*, *stationarity*, and *set unification* are relaxed by certain tolerances.

Definition 1 For $\varepsilon_{\text{act}}, \varepsilon_{\text{stat}}, \varepsilon_{\cup} > 0$ we say that

(i) y^k is ε_{act}-active for g^k at $\bar{x}$ if $g^k(\bar{x}, y^k) \in [-\varepsilon_{\text{act}}, 0]$,

(ii) $\bar{x}$ is $\varepsilon_{\text{stat}}$-stationary for *SIP* with ε_{act}-active indices if $\bar{x} \in M$ and if there exist $y^k \in Y$, $1 \le k \le n+1$, and $(\kappa, \lambda) \in S^{n+1}$ such that

$$\left\| \kappa \nabla f(\bar{x}) + \sum_{k=1}^{n+1} \lambda_k \nabla_x g(\bar{x}, y^k) \right\| \le \varepsilon_{\text{stat}}$$
$$\lambda_k \cdot g(\bar{x}, y^k) \in [-\lambda_k \cdot \varepsilon_{\text{act}}, 0], \quad 1 \le k \le n+1,$$

hold, and

(iii) the $\varepsilon_{\cup}$-union of E and $\tilde{\eta}$ is $E \cup \{\tilde{\eta}\}$ if

$$\min\{\tilde{\eta} - \eta^{k-1}, \eta^k - \tilde{\eta}\} > \varepsilon_{\cup} \cdot (\eta^k - \eta^{k-1})$$

holds for the $k \in K$ with $\tilde{\eta} \in [\eta^{k-1}, \eta^k]$, and E otherwise (i. e., $\tilde{\eta}$ is not unified with E if its distance from E is too small).

In [6] it is shown that Algorithm 1 is well-defined, convergent and finitely terminating. Furthermore, the following feasibility result holds.

Theorem 2 ([6]) *Let $(x^\nu)_\nu$ be a sequence of points generated by Algorithm 1. Then all x^ν, $\nu \in \mathbb{N}$, are feasible for SIP, the sequence $(x^\nu)_\nu$ has an accumulation point, each such accumulation point x^* is feasible for SIP, and $f(x^*)$ provides an upper bound for the optimal value of SIP.*

Numerical examples for the performance of the method from Chebyshev approximation and design centering are given in [6].

A Consistent Initial Approximation

Even if the feasible set M of *SIP* is consistent, there is no guarantee that its approximations $M_{\alpha BB}(E, \alpha)$ are also consistent. For Step 1 of Algorithm 1 [6] suggests the following phase I approach: use Algorithm 1 to construct adaptive convexifications of

$$SIP^{ph.I}: \min_{(x,z) \in X \times \mathbb{R}} z \quad \text{subject to} \quad g(x, y) \le z \quad \text{for all } y \in [0, 1]$$

Algorithm 1
(Adaptive convexification algorithm)

Step 1: Determine a uniform convexification parameter $\bar{\alpha}$ with (5), choose $N \in \mathbb{N}$, $\eta^k \in Y$ and $\alpha_k \le \bar{\alpha}$, $k \in K = \{1, \ldots, N\}$, such that $SIP_{\alpha BB}(E, \alpha)$ is consistent, as well as tolerances $\varepsilon_{\text{act}}, \varepsilon_{\text{stat}}, \varepsilon_{\cup} > 0$ with $\varepsilon_{\cup} \le 2\varepsilon_{\text{act}}/\bar{\alpha}$.

Step 2: By solving $P(E, \alpha)$, compute a stationary point x of $SIP_{\alpha BB}(E, \alpha)$ with ε_{act}−active indices y^k, $1 \le k \le n+1$, and multipliers (κ, λ).

Step 3: Terminate if x is $\varepsilon_{\text{stat}}$−stationary for *SIP* with $(2\varepsilon_{\text{act}})$-active indices y^k, $1 \le k \le n+1$, from Step 2 and multipliers (κ, λ) from Step 2. Otherwise construct a new set $\tilde{E}$ of subdivision points as the $\varepsilon_{\cup}$-union of E and $\{y^k | 1 \le k \le n+1\}$, and perform a refinement step for the elements in $\tilde{E} \setminus E$ to construct a new feasible set $M_{\alpha BB}(\tilde{E}, \tilde{\alpha})$.

Step 4: Put $E = \tilde{E}$, $\alpha = \tilde{\alpha}$, and go to Step 2.

Adaptive Convexification in Semi-Infinite Optimization, Algorithm 1

until a feasible point $(\bar{x}, \bar{z})$ with $\bar{z} \le 0$ of $SIP^{ph.I}_{\alpha BB}(E, \alpha)$ is found with some subdivision E and convexification parameters α. The point $\bar{x}$ is then obviously also feasible for $SIP_{\alpha BB}(E, \alpha)$ and can be used as an initial point to solve the latter problem. Due to the possible nonconvexity of the upper level problem of *SIP*, this phase I approach is not necessarily successful, but possible remedies for this situation are given in [6].

To initialize Algorithm 1 for phase I, select some point $\bar{x}$ in the box X and put $E^1 = \{0, 1\}$, that is, $Y^1 = Y = [0, 1]$. Compute α_1 according to (4) and solve the convex optimization problem $Q^1(\bar{x})$ with standard software. With its optimal value $\bar{z}$, the point $(\bar{x}, \bar{z})$ is feasible for $SIP^{ph.I}_{\alpha BB}(E^1, \alpha_1)$.

A Certificate for Global Optimality

After termination of Algorithm 1 one can exploit that the set $E \subset [0, 1]$ contains indices that should also yield a good outer approximation of M. The optimal value of the problem

$$P_{\text{outer}}: \min_{x \in X} f(x) \quad \text{subject to} \quad g(x, \eta) \le 0, \ \eta \in E,$$

yields a rigorous *lower* bound for the optimal value of *SIP*. If P_{outer} can actually be solved to global optimality (e. g., if a standard NLP solver is used, due to convexity with respect to x), then a comparison of this lower bound for the optimal value of *SIP* with the upper bound from Algorithm 1 can yield a certificate of global optimality for *SIP* up to some tolerance.

Conclusions

The adaptive convexification algorithm provides an easily implementable way to solve semi-infinite optimization problems with feasible iterates. To explain its basic ideas, in [6] the algorithm is presented in its simplest form. It can be improved in a number of ways, for example in the magnitude of the convexification parameters and in their adaptive refinement, or by using other convexification techniques. Although the numerical results from [6] are very promising, further work is needed on error estimates on the numerical solution of the auxiliary problem $P(E, \alpha)$, which is assumed to be solved to exact local optimality by the present adaptive convexification algorithm.

See also

- ► αBB Algorithm
- ► Bilevel Optimization: Feasibility Test and Flexibility Index
- ► Convex Discrete Optimization
- ► Generalized Semi-infinite Programming: Optimality Conditions

References

1. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB, for general twice-differentiable constrained NLPs – I: theoretical advances. Comput Chem Eng 22:1137–1158
2. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB, for general twice-differentiable constrained NLPs – II: implementation and computational results. Comput Chem Eng 22:1159–1179
3. Bhattacharjee B, Green WH Jr, Barton PI (2005) Interval methods for semi-infinite programs. Comput Optim Appl 30:63–93
4. Bhattacharjee B, Lemonidis P, Green WH Jr, Barton PI (2005) Global solution of semi-infinite programs. Math Program 103:283–307
5. Floudas CA (2000) Deterministic global optimization, theory, methods and applications. Kluwer, Dordrecht
6. Floudas CA, Stein O (2007) The adaptive convexification algorithm: a feasible point method for semi-infinite programming. SIAM J Optim 18:1187–1208
7. Hansen E (1992) Global optimization using interval analysis. Dekker, New York
8. Hettich R, Kortanek KO (1993) Semi-infinite programming: theory, methods, and applications. SIAM Rev 35:380–429
9. Hettich R, Zencke P (1982) Numerische Methoden der Approximation und semi-infiniten Optimierung. Teubner, Stuttgart
10. Kočvara M, Outrata J, Zowe J (1998) Nonsmooth approach to optimization problems with equilibrium constraints: theory, applications and numerical results. Kluwer, Dordrecht
11. Luo Z, Pang J, Ralph D (1996) Mathematical programs with equilibrium constraints. Cambridge University Press, Cambridge
12. Neumaier A (1990) Interval methods for systems of equations. Cambridge University Press, Cambridge
13. Polak E (1987) On the mathematical foundation of nondifferentiable optimization in engineering design. SIAM Rev 29:21–89
14. Polak E (1997) Optimization, algorithms and consistent approximations. Springer, Berlin
15. Reemtsen R, Görner S (1998) Numerical methods for semi-infinite programming: a survey. In: Reemtsen R, Rückmann J-J (eds) Semi-infinite programming. Kluwer, Boston, pp 195–275
16. Reemtsen R, Rückmann J-J (eds) (1998) Semi-infinite programming. Kluwer, Boston
17. Scholtes S, Stöhr M (1999) Exact penalization of mathematical programs with equilibrium constraints. SIAM J Control Optim 37:617–652
18. Stein O (2003) Bi-level strategies in semi-infinite programming. Kluwer, Boston
19. Stein O, Still G (2003) Solving semi-infinite optimization problems with interior point techniques. SIAM J Control Optim 42:769–788

Adaptive Global Search

J. M. Calvin
Department Computer and Information Sci.,
New Jersey Institute Techn., Newark, USA

MSC2000: 60J65, 68Q25

Article Outline

Keywords
See also
References

Keywords

Average case complexity; Adaptive algorithm; Wiener process; Randomized algorithms

This article contains a survey of some well known facts about the complexity of *global optimization*, and also describes some results concerning the *average-case complexity*.

Consider the following optimization problem. Given a class F of objective functions f defined on a compact subset of d-dimensional Euclidean space, the goal is to approximate the global minimum of f based on evaluation of the function at sequentially selected points. The focus will be on the error after n observations

$$\Delta_n = \Delta_n(f) = f_n - f^*,$$

where f_n is the smallest of the first n observed function values (other approximations besides f_n are often considered).

Complexity of optimization is usually studied in the worst- or average-case setting. In order for a *worst-case analysis* to be useful the class of objective functions F must be quite restricted. Consider the case where F is a subset of the continuous functions on a compact set. It is convenient to consider the class $F = C^r([0, 1]^d)$ of real-valued functions on $[0, 1]^d$ with continuous derivatives up to order $r \geq 0$. Suppose that $r > 0$ and f^r is bounded. In this case $\Theta(\epsilon^{-d/r})$ function evaluations are needed to ensure that the error is at most ϵ for any $f \in F$; see [8].

An *adaptive algorithm* is one for which the $(n + 1)$st observation point is determined as a function of the previous observations, while a nonadaptive algorithm chooses each point independently of the function values. In the worst-case setting, adaptation does not help much under quite general assumptions. If F is convex and symmetric (in the sense that $-F = F$), then the maximum error under an adaptive algorithm with n observations is not smaller than the maximum error of a nonadaptive method with $n + 1$ observations; see [4].

Virtually all global optimization methods in practical use are adaptive. For a survey of such methods see [6,9]. The fact that the worst-case performance can not be significantly improved with adaptation leads to consideration of alternative settings that may be more appropriate. One such setting is the average-case setting, in which a probability measure P on F is chosen. The object of study is then the sequence of random variables $\Delta_n(f)$, and the questions include under what conditions (for what algorithms) the error converges to zero and for convergent algorithms the speed of convergence. While the average-case error is often defined as the mathematical expectation of the error, it is useful to take a broader view, and consider for example convergence in probability of $a_n\Delta_n$ for some normalizing sequence $\{a_n\}$.

With the average-case setting one can consider less restricted classes F than in the worst-case setting. As F gets larger, the worst-case deviates more and more from the average case, but may occur on only a small portion of the set F. Even for continuous functions the worst-case is arbitrarily bad.

Most of what is known about the average-case complexity of optimization is in the one-dimensional setting under the *Wiener probability measure* on $C([0, 1])$. Under the Wiener measure, the increments $f(t)-f(s)$ have a normal distribution with mean zero and variance $t-s$, and are independent for disjoint intervals. Almost every f is nowhere differentiable, and the set of local minima is dense in the unit interval. One can thus think of the Wiener measure as corresponding to assuming 'only' continuity; i. e., a worst-case probabilistic assumption.

K. Ritter proved [5] that the best nonadaptive algorithms have error of order $n^{-1/2}$ after n function evaluations; the optimal order is achieved by observing at equally spaced points. Since the choice of each new observation point does not depend on any of the previous observations, the computation can be carried out in parallel. Thus under the Wiener measure, the optimal nonadaptive order of convergence can be accomplished with an algorithm that has computational cost that grows linearly with the number of observations and uses constant storage. This gives the base on which to compare adaptive algorithms.

Recent studies (as of 2000) have formally established the improved power of adaptive methods in the average-case setting by analyzing the convergence rates of certain adaptive algorithms. A *randomized algorithm* is described in [1] with the property that for any $0 < \delta < 1$, a version can be constructed so that under the Wiener measure, the error converges to zero at rate $n^{-1+\delta}$. This

algorithm maintains a memory of two past observation values, and the computational cost grows linearly with the number of iterations. Therefore, the convergence rate of this adaptive algorithm improves from the non-adaptive $n^{-1/2}$ rate to $n^{-1+\delta}$ with only a constant increase in storage.

Algorithms based on a random model for the objective function are well-suited to average-case analysis. H. Kushner proposed [3] a global optimization method based on modeling the objective function as a Wiener process. Let $\{z_n\}$ be a sequence of positive numbers, and let the $(n+1)$st point be chosen to maximize the probability that the new function value is less than the previously observed minimum minus z_n. This class of algorithms, often called *P-algorithms*, was given a formal justification by A. Žilinskas [7].

By allowing the $\{z_n\}$ to depend on the past observations instead of being a fixed deterministic sequence, it is possible to establish a much better convergence rate than that of the randomized algorithm described above. In [2] an algorithm was constructed with the property that the error converges to zero for any continuous function and furthermore, the error is of order e^{-nc_n}, where $\{c_n\}$ (a parameter of the algorithm) is a deterministic sequence that can be chosen to approach zero at an arbitrarily slow rate. Notice that the convergence rate is now almost exponential in the number of observations n. The computational cost of the algorithm grows quadratically, and the storage increases linearly, since all past observations must be stored.

See also

- ► Adaptive Simulated Annealing and its Application to Protein Folding
- ► Global Optimization Based on Statistical Models

References

1. Calvin J (1997) Average performance of a class of adaptive algorithms for global optimization. Ann Appl Probab 7:711–730
2. Calvin J (2001) A one-dimensional optimization algorithm and its convergence rate under the Wiener measure. J Complexity
3. Kushner H (1962) A versatile stochastic model of a function of unknown and time varying form. J Math Anal Appl 5: 150–167
4. Novak E (1988) Deterministic and stochastic error bounds in numerical analysis. Lecture Notes in Mathematics, vol 1349. Springer, Berlin
5. Ritter K (1990) Approximation and optimization on the Wiener space. J Complexity 6:337–364
6. Törn A, Žilinskas A (1989) Global optimization. Springer, Berlin
7. Žilinskas A (1985) Axiomatic characterization of global optimization algorithm and investigation of its search strategy. OR Lett 4:35–39
8. Wasilkowski G (1992) On average complexity of global optimization problems. Math Program 57:313–324
9. Zhigljavsky A (1991) Theory of global random search. Kluwer, Dordrecht

Adaptive Simulated Annealing and its Application to Protein Folding

ASA

RUTH PACHTER, ZHIQIANG WANG
Air Force Research Laboratory Materials & Manufacturing Directorate, Wright–Patterson AFB, Wright–Patterson AFB, USA

MSC2000: 92C05

Article Outline

Keywords
The ASA Method
  Monte-Carlo Configurations
  Annealing Schedule
  Re-Annealing
Application to Protein Folding
  Computational Details
  Met-Enkephalin
  Poly(L-Alanine)
Conclusion
Recent Studies and Future Directions
See also
References

Keywords

Optimization; Adaptive simulated annealing; Protein folding; Met-Enkephalin; Poly(L-Alanine)

The adaptive simulated annealing (ASA) algorithm [3] has been shown to be faster and more efficient than

simulated annealing and genetic algorithms [4]. In this article we first outline some of the aspects of the method and specific computational details, and then review the application of the ASA method to biomolecular structure determination [15], specifically for Met-Enkephalin and a model of the poly(L-Alanine) system.

The ASA Method

For a system described by a cost function $E(\{p^i\})$, where all p^i $(i = 1, \ldots, D)$ are parameters (variables) having ranges $[A_i, B_i]$, the ASA procedure to find the global optimum of 'E' contains the following elements.

Monte-Carlo Configurations

As the kth point is saved in a D-dimensional configuration space, the new point p^i_{k+1} is generated by:

$$p^i_{k+1} = p^i_k + y^i(B_i - A_i), \tag{1}$$

where the random variables y^i in $[-1, 1]$ (non-uniform) are generated from a random number u^i uniformly distributed in $[0, 1]$, and the temperature T_i associated with parameter p^i, as follows:

$$y^i = \mathrm{sgn}(u^i - 0.5)T_i\left[\left(1 + \frac{1}{T_i}\right)^{|2u^i - 1|} - 1\right]. \tag{2}$$

Note that if p^i_{k+1} is outside the range of $[A_i, B_i]$ it will be disregarded, with the process being repeated until it falls within the range. The choice of y^i is made so that the probability density distribution of the D parameters will satisfy the distribution of each parameter:

$$g^i(y^i; T_i) = \frac{1}{2(|y^i| + T_i)(1 + \frac{1}{T_i})}, \tag{3}$$

which is chosen to ensure that any point in configuration space can be sampled infinitely often in annealing time with a cooling schedule outlined below. Thus, at any annealing time k_0, the probability of not generating a global optimum, given infinite time, is zero:

$$\prod_{k=k_0}^{\infty}(1 - g_k) = 0, \tag{4}$$

where g_k is the distribution function at time step k. Note that all atoms move at each Monte-Carlo step in ASA. A Boltzmann acceptance criterion is then applied to the difference in the cost function.

Annealing Schedule

The annealing schedule for each parameter temperature from a starting temperature T_{0i}, and similarly for the cost temperature, is given by:

$$T_i(k_i) = T_{0i}\exp\left(-c_i k_i^{\frac{1}{D}}\right), \tag{5}$$

where c_i and k_i are the annealing scale and ASA step of parameter p^i. The index for re-annealing the cost function is determined by the number of accepted points instead of the number of generated points as is being used for the parameters. This choice was made since the Boltzmann acceptance criterion uses an exponential distribution which is not as 'fat-tailed' as the ASA distribution used for the parameters.

Re-Annealing

The temperatures may be periodically re-annealed or re-scaled according to the sensitivity of the cost function. At any given annealing time, the temperature range is 'stretched out' over the relatively insensitive parameters, thus guiding the search 'fairly' among the parameters. The sensitivity of the energy to each parameter is calculated by:

$$S_i = \frac{\partial E}{\partial p^i}, \tag{6}$$

while the re-annealing temperature is determined by:

$$T_i(k') = T_i(k)\frac{S_i}{S_{\max}}. \tag{7}$$

In this way, less sensitive parameters anneal faster. This is done approximately every 100 accepted events.

For comparison, within conventional simulated annealing [6] the cooling schedule is given by:

$$ST_k = T_0 e^{-(1-c)k} \quad (0 < c < 1), \tag{8}$$

where trial and error are applied to determine the annealing rate c—1 as well as the starting temperature T_0. A Monte-Carlo simulation is carried out at each temperature step k with temperature T_k. This cooling schedule is equivalent to $T_{k+1} = T_k\, c$.

The ASA algorithm is mostly suited to problems for which less is known about the system, and has proven to be more robust than other simulated annealing techniques for complex problems with multiple local minima, e. g., as compared to Cauchy annealing where T_i

$= T_0/k$, and Boltzmann annealing where $T_i = T_0/\ln k$. The annealing schedule in (8), faster than ASA for a large dimension of D, does not pass the infinitely often annealing-time test in (4), and is therefore referred to as *simulated quench* in the terminology of ASA.

Application to Protein Folding

Computational Details

A protein can be defined as a biopolymer of hundreds of amino acids bonded by peptide bonds, while the test models in this article contain less amino acids, namely oligopeptides. The Met-Enkephalin model was constructed as (H-Tyr-Gly-Gly-Phe-Met-OH). For 14(L-Alanine), the neutral —NH_2 and —COOH end groups were substituted at the termini. The conformation of a protein is described by the dihedral angles of the backbone (ϕ_i, ψ_i), side-chains (χ_i^j), and peptide bond (ω_i, often very close to 180°). Therefore, the conformation determination of the most stable protein is to find the set of $\{\phi, \psi, \chi, \omega\}$ which give the global minimal potential energy $E(\phi, \psi, \chi, \omega)$. Within the ASA nomenclature, the 'cost function' is the potential energy, while a 'parameter' is a dihedral angle variable.

Conformational analyses using conventional simulated annealing were carried out previously [9,11]. The modifications in these works include moving a number of dihedral angles in a Monte-Carlo step; adjusting the maximum deviation of the variables as the temperature decreases to insure that the acceptance ratio is more than 25%; and treating the variables differently according to their importance in the folding process, e. g., by increasing sampling for the backbone dihedral angles as compared to those of the side-chains. It is interesting to point out that within ASA these modifications are implicitly included.

Each ASA run in our work was started from a random initial configuration $\{\phi, \psi, \chi\}$. The dihedral angle ω was fixed to 180° in all of the ASA runs. The initial temperature was determined by the average energy of 5 or 10 random samplings, and a full search range of the dihedral angles $(-\pi, \pi)$ was set. The typical maximum number of calls to the energy function was 30000. An ASA run was terminated if it repeated the best energy value for 3 or 5 re-annealing cycles (each cycle generates 100 configurations). Further refinement of the final ASA optimized configuration was carried out by using the local minimizer SUMSL [1], or the conjugate gradient method. The combination of the ASA application and a local minimizer improved the efficiency of the search.

The ASA calculation is governed by various control parameters [3], for which the most important setting is the annealing rate for the temperatures of 'cost' and 'parameters', determined by the so-called 'temperature-ratio-scale' (the ratio of the final to the initial temperature after certain annealing steps) and the 'cost-parameter-scale'. The control parameters were varied to improve the search efficiency. Adequate control parameters used for obtaining the results reported in this study were: 'temperature-ratio-scale' = 10^{-4}; 'cost-parameter-scale' = 0.5. These parameter settings correspond to an annealing rate for energy of $c_{\text{cost}} = 3.6$, and for all dihedral angles of $c_{\text{parameter}} = 7.2$. Note that the annealing rate for all dihedral angles was chosen to be the same.

Met-Enkephalin

Met-Enkephalin has a complicated energy surface [11,16]. The lowest energy for Met-Enkephalin was found to be −12.9 kcal/mol with the force field being ECEPP/2 (Empirical Conformation Energy Program for Peptides) [8]. With all ω fixed, the lowest energy was found to be −10.7 kcal/mol by MCM [14]. Using different initial conformations and control parameter settings of the cooling schedule as described above, 55 independent ASA runs were carried out. Table 1 summarizes the energy distribution of these calculations. Most of the ASA calculations result in energies in the range of −8 to −3 kcal/mol, with 7 of the results determining conformations having energies that are only 3 kcal/mol above the known lowest energy, thus exhibiting the effectiveness of the approach. Moreover, as the range of search was somewhat narrowed, almost all of the ASA runs reach the global energy minimum.

Adaptive Simulated Annealing and its Application to Protein Folding, Table 1
The energy (in kcal/mol) distribution of ASA runs for Met-Enkephalin using a full search range

Energy	< -8	$(-8, -5)$	$(-5, -3)$	> -3
No. of runs	7	19	19	10

Adaptive Simulated Annealing and its Application to Protein Folding, Table 2
Energy and dihedral angles of the lowest energy conformations of Met-Enkephalin calculated by ASA. RMSD1 is the root-mean-square deviation (in Å) for backbone atoms, while RMSD2 is for all atoms

	A0	A	1	2	3	4
E	−12.9	−10.7	−10.6	−10.4	−10.1	−8.5
ϕ_1	−86	−87	−87	−87	−87	−87
ψ_1	156	154	153	153	156	153
ϕ_2	−155	−162	−161	−162	−166	−166
ψ_2	84	71	72	75	87	72
ϕ_3	84	64	64	63	68	63
ψ_3	−74	−93	−94	−95	−91	−97
ϕ_4	−137	−82	−83	−81	−103	−74
ψ_4	19	−29	−26	−30	−13	−30
ϕ_5	−164	−81	−79	−76	−76	−82
ψ_5	160	144	133	132	137	143
χ_1^1	−173	−180	180	179	−166	−180
χ_1^2	79	−111	−110	71	88	73
χ_1^3	−166	145	145	−35	−148	−179
χ_4^1	59	180	72	−179	71	179
χ_4^2	−86	−100	84	−100	−93	−100
χ_5^1	53	−65	−171	−173	−65	−65
χ_5^2	175	−179	176	176	−178	−179
χ_5^3	−180	−179	180	179	−178	−179
χ_5^4	−58	−180	−60	60	−178	−179
RMSD1		0	0.04	0.07	0.51	0.26
RMSD2		0	2.52	1.92	2.08	1.29

For the full range search, we identified three conformations with energies of −10.6, −10.4, and −10.1 kcal/mol, that exhibit the configuration of the known lowest geometry of −10.7 kcal/mol. Table 2 lists the conformations of these lowest energy configurations, as well as an additional low energy structure. Conformations A0 and A are the lowest-energy conformations with ω nonfixed and fixed, respectively, taken from [11,14]. The first two conformations, #1 and #2, have almost the same backbone configuration as that of A (−10.7 kcal/mol), with a backbone root-mean-square deviation (RMSD) of only 0.04 and 0.07Å, respectively. The all-atom RMSD of the listed conformations with energies ranging from −8.5 to −10.6 kcal/mol are about 2Å. For conformations #1 and #2, the noted differences are in the side-chains, corresponding to a 0.1 and 0.3 kcal/mol difference in energy, respectively.

Adaptive Simulated Annealing and its Application to Protein Folding, Table 3
The conformation of a model 14(L-Alanine) peptide as calculated by ASA

	2	3	4	5	6
ϕ	−99.4	−68.2	−68.0	−69.3	−66.9
ψ	158.1	−34.3	−38.8	−38.5	−38.6
	7	8	9	10	11
ϕ	−68.3	−66.7	−68.8	−67.1	−69.4
ψ	−39.2	−38.0	−38.7	−37.7	−39.6
	12	13	14	15	
ϕ	−65.0	−67.2	−87.7	−75.9	
ψ	−40.0	−44.6	65.8	−40.1	

Poly(L-Alanine)

The ASA algorithm was applied to a model of (L-Alanine) that is known to assume a dominant right-handed α-helical structure [13]. For a search range of dihedral angles that include both the right-handed (RH) α-helix and the β-sheet region in the Ramachandran's diagram, ψ: (−115°, − 180°) and ϕ: (− 115°, 0°), it was significant to find RH α-helices with $\phi \approx -68°$ and $\psi \approx -38°$ in all backbones except those near the end-groups, as shown in Table 3. The energy of such a geometry is typically −10.2 kcal/mol after a local minimization. The energy surfaces of the RH α-helical regions were found to be less complex than those of Met-Enkephalin. These results are consistent with a previous study [16].

Conclusion

The adaptive simulated annealing as a global optimization method intrinsically includes some of the modifications of conventional simulated annealing used for biomolecular structure determination. As applied to Met-Enkephalin, the performance of ASA is comparable to the simulated annealing study reported in [12], while better than the one reported in [11], although some differences other than the algorithms are noted. Utilizing a partial search range improves the efficiency significantly, showing that ASA may be useful for refinement of a molecular structure predicted or measured by other methods. A dominant right-handed α-helical conformation was found for the 14 residue (L-Alanine) model, with deviations observed only near the end groups.

Recent Studies and Future Directions

Recent studies have shown improved efficiency in the conformational search of Met-Enkephalin, *e.g.*, the so-called conformation space annealing (CSA), which combines the ideas of genetic algorithms, simulated annealing, a build up procedure, and local minimization [7]. The use of the multicanonical ensemble algorithm (ME) (one of the generalized-ensemble algorithms [2]), allows free random walks in energy space, escaping from any energy barrier. Both the ME and CSA algorithms outperform genetic algorithms (GA), simulated annealing (SA), GA with minimization (GAM) and Monte-Carlo with minimization (MCM). Our own work (unpublished) and the work in ref. [5] both show that simple GA alone underperforms simulated annealing for the Met-Enkephalin conformational search problem. Table 4 compares these algorithms for efficiency (the number of evaluations of energy and energy gradient, or the number of local minimizations) and effectiveness (the number of runs reaching the ground state conformation (hits) versus the number of total independent runs). Caution should be exercised since some differences exist between these studies, such as the version of the ECEPP potential used, the treatment of the peptide dihedral angle ω, etc. Ground state conformations are those having energy within approximately 1eV from the known global minimum energy. Note that the generalized-ensemble method can be carried out with both Monte-Carlo and molecular dynamics.

Adaptive Simulated Annealing and its Application to Protein Folding, Table 4
Comparison of the conformation search efficiency and effectiveness of Met-Enkephalin using different algorithms. N_E, $N_{\nabla E}$, and N_{minz} are the number of the evaluations of energy, energy gradient, and number of local minimizations of each run, in the unit of 10^3

	hits/total	N_E	$N_{\nabla E}$	N_{minz}
ME [2]	10/10	< 1900	0	0
MCM [11]	24/24	$*$	$*$	15
GAM [10]	5/5	$*$	$*$	50
ME [2]	18/20	950	0	0
CSA [7]	99/100	300	250	5
ME [2]	21/50	400	0	0
CSA [7]	50/100	170	130	2.6
SA [2]	8/20	1000	0	0
GA [5]	$< 1/27$	100	0	0.001

$*$: The total number of $E, \nabla E$ evaluations are not given, but can be estimated based on roughly 100 evaluations for each minimization.

In comparison to the studies summarized in Table 4, ASA seems to be using too small a number of function evaluations. Optimizing control parameters such as the annealing schedule and increasing the number of energy evaluations may improve the effectiveness. Search efficiency could also be improved by adopting parallellization to achieve scalable simulation for various algorithms. Extensive research on the protein conformational search using various hybrids of genetic algorithms and parallelization is in progress (as of 1999).

See also

- ▶ Adaptive Global Search
- ▶ Bayesian Global Optimization
- ▶ Genetic Algorithms
- ▶ Genetic Algorithms for Protein Structure Prediction
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Lennard–Jones and Morse Clusters
- ▶ Global Optimization in Protein Folding
- ▶ Molecular Structure Determination: Convex Global Underestimation
- ▶ Monte-Carlo Simulated Annealing in Protein Folding
- ▶ Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach
- ▶ Packet Annealing
- ▶ Phase Problem in X-ray Crystallography: Shake and Bake Approach
- ▶ Protein Folding: Generalized-ensemble Algorithms
- ▶ Random Search Methods
- ▶ Simulated Annealing
- ▶ Simulated Annealing Methods in Protein Folding
- ▶ Stochastic Global Optimization: Stopping Rules
- ▶ Stochastic Global Optimization: Two-phase Methods

References

1. Gay DM (1983) Subroutines for unconstrained minimization using a model/trust-region approach. ACM Trans Math Softw 9:503

2. Hansmann UHE (1998) Generalized ensembles: A new way of simulating proteins. Phys A 254:15
3. Ingber L (1989) Very fast simulated re-annealing. Math Comput Modelling 12:967 ASA code is available from: ftp.alumni.caltech.edu:pub/ingber
4. Ingber L, Rosen B (1992) Genetic algorithm and very fast simulated re-annealing: A comparison. Math Comput Modelling 16:87
5. Jin AY, Leung FY, Weaver DF (1997) Development of a novel genetic algorithm search method (GAP1.0) for exploring peptide conformational space. J Comput Chem 18:1971
6. Kirkpatrick S, Gelatt CD, Vecchi MP Jr (1983) Optimization by simulated annealing. Science 220:671
7. Lee J, Scheraga HA, Rackovsky S (1997) New optimization method for conformational energy calculations on polypeptides: conformational space annealing. J Comput Chem 18:222
8. Li Z, Scheraga HA (1987) Monte Carlo-minimization approach to the multiminima problem in protein folding. Proc Nat Aca Sci USA 84:6611
9. Li Z, Scheraga HA (1988) Structure and free energy of complex thermodynamic systems. J Mol Struct (Theochem) 179:333
10. Merkle LD, Lamont GB, Gates GH, Pachter R (May, 1996) Hybrid genetic algorithms for minimization of polypeptide specific energy model. Proc. IEEE Int. Conf. Evolutionary Computation, p 192
11. Nayeem A, Vila J, Scheraga HA (1991) A comparative study of the simulated-annealing and Monte Carlo-with minimization approaches to the minimum-energy structures of polypeptides: Met-Enkephalin. J Comput Chem 12:594
12. Okamoto Y, Kikuchi T, Kawai H (1992) Prediction of low-energy structure of Met-Enkephalin by Monte Carlo simulated annealing. Chem Lett (Chem Soc Japan):1275
13. Piela L, Scheraga HA (1987) On the multiple-minima problem in the conformational analysis of polypeptides: I. backbone degrees of freedom for a perturbed α-helix. Biopolymers 26:S33
14. Vasquez M (1999) Private communication
15. Wang Z, Pachter R (1997) Prediction of polypeptide conformation by the adaptive simulated annealing approach. J Comput Chem 18:323
16. Wilson SR, Cui W (1990) Applications of simulated annealing to peptides. Biopolymers 29:225

Affine Sets and Functions

LEONIDAS PITSOULIS
Princeton University, Princeton, USA

MSC2000: 51E15, 32B15, 51N20

Article Outline

Keywords
See also
References

Keywords

Linear algebra; Convex analysis

A subset S of $\mathbf{R}^n$ is an *affine set* if

$$(1-\lambda)x + \lambda y \in S,$$

for any $x, y \in S$ and $\lambda \in \mathbf{R}$. A function f: $\mathbf{R}^n \to \mathbf{R}$ is an *affine function* if f is finite, convex and concave (cf. ► Convex max-functions).

See also

► Linear Programming
► Linear Space

References

1. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton

Airline Optimization

GANG YU[1], BENJAMIN THENGVALL[2]
[1] Department Management Sci. and Information Systems Red McCombs School of Business, University Texas at Austin, Austin, USA
[2] CALEB Technologies Corp., Austin, USA

MSC2000: 90B06, 90C06, 90C08, 90C35, 90C90

Article Outline

Keywords
See also
References

Keywords

Network design and schedule construction; Fleet assignment; Aircraft routing; Crew scheduling; Revenue management; Irregular operations; Air traffic control and ground delay programs

The airline industry was one of the first to apply operations research methodology and techniques on a large scale. As early as the late 1950s, operations researchers were beginning to study how the developing fields of mathematical programming could be used to address a number of very difficult problems faced by the airline industry. Since that time many airline related problems have been the topics of active research [26]. Most optimization-related research in the airline industry can be placed in one of the following areas:

- network design and schedule construction;
- fleet assignment;
- aircraft routing;
- crew scheduling;
- revenue management;
- irregular operations;
- air traffic control and ground delay programs.

In the following, each of these problem areas will be defined along with a brief discussion of some of the operations research techniques that have been applied to solve them. The majority of applications utilize network-based models. Solution of these models range from traditional mathematical programming approaches to a variety of novel heuristic approaches. A very brief selection of references is also provided.

Construction of flight schedules is the starting point for all other airline optimization problems and is a critical operational planning task faced by an airline. The *flight schedule* defines a set of flight segments that an airline will service along with corresponding origin and destination points and departure and arrival times for each flight segment. An airline's decision to offer certain flights will depend in large part on market demand forecasts, available aircraft operating characteristics, available manpower, and the behavior of competing airlines [11,12].

Of course, prior to the construction of flight schedules, an airline must decide which markets it will serve. Before the 1978 'Airline Deregulation Act', airlines had to fly routes as assigned by the Civil Aeronautics Board regardless of the demand for service. During this period, most airlines emphasized long point-to-point routes. Since deregulation, airlines have gained the freedom to choose which markets to serve and how often to serve them. This change led to a fundamental shift in most airlines routing strategies from point-to-point flight networks to hub-and-spoke oriented flight networks. This, in turn, led to new research activities for finding optimal hub [3,18] and maintenance base [13] locations.

Following network design and schedule construction, an aircraft type must be assigned to each flight segment in the schedule. This is called the *fleet assignment problem*. Airlines generally operate a number of different fleet types, each having different characteristics and costs such as seating capacity, landing weights, and crew and fuel costs. The majority of fleet assignment methods represent the flight schedule via some variant of a time-space network with flight arcs between stations and inventory arcs at each station. A *multicommodity network flow problem* can then be formulated with arcs and nodes duplicated as appropriate for all fleets that can take a particular flight. Side constraints must be implemented to ensure each flight segment is assigned to only one fleet. In domestic fleet assignment problems, a common simplifying assumption is that every flight is flown every day of the week. Under this assumption, the network model need only account for one day's flights and a looping arc connects the end of the day with the beginning. The resulting models are mixed integer programs [1,16,27,30].

Aircraft routing is a fleet by fleet process of assigning individual aircraft to fly each flight segment assigned to a particular fleet. A primary consideration at this stage is *maintenance* requirements mandated by the Federal Aviation Administration. There are different types of maintenance activities that must be performed after a given number of flight hours. The majority of these maintenance activities can be performed overnight; however, not all stations are equipped with proper maintenance facilities for all fleets. During the aircraft routing process, individual aircraft from each fleet must be assigned to fly all flight segments assigned to that fleet in a manner that provides maintenance opportunities for all aircraft at appropriate stations within the required time intervals. This problem has been formulated and solved in a number of ways including as a general integer programming problem solved by Lagrangian relaxation [9] and as a set partitioning problem solved with a branch and bound algorithm [10].

As described above, the problems of fleet assignment and aircraft routing have been historically solved in a sequential manner. Recently, work has been done to solve these problems simultaneously using a string-

based model and a branch and price solution approach [5].

Crew scheduling, like aircraft routing, is done following fleet assignment. The first of two sequentially solved crew scheduling problems is the crew pairing problem. A crew pairing is a sequence of flight legs beginning and ending at a crew base that satisfies all governmental and contractual restrictions (some times called legalities). These crew pairings generally cover a period of 2–5 days. The problem is to find a minimum cost set of such crew pairings such that all flight segments are covered. This problem has generally been modeled as a set partitioning problem in which pairings are enumerated or generated dynamically [15,17]. Other attempts to solve this problem have employed a decomposition approach based on graph partitioning [4] and a linear programming relaxation of a set covering problem [21]. Often a practice called *deadheading* is used to reposition flight crews in which a crew will fly a flight segment as passengers. Therefore, in solving the crew-pairing problem, all flight segments must be covered, but they may be covered by more than one crew.

The second problem to be solved relating to crew scheduling is the monthly crew rostering problem. This is the problem of assigning individual crew members to crew pairings to create their monthly schedules. These schedules must incorporate time off, training periods, and other contractual obligations. Generally, a *preferential bidding system* is used to make the assignments in which each personalized schedule takes into account an employee's pre-assigned activities and weighted bids representing their preferences. While the crew pairing problem has been widely studied, a limited number of publications have dealt with the monthly crew rostering problem. Approaches include an integer programming scheme [14] and a network model [24].

Revenue management is the problem of determining fare classes for each flight in the flight schedule as well as the allocation of available seats to each fare class. Not only are seats on an airplane partitioned physically into sections such as first class and coach, but also seats in the same section are generally priced at many different levels. The goal is to maximize the expected revenue from a particular flight segment by finding the proper balance between gaining additional revenue by selling more inexpensive seats and losing revenue by turning away higher fare customers. A standard assumption is that fare classes are filled sequential from the lowest to the highest. This is often the case where discounted fares are offered in advance, while last minute tickets are sold at a premium. Recent research includes a probabilistic decision model [6], a dynamic programming formulation [31] and some calculus-based booking policies [8].

When faced with a lack of resources, airlines often are not able to fly their published flight schedule. This is frequently the result of aircraft mechanical difficulties, inclement weather, or crew shortages. As situations like these arise, decisions must be made to deal with the shortage of resources in a manner that returns the airline to the originally planned flight schedule in a timely fashion while attempting to reduce operational cost and keep passengers satisfied. This general situation is called the *airline irregular operations problem* and it involves aircraft, crew, gates, and passenger recovery.

The aircraft schedule recovery problem deals with re-routing aircraft during irregular operations. This problem has received significant attention among irregular operations topics; papers dealing with crew scheduling during irregular operations have only recently started to appear [28,35]. Most approaches for dealing with aircraft schedule recovery have been based on network models. Some early models were pure networks [19]. Recently, more comprehensive models have been developed that better represent the problem, but are more difficult to solve as side constraints have been added to the otherwise network structure of these problems [2,33,36]. In practice, many airlines use heuristic methods to solve these problems as their real-time nature does not allow for lengthy optimization run times.

Closely related to the irregular operations problem is the *ground delay problem* in air traffic control. Ground delay is a program implemented by the Federal Aviation Administration in cases of station congestion. During ground delay, aircraft departing for a congested station are held on the ground before departure. The rational for this behavior is that ground delays are less expensive and safer than airborne delays. Several optimization models have been formulated to decrease the total minutes of delay experienced throughout the system during a ground delay program. These problems have generally been modeled as integer programs ([22,23]), but the problem has also been solved using

stochastic linear programming [25] and by heuristic methods [34].

Optimization based methods have also been applied to a myriad of other airline related topics such as gate assignment [7], fuel management [29], short term fleet assignment swapping [32], demand modeling [20], and others. Airline industry is an exciting arena for the interplay between optimization theory and practice. Many more optimization applications in the airline industry will evolve in the future.

See also

- ► Integer Programming
- ► Vehicle Scheduling

References

1. Abara J (1989) Applying integer linear programming to the fleet assignment problem. Interfaces 19(4):20–28
2. Argüello MF, Bard JF, Yu G (1997) Models and methods for managing airline irregular operations aircraft routing. In: Yu G (ed) Operations Research in Airline Industry. Kluwer, Dordrecht
3. Aykin T (1994) Lagrangian relaxation based approaches to capacitated hub-and-spoke network design problem. Europ J Oper Res 79(3):501–523
4. Ball M, Roberts A (1985) A graph partitioning approach to airline crew scheduling. Transport Sci 19(2):107–126
5. Barnhart C, Boland NL, Clarke LW, Johnson EL, Nemhauser G, Shenoi RG (1998) Flight string models for aircraft fleeting and routing. Transport Sci 32(3):208–220
6. Belobaba PP (1989) Application of a probabilistic decision model to airline seat inventory control. Oper Res 37:183–197
7. Brazile RP, Swigger KM, Wyatt DL (1994) Selecting a modelling technique for the gate assignment problems: Integer programming, simulation, or expert system. Internat J Modelling and Simulation 14(1):1–5
8. Brumelle SL, McGill JI (1993) Airline seat allocation with multiple nested fare classes. Oper Res 41(1):127–137
9. Daskin MS, Panagiotopoulos ND (1989) A Lagrangian relaxation approach to assigning aircraft to routes in hub and spoke networks. Transport Sci 23(2):91–99
10. Desaulniers G, Desrosiers J, Dumas Y, Solomon MM, Soumis F (1997) Daily aircraft routing and scheduling. Managem Sci 43(6):841–855
11. Dobson G, Lederer PJ (1993) Airline scheduling and routing in a hub-and-spoke system. Transport Sci 27(3):281–297
12. Etschamaier MM, Mathaisel DFX (1985) Airline scheduling: An overview. Transport Sci 9(2):127–138
13. Feo TA, Bard JF (1989) Flight scheduling and maintenance base planning. Managem Sci 35(12):1415–1432
14. Gamache M, Soumis F, Villeneuve D, Desrosiers J (1998) The preferential bidding system at Air Canada. Transport Sci 32(3):246–255
15. Graves GW, McBride RD, Gershkoff I, Anderson D, Mahidhara D (1993) Flight crew scheduling. Managem Sci 39(6):736–745
16. Hane CA, Barnhart C, Johnson EL, Marsten RE, Nemhauser GL, Sigismondi G (1995) The fleet assignment problem: Solving a large-scale integer program. Math Program 70(2):211–232
17. Hoffman KL, Padberg M (1993) Solving airline crew scheduling problems by branch-and-cut. Managem Sci 39(6):657–680
18. Jaillet P, Song G, Yu G (1997) Airline network design and hub location problems. Location Sci 4(3):195–212
19. Jarrah AIZ, Yu G, Krishnamurthy N, Rakshit A (1993) A decision support framework for airline flight cancellations and delays. Transport Sci 27(3):266–280
20. Jorge-Calderon JD (1997) A demand model for scheduled airline services on international European routes. J Air Transport Management 3(1):23–35
21. Lavoie S, Minoux M, Odier E (1988) A new approach for crew pairing problems by column generation with an application to air transportation. Europ J Oper Res 35:45–58
22. Luo S, Yu G (1997) On the airline schedule perturbation problem caused by the ground delay program. Transport Sci 31(4):298–311
23. Navazio L, Romanin-Jacur G (1998) The multiple connections multi-airport ground holding problem: Models and algorithms. Transport Sci 32(3):268–276
24. Nicoletti B (1975) Automatic crew rostering. Transport Sci 9(1):33–48
25. Richetta O, Odoni AR (1993) Solving optimally the static ground-holding policy problem in air traffic control. Transport Sci 27(3):228–238
26. Richter H (1989) Thirty years of airline operations research. Interfaces 19(4):3–9
27. Rushmeier RA, Kontogiorgis SA (1997) Advances in the optimization of airline fleet assignment. Transport Sci 31(2):159–169
28. Stojkovic M, Soumis F, Desrosiers J (1998) The operational airline crew scheduling problem. Transport Sci 32(3):232–245
29. Stroup JS, Wollmer RD (1992) A fuel management model for the airline industry. Oper Res 40(2):229–237
30. Subramanian R, Scheff RP Jr, Quillinan JD, Wiper DS, Marsten RE (1994) Coldstart: Fleet assignment at delta air lines. Interfaces 24(1):104–120
31. Tak TC, Hersh M (1993) A model for dynamic airline seat inventory control with multiple seat bookings. Transport Sci 27(3):252–265
32. Talluri KT (1993) Swapping applications in a daily airline fleet assignment. Transport Sci 30(3):237–248

33. Thengvall BT, Bard JF, Yu G (2000) Balancing user preferences for aircraft schedule recovery during airline irregular operations. IEE Trans Oper Eng 32(3):181–193
34. Vranas PB, Bertsemas DJ, Odoni AR (1994) The multi-airport ground-holding problem in air traffic control. Oper Res 42(2):249–261
35. Wei G, Song G, Yu G (1997) Model and algorithm for crew management during airline irregular operations. J Combin Optim 1(3):80–97
36. Yan S, Tu Y (1997) Multifleet routing and multistop flight scheduling for schedule perturbation. Europ J Oper Res 103(1):155–169

Algorithmic Improvements Using a Heuristic Parameter, Reject Index for Interval Optimization

TIBOR CSENDES
University of Szeged, Szeged, Hungary

MSC2000: 65K05, 90C30

Article Outline

Keywords and Phrases
Introduction
Subinterval Selection
Multisection
Heuristic Rejection
References

Keywords and Phrases

Branch-and-bound; Interval arithmetic; Optimization; Heuristic parameter

Introduction

Interval optimization methods (► interval analysis: unconstrained and constrained optimization) have the guarantee not to loose global optimizer points. To achieve this, a deterministic branch-and-bound framework is applied. Still, heuristic algorithmic improvements may increase the convergence speed while keeping the guaranteed reliability.

The indicator parameter called RejectIndex

$$pf^*(X) = \frac{f^* - \underline{F}(X)}{\overline{F}(X) - \underline{F}(X)}$$

was suggested by L.G. Casado as a measure of the closeness of the interval X to a global minimizer point [1]. It was first applied to improve the work load balance of global optimization algorithms.

A subinterval X of the search space with the minimal value of the inclusion function $\underline{F}(X)$ is usually considered as the best candidate to contain a global minimum. However, the larger the interval X, the larger the overestimation of the range $f(X)$ on X compared to $F(X)$. Therefore a box could be considered as a good candidate to contain a global minimum just because it is larger than the others. To compare subintervals of different sizes we normalize the distance between the global minimum value f^* and $\underline{F}(X)$.

The idea behind pf^* is that in general we expect the overestimation to be symmetric, i. e., the overestimation above $f(X)$ is closely equal to the overestimation below $f(X)$ for small subintervals containing a global minimizer point. Hence, for such intervals X the relative place of the global optimum value inside the $F(X)$ interval should be high, while for intervals far from global minimizer points pf^* must be small. Obviously, there are exceptions, and there exists no theoretical proof that pf^* would be a reliable indicator of nearby global minimizer points.

The value of the global minimum is not available in most cases. A generalized expression for a wider class of indicators is

$$p(\hat{f}, X) = \frac{\hat{f} - \underline{F}(X)}{\overline{F}(X) - \underline{F}(X)},$$

where the $\hat{f}$ value is a kind of approximation of the global minimum. We assume that $\hat{f} \in F(X)$, i. e., this estimation is realistic in the sense that $\hat{f}$ is within the known bounds of the objective function on the search region. According to the numerical experience collected, we need a good approximation of the f^* value to improve the efficiency of the algorithm.

Subinterval Selection

I. Among the possible applications of these indicators the most promising and straightforward is in the *subinterval selection*. The theoretical and computational properties of the interval branch-and-bound optimization has been investigated extensively [6,7,8,9]. The most important statements proved are the follow-

ing for algorithms with balanced subdivision direction selection:

1. Assume that the inclusion function of the objective function is isotone, it has the zero convergence property, and the $p(f_k,Y)$ parameters are calculated with the f_k parameters converging to $\hat{f} > f^*$, for which there exists a point $\hat{x} \in X$ with $f(\hat{x}) = \hat{f}$. Then the branch-and-bound algorithm that selects that interval Y from the working list which has the maximal $p(f_i,Z)$ value can converge to a point $\hat{x} \in X$ for which $f(\hat{x}) > f^*$, i. e., to a point which is not a global minimizer point of the given problem.
2. Assume that the inclusion function of the objective function has the zero convergence property and f_k converges to $\hat{f} < f^*$. Then the optimization branch-and-bound algorithm will produce an everywhere dense sequence of subintervals converging to each point of the search region X regardless of the objective function value.
3. Assume that the inclusion function of the objective function is isotone and has the zero convergence property. Consider the interval branch-and-bound optimization algorithm that uses the cutoff test, the monotonicity test, the ► interval Newton step, and the concavity test as accelerating devices, and that selects as the next leading interval that interval Y from the working list which has the maximal $p(f_i,Z)$ value. A necessary and sufficient condition for the convergence of this algorithm to a set of global minimizer points is that the sequence $\{f_i\}$ converges to the global minimum value f^*, and there exist at most a finite number of f_i values below f^*.
4. If our algorithm applies the interval selection rule of maximizing the $p(f^*, X) = pf^*(X)$ values for the members of the list L (i. e., if we can use the known exact global minimum value), then the algorithm converges exclusively to global minimizer points.
5. If our algorithm applies the interval selection rule of maximizing the $p(\tilde{f}, X)$ values for the members of the list L, where $\tilde{f}$ is the best available upper bound for the global minimum, *and its convergence to f^* can be ensured*, then the algorithm converges exclusively to global minimizer points.
6. Assume that for an optimization problem $\min_{x\in X} f(x)$ the inclusion function $F(X)$ of $f(x)$ is isotone and α-convergent with given positive constants α and C. Assume further that the pf^* parameter is less than 1 for all the subintervals of X. Then an arbitrary large number $N(> 0)$ of consecutive leading intervals of the basic B&B algorithm that selects the subinterval with the smallest lower bound as the next leading interval may have the following properties:
   i. None of these processed intervals contains a stationary point.
   ii. During this phase of the search the pf^* values are maximal for these intervals.
7. Assume that the inclusion function of the objective function is isotone and it has the zero convergence property. Consider the interval branch-and-bound optimization algorithm that uses the cutoff test, the monotonicity test, the interval Newton step, and the concavity test as accelerating devices and that selects as the next leading interval that interval Y from the working list which has the maximal $pf(f_k,Z)$ value.
   i. The algorithm converges exclusively to global minimizer points if

$$\underline{f}_k \le f_k < \delta(\overline{f}_k - \underline{f}_k) + \underline{f}_k$$

   holds for each iteration number k, where $0 < \delta < 1$.
   ii. The above condition is sharp in the sense that $\delta = 1$ allows convergence to not optimal points.

Here $\underline{f}_k = \min\{\underline{F}(Y^l), l = 1, \ldots, |L_k|\} \le f_k < \tilde{f}_k = \overline{f}_k$, where $|L|$ stands for the cardinality of the elements of the list L.

II. These theoretical results are in part promising (e. g., 7), in part disappointing (5 and 6). The conclusions of the detailed numerical comparisons were that if the global minimum value is known, then the use of the pf^* parameter in the described way can accelerate the interval optimization method by orders of magnitude, and this improvement is especially strong for hard problems.

In case the global minimum value is not available, then its estimation, f_k, which fulfills the conditions of 7, can be utilized with similar efficacy, and again the best results were achieved on difficult problems.

Multisection

I. The multisection technique is a way to accelerate branch-and-bound methods by subdividing the actual interval into several subintervals in a single algorithm

step. In the extreme case half of the function evaluations can be saved [5,10]. On the basis of the RejectIndex value of a given interval it is decided whether simple bisection or two higher-degree multisections are to be applied [2,11]. Two threshold values, $0 < P_1 < P_2 < 1$, are used for selecting the proper multisection type.

This algorithm improvement can also be cheated in the sense that there exist global optimization problems for which the new method will follow for an arbitrary long number of iterations an embedded interval sequence that contains no global minimizer point, or that intervals in which there is a global minimizer have misleading indicator values.

According to the numerical tests, the new multisection strategies result in a substantial decrease both in the number of function evaluations and in the memory complexity.

II. The multisection strategy can also be applied to constrained global optimization problems [11]. The feasibility degree index for constraint $g_j(x) \leq 0$ can be formulated as

$$pu_{G_j}(X) = \min\left\{\frac{-\underline{G}_j(X)}{w(G_j(X))}, 1\right\} .$$

Notice that if $pu_{G_j}(X) < 0$, then the box is certainly infeasible, and if $pu_{G_j}(X) = 1$ then X certainly satisfies the constraint. Otherwise, the box is undetermined for that constraint. For boxes that are not certainly infeasible, i. e., for which $pu_{G_j}(X) \geq 0$ for all $j = 1, \ldots, r$ holds, the total infeasibility index is given by

$$pu(X) = \prod_{j=1}^{r} pu_{G_j}(X) .$$

We must only define the index for such boxes since certainly infeasible boxes are immediately removed by the algorithm from further consideration. With this definition,

- $pu(X) = 1 \Leftrightarrow X$ is certainly feasible and
- $pu(X) \in [0, 1) \Leftrightarrow X$ is undetermined.

Using the $pu(X)$ index, we now propose the following modification of the RejectIndex for constrained problems:

$$pup(\hat{f}, X) = pu(X) \cdot p(\hat{f}, X) ,$$

where $\hat{f}$ is a parameter of this indicator, which is usually an approximation of f^*. This new index works like $p(\hat{f}, X)$ if X is certainly feasible, but if the box is undetermined, then it takes the feasibility degree of the box into account: the less feasible the box is, the lower the value of $pu(X)$ is.

A careful theoretical analysis proved that the new interval selection and multisection rules enable the branch-and-bound interval optimization algorithm to converge to a set of global optimizer points assuming we have a proper sequence of $\{f_k\}$ parameter values. The convergence properties obtained were very similar to those proven for the unconstrained case, and they give a firm basis for computational implementation.

A comprehensive numerical study on standard global optimization test problems and on facility location problems indicated [11] that the constrained version interval selection rules and, to a lesser extent, also the new adaptive multisection rules have several advantageous features that can contribute to the efficiency of the interval optimization techniques.

Heuristic Rejection

RejectIndex can also be used to improve the efficiency of interval global optimization algorithms on very hard to solve problems by applying a rejection strategy to get rid of subintervals not containing global minimizer points. This heuristic rejection technique selects those subintervals on the basis of a typical pattern of changes in the pf^* values [3,4].

The RejectIndex is not always reliable: assume that the inclusion function $F(X)$ of $f(x)$ is isotone and α-convergent. Assume further that the RejectIndex parameter pf^* is less than 1 for all the subintervals of X. Then an arbitrary large number $N(> 0)$ of consecutive leading intervals may have the following properties:

i. Neither of these processed intervals contains a stationary point, and
ii. During this phase of the search the pf^* values are maximal for these intervals as compared with the subintervals of the current working list.

Also, when a global optimization problem has a unique global minimizer point x^*, there always exists an isotone and α-convergent inclusion function $F(X)$ of $f(x)$ such that the new algorithm does not converge to x^*.

In spite of the possibility of losing the global minimum, obviously there exist such implementations that allow a safe way to use heuristic rejection. For example, the selected subintervals can be saved on a hard disk for further possible processing if necessary.

Although the above theoretical results were not encouraging, the computational tests on very hard global optimization problems were convincing: when the whole list of subintervals produced by the B&B algorithm is too large for the given computer memory, then the use of the suggested heuristic rejection technique decreases the number of working list elements without missing the global minimum. The new rejection test may also make it possible to solve hard-to-solve problems that are otherwise unsolvable with the usual techniques.

References

1. Casado LG, García I (1998) New Load Balancing Criterion for Parallel Interval Global Optimization Algorithms. In: Proceedings of the 16th IASTED International Conference on Applied Informatics, Garmisch-Partenkirchen, pp 321–323
2. Casado LG, García I, Csendes T (2000) A new multisection technique in interval methods for global optimization. Computing 65:263–269
3. Casado LG, García I, Csendes T (2001) A heuristic rejection criterion in interval global optimization algorithms. BIT 41:683–692
4. Casado LG, García I, Csendes T, Ruiz VG (2003) Heuristic Rejection in Interval Global Optimization. JOTA 118:27–43
5. Csallner AE, Csendes T, Markót MC (2000) Multisection in Interval Branch-and-Bound Methods for Global Optimization I. Theoretical Results. J Global Optim 16:371–392
6. Csendes T (2001) New subinterval selection criteria for interval global optimization. J Global Optim 19:307–327
7. Csendes T (2003) Numerical experiences with a new generalized subinterval selection criterion for interval global optimization. Reliab Comput 9:109–125
8. Csendes T (2004) Generalized subinterval selection criteria for interval global optimization. Numer Algorithms 37:93–100
9. Kreinovich V, Csendes T (2001) Theoretical Justification of a Heuristic Subbox Selection Criterion for Interval Global Optimization. CEJOR 9:255–265
10. Markót MC, Csendes T, Csallner AE (2000) Multisection in Interval Branch-and-Bound Methods for Global Optimization II. Numerical Tests. J Global Optim 16:219–228
11. Markót MC, Fernandez J, Casado LG, Csendes T (2006) New interval methods for constrained global optimization. Math Programm 106:287–318

Algorithms for Genomic Analysis

EVA K. LEE, KAPIL GUPTA
Center for Operations Research in Medicine and HealthCare,
School of Industrial and Systems Engineering,
Georgia Institute of Technology, Atlanta, USA

MSC2000: 90C27, 90C35, 90C11, 65K05, 90-08, 90-00

Article Outline

Abstract
Introduction
Phylogenetic Analysis
  Methods Based on Pairwise Distance
  Parsimony Methods
  Maximum Likelihood Methods
Multiple Sequence Alignment
  Scoring Alignment
  Alignment Approaches
  Progressive Algorithms
  Graph-Based Algorithms
  Iterative Algorithms
Novel Graph-Theoretical Genomic Models
  Definitions
  Construction of a Conflict Graph from Paths of Multiple Sequences
  Complexity Theory
  Special Cases of MWCMS
  Computational Models: Integer Programming Formulation
Summary
Acknowledgement
References

Abstract

The genome of an organism not only serves as its blueprint that holds the key for diagnosing and curing diseases, but also plays a pivotal role in obtaining a holistic view of its ancestry. Recent years have witnessed a large number of innovations in this field, as exemplified by the Human Genome Project. This chapter provides an overview of popular algorithms used in genome analysis and in particular explores two important and deeply interconnected problems: phylogenetic analysis and multiple sequence alignment. We also describe our novel graph-theoretical approach that en-

compasses a wide variety of genome sequence analysis problems within a single model.

Introduction

Genomics encompasses the study of the genome in human and other organisms. The rate of innovation in this field has been breathtaking over the last decade, especially with the completion of Human Genome Project. The purpose of this chapter is to review some well-known algorithms that facilitate genome analysis. The material is presented in a way that is interesting to both the specialists working in this area and others. Thus, this review includes a brief sketch of the algorithms to facilitate a deeper understanding of the concepts involved. The list of problems related to genomics is very extensive; hence, the scope of this chapter is restricted to the following two related important problems: (1) phylogenetic analysis and (2) multiple sequence alignment. Readers interested in algorithms used in other fields of computational biology are recommended to refer to reviews by Abbas and Holmes [1] and Blazewicz et al. [7].

Genome refers to the complete DNA sequence contained in the cell. The DNA sequence consists of the four nucleotides adenine (A), thymine (T), cytosine (C), and guanine (G). Associated with each DNA strand (sequence) is a complementary DNA strand of the same length. The strands are complementary in that each nucleotide in one strand uniquely defines an associated nucleotide in the other: A and T are always paired, and C and G are always paired. Each pairing is referred to as a base pair; and bound complementary strands make up a DNA molecule. Typically, the number of base pairs in a DNA molecule is between thousands and billions, depending on the complexity of a given organism. For example, a bacterium contains about 600,000 base pairs, while human and mouse have some three billion base pairs. Among humans, 99.9% of base pairs are the same between any two unrelated persons. But that leaves millions of single-letter differences, which provide genetic variation between people.

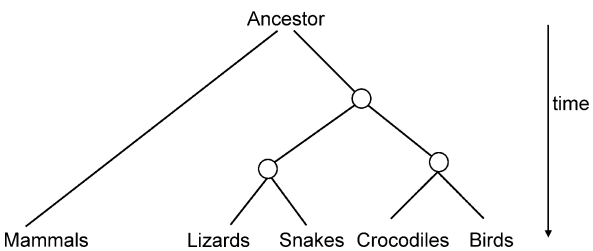
Understanding the DNA sequence is extremely important. It is considered as the blueprint for an organism's structure and function. The sequence order underlies all of life's diversity, even dictating whether an organism is human or another species such as yeast or a fruit fly. It helps in understanding the evolution of mankind, identifying genetic diseases, and creating new approaches for treating and controlling those diseases. In order to achieve these goals, research in genome analysis has progressed rapidly over the last decade.

The rest of this chapter is organized as follows. Section "Phylogenetic Analysis" discusses techniques used to infer the evolutionary history of species and Sect. "Multiple Sequence Alignment" presents the multiple sequence alignment problem and recent advances. In Sect. "Novel Graph-Theoretical Genomic Models", we describe our research effort for advancing genomic analysis through the design of a novel graph-theoretical approach for representing a wide variety of genomic sequence analysis problems within a single model. We summarize our theoretical findings, and present computational models based on two integer programming formulations. Finally, Sect. "Summary" summarizes the interdependence and the pivotal role played by the abovementioned two problems in computational biology.

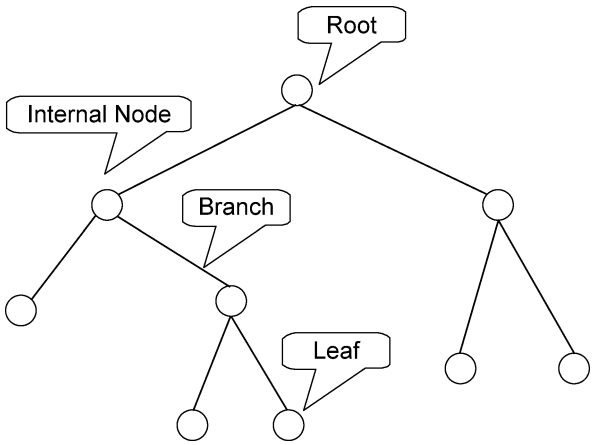
Phylogenetic Analysis

Phylogenetic analysis is a major aspect of genome research. It refers to the study of evolutionary relationships of a group of organisms. These hierarchical relationships among organisms arising through evolution are usually represented by a phylogenetic tree (Fig. 1). The idea of using trees to represent evolution dates back to Darwin. Both rooted and unrooted tree representations have been used in practice [17]. The branches of a tree represent the time of divergence and the root represents the ancestral sequence (Fig. 2).

The study of phylogenies and processes of evolution by the analysis of DNA or amino acid sequence data is



Algorithms for Genomic Analysis, Figure 1
An example of an evolutionary tree



Algorithms for Genomic Analysis, Figure 2
Tree terminology

called molecular phylogenetics. In this study, we will focus on methods that use DNA sequence data. There are two processes involved in inferring both rooted and unrooted trees. The first is estimating the branching structure or topology of the tree. The second is estimating the branch lengths for a given tree. Currently, there are wide varieties of methods available to conduct this analysis [16,19,55,79]. These available approaches can be classified into three broad groups: (1) distance methods; (2) parsimony methods; and (3) maximum likelihood methods. Below, we will discuss each of them in detail.

Methods Based on Pairwise Distance

In distance methods, an evolutionary distance d_{ij} is computed between each pair i, j of sequences, and a phylogenetic tree is constructed from these pairwise distances. There are many different ways of defining pairwise evolutionary distance used for this purpose. Most of the approaches estimate the number of nucleotide substitutions per site, but other measures have also been used [70,71]. The most popular one is the Jukes–Cantor distance [37], which defines d_{ij} as $-\frac{3}{4}\log(1-\frac{4f}{3})$, where f is the fraction of sites where nucleotides differ in the pairwise alignment [37].

There are a large number of distance methods for constructing evolutionary trees [78]. In this article, we discuss methods based on *cluster analysis* and *neighbor joining*.

Cluster Analysis: Unweighted Pair Group Method Using Arithmetic Averages The conceptually simplest and most known distance method is the unweighted pair group method using arithmetic averages (UPGMA) developed by Sokal and Michener [66]. Given a matrix of pairwise distances between each pair of sequences, it starts with assigning each sequence to its own cluster. The distances between the clusters are defined as $d_{ij} = \frac{1}{|C_i||C_j|}\sum_{p\in C_i, q\in C_j} d(p,q)$, where C_i and C_j denote sequences in clusters i and j, respectively. At each stage in the process, the least distant pair of clusters are merged to create a new cluster. This process continues until only one cluster is left. Given n sequences, the general schema of UPGMA is shown in Algorithm 1.

Algorithm 1 (UPGMA)

1. Input: Distance matrix d_{ij}, $1 \le i, j \le n$
2. **For** $i = 1$ to n **do**
3. Define singleton cluster C_i comprising of sequence i
4. Place cluster C_i as a tree leaf at height zero
5. **End for**
6. **Repeat**
7. Determine two clusters i, j such that d_{ij} is minimal.
8. Merge these two clusters to form a new cluster k having a distance from other clusters defined as the weighted average of the comprising two clusters. If C_k is the union of two clusters C_i and C_j, and if C_l is any other cluster, then $d_{kl} = \frac{d_{il}|C_i|+d_{jl}|C_j|}{|C_i|+|C_j|}$.
9. Define a node k at height $\frac{d_{ij}}{2}$ with daughter nodes i and j.
10. Until just a single cluster remains

The time and space complexity of UPGMA is $O(n^2)$, since there are $n-1$ iterations of complexity $O(n)$. A number of approaches have been developed which are motivated by UPGMA. Li [52] developed a similar approach which also makes corrections for unequal rates of evolution among lineages. Klotz and Blanken [43] presented a method where a present-day sequence serves as an ancestor in order to determine the tree regardless of the rates of evolution of the sequences involved.

Neighbor Joining Neighbor joining is another very popular algorithm based on pairwise distances [63]. This approach yields an unrooted tree and overcomes the assumption of the UPGMA method that the same rate of evolution applies to each branch.

Given a matrix of pairwise distances between each pair of sequences d_{ij}, it first defines the modified distance matrix $\bar{d}_{ij}$. This matrix is calculated by subtracting average distances to all other sequences from the d_{ij}, thus compensating for long edges. In each stage, the two nearest nodes (minimal $\bar{d}_{ij}$) of the tree are chosen and defined as neighbors in the tree. This is done recursively until all of the nodes are paired together.

Given n sequences, the general schema of neighbor joining is shown in Algorithm 2.

Algorithm 2 (Neighbor joining)

1. Input: Distance matrix d_{ij}, $\quad 1 \leq i, j \leq n$
2. **For** $i = 1$ to n
3. Assign sequence i to the set of leaf nodes of the tree (T)
4. **End for**
5. Set list of active nodes (L) = T
6. **Repeat**
7. Calculate the modified distance matrix $\bar{d}_{ij} = d_{ij} - (r_i + r_j)$, where $r_i = \frac{1}{|L|-2} \sum_{k \in L} d_{ik}$
8. Find the pair i, j in L having the minimal value of $\bar{d}_{ij}$
9. Define a new node u and set $d_{uk} = \frac{1}{2}(d_{ik} + d_{jk} - d_{ij})$, for all k in L
10. Add u to T joining nodes i, j with edges of length given by: $d_{iu} = \frac{1}{2}(d_{ij} + r_i - r_j)$, $d_{ju} = d_{ij} - d_{iu}$
11. Remove i and j from L and add u
12. **Until** only two nodes remain in L
13. Connect remaining two nodes i and j by a branch of length d_{ij}

Neighbor joining has a execution time of $O(n^2)$, like UPGMA. It has given extremely good results in practice and is computationally efficient [63,72]. Many practitioners have developed algorithms based on this approach. Gascuel [24] improved the neighbor-joining approach by using a simple first-order model of the variances and covariances of evolutionary distance estimates. Bruno et al. [10] developed a weighted neighbor joining using a likelihood-based approach. Goeffon et al. [25] investigated a local search algorithm under the maximum parsimony criterion by introducing a new subtree swapping neighborhood with an effective array-based tree representation.

Parsimony Methods

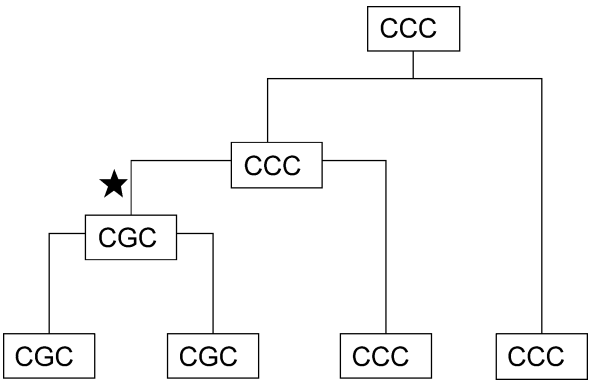
In science, notion of parsimony refers to the preference of simpler hypotheses over complicated ones. In the parsimony approach for tree building, the goal is to identify the phylogeny that requires the fewest necessary changes to explain the differences among the observed sequences. Of the existing numerical approaches for reconstructing ancestral relationships directly from sequence data, this approach is the most popular one. Unlike distance-based methods which build trees, it evaluates all possible trees and gives each a score based on the number of evolutionary changes that are needed to explain the observed sequences. The most parsimonious tree is the one that requires the fewest evolutionary changes for all sequences to derive from a common ancestor [69]. As an example, consider the trees in Fig. 3 and Fig. 4. The tree in Fig. 3 requires only one evolutionary change (marked by the star) compared with the tree in Fig. 4, which requires two changes. Thus, Fig. 3 shows the more parsimonious tree.

There are two distinct components in parsimony methods: given a labeled tree, determine the score; determine global minimum score by evaluating all possible trees, as discussed below.

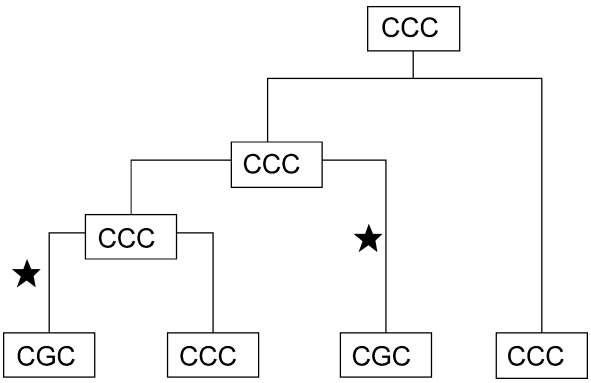
Score Computation Given a set of nucleotide sequences, parsimony methods treat each site (position) independently. The algorithm evaluates the score at each position and then sums them up over all the positions. As an example, suppose we have the following three aligned nucleotide sequences:

CCC
GGC
CGC

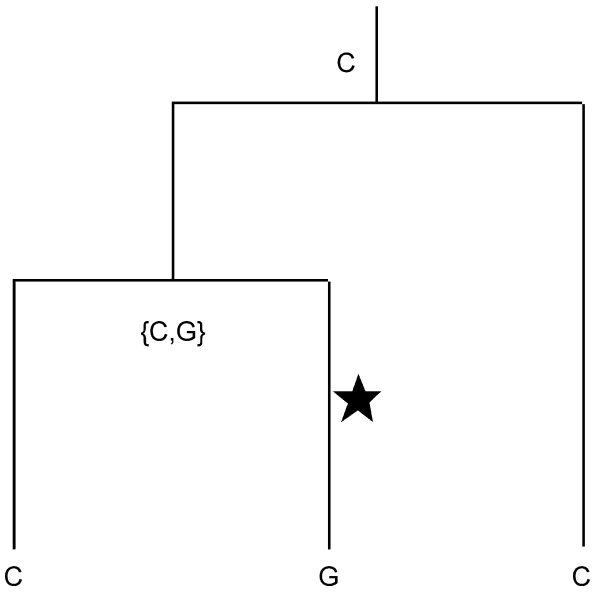
Then, for a given tree topology, we would calculate the minimal number of changes required at each of the three sites and then sum them up. Here, we investigate a traditional parsimony algorithm developed by Fitch [21], where the number of substitutions required is taken as a score. For a particular topology, this ap-



Algorithms for Genomic Analysis, Figure 3
Parsimony tree 1



Algorithms for Genomic Analysis, Figure 4
Parsimony tree 2



Algorithms for Genomic Analysis, Figure 5
The sets R_k for the first site of given three sequences

proach starts by placing nucleotides at the leaves and traverses toward the root of the tree. At each node, the nucleotides common to all of the descendant nodes are placed. If this set is empty then the union set is placed at this node. This continues until the root of the tree is reached. The number of union sets { equals} the number of substitutions required.

The general scheme for every position is shown in Algorithm 3.

Algorithm 3 (Parsimony: score computation)

1. Each leaf l is labeled with set R_l having observed nucleotide at that position.
2. Score $S = 0$
3. **For all** internal nodes k with children i and j having labels R_i and R_j **do**
4. $R_k = R_i \bigcap R_j$
5. **if** $R_k = \emptyset$ **then**
6. $R_k = R_i \bigcup R_j$
7. $S = S + 1$
8. **end if**
9. **End for**
10. Minimal score $= S$

Figure 5 shows the set R_k obtained by Algorithm 3. The computation is done for the first site of the three sequences shown above. The minimal score given by the algorithm is 1.

A wide variety of approaches have been developed by modifying Fitch's algorithm [68]. Sankoff and Cedergren [64] presented a generalized parsimony method which does not just count the number of substitutions, but also assigns a weighted cost for each substitution.

Ronquist [62] improved the computational time by including strategies for rapid evaluation of tree lengths and increasing the exhaustiveness of branch swapping while searching topologies.

Search of Possible Tree Topologies The number of possible tree topologies dramatically increases with the number of sequences. Consequently, in practice usu-

ally only a subset of them are examined using efficient search strategies. The most commonly used strategy is branch and bound methods to select branching patterns [60]. For large-scale problems, heuristic methods are typically used [69]. These exact and heuristic tree search strategies are implemented in various programs like PHYLIP (phylogeny inference package) and MEGA (molecular evolutionary genetic analysis) [20,47].

Maximum Likelihood Methods

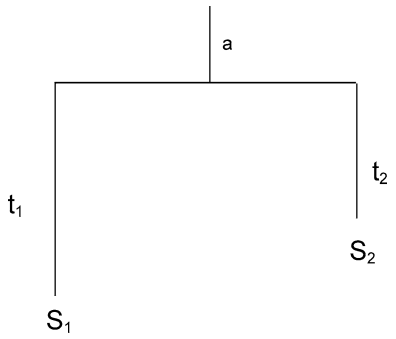
The method of maximum likelihood is one of the most popular statistical tools used in practice. In molecular phylogenetics, maximum likelihood methods find the tree that has the highest probability of generating observed sequences, given an explicit model of evolution. The method was first introduced by Felenstein [18]. We discuss herein both the evolution models and the calculation of tree likelihood.

Model of Evolution A model of evolution refers to various events like mutation, which changes one sequence to another over a period of time. It is required to determine the probability of a sequence S_2 arising from an ancestral sequence S_1 over a period of time t. Various sophisticated models of evolution have been suggested, but simple models like the Jukes–Cantor model are preferred in maximum likelihood methods.

The Jukes–Cantor [37] model assumes that all nucleotides (A, C, T, G) undergo mutation with equal probability, and change to all of the other three possible nucleotides with the same probability. If the mutation rate is 3α per unit time per site, the mutation matrix P_{ij} (probability that nucleotide i changes to nucleotide j in unit time) takes the form

$$\begin{pmatrix} 1-3\alpha & \alpha & \alpha & \alpha \\ \alpha & 1-3\alpha & \alpha & \alpha \\ \alpha & \alpha & 1-3\alpha & \alpha \\ \alpha & \alpha & \alpha & 1-3\alpha \end{pmatrix}.$$

The above matrix is integrated to evaluate mutation rates over time t and is then used to calculate $P(nt_2|nt_1, t)$, defined as the probability of nucleotide nt_1 being substituted by nucleotide nt_2 over time t.



Algorithms for Genomic Analysis, Figure 6
A simple tree

Various other evolution models like the Kimura model have also been mentioned in the literature [9,42].

Likelihood of a Tree The likelihood of a tree is calculated as the probability of observing a set of sequences given the tree.

$$L(\text{tree}) = \text{probability}[\text{sequences}|\text{tree}]$$

We begin with the simple case of two sequences S^1 and S^2 of length n having a common ancestor **a** as shown in Fig. 6. It is assumed that all different sites (positions) evolve independently, and thus the total likelihood is calculated as the product of the likelihood of all sites [15]. Here, the likelihood of each site is obtained using substitution probabilities based on an evolution model.

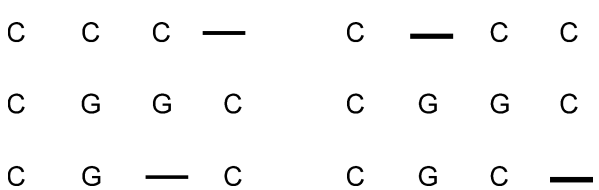
Given q_a is the equilibrium distribution of nucleotide a, the likelihood for the simple tree in Fig. 6 is calculated as $L(\text{tree}) = P(S^1, S^2) = \prod_{i=1}^{n} P(S_i^1, S_i^2)$, where $P(S_i^1, S_i^2) = \sum_a q_a P(S_i^1|a) P(S_i^2|a)$. To generalize this approach for m sequences, it is assumed that diverged sequences evolve independently after diverging. Hence, the likelihood for every node in a tree depends only on its immediate ancestral node and a recursive procedure is used to evaluate the likelihood of the tree. The conditional likelihood $L_{k,a}$ is defined as the likelihood of the subtree rooted at node k, given that the nucleotide at node k is a. The general schema for every site is shown in Algorithm 4. The likelihood is then maximized over all possible tree topologies and branch lengths.

Algorithm 4 (Likelihood: computation at given site)

1. **For all** leaf l **do**
2. **if** leaf has nucleotide a at that site **then**
3. $L_{l,a} = 1$
4. **else**
5. $L_{l,a} = 0$
6. **end if**
7. **End for**
8. **For all** internal nodes k with children i and j
9. define the conditional likelihood
$L_{k,a} = \sum_{b,c}[P(b|a)L_{i,b}][P(c|a)L_{j,c}]$
10. **End for**
11. Likelihood at given site = $\sum_a q_a\, L_{root,a}$

Recent Improvements The maximum likelihood approach has received great attention owing to the existence of powerful statistical tools. It has been made more sophisticated using advance tree search algorithms, sequence evolution models, and statistical approaches. Yang [80] extended it to the case where the rate of nucleotide substitutions differ over sites. Huelsenbeck and Crandall [34] incorporated the improvements in substitution models. Piontkivska [59] evaluated the use of various substitution models in the maximum likelihood approach and inferred that simple models are comparable in terms of both efficiency and reliability with complex models.

The enormously large number of possible tree topologies, especially while working with a large number of sequences, makes this approach computationally intensive [72]. It has been proved that reconstructing the maximum likelihood tree is nondeterministic polynomial time hard (NP) hard even for certain approximations [14]. In order to reduce computational time, Guindon and Gascuel [31] developed a simple hill-climbing algorithm based on the maximum-likelihood principle that adjusts tree topology and branch lengths simultaneously. Recently, parallel computation has been used to address huge computational requirement. Stamatakis et al. [67] have used OpenMP–parallelization for symmetric multiprocessing machines and Keane et al. [39] developed a distributed platform for phylogeny reconstruction by maximum likelihood.



Algorithms for Genomic Analysis, Figure 7
Two possible alignments for given three sequences

Multiple Sequence Alignment

Multiple sequence alignment is arguably among the most studied and difficult problems in computational biology. It is a vital tool because it compactly represents conserved or variable features among the family members. Alignment also allows character-based analysis compared to distance-based analysis and thus helps to elucidate evolutionary relationships better. Consequently, it plays a pivotal role in a wide range of sequence analysis problems like identifying conserved motifs among given sequences, predicting secondary and tertiary structures of protein sequences, and molecular phylogenetic analysis. It is also used for sequence comparison to find the similarity of a new sequence with pre-existing ones. This helps in gathering information about the function and structure of newly found sequences from existing ones in databases like GenBank in the USA and EMBL in Europe.

The multiple sequence alignment problem can be stated formally as follows. Let $\sum$ be the alphabet and let $\hat{\sum} = \sum \bigcup \{-\}$, where "–" is a symbol to represent "gaps" in sequences. For DNA sequences, alphabet $\hat{\sum} = \{\text{A,T,C,G,}-\}$.

An alignment for N sequences $S_1, \ldots, S_N$ is given by a set $\hat{S} = \{S_1, \ldots, S_N\}$ over the alphabet $\hat{\sum}$ which satisfy the following two properties: (1) the strings in $\hat{S}$ are of the same length; (2) S_i can be obtained from $\hat{S}_i$ by removing the gaps. Thus, an alignment in which each string $\hat{S}_i$ has length K can be interpreted as an alignment matrix of N rows and K columns, where row i corresponds to sequence S_i. Alphabets that are placed into the same column of the alignment matrix are said to be aligned with each other.

Figure 7 shows two possible alignments for given three sequences: S_1 = CCC, S_2 = CGGC, and S_3 = CGC.

For two sequences, the optimal multiple sequence alignment is easily obtained using dynamic program-

ming (Needleman–Wunsch algorithm). Unfortunately, the problem becomes much harder for more than two sequences, and the optimal solution can be found only for a limited number of sequences of moderate length (approximately 100) [8]. Researchers have tried to solve it by generalizing the dynamic programming approach to a multidimensional space. However, this approach has huge time and memory requirements and thus cannot be used in practice even for small problems of five sequences of length 100 each. This algorithm has been improved by identifying the portion of hyperspace which does not contribute to the solution and excluding it from the computation [11]. But even this approach of Carrillo and Lipman implemented in the multiple sequence alignment program can only align up to ten sequences [53]. Although, Gupta et al. [32] improved the space and time usage of this approach, it cannot align large data sets. To reduce the huge time and memory expenses, a wide variety of heuristic approaches for multiple sequence alignment have been developed [56].

There are two components for finding the multiple sequence alignment: (1) searching over all the possible multiple alignments; (2) scoring each of them to find the best one.

The problem becomes more complex for remotely related homologous sequences, i. e., sequences which are not derived from a common ancestor [28]. Numerous approaches have been proposed, but the quest for an approach which is accurate and fast is continuing. It must be remembered that even the choice of sequences and calculating the score of alignment is a nontrivial task and is an active research field in itself.

Scoring Alignment

There is no unanimous way of characterizing an alignment as the correct one and the strategy depends on the biological context. Different alignments are possible and we never know for sure which alignment is correct. Thus, one scores every alignment according to an appropriate objective function and alignments with higher scores are deemed to be better. A typical alignment scoring scheme consists of the following steps.

Independent Columns The score of alignment is calculated in terms of columns of alignments. The individual columns are assumed to be independent and thus the total score of an alignment is a simple summation over column scores. Thus, the score for an alignment $\text{score}(A) = \sum_j \text{score}(A_j)$, where A_j is column j of the multiple alignment A. Now, the score for every column j is calculated as the "sum-of-pairs" function using the scoring matrices described below. The sum-of-pairs score for column A_j is obtained as $\text{score}(A_j) = \sum_{k<l} \text{score}(A_j^k, A_j^l)$, where A_j^k and A_j^l are nucleotides in column j of the alignment corresponding to sequences k and l, respectively. If the gap costs are linear, score(nucleotide, –) and score(–, nucleotide) will be the insertion cost. But, this approach would not differentiate between opening a gap and extending it. So, affine gap penalties are often used where gap opening and extension penalty are treated as two different parameters. The correct value of both of these parameters is a major concern since their values can be set only empirically [75]. Also most schemes used in practice score columns as the weighted sum of pairwise substitutions instead of just addition as described before. The weights are decided in accordance with the amount of independent information each sequence possesses [4].

Both the assumption of treating every column independently and using the sum-of-pairs score for the column have limitations. The problem increases as the number of sequences increases.

Scoring Matrices Any alignment can be obtained by performing three evolution operations: insertion, deletion, and substitution. It is assumed that all the different operations occur independently and thus the complete score is evaluated as the sum of scores from every operation. Insertion and deletion scores are calculated as either linear or affine gap penalty. Substitutions scores are stored as a substitution score matrix, which contains the score for every pair of nucleotides. Thus, these scores S(A,B) can be treated as the score of aligning nucleotide A with nucleotide B.

These substitution score matrices can be obtained in various ways. One could adopt an ad hoc approach of setting up a score matrix which produces good alignments for a given set of sequences. The second approach would be more fundamental and look into the physical and chemical properties of nucleotides. If two nucleotides have similar properties, they would be more likely to be substituted by one another. The third and

the most prominent one is a statistical approach where the maximum likelihood principle is used in conjunction with probabilistic models of evolution [3].

Alignment Approaches

The number of different approaches for the multiple sequence alignment problem has steadily increased over the last decade and thus being exhaustive will not be possible. In this chapter, we will emphasize the most widely used class of algorithms and the new emerging and most promising approaches:

1. Progressive alignment algorithms: The most widely used type of algorithm based on using pairwise alignment information of input sequences. It assumes that input sequences are phylogenetically related, and uses these relationships to guide the alignment [13].
2. Graph-based algorithms: A new trend where graph-based models are used to approach this problem.
3. Iterative alignment algorithms: Typically an alignment is produced and is then refined through a series of iterations until no more improvement can be made.

Progressive Algorithms

Progressive alignment constitutes one of the simplest and most effective ways for multiple alignment. This strategy was introduced by various researchers, like Waterman and Perlwitz [77]. Among all the progressive algorithms, ClustalW is the most famous one. It is a noniterative, deterministic algorithm that attempts to optimize the weighted sums-of-pairs with affine gap penalties [73].

The typical progressive algorithm scheme is as follows:

- Compute the distance between all pairs of given sequences by aligning them. The distances represent the divergence of each pair of sequences. These distances could be calculated by fast approximation methods or by slower but more precise methods like complete dynamic programming. Since for given N sequences $\frac{N(N-1)}{2}$ pairwise scores have to be calculated and the scores are used just for construction of a guide tree and not the alignment itself, it is desirable to use approximation methods like k tuple matches.
- Find a guide tree from the distance matrix. This is typically achieved using the clustering algorithms discussed in the construction of an evolutionary tree. Once again, since the aim is to get the alignment and not the tree itself, approximation methods are used to construct the evolution trees.
- Align sequences progressively according to the branching order in the guide tree. The basic idea is to start from the leaves of the guide tree and move toward its root and to use a series of pairwise alignments to align larger and larger groups of sequences. Some algorithms have only a single growing alignment to which every remaining sequence is aligned, whereas other approaches align a subgroup of sequences and then merge the alignments.

There are three main shortcomings of the progressive algorithms.

1. There does not exist an undisputable "best" way of ordering the given sequences.
2. Once a sequence has been aligned, that alignment will not be modified even if it conflicts with sequences added later in the process. Hence, the order in which sequences are added becomes crucial, and since there is no undisputed best way to order the sequences, this approach returns suboptimal solutions.
3. For a given set of n sequences, $\binom{n}{2}$ pairwise alignments are generated; but while computing the final multiple alignment, most of these algorithms use fewer than n pairwise alignments. Thus, the resulting multiple alignment agrees with only a small amount of information available in the data.

Therefore, there is a growing need for an algorithm to align extremely divergent sequences whose pairwise alignments are likely to be incorrect. In order to address all these issues, some techniques have been developed; while they are innovative, it is understandable that they have their own assumptions and drawbacks.

Graph-Based Algorithms

Over the last few years, the field of genomics has undergone evolutionary changes with a rapid increase in new solution strategies. The use of graph-based models is easily seen as one of the most emerging and far-reaching trends. Just and Vedova [38] used a relation between the facility location problem and sequence

alignment to prove the NP-hardness of multiple sequence alignment. In this section, we review the most prominent integer programming approaches for finding multiple sequence alignment.

Maximum-Weight Trace Kececioglu et al. [40] used a solution of the maximum trace problem to construct alignment. The algorithm starts by calculating all pairwise alignments and using them to find a trace. To achieve this, given n sequences, an input alignment graph $G = (V, E)$ is constructed. It is an n-partite graph whose vertex set V represents the characters of the given sequences and whos edge set E represents the pairs of characters matched in the pairwise alignments. The subset of matching in E realized by an alignment is called a trace.

Alignment graph $G = (V, E)$ is extended to a mixed graph $G' = (V, E, A)$ by adding arc set A which connects the characters of every sequence to the next character in the same sequence. The objective of the algorithm is to find the maximum weight trace by finding cycles termed as "critical mixed cycles" in graph G' such that they satisfy sequence alignment properties [61].

The integer programming model for this problem is formulated as

$$\text{Maximize} \sum_{e \in E} w_e x_e \tag{1}$$

$$\text{subject to} \sum_{e \in P \cap E} x_e = |E \cap P| - 1 \; \forall \text{ critical mixed cycles } P \text{ in } G' \,, \; x_e \in \{0, 1\} \text{ for all } e \in E \,. \tag{2}$$

An implementation of a branch-and-cut algorithm is used to solve the above problem. Various valid inequalities for the polytope are added as cuts, some of which are facet-defining. The algorithm is capable of giving an exact solution under the sum-of-pairs objective function with linear gap costs. Kececioglu et al. [40] have made a significant contribution by introducing a polyhedral approach capable of obtaining exact solutions for a subclass of multiple sequence alignment. However, this method has its own drawbacks like not being able to capture the order of insertions and deletions between two matchings and affine gap costs. Recently, Althaus et al. [2] proposed a general model using this approach in which arbitrary gap costs are allowed.

Minimum-Spanning Tree and Traveling Salesman Problem Shyu et al. [65] explored the use of minimum spanning trees to determine the order of sequences. The idea of the approach is to preserve the most informative distances among the set of given sequences. The criterion used is meaningful and capable of working better than the traditional criteria like those in sum-of-pairs. The algorithm itself is very efficient for practical usage, and can be easily implemented. However, it fails to address the issue of using all the information in pairwise alignments, since it only uses the score and not the pairwise alignments themselves. Moreover, this approach has all the drawbacks of the progressive strategy.

A similar approach was also developed by Korostensky and Gonnet [44] using the traveling salesman problem. In this technique, a circular sum measure is used instead of a sum-of-pairs score. The cities in the traveling salesman problem correspond to the sequences and the scores of pairwise alignment are taken as the distances. The problem is to find the longest tour where each sequence is visited exactly once [45].

Eulerian Path Approach Zhang and Waterman [81] proposed a new approach motivated by the Eulerian method for fragment assembly in DNA sequencing. In their work, a consensus sequence is found and later pairwise alignments are obtained between each input sequence and consensus sequence. Finally, multiple sequence alignment is obtained according to these pairwise alignments. The most significant advantage of this method is the linear time and memory cost for finding the consensus sequence. And, if the consensus sequence is the one closest to all given sequences, good quality alignment can be obtained in a reasonable amount of time. Once again, this approach suffers from the prominent drawback of the progressive strategy and issues in graph formation while finding the consensus sequence.

Iterative Algorithms

The main shortcoming of the progressive strategy is the failure to remove errors in the alignment, which are introduced early. The iterative algorithms are developed precisely to overcome this flaw. They are based on the idea of reconsidering and realigning previously aligned

sequences with the goal of improving the overall alignment score. Each modification step is an iteration to improve the quality of the alignment.

These available approaches can be classified into two broad categories: probabilistic iterative algorithms, and deterministic iterative algorithms. We will briefly discuss them below.

Probabilistic Algorithms We will discuss both the traditional probabilistic optimization approaches like the genetic algorithm and relatively recent approaches based on a Bayesian idea.

- *Simulated annealing and genetic algorithm*. Simulated annealing and the genetic algorithm are very popular stochastic methods for solving complex optimization problems. While they are often viewed as separate and competing paradigms, both of them are iterative algorithms which search for new solutions "near" to already known good solutions. The fundamental difference between simulated annealing and the genetic algorithm is that simulated annealing performs a local move only on one solution to create a new solution, whereas the genetic algorithm also creates solutions by combining information from two different solutions. The performance of simulated annealing and the genetic algorithm varies with the problem and representation used.
  The algorithms starts with an initial alignment and the alignment score is taken to be the objective function [57]. Various operations like mutation, insertion, and substitution constitute the local move which is used to a get new solution from existing ones. Flexibility in the scoring systems and the ability to correct for errors introduced during the early phase makes these approaches desirable [41].
- *Hidden Markov model and Gibbs sampler*. The hidden Markov model and the Gibbs sampler are relatively recent approaches which view multiple sequence alignment in a statistical context. Both of them use the central Bayesian idea of simultaneously maximizing the data and the model. The Gibbs sampler find motifs using local alignment techniques [49]. It is essentially similar to the hidden Markov model with no insert and delete states.
  The hidden Markov model is a statistical model based on the Markov process, which has gained importance in various fields related to pattern recognition. It determines the hidden parameters of the system on the basis of the observable parameters of the model. For multiple sequence alignment, the hidden Markov model consists of three types of states: match states, insert states, and delete states [46]. Each state has its own emission probability of nucleotides and transition probability to other states. The standard expectation-maximization algorithm or gradient descent algorithms are used to train the model and evaluate the parameters.
  Although the hidden Markov model has been successfully used in other areas, it faces a lot of challenges. There need to be some minimum number of sequences (approximately 50) required to train the model and the hidden Markov model can be easily trapped in local optima like other hill-climbing approaches [35].

Deterministic Algorithms A deterministic iterative algorithm starts with an initial alignment and then attempts to improve it. This helps in overcoming the drawback of a progressive alignment strategy where partial alignments are "frozen" [6]. A typical scheme is as follows:

- Given N sequences $S_1, S_2, \ldots, S_N$, find alignment A.
- Remove sequence S_1 from alignment A and realign it to the profile of other aligned sequences $S_2, \ldots, S_N$ to get new alignment A'.
- Calculate the score of the new alignment A' and if it is better replace A by A'.
- Remove sequence S_2 from A' and realign it. Continue this procedure for $S_3, \ldots, S_N$.
- Repeat the realignment steps until the alignment score converges or the number of iterations reaches the user-specified limit.

Many iteration strategies which enable very accurate alignments have been developed [76]. The aim is to reduce the greedy nature of the algorithm and avoid getting trapped in a local optimum. One approach is to remove and realign every sequence to the rest in each iteration. Then, the alignment with the best score is taken to be the input for the next iteration. The other famous approach is to randomly split a set of sequences into two sets, which are then realigned.

Some researchers have incorporated the iterative strategy in the progressive alignment procedure itself. For instance, a double iteration loop has been

used to make the alignment, guide tree, and sequence weights mutually consistent [27]. Recently, Chakrabarti et al. [12] developed an approach which provides a fast and accurate method for refining existing block-based alignments.

Novel Graph-Theoretical Genomic Models

In this section, we present our research effort for a novel graph-theoretical approach for representing a wide variety of genomic sequence analysis problems within a single model [50]. The model allows incorporation of the operations "insertion," "deletion," and "substitution," and various parameters such as relative distances and weights. Conceptually, we refer the problem as the *minimum weight common mutated sequence* (MWCMS) problem. The MWCMS model has many applications, including the multiple sequence alignment problem, phylogenetic analysis, the DNA sequencing problem, and the sequence comparison problem, which encompass a core set of very difficult problems in computational biology. Thus, the model presented in this section lays out a mathematical modeling framework that allows one to investigate theoretical and computational issues, and to forge new advances for these distinct, but related problems.

DNA sequencing refers to determining the exact order of nucleotide sequences in a segment of DNA. This was the greatest technical challenge in the Human Genome Project. Achieving this goal has helped reveal the estimated 30,000 human genes that are the basic physical and functional units of heredity. The resulting DNA sequence maps are being used by scientists to explore human biology and other complex phenomena.

The structure of a DNA strand (sequence) is determined by experimentation. Typically, short sequences are determined to be in the strand, and the short sequences identified are then "connected" to form a long sequence. Recent advances attempting to identify DNA strand structure involve sequencing by hybridization [5,36]. Sequencing by hybridization is the process where every possible sequence of length n (4^n possibilities) is compared with a full DNA strand. Practical values for n are 8–12. Each short string either binds or does not bind to the full strand. Biologists can thus determine exactly which short strings are contained in the DNA strand and which are not.

However, the experiment does not identify the exact location of each short string in the full strand. Hence, an important issue involves how these short strings are connected together to form the complete strand. This problem can be viewed as a shortest common superstring problem and has been studied extensively [22,23,54]. Unfortunately, errors may arise during sequencing experiments. Three types of errors are deletions (a letter appears in an input string that should not be in the final sequence), insertions (a letter is missing from an input string), and substitutions (a letter in an input string should be substituted with another letter). *The MWCMS problem can be used to model and solve this shortest common superstring problem while addressing the issue of possible errors.*

Sequence comparison is one of the most crucial problems faced by researchers in the area of bioinformatics. The sequence patterns are conserved during evolution. Given a new sequence, it will be of interest to understand how much similarity it has with pre-existing sequences. Significant similarity between two sequences implies similarities in their structures and/or functions. There are lots of DNA databases containing DNA sequences and their functions. The major ones are GenBank in the USA and the EMBL data library in Europe. If one finds a new sequence similar to existing ones in these databases, one can transfer information about the function and structure [78]. Hence, an algorithm for sequence comparison which is efficient for a large number of sequences will play a pivotal role in rapid sequence analysis. The MWCMS problem can be used to address this issue.

Definitions

Our motivation for first defining the problem arose from the desire to help quantify the concept of the "best" representative sequence in the evolutionary distance problem. The evolutionary distance problem involves finding the DNA sequence of the most likely ancestor associated with a given set of DNA sequences from distinct but similar organisms. In other words, find the DNA strand that best represents a possible ancestor, if each of the organisms evolved from the same ancestor. Changes that contribute to differences between the given sequences and the ancestor are referred to as insertions, deletions, and substitutions.

These operations account for both evolutionary mutations and experimental errors in sequencing. Mathematically, given two sequences S and B, let ord(S, B) be an ordered collection of insertions, deletions, and substitutions to convert sequence S to sequence B. (For any two sequences S and B, there are an infinite number of collections ord(S, B).) Let $w(\text{ord}(S, B))$ be the weight of the conversion from S to B, where the weight is the sum of an expression involving values η, δ, and $\psi \in \Re^+$ which represent the weights associated with a single insertion, deletion, and substitution, respectively. Let $\text{ord}^*(S, B)$ be such that $w(\text{ord}^*(S, B)) \leq w(\text{ord}(S, B))$ for all ord(S, B). Define $d(S, B) = w(\text{ord}^*(S, B))$. Formally, the MWCMS problem can be stated as follows: Given positive weights η, δ, and ψ corresponding to a single insertion, deletion, and substitution respectively, a positive threshold κ, and finite sequences $S_1, \ldots, S_m$ from a finite alphabet, does there exist a sequence B such that $\sum_{i=1}^{m} d(S_i, B) \leq \kappa$?

We have defined the MWCMS problem—which incorporates the notions of insertion, deletion, and substitution—to help quantify the concept of the "best" representative sequence in the evolutionary distance problem. We now define precisely the operations of *insertion*, *deletion*, and *substitution*. Let $S = \{s_1, \ldots, s_n\}$ be a finite sequence of letters from a finite alphabet:

1. An *insertion* of an element x in position i of the sequence S is characterized by the addition of x between elements s_i and s_{i+1}. An insertion carries an associated penalty cost of η.
2. A *deletion* of an element in position i of S amounts to deleting s_i from the sequence S. The penalty for deletion is represented by δ.
3. A *substitution* of an element in position i of S amounts to replacing s_i with another letter from the alphabet. The penalty for substitution is represented by ψ.

We remark that a penalty cost for an operation could, more generally, depend on the position where the operation is performed and/or the element to be inserted/deleted/substituted.

Let $S_1 = \{s_{11}, \ldots, s_{1m}\}$ and $S_2 = \{s_{21}, \ldots, s_{2n}\}$ be two finite sequences of letters from a finite alphabet $\sum$. We say that the *relative distance* between elements s_{1i} and s_{2j} is k if $|i - j| = k$. We define a *k-restrictive* bipartite graph as a graph $G_k = (V_1, V_2, E_k)$ such that the nodes in V_1 and V_2 correspond, respectively, to each of the elements from the first and the second sequences. We assume the nodes in V_i are ordered in the same order as they appear in the sequence S_i. There is an edge between nodes $u \in V_1$ and $v \in V_2$ if u and v are identical (i. e., the same letter of the alphabet $\sum$) and if the relative distance between these two elements is less than or equal to k. The problem of identifying the "greatest similarity" between these two sequences can then be approached as the problem of finding a maximum cardinality matching between the associated node sets, subject to restrictions on which matchings are allowed. In particular, one must take into consideration the ordering of nodes so as to preserve the relative occurrence of the elements in the matching. In addition, matchings that have edge crossings must be prevented. When $k = \max\{|S_1|, |S_2|\} - 1$, we denote the graph by $G = (V_1, V_2, E)$, and the problem is equivalent to the well-studied longest common subsequence problem for two sequences, which is polynomial time solvable [23].

Construction of a Conflict Graph from Paths of Multiple Sequences

Let S_i, $i = 1, \ldots, m$, be a collection of finite sequences, each of length n, over a common alphabet $\sum$. Let $G_k = (V_1, \ldots V_m, E_1, E_2, \ldots, E_{m-1})$ be the *k-restrictive multilayer* graph in which each element in S_i forms a distinct node in V_i. Assume the nodes in V_i are ordered in the same order as they appear in the sequence S_i. E_i denotes the set of edges between nodes in V_i and V_{i+1}. There is an edge between nodes $u \in V_i$ and $v \in V_{i+1}$ if and only if u and v are the same letter in the alphabet $\sum$, and the relative distance between them is less than or equal to k. The multiple sequence comparison problem involves finding the longest common subsequence within the sequences S_i, $i = 1, \ldots, m$. We call a path $P = p_1, p_2, \ldots, p_m$ a *complete* path in G_k if $p_i \in V_i$ and $p_i p_{i+1} \in E_i$. Two complete paths are said to be *parallel* if their node sets are disjoint and the edges do not cross. Hence, a set of parallel complete paths in G_k corresponds to a feasible solution to longest common subsequence problem on the collection of sequences $S_i, i = 1, \ldots, m$. We say that two complete paths P_1 and P_2 *cross* if they are not parallel. We remark that the longest common subsequence problem with the number of sequences bounded,is polynomial time

solvable using dynamic programming [23]. In general, the problem remains NP-complete.

We can incorporate insertions by generating new paths which include inserted nodes on various layers. The weight for such a new path will be affected by the total number of insertions in the path. In particular, if L is a common subsequence for S_i and $|S_i| = n$ for all $i = 1, \ldots, m$, then the total number of unmatched elements remaining will be $m(n - |L|)$. These elements can be deleted completely, or for a given unmatched element, one can increase the size of L by 1 by appropriately inserting this element into various sequences. By doing so, one decreases the number of unmatched elements. Let l be the number of insertions needed to generate a new complete path. Then the number of unmatched elements will decrease by $m - l$. If we assume that at the end of the sequencing process all unmatched elements will be deleted, then the penalty for generating this new complete path will be given by $l\eta - (m - l)\delta$.

We next define the concept of a conflict graph relative to the complete paths in G_k.

Definition 1 Let $\mathcal{P} = \{P_1, \ldots, P_s\}$ be a finite collection of complete paths in G_k. The *conflict graph* $C_{\mathcal{P}} = (V_{\mathcal{P}}, E_{\mathcal{P}})$ associated with $\mathcal{P}$ is constructed as follows:

- $V_{\mathcal{P}} = \{P_1, \ldots, P_s\}$;
- there is an edge between two nodes P_i and P_j in $V_{\mathcal{P}}$ if and only if P_i and P_j cross each other.

This definition applies to any multilayer graph in general. Note that any stable set of nodes in $C_{\mathcal{P}}$ corresponds to a set of parallel complete paths for G_k, and thereby to a feasible solution to the longest common subsequence problem on the collection of sequences S_i, $i = 1, \ldots, m$.

We remark that when $m = 2$, the resulting conflict graph is weakly triangulated, and thus is perfect. For $m > 2$, the conflict graph can contain an antihole of size 6. However, these complete paths can be viewed as continuous functions on the interval from 0 to 1; thus, by construction, $C_{\mathcal{P}}$ is perfect [26].

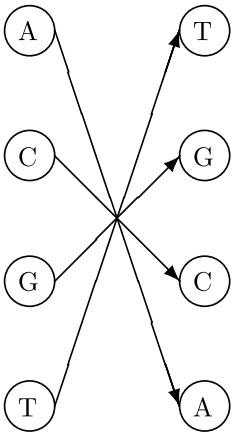
Complexity Theory

Recall that the notation $\text{ord}(S, B)$, $w(\text{ord}(S, B))$, $\text{ord}^*(S, B)$, and the formal definition of the MWCMS problem were given in Sect. "Definitions". As an optimization problem, the MWCMS problem can be stated as follows. Given a set of input sequences, the MWCMS problem seeks to mutate every input sequence to the same a priori unknown sequence using the operations of insertion, deletion, and substitution; weights are assigned for each operation, and the total weight associated with all mutations is to be minimized. Levenshtein [51] first considered a special case of this problem by changing a single input sequence to another sequence using insertions, deletions, and substitutions. Our study involves changing multiple input sequences to arrive at an a priori unknown common sequence.

Given positive weights η, δ, and ψ corresponding, respectively, to insertions, deletions, and substitutions and any two sequences S and B, clearly any $\text{ord}^*(S, B)$ will never contain more than $|B|$ insertions or substitutions. Proving that the MWCMS is in NP is not obvious. While one can transform the MWCMS to special applications (as described at beginning of Sect. "Novel Graph-Theoretical Genomic Models") to conclude that it is in NP, here we prove it directly for the general case. One needs to be able to evaluate $d(S, B)$ in polynomial time for any two sequences S and B. We next construct a graph that can be used to establish the existence of a polynomial-time algorithm for obtaining $d(S, B)$. The constructs and arguments used here typify those used to establish many of the results presented in this chapter. It is noteworthy that the notions of both conflict graph and perfect graph come into play.

Let $\sum$ be a finite alphabet, and define $\sum$-cross to be a directed bipartite graph consisting of $|\sum|$ vertices in each bipartition such that each vertex in the bipartition represents a distinct element in $\sum$. There is an arc between two vertices if the vertices correspond to the same element in $\sum$, and the geometric layout is rigidly constructed so that every arc crosses every other arc. This graph will be used as a "supernode" for insertion and substitution operations in our model. Figure 8 shows an example for $\sum$-cross when $\sum = \{\text{A,C,G,T}\}$.

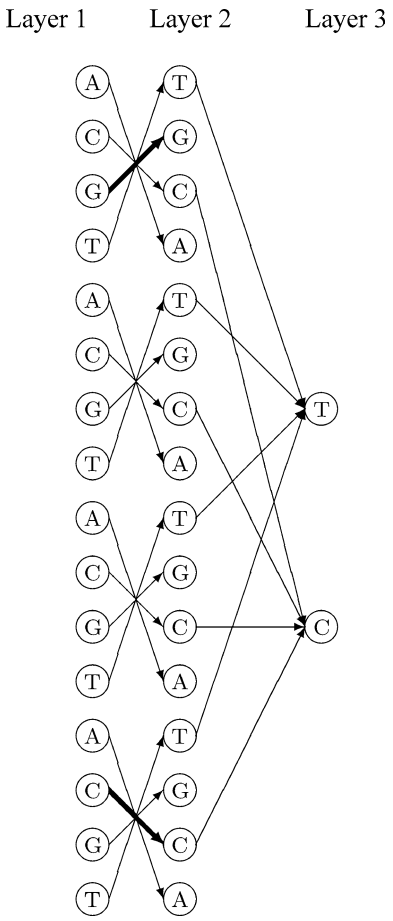
We now construct a *three-layer supergraph*, G_L, using the sequences S and B along with the $\sum$-cross graphs. Layers 1 and 2 consist of exactly $|B|(|S| + 1) + |S|$ $\sum$-crosses. The first $|B|$ $\sum$-crosses represent potential insertions before the first letter in S. The next $\sum$-cross represents either the first letter of S or a substitution of this letter. The next $|B|$ $\sum$-crosses represent potential insertions between the first and second letters of S. And this is followed by a $\sum$-cross rep-



Algorithms for Genomic Analysis, Figure 8
An example of $\sum$-cross when $\sum$ = {A, C, G, T}

resenting either the second letter of S or a substitution of this letter. This continues for each letter in S with the final $|B|$ $\sum$-crosses representing up to $|B|$ insertions after the last letter in S. Each $\sum$-cross is called either an *insertion supernode* or a *substitution supernode*, according to what it represents. The weight of all of the arcs in an insertion supernode is η. An arc in a substitution supernode has weight $-\delta$ if the arc represents the original letter in the sequences, or $\psi - \delta$ if the arc represents a substitution of the original letter. Layer 3 consists of the vertices represented by B. A vertex in layer 2 is connected to a vertex in layer 3 if they have the same letter. The weight of every arc between layers 2 and 3 is $M \leq -(\eta + \delta + \psi)$. A sample of a three-layer supergraph is given in Fig. 9. The bold arcs are used to denote the original letters in S (the weight of these arcs is $-\delta$). For simplicity, we omit the first two insertion supernodes before the first letter G. The first supernode thus represents the letter G from the original sequence, which allows for substitution. The second and third supernodes correspond to insertion supernodes, and the fourth supernode corresponds to the letter C and allows substitution as well. There are two more insertion supernodes which are omitted from the graph.

The main step in proving $d(S, B)$ to be polynomial time solvable for any sequences S and B involves the use of the conflict graph as defined in Definition 1. We state some preliminary theoretical results below. Detailed proofs can be found in Lee et al. [50].



Algorithms for Genomic Analysis, Figure 9
An example of the three-layer supergraph for converting the sequence S = GC to B = TC. *Bold arcs* are used to denote the original letters in S (the weight of these arcs is $-\delta$). For simplicity, we omit the first two insertion supernodes before the first letter G. The first supernode thus represents the letter G from the original sequence, which allows for substitution. The second and third supernodes correspond to insertions, and the fourth supernode corresponds to the letter C and allows substitution as well. There are two more insertion supernodes which are omitted from the graph

Lemma 1 *The following statements are equivalent:*

1. *There exists a conversion from S to B using no more than a total of $|B|$ insertions or substitutions.*
2. *There exist a set of noncrossing complete paths in the associated three-layer supergraph G_L of size $|B|$.*
3. *There exists a node packing of size $|B|$ in the associated conflict graph C.*

Lemma 2 *Calculating d(S, B) for any sequences S and B can be accomplished in polynomial time.*

The three-layer supergraph can be generalized to a multilayer supergraph when multiple sequences are considered. Clearly, such multilayer supergraphs are much too large for practical purposes, yet polynomiality is preserved in the construction, and it is therefore sufficient. We can now arrive at the result that the MWCMS is in NP.

Theorem 1 *The MWCMS is in NP.*

To prove that the MWCMS is polynomial time solvable when the number of input sequences is bounded by a positive constant, the following lemma is crucial, though trivial.

Lemma 3 *Given η, δ, $\psi \in \Re^+$, an optimal solution B to any MWCMS problem has the following properties. B has no substitutions from letters other than the original letters in S_i, and B will never have an element which is inserted in every sequence (in the same location). Therefore, there are at most $\sum_{i=1}^{m} |S_i|$ insertions in any sequence.*

In addition, we also require the construction of a (directed) $2m$-layer supergraph, G_L^m, similar to the three-layer supergraph, G_L.

Given sequences $S_1, \ldots, S_m$, generate a $2m$-layer (directed) graph $G_L^m = (V, E)$ as follows. Layers $2i-1$ and $2i$ consist of $(\sum_{j=1}^{m} |S_j|)(|S_i|+1) + |S_i|$ copies of $\sum$-crosses for $i = 1, \ldots, m$, constructed in exactly the same manner as layers 1 and 2 of the three-layer supergraph using the input sequence S_i. The first $\sum_{j=1}^{m} |S_j|$ $\sum$-crosses represent the possibility that $\sum_{j=1}^{m} |S_j|$ different letters can be inserted before the first element in S_i. The next $\sum$-cross corresponds to either the first letter in S_i or a substitution of this letter. This is repeated $|S_i|$ times (for each letter in S_i), and the final $\sum_{j=1}^{m} |S_j|$ $\sum$-crosses represent insertions after the final letter in S_i. Thus, the first $\sum_{j=1}^{m} |S_j|$ $\sum$-crosses represent the insertion supernodes, followed by one $\sum$-cross representing a letter in S_i or a substitution supernode, and so forth. An arc exists from a vertex in layer $2i$ to a vertex in layer $2i+1$ if the vertices correspond to the same letter. Observe that G_L^m is an acyclic directed graph which is polynomial in the size of the input sequences. Assign every arc between layers $2i$ and $2i+1$ a weight of 0. There are three different weights for arcs between layers $2i-1$ and $2i$ each corresponding to an insertion, deletion, or substitution. The assignment of weights on such arcs is analogous to the assignment in G_L: a weight of η is assigned to every arc contained in an insertion supernode; and an arc in a substitution supernode is assigned a weight of $-\delta$ if it corresponds to the original letter, or $\psi - \delta$, otherwise.

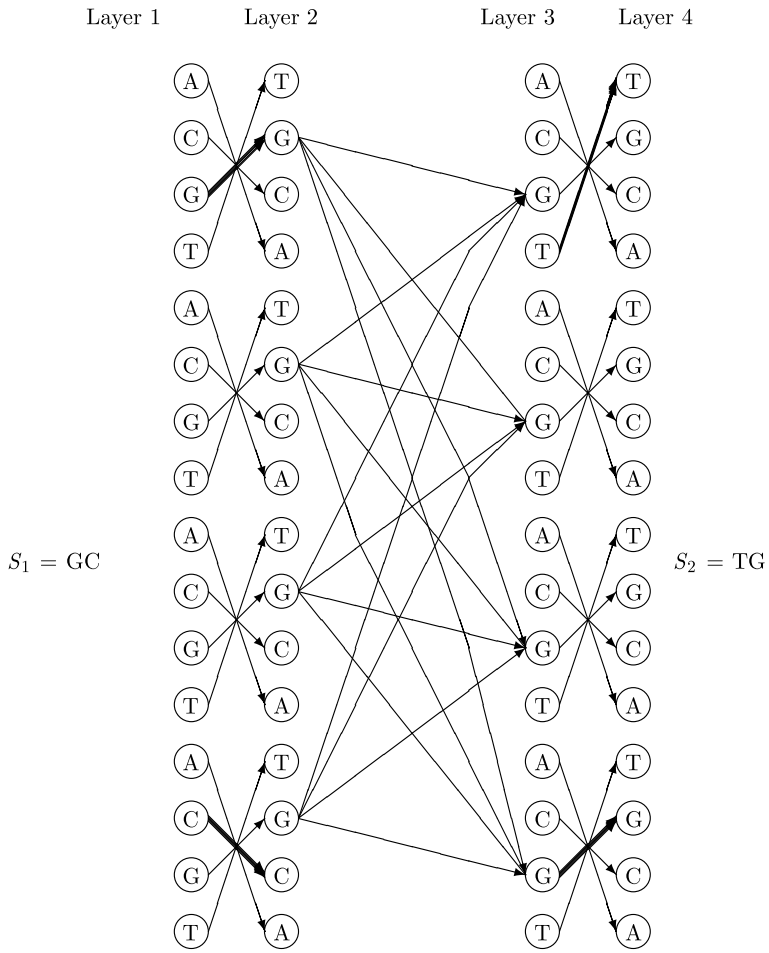
Figure 10 shows a sample graph for two sequences: $S_1 = \text{GC}$ and $S_2 = \text{TG}$. Observe that at most two insertions are needed in an optimal solution; thus, we can reduce the number of $\sum$-crosses as insertion supernodes from $\sum_{i=1}^{2} |S_i| = 4$ to 2. For simplicity, in the graph shown in Fig. 10, we have not included the two insertion supernodes before the first letter nor those after the last letter of each sequence. Thus, in the figure, the first $\sum$-cross represents the substitution supernode associated with the first letter in S_1. The second and third $\sum$-crosses represent two insertion supernodes. And the last $\sum$-cross represents the substitution supernode associated with the second letter in S_1. For simplicity, we include only arcs connecting vertices associated to the element G between layers 2 and 3. The arcs for other vertices follow similarly.

A conflict graph C associated with G_L^m can be generated by finding all complete paths (paths from layer 1 to layer $2m$) in G_L^m. These complete paths correspond to the set of vertices in C, as in Definition 1. If we assign a weight to each vertex equal to the weight of the associated complete path, then the following result can be established.

Theorem 2 *Every node packing in C represents a candidate solution to the MWCMS if and only if at most $\sum_{i=1}^{m} |S_i|$ letters can be inserted between any two original letters. Furthermore, the weight of the node packing is equal to the weight of the MWCMS $- \sum_{i=1}^{m} |S_i|\delta$.*

The supergraph G_L^m and its associated conflict graph are fundamental to our proof of the following theorem on the polynomial-time solvability of a restricted version of the MWCMS problem.

Theorem 3 *The MWCMS problem restricted to instances for which the number of sequences is bounded by a positive constant is polynomial time solvable.*



Algorithms for Genomic Analysis, Figure 10
A sample graph G_L^m of MWCMS with $S_1 = GC$ to $S_2 = \mathrm{TG}$, where $\sum = \{\mathrm{A,C,G,T}\}$

Special Cases of MWCMS

The MWCMS encompasses a very broad class of problems. In computational biology as discussed in this chapter, first and foremost, it represents a model for phylogenetic analysis. The MWCMS as defined is the "most likely ancestor problem," and the concept of the three-layer supergraph as described in Sect. "Complexity Theory" describes the evolutionary distance problem. An optimal solution to a multiple sequence alignment instance can be found using the solution of the MWCMS problem obtained on the $2m$-layer supergraph, G_L^m. The alignment is the character matrix obtained by placing together the given sequences incorporating the insertions into the solution of the MWCMS problem. Furthermore, DNA sequencing can be viewed as the shortest common superstring problem, while sequence comparison of a given sequence B to a collection of N sequences $S_1, \ldots, S_N$ is the MWCMS problem itself.

Broader than the computational biology applications, special cases of the MWCMS include shortest common supersequences, longest common subsequences, and shortest common superstring; these problems are of interest in their own right as combinatorial optimization problems and for their role in complexity theory.

Computational Models: Integer Programming Formulation

The construction of the multilayer supergraphs described in our theoretical study lays the foundation and provides direction for computational models and solution strategies that we will explore in future research. Although the theoretical results obtained are polynomial-time in nature, they present computational challenges. In many cases, calculating the worst-case scenario is not trivial. Furthermore, the polynomial-time result of a node-packing problem for a perfect graph by Grötschel et.al. [29,30] is existential in nature, and relies on the polynomial-time nature of the ellipsoid algorithm. The process itself involves solving an integer program relaxation multiple times. In our case, the variables of the integer program generated are the complete paths in the multilayer supergraph, G_L^m. Formally, the integer program corresponding to our conflict graph can be stated as follows.

Let x_p be the binary variable denoting the use or nonuse of the complete path p with weight w_p. Then the corresponding node-packing problem is

$$\begin{array}{lll} \text{Minimize} & \sum w_p x_p & \\ \text{subject to} & x_p + x_q \leq 1 & \text{if complete paths } p \text{ and } q \text{ cross} \\ & x_p \in \{0,1\} & \text{for all complete paths } p \text{ in } G_L^m . \end{array} \quad \text{(MIP1)}$$

We call the inequality $x_p + x_q \leq 1$ an adjacency constraint. A natural approach to improve the solution time for (MIP1) is to decrease the size of the graph G_L^m and thus the number of variables. Reductions in the size of G_L^m can be accomplished for shortest common superstrings, longest common subsequences, and shortest common supersequences. Among these three problems, the graph G_L^m is smallest for longest common subsequences. In longest common subsequences, all insertion and substitution supernodes can be eliminated.

Our theoretical results thus far rely on the creation of *all* complete paths. Clearly, the typical number of complete paths will be on the order of n^m, where $n = \max |S_i|$. In this case, an instance with three sequences and 300 letters in each sequence generates more than one million variables; hence, an exact formulation with all complete paths is impractical in general. A simultaneous column and row generation approach within a parallel implementation may lead to computational advances related to this formulation.

An alternative formulation can be obtained by examining G_L^m from a network perspective using arcs (instead of complete paths) in G_L^m as variables. Namely, let $x_{i,j}$ denote the use or nonuse of arc (i,j) in the final sequence, with $c_{i,j}$ the cost of the arc in G_L^m. The network formulation can be stated as

$$\begin{array}{ll} \text{Minimize} & \displaystyle\sum_{(i,j)\in E} c_{i,j} x_{i,j} \\ \text{subject to} & \displaystyle\sum_{i:(i,j)\in E} x_{i,j} = \sum_{k:(j,k)\in E} x_{j,k} \\ & \text{for all } j \in V \text{ in layers } 2,\ldots,2m-1 \\ & x_{i,j} + x_{k,l} \leq 1 \\ & \text{for all crossing arcs } (i,j) \text{ and } (k,l) \in E \\ & x_{i,j} \in \{0,1\} \\ & \text{for all } (i,j) \in E . \end{array} \quad \text{(MIP2)}$$

The first set of constraints ensures flow in equals flow out in all vertices contained in sequences $2,\ldots,m-1$ (complete paths). The second set of constraints ensures that no two arcs cross. This model grows linearly in the number of sequences. This alternative integer programming formulation is still large, but is manageable for even fairly large instances.

Utilizing a collection of DNA sequences (each with 40,000 base pairs in length) from a bacterium, and a collection of short sequences associated with genes found in breast cancer patients, computational tests of our graph-theoretical models are under way. We are seeking to develop computational strategies to provide reasonable running times for evolutionary distance problem instances derived from these data. In an initial test, when three sequences each with 100 letters are used, the initial linear program requires more than 10,000 s to provide a solution when tight constraints are employed (in this case, each adjacency constraint is replaced by a maximal clique constraint). Our ongoing computational effort will focus on developing and investigating solution techniques for practical problem instances, in-

cluding those based on the abovementioned two integer programming formulations, as well as development of fast heuristic procedures.

In [50], we outline a simple yet practical heuristic based on (MIP2) that we developed for solving the multiple sequence alignment problem; and we report on preliminary tests of the algorithm using different sets of sequence data. Motivation for the heuristic is derived from the desire to reduce computational time through various strategies for reducing the number of variables in (MIP2).

Summary

Multiple sequence alignment and phylogenetic analysis are deeply interconnected problems in computational biology. A good multiple alignment is crucial for reliable reconstruction of the phylogenetic tree [58]. On the other hand, most of the multiple alignment methods require a phylogenetic tree as the guide tree for progressive iteration.

Thus, the evolutionary tree construction might be biased by the guide tree used for obtaining the alignment. In order to avoid this pitfall, various algorithms have been developed which simultaneously find alignment and phylogenetic relationship among given sequences. Sankoff and Cedergren [64] developed a parsimony-based algorithm using a character-substitution model of gaps. The algorithm is guaranteed to find the evolutionary tree and alignment which minimizes tree-based parsimony cost. Hein [33] also developed a parsimony-type algorithm but used an affine gap cost, which is more realistic than the character-substitution gap model. This algorithm is also faster than Sankoff and Cedergreen's approach but makes simplifying assumptions in choosing ancestral sequences.

Like parsimony methods for finding a phylogenetic tree, both of the abovementioned approaches require a search over all possible trees to find the global optimum. This makes these algorithms computationally very intensive. Hence, there has been a strong focus on developing an efficient algorithm that considers both alignment and the tree. Vingron and Haeseler [74] have developed an approach based on three-way alignment of prealigned groups of sequences. It also allows change in the alignment made early in the course of computation. Many programs, like MEGA, are trying to develop an efficient integrated computing environment that allows both sequence alignment and evolutionary analysis [48].

We addressed this issue of simultaneously finding alignment and phylogenetic relationships by presenting a novel graph-theoretical approach. Indeed, our model can be easily tailored to find theoretically provable optimum solutions to a wide range of crucial sequence analysis problems. These sequence analysis problems were proven to be NP-hard, and thus understandably present computational challenges. In order to strike a balance between the time and the quality of the solution, a variety of parameters are provided. Ongoing research efforts are exploring the development of efficient computational models and solution strategies in a massive parallel environment.

Acknowledgement

This research was partially supported by grants from the National Science Foundation.

References

1. Abbas A, Holmes S (2004) Bioinformatics and management science: Some common tools and techniques. Oper Res 52(2):165–190
2. Althaus E, Caprara A, Lenhof H, Reinert K (2006) A branch-and-cut algorithm for multiple sequence alignment. Math Program 105(2-3):387–425
3. Altschul S (1991) Amino acid substitution matrices from an information theoretic perspective. J Mol Biol 219(3):555–565
4. Altschul SF, Carroll RJ, Lipman DJ (1989) Weights for data related by a tree. J Mol Biol 207(4):647–653
5. Bains W, Smith G (1988) A novel nethod for DNA sequence determination. J Theor Biol 135:303–307
6. Barton GJ, Sternberg MJE (1987) A strategy for the rapid multiple alignment of protein sequences: confidence levels from tertiary structure comparisons. J Mol Biol 198:327–337
7. Blazewicz J, Formanowicz P, Kasprzak M (2005) Selected combinatorial problems of computational biology. Eur J Oper Res 161:585–597
8. Bonizzoni P, Vedova G (2001) The complexity of multiple sequence alignment with SP-score that is a metric. Theor Comput Sci 259:63–79
9. Bos D, Posada D (2005) Using models of nucleotide evolution to build phylogenetic trees. Dev Comp Immunol 29(3):211–227

10. Bruno WJ, Socci ND, Halpern AL (2000) Weighted neighbor joining: A likelihood-based approach to distance-based phylogeny reconstruction. Mol Biol Evol 17:189–197
11. Carrillo H, Lipman D (1988) The multiple sequence alignment problem in biology. SIAM J Appl Math 48(5):1073–1082
12. Chakrabarti S, Lanczycki CJ, Panchenko AR, Przytycka TM, Thiessen PA, Bryant SH (2006) Refining multiple sequence alignments with conserved core regions. Nucleic Acids Res 34(9):2598–2606
13. Chenna R, Sugawara H, Koike T, Lopez R, Gibson TJ, Higgins DG, Thompson JD (2003) Multiple sequence alignment with the clustal series of programs. Nucleic Acids Res 31(13):3497–3500
14. Chor B, Tuller T (2005) Maximum likelihood of evolutionary trees: hardness and approximation. Bioinf 21(Suppl. 1):I97–I106
15. Clote P, Backofen R (2000) Computational Molecular Biology: An Introduction. Wiley, NY, USA
16. Delsuc F, Brinkmann H, Philippe H (2005) Phylogenomics and the reconstruction of the tree of life. Nature reviews. Genet 6(5):361–375
17. Durbin R, Eddy S, Krogh A, Mitchison G (1998) Biological Sequence Analysis. Cambridge University Press, UK
18. Felsenstein J (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. J Mol Evol 17(6): 368–376
19. Felsenstein J (1988) Phylogenies from molecular sequences: Inference and reliability. Annu Rev Genet 22:521–565
20. Felsenstein J (1989) PHYLIP – phylogeny inference package (version 3.2). Cladistics 5:164–166
21. Fitch WM (1971) Toward defining the course of evolution: Minimum change for a specific tree topology. Syst Zool 20(4):406–416
22. Gallant J, Maider D, Storer J (1980) On finding minimal length superstrings. J Comput Syst Sci 20:50–58
23. Garey M, Johnson D (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, San Francisco, USA
24. Gascuel O (1997) BIONJ: An improved version of the NJ algorithm based on a simple model of sequence data. Mol Biol Evol 14(7):685–695
25. Goeffon A, Richer J, Hao J (2005) Local search for the maximum parsimony problem. Lect Notes Comput Sci 3612:678–683
26. Golumbic MC, Rotem D, Urrutia J (1983) Comparability graphs and intersection graphs. Discret Math 43:37–46
27. Gotoh O (1996) Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. J Mol Biol 264(4):823–838
28. Gotoh O (1999) Multiple sequence alignment: algorithms and applications. Adv Biophys 36:159–206
29. Grötschel M, Lovász L, Schrijver A (1984) Polynomial algorithms for perfect graphs. Annals Discret Math 21:325–356
30. Grötschel M, Lovász L, Schrijver A (1988) Geometric algorithms and combinatorial optimization. Springer, New York
31. Guindon S, Gascuel O (2003) A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst Biol 52(5):696–704
32. Gupta S, Kececioglu J, Schaeffer A (1995) Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. J Comput Biol 2:459–472
33. Hein J (1989) A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when the phylogeny is given. Mol Biol Evol 6(6):649–668
34. Huelsenbeck J, Crandall K (1997) Phylogeny estimation and hypothesis testing using maximum likelihood. Annu Rev Ecol Syst 28:437–66
35. Hughey R, Krogh A (1996) Hidden markov models for sequence analysis: extension and analysis of the basic method. Comput Appl Biosci 12(2):95–107
36. Idury RM, Waterman MS (1995) A new algorithm for DNA sequence assembly. J Comput Biol 2(2):291–306
37. Jukes TH, Cantor CR (1969) Evolution of protein molecules. In: Munro HN (ed) Mammalian Protein Metabolism. Academic Press, New York, pp 21–123
38. Just W, Vedova G (2004) Multiple sequence alignment as a facility-location problem. INFORMS J Comput 16(4):430–440
39. Keane T, Naughton T, Travers S, McInerney J, McCormack G (2005) DPRml: distributed phylogeny reconstruction by maximum likelihood. Bioinf 21(7):969–974
40. Kececioglu J, Lenhof H, Mehlhorn K, Mutzel P, Reinert K, Vingron M (2000) A polyhedral approach to sequence alignment problems. Discret Appl Math 104:143–186
41. Kim J, Pramanik S, Chung MJ (1994) Multiple sequence alignment using simulated annealing. Bioinf 10(4):419–426
42. Kimura M (1980) A simple method for estimating evolutionary of base substitution through comparative studies of nucleotide sequences. J Mol Evol 16:111–120
43. Klotz L, Blanken R (1981) A practical method for calculating evolutionary trees from sequence data. J Theor Biol 91(2):261–272
44. Korostensky C, Gonnet GH (1999) Near optimal multiple sequence alignments using a traveling salesman problem approach. In: Proceedings of the String Processing and Information Retrieval Symposium. IEEE, Cancun, pp 105–114
45. Korostensky C, Gonnet GH (2000) Using traveling salesman problem algorithms for evolutionary tree construction. Bioinf 16(7):619–627
46. Krogh A, Brown M, Mian IS, Sjolander K, Haussler D (1994) Hidden markov models in computational biology: Applications to protein modeling. J Mol Biol 235:1501–1531

47. Kumar S, Tamura K, Nei M (1994) MEGA: Molecular evolutionary genetics analysis software for microcomputers. Comput Appl Biosci 10:189–191
48. Kumar S, Tamura K, Nei M (2004) MEGA3: integrated software for molecular evolutionary genetics analysis and sequence alignment. Brief Bioinform 5(2):150–163
49. Lawrence C, Altschul S, Boguski M, Liu J, Neuwald A, Wootton J (1993) Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. Science 262:208–214
50. Lee EK, Easton T, Gupta K (2006) Novel evolutionary models and applications to sequence alignment problems. Annals Oper Res 148(1):167–187
51. Levenshtein VL (1966) Binary codes capable of correcting deletions, insertions, and reversals. Cybern Control Theor 10(9):707–710
52. Li W (1981) Simple method for constructing phylogenetic trees from distance matrices. Proc Natl Acad Sci USA 78(2):1085–1089
53. Lipman D, Altschul S, Kececioglu J (1989) A tool for multiple sequence alignment. Proc Natl Acad Sci USA 86(12):4412–4415
54. Maier D, Storer JA (1977) A note on the complexity of the superstring problem. Technical Report 233, Princeton University, USA
55. Nei M (1996) Phylogenetic analysis in molecular evolutionary genetics. Annu Rev Genet 30:371–403
56. Notredame C (2002) Recent progress in multiple sequence alignment: a survey. Pharmacogenomics 3(1):131–144
57. Notredame C, Higgins D (1996) SAGA: sequence alignment by genetic algorithm. Nucleic Acids Res 24(8):1515–1524
58. Phillips A, Janies D, Wheeler W (2000) Multiple sequence alignment in phylogenetic analysis. Mol Phylogenet Evol 16(3):317–330
59. Piontkivska H (2004) Efficiencies of maximum likelihood methods of phylogenetic inferences when different substitution models are used. Mol Phylogenet Evol 31(3):865–873
60. Purdom P, Bradford PG, Tamura K, Kumar S (2000) Single column discrepancy and dynamic max-mini optimizations for quickly finding the most parsimonious evolutionary trees. Bioinformamtics 16:140–151
61. Reinert K, Lenhof H, Mutzel P, Mehlhorn K, Kececioglu J (1997) A branch-and-cut algorithm for multiple sequence alignment. In: Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB-97). ACM Press, Santa Fe, pp 241–249
62. Ronquist F (1998) Fast fitch-parsimony algorithms for large data sets. Cladistics 14:387–400
63. Saitou N, Nei M (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol 4:406–425
64. Sankoff D, Cedergren RJ (1983) Simultaneous comparison of three or more sequences related by a tree. In: Sankoff D, Kruskal JB (eds) Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison. Addison-Wesley, MA, USA, pp 253–264
65. Shyu SJ, Tsai YT, Lee R (2004) The minimal spanning tree preservation approaches for DNA multiple sequence alignment and evolutionary tree construction. J Comb Optim 8(4):453–468
66. Sokal R, Michener C (1958) A statistical method for evaluating systematic relationships. University of Kansas, Scientific Bull 38:1409–1438
67. Stamatakis A, Ott M, Ludwig T (2005) RAxML-OMP: An efficient program for phylogenetic inference on SMPs. Lect Notes Comput Sci 3606:288–302
68. Swofford DL, Maddison WP (1987) Reconstructing ancestral character states under wagner parsimony. Math Biosci 87:199–229
69. Swofford DL, Olsen GJ (1990) Phylogeny reconstruction. In: Hillis DM, Moritz G (eds) Molecular Systs. Sinauer Associates, MA, USA, pp 411–501
70. Tajima F, Nei M (1984) Estimation of evolutionary distance between nucleotide sequences. Mol Biol Evol 1(3):269–85
71. Tajima F, Takezaki N (1994) Estimation of evolutionary distance for reconstructing molecular phylogenetic trees. Mol Biol Evol 11:278–286
72. Takahashi K, Nei M (2000) Efficiencies of fast algorithms of phylogenetic inference under the criteria of maximum parsimony, minimum evolution, and maximum likelihood when a large number of sequences are used. Mol Biol Evol 17:1251–1258
73. Thompson JD, Higgins DG, Gibson TJ (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res 22(22):4673–4680
74. Vingron M, Haeseler A (1997) Towards integration of multiple alignment and phylogenetic tree construction. J Comput Biol 4(1):23–34
75. Vingron M, Waterman M (1994) Sequence alignment and penalty choice. review of concepts, case studies and implications. J Mol Biol 235(1):1–12
76. Wallace IM, O'Sullivan O, Higgins DG (2005) Evaluation of iterative alignment algorithms for multiple alignment. Bioinformatics 21(8):1408–14
77. Waterman M, Perlwitz M (1984) Line geometries for sequence comparisons. Bull Math Biol 46(4):567–577
78. Waterman MS (1995) Introduction to Computational Biology: Maps, Sequences and Genomes. Chapman and Hall
79. Whelan S, Lio P, Goldman N (2001) Molecular phylogenetics: state-of-the-art methods for looking into the past. Trends Genet 17(5):262–272
80. Yang Z (1993) Maximum-likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites. Mol Biol Evol 10(6):1396–401
81. Zhang Y, Waterman M (2003) An eulerian path approach to global multiple alignment for DNA sequences. J Comput Biol 10(6):803–819

Alignment Problem

CLAUDE G. DIDERICH[1], MARC GENGLER[2]
[1] Computer Sci. Department,
Swiss Federal Institute Technology-Lausanne,
Lausanne, Switzerland
[2] Ecole Sup. d'Ingénieurs de Luminy,
University Méditerrannée, Marseille, France

MSC2000: 05-02, 05-04, 15A04, 15A06, 68U99

Article Outline

Keywords
Alignment Problem
  Communication-Free Alignment Problem
  Constant-Degree Parallelism Alignment Problem
Solving the Alignment Problem
  Communication-Free Alignment Approaches
  Alignment Approaches Based
    on Generating HPF like Data Distributions
  Approaches Using a Graph Based Framework
  Approaches Using a Linear Algebra Framework
  Other Approaches
Conclusion
See also
References

Keywords

Alignment problem; Automatic parallelization; Computation and data mapping; Nested loops; Scheduling functions

Since the mid-1990s the need for techniques to parallelize numerical applications has increased. When parallelizing nested loops for distributed memory parallel computers, two major problems have to be solved: the scheduling of the loop iterations and the mapping of the computations and data elements onto the processors. The scheduling functions must satisfy all the data dependences existing in the sequential loop nests. The mapping functions should maximize the degree of parallelism obtained. Furthermore they should minimize the amount of communication overhead due to non local data references.

This survey presents the *alignment problem*, that is, the problem of mapping computation and data onto the processors. The alignment problem has been studied extensively since the beginning of the nineties, that is, since the beginning of the introduction of massively parallel distributed memory computers. For different sub-problems of the alignment problem, the most interesting results are surveyed.

Alignment Problem

The alignment problem is the problem of finding an alignment of loop iterations with the array elements accessed. This means computing mapping functions of the loop iterations, called computations, and mapping functions of the array elements, called data, to a multidimensional grid of virtual processors. The name of the problem comes from the idea of aligning the processors computing with the ones owning the data. The alignment problem is tightly related to the mapping of the computation and data objects onto a grid of virtual processors.

As input, programs containing nested loops are considered. Each loop nest may contain one or more instructions. For the sake of simplicity, only assignment instructions are considered. The data access functions are described by the functions F_l: $\mathbf{I}_j \rightarrow \mathbf{D}_K$, where $\mathbf{I}_j$ represents the iteration space surrounding instruction S_j and $\mathbf{D}_K$ the domain of the array K.

To solve the alignment problem, computation and data mapping functions C_j and D_K have to be computed such as to minimize the overall execution time of the resulting parallel program.

$$C_j : \mathbf{I}_j \rightarrow \mathbf{P},$$
$$D_K : \mathbf{D}_K \rightarrow \mathbf{P},$$

where $\mathbf{P}$ represents a multidimensional grid of virtual processors.

To minimize the overall execution time a solution to the alignment problem has to address the following needs:

i) maximize the degree of parallelism, that is, use as many dimensions of the virtual grid of processors as possible,
ii) minimize the need for non local data accesses, that is, distribute the array elements such that a minimal amount of communication overhead is required to

access data elements stored on different processors than the ones accessing them,

iii) guarantee the existence of scheduling functions compatible with the computation mapping functions.

Clearly the needs i)–iii) depend on each other. In this survey we only focus on the first two needs.

Need i) can be expressed by maximizing the dimension of the virtual processor grid **P** onto which the computations and data elements are mapped.

The need for a given data access F_l to be local is expressed by the equation (1) being satisfied:

$$C_j(\vec{\imath}) = D_K(F_l(\vec{\imath})). \tag{1}$$

Equation (1) is called *alignment constraint* or *locality constraint*. Depending on how the needs i) and ii) are satisfied, various subproblems of the alignment problem can be defined.

Communication-Free Alignment Problem

The *communication-free alignment problem* (CFAP) is the problem of finding computation and data mapping functions for each instruction and for each data array such that no communication is needed and the degree of parallelism obtained is maximal. The CFAP can be formulated as an optimization problem:

$$\begin{cases} \max_{C_j, D_K} & \text{dimension of } \mathbf{P} \\ \text{s.t.} & \forall j, l, K\colon\ C_j(\vec{\imath}) = D_K(F_l(\vec{\imath})). \end{cases}$$

Constant-Degree Parallelism Alignment Problem

Let $\mathcal{F}$ be the set of data access functions from a set of loop nests forming an alignment problem and d a positive constant. Let $c(\mathcal{F}', \mathcal{F})$ be a cost function on a subset $\mathcal{F}' \subseteq \mathcal{F}$ of data access functions. The *constant degree parallelism alignment problem* (CDPAP), denoted by $(\mathcal{F}, d)$, is the problem of finding a subset $\mathcal{F}' \subseteq \mathcal{F}$ of data access functions such that:

1) There exists a solution to the CFAP consisting of all data accesses in the set $\mathcal{F}'$ admitting a degree of parallelism of at least d.
2) The cost function $c(\mathcal{F}', \mathcal{F})$ on the subset $\mathcal{F}'$ is minimized.

As for the CFAP, the CDPAP can be formulated as follows as an optimization problem:

$$\begin{cases} \max_{C_j, D_K} & \sum_{j,l,K}[[C_j(\vec{\imath}) = D_K(F_l(\vec{\imath}))]] \\ \text{s.t.} & \text{dimension of } \mathbf{P} \geq d. \end{cases}$$

Example 1 The data accesses in this example are encoded by the three functions $F_1(i, j) = (i\, j + 1)$, $F_2(i, j) = (i{-}1\, j + 1)$ and $F_3(i, j) = (i + 1\, j + 1)$. A possible solution requiring no communication and admitting one degree of parallelism is given by $C(i, j) = j$ and $D_a(i, j) = j - 1$, **P** being a one-dimensional processor set.

```
DO i = 2, n − 1
    DO j = 2, n − 1
        a(i, j + 1) = a(i − 1, j + 1) + a(i + 1, j + 1)
    END DO
END DO
```

Solving the Alignment Problem

Communication-Free Alignment Approaches

C.-H. Huang and P. Sadayappan [17], in 1991, were the first to formulate the alignment problem in a linear algebra framework. They focus on a communication-free solution. The data array elements as well as the loop iterations are partitioned in disjoint sets represented by hyperplanes. Each set is mapped onto a different processor. The partitions are sought such that they result in the elimination of communication. A characterization of a necessary and sufficient condition for communication-free hyperplane partitioning is provided. Various results are given characterizing the situation where the iteration and data space can be partitioned along hyperplanes so that no communication is necessary. More precisely, two data elements accessed during a single iteration in a single instruction must be located on a single processor and two iterations in the same instruction accessing a single data element must be executed on the same processor.

In [30], a matrix notation is presented to describe array accesses in fully parallel loop nests. A sufficient condition on the matrices for computing a communication-free mapping of the arrays onto the processors is given. The owner computes rule is assumed for the computation mapping. The presented

existence condition for communication-free partitions is based on the connectivity of the data access graph which models the data access patterns. To compute data mapping functions, a set of systems of linear equations is constructed, one system of linear equations per pair of read and write data accesses. If there exists a solution to the set of systems of linear equations, then there exists a communication-free partitioning of the array elements into parallel hyperplanes.

In [2] a linear algebra approach is proposed, based on [17]. The communication-free alignment problem is solved by computing a basis of the null space of the application representing the alignment constraints. The problem of data replication is addressed.

In [6], T.-S. Chen and J.-P. Sheu consider perfect loop nests. They compute iteration and data space partitioning functions requiring no communication. Their work focuses only on uniformly generated data references. Sufficient conditions are given for the existence of a communication-free partition. The method for partitioning the data onto the processors is based on the computation of independent blocks called iteration and data partitions respectively. If no communication-free partitioning exists, data replication is considered.

In [24], an algorithm is presented that extracts all the degrees of communication-free parallelism that can be obtained via loop fission, fusion, interchange, reversal, skewing, scaling, re-indexing and statement reordering. The algorithm first assigns the iterations of the instructions in the program to processors via affine processor mapping functions. Then it generates the correct code by assuring that the semantics of the sequential program are satisfied.

Alignment Approaches Based on Generating HPF like Data Distributions

J. Li and M. Chen [22,23] are interested in the indices of the arrays that have to be aligned with one another to minimize remote data references. The techniques were initially developed for compiling the functional language 'Crystal', but can be applied in the process of compiling imperative languages like 'Fortran'. The parallelism is assumed to be specified explicitly and the single assignment form is used. The goal of their approach is to find alignment functions such that the dimensions of each array are projected onto the same space of a virtual processor grid. They consider four basic alignments:

i) permutations of the indices,
ii) embeddings,
iii) translations by a constant, and
iv) reflections.

To find a set of data accesses for which valid alignment functions exist, a component affinity graph is constructed. It represents the affinities between cross reference patterns. The nodes of the graph represent the components of the index domains to be aligned. An edge represents an affinity between the two corresponding domain components. The alignment problem then consists in partitioning the set of nodes of the component affinity graph into disjoint subsets with the restriction that no two nodes belonging to the same array are allowed in the same subset. A fast and quite efficient heuristic algorithm is presented.

M. Gupta, in his thesis in 1992 [16], presents a data distribution algorithm that operates in four passes. The first pass serves to compute an alignment of the array dimensions. The algorithm developed is based on the notion of component affinity graph introduced by Li and Chen [22]. In the second phase the arrays are partitioned using either block or cyclic data distributions. In the third pass, the block sizes of the arrays distributed are computed whereas the last pass computes the number of processors on which each array dimension is distributed.

K. Kunchithapadam and B.P. Miller [20], in opposition to other approaches, assume that a user-defined data distribution is given. The data accesses of a program are modeled by a colored proximity graph. Each vertex of the graph represents a part of an array and the color of a vertex represents the current processor to which this array part is assigned. Edges of the graph represent assignments of values arising from part of one or more arrays to part of another array assuming the owner computes rule for the computation mapping. Edges between vertices of different colors are assigned a weight representing the associated communication costs. The problem of improving a given set of data mapping functions is to find a sequence of color exchanges, that is, data redistributions, that minimize the weight of the graph, that is, the communication costs. A possible algorithm for solving this problem is presented.

B. Sinharoy and B.K. Szymanski [32] study the problem of finding computation and data alignment functions for regular iterative algorithms. A loop nest can be represented by a regular iterative algorithm if and only if all the data access functions are constant offset functions and the loop nest's instructions are in single assignment form. The communication cost function used is based on the distance of the processors exchanging data on the virtual processor grid. The authors show that finding computation and data mapping functions is equivalent to minimizing a sum of absolute values composed of sums. An exact enumeration algorithm is presented and a polynomial time algorithm for finding an approximate solution is described.

Approaches Using a Graph Based Framework

K. Knobe, J.D. Lukas and G.L. Steele Jr. [19] study the problem of aligning the array elements accessed amongst each other. They target their approach towards SIMD machines. Two different kinds of preferences are distinguished:

i) identity preferences representing alignment preferences due to different data accesses to the same array, and
ii) conformance preferences relating two different arrays.

To compute what preferences can be satisfied without loosing parallelism, a cyclic preference graph is constructed. Each data access is represented by a vertex and two vertices are related by an undirected weighted edge if there exists a preference between the two data accesses. The weight of each edge is defined by the loop depth at which the data accesses occur. Conflicts between preferences are represented by cycles in the cyclic preference graph. A heuristic, using a greedy approach, is presented to remove annoying cycles or to reduce the parallelism.

In [5] an intermediate representation of a program called the alignment-distribution graph is described. The *alignment-distribution graph* is a directed graph in which nodes represent communication and edges represent the data flow. It exposes the communication requirements of the program. The framework restricts the alignments computed to alignments in which each axis of an array maps to a different axis of an HPF like template and data elements are evenly spaced along the template axis. The alignments computed have three components:

i) the axis,
ii) the stride, and
iii) the offset.

The papers present two separate algorithms called the compact dynamic programming algorithm and the constraint graph method for minimizing a communication cost function.

A. Darte and Y. Robert [8] study the problem of mapping perfectly nested affine loops onto distributed memory parallel computers. The problem is formulated by introducing the communication graph that captures all the required information to align data and computations. Each instruction and each array is represented by a vertex, the directed edges representing read and write data accesses. The problem of message vectorization and the use of global communication operations, like broadcasting, is addressed.

In [11] an algorithm is presented for computing HPF like data distribution functions. A distribution graph is constructed representing the relation between the data access functions and the array accessed. Based on the distribution graph a decision tree, modeling all possible combinations of data distribution functions, is traversed using a branch and bound algorithm. The cost function minimized by the algorithm is based on a communication analysis tool. The computation mapping is done in accordance with the owner computes rule.

M. Wolfe and M. Ikey [33] propose in 1994 an adaption of the techniques introduced by Li and Chen [22,23] for the language 'Crystal' to the imperative language 'Tiny'. The alignment phase is decomposed into four operations:

i) finding reference patterns,
ii) adding implicit dimensions to the arrays when required,
iii) building a component affinity graph, and
iv) partitioning the component affinity graph.

As the partitioning problem is *NP*-hard, a heuristic is used. The authors furthermore describe an algorithm to generate SPMD code based on the alignments computed.

J. Garcia, E. Ayguagé and J. Labarta [15] proposed for an algorithm to compute data distribution functions that can be expressed using HPF distribute statements. This algorithm is based on the construction and traver-

sal of a single data structure, called the computation-parallelism graph. The computation-parallelism graph represents all possible data distributions along the dimensions of the arrays. Parallelism constraints are modeled as hyper-edges. Weights are associated to the edges to represent the associated communication costs. Negative costs are associated with the hyperedges to represent the associated parallelism. It is shown that distributing the data according to one dimension is equivalent to finding a path through the computation-parallelism graph fulfilling some additional constraints. The problem is formulated as a 0–1 integer programming problem. In contrast to other graph based approaches, the computation-parallelism graph models both the possible data distribution, that is, the locality constraints within a single data structure, and the possible parallelism.

W. Kelly and W. Pugh [18] describe a technique to minimize communication while preserving parallelism. The approach is not sensitive to the original program structure. For each array, the possible data mapping functions form a finite set of candidate space mappings. These sets consist of each dimension of the original iteration space being distributed. Next, for each candidate space, that is, for each possible data distribution function, all possible permutations of the surrounding loops are considered and the obtained parallelism measured. In a third step a weighted graph is constructed to model the parallelism as well as the communication cost associated with various data decompositions. One node in this weighted graph represents one candidate space mapping for each statement. The weight associated with a node is its degree of parallelism obtained. The edges represent the communication required and their weight models the communication costs. The alignment problem, as formulated in [18], is the problem of selecting one node per statement such that the sum of the weights of the selected nodes and edges is minimized. An algorithm to find such a set using various pruning strategies to reduce the size of the search space is presented.

Approaches Using a Linear Algebra Framework

Sheu and T.-H. Toi [31] introduced a method for the parallel execution of nested loops with constant loop-carried data dependences by reducing the communication overhead. First the nested loops are partitioned into large blocks which result in little inter-block communication. For a given linear transformation found by the hyperplane method [21], the iterations are partitioned into blocks such that the communication among the blocks is reduced while the execution order defined by the time transformation is not disturbed. The partitioning is based on projection techniques. In a second step these blocks are mapped onto message-passing multiprocessor systems according to specific properties of the target machine.

M. O'Boyle and G.A. Hedayat [26,27] express the alignment problem in a linear algebra framework. In this framework, aligned data can be viewed as forming a subspace in the iteration space. The problem solved is the computation of a transformation of the data access functions relative to one another such as to maximize the number of iteration points in the loop iteration space for which no communication is needed.

P. Feautrier [14] addresses the problem of finding an alignment function that maps the computations on a one-dimensional grid of virtual processors. The data mapping functions are defined by the owner computes rule which is imposed. The alignment constraints between computation and data accesses are derived from the data-flow graph of the program, procedure or loop nest considered. The data-flow graph is a directed graph. Vertices correspond to statements and the arcs to producers and consumers of data. For each statement, the alignment function is assumed to be an affine function of the iteration vectors with unknown parameters. The locality of data accesses is imposed by asking that the producer and the consumer of a data element be the same processor. Feautrier defines distance vectors between all pairs of producers and consumers. To any arc of the data-flow graph corresponds a distance vector that expresses the difference of the indices of the processor that computes the data and the one that uses it. Thus, a computation is local if and only if the corresponding distance vector is zero. The edges are hence transformed into affine equations and the problem consists in determining nontrivial parameters for the computation mappings that zero out as many distance vectors as possible. A heuristic is used to sort the equations in decreasing order of the communication traffic induced. The system of equations, which usually does not have a non trivial solution, is solved by successive

Gauss–Jordan eliminations as long as a feasible solution remains nontrivial. A solution is nontrivial if it has one degree of parallelism.

J.M. Anderson and M.S. Lam [1] describe necessary conditions for the data elements accessed by each processor to be local. They present a greedy algorithm to compute the computation and data mapping functions that can be satisfied. They incrementally add constraints as long as their conditions are satisfied, starting with the most frequently used array access functions. They only consider the linear part of the data access functions, taking care of the constant offsets in a second step. Their heuristic technique is close to the one defined in [9].

A. Platonoff [28,29] develops extensions to Feautrier's [14] automatic data distribution algorithm. A method is presented to extract global broadcast operations as well as translation operations to optimize the data mapping functions. In the data-flow graph, patterns representing broadcast and other global communication patterns are searched for. The data distribution is then chosen such as to maximize the number of global communication operations possible.

M. Dion and Robert [12,13] consider a problem in which all data access functions are of full rank and no smaller than d, the required degree of parallelism. This ensures that the parallelism obtained is indeed as large as wanted. By considering only the linear parts they compute the largest set of alignment constraints that can be satisfied while yielding the given degree of parallelism d. The constant offsets are considered subsequently, using techniques developed by Darte and Robert [8]. They consider a set of candidate solutions and search for an optimal one that verifies the largest number of constraints while effectively yielding the degree of parallelism desired. In their approach, Dion and Robert consider three basic cases depending on the structure of the data access function. Then, they build a directed graph defined as follows. Vertices correspond either to statements or arrays. There is an arc from vertex p to vertex q if and only if a mapping of rank d can be computed for q from a given mapping of rank d for p according to the basic cases enumerated previously. In this graph they search for a tree containing a maximal number of arcs. Obviously, choosing a mapping of rank d for the root of the computed tree implicitly determines mappings of rank d for all other vertices.

C. Mongenet [25] is interested in minimizing communication costs in the presence of systems of affine recurrence equations, that is, single assignment loop nests. The data dependences are subdivided into two classes:

i) auto dependences, and
ii) cross dependences.

Auto-dependences are data dependences between two data accesses to the same array. The domains of these arrays are projected onto hyperplanes such as to minimize the number of remote data accesses. Cross-dependences are dependences between data accesses to different arrays. Unimodular transformations are applied to the projected domains to align the different data array and so minimize the resulting communications. A heuristic based on these two steps is introduced.

C.G. Diderich [9] and Diderich and M. Gengler [10] present and extend the algorithm for solving this problem introduced in [2]. In a second step they introduce the constant degree parallelism alignment problem. It is the problem of finding computation and data mapping functions that minimize the number of remote data accesses for a given degree of parallelism. An exact implicit enumeration algorithm is presented. It proceeds by enumerating all interesting subsets of alignment constraints to be satisfied. To allow large alignment problems to be solved an efficient heuristic is presented and applied to various benchmarks.

Other Approaches

B.M. Chapman, T. Fahringer and H.P. Zima [4] for a software tool to provide automatic support for the mapping of the data onto the processors of the target machine. The computation is mapped by using the owner computes rule. The tool is integrated within the *Vienna Fortran Compilation System*, a compiler for Vienna Fortran, an HPF like Fortran dialect. The tool makes use of performance analysis methods and uses, via heuristics, empirical performance data. Once the performance data has been obtained for a given program, an inter-procedural alignment and pattern matching phase determines a suitable alignment of the arrays within each procedure. The alignments are then propagated through the call graph of the program.

Eventually more versions of a procedure are generated, corresponding to differently distributed actual arguments. Finally code is generated using the selected data distributions.

In [7], P. Crooks and R.H. Perrott present an algorithm for determining data mapping functions by generating HPF like directives. Their approach is based on identifying reference patterns. To each read/write pair is associated an ideal data distribution that minimized inter-processor communication. Once the preferences for the individual accesses are determined, a performance estimator is used to select the combination of preferences that gives the best performance estimate.

R. Bixby, K. Kennedy and U. Kremer [3] present an automatic data layout algorithm based on 0–1 integer programming techniques. The data mapping functions, following the HPF alignment structure, are optimized for a target distributed memory machine, a specific problem size and the number of available processors. The distribution analysis uses the alignment search space, that is, the space of all possible HPF like alignments, to build candidate data layout search spaces of reasonable data mapping functions for each loop nest. In a second step the inter-phase or inter-loop nests data layout problem is addressed. By using an integer programming formulation, a data mapping function is selected for each loop nest such that a single global cost function, modeling the communication costs, is minimized.

Conclusion

This article presents major advancements made in solving the alignment problem. Different subproblems are defined and described. One major open problem is how to incorporate scheduling information into the algorithms computing efficient alignment functions. See [9] for a first approach towards computing scheduling functions compatible with computation and data mapping functions. The question of which cost function to use when computing alignment functions has to be addressed with more details.

See also

► Integer Programming

References

1. Anderson JM, Lam MS (1993) Global optimizations for parallelism and locality on scalable parallel machines. In: ACM SIGPLAN Conf. Programming Language Design and Implementation (PLDI '93). ACM, New York, pp 112–125
2. Bau D, Kodukula I, Kotylar V, Pingali K, Stodghill P (1994) Solving alignment using elementary linear algebra. In: 7th Internat. Workshop Languages and Compilers for Parallel Computing (LCPC '94). In: Lecture Notes Computer Sci, vol 892. Springer, Berlin, pp 46–60
3. Bixby R, Kennedy K, Kremer U (1994) Automatic data layout using 0–1 integer programming. Internat. Conf. Parallel Architectures and Compilation Techniques (PACT '94). pp 111–122
4. Chapman BM, Fahringer T, Zima HP (1993) Automatic support for data distribution on distributed memory multiprocessor systems. In: 6th Internat. Workshop Languages and Compilers for Parallel Computing (LCPC '93). In: Lecture Notes Computer Sci, vol 768. Springer, Berlin, pp 184–199
5. Chatterjee S, Gilbert JR, Schreiber R, Sheffler TJ (1994) Array distribution in data-parallel programs. In: 7th Internat. Workshop Languages and Compilers for Parallel Computing (LCPC '94). In: Lecture Notes Computer Sci, vol 892. Springer, Berlin, pp 78–91
6. Chen T-S, Sheu J-P (1994) Communication-free data allocation techniques for parallelizing compilers on multicomputers. IEEE Trans Parallel and Distributed Systems 5(9):921–938
7. Crooks P, Perrott RH (1993) An automatic data distribution generator for distributed memory MIMD machines. In: 4th Internat. Workshop Compilers for Parallel Computers, pp 33–44
8. Darte A, Robert Y (1994) On the alignment problem. Parallel Proc Lett 4(3):259–270
9. Diderich CG (1998) Automatic data distribution for massively parallel distributed memory computers. PhD Thesis. Computer Sci. Dept. Swiss Federal Inst. Tech., Lausanne
10. Diderich CG, Gengler M (1997) The alignment problem in a linear algebra framework. In: Proc. Hawaii Internat. Conf. System Sci. (HICSS-30); Software Techn. Track. IEEE Computer Soc Press, New York, pp 586–595
11. Dierstein A, Hayer R, Rauber T (1994) The ADDAP system on the iPSC/860: Automatic data distribution and parallelization. J Parallel Distributed Comput 32(9):1–10
12. Dion M (1996) Alignement et distribution en parallélisation automatique. PhD Thesis. Ecole Normale Sup. Lyon (In French)
13. Dion M, Robert Y (1996) Mapping affine loop nests. Parallel Comput 22:1373–1397
14. Feautrier P (1992) Towards automatic distribution. Parallel Proc Lett 4(3):233–244
15. Garcia J, Ayguadé E, Labarta J (1995) A novel approach towards automatic data distribution. In: Supercomputing '95 Conf

16. Gupta M (1992) Automatic data partitioning on distributed memory multicomputers. PhD Thesis. Univ. Illinois at Urbana-Champaign, Urbana, IL
17. Huang C-H, Sadayappan P (1991) Communication-free hyperplane partitioning of nested loops. In: 4th Internat Workshop Languages and Compilers for Parallel Computing (LCPC '91), vol 589. In: Lecture Notes Computer Sci, vol 589. Springer, Berlin, pp 186–200
18. Kelly W, Pugh W (1996) Minimizing communication while preserving parallelism. In: 1996 ACM Internat. Conf. Supercomputing (ICS '96). ACM, New York, pp 52–60
19. Knobe K, Lukas JD, Steele GL Jr (1990) Data optimization: Allocation of arrays to reduce communication on SIMD machines. J Parallel Distributed Comput 8(2):102–118
20. Kunchithapadam K, Miller BP (1994) Optimizing array distributions in data-parallel programs. In: 7th Internat. Workshop Languages and Compilers for Parallel Computing (LCPC '94). In: Lecture Notes Computer Sci, vol 892. Springer, Berlin, pp 470–484
21. Lamport L (1974) The parallel execution of DO loops. Comm ACM 17(2):83–93
22. Li J, Chen M (1990) Index domain alignment: Minimizing cost of cross-referencing between distributed arrays. In: 3rd Symp. Frontiers of Massively Parallel Computation (Frontiers '90). IEEE Computer Soc Press, New York, pp 424–433
23. Li J, Chen M (1991) The data alignment phase in compiling programs for distributed-memory machines. J Parallel Distributed Comput 13:213–221
24. Lim AW, Lam MS (1994) Communication-free parallelization via affine transformations. In: 7th Internat. Workshop Languages and Compilers for Parallel Computing (LCPC '94). In: Lecture Notes Computer Sci, vol 892. Springer, Berlin, pp 92–106
25. Mongenet C (1995) Mappings for communications minimization using distribution and alignment. In: Internat. Conf. Parallel Architectures and Compilation Techniques (PACT '95). pp 185–193
26. O'Boyle M (1993) A data partitioning algorithm for distributed memory compilation. Techn Report Ser Univ Manchester, England UMCS-93-7-1
27. O'Boyle M, Hedayat GA (1992) Data alignment: Transformation to reduce communication on distributed memory architectures. In: Scalable High Performance Computing Conf. (SHPCC '92). IEEE Computer Soc Press, New York, pp 366–371
28. Platonoff A (1995) Automatic data distribution for massively parallel computers. In: Int. Workshop Compilers for Parallel Computers, pp 555–570
29. Platonoff A (1995) Contribution à la distribution automatique des données pour machines massivement parallèles. PhD Thesis. Ecole Normale Sup. Mines de Paris (In French)
30. Ramanujam J, Sadayappan P (1991) Compile-time techniques for data distribution in distributed memory machines. IEEE Trans Parallel and Distributed Systems 2(4): 472–482
31. Sheu J-P, Tai T-H (1991) Partitioning and mapping nested loops on multiprocessor systems. IEEE Trans Parallel and Distributed Systems 2(4):430–439
32. Sinharoy B, Szymanski BK (1994) Data and task alignment in distributed memory architectures. J Parallel Distributed Comput 21:61–74
33. Wolfe M, Ikei M (1994) Automatic array alignment for distributed memory multicomputers. 27th Annual Hawaii Internat. Conf. System Sci., vol II. IEEE Computer Soc. Press, New York, pp 23–32

αBB Algorithm

CLAIRE S. ADJIMAN,
CHRISTODOULOS A. FLOUDAS
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 49M37, 65K10, 90C26, 90C30

Article Outline

Keywords
General Framework
Convexification and Underestimation Strategy
  Function Decomposition
  Linear and Convex Terms
  Bilinear Terms
  Trilinear, Fractional and Fractional Trilinear Terms
  Univariate Concave Terms
  General Nonconvex Terms
  Overall Convexification/Relaxation Strategy
  Equality Constraints
Branching Variable Selection
  Least Reduced Axis Rule
  Term Measure
  Variable Measure
Variable Bound Updates
  Optimization-Based Approach
  Interval-Based Approach
Algorithmic Procedure
Computational Experience
Conclusions
See also
References

Keywords

Global optimization; Interval arithmetic; Twice-differentiable NLPs; Branch and bound; αBB algorithm

Deterministic global optimization techniques for nonconvex NLPs have been the subject of growing interest because they can potentially provide a very complete characterization of the problem being considered. In addition to guaranteeing identification of the global solution within arbitrary accuracy, they enable the location of all local and global solutions of the problem. As a result, they can be used to determine the feasibility of a given problem with certainty [1,2,3,4], or to find all solutions of a nonlinear system of equations [13]. They are especially valuable in the study of systems in which the global optimum solution is the only physically meaningful solution, as is the case of the phase equilibrium of non ideal mixtures [16,17,18,19,20]. Traditionally, a major theoretical limitation of these approaches has been their inability to tackle problems with arbitrary nonconvexities. However, the recent development of rigorous convex relaxation techniques for general twice continuously differentiable functions [2,3,4] has greatly expanded the class of problems that can be addressed through deterministic global optimization. These approaches have been incorporated within a branch and bound framework to create the *αBB global optimization algorithm* for twice continuously differentiable problems [3,6,12]. The theoretical basis of the algorithm as well as the efficient search strategies it uses are discussed in this article.

General Framework

The αBB algorithm guarantees finite ϵ-convergence to the global solution of nonlinear programming problems (NLPs) belonging to the general class

$$\begin{cases} \min_{\mathbf{x}} & f(\mathbf{x}) \\ \text{s.t.} & \mathbf{g}(\mathbf{x}) \leq 0 \\ & \mathbf{h}(\mathbf{x}) = 0 \\ & \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U], \end{cases} \tag{1}$$

where $f(\mathbf{x})$, $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ are continuous twice-differentiable functions.

The solution scheme is based on the generation of a nonincreasing sequence of upper bounds and a nondecreasing sequence of lower bounds on the global solution. The monotonicity of these sequences is ensured through successive partitioning of the search space which enables the construction of increasingly tight relaxations of the problem. The validity of the bounds obtained is of crucial importance in a rigorous global optimization approach. The *upper bounding* step does not present any theoretical difficulties and consists of a local optimization of the nonconvex problem. The *lower bounding* step is a more challenging operation in which the nonconvex problem must be convexified and underestimated in the current subdomain. The strategy adopted dictates the applicability of the algorithm and plays a pivotal role in its performance as it determines the tightness of the lower bounds obtained. The procedure followed in the αBB algorithm is discussed in the next section. Finally, the *branching* step involves the partition of the solution domain with the smallest lower bound on the global optimum solution into a covering set of subdomains. Although this is a simple task, the choice of partition has implications for the rate of convergence of the algorithm and efficient branching rules must be used.

Convexification and Underestimation Strategy

A convex relaxation of problem (1) is obtained by constructing convex underestimators for the nonconvex objective function and inequality constraints and by relaxing the nonlinear equality constraints, replacing them with less stringent linear equality constraints or a set of two convex inequalities. The general convexification/relaxation procedure used is first discussed for the objective function and nonconvex inequalities.

Function Decomposition

A convex underestimator for a twice continuously differentiable function is constructed by following a two-stage procedure. In the first stage, the function is decomposed into a summation of terms of special structure, such as linear, convex, bilinear, trilinear, fractional, fractional trilinear, concave in one variable and

general nonconvex terms. Then, based on the fact that the summation of convex functions results in a convex function, a tailored convex underestimator is used for each different term type. Thus, a twice-differentiable function $F(\mathbf{x})$ defined over the domain $[\mathbf{x}^L, \mathbf{x}^U]$ is written as

$$\begin{aligned} F(\mathbf{x}) = {} & c^\top \mathbf{x} + F_C(\mathbf{x}) + \sum_{i=1}^{bt} b_i x_{B_i,1} x_{B_i,2} \\ & + \sum_{i=1}^{tt} t_i x_{T_i,1} x_{T_i,2} x_{T_i,3} + \sum_{i=1}^{ft} f_i \frac{x_{F_i,1}}{x_{F_i,2}} \\ & + \sum_{i=1}^{ftt} ft_i \frac{x_{FT_i,1} x_{FT_i,2}}{x_{FT_i,3}} + \sum_{i=1}^{uct} F_{UC_i}(x_{UC_i}) \\ & + \sum_{i=1}^{nct} F_{NC_i}(\mathbf{x}), \end{aligned} \tag{2}$$

where c is a scalar vector; $F_C(\mathbf{x})$ is a convex function; bt is the number of bilinear terms, b_i is the coefficient of the ith bilinear term and $x_{B_i,1}$ and $x_{B_i,2}$ are the two variables participating in the bilinear term; tt is the number of trilinear terms, t_i is the coefficient of the ith trilinear term and $x_{T_i,1}$ $x_{T_i,2}$ and $x_{T_i,3}$ are the three variables participating in the trilinear term; ft is the number of fractional terms, f_i is the coefficient of the ith fractional term and $x_{F_i,1}$ and $x_{F_i,2}$ are the two variables participating in the fractional term; ftt is the number of fractional trilinear terms, ft_i is the coefficient of the ith fractional trilinear term and $x_{FT_i,1}$, $x_{FT_i,2}$ and $x_{FT_i,3}$ are the three variables participating in the fractional trilinear term; uct is the number of univariate concave terms, F_{UC_i} is the ith univariate concave term and x_{UC_i} is the variable participating in the univariate concave term; nct is the number of general nonconvex terms and $F_{NC_i}(\mathbf{x})$ is the ith general nonconvex term.

The decomposition phase serves two purposes: it can lead to the construction of a tight underestimator by taking advantage of the special structure of the function and it may reduce the complexity of the underestimation strategy by permitting the treatment of terms which involve a smaller number of variables than the overall nonconvex function. As will become apparent, this is especially important for general nonconvex terms.

Linear and Convex Terms

Any term that has been identified as linear or convex does not need to be modified during the convexification/underestimation procedure.

Bilinear Terms

The bilinear terms can be replaced by their convex envelope [5,15]. A new variable w_B substitutes a bilinear term x_1 x_2 and is bounded by a set of four inequality constraints which depend on the variable bounds.

$$\begin{cases} w_B \geq x_1^L x_2 + x_2^L x_1 - x_1^L x_2^L, \\ w_B \geq x_1^U x_2 + x_2^U x_1 - x_1^U x_2^U, \\ w_B \leq x_1^U x_2 + x_2^L x_1 - x_1^U x_2^L, \\ w_B \leq x_1^L x_2 + x_2^U x_1 - x_1^L x_2^U. \end{cases} \tag{3}$$

Trilinear, Fractional and Fractional Trilinear Terms

For trilinear, fractional and fractional trilinear terms, the convex underestimators proposed in [13] can be used. They are constructed in a fashion similar to the bilinear term underestimators: a new variable replaces the term and a set of inequality constraints provides bounds on this variable. For a trilinear term $x_1 x_2 x_3$, for instance, the substitution variable w_T is subject to

$$\begin{cases} w_T \geq & x_1 x_2^L x_3^L + x_1^L x_2 x_3^L \\ & + x_1^L x_2^L x_3 - 2 x_1^L x_2^L x_3^L, \\ w_T \geq & x_1 x_2^U x_3^U + x_1^U x_2 x_3^L \\ & + x_1^U x_2^L x_3 - x_1^U x_2^L x_3^L - x_1^U x_2^U x_3^U, \\ w_T \geq & x_1 x_2^L x_3^L + x_1^L x_2 x_3^U \\ & + x_1^L x_2^U x_3 - x_1^L x_2^U x_3^U - x_1^L x_2^L x_3^L, \\ w_T \geq & x_1 x_2^U x_3^L + x_1^U x_2 x_3^U \\ & + x_1^L x_2^U x_3 - x_1^L x_2^U x_3^L - x_1^U x_2^U x_3^U, \\ w_T \geq & x_1 x_2^L x_3^U + x_1^L x_2 x_3^L \\ & + x_1^U x_2^L x_3 - x_1^U x_2^L x_3^U - x_1^L x_2^L x_3^L, \\ w_T \geq & x_1 x_2^L x_3^U + x_1^L x_2 x_3^U \\ & + x_1^U x_2^U x_3 - x_1^L x_2^L x_3^U - x_1^U x_2^U x_3^U, \\ w_T \geq & x_1 x_2^U x_3^L + x_1^U x_2 x_3^L \\ & + x_1^L x_2^L x_3 - x_1^U x_2^U x_3^L - x_1^L x_2^L x_3^L, \\ w_T \geq & x_1 x_2^U x_3^U + x_1^U x_2 x_3^U \\ & + x_1^U x_2^U x_3 - 2 x_1^U x_2^U x_3^U. \end{cases} \tag{4}$$

For a fractional term x_1/x_2 with $x_2^L > 0$, the new variable w_F is bounded by

$$w_F \geq \begin{cases} \frac{x_1^L}{x_2} + \frac{x_1}{x_2^U} - \frac{x_1^L}{x_2^U} & \text{if } x_1^L \geq 0, \\ \frac{x_1}{x_2^U} - \frac{x_1^L x_2}{x_2^L x_2^U} + \frac{x_1^L}{x_2^L} & \text{if } x_1^L < 0, \end{cases}$$
$$w_F \geq \begin{cases} \frac{x_1^U}{x_2} + \frac{x_1}{x_2^L} - \frac{x_1^U}{x_2^L} & \text{if } x_1^U \geq 0, \\ \frac{x_1}{x_2^L} - \frac{x_1^U x_2}{x_2^L x_2^U} + \frac{x_1^U}{x_2^U} & \text{if } x_1^U < 0. \end{cases} \quad (5)$$

Finally, for a fractional trilinear term x_1x_2/x_3 with $x_1^L, x_2^L \geq 0$ and $x_3^L > 0$, the substitution variable w_{FT} is subject to

$$\begin{cases} w_{FT} \geq \frac{x_1 x_2^L}{x_3^U} + \frac{x_1^L x_2}{x_3^U} + \frac{x_1^L x_2^L}{x_3} - \frac{2x_1^L x_2^L}{x_3^U}, \\ w_{FT} \geq \frac{x_1 x_2^L}{x_3^U} + \frac{x_1^L x_2}{x_3^L} + \frac{x_1^L x_2^U}{x_3} - \frac{x_1^L x_2^U}{x_3^L} - \frac{x_1^L x_2^L}{x_3^U}, \\ w_{FT} \geq \frac{x_1 x_2^U}{x_3^L} + \frac{x_1^U x_2}{x_3^U} + \frac{x_1^U x_2^L}{x_3} - \frac{x_1^U x_2^L}{x_3^U} - \frac{x_1^U x_2^U}{x_3^L}, \\ w_{FT} \geq \frac{x_1 x_2^U}{x_3^U} + \frac{x_1^U x_2}{x_3^L} + \frac{x_1^L x_2^U}{x_3} - \frac{x_1^L x_2^U}{x_3^U} - \frac{x_1^U x_2^U}{x_3^L}, \\ w_{FT} \geq \frac{x_1 x_2^L}{x_3^U} + \frac{x_1^L x_2}{x_3^L} + \frac{x_1^U x_2^L}{x_3} - \frac{x_1^U x_2^L}{x_3^L} - \frac{x_1^L x_2^L}{x_3^U}, \\ w_{FT} \geq \frac{x_1 x_2^U}{x_3^U} + \frac{x_1^U x_2}{x_3^L} + \frac{x_1^L x_2}{x_3} - \frac{x_1^L x_2^U}{x_3^U} - \frac{x_1^U x_2^U}{x_3^L}, \\ w_{FT} \geq \frac{x_1 x_2^L}{x_3^U} + \frac{x_1^L x_2}{x_3^L} + \frac{x_1^U x_2^L}{x_3} - \frac{x_1^U x_2^L}{x_3^L} - \frac{x_1^L x_2^L}{x_3^U}, \\ w_{FT} \geq \frac{x_1 x_2^U}{x_3^L} + \frac{x_1^U x_2}{x_3^L} + \frac{x_1^U x_2^U}{x_3} - \frac{2x_1^U x_2^U}{x_3^L}. \end{cases} \quad (6)$$

Univariate Concave Terms

For univariate concave terms, the convexification/underestimation procedure does not require the introduction of new variables or constraints: a simple linearization of the term suffices. Thus, a univariate concave term $F_{UC}(x)$ is replaced by the linear term

$$F_{UC}(x^L) + \frac{F_{UC}(x^U) - F_{UC}(x^L)}{x^U - x^L}(x - x^L). \quad (7)$$

General Nonconvex Terms

For a general nonconvex term $F_{NC}(\mathbf{x})$, a convex underestimator $\check{F}_{NC}(\mathbf{x})$ over $[\mathbf{x}^L, \mathbf{x}^U]$ is constructed by subtracting a positive separable quadratic term from $F_{NC}(\mathbf{x})$ [12]:

$$\check{F}_{NC}(\mathbf{x}) = F_{NC}(\mathbf{x}) - \sum_{j=1}^{n} \alpha_j (x_j - x_j^L)(x_j^U - x_j), \quad (8)$$

where n is the number of variables and the α parameters are positive scalars.

The magnitude of the α parameters determines both the quality of the convex underestimator, that is, its tightness, and its convexity. It was shown in [12] that the maximum separation distance, $d_{\max}$, between the nonconvex term $F_{NC}(\mathbf{x})$ and its convex underestimator $\check{F}_{NC}(\mathbf{x})$ is given by

$$d_{\max} = \max_{\mathbf{x}} \left(F_{NC}(\mathbf{x}) - \check{F}_{NC}(\mathbf{x}) \right) = \frac{1}{4} \sum_{j=1}^{n} \alpha_j (x_j^U - x_j^L)^2 . \quad (9)$$

Thus, small α values are needed to construct a tight underestimator. The dependence of the maximum separation distance on the square of the variable ranges is especially important for the convergence proof of the algorithm [12]. Provided that the α values do not increase from a parent node to a child node, relation (9) guarantees that the convex relaxations become increasingly tight as the branch and bound iterations progress and smaller subdomains are generated. In the limit, the convex underestimators match the original functions. As a result, the monotonicity of the lower bound sequence can be ensured.

To meet the convexity requirement of $\check{F}_{NC}(\mathbf{x})$, the positive quadratic term needs to be sufficiently large to overcome the nonconvexity of $F_{NC}(\mathbf{x})$. This is achieved by manipulating the value of the α parameters. Based on the properties of convex functions, a necessary and sufficient condition for the convexity of $\check{F}_{NC}(\mathbf{x})$ is the positive semidefiniteness of the matrix $H_{F_{NC}}(\mathbf{x}) + 2\,\mathrm{diag}(\alpha_j)$ for all $\mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U]$, where $H_{F_{NC}}(\mathbf{x})$ is the Hessian matrix of the nonconvex term $F_{NC}(\mathbf{x})$. The diagonal matrix $\Delta = \mathrm{diag}(\alpha_j)$ results in a shift in the diagonal elements of the matrix $H_{F_{NC}}(\mathbf{x})$ and is therefore referred to as the *diagonal shift matrix*. The rigorous derivation

of a matrix Δ that satisfies the convexity condition is a difficult matter in the general case, primarily because of the nonlinear dependence of the Hessian matrix on the $\mathbf{x}$ variables. This problem can be alleviated by using *interval arithmetic* to generate an *interval Hessian matrix* $[H_{F_{NC}}]$ such that $H_{F_{NC}}(\mathbf{x}) \in [H_{F_{NC}}]$ for all $\mathbf{x} \in]\mathbf{x}^L, \mathbf{x}^U]$ [1,3,4]. This process allows the formulation of a sufficient convexity condition for the underestimator: if all real symmetric matrices in $[H_{F_{NC}}] + 2\,\mathrm{diag}(\alpha_j)$ are positive semidefinite, then $\breve{F}_{NC}(\mathbf{x})$ is convex over $[\mathbf{x}^L, \mathbf{x}^U]$.

Based on the interval Hessian matrix, a number of methods may be used to automatically and rigorously compute a diagonal shift matrix Δ that guarantees the convexity of $\breve{F}_{NC}(\mathbf{x})$. The first class of techniques generates a *uniform* diagonal shift matrix by equating all the diagonal elements of Δ with a single α value. In the second class of techniques, different α values are used and a *nonuniform* diagonal shift matrix is obtained [1,3].

In the first class of methods, the convexity condition is equivalent to the positive semidefiniteness of all real symmetric matrices in $[H_{F_{NC}}] + 2\,\mathrm{diag}(\alpha)$ and is satisfied by any α parameter such that

$$\alpha \geq \max\left\{0, -\frac{1}{2}\lambda_{\min}\left([H_{F_{NC}}]\right)\right\}, \tag{10}$$

where $\lambda_{\min}\left([H_{F_{NC}}]\right)$ is the minimum eigenvalue of $[H_{F_{NC}}]$ [3,12].

Consider a square symmetric interval Hessian matrix family $[H]$ whose element (ij) is the interval $[\underline{h}_{ij}, \overline{h}_{ij}]$ and whose radius matrix ΔH is defined as $(\Delta H)_{ij} = \frac{(\overline{h}_{ij} - \underline{h}_{ij})}{2}$. A lower bound on the minimum eigenvalue of $[H]$ can be obtained using one of the following methods [1,3,4]:

- Method I.1 — the Gershgorin theorem approach;
- Method I.2a — the E-matrix approach with $E = 0$;
- Method I.2b — the E-matrix approach with $E = \mathrm{diag}(\Delta H)$;
- Method I.3 — Mori–Kokame's approach;
- Method I.4 — the lower bounding Hessian approach;
- Method I.5 — an approach based on the Kharitonov theorem;
- Method I.6 — the Hertz approach.

Method I.1 is an extension of the *Gershgorin theorem* for real matrices to interval matrices. The minimum eigenvalue of $[H]$ is such that

$$\lambda_{\min}([H]) \geq \min_i \left[\underline{h}_{ii} - \sum_{j \neq i} \max\left(\left\|\underline{h}_{ij}\right\|, \left\|\overline{h}_{ij}\right\|\right)\right].$$

Methods I.2a and I.2b are a generalization of the results presented in [8,23]. It requires the computation of the modified midpoint matrix $\widetilde{H}_M$ such that $(\widetilde{H}_M)_{ij} = \frac{(\overline{h}_{ij} + \underline{h}_{ij})}{2}$ for $i \neq j$ and $(\widetilde{H}_M)_{ii} = 0$, as well as the computation of the modified radius matrix $\widetilde{\Delta H}$ such that $(\widetilde{\Delta H})_{ij} = \frac{(\overline{h}_{ij} - \underline{h}_{ij})}{2}$ for $i \neq j$ and $(\widetilde{\Delta H})_{ii} = \underline{h}_{ij}$. Given an arbitrary real symmetric matrix E, the minimum eigenvalue of the interval Hessian matrix $[H]$ is such that

$$\lambda_{\min}([H]) \geq \lambda_{\min}\left(\widetilde{H}_M + E\right) - \rho\left(\widetilde{\Delta H} + \|E\|\right),$$

where $\rho(M)$ denotes the spectral radius of the real matrix M. In practice, two E-matrices have been used: $E = 0$ (Method I.2a) and $E = \Delta H$ (Method I.2b).

Method I.3 is based on a result presented in [21], which uses the lower vertex matrix H, such that $(H)_{ij} = h_{ij}$, and the upper vertex matrix $\overline{H}$, such $(\overline{H})_{ij} = \overline{h}_{ij}$. The minimum eigenvalue of $[H]$ is such that

$$\lambda_{\min}([H]) \geq \lambda_{\min}(\underline{H}) - \rho(\overline{H} - \underline{H}).$$

Method I.4 uses a *lower bounding Hessian* of the interval Hessian matrix. Such a matrix is defined in [24] as a real symmetric matrix whose minimum eigenvalue is smaller than the minimum eigenvalue of any real symmetric matrix in the interval Hessian family. It therefore suffices to compute the minimum eigenvalue of this real matrix to obtain the desired lower bound. A lower bounding Hessian $L = (l_{ij})$ can be constructed from the following rule:

$$l_{ij} = \begin{cases} \underline{h}_{ii} + \sum_{k \neq i} \frac{\underline{h}_{ik} - \overline{h}_{ik}}{2}, & i = j, \\ \frac{\underline{h}_{ij} + \overline{h}_{ij}}{2}, & i \neq j. \end{cases}$$

Method I.5 is based on the *Kharitonov theorem* [11] which, by extension, gives a lower bound on the minimum eigenvalue of an interval Hessian matrix family [2]. First, the corresponding characteristic polynomial family must be derived

$$[K] = [\underline{c}_0, \overline{c}_0] + [\underline{c}_1, \overline{c}_1]\lambda + [\underline{c}_2, \overline{c}_2]\lambda^2 + [\underline{c}_3, \overline{c}_3]\lambda^3 + [\underline{c}_4, \overline{c}_4]\lambda^4 + [\underline{c}_5, \overline{c}_5]\lambda^5 + \cdots,$$

where the coefficients of λ depend on the elements of the interval Hessian matrix $[H]$. A lower bound on the roots of this polynomial can then obtained by calculating the minimum roots of only four real polynomials. The appropriate bounding polynomials are the Kharitonov polynomials

$$\begin{cases} K_1 = & \underline{c}_0 + \underline{c}_1\lambda + \overline{c}_2\lambda^2 + \overline{c}_3\lambda^3 \\ & + \underline{c}_4\lambda^4 + \underline{c}_5\lambda^5 + \cdots, \\ K_2 = & \overline{c}_0 + \overline{c}_1\lambda + \underline{c}_2\lambda^2 + \underline{c}_3\lambda^3 \\ & + \overline{c}_4\lambda^4 + \overline{c}_5\lambda^5 + \cdots, \\ K_3 = & \overline{c}_0 + \underline{c}_1\lambda + \underline{c}_2\lambda^2 + \overline{c}_3\lambda^3 \\ & + \overline{c}_4\lambda^4 + \underline{c}_5\lambda^5 + \cdots, \\ K_4 = & \underline{c}_0 + \overline{c}_1\lambda + \overline{c}_2\lambda^2 + \underline{c}_3\lambda^3 \\ & + \underline{c}_4\lambda^4 + \overline{c}_5\lambda^5 + \cdots. \end{cases}$$

Method I.6 allows the computation of the exact minimum eigenvalue of the family of symmetric matrices represented by the interval Hessian matrix. It requires the construction of 2^{n-1} vertex matrices H^k of the interval matrix $[H]$ as defined by

$$(H^k)_{ij} = \begin{cases} \underline{h}_{ii} & \text{if } i = j, \\ \underline{h}_{ij} & \text{if } u_i u_j \geq 0,\ i \neq j, \\ \overline{h}_{ij} & \text{if } u_i u_j < 0,\ i \neq j, \end{cases}$$

where all possible combinations of the signs of the arbitrary scalars u_i and u_j are enumerated. It was shown in [4,10] that the lowest minimum eigenvalue from this set of real matrices is the minimum eigenvalue of the interval matrix.

Three rigorous techniques for the generation of a non uniform shift matrix Δ can be used [1,3]:

- Method II.1a — the scaled Gershgorin theorem approach with scaling vector $\mathbf{d} = \underline{1}$;
- Method II.1b — the scaled Gershgorin theorem approach with scaling vector $\mathbf{d} = \mathbf{x}^U - \mathbf{x}^L$;
- Method II.2 — the H-matrix approach;
- Method II.3 — an approach based on the minimization of the maximum separation distance.

The main advantage of these techniques is that resorting to a different value of the α parameter for each variable may lead to tighter underestimators by taking into account the individual contribution of each variable to the overall nonconvexity of the term being considered. In the case of a uniform diagonal shift, the worst contribution is uniformly assigned to all variables.

Methods II.1a and II.1b bear resemblance with the Gershgorin theorem used for Method I.1. In the present case, however, each row is considered independently and the ith element of the diagonal shift matrix, α_i, is the maximum of zero and

$$-\frac{1}{2}\left(\underline{h}_{ii} - \sum_{j \neq i} \max\left\{\left\|\underline{h}_{ij}\right\|, \left\|\overline{h}_{ij}\right\|\right\} \frac{d_j}{d_i}\right),$$

where $\mathbf{d}$ is an arbitrary positive vector. In practice, $\mathbf{d} = \underline{1}$ (Method II.1a) and $\mathbf{d} = \mathbf{x}^U - \mathbf{x}^L$ (Method II.1b) have been used. The latter choice of scaling often helps to reduce the maximum separation distance between the nonconvex term and its underestimator by assigning smaller α values to variables with a larger range.

Method II.2 is an iterative method based on the properties of H-matrices: a square interval matrix that has the H-matrix property is regular and does not have 0 as an eigenvalue [22]. In order to determine whether a square interval matrix $[H]$ is an H-matrix, its comparison matrix $\langle H \rangle$ must first be defined. For $i \neq j$, the off-diagonal element $(\langle H \rangle)_{ij}$ of the comparison matrix is given by $-\max\{\left\|\underline{h}_{ij}\right\|, \left\|\overline{h}_{ij}\right\|\}$. A diagonal element $(\langle H \rangle)_{ii}$ of the comparison matrix is given by

$$\begin{cases} 0, & 0 \in [\underline{h}_{ii}, \overline{h}_{ii}], \\ \min\left\{\left\|\underline{h}_{ii}\right\|, \left\|\overline{h}_{ii}\right\|\right\}, & 0 \notin [\underline{h}_{ii}, \overline{h}_{ii}]. \end{cases}$$

A real matrix such as $\langle H \rangle$ is an M-matrix if all its off-diagonal elements are nonpositive – this is always true for $\langle H \rangle$ – *and* if there exists a real positive vector $\mathbf{u}$ such that $\langle H \rangle \mathbf{u} > 0$. The interval matrix $[H]$ is an H-matrix if its comparison matrix $\langle H \rangle$ is an M-matrix. Method II.2 follows an iterative procedure to construct a nonuniform diagonal shift matrix Δ such that $[H] + 2\,\Delta$ is an H-matrix whose modified midpoint matrix is positive definite. If these conditions are met, the diagonal elements of the shift matrix are guaranteed to lead to the construction of a *convex* underestimator for the nonconvex term. The initial guess chosen for Δ is the uniform diagonal shift matrix given by Method I.2.

Method II.3 aims to generate a non uniform diagonal shift matrix which minimizes the maximum separation distance between the nonconvex term and its underestimator. For this purpose, the following semidefinite programming problem is solved using an interior

point method [25]:

$$\begin{cases} \min_{\alpha_i} & (\mathbf{x}^U - \mathbf{x}^L)^\top \Delta (\mathbf{x}^U - \mathbf{x}^L) \\ \text{s.t.} & L + 2\,\mathrm{diag}(\alpha_i) \geq 0 \\ & \alpha_i \geq 0, \quad \forall i, \end{cases}$$

where L is the lower bounding Hessian matrix defined in Method I.4. Because this approach is based on the lower bounding Hessian matrix rather than the exact $\mathbf{x}$-dependent Hessian matrix, the solution found does not correspond to the smallest achievable maximum separation distance, but can be expected to be smaller than when Method I.4 is used.

A comparative study [1,3] of all the methods available for the generation of a diagonal shift matrix found that Methods II.1a, II.1b and II.3 usually give the tightest underestimators. However, Method II.3 is computationally intensive and therefore results in poorer convergence rates than Methods II.1a and II.1b. Since the least computationally expensive techniques for the generation of the diagonal shift matrix, Methods I.1, II.1a and II.1b, are of order $O(n^2)$, the decomposition of the nonconvex terms into a summation of terms involving a smaller number of variables may have a significant impact on the performance of the algorithm.

Overall Convexification/Relaxation Strategy

Based on the rigorous convexification/underestimation schemes for bilinear, trilinear, fractional, fractional trilinear, univariate concave and general nonconvex terms, the overall convex underestimator $\check{F}(\mathbf{x}, \mathbf{w})$ for a twice continuously differentiable function $F(\mathbf{x})$ decomposed according to (2) is

$$\begin{aligned} \check{F}(\mathbf{x}, \mathbf{w}) &= c^\top \mathbf{x} + F_C(\mathbf{x}) + \sum_{i=1}^{bt} b_i w_{B_i} \\ &+ \sum_{i=1}^{tt} t_i w_{T_i} + \sum_{i=1}^{ft} f_i w_{F_i} + \sum_{i=1}^{ftt} ft_i w_{FT_i} \\ &+ \sum_{i=1}^{uct} \Bigg(F_{UC_i}(x^L_{UC_i}) \\ &+ \frac{F_{UC_i}(x^U_{UC_i}) - F_{UC_i}(x^L_{UC_i})}{x^U_{UC_i} - x^L_{UC_i}} (x_{UC_i} - x^L_{UC_i}) \Bigg) \\ &+ \sum_{i=1}^{nct} \left(F_{NC_i}(\mathbf{x}) - \sum_{j=1}^{n} \alpha_{ij} (x_j - x^L_j)(x^U_j - x_j) \right), \end{aligned} \tag{11}$$

where the notation is as defined for (2). The introduction of the new variables w_{B_i}, w_{T_i}, w_{F_i} and w_{FT_i} is accompanied by the addition of convex inequalities of the type given in (3), (4), (5) and (6). For the trilinear, fractional and fractional trilinear terms, the specific form of these equations depends on the sign of the term coefficients and variable bounds.

The form given by (11) can be used to construct convex underestimators for the objective function and inequality constraints.

Equality Constraints

For nonlinear equality constraints, two different convexification/relaxation schemes are used, depending on the mathematical structure of the function. If the equality $h(\mathbf{x} = 0$ involves only linear, bilinear, trilinear, fractional and fractional trilinear terms, it is first decomposed into the equivalent equality constraint

$$\begin{aligned} c^\top \mathbf{x} + \sum_{i=1}^{bt} b_i x_{B_i,1} x_{B_i,2} + \sum_{i=1}^{tt} t_i x_{T_i,1} x_{T_i,2} x_{T_i,3} \\ + \sum_{i=1}^{ft} f_i \frac{x_{F_i,1}}{x_{F_i,2}} + \sum_{i=1}^{ftt} ft_i \frac{x_{FT_i,1} x_{FT_i,2}}{x_{FT_i,3}} = 0, \end{aligned} \tag{12}$$

where the notation is as previously defined. (12) is then replaced by

$$\begin{aligned} c^\top \mathbf{x} + \sum_{i=1}^{bt} b_i w_{B_i} + \sum_{i=1}^{tt} t_i w_{T_i} \\ + \sum_{i=1}^{ft} f_i w_{F_i} + \sum_{i=1}^{ftt} ft_i w_{FT_i} = 0, \end{aligned} \tag{13}$$

with the addition of convex inequalities of the type given by (3), (4), (5) and (6). If the nonlinear equality contains at least one convex, univariate concave or general nonconvex term, the convexification/relaxation strategy must first transform the equality constraint $h(\mathbf{x})$ into a set of two equivalent inequality constraints

$$\begin{cases} & h(\mathbf{x}) \leq 0 \\ - & h(\mathbf{x}) \leq 0, \end{cases} \tag{14}$$

which can then be convexified and underestimated independently using (11).

The transformation of a nonconvex twice-differentiable problem into a convex lower bounding problem

described in this section allows the generation of valid and increasingly tight lower bounds on the global optimum solution.

Branching Variable Selection

Once upper and lower bounds have been obtained for all the existing nodes of the branch and bound tree, the region with the smallest lower bound is selected for branching. The partitioning of the solution space can have a significant effect on the quality of the lower bounds obtained because of the strong dependence of the convex underestimators described by (3)–(8) on the variable bounds. It is therefore important to identify the variables which most contribute to the separation between the original problem and the convex lower bounding problem at the current node. Several branching variable selection criteria have been designed for this purpose [1].

Least Reduced Axis Rule

The first strategy leads to the selection of the variable that has least been branched on to arrive at the current node. It is characterized by the largest ratio

$$\frac{x_i^U - x_i^L}{x_{i,0}^U - x_{i,0}^L},$$

where $x_{i,0}^L$ and $x_{i,0}^U$ are the lower and upper bounds on variable x_i at the first node of the branch and bound tree and x_i^L and x_i^U are the current lower and upper bounds on variable x_i.

The main disadvantage of this simple rule is that it does not account for the specificities of the participation of each variable in the problem and therefore cannot accurately identify the critical variables that determine the quality of the underestimators.

Term Measure

A more sophisticated rule is based on the computation of a term measure μ_j^t for term t_j defined as

$$\mu_j^t = t_j(\mathbf{x}^*) - \check{t}_j(\mathbf{x}^*, \mathbf{w}^*), \tag{15}$$

where $t_j(\mathbf{x})$ is a bilinear, trilinear, fractional, fractional trilinear, univariate concave or general nonconvex term, $\check{t}_j(\mathbf{x}, \mathbf{w})$ is the corresponding convex underestimator, $\mathbf{x}^*$ is the solution vector corresponding to the minimum of the convex lower bounding problem, and $\mathbf{w}^*$ is the solution vector for the new variables at the minimum of the convex lower bounding problem. One of the variables participating in the term with the largest measure μ_j^t is selected for branching.

Variable Measure

A third strategy is based on a variable measure μ_i^v which is computed from the term measures μ_j^t. For variable x_i, this measure is

$$\mu_i^v = \sum_{j \in T_i} \mu_j^t, \tag{16}$$

where T_i is the set of terms in which x_i participates. The variable with the largest measure μ_i^v is branched on.

Variable Bound Updates

The effect of the variable bounds on the convexification/relaxation procedure motivates the tightening of the variable bounds. However, the trade-off between tight underestimators generated at a large computational cost and looser underestimators obtained more rapidly must be taken into account when designing a variable bound update strategy. For this reason, one of several approaches can be adopted, depending on the degree of nonconvexity of the problem [1,3]:

- variable bound updates
  - at the beginning of the algorithmic procedure only; or
  - at each iteration;
- bound updates
  - for all variables in the problem; or
  - bound updates for those variables that most affect the quality of the lower bounds as measured by the variable measure μ_i^v.

Two different techniques can be used to tighten the variable bounds. The first is based on the generation and solution of a series of convex optimization problems while the second is an iterative procedure relying on the interval evaluation of the functions in the nonconvex NLP.

Optimization-Based Approach

In the optimization approach, a new lower or upper bound for variable x_i is obtained by solving the convex

problem

$$\begin{cases} \min_{\mathbf{x},\mathbf{w}} \text{ or } \max_{\mathbf{x},\mathbf{w}} & x_i \\ \text{s.t.} & \breve{f}(\mathbf{x},\mathbf{w}) \le \overline{f}^* \\ & \breve{\mathbf{g}}(\mathbf{x},\mathbf{w}) \le 0 \\ & \breve{\mathbf{h}}_N^+(\mathbf{x},\mathbf{w}) \le 0 \\ & \breve{\mathbf{h}}_N^-(\mathbf{x},\mathbf{w}) \le 0 \\ & \breve{\mathbf{h}}_L(\mathbf{x},\mathbf{w}) = 0 \\ & \mathbf{n}(\mathbf{x},\mathbf{w}) \le 0 \\ & \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U], \\ & \mathbf{w} \in [\mathbf{w}^L, \mathbf{w}^U], \end{cases} \tag{17}$$

where $\breve{p}(\mathbf{x},\mathbf{w})$ denotes the convex underestimator of function $p(\mathbf{x})$ as defined in (11), $\overline{f}^*$ denotes the current best upper bound on the global optimum solution, $\mathbf{h}_L(\mathbf{x})$ denotes the set of equality constraints which involve only linear, bilinear, trilinear, fractional and fractional trilinear terms, $\mathbf{h}_N^+(\mathbf{x})$ denotes the set of equality constraints that involve other term types and $\mathbf{h}_N^-(\mathbf{x})$ denotes the negative of that set, $\mathbf{n}(\mathbf{x},\mathbf{w})$ denotes the set of additional constraints that arise from the underestimation of bilinear, trilinear, fractional and fractional trilinear terms, and $\mathbf{w}$ is the corresponding set of new variables.

Interval-Based Approach

In the interval-based approach, an iterative procedure is followed for each variable whose bounds are to be updated. The original functions in the problem are used without any transformations. An inequality constraint $g(\mathbf{x}) \le 0$ is infeasible in the domain $[\mathbf{x}^L, \mathbf{x}^U]$ if its range $[g^L, g^U]$, computed so that $g(\mathbf{x}) \in [g^L, g^U]\ \forall\ \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U[$, is such that $g^L > 0$. Similarly, an equality constraint $h(\mathbf{x}) = 0$ is infeasible in this domain if its range $[h^L, h^U]$, computed so that $h(\mathbf{x}) \in [h^L, h^U]$, $\forall\ \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U]$, is such that $0 \notin [h^L, h^U]$. The variable bounds are updated based on the feasibility of the constraints in the original problem and the additional constraint that the objective function should be less than or equal to the current best upper bound $\overline{f}^*$. The feasible region is therefore defined as

$$F = \left\{ \mathbf{x}: \quad \begin{array}{c} \mathbf{g}(\mathbf{x}) \le 0, \mathbf{h}(\mathbf{x}) = 0, \\ f(\mathbf{x}) \le \overline{f}^*, \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U] \end{array} \right\}.$$

The lower (upper) bound on variable $x_i \in [x_i^L, x_i^U]$ is updated as follows:

```
PROCEDURE interval-based bound update()
    Set initial bounds L = x_i^L and U = x_i^U;
    Set iteration counter k = 0;
    Set maximum number of iterations K;
    DO k < K
        Compute midpoint M = (U + L)/2;
        Set left region {x ∈ F : x_i ∈ [L, M]};
        Set right region {x ∈ F : x_i ∈ [L, M]};
        Test interval feasibility of left (right region);
        IF feasible,
            Set U = M  (L = M);
        ELSE,
            Test interval feasibility of right (left)
            region;
            IF feasible,
                Set L = M  (U = M);
            ELSE,
                Set L = U  (U = L);
                Set U = x_i^U  (U = x_i^L)
        IF k = 0 and L = x_i^U  (U = x_i^L),
            RETURN(infeasible node);
        Set k = k + 1;
    OD;
    RETURN(x_i^L = L  (x_i^U = U));
END interval-based bound update;
```

Interval-based bound update procedure

In general, the interval-based bound update strategy is less computationally expensive than the optimization-based approach. However, at the beginning of the branch and bound search, when the bound updates are most critical and the variable ranges are widest, the overestimations inherent in interval computations often lead to looser updated bounds in the interval-based approach than in the optimization-based technique.

Algorithmic Procedure

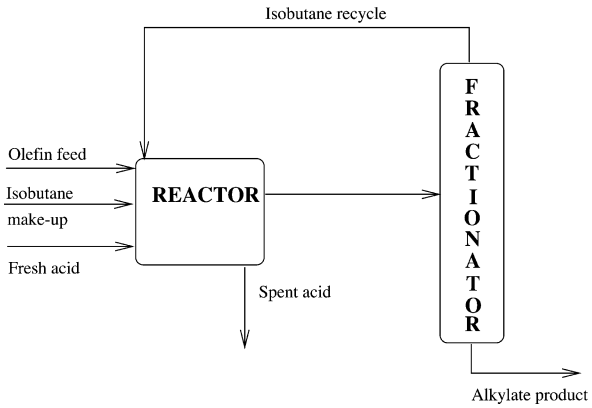
Based on the developments presented in previous sections, the procedure for the αBB algorithm can be summarized by the following pseudocode:

```
PROCEDURE αBB algorithm()
    Decompose functions in problem;
    Set tolerance ϵ;
    Set f* = f^0 = −∞ and f̄* = f̄^0 = +∞;
    Initialize list of lower bounds {f^0};
    DO f̄* − f* > ϵ
        Select node k with smallest lower bound, f^k,
        from list of lower bounds;
        Set f* = f^k;
        (Optional) Update variable bounds for cur-
        rent node using optimization or interval
        approach;
        Select branching variable;
        Partition to create new nodes;
        DO for each new node i
            Generate convex lower bounding NLP
                Introduce new variables, constraints;
                Linearize univariate concave terms;
                Compute interval Hessian matrices;
                Compute α values;
            Find solution f^i of convex lower bound-
            ing NLP;
            IF infeasible or f^i > f̄* + ϵ
                Fathom node;
            ELSE
                Add f^i to list of lower bounds;
                Find a solution f̄^i of nonconvex NLP;
                IF f̄^i < f̄*
                    Set f̄* = f̄^i;
        OD;
    OD;
    RETURN(f̄* and variables values at correspond-
    ing node);
END αBB algorithm;
```

A pseudocode for the αBB algorithm

Computational Experience

Significant computational experience with the αBB algorithm has been acquired through the solution of a wide variety of problems involving different types of nonconvexities and up to 16000 variables [1,2,3,4,6,9,12]. These include problems such as pooling/blending, design of reactor networks, design of batch plants under uncertainty [9], stability studies belonging to the class of generalized geometric programming problems, characterization of phase-equilibrium using activity coefficient models, identification of stable molecular conformations and the determination of all solutions of systems of nonlinear equations.



αBB Algorithm, Figure 1
Simplified alkylation process flowsheet

In order to illustrate the performance of the algorithm and the importance of variable bound updates, a medium-size example is presented. The objective is to maximize the profit for the simplified alkylation process presented in [7] and shown in Fig. 1.

An olefin feed (100% butene), a pure isobutane recycle and a 100% isobutane make up stream are introduced in a reactor together with an acid catalyst. The reactor product stream is then passed through a fractionator where the isobutane and the alkylate product are separated. The spent acid is also removed from the reactor. The formulation used here includes 7 variables and 16 constraints, 12 of which are nonlinear. The variables are defined as follows: x_1 is the olefin feed rate in barrels per day; x_2 is the acid addition rate in thousands of pounds per day; x_3 is the alkylate yield in barrels per day; x_4 is the acid strength (weight percent); x_5 is the motor octane number; x_6 is the external isobutane-to-olefin ratio; x_7 is the F-4 performance number. The profit maximization problem is then expressed as:

$$\begin{aligned}\text{Profit} = -\min(&1.715x_1 + 0.035x_1x_6 \\ &+ 4.0565x_3 + 10.0x_2 - 0.063x_3x_5)\end{aligned}$$

subject to:

$$
\begin{aligned}
&0.0059553571x_6^2x_1 + 0.88392857x_3 \\
&- 0.1175625x_6x_1 - x_1 \leq 0, \\
&1.1088x_1 + 0.1303533x_1x_6 \\
&- 0.0066033x_1x_6^2 - x_3 \leq 0, \\
&6.66173269x_6^2 + 172.39878x_5 \\
&- 56.596669x_4 - 191.20592x_6 \leq 10000, \\
&1.08702x_6 + 0.32175x_4 - 0.03762x_6^2 \\
&- x_5 \leq -56.85075, \\
&0.006198x_7x_4x_3 + 2462.3121x_2 \\
&- 25.125634x_2x_4 - x_3x_4 \leq 0, \\
&161.18996x_3x_4 + 5000.0x_2x_4 \\
&- 489510.0x_2 - x_3x_4x_7 \leq 0, \\
&0.33x_7 - x_5 + 44.333333 \leq 0, \\
&0.022556x_5 - 0.007595x_7 \leq 1, \\
&0.00061x_3 - 0.0005x_1 \leq 1, \\
&0.819672x_1 - x_3 + 0.819672 \leq 0, \\
&24500.0x_2 - 250.0x_2x_4 - x_3x_4 \leq 0, \\
&1020.4082x_4x_2 + 1.2244898x_3x_4 \\
&- 100000x_2 \leq 0, \\
&6.25x_1x_6 + 6.25x_1 - 7.625x_3 \leq 100000, \\
&1.22x_3 - x_6x_1 - x_1 + 1 \leq 0, \\
&1500 \leq x_1 \leq 2000, \\
&1 \leq x_2 \leq 120, \\
&3000 \leq x_3 \leq 3500, \\
&85 \leq x_4 \leq 93, \\
&90 \leq x_5 \leq 95, \\
&3 \leq x_6 \leq 12, \\
&145 \leq x_7 \leq 162.
\end{aligned}
$$

The maximum profit is \$1772.77 per day, and the optimal variable values are $x_1^* = 1698.18$, $x_2^* = 53.66$, $x_3^* = 3031.30$, $x_4^* = 90.11$, $x_5^* = 95.00$, $x_6^* = 10.50$, $x_7^* = 153.53$. In this example, variable bound tightening is performed using the optimization-based approach. An update of all the variable bounds therefore involves the solution of 14 convex NLPs. The computational cost is significant and may not always be justified by the corresponding decrease in number of iterations. Two extreme tightening strategies were used to illustrate this trade-off: an update of all variable bounds at the onset of the algorithm only ('Single Up'), or an update of all bounds at each iteration of the αBB algorithm ('One Up/Iter'). An intermediate strategy might involve bound updates for those variables that affect the underestimators most significantly or bound updates at only a few levels of the branch and bound tree. The results of runs performed on an HP9000/730 are summarized in the table below. t_U denotes the percentage of CPU time devoted to the construction of the convex underestimating problem.

Although the approach relying most heavily on variable bound updates results in tighter underestimators, and hence a smaller number of iterations, the time requirements for each iteration are significantly larger than when no bounds updates are performed. Thus, the overall CPU requirements often increase when all variable bounds are updated at each iteration.

	Single up			One Up/Iter		
Meth	Iter.	CPU sec.	t_U (%)	Iter.	CPU sec.	t_U (%)
I.1	74	37.5	0.5	31	41.6	0.0
I.2a	61	30.6	1.6	25	37.2	0.2
I.2b	61	29.2	1.0	25	35.4	0.1
I.3	69	32.8	1.9	25	31.5	0.2
I.4	61	31.6	1.4	25	33.1	0.2
I.5	61	32.8	12.3	25	36.7	1.7
I.6	59	32.9	1.4	25	32.8	0.5
II.1a	56	24.9	0.3	30	36.5	0.3
II.1b	38	13.6	1.7	17	19.9	0.5
II.2	62	32.7	0.6	25	34.5	0.3
II.3	54	21.8	16.7	23	30.4	5.0

Alkylation process design results

In order to determine the best technique for the construction of convex underestimators, the percentage of computational effort dedicated to this purpose, t_U, is tracked. As can be seen in the above table, the generation of the convex lower bounding does not consume a large share of the computational cost, regardless of the method. It is, however, significantly larger for Methods I.5 and II.3 as they require the solution of a polynomial and a semidefinite programming problem respectively. t_U decreases when bound updates are performed at each iteration as a large amount of time is

spent solving the bound updates problems. In this example, the scaled Gershgorin approach with $d_i = (x_i^U - x_i^L)$ (Method II.1b) gives the best results both in terms of number of iterations and CPU time.

Conclusions

The αBB algorithm is guaranteed to identify the global optimum solution of problems belonging to the broad class of twice continuously differentiable NLPs. It is a branch and bound approach based on a rigorous convex relaxation strategy, which involves the decomposition of the functions into a sum of terms with special mathematical structure and the construction of different convex underestimators for each class of term. In particular, the treatment of general nonconvex terms requires the analysis of their Hessian matrix through interval arithmetic. Efficient branching and variable bound update strategies can be used to enhance the performance of the algorithm.

See also

- ▶ Bisection Global Optimization Methods
- ▶ Continuous Global Optimization: Applications
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Convex Envelopes in Optimization Problems
- ▶ D.C. Programming
- ▶ Differential Equations and Global Optimization
- ▶ DIRECT Global Optimization Algorithm
- ▶ Eigenvalue Enclosures for Ordinary Differential Equations
- ▶ Generalized Primal-relaxed Dual Approach
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Batch Design Under Uncertainty
- ▶ Global Optimization in Binary Star Astronomy
- ▶ Global Optimization in Generalized Geometric Programming
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization in Phase and Chemical Reaction Equilibrium
- ▶ Global Optimization Using Space Filling
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Interval Global Optimization
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Global Optimization with αBB
- ▶ Reformulation-linearization Methods for Global Optimization
- ▶ Reverse Convex Optimization
- ▶ Semidefinite Programming and Determinant Maximization
- ▶ Smooth Nonlinear Nonconvex Optimization
- ▶ Topology of Global Optimization

References

1. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB, for general twice-differentiable constrained NLPs – II. Implementation and computational results. Comput Chem Eng 22:1159
2. Adjiman CS, Androulakis IP, Maranas CD, Floudas CA (1996) A global optimization method, αBB, for process design. Comput Chem Eng 20:S419–S424
3. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB, for general twice-differentiable constrained NLPs – I. Theoretical advances. Comput Chem Eng 22:1137
4. Adjiman CS, Floudas CA (1996) Rigorous convex underestimators for twice-differentiable problems. J Global Optim 9:23–40
5. Al-Khayyal FA, Falk JE (1983) Jointly constrained biconvex programming. Math Oper Res 8:273–286
6. Androulakis IP, Maranas CD, Floudas CA (1995) αBB: A global optimization method for general constrained nonconvex problems. J Global Optim 7:337–363
7. Bracken J, McCormick GP (1968) Selected applications of nonlinear programming. Wiley, New York
8. Deif AS (1991) The interval eigenvalue problem. Z Angew Math Mechanics 71:61–64
9. Harding ST, Floudas CA (1997) Global optimization in multiproduct and multipurpose batch design under uncertainty. Industr Eng Chem Res 36:1644–1664
10. Hertz D (1992) The extreme eigenvalues and stability of real symmetric interval matrices. IEEE Trans Autom Control 37:532–535
11. Kharitonov VL (1979) Asymptotic stability of an equilibrium position of a family of systems of linear differential equations. Differential Eq:1483–1485
12. Maranas CD, Floudas CA (1994) Global minimum potential energy conformations of small molecules. J Global Optim 4:135–170
13. Maranas CD, Floudas CA (1995) Finding all solutions of nonlinearly constrained systems of equations. J Global Optim 7:143–182

14. Maranas CD, Floudas CA (1997) Global optimization in generalized geometric programming. Comput Chem Eng 21:351–370
15. McCormick GP (1976) Computability of global solutions to factorable nonconvex programs: part I – Convex underestimating problems. Math Program 10:147–175
16. McDonald CM, Floudas CA (1994) Decomposition based and branch and bound global optimization approaches for the phase equilibrium problem. J Global Optim 5:205–251
17. McDonald CM, Floudas CA (1995) Global optimization and analysis for the Gibbs free energy function for the UNIFAC, Wilson, and ASOG equations. Industr Eng Chem Res 34:1674–1687
18. McDonald CM, Floudas CA (1995) Global optimization for the phase and chemical equilibrium problem: Application to the NRTL equation. Comput Chem Eng 19:1111–1141
19. McDonald CM, Floudas CA (1995) Global optimization for the phase stability problem. AIChE J 41:1798–1814
20. McDonald CM, Floudas CA (1997) GLOPEQ: A new computational tool for the phase and chemical equilibrium problem. Comput Chem Eng 21:1–23
21. Mori T, Kokame H (1994) Eigenvalue bounds for a certain class of interval matrices. IEICE Trans Fundam E77-A:1707–1709
22. Neumaier A (1992) An optimality criterion for global quadratic optimization. J Global Optim 2:201–208
23. Rohn J (1996) Bounds on eigenvalues of interval matrices. Techn Report Inst Computer Sci Acad Sci Prague 688
24. Stephens C (1997) Interval and bounding Hessians. In: Bonze IM et al (eds) Developments in Global Optimization. Kluwer, Dordrecht, pp 109–199
25. Vandenberghe L, Boyd S (1996) Semidefinite programming. SIAM Rev 38:49–95

Alternative Set Theory

AST

PETR VOPĚNKA, KATEŘINA TRLIFAJOVÁ
Charles University, Prague, Czech Republic

MSC2000: 03E70, 03H05, 91B16

Article Outline

Keywords
Classes, Sets and Semisets
Infinity
Axiomatic System of AST
Rational and Real Numbers
Infinitesimal Calculus
Topology
  Basic Definitions
Motion
Utility Theory
Conclusion
See also
References

Keywords

Sets; Semisets; Infinity; Countability; Continuum; Topology; Indiscernibility; Motion; Utility theory

Alternative set theory has been created and, together with his colleagues at Charles University, developed by P. Vopěnka since the 1970s. In agreement with Husserl's phenomenology, he based his theory on the natural world and the human view thereof.

The most important for any set theory is the way it treats infinity. A different approach to infinity forms the key difference between AST and classical set theories based on the *Cantor set theory* (CST). Cantor's approach led to the creation of a rigid, abstract world with an enormous scale of infinite cardinalities while Vopěnka's infinity, based on the notion of horizon, is more natural and acceptable.

Another source of inspiration were nonstandard models of Peano arithmetics with infinitely large (nonstandard) numbers. The way to build them in AST is easy and natural.

The basic references are [9,10,11].

Classes, Sets and Semisets

AST, as well as CST, builds on notions of 'set', 'class', 'element of a set' and, in addition, introduces the notion of 'semiset'. A *class* is the most general notion used for any collection of distinct objects. *Sets* are such classes that are so clearly defined and clean-cut that their elements could be, if necessary, included in a list. *Semisets* are classes which are not sets, because their borders are vague, however, they are parts of sets. For example, all living people in the world form a class—some are being born, some are dying, we do not know where all of them are. The citizens of Prague, registered at the given moment in the register, form a set. However, all the beautiful women in Prague or brave men in Prague

form a semiset, since it is not clear who belongs to this collection and who not.

In the real world, we may find many other semisets. Almost each property defines a semiset of objects, e. g., people who are big, happy or sick. Many properties are naturally connected with a vagueness. Also, what we see and perceive can be vague and limited by a horizon. Objects described in this way may form a semiset, e. g. flowers I can see in the blooming meadow, all my friends, sounds I can hear.

Infinity

This interpretation differs from the normal one and corresponds more to the etymological origin of the word infinity. We will call *finite* those classes any part of which is surveyable and forms a set. Any finite class is a set.

$$\mathrm{Fin}(X) \Leftrightarrow (\forall Y)(Y \subseteq X \implies \mathrm{Set}(Y)).$$

On the other side, *infinite classes* include ungrasped parts, semisets. This phenomenon may occur also when watching large sets in the case when it is not possible to capture them clearly as a whole.

There are two different forms of infinity traditionally called denumerability and continuum.

A countable (denumerable) class, in a way, represents a road towards the horizon. Its beginning is clear and definite but it comes less and less clear and its end loses in a vagueness. A *countable class* is defined as an infinite class with a linear ordering such that each initial part (segment) is finite. For instance, a railway track with cross-ties leading straight to the horizon, days of our life we are to live or ever smaller and smaller reflections in two mirrors facing each other. The most important example is a class of natural numbers that will be discussed later.

The phenomenon of denumerability corresponds to a road towards the horizon. Though we get to the last point we can see, we can still go a bit further, the road will not disappear immediately. People have always tried to look a bit behind the horizon, to gain understanding and to overcome it in their thinking. This experience is expressed here by the important *axiom of prolongation* (see Axiom A6).

The other type of infinity, *continuum*, is based on the following experience. If we watch an object, however, are not able to distinguish individual elements which form it since they lie beyond the horizon of our perception. For example, the class of all geometric points in the plane, class of all atoms forming a table or grains of sand which together form a heap.

In fact the classical infinite mathematics, when applied to the real world, then solely to the above two types of infinity.

The intention of AST is to built on the natural world and human intuition. There is no reason for other types of infinity which are enforced in CST by its assumption that natural numbers form a set and that a power set is a set. That is why there are only two infinite cardinalities in AST: denumerability and continuum (see Axiom A8).

All examples from mathematical and real worlds are intentionally set out here together. They serve the purpose of inspiration to see where the idea of infinity comes from, they should be kept in mind when one deals with infinity.

The mathematical world is an ideal one, it is a perfect world of objective truths abstracted from all that is external. There is only little space for subjectivity of perception in it. That is why not all semisets from the real world may be interpreted directly.

The axiomatic system bellow describes that part of the AST which can be expressed in a strictly formal way. This basis provides space for extending AST by semisets which are parts of big, however, classically finite sets and thus make a lot of applications possible.

Axiomatic System of AST

[3] The language of AST uses symbols $\in$ and $=$, symbols X, Y, Z, … for class variables and symbols x, y, z, … for set variables. *Sets* are created by iteration from the empty set by Axiom A3. *Classes* are defined by formulas by Axiom A2. Every set is a class. Formally, a set is a class that is a member of another class:

$$\mathrm{Set}(X) \Leftrightarrow (\exists Y)(X \in Y).$$

AST is a theory with the following axioms:

- A1 (*extensionality*). $(X = Y) \Leftrightarrow (\forall Z)(Z \in X) \Leftrightarrow (Z \in Y)$;
- A2 (*existence of classes*). If ψ is a formula, then

$$(\exists Y)(\forall x)(x \in Y \Leftrightarrow \psi(x, X_1, \ldots, X_n));$$

- A3 (*existence of sets*).

$$\mathrm{Set}(\emptyset) \wedge (\forall x, y) Set(x \cup \{y\}).$$

A *set-formula* is a formula in which only set variables and constants occur.

- A4 (*induction*). If ψ is a set-formula, then $(\psi(\emptyset) \wedge (\forall x, y)(\psi(x) \Rightarrow \psi(x \cup \{y\})) \Rightarrow (\forall x)\, \psi(x)$.
- A5 (*regularity*). If ψ is a set-formula, then $(\exists x)\psi(x) \Rightarrow (\exists x)(\psi(x) \wedge (\forall y \in x)\neg\psi(y))$.

As usual, the class of *natural numbers* N is defined in the von Neumann way

$$N = \left\{ x\colon \begin{array}{c} (\forall y \in x)(y \subseteq x) \\ \wedge(\forall y, z \in x)(y \in z \vee y = z \vee z \in y) \end{array} \right\}$$

The class of *finite natural numbers* (*FN*) consists of the numbers represented by a finite set. They are accessible, easy to overlook and lie before the horizon:

$$FN = \{x \in N\colon\ \mathrm{Fin}(x)\}$$

FN forms a countable class in the sense described above. The class *FN* correspond to classical natural numbers and the class N to their nonstandard model. Both N and *FN* satisfy the axioms of Peano arithmetic.

Two classes X, Y are *equivalent* if there is a one-one mapping of X onto Y, i. e. $X \approx Y$.

- A6 (*prolongation*). Every countable function can be prolonged to a function which is a set, i. e. $(\forall F)((\mathrm{Fnc}(F) \wedge (F \approx FN)) \Rightarrow (\exists f)(\mathrm{Fnc}(f) \wedge F \subseteq f))$.

An easy corollary is that a countable class is a semiset. Also *FN* is a semiset and it can be prolonged to a set which is an element of N and which is greater than all finite natural numbers and so it represents an infinitely large natural number. Consequently, the class N is not countable.

The *universal class* V includes all sets created by iteration from the empty set.

- A7 (*choice*). The universal class V can be well ordered.
- A8 (*two cardinalities*). Every two infinite classes that are not countable are equivalent.

Thus, any infinite class is either equivalent to *FN* or N.

Using ultrapowers, the relative consistency of AST can be proved.

Rational and Real Numbers

Rational numbers Q are constructed in the usual way from N as the quotient field of the class $N \cup \{-n; n \in N\}$ Because N includes infinitely large numbers, Q includes infinitely small numbers.

Finite rational numbers FQ are similarly constructed from finite natural numbers *FN*. They include quantities that are before the horizon with respect to distance and depth. Surely $FQ \subseteq Q$.

We define that $x, y \in Q$ are *infinitely near* by

$$x \doteq y \Leftrightarrow (\forall n \in FN) \begin{cases} |x - y| < \frac{1}{n} \\ \vee(x > n \wedge y > n) \\ \vee(x < -n \wedge y < -n). \end{cases}$$

This relation is an equivalence. The corresponding partition classes are called *monads*. For $x \in Q$

$$\mathrm{Mon}(x) = \{y\colon\ y \doteq x\}\,.$$

Rational numbers x that are elements of Mon(0), i. e. ($x \dot{\in} 0$), are *infinitely small*. All monads are of the same nature except for the two limit ones. These consists of *infinitely large positive* and *negative* numbers. The class of bounded rational numbers is

$$BQ = \{x \in Q\colon\ (\exists n)((n \in FN) \wedge (|x| < n))\}$$

Now, it is easy and natural to construct *real numbers*:

$$\mathbb{R} = \{\mathrm{Mon}(x)\colon\ x \in BQ\}\,.$$

Real numbers built in this way display the same characteristics as real numbers in CST.

This motivation for expressing real numbers as monads of rational numbers corresponds rather to etymology than to the traditional interpretation. Rational numbers are constructed by reason, perfectly exact; their existence is purely abstract. On the other hand, real numbers are more similar to those that are used in the real world. If we say: one eighth of a cake, we surely do not expect it to be the ideal eighth, it is rather a portion which differs from the ideal one by a difference which is beyond the horizon of our perception. A similar situation occurs in the case of a pint of milk or twenty miles.

Infinitesimal Calculus

[12] Infinitesimal calculus in AST is based on the same point of view and intuition as that of its founders, I. Newton and G.W. Leibniz. It is so because infinitely small or infinitesimal quantities are naturally available in AST. For example, the limit of a function and the continuity in $a \in Q$ are defined, respectively, by:

$$\lim_{x \to a} f(x) = b$$
$$\Leftrightarrow (\forall x)((x \doteq a \wedge x \neq a) \Rightarrow f(x) \doteq b));$$
$$(\forall x)(x \doteq a \Rightarrow f(x) \doteq f(a)).$$

This topic is discussed in detail in [9]. As a method, these definitions were successfully used for teaching students.

Topology

Classes described by arbitrary formulas can be complex and difficult to capture. The easiest are sets, also classes described by using set-formulas, so-called *set-definable classes* (*Sd-classes*) can be described well. Semisets which are defined by a positive property (big, blue or happy and also distinguishable or to be a finite natural number) can be described as a countable union of Sd-classes, the so-called *σ-classes*. On the other hand, classes whose definition is based on negation (not big, not happy, indistinguishable), are the so-called *π-classes*—countable intersections of Sd-classes. A class which is at the same time π and σ is an Sd-class. Using combinations of π and σ, a set hierarchy can be described.

One of the most important tasks of mathematics is to handle the notion of the continuum. AST is based on the assumption that this phenomenon is caused by that of the indiscernibility of elements of the observed class. That is why, for the study of topology, the basic notion is a certain *relation of indiscernibility* ($\equiv$). Two elements are indiscernible if, when observed, available criteria that might distinguish them fail. It is a negative feature, therefore it must be a π-class. The relation of indiscernibility is naturally reflexive and symmetric. In pure mathematics, it is in addition transitive (because *FN* is closed under addition), thus it is an equivalence. This relation must also be compact, i. e. for each infinite set $u \subseteq \mathrm{dom}(\equiv)$ there are $x, y \in u$ such that $x \neq y \wedge x \equiv y$. The corresponding topological space is a compact metric space.

The relation of infinite nearness in rational numbers represents a special case of equivalence of indiscernibility.

Monads and *figures* correspond to phenomena of points and shapes, respectively:

$$\mathrm{Mon}(x) = \{y\colon\ y \equiv x\},$$
$$\mathrm{Fig}(X) = \{y\colon\ (\exists x \in X)(y \equiv x)\}.$$

Basic Definitions

Two classes X, Y are *separable*, $\mathrm{Sep}(X, Y) \Leftrightarrow (\exists Z)(\mathrm{Sd}(Z) \wedge \mathrm{Fig}(X) \subseteq Z \wedge \mathrm{Fig}(Y) \cap Z = \emptyset)$.

A *closure* $\overline{X}$ of a class X is defined as $\overline{X} = \{x\colon\ \neg\, \mathrm{Sep}(\{x\}, X)\}$.

A class X is *closed* if $X = \overline{X}$.

A set u is *connected* if $(\forall w)(\emptyset \neq = u \Rightarrow \mathrm{Fig}(w) \cap (u - w) \neq \emptyset)$.

It is quite easy to prove basic topological theorems. Also proofs of some classical theorems are much simpler here. For instance the *Sierpinski theorem*: If v is a connected set then $\mathrm{Fig}(v)$ cannot be expressed as a countable union of disjoint closed sets.

The *fundamental indiscernibility* $\overset{\circ}{=}_c$ is defined as follows. If c is a set then $x \overset{\circ}{=}_c y$ if for any set-formula ψ with the constants from c and for any x, it is $\psi\ (x) \Leftrightarrow \psi\ (y)$.

This relation has a special position. For any relation of indiscernibility $\equiv$ there is a set c such that $\overset{\circ}{=}_{\{c\}}$ is finer than $\equiv$ i. e. $\overset{\circ}{=}_{\{c\{} \subseteq \equiv$.

Motion

Unlike classical mathematics, the motion is captured in AST by any relation of indiscernibility $\equiv$.

Everybody knows the way films work. Pictures coming one after another are almost indiscernible from each other, however, when shown in a rapid sequence, the pictures start to move. The continuous motion may be viewed like this, as a sequence of indiscernible stages in certain time intervals.

A function d is a *motion of a point* in the time $\delta \in N$ if $\mathrm{dom}(f) = \delta \wedge (\forall \alpha < \delta)(d(\alpha) \equiv d(\alpha{+}1))$.

If $\delta \in FN$ then the point does not move, it can move only in an infinitely big time interval.

A sequence $\{d(\alpha)\colon \alpha \in \mathrm{dom}(d)\}$ is a sequence of states. The number $\delta = \mathrm{dom}(d)$ is the number of moments and $\mathrm{rng}(d)$ is the trace of a moving point.

A trace is a connected set and for each nonempty connected set u there is a motion of a point such that u is the trace of d.

A *motion of a set* is defined similarly, only the last condition is different: $(\forall \alpha < \delta)(\mathrm{Fig}(d(\alpha)) = \mathrm{Fig}(d(\alpha{+}1)))$.

The following theorem is proved in [10,11]: Each motion of a set may be divided into motions of points. This does not involve only the mechanical motion, but any motion describing a continuous change. Thus, for example, even the growth of a tree from a planted seed may be divided into movements of individual points while all of their initial stages are already contained in the seed. In addition, it is possible to describe conditions under which such a change is still continuous.

Utility Theory

[7] The utility theory is one of nice examples of applying AST. Its aim is to find a valuation of elements of a class S. There is a preference relation $\succ$ on linear combinations of elements of S with finite rational coefficients, i. e. on the class

$$\left\{ \sum_{i=1}^{n} \alpha_i u_i : \begin{array}{c} (n \in FN) \\ \wedge(\forall i)((i \le n)(u_i \in S) \wedge (\alpha_i \in FQ)) \\ \wedge \sum_{i=1}^{n} \alpha_i = 1 \end{array} \right\}.$$

An interpretation of a combination is a game in which every u_i can be won with the probability α_i. The *preference relation* $\succ$ declares which of the two games is preferred.

The *valuation* is a function F from the class S to Q for which

$$\sum_{i=1}^{n} \alpha_i u_i \succ \sum_{j=1}^{m} \beta_j u_j \Leftrightarrow \sum_{i=1}^{n} \alpha_i F(u_i) > \sum_{j=1}^{m} \beta_j F(u_j).$$

It is not necessary to require the so-called Archimedes property on the relation of preference thanks to the possibility of using infinitely small and infinitely large rational numbers. It is possible to capture finer and more complex relations than in classic mathematics, e. g. the fact that the value of one element is incomparably higher than that of another element or it is possible to compare infinitely small differences of values.

For each class S with a preference relation a valuation may be found. Such a valuation is not uniquely defined, it is possible to construct it so that $\mathrm{rng}(F) \subseteq N$.

Conclusion

The aim of this short survey is to demonstrate the basic ideas of AST. Yet, there are other areas of mathematics which were studied in it, for instance measurability [8], ultrafilters [6], endomorphic universes [5] and automorphisms of natural numbers [2], representability [1] metamathematics [3] and models of AST [4].

See also

- ▶ Boolean and Fuzzy Relations
- ▶ Checklist Paradigm Semantics for Fuzzy Logics
- ▶ Finite Complete Systems of Many-valued Logic Algebras
- ▶ Inference of Monotone Boolean Functions
- ▶ Optimization in Boolean Classification Problems
- ▶ Optimization in Classifying Text Documents

References

1. Mlček J (1979) Valuation of structures. Comment Math Univ Carolinae 20:681–695
2. Mlček J (1985) Some automorphisms of natural numbers in AST. Comment Math Univ Carolinae 26:467–475
3. Sochor A (1992) Metamathematics of AST. From the logical point of view 1:61–75
4. Sochor A, Pudlák P (1984) Models of AST. J Symbolic Logic 49:570–585
5. Sochor A, Vopěnka P (1979) Endomorfic universes and their standard extensions. Comm Math Univ Carolinae 20:605–629
6. Sochor A, Vopěnka P (1981) Ultrafilters of sets. Comment Math Univ Carolinae 22:698–699
7. Trlifajová K, Vopěnka P (1985) Utility theory in AST. Comment Math Univ Carolinae 26:699–711
8. Čuda K (1986) The consistency of measurability of projective semisets. Comment Math Univ Carolinae 27:103–121
9. Čuda K, Sochor A, Vopěnka P, Zlatoš P (1989) Guide to AST. Proc. First Symp. Mathematics in AST, Assoc. Slovak Mathematicians and Physicists, Bratislava
10. Vopénka P (1979) Mathematics in AST. Teubner, Leipzig
11. Vopénka P (1989) Introduction to mathematics in AST. Alfa Bratislava, Bratislava
12. Vopénka P (1996) Calculus infinitesimalis-pars prima. Práh Praha, Praha

Approximation of Extremum Problems with Probability Functionals

APF

Riho Lepp
Tallinn Technical University, Tallinn, Estonia

MSC2000: 90C15

Article Outline

Keywords
See also
References

Keywords

Discrete approximation; Probability functionals

To ensure a certain level of reliability for the solution of an extremum problem under uncertainty it has become a spread approach to introduce probabilistic (chance) cost and/or constraints into the model. The stability analysis of chance constraint problems is rather complicated due to complicated properties of the *probability function* $v_t(x)$, defined as

$$v_t(x) = \mathsf{P}\{s\colon\ f(x,s) \le t\}\,. \tag{1}$$

Here $f(x, s)$ is a real valued function, defined on $\mathbf{R}^r \times \mathbf{R}^r$, t is a fixed level of reliability, s is a random vector and P denotes probability. The function $v_t(x)$ is never convex, only in some cases (e. g., $f(x, s)$ linear in s and distribution of the random parameter s normal), it is quasiconvex. Note that for a fixed x function $v_t(x)$, as a function of t, is the distribution function of the random variable $f(x, s)$.

The 'inverse', the *quantile function* $w_\alpha(x)$, to the probability function $v_t(x)$ is defined in such a way that the probability level α, $0 < \alpha < 1$, is fixed earlier, and the purpose is to minimize the reliability level t:

$$w_\alpha(x) = \min_t \{t\colon\ \mathsf{P}\{s\colon\ f(x,s) \le t\} \ge \alpha\}\,. \tag{2}$$

Varied examples of extremum problems with probability and quantile functions are presented in [7] and in [8]. Some of these models have such a complicated structure, see [8, Chap. 1.8], about correction of a satellite orbit, that we are forced to look for a solution x from a certain class of strategies, that means, the solution x itself depends on the random parameter s, $x = x(s)$.

This class of probability functions was introduced to *stochastic programming* by E. Raik, and lower semicontinuity and continuity properties of $v_t(x)$ and $w_\alpha(x)$ in Lebesgue L^p-spaces, $1 \le \mathrm{p} < \infty$, were studied in [12]. Simultaneously, in [4] problems with various classes of solutions $x(s)$ (measurable, continuous, linear, etc) were considered. Since the paper [4] solutions $x(s)$ are called *decision rules*, and we will follow also this terminology.

Differently from [4], here we will consider approximation of a decision rule $x(s)$ by sequences of vectors $\{x_n\}$, $x_n = (x_{1n}, \ldots, x_{nn})$, $n = 1, 2, \ldots$, with increasing dimension in order to maximize the value of the probability functional $v(x)$ under certain set C of decision rules. It will be assumed that the set C will be bounded in the space $L^1(S, \Sigma, \sigma) = L^1(\sigma)$ of integrable functions $x(s)$, $x \in L^1(\sigma)$:

$$\max_{x\in C} v_t(x) = \max_{x\in C} \mathsf{P}\{s\colon\ f(x(s),s) \le t\}\,. \tag{3}$$

Here S is the support of random variable s with distribution (probability measure) $\sigma(\cdot)$ and Σ denotes the sigma-algebra of Borel measurable sets from $\mathbf{R}^r$.

Due to technical reasons we are forced to assume that the random parameter s has bounded support $S \subset \mathbf{R}^r$, diam $S < \infty$, and its distribution σ is atomless,

$$\sigma\{s\colon\ |s - s_0| = \mathrm{const}\} = 0, \quad \forall s_0 \in \mathbb{R}^r. \tag{4}$$

Since the problem (3) is formulated in the function space $L^1(\sigma)$ of σ-integrable functions, the first step in its solution is the approximation step where we will replace the initial problem (3) by a sequence of finite-dimensional optimization problems with increasing dimension. Second step, solution methods were considered in a series of papers of the author (see, e. g., [9]), where the gradient projection method was suggested together with simultaneous Parzen–Rosenblatt kernel-type smooth approximation of the discontinuous integrand from (1).

There are several ways to divide the support S of the probability measure σ into smaller parts in discretization, e. g., taking disjoint subsets S_j, $j = 1, \ldots, k$, of S

from the initial sigma-algebra Σ as in [11], or using in the partition of S only convex sets from Σ, as in [5].

We will divide the support S into smaller parts by using only sets A_{in}, $i = 1, \ldots, n$, $n \in N = \{1, 2, \ldots\}$, with σ-measure zero of their boundary, i. e., $\sigma\ (\mathrm{int} A_{in}) = \sigma\ (A_{in}) = \sigma\ (\mathrm{cl} A_{in})$, where int A and cl A denote topological interior and closure of a set A, respectively. Such division is equivalent to *weak convergence* of a sequence of *discrete measures* $\{(m_n, s_n)\}$ to the initial probability measure σ, see, e. g. [14]:

$$\sum_{i=1}^{n} h(s_{in}) m_{in} \to \int_S h(s)\, \sigma(ds), \quad n \in \mathbb{N}, \tag{5}$$

for any continuous on S function $h(s)$, $h \in C(S)$.

The usage of the weak convergence of discrete measures in stochastic programming has its disadvantages and advantages. An example in [13] shows that, in general, the stability of a probability function with respect to weak convergence cannot be expected without additional smoothness assumptions on the measure σ. This is one of the reasons, why we should use only continuous measures with the property (4). An advantage of the usage of the weak convergence is that it allows us to apply in the approximation process instead of conditional means [11] the more simple, grid point approximation scheme.

Since the functional $v_t(x)$ is not convex, we are not able to exploit in the stability analysis of discrete approximation of the problem (3) the more convenient, weak topology, but only the strong (norm) topology. As the first step we will approximate $v_t(x)$ so, that the discrete analogue of continuous convergence of a sequence of approximate functionals will be guaranteed.

Schemes of *stability analysis* (e. g., finite-dimensional approximations) of extremum problems in Banach spaces require from the sequence of solutions of 'approximate' problems certain kind of compactness. Assuming that the constraint set C is compact in $L^1(\sigma)$, we, as the second step, will approximate the set C by a sequence of finite-dimensional sets $\{C_n\}$ with increasing dimension so, that the sequence of solutions of approximate problems is compact in a certain (discrete convergence) sense in $L^1(\sigma)$. Then the approximation scheme for the discrete approximation of (3) will follow formed schemes of approximation of extremum problems in Banach spaces, see e. g. [2,3,15].

Redefine the functional $v_t(x)$ by using the Heaviside zero-one function χ:

$$v_t(x) = \int_S \chi(t - f(x(s), s))\, \sigma(ds), \tag{6}$$

where

$$\chi(t - f(x(s), s)) = \begin{cases} 1 & \text{if } f(x(s), s) \le t, \\ 0 & \text{if } f(x(s), s) > t. \end{cases}$$

Since the integrand $\chi(\cdot)$ itself, as a zero-one function, is discontinuous, we will assume that the function $f(x, s)$ is continuous both in (x, s) and satisfies following growth and 'platform' conditions:

$$|f(x, s)| \le a(s) + \alpha\, |x|\,, \tag{7}$$

$$a \in L^1(\sigma), \quad \alpha > 0,$$

$$\sigma\,\{s\colon\ f(x, s) = \mathrm{const}\} = 0, \tag{8}$$

$$\forall (x, s) \in \mathbb{R}^r \times S.$$

The continuity assumption is technical in order to simplify the description of the approximation scheme below. The growth condition (7) is essential: without it the superposition operator $f(x) = f(x(s), s)$ will not map an element from L^1 to L^1 (is even not defined). Condition (8) means that the function $f(x, s)$ should not have horizontal platforms with positive measure.

Constraint set C is assumed to be a set of integrable functions $x(s)$, $x \in L^1(\sigma)$, with properties

$$\int_S |x(s)|\ \sigma(ds) \le M < \infty, \quad \forall x \in C \tag{9}$$

for some $M > 0$ (C is bounded in $L^1(\sigma)$);

$$\int_D |x(s)| \le K\sigma(D), \quad \forall x \in C, \quad D \in \Sigma \tag{10}$$

for some $K > 0$;

$$(x(s) - x(t), s - t) \ge 0 \quad \text{for a.a. } s, t \in S \tag{11}$$

(functions $x \in C$ are monotone almost everywhere and a.a. denotes abbreviation of 'almost all').

Conditions (9), (10) guarantee that the set C is weakly compact (i. e., compact in the (L^1, L^∞)-topology, see, e. g., [6, Chap. 9.1.2]). Condition (11) guarantees now, following [1, Lemma 3], that the set C is strongly compact in $L^1(\sigma)$. Then, following [11], we can conclude that assumptions (7)–(11) together with

atomless assumption (4) for the measure σ guarantee the existence of a solution of problem (3) in the Banach space $L^1(\sigma)$ of σ-integrable functions (the cost functional $v_t(x)$ is continuous in x and the constraint set C is compact in $L^1(\sigma)$).

Since approximate problems will be defined in $\mathbf{R}^{rn}$, we should define a system of connection operators $\mathcal{P} = \{p_n\}$ between spaces $L^1(\sigma)$ and $\mathbf{R}^{rn}$, $n \in \mathbf{N}$. In L^p-spaces, $1 \le p \le \infty$, systems of connection operators should be defined in a piecewise integral form (as conditional means):

$$(p_n x)_{in} = \sigma(A_{in})^{-1} \int_{A_{in}} x(s)\, \sigma(ds), \tag{12}$$

where $i = 1, \ldots, n$, and sets A_{in}, $i = 1, \ldots, n$, $n \in \mathbf{N}$, that define connection operators (12), satisfy following conditions A1)–A7):

A1) $\sigma(A_{in}) > 0$;
A2) $A_{in} \cap A_{jn} = \emptyset$, $i \ne j$;
A3) $\cup_{i=1}^{n} A_{in} = S$;
A4) $\sum_{i=1}^{n} |m_{in} - \sigma(A_{in})| \to 0$, $n \in \mathbf{N}$;
A5) $\max_i \operatorname{diam} A_{in} \to 0$, $n \in \mathbf{N}$;
A6) $s_{in} \in A_{in}$;
A7) $\sigma(\operatorname{int} A_{in}) = \sigma(A_{in}) = \sigma(\operatorname{cl} A_{in})$.

Remark 1 Weak convergence (5) is equivalent to the partition $\{\mathcal{A}_n\}$ of S, $\mathcal{A}_n = \{A_{1n}, \ldots, A_{nn}\}$, with properties A1)–A7), see [14].

Remark 2 Collection of sets $\{A_{in}\}$ with the property A7) constitutes an algebra $\Sigma_0 \subset \Sigma$, and if $S = [0, 1]$ and if σ is Lebesgue measure on $[0, 1]$, then integrability relative to $\sigma|_{\Sigma_0}$ means Riemann integrability.

Define now the *discrete convergence* for the space $L^1(\sigma)$ of σ-integrable functions.

Definition 3 A sequence of vectors $\{x_n\}$, $x_n \in \mathbf{R}^{rn}$, *$\mathcal{P}$-converges* (or *converges discretely*) to an integrable function $x(s)$, if

$$\sum_{i=1}^{n} |x_{in} - (p_n x)_{in}|\, m_{in} \to 0, \quad n \in \mathbb{N}. \tag{13}$$

Remark 4 Note that in the space $L^1(\sigma)$ of σ-integrable functions we are also able to use the projection methods approach, defining convergence of $\{x_n\}$ to $x(s)$ as follows:

$$\int_S \left| x(s) - \sum_{i=1}^{n} x_{in} \chi_{A_{in}}(s) \right| \sigma(ds) \to 0, \quad n \in \mathbb{N}.$$

Remark 5 Projection methods approach does not work in the space $L^\infty(\sigma)$ of essentially bounded measurable functions with vraisup-norm topology ($L^\infty(\sigma)$ is a non-separable Banach space and the space $C(S)$ of continuous functions is not dense there).

We need the space $L^\infty(\sigma)$, which is the topological dual to the space $L^1(\sigma)$ of σ-integrable functions, in order to define also the discrete analogue of the weak convergence in $L^1(\sigma)$.

Definition 6 A Sequence of vectors $\{x_n\}$, $x_n \in \mathbf{R}^{rn}$, $n \in \mathbf{N}$, *$w\mathcal{P}$-converges* (or *converges weakly discretely*) to an integrable function $x(s)$, $x \in L^1(\sigma)$, if

$$\sum_{i=1}^{n} (z_{in}, x_{in}) m_{in} \to \int_S (z(s), x(s))\, \sigma(ds), \quad n \in \mathbb{N}, \tag{14}$$

for any sequence $\{z_n\}$ of vectors, $z_n \in \mathbf{R}^{rn}$, $n \in \mathbf{N}$, and function $z(s)$, $z \in L^\infty(\sigma)$, such that

$$\max_{1 \le i \le n} |z_{in} - (p_n z)_{in}| \to 0, \quad n \in \mathbb{N}. \tag{15}$$

In order to formulate the discretized problem and to simplify the presentation, we will assume that in partition $\{\mathcal{A}_n\}$ of S, where $\mathcal{A}_n = \{A_{1n}, \ldots, A_{nn}\}$, with properties A1)–A7), in property A4) we will identify m_{in} and $\sigma(A_{in})$, i. e. $m_{in} = \sigma(A_{in})$ (e. g. squares with decreasing diagonal in $\mathbf{R}^2$).

Discretize now the probability functional $v_t(x)$:

$$v_{tn}(x_n) = \sum_{i=1}^{n} \chi(t - f(x_{in}, s_{in})) m_{in}, \tag{16}$$

and formulate the discretized problem:

$$\max_{x_n \in C_n} v_{tn}(x_n) = \max_{x_n \in C_n} \sum_{i=1}^{n} \chi(t - f(x_{in}, s_{in})) m_{in}, \tag{17}$$

where constraint set C_n will satisfy discrete analogues of conditions (9)–(11), covered to the set C:

$$\sum_{i=1}^{n} |x_{in}|\, m_{in} \le M \quad \forall x_n \in C_n, \tag{18}$$

$$\sum_{i \in I_n} |x_{in}|\, m_{in} \le K \sum_{i \in I_n} m_{in}, \quad \forall x_n \in C_n, \quad \forall I_n \subset \{1, \ldots, n\}, \tag{19}$$

$$\sum_{k=1}^{r}(x^k_{i_k n} - x^k_{j_k n})(i_k - j_k) \geq 0, \quad \forall i_k, j_k : \quad i_k < j_k \ , \tag{20}$$

and such that $0 \leq i_k, j_k \leq n$, $\forall n \in \mathbf{N}$.

Definition 7 A sequence of sets $\{C_n\}$, $C_n \subset \mathbf{R}^{rn}$, $n \in \mathbf{N}$, converges to the set $C \subset L^1(\sigma)$ in the *discrete Mosco sense* if

1) for any subsequence $\{x_n\}$, $n \in \mathbf{N}' \subset \mathbf{N}$, such that $x_n \in C_n$, from convergence $w\mathcal{P}\text{-lim } x_n = x$, $n \in \mathbf{N}$, it follows that $x \in C$;
2) for any $x \in C$ there exists a sequence $\{x_n\}$, $x_n \in C_n$, which $\mathcal{P}$-converges to x, $\mathcal{P}\text{-lim } x_n = x$, $n \in \mathbf{N}$.

Remark 8 If in the above definition also 'for any' part 1) is defined for $\mathcal{P}$-convergence of vectors, then it is said that sequence of sets $\{C_n\}$ converges to the set C in the *discrete Painlevé–Kuratowski sense*.

Denote optimal values and optimal solutions of problems (3) and (17) by v^*, x^* and v^*_n, x^*_n, respectively.

Let function $f(x, s)$ be continuous in both variables (x, s) and satisfy growth and platform conditions (7) and (8). Then from convergence $\mathcal{P}\text{-lim } x_n = x$, $n \in \mathbf{N}$, for any monotone a.e. function $x(s)$, it follows convergence $v_n(x_n) \to v(x)$, $n \in \mathbf{N}$.

Verification of this statement is quite lengthy and technically complicated: we should first approximate discontinuous function $\chi(t - f(x, s))$ by continuous function $\chi_c(t - f(x, s))$ in the following way:

$$\chi_c(t - f(x,s)) = \begin{cases} 1 & \text{if } f(x,s) \leq t, \\ 1 - \delta^{-1}[f(x,s) - t] & \text{if } t < f(x,s) \leq t + \delta, \\ 0 & \text{if } f(x,s) > t + \delta \end{cases}$$

for some (small) δ, and then a discontinuous solution $x(s)$, $x \in L^1(\sigma)$, by continuous function $x_c(s)$ (in L^1-norm topology).

Let constraint sets C and C_n satisfy conditions (9)–(11) and (18)–(20), respectively. Let discrete measures $\{(m_n, s_n)\}$ converge weakly to the measure σ. Then the sequence of sets $\{C_n\}$ converges to the set C in the discrete Painlevé–Kuratowski sense.

Verification of this statement relies on the two following convergences:

1) sequence of sets, determined by inequalities (18), (19) converges, assuming weak convergence of discrete measures (5), in discrete Mosco sense to the weakly compact in $L^1(\sigma)$ set, determined by inequalities (9), (10);
2) adding to both, approximate and initial sets of admissible solutions monotonicity conditions (20) and (11), respectively, we can guarantee the discrete convergence of sequence $\{C_n\}$ to C in Painlevé–Kuratowski sense.

Now we can formulate the discrete approximation conditions for a stochastic programming problem with probability cost function in the class of integrable decision rules.

Let function $f(x, s)$ be continuous in both variables (x, s) and satisfy growth and platform conditions (7) and (8), constraint set C satisfy conditions (9)–(11) and let discrete measures $\{(m_n, s_n)\}$ converge weakly to the atomless measure σ. Then $v^*_n \to v^*$, $n \in \mathbf{N}$, and sequence of solutions $\{x^*_n\}$ of approximate problems (17) has a subsequence, which converges discretely to a solution of the initial problem (3).

Remark 9 The usage of the space $L^1(\sigma)$ of integrable functions is essential. In reflexive L^p-spaces, $1 < p < \infty$, serious difficulties arise with application of the strong (norm) compactness criterion for a maximizing sequence.

As a rule, problems with probability cost function are maximized, whereas stochastic programs with quantile cost are minimized, see, e. g., [8,10].

Consider at last discrete approximation of the quantile minimization problem (2):

$$\min_{x \in C} w_\alpha(x) = \min_{x \in C} \min_t \{\mathsf{P}(f(x(s), s) \leq t) \geq \alpha\}, \tag{21}$$

It was verified in [10] that under certain (quasi)-convexity-concavity assumptions the quantile minimization problem (21) is equivalent to the following Nash game:

$$\max_{x \in C} v_t(x) = J^*_1, \tag{22}$$

$$\min_t (v_t(x) - \alpha)^2 = J^*_2. \tag{23}$$

Discretizing $v_t(x)$ as in (16) and $w_\alpha(x)$ as

$$w_{\alpha n}(x_n) = \min_t \left\{ \sum_{i=1}^{n} \chi(t - f(x_{in}, s_{in}))m_{in} \geq \alpha \right\},$$

we can, analogously to the probability functional approximation, approximate the quantile minimization problem (21) too. In other words, to replace the Nash game (22), (23) with the following finite-dimensional game:

$$\max_{x_n \in C_n} v_{tn}(x_n) = J_{1n}^*, \tag{24}$$

$$\min_t (v_{tn}(x_n) - \alpha)^2 = J_{2n}^*. \tag{25}$$

Verification of convergences $J_{1n}^* \to J_1^*$ and $J_{2n}^* \to J_2^*$, $n \in \mathbf{N}$, is a little bit more labor-consuming compared with approximate maximization of probability functional $v_t(x)$, since we should guarantee also convergence of the sequence of optimal quantiles $\{t_n^*\}$ of minimization problems (25).

See also

- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-Shaped Method for Two-Stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-Stage Stochastic Programs with Recourse

References

1. Banaš J (1989) Integrable solutions of Hammerstein and Urysohn integral equations. J Austral Math Soc (Ser A) 46:61–68
2. Daniel JW (1971) The approximate minimization of functionals. Prentice-Hall, Englewood Cliffs
3. Esser H (1973) Zur Diskretisierung von Extremalproblemen. Lecture Notes Math, vol 333. Springer, Berlin, pp 69–88
4. Garstka J, Wets RJ-B (1974) On decision rules in stochastic programming. Math Program 7:117–143
5. Hernandez-Lerma O, Runggaldier W (1994) Monotone approximations for convex stochastic control problems. J Math Syst, Estimation and Control 4:99–140
6. Ioffe AD, Tikhomirov VM (1979) Theory of extremal problems. North-Holland, Amsterdam
7. Kall P, Wallace SW (1994) Stochastic programming. Wiley, New York
8. Kibzun AI, Kan YS (1995) Stochastic programming problems with probability and quantile functions. Wiley, New York
9. Lepp R (1983) Stochastic approximation type algorithm for the maximization of the probability function. Proc Acad Sci Estonian SSR Phys Math 32:150–156

10. Malyshev VV, Kibzun AI (1987) Analysis and synthesis of high precision aircraft control. Mashinostroenie, Moscow, Moscow
11. Olsen P (1976) Discretization of multistage stochastic programming problems. Math Program Stud 6:111–124
12. Raik E (1972) On stochastic programming problem with probability and quantile functionals. Proc Acad Sci Estonian SSR Phys Math 21:142–148
13. Römisch W, Schultz R (1988) On distribution sensitivity in chance constrained programming. Math Res 45:161–168. Advances in Mathematical Optimization, In: Guddat J et al (eds)
14. Vainikko GM (1971) On convergence of the method of mechanical curvatures for integral equations with discontinuous kernels. Sibirsk Mat Zh 12:40–53
15. Vasin VV (1982) Discrete approximation and stability in extremal problems. USSR Comput Math Math Phys 22:57–74

Approximation of Multivariate Probability Integrals

TAMÁS SZÁNTAI
Technical University, Budapest, Hungary

MSC2000: 65C05, 65D30, 65Cxx, 65C30, 65C40, 65C50, 65C60, 90C15

Article Outline

Keywords
Lower and Upper Bounds
Monte-Carlo Simulation Algorithm
One- and Two-Dimensional Marginal Distribution Functions
Examples
Remarks
See also
References

Keywords

Boole–Bonferroni bounds; Hunter–Worsley bounds; Approximation; Probability integrals; Variance reduction; Probabilistic constrained stochastic programming

Approximation of *multivariate probability integrals* is a hard problem in general. However, if the domain of the probability integral is multidimensional interval, then the problem reduces to the approximation of *multivariate probability distribution function* values.

Lower and Upper Bounds

Let $\xi^{\mathsf{T}} = (\xi_1, \ldots, \xi_n)$ be a random vector with given multivariate probability distribution. Introducing the events

$$A_1 = \{\xi_1 < x_1\}, \ldots, A_n = \{\xi_n < x_n\},$$

where $x_1, \ldots, x_n$ are arbitrary real values the multivariate probability distribution function of the random vector ξ can be expressed in the following way:

$$\begin{aligned} &F(x_1, \ldots, x_n) \\ &= \mathsf{P}(\xi_1 < x_1, \ldots, \xi_n < x_n) \\ &= \mathsf{P}(A_1 \cap \cdots \cap A_n) \\ &= 1 - \mathsf{P}(\overline{A}_1 \cup \cdots \cup \overline{A}_n) \\ &= 1 - \overline{S}_1 + \overline{S}_2 - \cdots + (-1)^n \overline{S}^n, \end{aligned}$$

where

$$\overline{A}_i = \{\xi_i \geq x_i\}, \quad i = 1, \ldots, n,$$

and

$$\overline{S}_k = \sum_{1 \leq i_1 < \cdots < i_k \leq n} \mathsf{P}(\overline{A}_{i_1} \cap \cdots \cap \overline{A}_{i_k}), k = 1, \ldots, n\,.$$

First one shows that $\overline{S}_1, \overline{S}_2$ and so the individual probabilities $\mathsf{P}(\overline{A}_i)$, $i = 1, \ldots, n$, $\mathsf{P}(\overline{A}_i \cap \overline{A}_j)$, $i = 1, \ldots, n-1$, $j = i+1, \ldots, n$, involved in them can be expressed by $F_i(x_i)$, $i = 1, \ldots, n$, and $F_{ij}(x_i, x_j)$, $i = 1, \ldots, n-1, j = i+1, \ldots, n$, the *one-* and *two-dimensional marginal probability distribution functions* of the random vector ξ. One has

$$\begin{aligned} \overline{S}_1 &= \sum_{i=1}^{n} \mathsf{P}(\overline{A}_i) = \sum_{i=1}^{n} \mathsf{P}(\xi_i \geq x_i) \\ &= n - \sum_{i=1}^{n} \mathsf{P}(\xi_i < x_i) = n - \sum_{i=1}^{n} F_i(x_i) \end{aligned}$$

and

$$\begin{aligned}\overline{S}_2 &= \sum_{1\le i<j\le n} \mathsf{P}(\overline{A}_i \cap \overline{A}_j)\\ &= \sum_{1\le i<j\le n} \mathsf{P}(\xi_i \ge x_i,\ \xi_j \ge x_j)\\ &= \sum_{1\le i<j\le n} \{1-\mathsf{P}(\xi_i < x_i)\\ &\quad -\mathsf{P}(\xi_j < x_j) + \mathsf{P}(\xi_i < x_i, \xi_j < x_j)\}\\ &= \frac{n(n-1)}{2} - (n-1)\sum_{i=1}^{n} F_i(x_i)\\ &\quad + \sum_{1\le i<j\le n} F_{ij}(x_i, x_j).\end{aligned}$$

So if one can calculate the one- and two-dimensional marginal probability distribution functions of the random vector ξ then one can bound the multivariate probability distribution function by the very simple bounds given by C.E. Bonferroni [1]:

$$1-\overline{S}_1 \le F(x_1,\ldots,x_n) \le 1-\overline{S}_1+\overline{S}_2,$$

or by the sharp bounds, called *Boole–Bonferroni bounds* discovered independently by many authors (see [11] for a summary):

$$\begin{aligned}&1-\overline{S}_1+\frac{2}{n}\overline{S}_2\\ &\le F(x_1,\ldots,x_n)\\ &\le 1-\frac{2}{k^*+1}\overline{S}_1+\frac{2}{k^*(k^*+1)}\overline{S}_2,\end{aligned}$$

where

$$k^* = 1+\left\lfloor \frac{2\overline{S}_2}{\overline{S}_1}\right\rfloor.$$

When applying the above bounds usually the upper bound proves to be sharper. However one can improve the lower bound by the application of the bound discovered independently by D. Hunter [5] and K.J. Worsley [18]. This bound is an upper bound for $\mathsf{P}(\overline{A}_1\cup\cdots\cup\overline{A}_n)$ by the use of $\overline{S}_1$ and the individual probabilities $\mathsf{P}(\overline{A}_i\cap\overline{A}_j)$, $1\le i<j\le n$. It is constructed in the following way. Construct a nonoriented complete graph with n nodes and assign to node i the event $\overline{A}_i$ (or the probability $\mathsf{P}(\overline{A}_i)$) and to arc (i,j) the weight $\mathsf{P}(\overline{A}_i\cap\overline{A}_j)$. Let T^* be a maximum weight spanning tree in this nonoriented complete graph then one has

$$\mathsf{P}(\overline{A}_1\cup\cdots\cup\overline{A}_n) \le \overline{S}_1 - \sum_{(i,j)\in T^*}\mathsf{P}(\overline{A}_i\cap\overline{A}_j),$$

which is called the *Hunter–Worsley upper bound*. This results the following lower bound on the multivariate probability distribution function:

$$1-\overline{S}_1+\sum_{(i,j)\in T^*}\mathsf{P}(\overline{A}_i\cap\overline{A}_j) \le F(x_1,\ldots,x_n).$$

The individual probabilities $\mathsf{P}(\overline{A}_i\cap\overline{A}_j)$, $1\le i<j\le n$, can be stored when one calculates the value of $\overline{S}_2$ and the maximum weight spanning tree can be found by several fast algorithms, for example by *Kruskal's algorithm*, see [9]. Now one has three lower and two upper bounds on the multivariate probability distribution function and all of them are computable if the one- and two-dimensional marginal probability distribution functions are known. Let us denote these bounds in the following way:

$$\begin{aligned}L_1 &= 1-\overline{S}_1,\\ L_2 &= 1-\overline{S}_1+\frac{2}{n}\overline{S}_2,\\ L_3 &= 1-\overline{S}_1+\sum_{(i,j)\in T^*}\mathsf{P}(\overline{A}_i\cap\overline{A}_j),\\ U_1 &= 1-\overline{S}_1+\overline{S}_2,\\ U_2 &= 1-\frac{2}{k^*+1}\overline{S}_1+\frac{2}{k^*(k^*+1)}\overline{S}_2.\end{aligned}$$

As one has $L_1\le L_2\le L_3$ and $U_2\le U_1$, the best lower bound is L_3 and the best upper bound is U_2.

Monte-Carlo Simulation Algorithm

One can take the differences between the multivariate probability distribution function and its lower and upper bounds introduced before:

$$\begin{aligned}&F(x_1,\ldots,x_n)-L_1 = \overline{S}_2-\overline{S}_3+\cdots+(-1)^n\overline{S}_n,\\ &F(x_1,\ldots,x_n)-L_2\\ &\quad=\left(1-\frac{2}{n}\right)\overline{S}_2-\overline{S}_3+\cdots+(-1)^n\overline{S}_n,\end{aligned}$$

$$F(x_1,\dots,x_n) - L_3$$
$$= -\sum_{(i,j)\in T^*} \mathsf{P}(\overline{A}_i \cap \overline{A}_j) + \overline{S}_2 - \overline{S}_3 + \cdots + (-1)^n \overline{S}_n,$$
$$F(x_1,\dots,x_n) - U_1 = \overline{S}_3 + \cdots + (-1)^n \overline{S}_n,$$
$$F(x_1,\dots,x_n) - U_2$$
$$= \left(\frac{2}{k^*+1} - 1\right)\overline{S}_1 + \left(1 - \frac{2}{k^*(k^*+1)}\right)\overline{S}_2 - \overline{S}_3 + \cdots + (-1)^n \overline{S}_n .$$

A *Monte-Carlo simulation procedure* of the multivariate probability distribution function value based on the estimation of the differences above will be given. First however the so called crude Monte-Carlo simulation procedure will be described. Let the random vectors $(\xi_1^s, \dots, \xi_n^s)$, $s = 1, \dots, S$, be distributed according to the multivariate probability distribution function to be approximated. One must check the inequalities $\xi_1^s < x_1, \dots, \xi_n^s < x_n$ for all sample elements, $s = 1, \dots, S$. For this purpose let be defined the random values

$$\nu_0^s = \begin{cases} 1 & \text{if } \xi_1^s < x_1, \dots, \xi_n^s < x_n, \\ 0 & \text{otherwise,} \end{cases} \quad s = 1, \dots, S.$$

These random values are identically distributed and stochastically independent. All of them take on the value 1 with probability equal to the approximated multivariate probability distribution function value. The sum of them has binomial probability distribution with parameters S and $F(x_1, \dots, x_n)$. So the random variable

$$\nu_0 = \frac{1}{S}(\nu_0^1 + \cdots + \nu_0^S)$$

has expected value $\mathsf{P} = F(x_1, \dots, x_n)$ and variance $\frac{\mathsf{P}(1-\mathsf{P})}{S}$. This is why ν_0 can be regarded as an estimate, the so called crude Monte-Carlo estimate of $F(x_1, \dots, x_n)$. If one introduces κ^s as the number of those $\xi_1^s < x_1, \dots, \xi_n^s < x_n$ inequalities which are not fulfilled, i. e. the number of those $\xi_1^s \geq x_1, \dots, \xi_n^s \geq x_n$ inequalities which are fulfilled, or the number of those $\overline{A}_1^s, \dots, \overline{A}_n^s$ events which occur, $s = 1, \dots, S$, the ν_0^s random values can be expressed as

$$\nu_0^s = \begin{cases} 1 & \text{if } \kappa^s = 0, \\ 0 & \text{otherwise} \end{cases} \quad s = 1, \dots, S,$$

and on the other hand for the *binomial moments* of κ^s one has

$$\mathsf{E}\left[\binom{\kappa^s}{k}\right] = \overline{S}_k, \; k = 0, \dots, n, \quad s = 1, \dots, S.$$

The simplest proof of these equalities was given by L. Takács [17] and it was reproduced by A. Prékopa in [11]. If the random numbers λ^s, $s = 1, \dots, S$, are also introduced as the number of those $\overline{A}_i \cap \overline{A}_j = \{\xi_i^s \geq x_i, \xi_j^s \geq x_j\}$, $(i,j) \in T^*$, events which occur then for the expected value of λ^s one has

$$\mathsf{E}(\lambda^s) = \sum_{(i,j)\in T^*} \mathsf{P}(\overline{A}_i \cap \overline{A}_j), \quad s = 1, \dots, S.$$

Using these equalities one easily can see that the following random values have expected values equal to the differences between the multivariate probability distribution function and its bounds:

$$\nu_{L_1}^s = \binom{\kappa^s}{2} - \binom{\kappa^s}{3} + \cdots + (-1)^n \binom{\kappa^s}{n},$$
$$\nu_{L_2}^s = \left(1 - \frac{2}{n}\right)\binom{\kappa^s}{2} - \binom{\kappa^s}{3} + \cdots + (-1)^n \binom{\kappa^s}{n},$$
$$\nu_{L_3}^s = -\lambda^s + \binom{\kappa^s}{2} - \binom{\kappa^s}{3} + \cdots + (-1)^n \binom{\kappa^s}{n},$$
$$\nu_{U_1}^s = -\binom{\kappa^s}{3} + \cdots + (-1)^n \binom{\kappa^s}{n},$$
$$\nu_{U_2}^s = \left(\frac{2}{k^*+1} - 1\right)\binom{\kappa^s}{1} + \left(1 - \frac{2}{k^*(k^*+1)}\right)\binom{\kappa^s}{2} - \binom{\kappa^s}{3} + \cdots + (-1)^n \binom{\kappa^s}{n}.$$

By the binomial theorem one has

$$\binom{\kappa^s}{0} - \binom{\kappa^s}{1} + \cdots + (-1)^n \binom{\kappa^s}{n} = 0$$

and the above random values can be expressed as

$$\nu_{L_1}^s = \begin{cases} \kappa^s - 1 & \text{if } \kappa^s \geq 2, \\ 0 & \text{otherwise,} \end{cases}$$

$$\nu_{L_2}^s = \begin{cases} \frac{1}{n}(\kappa^s - 1)(n - \kappa^s) & \text{if } \kappa^s \geq 2, \\ 0 & \text{otherwise,} \end{cases}$$

$$\nu_{L_3}^s = \begin{cases} \kappa^s - 1 - \lambda^s & \text{if } \kappa^s \geq 2, \\ 0 & \text{otherwise,} \end{cases}$$

$$\nu_{U_1}^s = \begin{cases} \frac{1}{2}(\kappa^s - 1)(2 - \kappa^s) & \text{if } \kappa^s \geq 3, \\ 0 & \text{otherwise,} \end{cases}$$

$$\nu_{U_2}^s = \begin{cases} \frac{(k^* - \kappa^s)(\kappa^s - k^* - 1)}{k^*(k^* + 1)} & \text{if } \kappa^s \geq 1, \\ 0 & \text{otherwise.} \end{cases}$$

Taking the new random values ν_{L_1}, ν_{L_2}, ν_{L_3}, ν_{U_1}, ν_{U_2} and the estimate ν_0 introduced before:

$$\nu_0 = \frac{1}{S}(\nu_0^1 + \cdots + \nu_0^S),$$

$$\nu_{L_1} = L_1 + \frac{1}{S}(\nu_{L_1}^1 + \cdots + \nu_{L_1}^S),$$

$$\nu_{L_2} = L_2 + \frac{1}{S}(\nu_{L_2}^1 + \cdots + \nu_{L_2}^S),$$

$$\nu_{L_3} = L_3 + \frac{1}{S}(\nu_{L_3}^1 + \cdots + \nu_{L_3}^S),$$

$$\nu_{U_1} = U_1 + \frac{1}{S}(\nu_{U_1}^1 + \cdots + \nu_{U_1}^S),$$

$$\nu_{U_2} = U_2 + \frac{1}{S}(\nu_{U_2}^1 + \cdots + \nu_{U_2}^S),$$

one gets altogether six estimates of the multivariate probability distribution function. These estimates obviously are not stochastically independent so one can mix them to get a new estimate with minimal possible variance. This technique is called *regression method* and it means forming the estimate

$$\nu = w_0\nu_0 + w_{L_1}\nu_{L_1} + w_{L_2}\nu_{L_2} + w_{L_3}\nu_{L_3} + w_{U_1}\nu_{U_1} + w_{U_2}\nu_{U_2}$$

with $w_0 + w_{L_1} + w_{L_2} + w_{L_3} + w_{U_1} + w_{U_2} = 1$, where w_0, w_{L_1}, w_{L_2}, w_{L_3}, w_{U_1}, w_{U_2} are chosen so that the variance of ν be minimized. Let

$$\begin{pmatrix} c_{00} & c_{0L_1} & c_{0L_2} & c_{0L_3} & c_{0U_1} & c_{0U_2} \\ c_{L_10} & c_{L_1L_1} & c_{L_1L_2} & c_{L_1L_3} & c_{L_1U_1} & c_{L_1U_2} \\ c_{L_20} & c_{L_2L_1} & c_{L_2L_2} & c_{L_2L_3} & c_{L_2U_1} & c_{L_2U_2} \\ c_{L_30} & c_{L_3L_1} & c_{L_3L_2} & c_{L_3L_3} & c_{L_3U_1} & c_{L_3U_2} \\ c_{U_10} & c_{U_1L_1} & c_{U_1L_2} & c_{U_1L_3} & c_{U_1U_1} & c_{U_1U_2} \\ c_{U_20} & c_{U_2L_1} & c_{U_2L_2} & c_{U_2L_3} & c_{U_2U_1} & c_{U_2U_2} \end{pmatrix}$$

be the covariance matrix C of the six estimates, where C is a symmetrical matrix. Then the variance of ν is $w^\mathsf{T} Cw$, where $w = (w_0, w_{L_1}, w_{L_2}, w_{L_3}, w_{U_1}, w_{U_2})^\mathsf{T}$. The Lagrangian problem:

$$\begin{cases} \min & w^\top Cw \\ \text{s.t.} & w_0 + w_{L_1} + w_{L_2} + w_{L_3} + w_{U_1} + w_{U_2} = 1 \end{cases}$$

can easily be solved. In fact, the gradient of $w^\mathsf{T}\, Cw$ equals $2\, w^\mathsf{T}\, C$, hence one has to solve the system of linear equations

$$\begin{aligned} &c_{00}w_0 + c_{0L_1}w_{L_1} + c_{0L_2}w_{L_2} + c_{0L_3}w_{L_3} \\ &+ c_{0U_1}w_{U_1} + c_{0U_2}w_{U_2} - \lambda = 0, \\ &c_{L_10}w_0 + c_{L_1L_1}w_{L_1} + c_{L_1L_2}w_{L_2} + c_{L_1L_3}w_{L_3} \\ &+ c_{L_1U_1}w_{U_1} + c_{L_1U_2}w_{U_2} - \lambda = 0, \\ &c_{L_20}w_0 + c_{L_2L_1}w_{L_1} + c_{L_2L_2}w_{L_2} + c_{L_2L_3}w_{L_3} \\ &+ c_{L_2U_1}w_{U_1} + c_{L_2U_2}w_{U_2} - \lambda = 0, \\ &c_{L_30}w_0 + c_{L_3L_1}w_{L_1} + c_{L_3L_2}w_{L_2} + c_{L_3L_3}w_{L_3} \\ &+ c_{L_3U_1}w_{U_1} + c_{L_3U_2}w_{U_2} - \lambda = 0, \\ &c_{U_10}w_0 + c_{U_1L_1}w_{L_1} + c_{U_1L_2}w_{L_2} + c_{U_1L_3}w_{L_3} \\ &+ c_{U_1U_1}w_{U_1} + c_{U_1U_2}w_{U_2} - \lambda = 0, \\ &c_{U_20}w_0 + c_{U_2L_1}w_{L_1} + c_{U_2L_2}w_{L_2} + c_{U_2L_3}w_{L_3} \\ &+ c_{U_2U_1}w_{U_1} + c_{U_2U_2}w_{U_2} - \lambda = 0, \\ &w_0 + w_{L_1} + w_{L_2} + w_{L_3} + w_{U_1} + w_{U_2} - \lambda = 1. \end{aligned}$$

for the unknowns w_0, w_{L_1}, w_{L_2}, w_{L_3}, w_{U_1}, w_{U_2}, λ. As the covariance matrix C is not known in advance, so one must estimate its elements from the random sample during the Monte-Carlo simulation procedure. This means that one must sum up not only the individual random values ν_0^s, $\nu_{L_1}^s$, $\nu_{L_2}^s$, $\nu_{L_3}^s$, $\nu_{U_1}^s$, $\nu_{U_2}^s$ but their crossproducts, too. The crossproducts are many times trivial, so their calculation is not necessary. For example ν_0^s equals $\nu_0^s\nu_0^s$, further when ν_0^s equals nonzero ($\kappa^s = 0$) then all other random values $\nu_{L_1}^s$, $\nu_{L_2}^s$, $\nu_{L_3}^s$, $\nu_{U_1}^s$, $\nu_{U_2}^s$ are equal zero, so the corresponding crossproducts are all zero. One should also notice that the random values $\nu_{L_1}^s$, $\nu_{L_2}^s$, $\nu_{L_3}^s$ are always nonnegative while the random values $\nu_{U_1}^s$, $\nu_{U_2}^s$ are always nonpositive. So the corresponding crossproducts cannot be positive even they are many times negative yielding real variance reduction in the final estimate.

One- and Two-Dimensional Marginal Distribution Functions

For the applicability of the Monte-Carlo simulation algorithm of the previous section one has to show that the one- and two-dimensional marginal distribution function values can be evaluated efficiently. As in the cases of the *multivariate normal distribution*, (one parameter) gamma and Dirichlet distributions the marginal distributions are also normal, gamma and Dirichlet and the one-dimensional Dirichlet distribution is the beta distribution, the one-dimensional marginal probability distribution functions can be evaluated by known algorithms. For example in the *IMSL subroutine library* [6] the subroutines MDNOR, MDGAM and MDBETA provide these calculations. In the case of the normal distribution the two-dimensional marginal probability distribution function also can be evaluated by a standard IMSL subroutine called MDBNOR. Some details of the calculations provided by these subroutines can be found in [8].

In the case of the *multivariate gamma distribution*, introduced by Prékopa and T. Szántai in [12], only the evaluation of the joint probability distribution function of the random variables

$$\xi_1 = \eta_1 + \eta_2,$$
$$\xi_2 = \eta_1 + \eta_3$$

is necessary. Here the random variables η_1, η_2 and η_3 are independent and gamma distributed with parameters ϑ_1, ϑ_2 and ϑ_3. Taking the joint characteristic function of ξ_1 and ξ_2 and applying the inversion formula one easily gets the joint probability density function of them. This is in the form of series expansion involving Laguerre polynomials. Using some integral formulae of these orthogonal polynomials one can integrate the joint probability density function to get the final formula for the evaluation of the joint probability distribution function in the following form

$$\begin{aligned} F(z_1, z_2) = {} & F_{\vartheta_1+\vartheta_2}(z_1) F_{\vartheta_1+\vartheta_3}(z_2) \\ & + \sum_{k=1}^{\infty} C(\vartheta_1, \vartheta_2, \vartheta_3, k) \\ & \times f_{\vartheta_1+\vartheta_2+1}(z_1) L_{k-1}^{\vartheta_1+\vartheta_2}(z_1) \\ & \times f_{\vartheta_1+\vartheta_3+1}(z_2) L_{k-1}^{\vartheta_1+\vartheta_3}(z_2), \end{aligned}$$

where

$$\begin{aligned} C(\vartheta_1, \vartheta_2, \vartheta_3, k) = {} & \frac{(k-1)!}{k} \frac{\Gamma(\vartheta_1 + k)}{\Gamma(\vartheta_1)} \\ & \times \frac{\Gamma(\vartheta_1 + \vartheta_2 + 1)}{\Gamma(\vartheta_1 + \vartheta_2 + k)} \frac{\Gamma(\vartheta_1 + \vartheta_3 + 1)}{\Gamma(\vartheta_1 + \vartheta_3 + k)} \end{aligned}$$

and $f_\vartheta(z)$ and $F_\vartheta(z)$ are the one-dimensional gamma probability density, respectively distribution, functions. For the calculation of the Laguerre polynomial the following recursion formula can be used

$$\begin{aligned} & (k+1) L_{k+1}^{\vartheta}(z) \\ & = (2k + \vartheta + 1 - z) L_k^{\vartheta}(z) - (k + \vartheta) L_{k-1}^{\vartheta}(z), \\ & k = 0, 1, \ldots, \end{aligned}$$

where $L_0^\vartheta(z) = 1$ and $L_1^\vartheta(z) = \vartheta + 1 - z$. The convergence of the series for calculation of $F(z_1, z_2)$ has been established by Szántai in [14].

In the case of *Dirichlet distribution* the two-dimensional marginal probability density function of the components ξ_i, ξ_j is given by

$$f(z_1, z_2) = \frac{\Gamma(a)\Gamma(b)\Gamma(c)}{\Gamma(a+b+c)} \cdot z_1^{a-1} z_2^{b-1} (1 - z_1 - z_2)^{c-1},$$
$$\text{if } z_1 + z_2 \le 1,\ z_1 \ge 0,\ z_2 \ge 0,$$

where $a = \vartheta_i$, $b = \vartheta_j$ and $c = \sum_{k=1}^{n+1} \vartheta_k - \vartheta_i - \vartheta_j$. One obtains by direct calculation for the two-dimensional probability distribution function

$$\begin{aligned} F(z_1, z_2) = {} & \frac{\Gamma(a+b+c)}{\Gamma(a)\Gamma(b)\Gamma(c)} \\ & \times \int_0^{z_1} \int_0^{z_2} t_1^{a-1} t_2^{b-1} (1 - t_1 - t_2)^{c-1} \, dt_2 \, dt_1 \\ = {} & \frac{z_1^a}{a} \frac{z_2^b}{b} + \sum_{m=1}^{\infty} (1-c) \cdots (m-c) \\ & \times \left[\frac{z_1^a}{a} \frac{z_2^{b+m}}{(b+m)m!} \right. \\ & \left. + \sum_{k=0}^{m} \frac{z_1^{a+k}}{(a+k)k!} \frac{z_2^{b+m-k}}{(b+m-k)(m-k)!} \right]. \end{aligned}$$

The above formula is valid only if $z_1 + z_2 \le 1$, $z_1 \ge 0$, $z_2 \ge 0$; otherwise the statement a) of the following more general theorem can be applied.

Theorem 1 *Let $z_1^* \le \cdots \le z_n^*$ be the ordered sequence of $z_1, \ldots, z_n$, the arguments of the n-dimensional Dirichlet distribution function.*

a) If $z_1^* + z_2^* > 1$ then one has

$$F(z_1, \ldots, z_n) = 1 - n + \sum_{i=1}^{n} F_i(z_i).$$

b) If $z_1^* + z_2^* + z_3^* > 1$ then one has

$$F(z_1, \ldots, z_n) = \frac{n-1}{2} - (n-2) \sum_{i=1}^{n} F_i(z_i) + \sum_{1 \le i < j \le n} F_{ij}(z_i, z_j).$$

Here $F_i(z_i)$ and $F_{ij}(z_i, z_j)$ are the one- and two-dimensional marginal probability distribution functions.

This theorem was formulated and proved by Szántai in [13]. It also can be found in [11].

Examples

For illustrating the lower and upper bounds on the multivariate normal probability distribution function value and the efficiency of the *variance reduction technique* described before one can regard the following examples.

Example 2

$$\begin{aligned}
&n = 10,\\
&x_1 = 1.7, \quad x_2 = 0.8, \quad x_3 = 5.1,\\
&x_4 = 3.2, \quad x_5 = 2.4, \quad x_6 = 1.8,\\
&x_7 = 2.7, \quad x_8 = 1.5, \quad x_9 = 1.2,\\
&x_{10} = 2.6,\\
&r_{ij} = 0.0, \quad i = 2, \ldots, 10, \quad j = 1, \ldots, i-1,
\end{aligned}$$

except $r_{21} = -0.6$, $r_{43} = 0.9$, $r_{65} = 0.4$, $r_{87} = 0.2$, $r_{10,9} = -0.8$.

Number of trials: 10000.

Lower bound by S1, S2	0.524736
Lower bound by Hunter	0.563719
Upper bound by S1, S2	0.588646
Estimated value	0.582743
Standard deviation	0.000608
Time in seconds (PC-586)	0.77
Efficiency	65.73

Example 3

$$\begin{aligned}
&n = 15,\\
&x_1 = 2.9, \quad x_2 = 2.9, \quad x_3 = 2.9,\\
&x_4 = 2.9, \quad x_5 = 2.9, \quad x_6 = 2.9,\\
&x_7 = 2.9, \quad x_8 = 2.9, \quad x_9 = 2.9,\\
&x_{10} = 2.9, \quad x_{11} = 2.9, \quad x_{12} = 2.7\\
&x_{13} = 1.6, \quad x_{14} = 1.2, \quad x_{15} = 2.1,\\
&r_{ij} = 0.2, \quad i = 2, \ldots, 10, \quad j = 1, \ldots, i-1,\\
&r_{ij} = 0.0, \quad i = 11, \ldots, 15, \quad j = 1, \ldots, i-1
\end{aligned}$$

except $r_{13,12} = 0.3$, $r_{15,14} = -0.95$.

Number of trials = 10000.

Lower bound by S1, S2	0.790073
Lower bound by Hunter	0.798730
Upper bound by S1, S2	0.801745
Estimated value	0.801304
Standard deviation	0.000193
Time in seconds (PC-586)	1.38
Efficiency	417.84

Both of the above examples are taken from [2, Exam. 4; 6] and they are according to standard multivariate normal probability distributions, i. e. all components of the normally distributed random vector have expected value zero and variance one. The efficiency of the Monte-Carlo simulation algorithm was calculated according to the crude Monte-Carlo algorithm in the usual way, i. e. it equals to the fraction $(t_0\sigma_0^2)/(t_1\sigma_1^2)$ where t_0, t_1 are the calculation times and σ_0^2, σ_1^2 are the variances of the crude and the compared simulation algorithms.

Remarks

In many applications one may need finding the *gradient of multivariate distribution functions*, too. As one has the general formula

$$\frac{\partial F(z_1, \ldots, z_n)}{\partial z_i} = F(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n | z_i) \cdot f_i(z_i),$$

where $F(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n \mid z_i)$ is the conditional probability distribution function of the random variables $\xi_1, \ldots, \xi_{i-1}, \xi_{i+1}, \ldots, \xi_n$, given that $\xi_i = z_i$, and

$f_i(z)$ is the probability density function of the random variable ξ_i, finding the gradient of a multivariate probability distribution function can be reduced to finding conditional distribution functions. In the cases of multivariate normal and Dirichlet distributions the conditional distributions are also multivariate normal and Dirichlet, and in the case of multivariate gamma distribution they are different and more complicated as it was obtained by Prékopa and Szántai [12].

In the case of multivariate normal probability distribution I. Deák [2] proposed another simulation technique which proved to be as efficient as the method described here. The main advantage of Deák's method is that it easily can be generalized for calculation the probability content of more general sets in the multidimensional space, like convex polyhedrons, hyperellipsoids, circular cones, etc. Its main drawback is that it works only for the multivariate normal probability distribution. The methods of Szántai and Deák have been combined by H. Gassmann to compute the probability of an n-dimensional rectangle in the case of multivariate normal distribution (see [3]). Also in the case of multivariate normal probability distribution A. Genz proposed the transformation of the original integration region to the unit hypercube $[0, 1]^n$ and then the application of a crude Monte-Carlo method or some lattice rules for the numerical integration of the resulting multidimensional integral. A comparison of methods for the computation of multivariate normal probabilities can be found in [4]. When the three-dimensional marginal probability distribution function values are also calculated by numerical integration there exist some new, sharper bounds. See [16] for these bounds and their effect on the efficiency of the Monte-Carlo simulation algorithm.

Approximation of multivariate probability integrals has a central role in probabilistic constrained stochastic programming when the probabilistic constraints are joint. The *computer code PCSP* (probabilistic constrained stochastic programming) originally was developed for handling the multivariate normal probability distributions in this framework (see [15]). A new version of the code now can handle multivariate gamma and Dirichlet distributions as well. The calculation procedures of this paper also has been applied by J. Mayer in his code solving this type of stochastic programming problems by reduced gradient algorithm (see [10]). These codes have been integrated by P. Kall and Mayer into a more advanced computer system for modeling in *stochastic linear programming* (see [7]).

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds

- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Bonferroni CE (1937) Teoria statistica delle classi e calcolo delle probabilita. Volume in onordi Riccardo Dalla Volta: 1–62
2. Deák I (1980) Three digit accurate multiple normal probabilities. Numerische Math 35:369–380
3. Gassmann H (1988) Conditional probability and conditional expectation of a random vector. In: Ermoliev Y, Wets RJ-B (eds) Numerical Techniques for Stochastic Optimization. Springer, Berlin, pp 237–254
4. Genz A (1993) Comparison of methods for the computation of multivariate normal probabilities. Computing Sci and Statist 25:400–405
5. Hunter D (1976) Bounds for the probability of a union. J Appl Probab 13:597–603
6. IMSL (1977) Library 1 reference manual. Internat. Math. Statist. Library
7. Kall P, Mayer J (1995) Computer support for modeling in stochastic linear programming. In: Marti K, Kall P (eds) Stochastic Programming: Numerical Methods and Techn. Applications. Springer, Berlin, pp 54–70
8. Kennedy WJ Jr, Gentle JE (1980) Statistical computing. M. Dekker, New York
9. Kruskal JB (1956) On the shortest spanning subtree of a graph and the travelling salesman problem. Proc Amer Math Soc 7:48–50
10. Mayer J (1988) Probabilistic constrained programming: A reduced gradient algorithm implemented on PC. Working Papers IIASA WP-88-39
11. Prékopa A (1995) Stochastic programming. Akad. Kiadó and Kluwer, Budapest–Dordrecht
12. Prékopa A, Szántai T (1978) A new multivariate gamma distribution and its fitting to empirical streamflow data. Water Resources Res 14:19–24
13. Szántai T (1985) Numerical evaluation of probabilities concerning multivariate probability distributions. Thesis Candidate Degree Hungarian Acad Sci (in Hungarian)
14. Szántai T (1986) Evaluation of a special multivariate gamma distribution function. Math Program Stud 27:1–16
15. Szántai T (1988) A computer code for solution of probabilistic-constrained stochastic programming problems. In: Ermoliev Y, Wets RJ-B (eds) Numerical Techniques for Stochastic Optimization. Springer, Berlin, pp 229–235
16. Szántai T: Improved bounds and simulation procedures on the value of multivariate normal probability distribution functions. Ann Oper Res (to appear) Special Issue: Research in Stochastic Programming (Selected refereed papers from the VII Internat. Conf. Stochastic Programming, Aug. 10–14, Univ. British Columbia, Vancouver, Canada).
17. Takács L (1955) On the general probability theorem. Comm Dept Math Physics Hungarian Acad Sci 5:467–476 (In Hungarian.)
18. Worsley KJ (1982) An improved Bonferroni inequality and applications. Biometrika 69:297–302

Approximations to Robust Conic Optimization Problems

MELVYN SIM
NUS Business School, National University of Singapore, Singapore, Republic of Singapore

Article Outline

Introduction
Formulation
  Affine Data Dependency
  Tractable Approximations of a Conic Chance Constrained Problem
References

Introduction

We consider a general conic optimization problem under parameter uncertainty is as follows:

$$\begin{aligned} \max \quad & \boldsymbol{c}'\boldsymbol{x} \\ \text{s.t.} \quad & \sum_{j=1}^{n} \tilde{\boldsymbol{A}}_j x_j - \tilde{\boldsymbol{B}} \in \mathcal{K} \\ & \boldsymbol{x} \in X\,, \end{aligned} \tag{1}$$

where the cone $\mathcal{K}$ is a regular cone, i. e., a closed, convex and pointed cone. The space of the data $(\tilde{\boldsymbol{A}}_1, \ldots, \tilde{\boldsymbol{A}}_n, \tilde{\boldsymbol{B}})$ depends on the cone, $\mathcal{K}$. The most common cone is the cone of non-negative orthant, $\Re^m_+$ in which the conic constraint in Problem (1) becomes a set of m linear constraints. Two important cones, which have many applications, include the second-order cone,

$$\mathcal{L}^{m+1} = \{(y_0, \boldsymbol{y}) \colon \|\boldsymbol{y}\|_2 \le y_0,\ \boldsymbol{y} \in \Re^m\}$$

and the cone of symmetric positive semidefinite matrix,

$$S^m = \{\boldsymbol{Y}\colon \boldsymbol{Y} \text{ is a symmetric postive semidefinite matrix}\}\,.$$

The interested reader may refer to the references of Ben-Tal and Nemirovski [3] and Pardalos and Wolkowicz [13].

In the uncertain conic optimization problem (1), the data $(\tilde{\boldsymbol{A}}_1, \ldots, \tilde{\boldsymbol{A}}_n, \tilde{\boldsymbol{B}})$ are uncertain. It is therefore conceivable that as the data take values different than the nominal ones, the conic constraint may be violated, and the optimal solution found using the nominal data may no longer be feasible at the conic constraint. To control the feasibility level of the conic constraint, one may consider a conic chance constrained model as follows:

$$\begin{aligned} \max \quad & \boldsymbol{c}'\boldsymbol{x} \\ \text{s.t.} \quad & \mathrm{P}\Big(\sum_{j=1}^{n} \tilde{\boldsymbol{A}}_j x_j - \tilde{\boldsymbol{B}} \in \mathcal{K}\Big) \geq 1-\epsilon \\ & \boldsymbol{x} \in X\,, \end{aligned} \tag{2}$$

in which the level of constraint violation is controlled probabilistically. Unfortunately, the chance constrained conic optimization problem (2) destroys the convexity of the problem and hence its computational tractability.

Formulation

In modern robust optimization, we represent data uncertainty using uncertainty sets instead of probability distributions. We allow the data $(\tilde{\boldsymbol{A}}_1, \ldots, \tilde{\boldsymbol{A}}_n, \tilde{\boldsymbol{B}})$ to vary within an uncertainty set $\mathcal{U}$ without having to violate the conic constraint. We call the following problem the *robust counterpart* of Problem (1)

$$\begin{aligned} \max \quad & \boldsymbol{c}'\boldsymbol{x} \\ \text{s.t.} \quad & \sum_{j=1}^{n} \boldsymbol{A}_j x_j - \boldsymbol{B} \in \mathcal{K} \\ & \quad \forall (\boldsymbol{A}_1, \ldots, \boldsymbol{A}_n, \boldsymbol{B}) \in \mathcal{U} \\ & \boldsymbol{x} \in X\,. \end{aligned} \tag{3}$$

The robust counterpart is introduced by Ben-Tal and Nemirovski [1] and independently by El-Ghoui et al. [9]. An immediate consequence of the robust counterpart is the preservation of the convexity. Unfortunately, due to the possibly infinite number of scenarios corresponding to the extreme points of the uncertainty set $\mathcal{U}$, optimizing the robust counterpart for general conic optimization problems is intractable.

It is noteworthy that in robust optimization, the ellipsoidal uncertainty set is a popular choice because of the motivation from the laws of large numbers and normal distributions. Under the assumption of normality, we could design an ellipsoidal set that is large enough so that the robust model will remain feasible with high probability. However, it turns out this approach can grossly over estimate the size of ellipsoid necessary to ensure the same level of robustness. To illustrate this issue, consider a linear constraint $\tilde{\boldsymbol{a}}'\boldsymbol{x} \geq b$ such that $\tilde{\boldsymbol{a}}$ is a multivariate normal with mean $\bar{\boldsymbol{a}}$ and covariance $\sum$. It is natural to design an ellipsoidal uncertainty set of the form $\mathcal{U} = \{\boldsymbol{a}\colon (\boldsymbol{a}-\bar{\boldsymbol{a}})\Sigma^{-1}(\boldsymbol{a}-\bar{\boldsymbol{a}}) \leq \alpha^2\}$ so that the problem remains feasible if $\tilde{\boldsymbol{a}} \in \mathcal{U}$, which has a probability of $\chi_n^2(\alpha^2)$. However, when solving the equivalent robust counterpart, $\bar{\boldsymbol{a}}'\boldsymbol{x} - \alpha\sqrt{\boldsymbol{x}'\Sigma\boldsymbol{x}} \geq b$, the robust solution has a feasibility probability of at least $\Phi(\alpha)$, where $\Phi(\alpha)$ is the standard normal function. Clearly, the value $\chi_n^2(\alpha^2)$ would be a gross over estimate of the robustness of the uncertain linear constraint compared to the value $\Phi(\alpha)$. The reason for this disparity is the fact that the uncertainty set chosen does not take into account the structure of cone.

We focus on the robust optimization framework proposed by Bertsimas and Sim [5], which offers a simple and tractable approximation of uncertain conic optimization problems. Moreover, under reasonable probabilistic assumptions on data variation, the framework approximates the conic chance constraint problem (2) by relating its feasibility probability with the size of the uncertainty set and the structure of the cone. Note that more refined approximations of chance constrained problem are available for the case of linear cones, $\mathcal{K} = \Re_+^m$. Interested readers can refer to Ben-Tal and Nemirovski [2], Bertsimas and Sim [4], Chen, Sim and Sun [8], Chen and Sim [6], Chen et al.[7], Lin et al. [10] and Janak et al. [11].

Affine Data Dependency

We first assume that uncertain data $(\tilde{\boldsymbol{A}}_1, \ldots, \tilde{\boldsymbol{A}}_n, \tilde{\boldsymbol{B}})$ are affinely dependent on some primitive uncertainty

vector, $\tilde{\boldsymbol{z}} \in \Re^N$, as follows

$$\tilde{\boldsymbol{A}}_i = \boldsymbol{A}_i(\tilde{\boldsymbol{z}}) \triangleq \boldsymbol{A}_i^0 + \sum_{j=1}^{N} \boldsymbol{A}_i^j \tilde{z}_j \quad i = 1, \ldots, n$$
$$\tilde{\boldsymbol{B}} = \boldsymbol{B}(\tilde{\boldsymbol{z}}) \triangleq \boldsymbol{B}^0 + \sum_{j=1}^{N} \boldsymbol{B}^j \tilde{z}_j \,.$$

Note that we can always define a bijection mapping from a vector space of $\tilde{\boldsymbol{z}}$ to the data space of $(\tilde{\boldsymbol{A}}_1, \ldots, \tilde{\boldsymbol{A}}_n, \tilde{\boldsymbol{B}})$. Therefore, under the affine data dependency, it is always possible to map all the data uncertainties affecting the conic constraint to the primitive uncertainty vector, $\tilde{\boldsymbol{z}}$. It is more convenient to define the following linear function mapping with respect to $(z_0, \boldsymbol{z})$,

$$\boldsymbol{Y}((z_0, \boldsymbol{z})) = \sum_{j=0}^{N} \boldsymbol{Y}_j z_j \,,$$

in which the variables $\boldsymbol{x}$ are affinely mapped to the variables $(\boldsymbol{Y}_0, \ldots, \boldsymbol{Y}_N)$ as follows

$$\boldsymbol{Y}_j = \sum_{i=1}^{n} \boldsymbol{A}_i^j x_i - \boldsymbol{B}^j \quad \forall j = 0, \ldots, N \,.$$

For instance, under such transformation, Problem (2) is equivalent to

$$\begin{array}{ll} \max & \boldsymbol{c}'\boldsymbol{x} \\ \text{s.t.} & \boldsymbol{Y}_j = \sum_{i=1}^{n} \boldsymbol{A}_i^j x_i - \boldsymbol{B}^j \qquad \forall j = 0, \ldots, N \\ & \mathrm{P}(\boldsymbol{Y}((1, \tilde{\boldsymbol{z}})) \in \mathcal{K}) \geq 1 - \epsilon \\ & \boldsymbol{x} \in X \,, \end{array} \tag{4}$$

and Problem (3) is the same as

$$\begin{array}{lll} \max & \boldsymbol{c}'\boldsymbol{x} & \\ \text{s.t.} & \boldsymbol{Y}_j = \sum_{i=1}^{n} \boldsymbol{A}_i^j x_i - \boldsymbol{B}^j & \forall j = 0, \ldots, N \\ & \boldsymbol{Y}((1, \tilde{\boldsymbol{z}})) \in \mathcal{K} & \forall \boldsymbol{z} \in \mathcal{V} \\ & \boldsymbol{x} \in X \,, & \end{array} \tag{5}$$

in which the uncertainty set $\mathcal{U}$ is mapped accordingly to the uncertainty set $\mathcal{V}$.

Example: Quadratic Chance Constraint Consider the following quadratic chance constraint,

$$\mathrm{P}(\|\boldsymbol{A}(\tilde{\boldsymbol{z}})\boldsymbol{x}\|_2^2 + \boldsymbol{b}(\tilde{\boldsymbol{z}})'\boldsymbol{x} + c(\tilde{\boldsymbol{z}}) \leq 0) \geq 1 - \epsilon \,,$$

where $\boldsymbol{x} \in \Re^n$ is the decision variable and $(\boldsymbol{A}(\tilde{\boldsymbol{z}}),\ \boldsymbol{b}(\tilde{\boldsymbol{z}}),\ c(\tilde{\boldsymbol{z}})) \in \Re^{m \times n} \times \Re^n \times \Re$ are the input data, which are affinely dependent on its primitive uncertainties as follows:

$$\begin{array}{rcl} \boldsymbol{A}(\tilde{\boldsymbol{z}}) & \triangleq & \boldsymbol{A}^0 + \sum_{j=1}^{N} \boldsymbol{A}^j \tilde{z}_j \\ \boldsymbol{b}(\tilde{\boldsymbol{z}}) & \triangleq & \boldsymbol{b}^0 + \sum_{j=1}^{N} \boldsymbol{b}^j \tilde{z}_j \\ c(\tilde{\boldsymbol{z}}) & \triangleq & c^0 + \sum_{j=1}^{N} c^j \tilde{z}_j \,. \end{array}$$

Note that a quadratic constraint

$$\|\boldsymbol{A}(\tilde{\boldsymbol{z}})\boldsymbol{x}\|_2^2 + \boldsymbol{b}(\tilde{\boldsymbol{z}})'\boldsymbol{x} + c(\tilde{\boldsymbol{z}}) \leq 0$$

is second-order cone representable as follows

$$\begin{bmatrix} \frac{1 - \boldsymbol{b}(\tilde{\boldsymbol{z}})'\boldsymbol{x} - c(\tilde{\boldsymbol{z}})}{2} \\ \boldsymbol{A}(\tilde{\boldsymbol{z}})\boldsymbol{x} \\ \frac{1 + \boldsymbol{b}(\tilde{\boldsymbol{z}})'\boldsymbol{x} + c(\tilde{\boldsymbol{z}})}{2} \end{bmatrix} \in \mathcal{L}^{m+2} \,.$$

Therefore, under the affine relation,

$$\boldsymbol{y}_0 = \begin{bmatrix} \boldsymbol{A}^0\boldsymbol{x} \\ \frac{1 + \boldsymbol{b}^{0\prime}\boldsymbol{x} + c^0}{2} \\ \frac{1 - \boldsymbol{b}^{0\prime}\boldsymbol{x} - c^0}{2} \end{bmatrix},$$

and

$$\boldsymbol{y}_j = \begin{bmatrix} \boldsymbol{A}^j\boldsymbol{x} \\ \frac{\boldsymbol{b}^{j\prime}\boldsymbol{x} + c^j}{2} \\ \frac{-\boldsymbol{b}^{j\prime}\boldsymbol{x} - c^j}{2} \end{bmatrix} \quad \forall j = 1, \ldots, N$$

we transform the quadratic chance constraint problem into the following conic chance constraint

$$\mathrm{P}\left(\boldsymbol{y}_0 + \sum_{j=1}^{n} \boldsymbol{y}_j \tilde{z}_j \in \mathcal{L}^{m+2}\right) .$$

Hence, we treat the quadratic constraint as a special case of second-order cone constraint.

Tractable Approximations of a Conic Chance Constrained Problem

We focus on deriving a tractable approximation on the following conic chance constraint:

$$\mathrm{P}(\boldsymbol{Y}((1, \tilde{\boldsymbol{z}})) \in \mathcal{K}) \geq 1 - \epsilon \,. \tag{6}$$

For notational convenience, we define

$$\boldsymbol{X} \triangleq (\boldsymbol{Y}_0, \ldots, \boldsymbol{Y}_N) \,.$$

For a given a reference vector (or matrix), $\boldsymbol{V} \in \text{int}(\mathcal{K})$, where int($\mathcal{K}$) denotes the interior of the cone $\mathcal{K}$, we can define the function

$$f(\boldsymbol{X},\,(z_0,\boldsymbol{z})) \triangleq \max\{\theta : \boldsymbol{Y}((z_0,\,\boldsymbol{z})) - \theta \boldsymbol{V} \in \mathcal{K}\}\,,$$

which has the following properties:

Proposition 1 *For any $\boldsymbol{V} \in$ int($\mathcal{K}$), the function $f(\boldsymbol{X},\,(z_0,\,\boldsymbol{z}))$ satisfies the properties:*

(a) $f(\boldsymbol{X},\,(z_0,\,\boldsymbol{z}))$ *is bounded and concave in* $\boldsymbol{X}$ *and* $(z_0,\,\boldsymbol{z})$.

(b) $f(\boldsymbol{X},\,k(z_0,\,\boldsymbol{z})) = kf(\boldsymbol{X},\,(z_0,\,\boldsymbol{z}))\,,\ \forall k \geq 0.$

(c) $f(\boldsymbol{X},\,(z_0,\,\boldsymbol{z})) \geq s$ *if and only if* $\boldsymbol{Y}((z_0,\,\boldsymbol{z})) - s\,\boldsymbol{V} \in \mathcal{K}$.

(d) $f(\boldsymbol{X},\,(z_0,\,\boldsymbol{z})) > s$ *if and only if* $\boldsymbol{Y}((z_0,\,\boldsymbol{z})) - s\,\boldsymbol{V} \in$ *int*($\mathcal{K}$).

Hence, the conic chance constraint of (6) is equivalent to the following chance constraint

$$\mathrm{P}(f(\boldsymbol{X},\,(1,\,\tilde{\boldsymbol{z}})) \geq 0) \geq 1 - \epsilon\,. \tag{7}$$

In order to build a tractable framework that approximates the conic chance constraint problem, we first analyze the robust counterpart approach to uncertainty. Given an ellipsoidal uncertainty set

$$\mathcal{E}(\rho) = \{\boldsymbol{z}\colon \|\boldsymbol{z}\|_2 \leq \rho\}\,,$$

the robust counterpart

$$f(\boldsymbol{X},(1,\boldsymbol{z})) \geq 0 \qquad \forall \boldsymbol{z} \in \mathcal{E}(\rho)\,, \tag{8}$$

despite its convexity, is generally intractable. Instead we consider the following robust counterpart:

$$\begin{aligned} f(\boldsymbol{X},\,(1,\,\boldsymbol{0})) + \sum_{j=1}^{N} \{f(\boldsymbol{X},\,(0,\,\boldsymbol{e}_j))v_j \\ + f(\boldsymbol{X},\,(0,\,-\boldsymbol{e}_j))w_j\} \geq 0, \\ \forall (\boldsymbol{v},\,\boldsymbol{w}) \in \mathcal{V}(\rho) \end{aligned} \tag{9}$$

where $\boldsymbol{e}_j \in \Re^N$ is a unit vector with one at the jth entry and the uncertainty set

$$\mathcal{V}(\rho) = \left\{(\boldsymbol{v},\,\boldsymbol{w}) \in \Re^N_+ \times \Re^N_+ \mid \|\boldsymbol{v} + \boldsymbol{w}\|_2 \leq \rho\right\}\,. \tag{10}$$

Proposition 2 *The robust counterpart (9) is tractable relaxation of the robust counterpart, (8).*

Theorem 1

(a) The constraint (9) is equivalent to

$$f(\boldsymbol{X},\,(1,\,\boldsymbol{0})) \geq \rho\|\boldsymbol{s}\|_2\,, \tag{11}$$

where

$$\begin{aligned} s_j = \max\{-f(\boldsymbol{X},\,(0,\,\boldsymbol{e}_j)),\,-f(\boldsymbol{X},\,(0,\,-\boldsymbol{e}_j))\}, \\ \forall j = 1,\,\ldots\,,\,N\,. \end{aligned}$$

(b) Eq. (11) can be written as:

$$\begin{aligned} & f(\boldsymbol{X},\,(1,\,\boldsymbol{0})) \geq \rho y \\ & f(\boldsymbol{X},\,(0,\,\boldsymbol{e}_j)) + t_j \geq 0, \qquad \forall j \in N \\ & f(\boldsymbol{X},\,(0,\,-\boldsymbol{e}_j)) + t_j \geq 0, \qquad \forall j \in N \\ & \|\boldsymbol{t}\|_2 \leq y \\ & \text{for some } y \in \Re,\ \boldsymbol{t} \in \Re^N\,. \end{aligned} \tag{12}$$

From Proposition 1 and noting that

$$\boldsymbol{Y}((1,\,\boldsymbol{0})) = \boldsymbol{Y}_0$$

and

$$\boldsymbol{Y}((0,\,\pm\boldsymbol{e}_j)) = \pm\boldsymbol{Y}_j\,,$$

we can also represent the formulation (12) explicitly in conic constraints as follows:

$$\begin{aligned} & \boldsymbol{Y}_0 - \rho y \boldsymbol{V} \in \mathcal{K} \\ & \boldsymbol{Y}_j + t_j \boldsymbol{V} \in \mathcal{K}, \qquad \forall j \in N \\ & -\boldsymbol{Y}_j + t_j \boldsymbol{V} \in \mathcal{K}, \qquad \forall j \in N \\ & \|\boldsymbol{t}\|_2 \leq y \\ & \text{for some } y \in \Re,\ \boldsymbol{t} \in \Re^N, \end{aligned} \tag{13}$$

for a given reference vector, $\boldsymbol{V}$ in the interior of the cone, $\mathcal{K}$. The formulation (12) becomes a cartesian product of $2N + 1$ cones of the nominal problem plus an additional second-order cone, which is a computationally tractable cone. Hence, in theory the formulation (12) is not much harder to solve compared with its nominal problem.

One natural question is whether the simple approximation is overly conservative with respect to Problem (8). While there is lack of theoretical evidence on the closeness of the approximation, the framework does lead to an approximation of the conic chance constraint problem. An important component of the analysis is the relation among different norms, which we will subsequently present.

Recall that a norm satisfies $\|\boldsymbol{A}\| \geq 0$, $\|k\boldsymbol{A}\| = |k| \cdot \|\boldsymbol{A}\|$, $\|\boldsymbol{A}+\boldsymbol{B}\| \leq \|\boldsymbol{A}\|+\|\boldsymbol{B}\|$, and $\|\boldsymbol{A}\| = 0$, implies that $\boldsymbol{A} = \boldsymbol{0}$. For a given regular cone, $\mathcal{K}$, and its interior, $\boldsymbol{V}$, we define the following cone induced norm

$$\|\boldsymbol{Y}\|_{\mathcal{K},\boldsymbol{V}} \triangleq \min\{y,\ y\boldsymbol{V}-\boldsymbol{Y} \in \mathcal{K},\ \boldsymbol{Y}+y\boldsymbol{V} \in \mathcal{K}\}\,. \quad (14)$$

Proposition 3

$$\max\{-f(\boldsymbol{X},\ (z_0, \boldsymbol{z})),\ -f(\boldsymbol{X},\ -(z_0,\ \boldsymbol{z}))\} = \|\boldsymbol{Y}((z_0,\ \boldsymbol{z}))\|_{\mathcal{K},\boldsymbol{V}}\,.$$

We consider the common cones and the respective norms.

(a) Second-order cone:
Let $\boldsymbol{e}_1 \in \text{int}(\mathcal{L}^{n+1})$ be the reference vector, we have for any vector $(y_0,\ \boldsymbol{y}) \in \Re^{n+1}$

$$\begin{aligned}&\|(y_0,\ \boldsymbol{y})\|_{\mathcal{L}^{n+1},\boldsymbol{e}_{n+1}}\\ &= \min\{\theta\colon \|\boldsymbol{y}\|_2 \leq \theta - y_0, \|\boldsymbol{y}\|_2 \leq \theta + y_0\}\\ &= \|\boldsymbol{y}\|_2 + |y_0|\end{aligned}$$

(b) Cone of symmetric positive definite matrix:
Let the identity matrix $\boldsymbol{I}$ be the reference matrix, then for any $m \times m$ symmetric matrix, $\boldsymbol{Y}$,

$$\begin{aligned}\|\boldsymbol{Y}\|_{S_+^m,\boldsymbol{I}} &= \min\{y,\ y\boldsymbol{I}-\boldsymbol{Y} \in S_+^m,\ \boldsymbol{Y}-y\boldsymbol{I} \in S_+^m\}\\ &= \|\boldsymbol{Y}\|_2\,.\end{aligned}$$

Proposition 4 *Suppose $\boldsymbol{X}$ is feasible in Problem (12) then*

$$\begin{aligned}&P(f(\boldsymbol{X},\ (1,\ \tilde{\boldsymbol{z}})) < 0)\\ &\leq\ P\left(\left\|\sum_{j=1}^N \boldsymbol{Y}_j\tilde{z}_j\right\|_{\mathcal{K},\boldsymbol{V}} > \rho\sqrt{\sum_{j\in N}\|\boldsymbol{Y}_j\|^2_{\mathcal{K},\boldsymbol{V}}}\right)\,.\end{aligned}$$

To obtain explicit bounds, we focus on primitive uncertainties, $\tilde{\boldsymbol{z}}$ that are normally and independently distributed with mean zero and variance one. For a sum of random scalers, we have

$$\mathrm{P}\left(\left|\sum_{j=1}^N y_j\tilde{z}_j\right| > \rho\sqrt{\sum_{j=1}^N y_j^2}\right) \leq 1 - 2\Phi(\rho)\,.$$

To derive a similar large deviation result for the sum of random vectors used in Proposition 4, we consider the following generalization:

$$\mathrm{P}\left(\left\|\sum_{j=1}^N \boldsymbol{Y}_j\tilde{z}_j\right\|_{\mathcal{K},\boldsymbol{V}} > \rho\sqrt{\sum_{j=1}^N\|\boldsymbol{Y}_j\|^2_{\mathcal{K},\boldsymbol{V}}}\right) \leq \phi(\rho)\,,$$

where $\phi(\rho)$ is a non-trivial probability bound that depends on the choice of cone, $\mathcal{K}$, and possibly the dimension and the reference vector, $\boldsymbol{V}$.

An important component of the analysis is the relation among different norms. We denote by $\langle\ ,\ \rangle$ the inner product on a vector space, $\Re^m$, or the space of m by m symmetric matrices. The inner product induces a norm $\|\boldsymbol{X}\| \triangleq \sqrt{\langle \boldsymbol{X}, \boldsymbol{X}\rangle}$. For a vector space, the natural inner product is the Euclidian inner product, $\langle \boldsymbol{x},\ \boldsymbol{y}\rangle = \boldsymbol{x}'\boldsymbol{y}$, and the induced norm is the Euclidian norm $\|\boldsymbol{x}\|_2$. For the space of symmetric matrices, the natural inner product is the trace product or $\langle \boldsymbol{X},\ \boldsymbol{Y}\rangle = \text{trace}(\boldsymbol{X}\boldsymbol{Y})$ and the corresponding induced norm is the Frobenius norm, $\|\boldsymbol{Y}\|_F$.

We analyze the relation of the inner product norm $\sqrt{\langle \boldsymbol{X}, \boldsymbol{X}\rangle}$ with the norm $\|\boldsymbol{X}\|_{\mathcal{K},\boldsymbol{V}}$ for the conic optimization problems we consider. Since $\|\boldsymbol{X}\|_{\mathcal{K},\boldsymbol{V}}$ and the inner product norm $\|\boldsymbol{X}\|$ are valid norms in a finite dimensional space, there exist finite $\alpha_1,\ \alpha_2 > 0$ such that

$$\frac{1}{\alpha_1}\|\boldsymbol{X}\|_{\mathcal{K},\boldsymbol{V}} \leq \|\boldsymbol{X}\| \leq \alpha_2\|\boldsymbol{X}\|_{\mathcal{K},\boldsymbol{V}}\,,$$

for all $\boldsymbol{X}$ in the relevant space. Hence, we define the parameter

$$\alpha_{\mathcal{K},\boldsymbol{V}} = \underbrace{\left(\max_{\|\boldsymbol{X}\|=1}\|\boldsymbol{X}\|_{\mathcal{K},\boldsymbol{V}}\right)}_{=\alpha_1}\underbrace{\left(\max_{\|\boldsymbol{X}\|_{\mathcal{K},\boldsymbol{V}}=1}\|\boldsymbol{X}\|\right)}_{=\alpha_2} \quad (15)$$

which measures the disparity between the norm $\|\cdot\|_{\mathcal{K},\boldsymbol{V}}$ and the inner product norm $\|\cdot\|$.

Parameter $\alpha_{\mathcal{K},\boldsymbol{V}}$ of Common Cones

(a) Second-order cone:
Let $\boldsymbol{e}_{n+1}$ be the reference vector, then

$$\|(\boldsymbol{y}, y_{n+1})\|_{\mathcal{L}^{n+1},\boldsymbol{v}} = \|\boldsymbol{y}\|_2 + |y_{n+1}|\,.$$

Therefore,

$$\begin{aligned}\frac{1}{\sqrt{2}}\|(\boldsymbol{y},\ y_{n+1})\|_{\mathcal{L}^{n+1},\boldsymbol{e}_{n+1}} &\leq \|(\boldsymbol{y},\ y_{n+1})\|_2\\ &\leq \|(\boldsymbol{y},\ y_{n+1})\|_{\mathcal{L}^{n+1},\boldsymbol{e}_{n+1}}\end{aligned}$$

Approximations to Robust Conic Optimization Problems, Table 1
Probability bounds of $P(f(X,(1,\tilde{z}))\prec 0)$ for $\tilde{z}\sim\mathcal{N}(0,I)$.

Type	Probability bound of infeasibility
$\mathcal{L}^{m+1}$	$\sqrt{\frac{e}{2}}\Omega\exp(-\frac{\Omega^2}{4})$
S_+^m	$\sqrt{\frac{e}{m}}\Omega\exp(-\frac{\Omega^2}{2m})$

and hence,

$$\alpha_{\mathcal{L}^{n+1},V}=\sqrt{2}\,.$$

(b) Cone of symmetric positive definite matrix:
Let $\boldsymbol{I}$ be the reference matrix, then for any $m\times m$ symmetric matrix $\boldsymbol{Y}$

$$\|\boldsymbol{Y}\|_{S_+^m,\boldsymbol{I}}=\|\boldsymbol{Y}\|_2\,.$$

Let $\lambda_j,\ j=1,\ldots,m$ be the eigenvalues of the matrix $\boldsymbol{Y}$. Since $\|\boldsymbol{Y}\|_F=\sqrt{\mathrm{trace}(\boldsymbol{Y}^2)}=\sqrt{\sum_j\lambda_j^2}$ and $\|\boldsymbol{Y}\|_2=\max_j|\lambda_j|$, we have

$$\|\boldsymbol{Y}\|_2\le\|\boldsymbol{A}\|_F\le\sqrt{m}\|\boldsymbol{Y}\|_2\,.$$

Hence,

$$\alpha_{S_+^m,\boldsymbol{I}}=\sqrt{m}\,.$$

Theorem 2 *Given an inner product norm $\|\cdot\|$ and under the assumption that $\tilde{z}_j$ are normally and independently distributed with mean zero and variance one, i. e., $\tilde{\boldsymbol{z}}\sim\mathcal{N}(0,\boldsymbol{I})$, then*

$$P\left(\left\|\sum_{j=1}^N\boldsymbol{Y}_j\tilde{z}_j\right\|_{\mathcal{K},V}>\rho\sqrt{\sum_{j\in N}\|\boldsymbol{Y}_j\|_{\mathcal{K},V}^2}\,,\right)\le\frac{\sqrt{e}\rho}{\alpha_{\mathcal{K},V}}\exp\left(-\frac{\rho^2}{2\alpha_{\mathcal{K},V}^2}\right),\quad(16)$$

for all $\rho>\alpha_{\mathcal{K},V}$.

In order to have the smallest budget of uncertainty, ρ, it is reasonable to select $\boldsymbol{V}$ that minimizes $\alpha_{\mathcal{K},V}$, i. e.,

$$\alpha_{\mathcal{K}}=\min_{V\in\mathrm{int}(\mathcal{K})}\alpha_{\mathcal{K},V}\,.$$

For general conic optimization, we have shown that the probability bound depends on the the choice of $\boldsymbol{V}\in\mathrm{int}(\mathcal{K})$. A cone, $\mathcal{K}\subseteq\Re^n$ is *homogenous* if for any pair of points $\boldsymbol{A},\boldsymbol{B}\in\mathrm{int}(\mathcal{K})$ there exists an invertible linear map $\boldsymbol{M}:\Re^n\to\Re^n$ such that $\boldsymbol{M}(\boldsymbol{A})=\boldsymbol{B}$ and $\boldsymbol{M}(\mathcal{K})=\mathcal{K}$. It turns out that for *homogenous cones*, of which semidefinite and second-order cones are special cases, the probability bound does not depend on $\boldsymbol{V}\in\mathrm{int}(\mathcal{K})$.

Theorem 3 *Suppose the cone $\mathcal{K}$ is homogenous. For any $\boldsymbol{V}\in int(\mathcal{K})$, the probability bound of conic infeasibility satisfies*

$$P(\boldsymbol{Y}((1,\boldsymbol{z}))\notin\mathcal{K})\le\frac{\sqrt{e}\rho}{\alpha_{\mathcal{K}}}\exp\left(-\frac{\rho^2}{2\alpha_{\mathcal{K}}^2}\right).$$

For the second-order cone, $\alpha_{\mathcal{L}^{n+1}}=\sqrt{2}$ and for the symmetric positive semidefinite cone, $\alpha_{S_+^m}=\sqrt{m}$.

While different $\boldsymbol{V}$ lead to the same probability bounds, some choices of $\boldsymbol{V}$ may lead to better objectives. The following theorem suggests an iterative improvement strategy.

Theorem 4 *For any $\boldsymbol{V}\in int(\mathcal{K})$, if $\boldsymbol{X}$, $\boldsymbol{t}$ and $y>0$ are feasible in (13), then they are also feasible in the same problem in which $\boldsymbol{V}$ is replaced by*

$$\boldsymbol{W}=\boldsymbol{Y}_0/(\rho y)\,.$$

While we focus on the primitive uncertainty vector $\tilde{\boldsymbol{z}}$ being normally distributed, using the large deviation bounds of Nemirovski [12], we can also apply the same framework to other distributions. The interested reader may refer to Bertsimas and Sim [5].

References

1. Ben-Tal A, Nemirovski A (1998) Robust convex optimization. Math Oper Res 23:769–805
2. Ben-Tal A, Nemirovski A (2000) Robust solutions of Linear Programming problems contaminated with uncertain data. Math Program 88:411–424
3. Ben-Tal A, Nemirovski A (2001) Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications. MPR-SIAM Series on Optimization. SIAM, Philadelphia
4. Bertsimas D, Sim M (2004) Price of robustness. Oper Res 52:35–53
5. Bertsimas D, Sim M (2006) Tractable Approximations to Robust Conic Optimization Problems. Math Program 107(1):5–36

6. Chen W, Sim M (2007) Goal Driven Optimization. Working Paper, NUS Business School
7. Chen W, Sim M, Sun J, Teo C-P (2007) From CVaR to Uncertainty Set: Implications in Joint Chance Constrained Optimization. Working Paper, NUS Business School
8. Chen X, Sim M, Sun P (2006) A robust optimization perspective on stochastic programming. to appear in: Oper Res 55(6)
9. El Ghaoui L, Oustry F, Lebret H (1998) Robust Solutions to Uncertain Semidefinite Programs. SIAM J Optim 9(1):33–52
10. Lin X, Janak SL, Floudas CA (2004) A New Robust Optimization Approach for Scheduling under Uncertainty: I Bounded Uncertainty. Comput Chem Eng 28:1069–1085
11. Janak SL, Lin X, Floudas CA (2007) A New Robust Optimization Approach for Scheduling under Uncertainty: II Uncertainty with Known Probability Distribution. Comput Chem Eng 31:171–195
12. Nemirovski A (2003) On tractable approximations of ramdomly perturbed convex constraints. Proceedings of the 42nd IEEE Conference on Decision and Control, Maui, Hawaii, USA, December, pp 2419–2422
13. Pardalos PM, Wolkowicz H (1998) Topics in Semidefinite and Interior-Point Methods. Fields Institute Communications Series vol 18, American Mathematical Society

Archimedes and the Foundations of Industrial Engineering

PETROS XANTHOPOULOS
Industrial and Systems Engineering, University of Florida, Gainesville, USA

MSC2000: 01A20

Article Outline

Biographical Sketch
Archimedes' Work
Conclusion
References

Biographical Sketch

Archimedes (287–212 B.C.) was a famous Greek mathematician, engineer and philosopher. Born in the city of Syracuse on the Island of Sicily to an astronomer and mathematician named Phidias, Archimedes spent the first years of his life in his home city and went to Alexandria in Egypt to study mathematics. He soon became friends with Konon of Samos and Eratosthenes. After spending a considerable amount of time in Alexandria, he returned to Syracuse, where he remained for the rest of his life conducting mathematical research. He had a good relationships with king Hieron of Syracuse and his son Gelon. We know that he assisted king Hieron numerous times either with his inventions during the Second Punic War or by solving problems like the well-known case (the one that Archimedes jumped out of his bathtub crying out eureka) with the crown of king Hieron during peacetime.

In this article we will concentrate on the work of Archimedes, which is closely related to what we call today industrial engineering (including the mathematical theory of optimization, operations research, theory of algorithms, etc.). In particular, we will present Archimedes' definition of convex sets, his method of exhaustion for computing finite integrals, his contribution to recursive algorithms, and his approach to solving real-life operations research problems during the Second Punic War.

Archimedes' Work

One very important concept for optimization is the definition of convex sets. The first such definition was given by Euclid in his books *Elements*, but Archimedes elaborated this definition and gave us his definition, which was used until the first decades of the 20th century. In his work *On the sphere and the cylinder* he gives the following definition of the convex arc:

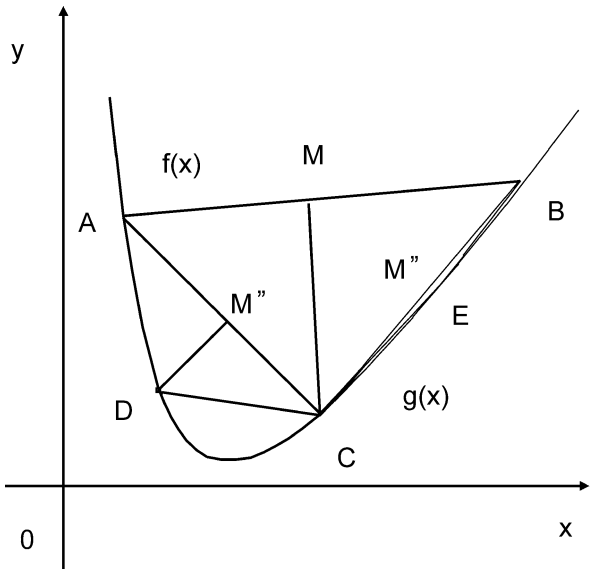
Definition 1 I call convex in one and the same directions the surfaces for which the straight line joining two arbitrary points lies on the same side of the surface.

On his work *On the equilibrium of planes* he gives a definition of the convex set using the center-of-gravity concept:

Definition 2 In any figure whose perimeter is convex the center of gravity must be within the figure.

It is worth mentioning that Archimedes' definitions of convex arcs and convex sets were those used until 1913, when E. Steinitz introduced the modern definitions of convexity.

Archimedes had invented a geometrical method called the method of exhaustion (or method of infinitesimals) in order to be able to compute areas under convex curves. This was one of the first geometrical methods devised to compute what we call today definite



Archimedes and the Foundations of Industrial Engineering, Figure 1
Illustration of Archimedes' exhaustion method

integrals. In modern notation Archimedes was able to compute

$$\int_a^b [f(x) - g(x)]dx, \tag{1}$$

where $f(x)$ is a line segment and $g(x)$ a convex function (usually parabola). An illustration of this method can be found in Fig. 1.

Suppose that we want to compute the area over a curve and below the line segment AB. Archimedes considered the triangle $\widehat{ABC}$, where C is the point below the midpoint M of the line segment AB(MC is the middle vertical of AB). If we iteratively repeat this process, we can see that the next two parabolic triangles have an area that is $\frac{1}{4}$ of the initial triangle. Therefore, the area of the curve was the infinite sum of $1 + \frac{1}{4} + \frac{1}{8} + \ldots$, where 1 corresponds to the area of the initial triangle $\widehat{ABC}$. In this way Archimedes was able to geometrically approximate the area of a convex parabolic curve.

According to [7] Archimedes was the first (in around 220 B.C.) to use a double recursive algorithm to solve the problem of the sand reckoner (Psammitis). In this book he tries to come up with of a number that is much larger than the number of grains of sand in the world and therefore prove that the number of grains of sand in the world is not infinite. For this he fixes a number α and defines the number $p_k(x)$ as follows (using a double recursion scheme):

$$\begin{aligned} p_0(x) &= 1, \\ p_{k+1}(0) &= p_k(\alpha), \\ p_{n+1}(x+1) &= a p_{k+1}(x)\ . \end{aligned} \tag{2}$$

Therefore, $p_k(x) = \alpha^{xk}$. Then he considers $p_\alpha(\alpha)$ for $\alpha = 10^8$, which was the largest number known at that time, and he comes up with the number $10^{10}17$, which was the largest number used in mathematics until 1933.

Apart from Archimedes' exceptional skills in theoretical research, he also became famous for his ability to deal with everyday life problems. Although operations research was developed during World War II, when mathematicians were looking for ways to make better decisions in utilizing certain materials subject to some constraint, some consider Archimedes the father of operations research as he helped his home city defend itself against the Romans during the Second Punic War.

Before King Hieron died, he asked Archimedes to organize the complete defense of Syracuse against Roman general Marcelus. Archimedes is said to have invented many mechanical war machines like the claw of Archimedes, a new version of catapult, an array of mirrors that was able to burn enemy ships, etc.

Archimedes was also responsible for organizing the defense of Syracuse and the redecoration of Fort Euryalus [6]. Due to Archimedes' clever defense plans, Syracuse managed to survive the Roman siege for 2 years.

Conclusion

Archimedes was a perfect example of a scientist who managed to combine theoretical research with practical problem solving. He managed to distinguish between the two by referring to his mechanical inventions as *parergon*. This shows that Archimedes was capable of performing both basic and applied research, but he regarded basic research as more important. In this sense he can be considered the father of the modern industrial engineer who utilizes theoretical methods to solve problems that arise in everyday life.

References

1. Brunschwig J, Lloyd GER (eds) (2000) Greek thought: a guide to classical knowledge. Belknap, Cambridge, Massachusetts
2. Dijksterhuis EJ (1987) Archimedes. Princeton University Press, Princeton, NJ
3. Gow M (2005) Archimedes: Mathematical Genius of the Ancient World. Great Minds of Science series. Enslow, Berkeley Heights, NJ
4. Heath TL (1897) The works of Archimedes. Cambridge University Press, Cambridge
5. Heath TL (ed) (2002) The Works of Archimedes. Dover, New York
6. Lawrence AW (1946) Archimedes and the Design of Euryalus Fort. J Hell Stud – JSTOR
7. Odifreddi P. Recursive Functions. In: Edward Zalta N (ed) The Stanford Encyclopedia of Philosophy (Summer 2005 edn). http://plato.stanford.edu/archives/sum2005/entries/recursive-functions. Accessed 21 Mar 2008
8. Simms DL (1995) Archimedes the Engineer. History of Technology, vol 17. Continuum International, London, pp 45–111
9. Stein S (1999) Archimedes: What Did He Do Besides Cry Eureka? Mathematical Association of America, Washington, DC

Asset Liability Management Decision Support System

KOSMIDOU KYRIAKI[1], ZOPOUNIDIS CONSTANTIN[2]

[1] Department of International European Economic Studies, Athens University Economics and Business, Athens, Greek

[2] Department of Production Engineering and Management, Technical University of Crete, Chania, Greece

MSC2000: 90C29, 65K99

Article Outline

Introduction
Background
Model
Conclusions
References

Introduction

Asset Liability Management (ALM) is an important dimension of risk management, where the exposure to various risks is minimized while maintaining the appropriate combination of asset and liability, in order to satisfy the goals of the firm or the financial institution (Kosmidou and Zopounidis [18]).

Up to the 1960's, liability management was aimless. In their majority, the banking institutions considered liabilities as exogenous factors contributing to the limitation of asset management. Indeed, for a long period the greater part of capital resources originated from savings deposits and deposits with agreed maturity.

Nevertheless, the financial system has radically changed. Competition among the banks for obtaining capital has become intense. Liability management is the main component of each bank strategy in order to ensure the cheapest possible financing. At the same time, the importance of decisions regarding the amount of capital adequacy is enforced. Indeed, the adequacy of the bank as far as equity, contributes to the elimination of bankruptcy risk, a situation in which the bank cannot satisfy its debts towards clients who make deposits or others who take out loans. Moreover, the capital adequacy of banks is influenced by the changes of stock prices in relation to the amount of the capital stock portfolio. Finally, the existence of a minimum amount of equity is an obligation of commercial banks to the Central Bank for supervisory reasons. It is worth mentioning that based on the last published data (31/12/2001) the Bank of Greece assigns the coefficient for the Tier 1 capital at 8%, while the corresponding European average is equal to 6%. This results in the configuration of the capital adequacy of the Greek banking system at higher levels than the European average rate. The high capital adequacy index denotes large margins of profitability amelioration, which reduces the risk of a systematic crisis.

Asset management in a contemporary bank cannot be distinct from liability management. The simultaneous management of assets and liabilities, in order to maximize the profits and minimize the risk, demands the analysis of a series of issues.

Firstly, there is the substantive issue of strategic planning and expansion. That is, the evaluation of the total size of deposits that the bank wishes to attract and the total number of loans that it wishes to provide.

Secondly, there is the issue of determination of the "best temporal structure" of the asset liability management, in order to maximize the profits and to ensure

the robustness of the bank. Deposits cannot all be liquidated in the same way. From the point of view of assets, the loans and various placements to securities constitute commitments of the bank's funds with a different duration time. The coordination of the temporal structure of the asset liability management is of major importance in order to avoid the problems of temporary liquidity reduction, which might be very injurious.

Thirdly, there is the issue of risk management of assets and liabilities. The main focus is placed on the assets, where the evaluation of the quality of the loans portfolio (credit risk) and the securities portfolio (market risk) is more easily measurable.

Fourthly, there is the issue of configuration of an integrated invoice, which refers to the entire range of bank operations. It refers mainly to the determination of interest rates for the total of loans and deposits as well as for the various commissions which the bank charges for specific mediating operations. It is obvious that in a bank market which operates in a competitive environment, there is no issue of pricing. This is true even in the case where all interest rates and commissions are set by monetary authorities, as was the situation in Greece before the liberalization of the banking system.

In reality, bank markets have the basic characteristics of monopolistic competition. Thus, the issue of planning a system of discrete pricing and product diversification is of major importance. The problem of discrete pricing, as far as the assets are concerned, is connected to the issue of risk management. It is a common fact that the banks determine the borrowing interest rate on the basis of the interest rates which increase in proportion to the risk as they assess it in each case. The product diversification policy includes all the loan and deposit products and is based on thorough research which ensures the best possible knowledge of market conditions.

Lastly, the management of operating cost and technology constitutes an important issue. The collaboration of a well-selected and fully skilled personnel, as well as contemporary computerization systems and other technological applications, constitutes an important element in creating a low-cost bank. This results in the acquisition of a significant competitive advantage against other banks, which could finally be expressed through a more aggressive policy of attracting loans and deposits with low loan interest rates and high deposit interest rates. The result of this policy is the increase of the market stake. However, the ability of a bank to absorb the input of the best strategic technological innovations depends on the human resources management.

The present research focuses on the study of bank asset liability management. Many are the reasons that lead us to study bank asset liability management, as an application of ALM. Firstly, bank asset/liability management has always been of concern to bank managers, but in the last years and especially today its importance has grown more and more. The development of information technology has led to such an increasing public awareness that the bank's performance, its politics and its management are closely monitored by the press and the bank's competitors, shareholders and customers and thereby highly affect the bank's public standing.

The increasing competition in the national and international banking markets, the changeover towards the monetary union and the new technological innovations herald major changes in the banking environment and challenge all banks to make timely preparations in order to enter into the new competitive monetary and financial environment.

All the above drove banks to seek out greater efficiency in the management of their assets and liabilities. Thus, the central problem of ALM revolves around the bank's balance sheet and the main question that arises is: What should be the composition of a bank's assets and liabilities on average given the corresponding returns and costs, in order to achieve certain goals, such as maximization of the bank's gross revenues?

It is well known that finding an appropriate balance between profitability, risk and liquidity considerations is one of the main problems in ALM. The optimal balance between these factors cannot be found without considering important interactions that exist between the structure of a bank's liabilities and capital and the composition of its assets.

Bank asset/liability management is defined as the simultaneous planning of all asset and liability positions on the bank's balance sheet under consideration of the different banking and bank management objectives and legal, managerial and market constraints. Banks are looking to maximize profit and minimize risk.

Taking into consideration all the above, the purpose of this paper is to develop a goal programming system

into a stochastic environment, focusing, mainly, on the change of the interest rate risk. This system provides the possibility to the administrative board and the managers of the bank to proceed to various scenarios related to their future economic process, aiming mainly to the management of the risks, emerged from the changes of the market parameters.

The rest of the paper is organized as follows. The next section includes a brief overview of bank ALM techniques. Section "Model" outlines the methodology used and describes the development of the ALM decision support system. Finally, the conclusions of the paper as well as future research perspectives are discussed in the last section.

Background

Looking to the past, we find the first mathematical models in the field of bank management. Asset and liability management models can be deterministic or stochastic (Kosmidou and Zopounidis [17]).

Deterministic models use linear programming, assume particular realizations for random events, and are computationally tractable for large problems. The deterministic linear programming model of Chambers and Charnes [6] is the pioneer in ALM. Chambers and Charnes were concerned with formulating, exploring and interpreting the use and construction which may be derived from a mathematical programming model which expresses more realistically than past efforts the actual conditions of current operations. Their model corresponds to the problem of determining an optimal portfolio for an individual bank over several time periods in accordance with requirements laid down by bank examiners which are interpreted as defining limits within which the level of risk associated with the return on the portfolio is an acceptable one.

Cohen and Hammer [9], Robertson [31], Lifson and Blackman [23], Fielitz and Loeffler [14] have realized successful applications of Chambers and Charnes' model. Even though these models have differed in their treatment of disaggregation, uncertainty and dynamic considerations, they all have in common the fact that they are specified to optimize a single objective profit function subject to the relevant linear constraints.

Eatman and Sealey [12] developed a multiobjective linear programming model for commercial bank balance sheet management considering profitability and solvency objectives subject to policy and managerial constraints.

Giokas and Vassiloglou [15] developed a goal-programming model for bank asset and liability management. They supported the idea that apart from attempting to maximize revenues, management tries to minimize risks involved in the allocation of the bank's capital, as well as to fulfill other goals of the bank, such as retaining its market share, increasing the size of its deposits and loans, etc. Conventional linear programming is unable to deal with this kind of problem, as it can only handle a single goal in the objective function. Goal programming is the most widely used approach that solves large-scale multi-criteria decision making problems.

Apart from the deterministic models, several stochastic models have been proposed since the 1970s. These models, including the use of chance-constrained programming [7,8,29], dynamic programming [13,25,26,32], sequential decision theory [3,35] and stochastic linear programming under uncertainty [2,10,11,16], presented computational difficulties. The stochastic models, in their majority, originate from the portfolio selection theory of Markowitz [24] and they are known as static mean-variance methods. Pyle [30] and Brodt [4] adapted Markowitz's theory and presented an efficient dynamic balance sheet management plan that considers only the risk of the portfolio and not other possible uncertainties or maximizes profits for a given amount of risk over a multi-period planning horizon respectively.

Wolf [35] proposed the sequential decision theoretic approach that employs sequential decision analysis to find an optimal solution through the use of implicit enumeration.

An alternative approach in considering stochastic models, is the stochastic linear programming with simple recourse. Kusy and Ziemba [19] employed a multiperiod stochastic linear program with simple recourse to model the management of assets and liabilities in banking while maintaining computational feasibility. Their results indicate that the proposed ALM model is theoretically and operationally superior to a corresponding deterministic linear programming model and that the computational effort required for its implementation is comparable to that of the deterministic

model. Another application of the multistage stochastic programming is the Russell-Yasuda Kasai model [5], which aims at maximizing the long term wealth of the firm while producing high income returns.

Mulvey and Vladimirou [27] used dynamic generalized network programs for financial planning problems under uncertainty and they developed a model in the framework of multi-scenario generalized network that captures essential features of various discrete time financial decision problems.

Finally, Mulvey and Ziemba [28] present a more detailed overview of various asset and liability modeling techniques, including models for individuals and financial institutions such as banks and insurance companies.

Moreover, over the years, many models have been developed in the area of financial analysis and financial planning techniques. Kvanli [20], Lee and Lerro [22], Lee and Chesser [21], Baston [1], Sharma et al. [34], among others have applied goal programming to investment planning. Giokas and Vassiloglou [15], Seshadri et al. [33] presented bank models using goal programming. These studies focus on the areas of banking and financial institutions and they use data from the bank financial statements.

Model

Kosmidou and Zopounidis [18] developed an asset liability management (ALM) methodology into a stochastic environment of interest rates in order to select the best direction strategies to the banking financial planning. The ALM model was developed through goal programming in terms of a one-year time horizon. The model used balance sheet and income statement information for the previous year of the year t to produce a future course of ALM strategy for the year $t+1$. As far as model variables are concerned, we used variables familiar to management and facilitated the specification of the constraints and goals. For example, goals concerning measurements such as liquidity, return and risk have to be expressed in terms of utilized variables.

More precisely, the asset liability management model that was developed can be expressed as follows:

$$\min z = \sum_{P} p_k (d_k^- + d_k^+) \tag{1}$$

subject to constraints:

$$\mathrm{K}\Phi_{X'} \le X' \le \mathrm{A}\Phi_{X'} \tag{2}$$

$$\mathrm{K}\Phi_{Y'} \le Y' \le \mathrm{A}\Phi_{Y'} \tag{3}$$

$$\sum_{i=1}^{n} X_i = \sum_{j=1}^{m} Y_j \quad \forall i = 1, \ldots, n, \quad \forall j = 1, \ldots, m \tag{4}$$

$$\sum_{j \in \Pi_{Y''}} Y_j - a \sum_{i \in \mathrm{E}_{X''}} X_i = 0 \tag{5}$$

$$\sum_{j \in \Pi_1} Y_j - \sum_{i \in \mathrm{E}} w_i X_i - d_s^+ + d_s^- = k_1 \tag{6}$$

$$\sum_{i \in \mathrm{E}_x} X_i - k_2 \sum_{j \in \Pi_k} Y_j + d_l^- - d_l^+ = 0 \tag{7}$$

$$\sum_{i=1}^{n} R_i^X X_i - \sum_{j=1}^{m} R_j^Y Y_j - d_r^+ + d_r^- = k_3 \tag{8}$$

$$\sum_{i \in \mathrm{E}_p} X_i + d_p^- - d_p^+ = l_p\,, \quad \forall p \tag{9}$$

$$\sum_{j \in \Pi_p} Y_j + d_p^- - d_p^+ = l_p\,, \quad \forall p \tag{10}$$

$$X_i \ge 0, Y_j \ge 0, d_k^+ \ge 0, d_k^- \ge 0\,, \quad \text{for all } i = 1, \ldots, n, j = 1, \ldots, m, k \in P \tag{11}$$

where

- $\boldsymbol{X_i}$: the element i of asset, $\forall i = 1, \ldots, n$, n is the number of asset variables
- $\boldsymbol{Y_j}$: the element j of liability, $\forall j = 1, \ldots, m$, m is the number of liability variables
- $\mathrm{K}\Phi_{X'}$ ($\mathrm{K}\Phi_{Y'}$): is the low bound of specific asset accounts X′ (liability Y′)
- $\mathrm{A}\Phi_{X'}$ ($\mathrm{A}\Phi_{Y'}$): is the upper bound of specific asset accounts X′ (liability Y′)
- $\mathrm{E}_{X''}$: specific categories of asset accounts
- $\Pi_{Y''}$: specific categories of liability accounts
- α: the desirable value of specific asset and liability data
- Π_1: the liability set, which includes the equity
- **E:** the set of assets
- $\boldsymbol{w_i}$: the degree of riskness of the asset data
- $\boldsymbol{k_1}$: the solvency ratio, as it is defined from the European Central Bank.

k_2: the liquidity ratio, as it is defined from the bank policy
E_χ: the set of asset data, which includes the loans
Π_κ: the set of liability data, which includes the deposits
R_i^X: the expected return of the asset i, $\forall i = 1, \ldots, n$
R_j^Y: the expected return of the liability j, $\forall j = 1, \ldots, m$
k_3: the expected value for the goal of asset and liability return
P: the goal imposed from the bank
L_p: the desirable value goal for the goal constraint p defined by the bank
d_k^+: the over-achievement of the goal k, $\forall k \in P$
d_k^-: the under-achievement of the goal k, $\forall k \in P$
p_k: the priority degree (weight) of the goal k

Certain constraints are imposed by the banking regulation on particular categories of accounts. Specific categories of asset accounts (X') and liability accounts (Y') are detected and the minimum and maximum allowed limit for these categories are defined based on the strategy and policy that the bank intends to follow (constraints 2–3).

The structural constraints (4–5) include those that contribute to the structure of the balance sheet and especially to the performance of the equation Assets = Liabilities + Net Capital.

The bank management should determine specific goals, such as the desirable structure of each financial institution's assets and liabilities for the units of surplus and deficit, balancing the low cost and the high return. The structure of assets and liabilities is significant, since it affects swiftly the income and profits of the bank.

Referring to the goals of the model, the solvency goal (6) is used as a risk measure and is defined as the ratio of the bank's equity capital to its total weighted assets. The weighting of the assets reflects their respective risk, greater weights corresponding to a higher degree of risk. This hierarchy takes place according to the determination of several degrees of significance for the variables of assets and liabilities. That is, the variables with the largest degrees of significance correspond to categories of the balance sheet accounts with the highest risk stages.

Moreover, a basic policy of the commercial banks is the management of their liquidity and specifically the measurement of their needs that is relative to the progress of deposits and loans. The liquidity goal (7) is defined as the ratio of liquid assets to current liabilities and indicates the liquidity risk, that indicates the possibility of the bank to respond to its current liabilities with a security margin, which allows the probable reduction of the value of some current data.

Furthermore, the bank aims at the maximization of its efficiency that is the accomplishment of the largest possible profit from the best placement of its funds. Its aim is the maximization of its profitability and therefore precise and consistent decisions should be taken into account during the bank management. These decisions will guarantee the combined effect of all the variables that are included on the calculation of the profits. This decision taking gives emphasis to several selected variables that are related to the bank management, such as to the management of the difference between the asset return and the liability cost, the expenses, the liquidity management and the capital management. The goal (8) determines the total expected return based on the expected returns for all the assets R^X and liabilities R^Y.

Beside the goals of solvency, liquidity and return of assets and liabilities, the bank could determine other goals that concern specific categories of assets and liabilities, in proportion to the demands and preferences of the bank managers. These goals are the deposit goal, the loan goal and the goal of asset and liability return.

The drawing of capital, especially from the deposits constitutes a major part of commercial bank management. All sorts of deposits constitute the major source of capital for the commercial banks, in order to proceed to the financing of the economy, through the financing of firms. Thus, it is given special significance to the deposits goal.

The goal of asset and liability return defines the goal for the overall expected return of the selected asset-liability strategy over the year of the analysis.

Finally, there are goals reflecting that variables such as cash, cheques receivables, deposits to the Bank of Greece and fixed assets, should remain at the levels of previous years. More analytically, it is known that the fixed assets are the permanent assets, which have a natural existence, such as buildings, machines, locations and equipment, etc. Intangible assets are the fixed assets, which have no natural existence but constitute rights and benefits. They have significant economic value, which sometimes is larger than the value of the

tangible fixed assets. These data have stable character and are used productively by the bank for the regular operation and performance of its objectives. Since the fixed assets, tangible or intangible, are presented at the balance sheet at their book value that is the initial value of cost minus the depreciation till today, it is assumed that their value does not change during the development of the present methodology.

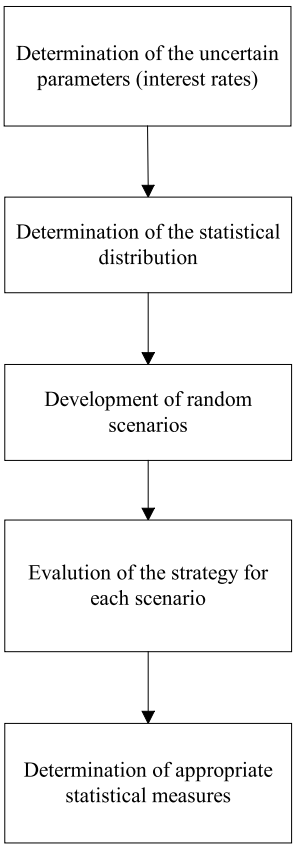
At this point, Kosmidou and Zopounidis [18] took into account that the banks should manage the interest rate risk, the operating risk, the credit risk, the market risk, the foreign exchange risk, the liquidity risk and the country risk.

More specifically, the interest rate risk indicates the effect of the changes to the net profit margin between the deposit and borrowing values, which are evolved as a consequence of the deviations to the dominant interest rates of assets and liabilities. When the interest rates diminish, the banks accomplish high profits since they can refresh their liabilities to lower borrowing values. The reverse stands to high borrowing values. It is obvious, that the changes of the inflation have a relevant impact on the above sorts of risk.

Considering the interest rate risk as the basic uncertainty parameter to the determination of a bank asset liability management strategy, the crucial question that arises concerns the determination of the way through which this factor of uncertainty affects the profitability of the pre-specified strategy. The estimation of the expected return of the pre-specified strategy and of its variance can render a satisfactory response to the above question.

The use of Monte Carlo techniques constitutes a particular widespread approach for the estimation of the above information (expected return – variance of bank asset liability management strategies). Monte Carlo simulation consists in the development of various random scenarios for the uncertain variable (interest rates) and the estimation of the essential statistical measures (expected return and variance), which describe the effect of the interest rate risk to the selected strategy. The general procedure of implementation of Monte Carlo simulation based on the above is presented in Fig. 1.

During the first stage of the procedure the various categories of the interest rate risks are identified. The risk and the return of the various data of bank asset and liability are determined from the different forms of interest rates. For example, the investments of a bank to government or corporate bonds are determined from the interest rates that prevail in the bond market, which are affected so by the general economic environment as by the rules of demand and supply. Similarly, the deposits and loans of the bank are determined from the corresponding interest rates of deposits and loans, which are assigned by the bank according to the conditions that prevail to the bank market. At this stage, the categories of the interest rates, which constitute crucial uncertain variables for the analysis, are detected. The determined interest rates categories depend on the type of the bank. For example, for a decisive commercial bank, the deposit and loan interest rates have a role, whereas for an investment bank more emphasis is given



Asset Liability Management Decision Support System, Figure 1
General Monte Carlo simulation procedure for the evaluation of the asset liability management strategies

to the interest rates and the returns of various investment products (repos, bonds, interest-bearing notes, etc.).

After the determination of the various categories of interest rates, which determine the total interest rate risk, at the second stage of the analysis the statistical distribution that follows each of the pre-specified categories should be determined.

Having determined the statistical distribution that describes the uncertain variables of the analysis (interest rates), a series of random independent scenarios is developed, through a random number generator. Generally, the largest the number of scenarios that are developed, the more reliable conclusions can be derived. However, the computational effort increases significantly, since for each scenario the optimal asset liability strategy should be determined and moreover its evaluation for each other scenario should take place. Thus, the determination of the number volume N of simulations (scenarios), which will take place should be determined, taking into account both the reliability of the results and the available computational resources.

For each scenario s_i ($i = 1, 2, \ldots, N$) over the interest rates the optimal asset liability management strategy Υ_i is determined through the solution of the goal programming problem. It is obvious that this strategy is not expected to be optimal for each of the other scenarios s_j ($j \neq i$). Therefore the results obtained from the implementation of the strategy Y_i under the rest N–1 possible scenarios s_j should be evaluated. The evaluation of the results can be implemented from various directions. The most usual is the one that uses the return. Representing as r_{ij} the outcome (return) of the strategy Υ_i under the scenario s_j, the expected return $\bar{r}_i$ of the strategy can be easily determined based on all the other N–1 scenarios s_j ($j \neq i$), as follows:

$$\bar{r}_i = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} r_{ij} \tag{12}$$

At the same time, the variance σ_i^2 of the expected return can be determined as a risk measure of the strategy Y_i, as follows:

$$\sigma_i^2 = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \left(r_{ij} - \bar{r}_i\right)^2 \tag{13}$$

These two statistical measures (average and variance) contribute to the extraction of useful conclusions concerning the expected efficiency of the asset liability management strategy, as well as the risks that it carries. Moreover, these two basic statistical measures can be used for the expansion of the analysis of the determination of other useful statistical information, such as the determination of the confidence interval for the expected return, the quantiles, etc.

Conclusions

The banking business has recently become more sophisticated due to technological expansion, economic development, creation of financial institutions and increased competition. Moreover, the mergers and acquisitions that have taken place the last years create large groups of banking institutions. The success of a bank depends mainly on the quality of its asset and liability management, since the latter deals with the efficient management of sources and uses of bank funds concentrating on profitability, liquidity, capital adequacy and risk factors.

It is obvious that in the last two decades modern finance has developed into a complex mathematically challenging field. Various and complicated risks exist in financial markets. For banks, interest rate risk is at the core of their business and managing it successfully is crucial to whether or not they remain profitable. Therefore, it has been essential the creation of the department of financial risk management within the banks. Asset liability management is associated with the changes of the interest rate risk. Although several models exist regarding asset liability management, most of them are focused on the general aspects and methodologies of this field and do not refer extensively to the hedging of bank interest rate risk through asset liability management. Thus, the main purpose of the present paper was to describe the development of a bank ALM decision support system, which gives the possibility to the decision maker to proceed to various scenarios of the economic process of the bank in order to monitor its financial situation and to determine the optimal strategic implementation of the composition of assets and liabilities. Moreover, we believe that the development of a bank asset liability management model that takes

into account the exogenous factors and the economic parameters of the market as well as the uncertainty of variations of the financial risks become essential.

Finally, despite the approaches described in this paper, little academic work has been done so far to develop a model for the management of assets and liabilities in the European banking industry. Based on the above we conclude that the quality of asset liability management in the European banking system has become significant as a resource of competitive advantage. Therefore, the development of new technological approaches in bank asset liability management in Europe is worth further research.

References

1. Baston RG (1989) Financial planning using goal programming. Long Range Plan 22(17):112–120
2. Booth GG (1972) Programming Bank Portfolios under Uncertainty: An Extension. J Bank Res 2:28–40
3. Bradley SP, Crane DB (1972) A Dynamic Model for Bond Portfolio Management. Manage Sci 19:139–151
4. Brodt AI (1978) Dynamic Balance Sheet Management Model for a Canadian Chartered Bank. J Bank Financ 2(3): 221–241
5. Carino DR, Kent T, Muyers DH, Stacy C, Sylvanus M, Turner AL, Watanabe K, Ziemba WT (1994) The Russell-Yasuda Kasai Model: An Asset/Liability Model for a Japanese Insurance Company Using Multistage Stochastic Programming. Interfaces 24:29–49
6. Chambers D, Charnes A (1961) Inter-Temporal Analysis and Optimization of Bank Portfolios. Manage Sci 7:393–410
7. Charnes A, Littlechild SC (1968) Intertemporal Bank Asset Choice with Stochastic Dependence. Systems Research Memorandum no.188, The Technological Institute, Nortwestern University, Evanston, Illinois
8. Charnes A, Thore S (1966) Planning for Liquidity in Financial Institution: The Chance Constrained Method. J Finance 21(4):649–674
9. Cohen KJ, Hammer FS (1967) Linear Programming and Optimal Bank Asset Management Decisions. J Financ 22: 42–61
10. Cohen KJ, Thore S (1970) Programming Bank Portfolios under Uncertainty. J Bank Res 2:28–40
11. Crane B (1971) A Stochastic Programming Model for Commercial Bank Bond Portfolio Management. J Finance Quant Anal 6:955–976
12. Eatman L, Sealey W (1979) A Multi-objective Linear Programming Model for Commercial bank Balance Sheet Management. J Bank Res 9:227–236
13. Eppen GD, Fama EF (1971) Three Asset Cash Balance and Dynamic Portfolio Problems. Manage Sci 17:311–319
14. Fielitz D, Loeffler A (1979) A Linear Programming Model for Commercial Bank Liquidity Management. Finance Manage 8(3):44–50
15. Giokas D, Vassiloglou M (1991) A Goal Programming Model for Bank Assets and Liabilities. Eur J Oper Res 50:48–60
16. Kallberg JG, White RW, Ziemba WT (1982) Short Term Financial Planning under Uncertainty. Manage Sci 28:670–682
17. Kosmidou K, Zopounidis C (2001) Bank Asset Liability Management Techniques: An Overview. In: Zopounidis C, Pardalos PM, Baourakis G (eds) Fuzzy Set Systems in Management and Economy. World Scientific Publishers, pp 255–268
18. Kosmidou K, Zopounidis C (2004) Combining Goal Programming Model with Simulation Analysis for Bank Asset Liability Management. Inf Syst Oper Res J 42(3):175–187
19. Kusy IM, Ziemba TW (1986) A Bank Asset and Liability Management model. Oper Res 34(3):356–376
20. Kvanli AH (1980) Financial planning using goal programming-OMEGA. Int J Manag Sci 8(2):207–218
21. Lee SM, Chesser DL (1980) Goal programming for portfolio selection. J Portf Manag 6:22–26
22. Lee SM, Lerro AJ (1973) Optimizing the portfolio selection for mutual funds. J Financ 28:1086–1101
23. Lifson KA, Blackman BR (1973) Simulation and Optimization Models for Asset Deployment and Funds Sources Balancing Profit Liquidity and Growth. J Bank Res 4(3):239–255
24. Markowitz HM (1959) Portfolio Selection. Efficient Diversification of Investments. Wiley, New York
25. Merton RC (1969) Lifetime portfolio selection under certainty: the continuous time case. Rev Eco Stat 3:373–413
26. Merton RC (1990) Continuous-Time Finance. Blackwell Publishers, Merton, UK
27. Mulvey JM, Vladimirou H (1989) Stochastic Network Optimization Models of Investment Planning. Annal Oper Res 20:187–217
28. Mulvey JM, Ziemba WT (1998) Asset and liability management systems for long-term investors: discussion of the issues. In: Ziemba W, Mulvey J (eds) Worldwide Asset and Liability Modelling. Cambridge University Press, Mulvey, UK, pp 3–38
29. Pogue GA, Bussard RN (1972) A Linear Programming Model for Short Term Financial Planning under Uncertainty. Sloan Manag Rev 13:69–98
30. Pyle DH (1971) On the Theory of Financial Intermediation. J Financ 26:737–746
31. Robertson M (1972) A Bank Asset Management Model. In: Eilon S, Fowkes TR (eds) Applications of Management Science in Banking and Finance. Gower Press, Epping, Essex, pp 149–158
32. Samuelson P (1969) Lifetime portfolio selection by dynamic stochastic programming. Rev Eco Stat (August), 239–246

33. Seshadri S, Khanna A, Harche F, Wyle R (1999) A method for strategic asset-liability management with an application to the federal home loan bank of New York. Oper Res 47(3):345–360
34. Sharma JK, Sharma DK, Adeyeye JO (1995) Optimal portfolio selection: A goal programming approach. Indian J Financ Res 7(2):67–76
35. Wolf CR (1969) A Model for Selecting Commercial Bank Government Security Portfolios. Rev Econ Stat 1:40–52

Assignment and Matching

AM

Dimitris Alevras
IBM Corporation, West Chester, USA

MSC2000: 90C35, 90C27, 90C10, 90C05

Article Outline

Keywords
Maximum Cardinality Bipartite Matching Problem
Weighted Bipartite Matching Problem
Weighted Matching Problem
Maximum Cardinality Matching Problem
See also
References

Keywords

Optimization; Integer programming; Graph theory; Marriage problem

Matching problems comprise an important set of problems that link the areas of graph theory and combinatorial optimization. The maximum cardinality matching problem (see below) is one of the first integer programming problems that was solved in polynomial time. Matchings are of great importance in graph theory (see [9]) as well as in combinatorial optimization (see e. g. [15]).

The matching problem and its variations arise in cases when we want to find an 'optimal' pairing of the members of two (not necessarily disjoint) sets. In particular, if we are given two sets of 'objects' and a 'weight' for each pair of objects, we want to match the objects into pairs in such a way that the total weight is maximal. In graph theory, the problem is defined on a graph $G = (V, E)$ where V is the node set of the graph, corresponding to the union of the two sets of objects, and E is the edge set of the graph corresponding to the possible pairs. A pair is possible if there exists an edge between the corresponding nodes. A *matching* M is a subset of the edges E with the property that each node in V is incident to at most one edge in M. If each node in V is met by exactly one edge in M, then M is called a *perfect matching*. There exist several versions of the matching problem, depending on whether the graph G is bipartite or not (i. e., the two sets of objects are disjoint or not), and on whether we want to find the maximum size (cardinality) or the maximum weight of the matching. The book [1] gives several applications of the matching problem.

Maximum Cardinality Bipartite Matching Problem

The graph G is *bipartite* if the node set V can be partitioned into two disjoint sets V_1 and V_2 such that no edge in E connects nodes from the same set. Finding a maximum cardinality matching on a bipartite graph can be solved by several efficient algorithms with a worst-case bound of $O(\sqrt{n}m)$, where n is the number of nodes and m the number of edges of the graph. See [1] for details.

Weighted Bipartite Matching Problem

This problem is known as the *assignment* or the *marriage* problem. In the traditional definition it is required that the sets V_1 and V_2 are of equal size, but even if not, one can add 'dummy' nodes to the smaller set to satisfy this condition. This problem can be formulated as a zero-one linear programming problem as follows:

$$\begin{cases} \min & \displaystyle\sum_{(u,v)\in E} f(u,v)x_{uv} \\ \text{s.t.} & \displaystyle\sum_{(u,v)\in E} x_{uv} = 1 \quad \text{for all } u \in V_1, \\ & \displaystyle\sum_{(u,v)\in E} x_{uv} = 1 \quad \text{for all } v \in V_2, \\ & x_{uv} \in \{0,1\} \quad \text{for all } u \in V_1,\ v \in V_2. \end{cases}$$

The *assignment problem* has the property that if solved as a linear programming problem in nonnegative x_{uv} it yields an integer solution, i. e., the zero-one integrality condition in the formulation is not necessary. This is

so because the constraint matrix of the equations is *totally unimodular*, i. e., the determinant of every square submatrix of it is 0 or ± 1. This means that if the right-hand sides of the equations are integer numbers, as is the case in the assignment problem, then the solution will be integer.

Linear programming algorithms are not as efficient as specialized algorithms for solving the assignment problem. The assignment problem is a special case of the minimum cost flow problem, and adaptations of algorithms for that problem that take into account the special structure of the assignment problem yield the most efficient algorithms. Probably the best known algorithm is the so called *Hungarian algorithm*, see [8], which is a primal-dual algorithm for the minimum cost flow problem. See [1] for details and other algorithms.

Variations of the bipartite matching include among others the order preserving assignment problem and the stable marriage problem. In the *order preserving assignment problem* the assignment must be such that a prespecified order among the objects of one of the node partitions is preserved. Although the linear programming formulation of this problem is more complicated than that of the assignment problem, the problem itself is easier to solve than the assignment problem and can be solved in $O(m)$ time where m is the number of edges in the graph; see [2,12]. In the *stable marriage problem* each object of one partition has a ranking (or preference) for each of the objects of the other partition, and the assignment must be such that there is no nonmatched pair of objects that its members prefer each other to the ones they are matched against. This problem can be solved in $O(n^2)$ time using a greedy algorithm (n is the number of nodes in one partition). See [1].

Weighted Matching Problem

The weighted matching problem can be formulated as a 0–1 programming problem as follows:

$$\begin{cases} \max & \displaystyle\sum_{(u,v)\in E} f(u,v)x_{uv} \\ \text{s.t.} & \displaystyle\sum_{(u,v)\in E} x_{uv} \le 1 \quad \text{for all } u \in V, \\ & x_{uv} \in \{0,1\} \quad \text{for all } (u,v) \in E. \end{cases}$$

Unlike the case of the assignment problem, relaxing the integrality constraints yields, in general, a fractional solution.

Maximum Cardinality Matching Problem

J. Edmonds showed in [5] that one more set of inequalities—the *odd-set constraints*—is needed in order to get a linear programming formulation of the matching problem. The odd-set or *blossom* inequalities are

$$\sum_{(u,v)\in E(U)} x_{uv} \le \left\lfloor \frac{|U|}{2} \right\rfloor, \quad \forall \text{odd } U \subseteq V,\ |U| \ge 3,$$

where $E(U)$ is the set of all edges in E with both end nodes in U. An odd set is a set of odd cardinality. See also [11].

Solving the matching problem on nonbipartite graphs is considerably more difficult than on bipartite ones. This is so because the path augmenting algorithms used in the case of bipartite matchings, may fail when a structure called blossom is encountered. Edmonds provided an $O(n^4)$ algorithm that would find an integer solution to the linear programming relaxation of the formulation (including the odd-set constraints) for any objective function, proving this way the completeness of the formulation. Several implementations that improved the performance of the algorithm have been proposed (see [1,10], among others) as well as data structures for the efficient implementation of such algorithms (see [3]). M. Grötschel and O. Holland [6] gave a cutting plane algorithm for the weighted matching problem, where they used an efficient separation algorithm to identify violated *blossom inequalities*, based on the algorithm of M.W. Padberg and M.R. Rao [14] for the *b-matching problem*.

The b-matching problem is an important generalization of the matching problem. In the b-matching problem each node $v \in V$ is met by no more than b_v edges; thus, in this context, the previous definition of matching corresponds to an 1-matching. A *perfect b-matching* is one in which each node $v \in V$ is met by exactly b_v edges. If it is permitted to chose an edge more than one times then the problem becomes a general integer program instead of a 0–1 program. The b-matching problem can be reduced to 1-matching problem on an appropriately constructed graph. Although this procedure is not polynomial in gen-

eral—and thus, Edmonds' algorithm can not be readily applied—the *b*-matching problem is polynomially solvable; see [14] and [7]. A linear inequality description for the integer *b*-matching problem is given in [15]. See also [11]. The perfect 0–1 2-matching problem is a relaxation of the *traveling salesman problem* (TSP). Solving the 0–1 2-matching problem yields a heuristic solution to the TSP which is an *NP*-hard problem; see [13].

See also

- ▶ Assignment Methods in Clustering
- ▶ Bi-Objective Assignment Problem
- ▶ Communication Network Assignment Problem
- ▶ Frequency Assignment Problem
- ▶ Maximum Partition Matching
- ▶ Quadratic Assignment Problem

References

1. Ahuja R, Magnanti T, Orlin J (1994) Network flows. Wiley, New York
2. Alevras D (1997) Order preserving assignments without contiguity. Discret Math 163:1–11
3. Ball MO, Derigs U (1983) An analysis of alternate strategies for implementing matching algorithms. Networks 13:517–549
4. Edmonds J (1965) Maximum matching and a polyhedron with (0, 1) vertices. J Res Nat Bureau Standards (B) 69B:125–130
5. Edmonds J (1965) Paths, trees, and flowers. Canad J Math 17:449–467
6. Grötschel M, Holland O (1985) Solving matching problems with linear programming. Math Program 33:243–259
7. Grötschel M, Lovasz L, Schrijver A (1988) Geometric algorithms and combinatorial optimization. Springer, Berlin
8. Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Res Logist Quart 2:83–97
9. Lovasz L, Plummer M (1986) Matching theory. North-Holland, Amsterdam
10. Micali S, Vazirani VV (1980) An $O(\sqrt{(|v|)}|E|)$ algorithm for finding maximum matching in general graphs. IEEE Symp Found Computer Sci, pp 17–27
11. Nemhauser GL, Wolsey L (1988) Integer and combinatorial optimization. Wiley, New York
12. Padberg M, Alevras D (1994) Order-preserving assignments. Naval Res Logist 41:395–421
13. Padberg MW, Grötschel M (1985) Polyhedral theory. The Traveling Salesman Problem. Wiley, New York, pp 251–305
14. Padberg M, Rao MR (1982) Odd minimum cut-sets and b-matchings. Math Oper Res 7:67–80
15. Pulleyblank WR (1989) Polyhedral combinatorics. Optimization, vol 1 of Handbook Oper Res and Management Sci. North-Holland, Amsterdam, pp 371–446

Assignment Methods in Clustering

L. J. HUBERT[1], P. ARABIE[2]
[1] University Illinois, Champaign, USA
[2] Rutgers University, Newark, USA

MSC2000: 62H30, 90C27

Article Outline

Keywords
Weighting Schemes
  for the Fixed (Target) Matrix Q
  Single Cluster Statistics
  Partition Statistics
  Partition Hierarchy Statistics
Alternative Assignment Indices
Modifications of the Target Matrix Q
See also
References

Keywords

Combinatorial optimization; Quadratic assignment; Clustering

The use of *assignment methods* in the formulation of various optimization problems encountered in *clustering* and *classification*, can be introduced through the well-known *quadratic assignment* (QA) model (see [5] for a comprehensive discussion of most of the topics presented in this entry). In its most basic form the QA optimization task can be stated using two $n \times n$ matrices, say $\mathbf{P} = \{ p_{ij} \}$, and $\mathbf{Q} = \{ q_{ij} \}$, and the identification of a one-to-one function (or a permutation), $\rho(\cdot)$, on the first n integers, to optimize (either by minimizing or maximizing) the cross-product index

$$\Gamma(\rho) = \sum_{i,j} p_{\rho(i)\rho(j)} q_{ij}. \tag{1}$$

Typically, the main diagonal entries in $\mathbf{P}$ and $\mathbf{Q}$ are considered irrelevant and can be set equal to zero. For arbitrary matrices $\mathbf{P}$ and $\mathbf{Q}$, the cross product index in (1)

may be rewritten as

$$\sum_{i,j}\left(\frac{p_{\rho(i)\rho(j)}+p_{\rho(j)\rho(i)}}{2}\right)\left(\frac{q_{ij}+q_{ji}}{2}\right)$$
$$+\sum_{i,j}\left(\frac{p_{\rho(i)\rho(j)}-p_{\rho(j)\rho(i)}}{2}\right)\left(\frac{q_{ij}-q_{ji}}{2}\right),$$

indicating that the optimization of (1) jointly involves the *symmetric* ($[\mathbf{P}+\mathbf{P}']/2$ versus $[\mathbf{Q}+\mathbf{Q}']/2$) and *skew-symmetric* ($[\mathbf{P}-\mathbf{P}']/2$ versus $[\mathbf{Q}-\mathbf{Q}']/2$) components of both **P** and **Q**. Because of this separation of **P** and **Q** into symmetric and skew-symmetric components, it is possible in the context of the clustering/classification tasks to be discussed below, to assume that both **P** and **Q** are symmetric or that both are skew-symmetric.

In applications to clustering, the matrix **P** usually contains numerical proximity information between distinct pairs of the n objects from some given set $S = \{O_1, \ldots, O_n\}$ that is of substantive interest. If **P** is symmetric, p_{ij} ($= p_{ji}$) denotes the degree to which objects O_i and O_j are similar (and keyed as what is referred to as a *dissimilarity* [or as a *similarity*] measure if smaller [or larger] values reflect greater object similarity). If **P** is skew-symmetric, p_{ij} ($= -p_{ji}$) is an index of dominance (or *flow*) between objects O_i and O_j, with the sign reflecting the directionality of *dominance* and the absolute value indicating the degree. The (target) matrix **Q**, as developed in detail in the next section, will typically be fixed, with the specific pattern of entries characterizing the type of structure to be identified for the set S, e. g., a single object *cluster*, a *partition*, or a *partition hierarchy*. An optimal permutation, say, $\rho^*(\cdot)$, based on the cross-product index in (1) for a specific target matrix **Q** will identify the (salient) combinatorial structure sought.

The QA *optimization* task as formulated through (1) has an enormous literature that will not be reviewed here (for an up-to-date and comprehensive source on QA, see [11]). For current purposes, one might consider the optimization of (1) through a simple object interchange heuristic that would begin with some permutation (possibly chosen at random), and then implement local interchanges until no improvement in the index can be made. By repeatedly initializing such a process randomly, a distribution over a set of *local optima* can be achieved. At least within the context of clustering/classification, such a distribution may be highly relevant diagnostically for explaining whatever structure is inherent in the data matrix **P**, and possibly of even greater interest than the identification of just a single optimal permutation. In a related framework, there are considerable applications for the QA model in a confirmatory context where the distribution of $\Gamma(\rho)$ is constructed over all $n!$ possible permutations considered equally-likely, and the index value associated with some identified permutation is compared to this distribution. Most *nonparametric statistical methods* popular in the literature can be rephrased through the device of defining the matrices **P** and **Q** appropriately (see [5] for a comprehensive development of these special cases as well as approximation methods based on closed-form expressions for the first three moments of $\Gamma(\rho)$). A few of these applications will be briefly noted below.

Weighting Schemes for the Fixed (Target) Matrix Q

Single Cluster Statistics

To identify a *single* salient cluster of fixed size K (that can be varied by the user), consider **Q** to have the partitioned form

$$\mathbf{Q}=\begin{pmatrix}\mathbf{Q}_{11} & \mathbf{Q}_{12}\\ \mathbf{Q}_{21} & \mathbf{Q}_{22}\end{pmatrix},$$

where within each submatrix of the size indicated, the (off-diagonal) entries are constant:

$$\mathbf{Q}_{11}=\begin{pmatrix}0 & \cdots & q_{11}\\ \vdots & \ddots & \vdots\\ q_{11} & \cdots & 0\end{pmatrix}_{K\times K}$$

$$\mathbf{Q}_{12}=\begin{pmatrix}\vdots & \\ \cdots\ q_{12} & \cdots\\ \vdots & \end{pmatrix}_{K\times(n-K)}$$

$$\mathbf{Q}_{21}=\begin{pmatrix}\vdots & \\ \cdots\ q_{21} & \cdots\\ \vdots & \end{pmatrix}_{(n-K)\times K}$$

$$\mathbf{Q}_{22}=\begin{pmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{pmatrix}_{(n-K)\times(n-K)}$$

Depending on how the values for q_{11}, q_{12}, and q_{21} are defined, different indices can be generated that measure

the salience of the subset constructed by any permutation $\rho(\cdot)$, i. e., for the identified cluster $S_\rho \equiv \{O_{\rho(1)}, \ldots, O_{\rho(K)}\}$.

For symmetric **P**:

A) letting

$$q_{11} = \frac{1}{K(K-1)}, \quad q_{12} = q_{21} = 0,$$

the index $\Gamma(\rho)$ is the average proximity within the subset S_ρ and defines a measure of *cluster 'compactness'*;

B) letting

$$q_{11} = 0, \quad q_{12} = q_{21} = \frac{1}{2K(n-K)},$$

$\Gamma(\rho)$ is the average proximity between the subset S_ρ and its complement, and defines a measure of *cluster 'isolation'* for either S_ρ or $S - S_\rho$; alternatively, it can be considered a measure of *'separation'* between S_ρ or $S - S_\rho$;

C) by contrasting A) and B) as

$$q_{11} = \frac{1}{K(K-1)},$$
$$q_{12} = q_{21} = -\frac{1}{2K(n-k)},$$

$\Gamma(\rho)$ characterizes the salience of the subset S_ρ by a trade-off between compactness and isolation. The optimization of $\Gamma(\rho)$ based on these latter weights identifies a cluster that would be both relatively compact and isolated, whereas the emphasis in A) and B) are on clusters that may be either compact or isolated but not necessarily both.

For skew-symmetric **P**:

D) letting

$$q_{11} = 0, \quad q_{12} = \frac{1}{2K(n-K)}, \quad q_{21} = -q_{12},$$

the index $\Gamma(\rho)$ is the average dominance (or flow) from the subset S_ρ to its complement, minus the average dominance (or flow) from the complement to the subset. Thus, its optimization (e. g., maximization) identifies a subset of S whose members tend to dominate those in its complement (or where aggregate outflow exceeds aggregate inflow).

In a confirmatory comparison context, the single-cluster statistic $\Gamma(\rho)$ can be used to generate a number of nonparametric test statistics for comparing the difference between two independent groups. For example, suppose observations are available on n objects, $x_1, \ldots, x_n$, where the first K belong to group I and the last $n - K$ to group II. If the (now asymmetric) proximity matrix is defined as $\mathbf{P} = \{p_{ij}\}$, where $p_{ij} = 1$ if $x_j < x_i$ and $= 0$ if $x_j \geq x_i$ then the weighting scheme in B) gives (a simple linear transform of) the well-known *Mann–Whitney statistic* for comparing two-independent groups, i. e., if two observations are drawn at random from groups I and II, then $\Gamma(\rho_o)$, for ρ_o the identity permutation, is the probability that the group I observation is the larger. The distribution of $\Gamma(\rho)$ over all $n!$ permutations generates the null distribution against which the observed index $\Gamma(\rho_o)$ can be compared. Because of the structure of **Q**, this null distribution is based on all $n!/(K!(n - K)!)$ distinct subsets considered equally-likely to be formed from the collection of size n. (See [3, Chap. 7], for a more complete discussion of the two-independent sample problem in this type of nonparametric framework.)

Although single-cluster statistics that depend on the comparison of mean proximities may be the most obvious to consider, a number of possible alternatives can be constructed by varying the definition for the weight matrices in **Q**. For example, for symmetric **P**, if $\mathbf{Q}_{11}$ is (re)defined to have the form

$$\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{pmatrix},$$

with entries of all ones immediately above and below the main diagonal, and $q_{12} = q_{21} = 0$, the salience of S_ρ is now based on (twice) the sum of adjacent proximities along a path of *length* K considered in the object order $O_{\rho(1)} \leftrightarrow \cdots \leftrightarrow O_{\rho(K)}$. Or, if $\mathbf{Q}_{11}$ is (re)defined to have the form

$$\begin{pmatrix} 0 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix},$$

and $q_{12} = q_{21} = 0$, the salience of S_ρ is now based on (twice) the sum of proximities between $O_{\rho(1)}$ and the

remaining objects $O_{\rho(2)}, \ldots, O_{\rho(K)}$ (this is called a *'star' cluster* of size K with object $O_{\rho(1)}$ as its center; see [10, Sect. 4.5.2] for a further discussion of clustering based on stars).

Partition Statistics

To identify a salient partition of S into M subsets, $S_1, \ldots, S_M$, of fixed sizes $n_1, \ldots, n_M$, respectively, consider $\mathbf{Q}$ to have the partitioned form

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} & \cdots & \mathbf{Q}_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{M1} & \mathbf{Q}_{M2} & \cdots & \mathbf{Q}_{MM} \end{pmatrix},$$

where the (off-diagonal) entries in each submatrix $\mathbf{Q}_{mm'}$ of size $n_m \times n_{m'}$, are all equal to a constant $q_{mm'}$, $1 \leq m, m' \leq M$. Again, depending on how these latter values are defined, a variety of different indices can be generated that now measure the salience of the partition generated by a permutation $\rho(\cdot)$. For a symmetric $\mathbf{P}$, three of the most popular alternatives are noted below that differ only in how the weights q_{mm}, $1 \leq m \leq M$, are defined, and which all assume $q_{mm'} = 0$ for $m \neq m'$:

a) $q_{mm} = 1$: each subset in a partition contributes in direct proportion to the number of object pairs it contains;
b) $q_{mm} = 1/(n_m(n_m - 1))$: each subset contributes equally irrespective of the number of objects (or object pairs) it contains;
c) $q_{mm'} = 1/n_m$: each subset contributes in direct proportion to the number of objects it contains.

In a confirmatory comparison context, the partition statistic $\Gamma(\rho)$ with weighting option c) can be used to construct a test-statistic equivalent to the common F-ratio in a *one-way analysis of variance* for assessing whether mean differences exist over K independent groups. Explicitly, suppose observations are available on n objects, $x_1, \ldots, x_n$, with the first n_1 belonging to group 1, the second n_2 belonging to group 2, and so on. If proximity is defined as $\mathbf{P} = \{p_{ij}\}$, where $p_{ij} = (x_i - x_j)^2$, then the weights in c) produce $\Gamma(\rho_o)$, for ρ_o the identity permutation, equal to twice the within group sum of squares. The distribution of $\Gamma(\rho)$ over all $n!$ permutations generates a distribution over all $n!/(n_1! \ldots n_M!)$ equally-likely ways the n observations can be grouped into subsets of sizes $n_1, \ldots, n_M$, and against which the observed index $\Gamma(\rho_o)$ can be compared. (See [9] for a more thorough discussion of thus evaluating a priori classifications.)

For a skew-symmetric $\mathbf{P}$, the partitioning of S would now be into M ordered subsets, $S_1 \prec \ldots \prec S_M$ of fixed sizes $n_1, \ldots, n_M$, with the most natural weights being $q_{mm} = 0$ for $1 \leq m \leq M$, $q_{mm'} = +1$ if $m < m'$, and $= -1$ if $m > m'$. Maximizing $\Gamma(\rho)$ is this case would be a search for an *ordered partition* in which objects in S_m tend to dominate those in $S_{m'}$ if $m < m'$, i. e., there are generally positive dominance values from a lower-placed subset to one that is higher.

There are several special cases of interest for the partition statistic:

i) for symmetric $\mathbf{P}$ and if for convenience it is assumed n is even and $n_m = 2$ for $1 \leq m \leq M$ (so, $n = 2M$), the weights in a) make $\Gamma(\rho)$ the index for a *matching* of the objects in S induced by $\rho(\cdot)$;
ii) if the proximity matrix $\mathbf{P}$ is itself constructed from a partition of S, then the index $\Gamma(\rho)$ can be interpreted as a measure of association for a *contingency table* defined by the n objects cross-classified using $\rho(\cdot)$ and the two partitions underlying $\mathbf{P}$ and $\mathbf{Q}$.

Depending on the choice of weights for $\mathbf{Q}$, and how proximity is defined in $\mathbf{P}$ based on its underlying partition, a number of well-known indices of association can be obtained: *Pearson's chi-square statistic*, *Goodman–Kruskal's* τ_b, and *Rand's index*. For a more complete discussion of these special cases, including the necessary definitions for $\mathbf{P}$, consult [5].

Partition Hierarchy Statistics

One straightforward strategy for extending QA to identify salient partition hierarchies having a specific form, begins with a given collection of T partitions of S, $\mathcal{P}_1, \ldots, \mathcal{P}_T$, that are hierarchically related. Here, $\mathcal{P}_1$ contains all n objects in n separate classes, $\mathcal{P}_T$ contains all n objects in one class, and $\mathcal{P}_{t+1}$ is formed from $\mathcal{P}_t$ for $t \geq 1$ by uniting one or more of the classes in the latter. If $\mathbf{Q} = \{q_{ij}\}$ is defined by $q_{ij} = \min\{t - 1: O_i, O_j \in \text{common object class in } \mathcal{P}_t\}$, then these latter entries satisfy the defining property of being an *ultrametric*, i. e., $q_{ij} \leq \max\{q_{ik}, q_{kj}\}$ for all $O_i, O_j, O_k \in S$ (see [2,10, Chap. 7] for an extensive discussion of ultrametrics). For symmetric $\mathbf{P}$, the optimization of $\Gamma(\rho)$ in (1) would be the search for a salient partition hierarchy having the generic form

defined by $\mathcal{P}_1, \ldots, \mathcal{P}_T$, and which optimizes the cross-product between the proximity information in **P** and the levels at which the object pairs are first placed into common classes in the hierarchy. It might be noted that both single clusters and partitions could be considered special cases of a partition hierarchy when $T = 3$ and the only nontrivial partition is $\mathcal{P}_2$, i. e., to obtain a single cluster, $\mathcal{P}_2$ can be defined by one subset of size K and $n - K$ subsets each of size one; to obtain a single partition, $\mathcal{P}_2$ merely has to be that partition with the desired number of classes and class sizes.

Alternative Assignment Indices

There are a variety of alternatives for replacing the cross-product in the QA index in (1) by a different function between the entries in **P** and **Q**. Depending on how the proximity information in **P** and the target given by **Q** are specified, one might adopt, for example, the sum of absolute differences, $\sum_{i,j} |p_{\rho(i)\rho(j)} - q_{ij}|$, or the sum of dichotomous indicators for equality, $\sum_{i,j} g(p_{\rho(i)\rho(j)}, q_{ij})$, where $g(x, y) = 1$ if $x = y$ and 0 otherwise, or even use *'bottleneck' measures* such as $\min_{i,j} p_{\rho(i)\rho(j)} q_{ij}$ or $\max_{i,j} p_{\rho(i)\rho(j)} q_{ij}$. Somewhat more well-developed in the literature than these possibilities (e. g., see [5, Chap. 5]) are generalizations of (1) that would maintain the basic cross-product structure but which would rely on higher-order functions of the entries in **P** and **Q** before the cross-products were taken. Again, variations would be possible, but two of the more obvious forms of extension are given below that depend solely on the order of the entries within **P** and within **Q**:

- *Three-argument functions*: Given **P** and **Q**, and letting $\text{sign}(x) = +1$ if $x > 0$, $= 0$ if $x = 0$, and $= -1$ if $x < 0$, define

$$\mathcal{A}(\rho) = \sum_{\substack{i \neq j \\ i \neq k}} \text{sign}(p_{\rho(i)\rho(j)} - p_{\rho(i)\rho(k)}) \, \text{sign}(q_{ij} - q_{ik}).$$

  The index $\mathcal{A}(\rho)$ can be interpreted as the difference between two counts, say $\mathcal{A}^+(\rho)$ and $\mathcal{A}^-(\rho)$, where $\mathcal{A}^+(\rho)$ (respectively, $\mathcal{A}^-(\rho)$) is the number of consistencies (inconsistencies) in the ordering of pairs of off-diagonal entries in $\{p_{\rho(i)\rho(j)}\}$ and their counterparts in $\{q_{ij}\}$, where the former pairs share a common (row) object $O_{\rho(i)}$.

- *Four-argument functions*: Define

$$\mathcal{B}(\rho) = \sum_{\substack{i \neq j \\ k \neq l}} \text{sign}(p_{\rho(i)\rho(j)} - p_{\rho(k)\rho(l)}) \, \text{sign}(q_{ij} - q_{kl}).$$

  Again, the index $\mathcal{B}(\rho)$ can be viewed as the difference between $\mathcal{B}^+(\rho)$ and $\mathcal{B}^-(\rho)$, where $\mathcal{B}^+(\rho)$ (respectively, $\mathcal{B}^-(\rho)$) is the number of consistencies (inconsistencies) in the ordering of pairs of off-diagonal entries in $\{p_{\rho(i)\rho(j)}\}$ and their counterparts in $\{q_{ij}\}$. In contrast to $\mathcal{A}(\rho)$, however, no common object need be present in the pairs of off-diagonal entries. The distinction between $\mathcal{A}(\rho)$ and $\mathcal{B}(\rho)$ in measuring the correspondence between **P** and **Q** rests on whether the proximity entries in **P** are strictly comparable only within rows (i. e., to what are called *row conditional proximity data*, e. g., see [1, p. 192]) or whether such comparisons make sense when performed across rows.

To illustrate the interpretation of $\mathcal{A}(\rho)$ and $\mathcal{B}(\rho)$ in the single cluster statistic context, suppose **Q** has the weight structure in A) that generated through (1) the measure of cluster compactness as the average within group proximity in $S_\rho = \{O_{\rho(1)}, \ldots, O_{\rho(K)}\}$. In using this specific target **Q** for $\mathcal{A}(\rho)$, the index is, in words, twice the difference between the number of instances in which a proximity for two objects both within S_ρ is greater than the proximity from one of these two objects to another in $S - S_\rho$, and the number of instances in which it is less. Depending on whether proximity is keyed as a similarity or a dissimilarity, a compact subset would be one for which $\mathcal{A}(\rho)$ is maximized or minimized, respectively. If instead, the weight structure for **Q** given in B) that defined the measure of cluster isolation, the index $\mathcal{A}(\rho)$ would now be twice the difference between the number of instances in which a proximity between two objects that span S_ρ and $S - S_\rho$ is greater than the proximity between two objects within S_ρ or within $S - S_\rho$ (where the latter have one member in common with the two that span S_ρ and $S - S_\rho$), and the number of instances in which it is less. Now, an isolated subset would be identified by maximizing or minimizing $\mathcal{A}(\rho)$ depending on the keying of proximity as a dissimilarity or similarity, respectively. For $\mathcal{B}(\rho)$, and the weight structure in A), the index is, in words, twice the difference between the number of instances in which a proximity for two objects both within S_ρ is greater than the prox-

imity between *any* two objects that span S_ρ and $S - S_\rho$ and the number of instances in which it is less. The index $\mathcal{B}(\rho)$ for the weight matrix in B) would be twice the difference between the number of instances in which a proximity between two objects that span S_ρ and $S - S_\rho$ is greater than the proximity between *any* two objects within S_ρ or within $S - S_\rho$.

In the partition context, a similar interpretation to the use of the single subset compactness measure would be present for $\mathcal{A}(\rho)$ and $\mathcal{B}(\rho)$ and for all of the three weighting options mentioned, but now all aggregated over the M subsets of the partition. In the partition hierarchy framework, the correspondence between $\{p_{\rho(i)\rho(j)}\}$ and $\mathbf{Q}$ is measured by the degree of consistency in the ordering of the object pairs by proximity and the ordering of the object pairs by the levels in which the objects are first placed into a common class.

In addition to replacing the QA index in (1) by the higher order functions adopted in $\mathcal{A}(\rho)$ and $\mathcal{B}(\rho)$ to effect a reliance only on the order properties of the entries within $\mathbf{P}$ and $\mathbf{Q}$, there are several other uses in a clustering/classification context for the definition of three- or four-argument functions. One alternative will be mentioned here that deals with what can be called the *generalized single cluster statistic*. Explicitly, suppose three- and four-argument function of the entries in $\mathbf{P}$ are denoted by $u(\cdot, \cdot, \cdot)$ and $r(\cdot, \cdot, \cdot, \cdot)$, respectively, and those in $\mathbf{Q}$ by $v(\cdot, \cdot, \cdot)$ and $s(\cdot, \cdot, \cdot, \cdot)$, and consider the general cross-product forms of

$$C(\rho) = \sum_{i,j,k} u(\rho(i), \rho(j), \rho(k)) v(i, j, k),$$

$$\mathcal{D}(\rho) = \sum_{i,j,k,l} r(\rho(i), \rho(j), \rho(k), \rho(l)) s(i, j, k, l).$$

It will be assumed here that both $v(\cdot, \cdot, \cdot)$ and $s(\cdot, \cdot, \cdot, \cdot)$ are merely indicator functions for a subset of size K, so $v(i, j, k) = 1$ if $1 \leq i, j, k \leq K$, and $= 0$ otherwise; $s(i, j, k, l) = 1$ if $1 \leq i, j, k, l \leq K$, and $= 0$ otherwise. Thus, the optimization of $C(\rho)$ or $\mathcal{D}(\rho)$ can be viewed as the search for a subset of size K with extreme values for the indices $\sum_{1 \leq i,j,k \leq K} u(\rho(i), \rho(j), \rho(k))$ or $\sum_{1 \leq i,j,k,l \leq K} r(\rho(i), \rho(j), \rho(k), \rho(l))$, and depending on how the functions $u(\cdot, \cdot, \cdot)$ and $r(\cdot, \cdot, \cdot, \cdot)$ are defined, a subset that is very salient with respect to the property that characterizes the latter.

A number of properties that may be desirable to optimize in a subset of size K have been considered (see [4] for a more complete discussion), of which the two listed below are directly relevant to the clustering/classification context:

i) a proximity matrix (with a dissimilarity keying) represents a perfect partition hierarchy if it satisfies the property of being an ultrametric: for all $1 \leq i, j, k \leq n$, $p_{ij} \leq \max\{p_{ik}, p_{kj}\}$, or equivalently, the two largest values among p_{ij}, p_{ik}, and p_{kj} are equal. Thus, if $u(\rho(i), \rho(j), \rho(k))$ equals the absolute difference between the two largest values among $p_{\rho(i)\rho(j)}$, $p_{\rho(i)\rho(k)}$, and $p_{\rho(j)\rho(k)}$, the minimization of $C(\rho)$ seeks a subset of size K that is as close to being an ultrametric as possible (as measured by $C(\rho)$);
ii) a proximity matrix (again, with a dissimilarity keying) represents a perfect *additive tree* where proximities can be reconstructed by minimum path lengths in a tree if they satisfy the four-point property: for all $1 \leq i, j, k, l \leq n$, $p_{ij} + p_{kl} \leq \max\{p_{ik} + p_{jl}, p_{il} + p_{jk}\}$, or equivalently, the largest two sums among $p_{ij} + p_{kl}$, $p_{ik} + p_{jl}$, and $p_{il} + p_{jk}$ are equal. Thus, if $r(\rho(i), \rho(j), \rho(k), \rho(l))$ equals the absolute difference between the two largest values among $p_{\rho(i)\rho(j)} + p_{\rho(k)\rho(l)}$, $p_{\rho(i)\rho(k)} + p_{\rho(j)\rho(l)}$, and $p_{\rho(i)\rho(l)} + p_{\rho(j)\rho(k)}$, the minimization of $\mathcal{D}(\rho)$ seeks a subset of size K that is as close to satisfying the four-point condition as possible (as measured by $\mathcal{D}(\rho)$).

Modifications of the Target Matrix Q

The optimization of an assignment index such as (1) assumes that the target matrix $\mathbf{Q}$ is fixed and given a priori. Based on this invariance, maximizing (1), for example, could be equivalently stated as the minimization of

$$\sum_{i,j} (p_{\rho(i)\rho(j)} - q_{ij})^2. \tag{2}$$

There has been a substantial recent literature (e. g., [6,7,8]) where not only is an optimal permutation, say $\rho^*(\cdot)$, sought that would minimize (2), but in which a specific target matrix $\mathbf{Q}$ is also constructed based on a collection of (linear inequality) constraints that would characterize some type of classificatory structure fitted to $\{p_{\rho(i)\rho(j)}\}$. The constraints imposed on $\mathbf{Q}$ are possibly based on the (sought for) permutation $\rho^*(\cdot)$.

In minimizing (2) but allowing the target matrix $\mathbf{Q}$ to itself be estimated, a typical iterative process would proceed as follows: on the basis of an initial target ma-

trix $\mathbf{Q}^{(0)}$, find a permutation, say $\rho^{(1)}(\cdot)$, to maximize the cross-product in (1). Using $\rho^{(1)}(\cdot)$, fit a target matrix $\mathbf{Q}^{(1)}$ to $\{p_{\rho^{(1)}(i)\rho^{(1)}(j)}\}$ minimizing (2). Continue the process for $\rho^{(t)}$ and $\mathbf{Q}^{(t)}$ for $t > 1$ until convergence. A variety of constraints for $\mathbf{Q}$ have been considered. Among these, there are

i) a sum of matrices each having what are called anti-Robinson forms (i. e., a matrix is *anti-Robinson* if within each row and column, the entries never decrease moving in any direction away from the main diagonal [6]);
ii) a sum of ultrametric matrices (characterized by the ultrametric condition given earlier [7]);
iii) a sum of additive tree matrices (again, as characterized by the four-point condition given earlier [7]);
iv) *unidimensional scales* (i. e., a matrix is a *linear unidimensional scale* if its entries can be given by $\{|x_j - x_i| + c\}$, where the estimated coordinates are $x_1 \leq \cdots \leq x_n$ and c is an estimated constant [8]); and
v) *circular unidimensional scales* (i. e., a matrix is so characterized if it can be represented as $\{\min\{|x_j - x_i|, x_0 - |x_j - x_i|\} + c\}$, where $x_1 \leq \cdots \leq x_n$, x_0 is the circumference of the circular structure, and c is an estimated constant [8]).

See also

- ▶ Assignment and Matching
- ▶ Bi-Objective Assignment Problem
- ▶ Communication Network Assignment Problem
- ▶ Frequency Assignment Problem
- ▶ Maximum Partition Matching
- ▶ Quadratic Assignment Problem

References

1. Carroll JD, Arabie P (1998) Multidimensional scaling. In: Birnbaum MH (ed) Measurement, judgement, and decision making. Handbook Perception and Cognition. Acad Press, New York, pp 179–250
2. De Soete G, Carroll JD (1996) Tree and other network models for representing proximity data. In: Arabie P, Hubert LJ, De Soete G (eds) Clustering and classification. World Sci, Singapore, pp 157–198
3. Gibbons JD (1971) Nonparametric statistical inference. McGraw-Hill, New York
4. Hubert LJ (1980) Analyzing proximity matrices: The assessment of internal variation in combinatorial structure. J Math Psych 21:247–264
5. Hubert LJ (1987) Assignment methods in combinatorial data analysis. M. Dekker, New York
6. Hubert LJ, Arabie P (1994) The analysis of proximity matrices through sums of matrices having (anti-)Robinson forms. British J Math Statist Psych 47:1–40
7. Hubert LJ, Arabie P (1995) Iterative projection strategies for the least-squares fitting of tree structures to proximity data. British J Math Statist Psych 48:281–317
8. Hubert LJ, Arabie P, Meulman J (1997) Linear and circular unidimensional scaling for symmetric proximity matrices. British J Math Statist Psych 50:253–284
9. Mielke PW, Berry KJ, Johnson ES (1976) Multi-response permutation procedures for a priori classifications. Comm Statist A5:1409–1424
10. Mirkin B (1996) Mathematical classification and clustering. Kluwer, Dordrecht
11. Pardalos PM, Wolkowicz H (eds) (1994) Quadratic assignment and related problems. DIMACS, Amer. Math. Soc., Providence, RI

Asymptotic Properties of Random Multidimensional Assignment Problem

PAVLO A. KROKHMAL
Department of Mechanical and Industrial Engineering, The University of Iowa, Iowa City, USA

MSC2000: 90C27, 34E05

Article Outline

Keywords and Phrases
Introduction
Expected Optimal Value of Random MAP
Expected Number of Local Minima in Random MAP
  Local Minima and p-exchange Neighborhoods in MAP
  Expected Number of Local Minima in MAP with $n = 2$
  Expected Number of Local Minima in a Random MAP with Normally Distributed Costs
Conclusions
References

Keywords and Phrases

Multidimensional assignment problem; Random assignment problem; Expected optimal value; Asymptotical analysis; Convergence bounds

Introduction

The Multidimensional Assignment Problem (MAP) is a higher dimensional version of the two-dimensional, or Linear Assignment Problem (LAP) [24]. If a classical textbook formulation of the Linear Assignment Problem is to find an optimal assignment of "N jobs to M workers", then, for example, the 3-dimensional Assignment Problem can be interpreted as finding an optimal assignment of "N jobs to M workers in K time slots", etc. In general, the objective of the MAP is to find tuples of elements from given sets, such that the total cost of the tuples is minimized. The MAP was first introduced by Pierskalla [26], and since then has found numerous applications in the areas of data association [4], image recognition [31], multisensor multitarget tracking [18,27], tracking of elementary particles [28], etc. For a discussion of the MAP and its applications see, for example, [7] and references therein.

Without loss of generality, a d-dimensional axial MAP can be written in a form where each dimension has the same number n of elements, i. e.,

$$\min_{x\in\{0,1\}^{n^d}} \left\{ \sum_{\substack{i_k\in\{1,\dots,n\}\\ k\in\{1,\dots,d\}}} c_{i_1\cdots i_d} x_{i_1\cdots i_d} \,\middle|\, \sum_{\substack{i_k\in\{1,\dots,n\}\\ k\in\{1,\dots,d\}\setminus j}} x_{i_1\cdots i_d} = 1, \; i_j = 1,\dots,n,\; j=1,\dots,d \right\} . \tag{1}$$

An instance of the MAP with different numbers of elements in each dimension, $n_1 \geq n_2 \geq \cdots \geq n_d$, is reducible to form (1) by introduction of dummy variables.

Problem (1) admits the following geometric interpretation: given a d-dimensional cubic matrix, find such a permutation of its rows and columns that the sum of the diagonal elements is minimized (which explains the term "axial"). This rendition leads to an alternative formulation of the MAP (1) in terms of permutations $\pi_1,\dots,\pi_{d-1}$ of numbers 1 to n, i. e., one-to-one mappings $\pi_i : \{1,\dots,n\} \mapsto \{1,\dots,n\}$,

$$\min_{\pi_1,\dots,\pi_{d-1}\in\Pi^n} \sum_{i=1}^{n} c_{i,\pi_1(i),\dots,\pi_{d-1}(i)} ,$$

where Π^n is the set of all permutations of the set $\{1,\dots,n\}$. A feasible solution to the MAP (1) can be conveniently described by specifying its cost,

$$z = c_{i_1^{(1)}\cdots i_d^{(1)}} + c_{i_1^{(2)}\cdots i_d^{(2)}} + \cdots + c_{i_1^{(n)}\cdots i_d^{(n)}} , \tag{2}$$

where $\left(i_j^{(1)}, i_j^{(2)}, \dots, i_j^{(n)}\right)$ is a permutation of the set $\{1,2,\dots,n\}$ for every $j = 1,\dots,d$. In contrast to the LAP that represents a $d = 2$ special case of the MAP (1) and is polynomially solvable [7], the MAP with $d \geq 3$ is generally NP-hard, a fact that follows from reduction of the 3-dimensional matching problem (3DM) [8].

Despite its inherent difficulty, several exact and heuristic algorithms [1,6,11,25] have been proposed to this problem. Most of these algorithms rely, at least partly, on repeated local searches in neighborhoods of feasible solutions, which brings about the question of how the number of local minima in a MAP impact these solution algorithms. Intuitively, if the number of local minima is small then one may expect better performance from meta-heuristic algorithms that rely on local neighborhood searches. A solution landscape is considered to be rugged if the number of local minima is exponential with respect to the dimensions of the problem [21]. Evidence in [5] showed that ruggedness of the solution landscape has a direct impact on the effectiveness of the simulated annealing heuristic in solving at least one other hard problem, the quadratic assignment problem. Thus, one of the issues that we address below is estimation of the expected number $\mathsf{E}[M]$ of local minima in random MAPs with respect to different local neighborhoods.

Another problem that we discuss is the behavior of the expected optimal value $Z^*_{d,n}$ of random large-scale MAPs, whose assignment costs are assumed to be independent identically distributed (iid) random variables from a given continuous distribution.

During the last two decades, expected optimal values of random assignment problems have been studied intensively in the context of random LAP. Perhaps, the most widely known result in this area is the conjecture by Mézard and Parisi [17] that the expected optimal value $\mathsf{E}[L_n] := Z^*_{2,n}$ of a LAP of size n with iid uniform or exponential with mean 1 cost coefficients satisfies $\lim_{n\to\infty} \mathsf{E}[L_n] = \frac{\pi^2}{6}$. In fact, this conjecture was preceded by an upper bound on the expected optimal value of the LAP with uniform (0,1) costs: $\limsup_{n\to\infty} L_n \leq 3$ due to

Walkup [32], which was soon improved by Karp [12]: $\limsup_{n\to\infty} L_n \le 2$. A lower bound on the limiting value of L_n was first provided by Lazarus [14]: $\liminf_{n\to\infty} L_n \ge 1 + e^{-1} \approx 1.37$, and then has been improved to 1.44 by Goemans and Kodialam [9] and 1.51 by Olin [20]. Experimental evidence in support of the Mézard-Parisi conjecture was provided by Pardalos and Ramakrishnan [22]. Recently, Aldous [2] has shown that indeed $\lim_{n\to\infty} \mathsf{E}[L_n] = \frac{\pi^2}{6}$, thereby proving the conjecture. Another conjecture due to Parisi [23] stating that the expected optimal value of a random LAP of finite size n with exponentially distributed iid costs is equal to $\mathsf{E}[L_n] = Z^*_{2,n} = \sum_{i=1}^{n} i^{-2}$ has been proven independently in [16] and [19].

Our work contributes to the existing literature on random assignment problems by establishing the limiting value and asymptotic behavior of the expected optimal cost $Z^*_{d,n}$ of random MAP with iid cost coefficients for a broad class of continuous distributions. The presented approach is constructive in the sense that it allows for deriving converging asymptotical lower and upper bounds for $Z^*_{d,n}$, as well as for estimating the rate of convergence for $Z^*_{d,n}$ in special cases.

Expected Optimal Value of Random MAP

Our approach to determining the asymptotic behavior of the expected optimal cost $Z^*_{d,n}$ of an MAP (1) with random cost coefficients involves analysis of the so-called *index tree*, a graph structure that represents the set of feasible solutions of the MAP. First introduced by Pierskalla [26], the index tree graph $\mathcal{G} = (V, E)$ of the MAP (1) has a set of vertices V which is partitioned into n levels[1] and a distinct *root node*. A node at level j of the graph represents an assignment $(i_1, \ldots, i_d)$ with $i_1 = j$ and cost $c_{ji_2\cdots i_d}$, whereby each level contains $\kappa = n^{d-1}$ nodes. The set E of arcs in the index tree graph is constructed in such a way that any feasible solution of the MAP (1) can be represented as a path connecting the root node to a leaf node at level n (such a path is called a *feasible* path); evidently, the index tree contains $n!^{d-1}$ feasible paths, by the number of feasible solutions of the MAP (1).

The index tree representation of MAP aids in construction of lower and upper bounds for the expected optimal cost of MAP (1) with random iid costs via the following lemmata [10].

Lemma 1. *Given the index tree graph $\mathcal{G} = (V, E)$ of $d \ge 3$, $n \ge 3$ MAP whose assignment costs are iid random variables from an absolutely continuous distribution, construct set $\mathcal{A} \subset V$ by randomly selecting α different nodes from each level of the index tree. Then, $\mathcal{A}$ is expected to contain a feasible solution of the MAP if*

$$\alpha = \left\lceil \frac{n^{d-1}}{n!^{\frac{d-1}{n}}} \right\rceil . \tag{3}$$

Lemma 2. *For a $d \ge 3$, $n \ge 3$ MAP whose cost coefficients are iid random variables from an absolutely continuous distribution F with existing first moment, define*

$$\underline{Z}^*_{d,n} := n\mathsf{E}_F\left[X_{(1|\kappa)}\right] \quad \text{and} \quad \overline{Z}^*_{d,n} := n\mathsf{E}_F\left[X_{(\alpha|\kappa)}\right] , \tag{4}$$

where $X_{(i|\kappa)}$ is the ith order statistic of $\kappa = n^{d-1}$ iid random variables with distribution F, and parameter α is determined as in (3). Then, $\underline{Z}^*_{d,n}$ and $\overline{Z}^*_{d,n}$ constitute lower and upper bounds for the expected optimal cost $Z^*_{d,n}$ of the MAP, respectively: $\underline{Z}^*_{d,n} \le Z^*_{d,n} \le \overline{Z}^*_{d,n}$.

Proofs of the lemmas are based on the probabilistic method [3] and can be found in [10]. In particular, the proof of Lemma 2 considers a set $\mathcal{A}_{\min}$ that is constructed by selecting from each level of the index tree α nodes with the smallest costs among the κ nodes at that level. The continuity of distribution F ensures that assignment costs in the MAP (1) are all different almost surely, hence locations of the nodes that comprise the set $\mathcal{A}_{\min}$ are random with respect to the array of nodes in each level of $\mathcal{G}(V, E)$. In the remainder of the paper, we always refer to α and κ as defined above.

By definition, the parameter $\kappa = n^{d-1}$ approaches infinity whenever n or d does; this allows us to denote the corresponding cases by $\kappa \xrightarrow{n} \infty$ and $\kappa \xrightarrow{d} \infty$, respectively. If certain statement holds for both cases of $n \to \infty$ and $d \to \infty$, we indicate this by $\kappa \xrightarrow{n,d} \infty$. The behavior of quantity α (3) when n or d increases is more contrasting. In the case $n \to \infty$ it approaches a finite limiting value,

$$\alpha \to \alpha^* := \lceil e^{d-1} \rceil, \quad \kappa \xrightarrow{n} \infty, \tag{5}$$

[1] In the general case of MAP with n_i elements in dimension $i = 1, \ldots, d$, the index graph would contain n_1 levels.

while in the case of fixed n and unbounded d it increases exponentially:

$$\alpha \sim \kappa^{\gamma_n}, \quad \kappa \xrightarrow{d} \infty, \quad \text{where} \quad \gamma_n = 1 - \frac{\ln n!}{n \ln n}\,, \tag{6}$$

and it is important to observe that $0 < \gamma_n < \frac{1}{2}$ for $n \geq 3$ [13].

The presented lemmata addresses MAPs with $d \geq 3, n \geq 3$. The case $d = 2$ represents, as noted earlier, the Linear Assignment Problem, whose asymptotic behavior is distinctly different from that of MAPs with $d \geq 3$. It can be shown that in the case of $d = 2$ Lemmas 1 and 2 produce only trivial bounds that are rather inefficient in determining the asymptotic behavior of the expected optimal value of the LAP within the presented approach. In the case $n = 2$ the costs of feasible solutions to the MAP (1) have the form

$$z = c_{i_1^{(1)} \cdots i_d^{(1)}} + c_{i_1^{(2)} \cdots i_d^{(2)}}, \quad \text{where} \quad i_j^{(1)}, i_j^{(2)} \in \{1, 2\}, i_j^{(1)} \neq i_j^{(2)}\,,$$

and consequently are iid random variables with distribution F_2, which is the convolution of F with itself: $F_2 = F * F$ [11]. This fact allows for computing the expected optimal value of $n = 2$ MAP exactly, without resorting to bounds (4):

$$Z^*_{d,2} = \mathsf{E}_{F*F}\left[X_{(1|2!^{d-1})}\right]. \tag{7}$$

In the general case $d \geq 3, n \geq 3$ the main challenge is constituted by computation of the upper bound $\overline{Z}^*_{d,n} = n\mathsf{E}_F\left[X_{(\alpha|\kappa)}\right]$, where $X_{(\alpha|\kappa)}$ is the α-th order statistic among κ independent F-distributed random variables. The subsequent analysis relies on representation of $\overline{Z}^*_{d,n}$ in the form

$$\overline{Z}^*_{d,n} = \frac{n\Gamma(\kappa+1)}{\Gamma(\alpha)\Gamma(\kappa-\alpha+1)} \cdot \int_0^1 F^{-1}(u) u^{\alpha-1}(1-u)^{\kappa-\alpha}\,\mathrm{d}u\,, \tag{8}$$

where F^{-1} denotes the inverse of the c.d.f. F of the the distribution of assignment costs in MAP (1). While it is practically impossible to evaluate the integral in (8) exactly in the general case, its asymptotic behavior for large n and d can be determined for a wide range of distributions F. For instance, in the case when distribution F has a finite left endpoint of its support set, the asymptotic behavior of the integral in (8) is obtained by means of the following

Lemma 3. *Let function $h(u)$ have the following asymptotic expansion at $0+$,*

$$h(u) \sim \sum_{s=0}^{\infty} a_s u^{(s+\lambda-\mu)/\mu}, \quad u \to 0+\,, \tag{9}$$

where $\lambda, \mu > 0$. Then for any positive integer m one has

$$\int_0^1 h(u) u^{\alpha-1}(1-u)^{\kappa-\alpha}\,\mathrm{d}u = \sum_{s=0}^{m-1} a_s \phi_s(\kappa) + \mathcal{O}\big(\phi_m(\kappa)\big), \quad \kappa \xrightarrow{n,d} \infty, \tag{10}$$

where $\phi_s(\kappa) = \mathrm{B}\big(\frac{s+\lambda}{\mu} + \alpha - 1,\ \kappa - \alpha + 1\big)$, $s = 0, 1, \ldots,$ provided that the integral is absolutely convergent for $\kappa = \alpha = 1$.

Above, $\mathrm{B}(x, y)$ is the Beta function. Using similar results for the cases when the support set of distribution F is unbounded from below, we obtain that the limiting behavior of the expected optimal value $Z^*_{d,n}$ of random MAP is determined by the location of the left endpoint of the support of F [13].

Theorem 1. Expected Optimal Value of Random MAP *Consider a $d \geq 3, n \geq 2$ MAP (1) with cost coefficients that are iid random variables from an absolutely continuous distribution F with existing first moment. If the distribution F satisfies either of the following conditions,*

1. $F^{-1}(u) = F^{-1}(0+) + \mathcal{O}(u^{\beta}), u \to 0+, \beta > 0$
2. $F^{-1}(u) \sim -\nu u^{-\beta_1}\big(\ln \frac{1}{u}\big)^{\beta_2}, \quad u \to 0+, 0 \leq \beta_1 < 1, \beta_2 \geq 0, \beta_1 + \beta_2 > 0, \nu > 0$

where $F^{-1}(0+) = \lim_{u \to 0+} F^{-1}(u)$, the expected optimal value of the MAP satisfies

$$\lim Z^*_{d,n} = \lim nF^{-1}(0+)\,,$$

where both limits are taken at either $n \to \infty$ or $d \to \infty$.

The obtained results can be readily employed to construct upper and lower asymptotical bounds for the expected optimal value of MAP when one of the parameters n or d is large but finite. The following statement follows directly from Lemma 3 and Theorem 1.

Corollary 1. *Consider a $d \geq 3, n \geq 3$ MAP (1) with cost coefficients that are iid random variables from an absolutely continuous distribution with existing first moment. Let $a \in \mathbb{R}$ be the left endpoint of the support set of this distribution, $a = F^{-1}(0+)$, and assume that the inverse $F^{-1}(u)$ of the c.d.f. $F(u)$ of the distribution is such that*

$$F^{-1}(u) \sim a + \sum_{s=1}^{\infty} a_s u^{s/\mu}, \quad u \to 0+, \mu > 0\,. \tag{11}$$

Then, for any integer $m \geq 1$, lower and upper bounds $\underline{Z}^*_{d,n}$, $\overline{Z}^*_{d,n}$ (4) on the expected optimal cost $Z^*_{d,n}$ of the MAP can be asymptotically evaluated as

$$\underline{Z}^*_{d,n} = an + \sum_{s=1}^{m-1} a_s \frac{n\Gamma(\kappa+1)\Gamma\left(\frac{s}{\mu}+1\right)}{\Gamma\left(\kappa+\frac{s}{\mu}+1\right)} + \mathcal{O}\left(n\frac{\Gamma(\kappa+1)\Gamma\left(\frac{m}{\mu}+1\right)}{\Gamma\left(\kappa+\frac{m}{\mu}+1\right)}\right), \kappa \xrightarrow{n,d} \infty\,, \tag{12a}$$

$$\overline{Z}^*_{d,n} = an + \sum_{s=1}^{m-1} a_s \frac{n\Gamma(\kappa+1)\Gamma\left(\frac{s}{\mu}+\alpha\right)}{\Gamma(\alpha)\Gamma\left(\kappa+\frac{s}{\mu}+1\right)} + \mathcal{O}\left(n\frac{\Gamma(\kappa+1)\Gamma\left(\frac{m}{\mu}+\alpha\right)}{\Gamma(\alpha)\Gamma\left(\kappa+\frac{m}{\mu}+1\right)}\right), \kappa \xrightarrow{n,d} \infty\,. \tag{12b}$$

It can be shown that the lower and upper bounds defined by (12a, 12b) are convergent, i. e., $|\overline{Z}^*_{d,n} - \underline{Z}^*_{d,n}| \to 0$, $\kappa \xrightarrow{n,d} \infty$, whereas the corresponding asymptotical bounds for the case of distributions with support unbounded from below may be divergent in the sense that $|\overline{Z}^*_{d,n} - \underline{Z}^*_{d,n}| \nrightarrow 0$ when $\kappa \xrightarrow{n,d} \infty$.

The asymptotical representations (12a, 12b) for the bounds $\underline{Z}^*_{d,n}$ and $\overline{Z}^*_{d,n}$ are simplified when the inverse F^{-1} of the c.d.f. of the distribution has a regular power series expansion in the vicinity of zero. Assume, for example, that function F^{-1} can be written as

$$F^{-1}(u) = a_1 u + \mathcal{O}(u^2), \quad u \to 0+\,. \tag{13}$$

It is then easy to see that for $n \gg 1$ and d fixed the expected optimal value of the MAP is asymptotically bounded as

$$\frac{a_1}{n^{d-2}} + \mathcal{O}\left(\frac{1}{n^{d-1}}\right) \leq Z^*_{d,n} \leq \frac{a_1\lceil e^{d-1}\rceil}{n^{d-2}} + \mathcal{O}\left(\frac{1}{n^{d-1}}\right), \quad n \to \infty\,, \tag{14}$$

which immediately yields the rate of convergence to zero for $Z^*_{d,n}$ as n approaches infinity:

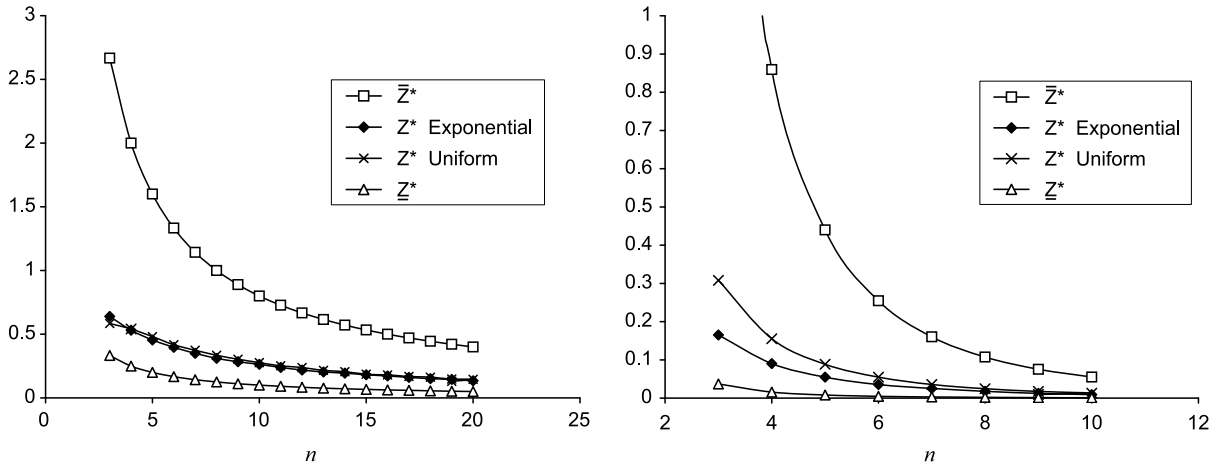
Corollary 2. *Consider a $d \geq 3, n \geq 3$ MAP (1) with cost coefficients that are iid random variables from an absolutely continuous distribution with existing first moment. Let the inverse F^{-1} of the c.d.f. of the distribution satisfy (13). Then, for a fixed d and $n \to \infty$ the expected optimal value $Z^*_{d,n}$ of the MAP converges to zero as $\mathcal{O}\left(n^{-(d-2)}\right)$.*

For example, the expected optimal value of 3-dimensional ($d = 3$) MAP with uniform $U(0,1)$ or exponential distributions converges to zero as $\mathcal{O}(n^{-1})$ when $n \to \infty$.

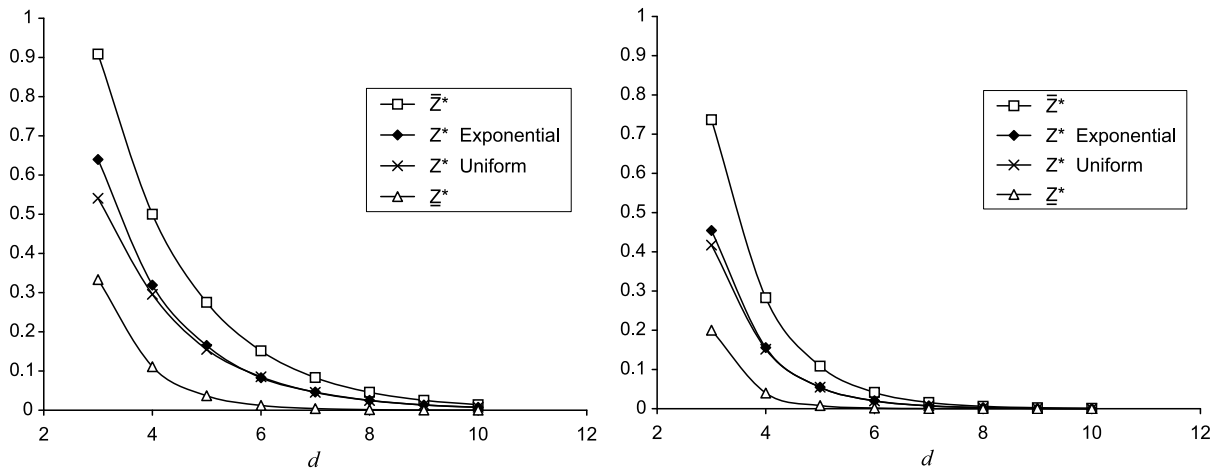
We illustrate the tightness of the developed bounds (12a, 12b) by comparing them to the computed expected optimal values of MAPs with coefficients $c_{i_1 \cdots i_d}$ drawn from the uniform $U(0,1)$ distribution and exponential distribution with mean 1. It is elementary that the inverse functions $F^{-1}(\cdot)$ of the c.d.f.'s for both these distributions are representable in form (13) with $a_1 = 1$.

The numerical experiments involved solving multiple instances of randomly generated MAPs with the number of dimensions d ranging from 3 to 10, and the number n of elements in each dimension running from 3 to 20. The number of instances generated for estimation of the expected optimal value of the MAP with a given distribution of cost coefficients varied from 1000 (for smaller values of d and n) to 50 (for problems with largest n and d).

To solve the problems to optimality, we used a branch-and-bound algorithm that navigated through the index tree representation of the MAP. Figures 1 and 2 display the obtained expected optimal values of MAP with uniform and exponential iid cost coefficients when d is fixed at $d = 3$ or 5 and $n = 3, \ldots, 20$, and when $n = 3$ or 5 and d runs from 3 to 10. This "asymmetry" in reporting of the results is explained by



Asymptotic Properties of Random Multidimensional Assignment Problem, Figure 1
Expected optimal value $Z^*_{d,n}$, lower and upper bounds $\underline{Z}^*_{d,n}$, $\overline{Z}^*_{d,n}$ of an MAP with fixed $d = 3$ (*left*) and $d = 5$ (*right*) for uniform $U(0, 1)$ and exponential (1) distributions



Asymptotic Properties of Random Multidimensional Assignment Problem, Figure 2
Expected optimal value $Z^*_{d,n}$, lower and upper bounds $\underline{Z}^*_{d,n}$, $\overline{Z}^*_{d,n}$ of an MAP with fixed $n = 3$ (*left*) and $n = 5$ (*right*) for uniform $U(0, 1)$ and exponential(1) distributions

the fact that the implemented branch-and-bound algorithm based on index tree is more efficient in solving "shallow" MAPs, i. e., instances that have larger n and smaller d. The solution times varied from several seconds to 20 hours on a 2GHz PC.

The conducted numerical experiments suggest that the constructed lower and upper bounds for the expected optimal cost of random MAPs are quite tight, with the upper bound $\overline{Z}^*_{d,n}$ being tighter for the case of fixed n and large d (see Figs. 1, 2).

Expected Number of Local Minima in Random MAP

Local Minima and p-exchange Neighborhoods in MAP

As it has been mentioned in the Introduction, we consider local minima of a MAP with respect to a local neighborhood, in the sense of [15]. For any $p = 2, \ldots, n$, we define the p-exchange local neighborhood $\mathcal{N}_p(i)$ of the ith feasible solu-

tion $\{i_1^{(1)}\cdots i_d^{(1)},\ldots,i_1^{(n)}\cdots i_d^{(n)}\}$ of the MAP (1) as the set of solutions obtained from i by permuting p or less elements in one of the dimensions $1,\ldots,d$. More formally, $\mathcal{N}_p(i)$ is the set of n-tuples $\{j_1^{(1)}\cdots j_d^{(1)},\ldots,j_1^{(n)}\cdots j_d^{(n)}\}$ such that $\{j_k^{(1)},\ldots,j_k^{(n)}\}$ is a permutation of $\{1,\ldots,n\}$ for all $1\le k\le d$, and, furthermore, there exists only one $k_0\in\{1,\ldots,d\}$ such that

$$2\le\sum_{r=1}^{n}\bar{\delta}_{i_{k_0}^{(r)}j_{k_0}^{(r)}}\le p,\quad\text{while}\quad\sum_{r=1}^{n}\bar{\delta}_{i_k^{(r)}j_k^{(r)}}=0\quad\text{for all}\quad k\in\{1,\ldots,d\}\backslash k_0\,, \tag{15}$$

where $\bar{\delta}_{ij}$ is the negation of the Kroneker delta, $\bar{\delta}_{ij}=1-\delta_{ij}$. As an example, consider the following feasible solution to a $d=3$, $n=3$ MAP: $\{111,222,333\}$. Then, one of its 2-exchange neighbors is $\{111,322,233\}$, another one is $\{131,222,313\}$; a 3-exchange neighbor is given by $\{311,122,233\}$, etc. Evidently, one has $\mathcal{N}_p\subset\mathcal{N}_{p+1}$ for $p=2,\ldots,n-1$.

Proposition 1. *For any $p=2,\ldots,n$, the size $|\mathcal{N}_p|$ of the p-exchange local neighborhood of a feasible solution of a MAP (1) is equal to*

$$|\mathcal{N}_p|=d\sum_{k=2}^{p}D(k)\binom{n}{k},\qquad\text{where}\quad D(k)=\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j!\,. \tag{16}$$

The quantity $D(k)$ in (16) is known as the number of *derangements* of a k-element set [29], i. e., the number of permutations $\{1,2,\ldots,k\}\mapsto\{i^{(1)},i^{(2)},\ldots,i^{(k)}\}$ such that $i^{(1)}\ne1,\ldots,i^{(k)}\ne k$, and can be easily calculated by means of the recurrent relation (see [29])

$$D(k)=kD(k-1)+(-1)^k,\quad D(1)=0\,,$$

so that, for example, $D(2)=1$, $D(3)=2$, $D(4)=9$, and so on. Then, according to Proposition 1, the size of a 2-exchange neighborhood is $|\mathcal{N}_2|=d\binom{n}{2}$, the size of a 3-exchange neighborhood is $|\mathcal{N}_3|=d\left[\binom{n}{2}+2\binom{n}{3}\right]$, etc.

Note also that size of the p-exchange neighborhood is *linear* in the number of dimensions d. Depending on p, $|\mathcal{N}_p|$ is either *polynomial* or *exponential* in the number of elements n per dimension, as follows from the representation

$$D(n)=n!\left(1-\tfrac{1}{1!}+\tfrac{1}{2!}-\tfrac{1}{3!}+\cdots+\tfrac{(-1)^n}{n!}\right)\approx\frac{n!}{e}\,,\qquad n\gg1\,.$$

The definition of a local minimum with respect to the p-exchange neighborhood is then straightforward. The kth feasible solution with cost z_k is a p-exchange local minimum iff $z_k\le z_j$ for all $j\in\mathcal{N}_p(k)$. Continuing the example above, the solution $\{111,222,333\}$ is a 2-exchange local minimum iff its cost $z_1=c_{111}+c_{222}+c_{333}$ is less than or equal to costs of all of its 2-exchange neighbors.

The number M_p of local minima of the MAP is obtained by counting the feasible solutions that are local minima with respect to neighborhoods $\mathcal{N}_p$. In a random MAP, where the assignment costs are random variables, M_p becomes a random quantity itself. In this paper we are interested in determining the expected number $\mathsf{E}[M_p]$ of local minima in random MAPs that have iid assignment costs with continuous distribution.

Expected Number of Local Minima in MAP with $n=2$

As it was noted above, in the special case of random MAP with $n=2$, $d\ge3$, the costs of feasible solutions are iid random variables with distribution $F*F$, where F is the distribution of the assignment costs. This special structure of the feasible set allows for a closed-form expression for the expected number of local minima $\mathsf{E}[M]$ (note that in a $n=2$ MAP the largest local neighborhood is $\mathcal{N}_2$, thus $M=M_2$), as established in [11].

Theorem 2. *In a $n=2$, $d\ge3$ MAP with cost coefficients that are iid continuous random variables, the expected number of local minima is given by*

$$\mathsf{E}[M]=\frac{2^{d-1}}{d+1}\,. \tag{17}$$

Equality (17) implies that in a $n=2$, $d\ge3$ MAP the number of local minima $\mathsf{E}[M]$ is *exponential* in d, when the cost coefficients are independently drawn from *any* continuous distribution.

Expected Number of Local Minima in a Random MAP with Normally Distributed Costs

Our ability to derive a closed-form expression (17) for the expected number of local minima $\mathsf{E}[M]$ in the previous section has relied on the independence of feasible solution costs (2) in a $n = 2$ MAP. As it is easy to verify directly, in the case $n \geq 3$ the costs of feasible solutions are generally not independent. This complicates analysis significantly if an arbitrary continuous distribution for assignment costs $c_{i_1 \cdots i_d}$ in (1) is assumed. However, as we show below, one can derive upper and lower bounds for $\mathsf{E}[M]$ in the case when the costs coefficients of (1) are independent normally distributed random variables. First, we develop bounds for the number of local minima $\mathsf{E}[M_2]$ defined with respect to 2-exchange neighborhoods $\mathcal{N}_2$ that are most widely used in practice.

2-exchange Local Neighborhoods Noting that in the general case the number N of the feasible solutions to MAP (1) is equal to $N = (n!)^{d-1}$, the expected number of local minima $\mathsf{E}[M_2]$ with respect to local 2-exchange neighborhoods can be written in the form

$$\mathsf{E}[M_2] = \sum_{k=1}^{N} \mathsf{P}\Big[\bigcap_{j \in \mathcal{N}_2(k)} z_k - z_j \leq 0 \Big] , \tag{18}$$

where $\mathcal{N}_2(k)$ is the 2-exchange neighborhood of the kth feasible solution, and z_i is the cost of the ith feasible solution, $i = 1, \ldots, N$. If we allow the n^d cost coefficients $c_{i_1 \cdots i_d}$ of the MAP to be independent standard normal $N(\mu, \sigma^2)$ random variables, then the probability term in (18) can be expressed as

$$\mathsf{P}\Big[\bigcap_{j \in \mathcal{N}_2(k)} z_k - z_j \leq 0 \Big] = F_{\Sigma}(\mathbf{0}) , \tag{19}$$

where F_{Σ} is the c.d.f. of the $|\mathcal{N}_2|$-dimensional random vector

$$\mathbf{Z} = (Z_{121}, \ldots, Z_{12d}, Z_{131}, \ldots, Z_{13d}, \cdots \cdots, Z_{rs1}, \ldots, Z_{rsd}, \cdots, Z_{n-1,n,1}, \ldots, Z_{n-1,n,d}) , \quad r < s . \tag{20}$$

Vector $\mathbf{Z}$ has a normal distribution $N(\mathbf{0}, \Sigma)$ with the covariance matrix Σ defined as

$$\mathrm{Cov}(Z_{rsq}, Z_{ijk}) = \begin{cases} 4\sigma^2, & \text{if} \quad i = r, j = s, q = k, \\ 2\sigma^2, & \text{if} \quad i = r, j = s, q \neq k, \\ \sigma^2, & \text{if} \quad (i = r, j \neq s) \quad \text{or} \quad (i \neq r, j = s) , \\ 0, & \text{if} \quad i \neq r, j \neq s. \end{cases} \tag{21}$$

While the value of $F_{\Sigma}(\mathbf{0})$ in (19) is difficult to compute exactly for large d and n, lower and upper bounds can be constructed using Slepian's inequality [30]. To this end, we introduce covariance matrices $\underline{\Sigma} = (\underline{\sigma}_{ij})$ and $\overline{\Sigma} = (\bar{\sigma}_{ij})$ as

$$\underline{\sigma}_{ij} = \begin{cases} 4\sigma^2, & \text{if} \quad i = j, \\ 2\sigma^2, & \text{if} \quad i \neq j \text{ and} \\ & (i-1) \operatorname{div} d = (j-1) \operatorname{div} d \\ 0, & \text{otherwise} \end{cases} , \tag{22a}$$

$$\bar{\sigma}_{ij} = \begin{cases} 4\sigma^2, & \text{if} \quad i = j, \\ 2\sigma^2, & \text{otherwise} \end{cases} , \tag{22b}$$

so that $\underline{\sigma}_{ij} \leq \sigma_{ij} \leq \overline{\sigma}_{ij}$ holds for all $1 \leq i, j \leq |\mathcal{N}_2|$, with σ_{ij} being the components of the covariance matrix Σ (21). Then, Slepian's inequality claims that

$$F_{\underline{\Sigma}}(\mathbf{0}) \leq F_{\Sigma}(\mathbf{0}) \leq F_{\overline{\Sigma}}(\mathbf{0}) , \tag{23}$$

where $F_{\underline{\Sigma}}(\mathbf{0})$ and $F_{\overline{\Sigma}}(\mathbf{0})$ are c.d.f.'s of random variables $\mathbf{X}_{\underline{\Sigma}} \sim N(\mathbf{0}, \underline{\Sigma})$ and $\mathbf{X}_{\overline{\Sigma}} \sim N(\mathbf{0}, \overline{\Sigma})$ respectively. The structure of matrices $\underline{\Sigma}$ and $\overline{\Sigma}$ allows the corresponding values $F_{\underline{\Sigma}}(\mathbf{0})$ and $F_{\overline{\Sigma}}(\mathbf{0})$ to be computed in a closed form, which leads to the following bounds for the expected number of local minima in random MAP with iid normal coefficients:

Theorem 3. *In a $n \geq 3, d \geq 3$ MAP with iid normal cost coefficients, the expected number of 2-exchange local minima is bounded as*

$$\frac{(n!)^{d-1}}{(d+1)^{n(n-1)/2}} \leq \mathsf{E}[M_2] \leq \frac{2(n!)^{d-1}}{n(n-1)d + 2} . \tag{24}$$

Note that both the lower and upper bounds in (24) coincide with the exact expression (17) for $\mathsf{E}[M_2]$ in the case $n = 2$. Also, from (24) it follows that for fixed $n \geq 3$, the expected number of local minima is *exponential* in the number of dimensions d for a fixed n.

Higher-Order Neighborhoods ($p \geq 3$) The outlined approach is applicable to general p-exchange neighborhoods. For convenience, here we consider the neighborhoods $\mathcal{N}_p^*$ as defined in Sect. "Local Minima and p-exchange Neighborhoods in MAP", i. e., the neighborhoods obtained from a given feasible solution by permuting *exactly* p elements in one of the d dimensions, so that for any feasible solution $i = \{i_1^{(1)} \cdots i_d^{(1)}, \ldots, i_1^{(n)} \cdots i_d^{(n)}\}$ and its p-exchange neighbor $j = \{j_1^{(1)} \cdots j_d^{(1)}, \ldots, j_1^{(n)} \cdots j_d^{(n)}\} \in \mathcal{N}_p^*(i)$ one has (compare to (15))

$$\sum_{r=1}^{n} \bar{\delta}_{i_{k_0}^{(r)} j_{k_0}^{(r)}} = p, k_0 \in \{1, \ldots, d\}, \quad \text{and}$$
$$\sum_{r=1}^{n} \bar{\delta}_{i_k^{(r)} j_k^{(r)}} = 0 \quad \text{for all} \quad k \in \{1, \ldots, d\} \backslash k_0 \, . \tag{25}$$

Then, upper and lower bounds for the expected number of local minima $\mathsf{E}[M_p^*]$ defined with respect to p-exchange neighborhoods $\mathcal{N}_p^*$ can be derived in a similar fashion. Namely, the sought probability

$$\mathsf{P}\Big[\bigcap_{i \in \mathcal{N}_p^*(k)} z_k - z_i \leq 0\Big] = F_{\Sigma_p}(\mathbf{0})$$

can be bounded as $F_{\underline{\Sigma}_p}(\mathbf{0}) \leq F_{\Sigma_p}(\mathbf{0}) \leq F_{\overline{\Sigma}_p}(\mathbf{0})$, where the matrices $\overline{\Sigma}_p, \underline{\Sigma}_p \in \mathbb{R}^{|\mathcal{N}_p^*| \times |\mathcal{N}_p^*|}$ are such that

$$\big(\overline{\Sigma}_p\big)_{ij} = \begin{cases} 2p\sigma^2, & \text{if } i = j, \\ (2p-2)\sigma^2, & \text{if } i \neq j, \end{cases} \tag{26a}$$

$$\big(\underline{\Sigma}_p\big)_{ij} = \begin{cases} 2p\sigma^2, & \text{if } i = j\,, \\ p\sigma^2, & \text{if } i \neq j \text{ and } (i-1) \operatorname{div} \big(dD(p)\big) \\ & = (j-1) \operatorname{div} \big(dD(p)\big)\,, \\ 0, & \text{otherwise}\,. \end{cases} \tag{26b}$$

The corresponding bounds for the expected number of local minima $\mathsf{E}[M_p^*]$ are established by the following theorem [11].

Theorem 4. *In a $n \geq 3, d \geq 3$ MAP with iid normal cost coefficients, the expected number of local minima M_p^* with respect to p-exchange local neighborhoods $\mathcal{N}_p^*$ is bounded as*

$$\frac{n!^{d-1}}{\big[dD(p)+1\big]^{\binom{n}{p}}} \leq \mathsf{E}[M_p^*] \leq n!^{d-1} \int_{-\infty}^{+\infty} \Big[\Phi\big(\sqrt{p-1}z\big)\Big]^{d\binom{n}{p}D(p)} \mathrm{d}\Phi(z)\,, \tag{27}$$

where $\Phi(z)$ is the c.d.f. of the standard normal $N(0,1)$ distribution. For 3-exchange neighborhoods $\mathcal{N}_3^*$, an improved upper bound holds:

$$\mathsf{E}[M_3^*] \leq \frac{3n!^{d-1}}{n(n-1)(n-2)d+3}\,. \tag{28}$$

It is interesting to note that for a fixed p the ratio of number of local minima to the number of feasible solutions becomes infinitely small as the dimensions of the problem increase (see (17), (24), and (27)).

Conclusions

We have discussed asymptotical analysis of the expected optimal value and the expected number of local minima of the Multidimensional Assignment Problem whose assignment costs are iid random variables drawn from a continuous distribution. It has been demonstrated that for a broad class of distributions, the asymptotical behavior of the expected optimal cost of a random MAP in the case when one of the problem's dimension parameters approaches infinity is determined by the location of the left endpoint of the support set of the distribution. The presented analysis is constructive in the sense that it allows for derivation of lower and upper asymptotical bounds for the expected optimal value of the problem for a prescribed probability distribution.

In addition, we have derived a closed-form expression for the expected number of local minima in a $n = 2$ random MAP with arbitrary distribution of assignment costs. In the case $n \geq 3$, bounds for the expected number of local minima have been derived in the assumption that assignment costs are iid normal random variables. It has been demonstrated that the expected number of local minima is exponential in the number of dimensions d of the problem.

References

1. Aiex RM, Resende MGC, Pardalos PM, Toraldo G (2005) GRASP with Path Relinking for Three-Index Assignment. INFORMS J Comput 17(2):224–247
2. Aldous D (2001) The $\zeta(2)$ limit in the random assignment problem. Random Struct Algorithm 18(4):381–418
3. Alon N, Spencer J (2000) The Probabilistic Method, 2nd edn, Interscience Series in Discrete Mathematics and Optimization. Wiley, New York
4. Andrijich SM, Caccetta L (2001) Solving the multisensor data association problem. Nonlinear Analysis 47:5525–5536
5. Angel E, Zissimopoulos V (2001) On the landscape ruggedness of the quadratic assignment problem. Theor Comput Sci 263:159–172
6. Balas E, Saltzman MJ (1991) An algorithm for the three-index assignment problem. Oper Res 39:150–161
7. Burkard RE (2002) Selected topics on assignment problems. Discret Appl Math 123:257–302
8. Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman, San Francisco
9. Goemans MX, Kodialam M (1993) A lower bound on the expected value of an optimal assignment. Math Oper Res 18:267–274
10. Grundel DA, Oliveira CAS, Pardalos PM (2004) Asymptotic properties of random multidimensional assignment problems. J Optim Theory Appl 122(3):487–500
11. Grundel DA, Krokhmal PA, Oliveira CAS, Pardalos PM (2007) Asymptotic properties of random multidimensional assignment problems. J Comb Optim 13(1):1–18
12. Karp RM (1987) An upper bound on the expected cost of an optimal assignment. In: Discret Algorithm Complexity. Academic Press, Boston, pp 1–4
13. Krokhmal PA, Grundel DA, Pardalos P (2007) Asymptotic Behavior of the Expected Optimal Value of the Multidimensional Assignment Problem. Math Program 109(2–3):525–551
14. Lazarus AJ (1993) Certain expected values in the random assignment problem. Oper Res Lett 14:207–214
15. Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling salesman problem. Oper Res 21:498–516
16. Linusson S, Wästlund J (2004) A proof of Parisi's conjecture on the random assignment problem. Probab Theory Relat Fields 128(3):419–440
17. Mézard M, Parisi G (1985) Replicas and optimization. J Phys Lett 46(17):771–778
18. Murphey R, Pardalos P, Pitsoulis L (1998) A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. In: DIMACS Series, vol 40, American Mathematical Society, pp 277–302
19. Nair C, Prabhakar B, Sharma M (2005) A Proof of the Conjecture due to Parisi for the Finite Random Assignment Problem. Random Struct Algorithms 27(4):413–444
20. Olin B (1992) Asymptotic properties of the random assignment problem. Ph.D thesis, Royal Institute of Technology, Stockholm, Sweden
21. Palmer R (1991) Optimization on rugged landscapes. In: Perelson A, Kauffman S (eds) Molecular Evolution on Rugged Ladscapes: Proteins, RNA, and the Immune System. Addison Wesley, Redwood City, pp 3–25
22. Pardalos PM, Ramakrishnan KG (1993) On the expected optimal value of random assignment problems: Experimental results and open questions. Comput Optim Appl 2:261–271
23. Parisi G (1998) A conjecture on random bipartite matching. Physics e-Print archive, http://xxx.lang.gov/ps/cond-_mat/9801176
24. Papadimitrou CH, Steiglitz K (1998) Combinatorial Optimization: Algorithms and Complexity. Dover, New York
25. Pasiliao EL (2003) Algorithms for Multidimensional Assignment Problems. PhD thesis, University of Florida
26. Pierskalla W (1968) The multidimensional assignment problem. Oper Res 16:422–431
27. Poore AB (1994) Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking. Comput Optim Appl 3:27–54
28. Pusztaszeri J, Rensing PE, Liebling TM (1995) Tracking elementary particles near their primary vertex: a combinatorial approach. J Glob Optim 16:422–431
29. Stanley R (1986) Enumerative Combinatorics. Wadsworth and Brooks, Belmont CA
30. Tong YL (1990) The Multivariate Normal Distribution. Springer, Berlin
31. Veenman CJ, Hendriks EA, Reinders MJT (1998) A fast and robust point tracking algorithm. Proc Fifth IEEE Int Conf Image Processing 653–657, Chicago, USA
32. Walkup DW (1979) On the expected value of a random assignment problem. SIAM J Comput 8:440–442

Asynchronous Distributed Optimization Algorithms

Ioannis P. Androulakis
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 90C30, 90C30, 90C52, 90C53, 90C55

Article Outline

Keywords
See also
References

Keywords

Asynchronous iterative algorithms; Distributed computing; Optimization

Many iterative algorithms, deterministic or stochastic, admit distributed implementations, whereby the work load for performing computational steps, identified as bottlenecks, is distributed among a variety of computational nodes. Extensive literature regarding distributed implementations of optimization algorithms in particular is available, [19]. In recent years, there has been an extremely fruitful interface between mathematical programming algorithms and computer science. This has resulted in major advances in the development of algorithms and implementation of sophisticated optimization algorithms on high performance parallel and distributed computers, [11,12]. Two major issues are important in designing an efficient distributed implementation, namely, *task allocation*, and *communication protocol*. Task allocation relates to the breakdown of the total work load and this can either be static or dynamic depending. Communication patterns and frequency are important since they can induce substantial overhead in cases where workload irregularities occur. Various important implementational details have been presented, among others, in [10]. The straightforward translation of serial to a distributed algorithm would assume some sort of global synchronization mechanism that would guarantee that information among processing nodes is being exchanged once a computational step has been performed. Processors must then synchronize so as to exchange information and proceed all with the same type of information to their next computational step. *Asynchronous algorithms* relax the assumption of a predetermined synchronization protocol, and allow each processing element to compute and communicate following local rates. The primary motivation for developing algorithms was to address situations in which:

- processors do not need to communicate to each other processor at each time instance;
- processors may keep performing computations without having to wait until they receive the messages that have been transmitted to them;
- processors are allowed to remain idle some of the time;
- some processors may be performing computations faster than others.

Such algorithms can alleviate communication overloads and they are not excessively slowed down by either communication delays nor by differences in the time it takes processors to perform one computation, [18]. Another major motivation is clearly to develop robust algorithms for distributed computation on heterogeneous networks of computers. The ideas of asynchronous, also known as *chaotic*, iterative schemes, can be traced by to [9], in which special schemes for solving linear systems of equations were developed. For discussing the basic principles and conditions of asynchronous iterations, the formalism of [8] will be followed. This work presented the first comprehensive treatment of the recent developments in the theory and practice of asynchronous iterations for a variety of problems, including deterministic and stochastic optimization. In essence, most iterative algorithms can be viewed as the search for a fixed point that corresponds to the solution of the original problem. The basic assumptions of the model of asynchronous (chaotic) iterations for determining fixed point of (non)linear mappings are as follows:

1) Let X be a vector space and $x = (x_1, \ldots, x_n) \in X$ are n-tuples describing any vector from this set. It is also assumed that $X = X_1 \times \cdots \times X_n$, with $x_i \in X_i$, $i = 1, \ldots, n$.
2) Let $f: X \to X$ be a function defined by $f(x) = (f_1(x), \ldots, f_n(x))$, $\forall x \in X$.
3) A point $X^\star \in X$ is a *fixed point* of $f(x)$ if $x^\star = f(x^\star)$ or, equivalently, $x_i^\star = f_i(x^\star)$, $i = 1, \ldots, n$.

For the solution of the aforementioned problem, one can define an iterative method as:

$$x_i := f_i(x), \quad i = 1, \ldots, n,$$

with $x_i(t)$ being the values of the ith component at time (iteration) t. In order to comprehend the concept of asynchronous iterations, we assume that there exists a set of times $T = \{0, 1, \ldots\}$ at which one or more (possibly none) components x_i of x are updated by some processor of a distributed computing system. We defined by T^i the set of times at which x_i is updated. Given that no synchronization protocol dictating the information exchange exists, it is quite conceivable that not all processors have access to the same and most recent values of of the corresponding components of x. It will be

therefore assumed that:

$$x_i(t+1) = \begin{cases} f_i(x_1\tau_1^i(t), \ldots, x_n(\tau_n^i(t)), \\ \quad \forall t \in T^i,\ 0 \le \tau_j^i(t) \le t, \\ x_i(t), \quad \forall t \notin T^i. \end{cases}$$

In the aforementioned definition of the iterative process, the difference $t - \tau_j^i(t)$ between the current time t and the time $\tau_j^i(t)$ corresponding to the jth component available at the processor updating $x_i(t)$ can be viewed as some form of communication delay. In studying the convergence behavior of algorithms of this type, two cases have to be considered. The operation can either be *totally asynchronous* or *partially asynchronous*. The concept of totally asynchronous algorithms was first introduced in [9], and subsequently analyzed in, among other, [1,5,15]. [5] proposed a general framework that ensembles a variety of instances. The cornerstone of his approach is based on the *asynchronous convergence theorem*, [8]. It defined a general pattern for proving convergence of the asynchronous counterparts of certain sequential algorithms. The asynchronous convergence theorem can be applied to variety of problems including:

- problems involving maximum norm contraction mappings;
- problems involving monotone mappings;
- the shortest path problem;
- linear and nonlinear network flow problems.

Qualitatively speaking, the fundamental difference between a synchronous and an asynchronous iterative mapping, is similar to the differences between a Jacobi and a Gauss–Seidel iteration. Consider the implementation of both these approaches in the minimization of function $F(x)$. The specifics of the minimization algorithm are irrelevant:

- Jacobi:

$$x_i(t+1) = \arg\min_{x_i} F(x_1(t), \ldots, x_n(t));$$

- Gauss–Seidel:

$$x_i(t+1) = \arg\min_{x_i} F(x_1(t+1), \ldots, x_i(t), \ldots, x_n(t)).$$

The Gauss–Seidel approach corresponds to the instantaneous communication, in a sequential manner, of the information as it being generated. The Jacobi iteration, forces processors to perform iterations utilizing 'outdated' information. The asynchronous iteration is reminiscent to a Jacobi one. A thorough analysis and comparison of these two extremes is presented in [16]. A major class of iterative schemes that can be shown to be convergent when implemented asynchronously, are defined by mappings which can be shown to be *contraction mappings* with respect to a suitably defined *weighted maximum norm*:

$$\|x\|_\infty^\omega = \max_i \frac{|x_i|}{\omega_i},$$
$$x \in \mathbb{R}^n, \quad \omega \in \mathbb{R}_+^n.$$

Let us consider the minimization of an unconstrained quadratic function F:

$$\begin{cases} \min & F(x) = \frac{1}{2}x^\top Ax - b^\top x \\ \text{s.t.} & x \in \mathbb{R}^n, \end{cases}$$

where A is an $n \times n$ positive definite symmetric matrix, and $b \in \mathbf{R}^n$. A gradient iteration of the form

$$x := (I - \gamma A)x + \gamma b$$

will be convergent provided that the maximum row sum of $I - \gamma\, A$ is less than 1, i. e.:

$$\left|1 - \gamma\alpha_{ij}\right| + \sum_{j:j\neq i} \gamma \left|a_{ij}\right| < 1, \quad i = 1, \ldots, n,$$

implying the *diagonal dominance condition*:

$$a_{ij} > \sum_{j:j\neq i} \left|a_{ij}\right|, \quad \forall i.$$

If we consider the general nonlinear unconstrained optimization problem:

$$\begin{cases} \min & g(x) \\ \text{s.t.} & x \in \mathbb{R}^n, \end{cases}$$

where g: $\mathbf{R}^n \to \mathbf{R}$ is a twice-differentiable convex function, with Hessian matrix $\nabla^2 g(x)$ which is positive definite. If one considers a Newton mapping given by:

$$f(x) = x - [\nabla^2 g(x)]^{-1} \nabla g(x)$$

The norm $\| \mathrm{x} \| = \max_i |x_i|$ makes f a contraction mapping in the neighborhood of $x^\star$ (the optimal point). Extensions of the ordinary gradient method

$$f(x) = x - \alpha \nabla g(x)$$

are also discussed in [5]. The shortest path problem is defined in terms of a directed graph consisting of n nodes. We denote by $A(i)$ the set of all nodes j for which there is an outgoing arc (i, j) from node i. The problem is to find a path of minimum length starting at node i and ending at node j. [4] considered the application of the asynchronous convergence theorem to fixed point iterations involving monotone mappings by considering the Bellman–Ford algorithm, [3], applied to the *shortest path problem*. This takes the form:

$$x_i(t+1) = \min_{j \in A(i)} (a_{ij} + x_j(\tau_j^i(t)),$$
$$i = 2, \ldots, n, \quad t \in T^i,$$
$$x_1(t+1) = 0.$$

$A(i)$ is the set of all nodes j for which there exists an arc (i, j). *Linear network flow problems* are discussed in [8] and asynchronous distributed versions of the auction algorithm are discussed. In the general linear network flow problem we are given a set of N nodes and a set of arcs A, each arc (i, j) has associated with it an integer a_{ij}, referred to as the cot coefficient. The problem is to optimally assign flows, f_{ij} to each one of the arcs, and the problem is represented mathematically as follows:

$$\begin{cases} \min & \displaystyle\sum_{(i,j)\in A} a_{ij} f_{ij} \\ \text{s.t.} & \displaystyle\sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)} f_{ji} = s_i, \ \forall i \in N, \\ & b_{ij} \le f_{ij} \le c_{ij}, \quad \forall (i,j) \in A, \end{cases}$$

where a_{ij}, b_{ij}, c_{ij} and s_i are integers. Extensions of the sequential *auction algorithms* are discussed in [6], in which asynchronism manifests itself in the sense that certain processors may be calculating actions bids which other update object prices. [7] extended the analysis to cover certain classes of *nonlinear network flow problems* in which the costs a_{ij} are functions of the flows f_{ij}:

$$\begin{cases} \min & \displaystyle\sum_{(i,j)\in A} a_{ij}(f_{ij}) \\ \text{s.t.} & \displaystyle\sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)} f_{ji} = s_i, \ \forall i \in N, \\ & b_{ij} \le f_{ij} \le c_{ij}, \quad \forall (i,j) \in A. \end{cases}$$

Imposing additional reasonable assumptions to the general framework of totally asynchronous iterative algorithms can substantially increase the applicability of the concept. A natural extension is therefore the *partially asynchronous iterative methods*, whereby two major assumptions are be satisfied:

a) each processor performs an update at least once during any time interval of length B;
b) the information used by any processor is outdated by at most B time units.

In other words, the partial asynchronism assumption extends the original model of computation by stating that:

There exists a positive integer B such that:

- For every i and for every $t \ge 0$, at least one of the elements of the set $\{t, \ldots, t + B - 1\}$ belongs to T^i.
- There holds:

$$t - B \le \tau_j^i(t) \le t,$$

for all i and j, and all $t \ge 0$ belonging to T^i.
- There holds $\tau_i^i(t) = t$ for all i and $t \in T^i$.

[17] developed a very elegant framework with important implications on the asynchronous minimization of continuous functions. It was established that, while minimize function $F(x)$, the asynchronous implementation of a gradient-based algorithm:

$$x := x - \gamma \nabla F(X)$$

is convergent if and only if the stepsize γ is small compared to the inverse of the asynchronism measure B. Specifically, let F: $\mathbf{R}^n \to \mathbf{R}$ be a cost function to be minimized subject to no constraints. It will be further assumed that:

1) $F(x) > 0$, $\forall x \in \mathbf{R}^n$;
2) $F(x)$ is *Lipschitz continuous*:

$$\|\nabla F(x) - \nabla F(y)\| \le K_1 \|x - y\|,$$
$$\forall x, y, \in \mathbb{R}^n.$$

The asynchronous gradient algorithm of the synchronous iteration:

$$x := x - \gamma \nabla F(x)$$

is denoted by:

$$x_i(t+1) := x_i(t) - \gamma s_i(t), \quad i = 1, \ldots, n,$$

where γ is a positive stepsize, and $s_i(t)$ is the update direction. It will be assumed that

$$s_i(t) = 0, \quad \forall t \notin T^i.$$

It is important to realize that processor i at time time t has knowledge of a vector $x^i(t)$ that is a, possibly, outdated version of $x(t)$. In other words: $x^i(t) = ((x_1(\tau_1^i(t)), \ldots, x_n(\tau_n^i(t)))$. It is further assumed that when x_i is being updated, the update direction s_i is a *descent direction*: For every i and t:

$$s_i(t)\nabla_i F(x^i(t)) \leq 0$$

there exists positive constants K_2, K_3 such that

$$K_1 \left|\nabla_i F(x^i(t))\right| \leq |s_i(t)| \leq K_3 \left|\nabla_i F(x^i(t))\right|, \quad \forall t \in T^i, \quad \forall i.$$

If all of the above is satisfied, then for the asynchronous gradient iteration it can be shown that: There exists some γ_0, depending on n, B, K_1, K_3, such that if $0 < \gamma < \gamma_0$, then $\lim_{t\to\infty} \lambda\, F(x(t)) = 0$.

It can actually be further shown that the choice

$$\gamma = \frac{1}{K_3 K_1 (1 + B + nB)}$$

can guarantee convergence of the asynchronous algorithm. This results clearly states that one can always, in principle, identify an adequate stepsize for any finite delay.

Furthermore, [14] elaborated on the use of gradient projection algorithm, within the asynchronous iterative framework, for addressing certain classes of constraint nonlinear optimization problems. The constrained optimization problems considered, is that of minimizing a convex function F: $\mathbf{R}^n \to \mathbf{R}$, defined over the space $X = \prod_{i=1}^n X_i$ of lower-dimensional sets $X_i \subset \mathbf{R}^{n_i}$, and $\sum_{i=1}^m n_i = n$. The ith component of the solution vector is now updated by

$$x_i(t+1) = [x_i(t) - \gamma \nabla_i F(x^i(t))]^+$$

where $[\cdot]^+$ denotes the projection on the set X_i. Once again: $x_i(t+1) = x_i(t)$, $t \notin T^i$. Once again, a gradient based algorithm is defined, for which

$$s_i(t) = \begin{cases} \frac{1}{\gamma}\left([x_i(t) - \gamma\nabla_i F(x^i(t))]^+ - x_i(t)\right), \\ \quad t \in T^i, \\ 0 \quad t \notin T^i. \end{cases}$$

It can actually be shown that for, provided that the partial asynchronism assumption holds, one can always define, in principle, a suitable stepsize γ_0 such that for any $0 < \gamma < 0$ the limit point, x^*, of the sequence generated by the partially asynchronous gradient projection iteration minimizes the Lipschitz continuous, convex function F over the set X. Recently, [2], analyzed asynchronous algorithms for minimizing a function when the communication delays among processors are assumed to be stochastic with Markovian character. The approach is also based on a gradient projection algorithm and was used to address a an optimal routing problem.

A major consideration in asynchronous distributed computing is the fact that since no globally controlling mechanism exists makes the use of any termination criterion which is based on local information obsolete. Clearly, when executing asynchronously a distributed iteration of the form $x_i - f_i(x)$ local error estimates can, and will be, misleading in terms of the global state of the system. Recently [13] made several suggestions as to how the standard model can be supplemented with an additional interprocessor communication protocol so as to address the issue of finite termination of asynchronous iterative algorithms.

See also

- ► Automatic Differentiation: Parallel Computation
- ► Heuristic Search
- ► Interval Analysis: Parallel Methods for Global Optimization
- ► Load Balancing for Parallel Optimization Techniques
- ► Parallel Computing: Complexity Classes
- ► Parallel Computing: Models
- ► Parallel Heuristic Search
- ► Stochastic Network Problems: Massively Parallel Solution

References

1. Baudet GM (1978) Asynchronous iterative methods for multiprocessors. J ACM 25:226–244
2. Beidas BF, Papavassilopoulos GP (1995) Distributed asynchronous algorithms with stochastic delays for constrained optimization problems with conditions of time drift. Parallel Comput 21:1431–1450

3. Bellman R (1957) Dynamic programming. Princeton University Press, Princeton
4. Bertsekas DP (1982) Distributed dynamic programming. IEEE Trans Autom Control AC-27:610–616
5. Bertsekas DP (1983) Distributed asynchronous computation of fixed points. Math Program 27:107–120
6. Bertsekas DP, Eckstein J (1987) Distributed asynchronous relaxation methodfs for linear network flow problems. Proc IFAC:39–56
7. Bertsekas DP, El Baz D (1987) Distributed asynchronous relaxation methods for convex network flow problems. SIAM J Control Optim 25:74–85
8. Bertsekas DP, Tsitsiklis JN (1989) Parallel and distributed computation: Numerical methods. Prentice-Hall, Englewood Cliffs, NJ
9. Chazan D, Miranker W (1968) Chaotic reaxation. Linear Alg Appl 2:199–222
10. Ferreira A, Pardalos PM (eds) (1997) Solving combinatorial optimization problems in parallel. Springer, Berlin
11. Pardalos PM, Phillips AT, Rosen JB (eds) (1992) Topics in parallel computing in mathematical programming. Sci Press, Marrickville, Australia
12. Pardalos PM (ed) (1992) Advances in optimization and parallel computing. North-Holland, Amsterdam
13. Savari SA, Bertsekas DP (1996) Finite termination of asynchronous iterative algorithms. Parallel Comput 22:39–56
14. Tseng P (1991) On the rate of convergence of a partially asynchronous gradient projection algorithm. SIAM J Optim 1:603–619
15. Tsitsiklis JN (1987) On the stability of asynchronous iterative processes. Math Syst Theory 20:137–153
16. Tsitsiklis JN (1989) A comparison of Jacobi and Gauss–Seidel parallel iterations. Appl Math Lett 2:167–170
17. Tsitsiklis JN, Bertsekas DP, Athans M (1986) Distributed asynchronous deterministic and stochastic gradient optimization algorithms. IEEE Trans Autom Control ac-31:803–813
18. Tsitsiklis JN, Bertsekas DP, Athnas M (1986) Distributed asynchronous deterministic and stochastic gradient optimization algorithms. IEEE Trans Autom Control AC-31:803–812
19. Zenios AS (1994) Parallel numerical optimization: Current status and annotated bibliography. ORSA J Comput 1: 20–42

Auction Algorithms

DIMITRI P. BERTSEKAS
Labor. Information and Decision Systems,
Massachusetts Institute Technol., Cambridge, USA

MSC2000: 90C30, 90C35

Article Outline

Keywords
The Auction Process
Optimality Properties at Termination
Computational Aspects: ϵ-Scaling
Parallel and Asynchronous Implementation
Variations and Extensions
See also
References

Keywords

Linear programming; Optimization; Assignment problem; Transshipment problem

The auction algorithm is an intuitive method for solving the classical assignment problem. It outperforms substantially its main competitors for important types of problems, both in theory and in practice, and is also naturally well suited for parallel computation. In this article, we will sketch the basic principles of the algorithm, we will explain its computational properties, and we will discuss its extensions to more general network flow problems. For a detailed presentation, see the survey paper [3] and the textbooks [2,4]. For an extensive computational study, see [8]. The algorithm was first proposed in the 1979 report [1].

In the classical *assignment problem* there are n persons and n objects that we have to match on a one-to-one basis. There is a benefit a_{ij} for matching person i with object j and we want to assign persons to objects so as to maximize the total benefit. Mathematically, we want to find a one-to-one assignment [a set of person-object pairs $(1, j_1), \ldots, (n, j_n)$, such that the objects $j_1, \ldots, j_n$ are all distinct] that maximizes the total benefit $\sum_{i=1}^{n} a_{ij_i}$.

The assignment problem is important in many practical contexts. The most obvious ones are resource allocation problems, such as assigning personnel to jobs, machines to tasks, and the like. There are also situations where the assignment problem appears as a subproblem in various methods for solving more complex problems.

The assignment problem is also of great theoretical importance because, despite its simplicity, it embodies a fundamental linear programming structure. The most important type of linear programming prob-

lems, the linear network flow problem, can be reduced to the assignment problem by means of a simple reformulation. Thus, any method for solving the assignment problem can be generalized to solve the linear network flow problem, and in fact this approach is particularly helpful in understanding the extension of auction algorithms to network flow problems that are more general than assignment.

The classical methods for assignment are based on iterative improvement of some cost function; for example a primal cost (as in primal simplex methods), or a dual cost (as in Hungarian-like methods, dual simplex methods, and relaxation methods). The auction algorithm departs significantly from the cost improvement idea; at any one iteration, it may deteriorate both the primal and the dual cost, although in the end it finds an optimal assignment. It is based on a notion of approximate optimality, called ϵ-*complementary slackness*, and while it implicitly tries to solve a dual problem, it actually attains a dual solution that is not quite optimal.

The Auction Process

To develop an intuitive understanding of the auction algorithm, it is helpful to introduce an economic equilibrium problem that turns out to be equivalent to the assignment problem. Let us consider the possibility of matching the n objects with the n persons through a market mechanism, viewing each person as an economic agent acting in his own best interest. Suppose that object j has a price p_j and that the person who receives the object must pay the price p_j. Then, the (net) value of object j for person i is $a_{ij} - p_j$ and each person i would logically want to be assigned to an object j_i with maximal value, that is, with

$$a_{ij_i} - p_{j_i} = \max_{j=1,\ldots,n} \{a_{ij} - p_j\}. \tag{1}$$

We will say that a person i is 'happy' if this condition holds and we will say that an assignment and a set of prices are at *equilibrium* when all persons are happy.

Equilibrium assignments and prices are naturally of great interest to economists, but there is also a fundamental relation with the assignment problem; it turns out that an equilibrium assignment offers maximum total benefit (and thus solves the assignment problem), while the corresponding set of prices solves an associated dual optimization problem. This is a consequence of the celebrated duality theorem of linear programming.

Let us consider now a natural process for finding an equilibrium assignment. I will call this process the *naive auction algorithm*, because it has a serious flaw, as will be seen shortly. Nonetheless, this flaw will help motivate a more sophisticated and correct algorithm.

The naive auction algorithm proceeds in 'rounds' (or 'iterations') starting with *any* assignment and *any* set of prices. There is an assignment and a set of prices at the beginning of each round, and if all persons are happy with these, the process terminates. Otherwise some person who is not happy is selected. This person, call him i, finds an object j_i which offers maximal value, that is,

$$j_i \in \arg \max_{j=1,\ldots,n} \{a_{ij} - p_j\}, \tag{2}$$

and then:

a) Exchanges objects with the person assigned to j_i at the beginning of the round;
b) Sets the price of the best object j_i to the level at which he is indifferent between j_i and the second best object, that is, he sets p_{j_i} to

$$p_{j_i} + \gamma_i, \tag{3}$$

where

$$\gamma_i = v_i - w_i, \tag{4}$$

v_i is the best object value,

$$v_i = \max_j \{a_{ij} - p_j\}, \tag{5}$$

and w_i is the second best object value

$$w_i = \max_{j \neq j_i} \{a_{ij} - p_j\}, \tag{6}$$

that is, the best value over objects other than j_i. (Note that γ_i is the largest increment by which the best object price p_{j_i} can be increased, with j_i still being the best object for person i.)

This process is repeated in a sequence of rounds until all persons are happy.

We may view this process as an *auction*, where at each round the bidder i raises the price of his or her preferred object by the *bidding increment* γ_i. Note that γ_i

cannot be negative since $v_i \geq w_i$ (compare (5) and (6)), so the object prices tend to increase. Just as in a real auction, bidding increments and price increases spur competition by making the bidder's own preferred object less attractive to other potential bidders.

Does this auction process work? Unfortunately, not always. The difficulty is that the bidding increment γ_i is zero when more than one object offers maximum value for the bidder i (cf. (4) and (6)). As a result, a situation may be created where several persons contest a smaller number of equally desirable objects without raising their prices, thereby creating a never ending cycle.

To break such cycles, we introduce a perturbation mechanism, motivated by real auctions where each bid for an object must raise its price by a minimum positive increment, and bidders must on occasion take risks to win their preferred objects. In particular, let us fix a positive scalar ϵ and say that a person i is 'almost happy' with an assignment and a set of prices if the value of its assigned object j_i is within ϵ of being maximal, that is,

$$a_{ij_i} - p_{j_i} \geq \max_{j=1,\ldots,n} \{a_{ij} - p_j\} - \epsilon. \tag{7}$$

We will say that an assignment and a set of prices are *almost at equilibrium* when all persons are almost happy. The condition (7), introduced first in 1979 in conjunction with the auction algorithm, is known as ϵ-*complementary slackness* and plays a central role in several optimization contexts. For $\epsilon = 0$ it reduces to ordinary complementary slackness (compare (1)).

We now reformulate the previous auction process so that the bidding increment is always at least equal to ϵ. The resulting method, the *auction algorithm*, is the same as the naive auction algorithm, except that the bidding increment γ_i is

$$\gamma_i = v_i - w_i + \epsilon, \tag{8}$$

(rather than $\gamma_i = v_i - w_i$ as in (4)). With this choice, the bidder of a round is almost happy at the end of the round (rather than happy). The particular increment $\gamma_i = v_i - w_i + \epsilon$ used in the auction algorithm is the maximum amount with this property. Smaller increments γ_i would also work as long as $\gamma_i \geq \epsilon$, but using the largest possible increment accelerates the algorithm. This is consistent with experience from real auctions, which tend to terminate faster when the bidding is aggressive.

We can now show that this reformulated auction process terminates in a finite number of rounds, necessarily with an assignment and a set of prices that are almost at equilibrium. To see this, note that once an object receives a bid for the first time, then the person assigned to the object at every subsequent round is almost happy; the reason is that a person is almost happy just after acquiring an object through a bid, and continues to be almost happy as long as he holds the object (since the other object prices cannot decrease in the course of the algorithm). Therefore, the persons that are not almost happy must be assigned to objects that have never received a bid. In particular, once each object receives at least one bid, the algorithm must terminate. Next note that if an object receives a bid in m rounds, its price must exceed its initial price by at least $m\epsilon$. Thus, for sufficiently large m, the object will become 'expensive' enough to be judged 'inferior' to some object that has not received a bid so far. It follows that only for a limited number of rounds can an object receive a bid while some other object still has not yet received any bid. Therefore, there are two possibilities: either

a) the auction terminates in a finite number of rounds, with all persons almost happy, before every object receives a bid; or
b) the auction continues until, after a finite number of rounds, all objects receive at least one bid, at which time the auction terminates. (This argument assumes that any person can bid for any object, but it can be generalized for the case where the set of feasible person-object pairs is limited, as long as at least one feasible assignment exists.)

Optimality Properties at Termination

When the auction algorithm terminates, we have an assignment that is almost at equilibrium, but does this assignment maximize the total benefit? The answer here depends strongly on the size of ϵ. In a real auction, a prudent bidder would not place an excessively high bid for fear that he might win the object at an unnecessarily high price. Consistent with this intuition, we can show that if ϵ is small, then the final assignment will be 'almost optimal'. In particular, we can show that the total benefit of the final assignment is within $n\epsilon$ of being

optimal. To see this, note that an assignment and a set of prices that are almost at equilibrium may be viewed as being at equilibrium for a *slightly different* problem where all benefits a_{ij} are the same as before, except for the n benefits of the assigned pairs which are modified by an amount no more than ϵ.

Suppose now that the benefits a_{ij} are all integer, which is the typical practical case (if a_{ij} are rational numbers, they can be scaled up to integer by multiplication with a suitable common number). Then, the total benefit of any assignment is integer, so if $n\epsilon < 1$, a complete assignment that is within $n\epsilon$ of being optimal must be optimal. It follows, that if

$$\epsilon < \frac{1}{n},$$

and the benefits a_{ij} are all integer, then the assignment obtained upon termination of the auction algorithm is optimal. Let us also note that the final set of prices is within $n\epsilon$ of being an optimal solution of the dual problem

$$\min_{\substack{p_j \\ j=1,\ldots,n}} \left\{ \sum_{j=1}^{n} p_j + \sum_{i=1}^{n} \max_j \{a_{ij} - p_j\} \right\}. \tag{9}$$

This leads to the interpretation of the auction algorithm as a dual algorithm (in fact an approximate coordinate ascent algorithm; see the cited literature).

Computational Aspects: ϵ-Scaling

The auction algorithm exhibits interesting computational behavior, and it is essential to understand this behavior to implement the algorithm efficiently. First note that the amount of work to solve the problem can depend strongly on the value of ϵ and on the maximum absolute object value

$$C = \max_{i,j} |a_{ij}|.$$

Basically, for many types of problems, the number of bidding rounds up to termination tends to be proportional to C/ϵ. Note also that there is a dependence on the initial prices; if these prices are 'near optimal,' we expect that the number of rounds to solve the problem will be relatively small.

The preceding observations suggest the idea of ϵ-*scaling*, which consists of applying the algorithm several times, starting with a large value of ϵ and successively reducing ϵ up to an ultimate value that is less than some critical value (for example, $1/n$, when the benefits a_{ij} are integer). Each application of the algorithm provides good initial prices for the next application. This is a very common idea in nonlinear programming, encountered for example, in barrier and penalty function methods. An alternative form of scaling, called *cost scaling*, is based on successively representing the benefits a_{ij} with an increasing number of bits, while keeping ϵ at a constant value.

In practice, it is a good idea to at least consider scaling. For sparse assignment problems, that is, problems where the set of feasible assignment pairs is severely restricted, scaling seems almost universally helpful. In theory, scaling leads to auction algorithms with a particularly favorable polynomial complexity (without scaling, the algorithm is pseudopolynomial; see the cited literature).

Parallel and Asynchronous Implementation

Both the bidding and the assignment phases of the auction algorithm are highly parallelizable. In particular, the bidding and the assignment can be carried out for all persons and objects simultaneously. Such an implementation can be termed *synchronous*. There are also *totally asynchronous* implementations of the auction algorithm, which are interesting because they are quite flexible and also tend to result in faster solution in some types of parallel machines. To understand these implementations, it is useful to think of a person as an autonomous decision maker who at unpredictable times obtains information about the prices of the objects. Each person who is not almost happy makes a bid at arbitrary times on the basis of its current object price information (that may be outdated because of communication delays).

See [7] for a careful formulation of the totally asynchronous model, and a proof of its validity, including extensive computational results on a shared memory machine, confirming the advantage of asynchronous over synchronous implementations.

Variations and Extensions

The auction algorithm can be extended to solve a number of variations of the assignment problem, such as the

asymmetric assignment problem where the number of objects is larger than the number of persons and there is a requirement that all persons be assigned to some object. Naturally, the notion of an assignment must now be modified appropriately. To solve this problem, the auction algorithm need only be modified in the choice of initial conditions. It is sufficient to require that all initial prices be zero. A similar algorithm can be used for the case where there is no requirement that all persons be assigned. Other variations handle efficiently the cases where there are several groups of 'identical' persons or objects ([5]).

There have been extensions of the auction algorithm for other types of linear network optimization problems. The general approach for constructing auction algorithms for such problems is to convert them to assignment problems, and then to suitably apply the auction algorithm and streamline the computations. In particular, the classical shortest path problem can be solved correctly by the naive auction algorithm described earlier, once the method is streamlined. Similarly, auction algorithms can be constructed for the max-flow problems, and are very efficient. These algorithms bear a close relation to preflow-push algorithms for the max-flow problem, which were developed independently of auction ideas.

The auction algorithm has been extended to solve linear transportation problems ([5]). The basic idea is to convert the transportation problem into an assignment problem by creating multiple copies of persons (or objects) for each source (or sink respectively), and then to modify the auction algorithm to take advantage of the presence of the multiple copies.

There are extensions of the auction algorithm for linear minimum cost flow (*transshipment*) problems, such as the so called ϵ-relaxation method, and the auction/sequential shortest path algorithm algorithm (see the cited literature for a detailed description). These methods have interesting theoretical properties and like the auction algorithm, are well suited for parallelization (see the survey [6], and the textbook [7]).

Let us finally note that there have been proposals of auction algorithms for convex separable network optimization problems with and without gains (but with a single commodity and without side constraints); see [9].

See also

- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Bertsekas DP (1979) A distributed algorithm for the assignment problem. Working Paper MIT Lab Information & Decision Systems
2. Bertsekas DP (1991) Linear network optimization: Algorithms and codes. MIT, Cambridge, MA
3. Bertsekas DP (1992) Auction algorithms for network flow problems: A tutorial introduction. Comput Optim Appl 1: 7–66
4. Bertsekas DP (1998) Network optimization: Continuous and discrete problems. Athena Sci., Belmont, MA
5. Bertsekas DP, Castañon DA (1989) The auction algorithm for transportation problems. Ann Oper Res 20:67–96
6. Bertsekas DP, Castañon DA, Eckstein J, Zenios S (1995) Parallel computing in network optimization. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Handbooks in OR and MS, vol 7, North-Holland, Amsterdam, pp 331–399
7. Bertsekas DP, Tsitsiklis JN (1989) Parallel and distributed computation: Numerical methods. Prentice-Hall, Englewood Cliffs, NJ
8. Castañon DA (1993) Reverse Auction Algorithms for Assignment Problems. In: Johnson DS, McGeoch CC (eds) Algorithms for network flows and matching. Amer Math Soc, Providence, pp 407–429
9. Tseng P, Bertsekas DP (1996) An epsilon-relaxation method for separable convex cost generalized network flow problems. Math Program (to appear), MIT Lab Information & Decision Systems P-2374

Automatic Differentiation: Calculation of the Hessian

LAURENCE DIXON
Numerical Optim. Centre, University Hertfordshire, Hatfield, England

MSC2000: 90C30, 65K05

Article Outline

Keywords
The Forward Mode
Illustrative Example 1: Forward Mode
The Mixed Method
Illustrative Example 2: Reverse Differentiation
Reverse Method
Illustrative Example 3: Reverse Gradient, Forward Hessian
See also
References

Keywords

Automatic differentiation; Gradient; Hessian; Doublet; Triplet

The *Hessian* of a scalar function $f(x)$ can be computed automatically in at least two ways. The first is a natural extension of the forward method for calculating gradients. The others extend the reverse method.

The Forward Mode

The concept of forward *automatic differentiation* was described by L.B. Rall [14]. When calculating the gradient vector of a function of n variables, a *doublet* data structure is introduced, consisting of $n + 1$ floating point numbers. To calculate the Hessian matrix, this data structure is extended to a *triplet*.

A triplet is a data structure that, in the simplest form, contains $1 + n + n(n+1)/2$ floating point numbers. If X is a variable that occurs in the evaluation of $f(x)$, then the triplet of X consists of

$$\left(X, \frac{\partial X}{\partial x_i}, \frac{\partial^2 X}{\partial x_i \partial x_j}\right)$$

for $i = 1, \ldots, n$ and $j \le i$.

The doublet consists of the first $n + 1$ elements of the triplet.

At the start of the function evaluation the triplets of the variables x_k must be set and these are simply $(x_k, e_k, 0)$ where e_k is the unit vector with 1 in the kth place, and 0 is the null matrix. If the function evaluation is expanded as a Wengert list [17] consisting of three types of operations,

- addition and subtraction,
- multiplication and division,
- nonlinear scalar functions,

then the arithmetic required to correctly update the triplets is easily deduced.

- If $X_k = X_l + X_m$, $l, m < k$, then to obtain the triplet of X_k, the elements of the triplets of X_l and X_m are simply added together element by element.
- If $X_k = X_l\, X_m$, $l, m < k$, then the background arithmetic is more complex as

$$\frac{\partial X_k}{\partial x_i} = X_l \frac{\partial X_m}{\partial x_i} + X_m \frac{\partial X_l}{\partial x_i}$$

and

$$\frac{\partial^2 X_k}{\partial x_i \partial x_j} = \frac{\partial X_l}{\partial x_j}\frac{\partial X_m}{\partial x_i} + X_l \frac{\partial^2 X_m}{\partial x_i \partial x_j} + \frac{\partial X_m}{\partial x_j}\frac{\partial X_l}{\partial x_i} + X_m \frac{\partial^2 X_l}{\partial x_i \partial x_j}.$$

As all these terms are stored in the triplets of X_l and X_m, given the triplets of X_l and X_m the triplet of X_k can be computed by a standard routine.

- If $X_k = \phi\,(X_m)$, $m < k$, then

$$\frac{\partial X_k}{\partial x_i} = \phi'(X_m)\frac{\partial X_m}{\partial x_i}$$

and

$$\frac{\partial^2 X_k}{\partial x_i \partial x_j} = \phi''(X_m)\frac{\partial X_m}{\partial x_i}\frac{\partial X_m}{\partial x_j} + \phi'(X_m)\frac{\partial^2 X_m}{\partial x_i \partial x_j}.$$

To perform this operation the values of $\phi'\,(X_m)$ and $\phi''\,(X_m)$ must be calculated with $\phi(X_m)$; all the other data is contained in the triplet of X_m.

Illustrative Example 1: Forward Mode

Consider the simple function

$$f(x) = (x_1 x_2 + \sin x_1 + 4)\left(3x_2^2 + 6\right)$$

In this case $n = 2$ and each triplet contains 6 floating point numbers, the value of X, its *gradient*, and the upper half of its Hessian. To evaluate the function, gradient, and Hessian, first expand the function in the Wengert list as shown in column 1 and then evaluate the triplets one by one. The evaluation is performed at the point (0, 1) below.

X_k	triplet(X_k)
$X_1 = x_1$	0, 1, 0, 0, 0, 0
$X_2 = x_2$	1, 0, 1, 0, 0, 0
$X_3 = X_1 X_2$	0, 1, 0, 0, 1, 0
$X_4 = \sin X_1$	0, 1 , 0, 0, 0, 0
$X_5 = X_3 + X_4$	0, 2, 0, 0, 1, 0
$X_6 = X_5 + 4$	4, 2, 0, 0, 1, 0
$X_7 = X_2^2$	1, 0, 2,0,0, 2
$X_8 = 3X_7$	3, 0, 6, 0, 0, 6
$X_9 = X_8 + 6$	9, 0, 6, 0, 0, 6
$X_{10} = X_6 X_9$	36, 18, 24, 0, 21, 24

The last row contains the values of the function, gradient and Hessian. The values for this simple problem can be easily verified by direct differentiation.

In practice forward automatic differentiation may be implemented in many ways, one possibility in many modern computer languages is to introduce the new data type triplet and over-write the meaning of the standard operators and functions so they perform the arithmetic described above. The code for the function evaluation can then be written normally without recourse to the Wengert list. Details of an implementation in Ada are given in [13]. A single run through a function evaluation code then computes the function, gradient and Hessian. If S is the store required to compute $f(x)$ then this method requires $(1 + n + n(n + 1)/2)S$ store. If M is the number of operations required to compute $f(x)$ then $(1 + 3n + 7n^2)M$ is a pessimistic bound on the operations required to compute the function, gradient and Hessian. Additional overheads are incurred to access the data type and the over-written operator subroutines. The efficiency is often improved by treating the triplet as a vector array and using sparse storage techniques. The number of zeros in the triplets of the above simple example illustrates the strength of the sparse form to calculate full Hessians. Maany reports the following results for the CPU time to differentiate the 50-dimensional Helmholz function (for details see [10]).

	Doublets		triplets	
	full	sparse	full	sparse
f	1.36	0.44	60.29	0.44
$f, \nabla f$	9.24	3.42	68.68	3.52
$f, \nabla f, \nabla^2 f$	N/A	N/A	476.36	20.69

The CPU time for calculating f alone within the full triplet package rises dramatically as although the derivative calculations are switched off the full package still allocates the space for the full triplet. Using the sparse package is also especially helpful if n is large and $f(x)$ is a partially separable function, i. e.

$$f(x) = \sum_k f_k(x)$$

where $f_k(x)$ only depends on a small number V_k of the n variables, as then, throughout the calculation of $f_k(x)$, the *sparse triplet* will only contain at most $1 + V_k + V_k(V_k + 1)/2$ nonzeros, and V_k will replace n in all the operation bounds, to give $\sum_k (1 + 3V_k + 7V_k^2)M_k$ operations.

One of the main purposes for calculating the Hessian matrix is to use it in optimization calculations. The truncated Newton method can be written so that it either requires the user to provide f, ∇f, and $\nabla^2 f$ at each outer iteration or f, ∇f at each outer iteration and $(\nabla^2 f)\, p$ at each inner iteration. The first method is ideally suited to be combined with sparse triplet differentiation. The algorithm is described in [9] and results given on functions of up to $n = 3000$ in [8]. The calculation of $(\nabla^2 f)\, p$ can also be undertaken simply by a modification of the triplet method.

In [7] the conclusion was drawn that

> the *sparse doublet* and sparse triplet codes in Ada enable normal code to be written for the func-

tion f and accurate values of ∇f and $\nabla^2 f$ to be obtained reliably by the computer. The major hope for automatic differentiation is therefore achieved.

Implementations are also available in Pascal.SC, C++, and Fortran90. The NOC Optima Library [1] code, *OPFAD*, implements the sparse doublet and triplet methods described above in Fortran90.

The Mixed Method

The advent of reverse automatic differentiation, A. Griewank [10], raised the hope that quicker ways could be found. The bound on the operations needed to compute the Hessian by the full forward triplet method contains the term $1/2n^2M$; by using a mixed method this is not required. The simplest mixed method is to use reverse automatic differentiation to compute the gradient which, [10], only requires $5M$ operations to compute the function and gradient for any value of n. This can be repeated at appropriate steps h along each axis, i. e. at $x + he_i$, $i = 1, \ldots, n$, and simple differences applied to the gradient vectors to calculate the Hessian in less than $5(n + 1)(M + 1)$ operations.

Illustrative Example 2: Reverse Differentiation

To obtain the gradient by *reverse differentiation* we must introduce the adjoint variables X_k^* and reverse back through the list. These rules are discussed in the previous article, but for convenience are repeated. If in the calculation of $f(x)$,

$$X_k = \phi(X_i, X_j), \qquad i, j < k,$$

then in the reverse pass

$$X_i^* = X_i^* + \frac{\partial \phi}{\partial X_i} X_k^*$$

and

$$X_j^* = X_j^* + \frac{\partial \phi}{\partial X_j} X_k^*.$$

For the same example the steps needed to calculate the gradient by reverse differentiation are

X_k^*	X_k^*
$X_{10}^* = 1$	1
$X_9^* = X_{10}^* X_6$	4
$X_6^* = X_{10}^* X_9$	9
$X_8^* = X_9^*$	4
$X_7^* = 3X_8^*$	12
$X_2^* = 2X_2 X_7^*$	24
$X_5^* = X_6^*$	9
$X_4^* = X_5^*$	9
$X_3^* = X_5^*$	9
$X_1^* = X_4^* \cos X_1$	9
$X_2^* = X_2^* + X_1 X_3^*$	24
$X_1^* = X_1^* + X_2 X_3^*$	18

giving the gradient as (18, 24) in agreement with the forward calculation. To perform this calculation the values of X_6 and X_9 were required which had been calculated during the function value calculation. The reverse gradient calculation must, therefore, follow a forward function evaluation calculation and the required data must be stored.

The bound $5M$ on the number of operations required to calculate the gradient is often very pessimistic, especially when the function evaluation uses matrix operations, [15], standard subroutines, [5], or when efficient sparse storage is used, [6]. The store required by this simple approach is simply that needed to calculate the gradient by reverse differentiation. The original reverse method required $O(M)$ store, but Griewank [11] describes how the store required can be reduced to $O(S \log M)$ at the cost of increasing the operation bound to $O(M \log M)$.

The accuracy obtained by calculating the Hessian by simple differences will depend on h but will often be sufficient as accurate Hessians are rarely required in optimization. Many software packages for calculating the gradient by reverse differentiation now exist, including the Optima Library Code *OPRAD* [1].

In 1998 the most widely used code to calculate gradients automatically is probably the *ADIFOR* code, [3], many examples of its use are given in that reference; unfortunately this implements a 'statement level hybrid mode'. In this, each assignment statement

$$Y_i = \Psi(Y_j, j < i, j \in J)$$

is treated in turn and the gradient, $\frac{\partial \Psi}{\partial Y_j}, j \in J$, computed efficiently by RAD but then to obtain the Doublet

$$\frac{\partial Y_i}{\partial x_m} = \sum_j \frac{\partial \Psi}{\partial Y_j} \frac{\partial Y_j}{\partial x_m}$$

many multiplications and additions may be required leading to a high operation count.

Reverse Method

A fully automatic approach could start by obtaining the Wengert list for the function and gradient as calculated by reverse automatic differentiation. This list will contain at most $5M$ steps. Then a forward sparse Doublet pass through this list could be performed that would need less than $(1 + 3n)\ 5M$ operations. The Doublet formed for the same example is illustrated below. In the Wengert list all identical Doublets are merged and composite steps involving more than one operation are split, it will be observed that the last two rows of the Doublet contain the gradient and Hessian, as desired, and that the number of operations, 22, is much less than the bound $5M = 50$. The storage requirement for this approach, when n is large, is considerably greater than that needed by the difference method. An alternative would be to perform a reverse pass through the gradient list. A full discussion is given in [4], who shows the two are identical in arithmetic, storage and operation count. His experience with his Ada implementation showed that the performance was very machine dependent. If the sparse Doublet approach is used with this reverse method on the partially separable function described above then the bound on the operations needed to obtain the Hessian reduces to $\sum_k 5(V_k + 1)(M_k + 1)$, a considerable saving. An early implementation, *PADRE2*, is described in [12]. A more recent code, *ADOL-F*, is described in [16]. Christianson's method is implemented in OPRAD, mentioned above. It should perhaps be mentioned that all the above methods can be hand-coded to solve any important problem without incurring the overheads still associated with most automatic packages, many of the helping hands described in [5] are still not implemented in an automatic package.

Further methods for speeding up the calculation of the Hessian are described in ► Automatic Differentiation: Calculation of Newton Steps.

Illustrative Example 3: Reverse Gradient, Forward Hessian

The variables in the Wengert list of the function and gradient calculation will be denoted by Y.

Y_k	Doublet Y_k
$Y_1 = x_1$	0, 1, 0
$Y_2 = x_2$	1, 0, 1
$Y_3 = Y_1 Y_2$	0, 1, 0
$Y_4 = \sin Y_1$	0, 1, 0
$Y_5 = Y_3 + Y_4$	0, 2, 0
$Y_6 = Y_5 + 4$	4, 2, 0
$Y_7 = Y_2^2$	1, 0, 2
$Y_8 = 3Y_7$	3, 0, 6
$Y_9 = Y_8 + 6$	9, 0, 6
$Y_{10} = Y_6 Y_9$	36, 18, 24
$Y_{11} = 1$	1, 0, 0
$Y_{12} = Y_{11} Y_6$	4, 2, 0
$Y_{13} = Y_{11} Y_9$	9, 0, 6
$Y_{14} = 3Y_{12}$	12, 6, 0
$Y_{15} = Y_2 Y_{14}$	12, 6, 12
$Y_{16} = 2Y_{15}$	24, 12, 24
$Y_{17} = \cos Y_1$	1, 0, 0
$Y_{18} = Y_{17} Y_{13}$	9, 0, 6
$Y_{19} = Y_1 Y_{13}$	0, 9, 0
$Y_{20} = Y_2 Y_{13}$	9, 0, 15
$Y_{21} = Y_{18} + Y_{20}$	18, 0, 21
$Y_{22} = Y_{16} + Y_{19}$	24, 21, 24

See also

- ► Automatic Differentiation: Calculation of Newton Steps
- ► Automatic Differentiation: Geometry of Satellites and Tracking Stations
- ► Automatic Differentiation: Introduction, History and Rounding Error Estimation
- ► Automatic Differentiation: Parallel Computation
- ► Automatic Differentiation: Point and Interval
- ► Automatic Differentiation: Point and Interval Taylor Operators
- ► Automatic Differentiation: Root Problem and Branch Problem
- ► Nonlocal Sensitivity Analysis with Automatic Differentiation

References

1. Bartholomew-Biggs MC Optima library. Numerical Optim. Centre Univ. Hertfordshire, England
2. Berz M, Bischof Ch, Corliss G, Griewank A (eds) (1996) Computational differentiation: techniques, applications, and tools. SIAM, Philadelphia
3. Bischof Ch, Carle A (1996) Users' experience with ADIFOR 2.0. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 385–392
4. Christianson DB (1992) Automatic Hessians by reverse accumulation. IMA J Numer Anal 12:135–150
5. Christianson DB, Davies AJ, Dixon LCW, Zee P Van der, (1997) Giving reverse differentiation a helping hand. Optim Methods Softw 8:53–67
6. Christianson DB, Dixon LCW, Brown S (1996) Sharing storage using dirty vectors. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 107–115
7. Dixon LCW (1993) On automatic differentiation and continuous optimization. In: Spedicato E (ed) Algorithms for continuous optimization: the state of the art. NATO ASI series. Kluwer, Dordrecht, pp 501–512
8. Dixon LCW, Maany Z, Mohsenina M (1989) Experience using truncated Newton for large scale optimization. In: Bromley K (ed) High Speed Computing, vol 1058. SPIE–The Internat. Soc. for Optical Engineering, Bellingham, WA, pp 94–104
9. Dixon LCW, Maany Z, Mohseninia M (1990) Automatic differentiation of large sparse systems. J Econom Dynam Control 14(2)
10. Griewank A (1989) On automatic differentiation. In: Iri M, Tanabe K (eds) Mathematical programming: recent developments and applications. Kluwer, Dordrecht, pp 83–108
11. Griewank A (1992) Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. Optim Methods Softw 1:35–54
12. Kubota K (1991) PADRE2, A FORTRAN precompiler yielding error estimates and second derivatives. In: Griewank A and Corliss GF (eds) Automatic differentiation of algorithms: theory, implementation, and application. SIAM, Philadelphia, pp 251–262
13. Maany ZA (1989) Ada automatic differentiation packages. Techn Report Hatfield Polytechnic NOC TR224
14. Rall LB (1981) Automatic differentiation: techniques and applications. Lecture Notes Computer Sci, vol 120. Springer, Berlin
15. Shiriaev D (1993) Fast automatic differentiation for vector processors and reduction of the spatial complexity in a source translation environment. PhD Thesis Inst Angew Math Univ Karlsruhe
16. Shiriaev D, Griewank A (1996) ADOL–F Automatic differentiation of Fortran codes. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 375–384
17. Wengert RE (1964) A simple automatic derivative evaluation program. Comm ACM 7(8):463–464

Automatic Differentiation: Calculation of Newton Steps

LAURENCE DIXON
Numerical Optim. Centre, University Hertfordshire, Hatfield, UK

MSC2000: 90C30, 65K05

Article Outline

Keywords
Jacobian Calculations
The Extended Matrix
Hessian Calculations
The Newton Step
Truncated Methods
See also
References

Keywords

Automatic differentiation; Jacobian matrix; Hessian matrix; Sparsity; Newton step

Many algorithms for solving optimization problems require the minimization of a merit function, which may be the original objective function, or the solution to sets of simultaneous nonlinear equations which may involve the constraints in the problem. To obtain second order convergence near the solution algorithms to solve both rely on the calculation of Newton steps.

When solving a set of nonlinear equations

$$s_j(x) = 0, \qquad j = 1, \ldots, n,$$

the *Newton step* d at $x^{(0)}$, $x \in \mathbf{R}^n$, is obtained by solving the linear set of equations

$$\sum_i \frac{\partial s_j}{\partial x_i} d_i = -s_j, \qquad j = 1, \ldots, n,$$

where both the derivatives $\partial s_j / \partial x_i$ and the vector function s_j are evaluated at a point, which we will denote by $x^{(0)}$.

For convenience we introduce the *Jacobian matrix* J and write the equation as

$$Jd = -s$$

When minimizing a function $f(x)$ the Newton equation becomes

$$\sum_i \frac{\partial^2 f}{\partial x_i \partial x_j} d_j = -\frac{\partial f}{\partial x_j}$$

where all the derivatives are calculated at a point, again denoted by $x^{(0)}$.

In terms of the *Hessian*, H, and the gradient, g, this can be written

$$Hd = -g$$

Automatic differentiation can be used to calculate the gradient, Hessian and Jacobian, but it can also be used to calculate the Newton step directly without calculating the matrices. In this article we will first discuss the calculation of the Jacobian, then extend briefly the calculation of the gradient and Hessian, which was the subject of ► Automatic differentiation: Calculation of the Hessian, and finally discuss the direct calculation of the Newton step.

Jacobian Calculations

If the functions s_j were each evaluated as separate entities, requiring M_j operations, then the derivatives could be evaluated by reverse automatic differentiation in 5 M_j operations. For many sets of functions it would, however, be very inefficient to evaluate the set s in this way, as considerable savings could be made by calculating threads of operations common to more than one s_j only once. In such situations the number of operations M required to evaluate the set s may be much less than $\sum_j M_j$. Under these circumstances the decision on how the Jacobian should be evaluated becomes much more complicated.

Before the advent of automatic differentiation the Jacobian was frequently approximated by one-sided differences

$$\frac{\partial s_j}{\partial x_i} = \frac{s_j\left(x^{(0)} + he_i\right) - s_j\left(x^{(0)}\right)}{h}$$

If the vector function s requires M Wengert operations, then the Jacobian would need $(n + 1)$ M operations by this approach. The accuracy of the result depends on a suitable choice of h. If simple forward automatic differentiation using doublets (see ► Automatic differentiation: Calculation of the Hessian) is used, an accurate Jacobian is obtained at a cost of $3nM$ operations. If a Newton step is to be calculated then the Jacobian must be square and so the simple reverse mode, which involves a backward pass through the Wengert list for each subfunction, would be bounded by 5 n M operations.

Most large Jacobians are sparse and M.J.D. Powell, A.R. Curtis, and J.R. Reid [5], introduced the idea of combining columns i that had no common nonzeros. Then, provided the sparsity pattern of J is known, the values in those columns can be reconstructed by a reduced number of differences. If the number of such *PCR groups* required to cover all the columns is c then the operations count is reduced to $(c + 1)$ M. For example, the columns of the following 5×5 sparse Jacobian could be divided into 3 groups

$$\begin{bmatrix} ■ & ▲ & 0 & 0 & ★ \\ ■ & ▲ & 0 & ★ & 0 \\ 0 & ▲ & ■ & ★ & 0 \\ 0 & 0 & ■ & 0 & ★ \\ ■ & ▲ & 0 & 0 & ★ \end{bmatrix}$$

indicated by ■, ▲, and ★.

This same grouping could be used with forward automatic differentiation to produce an accurate Jacobian in at most $3cM$ operations. If the sparse Doublet is used, the full benefit of *sparsity* within the calculation of the s_j is obtained, as well as the benefit due to sparsity in the Jacobian, without the need to determine the column groupings. Results showing the advantage of calculating large ($n = 5000$) Jacobians this way are given in [15] and summarised in [7].

It is possible for the calculation of some s_j to be independent of other s that do contain a common thread. It would obviously be efficient to calculate these s_j by reverse differentiation, requiring $5Mj$ operations. Reverse differentiation will also be appropriate if the common thread has less outputs than inputs. Then sparse reverse doublets, [2], should be used. These are implemented in OPRAD, see ► Automatic differentiation: Calculation of the Hessian.

T.F. Coleman et al. [3,4] demonstrated that calculating some columns using groups in the forward mode and some rows using groups in the reverse mode is considerably more efficient than using either alone. All nonzeros of the Jacobian must be included in a row and/or column computed. Similar results follow if some columns are computed using sparse doublets and some rows using the sparse reverse method. If C is the maximum number of nonzeros in a row within the columns computed forward and R the maximum number of nonzeros in a column within the rows computed in reverse then a crude bound on the number of operations is $(3C + 5R)M$. This bound does not allow for the additional sparsity in the early calculations nor for the fact that for some reverse calculations M_j should replace M. The selection of rows and columns taking account of such considerations is still unresolved.

But the advantages to be obtained can be appreciated by considering the arrow-head Jacobian, where only the diagonal elements and the last row and column contain nonzeros. If the gradient of s_n is computed using sparse reverse doublets this will require at most $5M_n$ operations and if the other gradients are computed using sparse forward doublets, no doublet will contain more than 2 nonzeros, so the operations will be bounded by $6M$. The total operations required in this case is independent of n.

The Extended Matrix

If the calculation of the functions s_j proceeds by a sequence of steps

$$X_k = x_k, \qquad k = 1, \ldots, n,$$
$$X_k = \phi_k(X_l, l \in L, l < k),$$
$$k = n + 1, \ldots, M + n,$$

with

$$s_j = X_{M+j}, \qquad j = 1, \ldots, n,$$

then

$$\frac{\partial X_k}{\partial x_m} = 0, \quad m \neq k, \quad k = 1, \ldots, n,$$
$$\frac{\partial X_k}{\partial x_k} = 1,$$

and

$$\frac{\partial X_k}{\partial x_m} = \sum_l \frac{\partial \phi_k}{\partial X_l} \frac{\partial X_l}{\partial x_m}.$$

If we now denote $\partial X_k / \partial x_m$ by Y_k and $\partial \phi_k / \partial X_l$ by L_{kl}, then this becomes

$$Y_k = \sum_l L_{kl} Y_l$$

i. e. the kth row of the matrix-vector product

$$(I - L)Y,$$

where the elements in the first n rows of L are all zeros, and then

$$\frac{\partial s_j}{\partial x_m} = Y_{M+j}.$$

Obtaining the Jacobian by the forward method may be considered as equivalent to solving

$$(I - L)Y = e_m.$$

Turning now to the reverse method if

$$X_k = \phi_k(X_l),$$

then the adjoint variable X_l^* contains a term

$$X_l^* = X_l^* + \frac{\partial \phi_k}{\partial X_l} X_k^*$$

which is the lth row of the matrix-vector product

$$(I - L^\top)X^*.$$

To obtain the gradient of s_m is therefore equivalent to solving

$$(I - L^\top)X^* = e_{M+m},$$

then

$$\frac{\partial s_m}{\partial x_i} = X_i^*.$$

So both the calculation of the Jacobian by the forward and backward method are equivalent to solving a very sparse set of equations. If the Wengert list is used, each row of L contains at most two nonzeros. It has therefore been suggested that methods for solving linear equations with sparse matrices could be used to calculate J, A. Griewank and S. Reese [14] suggested using the Markowitz rule, while U. Geitner, J. Utke and Griewank [11] applied the *method of Newsam and Ramsdell*.

Hessian Calculations

The calculation of the Hessian, as discussed in ► Automatic differentiation: Calculation of the Hessian, can also be formulated as a sparse matrix calculation. Using the notation of ► Automatic differentiation: Calculation of the Hessian if the calculation of $f(x)$ consists of

$$X_k = \phi_k(X_m, m < k, m \in M_k),$$

then the reverse gradient calculation consists of

$$X_m^* = X_m^* + X_k^* \frac{\partial \phi_k}{\partial X_m}, \quad m \in M_k.$$

If now we denote

$$Y_k = \frac{\partial X_k}{\partial x_i}, \qquad k = 1, \ldots, M,$$

and

$$Y_k = \frac{\partial X_{2M-k+1}^*}{\partial x_i}, \qquad k = M+1, \ldots, 2M,$$

then we obtain

$$Y_k = \frac{\partial \phi_k}{\partial X_m} Y_m, \qquad k = 1, \ldots, M,$$

and

$$Y_{2M+1-m} = Y_{2M+1-m} + Y_{2M+1-k} \frac{\partial \phi_k}{\partial X_m} + X_k^* \frac{\partial^2 \phi_k}{\partial X_m \partial X_j} Y_j.$$

The second derivatives are 1, if ϕ is a multiplication, 0 if ϕ is an addition, and if ϕ is unary only nonzero if $j = m$. If we denote these second order terms by B, the calculation of $H\, e_i$ is equivalent to solving

$$\begin{bmatrix} I - L & 0 \\ B & I - L^S \end{bmatrix} Y = \begin{pmatrix} e_i \\ 0 \end{pmatrix}.$$

Here the superscript S indicates that L has been transposed through both diagonals. The ith column of the Hessian is then the last n values of Y. For the illustrative example

$$f(x) = (x_1 x_2 + \sin x_1 + 4)(3x_2^2 + 6)$$

used in ► Automatic differentiation: Calculation of the Hessian, the off-diagonal nonzeros in the matrix which we will denote by K, are

$$\begin{aligned}
K_{3,1} &= K_{20,18} = X_2, \\
K_{3,2} &= K_{19,18} = X_1, \\
K_{4,1} &= K_{20,17} = \cos X_1, \\
K_{5,3} &= K_{18,16} = 1, \\
K_{5,4} &= K_{17,16} = 1, \\
K_{6,5} &= K_{16,15} = 1, \\
K_{7,2} &= K_{19,14} = 2X_2, \\
K_{8,7} &= K_{14,13} = 3, \\
K_{9,8} &= K_{13,12} = 1, \\
K_{10,6} &= K_{15,11} = X_9, \\
K_{10,9} &= K_{12,11} = X_6, \\
K_{12,6} &= K_{14,9} = X_{10}^*, \\
K_{19,1} &= K_{20,2} = X_3^*, \\
K_{19,2} &= 2X_7^*, \\
K_{20,1} &= -X_4^* \sin X_1,
\end{aligned}$$

L contains 11 nonzeros and B contains 6. The matrix is very sparse and the same sparse matrix techniques could be used to solve this system of equations.

The Newton Step

As the notation is easier we will consider the Jacobian case.

We have shown that if we solve $(I - L)\, Y = e_m$, then column m of the Jacobian J is in the last n terms of Y. If we wish to evaluate $J\, p$ we simply have to solve

$$(I - L)Y = p'$$

where p' has its first n terms equal to p and the remaining terms zero. Then the solution is again in the last n terms of Y. To calculate the Newton step we know $J\, d$ as it must be equal to $-\, s$, but we do not know d. We must therefore add the equations

$$Y_{M+i} = -s_i$$

to the equations, and delete the equations $Y_i = p_i$. For convenience we will partition L, putting the first n columns into A, retaining L for the remainder. So we have to solve

$$\begin{bmatrix} -A & I - L \\ 0 & E \end{bmatrix} \begin{pmatrix} d \\ Y \end{pmatrix} = \begin{pmatrix} 0 \\ -s \end{pmatrix}$$

for d. The matrix E is rectangular and is full of zeros except for the diagonals which are 1. Solving for d gives

$$E(I - L)^{-1}Ad = -s,$$

so

$$J = E(I - L)^{-1}A,$$

which is also the *Schur complement* of the sparse set of equations.

One popular way of solving a sparse set of equations is to form the Schur complement and solve the resulting equations, in this instance this becomes 'form J and solve $J\ d = -\ s$', which would be the normal indirect method. This also justifies the attention given in this article to the efficient calculation of J.

Griewank [12] observed that it may be possible to calculate the Newton step more cheaply than forming J and then solving the Newton equations. Utke [16] demonstrated that a number of ways of solving the sparse set of equations were indeed quicker. His implementation was compatible with ADOL-C and included many rules for eliminating variables. This approach was motivated by noting that if the Jacobian $J = D + a\ b^{\intercal}$, where D is diagonal and a and b vectors, then J is full and so solving $J\ x = -\ s$ is an $O(n^3)$ operation. However introducing one extra variable $z = b^{\intercal}\ x$ enables the *extended matrix* to be solved very cheaply

$$b^{\top}x - z = 0,$$
$$Dx + az = -s$$

gives

$$x = -D^{-1}(az + s),$$
$$z = -b^{\top}D^{-1}(az + s),$$

so

$$z = -(1 + b^{\top}D^{-1}a)^{-1}b^{\top}D^{-1}s,$$

and then x may be determined by substitution, which is an $O(n)$ operation. The challenge to find an automatic process that finds such short cuts is still open.

L.C.W. Dixon [6] noted that the extended matrix is an *echelon form*. An echelon matrix of degree k has ones on the k super-diagonal and zeros above it. If the lower part is sparse and contains NNZ nonzeros then the Schur complement can be computed in $kNNZ$ operations and the Newton step obtained by solving the resulting equations in $O(k^3)$ steps. The straight forward sparse system is an echelon form with $k = n$, so he suggested that by re-arranging rows and columns it might be possible to reduce k. This would reduce the operations needed for both parts of the calculation. Many sorting algorithms have been proposed for reducing the echelon index of sparse matrices. J.S. Duff et al. [9] discuss the performance of methods known as P^4 and P^5. R. Fletcher [10] introduced SPK1. Dixon and Z. Maany [8] introduced another which when applied to the extended matrix of the extended Rosenbrock function reduces the echelon index from n to $n/2$ and gives a diagonal Schur complement. It follows that this method, too, has considerable potential.

All these approaches still require further research.

Truncated Methods

Experience using the *truncated Newton* code has led many researchers to doubt the wisdom of calculating accurate Newton steps. Approximate solutions are often preferred in which the *conjugate gradient* method is applied to $H\ d = -\ g$; this can be implemented by calculating $H\ p$ at each inner iteration. $H\ p$ can be calculated very cheaply by a single forward doublet pass with initial values set at p through list for g obtained by reverse differentiation. The operations required to compute $H\ p$ are therefore bounded by $15M$.

If an iterative method is used to solve $J\ d = -\ s$, the products $J\ p$ and $J^{\intercal}\ v$ can both be obtained cheaply, the first by forward, the second by reverse automatic differentiation.

See also

- ► Automatic Differentiation: Calculation of the Hessian
- ► Automatic Differentiation: Geometry of Satellites and Tracking Stations
- ► Automatic Differentiation: Introduction, History and Rounding Error Estimation
- ► Automatic Differentiation: Parallel Computation
- ► Automatic Differentiation: Point and Interval
- ► Automatic Differentiation: Point and Interval Taylor Operators

- ▶ Automatic Differentiation: Root Problem and Branch Problem
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Interval Newton Methods
- ▶ Nondifferentiable Optimization: Newton Method
- ▶ Nonlocal Sensitivity Analysis with Automatic Differentiation
- ▶ Unconstrained Nonlinear Optimization: Newton–Cauchy Framework

References

1. Berz M, Bischof Ch, Corliss G, Griewank A (eds) (1996) Computational differentiation: techniques, applications, and tools. SIAM, Philadelphia
2. Christianson DB, Dixon LCW (1992) Reverse accumulation of Jacobians in optimal control. Techn Report Numer Optim Centre, School Inform Sci Univ Hertfordshire 267
3. Coleman TF, Cai JY (1986) The cyclic coloring problem and estimation of sparse Hessian matrices. SIAM J Alg Discrete Meth 7:221–235
4. Coleman TF, Verma A (1996) Structure and efficient Jacobian calculation. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 149–159
5. Curtis AR, Powell MJD, Reid JK (1974) On the estimation of sparse Jacobian matrices. J Inst Math Appl 13:117–119
6. Dixon LCW (1991) Use of automatic differentiation for calculating Hessians and Newton steps. In: Griewank A, Corliss GF (eds) Automatic Differentiation of Algorithms: Theory, Implementation, and Application. SIAM, Philadelphia, pp 114–125
7. Dixon LCW (1993) On automatic differentiation and continuous optimization. In: Spedicato E (ed) Algorithms for continuous optimisation: the state of the art. NATO ASI series. Kluwer, Dordrecht, pp 501–512
8. Dixon LCW, Maany Z (Feb. 1988) The echelon method for the solution of sparse sets of linear equations. Techn Report Numer Optim Centre, Hatfield Polytechnic NOC TR177
9. Duff IS, Anoli NI, Gould NIM, Reid JK (1987) The practical use of the Hellerman–Ranck P^4 algorithm and the P^5 algorithm of Erisman and others. Techn Report AERE Harwell CSS213
10. Fletcher R, Hall JAJ (1991) Ordering algorithms for irreducible sparse linear systems. Techn Report Dundee Univ NA/131
11. Geitner U, Utke J, Griewank A (1991) Automatic computation of sparse Jacobians by applying the method of Newsam and Ramsdell. In: Griewank A, Corliss GF (eds) Automatic Differentiation of Algorithms: Theory, Implementation, and Application, SIAM, Philadelphia, pp 161–172
12. Griewank A (1991) Direct calculation of Newton steps without accumulating Jacobians. In: Griewank A, Corliss GF (eds) Automatic Differentiation of Algorithms: Theory, Implementation, and Application, SIAM, Philadelphia, pp 126–137
13. Griewank A, Corliss GF (eds) (1991) Automatic differentiation of algorithms: theory, implementation, and application. SIAM, Philadelphia
14. Griewank A, Reese S (1991) On the calculation of Jacobian matrices by the Markowitz rule. In: Griewank A, Corliss GF (eds) Automatic Differentiation of Algorithms: Theory, Implementation, and Application, SIAM, Philadelphia, pp 126–135
15. Parkhurst SC (Dec. 1990) The evaluation of exact numerical Jacobians using automatic differentiation. Techn Report Numer Optim Centre, Hatfield Polytechnic, NOC TR224
16. Utke J (1996) Efficient Newton steps without Jacobians. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools, SIAM, Philadelphia, pp 253–264

Automatic Differentiation: Geometry of Satellites and Tracking Stations

DAN KALMAN
American University, Washington, DC, USA

MSC2000: 26A24, 65K99, 85-08

Article Outline

Keywords
Geometric Models
Sample Optimization Problems
  Minimum Range
  Direction Angles and Their Derivatives
  Design Parameter Optimization
Automatic Differentiation
  Scalar Functions and Operations
  Vector Functions and Operations
  Implementation Methods
Summary
See also
References

Keywords

Astrodynamics; Automatic differentiation; Satellite orbit; Vector geometry

Satellites are used in a variety of systems for communication and data collection. Familiar examples of these systems include satellite networks for broadcasting video programming, meteorological and geophysical data observation systems, the global positioning system (GPS) for navigation, and military surveillance systems. Strictly speaking, these are systems in which satellites are just one component, and in which there are other primary subsystems that have no direct involvement with satellites. Nevertheless, they will be referred to as *satellite systems* for ease of reference.

Simple geometric models are often incorporated in simulations of satellite system performance. Important operational aspects of these systems, such as the times when satellites can communicate with each other or with installations on the ground (e. g. *tracking stations*), depend on dynamics of satellite and station motion. The geometric models represent these motions, as well as constraints on communication or data collection. For example, the region of space from which an antenna on the ground can receive a signal might be modeled as a cone, with its vertex centered on the antenna and axis extending vertically upward. The antenna can receive a signal from a satellite only when the satellite is within the cone. Taking into account the motions of the satellite and the earth, the geometric model predicts when the satellite and tracking station can communicate.

Elementary optimization problems often arise in these geometric models. It may be of interest to determine the closest approach of two satellites, or when a satellite reaches a maximum elevation as observed from a tracking station, or the extremes of angular velocity and acceleration for a rotating antenna tracking a satellite. Optimization problems like these are formulated in terms of geometric variables, primarily distances and angles, as well as their derivatives with respect to time. The derivatives appear both in the optimization algorithms, as well as in functions to be optimized. One of the previously mentioned examples illustrates this. When a satellite is being tracked from the ground, the antenna often rotates about one or more axes so as to remain pointed at the satellite. The angular velocity and acceleration necessary for this motion are the first and second derivatives of variables expressed as angles in the geometric configuration of the antenna and satellite. Determining the extreme values of these *derivatives* is one of the optimization problems mentioned earlier.

Automatic differentiation is a feature that can be included in a computer programming language to simplify programs that compute derivatives. In the situation described above, satellite system simulations are developed as computer programs that include computed values for the distance and angle variables of interest. With automatic differentiation, the values of derivatives are an automatic by-product of the computation of variable values. As a result, the computer programmer does not have to develop and implement the computer instructions that go into calculating derivative values. As a specific example of this idea, consider again the rotating antenna tracking a satellite. Imagine that the programmer has worked out the proper equations to describe the angular position of the antenna at any time. The simulation also needs to compute values for the angular velocity and acceleration, the first and second derivatives of angular position. However, the programmer does not need to work out the proper equations for these derivatives. As soon as the equations for angular position are included in the computer program, the programming language provides for the calculation of angular velocity and acceleration *automatically*. That is the effect of automatic differentiation. Because the derivatives of geometric variables such as distances and angles can be quite involved, automatic differentiation results in computer programs that are much easier to develop, debug, and maintain.

The preceding comments have provided a brief overview of geometric models for satellite systems, as well as associated optimization problems and the use of automatic differentiation. The discussion will now turn to a more detailed examination of these topics.

Geometric Models

The geometric models for satellite systems are formulated in the context of three-dimensional real space. A conventional rectangular coordinate system is defined by mutually perpendicular x, y, and z axes. The earth is modeled as a sphere or ellipsoid centered at the origin (0, 0, 0), with the north pole on the positive z axis, and the equator in the xy plane. The coordinate axes are considered to retain a constant orientation rel-

ative to the fixed stars, so that the earth rotates about the z axis.

In this setting, tracking station and satellite locations are represented by points moving in space. Each such moving point is specified by a vector valued function $\mathbf{r}(t) = (x(t), y(t), z(t))$ where t represents time. Geometric variables such as angles and distances can be determined using standard vector operations:

$$
\begin{aligned}
&c(x, y, z) = (cx, cy, cz),\\
&(x, y, z) \pm (u, v, w) = (x \pm u, y \pm v, z \pm w),\\
&(x, y, z) \cdot (u, v, w) = xu + yv + zw,\\
&(x, y, z) \times (u, v, w)\\
&\quad = (yw - zv, zu - xw, xv - yu),\\
&\|(x, y, z)\| = \sqrt{x^2 + y^2 + z^2}\\
&\quad = \sqrt{(x, y, z) \cdot (x, y, z)}.
\end{aligned}
$$

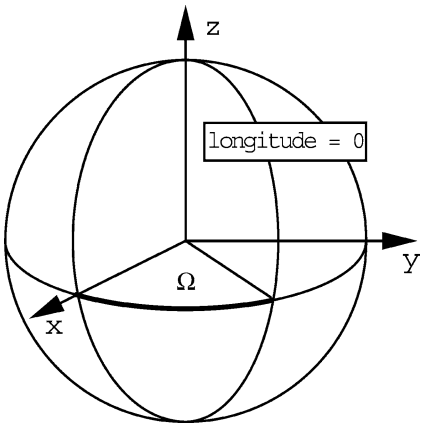
The distance between two points $\mathbf{r}$ and $\mathbf{s}$ is then given by $\| \mathbf{r} - \mathbf{s} \|$. The angle θ defined by rays from point $\mathbf{r}$ through points $\mathbf{p}$ and $\mathbf{q}$ is determined by

$$
\cos\theta = \frac{(\mathbf{p} - \mathbf{r}) \cdot (\mathbf{q} - \mathbf{r})}{\|\mathbf{p} - \mathbf{r}\| \cdot \|\mathbf{q} - \mathbf{r}\|}.
$$

A more complete discussion of vector operations, their properties, and geometric interpretation can be found in any calculus textbook; [9] is one example.

There are a variety of models for the motions of points representing satellites and tracking stations. The familiar conceptions of a uniformly rotating earth circled by satellites that travel in stable closed orbits is only approximately correct. For qualitative simulations of the performance of satellite systems, particularly at preliminary stages of system design, these models may be adequate. More involved models can take into account such effects as the asphericity of the gravitational field of the earth, periodic wobbling of the earth's axis of rotation, or atmospheric drag, to name a few. Modeling the motions of the earth and satellites with high fidelity is a difficult endeavor, and one that has been studied extensively. Good general references for this subject are [1,2,3,10].

For illustrative purposes, a few of the details will be presented for the simplest models, circular orbits



Automatic Differentiation: Geometry of Satellites and Tracking Stations, Figure 1
Earth rotation angle

around a spherical earth, uniformly spinning on a fixed axis. The radius of the earth will be denoted R_e.

As a starting point, the rotation of the earth can be specified by a single function of time, $\Omega(t)$, representing the angular displacement of the prime meridian from a fixed direction, typically the direction specified by the positive x axis (see Fig. 1.). At any time, the positive x axis emerges from the surface of the earth at some point on the equator. Suppose that at a particular time t, the point where the positive x axis emerges happens to be on the prime meridian, located at latitude 0 and longitude 0. Then $\Omega(t) = 0$ for that t. As time progresses, the prime meridian rotates away from the x axis, counter-clockwise as viewed by an observer above the north pole. The function Ω measures the angle of rotation, starting at 0 each time the prime meridian is aligned with the x axis, and increasing toward a maximum of 360° (2 π in radian measure) with each rotation of the earth. With a uniformly spinning earth, Ω increases linearly with t during each rotation.

Once Ω is specified, any terrestrial location given by a latitude ϕ, longitude λ, and altitude a can be transformed into absolute coordinates in space, according to the equations

$$\theta = \lambda + \Omega(t), \tag{1}$$

$$r = R_e + a, \tag{2}$$

$$x = r\cos\theta\cos\phi, \tag{3}$$

$$y = r \sin\theta \cos\phi, \tag{4}$$

$$z = r \sin\phi. \tag{5}$$

Holding latitude, longitude, and altitude constant, these equations express the position in space of a fixed location on the earth for any time, thereby modeling the point's motion. It is also possible to develop models for tracking stations that are moving on the surface of the earth, say on an aircraft or on a ship in the ocean. For example, if it is assumed that the moving craft is traveling at constant speed on a great circle arc or along a line of constant latitude, it is not difficult to express latitude and longitude as functions of time. In this case, the equations above reflect a dependence on t in λ and ϕ, as well as in Ω. A more complicated example would be to model the motion of a missile or rocket launched from the ground. This can be accomplished in a similar way: specify the trajectory in earth relative terms, that is, using latitude, longitude, and altitude, and then compute the absolute spatial coordinates (x, y, z). In each case, the rotation of the earth is accounted for solely by the effect of $\Omega(t)$.

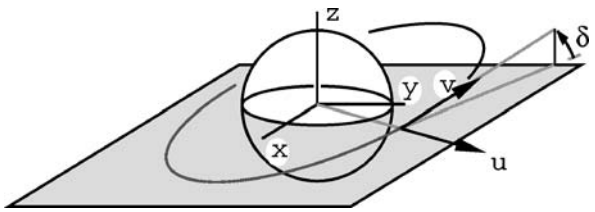
For a satellite in circular orbit, the position at any time is specified by an equation of the following form:

$$\mathbf{r}(t) = r[\cos(\omega t)\mathbf{u} + \sin(\omega t)\mathbf{v}].$$

In this equation, $\omega\, t$ is understood as an angle in radian measure for the sin and cos operations; r, ω, $\mathbf{u}$, and $\mathbf{v}$ are constants. The first, r is the length of the orbit circle's radius. It is equal to the sum of the earth's radius R_e and the satellite's altitude. The constant ω is the angular speed of the satellite. The satellite completes an orbit every $2\pi/\omega$ units of time, thus giving the *orbital period*. Both $\mathbf{u}$ and $\mathbf{v}$ are unit vectors: $\mathbf{u}$ is parallel to the initial position of the satellite; $\mathbf{v}$ is parallel to the initial velocity. See Fig. 2.

Mathematically, the equation above describes some sort of orbit no matter how the constants are selected. But not all of these are accurate descriptions of a free falling satellite in circular orbit. For one thing, $\mathbf{u}$ and $\mathbf{v}$ must be perpendicular to produce a circular orbit. In addition, there is a physical relationship linking r and ω. Assuming that the circular orbit follows Newton's laws of motion and gravitation, r and ω satisfy

$$\omega = Kr^{-\frac{3}{2}} \tag{6}$$



Automatic Differentiation: Geometry of Satellites and Tracking Stations, Figure 2
Circular orbit

where K is a physical constant that depends on both Newton's universal gravitational constant and the mass of the earth. Its numerical value also depends on the units of measurement used for time and distance. For units of hours and kilometers, the value of K is $2.27285 \cdot 10^6$. As this relationship shows, for a given altitude (and hence a given value of r), there is a unique angular speed at which a satellite will maintain a circular orbit. Equivalently, the altitude of a circular orbit determines the constant speed of the satellite, as well as the period of the satellite.

Generally, constants are chosen for a circular orbit based on some geometric description. Here is a typical approach. Assume that the initial position of the satellite is directly above the equator, with latitude 0, a given longitude, and a given altitude. In other words, assume that the initial position is in the plane of the equator, and so has a z coordinate of 0. (This is the situation depicted in Fig. 2.) Moreover, the initial heading of the satellite can be specified in terms of the angle it makes with the xy plane (which is the plane of the equator). Call that angle δ. From these assumptions we can determine values for the constants r, ω, $\mathbf{u}$, and $\mathbf{v}$ in the equation for $\mathbf{r}(t)$. Now the altitude for the orbit is constant, so the initial altitude determines r, as well as ω via equation (6). The initial latitude, longitude, and altitude also provide enough information to determine absolute coordinates (x, y, z) for the initial satellite position using equations (1)–(5). Accordingly, the unit vector $\mathbf{u}$ is given by

$$\mathbf{u} = \frac{(x, y, z)}{\|(x, y, z)\|}.$$

As already observed, the z coordinate of $\mathbf{u}$ will be 0. Finally, the unit vector $\mathbf{v}$ is determined from the initial position and heading. It is known that $\mathbf{v}$ make an angle

of δ with the xy plane, and hence makes an angle of $\pi/2 - \delta$ with the z axis. This observation can be expressed as the equation

$$\mathbf{v} \cdot (0, 0, 1) = \sin \delta.$$

It is also known that $\mathbf{v}$ must be perpendicular to $\mathbf{u}$, so

$$\mathbf{v} \cdot \mathbf{u} = 0.$$

Finally, since $\mathbf{v}$ is a unit vector,

$$\mathbf{v} \cdot \mathbf{v} = 1.$$

If $\mathbf{u} = (u_1, u_2, 0)$, then these three equations lead to $\mathbf{v} = (\pm u_2 \cos \delta, \mp u_1 \cos \delta, \sin \delta)$. The ambiguous sign can be resolved by assuming that the direction of orbit is either in agreement with or contrary to the direction of the earth's rotation. Assuming that the orbit is in the same direction as the earth's rotation, $\mathbf{v} = (-u_2 \cos \delta, u_1 \cos \delta, \sin \delta)$. The alternative possibility, that the satellite orbit opposes the rotation of the earth, is generally not practically feasible, so is rarely encountered.

The preceding paragraphs are intended to provide some insight about the mathematics used to describe the movement of satellites and terrestrial observers. Although the models presented here are the simplest ones available, they appear in the same general framework as much more sophisticated models. In particular, in any of these models, it is necessary to be able to compute instantaneous positions for satellites and terrestrial observers at any time during a simulation. Moreover, the use of vector algebra and geometry to set up the simple models is representative of the methods used in more complicated cases.

Sample Optimization Problems

Computer simulations of satellite system performance provide one tool for comparing alternative designs and making cost/benefit trade-offs in the design process. Optimization problems contribute both directly and indirectly. In many cases, system performance is characterized in terms of extreme values of variables: what is the maximum number of users that can be accommodated by a communications system? At a given latitude, what is the longest period of time during which at most three satellites can be detected from some point on the ground? In these examples, the optimization problems are directly connected with the goals of the simulation.

Optimization problems also arise indirectly as part of the logistics of the simulation software. This is particularly the case when a simulation involves events that trigger some kind of system response. Examples of such events include the passage of a satellite into or out of sunlight, reaching a critical level of some resource such as power or data storage, or the initiation or termination of radio contact with a tracking station. The detection of these events typically involves either root location or optimization. These processes are closely related: the root of an equation can usually be characterized as an extreme value of a variable within a suitable domain; conversely, optimization algorithms often generate candidate solutions by solving equations.

In many of these event identification problems, the independent variable is time. The objective functions ultimately depend on the geometric models for satellite and tracking station motion, and so can be formulated in terms of explicit functions of time. In contrast, some of the optimization problems that concern direct estimation of system performance seek to optimize that performance by varying design parameters. A typical approach to this kind of problem is to treat performance measures as functions of the parameters, where the values of the functions are determined through simulation. Both kinds of optimization are illustrated in the following examples.

Minimum Range

As a very simple example of an optimization problem, it is sometimes of interest to determine the closest approach of two orbiting bodies. Assume that a model has been developed, with $\mathbf{r}(t)$ and $\mathbf{s}(t)$ representing the positions at time t for the two bodies. The distance between them is then expressed as $\| \mathbf{r}(t) - \mathbf{s}(t) \|$. This is the objective function to be minimized. Observe that it is simply expressed as a composition of vector operations and the motion models for the two bodies.

A variation of this problem occurs when several satellites are required to stay in radio communication. In that case, an antenna on one satellite (at position A, say) may need to detect signals from two others (at positions B and C). In this setting, the measure of $\angle BAC$ is of interest. If the angle is wide, the antenna requires a correspondingly wide field of view. As the satellites proceed in their orbits, what is the maximum value of

the angle? Equivalently, what is the minimum value of the cosine of the angle? As before, the objective function in this minimization problem is easily expressed by applying vector operations to the position models for the satellites. If $\mathbf{a}(t)$, $\mathbf{b}(t)$, and $\mathbf{c}(t)$ are the position functions for the three satellites, then

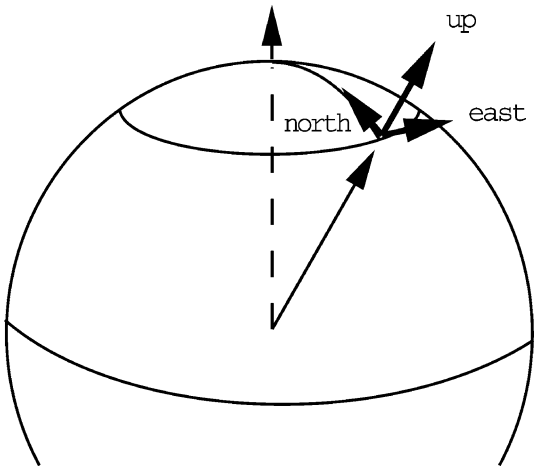
$$\cos \angle BAC = \frac{(\mathbf{b}-\mathbf{a})\cdot(\mathbf{c}-\mathbf{a})}{\|\mathbf{b}-\mathbf{a}\|\cdot\|\mathbf{c}-\mathbf{a}\|}.$$

This is a good example of combining vector operations with the models for satellite motion to derive the objective function in an optimization problem. The next example is similar in style, but mathematically more involved.

Direction Angles and Their Derivatives

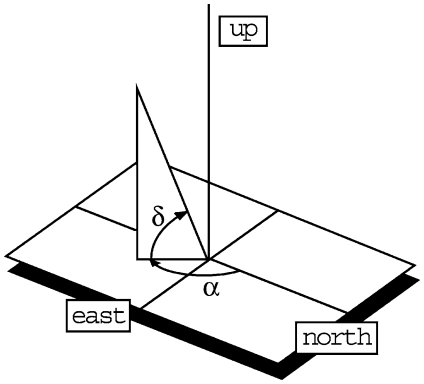
A common aspect of satellite system simulation is the representation of sensors of various kinds. The images that satellites beam to earth of weather systems and geophysical features are captured by sensors. Sensors are also used to locate prominent astronomical features such as the sun, the earth, and in some cases bright stars, in order to evaluate and control the satellite's attitude. Even the antenna used for communication is a kind of sensor. It is frequently convenient to define a coordinate system that is attached to a sensor, that is, define three mutually perpendicular axes which intersect at the sensor location, and which can be used as an alternate means to assign coordinates to points in space. Such a coordinate system is then used to describe the vectors from the sensor to other objects, and to model sensor sensitivity to signals arriving from various directions. With several different coordinate systems in use, it is necessary to transform information described relative to one system into a form that makes sense in the context of another system. This process also often involves what are called *direction angles*.

As a concrete example, consider an antenna at a fixed location on the earth, tracking a satellite in orbit. The coordinate system attached to the tracking antenna is the natural map coordinate system at that point on the earth: the local x and y axes point east and north, respectively, and the z axis points straight up (Fig. 3). The direction from the station to the satellite is expressed in terms of two angles: the elevation δ of the satellite above the local xy plane, and the compass angle α measured clockwise from north. (See Fig. 4.) To illustrate,



Automatic Differentiation: Geometry of Satellites and Tracking Stations, Figure 3
Local map coordinates

here is the meaning of an elevation of 30 degrees and a compass angle of 270 degrees. Begin by looking due north. Turn clockwise through 270 degrees, maintaining a line of sight that is parallel to the local xy plane. At that point you are looking due west. Now raise the line of sight until it makes a 30 degree angle with the local xy plane. This direction of view, with elevation 30 and compass angle 270 degrees, might thus be described as 30 degrees above a ray 270 degrees clockwise from due north. The elevation and compass angle are examples of direction angles. Looked at another way, if a spherical coordinate system is imposed on the local rectan-



Automatic Differentiation: Geometry of Satellites and Tracking Stations, Figure 4
Compass and elevation angles

gular system at the antenna, then every point in space is described by a distance and two angles. The angles are direction angles. Direction angles can be defined in a similar way for any local coordinate system attached to a sensor.

How are direction angles computed? In general terms, the basic idea is to define the local coordinate system in terms of moving vectors, and then to use vector operations to define the instantaneous value of direction angles. Here is a formulation for the earth based antenna. First, the local z axis points straight up. That means the vector from the center of the earth to the location of the antenna on the surface is parallel to the z axis. Given the latitude, longitude, and altitude of the antenna, its absolute position $\mathbf{r}(t) = (x, y, z)$ is computed using equations (1)–(5), as discussed earlier. The parallel unit vector is then given by $\mathbf{r}/\parallel \mathbf{r} \parallel$. To distinguish this from the global z axis, we denote it as the vector **up**. The vector pointing due east must be perpendicular to the up direction. It also must be parallel to the equatorial plane, and hence perpendicular to the global z axis. Using properties of vector cross products, a unit vector pointing east can therefore be expressed as

$$\mathbf{east} = \frac{(0,0,1) \times \mathbf{up}}{\|(0,0,1) \times \mathbf{up}\|}.$$

Finally, the third perpendicular vector is given by the cross product of the other two: **north** = **up** × **east**. Note that these vectors are defined as functions of time. At each value of t the earth motion model gives an instantaneous value for $\mathbf{r}(t)$, and that, in turn, determines the vectors **up**, **east**, and **north**.

Next, suppose that a satellite is included in the model, with instantaneous position $\mathbf{s}(t)$. The **view** vector from the antenna to the satellite is given by $\mathbf{v}(t) = [\mathbf{s}(t) - \mathbf{r}(t)]/\parallel \mathbf{s}(t) - \mathbf{r}(t) \parallel$. The goal is to calculate the direction angles α and δ for $\mathbf{v}$. Since δ measures the angle between $\mathbf{v}$ and the plane of **east** and **north**, the complimentary angle can be measured between $\mathbf{v}$ and **up**. This leads to the equation

$$\sin\delta = \mathbf{up} \cdot \mathbf{v}.$$

The angle α is found from

$$v_n = \mathbf{v} \cdot \mathbf{north}$$
$$v_e = \mathbf{v} \cdot \mathbf{east}$$

according to the equations

$$\cos\alpha = \frac{v_n}{\sqrt{v_n^2 + v_e^2}}$$
$$\sin\alpha = \frac{v_e}{\sqrt{v_n^2 + v_e^2}}.$$

These follow from the fact that the projection of $\mathbf{v}$ into the local xy plane is given by v_e **east** + v_n **north**.

In this example, direction angles play a role in several optimization problems. First, it may be of interest to predict the maximum value of δ as a satellite passes over the tracking station. This maximum value of elevation is an indication of how close the satellite comes to passing directly overhead, and may be used to determine whether communication will be possible between satellite and tracking station.

Additional optimization problems concern the derivatives of α and δ. In many designs, an antenna can turn about horizontal and vertical axes to point the center of the field of view in a particular direction. In order to stay pointed at a passing satellite, the antenna must be rotated on its axes so as to match the motion of the satellite, and α and δ specify exactly how far the antenna must be rotated about each axis at each time. However, there are mechanical limits on how fast the antenna can turn and accelerate. For this reason, during the time that the satellite is in view, the maximum values of the first and second derivatives of α and δ are of interest. If the first derivatives exceed the antenna's maximum turning speed, or if the second derivatives exceed the antenna's maximum acceleration, the antenna will not be able to remain pointed at the satellite.

Design Parameter Optimization

The preceding examples all involve simple kinds of optimization problems with objective functions depending only on time. There are also many situations in which system performance variables are optimized over some domain of design parameters. As one example of this, consider a system with a single satellite traveling in a circular orbit. Assume that the initial point of the orbit falls on the equator, with angle δ between the initial heading and the xy plane, as in Fig. 2. In this example, the object is to choose an optimal value of δ. The optimization problem includes several tracking stations on the ground that are capable of communicating with the

satellite. As it orbits, there may be times when the satellite cannot communicate with any of the tracking stations. At other times, one or more stations may be accessible. Over the simulation period, the total amount of time during which at least one tracking station is accessible will depend on the value of δ. It is this total amount of access time (denoted A) that is to be maximized.

In this problem, the objective function A is not given as a mathematical expression involving the variable δ. An appropriate simulation can be created to compute A for any particular δ of interest. This can then be used in conjunction with an optimization algorithm, with the simulation executed each time it is necessary to calculate $A(\delta)$.

The preceding example is a simple one, and the execution time required to compute $A(\delta)$ is small. For more complicated situations, each execution of the simulation can require a significant amount of time. In these cases, it may be more practical to use some sort of interpolation scheme. The idea would be to run the simulation for some values of the parameter(s), and to interpolate between these values as needed during the optimization process.

In some situations, there is a resource allocation problem that can add yet another level of complexity to optimizing system performance. For example, if there are several satellites that must compete for connection time with the various tracking stations, just determining how to assign the tracking stations to the satellites is not a simple matter. In this situation, there may be one kind of optimization problem performed during the simulation to make the resource allocations, and then a secondary optimization that considers the effect of changing system design parameters. An example of this kind of problem is described in detail in [6].

The preceding examples have been provided to illustrate the kinds of optimization problems that arise in simulations of satellite systems. Although there has been very little discussion of methods to solve these optimization problems, it should be clear that standard methods apply, especially in the cases for which the independent variable is time. In that context, the ability to compute derivatives relative to time for the objective function is of interest. In addition, it sometimes occurs that the objective function is, itself, defined as a derivative of some geometric variable, providing another motivation for computing derivatives. The next topic of discussion concerns the use of automatic differentiation for computing the desired derivatives.

Automatic Differentiation

Automatic differentiation refers to a family of techniques for automatically computing derivatives as a byproduct of function evaluation. A survey of different approaches and applications can be found in [5] and in-depth treatment appears in [4]. For the present discussion, attention will be restricted to what is called the *forward mode of automatic differentiation*, and in particular, the approach described in [8]. In this approach, to provide automatic calculation of the first m derivatives of real valued expressions of a single variable x, an algebraic system is defined consisting of real $m+1$ tuples, to which are extended the familiar binary operations and elementary functions generally defined on real variables. For concreteness, m will be assumed to be 3 below, but the discussion can be generalized to other values in an obvious way.

With $m = 3$, the objects manipulated by the automatic differentiation system are 4-tuples. The idea is that each 4-tuple represents the value of a function and its first 3 derivatives, and that the operations on tuples preserve this interpretation. Thus, if $a = (a_0, a_1, a_2, a_3)$ consists of the value of $f(t)$, $f'(t)$, $f''(t)$, and $f'''(t)$ at some t, and if $b = (b_0, b_1, b_2, b_3)$ is similarly defined for function g, then the product ab that is defined for the automatic differentiation system will consist of the value at t of fg and its first 3 derivatives. Similarly, the extension of the squareroot function to 4-tuples is so contrived that $\sqrt{a}$ will consist of the value of $\sqrt{f(t)}$ and its first 3 derivatives.

In the preceding remarks, the functions f and g are assumed to be real valued, but similar ideas work for vector valued functions. The principle difference is this: when $f(t)$ is a vector, then so are its derivatives, and the a_i referred to above are then vectors rather than scalars. In addition, for vector valued functions, there are different operations than for scalar valued functions. For example, vector functions may be combined with a dot product, as opposed to the conventional product of real scalars, and while the squareroot operation is not defined for vector valued functions, the norm operation $\| f(t) \|$ is.

In an automatic differentiation system built along these lines, there must be some functions that are evaluated directly to produce 4-tuples. For example, the constant function with value c can be evaluated directly to produce the tuple $(c, 0, 0, 0)$, and the identity function $I(t) = t$ can be evaluated directly to produce $(t, 1, 0, 0)$. For geometric satellite system simulations, it is also convenient to provide direct evaluation of tuples for the motion models. For example, let $\mathbf{r}(t)$ be the position vector for a tracking station, as developed in equations (1)–(5). It is a simple matter to work out appropriate formulas for the first three derivatives of $\mathbf{r}(t)$, each of which is also a vector. This is included in the automatic differentiation system so that when a particular value of t is given, the motion model computes the 4-tuple $(\mathbf{r}(t), \mathbf{r}'(t), \mathbf{r}''(t), \mathbf{r}'''(t)$. A similar arrangement is made for every moving object represented in the simulation, including satellites, tracking stations, ships, aircraft, and so on.

Here is a simple example of how automatic differentiation is used. In the earlier discussion of optimization problems, there appeared the following equation:

$$\cos \angle BAC = \frac{(\mathbf{b} - \mathbf{a}) \cdot (\mathbf{c} - \mathbf{a})}{\|\mathbf{b} - \mathbf{a}\| \cdot \|\mathbf{c} - \mathbf{a}\|}.$$

Using automatic differentiation, $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ would be 4-tuples, each consisting of four vectors. These are produced by the motion models for three satellites, as the values of position and its first three derivatives at a specific time. The operations used in the equation, vector difference, dot product, and norm, as well as scalar multiplication and division, are all special modified operations that work directly on 4-tuples. The end result is also a 4-tuple, consisting of the cosine of angle BAC, as well as the first three derivatives of that function, all at the specified value of t. As a result, the programmer can obtain computed values for the derivatives of the function without explicitly coding equations for these derivatives. More generally, after defining appropriate 4-tuples for all of the motion models, the programmer automatically obtains derivatives for any function that is defined by operating on the motion models, just by defining the operations. No explicit representation of the derivatives of the operations is needed. Some details of how the system works follow.

Scalar Functions and Operations

Consider first operations which apply to scalars. There are two basic types: binary operations $(+, -, \times, \div)$ and elementary functions (squareroot, exponential and logarithm, trigonometric functions, etc.). These operations must be defined for the 4-tuples of the automatic differentiation system in such a way that derivatives are correctly propagated.

The definition for multiplication will illustrate the general approach for binary operations. Suppose that (a, b, c, d) and (u, v, w, x) are two 4-tuples of scalars. They represent values of functions and their derivatives, say, $(a, b, c, d) = (f(t), f'(t), f''(t), f'''(t))$ and $(u, v, w, x) = (g(t), g'(t), g'''(t), g'''(t))$. The product is supposed to give $((fg)(t), (fg)'(t), (fg)''(t), (fg)''(t))$. Each of these derivatives can be computed using the derivatives of f and g.

$$\begin{aligned}
(fg)(t) &= f(t)g(t),\\
(fg)'(t) &= f'(t)g(t) + f(t)g'(t),\\
(fg)''(t) &= f''(t)g(t) + 2f'(t)g'(t) + f(t)g''(t),\\
(fg)'''(t)) &= f'''(t)g(t) + 3f''(t)g'(t)\\
&\qquad + 3f'(t)g''(t) + f(t)g'''(t)).
\end{aligned}$$

On the right side of each equation, now substitute the entries of (a, b, c, d) and (u, v, w, x).

$$\begin{aligned}
(fg)(t) &= au,\\
(fg)'(t) &= av + bu,\\
(fg)''(t) &= aw + 2bv + cu,\\
(fg)'''(t)) &= ax + 3bw + 3cv + du.
\end{aligned}$$

This shows that 4-tuples must be multiplied according to the rule

$$\begin{aligned}
&(a, b, c, d)(u, v, w, x)\\
&= (au, av + bu, aw + 2bv + cu,\\
&\qquad ax + 3bw + 3cv + du).
\end{aligned}$$

For addition, subtraction, and division a similar approach can be used. All that is required is that successive derivatives of the combination of f and g be expressed in terms of the derivatives of f and g separately. Replacing these derivatives with the appropriate components of (a, b, c, d) and (u, v, w, x) produces the desired formula for operating on 4-tuples.

To define the operation on a 4-tuple of an elementary function, a similar approach will work. Consider defining how a function h should apply to a 4-tuple $(a, b, c, d) = (f(t), f'(t), f''(t), f'''(t))$. This time, the desired end result should contain derivatives for the composite function $h \circ f$, and so should have the form $((h \circ f)(t), (h \circ f)'(t), (h \circ f)''(t), (h \circ f)'''(t))$ The derivative of $h \circ f$ is given by $h'(f(t))\, f'(t)$, which becomes $h'(a)\, b$ after substitution. Similar computations produce expressions for the second and third derivatives:

$$\begin{aligned}&(h \circ f)''(t)\\ &= h''(f(t))f'(t)^2 + h'(f(t))f''(t)\\ &= h''(a)b^2 + h'(a)c\end{aligned}$$

and

$$\begin{aligned}&(h \circ f)'''(t)\\ &= h'''(f(t))f'(t)^3 + 3h''(f(t))f'(t)f''(t)\\ &\quad + h'(f(t))f'''(t)\\ &= h'''(a)b^3 + 3h''(a)bc + h'(a)d.\end{aligned}$$

These results lead to

$$\begin{aligned}&h(a, b, c, d)\\ &= (h(a), h'(a)b, h''(a)b^2 + h'(a)c,\\ &\qquad h'''(a)b^3 + 3h''(a)bc + h'(a)d).\end{aligned}$$

As an example of how this is applied, let $h(t) = e^t$. Then $h(a) = h'(a) = h''(a) = h'''(a) = e^a$ so

$$\begin{aligned}&e^{(a,b,c,d)}\\ &= (e^a, e^a b, e^a b^2 + e^a c, e^a b^3 + 3e^a bc + e^a d)\\ &= e^a(1, b, b^2 + c, b^3 + 3bc + d).\end{aligned}$$

Other functions are a little more complicated, but the overall approach is generally correct.

The preceding discussion indicates how operations on 4-tuples would be built into an automatic differentiation system. However, the user of such a system would simply apply the operations. So, if an appropriate definition has been provided for $\Omega(t)$ as discussed earlier, along with the derivatives, the program would compute a 4-tuple for Ω and its derivatives at a particular time. Say that is represented in the program by the variable W. If the program later includes the call $\sin(W)$, the result would be a 4-tuple with values for $\sin(\Omega(t))$, and the first three derivatives.

Vector Functions and Operations

The approach for vector functions is basically the same as for scalar functions. The only modification that is needed is to recognize that the components of 4-tuples are now vectors. Because the rules for computing derivatives of vector operations are so similar to those for scalar operations, there is little difference in the appearance of the definitions. For example, here is the definition for the dot product of two 4-tuples, whose components are vectors:

$$\begin{aligned}&(a, b, c, d) \cdot (u, v, w, x)\\ &= (a \cdot u, a \cdot v + b \cdot u, a \cdot w + 2b \cdot v + c \cdot u,\\ &\qquad a \cdot x + 3b \cdot w + 3c \cdot v + d \cdot u).\end{aligned}$$

The formulation for vector cross product is virtually identical, as is the product of a scalar 4-tuple with a vector 4-tuple. For the vector norm, simply define

$$\|(a, b, c, d)\| = \sqrt{(a, b, c, d) \cdot (a, b, c, d)}.$$

Since both dot product of vector 4-tuples and square-root of scalar 4-tuples have already been defined in the automatic differentiation system, this equation will propagate derivatives correctly.

With a full complement of scalar and vector operations provided by the automatic differentiation system, all of the geometric variables discussed in previous examples can be included in a computer program, with derivatives generated automatically. As a particular case, reconsider the discussion earlier of computing elevation δ and compass angle α for a satellite as viewed from a tracking station. Assuming that r and s have been defined as 4-tuples for the vector positions of that station and satellite, the following fragment of pseudocode would carry out the computations described earlier:

up	= r/norm(r)
east	= cross(pole, up)
east	= east/norm(east)
north	= cross(up, east)
v	= (s—r)/norm(s—r)
vn	= dot(v, north)
ve	= dot(v, east)
vu	= dot(v, up)
delta	= asin(vu)
alpha	= atan2(ve, vn)

Executed in an automatic differentiation system, this code produces not just the instantaneous values of the angles α and δ, but their first three derivatives, as well. The programmer does not need to derive and code explicit equations for these derivatives, a huge savings in this problem. And all of the derivative information is useful. Recall that the first and second derivatives are of interest for their physical interpretations as angular velocities and accelerations. The third derivatives are used in finding the maximum values of the second derivatives (accelerations).

Implementation Methods

One of the simplest ways to implement automatic differentiation is to use a language like C^{++} that supports the definition of abstract data types and operator overloading. Then the automatic differentiation system would be implemented as a series of data types and operations, and included as part of the code for a simulation. A discussion of one such implementation can be found in [7].

Another approach is to develop a preprocessor that automatically augments code with the steps needed to compute derivatives. With such a system, the programmer develops code in a conventional language such as FORTRAN, with some additional features that control the application of automatic differentiation. Next, this code is operated on by the preprocessor, producing a modified program. That is then compiled and executed in the usual way. Examples of this approach can be found in [5].

Summary

Geometric models are very useful in representing the motions of satellites and terrestrial objects in simulations of satellite systems. These models are defined in terms of vector operations, which permit the convenient formulation of equations for geometric constructs such as distances and angles arising in the satellite system configuration. Equations which specify instantaneous positions in space of moving objects are a fundamental component of the geometric modeling framework.

Optimization problems occur in this framework in two guises. First, there are problems in which the objective functions are directly defined as features of the geometric setting. An example of this would be to find the minimum distance between two satellites. Second, measures of system performance are derived via simulation as a function of design parameters, and these measures are optimized by varying the parameters. An example of this kind of problem would be to seek a particular orbit geometry in order to maximize the total amount of time a satellite has available to communicate with a network of tracking stations.

Automatic differentiation is a feature of an environment for implementing simulations as computer programs. In an automatic differentiation system, the equations which define values of variables automatically produce the values of the derivatives, as well. In the geometric models of satellite systems, derivatives of some variables are of intrinsic interest as velocities and accelerations. Derivatives are also useful in solving optimization problems.

Automatic differentiation can be provided by replacing single operands with tuples, representing the operands and their derivatives. For some tuples, the derivatives must be explicitly provided. This is the case for the motion models. For tuples representing combinations of the motion models, the derivatives are generated automatically. These combinations can be defined using any of the supported operations provided by the automatic differentiation system, typically including the operations of scalar and vector arithmetic, as well as scalar functions such as exponential, logarithmic, and trigonometric functions. Languages which support abstract data types and operator overloading are a convenient setting for implementing an automatic differentiation system.

See also

- ► Automatic Differentiation: Calculation of the Hessian
- ► Automatic Differentiation: Calculation of Newton Steps
- ► Automatic Differentiation: Introduction, History and Rounding Error Estimation
- ► Automatic Differentiation: Parallel Computation
- ► Automatic Differentiation: Point and Interval
- ► Automatic Differentiation: Point and Interval Taylor Operators

► Automatic Differentiation: Root Problem and Branch Problem
► Nonlocal Sensitivity Analysis with Automatic Differentiation

References

1. Bate RR, Mueller DD, White JE (1971) Fundamentals of astrodynamics. Dover, Mineola, NY
2. Battin RH (1987) An introduction to the mathematics and methods of astrodynamics. AIAA Education Ser. Amer. Inst. Aeronautics and Astronautics, Reston, VA
3. Escobal PR (1965) Methods of orbit determination. R.E. Krieger, Huntington, NY
4. Griewank A (2000) Evaluating derivatives: Principles and techniques of algorithmic differentiation. SIAM, Philadelphia
5. Griewank A, Corliss GF (eds) (1995) Automatic differentiation of algorithms: Theory, implementation, and application. SIAM, Philadelphia
6. Kalman D (1999) Marriages made in the heavens: A practical application of existence. Math Magazine 72(2):94–103
7. Kalman D, Lindell R (1995) Automatic differentiation in astrodynamical modeling. In: Griewank A and Corliss GF (eds) Automatic differentiation of algorithms: Theory, implementation, and application. SIAM, Philadelphia, pp 228–241
8. Rall LB (Dec. 1986) The arithmetic of differentiation. Math Magazine 59(5):275–282
9. Thomas GB Jr, Finney RL (1996) Calculus and analytic geometry, 9th edn. Addison-Wesley, Reading, MA
10. Wertz JR (ed) (1978) Spacecraft attitude determination and control. Reidel, London

Automatic Differentiation: Introduction, History and Rounding Error Estimation

MASAO IRI, KOICHI KUBOTA
Chuo University, Tokyo, Japan

MSC2000: 65D25, 26A24

Article Outline

Keywords
Introduction
  Algorithms
  Complexity
History
Estimates of Rounding Errors
See also
References

Keywords

Differentiation; System analysis; Error analysis

Introduction

Most numerical algorithms for analyzing or optimizing the performance of a nonlinear system require the partial derivatives of functions that describe a mathematical model of the system. The *automatic differentiation* (abbreviated as AD in the following), or its synonym, *computational differentiation*, is an efficient method for computing the numerical values of the derivatives. AD combines advantages of numerical computation and those of symbolic computation [2,4].

Given a vector-valued function $\mathbf{f}$: $\mathbf{R}^n \to \mathbf{R}^m$:

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \equiv \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{pmatrix} \tag{1}$$

of n variables represented by a big program with hundreds or thousands of program statements, one often had encountered (before the advent of AD) some difficulties in computing the partial derivatives $\partial f_i/\partial x_j$ with conventional methods (as will be shown below). Now, one can successfully differentiate them with AD, deriving from the program for $\mathbf{f}$ another program that efficiently computes the numerical values of the partial derivatives.

AD is entirely different from the well-known numerical approximation with quotients of finite differences, or *numerical differentiation*. The quotients of finite differences, such as $(f(x + h) - f(x))/h$ and $(f(x + h) - f(x - h))/2h$, approximate the derivative $f'(x)$, where truncation errors are of $O(h)$ and $O(h^2)$, respectively, but there is an insurmountable difficulty to compute better and better approximation. For, although an appropriately small value of h is chosen, it may fail to compute the values of the function when $x \pm h$ is out of the domain of f, and, furthermore, the effect of rounding errors in computing the values of the functions is of problem.

AD is also different from symbolic differentiation with a symbolic manipulator. The symbolic differentiation derives the expressions of the partial derivatives rather than the values. The mathematical model of a large scale system may be described in thousands of program statements so that it becomes very difficult to handle whole of them with an existing symbolic manipulator. (There are a few manipulators combined with AD, which can handle such large scale programs. They should be AD regarded as a symbolic manipulator.)

Example 1 Program 1 computes an output value y_1 as a composite function f_1 for given input values $x_1 = 2$, $x_2 = 3$, $x_3 = 4$:

$$y_1 = f_1(x_1, x_2, x_3) = \frac{x_1(x_2 - x_3)}{\exp(x_1(x_2 - x_3)) + 1} . \qquad (2)$$

```
IF (x2.le.x3)
  THEN y1 = x1(x2 − x3)
  ELSE y1 = x1(x2 + x3)
ENDIF
y1 = y1/(exp(y1) + 1).
```

Automatic Differentiation: Introduction, History and Rounding Error Estimation, Program 1
Example

The execution of this program is traced by a sequence of assignment statements (Program 2).

```
y1 ← x1(x2 − x3),
y1 ← y1/(exp(y1) + 1).
```

Automatic Differentiation: Introduction, History and Rounding Error Estimation, Program 2
Program 1 expanded to straight line program for the specified input values

A set of unary or binary arithmetic operators (+, −, ∗, /) and elementary transcendental functions (exp, log, sin, cos, …) that may be used in the programs will be called *basic operations*. (Some special operations such as those generating 'constant' and 'input' are also to be counted among basic operations.) Program 2 can be expanded into a sequence of assignment statements each of whose right side has only one basic operation (Program 3), where $z_1, \ldots, z_s$ are temporary variables ($s = 2$ for this example).

```
1 | z1 ← x2 − x3,
2 | z1 ← x1 ∗ z1,
3 | z2 ← exp(z1),
4 | z2 ← z2 + 1,
5 | z1 ← z1/z2.
```

Automatic Differentiation: Introduction, History and Rounding Error Estimation, Program 3
Expanded history of execution with each line having only one basic operation

Moreover, it is useful to rewrite Program 3 into a sequence of *single assignment* statements, in which each variable appears at most once in the left sides (Program 4), hence, '←' can be replaced by ' = '.

```
1 | v1 ← x2 − x3,
2 | v2 ← x1 ∗ v1,
3 | v3 ← exp(v2),
4 | v4 ← v3 + 1.,
5 | v5 ← v2/v4,
```

Automatic Differentiation: Introduction, History and Rounding Error Estimation, Program 4
Computational process

The sequence is called a *computational process*, where the additional variables $v_1, \ldots, v_5$ are called *intermediate variables* that keep the intermediate results. A graph called a *computational graph*, $G = (V, A)$, may be used to represent the process (see Fig. 1).

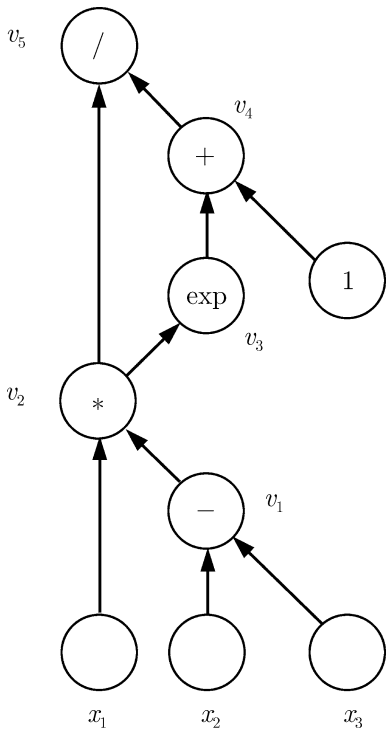
Algorithms

There are two modes for AD algorithm, *forward mode* and *reverse mode*. The forward mode is to compute $\partial y_i/\partial x_j$ ($i = 1, \ldots, m$) for a fixed j, whereas the reverse mode is to compute $\partial y_i/\partial x_j$ ($j = 1, \ldots, n$) for a fixed i.

The forward mode corresponds to tracing an expanded program such as Program 3 in the natural order. Assume that execution of the kth assignment in the program is represented as

$$z_c \leftarrow \psi_k(z_a, z_b) . \qquad (3)$$

When the values of both $\partial z_a/\partial x_j$ and $\partial z_b/\partial x_j$ are known, $\partial z_c/\partial x_j$ can be computed by applying the chain rule of



Automatic Differentiation: Introduction, History and Rounding Error Estimation, Figure 1
Computational graph

differentiation to (3):

$$\frac{\partial z_c}{\partial x_j} \leftarrow \frac{\partial \psi_k}{\partial z_a}\frac{\partial z_a}{\partial x_j} + \frac{\partial \psi_k}{\partial z_b}\frac{\partial z_b}{\partial x_j}\,. \tag{4}$$

$\partial\psi_k/\partial z_a$ and $\partial\psi_k/\partial z_b$ are called *elementary partial derivatives*, and are computed by Table 1 for various ψ_k.

Introducing new variables $\overline{z}_1, \ldots, \overline{z}_s$, $\overline{x}_1, \ldots, \overline{x}_n$ corresponding to $\partial z_1/\partial x_j$, …, $\partial z_s/\partial x_j$, $\partial x_1/\partial x_j$, …, $\partial x_n/\partial x_j$, respectively, and initializing $\overline{x}_k \leftarrow 0$ $(1 \le k \le n, k \ne j)$ and $\overline{x}_j \leftarrow 1$, we may express (4) as

$$\overline{z}_c \leftarrow \frac{\partial \psi_k}{\partial z_a}\overline{z}_a + \frac{\partial \psi_k}{\partial z_b}\overline{z}_b\,. \tag{5}$$

Thus, we can write down the whole program for the forward mode as shown in Program 5.

The reverse mode corresponds to tracing a computational process such as Program 4 backwards. The kth *computational step*, i. e., execution of the kth assignment in the program, can be written in general as

$$v_k = \psi_k(u_{k1}, u_{k2})|_{u_{k1}=v_{\alpha_k}, u_{k2}=v_{\beta_k}}\,, \tag{6}$$

Automatic Differentiation: Introduction, History and Rounding Error Estimation, Table 1
Elementary partial derivatives

$z_c = \psi(z_a, z_b)$	$\frac{\vartheta\psi}{\vartheta z_a}$	$\frac{\vartheta\psi}{\vartheta z_b}$
$z_c = z_a \pm z_b$	1	± 1
$z_c = z_a \cdot z_b$	z_b	z_a
$z_c = z_a/z_b$	$1/z_b$	$-z_a/z_b^2$ $(= -z_c/z_b)$
$z_c = \sqrt{z_a}$	$\frac{1}{2}/\sqrt{z_a} (= \frac{1}{2}/z_c)$	–
$z_c = \log(z_a)$	$1/z_a$	–
$z_c = \exp(z_a)$	$\exp(z_a) (= z_c)$	–
$z_c = \cos(z_a)$	$-\sin(z_a)$	–
$z_c = \sin(z_a)$	$\cos(z_a)$	–
⋮	⋮	⋮

	Initialization
	$\overline{x}_j \leftarrow 1$,
	$\overline{x} \leftarrow 0$ $(1 \le k \le n,\ k \ne j)$,
	Forward algorithm:
1	$z_1 \leftarrow x_2 - x_3$,
1′	$\overline{z}_1 \leftarrow 1 * \overline{x}_2 - 1 * \overline{x}_3$,
2′	$\overline{z}_1 \leftarrow z_1 * \overline{x}_1 + x_1 * \overline{z}_1$,
2	$z_1 \leftarrow x_1 * z_1$,
3	$z_2 \leftarrow \exp(z_1)$,
3′	$\overline{z}_2 \leftarrow z_2 * \overline{z}_1$,
4	$z_2 \leftarrow z_2 + 1$,
4′	$\overline{z}_2 \leftarrow 1 * \overline{z}_2$,
5	$z_1 \leftarrow z_1/z_2$,
5′	$\overline{z}_1 \leftarrow (1/z_2) * \overline{z}_1 - (z_1/z_2) * \overline{z}_2$

Automatic Differentiation: Introduction, History and Rounding Error Estimation, Program 5
Forward mode program for differentation

where $u_{k,1}$ and $u_{k,2}$ are formal parameters, v_{α_k} and v_{β_k} are real parameters representing some of $x_1, \ldots, x_n, v_1, \ldots, v_{k-1}$. If ψ_k is unary, $u_{k,2}$ and v_{β_k} are omitted. Let r be the total number of computational steps. In Program 4, we have $r = 5$ and, for $k = 2$, e. g., $\psi_2 =$ '$*$', $v_{\alpha_2} = x_1$ and $v_{\beta_2} = v_1$.

The total differentiation of (6) yields the relations among $dx_1, \ldots, dx_n, dv_1, \ldots, dv_r$ such as follows:

$$\mathrm{d}v_k = \frac{\partial \psi_k}{\partial u_{k,1}}\mathrm{d}v_{\alpha_k} + \frac{\partial \psi_k}{\partial u_{k,2}}\mathrm{d}v_{\beta_k} \quad (k = 1, \ldots, r)\,. \tag{7}$$

The computation of the partial derivatives of the ith component of the final result $y_i = f_i(x_1, \ldots, x_n)$ in (1)

with respect to $x_1, \ldots, x_n$ is that of the coefficients of the relation among $dx_1, \ldots, dx_n$ and dy_i.

Here, new variables $\overline{x}_1, \ldots, \overline{x}_n, \overline{v}_1, \ldots, \overline{v}_r$ are introduced for the computation of those coefficients. Without loss of generality, we may assume that the value of y_i is computed at v_r. After Program 4 is executed in the natural order with all the information on intermediate results preserved, these new variables are initialized as $\overline{x}_j \leftarrow 0$ $(j = 1, \ldots, n)$, $\overline{v}_k \leftarrow 0$ $(k = 1, \ldots, r-1)$ and $\overline{v}_r \leftarrow 1$, then the relation

$$\mathrm{d}y = \sum_{j=1}^{n} \overline{x}_j \mathrm{d}x_j + \sum_{k=1}^{r} \overline{v}_k \mathrm{d}v_k \tag{8}$$

holds. Secondly, $dv_r, dv_{r-1}, \ldots, dv_k$ can be eliminated from (8) in this order by modifying

$$\overline{v}_{\alpha_k} \leftarrow \overline{v}_{\alpha_k} + \overline{v}_k \frac{\partial \psi_k}{\partial v_{\alpha_k}}, \tag{9}$$

$$\overline{v}_{\beta_k} \leftarrow \overline{v}_{\beta_k} + \overline{v}_k \frac{\partial \psi_k}{\partial v_{\beta_k}}. \tag{10}$$

Finally, if we change k in the reverse order, i. e., $k = r$, $r-1$, …, 1, we can successfully eliminate all the dv_k $(k = 1, \ldots, r)$ to have

$$\mathrm{d}y = \sum_{j=1}^{n} \overline{x}_j \mathrm{d}x_j . \tag{11}$$

The final coefficient $\overline{x}_j$ indicates the value of $\partial f_i / \partial x_j$ $(j = 1, \ldots, n)$. Program 6 in which modifications (9) and (10) are embedded is the reverse mode program, which is sometimes called the *adjoint program* of Program 4.

It is easy to extend the algorithms for computing a linear combination of the column vectors of the Jacobian matrix J with the forward mode, and a linear combination of the row vectors of J with the reverse mode.

Complexity

It is proved that, for a constant C ($= 4 \sim 6$, varying under different computational models), the total operation count for $\partial y_i / \partial x_j$'s with a fixed j in the forward mode algorithm, as well as that for $\partial y_i / \partial x_j$'s with a fixed i in the reverse mode algorithm, is at most $C \cdot r$, i. e., in $O(r)$. Roughly speaking, r is proportional to the execution time T of the given program, so that the time complexity is in $O(T)$. Furthermore, we have to repeat such computation n times to get all the required partial derivatives by the forward mode, and m times by the reverse mode. What should be noted here is that the computational time of the forward or reverse mode algorithm for one set of derivatives does not depend on m or n but only on r.

	Forward sweep:
	(insert Program 4 here)
	Initialization: $(n = 3, r = 5)$
	$\overline{x}_j \leftarrow 0$ $(j = 1, \ldots, n)$,
	$\overline{v}_k \leftarrow 0$ $(k = 1, \ldots, r-1)$,
	$\overline{v}_r \leftarrow 1$,
	Reverse elimination:
$5''$	$\overline{v}_2 \leftarrow \overline{v}_2 + (1/v_4) * \overline{v}_5$,
	$\overline{v}_4 \leftarrow \overline{v}_4 + (-v_5/v_4) * \overline{v}_5$,
$4''$	$\overline{v}_3 \leftarrow \overline{v}_3 + 1 * \overline{v}_4$,
$3''$	$\overline{v}_2 \leftarrow \overline{v}_2 + v_3 * \overline{v}_3$,
$2''$	$\overline{x}_1 \leftarrow \overline{x}_1 + v_1 * \overline{v}_2$,
	$\overline{v}_1 \leftarrow \overline{v}_1 + x_1 * \overline{v}_2$,
$1''$	$\overline{x}_2 \leftarrow \overline{x}_2 + 1 * \overline{v}_1$,
	$\overline{x}_3 \leftarrow \overline{x}_3 + (-1) * \overline{v}_1$.

Automatic Differentiation: Introduction, History and Rounding Error Estimation, Program 6
Reverse mode program

Denoting the spatial complexity of the original program by S, that of the forward mode algorithm is in $O(S)$. However, the spatial complexity of the reverse mode is in $O(T)$, since the reverse mode requires a history of the forward sweep recorded in storage whose size is in $O(T)$.

A rough sketch of the proof is as follows. Without loss of generality, assume that the given program is expanded into a sequence of single assignment statements with a binary or unary basic operation as shown in Program 3 and 4. The operation count for computing the elementary partial derivatives (Table 1) is bounded by a constant. The additional operation count for modifying $\overline{v}_k$'s and $\overline{x}_j$'s in (5), (9) and (10) is also bounded since there are at most two additions and two multiplications. There are r operations in the original program, so that the total operation count in the forward mode algorithm as well as that in the reverse mode algorithm is in $O(r)$.

Note that the computational complexities of the forward mode and the reverse mode may not be optimal, but at least one can compute them in time proportional

to that for the computation of the given original program.

One can extend the AD algorithms to compute higher derivatives. In particular, it is well known how to compute a truncated Taylor series to get arbitrarily higher-order derivatives of a function with one variable [14]. One may regard a special function such as a Bessel function or a block of several arithmetic operations, such as the inner product of vectors, as a basic operation if the corresponding elementary partial derivatives are given with computational definitions. An analogy is pointed out in [7] between the algorithms for the partial derivatives and those of the computation of the shortest paths in an acyclic graph.

It has also been pointed out that there may be pitfalls in the derived program with AD. For example, a tricky program

IF (x.ne.1.0)
THEN $y = x*x$
ELSE $y = 1.0 + (x - 1.0) * b$
ENDIF

can compute the value of a function $f(x) = x^2$ correctly for all x. However, the derived program fails to compute $f'(1.0)$, because the differentiation of the second assignment with respect to x is not 2.0 but b. Thus conditional branches (or equations equivalent to conditional branches) should be carefully dealt with.

History

A brief history of AD is as follows. There were not a few researchers in the world who had more or less independently proposed essentially the same algorithms.

The first publication on the forward mode algorithm was presumably the paper by R.E. Wengert in 1964 [16]. After 15 years, books were published by L.B. Rall [14] and by H. Kagiwada et al. [9] which have been influential on the numerical-computational circle. The practical and famous software system for the forward mode automatic differentiation was Pascal-SC, and its descendants Pascal-XSC and C-XSC are popular now.

The paper [13] might be the first to propose systematically the reverse mode algorithm. But there are many ways through which to approach the reverse mode algorithm. In fact, it is related to Lagrange multipliers, error analysis, generation of adjoint systems, reduction of computational complexity of computing the gradient, neural networks, etc. Of course, the principles of the derived algorithms are the same. Some remarkable works on the reverse mode algorithm had been done by S. Linnainmaa [11] and W. Miller and C. Wrathall [12] from the viewpoint of the error analysis, by W. Baur and V. Strassen [1] from that of complexity, and by P.J. Werbos [17] from that of the optimization of neural networks. A practical program had been developed by B. Speelpenning in 1980 [15] and it was rewritten into Fortran by K.E. Hillstrom in 1985 (now registered in Netlib [5,6]).

Two proceedings of the international workshops held in 1991 and 1996 collect all the theories, techniques, practical programs, current works, and future problems as well as history on automatic differentiation [2,4]. It should be noted that, in 1992, A. Griewank proposed a drastic improvement of the reverse mode algorithm using the so-called checkpointing technique. He succeeded in reducing the order of the size of storage required for the reverse mode algorithm [3]. Several software tools for automatic differentiation have been developed and popular in the world, e. g., ADIC, ADIFOR, ADMIT-1, ADOL-C, ADOL-F, FADBAD, GRESS, Odyssée, PADRE2, TAMC, etc. (See [2,4].)

Estimates of Rounding Errors

In order to solve practical real-world problems, the approximation with floating-point numbers is inevitable so that it is important to analyze and estimate the accumulated rounding errors in a big numerical computation. Moreover, in terms of estimates of the accumulated rounding errors, one can define a normalized (or weighted) norm for a numerically computed vector, that is useful for checking whether the computed vector can be regarded as zero or not from the viewpoint of numerical computation [8].

For the previous example, let us denote as δ_k the rounding error generated at the execution of the basic operation to compute the value of v_k. Then, the rounding errors in the example is explicitly written:

$$
\begin{array}{l|l}
1 & \widetilde{v}_1 = \widetilde{x}_2 - \widetilde{x}_3 + \delta_1, \\
2 & \widetilde{v}_2 = \widetilde{x}_1 * \widetilde{v}_1 + \delta_2, \\
3 & \widetilde{v}_3 = \exp(\widetilde{v}_2) + \delta_3, \\
4 & \widetilde{v}_4 = \widetilde{v}_3 + 1 + \delta_4, \\
5 & \widetilde{v}_5 = \widetilde{v}_2 / \widetilde{v}_4 + \delta_5.
\end{array}
$$

Here, $\widetilde{v}_k$ is the value with accumulated rounding errors.

Defining a function $\widetilde{f}$ as

$$\widetilde{f}(x_1, x_2, x_3; \delta_1, \delta_2, \delta_3, \delta_4, \delta_5) = \frac{x_1(x_2 - x_3 + \delta_1) + \delta_2}{\exp(x_1(x_2 - x_3 + \delta_1) + \delta_2) + \delta_3 + 1 + \delta_4} + \delta_5,$$

one has

$$\widetilde{v}_5 = \widetilde{f}(x_1, x_2, x_3; \delta_1, \ldots, \delta_5),$$
$$v_5 = \widetilde{f}(x_1, x_2, x_3; 0, \ldots, 0) .$$

Here, $\widetilde{v}_5 - v_5$ is the accumulated rounding error in the function value. For $v_5 = v_2/v_4 = \varphi_5(v_2, v_4)$, one has

$$\widetilde{v}_5 - v_5 = \varphi_5(\widetilde{v}_2, \widetilde{v}_4) - \varphi_5(v_2, v_4) + \delta_5 = \frac{\partial \varphi_5}{\partial v_2}(\xi_2, \xi_4) \cdot (\widetilde{v}_2 - v_2) + \frac{\partial \varphi_5}{\partial v_4}(\xi_2, \xi_4) \cdot (\widetilde{v}_4 - v_4) + \delta_5 ,$$

where $\xi_2 = \theta'\widetilde{v}_2 + (1-\theta')v_2$ and $\xi_4 = \theta''\widetilde{v}_4 + (1-\theta'')v_4$ for $0 < \theta', \theta'' < 1$. Expanding $\widetilde{v}_2 - v_2$ and $\widetilde{v}_4 - v_4$ similarly and expanding the other intermediate variables sequentially, the approximation:

$$\widetilde{v}_5 - v_5 \simeq \sum_{k=1}^{5} \frac{\partial \widetilde{f}}{\partial \delta_k} \delta_k \tag{12}$$

is derived [10]. Note that $\frac{\partial \widetilde{f}}{\partial \delta_k}$ are computed as $\overline{v}_k$ in Program 6, which are the final results of (9) and (10).

The locally generated rounding error δ_k for the floating-point number system is bounded by

$$|\delta_k| \le c \cdot |v_k| \cdot \varepsilon_M , \tag{13}$$

where ε_M indicates so-called 'machine epsilon' and $c = 1$ may be adopted for arithmetic operations according to IEEE754 standard. Then $\Delta[f]_{\mathrm{A}}$, called *absolute estimation*, is defined by

$$\Delta[f]_{\mathrm{A}} \equiv \sum_{k=1}^{r} \left| \frac{\partial \widetilde{f}}{\partial \delta_k} \right| \cdot |v_k| \cdot \varepsilon_M , \tag{14}$$

which is an upper bound on the accumulated rounding error. Regarding the locally generated errors δ_k's as pseudo-probabilistic variables uniformly distributed over $[-|v_k|\,\varepsilon_M, |v_k|\varepsilon_M]$'s, $\Delta[f]_{\mathrm{P}}$, called *probabilistic estimate*, is defined by

$$\Delta[f]_{\mathrm{P}} \equiv \varepsilon_M \sqrt{\frac{1}{3} \sum_{k=1}^{r} \left(\frac{\partial \widetilde{f}}{\partial \delta_k} \cdot v_k \right)^2} . \tag{15}$$

There are several reports in which these estimates give quite good approximations to the actual accumulated rounding errors [8].

Moreover, one could answer the problem how to choose a norm for measuring the size of numerically computed vector. By means of the estimates of the rounding errors, a weighted norm of a vector $\mathbf{f} = [f_1, \ldots, f_m]$ whose components are numerically computed is defined by

$$||\mathbf{f}||_{\mathrm{N}} \equiv \left\| \left[\frac{f_1}{\Delta[f_1]_{\mathrm{A}}}, \ldots, \frac{f_m}{\Delta[f_m]_{\mathrm{A}}} \right] \right\|_p , \tag{16}$$

($p = 1,2$ or ∞). This weighted norm is called *normalized norm*, because it is normalized with respect to accumulated rounding errors. With this normalized norm, one can determine whether a computed vector approaches to zero or not in reference to the rounding errors accumulated in the components. Note that, since all the components of the vector are divided by the estimates of accumulated rounding errors, they have no physical dimension. The normalized norm may be used effectively as stopping criteria for iterative methods like the Newton–Raphson method.

See also

- Automatic Differentiation: Calculation of the Hessian
- Automatic Differentiation: Calculation of Newton Steps
- Automatic Differentiation: Geometry of Satellites and Tracking Stations
- Automatic Differentiation: Parallel Computation
- Automatic Differentiation: Point and Interval
- Automatic Differentiation: Point and Interval Taylor Operators

- ▶ Automatic Differentiation: Root Problem and Branch Problem
- ▶ Nonlocal Sensitivity Analysis with Automatic Differentiation

References

1. Baur W, Strassen V (1983) The complexity of partial derivatives. Theor Comput Sci 22:317–330
2. Berz M, Bischof C, Corliss G, Griewank A (eds) (1996) Computational differentiation: Techniques, applications, and tools. SIAM, Philadelphia
3. Griewank A (1992) Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. Optim Methods Soft 1:35–54
4. Griewank A, Corliss GF (eds) (1991) Automatic differentiation of algorithms: Theory, implementation, and application. SIAM, Philadelphia
5. Hillstrom KE (1985) Installation guide for JAKEF. Techn Memorandum Math and Computer Sci Div Argonne Nat Lab ANL/MCS-TM-17
6. Hillstrom KE (1985) User guide for JAKEF. Techn Memorandum Math and Computer Sci Div Argonne Nat Lab ANL/MCS-TM-16
7. Iri M (1984) Simultaneous computation of functions, partial derivatives and estimates of rounding errors – Complexity and practicality. Japan J Appl Math 1:223–252
8. Iri M, Tsuchiya T, Hoshi M (1988) Automatic computation of partial derivatives and rounding error estimates with applications to large-scale systems of nonlinear equations. J Comput Appl Math 24:365–392
9. Kagiwada H, Kalaba R, Rasakhoo N, Spingarn K (1986) Numerical derivatives and nonlinear analysis. Math. Concepts and Methods in Sci. and Engin., vol 31. Plenum, New York
10. Kubota K, Iri M (1991) Estimates of rounding errors with fast automatic differentiation and interval analysis. J Inform Process 14:508–515
11. Linnainmaa S (1976) Taylor expansion of the accumulated rounding error. BIT 16:146–160
12. Miller W, Wrathall C (1980) Software for roundoff analysis of matrix algorithms. Acad Press, New York
13. Ostrovskii GM, Wolin JM, Borisov WW (1971) Über die Berechnung von Ableitungen. Wiss Z Techn Hochschule Chemie 13:382–384
14. Rall LB (1981) Automatic differentiation – Techniques and applications. Lecture Notes Computer Science, vol 120. Springer, Berlin
15. Speelpenning B (1980) Compiling fast partial derivatives of functions given by algorithms. Report Dept Computer Sci Univ Illinois UIUCDCS-R-80-1002
16. Wengert RE (1964) A simple automatic derivative evaluation program. Comm ACM 7:463–464
17. Werbos P (1974) Beyond regression: New tools for prediction and analysis in the behavioral sciences. PhD Thesis, Appl. Math. Harvard University

Automatic Differentiation: Parallel Computation

CHRISTIAN H. BISCHOF[1], PAUL D. HOVLAND[2]

[1] Institute Sci. Computing, University Technol., Flachen, Germany

[2] Math. and Computer Sci. Div., Argonne National Lab., Argonne, USA

MSC2000: 65Y05, 68N20, 49-04

Article Outline

Keywords
Background
Implementation Approaches
AD of Parallel Programs
AD-Enabled Parallelism
  Data Parallelism
  Time Parallelism
Parallel AD Tools
Summary
See also
References

Keywords

Automatic differentiation; Parallel computing; MPI

Research in the field of *automatic differentiation* (AD) has blossomed since A. Griewank's paper [15] in 1989 and the Breckenridge conference [17] in 1991. During that same period, the power and availability of parallel machines have increased dramatically. A natural consequence of these developments has been research on the interplay between AD and *parallel computations*. This relationship can take one of two forms. One can examine how AD can be applied to existing parallel programs. Alternatively, one can consider how AD introduces new potential for parallelism into existing sequential programs.

Background

Automatic differentiation relies upon the fact that all programming languages are based on a finite number of elementary functions. By providing rules for the differentiation of these elementary functions, and by com-

bining these elementary derivatives according to the chain rule of differential calculus, an AD system can differentiate arbitrarily complex functions. The chain rule is associative—partial derivatives can be combined in any order. The *forward mode of AD* combines the partial derivatives in the order of evaluation of the elementary functions to which they correspond. The *reverse mode* combines them in the reverse order. For systems with a large ratio of dependent to independent variables, the reverse mode offers lower operation counts, at the cost of increased storage costs [15].

The forward and the reverse mode are the extreme ends of a wide algorithmic spectrum of accumulating derivatives. Recently, hybrid approaches have been developed which combine the forward and the reverse mode [5,10], or apply them in a hierarchical fashion [8,25]. In addition, efficient *checkpointing* schemes have been developed which address the potential storage explosion of the reverse mode by judicious recomputation of intermediate states [16,19]. Viewing the problem of automatic differentiation as an edge elimination problem on the program graph corresponding to a particular code, one can in fact show that the problem of computing derivatives with minimum cost is *NP*-hard [21]. The development of more efficient heuristics is an area of active research (see, for example, several of the papers in [3]).

Implementation Approaches

Automatic differentiation is a particular instantiation of a rule-based semantic transformation process. That is, whenever a floating-point variable changes, an associated derivative object must be updated according to the chain rule of differential calculus. For example, in the forward mode of AD, a derivative object carries the partial derivative(s) of an associated variable with respect to the independent variable(s). In the reverse mode of AD, a derivative object carries the partial derivative(s) of the dependent variable(s) with respect to an associated variable. Thus, any AD tool must provide an instantiation of a 'derivative object', maintain the association between an original variable and its derivative object, and update derivative objects in a timely fashion.

Typically AD is implemented in one of two ways: operator overloading or source transformation. In languages that allow operator overloading, such as C++ and Fortran90, each elementary function can be redefined so that in addition to the normal function, derivatives are computed as well, and either saved for later use or propagated by the chain rule. A simple class definition using the forward mode might be implemented as follows:

```
class adouble{
private:
  double value, grad[GRAD_LENGTH];
public:
  /* constructors omitted */
  friend adouble operator*(const
    adouble &, const adouble &);
  /* similar decs for other ops */
}
adouble operator*(const adouble &g1,
  const adouble &g2){
    int i;
    double newgrad[GRAD_LENGTH];
    for (i=0; i<GRAD_LENGTH;i++){
      newgrad[i] =
         (g1.value)*(g2.grad[i])+
         (g2.value)*(g1.grad[i]);
    }
    return adouble(g1.value*g2.value,
      newgrad);
}
```

An example of how this class could be used is given below.

In languages that do not support operator overloading, it can be faked by manually or automatically replacing operators such as + and ∗ with calls to subroutines.

```
main(){
  double temp[GRAD_LENGTH];
  adouble y;

  /* initialize x1 to (3.0,[1.0  0.0]),
                x2 to (4.0,[0.0 1.0])*/
  temp[0] = 1.0; temp[1] = 0.0;
  adouble  *x1 = new adouble(3.0,temp);
  temp[0] = 0.0; temp[1] = 1.0;
  adouble  *x2 = new adouble(4.0,temp);

   y = (*x1)*(x2);

  /* output (y,[dy/dx1 dy/dx2])  */
  cout << y;
  /* prints (12.0,[4.0 3.0]) */
}
```

As an alternative to operator overloading, a preprocessor can be used to transform source code for computing the function into source code for computing the function and its derivatives. This approach relies heavily on compiler technology and typically involves a combination of in-lining and subroutine calls to implement the propagation of derivatives. An example of ADIFOR-generated code (edited for clarity) invoking the SparsLinC library for transparent exploitation of *sparsity* [6] follows.

```
c derivation code for f=x*y/z

c preaccumulate partial derivates
      temp1 = x*y/z
      temp2 = 1.0/z
      temp3 = temp2*y
      temp4 = temp2*x
      temp5 = -temp1/z

c propagate derivatives
c (g_x, ... , g_f may by sparse)
      call sspg3q(g_f,temp3,g_x,temp4,
     +            g_y,temp5,g_z)

      f=temp1
```

The advantage of this approach is that it allows the exploitation of computational context in deciding how to propagate derivatives. For example, a recently developed *Hessian* module [1], adaptively determines the best strategy for each assignment statement in the code based on a machine-specific performance model for the implementation kernels employed.

A comparison of these two implementation approaches is provided in [9]. This paper also introduces an implementation design that separates the core issues of automatic differentiation from language-specific issues through the use of an interface layer called AIF (*AD intermediate form*), thus arriving at a system design that allows reuse of differentiation components across front-ends for different languages. Long-term, such a system design also allows the exploitation of the best features of both source transformation and operator overloading.

Current AD tools based on operator overloading include ADOL-C [18] and ADOL-F [29], both of which offer the option of using either the forward or the reverse mode, and to compute derivatives of arbitrary order Source transformation tools that use mostly the forward mode to provide first- and second order derivatives include ADIC [9] and ADIFOR [6]. The Odyssee [28] and TAMC [14] tools use the reverse mode in a source transformation context to provide first order derivatives. A more comprehensive survey of AD tools can be found at the website [31].

AD of Parallel Programs

In 1994, R.L. Hinkins reported on the application of AD to magnetic field calculations implemented in the data parallel languages MPFortran (MasPar Fortran) and CMFortran [22]. In 1997, P. Hovland addressed the larger issue of AD of parallel programs in general, paying close attention to message-passing parallel programs [23], but also considering other parallel programming paradigms, and A. Carle developed ADIFOR-MP, a prototype tool supporting a subset of MPI [30] and PVM [13] constructs. The focus on parallel programs employing a message-passing paradigm can be attributed to the popularity of this parallel programming paradigm and its relevance to all parallel programs targeting nonuniform memory access (NUMA) machines.

Correct AD of message-passing parallel programs requires that we maintain an association between a variable and its derivative object. In particular, when a variable is sent from one processor to another via a message, we must also send the associated derivative object. There are two ways of accomplishing this goal — we can pack the variable and derivative object together in one message or send two separate messages. Packing a variable and its associated derivative object into a single message may incur a copying overhead. On the other hand, sending separate messages requires a mechanism for associating the messages with one another at the receiving end and will increase delivery time on high-latency systems. In general, it is preferable to pack the variable and derivative object together in one message [24], minimizing copying cost through judiciously chosen derivative data structures. Other issues in ensuring correct AD of parallel programs include proper handling of nondeterminism, reduction operations at points of nondifferentiability, and seed matrix initialization [23].

In many instances, only a subset of the program input- and output variables is considered as indepen-

dent or dependent variables with respect to differentiation. An optimization technique that tries to exploit this fact is activity analysis, which seeks to reduce time and storage costs by identifying variables that do not lie on the computational path from independent to dependent variables. Such variables are termed passive and do not require an associated derivative object. Activity analysis depends on sophisticated compiler technology, namely interprocedural dataflow analysis. In message-passing parallel programs, sends and receives greatly complicate such an analysis. As the analysis needs to guarantee correctness, this fact leads to much more conservative assumptions, and as a result much optimization potential may be lost. Among the available options to circumvent this situation are user annotations, runtime analysis, or the use of a higher-level language such as HPF [26]. These issues are investigated in more detail in [23].

Another issue arising in the parallel setting is the computation of partial derivatives of new elementary functions, such as parallel *reduction operations*. For most of the common reduction operations, such as sum, maximum, and minimum, computing the partial derivatives is trivial. For the product reduction, the situation is more complex. The partial derivative of $y = \prod_{i=1}^{n} x_i$ with respect to x_i is $\partial y / \partial x_i = (\prod_{j=1}^{i-1} x_j)(\prod_{k=i+1}^{n} x_k)$. These partial derivatives can be computed using a parallel prefix and reverse parallel prefix operation. However, propagating the partial derivatives requires an additional sum reduction. We could instead combine the partial derivative computation and propagation into a single reduction. This increases the computational cost, but reduces the communication cost. In [24], Hovland and C. Bischof discuss the conditions under which each approach should be preferred and give experimental results to support the theory.

AD-Enabled Parallelism

As early as 1991, Bischof considered the problem of parallelizing the computation of derivatives computed via AD [4] to distribute the additional work introduced by AD. Applying AD to a program introduces two basic types of parallelism: data parallelism and time parallelism.

Data Parallelism

The potential for data parallelism arises whenever there are multiple independent variables (for the forward mode) or multiple dependent variables (for the reverse mode). Different processes can be employed to propagate partial derivatives with respect to a subset of the independent variables in parallel.

Such an implementation is feasible if one can employ light-weight threads for the parallel derivative computation. A limiting factor is the fact that the derivative computations are interspersed with the function computation. Thus, an alternative approach is to replicate the sequential computation on each processor, thereby virtually eliminating communication costs. This approach has proven effective for computations involving a large number of independent variables [7,32].

Time Parallelism

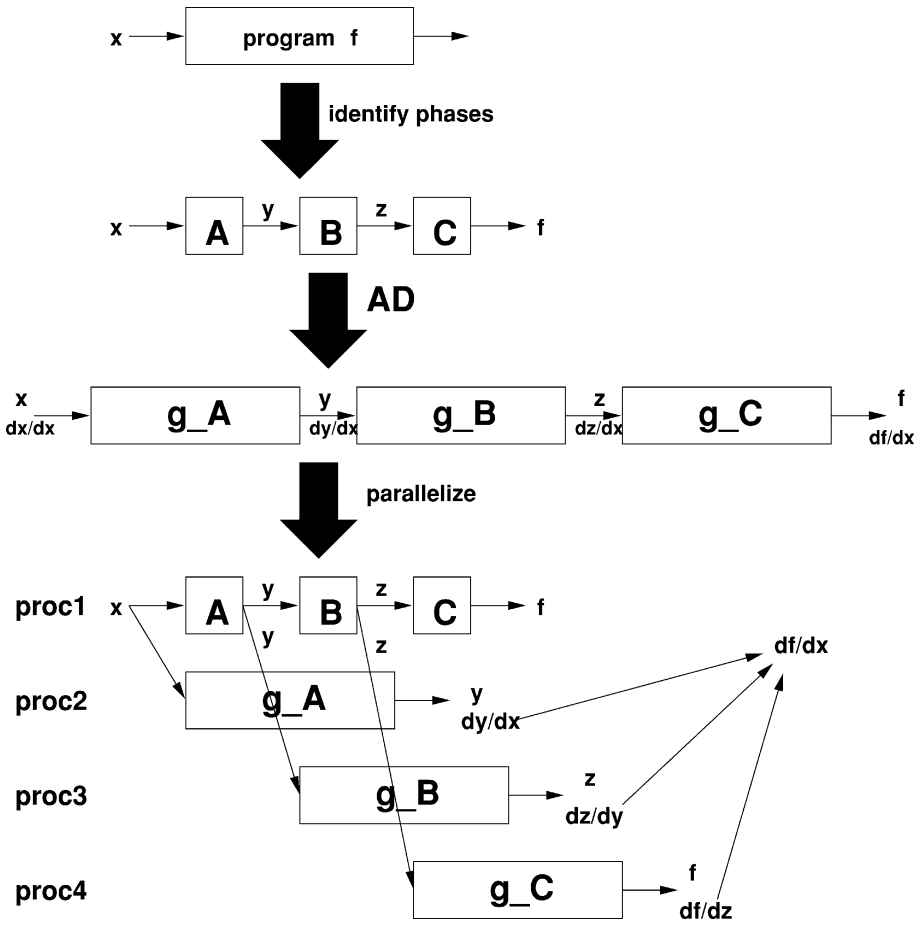
Time parallelism arises as a consequence of the associativity of the chain rule. By breaking the computation into several phases, we can compute and propagate partial derivatives over each phase simultaneously, then combine the results according to the chain rule. This approach is illustrated in Fig. 1. Before each phase, a derivative computation for that phase is forked off, using as input the results of the previous phase. At the conclusion of the derivative computations, the partial derivatives are combined according to the chain rule.

This illustration assumes the forward mode. If we were using the reverse mode, the derivative computation for phase A would be forked off after phase A had completed. The effectiveness of this approach has been demonstrated for both the forward mode [10] and the reverse mode [2]. The associativity of the chain rule makes it possible to apply this time-parallel approach to arbitrary computational structures, not just the linear schedule illustrated here.

Parallel AD Tools

Research in AD and parallelism is relatively new. Nonetheless, there are several such tools, at varying stages of development.

Hinkins developed special purpose libraries for the AD of programs written in MPFortran or CM-



Automatic Differentiation: Parallel Computation, Figure 1

Fortran [22]. Use of these libraries required that each arithmetic operation be manually replaced by a subroutine call. As part of his thesis [23], Hovland developed prototype tools for AD of FortranM [12], Fortran with a subset of MPI [20] message passing, and C with MPI. Carle is developing (1999) a prototype version of ADIFOR [11] supporting MPI and PVM. Roh is developing an extension to ADIC that seeks to automatically exploit the parallelism introduced by AD through the use of threads [27].

Summary

Since 1989, a great deal of progress has been made in the fields of automatic differentiation and parallel computation. Parallel computation and AD interact in two ways. AD can be applied to a parallel program. Alternatively, AD can be used as a source of new parallelism in a computation. Effective strategies exist for exploiting each of the two types of parallelism introduced: time parallelism and data parallelism.

In either case, ensuring that the resulting derivative computation is both correct and efficient requires AD tools that are more sophisticated than in the serial setting. Most of the existing tools are early in their development cycle, but can be expected to mature swiftly as they adopt advanced computational infrastructure developed in other fields of computer science, e. g., parallelizing compilers or parallel runtime systems. Thus, we expect the beginning of 2000 to also provide robust

and effective tools for the differentiation of parallel programs and the introduction of parallelism through differentiation.

See also

- ► Asynchronous Distributed Optimization Algorithms
- ► Automatic Differentiation: Calculation of the Hessian
- ► Automatic Differentiation: Calculation of Newton Steps
- ► Automatic Differentiation: Geometry of Satellites and Tracking Stations
- ► Automatic Differentiation: Introduction, History and Rounding Error Estimation
- ► Automatic Differentiation: Point and Interval
- ► Automatic Differentiation: Point and Interval Taylor Operators
- ► Automatic Differentiation: Root Problem and Branch Problem
- ► Heuristic Search
- ► Interval Analysis: Parallel Methods for Global Optimization
- ► Load Balancing for Parallel Optimization Techniques
- ► Nonlocal Sensitivity Analysis with Automatic Differentiation
- ► Parallel Computing: Complexity Classes
- ► Parallel Computing: Models
- ► Parallel Heuristic Search
- ► Stochastic Network Problems: Massively Parallel Solution

References

1. Abate J, Bischof Ch, Carle A, Roh L (1997) Algorithms and design for a second-order automatic differentiation module. Proc. Internat. Symp. Symbolic and Algebraic Computing (ISSAC) '97, ACM, New York, pp 149–155
2. Benary J (1996) Parallelism in the reverse mode. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 137–147
3. Berz M, Bischof Ch, Corliss G, Griewank A (1996) Computational differentiation: Techniques, applications, and tools. SIAM, Philadelphia
4. Bischof ChH (1991) Issues in parallel automatic differentiation. In: Griewank A, Corliss G (eds) Automatic Differentiation of Algorithms. SIAM, Philadelphia, pp 100–113
5. Bischof Ch, Carle A, Corliss G, Griewank A, Hovland P (1992) ADIFOR: Generating derivative codes from Fortran programs. Scientif Program 1(1):11–29
6. Bischof Ch, Carle A, Khademi P, Mauer A (1996) ADIFOR 2.0: Automatic differentiation of Fortran 77 programs. IEEE Comput Sci Eng 3(3):18–32
7. Bischof Ch, Green L, Haigler K, Knauff T (1994) Parallel calculation of sensitivity derivatives for aircraft design using automatic differentiation. Proc. 5th AIAA/NASA/USAF/ISSMO Symp. Multidisciplinary Analysis and Optimization, AIAA-94-4261, Amer Inst Aeronautics and Astronautics, Reston, VA, pp 73–84
8. Bischof ChH, Haghighat MR (1996) On hierarchical differentiation. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 83–94
9. Bischof Ch, Roh L, Mauer A (1997) ADIC – An extensible automatic differentiation tool for ANSI-C. Software Practice and Experience 27(12):1427–1456
10. Bischof Ch, Wu Po-Ting (1997) Time-parallel computation of pseudo-adjoints for a leapfrog scheme. Preprint Math and Computer Sci Div Argonne Nat Lab no. ANL/MCS-P639-0197
11. Carle A (1997) ADIFOR-MP – A prototype automatic differentiation tool for Fortran 77 with message-passing extensions. Personal communication
12. Foster IT, Chandy KM (1995) Fortran M: A language for modular parallel programming. J Parallel Distributed Comput 25(1)
13. Geist A, Beguelin A, Dongarra J, Jiang W, Manchek R, Sunderam V (1994) PVM – Parallel virtual machine: A users' guide and tutorial for network parallel computing. MIT, Cambridge, MA
14. Giering R, Kaminski Th (1996) Recipes for adjoint code construction. Max-Planck Inst Meteorologie, Hamburg no. 212
15. Griewank A (1989) On automatic differentiation. In: Iri M, Tanabe K (eds) Mathematical Programming: Recent Developments and Applications. Kluwer, Dordrecht, pp 83–108
16. Griewank A (1992) Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. Optim Methods Softw 1(1):35–54
17. Griewank A, Corliss G (1991) Automatic differentiation of algorithms. SIAM, Philadelphia
18. Griewank A, Juedes D, Utke J (1996) ADOL-C, a package for the automatic differentiation of algorithms written in C/C++. ACM Trans Math Softw 22(2):131–167
19. Grimm J, Pottier L, Rostaing-Schmidt N (1996) Optimal time and minimum space time product for reversing a certain class of programs. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation, Techniques, Applications, and Tools. SIAM, Philadelphia, pp 95–106

20. Gropp W, Lusk E, Skjellum A (1994) Using MPI – Portable parallel programming with the message passing interface. MIT, Cambridge, MA
21. Herley K (1993) On the NP-completeness of optimum accumulation by vertex elimination. Unpublished Manuscript
22. Hinkins R L (Sept. 1994) Parallel computation of automatic differentiation applied to magnetic field calculations. MSc Thesis Univ Calif
23. Hovland P (1997) Automatic differentiation of parallel programs. PhD Thesis Univ. Illinois at Urbana-Champaign
24. Hovland P, Bischof Ch (1998) Automatic differentiation of message-passing parallel programs. Proc. First Merged Internat. Parallel Processing Symp. and Symp. on Parallel and Distributed Processing, IEEE Computer Soc Press, New York
25. Hovland P, Bischof Ch, Spiegelman D, Casella M (1997) Efficient derivative codes through automatic differentiation and interface contraction: An application in biostatistics. SIAM J Sci Comput 18(4):1056–1066
26. Koelbel C, Loveman D, Schreiber R, Steele G Jr, Zosel M (1994) The high performance Fortran handbook. MIT, Cambridge, MA
27. Roh L (1997) Personal Communication
28. Rostaing N, Dalmas St, Galligo A (Oct. 1993) Automatic differentiation in Odyssee. Tellus 45a(5):558–568
29. Shiriaev D, Griewank A (1996) ADOL-F: Automatic differentiation of Fortran codes. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 375–384
30. Snir M, Otto SW, Huss-Lederman S, Walker DW, Dongarra Jack (1996) MPI: The complete reference. MIT, Cambridge, MA
31. WEB http://www.mcs.anl.gov/autodiff/adtools/
32. Zhang Y, Bischof Ch, Easter R, Wu Po-Ting (1997) Sensitivity analysis of O3 and photochemical indicators using a mixed-phase chemistry box model and automatic differentiation techniques. 90th Air and Waste Management Assoc. Annual Meeting and Exhibition June 8-13, 1997, Toronto. vol 97-WA68A.04, Air and Waste Management Assoc, Pittsburgh, PA, pp 1–16

Automatic Differentiation: Point and Interval

AD

L. B. Rall[1], George F. Corliss[2]
[1] University Wisconsin–Madison, Madison, USA
[2] Marquette University, Milwaukee, USA

MSC2000: 65H99, 65K99

Article Outline

Keywords
See also
References

Keywords

Differentiation; Computational methods

Automatic differentiation (abbreviated *AD*) is a computational method for evaluating derivatives or Taylor coefficients of algorithmically defined functions. Simply speaking, an *algorithmic definition* of a function is a step-by-step specification of its evaluation by arithmetic operations and library functions. Application of the rules of differentiation to the algorithmic definition of a differentiable function yields values of its derivatives. Examples of algorithmic definitions of functions are code lists, computer subroutines, and even entire computer programs.

Automatic differentiation differs from numerical differentiation based on difference quotients of function values in that automatic differentiation is exact in principle, but of course is subject to roundoff error in practice. In addition to roundoff error, difference quotients entail truncation error. Attempts to reduce this truncation error by decreasing stepsize results in cancellation of significant digits and a catastrophic increase in roundoff error in general. Automatic differentiation also differs significantly from the symbolic differentiation taught in school, the goal of which is the transformation of formulas for functions into formulas for their derivatives. Although automatic differentiation uses the same rules of differentiation as symbolic differentiation, these rules are applied to the algorithmic definition of the function, not to a formula for it, and the results are values of derivatives, not formulas. Furthermore, formulas may not be available for functions of interest defined only algorithmically by computer subroutines or programs to which automatic differentiation can be applied. In summary, automatic differentiation is more accurate than numerical differentiation and requires fewer resources and is more generally applicable than symbolic differentiation.

The simplest type of algorithmic definition of a function is a *code list*, which is similar to the segment of computer code for the evaluation of an expression

(i. e., a formula). For illustration, consider the function defined by the formula

$$f(x, y) = (xy + \sin x + 4)(3y^2 + 6).$$

An equivalent algorithmic definition of this function by a code list is

$$\begin{array}{lllllll} t_1 & = & x, & & t_6 & = & t_5 + 4, \\ t_2 & = & y, & & t_7 & = & t_2^2, \\ t_3 & = & t_1 t_2, & & t_8 & = & 3t_7, \\ t_4 & = & \sin t_1, & & t_9 & = & t_8 + 6, \\ t_5 & = & t_3 + t_4, & & t_{10} & = & t_6 t_9. \end{array}$$

Given the values of x and y, evaluation of the subsequent entries in the code list gives $t_{10} = f(x, y)$. Indeed, the first step in evaluation or symbolic differentiation of a function defined by a formula is to form a corresponding code list, perhaps subconsciously. The conversion of well-formed expressions into code lists is a fundamental process in computer science, sometimes called 'formula translation'. Although both automatic differentiation and symbolic differentiation are applicable in this case, automatic differentiation requires only the code list and produces only values of derivatives for given values of the input variables. To compute the *gradient* ∇f, the rules of differentiation applied to the code list above gives

$$\begin{aligned} \nabla t_1 &= \nabla x, \\ \nabla t_2 &= \nabla y, \\ \nabla t_3 &= t_1 \nabla t_2 + t_2 \nabla t_1, \\ \nabla t_4 &= (\cos t_1) \nabla t_1, \\ \nabla t_5 &= \nabla t_3 + \nabla t_4, \\ \nabla t_6 &= \nabla t_5, \\ \nabla t_7 &= 2t_2 \nabla t_2, \\ \nabla t_8 &= 3 \nabla t_7, \\ \nabla t_9 &= \nabla t_8, \\ \nabla t_{10} &= t_6 \nabla t_9 + t_9 \nabla t_6. \end{aligned}$$

It is evident from the chain rule that

$$\nabla t_{10} = \nabla f(x, y) = f_x(x, y) \nabla x + f_y(x, y) \nabla y.$$

Thus, once the code list for $f(x, y)$ is given and the 'seed' values of x, ∇x and y, ∇y are known, the values of the function and its gradient can be computed without formulas for either. In case x, y are independent variables, then $\nabla x = [1, 0]$, $\nabla y = [0, 1]$ and

$$\begin{aligned} \nabla f(x, y) &= [f_x(x, y), f_y(x, y)] \\ &= [t_9(t_2 + \cos t_1), 6t_2 t_6 + t_1 t_9]. \end{aligned}$$

This example illustrates the *forward mode* of automatic differentiation. This process is not restricted to first derivatives as long as the entries t_i of the code list have the desired number of derivatives.

Although the forward mode illustrated above is easy to understand and implement, it is usually more efficient to compute gradients in what is called the *reverse mode*. To explain this process, consider a general code list $t = (t_1, \ldots, t_n)$ which begins with m input variables $t_1, \ldots, t_m$, and ends with p output variables $t_{n-p+1}, \ldots, t_n$. For $i > m$, the entry $t_i = t_j \circ t_k$, where $j, k < i$ and $\circ$ denotes an arithmetic operation, or $t_i = \phi(t_j)$ with $j < i$, where ϕ is a function belonging to a library of standard functions. For convenience, arithmetic operations between constants and entries will be considered library functions in addition to the usual sine, cosine, and so on.

If K_i denotes the set of indices $k < i$ such that the entry t_i of the code list depends explicitly on t_k, then the forward mode of automatic differentiation consists of application of the chain rule in the form

$$\nabla t_i = \sum_{k \in K_i} \frac{\partial t_i}{\partial t_k} \nabla t_k$$

for $i = m + 1, \ldots, n$, to obtain the gradients of the intermediate variables and output. This process works because $\nabla t_1, \ldots, \nabla t_{i-1}$ are known or have been computed before they are needed for the evaluation of ∇t_i. If the seed gradients have dimension at most d, then the forward mode of automatic differentiation requires computational effort proportional to nd, that is, d times the effort required for evaluation of the output t_n. If $d > m$, then it is more efficient to consider the input variables to be independent and then compose ∇f by the standard formula given below. This limits the computational effort for the forward mode to an amount essentially proportional to nm.

The reverse mode is another way to apply the chain rule. Instead of propagating the seed gradients $\nabla t_1, \ldots, \nabla t_m$ throughout the computation, differentiation is applied to the code list in reverse order. In the case of

a single output variable t_n, first t_n is differentiated with respect to itself, then with respect to $t_{n-1}, \ldots, t_1$. The resulting *adjoints* $\partial t_n/\partial t_m, \ldots, \partial t_n/\partial t_1$ and the seed gradients then give

$$\nabla t_n = \sum_{i=1}^{m} \frac{\partial t_n}{\partial t_i} \nabla t_i .$$

Formally, the adjoints are given by

$$\frac{\partial t_n}{\partial t_n} = 1, \qquad \frac{\partial t_n}{\partial t_k} = \sum_{i \in I_k} \frac{\partial t_k}{\partial t_i} \frac{\partial t_i}{\partial t_k},$$

$k = n-1, \ldots, 1$, where I_k is the set of indices $i > k$ such that t_i depends explicitly on t_k. It follows that the computational effort to obtain adjoints in the reverse mode is proportional to n, the length of the code list, and is essentially independent of the number of input variables and the dimensionalities of the seed gradients. This can result in significant savings in computational time. In the general case of several output variables, the same technique is applied to each to obtain their gradients.

The reverse mode applied to the example code list gives

$$\begin{aligned}
\frac{\partial t_{10}}{\partial t_{10}} &= 1, \\
\frac{\partial t_{10}}{\partial t_9} &= t_6, \\
\frac{\partial t_{10}}{\partial t_8} &= \frac{\partial t_{10}}{\partial t_9}\frac{\partial t_9}{\partial t_8} = t_6 \cdot 1, \\
\frac{\partial t_{10}}{\partial t_7} &= \frac{\partial t_{10}}{\partial t_8}\frac{\partial t_8}{\partial t_7} = t_6 \cdot 3, \\
\frac{\partial t_{10}}{\partial t_6} &= t_9, \\
\frac{\partial t_{10}}{\partial t_5} &= \frac{\partial t_{10}}{\partial t_6}\frac{\partial t_6}{\partial t_5} = t_9 \cdot 1, \\
\frac{\partial t_{10}}{\partial t_4} &= \frac{\partial t_{10}}{\partial t_5}\frac{\partial t_5}{\partial t_4} = t_9 \cdot 1, \\
\frac{\partial t_{10}}{\partial t_3} &= \frac{\partial t_{10}}{\partial t_5}\frac{\partial t_5}{\partial t_3} = t_9 \cdot 1, \\
\frac{\partial t_{10}}{\partial t_2} &= \frac{\partial t_{10}}{\partial t_7}\frac{\partial t_7}{\partial t_2} + \frac{\partial t_{10}}{\partial t_3}\frac{\partial t_3}{\partial t_2} \\
&= (3t_6) \cdot (2t_2) + t_9 \cdot t_1, \\
\frac{\partial t_{10}}{\partial t_1} &= \frac{\partial t_{10}}{\partial t_4}\frac{\partial t_4}{\partial t_1} + \frac{\partial t_{10}}{\partial t_3}\frac{\partial t_3}{\partial t_1} = t_9 \cdot \cos t_1 + t_9 \cdot t_2 .
\end{aligned}$$

Although this computation appears to be complicated, a comparison of operation counts in the case x, y are independent variables shows that even for this low-dimensional example, the reverse mode requires 13 operations to evaluate ∇f in addition to the operations required to evaluate f itself, while the forward mode requires $22 = 2 + 10\ m$. In reverse mode, the entire code list has to be evaluated and its values stored before the reverse sweep begins. In forward mode, since the computation of t_i and each component of ∇t_i can be carried out independently, a parallel computer with a sufficient number of processors could compute t_n, ∇t_n in a single pass through the code list, that is, with effort proportional to n. A more detailed comparison of forward and reverse modes for calculating gradients can be found in the tutorial article [1, pp. 1–18] and the book [3].

Implementation of automatic differentiation can be by *interpretation*, *operator overloading*, or *code transformation*. Early software for automatic differentiation simply interpreted a code list by calling the appropriate subroutines for each arithmetic operation or library function. Although inefficient, this approach is still useful in interactive applications in which functions entered from the keyboard are parsed to form code lists, which are then interpreted to evaluate the functions and their derivatives.

Operator overloading is a familiar concept in mathematics, as the symbol '+' is used to denote addition of such disparate objects as integers, real or complex numbers, vectors, matrices, functions, etc. It follows that a code list as defined above can be evaluated in any mathematical system in which the required arithmetic operations and library function are available, including differentiation arithmetics [14, pp. 73–90]. These arithmetics can be used to compute derivatives or Taylor coefficients of any order of sufficiently smooth functions. In optimization, gradient and Hessian arithmetics are most frequently used. In gradient arithmetic, the basic data type is the ordered pair $(f, \nabla f)$ of a number and a vector representing values of a function and its gradient vector. Arithmetic operations in this system are defined by

$$\begin{aligned}
(f, \nabla f) \pm (g, \nabla g) &= (f \pm g, \nabla f \pm \nabla g), \\
(f, \nabla f)(g, \nabla g) &= (fg, f\nabla g + g\nabla f), \\
\frac{(f, \nabla f)}{(g, \nabla g)} &= \left(\frac{f}{g}, \frac{g\nabla f - f\nabla g}{g^2}\right),
\end{aligned}$$

division by 0 excluded. If ϕ is a differentiable library function, then its extension to gradient arithmetic is defined by

$$\phi(f, \nabla f) = (\phi(f), \phi'(f)\nabla f),$$

which is just the chain rule. Hessian arithmetic extends the same idea to triples $(f, \nabla f, Hf)$, where Hf is a matrix representing the value of the *Hessian* of f, $Hf = [\partial^2 f/\partial x_i\, \partial x_j]$.

Programming differentiation arithmetic is convenient in modern computer languages which support operator overloading [9, pp. 291–309]. In this setting, the program is written with expressions or routines for functions in the regular form, and the compiler produces executable code for evaluation of these functions and the desired derivatives. For straightforward implementations such as the one cited above, the differentiation mode will be forward, which has implications for efficiency.

Code transformation essentially consists of analyzing the code for functions to generate code for derivatives. This results in a new computer program which then can be compiled and run as usual. To illustrate this idea, note that in the simple example given above, the expressions

$$f_x(x, y) = t_9(t_2 + \cos t_1),$$
$$f_y(x, y) = 6t_2t_6 + t_1t_9,$$

were obtained for the partial derivatives of the function in either forward or reverse mode. This differs from symbolic differentiation in that values of intermediate entries in the code list for $f(x, y)$ are involved rather than the variables x, y. The corresponding lists for these expressions

$$tx_1 = \cos t_1,$$
$$tx_2 = t_2 + tx_1,$$
$$tx_3 = t_9tx_2,$$
$$ty_1 = t_2t_6,$$
$$ty_2 = 6ty_1,$$
$$ty_3 = t_1t_9,$$
$$ty_4 = ty_2 + ty_3,$$

can then be appended to the code list for the function to obtain a routine with output values $t_{10} = f(x, y)$, $tx_3 = f_x(x, y)$, and $ty_4 = f_y(x, y)$. Further, automatic differentiation can be applied to this list to obtain routines for higher derivatives of f [13]. As a practical matter, duplicate assignments can be removed from such lists before compilation.

Up to this point, the discussion has been of *point AD*, values have been assumed to be real or complex numbers with all operations and library functions evaluated exactly. In reality, the situation is quite different. Expressions, meaning their equivalent code lists, are evaluated in an approximate computer arithmetic known as floating-point arithmetic. This often yields very accurate results, but examples of simple expressions are known for which double and even higher precision calculation gives an answer in which even the sign is wrong for certain input values. Furthermore, such failures can occur without any outward indication of trouble. In addition, values of input variables may not be known exactly, thus increasing the uncertainty in the accuracy of outputs. The use of *interval arithmetic* (abbreviated *IA*) provides a computational way to attack these problems [11].

The basic quantities in interval arithmetic are finite closed real intervals $X = [x_1, x_2]$, which represent all real numbers x such that $x_1 \le x \le x_2$. Arithmetic operations $\circ$ on intervals are defined by

$$X \circ Y = \{x \circ y\colon\ x \in X, y \in Y\},$$

again an interval, division by an interval containing zero excluded. Library functions ϕ are similarly extended to interval functions Φ such that $\phi(x) \in \Phi(X)$ for all $x \in X$ with $\Phi(X)$ expected to be an accurate inclusion of the range $\phi(X)$ of ϕ on X. Thus, if $f(x)$ is a function defined by a code list, then assignment of the interval value X to the input variable and evaluation of the entries in interval arithmetic yields the output $F(X)$ such that $f(x) \in F(X)$ for all $x \in X$. The interval function F obtained in this way is called the *united extension* of f [11].

In the floating-point version of interval arithmetic, all endpoints are floating-point numbers and hence exactly representable in the computer. Results of arithmetic operations and calls of library functions are

rounded outwardly (upper endpoints up, lower endpoints down) to the closest or very close floating-point numbers to maintain the guarantee of inclusion. Thus, one is still certain that for the interval extension F of f actually computed, $f(x) \in F(X)$ for all $x \in X$. Thus, for example, an output interval $F(X)$ which is very wide for a point input interval $X = [x, x]$ would serve as a warning that the algorithm is inappropriate or ill-conditioned, in contrast to the lack of such information in ordinary floating-point arithmetic.

Automatic differentiation carried out in interval arithmetic is called *interval automatic differentiation*. Interval computation has numerous implications for optimization, with or without automatic differentiation [6]. Maxima and minima of functions can 'slip through' approximate sampling of values at points of the floating-point grid, but have to be contained in the computable interval inclusion $F(X)$ of $f(x)$ over the same interval region X, for example.

Although interval arithmetic properly applied can solve many optimization and other computational problems, a word of warning is in order. The properties of interval arithmetic differ significantly from those of real arithmetic, and simple 'plugging in' of intervals for numbers will not always yield useful results. In particular, interval arithmetic lacks additive and multiplicative inverses, and multiplication is only subdistributive across addition, $X(Y+Z) \subset XY+XZ$ [11]. A real algorithm which uses one or more of these properties of real arithmetic is usually inappropriate for interval computation, and should be replaced by one that is suitable if possible.

To this point, automatic differentiation has been applied only to code lists, which programmers customarily refer to as 'straight-line code'. Automatic differentiation also applies to subroutines and programs, which ordinarily contain loops and branches in addition to expressions. These latter present certain difficulties in many cases. A loop which is traversed a fixed number of times can be 'unrolled,' and thus is equivalent to straight-line code. However, in case the stopping criterion is based on result values, the derivatives may not have achieved the same accuracy as the function values. For example, if the inverse function of a known function is being computed by iterative solution of the equation $f(x) = y$ for $x = f^{-1}(y)$, then automatic differentiation should be applied to f and the derivative of the inverse function obtained from the standard formula $(f^{-1})'(y) = (f'(x))^{-1}$. Branches essentially produce piecewise defined functions, and automatic differentiation then provides the derivative of the function defined by whatever branch is taken. This can create difficulties as described by H. Fischer [4, pp. 43–50], especially since a smooth function can be approximated well in value by highly oscillatory or other nonsmooth functions such as result from table lookups and piecewise rational approximations. For example, one would not expect to obtain an accurate approximation to the cosine function by applying automatic differentiation to the library subroutine for the sine. As with any powerful tool, automatic differentiation should not be expected to provide good results if applied indiscriminately, especially to 'legacy' code. As with interval arithmetic, automatic differentiation will yield the best results if applied to programs written with it in mind.

Current state of the art software for point automatic differentiation of programs are ADOL-C, for programs written in C/C++ [5], and ADIFOR for programs in Fortran 77 [1, pp. 385–392].

Numerous applications of automatic differentiation to optimization and other problems can be found in the conference proceedings [1,4], which also contain extensive bibliographies. An important result with implications for optimization is that automatic differentiation can be used to obtain Newton steps *without* forming Jacobians and solving linear systems, see [1, pp. 253–264].

From a historical standpoint, the principles of automatic differentiation go back to the early days of calculus, but implementation is a product of the computer age, hence the designation 'automatic'. The terminology 'algorithmic differentiation', to which the acronym automatic differentiation also applies, is perhaps better. Since differentiation is widely understood, automatic differentiation literature contains many anticipations and rediscoveries. The 1962 Stanford Ph.D. thesis of R.E. Moore deals with both interval arithmetic and automatic differentiation of code lists to obtain Taylor coefficients of series solution of systems of ordinary differential equations. In 1964, R.E. Wengert [15] published on automatic differentiation of code lists and noted that derivatives could be recovered from Taylor coefficients. Early results in automatic differentiation were applied to code lists in forward mode, as described

in [13]. G. Kedem [8] showed that automatic differentiation applies to subroutines and programs, again in forward mode. The reverse mode was anticipated by S. Linnainmaa in 1976 [10], and in the Ph.D. thesis of B. Speelpenning (Illinois, 1980), and published in more complete form by M. Iri in 1984 [7]. automatic differentiation via operator overloading and the concept of differentiation arithmetics, which are commutative rings with identity, were introduced by L.B. Rall [9, pp. 291–309], [14, pp. 73–90], [4, pp. 17–24]. For additional information about the early history of automatic differentiation, see [13] and the article by Iri [4, pp. 3–16] for later developments.

Analysis of algorithms for automatic differentiation has been carried out on the basis of graph theory by Iri [7], A. Griewank [12, pp. 128–161], [3], and equivalent matrix formulation by Rall [2, pp. 233–240].

See also

- Automatic Differentiation: Calculation of the Hessian
- Automatic Differentiation: Calculation of Newton Steps
- Automatic Differentiation: Geometry of Satellites and Tracking Stations
- Automatic Differentiation: Introduction, History and Rounding Error Estimation
- Automatic Differentiation: Parallel Computation
- Automatic Differentiation: Point and Interval Taylor Operators
- Automatic Differentiation: Root Problem and Branch Problem
- Bounding Derivative Ranges
- Global Optimization: Application to Phase Equilibrium Problems
- Interval Analysis: Application to Chemical Engineering Design Problems
- Interval Analysis: Differential Equations
- Interval Analysis: Eigenvalue Bounds of Interval Matrices
- Interval Analysis: Intermediate Terms
- Interval Analysis: Nondifferentiable Problems
- Interval Analysis: Parallel Methods for Global Optimization
- Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- Interval Analysis: Systems of Nonlinear Equations
- Interval Analysis: Unconstrained and Constrained Optimization
- Interval Analysis: Verifying Feasibility
- Interval Constraints
- Interval Fixed Point Theory
- Interval Global Optimization
- Interval Linear Systems
- Interval Newton Methods
- Nonlocal Sensitivity Analysis with Automatic Differentiation

References

1. Berz M, Bischof Ch, Corliss G, Griewank A (eds) (1996) Computational differentiation, techniques, applications, and tools. SIAM, Philadelphia
2. Fischer H, Riedmueller B, Schaeffler S (eds) (1996) Applied mathematics and parallel computing. Physica Verlag, Heidelberg
3. Griewank A (2000) Evaluating derivatives: Principles and techniques of algorithmic differentiation. SIAM, Philadelphia
4. Griewank A, Corliss GF (eds) (1991) Automatic differentiation of algorithms, theory, implementation, and application. SIAM, Philadelphia
5. Griewank A, Juedes D, Utke J (1996) ADOL-C, a package for the automatic differentiation of programs written in C/C++. ACM Trans Math Softw 22:131–167
6. Hansen E (1992) Global optimization using interval analysis. M. Dekker, New York
7. Iri M (1984) Simultaneous computation of functions, partial derivatives, and rounding errors: complexity and practicality. Japan J Appl Math 1:223–252
8. Kedem G (1980) Automatic differentiation of computer programs. ACM Trans Math Softw 6:150–165
9. Kulisch UW, Miranker WL (eds) (1983) A new approach to scientific computation. Acad. Press, New York
10. Linnainmaa S (1976) Taylor expansion of the accumulated rounding error. BIT 16:146–160
11. Moore RE (1979) Methods and applications of interval analysis. SIAM, Philadelphia
12. Pardalos PM (eds) (1993) Complexity in nonlinear optimization. World Sci, Singapore
13. Rall LB (1981) Automatic differentiation: Techniques and applications. Springer, Berlin
14. Ullrich C (eds) (1990) Computer arithmetic and self-validating numerical methods. Acad Press, New York
15. Wengert RE (1964) A simple automatic derivative evaluation program. Comm ACM 7:463–464

Automatic Differentiation: Point and Interval Taylor Operators

AD, Computational Differentiation

JAMES B. WALTERS, GEORGE F. CORLISS
Marquette University, Milwaukee, USA

MSC2000: 65K05, 90C30

Article Outline

Keywords
Introduction
  Operator Overloading
  Automatic Differentiation
Taylor Coefficients
Point and Interval Taylor Operators
  Design of Operators
  Use of Interval Operators
One-at-a-Time Coefficient Generation
Trade-Offs
See also
References

Keywords

Automatic differentiation; Code list; Interval arithmetic; Overloaded operator; Taylor series

Frequently of use in optimization problems, automatic differentiation may be used to generate Taylor coefficients. Specialized software tools generate Taylor series approximations, one term at a time, more efficiently than the general AD software used to compute (partial) derivatives. Through the use of operator overloading, these tools provide a relatively easy-to-use interface that minimizes the complications of working with both point and interval operations.

Introduction

First, we briefly survey the tools of automatic differentiation and operator overloading used to compute point- and interval-valued Taylor coefficients. We assume that f is an analytic function $f : \mathbf{R} \to \mathbf{R}$. Automatic differentiation (AD or computational differentiation) is the process of computing the derivatives of a function f at a point $t = t_0$ by applying rules of calculus for differentiation [9,10,17,18]. One way to implement AD uses overloaded operators.

Operator Overloading

An overloaded (or generic) operator invokes a procedure corresponding to the types of its operands. Most programming languages implement this technique for arithmetic operations. The sums of two floating point numbers, two integers, or one floating point number and one integer are computed using three different procedures for addition. Fortran 77 or C denies the programmer the ability to replace or modify the various routines used implicitly for integer, floating point, or mixed-operand arithmetic, but Fortran 95, C++, and Ada support operator overloading for user-defined types. Once we have defined an overloaded operator for each rule of differentiation, AD software performs those operations on program code for f, as shown below. The operators either propagate derivative values or construct a code list for their computation. We give prototypical examples of operators overloaded to propagate Taylor coefficients below.

Automatic Differentiation

The AD process requires that we have f in the form of an algorithm (e. g. computer program) so that we can easily separate and order its operations. For example, given $f(t) = e^t/(2 + t)$, we can express f as an algorithm in Fortran 95 or in C++ (using an assumed AD module or class):

In this section, we use AD to compute first derivatives. In the next section, we extend to point- and interval-valued Taylor series. To understand the AD process, we parse the program above into a sequence of unary and binary operations, called a *code list*, *computational graph*, or 'tape' [9]:

$$x_0 = t_0, \qquad x_2 = 2 + x_0,$$
$$x_1 = \exp(x_0), \qquad x_3 = \frac{x_1}{x_2}.$$

```
program Example1
    use AD_Module
    type(AD_Independent) :: t
        AD_Independent(0)
    type(AD_Dependent) :: f
    f = exp(t)/(2 + t)
end program Example1
#include 'AD_class.h'
void main (void) {
    AD_Independent t(0);
    AD_Dependent f;
    f = exp(t)/(2 + t);
}
```

Automatic Differentiation: Point and Interval Taylor Operators, Figure 1
Fortran and C++ calls to AD operators

Differentiation is a simple mechanical process for propagating derivative values. Let $t = t_0$ represent the value of the independent variable with respect to which we differentiate. We know how to take the derivative of a variable, a constant, and unary and binary operations (i. e. +, −, ∗, /, sin, cos, exp, etc.). Then AD software annotates the code list:

$$
\begin{aligned}
x_0 &= t_0; \\
\nabla x_0 &= 1, \\
x_1 &= \exp(x_0); \\
\nabla x_1 &= \exp(x_0) * \nabla x_0, \\
x_2 &= 2 + x_0; \\
\nabla x_2 &= 0 + \nabla x_0, \\
x_3 &= \frac{x_1}{x_2}; \\
\nabla x_3 &= \frac{(\nabla x_1 - \nabla x_2 * x_3)}{x_2}.
\end{aligned}
$$

AD propagates values of derivatives, not expressions as symbolic differentiation does. AD values are exact (up to round-off), not approximations of unknown quality as finite differences. For more information regarding AD and its applications, see [2,8,9,10,17,18], or the bibliography [21].

AD software can use overloaded operators in two different ways. Operators can propagate both the value x_i and its derivative ∇x_i, as suggested by the annotated code list above. This approach is easy to understand and to program. We give prototypical Taylor operators of this flavor below.

The second approach has the operators construct and store the code list. Various optimizations and parallel scheduling [1,4,12] may be applied to the code list. Then the code list is interpreted to propagate derivative values. This is the approach of AD tools such as *ADOL-C* [11], *ADOL-F* [20], *AD01* [16], or *INTOPT_90* [13]. The second approach is much more flexible, allowing the code list to be traversed in either the forward or reverse modes of AD (see [9]) or with various arithmetics (e. g. point- or interval-valued series).

AD may be applied to functions of more than one variable, in which partial derivatives with respect to each are computed in turn, and to vector functions, in which the component functions are differentiated in succession. In addition, we can compute higher order derivative values. One application of AD involving higher order derivatives of f is the computation of Taylor (series) coefficients to which we turn in the next section.

Source code transformation is a third approach to AD software used by *ATOMFT* [5] for Taylor series and by *ADIFOR* [3], *PADRE2* [14], or *Odyssée* [19] for partial derivatives. Such tools accept the algorithm for f as data, rather than for execution, and produce code for computing the desired derivatives. The resulting code often executes more rapidly than code using overloaded operators.

Taylor Coefficients

We define the Taylor coefficients of the analytic function f at the point $t = t_0$:

$$(f|t_0)_i := \frac{1}{i!}\frac{d^i f(t_0)}{dt^i},$$

for $i = 0, 1, \ldots$, and let $F := ((f|t_0)_i)$ denote the vector of Taylor coefficients. Then *Taylor's theorem* says

that there exists some point τ (usually not practically obtainable) between t and t_0 such that

$$f(t) = \sum_{i=0}^{p} (f|t_0)_i (t - t_0)^i + \frac{1}{(p+1)!} \frac{d^{p+1} f(\tau)}{dt^{p+1}} (t - t_0)^{p+1}. \tag{1}$$

Computation of Taylor coefficients requires differentiation of f. We generate Taylor coefficients automatically using recursion formulas for unary and binary operations. For example, the recurrences we need for our example $f(t) = e^t/(2 + t)$ are

$$\begin{aligned}
&x(t) = \exp u(t) \Rightarrow x' = xu', \\
&\quad (x)_0 = \exp(u)_0, \\
&\quad (x)_i = \sum_{j=0}^{i-1} (x)_j * (u)_{i-j} * \frac{(i-j)}{i}; \\
&x(t) = u(t) + v(t), \\
&\quad (x)_i = (u)_i + (v)_i; \\
&x(t) = \frac{u(t)}{v(t)} \Rightarrow xv = u, \\
&\quad (x)_i = \frac{\left((u)_i - \sum_{j=0}^{i-1} (x)_j * (v)_{i-j}\right)}{(v)_0}.
\end{aligned}$$

The recursion relations are described in more detail in [17]. Except for + and −, each recurrence follows from Leibniz' rule for the Taylor coefficients of a product. The relations can be viewed as a lower triangular system. The recurrence represents a solution by forward substitution, but there are sometimes accuracy or stability advantages in an iterative solution to the lower triangular system. The recurrences for each operation can be evaluated in floating-point, complex, interval, or other appropriate arithmetic.

To compute the formal series for $f(t) = e^t/(2 + t)$ expanded at $t = 0$,

$$\begin{cases}
X_0 := (t_0, 1, 0, \ldots)(0, 1, 0, \ldots), \\
X_1 := \exp X_0 = \left(1, 1, \frac{1}{2!}, \frac{1}{3!}, \ldots\right), \\
X_2 := 2 + X_0 = (2, 1, 0, \ldots), \\
X_3 := \frac{X_2}{X_3} = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{48}, \ldots\right).
\end{cases} \tag{2}$$

```
class Taylor {      // Or make a template:
   private:
      cont int Max_Length = 20;
      Value_type coef[Max_Length];
   public:
      Taylor ( Value_type t_0 ) {
            // Constructor for Independents
         coef[0] = t_0; coef[1] = 1;
         for(int i  =  2;  i ¡ Max_Length;  i++)
            { coef[i] = 0;  }
      }
      Taylor ( void ) {
            // Constructor for Dependents
         for (int i = 0; i ¡  Max_Length; i++)
            { coef[i] = 0; }
      }
      Taylor ( Taylor &U) {
            // Copy Constructor
         for (int i=0; i ¡  Max_Length; i++)
            { coef[i] = Value_type(U.coef)[i]; }
      }
      friend Taylor operator +
            (int u, Taylor V) {
         V.coef[0] += u; return V;
      }
      friend Taylor operator /
            (Taylor U, Taylor V) {
         Taylor X;
         for (int i = 0; i ¡  Max_Length; i++) {
            Value_type sum = U.coef[i];
            for (int j = 0; j ¡  i; j++)
               { sum — =X.coef[j] * V.coef[i—j]; }
            X.coef[i] = sum / V.coef[0];
         }
         return X;
      }
      friend Taylor exp (Taylor U)
         { /* Similar to divide */ }
      Value_type getCoef (int i)
         { return coef[i]; }
};   // end class Taylor
```

Point and Interval Taylor Operators

As foreshadowed by this example, we define an abstract data type for Taylor series and use operator overloading to define actions on objects of that type using previously defined floating-point and interval operations.

Design of Operators

In this section, we give prototypical operators for the direct propagation of Taylor coefficients such as might be called from code similar to that shown in Fig. 1. Direct propagation of values works by translating each operation into a call to the appropriate AD routine at compile time. Thus, simply compiling the source code for f and linking it with the overloaded operator routines creates a program that computes the Taylor coefficients of f at $t = t_0$. For illustration, we provide only a stripped-down prototype with operators required for the example $f(t) = e^t/(2 + t)$. We suppress issues of references and the like that are essential to the design of a useful class. See [6] for a description of a set of interval Taylor operators in Ada.

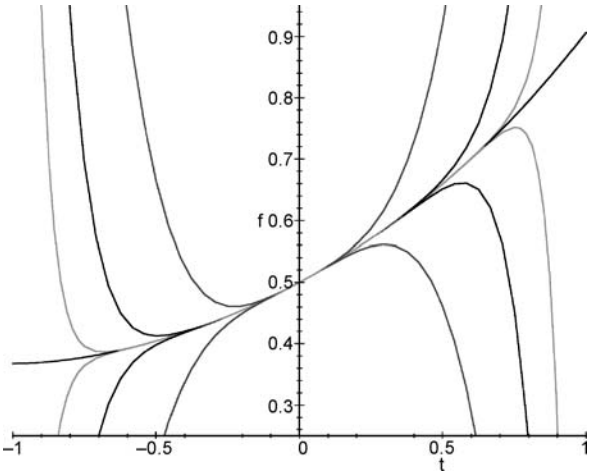
If instead, operators for AD_type record a code list, then an interpreter reads each node from the code list and calls the appropriate operator from class Taylor:

```
Taylor Operand[MemSize];
for (int i = 0; ¡ CodeSize; i++) {
   Node = getNextOperation ();
   switch (Node.OpCode) {
      case PLUS : Operand[Node.Result]
         = Operand[Node.Left]
            + Operand[Node.Right];
         break;
      …
      case EXP : Operand[Node.Result]
         = exp ( Operand[Node.Left] );
         break;
      …
   }
}
```

Use of Interval Operators

We have mentioned the possibility of working with interval values but not the significance of doing so. From equation (1) for an interval $\mathbf{t}$, and for all $t \in \mathbf{t}$,

$$f(t) \in \sum_{i=0}^{p} (f|t_0)_i (t - t_0)^i + \frac{1}{(p+1)!} \frac{d^{p+1} f(\mathbf{t})}{dt^{p+1}} (t - t_0)^{p+1}. \quad (3)$$



Automatic Differentiation: Point and Interval Taylor Operators, Figure 2
Taylor series enclosures of *f*

In a computer implementation, the summation is done in interval arithmetic to ensure enclosure. The series Taylor coefficients $(f|\mathbf{t}_0)_i$ are narrow intervals whose width comes only from outward rounding. The remainder term is the Taylor coefficient $(f|\mathbf{t})_i$, where the recurrence relations are evaluated in interval arithmetic. The series (3) can be used to bound the range of f, for validated quadrature [7], or for rigorous solution of ODEs [15]. For the example $f(t) = e^t/(2 + t)$, we repeat the sequence of computations of Equation (2) for the interval $\mathbf{t}_0 = [0, 0]$ and for $\mathbf{t} = [-1, 1]$:

$$\left((f|[0])_i\right) = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{48}, \ldots\right),$$

$$\left((f|[-1,1])_i\right) = ([0.12, 2.72], [-2.59, 2.68], [-2.64, 4.04], \ldots).$$

Assembling these according to (3) yields enclosures for all $t \in [-1, 1]$:

$$\begin{aligned} f(t) &\in (f|[-1,1])_0 = [0.12, 2.72] \\ &\in (f|[0])_0 + (f|[-1,1])_1 (t - 0) \\ &= \frac{1}{2} + [-2.59, 2.68]t \\ &\in (f|[0])_0 + (f|[0])_1 (t - 0) \\ &\quad + (f|[-1,1])_2 (t - 0)^2 \\ &= \frac{1}{2} + \frac{1}{4}t + [-2.64, 4.04]t^2 \\ &\vdots \end{aligned}$$

To demonstrate the true power of this approximation technique, we plot the corresponding 5, 10, and 20 term enclosures in Fig. 2.

One-at-a-Time Coefficient Generation

The Taylor operators described the preceding section accept vectors of p Taylor coefficients for operands u and v and return Taylor coefficients for result x with *complexity* $O(p^2)$. However, for applications such as ODEs or order-adaptive quadrature, the entire operand series is not known, and we need to compute terms one at a time [6]. For example, for the DE

$$u' = f(t,u) = \frac{\exp(u)}{(2+t)}, \quad u(0) = 1,$$

initial condition $u(0) = 1$ implies

$$(u|0)_0 = 1,$$

and DE $u' = \exp(u)/(2+t)$ implies

$$(u|0)_1 = \frac{\exp(1)}{(2+0)} = \frac{e}{2},$$

$u'' = \frac{u'\exp(u)}{2+t} - \frac{\exp(u)}{(2+t)^2}$ implies

$$(u|0)_2 = \frac{e}{2}\frac{\exp(1)}{(2+0)} - \frac{e}{4} = \frac{e(e-1)}{4},$$

etc.

Successive terms can be computed by interpreting the code list for $f(t,u)$ repeatedly for series of increasing length for u. Each iteration of the automatic generation process yields an additional Taylor coefficient. Unfortunately, a simple implementation of Taylor operators has complexity $O(p^3)$ because already known coefficients of u' are recomputed. However, since the order of operations is the same in each iteration, we can increase the efficiency of the computations by storing intermediate results [6]. Each overloaded operator routine calls a memory allocation procedure that refers it to the next space in an array. If that space is empty, we store Taylor coefficient values for that variable. Otherwise, the space must contain the previously computed Taylor coefficients of that variable, which we can then use to more quickly compute the next coefficient in the set. With clever book-keeping, we compute p floating-point or interval-valued Taylor coefficients one at a time in $O(p^2)$ time.

Trade-Offs

We may strive for three goals when writing software for point and interval Taylor operations: storage space efficiency, time efficiency, and ease of use. These three factors are often at odds with each other.

Carefully implemented operator overloading provides an easy to use interface and provides reasonable time and space efficiency. We may achieve greater time and space efficiency by using source code transformation.

In conclusion, automatic differentiation through Taylor operators shows merit as a technique for computing guaranteed interval enclosures about a function f. Further efforts to refine this technique may provide us with a tool that handles multivariate functions, and runs significantly faster thanks to parallelization and improved optimization techniques.

See also

- ▶ Automatic Differentiation: Calculation of the Hessian
- ▶ Automatic Differentiation: Calculation of Newton Steps
- ▶ Automatic Differentiation: Geometry of Satellites and Tracking Stations
- ▶ Automatic Differentiation: Introduction, History and Rounding Error Estimation
- ▶ Automatic Differentiation: Parallel Computation
- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Root Problem and Branch Problem
- ▶ Bounding Derivative Ranges
- ▶ Global Optimization: Application to Phase Equilibrium Problems
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Interval Analysis: Differential Equations
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Interval Analysis: Intermediate Terms
- ▶ Interval Analysis: Nondifferentiable Problems
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ▶ Interval Analysis: Systems of Nonlinear Equations

- ▶ Interval Analysis: Unconstrained and Constrained Optimization
- ▶ Interval Analysis: Verifying Feasibility
- ▶ Interval Constraints
- ▶ Interval Fixed Point Theory
- ▶ Interval Global Optimization
- ▶ Interval Linear Systems
- ▶ Interval Newton Methods
- ▶ Nonlocal Sensitivity Analysis with Automatic Differentiation

References

1. Benary J (1996) Parallelism in the reverse mode. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 137–147
2. Berz M, Bischof Ch, Corliss G, Griewank A (eds) (1996) Computational differentiation: Techniques, applications, and tools. SIAM, Philadelphia
3. Bischof Ch, Carle A, Khademi PM, Mauer A, Hovland P (1994) ADIFOR: 2.0 user's guide. Techn Memorandum Math and Computer Sci Div Argonne Nat Lab ANL/MCS-TM-192
4. Bischof Ch, Haghighat MR (1996) Hierarchical approaches to automatic differentiation. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 83–94
5. Chang YF, Corliss GF (1994) ATOMFT: Solving ODEs and DAEs using Taylor series. Comput Math Appl 28:209–233
6. Corliss GF (1991) Overloading point and interval Taylor operators. In: Griewank A, Corliss GF (eds) Automatic Differentiation of Algorithms: Theory, Implementation, and Application. SIAM, Philadelphia, pp 139–146
7. Corliss GF, Rall LB (1987) Adaptive, self-validating quadrature. SIAM J Sci Statist Comput 8(5):831–847
8. Griewank A (1989) On automatic differentiation. In: Iri M, Tanabe K (eds) Mathematical Programming: Recent Developments and Applications. Kluwer, Dordrecht, pp 83–108
9. Griewank A (1991) The chain rule revisited in scientific computing. SIAM News 24(3):20 Also: Issue 4, page 8.
10. Griewank A, Corliss GF (eds) (1991) Automatic differentiation of algorithms: Theory, implementation, and application. SIAM, Philadelphia
11. Griewank A, Juedes D, Utke J (1996) ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. ACM Trans Math Softw 22(2):131–167
12. Griewank A, Reese S (1991) On the calculation of Jacobian matrices by the Markowitz rule. In: Griewank A, Corliss GF (eds) Automatic Differentiation of Algorithms: Theory, Implementation, and Application. SIAM, Philadelphia, pp 126–135
13. Kearfott RB (1995) A Fortran 90 environment for research and prototyping of enclosure algorithms for nonlinear equations and global optimization. ACM Trans Math Softw 21(1):63–78
14. Kubota K (1996) PADRE2-Fortran precompiler for automatic differentiation and estimates of rounding error. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, 367–374
15. Lohner RJ (1987) Enclosing the solutions of ordinary initial and boundary value problems. In: Kaucher EW, Kulisch UW, Ullrich C (eds) Computer Arithmetic: Scientific Computation and Programming Languages. Wiley and Teubner, Stuttgart, pp 255–286
16. Pryce JD, Reid JK (1997) AD01: A Fortran 90 code for automatic differentiation. Techn Report Rutherford–Appleton Lab RAL-TR-97xxx
17. Rall LB (1981) Automatic differentiation: Techniques and applications. Lecture Notes Computer Sci, vol 120. Springer, Berlin
18. Rall LB, Corliss GF (1996) An introduction to automatic differentiation. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 1–17
19. Rostaing N, Dalmas S, Galligo A (1993) Automatic differentiation in Odyssée. Tellus 45A:558–568
20. Shiriaev D (1996) ADOL-F: Automatic differentiation of Fortran codes. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 375–384
21. Yang W, Corliss G (1996) Bibliography of computational differentiation. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 393–418

Automatic Differentiation: Root Problem and Branch Problem

HERBERT FISCHER
Fakult. Math., Techn. University München, München, Germany

MSC2000: 65K05

Article Outline

Keywords
Root Problem
Branch Problem
See also
References

Keywords

Automatic differentiation; Root problem; Branch problem

Automatic differentiation is a method in which a program for evaluating a function f is transformed into another program that evaluates both the function f and some of its derivatives. The key idea is the repeated use of the chain-rule for composing the derivatives of f from derivatives of parts of f. For more about automatic differentiation (AD), consult [2,3,5].

Proper combinations of differentiable functions produce differentiable functions. Some combinations of nondifferentiable functions also produce differentiable functions. Therefore the mere fact that a program defines a differentiable function is no guarantee that AD will work. Here we investigate two cases, where AD, applied to a program for a differentiable function, fails.

The *root problem* arises when a square-root is combined with other functions so that the resulting function is differentiable but the chain-rule is not applicable for certain arguments.

The *branch problem* arises when a program for evaluating a differentiable function f employs statements of the form $B(x)$ then $S1$ else $S2$, where x is from the domain of f, B is a Boolean function, and $S1$ and $S2$ represent subprograms. This reflects a piece-wise definition of the function f, and the derivative of one or the other piece may be quite different from the derivative of the function f.

Root Problem

An example that is typical of the root problem is shown in Table 1. The program P defines the function

$$f\colon\ \mathbb{R}^2 \to \mathbb{R}$$

with

$$f(x) = \sqrt{x_1^4 + x_2^4}.$$

This function is differentiable at any $x \in \mathbf{R}^2$, in particular $f'(0) = [0, 0]$. Standard AD (in the forward mode) transforms P into a program P' by inserting assignment statements for derivatives in proper places (see Table 2).

The program P' is supposed to compute $f(x)$ and $f'(x)$. But for $x = 0$ it does *not* compute the correct value $f'(0) = [0, 0]$, but rather it fails because of division by zero.

One can easily see that this failure is not limited to the forward mode, because the reverse mode encounters the same division-by-zero problem. Symbolic manipulation packages such as MAPLE also fail to produce $f'(0)$.

Automatic Differentiation: Root Problem and Branch Problem, Table 1
Program *P* for evaluating *f* at *x*

input: $x = (x_1, x_2) \in \mathbf{R}^2$		
y_1	$\leftarrow$	x_1
y_2	$\leftarrow$	x_2
y_3	$\leftarrow$	y_1^4
y_4	$\leftarrow$	y_2^4
y_5	$\leftarrow$	$y_3 + y_4$
y_6	$\leftarrow$	$\sqrt{y_5}$
$f(x)$	$\leftarrow$	y_6
output: $f(x)$		

Automatic Differentiation: Root Problem and Branch Problem, Table 2
Program *P′* for evaluating *f* and *f′* at *x*

input: $x = (x_1, x_2) \in \mathbf{R}^2$					
y_1	$\leftarrow$	x_1	y_1'	$\leftarrow$	$[1,0]$
y_2	$\leftarrow$	x_2	y_2'	$\leftarrow$	$[0,1]$
y_3	$\leftarrow$	y_1^4	y_3'	$\leftarrow$	$4y_1^3 \cdot y_1'$
y_4	$\leftarrow$	y_2^4	y_4'	$\leftarrow$	$4y_2^3 \cdot y_2'$
y_5	$\leftarrow$	$y_3 + y_4$	y_5'	$\leftarrow$	$y_3' + y_4'$
y_6	$\leftarrow$	$\sqrt{y_5}$	y_6'	$\leftarrow$	$\frac{1}{2\sqrt{y_5}} \cdot y_5'$
$f(x)$	$\leftarrow$	y_6	$f'(x)$	$\leftarrow$	y_6'
output: $f(x)$			output: $f'(x)$		

Automatic Differentiation: Root Problem and Branch Problem, Table 3
Program *Q* for evaluating *f* at *x*

input: $x \in D \subseteq \mathbf{R}^n$		
y_1	$\leftarrow$	$A(x)$
y_2	$\leftarrow$	$\sqrt{y_1}$
y_3	$\leftarrow$	$B(x, y_2)$
$f(x)$	$\leftarrow$	y_3
output: $f(x)$		

A more general setting for the root problem is shown in Table 3. Here, it is assumed that:

1) D_A is a nonempty open subset of $\mathbf{R}^n$;
2) the function A: $D_A \subseteq \mathbf{R}^n \to \mathbf{R}$ is differentiable;
3) D_B is a nonempty open subset of $\mathbf{R}^{n+1}$;
4) the function B: $D_B \subseteq \mathbf{R}^{n+1} \to \mathbf{R}$ is differentiable;
5) $D := \{x \in D_A : A(x) \geq 0, (x, A(x)) \in D_B\}$;
6) D is nonempty.

The program Q defines the function

$$f:\ D \subseteq \mathbb{R}^n \to \mathbb{R}$$

with

$$f(x) = B(x, \sqrt{A(x)}).$$

Standard AD (in the forward mode) transforms Q into a program Q'. The steps of Q' in evaluating $f'(x)$ can be seen in the formula

$$f'(x) = B_1(x, y_2) + B_2(x, y_2) \cdot \left(\frac{1}{2\sqrt{y_1}} \cdot y_1' \right),$$

where $[B_1(x, y_2), B_2(x, y_2)]$ is an appropriate partition of $B'(x, y_2)$. For $x \in D$ with $A(x) > 0$, the program Q' will produce $f'(x)$. And for $x \in D$ with $A(x) = 0$, the program Q' fails because of division by zero. The case in which $x^* \in D$ with $A(x^*) = 0$ is ambiguous. It says nothing about the existence of $f'(x^*)$. In this case, we distinguish the following four situations:

A) $f'(x^*)$ does not exist, for instance $n = 2$, $A(x) = x_1^2 + x_2^2$ and $B(x, y) = y$, $x^* = 0$.
B) A alone guarantees existence of $f'(x^*)$, for instance $n = 2$, $A(x) = x_1^4 + x_2^4$, $x^* = 0$.
C) B alone guarantees existence of $f'(x^*)$, for instance $B(x, y) = y^2$.
D) A and B together guarantee existence of $f'(x^*)$, for instance $n = 2$, $A(x) = x_1^2 + x_2^2$ and $B(x, y) = x_1 \cdot x_2 \cdot y$, $x^* = 0$.

What can be done to resolve the root problem?

The use of AD tools for higher derivatives may be helpful. Consider the simple case $n = 1$, $A \in \mathcal{C}^\infty$, $D_B = \mathbf{R}^{n+1}$, $B(x, y) = y$. So we have

$$D := \{x:\ x \in D_A,\ A(x) \geq 0\}$$

and f: $D \subseteq \mathbf{R} \to \mathbf{R}$ with $f(x) = \sqrt{A(x)}$.

Assume that for $x \in \mathbf{R}$ it can be decided whether or not $x \in D$, for instance by testing x in a program for evaluating A.

For $x^* \in D$, we require the value of the derivative $f'(x^*)$. Below, we list the relevant implications:

- $A(x^*) > 0 \Rightarrow f'(x^*) = \frac{1}{2\sqrt{A(x^*)}} \cdot A'(x^*)$.
- $A(x^*) = 0 \Rightarrow$ no answer possible.
- $A(x^*) = 0, A'(x^*) \neq 0 \Rightarrow f'(x^*)$ does not exist.
- $A(x^*) = 0, A'(x^*) = 0 \Rightarrow$ no answer possible.
- $A(x^*) = 0, A'(x^*) = 0, A''(x^*) \neq 0 \Rightarrow f'(x^*)$ does not exist.
- $A(x^*) = 0, A'(x^*) = 0, A''(x^*) = 0 \Rightarrow$ no answer possible.
- $A(x^*) = 0, A'(x^*) = 0, A''(x^*) = 0, A'''(x^*) \neq 0 \Rightarrow f'(x^*)$ does not exist.
- $A(x^*) = 0, A'(x^*) = 0, A''(x^*) = 0, A'''(x^*) = 0 \Rightarrow$ no answer possible.
- $A(x^*) = 0, A'(x^*) = 0, A''(x^*) = 0, A'''(x^*) = 0, A^{(4)}(x^*) > 0 \Rightarrow f'(x^*) = 0$.
- $A(x^*) = 0, A'(x^*) = 0, A''(x^*) = 0, A''(x^*) = 0, A^{(4)}(x^*) < 0 \Rightarrow f'(x^*)$ does not exist.
- $A(x^*) = 0, A'(x^*) = 0, A''(x^*) = 0, A'''(x^*) = 0, A^{(4)}(x^*) = 0 \Rightarrow$ no answer possible.

Let $n \in \{1, 2, 3 \ldots\}$ and $A^{(k)}(x^*) = 0$ for $k = 0, \ldots, 2n$.

- $A^{(2n+1)}(x^*) \neq 0 \Rightarrow f'(x^*)$ does not exist.
- $A^{(2n+1)}(x^*) = 0, A^{(2n+2)} > 0 \Rightarrow f'(x^*) = 0$.
- $A^{(2n+1)}(x^*) = 0, A^{(2n+2)} < 0 \Rightarrow f'(x^*)$ does not exist.
- $A^{(2n+1)}(x^*) = 0, A^{(2n+2)} = 0 \Rightarrow$ no answer possible.

For a nonstandard treatment of these implications see [6]. Of course in the general situation given in Table 3, the classification of cases is more problematic.

Branch Problem

A typical example for the branch problem is Gauss-elimination for solving a system of linear equations with parameters. For illustrative purposes, it suffices to consider two equations with a two-dimensional parameter x (see Table 4). Here, it is assumed that:

a) D is a nonempty open subset of $\mathbf{R}^2$;
b) the function M: $D \subseteq \mathbf{R}^2 \to \mathbf{R}^{2,2}$ is differentiable;
c) the function R: $D \subseteq \mathbf{R}^2 \to \mathbf{R}^2$ is differentiable;
d) $x \in D \Rightarrow$ the matrix $M(x)$ is regular.

The program GAUSS defines the function

$$F:\ D \subseteq \mathbb{R}^2 \to \mathbb{R}^2$$

with

$$M(x) \cdot F(x) = R(x).$$

Automatic Differentiation: Root Problem and Branch Problem, Table 4
Program GAUSS for evaluating *f* at *x*

	input: $x \in D$		
	M11	$\leftarrow$	$M_{11}(x)$
	M12	$\leftarrow$	$M_{12}(x)$
	M21	$\leftarrow$	$M_{21}(x)$
	M22	$\leftarrow$	$M_{22}(x)$
	R1	$\leftarrow$	$R_1(x)$
	R2	$\leftarrow$	$R_2(x)$
	IF M11 $\neq$ 0 THEN		
S1:	E	$\leftarrow$	M21 / M11
	M22	$\leftarrow$	M22 $-$ E $*$ M12
	R2	$\leftarrow$	R2 $-$ E $*$ R1
	F2	$\leftarrow$	R2 / M22
	F1	$\leftarrow$	(R1 $-$ M12 $*$ F2) / M11
	ELSE		
S2:	F2	$\leftarrow$	R1 / M12
	F1	$\leftarrow$	(R2 $-$ M22 $*$ F2) / M21
	output: $F(x)$ = (F1,F2)		

Since the matrix $M(x)$ is regular for $x \in D$, the program GAUSS and the function f are well-defined. Furthermore, the function f is differentiable.

Standard AD (in the forward mode) transforms GAUSS into a new program by inserting assignment statements for derivatives in proper places. The resulting program GAUSS' is also well-defined, and for $x \in D$ it is supposed to produce $F(x)$ and $F'(x)$.

Now choose

$$D = \left\{x \in \mathbb{R}^2 \colon\ 0 < x_1 < 2,\ 0 < x_2 < 2\right\}$$

and

$$M(x) = \begin{array}{|c|c|}\hline M_{11}(x) & M_{12}(x) \\ \hline M_{21}(x) & M_{22}(x) \\ \hline \end{array} = \begin{array}{|c|c|}\hline x_1 - x_2 & 1 \\ \hline 10 & x_1 + x_2 \\ \hline \end{array},$$

$$R(x) = \begin{array}{|c|}\hline R_1(x) \\ \hline R_2(x) \\ \hline \end{array} = \begin{array}{|c|}\hline 100(x_1 + 2x_2) \\ \hline 100(x_1 - 2x_2) \\ \hline \end{array}.$$

It is easy to see that D is a nonempty open subset of $\mathbf{R}^2$, that the functions M and R are differentiable, and that $M(x)$ is regular for $x \in D$.

GAUSS' produces

$$F'(1,1) = \begin{array}{|c|c|}\hline -40 & -90 \\ \hline 100 & 200 \\ \hline \end{array},$$

but the correct value is

$$F'(1,1) = \begin{array}{|c|c|}\hline -54 & -76 \\ \hline 170 & 130 \\ \hline \end{array}.$$

One can easily check that the wrong result is not limited to the forward mode, because the reverse mode yields exactly the same wrong result.

To better understand the situation we define

$$D_1 := \{x \colon\ x \in D, M_{11}(x) \neq 0\},$$
$$D_2 := \{x \colon\ x \in D, M_{11}(x) = 0\}.$$

The program GAUSS can be considered as a piecewise definition of the function F,

$$F(x) = \begin{cases} F(x) \text{ according to S1}, & \text{for } x \in D_1, \\ F(x) \text{ according to S2}, & \text{for } x \in D_2. \end{cases}$$

Normally, one is not too concerned about the domain of a function. But indeed in this case, we must be concerned.

Let $F|_{D_1}$ denote the restriction of F to D_1 and let $F|_{D_2}$ denote the restriction of F to D_2. Then, of course

$$F(x) = \begin{cases} (F|_{D_1})(x) & \text{for } x \in D_1, \\ (F|_{D_2})(x) & \text{for } x \in D_2. \end{cases}$$

The domain D_1 of the function $F|_{D_1}$ is an open set, $x \in D_1$ is an interior point of D_1, and hence

$$F'(x) = (F|_{D_1})'(x) \quad \text{for } x \in D_1,$$

and this is the value GAUSS' produces.

The domain D_2 of the function $F|_{D_2}$ is too thin, it has no interior points, and hence $F|_{D_2}$ is not differentiable. In other words, the function $F|_{D_2}$ does not provide enough information to obtain $F'(x)$ for $x \in D_2$. Thus GAUSS' *cannot* produce $F'(x)$ for $x \in D_2$. What GAUSS' actually presents for $F'(x)$ is the value for the derivative of another function, which is of no interest here. For more see [1].

In [4] it is claimed that the use of a certain branching function method makes the branch problem vanish.

This is true in certain cases, in our example the branching function method fails because it encounters division by zero. At least this suggests that something went wrong. For a partial solution to the branch problem, see [1] and for a nonstandard treatment of the branch problem, see [6].

A simple example of the branch problem is shown in the informal program

IF $x \neq 1$ THEN $f(x) \leftarrow x \cdot x$
ELSE $f(x) \leftarrow 1$.

This program defines the function

$$f\colon\ \mathbb{R} \to \mathbb{R} \quad \text{with } f(x) = x^2.$$

Of course, f is differentiable, in particular we have $f'(1) = 2$.

Standard AD software produces the wrong result $f'(1) = 0$. It is not surprising that symbolic manipulation packages produce the same wrong result. Here it is obvious that the else-branch does not carry enough information for computing the correct $f'(1)$.

Sometimes branching is done to *save work*. Consider the function

$$f\colon\ D \subseteq \mathbb{R}^n \to \mathbb{R}$$

with

$$f(x) = s(x) + c(x) \cdot E(x),$$

where D is an open set. The real-valued functions s, c, E may be given explicitly or by subroutines. Assume that $f(x)$ has to be evaluated many times for varying x-s, that $c(x) = 0$ for many interesting values of x, and that $E(x)$ is computationally costly. Then it is effective to set up a program for computing $f(x)$ as shown in Table 5.

Assume that the functions s, c, E are differentiable. Then f is differentiable too. For given $x \in D$ we ask for $f'(x)$.

Standard AD (in the forward mode) transforms SW into a new program by inserting assignment statements concerning derivatives. The resulting program SW' is well-defined, and for given $x \in D$ it is supposed to produce $f(x)$ and $f'(x)$.

Define the sets

$$D_1 := \{x\colon\ x \in D,\ c(x) \neq 0\},$$
$$D_2 := \{x\colon\ x \in D,\ c(x) = 0\}.$$

Automatic Differentiation: Root Problem and Branch Problem, Table 5
Program *SW* for computing *f*(*x*)

	input: $x \in D$		
	$c(x)$	$\leftarrow$	$\cdots$
	IF $c(x) \neq 0$ THEN		
S1:	$s(x)$	$\leftarrow$	$\cdots$
	$E(x)$	$\leftarrow$	$\cdots$
	$r(x)$	$\leftarrow$	$s(x) + c(x) \cdot E(x)$
	$f(x)$	$\leftarrow$	$r(x)$
	ELSE		
S2:	$s(x)$	$\leftarrow$	$\cdots$
	$f(x)$	$\leftarrow$	$s(x)$
	output: $f(x)$		

SW' works correctly to produce

$$f'(x) = r'(x) \quad \text{for } x \in D_1.$$

Looking at SW, it is tempting to assume:

$$f'(x) = s'(x) \quad \text{for } x \in D_2$$

and SW' actually follows this assumption. But it is clear that

$$f'(x) = s'(x) + E(x) \cdot c'(x) + c(x) \cdot E'(x) \quad \text{for } x \in D,$$

and in particular

$$f'(x) = s'(x) + E(x) \cdot c'(x) \quad \text{for } x \in D_2.$$

If $x \in D_2$, and if either $E(x) = 0$ or $c'(x) = 0$, then SW' produces the correct $F'(x)$, otherwise SW' fails.

See also

- ▶ Automatic Differentiation: Calculation of the Hessian
- ▶ Automatic Differentiation: Calculation of Newton Steps
- ▶ Automatic Differentiation: Geometry of Satellites and Tracking Stations
- ▶ Automatic Differentiation: Introduction, History and Rounding Error Estimation
- ▶ Automatic Differentiation: Parallel Computation

- ► Automatic Differentiation: Point and Interval
- ► Automatic Differentiation: Point and Interval Taylor Operators
- ► Nonlocal Sensitivity Analysis with Automatic Differentiation

References

1. Beck T, Fischer H (1994) The if-problem in automatic differentiation. J Comput Appl Math 50:119–131
2. Berz M, Bischof Ch, Corliss GF, Griewank A (eds) (1996) Computational differentiation: Techniques, applications, and tools. SIAM, Philadelphia
3. Griewank A, Corliss GF (eds) (1991) Automatic differentiation of algorithms: Theory, implementation, and application. SIAM, Philadelphia
4. Kearfott RB (1996) Rigorous global search: Continuous problems. Kluwer, Dordrecht
5. Rall LB (1981) Automatic differentiation: Techniques and applications. Lecture Notes Computer Sci, vol 120. Springer, Berlin
6. Shamseddine K, Berz M (1996) Exception handling in derivative computation with nonarchimedean calculus. In: Berz M, Bischof Ch, Corliss GF, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 37–51

B

Bayesian Global Optimization

BA

JONAS MOCKUS
Institute Math. and Informatics, Vilnius, Lithuania

MSC2000: 90C26, 90C10, 90C15, 65K05, 62C10

Article Outline

Keywords
See also
References

Keywords

Global optimization; Discrete optimization; Bayesian approach; Heuristics

The traditional numerical analysis considers optimization algorithms which guarantee some accuracy for all functions to be optimized. This includes the exact algorithms (that is the worst-case analysis). Limiting the maximal error requires a computational effort that often increases exponentially with the size of the problem. An alternative is *average case analysis* where the average error is made as small as possible. The average is taken over a set of functions to be optimized. The average case analysis is called the *Bayesian approach* (BA) [7,14].

There are several ways of applying the BA in optimization. The direct Bayesian approach (DBA) is defined by fixing a *prior distribution* P on a set of functions $f(x)$ and by minimizing the Bayesian risk function $R(x)$ [6,14]. The risk function describes the average deviation from the global minimum. The distribution P is regarded as a stochastic model of $f(x)$, $x \in \mathbf{R}^m$, where $f(x)$ might be a deterministic or a stochastic function. In the Gaussian case assuming (see [14] that the $(n+1)$th observation is the last one

$$R(x) = \frac{1}{\sqrt{2}\pi s_n(x)} \int_{-\infty}^{+\infty} \min(c_n, z) e^{-\frac{1}{2}\left(\frac{y - m_n(x)}{s_n(x)}\right)^2} \, \mathrm{d}z, \tag{1}$$

Here, $c_n = \min_i z_i - \epsilon$, $z_i = f(x_i)$, $m_n(x)$ is the conditional expectation given the values of z_i, $i = 1, \ldots n$, $d_n(x)$ is the conditional variance, and $\epsilon > 0$ is a correction parameter.

The objective of DBA (used mainly in continuous cases) is to provide as small average error as possible while keeping the convergence conditions.

The *Bayesian heuristic approach* (BHA) means fixing a prior distribution P on a set of functions $f_K(x)$ that define the best values obtained using K times some heuristic $h(x)$ to optimize a function $v(y)$ of variables $y \in \mathbf{R}^n$ [15]. As usual, the components of y are discrete variables. The heuristic $h(x)$ defines an expert opinion about the decision priorities. It is assumed that the heuristics or their 'mixture' depend on some continuous parameters $x \in \mathbf{R}^m$, where $m < n$.

The Bayesian stopping rules (BSR) [3] define the best on average stopping rule. In the BSR, the prior distribution is determined regarding only those features of the objective function $f(x)$ which are relevant for the stopping of the algorithm of global optimization.

Now all these ways will be considered in detail starting from the DBA. The Wiener process is common [11,16,19] as a stochastic model applying the DBA in the one-dimensional case $m = 1$.

The *Wiener model* implies that almost all the sample functions $f(x)$ are continuous, that increments $f(x_4) - f(x_3)$ and $f(x_2) - f(x_1)$, $x_1 < x_2 < x_3 < x_4$ are stochasti-

cally independent, and that $f(x)$ is Gaussian $(0, \sigma\ x)$ at any fixed $x > 0$. Note that the Wiener process originally provided a mathematical model of a particle in Brownian motion.

The Wiener model is extended to multidimensional case, too [14]. However, simple approximate stochastic models are preferable, if $m > 1$. These models are designed by replacing the traditional Kolmogorov consistency conditions because they require the inversion of matrices of nth order for computing the conditional expectation $m_n(x)$ and variance $d_n(x)$. The favorable exception is the Markov process, including the Wiener one. Extending the Wiener process to $m > 1$ the Markovian property disappears.

Replacing the regular consistency conditions by:

- continuity of the risk function $R(x)$;
- convergence of x_n to the global minimum;
- simplicity of expressions of $m_n(x)$ and $s_n(x)$,

the following simple expression of $R(x)$ is obtained using the results of [14]:

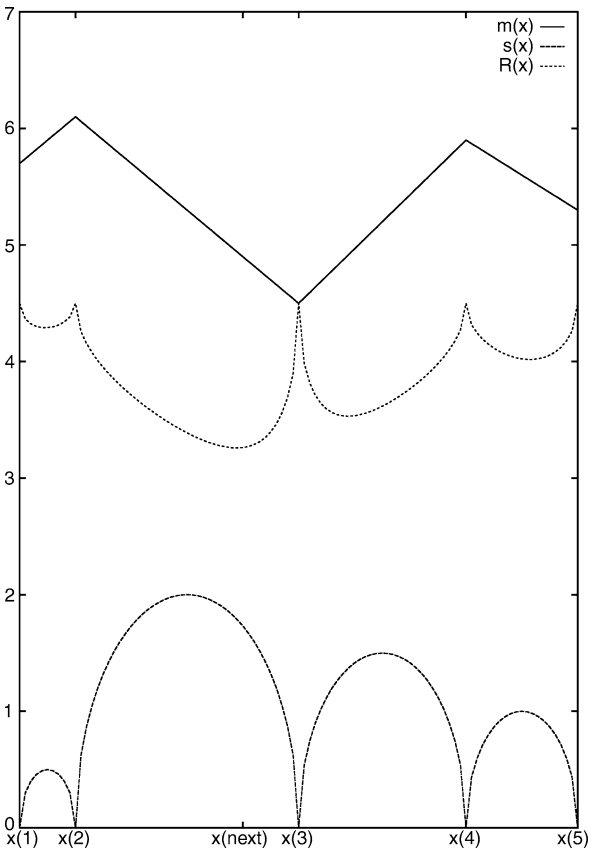
$$R(x) = \min_{1 \le i \le n} z_i - \min_{1 \le i \le n} \frac{\|x - x_i\|^2}{z_i - c_n}.$$

The aim of the DBA is to minimize the expected deviation. In addition, DBA has some good asymptotic properties, too. It is shown in [14] that

$$\frac{d^*}{d_a} = \left(\frac{f_a - f^* + \epsilon}{\epsilon}\right)^{1/2}, \quad n \to \infty,$$

where d^* is the density of x_i around the global optimum f^*, d_a and f_a are the average density of x_i and the average value of $f(x)$, and ϵ is the correction parameter in expression (1). That means that DBA provides convergence to the global minimum for any continuous $f(x)$ and greater density of observations x_i around the global optimum, if n is large. Note that the correction parameter ϵ has a similar influence as the temperature in simulated annealing. However, that is a superficial similarity. Using DBA, the good asymptotic behavior should be regarded just as an interesting 'by-product'. The reason is that Bayesian decisions are applied for the small size samples where asymptotic properties are not noticeable.

Choosing the optimal point x_{n+1} for the next iteration by DBA one solves a complicated auxiliary optimization problem minimizing the expected deviation



Bayesian Global Optimization, Figure 1
The Wiener model

$R(x)$ from the global optimum (see Fig. 1). That makes the DBA useful mainly for the computationally expensive functions of a few ($m < 20$) continuous variables. This happens in wide variety of problems such as maximization of the yield of differential amplifiers, optimization of mechanical system of shock absorber, optimization of composite laminates, estimation of parameters of immunological model and nonlinear time series, planning of extremal experiments on thermostable polymeric composition [14].

Using DBA the expert knowledge is included by defining the prior distribution. In BHA the expert knowledge is involved by defining the heuristics and optimizing their parameters using DBA.

If the number of variables is large and the objective function is not expensive, the Bayesian heuristic approach is preferable. That is the case in many discrete optimization problems. As usual, these problems

are solved using heuristics based on an expert opinion. Heuristics often involve randomization procedures depending on some empirically defined parameters. The examples of such parameters are the initial temperature, if the *simulated annealing* is applied, or the probabilities of different randomization algorithms, if their mixture is used. In these problems, the DBA is a convenient tool for optimization of the continuous parameters of various heuristic techniques. That is the Bayesian heuristic approach [15].

The example of *knapsack problem* illustrates the basic principles of BHA in *discrete optimization*. Given a set of objects $j = 1, \ldots, n$ with values c_j and weights g_j, find the most valuable collection of limited weight:

$$\begin{cases} \max\limits_{y} \quad v(y) = v(y) = \sum\limits_{j=1}^{n} c_j y_j \\ \text{s.t.} \quad \sum\limits_{j=1}^{n} g_j y_j \le g. \end{cases}$$

Here the objective function $v(y)$ depends on n Boolean variables $y = (y_1, \ldots, y_n)$, where $y_j = 1$ if object j is in the collection, and $y_j = 0$ otherwise. The well-known greedy heuristics $h_j = c_j/g_j$ is the specific value of object j. The greedy heuristic algorithm: 'take the greatest feasible h_j', is very fast but it may get stuck in some nonoptimal decision.

A way to force the heuristic algorithm out of such nonoptimal decisions is to make decision j with probability $r_j = \rho x(h_j)$, where $\rho_x(h_j)$ is an increasing function of h_j and $x = (x_1, \ldots x_m)$ is a parameter vector. The DBA is used to optimize the parameters x by minimizing the best result $f_K(x)$ obtained applying K times the randomized heuristic algorithm $\rho_x(h_j)$. That is the most expensive operation of BHA. Therefore, the parallel computation of $f_K(x)$ should be used when possible reducing the computing time in proportion to a number of parallel processors.

Optimization of x adapts the heuristic algorithm $\rho_x(h_j)$ to a given problem. Let us illustrate the parameterization of $\rho_x(h_j)$ using three randomization functions: $r_i^l = h_i^l \,/\, \sum_j h_j^l$, $l = 0, 1, \infty$. Here, the upper index $l = 0$ denotes the uniformly distributed component and $l = 1$ defines the linear component of randomization. The index ∞ denotes the pure heuristics with no randomization where $r_i^\infty = 1$ if $h_i = \max_j h_j$ and $r_i^\infty = 0$, otherwise. Here, parameter $x = (x_0, x_1, x_\infty)$ defines the probabilities of using randomizations $l = 0, 1, \infty$ correspondingly. The optimal x may be applied in different but related problems, too [15]. That is very important in the 'on-line' optimization adapting the BHA algorithms to some unpredicted changes.

Another simple example of BHA application is by trying different permutations of some feasible solution y^0. Then heuristics are defined as the difference $h_i = v(y^i) - v(y^0)$ between the permuted solution y^i and the original one y^0. The well-known simulated annealing algorithm illustrates the parameterization of $\rho_x(h_j)$ related to a single parameter x. Here the probability of accepting a worse solution is equal to $e^{-h_i/x}$, where x is the 'annealing temperature'.

The comparison of BHA with exact branch and bound algorithms solving a set of the flow-show problems is shown by the Table from [15]:

$R = 100, K = 1, J = 10, S = 10, O = 10$

Technique	f_B	d_B	x_0	x_1	x_∞
BHA	6.18	0.13	0.28	0.45	0.26
CPLEX	12.23	0.00	–	–	–

Here S is the number of tools, J is the number of jobs, O is the number of operations, f_B, x_0, x_1, x_∞ are the mean results, d_B is the variance, and 'CPLEX' denotes the standard MILP technique truncated after 5000 iterations. The table shows that in the randomly generated *flow-shop problems* the average make-span obtained by BHA was almost twice less that obtained by the exact branch and bound procedure truncated at the same time as BHA. The important conclusion is that stopping the exact methods before they reach the exact solution is not a good way to obtain the approximate solution.

The BHA has been used to solve the *batch scheduling* [15] and the clustering (parameter grouping) problems. In the clustering problem the only parameter x was the *initial annealing temperature* [8].

The main objective of BHA is to improve any given heuristic by defining the best parameters and/or the best 'mixtures' of different heuristics. Heuristic decision rules mixed and adapted by BHA often outperform (in terms of speed) even the best individual heuristics as judged by the considered examples. In addition, BHA provides almost sure convergence. However, the final

results of BHA depend on the quality of the specific heuristics including the expert knowledge. That means the BHA should be regarded as a tool for *enhancing the heuristics* but not for replacing them.

Many well-known optimization algorithms such as genetic algorithms (GA) [10], *GRASP* [13], and *tabu search* (TS) [14], may be regarded as generalized heuristics that can be improved using BHA. There are many heuristics tailored to fit specific problems. For example, the Gupta heuristic was the best one while applying BHA to the flow-shop problem [15].

Genetic algorithms [10] is an important 'source' of interesting and useful stochastic search heuristics. It is well known [2] that the results of the genetic algorithms depend on the mutation and cross-over parameters. The Bayesian heuristic approach could be used in optimizing those parameters.

In the GRASP system [13] the heuristic is repeated many times. During each iteration a greedy randomized solution is constructed and the neighborhood around that solution is searched for the local optimum. The 'greedy' component constructs a solution, one element at a time until a solution is constructed. A possible application of the BHA in GRASP is in optimizing a random selection of a candidate to be in the solution because different random selection rules could be used and their best parameters should be defined. BHA might be useful as a local component, too, by randomizing the local decisions and optimizing the corresponding parameters.

In tabu search the issues of identifying best combinations of short and long term memory and best balances of intensification and diversification strategies may be obtained using BHA.

Hence the Bayesian heuristics approach may be considered when applying almost any stochastic or heuristic algorithm of discrete optimization. The proven convergence of a discrete search method (see, for example, [1]) is an asset. Otherwise, the convergence conditions are provided by tuning the BHA [15], if needed.

The third way to apply the Bayesian approach is the *Bayesian stopping rules* (BSR) [3]. The first way, the DBA, considers a stochastic model of the whole function to be optimized. In the BSR the stochastic models regard only the features of the objective function which are relevant for the stopping of the multistart algorithm. In [20] a statistical estimate of the structure of multimodal problems is investigated. The results are applied developing BSR for the multistart global optimization methods [4,5,18].

Besides these three ways, there are other ways to apply the Bayesian approach in global optimization. For example, the Bayes theorem was used to derive the posterior distribution of the values of parameters in the simulated annealing algorithm to make an optimal choice in the trade-off between small steps in the control parameter and short Markov chains and large steps and long Markov chains [12].

In the information approach [17] a prior distribution is considered on the location parameter α of the global optimum of an one-dimensional objective function. Then an estimate of α is obtained maximizing the likelihood function after a number of evaluations of the objective function. This estimate is assumed as the next search point. For the solution of multidimensional problems, it is proposed to transform the problem into a one-dimensional problem by means of *Peano maps*.

See also

- ▶ Adaptive Simulated Annealing and its Application to Protein Folding
- ▶ Genetic Algorithms for Protein Structure Prediction
- ▶ Global Optimization Based on Statistical Models
- ▶ Monte-Carlo Simulated Annealing in Protein Folding
- ▶ Packet Annealing
- ▶ Random Search Methods
- ▶ Simulated Annealing
- ▶ Simulated Annealing Methods in Protein Folding
- ▶ Stochastic Global Optimization: Stopping Rules
- ▶ Stochastic Global Optimization: Two-Phase Methods

References

1. Andradottir S (1996) A global serach method for discrete stochastic optimization. SIAM J Optim 6:513–530
2. Androulakis IP, Venkatasubramanian V (1991) A genetic algorithmic framework for process design and optimization. Comput Chem Eng 15:217–228
3. Betro B (1991) Bayesian methods of global optimization. J Global Optim 1:1–14

4. Betro B, Schoen F (1987) Sequential stopping rules for the multistart algorithm in global optimization. Math Program 38:271–286
5. Boender G, Rinnoy-Kan A (1987) Bayesian stopping rules for multi-start global optimization methods. Math Program 37:59–80
6. DeGroot M (1970) Optimal statistical decisions. McGraw-Hill, New York
7. Diaconis P (1988) Bayesian numerical analysis. In: Statistical Decision Theory and Related Topics. Springer, Berlin, pp 163–175
8. Dzemyda G, Senkiene E (1990) Simulated annealing for parameter grouping. Trans Inform Th, Statistical Decision Th, Random Processes, 373–383
9. Glover F (1994) Tabu search: improved solution, alternatives. In: Mathematical Programming. State of the Art. Univ. Michigan, Ann Arbor, MI, pp 64–92
10. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA
11. Kushner HJ (1964) A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. J Basic Eng 86:97–100
12. van Laarhoven PJM, Boender CGE, Aarts EHL, Rinnooy-Kan AHG (1989) A Bayesian approach to simulated annealing. Probab Eng Inform Sci 3:453–475
13. Mavridou T, Pardalos PM, Pitsoulis LS, Resende MGC (1997) A GRASP for the biquadratic assignment problem. Europ J Oper Res
14. Mockus J (1989) Bayesian approach to global optimization. Kluwer, Dordrecht
15. Mockus J, Eddy W, Mockus A, Mockus L, Reklaitis G (1997) Bayesian heuristic approach to discrete and global optimization. Kluwer, Dordrecht
16. Saltenis VR (1971) On a metod of multiextremal optimization. Automatics and Computers (Avtomatika i Vychislitelnayya Tekchnika) 3:33–38. (In Russian)
17. Strongin RG (1978) Numerical methods in multi-extremal problems. Nauka, Moscow
18. Timmer GT (1984) Global optimization: A stochastic approach. PhD Thesis, Erasmus Univ. Rotterdam, The Netherlands
19. Törn A, Žilinskas A (1989) Global optimization. Springer, Berlin
20. Zielinski R (1981) A statistical estimate of the sructure of multiextremal problems. Math Program 21:348–356

Bayesian Networks

Alla R. Kammerdiner
Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA

Article Outline

Keywords
Synonyms
Introduction
Definitions
The Chain Rule for Bayesian Network
Cases/Models
Methods
Applications
See also
References

Keywords

Graphical models; Joint probability distribution; Bayesian statistics; Data mining; Optimization

Synonyms

Bayes nets

Introduction

After the initial introduction in 1982, Bayesian networks (BN) have quickly developed into a dynamic area of research. This is largely due to the special structure of Bayesian networks that allows them to be very efficient in modeling domains with inherent uncertainty. In addition, there is a strong connection between Bayesian networks and other adjacent areas of research, including data mining and optimization.

Bayesian networks have their lineage in statistics, and were first formally introduced in the field of artificial intelligence and expert systems by Pearl [17] in 1982 and Spiegelhalter and Knill-Jones [21] in 1984. The first real-life applications of Bayesian networks were Munin [1] in 1989 and Pathfinder [7] in 1992. Since the 1990s, the amount of research in Bayesian networks has increased dramatically, resulting in many modern applications of Bayesian networks to various problems of data mining, pattern recognition, image processing and data fusion, engineering, etc.

Bayesian networks comprise a class of interesting special cases, many of which were in consideration long before the first introduction of Bayesian networks. Among such interesting cases are some frequently used types of the model simplifying assumptions including naïve Bayes, the noisy-OR and noisy-AND mod-

els, as well as different models with specialized structure, in particular the time-stamped models, the strictly repetitive models, dynamic Bayesian networks, hidden Markov models, Kalman filter, Markov chains. Artificial neural networks are another subclass of Bayesian networks, which has many applications, in particular in biology and computer science.

Definitions

Based on classical probability calculus, the idea of a Bayesian network has its early origins in Bayesian statistics. On the other hand, it has an added benefit of incorporating the notions of graph theory and networks that allows us to visualize the relationships between the variables represented by the nodes of a Bayesian network. In other words, a Bayesian network is a graphical model providing a compact representation for communicating causal relationships in a knowledge domain. Below we introduce two alternative definitions of the general notion of a Bayesian network, based on the usual concepts of probability and graph theory (e. g. joint probability distribution, conditional probability distribution; nodes and edges of a graph, a parent of a node, a child of a node, etc.).

Roughly speaking, a Bayesian network can be viewed as an application of Bayesian calculus on a causal network. More precisely, one can describe a Bayesian network as a mathematical model for representing the joint distribution of some set of random variables as a graph with the edges characterized by the conditional distributions for each variable given its parents in the graph.

Given a finite collection of random variables $X = \{X_1, X_2, \ldots, X_n\}$, the formal definition of a Bayesian network can be stated as follows:

Definition 1 A *Bayesian network* is an ordered pair (G,D), where

- The first component G represents a directed acyclic graph with nodes, which correspond to the random variables $X_1, X_2, \ldots, X_n$, and directed arcs, which symbolize conditional dependencies between the variables. The set of all the arcs of G satisfies the following assumption: Each random variable in the graph is conditionally independent of its non-descendants in G, given its parents in G.
- The second component D corresponds to the set of parameters that, for each variable $X_i, 1 \leq i \leq n$, define its conditional distribution given its parents in the graph G.

Note that the variables in a Bayesian networks can follow discrete or continuous distributions. Clearly, for continuously distributed variables, there is a correspondent conditional probability density function $f(x_i|Pa(x_i))$ of X_i given its parents $Pa(X_i)$. (From now on we denote by x_i the realization of the correspondent random variable X_i.)

In many real-life applications modeled by Bayesian networks the set of states for each variable (node) in the network is finite. In the special case when all variables have finite sets of mutually exclusive states and follow the discrete distributions, the previous definition of a Bayesian network can be reformulated in the following fashion:

Definition 2 A *Bayesian network* is a structure that consists of the following elements:

- A collection of variables with a finite set of mutually exclusive states;
- A set of directed arcs between the variables symbolizing conditional independence of variables;
- A directed acyclic graph formed by the variables and the arcs between them;
- A potential table $\Pr(X_i|Pa(X_i))$ associated with each variable X_i having a set of parent variables denoted by $Pa(X_i)$.

Observe that we do not require causality in Bayesian networks, i. e. the arcs of a graph do not have to symbolize causal relationship between the variables. However, it is imperative that the so-called d-separation rules implied by the structure are satisfied [12,19]. If variables X and Y are *d-separated* in a Bayesian network under the presence of evidence e, then $\Pr(X|Y, e) = \Pr(X|e)$, i. e. the variables are conditionally independent given the evidence.

Furthermore, the d-separation rules are applied to prove one of the key laws used in Bayesian networks, a so-called *chain rule for Bayesian networks*.

The joint probability table $\Pr(X) = \Pr(X_1, X_2, \ldots, X_n)$ sufficiently describes the belief structure on the set $X = \{X_1, X_2, \ldots, X_n\}$ of variables in the model. In particular, for each variable X_i, using the joint probability table, one can easily calculate the prior

probabilities $\Pr(X_i)$ as well as the conditional probability $\Pr(X_i|e)$ given an evidence e. Nevertheless, with increase in the number of variables, the joint probability table quickly becomes unmanageably large, since the table size grows exponentially fast with the size n of the variable set. Thus, it is necessary to find another representation, which adequately and more efficiently describes the belief structure in the model. A Bayesian network over $X = \{X_1, X_2, \ldots, X_n\}$ provides such a representation. In fact, a graph in a Bayesian network gives a compact representation of conditional dependencies in the network, which allows one to compute the joint probability table from the conditional probabilities specified by the network using the chain rule below.

The Chain Rule for Bayesian Networks [8]

The joint probability distribution $\Pr(X) = \Pr(X_1, X_2, \ldots, X_n)$ of the variables $X = \{X_1, X_2, \ldots, X_n\}$ in a Bayesian network is given by the formula

$$\Pr(X) = \prod_{i=1}^{n} \Pr(X_i|Pa(X_i)) \,, \tag{1}$$

where $Pa(X_i)$ denotes the set of all parents of variable X_i.

The chain rule for Bayesian networks also provides an efficient way for probability updating when the new information is received about the model. There is a variety of different types of such new information, i. e. evidence. Two most common types of evidence are *finding* and *likelihood evidence*. Finding is evidence that specifies which states are possible for some variables, while likelihood evidence gives a proportion between the probabilities of two given states. Note that some types of evidence including likelihood evidence cannot be given in the form of findings.

Cases/Models

Bayesian networks provide a general framework for a number of specialized models, many of which were identified long before the concept of a Bayesian network was proposed. Such special cases of BN vary in their graph structures as well as the probability distribution.

The probability distributions for a Bayesian network can be defined in several ways. In some situations, it is possible to use theoretically well-defined distributions. In others, the probabilities can be estimated from data as frequencies. In addition, absolutely subjective probability estimates are often used for practical purposes. For instance, when the number of conditional probability distributions to acquire from the data is very large, some simplifying assumptions may be appropriate.

The simplest Bayesian network model is the well-known *naïve Bayes* (or *simple Bayes*) *model* [4], which can be summarized as follows:

- The graph structure of the model consists of one hypothesis variable H, and a finite set of information variables $I = \{I_1, I_2, \ldots, I_n\}$ with the arcs from H to every $I_k, 1 \leq k \leq n$. In other words, the variables form a diverging connection, where the hypothesis variable H is a common parent of variables $I_1, I_2, \ldots, I_n$;
- The probability distributions are given by the values $\Pr(I_k|H)$, for every information variable $I_k, 1 \leq k \leq n$.

The probability updating procedure based on the naïve Bayes model works in the following manner: Given a collection of observations $e_1, e_2, \ldots, e_n$ on the variables $I_1, I_2, \ldots, I_n$ respectively, the *likelihood* of H given $e_1, e_2, \ldots, e_n$ is computed:

$$L(H|e_1, e_2, \ldots, e_n) = \prod_{i=1}^{n} \Pr(e_i|H) \,. \tag{2}$$

Then the posterior probability of H is obtained from the formula:

$$\Pr(H|e_1, e_2, \ldots, e_n) = C \cdot \Pr(H) \cdot L(H|e_1, e_2, \ldots, e_n) \,, \tag{3}$$

where C is a normalization constant.

Another special case of BNs is a model underlined by the simplifying assumption called *noisy-OR* [18]. This model can be constructed as follows:

Let $A_1, A_2, \ldots, A_n$ represent some binary variables listing all parents of a binary variable B. Each event $A_i = x, x \in \{0, 1\}$, causes $B = x$ except when an *inhibitor* prevents it, with the probability p_i, i. e. $\Pr(B = 1 - x|A_i = x) = p_i$. Suppose that all inhibitors are independent.

Then the graph of a corresponding Bayesian network is represented by the converging connection with B as the child node of $A_1, A_2, \ldots, A_n$, while the conditional probabilities are given by $\Pr(B = x|A_i = x) = 1 - p_i$. Since the conditional distributions are independent of each other, then

$$\Pr(B = 1 - x|A_1, A_2, \ldots, A_n) = \prod_{i=1}^{n} p_i \,. \tag{4}$$

The noisy-OR assumption gives a significant advantage for efficient probability updating, since the number of distributions increases linearly with respect to the number of parents.

The construction complementary to noisy-OR is called *noisy-AND*. In the noisy-AND model, the graph is the convergent connection just as in the noisy-OR model, all the causes are required to be on in order to have an effect, and all the causes have mutually independent random inhibitors. Both noisy-OR and noisy-AND are special cases of a general method called *noisy functional dependence*.

Many modeling approaches have been developed which employ introduction of mediating variables in a Bayesian network. One of these methods, called *divorcing*, is the process separating parents $A_1, A_2, \ldots, A_i$ and $A_{i+1}, \ldots, A_n$ of a node B by introducing a mediating variable C as a child of divorced parent nodes $A_1, A_2, \ldots, A_i$ and a parent of the initial child node B. The divorcing of $A_1, A_2, \ldots, A_i$ is possible if the following condition is satisfied:

The set Γ of all configurations of $A_1, A_2, \ldots, A_i$ can be partitioned into the sets $c_1, c_2, \ldots, c_s$ so that for every $1 \leq j \leq m$, any two configurations $\gamma_1, \gamma_2 \in c_j$ have the same conditional probabilities:

$$\Pr(B|\gamma_1, A_{i+1}, \ldots, A_n) = \Pr(B|\gamma_2, A_{i+1}, \ldots, A_n) \,. \tag{5}$$

Other modeling methods, which engage the mediating variables, involve modeling undirected relations, and situations with expert disagreement. Various types of undirected dependencies, including logical constraints, are represented by adding an artificial child C of the constrained nodes $A_1, A_2, \ldots, A_n$ so that the conditional probability $\Pr(C|A_1, A_2, \ldots, A_n)$ emulates the relation. The situation, where k experts disagree on the conditional probabilities for different variables $B_1, B_2, \ldots, B_n$ in the model can be modeled by introducing a mediating node M with k states $m_1, m_2, \ldots, m_k$ so that the variables $B_1, B_2, \ldots, B_n$ on whose probabilities the experts disagree become the only children of expert node M. Another approach to modeling expert disagreements is by introducing alternative models with weights assigned to each model.

An important type of Bayesian networks are so-called *time-stamped models* [10]. These models reflect the structure which changes over time. By introducing a discrete time stamp in such structures, the time-stamped models are partitioned into submodels for every unit of time. Each local submodel is called a *time slice*. The complete time-stamped model consists of all its time slices connected to each other by *temporal links*.

A *strictly repetitive* model is a special case of a time-stamped model such that all its time slices have the same structure and all the temporal links are alike. The well-studied *hidden Markov models* is a special class of strictly repetitive time-stamped models for which the Markov property holds, i. e. given the present, the past is independent of the future.

A hidden Markov model with only one variable in each time slice connected to the variables outside the time slice is a *Kalman filter*. Furthermore, a *Markov chain* can be represented as a Kalman filter with only one variable in every time slice. It is possible to convert a hidden Markov model into a Markov chain by cross-multiplying all variables in each time slice.

The time-stamped models can have either *finite horizon* or *infinite horizon*. An infinite Markov chain would be an example of a time-stamped model with an infinite horizon. Furthermore, the repetitive time-stamped models with infinite horizon are also known as *dynamical Bayesian networks*. By utilizing the special structure of many repetitive temporal models, they can be compactly represented [2]. Such special representation can often facilitate the design of efficient algorithms in updating procedures.

Artificial neural networks can also be viewed as a special case of Bayesian networks, where the nodes are partitioned into n mutually exclusive *layers*, and the set of arcs represented by the links from the nodes on layer i to the nodes on $i + 1, 1 \leq i \leq n$. Layer 1 is usually called the *input layer*, while layer n is known as the output layer.

Methods

Just as the BNs have their roots in statistics, the approaches for discovering a BN structure utilize statistical methods. That is why a database of cases is instrumental for discovery of the graph configuration of a Bayesian network as well as probability updating. There are three basic types of approaches to extracting BNs from data: *batch learning*, *adaptation*, and *tuning*.

Batch Learning. *Batch Learning* is a process of extracting the information from a database of collected cases in order to establish a graph structure and the probability distributions for a certain Bayesian networks.

Often there are many ways to model a Bayesian network. For example, we may obtain two different probability distributions to model the true distribution of the variable in the network. To make an intelligent choice between two available distributions, it is important to have an appropriate measure of their accuracy. A logical way to approach this subject is by assigning penalties for a wrong forecast on the base of a specified distribution. For example, two widely accepted ways for assigning penalties are the *quadratic (Brier) scoring rule* and the *logarithmic scoring rule*.

Given the true distribution $p = (p_1, p_2, \ldots, p_m)$ of a discrete random variable with m states, and some approximate distribution $q = (q_1, q_2, \ldots, q_m)$, the quadratic scoring rule assigns the expected penalty as:

$$ES_Q(p,q) = \sum_{i=1}^{m} p_i \left((1-q_i)^2 + \sum_{j \neq i} q_j^2 \right) . \tag{6}$$

The distance between true distribution p and approximation q is given by the formula:

$$d_Q(p,q) = ES_Q(p,q) - ES_Q(p,p) . \tag{7}$$

Hence, from (6) we have:

$$d_Q(p,q) = \sum_{i=1}^{m} (p_i - q_i)^2 . \tag{8}$$

The distance $d_Q(p,q)$ given in (8) is called the *Euclidean distance*.

The logarithmic scoring rule assigns to each outcome i the corresponding penalty $S_L(q,i) = -\log q_i$. Hence, the expected penalty is calculated as:

$$ES_L(p,q) = -\sum_{i=1}^{m} p_i \log q_i . \tag{9}$$

From (7), we obtain an expression for the distance between the true distribution p, and the approximation q:

$$d_L(p,q) = \sum_{i=1}^{m} p_i \log \frac{p_i}{q_i} , \tag{10}$$

which is called the *Kulbach–Leibler divergence*.

Note that both definitions, the Euclidean distance and the Kulbach–Leibler divergence, can be easily extended in the case of continuous random variables. Moreover, both scoring rules, the quadratic and the logarithmic, possess the following useful property: only the true distribution minimizes the score. The scoring rules that exhibit this property are called *strictly proper*. Since the quadratic and the logarithmic scoring rules are strictly proper, then the corresponding distance measures d_Q and d_L both satisfy the following:

$$d(p,q) = 0 \quad \text{if and only if} \quad p = q .$$

Different scoring rules and corresponding distance measures for discrete and continuous random variables have been extensively studied in statistics. A comprehensive review of strictly proper scoring rules is given in [6].

Naturally, among several different Bayesian networks that model the situation equally closely, the one of the smallest "size" would be preferred.

Let M denote a Bayesian network over the variable set $X = \{X_1, X_2, \ldots, X_n\}$. Then the size of M is given by

$$\text{Size}(M) = \sum_{i=1}^{n} s(X_i) , \tag{11}$$

where $s(X_i)$ denotes the number of entries in the conditional probability table $\Pr(X_i | Pa(X_i))$, and $Pa(X_i)$ is the set of parents of X_i.

The following measure accounts for both the size of the model and its accuracy.

Given a Bayesian network M over X with the true probability distribution p, and an approximate Bayesian network model N with distribution q, we define the *acceptance measure* as

$$\alpha(p,N) = \text{Size}(N) + C \cdot d(p,q) , \tag{12}$$

where Size(·) is the network size defined by (11), $d(p, q)$ is a distance measure between probability distributions p and q, and C is a positive real constant.

The general approach to batch learning a Bayesian network from the data set of cases can be summarized as follows:

- Select an appropriate threshold τ for distance measure $d(p, q)$ between two distributions;
- Fix a suitable constant C in a definition of acceptance measure $\alpha(p, N)$;
- Among all Bayesian network models over X and distribution q such that $d(p, q) < \tau$, select the model that minimizes $\alpha(p, N)$.

Although simple, this approach has many practical issues. The data sets in batch learning are usually very large, the model space grows exponentially in the number of variables, there may be missing data in the data set, etc. To extract structure from such data, one often has to employ special heuristics for searching the model space. For instance, causality can be used to cluster the variables according to a causal hierarchy. In other words, we partition the variable set X into subsets $S_1, S_2, \ldots, S_k$, so that the arcs satisfy a partial order relation. If we find the model N having the distance $d(p, q) < \tau$, the search stops; otherwise we consider the submodel of N.

Adaptation It is often desirable to build a system capable of automatically adapting to different settings. *Adaptation* is a process of adjusting a Bayesian network model so that it is better able to accommodate to new accumulated cases.

When building a Bayesian network, usually there is an uncertainty whether the chosen conditional probabilities are correct. This is called the second-order uncertainty.

Suppose that we are not sure which table out of m different conditional probability tables $T_1, T_2, \ldots, T_m$ represents the true distribution for $\Pr(X_i|Pa(X_i))$ for some variable X_i in a network. By introducing a so-called *type variable* T with states $t_1, t_2, \ldots, t_m$ into the graph so that T is a parent of X_i, we can model this uncertainty into the network. Then the prior probability $\Pr(t_1, t_2, \ldots, t_m)$ represents our belief about the correctness of the tables $T_1, T_2, \ldots, T_m$ respectively. Next, we set $\Pr(X_i|Pa(X_i), t_j) = T_j$. Our belief about the correctness of the tables is updated each time we receive new evidence e. In other words, for the next case, we use $\Pr(t_1, t_2, \ldots, t_m|e)$ as the new prior probability of tables' accuracy.

Sometimes the second-order uncertainty about the conditional probabilities cannot be modeled by introducing type variables. In such cases, various statistical methods can be applied. Normally such methods exploit various properties of parameters, such as global independence, local independence, etc.

The property of *global independence* states that the second-order uncertainty for the variables is independent, i. e. the probability tables for the variables can be adjusted independently from each other.

The *local independence* property holds if and only if for any two different parent configurations π_1, π_2, the second-order uncertainty on $\Pr(A|\pi_1)$ is independent of the second-order uncertainty on $\Pr(A|\pi_2)$, and the two distributions can be updated independently from each other. In other words, local independence means the independence of the uncertainties of the distributions for different configurations of parents.

The *fractional updating* scheme [22], is an algorithm for reducing the second-order uncertainty about the distributions based on the received evidence. Suppose that the properties of global and local independence for the second-degree uncertainty hold simultaneously. For every configuration π of parents of variable X_i, the certainty about $\Pr(X_i|\pi)$ is given through an artificially selected sample size parameter n_i, and for any state x_i^j of variable X_i we have a corresponding *count* $n_i^j = n_i \cdot \Pr(x_i^j|\pi)$. After receiving an evidence e, we compute probabilities $\Pr(x_i^j, \pi|e)$. Then the updated count n_i^j is the sum of $\Pr(x_i^j, \pi|e)$ and the old n_i^j. Since $n_i = \sum_j n_i^j$, the old sample size parameter n_i becomes $n_i + \Pr(\pi|e)$.

Although efficient in reducing the uncertainty about the distributions, this scheme has some serious drawbacks. In fact, it tends to reduce the second-degree uncertainty too fast, by overestimating the counts. In order to avoid this, one can introduce a so-called *fading factor* f. Then after receiving an evidence, the sample size n_i is changed to $f \cdot n_i + \Pr(\pi|e)$, and the counts n_i^j are updated to $f \cdot n_i^j + \Pr(x_i^j, \pi|e)$. Therefore, the fading factor f insures that the influence of the past decreases exponentially [16].

After describing some approaches in adapting a Bayesian network to different settings of distribution

parameters, it is equally important to discuss the uncertainty in the graph structure. In many cases, we can compensate for the variability in the graph structure of a Bayesian network just by modifying the parameters of distributions in the network. Sometimes it may not be sufficient to adjust the distribution parameters in order to account for the change in the model. In fact, the difference in the graph structure may be so significant that it becomes impossible to accurately reflect the situation by a mere parameter change.

There are two main approaches to graph structure adaptation in Bayesian networks. The first method works by collecting the cases, and re-running the batch learning procedure to update the graph structure. The second method, also known as the expert disagreement approach, works simultaneously with a set of different models, and updates the weight of each model according to the evidence. More precisely, suppose there are m alternative models $M_1, M_2, \ldots, M_m$ with corresponding initial weights $w_1, w_2, \ldots, w_m$ that express our certainty of the models. Let Y be some variable in the network. After receiving an evidence e, we obtain the probabilities $\Pr_i(Y|e) := \Pr(Y|e, M_i)$ and $\Pr_i(e) := \Pr(e|M_i)$ according to each model M_i, for $1 \le i \le m$. Then,

$$\Pr(Y|e) := \sum_{i=1}^{m} w_i \cdot \Pr{}_i(Y|e)\,, \tag{13}$$

and the updated weights w_i are computed as the probabilities of the corresponding models M_i given the past evidence: $w_i = \Pr(M_i|e)$. Hence, by the well-known Bayes formula:

$$w_i = \frac{\Pr(e|M_i)\Pr(M_i)}{\sum_j w_j \Pr_j(e)}\,. \tag{14}$$

Note that the expert disagreement approach to graph structure adaptation can be further extended to include the adaptation of distribution parameters based on the above methods, such as fractional updating.

Tuning Tuning is the process of adjusting the distribution parameters so that some prescribed requests for the model distributions are satisfied. The commonly used approach to tuning is the gradient descent on the parameters similar to training in neural networks.

Let τ represent the set of parameters which are chosen to be altered. Let $p(\tau)$ denote the current model distribution, and q be the target distribution. Suppose $d(p, q)$ represents the distance between two distributions. The following gradient descent tuning algorithm is given in [9]:

- Compute the gradient of $d(p, q)$ with respect to the parameters τ;
- Select a step size $\alpha > 0$, and let $\Delta\tau = -\alpha \cdot \overrightarrow{\nabla} d(p, q)(\tau_0)$, i. e. give τ_0 a displacement $\Delta\tau$ in the opposite direction to the gradient of $d(p, q)(\tau_0)$;
- Repeat this procedure until the gradient is sufficiently close to zero.

Evolutionary methods, *simulated annealing*, *expectation-maximization* and non-parametric methods are among other commonly used methods for tuning or training Bayesian networks.

Applications

The concept of a Bayesian network can be interpreted in different contexts. From a statistical point of view, a Bayesian network can be defined as a compact representation of the joint probability over a given set of variables. From a broader point of view, a Bayesian network is a special type of graphical model capable of reflecting causality, as well as updating its beliefs in view of received evidence. All these features make a Bayesian network a versatile instrument that can be used for various purposes, including facilitating communication between human and computer, extracting hidden information and patterns from data, simplifying decision making, etc.

Due to their special structure, Bayesian networks have found many applications in various areas such as artificial intelligence and expert systems, machine learning and data mining. Bayesian networks are used for modeling knowledge in text analysis, image processing, speech pattern analysis, data fusion, engineering, biomedicine, gene and protein regulatory networks, and even meteorology. Furthermore, it has been expressed that the inductive inference procedures based on Bayesian networks can be used to introduce inductive reasoning in such a previously strictly deductive science as mathematics.

The large scope of different applications of Bayesian networks is especially impressive when taking into ac-

count that the theory of Bayesian networks has only been around for about a quarter of a century. Next, several examples of recent real-life applications of Bayesian networks are considered to illustrate this point.

Recent research in the field of automatic speech recognition [13] indicates that dynamic Bayesian networks can effectively model hidden features in speech including articulatory and other phonological features. Both hidden Markov models (HMM), which are a special case of dynamic Bayesian networks (DBN), and more general dynamic Bayesian networks have been applied for modeling audio-visual speech recognition. In particular, a paper by A.V. Nefian et al. [15] describes an application of the *coupled HMM* and the *factorial HMM* as two suitable statistical models for audio-video integration. The factorial HMM is a generalization of HMM, where the hidden state is represented by a collection of variables also called *factors*. These factors, although independent of each other, all impact the observations, and hence become connected indirectly. The coupled HMM is a DBN represented as two regular HMM whose hidden state nodes have links to the hidden state nodes from the next time slice. The coupled HMM has also been applied to model hand gestures, the interaction between speech and hand gestures, etc. In addition, face detection and recognition problems have been studied with the help of Bayesian networks.

Note that different fields of application may call for specialized employment of Bayesian network methods, and conversely, similar approaches can be successfully used in different application areas. For instance, along with the applications to speech recognition above, coupled hidden Markov models have been employed in modeling multi-channel EEG (electroencephalogram) data.

An interesting example of the application of a Bayesian network to expert systems includes developing strategies for troubleshooting complex electromechanical systems, presented in [23]. The constructed Bayesian network has the structure of a naïve Bayes model. In the decision tree for the troubleshooting model, the utility function is given by the cost of repair. Hence, the goal is to find a strategy minimizing the expected cost of repair.

An interesting recent study [3] describes some applications of Bayesian networks in meteorology from a data mining point of view. A large database of daily observations of precipitation levels and maximum wind speed is collected. The Bayesian network structure is constructed from meteorological data by using various approaches, including batch learning procedure and simulation techniques. In addition, an important data mining application of Bayesian networks is illustrated by giving an example of missing data values estimation from the evidence received.

Applications of Bayesian Networks to Data Mining; Naïve Bayes Rapid progress in data collection techniques and data storage has enabled an accumulation of huge amounts of experimental, observational and operational data. As the result, massive data sets containing a large amount of information can be found almost everywhere. A well-known example is the data set containing the observed information about the human genome. The need to quickly and correctly analyze or manipulate such enormous data sets facilitated the development of data mining techniques.

Data mining is research aimed at discovery of various types of knowledge from large data warehouses. Data mining can also be seen as an integral part of the more general process of knowledge discovery in databases. Two other parts of this knowledge discovery are preprocessing and postprocessing. As seen above, Bayesian networks can also extract knowledge from data, which is called evidence in the Bayesian framework. In fact, the Bayesian network techniques can be applied to solve data mining problems, in particular, classification.

Many effective techniques in data mining utilize methods from other multidisciplinary research areas such as database systems, pattern recognition, machine learning, and statistics. Many of these areas have a close connection to Bayesian networks. In actuality, data mining utilizes a special case of Bayesian networks, namely, naïve Bayes, to perform effective classification. In a data mining context, classification is the task of assigning objects to their relevant categories. The incentive for performing classification of data is to attain a comprehensive understanding of differences and similarities between the objects in different classes.

In the Bayesian framework, the data mining classification problem translates into finding the class param-

eter which maximizes the posterior probability of the unknown instance. This statement is called the *maximum a posteriori principle*. As mentioned earlier, the naïve Bayes is an example of a simple Bayesian network model.

Similarly to the naïve Bayes classifier, classification by way of building suitable Bayesian networks is capable of handling the presence of noise in the data as well as the missing values. Artificial neural networks can serve as an example of the Bayesian network classifier designed for a special case.

Application to Global and Combinatorial Optimization In the late 1990s, a number of studies were conducted that described how BN methodology can be applied to solve problems of global and combinatorial optimization. The connection between graphical models (e. g. Bayesian networks) and evolutionary algorithms (applied to optimization problems) was established. In particular, P. Larrañaga et al. combined some techniques from learning BN's structure from data with an evolutionary computation procedure called the Estimation of Distribution Algorithm [11] to devise a procedure for solving combinatorial optimization problems. R. Etxerberria and P. Larrañaga proposed a similar approach for global optimization [5].

Another method based on learning and simulation of BNs that is known as the Bayesian Optimization Algorithm (BOA) was suggested by M. Pelikan et al. [20]. The method works by randomly generating an initial population of solutions and then updating the population by using selection and variation. The operation of selection makes multiple copies of better solutions and removes the worst ones. The operation of variation, at first, constructs a Bayesian network as a model of promising solutions following selection. Then new candidate solutions are obtained by sampling of the constructed Bayesian network. New solutions are incorporated into the population in place of some old candidate solutions, and the next iteration is executed unless a termination criterion is reached.

For additional information on some real-world applications of Bayesian networks to classification, reliability analysis, image processing, data fusion and bioinformatics, see the recent book edited by A. Mittal et al. [14].

See also

- ► Bayesian Global Optimization
- ► Evolutionary Algorithms in Combinatorial Optimization
- ► Neural Networks for Combinatorial Optimization

References

1. Andreassen S (1992) Knowledge representation by extended linear models. In: Keravnou E (ed) Deep Models for Medical Knowledge Engineering. Elsevier, pp 129–145
2. Bangsø O, Wuillemin PH (2000) Top-down Construction and Repetitive Structures Representation in Bayesian Networks, Proceedings of the Thirteenth International FLAIRS Conference. AIII Press, Cambridge, MA
3. Cano R, Sordo C, Gutierrez JM (2004) Applications of Bayesian Networks in Meteorology, Advances in Bayesian Networks. In: Gamez et al (eds) Springer, pp 309–327
4. de Dombal F, Leaper D, Staniland J, McCan A, Harrocks J (1972) Computer-aided diagnostics of acute abdominal pain. Brit Med J 2:9–13
5. Etxerberria R, Larrañaga P (1999) Global optimization with Bayesian networks, II Symposium on Artificial Intelligence, CIMAF-99. Special Session on Distribution and Evolutionary Optimization. ICIMAF, La Habana, Cuba, pp 332–339
6. Gneiting T, Raftery AE (2005) Strictly proper scoring rules, prediction, and estimation, Technical Report no. 463R. Department of Statistics, University of Washington
7. Heckerman D, Horvitz E, Nathwani B (1992) Towards normative expert systems: Part I, the Pathfinder project. Method Inf Med 31:90–105
8. Jensen FV (1996) An Introduction to Bayesian Networks. UCL Press, London
9. Jensen FV (1999) Gradient descent training of Bayesian networks, Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU). Springer, Berlin, pp 190–200
10. Kjærulff U (1995) HUGS: Combining exact inference and Gibbs sampling in junction trees, Proceedings of the Eleventh Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco, CA, pp 368–375
11. Larrañaga P, Etxeberria R, Lozano JA, Peña JM (1999) Optimization by learning and simulation of Bayesian and Gaussian networks, Technical Report EHU-KZAA-IK-4/99. Department of Computer Science and Artificial Intelligence, University of the Basque Country
12. Lauritzen SL (1996) Graphical Models. Oxford University Press, Oxford
13. Livescu K, Glass J, Bilmes J (2003) Hidden feature modeling for speech recognition using dynamic Bayesian networks. Proc. EUROSPEECH, Geneva Switzerland, August–September

14. Mittal A, Kassim A, Tan T (2007) Bayesian Network Technologies: Applications and Graphical Models, Interface Graphics, Inc., Minneapolis, USA
15. Nefian AV, Liang L, Pi X, Liu X, Murphy K (2002) Dynamic Bayesian Networks for Audio-visual Speech Recognition. J Appl Signal Proc 11:1–15
16. Olesen KG, Lauritzen SL, Jensen FV (1992) aHUGIN: A system creating adaptive causal probabilistic networks, Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Francisco, pp 223–229
17. Pearl J (1982) Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach, National Conference on Artificial Intelligence. AAAI Press, Menlo Park, CA, pp 133–136
18. Pearl J (1986) Fusion, propagation, and structuring in belief networks. Artif Intell 29(3):241–288
19. Pearl J (1988) Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference Series in Representation and Reasoning. Morgan Kaufmann, San Francisco
20. Pelikan M, Goldberg DE, Cantú-Paz E (1999) BOA: The Bayesian Optimization Algorithm, Proceedings of the Genetic and Evolutionary Computation conference GECCO-99, vol 1. Morgan Kaufmann, San Francisco
21. Spiegelhalter DJ, Knill-Jones RP (1984) Statistical and knowledge-based approaches to clinical decision-support systems. J Royal Stat Soc A147:35–77
22. Spiegelhalter D, Lauritzen SL (1990) Sequential updating of conditional probabilities on directed graphical structures. Networks 20:579–605
23. Vomlel J (2003) Two applications of Bayesian networks, Proceedings of conference Znalosti. Ostrava, Czech Republic, pp 73–82

Beam Selection in Radiotherapy Treatment Design

Allen Holder
Department of Mathematics and the University of Texas Health Science Center at San Antonio, Department of Radiological Sciences, Trinity University, San Antonio, USA

Article Outline

Synonyms
Introduction
Definitions
Formulation
Models
Conclusions
See also
References

Synonyms

Beam orientation optimization; Beam angle optimization

Introduction

Cancer is typically treated with 3 standard procedures: 1) surgery – the intent of which is to physically rescind the disease, 2) chemotherapy – drug treatment that attacks fast proliferating cells, and 3) radiotherapy – the targeted treatment of cancer with ionizing beams of radiation. About half of all cancer patients receive radiotherapy, which is delivered by focusing high-energy beams of radiation on a patient's tumor(s). Treatment design is traditionally considered in three phases:

Beam Selection The process of deciding the number and trajectory of the beams that will pass through the patient.

Fluence Optimization Calculating the amount of dose to deliver along each of the selected beams so that the patient is treated as well as possible.

Delivery Optimization Deciding how to best deliver the treatment designed in the first two steps.

The fundamental question in optimizing radiotherapy treatments is how to best treat the patient, and such research requires detailed knowledge of medical physics and optimization. Unlike the numerous research pursuits within the field of optimization that require a specific expertise, the goals of this research rely on an overriding understanding of modeling, solving and analyzing optimization problems as well as an understanding of medical physics. The necessary spectrum of knowledge is commonly collected into a research group that is comprised of medical physicists, operations researchers, computer scientists, industrial engineers, and mathematicians.

In a modern clinic, the first phase of selecting beams is accomplished by a treatment planner, and hence, the quality of the resulting treatment depends on the expertise of this person. Fluence optimization is automatically conducted once beams are selected, and the resulting treatment is judged with a variety of metrics and visualization tools. If the treatment is acceptable,

the process ends. However, unacceptable treatments are common, and in this scenario the collection of beams is updated and fluence optimization is repeated with the new beams. This trial-and-error approach oscillates between the first two phases of treatment design and often continues for hours until an acceptable treatment is rendered. The third phase of delivery optimization strives to orient the treatment machinery so that the patient is treated as efficiently as possible, where efficiency is interpreted as shortest delivery time, shortest exposure time, etc.

The focus of this entry is Beam Selection, which has a substantial literature in the medical physics community and a growing one in the operations research community. As one would expect, no single phase of treatment design exists in isolation, and although the three phase approach pervades contemporary thinking, readers should be aware that future efforts to optimize the totality of treatment design are being discussed. The presentation below is viewed as part of this bigger goal.

Definitions

An understanding of the technical terms used to describe radiotherapy is needed to understand the scope of Beam Selection. Patient images such as CAT scans or MRI images are used to identify and locate the extent of the disease. Treatment design begins with the tedious task of delineating the target and surrounding tissues on each of the hundreds of images. The resulting 3D structures are individually classified as either a target, a critical structure, or normal tissue. An oncologist prescribes a goal dose for the target and upper bounds on the remaining tissues. This prescription is tailored to the optimization model used in the second phase of treatment design and is far from unique. A discussion of the myriad of models used for fluence optimization exceeds the confines of this article and is fortunately not needed.

The method of treatment depends on the clinic's technology, and we begin with the general concepts common to all modalities. A patient lies on a treatment couch that can be moved vertically and horizontally and rotated in the plane horizontal to the floor. A gantry rotates around the patient in a great circle, the head of which is used to focus the beam on the patient, see Fig. 1. Shaping and modulating the beam is important

Beam Selection in Radiotherapy Treatment Design, Figure 1
A typical treatment configuration

Beam Selection in Radiotherapy Treatment Design, Figure 2
A multileaf collimator

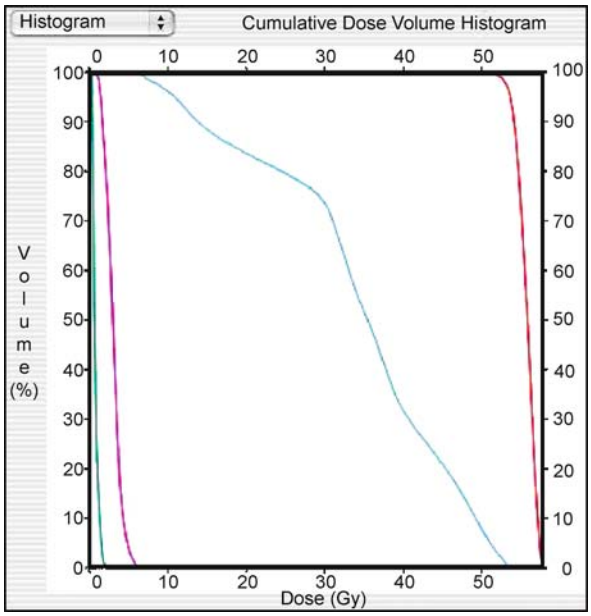
in all forms of treatment, and although these tasks are accomplished differently depending on the technology, it is common to control smaller divisions of each beam called sub-beams. As an example, the gantry's head often contains a multileaf collimator that is capable of dividing the beam (Fig. 2), a technology that is modeled by replacing the whole beam with a grid of rectangular sub-beams. Previous technology shaped and modulated the beam without a collimator, but the concept of a sub-beam remains appropriate.

The center of the gantry's rotation is called the isocenter, a point that is placed near the center of the target by repositioning the patient via couch adjust-

ments. The beam can essentially be focused on the patient from any point on a sphere with a one meter radius that encompasses the patient, although some positions are not possible due to patient-gantry interference. The beam selection problem is to choose a few of these positions so that the resulting treatment is of high quality. If the selection process is restricted to a single great circle, then the term beam is often replaced with angle (in fact these terms are used synonymously in much of the literature).

The collection of positions on the sphere from which we are allowed to select is denoted by $\mathcal{A}$. This set contains every point of the sphere in the continuum, but in practice $\mathcal{A}$ is a finite set of candidate beams. The problem of selecting beams depends on a judgment function, which is a mapping from the power set of $\mathcal{A}$, denoted $\mathcal{P}(\mathcal{A})$, into the nonnegative extended reals, denoted $\mathbb{R}_+^* = \{x \in \mathbb{R} : x \geq 0\} \cup \{\infty\}$. Assuming that low values correspond with high-quality treatments, we have that a judgment function is a mapping $f : \mathcal{P}(\mathcal{A}) \to \mathbb{R}_+^*$ with the monotonicity property that if $\mathcal{A}'$ and $\mathcal{A}''$ are subsets of $\mathcal{A}$ such that $\mathcal{A}' \supseteq \mathcal{A}''$, then $f(\mathcal{A}') \leq f(\mathcal{A}'')$. The monotonicity condition guarantees that treatment quality can not degrade if beams are added to an existing treatment.

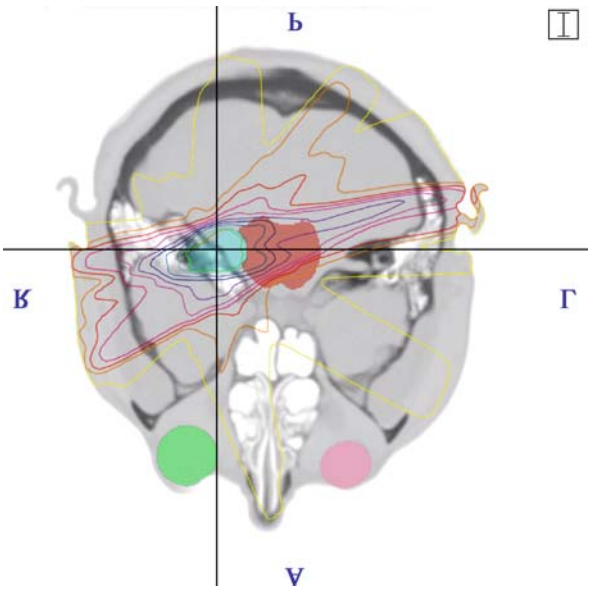
The judgment function is commonly the optimal value from the second phase of treatment design, and for any $\mathcal{A}' \in \mathcal{P}(\mathcal{A})$, we let $X(\mathcal{A}')$ be the feasible region of the optimization problem that decides fluences. An algebraic description of this set relies on the fact that we can accurately model how radiation is deposited as it passes through the anatomy. There are several competing radiobiological models that accomplish this task, each of which produces the rate coefficient $A_{(j,a,i)}$, which is the rate at which sub-beam i in beam a deposits energy into the anatomical position j. These values form a dose matrix A, with rows being indexed by j and columns by (a, i). The term used to measure a sub-beam's energy is fluence, and experimentation validates that anatomical dose, which is measured in Grays (Gy), is linear in fluence. So, if $x_{(a,i)}$ is the fluence of sub-beam i in beam a, then the linear map $x \mapsto Ax$ transforms fluence values into anatomical dose. We partition the rows of the dose matrix into those that correspond with anatomical positions in the target – forming the submatrix A_T, in a critical structure – forming the submatrix A_C, and in normal tissue – forming the submatrix A_N. With this notation, $A_T x$, $A_C x$ and $A_N x$ are the delivered doses to the target, the critical structures, and the normal tissues under treatment x.



Beam Selection in Radiotherapy Treatment Design, Figure 3
A dose-volume histogram, the horizontal axis is the anatomical dose (measured in Grays) and the vertical axis is the percent of volume

Treatment planners use visual and numerical methods to evaluate treatments. The two most common visual tools are the dose-volume histogram (DVH) and a collection of isocontours. A DVH is a plot of dose versus volume and allows a treatment planner to quickly gauge the extent to which each structure is irradiated, an example is found in Fig. 3. The curve in the upper right side of the figure corresponds to the target, which is the growth to the left of the brain stem in Fig. 4. The ideal curve for the target would be one that remains at 100% until the desired dose and then falls immediately to zero, and the ideal curves for the remaining structures would be ones that fall immediately to zero. The curve passing through the middle of Fig. 3 corresponds to the brain stem and indicates that approximately 80% of the brain stem is receiving half of the target dose.

What a DVH lacks is spatial detail about the anatomical dose, but this information is provided by the isocontours, which are level curves drawn on each of the patient images. For example, if the target's goal is



Beam Selection in Radiotherapy Treatment Design, Figure 4
A collection of isocontours on a single patient image

80 Gy, then the 90% isocontour contains the anatomical region that receives at least $0.9 \times 80 = 72$ Gy. Figure 4 illustrates the 100%, 90%, …, 10% isocontours on a single patient image. One would hope that these isocontour would tightly contain the target on each of the patient images, a goal commonly referred to as conformality. Although a DVH is often used to decide if a treatment is unacceptable, both the DVH and the isocontours are used to decide if a treatment is acceptable. Although treatments are commonly evaluated exclusively with a DVH and the isocontours, there are well established numerical scores that are also used. Such scores are called conformality indices and consider the ratios of under and over irradiated tissue, and as such, these values collapse the DVH into a numerical value. We do not discuss these measures here, but the reader should be aware that they exist.

Formulation

The N-beam selection problem for the judgment function f and candidate set of beams $\mathcal{A}$ is

$$\min\{f(\mathcal{A}') : \mathcal{A}' \in \mathcal{P}(\mathcal{A}),\ |\mathcal{A}'| = N\}\,. \tag{1}$$

The parameter N is provided by the treatment planner and is intended to control the complexity of the treatment. The prevailing thought is that fewer beams are preferred if all other treatment goals remain satisfactory, and if f adequately measures treatment quality, a model that represents this sentiment is

$$\min\{N : \min\{f(\mathcal{A}') : \mathcal{A}' \in \mathcal{P}(\mathcal{A}),\ |\mathcal{A}'| = N\} \le \varepsilon\}\,,$$

where ε defines the quality of an acceptable treatment.

As mentioned in the previous section, the judgment function is typically the objective value from fluence optimization. A common least-squares approach defines $X(\mathcal{A}')$ to be

$$\{x : x \ge 0, \sum_i x_{(a,i)} = 0 \quad \text{for } a \in \mathcal{A}\backslash\mathcal{A}'\}$$

and $f(\mathcal{A}')$ to be

$$\min\{\omega_T \cdot \|A_T x - TG\|_2 + \omega_C \cdot \|A_C x\|_2 + \omega_N \cdot \|A_N x\|_2 : x \in X(\mathcal{A}')\}\,, \tag{2}$$

where TG is a vector that expresses the target's treatment goal and ω_T, ω_C and ω_N weight the objective terms to express clinical desires. The prescription for this model is TG, but more complicated models with sophisticated prescriptions are common. In particular, dose-volume constraints that restrict the amount of each structure that is permitted to violate a bound are common. Readers interested in fluence optimization are directed to the entry on Cancer Radiation Treatment: Optimization Models.

Models

The N-beam selection problem is often addressed as a mixed integer problem. As an example, for the judgment function in (2) the N-beam selection problem can be expressed as

$$\left.\begin{array}{rl} \min & \omega_T \cdot \|A_T x - TG\|_2 + \omega_C \\ & \cdot\|A_C x\|_2 + \omega_N \cdot \|A_N x\|_2 \\ \text{subject to:} & \sum_i x_{(a,i)} \le M \cdot y_a,\ \text{for} a \in \mathcal{A} \\ & \sum_a y_a \le N \\ & x \ge 0 \\ & y \in \{0,1\}^{|\mathcal{A}|}\,, \end{array}\right\} \tag{3}$$

where M is an arbitrarily large value that bounds each beam's fluence. This is one of many possible models, with simple adjustments including the replacement of

the 2-norm with the 1 and ∞ norms, both of which result in a linear mixed integer problem.

A modest discretization of the sphere, with 72 great circles through the north and south poles equally spaced at 5 degrees at the equator and each great circle having beams equally spaced at 5 degrees, produces a set of 4902 candidate beams. This means the search tree associated with the mixed integer model above has $\binom{4902}{N}$ terminal nodes, which for the clinically valid $N = 10$ is approximately 2.2×10^{30}. Beyond the immenseness of this search space, branch-and-bound procedures are difficult for two reasons, 1) the number of N element subsets leading to near optimal solutions is substantial, and 2) the evaluation of the judgment function at each node requires the solution to an underlying fluence model, which in itself is time consuming. This inherit difficulty has driven the development of heuristic approaches, which separate into the two steps of: 1) assigning each beam a value that measures its worth to the overall treatment, and 2) using the individual beam values to select a collection of N beams. As a simple example, a scoring technique evaluates each beam and then simply selects the top N beams. The remainder of this section discusses several of the common heuristics.

A selection technique is called *informed* if it requires the evaluation of the underlying judgment function. One example would be to iteratively let $\mathcal{A}'$ be the singleton beam sets and evaluate $f(\mathcal{A}')$ for each. The N beams with the best scores would be selected for the treatment. If a selection method uses the data forming the optimization problem that defines f but fails to evaluate f, then the technique is called *weakly informed*. The preponderance of techniques suggested in the medical physics literature fall into this category. An example based solely on the dose matrix A is to value beam a with

$$\frac{\max_{(i,j)}\{A_{(j,a,i)} : j \in T\}}{\min_{(i,j)}\{A_{(j,a,i)} : j \in C \cup N\}},$$

where we assume the minimums in the denominator are nonzero. This ratio is high if a beam can deliver large amounts of dose to the target without damaging other tissues. A scoring technique based on this would terminate with the collection of N beams with the highest values. Since weakly informed methods do not require the solution of an optimization problem, they tend to be fast.

The concern about the size of the underlying fluence model has lead to a sampling heuristic that reduces the accuracy of the radiobiological model. Clinical relevance mandates that the anatomy be discretized so that dose is measured at distances no greater than 2 mm. For a $20\,\text{cm}^3$ portion of the anatomy, roughly the volume of the cranium, this means the coarsest 3D grid permitted in the clinic divides the anatomy into 10^6 sub-regions called voxels, which are indexed by j. Cases in the chest and abdomen are substantially larger and require a significant increase in the number of voxels. A natural question is whether or not all of these regions are needed for beam selection. One approach is to repeatedly sample these regions together with the candidate set of beams and solve (1). Each beam is valued by the number of times it has a high fluence. Beams with high values create $\mathcal{A}$ in (1) with j being indexed over all regions. The goal of this technique is to identify a candidate set of beams whose size is slightly larger than N, which keeps the search space manageable with the full compliment of voxels. The sampling procedure is crucial to the success of the procedure since it is known that beam selection depend on the collection of voxels.

Once beams are valued, there are many ways to use this information to construct a collection of favorable beams. As already discussed, common scoring methods select the best N beams. Another approach is based on set covering, which uses a high-pass filter to decide if a beam adequately treats the target. Allowing ε to be the threshold at which we say beam a treats position j within the target, we let

$$U_{(j,a)} = \begin{cases} 1, & \sum_i A_{(j,a,i)} \geq \varepsilon \\ 0, & \sum_i A_{(j,a,i)} < \varepsilon \end{cases},$$

for each $j \in T$. If each beam has a value of c_a, where low values are preferred, the set cover heuristic forms a collection of beams by solving

$$\min\left\{\sum_a c_a y_a : \sum_a U_{(j,a)} y_a \geq 1, \quad \text{for each } j \in T,\ y_a \in \{0,1\}\right\}. \quad (4)$$

This in itself is a binary optimization problem, and if ε is small enough to guarantee that every beam treats the target, which is typical, then the size of the search space is the same as the original problem in (1). How-

ever, the set cover problem has favorable solution properties, and this problem solves efficiently in practice. The search space decreases in size as ε increases, and designing an appropriate heuristic requires both a judicious selection of ε and an appropriate objective. This method can be informed or weakly informed depending on how the objective coefficients are constructed.

Another approach is to use the beam values as a probability distribution upon normalization. This allows one to address the problem probabilistically, a perspective that has been suggested within column generation and vector quantization. The column generation approach prices beams with respect to the likelihood that they will improve the judgment function, and beams with high probabilities are added to the current collection. The process of adding and deleting beams produces a sequence of beam sets $\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^n$, and problem (1) is solved with $\mathcal{A}$ replaced with $\mathcal{A}^k$, $k = 1, 2, \ldots, n$. Although it is possible for this technique to price all subsets of $\mathcal{A}$ whose cardinality is greater than N, which is significantly greater than the size of the original search space in (1), the pricing scheme tends to limit the number $\mathcal{A}^k$s.

The probabilistic perspective is further incorporated with heuristics based in information science. In particular, a method based on vector quantization, which is a modeling and solution procedure used in data compression, has been suggested. Allowing $\alpha(a)$ to be the probability associated with beam a, this heuristic constructs a collection of beams by solving

$$\min_{Q}\left\{\sum_{a}\alpha(a)\rho(a, Q(a)) : |Q(\mathcal{A})| = N\right\}, \tag{5}$$

where Q is a mapping from $\mathcal{A}$ into itself and ρ is a metric appropriate to the application. A common metric is to let $\rho(a, Q(a))$ be the arc length between a and $Q(a)$.

In the finite case, each N element subset, say $\mathcal{A}'$, of $\mathcal{A}$ uniquely defines Q by setting $Q(\mathcal{A}) = \mathcal{A}'$. Assuming this equality, we complete the definition by setting $Q(a) = a' \in \mathcal{A}'$ if and only if $\rho(a, a') \leq \rho(a, a'')$ for all $a'' \in \mathcal{A}'$, a condition referred to as the nearest neighbor condition. Since the optimization problem in (5) is defined over the collection of these functions, the size of the feasible region is the same as the original beam selection problem in (1). Unlike the set cover approach, which solves (4) to optimality, and the column generation technique, which repeatedly solves (1) to optimality with a restricted beam set, the vector quantization method often solves (5) heuristically. The most common heuristic is the Lloyd algorithm, a technique that begins with an initial collection of N beams and then iterates between

1. defining Q with the nearest neighbor condition, and
2. forming a new collection of beams with the centroids of $Q^{-1}(a)$, where beam a is in the current collection.

This technique guarantees that the objective in (5) decreases with each new collection.

Conclusions

Selecting beams is one of the three sub-problems in the design of radiotherapy treatments, a problem that currently does not have an appropriate solution outside the clinical practice of manually selecting beams through trial-and-error. However, research into automating the selection of beams with optimization is promising. We conclude with a few words on the totality of treatment design.

The overriding goal of treatment design is to remove the threat of cancer while sparing non-cancerous tissues. The status quo is to assume that a patient is static while designing a treatment. Indeed, treatment planners expand targeted regions to address the dynamic patient movement in the static approach, i. e. the target is increased to include the gross volume that contains the estimated movement of the actual target. The primary goal of the third phase of treatment design is to deliver the treatment as efficiently as possible to limit patient movement. This leads to a dilemma. The monotonicity property of the judgment function encourages treatments with many beams, but conventional wisdom dictates that the number of beams and the efficiency of the delivery are inversely proportional. However, in many settings the number of beams is a poor surrogate of efficiency. As an example, the most time demanding maneuver is to rotate the couch since it requires a technician to enter the treatment vault. So, treatments with many beams but fewer couch rotations are preferred to treatments with fewer beams but more couch rotations.

The point to emphasize from the previous paragraph is that the problem of selecting beams is always expressed in terms of the number of beams, which is a byproduct of the three-phase approach. Although the

separation of the design process into phases is natural and useful for computation, the division has drawbacks. Fluence models are large and difficult to solve, and every attempt is made to reduce their size. As already discussed, the voxels need to be under 2 mm^3 to reach clinical viability, and hence, the index set for j is necessarily large. The number and complexity of the sub-beams has increased dramatically with advanced technology, similarly making the index set for i large. This leaves the number of beams as the only control, and treatment designers are asked to select beams so that the fluence model is manageable. Years of experience have developed standard collections for many cancers, but asking a designer to select one of the 2.2×10^{30} possible collections for a 10 beam treatment in a non-standard case is daunting. A designer's instinct is to value a beam individually rather than as part of a collection. Several of the weakly informed selection methods from the medical physics literature have the same weakness. Such individual valuation typically identifies all but a few beams of a quality solution, but the last few are often unintuitive. Automating beam selection with an optimization process so that beams are considered within a collection is a step in the right direction.

The future of treatment design is to build global models and solution procedures that simultaneously address all three phases of treatment design. Such models are naturally viewed from the perspective of beam selection. What is missing is a judgment function that includes both fluence and delivery optimization. Learning how to model and solve these holistic models would alleviate the design process from a designer's (lack of) expertise and would provide a uniform level of care available to clinics with comparable technology. Such improvements are the promise of the field.

See also

- ► Credit Rating and Optimization Methods
- ► Evolutionary Algorithms in Combinatorial Optimization
- ► Optimization Based Frameworkfor Radiation Therapy

The literature on beam selection is mature within the medical physics community but is in its infancy within optimization. The five citations below cover the topics discussed in this article and contain bibliographies that adequately cite the work in medical physics.

References

1. Acosta R, Ehrgott M, Holder A, Nevin D, Reese J, Salter B (2007) Comparing Beam Selection Strategies in Radiotherapy Treatment Design: The Influence of Dose Point Resolution. In: Alves C, Pardalos P, Vicente L (eds) Optimization in Medicine, International Center for Mathematics, Springer Optimization and Its Applications. Springer, pp 1–25
2. Aleman D, Romeijn E, Dempsey J (2006) Beam orientation optimization methods in intensity modulated radiation therapy. IIE Conference Proceedings
3. Ehrgott M, Holder A, Reese J (2008) Beam Selection in Radiotherapy Design. In: Linear Algebra and Its Applications, vol 428. pp 1272–1312. doi:10.1016/j.laa.2007.05.039
4. Lim G, Choi J, Mohan R Iterative Solution Methods for Beam Angle and Fluence Map Optimization in Intensity Modulated Radiation Therapy Planning. to appear in OR Spectrum. doi:10.1007/s00291-007-0096-1
5. Lim G, Ferris M, Shepard D, Wright S, Earl M (2007) An Optimization Framework for Conformal Radiation Treatment Planning. INFORMS J Comput 19(3):366–380

Best Approximation in Ordered Normed Linear Spaces

HOSSEIN MOHEBI
Mahani Mathematical Research Center,
and Department of Mathematics,
University of Kerman, Kerman, Iran

MSC2000: 90C46, 46B40, 41A50, 41A65

Article Outline

Keywords and Phrases
Introduction
Metric Projection onto Downward and Upward Sets
Sets Z_+ and Z_-
Downward Hull and Upward Hull
Metric Projection onto a Closed Set
Best Approximation in a Class of Normed Spaces with Star-Shaped Cones
Characterization of Best Approximations
Strictly Downward Sets and Their Best Approximation Properties
References

Keywords and Phrases

Best approximation; Downward and upward sets; Global minimum; Necessary and sufficient conditions; Star-shaped set; Proximinal set

Introduction

We study the minimization of the distance to an arbitrary closed set in a class of ordered normed spaces (see [8]). This class is broad enough. It contains the space $C(Q)$ of all continuous functions defined on a compact topological space Q and the space $L^\infty(S, \Sigma, \mu)$ of all essentially bounded functions defined on a measure space (S, Σ, μ). It is assumed that these spaces are equipped with the natural order relation and the uniform norm. This class also contains direct products $X = \mathbb{R} \times Y$, where Y is an arbitrary normed space, with the norm $\|(c, y)\| = |c| + \|y\|$. The space X is equipped with the order relation induced by the cone $K = \{(c, y)\colon c \geq \|y\|\}$.

Let U be a closed subset of X, where X is a normed space from the given class, and let $t \in X$. We consider the problem $Pr(U, t)$:

$$\text{minimize} \quad \|u - t\| \quad \text{subject to} \quad u \in U. \tag{1}$$

It is assumed that there exists a solution of $Pr(U, t)$. This solution is called a metric projection of t onto U, or a best approximation of t by elements of U. We use the structure of the objective function in order to present necessary and sufficient conditions for the global minimum of $Pr(U, t)$ that give a clear understanding of the structure of a metric projection and can be easily verified for some classes of problems under consideration.

We use the so-called *downward and upward* subsets of a space X as a tool for analysis of $Pr(U, t)$. A set $U \subset X$ is called downward if $(u \in U, x \leq u) \Longrightarrow x \in U$. A set $V \subset X$ is called upward if $(v \in V, x \geq v) \Longrightarrow x \in V$. Downward and upward sets have a simple structure so the problem $Pr(U, t)$ can be easily analyzed for these sets U. If U is an arbitrary closed subset of X we can consider its downward hull $U_* = U - K$ and upward hull $U^* = U + K$, where $K = \{x \in X\colon x \geq 0\}$ is the cone of positive elements. These hulls can be used for examination of $Pr(U, t)$. We also suggest an approach based on a division of a normed space under consideration into two homogeneous not necessarily linear subspaces. A combination of this approach with the downward-upward technique allows us to give simple proofs of the proposed necessary and sufficient conditions.

Properties of downward and upward sets play a crucial role in this article. These properties have been studied in [6,13] for $X = \mathbb{R}^n$. We show that some results obtained in [6,13] are valid in a much more general case. In fact, the first necessary and sufficient conditions for metric projection onto closed downward sets in $\mathbb{R}^n$ have been given in [1, p. 132, Theorem 9]. Proposition 1(1) and (2) are extensions of $\mathbb{R}^n$ and $\mathbf{1} = (1, \ldots, 1)$, of [1, Proposition 1(a) and (b)], respectively. Also, Propositions 2 and 3 are extensions of [1, p. 116, Proposition 2]. Furthermore, Corollary 3 is an extension of [1, p. 116, Corollary 2 and p. 117, Remark 2]. In connection with Proposition 6, the downward hull U_* has been introduced in [1, Sect. 1], where the first results on the connection between $d(t, U)$ and $d(t, U_*)$ have been given, for the particular case where U is a normal subset of $\mathbb{R}^n_+$. We use methods of abstract convexity and monotonic analysis (see [11]) in this study.

Let X be a normed space. Let $K \subset X$ be a closed convex and pointed cone. (The latter means that $K \cap (-K) = \{0\}$.) The cone K generates the order relation $\geq$ on X. By definition $x \geq y \Longleftrightarrow x - y \in K$. We say that x is greater than y and write $x > y$ if $x - y \in K \setminus \{0\}$. Assume that K is solid, that is, the interior $\operatorname{int} K$ of K is nonempty. Let $\mathbf{1} \in \operatorname{int} K$. Using $\mathbf{1}$ we can define the following function:

$$p(x) = \inf\{\lambda \in \mathbb{R}\colon x \leq \lambda \mathbf{1}\}, \quad (x \in X). \tag{2}$$

It is easy to check that p is finite. It follows from (2) that

$$x \leq p(x)\mathbf{1}, \quad (x \in X). \tag{3}$$

It is easy to check (and well known) that p is a sublinear function, that is,

$$p(\lambda x) = \lambda p(x) \quad (\lambda > 0, x \in X),$$
$$p(x + y) \leq p(x) + p(y) \quad (x, y \in X).$$

We need the following definition (see [13] and references therein). A function $s\colon X \to \mathbb{R}$ is called topical if s is increasing: $x \geq y$ implies $s(x) \geq s(y)$ and $s(x + \lambda \mathbf{1}) = s(x) + \lambda$ for all $x \in X$ and $\lambda \in \mathbb{R}$.

It follows from the definition of p that p is topical. Consider the function

$$\|x\| := \max(p(x), p(-x))\,. \tag{4}$$

It is easy to check (and well known) that $\|\cdot\|$ is a norm on X. In what follows we assume that the norm (4) coincides with the norm of the space X.

It follows from (3) that

$$x \le \|x\|\mathbf{1}, \quad -x \le \|x\|\mathbf{1}, \quad (x \in X)\,. \tag{5}$$

The ball $B(t,r) = \{x \in X\colon \|x - t\| \le r\}$ has the form

$$B(t,r) = \{x \in X\colon t - r\mathbf{1} \le x \le t + r\mathbf{1}\}\,. \tag{6}$$

We now present three examples of spaces under consideration.

Example 1 Let X be a vector lattice with a strong unit $\mathbf{1}$. The latter means that for each $x \in X$ there exists $\lambda \in \mathbb{R}$ such that $|x| \le \lambda\mathbf{1}$. Then

$$\|x\| = \inf\{\lambda > 0\colon |x| \le \lambda\mathbf{1}\}\,,$$

where norm $\|\cdot\|$ is defined by (4). It is well known (see, for example, [21]) that each vector lattice X with a strong unit is isomorphic as a vector-ordered space to the space $C(Q)$ of all continuous functions defined on a compact topological space Q. For a given strong unit $\mathbf{1}$ the corresponding isomorphism ψ can be chosen in such a way that $\psi(\mathbf{1})(q) = 1$ for all $q \in Q$. The cone $\psi(K)$ coincides with the cone of all nonnegative functions defined on Q. If $X = C(Q)$ and $\mathbf{1}(q) = 1$ for all q, then

$$p(x) = \max_{q\in Q} x(q) \quad \text{and} \quad \|x\| = \max_{q\in Q} |x(q)|\,.$$

A well-known example of a vector lattice with a strong unit is the space $L^\infty(S, \Sigma, \mu)$ of all essentially bounded functions defined on a measure space (S, Σ, μ). If $\mathbf{1}(s) = 1$ for all $s \in S$, then $p(x) = \operatorname{ess\,sup}_{s\in S} x(s)$ and $\|x\| = \operatorname{ess\,sup}_{s\in S} |x(s)|$.

Example 2 Let $X = \mathbb{R} \times Y$, where Y is a normed space with a norm $\|\cdot\|$, and let $K \subset X$ be the epigraph of the norm $K = \{(\lambda, x)\colon \lambda \ge \|x\|\}$. The cone K is closed solid convex and pointed. It is easy to check and well known that $\mathbf{1} = (1,0)$ is an interior point of K. For each $(c,y) \in X$ we have

$$\begin{aligned} p(c,y) &= \inf\{\lambda \in \mathbb{R}\colon (c,y) \le \lambda\mathbf{1}\} \\ &= \inf\{\lambda \in \mathbb{R}\colon (\lambda,0) - (c,y) \in K\} \\ &= \inf\{\lambda \in \mathbb{R}\colon (\lambda - c, -y) \in K\} \\ &= \inf\{\lambda \in \mathbb{R}\colon \lambda - c \ge \|-y\|\} = c + \|y\|\,. \end{aligned}$$

Hence

$$\begin{aligned} \|(c,y)\| &= \max(p(c,y), p(-(c,y))) \\ &= \max(c + \|y\|, -c + \|y\|) = |c| + \|y\|\,. \end{aligned}$$

Example 3 Consider the space l^1 of all summable sequences with the usual norm. Let $Y = \{x = (x_i) \in l^1\colon x_1 = 0\}$. Then we can identify l^1 with the space $\mathbb{R} \times Y$. Let $y \in Y$ and $x = (x_1, y) \in l^1$. Then $\|x\| = |x_1| + \|y\|$. Let $K = \{x = (x_i) \in l^1\colon x_1 \ge \sum_{i=2}^\infty |x_i|\}$. Assume that l^1 is equipped with the order relation $\ge$ generated by K: if $x = (x_i)$ and $z = (z_i)$, then

$$x \ge z \iff x_1 - z_1 \ge \sum_{i=2}^\infty |x_i - z_i|\,.$$

Let $\mathbf{1} = (1, 0, \ldots, 0, \ldots)$. Consider the function p defined on l^1 by

$$p(x) = x_1 + \sum_{i=2}^\infty |x_i|, \quad x = (x_1, x_2, \ldots) \in l^1\,.$$

Then (see the previous example) $p(x) = \inf\{\lambda \in \mathbb{R}\colon x \le \lambda\mathbf{1}\}$ and $\|x\| = \sum_{i=1}^\infty |x_i|$ coincides with $\max(p(x), p(-x))$.

Let X be a normed vector space. For a nonempty subset U of X and $t \in X$, define $d(t,U) = \inf_{u\in U} \|t - u\|$. A point $u_0 \in U$ is called a metric projection of t onto U, or a best approximation of t by elements of U, if $\|t - u_0\| = d(t,U)$.

Let $U \subset X$. For $t \in X$, denote by $P_U(t)$ the set of all metric projections of t onto U:

$$P_U(t) = \{u \in U\colon \|t - u\| = d(t,U)\}\,. \tag{7}$$

It is wellknown that $P_U(t)$ is a closed and bounded subset of X. If $t \notin U$, then $P_U(t)$ is located in the boundary of U.

We shall use the following definitions. A pair (U,t) where $U \subset X$ and $t \in X$ is called *proximinal* if there exists a metric projection of t onto U. A pair (U,t) is called

Chebyshev if there exists a unique metric projection of t onto U. A set $U \subset X$ is called proximinal, if the pair (U, t) is proximinal for all $t \in X$. A set $U \subset X$ is called Chebyshev if the pair (U, t) is Chebyshev for all $t \in X$.

A set $U \subset X$ is called boundedly compact if the set $U_r = \{u \in U : \|u\| \le r\}$ is compact for each $r > 0$. (This is equivalent to the following: the intersection of a closed neighborhood of a point $u \in U$ with U is compact.) Each boundedly compact set is proximinal.

For any subset U of a normed space X we shall denote by int U, cl U, and bd U the interior, the closure, and the boundary of U, respectively.

Metric Projection onto Downward and Upward Sets

Definition 1 A set $U \subset X$ is called downward if $(u \in U, x \le u) \Longrightarrow x \in U$.

First we describe some simple properties of downward sets.

Proposition 1 *Let U be a downward subset of X and $x \in X$. Then the following assertions are true:*
(1) If $x \in U$, then $x - \varepsilon\mathbf{1} \in \operatorname{int} U$ for all $\varepsilon > 0$.
(2) $\operatorname{int} U = \{x \in X : x + \varepsilon\mathbf{1} \in U$ for some $\varepsilon > 0\}$.

Proof
(1) Let $\varepsilon > 0$ be given and $x \in U$. Let $N = \{y \in X : \|y - (x - \varepsilon\mathbf{1})\| < \varepsilon\}$ be an open neighborhood of $(x - \varepsilon\mathbf{1})$. Then, by (6) $N = \{y \in X : x - 2\varepsilon\mathbf{1} < y < x\}$. Since U is a downward set and $x \in U$, it follows that $N \subset U$, and so $x - \varepsilon\mathbf{1} \in \operatorname{int} U$.
(2) Let $x \in \operatorname{int} U$. Then there exists $\varepsilon_0 > 0$ such that the closed ball $B(x, \varepsilon_0) \subset U$. In view of (6), we get $x + \varepsilon_0\mathbf{1} \in U$.
Conversely, suppose that there exists $\varepsilon > 0$ such that $x + \varepsilon\mathbf{1} \in U$. Then, by (1): $x = (x + \varepsilon\mathbf{1}) - \varepsilon\mathbf{1} \in \operatorname{int} U$, which completes the proof. □

Corollary 1 *Let U be a closed downward subset of X and $u \in U$. Then, $u \in \operatorname{bd} U$ if and only if $\lambda\mathbf{1} + u \notin U$ for all $\lambda > 0$.*

Lemma 1 *The closure* cl U *of a downward set U is downward.*

Proof Let $x_k \in U$, $k = 1, 2, \ldots$, and $x_k \to x$ as $k \to +\infty$. Let $\|x_k - x\| = \varepsilon_k (k = 1, 2, \ldots)$. Using (6) we get $x - \varepsilon_k\mathbf{1} \le x_k$ for all $k \ge 1$. Since U is a downward set and $x_k \in U$ for all $k \ge 1$, we conclude that $x - \varepsilon_k\mathbf{1} \in U$ for all $k \ge 1$. Let $y \le x$ be arbitrary and $y_k = y - \varepsilon_k\mathbf{1} \le x - \varepsilon_k\mathbf{1} (k = 1, 2, \ldots)$. Then $y_k \in U (k = 1, \ldots)$. Since $y_k \to y$ as $k \to +\infty$, it follows that $y \in$ cl U. □

Proposition 2 *A closed downward subset U of X is proximinal.*

Proof Let $t \in X \setminus U$ be arbitrary and $r := d(t, U) = \inf_{u \in U} \|t - u\| > 0$. This implies that for each $\varepsilon > 0$ there exists $u_\varepsilon \in U$ such that $\|t - u_\varepsilon\| < r + \varepsilon$. Then, by (6):

$$-(r + \varepsilon)\mathbf{1} \le u_\varepsilon - t \le (r + \varepsilon)\mathbf{1}\,. \tag{8}$$

Let $u_0 = t - r\mathbf{1}$. Then

$$\|t - u_0\| = \|r\mathbf{1}\| = r = d(t, U)\,.$$

In view of (8), we have $u_0 - \varepsilon\mathbf{1} = t - r\mathbf{1} - \varepsilon\mathbf{1} \le u_\varepsilon$. Since U is a downward set and $u_\varepsilon \in U$, it follows that $u_0 - \varepsilon\mathbf{1} \in U$ for all $\varepsilon > 0$. The closedness of U implies $u_0 \in U$, and so $u_0 \in P_U(t)$. Thus the result follows. □

Remark 1 We proved that for each $t \in X \setminus U$ the set $P_U(t)$ contains the element $u_0 = t - r\mathbf{1}$ with $r = d(t, U)$. If $t \in U$, then $u_0 = t$ and $P_U(t) = \{u_0\}$.

Proposition 3 *Let U be a closed downward subset of X and $t \in X$. Then there exists the least element $u_0 := \min P_U(t)$ of the set $P_U(t)$, namely, $u_0 = t - r\mathbf{1}$, where $r := d(t, U)$.*

Proof If $t \in U$, then the result holds. Assume that $t \notin U$ and $u_0 = t - r\mathbf{1}$. Then, by Remark 1, $u_0 \in P_U(t)$. Applying (6) and the equality $\|t - u_0\| = r$ we get

$$x \ge t - r\mathbf{1} = u_0 \quad \forall\, x \in B(t, r)\,.$$

This implies that u_0 is the least element of the closed ball $B(t, r)$.

Now, let $u \in P_U(t)$ be arbitrary. Then $\|t - u\| = r$, and so $u \in B(t, r)$. Therefore, $u \ge u_0$. Hence, u_0 is the least element of the set $P_U(t)$. □

Corollary 2 *Let U be a closed downward subset of X, $t \in X$ and $u_0 = \min P_U(t)$. Then, $u_0 \le t$.*

Corollary 3 *Let U be a closed downward subset of X and $t \in X$ be arbitrary. Then*

$$d(t, U) = \min\{\lambda \ge 0 : t - \lambda\mathbf{1} \in U\}\,.$$

Proof Let $A = \{\lambda \geq 0 \colon t - \lambda \mathbf{1} \in U\}$. If $t \in U$, then $t - 0 \cdot \mathbf{1} = t \in U$, and so $\min A = 0 = d(t, U)$. Suppose that $t \notin U$; then $r := d(t, U) > 0$. Let $\lambda > 0$ be arbitrary such that $t - \lambda \mathbf{1} \in U$. Thus

$$\lambda = \|\lambda \mathbf{1}\| = \|t - (t - \lambda \mathbf{1})\| \geq d(t, U) = r\,.$$

Since, by Proposition 3, $t - r\mathbf{1} \in U$, it follows that $r \in A$. Hence, $\min A = r$, which completes the proof. □

The results obtained demonstrate that for the search of a metric projection of an element t onto a downward set U we need to solve the following optimization problem:

$$\text{minimize} \quad \lambda \quad \text{subject to} \quad t - \lambda \mathbf{1} \in U,\ \lambda \geq 0\,. \tag{9}$$

This is a one-dimensional optimization problem that is much easier than the original problem $Pr(U, t)$. Problem (9) can be solved, for example, by a common bisection procedure: first find numbers ρ_1 and σ_1 such that $t - \rho_1 \mathbf{1} \in U$ and $t - \sigma_1 \mathbf{1} \notin U$. Let $k \geq 1$. Assume that numbers ρ_k and σ_k are known such that $t - \rho_k \mathbf{1} \in U$ and $t - \sigma_k \mathbf{1} \notin U$. Then consider the number $\pi_k = 1/2(\rho_k + \sigma_k)$. If $t - \pi_k \mathbf{1} \in U$, then put $\rho_{k+1} = \pi_k$, $\sigma_{k+1} = \sigma_k$. If $t - \pi_k \mathbf{1} \notin U$, then put $\rho_{k+1} = \rho_k$, $\sigma_{k+1} = \pi_k$. The number $r = \lim_k \rho_k = \lim_k \sigma_k$ is the optimal value of (9).

The following necessary and sufficient conditions for the global minimum easily follow from the results obtained.

Theorem 1 *Let U be a closed downward set and $t \notin U$. Then $u_0 \in U$ is a solution of the problem $Pr(U, t)$ if and only if*

(i) $u_0 \geq \bar{u} := t - r\mathbf{1}$, *where* $r = \min\{\lambda \geq 0 \colon t - \lambda \mathbf{1} \in U\}$;

(ii) $p(t - u_0) \geq p(u_0 - t)$.

Proof Let $u_0 \in P_U(t)$. Since $\bar{u} := t - r\mathbf{1}$ is the least element of $P_U(t)$, it follows that $u_0 \geq \bar{u}$, so (i) is proved. We now demonstrate that (ii) is valid. In view of the equality $r = \|t - u_0\| = \max(p(t - u_0), p(u_0 - t))$, we conclude that $p(u_0 - t) \leq r$ and $p(t - u_0) \leq r$. We need to prove that $p(t - u_0) = r$. Assume on the contrary that $p(t - u_0) := \inf\{\lambda \colon t - u_0 \leq \lambda \mathbf{1}\} < r$. Then there exists $\varepsilon > 0$ such that $t - u_0 \leq (r - \varepsilon)\mathbf{1}$. This implies that $u_0 \geq t - r\mathbf{1} + \varepsilon \mathbf{1} = \bar{u} + \varepsilon \mathbf{1}$. Since $u_0 \in U$ and U is downward, it follows that $\bar{u} + \varepsilon \mathbf{1} \in U$, so $\bar{u}$ is an interior point of U. This contradicts the fact that $\bar{u}$ is a best approximation of t by U.

Assume now that both items (i) and (ii) hold. It follows from (i) that $t - u_0 \leq r\mathbf{1}$. Since p is a topical function, we conclude that $p(t - u_0) \leq r$. Item (ii) implies $\|t - u_0\| = p(t - u_0) \leq r$. Since $r = \min_{u \in U} \|t - u\|$, we conclude that $u \in P_U(t)$. □

We now turn to upward sets.

Definition 2 A set $V \subset X$ is called upward if $(v \in V, x \geq v) \Longrightarrow x \in V$.

Clearly V is upward if and only if $U = -V$ is downward, so all results obtained for downward sets can be easily reformulated for upward sets.

Proposition 4 *A closed upward subset V of X is proximinal.*

Proof This is an immediate consequence of Proposition 2. □

Theorem 2 *Let U be a closed upward set and $t \notin U$. Then u_0 is a solution of the problem $Pr(U, t)$ if and only if*

(i) $u_0 \leq t + r\mathbf{1}$, *where* $r = \min\{\lambda \geq 0 \colon t + \lambda \mathbf{1} \in V\}$.

(ii) $p(u_0 - t) \geq p(t - u_0)$.

Proof The result can be obtained by application of Theorem 1 to the problem $Pr(-U, -t)$. □

Corollary 4 *Let $V \subset X$ be a closed upward set and $t \in X$. Then $d(t, V) = \min\{\lambda \geq 0 \colon t + \lambda \mathbf{1} \in V\}$.*

Sets Z_+ and Z_-

Consider function s defined on X by

$$s(x) = \frac{1}{2}(p(x) - p(-x))\,.$$

We now indicate some properties of function s.

(1) s is homogeneous of degree one, that is, $s(\lambda x) = \lambda s(x)$ for $\lambda \in \mathbb{R}$. Indeed, we need to check that $s(-x) = -s(x)$ for all $x \in X$ and $s(\lambda x) = \lambda s(x)$ for all $x \in X$ and all $\lambda \in \mathbb{R}$. Both assertions directly follow from the definition of s.

(2) s is topical. It follows directly from the definition of s that s is increasing. We now check that $s(x + \mu \mathbf{1}) = s(x) + \mu$ for all $x \in X$ and all $\mu \in \mathbb{R}$.

Indeed,

$$\begin{aligned} s(x+\mu\mathbf{1}) &= \frac{1}{2}(p(x+\mu\mathbf{1}) - (p(-x-\mu\mathbf{1})) \\ &= \frac{1}{2}(p(x) - p(-x) + 2\mu) \\ &= s(x) + \mu\,. \end{aligned}$$

We will be interested in the level sets

$$Z_+ = \{x \in X\colon s(x) \geq 0\} \text{ and } Z_- = \{x \in X\colon s(x) \leq 0\}$$

of function s. The following holds:

$$x \in Z_+ \Longleftrightarrow p(x) \geq p(-x) \Longleftrightarrow p(x) = \|x\|\,.$$

$$x \in Z_- \Longleftrightarrow p(x) \leq p(-x) \Longleftrightarrow p(-x) = \|x\|\,.$$

Since s is homogeneous, it follows that $Z_- = -Z_+$. Let $Z_0 = \{x\colon s(x) = 0\}$. Then

$$Z_+ \cap Z_- = Z_0, \quad Z_- \cup Z_+ = X\,.$$

Since s is continuous, it follows that Z_+ and Z_- are closed subsets of X. Note that both Z_+ and Z_- are conic sets. (Recall that a set $C \subset X$ is called conic if $(x \in C,\ \lambda > 0) \Longrightarrow \lambda x \in C$).

Since s is increasing, it follows that Z_+ is upward and Z_- is downward. Let $R = \{\lambda\mathbf{1}\colon \lambda \geq 0\}$ be the ray passing through $\mathbf{1}$. In view of the topicality of s,

$$Z_+ = Z_0 + R, \quad Z_- = Z_0 - R\,.$$

Indeed, let $x \in Z_+$; then $s(x) := \lambda \geq 0$. Let $u = x - \lambda\mathbf{1}$. Then $s(u) = 0$, hence $u \in Z_0$. We demonstrated that $x \in Z_0 + R$, so $Z_+ \subset Z_0 + R$. The opposite inclusion trivially holds. Thus, $Z_+ = Z_0 + R$. We also have $Z_- = -Z_0 - R = Z_0 - R$. We now give some examples.

Example 4 Let $X = C(Q)$ be the space of all continuous functions defined on a compact topological space Q and $p(x) = \max_{q\in Q} x(q)$. Then $s(x) = \max_{q\in Q} x(q) + \min_{q\in Q} x(q)$; therefore $Z_0 = \{x \in C(Q)\colon \max_{q\in Q} x(q) = -\min_{q\in Q} x(q)\}$. Thus $x \in Z_0$ if and only if there exist points $q_+, q_- \in Q$ such that $|x(q_+)| = |x(q_-)| = \|x\|$ and $x(q_+) > 0$, $x(q_-) < 0$. Further, $x \in Z_+$ if and only if $\|x\| = \max_{q\in Q} x(q) > -\min_{q\in Q} x(q)$ and $x \in Z_-$ if and only if $\|x\| = \max_{q\in Q}(-x(q)) > -\min_{q\in Q}(-x(q)) = \max_{q\in Q} x(q)$.

Let Q consist of two points. Then $C(Q)$ coincides with $\mathbb{R}^2$ and $s(x) = x_1 + x_2$, that is, s is a linear function. If Q contains more than two points, then s is not linear.

Example 5 Let $X = \mathbb{R} \times Y$, where Y is a normed space (Example 2). Let $x = (c, y)$; then $p(x) = c + \|y\|$. Hence

$$s(x) = \frac{1}{2}[(c + \|y\|) - (-c + \| - y\|)] = c\,,$$

so s is linear. The following holds:

$$Z_0 = \{(c, y)\colon c = 0\}, \quad Z_+ = \{(c, y)\colon c \geq 0\},$$
$$Z_- = \{(c, y)\colon c \leq 0\}.$$

Example 6 Let $X = l^1$ (see Example 3). Then $s(x) = x_1$ and

$$Z_0 = \{x = (x_i) \in l^1\colon x_1 = 0\},$$
$$Z_+ = \{x = (x_i) \in l^1\colon x_1 \geq 0\},$$
$$Z_- = \{x = (x_i) \in l^1\colon x_1 \leq 0\}.$$

Downward Hull and Upward Hull

Let U be a subset of X. The intersection U_* of all downward sets that contain U is called the *downward* hull of U. Since the intersection of an arbitrary family of downward sets is downward, it follows that U_* is downward. Clearly U_* is the least (by inclusion) downward set, which contains U. The intersection U^* of all upward sets containing U is called the *upward* hull of U. The set U^* is upward and is the least (by inclusion) upward set containing U.

Proposition 5 ([15], Proposition 3) *Let $U \subset X$. Then*

$$U_* = U - K := \{u - v\colon u \in U,\ v \in K\},$$
$$U^* = U + K := \{u + v\colon u \in U,\ v \in K\}\,.$$

We need the following result:

Proposition 6 *Consider a closed subset U of X.*

(1) Let $t \in X$ be an element such that $t - U \subset Z_+$. Then $d(t, U) = d(t, U_*)$.

(2) Let $t \in X$ be an element such that $t - U \subset Z_-$. Then $d(t, U) = d(t, U^*)$.

Proof We shall prove only the first part of the proposition. The second part can be proved in a similar way. Let $r = d(t, U_*)$. Since $U \subset U_*$, it follows that $r \le d(t, U)$, so we need only check the reverse inequality. Let $u_* \in U_*$ be arbitrary. Then, by Proposition 5, there exist $u \in U$ and $v \in K$ such that $u_* = u - v$. Hence

$$t - u_* = t - u + v = x - u \text{ with } x := t + v \ge t\,.$$

By hypothesis, $t - u \in Z_+$. Since $x \ge t$ and Z_+ is upward, it follows that $x - u \in Z_+$. Since $\|z\| = p(z)$ for all $z \in Z_+$ and p is increasing, we have

$$\|t - u_*\| = \|x - u\| = p(x - u) \ge p(t - u) = \|t - u\|\,.$$

Thus for each $u_* \in U_*$ there exists $u \in U$ such that $\|t - u_*\| \ge \|t - u\|$. This means that $r := d(t, U_*) \ge d(t, U)$. We proved that $d(t, U) = r$. □

Proposition 7

(1) Let $t \in X$ be an element such that $t - U \subset Z_+$ and let U_* be a closed set. Then (U, t) is a proximinal pair.

(2) Let $t \in X$ be an element such that $t - U \subset Z_-$ and let U^* be a closed set. Then (U, t) is a proximinal pair.

Proof We shall prove only the first part of the proposition. Since U_* is a closed downward set in X, it follows, by Proposition 3, that the least element u_0 of the set $P_{U_*}(t)$ exists and $u_0 = t - r\mathbf{1}$, where $r = d(t, U_*)$. In view of Proposition 6, $r = d(t, U)$. Since $u_0 \in U_*$, by Proposition 5, there exist $u \in U$ and $v \in K$ such that $u_0 = t - r\mathbf{1} = u - v$. Then $t - u = r\mathbf{1} - v$ and

$$p(t - u) = p(r\mathbf{1} - v) \le p(r\mathbf{1}) = r\,.$$

Since, by hypothesis, $t - u \in Z_+$, it follows that $\|t - u\| = p(t - u) \le r$. On the other hand, $\|t - u\| \ge d(t, U) = r$. Hence $\|t - u\| = r$, and so $u \in P_U(t)$, which completes the proof. □

Remark 2 Let $U \subset X$ be a closed set. Assume that there exists a set $V \subset X$ such that $V \subset U \subset V_*$ and V_* is closed. Then $U_* = V_*$; hence U_* is closed. In particular, U_* is closed if there exists a compact set V such that $V \subset U \subset V_*$.

Proposition 7 can be used for the search of a metric projection of an element t onto a set U such that $t - U \subset Z_+$ and U_* is closed. In particular, we can give the following necessary and sufficient conditions for a solution of the problem $Pr(U, t)$ for these sets.

Theorem 3

(1) Let $t - U \subset Z_+$ and U_* is closed. Then $u_0 \in U$ is a solution of $Pr(U, t)$ if and only if

(i) $u_0 \ge t - r\mathbf{1}$ where $r = \min\{\lambda \ge 0\colon t - \lambda\mathbf{1} \in U - K\}$.

(ii) $p(t - u_0) \ge p(u_0 - t)$;

(2) Let $t - U \subset Z_-$ and U^* is closed. Then $u_0 \in U$ is a solution of $Pr(U, t)$ if and only if

(i′) $u_0 \le t + r\mathbf{1}$ where $r = \min\{\lambda \ge 0\colon t + \lambda\mathbf{1} \in U + K\}$.

(ii′) $p(u_0 - t) \ge p(t - u_0)$.

Proof We again prove only the first part of the theorem. Due to Proposition 6, we get $d(t, U) = d(t, U_*) = r$. Since U_* is closed and downward, it follows (Proposition 3) that $\bar{u} := t - r\mathbf{1} \in P_{U_*}(t)$. Let $u_0 \ge \bar{u}$ and $u_0 \in U$. Then $u_0 \in U_*$ and in view of Proposition 6, it holds:

$$d(u_0, U) = d(u_0, U_*) = r = \min\{\lambda \ge 0\colon t - \lambda\mathbf{1} \in U_*\}\,.$$

Applying Theorem 1 we conclude that u_0 is a best approximation of t by U_*. Since $u_0 \in U$, it follows that u_0 is a best approximation of t by U.

Consider now a best approximation u_0 of t by U. Applying again Proposition 6 we deduce that $\|t - u_0\| = d(t, U) = d(t, U_*) = r$. Theorem 1 demonstrates that both (i) and (ii) hold. □

Metric Projection onto a Closed Set

Downward and upward sets can be used for examination of best approximations by arbitrary closed sets (it is assumed that a metric projection exists).

We start with the following assertion.

Proposition 8 *Let U be a closed subset of X and $t \in X$. Consider the following sets:*

$$U_t^+ = U \cap (t - Z_+), \qquad U_t^- = U \cap (t - Z_-)\,. \quad (10)$$

Then

(1) $t - U_t^+ \subset Z_+$, $t - U_t^- \subset Z_-$.

(2) $U_t^+ \cup U_t^- = U$.

(3) $U_t^+ \cap U_t^- = U \cap (t - Z_0)$, *where* $Z_0 = \{x \in X : s(x) = 0\}$.
(4) U_t^+ *and* U_t^- *are closed.*
(5) *If* U *is downward, then* U_t^+ *is downward; if* U *is upward, then* U_t^- *is upward.*

Proof
(1) It is easy to check

$$(t - U) \cap Z_+ = t - [U \cap (t - Z_+)] = t - U_t^+ .$$

Hence $t - U_t^+ \subset Z_+$. A similar argument shows that $t - U_t^- \subset Z_-$.
(2) The following holds:

$$\begin{aligned} U_t^+ \cup U_t^- &= [(t - Z_+) \cap U] \cup [(t - Z_-) \cap U] \\ &= [(t - Z_+) \cup (t - Z_-)] \cap U \\ &= [t - (Z_+ \cup Z_-)] \cap U . \end{aligned}$$

Since $Z_+ \cup Z_- = X$, it follows that $U_t^+ \cup U_t^- = U$.
(3) The following holds:

$$\begin{aligned} U_t^+ \cap U_t^- &= [U \cap [(t - Z_+)] \cap [U \cap (t - Z_-)] \\ &= U \cap [(t - Z_+) \cap (t - Z_-)] \\ &= U \cap [t - (Z_+ \cap Z_-)] . \end{aligned}$$

Since $Z_+ \cap Z_- = Z_0$, the result follows.
(4) This is clear.
(5) It follows from the fact that $t - Z_+$ is downward and $t - Z_-$ is upward. □

Consider a fixed proximinal pair (U, t). Let U_t^+ and U_t^- be the sets defined by (10). Since $U_t^+ \cup U_t^- = U$, it follows that

$$\inf_{u \in U} \|t - u\| = \min\Big(\inf_{u^+ \in U_t^+} \|t - u^+\|, \inf_{u^- \in U_t^-} \|t - u^-\| \Big) . \tag{11}$$

It follows from (11) that at least one of the pairs (U_t^+, t) and (U_t^-, t) is proximinal and a metric projection of t onto U coincides with a metric projection onto at least one of the sets U_t^+ or U_t^-. Let

$$\begin{aligned} r_+ &= \inf_{u \in U_t^+} \|t - u\|, \\ r_- &= \inf_{u \in U_t^-} \|t - u\|, \\ r &= \inf_{u \in U} \|t - u\| = \min(r_+, r_-) . \end{aligned} \tag{12}$$

For examination of metric projections of t onto U we need to find numbers r_+ and r_-. The number r_+ can be found by solving a one-dimensional optimization problem of the form (9); r_- can be found by solving a similar problem.

If $r_+ < r_-$, then a metric projection of t onto U coincides with a metric projection of t onto U_t^+. Since $t - U_t^+ \subset Z_+$, we can use the results of this section for analyzing the problem $Pr(U, t)$ and its solution. In particular, if the downward hull $(U_t^+)_*$ of the set U_t^+ is closed, we can assert that the set $P_U(t)$ coincides with the set $P_{U_t^+}(t)$. Using Theorem 3 we can give necessary and sufficient conditions for the global minimum in this case in terms of the set U_t^+. They can be expressed in the following form:

$$\begin{aligned} P_U(t) = P_{U_t^+}(t) = \{u \in U_t^+ : u \geq t - r_+ \mathbf{1}, \\ p(t - u) \geq p(u - t)\} . \end{aligned}$$

If $r_- < r_+$, then a metric projection of t onto U coincides with a metric projection of t onto U_t^-. If the set $(U_t^-)^*$ is closed, we can assert that

$$\begin{aligned} P_U(t) = P_{U_t^-}(t) = \{u \in U_t^- : u \leq t + r \mathbf{1}, \\ p(u - t) \geq p(t - u)\} . \end{aligned}$$

If $r_- = r_+$, then we can use both sets U_t^+ and U_t^-.

We assume in the rest of this section that both pairs (U_t^+, t), (U_t^-, t) are proximinal. In particular, these pairs are proximinal for arbitrary t, if U is a locally compact set.

We are now interested in metric projections u of t onto U such that $s(u - t) = 0$. We introduce the following definition.

Definition 3 A pair (U, t) with $U \subset X$, $t \in X$ is called *strongly proximinal* if $s(u - t) = 0$ for each metric projection u of t onto U.

Recall that $s(u - t) = 0$ if and only if $u - t \in Z_+ \cap Z_-$.

Proposition 9 *The following assertions (i) and (ii) are equivalent:*
(i) (U, t) *is a strongly proximinal pair;*
(ii) $P_U(t) = P_{U_t^+}(t) \cap P_{U_t^-}(t)$.

Proof
(i) $\Longrightarrow$ (ii). Let $u \in P_U(t)$. Since $u - t \in Z_- = -Z_+$ and $u \in U$, it follows that $u \in U \cap (t - Z_+) = U_t^+$.

Then $\|t-u\| = \min_{u'\in U} \|t-u'\| \le \min_{u'\in U_t^+} \|t-u'\|$. Since $u \in U_t^+$, we conclude that the equality $\|t-u\| = \min_{u'\in U_t^+} \|t-u'\|$ holds. Thus $u \in P_{U_t^+}(t)$. A similar argument shows that $u \in P_{U_t^-}(t)$. Let $u \in P_{U_t^+}(t) \cap P_{U_t^-}(t)$. Then

$$\|u-t\| = d(t, U_t^+) = d(t, U_t^-)\,.$$

Combining the equality $U = U_t^+ \cup U_t^-$ with (11), we get $\|u-t\| = \min_{u'\in U} \|u'-t\|$, and hence $u \in P_U(t)$.

(ii) $\Longrightarrow$ (i). Since (ii) holds, it follows that

$$\begin{aligned} P_U(t) &= P_{U_t^+}(t) \cap P_{U_t^-}(t) \\ &= \{u \in U_t^+ : t - r\mathbf{1} \le u\} \\ &\qquad \cap \{u \in U_t^- : u \le t + r\mathbf{1}\} \\ &= \{u \in U_t^+ \cap U_t^- : t - r\mathbf{1} \le u \le t + r\mathbf{1}\}\,. \end{aligned}$$

Applying Proposition 8 (3), we conclude that

$$\begin{aligned} P_U(t) &= \{u \in U \cap (t - Z_0) : t - r\mathbf{1} \le u \le t + r\mathbf{1}\} \\ &= U \cap (t - Z_0) \cap B(t, r)\,. \end{aligned}$$

Since $P_U(t) = U \cap B(t,r)$ (by definition), it follows that $P_U(t) \subset t - Z_0$, that is, the pair (U,t) is strongly proximinal. □

Let (U,t) be a proximinal pair. We are interested in a description of conditions that guarantee that $\tilde{v} := t - \tilde{u}$, where $\tilde{u}$ is a metric projection of t onto U, belongs to $Z_+ \cap Z_- = Z_0$. First, we give the following definition:

Definition 4 We say that a set $U \subset X$ is weakly K-open if for each $u \in U$ there exists an element $q \in \operatorname{int} K$ such that $u + \delta q \in U$ for all δ with a small enough $|\delta|$.

Proposition 10 *Assume that (U, t) is a proximinal pair such that the set U is weakly K-open. Let $\tilde{u} \in P_U(t)$. Then $\tilde{v} := t - \tilde{u} \in Z_0$.*

Proof Let $\tilde{v} \notin Z_0$; then $\tilde{v} \notin (Z_+ \cap Z_-)$. Assume for the sake of definiteness that $\tilde{v} \in Z^+$, that is, $\|\tilde{v}\| = p(\tilde{v}) > p(-\tilde{v})$. Since U is weakly K-open and $\tilde{u} \in U$, it follows that there exists $q \in \operatorname{int} K$ such that $\tilde{u} + \delta q \in U$ for all small enough $\delta > 0$. Then:

$$p(\tilde{v}) > p(\tilde{v} - \delta q) \ge p(-\tilde{v} + \delta q) = p(-(\tilde{v} - \delta q))\,.$$

Hence $\|\tilde{v} - \delta q\| = p(\tilde{v} - \delta q) < p(\tilde{v}) = \|\tilde{v}\|$. Let $\bar{u} = \tilde{u} + \delta q$. Because U is weakly K-open, we conclude that $\bar{u} \in U$ for all small enough $\delta > 0$. Since $\tilde{v} - \delta q = t - \tilde{u} - \delta q = t - \bar{u}$, we obtain

$$\min_{u\in U} \|t-u\| \le \|t-\bar{u}\| = \|\tilde{v}-\delta q\| < \|\tilde{v}\| = \|t-\tilde{u}\|\,.$$

This is a contradiction because $\tilde{u} \in P_U(t)$. □

Example 7 Let $U' \subset X$ be a locally compact set and $q \in \operatorname{int} K$. Consider the set

$$U = U' + \{\lambda q : \lambda \in \mathbb{R}\} = \{u' + \lambda q : u' \in U', \lambda \in \mathbb{R}\}\,.$$

Clearly U is a locally compact set and U is weakly K-open. Then for each $t \in X$ the pair (U,t) is strongly proximinal.

Best Approximation in a Class of Normed Spaces with Star-Shaped Cones

The theory of best approximation by elements of convex sets in normed linear spaces is well developed and has found many applications [1,2,4,5,10,16,17,18,19,20]. However, convexity is sometimes a restrictive assumption, and therefore the problem arises of how to examine best approximation by not necessarily convex sets. Special tools for this are needed.

The aim of the present article is to develop a theory of best approximation by elements of closed sets in a class of normed spaces with star-shaped cones (see [9]). A star-shaped cone K in a normed space X generates a relation $\le_K$ on X, which is an order relation if and only if K is convex. It can be shown that each star-shaped cone K, such that the interior of the kernel K is not empty, can be represented as the union of closed solid convex pointed cones K_i ($i \in I$, where I is an index set) such that the interior of the cone $K_* := \cap_{i\in I} K_i$ is not empty. A point $\mathbf{1} \in \operatorname{int} K_*$ generates the norm $\|\cdot\|_*$ on X, where $\|x\|_* = \inf\{\lambda > 0 : x \le_{K_*} \lambda\mathbf{1}, -x \le_{K_*} \lambda\mathbf{1}\}$, and we assume that X is equipped with this norm. In the special case $I = \{1\}$ (that is, K is a closed convex solid pointed cone) the class of spaces under consideration contains such Banach lattices as the space $L^\infty(S, \Sigma, \mu)$ of all essentially bounded functions defined on a measure space (S, Σ, μ) and the space $C(Q)$ of all continuous functions defined on a compact topological space Q.

Now, let X be a normed space and $U \subset X$. The set $\operatorname{kern} U$ consisting of all $u \in U$ such that

$(x \in U,\ 0 \le \alpha \le 1) \Longrightarrow u + \alpha(x-u) \in U$ is called the convex kernel of U. A nonempty set U is called star-shaped if kern U is not empty. It is known (see, for example, [12]) that kern U is convex for an arbitrary star-shaped set U. If U is closed, then kern U is also closed. Indeed, let $u_k \in \operatorname{kern} U$, $k = 1, \ldots$ and $u_k \to u$. For each $k = 1, 2, \ldots$, $x \in U$ and $\alpha \in [0,1]$, we have $u_k + \alpha(x - u_k) \in U$, and so $u + \alpha(x-u) \in U$. This means that $u \in \operatorname{kern} U$.

We need the following statement.

Proposition 11 *Let $U \subset X$ be a set and let $u \in U$. Then the following assertions are equivalent:*

(i) There exists $\varepsilon > 0$, an index set I, and a family of convex sets $(U_i)_{i \in I}$ such that

$$U = \bigcup_{i \in I} U_i \quad \text{and} \quad U_i \supset B(u, \varepsilon) \ \ (i \in I). \tag{13}$$

(ii) U is a star-shaped set and $u \in \operatorname{int} \operatorname{kern} U$.

Proof

(i) $\Longrightarrow$ (ii). Let $z \in B(u, \varepsilon)$ and let $x \in U$, $\alpha \in [0,1]$. It follows from (13) that there exists $i \in I$ such that $x \in U_i$. Since U_i is convex and $z \in B(u,\varepsilon) \subset U_i$, we conclude that $z + \alpha(x - z) \in U_i \subset U$. Hence, $z \in \operatorname{kern} U$ for each $z \in B(u,\varepsilon)$, and so $B(u,\varepsilon) \subset \operatorname{kern} U$.

(ii) $\Longrightarrow$ (i). Let $I = U$. Since $u \in \operatorname{int} \operatorname{kern} U$, it follows that there exists $\varepsilon > 0$ such that $B(u,\varepsilon) \subset \operatorname{kern} U$. Let $x \in U$ and $U_x = \operatorname{co}(x \cup B(u,\varepsilon))$. Then the set U_x is convex and closed and $x \in U_x$. Hence, $U \subset \bigcup_{x \in U} U_x$. Applying the definition of the convex kernel we conclude that $U_x \subset U$. Hence, $\bigcup_{x \in U} U_x \subset U$. □

If $0 \in \operatorname{kern} U$, then the Minkowski gauge μ_U of U can be defined as follows:

$$\mu_U(x) = \inf\{\lambda > 0 \colon x \in \lambda U\}. \tag{14}$$

(It is assumed that $\inf \emptyset = 0$.)

Let $u \in \operatorname{kern} U$. Then, $0 \in \operatorname{kern}(U - u)$, and so we can consider the Minkowski gauge μ_{U-u} of the set $U - u$.

Theorem 4 *Let $u \in \operatorname{int} \operatorname{kern} U$. Then the Minkowski gauge μ_{U-u} of the set $U - u$ is Lipschitz.*

Theorem 4 has been proved in [11] (Theorem 5.2) for finite-dimensional spaces. The proof from [11] holds for an arbitrary normed space and we omit it.

In the sequel, we shall study star-shaped cones. Recall that a set $K \subset X$ is called a cone (or conic set) if $(\lambda > 0,\ x \in K) \Longrightarrow \lambda x \in K$. Let K be a star-shaped cone and $K_* = \operatorname{kern} K$. Then, K_* is also a cone. Indeed, let $u \in K_*$, $\lambda > 0$ and $x \in K$. Let $x' = x/\lambda$. Then, $x' \in K$, and so $u + \alpha(x' - u) \in K$ for all $\alpha \in [0,1]$. We have $\lambda u + \alpha(\lambda x' - \lambda u) = \lambda u + \alpha(x - \lambda u) \in K$. Since x is an arbitrary element of K, it follows that $\lambda u \in \operatorname{kern} K = K_*$. We now give an example.

Example 7 Let X coincide with the space $C(Q)$ of all continuous functions defined on a compact metric space Q and $K = \{x \in C(Q) \colon \max_{q \in Q} x(q) \ge 0\}$. Clearly K is a nonconvex cone. It is easy to check that K is a star-shaped cone and kern $K = K_+$, where

$$\begin{aligned} K_+ &= \{x \in C(Q) \colon x(q) \ge 0 \text{ for all } q \in Q\} \\ &= \{x \in C(Q) \colon \min_{x \in Q} x(q) \ge 0\}. \end{aligned}$$

Indeed, let $u \in K_+$. Consider a point $x \in K$. Then there exists a point $q' \in Q$ such that $x(q') \ge 0$. Since $u(q) \ge 0$ for all $q \in Q$, it follows that $\alpha u(q') + (1-\alpha)x(q') \ge 0$ for all $\alpha \in [0,1]$. Therefore, $\alpha u + (1-\alpha)x \in K$. We proved that $K_+ \subset \operatorname{kern} K$. Now, consider $u \notin K_+$. Then there exists a point q' such that $u(q') < 0$. Since u is continuous, we can find an open set $G \subset Q$ such that $u(q) < 0$ for $q \in G$. Let $x \in K$ be a function such that $x(q) < 0$ for all $q \notin G$ (such a function exists). Since the set $Q \setminus G$ is compact, it follows that $\max_{q \notin G} x(q) < 0$; hence $\alpha x(q) + (1-\alpha)u(q) < 0$ for all $q \in Q$ and small enough $\alpha > 0$. Therefore $\alpha x + (1-\alpha)u \notin K$ for these numbers α. The equality kern $K = K_+$ has been proved. Note that int kern $K \ne \emptyset$.

The following statement plays an important role in this paper.

Theorem 5 *Let $K \subset X$ be a closed cone and let $u \in K$. Then the following assertions are equivalent:*

(i) There exists $\varepsilon > 0$, an index set I and a family of closed convex cones $(K_i)_{i \in I}$ such that

$$K = \bigcup_{i \in I} K_i \quad \text{and} \quad K_i \supset B(u, \varepsilon) \ \ (i \in I). \tag{15}$$

(ii) K is a star-shaped cone and $u \in \operatorname{int} \operatorname{kern} K$.

Proof

(i) $\Longrightarrow$ (ii). It follows from Proposition 11 that K is a star-shaped set and $u \in \operatorname{int} \operatorname{kern} K$. Since K_i is a cone for each $i \in I$, it follows that K is a cone.

(ii) $\Longrightarrow$ (i). In view of Proposition 11, there exists a family of convex sets U_i, $(i \in I)$ such $U_i \supset B(u,\varepsilon)$ and $K = \bigcup_{i\in I} U_i$. Let K_i be the closed conic hull of U_i: $K_i = \operatorname{cl} \bigcup_{\lambda>0} \lambda U_i$. Then $K = \bigcup_{i\in I} K_i$. □

Remark 3

(1) Let K be a closed star-shaped cone with $\operatorname{int}\operatorname{kern} K \neq \emptyset$. Then the set $K_* = \operatorname{kern} K$ is a closed solid convex cone. (Recall that a convex cone K is called solid if $\operatorname{int} K \neq \emptyset$.)

(2) Note that in Theorem 5, the family $(K_i)_{i\in I}$ can be chosen such that each K_i is a closed solid pointed convex cone. Indeed, if $u \in \operatorname{int}\operatorname{kern} K$, then $u \neq 0$ and a neighborhood $B(u,\varepsilon) \subset \operatorname{kern} K$ can be chosen in such a way that $0 \notin B(u,\varepsilon)$. Then the closed conic hull $K_i = \operatorname{cl} \bigcup_{\lambda>0} \lambda U_i$ is a closed solid pointed convex cone.

Let K be a star-shaped cone and $K = \bigcup_{i\in I} K_i$, where K_i is a convex cone and $K_* = \bigcap_{i\in I} K_i$. Then $\operatorname{kern} K \supset K_*$. Indeed, let $u \in K_*$ and $x \in K$. Then there exists $j \in I$ such that $x \in K_j$. The inclusion $u \in \bigcap_{i\in I} K_i$ implies that $u \in K_j$. Since K_j is a convex cone, it follows that $\alpha x + (1-\alpha)u \in K_j$ for all $\alpha \in (0,1)$. This means that $u \in \operatorname{kern} K$.

Let K be a closed star-shaped cone and $u \in \operatorname{int}\operatorname{kern} K$. Consider the function

$$p_{u,K}(x) = \inf\{\lambda \in \mathbb{R} \colon \lambda u - x \in K\}\,. \tag{16}$$

Functions (16) are well known if K is a convex cone. These functions have been defined and studied in [12] for the so-called strongly star-shaped cones (see [11] for the definition of strongly star-shaped sets). Each star-shaped set U with $\operatorname{int}\operatorname{kern} U \neq \emptyset$ is strongly star-shaped. (It was shown in [11] for finite-dimensional space; however, the same argument is valid for arbitrary normed spaces.) It was shown [12] that $p_{u,K}$ is a finite positively homogeneous function of the first degree and the infimum in (16) is attained, so $p_{u,K}(x)u - x \in \operatorname{int} K$. The following equality holds:

$$p_{u,K}(x - \gamma u) = \mu_{K-u}(\gamma u - x)\,, \tag{17}$$

where μ_{K-u} is the Minkowski gauge of $K-u$. In view of Theorem 4, the function μ_{K-u} is Lipschitz, therefore $p_{u,K}$ is also Lipschitz. If K is a convex cone, then $p_{u,K}$ is a sublinear function. This function is also increasing in the sense of the order relation induced by the convex cone K. The following assertion holds (see [12]).

Proposition 12 *Let K be a star-shaped cone and $u \in \operatorname{int}\operatorname{kern} U$. Then:*

$$p_{u,K}(x+\lambda u) = p_{u,K}(x) + \lambda\,, \qquad x \in X,\ \lambda \in \mathbb{R} \tag{18}$$

and

$$\{x \colon p_{u,K}(x) \le \lambda\} = \lambda u - K, \qquad \lambda \in \mathbb{R}. \tag{19}$$

We also need the following assertion.

Proposition 13 *Let $(K_i)_{i\in I}$ be a family of closed star-shaped cones such that $\bigcap_{i\in I} \operatorname{int}\operatorname{kern} K_i \neq \emptyset$. Let $u \in \bigcap_{i\in I} \operatorname{int}\operatorname{kern} K_i$. Let $K = \bigcup_{i\in I} K_i$ and $K_* = \bigcap_{i\in I} K_i$. Then*

$$p_{u,K}(x) = \inf_{i\in I} p_{u,K_i}(x)\,,$$
$$p_{u,K_*}(x) = \sup_{i\in I} p_{u,K_i}(x), \quad (x \in X)\,.$$

Proof Let L be a cone such that $u \in \operatorname{int}\operatorname{kern} L$. For each $x \in X$ consider the set $\Lambda_x(L) = \{\lambda \in \mathbb{R} \colon \lambda u - x \in L\}$. It was proved in [12], Proposition 1, that this set is a closed segment of the form $[\lambda_x, +\infty)$, where $\lambda_x = p_{u,L}(x)$. We have

$$\begin{aligned}\Lambda_{x,K} &= \{\lambda \in \mathbb{R} \colon \lambda u \in x + \bigcup_{i\in I} K_i\} \\ &= \{\lambda \in \mathbb{R} \colon \lambda u \in \bigcup_{i\in I} (x + K_i)\} \\ &= \bigcup_{i\in I} \{\lambda \in \mathbb{R} \colon \lambda u \in x + K_i\} = \bigcup_{i\in I} \Lambda_{x,K_i}\,.\end{aligned}$$

Hence

$$\begin{aligned}p_{u,K}(x) &= \inf \Lambda_{x,K} = \inf \bigcup_{i\in I} \Lambda_{x,K_i} \\ &= \inf_{i\in I} \inf \Lambda_{x,K_i} = \inf_{i\in I} p_{u,K_i}(x)\,.\end{aligned}$$

The second part of the proposition can be proved by a similar argument. □

Let K be a closed star-shaped cone with $\operatorname{int}\operatorname{kern} K \neq \emptyset$. Then K can be represented as the union of a family of closed convex cones $(K_i)_{i\in I}$. One such family has been described in the proofs of Proposition 11 and Theorem 5: $I = K$, $K_i = \operatorname{cl}\operatorname{cone}\operatorname{co}\{i \cup B(u,\varepsilon)\}$,

where $u \in \operatorname{int}\operatorname{kern} K$ and $\varepsilon > 0$ so small such that $B(u,\varepsilon) \subset \operatorname{kern} K$. This family is very large; often we can find a much simpler presentation. For example, assume that a cone K is given as the union of a family of closed convex cones $(K_i)_{i\in I}$ such that the cone $\bigcap_{i\in I} K_i$ has a nonempty interior. Then this cone is contained in $\operatorname{kern} K$; we can use the given cones K_i in such a case. We always assume that cones K_i are pointed for all $i \in I$, that is, $K_i \cap (-K_i) = \{0\}$.

An arbitrary star-shaped cone K induces a relation $\geq_K$ on X, where $x \leq_K y$ means that $y - x \in K$. This relation is a preorder relation if and only if K is a convex set. Although $\geq_K$ is not necessarily an order relation, we will say that x is greater than or equal to y in the sense of K if $x \geq_K y$. We say that x is greater than y and write $x >_K y$ if $x - y \in K \setminus \{0\}$. Let $K = \bigcup_{i\in I} K_i$, where K_i is a convex cone. The cone K_i induces the order relation $\geq_{K_i}$. The relation $\geq_K$, which is induced by cone K, can be represented in the following form:

$$x \geq_K y \quad \text{if and only if there exists} \quad i \in I \quad \text{such that } x \geq_{K_i} y\,. \tag{20}$$

In the rest of this article, we assume that X is equipped with a closed star-shaped cone K with $\operatorname{int}\operatorname{kern} K \neq \emptyset$. We also assume that a family $(K_i)_{i\in I}$ of closed solid convex pointed cones K_i is given such that $K = \bigcup_{i\in I} K_i$ and $K_* = \bigcap_{i\in I} K_i$ has a nonempty interior. Let an element $\mathbf{1} \in \operatorname{int} K_*$ be fixed. It is clear that $\mathbf{1} \in \operatorname{int} K_i$ for all $i \in I$. We will also use the following notations:

$$p_{\mathbf{1},K} = p, \quad p_{\mathbf{1},K_i} = p_i, \quad p_{\mathbf{1},K_*} = p_*\,. \tag{21}$$

It follows from Proposition 13 that

$$p(x) = \inf_{i\in I} p_i(x), \quad p_*(x) = \sup_{i\in I} p_i(x)\,. \tag{22}$$

A function $f\colon X \to \mathbb{R}$ is called plus-homogeneous (with respect to $\mathbf{1}$) if

$$f(x + \lambda\mathbf{1}) = f(x) + \lambda \text{ for all } x \in X \text{ and } \lambda \in \mathbb{R}\,.$$

(The term *plus homogeneous* was coined in [13].) It follows from (18) that p_i $(i \in I)$, p and p_* are plus-homogeneous functions.

Let

$$B_i = \{x \in X\colon \mathbf{1} \geq_{K_i} x \geq_{K_i} -\mathbf{1}\} \quad i \in I\,. \tag{23}$$

Since K_i is a closed solid convex pointed cone, it is easy to check that B_i $(i \in I)$ can be considered as the unit ball of the norm $\|\cdot\|_i$ defined on X by

$$\|x\|_i := \max(p_i(x), p_i(-x)) \quad x \in X\,. \tag{24}$$

Let

$$\|x\|_* = \sup_{i\in I} \|x\|_i \quad (x \in X;\ i \in I)\,. \tag{25}$$

We now show that $\|x\|_* < +\infty$ for each $x \neq 0$. Indeed, since $\mathbf{1} \in \operatorname{int} K_* \subset \operatorname{int} K_i$, it follows that there exists $\varepsilon > 0$ such that $\mathbf{1} + \varepsilon\tilde{B} \subset K_i$ for all $i \in I$, where $\tilde{B} = \{x \in X\colon \|x\| \leq 1\}$ is the closed unit ball with respect to the initial norm $\|\cdot\|$ of the normed space X. Let $x \neq 0$. Then $x' = (\varepsilon/\|x\|)x \in \varepsilon\tilde{B}$; hence $\mathbf{1} - x' \in K_i$. This implies that

$$p_i(x') = \inf\{\lambda \in \mathbb{R}\colon \lambda\mathbf{1} - x' \in K_i\} \leq 1\,.$$

Since p_i is a positively homogeneous function, it follows that

$$p_i(x) = p_i\left(\frac{\|x\|}{\varepsilon}x'\right) = \frac{\|x\|}{\varepsilon}p_i(x') \leq \frac{\|x\|}{\varepsilon}\,.$$

The same argument demonstrates that $p_i(-x) \leq \|x\|/\varepsilon$. Hence

$$\|x\|_* = \sup_{i\in I} \|x\|_i = \sup_{i\in I} \max(p(x_i),\ p(-x_i)) \leq \frac{\|x\|}{\varepsilon} < +\infty\,.$$

Clearly $\|\cdot\|_*$ is a norm on X. It is easy to see that

$$\|x\|_* = \max(p_*(x),\ p_*(-x)) \qquad x \in X\,. \tag{26}$$

Due to (23), we have

$$B_i(x,r) := \{y \in X\colon \|y - x\|_i \leq r\} = \{y \in X\colon x + r\mathbf{1} \geq_{K_i} y \geq_{K_i} x - r\mathbf{1}\}\,, \tag{27}$$

where $x \in X$, $i \in I$ and $r > 0$. Let $x \in X$ and $r > 0$. Consider the closed ball $B(x,r)$ with center x and radius r with respect to $\|\cdot\|_*$:

$$B(x,r) := \{y \in X\colon \|y - x\|_* \leq r\} = \{y \in X\colon x + r\mathbf{1} \geq_{K_*} y \geq_{K_*} x - r\mathbf{1}\}\,. \tag{28}$$

It follows from (20), (27), and (28) that

$$B(x,r) = \bigcap_{i\in I} B_i(x,r)\,, \tag{29}$$

and

$$B(x,r) \subseteq \{y \in X\colon x + r\mathbf{1} \geq_K y \geq_K x - r\mathbf{1}\}\,. \tag{30}$$

We now present an example.

Example 8 Let $X = \mathbb{R}^2$. Consider the cones

$$A = \{(x,y) \in X\colon x \geq 0 \text{ and } y \geq 2x\},$$
$$B = \left\{(x,y) \in X\colon x \leq 0 \text{ and } y \geq \frac{1}{2}x\right\},$$
$$C = \left\{(x,y) \in X\colon x \geq 0 \text{ and } y \geq -\frac{1}{2}x\right\},$$
$$D = \{(x,y) \in X\colon x \leq 0 \text{ and } y \geq -2x\}\,.$$

Set $K_1 = A \cup B$, $K_2 = C \cup D$, $K = K_1 \cup K_2$, and $K_* := K_1 \cap K_2 = A \cup D$. It is easy to check that K is not a convex set while K_1, K_2 and K_* are convex sets. We also have:

$$p_*(\mathrm{x}) = \max(y-2x, y+2x) \text{ for all } \mathrm{x} = (x,y) \in X\,,$$
$$\|\mathrm{x}\|_* = |y| + 2|x| \quad \text{for all } \mathrm{x} = (x,y) \in X\,.$$

Example 9 Let X be a normed space with a norm $\|\cdot\|$. Let $Y = X \times \mathbb{R}$ and $K := \operatorname{epi}\|\cdot\| \subset Y$ be the epigraph of $\|\cdot\|$. (Recall that $\operatorname{epi}\|\cdot\| = \{(x,\lambda) \in Y\colon \lambda \geq \|x\|\}$.) Then K is a convex closed cone and $(0,1) \in \operatorname{int} K$. Assume now that X is equipped with two equivalent norms $\|\cdot\|_1$ and $\|\cdot\|_2$. Let $K_i = \operatorname{epi}\|\cdot\|_i$, $i = 1,2$, and $K = K_1 \cup K_2$. If there exist $x' \in X$ and $x'' \in X$ such that $\|x'\|_1 < \|x'\|_2$ and $\|x''\|_1 > \|x''\|_2$, then K is not convex. Clearly K is a pointed cone. The set $\operatorname{int} K$ contains $(0,1)$; hence it is nonempty. Clearly $K \setminus \{0\}$ is contained in the open half-space $\{(x,\lambda)\colon \lambda > 0\}$. Cone K is star-shaped. It can be proved that $\ker K = K_1 \cap K_2$.

In the remainder of the article, we consider a normed space X with a closed star-shaped cone K such that $\operatorname{int}\ker K$ is not empty. Assume that K is given as $K = \bigcup_{i\in I} K_i$, where

- I is an arbitrary index set;
- K_i, $(i \in I)$ is a closed solid convex pointed cone;
- The interior $\operatorname{int} K_*$ of the cone $K_* = \bigcap_{i\in I} K_i$ is nonempty.

In the sequel, assume that the norm $\|\cdot\|$ of X coincides with the norm $\|\cdot\|_*$ defined by (26).

Characterization of Best Approximations

Let $\varphi\colon X \times X \longrightarrow \mathbb{R}$ be a function defined by

$$\varphi(x,y) := \sup\{\lambda \in \mathbb{R}\colon x + y \geq_K \lambda\mathbf{1}\} \quad (x,\ y \in X)\,. \tag{31}$$

Since $\mathbf{1} \in \operatorname{int} K_*$, it follows that the set $\{\lambda \in \mathbb{R}\colon x + y \geq_K \lambda\mathbf{1}\}$ is nonempty and bounded from above (by the number $\|x+y\|_*$). Clearly this set is closed. It follows from the definition of φ that the function φ has the following properties:

$$-\infty < \varphi(x,y) \leq \|x+y\|_* \quad \text{for each} \quad x,\ y \in X\,, \tag{32}$$

$$x + y \geq_K \varphi(x,y)\mathbf{1} \quad \text{for all} \quad x,\ y \in X\,, \tag{33}$$

$$\varphi(x,y) = \varphi(y,x) \quad \text{for all} \quad x,\ y \in X\,, \tag{34}$$

$$\begin{aligned}\varphi(x,-x) &= \sup\{\lambda \in \mathbb{R}\colon 0 = x - x \geq_K \lambda\mathbf{1}\}\\ &= 0 \quad \text{for all} \quad x \in X\,,\end{aligned} \tag{35}$$

$$\varphi(x, y+\lambda\mathbf{1}) = \varphi(x,y) + \lambda \quad \text{for all} \quad x,\ y \in X \text{ and } \lambda \in \mathbb{R}\,, \tag{36}$$

$$\varphi(x+\lambda\mathbf{1}, y) = \varphi(x,y) + \lambda \quad \text{for all} \quad x,\ y \in X \text{ and } \lambda \in \mathbb{R}\,, \tag{37}$$

$$\varphi(\gamma x, \gamma y) = \gamma\varphi(x,y) \quad \text{for all} \quad x,\ y \in X \text{ and } \gamma > 0\,. \tag{38}$$

Proposition 14 *Let φ be the function defined by (31). Then*

$$\varphi(x,y) = -p(-x-y), \quad (x,\ y \in X)\,, \tag{39}$$

and hence

$$\varphi(x,y) = \sup_{i\in I}[-p_i(-x-y)] \quad (x,\ y \in X)\,. \tag{40}$$

Proof For each $x,\ y \in X$, we have

$$\begin{aligned}-\varphi(-x,-y) &= -\sup\{\lambda \in \mathbb{R}\colon -(x+y) \geq_K \lambda\mathbf{1}\}\\ &= \inf\{-\lambda \in \mathbb{R}\colon -(x+y) \geq_K \lambda\mathbf{1}\}\\ &= \inf\{\lambda' \in \mathbb{R}\colon -(x+y) \geq_K -\lambda'\mathbf{1}\}\\ &= \inf\{\lambda' \in \mathbb{R}\colon \lambda'\mathbf{1} \geq_K x+y\}\\ &= p(x+y)\,.\end{aligned}$$

Hence $\varphi(x, y) = -p(-x - y)$. In view of (21), we get (40). □

Now, consider $x, y \in X$. We define the functions $\varphi_x : X \longrightarrow \mathbb{R}$ and $\varphi_y : X \longrightarrow \mathbb{R}$ by

$$\varphi_x(t) = \varphi(x, t) \quad t \in X \tag{41}$$

and

$$\varphi_y(t) = \varphi(t, y) \quad t \in X\,. \tag{42}$$

Note that φ_x and φ_y are nonincreasing functions with respect to the relation generated by K on X. We have the following result:

Corollary 5 *Let φ be the function defined by (31). Then φ is Lipschitz continuous.*

Proof This is an immediate consequence of Lipschitz continuity of p and Proposition 14. □

Corollary 6 *For each $x, y \in X$, the functions defined by (41) and (42) are Lipschitz continuous.*

Proof It follows from Corollary 5. □

Proposition 15 *Let φ be the function defined by (31) and set*

$$\Lambda(y, \alpha) = \{x \in X : \varphi(x, y) \geq \alpha\} \quad (y \in X;\ \alpha \in \mathbb{R})\,.$$

Then, $\Lambda(y, \alpha) = K + \alpha\mathbf{1} - y$ for all $y \in X$ and all $\alpha \in \mathbb{R}$. □

Proof Fix $y \in X$ and $\alpha \in \mathbb{R}$. Then

$$x \in \Lambda(y, \alpha) \Longleftrightarrow \varphi(x, y) \geq \alpha\,.$$

Due to Proposition 14, this happens if and only if $-p(-x - y) \geq \alpha$, and hence by Proposition 12, if and only if $-x - y \in -\alpha\mathbf{1} - K$. This is equivalent to $x \in K + \alpha\mathbf{1} - y$, which completes the proof. □

Corollary 7 *Under the hypotheses of Proposition 15, we have*

$$\varphi(x, y) \geq \alpha \quad \text{if and only if} \quad x + y \geq_K \alpha\mathbf{1} \qquad (x,\ y \in X;\ \alpha \in \mathbb{R})\,.$$

Lemma 2 *Let W be a closed downward subset of X, $y_0 \in \operatorname{bd} W$ and φ be the function defined by (31). Then*

$$\varphi(w, -y_0) \leq 0 = \varphi(y_0, -y_0) \qquad \forall\, w \in W\,. \tag{43}$$

Proof The proof is similar to the proof of Lemma 4.3 in [7]. □

For $x \in X$ and a nonempty subset W of X, we will use the following notations:

$$d^i(x, W) := \inf_{w \in W} \|x - w\|_i \quad i \in I$$

and

$$P^i_W(x) = \{w \in W : \|x - w\|_i = d^i(x, W)\} \quad i \in I\,.$$

Lemma 3 *Let W be a closed downward subset of X, $x \in X \setminus W$, $r > 0$, and $i \in I$. Then $r = d^i(x, W)$ if and only if $x - r\mathbf{1} \in W$ and $p_i(x - w - r\mathbf{1})) \geq 0$ for all $w \in W$.*

Proof Let $r = d^i(x, W)$. In a manner analogous to the proof of Proposition 3, one can prove that $x - r\mathbf{1} \in P^i_W(x) \subset W$. Since $P^i_W(x) \subseteq \operatorname{bd} W$, it follows from Lemma 2 and Proposition 14 that $p_i(x - w - r\mathbf{1}) \geq 0$ for all $w \in W$. Conversely, suppose that $x - r\mathbf{1} \in W$ and $p_i(x - w - r\mathbf{1}) \geq 0$ for all $w \in W$. Let $w \in W$ be arbitrary. Since p_i is plus-homogeneous and $p_i(x - w - r\mathbf{1}) = p_i(x - w) - r$, it follows from (24) that

$$\|x - w\|_i \geq p_i(x - w) \geq r\,.$$

Since $\|x - (x - r\mathbf{1})\|_i = r$ and $x - r\mathbf{1} \in W$, we conclude that $r = d^i(x, W)$. □

Lemma 4 *Let W be a closed downward subset of X, $x \in X \setminus W$, and $r > 0$. Then $r = d(x, W)$ if and only if $x - r\mathbf{1} \in W$ and for some $i \in I$, $p_i(x - w - r\mathbf{1}) \geq 0$ for all $w \in W$.*

Proof Let $r = d(x, W)$. By Proposition 3 we have $x - r\mathbf{1} \in P_W(x) \subseteq \operatorname{bd} W$. Then it follows from Lemma 3 that $\varphi(w, r\mathbf{1} - x) \leq 0$ for all $w \in W$. In view of (40), we get $p_i(x - w - r\mathbf{1}) \geq 0$ for all $w \in W$ and all $i \in I$. Conversely, suppose that $x - r\mathbf{1} \in W$ and for some $i \in I$, $p_i(x - w - r\mathbf{1}) \geq 0$ for all $w \in W$. Consider $w \in W$. Since p_i is plus-homogeneous and $p_i(x - w - r\mathbf{1}) = p_i(x - w) - r$, it follows from (24) and (25) that

$$\|x - w\|_* \geq \|x - w\|_i \geq p_i(x - w) \geq r\,.$$

Since $r = \|x - (x - r\mathbf{1})\|_*$ and $x - r\mathbf{1} \in W$, one thus has $r = d(x, W)$. □

The following result is an immediate consequence of Lemmas 3 and 4.

Corollary 8 *Let W be a closed downward subset of X, $x \in X \setminus W$. Then*

$$d(x, W) = d^i(x, W) \quad \text{for all} \quad i \in I\,. \tag{44}$$

Corollary 9 *Let W be a closed downward subset of X, $x \in X \setminus W$, and $w_0 \in W$. Then, $w_0 \in P_W(x)$ if and only if $w_0 \in P^i_W(x)$ for each $i \in I$.*

Proof Let $w_0 \in P_W(x)$. Then $\|x - w_0\|_* = d(x, W)$. In view of (25) and (44), we have $\|x - w_0\|_i = d^i(x, W)$ for each $i \in I$. Therefore, $w_0 \in P^i_W(x)$ for each $i \in I$. Conversely, let $w_0 \in P^i_W(x)$ for each $i \in I$. Then $\|x - w_0\|_i = d^i(x, W)$ for each $i \in I$. Hence, by (44), we get $\|x - w_0\|_* = \max_{i\in I} \|x - w_0\|_i = d(x, W)$, that is, $w_0 \in P_W(x)$. □

Theorem 6 *Let W be a closed downward subset of X, $x_0 \in X \setminus W$, $y_0 \in W$, and $r_0 := \|x_0 - y_0\|_*$. Assume that φ is the function defined by (31). Then the following assertions are equivalent:*
(1) $y_0 \in P_W(x_0)$.
(2) There exists $l \in X$ such that

$$\varphi(w, l) \le 0 \le \varphi(y, l), \quad \forall\ w \in W,\ y \in B(x_0, r_0)\,. \tag{45}$$

Moreover, if (45) holds with $l = -y_0$, then $y_0 = w_0 = \min P_W(x_0)$, where $w_0 = x_0 - r\mathbf{1}$ is the least element of the set $P_W(x_0)$ and $r := d(x_0, W)$.

Proof
(1) $\Longrightarrow$ (2). Suppose that $y_0 \in P_W(x_0)$. Then $r_0 = \|x_0 - y_0\|_* = d(x_0, W) = r$. Since W is a closed downward subset of X, it follows from Proposition 3 that the least element $w_0 = x_0 - r_0\mathbf{1}$ of the set $P_W(x_0)$ exists. Let $l = -w_0$ and $y \in B(x_0, r_0)$ be arbitrary. Then, by (30), we have $y \ge_K -l$ or $y + l \ge_K 0$. It follows from Corollary 7 that $\varphi(y, l) \ge 0$. On the other hand, since $w_0 \in P_W(x_0)$, it follows that $w_0 \in \operatorname{bd} W$. Hence, by Lemma 2 we have $\varphi(w, l) \le 0$ for all $w \in W$.
(2) $\Longrightarrow$ (1). Assume that (2) holds. By (28) it is clear that $x_0 - r_0\mathbf{1} \in B(x_0, r_0)$. Therefore, by (45) we have $\varphi(x_0 - r_0\mathbf{1}, l) \ge 0$. Due to Corollary 7, we get $x_0 - r_0\mathbf{1} + l \ge_K 0$, and so $l - r_0\mathbf{1} \ge_K -x_0$. Hence there exists $j \in I$ such that

$$l - r_0\mathbf{1} \ge_{K_j} -x_0\,. \tag{46}$$

Now, let $w \in W$ be arbitrary. Since p_j is topical and (21), (39), and (45) hold, it follows from (46) that

$$\begin{aligned} p_j(x_0 - w) \ge p_j(r_0\mathbf{1} - l - w) &= p_j(-l - w) + r_0 \\ &\ge p(-l - w) + r_0 \\ &= -\varphi(w, l) + r_0 \\ &\ge 0 + r_0 = r_0\,. \end{aligned}$$

Then, by (24) and (25), we have

$$\begin{aligned} r_0 \le p_j(x_0 - w) &\le \|x_0 - w\|_j \\ &\le \|x_0 - w\|_* \text{ for all } w \in W\,. \end{aligned}$$

Thus $\|x_0 - y_0\|_* = d(x_0, W)$. Consequently, $y_0 \in P_W(x_0)$. Finally, suppose that (45) holds with $l = -y_0$. Then, by the implication (2) $\Longrightarrow$ (1), we have $y_0 \in P_W(x_0)$, and so $r_0 = \|x_0 - y_0\|_* = d(x_0, W)$ and $y_0 \ge_K w_0$, where $w_0 = x_0 - r\mathbf{1}$ is the least element of the set $P_W(x_0)$ and $r := d(x_0, W)$. Now, let $w \in P_W(x_0)$ be arbitrary. Then $\|x_0 - w\|_* = d(x_0, W) = r_0$, that is, $w \in B(x_0, r_0)$. It follows from (45) that $\varphi(w, -y_0) \ge 0$. In view of Corollary 7, we have $w - y_0 \ge_K 0$, and so $w \ge_K y_0$. This means that $y_0 = \min P_W(x_0) = w_0$. This completes the proof. □

Strictly Downward Sets and Their Best Approximation Properties

We start with the following definitions, which were introduced in [7] for downward subsets of a Banach lattice.

Definition 5 A downward subset W of X is called strictly downward if for each boundary point w_0 of W the inequality $w >_K w_0$ implies $w \notin W$.

Definition 6 Let W be a downward subset of X. We say that W is strictly downward at a point $w' \in \operatorname{bd} W$ if for all $w_0 \in \operatorname{bd} W$ with $w' \ge_K w_0$ the inequality $w >_K w_0$ implies $w \notin W$.

The following lemmas have been proved in [7]; however, those proofs hold for the case under consideration.

Lemma 5 *Let $f: X \longrightarrow \mathbb{R}$ be a continuous strictly increasing function. Then all nonempty level sets $S_c(f)(c \in \mathbb{R})$ of f are strictly downward.*

Lemma 6 *Let W be a closed downward subset of X. Then W is strictly downward at $w' \in \operatorname{bd} W$ if and only if*
(i) $w >_K w' \Longrightarrow w \notin W$;
(ii) $(w' \geq_K w_0, w_0 \in \operatorname{bd} W) \Longrightarrow w_0 = w'$.

Lemma 7 *Let W be a closed downward subset of X. Then W is strictly downward if and only if W is strictly downward at each of its boundary points.*

Lemma 8 *Let φ be the function defined by (31) and W be a closed downward subset of X that is strictly downward at a point $w' \in \operatorname{bd} W$. Then there exists unique $l \in X$ such that*

$$\varphi(w, l) \leq 0 = \varphi(w', l), \quad \forall\ w \in W\,.$$

Theorem 7 *Let φ be the function defined by (31). Then for a closed downward subset W of X the following assertions are equivalent:*
(1) W is strictly downward.
(2) For each $w_0 \in \operatorname{bd} W$ there exists unique $l \in X$ such that

$$\varphi(w, l) \leq 0 = \varphi(w_0, l) \quad \forall\ w \in W\,.$$

Proof The implication (1) $\Longrightarrow$ (2) follows from Lemma 8. We now prove the implication (2) $\Longrightarrow$ (1). Assume that for each $w_0 \in \operatorname{bd} W$ there exists unique $l \in X$ such that

$$\varphi(w, l) \leq 0 = \varphi(w_0, l) \quad \forall\ w \in W\,.$$

Let $w_0 \in \operatorname{bd} W$ and $y \in X$ with $y >_K w_0$. Assume that $y \in W$. We claim that $y + \lambda \mathbf{1} \notin W$ for all $\lambda > 0$. Suppose that there exists $\lambda_0 > 0$ such that $y + \lambda_0 \mathbf{1} \in W$. Since $y + \lambda_0 \mathbf{1} >_K w_0 + \lambda_0 \mathbf{1}$ and W is a downward set, we have $w_0 + \lambda_0 \mathbf{1} \in W$. In view of Corollary 1, it contradicts with $w_0 \in \operatorname{bd} W$, and so the claim is true. Then, by Corollary 1, we have $y \in \operatorname{bd} W$. Let $l = -y$. It follows from Lemma 2 that

$$\varphi(w, l) \leq 0 = \varphi(y, l) \quad \forall\ w \in W\,. \tag{47}$$

On the other hand, applying Lemma 2 to the point w_0 we have for $l' = -w_0$:

$$\varphi(w, l') \leq 0 = \varphi(w_0, l') \quad \forall\ w \in W\,. \tag{48}$$

Since $y >_{K_i} w_0$ for some $i \in I$ and p_i is increasing, it follows from (21), (39), and(48) that $0 = p_i(-w_0 - l') \geq p_i(-y - l') \geq p(-y - l') = -\varphi(y, l') \geq 0$. This, together with (48), implies that

$$\varphi(w, l') \leq 0 = \varphi(y, l') \quad \forall\ w \in W\,. \tag{49}$$

Since $w_0 \neq y$, it follows that $l' \neq l$. Hence (47) and (49) contradict the uniqueness of l. We have demonstrated that the assumption $y \in W$ leads to a contradiction. Thus $y \notin W$. This means that W is strictly downward. □

Corollary 10 *Let $f: X \longrightarrow \mathbb{R}$ be a continuous strictly increasing function and φ be the function defined by (31). Then for each $x \in X$ there exists unique $l = -x$ such that*

$$\varphi(w, l) \leq 0 = \varphi(x, l) \quad \forall\ w \in S_c(f)\,,$$

where $c = f(x)$.

Proof This is an immediate consequence of Lemma 5 and Theorem 7. □

Definition 7 Let W be a downward subset of X. A point $w' \in \operatorname{bd} W$ is said to be a Chebyshev point if for each $w_0 \in \operatorname{bd} W$ with $w' \geq_K w_0$ and for each $x_0 \notin W$ such that $w_0 \in P_W(x_0)$ it follows that $P_W(x_0) = \{w_0\}$, that is, the best approximation of x_0 is unique.

Definition 7 was introduced in [7] for a downward subset of a Banach lattice.

Definition 8 Let W be a downward subset of X. A point $w' \in \operatorname{bd} W$ is said to be a Chebyshev point of W with respect to each K_i $(i \in I)$ if for each $w_0 \in \operatorname{bd} W$ with $w' \geq_K w_0$ and for each $x_0 \notin W$ such that $w_0 \in P^i{}_W(x_0)$ for each $i \in I$ it follows that $P^i{}_W(x_0) = \{w_0\}$ for each $i \in I$.

Remark 4 In view of Corollary 8, we have that Definitions 7 and 8 are equivalent.

Theorem 8 *Let W be a closed downward subset of X and $w' \in \operatorname{bd} W$. If w' is a Chebyshev point of W with respect to each K_i $(i \in I)$, then W is a strictly downward set at w'.*

Proof Suppose that w' is a Chebyshev point of W with respect to each K_i $(i \in I)$. Assume, if possible, that W is not strictly downward at w'. Then we can find $w_0 \in \mathrm{bd}\, W$ and $w \in W$ such that $w' \geq_K w_0$ and $w >_K w_0$. Let $r \geq \|w - w_0\|_* > 0$. It follows from (27) that

$$r\mathbf{1} \geq_{K_i} w - w_0 \qquad \forall\ i \in I\,.$$

Thus, $w_0 + r\mathbf{1} \geq_{K_i} w$ for all $i \in I$. Set $x_0 = w_0 + r\mathbf{1} \in X$. Since $w_0 \in \mathrm{bd}\, W$, by Lemma 6 we have $\varphi(y, -w_0) \leq 0$ for all $y \in W$. Also, $x_0 - r\mathbf{1} = w_0 \in W$. Thus, by (21), Proposition 14, and Lemma 4 we get $r = d(x_0, W)$. Since $\|x_0 - w_0\|_i = \|r\mathbf{1}\|_i = r$ for all $i \in I$, it follows from (25) that $\|x_0 - w_0\|_* = r$, and hence $w_0 \in P_W(x_0)$. In view of Corollary 9, we obtain $w_0 \in P^i_W(x_0)$ for all $i \in I$.

On the other hand, we have $x_0 = w_0 + r\mathbf{1} \geq_{K_i} w$ for all $i \in I$. Since $w >_K w_0$, we conclude that there exists $j \in I$ such that $w >_{K_j} w_0$. It follows that $r\mathbf{1} = x_0 - w_0 >_{K_j} x_0 - w \geq_{K_j} 0$. Hence

$$\|x_0 - w\|_j \leq \|r\mathbf{1}\|_j = r = d^j(x_0, W) \leq \|x_0 - w\|_j\,.$$

Thus $\|x_0 - w\|_j = d^j(x_0, W)$, and so $w \in P^j{}_W(x_0)$ with $w \neq w_0$. Whence there exist a point $x_0 \in X \setminus W$ and a point $w_0 \in \mathrm{bd}\, W$ with $w' \geq_K w_0$ such that $w_0 \in P^i{}_W(x_0)$ for each $i \in I$ and $P^j{}_W(x_0)$ contains at least one point different from w_0. This is a contradiction because w' is a Chebyshev point of W with respect to each K_i $(i \in I)$, which completes the proof. □

Proposition 16 *Let W be a closed downward subset of X and $w' \in \mathrm{bd}\, W$. If W is a strictly downward set at w', then w' is a Chebyshev point of W.*

Proof The proof is similar to that of Theorem 4.2 (the implication (2) $\Longrightarrow$ (1)) in [7]. □

Corollary 11 *Let $f\colon X \longrightarrow \mathbb{R}$ be a continuous strictly increasing function. Then $S_c(f)(c \in \mathbb{R})$ is a Chebyshev subset of X.*

Proof This is an immediate consequence of Lemma 5 and Proposition 16. □

References

1. Chui CK, Deutsch F, Ward JD (1990) Constrained best approximation in Hilbert space. Constr Approx 6:35–64
2. Chui CK, Deutsch F, Ward JD (1992) Constrained best approximation in Hilbert space II. J Approx Theory 71:213–238
3. Deutch F (2000) Best approximation in inner product spaces. Springer, New York
4. Deutsch F, Li W, Ward JD (1997) A dual approach to constrained interpolation from a convex subset of a Hilbert space. J Approx Theory 90:385–414
5. Deutsch F, Li W, Ward JD (2000) Best approximation from the intersection of a closed convex set and a polyhedron in Hilbert space, weak Slater conditions, and the strong conical hull intersection property. SIAM J Optim 10: 252–268
6. Martinez-Legaz J-E, Rubinov AM, Singer I (2002) Downward sets and their separation and approximation properties. J Global Optim 23:111–137
7. Mohebi H, Rubinov AM (2006) Best approximation by downward sets with applications. J Anal Theory Appl 22(1):1–22
8. Mohebi H, Rubinov AM (2006) Metric projection onto a closed set: necessary and sufficent conditions for the global minimum. J Math Oper Res 31(1):124–132
9. Mohebi H, Sadeghi H, Rubinov AM (2006) Best approximation in a class of normed spaces with star-shaped cones. J Numer Funct Anal Optim 27(3–4):411–436
10. Mulansky B, Neamtu M (1998) Interpolation and approximation from convex sets. J Approx Theory 92:82–100
11. Rubinov AM (2000) Abstract convex analysis and global optimization. Kluwer, Boston Dordrecht London
12. Rubinov AM, Gasimov RN (2004) Scalarization and nonlinear scalar duality for vector optimization with preferences that are not necessarily a pre-order relation. J Glob Optim 29:455–477
13. Rubinov AM, Singer I (2001) Topical and sub-topical functions, downward sets and abstract convexity. Optimization 50:307–351
14. Rubinov AM, Singer I (2000) Best approximation by normal and co-normal sets. J Approx Theory 107:212–243
15. Singer I (1997) Abstract convex analysis. Wiley-Interscience, New York
16. Singer I (1970) Best approximation in normed linear spaces by elements of linear subspaces. Springer, New York
17. Jeyakumar V, Mohebi H (2005) A global approach to nonlinearly constrained best approximation. J Numer Funct Anal Optim 26(2):205–227
18. Jeyakumar V, Mohebi H (2005) Limiting and ε-subgradient characterizations of constrained best approximation. J Approx Theory 135:145–159
19. Vlasov LP (1967) Chebyshev sets and approximatively convex sets. Math Notes 2:600–605
20. Vlasov LP (1973) Approximative properties of sets in normed linear spaces. Russ Math Surv 28:1–66
21. Vulikh BZ (1967) Introduction to the theory of partially ordered vector spaces. Wolters-Noordhoff, Groningen

Bilevel Fractional Programming

HERMINIA I. CALVETE, CARMEN GALÉ
Dpto. de Métodos Estadísticos,
Universidad de Zaragoza, Zaragoza, Spain

MSC2000: 90C32, 90C26

Article Outline

Keywords
Introduction
Formulation
Theoretical Results
Algorithms
References

Keywords

Fractional bilevel programming; Hierarchical optimization; Nonconvex optimization

Introduction

Fractional bilevel programming (FBP), a class of bilevel programming [6,10], has been proposed as a generalization of standard fractional programming [9] for dealing with hierarchical systems with two decision levels. FBP problems assume that the objective functions of both levels are ratios of functions and the common constraint region to both levels is a nonempty and compact polyhedron.

Formulation

Using the common notation in bilevel programming, the FBP problem [1] can be formulated as:

$$\begin{aligned} \min_{x_1,x_2} \quad & f_1(x_1,x_2) = \frac{h_1(x_1,x_2)}{g_1(x_1,x_2)}, \\ & \text{where } x_2 \text{ solves} \\ \min_{x_2} \quad & f_2(x_1,x_2) = \frac{h_2(x_1,x_2)}{g_2(x_1,x_2)} \\ \text{s.t.} \quad & (x_1,x_2) \in S\ , \end{aligned}$$

where $x_1 \in \mathbb{R}^{n_1}$ and $x_2 \in \mathbb{R}^{n_2}$ are the variables controlled by the upper level and the lower level decision maker, respectively; h_i and g_i are continuous functions, h_i are nonnegative and concave and g_i are positive and convex on S; and $S = \{(x_1,x_2) : A_1x_1 + A_2x_2 \le b, x_1 \ge 0, x_2 \ge 0\}$, which is assumed to be nonempty and bounded.

Let S_1 be the projection of S on $\mathbb{R}^{n_1}$. For each $\tilde{x}_1 \in S_1$ provided by the upper level decision maker, the lower level one solves the fractional problem:

$$\begin{aligned} \min_{x_2} \quad & f_2(\tilde{x}_1,x_2) = \frac{h_2(\tilde{x}_1,x_2)}{g_2(\tilde{x}_1,x_2)} \\ \text{s.t.} \quad & A_2x_2 \le b - A_1\tilde{x}_1 \\ & x_2 \ge 0\ . \end{aligned}$$

Let $M(\tilde{x}_1)$ denote the set of optimal solutions to this problem. In order to ensure that the FBP problem is well posed it is also assumed that $M(\tilde{x}_1)$ is a singleton for all $\tilde{x}_1 \in S_1$.

The feasible region of the upper level decision maker, also called the inducible region (IR), is implicitly defined by the lower level decision maker:

$$\begin{aligned} \text{IR} = \{(\tilde{x}_1,\tilde{x}_2) : \tilde{x}_1 \ge 0,\ \tilde{x}_2 = \operatorname{argmin} \\ \{f_2(\tilde{x}_1,x_2) : A_1\tilde{x}_1 + A_2x_2 \le b, x_2 \ge 0\}\}\ . \end{aligned}$$

Therefore, the FBP problem can be stated as:

$$\begin{aligned} \min_{x_1,x_2} \quad & f_1(x_1,x_2) = \frac{h_1(x_1,x_2)}{g_1(x_1,x_2)} \\ \text{s.t.} \quad & (x_1,x_2) \in \text{IR}\ . \end{aligned}$$

Theoretical Results

The FBP problem is a nonconvex optimization problem but, taking into account the quasiconcavity of f_2 and the properties of polyhedra, in [1] it was proved that the inducible region is formed by the connected union of faces of the polyhedron S.

One of the main features of FBP problems is that, even with the more complex objective functions, they retain the most important property related to the optimal solution of linear bilevel programming problems. That is, there is an extreme point of S which solves the FBP problem [1]. This result is a consequence of the properties of IR as well as of the fact of f_1 being quasiconcave. The same conclusion is also obtained when both level objective functions are defined as the minimum of a finite number of functions which are ratios with the previously stated conditions or, in general, if they are quasiconcave.

Under the additional assumption that the upper level objective function is explicitly quasimonotonic, another geometrical property of the optimal solution of the FBP problem can be obtained by introducing the concept of boundary feasible extreme point. According to [7], a point $(x_1, x_2) \in \mathrm{IR}$ is a boundary feasible extreme point if there exists an edge E of S such that (x_1, x_2) is an extreme point of E, and the other extreme point of E is not an element of IR.

Let us consider the relaxed problem:

$$\begin{aligned} \min_{x_1,x_2} \quad & f_1(x_1, x_2) = \frac{h_1(x_1, x_2)}{g_1(x_1, x_2)}, \\ \text{s.t.} \quad & (x_1, x_2) \in S\ . \end{aligned} \tag{1}$$

Since f_1 is a quasiconcave function and S is a nonempty and compact polyhedron, an extreme point of S exists which solves (1). Obviously, if an optimal solution of (1) is a point of IR, then it is an optimal solution to the FBP problem. However, in general, this will not be true, since both decision makers usually have conflicting objectives.

Hence, if f_1 is explicitly quasimonotonic and there exists an extreme point of S not in IR that is an optimal solution of the relaxed problem (1), then a boundary feasible extreme point exists that solves the FBP problem [3].

Although FBP problems retain some important properties of linear bilevel problems, it is worth pointing out at this time some differences related to the existence of multiple optima when solving the lower level problem for given $x_1 \in S_1$. Different approaches have been proposed in the literature to make sure that the bilevel problem is well posed [6]. The most common one is to assume that $M(x_1)$ is single-valued for all $x_1 \in S_1$. Other approaches give rules for selecting $x_2 \in M(x_1)$ in order to be able to evaluate the upper level objective function $f_1(x_1, x_2)$. The optimistic approach assumes that the upper level decision maker has the right to influence the lower level decision maker so that the latter selects x_2 to provide the best value of f_1. On the contrary, the pessimistic approach assumes that the lower level decision maker always selects x_2 which gives the worst value of f_1.

It is well-known [8] that, under the optimistic approach, at least one optimal solution of the linear bilevel problem is obtained at an extreme point of the polyhedron defined by the common constraints. However, in [3] an example of the FBP problem is proposed in which $M(\tilde{x}_1)$ is not single-valued for given $\tilde{x}_1 \in S_1$ and this assertion is not true. Firstly, IR no longer consists of the union of faces of the polyhedron S. Secondly, if the pessimistic approach is used, then an optimal solution to the example does not exist. Finally, if the optimistic approach is taken the optimal solution to the example is not an extreme point of the polyhedron S.

Algorithms

Bearing in mind that there is an extreme point of S which solves the FBP problem, an enumerative algorithm can be devised which examines the set of extreme points of S in order to identify the best one regarding f_1, which is a point of IR. The bottleneck of the algorithm would be the generally large number of extreme points of a polyhedron together with the process of checking if an extreme point of S is a point of IR or not.

In the particular case in which f_1 is linear and f_2 is linear fractional (LLFBP problem), in [2] an enumerative algorithm has been proposed which finds a global optimum in a finite number of stages by examining implicitly only bases of the matrix A_2. This algorithm connects the points of IR with the bases of A_2, by applying the parametric approach to solve the fractional problem of the lower level. One of the main advantages of the procedure is that only linear problems have to be solved.

When f_1 is linear fractional and f_2 is linear (LFLBP problem), the algorithm developed in [2] combines local search in order to find an extreme point of IR with a better value of f_1 than any of its adjacent extreme points in IR and a penalty method when looking for another point of IR from which a new local search can start.

The Kth-best algorithm has been proposed in [3] to globally solve the FBP problem when both objective functions are linear fractional (LFBP). It essentially asserts that the best (in terms of the upper level objective function) of the extreme points of S which is a point of IR is an optimal solution to the problem. Moreover, the search for this point can be made sequentially by computing adjacent extreme points to the incumbent extreme point.

Finally, recently two genetic algorithms have been proposed [4,5] which allow us to solve LLFBP, LFBP and LFLBP problems. Both algorithms provide excellent results in terms of both accuracy of the solution and time invested, proving that they are effective and useful approaches for solving those problems. Both algorithms associate chromosomes with extreme points of S. The fitness of a chromosome evaluates its quality and penalizes it if the associated extreme point is not in IR. The algorithms mainly differ in the procedure of checking if an extreme point is in IR. When f_2 is linear, all lower level problems have the same dual feasible region, so it is possible to prove several properties which simplify the process.

References

1. Calvete HI, Galé C (1998) On the quasiconcave bilevel programming problem. J Optim Appl 98(3):613–622
2. Calvete HI, Galé C (1999) The bilevel linear/linear fractional programming problem. Eur J Oper Res 114(1):188–197
3. Calvete HI, Galé C (2004) Solving linear fractional bilevel programs. Oper Res Lett 32(2):143–151
4. Calvete HI, Galé C, Mateo PM (2007) A genetic algorithm for solving linear fractional bilevel problems. To appear in Annals Oper Res
5. Calvete HI, Galé C, Mateo PM (2008) A new approach for solving linear bilevel problems using genetic algorithms. Eur J Oper Res 188(1):14–28
6. Dempe S (2003) Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. Optimization 52:333–359
7. Liu YH, Hart SM (1994) Characterizing an optimal solution to the linear bilevel programming problem. Eur J Oper Res 73(1):164–166
8. Savard G (1989) Contribution à la programmation mathématique à deux niveaux. PhD thesis, Ecole Polytechnique de Montréal, Université de Montréal, Montréal, Canada
9. Schaible S (1995) Fractional programming. In: Horst R, Pardalos PM (eds) Handbook of global optimization. Kluwer, Dordrecht, pp 495–608
10. Vicente LN, Calamai PH (1994) Bilevel and multilevel programming: a bibliography review. J Glob Optim 5:291–306

Bilevel Linear Programming

JONATHAN F. BARD
University Texas, Austin, USA

MSC2000: 49-01, 49K10, 49M37, 90-01, 91B52, 90C05, 90C27

Article Outline

Keywords
Definitions
Theoretical Properties
Algorithmic Approaches
See also
References

Keywords

Bilevel linear programming; Hierarchical optimization; Stackelberg game; Multiple objectives; Complementarity

Many hierarchical optimization problems involving two or more decision makers can be modeled as a multilevel mathematical program. The two-level structure is commonly known as a *Stackelberg game* where a leader and a follower try to minimize their individual objective functions $F(x, y)$ and $f(x, y)$, respectively, subject to a series of interdependent constraints [2,9]. Play is defined as sequential and the mood as noncooperative. The decision variables are partitioned between the players in such a way that neither can dominate the other. The leader goes first and through his choice of $x \in \mathbf{R}^n$ is able to influence but not control the actions of the follower. This is achieved by reducing the set of feasible choices available to the latter. Subsequently, the follower reacts to the leader's decision by choosing a $y \in \mathbf{R}^m$ in an effort to minimizes his costs. In so doing, he indirectly affects the leader's solution space and outcome.

Two basic assumptions underlying the Stackelberg game are that full information is available to the players and that cooperation is prohibited. This precludes the use of correlated strategies and side payments. The vast majority of research on this problem has centered on the linear case known as the linear *bilevel program* (BLP) [3,6]. Relevant notation, the basic model, and a discussion of its theoretical properties follow.

For $x \in X \subset \mathbf{R}^n$, $y \in Y \subset \mathbf{R}^m$, $F: X \times Y \to \mathbf{R}^1$, and $f: X \times Y \to \mathbf{R}^1$, the linear bilevel programming problem can be written as follows:

$$\min_{x \in X} \quad F(x, y) = c_1 x + d_1 y, \tag{1}$$

$$\text{s.t.} \quad A_1x + B_1y \le b_1, \tag{2}$$

$$\min_{y \in Y} \quad f(x, y) = c_2x + d_2y, \tag{3}$$

$$\text{s.t.} \quad A_2x + B_2y \le b_2, \tag{4}$$

where $c_1, c_2 \in \mathbf{R}^n$, $d_1, d_2 \in \mathbf{R}^m$, $b_1 \in \mathbf{R}^p$, $b_2 \in \mathbf{R}^q$, $A_1 \in \mathbf{R}^{p \times n}$, $B_1 \in \mathbf{R}^{p \times m}$, $A_2 \in \mathbf{R}^{q \times n}$, $B_2 \in \mathbf{R}^{q \times m}$. The sets X and Y place additional restrictions on the variables, such as upper and lower bounds or integrality requirements. Of course, once the leader selects an x, the first term in the follower's objective function becomes a constant and can be removed from the problem. In this case, we replace $f(x, y)$ with $f(y)$.

The sequential nature of the decisions in (1)–(4) implies that y can be viewed as a function of x; i. e., $y = y(x)$. For convenience, this dependence will not be written explicitly.

Definitions

a) Constraint region of the linear BLP:

$$S = \{(x, y): \quad x \in X, y \in Y, \\ A_1x + B_1y \le b_1, A_2x + B_2y \le b_2\}.$$

b) Feasible set for follower for each fixed $x \in X$:

$$S(x) = \{y \in Y: \ A_2x + B_2y \le b_2\}.$$

c) Projection of S onto the leader's decision space:

$$S(X) = \{x \in X: \quad \exists\, y \in Y, \\ A_1x + B_1y \le b_1, A_2x + B_2y \le b_2\}.$$

d) Follower's *rational reaction set* for $x \in S(X)$:

$$P(x) = \{y \in Y: \\ y \in \arg\min \{f(x, \widehat{y}): \ \widehat{y} \in S(x)\}\}.$$

e) *Inducible region*:

$$\text{IR} = \{(x, y): \ (x, y) \in S, y \in P(x)\}.$$

To ensure that (1)–(4) is well posed it is common to assume that S is nonempty and compact, and that for all decisions taken by the leader, the follower has some room to respond; i. e., $P(x) \neq \emptyset$. The rational reaction set $P(x)$ defines the response while the inducible region IR represents the set over which the leader may optimize. Thus in terms of the above notation, the BLP can be written as

$$\min \{F(x, y): \ (x, y) \in \text{IR}\}. \tag{5}$$

Even with the stated assumptions, problem (5) may not have a solution. In particular, if $P(x)$ is not single-valued for all permissible x, the leader may not achieve his minimum payoff over IR. To avoid this situation in the development of algorithms, it is usually assumed that $P(x)$ is a point-to-point map. Because a simple check is available to see whether the solution to (1)–(4) is unique (see [2]) this assumption does not appear to be unduly restrictive.

It should be mentioned that in practice the leader will incur some cost in determining the decision space $S(X)$ over which he may operate. For example, when BLP is used as a model for a decentralized firm with headquarters representing the leader and the divisions representing the follower, coordination of lower level activities by headquarters requires detailed knowledge of production capacities, technological capabilities, and routine operating procedures. Up-to-date information in these areas is not likely to be available to corporate planners without constant monitoring and oversight.

Theoretical Properties

The linear bilevel program was first shown to be *NP*-hard by R.G. Jeroslow [7] using satisfiability arguments common in computer science. The complexity of the problem is further elaborated in ▶ Bilevel linear programming: Complexity, equivalence to minmax, concave programs. Issues related to the geometry of the solution space are now discussed. The main result is that when the linear BLP is written as a standard mathematical program (5), the corresponding constraint set or inducible region is comprised of connected faces of S and that a solution occurs at a vertex (see [1] or [8] for the proofs). For ease of presentation, it will be assumed that $P(x)$ is single-valued and bounded, S is bounded and nonempty, and that $Y = \{y : y \ge 0\}$.

Theorem 1 *The inducible region can be written equivalently as a piecewise linear equality constraint comprised of supporting hyperplanes of S.*

A straightforward corollary of this theorem is that the linear BLP is equivalent to minimizing F over a feasible region comprised of a piecewise linear equality constraint. In general, because a linear function $F = c_1x + d_1y$ is being minimized over IR, and because F is bounded below on S by, say, $\min\{c_1x + d_1y: (x, y) \in \text{IR}\}$, it can also be concluded that the solution to the linear BLP occurs at a vertex of IR. An alternative proof of this result was given by W.F. Bialas and M.H. Karwan [4] who noted that (5) could be written equivalently as

$$\min\{c_1x + d_1y: \ (x, y) \in \text{co IR}\},$$

where co IR is the convex hull of the inducible region. Of course, co IR is not the same as IR, but the next theorem states their relationship with respect to BLP solutions.

Theorem 2 *The solution (x^*, y^*) of the linear BLP occurs at a vertex of S.*

In general, at the solution (x^*, y^*) the hyperplane $\{(x, y): c_1x + d_1y = c_1x^* + d_1y^*\}$ will not be a support of the set S. Furthermore, a by-product of the proof of Theorem 2 is that any vertex of IR is also a vertex of S, implying that IR consists of faces of S. Comparable results were derived by Bialas and Karwan who began by showing that any point in S that strictly contributes in any convex combination of points in S to form a point in IR must also be in IR. This leads to the fact that if x is an extreme point of IR, then it is an extreme point of S. A final observation about the solution of the linear BLP can be inferred from this last assertion. Because the inducible region is not in general convex, the set of optimal solutions to (1)–(4) when not single-valued is not necessarily convex.

In searching for a way to solve the linear BLP, it would be helpful to have an explicit representation of IR rather than the implicit representation given by Definition e). This can be achieved by replacing the follower's problem (3)-(4) with his Kuhn–Tucker conditions and appending the resultant system to the leader's problem. Letting $u \in \mathbf{R}^q$ and $v \in \mathbf{R}^m$ be the dual variables associated with constraints (4) and $y \geq 0$, respectively, leads to the proposition that a necessary condition for (x^*, y^*) to solve the linear BLP is that there exists (row) vectors u^* and v^* such that (x^*, y^*, u^*, v^*) solves:

$$\begin{aligned}
\min \quad & c_1x + d_1y, && (6)\\
\text{s.t.} \quad & A_1x + B_1y \leq b_1, && (7)\\
& uB_2 - v = -d_2, && (8)\\
& u(b_2 - A_2x - B_2y) + vy = 0, && (9)\\
& A_2x + B_2y \leq b_2, && (10)\\
& x \geq 0, \quad y \geq 0, \quad u \geq 0, \quad v \geq 0. && (11)
\end{aligned}$$

This formulation has played a key role in the development of algorithms. One advantage that it offers is that it allows for a more robust model to be solved without introducing any new computational difficulties. In particular, by replacing the follower's objective function (3) with a quadratic form

$$f(x, y) = c_2x + d_2y + x^\top Q_1y + \frac{1}{2}y^\top Q_2y, \qquad (12)$$

where Q_1 is an $n \times m$ matrix and Q_2 is an $m \times m$ symmetric positive semidefinite matrix, the only thing that changes in (6)–(11) is constraint (8). The new constraint remains linear but now includes all problem variables; i. e.,

$$x^\top Q_1 + y^\top Q_2 + uB_2 - v = -d_2. \qquad (13)$$

From a conceptual point of view, (6)–(11) is a standard mathematical program and should be relatively easy to solve because all but one constraint is linear. Nevertheless, virtually all commercial nonlinear codes find complementarity terms like (9) notoriously difficult to handle so some ingenuity is required to maintain feasibility and guarantee global optimality.

Algorithmic Approaches

There have been nearly two dozen algorithms proposed for solving the linear BLP since the field caught the attention of researchers in the mid-1970s. Many of these are of academic interest only because they are either impractical to implement or highly inefficient. In general, there are three different approaches to solving (1)–(4) that can be considered workable. The first makes use of Theorem 2 and involves some form of vertex enumeration in the context of the simplex method. W. Candler and R. Townsely [5] were the first to develop an algorithm that was globally optimal. Their scheme repeatedly solves two linear programs, one for the leader in

all of the x variables and a subset of the y variables associated with an optimal basis to the follower's problem, and the other for the follower with all the x variables fixed. In a systematic way they explore optimal bases of the follower's problem for x fixed and then return to the leader's problem with the corresponding basic y variables. By focusing on the reduced cost coefficients of the y variables not in an optimal basis of the follower's problem, they are able to provide a monotonic decrease in the number of follower bases that have to be examined. Bialas and Karwan [4] offered a different approach that systematically explores vertices beginning with the basis associated with the optimal solution to the linear program created by removing (3). This is known as the *high point problem*.

The second and most popular method for solving the linear BLP is known as the *Kuhn–Tucker approach* and concentrates on (6)–(11). The fundamental idea is to use a *branch and bound strategy* to deal with the complementarity constraint (9). Omitting or relaxing this constraint leaves a standard linear program which is easy to solve. The various methods proposed employ different techniques for assuring that complementarity is ultimately satisfied (e. g., see [3,6]).

The third method is based on some form of penalty approach. E. Aiyoshi and K. Shimizu (see [8, Chap. 15]) addressed the general BLP by first converting the follower's problem to an unconstrained mathematical program using a barrier method. The corresponding stationarity conditions are then appended to the leader's problem which is solved repeatedly for decreasing values of the barrier parameter. To guarantee convergence the follower's objective function must be strictly convex. This rules out the linear case, at least in theory. A different approach using an exterior penalty method was proposed by Shimizu and M. Lu [8] that simply requires convexity of all the functions to guarantee global convergence.

In the approach of D.J. White and G. Anandalingam [10], the gap between the primal and dual solutions of the follower's problem for x fixed is used as a penalty term in the leader's problem. Although this results in a nonlinear objective function, it can be decomposed to provide a set of linear programs conditioned on either the decision variables (x, y) or the dual variables (u, v) of the follower's problem. They show that an exact penalty function exists that yields the global solution. Related theory and algorithmic details are highlighted in [8, Chap. 16], along with presentations of several vertex enumeration and Kuhn–Tucker-based implementations.

See also

- ▶ Bilevel Fractional Programming
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Bilevel Programming
- ▶ Bilevel Programming: Applications
- ▶ Bilevel Programming: Applications in Engineering
- ▶ Bilevel Programming: Implicit Function Approach
- ▶ Bilevel Programming: Introduction, History and Overview
- ▶ Bilevel Programming in Management
- ▶ Bilevel Programming: Optimality Conditions and Duality
- ▶ Multilevel Methods for Optimal Design
- ▶ Multilevel Optimization in Mechanics
- ▶ Stochastic Bilevel Programs

References

1. Bard JF (1984) Optimality conditions for the bilevel programming problem. Naval Res Logist Quart 31:13–26
2. Bard JF, Falk JE (1982) An explicit solution to the multi-level programming problem. Comput Oper Res 9(1):77–100
3. Bard JF, Moore JT (1990) A branch and bound algorithm for the bilevel programming problem. SIAM J Sci Statist Comput 11(2):281–292
4. Bialas WF, Karwan MH (1984) Two-level linear programming. Managem Sci 30(8):1004–1020
5. Candler W, Townsely R (1982) A linear two-level programming problem. Comput Oper Res 9(1):59–76
6. Hansen P, Jaumard B, Savard G (1992) New branch-and-bound rules for linear bilevel programming. SIAM J Sci Statist Comput 13(1):1194–1217
7. Jeroslow RG (1985) The polynomial hierarchy and a simple model for competitive analysis. Math Program 32:146–164
8. Shimizu K, Ishizuka Y, Bard JF (1997) Nondifferentiable and two-level mathematical programming. Kluwer, Dordrecht
9. Simaan M (1977) Stackelberg optimization of two-level systems. IEEE Trans Syst, Man Cybern SMC-7(4):554–556
10. White DJ, Anandalingam G (1993) A penalty function for solving bi-level linear programs. J Global Optim 3:397–419

Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs

JONATHAN F. BARD
University Texas, Austin, USA

MSC2000: 49-01, 49K45, 49N10, 90-01, 91B52, 90C20, 90C27

Article Outline

Keywords
Related Optimization Problems
Complexity of the Linear BLPP Problem
See also
References

Keywords

Bilevel linear programming; Hierarchical optimization; Stackelberg game; Computational complexity; Concave programming; Minmax problem; Bilinear programming

A sequential optimization problem in which independent decision makers act in a noncooperative manner to minimize their individual costs, may be categorized as a *Stackelberg game*. The bilevel programming problem (BLPP) is a static, open loop version of this game where the leader controls the decision variables $x \in X \subseteq \mathbf{R}^n$, while the follower separately controls the decision variables $y \in Y \subseteq \mathbf{R}^m$ (e. g., see [3,9]).

In the model, it is common to assume that the leader goes first and chooses an x to minimize his objective function $F(x, y)$. The follower then reacts by selecting a y to minimize his individual objective function $f(x, y)$ without regard to the impact this choice has on the leader. Here, $F\colon X \times Y \to \mathbf{R}^1$ and $f\colon X \times Y \to \mathbf{R}^1$. The focus of this article is on the linear case introduced in ► Bilevel linear programming and given by:

$$\min_{x \in X} \quad F(x, y) = c_1 x + d_1 y, \tag{1}$$
$$\text{s.t.} \quad A_1 x + B_1 y \le b_1, \tag{2}$$
$$\min_{y \in Y} \quad f(x, y) = c_2 x + d_2 y, \tag{3}$$
$$\text{s.t.} \quad A_2 x + B_2 y \le b_2, \tag{4}$$

where $c_1, c_2 \in \mathbf{R}^n$, $d_1, d_2 \in \mathbf{R}^m$, $b_1 \in \mathbf{R}^p$, $b_2 \in \mathbf{R}^q$, $A_1 \in \mathbf{R}^{p \times n}$, $B_1 \in \mathbf{R}^{p \times m}$, $A_2 \in \mathbf{R}^{q \times n}$, $B_2 \in \mathbf{R}^{q \times m}$. The sets X and Y place additional restrictions on the variables, such as upper and lower bounds. Note that it is always possible to drop components separable in x from the follower's objective function (3).

Out of practical considerations, it is further supposed that the feasible region given by (2), (4), X and Y is nonempty and compact, and that for each decision taken by the leader, the follower has some room to respond. The rational reaction set, $P(x)$, defines these responses while the inducible region, IR, represents the set over which the leader may optimize. These terms are defined precisely in ► Bilevel linear programming. In the play, y is restricted to $P(x)$.

Given these assumptions, the BLPP may still not have a well-defined solution. In particular, difficulties may arise when $P(x)$ is multivalued and discontinuous. This is illustrated by way of example in [2,3].

Related Optimization Problems

The linear *minmax problem* (LMMP) is a special case of (1)–(4) obtained by omitting constraint (2) and setting $c_2 = -c_1$, $d_2 = -d_1$. It is often written compactly without the subscripts as

$$\min_{x \in X} \max_{y \in Y} \{cx + dy\colon\ Ax + By = b\} \tag{5}$$

or equivalently as

$$\min_{x \in X} \left(cx + \max_{y \in S(x)} dy \right), \tag{6}$$

where $S(x) = \{y \in Y : By \le b - Ax\}$. Several restrictive versions of (5) where, for example, X and Y are polyhedral sets and $Ax + By \le b$ is absent, as well as related optimality conditions are discussed in [8]. Although important in its own right, the LMMP plays a key role in determining the computational complexity of the linear BLPP. This is shown presently.

Consider now the inner maximization problem in (6) with $Y = \{y = 0\}$. Its dual is: $\min\{u^\top (b - Ax)\colon u \in U\}$, where u is a q-dimensional decision vector and $U = \{u : u^\top B \ge d, u \ge 0\}$. Note that the dual objective function is parameterized with respect to the vector x. Replacing the inner maximization problem with its dual leads to

a second representation of (5):

$$\min_{x\in X,u\in U}(cx-u^{\top}Ax+u^{\top}b), \tag{7}$$

which is known as a disjoint *bilinear programming problem*. The theoretical properties of (7) along with its relationship to other optimization problems are highlighted in [1].

A more general version of a bilinear programming problem can be obtained directly from the linear BLPP. To see this, it is necessary to examine the Kuhn–Tucker formulation of the latter given by (6)–(11) in ► Bilevel linear programming. Placing the complementarity constraint in the objective function as a penalty term gives the following bilinear programming problem:

$$\begin{cases}\min & c_1x+d_1y \\ & \quad +M[u^{\top}(b_2-A_2x-B_2y)+v^{\top}y], \\ \text{s.t.} & A_1x+B_1y\le b_1, \\ & u^{\top}B_2-v^{\top}=-d_2, \\ & A_2x+B_2y\le b_2, \\ & x\ge 0,\quad y\ge 0,\quad u\ge 0,\quad v\ge 0,\end{cases} \tag{8}$$

where M is a sufficiently large constant. In [10] it is shown that a finite M exists for the solution of (8) to be a solution of (1)–(4), and that (8) is a concave program; that is, its objective function is concave. This point is further elaborated in the next section.

Complexity of the Linear BLPP Problem

(1)–(4) can be classified as *NP*-hard which loosely means that no *polynomial time algorithm* exists for solving it unless $P = NP$. To substantiate this claim, it is necessary to demonstrate that through a polynomial transformation, some known *NP*-hard problem can be reduced to a linear BLPP. This will be done below constructively by showing that the problem of minimizing a strictly concave quadratic function over a polyhedron (see [5]) is equivalent to solving a linear minmax problem (cf. [4]). For an alternative proof based on satisfiability arguments from computer science see [7].

Theorem 1 *The linear minmax problem is NP-hard.*

To begin, let x be an n-dimensional vector of decision variables, and $c\in\mathbf{R}^n$, $b\in\mathbf{R}^q$, $A\in\mathbf{R}^{q\times n}$, $D\in\mathbf{R}^{n\times n}$ be constant arrays. For A of full row rank and D positive definite, it will be shown that the following minimization problem can be transformed into a LMMP:

$$\begin{cases}\theta^*=\min\limits_{x} & cx-\frac{1}{2}x^{\top}Dx, \\ \text{s.t.} & Ax\le b,\end{cases} \tag{9}$$

where it is assumed that the feasible region in (9) is bounded and contains all nonnegativity constraints on the variables. The core argument centers on the fact that the Kuhn–Tucker conditions associated with the concave program (9) must necessarily be satisfied at optimality. These conditions may be stated as follows:

$$Ax\le b, \tag{10}$$

$$x^{\top}D-u^{\top}A=c, \tag{11}$$

$$u^{\top}(b-Ax)=0, \tag{12}$$

$$u\ge 0, \tag{13}$$

where u is a q-dimensional vector of dual variables. Now, multiplying (11) on the right by $x/2$, adding $cx/2$ to both sides of the equation, and rearranging gives

$$\frac{1}{2}(cx-u^{\top}Ax)=cx-\frac{1}{2}x^{\top}Dx. \tag{14}$$

From (12) we observe that $u^{\top}b=u^{\top}Ax$, so (14) becomes

$$\frac{1}{2}(cx-u^{\top}b)=cx-\frac{1}{2}x^{\top}Dx. \tag{15}$$

Replacing the objective function in (9) with the left-hand side of (15), and appending the Kuhn–Tucker conditions to (9) results in

$$\begin{cases}\theta^*=\min\limits_{x,u} & cx-u^{\top}b, \\ \text{s.t.} & Ax\le b, \\ & x^{\top}D-u^{\top}A=c, \\ & u^{\top}(b-Ax)=0, \\ & u\ge 0,\end{cases} \tag{16}$$

which is an alternative representation of (9). Thus a quadratic objective function in (9) has been traded for a complementarity constraint in (16).

Turning attention to this term, let z be a q-dimensional nonnegative vector and note that $u^{\top}(b-Ax)$

can be replaced by $z_i = \min[u_i, (b - Ax)_i]$, $i = 1, \ldots, m$, where $(b - Ax)_i$ is the ith component of $b - Ax$, as long as $\sum_i z_i = 0$. The aim is to show that the following linear minmax problem is equivalent to (16):

$$\begin{cases} \theta^0 = \min\limits_{x,u} & cx - u^\top b + \sum\limits_{i=1}^{q} Mz_i, \\ \text{s.t.} & Ax \le b, \\ & x^\top D - u^\top A = cu \ge 0, \\ \max\limits_{z} & \sum\limits_{i=1}^{q} Mz_i, \\ \text{s.t.} & z_i \le u_i, \quad i = 1, \ldots, q, \\ & z_i \le (b - Ax)_i, \quad i = 1, \ldots, q, \\ & z_i \ge 0, \quad i = 1, \ldots, q, \end{cases} \tag{17}$$

where M in the objective functions of problem (17) is a sufficiently large constant whose value must be determined.

Before proceeding, observe that an optimal solution to (16), call it (x^*, u^*), is feasible to (17) and yields the same value for the first objective function in (17). This follows because $\sum_i z_i^* = 0$, where $z_i^* = z_i^*(x^*, u^*)$. It must now be shown that (x^*, u^*, z^*) also solves (17). Assume the contrary; i. e., there exists a vector (x_0, u_0, z_0) in the inducible region of (17) such that $\theta^0 < \theta^*$ and $\sum_i z_i^0 > 0$. (Of course, if $\sum_i z_i^0 = 0$ and $\theta^0 < \theta^*$ this would contradict the optimality of (x^*, u^*).)

To exhibit a contradiction an appropriate value of M is needed. Accordingly, let S be the polyhedron defined by all the constraints in (17) and let

$$\theta^+ = \min\left\{cx - u^\top b \colon\ (x, u, z) \in S\right\}. \tag{18}$$

Evidently, because S is compact, θ^+ in (18) is finite. Compactness follows from the assumption that $\{x\colon Ax \le b\}$ is bounded, and the fact that A has full row rank which implies that u is bounded in the second constraint in (17). Now define:

$$M > \frac{\theta^* - \theta^+}{\sum_i z_i^0 - \epsilon} \ge 0,$$

where ϵ is any value in $(0, \sum_i z_i^0)$. This leads to the following series of inequalities:

$$\theta^0 = cx^0 - b^\top u^0 + M\sum_i z_i^0 < cx^* - b^\top u^* = \theta^*$$

or

$$(cx^* - b^\top u^*) - (cx^0 - b^\top u^0) > M\sum_i z_i^0. \tag{19}$$

But from the definition of M along with (19), one has

$$\begin{aligned} M\sum_i z_i^0 - M\epsilon &> \theta^* - \theta^+ \\ &\ge (cx^* - b^\top u^*) - (cx^0 - b^\top u^0) > M\sum_i z_i^0 \end{aligned}$$

which implies that the open interval $(0, \sum_i z_i^0)$ does not exist so $\sum_i z_i^0 = 0$, the desired contradiction.

Similar arguments can be used to show the reverse; therefore, if (x^*, u^*) solves (16), it also solves (17) and vice versa. Finally, note that the transformation from (9) to (17) is polynomial because it only involves the addition of $2q$ variables and $2q + n$ constraints to the formulation. The statement of the theorem follows from these developments. A straightforward corollary is that the linear BLPP is *NP*-hard.

In describing the size of a problem instance, I, it is common to reference two variables:

1) its Length[I], which is an integer corresponding to the number of symbols required to describe I under some reasonable encoding scheme, and
2) its Max[I], also an integer, corresponding to the magnitude of the largest number in I.

When a problem is said to be solvable in polynomial time, it means that an algorithm exists that will return an optimal solution in an amount of time that is a polynomial function of the Length[I]. A closely related concept is that of a *pseudopolynomial time algorithm* whose time complexity is bounded above by a polynomial function of the two variables Length[I] and Max[I]. By definition, any polynomial time algorithm is also a pseudopolynomial time algorithm because it runs in time bounded by a polynomial in Length[I]. The reverse is not true.

The theory of *NP*-completeness states that *NP*-hard problems are not solvable with polynomial time algorithms unless $P = NP$; however, a certain subclass may be solvable with pseudopolynomial time algorithms. Problems that do not yield to pseudopolynomial time algorithms are classified as *NP*-hard in the strong sense.

The linear BLPP falls into this category. The proof in [6], once again, is actually a corollary to the following theorem.

Theorem 2 *The linear minmax problem is strongly NP-hard.*

The proof is based on the notion of a *kernel K* of a graph $G = (V, E)$ which is a vertex set that is stable (no two vertices of K are adjacent) and absorbing (any vertex not in K is adjacent to a vertex of K). It is shown that the strongly *NP*-hard problem of determining whether or not G has a kernel (see [5]) is equivalent to determining whether or not a particular LMMP has an optimal objective function value of zero.

See also

- Bilevel Fractional Programming
- Bilevel Linear Programming
- Bilevel Optimization: Feasibility Test and Flexibility Index
- Bilevel Programming
- Bilevel Programming: Applications
- Bilevel Programming: Applications in Engineering
- Bilevel Programming: Implicit Function Approach
- Bilevel Programming: Introduction, History and Overview
- Bilevel Programming in Management
- Bilevel Programming: Optimality Conditions and Duality
- Concave Programming
- Minimax: Directional Differentiability
- Minimax Theorems
- Minimum Concave Transportation Problems
- Multilevel Methods for Optimal Design
- Multilevel Optimization in Mechanics
- Nondifferentiable Optimization: Minimax Problems
- Stochastic Bilevel Programs
- Stochastic Programming: Minimax Approach
- Stochastic Quasigradient Methods in Minimax Problems

References

1. Audet C, Hansen P, Jaumard B, Savard G (1996) On the linear maxmin and related programming problems. GERAD - École des Hautes Études Commerciales, Montreal, Working paper G-96–15
2. Bard JF (1991) Some properties of the bilevel programming problem. J Optim Th Appl 68(2):371–378
3. Bard JF, Falk JE (1982) An explicit solution to the multi-level programming problem. Comput Oper Res 9(1):77–100
4. Ben-Ayed O, Blair CE (1990) Computational difficulties of bilevel linear programming. Oper Res 38(1):556–560
5. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, New York
6. Hansen P, Jaumard B, Savard G (1992) New branch-and-bound rules for linear bilevel programming. SIAM J Sci Statist Comput 13(1):1194–1217
7. Jeroslow RG (1985) The polynomial hierarchy and a simple model for competitive analysis. Math Program 32:146–164
8. Shimizu K, Ishizuka Y, Bard JF (1997) Nondifferentiable and two-level mathematical programming. Kluwer, Dordrecht
9. Simaan M (1977) Stackelberg optimization of two-level systems. IEEE Trans Syst, Man Cybern SMC-7(4):554–556
10. White DJ, Anandalingam G (1993) A penalty function approach for solving bi-level linear programs. J Global Optim 3:397–419

Bilevel Optimization: Feasibility Test and Flexibility Index

MARIANTHI IERAPETRITOU
Department Chemical and Biochemical Engineering, Rutgers University, Piscataway, USA

MSC2000: 90C26

Article Outline

Keywords
Problem Statement
Local Optimization Framework
Design Optimization
Bilevel Optimization
Global Optimization Framework
Feasibility Test
Flexibility Index
Illustrative Example
Conclusions
See also
References

Keywords

Bilevel optimization; Uncertainty; Flexibility

Production systems typically involve significant *uncertainty* in their operation due to either external or internal resources. Variability of process parameters during operation and plant model mismatch (both parametric and structural) could give rise to suboptimality and even infeasibility of the deterministic solutions. Consequently, plant flexibility has been recognized to represent one of the important components in the operability of the production processes.

In a broad sense the area covers

- a *feasibility test* that requires constraint satisfaction over a specified space of uncertain parameters;
- a *flexibility index* associated with a given design that represents a quantitative measure of the range of uncertainty space that satisfies the feasibility requirement; and
- the integration of design and operations where trade-offs between design cost and plant flexibility are considered.

K.P. Halemane and I.E. Grossmann [21] proposed a feasibility measure for a given design based on the worst points for feasible operation, which can be mathematically formulated as a *max-min-max optimization problem* as will be discussed in detail in the next section.

Different approaches exist in the literature that quantify the flexibility for a given design involve the deterministic measures such as the resilience index, RI, proposed in [38], the flexibility index proposed in [41,42] and the stochastic measures such as the design reliability proposed in [27] and the stochastic flexibility index proposed in [37] and [40].

The incorporation of uncertainty into design optimization problems transforms the deterministic process models to stochastic/parametric problems, the solution of which requires the application of specialized optimization techniques. The consideration of the feasibility objective within the design optimization can be targeted towards the following two design capabilities. The first one concerns the design with *fixed degree of flexibility* that has the capability to cope with a finite number of different operating conditions ([19,20,32,34,40]). The second one considers the design optimization with *optimal degree of flexibility* that can be achieved by the trade-off of the cost of the plant and its flexibility ([22,33,35,36]). In the next section the feasibility test and the flexibility index problem will be considered in detail.

Problem Statement

The design problem can be described by a set of equality constraints I and inequality constraints J, representing plant operation and design specifications:

$$\begin{aligned} h_i(d,z,x,\theta) &= 0, \quad i \in I, \\ g_j(d,z,x,\theta) &\le 0, \quad j \in J, \end{aligned} \tag{1}$$

where d corresponds to the vector of design variables, z the vector of control variables, x the state variables and θ the vector of uncertain parameters. As has been shown in [21] for a specific design d, given this set of constraints, the design feasibility test problem can be formulated as the max-min-max problem:

$$\chi(d) = \max_{\theta\in T} \min_{z} \max_{\substack{j\in J,\\ i\in I}} \left\{ \begin{array}{l} h_i(d,z,x,\theta) = 0; \\ g_j(d,z,x,\theta) \le 0 \end{array} \right\}, \tag{2}$$

where the function $\chi(d)$ represents a feasibility measure for design d. If $\chi(d) \le 0$, design d is feasible for all $\theta \in T$, whereas if $\chi(d) > 0$, the design cannot operate for at least some values of $\theta \in T$. The above max-min-max problem defines a nondifferentiable global optimization problem which however can be reformulated as the following two-level optimization problem:

$$\begin{cases} \chi(d) & = \max\limits_{\theta\in T} \psi(d,\theta) \\ \text{s.t.} & \psi(d,\theta) \le 0 \\ \psi(d,\theta) & = \min\limits_{z,u} u \\ \text{s.t.} & h_i(d,z,x,\theta) = 0, \quad i \in I, \\ & g_j(d,z,x,\theta) \le u, \quad j \in J, \end{cases} \tag{3}$$

where the function $\psi(d,\theta) = 0$ defines the boundary of the feasible region in the space of the uncertain parameters θ.

Plant feasibility can be quantified by determining the flexibility index of the design. Following the definition of flexibility index as proposed in [41], this metric expresses the largest scaled deviation δ of any expected deviations $\Delta\,\theta^+$, $\Delta\,\theta^-$, that the design can handle. The mathematical formulation for the evaluation of design's

flexibility is the following:

$$\begin{cases} F & = \max \delta \\ \text{s.t.} & \chi(d) \\ & \quad = \max\limits_{\theta \in T} \min\limits_{z} \max\limits_{\substack{j \in J, \\ i \in I}} \left\{ \begin{array}{l} h_i(d,z,x,\theta) = 0; \\ g_j(d,z,x,\theta) \le 0 \end{array} \right\}, \\ & T(\delta) = \left\{ \theta : \begin{array}{l} \theta^N - \delta\Delta\theta^- \le \theta \\ \le \theta^N + \delta\Delta\theta^+ \end{array} \right\}, \\ & \delta \ge 0. \end{cases} \tag{4}$$

The design flexibility index problem can be reformulated to represent the determination of the largest hyperrectangle that can be inscribed within the feasible region of the design [41]. Following this idea, the mathematical formulation of the flexibility problem has the following form:

$$\begin{cases} F & = \min \delta \\ \text{s.t.} & \psi(d,\theta) = 0, \\ & \psi(d,\theta) = \min\limits_{z} u, \\ & h_i(d,z,x,\theta) = 0, \quad i \in I, \\ & g_j(d,z,x,\theta) \le u, \quad j \in J, \\ & T(\delta) = \left\{ \theta : \begin{array}{l} \theta^N - \delta\Delta\theta^- \le \theta \\ \le \theta^N - \delta\Delta\theta^+ \end{array} \right\}, \\ & \delta \ge 0. \end{cases} \tag{5}$$

Local Optimization Framework

For the case where the constraints are jointly 1-D quasiconvex in θ and quasiconvex in z it was proven [41] that the point θ_c that defines the solution to (3) lies at one of the vertices of the parameter set T. Based on this assumption, the critical uncertain parameter points correspond to the vertices and the feasibility test problem is reformulated in the following manner:

$$\chi(d) = \max_{k \in V} \psi(d, \theta^k), \tag{6}$$

where $\psi(d, \theta^k)$ is the evaluation of the function $\psi(d, \theta)$ at the parameter vertex θ^k and V is the index set for the 2^{n_p} vertices for the n_p uncertain parameters θ. In a similar fashion for the flexibility index, problem (4) is reformulated in the following way:

$$F = \min_{k \in V} \delta^k, \tag{7}$$

where δ^k is the maximum deviation along each vertex direction $\Delta\,\theta^k$, $k \in V$, and is determined by the following problem:

$$\begin{cases} \delta^k & = \max\limits_{\delta, z} \delta \\ \text{s.t.} & g_j(d,z,x,\theta) \le 0, \quad j \in J, \\ & h_i(d,z,x,\theta) = 0, \quad i \in I, \\ & \theta = \theta^N + \Delta\theta^k, \\ & \delta \ge 0. \end{cases} \tag{8}$$

Based on the above problem reformulations, a direct search method was proposed [21] that explicitly enumerate all the parameter set vertices. To avoid the explicit vertex enumeration, proposed two algorithms were proposed [41,42]: a heuristic vertex search and an implicit enumeration scheme. These algorithms however, rely on the assumption that the critical points correspond to the vertices of the parameter set T which is valid only for the type of constraints assumed above. To circumvent this limitation, a solution approach was proposed based on the following ideas [18]:

a) They replace the inner optimization problem:

$$\begin{cases} \psi(d,\theta) & = \min\limits_{z,u} u \\ & h_i(d,z,x,\theta) = 0, \quad \forall i \in I, \\ & g_j(d,z,x,\theta) \le u, \quad \forall j \in J, \end{cases}$$

by the *Karush–Kuhn–Tucker optimality conditions* (KKT):

$$\begin{aligned} & \sum_{j \in J} \lambda_j = 1, \\ & \sum_{j \in J} \lambda_j \frac{\partial g_j}{\partial z} + \sum_{i \in I} \mu_i \frac{\partial h_i}{\partial z} = 0, \\ & \sum_{j \in J} \lambda_j \frac{\partial g_j}{\partial x} + \sum_{i \in I} \mu_i \frac{\partial h_i}{\partial x} = 0, \\ & \lambda_j s_j = 0, \quad j \in J, \\ & s_j = u - g_j(d,z,x,\theta), \quad j \in J, \\ & \lambda_j, s_j \ge 0, \quad j \in J, \end{aligned}$$

where s_j are the slack variables of constraints j, λ_j, μ_i are the Lagrange multipliers for inequality and equality constraints, respectively.

b) For the inner problem the following property holds that if each square submatrix of dimension ($n_z \times$

n_z), where n_z is the number of control variables, of the partial derivatives of the constraints g_j, $\forall j \in J$ with respect to the control variables z is of full rank, then the number of the *active constraints* is equal to $n_z + 1$.

c) They utilize the discrete nature of the selection of the active constraints by introducing a set of binary variables y_j to express if constraint g_j is active. In particular:

$$\begin{aligned}
&\lambda_j - y_j \le 0, \quad j \in J,\\
&s_j - U(1 - y_j) \le 0, \quad j \in J,\\
&\sum_{j\in J} y_j = n_z + 1,\\
&\delta \ge 0,\\
&y_j = 0, 1, \quad \lambda_j, s_j \ge 0, \quad j \in J,
\end{aligned}$$

where U represents an upper bound to the slack variables s_j. Note that if $y_j = 1$, then $\lambda_j \ge 0$, $s_j = 0$ which indicates that the constraint j is active, on the other hand if $y_j = 0$, then $\lambda_j = 0$, $s_j \ge 0$ which indicates that the constraint j is inactive.

Based on these ideas, the feasibility test problem can be reformulated in the following way:

$$\begin{cases}
\chi(d) & = \max u\\
\text{s.t.} & h_i(d, z, x, \theta) = 0,\\
& g_j(d, z, x, \theta) + s_j - u = 0,\\
& \sum_{j\in J} \lambda_j = 1,\\
& \sum_{j\in J} \lambda_j \frac{\partial g_j}{\partial z} + \sum_{i\in I} \mu_i \frac{\partial h_i}{\partial z} = 0,\\
& \sum_{j\in J} \lambda_j \frac{\partial g_j}{\partial x} + \sum_{i\in I} \mu_i \frac{\partial h_i}{\partial x} = 0,\\
& \lambda_j - y_j \le 0, \quad j \in J,\\
& s_j - U(1 - y_j) \le 0, \quad j \in J,\\
& \sum_{j\in J} y_j = n_z, +1\\
& \theta^L \le \theta \le \theta^U,\\
& \delta \ge 0,\\
& y_j = 0, 1, \quad \lambda_j, s_j \ge 0, \quad j \in J,
\end{cases}$$

which corresponds to a *mixed integer optimization problem* either linear or nonlinear depending on the nature of the constraints. In a similar way the flexibility index problem takes the following form:

$$\begin{cases}
F & = \min \delta\\
\text{s.t.} & h_i(d, z, x, \theta) = 0,\\
& g_j(d, z, x, \theta) + s_j - u = 0,\\
& u = 0,\\
& \sum_{j\in J} \lambda_j = 1,\\
& \sum_{j\in J} \lambda_j \frac{\partial g_j}{\partial z} + \sum_{i\in I} \mu_i \frac{\partial h_i}{\partial z} = 0,\\
& \sum_{j\in J} \lambda_j \frac{\partial g_j}{\partial x} + \sum_{i\in I} \mu_i \frac{\partial h_i}{\partial x} = 0,\\
& \lambda_j - y_j \le 0, \quad j \in J,\\
& s_j - U(1 - y_j) \le 0, \quad j \in J,\\
& \sum_{j\in J} y_j = n_z + 1,\\
& \theta^L \le \theta \le \theta^U,\\
& \delta \ge 0,\\
& y_j = 0, 1, \quad \lambda_j, s_j \ge 0, \quad j \in J.
\end{cases}$$

Grossmann and C.A. Floudas [18] proposed the active set strategy for the solution of the above reformulated problems based on the property that for any combination of $n_z + 1$ binary variables that is selected (i. e., for a given set of active constraints), all the other variables can be determined as a function of θ. They proposed a procedure of systematically identifying the potential candidates for the active sets based on the signs of the gradients $\nabla_z g_j(d, z, x, \theta)$. The algorithm for the feasibility test problem involves the following steps:

a) For every potential active set determine the value u^k, $k = 1, \ldots, n_{\mathrm{AS}}$, through the solution of the following *nonlinear programming problem*:

$$\begin{cases}
u^k & = \max u\\
\text{s.t.} & h_i(d, z, x, \theta) = 0,\\
& g_j(d, z, x, \theta) - u = 0, \quad j \in \mathrm{AS}(k),\\
& \theta^L \le \theta \le \theta^U,\\
& \delta \ge 0,\\
& y_j = 0, 1, \quad \lambda_j, s_j \ge 0, \quad j \in J,
\end{cases}$$

or if the active set AS(k) involves lower and upper bound constraints u^k is given by:

$$u^k = \frac{1}{\alpha_k}\left(\sum_{j(l)\in \mathrm{AS}(k)} a_{j(l)} - \sum_{j(u)\in \mathrm{AS}(k)} b_{j(u)}\right),$$

where $j(l)$, $j(u)$ are the indices that correspond to those pairs of constraints representing the lower and upper bounds on the same function, and α_k is the total number of this type of constraints.

b) The solution for the feasibility test problem is given by:

$$\chi(d) = \max_{k\in \mathrm{AS}(k)} u^k.$$

A similar algorithm was proposed for the solution of the flexibility index problem. Under the conditions that the functions $\psi(d, \theta)$ are quasiconcave in z and θ and strictly quasiconvex in z for fixed θ, the approach guarantees global optimality.

For the linear case where the feasibility function problem has the following form:

$$\begin{cases} \psi(d,\theta) & = \min\limits_{z,u} u \\ & f_j(d,z,\theta) = \\ & \quad = \beta_{1j} d + \beta_{2j}(\theta) z - b_{2j}(\theta) \le u, \\ & \forall j \in J, \end{cases}$$

where f_j are the inequality constraints after the elimination of the state variables. For this case, analytical expressions have been derived [32,33] the function $\psi^k(d, \theta)$ for a given active set k:

$$\psi^k(d,\theta) = \sum_{j\in J^k_{\mathrm{AS}}} \lambda^k_j (\beta_{1j} d + \beta_{2j}(\theta) z - b_{2j}(\theta)).$$

A branch and bound approach based on the evaluation of upper and lower bounds of function $\chi(d)$ was proposed in [30]. Although the suggested bounding problems are simpler than the original feasibility test problem they correspond to bilevel optimization problems where global optimality cannot be guaranteed using local optimization methods, (see [29]).

Design Optimization

As mentioned above, the incorporation of the feasibility objectives within the design optimization framework can be targeted towards the design with fixed degree of flexibility that is able to accommodate a finite number of changing operational conditions and the design with optimal degree of flexibility determined by proper balance of economic optimality and plant feasibility. The design optimization for fixed degree of flexibility was considered in [20], which presents a general formulation for designing multipurpose plants considering a deterministic multiperiod model of the following form:

$$\begin{cases} \min & C^0(d) + \sum\limits_{i=1}^{N} C^i(d, z^i, x^i, t^i) \\ \text{s.t.} & h^i(d, z^i, x^i, t^i) = 0, \\ & g^i(d, z^i, x^i, t^i) \le 0, \\ & r(d, z^1, \ldots, z^N, x^1, \ldots, x^N, \\ & \quad t^1, \ldots, t^N) \le 0, \end{cases}$$

where d is the vector of design variables; z^i is the vector of control variables in period i; x^i is the vector of state variables in period i; t^i is the length of time for period i; h^i is the vector of equalities for period i; g^i is the vector of inequalities for period i; r is the vector of inequalities that involve variables of all periods; N is the number of periods where different operating conditions are considered. This problem formulation exhibits a block diagonal structure which has been exploited for computational efficiency. See [19] for the *projection-restriction strategy*, which is an iterative scheme between economic optimization and design feasibility. Based of the flexibility analysis and the assumption that the critical points lie on the vertices of the uncertain parameter set T, Halemane and Grossmann [21] proposed an iterative algorithm to solve the problem of design considering specific range of uncertainty. See [31], for a nested solution procedure combining *generalized Benders decomposition* and *outer approximation* algorithms to address the problem of multiperiod design of heat integrated distillation sequences. See [44] for an outer approximation based decomposition method for the solution of multiperiod design problems.

For design optimization with optimal degree of flexibility, E.N. Pistikopoulos and Grossmann proposed [33] an iterative scheme in order to construct the trade-

off curves relating retrofit cost and expected revenue to flexibility. In their later work, [34,35], they extend their approach to nonlinear systems. Briefly, their iterative scheme consists of two phases: first the trade-off curve of retrofit cost and flexibility is determined, from this curve a number of designs are obtained for which at the second phase the expected revenue is evaluated by employing a modified Cartesian integration method. See [40], for a solution method based on generalized decomposition for the maximization of the plant flexibility subject to cost constraint. Recently (1996), a decomposition based approach for the simultaneous optimization of design economics and plant feasibility was proposed [22]. The main ideas of the proposed approach are:

- the utilization of a modified Benders decomposition scheme where the design variables correspond to the complicating variables;
- the use of a numerical integration formula for the approximation of the multiple integral of expected revenue; and
- the determination of the unknown integration points as part of the optimization procedure through the solution of a series of feasibility subproblems.

The same approach has been employed for the solution of planning and capacity expansion problems, [24], for the design of batch plants where additional properties simplified the solution procedure, [25]. The limitation of the later approach however, is that it cannot guarantee global optimality for the general case.

Bilevel Optimization

It should be noted that the feasibility test problem and the flexibility index problem correspond to bilevel optimization problems where the inner level consists of the evaluation of the function $\psi(d, \theta)$ that defines the boundary of the feasible region. Approaches that exist in the literature to deal with the solution of the bilevel optimization problem for the linear case involve the *enumeration techniques* that based on the fact that the solution must occur at an extreme point of the feasible set as the methods proposed in [13,14], and [12], *reformulation techniques* based on the transformation of the original problem to a single optimization problem by employing the optimality KKT conditions of the inner level problem as the methods proposed in [11], based on branch and bound principles [17], based on mixed integer programming [26], based on parametric complementarity pivoting [8], based on local optimization approaches for nonlinear programming such as penalty and barrier function methods, and global optimization techniques based on the reformulation of the complementarity slackness constraint to a separable quadratic reverse convex inequality constraint ([6]), or the restatement of the original problem as a reverse convex program ([43]). Recently (1996), [45], a global optimization framework was proposed based on the reformulation of the bilevel linear problem utilizing the KKT optimality conditions of the inner level and the primal-dual global optimization approach proposed in [15,16].

For the nonlinear case, local optimization techniques has been proposed based on the one-dimensional search algorithm [10], and penalty function methods as the approach proposed in [5]. Recently (1998) a general global optimization approach was proposed [23] for the solution of the feasibility test and flexibility index problem based on a utilization of a branch and bound framework and the ideas of the *deterministic global optimization algorithm* αBB, [1,3,4,9]. Although the proposed approach was applied to design feasibility/flexibility problems, it can be extended to general nonlinear bilevel problems. In the next section the main ideas and basic steps of the later approach for the solution of the feasibility test and flexibility index problems.

Global Optimization Framework

The basic idea of the proposed framework that leads to the determination of the global optimal solution for both the feasibility test and the flexibility index problem is to generate a *relaxation/enlargement of the feasible region* based on the convexification of the original problem constraints. Since the enlarged feasible region involves more feasible points than the original feasible region, the resulting feasibility test and flexibility index problem will provide lower bounds to the global solutions. Based on this relaxation idea, the proposed approach involves the following key steps:

a) Since the constraints $g_j(d, z, x, \theta)$, $h_i(d, z, x, \theta)$ are nonconvex functions, the *Karush–Kuhn–Tucker*

optimality conditions (KKT) of the inner problem that correspond to the optimization of the $\psi(d, \theta)$ function, are not necessary and sufficient to guarantee global optimality of the feasibility test and the flexibility index problems. Hence, the first step of the proposed framework involves the convexification of the constraints $g_j(d, z, x, \theta)$, $h_i(d, z, x, \theta)$ of the original problem. For the convexified problem and assuming that the constraint qualification holds, the KKT optimality conditions are necessary and sufficient, [18], and therefore we maintain the equivalence of the transformed single stage optimization problem. The solution of the single stage problem provides a lower bound of the design flexibility function $\chi(d)$ and the flexibility index of the design, F.

b) An upper bound to the design flexibility function $\chi(d)$ and flexibility index F is determined through a feasible solution of the original MINLP formulation obtained by substituting the inner problem by the KKT optimality conditions.

c) The next step after establishing an upper and a lower bound on the global solution, is to refine them. This is accomplished by successfully partitioning the initial region of the uncertain and control variables into smaller ones. The partitioning strategy involves the successive subdivision of a hyperrectangle into two subrectangles by halving on the middle point of the longest side of the initial rectangle (bisection). In each iteration the lower bound of the feasibility test and the flexibility index problem is the minimum over all the minima found in every subrectangle composing the initial rectangle. Consequently, a nondecreasing sequence of lower bounds is generated by halving the subrectangle that is responsible for the infimum over the minima obtained at each iteration. An nonincreasing sequence of upper bounds is derived by solving the nonconvex MINLP single optimization problem obtained after the substitution of the inner problem by the KKT optimality conditions, and selecting as an upper bound the minimum over all previously determined upper bounds. If at any iteration the solution of the convexified MINLP in any subrectangle is found to be greater than the upper bound, this subrectangle is fathomed since the global solution cannot be found inside it.

Feasibility Test

The procedure for the global optimization of the design feasibility problem involves the following steps:

1) Consider the whole uncertainty space. Set the lower bound LB = $-\infty$, $K = 1$ and select a tolerance ϵ.
2) Evaluate the valid underestimators of the original constraints $g_j(d, z, x, \theta)$, $h_i(d, z, x, \theta)$ utilizing the basic principles of the deterministic global optimization algorithm αBB, [1,3,4,9].
3) Considering the convexified constraints substitute the inner optimization problem by the necessary and sufficient KKT optimality conditions.
4) Solve the resulting MINLP formulation to global optimality using the deterministic global optimization algorithm SMIN-αBB, or GMIN-αBB, [2]. If the obtained solution is greater than the current LB, update the LB.
5) Substitute the inner optimization problem of the original problem (i. e., without convexifying the constraints) Solve the resulting problem using a local MINLP optimizer (e. g. DICOPT, [46], MINOPT, [39]). Set the upper bound UB equal to the obtained solution.
6) Check for convergence. If UB $-$ LB $\leq \epsilon$, STOP, otherwise continue to step 7).
7) Apply one of the branching criteria to partition the initial domain into two subdomains to be considered at the next iteration. Once the branching variable is selected the subdivision is performed by halving on the middle point of the longest side of the initial rectangle (bisection). The selection of a branching variable can be made following different branching rules. Since the aim of the branching step is the generation of problems with tighter lower bounds, the control variables, u, that participate in nonconvex terms and the uncertain parameters, θ, are involved in the set of candidate branching variables. The control variable or uncertain parameter that is selected for branching, correspond to the least-reduced axis, that is, the largest

$$r_i = \frac{x_i^U - x_i^L}{x_{i,0}^U - x_{i,0}^L}.$$

Note, that alternative branching strategies may be applied as described in [1,3].

Flexibility Index

The procedure for the global optimization of the flexibility index problem (5), involves the following steps:

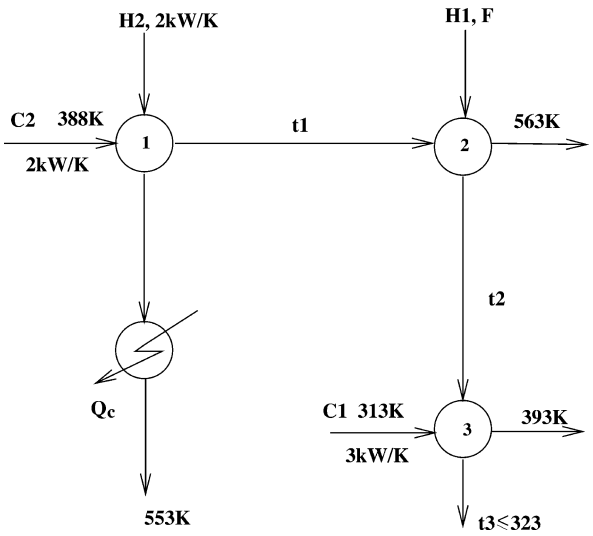
1) Consider the whole uncertainty space. Set the lower bound LB = $-\in$, $K = 1$ and select a tolerance ϵ.
2) Substitute the inner optimization problem by the KKT optimality conditions and construct the following single stage MINLP optimization problem. Solve the resulting problem using a local MINLP optimizer (e. g., DICOPT, [46], MINOPT, [39]). Set the upper bound UB equal to the obtained solution.
3) Determine the valid underestimators of the original constraints $g_j(d, z, x, \theta)$, $h_i(d, z, x, \theta)$ utilizing the basic principles of the deterministic global optimization algorithm αBB, [1,3,4,9].
4) Considering the convexified constraints substitute the inner optimization problem by the necessary and sufficient KKT optimality conditions.
5) Solve the resulting MINLP formulation to global optimality using the deterministic global optimization algorithm SMIN-αBB, or GMIN-αBB, [2]. If the obtained solution is greater than the current LB, update the LB.
6) Check for convergence. If UB $-$ LB $\leq \epsilon$, STOP, otherwise continue to step 7).
7) Apply one of the branching criteria to partition the initial domain into two subrectangles to be considered at the next iteration.

Illustrative Example

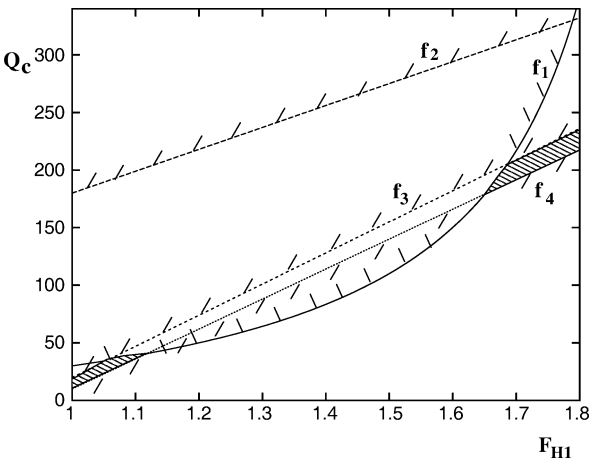
In this section an example of a heat exchanger network is considered to illustrate the steps of the approaches presented in the previous sections for the feasibility test and flexibility index problems.

The heat exchanger network given in [18] is considered here as shown in Fig. 1. The uncertain parameter is the heat flowrate of stream H1 which has a nominal value of 1kW/K and an expected deviation of +0.8kW/K. The following inequalities determine the feasible operation of this network as they formulated after the elimination of the state variables:

$$f_1 = -25F_{H1} + Q_c - 0.5Q_cF_{H1} + 10 \leq 0,$$
$$f_2 = -190F_{H1} + Q_c + 10 \leq 0,$$
$$f_3 = -270F_{H1} + Q_c + 250 \leq 0,$$
$$f_4 = 260F_{H1} - Q_c - 250 \leq 0.$$



Bilevel Optimization: Feasibility Test and Flexibility Index, Figure 1
Heat exchanger network



Bilevel Optimization: Feasibility Test and Flexibility Index, Figure 2
Feasible region

The feasible region of the network is illustrated in Fig. 2. Note that the feasible region consists of the two disconnected domains which are highlighted in black.

First the feasibility test problem is solved. Constraint f_1 corresponds to the only nonconvex constraint involving the bilinear term Q_cF_{H1}. By introducing a new variable w for the bilinear term Q_cF_{H1} and introducing the four linear inequality constraints (f_5–

f_8) that define its convex envelope, [4,7,28], we have:

$$
\begin{aligned}
f_1 &= -25F_{H1} + Q_c - 0.5w + 10 \le 0,\\
f_2 &= -190F_{H1} + Q_c + 10 \le 0,\\
f_3 &= -270F_{H1} + Q_c + 250 \le 0,\\
f_4 &= 260F_{H1} - Q_c - 250 \le 0,\\
f_5 &= 10F_{H1} + Q_c - w - 10 \le 0,\\
f_6 &= 236F_{H1} + 1.8Q_c - w - 424.8 \le 0,\\
f_7 &= -10F_{H1} - 1.8Q_c + w + 18 \le 0,\\
f_8 &= -236F_{H1} - Q_c + w + 236 \le 0.
\end{aligned}
$$

Considering this set of linear constraints and substituting the inner optimization problem of the feasibility test problem by the necessary and sufficient KKT optimality conditions the following MILP optimization formulation is obtained:

$$
\begin{cases}
\chi(d) = \max u\\
\text{s.t.}\quad f_1 = -25F_{H1} + Q_c\\
\qquad +10 - 0.5w + s_1 = u,\\
\qquad f_2 = -190F_{H1} + 10 + Q_c + s_2 = u,\\
\qquad f_3 = -270F_{H1} + 250 + Q_c + s_3 = u\\
\qquad f_4 = 260F_{H1} - 250 - Q_c + s_4 = u,\\
\qquad f_5 = 10F_{H1} + Q_c - w - 10 + s_5 = u,\\
\qquad f_6 = 236F_{H1} + 1.8Q_c\\
\qquad -w - 424.8 + s_6 = u,\\
\qquad f_7 = -10F_{H1} - 1.8Q_c\\
\qquad +w + 18 + s_7 = u,\\
\qquad f_8 = -236F_{H1} - Q_c\\
\qquad +w + 236 + s_8 - u = u,\\
\qquad \sum_{j=1}^{8} \lambda_j = 1,\\
\qquad \lambda_1 + \lambda_2 + \lambda_3 - \lambda_4 + \lambda_5\\
\qquad +1.8\lambda_6 - 1.8\lambda_7 - \lambda_8 = 0,\\
\qquad -0.5\lambda_1 - \lambda_5 - \lambda_6 + \lambda_7 + \lambda_8 = 0,\\
\qquad \lambda_j - y_j \le 0, \quad j = 1,\ldots,8,\\
\qquad s_j - U(1 - y_j) \le 0, \quad j = 1,\ldots,8,\\
\qquad \sum_{j=1}^{8} y_j = 3,\\
\qquad F_{H1}^N - \delta\Delta F_{H1}^- \le F_{H1} \le F_{H1}^N + \delta\Delta F_{H1}^+,\\
\qquad \delta \ge 0,\\
\qquad y_j = 0,1, \quad \lambda_j, s_j \ge 0, \quad j = 1,\ldots,8.
\end{cases}
$$

Note that due to the introduction of an additional control variable w, the number of active constraints is increased to three. The solution of the above MILP optimization using GAMS/CPLEX is found to be equal to 0. Since this value corresponds to the lower bound of network feasibility $\chi(d)$, this result suggests that the network is not feasible within the whole range of uncertainty, $F_{H1} \in (1, 1.8)$, and no further steps are required.

The plant flexibility is then determined. First the inner feasibility problem is substituted by the KKT optimality conditions. The resulting nonconvex MINLP problem is solved using the local MINLP solver MINOPT, [39]. The solution provides an upper bound of the heat exchanger network flexibility index of 0.148,

$$
\begin{cases}
F = \min \delta\\
\text{s.t.}\quad f_1 = -25F_{H1} + Q_c\\
\qquad +10 - 0.5Q_cF_{H1} + s_1 = 0,\\
\qquad f_2 = -190F_{H1} + 10 + Q_c + s_2 = 0,\\
\qquad f_3 = -270F_{H1} + 250 + Q_c + s_3 = 0,\\
\qquad f_4 = 260F_{H1} - 250 - Q_c + s_4 = 0,\\
\qquad \sum_{j=1}^{4} \lambda_j = 1,\\
\qquad -0.5F_{H1}\lambda_1 + \lambda_2 + \lambda_3 - \lambda_4 = 0,\\
\qquad \lambda_j - y_j \le 0, \quad j = 1,\ldots,4,\\
\qquad s_j - U(1 - y_j) \le 0, \quad j = 1,\ldots,4,\\
\qquad \sum_{j=1}^{4} y_j = 2,\\
\qquad F_{H1}^N - \delta\Delta F_{H1}^- \le F_{H1} \le F_{H1}^N + \delta\Delta F_{H1}^+,\\
\qquad \delta \ge 0,\\
\qquad y_j = 0,1, \quad \lambda_j, s_j \ge 0, \quad j = 1,\ldots,4.
\end{cases}
$$

Note that this formulation corresponds to a nonconvex MINLP due to the bilinear term (Q_c F_{H1}) in constraint f_1 and the bilinear term ($F_{H1}\lambda_1$) in the gradient KKT constraint.

In step 3), the original constraints are convexified using the αBB resulting in the set of linear constraints $f_1,\ldots,f_8$ as presented above in the solution of the feasibility test problem. In step 4), the KKT optimality conditions are written considering the new set of linear constraints leading to the formulation of the following

MILP problem:

$$
\begin{aligned}
F \quad &= \min \delta \\
\text{s.t.} \quad & f_1 = -25F_{\mathrm{H1}} + Q_{\mathrm{c}} \\
& \quad +10 - 0.5w + s_1 = 0, \\
& f_2 = -190F_{\mathrm{H1}} + 10 + Q_{\mathrm{c}} + s_2 = 0, \\
& f_3 = -270F_{\mathrm{H1}} + 250 + Q_{\mathrm{c}} + s_3 = 0, \\
& f_4 = 260F_{\mathrm{H1}} - 250 - Q_{\mathrm{c}} + s_4 = 0, \\
& f_5 = 10F_{\mathrm{H1}} + Q_{\mathrm{c}} - w - 10 + s_5 = 0, \\
& f_6 = 236F_{\mathrm{H1}} + 1.8Q_{\mathrm{c}} \\
& \quad -w - 424.8 + s_6 = 0, \\
& f_7 = -10F_{\mathrm{H1}} - 1.8Q_{\mathrm{c}} \\
& \quad +w + 18 + s_7 = 0, \\
& f_8 = -236F_{\mathrm{H1}} - Q_{\mathrm{c}} + w + 236 + s_8 = 0, \\
& \sum_{j=1}^{8} \lambda_j = 1, \\
& \lambda_1 + \lambda_2 + \lambda_3 - \lambda_4 + \lambda_5 \\
& \quad +1.8\lambda_6 - 1.8\lambda_7 - \lambda_8 = 0, \\
& -0.5\lambda_1 - \lambda_5 - \lambda_6 + \lambda_7 + \lambda_8 = 0, \\
& \lambda_j - y_j \le 0, \quad j = 1, \ldots, 8, \\
& s_j - U(1 - y_j) \le 0, \quad j = 1, \ldots, 8 \\
& \sum_{j=1}^{8} y_j = 3, \\
& F_{\mathrm{H1}}^N - \delta \Delta F_{\mathrm{H1}}^- \le F_{\mathrm{H1}} \le F_{\mathrm{H1}}^N + \delta \Delta F_{\mathrm{H1}}^+, \\
& \delta \ge 0, \\
& y_j = 0, 1, \quad \lambda_j, s_j \ge 0, \quad j = 1, \ldots, 8.
\end{aligned}
$$

The solution of this MILP problem using GAMS/CPLEX results in the network flexibility of 0.06 that provides a valid lower bound to the flexibility index problem. Hence, at the end of the first iteration we have an upper bound of 0.148 and a lower bound of 0.06 for the flexibility index problem. In step 7), since only one control variable is involved in the description of the problem, this corresponds to the branching variable resulting in the following subrectangles to be considered at the next iteration: subrectangle 1 described by $10 \le Q_c \le 123$ and subrectangle 1 described by $123 \le Q_c \le 236$. Steps 2) through 6) are then performed for each one of these subrectangles. For subrectangle 1, the resulting upper bounding MINLP gives a value of 0.148 the same as the lower bounding MILP in this region. Subrectangle 2, on the other hand results in a lower bound of 0.8138 which is larger than the current upper bound of 0.148 and consequently this region is fathomed and convergence is achieved to the global solution of network flexibility of 0.148.

Conclusions

The incorporation of uncertainty in the design stages is recognized to be one of the most important problems in the plant design analysis. Having efficient ways to test future plant feasibility and furthermore to quantify the capability of a plant to accommodate future variations of the operability parameters could lead to more efficient, economic and more flexible plants. Much of the work that appear in the literature to address the above problems was briefly presented in this paper. A general global optimization framework proposed in [23], was presented in more detail. Finally, an example problem was included to illustrate the main ideas of this framework.

See also

- ▶ Bilevel Fractional Programming
- ▶ Bilevel Linear Programming
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Programming
- ▶ Bilevel Programming: Applications
- ▶ Bilevel Programming: Applications in Engineering
- ▶ Bilevel Programming: Implicit Function Approach
- ▶ Bilevel Programming: Introduction, History and Overview
- ▶ Bilevel Programming in Management
- ▶ Bilevel Programming: Optimality Conditions and Duality
- ▶ Minimax: Directional Differentiability
- ▶ Minimax Theorems
- ▶ Multilevel Methods for Optimal Design
- ▶ Multilevel Optimization in Mechanics
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Stochastic Bilevel Programs
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Quasigradient Methods in Minimax Problems

References

1. Adjiman CS, Androulakis IP, Floudas CA (1997) A global optimization method αBB, for general twice-differentiable constrained NLPs - II. Implementation and computational results. Comput Chem Eng 22:1137–1158
2. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP problems in process synthesis. Comput Chem Eng 21:S445–S450
3. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1997) A global optimization method αBB, for general twice-differentiable constrained NLPs - I. Theoretical advances. Comput Chem Eng 22:1137–1158
4. Adjiman CS, Floudas CA (1996) Rigorous convex underestimates for general twice-differentiable problems. J Global Optim 9:23–40
5. Aiyoshi E, Shimizu K (1984) A solution method for the static constraint Stackelberg problem via penalty method. IEEE Trans Autom Control 29:1111
6. Al-Khayyal FA, Falk JE (1983) Jointly constrained biconvex programming. Math Oper Res 8:273–286
7. Al-Khayyal F, Horst R, Pardalos PM (1992) Global optimization on concave functions subject to quadratic constraints: An application in nonlinear bilevel programming. Ann Oper Res 34:125
8. Anandalingam G, White DJ (1990) A solution method for the linear static Stackelberg problem using penalty method. IEEE Trans Autom Control 35:1170
9. Androulakis IP, Maranas CD, Floudas CA (1995) αBB: A global optimization method for general constrained nonconvex problems. J Global Optim 7:337–363
10. Bard JF (1984) A solution method for the static constraint Stackelberg problem via penalty method. Naval Res Logist Quart 31:13
11. Bard JF, Falk JE (1982) An explicit solution to the multi-level programming problem. Comput Oper Res 9:77
12. Bard JF, Moore JT (1990) A branch and bound algorithm for the bilevel programming problem. SIAM J Sci Statist Comput 11:281
13. Bialas WF, Karwan MH (1984) Two-level linear programming. Managem Sci 30:1004
14. Candler W, Townsley R (1982) A linear two-level programming problem. Comput Oper Res 9:59
15. Floudas CA, Visweswaran V (1990) A global optimization algorithm (GOP) for certain classes of nonconvex NLPs - I. Theory. Comput Chem Eng 14:1397
16. Floudas CA, Visweswaran V (1993) Primal-relaxed dual global optimization approach. JOTA 78:187
17. Fortuny-Amat J, McCarl B (1981) A representation and economic interpretation of a two-level programming problem. J Oper Res Soc 32:783
18. Grossmann IE, Floudas CA (1987) Active constraint strategy for flexibility analysis in chemical processes. Comput Chem Eng 11:675–693
19. Grossmann IE, Halemane KP (1982) Decomposition strategy for designing flexible chemical plants. AIChE J 28: 686–694
20. Grossmann IE, Sargent RWH (1978) Optimum design of chemical plants with uncertain parameters. AIChE J 24:1021
21. Halemane KP, Grossmann IE (1983) Optimal process design under uncertainty. AIChE J 29:425–433
22. Ierapetritou MG, Acevedo J, Pistikopoulos EN (1996) An optimization approach for process engineering problems under uncertainty. Comput Chem Eng 20:703–709
23. Ierapetritou MG, Floudas CA (1998) Global optimization in design under uncertainty: Feasibility test and flexibility index problems. Manuscript in preparation
24. Ierapetritou MG, Pistikopoulos EN (1994) A novel optimization approach of stochastic planning models. Industr Eng Chem Res 18:163–189
25. Ierapetritou MG, Pistikopoulos EN (1996) Batch plant design and operations under uncertainty. Industr Eng Chem Res 35:772–787
26. Judice JJ, Faustino AM (1992) A sequential LCP method for bilevel linear programming. Ann Oper Res 89:34
27. Kubic WL, Stein FP (1988) A theory of design reliability using probability and fuzzy sets. AIChE J 34:583
28. McCormick GP (1975) Computability of global solutions to factorable nonconvex programs: Part I - Convex underestimation problems. Math Program 10:147–175
29. Migdalas A, Pardalos PM, Varbrand P (1998) Multilevel optimization: Algorithms and applications. Kluwer, Dordrecht
30. Ostrovsky GM, Volin YM, Barit EI, Senyavin MM (1994) Flexibility analysis and optimization of chemical plants with uncertain parameters. Comput Chem Eng 18:755–767
31. Paules GE, Floudas CA (1992) Stochastic programming in process synthesis: A two-stage model with MINLP recourse for multiperiod heat-integrated distillation sequences. Comput Chem Eng 16:189–210
32. Pistikopoulos EN, Grossmann IE (1988) Optimal retrofit design for improving process flexibility in linear systems. Comput Chem Eng 12:719–731
33. Pistikopoulos EN, Grossmann IE (1988) Stochastic optimization of flexibility in retrofit design of linear systems. Comput Chem Eng 12:1215–1227
34. Pistikopoulos EN, Grossmann IE (1989) Optimal retrofit design for improving process flexibility in nonlinear systems: -I. Fixed degree of flexibility. Comput Chem Eng 13:1003–1016
35. Pistikopoulos EN, Grossmann IE (1989) Optimal retrofit design for improving process flexibility in nonlinear systems: -II. Optimal degree of flexibility. Comput Chem Eng 13:1087–1096
36. Pistikopoulos EN, Ierapetritou MG (1995) A novel approach for optimal process design under uncertainty. Comput Chem Eng 19:1089–1110

37. Pistikopoulos EN, Mazzuchi TA (1990) A novel flexibility analysis approach for processes with stochastic parameters. Comput Chem Eng 14:991–1000
38. Saboo AK, Morari M, Woodcock DC (1983) Design of resilient processing plants – VIII. A resilience index for heat exchanger networks. Chem Eng Sci 40:1553–1565
39. Schweiger CS, Floudas CA (1996) MINOPT: A software package for mixed-integer nonlinear optimization, user's guide. Manual Princeton Univ, Computer-Aided Systems Lab, Dept Chem Engin Jan
40. Straub DA, Grossmann IE (1993) Design optimization of stochastic flexibility. Comput Chem Eng 17:339
41. Swaney RE, Grossmann IE (1985) An index for operational flexibility in chemical process design - Part I: Formulation and theory. AIChE J 26:139
42. Swaney RE, Grossmann IE (1985) An index for operational flexibility in chemical process design – Part II: Computational algorithms. AIChE J 31:631
43. Tuy H, Migdalas A, Varbrand P (1993) A quasiconcave minimization method for solving linear two-level programs. J Global Optim 4:243
44. Varvarezos DK, Grossmann IE, Biegler LT (1992) An outer-approximation for multiperiod design optimization. Industr Eng Chem Res 31:1466
45. Visweswaran V, Floudas CA, Ierapetritou MG, Pistikopoulos EN (1996) A decomposition-based global optimization approach for solving bilevel linear and quadratic programs. State of the Art in Global Optimization:139
46. Viwanathan J, Grossmann IE (April 1990) DICOPT ++: A program for mixed integer nonlinear optimization, user's guide. Manual Engin Design Res Center Carnegie-Mellon Univ

Bilevel Programming

PATRICE MARCOTTE[1], GILLES SAVARD[2]
[1] University Montréal, Montréal, Canada
[2] École Polytechnique, Montréal, Canada

MSC2000: 49M37, 90C26, 91A10

Article Outline

Keywords
See also
References

Keywords

Nonconvex optimization; Equilibrium; Game theory

Let us consider a sequential game where the first player ('leader') incorporates into his optimization process the optimal reaction vector y of the second player ('follower') to the leader's decision vector x. This situation is described mathematically by the *bilevel program*

$$\text{BLP} \quad \begin{cases} \min\limits_{x,y} & f(x,y) \\ \text{s.t.} & (x,y) \in X \\ & y \in \operatorname{argmin}\limits_{y' \in Y(x)} g(x,y'), \end{cases}$$

where it is understood that the leader is requested to select a vector x such that the parameterized set $Y(x)$ is nonempty.

This formulation is extremely general in that it subsumes linear *zero-one optimization*, *quadratic concave programming*, disjoint *bilinear programming*, *nonlinear complementarity*, etc. If one denotes by $y(x)$ the set of optimal answers to a given leader vector x, the above bilevel program can be recast as the 'standard' mathematical program

$$\begin{cases} \min\limits_{x,y} & f(x,y) \\ \text{s.t.} & (x,y) \in X \\ & y \in y(x). \end{cases}$$

The *induced region* of a bilevel program is defined as the feasible set of the above program. This set is usually nonconvex and might be disconnected. It is implicit that, whenever $y(x)$ is not a singleton, the leader is free to select that element $y \in y(x)$ that suits him best. This interpretation is legitimate in the case where side payments are allowed, i. e., the leader can bias the follower's objective in his favor. On the other hand, the behavior of a risk-averse leader which seeks to minimize, over the feasible set X, the objective

$$\max_{y \in y(x)} f(x,y).$$

has been considered in [4].

The algorithmic difficulty of bilevel programming stems mainly from the fact that the set $y(x)$ is ill-behaved, and usually not available in closed form. To gain some insight into this difficulty, let us consider the 'simple' situation where f, g are affine, the constraint $(x, y) \in X$ is absent and $Y(x) = \{y: Ax + By \geq b\}$ is a convex polyhedron. It is easy to show that, as in lin-

ear programming, bounded and feasible linear bilevel programs admit extremal solutions, hence the linear BLP lies in the class *NP* of problems polynomially solvable by a nondeterministic algorithm. Unfortunately, as shown in [2] and [3], the linear BLP is also strongly *NP*-hard. Moreover, its optimal solution(s) need not even be efficient ('Pareto optimal'). This is one of the features that distinguish bilevel programming from bicriterion optimization. Indeed consider the linear BLP illustrated in the figure below, where the arrows denote the players' respective steepest descent directions:

$$\begin{cases} \min\limits_{x,y} & \frac{1}{2}x + y \\ \text{s.t.} & y \in \operatorname{argmin}\limits_{y'} -y' \\ & x + y' \le 1 \\ & x, y' \ge 0. \end{cases}$$

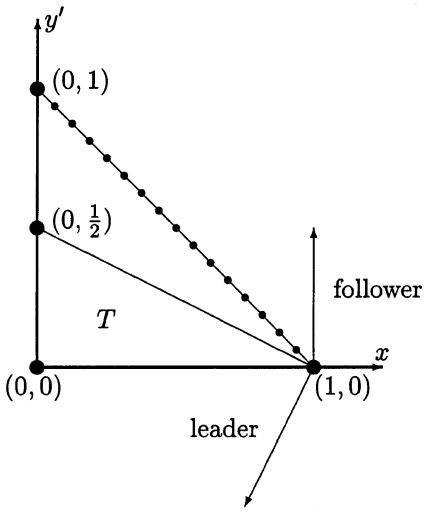
The induced region of this problem reduces to the dotted line segment of Fig. 1. Optimizing over this line segment yields the solution $(x, y) = (1, 0)$, which is strictly dominated by all points inside the triangle T with vertices (0, 1/2), (0, 0) and (1, 0). Since the set of efficient points is the segment $[(0, 0), (0, 1)]$, the only rational point that is also Pareto optimal is (0, 1), which is actually the worst possible outcome for the leader. Note that, in the case where the functions f, g are affine and the sets X, $Y(x)$ are polyhedral, the induced region is in general a nonconvex piecewise linear variety that contains several local minima.

Assume now that the following conditions are satisfied:

- $Y(x) = \{y: h_i(x, y) \le 0, 1 \le i \le n\}$;
- the functions g and h_i are continuously differentiable and convex;
- the set $Y(x)$ is regular for every x, i. e., some constraint qualification holds.

Then one can substitute for the follower's program its Kuhn–Tucker conditions, yielding the equivalent single-level program

$$\begin{cases} \min\limits_{(x,y)\in X} & f(x, y) \\ \text{s.t.} & \nabla_y g(x, y) + \sum\limits_{1\le i\le n} \lambda_i \nabla_y h_i(x, y) = 0 \\ & \lambda_i h_i(x, y) = 0, \quad 1 \le i \le n, \\ & \lambda_i \ge 0, \quad 1 \le i \le n. \end{cases}$$



Bilevel Programming, Figure 1

The complementarity constraints make this single-level problem difficult both theoretically (the constraint set is almost never regular) and algorithmically.

A useful variant of BLP occurs when $y(x)$ corresponds to the solution of an equilibrium system parameterized in x. If this system is modeled by means of a *variational inequality*, one obtains a generalized Kuhn–Tucker formulation where the gradient $\nabla_y g(x, y)$ is replaced by a function $F(x, y)$. (See [5].) In both cases, the complementarity constraint can be incorporated in the leader's objective as a penalty term $M \sum_{1 \le i \le n} \lambda_i h_i(x, y)$, thus greatly simplifying the constraint set. (It even becomes polyhedral in the linear case.) Under suitable assumptions, and for large but finite values of the penalty multiplier M, the penalized problem is equivalent to the original bilevel problem.

See also

- ▶ Bilevel Fractional Programming
- ▶ Bilevel Linear Programming
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Bilevel Programming: Applications
- ▶ Bilevel Programming: Applications in Engineering
- ▶ Bilevel Programming: Implicit Function Approach

- ► Bilevel Programming: Introduction, History and Overview
- ► Bilevel Programming in Management
- ► Bilevel Programming: Optimality Conditions and Duality
- ► Multilevel Methods for Optimal Design
- ► Multilevel Optimization in Mechanics
- ► Stochastic Bilevel Programs

References

1. Bard JF (1998) Practical bilevel optimization: Algorithms and applications. Kluwer, Dordrecht
2. Hansen P, Jaumard B, Savard G (1992) New branch-and-bound rules for linear bilevel programming. SIAM J Sci Statist Comput 13:1194–1217
3. Jeroslow RG (1985) The polynomial hierarchy and a simple model for competitive analysis. Math Program 32:146–164
4. Loridan P, Morgan J (1996) Weak via strong Stackelberg problem: new results. J Global Optim 8:263–287 ISSN: 0925-5001
5. Luo Z-Q, Pang J-S, Ralph D (1996) Mathematical programs with equilibrium constraints. Cambridge Univ. Press, Cambridge
6. Shimizu K, Ishizuka Y, Bard JF (1997) Nondifferentiable and two-level programming. Kluwer, Dordrecht

Bilevel Programming: Applications

PATRICE MARCOTTE[1], GILLES SAVARD[2]

[1] University Montréal, Montréal, Canada

[2] École Polytechnique, Montréal, Canada

MSC2000: 91B99, 90C90, 91A65

Article Outline

Keywords
See also
References

Keywords

Network design; Economic equilibrium; Energy; Principal agent

Bilevel programming (see ► Bilevel programming: Introduction, history and overview; ► Bilevel programming) is ideally suited to model situations where the decision maker does not have full control over all decision variables. Five such situations are described in this article.

Example 1 The first example involves the improvement of a road network through either capacity expansion, traffic signals synchronization, vehicle guidance systems, etc. While management may be assumed to control the design variables, it can only affect indirectly the travel choices of the network users. Let x denote the design vector, y the flow vector, X the set of feasible design variables and $c_i(x, y)$ the travel delay along link i. One wishes to minimize over the set X the system travel cost $\sum_i y_i\, c_i(x, y)$, where the vector y is required to be an equilibrium *traffic assignment* corresponding to the design vector x. Neglecting the latter equilibrium requirement could lead to suboptimal policies. However, as shown in [5] for a continuous variant of the network design problem, efficient heuristic procedures can generate near-optimal solutions at a low computational cost. Indeed it is in the interest of both the management and the network users to minimize travel delays, although the former is interested in minimizing total travel time, while the users optimize their own travel time.

Example 2 Next consider the maximization of revenues raised from tolls set on a transportation network. If tolls are set too high, traffic on the corresponding arcs will drop and revenues will be affected negatively. Conversely, low toll values will generate low revenues. One could strike the right balance by maximizing total revenue, subject to the network users y achieving an equilibrium with respect to the toll vector x. In the case where the network is uncongested, users are assigned to shortest paths linking their respective origin and destination. This yields the bilevel program with bilinear objectives

$$\begin{cases} \max\limits_{x,y} & \sum\limits_{i\in I_1} x_i y_i \\ \text{s.t.} & y \in \operatorname{argmin}\limits_{y'\in Y} \sum\limits_{i\in I_1}(c_i + x_i)y'_i + \sum\limits_{i\in I_2} c_i y'_i, \end{cases}$$

where I_1 represents the set of toll arcs, I_2 the set of toll-free arcs, and Y the polyhedron of demand-feasible flow vectors. In [4] it has been shown that this problem is reducible to a linear bilevel program with an economic interpretation in terms of 'second-best' choices, and can

also be reformulated as a *zero-one integer program* with few binary variables. Special cases are amenable to polynomial algorithms.

Example 3 The third example is the *Stackelberg–Nash–Cournot equilibrium* studied in [8] where the leader firm maximizes its revenue $x \cdot p(x + \sum_{1 \le i \le n} y_i) - c(x)$ (p denotes the inverse demand function and c the leader firm's production cost), subject to the vector y being a Cournot–Nash equilibrium with respect to the shifted inverse demand function $p_x(Q) = p(x + Q)$. This model subsumes the situations of monopoly ($n = 0$) as well as that of Stackelberg equilibrium ($n = 1$). It has been extended in [7] to the case of multiple leaders, but does not fit any more the framework of bilevel programming.

Example 4 A fourth example is provided by the energy sector, which is characterized by an extensive use of large scale techno-economic models describing specific subsectors or markets: gas and electricity subsectors, industrial and residential markets, etc. In this respect, it provides a rich source for bilevel models. A bilevel program arises when a utility, in its strategic planning process, takes explicitly into account the rational reaction of its competitors or customers to its own investment schedule. This approach has been applied to assess the impact of new demand management technologies for reducing power usage [3]. Another bilevel model arises when a utility is legally bound to buy any energy surplus from 'qualified small producers' at marginal cost. For example, a study of the impact of cogeneration in the pulp and paper industry on the electricity market has been conducted in [2].

Example 5 Finally we mention that bilevel programming subsumes the principal/agent paradigm of economics (see [1]), where the principal (leader) subcontracts a job to an agent (follower). The principal rewards the agent according to the quality of the final outcome ω, which may be random, while the agent maximizes its own objective, which is a function of the effort level y and the expected reward $x(\omega(y))$. The lower the effort level y, the lower the (expected) quality $\omega(y)$ of the finished job. Assuming that the agent accepts to perform the job only if his utility is larger than some 'reservation level' $g_{\min}$, one derives the bilevel program:

$$\begin{cases} \max\limits_{x(\cdot),y} & f(x(\omega(y)), \omega(y)) \\ \text{s.t.} & g(x(\omega(y)), y) \ge g_{\min} \\ & y \in \operatorname{argmax}\limits_{y' \in Y} g(x(\omega(y')), y'), \end{cases}$$

where the leader's decision variable x is a function defined over the set Y of possible effort levels. Whenever the set Y is not finite, this yields an infinite-dimensional optimization problem. The situation becomes all the more complex when the output ω is a random variable of the agent's effort y.

See also

- ▶ Bilevel Fractional Programming
- ▶ Bilevel Linear Programming
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Bilevel Programming
- ▶ Bilevel Programming: Applications in Engineering
- ▶ Bilevel Programming: Implicit Function Approach
- ▶ Bilevel Programming: Introduction, History and Overview
- ▶ Bilevel Programming in Management
- ▶ Bilevel Programming: Optimality Conditions and Duality
- ▶ Multilevel Methods for Optimal Design
- ▶ Multilevel Optimization in Mechanics
- ▶ Stochastic Bilevel Programs

References

1. van Ackere A (1993) The principal/agent paradigm: its relevance to various functional fields. Europ J Oper Res 70:83–103
2. Haurie A, Loulou R, Savard G (1992) A two player game model of power cogeneration in New England. IEEE Trans Autom Control 37:1451–1456
3. Hobbs BF, Nelson SK (1992) A non-linear bilevel model for analysis of electric utility demand-side planning issues. Ann Oper Res 34:255–274
4. Labbé M, Marcotte P, Savard G (1998) A bilevel model of taxation and its application to optimal highway pricing. Managem Sci 44:1608–1622
5. Marcotte P (1986) Network design with congestion effects: A case of bilevel programming. Math Program 34:142–162

6. Migdalas A, Pardalos PM, Värbrand P (1998) Multilevel optimization: Algorithms and applications. Kluwer, Dordrecht
7. Sherali HD (1984) A multiple leader Stackelberg model and analysis. Oper Res 32:390–404
8. Sherali HD, Soyster AL, Murphy FH (1983) Stackelberg–Nash–Cournot equilibria: characterizations and computations. Oper Res 31:253–276

Bilevel Programming: Applications in Engineering

ZEYNEP H. GÜMÜŞ[1,2], KEMAL SAHIN[3], AMY CIRIC[4]
[1] Department of Physiology and Biophysics, Weill Medical College, Cornell University, New York, USA
[2] The HRH Prince Alwaleed Bin Talal Bin Abdulaziz Alsaud Institute for Computational Biomedicine, Weill Medical College, Cornell University, New York, USA
[3] Huntsman (Germany) GmbH, Deggendorf, Germany
[4] Department of Chemical and Materials Engineering, University of Dayton, Dayton, USA

Article Outline

Keywords and Phrases
Introduction
Bilevel Programming in Traffic Management
Bilevel Programming in Chemical Process Synthesis
Bilevel Programming in Metabolic Engineering
Conclusions
References

Keywords and Phrases

Bilevel programming; Traffic control; Phase equilibrium; Metabolic engineering; Design

Introduction

Bilevel programming problems (BLPP) are encountered when one optimization problem is embedded within another one as a constraint. BLPPs arise in many areas of engineering, where hierarchical decision models are often encountered. Almost all areas of engineering can provide some examples in which two decision models interact and the outcome of one decision influences another; applications can be found in areas as diverse as traffic control and reactive distillation.

The general BLPP formulation is as follows:

$$\text{Outer optimization problem} \begin{cases} \min\limits_{x} F(\mathbf{x},\mathbf{y}) \\ \text{s.t. } \mathbf{G}(\mathbf{x},\mathbf{y}) \geq 0 \\ \quad \mathbf{H}(\mathbf{x},\mathbf{y}) = 0 \\ \text{inner optimiziation problem} \begin{cases} \min\limits_{y} f(\mathbf{x},\mathbf{y}) \\ \text{s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \geq 0 \\ \quad \mathbf{h}(\mathbf{x},\mathbf{y}) = 0 \end{cases} \\ \mathbf{x} \in X \subset \Re^{n_1}, \quad \mathbf{y} \in Y \subset \Re^{n_2} \end{cases}$$

where

$$f, F\colon \Re^{n_1} \times \Re^{n_2} \to \Re,$$
$$\mathbf{g} = [g_1, .., g_J]\colon \Re^{n_1} \times \Re^{n_2} \to \Re^{J},$$
$$\mathbf{G} = [G_1, \ldots, G_{J'}]\colon \Re^{n_1} \times \Re^{n_2} \to \Re^{J'},$$
$$\mathbf{h} = [h_1, \ldots, h_I]\colon \Re^{n_1} \times \Re^{n_2} \to \Re^{I},$$
$$\mathbf{H} = [H_1, \ldots, H_{I'}]\colon \Re^{n_1} \times \Re^{n_2} \to \Re^{I'}.$$

The outer optimization problem, which minimizes $F(\mathbf{x},\mathbf{y})$, is constrained by inequality constraints $\mathbf{G}$, equality constraints $\mathbf{H}$, and the inner optimization problem. This inner optimization minimizes its objective function by varying $\mathbf{y}$, while subject to its own inner constraints $\mathbf{g}$ and $\mathbf{h}$. The inner variables $\mathbf{y}$ may also appear in the outer constraints and objective function, and the inner constraints and objective function may be parameterized by $\mathbf{x}$. Novel global optimization strategies exist to solve the BLPP with twice continuously differentiable nonconvex nonlinear [18] and mixed-integer nonlinear constraints [19].

This article explores a diverse sampling of bilevel programming including examples from civil engineering traffic management, chemical engineering process design and metabolic engineering.

Bilevel Programming in Traffic Management

As urban populations increase and cities expand, traffic and its related problems effect the everyday life of all commuters.

Traffic problems follow the hierarchical structure of BLPPs. Each individual commuter travels upon a network of roads that is created and organized by a central regulatory agency. This agency plans the layout and carrying capacity of highways and streets, chooses where to place on-off ramps that connect limited access highways to local streets, decides where to install traffic lights, and sets their signaling rate.

As the regulatory agency plans and manages this network of roads, it must accommodate the traffic pattern formed by the individual decisions of the thousands of travelers who use the network each day. During each trip, each traveler takes a path that she believes will minimize her travel time, based on previous experience and ongoing traffic reports. Beckmann et al. [2] have shown that when this information is perfect and all travelers have access to it, the cumulative effect is to minimize the total time spent by all drivers on all roads in the network:

$$\min_{v} \sum_{a} \int_{o}^{v_a} t_a(x)\,\mathrm{d}x\,.$$

This behavior by the travelling public creates bilevel programming problems in traffic management. When a regulatory agency tries to set policies that minimize gas consumption, the travel time of all drivers, or some other objective, its options are constrained by the response of the travelling public.

One application of BLPPs in traffic is signal optimization, where the objective is to minimize travel time or gasoline consumption by varying the length of green lights and the cycle time of traffic lights [9]:

$$\begin{aligned} &\min \sum_{a} t_a v_l(t,s) \\ &\text{s.t } \min_{v} \sum_{a} \int_{o}^{v_a} t_a(x)\,\mathrm{d}x\,. \end{aligned}$$

In this problem, the outer objective sums over all costs based on the signaling policies.

This bilevel programming problem is used to plan road improvements, where central planning agency minimizes the cost of construction and similar activities, subject to the inner optimization problem that predicts the behavior of traffic on the road [1,9,21]. It is also used to optimize the flow of traffic onto limited access highways. The outer problem minimizes the total travel time of all travelers by optimizing traffic light lengths and other controls at the on- and off-ramps of the highway, while the inner optimization problem predicts the behavior of traffic on the road [24,25,26].

Bilevel Programming in Chemical Process Synthesis

Chemical process synthesis by optimization techniques is a vast area that includes plant design, the synthesis of reactor networks, separation systems, heat exchanger networks, and utility plants, and the planning of batch and multiperiod operations [3,10,11,12].

Inner Problems that Minimize the Gibbs Free Energy Many chemical engineering design problems involve distillation columns, liquid-liquid extractors and decanters, and reactors; modeling these unit operations usually requires modeling the chemical equilibria and phase equilibria (vapor-liquid equilibrium, liquid-liquid equilibrium, and vapor-liquid-liquid equilibrium) occurring within them. When the number of phases is known in advance, phase and chemical equilibrium can be modeled with a set of algebraic equations. When, however, the number of phases is not known *a priori*, these algebraic equations cannot be used; in these problems, the number of phases, phase equilibrium, and in some problems chemical equilibrium can be predicted by minimizing the Gibbs free energy. Maximizing the profit, minimizing the cost, or optimizing some other measure of a chemical process that contains a unit operation with an unknown number of phases is a bilevel programming problem:

$$\begin{aligned} &\max F(\mathbf{x}, n_{ik}) \\ &\text{s.t } \mathbf{G}(\mathbf{x}, n_{ik}) \geq 0 \\ &\quad \mathbf{H}(\mathbf{x}, n_{ik}) = 0 \\ &\quad \min_{n_{ik}} \sum_{i} \sum_{k} n_{ik}\mu_{ik} \\ &\quad \text{s.t. } \sum_{ik} a_{ij}\, n_{ik} = b_j, \quad \forall j \in E \\ &\qquad n_{\mathrm{ik}} \geq \delta, \quad i = 1,\ldots,\mathrm{NC}; \quad k = 1,\ldots,\mathrm{NP}\,. \end{aligned}$$

Here, the outer problem maximizes the profit F. Design specifications are captured by inequality constraints $\mathbf{G}$, while equality constraints $\mathbf{H}$ are the mass

and energy balances. The inner optimization minimizes the Gibbs free energy, equal to the summation of $n_{ik}\mu_{ik}$, the moles of species i and phase k multiplied by the corresponding chemical potential. This inner problem is constrained by mass balances assuring that the total number of atoms of element j is constant regardless of the phase or chemical distribution, and that the total number of moles of species i in phase k is positive.

Clark and Westerberg [7,8] used this strategy to optimize a reactor making aniline from nitrobenzene and hydrogen. The reaction also produces water, which may form a two-liquid phase mixture with nitrobenzene and aniline, depending upon the relative amounts of nitrobenzene, aniline, and water. The outer problem optimized the reactor temperature and pressure, while the inner problem found the simultaneous phase and chemical equilibrium by minimizing the Gibbs Free Energy.

Gümüş and Ciric [17] used bilevel programming to optimize a reactive distillation column that produces aniline from nitrobenzene and water. The outer problem minimizes cost by varying the number of trays, reflux and reboil ratios, and feed tray locations. A series of inner optimization problems predict the phase and chemical equilibrium in the condenser and on each tray in the column.

Bilevel Programming and Simultaneous Design and Control Bilevel programming has also been used to integrate the design of a chemical process with the synthesis of its control scheme [4]. The outer optimization problem maximizes the annual profit $D(z)$ minus the cost of off-spec product formed during process upsets, while an inner optimization problem simultaneously predicts the amount of off-spec product formed during process upsets and finds the settings of a model predictive controller that minimize this amount. The model is:

$$
\begin{aligned}
&\max D(z)\\
&-\kappa \sum_{l} CO_l^p \{z, p; u_l(t), x_l(t), y_l(t), p_c(t)\} - C_H\\
&\text{s.t } f(z, p) = 0\\
&\quad h(z, p) \;= 0\\
&\quad g(z, p) \;\geq 0\\
&\quad g_{h.l}^p(z, p, u_l(t), x_l(t), y_l(t), p_c(t)) \geq 0\\
&\min_{n_{ik}} CO_l \{z, p; u_l(t), x_l(t), y_l(t), p_c(t)\}\\
&\text{s.t. } \dot{x} = f(z, p, u_l(t), x_l(t), y_l(t), p_c(t))\\
&\quad x(t=0) = x_d\\
&\quad y(t=0) = y_d\\
&\quad u(t=0) = u_d\\
&\quad g_{h.l}(z, p, u_l(t), x_l(t), y_l(t), p_c(t)) \geq 0\\
&\quad h(z, p, u_l(t), x_l(t), y_l(t), p_c(t)) = 0\\
&\quad u^{\mathrm{L}} \leq u(t) \leq u^{H}.
\end{aligned}
$$

In this formulation, the cost of the fluctuations around the steady state, denoted by subscript d, will increase the cost of off spec production, CO_l^p. In the inner optimization, the actions u of a model predictive controller are based on the disturbance l.

Bilevel Programming and Design Under Uncertainty In the planning stage of a design, the range of uncertain parameters that the design can tolerate for feasible operation should be determined. The design under parametric uncertainty problem can be described by a set of equality constraints I and inequality constraints J representing plant operation and design specifications:

$$
\begin{aligned}
&h_{\mathrm{i}}(\mathbf{d}, \mathbf{z}, \mathbf{x}, \boldsymbol{\theta}) = 0, \quad \forall i \in I\\
&g_{\mathrm{j}}(\mathbf{d}, \mathbf{z}, \mathbf{x}, \boldsymbol{\theta}) \geq 0, \quad \forall j \in J
\end{aligned}
$$

where $\mathbf{z}$ is the vector of control variables, $\mathbf{x}$ is the vector of state variables and $\boldsymbol{\theta}$ is the vector of uncertain parameters. Feasibility concerns are incorporated into the design step by quantifying design feasibility and flexibility with the feasibility test and flexibility index measures. These measures are characterized by max-min-max formulations [20] that are further reformulated in the BLPP form [13,16,23]. For a specific design $\mathbf{d}$, the BLPP feasibility test problem is of the form:

$$
\begin{aligned}
&\max_{\boldsymbol{\theta} \in T} \psi(\mathbf{d}, \boldsymbol{\theta})\\
&\text{s.t } \psi(\mathbf{d}, \boldsymbol{\theta}) \leq 0\\
&\quad \psi(\mathbf{d}, \boldsymbol{\theta}) = \min_{z,u} u
\end{aligned}
$$

$$
\begin{aligned}
&\text{s.t. } h_i(\mathbf{d},\mathbf{z},\mathbf{x},\boldsymbol{\theta}) = 0, \quad \forall i \in I \\
&\quad u - g_j(\mathbf{d},\mathbf{z},\mathbf{x},\boldsymbol{\theta}) \geq 0, \quad \forall j \in J \\
&T = \left\{\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^L \leq \boldsymbol{\theta} \leq \boldsymbol{\theta}^U\right\}
\end{aligned}
$$

where the function $\psi(\mathbf{d},\boldsymbol{\theta})$ represents a feasibility measure for design $\mathbf{d}$. The boundary of the feasible region in the space of the uncertain variables is at $\psi(\mathbf{d},\boldsymbol{\theta}) = 0$. If $\psi(\mathbf{d},\boldsymbol{\theta}) \geq 0$, the design can not operate at least for some values of $\boldsymbol{\theta}$ in T, and the BLPP is infeasible.

For a specific design $\mathbf{d}$, the design flexibility test problem is also formulated as a BLPP:

$$
\begin{aligned}
&\min_{\boldsymbol{\theta}\in T} \delta \\
&\text{s.t } \psi(\mathbf{d},\boldsymbol{\theta}) = 0 \\
&\quad \psi(\mathbf{d},\boldsymbol{\theta}) = \min_{z} u \\
&\quad \text{s.t. } h_i(\mathbf{d},\mathbf{z},\mathbf{x},\boldsymbol{\theta}) = 0, \quad \forall i \in I \\
&\qquad g_j(\mathbf{d},\mathbf{z},\mathbf{x},\boldsymbol{\theta}) \leq u, \quad \forall j \in J \\
&\quad T(\delta) = \left\{\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^{\mathrm{N}} - \delta\Delta\boldsymbol{\theta}^{-} \leq \boldsymbol{\theta} \leq \boldsymbol{\theta}^{\mathrm{N}} + \delta\Delta\boldsymbol{\theta}^{+}\right\} \\
&\quad \delta \geq 0
\end{aligned}
$$

where δ is the largest scaled deviation of any expected deviations $\Delta\boldsymbol{\theta}^{-}$ and $\Delta\boldsymbol{\theta}^{+}$ the design can handle [13,16,23]. Higher δ signifies more flexible design towards parametric variations.

Bilevel Programming in Metabolic Engineering

Metabolic engineering involves optimization of genetic and regulatory processes within cells to increase overproduction of desired metabolites or proteins. These changes can have major effects on cell growth if the desired overproduction competes with growth resources, so the cell will redistribute the metabolic fluxes to maximize its growth rate. Metabolic flux distributions can be optimized utilizing in-silico genome scale metabolic network maps to develop overproduction strategies. Several different problems in this research area have recently been formulated as BLPPs. These involve the (i) determination of optimal gene knockouts, (ii) identification of stable steady state solutions and (iii) dynamic gene expression control strategies, all to achieve maximum product yield.

Gene Knockout Strategies Gene deletion strategies to increase the overproduction of a desired product can be straightforward and involve competing reaction pathways; however, many others can be complex and non-intuitive. Burgard et al. [5] introduced a BLPP formulation to address the optimal manipulation of gene knockout strategies to maximize overproduction, subject to maximizing cell's growth objective at the inner level. The inner problem is parameterized with gene knockout strategies that are chosen by the outer problem and constrained by metabolic flux balances and fixed substrate. This BLPP model for a steady-state metabolic network of N metabolites and M metabolic reactions fueled by a glucose substrate is formulated as:

$$
\begin{aligned}
&\max_{y_j} v_{\text{chemical}} \\
&\text{s.t } \max_{v_j} v_{\text{biomass}} \\
&\quad \text{s.t. } \sum_{j=1}^{M} S_{ij} v_j = 0, \quad \forall i \in N \\
&\qquad v_{\text{pts}} + v_{\text{glk}} - v_{\text{glc_uptake}} = 0 \\
&\qquad v_{\text{atp}} - v_{\text{atp_main}} \geq 0 \\
&\qquad v_{\text{biomass}} - v_{\text{biomass}}^{\text{target}} \geq 0 \\
&\qquad v_j^{\min} \cdot y_j \leq v_j \leq v_j^{\max} \cdot y_j, \quad \forall j \in M \\
&\qquad y_j = \{0,1\}, \quad \forall j \in M \\
&\qquad \sum_{j\in M} (1 - y_j) \leq K
\end{aligned}
$$

where v_{chemical} is the flux of the desired product, v_{biomass} is biomass formation, S_{ij} is the stoichiometric constant for metabolite i in reaction j, v_j is the flux of reaction j, v_{pts} and v_{glk} respectively represent the uptake of glucose through the phosphotransferase system and glucokinase, $V_{\text{glk_uptake}}$ is the basis glucose uptake scenario, $v_{\text{atp_main}}$ is the non-growth associated ATP maintenance requirement, K is the number of allowable knockouts, and $v_{\text{biomass}}^{\text{target}}$ is a minimum level of biomass production. The BLPP can be modified further to include additional bounds on O_2, CO_2 and NH_3 transport rates and secretion pathways for key metabolites in the inner problem [22].

Stable Metabolic Networks Stability considerations of a redesigned metabolic network can be addressed within a BLPP framework, such that the new system is stable around a neighborhood of the new steady state.

Here, the outer problem maximum product flux objective is subject to flux balances and an inner stability objective [6].

Temporal Flux Control Gene expression can be controlled dynamically using the BLPP structure to optimize the temporal flux profile of a key reaction, such that at the end of a batch, the total product formation is maximized [14]. In the outer problem, a flux in a specific reaction known to have an impact on the product formation is varied with time to maximize the total product formation at the end of a batch. The inner problem maximizes cellular growth at each sampling time over the batch period by optimizing the remaining fluxes. The BLPP can be modified to determine the optimal regulation time of the specific flux from an initial to a final value. Glycerol and ethanol production in *E. coli* have been studied using the BLPP formulation [14].

Gadkar et al. [15] coupled this BLPP model with control algorithms to determine genetic manipulation strategies in bioprocess applications. They introduced three alternative BLPP models to maximize ethanol production in anaerobic batch fermentation of *E. coli*, optimizing ethanol production, batch time and multibatch scheduling in the presence of parametric uncertainty and measurement noise. These include (i) optimizing growth regulation time and batch duration time by penalizing for longer batch times in the outer objective (ii) scheduling multiple batch runs to address inhibition due to product accumulation in the reactor, optimizing the number of batch runs, batch duration times, glucose allocation per run and the manipulated flux regulation time, and (iii) optimizing genetic alterations in the presence of growth inhibition and parametric uncertainty in the inhibition constant.

Conclusions

The hierarchical structure of many engineering problems lends themselves to bilevel programming formulations, where an inner optimization problem constrains a larger, 'outer' optimization problem. Applications in civil engineering design include traffic control, where an inner optimization problem predicting driver's behavior constrains an outer optimization problem that identifies the optimal control strategies. In chemical engineering, BLPPs are used to identify processes that are both economically optimal – maximizing revenue or minimizing cost – and simultaneously ensure that multiphase equilibrium is satisfied by determining the global minimum of Gibbs Free energy. Other applications include the combined optimization of a chemical process and it's controllers and chemical process design under parametric uncertainty, to ensure operational feasibility and flexibility. Alternative BLPP formulations have been introduced in modeling metabolic engineering systems. These address the maximization of product yield by determining optimal gene knockouts, identifying stable steady state solutions and dynamic gene expression control strategies. Metabolic engineering area is a recent and growing application field for BLPP.

References

1. Ben-Ayed O, Boyce DE, Blair III. CE (1988) A general bilevel linear programming formulations of the network Design problem. Transpn Res B 22B:311–318
2. Beckmann MJ, McGuire CB, Winston C (1956) Studies in the economics of transportation. Yale University Press, New Haven, CT
3. Biegler LT, Grossmann IE, Westerberg AW (1997) Systematic methods of chemical process design. Prentice-Hall, New Jersey
4. Brengel DD, Seider WD (1992) Coordinated design and control optimization of nonlinear processes. Comput Chem Eng 16:861–886
5. Burgard AP, Pharkya P, Maranas C (2003) OptKnock: A bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. Biotech Bioeng 84:647–657
6. Chang YJ, Sahinidis NV (2005) Optimization of metabolic pathways under stability considerations. Comput Chem Eng 29:467–479
7. Clark PA, Westerberg A (1990) Bilevel programming for chemical process design – I. Fundamentals and algorithms. Comput Chem Eng 14:87–97
8. Clark PA (1990) Bilevel programming for chemical process design – II. Performance study for nondegenerate problems. Comput Chem Eng 14:99–109
9. Fisk CS (1984) Game theory andr transportation systems modeling. Transp Res-B 18B:301–313
10. Floudas CA (1995) Nonlinear and mixed-integer optimization. Oxford University Press, USA
11. Floudas CA, Pardalos PM, Adjiman CS, Esposito WR, Gümüş ZH, Harding ST, Klepeis JL, Meyer CA, Schweiger CA (1999) Handbook of test problems in local and global optimization. Kluwer, Netherlands

12. Floudas CA (2000) Deterministic global optimization: theory, methods and applications. Kluwer, The Netherlands
13. Floudas CA, Gümüş ZH, Ierapetritou MG (2001) Global optimization in design under uncertainty: feasibility test and flexibility index problems. Ind Eng Chem Res 40:4267–4282
14. Gadkar KG, Doyle III. FJ, Edwards JS, Mahadevan R (2005) Estimating optimal profiles of genetic alterations using constraint-based models. Biotech Bioeng 89:243–251
15. Gadkar KG, Mahadevan R, Doyle III. FJ (2006) Optimal genetic manipulations in batch bioreactor control. Automatica 42:1723–1733
16. Grossmann IE, Floudas CA (1987) Active constraint strategy for flexible analysis in chemical processes. Comput Chem Eng 11:675–693
17. Gümüş ZH, Ciric AR (1997) Reactive distillation column design with vapor/liquid/liquid equilibria. Comput Chem Eng 21:983–988
18. Gümüş ZH, Floudas CA (2001) Global optimization of nonlinear bilevel programming problems. J Glob Optim 2:1–31
19. Gümüş ZH, Floudas CA (2005) Global optimization of mixed-integer bilevel programming problems. Comput Man Sci 2:181–212
20. Halemane KP, Grossmann IE (1983) Optimal process design under uncertainty. AIChE J 29:425–433
21. LeBlanc LJ, Boyce DE (1986) A bilevel programming algorithm for exact solution of the network design problem with user optimal flows. Transp Res-B 20B:259–265
22. Pharkya P, Burgard AP, Maranas C (2003) Exploring the overproduction of amino acids using the bilevel optimization framework optknock. Biotech Bioeng 84:887–899
23. Swaney RE, Grossmann IE (1985) An index for operational flexibility in chemical process design. Part I: Formulation and theory. AIChE J 31:621–630
24. Yang H, Yagar S, Iida Y, Asakura I (1994) An algorithm for the inflow control problem on urban freeway networks with user-optimal flows. Transp Res-B 28B:123–139
25. Yang H, Yagar S (1994) Traffic assignment and traffic control in general freeway-arterial corridor systems. Transp Res-B 28B:463–486
26. Yang H, Yagar S (1995) Traffic assignment and signal control in saturated roadnetworks. Transp Res-A 29A:125–139

Bilevel Programming Framework for Enterprise-Wide Process Networks Under Uncertainty

Efstratios N. Pistikopoulos, Nuno P. Faísca, Pedro M. Saraiva, Berç Rustem
Centre for Process Systems Engineering, Imperial College London, London, UK

Article Outline

Introduction
Formulation
  Bilevel Programming
  Bilevel Programming with Multi-Followers
Applications
Cases
  Global Optimum of a Bilevel Programming Problem
  Bilevel Programming Problem
  Bilevel Programming Problem with Multi-Followers
  Bilevel Programming with Uncertainty
References

Introduction

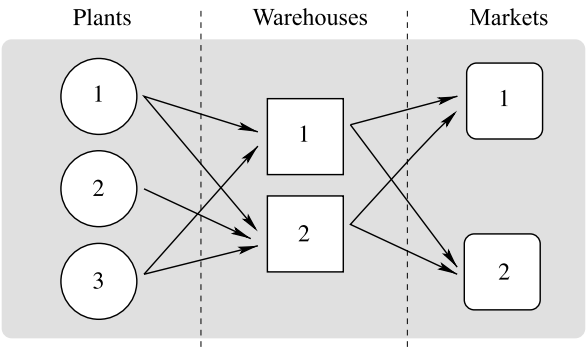
Optimisation of enterprise-wide process networks has attracted considerable attention in recent years; since it represents substantial economic savings there has been a growing concern to plan efficiently the operations within the complexity of decision networks. Often, in such complex networks, an hierarchy of decisions has to be followed and compromises made between identities with equivalent authority. For instance, numerous investigations have been done in the optimisation of supply chains, Fig. 1, and in the plant selection problem, Fig. 2. A detailed study of hierarchical decisions can be found in [13,14,26].

Formulation

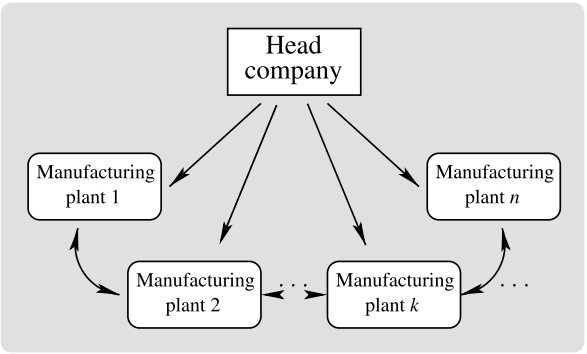
The general multilevel decentralised optimisation problem can be described as follows:

$$\begin{aligned}
&\min_{x,y_1^i,y_2^k,\ldots,y_m^l} f_1(x,y_1^i,y_2^k,\ldots,y_m^l)\,, \quad (1st\ level)\\
&\text{s.t. } g_1(x,y_1^i,y_2^k,\ldots,y_m^l)\le 0\,,\\
&\quad \text{where } \left[y_1^i,y_2^k,\ldots,y_m^l\right] \text{ solve}\,,
\end{aligned} \tag{1}$$

$$\begin{aligned}
\ldots,\ &\min_{y_1^i,y_2^k,\ldots,y_m^l} f_2^i(x,y_1^i,y_2^k,\ldots,y_m^l),\ldots \quad (2nd\ level)\\
&\text{s.t. } g_2^i(x,y_1^i,y_2^k,\ldots,y_m^l)\,,\ \le 0,\\
&\quad \text{where } \left[y_2^k,\ldots,y_m^l\right] \text{ solve},\\
&\quad \vdots\\
&\quad \ldots,\min_{y_m^l} f_m^l(x,y_1^i,y_2^k,\ldots,y_m^l),\ldots \quad (mth\ level)\\
&\quad \text{s.t. } g_m^l(x,y_1^i,y_2^k,\ldots,y_m^l)\le 0\,,
\end{aligned}$$



Bilevel Programming Framework for Enterprise-Wide Process Networks Under Uncertainty, Figure 1
Supply chain planning example



Bilevel Programming Framework for Enterprise-Wide Process Networks Under Uncertainty, Figure 2
Hierarchical decision planning example

where, f are real convex functions, g are vectorial real functions defining convex sets and x, y are sets of variables belonging to the group of real numbers; $i \in \{1, 2, \ldots, I\}$, $k \in \{1, 2, \ldots, K\}$, $l \in \{1, 2, \ldots, L\}$, implying that (2*nd level*) has I optimisation subproblems, (3*rd level*) K optimisation subproblems and (*mth level*) has L optimisation subproblems, respectively. For the sake of simplicity and without loss of generality, we analyse the relations in Problem (1) using two particular classes of multilevel programming problems: the bilevel programming problem, which organises vertically in two levels, and the bilevel programming problem with multi-followers, which is similar to bilevel programming but with several subproblems at the second level.

Bilevel Programming

Bilevel programming problems (BLPP) involve a hierarchy of two optimisation problems, of the following form [6,17,20,25,32]:

$$\begin{aligned} &\min_{x,y} \; F(x, y)\,, \\ &\text{s.t. } G(x, y) \leq 0\,, \\ &\quad x \in X\,, \\ &\quad y \in \operatorname{argmin}\{f(x, y)\colon g(x, y) \leq 0, y \in Y\}\,, \end{aligned} \tag{2}$$

where $X \subseteq \mathbb{R}^{nx}$ and $Y \subseteq \mathbb{R}^{ny}$ are both compact convex sets; F and f are real functions: $\mathbb{R}^{(nx+ny)} \to \mathbb{R}$; G and g are vectorial real functions, $G\colon \mathbb{R}^{(nx+ny)} \to \mathbb{R}^{nu}$ and $g\colon \mathbb{R}^{(nx+ny)} \to \mathbb{R}^{nl}$; $nx,\ ny \in \mathbb{N}$ and $nu, nl \in \mathbb{N} \cup \{0\}$. The following definitions are associated to Problem (2):

- Relaxed feasible set (or constrained region),

$$\Omega = \{x \in X, y \in Y\colon G(x, y) \leq 0, g(x, y) \leq 0\}; \tag{3}$$

- Lower level feasible set,

$$\mathrm{C}(x) = \{y \in Y\colon g(x, y) \leq 0\}; \tag{4}$$

- Follower's rational reaction set,

$$\mathrm{M}(x) = \{y \in Y\colon y \in \operatorname{argmin}\{f(x, y)\colon y \in \mathrm{C}(x)\}\}; \tag{5}$$

- Inducible region,

$$IR = \{x \in X, y \in Y\colon (x, y) \in \Omega, y \in \mathrm{M}(x)\}\,. \tag{6}$$

Note the parametric nature of the rational reaction set, (5), which reflects the dependence of the decisions taken at the upper levels on the decisions taken at the lower levels. This, in fact, is evidence that in bilevel programming problems the relations between the levels differ from the well-known Stackelberg game, where the decisions made by the followers don't affect the decision already taken by the leader [32].

Bilevel Programming with Multi-Followers

Bilevel programming problems with multi-followers involve two optimisation levels with several optimisa-

tion subproblems at the lower (*2nd level*):

$$\begin{aligned} \min_{x,y_1,y_2,\ldots,y_m} \quad & F(x,y_1,y_2,\ldots,y_m), \quad (1st\ level) \\ \text{s.t.} \quad & G(x,y_1,y_2,\ldots,y_m) \le 0, \\ & x \in X, \\ & y_i \in \operatorname{argmin}\{f_i(x,y_1,y_2,\ldots,y_m): \\ & g_i(x,y_1,y_2,\ldots,y_m) \le 0, y_i \in Y_i\}, \\ & \qquad\qquad (2nd\ level) \\ & i \in \{1,2,\ldots,m\}, \end{aligned} \tag{7}$$

with the following definitions:

- Feasible set for the *ith* follower,

$$\begin{aligned} &\Omega_i(x,y_1,y_2,\ldots,y_{i-1},y_{i+1},\ldots,y_m) \\ &= \{y_i \in Y_i: g_i(x,x,y_1,y_2,\ldots,y_m) \le 0\}, \end{aligned} \tag{8}$$

- Rational reaction set for the *ith* follower,

$$\begin{aligned} &\phi_i(x,y_1,y_2,\ldots,y_{i-1},y_{i+1},\ldots,y_m) = \{y_i \in Y_i: \\ &y_i \in \operatorname{argmin}\{f_i(x,y_1,y_2,\ldots,y_m): y_i \in \Omega_i(x)\}\}. \end{aligned} \tag{9}$$

Since one assumption is that followers may exchange information, conflicts naturally occur. The Nash equilibrium is often a preferred strategy to coordinate such decentralised systems [24]. Consequently, the optimisation subproblems positioned in the lower level reach a Nash equilibrium point, $(x, y_1^*, y_2^*, \ldots, y_m^*)$ [2]:

$$\begin{cases} f_1(x,y_1^*,y_2^*,\ldots,y_m^*) \le f_1(x,y_1,y_2^*,\ldots,y_m^*), \\ \forall y_1 \in Y_1, \\ f_2(x,y_1^*,y_2^*,\ldots,y_m^*) \le f_2(x,y_1^*,y_2,\ldots,y_m^*), \\ \forall y_2 \in Y_2, \\ \vdots \\ f_m(x,y_1^*,y_2^*,\ldots,y_m^*) \le f_m(x,y_1^*,y_2^*,\ldots,y_m), \\ \forall y_m \in Y_m. \end{cases} \tag{10}$$

Once more observe the parametric nature of the followers' rational reaction set, (9). In this case, however, each rational reaction set is a function of both the upper level decision variables and the decision variables of the other subproblems located in the same hierarchical level. Additionally, the priority remains to solve the leader's objective function to global optimality. Thus, we aim to compute the global optimum for the leader and the best possible equilibrium solution for the followers.

Applications

Applications of bilevel and multilevel programming include:

1. design optimisation problems in process systems engineering [4,5];
2. design of transportation networks [23];
3. agricultural planning [19];
4. management of multi-divisional firms [27] and
5. hierarchical decision-making structures [19].

Cases

Theoretical developments. Recently, Pistikopoulos and co-workers [9,10] have proposed novel solution algorithms which open the possibility of using a general framework to address general classes of bilevel and multilevel programming problems. These algorithms are based on parametric programming theory [1,11] and use of the Basic Sensitivity Theorem [15,16]. This approach can be classified as a *Reformulation Technique* [33] since the bilevel problem is transformed into a number of quadratic or linear problems. The main idea is to divide the follower's feasible area into different rational reaction sets, and search for the global optimum of a simple quadratic (or linear) programming problem in each area.

Global Optimum of a Bilevel Programming Problem

While for an optimal control problem (one-player problem) there is a well-defined concept for optimality, the same is not always true for multi-person games [2]. In the case of bilevel programming, [7,17,18,29,32,33] interpret the optimisation problem as a leader's problem, F, and search for the global minimum of F. The solution point obtained for the follower's problem, f, will respect the stationary (KKT) conditions and hence it can be any stationary point. Obviously, this solution strategy is acceptable when the player in the upper level of the hierarchy is in the most "powerful" position, and the other levels just react to the decision of their leader. Such an approach is sensible in many en-

gineering applications of bilevel programming (for instance, see [4,5]). It is also a valid strategy for the cases of decentralised manufacturing and financial structures when the leader has a full insight and control of the overall objectives and strategy of the corporation, while the follower does not.

However, this is not always the case. For example, using the *feedback Stackelberg solution*, where at every level of play a Stackelberg equilibrium point is searched, the commitment of the leader for his/her decision increases with the number of players involved. [3] present an example where the sacrifice of the leader's objective on behalf of the followers results in a better solution for both levels. Similar solution strategies have also been studied [3,22,28,30].

Theorem 1 [32] *If for each $x \in X$, f and g are twice continuously differentiable functions for every $y \in C(x)$, f is strictly convex for every $y \in C(x)$ and $C(x)$ is a convex and compact set, then $M(\cdot)$ is a real-valued function, continuous and closed.* □

If Theorem 1 applies and assuming that $M(x)$ is nonempty, then $M(x)$ will have only one element, which is $y(x)$. Thus, (2) can be reformulated as:

$$\begin{aligned} &\min_{x,y} \; F(x, y(x)) \\ &\text{s.t. } G(x, y(x)) \le 0 \\ &\quad x \in C_{rf} \\ &\quad C_{rf} = \{x \in X : \exists y \in Y, g(x, y) \le 0\} . \end{aligned} \tag{11}$$

Considering that f is a convex real function, the function $y(x)$ can be computed as a linear conditional function based on parametric programming theory, as follows [9]:

$$y(x) = \begin{cases} m^1 + n^1 x, & \text{if } H^1 x \le h^1 \\ m^2 + n^2 x, & \text{if } H^2 x \le h^2 \\ \vdots & \\ m^k + n^k x, & \text{if } H^k x \le h^k \\ \vdots & \\ m^K + n^K x, & \text{if } H^K x \le h^K \end{cases} \tag{12}$$

where, n^k, m^k and h^k are real vectors and H^k is a real matrix.

Theorem 2 [32] *If the assumptions of Theorem 1 hold, F is a real continuous function, X and the set defined by $G(x,y)$ are compact, and if $\{\exists x \in X : G(x, y(x)) \le 0\}$, then there is a global solution for Problem (2).* □

Since an explicit expression for y can be computed, if the assumptions of Theorem 2 hold, and the two players have convex functions to optimise, then the global optimum for Problem (2) can be obtained via the parametric programming approach. The advantage of using this approach is that the final solution will consider the possibility of existence of other global minima, which could correspond to better solutions for the follower. Moreover, the parametric nature of the leader's problem is preserved.

Regarding computational complexity, a number of authors have shown that bilevel programming problems are $\mathcal{NP}$-Hard [8,21]. Furthermore, [31] proved that even checking for a local optimum is an $\mathcal{NP}$-Hard problem.

The objective of this section is to describe a parametric programming framework which can solve different classes of multilevel programming problems to global optimality. We describe the fundamental developments for the quadratic bilevel programming case, and how the theory unfolds to address the existence of RHS uncertainty.

Bilevel Programming Problem

Consider the following general quadratic BLLP:

$$\begin{aligned} \min_{x,y} \; & F(x, y) = L_1 + L_2 x + L_3 y + \frac{1}{2} x^{\mathrm{T}} L_4 x + y^{\mathrm{T}} L_5 x \\ & \qquad + \frac{1}{2} y^{\mathrm{T}} L_6 y , \\ \text{s.t. } & G_1 x + G_2 y + G_3 \le 0 , \\ & \min_{y} f(x, y) = l_1 + l_2 x + l_3 y + \frac{1}{2} x^{\mathrm{T}} l_4 x + y^{\mathrm{T}} l_5 x \\ & \qquad + \frac{1}{2} y^{\mathrm{T}} l_6 y , \\ & \text{s.t. } g_1 x + g_2 y + g_3 \le 0 , \end{aligned} \tag{13}$$

where x and y are the optimisation variables, $x \in X \subseteq \mathbb{R}^{nx}$ and $y \in Y \subseteq \mathbb{R}^{ny}$. $[L_2]_{1xnx}$, $[L_3]_{1xny}$, $[L_4]_{nxxnx}$, $[L_5]_{nyxnx}$, $[L_6]_{nyxny}$, $[l_2]_{1xnx}$, $[l_3]_{1xny}$, $[l_4]_{nxxnx}$, $[l_5]_{nyxnx}$ and $[l_6]_{nyxny}$ are matrices defined in the

real space. The matrices $[G_1]_{nuxnx}$, $[G_2]_{nuxny}$, $[G_3]_{nux1}$, $[g_1]_{nuxnx}$, $[g_2]_{nuxny}$, $[g_3]_{nux1}$ correspond to the constraints, also defined in the real space.

By focusing attention on the follower's optimisation problem, considering x as a parameter vector and operating a variable change ($z = y + l_6^{-1} l_5 x$), it can be rewritten as the following mp-QP problem:

$$\begin{aligned} &\min_z f'(x,z) = l_1' + l_2' x + \frac{1}{2} x^{\mathrm{T}} l_4' x + \{ l_3' z + \frac{1}{2} z^{\mathrm{T}} l_6' z \} , \\ &\text{s.t. } g_2' z \le g_3' + g_1' x , \end{aligned} \tag{14}$$

where: $l_1' = l_1$; $l_2' = l_2 - l_3 l_6^{-1} l_5$; $l_3' = l_3$; $l_4' = l_4 - l_5^{\mathrm{T}} l_6^{-1} l_5$; $l_5' = 0$; $l_6' = l_6$; $g_1' = -(g_1 - g_2 l_6^{-1} l_5)$; $g_2' = g_2$; $g_3' = -g_3$. The mp-QP problem can be solved by applying the algorithm of [9]. As a result, a set of rational reaction sets (5) is obtained for different regions of x:

$$z^k = m^k + n^k x; \quad H^k x \le h^k, \quad k = 1, 2, \ldots, K . \tag{15}$$

Incorporating the expressions (15) into Problem (13) results in the following K quadratic problems:

$$\begin{aligned} &\min_x \; F'(x) = L_1'^k + L_2'^k x + \frac{1}{2} x^{\mathrm{T}} L_4'^k x, \\ &\text{s.t. } \; G_1'^k x \le G_3'^k , \end{aligned} \tag{16}$$

with:

$$\begin{aligned} L_1'^k =& L_1 + L_3 m^k + \frac{1}{2} m^{k^{\mathrm{T}}} L_6 m^k ; \\ L_2'^k =& L_2 + L_3 n^k - L_3 l_6^{-1} l_5 + m^{k^{\mathrm{T}}} \\ & \cdot L_5 + m^{k^{\mathrm{T}}} L_6 n^k - m^{k^{\mathrm{T}}} L_6 l_6^{-1} l_5 ; \\ L_4'^k =& L_4 + 2 n^k L_5 - 2 l_5^{\mathrm{T}} l_6^{-1} L_5 + n^{k^{\mathrm{T}}} \\ & \cdot L_6 n^k - 2 n^{k^{\mathrm{T}}} L_6 l_6^{-1} l_5 + l_5^{\mathrm{T}} l_6^{-1} L_6 l_6^{-1} l_5 ; \\ G_1' =& G_1 + G_2 n^k - G_2 l_6^{-1} l_5 ; \\ G_3' =& -(G_3 + G_2 m^k); \\ G_1'^k =& [G_1' | H^k]^{\mathrm{T}}_{(nx)\mathrm{x}(nu+n_{h^k})} ; \\ G_3'^k =& [G_3' | h^k]^{\mathrm{T}}_{(1)\mathrm{x}(nu+n_{h^k})} . \end{aligned}$$

Clearly, the solution of the BLLP Problem (13) is the minimum along the K solutions of Problem (16).

Remark 1 The artificial variable, z, introduced in Problem (14) is only necessary if $l_5 \neq \underline{0}$. In all other cases the multi-parametric problem can be easily formulated through algebraic manipulations.

Remark 2 When one of the matrices l_6', $L_4'^k$ is null the optimisation problem where these are involved becomes linear. In particular, if $l_6' = \underline{0}$, Problem (14) is transformed into an mp-LP; on the other hand, if $L_4'^k = \underline{0}$, Problem (16) becomes an LP problem. In both cases, the solution procedure is not affected, due to the fact that the Basic Sensitivity Theorem [15,16] also applies to the mp-LP problem.

Remark 3 The expression for the artificial variable introduced, z, is only valid when l_6 is symmetric. If not, with the following transformation:

$$\bar{l}_6 = \left\{ \frac{l_6 + l_6^{\mathrm{T}}}{2} \right\} ,$$

the resulting matrix is non-singular. If the resulting matrix is singular, the expression for the artificial variable should be given by:

$$\begin{aligned} &z = y + Ax , \\ &\text{where } A \text{ should satisfy:} \\ &\left\{ A \in \mathbb{R}^{nxxnx} : \; l_5 - (\frac{1}{2} l_6 + \frac{1}{2} l_6^{\mathrm{T}}) A = 0 \right\} . \end{aligned}$$

In this case, several solutions for the system above can exist. However, as long as the bilinear terms are eliminated in Problem (14) any solution can be selected.

Remark 4 This technique is not valid when at the same time:

1. f is a pure quadratic cost function,
2. f involves bilinear terms and
3. matrix $\bar{l}_6$ is singular.

Observing Formulation (16) we can conclude that the parametric programming approach, Alg. 1, transforms the original quadratic bilevel programming problem into simple quadratic problems, for which a global optimum can be reached.

Bilevel Programming Problem with Multi-Followers

Consider the bilevel programming problem with multi-followers, and assume quadratic objective functions,

linear constraints and two followers:

$$
\begin{aligned}
\min_{x,y_1,y_2} f_1 = & L_1^1 + \\
& + L_2^1 \cdot x + L_3^1 \cdot y_1 + L_4^1 \cdot y_2 + \quad (1st\ level) \\
& + \frac{1}{2} x^{\mathrm{T}} \cdot L_5^1 \cdot x + \frac{1}{2} y_1^{\mathrm{T}} \cdot L_6^1 \cdot y_1 \\
& + \frac{1}{2} y_2^{\mathrm{T}} \cdot L_7^1 \cdot y_2 + x^{\mathrm{T}} \cdot L_8^1 \cdot y_1 \\
& + y_2^{\mathrm{T}} \cdot L_9^1 \cdot x + y_2^{\mathrm{T}} \cdot L_{10}^1 \cdot y_1 , \\
& \left| \begin{array}{l} G_1^1 \cdot x + G_2^1 \cdot y_1 + G_3^1 \cdot y_2 \le 0 , \\ \qquad\qquad\qquad\qquad (2nd\ level) \\ \min\limits_{y_1} f_2 = L_1^2 + \qquad\qquad \underline{Follower\ 1} \\ + L_2^2 \cdot x + L_3^2 \cdot y_1 + L_4^2 \cdot y_2 \\ + \frac{1}{2} x^{\mathrm{T}} \cdot L_5^2 \cdot x + \frac{1}{2} y_1^{\mathrm{T}} \cdot L_6^2 \cdot y_1 \\ + \frac{1}{2} y_2^{\mathrm{T}} \cdot L_7^2 \cdot y_2 + x^{\mathrm{T}} \cdot L_8^2 \cdot y_1 \\ + y_2^{\mathrm{T}} \cdot L_9^2 \cdot x + y_2^{\mathrm{T}} \cdot L_{10}^2 \cdot y_1 , \\ \text{s.t. } G_1^2 \cdot x + G_2^2 \cdot y_1 + G_3^2 \cdot y_2 \le 0 , \\ \\ \min\limits_{y_2} f_3 = L_1^3 + \qquad\qquad \underline{Follower\ 2} \\ + L_2^3 \cdot x + L_3^3 \cdot y_1 + L_4^3 \cdot y_2 \\ + \frac{1}{2} x^{\mathrm{T}} \cdot L_5^3 \cdot x + \frac{1}{2} y_1^{\mathrm{T}} \cdot L_6^3 \cdot y_1 \\ + \frac{1}{2} y_2^{\mathrm{T}} \cdot L_7^3 \cdot y_2 + x^{\mathrm{T}} \cdot L_8^3 \cdot y_1 \\ + y_2^{\mathrm{T}} \cdot L_9^3 \cdot x + y_2^{\mathrm{T}} \cdot L_{10}^3 \cdot y_1 , \\ \text{s.t. } G_1^3 \cdot x + G_2^3 \cdot y_1 + G_3^3 \cdot y_2 \le 0 . \end{array} \right. \\
& \qquad\qquad\qquad\qquad\qquad\qquad (17)
\end{aligned}
$$

The difference between Problem (17) and Problem (13) is the existence of two optimisation subproblems in a single level. Accordingly, the concept of Nash equilibrium is introduced.

As in the bilevel programming case, each optimisation subproblem in (*2nd level*) is recast as a multi-parametric programming problem. In this problem, the parameters are all the variables from the optimisation problem at (*1st level*) as well as the optimisation variables of the other subproblems at the same level, *Follower* 1 or *Follower* 2 in this case (17). Thus, defining vectors, $\left[\omega^2\right]^{\mathrm{T}} = \left[x | y_2\right]$ and $\left[\omega^3\right]^{\mathrm{T}} = \left[x | y_1\right]$, we

Algorithm – Parametric Programming Algorithm for BLPP

1. Recast the inner problem as a multi-parametric programming problem, with the leader's variables being the parameters (14);
2. Solve the resulting problem using the suitable multi-parametric programming algorithm;
3. Substitute each of the K solutions in the leader's problem, and formulate the K one-level optimisation problems;
4. Compare the K optimum points and select the best one.

Bilevel Programming Framework for Enterprise-Wide Process Networks Under Uncertainty, Algorithm 1
Parametric Programming algorithm for a BLPP

rewrite the (*2nd level*) optimisation subproblems as,

$$
\begin{aligned}
\min_{y_1} f_2(y_1, \omega^2) = & L_1^2 + L_2^{2*} \cdot \omega^2 + L_3^2 \cdot y_1 \\
& + \frac{1}{2} \omega^{2^{\mathrm{T}}} \cdot L_5^{2*} \cdot \omega^2 + \frac{1}{2} y_1^{\mathrm{T}} \cdot L_6^2 \cdot y_1 \\
& + y_1^{\mathrm{T}} \cdot L_8^{2*} \cdot \omega^2 , \\
\text{s.t.} \quad & G_1^{2*} \cdot \omega^2 + G_2^2 \cdot y_1 \le 0 ,
\end{aligned}
\qquad (18)
$$

and,

$$
\begin{aligned}
\min_{y_2} f_3(y_2, \omega^2) = & L_1^3 + L_2^{3*} \cdot \omega^3 + L_4^3 \cdot y_2 \\
& + \frac{1}{2} \omega^{3^{\mathrm{T}}} \cdot L_5^{3*} \cdot \omega^{2a} + \frac{1}{2} y_2^{\mathrm{T}} \cdot L_7^3 \cdot y_2 \\
& + y_1^{\mathrm{T}} \cdot L_9^{3*} \cdot \omega^3 , \\
\text{s.t.} \quad & G_1^{3*} \cdot \omega^3 + G_3^3 \cdot y_2 \le 0 ,
\end{aligned}
\qquad (19)
$$

where ω^2 and ω^3 are the vectors of parameters. The bilinearities can be circumvented by using a similar strategy to the one used in the bilevel case. By using a multi-parametric programming algorithm [9], problems (18) and (19) result in the following parametric expressions:

$$
\begin{cases}
y_1 = \phi_1(x, y_2) & \rightarrow \text{ rational reaction set follower 1} , \\
y_2 = \phi_2(x, y_1) & \rightarrow \text{ rational reaction set follower 2} ,
\end{cases}
\qquad (20)
$$

Algorithm

1. Recast each of the subproblems in the lower level as a multi-parametric programming problem, with the variables out of their control being the parameters (18–19);
2. Solve the resulting problems using the suitable multi-parametric programming algorithm;
3. Compute a Nash equilibrium point by direct comparison of the rational reaction sets (21);
4. Substitute each of the K solutions in the leader's problem, and formulate the K one level optimisation problems;
5. Compare the K optima points and select the best one.

Bilevel Programming Framework for Enterprise-Wide Process Networks Under Uncertainty, Algorithm 2
Parametric programming algorithm for bilevel programming problems with multi-followers

which are then used to compute the Nash equilibrium (x, y_1^*, y_2^*):

$$\begin{cases} f_1(x, y_1^*, y_2^*) \le f_1(x, y_1, y_2^*), & \forall y_1 \in Y_1\,, \\ f_2(x, y_1^*, y_2^*) \le f_2(x, y_1^*, y_2), & \forall y_2 \in Y_2, \end{cases} \tag{21}$$

easily computed by direct comparison [24]:

$$\phi_1'(x, y_1) = \phi_2(x, y_1), \quad \rightarrow y_1 = \phi_2^*(x)\,, \tag{22a}$$

$$\phi_1(x, y_2) = \phi_2'(x, y_2), \quad \rightarrow y_2 = \phi_1^*(x)\,. \tag{22a}$$

Finally, substituting the expressions in (22) in the leader's optimisation problem, (1*st level*), we end up with a single-level convex optimisation problem, involving only the leader's optimisation variables, as follows:

$$\begin{aligned} &\min_x \; f_1^*(x, y_1(x, y_2^*(x)), y_2(x, y_1^*(x)))\,, \\ &\text{s.t. } G_1(x, y_1(x, y_2^*), y_2(x, y_1^*)) \le 0, \quad x \in C_{rf}\,, \\ &\quad C_{rf} = \{x \in X\colon \exists_{y_1, y_2} \in Y, Z, G_2(x, y_1, y_2) \le 0\,, \\ &\qquad G_3(x, y_1, y_2) \le 0\}\,. \end{aligned} \tag{23}$$

The algorithm is summarised in Alg. 2.

Bilevel Programming with Uncertainty

[12] highlighted the importance of considering uncertainty/risk (e. g. prices, technological attributes, etc.) in the solution of decentralised decision makers. A comprehensive analysis of linear bilevel programming problems can be found in [27], where uncertainty is considered unstructured, taking any value between its bounds. Here it is extended to the quadratic case. We address the following quadratic BLPP with uncertainty, θ:

$$\begin{aligned} \min_{x,y} \; & F(x, y, \theta) = L_1 + L_2 x + L_3 y + \frac{1}{2} x^{\mathrm{T}} L_4 x \\ & \qquad + y^{\mathrm{T}} L_5 x + \frac{1}{2} y^{\mathrm{T}} L_6 y \\ \text{s.t. } & G_1 x + G_2 y + G_3 \le G_4 \theta \\ & \min_y \; f(x, y, \theta) = l_1 + l_2 x + l_3 y \\ & \qquad \frac{1}{2} x^{\mathrm{T}} l_4 x + y^{\mathrm{T}} l_5 x + \frac{1}{2} y^{\mathrm{T}} l_6 y \\ & \text{s.t. } g_1 x + g_2 y + g_3 \le g_4 \theta\,, \end{aligned} \tag{24}$$

The steps for solving (24) are as follows:

1. Recast the inner problem as an mp-QP, with parameters being both x and θ. The solution obtained is similar to (15):

$$\begin{aligned} & z^k = m^k + n_b^k x + \tilde{n}_c^k \theta\,; \quad H^k x + \tilde{H}^k \theta \le h^k\,, \\ & k = 1, 2, \ldots, K\,. \end{aligned} \tag{25}$$

2. Incorporate expressions (25) in (24) to formulate K mp-QPs, with parameters being the uncertainty θ:

$$\begin{aligned} \min_x \; & F'(x, \theta) = \tilde{L}_1'^k + \tilde{L}_2'^k x + \frac{1}{2} x^{\mathrm{T}} \tilde{L}_4'^k x \\ \text{s.t. } & \tilde{G}_1'^k x \le \tilde{G}_3'^k + \tilde{G}_4'^k \theta\,, \end{aligned} \tag{26}$$

where $\tilde{L}_1'^k, \tilde{L}_2'^k, \tilde{L}_4'^k, \tilde{G}_1'^k, \tilde{G}_2'^k, \tilde{G}_4'^k$ are appropriate matrices derived by algebraic manipulations.

References

1. Acevedo J, Pistikopoulos EN (1997) A multiparametric programming approach for linear process engineering problems under uncertainty. Ind Eng Chem Res 36:717–728
2. Başar T, Olsder GJ (1982) Dynamic Noncooperative Game Theory. Academic Press, London
3. Cao D, Chen M (2006) Capacitated plant selection in a decentralized manufacturing environment: a bilevel optimization approach. Eur J Oper Res 169(1):97–110

4. Clark PA (1990) Bilevel programming for steady-state chemical process design – ii. performance study for non-degenerate problems. Comput Chem Eng 14(1):99–109
5. Clark PA, Westerberg AW (1990) Bilevel programming for steady-state chemical process design – i. fundamentals and algorithms. Comput Chem Eng 14(1):87–97
6. Dempe S (2003) Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. Optimization 52(3):33–359
7. Dempe S, Kalashnikov V, Ríos-Mercado RZ (2005) Discrete bilevel programming: Application to a natural gas cash-out problem. Eur J Oper Res 166:469–488
8. Deng X (1998) Complexity issues in bilevel linear programming. In: Multilevel optimization: algorithms and applications. Kluwer, Dordrecht, pp 149–164
9. Dua V, Bozinis A, Pistikopoulos EN (2002) A multiparametric programming approach for mixed-integer quadratic engineering problems. Comput Chem Eng 26:715–733
10. Dua V, Pistikopoulos EN (2000) An algorithm for the solution of multiparametric mixed integer linear programming problems. Ann Oper Res 99:123–139
11. Dua V (2000) Parametric programming techniques for process engineering problems under uncertainty. PhD thesis, Department of Chemical Engineering and Chemical Technology Imperial College of Science, Technology and Medicine London, London
12. Evans GW (1984) An overview of thecniques for solving multiobjective mathematical programs. Manag Sci 30(11):1268–1282
13. Faísca NP, Dua V, Saraiva PM, Rustem B, Pistikopoulos EN (2007) Parametric global optimisation for bilevel programming. J Glob Optim 38(4):609–623
14. Faísca NP, Saraiva PM, Rustem B, Pistikopoulos EN (2007) A multi-parametric programming approach for multilevel hierarchical and decentralised optimisation problems. Comput Manag Sci (in press)
15. Fiacco AV (1976) Sensitivity analysis for nonlinear programming using penalty methods. Math Program 10:287–311
16. Fiacco AV (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Academic Press, New York
17. Floudas CA (2000) Deterministic global optimization. Kluwer, Dordrecht
18. Floudas CA, Pardalos PM, Adjiman CS, Esposito WR, Gümüş ZH, Harding ST, Klepeis JL, Meyer CA, Schweiger CA (1999) Handbook of test problems in local and global optimization. Kluwer, Dordrecht
19. Fortuny-Amat J, McCarl B (1981) A representation and economic interpretation of a two-level programming problem. J Oper Res Soc 32(9):783–792
20. Gümüş ZH, Floudas CA (2001) Global optimization of nonlinear bilevel programming problems. J Glob Optim 20(1):1–31
21. Hansen P, Jaumard B, Savard G (1992) New brach-and-bound rules for linear bilevel programming. SIAM J Sci Stat Comput 13:1194–1217
22. Lai Y (1996) Hierarchical optimization: a satisfactory solution. Fuzzy Sets Syst 77:321–335
23. LeBlanc LJ, Boyce DE (1985) A bilevel programming algorithm for exact solution of network design problem with user-optimal flows. Transp Res B Methodol 20:259–265
24. Liu B (1998) Stackelberg-nash equilibrium for multilevel programming with multiple followers using genetic algorithms. Comput Math Appl 36(7):79–89
25. Migdalas A, Pardalos PM, Varbrand P (1997) Multilevel optimization: algorithm and applications. Kluwer, Dordrecht
26. Ryu J, Dua V, Pistikopoulos EN (2004) A bilevel programming framework for enterprise-wide process networks under uncertainty. Comput Chem Eng 28:1121–1129
27. Ryu J-H (2003) Design and operation of enterprise-wide process networks under uncertainty. PhD thesis, Department of Chemical Engineering and Chemical Technology Imperial College of Science, Technology and Medicine London, London
28. Shih H, Lai Y, Lee ES (1996) Fuzzy approach for multi-level programming problems. Comput Oper Res 23(1):73–91
29. Shimizu K, Ishizuka Y, Bard JF (1997) Nondifferentiable and two-level mathematical programming. Kluwer, Boston
30. Tabucanon MT (1988) Multiple Criteria Decision Making in Industry. Elsevier, Amsterdam
31. Vicente LN, Savard G, Júdice J (1994) Descent approaches for quadratic bilevel programming. J Optim Theor Appl 81:379–399
32. Vicente L (1992) Bilevel programming. Master's thesis, Department of Mathematics, University of Coimbra, Coimbra
33. Visweswaran V, Floudas MG, Ierapetritou CA, Pistikopoulos EN (1996) A decomposition-based global optimization approach for solving bilevel linear and quadratic programs. In: State of the art in global optimization. Kluwer, Dordrecht, pp 139–162

Bilevel Programming: Global Optimization

VISWANATHAN VISWESWARAN
SCA Technologies LLC, Pittsburgh, USA

MSC2000: 90C90, 90C30

Article Outline

Keywords
Definitions
Complexity
Multiple Solutions to the Follower's Problem

Solution Methods
Enumeration Methods
Complementary Pivot Methods
Branch and Bound Methods
Computational Results and Test Problems
See also
References

Keywords

Bilevel programming; Global optimization; Stackelberg game

A large number of mathematical programming problems have optimization problems in their constraints. Arising from the areas of game theory and multicriteria decision making, these *bilevel programming problems* (BPP) take the form:

$$\begin{cases} \min\limits_{x} & F(x,y) \\ \text{s.t.} & G(x,y) \le 0 \\ & y = \begin{cases} \operatorname{Arg}\min\limits_{y} & f(x,y) \\ \text{s.t.} & g(x,y) \le 0 \end{cases} \end{cases} \tag{1}$$

where $x \in \mathbf{R}^{n_1}$, $y \in \mathbf{R}^{n_2}$ and the functions $F(x, y)$, $f(x, y)$, $G(x, y)$ and $g(x, y)$ are continuous and twice differentiable. It is generally assumed that these functions are convex; the case of nonconvex functions has not been considered in the literature so far (as of 2000).

Bilevel programming has its origins in *Stackelberg game theory*, in particular from models of two-person nonzero-sum games. In these games, two players make alternate moves in a pre-established order. The first player (the leader) selects a move, x, that optimizes his own cost function. The second player (the follower) then has to make a move y that is constrained by the prior decision of the leader. The follower has access only to his own cost function, while the leader is aware of both his own as well as the follower's cost function, and can thus foresee the reaction of the follower to any move that the leader makes. If the cost functions of the two players are identical (called the *cooperative case*), then the two constraint sets can be merged and the problem can be solved as a single level optimization problem. If the cost functions are exactly opposite (that is, $f(x, y) = -F(x, y)$), then there can be neither cooperation or compromise. The most interesting (and normally studied) case is when the two objectives are neither identical nor opposite.

BPP also arises in *hierarchical decision making*. For example, a central planning office might decide upon national budgets which act as constraints for local governments and businesses. Other applications include long-range planning problems followed by short term scheduling in the chemical process industries and energy planning of businesses constrained by national government policy. A detailed list of references for applications of BPP can be found in [14]. See [13] for a full review of algorithms and applications of bilevel and multilevel programming.

Definitions

The following definitions will be used in the sequel. The *relaxed constraint region* for the BPP is defined as

$$S = \{(x,y)\colon\ G(x,y) \le 0,\ g(x,y) \le 0\}\,.$$

The follower's *feasible region* for a fixed x, $\sigma(x)$, is defined as

$$\sigma(x) = \{y\colon\ g(x,y) \le 0\}\,.$$

This set is parametric in x, and represents the allowable choices for the follower. The *rational reaction set* $M(x)$ is defined as

$$M(x) = \{y\colon\ y \in \operatorname{Arg}\min\{f(x,y)\colon\ y \in \sigma(x)\}\}\,.$$

Finally, the *inducible region* for the problem is

$$\mathrm{IR} = \{(x,y)\colon\ y \in M(x),\ (x,y) \in S\}\,.$$

The inducible region IR (which represents the follower's feasible region) is in general nonconvex. In terms of the bimatrix or Stackelberg games, IR represents 'equilibrium' points, that is, the set of compromise solutions between the leader and the follower. In the presence of first level constraints (1), IR may be empty, which implies that the BPP has no solution. However, it can be shown that the IR is compact and the BPP has a solution, if the following conditions are met [7]:

a) $F(x, y)$, $f(x, y)$, $G(x, y)$ and $g(x, y)$ are continuous and twice differentiable;
b) $f(\cdot, y)$ is strictly convex in y;
c) $\sigma(x)$ is a compact convex set; and
d) $F(x, y)$ and $G(x, y)$ are convex in x and y.

Note that the solution to the BPP need not be individually optimal for each of the leader's and follower's objective function (that is, it need not be an *efficient solution*).

The specific instance of BPP when all the functions involved are linear has received the most interest. The *linear bilevel programming problem* (BLPP) can be written as

$$\begin{cases} \min\limits_{x} & F_L(x,y) = c_1^\top x + d_1^\top y \\ \text{s.t.} & A_1 x + B_1 y \le b_1 \\ & y = \begin{cases} \operatorname{Arg\,min}\limits_{y} & c_2^\top x + d_2^\top y \\ \text{s.t.} & g(x,y) \\ & \quad = A_2 x + B_2 y \le b_2. \end{cases} \end{cases} \tag{2}$$

Complexity

Because of the nonconvexity of the induced region IR, BPP can be a hard problem to solve. It is generally known that even the linear problem, BLPP, is *NP*-hard. This has been shown by reducing the problem to a knapsack optimization problem [3], the standard KERNEL problem [11], and by reduction to a problem of minimizing a convex quadratic function over a polyhedron [1]. In fact, even checking for local optimality in BLPP is *NP*-hard [15].

Multiple Solutions to the Follower's Problem

In the absence of dual degeneracy, the follower's subproblem has a single solution for every x. However, if the follower's subproblem has multiple solutions for any x, then the overall BPP may not be well-defined. In this case, we need further assumptions about the cooperativeness of the follower with respect to the leader. Alternately, the follower's objective function can be modified as

$$f(x,y) = f(x,y) + \epsilon F(x,y);$$

in effect, allowing the leader to 'kick back' a small portion of its earnings to ensure that the follower selects a suitable solution.

Solution Methods

From the 1980s onwards, many approaches have been proposed for the solution of BPP. These can be classified as enumerative, complementary pivot, branch and bound, descent and penalty function methods. The last two categories of methods are only useful in finding stationary points and local minima, and will not be discussed here. The vast majority of the approaches address the linear case, BLPP. Some of the global optimization methods are discussed below.

Enumeration Methods

The linear BLPP is equivalent to maximizing the linear function $F_L(x,y)$ over a piecewise linear constraint region composed of the edges and hyperplanes of S, the feasible region. It can be shown that the global optimum to BLPP occurs at a vertex of S. This suggests an extreme point search procedure for solving BLPP. One such procedure is the *Bialas–Karwan Kth-best algorithm* [4]. The basic idea is to find an 'ordered' set of extreme points to the relaxed problem

$$\begin{cases} \min\limits_{x} & F_L(x,y) = c_1^\top x + d_1^\top y \\ \text{s.t.} & A_1 x + B_1 y \le b_1 \\ & A_2 x + B_2 y \le b_2. \end{cases}$$

The algorithm has the following steps:

0	Solve the relaxed problem. Let the solution be (x^1, y^1). Set $k = 1$.
1	Solve the inner problem with $x = x^k$. If y^k is in the solution set to the inner problem, then STOP.
2	Locate all adjacent extreme points (x_i, y_i) such that $$c_1^\top x_i + d_1^\top y_i \le c_1^\top x^k + d_1^\top y^k \quad \forall i.$$ Choose the adjacent extreme point j that minimizes $c_1^\top x_j + d_1^\top y_j$. Set $k = k+1$, $(x^{k+1}, y^{k+1}) = (x_j, y_j)$. Go to Step 1.

Since each successive pair of points tested in this algorithm is adjacent, it can be efficiently implemented using the dual simplex method.

Complementary Pivot Methods

Under proper regularity conditions, the inner problem to the BPP can be replaced by its Karush–Kuhn–Tucker

optimality conditions. For the case of the BLPP, this results in the following single-level optimization problem (KKT):

$$\begin{cases} \min\limits_{x} & F_L(x,y) = c_1^\top x + d_1^\top y \\ \text{s.t.} & A_1 x + B_1 y \le b_1 \\ & A_2 x + B_2 y \le b_2 \\ & \mu(A_2 x + B_2 y - b_2) = 0 \\ & d_2 + A_2^\top \mu = 0 \\ & \mu \ge 0. \end{cases} \tag{3}$$

The problem KKT has a linear complementarity pivot formulation. As such, it can be solved using a complementary pivoting method. Consider the following parametric formulation LCP(λ):

$$\begin{aligned} & c_1^\top x + d_1^\top y \le \lambda, \\ & A_1 x + B_1 y \le b_1, \\ & A_2 x + B_2 y \le b_2, \\ & \mu(A_2 x + B_2 y - b_2) = 0, \\ & d_2 + A_2^\top \mu = 0, \\ & \mu \ge 0. \end{aligned}$$

The global minimization of BLPP then corresponds to the identification of the minimum value of λ such that LCP(λ) has a solution. The following method can be used to solve LCP(λ):

0	Solve LCP(λ) without the first parametric constraint. Let (x^0, y^0) be the solution to this problem, with $\lambda_0 = c_1^\top x^0 + d_1^\top y^0$.
1	Solve LCP(λ^k). If LCP(λ^k) has no solution, go to Step 3. Otherwise, let (x^k, y^k) be the solution.
2	Set $\lambda^{k+1} = c_1^\top x^k + d_1^\top y^k - \gamma \left\| c_1^\top x^k + d_1^\top y^k \right\|$, where γ is a small positive number. Set $k = k+1$, go to Step 1.
3	If $k = 0$, then BLPP has no solution. Otherwise, x^k, y^k is an ϵ-global optimum to BLPP,where $\epsilon = \gamma \left\| c_1^\top x^k + d_1^\top y^k \right\|$.

The key to this algorithm is the ability to efficiently solve LCP(λ^k) in Step 1. J. Judice and A. Faustino [12] have proposed a hybrid enumerative method which works by branching on the complementarity conditions $\mu(A_2 x + B_2\, y - b_2) = 0$. Numerous heuristics can be used in each node of the resulting branch and bound tree, in order to reduce the search for a complementary solution.

Branch and Bound Methods

These methods work by identifying the set of inner-level constraints that are active at the optimal solution. The simplest method, due to J. Fortuny-Amat and B. McCarl [10], works by converting the KKT complementarity conditions in (3) to

$$\begin{aligned} & \mu(A_2 x + B_2 y - b_2) = 0, \mu_i \le M\alpha_i, \\ & A_2 x + B_2 y - b_2 \ge M(1 - \alpha_i), \\ & \alpha_i = 0 - 1, \quad \forall i, \end{aligned}$$

where M is a large constant. The variable α_i is equal to 1 if inner level constraint i is active at the optimal solution, and zero otherwise. This converts the one-level problem to a mixed integer linear program (MILP), which can be solved with commercial MILP codes. However, this requires the addition of $2 \cdot m$ constraints and m variables, where m is the number of inner-level constraints.

Note that at the optimal solution, at least one of the inner problem constraints must be active, that is,

$$\sum_{i=1}^{m} \alpha_i \ge 1. \tag{4}$$

Moreover, it can be shown that the following conditions must hold [11]:

$$\sum_{\{i:\ B_{2ij} > 0\}} \alpha_i \ge \quad \text{if } d_j < 0, \tag{5}$$

$$\sum_{\{i:\ B_{2ij} < 0\}} \alpha_i \ge 1 \quad \text{if } d_j > 0, \tag{6}$$

for $j = 1, \ldots, n_2$. It is possible to use (4)–(6) as branching criteria in a branch and bound tree. Each of these conditions, when tight, can be used to eliminate a variable from the inner constraints. By combining these conditions with the use of linear relaxations to obtain lower

bounds, a branch and bound algorithm can be developed to solve the BLPP [11].

An alternate method to the use of binary variables is to establish a one-to-one correspondence between each α_i and each μ_i, as follows:

$$\frac{1}{M}\alpha_i \le \mu_i \le M\alpha_i,$$

where M is a suitably large number. This ensures that if $\alpha_i = 0$, then $\mu_i = 0$, while if $\alpha_i = 1$, $\mu_i \ge (1/M)$ implying an inactive constraint. With this approach, BLPP can be transformed to:

$$\begin{cases} \min\limits_{x} & F_L(x,y) = c_1^\top x + d_1^\top y \\ \text{s.t.} & A_1 x + B_1 y \le b_1 \\ & A_2 x + B_2 y \le b_2 \\ & \alpha(A_2 x + B_2 y - b_2) = 0 \\ & d_2 + A_2^\top \mu = 0 \\ & \mu_i \le M\alpha_i \\ & \alpha_i \le M\mu_i \\ & \mu \ge 0, \quad \alpha = \{0,1\}. \end{cases}$$

By partitioning the variables into $\overline{x} = (x, y)$ and $\overline{y} = (\mu, \alpha)$, it can be seen that this problem is of the form

$$\begin{cases} \min\limits_{\overline{x},\overline{y}} & \overline{f}(\overline{x},\overline{y}) \\ \text{s.t.} & \overline{g}(\overline{x},\overline{y}) \le 0 \\ & \overline{h}(\overline{x},\overline{y}) = 0, \end{cases}$$

where $\overline{f}(\overline{x},\overline{y})$, $\overline{g}(\overline{x},\overline{y})$ and $\overline{h}(\overline{x},\overline{y})$ are bilinear functions. Thus, the *GOP algorithm* of [8,9] can be applied to solve this problem. The algorithm works by solving a set of primal and relaxed dual problems that bound the global solution. The primal problem is

$$\begin{cases} \min\limits_{\overline{x}} & \overline{f}(\overline{x},\overline{y^k}) \\ \text{s.t.} & \overline{g}(\overline{x},\overline{y^k}) \le 0 \\ & \overline{h}(\overline{x},\overline{y^k}) = 0, \end{cases}$$

where y^k is a fixed number. Because this problem is linear, it can be solved for its global solution, and yields an upper bound on the global solution. It also provides multipliers for the constraints, μ^k and λ^k, which can be used to construct a Lagrange function of the form

$$L(\overline{x},\overline{y},\mu^k,\lambda^k) = \overline{f}(\overline{x},\overline{y^k}) + \mu^k\overline{g}(\overline{x},\overline{y^k}) + \lambda^k\overline{h}(\overline{x},\overline{y^k}).$$

It is then possible to solve a dual problem

$$\begin{cases} \min\limits_{\overline{y}} & u \\ \text{s.t.} & u \ge L(\overline{x},\overline{y},\mu^k,\lambda^k), \end{cases}$$

which provides a lower bound on the global solution. The dual problem is actually solved by partitioning the $\overline{y}$-space using the gradients of L and solving a relaxed dual subproblem in each region. In [16] it has been shown that for the bilevel problems, only one dual subproblem needs to be solved at each iteration. This approach can also be used when the inner problem objective function is quadratic.

Another approach, proposed in [2], can also be used when the inner level problem has a convex quadratic objective function. The basic idea is to first solve the one-level linear problem by dropping the complementarity conditions. At each iteration, a check is made to see if the complementarity condition is satisfied. If it is, the corresponding solution is in the inducible region IR, and hence a candidate solution for BPP. If not, a branch and bound scheme is used to implicitly examine all combinations of complementary slackness.

Let $W_1 = \{i\colon \mu_i = 0\}$, $W_2 = \{i\colon g_i = 0\}$, $W_3 = \{i\colon i \notin W_1 \cup W_2\}$.

0	Set $k = 0$, $W_1 = W_2 = \emptyset$, $W_3 = i$, $\overline{F} = \infty$.
1	Set $\mu_i = 0$, $i \in W_1$, $g_i = 0$, $i \in W_2$. Solve the relaxed system. Let (x^k, y^k, μ^k) be the solution. If no solution exists, or if $F(x^k, y^k) \ge \overline{F}$, go to Step 4.
2	If $\mu_i g_i = 0$, $\forall i$, go to Step 3. Otherwise select i such that $\mu_i g_i$ is maximal, say $\hat{i}$. Let $W_1 = W_1 \cup \hat{i}$, $W_3 = W_3 \cup \hat{i}$, and go to Step 1.
3	Update $\overline{F} = F(x^k, y^k)$.
4	If all nodes in the three have been exhausted, go to Step 5. Else, branch to the newest unfathomed node, say j, and set $W_1 = W_1 \cup j$, $W_2 = W_2 \cup j$. Go to Step 1.
5	If $\overline{F} = \infty$, no solution exists to BPP. Otherwise, the point corresponding to $\overline{F}$ is the optimum.

Computational Results and Test Problems

The difficulty of solving bilevel problems depends on a number of factors, including the number of inner

versus outer level variables, degree of cooperation between the leader and follower objective functions, number of inner level constraints and the density of the constraints. Computational results have been reported by many authors, including [2,11,12] and [16]. Generally, these have so far been limited to problems involving up to 100 inner level variables and constraints. See [5,6] for methods for automatically generating linear and quadratic bilevel problems which can be used to test any of these and other algorithms for bilevel programming.

See also

- ▶ Bilevel Fractional Programming
- ▶ Bilevel Linear Programming
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Bilevel Programming
- ▶ Bilevel Programming: Applications
- ▶ Bilevel Programming: Applications in Engineering
- ▶ Bilevel Programming: Implicit Function Approach
- ▶ Bilevel Programming: Introduction, History and Overview
- ▶ Bilevel Programming: Optimality Conditions and Duality
- ▶ Multilevel Methods for Optimal Design
- ▶ Multilevel Optimization in Mechanics
- ▶ Stochastic Bilevel Programs

References

1. Bard JF (1991) Some properties of the bilevel programming problem. J Optim Th Appl 68:371–378
2. Bard JF, Moore J (1990) A branch and bound algorithm for the bilevel programming problem. SIAM J Sci Statist Comput 11:281–292
3. Ben-Ayed O, Blair C (1990) Computational difficulties of bilevel linear programming. Oper Res 38:556–560
4. Bialas W, Karwan M (1984) Two-level linear programming. Managem Sci 30:1004–1020
5. Calamai P, Vicente L (1993) Generating linear and linear-quadratic bilevel programming problems. SIAM J Sci Statist Comput 14:770–782
6. Calamai P, Vicente L (1994) Generating quadratic bilevel programming problems. ACM Trans Math Softw 20: 103–119
7. Edmunds T, Bard J (1991) Algorithms for nonlinear bilevel mathematical programming. IEEE Trans Syst, Man Cybern 21:83–89
8. Floudas CA, Visweswaran V (1990) A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: I. theory. Comput Chem Eng 14:1397
9. Floudas CA, Visweswaran V (1993) A primal-relaxed dual global optimization approach. J Optim Th Appl 78(2):187
10. Fortuny-Amat J, McCarl B (1981) A representation and economic interpretation of a two-level programming problem. J Oper Res Soc 32:783–792
11. Hansen P, Jaumard B, Savard G (1992) New branching and bounding rules for linear bilevel programming. SIAM J Sci Statist Comput 13:1194–1217
12. Júdice J, Faustino A (1992) A sequential LCP method for bilevel linear programming. Ann Oper Res 34:89–106
13. Migdalas A, Pardalos PM, Värbrand P (1998) Multilevel optimization: Algorithms and applications. Kluwer, Dordrecht
14. Vicente LN, Calamai PH (1994) Bilevel and multilevel programming: A bibliography review. J Global Optim 5: 291–306
15. Vicente L, Savard G, Júdice (1994) Descent approaches for quadratic bilevel programming. J Optim Th Appl 81: 379–399
16. Visweswaran V, Floudas CA, Ierapetritou MG, Pistikopoulos EN (1996) A decomposition-based global optimization approach for solving bilevel linear and quadratic programs. In: Floudas CA, Pardalos PM (eds) State of the Art in Global Optimization. Kluwer, Dordrecht, pp 139–162

Bilevel Programming: Implicit Function Approach

BP

Stephan Dempe
Freiberg University Mining and Technol., Freiberg, Germany

MSC2000: 90C26, 90C31, 91A65

Article Outline

Keywords
Reformulation as a One-Level Problem
Properties of the Solution Function
Optimality Conditions
  Conditions Using the Directional Derivative of the Solution Function
  Conditions Using the Generalized Jacobian of the Solution Function

Solution Algorithms
Descent Algorithms
Bundle Algorithms
See also
References

Keywords

Bilevel programming problem; Stackelberg game; Implicit function approach; Strongly stable solution; Piecewise continuously differentiable function; Necessary optimality conditions; Sufficient optimality conditions; Solution algorithms

The bilevel programming problem is a hierarchical problem in the sense that its constraints are defined in part by a second parametric optimization problem. Let $\Psi(x)$ be the solution set of this second problem (the so-called lower level problem):

$$\Psi(x) := \operatorname*{Argmin}_y \{f(x,y)\colon\ g(x,y) \le 0\}, \tag{1}$$

where f, $g_i \in C^2(\mathbf{R}^n \times \mathbf{R}^m, \mathbf{R})$, $i = 1, \ldots, p$. Then, the *bilevel programming problem* is defined as

$$\text{“}\min_x\text{”}\{F(x,y)\colon\ y \in \Psi(x), x \in X\} \tag{2}$$

with $F \in C^1(\mathbf{R}^n \times \mathbf{R}^m, \mathbf{R})$ and $X \subseteq \mathbf{R}^n$ is closed. Problem (2) is also called the *upper level problem*. The inclusion of equality constraints in the problem (1) is possible without difficulties. If inequalities and/or equations in both x and y appear in the problem (2), this problem becomes even more difficult since these constraints restrict the set $\Psi(x)$ after a solution y out of it has been chosen. This can make the selection of $y \in \Psi(x)$ a posteriori infeasible [6].

The bilevel programming problem can easily be interpreted in terms of *Stackelberg games* which are a special case of them widely used in economics. In Stackelberg games the inclusion of lower level constraints $g(x, y) \le 0$ is replaced by $y \in Y$ where $Y \subseteq \mathbf{R}^m$ is a fixed closed set. Consider two decision makers which select their actions in an hierarchical manner. First the leader chooses $x \in X$ and announces his selection to the follower. Knowing the selection x the follower computes his response $y(x)$ on it by solving the problem (1). Now, the leader is able to evaluate the value of his initial choice by computing $F(x, y(x))$. Having full knowledge about the follower's responses $y(x)$ for all $x \in X$ the leader's task is it to minimize the function $G(x) := F(x, y(x))$ over the set X, i. e.to solve problem (2).

The bilevel programming problem has a large number of applications e. g.in economics, natural sciences, technology (cf. [17,25] and the references therein).

The quotation marks in (2) have been used to indicate that, due to minimization only with respect to x in the upper level problem (2), this problem is not well defined in the case that the lower level problem (1) has not a uniquely determined optimal solution for all values of x [6]. Minimization only with respect to x in (2) takes place in many applications of bilevel programming, e. g.in the cases when the lower level problem represents the reactions of the nature on the leader's actions. If $\Psi(x)$ does not reduce to a singleton for all parameter values $x \in X$, either an optimistic or a pessimistic approach has to be used to obtain a well defined auxiliary problem.

In the optimistic case, problem (2) is replaced by

$$\min_{x,y}\{F(x,y)\colon\ y \in \Psi(x), x \in X\} \tag{3}$$

[6,11], where minimization is taken with respect to both x and y. The use of (3) instead of (2) means that the leader is able to influence the choice of the follower. If the leader is not able to force the follower to take that solution $y \in \Psi(x)$ which is the best possible for him, he has to bound the damage resulting from an unwelcome choice of the follower. Hence, the leader has to take the worst solution in $\Psi(x)$ into account for computing his decision. This leads to the auxiliary problem in the pessimistic case:

$$\min_x \left\{ \max_y \{F(x,y)\colon\ y \in \Psi(x)\} :\ x \in X \right\} \tag{4}$$

[15,16].

In the sequel it is assumed that the lower level problem (1) has a unique (global) optimal solution $y(x)$ for all $x \in X$. This is guaranteed to be true at least if the assumptions C), SCQ), and SSOC) below are satisfied. Then, the *implicit function approach to bilevel programming* can be used which means that problem (2) (and equivalently (3)) is replaced by

$$\min_x \{G(x) := F(x, y(x))\colon\ x \in X\}. \tag{5}$$

C) The functions $f(x,\cdot), g_i(x,\cdot)\colon \mathbf{R}^m \to \mathbf{R}$ are convex in y for each $x \in X$.

SCQ) For each $x \in X$ there exists a point $\widetilde{y}(x)$ such that $g(x, \widetilde{y}(x)) < 0$.

For convex problems, *Slater's condition* SCQ) implies that a feasible point $\overline{y}(x)$ to (1) is optimal if and only if the *Karush–Kuhn–Tucker conditions* for this problem are valid: There exists a point $\lambda \in \Lambda(x, \overline{y}(x))$, where

$$\Lambda(x, \overline{y}(x)) = \{\lambda \geq 0 \colon \nabla_y L(x, \overline{y}(x)) = 0, \lambda^\top g(x, \overline{y}(x)) = 0\} \tag{6}$$

with $L(x, y) = f(x, y) + \lambda^\top g(x, y)$ denoting the Lagrange function of the problem (1).

Reformulation as a One-Level Problem

There are several methods to reformulate (3) as an equivalent one-level problem.

The first possibility consists in replacing the lower level problem (1) by its Karush–Kuhn–Tucker conditions (6):

$$\min_{x,y} \left\{ F(x, y) \colon \begin{array}{c} \nabla_y L(x, y) = 0, \\ \lambda^\top g(x, y) = 0, \\ g(x, y) \leq 0, \\ \lambda \geq 0, x \in X \end{array} \right\}. \tag{7}$$

This is an optimization problem with constraints given in part by a parametric complementarity condition.

A second possibility is to use a variational inequality describing the set $\Psi(x)$. Let assumption C) be satisfied. Then, the problem (3) is equivalent to

$$\min_{x,y} \left\{ F(x, y) \colon \begin{array}{c} g(x, y) \leq 0, x \in X, \\ \nabla f(x, y)(z - y) \geq 0 \\ \forall z : g(x, z) \leq 0 \end{array} \right\}. \tag{8}$$

Both approaches (7) and (8) lead to a so-called *mathematical program with equilibrium constraints* (MPEC) [17].

SSOC) For each $x \in X$, for each $y \in \Psi(x)$, for all $\lambda \in \Lambda(x, y)$ and for all $d \neq 0$ satisfying

$$\nabla_y g_i(x, y)d = 0 \text{ for all } i \colon \lambda_i > 0,$$

the following inequality holds:

$$d^\top \nabla^2_{yy} L(x, y, \lambda)d > 0.$$

If at an optimal solution $y(\overline{x})$ of the convex problem (1) at $x = \overline{x}$ the assumptions SCQ) and SSOC) are satisfied, then $y(\overline{x})$ is a *strongly stable optimal solution* in the sense of M. Kojima [13]. This means that there exists an open neighborhood U of $\overline{x}$ and a uniquely determined continuous function $y \colon U \to \mathbf{R}^m$ such that $y(x)$ is the uniquely determined optimal solution of (1) for all $x \in U$. Hence, for convex problems (1), the assumptions SCQ) and SSOC) imply that there is a uniquely determined implicit function $y(x)$ describing the unique optimal solution of the problem (1) for all $x \in X$. This function can be inserted into the problem (2) which results in the third equivalent one-level problem (5). Problem (5) consists in minimizing the implicitly determined, generally nonsmooth, nonconvex objective function $F(x, y(x))$ on the set X. It has an optimal solution if the set X is compact or the function $F(\cdot, \cdot)$ satisfies some coercivity assumption [11].

Under suitable assumptions, the parametric complementarity problem as well as the parametric variational inequality describing the constraints in a mathematical program with equilibrium constraints also possess a uniquely determined continuous solution function [17]. Then, the implicit function approach can also be used to investigate MPECs.

Properties of the Solution Function

For the investigation of bilevel programming problems via (5) the knowledge of properties of the solution function $y \colon X \to \mathbf{R}^m$ is needed. If the assumptions C), SCQ), and SSOC) are satisfied, this function is continuous [13], upper Lipschitz continuous [22], Hölder continuous with exponent 1/2 [9] and directionally differentiable [3,24]. Let $\overline{z} = (\overline{x}, y(\overline{x}))$, $\overline{I} := \{j \colon g_j(\overline{z}) = 0\}$, $J(\lambda) := \{j : \lambda_j > 0\}$. The directional derivative

$$y'(\overline{x}; r) = \lim_{t \to 0+} t^{-1}[y(\overline{x} + tr) - y(\overline{x})]$$

of the function $y(\cdot)$ at a point $\overline{x}$ can be computed as the unique optimal solution $y'(\overline{x}; r)$ of the convex quadratic problem

$$\begin{aligned} &\frac{1}{2} d^\top \nabla^2_{yy} L(\overline{z}, \overline{\lambda})d + d^\top \nabla^2_{xy} L(\overline{z}, \overline{\lambda})r \to \min_d, \\ &\nabla_x g_i(\overline{z})r + \nabla_y g_i(\overline{z})d = 0, \quad \forall\, i \in J(\overline{\lambda}), \\ &\nabla_x g_i(\overline{z})r + \nabla_y g_i(\overline{z})d \leq 0, \quad \forall\, i \in \overline{I} \setminus J(\overline{\lambda}), \end{aligned} \tag{9}$$

for some suitably chosen Lagrange multiplier

$$\overline{\lambda} \in \operatorname*{Argmax}_{\lambda} \{\nabla_x L(\overline{z}, \lambda) r \colon \ \lambda \in \Lambda(\overline{z})\} \tag{10}$$

[3]. The correct choice of $\overline{\lambda}$ is a rather difficult task since it possibly belongs to the relative interior of some facet of the polyhedral set $\Lambda(\overline{z})$ [3]. For making the application of these properties of the solution function easier, a further assumption is used:

CR) For each pair $(\overline{x}, \overline{y})$, $\overline{x} \in X$, $\overline{y} \in \Psi(\overline{x})$, there is an open neighborhood $V \subseteq \mathbf{R}^n \times \mathbf{R}^m$ of $(\overline{x}, \overline{y})$ such that, for all $I \subseteq \overline{I}$, the family of gradients $\{\nabla_y g_i(x, y) : i \in I\}$ has constant rank on V.

If the assumptions C), SCQ), SSOC), and CR) are satisfied, the function $y\colon X \to \mathbf{R}^m$ is a *piecewise continuously differentiable function* [21], i. e. it is continuous and there exist an open neighborhood U of $\overline{x}$ and a finite number of continuously differentiable functions y^i $U \to \mathbf{R}^m$, $i = 1, \ldots, k$, such that $y(\cdot)$ is a selection of the y^i:

$$y(x) \in \left\{y^i(x) \colon \ i = 1, \ldots, k\right\}, \quad \forall\, x \in U.$$

The functions $y^i\colon U \to \mathbf{R}^m$ describe locally optimal solutions of auxiliary problems

$$\min_y \left\{f(x, y) \colon \ g_j(x, y) = 0, j \in I_i\right\},$$

where the sets I_i, $i = 1, \ldots, k$, satisfy the following two conditions:

- there exists a vertex $\overline{\lambda} \in \Lambda(\overline{x}, y(\overline{x}))$ such that $J(\overline{\lambda}) \subseteq I_i \subseteq \overline{I}$; and
- the gradients $\left\{\nabla_y g_j(\overline{x}, y(\overline{x})) \colon \ j \in I_i\right\}$ are linearly independent [14].

Let $IS(\overline{x})$ denote the family of all sets I_i having these two properties. Then, k is the cardinality of $IS(\overline{x})$. The functions $y^i\colon U \to \mathbf{R}^m$ are continuously differentiable at $\overline{x}$ [7]. For the computation of the Jacobian of the function $y^i(\cdot)$ at $x = \overline{x}$ the unique solution of a system of linear equations is to be computed.

Moreover, the directional derivative $y'(\overline{x}; r)$ is equal to the unique optimal solution of the quadratic problem (9) for each optimal solution $\overline{\lambda}$ of the linear problem (10) [21]. For fixed $\overline{x}$, it is a continuous, piecewise linear function of the direction r. The quadratic problem (9) has an optimal solution if and only if $\overline{\lambda}$ solves the linear problem (10). Hence, for computing a linear approximation of the function $y\colon X \to \mathbf{R}^m$ it is sufficient to solve the parametric quadratic optimization problems (9) for all vertices $\overline{\lambda} \in \Lambda(\overline{x}, y(\overline{x}))$.

Piecewise continuously differentiable functions are locally Lipschitz continuous [10]. The generalized Jacobian [1] of the function $y(\cdot)$ satisfies

$$\partial y(\overline{x}) \subseteq \operatorname{conv}\left\{\nabla y^i(\overline{x}) \colon \ i = 1, \ldots, k\right\} \tag{11}$$

[14]. Let $g_I(z) = (g_i(z))_{i \in I}$. If the assumption

FRR) For each $x \in X$, for each vertex $\overline{\lambda} \in \Lambda(\overline{z})$ with $\overline{z} = (\overline{x}, y(\overline{x}))$, the matrix

$$\begin{pmatrix} \nabla^2_{yy} L(\overline{z}, \overline{\lambda}) & \nabla^\top_y g_{J(\overline{\lambda})}(\overline{z}) & \nabla^2_{xy} L(\overline{z}, \overline{\lambda}) \\ \nabla_y g_{\overline{I}}(\overline{z}) & 0 & \nabla_x g_{\overline{I}}(\overline{z}) \end{pmatrix}$$

has full row rank

is added to C), SCQ), SSOC), and CR), then equality holds in (11) [5].

Optimality Conditions

Even under very restrictive assumptions, problem (5) is a nondifferentiable, nonconvex optimization problem. For the derivation of necessary and sufficient optimality conditions, various approaches of nondifferentiable optimization can be used.

Conditions Using the Directional Derivative of the Solution Function

Let $X = \{x\colon h_k(\mathrm{x}) \le 0, k \in K\}$, where $h_k \in C^1(\mathbf{R}^n, \mathbf{R})$, $k \in K$ and K is a finite set. Generalizations of the following results to larger classes of constraint sets are obvious. Let $\overline{x} \in X$, $y(\overline{x}) \in \Psi(\overline{x})$, $\overline{z} = (\overline{x}, y(\overline{x}))$. Let the assumptions C), SCQ), SSOC), and CR) as well as

MFCQ) There exists a direction d such that $\nabla h_k(\overline{x}) d < 0$ for all $k \in \overline{K} := \{l \colon \ h_l(\overline{x}) = 0\}$

be valid. Then, if $\overline{x}$ is a locally optimal solution of the problem (5) (and thus of the bilevel problem (2)), there cannot exist a feasible direction of descent, i. e.

$$\nabla_x F(\overline{z}) r + \nabla_y F(\overline{z}) y'(\overline{x}; r) \ge 0 \tag{12}$$

for all directions r satisfying $\nabla h_k(\overline{x}) \le 0$, $k \in \overline{K}$. By use of the above approach for computing the directional derivative of the solution function $y(\cdot)$, the verification of this *necessary optimality condition* can be done by solving a bilevel optimization problem of minimizing the function (12) subject to the condition that $y'(\overline{x}; r)$

is an optimal solution of the problem (9). By replacing problem (9) with its Karush–Kuhn–Tucker conditions and applying an active index set strategy the following condition is obtained: If $\overline{x}$ is a locally optimal solution of the problem (2) then

$$v := \min\{\varphi(\overline{x}, I):\ I \in IS(\overline{x})\} \geq 0, \tag{13}$$

where $\varphi(\overline{x}, I)$ denotes the optimal objective function value of the problem

$$\begin{aligned}
&\nabla_x F(\overline{z})r + \nabla_y F(\overline{z})d \to \min_{d,r,\alpha},\\
&\nabla_x h_k(\overline{x})r \leq 0, \qquad k \in \overline{K},\\
&\nabla^2_{xy} L(\overline{z}, \overline{\lambda})r + \nabla^2_{yy} L(\overline{z}, \overline{\lambda})d + \nabla_y^{\top} g_I(\overline{z})\alpha = 0,\\
&\nabla_x g_i(\overline{z})r + \nabla_y g_i(\overline{z})d = 0, \qquad i \in I,\\
&\nabla_x g_i(\overline{z})r + \nabla_y g_i(\overline{z})d \leq 0, \qquad i \in \overline{I} \setminus I,\\
&\alpha_i \geq 0, \quad i \in I \setminus J(\overline{\lambda}), \qquad \|r\| = 1,
\end{aligned}$$

and $\overline{\lambda}$ is the unique vertex of $\Lambda(\overline{z})$ with $J(\overline{\lambda}) \subseteq I$ [2]. Problem (13) is a combinatorial optimization problem and can be solved by enumeration algorithms.

In [2] a more general necessary optimality condition is given even without assuming CR). Then, the directional derivative of the solution function is in general discontinuous with respect to perturbations of the direction and is to be replaced by the contingent derivative of the solution function.

In [17] it is shown that nonexistence of directions of descent in the tangent cone to the feasible set is also a necessary optimality condition for MPECs. In general, this tangent cone is not convex. Using a so-called basic constraint qualification it is shown that it is equal to the union of a finite number of polyhedral cones. The resulting condition is similar to (13). Dualizing this condition, some kind of a Karush–Kuhn–Tucker condition for MPECs is obtained.

It is also possible to obtain a *sufficient optimality condition* by use of the directional derivative. Namely, if for the optimal function value in (13) the strict inequality $v > 0$ holds then, for each $c \in (0, v)$, there exists $\varepsilon > 0$ such that

$$F(x, y(x)) \geq F(\overline{x}, y(\overline{x})) + c\,\|x - \overline{x}\|$$

for all x satisfying $h(x) \leq 0$ and $\|x - \overline{x}\| \leq \varepsilon$ [2]. Necessary and sufficient optimality conditions of second order based on the implicit function approach (applied to the more general MPEC formulation) are given in [17].

Conditions Using the Generalized Jacobian of the Solution Function

By [1], the generalized differential of the function $G(x) := F(x, y(x))$ is equal to

$$\partial G(\overline{x}) = \operatorname{conv}\left\{\nabla_x F(\overline{z}) + \nabla_y F(\overline{z})\omega:\ \omega \in \partial y(\overline{x})\right\}, \tag{14}$$

provided that the conditions C), SCQ), SSOC), and CR) are satisfied. Hence, the application of the necessary optimality conditions from Lipschitz optimization to problem (5) leads to necessary optimality conditions for the bilevel problem (2). Thus, if $\overline{x}$ is a locally optimal solution of the problem (2) and the assumptions C), SCQ), SSOC), CR), and MFCQ) are satisfied, then there exist Lagrange multipliers $\gamma_i \geq 0$, $i \in \overline{K}$, such that

$$0 \in \partial G(\overline{x}) + \sum_{i \in \overline{K}} \gamma_i \{\nabla h_i(\overline{x})\}.$$

This is an obvious generalization of the necessary optimality condition given in [4], where no upper level constraints in (2) appeared, and is also a special case of the results in [19], where the general constraint set $x \in X$ in the upper level problem (2) together with more restrictive assumptions for the lower level problem are used. For the use of this necessary optimality condition in computations the explicit description of the generalized Jacobian in (11) (with equality instead of inclusion) is needed.

Solution Algorithms

The implicit function approach leads to the problem (5) of minimizing a nondifferentiable, nonconvex, implicitly determined function on a fixed set. Any algorithm solving nonsmooth optimization problems can be applied to this problem. Due to the structure of (5) the computation of function values and derivative information for the objective function is expensive. Two types of algorithms are proposed: descent and bundle algorithms. The convergence proofs show that the algorithms converge to points where the above optimality conditions are satisfied, i. e. to solutions where no descent direction exists respectively to Clarke stationary points.

Descent Algorithms

Let

$$X = \{x:\ h_k(x) \le 0,\ k \in K\}.$$

Descent algorithms are iterative methods which compute a sequence of feasible points $\{x^i\}_{i \in \mathrm{N}}$ by $x^{i+1} = x^i + t_i\, r^i$, $\forall i$, where r^i is a feasible direction of descent and t_i is a stepsize. For bilevel problems a feasible direction of descent is obtained by minimizing the function (12)

$$\nabla_x F(\overline{z})r + \nabla_y F(\overline{z})y'(\overline{x};r)$$

subject to r being an inner direction of the cone of feasible directions to X:

$$\min_{\alpha,r} \{\alpha:\ \nabla_x F(\overline{z})r + \nabla_y F(\overline{z})y'(\overline{x};r) \le \alpha,\ \nabla h_i(\overline{x})r \le \alpha,\quad i \in \overline{K},\quad \|r\| \le 1\}.$$

Inserting the Karush–Kuhn–Tucker conditions of the quadratic optimization problem (9) for the computation of $y'(\overline{x};r)$ and again using an active set strategy this problem is converted into an equivalent combinatorial optimization problem. For the computation of a stepsize, e. g., Armijo's rule can be applied. Such an algorithm is described in [6,8,17]. In [6] it is also investigated how this idea can be generalized to the case when the lower level problem (1) is not assumed to have a uniquely determined optimal solution for all values of the parameter. In [17] this approach is applied to the more general MPEC.

Bundle Algorithms

Let $X = \mathbf{R}^n$. Different constraint sets can be treated by use of approaches in [12]. As in descent algorithms, in *bundle algorithms* for minimizing Lipschitz nonconvex functions a sequence of iterates $\{x^i\}_{i \in \mathrm{N}}$ with $x^{i+1} = x^i + t_i r^i$, $\forall i$, is computed. For computing a direction a model of the function to be minimized is used. In the paper [23], the following bundle algorithm has been proposed. Let two sequences of points $\{x^i\}_{i=1}^k$, $\{z^i\}_{i=1}^k$ have already been computed. Then, for minimizing a nonconvex function $G(x)$, this model has the form

$$\max_{1 \le i \le k} \{v(z^i)^\top d - \alpha_{k,i}\} + \frac{u^k d^\top d}{2}, \tag{15}$$

where

$$\alpha_{k,i} = \max\left\{G(x^k) - v(z^i)^\top (x^k - z^i) - G(z^i),\ c_0 \left\|x^k - z^i\right\|\right\},$$

$v(z^i)$ is a subgradient of the function $G(x)$ at $x = z^i$ and u^k is a weight. If the direction computed by minimizing the model function (15) realizes a sufficient decrease, a serious step is made (i. e. $t_k = 1$ is used). Otherwise, either a short step (which means that t_k is computed according to a stepsize rule) or a null step (only the model is updated by computing a new subgradient) is made. For updating the model (15), in each iteration of the bundle algorithm a subgradient of the objective function is needed. For its computation formula (14) can be used.

The bundle algorithm is applied to problem (5) in [4,18,20]. In [4], the lower level problem is not assumed to have a uniquely determined optimal solution for all parameter values. The Lipschitz optimization problem (5) is obtained via a regularization approach in the lower level problem (1).

Numerical experience for solving bilevel problems (in the formulation (2) as well as in the more general MPEC formulation) with the bundle algorithm is reported in [18,20].

See also

- Bilevel Fractional Programming
- Bilevel Linear Programming
- Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- Bilevel Optimization: Feasibility Test and Flexibility Index
- Bilevel Programming
- Bilevel Programming: Applications
- Bilevel Programming: Applications in Engineering
- Bilevel Programming: Introduction, History and Overview
- Bilevel Programming in Management
- Bilevel Programming: Optimality Conditions and Duality
- Multilevel Methods for Optimal Design
- Multilevel Optimization in Mechanics
- Stochastic Bilevel Programs

References

1. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
2. Dempe S (1992) A necessary and a sufficient optimality condition for bilevel programming problems. Optim 25:341–354
3. Dempe S (1993) Directional differentiability of optimal solutions under Slater's condition. Math Program 59: 49–69
4. Dempe S (1997) An implicit function approach to bilevel programming problems. In: Migdalas A, Pardalos PM, Värbrand P (eds) Multilevel Optimization: Algorithms, Complexity and Applications. Kluwer, Dordrecht
5. Dempe S, Pallaschke D (1997) Quasidifferentiability of optimal solutions in parametric nonlinear optimization. Optim 40:1–24
6. Dempe S, Schmidt H (1996) On an algorithm solving two-level programming problems with nonunique lower level solutions. Comput Optim Appl 6:227–249
7. Fiacco AV, McCormic GP (1968) Nonlinear programming: Sequential unconstrained minimization techniques. Wiley, New York
8. Gauvin J, Savard G (1994) The steepest descent direction for the nonlinear bilevel programming problem. Oper Res Lett 15:265–272
9. Gfrerer H (1987) Hölder continuity of solutions of perturbed optimization problems under Mangasarian-Fromowitz constraint qualification. In: Guddat J et al (eds) Parametric Optimization and Related Topics. Akad Verlag, Berlin, pp 113–127
10. Hager WW (1979) Lipschitz continuity for constrained processes. SIAM J Control Optim 17:321–328
11. Harker PT, Pang J-S (1988) Existence of optimal solutions to mathematical programs with equilibrium constraints. Oper Res Lett 7:61–64
12. Kiwiel KC (1985) Methods of descent for nondifferentiable optimization. Springer, Berlin
13. Kojima M (1980) Strongly stable stationary solutions in nonlinear programs. In: Robinson SM (ed) Analysis and Computation of Fixed Points. Acad Press, New York pp 93–138
14. Kummer B (1988) Newton's method for non-differentiable functions. Adv Math Optim, In: Math Res, vol 45. Akad Verlag, Berlin
15. Loridan P, Morgan J (1989) ϵ-regularized two-level optimization problems: Approximation and existence results. In: Optimization: Fifth French-German Conf (Varez), In: Lecture Notes Math, vol 1405. Springer, Berlin, pp 99–113
16. Lucchetti R, Mignanego F, Pieri G (1987) Existence theorem of equilibrium points in Stackelberg games with constraints. Optim 18:857–866
17. Luo Z-Q, Pang J-S, Ralph D (1996) Mathematical programs with equilibrium constraints. Cambridge Univ Press, Cambridge
18. Outrata J (1990) On the numerical solution of a class of Stackelberg problems. ZOR: Methods and Models of Oper Res 34:255–277
19. Outrata JV (1993) Necessary optimality conditions for Stackelberg problems. J Optim Th Appl 76:305–320
20. Outrata J, Zowe J (1995) A numerical approach to optimization problems with variational inequality constraints. Math Program 68:105–130
21. Ralph D, Dempe S (1995) Directional derivatives of the solution of a parametric nonlinear program. Math Program 70:159–172
22. Robinson SM (1982) Generalized equations and their solutions, Part II: Applications to nonlinear programming. Math Program Stud 19:200–221
23. Schramm H, Zowe J (1992) A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. SIAM J Optim 2:121–152
24. Shapiro A (1988) Sensitivity analysis of nonlinear programs and differentiability properties of metric projections. SIAM J Control Optim 26:628–645
25. Vicente LN, Calamai PH (1994) Bilevel and multilevel programming: A bibliography review. J Global Optim 5(3)

Bilevel Programming: Introduction, History and Overview

BP

LUIS N. VICENTE
Department Mat., University de Coimbra, Coimbra, Portugal

MSC2000: 90C26, 90C30, 90C31

Article Outline

Keywords
See also
References

Keywords

Bilevel programming; Multilevel programming; Hierarchical optimization; Nondifferentiable optimization; Game theory; Stackelberg problems

The *bilevel programming* (BP) problem is a hierarchical optimization problem where a subset of the variables is constrained to be a solution of a given optimization

problem parameterized by the remaining variables. The BP problem is a multilevel programming problem with two levels. The hierarchical optimization structure appears naturally in many applications when lower level actions depend on upper level decisions. The applications of bilevel and multilevel programming include *transportation* (taxation, network design, trip demand estimation), *management* (coordination of multidivisional firms, network facility location, credit allocation), *planning* (agricultural policies, electric utility), and *optimal design*.

In mathematical terms, the BP problem consists of finding a solution to the upper level problem

$$\begin{cases} \min\limits_{x,y} & F(x,y) \\ \text{s.t.} & g(x,y) \le 0, \end{cases}$$

where y, for each value of x, is the solution of the lower level problem:

$$\begin{cases} \min\limits_{y} & f(x,y) \\ \text{s.t.} & h(x,y) \le 0, \end{cases}$$

with $x \in \mathbf{R}^{nx}$, $y \in \mathbf{R}^{ny}$, $F, f : \mathbf{R}^{nx+ny} \to \mathbf{R}$, $g : \mathbf{R}^{nx+ny} \to \mathbf{R}^{nu}$, and $h : \mathbf{R}^{nx+ny} \to \mathbf{R}^{nl}$ (nx, ny, nu, and nl are positive integers). The lower level problem is also referred as the follower's problem or the inner problem. In a similar way, the upper level problem is also called the leader's problem or the outer problem. One could generalize the BP problem in different ways. For instance, if either x or y or both are restricted to take integer values we would obtain an integer BP problem [22]. Or, if we replace the lower level problem by a variational inequality we would get a generalized BP problem [15].

For each value of the upper level variables x, the lower level constraints $h(x, y) \le 0$ define the constraint set $\Omega(x)$ of the lower level problem:

$$\Omega(x) = \{y\colon\ h(x,y) \le 0\}\,.$$

Then, the set $M(x)$ of solutions for the lower level problem is given by minimizing the lower level function $f(x, y)$ for all values in $\Omega(x)$ of the lower level variables y:

$$M(x) = \{y\colon\ y \in \operatorname{argmin}\{f(x,y)\colon\ y \in \Omega(x)\}\}\,.$$

Given these definitions the BP problem can be reformulated as:

$$\begin{cases} \min\limits_{x,y} & F(x,y) \\ \text{s.t.} & g(x,y) \le 0, \\ & y \in M(x). \end{cases}$$

The feasible set

$$\{(x,y)\colon\ g(x,y) \le 0,\ y \in M(x)\}$$

of the BP problem is called the induced or inducible region. The induced region is usually nonconvex and, in the presence of upper level constraints, can be disconnected or even empty. In fact, consider the following BP problem

$$\begin{cases} \min\limits_{x,y} & x - 2y \\ \text{s.t.} & -x + 3y - 4 \le 0, \end{cases}$$

where y, for each value of x, is the solution of:

$$\begin{cases} \min\limits_{y} & x + y \\ \text{s.t.} & x - y \le 0, \\ & -x - y \le 0. \end{cases}$$

For this problem we have:

$$\Omega(x) = \{y\colon\ y \ge |x|\}$$

and

$$M(x) = |x|\,.$$

Thus, the induced region is given by:

$$\begin{aligned} &\{(x,y)\colon\ -x + 3y - 4 \le 0,\ y \in M(x)\} \\ &= \{(x,y)\colon\ y = -x,\ -1 \le x \le 0\} \\ &\cup \{(x,y)\colon\ y = x,\ 0 \le x \le 2\}\,, \end{aligned}$$

which is nonconvex but connected. If the upper level constraints were changed to

$$\begin{aligned} &-x + 3y - 4 \le 0, \\ &-y + \frac{1}{2} \le 0, \end{aligned}$$

then the induced region would become

$$\begin{aligned} &\left\{(x,y)\colon\ y = -x,\ -1 \le x \le -\frac{1}{2}\right\} \\ &\cup \left\{(x,y)\colon\ y = x,\ \frac{1}{2} \le x \le 2\right\}, \end{aligned}$$

which would be a disconnected set. In either case the BP problem has two local minimizers $(-1, 1)$ and $(2, 2)$ and one global minimizer $(-1, 1)$.

This simple example illustrates many features of bilevel programming like the nonconvexity and the disconnectedness of the induced region and the existence of different local minimizers. In this example the induced region is compact. In fact, compactness of the induced region is important for the existence of a global minimizer and can be guaranteed under appropriate conditions [9].

The original formulation for bilevel programming appeared in 1973, in a paper authored by J. Bracken and J. McGill [5], although it was W. Candler and R. Norton [7] who first used the designation bilevel and multilevel programming. However, it was not until the early 1980s that these problems started receiving the attention they deserve. Motivated by the game theory of H. Stackelberg [20], several authors studied bilevel programming intensively and contributed to its proliferation in the mathematical programming community.

The theory of bilevel programming focuses on forms of optimality conditions and complexity results. A number of authors ([8,16], just to cite a few) have established original forms of optimality conditions for bilevel programming by either considering reformulations of the BP problem or by making use of nondifferentiable optimization concepts or even by appealing to the geometry of the induced region. The complexity of the problem has been addressed by a number of authors. It has been proved that even the linear BP problem, where all the involved functions are affine, is a strongly *NP*-hard problem [10]. It is not hard to construct a linear BP problem where the number of local minima grows exponentially with the number of variables [6]. Other theoretical results of interest have been established connecting bilevel programming to other fields in mathematical programming. For instance, one can show that minimax problems and linear, integer, bilinear and quadratic programming problems are special cases of BP. Other classes of problems different from but related to BP are multi-objective optimization problems and static Stackelberg problems. See [21] for references in these topics.

Many researchers have designed algorithms for the solution of the BP problem. One class of techniques consists of extreme point algorithms and has been mostly applied to the linear BP problem because for this problem, if there is a solution, then there is at least one global minimizer that is an extreme point of Ω [17]. Two other classes of algorithms are branch and bound algorithms and complementarity pivot algorithms that have in common the fact that exploit the complementarity part of the necessary optimality conditions of the lower level problem (assumed convex in y so that the necessary optimality conditions, under an appropriate constraint qualification, are also sufficient). These two classes of algorithms have been applied mostly to the case where the upper level is linear and the lower level is linear or convex quadratic (see for instance [10] and [12]) and, as the extreme point algorithms, find a global minimizer of the BP problem. On the other hand, the algorithms designed to solve nonlinear forms of BP appeal to descent directions (see, among others [14] and [18]) and penalty functions (for instance [1]) and are expected to find a local minimizer.

For additional material about bilevel programming, see the books [3,19], the survey papers [2,4,11,13,23], and the bibliography review [21].

See also

- ▶ Bilevel Fractional Programming
- ▶ Bilevel Linear Programming
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Bilevel Programming
- ▶ Bilevel Programming: Applications
- ▶ Bilevel Programming: Applications in Engineering
- ▶ Bilevel Programming: Implicit Function Approach
- ▶ Bilevel Programming in Management
- ▶ Bilevel Programming: Optimality Conditions and Duality
- ▶ Multilevel Methods for Optimal Design
- ▶ Multilevel Optimization in Mechanics
- ▶ Stochastic Bilevel Programs

References

1. Aiyoshi E, Shimizu K (1984) A solution method for the static constrained Stackelberg problem via penalty method. IEEE Trans Autom Control 29:1111–1114

2. Anandalingam G, Friesz T (1992) Hierarchical optimization: An introduction. Ann Oper Res 34:1–11
3. Bard JF (1998) Practical bilevel optimization: Algorithms and applications. Kluwer, Dordrecht
4. Ben-Ayed O (1993) Bilevel linear programming. Comput Oper Res 20:485–501
5. Bracken J, McGill J (1973) Mathematical programs with optimization problems in the constraints. Oper Res 21: 37–44
6. Calamai P, Vicente LN (1993) Generating linear and linear-quadratic bilevel programming problems. SIAM J Sci Statist Comput 14:770–782
7. Candler W, Norton R (1977) Multilevel programming. Techn Report World Bank Developm Res Center, Washington DC 20
8. Dempe S (1992) A necessary and a sufficient optimality condition for bilevel programming problems. Optim 25:341–354
9. Edmunds T, Bard J (1991) Algorithms for nonlinear bilevel mathematical programming. IEEE Trans Syst, Man Cybern 21:83–89
10. Hansen P, Jaumard B, Savard G (1992) New branch-and-bound rules for linear bilevel programming. SIAM J Sci Statist Comput 13:1194–1217
11. Hsu S, Wen U (1989) A review of linear bilevel programming problems. Proc Nat Sci Council Report China, Part A: Physical Sci Eng 13:53–61
12. Júdice J, Faustino A (1994) The linear-quadratic bilevel programming problem. INFOR 32:87–98
13. Kolstad C (1985) A review of the literature on bi-level mathematical programming. Techn Report Los Alamos Nat Lab LA-10284-MS/US-32
14. Kolstad C, Lasdon L (1990) Derivative evaluation and computational experience with large bilevel mathematical programs. J Optim Th Appl 65:485–499
15. Marcotte P, Zhu D (1996) Exact and inexact penalty methods for the generalized bilevel programming problem. Math Program 74:142–157
16. Outrata J (1994) On optimization problems with variational inequality constraints. SIAM J Optim 4:340–357
17. Savard G (1989) Contributions à la programmation mathématique à deux niveaux. PhD Thesis Ecole Polytechn. Univ. Montréal
18. Savard G, Gauvin J (1994) The steepest descent direction for the nonlinear bilevel programming problem. Oper Res Lett 15:275–282
19. Shimizu K, Ishizuka Y, Bard JF (1997) Nondifferentiable and two-level mathematical programming. Kluwer, Dordrecht
20. Stackelberg H (1952) The theory of the market economy. Oxford Univ. Press, Oxford
21. Vicente LN, Calamai PH (1994) Bilevel and multilevel programming: A bibliography review. Jogo 5:291–306
22. Vicente LN, Savard G, Júdice J (1996) The discrete linear bilevel programming problem. J Optim Th Appl 89: 597–614
23. Wen U, Hsu S (1991) Linear bi-level programming problems: A review. J Oper Res Soc 42:125–133

Bilevel Programming in Management

JONATHAN F. BARD
University Texas, Austin, USA

MSC2000: 90-01, 91B52, 91B74, 91B32, 90B30, 90B50

Article Outline

Keywords
Multilevel Model
Applications
Solutions
See also
References

Keywords

Bilevel programming; Hierarchical optimization; Stackelberg game; Production planning; Government regulation; Applications; Management

Decision-making in large, hierarchical organizations rarely proceeds form a single point of view. Two of the most prominent aspects of such organizations are specialization closely followed by coordination. The former arises from a practical need to isolate individual jobs or operations and to assign them to specialized units. This leads to departmentalization; however, to accomplish the overall task, the specialized units must be coordinated. The related process divides itself naturally into two parts:

i) the establishment of individual goals and operating rules for each unit; and
ii) the enforcement of these rules within the work environment.

The first deals with the selection of appropriate divisional or lower level performance criteria and, more generally, the selection of the modes of coordination and control. The second relates to the choice of coordination inputs.

An important control variable in the theory of departmentalization is the degree of self-containment of

the organization units. A unit is self-contained to the extent and degree that the conditions of carrying out its activities are independent of what is done elsewhere in the system. The corporate or higher level unit is then faced with the coordination problem of favorably resolving the divisional unit interactions. Mathematical programming has often been used as the basis for modeling these interactions with decomposition techniques providing solutions to problems of large scale (see, e. g., [9]). The central idea underlying decomposition techniques is very simple and can be envisioned as the following algorithmic process: top management, with its set of goals, asks each division of the company to calculate and submit an optimal production plan as though it were operating in isolation. Once the plans are submitted, they are modified with the overall benefit of the company in mind. Marginal profit figures are used to successively reformulate the divisional plans at each stage in the algorithm. An output plan ultimately emerges which is optimal for the company as a whole and which therefore represents the solution to the original programming problem.

Although this procedure attempts to mimic corporate behavior it fails on two counts. The first relates to the assumption that it is possible to derive a single objective or *utility function* which adequately captures the goals of both top management and each subordinate division. The second stems from lack of communications among the components of the organization; at an intermediary stage of the calculations there is no guarantee that each division's plan will satisfy the corporate constraints. In particular, if the production of some output by division k imposes burdens on other divisions by using up a scarce company resource, or by causing an upward shift in the cost functions pertaining to some other company operation, division k's calculation is likely to lead it to overproduce this item from the point of view of the company because the costs to other divisions will not enter its accounts. This is the classical problem of external diseconomies. Similarly, if one of division k's outputs yields external economies where a rise in its production increases the profitability of other divisions, division k may (considering just its own gains in its calculations) not produce enough of this product to maximize the company's profits as a whole. This may result in a final solution that does not realistically reflect the production plan that probably would have been achieved had each division been given the degree of autonomy it exercises in practice.

Another way of treating the multilevel nature of the resource allocation problem is through goal programming. T. Ruefli [11] was the first to apply this technique by proposing a generalized goal decomposition model. Others expanded on his work developing models capable of representing a wide range of operational characteristics including informational autonomy, interdependent strategies, and *bounded rationality* or individual goals. Combinations of these models have been used to solve problems related to government regulation, distribution, and control [9]. In [3], J.F. Bard presents an approach that derives from the complementary strategies of two-stage optimization [7], ► Bilevel linear programming, [10] and *equilibrium analysis* [13]. Decision-making between levels is assumed to proceed sequentially but with some amount of independence to account for the divergence of corporate and subordinate objectives. At the divisional level each unit simultaneously attempts to maximize its own production function and, in so doing, produces a balance of opposing forces. An example based on an integrated paper company operating three divisions is given to illustrate the differences between centralized and decentralized control. The corporate unit has little direct control over divisional schedules but may set internal transfer prices which affect production capacity and profits.

Multilevel Model

A distinguishing characteristic of multilevel systems is that the decision maker at one level may be able to influence the behavior of a decision maker at another level but not completely control his actions. In addition, the objective functions of each unit may, in part, be determined by variables controlled by other units operating at parallel or subordinate levels. For example, policies affected by corporate management relating to resource allocation and benefits may curtail the set of strategies available to divisional management. In turn, polices adopted at the lower levels affecting productivity and marketing may play a role in determining overall profitability and growth. W.F. Bialas and M.H. Karwan [7] have noted the following common features of multilevel organizations:

1) interactive decision-making units exist within a predominately hierarchical structure;
2) each subordinate level executes its policies after, and in view of, decisions made at a superordinate level;
3) these extramural effects enter a decision maker's problem through his objective function and feasible strategy set.

The need for specialization and decentralization has traditionally been met by the establishment of profit centers. In this context, divisions or departments are viewed as more or less independent units charged with the responsibility of operating in the best possible manner so as to maximize profit under the given constraints imposed by top management. The problem of decentralization is essentially how to design and impose constraints on the department units so that the well-being of the overall corporation is assured. The traditional way to coordinate decentralized organizations is by means of the pricing mechanism; coordination is designed by analogy with the operation of a free market or competitive economy. Exchange of products between departments is allowed and internal prices are specified for the exchange commodities. The problem of effective decentralization reduces to the selection of the internal prices.

The framework presented in this article is an extension of the bilevel programming problem introduced in ► Bilevel linear programming, and embodies a corporate management unit at the higher level and M divisions or subordinate units at the lower level. The latter may be viewed as either separate operating divisions of an organization or coequal departments within a firm, such as production, finance, and sales. This structure can be extended beyond two levels (e. g., see [4,9]) with the realization that attending behavioral and operational relationships become much more difficult to conceptualize and describe.

To formulate the problem mathematically, suppose the higher level decision maker wishes to maximize his objective function F and each of the M divisions wishes to maximize its own objective function f^i. Control of the decision variables is partitioned among the units such that the higher level decision maker may select a vector $x^0 \in S^0 \subset \mathbf{R}^{n_0}$ and each lower level decision maker may select a vector $x^i \in S^i \subset \mathbf{R}^{n_i}$ $i = 1, \ldots, M$. Letting $x \equiv (x^1, \ldots, x^M)$ and $n = \sum_{i=1}^{M} n^i$, in the most general case we have $F, f_1, \ldots, f_M$: $\mathbf{R}^n \to \mathbf{R}^1$. It shall be assumed that the corporate unit has the first choice and selects a strategy $x^0 \in S^0$, followed by the M subordinate units who select their strategies $x^i \in S^i$, simultaneously. In addition, the choice made at the higher level may affect the set of feasible strategies available at the lower level, while each lower-level decision maker may influence the choices available to his peers. The strategies sets will be given explicitly by

$$S^0 = \left\{x^0 :\ g(x^0, \bar{x}^0) \le 0\right\},$$
$$S^i = \left\{x^i :\ g(x^i, \bar{x}^i) \le 0\right\},\ i = 1, \ldots, M,$$

where $\bar{x}^i \equiv (x^0, x^1, \ldots, x^{i-1}, x^{i+1}, \ldots, x^M)$ and g^i: $\mathbf{R}^n \to \mathbf{R}^{m_i}$, $i = 1, \ldots, M$.

To assure that the problem is well posed, it is common to assume that all functions are twice continuously differentiable and that the sets S^i, $i = 0, \ldots, M$, are nonempty and compact; i. e., the ith unit always has some recourse. The bilevel multidivisional programming problem (BMPP) can now be defined:

$$\max_{x^0} \quad F(x^0, \bar{x}^0), \tag{1}$$

$$\text{s.t.} \quad g^0(x^0, \bar{x}^0) \le 0, \tag{2}$$

$$\max_{x^i} \quad f^i(x^i, \bar{x}^i), \qquad i = 1, \ldots, M, \tag{3}$$

$$\text{s.t.} \quad g^i(x^i, \bar{x}^i) \le 0, \tag{4}$$

When $M = 0$, problem (1)–(4) reduces to a standard mathematical program; when $M = 1$ a *bilevel program* results; when (1) is removed an equilibrium programming problem remains [13]. A solution to the latter is often taken as an equilibrium point; call it $x_E = (x^i, \bar{x}^i)$, where x^i solves subproblem (3)–(4), $i = 1, \ldots, M$, for $\bar{x}^i$ given. Thus, x_E represents a point of stability. No incentive exists at x_E for any of the divisions to deviate from $\bar{x}^i$ because each has optimized its individual objective function. For the linear BMPP, results similar to those presented for the linear bilevel programming problem in ► Bilevel linear programming hold (see [3]).

Applications

Most applications of bilevel programming, including bilevel multidivisional programming, that have appeared in the literature have dealt with central economic planning at the regional or national level. In this

context, the government is considered the leader and controls a set of policy variables such as tax rates, subsidies, import quotas, and price supports (e. g., see [5] and accompanying papers). The particular industry targeted for regulation is viewed as the follower. In most cases, the follower tries to maximize net income subject to the prevailing technological, economic, and governmental constraints. Possible leader objectives include maximizing employment, maximizing production of a given product, or minimizing the use of certain resources.

The early work of W. Candler and R. Norton [8], focusing on agricultural development in northern Mexico, illustrates how bilevel programming can be used to analyze the dynamics of a regulated economy. Similarly, J. Fortuny-Amat and B. McCarl [10] present a regional model that pits fertilizer suppliers against local farm communities, while E. Aiyoshi and K. Shimizu [1] and Bard [3] discuss resource allocation in a decentralized firm. In the case of the latter, a central unit supplies resources to its manufacturing facilities which make decisions concerning production mix and output. Organizational procedures and conflicting objectives over efficiency, quality and performance lead to a hierarchical formulation. In a work related to the original Stackelberg model of a single leader-follower oligopolistic market in which a few firms supply a homogeneous product, H.D. Sherali [12] presents an extension to N leader firms and discusses issues related to the existence, uniqueness, and derivation of equilibrium solutions. His analysis provides sufficient conditions for some useful convexity and differentiability properties of the followers' reaction curves.

In a recent study [5], the French government has used bilevel programming to examine the economics of promoting biofuel production from farm crops within the petro-chemical industry. The stumbling block to this policy is that industry's costs for producing fuels from hydrocarbon-based raw materials is significantly less than it is for producing biofuels. Without incentives in the form of tax credits, industry will not buy farm output for conversion. The problem faced by the government is to determine the level of tax credits for each final product or biofuel that industry can produce while minimizing public outlays. A secondary objective is to realize some predefined level of land usage for nonfood crops. Industry is assumed to be neutral in this scenario and will produce any biofuel that is profitable. In the model, the agricultural sector is represented by a subset of farms in an agriculturally intensive region of France and is a profit maximizer. It will use the land available for nonfood crops only as long as the revenue generated from this activity exceeds the difference between the set-aside payments now received directly from the government and the maintenance costs incurred under the current support program. The resultant bilevel model contains 3628 variables and 3230 constraints at the lower level, and 8 variables and 10 constraints at the upper level. Both objective functions are quadratic and all constraints are linear.

In an earlier effort, G. Anandalingam and V. Apprey [2] investigated the problem of conflict resolution by postulating the existence of an arbitrator who acts as the leader in a Stackelberg game. They presented models for different configurations of the resulting multilevel linear programs and proposed a series of solution algorithms. The models were illustrated with an application involving a water conflict problem between India and Bangladesh; it is shown that both parties could gain by the arbitration of an international agency such as the United Nations.

Recently, researchers have tried to apply bilevel models to the *network design problem* arising in transportation and telecommunications systems. In the accompanying formulation, a central planner controls investment costs at the system level, while operational costs depend on traffic flows which are determined by the individual user's route selection. Because users are assumed to make decisions so as to maximize their individual utility functions, their choices do not necessarily coincide (and may, in fact, conflict) with the choices that are optimal for the system. Nevertheless, the central planner can influence the users' choices by improving some links to make them relatively more attractive than the others. In deciding on these improvements, the central planner tries to influence the users' preferences in such a way that total costs are minimized. The partition of the control variables between the upper and lower levels naturally leads to a bilevel formulation.

A conceptual framework for the optimization of Tunisia's inter-regional highways was proposed in [6]. The accompanying formulation included 2683 variables (2571 at the lower level) and 820 constraints (all at the lower level); the follower's problem was divided

into two separate subproblems as a direct consequence of the bilevel approach. The first centered on the user-optimized flow requirement (user-equilibrium) and the second on the nonconvex improvement functions. Because none of the standard algorithmic approaches could handle problems of this size, a specialized algorithm was devised to deal with each of the two lower-level problems separately. At each iteration, the algorithm tries to find a better compromise with the user, while including the smallest possible number of nonconvex improvement functions to get the exact solution with the minimum computational effort. Despite the large number of variables and constraints, optimality was achieved.

Solutions

An assessment of existing algorithms for solving various classes of bilevel programs indicates that exact solutions can only be guaranteed for problem instances with up to a few hundred variables and constraints, and then only for the linear case. When nonlinear (nonconvex) functions are included in the model, virtually all algorithms stumble in the presence of more than a handful of variables and constraints. The ability of those working in the field to formulate problems far outstrips the capacity of current techniques to solve them optimally.

When faced with the problem of actually having to provide solutions to large scale formulations, researchers have inevitably fallen back on heuristics and ad hoc procedures. Simulated annealing, tabu search and genetic algorithm-based approaches are examples of the more formal techniques adapted, at least for the linear case. In many instances, code developers were able to demonstrate global optimality by comparing results with exact methods. The conclusion that can be drawn from these observations and related experience is that the need for efficient algorithms remains undiminished. This is the primary reason why realistic applications continue to lag behind theory and the development new codes.

See also

- ► Bilevel Fractional Programming
- ► Bilevel Linear Programming
- ► Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ► Bilevel Optimization: Feasibility Test and Flexibility Index
- ► Bilevel Programming
- ► Bilevel Programming: Applications
- ► Bilevel Programming: Applications in Engineering
- ► Bilevel Programming: Implicit Function Approach
- ► Bilevel Programming: Introduction, History and Overview
- ► Bilevel Programming: Optimality Conditions and Duality
- ► Multilevel Methods for Optimal Design
- ► Multilevel Optimization in Mechanics
- ► Stochastic Bilevel Programs

References

1. Aiyoshi E, Shimizu K (1981) Hierarchical decentralized systems and its new solution by a barrier method. IEEE Trans Syst, Man Cybern SMC-11(6):444–449
2. Anandalingam G, Apprey V (1991) Multi-level programming and conflict resolution. Europ J Oper Res 51:233–247
3. Bard JF (1983) Coordination of a multidivisional organization through two levels of management. OMEGA Internat J Management Sci 11(1):457–468
4. Bard JF (1985) Geometric and algorithmic developments for a hierarchical planning problem. Europ J Oper Res 19:372–383
5. Bard JF, Plummer J, Sourie JC (1997) Determining tax credits for converting nonfood crops to biofuels: An application of bilevel programming. In: Migdalas A, Pardalos PM, Varbrand P (eds) Multilevel Optimization: Algorithms and Applications. Kluwer, Dordrecht, pp 23–50
6. Ben-Ayed O, Blair CE, Boyce DE, Leblanc LJ (1992) Construction of a real-world bilevel programming model of the highway network design problem. Ann Oper Res 34(1–4):219–254
7. Bialas WF, Karwan MH (1984) Two-level linear programming. Managem Sci 30(8):1004–1020
8. Candler W, Norton R (1977) Multi-level programming and development policy. Working Paper 258, World Bank, Washington, DC
9. Dirickx YMI, Jennergren LP (1979) Systems analysis with multilevel methods with applications to economics and management. Wiley, New York
10. Fortuny-Amat J, McCarl B (1981) A representation and economic interpretation of a two-level programming problem. J Oper Res Soc 32(9):783–792
11. Ruefli T (1971) A generalized goal decomposition model. Managem Sci 17(9):B505–B518
12. Sherali HD (1984) A multiple leader Stackelberg model and analysis. Oper Res 3(2):390–404
13. Zangwill WI, Garcia CB (1981) Pathways to solutions, fixed points, and equilibra. Prentice-Hall, Englewood Cliffs, NJ

Bilevel Programming: Optimality Conditions and Duality

SANJO ZLOBEC
Department Math. Statist., McGill University, Montreal, Canada

MSC2000: 90C25, 90C29, 90C30, 90C31

Article Outline

Keywords
Basic Difficulties
Optimality
Parametric Approach To Optimality
Duality
See also
References

Keywords

Bilevel programming; Optimality conditions; Duality; Stability; Parametric programming

The *bilevel programming problem* (abbreviation: BPP) is a mathematical program in two variables x and θ, where $x = x^\circ(\theta)$ is an optimal solution of another program. Specifically, BPP can be formulated in terms of two ordered objective functions φ and Ψ as follows:

$$\begin{cases} \min\limits_{(x,\theta)} & \varphi(x,\theta) \\ \text{s.t.} & f^i(x,\theta) \le 0, \quad i \in P, \end{cases} \tag{1}$$

where $x = x^\circ(\theta)$ is an optimal solution of the program

$$\begin{cases} \min\limits_{(x)} & \Psi(x,\theta) \\ \text{s.t.} & g^j(x,\theta) \le 0, \quad j \in Q. \end{cases} \tag{2}$$

Here the functions φ, Ψ, f^i, $g^j : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}$, $i \in P$, $j \in Q$, are assumed to be continuous; $x \in \mathbf{R}^n$, $\theta \in \mathbf{R}^m$; P, Q are finite index sets. Program (1) is often called the *upper* (*first level*, *outer*, *leader's*) problem; then (2) is the *lower* (*second level*, *inner*, *follower's*) problem. Many mathematical programs, such as minimax problems, linear integer, bilinear and quadratic programs, can be stated as special cases of bilevel programs. In view of the so-called *Reduction Ansatz*, developed in [18,44], *semi-infinite programs* can be considered as special cases of bilevel programs. For stability and deformations of these see, e. g., [20,21]. Problems appearing in such seemingly unrelated areas as best approximation problems and *data envelopment analysis* can be viewed as bilevel programs. In the former, one is often interested in finding a least-norm solution in the set of all best approximate solutions, while, in the latter, one wants to rank, or decrease the number of, efficient decision making units by a 'post-optimality analysis'. For history of *bilevel programs*, reviews of numerical methods and applications, especially for connections with *von Stackelberg games* of market economy see, e. g., [14,22,30,39]. In this contribution we will focus only on optimality conditions and duality.

Basic Difficulties

The study of bilevel programming problems requires some familiarity with point-to-set topology; see, e. g., [1,2,6,15]. Since the lower level *optimal solution mapping* $x^\circ : \theta x^\circ(\theta)$ is a *point-to-set mapping* (rather than a vector function), the optimal value function of the BPP may be discontinuous. This is illustrated by the following example:

Example 1 Consider the bilevel program with the upper level objective $\varphi\ (x, \theta) = -x_1/\theta$, the lower level objective $\Psi\ (x, \theta) = -\ x_1 - x_2$, and the lower level feasible set determined by $x_1 + \theta$, $x_2 \le 1$, $x_1 \ge 0$, $x_2 \ge 0$. The lower level optimal solutions $x = x^\circ(\theta)$ are the segment $\{x_1 + x_2 = 1, x_1 \ge 0, x_2 \ge 0\}$, for $\theta = 1$, and the singleton $[0, 1/\theta]$, when $0 < \theta < 1$. The corresponding upper level optimal solutions, i. e., the BPP optimal solutions, are the points $[1, 0]$ and $[0, 1/\theta]$, respectively. Here the corresponding optimal value of the BPP jumps from -1 to 0, as θ assumes the value 1.

Note that the lower level feasible set mapping, in Example 1, is lower semicontinuous (open) at $\theta = 1$. Hence we conclude that discontinuity of the optimal value can occur even if the lower level model is stable.

The fact that the set of optimal solutions is generally discontinuous in a stable situation is well known in linear programming. It may manifest itself in a chaotic behavior of the optimal solutions, but not the optimal

value, when the program is solved by computer repeatedly with small perturbations of data; see ► Nondifferentiable optimization: Parametric programming. The topological loss of continuity is generally unrelated to the conditioning, which describes numerical sensitivity of the solutions relative to roundoff errors. In particular, a linear program with an *ill-conditioned coefficient matrix* can be stable.

Another difficulty results from the fact that the optimal solutions mapping $x^\circ : \theta \; x^\circ(\theta)$ is not generally closed. Hence a BPP may not have an optimal solution even if the feasible set of the lower program is compact:

Example 2 Consider the bilinear BPP:

$$\min x + \theta,$$

where $x = x^\circ(\theta)$ solves

$$\begin{cases} \min & -x, \\ \text{s.t.} & x\theta = 0, \\ & 0 \le x \le 1, \quad 0 \le \theta \le 1. \end{cases}$$

Here the optimal solutions mapping is the function $x^\circ(\theta) = 0$, if $\theta > 0$, and $x^\circ(0) = 1$, if $\theta = 0$. The feasible set of the lower level problem is a unit square in the (θ, x)-plane, while the feasible set of the BPP is a disjoint noncompact set consisting of the segment $0 < \theta \le 1$ and the point $[0, 1]$. Since the origin is not a feasible point, the BPP does not have a solution. Note that the function $x^\circ(\theta)$ is not continuous here because the lower level feasible set mapping is not lower semicontinuous at the origin, i. e., the lower level problem is unstable.

Optimality

A popular approach to the study of optimality in BPP is to reduce the program to a one-level program. This can be done as follows: Denote the optimal value of the lower level program (2) by $\Psi^\circ(\theta)$ and introduce the new constraint $f^\circ(x, \theta) = \Psi(x, \theta) - \Psi^\circ(\theta)$. Now the BPP can be reformulated as

$$\begin{cases} \min\limits_{(x,\theta)} & \varphi(x, \theta) \\ \text{s.t.} & f^i(x, \theta) \le 0, \\ & i \in R = \{0\} \cup P. \end{cases} \tag{3}$$

Difficulties with this formulation generally include discontinuity of the leading constraint f° and the lack of classical constraint qualifications. The latter can be handled in convex case using the results on optimality conditions from, e. g., [5,15,47]. One of the first attempts to formulate optimality conditions for bilevel programming problems, using (3), was made in [2]. However a counterexample to these conditions was given in [4,12,17], also see [10]. The one-level approach leads, under assumptions that guarantee Lipschitz continuity of the optimal value function, to necessary *conditions of the Fritz John type*. Under a *partial calmness condition*, and a *constraint qualification* for the lower level problem, one obtains *conditions of the Karush–Kuhn–Tucker type*. The concept of partial calmness is equivalent to the 'exact penalization' and it is satisfied, in particular, for the minimax problem and if the lower level problem is linear. This approach in a nonsmooth framework is used in, e. g., [11] and [46]. The relationship between the BPP and an associated *exact penalty function* was explored also in [7] to derive other types of necessary and sufficient optimality conditions. Other approaches to optimality conditions, that use nonsmooth analysis, include [13,19,32]. Another approach to reducing the BPP to a single-level program is to replace the lower level problem by an optimality condition. This is usually done in formulations of numerical methods; see, e. g., [42]. There are also approaches that use the specific geometry of BPP. One of these applies properties of the steepest descent directions to BPP and it yields a necessary condition for optimality, see [33]. Adaptations of the well-known first and second order optimality conditions of mathematical programming to BPP appeared in [40]. Checking local optimality for linear BPP is *NP*-hard; see [41]. Examples of linear BPPs with an exponential number of local minima can be generated by a technique proposed in [9].

Many authors have studied links between two-objective and bilevel programming, looking for conditions that guarantee that the optimal solution of a given BPP be Pareto optimal for both upper and lower level objective functions, and vice versa; e. g., [28,29,30,37]. The idea is to find an optimal solution of the BPP by solving a bi-objective program. It was shown in [43] that an optimal solution in linear BPP may not be a *Pareto optimum* for the objective function of the outer program and the optimal value function of the lower program, contrary to a claim made in [38]. The authors of [43] also give a sufficient condition for the implica-

tion to hold. If an optimal solution exists, in the linear BPP case with a compact feasible set at the lower level, then at least one optimal solution is assumed at a vertex of this set, see [3]. Necessary conditions for optimality can also be stated using marginal value formulas for optimal value functions. However, these formulas can not assume a usual constraint qualification in order to be applied to the formulation (3). One such formula in parametric convex programming is given in [48] and, under slightly different assumptions, in [49]. In the latter, it is used in the context of data envelopment analysis to rank efficiently administered university libraries by their radii of rigidity. Existence of optimal solutions is studied in [16,23,24]; constraints in [24] are defined by an implicit variational problem. Both, existence and stability of solutions and approximate solutions are studied in [27]. Optimality conditions are important for checking optimality, formulation of duality theories, and for numerical methods.

Parametric Approach To Optimality

A parametric approach to characterizing global and local optimal solutions in convex BPP can be described as follows: Denote, for every θ, the optimal value of (3) by

$$\varphi^{\circ}(\theta) = \begin{cases} \min\limits_{(x)} & \varphi(x,\theta) \\ \text{s.t.} & f^i(x,\theta) \le 0, \quad i \in R = \{0\} \cup P. \end{cases}$$

Also, denote the feasible set in the x variable by $F(\theta) = \{x : f^i(x, \theta) \le 0, i \in R$, and the feasible set in the θ variable by

$$F = \{\theta \in R^m : \ F(\theta) \neq \emptyset\}.$$

A parametric formulation of the BPP is

$$\begin{cases} \min & \varphi^{\circ}(\theta) \\ \text{s.t.} & \theta \in F. \end{cases} \tag{4}$$

Here we optimize the optimal value of the outer problem over the feasible set in the variable θ, considered as a 'parameter'. The problem of the form (4) is a basic problem of *parametric programming*, e. g., ► Nondifferentiable optimization: Parametric programming. It has been extensively studied in the literature from both the theoretical and the numerical side. In particular, various optimality conditions have been formulated for it, e. g., in the context of *input optimization*; see [48]. The key observation in the parametric approach is that, under the assumption that the feasible set of the lower program is compact, every θ^* that globally solves the parametric program (4), with the corresponding optimal solution x^* of the program (3), is a global optimal solution of the bilevel program, and vice versa. However, under the compactness assumption, both sets can be empty (as demonstrated by Example 2). A necessary and sufficient condition for global optimality in convex BPP can be given over a *'region of cooperation'* in terms of the existence of a saddle point; see [15]: Given a candidate for global optimality θ^* and the set of all optimal solutions at the lower level $\{x^{\circ}(\theta)\}$, $\theta \in F$. Denote by $K(\theta^*)$ the region in the θ-space, where the minimal index set of active constraints $R^{=}(\theta) = \{i \in R : x \in \{x^{\circ}(\theta)\} \Rightarrow f^i(x, \theta) = 0\}$ does not strictly increase, i. e., $K(\theta^*) = \{\theta \in F : R^{=}(\theta) \subset R^{=}(\theta^*)\}$. Then the region of cooperation at θ^* is defined as the set $\{(\theta, x)\} : \theta \in K(\theta^*), x \in F(\theta)\}$. One can characterize global optimality on the entire feasible set for linear BPP, and also for convex BPP provided that the constraints are *'LFS functions'*, e. g. [35,48]. These functions form a large class of convex functions that includes all linear and polyhedral functions. Characterizations of global optimality are simplified under the so-called *sandwich condition*. This is a two-sided global inclusion involving the set of optimal solutions of the inner program, e. g., [15]. Characterizations of *locally* optimal parameters θ^* for convex (4) require lower semicontinuity of the optimal solutions mapping x°. The results apply to the convex BPP with the additional assumption that the corresponding optimal solution $x^* \in \{x^{\circ}(\theta^*\}$ is unique; see, e. g., [15]. The uniqueness assumption in the characterization of local optimality cannot be replaced by the requirement that the set $\{x^{\circ}(\theta^*)\}$ be compact. The following example illustrates a situation where a local optimum of the BPP can not be recovered by the parametric approach.

Example 3 Consider the program $\min \varphi(x, \theta) = x\theta^2$, where x solves $\min \Psi(x, \theta) = 0$, subject to $-1 \le x, \theta \le 1$. Here $x^* = 1, \theta^* = 0$ is a local minimum of the bilevel program. But $\varphi^{\circ}(\theta) = -\theta^2$ and $\theta^* = 0$ is not its local minimum; in fact, it is an isolated global maximum!

Duality

Duality theories for bilevel programming problems can be formulated by adjusting the duality theories of mathematical programming (see, e. g., [34]) to the single-objective model (3). Let us outline how this works using a parametric approach; we follow the ideas from [15]. Instead of a single 'dual' one obtains a collection of several 'subduals', each closely related to the original (primal) program. The number of these subduals is cardinality of the set

$$\Pi = \{\Omega \subset R\colon\ \Omega = R^{=}(\theta) \text{ for some } \theta \in F\}\,.$$

First, with each $\Omega \subset \Pi$, one associates the feasible subregion $\mathbf{S}_\Omega = \{\theta \in F : R^{=}(\theta) = \Omega\}$, the Lagrangian $L_\Omega\ (x, \theta; u) = \varphi\ (x, \theta) + \sum_{i \in R \setminus \Omega} u_i f^i\ (x, \theta)$, and the point-to-set mapping $F_\Omega : F \to R^n$ defined by $F_\Omega\ (\theta) = \{x : f^i\ (x, \theta) \le 0, i \in \Omega\}$. The corresponding *subdual function* is

$$\Phi_\Omega(u) = \inf\{L_\Omega(x, \theta; u)\colon\ \theta \in \mathbf{S}_\Omega,\ x \in F_\Omega(\theta)\}$$

and the subdual (D, Ω) is defined as

$$\sup\left\{\Phi_\Omega(u)\colon\ u \in [\mathbf{S}_\Omega \to R_+^{\operatorname{card} R \setminus \Omega}]\right\}. \tag{5}$$

Here u belongs to the set of all nonnegative vector functions defined on $\mathbf{S}_\Omega$. The duality results, stated for *partly convex programs* in, e. g., [47] can be reformulated for the outer convex model and hence BPP. In particular, if, for some $\Omega \subset \Pi$, $u^* \in [\mathbf{S}_\Omega \to R_+^{\operatorname{card} R \setminus \Omega}]$, and an optimal solution x^* of the inner program for some fixed $\theta^* \in \mathbf{S}_\Omega$, one has $\Phi_\Omega(u^*) = \varphi\ (x^*, \theta^*)$, then u^* solves the subdual (5) and θ^* solves (4) on $\mathbf{S}_\Omega$.

If optimization of the optimal value function in (4) is performed from some fixed 'initial' θ, but using only parameter paths that preserve continuity of the optimal solutions mapping of the lower problem, then we talk about *stable BPP*. This approach, in the convex case, guarantees that the optimal solutions mapping in BPP is closed and that the optimal value function is continuous, thus removing the two basic difficulties mentioned in Section 1. However, the optimal solutions now depend on the initial choice of the parameter and on a particular class of stable paths used. *Stable parametric programming* has been studied in [48], stable BPP is mentioned (but not studied) in [15]; see [36].

See also

- Bilevel Fractional Programming
- Bilevel Linear Programming
- Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- Bilevel Optimization: Feasibility Test and Flexibility Index
- Bilevel Programming
- Bilevel Programming: Applications
- Bilevel Programming: Applications in Engineering
- Bilevel Programming: Implicit Function Approach
- Bilevel Programming: Introduction, History and Overview
- Bilevel Programming in Management
- Multilevel Methods for Optimal Design
- Multilevel Optimization in Mechanics
- Stochastic Bilevel Programs

References

1. Bank B, Guddat J, Klatte D, Kummer B, Tammer K (1982) Nonlinear parametric optimization. Akad Verlag, Berlin
2. Bard J (1984) Optimality conditions for the bilevel programming problem. Naval Res Logist Quart 31:13–26
3. Bard J (1988) Convex two-level optimization. Math Program 40:15–27
4. Ben-Ayed O, Blair C (1990) Computational difficulties of bilevel linear programming. Oper Res 38:556–560
5. Ben-Israel A, Ben-Tal A, Zlobec S (1981) Optimality in nonlinear programming: A feasible directions approach. Wiley/Interscience, New York
6. Berge C (1963) Topological spaces. Oliver and Boyd, Edinburgh
7. Bi Z, Calamai P (1991) Optimality conditions for a class of bilevel programming problems. Techn Report Dept Systems Design Engin, Univ Waterloo, no. 191-O-191291
8. Bracken J, Falk J, McGill J (1974) Equivalence of two mathematical programs with optimization problems in the constraints. Oper Res 22:1102–1104
9. Calamai P, Vicente LN (1993) Generating linear and linear-quadratic bilevel programming problems. SIAM J Sci Statist Comput 14:770–782
10. Candler W (1988) A linear bilevel programming algorithm: A comment. Comput Oper Res 15:297–298
11. Chen Y, Florian M (1995) The nonlinear bilevel programming problem: formulations, regularity and optimality conditions. Optim 32:193–209
12. Clarke P, Westerberg A (1988) A note on the optimality conditions for the bilevel programming problem. Naval Res Logist Quart 35:413–418

13. Dempe S (1992) A necessary and sufficient optimality condition for bilevel programming problems. Optim 25:341–354
14. Dempe S (1997) On the leader's dilemma and a new idea for attacking bilevel programming problems. Preprint Techn Univ Chemnitz
15. Floudas CA, Zlobec S (1998) Optimality and duality in parametric convex lexicographic programming. In: Pardalos PM Migdalas A and Värbrand P (eds) Multilevel Optimization: Algorithms and Applications. Kluwer, Dordrecht, pp 359–379
16. Harker P, Pang J-S (1988) Existence of optimal solutions to mathematical programs with equilibrium constraints. Oper Res Lett 7:61–64
17. Haurie A, Savard G, White D (1990) A note on: An efficient point algorithm for a linear two-stage optimization problem. Oper Res 38:553–555
18. Hettich R, Jongen HTh (1978) Semi-infinite programming: conditions of optimality and applications. In: Stoer J (ed) Optimization Techniques, Part 2. Lecture Notes Control Inform Sci. Springer, Berlin, pp 1–11
19. Ishizuka Y (1988) Optimality conditions for quasi-differentiable programs with applications to two-level optimization. SIAM J Control Optim 26:1388–1398
20. Jongen HTh, Rückmann J-J, Stein O (1998) Generalized semi-infinite optimization: A first order optimality condition and examples. Math Program 83:145–158
21. Jongen HTh, Rückmann J-J (1998) On stability and deformation in semi-infinite optimization. In: Reemtsen R, Rückmann J-J (eds) Semi-Infinite Programming. Kluwer, Dordrecht, pp 29–67
22. Kolstad CD (Oct. 1985) A review of the literature on bilevel mathematical programming. Techn Report Los Alamos Nat Lab no. LA-10284-MS, UC-32
23. Lignola MB, Morgan J (1995) Topological existence and stability for Stackelberg problems. J Optim Th Appl 84: 145–169
24. Lignola MB, Morgan J (1998) Existence of solutions to generalized bilevel programing problem. In: Migdalas A, Pardalos PM, Värbrand P (eds) Multilevel Optimization: Algorithms and Applications. Kluwer, Dordrecht, pp 315–332
25. Liu Y, Hart S (1994) Characterizing an optimal solution to the linear bilevel programming problem. Europ J Oper Res 166:164–166
26. Loridan P, Morgan J (1989) New results on approximate solutions in two-level optimization. Optim 20:819–836
27. Mallozzi L, Morgan J (1995) Weak Stackelberg problem and mixed solutions under data perturbations. Optim 32:269–290
28. Marcotte P, Savard G (1991) A note on Pareto optimality of solutions to the linear bilevel programming problem. Comput Oper Res 18:355–359
29. Migdalas A (1995) When is a Stackelberg equilibrium Pareto optimum? In: Pardalos PM, Siskos Y, Zopounidis C (eds) Advances in Multicriteria Analysis. Kluwer, Dordrecht, pp 175–181
30. Migdalas A, Pardalos PM (1996) Editorial: Hierarchical and bilevel programming. J Global Optim 8:209–215
31. Migdalas A Pardalos PM, Värbrand P (eds) (1998) Multilevel optimization: Algorithms and applications. Kluwer, Dordrecht, pp 29–67
32. Outrata J (1993) Necessary optimality conditions for Stackelberg problems. J Optim Th Appl 76:305–320
33. Savard G, Gauvin J (1994) The steepest descent direction for the nonlinear bilevel programming problem. Oper Res Lett 15:275–282
34. Tammer K, Rückmann J-J (1990) Relations between the Karush–Kuhn–Tucker points of a nonlinear optimization problem and of a generalized Lagrange dual. In: Sebastian H-J, Tammer K (eds) System Modelling and Optimization. Lecture Notes Control Inform Sci. Springer, Berlin
35. Trujillo-Cortez R (1997) LFS functions in stable bilevel programming. PhD Thesis Dept Math and Statist, McGill Univ
36. Trujillo-Cortez R (2000) Stable bilevel programming and applications. PhD Thesis McGill Univ, in preparation
37. Tuy H (1998) Bilevel linear programming, multiobjective programming, and monotonic reverse convex programming. In: Migdalas A Pardalos PM, Värbrand P (eds) Multilevel Optimization: Algorithms and Applications. Kluwer, Dordrecht, 295–314
38. Ünlü G (1987) A linear bilevel programming algorithm based on bicriteria programming. Comput Oper Res 14:173–179
39. Vicente LN, Calamai PH (1994) Bilevel and multilevel programming: A bibliography review. J Global Optim 5:291–306
40. Vicente LN, Calamai PH (1995) Geometry and local optimality conditions for bilevel programs with quadratic strictly convex lower levels. In: Dhu D-Z, Pardalos PM (eds) Minimax and Applications. Kluwer, Dordrecht, pp 141–151
41. Vicente LN, Savard G, Judice J (1994) Descent approaches for quadratic bilevel programming. J Optim Th Appl 81: 379–399
42. Visweswaran V, Floudas CA, Ierapetritou MG, Pistikopoulos EN (1996) A decomposition-based global optimization approach for solving bilevel linear and quadratic programs. In: Floudas CA, Pardalos PM (eds) State of the Art in Global Optimization. Kluwer, Dordrecht, 139–162
43. Wen U-P, Hsu S-T (1989) A note on a linear bilevel programming algorithm based on bicriteria programming. Comput Oper Res 16:79–83
44. Wetterling W (1970) Definitheitsbedingungen für relative Extrema bei Optimierungs- und Approximationsaufgaben. Numerische Math 15:122–136
45. Ye J (1995) Necessary conditions for bilevel dynamic optimization problems. SIAM J Control Optim 33:1208–1223
46. Ye JJ, Zhu DL (1995) Optimality conditions for bilevel programming problems. Optim 33:9–27

47. Zlobec S (1996) Lagrange duality in partly convex programming. In: Floudas CA, Pardalos PM (eds) State of the Art in Global Optimization. Kluwer, Dordrecht, pp 1–18
48. Zlobec S (1998) Stable parametric programming. Optim 45:387–416
49. Zlobec S (2000) Parametric programming: An illustrative mini-encyclopedia. Math Commun 5:1–39

Bilinear Programming

ARTYOM G. NAHAPETYAN
Center for Applied Optimization,
Industrial and Systems Engineering Department,
University of Florida, Gainesville, USA

Article Outline

Keywords
Introduction
Formulation
  Equivalence to Other Problems
  Properties of a Solution
Methods
References

Keywords

Congestion toll pricing; Dynamic toll set; Toll pricing framework

Introduction

A function $f(\boldsymbol{x}, \boldsymbol{y})$ is called bilinear if it reduces to a linear one by fixing the vector $\boldsymbol{x}$ or $\boldsymbol{y}$ to a particular value. In general, a bilinear function can be represented as follows:

$$f(\boldsymbol{x}, \boldsymbol{y}) = a^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}Q\boldsymbol{y} + b^{\mathrm{T}}\boldsymbol{y}\,,$$

Where $a, \boldsymbol{x} \in \mathbb{R}^n$, b, $\boldsymbol{y} \in \mathbb{R}^m$, and Q is a matrix of dimension $n \times m$. It is easy to see that bilinear functions compose a subclass of quadratic functions. We refer to optimization problems with bilinear objective and/or constraints as bilinear problems, and they can be viewed as a subclass of quadratic programming.

Bilinear programming has various applications in constrained bimatrix games, Markovian assignment and complementarity problems. Many 0–1 integer programs can be formulated as bilinear problems. An extensive discussion of different applications can be found in [5]. Concave piecewise linear network flow problems, fixed charge network flow problems, and multi-item dynamic pricing problems, which are very common in the supply chain management, can be also solved using bilinear formulations (see, e. g., [7,8,9]). It should be noted that more general convex/non-convex optimization problems can be reduced to a bilinear problem as well, and different reduction techniques can be found in [1,2,10].

Formulation

Despite a variety of different bilinear problems, most of the practical problems involve a bilinear objective function and linear constraints, and theoretical results are derived for those cases. In our discussion we consider the following bilinear problem, which we refer to as BP.

$$\min_{\boldsymbol{x}\in X, \boldsymbol{y}\in Y} f(\boldsymbol{x}, \boldsymbol{y}) = a^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}Q\boldsymbol{y} + b^{\mathrm{T}}\boldsymbol{y}\,,$$

where X and Y are nonempty polyhedra. The BP formulation is also known as a bilinear problem with a *disjoint* feasible region because the feasibility of $\boldsymbol{x}$ ($\boldsymbol{y}$) is independent form the choice of the vector $\boldsymbol{y}$ (x).

Equivalence to Other Problems

Below we discuss some theoretical results, which reveal the equivalence between bilinear problems and some of concave minimization problems.

Let $V(\boldsymbol{x})$ and $V(\boldsymbol{y})$ denote the set of vertices of X and Y, respectively, and $g(\boldsymbol{x}) = \min_{\boldsymbol{y}\in Y} f(\boldsymbol{x}, \boldsymbol{y}) = a^{\mathrm{T}}\boldsymbol{x} + \min_{\boldsymbol{y}\in Y}\{\boldsymbol{x}^{\mathrm{T}}Q\boldsymbol{y} + b^{\mathrm{T}}\boldsymbol{y}\}$. Note that $\min_{\boldsymbol{y}\in Y} f(\boldsymbol{x}, \boldsymbol{y})$ is a linear programm. Because the solution of a linear problem attains on a vertex of the feasible region, $g(\boldsymbol{x}) = \min_{\boldsymbol{y}\in Y} f(\boldsymbol{x}, \boldsymbol{y}) = \min_{\boldsymbol{y}\in V(Y)} f(\boldsymbol{x}, \boldsymbol{y})$. Using those notations, the BP problem can be restated as

$$\begin{aligned}\min_{\boldsymbol{x}\in X, \boldsymbol{y}\in Y} f(\boldsymbol{x}, \boldsymbol{y}) &= \min_{\boldsymbol{x}\in X}\{\min_{\boldsymbol{y}\in Y} f(\boldsymbol{x}, \boldsymbol{y})\} \\ &= \min_{\boldsymbol{x}\in X}\{\min_{\boldsymbol{y}\in V(Y)} f(\boldsymbol{x}, \boldsymbol{y})\} = \min_{\boldsymbol{x}\in X} g(\boldsymbol{x})\,. \quad (1)\end{aligned}$$

Observe that the set of vertices of Y is finite, and for each $\boldsymbol{y} \in Y$, $f(\boldsymbol{x}, \boldsymbol{y})$ is a linear function of $\boldsymbol{x}$; therefore, function $g(\boldsymbol{x})$ is a piecewise linear concave function of x. From the later it follows that BP is equivalent to a piecewise linear concave minimization problem with linear constraints.

It also can be shown that any concave minimization problem with a piecewise linear separable objective function can be reduced to a bilinear problem. To establish this relationship consider the following optimization problem:

$$\min_{x\in X} \sum_i \phi_i(x_i)\,, \tag{2}$$

where X is an arbitrary nonempty set of feasible vectors, and $\phi_i(x_i)$ is a concave piecewise linear function of only one component x_i, i. e.,

$$\phi_i(x_i) = \begin{cases} c_i^1 x_i + s_i^1 (= \phi_i^1(x_i)) & x_i \in [\lambda_i^0, \lambda_i^1) \\ c_i^2 x_i + s_i^2 (= \phi_i^2(x_i)) & x_i \in [\lambda_i^1, \lambda_i^2) \\ \dots & \dots \\ c_i^{n_i} x_i + s_i^{n_i} (= \phi_i^{n_i}(x_i)) & x_i \in [\lambda_i^{n_i-1}, \lambda_i^{n_i}] \end{cases}$$

with $c_i^1 > c_i^2 > \ldots > c_i^{n_i}$. Let $K_i = \{1, 2, \ldots, n_i\}$. Because of the concavity of $\phi_i(x_i)$, the function can be written in the following alternative form

$$\phi_i(x_i) = \min_{k\in K_i}\{\phi_i^k(x_i)\} = \min_{k\in K_i}\{c_i^k x_i + s_i^k\}\,. \tag{3}$$

Construct the following bilinear problem:

$$\begin{aligned} \min_{x\in X, y\in Y} f(x,y) &= \sum_i \sum_{k\in K_i} \phi_i^k(x_i) y_i^k \\ &= \sum_i \sum_{k\in K_i} (c_i^k x_i + s_i^k) y_i^k \end{aligned} \tag{4}$$

where $Y = [0,1]^{\sum_i |K_i|}$. The proof of the following theorem follows directly from Equation (3), and for details we refer to the paper [7].

Theorem 1 *If (x^*, y^*) is a solution of the problem (4) then x^* is a solution of the problem (2).*

Observe that X is not required to be a polytop. If X is a polytop then the structure of the problem (4) is similar to BP.

Furthermore, it can be shown that any quadratic concave minimization problem can be reduced to a bilinear problem. Specifically, consider the following optimization problem:

$$\min_{x\in X} \phi(x) = 2a^{\mathrm{T}}x + x^{\mathrm{T}}Qx\,, \tag{5}$$

where Q is a symmetric negative semi-definite matrix. Construct the following bilinear problem

$$\min_{x\in X, y\in Y} f(x,y) = a^{\mathrm{T}}x + a^{\mathrm{T}}y + x^{\mathrm{T}}Qy\,, \tag{6}$$

where $Y = X$.

Theorem 2 *(see [4]) If x^* is a solution of the problem (5) then (x^*, x^*) is a solution of the problem (6). If $(\hat{x}, \hat{y})$ is a solution of the problem (6) then $\hat{x}$ and $\hat{y}$ solve the problem (5).*

Properties of a Solution

In the previous section we have shown that BP is equivalent to a piecewise linear concave minimization problem. On the other hand it is well known that a concave minimization problem over a polytop attains its solution on a vertex (see, for instance, [3]). The following theorem follows from this observation.

Theorem 3 *(see [4] and [3]) If X and Y are bounded then there is an optimal solution of BP, (x^*, y^*), such that $x^* \in V(X)$ and $y^* \in V(Y)$.*

Let (x^*, y^*) denote a solution of BP. By fixing the vector x to the value of the vector x^*, the BP problem reduces to a linear one, and y^* should be a solution of the resulting problem. From the symmetry of the problem, a similar result holds by fixing the vector y to the value of the vector y^*. The following theorem is a necessary optimality condition, and it is a direct consequence of the above discussion.

Theorem 4 *(see [4] and [3]) If (x^*, y^*) is a solution of the BP problem, then*

$$\min_{x\in X} f(x, y^*) = f(x^*, y^*) = \min_{y\in Y} f(x^*, y) \tag{7}$$

However, (7) is not a sufficient condition. In fact it can only guarantee a local optimality of (x^*, y^*) under some additional requirements. In particular, y^* has to be the unique solution of $\min_{y\in Y} f(x^*, y)$ problem. From the later it follows that $f(x^*, y^*) < f(x^*, y)$, $\forall y \in V(Y)$, $y \neq y^*$. Because of the continuity of the function $f(x, y)$, for any $y \in V(y)$, $y \neq y^*$, $f(x^*, y^*) < f(x, y)$ in a small neighborhood U_y of the point x^*. Let $U = \bigcap_{y\in V(Y), y\neq y^*} U_y$. Then $f(x^*, y^*) < f(x, y)$, $\forall x \in U$, $y \in V(Y)$, $y \neq y^*$. At last observe that Y is a polytop, and any point of the set

can be expressed through a convex combination of its vertices. From the later it follows that $f(\boldsymbol{x}^*, \boldsymbol{y}^*) \leq f(\boldsymbol{x}, \boldsymbol{y})$, $\forall \boldsymbol{x} \in U$, $\boldsymbol{y} \in Y$, which completes the proof of the following theorem.

Theorem 5 *If* $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ *satisfies the condition (7) and* $\boldsymbol{y}^*$ *is the unique solution of the problem* $\min_{y \in Y} f(\boldsymbol{x}^*, \boldsymbol{y})$ *then* $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ *is a local optimum of BP.*

Recall that BP is equivalent to a piecewise concave minimization problem. Under the assumptions of the theorem, it is easy to show that $\boldsymbol{x}^*$ is a local minimum of the function $g(\boldsymbol{x})$ as well (see [4]).

Methods

In this section we discuss methods to find a solution of a bilinear problem. Because BP is equivalent to a piecewise linear concave minimization problem, any solution algorithm for the later can be used to solve the former. In particular, one can employ a cutting plain algorithms developed for those problems. However, the symmetric structure of the BP problem allows constructing more efficient cuts. In the paper [6], the author discusses an algorithm, which converges to a solution that satisfies condition (7), and then proposes a cutting plain algorithm to find the global minimum of the problem.

Assume that X and Y are bounded. Algorithm 1, which is also known as the "mountain climbing" procedure, starts from an initial feasible vector $\boldsymbol{y}^0$ and iteratively solves two linear problems. The first LP is obtained by fixing the vector $\boldsymbol{y}$ to the value of the vector $\boldsymbol{y}^{m-1}$. The solution of the problem is used to fix the value of the vector $\boldsymbol{x}$ and construct the second LP. If $f(\boldsymbol{x}^m, \boldsymbol{y}^{m-1}) \neq f(\boldsymbol{x}^m, \boldsymbol{y}^m)$, then we continue solving the linear problems by fixing the vector $\boldsymbol{y}$ to the value of $\boldsymbol{y}^m$. If the stopping criteria is satisfied, then it is easy to show that the vector $(\boldsymbol{x}^m, \boldsymbol{y}^m)$ satisfies the condition (7). In addition, observe that $V(X)$ and $V(Y)$ are finite. From the later and the fact that $f(\boldsymbol{x}^m, \boldsymbol{y}^{m-1}) \geq f(\boldsymbol{x}^m, \boldsymbol{y}^m)$ it follows that the algorithm converges in a finite number of iterations.

Let $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ denote the solution obtained by the Algorithm 1. Assuming that the vertex $\boldsymbol{x}^*$ is not degenerate, denote by D the set of directions d_j along the ages emanating from the point $\boldsymbol{x}^*$. Recall that $g(\boldsymbol{x}) = \min_{y \in Y} f(\boldsymbol{x}, \boldsymbol{y})$ is a concave function. To construct a valid cut, for each direction d_j find the maximum value of θ_j such that $g(\boldsymbol{x}^* + \theta_j d_j) \geq f(\boldsymbol{x}^*, \boldsymbol{y}^*) - \varepsilon$, i. e.,

$$\theta_j = \text{argmax}\{\theta_j | g(\boldsymbol{x}^* + \theta_j d_j) \geq f(\boldsymbol{x}^*, \boldsymbol{y}^*) - \varepsilon\},$$

where ε is a small positive number. Let $C = (d_1, \ldots, d_n)$,

$$\Delta_{\boldsymbol{x}}^1 = \left\{ \boldsymbol{x} | \left(\frac{1}{\theta_1}, \ldots, \frac{1}{\theta_n} \right)^{\mathrm{T}} C^{-1} (\boldsymbol{x} - \boldsymbol{x}^*) \geq 1 \right\},$$

and $X_1 = X \bigcap \Delta_{\boldsymbol{x}}^1$. If $X_1 = \emptyset$ then

$$\min_{\boldsymbol{x} \in X, \boldsymbol{y} \in Y} f(\boldsymbol{x}, \boldsymbol{y}) \geq f(\boldsymbol{x}^*, \boldsymbol{y}^*) - \varepsilon,$$

and $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is a global ε-optimum of the problem. If $X_1 \neq \emptyset$ then one can replace X by the set X_1, i. e., consider the optimization problem

$$\min_{\boldsymbol{x} \in X_1, \boldsymbol{y} \in Y} f(\boldsymbol{x}, \boldsymbol{y}),$$

and run Algorithm 1 to find a better solution. However, because of the symmetric structure of the problem, a similar procedure can be applied to construct

Step 1: Let $y^0 \in Y$ denote an initial feasible solution, and $m \leftarrow 1$.

Step 2: Let $x^m = argmin_{x \in X}\{f(x, y^{m-1})\}$, and $y^m = argmin_{y \in Y}\{f(x^m, y)\}$.

Step 3: If $f(x^m, y^{m-1}) = f(x^m, y^m)$ then stop. Otherwise, $m \leftarrow m + 1$ and go to Step 2.

Bilinear Programming, Algorithm 1
Mountain Climbing Procedure

Step 1: Apply Algorithm 1 to find a vector (x^*, y^*) that satisfies the relationship (7).

Step 2: Based on the solution (x^*, y^*), compute the appropriate cuts and construct the sets X_1 and Y_1.

Step 3: If $X_1 = \emptyset$ or $Y_1 = \emptyset$, then stop; (x^*, y^*) is a global ε-optimal solution. Otherwise, $X \leftarrow X_1$, $Y \leftarrow Y_1$, and go to Step 1.

Bilinear Programming, Algorithm 2
Cutting Plane Algorithm

a cut for the set Y. Let Δ_y^1 denote the corresponding half-space, and $Y_1 = Y \bigcap \Delta_y^1$. By updating both sets, i. e., considering the optimization problem

$$\min_{x \in X_1, y \in Y_1} f(x, y) ,$$

the cutting plane algorithm (see Algorithm 2) might find a global solution of the problem using less number of iterations.

References

1. Floudas CA, Visweswaran V (1990) A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: I. Theory. Comput Chem Eng 14:1397–1417
2. Floudas CA, Visweswaran V (1993) A primal-relaxed dual global optimization approach. J Optim Theor Appl 78: 187–225
3. Horst R, Pardalos P, Thoai N (2000) Introduction to global optimization, 2nd edn. Springer, Boston
4. Horst R, Tuy H (1996) Global optimization, 3rd edn. Springer, New York
5. Konno H (1971) A bilinear programming: Part II. Applications of bilinear programming. Technical Report 71–10, Operations Research House, Stanford University, Stanford, CA
6. Konno H (1976) A cutting plane algorithm for solving bilinear programs. Math Program 11:14–27
7. Nahapetyan A, Pardalos P (2007) A bilinear relaxation based algorithm for concave piecewise linear network flow problems. J Ind Manag Optim 3:71–85
8. Nahapetyan A, Pardalos P (2008) Adaptive dynamic cost updating procedure for solving fixed charge network flow problems. Comput Optim Appl 39:37–50. doi:10.1007/s10589-007-9060-x
9. Nahapetyan A, Pardalos P (2008) A Bilinear Reduction Based Algorithm for Solving Capacitated Multi-Item Dynamic Pricing Problems. J Comput Oper Res 35:1601–1612. doi:10.1016/j.cor.2006.09.003
10. Visweswaran V, Floudas CA (1990) Global Optimization Algorithm (GOP) for Certain Classes of Nonconvex NLPs: II. Applications of Theory and Test Problems. Comput Chem Eng 14:1417–1434

Bilinear Programming: Applications in the Supply Chain Management

ARTYOM G. NAHAPETYAN
Center for Applied Optimization,
Industrial and Systems Engineering Department,
University of Florida, Gainesville, USA

Article Outline

Introduction
Formulation
  Concave Piecewise Linear Network Flow Problem
  Fixed Charge Network Flow Problem
  Capacitated Multi-Item Dynamic Pricing Problem
Methods
References

Introduction

Many problems in the supply chain management can be formulated as a network flow problem with specified arc cost functions. Let $G(N, A)$ represent a network where N and A are the sets of nodes and arcs, respectively, and $f_a(x_a)$ denotes an arc cost function. In the network, there are supply and demand nodes, and the main objective of the problem is to minimize the total cost by satisfying the demand from the available supply. In addition, one can assume that the arc flows are bounded, which corresponds to the cases where a shipment along an arc should not exceed a specified capacity. The mathematical formulation of the problem can be stated as

$$\min_x f(x) = \sum_{a \in A} f_a(x_a) \tag{1}$$

$$\text{s.t.} \quad Bx = b \tag{2}$$

$$x_a \in [0, \lambda_a] \quad \forall a \in A \tag{3}$$

where B is the node-arc incident matrix of network G, and b is a suplly/demand vector. In the next section, we discuss two formulations where $f_a(x_a)$ is either a concave piecewise linear or fixed charge function of the arc flow. The concave piecewise linear functions are typically used in the cases where merchandisers encourage to buy more products by offering discounts in the unit price for large orders. In [6], the authors showed that the problem in these settings is NP hard. Some heuristic procedures to solve the problem are discussed in [10] and [13]. The fixed charge functions are used in the cases where regardless the quantity of the shipment it is required to pay a fixed cost to ship along an arc. The fixed cost might be the cost of renting a truck, ship, airplane, or train to transport goods between nodes of the network. The problem can be modeled as a 0–1 mixed integer linear program and most solution approaches

utilize branch-and-bound techniques to find an exact solution (see [1,2,5,7,16]). Some heuristic procedures are discussed in [3,4,8,9,12,14]. In this article we show that both problems are equivalent to a bilinear problem with a disjoint feasible region.

In addition to choosing a proper production level, sometimes managers have to make pricing decisions as well. In particular, one can assume that the satisfied demand is a function of the price, i. e., lower prices generate an additional demand. Such functional relationship between the prices and the satisfied demand is commonly used by economists. However, because of the production capacity restrictions, fixed costs related to the production process, seasonality and other factors, often it is not feasible to satisfy the optimal level of demand, and managers should consider optimal production and inventory levels in combination with pricing decisions to maximize the net profit during a specified time period. One of such problems and an equivalent bilinear formulation are discussed in the next section as well.

In addition to the bilinear formulations of the supply chain problems, in Sect. "Methods" we explore the structure of the bilinear problems and discuss difficulties in applying the standard computational methods. Despite the intricacy, the section proposes some heuristic methods to find a near optimum solution to the problems. The solution obtained by a heuristic procedure can also be used to expedite exact algorithms.

Formulation

Concave Piecewise Linear Network Flow Problem

In the problem (1)–(3), assume that $f_a(x_a)$ is a piecewise linear concave function, i. e.,

$$f_a(x_a) = \begin{cases} c_a^1 x_a + s_a^1 (= f_a^1(x_a)) & x_a \in [0, \xi_a^1) \\ c_a^2 x_a + s_a^2 (= f_a^2(x_a)) & x_a \in [\xi_a^1, \xi_a^2) \\ \cdots & \cdots \\ c_a^{n_a} x_a + s_a^{n_a} (= f_a^{n_a}(x_a)) & x_a \in [\xi_a^{n_a-1}, \lambda_a] \end{cases},$$

with $c_a^1 > c_a^2 > \ldots > c_a^{n_a}$. Let $K_a = \{1, 2, \ldots, n_a\}$. Because of the concavity of $f_a(x_a)$, it can be written in the following alternative form

$$f_a(x_a) = \min_{k \in K_a} \{f_a^k(x_a)\} = \min_{k \in K_a} \{c_a^k x_a + s_a^k\}. \tag{4}$$

By introducing additional variables $y_a^k \in [0, 1]$, $k \in K_a$, construct the following bilinear problem.

$$\min_{x,y} g(x, y) = \sum_{a \in A} \left[\sum_{k \in K_a} c_a^k y_a^k \right] x_a + \sum_{a \in A} \sum_{k \in K_a} s_a^k y_a^k = \sum_{a \in A} \sum_{k \in K_a} f_a^k(x_a) y_a^k \tag{5}$$

$$\text{s.t.} \quad Bx = b \tag{6}$$

$$\sum_{k \in K_a} y_a^k = 1 \quad \forall a \in A \tag{7}$$

$$x_a \in [0, \lambda_a], y_a^k \geq 0 \quad \forall a \in A \quad \text{and} \quad k \in K_a \tag{8}$$

In [13], the authors show that at any local minima of the bilinear problem, $(\hat{x}, \hat{y})$, $\hat{y}$ is either binary vector or can be used to construct a binary vector with the same objective function value. Although the vector $\hat{y}$ may have a fractional components, the authors note that in practical problems it is highly unlikely. The proof of the theorem below follows directly from (4). Details on the proof as well as transformation of the problem (1)–(3) into (5)–(8) can be found in [13].

Theorem 1 *If (x^*, y^*) is a global optima of the problem (5)–(8) then x^* is a solution of the problem (1)–(3).*

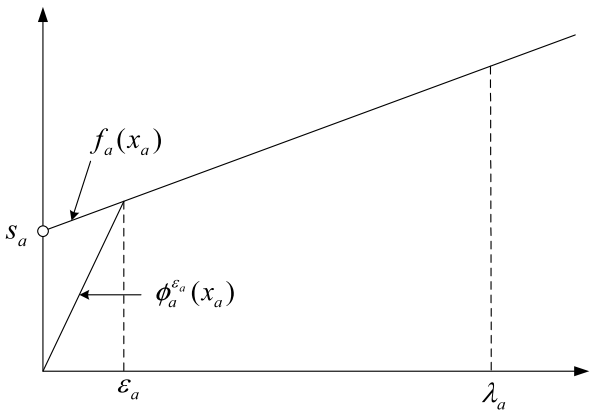
According to the theorem, the concave piecewise linear network flow problem is equivalent to a bilinear problem in a sense that the solution of the later is a solution of the former. It is important to notice that the problem (5)–(8) does not have binary variables, i. e., all variables are continuous. However, at optimum y^* is a binary vector, which makes sure that in the objective only one linear piece is employed.

Fixed Charge Network Flow Problem

In the case of the fixed charge network flow problem, we assume that the function $f_a(x_a)$ has the following structure.

$$f_a(x_a) = \begin{cases} c_a x_a + s_a & x_a \in (0, \lambda_a] \\ 0 & x_a = 0 \end{cases},$$

Observe that the function is discontinuous at the origin and linear on the interval $(0, \lambda_a]$.



Bilinear Programming: Applications in the Supply Chain Management, Figure 1
Approximation of function $f_a(x_a)$

Let $\varepsilon_a \in (0, \lambda_a]$, and define

$$\phi_a^{\varepsilon_a}(x_a) = \begin{cases} c_a x_a + s_a & x_a \in [\varepsilon_a, \lambda_a] \\ c_a^{\varepsilon_a} x_a & x_a \in [0, \varepsilon_a) \end{cases}$$

where $c_a^{\varepsilon_a} = c_a + s_a/\varepsilon_a$. It is easy to see that $\phi_a^{\varepsilon_a}(x_a) = f_a(x_a)$, $\forall x_a \in \{0\} \bigcup [\varepsilon_a, \lambda_a]$ and $\phi_a^{\varepsilon_a}(x_a) < f_a(x_a)$, $\forall x_a \in (0, \varepsilon_a)$, i. e., $\phi_a^{\varepsilon_a}(x_a)$ approximates the function $f_a(x_a)$ from below. (see Fig. 1). Let us construct the following concave two-piece linear network flow problem.

$$\min_x \phi^{\varepsilon}(x) = \sum_{a \in A} \phi_a^{\varepsilon_a}(x_a) \tag{9}$$

$$\text{s.t.} \quad Bx = b, \tag{10}$$

$$x_a \in [0, \lambda_a], \quad \forall a \in A\,, \tag{11}$$

where ε denotes the vector of ε_a. Function $\phi^{\varepsilon}(x)$ as well as the problem (9)–(11) depends on the value of the vector ε. In the paper [14], the authors show that for any value of $\varepsilon_a \in (0, \lambda_a]$, a global solution of the problem (9)–(11) provides a lower bound for the fixed charge network flow problem, i. e., $\phi^{\varepsilon}(x^{\varepsilon}) \le f(x^*)$, where x^{ε} and x^* denote the solutions of the corresponding problems.

Theorem 2 (see [14]) *For all ε such that $\varepsilon_a \in (0, \lambda_a]$ for all $a \in A$, $\phi^{\varepsilon}(x^{\varepsilon}) \le f(x^*)$.*

Furthermore, by choosing a sufficiently small value for ε_a one can ensure that both problems have the same solution. Let $\delta = \min\{x_a^v | x^v \in V(x), a \in A, x_a^v > 0\}$, where $V(x)$ denotes the set of vertices of the polyhedra (10)–(11). Observe that δ is the minimum among all positive components of all vectors $x^v \in V(x)$; therefore, $\delta > 0$.

Theorem 3 (see [14]) *For all ε such that $\varepsilon_a \in (0, \delta]$ for all $a \in A$, $\phi^{\varepsilon}(x^{\varepsilon}) = f(x^*)$.*

Theorem 3 proves the equivalence between the fixed charge network flow problem and the concave two-piece linear network flow problem (9)–(11) in a sense that the solution of the later is a solution of the former. As we have seen in the previous section, concave piecewise linear network flow problems are equivalent to bilinear problems. In particular, problem (9)–(11) is equivalent to the following bilinear problem.

$$\min_{x,y} \sum_{a \in A} [c_a x_a + s_a]\, y_a + c_a^{\varepsilon_a} x_a \left[1 - y_a\right] \tag{12}$$

$$\text{s.t.} \quad Bx = b\,, \tag{13}$$

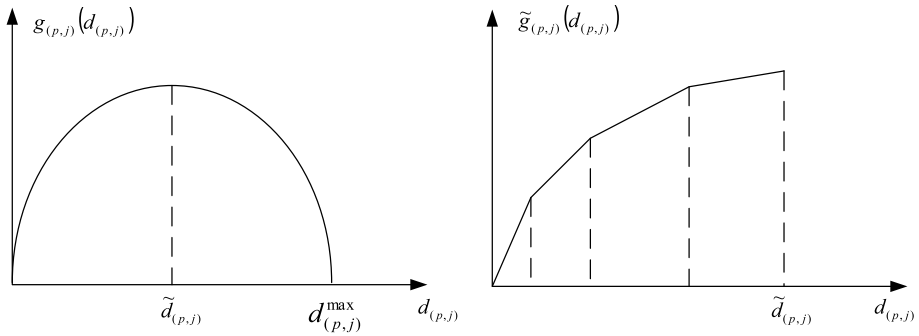
$$x_a \ge 0\,, \quad \text{and} \quad y_a \in [0, 1]\,, \quad \forall a \in A\,, \tag{14}$$

where $\varepsilon_a \in (0, \delta]$.

Capacitated Multi-Item Dynamic Pricing Problem

In the problem, we assume that a company during a discrete time period Δ is able to produce different commodities from a set P. In addition, we assume that at each point of time $j \in \Delta$ and for each product $p \in P$ a functional relationship $f_{(p,j)}(d_{(p,j)})$ between the satisfied demand and the price is given, i. e., in order to satisfy the demand $d_{(p,j)}$ of the product p, the price of the product at time j should be equal to $f_{(p,j)}(d_{(p,j)})$. As a result, the revenue generated from the sales of the product p at time j is $g_{(p,j)}(d_{(p,j)}) = f_{(p,j)}(d_{(p,j)})d_{(p,j)}$. Although we do not specify the function $f_{(p,j)}(d_{(p,j)})$, it should ensure that $g_{(p,j)}(d_{(p,j)})$ is a concave function (see Fig. 2a).

Because of the concavity of $g_{(p,j)}(d_{(p,j)})$, there exists a point $\tilde{d}_{(p,j)}$, such that the function reaches its maximum, and producing and selling more than $\tilde{d}_{(p,j)}$ is not profitable. Therefore, without lost of generality, we can assume that $d_{(p,j)} \in [0, \tilde{d}_{(p,j)}]$. According to the definition of $g_{(p,j)}(d_{(p,j)})$, it is a concave monotone function on the interval $[0, \tilde{d}_{(p,j)}]$. To avoid nonlinearity in the objective, one can approximate it by a concave piecewise linear function. Doing so, divide



a The Revenue Function b Approximation of the Revenue Function

Bilinear Programming: Applications in the Supply Chain Management, Figure 2
The revenue function and its approximation

$[0, \tilde{d}_{(p,j)}]$ into intervals of equal length, and let $d^k_{(p,j)}$, $k \in \{1, \ldots, N\} \bigcup \{0\} = K \bigcup \{0\}$, denote the end points of the intervals. Then the approximation can be defined as

$$\tilde{g}_{(p,j)}(\lambda_{(p,j)}) = \sum_{k=1}^{N} g^k_{(p,j)} \lambda^k_{(p,j)} ,$$

where $g^k_{(p,j)} = g_{(p,j)}(d^k_{(p,j)}) = f_{(p,j)}(d^k_{(p,j)}) d^k_{(p,j)}$, $\sum_{k=0}^{N} \lambda^k_{(p,j)} = 1$, and $\lambda^k_{(p,j)} \geq 0, \forall p \in P, j \in \Delta$ (see Fig. 2b).

Let $x^k_{(p,i,j)}$ denote the amount of product p that is produced at time i and sold at time j using the unit price $g^k_{(p,j)}/d^k_{(p,j)} = f^k_{(p,j)} = f_{(p,j)}(d^k_{(p,j)})$. In addition, let $y_{(p,i)}$ denote a binary variable, which equals one if $\sum_k \sum_j x^k_{(p,i,j)} > 0$ and zero otherwise. Costs associated with the production process include inventory costs $c^{in}_{(p,i,j)}$, production costs $c^{pr}_{(p,i)}$, and setup costs $c^{st}_{(p,i)}$. At last, let C_i represent the production capacity at time i, which is "shared" by all products. Using those definitions, one can construct a linear mixed integer formulation of the problem. Below we provide a simplified formulation of the problem, where the variables $\lambda_{(p,j)}$ are eliminated from the formulation. For the details on the mathematical formulation of the problem and its simplification we refer to [15].

$$\max_{x,y} \sum_{p \in P} \sum_{i \in \Delta} \left[\sum_{j \in \Delta | i \leq j} \sum_{k \in K} q^k_{(p,i,j)} x^k_{(p,i,j)} - c^{st}_{(p,i)} y_{(p,i)} \right] \tag{15}$$

$$\sum_{p \in P} \sum_{j \in \Delta | i \leq j} \sum_{k \in K} x^k_{(p,i,j)} \leq C_i, \forall i \in \Delta , \tag{16}$$

$$\sum_{j \in \Delta | i \leq j} \sum_{k \in K} x^k_{(p,i,j)} \leq C_i y_{(p,i)}, \quad \forall p \in P \quad \text{and} \quad i \in \Delta , \tag{17}$$

$$\sum_{k \in K} \sum_{i \in \Delta | i \leq j} \frac{x^k_{(p,i,j)}}{d^k_{(p,j)}} \leq 1 , \quad \forall p \in P \quad \text{and} \quad j \in \Delta , \tag{18}$$

$$x^k_{(p,i,j)} \geq 0 , \quad y_{(p,i)} \in \{0, 1\} , \quad \forall p \in P, i, j \in \Delta \quad \text{and} \quad k \in K , \tag{19}$$

where $q^k_{(p,i,j)} = f^k_{(p,j)} - c^{in}_{(p,i,j)} - c^{pr}_{(p,i)}$.

Let $X = \{x | x \geq 0$ and $x^k_{(p,i,j)}$ be feasible to (16) and (18)$\}$, and $Y = [0, 1]^{|P||\Delta|}$. Consider the following bilinear problem.

$$\max_{x \in X, y \in Y} \varphi(x, y) = \sum_{p \in P} \sum_{i \in \Delta} \left[\sum_{j \in \Delta | i \leq j} \sum_{k \in K} q^k_{(p,i,j)} x^k_{(p,i,j)} - c^{st}_{(p,i)} \right] y_{(p,i)} \tag{20}$$

Theorem 4 (see [15]) *A global maximum of the bilinear problem (20) is a solution or can be transformed into a solution of the problem (15)–(19).*

Methods

In the previous section, we have discussed several problems arising in the supply chain management. To solve the bilinear formulations of the problems, one can em-

ploy techniques applicable for general bilinear problems. In particular, a cutting plain algorithm proposed by Konno can be applied to find a global solution of the problems. In addition, he proposes an iterative procedure, which converges to a local minimum of the problem in a finite number of iterations. For details on the procedure, which is also known as "mountain climbing" procedure (MCP), and the cutting plain algorithm we refer to the paper [11] or Bilinear Programming section of this encyclopedia.

Below, we discuss problem specific difficulties of applying the above mentioned algorithms and some effective heuristic procedures, which are able to provide a near optimum solution using negligible computer resources. The MCP, which is used by the heuristics to find a local minimum/maximum of the problems, is very fast due to a special structure of both LP problems employed by the procedure. However, to obtain a high quality solution, in some problems it is necessary to solve a sequence of approximate problems. The bilinear formulations of the supply chain problems typically have many local minima. Therefore, cutting plain algorithms may require many cuts to converge. By combining the heuristic procedures with the cutting plain algorithm, one can reduce the number of cuts by generating deep cuts.

One of the main properties of a bilinear problem with a disjoint feasible region is that by fixing vectors x or y to a particular value, the problem reduces to a linear one. The "mountain climbing" procedure employs this property and iteratively solves two linear problems by fixing the corresponding vectors to the solution of the corresponding linear programs. In the case of concave piecewise linear network flow problem, given the vector $\hat{x}$, the problem (5)–(8) can be decomposed into $|A|$ problems,

$$\min_{\{y_a^k | k \in K_a\}} \sum_{k \in K_a} [c_a^k \hat{x}_a + s_a^k] y_a^k$$

$$\text{s.t.} \quad \sum_{k \in K_a} y_a^k = 1\,, \quad y_a^k \geq 0 \quad \forall k \in K_a\,.$$

Furthermore, it can be shown that a solution of the problem is a binary vector, which has to satisfy the inequality

$$\sum_{k \in K_a} \xi_a^{k-1} y_a^k \leq \hat{x}_a \leq \sum_{k \in K_a} \xi_a^k y_a^k\,.$$

As a result, one can employ a search technique by assigning $y_a^{\hat{k}} = 1$ if $\xi_a^{\hat{k}-1} \leq \hat{x}_a \leq \xi_a^{\hat{k}}$ and $y_a^k = 0$, $\forall k \in K_a, k \neq \hat{k}$. On the other hand, by fixing the vector y to the value of the constructed vector $\hat{y}$, the problem (5)–(8) reduces to the following network flow problem.

$$\min_x \sum_{a \in A} \left[\sum_{k \in K_a} c_a^k \hat{y}_a^k \right] x_a$$

$$\text{s.t.} \quad Bx = b, \quad x_a \geq 0, \quad \forall a \in A$$

Observe that $\sum_{k \in K_a} c_a^k \hat{y}_a^k = c_a^{\hat{k}}$, and different vectors $\hat{y}$ change the cost vector in the problem.

Although the MCP converges to a local minimum, it can provide a near optimum solution for the problem (5)–(8) if the initial vector $\hat{y}$ is such that $\hat{y}_a^{n_a} = 1$ and $\hat{y}_a^k = 0$, $\forall k \in K_a$, $k \neq n_a$. The effectiveness of the procedure is partially due to the fact that in the supply chain problems $f_a(x_a)$ is an increasing function. In addition, the procedure requires less computer resources to converge because both linear problems are relatively easy to solve. A detailed description of the procedure, properties of the linear problems, and computational experiments can be found in [13].

In the case of fixed charge network flow problems, it is not obvious how to choose the vector ε. Theorem 3 guarantees the equivalence between the fixed charge network flow problem and the bilinear problem (12)–(14) if $\varepsilon_a \in (0, \delta]$. However, according to the definition, it is necessary to find all vertices of the feasible region to compute the value of δ, which is computationally expensive. Even if the correct value of δ is known, typically it is a very small number. As a result, the value of ε_a is close to zero, and $c_a^{\varepsilon_a}$ is very large compared to the value of c_a. The later creates some difficulties for finding a global solution of the bilinear problem. In particular, the MCP may converge to a local minimum, which is far from being a global solution.

To overcome those difficulties, [14] proposes a procedure where it gradually decreases the value of ε (see Algorithm 1). The algorithm starts from an initial value for the vector ε, i. e., $\varepsilon_a = \lambda_a$. After constructing the corresponding bilinear problem, it employs the MCP to find a local minimum of the problem. If the stopping criteria is not satisfied, the value of ε is updated, i. e., $\varepsilon_a = \alpha \varepsilon_a$ where $\alpha \in (0, 1)$, and the algorithm again

solves the updated bilinear problem using the current solution as an initial vector for the MCP.

The choice of α has a direct influence on the CPU time of the algorithm and the quality of the solution. Specifically, if the value of α is closer to one, then due to the fact that ε decreases slowly, the algorithm requires many iterations to stop. On the other hand, if the values of the parameter is closer to zero, it may worsen the quality of the solution. A proper choice of the parameter depends on the problem, and it should be chosen by trials and errors. In the paper [14], the authors test the algorithm on various randomly generated test problems and found satisfactory to choose $\alpha = 0.5$.

As for the stopping criteria, it is possible to show that the solution of the final bilinear problem is the solution of the fixed charge network flow problem if on Step 2 one is able to find a global solution of the corresponding bilinear problems. For details on the numerical experiments, stopping criteria and other properties of the algorithm, we refer to [14].

In the problems with pricing decisions, one may also experience some difficulties to employ the MCP for finding a near optimum solution. To explore the properties of the problem, consider the following two linear problems, which are constructed from the problem (20) by fixing either vector x or y to the value of the vector $\hat{x}$ or $\hat{y}$, respectively.

LP_1 :

$$\max_{y\in Y}\sum_{p\in P}\sum_{i\in\Delta}\left[\sum_{j\in\Delta|i\le j}\sum_{k\in K}q^k_{(p,i,j)}\hat{x}^k_{(p,i,j)}-c^{st}_{(p,i)}\right]y_{(p,i)}$$

LP_2 :

$$\max_{x\in X}\sum_{p\in P}\sum_{i\in\Delta}\sum_{j\in\Delta|i\le j}\sum_{k\in K}\left[q^k_{(p,i,j)}\hat{y}_{(p,i)}\right]x^k_{(p,i,j)}\,.$$

The MCP solves iteratively LP_1 and LP_2 problems, where the solution of the first problem is used to fix the corresponding vector in the second problem. However, if one of the components of the vector y equals to zero during one of the iterations, e. g., $\hat{y}_{(p,i)} = 0$, then in the second problem coefficients of the corresponding variables $x^k_{(p,i,j)}$ are equal to zero as well. As a result, changes in the values of those variables do not have any influence on the objective function value. Furthermore, because the products "share" the capacity and other products may have positive coefficients in the objective, it is likely that at optimum of LP_2, $\hat{x}^k_{(p,i,j)} = 0, \forall j \in \Delta$, $k \in K$. From the later, it follows that $\hat{y}_{(p,i)} = 0$ during the next iteration, and one concludes that if some products are eliminated from the problem during the iterative process, the MCP does not consider them again. Therefore, it is likely that the solution returned by the algorithm is far from being a global one. To avoid zero coefficients in the objective of LP_2, [15] proposes an approximation to the problem (20), which can be used in the MCP to find a near optimum solution.

Step 1: Let $\varepsilon_a \leftarrow \lambda_a$, $x^0_a \leftarrow 0$, $y^0_a \leftarrow 0$, and $m \leftarrow 1$.

Step 2: Find a local minimum of the problem (12)-(14) using the MCP. Let (x^m, y^m) denote the solution found by the algorithm.

Step 3: If $\exists a \in A$ such that $x^m_a \in (0, \varepsilon^m_a)$ then $\varepsilon_a \leftarrow \alpha\varepsilon_a$, $m \leftarrow m + 1$, and go to step 2. Otherwise, stop.

Bilinear Programming: Applications in the Supply Chain Management, Algorithm 1

To construct the approximate problem, let

$$\varphi^1_{(p,i)}(x_{(p,i)}) = \sum_{j\in\Delta|i\le j}\sum_{k\in K}q^k_{(p,i,j)}x^k_{(p,i,j)} - c^{st}_{(p,i)}\,,$$

and

$$\varphi^2_{(p,i)}(x_{(p,i)}) = \frac{\varepsilon_{(p,i)}}{\varepsilon_{(p,i)} + c^{st}_{(p,i)}}\sum_{j\in\Delta|i\le j}\sum_{k\in K}q^k_{(p,i,j)}x^k_{(p,i,j)}\,,$$

Step 1: Let $\varepsilon_{(p,i)}$ be a sufficiently large number, $y^0_{(p,i)} = 1, \forall p \in P$, $i \in \Delta$, and $m \leftarrow 0$.

Step 2: Construct the approximation problem (21), and find a local maximum of the problem using the MSP. Let (x^{m+1}, y^{m+1}) denote the solution returned by the algorithm.

Step 3: If $\exists p \in P$ and $i \in \Delta$ such that $\sum_{j\in\Delta|i\le j}\sum_{k\in K}q^k_{(p,i,j)}x^{(m+1)k}_{(p,i,j)} - c^{st}_{(p,i)} \le \varepsilon^m_{(p,i)}$ and $\sum_{j\in\Delta|i\le j}\sum_{k\in K}x^{(m+1)k}_{(p,i,j)} > 0$ then $\varepsilon \leftarrow \alpha\varepsilon$, $m \leftarrow m + 1$ and go to Step 2. Otherwise, stop.

Bilinear Programming: Applications in the Supply Chain Management, Algorithm 2

where $\varepsilon_{(p,i)} > 0$, and $x_{(p,i)}$ is the vector of $x^k_{(p,i,j)}$. Using those functions, construct the following bilinear problem

$$\max_{x\in X, y\in Y} \varphi^{\varepsilon}(x,y) = \sum_{p\in P}\sum_{i\in \Delta}\left[\varphi^1_{(p,i)}(x_{(p,i)})y_{(p,i)} + \varphi^2_{(p,i)}(x_{(p,i)})(1-y_{(p,i)})\right], \tag{21}$$

where the feasible region is the same as in the problem (20). The authors show that $\varphi^{\varepsilon}(x,y)$ approximates the function $\varphi(x,y)$ from above.

Theorem 5 (see [15]) *There exists a sufficiently small $\varepsilon > 0$ such that a solution of the problem (20) is a solution of the problem (21).*

Algorithm 2 starts from a sufficiently large value of $\varepsilon_{(p,i)}$ and finds a local maximum of the corresponding bilinear problem (21) using the MCP. If the stopping criteria is not satisfied then it updates the value of ε to $\alpha\varepsilon$, updates the bilinear problem (21), and employs the MCP to find a better solution. Similar to the fixed charge network flow problem, the choice of α has a direct influence on the CPU time of the algorithm and the quality of the returned solution. The running time of the algorithm and the quality of the solution for the different values of α are studied in [15].

In addition to α, one has to find a proper initial value for the parameter $\varepsilon_{(p,i)}$. Ideally, it should be equal to the maximum profit that can be generated by producing only product p at time i. However, it requires solving a linear problem for each pair $(p,i) \in P \times \Delta$, which is computationally expensive. On the other hand, it is not necessary to find an exact solution of those LPs, and one might consider a heuristic procedure which provides a quality solution within a reasonable time. One of such procedures is discussed in [15].

References

1. Barr R, Glover F, Klingman D (1981) A New Optimization Method for Large Scale Fixed Charge Transportation Problems. Oper Res 29:448–463
2. Cabot A, Erenguc S (1984) Some Branch-and-Bound Procedures for Fixed-Cost Transportation Problems. Nav Res Logist Q 31:145–154
3. Cooper L, Drebes C (1967) An Approximate Solution Method for the Fixed Charge Problem. Nav Res Logist Q 14:101–113
4. Diaby M (1991) Successive Linear Approximation Procedure for Generalized Fixed-Charge Transportation Problem. J Oper Res Soc 42:991–1001
5. Gray P (1971) Exact Solution for the Fixed-Charge Transportation Problem. Oper Res 19:1529–1538
6. Guisewite G, Pardalos P (1990) Minimum concave-cost network flow problems: applications, complexity, and algorithms. Ann Oper Res 25:75–100
7. Kennington J, Unger V (1976) A New Branch-and-Bound Algorithm for the Fixed Charge Transportation Problem. Manag Sci 22:1116–1126
8. Khang D, Fujiwara O (1991) Approximate Solution of Capacitated Fixed-Charge Minimum Cost Network Flow Problems. Netw 21:689–704
9. Kim D, Pardalos P (1999) A Solution Approach to the Fixed Charge Network Flow Problem Using a Dynamic Slope Scaling Procedure. Oper Res Lett 24:195–203
10. Kim D, Pardalos P (2000) Dynamic Slope Scaling and Trust Interval Techniques for Solving Concave Piecewise Linear Network Flow Problems. Netw 35:216–222
11. Konno H (1976) A Cutting Plane Algorithm for Solving Bilinear Programs. Math Program 11:14–27
12. Kuhn H, Baumol W (1962) An Approximate Algorithm for the Fixed Charge Transportation Problem. Nav Res Logist Q 9:1–15
13. Nahapetyan A, Pardalos P (2007) A Bilinear Relaxation Based Algorithm for Concave Piecewise Linear Network Flow Problems. J Ind Manag Optim 3:71–85
14. Nahapetyan A, Pardalos P (2008) Adaptive Dynamic Cost Updating Procedure for Solving Fixed Charge Network Flow Problems. Comput Optim Appl 39:37–50. doi:10.1007/s10589-007-9060-x
15. Nahapetyan A, Pardalos P (2008) A Bilinear Reduction Based Algorithm for Solving Capacitated Multi-Item Dynamic Pricing Problems. Comput Oper Res J 35:1601–1612. doi:10.1016/j.cor.2006.09.003
16. Palekar U, Karwan M, Zionts S (1990) A Branch-and-Bound Method for Fixed Charge Transportation Problem. Manag Sci 36:1092–1105

Bi-Objective Assignment Problem

JACQUES TEGHEM
Lab. Math. & Operational Research Fac.,
Polytechn. Mons, Mons, Belgium

MSC2000: 90C35, 90C10

Article Outline

Keywords
Direct Methods

Two-Phase Methods
First Step
Second Step
Heuristic Methods
Preliminaries
Determination of $PE(\lambda^{(l)})$, $l = 1, \ldots, L$
Generation of $\widehat{E(\mathrm{P})}$
Concluding Remarks
See also
References

Keywords

Multi-objective programming; Combinatorial optimization; Assignment

Until recently (1998), *multi-objective combinatorial optimization* (MOCO) did not receive much attention in spite of its potential applications. The reason is probably due to specific difficulties of MOCO models as pointed out in ► Multi-objective combinatorial optimization. Here we consider a particular bi-objective MOCO problem, the *assignment problem* (AP). This is a basic well-known combinatorial optimization problem, important for applications and as a subproblem of more complicated ones, like the transportation problem, distribution problem or traveling salesman problem. Moreover, its mathematical structure is very simple and there exist efficient polynomial algorithms to solve it in the single objective case, like the *Hungarian method*. In a bi-objective framework, the assignment problem can be formulated as:

$$(\mathrm{P}) \quad \begin{cases} \text{'min'} & z_k(X) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{(k)} x_{ij}, \\ & k = 1, 2, \\ & \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n, \\ & \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \\ & x_{ij} \in \{0, 1\} \end{cases}$$

where c_{ij}^{k} are nonnegative integers and $X = (x_{11}, \ldots, x_{nn})$. Our aim is to generate the set of efficient solutions $E(\mathrm{P})$. It is important to stress that the distinction between the *supported efficient solutions* (belonging to SE (P)), i. e. those which are optimal solutions of the single objective problem obtained by a linear aggregation of the objectives, and the *nonsupported efficient solutions* (belonging to $NSE(\mathrm{P}) = E(P)\backslash SE(P)$) (see ► Multi-objective integer linear programming) is still necessary even if the constraints of the problem satisfy the so-called 'totally unimodular' or 'integrality' property: when this property is verified, the integrality constraints of the single objective problem can be relaxed without any deterioration of the objective function, i. e. the optimal values of the variables are integer even if only the linear relaxation of the problem is solved. It is well known that the single objective assignment problem satisfies this integrality property, and thus this is true for the problem (see ► Multi-objective combinatorial optimization):

$$(\mathrm{P}_\lambda) \quad \begin{cases} \min & z_\lambda(X) = \lambda_1 z_1(X) + \lambda_2 z_2(X) \\ & \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n, \\ & \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \\ & x_{ij} \in \{0, 1\} \\ & \lambda_1 \geq 0, \quad \lambda_2 \geq 0. \end{cases}$$

Nevertheless, in the multi-objective framework, there exist nonsupported efficient solutions, as indicated by the following didactic example:

$$C^{(1)} = \begin{pmatrix} 5 & 1 & 4 & 7 \\ 6 & 2 & 2 & 6 \\ 2 & 8 & 4 & 4 \\ 3 & 5 & 7 & 1 \end{pmatrix},$$

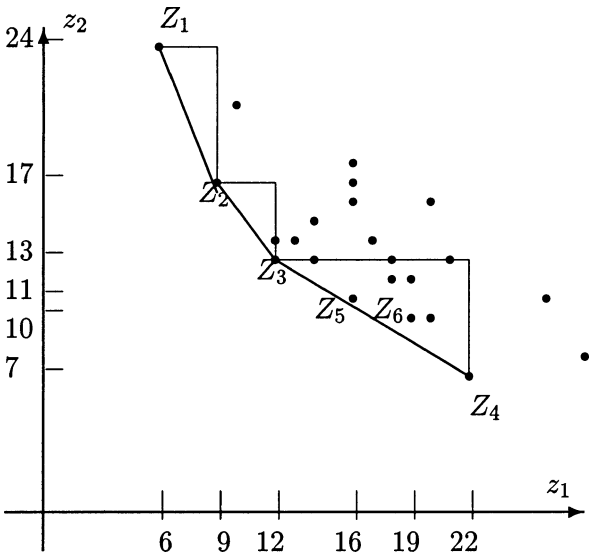
$$C^{(2)} = \begin{pmatrix} 3 & 6 & 4 & 2 \\ 1 & 3 & 8 & 3 \\ 5 & 2 & 2 & 3 \\ 4 & 2 & 3 & 5 \end{pmatrix}.$$

The values of the feasible solutions are represented in the objective space in Fig. 1

There are four supported efficient solutions, corresponding to points Z_1, Z_2, Z_3 and Z_4; two nonsupported efficient solutions corresponding to points Z_5 and Z_6; the eighteen other solutions are nonefficient.

Remark 1 In [7], D.J. White analyzes a particular case of problem (P) corresponding to

$$c_{ij}^{(k)} = c_{ij}\delta_{jk}$$



Bi-Objective Assignment Problem, Figure 1
The feasible points in the (z_1, z_2)-space for the didactic example

where

$$\delta_{jk} = \begin{cases} 1 & \text{if } j = k, \\ 0 & \text{if } j \neq k. \end{cases}$$

For this particular problem, he proves that $E(\mathrm{P}) = SE(\mathrm{P})$.

We consider the problem to generate $E(\mathrm{P})$ and (see ► Multi-objective combinatorial optimization) we can distinguish three methodologies: direct methods; two-phase methods and heuristic methods.

Direct Methods

In [1], the authors propose a theoretical enumerative procedure to generate $E(\mathrm{P})$ in the order of increasing values of z_1: at each step they consider the admissible edges incident at the current basis and among the set of possible new bases, they selected the one with the best value of z_1: they affirm that this basis corresponds to a new efficient solution. As proved by the example described above, this procedure appears false: for instance from point $Z_5 = (16, 11)$, corresponding to the solution $x_{14} = x_{22} = x_{33} = x_{41} = 1$, it is impossible to obtain by an unique change of basis the following point $Z_6 = (19, 10)$, corresponding to the solution $x_{13} = x_{21} = x_{34} = x_{42} = 1$. Moreover the real difficulties induced by the high degeneracy of the assignment problem are not taken into account in [1].

Two-Phase Methods

The principle of this approach, and the first phase designed to generate $SE(\mathrm{P})$, are described in ► Multi-objective combinatorial optimization; by complementary, we analyse here the second phase [3].

The purpose is to examine each triangle $\Delta Z_r Z_s$ determined by two successive solutions X^r and X^s of $SE(\mathrm{P})$ (see Fig. 2) and to determine the possible nonsupported solutions whose image lies inside this triangle. We note that

$$z_\lambda(X) = \lambda_1 z_1(X) + \lambda_2 z_2(X)$$

with $\lambda_1 = z_{2r} - z_{2s}$ and $\lambda_2 = z_{1s} - z_{1r}$ and $c_{ij}^{(\lambda)} = \lambda_1\, c_{ij}^{(1)} + \lambda_2\, c_{ij}^{(2)}$.

In the first phase, the objective function $z_\lambda(X)$ has been optimized by the Hungarian method giving

- $\widetilde{z}_\lambda = \lambda_1 z_{1r} + \lambda_2 z_{2r} = \lambda_1 z_{1s} + \lambda_2 z_{2s}$, the optimal value of $z_\lambda(X)$;
- the optimal value of the reduced cost $\overline{c}_{ij}^{(\lambda)} = c_{ij}^{(\lambda)} - (u_i + v_j)$, where u_i and v_j are the dual variables associated respectively to constraints i and j of problem $(\mathrm{P}\lambda)$.

At optimality, we have $\overline{c}_{ij}^{(\lambda)} \geq 0$ and $\widetilde{x}_{ij} = 1 \Rightarrow \overline{c}_{ij}^{(\lambda)} = 0$.

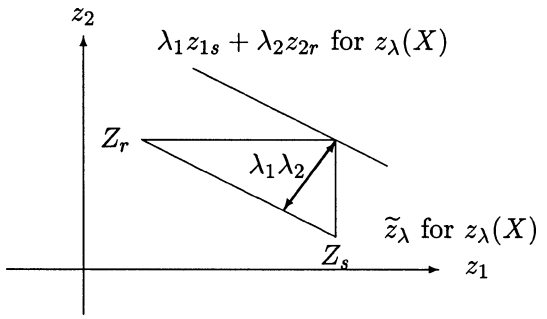
First Step

We consider $L = \left\{x_{ij}:\ \overline{c}_{ij}^{(\lambda)} > 0\right\}$. To generate nonsupported efficient solution in triangle $\Delta Z_r Z_s$, each variable $x_{ij} \in L$ is candidate to be fixed to 1. Nevertheless, a variable can be eliminated if we are sure that the reoptimization of problem $(\mathrm{P}\lambda)$ will provide a dominated point in the objective space. If $x_{ij} \in L$ is set to 1, a lower bound l_{ij} of the increase of $\widetilde{z}_\lambda$ is given by

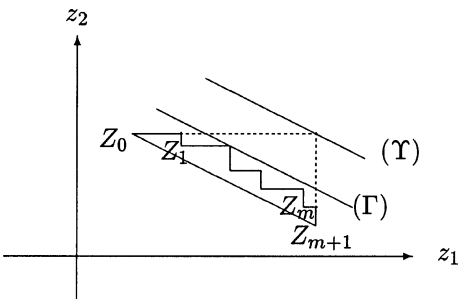
$$l_{ij} = \overline{c}_{ij}^{(\lambda)} + \min\left(\overline{c}_{i_r j_r}^{(\lambda)}; \min_{k \neq j} \overline{c}_{i_r k}^{(\lambda)} + \min_{k \neq i} \overline{c}_{k j_r}^{(\lambda)}; \overline{c}_{i_s j_s}^{(\lambda)}; \min_{k \neq j} \overline{c}_{i_s k}^{(\lambda)} + \min_{k \neq i} \overline{c}_{k j_s}^{(\lambda)}\right),$$

where the indices i_r and j_r (i_s and j_s) are such that in the solution X^r (respectively, X^s) we have

$$x_{i_r j} = x_{i j_r} = 1, \qquad (x_{i_s j} = x_{i j_s} = 1).$$



Bi-Objective Assignment Problem, Figure 2
Test 1



Bi-Objective Assignment Problem, Figure 3
Test 2

Effectively, to re-optimize problem (P_λ) with $x_{ij} = 1$, in regard with its optimal solution X^r (respectively, X^s), it is necessary to determine, at least, a new assignment in the line i_r (respectively, i_s) and in the column j_r (respectively, j_s). But clearly, to be inside the triangle $\Delta Z_r Z_s$, we must have (see Fig. 2)

$$\widetilde{z}_\lambda + l_{ij} < \lambda_1 z_{1s} + \lambda_2 z_{2r}.$$

Consequently, we obtain the following fathoming test:

- (*Test 1*): $x_{ij} \in L$ can be eliminated if $\widetilde{z}_\lambda + l_{ij} \geq \lambda_1 z_{1s} + \lambda_2 z_{2r}$ or, equivalently, if $l_{ij} \geq \lambda_1\lambda_2$.

So in this first step, the lower bound l_{ij} is determined for all $x_{ij} \in L$; the list is ordered by increasing values of l_{ij}.

Only the variables not eliminated by test 1 are kept. Problem (P_λ) is re-optimized successively for each noneliminated variable; let us note that only one iteration of the Hungarian method is needed. After the optimization, the solution is eliminated if its image in the objective space is located outside the triangle $\Delta Z_r Z_s$. Otherwise, a nondominated solution is obtained and put in a list NS_{rs}; at this time, the second step is applied.

Second Step

When nondominated points $Z_1, \ldots, Z_m \in NS_{rs}$ are found inside the triangle $\Delta Z_r Z_s$, then test 1 can be improved. Effectively (see Fig. 3), in this test the value

$$\lambda_1 z_{1s} + \lambda_2 z_{2r}$$

can be replaced by the lower value

$$(\Gamma) \equiv \max_{i=o,\ldots,m} \left(\lambda_1 z_{1,i+1} + \lambda_2 z_{2,i}\right),$$

where $Z_o \equiv Z_r$, $Z_m + 1 \equiv Z_s$, with $\Upsilon \equiv \lambda_1 z_{1,m+1} + \lambda_2 z_{2,0}$.

The new value corresponds to an updated upper bound of $z_\lambda(X)$ for nondominated points. More variables of L can be eliminated with the new test

- (*Test 2*): $x_{ij} \in L$ can be eliminated if

$$\widetilde{z}_\lambda + l_{ij} \geq \max_{i=o,\ldots,m} \left(\lambda_1 z_{1,i+1} + \lambda_2 z_{2,i}\right).$$

Each time a new nondominated point is obtained, the list NS_{rs} and the test 2 are updated. The procedure stops when all the $x_{ij} \in L$ have been either eliminated or analyzed. At this moment the list NS_{rs} contains the nonsupported solutions corresponding to the triangle $\Delta Z_r Z_s$.

When each triangle have been examined

$$NSE(P) = \cup_{rs} NS_{rs}.$$

Numerical results are given in [3].

Heuristic Methods

As described in ► Multi-objective combinatorial optimization, the MOSA method is an adaptation of the simulated annealing heuristic procedure to a multiobjective framework. Its aim is to generate a good approximation, denoted $\widehat{E(P)}$, of $E(P)$ and the procedure is valid for any number $K \geq 2$ of objectives. Similarly to a single objective heuristic in which a potentially optimal solution emerges, in the MOSA method the set $\widehat{E(P)}$ will contain potentially efficient solutions.

Preliminaries

- A wide diversified set of weights is considered: different weight vectors $\lambda^{(l)}$, $l \in L$, are generated where $\lambda^{(l)} = (\lambda_k^{(l)})_{k=1,\ldots,K}$ with $\lambda_k^{(l)} > 0$, $\forall k$ and

$$\sum_{k=1}^{K} \lambda_k^{(l)} = 1, \quad \forall l \in L.$$

- A scalarizing function $s(z, \lambda)$ is chosen, the effect of this choice on the procedure is small due to the stochastic character of the method. The weighted sum is very well known and it is the easiest scalarizing function:

$$s(z, \lambda) = \sum_{k=1}^{K} \lambda_k z_k.$$

- The three classic parameters of a simulated annealing procedure are initialized
  - T_0: initial temperature (or alternatively an initial acceptance probability P_0);
  - α (< 1): the cooling factor;
  - N_{step}: the length of temperature step in the cooling schedule;

  and the two stopping criteria are fixed:
  - T_{stop}: the final temperature;
  - N_{stop}: the maximum number of iterations without improvement
- A neighborhood $V(X)$ of feasible solutions in the vicinity of X is defined. This definition is problem dependent. It is particularly easy to define $V(X)$ in the case of the assignment problem: if X is characterized by $x_{ij_i} = 1$, $i = 1, \ldots, n$, then $V(X)$ contains all the solutions Y satisfying

$$y_{ij_i} = 1, \quad i \in \{1, \ldots, n\} \setminus \{a, b\},$$
$$y_{aj_b} = y_{bj_a} = 1,$$

  where a, b are chosen randomly in $\{1, \ldots, n\}$.

Determination of $PE(\lambda^{(l)})$, $l = 1, \ldots, L$

For each $l \in L$ the following procedure is applied to determine a list $PE(\lambda^{(l)})$ of potentially efficient solutions.

a) (Initialization):
   - Draw at random an initial solution X_0.
   - Evaluate $z_k(X_0)$, $\forall k$.
   - $PE(\lambda^{(l)}) = \{X_0\}$; $N_c = n = 0$.

b) (Iteration n):
   - Draw at random a solution $Y \in V(X_n)$
   - evaluate $z_k(Y)$ and determine

$$\Delta z_k = z_k(Y) - z_k(X_n), \quad \forall k.$$

   - Calculate

$$\Delta s = s(z(Y), \lambda) - s(z(X_n), \lambda).$$

If $\Delta s \leq 0$, we accept the new solution:

$$X_{n+1} \leftarrow Y \qquad N_c = 0.$$

Else we accept the new solution with a certain probability $p = \exp(-\Delta s / T_n)$:

$$X_{n+1} \begin{cases} \overset{p}{\leftarrow} Y, & N_c = 0, \\ \overset{1-p}{\leftarrow} X_n, & N_c = N_c + 1. \end{cases}$$

   - If necessary, update the list $PE(\lambda^{(l)})$ in regard to the solution Y.
   - $n \leftarrow n + 1$

     IF $n(\quad \text{mod } N_{\text{step}}) = 0$
     THEN $T_n = \alpha T_{n-1}$; ELSE $T_n = T_{n-1}$.
     IF $N_c = N_{\text{stop}}$ OR $T < T_{\text{stop}}$
     THEN stop ELSE iterate.

Generation of $\widehat{E(\mathrm{P})}$

Because of the use of a scalarizing function, a given set of weights $\lambda^{(l)}$ induces a privileged direction on the efficient frontier. The procedure generates only a good subset of potentially efficient solutions in that direction. Nevertheless, it is possible to obtain solutions which are not in this direction, because of the large exploration of D at high temperature; these solutions are often dominated by some solutions generated with other weight sets.

To obtain a good approximation $\widehat{E(\mathrm{P})}$ to $E(\mathrm{P})$ it is thus necessary to filter the set

$$\cup_{l=1}^{|L|} PE(\lambda^{(l)})$$

by pairwise comparisons to remove the dominated solutions. This filtering procedure is denoted by $\wedge$ such that

$$\widehat{E(\mathrm{P})} = \wedge_{l=1}^{|L|} PE(\lambda^{(l)}).$$

A great number of experiments is required to determine the number L of set of weights sufficient to give a good approximation of the whole efficient frontier.

Concluding Remarks

Details and numerical results are given in [3] and [5].

Let us add that it is easy to adapt the MOSA method in an interactive way [2]; a special real case study of an assignment problem is treated in this manner in [6].

See also

- ▶ Assignment and Matching
- ▶ Assignment Methods in Clustering
- ▶ Communication Network Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Frequency Assignment Problem
- ▶ Fuzzy Multi-Objective Linear Programming
- ▶ Maximum Partition Matching
- ▶ Multicriteria Sorting Methods
- ▶ Multi-Objective Combinatorial Optimization
- ▶ Multi-Objective Integer Linear Programming
- ▶ Multi-Objective Optimization and Decision Support Systems
- ▶ Multi-Objective Optimization: Interaction of Design and Control
- ▶ Multi-Objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-Objective Optimization: Lagrange Duality
- ▶ Multi-Objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling
- ▶ Quadratic Assignment Problem

References

1. Malhotra R, Bhatia HL, Puri MC (1982) Bicriteria assignment problem. Oper Res 19(2):84–96
2. Teghem J, Tuyttens D, Ulungu EL (2000) An interactive heuristic method for multi-objective combinatorial optimization. Comput Oper Res 27:621–624
3. Tuyttens D, Teghem J, Fortemps Ph, Van Nieuwenhuyse K (1997) Performance of the MOSA method for the bicriteria assignment problem. Techn Report Fac Polytechn Mons (to appear in J. Heuristics)
4. Ulungu EL, Teghem J (1994) Multi-objective combinatorial optimization problems: A survey. J Multi-Criteria Decision Anal 3:83–104
5. Ulungu EL, Teghem J, Fortemps Ph, Tuyttens D (1999) MOSA method: A tool for solving MOCO problems. J Multi-Criteria Decision Anal 8:221–236
6. Ulungu EL, Teghem J, Ost Ch (1998) Efficiency of interactive multi-objective simulated annealing through a case study. J Oper Res Soc 49:1044–1050
7. White DJ (1984) A special multi-objective assignment problem. J Oper Res Soc 35(8):759–767

Biquadratic Assignment Problem

BiQAP

Leonidas Pitsoulis
Princeton University, Princeton, USA

MSC2000: 90C27, 90C11, 90C08

Article Outline

Keywords
See also
References

Keywords

Optimization

The *biquadratic assignment problem* was first introduced by R.E. Burkard, E. Çela and B. Klinz [2], as a nonlinear assignment problem that has applications in very large scale integrated (VLSI) circuit design. Given two fourth-dimensional arrays $A = (a_{ijkl})$ and $B = (b_{mpst})$ with n^4 elements each, the nonlinear integer programming formulation of the BiQAP is

$$\begin{cases} \min & \displaystyle\sum_{i,j,k,l}\sum_{m,p,s,t} a_{ijkl}b_{mpst}x_{im}x_{jp}x_{ks}x_{lt} \\ \text{s.t.} & \displaystyle\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,\ldots,n, \\ & \displaystyle\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,\ldots,n, \\ & x_{ij} \in \{0,1\}, \quad i,j = 1,\ldots,n. \end{cases}$$

The BiQAP is a generalization of the quadratic assignment problem (cf. ▶ Quadratic assignment problem) (QAP), where the objective function is a fourth degree multivariable polynomial and the feasible domain is the

assignment polytope as in the QAP. An equivalent formulation of the BiQAP using permutations is the following:

$$\min_{\phi \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} a_{ijkl} b_{\phi(i)\phi(j)\phi(k)\phi(l)},$$

where S_n denotes the set of all permutations of the integer set $N = \{1, \ldots, n\}$.

Burkard, Çela and Klinz [2] showed that the BiQAP is *NP*-hard. They computed lower bounds for BiQAP derived from lower bounds of the QAP. The computational results showed that these bounds are weak and deteriorate as the dimension of the problem increases. This observation suggests that branch and bound methods (cf. also ► Integer programming: Branch and bound methods) will only be effective on very small instances. For larger instances, efficient heuristics, that find good-quality approximate solutions, are needed.

Burkard and Çela [1] developed several heuristics for the BiQAP, in particular deterministic improvement methods and variants of simulated annealing and tabu search. Computational experiments on test problems with known optimal solutions [1], suggest that one version of simulated annealing is best among those tested. T. Mavridou, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende develop a GRASP heuristic for solving the BiQAP in [3], which finds the optimal solution for all the test problems presented in [1].

See also

- ► Feedback Set Problems
- ► Generalized Assignment Problem
- ► Graph Coloring
- ► Graph Planarization
- ► Greedy Randomized Adaptive Search Procedures
- ► Quadratic Assignment Problem

References

1. Burkard RE, Çela E (1995) Heuristics for biquadratic assignment problems and their computational comparison. Europ J Oper Res 83:283–300
2. Burkard RE, Çela E, Klinz B (1994) On the biquadratic assignment problem. In: Pardalos PM , Wolkowicz H (eds) Quadratic assignment and related problems. DIMACS Amer Math Soc, Providence, RI, pp 117–146
3. Mavridou T, Pardalos PM, Pitsoulis LS, Resende MGC (1998) A GRASP for the biquadratic assignment problem. Europ J Oper Res 105(3):613–621

Bisection Global Optimization Methods

Graham Wood
Institute Information Sci. and Technol. College of Sci., Massey University, Palmerston North, New Zealand

MSC2000: 90C30, 65K05

Article Outline

Keywords
See also
References

Keywords

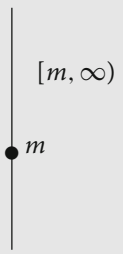
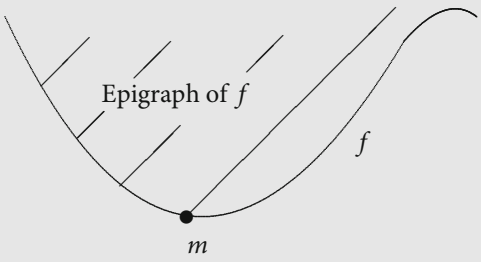
Lipschitz continuity; Bisection; Multidimensional bisection; Bracket; Simplex; Epigraph; System; Reduction; Elimination; Linear convergence; Tiling

The centuries-old method of bisection can be generalized to provide a global optimization algorithm for Lipschitz continuous functions. Full details of the algorithm, acceleration methods and its performance can be found in [1,6,7]. (Recall that $f : \mathbf{R}^n \to \mathbf{R}$ is *Lipschitz continuous* if there is an $M \geq 0$ such that $|f(x) - f(y)| \leq M \| x - y \|$ for all $x, y \in \mathbf{R}^n$. We then term M a *Lipschitz constant* of f.)

The familiar *bisection method* enables us to find a point of interest on the line by first bracketing the point in an interval, and then successively halving the interval. It is used in this way, for example, to find the root of a continuous function or to show that a bounded sequence always has a limit point. The bisection method is simple and convergence is assured and linear.

The bisection method can also be used (although we never think of it in this role) to find the minimum of a semi-infinite interval $[m, \infty)$, as illustrated in the left-hand side of Table 1. Given an initial interval bracket around m we examine the midpoint: if the midpoint is

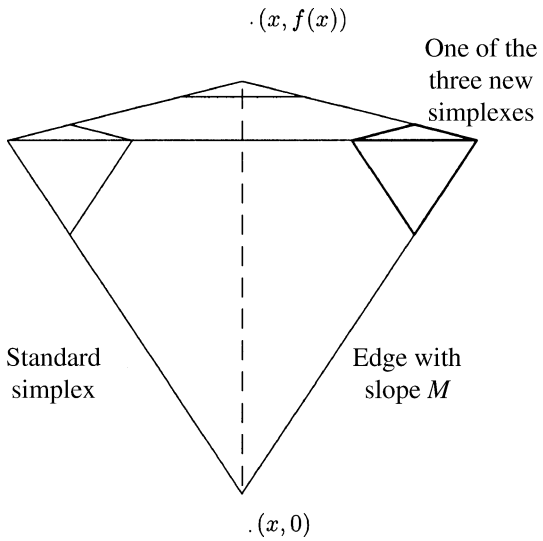
Bisection Global Optimization Methods, Table 1
A comparison of the bisection method and the generalization to higher dimensions

Bisection ($n = 0$)	*Multidimensional bisection* ($n > 0$)
Problem	
	
Find m	Find m
Natural domain (in $\mathbf{R}^n$*)*	
Point	line segment ($n = 1$) hexagon ($n = 2$) rhombic dodecahedron ($n = 3$) ...
Initial bracket (in $\mathbf{R}^{n+1}$*)*	
Single interval	Union of $(n + 1)$-dimensional simplexes
Bracket reduction	
Interval halving	Reduction of simplexes, followed by elimination
Convergence	
$\cap$all brackets $= \{m\}$ Bracket size halves	$\cap$all brackets = {all global minima} Bracket depth reduces linearly

in $[m, \infty)$ then we retain the lower interval whereas if the midpoint is not in $[m, \infty)$ we retain the upper interval. It is this idea that has been generalized to higher dimensions to give the algorithm, detailed here, that has been termed in the literature *multidimensional bisection*.

It can be shown (see [7]) that the analogue in $\mathbf{R}^{n+1}$ of an upper semi-infinite interval in $\mathbf{R}$ is the *epigraph* (everything above and including the graph) of a Lipschitz continuous function. Multidimensional bisection finds the set of global minima of a Lipschitz continuous function f of n variables over a compact domain, in a manner analogous to the bisection method. At any stage in the iteration the bracket is a union of similar simplexes in $\mathbf{R}^{n+1}$, with the initial bracket a single *simplex*. (A simplex is a convex hull of affinely independent points, so a triangle, a tetrahedron and so on.) In the raw version of the algorithm the depth of the bracket decreases linearly and the infinite intersection of all brackets is the set of global minima of the graph of the function.

The algorithm works thanks to two simple facts and a very convenient piece of geometry. First, however, we note a property of a Lipschitz continuous function with Lipschitz constant M: if $x \in \mathbf{R}^n$ lies in the domain of the function and (x, y) (with $y \in \mathbf{R}$) lies in the epigraph of the function, then $(x, y) + C$ lies in the epigraph, where C is an upright spherically based cone of slope M, with apex at the origin.



Bisection Global Optimization Methods, Figure 1
A standard simplex and the three smaller standard simplexes resulting from reduction; when $(x, f(x)) - \Delta$ is removed from the standard simplex three similar standard simplexes remain

Now for the two simple facts: if we evaluate the function f at any point in the domain, then no point higher than $(x, f(x))$ can be the global minimum on the graph of f and no point in the interior of a $(x, f(x)) - C$ can be the global minimum. Informally, this means that every evaluation of f lets us slice away an upper half space and an upside down ice-cream cone, with apex at $(x, f(x))$, from the space $\mathbf{R}^{n+1}$; we are sure the global optima are not there. These two operations coalesce in the familiar bisection method.

Now for the convenient geometry, which comes to light as soon as we attempt to generalise the bisection method. Spherically based cones are ideal to use, but hard to keep track of efficiently [3], so we use a simplicial approximation to the spherical base of the cone to make the bookkeeping easy. Such a simplex-based cone, Δ, has a cap which we call a *standard simplex*; one is shown as the large simplex in Fig. 1, for the case when $n = 2$. It fits snugly inside C, so the sloping edges have slope M. If we know that the global optimum lies in this simplex bracket and evaluate f at x, then we can remove $(x, f(x)) - \Delta$ from the space. Conveniently, this leaves three similar standard simplexes whose union must contain the global minima, as shown in Fig. 1. This process is termed *reduction* of the simplex.

What does a typical iteration of the algorithm do? At the start of each iteration the global minima are held in a *multidimensional bracket*, a union of similar standard simplexes. We denote this set of simplexes, or *system*, by $\mathcal{S}$. An iteration consists of reducing some (possibly all) of these simplexes, followed by *elimination*, or retaining the portions of the bracket at the level of, or below, the current lowest function evaluation. For this reason an iteration can be thought of informally as 'chop and drop', or formally as 'reduce and eliminate'.

How do we start off? The algorithm operates on certain *natural domains* which we must assume contain a global minimizer (just as we begin in the familiar bisection method by containing the point of interest in an interval). For functions of one variable a natural domain is an interval, for functions of two variables it is a hexagon, while for functions of three variables the natural domain is a rhombic dodecahedron (the honeycomb cell). For higher dimensions the pattern continues; in each dimension the natural domains are capable of tiling the space. By means of $n + 1$ function evaluations at selected vertices of the natural domain it is possible to bracket the global optima over the natural domain in an initial single standard simplex, termed the *initial system*.

In brief, given a Lipschitz continuous function f on a standard domain, the algorithm can be summarised as:

1. Set $i = 0$ and form the initial system S_0.
2. Form S_{i+1}, by applying reduction and then elimination to the system S_i.
3. If a stopping criterion is satisfied (such as that the variation of the system is less than a preassigned amount), then stop. Otherwise, increment i and return to Step 2.

Multidimensional bisection

By the *variation of the system* is meant the height from top to bottom of the current set of simplexes. The following example illustrates the course of a run of multidimensional bisection.

Take $f(x_1, x_2) = -e^{-x_1^2} \sin x_1 + |x_2|$, which has a global minimum on its graph at (0.653273, 0, −0.396653). There are also local minima along the

Bisection Global Optimization Methods, Table 2
Example of a run of multidimensional bisection. Note how the number of simplexes in the system decreases in the 8th iteration; this corresponds to the elimination of simplexes around local, and nonglobal, minima

Iter	Simpl. in the system	Variat.	Best point to date
0	1	33.300	(10.000, −10.000, 10.000)
1	3	20.000	(10.000, 6.667, 6.667)
2	9	9.892	(1.340, 1.667, 1.505)
5	108	1.959	(25.637, 0.185, 0.185)
7	264	0.504	(0.839, 0.074, −0.294)
8	39	0.257	(0.649, −0.036, −0.361)
15	369	0.007	(0.669, 0.000, −0.396)
18	924	0.001	(0.653, 0.000, −0.397)
19	1287	0.000	(0.651, 0.000, −0.397)

x_1-axis. We use as our standard domain the regular hexagon with center at (10, 10) and radius 20, and use $M = 1$. Table 2 provides snapshots of the progress of the algorithm to convergence; it stops when the variation is less than 0.001. We carry the best point to date, shown in the final column of the table.

In this example we reduced all simplexes in the system at each iteration. This ensures that the infinite intersection of the brackets is the set of global minima. In [6] it is shown that, under certain conditions, the optimal one-step strategy is to reduce only the deepest simplex in each iteration. With this reduction and $n = 1$ multidimensional bisection is precisely the *Piyavskii–Shubert algorithm* [4,5].

Raw multidimensional bisection can require a large number of function evaluations, but can be economical with computer time (see [2]). As described so far, the method does not use the full power of the spherical cone, rather a simplicial approximation, and this approximation rapidly worsens as the dimension increases. Fortunately, much of the spherical power can be utilized very simply, by raising the function evaluation to an effective height. This is trivial to implement and has been called *spherical reduction* [6]. Reduction, as described so far, removes material only from a single simplex, whose apex determines the evaluation point. Simplexes overlap when $n \geq 2$, and it is possible to remove material from many simplexes rather than just one. This is harder to implement, but has been carried out in [1] where it is termed *complete reduction*. The algorithm operates more efficiently when such improved reduction methods are used.

Multidimensional bisection collapses to bisection with $n = 0$ when we use a primitive reduction process, one which depends only on whether the point in $\mathbf{R}^{n+1}$ considered lies in the epigraph of f; this is described in [7]. A summary comparison of bisection and multidimensional bisection is given in Table 1.

See also

► αBB Algorithm

References

1. Baoping Zhang, Wood GR, Baritompa WP (1993) Multidimensional bisection: the performance and the context. J Global Optim 3:337–358
2. Horst R, Pardalos PM (eds) (1995) Handbook of Global Optimization. Kluwer, Dordrecht
3. Mladineo RG (1986) An algorithm for finding the global maximum of a multimodal, multivariate function. Math Program 34:188–200
4. Piyavskii SA (1972) An algorithm for finding the absolute extremum of a function. USSR Comput Math Math Phys 12: 57–67
5. Shubert BO (1972) A sequential method seeking the global maximum of a function. SIAM J Numer Anal 9:379–388
6. Wood GR (1991) Multidimensional bisection and global optimisation. Comput Math Appl 21:161–172
7. Wood GR (1992) The bisection method in higher dimensions. Math Program 55:319–337

Boolean and Fuzzy Relations

Ladislav J. Kohout
Department Computer Sci., Florida State University, Tallahassee, USA

MSC2000: 03E72, 03B52, 47S40, 68T27, 68T35, 68Uxx, 91B06, 90Bxx, 91Axx, 92C60

Article Outline

Keywords
Boolean Relations
Propositional Form
Heterogeneous and Homogeneous Relations

The Satisfaction Set
The Extensionality Convention
The Digraph Representation
Foresets and Aftersets of Relations
Matrix Representation
Operations and Inclusions in $\mathcal{R}(A \rightsquigarrow B)$
Unary Operations
Binary Operations on Successive Relations
Matrix Formulation of the Binary Operations
Non-Associative Products of Relations
Characterization of Special Properties of Relations Between Two Sets
Relations on a Single Set: Special Properties
Partitions IN and ON a Set
Tolerances and Overlapping Classes
Hierarchies in and on a Set: Local and Global Orders and Pre-orders
Fuzzy Relations
Definitions
Operations and Inclusion on $\mathcal{R}_F$ $(X \rightsquigarrow Y)$
Fuzzy Relations with Min, Max Connectives
Fuzzy Relations Based on Łukasiewicz Connectives
Fuzzy Relations With t-Norms and Co-Norms
Products: $\mathcal{R}_F(X \rightsquigarrow Y) \times \mathcal{R}_F(Y \rightsquigarrow Z) \to \mathcal{R}_F(X \rightsquigarrow Z)$
N-ary Relations
Special Properties of Fuzzy Relations
Alpha-cuts of Fuzzy Relations
Fuzzy Partitions, Fuzzy Clusters and Fuzzy Hierarchies
Closures and Interiors with Special Properties
Applications of Relational Methods in Engineering, Medicine and Science
Brief Review of Theoretical Development
Basic Books and Bibliographies
See also
References

Keywords

Fuzzy relations; Local relational properties; Closures; Interiors; Pre-order; Tolerances; Equivalences; BK-products; Relational compositions; Nonassociative products; Generalized morphism; Universal properties of relations; n-ary relation; Scientific applications; Medicine; Psychology; Engineering applications; Artificial intelligence; Value analysis; Decision theory

The conventional nonfuzzy relations using the classical two-valued Boolean logic connectives for defining their operations will be called crisp. The extensions that replace the 2-valued Boolean logic connectives by many-valued logic connectives will be called fuzzy. A unified approach of relations is provided here, so that the Boolean (crisp, nonfuzzy) relations and sets are just special cases of fuzzy relational structures. The first part of this entry on nonfuzzy relations can be used as reference independently, without any knowledge of fuzzy sets. The second part on fuzzy structures, however, refers frequently to the first part. This is so because most formulas in the matrix notation carry over to the many-valued logics based extensions.

In order to make this material useful not only theoretically but also in practical applications, we have paid special attention to the form in which the material is presented. There are seven distinguishing features of our approach that facilitate the unification of crisp and fuzzy relations and enhance their practical applicability:

1) Relations in their predicate forms are distinguished from their satisfaction sets.
2) Foresets and aftersets of relations are used in addition to relational predicates.
3) Relational properties are not only global but also local (important for applications).
4) Nonassociative *BK-products* are introduced and used both in definitions of relational properties and in computations.
5) The unified treatment of computational algorithms by means of matrix notation is used which is equally applicable to both crisp and fuzzy relations.
6) The theory unifying crisp and fuzzy relations makes it possible to represent a whole *finite nested family* of crisp relations with special properties as a *single* cutworthy fuzzy relation for the purpose of computation. After completing the computations, the resulting fuzzy relation is again converted by α-cuts to a nested family of crisp relations, thus increasing the computing performance considerably.
7) *Homomorphisms* between relations are extended from mappings used in the literature to general relations. This yields *generalized morphisms* important for practical solving of relational inequalities and equations.

These features were first introduced in 1977 by W. Bandler and L.J. Kohout [1] and extensively developed over the years both in theory and practical applications [7,30,52].

Boolean Relations

Propositional Form

A *binary relation* (from A to B) is given by an open predicate __P__ with two empty slots; when the first is filled with the name a of an element of A and the second with the name b of an element of B, there results a proposition, which is either true or false. If aPb is true, we write aR_Pb and say that 'a is R_P-related to b'. If $a\ P\ b$ is false, we write $a \neg R_P\ b$ and say that 'a is not R_P-related to b', etc. When it is unnecessary to emphasize the propositional form the subscript is dropped in R_P, writing: R, $a\ Rb$, $a \neg Rb$, respectively.

Heterogeneous and Homogeneous Relations

The lattice of all binary (two-place, 2-argument) relations from A to B is denoted by $\mathcal{R}(A \rightsquigarrow B)$. Relations of this kind are usually called *heterogeneous*. Nothing forbids the set B to be the same as A, in which case we speak of relations 'within a set' or 'in a set', or 'on a set', and call these *homogeneous*.

Relations from A to B can always be considered as relations within $A \cup B$, but so 'homogenized' relations may lose some valuable properties (discussed below), when so viewed. For this reason, we do not attempt to assimilate relations between distinct sets to those within a set.

The Satisfaction Set

The *satisfaction set* or *representative set* or *extension set* of a relation $R \in \mathcal{R}(A \rightarrow B)$ is the set of all those pairs $(a, b) \in A \times B$ for which it holds:

$$R_S = \{(a, b) \in A \times B\colon\ aRb\}\,.$$

Clearly R_S is a subset of the Cartesian product $A \times B$. Knowing R_P, we know R_S; knowing R_S, we know everything about R_P except the wording of its 'name' __P__.

The Extensionality Convention

This convention says that, regardless of their propositional wordings, two relations should be regarded as the same if they hold, or fail to hold between exactly the same pairs: $R_S = R_S' \Rightarrow R_P = R_P'$. In the set theory, this appears as the *axiom of extensionality*. This convention is not universally convenient; it is perhaps partly responsible for delays in the application of relation theory in the engineering, social and economical sciences and elsewhere.

Once the extensionality convention has been adopted, it becomes a matter of indifference, or mere convenience, whether a relation is given by an open predicate or by the specification of its satisfaction set. There is a one-to-one correspondence between the subsets R_S of $A \times B$ and the (distinguishable) relations R_P in $\mathcal{R}(A \rightarrow B)$. Since R_S and R_P now uniquely determine each other, the current fashion for set-theoretical parsimony suggests that they be identified. This view is common in the literature, which often *defines* relations as being satisfaction sets. We, however, maintain the distinction in principle.

Example of the failure of the extensionality convention

$R_{\geq}, Q_{>} \in \mathcal{R}(A \rightsquigarrow B)$; $A = \{1, 6, 8\}$, $B = \{0, 5, 7\}$.
Predicates:
$P1$:= '__ $\geq$ __' ('__ is greater than or equal to __')
$P2$:= '__ > __' ('__ is greater than __')
Relations in their Predicate Form:
$R_{\geq} = \{1 \geq 0, 8 \geq 0, 8 \geq 5, 8 \geq 7, 6 \geq 0, 6 \geq 5\}$
$Q_{>} = \{1 > 0, 8 > 0, 8 > 5, 8 > 7, 6 > 0, 6 > 5\}$
The Satisfaction Sets:
$R_S = Q_S = \{(1, 0), (8, 0), (8, 5), (8, 7), (6, 0), (6, 5)\}$.
By the *extensionality convention*:
$R_S = Q_S \Rightarrow R_{\geq} = Q_{>}$.
So, R should be the same relation as Q. This is not the case, because the predicates are not equivalent:
$(\forall x)\ x\ P1\ x$ is true, but $(\forall x)\ x\ P2\ x$ is false.
Hence the extensionality convention fails for these relations.

The Digraph Representation

When $B = A$, so that we are dealing with a relation within a set, we may use the *digraph* R_D to represent it; in which an arrow goes from a to a' if and only if $a\ R\ a'$. Any relation within a finite or countably infinite set can, in principle, be shown in a digraph; conversely, every digraph (with unlabelled arrows) represents a relation in the set of its vertices. Interesting properties of relations are often derived from digraphical considerations; there is a whole literature on digraphs.

Foresets and Aftersets of Relations

These are defined for any relation R from A to B.

- The *afterset* of $a \in A$ is

$$aR = \{b \in B\colon\ aRb\}\,.$$

- The *foreset* of $b \in B$ is

$$Rb = \{a \in A\colon\ aRb\}\,.$$

Mnemonically and semantically, an afterset consists of all those elements which can correctly be written after a given element, a foreset of those which can correctly be written before it. An afterset or foreset may well be empty.

Clearly, $b \in a\ R$ if and only if $a \in R\ b$. A relation is completely known if all its foresets or all its aftersets are known.

Matrix Representation

Very important computationally and even conceptually, as well as being a useful visual aid, is the *incidence matrix* R_M of a relation R. This arises from a table in which the row-headings are the elements of A and the column-headings are the elements of B, so that the cells represent $A \times B$. In the (a, b)-cell is entered 1 if $a\ R\ b$, and 0 if $a \neg Rb$. For visual purposes it is better to suppress the 0s, but they should be understood to be there for computational purposes.

	b_1	b_2	b_3
a_1	1		
a_2	1	1	1
a_3	1	1	
a_4			

$$a_1R = \{b_1\}$$
$$a_2R = \{b_1, b_2, b_3\}$$
$$a_3R = \{b_1, b_2\}$$
$$a_1R = \emptyset$$

Example: The matrix representation R_M and the afterset representation of a relation R

Clearly there is a one-to-one correspondence (bijection) between distinct tables and distinct relations, and, as soon as there has been agreement on the names and ordering of the row and column headings, between either of these and distinct matrices of size $|A| \times |B|$ with entries from {0, 1}.

Furthermore, the afterset a_iR is in one-to-one correspondence with the nonzero entries of the ith row of R_M; the foreset Rb_j is in one-to-one correspondence with the nonzero entries of the jth column of R_M.

Operations and Inclusions in $\mathcal{R}(A \rightsquigarrow B)$

There are a considerable number of natural and important operations. We begin with unary operations and then proceed to several kinds of binary ones.

Unary Operations

The *negated* or *complementary relation* of $R \in \mathcal{R}(A \to B)$ is $\neg R \in \mathcal{R}(A \to B)$ given by $a \neg R\ b$ if and only if it is not the case that aRb.

The *converse* or *transposed relation* of $R \in \mathcal{R}(A \to B)$ is $R^{\intercal} \in \mathcal{R}(B \to A)$ given by

$$bR^{\top}a \quad \Leftrightarrow \quad aRb.$$

(It is also called the *inverse* and is therefore often written R^{-1}. In no algebraic sense it is an inverse, in general.)

Both operators $^{\intercal}$ and $\neg$ are *involutory*, that is, when applied twice they give the original object: $(R^{\intercal})^{\intercal} = R$, $\neg(\neg R) = R$. They commute with each other: $\neg(R^{\intercal}) = (\neg R)^{\intercal}$, so that the parentheses may be omitted safely. One can write: $\neg R^{\intercal}$.

Definition 1 (Binary operators and a binary relation on $\mathcal{R}(A \to B)$)

- The *intersection* or *meet* or *AND-ing*:

$$a(R \sqcap R')b \quad \Leftrightarrow \quad aRb \text{ and } aR'b.$$

- The *union* or *join* or *OR-ing*:

$$a(R \sqcup R')b \quad \Leftrightarrow \quad aRb \text{ or } aR'b.$$

- A relation R 'is contained in' (is a *subrelation* of) a relation R', and R' 'contains' (is a *superrelation* of) R', $R \sqsubseteq R'$:

$$R \sqsubseteq R' \quad \Leftrightarrow \quad (\forall a)(\forall b)(aRb \to aR'b)$$
$$\Leftrightarrow \quad R \sqcap R' = R \quad \Leftrightarrow \quad R \sqcup R' = R',$$

where $\to$ is the Boolean *implication operator*.

Definition 2 The *relative complement* of R with respect to R', or *difference* between R' and R, is $R' \setminus R$, given by:

$$a(R' \setminus R)b \quad \Leftrightarrow \quad aR'b \text{ but } a\neg Rb,$$

that is, by $R' \setminus R = R' \sqcap \neg R$.

Binary Operations on Successive Relations

Definition 3 (Circle and square products) Where $R \in \mathcal{R}(A \to B)$ and $S \in \mathcal{R}(B \to C)$, the following compositions give a relation in $\mathcal{R}(A \to C)$:

- The *circle product* or *round composition* is $\circ$, given by $aR \circ Sc \Leftrightarrow aR \cap Sc \neq \emptyset$.
- The *square composition* or *square product* is $\square$, given by $aR \square Sc \Leftrightarrow aR = Sc$.

The circle product is the usual one, to be found throughout the literature going back at least to the nineteenth century. The square product is a more recent (1977) innovation. The $\square$ product belongs to the family of products sometimes called *BK-products*. Further interesting kinds of BK-products and their uses are discussed in the sequel.

Proposition 4 *(Properties of $\square$-product)*

1) $(R \square S) \sqcap (R' \square S) \sqsubseteq (R \sqcap R') \square S \sqsubseteq (R \sqcup R') \square S \sqsubseteq (R \square S) \sqcup (R' \square S)$;
2) $(R \square S)^{-1} = S^{-1} \square R^{-1}$;
3) $R \square S = \neg R \square \neg S$;
4) *the square product is not associative.*

Matrix Formulation of the Binary Operations

All of the *binary operations on relations* have a convenient formulation in matrix terms – using the matrix operations given in Proposition 6. The matrix operations use in their definitions standard Boolean logic connectives for *crisp relations*. By replacing these by the connectives of suitable many-valued logics, all the formulas easily generalize to fuzzy relations. Thus matrix formulation of binary operations and compositions unifies computationally crisp and fuzzy relations.

Definition 5 The *Boolean connectives* $\wedge$, $\vee$, $\leftrightarrow$, on the set $\mathcal{B}_2 = \{0, 1\}$ are given by:

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

$\leftrightarrow$	0	1
0	1	0
1	0	1

For a pair (x_1, x_2) of elements from $\mathcal{B}_2$, we infix the operators: $x_1 \wedge x_2$, etc., while for a list $(x_k)_{k=1,\ldots,n}$ or $(x_k)_{k \in K}$ of elements from $\mathcal{B}_2$, we write $\bigwedge_{k=1}^{n} x_k$ or $\bigwedge_{k \in K} x_k$ or simply $\bigwedge_k x_k$. (Note that K can be denumerably infinite, or even greater, without spoiling the definition; no convergence problems are involved.)

Proposition 6 (*Matrix notation*)

1) $(R \sqcap S)_{ij} = R_{ij} \wedge S_{ij}$;
2) $(R \sqcup S)_{ij} = R_{ij} \vee S_{ij}$;
3) $(R \circ S)_{ij} = \bigvee_k (R_{ik} \wedge S_{kj})$;
4) $(R \bullet S)_{ik} = \bigwedge_j (R_{ij} \vee S_{jk})$;
5) $(R \square S)_{ik} = \bigwedge_j (R_{ij} \equiv S_{jk})$;
6) $(R_1 \times R_2)_{i_1 i_2 j_1 j_2} = (R_1)_{i_1 j_1} \wedge (R_2)_{i_2 j_2}$.

Non-Associative Products of Relations

Definition 7 (Triangle products)

- *Subproduct* $\triangleleft$: $x(R \triangleleft S)\, z \Leftrightarrow xR \subseteq Sz$;
- *Superproduct* $\triangleright$: $x(R \triangleright S)\, z \Leftrightarrow xR \supseteq Sz$.

The matrix formulation of $\triangleleft$ and $\triangleright$ products uses the Boolean connectives $\rightarrow$, $\leftarrow$, $\oplus$ on the set $\mathcal{B}_2 = \{0, 1\}$ given by

$\rightarrow$	0	1
0	1	1
1	0	1

$\leftarrow$	0	1
0	1	0
1	1	1

$\oplus$	0	1
0	0	1
1	1	0

Proposition 8 (Logic notation for $\triangleleft$ and $\triangleright$)

- $(R \triangleleft S)_{ik} = \bigwedge_j (R_{ij} \rightarrow S_{jk})$;
- $(R \triangleright S)_{ik} = \bigwedge_j (R_{ij} \leftarrow S_{jk})$.

Only the conventional $\circ$ -product is associative. The $\square$-product is not associative [2].

Proposition 9 *The following mixed pseudo-associativities hold for the triangle products, with $Q \in \mathcal{B}(W \rightsquigarrow X)$ and the triple products in $\mathcal{B}(W \rightsquigarrow Z)$:*

- $Q \triangleleft (R \triangleright S) = (Q \triangleleft R) \triangleright S$;
- $Q \triangleleft (R \triangleleft S) = (Q \circ R) \triangleleft S$;
- $Q \triangleright (R \triangleright S) = Q \triangleright (R \circ S)$.

Characterization of Special Properties of Relations Between Two Sets

Definition 10 (Special properties of a heterogeneous relation $R \in \mathcal{R}(X \rightsquigarrow Y)$):

- R is *covering* if and only if $(\forall x) \in X\ (\exists y) \in Y$ such that xRy.
- R is *onto* if and only if $(\forall y) \in Y\ (\exists x) \in X$ such that xRy.
- R is *univalent* if and only if $(\forall x) \in X$, if xRy and xRy' then $y = y'$.
- R is *separating* if and only if $(\forall y) \in Y$, if xRy and $x'Ry$ then $x = x'$.

Composed properties can be defined by combining these four basic properties. Well-known is the combination 'covering' and 'univalent' which defines *functional*. Other frequently used combination is 'onto' and 'separating'.

The self-inverse circle product is very useful in the characterization of *special properties of relations* between two distinct sets. Using the product, one can characterize these properties in purely relational way, without directly referring to individual elements of the relations involved.

Proposition 11 *(Special properties of a heterogeneous relation $R \in \mathcal{R}\ (X \rightsquigarrow Y)$):*

- *R is covering if and only if $E_X \sqsubseteq R \circ R^{-1}$.*
- *R is univalent if and only if $R^{-1} \circ R \sqsubseteq E_Y$.*
- *R is onto if and only if (for all) $E_Y \sqsubseteq R^{-1} \circ R$.*
- *R is separating if and only if $R \circ R^{-1} \sqsubseteq E_X$.*

Here E_X and E_Y are the left and right identities, respectively.

Relations on a Single Set: Special Properties

The *self-inverse products* are a fertile source of relations on the single set X. There are certain well-known special properties which a relation may possess (or may lack), of which the most important are reflexivity, symmetry, antisymmetry, strict antisymmetry, and transitivity, together with their combinations, forming *preorders* (reflexive and transitive) (*partial*) *orders* (reflexive, antisymmetric and transitive), *equivalences* (reflexive, transitive and symmetric).

Definition 12 (Special properties of binary relations from X to X)

- *Covering*: every x_i is related by R to something $\Leftrightarrow \forall i \in I\ \exists j \in I$ such that $R_{ij} = 1$.
- *Locally reflexive*: if x_i is related to anything, or if anything is related to x_i, then x_i is related to itself $\Leftrightarrow \forall i \in I\ R_{ii} = \max_j (R_{ij}, R_{ji})$.
- *Reflexive*: covering and locally reflexive $\Leftrightarrow \forall i \in I\ R_{ii} = 1$.
- *Transitive* $\forall i, j, k \in I\ (x_i R x_j$ and $x_j R x_k \Rightarrow x_i R x_k) \Leftrightarrow R^2 \sqsubseteq R$.
- *Symmetric*: $(x_i R x_j \Rightarrow x_j R x_i) \Leftrightarrow R^{\mathsf{T}} = R$.
- *Antisymmetric*: $(x_i R x_j$ and $x_j R x_i \Rightarrow x_i = x_j) \Leftrightarrow$ if $i \neq j$ then $\min(R_{ij}, R_{ji}) = 0$.
- *Strictly antisymmetric*: never both $x_i R x_j$ and $x_j R x_i \Leftrightarrow \forall i, j \in I\ \min(R_{ij}, R_{ji}) = 0$.

Most of the properties listed above are common in the literature. Local reflexivity is worthwhile exception. It appeared in [1] and was generalized to fuzzy relations in [4], leading to new computational algorithms for both crisp and fuzzy relations [4,10]. Unfortunately, it is absent from the textbooks, yet it is extremely important in applications of relational methods to analysis of the real life data (see the notion of participant in the next two sections).

Partitions IN and ON a Set

A *partition on a set* X is a division of X into nonoverlapping (and nonempty) subsets called blocks. A *partition in a set* X is a partition on the subset of X [17,18] called the *subset of participants*.

There is a one-to-one correspondence between partitions in X and *local equivalences* (i. e. locally reflexive, symmetric and transitive relations) in $\mathcal{R}(X \rightsquigarrow B)$. The partitions in X (so also the local equivalences in $\mathcal{R}(X \rightsquigarrow B)$) form a lattice with '__is-finer-than__' as its ordering relation. This whole subject is coextensive with *classification* or *taxonomy*, i. e., very extensive indeed. Furthermore, classification is the first step in abstraction, one of the fundamental processes in human thought.

Tolerances and Overlapping Classes

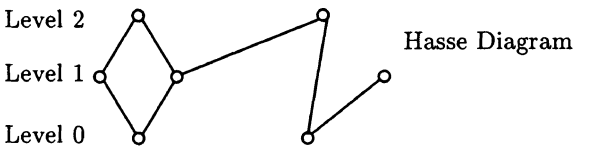
Some tests for tolerance and equivalence are as follows:

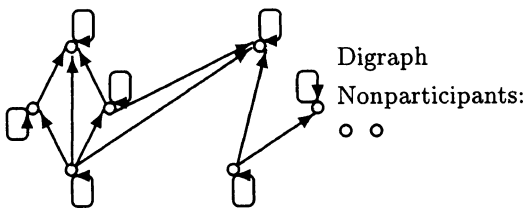
- $R \circ R^{\mathsf{T}}$ is always symmetric and locally reflexive.
- $R \circ R^{\mathsf{T}}$ is a tolerance if and only if R is covering.
- $R \square R^{\mathsf{T}}$ is always a (local) tolerance.
- $R \square R^{\mathsf{T}} \sqsubseteq R$ if and only if R is reflexive.
- $E \sqsubseteq R \sqsubseteq R \square R^{\mathsf{T}}$ if and only if R is an equivalence.
- $R \square R^{\mathsf{T}} = R$ if and only if R is an equivalence.
- $R \square R^{\mathsf{T}} \sqsubseteq R \circ R^{\mathsf{T}}$ if and only if R is covering.

It is not always the case that one manages, or even attempts, to classify participants into nonoverlapping blocks. *Local tolerance relations* (i. e. locally reflexive and symmetric) lead to classes which may well overlap, where one participant may belong to more than one class. The classic case, giving its name to this kind of relation, is '__is-within-one milimeter-of__'. This is quite a different model from the severe partitions [80], and has been for a long time unduly neglected both in theory and applications, even when the data mutely favor it.

Hierarchies in and on a Set: Local and Global Orders and Pre-orders

An example of a *hierarchy in a finite set X* is displayed in Fig. 1. In such a hierarchy, there is a finite number of levels and there is no ambiguity in the assignment of a level to an element. The elements which appear eventually in the hierarchy are the *participants*; those which do not are *nonparticipants*; if all of X participates, then the hierarchy is on X.





Every *local order* (i. e. locally reflexive, transitive and antisymmetric relation) from a finite set to itself establishes a hierarchy in that set, that is, can be used as the 'precedes' relation in the hierarchy. Conversely, given any hierarchy, its '*__precedes__*' is a local order. The hierarchy is on X exactly when the local order is the global one.

The picture of the hierarchy is called its *Hasse diagram*. It can always be obtained from the digraph of the local-order relation by the suppression of loops and of those arrows which directly connect nodes between which there is also a longer path.

The formulas of Theorem 13 can be used for fast computational testing of the listed properties.

Theorem 13 *The following conditions universally characterize the transitivity, reflexivity and pre-order on* $R \in \mathcal{R}(X \rightsquigarrow X)$*:*

- R *is transitive if and only if* $R \sqsubseteq R \triangleright R^{-1}$.
- R *is reflexive if and only if* $R \triangleright R^{-1} \sqsubseteq R$.
- R *is a pre-order if and only if* $R = R \triangleright R^{-1}$.

More complex relational structures are investigated by theories of homomorphisms, which can be further generalized [6].

Definition 14 Let F, R, G, S be heterogeneous relations between the sets A, B, C, D such that $R \in \mathcal{R}(A \rightsquigarrow B)$. The conditions that (for all $a \in A$, $b \in B$, $c \in C$, $d \in D$) the expression $(aFc \wedge aRb \wedge bGd) \rightarrow cSd$ we denote by *FRG*:*S*. We say that *FRG*:*S* is *forward compatible*, or, equivalently, that F, G are *generalized morphisms*.

The following *Bandler–Kohout compatibility theorem* holds, [6]:

Theorem 15 (Generalized morphisms)

- *FRG : S are forward compatible if and only if* $F^{\mathsf{T}} \circ R \circ G \sqsubseteq S$.
- *Formulas for computing the explicit compatibility criteria for F and G are: FRG : S are forward-compatible if and only if* $F \sqsubseteq R \triangleleft (G \triangleleft S^{\mathsf{T}})$.

The R's of forward compatibility constitute a lower ideal. Similarly, the *backward compatibility* given by $F \circ S \circ G^{\mathsf{T}} \sqsubseteq R$ gives a generalized proteromorphism. It constitutes an upper ideal or filter: *FRG* : *S* are backward compatible if and only if $F \circ S \circ G^{\mathsf{T}} \sqsubseteq R$ if and only if $S \sqsubseteq F^{\mathsf{T}} \triangleleft R \triangleright G$.

FRG : *S* are *both-way compatible* if they are both forward and backward compatible. The conventional *homomorphism* is a special case of both-way compatibility, where F and G are not general relations but just many-to-one mappings.

The generalized morphisms of Bandler and Kohout [6] are relevant not only theoretically, but have also an important practical use in solving systems of inequalities and equations on systems of relations.

For partial homomorphisms the situation becomes more complicated. In partial structures the conventional homomorphism splits into mutually related *weak*, *strong* and *very strong* kinds of homomorphism [5].

Fuzzy Relations

Mathematical relations can contribute to investigation of properties of a large variety of structures in sciences and engineering. The power of *relational analysis* stems from the elegant algebraic structure of relational systems that is supplemented by the computational power of *relational matrix notation*. This power is further enhanced by many-valued logic based (fuzzy) extensions of the relational calculus.

As often in mathematics, where terms are used inclusively, the crisp (nonfuzzy) sets and relations are merely special cases of fuzzy sets and relations, in which the actual degrees happen to be the extreme ones. On the theoretical side, *fuzzy relations* are extensions of standard nonfuzzy (crisp) relations. By replacing the usual Boolean algebra by many-valued logic algebras, one obtains extensions that contain the classical relational theory as a special case.

Definitions

A *fuzzy set* is one to which any element may belong to various degrees, rather than either not at all (degree 0) or utterly (degree 1). Similarly, a *fuzzy relation* is one which may hold between two elements to any degree between 0 and 1 inclusive. The sentence $x_i R y_j$ takes its value $\delta\ (x_i\ R\ y_j) = R_{ij}$, from the interval [0, 1] of real numbers. In early papers on fuzzy relations $\mu_R(x_i, y_j)$ was usually written instead of R_{ij}.

The matrix notation used in the previous sections for nonfuzzy (crisp) relations is directly applicable to the fuzzy case. Thus, all the definitions of operations, compositions and products can be directly extended to the fuzzy case.

Operations and Inclusion on $\mathcal{R}_F\ (X \rightsquigarrow Y)$

Fuzzy Relations with Min, Max Connectives

This has been the most common extension of relations to the fuzzy realm. Boolean $\wedge$ and $\vee$ are replaced by many-valued connectives min, max in all crisp definitions.

In matrix terms, this yields the following intersection and union operations:

$$(R \sqcap S)_{ij} = \min(R_{ij}, S_{ij}),$$
$$(R \sqcup S)_{ij} = \max(R_{ij}, S_{ij}).$$

(In older μ-notation, $\mu_{(R \sqcap S)}(x_i, x_j) = \min(\mu_R(x_i, y_j), \mu_S(x_i, y_j))$, etc.)

The negation of R is given by $(\neg R)_{ij} = 1 - R_{ij}$. The converse of R is given by $(R^\intercal)_{ij} = R_{ji}$.

Fuzzy Relations Based on Łukasiewicz Connectives

When the *bold* (*Łukasiewicz*) connectives $x \vee y = \min(1, x + y)$, $x \wedge y = \max(0, x + y - 1)$ are used to define $\sqcup$, $\sqcap$ operations, this is an instance of relations in *MV-algebras*.

Fuzzy Relations With *t*-Norms and Co-Norms

Fuzzy logics can be further generalized. $\wedge$ and $\vee$ are obtained by replacing min and max by a *t-norm* and a *t-conorm*, respectively. A t-norm is an operation $* : [0, 1]^2 \to [0, 1]$ which is commutative, associative, nondecreasing in both arguments and having 1 as the unit element and 0 as the zero element. Taking a continuous t-norm, by *residuation* we obtain a *many-valued logic implication* $\to$. Using $\{ \wedge, \vee, *, \to \}$ one can define families of deductive systems for fuzzy logics called *BL-logics* [31]. In relational systems using BL-logics, one can define again various t-norm based relational properties [53,83], *BK-products* and *generalized morphisms of relations* [47].

Definition 16 (*Inclusion of relations*) A relation R is 'contained in' or is a *subrelation* of a relation S, written $R \sqsubseteq S$, if and only if $(\forall i)(\forall j)\ R_{ij} \leq S_{ij}$.

This definition guarantees that R is a subrelation of R' if and only if every R_α is a subrelation of its corresponding R_α. (This convenient meta-property is called *cutworthiness*, see Theorem 17 below.)

Products: $\mathcal{R}_F(X \rightsquigarrow Y) \times \mathcal{R}_F(Y \rightsquigarrow Z) \to \mathcal{R}_F(X \rightsquigarrow Z)$

For fuzzy relations, there are two versions of products: harsh and mean [3,52]. Most conveniently, again, in matrix terms *harsh products* syntactically correspond to matrix formulas for the crisp relations. The *fuzzy relational products* are obtained by replacing the Boolean logic connectives AND, OR, both implications and the equivalence of crisp products by connectives of some many-valued logic chosen according to the properties of the products required. Thus the $\circ$-product and $\Box$-product are given exactly as in Proposition 6 above by formulas 3) and 5), respectively; for triangle products as given in Proposition 8 above. For the MVL implication operators most often used to define *fuzzy triangle products*, see ► Checklist paradigm semantics for fuzzy logics, Table 1, or [8]. The details of choice of the appropriate many-valued connectives are discussed in [3,7,8,40,43,52].

Given the general formula $(R@S)_{ik} := \#\ (R_{ij} * S_{jk})$ for a relational product, a *mean product* is obtained by

Boolean and Fuzzy Relations, Table 1
Closures and an interior

1.	The locally reflexive closure of R: locref clo $R = R \sqcup E_R$.
2.	The symmetric closure of R: sym clo $R = R \sqcup R^{\mathsf{T}}$.
3.	*symmetric interior* of R: sym int $R = R \sqcap R^{\mathsf{T}}$.
4.	The transitive closure of R: tra clo $R = R \sqcap R^2 \sqcap \cdots = \sqcap_{k \in Z^+} R^k$.
5.	The *local tolerance closure* of R: loctol clo R = locref clo (sym clo R).
6.	The *local pre-order closure* of R: locpre clo R = locref clo (tra clo R) = tra clo (locref clo R).
7.	The *local equivalence closure* of R: locequ clo R = tra clo (sym clo (locref clo R)).
8.	The *reflexive closure* of R: ref clo $R = R \sqcup E_X$.
9.	The *tolerance closure* of R: tol clo R = ref clo (sym clo R).
10.	The *pre-order closure* of R: pre clo R = ref clo (tra clo R).
11.	The *equivalence closure* of R: equ clo R = tra clo (tol clo R).

replacing the outer connective # by $\sum$and normalizing the resulting product appropriately. In more concrete terms, in order to obtain the mean products, the outer connectives $\bigvee_j$ in $\circ$ and $\bigwedge_j$ in $\square$, $\triangleleft$, $\triangleright$ are replaced by $1/n) \sum_j$ [3].

N-ary Relations

An *n-ary relation* R is an open sentence with n slots; when these are filled in order by the names of elements from sets $X_1, \ldots, X_n$, there results a proposition that is either true or false if the relation is crisp, or is judged to hold to a certain degree if the relation is fuzzy. This 'intensional' definition is matched by the satisfaction set R_S of R, which is a fuzzy subset the n-tuple of $X_1, \ldots, X_n$, and can be used, if desired as its extensional definition. The matrix notation works equally well for n-ary relations and all the types of the BK-products are also defined. For details see [9].

Special Properties of Fuzzy Relations

The *special properties of crisp relations* can be generalized to fuzzy relations exactly as they stand in Definition 12, using in each case the second of the two given definitions. It is perhaps worthwhile spelling out the requirements for transitivity in more detail:

$$R^2 \sqsubseteq R \;\Leftrightarrow\; (\forall i, k) \max_i(\min R_{ij}, R_{jk})) \leq R_{ik}.$$

Useful references provide further pointers to the literature: general [43] on fuzzy partitions [14,69], fuzzy similarities [69], tolerances [34,75,85].

Alpha-cuts of Fuzzy Relations

It is often convenient to study fuzzy relations through their α-*cuts*; for any α in the half-open interval [0, 1], the α-cut of a fuzzy relation R is the crisp relation R_α given by

$$(R_\alpha)_{ij} = \begin{cases} 1 & \text{if } R_{ij} \geq \alpha, \\ 0 & \text{otherwise.} \end{cases}$$

Compatibility of families of crisp relations with their fuzzy counterpart (the original relation on which the α-cuts have been performed) is guaranteed by the following theorem on cutworthy properties [10]:

Theorem 17 *It is true of each simple property P (given in Definition 12) and every compound property P (listed in Table 1), that every α-cut of a fuzzy relation R possesses P in the crisp sense, if and only if R itself possesses in the fuzzy sense. (Such properties are called cutworthy.)*

Fuzzy Partitions, Fuzzy Clusters and Fuzzy Hierarchies

Via their α-cuts, fuzzy local and global equivalences provide precisely the nested families of partitions in and on a set which are required by the theory and for the applications in taxonomy envisaged in [17,18]. Fuzzy local and global tolerances similarly provide families of tolerance classes for the cluster type of classification which allows overlaps. Fuzzy local and global orders furnish nested families of hierarchies in and on a set, with their accompanying families of Hasse diagrams.

The importance of fuzzy extensions cannot be overestimated. Thus, one may identify approximate similarities in data, approximate equivalences and orders. Such approximations are paramount in many applications, in situations when only incomplete, partial information about the domain of scientific or technological application is available.

Closures and interiors of relations play an important role in design of fast fuzzy relational closure algorithms [4,9,10,11] for computing such approximations.

Theorem 17 and other theorems on commuting of cuts with closures [11,42] guarantee their correctness.

Closures and Interiors with Special Properties

For certain properties P which a fuzzy relation may have or lack, there always exists a well-defined *P-closure* of R, namely the least inclusive relation V which contains R and has the property P. Also, for some properties P, the P-interior of R is the most inclusive relation Q contained in R and possessing P. Clearly, where the P-closure exists, R itself possesses P if and only if R is equal to P-clo(R), and the same for interiors.

Certain closures use the local equality E_R of R, given by $(E_R)_{ii} = \max_i(\max(R_{ij}, R_{ji}))$, $(E_R)_{ij} = 0$ if $j \neq i$. Others use the equality on X given by $(E_X)_{ii} = 1$, $(E_X)_{ij} = 0$ for $j \neq i$.

Important closures and one important interior are given in Table 1. See [4,10] for further details.

Applications of Relational Methods in Engineering, Medicine and Science

Relational properties are important for obtaining knowledge about characteristics and interactions of various parts of a relational model used in real life applications. Identification of composite properties of mathematical relations, such as local or global pre-orders, orders, tolerances or equivalences, plays an important role in *evaluation of empirical data*, (e. g. medical data, commercial data etc. or data for technological forecasting) and building and evaluating *relational models* based on such data [48,49].

The local and global properties detect important semantic distinctions between various concepts captured by relational structures. For example the interactions between technological parts, processes etc., or relationships of *cognitive constructs* elicited experimentally [37,39,41,55]. Capturing both, local and global properties is important for distinguishing participants from nonparticipants in a relational structure. This distinction is crucial for obtaining a nondistorted picture of reality.

In the general terms, the abstract theoretical tools supporting identification and representation of relational properties are fuzzy closures and interiors [4,10]. Having such means for *testing relational properties* opens the avenue to linking the empirical structures that can be observed and captured by fuzzy relations with their abstract, symbolic representations that have well defined mathematical properties.

This opens many possibilities for computer experimentation with empirically identified logical, say, predicate structures. These techniques found practical use in directing *resolution based theorem prover* strategy [56], relation-based inference in *medical diagnosis* [48,58] and at extracting predicate structures of 'train of thought' from questionnaires presented to people by means of Kelly's *repertory grids*. BK-relational products and fast fuzzy relational algorithms based on fuzzy closures and interiors have been essential for computational progress of in this field and for *optimization of computational performance*. See the survey in [52] with a list of 50 selected references on the mathematical theory and applications of BK-products in various fields of science and engineering. Further extensions or modifications of BK-products have been suggested in [19,20,21,30].

Applications of relational theories, computations and modeling include the areas of medicine [48,59], psychology [49], cognitive studies [36,38], nuclear engineering [84], industrial engineering and management [25,46], architecture and urban studies [65,66] value analysis in business and manufacturing [60] information retrieval [51,54], computer security [45,50] databases, theoretical computer science [13,68,71], software engineering [78], automated reasoning [56], and logic [12,28,63]. Particularly important for software engineering is the contribution of C.A.R. Hoare and He Jifeng [33] who use the crisp triangle BK-superproduct for software specification, calling the crisp $\triangleleft$ products in fact *'weak prespecifications'*.

Relational equations [22] play an important role in applications [70] in general, and also in AI and applications of causal reasoning [24]; fuzzy inequalities in

mathematical programming [72]. Applications in game theory of crisp relations is well established [78,79].

Brief Review of Theoretical Development

Binary (two place) relations were first perceived in their abstract mathematical form by Galen of Pergamon in the 2nd century AD [57]. After a long gap, first systematic development of the calculus of relations (concerned with the study of logical operations on binary relations) was initiated by A. DeMorgan, C.S. Pierce and E. Schröder [9,64]. Significant investigation into the logic of relations was the 1900 paper of B. Russell [76] and axiomatization of the relational calculus in 1941 by A. Tarski [64,81]. Extensibility of Tarski's axioms to the fuzzy domain has been investigated by Kohout [44].

Later algebraic advances in relational calculus [9] stem jointly from the elegant work of J. Riguet (1948) [74], less widely known but important work of O. Borůvka (1939) [15,16,17,18] and the stimulus of fuzzy set theory of L.A. Zadeh (1965) [35,85,86], and include a sharpened perception of special properties and the construction of new kinds of relational products [3], together with the extension of the theory from Boolean to multiple-valued logic based relations [2,9]. The triangle subproduct $R \triangleleft S$, the triangle superproduct $R \triangleright S$, and square product $R \square S$ were introduced in their general form defined below by Bandler and Kohout in 1977, and are referred to as the *BK-products* in the literature [19,20,30]. The square product, however, stems from Riguet (1948) [74], needing only to be made explicit [1,9]. E. Sanchez independently defined an α-compostition [77] which is in fact $\triangleleft$ using Heyting–Gödel implication. The special instances of the triangle BK-products were more recently rediscovered and described in 1986 by J.P. Doignon, B. Monjardet, M. Roubens, and P. Vincke [23,26] calling these *'traces of relations'*. Hence, a 'trace-of-relation' is a BK-triangle superproduct in which $\rightarrow$ is the residuum of a commutative $\wedge$. The crisp square product was also independently introduced in 1986 by R. Berghammer, G. Schmidt and H. Zierer [13] as a generalization of Riguet's 'noyau' [74].

On the other hand, advances in abstract relational algebras stems from the work of Tarski [81] and his school [32,64,67,82]. Tarski's axiomatization [81] of homogeneous relational calculus takes relations and operations over relations as the primitives. It applies only to homogeneous relations as it has only one constant entity, the identity relation E. For heterogeneous relations, taking e. g. U_{XY} as the universal relation we have a finite number of separate identity relations (constants) i. e. E_{XY}, E_{YZ}, …, etc. [4,10]. Therefore viewed syntactically through the logic axioms, the axiomatization of heterogeneous relations (containing a whole family of universal relations) would be a many-sorted theory [30], each universal relation belonging to a different sort.

Tarski's axioms of homogeneous relations

$$
\begin{array}{l}
R \circ E = E \circ R;\ (R \circ S)^{\mathsf{T}} = S^{\mathsf{T}} \circ R^{\mathsf{T}};\\
(R^{\mathsf{T}})^{\mathsf{T}} = R;\ (\neg R)^{\mathsf{T}} = \neg (R^{\mathsf{T}});\\
(R \sqcup S)^{\mathsf{T}} = R^{\mathsf{T}} \sqcup S^{\mathsf{T}};\\
(R \circ S) \circ T = R \circ (S \circ T);\\
(R \sqcup S) \circ T = (R \circ T) \sqcup (S \circ T);\\
R \circ (S \sqcup T) = (R \circ S) \sqcup (R \circ T);\\
(R^{\mathsf{T}} \circ \neg (R \circ S)) \sqcup \neg S = \neg S.
\end{array}
$$

Taking the axioms on their own opens the way to abstract relational algebras (RA) with new problems at hand. Tarski and his school have investigated the interrelationship of various generalizations of associative RAs in a purely abstract way. In some of these generalizations, the axiom of associativity for relational composition is dropped. This leads from representable (RRA) to semi-associative (SA), weakly associative (WA) and nonassociative (NA) relational algebras. In 1982 R.D. Maddux [62] gave the following result:

$$\text{RRA} \subset \text{RA} \subset \text{SA} \subset \text{WA} \subset \text{NA}.$$

All these generalizations deal only with one relational composition. The equations for pseudo-associativities given above (Proposition 9) and the *nonassociativity* of the square product (Proposition 4) show that there exist nonassociative representations of relational algebras (RA) in the relational calculus. Theorem 15 and Proposition 8 show that the interplay of several relational compositions is essentially involved in the computationally more powerful formulas of the relational calculus. The Tarskian RA axiomatizations, however, do not express fully the richness of the calculus of binary relations and the mutual interplay of associative $\circ$, pseudo-associative $\triangleright$, $\triangleleft$ and nonassociative $\square$ prod-

ucts. Considerable scope for further research into new axiomatizations still remains. Our results based on nonassociative BK-products of Bandler and Kohout that historically precede abstract nonassociative generalizations in relational algebras of Maddux show that the nonassociative products have representations and that these representations offer various computational advantages. There is also a link of RA with projective geometries [61].

Basic Books and Bibliographies

The best general books on theory of crisp relations and applications are [78] and [80]. In fuzzy field, there is no general book available at present. There are, however, extant some more specialized monographs: on solving fuzzy relations equations [27], on preference modeling and multicriteria decision making [39], on representation of cognitive maps by relations [39] and on crisp and fuzzy BK-products of relations [53]. One can also find some specialized monographs on logic foundations and relational algebras: [32,82]. All these books also contain important list of references. The most important bibliography of selected references on the topic related to *fuzzy sets* and relations is contained in [43]. The early years of fuzzy sets (1965–1975) are covered very comprehensively in the critical survey and annotated bibliography [29]. Many-valued logic connectives form an important foundation for fuzzy sets and relations. The book of N. Rescher [73] still remains the best comprehensive survey that is also accessible to a nonlogician. It contains almost complete bibliography of many-valued logics from the end of the 19th century to 1968.

See also

- ► Alternative Set Theory
- ► Checklist Paradigm Semantics for Fuzzy Logics
- ► Finite Complete Systems of Many-valued Logic Algebras
- ► Inference of Monotone Boolean Functions
- ► Optimization in Boolean Classification Problems
- ► Optimization in Classifying Text Documents

References

1. Bandler W, Kohout LJ (1977) Mathematical relations, their products and generalized morphisms. Techn Report Man-Machine Systems Lab Dept Electrical Engin Univ Essex, Colchester, Essex, UK, EES-MMS-REL 77-3. Reprinted as Chap. 2 of Kohout LJ, Bandler W (eds) Survey of Fuzzy and Crisp Relations, Lect Notes in Fuzzy Mathematics and Computer Sci, Creighton Univ Omaha
2. Bandler W, Kohout LJ (1980) Fuzzy relational products as a tool for analysis and synthesis of the behaviour of complex natural and artificial systems. In: Wang PP, Chang SK (eds) Fuzzy Sets: Theory and Appl. to Policy Analysis and Information Systems. Plenum, New York, pp 341–367
3. Bandler W, Kohout LJ (1980 1981) Semantics of implication operators and fuzzy relational products. Internat J Man-Machine Studies 12:89–116 Reprinted in: In: Mamdani EH, Gaines BR (eds) Fuzzy Reasoning and its Applications. Acad. Press, New York, 219–246
4. Bandler W, Kohout LJ (1982) Fast fuzzy relational algorithms. In: Ballester A, Cardús D, Trillas E (eds) Proc. Second Internat. Conf. Math. at the Service of Man (Las Palmas, Canary Islands, Spain, 28 June-3 July), Univ. Politechnica de Las Palmas, pp 123–131
5. Bandler W, Kohout LJ (1986) On new types of homomorphisms and congruences for partial algebraic structures and n-ary relations. Internat J General Syst 12:149–157
6. Bandler W, Kohout LJ (1986) On the general theory of relational morphisms. Internat J General Syst 13:47–66
7. Bandler W, Kohout LJ (1986) A survey of fuzzy relational products in their applicability to medicine and clinical psychology. In: Kohout LJ, Bandler W (eds) Knowledge Representation in Medicine and Clinical Behavioural Sci. Abacus Book. Gordon and Breach, New York, pp 107–118
8. Bandler W, Kohout LJ (1987) Fuzzy implication operators. In: Singh MG (ed) Systems and Control Encyclopedia. Pergamon, Oxford, pp 1806–1810
9. Bandler W, Kohout LJ (1987) Relations, mathematical. In: Singh MG (ed) Systems and Control Encyclopedia. Pergamon, Oxford, pp 4000–4008
10. Bandler W, Kohout LJ (1988) Special properties, closures and interiors of crisp and fuzzy relations. Fuzzy Sets and Systems 26(3) (June):317–332
11. Bandler W, Kohout LJ (1993) Cuts commute with closures. In: Lowen B, Roubens M (eds) Fuzzy Logic: State of the Art. Kluwer, Dordrecht, pp 161–167
12. Benthem J van (1994) General dynamic logic. In: Gabbay DM (ed) What is a Logical System. Oxford Univ. Press, Oxford pp 107–139
13. Berghammer R, Schmidt G (1989/90) Symmetric quotients and domain constructions. Inform Process Lett 33:163–168
14. Bezdek JC, Harris JD (1979) Convex decompositions of fuzzy partitions. J Math Anal Appl 67:490–512
15. Borůvka O (1939) Teorie grupoidu (Gruppoidtheorie, I. Teil). Publ Fac Sci Univ Masaryk, Brno, Czechoslovakia 275:1–17, In Czech, German summary
16. Borůvka O (1941) Über Ketten von Faktoroiden. MATH-A 118:41–64

17. Borůvka O (1945) Théorie des décompositions dans un ensemble. Publ Fac Sci Univ Masaryk, Brno, Czechoslovakia:278 1–37 (In Czech, French summary)
18. Borůvka O (1974) Foundations of the theory of groupoids and groups. VEB Deutsch. Verlag Wissenschaft., Berlin, Also published as Halsted Press book by Wiley, 1976
19. DeBaets B, Kerre E (1993) Fuzzy relational compositions. Fuzzy Sets and Systems 60(1):109–120
20. DeBaets B, Kerre E (1993) A revision of Bandler–Kohout composition of relations. Math Pannonica 4:59–78
21. DeBaets B, Kerre E (1994) The cutting of compositions. Fuzzy Sets and Systems 62(3):295–310
22. DiNola A, Pedrycz W, Sanchez E (1989) Fuzzy relation equations and their applications to knowledge engioneering. Kluwer, Dordrecht
23. Doignon JP, Monjardet B, Roubens M, Vincke P (1986) Biorders families, valued relations and preference modelling. J Math Psych 30:435–480
24. Dubois D, Prade H (1995) Fuzzy relation equations and causal reasoning. Fuzzy Sets and Systems 75(2):119–134
25. Dubrosky B, Kohout LJ, Walker RM, Kim E, Wang HP (1997) Use of fuzzy relations for advanced technological cost modeling and affordability decisions. In: 35th AIAA Aerospace Sci. Meeting and Exhibit (Reno, Nevada, January 6-9, 1997), Amer. Inst. Aeronautics and Astronautics, Reston, VA, 1–12, Paper AIAA 97-0079
26. Fodor JC (1992) Traces of fuzzy binary relations. Fuzzy Sets and Systems 50(3):331–341
27. Fodor J, Roubens M (1994) Fuzzy preference modelling and multicriteria decision support. Kluwer, Dordrecht
28. Gabbay DM (1994) What is a logical system? In: Gabbay DM (ed) What is a Logical System? Oxford Univ. Press, Oxford, pp 179–216
29. Gaines BR, Kohout LJ (1977) The fuzzy decade: A bibliography of fuzzy systems and closely related topics. Internat J Man-Machine Studies 9:1–68 (A critical survey with bibliography.) Reprinted in: Gupta MM, Saridis GN, Gaines BR (eds) (1988) Fuzzy Automata and Decision Processes. Elsevier/North-Holland, Amsterdam, pp 403–490
30. Hájek P (1996) A remark on Bandler–Kohout products of relations. Internat J General Syst 25(2):165–166
31. Hájek P (1998) Metamathematics of fuzzy logic. Kluwer, Dordrecht
32. Henkin L, Monk JD, Tarski A (1985) Cylindric algebras, vol II. North-Holland, Amsterdam
33. Hoare JAR, Jifeng He (1986) The weakest prespecification I-II. Fundam Inform 9:51–84; 217–251
34. Höhle U (1988) Quotients with respect to similarity relations. Fuzzy Sets and Systems 27(1):31–44
35. Höhle U, Klement EP (1995) Non-classical logics and their applications to fuzzy subsets: A handbook of mathematical foundations of fuzzy sets. Kluwer, Dordrecht
36. Juliano BA (1993) A fuzzy logic approach to cognitive diagnosis. PhD Thesis, Dept. Comput. Sci., Florida State Univ., Tallahassee, Fl
37. Juliano BA (1996) Towards a meaningful fuzzy analysis of urbanistic data. Inform Sci 94(1–4):191–212
38. Juliano BA, Bandler W (1989) A theoretical framework for modeling chains-of-thought: Automating fault detection and error diagnosis in scientific problem solving. In: Fishman MB (ed) Proc. Second Florida Artificial Intelligence Res. Symp., Florida AI Res. Soc., FLAIRS, pp 118–122
39. Juliano B, Bandler W (1996) Tracing chains-of-thought: Fuzzy methods in cognitive diagnosis. Physica Verlag, Heidelberg
40. Kandel A (1986) Fuzzy mathematical techniques with applications. Addison-Wesley, Reading, MA
41. Kim E, Kohout LJ, Dubrosky B, Bandler W (1996) Use of fuzzy relations for affordability decisions in high technology. In: Adey RA, Rzevski G, Sunol AK (eds) Applications of Artificial Intelligence in Engineering XI. Computational Mechanics Publ., Billerica, MA
42. Kitainik L (1992) For closeable and cutworthy properties, closures always commute with cuts. In: Proc. IEEE Internat. Conf. Fuzzy Systems, IEEE, New York, pp 703–704
43. Klir GJ, Yuan B (1995) Fuzzy sets and fuzzy logic: Theory and applications. Prentice-Hall, Englewood Cliffs, NJ
44. Kohout LJ (2000) Extension of Tarski's axioms of relations to t-norm fuzzy logics. In: Wang PP (ed) Proc. 5th Joint Conf. Information Sciences, I Assoc. Intelligent Machinery, Durham44–47
45. Kohout LJ (1990) A perspective on intelligent systems: A framework for analysis and design. Chapman and Hall and v. Nostrand, London–New York
46. Kohout LJ (1997) Fuzzy relations and their products. In: Wang P (ed) Proc. 3rd Joint Conf. Inform. Sci. JCIS'97, Duke Univ., March,), Keynote Speech VIII: Prof. W. Bandler Memorial Lecture; to appear in: Inform. Sci.
47. Kohout LJ (1998 1999) Generalized morphisms in BL-logics. In: Logic Colloquium: The 1998 ASL Europ. Summer Meeting, Prague, August 9-15 1998, Assoc. Symbolic Logic (Extended abstract presenting the main mathematical theorems). Reprinted in: Bull Symbolic Logic (1999) 5(1):116–117
48. Kohout LJ, Anderson J, Bandler W, et al. (1992) Knowledge-based systems for multiple environments. Ashgate Publ. (Gower), Aldershot, Hampshire, UK
49. Kohout LJ, Bandler W (eds) (1986) Knowledge representation in medicine and clinical behavioural science. Abacus Book. Gordon and Breach, New York
50. Kohout LJ, Bandler W (1987) Computer security systems: Fuzzy logics. In: Singh MG (ed) Systems and Control Encyclopedia. Pergamon, Oxford
51. Kohout LJ, Bandler W (1987) The use of fuzzy information retrieval techniques in construction of multi-centre knowledge-based systems. In: Bouchon B, Yager RR (eds) Uncertainty in Knowledge-Based Systems. Lecture Notes Computer Sci. Springer, Berlin, pp 257–264
52. Kohout LJ, Bandler W (1992) Fuzzy relational products in knowledge engineering. In: Novák V, et al (eds) Fuzzy Ap-

proach to Reasoning and Decision Making. Academia and Univ. Press, Prague, pp 51–66
53. Kohout LJ, Bandler W (1999) A survey of fuzzy and crisp relations. Lecture Notes Fuzzy Math and Computer Sci Creighton Univ., Omaha, NE
54. Kohout LJ, Keravnou E, Bandler W (1984) Automatic documentary information retrieval by means of fuzzy relational products. In: Zadeh LA, Gaines BR, Zimmermann H-J (eds) Fuzzy Sets in Decision Analysis. North-Holland, Amsterdam, pp 383–404
55. Kohout LJ, Kim E (1997) The role of semiotic descriptors in relational representation of fuzzy granular structures. In: Albus J (ed) ISAS '97 Intelligent Systems and Semiotics: A Learning Perspective. Special Publ. Nat. Inst. Standards and Techn., US Dept. Commerce, Washington, DC, pp 31–36
56. Kohout LJ, Kim Yong-Gi (1993) Generating control strategies for resolution-based theorem provers by means of fuzzy relational products and relational closures. In: Lowen B, Roubens M (eds) Fuzzy Logic: State of the Art. Kluwer, Dordrecht, pp 181–192
57. Kohout LJ, Stabile I (1992) Logic of relations of Galen. In: Svoboda V (ed) Proc. Internat. Symp. Logica '92, Inst. Philosophy Acad. Sci. Czech Republic, Prague, pp 144–158
58. Kohout LJ, Stabile I, Bandler W, Anderson J (1995) CLINAID: Medical knowledge-based system based on fuzzy relational structures. In: Cohen M, Hudson D (eds) Comparative Approaches in Medical Reasoning. World Sci., Singapore, pp 1–25
59. Kohout LJ, Stabile I, Kalantar H, San-Andres M, Anderson J (1995) Parallel interval-based reasoning in medical knowledge-based system Clinaid. Reliable Computing (A special issue on parallel systems) 1(2):109–140
60. Kohout LJ, Zenz G (1997) Activity structures and triangle BK-products of fuzzy relations – a useful modelling and computational tool in value analysis studies. In: Mesiar R, et al (eds) Proc. IFSA 1997 (The World Congress of Internat. Fuzzy Systems Assoc., Prague), IV, Academia, Prague, pp 211–216
61. Lyndon RC (1961) Relation algebras and projective geometries. Michigan Math J 8:21–28
62. Maddux RD (1982) Some varieties containing relation algebras. Trans Amer Math Soc 272(2):501–526
63. Maddux RD (1983) A sequent calculus for relation algebras. Ann Pure Appl Logic 25:73–101
64. Maddux RD (1991) The origin of relation algebras in the development and axiomatization of the calculus of relations. Studia Logica 50(3–4):421–455
65. Mancini V, Bandler W (1988) Congruence of structures in urban knowledge representation. In: Bouchon B, Saita L, Yager R (eds) Uncertainty and Intelligent Systems. Lecture Notes Computer Sci. Springer, Berlin, pp 219–225
66. Mancini V, Bandler W (1992) Design for designing: Fuzzy relational environmental design assistant (FREDA). In: Kandel A (ed) Fuzzy Expert Systems. Addison-Wesley, Reading, MA, pp 195–202
67. McKinsey JCC (1940) Postulates for the calculus of binary relations. J Symbolic Logic 5:85–97
68. Nemeti I (1991) Algebraization of quantifier logics, an introductory overview. Studia Logica 50(3–4):485–569
69. Ovchinikov S (1991) Similarity relations, fuzzy partitions, and fuzzy ordering. Fuzzy Sets and Systems 40(1):107–126
70. Pedrycz W (1991) Processing in relational structures: Fuzzy relational equations. Fuzzy Sets and Systems 40(1): 77–106
71. Pratt V (1991) Dynamic algebras: examples, constructions, applications. Studia Logica 50(3–4):571–605
72. Ramik J, Rommelfanger H (1996) Fuzzy mathematical programming based on some new inequality relations. Fuzzy Sets and Systems 81(1):77–87
73. Rescher N (1969) Many-valued logic. McGraw-Hill, New York
74. Riguet J (1948) Relations binaires, fermetures, correspondences de Galois. Bull Soc Math France 76:114–155
75. Rundensteiner E, Bandler W, Kohout L, Hawkes LW (1987) An investigation of fuzzy nearness measure. In: Proc. Second IFSA Congress, Internat. Fuzzy Systems Assoc., pp 362–365
76. Russell B (1900/1) The logic of relations: with some applications to the theory of series. Rivisita di Mat 7:115–148, English translation (revised by Lord Russell) in: Marsh RC (ed) (1956) Logic and Knowledge – Essays 1901-1950. Allen–Unwin, London, 1–38 (in French)
77. Sanchez E (1988) Solutions in composite fuzzy relation equations. In: Gupta MM, Saridis GN, Gaines BR (eds) Fuzzy Automata and Decision Processes. Elsevier and Univ. Press, Amsterdam, pp 221–234
78. Schmidt G, Ströhlein T (1993) Relations and graphs: Discrete mathematics for computer scientists. Springer, Berlin
79. Schmidt G, Ströohlein T (1985) On kernels of graphs and solutions of games: A synopsis based on relations and fixpoints. SIAM J Alg Discrete Meth 6:54–65
80. Schreider JuA (1975) Equality, resemblance, and order. MIR, Moscow
81. Tarski A (1941) Calculus of relations. J Symbolic Logic 6(3): 73–89
82. Tarski A, Givant S (1987) A formalization of set theory without variables. Colloq Publ, vol 41. Amer. Math. Soc., Providence, RI
83. Valverde L (1985) On the structure of F-indistinguishability operators. Fuzzy Sets and Systems 17:313–328
84. Walle B Van der, DeBaets B, Kerre EE (1995) Fuzzy multicriteria analysis of cutting techniques in a nuclear reactor dismantling project. Fuzzy Sets and Systems 74(1):115–126
85. Zadeh LA (1987) Fuzzy sets: Selected papers I. In: Yager R et al (eds) Wiley, New York
86. Zadeh LA (1996) Fuzzy sets: Selected papers II. In: Klir G, Yuan B (eds) World Sci., Singapore

Bottleneck Steiner Tree Problems

BSTP

ALEXANDER ZELIKOVSKY
Georgia State University, Atlanta, USA

MSC2000: 05C05, 05C85, 68Q25, 90B80

Article Outline

Keywords
See also
References

Keywords

Bottleneck Steiner trees; Facility location; Geometric algorithms; Minmax multicenter; Approximation algorithms

A *bottleneck Steiner tree* (or a *min-max Steiner tree*) is a Steiner tree (cf. ► Steiner tree problems) in which the maximum edge weight is minimized. Several *multifacility location* and *VLSI routing* problems ask for bottleneck Steiner trees.

Consider the problem of choosing locations for a number of hospitals serving homes where the goal is to minimize maximum weighted distance to any home from the hospital that serves it and between hospitals. The solution is a tree which spans all hospitals and connects each home to the closest hospital. This tree can be seen as a Steiner tree where the homes are terminals and hospitals are Steiner points (cf. ► Steiner tree problems). Unlike the classical Steiner tree problem where the total length of Steiner tree is minimized, in this problem it is necessary to minimize maximum edge weight.

The other instance of the bottleneck Steiner tree problem occurs in electronic physical design automation where nets are routed subject to delay minimization [2,3]. The terminals of a net are interconnected possibly through intermediate nodes (Steiner points) and for electrical reasons one would like to minimize maximum distance between each pair of interconnected points.

The most popular versions of the bottleneck Steiner tree problem in the literature are geometric. Note that if the number of Steiner points is not bounded, then any edge can be subdivided into infinitely small segments and the resulting maximum edge length becomes zero. Therefore, any meaningful formulation should bound the number of Steiner points. One such formulation is suggested in [9].

Problem 1 Given a set of n points in the plane (called terminals), find a bottleneck Steiner tree spanning all terminals such that degree of any Steiner point is at least 3.

Instead of introducing constraints, one can minimize the number of Steiner points. The following formulation has been proved to be *NP*-hard [15] and approximation algorithms have been suggested in [11,14].

Problem 2 Given a set of n terminals in the plane and $\lambda > 0$, find a Steiner tree spanning n terminals with the minimum number of Steiner points such that every edge is not longer than λ.

Sometimes the bottleneck Steiner tree has predefined *topology*, i. e. the unweighted tree consisting of edges between terminals and Steiner points [4,5,10]. Then it is necessary to find the optimal positions of all Steiner points. Since the number of different topologies for a given set of terminals grows exponentially, fixing the topology greatly reduces the complexity of the bottleneck Steiner tree problem.

Problem 3 Find a bottleneck Steiner tree with a given topology T which spans a set of n terminals in the plane.

The first algorithms for the Euclidean case of Problem 3 are based on nonlinear optimization [7] and [13]. For a given $\lambda > 0$, the algorithm from [15] finds whether a Steiner tree ST with the maximum edge weight λ exists as follows.

The topology T is first transformed into a forest by removing edges between terminals, if any such edge has length more than λ, then ST does not exist. Each connected component T is processed separately. The following regions are computed in bottom-up fashion:

i) the region of the plane $R(s)$ where a Steiner point s can be placed; and
ii) the region $R^+(s)$ where the Steiner point adjacent to s can be placed which is the area within distance at most λ from $R(s)$.

If a Steiner point p is adjacent to nodes $s_1, \ldots, s_k$ in T_i, then $R(s) = R^+(s_1) \cap \cdots \cap R^+(s_k)$. The number $a(s)$

of arcs bounding $R(s)$ may be as high as the number of leaves in T_i. In order to keep this number low, the tree K can be decomposed in $O(\log n)$ levels such that in total there will be only $O(n)$ arcs in all regions. Thus the runtime of the algorithm is $O(n \log n)$ [15].

When the distance between points is rectilinear, several efficient algorithms are suggested for Problem 3 [4,9,10]. The algorithm above can be adjusted for the rectilinear plane: the regions $R(s)$ are rectangles. The fastest known algorithm solves Problem 3 in time $O(n^2)$ [9].

Each bottleneck Steiner problems can be generalized to arbitrary weights on edges and formulated for weighted graphs [6].

Problem 4 Given a graph $G = (V, E, w)$ with nonnegative weight w on edges, and a set of terminals $S \subset V$, find a Steiner tree spanning S with the smallest maximum edge weight.

Problem 4 can be solved efficiently in the optimal time $O(|E|)$ time [6]. Unfortunately, the above formulation does not bound the number of Steiner points. To bound the number of Steiner points it is necessary to take in account that unlike the classical Steiner tree problem in graphs (cf. ► Steiner tree problems), an edge cannot be replaced with a shortest path without affecting the bottleneck objective. The following graph-theoretical generalization of Problem 1 considered in [1,9] has been proved to be *NP*-hard.

Problem 5 Given a complete graph $G = (V, E, w)$ with nonnegative weight w on edges, and a set of terminals $S \subset V$, find a Steiner tree spanning S with the smallest maximum edge weight such that each Steiner point has degree at least 3.

Similarly to the classical Steiner tree problem, if no Steiner points are allowed, the minimum spanning tree (cf. also ► Capacitated minimum spanning trees) is the optimal solution for Problems 1 and 5. Therefore, similarly to the *Steiner ratio*, it is valid to consider the *bottleneck Steiner ratio* $\rho_B(n)$. The bottleneck Steiner ratio is defined as the supremum over all instances with n terminals of the ratio of the maximum edge weight of the minimum spanning tree over the maximum edge weight of the bottleneck Steiner tree. It has been proved that $\rho_B(n) = 2 \lfloor \log_2 n \rfloor - \delta$, where δ is either 0 or 1 depending on whether mantissa of $\log_2 n$ is greater than $\log_2 3/2$ [9].

The approximation complexity of the Problem 5 is higher than for the classical Steiner tree problem: even $(2 - \epsilon)$-approximation is *NP*-hard for any $\epsilon > 0$ [1]. On the other hand, the best known approximation algorithm for Problem 5 has *approximation ratio* $\log_2 n$ [1]. The algorithm looks for an approximate bottleneck Steiner tree in the collection C of edges between all pairs of terminals and minimum bottleneck Steiner trees for all triples of terminals. Using Lovasz' algorithm [12] it is possible to find out whether such a collection contains a valid Steiner tree, i. e. a Steiner tree with all Steiner points of degree at least three. The algorithm finds the smallest λ such that C still contains valid Steiner tree if all edges of weight more than λ are removed. It has been shown that $\lambda \leq M \cdot \log_2 n$, where M is the maximum edge weight of the optimal bottleneck Steiner tree.

See also

- ► Capacitated Minimum Spanning Trees
- ► Directed Tree Networks
- ► Minimax Game Tree Searching
- ► Shortest Path Tree Algorithms
- ► Steiner Tree Problems

References

1. Berman P, Zelikovsky A (2000) On the approximation of power-p and bottleneck Steiner trees. In: Adv. in Steiner Trees. Kluwer, Dordrecht, pp 117–135
2. Boese KD, Kahng AB, McCoy BA, Robins G (1995) Near-optimal critical sink routing tree constructions. IEEE Trans Computer-Aided Design Integr Circuits and Syst 14:1417–11436
3. Chiang C, Sarrafzadeh M, Wong CK (1990) Global routing based on Steiner min-max trees. IEEE Trans Computer-Aided Design Integr Circuits and Syst 9:1318–1325
4. Dearing PM, Francis RL (1974) A network flow solution to a multifacility location problem involving rectilinear distances. Transport Sci 8:126–141
5. Drezner Z, Wesolowsky GO (1978) A new method for the multifacility minimax location problem. J Oper Res Soc 29:1095–1101
6. Duin CW, Volgenant A (1997) The partial sum criterion for Steiner trees in graphs and shortest paths. Europ J Oper Res 97:172–182
7. Elzinga J, Hearn D, Randolph WD (1976) Minimax multifacility location with Euclidean distances. Transport Sci 10:321–336

8. Erkut E, Francis RL, Tamir A (1992) Distance-constrained multifacility minimax location problems on tree networks. Networks 22(1):37–54
9. Ganley JL, Salowe JS (1996) Optimal and approximate bottleneck Steiner trees. Oper Res Lett 19:217–224
10. Ichimori T (1996) A shortest pathe approach to a multifacility minimax location problem with rectilinear distances. J Res Soc Japan 19:217–224
11. Lin G-H, Hue G (1999) Steiner tree problem with minimum number of Steiner points and bounded edge-length. Inform Process Lett 69:53–57
12. Lovasz L, Plummer MD (1986) Matching theory. Elsevier, Amsterdam
13. Love RF, Weselowsky GO, Kraemer SA (1997) A multifacility minimax location method for Euclidean distances. Internat J Production Res 97:172–182
14. Mandoiu II, Zelikovsky AZ (2000) A note on the MST heuristic for bounded edge-length Steiner tress with minimum number of Steiner points. Inform Process Lett 75:165–167
15. Sarrafzadeh M, Wong CK (1992) Bottleneck Steiner trees in the plane. IEEE Trans Comput 41:370–374

Boundary Condition Iteration BCI

Rein Luus
Department Chemical Engineering,
University Toronto, Toronto, Canada

MSC2000: 93-XX

Article Outline

Keywords
Illustration of the Boundary Condition Iteration Procedure
Sensitivity Information Without Evaluating the Transition Matrix
See also
References

Keywords

Optimal control; Boundary condition iteration; BCI; Control vector iteration; Pontryagin's maximum principle; Iterative dynamic programming; IDP

In solving *optimal control* problems involving nonlinear differential equations, some iterative procedure must be used to obtain the optimal control policy. From *Pontryagin's maximum principle* it is known that the minimum of the performance index corresponds to the minimum of the Hamiltonian. Obtaining the minimum value for the Hamiltonian usually involves some iterative procedure. Here we outline a procedure that uses the necessary condition for optimality, but the boundary conditions are relaxed. In essence we have the optimal control policy at each iteration to a wrong problem. Iterations are performed, so that in the limit the boundary conditions, as specified for the optimal control problem, are satisfied. Such a procedure is called *approximation to the problem* or *boundary condition iteration method* (BCI). Many papers have been written about the method. As was pointed out in [1], the method is fundamentally very simple and computationally attractive for some optimal control problems. In [3] some evaluations and comparisons of different approaches were carried out, but the conclusions were not very definitive [5]. Although for *control vector iteration* (CVI) many papers are written to describe and evaluate different approaches with widely different optimal control problems, see for example [14], for BCI such comparisons are much more limited and there is sometimes the feeling that the method works well only if the answer is already known. However, BCI is a useful procedure for determining the optimal control policy for many problems, and it is unwise to dispatch it prematurely.

To illustrate the boundary condition iteration procedure, let us consider the *optimal control problem*, where the system is described by the differential equation

$$\|x\|\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x},\mathbf{u}), \quad \text{with } \mathbf{x}(0) \text{ given}, \tag{1}$$

where $\mathbf{x}$ is an n-dimensional state vector and $\mathbf{u}$ is an r-dimensional control vector. The optimal control problem is to determine the control $\mathbf{u}$ in the time interval $0 \le t < t_f$, so that the performance index

$$I = \int_0^{t_f} \psi(\mathbf{x},\mathbf{u})\,\mathrm{d}t \tag{2}$$

is minimized. We consider the case where the final time t_f is given and there are no constraints on the control or the state variables. According to Pontryagin's maximum principle, the minimum value of the performance index in (2) is obtained by minimizing the *Hamiltonian*

$$H = \psi + \mathbf{z}^{\top}\mathbf{f}\,. \tag{3}$$

The adjoint variable $\mathbf{z}$ is defined by

$$\frac{d\mathbf{z}}{dt} = -\frac{\partial H}{\partial \mathbf{x}}, \quad \text{with } \mathbf{z}(t_f) = \mathbf{0}, \tag{4}$$

which may be written as

$$\frac{d\mathbf{z}}{dt} = -\frac{\partial \mathbf{f}^\top}{\partial \mathbf{x}}\mathbf{z} - \frac{\partial \psi}{\partial \mathbf{x}}, \quad \text{with } \mathbf{z}(t_f) = \mathbf{0}. \tag{5}$$

The necessary condition for the minimum of the Hamiltonian is

$$\frac{\partial H}{\partial \mathbf{u}} = \mathbf{0}. \tag{6}$$

Let us assume that (6) can be solved explicitly for the control vector

$$\mathbf{u} = \mathbf{g}(\mathbf{x}, \mathbf{z}). \tag{7}$$

If we now substitute (7) into (1) and (5), and integrate these equations simultaneously backward from $t = t_f$ to $t = 0$ with some value assumed for $\mathbf{x}(t_f)$, we have the optimal control policy for a wrong problem, because there is no assurance that upon backward integration the given value of the initial state $\mathbf{x}(0)$ will be obtained. Therefore it is necessary to adjust the guessed value for the final state, until finally an appropriate value for $\mathbf{x}(t_f)$ is found. For this reason the method is called the boundary condition iteration method (BCI).

In order to find how to adjust the final value of the state, based on the deviation obtained from the given initial state, we need to find the mathematical relationship to establish the effect of the change in the final state on the change in initial state. Many papers have been written in this area. The development of the necessary *sensitivity* equations is presented very nicely in [1]. In essence, the sensitivity information can be obtained by getting the *transition matrix* for the linearized state equation. Linearization of (1) gives

$$\frac{d\delta\mathbf{x}}{dt} = \left(\frac{\partial \mathbf{f}^\top}{\partial \mathbf{x}}\right)^\top \delta\mathbf{x} + \left(\frac{\partial \mathbf{f}^\top}{\partial \mathbf{u}}\right)^\top \delta\mathbf{u}. \tag{8}$$

The transition matrix Φ is thus obtained from solving

$$\frac{d\Phi}{dt} = \left(\frac{\partial \mathbf{f}^\top}{\partial \mathbf{x}}\right)^\top \Phi, \quad \text{with } \Phi(t_f) = \mathbf{I}, \tag{9}$$

where $\mathbf{I}$ is the $(n \times n)$ identity matrix.

Suppose at iteration j the use of $\mathbf{x}^{(j)}(t_f)$ gives the initial state $\mathbf{x}^{(j)}(0)$ which is different from the given initial state $\mathbf{x}(0)$. Then a new choice will be made at iteration $(j+1)$ through the use of

$$\mathbf{x}^{(j+1)}(t_f) = \mathbf{x}^{(j)}(t_f) + \epsilon\Phi(0)(\mathbf{x}^{(j)}(0) - \mathbf{x}(0)), \tag{10}$$

where a stabilizing parameter ϵ is introduced to avoid overstepping. A convenient way of measuring the deviation from the given initial state is to define the error as the Euclidean norm

$$e^{(j)} = \left\| \mathbf{x}^{(j)}(0) - \mathbf{x}(0) \right\|. \tag{11}$$

Once the error is sufficiently small, say less than 10^{-6}, then the iteration procedure can be stopped.

The algorithm for boundary condition iteration may thus be presented as follows:

- Choose an initial value for the final state $\mathbf{x}^{(1)}(t_f)$ and a value for ϵ; set the iteration index j to 1.
- Integrate (1), (2), (5) and (9) backwards from $t = t_f$ to $t = 0$, using for control (7). (2) is not needed for the algorithm, but it will give the performance index.
- Evaluate the error in the initial state from (11), and if it is less than the specified value, end the iteration.
- Increment the iteration index j by one. Choose a new value for the final state $\mathbf{x}^{(j)}(t_f)$ from (10) and go to step 2.

The procedure is therefore straightforward, since the equations are all integrated in the same direction. Furthermore, there is no need to store any variables over the trajectory. There is the added advantage that the control appears as a continuous variable, and therefore the accuracy of results will not depend on the size of the integration time step. Theoretically the results should be as good as can be obtained by the second variation method in control vector iteration. It is important to realize, however, that the Hamiltonian must be well behaved, so that (7) can be obtained analytically. The only drawback is the potential instability since the state equation and the sensitivity equation are integrated backwards, and problems may arise if the final time t_f is too large. For many problems in chemical engineering the BCI method can be easily applied as is shown in the following example.

Illustration of the Boundary Condition Iteration Procedure

Let us consider the nonlinear continuous stirred tank reactor that has been used for optimal control studies in [4, pp. 308–318], and which was shown in [13] to exhibit multiplicity of solutions. The system is described by the two equations

$$\frac{dx_1}{dt} = -2(x_1 + 0.25) + (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right) - u(x_1 + 0.25), \quad (12)$$

$$\frac{dx_2}{dt} = 0.5 - x_2 - (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right), \quad (13)$$

with the initial state $x_1(0) = 0.09$ and $x_2(0) = 0.09$. The control u is a scalar quantity related to the valve opening of the coolant. The state variables x_1 and x_2 represent deviations from the steady state of dimensionless temperature and concentration, respectively. The performance index to be minimized is

$$I = \int_0^{t_f} (x_1^2 + x_2^2 + 0.1u^2)\, dt, \quad (14)$$

where the final time $t_f = 0.78$. The Hamiltonian is

$$H = z_1(-2(x_1 + 0.25) + R - u(x_1 + 0.25)) + z_2(0.5 - x_2 - R) + x_1^2 + x_2^2 + 0.1u^2, \quad (15)$$

where $R = (x_2 + 0.5)\exp(25\, x_1/(x_1 + 2))$. The adjoint equations are

$$\frac{dz_1}{dt} = (u + 2)z_1 - 2x_1 + 50R\frac{(z_2 - z_1)}{(x_1 + 2)^2}, \quad (16)$$

$$\frac{dz_2}{dt} = -2x_2 + \frac{(z_2 - z_1)}{(x_2 + 0.5)}R + z_2. \quad (17)$$

The gradient of the Hamiltonian is

$$\frac{\partial H}{\partial u} = 0.2u - (x_1 + 0.25)z_1, \quad (18)$$

so the optimal control is given by

$$u = 5(x_1 + 0.25)z_1. \quad (19)$$

The equations for the transition matrix are:

$$\frac{d\Phi_{11}}{dt} = \frac{\partial f_1}{\partial x_1}\Phi_{11} + \frac{\partial f_1}{\partial x_2}\Phi_{21},$$
$$\frac{d\Phi_{12}}{dt} = \frac{\partial f_1}{\partial x_1}\Phi_{12} + \frac{\partial f_1}{\partial x_2}\Phi_{22},$$
$$\frac{d\Phi_{21}}{dt} = \frac{\partial f_2}{\partial x_1}\Phi_{11} + \frac{\partial f_2}{\partial x_2}\Phi_{21},$$
$$\frac{d\Phi_{22}}{dt} = \frac{\partial f_2}{\partial x_1}\Phi_{12} + \frac{\partial f_2}{\partial x_2}\Phi_{22}$$

where

$$\frac{\partial f_1}{\partial x_1} = -2 + \frac{50R}{(x_1 + 2)^2} - u,$$
$$\frac{\partial f_1}{\partial x_2} = \frac{R}{(x_2 + 0.5)},$$
$$\frac{\partial f_2}{\partial x_1} = -\frac{50R}{(x_1 + 2)^2},$$
$$\frac{\partial f_2}{\partial x_2} = -1 + \frac{R}{(x_2 + 0.5)}.$$

The adjustment of the final state is carried out by the following two equations:

$$x_1^{(j+1)}(t_f) = x_1^{(j)}(t_f) + \epsilon\left[\Phi_{11}(0)(x_1^{(j)}(0) - x_1(0)) + \Phi_{12}(0)(x_2^{(j)}(0) - x_2(0))\right], \quad (20)$$

$$x_2^{(j+1)}(t_f) = x_2^{(j)}(t_f) + \epsilon\left[\Phi_{21}(0)(x_1^{(j)}(0) - x_1(0)) + \Phi_{22}(0)(x_2^{(j)}(0) - x_2(0))\right]. \quad (21)$$

To illustrate the computational aspects of BCI, the above algorithm was used with a Pentium-120 personal computer using WATCOM Fortran compiler version 9.5. The calculations were done in double precision. When the performance index is included, there are 9 differential equations to be integrated backwards at each iteration. Standard fourth order Runge–Kutta method was used for integration with a stepsize of 0.01. For stability, it was found that ϵ had to be taken of the order of 0.1. For all the runs, therefore, this value of ϵ was used. As is shown in Table 1, to get the error less than 10^{-6}, a large number of iterations are required, but the computation time is quite reasonable. The optimal value of the performance index is very close to the value $I = 0.133094$ reported in [13] with the second variation

Boundary Condition Iteration BCI, Table 1
Application of BCI to CSTR

Initial choice $x_1(t_f) = x_2(t_f)$	Performance index	Number of iterations	CPU time s
0.045	0.133095	2657	13.9
0.00	0.133097	2858	14.9
−0.045	0.133097	2931	15.3
0.01	0.133097	2805	14.7

method and is essentially equivalent to $I = 0.133101$ obtained in [6] by using 20 stages of piecewise linear control with iterative dynamic programming. By refining the error tolerance to $e < 10^{-8}$ required no more than an additional thousand iterations with an extra expenditure of about 6 seconds of computation time in each case. Then the final value of the performance index for each of the four different initial starting points was $I = 0.133096$.

Now that computers are very fast and their speed is rapidly being improved, and computation time is no longer prohibitively expensive, the large number of iterations required by BCI should not discourage one from using the method. Since the control policy is directly inside the integration routine, equivalent results to those obtained by second variation method can be obtained. The number of equations, however, to be integrated is quite high with a moderately high-dimensional system. If we consider a system with 10 state variables, there are 121 differential equations to be integrated simultaneously. Although computationally this does not represent a problem, the programming could be a challenge to derive and enter the equations without error. Therefore, BCI methods for which the $(n \times n)$ transition matrix is not used may find a more widespread application. One possible approach is now presented.

Sensitivity Information Without Evaluating the Transition Matrix

Suppose at iteration j we have n sets of final states $\mathbf{x}^{(j-n+1)}(t_f), \ldots, \mathbf{x}^{(j)}(t_f)$ with corresponding values for the initial state obtained by integration $\mathbf{x}^{(j-n+1)}(0), \ldots, \mathbf{x}^{(j)}(0)$. Then we can write the transformation

$$\mathbf{P} = \mathbf{AQ}, \tag{22}$$

where

$$\mathbf{P} = \left(\mathbf{x}^{(j-n+1)}(t_f) \quad \cdots \quad \mathbf{x}^{(j)}(t_f)\right), \tag{23}$$

and

$$\mathbf{Q} = \left(\mathbf{x}^{(j-n+1)}(0) \quad \cdots \quad \mathbf{x}^{(j)}(0)\right). \tag{24}$$

The transformation matrix

$$\mathbf{A} = \mathbf{PQ}^{-1} \tag{25}$$

and the next vector at t_f is chosen as

$$\mathbf{x}^{(j+1)}(t_f) = \mathbf{Ax}(0). \tag{26}$$

(1) and (5) are integrated backward to obtain $\mathbf{x}^{(j+1)}(0)$, and the matrices $\mathbf{P}$ and $\mathbf{Q}$ are updated and the procedure continued. If the initial guesses are sufficiently close to the optimal, very rapid convergence is expected.

1	Pick n sets of values for $x(t_f)$ and integrate (1) and (5) backward from $t = t_f$ to $t = 0$. using (7) for control, to give n sets of initial state vectors.
2	From these two sets of vectors form the $(n \times n)$ matrices $\mathbf{P}$ and $\mathbf{Q}$.
3	Calculate $\mathbf{A}$ from (25), and calculate a new vector $\mathbf{x}^{(j+1)}(t_f)$ from (26).
4	With the vector from Step 3 as a starting condition, integrate (1) and (5) backward to give $\mathbf{x}^{(j+1)}(0)$.
5	Use the vectors in Steps 3 and 4 to replace $\mathbf{x}^{(j-n+1)}(t_f)$ and $\mathbf{x}^{(j-n+1)}(0)$ imn matrices $\mathbf{P}$ and $\mathbf{Q}$ and continue until the error as calculated from (11) is below some tolerance, such as 10^{-8}.

Boundary Condition Iteration BCI, Algorithm

For good starting conditions, one may use *iterative dynamic programming* (IDP) [9], and pick the final states obtained after each of the first n passes. F. Hartig and F.J. Keil [2] found that in the optimization of spherical reactors, IDP provided excellent values which were refined by the use of sequential quadratic programming. For convergence here we need good starting conditions. This is now illustrated with the above example.

By using IDP, as described in [6,7,8] for piecewise linear continuous control, with 3 randomly chosen points and 10 iterations per pass for piecewise linear control with 15 time stages, the data for the first four passes in Table 2 give good starting conditions for BCI.

By using as starting conditions the final states obtained in passes 1 and 2 as given in Table 2, the convergence is very fast with the above algorithm as is shown in Table 3. Only 9 iterations are required to yield $I = 0.133096$.

As expected, if the initial set of starting points is better, then the *convergence rate* is also better as is seen in comparing Table 4 to Table 3. However, in each case the total computation time was only 0.05 seconds on a Pentium-120. Taking into account that it takes 0.77 seconds to generate the initial conditions with IDP, it is observed that the optimum is obtained in less than 1 second of computation time. Therefore, BCI is a very useful procedure if (6) can be solved explicitly for the control and the final time t_f is not too large. Simple constraints on control can be readily handled by clipping technique, as shown in [12]. Further examples with this approach are given in [10].

Boundary Condition Iteration BCI, Table 2
Results of the first four passes of IDP

Pass no.	Perf. index	$x_1(t_f)$	$x_2(t_f)$	CPU time s
1	0.1627	0.05359	−0.13101	0.39
2	0.1415	0.01940	−0.05314	0.77
3	0.1357	0.05014	−0.09241	1.16
4	0.1334	0.05670	−0.10084	1.54

Boundary Condition Iteration BCI, Table 3
Convergence with the above algorithm from the starting points obtained in passes 1 and 2 by IDP

Iteration no.	Perf. index	Error ε
1	0.014520	0.1215
2	0.031301	0.1031
3	0.129568	$0.1852 \cdot 10^{-2}$
4	0.136682	$0.2414 \cdot 10^{-2}$
5	0.135079	$0.1350 \cdot 10^{-2}$
6	0.133218	$0.8293 \cdot 10^{-4}$
7	0.133093	$0.2189 \cdot 10^{-5}$
8	0.133096	$0.1373 \cdot 10^{-6}$
9	0.133096	$0.5209 \cdot 10^{-8}$

Boundary Condition Iteration BCI, Table 4
Convergence with the above algorithm from the starting points obtained in passes 3 and 4 by IDP

Iteration no.	Perf. index	Error ε
1	0.121769	$0.7353 \cdot 10^{-2}$
2	0.135249	$0.1415 \cdot 10^{-2}$
3	0.133317	$0.1531 \cdot 10^{-3}$
4	0.133138	$0.2861 \cdot 10^{-4}$
5	0.133094	$0.1703 \cdot 10^{-5}$
6	0.133096	$0.1190 \cdot 10^{-7}$
7	0.133096	$0.5364 \cdot 10^{-10}$

See also

► Control Vector Iteration

References

1. Denn MM, Aris R (1965) Green's functions and optimal systems – Necessary conditions and an iterative technique. Industr Eng Chem Fundam 4:7–16
2. Hartig F, Keil FJ (1993) Large scale spherical fixed bed reactors – modelling and optimization. Industr Eng Chem Res 32:57–70
3. Jaspan RK, Coull J (1972) Trajectory optimization techniques in chemical engineering. II. Comparison of the methods. AIChE J 18:867–869
4. Lapidus L, Luus R (1967) Optimal control of engineering processes. Blaisdell, Waltham
5. Luus R (1974) BCI vs. CVI. AIChE J 20:1039–1040
6. Luus R (1993) Application of iterative dynamic programming to very high dimensional systems. Hungarian J Industr Chem 21:243–250
7. Luus R (1993) Piecewise linear continuous optimal control by iterative dynamic programming. Industr Eng Chem Res 32:859–865
8. Luus R (1996) Numerical convergence properties of iterative dynamic programming when applied to high dimensional systems. Chem Eng Res Des 74:55–62
9. Luus R (1998) Iterative dynamic programming: from curiosity to a practical optimization procedure. Control and Intelligent Systems 26:1–8
10. Luus R (2000) Iterative dynamic programming. Chapman and Hall/CRC, London
11. Luus R (2000) A new approach to boundary condition iteration in optimal control. In: Proc. IASTED Internat. Conf. Control and Applications, Cancun, Mexico, May 24–27, 2000, pp 172–176
12. Luus R (2001) Further developments in the new approach to boundary condition iteration in optimal control. Canad J Chem Eng 79:968–976

13. Luus R, Cormack DE (1972) Multiplicity of solutions resulting from the use of variational methods in optimal control problems. Canad J Chem Eng 50:309–311
14. Rao SN, Luus R (1972) Evaluation and improvement of control vector iteration procedures for optimal control. Canad J Chem Eng 50:777–784

Bounding Derivative Ranges

George F. Corliss[1], L. B. Rall[2]
[1] Marquette University, Milwaukee, USA
[2] University Wisconsin–Madison, Madison, USA

MSC2000: 90C30, 90C26

Article Outline

Keywords
Evaluation of Functions
Monotonicity
Taylor Form
Intersection and Subinterval Adaptation
Software Availability
See also
References

Keywords

Interval arithmetic; Automatic differentiation; Taylor series

Interval arithmetic can be used to bound the range of a real function over an interval. Here, we bound the ranges of its *Taylor coefficients* (and hence derivatives) by evaluating it in an interval Taylor arithmetic. In the context of classical numerical methods, truncation errors, Lipschitz constants, or other constants related to existence or convergence assertions are often phrased in terms of bounds for certain derivatives. Hence, interval inclusions of Taylor coefficients can be used to give *guaranteed bounds* for quantities of concern to classical methods.

Evaluating the expression for a function using interval arithmetic often yields overly pessimistic bounds for its range. Our goal is to tighten bounds for the range of f and its derivatives by using a *differentiation arithmetic* for series generation. We apply *monotonicity* and *Taylor form tests* to each intermediate result of the calculation, not just to f itself. The resulting inclusions for the range of derivative values are several orders of magnitude tighter than bounds obtained from differentiation arithmetic and interval calculations alone. Tighter derivative ranges allow validated applications such as optimization, nonlinear equations, quadrature, or differential equations to use larger steps, thus improving their computational efficiency.

Consider the set of q times continuously differentiable functions on the real interval $\mathbf{x} = [\underline{x}, \overline{x}]$ denoted by $f(x) \in C^q[\mathbf{x}]$. We wish to compute a tight inclusion for

$$R(f^{(p)}; \mathbf{x}) := \left\{ f^{(p)}(x) \colon \underline{x} \le x \le \overline{x} \right\}, \tag{1}$$

where $p \le q$. We assume that f is sufficiently smooth for all indicated computations, and that all necessary derivatives are computed using *automatic differentiation* (cf. [5], ▶ Automatic differentiation: Point and interval Taylor operators).

Computing an inclusion for the range of $f^{(p)}$ is a generalization of the problem of computing an inclusion for the range of f, $R(f; \mathbf{x})$. Moore's *natural interval extension* [3] gives an inclusion which is often too gross an overestimation to be practical. H. Ratschek and J. Rokne [8] gives a number of improved techniques and many references. The approach of this paper follows from two papers of L.B. Rall [6,7] and from [1]. Taken together, Rall's papers outline four approaches to computing tight inclusions of $R(f; \mathbf{x})$, which we apply to derivatives:

- monotonicity,
- *mean value* and *Taylor forms*,
- *intersection*, and
- *subinterval adaptation*.

We apply the monotonicity tests and the Taylor form to each term of the Taylor polynomial of a function. Whenever we compute more than one enclosure for a quantity, either a derivative or an intermediate value, we compute intersections of all such enclosures. We apply these tests to each intermediate result of the calculation, not just to f itself. The bounds we compute for $R(f^{(p)}; \mathbf{x})$ are often *several orders of magnitude* tighter than bounds computed from natural interval extensions. In one example, we improve the interval inclusion for $R(f^{(10)}; \mathbf{x})$ from [− 3.8*E*10, 7.8E10] (width = 1.1*E*11) to [− 2.1*E*03, 9.6*E*03] (width = 1.1*E*04).

This improvement by a factor of 10^7 allows a Gaussian quadrature using 5 points per panel or a 10th order ODE solver (applications for which bounds for $R(f^{(10)}; \mathbf{x})$ might be needed) to increase their stepsizes, and hence their computational efficiency, by a factor of $10^{7/10} \approx 5$.

We discuss the evaluation of a function from a *code list* representation (see also ► Automatic differentiation: Point and interval Taylor operators). Then we discuss how monotonicity tests and Taylor form representations can be used to give tighter bounds for $R(f^{(p)}; \mathbf{x})$.

Evaluation of Functions

Functions are expressed in most computer languages by arithmetic operations and a set Φ of *standard functions*, for example, $\Phi = \{\text{abs, arctan, cos, exp, ln, sin, sqr, sqrt}\}$. A formula (or expression) can be converted into a code list or *computational graph* $\{t_1, \ldots, t_n\}$ (cf. [5], ► Automatic differentiation: Point and interval Taylor operators). The value of each term t_i is the result of a unary or binary operation or function applied to constants, values of variables, or one or two previous terms of the code list. For example, the function

$$f(x) = \frac{x^4 - 10x^2 + 9}{x^3 - 4x - 5}$$

can be converted into the code list

$t_1 := \text{sqr}(x);$	$t_6 := x \cdot t_1;$
$t_2 := \text{sqr}(t_1);$	$t_7 := 4 \cdot x;$
$t_3 := 10 \cdot t_1;$	$t_8 := t_6 - t_7;$
$t_4 := t_2 - t_3;$	$t_9 := t_8 - 5;$
$t_5 := t_4 + 9;$	$t_{10} := t_5/t_9.$

Bounding Derivative Ranges, Figure 1
Code list

The final term t_n of the code list (t_{10} in this case) gives the value of $f(x)$, if defined, for a given value of the variable x. The conversion of a formula into an equivalent code list can be carried out automatically by a computer subroutine.

The code list serves equally well for various kinds of arithmetic, provided the necessary arithmetic operations and standard functions are defined for the type of elements considered. Thus, the code list in Fig. 1 can serve for the computation of $f(x)$ in real, complex, interval, or differentiation arithmetic. When x is an interval, one gets an interval inclusion $f(\mathbf{x})$ of all real values $f(x)$ for real $x \in \mathbf{x}$ [3,4].

The process of automatic differentiation to obtain derivatives or Taylor coefficients of $f(x)$ can be viewed as the evaluation of the code list for $f(x)$ using a differentiation arithmetic in which the arithmetic operations and standard functions are defined on the basis of the well-known recurrence relations for Taylor coefficients (cf. also [3,4,5], ► Automatic differentiation: Point and interval Taylor operators). Let $(f)_i := f^{(i)}(\check{x})/i!$ be the value of the ith Taylor coefficient of $f(x) = f(\check{x} + h)$. Then we can express a Taylor series as

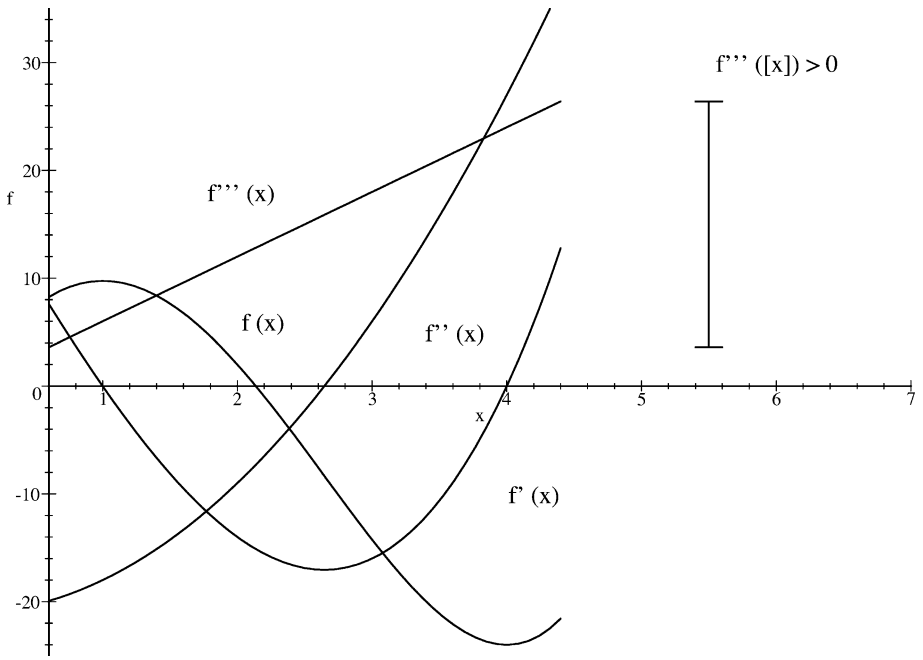
$$f(x) = \sum_{i=0}^{\infty} f^{(i)}(\check{x}) \frac{h^i}{i!} = \sum_{i=0}^{\infty} (f)_i h^i,$$

and the elements of Taylor series arithmetic are vectors $f = ((f)_0, \ldots, (f)_p)$. In Taylor arithmetic, constants c have the representation $c = (c, 0, \ldots, 0)$, and $x = (x_0, 1, 0, \ldots, 0)$ represents the independent variable $x = x_0 + h$. For example, multiplication $f(x) = u(x) \cdot v(x)$ of Taylor variables is defined in terms of the Taylor coefficients of u and v by $(f)_i = \sum_{j=0}^{i} (u)_j \cdot (v)_{i-j}$, $i = 0, \ldots, p$.

Monotonicity

We extend an idea of R.E. Moore for using monotonicity [4]: we check for the monotonicity of every derivative of f and of every intermediate function t_i from the code list. If the ith derivative of f is known to be of one sign on $\mathbf{x}$ ($R(f^{(i)}; \mathbf{x}) \geq 0$ or ≤ 0), then $f^{(i-1)}$ is monotonic on the interval $\mathbf{x}$, and its range is bounded by the real values $f^{(i-1)}(\underline{x})$ and $f^{(i-1)}(\overline{x})$. This is important because the bounds of $R(f^{(i-1)}; \mathbf{x})$ by $f^{(i-1)}(\underline{x})$ and $f^{(i-1)}(\overline{x})$ may be tighter than the bounds computed by the naive interval evaluation of $f^{(i-1)}(\mathbf{x})$. Hence, in addition to the ranges $R(f^{(i)}; \mathbf{x})$, we propagate enclosures of the values at the endpoints $R(f^{(i)}; \underline{\mathbf{x}})$ and $R(f^{(i)}; \overline{\mathbf{x}})$ so that those values are available. (We use $R(f^{(i)}; \underline{\mathbf{x}})$ and $R(f^{(i)}; \overline{\mathbf{x}})$ instead of $R(f^{(i)}; \underline{x})$ and $R(f^{(i)}; \overline{x})$ to denote that $f^{(i)}$ at the endpoints is evaluated in interval arithmetic.)

Similarly, if $R(f^{(i)}; \mathbf{x}) \geq 0$ (or ≤ 0), then $f^{(i-2)}$ is convex (resp. concave), and its maximum value is $\max(f^{(i-2)}(\underline{x}), f^{(i-2)}(\overline{x}))$ (resp. minimum is $\min(f^{(i-2)}(\underline{x}), f^{(i-2)}(\overline{x}))$).



Bounding Derivative Ranges, Figure 2
$R(f^{(3)}; \mathbf{x}) \geq 0$ implies f is monotonic and f' is convex

We apply the monotonicity test to each term of each intermediate result because an intermediate result may be monotonic when f is not. Further, by proceeding with tighter inclusions for the terms of the intermediate results, we reduce subsequent over-estimations and improve our chances for validating the monotonicity of higher derivatives. If $f^{(i-1)}$ is found to be monotonic, the tightened enclosure for $R(f^{(i-1)}; \mathbf{x})$ may allow us to validate $R(f^{(i-1)}; \mathbf{x}) \geq 0$ (or ≤ 0), so we backtrack to lower terms of the series as long as we continue to find monotonicity. In the recurrence relations for divide and for all of the standard functions, the value of $f^{(i)}(x)$ depends on the value of $f^{(i-1)}(x)$. Hence, if the enclosure for $R(f^{(i-1)}; \mathbf{x})$ is tightened, we recompute the enclosure for $R(f^{(i)}; \mathbf{x})$ and all subsequent terms.

Table 1 shows (some of) the results when the monotonicity test is applied to each of the intermediate results of

$$f(x) = \frac{x^4 - 10x^2 + 9}{x^3 - 4x - 5}$$

on the interval $\mathbf{x} := [1, 2]$. Each row shows enclosures for Taylor coefficients. The row 'x' has two entries for the function x evaluates on the interval $\mathbf{x}$ and its derivative. All higher-order derivatives are zero. Similarly, rows t_4 and t_5 have five nonzero derivatives.

A few entries show where tightening occurs because of the monotonicity test. For example, at ^{1}, the 3rd derivative of t_4 is positive. Hence, t_4 is monotonic, but that knowledge yields no tightening. Also $t_4{}'$ is convex, a fact which *does* allow us to tighten the upper bound from 12 to -8. Similarly at ^{2}, finding that t_8 is positive allows us to improve the upper bound for t_8. In this example, the monotonicity tests allow us only two relatively modest tightenings, but those two tighter values propagate through the recurrences to reduce the width of the bound finally computed for $t_{10}^{(5)}$ from about 2.3*E*6 to 300, an improvement of nearly a factor of 10^4.

Taylor Form

In [6], Rall proves that if $\check{x} \in x$, then

$$R(f; \mathbf{x}) \subset F_p(\mathbf{x}) := \sum_{i=0}^{p-1} (f)_i (\mathbf{x} - \check{x})^i + F^{(p)} \frac{(\mathbf{x})(\mathbf{x} - \check{x})^p}{p!}, \quad (2)$$

Bounding Derivative Ranges, Table 1
Numerical results of applying monotonicity tests

x					
$x(\mathbf{x})$	$[1, 2]$	1			
$t_1 := x^2$					
$t_1(\mathbf{x})$	$[1, 4]$	$[2, 4]$	1		
$t_2 := t_1^2 = x^4$					
$t_2(\mathbf{x})$	$[1, 16]$	$[4, 32]$	$[6, 24]$	$[4, 8]$	1
$t_3 := 10 * t_1 = 10x^2$					
$t_3(\mathbf{x})$	$[10, 40]$	$[20, 40]$	10		
$t_4 := t_2 - t_3 = x^4 - 10x^2$					
$t_4(\mathbf{x})$	$[-39, 6]$	$[-36, 12]$	$[-4, 14]$	$[4, 8]^1$	1
Tightened to:					
$t_4(\mathbf{x})$	$[-24, -9]$	$[-36, -8]$	$[-4, 14]$	$[4, 8]$	1
$t_5 := t_4 + 9 = x^4 - 10x^2 + 9$					
$t_5(\mathbf{x})$	$[-30, 15]$	$[-36, 12]$	$[-4, 14]$	$[4, 8]$	1
Tightened as the result of tightening t_4:					
$t_5(\mathbf{x})$	$[-15, 0]$	$[-36, -8]$	$[-4, 14]$	$[4, 8]$	1
...					
$t_8 := t_6 - t_7 = x^3 - 4x$					
$t_8(\mathbf{x})$	$[-7, 4]$	$[-1, 8]$	$[3, 6]^2$	1	
Tightened to:					
$t_8(\mathbf{x})$	$[-7, 0]$	$[-1, 8]$	$[3, 6]$	1	
$t_9 := t_8 - 5 = x^3 - 4x - 5$					
$t_9(\mathbf{x})$	$[-12, -1]$	$[-1, 8]$	$[3, 6]$	1	
Tightened as the result of tightening t_8:					
$t_9(\mathbf{x})$	$[-12, -5]$	$[-1, 8]$	$[3, 6]$	1	
$t_{10} := t_5/t_9 = f$					
$t_{10}(\mathbf{x})$	$[-15, 30]$			$[-10097, 20823]$	
		$[-132, 276]$			$[-87881, 181229]$
			$[-1160, 2392]$		$[-764851, 1577270]$
Tightened as the result of tightening t_5 and t_9:					
$t_{10}(\mathbf{x})$	$[-9.73E{-}0.6, 3.01]$			$[-9.68, 51.97]$	
		$[0.41, 12.01]$			$[-21.84, 113.67]$
			$[-5.21, 23.61]$		$[-47.58, 248.96]$

and $F^{(p)}$ is an interval extension of $f^{(p)}$. The F_p given by (2) is called the (elementary) Taylor form of f of order p.

We expand the Taylor series for the function f and all intermediate functions t_i appearing in the code list at three points, $x = a := x$, $x = c :=$ midpoint $(\mathbf{x})$, and $x = b := \overline{x}$. The series for f at $\underline{x}$ and $\overline{x}$ are already available since they were computed for the monotonicity test. The extra work required to generate the series at c is often justified because the midpoint form is much narrower than either of the endpoint forms. Let $h :=$ width $(\mathbf{x})$. We compute the Taylor form (2) for f and each t_i at the left endpoint, center, and right endpoint to all available orders and intersect. The remainder using $R(f^{(i+1)}; \mathbf{x})$ has the potential for tightening all previous terms:

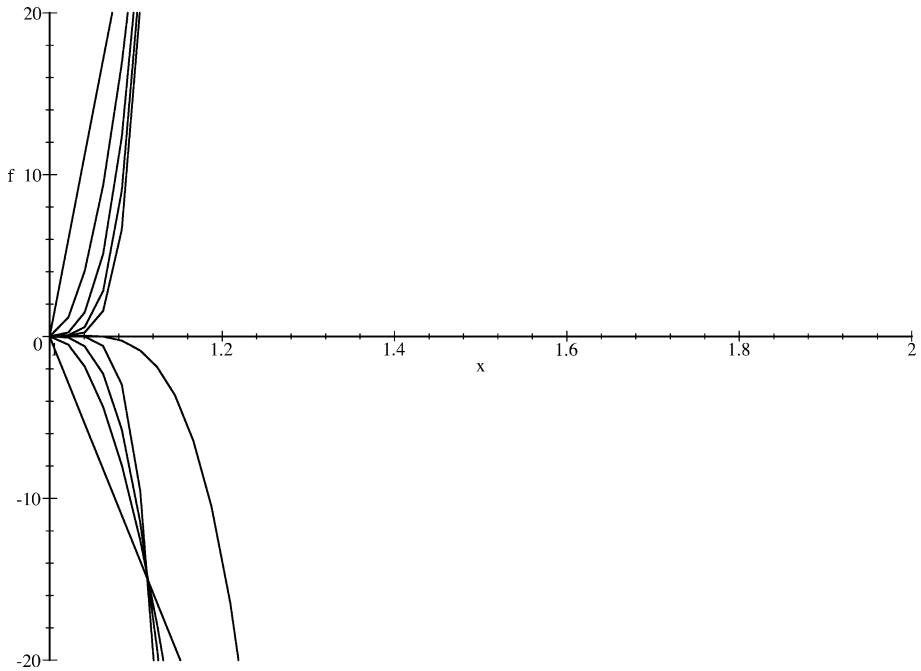
$$R(f;\mathbf{x}) \subset \left(f(a) + f'(a)h[0,1] + f''(a)\frac{h^2}{2!} * [0,1] + \cdots + f^{(i)}(a)\frac{h^i}{i!} * [0,1] + R(f^{(i+1)};\mathbf{x})\frac{h^{i+1}}{(i+1)!} * [0,1] \right)$$

Bounding Derivative Ranges, Table 2
Numerical results of applying Taylor from tests

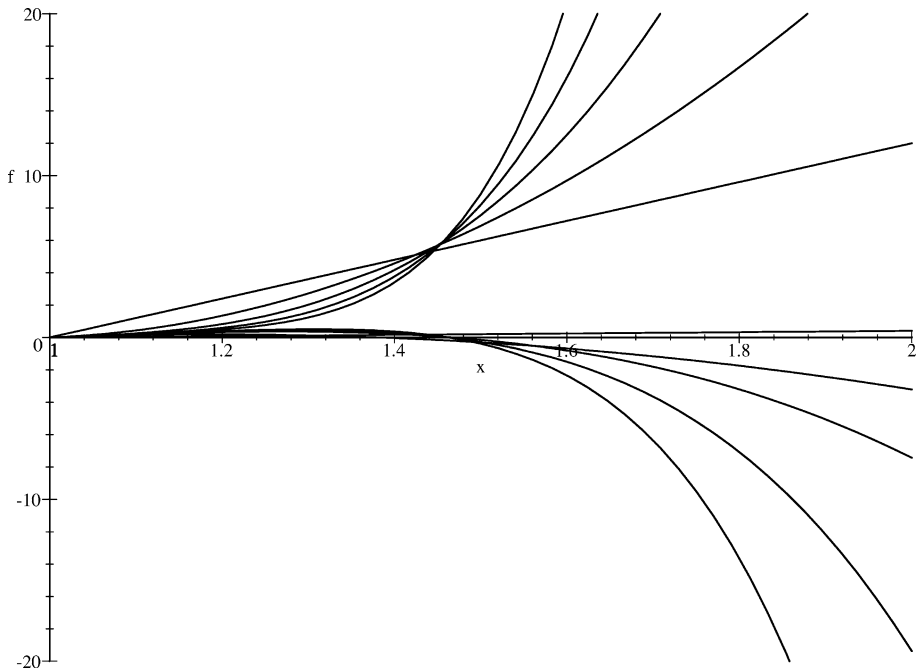
x					
$x(\mathbf{x})$	[1,2]	1			
$t_1 = x^2$					
$t_1([a,b])$	[1, 4]	[2, 4]	1		
No tightening occurs.					
...					
$t_4 = t_2 - t_3 = x^4 - 10x^2$					
$t_4(\mathbf{x})$	[−39, 6]	[−36, 12]	[−4, 14]	[4, 8]	1
		[−24, 12]	tightened by $f(a)$ using Fab(2)		
		[−24, -2.5]	tightened by $f(c)$ using Fab(2)		
		[−20, -7]	tightened by $f(c)$ using Fab(3)		
		[−20, -8]	tightened by $f(c)$ using Fab(4)		
	[−29, -9]	tightened by $f(a)$ using Fab(1)			
	[−27.438, -9]	tightened by $f(c)$ using Fab(1)			
	[−24, -9]	tightened by $f(b)$ using Fab(1)			
Tightened to :					
$t_4(\mathbf{x})$	[−24, -9]	[−20, -8]	[−4, 14]	[4, 8]	1
$t_5 = t_4 + 9 = x^4 - 10x^2 + 9$					
$t_5(\mathbf{x})$	[−30, 15]	[−36, 12]	[−4, 14]	[4, 8]	1
Tightened as the result of tightening t_4:					
$t_5(\mathbf{x})$	[−15, 0]	[−20, -8]	[−4, 14]	[4, 8]	1
$t_8 = t_6 - t_7 = x^3 - 4x$					
$t_8(\mathbf{x})$	[−7, 4]	[−1, 8]	[3, 6]	1	
Tightened to:					
$t_8(\mathbf{x})$	[−4, 0]	[−1, 8]	[3, 6]	1	
$t_{10} = t_5/t_9$					
$t_{10}(\mathbf{x})$	[−15, 30]			[−10097, 20823]	
		[−132, 276]			[−87881, 181229]
			[−1160, 2392]		[−764851, 1577270]
Tightened as the result of tightening t_5 and t_9:					
$t_{10}(\mathbf{x})$	[−9.73E−06, 3.01]			[−8.57, 39.94]	
		[0.55, 8.81]			[−19.27, 87.64]
			[−4.57, 18.49]		[−42.02, 191.84]
	[−6.08E−06, 3.01] tightened by f(c) using Fab(1)				

$$\cap \left(f(c) + f'(c)h * \frac{[-1,1]}{2} + f''(c)\frac{h^2}{2!} * \frac{[-1,1]}{4} + \cdots + f^{(i)}(c)\frac{h^i}{i!} * \frac{[-1,1]^i}{2^i} + R(f^{(i+1)};\mathbf{x})\frac{h^{i+1}}{(i+1)!} * \frac{[-1,1]^{i+1}}{2^{i+1}} \right)$$

$$\cap \left(f(b) + f'(b)h * [-1,0] + f''(b)\frac{h^2}{2!} * [0,1] + \cdots + f^{(i)}(b)\frac{h^i}{i!} * [-1,0]^i + R(f^{(i+1)};\mathbf{x})\frac{h^{i+1}}{(i+1)!} * [-1,0]^{i+1} \right).$$



Bounding Derivative Ranges, Figure 3
Taylor polynomial enclosures for *f*, remainders from naive interval evaluation



Bounding Derivative Ranges, Figure 4
Taylor polynomial enclosures for *f*, remainders tightened by Taylor form

For higher-order derivatives, $R(f^{(i)}; \mathbf{x})$ is contained in similar Taylor forms involving $f^{(i+n)}(\mathbf{x})$, for $n > 0$.

We apply the Taylor form to each intermediate result. Except for the operators +, −, ∗, and sqr, whenever one term is tightened, all following terms can be recomputed more tightly. This can result in an iterative process which is finite only by virtue of Moore's theorem on interval iteration [4]. In practice then, we restrict the number of times subsequent terms are recomputed starting at a given order.

Table 2 shows (some of) the results when the Taylor form is applied to each of the intermediate results of

$$f(x) = \frac{x^4 - 10x^2 + 9}{x^3 - 4x - 5}$$

on the interval $\mathbf{x} := [1, 2]$. 'Tightened by $f(a)$, $f(c)$, or $f(b)$' indicates whether the left endpoint, the midpoint, or the right endpoint expansion was used. 'Using Fab(n)' indicates that $f(i)$ was tightened using $f^{(i+n)}$.

The pattern of Table 2 is typical: Most Taylor forms give no tightening; there are many small improvements; and the compound effect of many small improvements is significant. Here we have reduced the width of the enclosure for the 6th Taylor coefficient from about 2.3*E*7 to 2.3*E*2. Figures 3 and 4 compare the Taylor polynomial enclosures for f resulting from naive interval evaluation of the remainders with the enclosures tightened by the Taylor form computations shown in Table 2.

For this example, the bounds achieved using the Taylor form are tighter than those achieved using the monotonicity test. For other examples, the monotonicity test performs better. Hence in practice, we apply both techniques. If the expression for f is rewritten in a mathematically equivalent form to yield tighter interval bounds for $R(f; \mathbf{x})$, the techniques of this paper can still be used profitably to tighten enclosures of higher derivatives.

Intersection and Subinterval Adaptation

The third general technique described by Rall for tightening enclosures of $R(f; \mathbf{x})$ is to intersect all enclosures for each quantity, as we have done here. That is, whatever bounds for $R(f^{(i)}; \mathbf{x})$ we compute using monotonicity or Taylor form of any degree, we intersect with the tightest bound previously computed. Each new bound may improve our lower bound, our upper bound, both, or neither. Some improvements are large. Others are so small as to seem insignificant, but even the smallest improvements may be magnified by later operations.

Rall's fourth technique is the *adaptive partitioning* of the interval $\mathbf{x}$. The over-estimation of $R(f^{(i)}; \mathbf{x})$ by naive interval evaluation decreases linearly with width ($\mathbf{x}$), while the over-estimation by the Taylor form decreases quadratically. Hence, partitioning $\mathbf{x}$ into smaller subintervals is very effective. However, we view subinterval adaptation as more effectively controlled by the application (e. g., optimization, quadrature, DE solution) than by the general-purpose interval Taylor arithmetic outlined here. Hence, we do not describe it further.

Software Availability

An implementation in Ada of interval Taylor arithmetic operators for +, −, ∗, /, and sqr is available at [9]. Similar implementations could be written in Fortran 90, C++, or any other language supporting operator overloading.

See also

- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Point and Interval Taylor Operators
- ▶ Global Optimization: Application to Phase Equilibrium Problems
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Interval Analysis: Differential Equations
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Interval Analysis: Intermediate Terms
- ▶ Interval Analysis: Nondifferentiable Problems
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ▶ Interval Analysis: Systems of Nonlinear Equations
- ▶ Interval Analysis: Unconstrained and Constrained Optimization
- ▶ Interval Analysis: Verifying Feasibility

- ▶ Interval Constraints
- ▶ Interval Fixed Point Theory
- ▶ Interval Global Optimization
- ▶ Interval Linear Systems
- ▶ Interval Newton Methods

References

1. Corliss GF, Rall LB (1991) Computing the range of derivatives. In: Kaucher E, Markov SM, Mayer G (eds) Computer Arithmetic, Scientific Computation and Mathematical Modelling. IMACS Ann Computing Appl Math. Baltzer, Basel, pp 195–212
2. Gray JH, Rall LB (1975) INTE: A UNIVAC 1108/1110 program for numerical integration with rigorous error estimation. MRC Techn Summary Report Math Res Center, Univ Wisconsin–Madison 1428
3. Moore RE (1966) Interval analysis. Prentice-Hall, Englewood Cliffs, NJ
4. Moore RE (1979) Methods and applications of interval analysis. SIAM, Philadelphia
5. Rall LB (1981) Automatic differentiation: techniques and applications. Lecture Notes Computer Sci, vol 120. Springer, Berlin
6. Rall LB (1983) Mean value and Taylor forms in interval analysis. SIAM J Math Anal 2:223–238
7. Rall LB (1986) Improved interval bounds for ranges of functions. In: Nickel KLE (ed) Interval Mathematics (Freiburg, 1985). Lecture Notes Computer Sci, vol 212. Springer, Berlin, pp 143–154
8. Ratschek H, Rokne J (eds) (1984) Computer methods for the range of functions. Horwood, Westergate
9. Website: www.mscs.mu.edu/~georgec/Pubs/eoo_da.tar.gz

Bounds and Solution Vector Estimates for Parametric NLPS

VIVEK DUA, EFSTRATIOS N. PISTIKOPOULOS
Imperial College, London, UK

MSC2000: 90C31

Article Outline

Keywords
Parametric Lower Bound
Parametric Upper Bound
See also
References

Keywords

Sensitivity analysis; Linear approximation; Parametric upper and lower bounds

In this article, we present some important theoretical results based upon which solution of parametric nonlinear programming problems can be approached. The need for these results arises from the fact that while stability, continuity and convexity properties of objective function value for linear programs are readily available [7], their counterparts in nonlinear programs are valid only for a special class of nonlinear programs. It is not surprising then that a large amount of research has been devoted towards establishing these conditions (see [1] and [3] for a comprehensive list of references). Further, due to the existence of strong duality results for linear models, parametric programming can be done by extending the simplex algorithm for linear models [6]. On the other hand, for nonlinear programs the parametric solution is given by an approximation of the optimal solution. This approximation or estimation of the optimal solution can be achieved by obtaining the optimal solution as a function of parameters. In order to derive these results we first state the following *implicit function theorem*:

Theorem 1 (see for example [3,8]) *Suppose that $\phi(x, \theta)$ is a $(r \times 1)$ vector function defined on $E^n \times E^m$, with $x \in E^n$ and $\theta \in E^m$, and $D_x \phi(x, \theta)$ and $D_\theta \phi(x, \theta)$ indicate the $(r \times n)$ and $(r \times m)$ matrix of first derivatives with respect to x and θ respectively. Suppose that $\phi: E^{m+n} \to E^n$. Let $\phi(x, \theta)$ be continuously differentiable in x and θ in an open set at (x_0, θ_0) where $\phi(x_0, \theta_0) = 0$. Suppose that $D_x \phi(x_0, \theta_0)$ has an inverse.*

Then there is a function $x(\theta)$ defined in a neighborhood of θ_0 where for each $\widehat{\theta}$ in that neighborhood $\phi[x(\widehat{\theta}), \widehat{\theta}] = 0$. Furthermore, $x(\theta)$ is a continuously differentiable function in that neighborhood and

$$
\begin{aligned}
&x'(\theta_0) \\
&= -D_x\phi[x(\theta_0), \theta_0]^{-1} D_\theta \phi[x(\theta_0), \theta_0] \\
&= -D_x\phi(x_0, \theta_0)^{-1} D_\theta \phi(x_0, \theta_0),
\end{aligned}
$$

where $x'(\theta_0)$ denotes the derivative of x evaluated at θ_0.

Consider the parametric nonlinear programming problem of the following form:

$$\begin{cases} z(\theta) = & \min\limits_{x} f(x,\theta) \\ \text{s.t.} & g_i(x,\theta) \geq 0, \quad i = 1,\ldots,p, \\ & h_j(x,\theta) = 0, \quad j = 1,\ldots,q, \\ & x \in X, \end{cases} \tag{1}$$

where f, g and h are twice continuously differentiable in x and θ. The *first order KKT conditions* for (1) are given as follows:

$$\begin{aligned} &\nabla f(x,\theta) - \sum_{i=1}^{p} \lambda_i \nabla g_i(x,\theta) + \sum_{j=1}^{q} \mu_j \nabla h_j(x,\theta) = 0, \\ &\lambda_i g_i(x,\theta) = 0, \quad i = 1,\ldots,p, \\ &h_j(x,\theta) = 0, \quad j = 1,\ldots,q\,. \end{aligned} \tag{2}$$

An application of the implicit function theorem 1 to the KKT conditions (2) results in the following *basic sensitivity theorem*:

Theorem 2 ([2,3,8]) *Let θ_0 be a vector of parameter values and (x_0, λ_0, μ_0) a KKT triple corresponding to (2), where λ_0 is nonnegative and x_0 is feasible in (1). Also assume that:*

i) strict complementary slackness holds;
ii) the binding constraint gradients are linearly independent;
iii) the second order sufficiency conditions hold.

Then, in neighborhood of θ_0, there exists a unique, once continuously differentiable function $[x(\theta), \lambda(\theta), \mu(\theta)]$ satisfying (2) with $[x(\theta_0), \lambda(\theta_0), \mu(\theta_0)] = (x_0, \lambda_0, \mu_0)$, where $x(\theta)$ is a unique isolated minimizer for (1), and

$$\begin{pmatrix} \frac{dx(\theta_0)}{d\theta} \\ \frac{d\lambda(\theta_0)}{d\theta} \\ \frac{d\mu(\theta_0)}{d\theta} \end{pmatrix} = -(M_0)^{-1} N_0, \tag{3}$$

where

$$M_0 = \begin{pmatrix} \nabla^2 L & -\nabla g_1 & \cdots & -\nabla g_p & \nabla h_1 & \cdots & \nabla h_q \\ \lambda_1 \nabla^\top g_1 & g_1 & & & & & \\ \vdots & & \ddots & & & & \\ \lambda_p \nabla^\top g_p & & & g_p & & & \\ \nabla^\top h_1 & & & & & & \\ \vdots & & & & & & \\ \nabla^\top h_r & & & & & & \end{pmatrix}$$

and

$$\begin{aligned} N_0 &= (\nabla^2_{\theta x} L, \lambda_1 \nabla^\top_\theta g_1, \ldots, \lambda_p \nabla^\top_\theta g_p, \\ &\nabla^\top_\theta h_1, \ldots, \nabla^\top_\theta h_q)^\top, \\ L(x,\lambda,\mu,\theta) &= f(x,\theta) \\ &+ \sum_{i=1}^{p} \lambda_i g_i(x,\theta) + \sum_{j=1}^{q} \mu_j h_j(x,\theta). \end{aligned}$$

However, for a special case of (1) when the parameters are present on the right-hand side of the constraints, (1) can be rewritten in the following form:

$$\begin{cases} z(\theta) & = \min\limits_{x} f(x) \\ \text{s.t.} & g(x) \geq \theta, \\ & x \in X. \end{cases} \tag{4}$$

A simplified version of an equivalent of (3) for (4) can also be obtained (see for example [8] for details). Another important result that can be derived for (4) is that the rate of change of the optimal value function, $z(\theta)$, with change in θ is given by KKT multiplier. Thus, given an optimal solution of (4) at a fixed point in θ_0, an estimate of the optimal solution in the neighborhood of θ_0 can be obtained by using the KKT multiplier obtained at θ_0 (see [8] and [9]). For a special case of (4), when (4) is convex in x and θ is bounded between certain lower and upper bounds, say 0 and 1, one can obtain a piecewise linear approximation of the optimal value function for the whole range of θ. In order to derive these results, we first state the following properties.

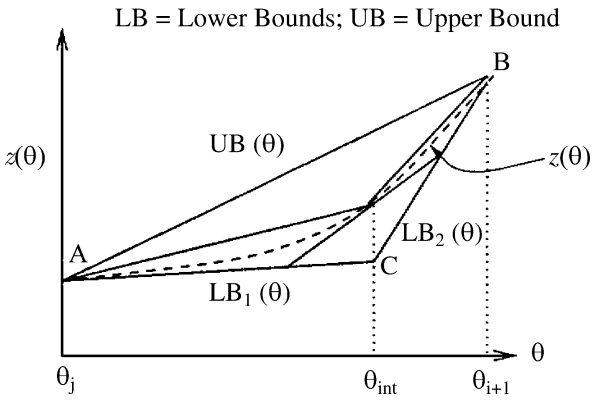
Theorem 3 (*continuity property of the objective function value*; [3]) *Let*

$$z(\theta) = \inf_{x} \{ f(x) \colon\ x \in X,\ g(x) \geq \theta \}\,.$$

Suppose

i) X is a compact convex set in E^n,
ii) f and g are both continuous on $X \times E^n$, and
iii) each component of g is strictly concave on X for each θ.

Then $z(\theta)$ is continuous on its effective domain.



Bounds and Solution Vector Estimates for Parametric NLPS, Figure 1
Bounds on the optimal value function $z(\theta)$

Theorem 4 (*convexity property of the solution space*; [4]) *Suppose*
i) g_i are jointly quasiconcave on X and θ, and
ii) X is convex.
Then the solution space $R(\theta) = \{x \in X: g(x) \geq \theta\}$, is convex.

Theorem 5 (*convexity property of the objective function value*; [4]) *Suppose*
i) f is convex on X, and
ii) the solution space $R(\theta)$ is essentially convex.
Then $z(\theta)$ is convex on θ.

Since $z(\theta)$ is continuous and convex under above conditions, for a given interval $[\theta_i, \theta_{i+1}]$, we can obtain [5] (see Fig. 1) parametric lower and upper bounds as follows.

Parametric Lower Bound

A linear underestimator of the convex function, $z(\theta)$, will be a global underestimator, hence lower bounds at θ_i and θ_{i+1} given by:

$$\mathrm{LB}_i(\theta) = z^*(\theta_i) + \nabla_\theta z^*(\theta_i)(\theta - \theta_i),$$
$$\mathrm{LB}_{i+1}(\theta) = z^*(\theta_{i+1}) + \nabla_\theta z^*(\theta_{i+1})(\theta - \theta_{i+1}),$$

where $\nabla_\theta\, z^*(\theta)$ is given by the Lagrange multipliers as discussed earlier, provide global underestimators to $z(\theta)$.

Parametric Upper Bound

A linear interpolation between the objective function value at the end points θ_i and θ_{i+1} given by:

$$\widehat{z}(\overline{y}, \theta) = \alpha z^*(\overline{y}, \theta_i) + (1-\alpha) z^*(\overline{y}, \theta_{i+1}), \alpha \in (0,1),$$

gives a valid upper bound because of the convexity of the objective function.

It may be mentioned that in this simple way we can obtain a region, ABC, within which the value of objective function will lie. An intersection point, θ_{int}, of the two lower bounds, $\mathrm{LB}_i(\theta)$ and $\mathrm{LB}_{i+1}(\theta)$, is then determined. At this point the value of lower and upper bounds are compared, and if the difference is within certain tolerance, ϵ, we stop, otherwise, the interval $[\theta_i, \theta_{i+1}]$ is subdivided into two intervals $[\theta_i, \theta_{int}]$ and $[\theta_{int}, \theta_{i+1}]$. In each of these regions a similar bounding procedure is repeated until we meet the tolerance criterion.

See also

- ▶ Multiparametric Linear Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Parametric Global Optimization: Sensitivity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Parametric Optimization: Embeddings, Path Following and Singularities
- ▶ Selfdual Parametric Method for Linear Programs

References

1. Bank B, Guddat J, Klatte D, Kummer B, Tammer K (1983) Nonlinear parametric optimization. Akad Verlag, Berlin
2. Fiacco AV (1976) Sensitivity analysis for nonlinear programming using penalty methods. Math Program 10:287–311
3. Fiacco AV (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Acad Press, New York
4. Fiacco AV, Kyparisis J (1986) Convexity and concavity properties of the optimal value function in parametric nonlinear programming. J Optim Th Appl 48:95–126
5. Fiacco AV, Kyparisis J (1988) Computable bounds on parametric solutions of convex problems. Math Program 40:213–221

6. Gal T (1995) Postoptimal analyses, parametric programming, and related topics. De Gruyter, Berlin
7. Gal T, Nedoma J (1972) Multiparametric linear programming. Managem Sci 18:406–422
8. McCormick GP (1983) Nonlinear programming: Theory, algorithms and applications. Wiley, New York
9. Stolbjerg A, Lasdon L (1997) Nonlinear programming. In: Gal T, Greenberg HJ (eds) Advances in Sensitivity Analysis and Parametric Programming. Kluwer, Dordrecht

Branch and Price: Integer Programming with Column Generation

BP

MARTIN W. P. SAVELSBERGH
School of Industrial and Systems Engineering, Georgia Institute Technology, Atlanta, USA

MSC2000: 68Q99

Article Outline

Keywords
Nonidentical Machines
Identical Machines
See also
References

Keywords

Optimization; Integer programming; Decomposition; Column generation

Branch and price is a generalization of *linear programming* (LP) based *branch and bound* specifically designed to handle *integer programming* (IP) formulations that contain a huge number of variables. The basic idea of branch and price is simple. Columns are left out of the *LP relaxation* because there are too many columns to handle efficiently and most of them will have their associated variable equal to zero in an optimal solution anyway. Then to check the optimality of an LP solution, a subproblem, called the *pricing problem*, is solved to try to identify columns with a profitable *reduced cost*. If such columns are found, the LP is reoptimized. Branching occurs when no profitable columns are found, but the LP solution does not satisfy the integrality conditions. Branch and price applies *column generation* at every node of the branch and bound tree.

There are several reasons for considering IP formulations with a huge number of variables.

- A compact formulation of an IP may have a weak LP relaxation. Frequently the relaxation can be tightened by a reformulation that involves a huge number of variables.
- A compact formulation of an IP may have a symmetric structure that causes branch and bound to perform poorly because the problem barely changes after branching. A reformulation with a huge number of variables can eliminate this symmetry.
- Column generation provides a *decomposition* of the problem into master and subproblems. This decomposition may have a natural interpretation in the contextual setting allowing for the incorporation of additional important constraints or nonlinear cost functions.
- A formulation with a huge number of variables may be the only choice.

At first glance, it may seem that branch and price involves nothing more than combining well-known ideas for solving linear programs by column generation with branch and bound. However, it is not that straightforward. There are fundamental difficulties in applying column generation techniques for linear programming in integer programming solution methods. These include:

- Conventional integer programming branching on variables may not be effective because fixing variables can destroy the structure of the pricing problem.
- Column generation often converges slowly and solving the LPs to optimality may become computationally prohibitive.

We illustrate the concepts of branch and price and the difficulties that may arise by means of an example.

In the *generalized assignment problem* (GAP) the objective is to find a maximum profit assignment of m tasks to n machines such that each task is assigned to precisely one machine subject to capacity restrictions on the machines. For reasons that will become apparent later, we will consider separately the two cases of nonidentical and identical machines.

Nonidentical Machines

The natural integer programming formulation of GAP is

$$\begin{cases} \max & \displaystyle\sum_{1\le i\le m}\sum_{1\le j\le n} p_{ij}z_{ij} \\ \text{s.t.} & \displaystyle\sum_{1\le j\le n} z_{ij} = 1, \quad i = 1,\ldots,m, \\ & \displaystyle\sum_{1\le i\le m} w_{ij}z_{ij} \le d_j, \quad j = 1,\ldots,n, \\ & z_{ij} \in \{0,1\}, \\ & \quad i = 1,\ldots,m, \quad j = 1,\ldots,n, \end{cases}$$

where p_{ij} is the profit associated with assigning task i to machine j, w_{ij} is the amount of the capacity of machine j used by task i, d_j is the capacity of machine j, and z_{ij} is a 0–1 variable indicating whether task i is assigned to machine j.

An alternative formulation of GAP in terms of columns representing feasible assignments of tasks to machines is

$$\begin{cases} \max & \displaystyle\sum_{j\le 1\le n}\sum_{1\le k\le K_j}\left(\sum_{1\le i\le m} p_{ij}y_{ik}^j\right)\lambda_k^j \\ \text{s.t.} & \displaystyle\sum_{1\le j\le n}\sum_{1\le k\le K_j} y_{ik}^j\lambda_k^j = 1, \quad i = 1,\ldots,m, \\ & \displaystyle\sum_{1\le k\le K_j} \lambda_k^j = 1, \quad j = 1,\ldots,n, \\ & \lambda_k^j \in \{0,1\}, \\ & \quad j = 1,\ldots,n, \quad k = 1,\ldots,K_j, \end{cases}$$

where the first m entries of a column, given by $y_k^j = (y_{1k}^j, \ldots, y_{mk}^j)$, satisfy the *knapsack constraint*

$$\sum_{1\le i\le m} w_{ij}x_i \le d_j, x_i \in \{0,1\}, \quad i = 1,\ldots,m,$$

and where K_j denotes the number of feasible solutions to the above knapsack constraint. The first set of constraints ensures that each task is assigned to a machine, and the second set of constraints, the convexity constraints, ensures that exactly one feasible assignment of tasks to machines is selected for each machine. This is in fact the formulation that is obtained when we apply *Dantzig–Wolfe decomposition* to the natural formulation of GAP with the assignment constraints defining the master problem and the machine capacity constraints defining the subproblems.

The reason for considering this alternative formulation of GAP is that the LP relaxation of the master problem is tighter than the LP relaxation of the natural formulation because certain fractional solutions are eliminated. Namely, all fractional solutions that are not *convex combinations* of 0–1 solutions to the knapsack constraints.

Unfortunately, the LP relaxation of the master problem cannot be solved directly due to the exponential number of columns. However, the LP relaxation of a restricted version of the master problem that considers only a subset of the columns can be solved directly using, for instance, the *simplex method*. Furthermore, if the reduced costs of all the columns that were left out are nonnegative, then the LP solution obtained is also optimal for the LP relaxation of the unrestricted master problem. To check whether there exist a column with positive reduced cost we solve the *pricing problem*

$$\max_{1\le j\le n}\{z(\mathrm{KP}_j) - v_j\},$$

where v_j is the optimal *dual price* from the solution to the LP relaxation of the restricted master problem associated with the convexity constraint of machine j, and $z(\mathrm{KP}_j)$ is the value of the optimal solution to the *knapsack problem*

$$\begin{cases} \max & \displaystyle\sum_{1\le i\le n} (p_{ij} - u_i)x_i \\ \text{s.t.} & \displaystyle\sum_{1\le i\le n} w_{ij}x_i \le d_j \\ & x_i \in \{0,1\}, \\ & \quad i \in \{1,\ldots,n\}, \end{cases}$$

with u_i being the optimal dual price from the solution to the LP relaxation of the restricted master problem associated with the assignment constraint of task i. If the optimal value of the pricing problem is positive, we have identified a column with positive reduced cost. In that case, we add the column to the restricted master problem and reoptimize.

The LP relaxation of the master problem solved by column generation may not have an integral optimal solution and applying a standard branch and bound procedure to the master problem over the ex-

isting columns is unlikely to find an optimal, or good, or even feasible solution to the original problem. Therefore it may be necessary to generate additional columns in order to solve the linear programming relaxations of the master problem at nonroot nodes of the search tree.

Standard branching on the λ-variables creates a problem along a branch where a variable has been set to zero. Recall that y_k^j represents a particular solution to the jth knapsack problem. Thus $\lambda_k^j = 0$ means that this solution is excluded. However, it is possible (and quite likely) that the next time the knapsack problem for the jth machine is solved the optimal solution is precisely the one represented by y_k^j. In that case, it would be necessary to find the second best solution to the knapsack problem. At depth l in the branch and bound tree we may need to find the lth best solution, which is very hard. Fortunately, there is a simple remedy to this difficulty. Instead of branching on the λs in the master problem, we use a branching rule that corresponds to branching on the original variables z_{ij}. When $z_{ij} = 1$, all existing columns in the master that do not assign task i to machine j are deleted and task i is permanently assigned to machine j, i. e., variable x_i is fixed to 1 in the jth knapsack. When $z_{ij} = 0$, all existing columns in the master that assign job i to machine j are deleted and task i cannot be assigned to machine j, i. e., variable x_i is removed from the jth knapsack. Note that each of the knapsack problems contains one fewer variable after the branching has been done.

Observe that the branching scheme discussed above is specific to the GAP. This is typical of branch and price algorithms. Each problem requires its own 'problem-specific' branching scheme.

In practice, one of the computational difficulties encountered when applying branch and price is the so-called *tailing-off* effect of the column generation, i. e., the large number of iterations needed to prove the optimality of the LP solution. Potentially, this may happen at every node of the search tree. Also, the pricing problem that needs to be solved at each column generation iteration may be difficult and time consuming. Fortunately, the branch and bound framework has some inherent flexibility that can be exploited effectively in branch and price algorithms. Branch and bound is an enumeration scheme that is enhanced by fathoming based on bound comparisons. To control the size of the branch and bound tree it is best to work with strong bounds; however, the method will work with any bound. Therefore, instead of solving the linear program to optimality, i. e., generating columns as long as profitable columns exist, we can choose to prematurely end the column generation process and work with bounds on the final LP value.

Again, consider the alternative formulation of GAP. By dualizing the assignment constraints, we obtain the following *Lagrangian relaxation*, which provides an upper bound on the value of the LP for any vector u.

$$
\begin{cases}
\max & \displaystyle\sum_{1\le j\le n}\sum_{1\le k\le K_j}\left(\sum_{1\le i\le m} p_{ij}y_{ik}^j\right)\lambda_k^j \\
& \quad + \displaystyle\sum_{1\le i\le m} u_i\left(1 - \sum_{1\le j\le n}\sum_{1\le k\le K_j} y_{ik}^j\lambda_k^j\right) \\
\text{s.t.} & \displaystyle\sum_{1\le k\le K_j}\lambda_k^j = 1, \quad j = 1,\dots,n, \\
& \lambda_k^j \in \{0,1\}, \\
& \quad j = 1,\dots,n, \quad ; k = 1,\dots,K_j.
\end{cases}
$$

After some algebraic manipulations, we obtain

$$
\begin{cases}
& \displaystyle\sum_{1\le i\le m} u_i + \sum_{1\le j\le n} \max \\
& \quad \displaystyle\sum_{1\le k\le K_j}\left(\sum_{1\le i\le m}(p_{ij} - u_j)y_{ik}^j\right)\lambda_k^j \\
\text{s.t.} & \displaystyle\sum_{1\le k\le K_j}\lambda_k^j = 1, \quad j = 1,\dots,n, \\
& \lambda_k^j \in \{0,1\}, \\
& \quad j = 1,\dots,n, \quad k = 1,\dots,K_j,
\end{cases}
$$

which is equivalent to

$$\sum_{1\le i\le m} u_i + \sum_{1\le j\le n} z(\mathrm{KP}_j).$$

This shows that after solving the pricing problem, we have all the information necessary to compute an upper bound on the value of the final LP solution. Therefore, after every column generation iteration, we may decide to prematurely end the column generation pro-

cess if the value of the LP solution to the current restricted master problem, which provides a lower bound on the final LP value, and this upper bound are sufficiently close.

Identical Machines

This is a special case of the problem with nonidentical machines and therefore the methodology described above applies. However, we need only one subproblem since all of the machines are identical, which implies that the λ_k^j can be aggregated by defining $\lambda_k = \sum_j \lambda_k^j$ and that the convexity constraints can be combined into a single constraint $\sum_{1 \le k \le K} \lambda_k = n$ where λ_k is restricted to be integer. In some cases the aggregated constraint will become redundant and can be deleted altogether. An example of this is when the objective is to minimize $\sum \lambda_k$, i. e., the number of machines needed to process all the tasks. Note that this special case of GAP is equivalent to a 0–1 *cutting-stock problem*.

A much more important issue here concerns symmetry, which causes branching on the original variables to perform very poorly. With identical machines, there are an exponential number of solutions that differ only by the names of the machines, i. e. by swapping the assignments of 2 machines we get 2 solutions that are the same but have different values for the variables. This statement is true for fractional as well as 0–1 solutions. The implication is that when a fractional solution is excluded at some node of the tree, it pops up again with different variable values somewhere else in the tree. In addition, the large number of alternate optima dispersed throughout the tree renders pruning by bounds nearly useless.

The remedy here is a different branching scheme that works directly on the master problem but focuses on pairs of tasks. In particular, we consider rows of the master with respect to tasks r and s. Branching is done by dividing the solution space into one set in which r and s appear together, in which case they can be combined into one task when solving the knapsack, and into another set in which they must appear separately, in which case a constraint $x_r + x_s \le 1$ is added to the knapsack. Note that the structure of the subproblems is no longer the same on the different branches.

Most of the material presented above is based on [3], in which the term branch and price was first introduced, and [1], in which the concepts of branch and price are covered in much more detail. Another important source of information on branch and price is [4], in which various general branching schemes and bounding schemes are discussed. Routing and scheduling has been a particularly fruitful application area of branch and price, see [2] for a survey of these results.

See also

- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-Objective Integer Linear Programming
- ▶ Multi-Objective Mixed Integer Programming
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-Dependent Traveling Salesman Problem

References

1. Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: Column generation for solving integer programs. Oper Res 46:316–329
2. Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995) Time constrained routing and scheduling. In: Ball ME, Magnanti TL, Monma C, Nemhauser GL (eds) Network RoutingHandbook Oper Res and Management Sci. 8, Elsevier, Amsterdam, pp 35–140
3. Savelsbergh MWP (1997) A branch-and-price algorithm for the generalized assignment problem. Oper Res 6:831–841
4. Vanderbeck F, Wolsey LA (1996) An exact algorithm for IP column generation. Oper Res Lett 19:151–160

Branchwidth and Branch Decompositions[1]

ILLYA V. HICKS
Computational and Applied Mathematics,
Rice University, Houston, USA

MSC2000: 90C27, 68R10

Article Outline

Keywords
Introduction
Graph Minors Theorem
Tangles
Constructing Branch Decompositions
Branch-Decomposition-Based Algorithms
Branchwidth of Matroids
Treewidth and Tree Decompositions
References

Keywords

Branchwidth; Branch decomposition; Tangle; Graph minors theorem; Branch-decomposition-based algorithm

Introduction

Let G be a graph (or hypergraph) with node set $V(G)$ and edge set $E(G)$. Let T be a tree having $|\,E(G)\,|$ leaves in which every non-leaf node has degree 3. Let ν be a bijection (one-to-one and onto function) from the edges of G to the leaves of T. The pair (T, ν) is called a *branch decomposition* of G. Notice that removing an edge, say e, of T partitions the leaves of T and the edges of G into two subsets A_e and B_e. The *middle set* of e and of (A_e, B_e), denoted by $mid(e)$ or $mid(A_e, B_e)$, is the set $V(G[A_e]) \cap V(G[B_e])$ where $G[A_e]$ is the subgraph of G induced by A_e and similarly for $G[B_e]$. The *width* of a branch decomposition (T, ν) is the maximum order of the middle sets over all edges in T. The *branchwidth* of G, denoted by $\beta(G)$, is the minimum width over all branch decompositions of G. A branch decomposition of G is *optimal* if its width is equal to the branchwidth of G. For example, Fig. 1 gives an optimal branch decomposition of an example graph where some of the middle sets of the edges of the branch decomposition are provided.

An edge e is contracted if e is deleted and the ends of e are identified into one node and a graph H is a *minor* of a graph G if H can be obtained from a subgraph of G by contracting edges. Graphs of small branchwidth are characterized by the following theorem.

Theorem 1 (Robertson and Seymour [49]) *A graph G has branchwidth:*

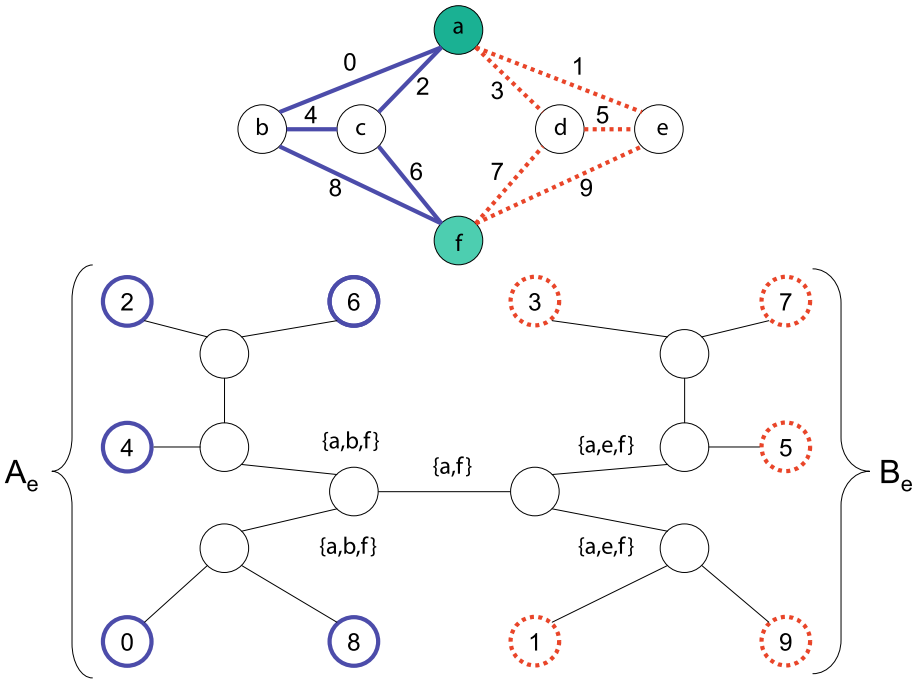
- *0 if and only if every component of G has ≤ 1 edge*
- *≤ 1 if and only if every component of G has ≤ 1 node of degree ≥ 2*
- *≤ 2 if and only if G has no K_4 minor.* □

Other classes of graphs with known branchwidth are grids, complete graphs, Halin graphs, and chordal graphs. The branchwidth of a $a \times b$-grid is the minimum of a and b while the branchwidth of a complete graph G with at least 3 nodes is $\lceil \frac{2}{3} \mid V(G) \mid \rceil$ [49]. Halin graphs have branchwidth 3 and the branchwidth of chordal graphs is bounded below by $\lceil \frac{2}{3} \mid \omega(G) \mid \rceil$ and above by $\omega(G)$ where $\omega(G)$ denotes the clique number of the graph [30].

Graph Minors Theorem

A *planar graph* is a graph that can be drawn on a sphere or plane without having edges that cross. A subdivision of a graph G is a graph obtained from G by replacing its edges by internally vertex disjoint paths. In the 1930s, Kuratowski [42] proved that a graph G is planar if and only if G does not contain a subdivision of K_5 or $K_{3,3}$. Let $\mathcal{F}$ be a class of graphs. $\mathcal{F}$ is *minor closed* when all the minors of any member of $\mathcal{F}$ also belong to $\mathcal{F}$. Given a minor closed class of graphs $\mathcal{F}$, the *obstruction set* of $\mathcal{F}$ is the set of minor minimal graphs that are not elements of $\mathcal{F}$. Clearly, any class of graphs embeddable on a given surface is a minor closed class. Erdös, also in the 1930's, posed the question of whether the obstruction set for a given surface is finite. Wagner [57] later proved that the sphere has a finite obstruction set, K_5 and $K_{3,3}$. The question of characterizing the obstruction set for surfaces other than the sphere remained open until 1979–1980 when Archdeacon [4] and Glover et al. [27] solved the case for the

[1]This research was partially supported by NSF grant DMI-0217265



Branchwidth and Branch Decompositions, Figure 1
Example Graph *G* with Optimal Branch Decomposition (*T*, *ν*) with width 3

projective plane where they proved that there are 35 minor minimal "non-projective-planar" graphs. Archdeacon and Huneke [5] proved that the obstruction set for any non-orientable surface is finite and Robertson and Seymour [48] proved the case for any surface as a corollary of the Graph Minors Theorem (formerly known as Wagner's conjecture): every minor closed class has a finite obstruction set. Branch decompositions, tangles, and tree decompositions, discussed in later sections, were beneficial to the proof of the Graph Minors Theorem.

Tangles

Let G be a graph (or hypergraph) and let $k \geq 1$ be an integer. A *separation* of a graph G is a pair (G_1, G_2) of subgraphs of G with $G_1 \cup G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2)) = G, E(G_1) \cap E(G_2) = \emptyset$ and the *order* of this separation is defined as $| V(G_1) \cap V(G_2) |$ where $V(G_1) \cap V(G_2)$ is called the *middle set* of the separation. For a hypergraph G, define $I(G)$ to be the bipartite graph such that the nodes of $I(G)$ correspond to the nodes and edges of G and an edge ev in $I(G)$ corresponds to the edge e of G being incident with the node v in G. A hypergraph G is called *connected* if $I(G)$ is connected. Also, denote $\gamma(G)$ as the largest cardinality of a set of nodes incident to an edge of G. A *tangle* in G of *order* k is a set $\mathcal{T}$ of separations of G, each of order $< k$ such that:

(T1) for every separation (A, B) of G of order $< k$, one of (A, B), (B, A) is an element of $\mathcal{T}$;
(T2) if $(A_1, B_1), (A_2, B_2), (A_3, B_3) \in \mathcal{T}$ then $A_1 \cup A_2 \cup A_3 \neq G$; and
(T3) if $(A, B) \in \mathcal{T}$ then $V(A) \neq V(G)$.

These are called the first, second and third tangle axioms. The *tangle number* of G, denoted by $\theta(G)$, is the maximum order of any tangle of G. Figure 2 gives an example of a tangle of order 3 for the graph in Fig. 1. Notice in Fig. 2 that the inclusion of separations of the graph of order 3 to the tangle would result in a violation of one of the tangle axioms. A tangle $\mathcal{T}$ of G with order k can be thought of as a "k-connected" component of G because some "k-connected" component of G will either be on one side or the other for any separa-

Separation of order 0
$(\emptyset, G)$
Separation of order 1
$(v, G)\ \forall v \in V(G)$
Separation of order 2
$(\{v, w\}, G)\ \forall v, w \in V(G)$
$(G[e], G[E(G)\backslash e])\ \forall e \in E(G)$
$(G[0, 2, 4, 6, 8], G[1, 3, 5, 7, 9])$

Branchwidth and Branch Decompositions, Figure 2
Tangle of Order 3 for the Example Graph of Fig. 1

tion of $\mathcal{T}$. Robertson and Seymour [49] proved a min-max relationship between tangles and branch decompositions, given below.

Theorem 2 (Robertson and Seymour [49]) *For any hypergraph G such that $E(G) \neq \emptyset$, $max\{\beta(G), \gamma(G)\} = \theta(G)$.* □

Related structures to tangles are respectful tangles and tangle bases. Respectful tangles of a graph G embedded on a surface Σ are tangles that are restricted according to the graph's embedding on Σ and the order of these tangles is limited by the graph's representativeness on Σ. Respectful tangles were discussed in the work of Robertson and Seymour [50] and created the foundation for the Seymour and Thomas [53] result for planar graphs. Tangle bases were introduced by Hicks [33] to assist in a branch-decomposition-based algorithm, discussed in a later section, to compute optimal branch decompositions for general graphs. Tangle bases are also restricted in the sense that the only members of a tangle basis are edges (just considering the first part of a separation) and separations which can be constructed from the union of edges. A formal definition is given below.

For an integer k and hypergraph G, a *tangle basis*, $\mathcal{B}$, of order k is a set of separations of G with order $< k$ such that:

(B1) $(G[e], G[E(G) \setminus e]) \in \mathcal{B} \forall e \in E(G)$ if $\gamma(e) < k$
(B2) if $(C, D) \in \mathcal{B}$ and $\nexists e \in E(G)$ such that $G[e] = C$, then $\exists (A_1, B_1), (A_2, B_2) \in \mathcal{B}$ such that $A_1 \cup A_2 = C$ and $B_1 \cap B_2 = D$
(B3) $\mathcal{B}$ obeys the tangle axioms T2 and T3.

Separations of order 2
$(G[e], G[E(G)\backslash e])\ \forall e \in E(G)$

Branchwidth and Branch Decompositions, Figure 3
Connected Tangle Basis of Order 3 for the Graph of Fig. 1

A tangle basis, $\mathcal{B}$, in G of order k is *connected* if every separation (A, B) of $\mathcal{B}$ has A connected and define the connected tangle basis number of G, denoted by $\theta'(G)$, as the maximum order of any connected tangle basis of G. An example of a connected tangle basis for the graph in Fig. 1 is given in Fig. 3. Notice that the number of separations of the connected tangle basis of Fig. 1 is lower than the number of separations of the tangle of Fig. 1 offered by Fig. 2 but still contains the essential members of the tangle. Below is a min-max theorem relationship between tangle bases and branchwidth.

Theorem 3 (Hicks [33]) *If hypergraph G is connected such that $\beta(G) \geq \gamma(G)$, then the tangle basis number $\theta'(G)$ is equal to the $\beta(G)$.* □

Constructing Branch Decompositions

In terms of finding branch decompositions for general graphs, there is an algorithm in Robertson and Seymour [51] to approximate the branchwidth of a graph within a factor of 3. For example, the algorithm decides if a graph has branchwidth at least 10 or finds a branch decomposition with width at most 30. This algorithm has not been used in a practical implementation and its improvements by Bodlaender [8], Bodlaender and Kloks [13], and Reed [46] have not been shown to be practical either. Bodlaender and Thilikos [16] presented a tree-decomposition-based linear time algorithm for finding an optimal branch decomposition but it appears to be impractical. Tree-decomposition-based algorithms are discussed in a later section. In addition, Bodlaender and Thilikos [17] gave an algorithm to compute the optimal branch decomposition for any chordal graph with maximum clique size at most 4 but the algorithm has been only shown practical for a particular type of 3-tree.

Under practical algorithms, Kloks et al. [39] gave a polynomial time algorithm to compute the branchwidth of interval graphs, but for general graphs, one

has to rely on heuristics. Cook and Seymour [20,21] gave a heuristic algorithm to produce branch decompositions that shows promise. In addition, Hicks [30,31] also found another branchwidth heuristic that was comparable to the algorithm of Cook and Seymour. Recently, Tamaki [54] has presented a linear time heuristic for constructing branch decompositions of planar graphs. This algorithm performs well when compared to the heuristics of Cook and Seymour [21] and Hicks [31]. Recently, Hicks [33] has developed a branch-decomposition-based algorithm for constructing optimal branch decompositions and it seems to be practical for sparse graphs with branchwidth at most 8.

For planar graphs, Seymour and Thomas showed that the branchwidth and an optimal branch decomposition of a graph can be computed in polynomial time. The complexity for the branchwidth is $O(n^3)$ and the complexity for computing an optimal branch decomposition is $O(n^4)$ [53]. Hicks [34,35] gave a practical implementation of these algorithms. Recently, Gu and Tamaki [28] introduced an $O(n^3)$ algorithm to compute an optimal branch decomposition of a planar graph by restricting the number of calls to the Seymour and Thomas algorithm for computing branchwidth to $O(n)$. More work in this area is encouraged to decrease the bound further.

Branch-Decomposition-Based Algorithms

Branch decompositions are of algorithmic importance for their appeal to solve intractable problems that can be modelled on graphs with bounded branchwidth. Courcelle [22] and Arnborg et al. [6] showed that several NP-complete problems can be solved in polynomial time using dynamic programming techniques on input graphs with bounded treewidth, discussed in a later section. Similar results have been obtained by Borie et al. [18]. The result is also equivalent to graphs with bounded branchwidth since the branchwidth and treewidth of a graph bound each other by constants [49]. In contrast, Seymour and Thomas [53] proved that testing if a general graph has branchwidth at most k, is NP-complete. The use of dynamic programming techniques in conjunction with a branch decomposition or a tree decomposition is referred to as a *branch-decomposition-based* or a *tree-decomposition-based* algorithm and these types of algorithms are part of the class of algorithms called *fixed parameter tractable algorithms* [1].

Some examples of branch-decomposition-based algorithms proposed in theory are Fomin and Thilikos [24] and Alekhnovich and Razborov [2]. Fomin and Thilikos used their result of improving a bound of Alon et al. [3] for the upper bound on the branchwidth of planar graphs to design a branch-decomposition-based algorithm in theory for vertex cover and dominating set for planar graphs [24]. Alekhnovich and Razborov [2] used the branchwidth of hypergraphs to design a branch-decomposition-based algorithm in theory to solve satisfiability problems.

Although theory indicates the fruitful potential of branch-decomposition-based algorithms, the number of branch-decomposition-based algorithms in the literature is exiguous. One noted exception is the work of Cook and Seymour [21] who produced the best known solutions for the 12 unsolved problems in TSPLIB95, a library of standard test instances for the TSP [47]. Hicks also presented a practical branch-decomposition-based algorithm for general minor containment [32] and constructing optimal branch decompositions [33]. One is also referred to the work of Christian [19].

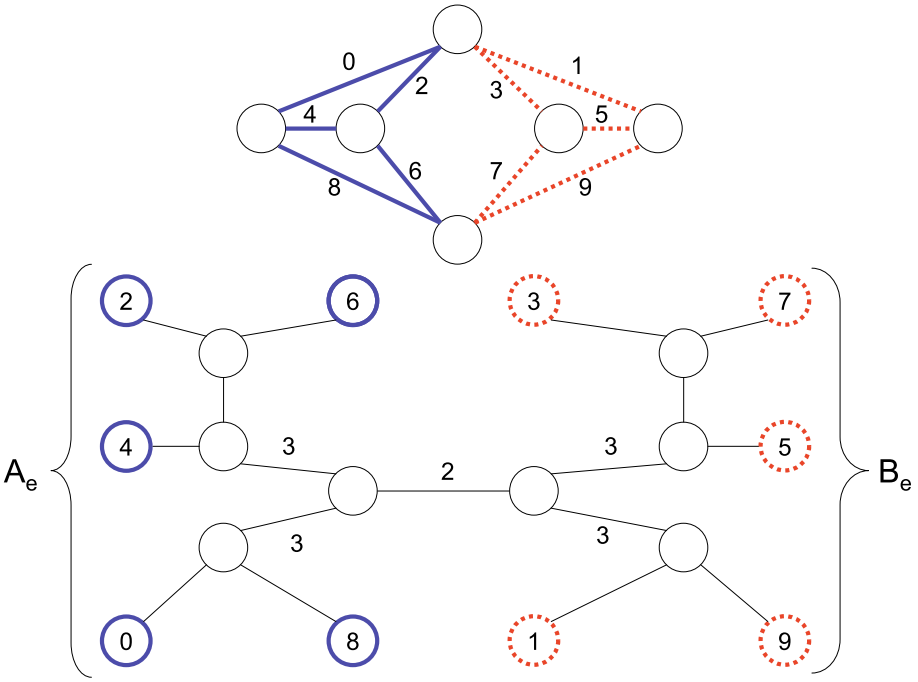
Branchwidth of Matroids

Since graph theory and matroid theory have a symbiotic relationship, it is only natural that branch decompositions can be extended to matroids. In fact, branch decompositions have been used to produce a matroid analogue of the graph minors theorem [26]. A formal definition for the branchwidth of a matroid is given below.

The reader is referred to the book by Oxley [43] if not familiar with matroid theory. Let M be a matroid with finite ground set $S(M)$ and rank function ρ. The rank function of M^*, the dual of M, is denoted ρ^*.

A *separation* (A, B) of a matroid M is a pair of complementary subsets of $S(M)$ and the order of the separation, denoted $\rho(M, A, B)$, is defined to be following:

$$\rho(M, A, B) = \begin{cases} \rho(A) + \rho(B) - \rho(M) + 1 & \text{if } A \neq \emptyset \\ & \neq B, \\ 0 & \text{else}, \end{cases}$$



Branchwidth and Branch Decompositions, Figure 4
Example Graph *G* from Fig. 1 with Optimal Branch Decomposition (T, μ) with width 3 for its Cycle Matroid $M(G)$

A *branch decomposition* of a matroid M is a pair (T, μ) where T is a tree having $| S(M) |$ leaves in which every non-leaf node has degree 3 and μ is a bijection from the ground set of M to the leaves of T. Notice that removing an edge, say e, of T partitions the leaves of T and the ground set of M into two subsets A_e and B_e. The *order* of e and of (A_e, B_e), denoted *order*(e) or *order*(A_e, B_e), is equal to $\rho(M, A_e, B_e)$. The *width* of a branch decomposition (T, μ) is the maximum order of all edges in T. The *branchwidth* of M, denoted by $\beta(M)$, is the minimum width over all branch decompositions of M. A branch decomposition of M is *optimal* if its width is equal to the branchwidth of M. The cycle matroid of graph G, denoted $M(G)$, has $E(G)$ as its ground set and the cycles of G as the cycles of $M(G)$. For example, Fig. 4 gives an optimal branch decomposition of the cycle matroid of the example graph given Fig. 1 where all of the orders for the edges of the branch decomposition are provided.

There is also a corresponding notion of a tangle and tangle number for matroids, provided by Dharmatilake [23]. In addition, Dharmatilake gave a min-max relationship between tangles of matroids and the branchwidth of matroids, given below.

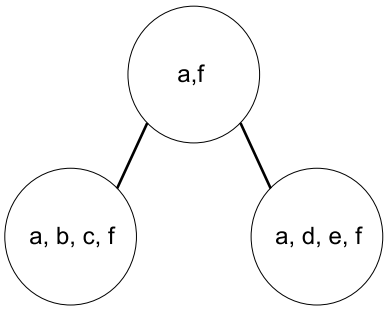
Theorem 4 (Dharmatilake [23]) *Let M be a matroid. Then $\beta(M) = \theta(M)$ if and only if M has no coloop and $\beta(M) \neq 1$.* □

It was posed by Robertson and Seymour [49] that the branchwidth of a graph and the branchwidth of the graph's cycle matroid are equivalent if the graph has a cycle of length at least 2. Recently, this conjecture was proved in the positive by Hicks and McMurray [37]. One is also referred to the work of Geelen et al. [26], Geelen et al. [25], Hall et al. [29], and Hliněný [38] for more detailed discussions on the branchwidth of matroids.

Treewidth and Tree Decompositions

This text would be remiss if a definition for treewidth and tree decompositions were not given.

A *tree decomposition* of a graph (or hypergraph) G is a pair, (T, σ), where T is a tree and for $t \in V(T)$, $\sigma(t)$ is a subset of $V(G)$ with the following properties:



Branchwidth and Branch Decompositions, Figure 5
Optimal Tree Decomposition (T, σ) of Example Graph in Fig. 1 with width 3

- $\bigcup_{t \in V(T)} \sigma(t) = V(G)$
- $\forall e \in E(G), \exists t \in V(T)$ such that the ends of e are contained in $\sigma(t)$
- for $t, t', t'' \in V(T)$, if t' is on the path of T between t and t'' then $\sigma(t) \cap \sigma(t'') \subseteq \sigma(t')$.

The *width* of a tree decomposition is the largest value of $|\sigma(t) - 1|$ over all nodes $t \in V(T)$. The *treewidth* of a graph G, denoted by $\tau(M)$, is the minimum width over all tree decompositions of G. A tree decomposition of G is called *optimal* if its width is equal to the treewidth of G. For example, Fig. 5 gives an optimal tree decomposition of the example graph in Fig. 1. If T is restricted to be a path then (T, σ) is called a *path decomposition* and its corresponding connectivity invariant for a graph G is called the *pathwidth* of G.

The relationship between branchwidth and treewidth is characterized in the following theorem.

Theorem 5 (Robertson and Seymour [49]) *For any hypergraph* $G, max(\beta(G), \gamma(G)) \leq \tau(G) + 1 \leq max(\lfloor \frac{3}{2}\beta(G) \rfloor, \gamma(G), 1)$. □

Tree decompositions and the associated connectivity invariant, treewidth, have been extensively researched by Thomas [56], Seymour and Thomas [52], Bodlaender [8,10], Bodlaender and Kloks [12,13], Bodlaender et al. [11], Bodlaender et al. [15], Bodlaender et al. [14], Ramachandramurthi [44], Reed [45,46] and many others (see the survey papers by Bodlaender [7,9]). One is also referred to the work of Koster et al. [40], Koster et al. [41], Telle and Proskurowski [55], and Alber and Neidermeier [1] for literature related to tree decompositions and tree-decomposition-based algorithms. In addition, one is referred to Hicks et al. [36] for a more thorough survey of branch and tree decomposition techniques related to optimization.

References

1. Alber J, Niedermeier R (2002) Improved tree decomposition based algorithms for domination-like problems. In: Proceedings of the 5th Latin American Theoretical Informatics (LATIN 2002). Lecture Notes in Computer Science, vol 2286. Springer, Heidelberg, pp 613–627
2. Alekhnovich M, Razborov A (2002) Satisfiability, branchwidth and tseitin tautologies. In: 43rd Annual IEEE Symposium on Foundations of Computer Science. IEEE Computer Society, pp 593–603
3. Alon N, Seymour PD, Thomas R (1994) Planar separators. SIAM J Discret Math 7:184–193
4. Archdeacon D (1980) A Kuratowski Theorem for the Projective Plane. PhD thesis, Ohio State University
5. Archdeacon D, Huneke P (1989) A Kuratowski theorem for non-orientable surfaces. J Combin Theory Ser B 46(2):173–231
6. Arnborg S, Lagergren J, Seese D (1991) Easy problems for tree-decomposable graphs. J Algorithms 12:308–340
7. Bodlaender H (1993) A tourist guide through treewidth. Acta Cybernetica 11:1–21
8. Bodlaender H (1996) A linear time algorithm for finding tree-decompositions of small treewidth. SIAM J Comput 25:1305–1317
9. Bodlaender H (1997) Treewidth: Algorithmic techniques and results. In: Privara I, Rvzicka P (eds) Proceedings of the 22nd International Symposium on Mathematical Foundations of Computer Science, MFCS'97. Lecture Notes in Computer Science, vol 1295. Springer, Berlin, pp 29–36
10. Bodlaender H (1998) A partial k-arboretum of graphs with bounded treewidth. Theoret Comput Sci 209:1–45
11. Bodlaender H, Gilbert J, Hafsteinsson H, Kloks T (1992) Approximation treewidth, pathwidth, and minimum elimination tree height. In: Schmidt G, Berghammer R (eds) Proceedings 17th International Workshop on Graph-Theoretic Concepts in Computer Science WG 1991. Lecture Notes in Computer Science, vol 570. Springer, Berlin, pp 1–12
12. Bodlaender H, Kloks T (1992) Approximating treewidth and pathwidth of some classes of perfect graphs. In: Proceedings Third International Symposium on Algorithms and Computation, ISAAC 1992. Lecture Notes in Computer Science, vol 650. Springer, Berlin, pp 116–125
13. Bodlaender H, Kloks T (1996) Efficient and constructive algorithms for the pathwidth and treewidth of graphs. J Algorithms 21:358–402
14. Bodlaender H, Kloks T, Kratsch D, Muller H (1998) Treewidth and minimum fill-in on d-trapezoid graphs. J Graph Algorithms Appl 2(5):1–23
15. Bodlaender H, Tan R, Thilikos D, van Leeuwen J (1997) On interval routing schemes and treewidth. Inf Comput 139:92–109

16. Bodlaender H, Thilikos D (1997) Constructive linear time algorithms for branchwidth. In: Degano P, Gorrieri R, Marchetti-Spaccamela A (eds) Lecture Notes in Computer Science: Proceedings of the 24th International Colloquium on Automata, Languages, and Programming. Springer, Berlin, pp 627–637
17. Bodlaender H, Thilikos D (1999) Graphs with branchwidth at most three. J Algorithms 32:167–194
18. Borie RB, Parker RG, Tovey CA (1992) Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. Algorithmica 7:555–581
19. Christian WA (2003) Linear-Time Algorithms for Graphs with Bounded Branchwidth. PhD thesis, Rice University
20. Cook W, Seymour PD (1994) An algorithm for the ring-router problem. Technical report, Bellcore
21. Cook W, Seymour PD (2003) Tour merging via branch-decomposition. INFORMS J Comput 15(3):233–248
22. Courcelle B (1990) The monadic second-order logic of graphs I: Recognizable sets of finite graphs. Inf Comput 85:12–75
23. Dharmatilake JS (1996) A min-max theorem using matroid separations. Contemp Math 197:333–342
24. Fomin F, Thilikos D (2003) Dominating sets in planar graphs: Branch-width and exponential speed-up. In: Proceedings of the Fourthteenth Annual ACM-SIAM Symposium on Discrete Algorithms (Baltimore, MD 2003). ACM, New York, pp 168–177
25. Geelen JF, Gerards AMH, Robertson N, Whittle GP (2003) On the excluded minors for the matroids of branch-width k. J Combin Theory Ser B 88:261–265
26. Geelen JF, Gerards AMH, Whittle G (2002) Branch width and well-quasi-ordering in matroids and graphs. J Combin Theory Ser B 84:270–290
27. Glover H, Huneke P, Wang CS (1979) 103 graphs that are irreducible for the projective plane. J Combin Theory Ser B 27:332–370
28. Gu QP, Tamaki H (2005) Optimal branch-decomposition of planar graphs in $o(n^3)$ time. In: Proceedings of the 31st International Colloquium on Automata, Languages and Programming. LNCS, vol 3580, pp 373–384
29. Hall R, Oxley J, Semple C, Whittle G (2002) On matroids of branch-width three. J Combin Theory Ser B 86:148–171
30. Hicks IV (2000) Branch Decompositions and their Applications. PhD thesis, Rice University
31. Hicks IV (2002) Branchwidth heuristics. Congressus Numerantium 159:31–50
32. Hicks IV (2004) Branch decompositions and minor containment. Networks 43(1):1–9
33. Hicks IV (2005) Graphs, branchwidth, and tangles! oh my! Networks 45:55–60
34. Hicks IV (2005) Planar branch decompositions I: The ratcatcher. INFORMS J Comput 17(4):402–412
35. Hicks IV (2005) Planar branch decompositions II: The cycle method. INFORMS J Comput 17(4):413–421
36. Hicks IV, Koster AMCA, Kolotoğlu E (2005) Branch and tree decomposition techniques for discrete optimization. In: Cole Smith J (ed) Tutorials in Operations Research 2005. INFORMS, Hanover, MD, pp 1–29
37. Hicks IV, McMurray N (2007) The branchwidth of graphs and their cycle matroids. J Combin Theory Ser B 97: 681–692
38. Hliněný P (2002) On the exclued minors for matroids of branch-width three. preprint
39. Kloks T, Kratochvil J, Müller H (1999) New branchwidth territories. In: Meinel C, Tison S (eds) STAC'99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 1999 Proceedings. Springer, Berlin, pp 173–183
40. Koster A, van Hoesel S, Kolen A (2002) Solving partial constraint satisfaction problems with tree-decompositions. Networks 40:170–180
41. Koster AMCA, Bodlaender HL, van Hoesel SPM (2001) Treewidth: Computational experiments. Electr Notes Discret Math 8:54–57
42. Kuratowski K (1930) Sur le probleme des courbes gauches en topologie. Fundamenta Mathematicae 15:271–283
43. Oxley JG (1992) Matroid Theory. Oxford University Press, Oxford
44. Ramachandramurthi S (1997) The structure and number of obstructions to treewidth. SIAM J Discret Math 10:146–157
45. Reed B (1992) Finding approximate separators and computing tree width quickly. In: Proceeding of the 24th Annual Association for Computing Machinery Symposium on Theory of Computing. ACM Press, New York, pp 221–228
46. Reed B (1997) Tree width and tangles: A new connectivity measure and some applications. In: Bailey RA (ed) Survey in Combinatorics. Cambridge University Press, Cambridge, pp 87–162
47. Reinelt G (1991) TSPLIB – a traveling salesman library. ORSA J Comput 3:376–384
48. Robertson N, Seymour PD (1985) Graph minors: A survey. In: Surveys in Combinatorics, London Math Society Lecture Note Series, edition 103. Cambridge University Press, Cambridge, pp 153–171
49. Robertson N, Seymour PD (1991) Graph minors X: Obstructions to tree-decompositions. J Combin Theory Ser B 52:153–190
50. Robertson N, Seymour PD (1994) Graph minors XI: Circuits on a surface. J Combin Theory Ser B 60:72–106
51. Robertson N, Seymour PD (1995) Graph minors XIII: The disjoint paths problem. J Combin Theory Ser B 63:65–110
52. Seymour P, Thomas R (1993) Graph searching and a min-max theorem for tree-width. J Combin Theory Ser B 58:22–33
53. Seymour PD, Thomas R (1994) Call routing and the ratcatcher. Combinatorica 14(2):217–241
54. Tamaki H (2003) A linear time heuristic for the branch-decomposition of planar graphs. Technical Report MPI-I-2003-1-010, Max-Planck-Institut fuer Informatik

55. Telle JA, Proskurowski A (1997) Algorithms for vertex partitioning problems on partial *k*-trees. SIAM J Discret Math 10(4):529–550
56. Thomas R (1990) A Menger-like property of tree-width: The finite case. J Combin Theory Ser B 48:67–76
57. Wagner K (1937) Uber eine eigenschaft der ebenen komplexe. Math Annal 115:570–590

Broadcast Scheduling Problem

CLAYTON W. COMMANDER
Air Force Research Laboratory, Munitions Directorate, and Dept. of Industrial and Systems Engineering, University of Florida, Gainesville, USA

Article Outline

Synonyms
Introduction
  Organization
  Idiosyncrasies
Formulation
Methods
  Sequential Vertex Coloring
  Mixed Neural-Genetic Algorithm
  Greedy Randomized Adaptive Search Procedures (GRASP)
  Multi-start Combinatorial Algorithm
  Computational Effectiveness
Conclusion
See also
References

Synonyms

BSP; The BROADCAST SCHEDULING PROBLEM is also referred to as the TDMA MESSAGE SCHEDULING PROBLEM [6]

Introduction

Wireless mesh networks (WMNs) have become an important means of communication in recent years. In these networks, a shared radio channel is used in conjunction with a packet switching protocol to provide high-speed communication between many potentially mobile users. The stations in the network act as transmitters and receivers, and are thus capable of utilizing a multi-hop transmission procedure. The advantage of this is that several stations can be used as relays to forward messages to the intended recipient. This allows beyond line of sight communication between stations which are geographically disbursed and potentially mobile [2].

Mesh networks have increased in popularity in recent years and the number of applications is steadily increasing [25]. As mentioned in [1], WMNs allow users to integrate various networks, such as Wi-Fi, the internet and cellular systems. WMNs can also be utilized in a military setting in which tactical datalinks network various communication, intelligence, and weapon systems allowing for streamlined communication between several different entities [6]. For a survey of wireless mesh networks, the reader is referred to [1].

In WMNs, the critical problem involves efficiently utilizing the available bandwidth to provide collision free message transmissions. Unfettered transmission by the network stations over the shared channel will lead to message collisions. Therefore, some medium access control (MAC) scheme should be employed to schedule message transmissions so as to avoid message collisions. The time division multiple access (TDMA) protocol is a MAC scheme introduced by Kleinrock in 1987 which was shown to provide collision free broadcast schedules [19]. In a TDMA network, time is divided into frames with each frame consisting of a number of unit length slots in which the messages are scheduled. Stations scheduled in the same slot broadcast simultaneously. Thus, the goal is to schedule as many stations as possible in the same slot so long as there are no message collisions.

When considering the broadcast scheduling problem on TDMA networks, there are two optimization problems which must be addressed [31]. The first involves finding the minimum frame length, or the number of slots required to schedule all stations at least once. The second problem is that of maximizing the number of stations scheduled within each slot, thus maximizing the throughput. Both of these problems however, are known to be $\mathcal{NP}$-hard [2]. Therefore, efficient heuristics are typically used to quickly provide high quality solutions to real-world instances.

Organization

The organization of this article is as follows. In the following section, we formally define the problem statement and provide a mathematical programming for-

mulation. We also examine the computational complexity the problem. In Sect. "Methods", we review several solution techniques which appear in the literature. We provide some concluding remarks in Sect. "Conclusion" and indicate directions of future research. Finally, a list of cross references is provided in Sect. "See also".

Idiosyncrasies

We will now briefly introduce some of the symbols and notations we will employ throughout this paper. Denote a graph $G = (V, E)$ as a pair consisting of a set of vertices V, and a set of edges E. All graphs in this paper are assumed to be undirected and unweighted. We use the symbol "$b := a$" to mean "the expression a defines the (new) symbol b" in the sense of King [18]. Of course, this could be conveniently extended so that a statement like "$(1 - \epsilon)/2 := 7$" means "define the symbol ε so that $(1 - \epsilon)/2 = 7$ holds". Finally, we will use *italics* for emphasis and SMALL CAPS for problem names. Any other locally used terms and symbols will be defined in the sections in which they appear.

Formulation

A TDMA network can be conveniently described as a graph $G = (V, E)$ where the vertex set V represents the stations and the set of edges E represents the set of communication links between adjacent stations. There are two types of message collisions which must be avoided when scheduling messages in TDMA networks. The first, called a *direct collision* occurs between *one-hop neighboring stations*, or those stations $i, j \in V$ such that $(i, j) \in E$. One-hop neighbors which broadcast during the same slot cause a direct collision. Further, if $(i, j) \notin E$, but $(i, k) \in E$ and $(j, k) \in E$, then i and j are called *two-hop neighbors*. Two-hop neighbors transmitting in the same slot cause a so-called *hidden collision* [2].

Assume that there are M slots per frame. Further, assume that packets are sent at the beginning of each time slot and are received in the same slot in which they are sent. Let $x\colon M \times V \mapsto \{0, 1\}$, be a surjection defined by

$$x_{mn} := \begin{cases} 1, \text{if station } n \text{ scheduled in slot } m\,, \\ 0, \text{otherwise}\,. \end{cases} \tag{1}$$

Also, let $c\colon E \mapsto \{0, 1\}$ return 1 if i and j are one-hop neighbors, i. e., if $(i, j) \in E$ and $i \neq j$.

Using the aforementioned definitions and assumptions, we can now formulate the BROADCAST SCHEDULING PROBLEM (BSP) on TDMA networks as the following multiobjective optimization problem:

$$\text{Minimize} M$$

$$\text{Maximize} \sum_{i=1}^{M} \sum_{j=1}^{|V|} x_{ij}$$

subject to:

$$\sum_{m=1}^{M} x_{mn} \geq 1, \quad \forall n \in V\,, \tag{2}$$

$$c_{ij} + x_{mi} + x_{mj} \leq 2, \quad \forall i, j \in V, i \neq j, m = 1, \ldots, M\,, \tag{3}$$

$$c_{ik} x_{mi} + c_{kj} x_{mj} \leq 1, \forall i, j, k \in V, i \neq j, j \neq k, k \neq i, m = 1, \ldots, M\,, \tag{4}$$

$$x_{mn} \in \{0, 1\}, \quad \forall n \in V, m = 1, \ldots, M\,, \tag{5}$$

$$M \in \mathbb{Z}^+\,. \tag{6}$$

The objective provides a minimum frame length with maximum bandwidth utilization, while constraint (2) ensures that all stations broadcast at least once. Constraints (3) and (4) prevent direct and hidden collisions, respectively. Constraints (5) and (6) define the proper domain of the decision variables.

Suppose that we relax the BSP and only the consider the first objective function. This is referred to as the FRAME LENGTH MINIMIZATION PROBLEM (FLMP) and is given by the following integer program: $\min\{M\colon (2) - (6)\}$. Clearly any feasible solution to this problem is feasible for BSP. Now, consider a graph $G' = (V, E')$ where V follows from the original communication graph G, but whose edge set is given by $E' = E \cup \{(i, j)\colon i, j \text{ are two-hop neighbors}\}$. Then using this augmented graph, we can formulate the following theorem due to Butenko et al. [2].

Theorem 1 *The* FRAME LENGTH MINIMIZATION PROBLEM *on* $G = (V, E)$ *is equivalent to finding an optimal coloring of the vertices of* $G'(V, E')$.

Proof Recall that in order for a message schedule to be feasible, all stations must broadcast at least once and no collisions occur, either hidden or direct. Notice now that E' contains both one-hop and two-hop neighbors, and in any feasible solution, neither of these can transmit in the same slot. Thus, there is a one-to-one correlation between time slots in G and vertex colors in G'. Hence, a minimum coloring of the vertices of G' provides the minimum required slots needed for a collision free broadcast schedule on G. □

After one has successfully solved the FLMP by solving the corresponding GRAPH COLORING PROBLEM, an optimal frame length M^* is attained. With this, the THROUGHPUT MAXIMIZATION PROBLEM (TMP) given as follows $\max\{\sum_{i=1}^{M^*}\sum_{j=1}^{|V|} x_{ij} : (2)-(6)\}$ can be solved, where M is replaced by M^* in $(2)-(6)$. A direct result of Theorem 1 is that finding an optimal frame length for a general instance of the BSP is $\mathcal{NP}$-hard [11]. The reader is referred to the paper by Butenko et al. [2] for the complete proof. Also, in [8], the TMP was also shown to be $\mathcal{NP}$-hard [8]. Thus it is unlikely that a polynomial algorithm exists for finding an optimal broadcast schedule for an instance of the BSP [11]. It is interesting to note however, that if we ignore constraint (4) which prevents two-hop neighbors from transmitting simultaneously, then the resulting problem is in $\mathcal{P}$, and a polynomial time algorithm is provided in [13].

Due to the computational complexity of the BSP, several heuristics have been applied and appear throughout the literature [2,3,6,28,31]. In the following section, we highlight several of these methods and examine their effectiveness when applied to large-scale instances.

Methods

In this section, we review many of the heuristics which have been applied to the BSP. We analyze the techniques used and compare their relative performance as reported in [6]. The particular algorithms we examine are as follows:

- Sequential vertex coloring [31];
- Mixed neural-genetic algorithm [27];
- Greedy randomized adaptive search procedures (GRASP) [2,3];
- A multi-start combinatorial algorithm [6].

We note here that none of the heuristics which we describe in this section attempt to solve the BSP by using the typical multiobjective optimization approach, in which one combines the multiple objectives into one scalar objective whose optimal value is a Pareto optimal solution to the original problem. Instead all of the methods decouple the objectives and handle each independently. This is done because for instances of the BSP, frame length minimization usually takes precedence over the utilization maximization problem [27,28,31].

Sequential Vertex Coloring

Yeo et al. [31] propose a two-phase approach based on sequential vertex coloring (SVC). The first phase computes an approximate solution for the FLMP. Then using the computed frame length, the TMP is considered in the second phase. Specific details are as follows.

Frame Length Minimization For this phase, the FRAME LENGTH MINIMIZATION PROBLEM is considered and an approximate solution is computed by solving a graph coloring problem in the augmented graph. A sequential vertex ordering approach is used whereby the stations are first ordered in descending order of the number of one-hop and two-hop neighbors. The first vertex is colored and the list of the other $N-1$ vertices are scanned downward. The remaining vertices are colored with the smallest color which has not already been assigned to one of its one-hop neighboring station. The process is continued until all vertices have been colored.

Throughput Maximization To solve the TMP in the frame length computed in phase 1, an ordering method of the sequential vertex coloring algorithm is applied. The stations are now ordered in ascending order of the the number of one-hop and two-hop neighbors. The first ordered station is then assigned to any slots in which it can simultaneously broadcast with the previously assigned stations. This process is repeated for every station in the ordered list.

Mixed Neural-Genetic Algorithm

As with the coloring heursitic presented described above, Salcedo-Sanz et al. [27] introduced a two-phase

heuristic based on combining both Hopfield neural networks [15] and genetic algorithms as in [29]. As with the vertex coloring algorithm, phase one considers the FLMP and phase two attempts to maximize the throughput.

Frame Length Minimization In order to solve the FRAME LENGTH MINIMIZATION PROBLEM, a discrete-time binary Hopfield neural network (HNN) is used. As described in [27], the HNN can be represented as a graph whose vertices are the neurons (stations) and whose edges represent the direct collisions. The neurons are updated one at a time after a randomized initialization until the system converges. For specific implementation details, the reader should see [27].

Utilization Maximization In this phase, a genetic algorithm [12] is used to maximize the throughput within the frame length that was determined in phase one. Genetic algorithms (GAs) get their names from the biological process which they mimic. Motivated by Darwin's Theory of Natural Selection [7], these algorithms evolve a *population* of solutions, called *individuals*, over several *generations* until the best solution is eventually reached. Each component of an individual is called a *allele*. Individuals in the population mate through a process called *crossover*, and new solutions having traits, i. e. alleles of both parents are produced. In successive generations, only those solutions having the best *fitness* are carried to the next generation in a process which mimics the fundamental principle of natural selection, *survival of the fittest* [12]. Again, the reader should reference [27] for implementation specific information.

Greedy Randomized Adaptive Search Procedures (GRASP)

GRASP [9] is a multi-start metaheuristic that has been used with great success to provide solutions for several difficult combinatorial optimization problems [10], including SATISFIABILITY [24], QUADRATIC ASSIGNMENT [21,23], and most recently the COOPERATIVE COMMUNICATION PROBLEM ON AD-HOC NETWORKS [4,5].

GRASP is a two-phase procedure which generates solutions through the controlled use of random sampling, greedy selection, and local search. For a given problem Π, let F be the set of feasible solutions for Π. Each solution $X \in F$ is composed of k discrete components $a_1, \ldots, a_k$. GRASP constructs a sequence $\{X\}_i$ of solutions for Π, such that each $X_i \in F$. The algorithm returns the best solution found after all iterations.

Construction Phase The construction phase for the GRASP constructs a solution iteratively from a partial broadcast schedule which is initially empty. The stations are first sorted in descending order of the number of one-hop and two-hop neighbors. Next, a so-called *Restricted Candidate List* (RCL) is created and consists of the stations which may broadcast simultaneously with the stations previously assigned to the current slot. From this RCL a station is randomly chosen and assigned. A new RCL is created and another station is randomly selected. This process continues the RCL is empty, at which time the slot number is incremented and the procedure is repeated recursively for the subgraph induced by the set of all vertices whose corresponding stations have not yet been assigned to a time slot.

Local Search The local search phase used is a swap-based procedure which is adapted from a similar method for graph coloring implemented by Laguna and Martí in [20]. First, the two slots with the fewest number of scheduled transmissions are cif stationombined and the total number of slots is now given as $k = m-1$, where m is the frame length of the schedule computed in the construction phase. Denote the new broadcast schedule as $\{x_{m',n}, m' = 1, \ldots, k, n = 1, \ldots, N\}$. Now, let the function $f(x) = \sum_{i=1}^{k} E(m'_i)$, where $E(m'_i)$ is the set of collisions in slot m'_i. $f(x)$ is then minimized by the application of a local search procedure as follows.

A colliding station in the combined slot is chosen randomly and every attempt is made to swap this station with another from the remaining $k-1$ slots. After a swap is made, $f(x)$ is re-evaluated. If $f(x)$ has a lower value than before the swap, the swap is kept and the process repeated with the remaining colliding stations. If after every attempt to swap a colliding station the result is unimproved, a new colliding station is chosen and the swap routine is attempted. This continues until either a successful swap is made or for some specified number of iterations. If a solution is improved

such that $f(x) = 0$, then the frame length has been successfully decreased by one slot. The value of k is then decremented and the process is repeated.If the procedure ends with $f(x) > 0$, then no improved solution was found.

Multi-start Combinatorial Algorithm

To our knowledge, the most recent heuristic for the BSP is a hybrid multi-start method by Commander and Pardalos [6]. This heuristic combines a graph coloring heuristic with a randomized local search to provide high-quality solutions for large-scale instances on the problem. As with the previously described method, this heuristic is also a two-phase approach. The reader should see [6] for pseudo-code and other implementation specific details.

Frame Length Minimization First a greedy randomized construction heuristic was used to determine the value for M. As a result of Theorem 1, the method is based on the construction phase of the Greedy Randomized Adaptive Search Procedure (GRASP) [26] for coloring sparse graphs proposed by Laguna and Martí in [20]. This particular method was chosen because it is able to quickly provide excellent solutions for the frame length. That being said, any other coloring heuristic would provide a value for M such as the Sequential Vertex Coloring method described above. However, the randomized approach of the selected method allows the search space to be more thoroughly investigated. This is due to the fact that different optimal colorings will yield different solutions in the second phase.

Throughput Maximization The solution from the first phase will not provide an optimal throughput in general, because each station will only be scheduled to transmit once in the frame. Therefore, a randomized local improvement method is used to schedule each station as many times as possible in the frame. This method locally optimizes each slot by considering the set of nodes which may transmit with the currently scheduled slot. A node from this set is randomly selected and the process repeats until no other stations may broadcast in the current slot. The next slot is then considered and the process is repeated until the solution is locally optimal.

Computational Effectiveness

In [6], the authors performed an extensive computational experiment comparing the effectiveness of the aforementioned heuristics. They tested all of the algorithms on a common platform and reported solutions for 63 instances ranging from 15 to 100 stations with varying densities. In addition, they implemented the integer programming model from Sect. "Formulation" using the Xpress-MP™ optimization suite from Dash Optimization [17]. Xpress-MP contains an implementation of the simplex method [14], and uses a branch and bound algorithm [30] together with advanced cutting-plane techniques [16,22].

For each instance tested, the combinatorial algorithm of [6] is superior to the other heuristics mentioned. For all 63 instances tested, the method found solutions at least as good as any of the other algorithms from the literature for all of the networks, outperforming them on 56 cases. The performance of the GRASP [2] and the Mixed Neural-Genetic Algorithm [27] were comparable, with GRASP performing slightly better on average. The weakest of the methods was the Sequential Vertex Coloring [31] algorithm. For specific numerical results, see [6].

Conclusion

In this article, we introduced the BROADCAST SCHEDULING PROBLEM on TDMA networks. The BSP is an important problem that occurs in wireless mesh networks regarding efficiently scheduling collision free broadcasts for the network stations. We formally defined the problem, examined the computational complexity, and discussed several algorithms which have been applied to the BSP, all with competitive results.

We conclude with a few words on possible directions of future research. In addition to the ones described, other metaheuristics could be considered and approximation algorithms developed. Also, a heuristic exploration of cutting plane algorithms on the IP formulation would be an interesting alternative. Another alternative would be to consider instances of the problem in which the stations are part of a mobile ad-hoc network. In this case, the topology of the network would change as the stations change position. This could potentially cause significant difficulties in determining the evolving sets of one-hop and two-hop

neighbors. There is no doubt that as technology advances and research on ad-hoc networks increases, so too will applications of the BSP which will require advanced solution techniques [25].

See also

- ▶ Frequency Assignment Problem
- ▶ Genetic Algorithms
- ▶ Graph Coloring
- ▶ Greedy Randomized Adaptive Search Procedures
- ▶ Multi-objective Integer Linear Programming
- ▶ Optimization Problems in Unit-Disk Graphs
- ▶ Simulated Annealing

References

1. Akyildiz IF, Wang X, Wang W (2005) Wireless mesh networks: a survey. Comp Network 47(4):445–487
2. Commander CW, Butenko SI, Pardalos PM (2004) On the performance of heuristics for broadcast scheduling. In: Grundel D, Murphey R, Pardalos P (eds) Theory and Algorithms for Cooperative Systems. World Scientific, Singapore, pp 63–80
3. Commander CW, Butenko SI, Pardalos PM, Oliveira CAS (2004) Reactive grasp with path relinking for the broadcast scheduling problem. In: Proceedings of the 40th Annual International Telemetry Conference. pp 792–800
4. Commander CW, Festa P, Oliveira CAS, Pardalos PM, Resende MGC, Tsitselis M (2006) GRASP with path-relinking for the cooperative communication problem on ad-hoc networks. SIAM J Control Optim submitted
5. Commander CW, Oliveira CAS, Pardalos PM, Resende MGC (2005) A GRASP heuristic for the cooperative communication problem in ad hoc networks. In: Proceedings of the VI Metaheuristics International Conference. pp 225–230
6. Commander CW, Pardalos PM (2007) A combinatorial algorithm for the TDMA message scheduling problem. Comput Optim Appl to appear
7. Darwin C (1872) The Origin of Species, 6th edn. Murray, London
8. Ephremides A, Truong TV (1990) Scheduling broadcasts in multihop radio networks. IEEE Trans Commun 38(4):456–460
9. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Global Optim 6:109–133
10. Festa P, Resende MGC (2002) GRASP: An annotated bibliography. In: Ribeiro C, Hansen P (eds) Essays and surveys in metaheuristics, Kluwer, Dordrecht, pp 325–367
11. Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman WH and Company, New York
12. Goldberg DE (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Kluwer, Dordrecht
13. Hajek B, Sasaki G (1988) Link scheduling in polynomial time. IEEE Trans Inf Theor 34:910–918
14. Hillier FS, Lieberman GJ (2001) Introduction to Operations Research. McGraw Hill, New York
15. Hopfield JJ, Tank DW (1982) Neural networks and physical systems with emergent collective computational abilities. In: Proceedings of the National Academy of Science, pp 2541–2554
16. Horst R, Pardalos PM, Thoai NV (1995) Introduction to Global Optimization, volume 3 of Nonconvex Optimization and its Applications. Kluwer, Dordrecht
17. Dash Optimization Inc (2003) Xpress-Optimizer Reference Manual
18. King J (1994) Three problems in search of a measure. Am Math Mon 101:609–628
19. Kleinrock L, Silvester J (1987) Spatial reuse in multihop packet radio networks. In: Proceedings of the IEEE 75
20. Laguna M, Martí R (2001) A grasp for coloring sparse graphs. Comput Optim Appl 19(2):165–178
21. Li Y, Pardalos PM, Resende MGC (1994) A greedy randomized adaptive search procedure for the quadratic assignment problem. In: Pardalos PM, Wolkowicz H (eds) Quadratic Assignment and Related Problems, volume 16 of DIMACS Series on DIscrete Mathematics and Theoretical Computer Science, pp 237–261
22. Oliveira CAS, Pardalos PM, Querido TM (2005) A combinatorial algorithm for message scheduling on controller area networks. Int J Oper Res 1(1/2):160–171
23. Oliveira CAS, Pardalos PM, Resende MGC (2003) Grasp with path-relinking for the qap. In: 5th Metaheuristics International Conference, pp 57.1–57.6
24. Resende MGC, Feo TA (1996) A grasp for satisfiability. In: Johnson DS, Trick MA (eds) Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenges, volume 26, American Mathematical Society, Providence, RI, pp 499–520
25. Resende MGC, Pardalos PM (2006) Handbook of Optimization in Telecommunications. Springer, Berlin
26. Resende MGC, Ribeiro CC (2003) Greedy randomized adaptive search procedures. In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics. Kluwer, Dordrecht, pp 219–249
27. Salcedo-Sanz S, Busoño-Calzón C, Figueiral-Vidal AR (2003) A mixed neural-genetic algorithm for the broadcast scheduling problem. IEEE Trans Wirel Commun 2(2):277–283
28. Wang G, Ansari N (1997) Optimal broadcast scheduling in packet radio networks using mean field annealing. IEEE J Sel Areas Commun 15(2):250–260
29. Watanabe Y, Mizuguchi N, and Fujii Y (1998) Solving optimization problems by using a hopfield neural network and genetic algorithm combination. Syst Comput Jpn 29(10):68–73

30. Wolsey L (1998) Integer Programming. Wiley, New York
31. Yeo J, Lee H, and Kim S (2002) An efficient broadcast scheduling algorithm for TDMA ad-hoc networks. Comput Oper Res 29:1793–1806

Broyden Family of Methods and the BFGS Update

BFM

VASSILIOS S. VASSILIADIS, RAÚL CONEJEROS
Chemical Engineering Department,
University Cambridge, Cambridge, UK

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Unconstrained optimization; BFGS update; DFP update; Broyden family of methods; Rank-two updates; Quasi-Newton methods

Quasi-Newton methods attempt to update a Hessian approximation (or the inverse of it) instead of evaluating the Hessian matrix exactly at each iteration, as in the basic Newton method for *unconstrained optimization*. Consider the optimization problem:

$$\min_{\mathbf{x}} f(\mathbf{x}).$$

For this problem the Newton method requires the solution and updating iteratively of the solution point according to:

$$\mathbf{H}(x^{(k)})\Delta x^{(k)} = -\mathbf{g}(x^{(k)}), \tag{1}$$

where $\mathbf{H}(x^{(k)})$ denotes the Hessian matrix at point $x^{(k)}$ (kth iteration of Newton's method), $\mathbf{g}(x^{(k)})$ is the gradient vector at the same point, and finally $\Delta x^{(k)}$ is the correction to the point $x^{(k)}$. The correction is applied according to:

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x^{(k)}$$

where for the standard Newton method $\alpha = 1$, but otherwise in practical applications and to force theoretically 'global convergence' (not just in the neighborhood of the minimizer) one conducts a line search to estimate optimally the value of α at each iteration. Alternative algorithms use the concept of *trust regions*.

There exist symmetric updating formulae of rank-two corrections for both the inverse Hessian and the Hessian, all belonging to the broad category of *Broyden methods*. The general family updates either the Hessian ($\mathbf{H}$) or the inverse Hessian ($\mathbf{G} = \mathbf{H}^{-1}$). There are two well-known schemes, the *Davidon–Fletcher–Powell rank-two update* (DFP update), originally proposed by W.C. Davidon [3], and later by R. Fletcher and M.J.D. Powell [6], and the well-known *Broyden–Fletcher–Goldfarb–Shanno update formula* (BFGS update). This was proposed by C.G. Broyden [1,2], Fletcher [4], D. Goldfarb [7], and D.F. Shanno [9]. Both of these methods preserve positive definiteness of the updated matrices.

The definitions of $\mathbf{p}$ and $\mathbf{q}$ used below are introduced first:

$$\mathbf{p}_k = \mathbf{x}_{k+1} - \mathbf{x}_k,$$
$$\mathbf{q}_k = \mathbf{g}_{k+1} - \mathbf{g}_k.$$

The DFP updating scheme of the inverse Hessian is given by:

$$\mathbf{G}_{k+1}^{\mathrm{DFP}} = \mathbf{G}_k + \frac{\mathbf{p}_k\mathbf{p}_k^\top}{\mathbf{p}_k^\top\mathbf{q}_k} - \frac{\mathbf{G}_k\mathbf{q}_k\mathbf{q}_k^\top\mathbf{G}_k}{\mathbf{q}_k^\top\mathbf{G}_k\mathbf{q}_k}.$$

The complementary updating formula to any updating Hessian (or inverse Hessian) scheme can be found by exchanging $\mathbf{G}$ with $\mathbf{H}$ and $\mathbf{q}$ with $\mathbf{p}$ (for example as discussed in [8]). By applying this property to the DFP update above, it is obtained:

$$\mathbf{H}_{k+1}^{\mathrm{BFGS}} = \mathbf{H}_k + \frac{\mathbf{q}_k\mathbf{q}_k^\top}{\mathbf{q}_k^\top\mathbf{p}_k} - \frac{\mathbf{H}_k\mathbf{p}_k\mathbf{p}_k^\top\mathbf{H}_k}{\mathbf{p}_k^\top\mathbf{H}_k\mathbf{p}_k},$$

which is the BFGS updating scheme for the Hessian.

By taking the inverse of this one can obtain the inverse Hessian BFGS updating formula:

$$\mathbf{G}_{k+1}^{\mathrm{BFGS}} = \mathbf{G}_k + \left(\frac{1 + \mathbf{q}_k^\top\mathbf{G}_k\mathbf{q}_k}{\mathbf{q}_k^\top\mathbf{p}_k}\right)\left(\frac{\mathbf{p}_k\mathbf{p}_k^\top}{\mathbf{p}_k^\top\mathbf{q}_k}\right) - \left(\frac{\mathbf{p}_k\mathbf{q}_k^\top\mathbf{G}_k + \mathbf{G}_k\mathbf{q}_k\mathbf{p}_k^\top}{\mathbf{q}_k^\top\mathbf{p}_k}\right).$$

The general class of Broyden methods can be derived by the linear combination of the two types of updates, since they are both symmetric rank-two type corrections, being constructed from the same vectors $\mathbf{p}_k$ and $\mathbf{G}_k\ \mathbf{q}_k$. Thus it can be obtained that (for example see [5,8]):

$$\mathbf{G}_{k+1}^{\phi} = (1-\phi)\mathbf{G}_{k+1}^{\mathrm{DFP}} + \mathbf{G}_{k+1}^{\mathrm{BFGS}},$$

which yields:

$$\mathbf{G}_{k+1}^{\phi} = \mathbf{G}_k + \frac{\mathbf{p}_k\mathbf{p}_k^{\top}}{\mathbf{p}_k^{\top}\mathbf{q}_k} - \frac{\mathbf{G}_k\mathbf{q}_k\mathbf{q}_k^{\top}\mathbf{G}_k}{\mathbf{q}_k^{\top}\mathbf{G}_k\mathbf{q}_k} + \phi\mathbf{v}_k\mathbf{v}_k^{\top}, \quad (2)$$

which is the general family of Broyden methods, with:

$$\mathbf{v}_k = \left(\mathbf{q}_k^{\top}\mathbf{G}_k\mathbf{q}_k\right)^{\frac{1}{2}}\left(\frac{\mathbf{p}_k}{\mathbf{p}_k^{\top}\mathbf{q}_k} - \frac{\mathbf{G}_k\mathbf{q}_k}{\mathbf{q}_k^{\top}\mathbf{G}_k\,\mathbf{q}_k}\right).$$

A pure Broyden method is one that uses a constant value of ϕ in all iterations. The Broyden family does not preserve positive definiteness of the updated inverse Hessian $\mathbf{G}_{k+1}^{\phi}$ for all values of ϕ.

Generally, of all these schemes the varying ϕ variant is never used nowadays (2000), with the BFGS scheme being the method of choice whenever an updating scheme is chosen. This is because computational experience has proven the BFGS to be more effective than the DFP scheme.

See also

- ▶ Conjugate-Gradient Methods
- ▶ Large Scale Unconstrained Optimization
- ▶ Numerical Methods for Unary Optimization
- ▶ Unconstrained Nonlinear Optimization: Newton–Cauchy Framework
- ▶ Unconstrained Optimization in Neural Network Training

References

1. Broyden CG (1970) The convergence of a class of double rank minimization algorithms. Part I. J Inst Math Appl 6:76–90
2. Broyden CG (1970) The convergence of a class of double rank minimization algorithms. Part II. J Inst Math Appl 6:222–231
3. Davidon WC (1959) Variable metric method for minimization. Techn Report AEC Res Developm ANL-5590
4. Fletcher R (1970) A new approach to variable metric algorithms. Comput J 13:317–322
5. Fletcher R (1991) Practical methods of optimization, 2nd edn. Wiley, New York
6. Fletcher R, Powell MJD (1963) A rapidly convergent descent method for minimization. Comput J 6:163–168
7. Goldfarb D (1970) A family of variable metric methods derived by variational means. Math Comput 24:23–26
8. Luenberger DG (1984) Linear and nonlinear programming, 2nd edn. Addison-Wesley, Reading, MA
9. Shanno DF (1970) Conditioning of quasi-Newton methods for function minimization. Math Comput 24:647–656

C

Capacitated Minimum Spanning Trees

STEFAN VOSS
Institute Wirtschaftswissenschaften,
Techn University Braunschweig,
Braunschweig, Germany

MSC2000: 90C27, 68T99

Article Outline

Keywords
Applications
Mathematical Programming Formulations
Exact Algorithms
Heuristics
  Finding Initial Feasible Solutions
  Construction Methods
  Savings Procedures
  Dual Procedures
  Additional Procedures
Improvement Procedures
  Neighborhood Definition
  Second Order Algorithms
  Computational Results
Metaheuristics
Problem Modifications and Related Problems
Conclusions
See also
References

Keywords

Telecommunication; Combinatorial optimization; Spanning tree; Capacitated minimum spanning tree problem; Terminal layout problem; Resource-constrained minimum spanning tree problem

The *capacitated minimum spanning tree problem* (CMST) or *terminal layout problem* is usually described as the problem of determining a rooted spanning tree of minimum cost in which each of the subtrees off the root node contains at most K nodes. That is, the CMST is a generalization of the well-known *minimum spanning tree problem* (MST) where the objective is to find a minimum cost tree spanning a given set of nodes such that some capacity constraints are observed.

As a graph theoretic problem we consider a connected graph $G = (V, A, b, c)$ with node set $V = \{0, \ldots, n\}$ and arc set A. Each node $i \in V$ has a nonnegative node weight b_i which may be interpreted as capacity requirement whereas a nonnegative arc weight c_{ij} represents the cost of using arc $(i, j) \in A$. Node 0 denoted as the *center node* will be the root of the tree (with $b_0 := 0$). We define a *subtree* or *component* C_i of a tree spanning V as its maximal subgraph uniquely connected to the center by arc $(0, i)$ (denoted as *central arc*). The demand of a subtree is the sum of the node weights of the included nodes. To satisfy the *capacity constraint* the demand of each subtree must not exceed a given capacity K. (Without loss of generality we may assume $b_i \leq K$ for all i.) By means of these definitions the CMST is the problem of finding a minimum cost tree spanning node set V where all subtrees satisfy the capacity constraint.

In spite of existing polynomial algorithms for the unconstrained MST the CMST has been shown to be *NP*-hard [32] even when all b_i-values are identical. This case of the CMST is referred to as *unit weight CMST* or *equal demand* CMST; otherwise it is called the *nonunit weight* case. Most references in the literature deal with the unit weight case with only a few exceptions treating the more general case. For a comprehensive survey of the (unit weight) CMST up to the mid-1990s see [4]. The CMST in undirected graphs requires a sym-

metric cost matrix. Otherwise, the direction of the arcs has to be considered, i. e., all subtrees are directed. This *capacitated minimum spanning arborescence problem* (CMDT) includes the CMST as a special case.

Motivated by the intractability of the problem both heuristic as well as exact algorithms have been developed. In the sequel various algorithmic concepts are reviewed mainly for the unit weight CMST (with special emphasis on progress made in the late nineties; for some older yet important references not given here see [4]). However, first we sketch some applications of the CMST some of which may also lead to important modifications of the problem.

Applications

The CMST has a great variety of applications especially in the field of telecommunications network design. For instance, in the design of minimum cost teleprocessing networks terminals (nodes) have to be connected to a central facility (the center node) by so-called multipoint lines (the subtrees) which have to be restricted with respect to the traffic transfered between the center and the included terminals or the number of terminals included in the line. The latter is sometimes called reliability constraint because it limits the maximal number of terminals disconnected from the central facility in the case of a single link breakdown. Although different constraints may be referred to as capacity constraints (e. g. considering arc weights instead of node weights or even nonlinear weight functions depending on the distance of a node or arc from the center) most formulations in the literature consider only one of them.

Mathematical Programming Formulations

For the CMST a great variety of formulations may be found in the literature; see, e. g., [14,19,20,21,22]. Here we restrict ourselves to the presentation of a well-known flow-based formulation. As relaxations of directed formulations may be advantageous we consider the CMDT.

Assume $b_i = 1$ for all $i = 1, \ldots, n$, and $b_0 = 0$, then the CMDT can be described as a mixed integer linear programming formulation as follows. Define $x_{ij} = 1$, if arc (i, j) is included in the solution, and $x_{ij} = 0$, otherwise. Furthermore, let y_{ij} denote the flow on arc (i, j) for all i, j, i. e., $i = 0, \ldots, n$ and $j = 1, \ldots, n$. Ensure variables x_{ij} and y_{ij} with $(i, j) \notin A$ to be equal to zero by assigning prohibitively large weights to them. The following single-commodity flow formulation gives a minimum cost directed capacitated spanning tree with center node 0 as the root:

$$
\text{(P)} \quad \begin{cases}
\min & \displaystyle\sum_{i=0}^{n}\sum_{j=1}^{n} c_{ij} \cdot x_{ij} \\
\text{s.t.} & \displaystyle\sum_{i=0}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \\
& \displaystyle\sum_{i=0}^{n} y_{ij} - \sum_{i=1}^{n} y_{ji} = 1, \\
& \quad j = 1, \ldots, n, \\
& x_{ij} \le y_{ij} \le (K - b_i) \cdot x_{ij} \\
& \quad \text{for all } i, j, \\
& x_{ij} \in \{0, 1\}, \quad y_{ij} \ge 0 \quad \text{for all } i, j.
\end{cases}
$$

The first set of equalities ensures that exactly one arc is reaching each noncentral node. The coupling constraints in combination with the flow conservation ensure that no cycles are allowed and that the capacity constraint is satisfied in each subtree. For a formal proof of cycle prevention see [14], i. e., a tree spanning all nodes is guaranteed.

Exact Algorithms

Most exact algorithms for solving the CMST are based on the branch and bound or the branch and cut paradigm, while other approaches are usually not competitive due to time and space complexity (e. g. dynamic programming [23]).

When describing the concepts from the literature in most cases we do not report computational experiments as there is no fair comparison. When reporting problem sizes solved to optimality by a specific algorithm different authors have proposed various ways of conducting experiments (e. g. the way of data generation), i. e., comparability is not always guaranteed. Moreover, problem instances with a larger number of nodes might be easier to solve than those with a smaller number of nodes, depending on the respective values of K [20,24]. Another aspect which seems to have considerable impact on the performance of most algorithms is

the location of the root. For instance, instances with the root in the center of a rectangle in the Euclidean plane may be solved very easily compared to instances with the root in the corner of the rectangle. Recently, a set of problem instances with 40 and 80 nodes became used consistently (see, e. g., [20,22]).

Branch and bound methods for the CMST can be divided into two classes. *Node oriented methods* branch by fixing nodes, *arc oriented methods* branch by including an arc (i, j) into the solution (i. e. fixing $x_{ij} = 1$) or excluding it from the solution ($x_{ij} = 0$). A node is called *established* if the path from the root to this node only consists of arcs fixed to 1 (these arcs are called established, too). Usually, only those arcs incident with exactly one established node are allowed to be fixed. If an arc is fixed to 1, it becomes established and both of its end nodes are established. Correspondingly, an arc is called *disallowed*, if $x_{ij} = 0$ is fixed.

A well-known relaxation of the CMST is the MST relaxation which can be easily solved to optimality. If the MST solution is feasible for the CMST, then it is also optimal and the respective problem can be fathomed.

In the early 1970s an arc oriented branch and bound algorithm based on the MST relaxation was very popular. Subproblems are, e. g., branched by defining the first not yet established arc of an infeasible subtree (the first counted from the center) as established or disallowed, respectively [8]. This approach may be improved by using logical tests and tighter lower bounds [10]. Let node i be established, then in a subproblem disallowing arc (i, j) all arcs (i, k) with node k being an established node of the same subtree as j may be disallowed, too, without loosing optimality. If an optimal solution is lost by disallowing these arcs, the complementary subproblem with established arc (i, j) contains another optimal solution. A dominance criterion is used to fathom some subproblems. In addition, the lower bounds are improved using a special case of the degree constrained MST considering that the degree of the center node — and hence the number of subtrees — is greater or equal to the ratio of the total demand and the capacity K of each subtree.

A. Kershenbaum and R.R. Boorstyn [27] propose two branch and bound algorithms both using last-in first-out to choose the subproblem that is next to be considered. One of the algorithms is node oriented. It starts with n subtrees and each node being 'permissible' for each subtree. A subproblem is branched by including or excluding a node from a specific subtree. Lower bounds are obtained from a partitioning algorithm. The node weights used in this algorithm are originally derived from the MST solution and then, during the branch and bound, transformed in a weight exchange process. Theoretically, these bounds are at least as good as those from the MST relaxation, in practice they are much better. With the same partitioning technique an arc oriented branch and bound algorithm similar to the one of [8] is developed.

B. Gavish [14] compares several relaxations of the CMST with respect to lower and upper bounds. Best results are obtained with a Lagrangian relaxation with an additional degree constraint combined with a subgradient optimization procedure.

Outperforming his previous methods, Gavish [15] develops a new binary programming formulation for the CMST based on an extension of the subtour elimination constraints known from the traveling salesman problem (TSP). Because of the large number of these constraints involved in the formulation an augmented Lagrangian procedure is developed where a dual ascent algorithm is used to obtain initial multipliers and a subgradient procedure to optimize them.

L. Gouveia [20] presents a flow formulation with binary variables z_{ijq} being 1 if a flow of q units goes through arc (i, j). Instead of the $O(n^2)$ constraints of the above flow formulation (P) only $O(n)$ constraints are required. The linear relaxation of the new formulation yields lower bounds as good as those produced by the original formulation. With additional constraints different Lagrangian relaxation schemes are obtained that yield some improvements on the bounds of Gavish [15], especially for problem instances with small capacity K and the center in the 'corner' of a rectangle containing the nodes.

K. Malik and G. Yu [30] present another branch and bound algorithm with Lagrangian subgradient optimization. They give a formulation for the CMST (closely related to the one of [15]) and additional tightening constraints which are added to the problem during the optimization process. Based on a multicommodity flow formulation R. Kawatra [26] uses a Lagrangian approach, too.

L. Hall [24] reports on experience with a cutting plane algorithm for instances with up to 200 nodes

making clever use of polyhedral methods. Gouveia and P. Martins [22] propose a hop-indexed generalization of formulation (P). Further improvements on the lower bounds for problem instances with the root in the corner of a rectangle are obtained.

P. Toth and D. Vigo [38] provide an exact algorithm for the CMDT and numerical results are also provided for problem instances with up to 200 nodes. Their approach uses an additive lower bounding procedure combining a Lagrangian lower bound and a lower bound based on solving minimum cost flow problems.

Heuristics

Before presenting heuristics for the CMST it is useful to consider the characteristics of feasible and infeasible solutions [4]. A solution consists of a set of components $C_i = (V_i, A_i)$ with node set V_i and arc set A_i where usually C_i is a spanning tree for V_i. Each component includes only one central arc so that two different node sets V_i and V_j may have the center node as the only common node. Joining all node sets V_i would yield the entire node set V.

A component is called *feasible* if it does not violate the capacity constraint, and *infeasible*, otherwise. It is referred to as *central* if it includes the center node and *noncentral*, otherwise (i. e. a noncentral component results from a component by eliminating the central arc and the center node). Sets of components having both infeasible and noncentral components are not considered as a solution.

A solution is called *feasible*, if every component contained in the solution is central and feasible itself. It is *incomplete*, if every component is feasible but at least one is noncentral. If all components are central but at least one is not feasible then a solution is called *infeasible*.

The following special solutions of the CMST may be emphasized. The incomplete solution with $n + 1$ components $C_i = (\{i\}, \emptyset)$, $i = 0, \ldots, n$, is called an *empty tree*. All components are feasible and all except C_0 are noncentral. The feasible solution with n components $C_i = (\{0, i\}, \{(0, i)\})$, $i = 1, \ldots, n$, is called a *star*. All components are central and feasible. In the case of a sparse graph with only a subset of nodes being directly connected to the center artificial arcs with high cost values should be introduced to complete the graph. The star then might be feasible only for the modified problem.

Finding Initial Feasible Solutions

Most procedures for determining initial feasible solutions (start procedures) for the CMST may be classified as *construction procedures*, *savings procedures* or *dual procedures*.

Construction Methods

Construction methods start with an incomplete solution, usually the empty tree, and successively enlarge it until the solution is feasible. Most procedures in this category replace two components and the chosen arc that connects them by a new component. We may distinguish between arc oriented and node oriented methods.

Arc oriented (or *best arc*) procedures choose in a greedy fashion arcs which are used to join its two incident components. The procedures stop when a feasible solution is obtained. The components of the final solution generally are not built one by one but simultaneously. It is not necessary to finish one component before starting another one.

As examples one may use the basic principle of well-known MST algorithms. The *modified Kruskal algorithm* [7] in each iteration chooses a feasible arc with lowest cost and joins the two corresponding components. All arcs that have become infeasible in this step are removed from consideration for the next iterations. Correspondingly, the *modified Prim algorithm* in each iteration chooses an arc with minimal cost which is incident to the center or a central component (with not yet exhausted capacity).

Node oriented (or *best node*) procedures choose in a greedy fashion a node or component and join it to its nearest neighbor component by the best possible arc incident to the chosen component, while preserving feasibility.

An obvious idea is to cluster the nodes into groups of no more than K nodes and then to choose the arc set according to an MST for the nodes of each group and the center. Assuming that coordinates of the nodes are given this approach may be referred to as *clustering* (or *sweep*) algorithm [36].

The *Martin algorithm* [25,31] chooses the component which is most distant to the center and joins it to its nearest feasible neighbor component. If χ_i is the cost of connecting component C_i to the center, the component with maximal χ_i is chosen.

The *regret method* (or *Vogel approximation method* (VAM, [8]) computes for every component C_i a regret $r_i = a_2(i) - a_1(i)$ that has to be accepted if C_i is not joined to its nearest feasible neighbor component with cost $a_1(i)$ but to its second nearest feasible neighbor with cost $a_2(i)$. The component with maximal regret is chosen and joined to its nearest neighbor. The regrets are recomputed and the procedure continues until the solution is feasible.

Mixed procedures combine arc and node aspects. They assign a weight w_i to each node i and compute for every arc (i, j) the trade-off function value as $t_{ij} = w_i - c_{ij}$. The feasible arc with largest t_{ij} is chosen and the respective components are joined. In general, the weights have to be updated after each iteration. With an appropriate definition of the weight function and an update rule all preceding heuristics except the clustering procedures can be incorporated in this concept [29]:

- The Kruskal algorithm is obtained for $w_i = 0$ for all i. Obviously, no update is needed.
- For the Prim algorithm assign weights $w_i = 0$ to all central components and $w_i = -\infty$ to all other components. If a noncentral component is joined with a central component, the weight of the new component is set to zero.
- The Martin algorithm requires $w_i = \chi_i + a_1(i)$, and with $w_i = a_2(i) = r_i + a_1(i)$ one obtains the VAM. The weights w_i have to be recomputed if the values of $a_1(i)$ or $a_2(i)$ have changed, respectively.

Mixed VAM is a combined regret-best arc procedure [18,39]. The regret r_i is used as node weight w_i and thus, the trade-off function is $t_{ij} = r_i - c_{ij}$.

The *unified algorithm* [29] proposes a parameterization of the weight function and the trade-off function.

Savings Procedures

Savings procedures for the CMST usually start with the star. The best feasible change, i. e. the change which yields the largest savings, is performed. This is iteratively repeated until no savings can be obtained any more. The methods could easily be applied to other feasible solutions and so they could be classified as improvement procedures, too.

The *Esau–Williams algorithm* (EW, [11]) joins the two components which yield the maximal savings in cost. The savings s_{ij} of joining C_i and C_j is defined as $s_{ij} = \max\{\chi_i, \chi_j\} - c^*_{ij}$ if joining of C_i and C_j is feasible, and $s_{ij} = \infty$, otherwise, with χ_i again being the minimal cost of the connection from the center to the nodes of C_i and c^*_{ij} being the minimal cost of an arc connecting C_i and C_j. Then all savings concerning the new component have to be recomputed and again the maximal savings is chosen. The process is stopped if no more positive savings are available.

The EW is closely related to the above mentioned best node procedures. For instance, the Martin algorithm may be referred to as a less greedy version of the EW.

The EW can also be described as a special case of the unified algorithm starting with the empty tree and at each step adding a feasible arc with maximal trade-off $t_{ij} = \chi_i - c_{ij}$.

Whitney's savings heuristic [10,39] modifies the EW by allowing noncentral arcs to be deleted as well as central arcs. This leads to a possible recombination of segments of the components. Here we see again that savings algorithms are closely related to the class of improvement procedures.

The *parallel savings algorithm* (PSA) [18] computes savings like the EW. However, one iteration does not only join one pair of components but a set of pairs with maximal total savings. This set is determined by solving a maximum weight matching (maximal with respect to the savings) in an adequate graph.

To avoid the parallel construction of nearly equal sized components which cannot be joined any longer if they exceed half of the capacity, Gavish [16] proposes consideration of dummy nodes which yield high savings for any component joined with them. Thus, in this *PSA with dummy nodes* the number of joins between original components in one iteration is reduced by half the number of dummy nodes.

Dual Procedures

Dual procedures start with an infeasible low cost solution, usually the MST solution. The *violation of the con-*

straint(s) is iteratively reduced at the expense of a total cost increase until the solution becomes feasible.

The start procedure of D. Elias and M.J. Ferguson [10] examines every arc (i, j) of any infeasible component. If (i, j) is deleted the resulting noncentral component C_k is connected to another central component by arc (k, l). This arc is chosen such that the total capacity overflow is reduced. Ties are broken such that the smallest cost increase is chosen. The procedure deletes that arc (i, j) which leads to minimal total cost increase $c_{kl} - c_{ij}$ and adds (k, l) to the solution. (As a modification the arc (i, j) with minimal ratio of cost increase and capacity overflow reduction could be chosen.) Given integer cost weights, the procedure terminates with a feasible solution after a finite number of iterations, because in each iteration the total capacity overflow is reduced by at least one unit.

Given a feasible solution one can try to improve the solution by recombining segments of the components in a similar way [10]. To increase flexibility, consider a slight modification: Exchanging two arcs should be allowed even if it leads to an increase of total capacity overflow whenever the cost of the solution does not increase and the arc that is to be included never had been in the solution before.

Dual procedures may well be related to other concepts. For instance, a dual procedure may be seen as a constructive savings procedure starting within the infeasible region of the solution space. In that sense it might be related to metastrategies as, e. g., tabu search described below in the sense that it performs a recover phase within a strategic oscillation approach.

Additional Procedures

Besides classifying construction, savings or dual procedures, there are procedures using aggregation and decomposition techniques combined with dynamic programming. In addition, some heuristics which start with generating a TSP tour are not considered in that scheme.

Gouveia and J. Paixão [23] present two heuristics for the CMST which are based on problem size reduction by aggregation and decomposition techniques. In the *aggregation heuristic* the nodes are clustered using the EW — thus forming new nodes with higher and in general nonidentical weights — until the resulting aggregated problem is small enough to be solved to optimality in time limits deemed practical. The *decomposition heuristic* creates for each central arc of the MST solution a subproblem by considering only the nodes of the respective subtree. Subproblems which are small enough are solved to optimality. For the remaining subproblems the aggregation heuristic is used.

Note that the above mentioned sweep algorithm might be classified as aggregation procedure, too.

For the case of unit weights, K. Altinkemer and Gavish [2] provide a modified PSA with a worst-case error bound of $3 - 2/K$ and derive a bound of 4 for the case of nonunit weights. First, a TSP tour is constructed and then it is partitioned into feasible subtrees by adding some central arcs and removing respective noncentral arcs. Note that the (noncenter) nodes of the resulting subtrees are always connected in the same order as in the TSP tour.

In the case of unit weights a *K-iterated tour partitioning algorithm* is used: K solutions are constructed. In each solution the first subtree starts with the first (noncenter) node of the TSP-tour, the second subtree starts with node 2 (first solution), node 3 (second solution), …, node $K + 1$ (Kth solution). Apart from the first and the last each subtree contains exactly K nodes. The best out of these K solutions is chosen.

For nonunit weights a *nearest insertion optimal partitioning algorithm* may be applied: In the nearest insertion tour the nodes are renumbered according to their position. Modified costs c_{ij}' are computed as the cost of a tree linking the center with node i, node i with node $i + 1$, etc., and node $j - 1$ with node j. If such a tree is infeasible (due to capacity), the respective cost is set to infinity. With these definitions, the shortest path with respect to c_{ij}' from the center to node n represents the optimal partitioning.

In each procedure a final step can be added: The solution is improved by computing MSTs for the derived components.

Improvement Procedures

Improvement procedures for the CMST can be classified as either *local exchange procedures* or *second order procedures*.

Neighborhood Definition

Local exchange procedures start with a feasible solution and seek to improve it by modifying the current solution in a prespecified way: Sets of arcs are included in or excluded from the solution. If more than one change of the solution is possible the best one (with respect to cost) is chosen. The procedure continues as long as improvements are possible.

Given a feasible solution, H. Frank et al. [13] examine for every node i the following exchange: Connect i to its nearest neighbor not yet connected to i and remove the arc with highest cost from the resulting cycle while still preserving feasibility. The exchange with greatest cost decrease is chosen as long as improvements are positive. The authors describe this procedure for a network design problem with variable arc capacities and cost but it can be naturally applied to the special case with only one available capacity and fixed cost for each arc as in the CMST.

Elias and Ferguson [10] try to improve the solution by recombining segments of the components, i. e., deleting an arc and reconnecting the resulting noncentral component without loosing feasibility (cf. the Whitney savings heuristic above).

The previously reported improvement procedures alter a current solution by including or excluding arcs. In contrast, a node exchange procedure transforms one feasible solution to a neighbor solution by changing the assignment of the nodes to the subtrees. Such a transformation is called *move*. Subsequently a certain number of moves is performed thus trying to find improved solutions.

Starting from the EW solution, in their CMST procedure A. Amberg et al. [4] consider two types of moves: *Shift moves* choose one node and shift it from its actual component to another one. *Exchange moves* choose two nodes belonging to different subtrees and exchange them. Both types may be simultaneously used whereas only *feasible moves* are allowed, i. e. those leading again to feasible solutions.

A modified neighborhood definition involves cutting a subtree from a given solution and to paste it within another subtree or to connect it to the root node [35]. Additional neighborhood structures are given in [1]. Contrary to the previous neighborhood structures the authors do not restrict themselves to the consideration of two subtrees to be involved in one move but into a chain of moves performed simultaneously (called cyclic exchanges and path exchanges). That is, the number of exchanges grows exponentially with the problem size. Based on a shortest path algorithm some profitable exchanges may be determined in way which may be termed *Lin–Kernighan neighborhood* or *ejection chain*.

Second Order Algorithms

Second order algorithms iteratively apply a slave procedure to different start solutions (where some arcs are fixed to be included) and/or modified cost matrices (where inhibitively high cost has been assigned to some arcs) thus forcing arcs into or out of the solution. Savings procedures as the EW or the PSA are applied as slave procedures to complete the solution. In each iteration, all possible modifications according to a given rule are checked. The best one is realized and the respective modifications are made permanent for the remaining iterations. Two important second order algorithms are inhibit and join [25].

The *inhibit procedure* examines for every arc of the current solution the effect of excluding this arc by applying the EW to a modified graph where the cost of the respective arc has been made inhibitively high. The inhibition yielding the lowest cost solution is made permanent (the arc is inhibited for the remaining iterations) and the process is repeated until no further cost reduction can be obtained. At most $O(n^2)$ iterations, each with at most $O(n)$ inhibitions, have to be considered.

The *join procedure* determines for every node i its nearest neighbor i_1 as well as the closest neighbor i_2 closer to the center than i (if different from i_1). It computes the effect on the cost of the solution if node i is directly connected to node i_1 or alternatively to node i_2 (if this is not already done in the actual solution) by applying the start procedure on a modified graph. The joining which produces the best solution is made permanent and the procedure is repeated with this solution. In each of the $O(n)$ iterations $O(n)$ joins have to be considered.

It should be noted that both procedures, inhibit and join, are already look ahead procedures (trying to overcome a shortsighted myopic behavior). Both improvement procedures can be used alone or in combina-

tion with each other performing one iteration of join after an iteration of inhibit and vice versa. Combining the procedures restricts the number of iterations to $O(n)$ (from the join procedure) yielding a complexity of $O(n^2)$ times the EW complexity.

For the improvement procedure of [28] in a first step the MST solution and the EW solution have to be determined. Then the following iteration is performed. Define T as the set of arcs which are in the MST but not in the EW solution. For every nonempty subset S of T generate a (incomplete) solution including these arcs (if this is feasible), then exclude all arcs of the remaining subset $T\backslash S$ (by modifying the respective arc costs) and complete the solution by applying the heuristic. Choose the subset S^* which yields the largest improvement and permanently include these arcs into the solution. Repeat this iteration with modified $T := T\backslash S^*$.

The *min-exchange heuristic* outlined in [17] starts with any given feasible solution and determines for every pair of components C_p and C_q the cheapest arc (i, j) connecting the two components. All arcs incident to i or j are deleted. C_p and C_q are decomposed into two noncentral single node components C_i and C_j and some remaining components. Now the noncentral components are connected with the center; hereby the minimal cost arcs are chosen. The PSA completes this modified solution. The authors propose to split all components simultaneously.

Computational Results

In the early CMST literature the EW has been found to perform best on average when compared to procedures with similar computation times. Therefore, even nowadays EW is taken as a benchmark to check the performance of other procedures. Kershenbaum and W. Chou [29] report that the unified algorithm running with 3 to 10 different parameter combinations and correspondingly multiplied computation times yields 1–5% improvement over EW. Unfortunately, no specific parameter combination produces improvements in general.

Gouveia and Paixão [23] admit the nearest insertion optimal partitioning algorithm to perform much worse than EW on average with some significant exceptions. This shows that no general dominance of EW considering single problem instances can be derived.

Gavish and Altinkemer [16,18] report for test problems with up to 400 nodes that the PSA yields improvements of 2–4% in the unit weight case, but performs poorly for nonidentical weights. In the latter case, the min-exchange heuristic applied to the PSA solution gives results comparable to those of EW [17]. Gavish [16] reports that the PSA with dummy nodes attains improvements over EW (up to 6% some cases). However, in the nonunit weight case EW performs still better. Here, the PSA with constant number of joins gives consistently better results than EW. Gouveia and Paixão [23] apply this variant of the PSA with the number of joins varying between 1 (which is in fact the EW) and 12 on unit weight test problems with up to 200 nodes: Significant improvements over EW with computation times raised by a factor of up to 250 are obtained. They also report that the (original) PSA performs best when the capacity is a power of 2 (in the unit weight case). Their aggregation heuristic on average yields a slight improvement over EW (up to 3% in some cases). In test problems, that have the center in the 'middle' of the rectangle containing the nodes, the decomposition procedure has larger computation times than the aggregation algorithm (factors of slightly more than 1 up to 3 are found) and better results, whereas in cases with the center on the 'corner' of the rectangle both methods in almost all instances have similar running times and solutions. Apart from a few cases the PSA with constant number of joins (and varied parameters) on average performs better than both procedures.

M. Karnaugh [25] tests inhibit and join on problems with up to 150 nodes. The combination of the procedures gives 2–3% improvement over EW while the running time is increased by a factor 100 (derived for the 150-node problems). Applying only inhibit performs slightly worse.

Kershenbaum et al. [28] found that inhibit, join and their own procedure yield improvements of around 2% over EW. Thus, their own procedure requiring only 2 to 3 times more computation time than EW, outperforms join.

Metaheuristics

Given a local search mechanism, a metastrategy like tabu search or simulated annealing as a *guiding process* decides which of the possible moves is chosen and for-

wards its decision to the *application process* which then executes the chosen move. In addition, it provides some information for the guiding process (depending on the requirements of the respective metastrategy) like the recomputed set of possible moves.

Contrary to the improvement procedures reported in the last section, the cost of a new solution may exceed the cost of the previous one. Moves leading to a cost increase are allowed in order to overcome local optima. Which of the available feasible moves should be chosen to transform the current solution? The answer to this question is not clear and various approaches may lead to good solutions. The guiding process may use, e. g., the two metastrategies simulated annealing and tabu search.

Simulated annealing (SA) randomly chooses one of the feasible moves and its change in cost is computed. If the change is a cost decrease the move is performed. Otherwise, the new solution is accepted with a certain probability. The probability function usually is logarithmic and — intending to favor good solutions — decreases with raising amount of cost increase. It decreases with the number of iterations already performed thus intensifying the search in the current area of the solution space when the execution time is growing. A parameter called *start temperature* has to be specified to adapt the probability function to the actual problem. SA does not require any additional information. If the new solution is rejected the current solution remains unchanged in this step. The next iteration tries again to alter the same solution. Simulated annealing implementations for the CMST are given in [4,6].

Tabu search (TS) examines all feasible moves. The best move — leading to the highest cost decrease or the lowest cost increase, respectively — is chosen and performed. Now suppose that a local optimum is reached. Without further instructions the procedure could permanently alternate between this local optimum and its best neighbor. For that reason a so-called *tabu list* is created: To prevent that a yet explored solution is examined again, all moves that (could) lead to such a solution are stored in the tabu list. Which moves have to be set tabu is derived from the *running list* (RL) containing all performed moves in their sequence of execution. Both lists have to be updated after each iteration.

In the literature there are several distinct ways of deriving the tabu list. They are referred to, e. g., as static tabu search STS, reverse elimination method REM and cancellation sequence method CSM (see [4] for the CMST). For STS and CSM some parameters have to be specified to adopt the methods to a specific problem and problem instances (especially problem size and scaling of cost).

The storage complexity of the application process is $O(n^2)$ and the time complexity $O(K^2)$ per iteration because of the recomputation of MSTs in the changed components. Using simulated annealing we have a time complexity of the guiding process of $O(K^2)$: To compute the probability of acceptance for the new solution two new subtrees have to be computed. This is part of the application process and need not be performed twice. Thus, additional effort only arises if a solution is rejected which does not influence the overall complexity. The storage complexity also is not raised if simulated annealing is used.

The complexity of the guiding process depends on the special tabu search method. Different tabu search implementations are described in [4,35]. Whereas [4] seem to provide better results for the benchmark instances with up to 80 nodes than [35], both seem to be outperformed by the more recent (as of 2000) algorithm in [1] based on their more powerful neighborhood structures.

Besides TS and SA additional modern heuristic search concepts have been investigated for the CMST. A neural network approach is investigated in [33]. A GRASP implementation is provided in [34]. The results for both approaches seem to be behind some of those described in the previous paragraphs.

Problem Modifications and Related Problems

Additionally to considering arc costs one may take into account unreliable arcs and node outage costs which are incurred by the user whenever a terminal node is unable to communicate with the central node, i. e., costs associated with link failures [9].

An interesting modification of the CMDT is the resource-constrained minimum spanning tree problem in directed graphs [12]. Here each node, say i, has a certain amount of scarce resources available (a capacity) which may be used to fulfill capacity requirements of all arcs leaving i. Instead of measuring capacity requirements for subgraphs off the root node here the con-

sideration restricts to the set of incident arcs leaving a node. The current state of the art for solving this problem circumvents a branch and cut approach [12].

In practice we might be faced with the problem that a solution of the design phase need not be a tree but a forest with more than one root node. Most of the approaches developed for the CMST might be applied in a slightly modified way to this so-called multicenter CMST (see e. g. [3] for an extension of the partitioning heuristics with corresponding worst-case bounds).

When multiple centers are considered in arc oriented vehicle routing then the capacitated arc routing problem (CARP) may be transformed in a way that subproblems are successively solved as CMST. Amberg et al. [5] develop this transformation and apply their TS and SA approaches to this multiple center CARP.

Besides solving the CMST as a pure combinatorial optimization problem it may also be embedded into a problem of users with traffic requirements who have to build contracts with, e. g., a telephone company for the provision of service. This may lead to the consideration of some game-theoretic concepts associated with a cost allocation problem arising from the CMST or more general capacitated network design problems [37].

Conclusions

In this paper we have provided a survey on existing methods for solving the CMST.

With respect to considered algorithmic concepts it might be interesting to incorporate some sort of either exact or heuristic reduction techniques.

See also

- ► Bottleneck Steiner Tree Problems
- ► Directed Tree Networks
- ► Minimax Game Tree Searching
- ► Shortest Path Tree Algorithms

References

1. Ahuja RK, Orlin JB, Sharma D (1998) New neighborhood search structures for the capacitated minimum spanning tree problem. Techn Report Sloan School Management, MIT
2. Altinkemer K, Gavish B (1988) Heuristics with constant error guarantees for the design of tree networks. Managem Sci 32:331–341
3. Altinkemer K, Pirkul H (1992) Heuristics with constant error guarantees for the multi center capacitated minimum spanning tree problem. J Inform Optim Sci 13:49–71
4. Amberg A, Domschke W, Voss S (1996) Capacitated minimum spanning trees: Algorithms using intelligent search. Combin Optim: Theory and Practice 1:9–39
5. Amberg A, Domschke W, Voss S (2000) Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. Europ J Oper Res 124:360–376
6. Andersen K, Vidal RVV, Iversen VB (1993) Design of a teleprocessing communication network using simulated annealing. In: Vidal RVV (ed) Applied simulated annealing. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 201–215
7. Boorstyn RR, Frank H (1977) Large-scale network topological optimization. IEEE Trans Communications 25:29–47
8. Chandy KM, Russell RA (1972) The design of multipoint linkages in a teleprocessing tree network. IEEE Trans Comput 21:1062–1066
9. Dutta A, Kawatra R (1994) Topological design of a centralized communication network with unreliable links and node outage costs. Europ J Oper Res 77:344–356
10. Elias D, Ferguson MJ (1974) Topological design of multipoint teleprocessing networks. IEEE Trans Communications 22:1753–1762
11. Esau LR, Williams KC (1966) On teleprocessing system design. IBM Systems J 5:142–147
12. Fischetti M, Vigo D (1997) A branch-and-cut algorithm for the resource-constrained minimum-weight arborescence problem. Networks 29:55–67
13. Frank H, Frisch IT, Slyke R Van, Chou WS (1971) Optimal design of centralized computer design network. Networks 1:43–57
14. Gavish B (1983) Formulations and algorithms for the capacitated minimal directed tree problem. J ACM 30:118–132
15. Gavish B (1985) Augmented Lagrangean based algorithms for centralized network design. IEEE Trans Communications 33:1247–1257
16. Gavish B (1991) Topological design of telecommunication networks – local access design methods. Ann Oper Res 33:17–71
17. Gavish B, Altinkemer K (1986) A parallel savings heuristic for the topological design of local access tree networks. Techn Report Graduate School Management Univ Rochester, NY
18. Gavish B, Altinkemer K (1986) Parallel savings heuristics for the topological design of local access tree networks. Proc IEEE INFOCOM 86 Conf, pp 130–139
19. Gouveia L (1993) A comparison of directed formulations for the capacitated minimal spanning tree problem. Telecommunication Systems 1:51–76

20. Gouveia L (1995) A 2n constraint formulation for the capacitated minimal spanning tree problem. Oper Res 43:130–141
21. Gouveia L, Hall L (1998) A comparative study of network flow formulations for the capacitated spanning tree problem. Working Paper Fac Ciencias Univ Lisboa 10
22. Gouveia L, Martins P (1999) The capacitated minimal spanning tree problem: An experiment with a hop-indexed model. Ann Oper Res 86:271–294
23. Gouveia L, Paixao J (1991) Dynamic programming based heuristics for the topological design of local access networks. Ann Oper Res 33:305–327
24. Hall L (1996) Experience with a cutting plane approach for the capacitated spanning tree problem. INFORMS J Comput 8:219–234
25. Karnaugh M (1976) A new class of algorithms for multipoint network optimization. IEEE Trans Communications 24:500–505
26. Kawatra R (1994) A multicommodity network flow application for the capacitated minimal spanning tree problem. Opsearch 31:296–308
27. Kershenbaum A, Boorstyn RR (1983) Centralized teleprocessing network design. Networks 13:279–293
28. Kershenbaum A, Boorstyn RR, Oppenheim R (1980) Second-order greedy algorithms for centralized teleprocessing network design. IEEE Trans Communications 28:1835–1838
29. Kershenbaum A, Chou W (1974) A unified algorithm for designing multidrop teleprocessing networks. IEEE Trans Communications 22:1762–1772
30. Malik K, Yu G (1993) A branch and bound algorithm for the capacitated minimum spanning tree problem. Networks 23:525–532
31. Martin J (1967) Design of real-time computer systems. Prentice-Hall, Englewood Cliffs, NJ
32. Papadimitriou CH (1978) The complexity of the capacitated tree problem. Networks 8:217–230
33. Patterson RA (1995) Hybrid neural networks and network design. PhD Thesis Ohio State Univ
34. Rolland E, Patterson RA, Pirkul H (1999) Memory adaptive reasoning and greedy assignment techniques for the capacitated minimum spanning tree problem. In: Voss S, Martello S, Osman IH, Roucairol C (eds) Meta-heuristics: Advances and trends in local search paradigms for optimization. Kluwer, Dordrecht, pp 487–498
35. Sharaiha YM, Gendreau M, Laporte G, Osman IH (1997) A tabu search algorithm for the capacitated shortest spanning tree problem. Networks 29:161–171
36. Sharma RL, El-Bardai MT (1970) Suboptimal communications network synthesis. Proc 1970 Internat Conf Comm (19.11-19.16.)
37. Skorin-Kapov D, Beltran HF (1994) An efficient characterization of some cost allocation solutions associated with capacitated network design problems. Telecommunication Systems 3:91–107
38. Toth P, Vigo D (1995) An exact algorithm for the capacitated shortest spanning arborescence. Ann Oper Res 61:121–141
39. Whitney VKM (1970) A study of optimal file assignment and communication network configuration in remote access computer message processing and communication systems. PhD Thesis Univ Michigan, Ann Arbor

Carathéodory, Constantine

NICOLAS HADJISAVVAS[1], PANOS M. PARDALOS[2]
[1] Department Math., University Aegean Karlovassi, Samos, Greece
[2] Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 01A99

Article Outline

Keywords
See also
References

Keywords

Carathéodory; Calculus of variations; Measure theory

Constantin Carathéodory, a mathematician of Greek origin, was born in Berlin on September 13, 1873 and died on February 2, 1950, in Munich, Germany. He made important contributions to the theory of real functions, to the *calculus of variations*, and to *measure theory*.

He first studied in the Brussels' Military School, where he received a solid mathematical background. After two years as an assistant engineer with the British Asyut Dam project in Egypt, Carathéodory began his study of mathematics at the Univ. of Berlin in 1900, where he attended the courses of L. Fuchs, G. Frobenius and H. Schwarz. He was particularly influenced by Schwarz' lectures with whom he became a close friend. In 1902 he entered the Univ. of Göttingen, where he received his PhD [1] under the German mathematician H. Minkowski. In 1909 he became a full Professor in the Univ. of Hannover. In 1913 he obtained the chair held previously by F. Klein in Göttingen and in 1918

he succeeded Frobenius in the Univ. of Berlin. Then, in 1920, he accepted to help the Greek Government in creating the Univ. of Smyrna, Asia Minor, which then belonged to the Greeks. When the Turks razed Smyrna in 1922, Carathéodory managed to save the university library, which he moved to the Univ. of Athens, where he taught until 1924. He then was appointed professor of mathematics at the Univ. of Munich.

Carathéodory made important contributions to various branches of mathematics. In the calculus of variations, besides a comprehensive study of discontinuous solutions, which was contained in his PhD thesis, he also added important results linking the theory with first order partial differential equations. His work on the problems of variation of m-dimensional surfaces in an n-dimensional space marked the first far-reaching results for the general case. He also applied the calculus of variations to specific problems of mechanics and physics. He contributed important findings in his book [6]. The theory of functions and measure theory are two additional areas where the work of Carathéodory is very important. His book [3] is a classic of the field. In the theory of functions of several variables he simplified the proof of the main theorem of conformal representation of simply connected regions on the unit-radius circle. His investigations of the geometrical-set theoretic properties of boundaries resulted in his theory of boundary correspondence. Already in 1909 he published a far-reaching paper on the foundations of *thermodynamics* [2]. The paper remained unnoticed by the physicists, because it was published in a mathematical journal. Only in 1921 M. Born brought the paper to the attention of the physics community, and since then the paper and the *Carathéodory principle* became classics. He also contributed to Einstein's special theory of relativity. His published works include [4,5,7,8,9].

See also

- ► Carathéodory Theorem
- ► History of Optimization

References

1. Carathéodory C (1904) On the discontinuous solutions in the calculus of variations. PhD Thesis
2. Carathéodory C (1909) Untersuchungen über die Grundlage der Thermodynamik. Math Ann 67:355–386
3. Carathéodory C (1918) Vorlesungen über reelle Funktionen. Teubner, Leipzig, 2nd edn. 1928. Also: Chelsea Publ. 1948
4. Carathéodory C (1932) Conformal representation. Tracts in Math and Math Phys, vol 28. Cambridge Univ. Press, Cambridge
5. Carathéodory C (1935) Variationsrechnung und partielle Differentialgleichungen erster Ordnung, 2nd edn. Teubner, Leipzig, 159 p
6. Carathéodory C (1937) Geometrische Optik. Ergebnisse der Math und ihre Grenzgeb. Springer, Berlin
7. Carathéodory C (1939) Reelle Funktionen, vol I. Teubner, Leipzig
8. Carathéodory C (1950) Funktionentheorie, vol 1–2. Birkhäuser, Basel
9. Carathéodory C (1956) Mass und Integral und ihre Algebraisierung. Birkhäuser, Basel

Carathéodory Theorem

GABRIELE E. DANNINGER-UCHIDA
University Vienna, Vienna, Austria

MSC2000: 90C05

Article Outline

Keywords
See also
References

Keywords

Polytope; Convex hull; Representation

One of the basic results [3] in convexity, with many applications in different fields. In principle it states that every point in the convex hull of a set $S \subset \mathbf{R}^n$ can be represented as a convex combination of a finite number $(n+1)$ of points in the set S. See for example [1,4,6,7,9,10]. Generalizations of the theorem can be found in [2] and [5].

Theorem 1 *Let S be any subset of $\boldsymbol{R}^n$.*

For every $x \in conv(S)$ (the convex hull of S), there exist $n+1$ points $x_0, \ldots, x_n \in S$ such that $x \in conv(x_0, \ldots, x_n)$.

Proof Since $x \in \mathrm{conv}(S)$, there exists a representation $x = \sum_{i=0}^{k} \alpha_i x_i$, $x_i \in S$ for $i = 0, \ldots, k$ and $\sum_{i=0}^{k} \alpha_i = 1$.

If $k \leq n$, we are finished.

Now suppose $k > n$. Note that then $x_1 - x_0, \ldots, x_k - x_0$ are linearly dependent. There then exist scalars $\lambda_1, \ldots, \lambda_k$, not all zero, such that $\sum_{i=1}^{k} \lambda_i (x_i - x_0) = 0$.

Let now $\lambda_0 = -\sum_{i=1}^{k} \lambda_i$; it then follows that $\sum_{i=0}^{k} \lambda_i x_i = 0$ and we can find at least one $\lambda_i > 0$. So we have,

$$x = \sum_{i=0}^{k} \alpha_i x_i - \gamma \cdot 0 = \sum_{i=0}^{k} \alpha_i x_i - \gamma \sum_{i=0}^{k} \lambda_i x_i = \sum_{i=0}^{k} (\alpha_i - \gamma \lambda_i) x_i$$

for any $\gamma \in \mathbf{R}$.

Choose γ in the following way:

$$\gamma = \min_{0 \le i \le k} \left\{ \frac{\alpha_i}{\lambda_i} : \lambda_i > 0 \right\} = \frac{\alpha_j}{\lambda_j}$$

for some $j \in \{0, \ldots, k\}$ so, $\alpha_i - \gamma \lambda_i \ge 0$ for all $i = 0, \ldots, k$.

Then we obtain $x = \sum_{i=0}^{k} (\alpha_i - \gamma \lambda_i)\, x_i$ with $\alpha_i - \gamma \lambda_i \ge 0$ for $i = 0, \ldots, k$, $\sum_{i=0}^{k} (\alpha_i - \gamma \lambda_i) = 1$ and $\alpha_j - \gamma \lambda_j = 0$.

And so x is represented as a convex combination of at most k points in S. We can now repeat these steps until $k = n$.

See also

- ► Carathéodory, Constantine
- ► Krein–Milman Theorem
- ► Linear Programming

References

1. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming. Wiley, New York
2. Bonnice W, Klee VL (1963) The generation of convex hulls. Math Ann 152:1–29
3. Carathéodory C (1911) Ueber den Variabilitätsbereich der Fourierschen Konstanten von positiven harmonischen Funktionen. Rend Circ Mat Palermo 32:193–217
4. Grünbaum B (1967) Convex polytopes. Interscience, New York
5. Reay JR (1965) Generalizations of a theorem of Carathéodory. Memoirs Amer Math Soc 54:
6. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton
7. Stoer J, Witzgall Ch (1970) Convexity and optimization in finite dimensions I. Springer, Berlin
8. Valentine FA (1964) Convex sets. McGraw-Hill, New York
9. Wets RJ-B (1976) Grundlagen Konvexer Optimierung. Lecture Notes Economics and Math Systems, vol 137. Springer, Berlin
10. Ziegler GM (1995) Lectures on polytopes. Springer, Berlin

Checklist Paradigm Semantics for Fuzzy Logics

LADISLAV J. KOHOUT
Department Computer Sci., Florida State University, Tallahassee, USA

MSC2000: 03B52, 03B50, 03C80, 62F30, 62Gxx, 68T27

Article Outline

Keywords
Why the Checklist Paradigm?
  Fuzzy Logics
  Approximate Reasoning
Many-Valued Logics in Fuzzy Sets
The Checklist Paradigm
  Mathematics of the Checklist Paradigm
  Interval Inference and The Checklist Paradigm
  Other Systems of Fuzzy Logic Connectives for Interval Inference
  Optimization of Interval Inference
Other Systems of Checklist Paradigm Connectives
Collapse of Intervals into Points Under the Additional Probabilistic Constraints
Checklist Paradigm and Generalized Quantifiers
Checklist Paradigm and Four Modes of Reasoning
Group Transformations of Logic Connectives and the Checklist Paradigm
  Group Transformations of Logic Connectives and the Checklist Paradigm
  An 8-Element Group of Logic Transformations
  A 16-Element $S_{2\times2\times2\times2}$ Group of Logic Transformations
Conclusion
See also
References

Keywords

Checklist paradigm; Many-valued logics; GUHA; Exploratory statistical analysis; Semantics of MVL connectives; Generalized quantifier; Observational quantifiers; Fuzzy sets; Logic of approximation; Interval computing; Approximate reasoning

Why the Checklist Paradigm?

The classical logic deals with two logical values—'truth' and 'falsity'. It can be characterised algebraically and semantically by a *Boolean algebra*. Not all issues of logic can, however, be settled by a system of classical two-valued logic. For example some modalities, such as necessity and possibility cannot, in general, be expressed in any system that admits only a finite number of logical values. Also some temporal logics [33] characterising time require an infinite number of logical values in their semantics.

Fuzzy Logics

Many-valued logic algebras are needed for developing the mathematics of *fuzzy relations* [28] and sets [18]. For example, in order to compute the degree δ to which two fuzzy sets intersect, we use the formula $\delta_{A\cap B}(x) = \delta_A(x) \wedge \delta_B(x)$, where $\wedge$ is a many-valued 'AND' connective and $\delta_A(x)$, $\delta_B(x)$ are some logical values: either truth-values, possibilities, probabilities, etc. Depending on the *epistemological interpretation* of the logical values, we read the statement $\delta(A)(x)$ 'The degree to which it is true that $x \in A$', 'The degree to which it is possible that $x \in A$', 'The degree to which it is probable that $x \in A$', etc.

Computing the *degree of inclusion* of two sets [2] is done by the formula $\delta(A \subseteq B) = (\forall x)\delta_A(x) \rightarrow \delta_B(x)$, where x ranges over elements of the universe U from which the elements of A and B are drawn. Here $\rightarrow$ is a many-valued *implication operator*.

Approximate Reasoning

Many-valued logic systems are also required for algebraic characterization of logics of *approximate reasoning*. The *premises of an inference* (i. e. the antecedent formulas that form the arguments of the rules of approximate inference) are used by the rules to generate the succedent formulas — the *conclusion*(s).

If each of these logic formulas attains as its logic value a single value from some *lattice*, we speak of a *point-based logic* system of approximate reasoning. If the logic value is a whole interval $[\delta_k, \delta_l]$ such that $\delta_k \leq \delta_l$ it is an *interval logic*.

Hence, many-valued logics play a key role in all the areas of mathematics and logic discussed above. There is not one many-valued logic, there is an infinite number of families of logic systems of various kinds. Hence, according to the purpose of its use, one has to choose an appropriate many-valued system. But even after the choice is made, the two key questions still remain:

- Where the logic values come from?
- Is there any basic epistemic or semantic procedure by which the basic logic connectives can be meaningfully derived?

These questions are answered by the *checklist paradigm*.

Many-Valued Logics in Fuzzy Sets

The theory of fuzzy sets and relations requires a many-valued logic in which to manipulate the degrees of truth which attach to fuzzy statements. As in classical two-valued logic (in which the statements are judged to be either utterly true or utterly false), one wishes a *truth-functional* connection between the truth values assigned to 'p' and to 'q' and those to be assigned to 'p or q' and 'p and q' and 'if p then q', as well as to 'not-p' and 'not-q', that is, one wishes the evaluation of the derived formulas to depend solely on the evaluation of the original formulas, without further reference to their contents.

There are a number of such many-valued logical systems, with truth values in the closed real interval [0, 1]. Everyone agrees that the values assigned in the crisp 'corners', where the values $|p|$ of p and $|q|$ of q are zero (false) or one (true), must accord with the classical Boolean logic. Most agree in setting

$$|\text{not} - p| = |\neg p| = 1 - |p|$$

and the most usual 'or' and 'and' connectives are given by

$$|p \text{ or } q| = |p \vee q| = \max(p, q),$$
$$|p \text{ and } q| = |p \wedge q| = \min(p, q),$$

although other have been proposed and have something to be said for them.

Selecting max and min as the functions for computing the logical values of the *connectives* $\vee$ and $\wedge$ does not yet determine the system of many-valued logic fully. Indeed, a number of different systems employ these. Third determining factor is the choice the *impli-*

cation operator $\rightarrow$. Some frequently used $\rightarrow$ are listed below.

Checklist Paradigm Semantics for Fuzzy Logics, Table 1
Some important many-valued implication operators

No	Opr.	Definition
2.	S	Standard Strict $a \overset{2}{\rightarrow} b = \begin{cases} 1, & a \le b \\ 0, & \text{otherwise} \end{cases}$
3.	S^*	Gödel $a \overset{3}{\rightarrow} b = \begin{cases} 1, & a \le b \\ b, & \text{otherwise} \end{cases}$
4.	G43	product ply (also: Goguen–Gaines) $a \overset{4}{\rightarrow} b = \min\left(1, \frac{b}{a}\right)$
4′.	G43′	Modified G43 $a \overset{4'}{\rightarrow} b = \min\left(1, \frac{b}{a}, \frac{1-a}{1-b}\right)$
5.	L	Lukasiewicz $a \overset{5}{\rightarrow} b = \min(1, 1 - a + b)$
5.5	KDL	Reichenbach $a \overset{5.5}{\longrightarrow} b = \min(1, 1 - a + ab)$
6.	KD	Kleene–Dienes $a \overset{6}{\rightarrow} b = (1 - a) \vee b$
7.	EZ	Early Zadeh $a \overset{7}{\rightarrow} b = (a \wedge b) \vee (1 - a) = (a \overset{6}{\rightarrow} b) \wedge ka$ where $ka = (1 - a) \vee a$
8.	W	Willmott $a \overset{8}{\rightarrow} b = (a \overset{7}{\rightarrow} b) \wedge kb$

Not only properties of the many-valued logic systems but also of the systems of *fuzzy sets* crucially depend on the choice of the implication operator. For example both the definition of a *fuzzy power set* (i. e. the set of all subsets) and of the *fuzzy set-inclusion operator* depend on its choice. The very first paper on fuzzy sets by L.A. Zadeh [44,45] uses max and min connectives to define the intersection $\cap$ and the union $\cup$ of two fuzzy sets. The set inclusion operator Zadeh defines by the formula

$$\mu(A \subseteq B) = 1 \quad \Leftrightarrow \quad (\forall x)\mu_A \le \mu_B(x).$$

Using the 'Standard Strict' $\rightarrow_2$ in the formula given of the first section above we obtain

$$\begin{aligned} &\delta(A \subseteq B) \\ &= (\forall x)\delta_A(x) \rightarrow \delta_B(x) \\ &= (\forall x)\mu_A(x) \overset{2}{\rightarrow} \mu_B(x) \\ &= \min_{\{x \in U\}} (\mu_A(x) \overset{2}{\rightarrow} \mu_B(x)) \,. \end{aligned}$$

This formula is equivalent to Zadeh's early definition of fuzzy set inclusion which in fact is crisp (nonfuzzy). Power set theories with proper fuzzy set inclusion have been first investigated in [2,43] (using the *implication operators* listed in the table above).

Since 1965, when the first paper on fuzzy sets was written by Zadeh, not only max and min but also other many-valued logic connectives were used to define the union $\cup$ and the intersection $\cap$ of fuzzy sets. An important pair are the so called 'bold connectives': '$a \wedge_5 b = \max(0, a + b - 1)$' and '$a \vee_5 b = \min(1, a+ b)$'. As the subscript indicates, these are related to the *Łukasiewicz implication* operator. These represent the so-called *MV algebras* which play an important role in application of fuzzy sets in quantum logics [35] and elsewhere. Both types of connectives, the pairs 'max-min' and the 'bold' connectives are special instances of the so-called triangular norms (*t-norms*) and conorms (*t-conorms*) [17,36]. These associative operations with special properties, defined on [0, 1], algebraically characterise the whole infinite family of OR-AND pairs of many-valued logic connectives and play a crucial role in the theory and applications of fuzzy sets.

The Checklist Paradigm

The *checklist paradigm* provides the mechanism by which several types of very different families of many-valued logic connectives *emerge* from some more basic considerations.

- It provides the *semantics* of systems that use single value as its logic value.
- It provides the justification for *interval logics*.
- It provides a link of *many-valued logics* connectives with *generalized quantifiers*.

Mathematics of the Checklist Paradigm

A *checklist template* Q is a finite family of properties $\langle P_1, \ldots, P_n \rangle$. With a template Q, and a given *proposition* A, one can associate a specific checklist $Q_A = \langle Q, A \rangle$. A *valuation* f_A of a checklist Q_A is a function from Q to $\{0, 1\}$.

The value a_Q of the proposition A with respect to a template Q (which is the summarised value of the valuation f_A) is given by the formula

$$a_Q = \sum_{i=1}^{n} p_i^A$$

where $n = \text{card } Q$ and $p_i^A = f_A(P_i)$.

A *fine valuation structure*, a pair of propositions A, B with respect to the template Q, is a function $f_{A,B}^Q$ from Q into $\{0, 1\}$ assigning to each attribute P_i the ordered pair of its values $\langle p^A, p^B \rangle$.

Let $\alpha_{j,k}$ be the cardinality of the set of all attributes P_i such that $f_{A,B}^Q(P_i) = \langle j, k \rangle$.

Obviously we have the following constraint on the values: $\alpha_{00} + \alpha_{01} + \alpha_{10} + \alpha_{11} = n$. Further, we define $r_0 = \alpha_{00} + \alpha_{01}$, $r_1 = \alpha_{10} + \alpha_{11}$, $c_0 = \alpha_{00} + \alpha_{10}$, $c_1 = \alpha_{01} + \alpha_{11}$.

These entities can be displayed systematically in a *contingency table*. In such a table, the inner fine-summarization structure consists of the four $\alpha_{j,k}$ appropriately arranged, and of margins c_0, c_1, r_0, r_1 (see Fig. 1).

Now let F be any logical propositional function of propositions A and B. For $i, j \in \{0, 1\}$, let $f(i, j)$ be the classical truth value of F for the pair i, j of truth values; let $u(i, j) = \alpha_{i,j} n$, the ratio of the number in the ij-cell of the constraint table, to the grand total. Then we define the (nontruth-functional) *fuzzy assessment of the truth* of the proposition $F(A, B)$ to be

$$m(F(A, B)) = \sum_{i,j} f(i, j) \cdot u_{ij}.$$

	No for B	Yes for B	Row total
No for A	α_{00}	α_{01}	r_0
Yes for A	α_{10}	α_{11}	r_1
Column Total	c_0	c_1	n

Checklist Paradigm Semantics for Fuzzy Logics, Figure 1
Checklist paradigm of the assignment of fuzzy values. Define: $a = r_1/n$; $b = c_1/n$

This assessment operator will be called the *contraction/approximation measure*.

The four interior cells α_{00}, α_{01}, α_{10}, α_{11} of the constraint table constitute its *fine structure*; the margins r_0, r_1, c_0, c_1 constitute its *coarse structure* (see Fig. 1).

The fine structure gives us the appropriate fuzzy assessments for all propositional functions of A and B; the coarse structure gives us only the fuzzy assessments of A and B themselves. Our central question is:

> to what extent can the fine structure be reconstructed from the coarse?

As shown elsewhere [3,5,6,8] the coarse structure imposes bounds upon the fine structure, without determining it completely. Hence, associated with the various logical connectives between propositions are their extreme values.

There are four extremes that the fine structure of the contingency table (see Fig. 1) can attain [6,8]:

i) the two *mindiag fine structures* with the diagonal values minimized ($\alpha_{00} = 0$ or $\alpha_{11} = 0$); and
ii) the two *maxdiag fine structures* with the diagonal values maximized ($\alpha_{01} = 0$ or $\alpha_{10} = 0$).

Thus we obtain the inequality restricting the possible values of $m(F)$:

$$\text{con top} \geq m(F) \geq \text{con bot},$$

where 'con' is the name of connective represented by $f(i, j)$. Choosing for the logical type of the connective 'con' the *implication* and making the assessment of the fuzzy value of the truth of a proposition by the formula $m_1(F) = 1 - u_{10}$ we obtain:

$$\min(1, 1 - a + b) \geq m_1(A \to B) \geq \max(1 - a, b).$$

We can see that the checklist paradigm generated the *Łukasiewicz implication* operator, and the *Kleene–Dienes implication* operator.

We have already noted that choosing for 'F' the connective type 'AND' [5,8]) and m_1, we obtain the bounds

$$\min(a, b) \geq m_1(\text{AND}) \geq \max(0, a + b - 1).$$

These bounds are formally identical with those of B. Schweitzer and A. Sklar [36] giving the bounds on *copulas* which play an important role in their theory

of *t-norms* and *t-conorms*. Surprisingly, these checklist paradigm bounds also coincide with Novák's recent (1991) derivation [31] of bounds on fuzzy sets approximating classes of Vopěnka's alternative set theory [41,42]. E. Hisdal derives the same inequalities as the bounds on some connectives of her TEE model and comments on a possible link (cf. [16, Appendix A2]). In the context of modalities in fuzzy logics, checklist paradigm-like inequalities for $F = \{\text{AND, OR}\}$ were recently (1992) also independently discovered in [34]. Yet all these models are neither formally nor epistemologically identical. This indicates the need for a more precise meta- and metametalogical formulation of many-valued based mathematical systems, that would include in their full definition a part formulating their 'mathematical epistemology'.

Interval Inference and The Checklist Paradigm

The checklist paradigm puts ordering on the pairs of distinct implication operators and other pairs of connectives. Hence it provides a theoretical justification of *interval-valued approximate inference*. For the m_1 contraction/approximation measure, there are 16 inequalities linking the *TOP and BOT types of connectives* [5,8], thus yielding 16 logical types of TOP-BOT pairs of connectives. Ten of these interval pairs generated by m_1 are listed in Table 2.

Checklist Paradigm Semantics for Fuzzy Logics, Table 2
Two-argument interval pairs of connectives generated by m_1

Logical Type of Connective	Valuation BOTTOM≤TOP
AND $a\&b$	$\max(0, a+b-1) \le \min(a, b)$
Nicod $a \downarrow b$	$\max(0, 1-a-b) \le \min(1-a, 1-b)$
Sheffer $a\|b$	$\max(1-a, 1-b) \le \min(1, 2-a-b)$
OR $a \vee b$	$\max(a, b) \le \min(1, a+b)$
Nonimplication $a \nleftarrow b$	$\max(0, b-a) \le \min(1-a, b)$
Nonimplication $a \nrightarrow b$	$\max(0, a-b) \le \min(a, 1-b)$
Implication $a \leftarrow b$	$\max(a, 1-b) \le \min(1, 1+a-b)$
Implication $a \rightarrow b$	$\max(1-a, b) \le \min(1, 1-a+b)$
Equivalence $a \equiv b$	$\max(1-a-b, a+b-1) \le \min(1-a+b, 1+a-b)$
Exclusive OR $a \oplus b$	$\max(a-b, b-a) \le \min(2-a-b, a+b)$

Other Systems of Fuzzy Logic Connectives for Interval Inference

In Boolean (crisp) logic, the values of a logical formula written in the *disjunctive normal form* (*DNF*) are equal to the values the formula expressed in the *conjunctive normal form* (*CNF*). This does not hold for every system of many-valued connectives.

I.B. Türksen [38,39,40] has shown that for $\max(a, b)$, $\min(a, b)$ and some other *t*-norm and *t*-conorm based CNFs and DNFs the inequality $\text{DNF}(a\ \text{CON}\ b) \le \text{CNF}(a\ \text{CON}\ b)$ holds for all 16 basic many-valued connectives CON.

Taking for example the max-min based CNF and DNF, the corresponding implications are given by

$$\text{CNF}(a \rightarrow b) = (\neg a \vee b),$$
$$\text{DNF}(a \rightarrow b) = (a \wedge b) \vee (\neg p \wedge q) \vee (\neg a \wedge \neg b).$$

For further information on other systems of connectives for *fuzzy interval inference* see [11,27].

Optimization of Interval Inference

Formulas that are equivalent logically [6,27] may not be equivalent when compared by their formula complexity. This is well-known phenomenon when expressing logical formulas in DNF or CNF normal forms [19]. The same logical function expressed in one of these forms may have more complicated expression in the other normal form. Similarly, this can be observed with other logic connectives [40]. So, transformations between logically equivalent formulas expressed by different connectives may have different formula complexity. Hence, the knowledge of such transformations is useful in optimization of interval inference.

For example, *exclusive OR*, or the *eor operator*, is conveniently defined in two ways: as 'a without b' or

'b without a', or else as 'a or b but not both', thus

$$a \text{ eor } b = (a \text{ and } \neg b) \text{ or } (\neg a \text{ and } b) = (a \text{ or } b) \text{ and } (\neg a \text{ and } b).$$

Using these definitions together with the definitions of the previous section easy calculations bring the results of the following Table [6]:

$a \equiv_{\text{top}} b$	$= (a \rightarrow_{\text{top}} b) \wedge_{\text{top}} (b \rightarrow_{\text{top}} a)$
	$= (a \wedge_{\text{top}} b) \vee_{\text{top}} (\neg a \wedge_{\text{top}} \neg b)$
$a \equiv_{\text{bot}} b$	$= (a \rightarrow_{\text{bot}} b) \wedge_{\text{bot}} (b \rightarrow_{\text{bot}} a)$
	$= (a \wedge_{\text{bot}} b) \vee_{\text{bot}} (\neg a \wedge_{\text{bot}} \neg b)$
$a \oplus_{\text{top}} b$	$= (a \wedge_{\text{top}} \neg b) \vee_{\text{top}} (\neg a \wedge_{\text{top}} b)$
	$= (a \vee_{\text{top}} b) \wedge_{\text{top}} \neg(a \wedge_{\text{bot}} b)$
$a \oplus_{\text{bot}} b$	$= (a \wedge_{\text{bot}} \neg b) \vee_{\text{bot}} (\neg a \wedge_{\text{bot}} b)$
	$= (a \vee_{\text{bot}} b) \wedge_{\text{bot}} \neg(a \wedge_{\text{top}} b)$

Formulas for equivalence (IFF) and exclusive-OR (EOR)

Other useful formulas are those that give universal bounds on classes of *fuzzy interval pairs* of formulas. If we define the *unnormalized fuzziness* of x [1] as $\phi x = \min(x, 1 - x)$, then for x in the range [0, 1], ϕx is in the range [0, 0.5], with value 0 if and only if x is crisp, and value .5 if and only if x is .5. The following *gap theorem* holds [8]:

Theorem 1

$$\begin{aligned}
&(a \wedge_{\text{top}} b) - (a \wedge_{\text{bot}} b) \\
&\quad = (a \vee_{\text{top}} b) - (a \vee_{\text{bot}} b) \\
&\quad = (a \rightarrow_{\text{top}} b) - (a \rightarrow_{\text{bot}} b) \\
&\quad = \min(\phi a, \phi b). \\
&(a \equiv_{\text{top}} b) - (a \equiv_{\text{bot}} b) \\
&\quad = (a \oplus_{\text{top}} b) - (a \oplus_{\text{bot}} b) \\
&\quad = 2 \min(\phi a, \phi b).
\end{aligned}$$

The width of the interval produced by an application of a pair of associated connectives (i. e. TOP and BOT connectives) characterises the margins of imprecision of an interval logic expression. Because the interval between the TOP connective and the BOT connective is directly linked to the concept of fuzziness ϕ, the margins of imprecision can be directly measured by the degree of ϕ.

Other Systems of Checklist Paradigm Connectives

Several measures other than m_1 that yield interesting results are also important. For implication again, but only the evaluation 'by performance' (that is, we are only concerned with the cases in which the evaluation of A is 1; see Fig. 1, we use $m_2 = u_{11}/u_{10} + u_{11}$) and obtain the inequality

$$\min\left(1, \frac{b}{a}\right) \geq m_2(F) \geq \max\left(0, \frac{a + b - 1}{a}\right),$$

in which the left-hand side is the well-known *Goguen–Gaines implication* (cf. e. g. [3]). Still another *contracting measure* which distinguishes the proportion of satisfactions 'by performance', $u(1, 1)$, and 'by default', $u(0, 0) + u(0, 1)$. This measure given by the formula $m_3 = u_{11} \vee (u_{00} + u_{01})$ yields [3]

$$\max[\min(a, b), 1 - a] \geq m_3(F) \geq \max(a + b - 1, 1 - a).$$

Two variations on measure m_3 have turned out to be of interest [3]. One is its lower *contrapositivization* given by the formula

$$m_4 = (u_{11} \vee (u_{00} + u_{01})) \vee (u_{00} \vee (u_{01} + u_{11}))$$

which gives the following inequality:

$$\min[\max(a + b - 1, 1 - a), \max(b, 1 - a - b)] \leq m_4 \leq \min[\max(1 - a, b), \kappa a, \kappa b],$$

where $\kappa a = a \vee (1 - a)$.

The other arises by taking for the 'performance' part the less conservative m_2 thus obtaining the formula for $m_5 = m_2 \vee (u_{00} + u_{11})$. This yields

$$\max\left[\min\left(1, \frac{b}{a}\right), 1 - a\right] \geq m_5 \geq \max\left[\frac{a + b - 1}{a}, 1 - a\right].$$

For the proofs of the results presented in this subsection and further explanation see [6, Sect. 5] (this is the first paper on the checklist paradigm, published in 1980).

Collapse of Intervals into Points Under the Additional Probabilistic Constraints

When only the row and column totals r_i, c_j of the fine structure are known (see Fig. 1), one can ask what are the expected values for the α_{ij} [3, Sect. 7].

Suppose the ways in which numbers can be distributed within the cells of the fine structure (so as to give the fixed coarse totals) constitute a hypergeometric distribution. Then the means of the distribution for each cell, give the *expected* configuration of the fine structure. The inequalities determining the interval BOTCON $\leq$ TOPCON now turn into equalities:

$$\frac{\alpha_{ij}}{\alpha_{ik}} = \frac{c_j}{c_k}; \qquad \frac{\alpha_{ij}}{\alpha_{hk}} = \frac{r_i}{r_h}.$$

Surprisingly, introducing the expected value (a probabilistic notion) this way causes the fuzzy interval to collapse into a single point: the expected value thus generating the values of the *mid connective* [3,5]. For example, the interval pair $(\rightarrow_5, \rightarrow_6)$ generated by m_1 consisting of the Łukasiewicz implication and Kleene–Dienes implication operators collapses into the *Reichenbach implication* operator.

For further details on the *'probabilistic collapse'* of the interval pairs generated by other measures see [3,5,8,20,26].

Checklist Paradigm and Generalized Quantifiers

Some of the notions and results of the checklist paradigm are in a remarkable relation to the theory of observational generalized quantifiers, as studied by P. Hájek and T. Havránek [14] in connection with the method of *automated hypothesis formation* [12,13]. Namely, a particular type of implication operator and a particular type of implicational quantifier are mutually definable. The link is given by the contingency tables of the checklist paradigm and the statistics of the observational quantifiers [15].

Checklist Paradigm and Four Modes of Reasoning

Classical two-valued logic has presented certain modes of reasoning, of which only two concern us: *modus ponens* and *modus tollens*, respectively.

The first of these derives from the two premises 'if *a* then *b*' and '*a*', the conclusion '*b*'; the second derives from 'if *a* then *b*' and 'not *b*', the conclusion 'not *a*'. The validity of these modes is trivial.

On the other hand, there are two modes of reasoning which are classically illegitimate, although in the daily life we all use something very much like them all the time. These, so-called *plausible rules* [4,32], are shown as the central pair in [26, Fig. 1]. *Denial* derives from 'if *a* then *b*' and 'not *a*', the assertion 'not *b*', while *confirmation* derives from 'if *a* then *b*' and '*b*', the assertion '*a*'.

The reason why these errors in classical reasoning retain a strong intuitive attraction is that most human reasoning does not deal with crisp or two-valued or Boolean truth-versus-falsity, but with graded *degrees of credence*, or belief-worthiness, or whatever you like to call it. Because in multiple-valued logics the plausible modes *gain legitimacy* this intuition about human reasoning gains mathematical legitimacy. Indeed, human reasoning, 'good' human reasoning, is best modeled in multiple-valued logic which admits in addition to the modus ponens and modus tollens also the two modes of plausible reasoning.

In classical logic, an *evaluation* takes each of a given set of propositions into one of the extreme truth-values 0 (false) or 1 (true), subject to some semantic consistency rules. In multiple-valued logic, an *evaluation* is a mapping of the set of propositions into a somewhat richer set, which for present purposes may be taken to be the closed interval [0, 1] from 0 to 1, again subject to certain semantic consistency rules.

Hence, in multiple-valued logic, for any fixed choice among the distinguished implication operators, to the classically valid modes of modus ponens and modus tollens are to be added fuzzily valid modes of denial and confirmation (*modus negans* and *modus confirmans*) [4,7]. Although the out-of-bounds constraints were addressed elsewhere [4], one may wonder, what does the checklist paradigm have to offer when applied to the four plausible modes of inference.

Checklist paradigm is applicable not only to the components of the object language, such as logical operators and connectives, but also at the meta-level, thus providing an interval logic based semantics for various rules of inference. As shown below and in [8], it also provides a justification and the proofs of validity of nonclassical interval-based rules (plausible modes) of reasoning called denial and confirmation (modus negans and modus confirmans) [4,7,8,24].

As already mentioned, these do not have a nontrivial analogy in Boolean crisp logic. Thus we have the following theorems.

Theorem 2 (*checklist modus ponens*) *Given $r = m(A \rightarrow B)$ and $a = m(A)$ satisfying the consistency condition $r \geq 1 - a$, the values of $b = m(B)$ are subject to*

$$r - (1 - a) \leq b \leq r. \tag{1}$$

Theorem 3 (*checklist confirmation*) *Given $r = m(A \rightarrow B)$ and $b = m(B)$ subject to the consistency condition $b \leq r$, the values of $a = m(A)$ are subject to*

$$1 - r \leq a \leq 1 - (r - b). \tag{2}$$

Theorem 4 (*checklist modus tollens*) *Given $r = m(A \rightarrow B)$ and $\neg b = m\ (not - B)$ satisfying the consistency condition $r \geq b$, the values of $\neg a = m\ (not - A)$ are subject to*

$$r - b \leq \neg a \leq r. \tag{3}$$

Theorem 5 (*checklist denial*) *$r = m(A \rightarrow B)$ and $\neg a = m\ (not - A)$ subject to the consistency condition $1 - a \leq r$, the values of $\neg b = m\ (not - B)$ are subject to*

$$1 - r \leq \neg b \leq 2 - (r + a). \tag{4}$$

Group Transformations of Logic Connectives and the Checklist Paradigm

Let us recall that a *realization of an abstract group* is any group of concretely realizable operations which has the same algebraic structure as the given abstract group. It is well known that any abstract group can be concretely realized by a family of permutations. So a specific abstract group provides a global structural characterization of a specific family of permutations that concretely represent this abstract group. This idea can be used for global characterization of logic connectives.

The Piaget Group of Transformations

Such a global characterization of two-valued connectives of logic was first given by Piaget in the context of studies of human cognitive development. J. Piaget and his collaborators have shown that an important role in child's mental development is transition from more concrete to more abstract thinking. This transition plays a role in development of intelligence, which is viewed in the Piagetian setup as a transition from totally ambiguous and vague notions to crisp propositions in two-valued logic.

Given a family of logical connectives one can apply to them various transformations. Individual logic connectives are 2-argument logic functions. Transformations are functors that, taking one connective as the argument will produce another connective.

Let 4 transformations on basic propositional functions $f(x, y)$ of 2 arguments be given as follows:

$$I(f) = f(x, y), \quad D(f) = \neg f(\neg x, \neg y),$$
$$C(f) = f(\neg x, \neg y), \quad N(f) = \neg f(x, y).$$

In 1940, Piaget discovered experimentally a specific concrete form of such transformations. In the set of the above transformations $T_p = \{I, D, C, N\}$ these individual transformations are called *identity*, *dual*, *contradual*, *negation transformation*, respectively.

It has been shown that the *Piaget group of transformation* is satisfied by some many-valued logics (cf. [5,6,10,37]).

The system of connectives

$$\{\equiv_{\mathrm{TOP}}, \oplus_{\mathrm{BOT}}, \equiv_{\mathrm{BOT}}, \oplus_{\mathrm{TOP}}\}$$

obeys the Piaget group of transformations. Hence it possesses the abstract structure of the *Klein 4-element group*.

An 8-Element Group of Logic Transformations

Adding new nonsymmetrical transformations to those defined by Piaget enriches the algebraic structure of logic transformations. In 1979 L.J. Kohout and W. Bandler added the following nonsymmetric operations [22,23]:

$$LC(f) = f(\neg x, y), \quad RC(f) = f(x, \neg y)$$
$$LD(f) = \neg f(\neg x, y), \quad RD(f) = \neg f(x, \neg y)$$

to the above defined four symmetrical transformations. This yields a new 8-element group of transformations.

The abstract 8-element group $[T, *]$ that captures the structure of the above defined logic transformations is also commutative and is called the *symmetric* $S_{2\times2\times2}$ group in the standard terminology of group theory. The interval logic system based on m_1 can be characterized by such groups of transformations.

Given a set of connectives CON and a set of transformations $\mathcal{T}$, we say that $T_{CON,\mathcal{T}} = \mathcal{T}(\mathrm{CON})$ is the

set of connectives generated by the application of $\mathcal{T}$ to the set CON. For example, $a \to_5 b$ will generate such a set of connectives. This generated set is a realization of $S_{2\times2\times2}$.

A 16-Element $S_{2\times2\times2\times2}$ Group of Logic Transformations

The implication operators $a \to_5 b$ and $a \to_6 b$ yielding the measure m_1 are contrapositive. This means that their valuations satisfy the semantic equality $a \to b = \neg b \to \neg a$.

If we are interested in extending the interval logics into the domain of noncontrapositive $\to$ then the corresponding $\wedge$ is not commutative. In order to distinguish the contrapositive cases from noncontrapositive ones in a syntactically correct formal way, an additional operator is introduced.

This operator called, *commutator* K, satisfies the equality $a * b = K(b * a)$. The commutativity as well as the contrapositivity involves restrictions on transformations of connectives. In the abstract group, these restrictions are expressed abstractly as congruences [30]. It is convenient to express such restrictions equationally. For any contrapositive $\to$, the following equalities hold: $C[K(a \to b)] = K[C(a \to b)] = a \to b$. For a noncontrapositive $\to$, (1) fails, but the following equality holds:

$$(K(C(K(C(a \to b))))) = a \to b.$$

The following holds [29]:

Theorem 6 *The closed set of connectives generated by* $\{ \to_4, \leftarrow_4, K\}$ *is a representation of the symmetric 16 element abstract group* $S_{2\times2\times2\times2}$.

Conclusion

The checklist paradigm clearly demonstrates the following general meta-principle: a system of logic connectives is formed by a specific family of connectives together with some common process/structure/principles that involve the said family of connectives in some unifying way, causing these to interact.

In the checklist paradigm semantic model we use two basic unifying principles:

i) approximation (contraction) measures;

ii) transformations of logical types of connectives leading to a global characterization of logics by their groups of transformations [23].

The methods of the checklist paradigm surveyed here give the theoretical bounds on the performance of particular many-valued implication operators and other connectives by deriving these from deeper epistemological and formal assumptions. Hence, it provides a theoretical justification of interval-valued approximate inference. The checklist paradigm, together with fuzzy questionnaires and square and triangle relational products also plays an important role in the experimental identification of fuzzy membership functions and structures [21,25] (see also ► Boolean and fuzzy relations). The results can be extended to the groupoid-based many-valued *Pinkava algebras* (see ► Finite complete systems of many-valued logic algebras) that are used in the design of knowledge-based and other systems [19]. This theoretical work is supplemented by empirical studies of the adequacy of various logical connectives in practical applications of fuzzy sets and relations [9].

See also

- ► Alternative Set Theory
- ► Boolean and Fuzzy Relations
- ► Finite Complete Systems of Many-Valued Logic Algebras
- ► Inference of Monotone Boolean Functions
- ► Optimization in Boolean Classification Problems
- ► Optimization in Classifying Text Documents

References

1. Bandler W, Kohout LJ (1978) Fuzzy relational products and fuzzy implication operators. Internat. Workshop on Fuzzy Reasoning Theory and Appl. Queen Mary College Univ. London, London
2. Bandler W, Kohout LJ (1980) Fuzzy power sets and fuzzy implication operators. Fuzzy Sets and Systems 4:13–30. Reprinted in: Dubois D, Prade H, Yager R (eds) (1993) Readings in Fuzzy Sets for Intelligent Systems. Morgan Kaufmann, San MAteo, CA, pp 88–96
3. Bandler W, Kohout LJ (1980) Semantics of implication operators and fuzzy relational products. Internat J Man-Machine Studies 12:89–116 Reprinted in: Mamdani EH, Gaines BR (eds) (1981) Fuzzy Reasoning and its Applications. Acad Press, New York, pp 219–246

4. Bandler W, Kohout LJ (1984) The four modes of inference in fuzzy expert systems. In: Trappl R (ed) Cybernetics and Systems Res, vol 2. North-Holland, Amsterdam, pp 581–586
5. Bandler W, Kohout LJ (1984) Unified theory of multiple-valued logical operators in the light of the checklist paradigm. In: Proc 1984 IEEE Conf Systems, Man Cybern, IEEE, New York, pp 356–364
6. Bandler W, Kohout LJ (1985) The interrelations of the principal fuzzy logical operators. In: Gupta MM Kandel A, Bandler W, Kiszka JB (eds) Approximate Reasoning in Expert Systems. North-Holland, Amsterdam, pp 767–780
7. Bandler W, Kohout LJ (1985) Probabilistic vs. fuzzy production rules in expert systems. Internat J Man-Machine Studies 22:347–353
8. Bandler W, Kohout LJ (1986) The use of checklist paradigm in inference systems. In: Negoita CV, Prade H (eds) Fuzzy Logic in Knowledge Engineering. TÜV Rheinland, Köln, pp 95–111
9. Ben-Ahmeida B, Kohout LJ, Bandler W (1992) The use of fuzzy relational products in comparison and verification of correctness of knowledge structures. In: Kohout LJ, Anderson J, Bandler W (eds) Knowledge-Based Systems for Multiple Environments. Ashgate Publ. (Gower), Aldershot, Hampshire, UK
10. Dubois D, Prade H (1980) Fuzzy sets and systems: Theory and applications. Acad Press, New York
11. Dubois D, Prade H (1991) Fuzzy sets in approximate reasoning, Part I: Inference with possibility distributions. Fuzzy Sets and Systems 40(1):143–202
12. Hájek P (1978) Special issue on GUHA method. Internat J Man-Machine Studies 10(1):1–93 (guest editor).
13. Hájek P (1981) Special issue on GUHA method. Internat J Man-Machine Studies 15: (guest editor)
14. Hájek P, Havránek T (1978) Mechanizing hypothesis formation: Mathematical foundations of a general theory. Springer, Berlin
15. Hájek P, Kohout LJ (1996) Fuzzy implications and generalized quantifiers. Internat J Uncertainty, Fuzziness and Knowledge Based Systems 4(3):225–233
16. Hisdal E (1990) Infinite-valued logic based of two-valued logic and probability. Pt. 1.4 The TEE model for grades of membership. Inst. Informatics Univ. Oslo Oslo, Oslo
17. Klement EP, Mesiar R (1997) Triangular norms. In: Mesiar R, Riečan B (eds) Tatra Mountains, Math. Publ. (Special Issue: Fuzzy Structures – Current Trends). 13, Math Inst Slovak Acad Sci Bratislava, Bratislava, pp 169–194
18. Klir GJ, Yuan B (1995) Fuzzy sets and fuzzy logic: Theory and applications. Prentice-Hall, Englewood Cliffs, NJ
19. Kohout LJ (1990) A perspective on intelligent systems: A framework for analysis and design. Chapman and Hall and v. Nostrand, London–New York
20. Kohout LJ (1995) Epistemological aspects of many-valued logics and fuzzy structures. In: Höhle U, Klement EP (eds) Non-Classical Logics and their Applications to Fuzzy Subsets: A Handbook of Mathematical Foundations of Fuzzy Set Theory. Kluwer, Dordrecht, pp 291–339
21. Kohout LJ, Anderson J, Bandler W et al. (1992) Knowledge-based systems for multiple environments. Ashgate Publ. (Gower), Aldershot, Hampshire, UK
22. Kohout LJ, Bandler W (1979) Checklist paradigm and group transformations. Techn. Note, Dept. Electrical Engin. Univ. Essex, no. EES-MMS-ckl91.2
23. Kohout LJ, Bandler W (1992) How the checklist paradigm elucidates the semantics of fuzzy inference. Proc. IEEE Internat Conf Fuzzy Systems (1992), IEEE, New York, pp 571–578
24. Kohout LJ, Bandler W (1992) Modes of plausible reasoning viewed via the checklist paradigm. IPMU'92: Proc. Internat. Conf. Information Processing and the Management of Uncertainty in Knowlege-Based Systems, Mallorca, July 6-10 (1992), Univ. Illes Balears, Palma, Mallorca, Spain, Palma de Mallorca, pp 411–414
25. Kohout LJ, Bandler W (1992) Use of fuzzy relations in knowledge representation, acquisition and processing. In: Zadeh LA, Kacprzyk J (eds) Fuzzy Logic for the Management of Uncertainty. Wiley, New York, pp 415–435
26. Kohout LJ, Bandler W (1993) Interval-valued systems for approximate reasoning based on the checklist paradigm. In: Wang P (ed) Advances in Fuzzy Theory and Technology, vol 1, Bookwrights Press, Durham, NC, pp 167–193
27. Kohout LJ, Bandler W (1996) Fuzzy interval inference utilizing the checklist paradigm and BK-relational products. In: Kearfott RB, Kreinovich V (eds) Applications of Interval Computations. Kluwer, Dordrecht, pp 291–335
28. Kohout LJ, Bandler W (2001) A survey of fuzzy and crisp relations. Lecture Notes Fuzzy Math and Computer Sci Creighton Univ, Omaha, NE
29. Kohout LJ, Kim E (1997) Global characterization of fuzzy logic systems with para-consistent and grey set features. In: Wang P (ed) Proc 3rd Joint Conf Inform Sci. JCIS'97 (5th Internat Conf on Fuzzy Theory and Techn), 1, Duke Univ., Durham, NC, pp 238–241
30. Kohout LJ, Kim E (1997) Group transformations of systems of logic connectives. In: Proc. IEEE-FUZ'97, 1, IEEE, New York, pp 157–162
31. Novák V (1991) On the position of fuzzy sets in modelling of vague phenomena. In: Lowen R, Roubens M (eds) IFSA '91 Brussels: Artificial Intelligence, pp 165–167
32. Polya G (1954) Mathematics and plausible reasoning. Princeton Univ. Press, Princeton
33. Prior A (1962) Formal logic. Oxford Univ. Press, Oxford
34. Resconi G, Klir GJ, Clair USt (1992) Hierarchical uncetrainty metatheory based upon modal logic. Internat J General Syst
35. Riečan B, Neubrunn T (1997) Integral, measure, and ordering. Kluwer, Dordrecht
36. Schweizer B, Sklar A (1983) Probabilistic metric spaces. North-Holland, Amsterdam

37. Turksen IB (1979) Containment and Klein groups of fuzzy propositions. Working Paper Dept. Industr. Engin. Univ. Toronto 79-010
38. Turksen IB (1986) Interval-valued fuzzy sets based on normal forms. Fuzzy Sets and Systems 20:191–210
39. Turksen IB (1989) Four methods of approximate reasoning with interval-valued fuzzy sets. Internat J Approximate Reasoning 3:121–142
40. Türksen IB (1995) Type I and interval-valued type II fuzzy sets and logics. In: Wang PP (ed) Advances in Fuzzy Theory and Techn, vol 3. Duke Univ, Durham, NC, pp 31–81
41. Vopěnka P (1979) Mathematics in the alternative set theory. Teubner, Leipzig
42. Vopěnka P (1991) The philosophical foundations of alternative set theory. Internat J General Syst (Special Issue of Fuzzy Sets and Systems in Czechoslovakia) 20(1):115–126
43. Willmott R (1980) Two fuzzier implication operators in the theory of fuzzy power sets. Fuzzy Sets and Systems 4(31–36):31–36
44. Zadeh LA (1965) Fuzzy sets. Inform and Control 8:338–353
45. Zadeh LA (1996) Fuzzy sets: Selected papers II. World Sci, Singapore, edited by Klir G, Yuan B

Chemical Process Planning

Shabbir Ahmed, Nikolaos V. Sahinidis
University Illinois, Urbana-Champaign, USA

MSC2000: 90C90

Article Outline

Keywords
The Long Range Planning Problem
Computational Complexity
Solution Strategies
Integer Programming Approach
Continuous Global Optimization
Approximation Schemes
Dealing with Uncertainty
Conclusion
See also
References

Keywords

Long range planning; Complexity; Mixed integer linear programming; Concave programming; Probabilistic analysis; Stochastic programming

As companies are increasingly concerned about long term stability and profitability, recent years have witnessed growing demand for long range planning tools in all sectors. The chemical process industries are no exception. New environmental regulations, rising competition, new technology, uncertainty of demand, and fluctuation of prices have all led to an increasing need for decision policies that will be 'best' over a long time horizon. Quantitative techniques have long established their importance in such decision making problems. It is therefore no surprise that there is a considerable number of papers in the optimization literature devoted to the problem of long range planning in the processing industries. The purpose of this article is to review recent advances in this area. We will describe the main modeling issues, and discuss the computational complexity, formulations and solution algorithms for this problem.

The Long Range Planning Problem

Consider a plant comprising of several processes to produce a set of chemicals for sale. Each process intakes a number of raw materials and produces a main product along with some by-products. Any of these main or by-products could be the raw materials for another process. Considering the ingredients and final product of all the processes, we have a list of chemicals consisting of all raw materials we consider purchasing from the market, all products we consider offering for sale on the market, and all possible intermediates. The plant can then be represented as a network comprising of nodes representing processes and the chemicals in the list, interconnected by arcs representing the different alternatives that are possible for processing, and purchases to and sales from different markets.

The *process planning problem* then consists of choosing among the various alternatives in such way as to maximize profit. Once we know the prices of chemicals in the various markets and the operating costs of processes, the problem is then to decide the operating level of each process and amount of each chemical to be purchased and sold to the various markets. The problem in itself grows combinatorially with the number of chemicals and processes and is further complicated once we start planning over multiple time periods.

Let us now consider the operation of the plant over a number of time periods. It is reasonable to expect

that prices and demands of chemicals in various markets would fluctuate over the planning horizon. These fluctuations along with other factors, such as new environmental regulations or technology obsolescence, might necessitate the decrease or complete elimination of the production of some chemicals while requiring increase or introduction of others. Thus, we have some additional new decision variables: capacity expansion of existing processes, installation of new processes and shut down of existing processes. Moreover, due to the broadening of the planning horizon, the effect of discount factors and interest rates will become prominent in the cost and price functions. Thus, the planning objective should be to maximize the net present value instead of short term profit or revenue. This is the problem that we shall devote our attention to. The problem can be stated as follows: assuming a given network of processes and chemicals, and characterization of future demands and prices of the chemicals and operating and installation costs of the existing as well as potential new processes, we want to find an operational and capacity planning policy that would maximize the net present value.

Computational Complexity

The number of possible alternatives, regarding which processes to expand and when, increases with the number of processes and the number of time periods. Even though this increase is clearly exponential in the number of processes and time periods, it was not until recently that a formal computational complexity characterization was provided for this problem. In particular, the general long range process planning problem has been shown by S. Ahmed and N.V. Sahinidis [3] to be *NP*-hard by identifying two known *NP*-hard problems as special cases.

Consider first a single-process, multiperiod problem where the decisions consist of determining the expansion sequence to satisfy given demands over a number of time periods at a minimum cost. It can be shown that this problem is equivalent to the *NP*-hard *capacitated lot-sizing problem*, where one has to determine production lot sizes to satisfy demands at a minimum cost. Similarly, a multiple-process, single-time period problem, where the decisions are to determine which processes to install to satisfy demand at a minimum cost, can be shown to be equivalent to the *NP*-hard *knapsack problem*, where one has to select items from a set to place into a knapsack such that weight restrictions are not violated and utility is maximized.

Solution Strategies

Some of the early approaches for the long range planning problem were based on dynamic programming as described by S.M. Roberts [18]. A.S. Manne [5,15] used integer programming approaches to account for economies of scale. D.M. Himmelblau and T.C. Bickel [7] presented a nonlinear programming formulation for a hydrodesulfurization process, and I.E. Grossmann and J. Santibez [6] developed a multi period mixed integer linear programming formulation. Y. Shimizu and T. Takamatsu [22] discussed a goal programming approach where in addition to cost minimization, minimizing the number of expansions is also suggested. M. Santiago, O.A. Iglesias and C.N. Pamiagua [21] developed a method to handle nonlinear concave cost functions arising in planning models. A.G. Jimenez and D.F. Rudd [9] presented a recursive mixed integer linear programming technique and applied it to the Mexican petrochemical industry. We next describe some of the more contemporary approaches to these problems.

Integer Programming Approach

Under the assumption of linear mass balances in the processes and fixed charge cost models, Sahinidis et al. [20] developed a mixed integer linear programming (MILP) formulation of the long range process planning problem as described below.

Indices

- i For the set of *NP* processes
- j For the set of *NC* chemicals
- l For the set of *NM* markets
- t For the set of *NT* time periods

Parameters

- X_{it}^L, X_{it}^U Lower and upper bounds on the expansion of process i in period t.
- a_{jlt}^L, a_{jlt}^U Lower and upper bounds on the availability of chemical j in market l in period t.
- d_{jlt}^L, d_{jlt}^U Lower and upper bounds on the demand of chemical j in market l in period t.

$\Gamma_{jlt}, \gamma_{jlt}$ Forecasted buying and selling prices of chemical j in market l in period t.

μ_{ij}, η_{ij} Input and output proportionality constants for chemical j in process i.

α_{it}, β_{it} Variable and fixed cost for the expansion of process i at the beginning of period t.

Variables

$\boldsymbol{X_{it}}$ Capacity expansion of process i at the beginning of period t.

$\boldsymbol{P_{jlt}}$ Amount of chemical j purchased from market l at the beginning of period t.

$\boldsymbol{Q_{it}}$ Total capacity of process i in period t. The capacity of a process is expressed in terms of its main product.

$\boldsymbol{S_{jlt}}$ Units of chemical j sold to market l at the end of period t.

$\boldsymbol{W_{it}}$ Operating level of process i in period t expressed in terms of output of its main product.

$\boldsymbol{y_{it}}$ A 0–1 integer variable. If process i is expanded during period t then $y_{it} = 1$, else $y_{it} = 0$

Formulation

$$\max NPV = \sum_{t=1}^{NT} \left\{ -\sum_{i=1}^{NP} (\alpha_{it} X_{it} + \beta_{it} y_{it} + \delta_{it} W_{it}) + \sum_{j=1}^{NC} \sum_{l=1}^{NM} (\gamma_{jlt} S_{jlt} - \Gamma_{jlt} P_{jlt}) \right\} \quad (1)$$

subject to

$$y_{it} X_{it}^L \le X_{it} \le y_{it} X_{it}^U, \quad \forall i, t \quad (2)$$

$$Q_{it} = Q_{it-1} + X_{it}, \quad \forall i, t \quad (3)$$

$$W_{it} \le Q_{it}, \quad \forall i, t \quad (4)$$

$$\sum_{i=1}^{NP} (\eta_{ij} - \mu_{ij}) W_{it} = \sum_{l=1}^{NM} S_{jlt} - \sum_{l=1}^{NM} P_{jlt}, \quad \forall j, t \quad (5)$$

$$a_{jlt}^L \le P_{jlt} \le a_{jlt}^U, \quad \forall j, l, t \quad (6)$$

$$d_{jlt}^L \le S_{jlt} \le d_{jlt}^U, \quad \forall j, l, t \quad (7)$$

$$X_{it}, Q_{it}, W_{it} \ge 0, \quad \forall i, t \quad (8)$$

$$y_{it} \in \{0, 1\}, \quad \forall i, t. \quad (9)$$

The objective (1) in the above formulation is to maximize the difference between the sales revenues of the final products and the investment, operating, and raw material costs. Equation (2) is a constraint that bounds capacity expansions. A zero value of y_{it} forces the capacity expansion of process i at period t to zero. If the binary variable equals 1, then the capacity expansion is performed within prescribed bounds. Constraint (3) in the above formulation defines the total capacity available at period t as a sum of capacity available in period $t-1$ and the capacity expansion at the beginning of period t. The condition that the operating level of any process cannot exceed the installed capacity is modeled by constraint (4). Equation (5) expresses mass balances for chemicals across processes and markets. Constraints (6) and (7) are bounds on the purchase and sales quantities. The nonnegativity and binary restrictions are imposed through constraints (8) and (9). Various extensions of this general model are discussed in the recent survey article [2].

Sahinidis et al. [20] developed strong bounding techniques and cutting planes to be used within a *branch and bound* framework to solve the above problem. The fact that the problem is decomposable in the number of time periods can also be exploited by using *Benders decomposition*. Further improvement of the bounding schemes are suggested by reformulating the problem to exploit lot sizing substructure in [19]. The reformulated problem results in a large number of constraints and variables. In [10], the reformulated problem is projected onto a lower-dimensional space to reduce the number of variables, and is solved using a cutting plane strategy along with branch and bound. Computational results in [10,19] and [20] suggest the following:

- Branch and bound with strong bounding techniques performs much better than Benders decomposition for large problems.
- For small sized problems, the reformulation and projection approach do not provide appreciable gains.
- For large problems, the best approach is to use a cutting plane method based on the projected model.

Continuous Global Optimization

In the MILP model, economies of scale in the investment cost functions were modeled by the introduction of a set of binary decision variables (y_{it}) to impose a fixed charge on the decision to expand in addition to the linear term for variable costs. In reality, variable costs are not directly proportional to expansion quantity. Rather, the investment cost is a concave function because of the presence of quantity discounts. Thus, a more realistic model for the investment cost would be:

$$f(X_{it}) = \begin{cases} 0 & \text{when} \quad X_{it} = 0, \\ \beta_{it} + a_{it} X_{it}^{b_{it}} & \text{when} \quad X_{it} > 0, \end{cases}$$

where $a_{it} > 0$ and $0 < b_{it} < 1$. In this formulation, the integer variables have been discarded and the linear variable cost function has been replaced by a concave function in X_{it} with coefficient a_{it} and exponent b_{it}. Note that this function is discontinuous at $X_{it} = 0$. M.L. Liu, Sahinidis and J.P. Shectman [14] present two formulations using these concave cost functions. In the fixed charge concave programming model (FCP), the linear cost relation is retained but the discrete variables are eliminated by using the following concave function:

$$f(X_{it}) = \begin{cases} 0 & \text{when} \quad X_{it} = 0, \\ \beta_{it} + \alpha_{it} X_{it} & \text{when} \quad X_{it} > 0. \end{cases}$$

In the continuous concave programming model (CCP), the discontinuity at $X_{it} = 0$ is avoided by using the following function:

$$f(X_{it}) = a_{it} X_{it}^{b_{it}}.$$

Both (FCP) and (CCP) are problems with concave objective functions to be minimized over a set of linear constraints. These can be solved by a *concave programming* method based on the branch and bound procedure. Computational experience with these models suggests that the algorithm for (FCP) outperforms the straightforward branch and bound for the MILP formulation.

Approximation Schemes

Despite the success of optimization models and algorithms in solving problems of industrial relevance, the majority of approaches in current industrial-level planning practice are still based on *heuristics* rather than integer programming techniques. However, the performance characterization of these approximate methods is based on empirical evidence and little has been done in the way of analytical investigations. Liu and Sahinidis [12] developed a simple heuristic for the process planning problem. The method is based upon solving the LP relaxation of the MILP, and then shifting capacity expansions from latter periods to earlier periods while maintaining feasibility. Worst-case bounds on the performance of this heuristic have also been developed and *probabilistic analysis* of the heuristic has shown that, under standard assumptions on the problem data, the heuristic solution converges to the optimal solution *almost surely* as the problem size increases. A modification of this heuristic for process planning problems with a restriction on the number of allowed expansions has been presented in [3]. The modified heuristic has been proven to be asymptotically optimal *in expectation*.

Dealing with Uncertainty

Uncertainty is an integral part of the long range process planning problem. In the deterministic models discussed above, it is assumed that all uncertainty has been accounted for in the estimation of the problem parameters. Stochastic models, on the other hand, provide explicit means of handling parameter uncertainties.

In process planning problems under uncertainty, the decision maker is interested in a plan that optimizes some sort of a stochastic objective. Two most common such objective functions in the literature are the expected cost/profit of the plan and the plan's flexibility.

Problems with the expected cost objective have been formulated as *two-stage stochastic linear programs* (2S-SLP). In such problems, the uncertain parameters are treated as random variables with known distributions. The desired degree of flexibility of the plan is pre-specified by identifying the probability space over which the plan is required to be feasible. The decision variables of the problem are partitioned into two sets. The *first stage* variables, which are often known as 'design' variables, have to be decided before the actual realization of the random parameters. Subsequently, once the values of the design variables have been decided and

the random events presented themselves, further policy improvements can be made by deciding the values of the *second stage* variables, also known as 'control' or 'operating' variables. The choice of the design variables should be such that the first stage costs and the second stage expected costs are minimized. These problems have been solved using decomposition schemes, where the expectation functional over the uncertain parameter space has been approximated using Monte-Carlo sampling [11], successive disaggregation [4] or by Gaussian quadrature [8]. Computational results in [11] show that a combination of Benders decomposition with Monte-Carlo sampling provide optimal or excellent near-optimal solutions. Problems with up to 10 processes, 4 products, 6 chemicals, and with up to 5^{24} scenarios were solved in at most a few CPU minutes on a standard workstation.

From the flexibility objective point of view, one is interested in a plan that maximizes the range of the uncertain parameters over which the plan remains feasible. Problems of this type are typically harder to formulate and require the identification of a suitable measure of flexibility that one can optimize. Such a formulation has been presented in [24] which maximizes their *stochastic flexibility* metric [23] subject to a cost constraint.

The objectives of optimizing cost or profit and maximizing flexibility are typically conflicting. Formulations that combine the objectives by associating a retrofit cost corresponding to design flexibility have been presented in [16,17].

Liu and Sahinidis [13] applied a *fuzzy programming* approach for the problem of process planning under uncertainty. In this model, the uncertain parameters are considered to be fuzzy numbers with a known range of values, and constraints are treated as 'soft,' i. e. some violation is allowed. The degree of satisfaction of the constraints is then measured in terms of *membership functions*, and the objective is to optimize a measure of constraint satisfaction.

The standard stochastic programming formulation does not address the variability of the uncertain recourse costs across the uncertain parameter scenarios. The need for enforcing robustness of these costs is particularly important to a risk aversive planner in a high variability environment. The stochastic programming formulation of the process planning problem has been extended in [1] to account for robustness of the recourse costs through the use of an appropriate variability criterion. In particular, upper partial mean has been proposed as the measure of variability for its intuitive appeal and to avoid nonlinear formulations. These models provide the decision maker with a tool to analyze the trade-off associated with the expected profit and its variability. To overcome the difficulty associated with solving the robust models which include nonseparable terms, a heuristic procedure for the restricted recourse formulation has been developed. This method iteratively enforces recourse robustness while solving the standard stochastic program in each step. The heuristic generates similar but more conservative trade-off frontiers for the profit and its upper partial mean.

Conclusion

The purpose of this article has been to review the recent advances in the use of optimization techniques in long range chemical process planning. Considerable attention has been devoted to the mixed integer linear programming formulation of the problem and efficient solution schemes that exploit the structure of the problem have been developed. Continuous models have also been successfully solved using global optimization techniques. The combinatorial complexity of the problem has recently motivated the need for heuristics and their performance analysis. Some exciting new results have been obtained in this regard. Uncertainty of the problem parameters has been dealt with through stochastic programming and fuzzy programming models. Various different objective criteria including expected value, flexibility and variability have been considered in extensions of the two-stage stochastic programming formulation and a number of efficient algorithms have been developed for industrially relevant problems.

See also

- ► Extended Cutting Plane Algorithm
- ► Generalized Benders Decomposition
- ► Generalized Outer Approximation
- ► MINLP: Application in Facility Location-allocation
- ► MINLP: Applications in Blending and Pooling Problems

- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming

References

1. Ahmed S, Sahinidis NV (1998) Robust process planning under uncertainty. Industr Engin Chem Res 37:1883–1892
2. Ahmed S, Sahinidis NV (1998) Techniques in long range planning in chemical manufacturing systems. In: Leondes CT (ed) Computer Aided and Integrated Manufacturing Systems: Techniques and Applications. Internat Ser Engin, Techn and Applied Sci. Gordon and Breach, New York
3. Ahmed S, Sahinidis NV (2000) Analytical investigations of the process planning problem. Comput Chem Eng 23:1605–1621
4. Clay RL, Grossmann IE (1996) A disaggregation algorithm for the optimization of stochastic planning models. Comput Chem Eng 21(7):751–774
5. Goreux LM, Manne AS (1973) Multi-level planning: Case studies in Mexico. North-Holland, Amsterdam
6. Grossmann IE, Santibez J (1980) Application of mixed-integer linear programming in process synthesis. Comput Chem Eng 4(205)
7. Himmelblau DM, Bickel TC (1980) Optimal expansion of a hydrodesulfurization process. Comput Chem Eng 4(101)
8. Ierapetritou MG, Pistikopoulos EN (1994) Novel optimization approach of stochastic planning models. Industr Eng Chem Res 33:1930–1942
9. Jiminez AG, Rudd DF (1987) Use of a recursive mixed-integer programming model to detect an optimal integration sequence for the mexical petrochemical industry. Comput Chem Eng 3(291)
10. Liu ML, Sahinidis NV (1995) Long range planning in the process industries: A projection approach. Comput Oper Res 23(3):237–253
11. Liu ML, Sahinidis NV (1996) Optimization in process planning under uncertainty. Industr Eng Chem Res 35:4154–4165
12. Liu M, Sahinidis NV (1997) Bridging the gap between heuristics and optimization: The capacity expansion case. AIChE J 43:2289–2299
13. Liu ML, Sahinidis NV (1997) Process planning in a fuzzy environment. Europ J Oper Res 100(1):142–169
14. Liu ML, Sahinidis NV, Shechtman JP (1996) Planning of chemical processes via global concave minimization. In: Grossmann IE (ed) Global Optimization in Engineering Design. Kluwer, Dordrecht
15. Manne AS (ed) (1967) Investments for capacity expansion. MIT, Cambridge, MA
16. Pistikopoulos EN, Grossmann IE (1988) Stochastic optimization of flexibility in retrofit design on linear systems. Comput Chem Eng 12:1215–1227
17. Pistikopoulos EN, Grossmann IE (1989) Optimal retrofit design for improving process flexibility in nonlinear systems-II. Optimal level of flexibility. Comput Chem Eng 13:1087–1096
18. Roberts SM (1964) Dynamic programming in chemical engineering and process control. Acad. Press, New York
19. Sahinidis NV, Grossmann IE (1992) Reformulation of the multiperiod MILP model for capacity expansion of chemical processes. Oper Res 40(Supp 1):S127–S144
20. Sahinidis NV, Grossmann IE, Fornari RE, Chathrathi M (1989) Optimization model for long range planning in the chemical industry. Comput Chem Eng 13(9):1049–1063
21. Santiago M, Iglesias OA, Pamiagua CN (1986) Optimal technical paths for chemical processes. Comput Chem Eng 10:421–431
22. Shimizu Y, Takamatsu T (1985) Application of mixed integer linear programming in multiterm expansion planning under multiobjectives. Comput Chem Eng 9(367)
23. Straub DA, Grossmann IE (1990) Integrated stochastic metric of flexibility for systems with discrete state and continuous parameter uncertainties. Comput Chem Eng 14:967–980
24. Straub DA, Grossmann IE (1993) Design optimization of stochastic flexibility. Comput Chem Eng 17:339–354

Cholesky Factorization

CAROLINE N. HADDAD
Department Math., State University New York, Geneseo, USA

MSC2000: 15-XX, 65-XX, 90-XX

Article Outline

Keywords
See also
References

Keywords

Symmetric; Ill-conditioned; Well-conditioned; Positive definite; Matrix decomposition or factorization; LU-decomposition; Least squares; Banded; Bandwidth; Normal equation; Pivot; Pivoting; Singular

Solving a linear system of the form $Ax = b$ where $A \in \mathbf{R}^{n\times n}$, $b \in \mathbf{R}^n$ is one of the most fundamental problems in mathematics and science. The two basic categories of numerical solutions are direct methods and iterative methods. Of the direct methods, *Gaussian elimination with backsolving* is the most commonly used technique. Straightforward Gaussian elimination uses row-operations to reduce the system to upper-triangular form before backsolving to find the solution, whereas the equivalent *LU-decomposition* initially factors A into the product of a lower- and upper-triangular matrix: $A = LU$. In the special case where A is both symmetric and positive definite, the matrix may be decomposed into $A = \widetilde{L}\widetilde{L}^\top$ where $\widetilde{L} \in \mathbb{R}^{n\times n}$ is lower-triangular. This decomposition is known as the *Cholesky factorization*, and is named for A.L. Cholesky.

The LU-decomposition of a square matrix, A, is the factorization of A into the product of a lower-triangular matrix, $L \in \mathbf{R}^{n\times n}$ and an upper-triangular matrix, $U \in \mathbf{R}^{n\times n}$. The system $Ax = (LU)x = b$ is then solved by forward solving $Ly = b$ where $y = Ux$, and then backsolving $Ux = y$. The solution can be found in roughly the same number of *floating point operations* (*flops*) as Gaussian elimination with backward substitution. More specifically, both methods require about $n^3/3$ multiplications/divisions and $n^3/3$ additions/subtractions for large n. The main advantage of this method is that once the matrix is factored (which requires $O(n^3)$ steps), the system can be solved repeatedly for different b, which only requires $O(n^2)$ steps. One drawback of this method is that pivoting may be required to find the decomposition.

A special class of problems arises if the matrix in the system is *positive definite*, i. e. $x^\mathsf{T} Ax > 0$ for $x \neq 0$. Note that if A is positive definite, but not symmetric, this implies that $1/2(A+A^\mathsf{T})$, the symmetric part of A, is positive definite. The matrix A, can then be decomposed into the form $A = LDM^\mathsf{T}$ where L, M are lower-triangular and D is a diagonal matrix containing the *pivots* of A. If, in addition to being positive definite, A is also *symmetric*, i. e. $A = A^\mathsf{T}$, then L is symmetric, $M = L$, and the matrix has the special decomposition $A = LDL^\mathsf{T}$, where D has positive entries. Therefore $\sqrt{D}$ exists and A can be decomposed into $A = \widetilde{L}\widetilde{L}^\top$, where $\widetilde{L} = L\sqrt{D}$ and is referred to as the *Cholesky triangle*. Hence the Cholesky factorization is often referred to as the 'square-rooting method' [5]. The major advantage of this is that it requires around half the flops of the standard LU-decomposition.

The Cholesky factorization, presented below for symmetric and positive definite $A \in \mathbf{R}^{n\times n}$ in pseudocode, is taken from [2].

FOR $k = 1, \ldots, n$,
$\quad a_{kk} := \left(a_{kk} - \sum_{p=1}^{k-1} a_{kp}^2\right)^{1/2}$
$\quad$ FOR $i = k+1, \ldots, n$
$\quad\quad a_{ik} := \left(a_{ik} - \sum_{p=1}^{k-1} a_{ip}a_{kp}\right) \Big/ a_{kk}$

Cholesky Factorization, Algorithm 1
A pseudocode for the Cholesky factorization

Example 1 Let $A = \begin{pmatrix} 4 & 2 \\ 2 & 5 \end{pmatrix}$. This matrix is both symmetric and positive definite. Therefore a Cholesky factorization exists for A (see [3] for a proof). An LU-decomposition for it is

$$A = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 4 & 2 \\ 0 & 4 \end{pmatrix}.$$

Note that this is not unique, as another LU-decomposition is

$$A = \begin{pmatrix} 4 & 0 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{pmatrix}.$$

The pivots in both cases are 4 and 4. Hence, the LDL^T-decomposition is

$$A = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{pmatrix}.$$

Finally, the Cholesky factorization is

$$A = LL^\top = \begin{pmatrix} 2 & 0 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}.$$

Additionally, this factorization is unique for symmetric, positive definite matrices. The decomposition can be

performed in fixed-point with *no pivoting required* [9]. This implies that the Cholesky decomposition is *guaranteed to be stable without pivoting*.

If the matrix A is *ill-conditioned*, i. e. has *condition number* $\kappa = \|A\| \, \|A^{-}1\| \gg 1$, then the matrix may be nearly singular and the computed solution to the system, $Ax = b$, may not be sufficiently accurate. A process of iterative refinement may be used to assess the accuracy of the solution and then improve upon it when working in higher precision is not practical. See [2,9] for a more detailed explanation.

The Cholesky factorization can also be used to find the inverse and determinant of a symmetric, positive definite matrix (the LU-decomposition can be used for general $A \in \mathbf{R}^{n\times n}$). It is important for the matrix to be positive definite for a variety of reasons, for example, if A is symmetric, but not positive definite, then the stability is not guaranteed. In the case of finding the inverse of a matrix A, a poor inverse may be obtained even if A is *well-conditioned*, i. e. κ is 'close' to 1.

The efficiency of the Cholesky factorization can be further improved if the matrix is 'banded'. A matrix $A = [a_{ij}]$ is said to have *upper bandwidth* q if $a_{ij} = 0$ whenever $j > i + q$ and *lower bandwidth* p if $a_{ij} = 0$ whenever $i > j + p$. Since A is symmetric, when A has lower bandwidth p, it also has upper bandwidth p. In this case A is said to have *bandwidth* p. For example, if $p = 1$, then A is tridiagonal. The following algorithm from [2] takes advantage of the fact that A is symmetric, positive definite and has bandwidth p. It requires n square roots and

$$\frac{np^2}{2} - \frac{p^3}{3} + \frac{3}{2}(np - p^2)$$

flops or approximately $\frac{n(p^2+3p)}{2}$ flops for $p \ll n$.

FOR $i = 1, \ldots, n$
  FOR $j = \max\{1, i-p\}, \ldots, i-1$

$$a_{ij} := \left(a_{ij} - \sum_{k=\max\{1,i-p\}}^{j-1} a_{ik}a_{jk}\right) \Big/ a_{jj}$$

$$a_{ii} := \left(a_{ii} - \sum_{k=\max\{1,i-p\}}^{i-1} a_{ik}^2\right)^{1/2}$$

Cholesky Factorization, Algorithm 2
A pseudocode for the banded Cholesky factorization

Another important application of the Cholesky factorization is in the key role it plays in one of the most commonly used numerical techniques for solving the least squares problem (LS problem; cf. also ▶ least squares problems). The *least squares problem* is to find the 'best' solution to $Ax = b$ when the system is inconsistent for $A \in \mathbf{R}^{m\times n}$. Instead, the system $A^{\mathsf{T}} Ax = A^{\mathsf{T}} b$, more commonly known as the *normal equations*, is solved by first finding the Cholesky factorization of the symmetric matrix $A^{\top}A = \widetilde{L}\widetilde{L}^{\top}$, which is positive definite if A has rank n. Next, $\widetilde{L}y = A^{\top}b$ is forward-solved, and finally, the 'best least squares' solution, $\widehat{x}$, is found by backsolving $\widetilde{L}^{\top}x = y$. Note that $\widehat{x}$ minimizes $\|Ax - b\|_2$ and the algorithm requires $O(n^3)$ flops. For the algorithm and an analysis of the accuracy of the method, see [2].

See also

- ▶ ABS Algorithms for Linear Equations and Linear Least Squares
- ▶ Large Scale Trust Region Problems
- ▶ Large Scale Unconstrained Optimization
- ▶ Linear Programming
- ▶ Orthogonal Triangularization
- ▶ Overdetermined Systems of Linear Equations
- ▶ QR Factorization
- ▶ Solving Large Scale and Sparse Semidefinite Programs
- ▶ Symmetric Systems of Linear Equations

References

1. Burden RL, Faires JD (2005) Numerical Analysis, 8th edn. PWS/Kent, Boston
2. Golub GH, Loan CF Van (1996) Matrix Computations, 3rd edn. Johns Hopkins Univ. Press, Baltimore
3. Griffel DH (1989) Linear Algebra and its Applications, vol 2. Halsted Press, New York
4. Hager WW (1995) Applied Numerical Linear Algebra. Dept. Math. Univ. Florida, Gainesville
5. Householder AS (1964) The Theory of Matrices in Numerical Analysis. Dover, New York
6. Kincaid D, Cheney W (2002) Numerical Analysis, 3rd edn. Brooks/Cole, Pacific Grove, CA
7. Leon SJ (2005) Linear Algebra with Applications, 7th edn. Prentice-Hall, Upper Saddle River
8. Strang G (2003) Introduction to Linear Algebra, 3rd edn. Wellesley and Cambridge Press, Wellesley, MA
9. Wilkinson JH (1963) Rounding Errors in Algebraic Processes. Prentice-Hall, Upper Saddle River

Combinatorial Matrix Analysis

RICHARD A. BRUALDI
Department Math., University Wisconsin,
Madison, USA

MSC2000: 90C10, 90C09

Article Outline

Keywords
Matrix Patterns and Various Graphs
Eigenvalues and Digraphs
Sign-Nonsingular Matrices
Doubly Stochastic Matrices
See also
References

Keywords

Matrix analysis

Broadly speaking, *matrix analysis* is the study of non-algebraic properties of matrices and the analysis of matrices in order to reveal their finer properties and structure. (Here, 'algebraic' is understood at least in the classical sense of algebra.) The matrices are usually real or complex matrices. *Combinatorial matrix analysis* is the study of combinatorial properties of matrices and the analysis of matrices which takes into account combinatorial structure. Here combinatorial structure usually refers to the *zero-nonzero pattern* of a matrix, captured through the use of either the directed graph or bipartite graph associated with the nonzero entries of a matrix (or the graph of the nonzero entries in the case of a symmetric matrix), or to the *positive-negative-zero pattern* of a real matrix, captured through the use of the signed digraph or signed bipartite graph of a matrix.

This article is intended as an introduction to combinatorial matrix analysis. More detail can be found in [5] and [6], and the references contained therein. We do not discuss here the many applications of matrix theory and linear algebra to combinatorics, graphs, and discrete structures.

Matrix Patterns and Various Graphs

Let $A = [a_{ij}]$ be a matrix of order n whose entries a_{ij} are real or complex numbers. To A there corresponds a *directed graph* (or *digraph*) $D(A)$ with vertex set $V = \{1, \ldots, n\}$ and with an arc (i, j) from vertex i to vertex j if and only if $a_{ij} \neq 0$. The *bipartite graph* (or *bigraph*) $BG(A)$ of A has vertex set $\{1_r, \ldots, n_r\}$ (corresponding to the rows of A) and $\{1_c, \ldots, n_c\}$ (corresponding to the columns of A); the edges of $BG(A)$ are all pairs $\{i_r, j_c\}$ for which $a_{ij} \neq 0$. The bipartite graph of a matrix can be defined for a rectangular $m \times n$ matrix in the same way except that the vertices corresponding to the rows are $\{1_r, \ldots, m_r\}$. Both the digraph and the bigraph reveal the zero-nonzero pattern of a square matrix A.

If A is a real matrix and we want to capture the sign $(+, -, 0)$ of the entries of A, then we assign a $+$ or $-$ to each arc of $D(A)$ (to each edge of $BG(A)$) according as the corresponding entry of A is positive or negative, and in this way obtain the *signed digraph* and *signed bigraph* of A. We use the same notations $D(A)$ and $BG(A)$ for the signed versions of the digraph and bigraph of A. Thus two matrices A and B have the same *sign pattern* if and only if they have the same signed digraphs (equivalently, the same signed bigraphs).

If A is a symmetric matrix (or has a symmetric pattern in the sense that $a_{ij} \neq 0$ if and only if $a_{ji} \neq 0$), then the *graph* $G(A)$ of A has vertex set $\{1, \ldots, n\}$ with an edge $\{i, j\}$ between i and j if and only if $a_{ij} \neq 0$ (equivalently, $a_{ji} \neq 0$). Thus $G(A)$ is obtained from $D(A)$ by 'removing' the directions on arcs (this may result in two edges joining certain pairs of vertices and one edge of each such pair is removed as well). Sometimes in $D(A)$ and $G(A)$ it is convenient to ignore the arcs (i, i) and edges $\{i, i\}$ (called loops) corresponding to nonzero entries a_{ii} on the main diagonal of A.

A square matrix A of order n is *irreducible* provided there does not exist a permutation matrix P such that

$$PAP^{\top} = \begin{bmatrix} A_1 & O \\ A_{21} & A_2 \end{bmatrix},$$

where A_1 is a square matrix of order k for some k with $0 < k < n$. (The matrix PAP^{T} is obtained from A by simultaneously permuting its rows and columns. The digraphs of A and PAP^{T} are isomorphic.) A digraph is *strongly connected* provided for each ordered pair of distinct vertices i and j there is a path from i to j.

Proposition 1 *The matrix A is irreducible if and only if the digraph $D(A)$ is strongly connected, [5].*

A square matrix can be brought to a very special form by simultaneous row and column permutations.

Theorem 2 *Let A be a matrix of order n. Then there exist a permutation matrix P and an integer $k \geq 1$ such that*

$$PAP^{\top} = \begin{bmatrix} A_1 & O & \cdots & O \\ A_{21} & A_2 & \cdots & O \\ \vdots & \vdots & \ddots & \vdots \\ A_{k1} & A_{k2} & \cdots & A_k \end{bmatrix},$$

where $A_1, \ldots, A_k$ are square, irreducible matrices. The matrices $A_1, \ldots, A_k$ are uniquely determined to within simultaneous permutations of their rows and columns, but their order on the diagonal is not necessarily unique.

The matrices $A_1, \ldots, A_k$ in this theorem are called the *irreducible components* of A and correspond to the *strongly connected components* of the digraph $D(A)$.

Irreducible matrices have an *inductive structure* that is revealed in the next theorem [5].

Theorem 3 *Let A be an irreducible matrix of order $n \geq 2$. Then there exist a permutation matrix P and an integer $m \geq 2$ such that*

$$PAP^{\top} = \begin{bmatrix} A_1 & O & \cdots & O & E_1 \\ E_2 & A_2 & \cdots & O & O \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ O & O & \cdots & A_{m-1} & O \\ O & O & \cdots & E_m & A_m \end{bmatrix},$$

where $A_1, \ldots, A_m$ are irreducible matrices and $E_1, \ldots, E_m$ are matrices having at least one nonzero entry.

Allowing independent row and column permutations in the definition of irreducibility leads to full indecomposability. A square matrix A of order n is *fully indecomposable* provided there do not exist permutation matrices P and Q such that

$$PAQ = \begin{bmatrix} A_1 & O \\ A_{21} & A_2 \end{bmatrix},$$

where A_1 is a square matrix of order k for some k with $0 < k < n$. The matrices A and PAQ have isomorphic bigraphs.

Theorems analogous to Theorems 2 and 3 hold with independent permutations replacing simultaneous permutations and fully indecomposable replacing irreducible. The connection is provided by the fact that a square matrix A is fully indecomposable if and only if there are permutation matrices P and Q such that PAQ has a nonzero main diagonal and PAQ is irreducible [5].

Eigenvalues and Digraphs

The following theorem is the Perron–Frobenius theorem [7,8] and is one of the first instances of the influence of the digraph of a matrix on its spectral properties.

Theorem 4 (*Perron–Frobenius theorem*) *Let A be a matrix of order $n > 1$ each of whose entries is a nonnegative real number. Assume that A is irreducible, equivalently $D(A)$ is strongly connected. Then there is a positive number $\rho(A)$ such that*

1) *$\rho(A)$ is a simple eigenvalue of A;*
2) *every eigenvalue λ of A satisfies $|\lambda| \leq \rho(A)$;*
3) *the number of eigenvalues λ of A with $|\lambda| = \rho(A)$ equals the greatest common divisor k of the lengths of the circuits of $D(A)$, and these eigenvalues are $\rho(A)e^{2\pi ij/k}$, $(j = 1, \ldots, k)$.*

A more recent application of the digraph of a matrix to localization of its eigenvalues concerns a generalization [2] of the *Gershgorin theorem*. Let $A = [a_{ij}]$ be a complex matrix of order n and let

$$R_i = \sum_{j \neq i} |a_{ij}| \qquad (1 \leq i \leq n).$$

Then Gershgorin's theorem asserts that the n eigenvalues of A lie in that part $\mathcal{R}$ of the complex plane determined by the union of the n closed disks

$$\{z\colon\ |z - a_{ii}| \leq R_i\}, \qquad (1 \leq i \leq n).$$

If A is irreducible, then a boundary point of $\mathcal{R}$ is an eigenvalue of A only if it is a boundary point of each of the n closed disks.

By considering the *circuits*, or *directed cycles*, of the digraph of A, a better inclusion region can be obtained.

Theorem 5 *Let $A = [a_{ij}]$ be a complex matrix of order n. Then the n eigenvalues of A lie in that part S of the*

complex plane determined by the union of the regions

$$S(\gamma) = \left\{ z\colon \prod_{\gamma} |z - a_{ii}| \le \prod_{\gamma} R_i \right\},$$

(γ *a circuit of* $D(A)$),

where $\prod_\gamma$ *denotes the product over all vertices i belonging to the circuit* γ. *If A is irreducible, then a boundary point of S is an eigenvalue of A only if it is a boundary point of each* $S(\gamma)$.

Theorems 4 and 5 demonstrate how information concerning the combinatorial structure of a matrix can be used to give information on spectral properties of the matrix.

Sign-Nonsingular Matrices

It is easy to characterize real matrices $A = [a_{ij}]$ of order n whose singularity is a consequence of their zero pattern, equivalently, of their nonzero pattern or bigraph $BG(A)$. Let $\mathcal{Z}(A)$ denote the set of all real matrices B of order n that have the same zero pattern as A, that is, satisfy $BG(B) = BG(A)$. Then the following are equivalent:

i) Each matrix $B \in \mathcal{Z}(A)$ is singular;
ii) Each of the $n!$ terms in the standard determinant expansion of A is zero (the *standard determinant expansion of a matrix A* is

$$\det A = \sum \epsilon(i_1, \ldots, i_n) a_{1i_1} \cdots a_{ni_n}$$

where the sum extends over each permutation $i_1 \ldots i_n$ of $\{1, \ldots, n\}$ and $\epsilon(i_1 \ldots i_n)$ is + or − depending on whether the permutation is even or odd);
iii) The bigraph $BG(A)$ does not have a perfect matching (i. e. a set of n pairwise vertex disjoint edges meeting all vertices);
iv) There is a set of fewer than n rows and columns which together contain all the nonzero entries of A;
v) There is a set of fewer that n vertices of $BG(A)$ which together meet all the edges of $BG(A)$.

Properties ii) and iii) are clearly equivalent, as are properties iv) and v). Properties i) and ii) are equivalent, since if there is a nonzero term in the standard determinant expansion of A, then by sufficiently emphasizing the entries of A in that term we obtain a nonsingular matrix. Properties iii) and iv) are equivalent by the Frobenius–König theorem [5].

Now let $\mathcal{Q}(A)$ denote the set of all real matrices of order n that have the same sign pattern (+, −, 0) as A. $\mathcal{Q}(A)$ consists of all real matrices of order n that have the same signed digraph (equivalently, the same signed bigraph) as A and is called the *qualitative class* of A. The matrix A is called *sign-nonsingular* provided each matrix in $\mathcal{Q}(A)$ is nonsingular. Some equivalent characterizations of sign-nonsingularity are:

i) A is sign-nonsingular;
ii) There is a nonzero term in the standard determinant expansion of A and each such nonzero term has the same sign;
iii) $\det(A) \neq 0$ and the determinants of the matrices in $\mathcal{Q}(A)$ all have the same sign.

The matrix

$$\begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix}$$

is a sign-nonsingular matrix.

If a matrix $A = [a_{ij}]$ is sign-nonsingular, then so is every matrix PAQ where P and Q are permutation matrices, as is every matrix of the form DA where D is a nonsingular diagonal matrix. Also a sign-nonsingular matrix must have a nonzero term in its standard determinant expansion. Thus in dealing with sign-nonsingular matrices we may assume that each entry on the main diagonal is negative, that is, A has a *negative main diagonal*. With this normalization, we have the following theorem [1]. The *sign of a circuit*

$$\gamma\colon j_1 \to \cdots \to j_k \to j_1$$

of the signed digraph of A is

$$\operatorname{sign}(\gamma) = \operatorname{sign} a_{j_1 j_2} \cdots a_{j_{k-1} j_k} a_{j_k j_1},$$

the products of the signs of the arcs of the cycle.

Theorem 6 (*Bassett–Maybee–Quirk theorem*) *Let A be a real matrix of order n with a negative main diagonal. Then A is a sign-nonsingular matrix if and only if each circuit of the signed digraph of A is negative.*

Sign-nonsingularity allows one to characterize square, homogeneous systems of linear equations $Ax = 0$ for which $\widetilde{A}x = 0$ has only the zero solution (thus all solutions of $\widetilde{A}x = 0$ have the same sign pattern) for all matrices $\widetilde{A}$ with the same sign pattern as A. A more general

problem is to characterize linear systems $Ax = b$ that are sign-solvable in the sense that the sign pattern of the solution is determined solely by the sign patterns of A and b. More precisely, $Ax = b$ is *sign-solvable* provided that for all $\widetilde{A} \in \mathcal{Q}(A)$ and all $\widetilde{b} \in \mathcal{Q}(b)$ there is a vector $\widetilde{x}$ such that $\widetilde{A}\widetilde{x} = \widetilde{b}$ and all of the vectors in

$$\left\{\widetilde{x}\colon\ \exists \widetilde{A} \in \mathcal{Q}(A), \widetilde{b} \in \mathcal{Q}(b) \text{ s.t. } \widetilde{A}\widetilde{x} = \widetilde{b}\right\}$$

have the same sign pattern.

Sign-solvable linear systems can be characterized in terms of two classes of matrices, called S^*-matrices and L-matrices. An $n \times (n+1)$ matrix B is an *S^*-matrix* provided each matrix of order n obtained from B by deleting a column is a sign-nonsingular matrix. Cramer's rule implies that B is an S^*-matrix if and only if there is a vector w with no zero coordinates such that the right null spaces of the matrices in $\widetilde{B} \in \mathcal{Q}(B)$ are contained in $\{0\} \cup \mathcal{Q}(w) \cup \mathcal{Q}(-w)$. The matrix

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & 1 & 1 & -1 \end{bmatrix}$$

is an S^*-matrix. A matrix A is an *L-matrix* provided every matrix in $\mathcal{Q}(A)$ has linearly independent rows. Sign-nonsingular matrices are square L-matrices. Every S^*-matrix is an L-matrix and so is any matrix obtained from an L-matrix by appending columns.

The following theorem characterizes sign-solvable linear systems [9].

Theorem 7 *Let $A = [a_{ij}]$ be an $m \times n$ matrix and let b be an $m \times 1$ column vector. Let $z = (z_1, \ldots, z_n)^\mathsf{T}$ be a solution of the linear system $Ax = b$, and let*

$$\beta = \{j\colon\ z_j \neq 0\},$$
$$\alpha = \{i\colon\ a_{ij} \neq 0 \quad \textit{for some } j \in \beta\}.$$

Then $Ax = b$ is sign-solvable if and only if the matrix

$$[A[\alpha, \beta] - b[\beta]]$$

is an S^*-matrix and the matrix $A(\alpha, \beta)^\mathsf{T}$ is an L-matrix.

(Here $A[\alpha, \beta]$, respectively $A(\alpha, \beta)$, is the submatrix of A formed by the rows in α and the columns in β, respectively not in α and not in β, and $b[\alpha] = b[\alpha, \{1\}]$.)

A detailed study of sign-solvability and related issues is contained in [6].

Doubly Stochastic Matrices

A real matrix $A = [a_{ij}]$ of order n is *doubly stochastic* provided each of its entries is nonnegative, and all row and column sums equal 1:

$$a_{ij} \geq 0 \quad (i, j = 1, \ldots, n),$$
$$\sum_{j=1}^{n} a_{ij} = 1, \quad \sum_{i=1}^{n} a_{ij} = 1.$$

Doubly stochastic matrices arise quite naturally in many different contexts:

i) Let $U = [u_{ij}]$ be a real orthogonal matrix or a complex unitary matrix. Then
$$\widehat{U} = [|u_{ij}|^2] \qquad (i, j = 1, \ldots, n)$$
is a doubly stochastic matrix.

ii) (*Optimal assignment problem*) Consider an assignment of n people to n positions in which the 'value' of the ith person to the jth position is $v_{ij} \geq 0$ $(i, j = 1, \ldots, n)$. An *optimal assignment* is an assignment $i \to j_i$ $(i = 1, \ldots, n)$ of people to positions (here $j_1 \ldots j_n$ is a permutation of $\{1, \ldots, n\}$) which maximizes the total value $\sum_{i=1}^{n} v_{ij_i}$. The set Ω_n of doubly stochastic matrices of order n is a convex polytope and, according to *Birkhoff's theorem* [5], the set of vertices of this polytope is the set $\mathcal{P}_n$ of permutation matrices of order n. Thus the vertices of Ω_n correspond to the $n!$ possible assignments, and the optimal assignment problem can be solved as a linear programming problem on Ω_n.

iii) Let $\mathcal{P}_n = \{P_1, \ldots, P_{n!}\}$, and let $(c_i\colon\ i = 1, \ldots, n!)$ be a probability distribution on $\mathcal{P}_n$: $c_i \geq 0$ $(i = 1, \ldots, n!)$ and $\sum_{i=1}^{n!} c_i = 1$. Then the expectation of a permutation $R \in \mathcal{P}_n$ chosen at random is
$$E = E[R] = \sum_{i=1}^{n!} c_i P_i = [e_{ij}],$$
a doubly stochastic matrix. It is a consequence of *Birkhoff's theorem* that every doubly stochastic matrix of order n arises from a probability distribution on $\mathcal{P}_n$ in this way. The probability that a function f chosen at random according to the probabilities
$$\text{prob}(f(i) = j) = e_{ij} \qquad (i, j = 1, \ldots, n)$$
is a permutation equals the *permanent* of A defined by
$$\text{per}(A) = \sum a_{1j_1} \cdots a_{nj_n},$$

where the sum extends over all permutations $j_1 \ldots j_n$ of $\{1, \ldots, n\}$.

Let $A = [a_{ij}]$ be a real, symmetric matrix (or a complex, Hermitian matrix). Then there exists a real, orthogonal matrix U such that

$$UAU^\top = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix},$$

where $\lambda_1 \geq \ldots \geq \lambda_n$ are the n eigenvalues of A. Comparing diagonal entries, we get

$$(\lambda_1, \ldots, \lambda_n)^\top = S(a_{11}, \ldots, a_{nn})^\top, \tag{1}$$

where $S = \widehat{U}$ is a doubly stochastic matrix. Without loss of generality, assume that $a_{11} \geq \cdots \geq a_{nn}$. Then equation (1) implies that

$$\lambda_1 + \cdots + \lambda_i \leq a_{11} + \cdots + a_{ii} \quad (i = 1, \ldots, n), \tag{2}$$

with equality for $i = n$. When the inequalities (2), with equality for $i = n$, hold between two vectors $\lambda = (\lambda_1, \ldots, \lambda_n)$ and $\mu = (a_{11}, \ldots, a_{nn})$ (that have been arranged in nonincreasing order), then λ is said to be *majorized* by μ. A *Hardy–Littlewood–Pólya theorem* states that if λ is majorized by μ, then there exists a doubly stochastic matrix S such that $\lambda = S\mu$ [10]. Hence by Birkhoff's theorem, λ is majorized by μ if and only if λ is in the convex hull of all vectors obtained from μ by permuting its coordinates. There exist doubly stochastic matrices S of very special form such that $\lambda = S\mu$ when λ is majorized by μ [4,10].

As noted above, the vector of eigenvalues of a real, symmetric matrix is majorized by the vector of its entries on the main diagonal. Conversely, if λ and μ are two n-vectors with λ majorized by μ, then according to a theorem of A. Horn, there exists a real, symmetric matrix of order n, whose eigenvalues are given by λ and whose main diagonal entries are given by μ [10].

Let A be a doubly stochastic matrix, and let $A_1, \ldots, A_k$ $(k \geq 1)$ be the fully indecomposable components of A. Since all row and column sums of A equal 1, it follows easily that up to row and column permutations A is the direct sum of its fully indecomposable components: there exist permutation matrices P and Q such that

$$PAQ = A_1 \oplus \cdots \oplus A_k,$$

where $\oplus$ denotes direct sum. The polytope Ω_n has dimension $(n-1)^2$. Each doubly stochastic matrix determines a face of Ω_n equal to the set of all doubly stochastic matrices S such that $BG(S)$ is a subgraph of $BG(A)$ (i. e. each edge of $BG(A)$ is also an edge of $BG(A)$). This face is the smallest face of Ω_n containing A, and each nonempty face of Ω_n arises in this way. Since no entry of a doubly stochastic matrix can exceed 1, the nonempty faces of Ω_n can be described as follows: Let C be a (0, 1)-matrix of order n which, up to row and column permutations, is a direct sum of fully indecomposable matrices (such matrices are said to have *total support*). Then

$$\mathcal{F}(C) = \{A\colon\ A \in \Omega_n, A \leq C \text{ entrywise}\}$$

is a face of Ω_n and its dimension equals $\sigma(C) - 2n + k$, where $\sigma(C)$ is the number of 1s of C and k is the number of its fully indecomposable components [3].

See also

- Combinatorial Optimization Algorithms in Resource Allocation Problems
- Combinatorial Optimization Games
- Evolutionary Algorithms in Combinatorial Optimization
- Fractional Combinatorial Optimization
- Multi-Objective Combinatorial Optimization
- Neural Networks for Combinatorial Optimization
- Replicator Dynamics in Combinatorial Optimization

References

1. Bassett L, Maybee J, Quirk J (1968) Qualitative economics and the scope of the correspondence principle. Econometrica 36:544–63
2. Brualdi RA (1982) Matrices, eigenvalues and directed graphs. Linear Multilinear Algebra 11:143–65
3. Brualdi RA, Gibson PM (1977) The convex polytope of doubly stochastic matrices I: Applications of the permanent function. J Combin Th A 22:194–230
4. Brualdi RA, Hwang S-G (1996) Vector majorization via Hessenberg matrices. J London Math Soc Ser 2 53:28–38
5. Brualdi RA, Ryser HJ (1991) Combinatorial matrix theory. Encycl Math Appl, vol 39. Cambridge Univ. Press, Cambridge
6. Brualdi RA, Shader BL (1995) Matrices of sign-solvable linear systems. Cambridge Tracts in Math, vol 116. Cambridge Univ. Press, Cambridge

7. Horn R, Johnson CR (1985) Matrix analysis. Cambridge Univ Press, Cambridge
8. Horn R, Johnson CR (1991) Topics in matrix analysis. Cambridge Univ Press, Cambridge
9. Klee V, Ladner R, Manber R (1984) Sign solvability revisited. Linear Alg Appl 59:131–57
10. Olkin I, Marshall AW (1979) Inequalities: Theory of majorization and its applications. Acad Press, New York

Combinatorial Optimization Algorithms in Resource Allocation Problems

NAOKI KATOH
Kyoto University, Kyoto, Japan

MSC2000: 90C09, 90C10

Article Outline

Keywords
α: Objective Functions
β: Constraints
Algorithms
Generalizations
See also
References

Keywords

Resource allocation; Combinatorial algorithm

The *resource allocation problem* seeks to find an optimal allocation of a fixed amount of resources to activities so as to minimize the cost incurred by the allocation. A simplest form of the problem is to minimize a separable convex function under a single constraint concerning the total amount of resources to be allocated. The amount of resources to be allocated to each activity is treated as a continuous or integer variable, depending on the cases. This can be viewed as a special case of the nonlinear programming problem or the nonlinear integer programming problem.

Due to its simple structure, this problem is encountered in a variety of application areas, including load distribution, production planning, computer resource allocation, queueing control, portfolio selection, and apportionment. The first explicit investigation of the resource allocation problem is due to B.O. Koopman [15] (1953), who dealt with the problem of the *optimal distribution of efforts*, which arises in the problem of searching for an object whose position is a random variable. Since then, a great number of papers have been published on resource allocation problems. Efficient algorithms have also been developed, depending on the form of objective functions and constraints or on the type of variables (i. e., continuous or integer).

See [11] for a comprehensive review of the state-of-the-art of the problems (as of 1988). After this book was published, many papers have been published on resource allocation problems. A significant progress has been made on the algorithm side. Also, new generalizations and variants of the problem have been investigated, and new application fields have been discovered. Such new progress has been reviewed in [13].

We first classify the resource allocation problems. A generic form of the resource allocation problem discussed in this article is described as follows:

$$\text{(P)} \quad \begin{cases} \min & f(x_1, \ldots, x_n) \\ \text{s.t.} & \sum_{j=1}^{n} x_j = N, \\ & x_j \geq 0, \quad j = 1, \ldots, n. \end{cases} \tag{1}$$

That is, given one type of resource whose total amount is equal to N, we want to allocate it to n activities so that the objective value $f(x_1, \ldots, x_n)$ is minimized. The objective value may be interpreted as the cost or loss, or the profit or reward, incurred by the resulting allocation. In case of profit or reward, it is natural to maximize f, and we shall sometimes consider maximization problems. The difference between maximization and minimization is not essential because maximizing f is equal to minimizing $-f$.

Each variable x_j represents the amount of resource allocated to activity j. If it represents persons, processors or trucks, however, variable x_j becomes a discrete variable that takes nonnegative integer values, and the constraint

$$x_j : \text{ integer}, \quad j = 1, \ldots, n, \tag{2}$$

is added to the constraints in (1). The resource allocation problem with this constraint is often referred to as the *discrete resource allocation problem*.

As for the objective function, it usually has some special structure according to the intended applications. Typically, the following special case, called *separable*, is often considered:

$$\sum_{j=1}^{n} f_j(x_j). \tag{3}$$

If each f_j is convex, the objective function is called *separable convex objective function*.

Resource allocation problems are classified according to the types of objective functions, constraints and variables. We shall describe the classification scheme, and several types of problem formulations according to the classification scheme. In general, we use the notation $\alpha/\beta/\gamma$ to denote the type of a resource allocation problem. Here, α specifies the type of objective function, β the constraint type, and γ the variable type;

$$\gamma = D, \quad \gamma = C$$

denote the case of integer variable, respectively continuous variable. We shall now explain the notations for α and β.

α: Objective Functions

The objective function $f(x_1, \ldots, x_n)$ may take the following special structures:

1) *Separable* (S, for short): $\sum_{j=1}^{n} f_j(x_j)$, where each f_j is a function of one variable.
2) *Separable and convex* (SC, for short): $\sum_{j=1}^{n} f_j(x_j)$, where each f_j is a convex function of one variable. In particular, if each f_j is quadratic and convex, we denote such a subclass by SQC.
3) *Minimax*: minimize $\max_{1 \le j \le n} f_j(x_j)$, or *Maximin*: maximize $\min_{1 \le j \le n} f_j(x_j)$; here, all f_j are monotone nondecreasing in x_j.
4) *Lexicographically minimax* (Lexico-Minimax, for short): Since the objective value of Minimax is determined by the single variable x_k^* satisfying $f_k(x_k^*) = \max_j f_j(x_j^*)$, there may be many optimal solutions. To remove such ambiguity, we introduce the *lexicographical ordering for n-dimensional vectors*: Given $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$, a is *lexicographically smaller* than b (or b is *lexicographically greater* than a) if $a_j = b_j$ for $j = 1, \ldots, k-1$ and $a_k < b_k$ some k. This is denoted by $a \le_{\text{lex}} b$ or $b \ge_{\text{lex}} a$. For $a = (a_1, \ldots, a_n)$, let DEC(a) (respectively, INC(a)) denote the n-tuple of $a_j, j = 1, \ldots, n$, arranged in nonincreasing order (respectively, nondecreasing order) of their values (e. g., for $a = (4, 3, 1, 5)$, we have DEC(a) = (5, 4, 3, 1) and INC(a) = (1, 3, 4, 5)). The objective of Lexico-Minimax is to find an allocation vector $x = (x_1, \ldots, x_n)$ such that DEC(x) is minimal. Notice that an optimal solution to Lexico–Minimax is also optimal to Minimax, but the converse is not generally true. This is a refined objective of Minimax. Similarly, we define Lexico-Maximin as the one that maximizes INC(x).
5) *Fair*: minimize the expression

$$g\left(\max_{1 \le j \le n} f_j(x_j), \min_{1 \le j \le n} f_j(x_j)\right),$$

where $g(u, v)$ is nondecreasing (respectively, nonincreasing) in u (respectively, v). This objective is a generalization of Minimax and Maximin.

β: Constraints

In addition to the simple first resource constraint of (1), other additional constraints are also imposed. Typical additional constraints which appeared in various resource allocation problems are as follows. We refer the case of no additional constraints as 'simple'.

1) *Lower and upper bounds* (LUB, for short): $l_j \le x_j \le u_j$, j = 1, …, n.
2) *Generalized upper bounds* (GUB, for short): $\sum_{j \in S_i} x_j \le b_i$, $i = 1, \ldots, m$, where $S_1, \ldots, S_m$ is a partition of $\{1, \ldots, n\}$.
3) *Nested constraints* (Nested, for short): $\sum_{j \in S_i} x_j \le b_i$, $i = 1, \ldots, m$, where $S_1 \subset \ldots \subset S_m$. We can assume $b_1 \le \ldots \le b_m$, since if $b_i > b_{i+1}$, the ith constraint is redundant.
4) *Tree constraints* (Tree, for short): $\sum_{j \in S_i} x_j \le b_i$, $i = 1, \ldots, m$, where the sets S_i are derived by some hierarchical decomposition of E into disjoint subsets.
5) *Network constraints* (Network, for short): The constraint is defined in terms of a directed network with a single source and multiple sinks. Given a directed graph $G = (V, A)$ with node set V and arc set A, let $s \in V$ be the source and $T \subseteq V$ be the set of sinks. The amount of supply from the source is $N > 0$, and

the capacity of arc (u, v) is $c(u, v)$. Denote the flow vector by $\varphi = \{\varphi(u, v):(u, v) \in A\}$. φ is a *feasible flow* in G if it satisfies

$$0 \le \varphi(u,v) \le c(u,v), \quad (u,v) \in A,$$
$$\sum_{(v,w)\in A_+(v)} \varphi(v,w) - \sum_{(u,v)\in A_-(v)} \varphi(u,v) = 0\ , \tag{4}$$

$$v \in V - T - \{s\}\ , \tag{5}$$

$$\sum_{(s,v)\in A_+(s)} \varphi(s,v) - \sum_{(u,s)\in A_-(s)} \varphi(u,s) = N\ , \tag{6}$$

$$x_t(\varphi) \equiv \sum_{(u,t)\in A_-(t)} \varphi(u,t) - \sum_{(t,v)\in A_+(t)} \varphi(t,v) \ge 0, \quad t \in T\ , \tag{7}$$

$$\sum_t x_t(\varphi) = N\ . \tag{8}$$

The value $x_t(\varphi)$ denotes the amount of flow entering a sink $t \in T$. For a feasible flow φ, the vector $x_t(\varphi)x \in T\}$ is called the *feasible flow vector* with respect to φ. For instance, the problem SC/Network/C (i. e., the separable convex resource allocation problem under network constraints) is defined as follows:

$$\begin{cases} \min & \sum_{t\in T} f_t(x_t(\varphi)) \\ \text{s.t.} & (4) - (8)\ , \end{cases} \tag{9}$$

where f_t, for each $t \in T$, is a convex function.

6) *Submodular constraints* (SM, for short): A set of feasible solutions is defined by a base polyhedron $B(r) = x \in \mathbf{R}^E : x(S) \le r(S)$ for all $S \in \mathcal{D}$, $x(E) = r(E)\}$ of a submodular system $(\mathcal{D}, r)$, i. e.,

$$x \in B(r)\ . \tag{10}$$

Here, we use the notation $E = \{1, \ldots, n\}$, and $x(S) \equiv \sum_{i\in S} x_i$ for $S \subseteq E$ and $x \in \mathbf{R}^E$. $\mathcal{D} \subseteq 2^E$ is a *distributive lattice* such that $\emptyset, E \in \mathcal{D}$, i. e., $\mathcal{D}$ is closed under union and intersection operations. Also, the function $r : \mathcal{D} \to \mathbf{Z}$ is *submodular* over $\mathcal{D}$, i. e.,

$$r(X) + r(Y) \ge r(X \cup Y) + r(X \cap Y)\ .$$

For a *submodular system* $(\mathcal{D}, r)$,

$$P(r) = \{x \in \mathbb{R}^E :\ x(S) \le r(S) \text{ for all } S \in \mathcal{D}\}$$

is called the *submodular polyhedron* of $(\mathcal{D}, r)$.

Notice that the first constraint in (1) is included in the constraints of (10), as $x(E) = r(E)$ in the above definition. If we consider the case of integer variables, the constraint is defined by

$$x \in B(r) \cap \mathbb{Z}^E.$$

It is assumed, in general, that $B(r)$ of the constraint (10) is not explicitly given as an input, but is implicitly given through an *oracle* that tells the value $r(X)$ when X is given.

7) *General linear constraints* (Linear, for short): Constraints defined by a set of linear inequalities

$$\sum_{j=1}^{n} a_{ij}x_j \le b_i, \quad i = 1, \ldots, m. \tag{11}$$

No other special assumption is imposed on the structure of the constraints.

Notice that all the constraints, LUB, GUB, Nested, Tree, Network are special cases of submodular constraints (see [11]), and SM is a special case of Linear.

Algorithms

We first introduce an *incremental algorithm* for the simple resource allocation problem, SC/Simple/D. We assume that each f_j is defined over the interval $[0, N]$. Since f_j is convex, we have

$$d_j(1) \le \cdots \le d_j(N), \tag{12}$$

where

$$d_j(y_j) = f_j(y_j) - f(y_j - 1).$$

The incremental algorithm is a kind of *greedy algorithm*, and is also called a *marginal allocation* method. Starting with the initial solution $x = (0, \ldots, 0)$, one unit of resource is allocated at each iteration to the most favorable activity (in the sense of minimizing the increase in the current objective value) until $\sum x_j = N$ is attained.

Input: An instance of SC/Simple/D.
Output: An optimal solution x^*.

Let $x := (0, \ldots, 0)$ and $k := 0$;
WHILE $k < N$ DO
  Find j^* such that
  $d_{j^*}(x_{j^*} + 1) = \min_{1 \le j \le n} d_j(x_j + 1)$;
  $x_{j^*} := x_{j^*} + 1$;
  $k := k + 1$
END;
Output x as x^*.

Procedure INCREMENT

It has been shown this procedure correctly computes an optimal solution in $O(N \log n + n)$ time.

Several polynomial time algorithms have been developed for problem SC/Simple/D [3,7,14]. The fastest among them is proposed in [3]. Its running time is $O(\max\{n, n \log(N/n)\})$, but the algorithm is very complicated. All these algorithms are based on *divide-and-conquer*.

The incremental algorithm presented above also works for problem SC/SM/D. In this case, among all the elements such that $x + e(j) \in P(r)$ (i. e., feasible except for the constraint $(x + e(j))(E) = r(E)$), the x_j with the minimum increase in $f_j(x_j)$ is incremented by one. This process is repeated until $x(E) = r(E)$ is finally attained.

A polynomial time algorithm for SC/SM/D is also known [2,4]. It first solves a problem of SC/Simple/D type, which is obtained from the original problem by by considering only the simple constraint $x(E) = r(E)$ but disregarding the rest. If the obtained solution y is feasible, we are done, i. e., it is an optimal solution of the original problem. Otherwise, the problem is decomposed into subproblems using the information obtained from the vector y and the submodular constraints.

When specialized to problem SC/Network/D, the running time becomes $O(|\mathrm{T}|(\tau(n, m, C_{\max}) + |\mathrm{T}| \log(N|\mathrm{T}|)))$, where $\tau(n, m, C_{\max})$ denotes the running time for the maximum flow algorithm for a graph with n vertices, m arcs and the maximum arc capacity $C_{\max}$. The direct consequence of this result is that problems SC/GUB/D, SC/Nested/D and SC/Tree/D can be solved in $O(n^2 \log(Nn))$ time. For SC/Nested/D, the running time was improved to $O(n \log n \log(N/n))$ in [8]. The idea of the improvement is based on a general and beautiful proximity theorem between integral and continuous optimal solutions for SC/SM/D and SC/SM/C.

For SQC/–/D with – equal to Simple, GUB, Nested, Tree or Network, D.S. Hochbaum and S. Hong [9] developed improved algorithms based on proximity result between SQC/–/C and SQC/–/D and efficient algorithms for SQC/–/C.

Minimax/–/D and Maximin/–/D, are equivalently transformed into problems of SC/–/D. Therefore, equally efficient algorithms can be developed for minimax and maximin problems. The transformation is done as follows: We only show this fact for the most general case, i. e., Minimax/SM/D, which are described as follows:

$$\text{MINIMAX} \begin{cases} \min & \max\limits_{j \in E} f_j(x_j), \\ \text{s.t.} & x \text{ is an integral base of } B(r). \end{cases}$$

Here, all f_j, $j \in E$, are assumed to be nondecreasing. When all f_j are nonincreasing, problems MINIMAX and MAXIMIN are mutually transformed into MAXIMIN and MINIMAX, respectively, by the following identities:

$$-\min_x \max_{j \in E} f_j(x_j) = \max_x \min_{j \in E} -f_j(x_j),$$
$$-\max_x \min_{j \in E} f_j(x_j) = \min_x \max_{j \in E} -f_j(x_j).$$

Define for $j \in E$,

$$g_j(x_j) = \sum_{y=0}^{x_j} f_j(y), \quad x_j = 0, 1, \ldots. \tag{13}$$

Note that

$$g_j(x_j) - g_j(x_j - 1) = f_j(x_j)$$

holds for each $x_j = 0, 1, \ldots$. From the nondecreasingness of f_j, it follows that g_j is convex over the nonnegative integers. Now consider the following problems of SC/SM/D:

$$Q_g: \min \left\{ \sum_{j \in E} g_j(x_j) : x \in B(r) \cap Z^E \right\}.$$

It is then shown that an optimal solution of problem Q_g is optimal to MINIMAX.

Generalizations

We finally note a recent development. K. Ando, S. Fujishige and T. Naitoh [1,6] considered the separable convex resource allocation problem for a *bisubmodular system* and for a *finite jump system*, whose underlying constraint can be viewed as a generalization of the submodular constraint. They developed greedy algorithms for such problems. For the case of a bisubmodular system, a polynomial time algorithm has been given in [5]. Also, Hochbaum and J.G. Shanthikumar [10] showed that, for a class of general linear constraints, efficient algorithms can be developed. The running time of their algorithm depends on the maximum absolute value of the subdeterminants, Δ, and if $\Delta = 1$ (i. e., the constraint matrix is *totally unimodular*), the running time becomes polynomial. The idea is based on the proximity result between the integral and continuous optimal solutions. When $\Delta = 1$, V.V. Karzanov and S.T. McCormick [12] proposed another polynomial time algorithm.

In addition to these efforts to generalize the constraints, new progress has recently been made towards generalizing objective functions for which efficient algorithms can still be developed. This research was done by K. Murota [16,17] who identified a subclass of nonseparable convex functions, M-convex functions, which is defined on the base polyhedron of a submodular system as follows.

A function $f : \mathbf{Z}^E \to \mathbf{R} \cup \{\infty\}$ is said to be *M-convex* if it satisfies the following property:

- (M-EXC): For any x, $y \in \operatorname{dom} f$ and for any $i \in \operatorname{supp}^+(x-y)$, there exists a $j \in \operatorname{supp}^-(x-y)$ such that

$$f(x) + f(y) \geq f(x - e(i) + e(j)) + f(y + e(i) - e(j)),$$

where

$$\operatorname{dom} f = \{x \in \mathbb{Z}^E : \ f(x) < +\infty\},$$
$$\operatorname{supp}^+(x-y) = \{k \in E : \ x_k > y_k\},$$
$$\operatorname{supp}^-(x-y) = \{k \in E : \ x_k < y_k\}.$$

The M-convex functions can enjoy nice theorems of *discrete convex analysis* in a parallel manner to the traditional convex analysis. A polynomial time algorithm has been developed for this class of problems [18].

See also

- ▶ Combinatorial Matrix Analysis
- ▶ Combinatorial Optimization Games
- ▶ Competitive Facility Location
- ▶ Evolutionary Algorithms in Combinatorial Optimization
- ▶ Facility Location with Externalities
- ▶ Facility Location Problems with Spatial Interaction
- ▶ Facility Location with Staircase Costs
- ▶ Fractional Combinatorial Optimization
- ▶ Global Optimization in Weber's Problem with Attraction and Repulsion
- ▶ MINLP: Application in Facility Location-Allocation
- ▶ Multifacility and Restricted Location Problems
- ▶ Multi-Objective Combinatorial Optimization
- ▶ Network Location: Covering Problems
- ▶ Neural Networks for Combinatorial Optimization
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Production-distribution System Design Problem
- ▶ Replicator Dynamics in Combinatorial Optimization
- ▶ Resource Allocation for Epidemic Control
- ▶ Simple Recourse Problem
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-Objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-Objective Rectilinear Distance Location
- ▶ Stochastic Transportation and Location Problems
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Ando K, Fujishige S, Naitoh T (1994) A greedy algorithm for minimizing a separable convex function over an integral bisubmodular polyhedron. J Oper Res Soc Japan 37:188–196
2. Federguruen A, Groenevelt H (1986) The greedy procedure for resource allocation problems - necessary and sufficient conditions for optimality. Oper Res 34:909–918
3. Frederickson GN, Johnson DB (1982) The complexity of selection and ranking in X +Y and matrices with sorted columns. J Comput Syst Sci 24:197–208
4. Fujishige S (1980) Lexicographically optimal base of a polymatroid with respect to a weight vector. Math Oper Res 21:186–196

5. Fujishige S (1997) A min-max theorem for bisubmodular polyhedra. SIAM J Discret Math 10:294–308
6. Fujishige S, Naitoh T (1995) A greedy algorithm for minimizing a separable convex function over a finite jump system. J Oper Res Soc Japan 38:362–375
7. Galil Z, Megiddo N (1979) A fast selection algorithm and the problem of optimum distribution of effort. J ACM 26:58–64
8. Hochbaum DS (1994) Lower and upper bounds for the allocation problem and other nonlinear optimization problems. Math Oper Res 19:390–409
9. Hochbaum DS, Hong S (1995) About strongly polynomial time algorithms for quadratic optimization over submodular constraints. Math Program 55:269–309
10. Hochbaum DS, Shanthikumar JG (1990) Nonlinear separable optimization is not much harder than linear optimization. J ACM 37:843–862
11. Ibaraki T, Katoh N (1988) Resource allocation problems: Algorithmic approaches. MIT, Cambridge, MA
12. Karzanov AV, McCormick ST (1997) Polynomial methods for separable convex optimization in totally unimodular linear spaces with applications. SIAM J Comput 26:1245–1275
13. Katoh N, Ibaraki T (1998) Resource allocation problems. In: Du D-Z, Pardalos PM (eds) Handbook Combinatorial Optim., vol 2. Kluwer, Dordrecht
14. Katoh N, Ibaraki T, Mine H (1979) A polynomial time algorithm for the resource allocation problem with a convex objective function. J Oper Res Soc 30:159–260; 449–455
15. Koopman BO (1953) The optimum distribution of effort. Oper Res 1:52–63
16. Murota K (1996) Convexity and Steinitz's exchange property. Adv Math 124:272–311
17. Murota K (1998) Discrete convex analysis. Math Program 83:313–371
18. Shioura A (1998) Minimization of an M-convex function. Discrete Appl Math 84:215–220

Combinatorial Optimization Games

CRPM

Xiaotie Deng
City University Hong Kong, Kowloon, China

MSC2000: 91A12, 90C27, 90C60

Article Outline

Keywords
See also
References

Keywords

Combinatorial optimization; Cooperative game; Complexity; Algorithms

In combinatorial optimization games, we consider *cooperative games* for which the value of the game is obtained via a *combinatorial optimization* problem. For a cooperative game (a class of *games with side payments*), the set of participating players is denoted by N and a value $v(S)$ is achieved by each subset S of players without any help from other players (in the set $N - S$). Usually, we set $v(\emptyset) = 0$. In general, the representation of the game requires an input size exponential in the number of players. For a combinatorial optimization game, however, the value $v(S)$ is often succinctly defined as a solution to a combinatorial optimization problem for which the combinatorial structure is determined by the subset S of players. The income distributed to individual player i is represented by x_i, $1 \leq i \leq N$, and $x = (x_1, \ldots, x_N)$.

The main issue in cooperative games is how to fairly distribute the income collectively earned by the whole group of players in the game, cooperating with each other. For simplicity, let $x(s) = \sum_{i \in S} x_i$. The income vector x is called an *imputation* if $x(N) = v(N)$, and $\forall i \in N$: $x_i \geq v(\{i\})$ (*individual rationality*). Additional requirements may be added to ensure fairness, stability and rationality. And they lead to different sets of income vectors which are generally referred to as solution concepts. Among many of these solution concepts, the core, which consists of all the imputations satisfying the subgroup rationality condition $\forall S \subseteq N$: $x(s) \geq v(s)$, is naturally defined and has attracted much attention from researchers. It has also led to many fruitful results in combinatorial optimization games. Our focus in this article will be on the core. Readers interested in other solution concepts for cooperative games in general can find them in many *game theory* books and survey papers. For example, [12] gives an interesting discussion for several classical solution concepts in cooperative games and their applications to political economy.

Recently, *computational complexity* has been suggested as another metric for evaluating the rationality of these solution concepts [2]. In this argument, computational complexity is suggested as a measure of bounded rationality [13] for players not to spend *super-*

polynomial time to search for the most suitable solution. For combinatorial optimization games, N. Megiddo [7] suggested that algorithms polynomial in the number of players (as good algorithms following the concept introduced by J. Edmonds [3]) be sought for solutions. As the value of any subset of players is defined as the optimal solution to a combinatorial optimization problem, the input size can often be restricted to be bounded by a polynomial in the number of players. This is usually the case for many practical collective optimization problems. The value of a subgroup of players is the optimal objective function value that this subgroup can achieve under the constraints imposed by resources controlled by players in the subgroup. Very often the collective optimization problem requires an integer solution. It is under this context the game is then referred to as a *combinatorial optimization game*.

An example to formulate a two-sided market (the assignment game) is given in [11]. The underlying structure is a bipartite graph $(V_1, V_2; E)$. One interpretation given by L.S. Shapley and M. Shubik is that V_1 is the set of sellers, and V_2 is the set of buyers. For the simplest case, each seller has an item (say a house) to sell and each buyer wants to purchase an item. The ith seller, $i \in V_1$, values its item at c_i dollars and the jth buyer values the item of the ith seller at h_{ij} dollars. Between this pair, we may define a value $v(\{i, j\}) = h_{ij} - c_i$ if $h_{ij} \geq c_i$ and set (i, j) an edge in E with weight $v(\{i, j\})$. Otherwise, there is no edge between i and j since no deal is possible if the seller values the item more than the buyer does. Considering a game with side-payment, the value $v(s)$ of a subset S of buyers and sellers is defined to be the weight of maximum matching in the bipartite graph $G[S]$ induced by the corresponding set S of vertices (an edge is in $G[S]$ if and only if its two end vertices are both in S). In a linear programming formulation, this is

$$\begin{cases} v(S) = \max & \sum_{(i,j)\in E} v(i,j)x_{ij} \\ \text{s.t.} & \sum_{i\in V_1\cap S} x_{ij} \leq 1 \\ & \sum_{j\in V_2\cap S} x_{ij} \leq 1 \\ & x \geq 0. \end{cases}$$

Shapley and Shubik have shown that the core for this assignment game is precisely the set of solutions for the dual program of the above linear program with $S = V_1 \cup V_2$. Such nice properties are not unusual in combinatorial optimization games. For example, the same fact is established for another game, a cost allocation game on trees, by A. Tamir. Tamir has shown that the core is exactly the set of optimal solutions to the dual program of the linear program formulation for the total cost of the cost allocation problem on trees [14].

The Shapley–Shubik model is a theoretical formulation for a *pure exchange economy*. The linear production game of G. Owen [8] applies their ideas to a production economy. In Owen's model, each player j ($j \in N$) owns a resource vector, b^j. For a subset S of players, their value is the objective function value of the optimal solution for the following linear program:

$$\begin{cases} \max & c^\top y \\ \text{s.t.} & Ay \leq \sum_{j\in S} b^j \\ & y \geq 0. \end{cases} \tag{1}$$

Thus, the value is what the subset of players can achieve in the linear production model with the resources under their control. The core for the lin- ear production game is always nonempty [8] if all the above linear programs have finite optimum. A constructive proof presented by Owen obtains an imputation in the core from any optimal solution of the dual program

$$\begin{cases} \min & \sum_{j\in N} w^\top b^j \\ \text{s.t.} & w^\top A \geq c^\top \end{cases}$$

of the linear program for all the players

$$\begin{cases} \max & c^\top y \\ \text{s.t.} & Ay \leq \sum_{j\in N} b^j. \end{cases}$$

In fact, let w be the optimal solution for the dual program. Set $x_j = w^\top b^j$, $j \in N$. Then $x = (x_1, \ldots, x_N)$ is an imputation in the core. To see so, for each subset $S \subseteq N$, consider $x(s) = \sum_{i\in S} x_i$. By definition of x, we have $x(s) = \sum_{i\in S} w^\top b^i$. Let y_S^* be the optimal solution for the linear program for $v(s)$. Then, $Ay_S^* \leq \sum_{y\in S} b^j$. Therefore, $x(s) \geq w^\top Ay_S^*$. On the other hand, $w^\top A \geq c^\top$. It follows that $x(s) \geq c^\top y_S^*$, which is the same as $x(s) \geq v(s)$ since $v(s) = c^\top y_S^*$.

Notice that this proof depends on the fact that, if for each $S \subseteq N$, the linear program (1) has a finite optimal value. In general, a linear program may be unbounded or infeasible. If for any $S \subseteq N$, the linear program (1) is unbounded, obviously the core does not exist. If it is infeasible, we may define $v(s) = -\infty$. This allows the extension of the above result to the case when the following conditions are satisfied:

1)

$$\begin{cases} \max & c^\top y \\ \text{s.t.} & Ay \le \sum_{j \in N} b^j \\ & y \ge 0 \end{cases}$$

has a finite optimal value.

2) For each $S \subseteq N$, 1) has a finite optimal value or is infeasible.

However, unlike the assignment game, there may in general be imputations in the core which cannot be obtained from the dual program for Owen's linear production game [8]. In general, it is not known how to decide whether an imputation is in the core in polynomial time.

There is a weakness in applying the linear production game model to the studies of coalition optimization problems. That is, in reality, many variables are required to be of integer values. It happens that for the assignment game of Shapley and Shubik, the linear production model of Owen's always results in an integer solution. There are, however, many other situations for which the integer optimal solution cannot be obtained in the framework of the linear program.

A generalized linear production model introduced by D. Granot retains the main linear program structure of Owen's model but allows right-hand sides of the resource constraints not to be linear in the resource vectors b^j of individual players [6]. Thus, $v(S)$ is defined to be $\max\{c^\top y: Ay \le b(S), y \ge 0\}$, where $b(S) = (b_1(s), \ldots, b_m(s))$ is a general function of S. It is shown that, if for each i, $1 \le i \le N$, the game consisting of player set N with value function $b_i(s)$ has a nonempty core, the generalized linear production game has a nonempty core. As the game of Owen's model, an imputation in the core is constructed from the optimal solution for the dual program and vectors in the core associated with (N, b_i) [6]. This would in general need an exponential number of function values $b(s)$ for all the subset S of N. For some collective combinatorial optimization problems, $b(S)$ is given implicitly as a solution to some optimization problem and thus the problem input size is polynomially bounded. The extended power of Granot's model can be applied to prove nonemptiness for the cores of many games beyond those of Owen's linear production game.

In particular, the generalized linear production game model is applied to show the nonemptiness of a certain minimum cost spanning tree game [6]. In this problem, we have a complete graph as the underlying structure. A cost is assigned to each edge. There is a distinguished node 0. Players are vertices $\{1, \ldots, n\}$. The cost $c(S)$ of a subset S of players is defined to be the cost of minimum spanning tree in the graph $G[S \cup 0\}]$ induced by $S \cup \{0\}$. (Notice that the cost game is different from the value game defined as above but can be handled similarly.) Even though an imputation in the core can be found in polynomial time for this game, in [4] it is shown that it is *NP*-hard to decide whether an imputation is not in the core.

Another way to extend Owen's model to include games of combinatorial optimization nature is to explicitly require integer solutions in the definition of the linear production model. That is, one may define game value $v(s)$ for a subset $S \subseteq N$ to be the maximum value of an integer program instead of a linear program. Therefore,

$$v(S) = \max\left\{ c^\top x : \; Ax \le \sum_{j \in S} b^j, \; x \text{ integers} \right\}.$$

For the assignment game of Shapley and Shubik and the cost allocation game on trees of Tamir, the integer program can be solved by its linear program relaxation, since there is always an integer solution for the latter. In the work of Shapley and Shubik, as well as that of Tamir, b^j is a unit vector and $b(N)$ is a vector of all ones. It is this particular structure of linear constraints that makes the core to be identified with the set of optimal solutions for the dual linear program to the linear program of the game value for the set of players [11,14]. It is no wonder this property is further exploited in [5] for a partition game, and in [1] for packing/covering games.

The *packing game*, for example, is defined for a set N of players whose game value is given by the following integer program

$$\begin{cases} \max & c^\top x \\ \text{s.t.} & x^\top A_{M,N} \le 1^N \\ & x \in \{0,1\}^m, \end{cases}$$

where 1^N is a vector of $|N|$ ones, and $A_{M,N}$ is a 0–1 matrix of rows indexed by M and columns indexed by N. For each subset S of players, its value is given by

$$\begin{cases} \max & c^\top x \\ \text{s.t.} & x^\top A_{M,S} \le 1^\top_{|S|}, \quad x^\top A_{M,\overline{S}} \le 0^\top_{n-|S|}, \\ & x \in \{0,1\}^m, \end{cases}$$

where $A_{M,S}$ is the submatrix of A with row set M and column set S, $\overline{S} = N - S$ and $v(\emptyset)$ is defined to be 0.

The covering game and the partition game are defined similarly. It is a necessary and sufficient condition for the core of the packing (and covering, and partitioning) game to be nonempty that the linear relaxation of the corresponding optimization problem always has an integer optimal solution. In additional, the core, if nonempty, is exactly the set of optimal solutions to the dual program of the linear relaxation of the corresponding integer program [1,5].

These results allow for a characterization of combinatorial structures for the corresponding combinatorial optimization game to have a nonempty core. Because of the linear program characterization of the core, questions such as whether the core is empty or not, whether we can find an imputation in the core, and whether an imputation is in the core, can often be determined in polynomial time. Notice that, there are cases that the linear program may be of exponential size in the number of players, it is not immediate that all these questions can be solved in polynomial time. But even for cases when there are an exponential number of constraints, the linear program may be solvable in polynomial time [9].

First established by Shapley and Shubik for the assignment game, the connection of the core for a combinatorial optimization game with dual program of the linear program relaxation has been a successful tool in the characteration of the core, design and analysis of algorithms to find an imputation in the core and to test membership of an imputation in the core. It is expected that this approach would continously lead to fruitful results in cooperative game theory.

See also

- ▶ Combinatorial Matrix Analysis
- ▶ Evolutionary Algorithms in Combinatorial Optimization
- ▶ Fractional Combinatorial Optimization
- ▶ Multi-Objective Combinatorial Optimization
- ▶ Neural Networks for Combinatorial Optimization
- ▶ Replicator Dynamics in Combinatorial Optimization

References

1. Deng X, Ibaraki T, Nagamochi H (1999) Algorithmic aspects of the core of combinatorial optimization games. Math Oper Res 24(3):751–766
2. Deng X, Papadimitriou C (1994) On the complexity of cooperative game solution concepts. Math Oper Res 19(2):257–266
3. Edmonds J (1965) Paths, tree, and flowers. Canad J Math 17:449–469
4. Faigle U, Fekete S, Hochstättler W, Kern W (1997) On the complexity of testing membership in the core of min-cost spanning tree games. Internat J Game Theory 26:361–366
5. Faigle U, Kern W (1995) Partition games and the core of hierarchically convex cost games. Memorandum Fac Toegepaste Wiskunde Univ Twente 1269, no. June
6. Granot D (1986) A generalized linear production model: A unified model. Math Program 34:212–222
7. Megiddo N (1978) Computational complexity and the game theory approach to cost allocation for a tree. Math Oper Res 3:189–196
8. Owen G (1975) On the core of linear production games. Math Program 9:358–370
9. Schrijver A (1986) Theory of linear and integer programming. Wiley, New York
10. Shapley LS (1967) On balanced sets and cores. Naval Res Logist Quart 14:453–460
11. Shapley LS, Shubik M (1972) The assignment game. Internat J Game Theory 1:111–130
12. Shubik M (1981) Game theory models and methods in political economy. In: Arrow KJ, Intriligator MD (eds) Handbook Math. Economics, vol 1. North Holland, Amsterdam, pp 285–330
13. Simon H (1972) Theories of bounded rationality. In: Radner R (ed) Decision and Organization. North Holland, Amsterdam

14. Tamir A (1981) On the core of cost allocation games defined on location problems. Preprints, Second Internat. Conf. Locational Decisions (ISOLDE 81) (Skodsborg, Denmark), pp 387–402

Combinatorial Test Problems and Problem Generators

Don Grundel[1], David Jeffcoat[2]
[1] 671 ARSS/SYEA, Eglin AFB, USA
[2] AFRL/RWGN, Eglin AFB, USA

MSC2000: 90B99, 05A99

Article Outline

Keywords
Introduction
Libraries and Generators
Combinatorial Auctions
Frequency Assignment Problem
Graph Colorability
Linear Ordering Problem
Maximum Clique Problem
Minimum Cut-Set
Minimum Vertex Cover Problem
Multidimensional Assignment Problem (MAP)
Quadratic Assignment Problem (QAP)
Satisfiability
Steiner Problem in Graphs
Traveling Salesman Problem (TSP)
Vehicle Routing Problem
Conclusions
References

Keywords

Test problems; Problem generators

Introduction

Test problems are instances of a mathematical problem used to establish the accuracy or efficiency of a solution method. Test problems provide a common baseline against which to compare a new solution algorithm with an existing procedure. A problem generator is an algorithm to produce a test instance for a specific combinatorial problem. In this section, we provide an overview of test problems and generation methods, as well as sources of test problems for a number of well-known combinatorial problems. The design of test problems is a critical step in the design of combinatorial algorithms, and a sufficient number and variety of test problems must be available to determine the performance of a proposed algorithm across a range of problem types.

There are four basic sources of test problems:

1. Problems taken from real-world applications
2. Libraries of standard test problems
3. Test problems with parameters generated randomly from a specified probability distribution
4. Test problems generated by an algorithm designed to produce problem instances with specific characteristics: e. g., problems with a known solution.

Each of these sources has associated advantages and disadvantages. For example, problems taken from real-world applications have a degree of complexity consistent with at least some problems encountered in practice [28], and provide a context for presenting proposed solutions that promotes understanding and acceptance. However, we typically cannot find a sufficient number of such problems to constitute a satisfactory experiment.

Libraries of standard test problems can provide problems that were used by other researchers, facilitating comparisons with existing solution procedures. However, as with real-world cases, libraries may not provide a sufficient number or variety of problem instances. Procedures that randomly generate test problems can quickly provide an essentially unlimited number of problem instances, but the optimal solution to large randomly generated problems may remain unknown. An additional hazard with randomly generated problems is that such problems are sometimes artificially easy to solve [6,33].

Constructive procedures designed to generate test problems with known solutions can be very useful for evaluating an algorithm's performance, and can also provide a large number of test instances. Problem generation procedures must be carefully examined to determine the difficulty, realism, and other characteristics of the problems generated. An ideal generator would produce problems in polynomial time, with a known solution, of appropriate hardness, and with sufficient diversity [31]. Of course, it can be difficult to simultaneously meet all these requirements. For example, a trivial problem instance might be generated in polynomial time, but provide no real test for a proposed solution

procedure. Problem instances should also be posed using standardized representations [10].

A good set of test problems is only one part of the evaluation of an algorithm. Barr, et al. [2] provide guidelines for designing computational experiments and for reporting results of solution algorithm performance.

The following section provides sources of standard test problems and problem generators for a number of well-known combinatorial optimization problems.

Libraries and Generators

The INFORMS OR/MS Resource Collection [16] and the OR-Library maintained by Beasley [3,4] both provide extensive collections of test data sets for a variety of operations research problems. The Zuse Institute [19] maintains a collection of various problems related to mathematical programming. A handbook of test problems [11] provides a collection of test problems from a wide variety of engineering applications. The Discrete Mathematics and Theoretical Computer Science (DIMACS) Challenges [8] encourage experimental evaluations of algorithms using standard test problems. Over the past decade, challenges have been held for TSP, cliques, coloring, and satisfiability. An overview of sources for specific combinatorial problems is provided below.

Combinatorial Auctions

This problem involves auctions in which bidders place unrestricted bids for bundles of goods. A seller faced with a set of offers for bundles of goods wishes to maximize his revenue. The Combinatorial Auction Test Suite (CATS) provides an algorithm for generating problem instances of differing levels of realism [21].

Frequency Assignment Problem

A library of frequency assignment problems in the context of wireless communication networks is available at [9]. This website includes an extensive bibliography on frequency assignment problems.

Graph Colorability

Sanchis [31] provides an algorithm for generating graph colorability problems with known solutions. This reference also provides a generator for the minimum dominating set problem.

Linear Ordering Problem

Reinelt [29] maintains a library of problems instances for the linear ordering problem, including problem data and optimal solutions. This library also includes software and data for several other discrete optimization problems. Another library is maintained by Martí [22] in which there are large randomly generated problems with best known solutions.

Maximum Clique Problem

Hasselberg, et al. [13] consider a number of interesting problems, including the maximum clique problem. They introduce different test problem generators motivated by a variety of practical applications, including coding theory and fault diagnosis.

Minimum Cut-Set

Krishnamurthy [20] provides a problem generator for partitioning heuristics, including the minimum cut-set problem. Generated instances of this problem are useful in circuit design applications.

Minimum Vertex Cover Problem

Sanchis and Jagota [32] discuss a test problem generator that builds instances of the minimum vertex cover problem. The generator provides construction parameters to control problem difficulty. Sanchis [31] provides an algorithm to generate minimum vertex cover problems that are diverse, hard and of known solution.

Multidimensional Assignment Problem (MAP)

The axial MAP is a generalization of the linear assignment problem. Grundel and Pardalos [12] provide a MAP generator that produces difficult problems with known unique optimal solutions.

Quadratic Assignment Problem (QAP)

Pardalos [25] provides a method for constructing test problems for constrained bivalent quadratic programming. This reference includes a standardized random

test problem generator for the unconstrained quadratic zero-one programming problem. Yong and Pardalos [35] provide methods for generating test problems with known optimal solutions for more general cases of the QAP. Calamai, et al. [7] describe a technique for generating convex, strictly concave and indefinite QAP instances. Palubeckis [24] provides a method for generating hard rectilinear instances of the QAP with known optimal solutions. Burkard, et al. [5] give additional useful information concerning this difficult problem.

Satisfiability

Achlioptas, et al. [1] propose a generator for satisfiability problems that controls the hardness of the instances. A web page maintained by Uchida, Motoki, and Watanabe [34] is dedicated to two methods of generating instances of 3-satisfiability. A library of satisfiability problem instances and solvers is available on a Darmstadt University website [15].

Steiner Problem in Graphs

Khoury, et al. [17] use a binary-programming formulation to generate test problems with known solutions by applying the Karush-Kuhn-Tucker optimality conditions to the corresponding quadratically-constrained optimization problem. Koch, et al. [18] provide a library of Steiner tree problems with information about the origin, solvability, and other characteristics of this problem.

Traveling Salesman Problem (TSP)

Moscato [23] maintains a web site with resources for the generation of TSP instances with known optimal solutions. An approach for generating discrete instances of the symmetric TSP with known optima is provided by Pilcher and Rardin [27]. A number of libraries (e. g [4,30]) provide test cases for the TSP.

Vehicle Routing Problem

Homberger [14] provides a large set of Vehicle Routing Problems with Time Windows, including instances with up to one thousand customers.

Conclusions

Researchers need a large set of well-designed test problems to effectively compare the performance of existing solution algorithms or to evaluate a new algorithm. Although practitioners may prefer real-world problems for such tests, a sufficient number of test problems may not be available to conduct a thorough experiment. Randomly generated test problems can provide an essentially limitless supply of instances. However, random test instances may be artificially easy to solve, or, at the other extreme, may have no known solution, making it difficult to judge the performance of a new solution algorithm. Test problem generators, if properly designed, can provide a large supply of hard problem instances with known optimal solutions. Many such generators are readily available to researchers. Libraries of test problem are also available, providing a variety of problem types and sizes.

References

1. Achlioptas D, Gomes C, Kautz H, Selman B (2000) Generating satisfiable problem instances. In: Proceedings of the 17th National Conference on Artifical Intelligence, Austin, USA, 31 July–2 Aug 2000. AAAI Press, Menlo Park, USA, pp 256–261
2. Barr RS, Golden BL, Kelly JP, Resende MGC, Stewart WR (1995) Designing and Reporting on Computational Experiments with Heuristic Methods. Heuristics J 1: 9–32
3. Beasley JE (1990) OR-Library: distributing test problems by electronic mail. J Oper Res Soc 41(11):1069–1072
4. Beasley JE, OR-Library. http://people.brunel.ac.uk/~mastjjb/jeb/info.html. Accessed 7 Dec 2007
5. Burkard R, Çela E, Karisch S, Rendlqaplib F A Quadratic Assignment Problem Library. http://www.opt.math.tu-graz.ac.at/qaplib/. Accessed 7 Dec 2007
6. Burkard R, Fincke U (1985) Probabilistic asymptotic properties of some combinatorial optimization problems. Discret Appl Math 12:21–29
7. Calamai PH, Vicente LN, Júdice JJ (1993) A new technique for generating quadratic programming test problems. Math Program 61:215–231
8. Center for Discrete Mathematics & Theoretical Computer Science, Rutgers University, NJ. http://dimacs.rutgers.edu/. Accessed 7 Dec 2007
9. Eisenblätter A, Koster A. FAP web, A website about Frequency Assignment Problems. http://fap.zib.de/index.php. Accessed 7 Dec 2007
10. Fourer R, Lopes L, Martin K (2004) LPFML: A W3C XML Schema for Linear Programming. http://www.

optimization-online.org/DB_HTML/2004/02/817.html. Accessed 7 Dec 2007
11. Floudas CA, Pardalos PM, Adjiman CS, Esposito WR, Gümüs ZH, Harding ST, Klepeis JL, Meyer CA, Schweiger CA (1999) Handbook of Test Problems in Local and Global Optimization. Kluwer, Dordrecht, Netherlands
12. Grundel D, Pardalos P (2005) Test Problem Generator for the Multidimensional Assignment Problem. Comput Optim Appl 31(3)133–146
13. Hasselberg J, Pardalos PM, Vairaktarakis G (1993) Test case generators and computational results for the maximum clique problem. J Glob Optim 3:463–482
14. Homberger J Extended SOLOMON's VRPTW instances. http://www.fernuni-hagen.de/WINF/touren/menuefrm/probinst.htm. Accessed 7 Dec 2007
15. Hoos H, Stützle T (2000) SATLIB: An Online Resource for Research on SAT. In: Gent IP, v Maaren H, Walsh T (eds) SAT 2000. IOS Press, pp 283–292. http://www.satlib.org/. Accessed 7 Dec 2007
16. INFORMS® Online, OR/Resource MS Collection: Resources: Problem Instances. http://www.informs.org/Resources/Resources/Problem_Instances/. Accessed 7 Dec 2007
17. Khoury BN, Paradalos PM, Du D–Z (1993) A Test Problem Generator for the Steiner Problem in Graphs. Trans ACM Math Softw 19(4):509–522
18. Koch T, Martin A, Voß S SteinLib: An Updated Library on Steiner Tree Problems in Graphs. http://elib.zib.de/steinlib/steinlib.php. Accessed 7 Dec 2007
19. Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), a non-university research institute of the state Berlin (1994–2002) http://elib.zib.de/pub/Packages/mp-testdata/index.html. Accessed 7 Dec 2007
20. Krishnamurthy B (1987) Constructing test cases for partitioning heuristics. Trans IEEE Comput vol C-36, Num. 9, September 1987, pp 1112–1114
21. Leyton-Brown K, Pearson M, Shoham Y (2000) Towards a Universal Test Suite for Combinatorial Auction Algorithms. In: Proceedings of Conference ACM on Electronic Commerce, Minneapolis, USA, 17–20 Oct. Sponsored by Association for Computing Machinery (ACM) Special Interest Group on E-commerce. Test suite available via http://www.cs.ubc.ca/~kevinlb/CATS/. Accessed 7 Dec 2007
22. Martí R. Linear Ordering: Publications and Working Papers. http://www.uv.es/~rmarti/paper/lop.html. Accessed 7 Dec 2007
23. Moscato P Fractal Instances of the Traveling Salesman Problem. Densis, FEEC, Universidade UNICAMP Estadual de Campinas. http://www.ing.unlp.edu.ar/cetad/mos/FRACTAL_TSP_home.html. Accessed 7 Dec 2007
24. Palubeckis G (1999) Generating hard test instances with known optimal solution for the rectilinear quadratic assignment problem. J Glob Optim 15:127–156
25. Pardalos P (1991) Construction of test problems in quadratic bivalent programming. Trans ACM Math Softw 17(1):74–87, March 1991
26. Pardalos P, Pitsoulis L (eds) (2000) Nonlinear Assignment Problems. Algorithms and Applications. Kluwer, Dordrecht, pp 1–12
27. Pilcher M, Rardin R (1992) Partial polyhedral description and generation of discrete optimization problems with known optima. Nav Res Logist 39:839–858
28. Reilly CH (1999) Input models for synthetic optimization problems. In: Farrington PA, Nembhard HB, Sturrock DT, Evans GW (eds) Proceedings of the 1999 Winter Simulation Conference, Phoenix, USA
29. Reinelt G Linear Ordering Library (LOLIB). http://www.iwr.uni-heidelberg.de/groups/comopt/software/LOLIB/. Accessed 7 Dec 2007
30. Reinelt G. TSLIB. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/. Accessed 7 Dec 2007
31. Sanchis L (1995) Generating hard and diverse test sets for NP-hard graph problems. Discret Appl Math 58:35–66
32. Sanchis L, Jagota A (1996) Some experimental and theoretical results on test case generators for the maximum clique problem. INFORMS J Comput 8(2):87–102
33. Selman B, Mitchell D, Levesque H (1996) Generating Hard Satisfiability Problems. Artif Intell 81:17–29
34. Uchida T, Motoki M, Watanabe O Instance SAT Generation Page, Watanabe research group of Dept. of Math. and Computing Sciences, Tokyo Inst. of Technology. http://www.is.titech.ac.jp/~watanabe/gensat/index.html. Accessed 7 Dec 2007
35. Yong L, Pardalos PM (1992) Generating quadratic assignment test problems with known optimal permutations. Comput Optim Appl 1(2):163–184

Communication Network Assignment Problem

CAP

Rainer E. Burkard
Technical University Graz, Graz, Austria

MSC2000: 90B80, 90C05, 90C27, 68Q25

Article Outline

Keywords
See also
References

Keywords

Optimization; Assignment problem; Communication network; Computational complexity; Exact algorithms; Heuristic approaches; Asymptotic behavior

In the *communication network assignment problem* (CAP) a system of communication centers $C_1, \ldots, C_n$ is given. The centers have to be embedded into a given (undirected) network $N = (V, E)$ with vertex set V, $|V| = n$, and edge set E. The centers exchange messages at given rates per time unit through a selected routing pattern. Let t_{ij} be the amount of messages sent from center C_i to center C_j per time unit. If there is no direct connection between C_i and C_j the messages sent from C_i to C_j pass through several intermediate centers. The messages exchanged between C_i and C_j may be sent along a single path or they may be split into several parts, each part being sent along its own path. For any fixed embedding $\mathcal{E}$ of the centers into the network and for any-fixed routing pattern ρ of the messages, let $IMT_{\mathcal{E},\rho}(C_i)$ denote the overall amount of traffic going through the center C_i as intermediate center. The goal is to find an embedding $\mathcal{E}$ of the centers into the network and a routing pattern ρ which minimizes the maximum intermediate traffic over all centers:

$$\min_{\mathcal{E},\rho} \max \left\{ IMT_{\mathcal{E},\rho}(C_i) : \ 1 \le i \le n \right\} \tag{1}$$

A typical application of the problem arises in the case of locating stations (terminals, computers) in a *local-area computer network* (LAN) as described by T.B. Boffey and J. Karkazis [1]. Usually, a given segment of the LAN serves different pairs of communicating stations. In order to prevent interference and garbled messages, only one message at a time can be sent through a given segment of the LAN. On the other hand one has to restrict the offered traffic through the same segment so as to maintain a reasonable throughput in the network. To this end it is reasonable to locate *bridges* at the endpoints of each segment. All bridges will work as intermediate centers and all stations will work as bridges. The result is that each pair of stations (or bridges) communicates through its own segment. It is reasonable to require an embedding of stations and additional bridges into the nodes of the LAN such that the intermediate traffic going through the busier station (or bridge) is minimized. Boffey and Karkazis [1] proposed and discussed also a continuous version of the problem.

A similar problem, the so-called *elevator problem* leads also to the optimization problem (1) as described by Karkazis [5]. The elevator problem arises when a single elevator has to be replaced by two elevators, each covering contiguous subsets of floors. It might be reasonable to place the connecting landing so as to minimize the traffic intensity on the busier elevator. More specifically assume that the first elevator serves floors $\{1, 2, \ldots, i\}$ and the second elevator serves floors $\{i, i+1, \ldots, n\}$, and let t_{ij} represent the traffic intensity from floor i to floor j. Then the traffic load of the first elevator is given as $T_i^{(1)} = \sum_{k=1}^{i} \sum_{l=1}^{n} (t_{kl} + t_{lk})$ and the traffic load of the second elevator is given as $T_i^{(2)} = \sum_{k=i}^{n} \sum_{l=1}^{n} (t_{kl} + t_{lk})$. Then we want to choose i so as to minimize $\max\{T_i^{(1)}, T_i^{(2)}\}$. Obviously this problem setting can be generalized for more than two elevators.

Essentially there are two distinct models of routing patterns in (1): the *single path* model and the *fractional* model. In the single path model, for every pair of communication centers C_i and C_j, a single route in the network is selected and all t_{ij} messages are sent along this fixed route. In the fractional model, the amount t_{ij} is split into a number of positive parts and every part is sent along its own path. Most of the results available in the literature concern the CAP on *trees*. In this case, for each pair of vertices in the network there is only one path to join them and hence, both models coincide.

R.E. Burkard, E. Çela and G.J. Woeginger have proved in [3] that in general the CAP is *NP*-hard. More specifically has been shown that the CAP is *NP*-hard for networks that are i) paths; ii) stars of branch length three; iii) cycles *NP*-hardness in both models); or iv) doublestars (*NP*-hardness in the single path model). Moreover, it has been proved in [3] that the CAP is polynomially solvable in the case of stars of branch length two and in the case of doublestars in the fractional model. In the case of a star of branch length two the CAP can be formulated as a maximum weight perfect matching problem (MWPMP) if the communication center to be assigned to the central node of the star is kept fixed. Since there are only n possibilities for the selection of the center to be placed at the central node, one just has to solve n MWPMPs. In the fractional model, finding an embedding of the communication centers into the nodes of the network and a routing pattern which minimize the intermediate traffic can be done by solving a specified number of linear programs with $O(P)$ variables and $O(n^2)$ constraints each, where P is the number of pairwise disjoint paths in N. In the case of doublestars $P = O(n^2)$ and there are $O(n^2)$ pro-

grams to be solved (see [3]). This implies that in this case the CAP is polynomially solvable in the fractional model.

Some *exact algorithms* and *heuristic approaches* to solve the CAP on trees have been proposed in [3,5]. Karkazis has proposed a *branch and bound* algorithm in the case where N is a path [5], and Burkard, Çela and Woeginger [3] have proposed a branch and bound approach in the case that N is a tree. The algorithms have been tested on randomly generated trees and communication rates t_{ij}. The tests show that only small instances of the CAP of size up to 12 can be solved in reasonable time. For large instances the number of the branched nodes in the branch and bound tree explodes. In order to approximately solve larger instances of the CAP on trees Burkard, Çela and T. Dudàs proposed in [2] *simulated annealing* and *tabu search* approaches. The proposed heuristics are tested on randomlygenerated instances of size up to 32. The comparison of the heuristic solutions with the optimal solution produced by the branch and bound algorithm for instances of small size shows that the performance of these heuristics is quite satisfactory.

Finally, in [2] the *asymptotic behavior* of the CAP on trees has been investigated.Under natural probabilistic assumptions on the problem data the CAP on trees shows a very interesting behavior: The ratio between the maximum and the minimum values of the objective function, i. e., the ratio between the maximum and the minimum values of the intermediate traffic through the busiest center, approaches 1 with probability tending to 1 as the size of the problem tends to infinity. The proof of this fact is based on the strong relationship between the CAP-T and a special version of the *quadratic assignment problem*. It is shown that the latter fulfills the condition of a theorem of Burkard and U. Fincke [4] on the asymptotic behavior of combinatorial optimization problems. From a practical point of view the asymptotic behavior described above implies that the CAP on trees becomes trivial as its size tends the infinity: every feasible solution provides a good approximation of an optimal solution.

See also

- ► Assignment and Matching
- ► Assignment Methods in Clustering
- ► Auction Algorithms
- ► Bi-Objective Assignment Problem
- ► Directed Tree Networks
- ► Dynamic Traffic Networks
- ► Equilibrium Networks
- ► Evacuation Networks
- ► Frequency Assignment Problem
- ► Generalized Networks
- ► Maximum Flow Problem
- ► Maximum Partition Matching
- ► Minimum Cost Flow Problem
- ► Network Design Problems
- ► Network Location: Covering Problems
- ► Nonconvex Network Flow Problems
- ► Piecewise Linear Network Flow Problems
- ► Quadratic Assignment Problem
- ► Shortest Path Tree Algorithms
- ► Steiner Tree Problems
- ► Stochastic Network Problems: Massively Parallel Solution
- ► Survivable Networks
- ► Traffic Network Equilibrium

References

1. Boffey TB, Karkazis J (1989) Location of transfer centers on segments of a communication network with proportional traffic. J Oper Res Soc 40:729–734
2. Burkard RE, Çela E, Dudàs T (1996) A communication assignment problem on trees: heuristics and asymptotic behavior. Network optimization. In: Lecture Notes Economics and Math Systems, vol 450. Springer, Berlin, pp 127–156
3. Burkard RE, Çela E, Woeginger GJ (1995) A minimax assignment problem in treelike communication networks. Europ J Oper Res 87:670–684
4. Burkard RE, Fincke U (1985) Probabilistic asymptotic properties of some combinatorial optimization problems. Discrete Appl Math 12:21–29
5. Karkazis J (1993) A minimax assignment problem on a linear communication network. Belgian J Oper Res Statist Comput Sci 33:5–17

Competitive Facility Location

Tammy Drezner
California State University,
Fullerton, USA

MSC2000: 90B60, 90B80, 90B85

Article Outline

Keywords and Phrases
Introduction
The Proximity Model
The Location-Allocation Model
The Deterministic Utility Model
The Random Utility Model
Gravity Based Models
Anticipating Future Competition
Conclusions
References

Keywords and Phrases

Location; Competitive

Introduction

Facility location models deal, for the most part, with the location of plants, warehouses, distribution centers and other industrial facilities. These location models do not account for competition or for differences among facilities and therefore allocate customers to facilities by proximity. In reality, retail facilities operate in a competitive environment with an objective of profit and market share maximization. These facilities are also different from each other in their overall attractiveness to consumers. One branch of location analysis focuses on the location of retail and other commercial facilities which operate in a competitive environment, namely competitive facility location. The basic problem is the optimal location of one or more new facilities in a market area where competition already exists or will exist in the future. Assuming that profit increases when market share increases, maximizing profit is equivalent to maximizing market share. It follows, then, that the location objective is to locate the retail outlet at the location that maximizes its market share.

A unique feature of competitive facility location models is facility attractiveness (its appeal to consumers). Facilities differ in the total "bundle of benefits" they offer customers. They vary in one or more of the attributes which make up their total attractiveness to customers. Furthermore, varying importance assigned to each of these attributes by different customers will result in a selective set of consumers patronizing each. Facility attractiveness level, therefore, needs to be incorporated in the location model. Facility attractiveness needs to first be assessed using one of a variety of methods. Once attractiveness is assumed known, market share captured can be calculated. Facility attractiveness is estimated using a utility function (a composite index of attractiveness) or some other measure (floor area) serving as a surrogate for a latent attractiveness. Utility models are predicated on consumer spatial choice models as well as on the premise that facilities of the same type are not necessarily comparable.

Also unique to competitive facility location is the modeling of demand in terms of buying power. Income levels and discretionary spending become a measure of demand. For a review of competitive models see [4,15].

The underlying theme running through all competitive models is the existence of an interrelationship between four variables: buying power(demand), distance, facility attractiveness, and market share, with the first three variables being independent variables and the last the dependent variable. Buying power, or effective buying income, is known (for example, Sales and Marketing Management magazine). Distance from demand points to facilities can be measured. The most difficult link in the interrelationship between the four variables is the determination of facility attractiveness. For a discussion of the determination of facility attractiveness see [6,9]. As is mentioned above and discussed below, it is estimated using a utility function. Once buying power, distance, and attractiveness are known, market share can be calculated.

The Proximity Model

The first modern paper on competitive facility location is generally agreed to be Hotelling's paper on duopoly in a linear market [21]. Hotelling considered the location of two competing facilities on a segment (for example, two ice-cream vendors along a beach strip). The distribution of buying power along the segment is assumed uniform and customers patronize the closest facility. When one facility is located and there is no competition, all customers patronize the existing facility. However, when a competing facility is introduced and is located at a different point on the segment, the customers on one side of the midpoint between the two facilities patronize one facility and the customers on the other side of the midpoint patronize the second facil-

ity. If one facility is held fixed in place, the best location for the second is either immediately left or right of the fixed one, depending on which segment – left or right of the existing facility – is longer. In models based on Hotelling's formulation it is assumed that customers patronize the closest facility.

The Location-Allocation Model

An extension to Hotelling's approach is the location-allocation model for the selection of sites for facilities that serve a spatially dispersed population. Both the facilities' locations and the allocation of customers to them are determined simultaneously. The allocation of customers to facilities is made using Hotelling's proximity assumption – each facility attracts the consumers closest to it. The market share attracted by each facility is calculated and the best locations for the new facilities are then found. Multifacility location-allocation models analyze the system-wide interactions among all facilities. Revelle [28] introduced location-allocation models to competitive location. Goodchild [19] suggested the location-allocation market share model (MSM). A retail firm is planning to open a chain of outlets in a market in which a competing chain already exists. The entering firm's goal is to maximize the total market share captured by the entire chain. Most location-allocation solution methods rely on heuristic approaches that do not guarantee an optimal solution, rather they provide good solutions for implementation. The best locations are selected from a user-provided, prespecified set of potential sites. Typically, these problems are formulated on a network and the location solution is on a node. A book edited by Ghosh and Rushton [18] provides a collection of papers on the subject. A comprehensive review of location-allocation models can be found in [17].

The assumption that customers patronize the facility closest to them implies that the competing facilities are equally attractive. For equally attractive facilities, the plane is partitioned by a Voronoi diagram [26,27]. It is implicitly assumed that all customers located at a demand point patronize the same facility. This, in turn, implies an "all or nothing" property. The combined buying power at a demand point is assigned entirely to one facility and none is assigned to other facilities, unless two or more facilities are equidistant. A solution procedure for solving the multiple competitive facility location in the plane is proposed in [29].

The Deterministic Utility Model

When the facilities are not equally attractive, the proximity premise for allocating consumers to facilites is no longer valid. To account for variations in facility attractiveness, a deterministic utility model for competitive facility location is introduced by T. Drezner [2]. Hotelling's approach is extended by relaxing the proximity assumption. Consumers are known to make their choice of a facility based on factors other than distance alone. Therefore, it is assumed that customers base their choice of a facility on facility attractiveness which is represented by a utility function. This utility function is a composite index of facility attributes and the distance to the facility, representing the expected satisfaction from that facility (either an additive or a multiplicative utility function). It is generally agreed that customers, through a decision-making process, choose the facility with the highest utility, the facility which is expected to maximize their satisfaction. This choice is determined by some formula according to which customers evaluate alternative facilities' attributes weighted by their personal salience to arrive at an overall facility attractiveness.

A trade-off between distance and attractiveness takes place. Based on this premise the degree of expected satisfaction with each alternative as a function of the relevant characteristics of that facility is measured. It is suggested that a customer will patronize a better and farther facility as long as the extra distance to it does not exceed its attractiveness advantage. For example, paramedics transporting a motor vehicle accident victim will by-pass a nearby hospital in favor of a farther, better equipped trauma center as long as the difference in quality of care exceeds the adverse effect to the patient caused by the extra distance and time delay. A break-even distance is defined. At the break-even distance the attractiveness of two competing facilities is equal. This break-even distance, therefore, is the maximum distance that a customer will be willing to travel to a farther facility (new or existing) based on his perception of its attractiveness and advantage relative to other facilities. All customers at a demand point will patronize the new facility if it is located within the break-even

distance. While customers are no longer assumed to patronize the closest facility, customers at a certain demand point are assumed to apply the same utility function, therefore, they all patronize the same facility. The "all or nothing" property is maintained in this extension.

Based on aggregated utility values for existing facilities and a utility function for a new facility, the best location is found for the new one. The optimal location for the new facility is sensitive to its attractiveness. Different attractiveness levels may yield different optimal locations.

The Random Utility Model

To address the "all or nothing" assumption of the deterministic utility model and to account for variations in individual utility functions, a random utility model is introduced by Drezner and Drezner [7]. The deterministic utility model is extended by assuming that each customer draws his utility from a random distribution of utility functions. The probability that a customer will prefer a certain facility over all other facilities is calculated by applying the multivariate normal distribution. Once the probabilities are calculated, the market share captured by a particular facility (new or existing) can be calculated as a weighted sum of the buying power at all demand points. This formulation eliminates the "all or nothing" property since a probability that a customer will patronize a particular facility can be established and is no longer either 0% or 100%. To circumvent the mathematically complicated formulation of the random utility model, Drezner et al. [14] suggested using the simpler logit model. The probability that a customer will patronize a facility as a function of the distance to that facility, can be approximated by a logit function of the distance.

Gravity Based Models

An alternative approach to the location of competing facilities, based on the gravity model, was introduced by Huff [22,23] and is extensively used by marketers. According to the gravity model two cities attract retail trade from an intermediate town in direct proportion to the populations of the two cities and in inverse proportion to the square of the distances from them to the intermediate town. Huff proposed that the probability that a consumer patronizes a retail facility is proportional to its size (floor area) and inversely proportional to a power of the distance to it. Facility size, or square footage, is a surrogate for facility attractiveness. Huff depicted equi-probability lines. A customer located on such a line between two facilities patronizes the two facilities with equal probability. These equi-probability lines divide the region into catchment areas, each dominated by a facility, in a manner similar to the Voronoi diagram [26]. These lines do not define an "all or nothing" assignment of customers to facilities, rather, at any demand point, the proportion of consumers attracted to each facility is a function of its square footage (attractiveness) and distance. The model finds the market share captured at each potential site, and thus the best location for new facilities whose individual measures of attractiveness are known.

Suppose there are k existing facilities and n demand points. The attractiveness of facility j is A_j for $j = 1, \ldots, k$, and the distance between demand point i and facility j is d_{ij}. The buying power at demand point i is b_i. Therefore, the proportion of the buying power (market share) M_j attracted by facility j is:

$$M_j = \sum_{i=1}^{n} b_i \frac{\frac{A_j}{d_{ij}^{\lambda}}}{\sum_{m=1}^{k} \frac{A_m}{d_{im}^{\lambda}}} \tag{1}$$

where λ is the power to which distances are raised.

In the original Huff formulation, facility floor area serves as a surrogate for attractiveness. A major improvement on Huff's approach was suggested by Nakanishi and Cooper [25] who introduced the multiplicative competitive interaction (MCI) model. The MCI coefficient replaces the floor area with a product of factors, each a component of attractiveness. Each factor in the product is raised to a power. Thus, the attractiveness of a facility is a composite of a set of attributes rather than the floor area alone. Nakanishi and Cooper's idea was elaborated on and applied by Jain and Mahajan [24] to food retailing using specific attractiveness attributes. Gravity based models suggest the evaluation in terms of market share of a user provided discrete set of potential sites for the location of a new facility.

Huff's and Nakanishi and Cooper's models were extended to the location of multiple facilities by [1,16].

Achabal et al. [1] extended the MCI model to the location of multiple facilities which belong to the same chain. The problem was modeled as a nonlinear integer programming problem and a random search procedure combined with an interchange heuristic was employed to identify optimal and near-optimal sets of locations. Ghosh and Craig [16] proposed a franchise distribution model. An expanding franchise seeks to maximize sales while minimizing cannibalization of franchise outlets. This model was also formulated as a nonlinear integer programming problem but included additional factors such as advertising. These two models select the best locations from a user-provided set of alternative sites as well.

Other papers [20,30] suggest variations on Huff's formulation by replacing the distance raised to a power with an exponent of the distance. This formulation accelerates the distance decay.

Finding the best location for a new facility (or multiple facilities) in a continuous space using the gravity model objective is discussed in T. Drezner [3] and Drezner and Drezner [10] for the single facility case, and in T. Drezner [5] and T. Drezner et al. [12] for the location of multiple facilities.

Finding the best location for a competing facility that minimizes the probability of not meeting a given minimum threshold of market share is discussed in [13].

All models discussed above assume that demand is distributed among the competing facilities. For non-essential services, some of the demand may not be satisfied. A model which assumes that some of the demand is lost is proposed in [11].

Anticipating Future Competition

The competitive facility location models discussed above are myopic and short-term oriented in that they attempt to find the optimal location for a new facility (facilities) by maximizing current market share against existing competition. A different approach to competitive location focuses on anticipating and preempting future competition. It is assumed that a new competing facility will enter the market at some point in the future. The competitor will establish his facility at the location which maximizes *his* market share. Therefore, one's present location decision will affect the competitor's location decision. Conversely, assuming a future competitive entry has implications for one's present location decision. The objective is to find the location that maximizes the market share captured by one's own facility *following* the competitor's entry. This problem is known in the economic literature as the Stackelberg equilibrium problem or the leader-follower problem and as the Simpson's problem in voting theory. See [8] for a review of the topic.

Conclusions

There are two main applications for competitive facility location models. The first application is the location analysis of a new facility. The best location for the new facility, based on market share maximization at that location, is found. The second application is an analysis of the impact of changes in quality in existing facilities (either own's, competitor's, or both) on the market share captured by one's facility and on its optimal location. In addition, a decision maker will be able to perform a "what-if analysis" and anticipate the impact on his facility of either competitor's improvements or of the introduction of a new facility. In this case one needs to know the overall attractiveness of the proposed new facility or the difference in overall attractiveness pre-post improvements in an existing one. Using the models, a decision maker can assess:

1. the impact on location of changes in attractiveness for his new facility;
2. the impact on market share of change in location for his new facility;
3. the impact on market share of changes in attractiveness at his existing facility(ies);
4. the impact on his facility of changes in other facilities or the introduction of a new facility.

These models afford the anticipation and analysis of the impact of likely future scenarios. In a highly competitive market such as exists domestically, and in the face of increasing global competition, the ability to optimize location in terms of market share provides a strategic advantage for decision makers.

References

1. Achabal D, Gorr WL, Mahajan V (1982) MULTILOC: A multiple store location decision model. J Retail 58:5–25

2. Drezner T (1994) Locating a single new facility among existing unequally attractive facilities. J Reg Sci 34:237–252
3. Drezner T (1994) Optimal continuous location of a retail facility, facility attractiveness, and market share: an interactive model. J Retail 70:49–64
4. Drezner T (1995) Facility Location: A survey of applications and methods, ch. Competitive facility location in the plane. Springer, New York
5. Drezner T (1998) Location of multiple retail facilities with a limited budget. J Retail Consum Serv 5:173–184
6. Drezner T (2006) Derived Attractiveness of Shopping Malls. IMA J Manag Math 4:349–358
7. Drezner T, Drezner Z (1996) Competitive facilities: market share and location with random utility. J Reg Sci 36:1–15
8. Drezner T, Drezner Z (1998) Location of retail facilities in anticipation of future competition. Locat Sci 6:155–173
9. Drezner T, Drezner Z (2002) Validating the Gravity-Based Competitive Location Model Using Inferred Attractiveness. Annals Oper Res 11:227–237
10. Drezner T, Drezner Z (2004) Finding the Optimal Solution to the Huff Based Competitive Location Model. Comput Manag Sci 1:193–208
11. Drezner T, Drezner Z (2008) Lost Demand in a Competitive Environment. J Oper Res Soc, in press
12. Drezner T, Drezner Z, Salhi S (2002) Solving the Multiple Competitive Facilities Location Problem. Eur J Oper Res 142:138–151
13. Drezner T, Drezner Z, Shiode SA (2002) Threshold Satisfying Comprtitive Location Model. J Reg Sci 42:287–299
14. Drezner T, Drezner Z, Wesolowsky GO (1998) On the logit approach to competitive facility location. J Reg Sci 38:313–327
15. Drezner Z, Drezner T (1998) Modern Methods for Business Research, ch. Applied location models. Lawrence Erlbaum Associates, Mahwah
16. Ghosh A, Craig CS (1991) FRANSYS: A franchise location model. J Retail 67:212–234
17. Ghosh A, Harche F (1993) Location-allocation models in the private sector: progress, problems, and prospects. Locat Sci 1:81–106
18. Ghosh A, Rushton G (1987) Spatial Analysis and Location Allocation Models. Van Nostrand Reinhold, New York
19. Goodchild MF (1984) ILACS: A location allocation model for retail site selection. J Retail 60:84–100
20. Hodgson JM (1981) A location-allocation model maximizing consumers' welfare. Reg Studies 15:493–506
21. Hotelling H (1929) Stability in competition. Eco J 39:41–57
22. Huff DL (1964) Defining and estimating a trade area. J Mark 28:34–38
23. Huff DL (1966) A programmed solution for approximating an optimum retail location. Land Eco 42:293–303
24. Jain AK, Mahajan V (1979) Research in Marketing, ch. Evaluating the competitive environment in retailing using multiplicative competitive interactive models. JAI Press, Greenwich
25. Nakanishi M, Cooper LG (1974) Parameter estimate for multiplicative interactive choice model: least squares approach. J Mark Res 11:303–311
26. Okabe A, Boots B, Sugihara K (1992) Spatial tessellations: concepts and applications of Voronoi diagrams. Wiley, Chichester
27. Okabe A, Suzuki A (1987) Stability of spatial competition for a large number of firms on a bounded two-dimensional space. Env Plan A 16:107–114
28. ReVelle C (1986) The maximum capture or sphere of influence problem: Hotelling revisited on a network. J Reg Sci 26:343–357
29. Suzuki A, Drezner Z, Drezner T (2007) Locating Multiple Facilities in a Planar Competitive Environment. J Oper Res Soc Japan 50:1001–1014
30. Wilson AG (1976) Theory and Practice in Regional Science, ch. Retailers' profits and consumers' welfare in a spatial interaction shopping model. Pion, London

Competitive Ratio for Portfolio Management

CRPM

Xiaotie Deng
City University Hong Kong, Kowloon, China

MSC2000: 91B28, 68Q25

Article Outline

Keywords
See also
References

Keywords

Portfolio management; Unknown information; Algorithms; Competitive ratio

Portfolio management is a typical decision making problem under incomplete, sometimes unknown, information. Very often, a *probability distribution* is assumed for stock/bond prices in the future. In the classical work of H.M. Markowitz [9], the investors are assumed to base their decisions for portfolio management on their preference of *return and risk*. In this model, the return is specified as the expected value of the portfolio, and the risk its variance. One of the great achievements

of this work is its predictive power of *diversified investment decisions*.

The assumption that future events would follow some *probability distribution* is also widely accepted for many other problems for which information on future events is uncertain. Very often, *uncertainty* is used as a synonym for probability distribution. However, a fundamental problem still remains: What decisions should we make in presence of future unknown events for which we are simply ignorant of any information? In such situations, the quality of a solution made under ignorance can only be known after future events reveal themselves. Therefore, the quality of a decision should be evaluated in comparison with the optimal available strategy we could have chosen knowing the outcome. Along this approach, the concept of *competitive ratio*, which optimizes the ratio of the outcome of a strategy under *incomplete information* and the optimal outcome under complete information, has been widely applied to solve computational problems under incomplete information, [4,7,8,11]. In particular, R. El-Yaniv, et al., applied competitive analysis to the problem of foreign currency purchase [5]. X. Deng has suggested to apply the competitive analysis to portfolio management problems [3].

Consider a maximization problem. Let $X = (x_1, \ldots, x_n)$ be the variables we have no complete information until in the future. Let $Y = (y_1, \ldots, y_m)$ be the decision variablesfor which we have to choose their values now. Let A = $(a_1, \ldots, a_k)$ be the variables we know of their values at the time we make decisions on Y. A decision rule is a function S: $A- > Y$. Let $v(A, Y, X)$ be the value of the *objective function*. Denote by $v_S(A, X) = v(A, S(A), X)$ be the value of the objective function achieved by the decision rule S if the future outcome is X. Let OPT $(A, X) = \max_{\text{all } Y} v(A, Y, X)$. The competitive ratio of decision rule S is

$$\min_{\text{all } X} \frac{v_S(A, X)}{\text{OPT}(A, X)}.$$

We are interested in a decision rule which achieves the optimal competitive ratio:

$$\max_{\text{all } S} \min_{\text{all } X} \frac{v_S(A, X)}{\text{OPT}(A, X)}.$$

Consider the portfolio management problem of choosing from a set of n stocks. We may scale units of stocks so that one unit of money and the current price for each stock is one. The portfolio choice decision can be represented by a vector $(x_1, \ldots, x_n)$, $1 \le i \le n$, $\sum_{i=1}^{n} x_i = 1$.

To illustrate the competitive analysis method, we first consider the extreme case when we know no information about future prices of the stocks. A simple strategy is to distribute the fund equally to all the stocks such that $x_1 = \cdots = x_n = 1/n$. Let $1 + c_i$ be the price of stock i at the end of the period. Therefore, in retrospective, the best strategy would be to invest all the money in the stock of the best performance: $1 + c_k = \max\{1 + c_i: 1 \le i \le n\}$. The income of the above strategy achieves $\sum_{i=1}^{n}(1+ c_i)/n$. Since we may assume that $1+ c_i \ge 0$, we have

$$\frac{\sum_{i=1}^{n} \frac{1+c_i}{n}}{1 + c_k} \ge \frac{1}{n}.$$

This simple strategy achieves a competitive ratio of $1/n$. On the other hand, it is natural that this strategy is optimal when we have no information whatsoever about the stocks. Consider any strategy which invests x_i in stock i ($\sum_{i=1}^{n} x_i = 1$). Its outcome will be $\sum_{i=1}^{n} x_i(1+c_i)$. Since $\sum_{i=1}^{n} x_i = 1$, there existssome j such that $x_j \le 1/n$. In the worst case, it may happen that we have $1 + c_j = 2$ and $1 + c_i = 0$ for all other stocks. Therefore, the optimal investment will be put all the money in stock c_j. The competitive ratio of this strategy is no more than $x_j(1 + c_j)/(1 + c_j) \le 1/n$. Therefore, the above simple strategy achieves the optimal competitiveratio when no information is available.

To illustrate this idea further, consider another case is when we have some information about future prices of the stocks. Suppose that the onlyinformation we have is that stock i will fluctuate between $[(1 - \epsilon_i), (1 + \delta_i)]$ $(-\epsilon_i \le \delta_i)$, $1 \le i \le n$. It is easy to see that we may normalize the value versus the risk-free rate of interest and make it as the first option so that $-\epsilon_1 = \delta_1 = 0$. E.g., we may divide outcomes of other securities by $(1 + r)$, the *riskless interest rate*.

Given a portfolio choice decision, x_i, $1 \le i \le n$, $\sum_{i=1} x_i = 1$. That is, one unit of investment is distributed to n options with a fraction of x_i on option i: $1 \le i \le n$. Let $(1 + c_i)$ be the unknown future price of option i by the projected time of sales $(-\epsilon_i \le c_i \le \delta_i)$. In retrospect, the optimal solution wouldhave been $\max_{j=1}^{n}(1 + c_j)$ by investing all one unit on the option

achieving the optimum. For a fixed strategy of assigning x_i: $1 \le i \le n$, its ratio versus the optimum will be

$$\frac{\sum_{i=1}^{n}(1+c_i)x_i}{\max\{1+c_i:\ 1 \le i \le n\}}.$$

Taking all situations into consideration, the competitive ratio of this strategy is

$$\min\left\{\frac{\sum_{i=1}^{n}(1+c_i)x_i}{\max\{1+c_j:\ 1 \le j \le n\}}:\ -\epsilon_i \le c_i \le \delta_i\right\},$$

where the minimum is taken over all the ranges of c_i: $-\epsilon_i \le c_i \le \delta_i$, $1 \le i \le n$. Suppose now that $\max_{j=1}^{n}(1+c_j)$ is achieved at some i: $1 \le i \le n$. Then, the above ratio is at least x_i since $x_j \ge 0$ and $1 + c_j \ge 0$, for all j: $1 \le j \le n$. If $c_i < \delta_i$, the adversary can choose a new value $c'_i = \delta_i$. In this case, the denominator $\max_{j=1}^{n}(1 + c_j)$ increases by $\delta_i - c_i > 0$. The numerator $\sum_{i=1}^{n}(1 + c_i)x_i$ increases by $(\delta_i - c_i)x_i$. Thereforeit is to the benefit of the adversary to choose $c_i = \delta_i$. Similarly, it is to the benefit of the adversary to set $c_j = -\epsilon_i$, for all $j \ne i$. That is, the minimum ratio is achieved at

$$\frac{(1+\delta_i)x_i + \sum_{j\ne i}(1-\epsilon_j)x_j}{1+\delta_i}$$

for some i, $1 \le i \le n$. Therefore, the adversary will choose some i such that

$$\min\left\{\frac{(1+\delta_i)x_i + \sum_{j\ne i}(1-\epsilon_j)x_j}{1+\delta_i}:\ 1 \le i \le n\right\}.$$

Given a portfolio decision vector x, we can search through all n possible situations to find the minimum in *polynomial time*. This allows us to evaluate the quality of portfolio choices in terms of their competitive ratios.

As a portfolio manager aiming at a solution with the best competitive ratio, its goal is to choose the decision vector x which maximizes

$$\min\left\{\frac{(1+\delta_i)x_i + \sum_{j\ne i}(1-\epsilon_j)x_j}{1+\delta_i}:\ 1 \le i \le n\right\}.$$

In a *linear program* formulation, this is

$$\begin{cases} \max & z \\ \text{s.t.} & \frac{(1+\delta_i)x_i+\sum_{j\ne i}(1-\epsilon_j)x_j}{1+\delta_i} \ge z, \\ & 1 \le i \le n, \\ & x_i \ge 0, \quad \sum_{i=1}^{n} x_i = 1. \end{cases}$$

Therefore, the optimal competitive ratio for the above portfolio management problem can be solved in polynomial time.

In the general case, information about future may be different for different investigators. Compare two situations where two investors each has two options, one government bond of riskless interest rate $1 + r$ and a security. One investor knows that the future price of the security will be in $[1 + \epsilon, 1 + \delta]$ with $\epsilon < \delta$ and another knows nothing about future prices of the security. The most interesting case will be $2\epsilon < r < \delta$. Apply the above analysis, the more informed investor will decide a proportion x of his fortune on riskless bond with

$$\frac{x(1+r)+(1-x)(1+\epsilon)}{1+r} = \frac{x(1+r)+(1-x)(1+\delta)}{1+\delta}.$$

Therefore, it invests

$$x = \frac{(1+\delta)(r-\epsilon)}{(1+\delta)(r-\epsilon)+(1+r)(\delta-r)}$$

on the riskless bond and

$$1 - x = \frac{(1+r)(\delta-r)}{(1+\delta)(r-\epsilon)+(1+r)(\delta-r)}$$

on the other security. Its competitive ratio will be

$$\frac{(1+\delta)(r-\epsilon)+(1+\epsilon)(\delta-r)}{(1+\delta)(r-\epsilon)+(1+r)(\delta-r)}.$$

Applying the analysis above, we see that the person knowing nothing will invest 1/2 for the riskless bond and 1/2 for the other security. However, the worst situations considered by the less informed investor would not occur at all. Therefore, its competitive ratio will be the minimum of

$$\frac{(1+r)+(1+\epsilon)}{2(1+r)}$$

and

$$\frac{(1+r)+(1+\delta)}{2(1+\delta)}.$$

From the above discussion, the decision of the investor knowing nothing has a worse competitive ratio than that of the more informed one.

In comparison with the general approach of using probability distribution for events of uncertainty, the above situation shows that the *competitive analysis* method allows analysis for information asymmetry of investors. It is not easy to apply the probability method here since, in principle, the real world should not have two different probability distributions. This advantage is not only for the above case when the range of future prices is known. It can also be applied to other types of information about future.

Other decision rules based on rationality other than probability argument have also been suggested for financial problems. In particular, T.M. Cover has suggested a solution, the *universal portfolio*, which requires no information (not even probability distribution) about the future prices of the stocks under consideration [1]. In contrast to competitive analysis which evaluates a strategy with all other strategies, Cover has evaluated his solution in comparison with a class of strategies called constant rebalanced portfolio, which maintains a fixed proportion of one's fortune in each of the securities. Notice that, this would require frequent adjustment the holdings of the securities as their prices change. Surprisingly, Cover has shown his solution to approximate, under mild conditions, the best constant rebalanced portfolio (chosen after the stock outcomes are known) which out-perform any *constant rebalanced portfolio*, any single stock and index fund such as Down Jones Index Average (DJIA) [1]. However, Cover's algorithm requires higher-dimensional integration to calculate his solution and the dimension grows with the number of securities under consideration. This may make it computationally difficult to apply this method. In comparison, the competitive analysis would suggest a solution which is a constant rebalanced portfolio with the same weight for all the securities.

Dembo and King have discussed a tracking model for asset allocation which minimizes an investor's regret (defined as the difference of the solution of a strategy under incomplete information and the optimal solution) distribution in the L_2 metric [2]. In general, one may express the regret of a decision maker with strategy S as a function of $f(v_S(X), \mathrm{OPT}(X))$, where X is the revealed future event, $v_S(X)$ is the value achieved under strategy S operating under ignorant of the future event X, and $\mathrm{OPT}(X)$ is the optimal value achievable knowing the complete information. One such function often used is the L_∞ metric distance of these two values in the feasible space [10]. However, since the authors use the absolute difference for the basis of evaluation of strategies, probability assumption is still necessary in this model. The competitive analysis and the solution of Cover [1] base the evaluation on the ratio of the performance of a strategy with unknown information and the performance of the best solution in the class of strategies under consideration.

R.M. Hogarth and H. Kunreuther have discussed situations when financial decisions are made under ignorance. They have designed experiments to study it by evaluating human empirical judgements. However, decision making processes ofeconomic agents are ignored in this study [6].

Some information is still available in reality, though not necessarily in the form of a well shaped probability distribution. The competitive analysis provides an approach which does not rely on probability distribution, allows for analysis under *asymmetrical information* of agents in the market, and in principle, has no difficulty to include available information in the analysis. The remaining difficulties in applying it successful to portfolio management are mainly modeling of available information and *efficient algorithms* for computational purpose.

See also

- ► Financial Applications of Multicriteria Analysis
- ► Financial Optimization
- ► Portfolio Selection and Multicriteria Analysis
- ► Robust Optimization
- ► Semi-Infinite Programming and Applications in Finance

References

1. Cover TM (1991) Universal portfolios. Math Finance 1:1–29
2. Dembo RS, King AJ (1992) Tracking models and the optimal regret distribution in asset allocation. Applied Stochastic Models and Data Analysis 8:151–157
3. Deng X (1996) Portfolio management with optimal regret ratio. Proc Internat Conf Management Sci, Hong Kong, pp 289–295
4. Deng X, Papadimitriou CH (1996) Competitive distributed decision-making. Algorithmica 16:133–150
5. El-Yaniv R, Fiat A, Karp RM, Turpin G (1992) Competitive analysis of financial games. In: Proc. 33rd Annual Symp. Foundations of Computer Sci., pp 327–333

6. Hogarth RM, Kunreuther H (1995) Decision making under ignorance: Arguing with yourself. J Risk and Uncertainty 10:15–36
7. Karlin AR, Manasse MS, Rudolph L, Sleator DD (1988) Competitive snoopy caching. Algorithmica 3:79–119
8. Manasse MS, McGeoch LA, Sleator DD (1990) Competitive algorithms for on-line problems. J Algorithms 11: 208–230
9. Markowitz HM (1959) Portfolio selection: efficient diversification of investments. Wiley, New York
10. Savage LJ (1951) The theory of statistical decision. J Amer Statist Assoc 46:55–67
11. Sleator DD, Tarjan RE (1985) Amortized efficiency of list update and paging rules. Comm ACM 28:202–208

Complementarity Algorithms in Pattern Recognition

C. Cifarelli, Laura Di Giacomo, Giacomo Patrizi
Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università di Roma "La Sapienza", Rome, Italy

Article Outline

Introduction
Definitions
Formulation
Methods and Applications
Models
Cases
Conclusions
See also
References

Introduction

Basic to the process of human understanding and learning, the problem of recognition, which includes classification and machine learning and the more general approach of pattern recognition, consists of a set of algorithms or procedures to determine in which of a number of alternative classes an object belongs.

While recognition is a human process whose functioning is largely unknown [11], pattern recognition and classification and machine learning are algorithms or heuristic procedures with a precise functional characterization to determine as precisely as possible the class membership of an object.

The two approaches, pattern recognition on the one hand and classification and machine learning on the other, emphasize two different aspects of the learning methodology, similar to a distinction often made in numerical analysis between extrapolation and interpolation [13].

In pattern recognition, given a feature of a population, it is desired that all objects that belong to that population be recognized with an acceptable small error, since the paramount aspect of this activity is to recognize the object so as to be able to proceed accordingly. It is not of interest to diagnose a varying percentage of sick individuals, but rather it is essential to recognize correctly the pathology. Thus in pattern recognition, given an object, it is desired to determine if the object belongs to the population specified and, if so, to determine precisely to which class it belongs [5].

In classification and machine learning, a population is considered given and some objects belong to known classes, while other objects belong to as yet unknown classes, so it is desired to determine the class membership of objects that are known to belong to that population. Depending on the definition of the populations considered and the algorithms used, the classification rate may differ from one application to another. Classification and machine learning procedures are often defined in terms of heuristics, such as support vector machines with kernel methods. The kernel to be applied to a given problem cannot be determined except by trial and error, so that the existence of a suitable kernel is not guaranteed. Thus results may differ markedly from application to application [5].

Here we shall be concerned with pattern recognition problems that must consider:

- The collection of objects to examine and the training set available for learning the classes.
- The attributes that can be defined precisely on the objects in the training set and on the objects to be recognized (which may be as yet unknown).
- The precision with which the recognition is required, as well as the possible structures defined on the data sets.

The pattern recognition algorithm used to perform this will be formulated as a complementarity problem rather than an optimization algorithm as it may be con-

sidered more general, and the known differences that may exist in the attributes of the classes allows additional constraints to be defined, which permit more precise results to be obtained.

Definitions

Consider a set of objects, characterized by a set of common attributes, which have been assigned to suitable classes, so that their class labels are known. This is called a training set [5].

Definition 1 A subset of a data set is termed a training set if every entity in the training set has been assigned a class label.

Definition 2 Suppose there is a set of entities E and a set $P = \{P_1, P_2, \ldots, P_n\}$ of subsets of the set of entities, i. e. $P_j \subseteq E$, $j \in J = \{1, 2, \ldots, n\}$. A subset $\hat{J} \subseteq J$ forms a cover of E if $\bigcup_{j\in\hat{J}} P_j = E$. If, in addition, for every k, $j \in \hat{J}$, $j \neq k$, $P_j \cap P_k = \emptyset$, it is a partition.

Definition 3 The data set is coherent if there exists a partition that satisfies the following properties:

1. The relations defined on the training set and in particular the membership classes, defined over the data set, consist of disjoint unions of the subsets of the partition.
2. Stability: the partition is invariant to additions to the data set. This invariance should apply both to the addition of duplicate entities and to the addition of new entities obtained in the same way as the objects under consideration.
3. Extendability: if the dimension of the set of attributes is augmented, so that the basis will be composed of $p + 1$ attributes, then the partition obtained by considering the smaller set will remain valid, even for the extension, as long as this extension does not alter the relations defined on the data set.

Definition 4 A data set is linearly separable if there exist linear functions such that the entities belonging to one class can be separated from the entities belonging to the other classes. It is pairwise linearly separable if every pair of classes is linearly separable. A set is piecewise separable if every element of each class is separable from all the other elements of all the other classes.

Clearly if a set is linearly separable, it is pairwise linearly separable and piecewise separable, but the converse is not true. The following results are straightforward:

Theorem 1 *If a data set is coherent, then it is piecewise separable.*

A given class is formed from distinct subsets of the partition, so no pattern can belong to two classes. Therefore each pattern of a given class will be separable from every pattern in the other subsets of the partition. Consequently the data set is piecewise separable.

Theorem 2 *Given a data set that does not contain two identical patterns assigned to different classes, a correct classifier can be formulated that realizes the given partition on this training set.*

Corollary 1 *Given that the training set does not contain two or more identical patterns assigned to different classes, the given partition yields a completely correct classification of the patterns.*

The avoidance of the juxtaposition property, i. e. two identical patterns belong to different classes, entails that the Bayes error is zero [2].

In general this does not mean that in any given neighborhood of a pattern there cannot be other patterns of other classes, but only that they cannot lie on the same point. Thus the probability distribution of the patterns with respect to the classes may overlap, if such distributions exist, although they will exhibit discontinuities in the overlap region, so that juxtaposition is avoided.

Formulation

The classification algorithm to be formulated may be specified as a combinatorial problem in binary variables [6].

Suppose that a training set is available with n patterns, represented by appropriate feature vectors indicated by $x_i \in \mathbf{R}^p, \forall\, i = 1, 2, \ldots, n$ and grouped in c classes. An upper bound is selected to the number of barycentres that may result from the classification, which can be taken "ad abundantiam" as m, or on the basis of a preliminary run of some classification algorithm.

The initial barycenter matrix will be an $p \times mc$ matrix which is set to zero. The barycentres when calculated will be written in the matrix by class. Thus a barycenter of class k will occupy a column of the matrix between $(m(k-1)+1)$ and mk.

Since we are considering a training set, the feature vectors can be ordered by increasing class label. Thus the first n_1 columns of the training set matrix consists of patterns of class 1, from $n_1 + 1$ to n_2 of class 2 and in general from $n_{k-1} + 1$ to n_k of class k.

Thus consider the following inequality constrained optimization problem, from which we shall derive the non-linear complementarity specification. Let the following hold:

- $x_i \in \mathbf{R}^p$: the p-dimensional pattern vector of pattern i;
- c classes are considered, $k = 0, 1, \ldots, (c-1)$. Let the number of patterns in class c_k be indicated by n_k; then the n patterns can be subdivided by class so that $n = \sum_{k=0}^{c-1} n_k$;
- $z_j \in \{0, 1\}$, integer: $\{j = 1, 2, \ldots mc\}$ if $z_j = 1$ then the barycenter vector $j \in \{mk+1\}, \ldots, m(k+1)\}$ belonging to recognition class $c_k \in \{0, \ldots, c-1\}$,
- $y_{ij} \in \{0, 1\}$, integer: pattern i has been assigned to the barycenter j $(y_{ij} = 1)$;
- $t_j \in \mathbf{R}^p$: the sum of the elements of the vectors of the patterns assigned to the barycenter $j = \{1, 2, \ldots, mc\}$;
- M is a large scalar.

$$\text{Min } Z = \sum_{j=1}^{mc} z_j \tag{1}$$

$$\text{s.t.} \sum_{j=km+1}^{m(k+1)} y_{ij} - 1 \geq 0 \quad \forall k = 0, 1, \ldots, (c-1); \quad \forall i = n_{k-1} + 1, \ldots, n_k \tag{2}$$

$$-\sum_{i=1}^{n}\sum_{j=1}^{mc} y_{ij} + n \geq 0 \tag{3}$$

$$Mz_j - \sum_{i=1}^{n} y_{ij} \geq 0 \quad \forall j = 1, 2, \ldots, mc \tag{4}$$

$$t_j - \sum_{i=1}^{n} x_i y_{ij} \geq 0 \quad \forall j = 0, 1, \ldots, mc \tag{5}$$

$$-\sum_{j=1}^{mc}\left(t_j - \sum_{i=1}^{n} x_i y_{ij}\right) \geq 0 \tag{6}$$

$$\left(x_i - \frac{t_h}{\sum_{s=lm+1}^{m(l+1)} y_{sh}}\right)^{\mathrm{T}} \left(x_i - \frac{t_h}{\sum_{s=lm+1}^{m(l+1)} y_{sh}}\right) - \sum_{j=km+1}^{m(k+1)} \left(x_i - \frac{t_j}{\sum_{r=km+1}^{m(k+1)} y_{rj}}\right)^{\mathrm{T}} \left(x_i - \frac{t_j}{\sum_{r=km+1}^{m(k+1)} y_{rj}}\right) \times y_{ij} \geq 0$$
$$\forall i = 1, 2, \ldots, n; \quad h = 1, 2, \ldots, mc; \quad k, l = 0, 1, \ldots, c-1; \tag{7}$$

$$z_j, y_{ij} \in \{0, 1\} \text{ integer}. \tag{8}$$

The solution of this optimization problem assigns each pattern to a mean vector, called a barycenter $(z_j, j = 1, 2, \ldots, mc)$, whose values are given by the vectors $t_j \in \mathbf{R}^p$, $j = \{1, 2, \ldots, mc\}$ divided by the number of patterns assigned to that barycenter. The least number of barycentres, indicated by the objective function Eq. (1), which will satisfy the stated constraints is determined.

The n constraints Eqs. (2) and (3) state that each feature vector from a pattern in a given class must be assigned to some barycenter vector of that class. As patterns and barycentres have been ordered by class, the summation should be run over the appropriate index sets.

The mc constraints Eq. (4) impose that no pattern be assigned to a non-existing barycenter.

Instead, constraints Eqs. (5) and (6) determine the vector of the total sum element by element assigned to a barycenter. Notice that x_i is a vector, so the number of inequalities will be $2mc$ times the number of elements in the feature vector.

The last set of inequalities Eq. (7) indicates that each feature vector must be nearer to the assigned barycenter of its own class than to any other barycenter. Should the barycenter be null, this is immediately verified, while if it is non-zero, this must be imposed.

Finally, Eq. (8) indicates that the vectors $z \in R^{mc}$ and $y \in R^{nmc}$ are binary.

The solution will determine that each pattern of the training set is nearer to a barycenter of its own class than to a barycenter of another class. Each barycenter has the class label of the patterns assigned to it, which will belong by construction to a single class. This defines a partition of the pattern space.

A new pattern can be assigned to a class by determining its distance from each barycenter formed by the algorithm and then assigning the pattern to the class of the barycenter to which it is nearest.

In general, other constraints which characterize relationships between objects of different classes can be easily introduced in this specification, as well as dynamical relationships regarding the attributes of the objects.

The problem can also be formulated as a non-linear complementarity problem in binary variables, which will be solved through iterating on a set of linear complementarity problems in binary variables, by using a linear programming technique with parametric variation in one scalar variable [7] which has given good results [3].

For simplicity in the representation and analysis, write the constraints (7) as:

$$
\begin{aligned}
& g(y,x,t) \\
& = \left(x_i - \frac{t_h}{\sum_{s=lm+1}^{m(l+1)} y_{sh}}\right)^{\mathrm{T}} \left(x_i - \frac{t_h}{\sum_{s=lm+1}^{m(l+1)} y_{sh}}\right) \\
& - \sum_{j=km+1}^{m(k+1)} \left(x_i - \frac{t_j}{\sum_{r=km+1}^{m(k+1)} y_{rj}}\right)^{\mathrm{T}} \left(x_i - \frac{t_j}{\sum_{r=km+1}^{m(k+1)} y_{rj}}\right) \\
& \times y_{ij} \qquad (9)
\end{aligned}
$$

The following additional notation should be adopted to write the optimization problem (1)–(8) as a non-linear complementarity problem:

- e is an appropriate dimensional vector of ones.
- $E \in \mathbf{R}^{n \times nmc}$ is a matrix composed of mc identity matrices of dimension $n \times n$.
- $H \in \mathbf{R}^{mc \times n}$ matrix of ones.
- η is a scalar to be assigned by dichotomous search during the iterations.

The data matrix of patterns indicated as X of dimension $(p \times m \times c) \times (n \times m \times c)$ is written in diagonal block form with blocks of dimension $p \times n$ elements containing the original data matrix.

This block is repeated mc times with the first element of the block placed at the position $\big((j-1)p+1, (j-1)n\big)$, $j = 1, 2, \ldots, mc$.

In fact the size of matrices E, H and X can be greatly reduced in applications since the patterns in the training set are ordered conformably with the barycenter vector $t = \{t_j\} \in \mathbf{R}^{pmc}$ and each class is of known cardinality.

The non-linear complementarity problem can therefore be written as:

$$
\begin{pmatrix} -z \\ -y \\ 0 \\ Ey \\ -e^{\mathrm{T}} y \\ Mz - Hy \\ t - Xy \\ -e^{\mathrm{T}}(t - Xy) \\ g(y,x,t) \\ -e^{\mathrm{T}} z \end{pmatrix} + \begin{pmatrix} e \\ e \\ 0 \\ -e \\ n \\ 0 \\ 0 \\ 0 \\ 0 \\ \eta \end{pmatrix} \geq 0 \qquad (10)
$$

$$
\begin{pmatrix} z \\ y \\ t \\ \lambda_1^{\mathrm{T}} \\ \lambda_2^{\mathrm{T}} \\ \lambda_3^{\mathrm{T}} \\ \lambda_4^{\mathrm{T}} \\ \lambda_5^{\mathrm{T}} \\ \lambda_6^{\mathrm{T}} \\ \lambda_7^{\mathrm{T}} \end{pmatrix} \geq 0 \qquad (11)
$$

$$
\begin{aligned}
& \begin{pmatrix} z^{\mathrm{T}}, & y^{\mathrm{T}} & t & \lambda_1^{\mathrm{T}}, & \lambda_2^{\mathrm{T}}, & \lambda_3^{\mathrm{T}}, & \lambda_4^{\mathrm{T}}, & \lambda_5^{\mathrm{T}}, & \lambda_6^{\mathrm{T}}, & \lambda_7^{\mathrm{T}} \end{pmatrix} \\
& \times \left(\begin{pmatrix} -z \\ -y \\ 0 \\ Ey \\ -e^{\mathrm{T}} y \\ Mz - Hy \\ t - Xy \\ -e^{\mathrm{T}}(t - Xy) \\ g(y,x,t) \\ -e^{\mathrm{T}} z \end{pmatrix} + \begin{pmatrix} e \\ e \\ 0 \\ -e \\ n \\ 0 \\ 0 \\ 0 \\ 0 \\ \eta \end{pmatrix} \right) = 0 \qquad (12)
\end{aligned}
$$

Binary values to the z, y variables are imposed by the constraints Eq. (10) and the complementarity condition Eq. (12).

Finally, by recursing on the parameter η fewer and fewer barycentres will be created, as long as the problem remains feasible and thus ensuring a minimal solution.

Methods and Applications

The aim of this section is to describe the method to solve the non-linear complementarity problem specified in the previous section. The convergence of the non-linear complementarity problem Eqs. (10)–(12) has been given elsewhere [1].

In a small enough neighborhood, the approximation of the non-linear complementarity problem by a linear complementarity problem will be sufficiently accurate so that, instead of solving the original system, a linear complementarity system approximation can be solved, which may be thus represented:

$$\begin{pmatrix} -I & 0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & -I & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & 0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & E & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & -e^{\mathrm{T}} & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ MI & -H & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & -X & I & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & e^{\mathrm{T}}X & -e^{\mathrm{T}} & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & \nabla g_y(\hat{t},\hat{y}) & \nabla g_t(\hat{t},\hat{y}) & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & 0 & D & 0\,0\,0\,0\,0\,0\,0\,0 \\ -e^{\mathrm{T}} & 0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \end{pmatrix} \begin{pmatrix} z \\ y \\ t \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \\ \lambda_8 \end{pmatrix}$$

$$+ \begin{pmatrix} e \\ e \\ 0 \\ -e \\ n \\ 0 \\ 0 \\ 0 \\ -g(\hat{t},\hat{y}) + \nabla g(\hat{t},\hat{y})\hat{y} \\ -d \\ \eta \end{pmatrix} \geq 0 \tag{13}$$

$$\begin{pmatrix} z \\ y \\ t \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \\ \lambda_8 \end{pmatrix} \geq 0 \tag{14}$$

$$\left(z^{\mathrm{T}},\ y^{\mathrm{T}},\ t^{\mathrm{T}},\ \lambda_1^{\mathrm{T}},\ \lambda_2,\ \lambda_3^{\mathrm{T}},\ \lambda_4^{\mathrm{T}},\ \lambda_5,\ \lambda_6^{\mathrm{T}},\ \lambda_7^{\mathrm{T}},\ \lambda_8\right) \times$$

$$\left(\begin{pmatrix} -I & 0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & -I & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & 0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & E & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & -e^{\mathrm{T}} & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ MI & -H & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & -X & I & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & e^{\mathrm{T}}X & -e^{\mathrm{T}} & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & \nabla g_y(\hat{t},\hat{y}) & \nabla g_t(\hat{t},\hat{y}) & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0 & 0 & D & 0\,0\,0\,0\,0\,0\,0\,0 \\ -e^{\mathrm{T}} & 0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0 \end{pmatrix} \begin{pmatrix} z \\ y \\ t \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \\ \lambda_8 \end{pmatrix} + \begin{pmatrix} e \\ e \\ 0 \\ -e \\ n \\ 0 \\ 0 \\ 0 \\ -g(\hat{t},\hat{y}) + \nabla g(\hat{t},\hat{y})\hat{y} \\ -d \\ \eta \end{pmatrix} \right) = 0 \tag{15}$$

The problem (10)–(12) is then solved by expanding the vectorial function $g(y,x,t)$ in a Taylor series around the iteration point and solving the resulting linear complementarity problem approximation (13)–(15) of the given non-linear complementarity problem within a suitable trust region. It is easy to show:

Theorem 3 *The following are equivalent:*

1. *The non-linear optimization problem defined by (1)–(8) has a solution;*

2. *The non-linear complementarity problem defined by (10)–(12) has a solution;*
3. *The linear complementarity problem defined by (13)–(15) has a solution.*

Thus the computational specification of this algorithm is:

Algorithm 1 (CASTOR)
Begin;
- **Given**: a training set $A \in R^{p\times n}$ with n patterns each with p elements belonging to c classes;
- **Construct**: the matrices $E \in \mathbf{R}^{n\times nmc}$, $H \in \mathbf{R}^{mc\times n}$, $X \in \mathbf{R}^{(pmc)\times(mnc)}$, $D \in \mathbf{R}^{pmc\times pmc}$;
- **Set** y^0, d^0, η^0;

For $k = 1, 2, \ldots$;
- **while** $z^{k+1}, y^{k+1}, t^{k+1}$ is a solution to LCP Eqs. (13)–(15) **Do**;
- **Begin**: recursion on $g(x,y,t)$
  - **while** $(z^{k+1}, y^{k+1}, t^{k+1}) \neq (z^k, y^k, t^k)$ **Do**;
  - $(z^k, y^k, t^k) \leftarrow (z^{k+1}, y^{k+1}, t^{k+1})$
  - **Determine** $\nabla g_y(x^k, y^k, t^k,)$
    *Begin: dichotomous search on η^k;
    $(z^{k+1}, y^{k+1}, t^{k+1}) \leftarrow LCP(z^k, y^k, t^k)$
    *end;
- **end**;

the solution is (z^k, y^k, t^k)
end;

The termination of the classification algorithm may now be proved under a consistency condition.

Theorem 4 *Given a set which does not contain two identical patterns assigned to different classes, a correct classifier will be determined by Algorithm 1.*

Models

Suppose a training set is available, defined over a suitable representation space, which is piecewise separable and coherent, what properties should such a training set have to determine a precise classification with regard to a set of data of as yet unknown classes?

The algorithm **CASTOR** (**C**omplementarity **A**lgorithm **S**ystem for **TO**tal **R**ecognition) described in the Sects. "Formulation" and "Methods" will determine a classification rule to apply, on the data set, just that partition which has been found for the training set, so that to each entity in the data set a class is assigned. If the training set forms a random sample and the data set which includes the training set is coherent, then this classification can be performed to any desired degree of accuracy by extending the size of the training sample. Sufficient conditions to ensure that these properties hold are given by selecting the data set and the verification set by non-repetitive random sampling.

Theorem 5 *Suppose that the data set is coherent; then the data set can be classified correctly.*

To avoid having to introduce distributional properties on the data set considered, the empirical risk minimization inductive principle may be applied [12]:

Definition 5 A data set is stable, according to definition 3, with respect to a partition and a population of entities if the relative frequency of misclassification is $R_{emp}(\alpha^*) \geq 0$ and

$$\lim_{n\to\infty} pr\{R_{emp}(\alpha^*) > \epsilon\} = 0, \tag{16}$$

where α^* is the classification procedure applied, $\epsilon > 0$ for given arbitrary small value and $pr\{.\}$ is the probability of the event included in the braces.

In some diagnostic studies the set of attributes considered have no significant relationship with the outcome or the classification of the entity. Typically the classes could be eye color and the attributes the weight, height and sex of a person. Such a classification would be spurious since there is no relation between eye color and body indices.

A spurious collection of entities, in which there is no similarity relations, may occur and should be recognized. With this algorithm, this occurrence is easily determined, as very many barycentres are formed, almost one per object. Such spuriousness may arise even in the presence of some meaningful relationships in the data, which are, however, swamped by noise, and so data reduction techniques may be useful [5].

In general, by considering smaller and smaller subsets of the attribute space X, if there exists a relationship between the attributes and the classes of the entities, for certain of these subsets the frequency of the entities of a given class will increase to the upper limit of one, while in other subsets it will decrease to a lower limit of zero. Thus for a very fine subdivision of the attribute space, each subset will tend to include entities only of a given class.

Definition 6 A proper subset S_k of the attribute space X of the data set will give rise to a spurious classification if the conditional probability of a pattern belonging to a given class c is equal to its unconditional probability over the attribute space. The data set is spurious if this holds for all subsets of the attribute space X.

$$pr\{y_i = c \mid (y_i, x_i) \cap S_k\} = pr\{y_i = c \mid (y_i, x_i) \cap X\} \tag{17}$$

The following results can now be presented, which are proved elsewhere [1].

Theorem 6 *Consider a training set of n patterns randomly selected, assigned to two classes, where the unconditional probability of belonging to class one is p. Let a be a suitable large number and let* $(n > a)$. *Let the training set form* b_n *barycentres. Then, under* ***CASTOR****, this training set will provide a spurious classification if*

$$\frac{b_n}{n} \geq (1 - p) \quad n > a. \tag{18}$$

Theorem 7 *Let the probability of a pattern belonging to class one be p. Then the number of barycentres required to partition correctly a subset S, containing* $n_s > a$ *patterns, which is not spurious, formed from the* ***CASTOR*** *algorithm, is* $b_s < n_s$, $\forall n_s > a$.

Corollary 2 ([12]) *The Vapnik–Cervonenkis dimension (VC dimension), s(C,n), for the class of sets defined by the* ***CASTOR*** *algorithm, restricted to the classification of a non-spurious data set which is piecewise separable, with* n_s *elements and two classes, is less than* 2^{n_s}, *if* $n_s > a$.

Theorem 8 ([2]) *Let C be a class of decision functions and* ψ_n^* *a classifier restricted to the classification of a data set which is not spurious and returns a value of empirical error equal to zero based on the training sample* $(z_1, z_2, \ldots, z_n)$. *Thus* $Inf_{\psi \in C} L(\psi) = 0$, *i. e. the Bayes decision is contained in C. Then*

$$pr\left\{L(\psi_n^*) > \epsilon\right\} \leq 2s(C, 2n)2^{\frac{-n\epsilon}{2}}. \tag{19}$$

By calculating bounds on the VC dimension, the universal consistency property can be established for this algorithm applied to the classification of a data set which is not spurious.

Corollary 3 ([5]) *A non-spurious classification problem with a piecewise separable training set is strongly universally consistent.*

Cases

To use the **CASTOR** algorithm in applications, it is necessary to determine, first, whether the data set is spurious or not, for the given problem with the specific pattern vectors adopted. The way the pattern vectors are defined based on the data available may affect strongly the results obtainable.

Further, the coherence of the data set must be tested to ensure that the patterns extracted are sufficiently rich to ensure the proper classification, stability and extendability of the data set (Definition 5). Then the algorithm can be applied, but the results will only hold if the data set, training set and verification set are random samples, taken from a known or unknown population, as otherwise the sample may not be representative of the population.

Note that with this method, if the data come from a set of unknown populations, a suitable partition of the data set will form accordingly, even though the operator may not know to which population an individual barycenter belongs. If the number of objects coming from different populations is so high with respect to the training set, then the problem may be recognized as spurious, only to signify that too many barycentres are formed with respect to the available training objects [8,9].

Consider a set of proteins randomly sampled from a population of proteins, and the set of proteins whose structure should be determined also belongs to that population, but are of unknown structure. Probability limits can be imposed on the likelihood of the structure identified being the correct one. Therefore, accurate limits on the precision of the recognition of the the new protein's structure can be specified.

Results could be obtained also by selecting "purposefully" representative proteins and subjecting these to a suitable algorithm. The results could be better on particular sets than the asymptotic mean precision measures, but generally, and, almost surely, on using new data, the results will turn out to have a greater variance and a lower mean precision, as is well known from sampling theory [4]. Thus to minimize the asymp-

Complementarity Algorithms in Pattern Recognition, Table 1
Q_3 Classification results on the Rost 126 verification set by similarity classes (15 proteins selected)

Sim. class	CASTOR	PHD	DSC	PRED	MUL	NNSSP	Zpred	CONS
0	0.82	0.74	0.73	0.72	0.68	0.78.	0.66.	0.80
1	0.96	0.75	0.77	0.64	0.64	0.70	0.76	0.76
3	1.00	0.84	0.87	0.83	0.69	0.83	0.69	0.84
5	1.00	0.81	0.83	0.75	0.76	0.68	0.73	0.77
6	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
7	0.98	0.67	0.69	0.61	0.59	0.66	0.55	0.68

Complementarity Algorithms in Pattern Recognition, Table 2
Q_3 estimate of the classification precision of CASTOR and other classification procedures on 56 randomly selected proteins of the Cuff 513 data set

Sim. class	CASTOR	PHD	DSC	PRED	MUL	NNSSP	Zpred	CONS
0	0.73	0.73	0.71	0.72	0.66	0.71	0.63	0.75
1	0.95	0.79	0.76	0.69	0.68	0.74	0.75	0.78
2	0.87	0.84	0.81	0.89	0.62	0.95	0.65	0.90
3	0.94	0.77	0.79	0.78	0.72	0.82	0.67	0.80
5	1.00	0.81	0.83	0.75	0.76	0.68	0.73	0.77
6	0.95	0.85	0.81	0.80	0.79	0.79	0.78	0.82
7	0.85	0.71	0.70	0.67	0.61	0.69	0.57	0.72
Mean	0.84	0.73	0.72	0.70	0.64	0.71	0.61	0.74

totic misclassification error a sample, as large as possible, drawn randomly from the given population should be used, which will then ensure that under mild conditions the properties derived above are satisfied.

Also a distinction is often introduced regarding the similarity between classes of subsets of proteins [10]. In this case, it may be considered relevant that a cover be defined on the population of proteins, so as to form eight or more subpopulations. The samples can still be drawn randomly from the relevant subpopulation and the classifier determined for each subpopulation. To determine the structure of a new protein, first the similarity in the residue chain must be determined with respect to each subpopulation and then the classifier of the subpopulation with the highest similarity coefficient is applied. Here, the proper sampling method should consist of a stratified non-repetitive random sampling design, but this would be warranted only if there are significant differences in the results for the subpopulations.

The limitations of not using stratified random data sets and using ad hoc heuristics, instead of demonstrably convergent algorithms, is well brought out in the following tables [1], where classification results are compared between the **CASTOR** algorithm and seven popular alternative procedures, for two well-known data sets the Rost 126 and the Cuff 513 data sets.

Table 1 presents the Q_3 classification results on the Rost 126 verification set for the various similarity classes which have appeared in the sample of 15 proteins and in the random verification set. The precision of the classification results found by applying the **CASTOR** algorithm dominate all other procedures, and usually by over 15%.

In Table 2 the Q_3 estimates are given for the classification precision of **CASTOR** and the other classification procedures on 56 randomly selected proteins of the Cuff 513 data set. It is seen that the precision obtained by **CASTOR** dominates all the other entries except four. Two of these entries occur for the CONS algorithm for similarity classes 0 and 2, while the two other entries which dominate the results by the **CASTOR** algorithm occur for similarity class 2, for the procedures PRED and NNSSP.

Conclusions

The experiments described above shed some light on two important aspects, which are very closely related: the sampling procedure to adopt and the classification procedure to apply. Moreover, these results show the essential non-linearity and complexity of the pattern recognition problem.

Random sampling is necessary for precise and stable estimates, with the required accuracy, obtainable in predictions, since invariably the choice of special sets in verification or in training will alter the expected prediction accuracy, as a non-random sample will contain a different distribution of classes from the one regarding the population. As the prediction precision varies with the class distribution, this will have a significant effect on recognition.

Heuristics compared to algorithms will bias the results in the same way: they will be accurate in some cases, unstable in others. Moreover, when the classification results are poor, with a heuristic the source of the problem cannot usually be determined. With an algorithm, such as the one indicated, the root of the problem will invariably be tied to one of the mild assumptions not being satisfied.

This can be checked and remedied.

See also

- ► Generalizations of Interior Point Methods for the Linear Complementarity Problem
- ► Generalized Eigenvalue Proximal Support Vector Machine Problem

References

1. Cifarelli C, Patrizi G (2007) Solving large protein secondary structure classification problems by non-linear complementarity algorithm with {0,1} variables. Optim Softw 22:25–49
2. Devroye L, Gyorfi L, Lugosi G (1996) A Probabilistic Theory of Pattern Recognition. Springer, Berlin
3. Di Giacomo L, Argento E, Patrizi G (2004) Linear complementarity methods for the solution of combinatorial problems. Submitted for publication: copy at http://banach.sta.uniroma1.it/patrizi/
4. Konjin HS (1973) Statistical Theory of Sample Design and Analysis. North Holland, Amsterdam
5. Nieddu L, Patrizi G (2000) Formal properties of pattern recognition algorithms: A review. Eur J Oper Res 120: 459–495
6. Patrizi G (1979) Optimal clustering properties. Ricerca Operativa 10:41–64
7. Patrizi G (1991) The equivalence of an lcp to a parametric linear program with a scalar parameter. Eur J Oper Res 51:367–386
8. Patrizi G, Addonisio G, Giannakakis C, AO Muda, Patrizi G, Faraggiana T (2007) Diagnosis of alport syndrome by pattern recognition techniques. In: Pardalos PM, Boginski VL, Vazacopoulos A (eds) Data Mining in Biomedicine. Springer, Berlin, pp 209–230
9. Patrizi G, Patrizi G, Di Cioccio L, Bauco C (2007) Clinical analysis of the diagnostic classification of geriatric disorders. In: Pardalos PM, Boginski VL, Vazacopoulos A (eds) Data Mining in Biomedicine. Springer, Berlin, pp 231–260
10. Rost B (1999) Twilight zone of protein sequence alignment. Protein Eng 12:85–94
11. Simon HA, Newell A (1972) Human problem solving. Prentice-Hall, Englewood Cliffs, NJ
12. Vapnik VN (1998) Learning Theory. Wiley, New York
13. Watanabe S (1985) Pattern Recognition: Human and Mechanical. Wiley, New York

Complexity Classes in Optimization

H. B. Hunt III
University Albany, New York, USA

MSC2000: 90C60

Article Outline

Keywords
Time and Space Complexity of Turing Machines
Definition of the Complexity Classes
  General Comments
Efficient Reducibility and 'Hard' Problems
See also
References

Keywords

Complexity; Complexity classes

We survey a number of the basic ideas, results, and references, for the complexity classes *P*, *NP*, *CoNP*, *PSPACE*, *DEXPTIME*, *NDEXPTIME*, and *EXSPACE*, the most important complexity classes in optimization. These ideas and results include the following:

i) the time and space complexities of both deterministic and nondeterministic multitape Turing machines as formalized in [1];
ii) the complexity classes above including the known results on their intercontainments; and
iii) the concepts of *polynomial reducibility*, *F-hardness*, and *F-completeness*, for the complexity classes F above.

We present brief historical surveys of the results obtained in ii) and iii). We also briefly survey many of the results on the above complexity classes in the basic references [1,12,16,25,29,30,37], emphasizing results especially relevant to the area of optimization.

The following are a list of some of the basic notation and terminology used here.

Definition 1 A *finite alphabet* Σ is a finite nonempty set of characters. A set of strings over some finite alphabet is said to be a *language*.

Further, we denote 'infinitely often' by 'i. o.'.

Definition 2 By an *exponential function* in n we mean a function $f(n) = 2^{c \cdot n^r}$ where $c, r > 0$ are constants independent of n.

The languages or problems *3-SATISFIABILITY* (*3-SAT*), *3-DIMENSIONAL MATCHING*, *VERTEX COVER*, *CLIQUE*, *HAMILTON CIRCUIT*, and *PARTITION* are defined as in [12]. Thus, for example, the language 3-SAT is defined to be the set of all satisfiable CNF formulas with no more than 3 literals per clause, when suitably encoded as a language over some finite alphabet. We note that this language, its quantified variants, and its succinctly-specified variants are the languages in the literature most widely used to prove *NP-*, *PSPACE-*, *DEXPTIME-*, and *NDEXPTIME-*hardness results (Definition 8 and [12,21,22,29,30]).

Finally, we denote the linear programming, {0, 1}-integer linear programming, integer linear programming, and quadratic programming problems as defined in [12,25,30,37] by LP, {0, 1}-ILP, ILP, and QP, respectively.

Time and Space Complexity of Turing Machines

In the literature of computational complexity, the most common models of computational devices and the problems solvable by such devices are *Turing machines* (TMs) and *language recognition problems*, respectively [1,9,12,15,17,29]. Here, we only consider multitape deterministic and nondeterministic Turing machines (denoted DTMs and NDTMs, respectively) and their associated language recognition problems as described in [1]. Informally, such a Turing machine M consists of the following:

1) a finite state control together with a finite nonempty set Q of possible *states of the control*;
2) finite nonempty *tape* and *input alphabets* T and I, respectively, and distinct symbols b and $\vdash$, denoting 'blank' and 'leftmost cell of tape', such that $I \subset T$ and $b, \vdash \in T - I$;
3) a finite number $k \geq 1$ of *tapes*, each of which is infinite to the right only and is divided into individual *tape cells* such that each cell can contain exactly one symbol in T at any one time;
4) k *tape heads*, one for each tape, each head capable of scanning a single cell at any one time;
5) a *start state* $q^o \in Q$ and a set of *accepting states* $F \subset Q$; and
6) a finite set μ of *moves*, each of the form

$$(q, s_1, \ldots, s_k, r, (t_1, d_1), \ldots, (t_k, d_k)) \in Q \times T^k \times Q \times (T \times \{L, R, S\})^k.$$

M is said to be *deterministic* if, for each $k+1$ tuple $(q, s_1, \ldots, s_k) \in Q \times T^k$, there is at most one move in μ whose initial $k+1$ components are $q, s_1, \ldots, s_k$, respectively. Otherwise, M is said to be *nondeterministic*. A state $q \in Q$ is said to be *final* if there is no move in μ whose first component equals q. We assume that the following two restrictions hold on F and μ:

7) Each accepting state is final.
8) There are no moves

$$(q, s_1, \ldots, s_k, r, (t_1, d_1), \ldots, (t_k, d_k))$$

in μ such that letting $1 \leq i \leq k$, $s_i = \vdash$ and $t_i \neq \vdash$, $s_i \neq \vdash$ and $t_i = \vdash$, or $s_i = \vdash$ and $d_i = L$.

Let $w \in I^*$. A *partial computation of M on w* is a finite sequence $\sigma_w = (m_1, \ldots, m_l)$ with $l \geq 0$ of moves of M such that the following hold:

1) Initially, M is in its start state q^o; the first tape of M holds the string $\vdash w$, one symbol per cell starting at its leftmost cell; the contents of the leftmost cells of each of the other tapes of M equal $\vdash$ the contents of all other cells of M equal b; and each of the tape heads of M scans the leftmost cell of its corresponding tape.

2) When initialized as in 1), M executes the move-rules $m_1, \ldots, m_l$ consecutively and in that order. The *length of the partial computation* $\sigma_w = (m_1, \ldots, m_l)$ of M on w equals l. A partial computation $\sigma_w = (m_1, \ldots, m_l)$ of M on w is said to be an *accepting computation* (respectively, a *nonaccepting computation*) of M on w if, after executing the sequence of moves σ_w, M is in an accepting state (respectively, M is in afinal state that is not an accepting state).

The restrictions 7) and 8) on the sets F and μ ensure that *no* accepting or nonaccepting computation of M on w is an initial subsequence of any other partial computation of M on w and at *no* point during a partial computation of M on w does one of the tape heads of M attempt to move off the left end of its corresponding tape.

The *language accepted by a Turing machine M*, denoted by $L(M)$, is the set of all strings $w \in I^*$ such that there exists an accepting computation on M on w. The *language recognition problem* of M is the problem of verifying, given $w \in I^* \cap L(M)$, that w is, in fact an element of $L(M)$. The time and space complexities of M are defined in terms of partial computations of M on strings $w \in I^*$ as follows:

Definition 3 Let M be a deterministic Turing machine. The *time complexity* of M, denoted by $T_M(\cdot)$, is the function from $\mathbf{N}$ to $\mathbf{N} \cup \{\infty\}$ defined, for all $n \in \mathbf{N}$, by

- $T_M(n) = \max\{l \in \mathbf{N} : l$ is the length of a partial computation of M on w, where $w \in I^n\}$, if this maximum exists;
- $T_M(n) = \infty$ otherwise.

The *space complexity* of M, denoted by $S_M(\cdot)$ is the function from $\mathbf{N}$ to $\mathbf{N} \cup \{\infty\}$ defined, for all $n \in \mathbf{N}$, by

- $S_M(n) = \max\{ s \in \mathbf{N} : s$ is the maximum number of tape cells scanned on any of the tapes of M during a partial computation of M on $w\}$, where $w \in I^n$, if this maximum exists;
- $S_M(n) = \infty$ otherwise.

Let M be a nondeterministic Turing machine. The *time complexity* of M, denoted by $T_M(\cdot)$, is the function from $\mathbf{N}$ to $\mathbf{N}$ defined, for all $n \in \mathbf{N}$, by

- $T_M(n) = 0$ if no string $w \in I^n$ is in $L(M)$;
- $T_M(n) = \max\{ l \in \mathbf{N} : l$ is the minimum length of an accepting computation M on $w\}$, where $w \in I^n \cap L(M)$;
- $T_M(n) = \infty$ otherwise.

The *space complexity* of M, denoted by $S_M(\cdot)$ is the function from $\mathbf{N}$ to $\mathbf{N}$ defined, for all $n \in \mathbf{N}$, by

- $S_M(n) = 0$ if no string $w \in I^n$ is in $L(M)$;
- $S_M(n) = \max\{ m \in \mathbf{N} \mid m$ equals the minimum of the maximum numbers of tape cells scanned on any of the tapes of M during an accepting computation of M on w, where $w \in I^n \cap L(M)\}$;
- $S_M(n) = \infty$ otherwise.

There is a fundamental difference between the definitions of time and space complexity, for DTMs and for NDTMs, respectively. For a DTM M, the functions T_M and S_M are defined in terms of the numbers of moves executed and tape cells scanned in an *arbitrary* partial computation of M on strings $w \in I^*$. In contrast, for an NDTM M, these functions are defined *only* in terms of the numbers of moves executed and tape cells scanned in minimum 'cost accepting computations' of M on strings $w \in I^* \cap L(M)$. The following are two easy implications of this difference:

Proposition 4 *If L is accepted by a DTM M with time and space complexities T_M and S_M, then L is also accepted by an NDTM M' with time and space complexities $T_{M'}$ and $S_{M'}$ such that, for all $n \in \mathbf{N}$,*

$$T_{M'}(n) \leq T_M(n) \quad \textit{and} \quad S_{M'}(n) \leq S_{M'}(n).$$

Proposition 5 *If L is accepted by a DTM M with input alphabet I such that, for all $n \in \mathbf{N}$, $T_M(n) \in \mathbf{N}$ (equivalently, $T_M(n) \neq \infty$), then the language $I^* - L$ is accepted by a DTM M' such that, for all $n \in \mathbf{N}$, $T_M'(n) = T_M(n)$.*

Consequently, deterministic time complexity classes as defined in Definition (6) are closed under complementation. At present (1999), nondeterministic time complexity classes are not known to be closed under complementation.

Definition of the Complexity Classes

In Definition (6) we use Definition (3) to define the time and space complexity classes most relevant to optimization. Next in Theorem (7), we give several basic properties of these classes, whose proofs do not require the concept of polynomial time reducibility defined in Definition (8).

Definition 6 Let $T, S : \mathbf{N} \to \mathbf{N}$.

A Turing machine M is said to be *polynomially time-bounded*, respectively *polynomially space-bounded*, if the function $T_M(n)$, respectively $S_M(n)$, is bounded above by a polynomial function in n. M is said to be *exponentially time-bounded*, respectively *exponentially space-bounded*, if the function $T_M(n)$, respectively $S_M(n)$, is bounded above by an exponential function in n.

- *DTIME*$(T(n))$, respectively *NDTIME*$(T(n))$, is the class of all languages L for which there exist a DTM, respectively an NDTM, M and and a $c > 0$ such that

  $$L = L(M)$$

  and, for all $n \in \mathbf{N}$,

  $$T_M(n) \leq c \cdot T(n).$$

- *DSPACE* $(S(n))$, respectively *NDSPACE* $(S(n))$, is the class of all languages L for which there exist a DTM, respectively an NDTM, M and a $c > 0$ such that

  $$L = L(M)$$

  and, for all $n \in \mathbf{N}$,

  $$S_M(n) \leq c \cdot S(n).$$

- P, NP, $PSPACE$, $DEXPTIME$, $NDEXPTIME$, and $EXSPACE$ are the classes of all languages L such that L is the language accepted by a polynomially time-bounded DTM, a polynomially time-bounded NDTM, a polynomially space-bounded TM, an exponentially time-bounded DTM, an exponentially time-bounded NDTM, and an exponentially space-bounded TM, respectively.
- *CoNP*, respectively CoNDEXPTIME, is the class of all languages L for which there exists an NDTM M with tape alphabet I such that $L = I^* - L(M)$ and the function $T_M(n)$ is polynomially time-bounded, respectively exponentially time-bounded.

Theorem 7

1) *The following containments hold among the complexity classes defined in Definition 6:*
   a) $P \subset NP \cap CoNP$.
   b) $NP, CoNP \subset PSPACE \subset DEXPTIME$.
   c) $DEXPTIME \subset NDEXPTIME \cap CoNDEXPTIME$.
   d) $NDEXPTIME, CoNDEXPTIME \subset EXSPACE$.
2)
   a) *$P = NP$ if and only if $NDTIME(n) \subset P$;*
   b) *$P = PSPACE$ if and only if $\exists \epsilon > 0$ such that $DSPACE(n^\epsilon) \subset P$;*
   c) *$NP = PSPACE$ if and only if $\exists \epsilon \geq 0$ such that $DSPACE(n\epsilon) \subset NP$;*
   d) *for all integers $k \geq 1$, $NDTIME(n^k) \subset DSPACE(n^k)$.*
3) *$PSPACE = DEXPTIME$ if and only if $\exists \epsilon \geq 0$ such that $DTIME(2^{n\epsilon}) \subset PSPACE$.*
4) *If we restrict the classes P and NP to languages over a single letter alphabet, denoting these restrictions by P_{sla} and NP_{sla}, respectively, then*
   a) *$P_{sla} = NP_{sla}$ if and only if $\cup_{k \geq 1} DTIME(2^{kn}) = \cup_{k \geq 1} NDTIME(2^{kn})$;*
   b) *NP_{sla} is closed under complementation if and only if the class $\cup_{k \geq 1} NDTIME(2^{kn})$ is closed under complementation.*

Proof The claims in 1), 2), and 3) of the theorem follow directly from Definitions 3 and 6, the discussion after Definition 3, and simple well-known arguments involving 'padding', e. g. see [4,13,17]. As an example, let $L \in DSPACE(n^l)$, where $l \geq 1$ is an integer. Let $k \geq 2$ be an integer. Let $L' = \{w \cdot \#^m : m = k \cdot l, w \in L\}$, with $\#$ a symbol not occurring in L. Then $L' \in DSPACE(n^{1/k})$; and $L' \in P$ (respectively $L' \in NP$) if and only if $L \in P$ (respectively $L \in NP$). The claims of 4) follow from simple arguments about 'tally languages', see [13].

General Comments

The Turing machine model is due to A.M. Turing [36]. Additional discussions of this model can be found in [1,9,17].

The time and space complexity of DTMs were first studied in [15], and [14], respectively.

The time-bounded complexity classes *P*, *NP*, *CoNP*, etc. are invariant under several otherformal computer models, including multihead multitape Turing machines, Turing machines with multidimensional tapes, and both the *RAM* and *RASP* models of [35] and [11] under the logarithmic cost function. (For a detailed discussion of this, see [1, Chap. 1].

It is widely assumed that a computational problem is 'practically computationally tractable' only if it can be solved by a deterministic polynomially time-bounded Turing machine [1,12,25,29,30,37]. The resulting importance of the class P was first observed by A. Cobham [7] and J. Edmonds [10].

By the well-known *Savitch theorem* [33] that $NDSPACE(\log n) \subset DSPACE([\log n]^2)$, the classes of languages accepted by polynomially space-bounded DTMs and NDTMs are equal and the classes of languages accepted by exponentially space-bounded DTMs and NDTMs are equal. This is the reason for defining the complexity classes *PSPACE* and *EXSPACE*, rather than the classes *DPSPACE*, *NDPSPACE*, *DEXSPACE*, and *NDEXSPACE*.

Efficient Reducibility and 'Hard' Problems

Throughout this section F is a class of languages. Following [1,12,17], we define polynomial reducibility, F-hardness and F-completeness under such reducibilities. To do this, we must extend the definition of DTMs given above so that the resulting machines have outputs, and thus can be viewed as computing partial functions of their inputs. This is accomplished by augmenting each DTM M with an additional 'output' tape o_M such that the tape head of o_M can only move one cell to the right or stay stationary during any move of M. In addition to all usual constraints on M, for all inputs w of M:

1) initially, during a partial computation of M on w, all cells of o_M are blank and the tape head of o_M scans its leftmost cell;
2) the value computed of M on w is the final nonblank contents of o_M during the accepting computation or the nonaccepting computation of M on w, if such a computation exists, and is undefined otherwise.

The time complexity T_M of such augmented DTMs (henceforth, referred to simply as DTMs) is defined exactly as in Definition 3.

Definition 8 Let Σ and Δ be finite alphabets.

A function f from Σ^* to Δ^* is said to be *polynomial time computable* if and only if there exists a polynomially time-bounded DTM M with input alphabet Σ such that, for all $w \in \Sigma^*$, the value computed by M on input w is $f(w)$.

Let $L \subset \Sigma^*$ and $M \subset \Delta^*$. L is said to be *polynomially reducible* to M, denoted $L \leq_p M$, if and only if there is an $f : \Sigma^* \to \Delta^*$ such that f is polynomial time computable and, for all $w \in \Sigma^* : w \in L$ if and only if $f(w) \in M$.

A language L is said to be *F-hard* if and only if for all languages $L' \in F$, $L' \leq_p L$. L is said to be *F-complete* if and only if L is both F-hard and is in F.

Henceforth, let F be any of the complexity classes *NP*, *CoNP*, *PSPACE*, *DEXPTIME*, *NDEXPTIME*, *CoNDEXPTIME*, or *EXSPACE*. The following two propositions underlie most of the work on F-hard and F-complete problems in the literature on algorithmic analysis and computational complexity.

Proposition 9

1) *Let Σ, Δ, and Π be finite alphabets. Let $L \subset \Sigma^*$, $M \subset \Delta^*$, and $N \subset \Pi^*$. If $L \leq_p M$ and $M \leq_p N$, then $L \leq_p N$. (Thus, polynomial reducibility is transitive.)*
2) *If L and M are languages such that $L \leq_p M$ and L is F-hard, then M is also F-hard.*
3) *$P = NP$ if and only if some NP-complete language is in P.*
4) *$P = PSPACE$ if and only if some PSPACE-complete language is in P.*
5) *$NP = PSPACE$ if and only ifsome PSPACE-complete language is in NP.*
6) *$NP = CoNP$ if and only if some CoNP-complete language is in NP if and only if some NP-complete language is in $CoNP$.*
7) *If L is DEXPTIME-, NDEXPTIME-, or EXSPACE-hard, then the recognition of L requires more than 2^{n^ϵ} time, 2^{n^ϵ} time, respectively 2^{n^ϵ} space, i. o. on any DTM, NDTM, respectively TM, where $\epsilon > 0$ is a constant independent of n.*

Proof We sketch a proof: 1) follows from the fact that the polynomial time computable functions are closed under composition. The proofs of 2) through 6) follow directly from the correctness of 1) and Definitions 6 and 8. The proof of 7) follows directly from well-known 'hierarchy' theorems, for deterministic and nondeterministic time and space-bounded Turing machines [1,12,14,15,17,34].

Proposition 10 *There exists an F-complete language, for each of the complexity classes F.*

Proof It is easy to construct F-complete languages defined in terms of Turing machines and coded versions

of their inputs, for each of the complexity classes F. For example, the language $L = \{\# \cdot M_i \cdot \# \cdot \text{code}(x_1, \ldots, x_n) \cdot \#^m : x_1 \cdot x_n$ is accepted by the one-tape NDTM M_i in time $t\}$, where $m = 3 \cdot |M_i| \cdot t$, is both an element of $NDTIME(n)$ and is NP-hard.

The first complete problems for NP and, consequently, the importance of the class NP in nonnumerical computation are due to S.A. Cook [8], and R.M. Karp [18]. Following the terminology of [12], these initial NP-complete problems include the problems

- 3-SAT,
- 3-DIMENSIONAL MATCHING,
- VERTEX COVER, CLIQUE,
- HAMILTONIAN CIRCUIT, and
- PARTITION.

The first complete problems for $PSPACE$ and $NDEXPTIME$, are due to A.R. Meyer and L.J. Stockmeyer [24]. Subsequently, a very large number of natural computational problems have been shown to be F-hard or F-complete, for each of the complexity classes F. Many examples and historical references can be found in [1,12,16,25,29,30,37]. References [12,25,28,30,37], are especially relevant to problems in the area of optimization, including:

- LP (which is solvable deterministically in polynomial time as show initially in [19]);
- {0, 1}-ILP and ILP (the feasibility problems of which are NP-complete [5,12,17,30]); and
- QP (which is NP-complete) [12,32,37].

Reference [12] also discusses much of the early work (prior to 1979) on the complexity of approximating NP-hard optimization problems. Reference [16] consists of several separately authored chapters surveying many of the more recent results on the complexity of approximating NP-hard optimization problems. These results include the important result of [3] that unless $P = NP$, no MAX SNP-hard optimization problem [31] has a PTAS (i. e. a polynomial time approximation scheme, [12]).

Many basic polynomial time solvable and NP-hard optimization problems become $PSPACE$-hard, $DEXPTIME$-hard, $NDEXPTIME$-hard, and even $EXSPACE$-hard, when problem instances are specified succinctly by hierarchical specifications or by 1- and 2-dimensional periodic specifications, see [20,21,22,26,27,29]. References [2,6] discuss algorithms for and the computational complexity of solving systems of multivariable polynomial equations over real closed fields. Most of these last problems are NP-hard or worse. Finally, the problems of determining the solvability of a system of multivariable polynomial equations over $\mathbf{N}$ or $\mathbf{Z}$ are recursively undecidable, by a straightforward effective reduction from *Hilbert's tenth problem* [9,23].

See also

- ▶ Complexity of Degeneracy
- ▶ Complexity of Gradients, Jacobians, and Hessians
- ▶ Complexity Theory
- ▶ Complexity Theory: Quadratic Programming
- ▶ Computational Complexity Theory
- ▶ Fractional Combinatorial Optimization
- ▶ Information-based Complexity and Information-based Optimization
- ▶ Kolmogorov Complexity
- ▶ Mixed Integer Nonlinear Programming
- ▶ NP-complete Problems and Proof Methodology
- ▶ Parallel Computing: Complexity Classes

References

1. Aho AV, Hopcroft JE, Ullman JD (1974) The design and analysis of computer algorithms. Addison-Wesley, Reading, MA
2. Arnon DS, Buchberger B (1988) Algorithms in real algebraic geometry. Acad. Press, New York
3. Arora S, Lund C, Motwani R, Sudan M, Szegedy M (1992) Proof verification and hardness of approximation problems. Proc 33rd IEEE Symp Foundations of Computer Sci., pp 14–23
4. Book RV (1974) Comparing complexity classes. JCCS 9:213–229
5. Borosh I, Treybig LB (1976) Bounds on positive integral solutions of linear Diophantine equations. In: Proc Amer Math Soc, vol 55, pp 294–304
6. Canny J (1988) Some algebraic and geometric computations in PSPACE. Proc. 20th Annual ACM Symposium on Theory of Computing, 460–467
7. Cobham A (1964) The intrinsic computational difficulty of functions. In: Bar-Hillel Y (ed) Proc. 1964 Internat. Congress for Logic, Methodology and Philosophy of Sci. North-Holland, Amsterdam, 24–30
8. Cook SA (1971) The complexity of theorem-proving procedures, Proc. Third ACM Symp. Theory of Computing, pp 151–158
9. Davis M (1958) Computability and unsolvability. McGraw-Hill, New York

10. Edmonds J (1965) Paths, trees, and flowers. Canad J Math 17:449–467
11. Elgot CC, Robinson A (1964) Random access stored program machines. J ACM 11:365–399
12. Garey MR, Johnson DS (1979) Computers and intractability. A guide to the theory of NP-completeness. Freeman, New York
13. Hartmanis J, Hunt III HB (1974) The LBA problem and its importance in the theory of computing. Complexity of Computation 1–26
14. Hartmanis J, Lewis PM, Stearns RE (1965) Classification of computations by time and memory requirements. Proc. IFIP Congress, 31–35
15. Hartmanis J, Stearns RE (1965) On the computational complexity of algorithms. Trans Amer Math Soc 117:285–306
16. Hochbaum DS (1979) Approximation algorithm for NP-hard problems. PWS, Boston, MA
17. Hopcroft JE, Ullman JD (1969) Formal languages and their relation to automata. Addison-Wesley, Reading, MA
18. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations. Plenum, New York, pp 85–103
19. Khachian LG (1979) A polynomial algorithm for linear IV programming. Soviet Math Dokl 20, 191–194. (Dokl Akad Nauk USSR 224 (1979), 1093–1096)
20. Lengauer T, Wagner KW (1992) The correlation between the complexities of non-hierarchical and hierarchical versions of graph problems. JCSS 44:63–93
21. Marathe MV, Hunt III HB, Rosenkrantz DJ, Stearns RE (1998) Theory of periodically specified problems: complexity and approximability. Proc. 13th IEEE Conf. Computational Complexity
22. Marathe MV, Hunt III HB, Stearns RE, Radhakrishnan V (1997) Complexity of hierarchically and 1-dimensional periodically specified problems I: hardness results. In: Du D-Z, Gu J, Pardalos PM (eds), Satisfiability Problem Theory and Appl. DIMACS 35. Amer. Math. Soc., pp 225–259
23. Matijasevic YV (1970) Enumerable sets are Diophantine, Soviet Math Dokl 11:354–357 (Dokl Akad Nauk USSR 191 (1970), 279–282)
24. Meyer AR, Stockmeyer LJ (1972) The equivalence problem for regular expressions with squaring requires exponential times. Proc. 13th Annual Symposium on Switching and Automata Theory, pp 125–129
25. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Interscience Ser Discrete Math and Optim. Wiley, New York
26. Orlin JB (1984) Some problems on dynamic/periodic graphs. Program. Combin. Optim. Acad. Press, New York, pp 273–293
27. Orlin JB (1985 1978) The complexity of dynamic/periodic languages and optimization problems. Sloan WP, vol 1676-86. MIT, Cambridge, MA. A preliminary version of this paper appears in Proc. 13th annual ACM Symposium on Theory of Computing, 1978, pp 218–227
28. Padalos PM (1993) Complexity in numerical optimization. World Sci., Singapore
29. Papadimitriou CH (1994) Computational complexity. Addison-Wesley, Reading, MA
30. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: Algorithms and complexity. Prentice-Hall, Englewood Cliffs, NJ
31. Papadimitriou CH, Yannakakis M (1991) Optimization, approximation, and complexity classes. JCCS 43:425–440
32. Sahni S (1974) Computationally related problems. SIAM J Comput 3:262–279
33. Savitch WJ (1970) Relationship between nondeterministic and deterministic tape complexities. JCSS 4:177–192
34. Seiferas JJ, Fischer MJ, Meyer AR (1973) Refinements of nondeterministic time and space hierarchies, Proc. 14th Annual IEEE Symp. Switching and Automata Theory, pp 130–137
35. Sheperdson JC, Sturgis HE (1963) Computability of recursive functions. JACM 10:217–255
36. Turing AM (1936/7) On computable numbers, with an application to the Entscheidungsproblem. Proc London Math Soc (2) 49:230–265
37. Vavasis SA (1991) Nonlinear optimization: Complexity issues. Oxford Sci. Publ., Oxford

Complexity of Degeneracy

KATTA G. MURTY
Department IOE, University Michigan,
Ann Arbor, USA

MSC2000: 90C60

Article Outline

Keywords
Degeneracy in Linear Programming
  Degeneracy in Standard Form Systems
  The Complexity of Checking Whether a System of Constraints is Degenerate
  Problems Posed By Degeneracy for the Simplex Method of LP
  Cycling
  Stalling
  Degeneracy Handling in Commercial Codes
  Effect of Degeneracy on the Optimum Face of an LP
  Effects of Degeneracy on Post-optimality Analysis in LP
  Effects of Degeneracy in Interior Point Methods for LP
  Effect of Degeneracy on Algorithms for Enumerating Extreme Point Solutions
  Effect of Degeneracy in Extreme Point Ranking Methods

Degeneracy in Nonlinear Programming
See also
References

Keywords

Degeneracy; Nondegeneracy; Near degeneracy; Active constraints; Tight constraints; Inactive constraints; Slack constraints; Basic feasible solution; Regular polyhedron; Simple polyhedron; *NP*-complete problem; Cycling; Stalling; Resolving degeneracy; Positive marginal values; Negative marginal values; Extreme point enumeration; Extreme point ranking; Assignment ranking; Segments of polyhedra; Active set methods

Degeneracy in Linear Programming

In mathematical programming, the terms *degeneracy*, and its absence, *nondegeneracy*, have arisen first in the simplex method of linear programming (LP), where they have been given precise definitions. The notions were first introduced by G.B. Dantzig in his seminal paper [7] when he invoked the nondegeneracy assumption to prove the finite convergence of the simplex method.

In the study of the simplex method for LP, degeneracy and nondegeneracy are properties defined for basic feasible (or *extreme point*) solutions of systems of linear constraints or for the systems themselves. To give the definition, let us consider the general system of linear constraints

$$A_i.x \begin{cases} = b_i, & i = 1, \ldots, r, \\ \geq b_i, & i = r+1, \ldots, m, \end{cases} \tag{1}$$

where $A_{i\cdot} \in \mathbf{R}^n$ is the row vector of coefficients in the ith constraint, and we assume that the equality constraints in it are linearly independent. Let K denote its set of feasible solutions.

Given $\overline{x} \in K$, the ith constraint in (1) is said to be: *active* or *tight* at $\overline{x}$ if either it is an equality constraint (i. e., $i \in \{1, \ldots, r\}$), or it is an inequality constraint that holds as an equation at $\overline{x}$; *inactive* or *slack* at $\overline{x}$ otherwise. We denote the index set of active constraints at $\overline{x}$, i. e., $\{i\colon\ i \in \{1, \ldots, m\}, A_i.\overline{x} = b_i\}$ by $I(\overline{x})$.

The feasible solution $\overline{x}$ for (1) is said to be an *extreme feasible solution* or a *BFS* (*basic feasible solution*) if it is the unique solution of the system of equations defined by the active constraints at it in (1); i. e., $A_i.\overline{x} = b_i$ for $i \in I(\overline{x})$.

The BFS $\overline{x}$ for (1) is said to be: a *nondegenerate BFS* if the set of active constraints at it, treated as equations, forms a square nonsingular system of equations; a *degenerate BFS* if that system has one or more redundant equations, i. e., if $|I(\overline{x})| > n$. Thus at a degenerate BFS, this system of equations formed from the active constraints is an overdetermined system of linear equations with a unique solution.

The general system (1) is said to be: a *degenerate system* if it has at least one degenerate BFS; *nondegenerate system* if all its BFSs are nondegenerate.

Degeneracy in Standard Form Systems

Before solving an LP, the simplex method transforms the constraints into a *standard form* which is

$$\begin{aligned} Ax &= b, \\ x &\geq 0, \end{aligned} \tag{2}$$

where the matrix A is of order $m \times n$, and without any loss of generality we assume that rank$(A) = m$. Let $A_{\cdot j}$ denote the jth column vector of A for $j = 1, \ldots,$ n, it is the column of x_j in (2) and we assume it is $\neq 0$ for all j. Let Γ denote the set of feasible solutions of (2).

Specializing the above definitions to the standard form, we conclude that a feasible solution $\overline{x}$ of (2) is a BFS if and only if $\{A_{\cdot j}\colon\ j$ such that $\overline{x}_j > 0\}$ is linearly independent. The BFS $\overline{x}$ is: degenerate if $|\{j\colon\ \overline{x}_j > 0\}| < m$, nondegenerate if $|\{j\colon\ \overline{x}_j > 0\}| = m$.

So, for a nondegenerate BFS $\overline{x}$, the submatrix with column vectors $\{A_{\cdot j}\colon\ j$ such that $\overline{x}_j > 0\}$ is a basis for the system of equations in (2). If $\overline{x}$ is a degenerate BFS, this submatrix has to be augmented with the columns of some variables having 0 values in $\overline{x}$ in order to become a basis; usually this augmentation can be carried out in many ways. Hence, while each nondegenerate BFS is associated with a unique basis, each degenerate BFS is associated with several (usually a huge number of) bases.

System (2) is said to be: degenerate if it has at least one degenerate BFS; nondegenerate otherwise. From this we see that if system (2) is degenerate, then the right-hand side constants vector b lies in a subspace that

is the linear hull of a set of $m-1$ or less column vectors of the coefficient matrix A. These observations imply the following facts.

1) Keeping the coefficient matrix A fixed in (2), but letting the right-hand side constants vector b vary over $\mathbf{R}^m$, the set of all b for which (2) is degenerate is a set of Lebesgue measure zero in $\mathbf{R}^m$.
2) If (2) is degenerate, the right-hand side constants vector b in it can be perturbed ever so slightly to make the *perturbed system* nondegenerate.
   The *perturbation technique* for resolving the problem of cycling in the simplex method caused by degeneracy is based on this fact. We will discuss more on this later.
3) If (2) has at least one nondegenerate BFS, the dimension of Γ is $d = n - m$.
   Whether (2) is degenerate or not, every nondegenerate BFS of (2) is incident to exactly $d = n - m$ edges of Γ.
   However, a degenerate BFS is usually incident to more than d (could be very large) edges of Γ, and the number of edges of Γ incident at different degenerate extreme points of Γ may be very different.
4) If (2) is nondegenerate, Γ is said to be a *regular* or *simple polyhedron* because every one of its vertices is incident to exactly d (the dimension of Γ) edges. This nice regular property may not hold for Γ if (2) is degenerate. Thus, degeneracy has the effect of making the polyhedron more complex geometrically.

The Complexity of Checking Whether a System of Constraints is Degenerate

Despite the rarity of degeneracy among all possible LP models, surprisingly, many real world LP models turn out to be degenerate. However, R. Chandrasekaran, S.N. Kabadi and K.G. Murty [4] showed that the problem of checking whether a given system of linear constraints is degenerate is *NP-complete*.

A nondegenerate BFS of (2) is said to be *nearly degenerate* if some variables have positive values which are very close to zero in it. In practice, while computing BFSs of (2), unless exact arithmetic is used, it is very hard to distinguish between degenerate and nearly degenerate BFSs because of *round-off errors* introduced in digital computation.

Problems Posed By Degeneracy for the Simplex Method of LP

Cycling

Very soon after developing the simplex method for LP, Dantzig realized that it may not lead to an optimum solution under degeneracy but instead may cycle indefinitely among a set of nonoptimal degenerate bases. The first example of *cycling* in the simplex method was constructed by A.J. Hoffman [13].

For implementing the simplex method, the user has to select two *tie breaking rules* to be used in each pivot step, one for selecting the entering nonbasic variable, and the other for selecting the dropping basic variable, among all those that tie. For cycling to occur, these tie breaking rules are very crucial.

Any technique that makes sure that the simplex method cannot cycle under degeneracy is said to *resolve degeneracy*. Quite early in the development of LP, techniques for resolving degeneracy in theory, using a virtual perturbation involving powers of an infinitesimal indeterminate, without altering the data were developed ([5,8,21]; also see [16] for an extension of this technique to the bounded variable simplex method). These fix the tie breaker for the dropping variable as one based on lexicographic ordering, but leave the entering variable choice arbitrary among those eligible. Over the years several other techniques have been developed for resolving degeneracy in theory; some, e. g., *Bland's technique* [3], fix the tie breakers for both the entering and dropping variables. Bland's technique and others like it, however, lead to implementations which are very slow in practice.

Computationally, it is also important that techniques for resolving degeneracy pay attention to the possible effects of round-off errors in near degenerate solutions. See [10].

It is commonly believed that the problem of cycling is not encountered in practice; however, degeneracy related problems have been discovered to contribute substantially to the difficulty in using LP based methods in scheduling and related combinatorial and integer programming problems.

Stalling

Even after resolving the problem of cycling, yet another phenomenon called *stalling* at a degenerate BFS can oc-

cur in the simplex method. Unlike cycling which is an infinite repetition of the same sequence of degenerate bases, stalling is a finite but exponentially long sequence of consecutive degenerate pivot steps at the same objective value. Examples of stalling in network flow models have been exhibited by J. Edmonds, see [6,17].

A technique is said to resolve both cycling and stalling under degeneracy, if it can be established that the total number of consecutive degenerate pivot steps in the simplex method, using this technique, is bounded above by a polynomial function of n and the size of the LP. Such techniques that fix the tie breakers for both the entering and the dropping variables have been developed in [6] for the special case of minimum cost pure network flow problems. Extending this work to the general LP model seems to be hard, as it can be shown that resolving both cycling and stalling in a general LP model is only possible if there exist tie breakers for both the entering and dropping variables which guarantee to make the simplex method a polynomial time method for LP. To establish whether such tie breakers exist has been a long standing open problem in LP theory.

Degeneracy Handling in Commercial Codes

In spite of the folklore that cycling is very unlikely to occur in practice, commercial LP codes have sought to implement *anti-cycling procedures* that involve little overhead and are effective in practice.

The lexicographic technique for resolving degeneracy is not very desirable, as it needs the explicit basis inverse in every step (most commercial codes do not compute the basis inverse explicitly, they use matrix factorizations of the basis inverse for preserving sparsity and for numerical stability).

For handling degeneracy, commercial codes normally use procedures based on perturbing the bounds on the variables. If there is no progress in the objective value after some number of iterations dependent on problem size, then the bounds on the variables in the present basic vector are enlarged (i. e., if the previous lower and upper bounds on x_j are ℓ_j and u_j, they are changed to $\ell_j - \delta_j$ and $u_j + \delta_j$, where δ_j is a small positive quantity chosen appropriately), and the application of the algorithm is continued on the perturbed problem. When the perturbed problem reaches optimality, the bounds are reset to their original values to see if the resulting basis is optimal to the original problem (this happens very often). Otherwise, the resulting basis satisfies the optimality criterion but may be infeasible. Then a Phase I procedure is used to get feasibility, this works fine in almost all cases since the optimal basis for the perturbed problem is close to one for the original problem. The dual simplex algorithm can also be used for this later part. See [11] for details.

Effect of Degeneracy on the Optimum Face of an LP

From LP theory we know that if an LP has at least one optimum nondegenerate BFS, then the dual problem has a unique optimum solution. Conversely, if the dual problem has at least one optimum nondegenerate BFS, then the primal LP optimum solution is unique.

Effects of Degeneracy on Post-optimality Analysis in LP

After having found an optimum solution for an LP, an integral part of a good report generator is *marginal analysis*.

Consider the LP in standard form: minimize $z = cx$ subject to (2), and let $z^*(b)$ denote the optimum objective value in this LP as a function of the right-hand side constants vector b while all the other data remains fixed. The *marginal value* or *shadow price* vector for this LP is defined to be $(\partial z(b)/\partial b_i : i = 1, \ldots, m)$ when it exists.

If the LP has a nondegenerate optimum BFS, then the dual optimum solution is unique, it is the vector of marginal values; and for each right hand side constant b_i there is an interval of positive length containing the present value of b_i in its interior, which is its optimality range. As b_i varies in this range while all the other data remains fixed, the dual optimum solution remains unchanged and remains as the marginal value vector.

In practical applications, the right-hand side constants vector b in the LP model for a company's operations usually contains parameters such as the limits on raw material supplies, etc. When it exists, practitioners use the marginal value vector to derive many facts of great use in planning, such as identifying which raw material supplies are critical, what the break even price is for additional supply of each raw material, etc. Marginal analysis is the process of drawing such conclusions, and practitioners rely on it heavily to provide valuable planning information.

The situation changes dramatically when all the optimum BFSs of the LP are degenerate. The dual optimum solution may not be unique, and the marginal value vector as defined above may not exist. In its place we have two-sided marginal values: a *positive* (and a *negative*) *marginal value* giving the rate of change in the optimum objective value per unit increase (decrease) in b_i. M. Akgul [1] proved the existence of these two-sided marginal values using convex analysis. Simple proofs based on parametric LP are given in [16] where it is shown that the positive (negative) marginal value with respect to b_i is

$$\max\{\pi_i : \text{ over the dual optimum face}\}$$
$$(\min\{\pi_i : \text{ over the dual optimum face}\}).$$

Effects of Degeneracy in Interior Point Methods for LP

Unlike the simplex method which walks along edges of the polyhedron, the paths traced by interior point methods (IPMs) are contained in the strict interior of the polyhedron. There are many different classes of IPMs based on the strategy used. At first glance, degeneracy, a concept based on properties of extreme point solutions, does not seem to be as serious a problem for IPMs as it is for simplex methods. In fact proofs of polynomiality for IPMs of the projective, path following, and affine potential reduction categories hold true without any nondegeneracy assumption.

However, degeneracy affects the convergence of the primal-dual pair in the affine scaling method. Under primal nondegeneracy, this method has been shown to be globally convergent for any steplength as long as all the iterates remain in the interior of the feasible region. But this technique breaks down when the primal nondegeneracy assumption is removed, in fact L.H. Hall and R.J. Vanderbei [12] constructed a degenerate example to show that the dual sequence cannot be convergent anymore if any fixed steplength greater than 2/3 to the boundary is taken. Thus stepsize 2/3 to the boundary is the longest stepsize for the affine scaling algorithm that guarantees convergence of the primal-dual pair in the presence of degeneracy.

Although other IPMs go through the interior of the feasible region, degeneracy still has a role to play in them. But the problems here are different from the cycling and stalling problems occurring in the simplex method. Degeneracy and redundant constraints affect the central path which most IPMs aim to follow. Numerical performance of the algorithms may suffer from numerical instability and ill-conditioning if the optimum solutions are degenerate or near degenerate. Also, generating an optimum basis from the near optimum interior solution at the termination of the IPM is strongly polynomial, but the computational effort depends on the degree of degeneracy.

Effect of Degeneracy on Algorithms for Enumerating Extreme Point Solutions

Consider the problem of *enumerating all the extreme point solutions* of a system of linear constraints, say (2). Let ℓ_0 denote the unknown number of extreme point solutions.

If (2) is nondegenerate, all its extreme point solutions can be enumerated in time $O(\ell_0 mn)$, an effort which grows linearly with ℓ_0, D. Avis and K. Fukuda [2]. If (2) is degenerate and is the system of constraints for a network linear program, J.S. Provan [19] has an algorithm for enumerating all its extreme point solutions in time polynomial in ℓ_0 and the input size.

However, it remains an open question whether there is an algorithm for enumerating all extreme point solutions in time polynomial in ℓ_0 and input size, when (2) is a general degenerate system of constraints.

Murty and S.-J. Chung [18] have shown that degenerate polyhedra have proper subsets called *segments* satisfying certain facial incidence properties. For each nondegenerate polyhedron, the only segment possible is the whole polyhedron itself. The difficulty of enumerating extreme point solutions of degenerate systems efficiently is related to the problem of recognizing whether a given segment is the whole polyhedron or a proper subset of it.

Effect of Degeneracy in Extreme Point Ranking Methods

Consider the objective function $z(x) = cx$ defined over Γ, the set of feasible solutions of (2). For simplicity assume that Γ is a convex polytope, i. e., it is bounded. An algorithm for *ranking the extreme points* of Γ in increasing order of $z(x)$ has been discussed in [15]. In each step, this algorithm carries out the operation of

enumerating the adjacent extreme points of a given extreme point of Γ.

If (2) is nondegenerate, every extreme point has exactly $n - m$ adjacent extreme points, and the above operation can be carried out efficiently by pivot steps. Hence, the complexity of generating k extreme points in the ranked sequence grows linearly with k, and the ranking algorithm becomes practically effective.

If (2) is degenerate, the number of adjacent extreme points of a degenerate extreme point of Γ may be very large, and the ranking algorithm becomes almost impractical.

The *assignment problem* is a well known example of a highly degenerate problem. However all its extreme point solutions known as assignments are $0 - 1$ vectors, using this property an efficient special algorithm has been developed in [14] for ranking the assignments in increasing order of a linear objective function.

Degeneracy in Nonlinear Programming

In contrast to linear programming where the concept of degeneracy is defined purely using extreme point solutions; in nonlinear programming it is defined for any solution point.

Discussion of degeneracy arises in nonlinear programming, particularly in methods known as *active set methods*. These methods are popular for solving nonlinear programs in which the constraints are linear, say of the form (1); but also used when there are nonlinear constraints in the system. In these methods, when at a feasible point x^0, certain constraints indexed by an active set $\mathcal{A}$ are treated as equations, and the rest are temporarily disregarded, and a search direction y^0 is generated. The next point is taken ideally as the best feasible point on the half-line $\{x^0 + \lambda y^0 : \lambda \geq 0\}$. However, if one or more inequality constraints not from the set $\mathcal{A}$ are violated by $x^0 + \lambda y^0$ whenever $\lambda > 0$ and sufficiently small, then those constraints allow no progress in the search direction, and we have a degenerate situation. See [10] and [11] for a discussion of this degeneracy in active set methods, and its resolution.

There are also other generalizations of the notions of degeneracy and nondegeneracy, to systems of nonlinear constraints. In these generalizations, degeneracy is taken to mean any measure of departure of problem structure from some idealized norm. Simply put, nondegenerate means well-posed in some context, degenerate means absence of such nice structure. For a nonlinear program, nondegeneracy at a solution point has been defined variously as the satisfaction of: LICQ (linear independence constraint qualification of the binding constraint gradients), KKT first order necessary conditions for a local minimum, second order sufficient conditions for a local minimum, or the strict complementary slackness condition. Also connections between nondegeneracy and performance of algorithms has been studied, addressing the local effects of special kinds of nondegeneracy or its lack at a local minimizer. See [9].

See also

- ▶ Complexity Classes in Optimization
- ▶ Complexity of Gradients, Jacobians, and Hessians
- ▶ Complexity Theory
- ▶ Complexity Theory: Quadratic Programming
- ▶ Computational Complexity Theory
- ▶ Fractional Combinatorial Optimization
- ▶ Information-Based Complexity and Information-Based Optimization
- ▶ Kolmogorov Complexity
- ▶ Mixed Integer Nonlinear Programming
- ▶ NP-Complete Problems and Proof Methodology
- ▶ Parallel Computing: Complexity Classes

References

1. Akgul M (1984) A note on shadow prices in linear programming. J Oper Res Soc 35:425–431
2. Avis D, Fukuda K (1992) A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. Discrete Comput Geom 8:295–313
3. Bland RG (1977) New finite pivoting rules for the simplex method. Math Oper Res 2:103–107
4. Chandrasekaran R, Kabadi SN, Murty KG (1982) Some NP-complete problems in linear programming. Oper Res Lett 1:101–104
5. Charnes A (1952) Optimality and degeneracy in linear programming. Econometrica 20:160–170
6. Cunningham WH (1979) Theoretical properties of the network simplex method. Math Oper Res 4:196–208
7. Dantzig GB (1951) Maximization of a linear function of variables subject to linear inequalities. In: Koopmans TC (ed) Activity Analysis of Production and Allocation. Wiley, New York, pp 339–347

8. Dantzig GB, Orden A, Wolfe P (1955) The generalized simplex method for minimizing a linear form under linear inequality restraints. Pacific J Math 5:183–195
9. Fiacco AV, Liu J (1993) Degeneracy in nonlinear programming and the development of results motivated by its presence. In: Gal T (ed) Degeneracy in optimization problems. vol 46 of Ann Oper Res, pp 61–80
10. Fletcher R (1987) Practical methods of optimization, 2nd edn. Wiley, New York
11. Gill PE, Murray W, Saunders MA, Wright MH (1989) A practical anti-cycling procedure for linearly constrained optimization. Math Program 45:437–474
12. Hall LA, Vanderbei RJ (1993) Two-thirds is sharp for affine scaling. Oper Res Lett 13:197–201
13. Hoffman AJ (1953) Cycling in the simplex algorithm. NBS Report 2974 Washington D.C.
14. Murty KG (1968) An algorithm for ranking all the assignments in increasing order of cost. Oper Res 16:682–687
15. Murty KG (1968) Solving the fixed charge problem by ranking the extreme points. Oper Res 16:268–279
16. Murty KG (1983) Linear programming. Wiley, New York
17. Murty KG (1992) Network programming. Prentice-Hall, Englewood Cliffs, NJ
18. Murty KG, Chung S-J (1995) Segments in enumerating faces. Math Program 70:27–45
19. Provan JS (1994) Efficient enumeration of the vertices of polyhedra associated with network LPs. Math Program 63:47–64
20. Gal T (ed) (1993) Degeneracy in optimization problems. Ann Oper Res, vol 46
21. Wolfe P (1963) A technique for resolving degeneracy in linear programming. SIAM J 11:205–211

Complexity of Gradients, Jacobians, and Hessians

SA

Andreas Griewank
Institute Sci. Comput. Department Math.,
Techn. University Dresden, Dresden, Germany

MSC2000: 65D25, 68W30

Article Outline

Keywords
'Numerical' Differentiation Methods
'Analytical' Differentiation Methods
Two-Stranded Chain Scenario
Computational Model
Indefinite Integral Scenario
Lack of Smoothness
Predictability of Complexities
Goal-Oriented Differentiation
The Computational Graph
Forward Mode
Bauer's Formula
Reverse Mode
Second Order Adjoints
Operations Counts and Overheads
Worst-Case Optimality
Expensive ≡ Redundant?
Preaccumulation and Combinatorics
Summary
See also
References

Keywords

Automatic differentiation; Divided differences; Computer algebra; Symbolic manipulation

The evaluation or approximation of derivatives is a central part of most nonlinear optimization calculations. The gradients of objectives and active constraints enter directly into they Karush–Kuhn–Tucker conditions so that inaccuracies in their evaluation limit the achievable solution accuracy. The latter depends also crucially on the conditioning of the projected Hessian of the Lagrangian. Hence accurate values of this symmetric matrix allow the design of appropriate stopping criteria including the verification of second order conditions. Second derivatives also facilitate a rapid final rate of convergence, provided the step-defining linear systems can be solved by factorization or iteration at a reasonable cost. The same observations apply to more general optimization calculations like the solution of nonlinear complementarity problems.

Whether or not the obvious benefits of evaluating first and higher derivatives accurately justify the costs incurred, does strongly depend on the suitability of the differentiation method employed for the particular problem at hand. We may distinguish five principal options for evaluating or approximating derivatives

- *symbolic differentiation*;
- *handcoded derivatives*;
- *automatic differentiation*;

- *difference quotients*;
- *secant updating*.

'Numerical' Differentiation Methods

The last two options are widely used in practical optimization, primarily because they require no extra effort whatsoever on the part of the user. Difference quotients are often called *divided differences* or *finite differences*, though the last term invites confusion with a related method for discretizing differential equations. Other popular labels are *differencing* or *numerical differentiation*, because the results are floating point numbers rather than algebraic expressions. The latter are often presumed to be the output of more symbolic methods. Even though we shall see that the distinction is not quite that easy, there is no doubting the importance of the fundamental relation

$$F'(x)\dot{x} = \frac{d}{d\alpha} F(x + \alpha\dot{x})\Big|_{\alpha=0} = \frac{1}{\varepsilon}\big[F(x + \varepsilon\dot{x}) - F(x)\big] + O(\varepsilon).$$

Here, the vector function F: $\mathbf{R}^n \to \mathbf{R}^m$ is assumed Lipschitz-continuously differentiable on some neighborhood of the base point $x \in \mathbf{R}^n$. In other words, the directional derivative of F along some vector $\dot{x} \in \mathbf{R}^n$ is the product of the Jacobian matrix $F'(x) \in \mathbf{R}^{m\times n}$ with the direction $\dot{x}$ and it can be approximated by a difference quotient. The quality of this approximation depends strongly on the choice of ε and one must expect a halving in the number of significant digits under the best of circumstances. Quasi-Newton, or secant methods may be viewed as an ingenious way of sequentially incorporating difference quotients into a Jacobian approximation while iterating towards the solution vector of a nonlinear system of equations. The corresponding theory of superlinear convergence is quite beautiful from a mathematical point of view, though perhaps not terribly relevant in practice for large, structured problems.

It is important to note that the quality of the approximate derivative matrices generated by quasi-Newton methods influences only the rate of convergence but not so much the solution accuracy itself. The latter depends on the accurate evaluation of residual vectors, which may be composed of gradients as is the case for the KKT conditions. The importance of accurate residual values is particularly well understood in numerical linear algebra, and replacing them with approximations of uncertain reliability is generally a dicy proposition. Fortunately, it just so happens that gradients can usually be evaluated with working precision at a moderate cost relative to that of the underlying functions. This is far from true for Jacobians and Hessians, whose cost is very hard to predict (and even define) as we shall demonstrate further below on various examples.

The relative cost of evaluating one-sided difference quotients in p directions $\dot{x}$ from the same base point x is clearly $p + 1$. Theoretically one might sometimes reduce the evaluation costs by exploiting the fact that the p points $x + \varepsilon\dot{x}$ are close to x. This proximity may arise in the topological sense that the stepsize $\varepsilon\|\dot{x}\|$ is small as well as in the structural sense that $\dot{x}$ is sparse and thus leaves many components of x unchanged. In practice such savings are rarely realized and they would certainly destroy the main advantage of differencing, namely its black box quality, which does not require any insight or access to the process by which function values are generated. Of course, there is the optimistic assumption that they vary smoothly as a function of the argument x, and usually the selection of a suitable increment ε causes enough trouble for the user and possibly even quite a few extra trial evaluations.

Hence it is indeed fair to assume that one-sided or centered differences in p directions $\dot{x}$ at a common x require $1 + p$ or $1 + 2p$ separate function evaluations but little extra storage. By letting $\dot{x}$ range over all n Cartesian basis vectors one obtains an approximate Jacobian with first or second order accuracy at the cost of $1 + n$ or $1 + 2\,n$ function evaluations. The number of dependent variables does not matter for differencing so that the cost of a gradient, where $m = 1$, is also $1 + n$ or $1 + 2n$ times that of the underlying scalar function. To compute the Hessian or more generally a full second derivative tensor one needs $n(n + 1)/2$ function evaluations for one-sided and twice that many for the more accurate centered differences.

Since multiple function evaluations are an 'embarrassingly' parallel task the availability of several processors can be used to achieve a nearly perfect speed up for derivative approximations by differencing [7]. In the sparse case, the number of independent variables n can be replaced in the cost ratios above by a num-

ber $p \leq n$ that represents either the maximal number of nonzeros in any row of $F'(x)$ or the usually slightly larger chromatic number of the *column incidence graph*. The latter reduction can be achieved by the by now classical grouping or coloring technique originally due to Curtis–Powell–Reid [6] and further developed by Coleman–Moré [4]. An alternative way to compress the rows of the Jacobian even further at the expense of some linear equation solving is due to Newsam–Ramsdell [12] and has recently been adopted to automatic differentiation.

'Analytical' Differentiation Methods

The first three options listed at the beginning are based on the chain rule and may therefore be combined under the label *analytical differentiation*. They all would yield exact derivative values if real arithmetic could be performed in infinite precision. Moreover, even the actual sequence of operations performed to evaluate a particular partial derivative would quite likely be the same and thus yield identical results if the same floating point arithmetic was used. Only the way in which the instruction for this floating point calculation are generated and stored differ significantly between the three approaches. Also, there may be more or less recalculation of intermediates that are common to several partial derivatives, which can have drastic effects on the computational efficiency.

The result of the second option *handcoding* may in principle be always similarly obtained by symbolic or automatic differentiation, provided the computer algebra package or the differentiation software is sufficiently smart. Hence we will discuss only the pure options one and three, which might of course also be combined by a highly sophisticated programmer or software tool.

Symbolic differentiation is usually performed in *computer algebra* packages like Maple, Mathematica and Reduce. Most users have the notion that the differentiation commands in these sophisticated systems turn formulas for functions into formulas for derivatives. Moreover there is a tendency to assume that having a 'formula' means directly expressing dependent variables as algebraic expressions of independents without allowing any named intermediates. The natural data structures for such formulas would be expression trees. There, every node has only one parent, so that the whole thing can be easily linearized and printed by enumeration in a depth first order. In reality computer algebra packages do not restrict themselves to expression trees, because for any nontrivial function the corresponding tree structure is very likely to represent an incredible amount of *redundancy*, even before any differentiation takes place.

Two-Stranded Chain Scenario

Consider for example a sequence of complex function evaluations

$$x_{k+1} + iy_{k+1} = \phi_k(x_k + iy_k) + i\psi_k(x_k + iy_k)$$

for $k = 0, \ldots, l-1$ starting from some initial $x_0 + iy_0 \in \mathbf{C}$. Suppose all function pairs $\phi_k + i\,\psi_k$ are nonlinear and do not allow any algebraic simplifications. Then eliminating the intermediates x_1 and y_1 yields the formula

$$\begin{aligned} x_2 + iy_2 = {} & \phi_1(\phi_0(x_0, y_0) + i\varphi_0(x_0, y_0)) \\ & + \psi_1(\phi_0(x_0, y_0) + i\varphi_0(x_0, y_0)), \end{aligned}$$

which involves already twice as many terms as the one-level original formula. The same doubling occurs at each subsequent level so that expressing x_l and y_l directly in terms of the initial components x_0 and y_0 yields an exponentially long formula with the symbols x_0 and y_0 each occurring exactly 2^l times. In this case one could avoid the highly undesirable expression swell by merely substituting $z_k \equiv x_k + iy_k$, which turns the binary expression tree into a simple chain of the same height l.

While this example may appear rather algebraic and somewhat contrived, exactly the same effect occurs if the real pairs (x_k, y_k) specify straight lines in the plane. Specifically, one might think of light-beams being reflected in a maze of mirrors or some other optical arrangement in the plane. Each ray (x_k, y_k) that is incoming to a mirror or lense uniquely determines an outgoing ray $(x_k + 1, y_k + 1)$ via some simple algebraic relationship. Then expressing the final ray parameters (x_l, y_l) directly as functions of the initial parameters (x_0, y_0) will again yield an expression of size 2^l.

Rather than dealing with this algebraic monster one should of course keep all the intermediate pairs (x_k, y_k) with $0 \leq k \leq l$ as named variables. Along this chain

one can easily propagate all information of interest, including the 2×2 Jacobian of (x_k, y_k) with respect to (x_0, y_0), at a temporal and spatial complexity of order l. To achieve this result one may employ suitable variants of computer algebra, automatic differentiation or, of course, hand-coding. Before discussing them in more detail let us discuss a general model of function and derivative evaluations.

Computational Model

All analytical differentiation methods are based on the observation that most vector functions F of practical interest are being evaluated by a sequence of assignments

$$v_i = \varphi_i(v_j)_{j<i} \quad \text{for } i = 1, \ldots, l + m. \tag{1}$$

Here, the variables v_i are real scalars and the elemental functions φ_i are either binary arithmetic operations or univariate intrinsics. Consequently, only one or two of the partial derivatives

$$c_{ij} \equiv \frac{\partial}{\partial v_j} \varphi_i \, (v_k)_{k<i}$$

do not vanish identically and can be evaluated at a cost comparable to that of the underlying φ_i itself.

Without loss of generality we may require that the first n variables $v_{j-n} = x_j$ with $j = 1, \ldots, n$ represent the *independent variables* and the last m variables $y_i = v_l + i$ with $i = 1, \ldots, m$ represent the *dependent variables*. Then the function $y = F(x)$ is defined by the program (1). Here, the nonnegative integer l represents the number of intermediate variables, which we expect to be much larger than both n and m for seriously nonlinear problems. We will also assume that within a small constant all elemental functions have the same complexity so that we have the approximate operations count

$$\text{OPS}(x \overset{\text{prog}}{\longmapsto} y) \sim l \equiv \#\text{intermediates}.$$

Throughout this article, $\sim$ means proportional with small constants that are independent of the particular problem at hand. Each intermediate variable may be viewed and thus later differentiated as a function $v_i \equiv v_i(x)$ of the independent variable vector x. As long as all intermediates v_i are stored in separate locations the memory requirement for evaluating F will also be proportional to l. This is a very unrealistic assumption as most evaluation programs involve shared allocation of intermediates. Due to space constraints we will not be able to discuss any aspects of spatial complexity in this article. For a detailed treatment of various trade-offs between space and time see [9].

The way in which the elemental partials c_{ij} are handled differs amongst various analytical differentiation methods. They are always evaluated as floating point numbers at the current argument in what is variously known as *automatic* or *algorithmic* or *computational differentiation*. The same can be assumed for hand written derivative codes unless they are programmed within a computer algebra system, where the c_{ij} can be defined and manipulated as algebraic expressions. In some cases applying the chain rule to these expressions may theoretically lead to significant simplifications and thus potentially provide the user with analytical insight. In the following section we reverse engineer one such class of examples and arrive at the tentative conclusion that the practical potential for symbolic simplifications during the differentiation process appears to be very slim indeed.

Indefinite Integral Scenario

Suppose that

$$F(x) = \int_a^x \frac{P(\widetilde{x})}{Q(\widetilde{x})} \, d\widetilde{x}$$

for two polynomials $P(x)$ and $Q(x)$ with deg(Q)> deg(P). Besides a rational term the symbolic expression for $F(x)$ is then likely to contain a welter of logarithms and arcus tangents, whose complexity may easily exceed that of the integrand $f(x) \equiv P(x) \,/\, Q(x)$ by orders of magnitude. Then fully symbolic differentiation will of course lead back to an algebraic expression for $f(x)$, while automatic differentiation will combine the c_{ij} in floating point arithmetic according to some variant of the chain rule and obtain'just' a numerical value of $f(x)$ at the given point $x \in \mathbf{R}$. Moreover, due to cancellations that value may well be less accurate than that obtained by plugging the particular argument x into the formula for $f(x)$.

However, similar numerical instabilities are likely to already affect the evaluation of $F(x)$ itself. They may

also show up in the form of an imaginary component when the coefficients of $P(x)$ and $Q(x)$ are real but given in floating point format. Then the roots of the denominator polynomial are already perturbed by unavoidable round-off and symbolic differentiation of the resulting expression for $F(x)$ will usually not lead back to $f(x)$ but some other rational function with a higher polynomial degree in the numerator or denominator. To avoid this effect all coefficients of $f(x)$ must be specified as algebraic numbers so that the symbolic integration can be performed exactly. This process which typically involves rational numbers with enormous coefficients and thus requires a large computational effort.

Hence on practical models one may well be better advised to evaluate $F(x)$ by a numerical quadrature yielding highly accurate results at a fraction of the computing time. Analytically differentiating a nonadaptive quadrature procedure yields the same quadrature applied to the derivatives of the integrand, namely $f'(x) = F''(x)$. Hence the resulting values are quite likely to be good approximations to the original integrand $f(x)$ and they are the exact derivatives of the approximate values computed for $F(x)$ by the quadrature.

Lack of Smoothness

Adaptive quadratures on the other hand may vary grid points and coefficient values in a nondifferentiable or even discontinuous fashion. Then derivatives of the quadrature value may well not exist in the classical sense at some critical arguments x. This difficulty is likely to arise in the form of program branches in all substantial scientific codes and there is no agreement yet on how to deal with it. In most situations one can still compute one-sided directional derivatives as well as generalized gradients and Jacobians [9]. Naturally, computing difference quotients of nonsmooth functions is also a risky proposition. Generally, optimal results in terms of accuracy and efficiency can only be expected from a derivatives code developed by a knowledgeable user, possibly with the help of program analysis and transformation tools.

Predictability of Complexities

With regards to spatial and temporal complexity the following basic distinction applies between the analytical differentiation methods sketched above. The cost of fully symbolic differentiation seems impossible to predict. It can sometimes be very low due to fortuitous cancellations but it is more likely to grow drastically with the complexity of the underlying function. In contrast the relative cost incurred by the various modes of automatic differentiation can always be a priori bounded in terms of the number of independent and dependent variables. Moreover, as we will see below these bounds can sometimes be substantially undercut for certain structured problems.

Another advantage of automatic differentiation compared to a fully symbolic approach is that restrictions and projections of Jacobians and Hessians to certain subspaces of the functions domain and range can be built into the differentiation process with corresponding savings in computational complexity. In the remainder of this article we will therefore focus on the complexity of various automatic differentiation techniques; always making sure that no other known approach is superior in terms of accuracy and complexity on general vector functions defined by a sequence of elemental assignments.

Goal-Oriented Differentiation

The two-stranded chain scenario above illustrates the crucial importance of suitable representations of the mathematical objects, whose complexity we try to quantify here. So one really has to be more specific about what one means by *computing* a function, gradient, Jacobian, Hessian, or their restriction and projection to certain subspaces. At the very least we have to distinguish the (repeated) *evaluation* in floating point arithmetic at various arguments from the *preparation* of a suitable procedure for doing so. This preparation stage comes actually first and might be considered the symbolic part of the differentiation process. It usually involves no floating point operations, except possibly the propagation and simplification of some constants. This happens for example when a source code for evaluating F is precompiled into a source code for jointly evaluating $F(x)$ and its Jacobian $F'(x)$ at a given argument x. In the remainder we will neglect the preparation effort presuming that it can be amortized over many numerical evaluations as is typically the case in iterative or time-dependent computations.

In general, it is not a priori understood that $F'(x)$ should be returned as a rectangular array of floating point numbers, especially if it is sparse or otherwise structured. Its cheapest representation is the sparse triangular matrix

$$C = C(x) \equiv (c_{ij})_{i=1-n,\ldots,l+m}^{j=1-n,\ldots,l+m} .$$

The nonzero entries in C can be obtained during the evaluation of F at a given x for little extra cost in terms of arithmetic operations so that

$$\mathrm{OPS}\{x \mapsto C\} \sim \mathrm{OPS}\{x \overset{\mathrm{prog}}{\longmapsto} F\} .$$

As we will see below, the nonzeros in C allow directly the calculation of the products

$$F'(x)\dot{x} \in \mathbb{R}^m \quad \text{for } \dot{x} \in \mathbb{R}^n,$$

and

$$F'(x)^\top \bar{y}^\top \in \mathbb{R}^n \quad \text{for } \bar{y}^\top \in \mathbb{R}^m,$$

using just one multiplication and addition per $c_{ij} \not\equiv 0$. So if our goal is the iterative calculation of an approximate Newton-step using just a few matrix-vector products, we are well advised to just work with the collection of nonzero entries of C provided it can be kept in memory. If on the other hand we expect to take a large number of iterations or wish to compute a matrix factorization of the Jacobian we have to first accumulate all mn partial derivatives $\partial y_i / \partial x_j$ from the elemental partials c_{ij}. It is well understood that a subsequent inplace triangular factorization of the Jacobian $F'(x)$ yields an ideal representation if one needs to multiply itself as well as its inverse by several vectors and matrices from the left or right. Hence we have at least three possible ways in which a Jacobian can be represented and kept in storage:

- unaccumulated: computational graph;
- accumulated: rectangular array;
- factorized: two triangular arrays.

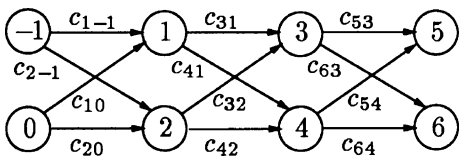
Here the arrays may be replaced by sparse matrix structures. For the time being we note that Jacobians and Hessians can be provided in various representation at various costs for various purposes. Which one is most appropriate depends strongly on the structure of the problem function $F(x)$ at hand and the final numerical purpose of evaluating derivatives in the first place. The interpretation of C as computational graph goes back to L.V. Kantorovich and requires a little more explanation.

The Computational Graph

With respect to the precedence relation

$$j \prec i \quad \Longleftrightarrow \quad c_{ij} \not\equiv 0 \quad \Longleftrightarrow \quad (j,i) \in E ,$$

the indices $i, j \in V \equiv [1-n, \ldots, 1+m]$ form a directed graph with the edge set E. Since by assumption $j \prec i$ implies $j < i$ the graph is acyclic and the transitive closure of $\prec$ defines a partial ordering between the corresponding variables v_i and v_j. The minimal and maximal elements with respect to that order are exactly the independent and dependent variables $v_{j-n} \equiv x_j$ with $j = 1, \ldots, n$ and the $v_{m+i} \equiv y_i$ with $i = 1, \ldots, m$, respectively. For the two stranded chain scenario with $l = 3$ one obtains a computational graph of the following form:



Assuming that all elemental φ_i are unary functions or binary operations we find $|E| \leq 2(l+m) \sim l$. One may always annotate the graph vertices with the elemental functions φ_i and the edges with the nonvanishing elemental partials c_{ij}. For most purposes the φ_i do not really matter and we may represent the graph (V, E) simply by the sparse matrix C.

Forward Mode

Given some vector $\dot{x} \equiv (\dot{v}_{j-n})_j = 1, \ldots, n \in \mathbf{R}^n$, there exist derivatives

$$\dot{v}_i \equiv \frac{d}{d\alpha} v_i(x + \alpha\dot{x})\Big|_{\alpha=0} \quad \text{for } 1 \leq i \leq l+m .$$

By the chain rule these $\dot{v}_i$ satisfy the recurrence

$$\dot{v}_i \equiv \sum_{j \prec i} c_{ij}\dot{v}_j \quad \text{for } i = 1, \ldots, l+m . \tag{2}$$

The resulting tangent vector $\dot{y} \equiv (\dot{v}_{l+i})_{i=1,\ldots,m}$ satisfies $\dot{y} = \mathrm{F}'(x)\dot{x}$ and it is obtained at a cost propor-

tional to l. Instead of propagating derivatives with respect to just one direction vector $\dot{x}$ one may amortize certain overheads by bundling p of them into a matrix $\dot{X} \in \mathbf{R}^{n \times p}$ and then computing simultaneously $\dot{Y} = F'(x)\dot{X} \in \mathbf{R}^{m \times p}$. The cost of this *vector forward* mode of automatic differentiation is given by

$$\mathrm{OPS}\{C \overset{\text{forw}}{\longmapsto} \dot{Y}\} \sim pl \sim p\,\mathrm{OPS}\{x \overset{\text{prog}}{\longmapsto} y\} \,. \tag{3}$$

If the columns of $\dot{X}$ are Cartesian basis vectors $e_j \in \mathbf{R}^n$ the corresponding columns of the resulting $\dot{Y}$ are the jth columns of the Jacobian. Hence by setting $\dot{X} = I$ with $p = n$ we may compute the whole Jacobian at a temporal complexity proportional to nl. Fortunately, in many applications the whole Jacobian is either not needed at all or due to its sparsity pattern it may be reconstructed from its *compression* $\dot{Y} = F'(x)\dot{X}$ for a suitable *seed matrix* $\dot{X}$. As in the case of difference quotients this matrix may be chosen according to the Curtis–Powell–Reid [6] or the Newsam–Ramsdell [12] approach with p usually close to the maximal number of nonzeros in any row of the Jacobian.

Bauer's Formula

Using the recurrence for the $\dot{v}_i$ given above one may also obtain an explicit expression for each individual partial derivative $\partial y_i / \partial x_j$. Namely, it is given by the sum over the products of all arc values c_{ij} along all paths connecting the minimal node v_{j-n} with the maximal node v_{l+i}. This formula due to F.L. Bauer [1] implies in particular that the ijth Jacobian entry vanishes identically exactly when there is no path connecting nodes $j - n$ and $l + i$ in the computational graph. In general the number of distinct paths in the graph is very large and it represents exactly the lengths of the formulas obtained if one expresses each y_i directly in terms of all x_j that it depends on. Hence we may conclude

$$\mathrm{OPS}\{C \overset{\text{bauer}}{\longmapsto} F'\} \sim \mathrm{OPS}\{x \overset{\text{formul}}{\longmapsto} y\} \,.$$

In the two-stranded chain scenario considered above, both operations counts would be of order 2^l, which is obviously an unacceptable effort. Fortunately, vector forward and Bauer's formula are just two special choices amongst many ways for accumulating the Jacobian $F'(x)$ from the computational graph C. The most celebrated alternative is the reverse or backward mode of automatic differentiation.

Reverse Mode

Rather than propagating directional derivatives $\dot{v}_i$ forward through the computational graph one may also propagate adjoint quantities $\overline{v}_i$ backward. To define them properly one must perturb the original evaluation loop by rounding errors δ_i so that now

$$v_i = \delta_i + \varphi_i(v_j)_{j<i} \quad \text{for } i = 1 - n, \ldots, l \,.$$

Then the resulting vector y is a function not only of x but also of the vector of small perturbations $(\delta_i)_{i=1-n,\ldots,l}$. Given any row vector of weights $\overline{y} = (\overline{v}_{l+i})_{i=1,\ldots,m}$ we obtain the sensitivities

$$\overline{v}_i \equiv \frac{\partial}{\partial \delta_i} \overline{y} y \bigg|_{\delta_i = 0} \qquad \text{for } 1 - n \le i \le l \,,$$

where all other perturbations δ_j with $j \neq i$ are set to zero during the differentiation. The adjoint components $\overline{v}_{j-n} = \overline{x}_j$ form the row vector $\overline{x} = \overline{y}F'(x) \in \mathbb{R}^n$, which is simply the gradient of the linear combination $\overline{y}F(x)$. In the optimization context this scalar valued function is usually a Lagrangian, whose gradient and Hessian figure prominently in the first and second order optimality conditions. The amazing thing is that as a consequence of the chain rule such gradients can be computed at the same cost as tangents by using the backward recurrence

$$\overline{v}_j = \sum_{i \succ j} \overline{v}_i c_{ij} \quad \text{for } j = l, \ldots, 1 - n \,. \tag{4}$$

Just like in the forward scalar recurrence (2), each elemental partial $c_{ij} \neq 0$ occurs exactly once and we may amortize costs by bundling several $\overline{y}$ into an adjoint seed matrix $\overline{Y} \in \mathbb{R}^{q \times m}$. This vector reverse mode yields the matrix $\overline{X} = \overline{Y}F'(x) \in \mathbb{R}^{q \times n}$ at the cost

$$\mathrm{OPS}\{C \overset{\text{rev}}{\longmapsto} \overline{X}\} \sim ql \sim q\,\mathrm{OPS}\{x \overset{\text{prog}}{\longmapsto} y\} \,.$$

Again the whole Jacobian is obtained directly if we seed $\overline{Y} = I$ with $q = m$. Hence we find by comparison with (3) as a rule of thumb that the reverse mode is preferable if $m \ll n$, i. e., if there are not nearly as many dependents as independents. In classical NLPs we may

think of m as the number of active constraints plus one, which is often much smaller then n, the number of variables. In unconstrained optimization we have $m = 1$ so that the gradient of the objective F can be computed with essentially the same effort as F itself. In the sparse case we may now employ column rather than row compression with q roughly equal to the maximal number of nonzeros in any column of $F'(x)$.

For suitable seeds $\overline{Y}$ the column compression $\overline{X} = \overline{Y}F'(x)$ allows the reconstruction of the complete Jacobian $F'(x)$. Furthermore, row and column compression can be combined yielding for example Jacobians with arrow head structure at the cost of roughly $p + q = 3$ function evaluations. In that case one may use

$$\dot{X} \equiv \begin{pmatrix} 1 & 1 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}^\top \quad \text{and} \quad \overline{Y} = e_m^\top .$$

Then $\overline{X} = \overline{Y}F'(x)$ is the last row of the arrowhead matrix $F'(x)$ and the two columns of $\dot{Y} = F'(x)\dot{X}$ contain all other nonzero entries. For pure row or column compression dense rows or columns always force $p = n$ or $q = m$, respectively. Hence the combination of forward and reverse differentiation offers the potential for great savings. In either case projections and restrictions of the Jacobian to subspaces of the vector functions domain and range can be built into the differentiation process, which is part of the goal-orientation we alluded to before.

Second Order Adjoints

Rather than separately propagating some first derivatives forward, others reverse, and then combining the results to compute Jacobian matrices efficiently, one may compose these two fundamental modes to compute second derivatives like Hessians of Lagrangians. More specifically, we obtain by directional differentiation of the adjoint relation $\overline{x} = \overline{y}F'(x)$ the second order adjoint

$$\dot{\overline{x}} = \overline{y}F''(x)\dot{x} \in \mathbb{R}^n .$$

Here we have assumed that the adjoint vector $\overline{y}$ is constant. We also have taken liberties with matrix vector notation by suggesting that the $m \times n \times n$ derivative tensor $F''(x)$ can be multiplied by the row vector $\overline{y} \in \mathbb{R}^m$ from the left and the column vector $\dot{x} \in \mathbb{R}^n x \in \mathbf{R}^n$ from the right yielding a row vector $\dot{\overline{x}}$ of dimension n. In an optimization context $\overline{y}$ should be thought of as a vector of Lagrange multipliers and $\dot{x}$ as a feasible direction. By composing the complexity bounds for the reverse and the forward mode one obtains the estimates

$$\begin{aligned} \mathrm{OPS}\{x \overset{\text{prog}}{\longmapsto} y\} &\sim \mathrm{OPS}\{x, \dot{x} \overset{\text{forw}}{\longmapsto} \dot{y}\} \\ &\sim \mathrm{OPS}\{x, \overline{y} \overset{\text{rev}}{\longmapsto} \overline{x}\} \sim \mathrm{OPS}\{x, \dot{x}, \overline{y} \overset{\text{ad}}{\longmapsto} \dot{\overline{x}}\} . \end{aligned}$$

Here, ad represents reverse differentiation followed by forward differentiation or vise versa. The former interpretation is a little easier to implement and involves only one forward and one backward sweep through the computational graph.

Operations Counts and Overheads

From a practical point of view one would of course like to know the proportionality factors in the relations above. If one counts just multiplication operations then $\dot{y}$ and $\overline{x}$ are at worst 3 times as expensive as y, and $\dot{\overline{x}}$ is at most 9 times as expensive. A nice intuitive example is the calculation of the determinant y of a $\sqrt{n} \times \sqrt{n}$ matrix whose entries form the variable vector x. Then we have $m = 1$ and

$$\mathrm{OPS}\{x \mapsto y\} = \frac{1}{3}\sqrt{n}^3 + O(n)$$

multiplications if one uses an LU factorization. Then it can be seen that $\overline{y} = 1/y$ makes $\overline{x}$ the transpose of the inverse matrix and the resulting cost estimate of $\sqrt{n}^3 + O(n)$ multiplications conforms exactly with that for the usual substitution procedure.

However, these operations count ratios are no reliable indications of actual runtimes, which depend very strongly on the computing platform, the particular problem an hand, and the characteristics of the AD tool. Implementations of the vector forward mode like ADIFOR [3] that generate compilable source codes can easily compete with divided differences, i. e. compute p directional derivatives in the form $\dot{Y} = F'(x)\dot{X}$ at the cost of about p function evaluations. For sizeable $p \approx 10$ they are usually faster than divided differences, unless the roughly p-fold increase in storage results in too much paging onto disk. The reverse mode is an entirely different ball-game since most intermediate values v_i and some control flow hints need to be first saved

and later retrieved, which can easily make the calculation of adjoints memory bound. This memory access overhead can be partially amortized in the vector reverse mode, which yields a bundle $\overline{X} = \overline{Y}F'(x)$ of q gradient vectors. For example in multicriteria optimization one may well have $q \approx 10$ objectives or soft constraints, whose gradients are needed simultaneously.

Worst-Case Optimality

Counting only multiplications we obtain for Jacobians $F' \in \mathbf{R}^{m \times n}$ the complexity bound

$$\mathrm{OPS}\{x \overset{\mathrm{ad}}{\mapsto} F'\} \leq 3 \min(n, m)\, \mathrm{OPS}\{x \overset{\mathrm{prog}}{\longmapsto} y\} \ .$$

Here, n and m can be reduced to the maximal number of nonzero entries in the rows and columns of the Jacobian, respectively.

Similarly, we have for the one-sided projection of the Lagrangian Hessian

$$H(x, \overline{y}) \equiv \overline{y}F'' \equiv \sum_{i=1}^{m} \overline{y}_i \nabla^2 F_i \in \mathbb{R}^{n \times n}$$

onto the space spanned by the columns of $\dot{X}$:

$$\mathrm{OPS}\{x \overset{\mathrm{ad}}{\mapsto} H(x, \overline{y})\dot{X}\} \leq 9p\, \mathrm{OPS}\{x \overset{\mathrm{prog}}{\longmapsto} y\} \ .$$

As we already discussed for indefinite integrals there are certainly functions whose derivatives can be evaluated much cheaper than they themselves for example using a computer algebra package. Note that here again we have neglected the preparation effort, which may be very substantial for symbolic differentiation. Nevertheless, the estimates given above for AD are optimal in the sense that there are vector functions F defined by evaluation procedures of the form (1), for which no differentiation process imaginable can produce the Jacobian and projected Hessian significantly cheaper than the given cost bound divided by a small constant. Here, producing these matrices is understood to mean calculating all its elements explicitly, which may or may not be actually required by the overall computation.

Consider, for example, the cubic vector function

$$F(x) = x + \frac{b(a^\top x)^3}{2} \quad \text{with } a, b \in \mathbb{R}^n \ .$$

Its Jacobian and projected Hessian are given by

$$F'(x) = I + b\left(a^\top x\right)^2 a^\top \in \mathbb{R}^{n \times n}$$

and

$$H(x, \overline{y})\dot{X} = 2a(\overline{y}b)(a^\top x)a^\top \dot{X} \in \mathbb{R}^{n \times p} \ .$$

For general a, b and $\dot{X}$, all entries of the matrices $F'(x)$ and $H(x, \overline{y})\dot{X}$ are distinct and depend nontrivially on x. Hence their explicit calculation by any method requires at least n^2 or np arithmetic operations, respectively. Since the evaluation of F itself can be performed using just $3n$ multiplications and a few additions, the operations count ratios given above cannot be improved by more than a constant. There are other, more meaningful examples [9] with the same property, namely that their Jacobians and projected Hessians are orders of magnitude more expensive than the vector function itself. At least this is true if we insist on representing them as rectangular arrays of reals. This does not contradict our earlier observation that gradients are cheap, because the components of $F(x)$ cannot be considered as independent scalar functions. Rather, their simultaneous evaluation may involve many common subexpressions, as is the case for our rank-one example. These appear to be less beneficial for the corresponding derivative evaluation, thus widening the gap between function and derivative complexities.

Expensive ≡ Redundant?

The rank-one problem and similar examples for which explicit Jacobians or Hessians appear to be expensive have a property that one might call *redundancy*. Namely, as x varies over some open neighborhood in its domain, the Jacobian $F'(x)$ stays in a lower-dimensional manifold of the linear space of all matrices with its format and sparsity pattern. In other words, the nonzero entries of the Jacobian are not truly independent of each other so that computing them all and storing them separately may be wasteful. In the rank-one example the Jacobian $F'(x)$ is dense but belongs at all x to the one-dimensional affine variety $\{\mathrm{I} + \mathrm{b}\alpha a^\mathsf{T} : \alpha \in \mathbf{R}\}$. Note that the vectors a, $b \in \mathbf{R}^n$ are assumed to be dense and constant parameter vectors of the problem at hand. Their elements all play the role of elemental partials c_{ij} with the corresponding operation φ_i being multiplications. Hence accumulating the extremely sparse trian-

gular matrix C, which involves only $O(n)$ nonzero entries, to the dense $n \times n$ array $F'(x)$ is almost certainly a bad idea, no matter what the ultimate purpose of the calculation. In particular, if one wishes to solve linear systems in the Jacobian, the inverse formula of Sherman–Morrison–Woodbury provides a way of computing the solution of rank-one perturbations to diagonal matrices with $O(n)$ effort. This formula may be seen as a very special case of embedding linear systems in F' into a much larger and sparse linear system involving C as demonstrated in [11] and [5].

As of now, all our examples for which the array representation of Jacobians and Hessians are orders of magnitude more expensive to evaluate than the underlying vector function exhibit this redundancy property. In other words, we know of no convincing example where vectors that one may actually wish to calculate as end products are necessarily orders of magnitude more expensive than the functions themselves. Especially for large problems it seems hard to imagine that array representations of the Jacobians and Hessians themselves are really something anybody would wish to look at rather than just use as auxiliary quantities within the overall calculation.

So evaluating complete derivative arrays is a bit like fitting a handle to a wooden crate that needs to be moved about frequently. If the crate is of small weight and size this job is easily performed using a few screws. If, on the other hand, the crate is large and heavy, fitting a handle is likely to require additional bracing and other reinforcements. Moreover, this effort is completely pointless since nobody can just pick up the crate by the handle anyhow and one might as well use a fork left in the first place.

Preaccumulation and Combinatorics

The temporal complexity for both the forward and the reverse (vector) mode are proportional to the number of edges in the linearized computational graph. Hence one may try to reduce the number of edges by certain algebraic manipulations that leave the corresponding Jacobian, i. e., the linear mapping between $\dot{x}$ and $\dot{y} = F'(x)\dot{x}$ and equivalently also that between $\overline{y}$ and $\overline{x} = \overline{y}F'(x)$ unchanged. It can be easily checked that this is the case if given an index j one updates first

$$c_{ik} += c_{ij}c_{jk}$$

either for fixed $i \succ j$ and all $k \prec j$, or for fixed $k \prec j$ and all $i \succ j$, and then sets $c_{ij} = 0$ or $c_{jk} = 0$, respectively. In other words, either the edge (j, i) or the edge (k, j) is eliminated from the graph. This leads to fill-in by the creation of new arcs, unless all updated c_{ik} were already nonzero beforehand. Eliminating all edges (k, j) with $k \prec j$ or all edges (j, i) with $i \succ j$ is equivalent and amounts to eliminating the vertex j completely from the graph. After all intermediate vertices $1 \leq j \leq l$ are eliminated in some arbitrary order, the remaining edges c_{ij} directly connect independent variables with dependent variables and are therefore entries of the Jacobian $F'(x)$. Hence, one refers to the *accumulation* of the Jacobian F' if all intermediate nodes are eliminated and to *preaccumulation* if some of them remain so that the Jacobian is represented by a simplified graph.

As we have indicated in the section on goal oriented differentiation one would have to carefully look at the problem function and the overall computational task to decide how much preaccumulation should be performed. Moreover, there are $\widetilde{l}!$ different orders in which a particular set of $\widetilde{l} \leq l$ intermediate nodes can be eliminated and even many more different ways of eliminating the corresponding set of edges. So far there have only been few studies of heuristic criteria for finding efficient elimination orderings down to an appropriate preaccumulation level [9].

Summary

First and second derivative vectors of the form $\dot{y} = F'(x)\dot{x}$, $\overline{x} = \overline{y}F'(x)$ and $\dot{\overline{x}} = \overline{y}F''(x)\dot{x}$ can be evaluated for a fixed small multiple of the temporal complexity of the underlying relation $y = F(x)$. The calculation of the gradient $\overline{x}$ and the second order adjoint $\dot{\overline{x}}$ by the basic reverse method may require storage of order $l \equiv$ #intermediates. This possibly unacceptable amount can be reduced to order $\log(l)$ at a slight increase in the operations count (see [8]).

Jacobians and one-sided projected Hessians can be composed column by column or row by row from vectors of the kind $\dot{y}$, $\overline{x}$ and $\dot{\overline{x}}$. For sparse derivative matrices row and/or column compression using suitable seed matrices of type CPR or NR allow a substantial reduction of the computational effort. In some cases the nonzero entries of derivative matrices may be redundant, so that their calculation should be avoided, if

the overall computational goal can be reached in some other way. The attempt to evaluate derivative array with absolutely minimal effort leads to hard combinatorial problems.

See also

- ▶ Complexity Classes in Optimization
- ▶ Complexity of Degeneracy
- ▶ Complexity Theory
- ▶ Complexity Theory: Quadratic Programming
- ▶ Computational Complexity Theory
- ▶ Fractional Combinatorial Optimization
- ▶ Information-Based Complexity and Information-Based Optimization
- ▶ Kolmogorov Complexity
- ▶ Mixed Integer Nonlinear Programming
- ▶ *NP*-Complete Problems and Proof Methodology
- ▶ Parallel Computing: Complexity Classes

References

1. Bauer FL (1974) Computational graphs and rounding error. SIAM J Numer Anal 11:87–96
2. Berz M, Bischof Ch, Corliss G, Griewank A (eds) (1996) Computational differentiation: Techniques, applications, and tools. SIAM, Philadelphia
3. Bischof Ch, Carle A, Corliss G, Griewank A, Hovland P (1992) ADIFOR: Generating derivative codes from Fortran programs. Scientif Program 1:1–29
4. Coleman TF, Morée JJ (1984) Estimation of sparse Jacobian matrices and graph coloring problems. SIAM J Numer Anal 20:187–209
5. Coleman TF, Verma A (1996) Structure and efficient Jacobian calculation. In: Berz M, Bischof Ch, Corliss G, Griewank A (eds) Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia, pp 149–159
6. Curtis AR, Powell MJD, Reid JK (1974) On the estimation of sparse Jacobian matrices. J Inst Math Appl 13:117–119
7. Griewank A (1991) The chain rule revisited in scientific computing, I–II. SIAM News
8. Griewank A (1992) Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. Optim Methods Softw 1:35–54
9. Griewank A (2000) Evaluating derivatives, principles and techniques of algorithmic differentiation. Frontiers in Appl Math, vol 19. SIAM, Philadelphia
10. Griewank A, Corliss GF (eds) (1991) Automatic differentiation of algorithms: Theory, implementation, and application. SIAM, Philadelphia
11. Griewank A, Reese S (1991) On the calculation of Jacobian matrices by the Markowitz rule. In: Griewank A and Corliss GF (eds) Automatic Differentiation of Algorithms: Theory, Implementation, and Application. SIAM, Philadelphia, pp 126–135
12. Newsam GN, Ramsdell JD (1983) Estimation of sparse Jacobian matrices. SIAM J Alg Discrete Meth 4:404–417

Complexity and Large-Scale Least Squares Problems

JOSEF KALLRATH
GVCS, BASF Aktiengesellschaft,
Ludwigshafen, Germany

MSC2000: 93E24, 34-xx, 34Bxx, 34Lxx

Article Outline

Introduction
  A Standard Formulation for Unconstrained Least Squares Problem
  Solution Methods
  Explicit Versus Implicit Models
  Practical Issues of Solving Least Squares Problems
Parameter Estimation in ODE Models
  The Initial Value Problem Approach
  The Boundary Value Problem Approach
Parameter Estimation in DAE Models
Parameter Estimation in PDE Models
  Methodology
Least Squares Problems with Massive Data Sets
  The Matching Approach
Conclusions
Acknowledgments
References

Introduction

Least squares problems and solution techniques to solve them have a long history briefly addressed by Björck [4]. In this article we focus on two classes of complex least squares problems. The first one is established by models involving *differential equations*. The other class is made by least squares problems involving difficult models which need to be solved for many independent observational data sets. We call this *least squares problems with massive data sets*.

A Standard Formulation for Unconstrained Least Squares Problem

The unconstrained least squares problem can be expressed by

$$\min_{\mathbf{p}} l_2(\mathbf{p})\,,\ l_2(\mathbf{p}) := \left\|\mathbf{r}_1\left[\mathbf{x}(t_1),\ldots,\mathbf{x}(t_k),\mathbf{p}\right]\right\|_2^2 = \sum_{k=1}^{N}\left[r_{1k}(\mathbf{p})\right]^2\,,\quad \mathbf{r}_1\in\mathbb{R}^N\,. \tag{1}$$

The minimization of this functional, i. e., the minimization of the sum of weighted quadratic residuals, under the assumption that the statistical errors follow a Gaußian distribution with variances as in (4), provides a maximum likelihood estimator ([7] Chap. 7) for the unknown parameter vector $\mathbf{p}$. This objective function dates back to Gauß [14] and in the mathematical literature the problem is synonymously called least squares or ℓ_2 approximation problem.

The least squares structure (1) may arise either from a nonlinear over-determined system of equations

$$r_{1k}(\mathbf{p}) = 0\,,\quad k = 1,\ldots,N\,,\quad N > n\,, \tag{2}$$

or from a data fitting problem with N given data points $(t_k,\tilde{Y}_k)$ and variances σ_ν, a model function $\tilde{F}(t,\mathbf{p})$, and n adjustable parameters $\mathbf{p}$:

$$r_{1k} := r_{1k}(\mathbf{p}) = Y_k - F_k(\mathbf{p}) = \sqrt{w_k}\left[\tilde{Y}_k - \tilde{F}(t_k,\mathbf{p})\right]\,. \tag{3}$$

The weights w_k are related to the variances σ_k by

$$w_k := \beta/\sigma_k^2\,. \tag{4}$$

Traditionally, the weights are scaled to a variance of unit weights. The factor β is chosen so as to make the weights come out in a convenient range. In short vector notation we get

$$\mathbf{r_1} := \mathbf{Y} - \mathbf{F}(\mathbf{p}) = \left[r_{11}(\mathbf{p}),\ldots,r_{1N}(\mathbf{p})\right]^{\mathrm{T}}\,,\quad \mathbf{F}(\mathbf{p}),\mathbf{Y}\in\mathbb{R}^N\,.$$

Our least squares problem requires us to provide the following input:

1. model,
2. data,
3. variances associated with the data,
4. measure of goodness of the fit, e. g., the Euclidean norm.

In many practical applications, unfortunately, less attention is paid to the variances. It is also very important to point out that the use of the Euclidean norm requires pre-information related to the problem and statistical properties of the data.

Solution Methods

Standard methods for solving linear version of (1), i. e., $\mathbf{F}(\mathbf{p}) = \mathbf{A}\mathbf{p}$, are reviewed by Björck [4]. Nonlinear methods for unconstrained least squares problems are covered in detail by Xu [35,36,37]. In addition, we mention a popular method to solve unconstrained least squares problems: the Levenberg–Marquardt algorithm proposed independently by Levenberg [21] and Marquardt [22] and sometimes also called "damped least squares". It modifies the eigenvalues of the normal equation matrix and tries to reduce the influence of eigenvectors related to small eigenvalues (cf. [8]). Damped (step-size cutting) Gauß–Newton algorithms combined with orthogonalization methods control the damping by natural level functions [6,9,10] seem to be superior to Levenberg–Marquardt type schemes and can be more easily extended to nonlinear constrained least squares problems.

Explicit Versus Implicit Models

A common basic feature and limitation of least squares methods, but seldom explicitly noted, is that they require some *explicit* model to be fitted to the data. However, not all models are explicit. For example, some pharmaceutical applications for receptor-ligand binding studies are based on specifically coupled mass equilibrium models. They are used, for instance, for the radioimmunological determination of Fenoterol or related substances, and lead to least squares problems in systems of nonlinear equations [31], in which the model function $\mathbf{F}(\mathbf{p})$ is replaced by $\mathbf{F}(t;\mathbf{p},\mathbf{z})$ which, besides the parameter vector $\mathbf{p}$ and the time t, depends on a vector function $\mathbf{z} = \mathbf{z}(t;\mathbf{p})$ implictly defined as the solution of the nonlinear equations

$$\mathbf{F}_2(t;\mathbf{p},\mathbf{z}) = \mathbf{0}\,,\quad \mathbf{F}_2(\mathbf{p})\in\mathbb{R}^{n_2}\,. \tag{5}$$

This is a special case of an implicit model. There is a much broader class of implicit models. Most models

in science are based on physical, chemical and biological laws or include geometry properties, and very often lead to differential equations which may, however, not be solvable in a closed analytical form. Thus, such models do not lead to explicit functions or models we want to fit to data. We rather need to fit an implicit model (represented by a system of differential equations or another implicit model). The demand for and the applications of such techniques are widespread in science, especially in the rapidly increasing fields of nonlinear dynamics in physics and astronomy, nonlinear reaction kinetics in chemistry [5], nonlinear models in material sciences [16] and biology [2], and nonlinear systems describing ecosystems [28,29] in biology, or the environmental sciences. Therefore, it seems desirable to focus on least squares algorithms that use nonlinear equations and differential equations as constraints or side conditions to determine the solution implicitly.

Practical Issues of Solving Least Squares Problems

Solving least squares problems involves various difficulties among them to find an appropriate model, nonsmooth models with discontinuous derivatives, data quality and checking the assumption of the underlying error distribution, and dependence on initial parameter or related questions of global convergence.

Models and Model Validation A model may be defined as an appropriate abstract representation of a real system. In the natural sciences (e. g., Physics, Astronomy, Chemistry and Biology) models are used to gain a deeper understanding of processes occurring in nature (an epistemological argument). The comparison of measurements and observations with the predictions of a model is used to determine the appropriateness and quality of the model. Sir Karl Popper [26] in his famous book *Logic of Scientific Discovery* uses the expressions *falsification* and *verification* to describe tasks that the models can be used to accomplish as an aid to scientific process. Models were used in early scientific work to explain the movements of planets. Then, later, aspects and questions of accepting and improving global and fundamental models (e. g., general relativity or quantum physics) formed part of the discussion of the philosophy of science. In science models are usually falsified, and, eventually, replaced by modified or completely different ones.

In industry, models have a rather local meaning. A special aspect of reality is to be mapped in detail. Pragmatic and commercial aspects are usually the motivation. The model maps most of the relevant features and neglect less important aspects. The purpose is to

- provide insight into the problem,
- allow numerical, *virtual* experimentation but avoid expensive and/or dangerous *real* experiments, or
- tune a model for later usage, i. e., determine, for instance, the reaction coefficients of a chemical system – once these parameters are known the dynamics of the process can be computed.

A (mathematical) model represents a *real-world problem* in the language of mathematics, i. e., by using mathematical symbols, variables (in this context: the adjustable least squares parameters), equations, inequalities, and other relations. How does one get a mathematical model for a real-world problem? To achieve that is neither easy nor unique. In some sense it is similar to solving exercises in school where problems are put in a verbal way [25]. The following points are useful to remember when trying to build a model:

- there will be no precise recipe telling the user how to build a model,
- experience and judgment are two important aspects of model building,
- there is nothing like a *correct* model,
- there is no concept of a *unique* model, as different models focusing on different aspects may be appropriate.

Industrial models are eventually validated which means that they reached a sufficient level of consensus among the community working with these models.

Statistics provide some means to discriminate models but this still is an art and does not replace the need for appropriate model validation. The basic notion is: *with a sufficient number of parameters on can fit an elefant*. This leads us to one important consequence: it seems to be necessary that one can interpret these model parameters. A reasonable model derived from the laws of science with interpretable parameters is a good candidate to become accepted. Even, if it may lead to a somewhat worse looking fits than a model with a larger number of formal parameters without interpretation.

Non-Smooth Models The algorithm reviewed by Xu [35,36,37] for solving least squares problems usually require the continuous first derivatives of the model function with respect to the parameters. We might, however, encounter models for which the first derivatives are discontinuous. Derive-free methods such as Nelder and Mead's [23] downhill Simplex method, or direction set methods; cf. ([27], p. 406) have been successfully used to solve least squares problems. The Simplex method provides the benefit of exploring parameter space and good starting values for derivative based methods. Powell's direction set method with appropriate conjugate directions preserve the derivative free nature of the method.

Global Convergence Nonlinear least squares algorithms usually converge only if the initial parameters are close to the best fit parameters. Global convergence can be established for some algorithms, i. e., they converge for all initial parameters. An essential support tool accompanying the analysis of difficult least squares problem is to visualize the data and the fits. Inappropriate or premature fits can easily be excluded. Inappropriate fits are possible because all algorithms mentioned in Sect. "Introduction", "Parameter Estimation in ODE Models", and "Parameter Estimation in DAE Models" are local algorithm. Only if the least squares problem is convex, they yield the global least squares minimum. Sometimes, it is possible to identify false local minima from the residuals.

Data and Data Quality Least squares analysis is concerned by fitting data to a model. The data are not exact but subject to unknown random errors $\boldsymbol{\varepsilon}_k$. In ideal cases these errors follow a Gaussian normal distribution. One can test this assumption after the least squares fit by analyzing the distribution of the residuals as described in Sect. "Residual Distributions, Covariances and Parameter Uncertainties". Another important issue is whether the data are appropriate to estimate all parameters. Experimental design is the discipline which addresses this issue.

Residual Distributions, Covariances and Parameter Uncertainties Once the minimal least squares solution has been found one should at first check with the χ^2-test or Kolmogoroff–Smirnov test whether the usual assumption that the distribution really follows a Gaussian normal distribution. With the Kolmogoroff–Smirnov test (see, e. g., [24]) it is possible to check as follows whether the residuals of a least-squares solution are normally distributed around the mean value 0.

1. let $M := (x_1, x_2, \ldots, x_n)$ be a set of observations for which a given hypothesis should be tested;
2. let $G : x \in M \to \mathbb{R}$, $x \to G(x)$, be the corresponding cumulative distribution function;
3. for each observation $x \in M$ define $S_n(x) := k/n$, where k is the number of observations less than or equal to x;
4. determine the maximum $D := \max(G(x) - S_n(x) \mid x \in M)$;
5. D_{crit} denotes the maximum deviation allowed for a given significance level and a set of n elements. D_{crit} is tabulated in the literature, e. g., ([24], Appendix 2, p. 560); and
6. if $D < D_{\text{crit}}$, the hypothesis is accepted.

For the least squares problem formulated in Sect. "A Standard Formulation for Unconstrained Least Squares Problem" the hypothesis is "The residuals $\mathbf{x} := \mathbf{r_1} = \mathbf{Y} - \mathbf{F}(\mathbf{p})$ are normally distributed around the mean value 0". Therefore, the cumulative distribution function $G(x)$ takes the form

$$\begin{aligned}\sqrt{2\pi}G(x) &= \int_{-\infty}^{x} g(z)\mathrm{d}z \\ &= \int_{-\infty}^{-x_0} g(z)\mathrm{d}z + \int_{-x_0}^{x} g(z)\mathrm{d}z, \\ g(z) &:= e^{-\frac{1}{2}z^2}\,.\end{aligned}$$

The value x_0 separates larger residuals; this is problem specific control parameter.

The derivative based least squares methods usually also give the covariance matrix from which the uncertainties of the parameter are derived; cf. [7], Chap. 7. Least squares parameter estimations without quantifying the uncertainty of the parameters are very doubtful.

Parameter Estimation in ODE Models

Consider a differential equation with independent variable t for the state variable

$$\mathbf{x}'(t) = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(t, \mathbf{x}, \mathbf{p}), \quad \mathbf{x} \in \mathbb{R}^{n_d} \quad , \quad \mathbf{p} \in \mathbb{R}^{n_p} \quad (6)$$

with a right hand side depending on an unknown parameter vector $\mathbf{p}$. Additional requirements on the solu-

tion of the ODE (1) like periodicity, initial or boundary conditions or range restrictions to the parameters can be formulated in vectors $\mathbf{r}_2$ and $\mathbf{r}_3$ of (component wise) equations and inequalities

$$\begin{aligned} &\mathbf{r}_2\left[\mathbf{x}(t_1),\ldots,\mathbf{x}(t_k),\mathbf{p}\right]=0 \quad \text{or} \\ &\mathbf{r}_3\left[\mathbf{x}(t_1),\ldots,\mathbf{x}(t_k),\mathbf{p}\right]\geq 0\,. \end{aligned} \tag{7}$$

The multi-point boundary value problem is linked to experimental data via minimization of a least squares objective function

$$l_2(\mathbf{x},\mathbf{p}) := \left\|\mathbf{r}_1\left[\mathbf{x}(t_1),\ldots,\mathbf{x}(t_k),\mathbf{p}\right]\right\|_2^2\,. \tag{8}$$

In a special case of (8) the components ℓ of the vector $\mathbf{r}_1 \in \mathbb{R}^L$ are "equations of condition" and have the form

$$r_{1\ell} = \sigma_{ij}^{-1}\left[\eta_{ij} - g_i(\mathbf{x}(t_j),\mathbf{p})\right]\,,$$
$$\ell = 1,\ldots,L := \sum_{i=1}^{N_j} J_i\,. \tag{9}$$

This case leads us to the least squares function

$$l_2(\mathbf{x},\mathbf{p}) := \sum_{j=1}^{N^D}\sum_{i=1}^{N_j}\sigma_{ij}^{-2}\left[\eta_{ij} - g_i(\mathbf{x}(t_j),\mathbf{p})\right]^2\,. \tag{10}$$

Here, N^D denotes the number of values of the independent variable (here called time) at which observed data are available, N_j denotes the number of observables measured at time t_j and η_{ij} denotes the observed value which is compared with the value of observable i evaluated by the model where the functions $g_i(\mathbf{x}(t_j),\mathbf{p})$ relate the state variables to $\mathbf{x}$ this observable

$$\eta_{ij} = g_i(\mathbf{x}(t_j),\mathbf{p}) + \varepsilon_{ij}\,. \tag{11}$$

The numbers ε_{ij} are the measurement errors and σ_{ij}^2 are weights that have to be adequately chosen due to statistical considerations, e. g. as the variances. The unknown parameter vector $\mathbf{p}$ is determined from the measurements such that the model is optimally adjusted to the measured (observed) data. If the errors ε_{ij} are independent, normally distributed with the mean value zero and have variances σ_{ij}^2 (up to a common factor β^2), then the solution of the least squares problem is a *maximum likelihood estimate*.

The Initial Value Problem Approach

An obvious approach to estimate parameters in ODE which is also implemented in many commercial packages is the initial value problem approach. The idea is to guess parameters and initial values for the trajectories, compute a solution of an initial value problem (IVP) (6) and iterate the parameters and initial values in order to improve the fit. Characteristic features and disadvantages are discussed in, e. g., [6] or [18]. In the course of the iterative solution one has to solve a sequence of IVPs. The state variable $\mathbf{x}(t)$ is eliminated for the benefit of the unknown parameter $\mathbf{p}$ and the initial values. Note that no use is made of the measured data while solving the IVPs. They only enter in the performance criterion. Since initial guesses of the parameters may be poor, this can lead to IVPs which may be hard to solve or even have no solution at all and one can come into badly conditioned regions of the IVPs, which can lead to the loss of stability.

The Boundary Value Problem Approach

Alternatively to the IVP approach, in the "boundary value problem approach" invented by Bock [5], the inverse problem is interpreted as an over-determined, constrained, multiple-point boundary problem. This interpretation does not depend on whether the direct problem is an initial or boundary value problem. The algorithm used here consists of an adequate combination of a multiple shooting method for the discretization of the boundary value problem side condition in combination with a generalized Gauss-Newton method for the solution of the resulting structured nonlinear constrained least squares problem [5,6]. Depending on the vector of signs of the state and parameter dependent switching functions $\mathbf{Q}$ it is even possible to allow piecewise smooth right hand side functions f, i. e., differential equations with switching conditions

$$\mathbf{x}' = f(t,\mathbf{x},\mathbf{p};sign(\mathbf{Q}(t,\mathbf{x},\mathbf{p})))\,, \tag{12}$$

where the right side may change discontinuously if the vector of signs of the switching functions $\mathbf{Q}$ changes. Such discontinuities can occur, e. g. as a result of unsteady changes of physical values. The switching points are in general given by the roots of the state-dependent components of the switching functions

$$\mathbf{Q}_i(t,\mathbf{x},\mathbf{p}) = 0\,. \tag{13}$$

Depending on the stability behavior of the ODE and the availability of information about the process (measured data, qualitative knowledge about the problem, etc.) a grid $\mathcal{T}_m$

$$\mathcal{T}_m : \tau_1 < \tau_2 < \ldots < \tau_m\,, \quad \Delta\tau_j := \tau_{j+1} - \tau_j\,, \quad 1 \le j \le m-1\,, \tag{14}$$

of m multiple shooting nodes τ_j ($m-1$ subintervals I_j) is chosen. The grid is adapted to the problem and data and is defined such that it includes the measuring interval ($[\tau_1, \tau_m] = [t_0, t_f]$). Usually, the grid points τ correspond to values of the independent variable t at which observations are available but additional grid points may be chosen for strongly nonlinear models. At each node τ_j an IVP

$$\mathbf{x}'(t) = \mathbf{f}(t, \mathbf{x}, \mathbf{p})\,, \quad \mathbf{x}(t = \tau_j) = \mathbf{s}_j \in \mathbb{R}^{n_d} \tag{15}$$

has to be integrated from τ_j to τ_{j+1}. The $m-1$ vectors of (unknown) initial values $\mathbf{s}_j$ of the partial trajectories, the vector $\mathbf{s}_m$ representing the state at the end point and the parameter vector $\mathbf{p}$ are summarized in the (unknown) vector $\mathbf{z}$

$$\mathbf{z}^T := (\mathbf{s}_1^T, \ldots, \mathbf{s}_m^T, \mathbf{p}^T)\,. \tag{16}$$

For a given guess of $\mathbf{z}$ the solutions $\mathbf{x}(t; \mathbf{s}_j, \mathbf{p})$ of the $m-1$ independent initial value problems in each sub interval I_j are computed. This leads to an (at first discontinuous) representation of $\mathbf{x}(t)$. In order to replace (6) equivalently by these $m-1$ IVPs matching conditions

$$\mathbf{h}_j(\mathbf{s}_j, \mathbf{s}_{j+1}, p) := \mathbf{x}(\tau_{j+1}; \mathbf{s}_j, \mathbf{p}) - \mathbf{s}_{j+1} = 0\,, \quad \mathbf{h}_j : \mathbb{R}^{2n_d+n_p} \to \mathbb{R}^{n_d} \tag{17}$$

are added to the problem. (17) ensures the continuity of the final trajectory $\mathbf{x}(t)$.

Replacing $\mathbf{x}(t_i)$ and $\mathbf{p}$ in (10) by $\mathbf{z}$ the least squares problem is reformulated as a nonlinear constrained optimization problem with the structure

$$\min_{\mathbf{z}} \left\{ \frac{1}{2}\|\mathbf{F}_1(\mathbf{z})\|_2^2 \,|\mathbf{F}_2(\mathbf{z}) = 0 \in \mathbb{R}^{n_2}\,, \quad \mathbf{F}_3(\mathbf{z}) \ge 0 \in \mathbb{R}^{n_3} \right\}\,, \tag{18}$$

wherein n_2 denotes the number of the equality and n_3 the number of the inequality constraints. This usually large constrained structured nonlinear problem is solved by a damped generalized Gauss-Newton method [5]. If $\mathrm{J}_1(\mathbf{z}_k) := \partial_z \mathbf{F}_1(\mathbf{z}_k)$, $\mathrm{J}_2(z_k) := \partial_z \mathbf{F}_2(\mathbf{z}_k)$ vis. $\mathrm{J}_3(\mathbf{z}_k) := \partial_z \mathbf{F}_3(\mathbf{z}_k)$ denote the Jacobi matrices of $\mathbf{F}_1$, $\mathbf{F}_2$vis. $\mathbf{F}_3$, then the iteration proceeds as

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k \boldsymbol{\Delta z}_k \tag{19}$$

with damping constant $\alpha_k, 0 < \alpha_{min} \le \alpha_k \le 1$, and the increment $\boldsymbol{\Delta z}_k$ determined as the solution of the constrained linear problem

$$\min_{\mathbf{z}} \left\{ \frac{1}{2}\|\mathrm{J}_1(\mathbf{z}_k)'\mathbf{z}_k + \mathbf{F}_1(\mathbf{z}_k)\|_2^2 \,\middle|\, \begin{array}{l} \mathrm{J}_2(\mathbf{z}_k)\boldsymbol{\Delta z}_k + \mathbf{F}_2(\mathbf{z}_k) = 0 \\ \mathrm{J}_3(\mathbf{z}_k)\boldsymbol{\Delta z}_k + \mathbf{F}_3(\mathbf{z}_k) \ge 0 \end{array} \right\}\,. \tag{20}$$

Global convergence can be achieved if the damping strategy is properly chosen [6].

The inequality constraints that are active in a feasible point are defined by the index set

$$\mathcal{I}(\mathbf{z}_k) := \{i | F_{3i}(\mathbf{z}_k) = 0\,, \quad i = 1, \ldots, n_3\}\,. \tag{21}$$

The inequalities which are defined by the index set $\mathcal{I}(\mathbf{z}_k)$ or their derivatives are denoted with $\hat{\boldsymbol{F}}_3$ or $\hat{\mathrm{J}}_3$ in the following. In addition to (21) we define

$$\mathbf{F}_c := \begin{pmatrix} \mathbf{F}_2 \\ \hat{\boldsymbol{F}}_3 \end{pmatrix}\,, \quad \mathrm{J}_c := \begin{pmatrix} \mathrm{J}_2 \\ \hat{\mathrm{J}}_3 \end{pmatrix}\,. \tag{22}$$

In order to derive the necessary conditions that have to be fulfilled by the solution of the problem (18) the Lagrangian

$$L(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) := \frac{1}{2}\|\mathbf{F}_1(\mathbf{z})\|_2^2 - \boldsymbol{\lambda}^T\mathbf{F}_2(\mathbf{z}) - \boldsymbol{\mu}^T\mathbf{F}_3(\mathbf{z}) \tag{23}$$

and the reduced Lagrangian

$$\hat{L}(\mathbf{z}, \boldsymbol{\lambda}_c) := \frac{1}{2}\|\mathbf{F}_1(\mathbf{z})\|_2^2 - \boldsymbol{\lambda}_c^T\mathbf{F}_c(\mathbf{z}) \quad , \quad \boldsymbol{\lambda}_c := \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\mu}_c \end{pmatrix} \tag{24}$$

are defined. The Kuhn–Tucker-conditions, i. e. the necessary conditions of first order, are the feasibility conditions

$$\mathbf{F}_2(\mathbf{z}^*) = 0\,, \quad \mathbf{F}_3(\mathbf{z}^*) \ge 0 \tag{25}$$

ensuring that $\mathbf{z}^*$ is feasible, and the stationarity conditions stating that the adjoined variables λ^*, μ^* exist as

solution of the stationary conditions

$$\frac{\partial L}{\partial z}(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = \mathbf{F}_1^T(\mathbf{z}^*) \cdot \mathsf{J}(\mathbf{z}^*) - \left(\boldsymbol{\lambda}^*\right)^T \mathsf{J}_2(\mathbf{z}^*) - \left(\boldsymbol{\mu}^*\right)^T \mathsf{J}_3(\mathbf{z}^*) = 0 \quad (26)$$

and

$$\boldsymbol{\mu}^* \geq 0\,, \quad i \notin \mathcal{I}(\mathbf{z}^*) \Rightarrow \mu_i^* = 0\,. \quad (27)$$

If $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ fulfills the conditions (25), (26) and (27), it is called a Kuhn–Tucker-point and $\mathbf{z}^*$ a stationary point. The necessary condition of second order means that for all directions

$$\mathbf{s} \in T(\mathbf{x}^*) := \left\{ \mathbf{s} \neq 0 \;\middle|\; \begin{array}{l} \mathsf{J}_2(\mathbf{z}^*)\mathbf{s} = 0 \\ \mathsf{J}_3(\mathbf{z}^*)\mathbf{s} \geq 0 \end{array} \;,\; \mu_i \mathsf{J}_{3i}(\mathbf{z}^*)\mathbf{s} = 0 \right\} \quad (28)$$

the Hessian $\mathsf{G}(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ of the Lagrangian is positive semi-definite:

$$\mathbf{s}^T \mathsf{G}(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\mathbf{s} \geq 0\,,$$
$$\mathsf{G}(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) := \frac{\partial^2}{\partial \mathbf{z}^2} L(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\,. \quad (29)$$

As $\mu_i = 0$ for $i \notin \mathcal{I}(\mathbf{z}^*)$ it is sufficient to postulate the stationary condition for the reduced Lagrangian (24). For the linear problem (20) follows: $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ is a Kuhn–Tucker-point of the nonlinear problem (18) if and only, if $(0, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ is a Kuhn–Tucker-point of the linear problem. The necessary conditions for the existence of a local minimum of problem (18) are:

1. $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ is a Kuhn–Tucker-point of the nonlinear problem
2. the Hessian $\mathsf{G}(\mathbf{z}^*, \boldsymbol{\lambda}^*, —^*)$ of the Lagrangian is positive definite for all directions $\mathbf{s} \in T(\mathbf{x}^*)$, vis. $\mathbf{s}^T G(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\mathbf{s} > 0$

If the necessary conditions for the existence of the local minimum and the condition $\mu_i \neq 0$ for $i \in \mathcal{I}(\mathbf{z}^*)$ are fulfilled, two perturbation theorems [6] can be formulated. If the sufficient conditions are fulfilled it can be shown for the neighborhood of a Kuhn–Tucker-point $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ of the nonlinear problem (18) that the local convergence behavior of the inequality constrained problem corresponds to that of the equality constrained problem which represents active inequalities and equations. Under the assumption of the regularity of the Jacobians J_1 and J_c, i. e.

$$\operatorname{rank}\begin{pmatrix} \mathsf{J}_1(\mathbf{z}_k) \\ \mathsf{J}_c(\mathbf{z}_k) \end{pmatrix} = n_d + n_p\,, \quad \operatorname{rank}(\mathsf{J}_c(\mathbf{z}_k)) = n_c\,, \quad (30)$$

a unique solution $\Delta\mathbf{z}_k$ of the linear problem (20) exists and an unique linear mapping J_k^+ can be constructed which satisfies the relation

$$\Delta\mathbf{z}_k = -\mathsf{J}_k^+ \mathbf{F}(\mathbf{z}_k)\,, \quad \mathsf{J}_k^+ \mathsf{J}_k \mathsf{J}_k^+ = \mathsf{J}_k^+\,,$$
$$\mathsf{J}_k^T := \left[\mathsf{J}_1^T(\mathbf{z}_k), \mathsf{J}_c^T(\mathbf{z}_k)\right]\,. \quad (31)$$

The solution $\Delta\mathbf{z}_k$ of the linear problem or formally the generalized inverse J_k^+ [5] of J_k results from the Kuhn–Tucker conditions. But it should be noticed that $\mathbf{z}_k$ is not calculated from (31) because of reasons of numerical efficiency but is based on a decomposition procedure using orthogonal transformations.

By taking into consideration the special structure of the matrices J_i caused by the continuity conditions of the multiple shooting discretization (18) can be reduced by a condensation algorithm described in [5,6] to a system of lower dimension

$$\min\left\{ \frac{1}{2}\|\mathsf{A}_1\mathbf{x}_k + \mathbf{a}_1\|_2^2 | \mathsf{A}_2\mathbf{x}_k + \mathbf{a}_2 = 0\,, \right.$$
$$\left. \mathsf{A}_3\mathbf{x}_k + \mathbf{a}_3 \geq 0\right\}, \quad (32)$$

from which $\mathbf{x}_k$ can be derived at first and at last $\Delta\mathbf{z}_k$. This is achieved by first performing a "backward recursion", the "solution of the condensed problem" and a "forward recursion" [6]. Kilian [20] has implemented an active set strategy following the description in [6] and [33] utilizing the special structure of J_2.

The details of the parameter estimation algorithms which are incorporated in the efficient software package PARFIT (a software package of stable and efficient boundary value problem methods for the identification of parameters in systems of nonlinear differential equations) are found in [6]. The damping constant α^k in the k-th iteration is computed with the help of *natural level functions* which locally approximate the distance $\|\mathbf{z}_k - \mathbf{z}^*\|$ of the solution from the Kuhn–Tucker point $\mathbf{z}^*$.

The integrator METANB (for the basic discretization see, for instance, [3]) embedded in PARFIT is also suitable for the integration of stiff differential equation sys-

tems. It allows the user to compute simultaneously the sensitivity matrixes G,

$$\mathsf{G}(t; t_0, \mathbf{x}_0, \mathbf{p}) := \frac{\partial}{\partial \mathbf{x}_0}\mathbf{x}(t; t_0, \mathbf{x}_0, \mathbf{p}) \in \mathcal{M}(n_d, n_d) \quad (33)$$

and H,

$$\mathsf{H}(t; t_0, \mathbf{x}_0, \mathbf{p}) := \frac{\partial}{\partial \mathbf{p}}\mathbf{x}(t; t_0, \mathbf{x}_0, \mathbf{p}) \in \mathcal{M}(n_d, n_p) \quad (34)$$

which are the most costly blocks of the Jacobians J_i via the so-called *internal numerical differentiation* as introduced by Bock [5]. This technique does not require the often cumbersome and error prone formulation of the variational differential equations

$$\mathsf{G}' = \mathbf{f}_{\mathbf{x}}(t, \mathbf{x}, \mathbf{p}) \cdot \mathsf{G}\,, \quad \mathsf{G}(t_0; t_0, \mathbf{x}_0, \mathbf{p}) = \mathbb{1} \quad (35)$$

and

$$\mathsf{H}' = \mathbf{f}_x(t, \mathbf{x}, \mathbf{p}) \cdot \mathsf{H} + \mathbf{f}_p(t, \mathbf{x}, \mathbf{p})\,, \quad \mathsf{H}(t_0; t_0, \mathbf{x}_0, \mathbf{p}) = 0 \quad (36)$$

by the user.

Using the multiple shooting approach described above, differential equation systems with poor stability properties and even chaotic systems can be treated [18].

Parameter Estimation in DAE Models

Another, even more complex class of problems, are parameter estimation in mechanical multibody systems, e. g., in the planar slider crank mechanisms, a simple model for a cylinder in an engine. These problems lead to boundary problems for higher index differential algebraic systems [34]. Singular controls and state constraints in optimal control also lead to this structure. Inherent to such problems are invariants that arise from index reduction but also additional physical invariants such as the total energy in conservative mechanical systems or the Hamiltonian in optimal control problems.

A typical class of DAEs in mechanical multibody systems is given by the equations of motion

$$\begin{aligned} \dot{\boldsymbol{x}} &= \mathbf{v} \\ \mathsf{M}(t, \mathbf{x})\dot{\boldsymbol{v}} &= \mathbf{f}(t, \mathbf{x}) - \nabla_x \mathbf{g}(t, \mathbf{x})\boldsymbol{\lambda}\,, \\ 0 &= \mathbf{g}(t, \mathbf{x}) \end{aligned} \quad (37)$$

where $\mathbf{x} = \mathbf{x}(t)$ and $\mathbf{v} = \mathbf{v}(t)$ are the coordinates and velocities, M is the mass matrix, $\mathbf{f}$ denotes the applied forces, $\mathbf{g}$ are the holonomic constraints, and $\boldsymbol{\lambda}$ are the generalized constraint forces. Usually, M is symmetric and positive definite. A more general DAE system might have the structure

$$\begin{aligned} \dot{\boldsymbol{x}} &= \mathbf{f}(t, \mathbf{x}, \mathbf{z}; \mathbf{p}) \\ 0 &= \mathbf{g}(t, \mathbf{x}, \mathbf{z}; \mathbf{p})\,, \end{aligned}$$

where $\mathbf{p}$ denotes some parameters and $\mathbf{z} = \mathbf{z}(t)$ is a set of algebraic variables, i. e., the differentials $\dot{\boldsymbol{z}}$ do not appear; in (37) $\boldsymbol{\lambda}$ is the algebraic variable. In addition we might have initial values $\mathbf{x}_0$ and $\mathbf{z}_0$. Obviously, some care is needed regarding the choice of $\mathbf{z}_0$ because it needs to be consistent with the constraint. In some exceptional cases (in which $\mathsf{Z} := \nabla_z \mathbf{g}$ has full rank and can be inverted analytically) we might insert $\mathbf{z} = \mathbf{z}(t, \mathbf{x}; \mathbf{p})$ into the differential equation. DAE systems with a regular matrix Z are referred to as index-1 systems. Index-1-DAEs can be transformed into equivalent ordinary differential equations by differencing the equations w.r.t. t. At first we get the implicit system of differential equations

$$\mathbf{g}_t + \mathsf{X}\,\dot{\boldsymbol{x}} + \mathsf{Z}\dot{\boldsymbol{z}} = \mathbf{0}\,, \quad \mathsf{X} := \nabla_x \mathbf{g}$$

which, according to the assumption of the regularity of Z, can be written as the explicit system

$$\dot{\boldsymbol{z}} = \mathsf{Z}^{-1}\left(\mathbf{g}_t + \mathsf{X}\mathbf{f}\right)\,.$$

Many practical DAEs have index 1, e. g., in some chemical engineering problems, where algebraic equations are introduced to describe, for instance, mass balances or the equation of state. However, multibody systems such as (37) have higher indices; (37) is of index 3. The reason is, that the multiplier variables, i. e., the algebraic variables, do not occur in the algebraic constraints and it is therefore not possible to extract them directly without further differentiation. If Z does not have full rank the equations are differentiated successively, until the algebraic variables can be eliminated. The smallest number of differentiations required to transform the original DAE system to an ODE system is called the *index* of the DAE. The approach developed and described by Schulz et al. [34] is capable to handle least squares problems without special assumption to the index.

An essential problem for the design, optimization and control of chemical systems is the estimation of parameters from time-series. These problems lead to nonlinear DAEs. The parameters estimation problem leads to a non-convex optimization problem for which several local minima exist. Esposito and Floudas [13] developed two global branch-and-bound and convex-underestimator based optimization approaches to solve this problem. In the first approach, the dynamical system is converted into an algebraic system using orthogonal collocation on finite elements. In the second approach, state profiles are computed by integration. In Esposito and Floudas [12] a similar approach is used to solve optimal control problems.

Parameter Estimation in PDE Models

A very complex class of least squares problems are data fitting problems in partial differential equations based models. These include eigenvalue problems, as well as initial and boundary value problems and cover problems in atomic physics, elasticity, electromagnetic fields, fluid flow or heat transfer. Some recent problems are, for instance, in models describing the water balance and solid transport used to analyze the distributions of nutrients and pesticides [1], in the determination of diffusive constants in water absorption processes in hygroscopic liquids discussed in [15], or in multispecies reactive flows through porous media [38]. Such nonlinear multispecies transport models can be used to describe the interaction between oxygen, nitrate, organic carbon and bacteria in aquifers. They may include convective transport and diffusion/dispersion processes for the mobile parts (that is the mobile pore water) of the species. The immobile biophase represents the part where reactions caused by microbial activity take place and which is coupled to transport through mobile pore water. The microorganisms are assumed to be immobile. The model leads to partial differential algebraic equations

$$\begin{aligned} \mathsf{M}\partial_t \mathbf{u} - \nabla(D\nabla \mathbf{u}) + q\nabla \mathbf{u} &= \mathbf{f}_1(\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{p}) \,, \\ \partial_t \mathbf{v} &= \mathbf{f}_2(\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{p}) \,, \\ 0 &= \mathbf{g}(\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{p}) \,, \end{aligned} \tag{38}$$

where D and q denote the hydraulic parameters of the model, $\mathbf{p}$ denotes a set of reaction parameters, $\mathbf{u}$ and $\mathbf{v}$ refer to the mobile and immobile species, and $\mathbf{z}$ is related to source and sink terms.

Methodology

To solve least squares problems based on PDE models requires sophisticated numerical techniques but also great attention with respect to the quality of data and identifiability of the parameters. To solve such problems we might use the following approaches:

1. Unstructured approach: The PDE model is, for fixed parameters $\mathbf{p}$, integrated by any appropriate method yielding estimations of the observations. The parameters are adjusted by a derivative-free optimization procedure, e. g., by the Simplex method by Nelder and Mead [23]. This approach is relatively easy to implement, it solves a sequence of direct problems, and is comparable to what in Sect. "Parameter Estimation in ODE Models" has been called the IVP approach. Arning [1] uses such an approach.
2. Structured approach (for initial value PDE problems): Within the PDE model spatial coordinates and time are discretized separately. Especially for models with only one spatial coordinate, it is advantageous to apply finite difference or finite element discretizations to the spatial coordinate. The PDE system is transformed into a system of (usually stiff) ordinary differential equations. This approach is known as the *method of lines* (see, for example, [30]). It reduces parameter estimation problems subject to time-dependent partial differential equations to parameter identification problems in systems of ordinary differential equations to be integrated w.r.t. time. Now it is possible to distinguish again between the IVP and BVP approach. Schittkowski [32] in his software package `EASY-FIT` applies the method of lines to PDEs with one spatial coordinate and uses several explicit and implicit integration methods to solve the ODE system. The integration results are used by an SQP optimization routine or a Gauß–Newton method to estimate the parameters. Zieße et al. 38 and Dieses et al. [11], instead, couple the method of lines (in one and two spatial coordinates) with Bock's [6] BVP approach, discretize time, for instance, by multiple shooting and use an extended version of `PARFIT`.

The method of lines has become one of the standard approaches for solving time-dependent PDEs with only one spatial coordinate. It is based on a partial discretization, which means that only the spatial derivative is discretized but not the time derivative. This leads to a system of N coupled ordinary differential equation, where N is the number of discretization points. Let us demonstrate the method by applying it to the diffusion equation

$$\frac{\partial}{\partial t}c(t,z) = D\frac{\partial^2}{\partial z^2}c(t,z)\,, \qquad \begin{array}{l} 0 \le t < \infty \\ 0 \le z \le L \end{array} \tag{39}$$

with constant diffusion coefficient D. We discretize the spatial coordinate z according to

$$\begin{aligned} z_i &= i\Delta z\,, \quad \Delta z := \frac{L}{N}\,, \\ c_i &= c_i(t) = c(t,z_i)\,, \quad i = 0,\ldots,N\,. \end{aligned} \tag{40}$$

If we choose a finite difference approximation we get

$$\begin{aligned} \frac{\partial^2}{\partial z^2}c(t,z) &\approx \frac{c(t,z-\Delta z) - 2c(t,z) + c(t,z+\Delta z)}{(\Delta z)^2} \\ &= \frac{c_{i-1} - 2c_i + c_{i+1}}{(\Delta z)^2}\,, \end{aligned} \tag{41}$$

which replaces the diffusion Eq. (39) by N ordinary differential equations

$$\dot{c}_i(t) = \frac{c_{i-1} - 2c_i + c_{i+1}}{(\Delta z)^2}\,. \tag{42}$$

A detailed example of this method is discussed in [15]. The water transport and absorption processes within a hygroscopic liquid are described by a model containing the diffusion Eq. (39) describing the water transport within the hygroscopic liquid, a mixed Dirichlet–Neumann condition representing a flux balance equation at the surface of the liquid, and an additional integral relation describing the total amount of water in the liquid. The model included three parameters to be estimated.

The available measurement data provide the total time dependent concentration $C(t)$ of water in the liquid. A further complication was that the mathematical solution of the diffusion equation is the water concentration $c(t,z)$ in the hygroscopic liquid and it is a function of time *and* location. Therefore, in order to compare the mathematical solution with the observed data one had to integrate $c(t,z)$ over the space coordinate z, i. e., the depth of the fluid.

Least Squares Problems with Massive Data Sets

We motivate the necessity to analyze massive data sets by an example taken from astrophysics [19]. We outline the method for a huge set of millions of observed data curves in which time is the independent parameter and for each of the N, $N \simeq 10^6$, curves there is a different underlying parameter set we want to estimate by a least squares method. Note that we assume that there is a model in the sense of (6) or (10) available involving an adjustable parameter vector $\mathbf{p}$. We are further assume that we are dealing with nonlinear least squares problems which are not easy to solve. The difficulties could arise from the dependence on initial parameters, non-smoothness of the model, the number of model evalutions, or the CPU time required for one model evaluation. For each available curve we can, of course, solve this least squares problem by the techniques mentioned or discussed earlier in this article. However, the CPU time required to solve this least squares problem for several million curves is prohibitive. The archive approach described in this section is appropriate for this situation.

Examples of massive data sets subject to least squares analyses are surveys in astrophysics where millions of stars are observed over a range of time. About 50% of them are binary stars or multiple systems. The observed data could be flux of photons (just called *light* in the discipline of binary star researchers) in a certain wavelength region or radial velocity as a function of time. Thus we have to analyze millions of light and radial velocity curves. There are well validated models and methods (cf., [17] to compute such curves on well defined physical and geometrical parameters of the binary systems, e. g., the mass ratio, the ratio of their radii, their temperatures, inclination, semi-major axis and eccentricity to mention a few. Thus one is facing the problem how to analyze the surveys and to derive the stellar parameters $\mathcal{P}$ relevant to astrophysicists. In this eclipsing binary star example it suffices to consider the range $[0, P]$ for the independent parameter time because the observed curves are periodic with respect to

the period P. The period could be determined a priori from a frequency analysis of the observed curve. Under certain assumptions, in eclipsing binary star analyses, time can be replaced by phase.

The critical issues are speed and stability. Speed is obviously necessary to analyze large number of data, light and radial velocity curves in the example. Stability is required to automatize the procedure. Automatization enables the user to analyze large sets of eclipsing binary data produced by surveys. Stability and automatization need to overcome the problem of initial parameters usually experienced in nonlinear least squares. There is a price to be paid in terms of accuracy. But nevertheless, such an approach will produce good approximate results and may indicate interesting eclipsing binary stars for detailed follow-up analysis.

The method we propose to solve least squares problems with massive data sets is a matching approach: match one or several curves to a large test sets of precomputed archive curves for an appropriate set of combinations of $|\mathcal{P}|$ parameters.

The Matching Approach

Let for a given binary system ℓ^o_{ic} be any observed light value for observable c, $c = 1 \ldots C$, at phase θ_i, $i = 1, \ldots, I$. Correspondingly, ℓ^c_{ick} denotes the computed light value at the same phase θ_i for the *archive* light curve k, $k = 1 \ldots K$. Note that K easily might be a large number such as 10^{10} . Each archive light curve k is computed by a certain parameter combination.

The idea of the matching approach is to pick that light curve from the archive which matches the observed curve of binary j best. The best fit solution is obtained by linear regression. The matching approach returns, for each j, the number of the archive light curve which fits best, a scaling parameter, a, and a shift parameter, b, (which might be interpreted as a constant third light) by solving the following nested minimization problem for all j, $j = 1, \ldots, N$:

$$\min_k \left\{ \min_{a_{kc}, b_{kc}} \sum_{i=1}^{I} w_i \left[\ell^o_{ic} - \left(a_{kc} \ell^c_{ick} + b_{kc} \right) \right]^2 \right\}$$

Note that the inner minimization problem requires just to solve a linear regression problem. Thus, for each k, there exists an analytic solution for the unknown parameters a_{kc} and b_{kc}. Further note that the ℓ^c_{ick} values might be obtained by interpolation. The archive light curves are generated in such a way that they have a good covering in the eclipses while a few points will do in those parts of the light curves which show only small variation with phase. Thus, there might be a non-equidistant distribution of phase grid points. A cubic interpolation will probably suffice.

Thus, the matching approach requires us to provide the following components:

1. solving linear regression problems determining a and b for all archive curves and all observed curves (the sequence of the loops is important),
2. generating the archive curves,
3. cubic interpolation in the independent time-like quantity and interpolation after the best matching solution has been found.

In the sequel we briefly comment on the last two components.

Generating and Storing the Archive Curves As the number of archive curves can easily reach 10^{10} one should carefully think about storing them. That requires also appropriate looping over the parameters $p = 1, \ldots, |\mathcal{P}|$. For the eclipsing binary example the details are given in [19]. Among the efficiency issues is the usage of non-equidistant parameter grids exploiting the sensitivity of the parameters on the model function ℓ^c_{ic}.

One might think to store the archive light curves in a type of data base. However, data base techniques become very poor when talking about 10^{10} curves. Therefore, it is probably easier to use a flat storage scheme. In the simplest case, for each k we store the physical and geometric parameters, then those parameters describing observable c, and then the values of the observable. If we use the same number of phase values for each observable and each k, we have the same amount of data to be stored.

Exploiting Interpolation Techniques Within the matching approach interpolation can be used at two places. The first occurrence is in the regression phase. The test curves in the archive are computed for a finite grid of the independent parameter time (phase in this example). The observed curves might be observed at time values not contained in the archive. We can inter-

polate from the archive values by linear or cubic interpolation to the observed time values. However, it may well pay out to have some careful thoughts on the generation of the time grid points.

The second occurrence is when it comes to determining the best fit. The linear regression returns that parameter set which matches the observed one best. Alternatively, we could exploit several archive points to obtain a better fit to the observed curve. Interpolation in an appropriately defined neighborhoods of the best archive solution can improve the fit of the observed curve.

Numerical Efficiency The efficiency of a least squares method could be measured by the number of function or model evaluation per unknown parameter. If we assume that for each model parameter p we generate n_p archive curves in the archive, the archive contains test curves $N_c = \prod_{p=1}^{|\mathcal{P}|} n_p$ and thus requires N_c model evaluation; n_p is the number of archive grid points of parameter p.

Conclusions

This contribution outlines how to solve ODE and PDE based least squares problems. Academic and commercial least squares solvers as well as software packages are available. Massive data sets and observations arise in data mining problems, medicine, the stock market, and surveys in astrophysics. The approach described in Sect. "The Matching Approach" has been proven efficient for surveys in astrophysics. It can also support the generation of impersonal good initial parameter estimations for further analysis. The archive approach is also suitable for parameter fitting problems with non-smooth models. Another advantage is that on the archve grid it provides the global least squares minimum.

Acknowledgments

Thanks is directed to Steffen Rebennack (University of Florida, Gainesville, FL) for a careful reading of the manuscript, and Johannes P. Schlöder (IWR, Universität Heidelberg, Germany) and Gerhard Krennrich (BASF Aktiengesellschaft, Ludwigshafen) for discussions on the subject of parameter estimation.

References

1. Arning M (1994) Lösung des Inversproblems von partiellen Differentialgleichungen beim Wassertransport im Boden. Dissertation, TU Carolo-Wilhelmina zu Braunschweig
2. Baake E, Schlöder JP (1992) Modelling the Fast Fluorescence Rise of Photosynthesis. Bull Math Biol 54:999–1021
3. Bader G, Deuflhard P (1981) A Semi-Implicit Mid-Point Rule for Stiff Systems of Ordinary Differential Equations. Preprint 114, Universität Heidelberg SFB 123, Institut für Angewandte Mathematik, Heidelberg
4. Björck A (2001) Least Squares Problems. In: Floudas CA, Pardalos P (eds) Encyclopedia of Optimization. Kluwer, Dordrecht, pp 160–170
5. Bock HG (1981) Numerical Treatment of Inverse Problems in Chemical Reaction Kinetics. In: Ebert KH, Deuflhard P, Jäger W (eds) Modelling of Chemical Reaction Systems, Series in Chemical Physics. Springer, Heidelberg, pp 102–125
6. Bock HG (1987) Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen. Preprint 142, Universität Heidelberg SFB 123, Institut für Angewandte Mathematik, Heidelberg
7. Brand S (1976) Statistical and Computational Methods in Data Analysis, 2nd edn. North Holland, Amsterdam
8. Dennis JE, Schnabel RB (1983) Numerical Methods for Unconstrained Optimisation and Nonlinear Equations. Prentice Hall, Englewood Cliffs
9. Deuflhard P, Apostolescu V (1977) An Underrelaxed Gauss-Newton Method for Equality Constrained Nonlinear Least Squares Problems. In: Stoer J (ed) Proc. 8th Conf IFIP Würzburg Symposium on the Theory of Computing, number 23 in Springer Lecture Notes Control Inf. Sci. Springer, Heidelberg Berlin New York
10. Deuflhard P, Apostolescu V (1980) A Study of the Gauss–Newton Method for the Solution of Nonlinear Least Squares Problems. In: Frehse J, Pallaschke D, Trottenberg U (eds) Special Topics of Applied Mathematics. North-Holland, Amsterdam, pp 129–150
11. Dieses AE, Schlöder JP, Bock HG, Richter O (1999) Parameter Estimation for Nonlinear Transport and Degradation Processes of Xenobiotica in Soil. In: Neumann J (ed) Proceedings of the 2nd International Workshop on Scientific Computing in Chemical Engineering. Technical University Hamburg-Harburg (TUUH), Hamburg
12. Esposito WR, Floudas CA (2000) Deterministic Global Optimization in Nonlinear Optimal Control Problems. J Glob Optim 17:97–126
13. Esposito WR, Floudas CA (2000) Global Optimization for the Parameter Estimation of Differential-Algebraic Systems. Ind Eng Chem Res 39(5):1291–1310
14. Gauß CF (1809) Theoria Motus Corporum Coelestium in Sectionibus Conicus Solem Ambientium. Perthes, Hamburg
15. Kallrath J (1999) Least Squares Methods for Models Including Ordinary and Partial Differential Equations. In: Dvorak

R, Haupt HF, Wodnar K (eds) Modern Astrometry and Astrodynamics honouring Eichhorn Heinrich. Austrian Academy of Sciences, Vienna, pp 61–75
16. Kallrath J, Altstädt V, Schlöder JP, Bock HG (1999) Analysis of Crack Fatigue Growth Behaviour in Polymers and their Composites based on Ordinary Differential Equations Parameter Estimation. Polym Test 18:11–35
17. Kallrath J, Milone EF (1999) Eclipsing Binary Stars: Modeling and Analysis. Springer, New York
18. Kallrath J, Schlöder J, Bock HG (1993) Parameter Fitting in Chaotic Dynamical Systems. CMDA 56:353–371
19. Kallrath J, Wilson RE (2008) Eclipsing Binary Analysis via Light Curve Archives. ApJ, in preparation
20. Kilian C (1992) Numerische Behandlung von Ungleichheitsrestriktionen für Parameterschätzprobleme in Systemen gewöhnlicher Differentialgleichungen. Diploma thesis, Fachhochschule Darmstadt
21. Levenberg K (1944) A Method for the Solution of Certain Non-Linear Problems in Least Squares. Q Appl Math 2:164–168
22. Marquardt DW (1963) An Algorithm for Least Squares Estimation of Nonlinear Parameters. Appl SIAM J Math 11:431–441
23. Nelder JA, Mead R (1965) A Simplex Method for Function Minimization. Comput J 7:308–313
24. Ostle B (1963) Statistics in Research. Iowa State University Press, Ames
25. Polya G (1979) Vom Lernen und Lösen mathematischer Aufgaben. Einsicht und Entdeckung. Lernen und Lehren. Birkhäuser Verlag, Basel
26. Popper KR (1980) The Logic of Scientific Discovery, 10th edn. Hutchinson, London
27. Press WH, Flannery BP, Teukolsky SA, Vetterling WT (1992) Numerical Recipes The – Art of Scientific Computing, 2nd edn. Cambridge University Press, Cambridge
28. Richter O, Nörtersheuser P, Pestemer W (1992) Non-linear parameter Estimation in Pesticide Degradation. Sci Total Environ 123/124:435–450
29. Richter O, Söndgerath D (1990) Parameter Estimation in Ecology. VCH-Verlag, Weinheim
30. Schiesser WE (1991) The Numerical Methods of Lines. Academic Press, San Diego
31. Schittkowski K (1994) Parameter Estimation in Systems of Nonlinear Equations. Numer Math 68:129–142
32. Schittkowski K (1997) Parameter Estimation in One-Dimensional Time-Dependent Partial Differential Equations. Optim Method Softw 7:165–210
33. Schlöder JP (1988) Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung. Preprint 187, Bonner Mathematische Schriften, Institut für Angewandte Mathematik, Bonn
34. Schulz VH, Bock HG, Steinbach MC (1998) Exploiting Invariants in the Numerical Solution of Multipoint Boundary Value Problems for DAE. J SIAM Sci Comput 19: 440–467
35. Xu C (2001) Nonlinear Least Squares. In: Floudas CA, Pardalos P (eds) Encyclopedia of Optimization. Kluwer, Dordrecht, pp 75–80
36. Xu C (2001) Nonlinear Least Squares: Newton-type Methods. In: Floudas CA, Pardalos P (eds) Encyclopedia of Optimization. Kluwer, Dordrecht, pp 67–69
37. Xu C (2001) Nonlinear Least Squares: Trust Region Methods. In: Floudas CA, Pardalos P (eds) Encyclopedia of Optimization. Kluwer, Dordrecht, pp 80–86
38. Zieße MW, Bock HG, Gallitzendörfer JV, Schlöder JP (1996) Parameter Estimation in Multispecies Transport Reaction Systems Using Parallel Algorithms. In: Gottlieb J, DuChateau P (eds) Parameter Identification and Inverse Problems in Hydrology, Geology and Ecology. Kluwer, Dordrecht, pp 273–282

Complexity Theory

STEPHEN A. VAVASIS
Cornell University, Ithaca, USA

MSC2000: 90C60

Article Outline

Keywords
See also
References

Keywords

Complexity; Turing machine; *NP*-hard; *NP*-complete; Polynomial time; Information-based complexity; Real number model; Decision problem

Complexity theory poses the question: How much computing time is required to solve a problem, as a function of the size of the problem? A similar questions may be asked about other computing resources like memory space. In the context of optimization, the commonly-asked complexity question is how much computing time, as a function of m and n, is required to solve a certain class of mathematical programming problems with n variables and m constraints. This form of asymptotic complexity analysis was introduced by J. Hartmanis and R.E. Stearns [4].

Several different complexity theories have been developed to address this question. The best known complexity theory is based on Turing machines. Before

defining this term, we start with a definition of 'problem'. Formally, a *problem* is a function F that takes as input an *instance* and produces as output a *result*. For example, in the context of *linear programming* the instance would be a triple $(A, \mathbf{b}, \mathbf{c})$ specifying a standard-form linear program LP:

$$\begin{cases} \min & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}. \end{cases}$$

The value of $F(A, \mathbf{b}, \mathbf{c})$ is the optimal value of the LP instance, or perhaps the optimizer. The range of F must also include special output values to signify an infeasible instance, an unbounded instance, or an ill-formed instance, e. g., dimensions of A and $\mathbf{b}$ are incompatible. Thus, in the context of complexity theory, the word 'problem' refers not to a specific instance but to a class of instances.

For a Turing machine, all instances must be specified as finite-length strings of symbols where the symbols are chosen from a fixed, finite alphabet. For LP and other optimization problems, a reasonable alphabet would include the ten digits and delimiter marks like decimal points, commas, parentheses. A cardinality argument shows that this stipulation of finite string over finite alphabet precludes the consideration of problems with arbitrary real number data. Thus, Turing machine solution of linear programming is generally restricted to rational or integer data. Rational and integer data are essentially equivalent since one can transform rational to integer by multiplying by a common denominator.

A second limitation of the Turing machine model is that there is no simple way to specify a general objective function or constraint function of an optimization problem as part of the input. There is a generalization of the Turing machine definition to overcome this limitation (so-called 'oracle' Turing machines), but in this article we limit attention to conventional Turing machines. This limitation means that our Turing machine complexity analysis focuses on optimization problems with predefined classes of objective functions and constraints in which the only free parameters are numeric data, e. g., $(A, \mathbf{b}, \mathbf{c})$ in linear programming.

A *Turing machine* (TM) is a computational device equipped with an infinitely-long tape used for memory and a controller with a finite program. The tape contains an infinite number of cells numbered 0, 1, 2, …, and each cell is capable of holding one symbol chosen from a finite alphabet. The alphabet of the tape is a superset of the alphabet used for the input. Initially the tape contains the input instance written one symbol per cell starting at the left end of the tape (cell 0). The remaining cells contain a special symbol meaning 'blank'.

The Turing machine controller has a tape head that is above one cell of the tape at any particular time. The controller is always in a *state* chosen from a finite list of states. Finally, the TM obeys a finite list of *transition rules*. Each transition rule has the form: 'if the current symbol under the head is x and the current state is y, then change the symbol to x', change the state to y' and move the tape head one cell in direction d', where d is either 'left' or 'right'. Thus, a TM is fully specified by its input alphabet, its tape alphabet, its list of states, and its list of transition rules. If, for any given combination of current symbol/current state, there is at most one applicable transition rule, the TM is said to be *deterministic* else it is said to be *nondeterministic*. In this article we consider deterministic TMs only.

An *execution* of a Turing machine consists of a sequence of *moves*. Initially, as mentioned above, the input is written on the tape, the head is at position 0, and the machine is in a specially designated state called the 'start' state. The applicable transition rule is selected and executed, meaning that cell 0 is rewritten and the head is moved. Each execution of a transition rule is called a 'move'. The Turing machine continues to make moves until it reaches another special state called the 'halt' state.

The Turing machine is said to *solve* problem F, if given an input instance x, the TM (eventually) writes $F(x)$ on its tape starting at position 0, followed by blanks, and then halts. If for some input the TM could ever execute an illegal operation, e. g., move left from cell 0, or enter a state/symbol combination before halting for which there is no applicable transition rule, then it does not solve F. Furthermore, we require that the Turing machine can correctly handle every possible finite string that can be written with the input alphabet. For incorrectly formatted strings, the Turing machine should output a special string indicating incorrect formatting.

The *running time* of a Turing machine for a given input instance is the number of moves required before

it halts. The running time for the whole problem F is usually expressed as a function of the *size of the input*, that is, the number of symbols in the input string.

It can be proved using lengthy constructions that a Turing machine is capable of all the operations of an ordinary computer: it can simulate consecutively numbered memory cells each holding a separate integer or rational number that are individually addressable, it can multiply, divide, add, and subtract two such numbers, etc. For a more detailed treatment of Turing machines, see [5].

A Turing machine is said to solve problem F in *polynomial time* if its running time is no more than a polynomial function of the size of the input. Examples of optimization problems that can be solved in polynomial time include linear and convex quadratic programming.

A *decision problem* is a problem F in which the range of F consists of just two entries, 'YES' and 'NO'. Optimization problems can often be recast as decision problems. For instance, in the case of linear programming, the input instance consists of $(A, \mathbf{b}, \mathbf{c}, r)$, where r is a rational number, and the TM outputs 'YES' if the minimal solution to the LP problem is r or less, else it outputs 'NO'. For incorrectly formatted and infeasible problems, the TM also outputs 'NO'. The decision problem F partitions the input space into two sets of strings, 'YES'-instances and 'NO'-instances. A synonym for 'decision problem' is *language recognition problem*.

The set $\mathcal{P}$ is defined to be all decision problems that can be solved in polynomial time. This set includes linear programming (as recast in the previous paragraph) and many combinatorial optimization problems such as the minimum spanning tree problem and the shortest path problem (cf. also ► Shortest path tree algorithms).

Many interesting problems, such as *nonconvex quadratic programming* and *Boolean satisfiability*, are not known to be in $\mathcal{P}$, but are also not proven to lie outside of $\mathcal{P}$. To analyze these cases, we introduce a second complexity class called *NP*. A decision problem F is said to lie in *NP* if there exists a polynomial time 'certificate-checking' machine M outputting 'YES' or 'NO', and polynomials $p(\cdot)$, $q(\cdot)$ with the following properties. For every 'YES'-instance x of F, there exists another string y, called the *certificate* of x, such that the size of y is no more than $p(\text{size}(x))$ such that the pair (x, y) (i. e., the string concatenation of x and y properly delimited) is a 'YES'-instance of M. On the other hand, for every 'NO'-instance x of F, and for every possible certificate y, M outputs 'NO' for the input pair (x, y). Finally, in both cases, M is required to run in time no more than $q(\text{size}(x) + \text{size}(y))$.

Notice this definition is asymmetric between 'YES'- and 'NO'-instances. Thus, it is not necessarily true that if problem F is in *NP*, then the problem $\overline{F}$ that results from complementing F's output (i. e., 'YES'-instances of F are 'NO'-instances of $\overline{F}$ and vice-versa) is still in *NP*. Indeed, the question of whether $F \in NP \Leftrightarrow \overline{F} \in NP$ is a well-known open question.

Another observation from the above definition is that $\mathcal{P} \subset NP$. In particular, if a decision problem F lies in $\mathcal{P}$, then it has polynomial time Turing machine T that distinguishes 'YES'-instances from 'NO'-instances. In this case, it is simple to design the certificate checking machine M needed for the definition of *NP*: in particular, M takes as input (x, y), it discards y, i. e., overwrites it with blank cells, and then switches to running T on x.

The most famous open question in complexity theory is whether this containment is actually equality, i. e., whether $\mathcal{P} = NP$. It turns out that the $\mathcal{P} = NP$ question hinges on *NP-complete problem* the prototype of which is the satisfiability problem. The *satisfiability problem* is as follows. An instance is a Boolean formula with variables $x_1, \ldots, x_m$, conjunctions, disjunctions and complement operations. For example, $x_1 \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (x_2 \vee x_3)$ is a satisfiability instance. The decision problem is to determine whether there is an *assignment* of the variables, each one either 'TRUE' or 'FALSE', to make the entire formula true following the usual laws of boolean algebra. For example, the preceding formula is a 'YES'-instance because there is a satisfying assignment, namely x_1 = 'TRUE', x_2 = 'FALSE', x_3 = 'TRUE'. It is easy to see that this problem is in *NP*: the certificate for a 'YES'-instance is the satisfying assignment. The certificate-checking machine M substitutes the satisfying assignment into the formula and verifies that the formula evaluates as 'TRUE'. Thus, every satisfiable formula has a certificate, but every nonsatisfiable instance is rejected by M no matter what certificate is given.

S.A. Cook [2] proved that every problem in *NP* is polynomially transformable to satisfiability. We say that

decision problem F is *polynomially transformable* to F' if there exists a Turing machine T that takes as input an instance x of F and produces as output an instance x' of F', such that the running time of T is no more than a polynomial in the size of the input, and such that x is a 'YES'-instance of F if and only if x' is a 'YES'-instance of F'.

Cook's result means that given any decision problem F in *NP*, there is a Turing machine M depending on F that takes as input an instance x of F and proceeds as follows. In polynomial time, M constructs a Boolean formula x' from x such that x' is satisfiable if and only x is a 'YES'-instance of F. The construction of M is as as follows. The Boolean formula x' simulates the action of the certificate-checking machine in on x. The actual entries of the certificate are represented by unknown Boolean variables, as are the entries on the tape of M after the first move. The formula is composed of clauses that require the Turing machine to obey all transition rules and to end up halting with 'YES' as the output.

Thus, Cook's theorem implies that if there were a polynomial time algorithm for satisfiability, then there would be a polynomial time algorithm for every other problem in *NP*. Thus, the famous open question 'is $\mathcal{P} = NP$?' is now reduced to the (apparently) simpler question 'is there a polynomial time algorithm for the satisfiability problem?'

It is not yet (1999) known whether the answer to either question in the last paragraph is 'yes'. But many in the field suspect that the answer is 'no', i. e., many believe that there is no polynomial time algorithm for satisfiability. If indeed it is proved some day that no such polynomial time algorithm exists, we would say that satisfiability is *intractable*.

A decision problem F is said to be *NP-complete* if it has these two properties, namely

1) F is in *NP*; and
2) every problem in *NP* can be polynomially transformed to F.

Cook's result can be restated as: satisfiability is *NP*-complete. Furthermore, since polynomial transformations can be composed, any problem F in *NP* to which any known *NP*-complete problem F' can be transformed must itself be *NP*-complete. After Cook's result was announced, R.M. Karp [6] showed that many well-known combinatorial problems, such as the *Hamiltonian cycle problem* ('given an undirected graph, is there a cycle containing each vertex exactly once?') and the *max-clique problem* ('given an undirected graph and an integer k, is there a set of k vertices that are all mutually connected by edges?'), are *NP*-complete. By 1979, already thousands of problems were known to be *NP*-complete and many were catalogued in [3]. A proof that a problem is *NP*-complete is regarded as strong evidence of the problem's intractability.

Although the first batch of *NP*-completeness proofs applied to combinatorial problems, many continuous optimization problems are also known to be *NP*-complete; see ► *NP*-complete problems and proof methodology.

A generalization of '*NP*-complete' is the notion of '*NP*-hard'. A problem F is said to be *NP-hard* if satisfiability (or any other *NP*-complete problem) can be polynomially transformed to F. Thus, an *NP*-hard problem does not necessarily lie in *NP*. Indeed, the term *NP*-hard is often used to describe problems that are not even decision problems.

The Turing machine is not the only model of complexity used in the literature. In fact, some feel that the TM is inadequate for modeling continuous optimization problems. Most continuous optimization problem are based on computation with real numbers, but true real number computation is not possible with a TM. One model of real number computation is the *information-based model*. In this model, an algorithm is composed of operations on real numbers. Operations on real numbers are often counted as cost-free in this model, and the only costly operation is the evaluation of the functions defining the objective and constraints of the optimization problem. The objective functions and constraints are considered external black-box subroutines that take as input a vector and return as output the value of the function and possibly derivative values. In information-based complexity, a parameter $\epsilon > 0$ that specifies the desired degree of accuracy in the solution is always part of the input, since the information-based model rarely permits any problem to be solved exactly. This information-based model was used to analyze the *ellipsoid method* by its inventors D.B. Yudin and A.S. Nemirovsky [7]. It has also been used to analyze complexity of *local optimization* by S.A. Vavasis [12]. This model has also been used extensively to analyze other numerical algorithms not related to optimization, e. g., quadrature and other linear problems [9].

A second model of computation is a *real number model* in which each operation is unit cost, and in which there is no concept of external black-box function evaluation. In this model it is possible to develop real number analogs of complexity classes $\mathcal{P}$ and *NP*, and also a reasonable definition for *NP*-complete; see [1]. This model can be used to analyze linear programming and other problems specified by a finite number of real parameters. The complexity of linear programming problem in this model is not fully understood (see [13] and [10]).

For a more detailed look at complexity in optimization up to 1991, see [11]. For a more recent collection of papers on this topic, see [8].

See also

- ▶ Complexity Classes in Optimization
- ▶ Complexity of Degeneracy
- ▶ Complexity of Gradients, Jacobians, and Hessians
- ▶ Complexity Theory: Quadratic Programming
- ▶ Computational Complexity Theory
- ▶ Fractional Combinatorial Optimization
- ▶ Information-Based Complexity and Information-Based Optimization
- ▶ Kolmogorov Complexity
- ▶ Mixed Integer Nonlinear Programming
- ▶ *NP*-Complete Problems and Proof Methodology
- ▶ Parallel Computing: Complexity Classes

References

1. Blum L, Cucker F, Shub M, Smale S (1998) Complexity and real computation. Springer, Berlin
2. Cook SA (1971) The complexity of theorem-proving procedures. In: Proc 3rd Annual ACM Symp Theory of Computing. ACM, pp 151–158
3. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, New York
4. Hartmanis J, Stearns RE (1965) On the computational complexity of algorithms. Trans Amer Math Soc 117:285–306
5. Hopcroft J, Ullman J (1979) Introduction to automata theory, languages and computation. Addison-Wesley, Reading, MA
6. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations. Plenum, New York, pp 85–103
7. Nemirovsky AS, Yudin DB (1983) Problem complexity and method efficiency in optimization. Wiley, New York
8. Pardalos PM (1993) Complexity in numerical optimization. World Sci, Singapore
9. Traub JF, Wasilkowski GW, Woźniakowski H (1988) Information-based complexity. Acad Press, New York
10. Traub JF, Woźniakowski H (1982) Complexity of linear programming. Oper Res Lett 1:59–62
11. Vavasis SA (1991) Nonlinear optimization: Complexity issues. Oxford Univ. Press, Oxford
12. Vavasis SA (1993) Black-box complexity of local minimization. SIAM J Optim 3:60–80
13. Vavasis SA, Ye Y (1996) A primal-dual interior point method whose running time depends only on the constraint matrix. Math Program 74:79–120

Complexity Theory: Quadratic Programming

STEPHEN A. VAVASIS
Cornell University, Ithaca, USA

MSC2000: 90C60

Article Outline

Keywords
See also
References

Keywords

Quadratic programming; Complexity; *NP*-hard; *NP*-complete; Local minimization; Trust region problem; Polynomial time; Strongly polynomial time; Knapsack problem; Simplicial constraints; Box constraints; Ellipsoid method; Interior point methods; Approximation algorithms

Nowhere in optimization is the dichotomy between convex and nonconvex programming more apparent than in complexity issues for quadratic programming. *Quadratic programming*, abbreviated QP, refers to minimizing a quadratic function $q(\mathbf{x}) = \mathbf{x}^\top H\mathbf{x}/2 + \mathbf{c}^\top \mathbf{x}$ subject to linear constraints $A\mathbf{x} \geq \mathbf{b}$. The problem is thus specified by the four-tuple $(H, A, \mathbf{b}, \mathbf{c})$ where H is a symmetric $n \times n$ matrix, A is an $m \times n$ matrix, $\mathbf{b}$ is an m-vector and $\mathbf{c}$ is an n-vector. Minimizing a quadratic function subject to convex quadratic constraints is also an interesting problem and is considered at the end of this article. The quadratic function $q(\mathbf{x})$ is said to be

convex if the matrix H is positive semidefinite. A special case of a convex function is when $H = 0$, in which case the problem is now called *linear programming*.

Convex quadratic programming inherits all the desirable attributes of the general convex programming. In particular, there is no local minimizer other than the global minimizer(s). Furthermore, general convex programming techniques like the *ellipsoid method* and *interior point methods* can be applied.

With either the ellipsoid or interior point method, convex quadratic programming can be solved in *polynomial time*. In more detail, assume that (H, A, $\mathbf{b}$, $\mathbf{c}$) contain all integer data so that the problem is finitely represented for a *Turing machine* (see ► Complexity theory). Let L denote the length of the input data, that is, the total number of digits to write (H, A, $\mathbf{b}$, $\mathbf{c}$). Assume $L \geq m^2$ since H has m^2 entries. Then M.K. Kozlov et al. [9] showed that the ellipsoid algorithm of A.S. Nemirovsky and D.B. Yudin [14] can solve a convex QP instance in time $O(m^2L)$ iterations, where each iteration requires $O(m^2)$ arithmetic operations on integers, each of which has at most $O(L)$ digits. This result built on an analogous result for the LP case by L.G. Hačijan (also spelled L.G. Khachiyan) [4]. Thus, the total running time of this algorithm is polynomial in the size of the input. Note that the the global minimizer for quadratic programming (either convex or nonconvex), if it exists, can be written down with $O(nL)$ digits, and hence computing the true global minimizer in a Turing machine setting is possible.

Later, S. Kapoor and P.M. Vaidya [6] and Y. Ye and E. Tse [29] proved that an interior point method can solve convex quadratic programming in polynomial time under similar assumptions. This result built on the earlier result for the LP case by N.K. Karmarkar [7]. The best known running time for an interior point method for convex QP is $O(m^{1/2}L)$ iterations, each iteration requiring $O(m^3)$ arithmetic operations on integers each of which has at most $O(L)$ digits and is based on work by J. Renegar [17].

The running times of both the ellipsoid and interior point algorithms are 'weakly' polynomial, meaning that the number of arithmetic operations is bounded by a polynomial in L rather than by a polynomial in m and n. In contrast, polynomial time algorithms for other problems like solving a system of linear equations or finding a minimum flow a network are *strongly polynomial time*, meaning that the number of operations is bounded by a polynomial in the combinatorial dimension of the input data. A well-known open (1999) question asks whether there is a strongly polynomial time algorithm for convex QP (or, more specifically, for LP). A strongly polynomial algorithm would involve a number of arithmetic operations polynomially bounded in m, n, in which each operation involves integers with a number of digits bounded by a polynomial in L. Some progress related to this question is as follows. If the dimension n is restricted to a small integer, then QP can be solved in time linear in m. This result is due to I. Adler and R. Shamir [1] and builds on [10] and [2]. Since the constant of proportionality (or perhaps an additive term) is exponential in n, this algorithm is not so useful except when $n \ll m$. An example would be quadratic programming arising from a geometric problem, such as finding the point in a 3D polyhedron closest to the origin.

In the case of linear programming, a modified ellipsoid algorithm has a number of operations depending only on L_A, where L_A is the number of digits in A (i. e., the number of operations no longer depends on $\mathbf{b}$ or $\mathbf{c}$), a result due to É. Tardos [19] and extended in [25]. Finally, some special cases of quadratic programming are known to be solvable in strongly polynomial time such as the *convex quadratic knapsack problem* [5] which is:

$$\begin{cases} \min & q_1(x_1) + \cdots + q_n(x_n) \\ \text{s.t.} & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\ & \mathbf{b}^\top \mathbf{x} = \gamma. \end{cases} \tag{1}$$

Here $q_1, \ldots, q_n$ are convex quadratic functions of one variables (each specified by a quadratic and a linear coefficient) $\mathbf{l}$, $\mathbf{u}$, $\mathbf{b}$, γ are also part of the problem data.

Nonconvex quadratic programming is much harder than convex quadratic programming. If H is not positive semidefinite, then the QP instance is said to be *nonconvex*. A special case of nonconvex problems is when H is negative semidefinite, in which case the problem is said to be *concave quadratic programming*. When H is neither positive nor negative semidefinite, the problem is *indefinite*. Nonconvex quadratic programming was shown to be *NP*-hard by S. Sahni [18]. If the problem is posed as a decision problem, then it lies in *NP* (and is therefore *NP*-complete), a result due to S.A. Vavasis [20]. (See ► Complexity theory or ► *NP*-complete

problems and proof methodology for the definitions of *NP*-complete and *NP*-hard.) Even the problem of finding a *local minimizer* is known to be *NP*-hard, a result due to K.G. Murty and S.N. Kabadi [13].

Many restricted versions of the problem are still *NP*-hard. The nonconvex quadratic knapsack problem, that is, (1) with general (not necessarily convex) quadratic functions $q_1, \ldots, q_n$, is *NP*-hard [18]. QP with only *box constraints*, that is, minimize $q(\mathbf{x})$ subject to $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$, is also *NP*-hard. Similarly, minimizing $q(\mathbf{x})$ subject to *simplicial constraints*, that is, constraints $\mathbf{x} \geq \mathbf{0}$ and $x_1 + \cdots + x_n = 1$, is *NP*-hard as proved in [15] using a theorem of T.S. Motzkin and E.G. Straus [12]. The simplicial case is interesting because minimizing either a concave or convex quadratic function on a simplex can be solved in polynomial time. P.M. Pardalos and Vavasis [16] showed that quadratic programming in which H has a single negative eigenvalue (i. e., H is 'almost' positive semidefinite) is *NP*-hard.

The hardness results have motivated a search for approximation algorithms to nonconvex quadratic programming problems. Vavasis [22,24] proposed approximation algorithms for concave and indefinite QP in which the complexity depends exponentially on the number of negative eigenvalues of H. An additional result is a fully polynomial time approximation scheme for the indefinite knapsack problem. Ye [28] gave a constant-factor polynomial time approximation scheme for indefinite qua- dratic programming with box constraints.

Because computing even a local minimum of a quadratic programming instance is hard, several researchers have looked at approximations and special cases for the local minimization problem. J.J. Moré and Vavasis [11] proved that a local minimizer for the concave knapsack problem can be found in polynomial time; this result was extended to the indefinite case in [23]. Ye [28] gave a polynomial time algorithm to find an approximate *KKT point* of general nonconvex QP.

So far we have considered only linear constraints. A *convex quadratic constraint*, also called an *ellipsoidal constraint*, is a constraint of the form $(\mathbf{x} - \mathbf{c})^\top A(\mathbf{x} - \mathbf{c}) \leq 1$, where A is a symmetric positive semidefinite matrix. The problem of minimizing a nonconvex quadratic function subject to a single ellipsoidal constraint is called the *trust region problem* and has received extensive attention in the literature because algorithms to solve this problem are often used as subroutines by general-purpose optimization algorithms. A polynomial time algorithm for the trust region problem was proposed independently by Ye [27] and Karmarkar [8]. The sense in which this algorithm is 'polynomial time' is weaker than the analogous claim for QP because in the trust region case, the optimizer $\mathbf{x}$ cannot be written in a finite number of digits even if the input data is all integer (because the solution may be irrational). But Vavasis and R. Zippel [26] showed nonetheless that this algorithm leads to a proof that the associated decision problem lies in $\mathcal{P}$. The trust region problem is thus one of the very few nonconvex optimization problem solvable in polynomial time. M. Fu, Z.-Q. Luo and Ye [3] have considered generalizing this result to more than one ellipsoidal constraint, although the results are not as strong as the single-constraint case.

All of the pre-1991 material in this article is covered in more depth by [21].

See also

- Complexity Classes in Optimization
- Complexity of Degeneracy
- Complexity of Gradients, Jacobians, and Hessians
- Complexity Theory
- Computational Complexity Theory
- Fractional Combinatorial Optimization
- Information-Based Complexity and Information-Based Optimization
- Kolmogorov Complexity
- Mixed Integer Nonlinear Programming
- *NP*-Complete Problems and Proof Methodology
- Parallel Computing: Complexity Classes
- Quadratic Assignment Problem
- Quadratic Fractional Programming: Dinkelbach Method
- Quadratic Knapsack
- Quadratic Programming with Bound Constraints
- Quadratic Programming Over an Ellipsoid
- Standard Quadratic Optimization Problems: Algorithms
- Standard Quadratic Optimization Problems: Applications
- Standard Quadratic Optimization Problems: Theory

References

1. Adler I, Shamir R (1993) A randomization scheme for speeding up algorithms for linear and convex quadratic programming problems with a high constraints-to-variables ratio. Math Program 61:39–52
2. Clarkson KL (1995) Las Vegas algorithms for linear and integer programming when the dimension is small. J ACM 42(2):488–499
3. Fu M, Luo Z-Q, Ye Y (1996) Approximation algorithms for quadratic programming. Working Paper Dept. Management Sci Univ Iowa
4. Hačijan LG (1979) A polynomial algorithm in linear programming. Soviet Math Dokl 20:191–194. (Dokl. Akad. Nauk SSSR 244 (1979), 1093–1096)
5. Helgason R, Kennington J, Lall H (1980) A polynomially bounded algorithm for a singly constrained quadratic program. Math Program 18:338–343
6. Kapoor S, Vaidya PM (1986) Fast algorithms for convex quadratic programming and multicommodity flows. In: Proc. 18th Annual ACM Symp. Theory of Computing. ACM, pp 147–159
7. Karmarkar N (1984) A new polynomial-time algorithm for linear programming. Combinatorica 4:373–395
8. Karmarkar N (1989) An interior-point approach to NP-complete problems. Preprint
9. Kozlov MK, Tarasov SP, Hačijan LG (1979) Polynomial solvability of convex quadratic programming. Soviet Math Dokl 20:1108–1111 (Dokl Akad Nauk SSSR 248 (1979), 1049–1051)
10. Megiddo N (1984) Linear programming in linear time when the dimension is fixed. J ACM 31:114–127
11. Moré JJ, Vavasis SA (1991) On the solution of concave knapsack problems. Math Program 49:397–411
12. Motzkin TS, Straus EG (1965) Maxima for graphs and a new proof of a theorem of Turán. Canad J Math 17: 553–540
13. Murty KG, Kabadi SN (1987) Some NP-complete problems in quadratic and nonlinear programming. Math Program 39:117–123
14. Nemirovsky AS, Yudin DB (1983) Problem complexity and method efficiency in optimization. Wiley, New York
15. Pardalos PM, Han C, Ye Y (1991) Algorithms for the solution of quadratic knapsack problems. Linear Alg Appl 152: 69–91
16. Pardalos PM, Vavasis SA (1991) Quadratic programming with one negative eigenvalue is NP-hard. J Global Optim 1:15–22
17. Renegar J (1988) A polynomial-time algorithm based on Newton's method for linear programming. Math Program 40:59–94
18. Sahni S (1974) Computationally related problems. SIAM J Comput 3:262–279
19. Tardos E (1986) A strongly polynomial algorithm to solve combinatorial linear programs. Oper Res 34:250–256
20. Vavasis SA (1990) Quadratic programming is in NP. Inform Process Lett 36:73–77
21. Vavasis SA (1991) Nonlinear optimization: Complexity issues. Oxford Univ. Press, Oxford
22. Vavasis SA (1992) Approximation algorithms for indefinite quadratic programming. Math Program 57:279–311
23. Vavasis SA (1992) Local minima for indefinite quadratic knapsack problems. Math Program 54:127–153
24. Vavasis SA (1992) On approximation algorithms for concave quadratic programming. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton Univ. Press, Princeton, pp 3–18
25. Vavasis SA, Ye Y (1996) A primal-dual interior point method whose running time depends only on the constraint matrix. Math Program 74:79–120
26. Vavasis SA, Zippel R (1990) Proving polynomial-time for sphere-constrained quadratic programming. Techn. Report Dept Computer Sci. Cornell Univ. 90-1182
27. Ye Y (1992) On affine scaling algorithms for nonconvex quadratic programming. Math Program 56:285–300
28. Ye Y (1997) Approximating quadratic programming with bound constraints. Working Paper Dept Management Sci Univ Iowa
29. Ye Y, Tse E (1989) An extension of Karmarkar's projective algorithm for convex quadratic programming. Math Program 44:157–179

Composite Nonsmooth Optimization

CNSO

V. Jeyakumar
School of Math., University Sydney,
Sydney, Australia

MSC2000: 46A20, 90C30, 52A01

Article Outline

Keywords
Real-Valued CNSO
Extended Real-Valued CNSO
Multi-Objective CNSO
See also
References

Keywords

Nonsmooth analysis; Convex composite programming; Optimality conditions; Nonsmooth optimization; Vector optimization

By composite nonsmooth optimization (CNSO) we mean a class of optimization problems involving composite functions of the form $f(x) := g(F(x))$, where F: $\mathbf{R}^n \to \mathbf{R}^m$ is a (differentiable) smooth map and g: $\mathbf{R}^m \to \mathbf{R}$ is a nonsmooth function. The function g is often a nonsmooth convex function. Problems of CNSO occur when solving nonlinear equations $F_i(x) = 0, i = 1, \ldots, m$, by minimizing the norm $\| F(x) \|$. Similar problems arise when finding a feasible point of a system of nonlinear inequalities $F_i(x) \leq 0, i = 1, \ldots, m$, by minimizing $\| F(x)^+ \|$ where $F_i^+ = \max(F_i, 0)$. Composite functions f also appear in the form of an exact penalty function when solving a nonlinear programming problem. Another type of CNSO problem which frequently arises in (electrical) engineering [4] is to minimize the *max-function* $\max_i F_i(x)$, where the maximum is taken over some (finite) set. All these examples can be cast within the structure of CNSO. Moreover, CNSO provides a unified framework in which to study theoretical properties and convergence behavior of various numerical methods for constrained optimization problems. There have been many contributors to the study of CNSO problems both in finite and infinite dimensions. (See for example [2,6,7,10,11,13,15,16,19].) In this article we only discuss different forms of composite model problems in finite dimensions and provide a brief account of their first and *second order Lagrangian theory of CNSO problems*. The implications for numerical optimization are not discussed here. For details on this see, for instance, [1,6].

Real-Valued CNSO

Consider the problem

$$(\mathrm{P}) \quad \min_{x \in \mathbb{R}^n} g(F(x)).$$

Notably, A.D. Ioffe [7,8,9] provided the theoretical foundation for CNSO problems in the case where the function g is sublinear (convex plus positively homogeneous). Then J.V. Burke [2] extended the theory to the case where g is convex. A fundamental local dualization technique plays a significant role in the development of first – and second order *Lagrangian theory* for (P). To see the dualization result, let us define the *Lagrangian* of (P) as

$$L(x, y^*) = \langle y^*, F(x) \rangle - g^*(y^*),$$

where g^* is the Fenchel conjugate of g [14]. Let

$$L_0(z) = \{ y^* : \ y^* \in \partial g(F(z)), \ y^* F'(z) = 0 \}$$

and let

$$L_{\eta\epsilon}(z) = \{ y^* : \ y^* \in \partial_\epsilon g(F(z)), \ \| y^* F'(z) \| \leq \eta \},$$

where $F'(z)$ is the derivative of F at z, $\partial g(y)$ is the *convex subdifferential* of g at y, $\epsilon > 0$, $\eta > 0$ and $\partial_\epsilon g(F(z))$ is the *ϵ-subdifferential* of g at $F(z)$. The set $L_0(z)$ is the set of Lagrange multipliers for (P) at z (see [2,11]). Define

$$\phi_{\eta\epsilon}(x) := \max_{y^* \in L_{\eta\epsilon}(z)} L(x, y^*).$$

A general form of the *Ioffe–Burke local dualization* result [11] states that if g is a lower semicontinuous convex function and F is a locally Lipschitzian and (Gâteaux) differentiable function then the following statements are equivalent:

i) $g(F(x))$ attains a local minimum at z.
ii) $L_0(z) \neq \emptyset$ and $\phi_{\eta\epsilon}$ attains a local minimum at z, for *any* $\eta > 0, \epsilon > 0$.
iii) $L_0(z) \neq \emptyset$ and $\phi_{\eta\epsilon}$ attains a local minimum at z, for *some* $\eta > 0, \epsilon > 0$.

These conditions also provide first order Lagrangian conditions for (P). Moreover, this local *dualization* result and a generalized Taylor expansion of V. Jeyakumar and X.Q. Yang [11] yield second order optimality conditions for (P). If g is a lower semicontinuous convex function and F is a differentiable map with locally Lipschitzian derivative F' (i. e. $C^{1,1}$) then a necessary condition for $a \in \mathbf{R}^n$ to be a local minimizer of (P) is

$$\max_{y^* \in L_0(a)} L^{\circ\circ}(a, y^*; u, u) \geq 0, \quad \forall u \in \overline{K(a)}.$$

On the other hand if $a \in \mathbf{R}^n$, $L_0(a) \neq \emptyset$ and

$$\max_{y^* \in L_0(a)} -L^{\circ\circ}(a, y^*; u, -u) > 0, \quad \forall u \in D(a),$$

then a is a strict local minimizer of order 2 for (P), i. e., there exist $\epsilon > 0$, $\rho > 0$ such that whenever $\| x - a \| < \rho, f(x) \geq f(a) + \epsilon \| x - a \|^2$. Here

$$K(a) = \left\{ u \in \mathbb{R}^n : \begin{array}{c} \exists t > 0, \\ g(F(a) + tF'(a)u) \\ \leq g(F(a)) \end{array} \right\},$$

$D(a) = \{u \in \mathbf{R}^n : f'(a;u) \le 0\}$, and the directional derivative of f at a is given by $f'(a;d) = g'(F(a);F'(a)d)$. The *generalized second order directional derivative* of L at a in the directions $(u,v) \in \mathbf{R}^n \times \mathbf{R}^n$, $L^{\circ\circ}(a;u,v)$, is defined by

$$\limsup_{y \to a,\, s \to 0} \frac{\langle \nabla L(y+su), v\rangle - \langle \nabla L(y), v\rangle}{s}.$$

Special cases of these optimality conditions under twice continuously differentiability hypothesis can be found in [2,9]. Composite problems where the map F is $C^{1,1}$, but is not necessarily twice continuously differentiable are discussed in [19].

Extended Real-Valued CNSO

A composite problem form which has greater versatility than the traditional form (P) is the following nonfinite valued problem [15,16]

$$(\text{PE}) \quad \begin{cases} \min & g(F(x)) \\ \text{s.t.} & x \in \mathbb{R}^n, \\ & F(x) \in \operatorname{dom}(g), \end{cases}$$

where $g: \mathbf{R}^m \to \mathbf{R} \cup \{+\infty\}$ is a convex function and $F: \mathbf{R}^n \to \mathbf{R}^m$ is a smooth map. For instance, constrained CNSO problems of the form,

$$\begin{cases} \min & g_0(F_0(x)) \\ \text{s.t.} & x \in C, \\ & g_j(F_j(x)) \le 0, \quad j = 1, \ldots, m, \end{cases}$$

studied in [10,17], can be re-written in the form of (PE) [11]. Here C is a closed convex subset of $\mathbf{R}^n$, g_j, $j = 0, \ldots, m$, are locally Lipschitz functions and F_j, $j = 0, \ldots, m$, are differentiable functions. Optimality conditions for (PE) can be derived by reducing (PE) to a real-valued minimization problem as it was shown in [3]. This requires a regularity condition known as a constraint qualification in the nonlinear programming literature. The following regularity condition, introduced in [15] as a *basic constraint qualification*, permits one to establish a reduction theorem. If $g: \mathbf{R}^m \to \mathbf{R} \cup \{+\infty\}$ is a lower semicontinuous convex function and if $F: \mathbf{R}^n \to \mathbf{R}^m$ is locally Lipschitzian then the function $f(x) := g(F(x))$ is said to satisfy the basic constraint qualification at a point $x \in \operatorname{dom}(f)$ if the only point $w \in N(F(x)|\operatorname{dom}(g))$ for which $0 \in w^{\mathsf{T}}\, \partial F(x)$ is $w = 0$, where $N(F(x)|\operatorname{dom}(g))$ is the normal cone to $\operatorname{dom}(g)$ at $F(x)$ and $\partial F(x)$ is the generalized Jacobian of F at x [5]. The basic constraint qualification is equivalent to the *Mangasarian–Fromovitz constraint qualification* for the standard nonlinear programming problem with inequality and equality constraints (see [15]). The *Burke–Poliquin reduction* result gives us the following second order conditions for (PE). For problem (PE), suppose that $F(a) \in \operatorname{dom}(g)$, g is lower semicontinuous convex and F is $C^{1,1}$. Then the following statements (i) and (ii) hold.

i) If a is a local minimizer of (PE) at which the basic constraint qualification holds, then

$$\max_{y^* \in L_0(a)} L^{\circ\circ}(a, y^*; u, u) \ge 0, \quad \forall u \in \overline{K(a)}.$$

ii) If $L_0(a) \ne \emptyset$ and

$$\max_{y^* \in L_0(a)} -L^{\circ\circ}(a, y^*; u, -u) > 0, \ \forall u \in D(a),$$

then a is a strict local minimizer of order 2 for (PE). With the aid of a representation condition, second order conditions can also be obtained for a *global minimizer* of (PE) in the case where F is twice strictly differentiable. This was shown in [19]. The problems (PE) have also been extensively studied by R.T. Rockafellar [15,16] in the case where F is twice continuously differentiable and g is a proper convex function that is *piecewise linear quadratic* in the sense that the $\operatorname{dom}(g)$ is expressible as the union of finitely many polyhedral sets, relative to each of which g is given by the formula that is quadratic (or affine).

Multi-Objective CNSO

Nonsmooth vector optimization problems (cf. ▶ Vector optimization) where the functions involved are compositions of convex functions and smooth functions arise in various applications. The following model problem was examined in [12]:

$$(\text{MP}) \quad \begin{cases} \text{V} - \min & (f_1(F_1(x)), \ldots, f_p(F_p(x))) \\ \text{s.t.} & x \in C, \\ & g_j(G_j(x)) \le 0, \\ & \quad j = 1, \ldots, m, \end{cases}$$

where C is a convex subset of $\mathbf{R}^n$, f_i, g_j are real valued convex functions on $\mathbf{R}^n$, F_i, G_j are locally Lips-

chitz and differentiable functions from $\mathbf{R}^n$ into $\mathbf{R}^n$. Note here that 'V-min' stands for vector minimization. This model is broad and flexible enough to cover many common types of vector optimization problems. In particular, this model includes the penalty representation of the standard vector nonlinear programming problems, examined in [18], and many vector approximation problems. By employing the Clarke subdifferential, first order Lagrangian optimality and duality results can be discussed as it was shown in [12]. Second order optimality conditions for a special case of the problem (*MP*) are discussed in [19,21]

See also

- ► Nonconvex-Nonsmooth Calculus of Variations
- ► Nonsmooth and Smoothing Methods for Nonlinear Complementarity Problems and Variational Inequalities
- ► Solving Hemivariational Inequalities by Nonsmooth Optimization Methods

References

1. Burke JV (1985) Descent methods for composite nondifferentiable optimization problems. Math Program 33:260–279
2. Burke JV (1987) Second-order necessary and sufficient conditions for convex composite NDO. Math Program 38:287–302
3. Burke JV, Poliquin RA (1992) Optimality conditions for nonfinite valued convex composite functions. Math Program B 57:103–120
4. Charalambous C (1979) Acceleration of the least pth- algorithm for minimax optimization with engineering applications. Math Program 17:270–297
5. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
6. Fletcher R (1987) Practical methods of optimization. Wiley, New York
7. Ioffe AD (1979) Necessary and sufficient conditions for a local minimum, 1: A reduction theorem and first-order conditions. SIAM J Control Optim 17:245–250
8. Ioffe AD (1979) Necessary and sufficient conditions for a local minimum, 2: Conditions of Levitin–Miljutin–Osmolovskii type. SIAM J Control Optim 17:251–265
9. Ioffe AD (1979) Necessary and sufficient conditions for a local minimum, 3: second order conditions and augmented duality. SIAM J Control Optim 17:266–288
10. Jeyakumar V (1991) Composite nonsmooth programming with Gâteaux differentiability. SIAM J Optim 1:30–41
11. Jeyakumar V, Yang XQ (1993) Convex composite multi-objective nonsmooth programming. Math Program 59:325–343
12. Jeyakumar V, Yang XQ (1995) Convex composite minimization with C1, 1 functions. J Optim Th Appl 86:631–648
13. Penot JP (1994) Optimality conditions in mathematical programming and composite optimization. Math Program 67:225–245
14. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton
15. Rockafellar RT (1988) First- and second-order epi-differentiability in nonlinear programming. Trans Amer math Soc 307:75–108
16. Rockafellar RT (1989) Second-order optimality conditions in nonlinear programming obtained by way of epi-derivatives. Math Oper Res 14:462–484
17. Studniarski M, Jeyakumar V (1995) A generalized mean-value theorem and optimality conditions in composite nonsmooth minimization. Nonlinear Anal Th Methods Appl 24:883–894
18. White DJ (1984) Multi-objective programming and penalty functions. J Optim Th Appl 43:583–599
19. Yang XQ (1994) Generalized second-order directional derivatives and optimality conditions. PhD Thesis Univ. New South Wales, Australia
20. Yang XQ (1998) Second-order global optimality conditions for convex composite optimization. Math Program 81:327–347
21. Yang XQ, Jeyakumar V (1997) First and second-order optimality conditions for convex composite multi-objective optimization. J Optim Th Appl 95:209–224

Computational Complexity Theory

HAMILTON EMMONS, SANATAN RAI
Department OR and Operations Management,
Case Western Reserve University, Cleveland, USA

MSC2000: 90C60

Article Outline

Keywords
Definitions
The Nature of the Time Complexity Function
Polynomial Versus Exponential Algorithms
Reducibility
Classification of Hard Problems
Using Reduction to Establish Complexity
See also
References

Keywords

Computational complexity; Complexity theory; Combinatorial optimization; Decision problem; Recognition problem; Time complexity function; Efficient algorithm; Polynomial algorithm; Exponential algorithm; Reducibility; Nondeterministic polynomial algorithm; *NP*-hard problem; *NP*-complete problem

Many problems that arise in operations research and related fields are *combinatorial* in nature: problems where we seek the optimum from a very large but finite number of solutions. Sometimes such problems can be solved quickly and efficiently, but often the best solution procedures available are slow and tedious. It therefore becomes important to assess how well a proposed procedure will perform.

The theory of *computational complexity* addresses this issue. Complexity theory is a comparatively young field, with seminal papers dating from 1971–1972 ([1,5]). Today, it is a wide field encompassing many subfields. For a formal treatment, see [6]. As we shall see, the theory partitions all realistic problems into two groups: the 'easy' and the 'hard' to solve, depending on how complex (hence how fast or slow) the computational procedure for that problem is. The theory defines still other classes, but all but the most artificial mathematical constructs fall into these two. Each of them can be further subdivided in various ways, but these refinements are beyond our scope. It should be noted that we have not here used the accepted terminology, which is introduced below.

Definitions

A *problem* is a well-defined question to which an unambiguous answer exists. *Solving the problem* means answering the question. The problem is stated in terms of several *parameters*, numerical quantities which are left unspecified but are understood to be predetermined. They make up the data of the problem. An *instance* of a problem gives specified values to each parameter. A *combinatorial optimization problem*, whether *maximization* or *minimization*, has for each instance a finite number of candidates from which the answer, or optimal solution, is selected. The choice is based on a real-valued objective function which assigns a value to each candidate solution. A *decision problem* or *recognition problem* has only two possible answers, YES or NO.

Example 1 For example, consider the problem of solving a given system of linear equations. Stated as a question, it becomes: 'what is the solution to $\mathbf{A}\,\mathbf{x} = \mathbf{b}$?' with parameters m, n, $a_{i,j}$, b_i, x_j where $i = 1, \ldots, m$, $j = 1, \ldots, n$. An instance might be: 'What is the solution to $7x_1 - 3x_2 = 16$ and $2x_1 + 5x_2 = 9$?' with parameters $m = 2$, $n = 2$ etc.

This is neither an optimization problem nor a decision problem. An example of an optimization problem is a linear program, which asks: 'what is the greatest value of $\mathbf{cx}$ subject to $\mathbf{Ax} \leq \mathbf{b}$?' To make this a combinatorial optimization problem, we might make the variable $\mathbf{x}$ bounded and integer-valued so that the number of candidate solutions is finite. A decision problem is: 'does there exist a solution to the linear program with $\mathbf{cx} \geq k$?'

To develop *complexity theory*, it is convenient to state all problems as decision problems. An optimization (say, maximization) problem can always be replaced by a sequence of problems of determining the existence of solutions with values exceeding $k_1, k_2, \ldots$. An *algorithm* is a step-by-step procedure which provides a solution to a given problem; that is, to all instances of the problem. We are interested in how fast an algorithm is. We now introduce a measure of algorithmic speed: the time complexity function.

The Nature of the Time Complexity Function

Complexity theory does not measure the speed of an algorithm directly; that would depend on the speed of the computer being used and other extraneous factors. Rather, it considers the rate of growth of the solution time as a function of the instance size. Since different instances of the same size may require dramatically different solution times, we use the 'worst case' or longest time that any instance of that size requires. This maximal time needed to solve a problem instance, as a function of its size, is called the *time complexity function* (TCF) or simply the *complexity of the algorithm*. When we speak of the complexity of the problem, we mean the complexity of the most efficient algorithm (known or unknown) that solves it.

We need to clarify what we mean by the 'time required' and the 'size of an instance'. First, note that

we always think of solving problems using a computer. Thus, an algorithm is a piece of computer code. Similarly, the *size of a problem instance* is technically the number of characters needed to specify the data, or the length of the input needed by the program. For a decision problem, an algorithm receives as input any string of characters, and produces as output either YES or NO or 'this string is not a problem instance'. An algorithm *solves the instance or string in time m* if it requires m basic operations to reach one of the three conclusions and stop.

In order to avoid detailed consideration of the exact input length (are binary or alphanumeric characters used? what encoding scheme is used?), as well as avoiding precise measurement of solution times, the theory requires no more than *orders of magnitude* of these measurements. Recall, we are only concerned with the rate of increase of solution time as instances grow. For example, we may ask how much longer it takes if we double the instance size. As long as we enter data consistently, an instance that is twice as big as another under one data entry scheme remains twice as big under another. (For a rigorous proof and other technical issues, see [4]). Indeed, it is customary to use as a surrogate for instance size, any number that is roughly proportional to the true value. We shall use the symbol n, $n = 1, 2, \ldots$, to represent the size of a problem instance. In summary, for a decision problem Π:

Definition 2 The *time complexity function* (TCF) of algorithm A is:

$$T_A(n) = \begin{cases} \text{maximal time for } A \\ \text{to solve any string of length } n. \end{cases}$$

In what follows, the big oh notation ($\mathcal{O}$) introduced in [3] will be used when expressing the time complexity function. We say that, for two real-valued functions f and g, '$f(n)$ is $\mathcal{O}(g(n))$', or '$f(n)$ is of the same order as $g(n)$', if $|f(n)| \leq k \cdot |g(n)|$ for all $n \geq 0$ and some $k > 0$.

Polynomial Versus Exponential Algorithms

An *efficient, polynomially bounded, polynomial time algorithm*, or simply a *polynomial algorithm*, is one which solves a problem instance in time bounded by a power of the instance size. Formally:

Definition 3 An algorithm A is *polynomial time* if there exists a polynomial p such that

$$T_A(n) \leq p(n), \quad \forall n \in \mathbb{Z}^+ \equiv \{1, 2, \ldots\}.$$

More specifically, an algorithm is *polynomial of degree c*, or *has complexity* $\mathcal{O}(n^c)$, or *runs in* $\mathcal{O}(n^c)$ *time* if, for some $k > 0$, the algorithm never takes longer than kn^c (the TCF) to solve an instance of size n.

Definition 4 The collection P comprises all problems for which a polynomial time algorithm exists.

Problems which belong to P are the ones we referred to earlier as 'easy'. All other algorithms are called *exponential time* or just *exponential*, and problems for which nothing quicker exists are 'hard'. Although not all algorithms in this class have TCF's that are technically exponential functions, we may think of a typical one as running in $\mathcal{O}(c^{p(n)})$ for some polynomial $p(n)$. Other examples of exponential rates of growth are n^n and $n!$.

The terms 'hard' and 'easy' are somewhat misleading, even though exponential TCFs clearly lead to far more rapid growth in solution times. Suppose an 'easy' problem has an algorithm with running time bounded by, say kn^5. Such a TCF may not be exponential, but it may well be considered pretty rapidly growing. Furthermore, some algorithms take a long time to solve even small problems (large k), and hence are unsatisfactory in practice even if the time grows slowly. On the other hand, an algorithm for which the TCF is exponential is not always useless in practice. Recall, the concept of the TCF is a worst case estimate, so complexity is only an upper bound on the amount of time required by an algorithm. This is a conservative measure and usually useful, but it is too pessimistic for some popular algorithms. The simplex algorithm for linear programming, for example, has a TCF that is $\mathcal{O}(2^n)$, but it has been shown that for the average case the complexity is only $\mathcal{O}(n)$. Thus, the algorithm is actually very fast for most problems encountered.

Despite these caveats, exponential algorithms generally have running times that tend to increase at an exponential rate and often seem to 'explode' when a certain problem size is exceeded. Polynomial time algorithms usually turn out to be of low degree ($\mathcal{O}(n^3)$ or better), run pretty efficiently, and are considered desirable.

Reducibility

A problem can be placed in P as soon as a polynomial time algorithm is found for it. Sometimes, rather than finding such an algorithm, we may place it in P by showing that it is 'equivalent' to another problem which is already known to be in P. We explain what we mean by equivalence between problems with the following definitions.

Definition 5 A problem Π' is *reducible* or *transformable* to a problem Π ($\Pi' \propto \Pi$) if, for any instance I' of Π', an instance I of Π can be constructed in polynomially bounded time, such that the solution to I is sufficient to find the solution to I' in polynomial time.

We call the construction of the I that corresponds to I' a *polynomial transformation* of I' into I.

Definition 6 Two problems are *equivalent* if each is reducible (or simply *reduces*) to the other.

Since reduction, and hence equivalence, are clearly transitive properties, we can define *equivalence classes of problems*, where all problems in the same equivalence class are reducible (or equivalent) to each other. Consider polynomial problems. Clearly, for two equivalent problems, if one is known to be polynomial, the other must be, too. Also, if two problems are each known to be polynomial, they are equivalent. This is because any problem $\Pi' \in P$ is reducible to any other problem $\Pi \in P$, or indeed to any $\Pi \notin P$, in the following trivial sense. For any instance I' of Π', we can pick any instance of Π, ignore its solution, and find the solution to I' directly. We conclude that P is an equivalence class.

We state a third simple result for polynomial problems as a theorem.

Theorem 7 *If $\Pi \in P$, then $\Pi' \propto \Pi \Rightarrow \Pi' \in P$.*

Given any instance I' of Π', one can find an instance I of Π by applying a polynomial time transformation to I'. Since $\Pi \in P$, there is a polynomial time algorithm that solves I. Hence, using the transformation followed by the algorithm, I' can be solved in polynomial time.

Classification of Hard Problems

In practice, we do not usually use reduction to show a problem is polynomial. We are more likely to start optimistically looking for an *efficient algorithm* directly, which may be easier than seeking another problem known to be polynomial, for which we can find an appropriate transformation. But suppose we cannot find either an efficient algorithm or a suitable transformation. We begin to suspect that our problem is not 'easy' (i. e., is not a member of P). How can we establish that it is in fact 'hard'? We start by defining a larger class of problems, which includes P and also all the difficult problems we ever encounter. To describe it, consider any combinatorial decision problem. For a typical instance, there may be a very large number of possible solutions which may have to be searched. Picture a candidate solution as a set of values assigned to the variables $\mathbf{x} = (x_1, \ldots, x_n)$. The question may be 'for a given vector $\mathbf{c}$ is there a solution $\mathbf{x}$ such that $\mathbf{cx} \leq B$?' and the algorithm may search the solutions until it finds one satisfying the inequality (whereupon it stops with the answer YES) or exhausts all solutions (and stops at NO).

This may well be a big job. But suppose we are told 'the answer is YES, and here is a solution $\mathbf{x}$ that satisfies the inequality'. We feel we must at least *verify* this, but that is trivial. Intuitively, even for the hardest problems, the amount of work to check that a given candidate solution confirms the answer YES should be small, even for very large instances. We will now define our 'hard' problems as those which, though hard to solve, are easy to verify, where as usual 'easy' means taking a time which grows only polynomially with instance size. To formalize this, let:

$$V_A(n) = \begin{cases} \text{maximal time for } A \\ \text{to verify that a given solution} \\ \text{establishes the answer YES} \\ \text{for any instance of length } n. \end{cases}$$

Definition 8 An algorithm $\tilde{A}$ is *nondeterministic polynomial time* if there exists a polynomial p such that for every input of length n with answer YES, $V_{\tilde{A}}(n) \leq p(n)$

Definition 9 The collection *NP* comprises all problems for which a nondeterministic polynomial algorithm exists.

It may be noted that a problem in *NP* is solvable by searching a decision tree of polynomially bounded depth, since verifying a solution is equivalent to trac-

ing one path through the tree. From this, it is easy to see that $P \subseteq NP$. Strangely, complexity theorists have been unable to show that $P \subset NP$; it remains possible that all the problems in *NP* could actually be solved by polynomial algorithms, so that $P = NP$. However, since so many brilliant researchers have worked on so many difficult problems in *NP* for so many years without success, this is regarded as being very unlikely. Assuming $P \neq NP$, as we shall hereafter, it can be shown that the problems in *NP* include an infinite number of equivalence classes, which can be ranked in order of increasing difficulty; where an equivalence class *C* is 'more difficult' than another class C' if, for every problem $\Pi \in C$ and every $\Pi' \in C'$, $\Pi' \propto \Pi$ but $\Pi \not\propto \Pi'$. There also exist problems that cannot be compared: neither $\Pi \propto \Pi'$ nor $\Pi' \propto \Pi$.

Fortunately, however, all problems that arise naturally have always been found to lie in one of two equivalence classes: the 'easy' problems in *P*, and the 'hard' ones, which we now define.

The class of *NP-hard problem* (*NPH*) is a collection of problems with the property that every problem in *NP* can be reduced to the problems in this class. More formally,

Definition 10

$$NPH = \left\{\Pi:\ \forall \Pi' \in NP:\ \Pi' \propto \Pi\right\}.$$

Thus each problem in *NPH* is at least as hard as any problem in *NP*. We know that some problems in *NPH* are themselves in *NP*, though some are not. Those that are include the toughest problems in *NP*, and form the class of *NP-complete problem* (*NPC*). That is,

Definition 11

$$NPC = \left\{ \Pi: \begin{array}{c} (\Pi \in NP) \\ \text{and} \\ (\forall \Pi' \in NP:\ \Pi' \propto \Pi) \end{array} \right\}.$$

The problems in *NPC* form an equivalence class. This is so because all problems in *NP* reduce to them, hence, since they are all in *NP*, they reduce to each other. The class *NPC* includes the most difficult problems in *NP*. As we mentioned earlier, by a surprising but happy chance, all the problems we ever encounter outside the most abstract mathematics turn out to belong to either *P* or *NPC*.

Using Reduction to Establish Complexity

When tackling a new problem Π, we naturally wonder whether it belongs to *P* or *NPC*. As we said above, to show that the problem belongs to *P*, we usually try to find a polynomial time algorithm, though we could seek to reduce it to a problem known to be polynomial. If we are unable to show that the problem is in *P*, the next step generally is to attempt to show that it lies in *NPC*; if we can do so, we are justified in not developing an efficient algorithm. To do this, clearly no direct algorithmic development is possible, and only a reduction argument will do. This is based on the following theorem, which should be clear enough to require no proof.

Theorem 12 $\forall \Pi\ \Pi' \in NP$: *($\Pi' \in NPC$) and ($\Pi' \propto \Pi$) imply* $\Pi \in NPC$.

Thus, we need to find a problem $\Pi' \in NPC$ and show $\Pi' \propto \Pi$, thereby demonstrating that Π is at least as hard as any problem in *NPC*. To facilitate this, we need a list of problems known to be in *NPC*. Several hundred are listed in [2] in a dozen categories such as graph theory, mathematical programming, sequencing and scheduling, number theory, etc., and more are being added all the time. Even given an ample selection, a good deal of tenacity and ingenuity are needed to pick one with appropriate similarities to ours and to fill in the precise details of the transformation.

Of course, to build up the membership in *NPC* using Theorem 12, we need other problems that have already been show to belong to that class. To begin this process, at least one problem needs to be in *NPC*. It was S.A. Cook [1] who showed that the satisfiability problem is *NP*-complete, using direct arguments that did not involve reduction. This very important result is called *Cook's theorem*. For a proof, see [2].

As a simple illustration of reduction, we show that the traveling salesperson decision problem (TSP) is in *NPC*. To do so, we first select a closely related problem, the Hamiltonian circuit problem (HCP), which we assume has already been shown to be *NP*-complete. We then find a reduction of HCP to TSP. The problems are defined as follows.

Definition 13 (*TSP*, *traveling salesperson problem*).

Instance:

- a positive integer n;
- a finite set $C = \{c_1, \ldots, c_n\}$ of 'cities';
- 'distances', $d_{ij} \in \mathbf{Z}^+$. $\forall i, j : c_i, c_j \in C$;
- a bound $B \in \mathbf{Z}^+$.

Question: Does there exist a *tour* (i. e., a closed path that visits every city exactly once), of length no greater than B?

Definition 14 (***HCP, Hamiltonian circuit problem***) Instance: A graph $G = (V, E)$, where V is the set of m vertices, and E the set of edges.

Question: Does G contain a *Hamiltonian circuit*, i. e., a tour that traverses all vertices exactly once?

Example 15 To show: HCP $\propto$ TSP.

In TSP, we have a complete graph and seek the shortest tour, whereas in HCP, given an arbitrary graph we require any tour. Thus, given the challenge of showing that the traveling salesperson problem (or the decision version of it) is *NP*-complete, we have found a similar problem whose membership in *NPC* is already established. We may still be unable to find a polynomial transformation from HCP, in which case another problem must be sought. A transformation of Π' to Π is a way of computing each parameter of Π in terms of the parameters of Π'. In this case, the reduction is relatively simple. The parameters of HCP are:

- m = cardinality V;
- $E = \left\{(i, j): \begin{array}{c} G \text{ contains an arc} \\ \text{between vertices } i, j \end{array}\right\}$.

The parameters of TSP are computed as follows:

- $n = m$;
- $d_{ij} = \begin{cases} 1 & (i, j) \in E, \\ N & \text{otherwise}; \end{cases}$
- $B = m$.

Here, N can be any number larger than 1; say, 2. Clearly, the shortest possible tour in TSP has length m, and this only occurs when arcs in E are used exclusively; that is, when a tour in HCP exists.

To complete the reduction, we need to show that the transformation can be performed in polynomial time. For that, given a pair of nodes in TSP, we need to check if an arc exists in HC, and this requires time

$$\mathcal{O}\left[\frac{m(m-1)}{2}\right] = \mathcal{O}(m^2).$$

See also

- ▶ Complexity Classes in Optimization
- ▶ Complexity of Degeneracy
- ▶ Complexity of Gradients, Jacobians, and Hessians
- ▶ Complexity Theory
- ▶ Complexity Theory: Quadratic Programming
- ▶ Fractional Combinatorial Optimization
- ▶ Information-Based Complexity and Information-Based Optimization
- ▶ Kolmogorov Complexity
- ▶ Mixed Integer Nonlinear Programming
- ▶ Parallel Computing: Complexity Classes

References

1. Cook SA (1971) The complexity of theorem proving procedures. Proc. 3rd Annual ACM Symposium on Theory of Computing. ACM, New York, pp 151–158
2. Garey MR, Johnson DS (1979) Computers and intractability. Freeman, New York
3. Hardy GH, Wright EM (1979) An introduction to the theory of numbers. Clarendon Press, Oxford
4. Hopcroft JE, Ullman JD (1979) Introduction to automata theory, languages, and computation. Addison-Wesley, Reading, MA
5. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations. Plenum, New York, pp 85–103
6. Papadimitriou CH (1994) Computational complexity. Addison-Wesley, Reading, MA
7. Pinedo M (1995) Scheduling: Theory, algorithms and systems. Prentice-Hall, Englewood Cliffs, NJ

Concave Programming

HAROLD P. BENSON
Department Decision and Information Sci.,
University Florida, Gainesville, USA

MSC2000: 90C25

Article Outline

Keywords
See also
References

Keywords

Concave programming; Multi-extremal global optimization

Concave programming constitutes one of the most fundamental and intensely-studied problem classes in deterministic nonconvex optimization. There are at least three reasons for this. First, many of the mathematical properties of concave programming are intriguingly attractive. Some are even identical to properties of linear programming. Second, concave programming has a remarkably broad range of direct and indirect applications. Third, the algorithmic ideas used in concave programming have played and continue to play an active and often fundamental role in the development of solution procedures for other types of nonconvex programming problems.

The *concave programming*, or *concave minimization*, problem (CMP) can be written

$$\begin{cases} \text{globmin} & f(x), \\ \text{s.t.} & x \in D, \end{cases}$$

where D is a nonempty, closed convex set in $\mathbf{R}^n$ and f is a real-valued, concave function defined on some open convex set A in $\mathbf{R}^n$ that contains D. The goal in CMP is to find the global minimum value that f achieves over D, and, if this value is not equal to $-\infty$, to find, if it exists, at least one point in D that achieves this value. In many applications, D is compact and A equals all of $\mathbf{R}^n$. CMP invariably contains many points in D that are local, but not global, minimizers of f over D. For this reason, CMP is an example of a *(multi-extremal) global optimization* problem [7].

The application of standard algorithms designed for solving constrained convex programming problems generally will fail to solve CMP. Even instances of CMP with relatively simple components can apparently present very significant solution challenges. For example, B. Kalantari [8] has shown that in problems involving the minimization of concave quadratic functions over rectangles in $\mathbf{R}^n$, an exponential number of extreme point local minima can exist. Additionally, P.M. Pardalos and G. Schnitger [13] have shown that minimizing a concave quadratic function over a hypercube is an *NP*-hard problem.

Although CMP is more difficult to solve than a convex programming problem, it possesses some highly interesting, special mathematical properties. A number of these properties have been exploited by researchers to create successful algorithms for solving the problem.

For instance, if D contains at least one extreme point, and CMP has at least one global optimal solution, then there must exist a global optimal solution which is an extreme point of D [14]. This is perhaps the most important and striking property of concave minimization problems. As a result of this property, just as in linear programming, if CMP has a global optimal solution, then one can confine the search for such a solution to the set of extreme points of D, provided that this set is nonempty. This property holds, as in linear programming, even when D is unbounded. A number of algorithms for CMP are based upon this property.

Another highly important property for CMP is that if D is a compact set, then CMP must have a global optimal solution which is an extreme point of D. This is perhaps the most widely-known theoretical result in concave minimization [1]. Like the property stated in the previous paragraph, it forms the basis for a number of important concave minimization algorithms.

For cases where D is a polyhedron, possibly unbounded, that contains at least one extreme point, it has been shown that either CMP has a global optimal solution which is an extreme point of D, or CMP must be unbounded and f must be unbounded from below over some extreme direction of D. Notice that the same property, remarkably, holds in the case of linear programming. This property is used by a large number of the algorithms designed to solve CMP when D is a nonempty polyhedron.

CMP displays a remarkable diversity of applications. Each application is either direct or indirect. By direct, we mean that the original model formulation takes the form of CMP immediately or, if not, with only relatively simple algebraic manipulations. The indirect applications involve problems whose direct formulations do not take the form of CMP, but existing theory can be used to reformulate these problems in the form of CMP.

Some of the oldest and most diverse direct applications of CMP belong to a class of problems called *fixed charge* problems. In these problems, the objective func-

tion f is *separable*, i. e., it is of the form

$$f(x) = \sum_{i=1}^{n} f_i(x_i) \, .$$

For each $i = 1, \ldots, n$, in these problems f_i is a concave function on $\{x_i \in \mathbf{R}: x_i \geq 0\}$ of the form

$$f_i(x_i) = \begin{cases} 0 & \text{if } x_i = 0, \\ c_i + w_i(x_i) & \text{if } x_i > 0, \end{cases}$$

where $c_i > 0$ is the fixed *setup cost* of undertaking activity i at a positive level, and $w_i(x_i)$ is a continuous concave function on $\{x_i \in \mathbf{R} : x_i > 0\}$ that represents the *variable cost* of undertaking the activity at level x_i.

When the functions $w_i(x_i)$, $i = 1, \ldots, n$, are linear, the classical *linear fixed charge* problem is obtained. Some of the oldest applications of concave minimization involve solving problems of this type. Among these are applications to transportation planning, site selection, production lot sizing and network design. For cases where at least one of the functions $w_i(x_i)$, $i = 1, \ldots, n$, is piecewise linear, several types of applications have been reported. Included among these are problems involving price breaks, such as bid evaluation problems, certain inventory planning problems and various plant location problems. When $w_i(x_i)$ is a general concave function for some $i = 1, \ldots, n$, applications involving *economies of scale*, for instance, can be solved.

More recently, CMP has been directly applied to a class of problems called *multiplicative programming* problems. These are problems of the form CMP where

$$f(x) = \prod_{j=1}^{p} f_j(x) \tag{1}$$

for some set of $p \geq 2$ functions f_j, $j = 1, \ldots, p$, that are each nonnegative over D. Notice that if $f_k(\overline{x}) = 0$ for some $k \in \{1. \ldots, p\}$ and some $\overline{x} \in D$, then it is easy to see that the global minimum value for CMP is 0, and $\overline{x}$ is a global optimal solution. Therefore, it is generally assumed in multiplicative programming problems that each function f_j is positive over D. Let us make this assumption henceforth.

The objective function (1) of a multiplicative program is generally not a concave function. But, when each function f_j, $j = 1, \ldots, p$, is a concave function on $\mathbf{R}^n$, some simple transformations of (1) yield concave functions over D. For instance, if, for each $x \in D$, we define w_1 and w_2 by

$$w_1(x) = \ln f(x) = \sum_{j=1}^{p} \ln f_j(x),$$

$$w_2(x) = [f(x)]^{\frac{1}{p}},$$

respectively, then, whenever each f_j, $j = 1, \ldots, p$, is a concave function on $\mathbf{R}^n$, both w_1 and w_2 are concave functions on D [3,10]. Thus, by using w_1 or w_2, multiplicative programming problems in which f_j, $j = 1, \ldots, p$, are concave functions can be easily transformed to concave minimization problems.

Various applications of multiplicative programming problems with concave or linear functions f_j, $j = 1, \ldots, p$, in (1) have arisen, especially since the 1960s. For example, the linear case has been applied to the problem of optimizing value functions for multiple objective programming problems subject to linear or nonlinear constraints. For $p = 2$, the linear case has been used to help solve the modular design problem, to design integrated circuit chips and to select bond portfolios. The concave case has been used to analyze and solve a number of problems in microeconomics.

Subject to occasional restrictions, large classes of integer programming problems can be converted by various means into the form of CMP and solved as concave minimization problems. As a result, these integer programming problems, indirectly, are applications of concave minimization. The transformation processes used to accomplish the conversion, however, can be rather involved. They may also increase the size of the original problem [4], and they may call for choosing values for parameters that are difficult to determine [5,9].

In particular, by using a certain general transformation process, any feasible linear integer or quadratic integer programming problem over a polyhedron with nonnegative, bounded variables can be converted into the form of CMP and solved as a concave minimization problem. By using more customized conversion processes, linear zero-one programs, quadratic assignment problems, and other special integer programming problems can also be transformed to the form of CMP and solved as concave minimization problems. The specialized transformations generally take advantage of

some aspect of the original integer programming problem that the general processes ignore.

There are many other indirect applications of CMP, including *d.c. optimization*, *indefinite quadratic programming*, and *bilinear programming*, for instance. For further details and additional direct and indirect applications, see [1,2,6,7,11,12].

To solve CMP, a large number of algorithms have been developed. Many of these rely on one or more of the following four approaches.

In the *cutting plane approach*, a local minimum for f over D is found. Subsequently, a hyperplane is constructed and used to cut off all points of D whose objective function values are not less than that of the local minimum. This yields a new closed convex set $D^1 \subset D$. This process is then repeated with D^1 in the role of D. By iterating this process, the portion of D remaining to be explored is progressively reduced. Termination occurs when it can be shown that $f(y) \geq f(x^k)$ for all $y \in D^k$, where D^k is the portion of D remaining at iteration k, and x^k is the local minimum found through iteration k with the smallest objective function value.

In a typical *outer approximation* approach, D is assumed to be compact. To initiate the approach, a simple bounded polyhedron P^1 containing D whose vertices can be enumerated is constructed. A vertex v^1 of P^1 of minimum objective function value among all of the vertices of P^1 is found. This gives a lower bound $f(v^1)$ for the optimal value of CMP. If $v^1 \in D$, v^1 is a global optimal solution for CMP and termination occurs. Otherwise, a new bounded polyhedron $P^2 \subset P^1$ is constructed that contains D, and its vertices are enumerated. With P^2 in the role of P^1, the process is repeated. By repeating this process, a sequence of telescoping bounded polyhedra containing D is obtained. Termination occurs in the first iteration k where the vertex v^k found that minimizes f over all of the vertices of P^k lies in D.

In *inner approximation* (*polyhedral annexation*) approaches for CMP, D is assumed to be a bounded polyhedron. Typically, at each major iteration of an inner approximation algorithm, a local minimum extreme point solution $\overline{x}$ for CMP is available. A sequence of expanding inner approximating compact polyhedra for $(D \cap G)$ is constructed via a series of subiterations, where $G = \{x \in \mathbb{R}^n : f(x) \geq f(\overline{x})\}$. During this process, either an improved local minimum extreme point $\overline{\overline{x}}$ is found, or, after k subiterations, the algorithm detects that $D \subseteq P^k$, where P^k is the current inner approximation of $(D \cap G)$. In the former case, $\overline{\overline{x}}$ replaces $\overline{x}$ and a new major iteration begins. In the latter case, since $P^k \subseteq (D \cap G)$, it follows that $D \subseteq G$, and the algorithm therefore terminates with the global optimal solution $\overline{x}$.

In the *branch and bound* approaches for CMP, D is repeatedly subdivided into finer and finer partitions. A lower bound for f over each partition element is calculated. The lowest of these lower bounds at any step k of the process gives a global lower bound LB_k for f over D. At any stage, typically some feasible solutions for CMP have been detected. A feasible solution $\overline{y}$ with the smallest f value among all feasible solutions detected through any point in the algorithm is always available. This solution is called the *incumbent solution*. When, at some step k, the inequality $LB_k \geq f(\overline{y})$ holds for the first time, the algorithm stops and returns the global optimal solution $\overline{y}$.

Details concerning these and other solution approaches can be found in [1,2,6,7].

See also

- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Minimum Concave Transportation Problems

References

1. Benson HP (1995) Concave minimization: Theory, applications and algorithms. In: Horst R, Pardalos PM (eds) Handbook global optimization. Kluwer, Dordrecht, pp 43–148
2. Benson HP (1996) Deterministic algorithms for constrained concave minimization: A unified critical survey. Naval Res Logist 73:765–795
3. Benson HP, Boger GM (1997) Multiplicative programming problems: Analysis and efficient point search heuristic. J Optim Th Appl 94:487–510
4. Garfinkel RS, Nemhauser GL (1972) Integer programming. Wiley, New York
5. Giannessi F, Niccolucci F (1976) Connections between nonlinear and integer programming problems. Symp Mat Inst Naz di Alta Mat 19:161–176
6. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
7. Horst R, Tuy H (1993) Global optimization: Deterministic approaches, 2nd revised edn. Springer, Berlin
8. Kalantari B (1986) Quadratic functions with exponential number of local maxima. Oper Res Lett 5:47–49
9. Kalantari B, Rosen JB (1982) Penalty for zero-one integer equivalent problems. Math Program 24:229–232

10. Konno H, Kuno T (1995) Multiplicative programming problems. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 369–405
11. Pardalos PM (1994) On the passage from local to global in optimization. In: Birge JR, Murty KG (eds) Mathematical Programming: State of the Art. Braun-Brumfield, Ann Arbor, MI, pp 220–247
12. Pardalos PM, Rosen JB (1987) Constrained global optimization: Algorithms and applications. Springer, Berlin
13. Pardalos PM, Schnitger G (1988) Checking local optimality in constrained quadratic programming is NP-hard. Oper Res Lett 7:33–35
14. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton

Conjugate-Gradient Methods

J. L. NAZARETH[1,2]
[1] Department Pure and Applied Math., Washington State University, Pullman, USA
[2] Department Applied Math., University Washington, Seattle, USA

MSC2000: 90C30

Article Outline

Keywords
Introduction
  Notation and Preliminaries
Linear CG Algorithms
Nonlinear CG Algorithms
Nonlinear CG-Related Algorithms
  Classical Alternatives to CG
  Nonlinear CG Variants
  Variable-Storage/Limited-Memory Algorithms
  Affine-Reduced-Hessian Algorithms
Conclusion
See also
References

Keywords

Unconstrained minimization; Conjugate gradients; Linear CG method; Nonlinear CG method; Nonlinear CG-related algorithms; Variable-storage; Limited-memory; Affine-reduced-Hessian; Three-term-recurrence

Introduction

Conjugate-gradient methods (CG methods) are used to solve large-dimensional problems that arise in computational linear algebra and computational nonlinear optimization. These two subjects share a broad common frontier, and one of the most easily traversed crossing points is via the following simple observation: the problem of solving a system of linear equations $\mathbf{Ax} = \mathbf{b}$ for the unknown vector $\mathbf{x} \in \mathbf{R}^n$, where $\mathbf{A}$ is a positive definite, symmetric matrix and $\mathbf{b}$ is a given vector, is mathematically equivalent to finding the minimizing point of the strictly convex quadratic function

$$q(\mathbf{x}) = -\mathbf{b}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathbf{Ax}.$$

The *linear CG method* for solving the system of linear equations is able to capitalize on this equivalent optimization formulation. It was developed in the pioneering 1952 paper of M.R. Hestenes and E.L. Stiefel [11] who, in turn, cite antecedents in the contributions of several other authors (see [9]). The method fell out of favor with numerical analysts during the 1960s because it did not compete with direct methods, in particular, Gaussian elimination, but it continued to be widely used in real-world applications by specialists in other areas. Interest in CG as an iterative method, downplaying its finite-termination properties, revived in the 1970s when the solution of *large scale linear systems* was coming to the forefront of academic research.

The *nonlinear CG method* extends the linear CG approach to the problem of minimizing a smooth, nonlinear function $f(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^n$, where n can be large. It was developed in another landmark article published in 1964 by R. Fletcher and C. Reeves [8]. Optimization techniques for this class of problems, which are inherently iterative in nature, form a direction of descent at an approximation to the solution (the current iterate), and search along this direction to obtain a new iterate with an improved function value. The *Fletcher–Reeves algorithm* combined a search direction derived from the *Hestenes–Stiefel approach* with an efficient line search procedure along this direction vector adapted from the 1959 variable-metric breakthrough algorithm of W.C. Davidon [6]. The resulting CG algorithm was a marked enhancement of the classical steepest descent method of A.-L. Cauchy.

Like steepest descent, the nonlinear CG method is storage-efficient and requires only a few n-vectors of computer storage beyond that needed to specify the problem itself. During the three decades after its discovery, a large number of storage-efficient *nonlinear CG-related algorithms* were proposed. In particular, a structural connection between conjugate-gradient and variable-metric techniques for defining search direction vectors provided the springboard for effective new families of variable-storage/limited-memory algorithms and affine-reduced-Hessian algorithms that occupy a middle ground.

These three classes of algorithms, namely, linear CG, nonlinear CG and nonlinear CG-related, will be discussed in the respective Sections below.

Notation and Preliminaries

Lowercase boldface letters, e. g., $\mathbf{x}$, denote vectors, and uppercase boldface letters, e. g., $\mathbf{A}$, denote symmetric, positive definite matrices. The *residual* at $\mathbf{x}$ of the linear system $\mathbf{Ax} = \mathbf{b}$ is $\mathbf{r} = \mathbf{Ax} - \mathbf{b}$. It equals the gradient vector $\mathbf{g} = -\mathbf{b} + \mathbf{Ax}$ of the strictly convex, quadratic form

$$q(\mathbf{x}) = -\mathbf{b}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathbf{Ax}$$

at $\mathbf{x}$. The gradient vector of q vanishes only at the unique solution $\mathbf{A}^{-1}\mathbf{b}$ of the linear system.

Linear CG Algorithms

A basic CG algorithm for solving the system of linear equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a positive definite, symmetric matrix, is as follows:

	(Initialization)
0	$\mathbf{x}_1$ = arbitrary;
	$\mathbf{r}_1$ = residual of linear system at x_1;
	$\mathbf{d}_1 = -\mathbf{r}_1$;
	(Iteration i)
1	$\mathbf{x}_{i+1}$ = unique minimazing point of q on halfline through $\mathbf{x}_i$ along direction $\mathbf{d}_i$;
2	$\mathbf{r}_{i+1}$ = residual of linear system at $\mathbf{x}_{i+1}$;
3	$\beta_i = \|\|\mathbf{r}_{i+1}\|\|^2/\|\|\mathbf{r}_i\|\|^2$;
4	$\mathbf{d}_{i+1} = -\mathbf{r}_{i+1} + \beta_i \mathbf{d}_i$.

In the computational linear algebra setting, the matrix $\mathbf{A}$ is provided exogenously. The residual $\mathbf{r}_{i+1}$, at step 2, is computed as $\mathbf{Ax}_{i+1} - \mathbf{b}$ or else obtained by updating $\mathbf{r}_i$. The direction $\mathbf{d}_i$ is always a descent direction for q at $\mathbf{x}_i$. At step 1, the minimizing point is computed as follows:

$$\alpha_i = -\frac{\mathbf{r}_i^\top \mathbf{d}_i}{\mathbf{d}_i^\top \mathbf{Ad}_i}; \quad \mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i.$$

There are numerous variants on this basic algorithm that seek to enhance convergence through problem preconditioning (transformation of variables), to improve algorithm stability and to solve related computational linear algebra problems. A contextual overview and further references can be found in [2].

Here our focus is optimization. In this setting, the residual at $\mathbf{x}_{i+1}$ is given its alternative interpretation and representation as the gradient vector $\mathbf{g}_{i+1}$ of q at $\mathbf{x}_{i+1}$, and this gradient is assumed to be provided exogenously. The minimizing point at step 1 is computed, alternatively, as follows:

$$\alpha_i = -\frac{\mathbf{g}_i^\top (\bar{\mathbf{x}}_i - \mathbf{x}_i)}{\mathbf{d}_i^\top (\bar{\mathbf{g}}_i - \mathbf{g}_i)},$$

where $\bar{\mathbf{x}}_i$ is any point on the ray through $\mathbf{x}_i$ in the direction $\mathbf{d}_i$ and $\bar{\mathbf{g}}_i$ is its corresponding gradient vector; $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i$. The expression for α_i is derived from the previous linear systems version and the relation $\mathbf{A}(\bar{\mathbf{x}}_i - \mathbf{x}_i) = \bar{\mathbf{g}}_i - \mathbf{g}_i$.

We will call the resulting optimization algorithm the *CG-standard for minimizing q*. It will provide an important guideline for defining CG algorithms in the subsequent discussion.

Nonlinear CG Algorithms

A nonlinear CG algorithm is used to find a minimizing point of the nonlinear function $f(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^n$, when n is large and/or computer storage is at a premium. A basic algorithm can be stated as follows:

	(Initialization)
0	$\mathbf{x}_1$ = arbitrary;
	$\mathbf{g}_1$ = gradient of f at $\mathbf{x}_1$;
	$\mathbf{d}_1 = -\mathbf{g}_1$;
	(Iteration i)
1	$\mathbf{x}_{i+1}$ = an improved iterate on halfline through x_i along direction $\mathbf{d}_i$;
2	$\mathbf{g}_{i+1}$ = gradient of f at $\mathbf{x}_{i+1}$;
3	$\beta_i = \|\|\mathbf{g}_{i+1}\|\|^2/\|\|\mathbf{g}_i\|\|^2$;
4	$d_{i+1} = -\mathbf{g}_{i+1} + \beta_i \mathbf{d}_i$.

Note that this algorithm is closely patterned after the *CG-standard*. The improved iterate at step 1 is obtained by a *line search* procedure, which is normally based on quadratic or cubic polynomial fitting (suitably safeguarded). When $f = q$, such a line search procedure can immediately locate the minimizing point along the line of search, once it has gathered the requisite information to make an exact fit. In other words, when applied to the minimization of q, the foregoing nonlinear CG algorithm is able to replicate the CG-standard precisely. This property characterizes a nonlinear CG algorithm.

Considerable research has gone into alternative expressions for the quantity β_i. The four leading contenders are the *Fletcher–Reeves* (FR), *Hestenes–Stiefel* (HS), *Polyak–Polak–Ribière* (PPR) and *Dai–Yuan* (DY) choices (see [5,8,11,20,22]). These define β_i as follows:

$$\begin{aligned}
\text{FR}: \quad & \beta_i = \frac{\mathbf{g}_{i+1}^\top \mathbf{g}_{i+1}}{\mathbf{g}_i^\top \mathbf{g}_i}; \\
\text{HS}: \quad & \beta_i = \frac{\mathbf{g}_{i+1}^\top \mathbf{y}_i}{\mathbf{d}_i^\top \mathbf{y}_i}; \qquad (1) \\
\text{PPR}: \quad & \beta_i = \frac{\mathbf{g}_{i+1}^\top \mathbf{y}_i}{\mathbf{g}_i^\top \mathbf{g}_i}; \\
\text{DY}: \quad & \beta_i = \frac{\mathbf{g}_{i+1}^\top \mathbf{g}_{i+1}}{\mathbf{d}_i^\top \mathbf{y}_i}, \qquad (2)
\end{aligned}$$

where $y_i = \mathbf{g}_{i+1} - \mathbf{g}_i$ is the gradient change that corresponds to the step $\mathbf{s}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$.

When line searches are exact and the function is quadratic, the following relations hold:

$$\mathbf{g}_{i+1}^\top \mathbf{g}_{i+1} = \mathbf{g}_{i+1}^\top \mathbf{y}_i, \quad \mathbf{g}_i^\top \mathbf{g}_i = \mathbf{d}_i^\top \mathbf{y}_i. \qquad (3)$$

Thus, the values of the scalar β_i are identical for all four choices, and each of the associated algorithms becomes the CG-standard. In general, however, they are applied to nonquadratics and use inexact line searches, resulting in four distinct, nonlinear CG algorithms that can exhibit behavior very different from one another.

The following generalization yields a *two-parameter family*:

$$\beta_i = \frac{\lambda_i(\mathbf{g}_{i+1}^\top \mathbf{g}_{i+1}) + (1-\lambda_i)(\mathbf{g}_{i+1}^\top \mathbf{y}_i)}{\mu_i(\mathbf{g}_i^\top \mathbf{g}_i) + (1-\mu_i)(\mathbf{d}_i^\top \mathbf{y}_i)}, \qquad (4)$$

where $\lambda_i \in [0, 1]$ and $\mu_i \in [0, 1]$. For any choice of λ_i and μ_i in these ranges, the associated algorithm reduces to the CG-standard when f is quadratic and line searches are exact, which follows from (3). If the scalars λ_i and μ_i take only their extreme values, 0 or 1, then one obtains four possible combinations corresponding to (1)–(2). The above two-parameter family of nonlinear CG algorithms, which subsumes FR, HS, PPR and DY, and its subfamilies (defined, for example, by taking $\lambda_i \equiv 1$) are currently a topic of active research.

When the line search is sufficiently accurate, a nonlinear CG algorithm always produces a direction of descent at step 4. Suitable inexact line search termination conditions, in conjunction with different choices of β_i, have been extensively studied, both theoretically and computationally. A good overview of the theory and convergence analysis of nonlinear CG algorithms can be found in [19]. The nonlinear CG algorithm based on the PPR choice for β_i [20,22] and a suitable restarting strategy [23] has emerged as the most efficient in practice. However, it is well known that no single nonlinear CG algorithm works well all the time. There is enormous variability in performance on different problems or even within different regions of the same problem.

Nonlinear CG-Related Algorithms

We informally characterize a *nonlinear CG-related algorithm* as follows:

- its computer storage requirements are 'similar' to those of an implemented nonlinear CG algorithm, for example, [23];
- its path traverses the iterates of the CG-standard when the function is a strictly convex quadratic, line searches are exact, and the same initialization is used.

In other words, it may use a few more n-vectors of computer storage than, say, the PPR nonlinear CG algorithm, and it is permitted to generate additional intermediate iterates and form search vectors in novel ways. A nonlinear CG-related algorithm does not have to imitate the 'structure' of the basic nonlinear CG algorithm of the previous Section. But the above requirement that its path must cover the iterates of the CG-standard of the first Section implies the following: if the candidate algorithm does not exhibit finite termination when applied to a quadratic q then it is not a CG-related algorithm.

Let us now briefly categorize the main lines of development.

Classical Alternatives to CG

These seek to enhance or accelerate the steepest descent algorithm of Cauchy more directly, without explicitly introducing notions of conjugacy. The year of publication of [8] was indeed a banner year for such developments. Two particularly noteworthy contributions, which coincidentally also appeared in 1964, were the *parallel-tangents* or *PARTAN* algorithm of B.V. Shah, R.J. Buehler and O. Kempthorne [24] and the *heavy ball algorithm* of B.T. Polyak [21]. For a modern description of the former, and its subsequently discovered CG-related properties, see [14]. The basic iteration of the latter algorithm is as follows:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \mathbf{g}_i + \beta(\mathbf{x}_i - \mathbf{x}_{i-1}),$$

where α is a constant positive stepsize and β is a scalar with $0 < \beta < 1$. Although, strictly speaking, it is not CG-related in this form, the algorithm has CG-like rate of convergence properties on a quadratic under optimal choices of the algorithm parameters α and β (see [3]). The algorithms in [24] and [21] both used very simple steplength techniques to move from one iterate to the next, in contrast to the nonlinear CG implementation of [8]. The introduction of the line-search technique of Davidon [6] into either of these 1964 algorithms, as in [8], could have propelled them much more into the limelight at the time. More recently, the important contribution of Yu.E. Nesterov [17] on global rate of convergence is based on an algorithm akin to PARTAN [24]. However, in general, classical alternatives to CG remain on the sidelines.

Nonlinear CG Variants

These are premised on retaining the algorithmic structure, and the conjugacy properties on quadratics, of the basic nonlinear CG algorithm of the above Section when the initial direction is not along the negative gradient and/or line searches are inexact. For instance, the *three-term-recurrence algorithm* (TTR) is able to simultaneously relax both CG-standard requirements. The overall iteration follows the basic algorithm of the previous Section, but with the computation of the search direction at steps 3 and 4 replaced as follows:

$$\beta_i = \frac{\mathbf{y}_i^\top \mathbf{y}_i}{\mathbf{y}_i^\top \mathbf{d}_i}, \quad \gamma_i = \frac{\mathbf{y}_{i-1}^\top \mathbf{y}_i}{\mathbf{y}_{i-1}^\top \mathbf{d}_{i-1}},$$

and

$$\mathbf{d}_{i+1} = -\mathbf{y}_i + \beta_i \mathbf{d}_i + \gamma_i \mathbf{d}_{i-1}.$$

Conjugacy of search directions is retained when the initial search direction is chosen to be an arbitrary direction of descent. If this initial direction is along the negative gradient and line searches are exact, then the TTR generates the same search directions and iterates as the CG-standard on a positive definite quadratic. A drawback of the TTR algorithm is that it does not guarantee a descent direction on more general functions even if line searches are exact. But, in practice, the direction almost always satisfies the condition $\mathbf{g}_{i+1}^\top \mathbf{d}_{i+1} < 0$.

For references to other nonlinear CG variants, see [10] and the survey articles in [1]. Despite theoretical advantages on quadratics, algorithms in this category, in practice, have not proved to be significantly superior to the PPR nonlinear CG algorithm of the above Section.

Variable-Storage/Limited-Memory Algorithms

These are premised on a key structural relationship between the nonlinear CG algorithm and the BFGS variable-metric algorithm, and its properties on quadratics, see [4,12,15]. The most effective CG-related algorithm, to date, is the L-BFGS algorithm of J. Nocedal [18]. This is described in more detail in ► Unconstrained nonlinear optimization: Newton–Cauchy framework and is not repeated here. The algorithm has the property that it produces a descent direction under weak termination conditions on the line search. It has proved to be an efficient and versatile algorithm (it can exploit additional computer storage when available) that generally outperforms the PPR algorithm in practice.

For an overview of other variable-storage algorithms that draw on the *BFGS-CG relationship*, see the survey articles in [1].

Affine-Reduced-Hessian Algorithms

These make estimates of curvature, i. e., approximations to the Hessian or its inverse, in an affine sub-

space usually of low dimension and defined by the most recent gradient and one or more prior steps and/or gradients. For algorithms of this type and their CG-related properties, see [7,13,16] and references cited therein.

The foregoing principle, on which such algorithms are premised, has the conceptual advantage that it provides a *true continuum* between the nonlinear CG and full-storage variable-metric (and Newton) algorithms. But, practical affine-reduced-Hessian implementations are not yet widespread.

Conclusion

CG algorithms are among the simplest and most elegant algorithms of computational nonlinear optimization. They can be surprisingly effective in practice, and thus will always have an honored place in the repertoire. Nevertheless, the subject still lacks a comprehensive underlying theory, and many interesting algorithmic issues remain to be explored.

Some references cited in the present discussion are listed below, and other key references can be traced, in turn, through their bibliographies.

See also

- ► Broyden Family of Methods and the BFGS Update
- ► Large Scale Trust Region Problems
- ► Large Scale Unconstrained Optimization
- ► Local Attractors for Gradient-Related Descent Iterations
- ► Nonlinear Least Squares: Newton-Type Methods
- ► Nonlinear Least Squares: Trust Region Methods
- ► Unconstrained Nonlinear Optimization: Newton–Cauchy Framework
- ► Unconstrained Optimization in Neural Network Training

References

1. Adams L, Nazareth JL (1996) Linear and nonlinear conjugate gradient-related methods. SIAM, Philadelphia
2. Barrett R, Berry M, Chan TF, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C, Vorst H van der (1993) Templates for the solution of linear systems. SIAM, Philadelphia
3. Bertsekas DP (1999) Nonlinear programming, 2nd edn. Athena Sci, Belmont, MA
4. Buckley A (1978) Extending the relationship between the conjugate gradient and BFGS algorithms. Math Program 15:343–348
5. Dai YH (1997) Analyses of nonlinear conjugate gradient methods. PhD Diss Inst Computational Math Sci/Engin Computing Chinese Acad Sci, Beijing, China
6. Davidon WC (1991) Variable metric method for minimization. SIAM J Optim 1:1–17, (Original (with a different preface): Argonne Nat Lab Report ANL-5990 (Rev, Argonne, Illinois)
7. Fenelon MC (1981) Preconditioned conjugate-gradient-type methods for large-scale unconstrained optimization. PhD Diss Stanford Univ, Stanford, CA
8. Fletcher R, Reeves C (1964) Function minimization by conjugate gradients. Computer J 7:149–154
9. Golub GH, O'Leary DP (1989) Some history of the conjugate gradient and Lanczos algorithms 1948–1976. SIAM Rev 31:50–102
10. Hestenes MR (1980) Conjugate direction methods in optimization. Appl Math, vol 12. Springer, Berlin
11. Hestenes MR, Stiefel EL (1952) Methods of conjugate gradients for solving linear systems. J Res Nat Bureau Standards (B) 49:409–436
12. Kolda TG, O'Leary DP, Nazareth JL (1998) BFGS with update skipping and varying memory. SIAM J Optim 8:1060–1083
13. Leonard MW (1995) Reduced Hessian quasi-Newton methods for optimization. PhD Diss Univ Calif, San Diego, CA
14. Luenberger DG (1984) Linear and nonlinear programming, 2nd edn. Addison-Wesley, Reading, MA
15. Nazareth JL (1979) A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms. SIAM J Numer Anal 16:794–800
16. Nazareth JL (1986) The method of successive affine reduction for nonlinear minimization. Math Program 35:97–109
17. Nesterov YE (1983) A method of solving a convex programming problem with convergence rate O(1/k2). Soviet Math Dokl 27:372–376
18. Nocedal J (1980) Updating quasi-Newton matrices with limited storage. Math Comput 35:773–782
19. Nocedal J (1992) Theory of algorithms for unconstrained optimization. Acta Numer 1:199–242
20. Polak E, Ribière G (1969) Note sur la convergence de methode de directions conjuguees. Revue Franc Inform Rech Oper 16:35–43
21. Polyak BT (1964) Some methods of speeding up the convergence of iteration methods. USSR Comput Math Math Phys 4:1–17
22. Polyak BT (1969) The conjugate gradient method in extremal problems. USSR Comput Math Math Phys 9:94–112
23. Powell MJD (1977) Restart procedures for the conjugate gradient method. Math Program 12:241–254
24. Shah BV, Buehler RJ, Kempthorne O (1964) Some algorithms for minimizing a function of several variables. J SIAM 12:74–91

Contact Map Overlap Maximization Problem, CMO

Wei Xie[1], Nikolaos V. Sahinidis[2]
[1] American Airlines Operations Research and Decision Support Group, Fort Worth, USA
[2] Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, USA

Article Outline

Introduction
Definition
Methods
Exact algorithms
Approximate Algorithms
Conclusions
References

Introduction

Contact map overlap maximization is a problem that arises in computational biology as an important approach to compare structural similarity of proteins. Contact map overlap was proposed in [6] as a measure for protein structural similarity, and is employed in the Critical Assessment of Techniques for Protein Structure Prediction (CASP).

Proteins consist of amino acid residues and assume specific 3-dimensional structures. Proteins of similar structures often have similar function and properties. Therefore, structure alignment provides critical insights into the relation of existing proteins, and has important applications in designing knowledge-based potential functions that are useful for protein folding prediction.

Definition

Given two proteins A and B with m and n residues, respectively, we denote the residues of A with indices i, i', and i'', and the residues of B with indices j, j' and j''. If two residues in the same protein are close in space, we say that they are *in contact*. A list of residue pairs is known as the *contact map* of a given protein. In this article, the contact map for protein A (resp. B) is denoted as E^A (resp. E^B) so that $E^A_{ii'} = 1$ (resp. $E^B_{jj'} = 1$) if residues i and i' in protein A (resp. j and j' in protein B) are in contact, and $E^A_{ii'} = 0$ (resp. $E^B_{jj'} = 0$) otherwise.

From a graph-theoretic perspective, a contact map is a node-node incident graph where nodes represent residues and edges represent contacts. The *contact map overlap* maximization problem aims at identifying an ordered residue correspondence between two contact maps so as to result in a maximum common subgraph. To solve this problem, a correspondence (alignment) must be established between the node sets (residues) of the contact maps so that the number of common contacts (edges) can be maximized. If residue i in protein A aligns with residue j in protein B, they form a *pair* (i,j). If pairs (i,j) and (i',j') result in common contacts, i. e., $E^A_{ii'} = 1$ and $E^B_{jj'} = 1$, then they form an *overlap*. If two pairs (i,j) and (i',j') form an overlap, we set $h(i,j,i',j') = 1$. Otherwise, $h(i,j,i',j')$ is set to zero.

An important requirement for structure alignment is that the relative orders of residues in the original sequences agree – a property that is known as the *non-crossing* property in the CMO literature. For two pairs (i,j) and (i',j') to be non-crossing, either $i < i'$ and $j < j'$ or $i > i'$ and $j > j'$ must hold. In this paper, non-crossing pairs are also referred to as *parallel* pairs.

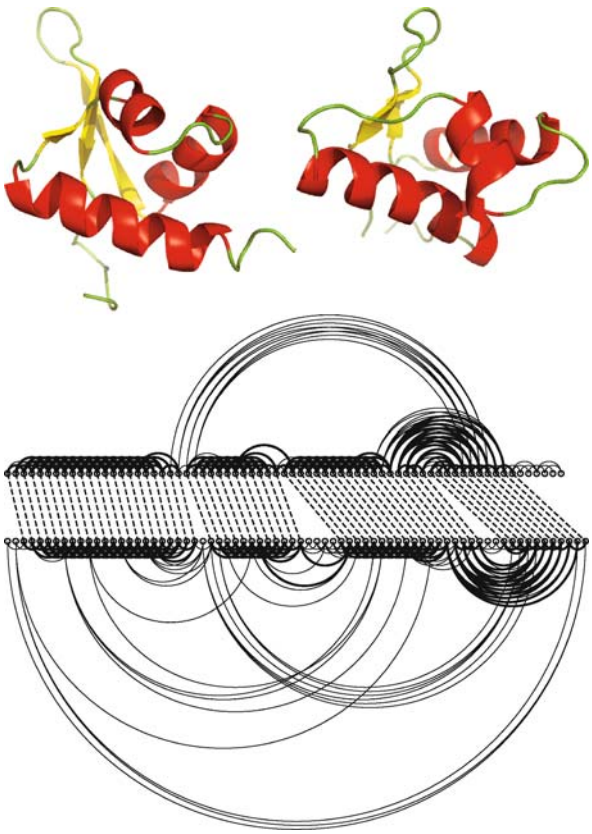
For convenience of the presentation, $[i,i']$ (resp. $[j,j']$) will denote the set of residues $\{i'' : i \le i'' \le i'\}$ (resp. $\{j'' : j \le j'' \le j'\}$). The interval product $[i,i'] \times [j,j']$ therefore denotes the set of pairs $\{(i'',j'') : i \le i'' \le i', j \le j'' \le j')\}$. For any given set of residue pairs S, we will use $\mathbb{Q}(S)$ to denote the set of subsets of S that contain only parallel pairs. The objective of CMO is to identify a set of parallel pairs that maximize the resultant number of overlaps. For proteins A and B, the problem can be stated as follows:

$$\max_{Q \in \mathbb{Q}([1,m]\times[1,n])} \frac{1}{2} \sum_{(i,j)\in Q} \sum_{(i',j')\in Q} h(i,j,i',j') \,.$$

An optimal alignment of the contact maps of the human Rap30 DNA-binding domain (1BBY) and the DNA binding domain of Escherichia coli LexA repressor (1LEA) is shown in Fig. 1 together with their 3-dimensional structures.

Methods

Goldman et al. [7] proved that CMO is APX-hard, which practically defies the existence of a polynomial time exact algorithm or even a polynomial time approximation scheme. In the remainder of this section, we



Contact Map Overlap Maximization Problem, CMO, Figure 1
An instance of CMO involving 1BBY and 1LEA

will survey both exact and approximate algorithms for this problem.

Exact algorithms

Four exact algorithms have been proposed for the CMO problem so far. All embrace the branch-and-bound framework to ensure global optimality. The branch-and-reduce algorithm [12,13] currently has an edge against all other exact algorithms both in terms of computational speed as well as its ability to solve challenging instances.

Integer Linear Program Carr et al. [5] proposed an integer linear programming formulation for the problem. This formulation was further developed in [9] and involves two sets of binary variables: x_{ij} and $y_{iji'j'}$. Variable x_{ij} equals one if pair (i,j) is chosen in the optimal alignment and zero otherwise. Variable $y_{iji'j'}$ equals one if both pairs (i,j) and (i',j') are chosen in the final solution. Otherwise, variable $y_{iji'j'}$ is set to zero. The model is as follows:

$(M1)$

$$\max \sum_{(i,j,i',j'):E^A(i,i')=1,E^B(j,j')=1,i<i',j<j'} y_{iji'j'} \quad (1)$$

$$\text{s.t.} \sum_{i':E^A(i,i')=1,i'>i} y_{iji'j'} \le x_{ij} \quad (2)$$

$$\sum_{j':E^B(j,j')=1,j'>j} y_{iji'j'} \le x_{ij} \quad (3)$$

$$\sum_{i:E^A(i,i')=1,i<i'} y_{iji'j'} \le x_{i'j'} \quad (4)$$

$$\sum_{j:E^B(j,j')=1,j<j'} y_{iji'j'} \le x_{i'j'} \quad (5)$$

$$x_{ij} + x_{i'j'} \le 1 \quad \forall \text{ crossing pairs } (i,j) \text{ and } (i',j') \quad (6)$$

$$x_{ij} \in \{0,1\}, y_{iji'j'} \in \{0,1\}$$

The sum of common contacts in Eq. (1) constitutes the number of overlaps by definition since Eq. (6) prohibits the existence of crossing pairs in the final solution. Equations (2)–(5) ensure that the optimal solution contains at most one pair from each residue that does not cross a chosen pair. In addition to these necessary constraints, two classes of cuts, *clique-cut*s and *odd-hole-cut*s, can be optionally generated in polynomial time and append to (M1) as was shown in [9].

Lagrangian Relaxation In [3], Caprara and Lancia proposed a Lagrangian relaxation approach for the CMO problem. Their approach begins with an integer linear program formulation of CMO as is shown in model (M2) below. In this model, x_{ij} and $y_{iji'j'}$ bear same definitions as their counterparts in (M1), and $\mathcal{I}$ is the set of maximal set of crossing pairs.

$(M2)$

$$\max \sum_{(i,j,i',j')} \frac{1}{2} h_{iji'j'} y_{iji'j'} \quad (7)$$

$$\text{s.t.} \sum_{(i,j):(i,j)\in I} x_{ij} \le 1, \qquad \forall I \in \mathcal{I} \quad (8)$$

$$\sum_{(i,j)} y_{iji'j'} \leq x_{i'j'}, \qquad \forall I \in \mathcal{I}, (i', j') \tag{9}$$

$$y_{iji'j'} = y_{i'j'ij}, \qquad \forall i < i', j < j' \tag{10}$$

$$x_{ij}, y_{iji'j'} \in \{0, 1\} \tag{11}$$

It is easy to verify that Eq. (7) yields the number of overlaps given the validity of Eq. (10). Since any two pairs in a maximal set of crossing pairs would cross, constraint (8) enforces the non-crossing property by prohibiting any such two pairs to co-exist in the final alignment. Constraint (9) ensures that, if an arbitrary pair (i', j') is chosen, the final solution should contain no more than one pair (i, j) from any maximal crossing pair set that could form an overlap with pair (i', j').

By introducing multipliers $\lambda_{iji'j'}$ for Eq. (9), Caprara and Lancia obtained the following Lagrangian relaxation of (M2):

$(LM2)$

$$\min_{\lambda} \max_{i,j} \sum_{(i,j,i',j')} \frac{1}{2} h_{iji'j'} y_{iji'j'} + \sum_{(i,j,i',j'):i<i',j<j'} \lambda_{iji'j'} \left(y_{iji'j'} - y_{i'j'ij}\right)$$

s.t. Constraints (8), (9), and (11).

A subgradient method was used to iteratively improve the multipliers, while an $\mathcal{O}(|E^A||E^B|)$ algorithm was employed to solve (LM2) for the set of multipliers at each iteration.

Reformulation as Maximum Clique Strickland et al. [11] showed that CMO can be cast into a maximum clique problem. To this end, they considered a two-dimensional lattice whose rows correspond to the contacts in E^A and columns correspond to the contacts in E^B. Each vertex of the lattice, if chosen, contributes a unit overlap to the objective value. In addition, an edge is drawn between two vertices if the corresponding pair of overlaps are admissible to some feasible alignments. It is not difficult to see that a maximum clique in the resultant graph indeed corresponds to an optimal solution for the CMO problem. This algorithm involves a number of preprocessing routines to reduce the problem size before calling a maximum clique solver. In a more recent work [10], the authors proposed improved data structures to enhance the algorithm performance.

Combinatorial Branch-and-Reduce Xie and Sahinidis [12,13] developed a branch-and-reduce algorithm, which combines the generic branch-and-bound framework with problem-specific reduction schemes. The algorithm initializes a branch-and-bound tree with a root node where all pairs are allowed. Reduction schemes, both based on domination and the current best solution value, are used to remove inferior pairs. Lower and upper bounds on the overlaps for the current node are then computed using dynamic programming. If the lower and upper bounds agree, the search is terminated with a global optimal alignment. Otherwise, the algorithm chooses a branching pair and creates two children nodes. The branching pair is enforced in one of the node, while it is disallowed in the other node.

A key step in their algorithm is the computation of the contribution to the overlaps by a given pair on a set of pairs. Define $\mathbb{Q}(S)$ to be the set of all subsets of S that contain only parallel pairs. Then, the contribution of pair (i, j) to the objective value on the set S is defined as

$$p(i, j, S) := \max_{Q \in \mathbb{Q}(S)} \sum_{(i',j') \in Q} h(i, j, i', j') .$$

In particular, let $p^+(i, j, i', j')$ and $p^-(i, j, i', j')$ denote $p(i, j, S)$ when $S = [i', m] \times [j', n]$ and when $S = [1, i'] \times [1, j']$, respectively. They proved that computing a single term of $p^+(\cdot)$ or $p^-(\cdot)$ can be accomplished in $\mathcal{O}(mn)$ time with preprocessing time and space complexity of $\mathcal{O}(m + n)$.

The upper bounding scheme is summarized in Proposition 1, where, for a node V of the search tree, $\mathbb{C}(V)$ is the set of pairs that must be included and $\mathbb{F}(V)$ is the set of pairs that have the freedom to be in the solution or not. In addition, define

$$g(i, j) := \sum_{(i',j') \in \mathbb{C}(V)} h(i, j, i', j'). \forall (i, j) \in \mathbb{F}(V)$$

Then:

Proposition 1 *Define*

$$t(i,j) := \begin{cases} g(i,j), \\ \qquad \textit{if } (i,j) \in \mathbb{F}(V); \quad i = 1 \textit{ or } j = 1\,, \\ \max\{g(i,j), g(i,j) + w(i,j)\}\,, \\ \qquad \textit{if } (i,j) \in \mathbb{F}(V); i > 1 \textit{ and } j > 1\,, \\ -\infty, \quad \textit{otherwise}\,, \end{cases}$$

where

$$\bar{h}(i,j,i',j') := \begin{cases} h(i,j,i',j')\,, \\ \qquad \textit{if } (i,j),(i',j') \in \mathbb{F}(V) \\ 0, \quad \textit{otherwise}\,, \end{cases}$$

$$w(i,j) := \begin{cases} -\infty, \\ \quad \textit{if } [1,i-1] \times [1,j-1] \cap \mathbb{F}(V) = \emptyset\,, \\ \max_{(i',j') \in [1,i-1] \times [1,j-1] \cap \mathbb{F}(V)} \left\{ t(i',j') \right. \\ \qquad \left. + \bar{h}(i',j',i,j) + \frac{1}{2} u(i',j',i,j) \right\}, \\ \quad \textit{otherwise}\,, \end{cases}$$

and

$$u(i',j',i,j) := p^+(i',j',i+1,j+1) + p^-(i,j,i'-1,j'-1)\,.$$

Then

$$\frac{1}{2} \sum_{(i,j) \in \mathbb{C}(V)} \sum_{(i',j') \in \mathbb{C}(V)} h(i,j,i',j') + \left\lfloor \max_{(i,j) \in \mathbb{F}(V)} t(i,j) \right\rfloor$$

is an upper bound for the current node V.

Proposition 1 suggests an $\mathcal{O}(m^2 n^2)$ algorithm to compute the upper bound. In addition, the upper bounding scheme also provides a natural lower bound as was shown in [13].

Approximate Algorithms

This subsection outlines both approximation algorithms (i. e., with performance guarantee) and heuristics (i. e., without performance guarantee) that have been proposed in the literature for the CMO problem.

Goldman et al. [7] considered a special class of CMO instances from 2-dimensional self-avoiding walk, and proposed a 3-approximation algorithm that runs in $\mathcal{O}(n^6)$ time. Agarwal et al. [1] proposed a 6-approximation algorithm for the same class of problems, with however a better complexity of $\mathcal{O}(n^3 \log n)$. In addition, they proved the special class of CMO from 3-dimensional self-avoiding walk is MAXSNP-hard, and proposed a $\mathcal{O}(\sqrt{n})$-approximation algorithm for this class of CMO instances.

Carr et al. [4] proposed to use a memetic algorithm, which combines global search and local search, to solve general instances of CMO. In [2], Caprara et al. proposed several heuristics, including a genetic algorithm, local search, and greedy algorithms based on Lagrangian relaxation. Existing computational studies [8,13] suggest that Lagrangian-relaxation-based greedy algorithms perform the best among existing heuristics for a large set of test instances.

Conclusions

The contact map overlap maximization problem is a very important problem in computational biology. Efficient algorithms, both exact and approximate, for this problem are of great interest. Despite the inherent difficulty of this problem, many large-scale practical instances have been solved within reasonable amounts of time. Many challenging instances of the problem currently remain unsolved [13].

References

1. Agarwal PK, Mustafa NH, Wang Y (2007) Fast molecular shape matching using contact maps. J Comput Biol 14(2):131–143
2. Caprara A, Carr R, Istrail S, Lancia G, Walenz B (2004) 1001 optimal PDB structure alignments: Integer programming methods for finding the maximum contact map overlap. J Comput Biol 11(1):27–52
3. Caprara A, Lancia G (2002) Structural alignment of large-size proteins via Lagrangian relaxation. In: Proceedings of International Conference on Computational Biology (RECOMB), Washington, April 2002. pp 100–108
4. Carr B, Hart WE, Krasnogor N, Burke EK, Hirst JD, Smith JE (2002) Alignment of protein structures with a memetic evolutionary algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, New York, July 2002. Morgan Kaufmann Publishers, pp 1027–1034
5. Carr RD, Lancia G, Istrail S (2000) Branch-and-cut algorithms for independent set problems: Integrality gap and an application to protein structural alignment, Technical report. Sandia National laboratories
6. Godzik A, Skolnick J (1994) Flexible algorithm for direct multiple alignment of protein structures and sequences. Comput Appl Biosci 10(6):587–596

7. Goldman D, Papadimitriou C, Istrail S (1999) Algorithmic aspects of protein structure similarity. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS), New York, October 1999. IEEE Computer societypages, pp 512–522
8. Krasnogor N, Lancia G, Zemla A, Hart WE, Carr RD, Hirst JD, Burke EK (2005) A comparison of computational methods for the maximum contact map overlap of protein pairs. http://citeseer.ist.psu.edu/659931.html. Accessed 30 Jan 2008
9. Lancia G, Carr R, Walenz B, Istrail S (2001) 101 optimal PDB structure alignments: A branch-and-cut algorithm for the maximum contact map overlap problem. In: Proceedings of Annual International Conference on Computational Biology (RECOMB), Montreal, April 2001. pp 193–202
10. Melvin J, Sokol J, Tovey C (2006) Finding optimal solutions to large CMO instances. (preprint)
11. Strickland DM, Barnes E, Sokol JS (2005) Optimal protein structure alignment using maximum cliques. Oper Res 53(3):389–402
12. Xie W, Sahinidis NV (2006) A branch-and-reduce algorithm for the contact map overlap problem. In: Apostolico A, Guerra C, Istrail S, Pevzner P, Waterman M (eds) RECOMB 2006. Lect Note Comput Sci 3909:516–529
13. Xie W, Sahinidis NV (2006) A reduction-based exact algorithm for the contact map overlap problem. J Comput Biol 14(5):637–654

Continuous Approximations to Subdifferentials

Adil Bagirov
Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Ballarat, Australia

MSC2000: 65K05, 90C56

Article Outline

Introduction
Definitions
  Continuous Approximations
Methods
  Computation of the Continuous Approximations
  Computation of Subgradients
  Discrete Gradients
  Continuous Approximations to the Quasidifferential
Conclusions
References

Introduction

In this paper we use the following notation: $\mathbb{R}^n$ is an n-dimensional space, where the scalar product will be denoted by $\langle x, y\rangle$:

$$\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i \ ,$$

and $\|\cdot\|$ will denote the associated norm. The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}^1$ will be denoted by ∇f.

A function f is differentiable at a point $x \in \mathbb{R}^n$ with respect to a direction $g \in \mathbb{R}^n$ if the limit

$$f'(x, g) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha g) - f(x)}{\alpha}$$

exists. The closed unit ball will be denoted by B: $B = \{x \in \mathbb{R}^n : \|x\| \le 1\}$.

Consider a function f defined on $\mathbb{R}^n$. This function is called locally Lipschitz continuous if for any bounded subset $X \subset \mathbb{R}^n$ there exists an $L > 0$ such that

$$|f(x) - f(y)| \le L\|x - y\| \quad \forall x, y \in X \ .$$

If f is convex then one can define a subdifferential $\partial f(x)$ of this function at a point $x \in \mathbb{R}^n$ as follows [13]:

$$\partial f(x) = \{v \in \mathbb{R}^n : f(y) - f(x) \ge \langle v, y - x\rangle, y \in \mathbb{R}^n\} \ .$$

Elements $v \in \partial f(x)$ of the subdifferential $\partial f(x)$ are called subgradients of f at the point x. For a convex function f defined on $\mathbb{R}^n$ the subdifferential $\partial f(x)$ is nonempty, convex and compact at any $x \in \mathbb{R}^n$. A set-valued mapping $x \mapsto \partial f(x)$ is upper semicontinuous.

An ε-subdifferential $\partial_\varepsilon f(x)$, $\varepsilon > 0$ of the convex function f at a point $x \in \mathbb{R}^n$ is defined as [13]

$$\partial_\varepsilon f(x) = \{v \in \mathbb{R}^n : f(y) - f(x) \ge \langle v, y - x\rangle - \varepsilon, y \in \mathbb{R}^n\} \ .$$

Elements $v \in \partial_\varepsilon f(x)$ of the subdifferential $\partial_\varepsilon f(x)$ are called ε-subgradients of f at the point x. For a convex function f defined on $\mathbb{R}^n$ the ε-subdifferential $\partial_\varepsilon f(x)$ is nonempty, convex and compact at any $x \in \mathbb{R}^n$ and a set-valued mapping $x \mapsto \partial_\varepsilon f(x)$ is continuous in Hausdorff metric at any x.

Most efficient methods in nonsmooth optimization such as the bundle method and its variations are based

on ε-subgradients of convex functions (see, for example, [10,11]).

The analysis of nonsmooth, nonconvex functions has been area of intensive research for more than three decades. Clarke [4,5] introduced the notion of generalized gradient. We define a Clarke subdifferential for locally Lipschitz continuous functions defined on $\mathbb{R}^n$. A locally Lipschitz function f is differentiable almost everywhere and one can define for it a Clarke subdifferential [5] by

$$\partial f(x) = \mathrm{co}\Big\{v \in \mathbb{R}^n : \exists (x^k \in D(f), x^k \to x, \; k \to +\infty) : v = \lim_{k \to +\infty} \nabla f(x^k)\Big\},$$

where $D(f)$ denotes the set where f is differentiable, and co denotes the convex hull of a set. It is shown in [5] that the mapping $\partial f(x)$ is upper semicontinuous and bounded on bounded sets. The generalized directional derivative of f at x in the direction g is defined to be

$$f^0(x,g) = \limsup_{y \to x, \alpha \downarrow 0} \alpha^{-1}[f(y+\alpha g) - f(y)] .$$

The generalized directional derivative always exists and

$$f^0(x,g) = \max\{\langle v, g\rangle : v \in \partial f(x)\} .$$

f is called a Clarke regular function on $\mathbb{R}^n$ if it is differentiable with respect to any direction $g \in \mathbb{R}^n$ and $f'(x,g) = f^0(x,g)$ for all $x, g \in \mathbb{R}^n$. For nonregular functions the Clarke subdifferential has calculus only by means of inclusions, which makes very difficult the computation of subgradients of some complex nonsmooth, nonconvex functions. The cluster function from the cluster analysis is one such example [2,3].

Demyanov and Rubinov [6,7] introduced the notion of quasidifferential. Let f be a locally Lipschitz continuous function defined on $\mathbb{R}^n$. This function is called quasidifferentiable at a point $x \in \mathbb{R}^n$ if it is directionally differentiable and there exist compact, convex sets $\underline{\partial} f(x)$ and $\overline{\partial} f(x)$ such that

$$f'(x,g) = \max\{\langle v, g\rangle, \sim v \in \underline{\partial} f(x)\} + \min\{\langle w, g\rangle, \sim w \in \overline{\partial} f(x)\} .$$

The pair

$$Df(x) = [\underline{\partial} f(x), \overline{\partial} f(x)]$$

is called a quasidifferential of the function f at a point x. The set $\underline{\partial} f(x)$ is said to be a subdifferential and the set $\overline{\partial} f(x)$ a superdifferential of the function f at x.

Unlike the Clarke subdifferential, the quasidifferential enjoys a full-scale calculus; however, set-valued mappings $x \mapsto \underline{\partial} f(x)$ and $x \mapsto \overline{\partial} f(x)$ need not be even upper semicontinuous.

Unfortunately, the notion of an ε-subdifferential cannot be extended for nonsmooth, nonconvex functions. Instead one can define the Goldstein ε-subdifferential [9]. However, in general the Goldstein ε-subdifferential is only upper semicontinuous.

In this paper, we consider continuous approximations to subdifferentials and quasidifferentials. We will also describe an algorithm for computation of elements of such approximations.

Definitions

Continuous Approximations

Let X be a compact subset of the space $\mathbb{R}^n$. We consider a family $C(x,\varepsilon) = C_\varepsilon(x)$ of set-valued mappings depending on a parameter $\varepsilon > 0$. For each $\varepsilon > 0$

$$C(\cdot,\varepsilon) : X \to 2^{\mathbb{R}^n} .$$

We suppose that $C(x,\varepsilon)$ is a compact convex set for all $x \in X$ and $\varepsilon > 0$. It is assumed that there exists a number $K > 0$ such that

$$\sup\{\|v\| : v \in C(x,\varepsilon), x \in X, \varepsilon > 0\} \le K . \qquad (1)$$

Definition 1 The limit $C_L(x)$ of the family $\{C(x,\varepsilon)\}, \varepsilon > 0$ at a point x is defined as follows:

$$C_L(x) = \Big\{v \in \mathbb{R}^n : \exists (x^k \to x, \varepsilon_k \to +0, k \to +\infty, \; v^k \in C(x^k, \varepsilon_k)) : v = \lim_{k \to +\infty} v^k\Big\} .$$

It is possible that the limit $C_L(x)$ is not convex even if all the sets $C(x,\varepsilon)$ are convex. We consider $\mathrm{co}C_L(x)$ the convex hull of $C_L(x)$. It follows from Definition 1 and the inequality (1) that the mapping $\mathrm{co}C_L$ has compact convex images.

Definition 2 A family $\{C(x,\varepsilon)\}, \; \varepsilon > 0$ is called a continuous approximation to a subdifferential ∂f on X if the following holds:

1. $C(x,\varepsilon)$ is a Hausdorff continuous mapping with respect to x on X for all $\varepsilon > 0$.
2. The subdifferential $\partial f(x)$ is the convex hull of the limit of the family $\{C(x,\varepsilon)\}, \varepsilon > 0$ on X, i. e. for all $x \in X$

$$\partial f(x) = \mathrm{co} C_L(x) .$$

Some properties of the continuous approximations were studied in [1].

Such continuous approximations need not be monotonically decreasing as $\varepsilon \to +0$. Uniform and strongly continuous approximations to the subdifferential studied in [14] have such a property. Let f be a locally Lipschitz continuous function defined on an open set which contains a compact set X. We consider a family of set-valued mappings $A_\varepsilon f\colon \mathbb{R}^n \to 2^{\mathbb{R}^n}, \varepsilon > 0$. Assume that the sets $A_\varepsilon f(x)$ are nonempty and compact for all $\varepsilon > 0$ and $x \in X$. We will denote by $\partial f(x + B_\delta)$ the set $\bigcup\{\partial f(y)\colon y \in B_\delta(x)\}$, where $B_\delta(x) = \{y \in \mathbb{R}^n : \|x - y\| \le \delta\}$.

Definition 3 ([14]) We say that the family $\{A_\varepsilon f(\cdot)\}_{\varepsilon>0}$ is a uniform continuous approximation to the subdifferential ∂f on X, if the following conditions are satisfied:

1. For each given $\varepsilon > 0, \mu > 0$ there exists $\tau > 0$, such that for all $x \in X$

$$\partial f(x + B_\tau) \subset A_\varepsilon f(x) + B_\mu .$$

2. For each $x \in X$ and for all $0 < \varepsilon_1 < \varepsilon_2$:

$$A_{\varepsilon_1} f(x) \subset A_{\varepsilon_2} f(x) .$$

3. $A_\varepsilon f(x)$ is Hausdorff continuous with respect to x on X.
4. For each $x \in X$

$$\bigcap_{\varepsilon>0} A_\varepsilon f(x) = \partial f(x) .$$

Definition 4 ([14]) We say that the family $\{A_\varepsilon f(\cdot)\}_{\varepsilon>0}$ is a strong continuous approximation to the subdifferential ∂f on X, if $\{A_\varepsilon f(\cdot)\}_{\varepsilon>0}$ satisfies properties 1–3 above and instead of property 4 the following is valid:

4′. For every $\gamma, \mu > 0$ there exists $\varepsilon > 0$ such that for all $x \in X$

$$\partial f(x) \subset A_\varepsilon f(x) \subset \partial f(x + B_\gamma) + B_\mu .$$

For the set-valued mapping $C(x,\varepsilon)$ we set

$$C_0(x) = \Big\{v \in \mathbb{R}^n\colon \exists(\varepsilon_k \to +0, k \to +\infty, v^k \in C(x,\varepsilon_k))\colon v = \lim_{k\to+\infty} v^k\Big\}$$

and let

$$C(x,0) = C_0(x) .$$

Theorem 1 ([1]) *Let the family $\{A_\varepsilon f(\cdot)\}_{\varepsilon>0}$ be a uniform continuous approximation to the subdifferential ∂f on a compact set X. Then $C(x,\varepsilon) = A_\varepsilon f(x)$ is a continuous approximation to the subdifferential ∂f in the sense of Definition 2.*

Corollary 1 *It was shown in [14] that a strong continuous approximation is a uniform continuous approximation. So a strong continuous approximation is a continuous approximation in the sense of Definition 2.*

Theorem 2 ([1]) *Let the family $C(x,\varepsilon)$ be a continuous approximation to the subdifferential ∂f on a compact set X and the mapping $C(x,\varepsilon)$ be continuous with respect to $(x,\varepsilon), x \in X, \varepsilon > 0$. Assume $\mathrm{co} C_L(x) = C_0(x)$ for all $x \in X$. Then the mapping*

$$Q(x,\varepsilon) = \mathrm{co}\bigcup\{C(x,t) : 0 \le t \le \varepsilon\}$$

is a uniform continuous approximation to ∂f on X.

Corollary 2 *Let the family $C(x,\varepsilon)$ be a continuous approximation to the subdifferential ∂f on a compact set X and the mapping $C(x,\varepsilon)$ be a continuous with respect to $(x,\varepsilon), x \in X, \varepsilon > 0$. Assume $\mathrm{co} C_L(x) = C_0(x)$ for all $x \in X$. Then the mapping Q is upper semicontinuous with respect to (x,ε) at the point $(x,0)$.*

One can get a chain rule for continuous approximations [1,14]. However it is not always applicable to compute elements of continuous approximations. In the next section we propose an algorithm to compute those elements.

Methods

Computation of the Continuous Approximations

We consider a locally Lipschitz continuous function f defined on $\mathbb{R}^n$ and assume that this function is semismooth and quasidifferentiable (for the definition of semismooth functions see [12]). We also assume that

both sets $\underline{\partial} f(x)$ and $\overline{\partial} f(x)$ are represented as a convex hull of a finite number of points at any $x \in \mathbb{R}^n$, that is at a point $x \in \mathbb{R}^n$ there exist sets

$$A = \{a^1, \ldots, a^m\},\ a^i \in \mathbb{R}^n,\ i = 1, \ldots, m, m \geq 1$$

and

$$B = \{b^1, \ldots, b^p\},\ b^j \in \mathbb{R}^n,\ j = 1, \ldots, p, p \geq 1$$

such that

$$\underline{\partial} f(x) = \mathrm{co} A,\ \overline{\partial} f(x) = \mathrm{co} B\,.$$

In other words we assume that the subdifferential and the superdifferential of the function f are polytopes at any $x \in \mathbb{R}^n$. This assumption is true, for example, for functions represented as a maximum, minimum or max–min of a finite number of smooth functions.

We take a direction $g \in \mathbb{R}^n$ such that

$$g = (g_1, \ldots, g_n),\ |g_i| = 1,\ i = 1, \ldots, n$$

and consider the sequence of n vectors $e^j = e^j(\alpha)$, $j = 1, \ldots, n$ with $\alpha \in (0, 1]$:

$$\begin{aligned} e^1 &= (\alpha g_1, 0, \ldots, 0)\,,\\ e^2 &= (\alpha g_1, \alpha^2 g_2, 0, \ldots, 0)\,,\\ \ldots &= \ldots\ldots\ldots,\\ e^n &= (\alpha g_1, \alpha^2 g_2, \ldots, \alpha^n g_n)\,. \end{aligned}$$

We introduce the following sets:

$$\begin{aligned} &\underline{R}_0 = A,\ \overline{R}_0 = B\,,\\ &\underline{R}_j = \left\{v \in \underline{R}_{j-1} \colon v_j g_j = \max\{w_j g_j \colon w \in \underline{R}_{j-1}\}\right\}\,,\\ &\overline{R}_j = \left\{v \in \overline{R}_{j-1} \colon v_j g_j = \min\{w_j g_j \colon w \in \overline{R}_{j-1}\}\right\}\,. \end{aligned}$$

It is clear that

$$\underline{R}_j \neq \emptyset,\ \forall j \in \{0, \ldots, n\},\ \underline{R}_j \subseteq \underline{R}_{j-1},\ \forall j \in \{1, \ldots, n\}$$

and

$$\overline{R}_j \neq \emptyset,\ \forall j \in \{0, \ldots, n\},\ \overline{R}_j \subseteq \overline{R}_{j-1},\ \forall j \in \{1, \ldots, n\}\,.$$

Moreover

$$v_r = w_r\ \forall v, w \in \underline{R}_j,\ r = 1, \ldots, j \tag{2}$$

and

$$v_r = w_r\ \forall v, w \in \overline{R}_j,\ r = 1, \ldots, j\,. \tag{3}$$

Consider the following two sets:

$$\underline{R}(x, e^j(\alpha)) = \left\{v \in A \colon \langle v, e^j\rangle = \max_{u \in A}\langle u, e^j\rangle\right\}\,,$$

$$\overline{R}(x, e^j(\alpha)) = \left\{w \in B \colon \langle w, e^j\rangle = \min_{u \in B}\langle u, e^j\rangle\right\}\,.$$

Proposition 1 *Assume that the function f is quasidifferentiable and its subdifferential and superdifferential are polytopes at a point x. Then there exists $\alpha_0 > 0$ such that*

$$\underline{R}(x, e^j(\alpha)) \subset \underline{R}_j,\ \overline{R}(x, e^j(\alpha)) \subset \overline{R}_j,\ j = 1, \ldots, n$$

for all $\alpha \in (0, \alpha_0)$.

Corollary 3 *Assume that the function f is quasidifferentiable and its subdifferential and superdifferential are polytopes at a point x. Then there exists $\alpha_0 > 0$ such that*

$$f'(x, e^j(\alpha)) = f'(x, e^{j-1}(\alpha)) + v_j \alpha^j g_j + w_j \alpha^j g_j,\quad \forall v \in \underline{R}_j,\ w \in \overline{R}_j,\ j = 1, \ldots, n$$

for all $\alpha \in (0, \alpha_0]$.

Proposition 2 *Assume that the function f is quasidifferentiable and its subdifferential and superdifferential are polytopes at a point x. Then the sets $\underline{R}_n$ and $\overline{R}_n$ are singletons.*

In the next subsection we propose an algorithm to approximate subgradients. This algorithm finds a subgradient which can be represented as a sum of elements of the sets $\underline{R}_n$ and $\overline{R}_n$.

Computation of Subgradients

Let $g \in \mathbb{R}^n$, $|g_i| = 1, i = 1, \ldots, n$ be a given vector and $\lambda > 0, \alpha > 0$ be given numbers. We define the following points

$$x^0 = x,\quad x^j = x^0 + \lambda e^j(\alpha),\quad j = 1, \ldots, n\,.$$

It is clear that

$$x^j = x^{j-1} + (0, \ldots, 0, \lambda \alpha^j g_j, 0, \ldots, 0),\ j = 1, \ldots, n.$$

Let $v = v(\alpha, \lambda) \in \mathbb{R}^n$ be a vector with the following coordinates:

$$v_j = (\lambda \alpha^j g_j)^{-1}\left[f(x^j) - f(x^{j-1})\right],\ j = 1, \ldots, n. \tag{4}$$

For any fixed $g \in \mathbb{R}^n, |g_i| = 1, i = 1, \dots, n$ and $\alpha > 0$ we introduce the following set:

$$V(g,\alpha) = \Big\{ w \in \mathbb{R}^n : \exists (\lambda_k \to +0,\ k \to +\infty),\ w = \lim_{k \to +\infty} v(\alpha, \lambda_k) \Big\} .$$

Proposition 3 *Assume that f is a quasidifferentiable function and its subdifferential and superdifferential are polytopes at x. Then there exists $\alpha_0 > 0$ such that*

$$V(g,\alpha) \subset \partial f(x)$$

for all $\alpha \in (0, \alpha_0]$.

Remark 1 It follows from Proposition 3 that in order to approximate subgradients of quasidifferentiable functions one can choose a vector $g \in \mathbb{R}^n$ such that $|g_i| = 1, i = 1, \dots, n$, sufficiently small $\alpha > 0, \lambda > 0$ and apply (4) to compute a vector $v(\alpha, \lambda)$. This vector is an approximation to a certain subgradient.

Remark 2 A class of quasidifferentiable functions presents a broad class of nonsmooth functions, including many interesting nonregular functions. Thus, the scheme proposed in this section allows one to approximate subgradients of a broad class of nonsmooth functions.

Discrete Gradients

In the previous subsection we demonstrated an algorithm for the computation of subgradients. In this subsection we consider an algorithm for the computation of subdifferentials. This algorithm is based on the notion of a discrete gradient. We start with its definition [1].

Let f be a locally Lipschitz continuous function defined on $\mathbb{R}^n$. Let

$$\begin{aligned} S_1 &= \{ g \in \mathbb{R}^n : \|g\| = 1 \}, \\ G &= \{ e \in \mathbb{R}^n : e = (e_1, \dots, e_n), |e_j| = 1, \\ &\quad j = 1, \dots, n \}, \\ P &= \{ z(\lambda) : z(\lambda) \in \mathbb{R}^1,\ z(\lambda) > 0,\ \lambda > 0, \\ &\quad \lambda^{-1} z(\lambda) \to 0,\ \lambda \to 0 \} . \end{aligned}$$

Here S_1 is the unit sphere, G is the set of vertices of the unit hypercube in $\mathbb{R}^n$ and P is the set of univariate positive infinitesimal functions.

We take any $g \in S_1$ and define $|g_i| = \max\{|g_k|, k = 1, \dots, n\}$. We also take any $e = (e_1, \dots, e_n) \in G$, a positive number $\alpha \in (0, 1]$ and define the sequence of n vectors $e^j(\alpha), j = 1, \dots, n$. Then for given $x \in \mathbb{R}^n$ and $z \in P$ we define a sequence of $n + 1$ points as follows:

$$\begin{aligned} x^0 &= x + \lambda g , \\ x^1 &= x^0 + z(\lambda) e^1(\alpha) , \\ x^2 &= x^0 + z(\lambda) e^2(\alpha) , \\ \dots &= \dots \dots , \\ x^n &= x^0 + z(\lambda) e^n(\alpha) . \end{aligned}$$

Definition 5 The discrete gradient of the function f at the point $x \in \mathbb{R}^n$ is the vector $\Gamma^i(x, g, e, z, \lambda, \alpha) = (\Gamma_1^i, \dots, \Gamma_n^i) \in \mathbb{R}^n, g \in S_1$ with the following coordinates:

$$\begin{aligned} \Gamma_j^i &= [z(\lambda)\alpha^j e_j)]^{-1} \left[f(x^j) - f(x^{j-1}) \right], \\ &\quad j = 1, \dots, n,\ j \neq i , \\ \Gamma_i^i &= (\lambda g_i)^{-1} \left[f(x + \lambda g) - f(x) - \lambda \sum_{j=1, j \neq i}^{n} \Gamma_j^i g_j \right] . \end{aligned}$$

It follows from the definition that

$$f(x + \lambda g) - f(x) = \lambda \langle \Gamma^i(x, g, e, z, \lambda, \alpha), g \rangle \qquad (5)$$

for all $g \in S_1,\ e \in G,\ z \in P,\ \lambda > 0,\ \alpha > 0$.

Remark 3 One can see that the discrete gradient is defined with respect to a given direction $g \in S_1$ and in order to compute the discrete gradient $\Gamma^i(x, g, e, z, \lambda, \alpha)$ first we define a sequence of points $x^0, \dots, x^n$ and compute the values of the function f at these points, that is we compute $n + 2$ values of this function including the point x. $n - 1$ coordinates of the discrete gradient are defined similar to those of the vector $v(\alpha,\lambda)$ from (4) and ith coordinate is defined so as to satisfy the equality (5), which can be considered as some version of the mean value theorem.

Proposition 4 *Let f be a locally Lipschitz continuous function defined on $\mathbb{R}^n$ and $L > 0$ be its Lipschitz constant. Then for any $x \in \mathbb{R}^n,\ g \in S_1,\ e \in G,\ \lambda > 0,\ z \in P,\ \alpha > 0$*

$$\|\Gamma^i\| \leq C(n) L,\ C(n) = (n^2 + 2n^{3/2} - 2n^{1/2})^{1/2} .$$

For a given $\alpha > 0$ we define the following set:

$$B(x,\alpha) = \{v \in \mathbb{R}^n : \exists (g \in S_1,\ e \in G,\ z_k \in P,\ z_k \to +0,\ \lambda_k \to +0,\ k \to +\infty)\ ,$$

$$v = \lim_{k \to +\infty} \Gamma^i(x, g, e, z_k, \lambda_k, \alpha)\}\ . \tag{6}$$

Proposition 5 *Let the function f be a differentiable with respect to any direction $g \in \mathbb{R}^n$. Then for any $g \in \mathbb{R}^n$ there exists $v \in B(x,\alpha),\ \alpha > 0$ such that*

$$f'(x,g) = \langle v, g \rangle\ .$$

Proposition 6 *Let the function f be a locally Lipschitz continuous, differentiable with respect to any direction $g \in \mathbb{R}^n$ and $x \in D(f)$. Then $\nabla f(x) \in B(x,\alpha),\ \alpha > 0$.*

Proposition 7 *Assume that f is a semismooth, quasidifferentiable function and its subdifferential and superdifferential are polytopes at a point x. Then there exists $\alpha_0 > 0$ such that*

$$coB(x,\alpha) \subset \partial f(x)$$

for all $\alpha \in (0, \alpha_0]$.

Remark 4 Proposition 7 implies that discrete gradients can be applied to approximate subdifferentials of a broad class of semismooth, quasidifferentiable functions.

Remark 5 The discrete gradient contains three parameters: $\lambda > 0$, $z \in P$ and $\alpha > 0$. $z \in P$ is used to exploit semismoothness of the function f and it can be chosen sufficiently small. In general α depends on x. However if f is a semismooth quasidifferentiable function and its subdifferential and superdifferential are polytopes at any $x \in \mathbb{R}^n$ then there exist $\delta > 0$ and $\alpha_0 > 0$ such that $\alpha(x) \in (0, \alpha_0]$ for all $y \in B_\delta(x)$. The most important parameter is $\lambda > 0$. In the sequel we assume that $z \in P$ and $\alpha > 0$ are sufficiently small.

Consider the following set at a point $x \in \mathbb{R}^n$:

$$D_0(x,\lambda,z) = \mathrm{cl\,co}\Big\{v \in \mathbb{R}^n : \exists (g \in S_1, e \in G) : v = \Gamma^i(x, g, e, \lambda, z, \alpha)\Big\}\ .$$

Proposition 4 implies that the set $D_0(x,\lambda,z)$ is compact and convex for any $x \in \mathbb{R}^n$.

Corollary 4 *Let f be a quasidifferentiable semismooth function. Assume that its subdifferential and superdifferential are polytopes and that in the equality*

$$f(x+\lambda g) - f(x) = \lambda f'(x,g) + o(\lambda, g),\ g \in S_1$$

$\lambda^{-1} o(\lambda, g) \to 0$ *as* $\lambda \to +0$ *uniformly with respect to* $g \in S_1$. *Then for any $\delta > 0$ there exist $\lambda_0 > 0$ and $z_0 \in P$ such that*

$$D_0(x,\lambda,z) \subset \partial f(x) + B_\delta$$

for all $\lambda \in (0,\lambda_0)$ and $z \in (0, z_0)$.

Consider the continuous approximation $C(x,\varepsilon)$ to the subdifferential $\partial f(x)$. Then Corollary 4 implies that for any $\delta > 0$ there exist $\varepsilon_0 > 0, \lambda_0 > 0$ and $z_0 \in P$ such that

$$D_0(x,\lambda,z) \subset C(x,\varepsilon) + B_\delta$$

for all $\varepsilon \in (0,\varepsilon_0),\ \lambda \in (0,\lambda_0)$ and $z \in (0, z_0)$. Thus, discrete gradients can be used to compute subsets of continuous approximations to the subdifferential in the sense of Definition 2. Consequently they can also be used to compute subsets of uniform and strongly uniform continuous approximations.

Continuous Approximations to the Quasidifferential

In this subsection we will consider continuous approximations to the Demyanov–Rubinov quasidifferential. We consider a function of the form

$$f(x) = F(x, y_1(x), \ldots, y_m(x))\ , \tag{7}$$

where $x \in \mathbb{R}^n$, the function F is continuously differentiable in $\mathbb{R}^{n+m}$, $y_i(x),\ i \in I = \{1, \ldots, m\}$, are semismooth, regular functions and their subdifferentials are polytopes. It is easy to see that the function f is differentiable with respect to any direction and

$$f'(x,g) = \left\langle \frac{\partial F(x, y(x))}{\partial x}, g \right\rangle + \sum_{i \in I} \frac{\partial F(x, y(x))}{\partial y_i} y_i'(x,g)\ ,$$

where $y(x) = (y_1(x), \ldots, y_m(x))$, $\partial F(x, y(x))/\partial x$ is the gradient of the function F with respect to x, and $\partial F(x, y(x))/\partial y_i$ is the partial derivative of the function F with respect to $y_i,\ i \in I$.

Let

$$I_1(x) = \left\{ i \in I \colon \frac{\partial F(x, y(x))}{\partial y_i} > 0 \right\},$$

$$I_2(x) = \left\{ i \in I \colon \frac{\partial F(x, y(x))}{\partial y_i} < 0 \right\}.$$

Consider the mappings $B_i(x, \alpha)$ corresponding to the functions $y_i(x)$, $i \in I$. We introduce the following two sets:

$$\underline{Z}(x, \alpha) = \operatorname{co}\left\{ v \in \mathbb{R}^n \colon v = \frac{\partial F(x, y(x))}{\partial x} + \sum_{i \in I_1(x)} \frac{\partial F(x, y(x))}{\partial y_i} v^i, \; v^i \in B_i(x, \alpha), \, i \in I_1(x) \right\},$$

$$\overline{Z}(x, \alpha) = \operatorname{co}\left\{ w \in \mathbb{R}^n \colon w = \sum_{i \in I_2(x)} \frac{\partial F(x, y(x))}{\partial y_i} w^i, \; w^i \in B_i(x, \alpha), \, i \in I_2(x) \right\}.$$

Proposition 8 *Assume that the function F is continuously differentiable in $\mathbb{R}^{n+m}$, functions $y_i(x)$, $i \in I$, are semismooth and regular and their subdifferentials are polytopes. Then the function f is quasidifferentiable and there exists $\alpha_0 > 0$ such that*

$$\underline{Z}(x, \alpha) \subset \underline{\partial} f(x)$$

and

$$\overline{Z}(x, \alpha) \subset \overline{\partial} f(x)$$

for all $\alpha \in (0, \alpha_0)$.

Corollary 5 *Suppose we are given the function $f(x) = f_1(x) - f_2(x)$, where f_1 and f_2 are semismooth, regular functions and their subdifferentials are polytopes, and $B_1(x, \alpha)$ and $B_2(x, \alpha)$ are mappings corresponding to the functions f_1 and f_2, respectively. Then the function f is quasidifferentiable and there exists $\alpha_0 > 0$ such that*

$$B_1(x, \alpha) \subset \partial f_1(x), \; B_2(x, \alpha) \subset \partial f_2(x)$$

for all $\alpha \in (0, \alpha_0)$.

Let $D_{0i}(x, \lambda, z)$ be mappings corresponding to the functions $y_i(x)$, $i \in I$. We set

$$D_1(x, \lambda, z) = \operatorname{co}\{ v \in \mathbb{R}^n \colon v = \frac{\partial F(x, y(x))}{\partial x} + \sum_{i \in I_1(x)} \frac{\partial F(x, y(x))}{\partial y_i} v^i, \; v^i \in D_{0i}(x, z, \lambda), \, i \in I_1(x) \},$$

$$D_2(x, z, \lambda, \beta) = \operatorname{co}\{ w \in \mathbb{R}^n \colon w = \sum_{i \in I_2(x)} \frac{\partial F(x, y(x))}{\partial y_i} w^i, \; w^i \in D_{0i}(x, z, \lambda), \, i \in I_2(x) \}.$$

Note that the mappings $D_1(x, \lambda, z)$, $D_2(x, \lambda, z)$ are Hausdorff continuous with respect to x for any fixed $\lambda > 0$, $z \in P$.

It follows from Corollary 4 that the sets $D_1(x, \lambda, z)$, $D_2(x, \lambda, z)$ can be used to compute subsets of continuous approximations to the subdifferential and superdifferential of the function (7).

Conclusions

In this paper we introduced continuous approximations to the subdifferential and the quasidifferential of the nonsmooth, nonconvex functions. We proposed the algorithm for their computation. This algorithm allows one to approximate subgradients a broad class of nonsmooth functions.

References

1. Bagirov AM (2003) Continuous subdifferential approximations and their applications. J Math Sci 115(5):2567–2609
2. Bagirov AM, Rubinov AM, Sukhorukova NV, Yearwood J (2003) Supervised and unsupervised data classification via nonsmooth and global optimisation. TOP: Spanish Oper Res J 11(1):1–93
3. Bagirov AM, Yearwood J (2006) A new nonsmooth optimisation algorithm for minimum sum-of-squares clustering problems. Eur J Oper Res 170(2):578–596
4. Clarke FH (1973) Necessary Conditions for Nonsmooth Problems in Optimal Control and the Calculus of Variations, PhD Thesis. University of Washington, Seattle
5. Clarke FH (1990) Optimization and Nonsmooth Analysis. Classics Appl Math 5. SIAM, Philadelphia
6. Demyanov VF, Rubinov AM (1985) Quasidifferentiable Calculus. Optimization Software, New York

7. Demyanov VF, Rubinov AM (1995) Constructive Nonsmooth Analysis. Peter Lang, Frankfurt am Main
8. Demyanov VF, Vasilyev L (1985) Nondifferentiable Optimization. Optimization Software, New York
9. Goldstein AA (1977) Optimization of Lipschitz continuous functions. Math Programm 13:14–22
10. Hiriart-Urruty JB, Lemarechal C (1993) Convex Analysis and Minimization Algorithms, II: Advanced Theory and Bundle Methods. Springer, Berlin
11. Kiwiel KC (1985) Methods of Descent for Nondifferentiable Optimization, Lecture Notes in Math., 1133. Springer, Berlin, New York
12. Mifflin R (1977) Semismooth and semiconvex functions in constrained optimization. SIAM J Control Optim 15(6): 959–972
13. Rockafellar RT (1970) Convex Analysis. Princeton University Press, Princeton
14. Xu H, Rubinov AM, Glover B (1999) Continuous approximations to generalized Jacobians. Optim 46:221–246

Continuous Global Optimization: Applications

JÁNOS D. PINTÉR
Pintér Consulting Services, Inc.,
and Dalhousie University, Halifax, Canada

MSC2000: 90C05

Article Outline

Keywords
Nonlinear Systems and Global Optimization
The Continuous Global Optimization Model
Test Problems
Illustrative Applications
See also
References

Keywords

Nonlinear decision models; Multi-extremality; Global optimization; Scientific applications; Engineering applications; Economic applications

Nonlinear Systems and Global Optimization

Man-made systems and processes can often be modeled to reasonable accuracy by postulating the exclusive use of continuous linear functions. For instance, one may think of the simplest production and distribution models known from the OR literature. (Models with integer variables will not be discussed here, even though they can be equivalently reformulated, to fit into the present framework.) If we attempt, however, the analysis of natural — physical, chemical, biological, environmental, or even economic, financial and societal — systems and their governing processes, then nonlinear functions start to play a significant role in the quantitative description. To illustrate this point, one may think of the most prominent (basic) function forms in physics: probably polynomials, power functions, the exponential-logarithmic pair and trigonometric functions come to mind first. Clouds, water flows, rugged terrains, plants and animals — as well as many other natural objects — all possess visible nonlinearities. For sophisticated examples and general principles, one may think of discussions of nonlinear dynamics, chaos, self-organizing systems and the fractal nature of the Universe: consult, e. g., [3,5,14,19,25].

Prescriptive (control, management, optimization) models which attempt to describe and optimize the behavior of inherently nonlinear systems — as a rule — lead to nonlinear decision problems. Since nonlinear decision models frequently possess multiple local optima, the general relevance of global optimization (GO) becomes obvious. In this brief article we present a list of important and challenging GO applications. We also provide several illustrative references: these describe numerous further application areas.

The Continuous Global Optimization Model

We shall consider problems in the general form

$$\begin{cases} \min\limits_{x\in D} & f(x) \\ \text{s.t.} & D := \left\{ x: \begin{array}{c} l \le x \le u; \\ g(x) \le 0, \\ j = 1,\dots,J \end{array} \right\}. \end{cases} \tag{1}$$

In (1) we use the following notation:

- x is a vector which represents decision alternatives in $\mathbf{R}^n$;
- $f(x)$ is a continuous objective function;

- D is a nonempty set of feasible decisions, defined by
  - $g(x)$, an m-vector of continuous constraint functions; and
  - l. u, explicit (finite, componentwise) n-vector bounds.

Explicit bounds on the constraint function values can also be imposed; however, such more specialized models are directly amenable to the form (1).

First of all, note that if all functions are continuous, then – by the classical theorem of Weierstrass – the optimal solution set to (1) is nonempty. At the same time, without further structural assumptions, (1) can be a very difficult global optimization problem. In other words – unless additional information is provided – there may well exist multiple (local) solutions of various quality to (1). Naturally, in most cases one would like to find the 'very best' (global) solution to the underlying decision problem, avoiding the 'traps' offered by local optima. To attain this objective, a considerable variety of GO models and solution approaches have been suggested: consult, e. g., [12].

Test Problems

Although our primary topic is real-world GO applications, one should at least mention several standardized test problem suites, since these often originate from real-world applications. For collections of (both convex and nonconvex) nonlinear programming test problems, consult, e. g., [11,18]. See [6,7,13,22] for collections of GO test problems. On the WWW, see [1,8] and [21]; especially [21] provides numerous further links and pointers, including discussions of test and real-world problems.

Illustrative Applications

Since GO problems are literally ubiquitous in scientific, engineering and economic decision making, we shall only list a number of illustrative applications. All application areas will be listed simply in alphabetical order, by information source. (The reader will notice overlaps among the problems studied in different works.)

The test problem collection [6] presents application models from the following fields:

- chemical reactor networks;
- distillation column sequencing;
- heat exchanger network synthesis;
- indefinite quadratic programming;
- mechanical design;
- general nonlinear programming;
- phase and chemical reaction equilibrium;
- pooling and blending;
- quadratically constrained problems;
- reactor-separator-recycling systems;
- VLSI design.

The volume [7] significantly expands upon the above material, adding more specific classes of nonlinear programming models, combinatorial optimization problems, and dynamic models, as well as further practical examples (see later on).

The MINPACK-2 collection presented at [1] includes models related to the following types of problems:

- brain activity;
- Chebychev quadrature;
- chemical and phase equilibria;
- coating thickness standardization;
- combustion of propane;
- control systems (analysis and design);
- database optimization;
- design with composites;
- elastic-plastic torsion;
- enzyme reaction analysis;
- exponential data fitting;
- flow in a channel;
- flow in a driven cavity;
- Gaussian data fitting;
- Ginzburg–Landau problem;
- human heart dipole;
- hydrodynamic modeling;
- incompressible elastic rods;
- isomerization of alpha-pinene;
- Lennard–Jones clusters;
- minimal surfaces;
- pressure distribution;
- Ramsey graphs;
- solid fuel ignition;
- steady-state combustion;
- swirling flow between disks;
- thermistor resistance analysis;
- thin film design;
- VLSI design.

A detailed discussion of several GO case studies and applications is presented in [22]. These problems were

analyzed by using LGO, an integrated model development environment to formulate and solve GO problems; consult also [23]. The current list of LGO applications includes, for instance, the following areas:

- bio-mechanical design;
- 'black box' (closed, confidential, etc.) system design and operation;
- combination of negotiated expert opinions (forecasts, votes, assessments, etc.);
- data classification, pattern recognition;
- dynamic population and resource management;
- extremal energy (point arrangement) problems, free and surface-constrained forms;
- inverse model fitting to observation data (calibration);
- multifacility location-allocation problems;
- nonlinear approximation, nonlinear regression, and other curve/surface fitting problems;
- optimized tuning of equipment and instruments in medical research and other areas;
- reactor maintenance policy analysis;
- resource allocation (in cutting, loading, scheduling, sequencing, etc. problems);
- risk analysis and control in various environmental management contexts;
- robotics design issues;
- robust product/mixture design;
- satisfiability problems;
- statistical modeling;
- systems of nonlinear equations and inequalities;
- therapy (dosage and schedule) optimization.

The WWW site [21] discusses, inter alia, the following application areas:

- bases for finite elements;
- boundary value problems;
- chemical engineering problems;
- chemical phase equilibria;
- complete pivoting example;
- distance geometry models;
- extreme forms;
- identification of dynamical systems with matrices depending linearly on parameters;
- indefinite quadratic programming models;
- minimax problems;
- nonlinear circuits;
- optimal control problems;
- optimal design;
- parameter identification with data of bounded error;
- PDE defect bounds;
- PDE solution by least squares;
- pole assignment;
- production planning;
- propagation of discrete dynamical systems;
- protein-folding problem;
- pseudospectrum;
- quadrature formulas;
- Runge–Kutta formulas;
- spherical designs (point configurations);
- stability of parameter matrices.

The collection of test problems [7] includes models from the following application areas:

- batch plant design under uncertainty;
- chemical reactor network synthesis;
- conformational problems in clusters of atoms and molecules;
- dynamic optimization problems in parameter estimation;
- homogeneous azeotropic separation system;
- network synthesis;
- optimal control problems;
- parameter estimation and data reconcilliation;
- phase and chemical reaction equilibrium;
- pooling/blending operations;
- pump network synthesis;
- robust stability analysis;
- trim loss minimization.

The article [4] reviews several significant applications of rigorous global optimization (based on the interval branch and bound approach). These applications include:

- currency trading;
- finite element analysis (in a high-tech engineering design context);
- gene prediction in genome therapeutics;
- magnetic resonance imaging (in a medical application);
- numerical mathematics (search for an approximate greatest common divisor of given polynomials);
- parameter estimation in signal processing;
- portfolio management;

One can immediately add here the application of interval techniques to an issue of paramount significance in numerical modeling:

- solving systems of nonlinear (and linear) equations; consult, e. g., [20]

The volume [24] also covers a broad range of applications from the following areas:

- agro-ecosystem management;
- analysis of nucleid acid sequences;
- assembly line design;
- cellular mobile network design;
- chemical process optimization;
- chemical product design;
- computational modeling of atomic and molecular structures;
- controller design for motors;
- electrical engineering design;
- feeding strategies in animal husbandry;
- financial modeling;
- laser equipment design;
- mechanical engineering design;
- radiotherapy equipment calibration;
- robotics design;
- satellite data analysis (interferometry problem);
- virus structure reconstruction;
- water resource distribution systems.

As the above lists illustrate, the application potentials of global optimization are indeed most diverse.

For additional literature on real-world applications, see, e. g., the following references:

- network problems, combinatorial optimization (knapsack, traveling salesman, flow-shop problems), batch process scheduling: [17];
- GO algorithms and their applications (primarily) in chemical engineering design: [9];
- contributions on decision support systems and techniques for solving GO problems, but also on molecular structures, queueing systems, image reconstruction, location analysis and process network synthesis: [2];
- multilevel optimization algorithms and their applications: [15];
- engineering applications of the finite element method: [16];
- a variety of applications, e. g., from the fields of environmental management, geometric design, robust product design, and parameter estimation: [10].

Numerous issues of the *Journal of Global Optimization* – as well as a large number of other professional OR/MS, natural science and engineering journals – also publish articles describing interesting GO applications.

See also

- ▶ αBB Algorithm
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Differential Equations and Global Optimization
- ▶ DIRECT Global Optimization Algorithm
- ▶ Forecasting
- ▶ Global Optimization in the Analysis and Management of Environmental Systems
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Binary Star Astronomy
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization Using Space Filling
- ▶ Interval Global Optimization
- ▶ Mixed Integer Nonlinear Programming
- ▶ Topology of Global Optimization

References

1. Argonne National Lab (1993) MINPACK-2 test problem collection. Argonne National Lab., Argonne, IL, see also the accompanying notes Large-scale optimization: Model problems. BM Averick and Moré JJ www-c.mcs.anl/gov/gov/home/more/tprobs/html
2. Bomze IM, Csendes T, Horst R, Pardalos PM (eds) (1997) Developments in global optimization. Kluwer, Dordrecht
3. Casti JL (1990) Searching for certainty. Morrow, New York
4. Corliss GF, Kearfott RB (1999) Rigorous global search: Industrial applications. In: Csendes T (ed) Developments in Reliable Computing. Kluwer, Dordrecht, pp 1–16
5. Eigen M, Winkler R (1975) Das Spiel. Piper, Munich
6. Floudas CA, Pardalos PM (1990) A collection of test problems for constrained global optimization algorithms. No. 455 of Lecture Notes Economics and Math Systems. Springer, Berlin
7. Floudas CA, Pardalos PM, Adjiman C, Esposito WR, Gumus ZH, Harding ST, Klepeis JL, Meyer CA, Schweiger CA (1999) Handbook of test problems in local and global optimization. Kluwer, Dordrecht
8. Gray P, Hart WE, Painton L, Phillips C, Trahan M, Wagner J (1999) A survey of global optimization methods. Sandia National Lab, Albuquerque, NM http://www.cs.sandia.gov/opt/survey/main.html
9. Grossmann IE (ed) (1996) Global optimization in engineering design. Kluwer, Dordrecht
10. Hendrix EMT (1998) Global optimization at work. PhD Thesis LU Wageningen, The Netherlands

11. Hock W, Schittkowski K (eds) (1987) Test examples for nonlinear programming codes. Lecture Notes Economics and Math Systems. Springer, Berlin
12. Horst R, Pardalos PM (eds) (1995) Handbook of global optimization. Kluwer, Dordrecht
13. Jansson C, Knöppel O (1992) A global minimization method: The multi-dimensional case. Res Report TUHH, Hamburg-Harburg, Germany
14. Mandelbrot BB (1983) The fractal geometry of nature. Freeman, New York
15. Migdalas A, Pardalos PM, Värbrand P (eds) (1997) Multilevel optimization: Algorithms and applications. Kluwer, Dordrecht
16. Mistakidis E, Stavroulakis GE (1997) Nonconvex optimization. Algorithms, heuristics and engineering applications of the F.E.M. Kluwer, Dordrecht
17. Mockus J, Eddy W, Mockus A, Mockus L, Reklaitis G (1996) Bayesian heuristic approach to discrete and global optimization. Kluwer, Dordrecht
18. Moré JJ, Garbow BS, Hillström KE (1981) Testing unconstrained optimization software. ACM Trans Math Softw 7:17–41
19. Murray JD (1983) Mathematical biology. Springer, Berlin
20. Neumaier A (1990) Interval methods for systems of equations. Cambridge Univ. Press, Cambridge
21. Neumaier A (1999) Global optimization. http://solon.cma.univie.ac.at/~neum/glopt.html
22. Pinter JD (1996) Global optimization in action. Kluwer, Dordrecht
23. Pintér JD (1999) LGO-A model development system for continuous global optimization. User's guide. Pintér Consulting Services, Halifax, NS, Halifax, NS
24. Pintér JD (ed) (2001) Global optimization – Selected case studies. Kluwer, Dordrecht
25. Schroeder M (1991) Fractals, chaos, power laws. Freeman, New York

Continuous Global Optimization: Models, Algorithms and Software

JÁNOS D. PINTÉR
Pintér Consulting Services, Inc.,
and Dalhousie University, Halifax, Canada

MSC2000: 90C05

Article Outline

Keywords
The Continuous Global Optimization Model
Model Types
Exact Methods
Naive Approaches
Complete (Enumerative) Search Strategies
Homotopy (Parameter Continuation), Trajectory Methods, and Related Approaches
Successive Approximation (Relaxation) Methods
Branch and Bound Algorithms
Bayesian Search (Partition) Algorithms
Adaptive Stochastic Search Algorithms
Heuristic Strategies
'Globalized' Extensions of Local Search Methods
Genetic Algorithms, Evolution Strategies
Simulated Annealing
Tabu Search
Approximate Convex Global Underestimation
Continuation Methods
Sequential Improvement of Local Optima
Global Optimization Software
Software Evaluation
Software Applicability Range (Solvable Model Types)
GO Methodology Applied
Hardware and Software Requirements
Test Results
Additional Software Information
See also
References

Keywords

Nonlinear decision models; Continuous global optimization; Model types; Solution strategies; Software development and evaluation

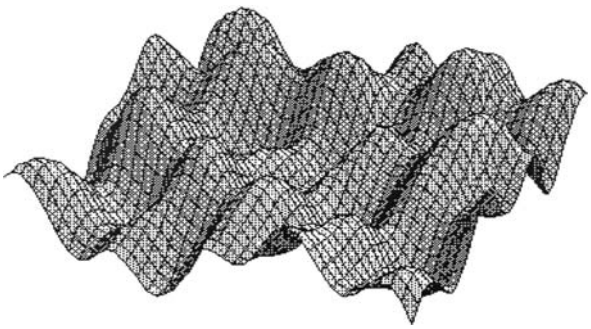
The Continuous Global Optimization Model

We shall consider the continuous global optimization problem (GOP) in the general form

$$\begin{cases} \min\limits_{x\in D} & f(x) \\ \text{s.t.} & D := \left\{ x: \begin{array}{c} l \le x \le u; \\ g(x) \le 0, \\ j = 1,\dots,J \end{array} \right\} . \end{cases} \tag{1}$$

In (1) the following assumptions are used:

- x is a vector representing decision alternatives in $\mathbf{R}^n$;
- D is a nonempty set of feasible decisions, defined by
  - l, u: explicit (finite, componentwise) n-vector bounds of x, and
  - $g(x)$ is an m-vector of continuous constraint functions defined on $[l, u]$;



Continuous Global Optimization: Models, Algorithms and Software, Figure 1
A two-variable multi- extremal function

- $f(x)$ is a continuous objective function defined on D.

Explicit bounds on the constraint function values can also be imposed; however, such more specialized models are also directly amenable to the form (1).

Since the functions f and g are all continuous in D, the GOP (1) evidently has a nonempty globally optimal solution set X^*. At the same time, one can immediately realize that — in its full generality — instances of model (1) can pose a very significant numerical challenge. Since the usual convexity assumptions are absent, D may be disconnected and/or nonconvex, and the objective function f may also be multi-extremal. That is, the number of local (pseudo) solutions to (1) is typically unknown and it can be large; the quality of the various local and global solutions may differ significantly. To illustrate this point, see Fig. 1, which depicts a 'hilly landscape' (in fact, the surface plot of a relatively simple composition of trigonometric functions with embedded polynomial arguments, in just two variables). For instance, this function could be the objective in (1) defined on the corresponding interval feasible region $[l, u]$.

To solve the GOP (1) – in a strict mathematical sense – means to find the complete set of globally optimal solutions X^*, and the associated global optimum value $f^* = f(x^*)$, $x^* \in X^*$. In most cases, at least in the realm of continuous GO, we need to replace this 'ambitious' objective by finding a verified estimate – upper and lower bounds – of f^*, and corresponding approximation(s) of points from the set X^*. Naturally, such estimates are to be determined on the basis of a finite number of algorithmically generated sample points from D, or from the embedding interval $[l, u]$.

For reasons of better analytical and numerical tractability, usually the following additional assumptions are made:

- D is a full-dimensional subset (a 'body') in $\mathbf{R}^n$;
- X^* is at most countable;
- g (i. e., each of its component functions) and f are Lipschitz-continuous on $[l, u]$.

Observe that the first assumption makes algorithmic search possible within the set D. With respect to the second assumption, note that – in most practical contexts – the set of global optimizers consists only of a single point, or of several points. Finally, the Lipschitz assumption – i. e., that changes in function values are uniformly controlled by changes in their argument – is a sufficient condition for estimating f^* on the basis of a finite set of search points. We emphasize that the factual knowledge of the smallest suitable Lipschitz constant is not required – and in practice it is typically unknown indeed. The Lipschitz criterion is evidently met, e. g., by all continuously differentiable functions defined on $[l, u]$; however, their class is even broader.

Due to the very general model structure postulated above, classical (convexity-based) numerical approaches are, generally speaking, not directly applicable to solve GOPs: instead, truly global scope methodology is needed. In the past decades, a considerable variety of GO models and solution approaches have been proposed and analyzed. Below we shall provide a concise review, with a view towards software development. For detailed discussions, consult, e. g., the illustrative list of references.

Model Types

The most important GO model classes that have been extensively studied include the following. (Note that postulated properties of g – such as e. g., convexity – are required componentwise.)

- Bilinear and biconvex programming (f is bilinear or biconvex, D is convex).
- Combinatorial optimization (problems that have discrete decision variables in f and/or in g can be equivalently reformulated as GO problems in continuous variables).

- Concave minimization (f is concave, D is convex).
- Continuous global optimization (f and g are arbitrary continuous functions).
- Differential convex (DC) optimization (f and the components in g can all be explicitly represented, as the difference of two corresponding convex functions).
- Fractional programming (f is the ratio of two real functions, and g is convex).
- Linear and nonlinear complementarity problems (f is the scalar product of two vector functions, D is typically assumed to be convex).
- Lipschitz optimization (f and g are arbitrary Lipschitz-continuous functions).
- Minimax problems (f is some minimax objective, the maximum is considered over a discrete set or a convex set, D is convex).
- Multilevel optimization (e. g., models of noncooperative games, involving hierarchies of decision makers, the conflicting criteria are aggregated by f; D is usually assumed to be convex).
- Multi-objective programming (e. g., determination of the efficient set, when several conflicting objectives are to be optimized over the region D).
- Multiplicative programming (f is the product of several convex functions, and g is convex, or – more generally – also multiplicative).
- Network problems (f can be taken from several nonconvex function classes, and g is typically linear or convex).
- Parametric nonconvex programming (in these the feasible region D and/or the objective f may also depend also on a parameter vector).
- Quadratic optimization (f is an arbitrary – indefinite – quadratic function; g is linear or, in the more general case, is also made up by arbitrary quadratic functions).
- Reverse convex programming (at least one of the functions in g expresses a reverse convex constraint).
- Separable global optimization (f is an arbitrary nonlinear – in general, nonconvex – separable function, D is typically convex).
- Stochastic (nonconvex) models in which the functions f, g depend on random factors.
- Various other nonlinear programming problems, in absence of a verified convex structure: this broad category includes, e. g., models in which some of the functions f, g are defined by complex 'black box' computational procedures.

Note that the problem classes listed are not necessarily distinct; in fact, several of them are hierarchically contained in the more general problem types listed. For detailed descriptions of most of these model types and their connections consult, e. g., [13], with numerous further references.

Observe also that in the list presented, there are specifically structured models (such as e. g., a concave minimization problem under linear or convex constraints), as well as far more general ones (such as e. g., differential convex, Lipschitz or continuous problems). Hence, one can reasonably expect that the most suitably tailored solution approaches will also vary to a considerable extent. Very general search strategies should work for most models – albeit their efficiency might be low for specialized problems. At the same time, strictly specialized solvers may not work at all for problem classes outside of their scope.

Several of the most important GO strategies are listed below, together with additional remarks and references. Again, the items of the list are not necessarily exclusive. Most GO software implementations are based upon one of these approaches, possibly combining ideas from several strategies.

Exact Methods

Naive Approaches

These include the most well known passive (simultaneous) or direct (not fully adaptive) sequential GO strategies: uniform grid, space covering, and pure random searches. Note that such methods are obviously convergent under mild assumptions, but are – as a rule – impracticable in higher-dimensional problems. Consult corresponding chapters in [13,24,30].

Complete (Enumerative) Search Strategies

These are based upon an exhaustive (and typically streamlined) enumeration of all possible solutions. Applicable to combinatorial problems, as well as to certain 'well-structured' continuous GO problems such as, e. g., concave programming. See, e. g., [14].

Homotopy (Parameter Continuation), Trajectory Methods, and Related Approaches

These methods have the 'ambitious' objective of visiting all stationary points of the objective function: this, in turn, leads to the list of all – global as well as local – optima. This general approach includes differential equation model based, path following search strategies, as well as fixed-point methods and pivoting algorithms. See, for instance, [5] and [8].

Successive Approximation (Relaxation) Methods

The initial optimization problem is replaced by a sequence of relaxed subproblems that are easier to solve. Successive refinement of subproblems to approximate the initial problem; cutting planes and more general cuts, diverse minorant function constructions, nested optimization and decomposition strategies are also possible. Applicable to structured GO problems such as, e. g., concave minimization and DC problems [14].

Branch and Bound Algorithms

A variety of partition strategies have been proposed to solve GOPs. These are based upon adaptive partition, sampling, and subsequent lower and upper bounding procedures: these operations are applied iteratively to the collection of active (remaining 'candidate') subsets within the feasible set D. Their exhaustive search feature is similar in spirit to analogous integer programming methodology. Branch and bound subsumes many specific approaches, and allows for a range of implementations.

Branch and bound methods typically rely on some a priori structural knowledge about the problem. This information may relate, for instance to how rapidly each function can vary (e. g. the knowledge of a suitable 'overall' Lipschitz constant, for each function f and g); or to the availability of an analytic formulation – and guaranteed smoothness – of all functions (for instance, in interval arithmetic based methods).

The branch and bound methodology is applicable to broad classes of GO problems: e. g., in combinatorial optimization, concave minimization, reverse convex programs, DC programming, and Lipschitz optimization. For details, consult [12,14,15,20,24,26].

Bayesian Search (Partition) Algorithms

These methods are based upon some postulated statistical information, to enable a prior stochastic description of the function class modeled. During optimization, the problem instance characteristics are adaptively estimated and updated. Note that, typically only the corresponding one-dimensional model development is exact; furthermore, that in most practical cases 'myopic' approximate decisions govern the search procedure.

This general approach is applicable also to (merely) continuous GO problems. Theoretically, convergence to the optimal solution set is guaranteed only by generating an everywhere dense set of search points. One of the obvious challenges of using statistical methods is the choice and verification of an 'appropriate' statistical model, for the class of problems to which they are applied. Additionally, it seems to be difficult to implement rigorous and computationally efficient versions of these algorithms for higher-dimensional optimization problems. Note, however, that if one 'skips' the underlying Bayesian paradigm, then these methods can also be pragmatically viewed as adaptive partition algorithms, and – as such – they can be directly extended to higher dimensions: see [24]. For detailed expositions on Bayesian approaches, consult, e. g., [18,19,27].

Adaptive Stochastic Search Algorithms

This is another broad class of methods, based upon random sampling in the feasible set. In its basic form, it includes various random search strategies that are convergent, with probability one. Search strategy adjustments, clustering and deterministic solution refinement options, statistical stopping rules, etc. can also be added as enhancements.

The methodology is applicable to both discrete and continuous GO problems under very mild conditions. Consult, for instance, [2,24,30].

Heuristic Strategies

'Globalized' Extensions of Local Search Methods

These are partially heuristic algorithms, yet often successful in practice. The essential idea is to apply a preliminary grid search or random search based global phase, followed by applying a local (convex program-

ming) method. For instance, random multistart performs a local search from several points selected randomly from the search domain D. Note that even such sampling is not trivial, when D has a complicated shape, as being defined, e. g., by (merely) continuous nonlinear functions.

Frequently, sophisticated algorithm enhancements are added to this basic strategy. For instance, the clustering of sample points is aimed at selecting only a single point from each sampled 'basin' of f from which then a local search method is initiated. For more details, consult, for instance, [27].

Genetic Algorithms, Evolution Strategies

These methods 'mimic' biological evolution: namely, the process of natural selection and the 'survival of the fittest' principle. An adaptive search procedure based on a 'population' of candidate solution points is used. Iterations involve a competitive selection that drops the poorer solutions. The remaining pool of candidates with higher 'fitness value' are then 'recombined' with other solutions by swapping components with another; they can also be 'mutated' by making some smaller-scale change to a candidate. The recombination and mutation moves are applied sequentially; their aim is to generate new solutions that are biased towards subsets of D in which good – although not necessarily globally optimized – solutions have already been found.

Numerous variants of this general strategy, based on diverse evolution 'game rules', can be constructed. The different types of evolutionary search methods include approaches that are aimed at continuous GOPs, and also others that are targeted towards solving combinatorial problems. The latter group is often called genetic algorithms. For details, consult, e. g., [10,17,22,29].

Simulated Annealing

These techniques are based upon the physical analogy of cooling crystal structures that spontaneously attempt to arrive at some stable (globally or locally minimal potential energy) equilibrium. This general principle is applicable to both discrete and continuous GO problems under mild structural requirements: consult, e. g., [1,22,28].

Tabu Search

In this general category of metaheuristics, the essential idea during search is to 'forbid' moves to points already visited in the (usually discrete) search neighborhood, at least for a number of upcoming steps. This way, one can temporarily accept new inferior solutions, in order to avoid (sub)paths already investigated. This approach can lead to exploring new regions of D, with the goal of finding a solution by 'globalized' search.

Tabu search has traditionally been applied to combinatorial optimization (e. g., scheduling, routing, traveling salesman) problems. The technique can be made – at least, in principle – directly applicable to continuous GOPs by a discrete approximation (encoding) of the problem, but other extensions are also possible. See [9,22,29].

Approximate Convex Global Underestimation

This heuristically attractive strategy attempts to estimate the (postulated) large scale, 'overall' convexity characteristics of the objective function f based on directed sampling in D. Applicable to smooth problems. See, e. g., [6].

Continuation Methods

These first transform the potential function into a more smooth ('simpler') function which has fewer local minimizers, and then attempt to trace the minimizers back to the original function. Again, this methodology is applicable to smooth problems. For theoretical background, see, for instance, [8].

Sequential Improvement of Local Optima

These methods usually operate on adaptively constructed auxiliary functions, to assist the search for gradually better optima. The general heuristic principle is realized by so-called tunneling, deflation, and filled function approaches; consult, for example, [16].

Global Optimization Software

In spite of significant theoretical advances in GO, software development and 'standardized' use lag behind. This can be expected due to the potential numerical difficulty of GOPs; recall Fig. 1. Even 'much simpler' problem instances – such as e. g., concave minimization, or

indefinite quadratic programming – belong to the hardest (*NP*) class of mathematical programming problems.

As summarized above, there exist several broad classes of algorithmic GO approaches that possess strong theoretical convergence properties, and – at least in principle – are straightforward to implement. However, all such rigorous approaches involve a computational demand that increases exponentially as a function of problem size, even in case of the simplest GO problem instances. (Consult, for example, [13] for related discussions.) Therefore many practical GO strategies are completed by a 'traditional' local optimization phase. Global convergence, however, needs to be guaranteed by the global scope algorithm component: the latter – at least in theory – should be used in a complete, 'exhaustive' fashion. The above remarks indicate the basic inherent theoretical (and practical) difficulty of developing robust, yet efficient GO software.

Since the computational demand of rigorous strategies can be expected to be some exponential function of the problem dimensionality, GO problems in $\mathbf{R}^n$ (n being just 5, 10, 20, 50, 100, ...) may have rapidly increasing – possibly straight enormous – numerical complexity. This is (and will remain) true, in spite of the fact that computational power seems to grow at an unbelievable pace: the so-called 'curse of dimensionality' is here to stay.

In 1996, a survey on continuous GO software was prepared for the newsletter of the Mathematical Programming Society [23]. Additional information has been collected from the Internet, from several GO books, and from the *Journal of Global Optimization*. Drawing on the responses of software developers and the additional information available, over 50 software products were annotated in that review. (In order to assist in obtaining further information, contact person(s), their e-mail addresses, ftp and/or WWW sites have also been listed.)

Most probably, by now the number of solvers aimed at GOPs is around one hundred (or even more). The general impression is, however, that many of these software products are still at an experimental development stage, and of dominantly 'academic' character, as opposed to 'industrial strength' tools. (Of course, it is not impossible that proprietary software products used by industry and private companies are not announced publicly.)

Below we shall list some key aspects that should be addressed by professional quality GO software development:

- well-specified hardware and software environments (supported development platforms);
- quality user guidance: clearly outlined model development procedure, sensible modeling and troubleshooting tips, user file templates, and (also) nontrivial numerical examples;
- fully functional, 'friendly' user interface;
- 'fool-proof' solver selection and execution procedures;
- good runtime communication and documentation: clear system output for all foreseeable program execution versions and situations, including proper error messages, and result file(s);
- visualization features which are especially desirable in nonlinear systems modeling, to avoid problem misrepresentation, and to assist in finding alternative models and solution procedures;
- reliable, high-quality user support;
- continuous product maintenance and development (since not only science progresses, but hardware devices, operating systems, as well as development platforms are in permanent change).

This tentative 'wish-list' of requirements indicates that although the task is not impossible, it is a challenge. As for an example, we refer to LGO – an integrated model development and solver system – that has been developed with a view towards the desiderata listed above. Details regarding LGO are described, e. g., in [24,25].

Software Evaluation

In order to obtain information regarding the scope and usability of GO software, it needs to be thoroughly tested. This is a demanding task, when done properly. Consideration needs to be given to the selection of appropriate – nontrivial and practically meaningful – examples. Computational experiments should be carefully designed; and the results should be reported in sufficient details, to assure a fair and accurate assessment. For corresponding discussions and GO (or other) test problems, consult, e. g., [3,4,7,21].

A GO software evaluation framework can be proposed, for instance, along the following guidelines.

Software Applicability Range (Solvable Model Types)

- objective function: concave, DC, Lipschitz, continuous, or some other (general or more special) function form;
- constraints: unconstrained problems, bound constrained problems, linear constraints, general nonlinear smooth constraints;
- additional information related to solvable model types and sizes, with corresponding expected runtimes (within given hardware and software environments).

GO Methodology Applied

- summary (or more detailed) description of basic principles;
- adequate list of references;

Hardware and Software Requirements

- supported hardware platforms;
- minimal hardware configuration needed;
- operating systems;
- programming languages and environments;
- compiler(s) needed;
- connectivity to other development environments;
- portability to other hardware and software platforms.

Test Results

- test problem description, mathematical and/or coded form;
- real world background information (when applicable);
- best known results, with references;
- accuracy requirements, stopping criteria;
- hardware and software environment used in testing;
- standard timing (to facilitate comparisons among different platforms);
- time and computational demand, in order to find the (estimated) global optimum;
- comparative success rate;
- information regarding the reproducibility of results.

Additional Software Information

- installation procedure;
- user interface features;
- academic and/or professional licenses; conditions of use;
- user support (manual, on-line help, example files, input and result handling, etc.);
- other points of interest.

Of course – at least from a practical point of view – the most meaningful test is to apply GO methods to problems that are of interest in the real world. For numerous existing and prospective GO applications, please consult the related articles ► Global Optimization in the Analysis and Management of Environmental Systems and ► Continuous Global Optimization: Applications.

See also

► αBB Algorithm
► Convex Envelopes in Optimization Problems
► Differential Equations and Global Optimization
► DIRECT Global Optimization Algorithm
► Global Optimization Based on Statistical Models
► Global Optimization in Batch Design Under Uncertainty
► Global Optimization in Binary Star Astronomy
► Global Optimization in Generalized Geometric Programming
► Global Optimization of Heat Exchanger Networks
► Global Optimization Methods for Systems of Nonlinear Equations
► Global Optimization in Phase and Chemical Reaction Equilibrium
► Global Optimization Using Space Filling
► Interval Global Optimization
► Large Scale Unconstrained Optimization
► MINLP: Branch and Bound Global Optimization Algorithm
► MINLP: Heat Exchanger Network Synthesis
► MINLP: Mass and Heat Exchanger Networks
► Mixed Integer Linear Programming: Heat Exchanger Network Synthesis
► Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
► Modeling Languages in Optimization: A New Paradigm
► Optimization Software
► Smooth Nonlinear Nonconvex Optimization
► Topology of Global Optimization

References

1. Aarts E, Lenstra JK (eds) (1997) Local search in combinatorial optimization. Wiley, New York
2. Boender CGE, Romeijn E (1995) Stochastic methods. In: Horst R, Pardalos PM (eds) Handbook of Global of Optimization. Kluwer, Dordrecht, pp 829–869
3. Bomze IM, Csendes T, Horst R, Pardalos PM (eds) (1997) Developments in global optimization. Kluwer, Dordrecht
4. De Leone R, Murli A, Pardalos PM, Toraldo G (eds) (1998) High performance software for nonlinear optimization: Status and perspectives. Kluwer, Dordrecht
5. Diener I (1995) Trajectory methods in global optimization. In: Horst R, Pardalos PM (eds) Handbook of Global Optimization. Kluwer, Dordrecht, pp 649–668
6. Dill KA, Phillips AT, Rosen JB (1997) Molecular structure prediction by global optimization. In: Bomze IM, Csendes T, Horst R, Pardalos PM (eds) Developments in Global Optimization. Kluwer, Dordrecht, pp 217–234
7. Floudas CA, Pardalos PM, Adjiman C, Esposito WR, Gumus ZH, Harding ST, Klepeis JL, Meyer CA, Schweiger CA (1999) Handbook of test problems in local and global optimization. Kluwer, Dordrecht
8. Forster W (1995) Homotopy methods. In: Horst R, Pardalos PM (eds) Handbook of Global Optimization. Kluwer, Dordrecht, pp 669–750
9. Glover F, Laguna M (1997) Tabu search. Kluwer, Dordrecht
10. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading
11. Gray P, Hart WE, Painton L, Phillips C, Trahan M, Wagner J (1999) A survey of global optimization methods. Sandia National Laboratories, http://www.cs.sandia.gov/opt/survey/main.html
12. Hansen ER (1992) Global optimization using interval analysis. M. Dekker, New York
13. Horst R, Pardalos PM (eds) (1995) Handbook of global optimization. Kluwer, Dordrecht
14. Horst R, Tuy H (1996) Global optimization–deterministic approaches, 3rd edn. Springer, Berlin
15. Kearfott RB (1996) Rigorous global search: Continuous problems. Kluwer, Dordrecht
16. Levy AV, Gomez S (1984) The tunneling method applied to global optimization. Numerical Optimization. In: Boggs PT (ed). SIAM, Philadelphia, pp 213–244
17. Michalewicz Z (1996) Genetic algorithms + data structures = evolution programs, 3rd edn. Springer, Berlin
18. Mockus J (1989) Bayesian approach to global optimization. Kluwer, Dordrecht
19. Mockus J, Eddy W, Mockus A, Mockus L, Reklaitis G (1996) Bayesian heuristic approach to discrete and global optimization. Kluwer, Dordrecht
20. Neumaier A (1990) Interval methods for systems of equations. Cambridge Univ. Press, Cambridge
21. Neumaier A (1999) Global optimization. http://solon.cma.univie.ac.at/~neum/glopt.html
22. Osman IH, Kelly JP (eds) (1996) Meta-heuristics: Theory and applications. Kluwer, Dordrecht
23. Pintér JD (1996) Continuous global optimization software: A brief review. Optima 52:1–8, http://plato.la.asu.edu/gom.html
24. Pintér JD (1996) Global optimization in action. Kluwer, Dordrecht
25. Pintér JD (1998) A model development system for global optimization. In: De Leone R, Murli A, Pardalos PM, Toraldo G (eds) High Performance Software for Nonlinear Optimization: Status and Perspectives. Kluwer, Dordrecht, pp 301–314
26. Ratschek H, Rokne JG (1995) Interval methods. In: Horst R, Pardalos PM (eds) Handbook of Global Optimization. Kluwer, Dordrecht, pp 751–828
27. Törn AA, Žilinskas A (1989) Global optimization. In: Lecture Notes Computer Sci, vol 350. Springer, Berlin
28. Van Laarhoven PJM, Aarts EHL (1987) Simulated annealing: Theory and applications. Kluwer, Dordrecht
29. Voss S, Martello S, Osman IH, Roucairol C (eds) (1999) Meta-heuristics: Advances and trends in local search paradigms for optimization. Kluwer, Dordrecht
30. Zhigljavsky AA (1991) Theory of global random search. Kluwer, Dordrecht

Continuous Reformulations of Discrete-Continuous Optimization Problems

Oliver Stein
School of Economics and Business Engineering, University of Karlsruhe, Karlsruhe, Germany

MSC2000: 90C11, 90C10, 90C33, 90C27

Article Outline

Introduction
Definitions
Formulations
  Representing the Discrete Decisions by Approximate Continuous Variables
  Representing the Discrete Decisions by Exact Continuous Variables
  Modeling Propositional Logic Constraints with Exact Continuous Variables
  The Case of Inconsistent Equalities
Conclusions
See also
References

Introduction

Nonlinear optimization problems involving discrete decision variables, also known as generalized disjunctive programming (GDP) or mixed-integer nonlinear programming (MINLP) problems, arise frequently in applications. Examples from process engineering include the synthesis of heat exchanger or reactor networks, the optimization of separation processes, such as sequencing and tray optimization problems of distillation columns, and the optimization of entire process flowsheets [3]. The discrete decisions in these problems are usually related to the structure of the process whereas typical continuous variables are process states such as temperatures, concentrations or flows.

Connections between continuous and discrete optimization problems have been studied for several decades (see, e. g., [4]). In particular, in [9] it was observed that discrete variables can be modeled by complementarity constraints, that is, the discrete model is replaced by a continuous model. A broad survey on other approaches to model discrete decisions by continuous formulations is given in [10], including concave optimization problems and relaxation by semi-definite programming, with applications to the maximum clique problem, satisfiability, the Steiner tree problem, and minimax problems.

Extensive work has been addressed to discrete-continuous problems with linear objective function and constraints, known as mixed-integer linear programming (MILP) problems. In fact, a number of powerful algorithms have been developed which are ready to solve practically relevant, large-scale problems of this type. As soon as the objective function and the constraints comprise nonlinear terms in the continuous variables, as it is usually the case, for example, for problems in process engineering, the optimization problem is referred to as a mixed-integer nonlinear programming (MINLP) problem. Algorithms for MINLP problems are either based on branch and bound with nonlinear programming (NLP) subproblems or on decomposition methods that alternately solve NLP and MILP subproblems. These algorithms are guaranteed to locate the global optimum if the nonlinearities are *convex*.

Optimization problems involving *nonconvex* objective function and constraints are by far more difficult to solve. In [11] it is proposed to reformulate discrete-continuous optimization problems by the idea from [9], as purely continuous optimization problems with complementarity constraints. In this approach, the discrete variable set of an MINLP problem is replaced by continuous variables which are restricted to take discrete values by enforcing a special type of either non-differentiable or degenerate continuous constraints.

[15] complements this approach by purely continuous reformulations of MINLP problems with better theoretical properties, as will be explained below. As all continuous reformulation approaches inevitably lead to nonconvex optimization problems, searching for a global solution may be numerically challenging. On the other hand, these continuous reformulations yield efficient ways to locally solve MINLP problems on the basis of NLP solution methods.

Definitions

Consider a generalized disjunctive representation [12] of nonlinear optimization problems, where an objective function is minimized subject to two different types of constraints, namely global constraints that hold irrespectively of any discrete decision, and constraints contained in disjunctions that are only enforced if a corresponding Boolean variable $Y_{i,k}$ is True. The optimization problem is then formulated as follows:

$$(GDP)\ \min_{x,Y} \Phi(x) + \sum_{k\in K} b_k$$

$$\text{s.t.} \quad f(x) = 0\,, \tag{1}$$

$$g(x) \le 0\,, \tag{2}$$

$$\bigvee_{i\in D_k} \begin{bmatrix} Y_{i,k} \\ h_{i,k}(x) = 0\,, \\ r_{i,k}(x) \le 0\,, \\ b_k = \gamma_{i,k}\,, \end{bmatrix}, k \in K\,, \tag{3}$$

$$D_k = \{1, 2, \ldots, n_k\}\,,$$

$$\Omega(Y) = \text{ True}, Y_{i,k} \in \{\text{True}, \text{False}\}\,. \tag{4}$$

In *GDP*, x represents a vector of continuous decision variables and $Y_{i,k}$ are Boolean variables. b_k is a scalar and $\gamma_{i,k}$ represents a fixed charge. The objective function comprises the sum of all fixed charges and a nonlinear term $\Phi(x)$. Whereas the model equations (1) and inequality constraints (2) hold irrespective

of discrete choices, there are further equations and inequality constraints (3) that are contained in n_k, $k \in K$, disjunctions. Each disjunction k may consist of several terms $i \in D_k$, where the index set D_k defines the number of terms for each disjunction. Note that exactly one term $i \in D_k$ holds per disjunction, that is, $\bigvee_{i \in D_k}$ is understood as an 'exclusive or' operator. The disjunctive constraints are only enforced if the Boolean variable value $Y_{i,k}$ is True. Otherwise, if $Y_{i,k}$ is False, the corresponding constraints are removed from the optimization problem.

The Boolean variables themselves are related to each other by so called propositional logic constraints (4). These logic constraints are used to model interrelationships between disjunctive constraints. For example, assume that the first disjunctive term from disjunction $k = 1$ has to be selected ($Y_{1,1} = \text{True}$) if another term from disjunction $k = 2$ is removed from the constraint set ($Y_{1,2} = \text{False}$). This situation can be expressed by the implication $\neg\, Y_{1,2} \Rightarrow Y_{1,1}$, which can be transformed into a constraint of type (4):

$$Y_{1,2} \vee Y_{1,1} = \text{True}\,. \tag{5}$$

Any optimization problem in disjunctive form *GDP* can be posed as an equivalent MINLP problem [5] by, for example, transforming the disjunctive constraints into big-M or binary multiplication constraints and by replacing the Boolean variables $Y_{i,k}$ by binary variables $y_{i,k} \in \{0, 1\}$.

A problem reformulation based on binary multiplication is:

$$(BM) \qquad \min_{x,y} \quad \Phi(x) + \sum_{k \in K} b_k$$

$$\text{s.t.} \quad f(x) = 0\,,$$

$$g(x) \le 0\,,$$

$$y_{i,k} \cdot h_{i,k}(x) = 0\,, \tag{6}$$

$$y_{i,k} \cdot r_{i,k}(x) \le 0\,, \tag{7}$$

$$y_{i,k} \cdot (b_k - \gamma_{i,k}) = 0\,, \tag{8}$$

$$Ay \le a\,, \tag{9}$$

$$\sum_{i \in D_k} y_{i,k} = 1\,, \tag{10}$$

$$y_{i,k} \in \{0, 1\}\,, \quad i \in D_k\,, k \in K\,, \tag{11}$$

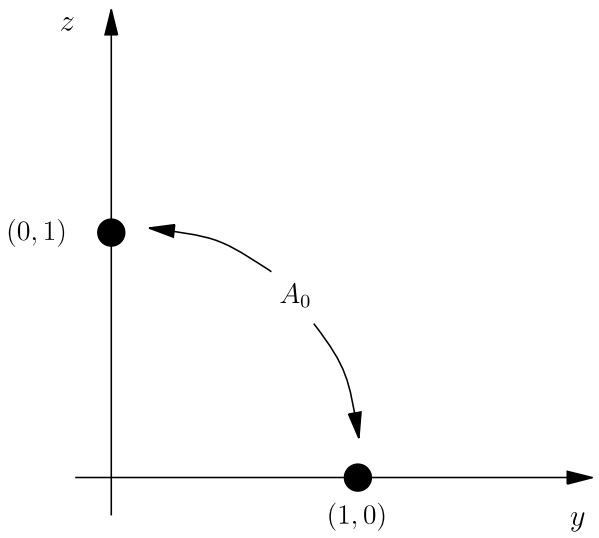
where each disjunctive constraint is multiplied by a variable $y_{i,k}$. If $y_{i,k} = 0$, the corresponding constraint becomes redundant. On the other hand, a constraint contained in a disjunction is enforced with $y_{i,k} = 1$. The propositional logic constraints (4) can be modeled by the linear constraints (9) on the binary variables. Note that with these inequalities not only exclusive but also inclusive 'or'-relations can be modeled, although a binary variable itself takes only *exclusively* the values 0 or 1. In fact, for two binary variables y_1 and y_2 the inclusive relation $y_1 + y_2 \ge 1$, modeling (5), becomes exclusive under the additional relation $y_1 + y_2 \le 1$.

It is important to note that the problem formulation *BM* has the drawback of being nonconvex even if the nonlinear, disjunctive constraints of the original optimization problem are convex. Thus, a problem reformulation based on binary multiplication would be employed only if the disjunctive optimization problem was nonconvex itself, as it is the case, for example, in a large portion of process engineering applications. Hence, this drawback should not be regarded as a strong limitation. Also note that the nonconvex expressions in (7) can be convexified if $r_{i,k}$ is a convex function [16]. However, in the following this assumption will not be made.

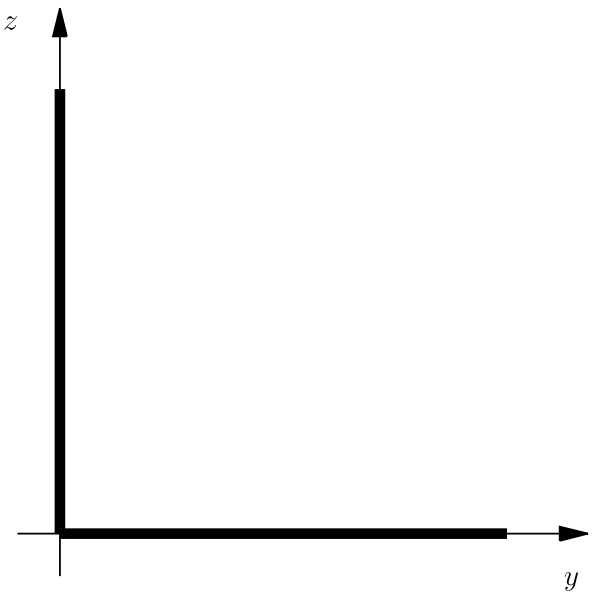
Formulations

Instead of applying an MINLP algorithm for solving the discrete-continuous optimization problem *BM* introduced in the previous section directly, one can reformulate the problem such that no discrete variables are present anymore. In particular, the discrete set defined in (10),(11) can be replaced by a set of restrictions involving continuous variables only, which can be used as constraints to form a purely continuous NLP. Since NLP solvers are usually designed to work with continuous variables, that is, variables from at least one-dimensional sets, the basic idea here is to increase the dimension of the constraint sets for $y_{i,k}$. Note that the discrete variables $y_{i,k}$ as defined in (11) are contained in a set of dimension zero.

For the explanation of the main ideas consider a single disjunction with $n_k = 2$ as it appears in (3). Put $y_k := y_{1,k}$ as well as $z_k := y_{2,k}$ and drop the fixed index k. This leads to a single binary decision variable



Continuous Reformulations of Discrete-Continuous Optimization Problems, Figure 1
Values of the discrete variable (y, z)



Continuous Reformulations of Discrete-Continuous Optimization Problems, Figure 2
The points which satisfy the complementarity condition (14),(15)

$y \in \{0, 1\}$ and its negation z, where the pair (y, z) can then attain exactly one of the values $(1, 0)$ and $(0, 1)$, that is,

$$(y, z) \in A_0 = \{(1, 0), (0, 1)\}, \tag{12}$$

(cf. Fig. 1). Hence, in this case the conditions (10),(11) are replaced equivalently by (12). In the general case $n_k \geq 2$ there are several ways to use the set A_0 to reformulate (10),(11) equivalently. A first possibility is to introduce additional variables $z_{i,k} = 1 - y_{i,k}$ and replace only (11) by the conditions $(y_{i,k}, z_{i,k}) \in A_0$, $i \in D_k$.

Note that the constraint (10) guarantees that exactly one of the variables $y_{i,k}$, $i \in D_k$, is equal to 1, since these variables *can only take* the values 0 and 1. This restriction can be relaxed in conjunction with an alternative approach for modeling binary decision variables explained below. Having these later developments in mind, note that an alternative reformulation of (10), (11) using A_0 is

$$\left(y_{i,k}, \sum_{j \in D_k \setminus \{i\}} y_{j,k} \right) \in A_0, \quad i \in D_k, k \in K. \tag{13}$$

An advantage of the latter reformulation is that it does not increase the problem dimension by auxiliary variables $z_{i,k}$.

Representing the Discrete Decisions by Approximate Continuous Variables

There are a number of ways to describe A_0 with continuous constraints. A suggestion of [9,11] is to replace (12) with the equivalent set of constraints

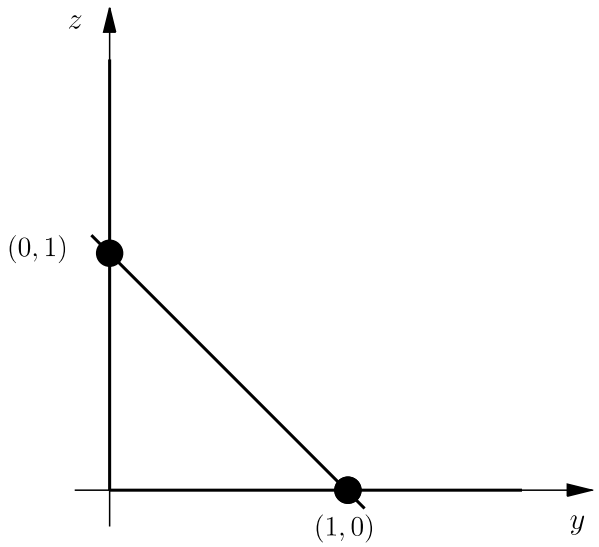
$$y \cdot z = 0, \tag{14}$$

$$y \geq 0, z \geq 0, \tag{15}$$

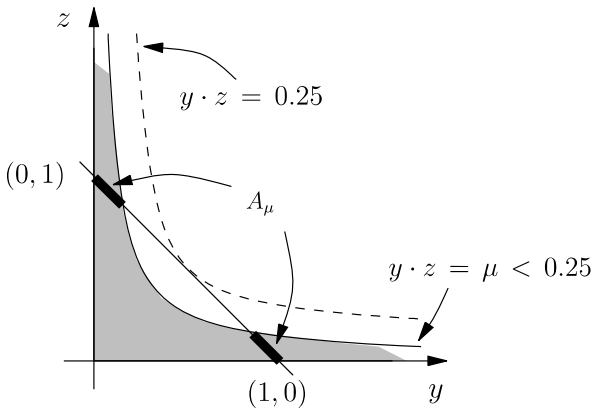
$$y + z = 1. \tag{16}$$

In fact, the constraints (14),(15) are known as a *complementarity* condition. They model a piecewise linear set with one kink at the origin in $\mathbb{R}^2$, as depicted in Fig. 2. Together with the constraint (16) one obtains exactly the set A_0 (cf. Fig. 3).

It is well-known that sets whose description contains complementarity conditions are not easy to treat numerically. In fact, the so-called Mangasarian–Fromovitz constraint qualification is violated everywhere in the feasible set as soon as a complementarity condition appears [14]. This constraint qualification, however, is known to characterize the (numerical) stability of the described set [6,13].



Continuous Reformulations of Discrete-Continuous Optimization Problems, Figure 3
Modeling discrete variables with a complementarity condition



Continuous Reformulations of Discrete-Continuous Optimization Problems, Figure 4
Continuous variables for the relaxed complementarity condition

There are many suggestions on how a complementarity condition can be treated numerically, in particular in the literature on so-called mathematical programs with equilibrium constraints (MPECs) which are optimization problems with complementarity conditions in the constraints [7,8].

One approach is to use regularization techniques, for example to replace the condition (14) by its relaxation $y \cdot z \leq \mu$ with some positive parameter μ. The idea is to trace the solutions of the corresponding auxiliary problems to a solution of the original problem while driving μ to zero.

For the reformulation of binary variables this approach means that the discrete set A_0 is replaced by the one-dimensional set

$$A_\mu = \{(0,1) + t \cdot (1,-1) | t \in [0, 0.5 - \sqrt{0.25 - \mu}] \cup [0.5 + \sqrt{0.25 - \mu}, 1]\},$$

which is disconnected for $\mu < 0.25$ as illustrated in Fig. 4.

Hence, this approach replaces discrete by continuous variables, at least via an approximation. [15] refers to the variables from the set A_μ as *approximate continuous*. In view of (13), a possible approximation of binary variables from a general disjunction is

$$\left(y_{i,k}, \sum_{j \in D_k \setminus \{i\}} y_{j,k} \right) \in A_\mu, \quad i \in D_k, \; k \in K$$

with $\mu > 0$.

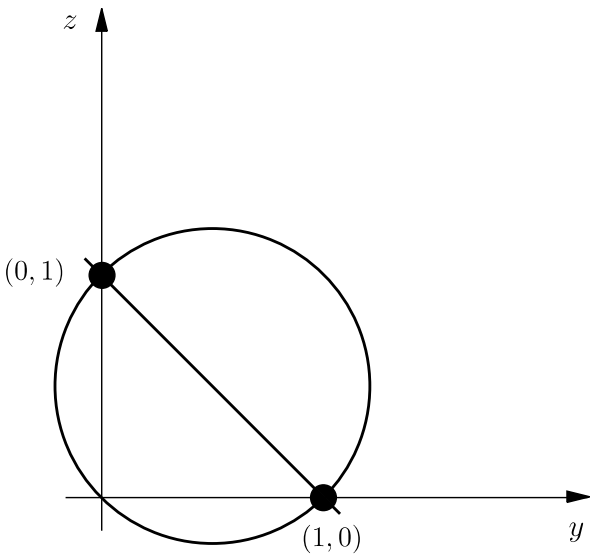
Note that there are two serious drawbacks of the reformulation by a complementarity condition. First, a look at Fig. 3 shows that the kink at the origin is irrelevant for the description of A_0 because of the additional constraint (16). Thus, there is no need to use the numerically demanding complementarity condition together with (16), but any function with a smooth zero set and the correct intersection points would do. For example, one can use the constraint

$$\left(y - \frac{1}{2}\right)^2 + \left(z - \frac{1}{2}\right)^2 = \frac{1}{2},$$

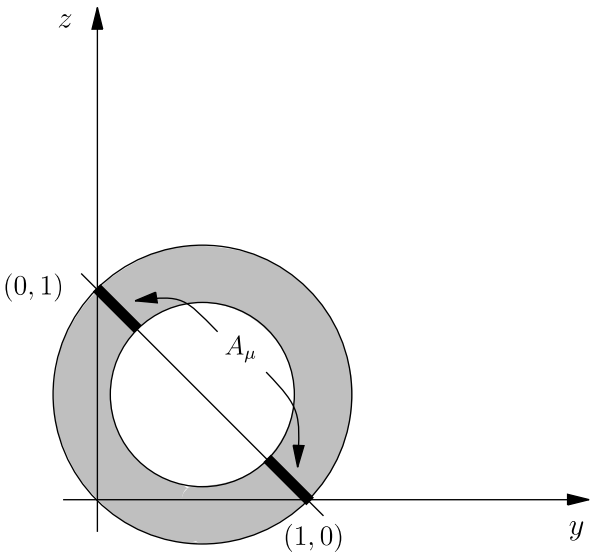
which is illustrated in Fig. 5.

Here, the Mangasarian–Fromovitz constraint qualification and even the stronger linear independence constraint qualification are satisfied everywhere in the set A_0 (for background information on constraint qualifications see [1]). This can be seen as an important advantage when compared to the properties of A_0 represented by the complementarity condition.

A second drawback which the circle condition shares with the reformulation by a complementarity condition is that the variables are still contained in the



Continuous Reformulations of Discrete-Continuous Optimization Problems, Figure 5
The circle condition



Continuous Reformulations of Discrete-Continuous Optimization Problems, Figure 6
The circle relaxation

discrete set A_0. In order to obtain a one-dimensional set one can again relax the conditions that describe A_0 to obtain a set A_ν corresponding to A_μ from the MPEC relaxation above. In fact, the constraints

$$\left(y-\frac{1}{2}\right)^2+\left(z-\frac{1}{2}\right)^2 \le \frac{1}{2}$$
$$\left(y-\frac{1}{2}\right)^2+\left(z-\frac{1}{2}\right)^2 \ge \left(\frac{1}{\sqrt{2}}-\nu\right)^2$$
$$y+z=1\,,$$

with $\nu > 0$ describe sets A_ν (cf. Fig. 6), which correspond to the sets $A_\mu(\mu > 0)$ via a reparametrization, that is, one arrives at the same set of approximate continuous variables.

Unfortunately, for the limiting case $\nu = 0$ the circle is not described by one equality constraint but by two inequalities with gradients pointing in opposite directions, so that the Mangasarian–Fromovitz constraint qualification is then again violated in A_0.

Representing the Discrete Decisions by Exact Continuous Variables

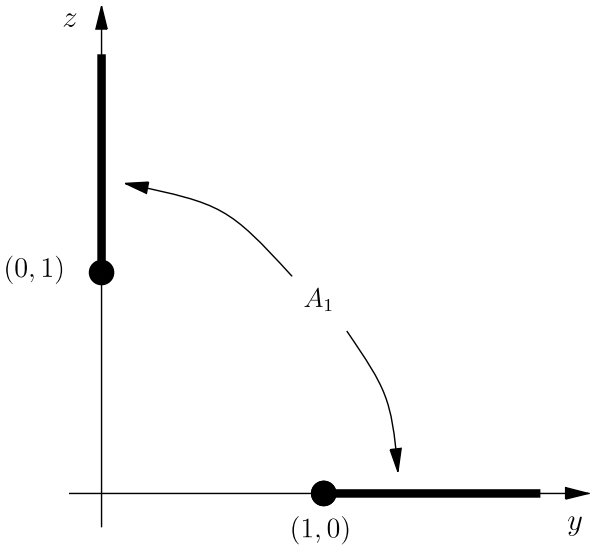
Although both the reformulation by a complementarity condition and the reformulation by a circle condition lead to well performing numerical methods for small examples [11,15] they share two intrinsic drawbacks:

- the replacement for A_0 is one-dimensional, but only approximate,
- in the (limiting) case of an exact description for A_0, the Mangasarian–Fromovitz constraint qualification is violated,
- the one-dimensional set becomes discrete if equality constraints are inconsistent (see below).

Since these properties may affect the numerical solution of large problems, [15] proposes a different continuous reformulation of the integrality constraints with better theoretical features.

The subsequent considerations are based on an alternative model reformulation that allows to replace the discrete decision variables defined in (10) by variables $y_{i,k}$, which are *not* defined on a discrete set as, for example, A_0. In fact, this model reformulation has the property of being equivalent to the corresponding disjunctive optimization problem in conjunction with one-dimensional rather than discrete variables $y_{i,k}$. Before describing the model reformulation in detail, we focus on the variables $y_{i,k}$ and show how a continuous, one-dimensional set A_1 can be defined using appropriate constraints.

In fact, since in *BM* any disjunctive constraint is not only enforced by $y_{i,k} = 1$, but alternatively also by



Continuous Reformulations of Discrete-Continuous Optimization Problems, Figure 7
A one-dimensional feasible set for (y, z).

$y_{i,k} \geq 1$, one may define $y_{i,k}$ as continuous variables of dimension one according to:

$$y_{i,k} \in \{0\} \cup [1, \infty)\,.$$

Of course, now the negation of $y_{i,k}$ in general does not coincide with $1 - y_{i,k}$, as the value of $y_{i,k}$ might exceed 1. On the other hand, for the case $n_k = 2$ as above we obtain

$$(y, z) \in A_1 = ([1, \infty) \times \{0\}) \cup (\{0\} \times [1, \infty)) \quad (17)$$

(cf. Fig. 7). Hence, the negation of $y_{i,k}$ is coded by the variable $z_{i,k}$. Moreover, one can now describe the binary decisions via

$$\left(y_{i,k}\,,\ \sum_{j \in D_k \setminus \{i\}} y_{j,k} \right) \in A_1\,, \quad i \in D_k\,,\ k \in K\,. \quad (18)$$

The set A_1 is obviously one-dimensional and is an *exact* rather than approximate model of a discrete decision. Therefore, the variables defined by the set A_1 are referred to as *exact continuous*.

To be able to apply an NLP solution algorithm, one has to describe A_1 by continuous constraints. One possibility, of course, is to use the (degenerate) complementarity condition (14),(15) with the additional constraint $y + z \geq 1$. However, it is also possible to choose a function with an appropriate zero set, such that the linear independence constraint qualification holds everywhere in the feasible set. A function with these properties is the so-called *Fischer–Burmeister* function $\varphi_{FB}(y, z) = y + z - \sqrt{y^2 + z^2}$. This means that one can write

$$A_1 = \{\,(y, z) \in \mathbb{R}^2 \,|\, \varphi_{FB}(y, z) = 0\,,\ y + z \geq 1\,\}\,. \quad (19)$$

Equivalently, one could use a multitude of other so-called NCP-functions (for a survey see [2]). NCP-functions are used for the description of nonlinear complementarity problems. They are designed such that their zero set coincides with the set defined by (14),(15) (cf. Fig. 2). A description like (19) reveals better numerical properties than the original description via (14),(15). For example, whereas the Mangasarian–Fromovitz constraint qualification is violated everywhere in the set under a description via (14),(15), the description as the zero set of the Fischer–Burmeister function even leads to the validity of the linear independence constraint qualification everywhere in the set, except for the origin (which does not play a role here). In terms of the Fischer–Burmeister function, and using (19), the condition (18) is equivalent to

$$\varphi_{FB}\left(y_{i,k}\,,\ \sum_{j \in D_k \setminus \{i\}} y_{j,k} \right) = 0\,, \quad i \in D_k\,, k \in K\,,$$
$$\sum_{i \in D_k} y_{i,k} \geq 1\,,\ k \in K\,.$$

Modeling Propositional Logic Constraints with Exact Continuous Variables

The question remains how logical conditions on two logical variables Y_1 and Y_2 should be modeled when (y_1, z_1) and (y_2, z_2) are not discrete but continuous as proposed in (17). This can easily be done by adding inequality constraints. In fact, $Y_1 \wedge Y_2$ is true if and only if $y_1 \geq 1$ and $y_2 \geq 1$. Moreover, $Y_1 \vee Y_2$ is true if and only if $y_1 + y_2 \geq 1$. For the negation of Y_1 one may *not* use $1 - y_1$, as y_1 might take a value strictly larger than one. On the other hand, for $n_k = 2$ the negation of Y_1 is already coded in the variable z_1. Moreover, for $n_k > 2$ the negation of $y_{i,k}$ is coded in $\sum_{j \in D_k \setminus \{i\}} y_{j,k}$, and one can proceed as above. Just like in the discrete case, inclusive as well as exclusive 'or'-relations can be modeled with exact continuous variables.

To circumvent the introduction of nonconvexity into the model by binary multiplication in *BM*, [15] presents an alternative, convex reformulation approach on the basis of tailored big-M constraints which can also be used in conjunction with exact continuous variables as defined in equation (17). A distinctive property of the binary multiplication-based model formulation *BM*, however, is the treatment of inconsistent equality constraints.

The Case of Inconsistent Equalities

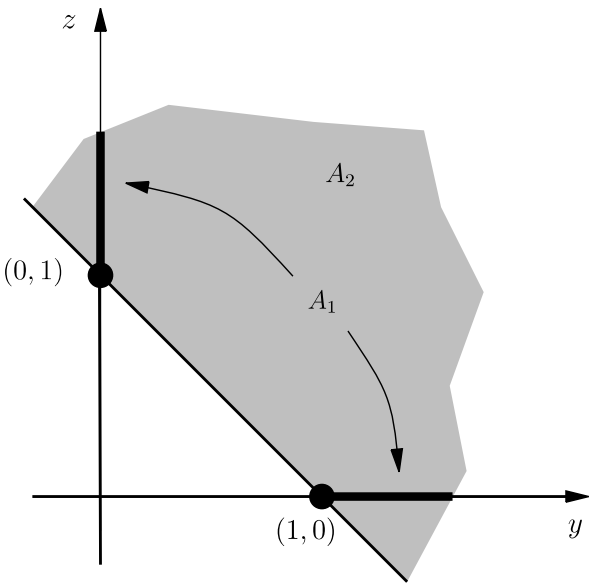
In many applications, the constraints (6)–(8) in *BM* lead to implicit restrictions on the exact continuous variables. In particular, (6) and (8) have to hold simultaneously for $i \in D_k, k \in K$. In process engineering applications, the underlying equations (i. e. $h_{i,k}(x) = 0,\ i \in D_k$ as well as $b_k - \gamma_{i,k} = 0,\ i \in D_k$) are often inconsistent for fixed $k \in K$, that is, they do not admit a common solution or, put geometrically, the sets described by these equations are disjoint. Note that this is an inherent property of a GDP problem with so-called disjoint disjunctions which have non-empty intersecting feasible regions [17].

This is particularly the case, if for fixed $k \in K$ the values $\gamma_{i,k}$ are pairwise distinct for $i \in D_k$. It implies that *at most* one of the variables $y_{i,k}, i \in D_k$, is non-vanishing. In the case $n_k = 2$ with $y = y_1$ and $z = y_2$ this means that the equation $y \cdot z = 0$ holds automatically. As a consequence, the only constraint needed for the description of A_1 (cf. Fig. 7) is $y + z \geq 1$, that is, the set

$$A_2 = \{\, (y, z) \in \mathbb{R}^2 \mid y + z \geq 1 \,\}$$

coincides with A_1 in the case of inconsistent equalities (cf. Fig. 8).

Although the pair (y, z) does not vary in the complete two-dimensional set A_2 from Fig. 8, in the restrictions one does not code the same information twice. This can be expected to lead to better numerical performance when NLP solvers are applied.



Continuous Reformulations of Discrete-Continuous Optimization Problems, Figure 8
A two-dimensional feasible set for (*y*, *z*)

Conclusions

In [15] several example problems involving discrete and continuous decision variables from process engineering are treated numerically, with approximate as well as exact continuous variables representing the discrete decisions. It is shown that, using these reformulations, an efficient numerical treatment of disjunctive optimization problems is possible, but one can only expect to find local solutions when using standard NLP solvers. This is due to the fact that any continuous reformulation of a disjunctive optimization problem leads to a nonconvex optimization problem. Consequently, the reformulation approaches may be combined with global optimization algorithms whenever the problem size admits to do so.

See also

- ► Disjunctive Programming
- ► Mixed Integer Programming/Constraint Programming Hybrid Methods
- ► Order Complementarity

References

1. Bazaraa M, Sherali H, Shetty C (1993) Nonlinear Programming. Wiley, Hoboken, New Jersey

2. Chen B, Chen X, Kanzow C (2000) A penalized Fischer–Burmeister NCP-function. Math Program 88:211–216
3. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, New York
4. Giannesi F, Niccolucci F (1976) Connections between nonlinear and integer programming problems. In: Institut Nazionale di Alta Mathematica (ed) Symposia Mathematica, vol XIX. Acad. Press, New York, pp 161–176
5. Grossmann IE, Hooker J (2000) Logic based approaches for mixed integer programming models and their application in process synthesis. In: Malone M, Trainham J, Carnahan B (eds) Foundations of Computer-Aided Process Design 323. AIChE Symp. Series. CACHE Publications, Austin, Texas, pp 70–83
6. Jongen HT, Weber G-W (1991) Nonlinear optimization: characterization of structural stability. J Glob Optim 1: 47–64
7. Leyffer S (2006) Complementarity constraints as nonlinear equations: theory and numerical experiences. In: Dempe S, Kalashnikov V (eds) Optimization and Multivalued Mappings. Springer, Dordrecht, pp 169–208
8. Luo Z, Pang J, Ralph D (1996) Mathematical Programs with Equilibrium Constraints. Cambridge University Press, Cambridge
9. Pardalos PM (1994) The linear complementarity problem. In: Gomez S, Hennart JP (eds) Advances in Optimization and Numerical Analysis. Springer, New York, pp 39–49
10. Pardalos PM, Prokopyev OA, Busygin S (2006) Continuous approaches for solving discrete optimization problems. In: Appa G, Pitsoulis L, Williams HP (eds) Handbook on Modelling for Discrete Optimization. Springer, New York, pp 39–60
11. Raghunathan A, Biegler L (2003) Mathematical programs with equilibrium constraints (MPEC) in process engineering. Comp Chem Eng 27(10):1381–1392
12. Raman R, Grossmann IE (1994) Modelling and computational techniques for logic based integer programming. Comput Chem Eng 18(7):563–578
13. Robinson S (1976) Stability theory for systems of inequalities, part II: differentiable nonlinear systems. SIAM J Numer Anal 13:497–513
14. Scheel H, Scholtes S (2000) Mathematical programs with complementarity constraints: stationarity, optimality, and sensitivity. Math Oper Res 25:1–22
15. Stein O, Oldenburg J, Marquardt W (2004) Continuous reformulations of discrete-continuous optimization problems. Comput Chem Eng 28:1951–1966
16. Stubbs R, Mehrotra S (1999) A branch-and-cut method for 0-1 mixed convex programming. Math Program 86: 515–532
17. Vecchietti A, Lee S, Grossmann IE (2003) Modeling of discrete/continuous optimization problems: characterization and formulation of disjunctions and their relaxations. Comput Chem Eng 27(3):433–448

Continuous Review Inventory Models: (*Q*, *R*) Policy

Ismail Capar[1], Burak Eksioglu[2]
[1] Department of Engineering Technology and Industrial Distribution, Texas A&M University, College Station, USA
[2] Department of Industrial and Systems Engineering, Mississippi State University, Mississippi State, USA

MSC2000: 49-02, 90-02

Article Outline

Keywords
Introduction
Models
  Single-Echelon Models
  Multi-Echelon Models
Conclusions
References

Keywords

Continuous review inventory models; (*Q*, *R*) models

Introduction

Inventory control is an important issue in supply chain management. Today, many different approaches are used to solve the complicated inventory control problems. While some of the approaches use a periodic review cycle, others use methods based on continuous review of inventory. In this survey, stochastic inventory theory that is based on continuous review is analyzed.

One of the challenging tasks in continuous review inventory problems is finding the order quantity (Q) and the reorder point (R) such that the total cost is minimized and fill rate constraints are satisfied. The total cost includes ordering cost, backorder cost, and inventory holding cost. The fill rate is defined as the fraction of demand satisfied from inventory on hand. Under the continuous inventory control methodology, when the inventory position (on-hand inventory plus outstanding orders minus backorders) drops down to or below a reorder point, R, an order of size Q is placed.

Although they are all the same, there are many different representations of this inventory model such as

(*Q, r*) (Boyaci and Gallego [5]), (*Q, R*) (Hing et al. [14]), (*R, Q*) (Axsater [2,3] and Marklund [16]). In addition, for some of the problems it is assumed that the order quantity (nQ) is a multiple of minimum batch size, Q. Here, n is the minimum integer required to increase the inventory position to above R (Chen and Zheng[8]). In this case, the problem is formulated as a (R, nQ) type model.

Models

When the literature of (*Q, R*) models is investigated, some similarities and differences among the publications can easily be identified. Thus, the publications can be classified according to those similarities and differences. Two of the most distinctive attributes of (*Q, R*) models are as follows:

1. Type of supply chain: While some articles only consider one entity that uses a (*Q, R*) policy [1,5,14], others consider a multi echelon inventory system [2,3,8,16].
2. Exact evaluation or near-optimal evaluation: The (*Q, R*) inventory problems are not easy to solve; thus, many of the research papers give approximate solution approaches or try to find bounds on the solutions [1,2,4,5,20], only a small number of articles give the exact evaluation of the (*Q, R*) inventory system [3,9,21].

In the next section, the literature based on type of supply chain considered and the evaluation methods used is reviewed. First, heuristic methods are analyzed. Second, publications providing optimal methods are reviewed. In the last section, we give some concluding remarks.

Single-Echelon Models

Hing et al. [14] focus on average inventory level approximation in a (*Q, R*) system with backorders. They compare different approaches proposed in the literature. Their numerical analysis shows that the approximation developed by Hadley and Whitin [13], 1/2Q + safety stock, is more robust than other approximations that have been proposed so far. Then, the authors propose a new methodology based on spreadsheet optimization. Using numerical examples they show that spreadsheet optimization based approach is better than those methods proposed in the literature.

Agrawal and Seshadri [1] provide upper and lower bounds for optimal R and Q subject to fill rate constraints. Although the authors consider backorder cost, the algorithm that was developed to find bounds can be used when backorder costs are zero. Another important application of the algorithm is that it can be applied when there are no service level constraints.

Like Agrawal and Seshadri [1], Platt et al. [19] also consider fill rate constraints and propose two heuristics that can be used for (*Q, R*) policy models. While the first heuristic is suitable for deterministic lead time demand models, the second one assumes that demand during the lead time follows a normal distribution. Both heuristics are used to find R and Q values. The authors compare the proposed heuristics with others that have been proposed in the literature. Their analysis shows that their heuristics do not necessarily outperform the other heuristics in each problem instance.

Boyaci and Gallego [5] propose a new (*Q, R*) model that minimizes average holding and ordering costs subject to upper bounds on the expected and maximum waiting times for the backordered items. They provide optimality conditions and an exact algorithm for the problem. Boyaci and Gallego [5] conclude their study by performing a numerical analysis.

Gallego [12] proposes heuristics to find distribution-free bounds on the optimal cost and optimal batch size when a (*Q, R*) policy is used. He also shows that the heuristics work well when the demand distribution is Poisson or compound Poisson.

Bookbinder and Cakanyildirim [4] consider a (*Q, R*) policy where lead time is not constant. They treat lead time as a random variable and develop two probabilistic models. While in the first model the lead time is fixed, in the second model the lead time can be reduced by using an expediting factor (τ). The order quantity, reorder point, and expediting factor are the the three decision variables in the second model. The authors show that for both models the expected cost per unit time is jointly convex. They also make a sensitivity analysis with respect to cost parameters.

Ryu and Lee [20] consider the lead time as a decision variable. However, in this study the demand is constant. In their model, Ryu and Lee [20] assume that there are two suppliers for the items to be procured. They mainly consider two cases. In the first case, lead time cannot be decreased but in the second case, or-

ders can be expedited. The authors also assume that lead-time distributions are non-identical exponential. For the first case, their objective is to determine a *Q*, an *R*, and an order-splitting proportion. In the second case, they find new values for the lead times using the order-splitting proportion. Their sensitivity analysis shows that the order-splitting proportion tends to be a half, and it is biased by the coefficient of the expediting function.

Cakanyildirim et al. [6] develop a model that considers lead-time variability. The authors assume that lead time is effected by both the lot size and the reserved capacity. The authors come up with a closed-form solution for the situation where lead time is proportional to the lot size. Cakanyildirim et al. [6] also present the effect of linear and concave lead times on the value of cost function. In the model, in addition to the order quantity and the reorder point, the reserved capacity is also a decision variable. Finally, the authors consider a case in which fixed proportion of capacity is allocated at the manufacturing facility.

Most of the articles in the literature consider lead time as a constant and focus on demand during the lead time. However, Wu and Ouyang [21] assume that lead time is a decision variable and that lead-time demand follows a normal distribution. They also assume that an arrival order may contain some defective parts and that those parts will be kept in inventory until next delivery. Moreover, they include an inspection cost for defective parts to the model. Their model is defined as (*Q, R, L*) inventory model where order quantity (*Q*), reorder point (*R*), and lead time (*L*) are decision variables. The objective is to minimize the total cost which includes ordering costs, inventory holding costs (defective and non-defective), lost sales costs, backorder costs, and inspection costs. The authors present an algorithm to find the optimal solutions for the given problem.

Duran et al. [9] present a (*Q, R*) policy where orders can be expedited. At the time of order release, if inventory position is less than or equal to a critical value r_e, the order is expedited at an additional cost. If the inventory level is higher then r_e and lower than or equal to the reorder point *R*, then order is not expedited. The aim is to find the order quantity (*Q*), the reorder point (*R*), and the expediting point r_e which minimize average cost (note that this does not include backorder costs). The authors present an optimal algorithm to obtain the *Q*, *R*, and r_e values if they are restricted to be integers.

The model proposed by Kao and Hsu [15] is different from other models reviewed in this paper because the authors discuss the order quantity and reorder point with fuzzy demand. Kao and Hsu [15] use this fuzzy demand to construct the fuzzy total inventory cost. The authors derive five pairs of simultaneous nonlinear equations to find the optimal order quantity *Q* and the reorder point *R*. The authors show that when the demand is a trapezoid fuzzy number, the equations can be reduced to a set of closed-form equations. Then, they prove that the solution to these equations give an optimal solution. Kao and Hsu [15] also present a numerical example to show that the solution methodology developed in the paper is easy to apply in practice.

Multi-Echelon Models

Moinzadeh and Lee [18] present a model to determine the batch size in a multi-echelon system with one central depot and *M* sites. In their problem, when the number of failed items is equal to the order quantity *Q* at any site, then those items are sent to the depot. If the depot has sufficient inventory on hand, it delivers the items immediately; otherwise, the items are backlogged. Although all sites use a (*Q, R*) policy, the depot uses a (*S-1, S*) policy. In other words, whenever the depot receives an order of size *Q*, it places an order simultaneously to replenish its stock. After determining the *Q* and *R* values for each site, the authors use an approximation to estimate the total system stock and the backorder levels. The numerical results show that the (*Q, R*) policy is better than the (*S-1, S*) policy for such systems.

Forsberg [10] deals with a multi-echelon inventory system with one warehouse and multiple non-identical retailers. The author assumes that the retailers face independent Poisson demand and both the warehouse and the retailers use (*Q, R*) policies. Forsberg [10] evaluates inventory holding and shortage costs using an exact solution approach.

Chen and Zheng [8] study a (*nQ, R*) policy in a multi-stage serial inventory system where *stage 1* orders from *stage 2*, *stage 2* from *stage 3*, etc., and *stage N* places orders to an outside supplier with unlimited capacity. The demand seen by *stage 1* is compound Pois-

son and excess demand is backlogged at every stage. The transportation lead times among stages are constant. By using a two-step approach, Chen and Zheng [8] provide near-optimal solution. In the first step, they find the lower and upper bounds on the cost function by changing the penalty cost of being short on inventory. In the second step, the authors minimize the bounds by using three different heuristic approaches. Chen and Zheng [8] also propose an optimal algorithm that requires additional computational effort.

Axsater [2] considers a two-stage inventory system with one warehouse and N non-identical retailers. He presents an exact method to evaluate inventory holding and shortage costs when there are only two retailers. He focuses on the timing of the warehouse orders for the sub-batches of Q. He identifies three possibilities and evaluates the cost for each case separately. At the end, total cost is calculated by summing the costs for the three cases. When there are more than two retailers, he extends his evaluation technique by combining the retailers into two groups, and then uses the same approach he developed for the two retailer case. The author also presents a model where the lead times are constant and all facilities use (Q, R) policies with different Q and R values. In this model, all stockouts are backordered, delayed orders are delivered on a first-come-first-served basis, and partial shipments are also allowed. In order to simplify the problem, Axsater [2] assumes that all batch sizes are multiples of the smallest batch size. In the objective function, the author only considers expected inventory holding cost and backorder cost.

Like Axsater [2], Marklund [16] also considers a two-stage supply chain with one central warehouse and an arbitrary number of non-identical retailers. Customer demands occur only at the retailers. The retailers use (Q, R) policies with different parameters, and they request products from the central warehouse whenever their inventory positions reach R or fall below R. The author proposes a new policy (Q_0, a_0) that is motivated by relating the traditional echelon stock model to the installation stock (Q, R) model where the order quantity Q is a multiple of a minimum batch size. In the article, Marklund [16] gives the detailed derivation of the exact cost function when the retailers use (Q, R) policies and the warehouse uses the new (Q_0, a_0) policy. The performance of the new policy is compared to traditional echelon stock policy and (Q, R) policy through numerical examples. Although the results show that the proposed policy outperforms the other policies in all numerical examples, the author does not guarantee that the policy will always give the best result.

Fujiwara and Sedarage [11] apply a (Q, R) policy for a multi-part assembly system under stochastic lead times. The objective of the article is to simultaneously determine the order quantity and the assembly lot size so that the average total cost per unit time is minimized. The total cost includes setup costs, inventory holding costs of parts and assembled items, and shortage costs of assembled items. The authors try to find separate reorder points, r_i, for each part and a global order quantity, Q, which will be used for all parts. Although the authors propose a global order quantity Q, they also mention that this kind of policy may not be optimal. They suggest that instead of a global Q, a common Q where all order quantities are multiples of Q might be more sensible.

Chen and Zheng [7] consider a distribution system with one warehouse and multiple retailers. The retailers' demands follow an independent compound Poisson process. It is assumed that the order quantity is a multiple of the smallest batch size. The order quantity and the reorder point are calculated by using a heuristic. The authors present an exact procedure for evaluating the performance (average cost) of the (nQ, R) policy when the demand is a Poisson process. Chen and Zheng [7] also give two approximation procedures for the case with compound Poisson processes. The approximations are based on exact formulations of the case with Poisson processes.

Axsater [3] presents an exact analysis of a two-stage inventory system with one warehouse and multiple retailers. The demand for each retailer follows an independent compound Poisson process. The retailers replenish their stock from the warehouse, and the warehouse replenishes its stock from an outside supplier. The transportation times from the warehouse to the retailers and from the outside supplier to the warehouse are constant. In addition, if there is a shortage, then additional delay may also occur since shortages and stockouts are backordered. The author emphasizes that the approach developed is not directly applicable for items with large demand. Instead, it is suitable mostly for slow-moving parts such as spare parts.

Moinzadeh [17] also considers a supply chain with one warehouse and multiple identical retailers. The author assumes that demand at the retailers is random but stationary and that each retailer places its order according to a (Q, R) policy. In addition, Moinzadeh [17] assumes that the warehouse receives online information about the demand. The author shows the effect of information sharing on order replenishment decisions of the supplier. In the article, the author first proposes a possible replenishment policy for the supplier and then provides an exact analysis for the operating measures of such systems. The author concludes the article by giving information about when information sharing is most beneficial.

Conclusions

We provide a literature review on continuous review (Q, R) inventory policies. Although we review most of the well known papers that deal with (Q, R) policy, this is not an exhaustive review of the literature. Our aim is to present the importance of the (Q, R) policy and show possible extensions of the simple (Q, R) model.

References

1. Agrawal V, Seshadri S (2000) Distribution free bounds for service constrained (Q, r)inventory systems. Nav Res Logist 47:635–656
2. Axsater S (1998) Evaluation of installation stock based (R, Q)-policies for two-level inventory systems with poisson demand. Oper Res 46(3):135–145
3. Axsater S (2000) Exact analysis of continuous review (R, Q) policies in two-echelon inventory systems with compound poisson demand. Oper Res 48(5):686–696
4. Bookbinder JH, Cakanyildirim M (1999) Random lead times and expedited orders in (Q, r) inventory systems. Eur J Oper Res 115:300–313
5. Boyaci T, Gallego G (2002) Managing waiting times of backordered demands in single-stage (Q, r) inventory systems. Nav Res Logist 49:557–573
6. Cakanyildirim M, Bookbinder JH, Gerchak Y (2000) Continuous review inventory models where ran- dom lead time depends on lot size and reserved capacity. Int J Product Econ 68:217–228
7. Chen F, Zheng YS (1997) One warehouse multi-retailer system with centralized stock information. Oper Res 45(2): 275–287
8. Chen F, Zheng YS (1998) Near-optimal echelon-stock (R, nQ) policies in multistage serial systems. Oper Res 46(4):592–602
9. Duran A, Gutierrez G, Zequeira RI (2004) A continuous review inventory model with order expediting. Int J Product Econ 87:157–169
10. Forsberg R (1997) Exact evaluation of (R, Q)-policies for two-level inventory systems with Poisson demand. Eur J Oper Res 96:130–138
11. Fujiwara O, Sedarage D (1997) An optimal (Q, r) policy for a multipart assembly system under stochastic part procurement lead times. Eur J Oper Res 100:550–556
12. Gallego G (1998) New bounds and heuristics for (Q, r) policies. Manag Sci 44(2):219–233
13. Hadley G, Whitin TM (1963) Analysis of inventory systems. Prentice-Hall, Englewood Cliffs, NJ
14. Hing A, Lau L, Lau HS (2002) A comparison of different methods for estimating the average inventory level in a (Q; R) system with backorders. Int J Product Eco 79:303–316
15. Kao C, Hsu WH (2002) Lot size-reorder point inventory model with fuzzy demands. Comput Math Appl 43:1291–1302
16. Marklund J (2002) Centralized inventory control in a two-level distribution system with poisson demand. Nav Res Logist 49:798–822
17. Moinzadeh K (2002) A multi-echelon inventory system with information exchange. Manag Sci 48(3):414–426
18. Moinzadeh K, Lee HL (1986) Batch size and stocking levels in multi-echelon repariable systems. Manag Sci 32(12): 1567–1581
19. Platt DE, Robinson LW, Freund RB (1997) Tractable (Q, R) heuristic models for constrained service level. Manag Sci 43(7):951–965
20. Ryu SW, Lee KK (2003) A stochastic inventory model of dual sourced supply chain with lead-time reduction. Int J Product Eco 81–82:513–524
21. Wu KS, Ouyang LY (2001) (Q, r, L) inventory model with defective item. Comput Ind Eng 39:173–185

Contraction-Mapping

C. T. Kelley
Department Math. Center for Research in Sci., North Carolina State University, Raleigh, USA

MSC2000: 65H10, 65J15

Article Outline

Keywords
Statement of the Result
Affine Problems
Nonlinear Problems
Integral Equations Example

See also
References

Keywords

Nonlinear equations; Linear equations; Integral equations; Iterative method; Contraction mapping

Statement of the Result

The method of *successive substitution*, *Richardson iteration*, or *direct iteration* seeks to find a *fixed point* of a map K, that is a point u^* such that

$$u^* = K(u^*).$$

Given an initial iterate u_0, the iteration is

$$u_{k+1} = K(u_k), \qquad \text{for} \quad k \geq 0. \tag{1}$$

Let X be a Banach space and let $D \subset X$ be closed. A map $K : D \to D$ is a *contraction* if

$$\|K(u) - K(v)\| \leq \alpha \, \|u - v\| \tag{2}$$

for some $\alpha \in (0, 1)$ and all $u, v \in D$. The contraction mapping theorem, [3,7,13,14], states that if K is a contraction on D then

- K has a unique fixed point u^* in D, and
- for any $u_0 \in D$ the sequence $\{u_k\}$ given by (1) converges to u^*.

The message of the contraction mapping theorem is that if one wishes to use direct iteration to solve a fixed point problem, then the fixed point map K must satisfy (2) for some D and relative to some choice of norm. The choice of norm need not be made explicitly, it is determined implicitly by the K itself. However, if there is no norm for which (2) holds, then another, more robust, method, such as Newton's method with a line search, must be used, or the problem must be reformulated.

One may wonder why a Newton-like method is not always better than a direct iteration. The answer is that the cost for a single iteration is very low for Richardson iteration. So, if the equation can be set up to make the contraction constant α in (2) small, successive substitution, while taking more iterations, can be more efficient than a Newton-like iteration, which has costs in linear algebra and derivative evaluation that are not incurred by successive substitution.

Affine Problems

An affine fixed point map has the form

$$K(u) = Mu + b$$

where M is a linear operator on the space X. The fixed point equation is

$$(I - M)u = b, \tag{3}$$

where I is the identity operator. The classical stationary iterative methods in numerical linear algebra, [8,13], are typically analyzed in terms of affine fixed point problems, where M is called the iteration matrix. Multigrid methods, [2,4,5,9], are also stationary iterative methods. We give an example of how multigrid methods are used later in this article.

The contraction condition (2) holds if

$$\|M\| \leq \alpha < 1. \tag{4}$$

In (4) the norm is the operator norm on X. M may be a well defined operator on more than one space and (4) may not hold in all of them. Similarly, if X is finite dimensional and all norms are equivalent, (4) may hold in one norm and not in another. It is known, [10], that (4) holds for some norm if and only if the spectral radius of M is < 1.

When (4) does not hold it is sometimes possible to form an approximate inverse *preconditioner* P so that direct iteration can be applied to the equivalent problem

$$u = (I - P(I - M))u - Pb. \tag{5}$$

In order to apply the contraction mapping theorem and direct iteration to (5) we require that

$$\|I - P(I - M)\| \leq \alpha < 1$$

in some norm. In this case we say that P is an *approximate inverse* for $I - M$. In the final section of this article we give an example of how approximate inverses can be built for discretizations of integral operators.

Nonlinear Problems

If the nonlinear fixed point map K is sufficiently smooth, then a Newton-like method may be used to solve

$$F(u) = u - K(u) = 0.$$

The transition from a current approximation u_c of u^* to an update u_+ is

$$u_+ = u_c - P(u_c - K(u_c)), \tag{6}$$

where

$$P \approx F'(u^*)^{-1} = (I - K'(u^*))^{-1}.$$

$P = F(u_c)^{-1}$ is Newton's method and $P = F'(u_0)^{-1}$ is the chord method.

It is easy to show [7,13,14] that if u is near u^* and P is an approximate inverse for $F'(u^*)$ then the preconditioned fixed point problem

$$u = u - P(u - K(u))$$

is a contraction on a neighborhood D of u^*. This is, in fact, one way to analyze the convergence of Newton's method. In this article our focus is on preconditioners that remain constant for several iterations and do not require computation of the derivative of K.

The point to remember is that, if the goal is to transform a given fixed point map into a contraction, preconditioning of nonlinear problems can be done by the same process (formation of an approximate inverse) as for linear problems.

Integral Equations Example

We close this article with the *Atkinson–Brakhage preconditioner* for integral operators [2,4]. We will begin with the linear case, from which the nonlinear algorithm is a simple step. Let $\Omega \in \mathbf{R}^N$ be compact and let $k(x, y)$ be a continuous function on $\Omega \times \Omega$. We consider the affine fixed point problem

$$u(x) = f(x) + (\mathbf{K}u)(x) = f(x) + \int_\Omega k(x, y)u(y)\, \mathrm{d}y,$$

where $f \in X = C(\Omega)$ is given and a solution $u^* \in X$ is sought. In this example $D = X$. We will assume that the linear operator $I - \mathbf{K}$ is nonsingular on X.

We consider a family of increasingly accurate quadrature rules, indexed with a level l, with weights $\{w_i^l\}_{i=1}^{N_l}$ and nodes $\{x_i^l\}_{i=1}^{N_l}$ that satisfy

$$\lim_{l\to\infty} \sum_{j=1}^{N_l} f(x_j^l)w_j^l = \int_\Omega f(x)\, \mathrm{d}x$$

for all $f \in X$. The family of operators $\{\mathbf{K}_l\}$ defined by

$$\mathbf{K}_l u(x) = \sum_{j=1}^{N_l} k(x, x_j^l)u(y)w_j^l$$

converges strongly to $\mathbf{K}$, that is

$$\lim_{l\to\infty} \mathbf{K}_l u = \mathbf{K}u$$

for all $u \in X$. The family $\{\mathbf{K}_l\}$ is also *collectively compact*, [1]. This means that if $\mathbf{B}$ is a bounded subset of X, then

$$\cup_l \mathbf{K}_l(\mathbf{B})$$

is precompact in X. The direct consequences of the strong convergence and collective compactness are that $I - \mathbf{K}_l$ are nonsingular for l sufficiently large and

$$(I - \mathbf{K}_l)^{-1} \to (I - \mathbf{K})^{-1} \tag{7}$$

strongly in X. The Atkinson–Brakhage preconditioner is based on these results.

For $g \in X$ one can compute

$$v = (I - \mathbf{K}_l)^{-1} g$$

by solving the finite-dimensional linear system

$$v_i = g(x_i^l) + \sum_{j=1}^{N_l} k(x_i^l, x_j^l)v_j w_j^l \tag{8}$$

for the values $v(x_i^l) = v_i$ of v at the nodal points and then applying the *Nyström interpolation*

$$v(x) = g(x) + \sum_{j=1}^{N_l} k(x, x_j^l)v_j w_j^l = g(x) + (\mathbf{K}_l v)(x)$$

to recover $v(x)$ for all $x \in \Omega$. (8) can be solved at a cost of $O(N_l^3)$ floating point operations if direct methods for linear equations are used and for much less if iterative methods such as GMRES [15] are used. In that case, only $O(1)$ matrix-vector products are need to obtain a solution that is accurate to truncation error [6]. This is, up to a multiplicative factor, optimal. The Atkinson–Brakhage preconditioner can dramatically reduce this factor, however.

The results in [1] imply that

$$M_l = I + (I - \mathbf{K}_l)^{-1}\mathbf{K},$$

the Atkinson–Brakhage preconditioner, converges to $(I-K)^{-1}$ *in the operator norm*. Hence, for l sufficiently large (coarse mesh sufficiently fine) Richardson iteration can be applied to the system

$$u = u - M_l(I - \mathbf{K}_l)u - M_l f,$$

where $L \gg l$. Applying this idea for a sequence of grids or levels leads to the optimal form of the Atkinson–Brakhage iteration [11]. The algorithm uses a coarse mesh, which we index with $l = 0$, to build the preconditioner and then cycles through the grids sequentially until the solution at a desired fine ($l = L$) mesh is obtained. One example of this is a sequence of composite midpoint rule quadratures in which $N_{l+1} = 2N_l$. Then, [2,11], if the coarse mesh is sufficiently fine, only one Richardson iteration at each level will be needed. The cost at each level is two matrix vector products at level l and a solve at level 0.

1) Solve $u_0 - \mathbf{K}_0\, u_0 = f$; set $u = u_0$.
2) For $l = 1, \ldots, L$:
   a) Compute $r = u - \mathbf{K}_l u - f$;
   b) $u = u - M_0\, r$.

Nonlinear problems can be solved with exactly the same idea. We will consider the special case of *Hammerstein equations*

$$u(x) = \mathbf{K}(u)(x) = \int_\Omega k(x, y, u(y))\ \mathrm{d}y.$$

If we use a sequence of quadrature rules as in the linear case we can define

$$\mathbf{K}_l(u)(x) = \sum_{j=1}^{N_l} k(x, x_j^l, u(x_j^l))w_j^l.$$

The nonlinear form of the Atkinson–Brakhage algorithm for Hammerstein equations simply uses the approximation

$$I + (I - \mathbf{K}_0'(u_0))^{-1}\mathbf{K}'(u) \approx (I - \mathbf{K}_l'(u))^{-1}$$

in a Newton-like iteration. One can see from the formal description below that little has changed from the linear case.

1) Solve $u_0 - \mathbf{K}_0\,(u_0) = 0$; set $u = u_0$.
2) For $l = 1, \ldots, L$:
   a) Compute $r = u - \mathbf{K}_l(u)$;
   b) $u = u - (I + (I - \mathbf{K}_l'(u_0))^{-1}\, \mathbf{K}'(u))r$.

The Atkinson–Brakhage algorithm can, under some conditions, be further improved, [12] and the number of fine mesh operator-function products per level reduced to one. There is also no need to explicitly represent the operator as an integral operator with a kernel.

See also

- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Interval Analysis: Systems of Nonlinear Equations
- ▶ Nonlinear Least Squares: Newton-Type Methods
- ▶ Nonlinear Systems of Equations: Application to the Enclosure of all Azeotropes

References

1. Anselone PM (1971) Collectively compact operator approximation theory. Prentice-Hall, Englewood Cliffs, NJ
2. Atkinson KE (1973) Iterative variants of the Nyström method for the numerical solution of integral equations. Numer Math 22:17–31
3. Banach S (1922) Sur les opérations dans les ensembles abstraits et leur applications aux équations intégrales. Fundam Math 3:133–181
4. Brakhage H (1960) Über die numerische Behandlung von Integralgleichungen nach der Quadraturformelmethode. Numer Math 2:183–196
5. Briggs W (1987) A multigrid tutorial. SIAM, Philadelphia
6. Campbell SL, Ipsen ICF, Kelley CT, Meyer CD, Xue ZQ (1996) Convergence estimates for solution of integral equations with GMRES. J Integral Eq Appl 8:19–34
7. Dennis JE, Schnabel RB (1996) Numerical methods for nonlinear equations and unconstrained optimization. Classics Appl Math, vol 16, SIAM, Philadelphia
8. Golub GH, VanLoan CG (1983) Matrix computations. Johns Hopkins Univ. Press, Baltimore, MD
9. Hackbusch W (1985) Multi-grid methods and applications. Comput Math, vol 4. Springer, Berlin
10. Isaacson E, Keller HB (1966) Analysis of numerical methods. Wiley, New York
11. Kelley CT (1990) Operator prolongation methods for nonlinear equations. In: Allgower EL, Georg K (eds) Computational Solution of Nonlinear Systems of Equations. Lect Appl Math Amer Math Soc, Providence, RI, pp 359–388
12. Kelley CT (1995) A fast multilevel algorithm for integral equations. SIAM J Numer Anal 32:501–513
13. Kelley CT (1995) Iterative methods for linear and nonlinear equations. No. in Frontiers in Appl Math, vol 16 SIAM, Philadelphia
14. Ortega JM, Rheinboldt WC (1970) Iterative solution of nonlinear equations in several variables. Acad Press, New York

15. Saad Y, Schultz MH (1986) GMRES a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Statist Comput 7:856–869

Control Vector Iteration CVI

Rein Luus
Dept. Chemical Engineering, Univ. Toronto, Toronto, Canada

MSC2000: 93-XX

Article Outline

Keywords and Phrases
Optimal Control Problem
Second Variation Method
Determination of Stepping Parameter
Illustration of the First Variation Method
See also
References

Keywords and Phrases

Optimal control; Control vector iteration; Variation method; Pontryagin's maximum principle

In solving optimal control problems involving nonlinear differential equations, some iterative procedure must be used to obtain the optimal control policy. As is true with any iterative procedure, one is concerned about the convergence rate and also about the reliability of obtaining the optimal control policy. Although from Pontryagin's maximum principle it is known that the minimum of the performance index corresponds to the minimum of the Hamiltonian, to obtain the minimum value for the Hamiltonian is not always straightforward. Here we outline a procedure that changes the control policy from iteration to iteration, improving the value of the performance index at each iteration, until the improvement is less than certain amount. Then the iteration procedure is stopped and the results are analyzed. Such a procedure is called control vector iteration method (CVI), or iteration in the policy space.

Optimal Control Problem

To illustrate the procedure, let us consider the optimal control problem, where the system is described by the differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad \text{with } \mathbf{x}(0) \text{ given}, \tag{1}$$

where $\mathbf{x}$ is an n-dimensional state vector and $\mathbf{u}$ is an r-dimensional control vector. The optimal control problem is to determine the control $\mathbf{u}$ in the time interval $0 \le t < t_f$, so that the performance index

$$I = \int_0^{t_f} \phi(\mathbf{x}, \mathbf{u})\, dt \tag{2}$$

is minimized. We consider the case where the final time t_f is given. To carry out the minimization of the performance index in (2) subject to the constraints in (1), we consider the augmented performance index

$$J = \int_0^{t_f} \left[\phi + \mathbf{z}^T \left(\mathbf{f} - \frac{d\mathbf{x}}{dt}\right)\right] dt\,, \tag{3}$$

where the n-dimensional vector of Lagrange multipliers $\mathbf{z}$ is called the adjoint vector. The last term in (3) can be thought of as a penalty function to ensure that the state equation is satisfied throughout the given time interval. We introduce the Hamiltonian

$$H = \phi + \mathbf{z}^T \mathbf{f} \tag{4}$$

and use integration by parts to simplify (3) to

$$J = \int_0^{t_f} \left(H + \frac{d\mathbf{z}^T}{dt}\mathbf{x}\right) dt - \mathbf{z}(t_f)\mathbf{x}(t_f) + \mathbf{z}^T(0)\mathbf{x}(0)\,. \tag{5}$$

The optimal control problem now reduces to the minimization of J.

To minimize J numerically, we assume that we have evaluated J at iteration j by using control policy denoted by $\mathbf{u}^{(j)}$. Now the problem is to determine the control policy $\mathbf{u}^{(j+1)}$ at the next iteration. Since the goal is to minimize J, obviously we want to make the change in J negative and numerically as large as possible. If we let $\delta\mathbf{u} = \mathbf{u}^{(j+1)} - \mathbf{u}^{(j)}$, the corresponding change in J is obtained by using Taylor series expansion up to the

quadratic terms:

$$\delta J = \int_0^{t_f} \left[\left(\left(\frac{\partial H}{\partial \mathbf{x}} \right)^{\mathrm{T}} + \frac{\mathrm{d}\mathbf{z}^{\mathrm{T}}}{\mathrm{d}t} \right) \delta\mathbf{x} + \left(\frac{\partial H}{\partial \mathbf{u}} \right)^{\mathrm{T}} \delta\mathbf{u} \right] \mathrm{d}t + \frac{1}{2} \int_0^{t_f} \left[\delta\mathbf{x}^{\mathrm{T}} \frac{\partial^2 H}{\partial \mathbf{x}^2} \delta\mathbf{x} + 2\delta\mathbf{x}^{\mathrm{T}} \frac{\partial^2 H}{\partial \mathbf{x} \partial \mathbf{u}} \delta\mathbf{u} + \delta\mathbf{u}^{\mathrm{T}} \frac{\partial^2 H}{\partial \mathbf{u}^2} \delta\mathbf{u} \right] \mathrm{d}t - \mathbf{z}^{\mathrm{T}}(t_f) \delta\mathbf{x}(t_f) \,. \tag{6}$$

The necessary condition for minimum of J is that the first integral in (6) should be zero; i. e.,

$$\frac{\mathrm{d}\mathbf{z}}{\mathrm{d}t} = -\frac{\partial H}{\partial \mathbf{x}}, \quad \text{with } \mathbf{z}(t_f) = \mathbf{0} \,. \tag{7}$$

and

$$\frac{\partial H}{\partial \mathbf{u}} = \mathbf{0} \,. \tag{8}$$

In control vector iteration, we relax the necessary condition in (8) and choose $\delta\mathbf{u}$ to make δJ negative and in the limit (8) is satisfied. One approach is to choose

$$\delta\mathbf{u} = -\epsilon \frac{\partial H}{\partial \mathbf{u}} \,, \tag{9}$$

where ϵ is a positive parameter which may vary from iteration to iteration. This method is sometimes called first variation method, since the driving force for the change in the control policy is based only on the first term of the Taylor series expansion. The negative sign in (9) is required to minimize the Hamiltonian, as is required by Pontryagin's maximum principle. Numerous papers have been written on the determination of the stepping parameter ϵ [7].

Second Variation Method

Instead of arbitrarily determining the stepping parameter ϵ, one may solve the accessory minimization problem, where $\delta\mathbf{u}$ is chosen to minimize δJ given by (6) after the requirements for the adjoint are satisfied; i. e., it is required to find $\delta\mathbf{u}$ to minimize δJ given by

$$\delta J = \int_0^{t_f} \left[\left(\frac{\partial H}{\partial \mathbf{u}} \right)^{\mathrm{T}} \delta\mathbf{u} + \frac{1}{2} \delta\mathbf{x}^{\mathrm{T}} \frac{\partial^2 H}{\partial x^2} \delta\mathbf{x} + \delta\mathbf{x}^{\mathrm{T}} \frac{\partial^2 H}{\partial \mathbf{x} \partial \mathbf{u}} \delta\mathbf{u} + \frac{1}{2} \delta\mathbf{u}^{\mathrm{T}} \frac{\partial^2 H}{\partial \mathbf{u}^2} \delta\mathbf{u} \right] \mathrm{d}t \,, \tag{10}$$

subject to the differential equation

$$\frac{\mathrm{d}\delta\mathbf{x}}{\mathrm{d}t} = \left(\frac{\partial \mathbf{f}^{\mathrm{T}}}{\partial \mathbf{x}} \right)^{\mathrm{T}} \delta\mathbf{x} + \left(\frac{\partial \mathbf{f}^{\mathrm{T}}}{\partial \mathbf{u}} \right)^{\mathrm{T}} \delta\mathbf{u}, \quad \text{with } \delta\mathbf{x}(0) = \mathbf{0} \,. \tag{11}$$

The solution to this accessory minimization problem is straightforward, since (11) is linear and the performance index in (10) is almost quadratic, and can be easily done, as shown in ([1], pp. 259–266) and [7]. The resulting equations, to be integrated backwards from $t = t_f$ to $t = 0$ with zero starting conditions, are

$$\frac{\mathrm{d}\mathbf{J}}{\mathrm{d}t} + \mathbf{J} \left(\frac{\partial \mathbf{f}^{\mathrm{T}}}{\partial \mathbf{x}} \right)^{\mathrm{T}} + \frac{\partial \mathbf{f}^{\mathrm{T}}}{\partial \mathbf{x}} \mathbf{J} + \frac{\partial^2 H}{\partial \mathbf{x}^2} - \mathbf{S}^{\mathrm{T}} \left(\frac{\partial^2 H}{\partial \mathbf{u}^2} \right)^{-1} \mathbf{S} = \mathbf{0} \,, \tag{12}$$

where the $(r \times n)$-matrix $\mathbf{S} = \partial^2 H / \partial\mathbf{u}\partial\mathbf{x} + \partial\mathbf{f}^T / \partial\mathbf{x}\mathbf{J}$, and

$$\frac{\mathrm{d}\mathbf{g}}{\mathrm{d}t} - \mathbf{S}^{\mathrm{T}} \left(\frac{\partial^2 H}{\partial \mathbf{u}^2} \right)^{-1} \left(\frac{\partial H}{\partial \mathbf{u}} + \frac{\partial \mathbf{f}^{\mathrm{T}}}{\partial \mathbf{x}} \mathbf{g} \right) + \left(\frac{\partial \mathbf{f}^{\mathrm{T}}}{\partial \mathbf{x}} \right) \mathbf{g} = \mathbf{0} \,. \tag{13}$$

The control policy is then updated through the equation

$$\mathbf{u}^{(j+1)} = \mathbf{u}^{(j)} - \left(\frac{\partial^2 H}{\partial \mathbf{u}^2} \right)^{-1} \left(\frac{\partial H}{\partial \mathbf{u}} + \frac{\partial \mathbf{f}^{\mathrm{T}}}{\partial \mathbf{x}} \mathbf{g} \right) - \left(\frac{\partial^2 H}{\partial \mathbf{u}^2} \right)^{-1} \mathbf{S} \left(\mathbf{x}^{(j+1)} - \mathbf{x}^{(j)} \right) \,. \tag{14}$$

This method of updating the control policy is called the second variation method. In (12) the $(n \times n)$-matrix $\mathbf{J}$ is symmetric, so the total number of differential equations to be integrated backwards is $n(n+1)/2 + 2n$. However, the convergence is quadratic if the initial control policy is close to the optimum.

To obtain good starting conditions, Luus and Lapidus [6] suggested the use of first variation method for the first few iterations and then to switch over to the second variation method.

One additional feature of the second variation method is that the control policy given in (14) is a function of the present state, so that the control policy is treated as being continuous and is not restricted to be-

ing piecewise constant over an integration time step, as is the case with the first variation method. As was shown in ([1], pp. 316–317), for the linear six-plate gas absorber example, when the system equation is linear and the performance index is quadratic, the second variation method yields the optimal control policy in a single step.

Determination of Stepping Parameter

However, the large number and complexity of equations required for obtaining the control policy and the instability of the method for very complex systems led to investigating different means of obtaining faster convergence with the first variation method. The effort was directed on the best means of obtaining the stepping parameter ϵ in (9). When ϵ is too large, overstepping occurs, and if ϵ is too small, the convergence rate is very small.

Numerous papers have been written on the determination of ϵ. Several methods were compared by Rao and Luus [8] in solving typical optimal control problems. Although they suggested a means of determining the 'best' method for performance indices that are almost quadratic, it is found that a very simple scheme is quite effective for a wide variety of optimal control problems. Instead of trying to get very fast convergence and risk instability, the emphasis is placed on the robustness. The strategy is to obtain the initial value for ϵ from the magnitude of $\partial H/\partial \mathbf{u}$, and then increasing ϵ when the iteration has been successful, and reducing its value if overstepping occurs. This type of approach was used in [2] in solving the optimal control of a pyrolysis problem. When the iteration was successful, the stepping parameter was increased by 10 percent, and when overstepping resulted, the stepping parameter was reduced to half its value. The algorithm for first variation method may be presented as follows:

- Choose an initial control policy $\mathbf{u}^{(0)}$ and a value for ϵ; set the iteration index j to 0.
- Integrate (1) from $t = 0$ to $t = t_f$ and evaluate the performance index in (2). Store the values of the state vector at the end of each integration time step.
- Integrate the adjoint equation (7) from $t = t_f$ to $t = 0$, using for $\mathbf{x}$ the stored values of the state vector in Step 2. At each integration time step evaluate the gradient $\partial H/\partial \mathbf{u}$.
- Choose a new control policy

$$\mathbf{u}^{(j+1)} = \mathbf{u}^{(j)} - \epsilon \frac{\partial H}{\partial \mathbf{u}} \,. \tag{15}$$

- Integrate (1) from $t = 0$ to $t = t_f$ and evaluate the performance index in (2). Store the values of the state vector at the end of each integration time step. If the performance index is worse (i. e., overstepping has occurred), reduce ϵ to half its value and go to Step 4. If the performance index has been improved increase ϵ by a small factor, such as 1.10 and go to Step 3, and continue for a number of iterations, or terminate the iterations when the change in the performance index in an iteration is less than some criterion, and interpret the results.

Illustration of the First Variation Method

Let us consider the nonlinear continuous stirred tank reactor that has been used for optimal control studies in ([1], pp. 308–318) and [6], and which was shown in [4] to exhibit multiplicity of solutions. The system is described by the two equations

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = -2\,(x_1 + 0.25) + (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right) - u\,(x_1 + 0.25) \,, \tag{16}$$

$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = 0.5 - x_2 - (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right) \,, \tag{17}$$

with the initial state $x_1(0) = 0.09$ and $x_2(0) = 0.09$. The control u is a scalar quantity related to the valve opening of the coolant. The state variables x_1 and x_2 represent deviations from the steady state of dimensionless temperature and concentration, respectively. The performance index to be minimized is

$$I = \int_0^{t_f} \left(x_1^2 + x_2^2 + 0.1u^2\right) \mathrm{d}t \,, \tag{18}$$

where the final time $t_f = 0.78$. The Hamiltonian is

$$H = z_1\,(-2\,(x_1 + 0.25) + R - u\,(x_1 + 0.25)) + z_2\,(0.5 - x_2 - R) + x_1^2 + x_2^2 + 0.1u^2 \,, \tag{19}$$

where $R = (x_2 + 0.5)\exp(25x_1/(x_1 + 2))$. The adjoint equations are

$$\frac{dz_1}{dt} = (u + 2)\, z_1 - 2x_1 + 50R\frac{(z_2 - z_1)}{(x_1 + 2)^2}\,, \qquad (20)$$

$$\frac{dz_2}{dt} = -2x_2 + \frac{(z_2 - z_1)}{(x_2 + 0.5)}R + z_2\,, \qquad (21)$$

and the gradient of the Hamiltonian is

$$\frac{\partial H}{\partial \mathbf{u}} = 0.2u - (x_1 + 0.25)\, z_1\,. \qquad (22)$$

To illustrate the computational aspects of CVI, the above algorithm was used with a Pentium-120 personal computer using WATCOM Fortran compiler version 9.5. The calculations were done in double precision. As found in [4], convergence to the local optimum was obtained when small values for the initial control policy were used, and the global optimum was obtained when large values were used as initial policy. As is seen in Table 1, when an integration time step of 0.0065 was used (allowing 120 piecewise constant steps), in spite of the large number of iterations, the optimal control policy can be obtained in less than 2 s of computer time. The iterations were stopped when the change in the performance index from iteration to iteration was less than 10^{-6}.

The total computation time for making this run with 11 different initial control policies was 9.6 s on the Pentium-120 digital computer. When an integration time step of 0.00312 was used, the value of the performance index at the global optimum was improved to 0.133104. When a time step of 0.001 was used, giving 780 time steps, the optimal control policy yielded $I = 0.133097$. Even here the computation time for the 11 different initial conditions was only 31 s. With the use of piecewise linear control and only 20 time stages, a performance index of $I = 0.133101$ was obtained in [3] with iterative dynamic programming (IDP). To obtain this result with IDP, by using 5 randomly chosen points and 10 passes, each consisting of 20 iterations, took 13.4 s on a Pentium-120. The use of 15 time stages yielded $I = 0.133112$ and required 7.8 s. Therefore, computationally CVI is faster than IDP for this problem, but the present formulation does not allow piecewise linear control to be used in CVI.

Control Vector Iteration CVI, Table 1
Application of First Variation Method to CSTR

Initial policy $u^{(0)}$	Performance index	Number of iterations	CPU time s
1.0	0.244436	16	0.16
1.2	0.244436	17	0.17
1.4	0.244436	18	0.11
1.6	0.244436	18	0.16
1.8	0.244436	19	0.22
2.0	0.133128	143	1.49
2.2	0.133128	149	1.53
2.4	0.133128	149	1.54
2.6	0.133130	133	1.43
2.8	0.133129	142	1.37
3.0	0.133130	136	1.38

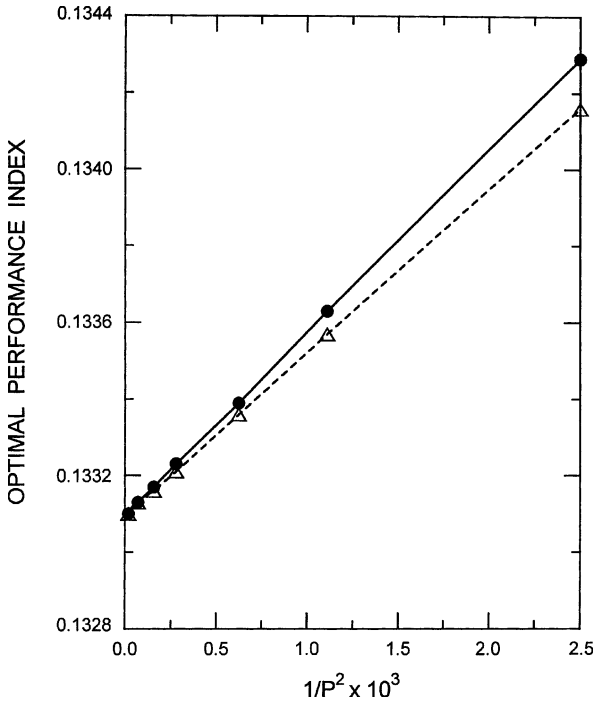
Control Vector Iteration CVI, Table 2
Effect of the number of time stages P on the optimal performance index

Number of time stages P	Optimal I by CVI	Optimal I by IDP
20	0.13429	0.13416
30	0.13363	0.13357
40	0.13339	0.13336
60	0.13323	0.13321
80	0.13317	0.13316
120	0.13313	0.13313
240	0.13310	0.13310

The effect of the number of time stages for piecewise constant control is shown in Table 2, where CVI results are compared to those obtained by IDP in [5].

As can be seen, the given algorithm gives results very close to those obtained by IDP, and the deviations decrease as the number of time stages increases, because the approximations introduced during the backward integration when the stored values for the state vector are used, and in the calculation of the gradient of the Hamiltonian in CVI become negligible as the time stages become very small. As is shown in Fig. 1, when the optimal value of the performance index is plotted against $1/P^2$, the extrapolated value, as $1/P^2$ approaches zero, gives the value obtained with the second variation method.

The first variation method is easy to program and will continue to be a very useful method of determining the optimal control of nonlinear systems.



Control Vector Iteration CVI, Figure 1
Linear variation of optimal performance index with P^{-2}; •–•–• CVI, Δ -- Δ -- Δ IDP

See also

- ▶ Boundary Condition Iteration BCI
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Optimal Control of a Flexible Arm
- ▶ Optimization Strategies for Dynamic Systems
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Lapidus L, Luus R (1967) Optimal control of engineering processes. Blaisdell, Waltham
2. Luus R (1978) On the optimization of oil shale pyrolysis. Chem Eng Sci 33:1403–1404
3. Luus R (1993) Application of iterative dynamic programming to very high dimensional systems. Hungarian J Industr Chem 21: 243–250
4. Luus R, Cormack DE (1972) Multiplicity of solutions resulting from the use of variational methods in optimal control problems. Canad J Chem Eng 50:309–311
5. Luus R, Galli M (1991) Multiplicity of solutions in using dynamic programming for optimal control. Hungarian J Industr Chem 19:55–62
6. Luus R, Lapidus L (1967) The control of nonlinear systems. Part II: Convergence by combined first and second variations. AIChE J 13:108–113
7. Merriam CW (1964) Optimization theory and the design of feedback control systems, McGraw-Hill, New York, pp 259–261
8. Rao SN, Luus R (1972) Evaluation and improvement of control vector iteration procedures for optimal control. Canad J Chem Eng 50:777–784

Convex Discrete Optimization

SHMUEL ONN
Technion – Israel Institute of Technology, Haifa, Israel

MSC2000: 05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C

Article Outline

Abstract
Introduction
  Limitations
  Outline and Overview of Main Results and Applications
  Terminology and Complexity
Reducing Convex to Linear Discrete Optimization
  Edge-Directions and Zonotopes
  Strongly Polynomial Reduction of Convex to Linear Discrete Optimization
  Pseudo Polynomial Reduction when Edge-Directions Are not Available

Convex Combinatorial Optimization and More
From Membership to Linear Optimization
Linear and Convex Combinatorial Optimization in Strongly Polynomial Time
Linear and Convex Discrete Optimization over any Set in Pseudo Polynomial Time
Some Applications
Linear N-fold Integer Programming
Oriented Augmentation and Linear Optimization
Graver Bases and Linear Integer Programming
Graver Bases of N-fold Matrices
Linear N-fold Integer Programming in Polynomial Time
Some Applications
Convex Integer Programming
Convex Integer Programming over Totally Unimodular Systems
Graver Bases and Convex Integer Programming
Convex N-fold Integer Programming in Polynomial Time
Some Applications
Multiway Transportation Problems and Privacy in Statistical Databases
Multiway Transportation Problems and Privacy in Statistical Databases
The Universality Theorem
The Complexity of the Multiway Transportation Problem
Privacy and Entry-Uniqueness
References

Abstract

We develop an algorithmic theory of convex optimization over discrete sets. Using a combination of algebraic and geometric tools we are able to provide polynomial time algorithms for solving broad classes of convex combinatorial optimization problems and convex integer programming problems in variable dimension. We discuss some of the many applications of this theory including to quadratic programming, matroids, bin packing and cutting-stock problems, vector partitioning and clustering, multiway transportation problems, and privacy and confidential statistical data disclosure. Highlights of our work include a strongly polynomial time algorithm for convex and linear combinatorial optimization over any family presented by a membership oracle when the underlying polytope has few edge-directions; a new theory of so-termed n-fold integer programming, yielding polynomial time solution of important and natural classes of convex and linear integer programming problems in variable dimension; and a complete complexity classification of high dimensional transportation problems, with practical applications to fundamental problems in privacy and confidential statistical data disclosure.

Introduction

The general linear discrete optimization problem can be posed as follows.

LINEAR DISCRETE OPTIMIZATION. Given a set $S \subseteq \mathbb{Z}^n$ of integer points and an integer vector $w \in \mathbb{Z}^n$, find an $x \in S$ maximizing the standard inner product $wx := \sum_{i=1}^{n} w_i x_i$.

The algorithmic complexity of this problem, which includes *integer programming* and *combinatorial optimization* as special cases, depends on the presentation of the set S of feasible points. In integer programming, this set is presented as the set of integer points satisfying a given system of linear inequalities, which in standard form is given by

$$S \quad = \quad \{x \in \mathbb{N}^n : Ax = b\} ,$$

where $\mathbb{N}$ stands for the nonnegative integers, $A \in \mathbb{Z}^{m \times n}$ is an $m \times n$ integer matrix, and $b \in \mathbb{Z}^m$ is an integer vector. The input for the problem then consists of A, b, w. In combinatorial optimization, $S \subseteq \{0,1\}^n$ is a set of $\{0,1\}$-vectors, often interpreted as a family of subsets of a ground set $N := \{1, \ldots, n\}$, where each $x \in S$ is the indicator of its support $\mathrm{supp(x)} \subseteq \mathrm{N}$. The set S is presented implicitly and compactly, say as the set of indicators of subsets of edges in a graph G satisfying a given combinatorial property (such as being a matching, a forest, and so on), in which case the input is G, w. Alternatively, S is given by an oracle, such as a *membership oracle* which, queried on $x \in \{0,1\}^n$, asserts whether or not $x \in S$, in which case the algorithmic complexity also includes a count of the number of oracle queries needed to solve the problem.

Here we study the following broad generalization of linear discrete optimization.

CONVEX DISCRETE OPTIMIZATION. Given a set $S \subseteq \mathbb{Z}^n$, vectors $w_1, \ldots, w_d \in \mathbb{Z}^n$, and a convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$, find an $x \in S$ maximizing $c(w_1 x, \ldots, w_d x)$.

This problem can be interpreted as *multi-objective* linear discrete optimization: given d linear functionals $w_1 x, \ldots, w_d x$ representing the values of points $x \in S$ under d criteria, the goal is to maximize their "con-

vex balancing" defined by $c(w_1x, \ldots, w_dx)$. In fact, we have a hierarchy of problems of increasing generality and complexity, parameterized by the number d of linear functionals: at the bottom lies the linear discrete optimization problem, recovered as the special case of $d = 1$ and c the identity on $\mathbb{R}$; and at the top lies the problem of maximizing an arbitrary convex functional over the feasible set S, arising with $d = n$ and with $w_i = \mathbf{1}_i$ the ith standard unit vector in $\mathbb{R}^n$ for all i.

The algorithmic complexity of the convex discrete optimization problem depends on the presentation of the set S of feasible points as in the linear case, as well as on the presentation of the convex functional c. When S is presented as the set of integer points satisfying a given system of linear inequalities we also refer to the problem as *convex integer programming*, and when $S \subseteq \{0,1\}^n$ and is presented implicitly or by an oracle we also refer to the problem as *convex combinatorial optimization*. As for the convex functional c, we will assume throughout that it is presented by a *comparison oracle* that, queried on $x, y \in \mathbb{R}^d$, asserts whether or not $c(x) \leq c(y)$. This is a very broad presentation that reveals little information on the function, making the problem, on the one hand, very expressive and applicable, but on the other hand, very hard to solve.

There is a massive body of knowledge on the complexity of linear discrete optimization – in particular (linear) integer programming [55] and (linear) combinatorial optimization [31]. The purpose of this monograph is to provide the first comprehensive unified treatment of the extended convex discrete optimization problem. The monograph follows the outline of five lectures given by the author in the Séminaire de Mathématiques Supérieures Series, Université de Montréal, during June 2006. Colorful slides of theses lectures are available online at [46] and can be used as a visual supplement to this monograph. The monograph has been written under the support of the ISF – Israel Science Foundation. The theory developed here is based on and is a culmination of several recent papers including [5,12,13,14,15,16,17,25,39,47,48,49,50,51] written in collaboration with several colleagues – Eric Babson, Jesus De Loera, Komei Fukuda, Raymond Hemmecke, Frank Hwang, Vera Rosta, Uriel Rothblum, Leonard Schulman, Bernd Sturmfels, Rekha Thomas, and Robert Weismantel. By developing and using a combination of geometric and algebraic tools, we are able to provide polynomial time algorithms for several broad classes of convex discrete optimization problems. We also discuss in detail some of the many applications of our theory, including to quadratic programming, matroids, bin packing and cutting-stock problems, vector partitioning and clustering, multiway transportation problems, and privacy and confidential statistical data disclosure.

We hope that this monograph will, on the one hand, allow users of discrete optimization to enjoy the new powerful modelling and expressive capability of convex discrete optimization along with its broad polynomial time solvability, and on the other hand, stimulate more research on this new and fascinating class of problems, their complexity, and the study of various relaxations, bounds, and approximations for such problems.

Limitations

Convex discrete optimization is generally intractable even for small fixed d, since already for $d = 1$ it includes linear integer programming which is NP-hard. When d is a variable part of the input, even very simple special cases are NP-hard, such as the following problem, so-called *positive semi-definite quadratic binary programming*,

$$\max\{(w_1x)^2 + \cdots + (w_nx)^2 \ : \ x \in \mathbb{N}^n \, , \quad x_i \leq 1 \, , \quad i = 1, \ldots, n\} \, .$$

Therefore, throughout this monograph we will assume that d is fixed (but arbitrary).

As explained above, we also assume throughout that the convex functional c which constitutes part of the data for the convex discrete optimization problem is presented by a comparison oracle. Under such broad presentation, the problem is generally very hard. In particular, if the feasible set is $S := \{x \in \mathbb{N}^n : Ax = b\}$ and the underlying polyhedron $P := \{x \in \mathbb{R}^n_+ : Ax = b\}$ is *unbounded*, then the problem is inaccessible even in one variable with no equation constraints. Indeed, consider the following family of univariate convex integer programs with convex functions parameterized by $-\infty < u \leq \infty$,

$$\max\{c_u(x) \ : \ x \in \mathbb{N}\} \, ,$$

$$c_u(x) := \begin{cases} -x, & \text{if } x < u; \\ x - 2u, & \text{if } x \geq u. \end{cases} \, .$$

Consider any algorithm attempting to solve the problem and let u be the maximum value of x in all queries to the oracle of c. Then the algorithm can not distinguish between the problem with c_u, whose objective function is unbounded, and the problem with c_∞, whose optimal objective value is 0. Thus, convex discrete optimization (with an oracle presented functional) over an *infinite* set $S \subset \mathbb{Z}^n$ is quite hopeless. Therefore, an algorithm that solves the convex discrete optimization problem will either return an optimal solution, or assert that the problem is infeasible, or assert that the underlying polyhedron is unbounded. In fact, in most applications, such as in combinatorial optimization with $S \subseteq \{0,1\}^n$ or integer programming with $S := \{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\}$ and $l, u \in \mathbb{Z}^n$, the set S is finite and the problem of unboundedness does not arise.

Outline and Overview of Main Results and Applications

We now outline the structure of this monograph and provide a brief overview of what we consider to be our main results and main applications. The precise relevant definitions and statements of the theorems and corollaries mentioned here are provided in the relevant sections in the monograph body. As mentioned above, most of these results are adaptations or extensions of results from one of the papers [5,12,13,14,15,16,17,25,39,47,48,49,50,51]. The monograph gives many more applications and results that may turn out to be useful in future development of the theory of convex discrete optimization.

The rest of the monograph consists of five sections. While the results evolve from one section to the next, it is quite easy to read the sections independently of each other (while just browsing now and then for relevant definitions and results). Specifically, Sect. "Convex Combinatorial Optimization and More" uses definitions and the main result of Sect. "Reducing Convex to Linear Discrete Optimization"; Sect. "Convex Integer Programming" uses definitions and results from Sections "Reducing Convex to Linear Discrete Optimization" and "Linear N-fold Integer Programming"; and Sect. "Multiway Transportation Problems and Privacy in Statistical Databases" uses the main results of Sections "Linear N-fold Integer Programming" and "Convex Integer Programming".

In Sect. "Reducing Convex to Linear Discrete Optimization" we show how to reduce the convex discrete optimization problem over $S \subset \mathbb{Z}^n$ to strongly polynomially many linear discrete optimization counterparts over S, provided that the convex hull $\mathrm{conv}(S)$ satisfies a suitable geometric condition, as follows.

Theorem 1 *For every fixed d, the convex discrete optimization problem over any finite $S \subset \mathbb{Z}^n$ presented by a linear discrete optimization oracle and endowed with a set covering all edge-directions of* $\mathrm{conv}(S)$*, can be solved in strongly polynomial time.*

This result will be incorporated in the polynomial time algorithms for convex combinatorial optimization and convex integer programming to be developed in Sect. "Convex Combinatorial Optimization and More" and Sect. "Convex Integer Programming".

In Sect. "Convex Combinatorial Optimization and More" we discuss convex combinatorial optimization. The main result is that convex combinatorial optimization over a set $S \subseteq \{0,1\}^n$ presented by a membership oracle can be solved in strongly polynomial time provided it is endowed with a set covering all edge-directions of $\mathrm{conv}(S)$. In particular, the standard linear combinatorial optimization problem over S can be solved in strongly polynomial time as well.

Theorem 2 *For every fixed d, the convex combinatorial optimization problem over any $S \subseteq \{0,1\}^n$ presented by a membership oracle and endowed with a set covering all edge-directions of the polytope* $\mathrm{conv}(S)$*, can be solved in strongly polynomial time.*

An important application of Theorem 2 concerns convex matroid optimization.

Corollary 1 *For every fixed d, convex combinatorial optimization over the family of bases of a matroid presented by membership oracle is strongly polynomial time solvable.*

In Sect. "Linear N-fold Integer Programming" we develop the theory of linear *n-fold integer programming*. As a consequence of this theory we are able to solve a broad class of linear integer programming problems in variable dimension in polynomial time, in contrast with the general intractability of linear integer programming. The main theorem here may seem a bit technical at a first glance, but is

really very natural and has many applications discussed in detail in Sect. "Linear N-fold Integer Programming", Sect. "Convex Integer Programming" and Sect. "Multiway Transportation Problems and Privacy in Statistical Databases". To state it we need a definition. Given an $(r+s)\times t$ matrix A, let A_1 be its $r\times t$ sub-matrix consisting of the first r rows and let A_2 be its $s\times t$ sub-matrix consisting of the last s rows. We refer to A explicitly as $(r+s)\times t$ *matrix*, since the definition below depends also on r and s and not only on the entries of A. The *n-fold matrix* of an $(r+s)\times t$ matrix A is then defined to be the following $(r+ns)\times nt$ matrix,

$$A^{(n)} := (\mathbf{1}_n \otimes A_1) \oplus (I_n \otimes A_2) = \begin{pmatrix} A_1 & A_1 & A_1 & \cdots & A_1 \\ A_2 & 0 & 0 & \cdots & 0 \\ 0 & A_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & A_2 \end{pmatrix}.$$

Given now any $n \in \mathbb{N}$, lower and upper bounds $l, u \in \mathbb{Z}_\infty^{nt}$ with $\mathbb{Z}_\infty := \mathbb{Z} \uplus \{\pm\infty\}$, right-hand side $b \in \mathbb{Z}^{r+ns}$, and linear functional wx with $w \in \mathbb{Z}^{nt}$, the corresponding linear n-fold integer programming problem is the following program in variable dimension nt,

$$\max\{wx \,:\, x \in \mathbb{Z}^{nt},\ A^{(n)}x = b,\ l \le x \le u\}.$$

The main theorem of Sect. "Linear N-fold Integer Programming" asserts that such integer programs are polynomial time solvable.

Theorem 3 *For every fixed $(r+s)\times t$ integer matrix A, the linear n-fold integer programming problem with any n, l, u, b, and w can be solved in polynomial time.*

Theorem 3 has very important applications to high-dimensional transportation problems which are discussed in Sect. "Three-Way Line-Sum Transportation Problems" and in more detail in Sect. "Multiway Transportation Problems and Privacy in Statistical Databases". Another major application concerns bin packing problems, where items of several types are to be packed into bins so as to maximize packing utility subject to weight constraints. This includes as a special case the classical cutting-stock problem of [27]. These are discussed in detail in Sect. "Packing Problems and Cutting-Stock".

Corollary 2 *For every fixed number t of types and type weights $v_1, \ldots, v_t$, the corresponding integer bin packing and cutting-stock problems are polynomial time solvable.*

In Sect. "Convex Integer Programming" we discuss convex integer programming, where the feasible set S is presented as the set of integer points satisfying a given system of linear inequalities. In particular, we consider convex integer programming over n-fold systems for any fixed (but arbitrary) $(r+s)\times t$ matrix A, where, given $n \in \mathbb{N}$, vectors $l, u \in \mathbb{Z}_\infty^{nt}$, $b \in \mathbb{Z}^{r+ns}$ and $w_1, \ldots, w_d \in \mathbb{Z}^{nt}$, and convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$, the problem is

$$\max\{c(w_1x, \ldots, w_dx) \,:\, x \in \mathbb{Z}^{nt},\ A^{(n)}x = b,\ l \le x \le u\}.$$

The main theorem of Sect. "Convex Integer Programming" is the following extension of Theorem 3, asserting that convex integer programming over n-fold systems is polynomial time solvable as well.

Theorem 4 *For every fixed d and $(r+s)\times t$ integer matrix A, convex n-fold integer programming with any n, l, u, b, $w_1, \ldots, w_d$, and c can be solved in polynomial time.*

Theorem 4 broadly extends the class of objective functions that can be efficiently maximized over n-fold systems. Thus, all applications discussed in Sect. "Some Applications" automatically extend accordingly. These include convex high-dimensional transportation problems and convex bin packing and cutting-stock problems, which are discussed in detail in Sect. "Transportation Problems and Packing Problems" and Sect. "Multiway Transportation Problems and Privacy in Statistical Databases".

Another important application of Theorem 4 concerns vector partitioning problems which have applications in many areas including load balancing, circuit layout, ranking, cluster analysis, inventory, and reliability, see e. g. [7,9,25,39,50] and the references therein. The problem is to partition n items among p players so as to maximize social utility. With each item is associated a k-dimensional vector representing its utility under k criteria. The social utility of a partition is a convex function of the sums of vectors of items that each

player receives. In the constrained version of the problem, there are also restrictions on the number of items each player can receive. We have the following consequence of Theorem 4; more details on this application are in Sect. "Vector Partitioning and Clustering".

Corollary 3 *For every fixed number p of players and number k of criteria, the constrained and unconstrained vector partition problems with any item vectors, convex utility, and constraints on the number of item per player, are polynomial time solvable.*

In the last Sect. "Multiway Transportation Problems and Privacy in Statistical Databases" we discuss multiway (high-dimensional) transportation problems and secure statistical data disclosure. Multiway transportation problems form a very important class of discrete optimization problems and have been used and studied extensively in the operations research and mathematical programming literature, as well as in the statistics literature in the context of secure statistical data disclosure and management by public agencies, see e. g. [4,6,11,18,19,42,43,53,60,62] and the references therein. The feasible points in a transportation problem are the multiway tables ("contingency tables" in statistics) such that the sums of entries over some of their lower dimensional sub-tables such as lines or planes ("margins" in statistics) are specified. We completely settle the algorithmic complexity of treating multiway tables and discuss the applications to transportation problems and secure statistical data disclosure, as follows.

In Sect. "The Universality Theorem" we show that "short" 3-way transportation problems, over $r \times c \times 3$ tables with variable number r of rows and variable number c of columns but fixed small number 3 of layers (hence "short"), are *universal* in that *every* integer programming problem is such a problem (see Sect. "The Universality Theorem" for the precise stronger statement and for more details).

Theorem 5 *Every linear integer programming problem* $\max\{cy : y \in \mathbb{N}^n : Ay = b\}$ *is polynomial time representable as a short 3-way line-sum transportation problem*

$$\max\left\{ wx \,:\, x \in \mathbb{N}^{r\times c\times 3} \,:\, \sum_i x_{i,j,k} = z_{j,k}\,,\ \sum_j x_{i,j,k} = v_{i,k}\,,\ \sum_k x_{i,j,k} = u_{i,j} \right\}.$$

In Sect. "The Complexity of the Multiway Transportation Problem" we discuss k-way transportation problems of any dimension k. We provide the first polynomial time algorithm for convex and linear "long" $(k+1)$-way transportation problems, over $m_1 \times \cdots \times m_k \times n$ tables, with k and $m_1, \ldots, m_k$ fixed (but arbitrary), and variable number n of layers (hence "long"). This is best possible in view of Theorem 21. Our algorithm works for any *hierarchical collection of margins*: this captures common margin collections such as all line-sums, all plane-sums, and more generally all h-flat sums for any $0 \le h \le k$ (see Sect. "Tables and Margins" for more details). We point out that even for the very special case of linear integer transportation over $3 \times 3 \times n$ tables with specified line-sums, our polynomial time algorithm is the only one known. We prove the following statement.

Corollary 4 *For every fixed $d, k, m_1, \ldots, m_k$ and family $\mathcal{F}$ of subsets of $\{1, \ldots, k+1\}$ specifying a hierarchical collection of margins, the convex (and in particular linear) long transportation problem over $m_1 \times \cdots \times m_k \times n$ tables is polynomial time solvable.*

In our last subsection Sect. "Privacy and Entry-Uniqueness" we discuss an important application concerning privacy in statistical databases. It is a common practice in the disclosure of a multiway table containing sensitive data to release some table margins rather than the table itself. Once the margins are released, the security of any specific entry of the table is related to the set of possible values that can occur in that entry in any table having the same margins as those of the source table in the data base. In particular, if this set consists of a unique value, that of the source table, then this entry can be exposed and security can be violated. We show that for multiway tables where one category is significantly richer than the others, that is, when each sample point can take many values in one category and only few values in the other categories, it is possible to check entry-uniqueness in polynomial time, allowing disclosing agencies to make learned decisions on secure disclosure.

Corollary 5 *For every fixed $k, m_1, \ldots, m_k$ and family $\mathcal{F}$ of subsets of $\{1, \ldots, k+1\}$ specifying a hierarchical collection of margins to be disclosed, it can be decided in polynomial time whether any specified entry $x_{i_1,\ldots,i_{k+1}}$ is the same in all long $m_1 \times \cdots \times m_k \times n$ tables with the disclosed margins, and hence at risk of exposure.*

Terminology and Complexity

We use $\mathbb{R}$ for the reals, $\mathbb{R}_+$ for the nonnegative reals, $\mathbb{Z}$ for the integers, and $\mathbb{N}$ for the nonnegative integers. The sign of a real number r is denoted by $\text{sign}(r) \in \{0, -1, 1\}$ and its absolute value is denoted by $|r|$. The ith standard unit vector in $\mathbb{R}^n$ is denoted by $\mathbf{1}_i$. The *support* of $x \in \mathbb{R}^n$ is the index set $\text{supp(x)} := \{\text{i} : \text{x}_\text{i} \neq 0\}$ of nonzero entries of x. The *indicator* of a subset $I \subseteq \{1, \ldots, n\}$ is the vector $\mathbf{1}_I := \sum_{i \in I} \mathbf{1}_i$ so that $\text{supp}(\mathbf{1}_\text{I}) = \text{I}$. When several vectors are indexed by subscripts, $w_1, \ldots, w_d \in \mathbb{R}^n$, their entries are indicated by pairs of subscripts, $w_i = (w_{i,1}, \ldots, w_{i,n})$. When vectors are indexed by superscripts, $x^1, \ldots, x^k \in \mathbb{R}^n$, their entries are indicated by subscripts, $x^i = (x^i_1, \ldots, x^i_n)$. The integer lattice $\mathbb{Z}^n$ is naturally embedded in $\mathbb{R}^n$. The space $\mathbb{R}^n$ is endowed with the standard inner product which, for $w, x \in \mathbb{R}^n$, is given by $wx := \sum_{i=1}^n w_i x_i$. Vectors w in $\mathbb{R}^n$ will also be regarded as linear functionals on $\mathbb{R}^n$ via the inner product wx. Thus, we refer to elements of $\mathbb{R}^n$ as points, vectors, or linear functionals, as will be appropriate from the context. The *convex hull* of a set $S \subseteq \mathbb{R}^n$ is denoted by $\text{conv}(S)$ and the set of *vertices* of a polyhedron $P \subseteq \mathbb{R}^n$ is denoted by $\text{vert}(P)$. In linear discrete optimization over $S \subseteq \mathbb{Z}^n$, the *facets* of $\text{conv}(S)$ play an important role, see Chvátal [10] and the references therein for earlier work, and Grötschel, Lovász and Schrijver [31,45] for the later culmination in the equivalence of separation and linear optimization via the ellipsoid method of Yudin and Nemirovskii [63]. As will turn out in Sect. "Reducing Convex to Linear Discrete Optimization", in convex discrete optimization over S, the *edges* of $\text{conv}(S)$ play an important role (most significantly in a way which is *not* related to the Hirsch conjecture discussed in [41]). We therefore use extensively convex polytopes, for which we follow the terminology of [32,65].

We often assume that the feasible set $S \subseteq \mathbb{Z}^n$ is finite. We then define its *radius* to be its l_∞ radius $\rho(S) := \max\{\|x\|_\infty : x \in S\}$ where, as usual, $\|x\|_\infty := \max_{i=1}^n |x_i|$. In other words, $\rho(S)$ is the smallest $\rho \in \mathbb{N}$ such that S is contained in the cube $[-\rho, \rho]^n$.

Our algorithms are applied to rational data only, and the time complexity is as in the standard Turing machine model, see e. g. [1,26,55]. The input typically consists of rational (usually integer) numbers, vectors, matrices, and finite sets of such objects. The *binary length* of an integer number $z \in \mathbb{Z}$ is defined to be the number of bits in its binary representation, $\langle z \rangle := 1 + \lceil \log_2(|z| + 1) \rceil$ (with the extra bit for the sign). The length of a rational number presented as a fraction $r = \frac{p}{q}$ with $p, q \in \mathbb{Z}$ is $\langle r \rangle := \langle p \rangle + \langle q \rangle$. The length of an $m \times n$ matrix A (and in particular of a vector) is the sum $\langle A \rangle := \sum_{i,j} \langle a_{i,j} \rangle$ of the lengths of its entries. Note that the length of A is no smaller than the number of entries, $\langle A \rangle \geq mn$. Therefore, when A is, say, part of an input to an algorithm, with m, n variable, the length $\langle A \rangle$ already incorporates mn, and so we will typically not account additionally for m, n directly. But sometimes, especially in results related to n-fold integer programming, we will also emphasize n as part of the input length. Similarly, the length of a finite set E of numbers, vectors or matrices is the sum of lengths of its elements and hence, since $\langle E \rangle \geq |E|$, automatically accounts for its cardinality.

Some input numbers affect the running time of some algorithms through their unary presentation, resulting in so-called "pseudo polynomial" running time. The *unary length* of an integer number $z \in \mathbb{Z}$ is the number $|z|+1$ of bits in its unary representation (again, an extra bit for the sign). The unary length of a rational number, vector, matrix, or finite set of such objects are defined again as the sums of lengths of their numerical constituents, and is again no smaller than the number of such numerical constituents.

When studying convex and linear integer programming in Sect. "Linear N-fold Integer Programming" and Sect. "Convex Integer Programming" we sometimes have lower and upper bound vectors l, u with entries in $\mathbb{Z}_\infty := \mathbb{Z} \uplus \{\pm\infty\}$. Both binary and unary lengths of a $\pm\infty$ entry are constant, say 3 by encoding $\pm\infty := \pm$"00".

To make the input encoding precise, we introduce the following notation. In every algorithmic statement we describe explicitly the input encoding, by listing in square brackets all input objects affecting the running time. Unary encoded objects are listed directly whereas binary encoded objects are listed in terms of their length. For example, as is often the case, if the input of an algorithm consists of binary encoded vectors (linear functionals) $w_1, \ldots, w_d \in \mathbb{Z}^n$ and unary encoded integer $\rho \in \mathbb{N}$ (bounding the radius $\rho(S)$ of the feasible set) then we will indicate that the input is *encoded as* $[\rho, \langle w_1, \ldots, w_d \rangle]$.

Some of our algorithms are strongly polynomial time in the sense of [59]. For this, part of the input is regarded as "special". An algorithm is then *strongly polynomial time* if it is polynomial time in the usual Turing sense with respect to all input, and in addition, the number of arithmetic operations (additions, subtractions, multiplications, divisions, and comparisons) it performs is polynomial in the special part of the input. To make this precise, we extend our input encoding notation above by splitting the square bracketed expression indicating the input encoding into a "left" side and a "right" side, separated by semicolon, where the entire input is described on the right and the special part of the input on the left. For example, Theorem 1, asserting that the algorithm underlying it is strongly polynomial with data *encoded as* $[n, |E|; \langle \rho(S), w_1, \ldots, w_d, E\rangle]$, where $\rho(S) \in \mathbb{N}$, $w_1, \ldots, w_d \in \mathbb{Z}^n$ and $E \subset \mathbb{Z}^n$, means that the running time is polynomial in the binary length of $\rho(S)$, $w_1, \ldots, w_d$, and E, and the number of arithmetic operations is polynomial in n and the cardinality $|E|$, which constitute the special part of the input.

Often, as in [31], part of the input is presented by oracles. Then the running time and the number of arithmetic operations count also the number of oracle queries. An oracle algorithm is *polynomial time* if its running time, including the number of oracle queries, and the manipulations of numbers, some of which are answers to oracle queries, is polynomial in the length of the input encoding. An oracle algorithm is *strongly polynomial time* (with specified input encoding as above), if it is polynomial time in the entire input (on the "right"), and in addition, the number of arithmetic operations it performs (including oracle queries) is polynomial in the special part of the input (on the "left").

Reducing Convex to Linear Discrete Optimization

In this section we show that when suitable auxiliary geometric information about the convex hull $\mathrm{conv}(S)$ of a finite set $S \subseteq \mathbb{Z}^n$ is available, the convex discrete optimization problem over S can be reduced to the solution of strongly polynomially many linear discrete optimization counterparts over S. This result will be incorporated into the polynomial time algorithms developed in Sect. "Convex Combinatorial Optimization and More" and Sect. "Convex Integer Programming" for convex combinatorial optimization and convex integer programming respectively. In Sect. "Edge-Directions and Zonotopes" we provide some preliminaries on edge-directions and zonotopes. In Sect. "Strongly Polynomial Reduction of Convex to Linear Discrete Optimization" we prove the reduction which is the main result of this section. In Sect. "Pseudo Polynomial Reduction when Edge-Directions are not Available" we prove a pseudo polynomial reduction for any finite set.

Edge-Directions and Zonotopes

We begin with some terminology and facts that play an important role in the sequel. A *direction* of an edge (1-dimensional face) $e = [u, v]$ of a polytope P is any nonzero scalar multiple of $u - v$. A set of vectors E *covers all edge-directions of* P if it contains a direction of each edge of P. The *normal cone* of a polytope $P \subset \mathbb{R}^n$ at its face F is the (relatively open) cone C_P^F of those linear functionals $h \in \mathbb{R}^n$ which are maximized over P precisely at points of F. A polytope Z is a *refinement* of a polytope P if the normal cone of every vertex of Z is contained in the normal cone of some vertex of P. If Z refines P then, moreover, the closure of each normal cone of P is the union of closures of normal cones of Z. The *zonotope* generated by a set of vectors $E = \{e_1, \ldots, e_m\}$ in $\mathbb{R}^d$ is the following polytope, which is the projection by E of the cube $[-1, 1]^m$ into $\mathbb{R}^d$,

$$\begin{aligned} Z &:= \mathrm{zone}(E) \\ &:= \mathrm{conv}\left\{\sum_{i=1}^m \lambda_i e_i \,:\, \lambda_i = \pm 1\right\} \subset \mathbb{R}^d. \end{aligned}$$

The following fact goes back to Minkowski, see [32].

Lemma 1 *Let P be a polytope and let E be a finite set that covers all edge-directions of P. Then the zonotope $Z := \mathrm{zone}(E)$ generated by E is a refinement of P.*

Proof Consider any vertex u of Z. Then $u = \sum_{e \in E} \lambda_e e$ for suitable $\lambda_e = \pm 1$. Thus, the normal cone C_Z^u consists of those h satisfying $h\lambda_e e > 0$ for all e. Pick any $\hat{h} \in C_Z^u$ and let v be a vertex of P at which $\hat{h}$ is maximized over P. Consider any edge $[v, w]$ of P. Then $v - w = \alpha_e e$ for some scalar $\alpha_e \neq 0$ and some $e \in E$, and $0 \leq \hat{h}(v - w) = \hat{h}\alpha_e e$, implying $\alpha_e \lambda_e > 0$.

It follows that every $h \in C_Z^u$ satisfies $h(v-w)>0$ for every edge of P containing v. Therefore h is maximized over P uniquely at v and hence is in the cone C_P^v of P at v. This shows $C_Z^u \subseteq C_P^v$. Since u was arbitrary, it follows that the normal cone of every vertex of Z is contained in the normal cone of some vertex of P. □

The next lemma provides bounds on the number of vertices of any zonotope and on the algorithmic complexity of constructing its vertices, each vertex along with a linear functional maximized over the zonotope uniquely at that vertex. The bound on the number of vertices has been rediscovered many times over the years. An early reference is [33], stated in the dual form of 2-partitions. A more general treatment is [64]. Recent extensions to p-partitions for any p are in [3,39], and to Minkowski sums of arbitrary polytopes are in [29]. Interestingly, already in [33], back in 1967, the question was raised about the algorithmic complexity of the problem; this is now settled in [20,21] (the latter reference correcting the former). We state the precise bounds on the number of vertices and arithmetic complexity, but will need later only that for any fixed d the bounds are polynomial in the number of generators. Therefore, below we only outline a proof that the bounds are polynomial. Complete details are in the above references.

Lemma 2 *The number of vertices of any zonotope $Z := \mathrm{zone}(E)$ generated by a set E of m vectors in $\mathbb{R}^d$ is at most $2\sum_{k=0}^{d-1}\binom{m-1}{k}$. For every fixed d, there is a strongly polynomial time algorithm that, given $E \subset \mathbb{Z}^d$, encoded as $[m := |E|; \langle E\rangle]$, outputs every vertex v of $Z := \mathrm{zone}(E)$ along with a linear functional $h_v \in \mathbb{Z}^d$ maximized over Z uniquely at v, using $O(m^{d-1})$ arithmetics operations for $d \geq 3$ and $O(m^d)$ for $d \leq 2$.*

Proof We only outline a proof that, for every fixed d, the polynomial bounds $O(m^{d-1})$ on the number of vertices and $O(m^d)$ on the arithmetic complexity hold. We assume that E linearly spans $\mathbb{R}^d$ (else the dimension can be reduced) and is generic, that is, no d points of E lie on a linear hyperplane (one containing the origin). In particular, $0 \notin E$. The same bound for arbitrary E then follows using a perturbation argument (cf. [39]).

Each oriented linear hyperplane $H = \{x \in \mathbb{R}^d : hx = 0\}$ with $h \in \mathbb{R}^d$ nonzero induces a partition of E by $E = H^- \biguplus H^0 \biguplus H^+$, with $H^- := \{e \in E : he < 0\}$, $E^0 := E \cap H$, and $H^+ := \{e \in E : he > 0\}$. The vertices of $Z = \mathrm{zone}(E)$ are in bijection with ordered 2-partitions of E induced by such hyperplanes that avoid E. Indeed, if $E = H^- \biguplus H^+$ then the linear functional $h_v := h$ defining H is maximized over Z uniquely at the vertex $v := \sum\{e : e \in H^+\} - \sum\{e : e \in H^-\}$ of Z.

We now show how to enumerate all such 2-partitions and hence vertices of Z. Let M be any of the $\binom{m}{d-1}$ subsets of E of size $d-1$. Since E is generic, M is linearly independent and spans a unique linear hyperplane $\mathrm{lin}(M)$. Let $\hat{H} = \{x \in \mathbb{R}^d : \hat{h}x = 0\}$ be one of the two orientations of the hyperplane $\mathrm{lin}(M)$. Note that $\hat{H}^0 = M$. Finally, let L be any of the 2^{d-1} subsets of M. Since M is linearly independent, there is a $g \in \mathbb{R}^d$ which linearly separates L from $M \setminus L$, namely, satisfies $gx < 0$ for all $x \in L$ and $gx > 0$ for all $x \in M \setminus L$. Furthermore, there is a sufficiently small $\epsilon > 0$ such that the oriented hyperplane $H := \{x \in \mathbb{R}^d : hx = 0\}$ defined by $h := \hat{h} + \epsilon g$ avoids E and the 2-partition induced by H satisfies $H^- = \hat{H}^- \biguplus L$ and $H^+ = \hat{H}^+ \biguplus (M \setminus L)$. The corresponding vertex of Z is $v := \sum\{e : e \in H^+\} - \sum\{e : e \in H^-\}$ and the corresponding linear functional which is maximized over Z uniquely at v is $h_v := h = \hat{h} + \epsilon g$.

We claim that any ordered 2-partition arises that way from some M, some orientation $\hat{H}$ of $\mathrm{lin}(M)$, and some L. Indeed, consider any oriented linear hyperplane $\tilde{H}$ avoiding E. It can be perturbed to a suitable oriented $\hat{H}$ that touches precisely $d-1$ points of E. Put $M := \hat{H}^0$ so that $\hat{H}$ coincides with one of the two orientations of the hyperplane $\mathrm{lin}(M)$ spanned by M, and put $L := \tilde{H}^- \cap M$. Let H be an oriented hyperplane obtained from M, $\hat{H}$ and L by the above procedure. Then the ordered 2-partition $E = H^- \biguplus H^+$ induced by H coincides with the ordered 2-partition $E = \tilde{H}^- \biguplus \tilde{H}^+$ induced by $\tilde{H}$.

Since there are $\binom{m}{d-1}$ many $(d-1)$-subsets $M \subseteq E$, two orientations $\hat{H}$ of $\mathrm{lin}(M)$, and 2^{d-1} subsets $L \subseteq M$, and d is fixed, the total number of 2-partitions and hence also the total number of vertices of Z obey the upper bound $2^d\binom{m}{d-1} = O(m^{d-1})$. Furthermore, for each choice of M, $\hat{H}$ and L, the linear functional $\hat{h}$ defining $\hat{H}$, as well as g, ϵ, $h_v = h = \hat{h} + \epsilon g$, and the vertex $v = \sum\{e : e \in H^+\} - \sum\{e : e \in H^-\}$ of Z at which h_v is uniquely maximized over Z, can all be com-

puted using $O(m)$ arithmetic operations. This shows the claimed bound $O(m^d)$ on the arithmetic complexity. □

We conclude with a simple fact about edge-directions of projections of polytopes.

Lemma 3 *If E covers all edge-directions of a polytope P, and $Q := \omega(P)$ is the image of P under a linear map $\omega\colon \mathbb{R}^n \longrightarrow \mathbb{R}^d$, then $\omega(E)$ covers all edge-directions of Q.*

Proof Let f be a direction of an edge $[x, y]$ of Q. Consider the face $F := \omega^{-1}([x, y])$ of P. Let V be the set of vertices of F and let $U = \{u \in V : \omega(u) = x\}$. Then for some $u \in U$ and $v \in V \setminus U$, there must be an edge $[u, v]$ of F, and hence of P. Then $\omega(v) \in (x, y]$ hence $\omega(v) = x + \alpha f$ for some $\alpha \neq 0$. Therefore, with $e := \frac{1}{\alpha}(v - u)$, a direction of the edge $[u, v]$ of P, we find that $f = \frac{1}{\alpha}(\omega(v) - \omega(u)) = \omega(e) \in \omega(E)$. □

Strongly Polynomial Reduction of Convex to Linear Discrete Optimization

A *linear discrete optimization oracle* for a set $S \subseteq \mathbb{Z}^n$ is one that, queried on $w \in \mathbb{Z}^n$, either returns an optimal solution to the linear discrete optimization problem over S, that is, an $x^* \in S$ satisfying $wx^* = \max\{wx : x \in S\}$, or asserts that none exists, that is, either the problem is infeasible or the objective function is unbounded. We now show that a set E covering all edge-directions of the polytope $\mathrm{conv}(S)$ underlying a convex discrete optimization problem over a finite set $S \subset \mathbb{Z}^n$ allows to solve it by solving polynomially many linear discrete optimization counterparts over S. The following theorem extends and unifies the corresponding reductions in [49] and [12] for convex combinatorial optimization and convex integer programming respectively. Recall from Sect. "Terminology and Complexity" that the *radius* of a finite set $S \subset \mathbb{Z}^n$ is defined to be $\rho(S) := \max\{|x_i| : x \in S,\ i = 1, \ldots, n\}$.

Theorem 6 *For every fixed d there is a strongly polynomial time algorithm that, given finite set $S \subset \mathbb{Z}^n$ presented by a linear discrete optimization oracle, integer vectors $w_1, \ldots, w_d \in \mathbb{Z}^n$, set $E \subset \mathbb{Z}^n$ covering all edge-directions of* conv(S)*, and convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[n, |E|; \langle \rho(S), w_1, \ldots, w_d, E\rangle]$, solves the convex discrete optimization problem*

$$\max\{c(w_1 x, \ldots, w_d x) : x \in S\}\,.$$

Proof First, query the linear discrete optimization oracle presenting S on the trivial linear functional $w = 0$. If the oracle asserts that there is no optimal solution then S is empty so terminate the algorithm asserting that no optimal solution exists to the convex discrete optimization problem either. So assume the problem is feasible. Let $P := \mathrm{conv}(S) \subset \mathbb{R}^n$ and $Q := \{(w_1 x, \ldots, w_d x) : x \in P\} \subset \mathbb{R}^d$. Then Q is a projection of P, and hence by Lemma 3 the projection $D := \{(w_1 e, \ldots, w_d e) : e \in E\}$ of the set E is a set covering all edge-directions of Q. Let $Z := \mathrm{zone}(D) \subset \mathbb{R}^d$ be the zonotope generated by D. Since d is fixed, by Lemma 2 we can produce in strongly polynomial time all vertices of Z, every vertex v along with a linear functional $h_v \in \mathbb{Z}^d$ maximized over Z uniquely at v. For each of these polynomially many h_v, repeat the following procedure. Define a vector $g_v \in \mathbb{Z}^n$ by $g_{v,j} := \sum_{i=1}^{d} w_{i,j} h_{v,i}$ for $j = 1, \ldots, n$. Now query the linear discrete optimization oracle presenting S on the linear functional $w := g_v \in \mathbb{Z}^n$. Let $x_v \in S$ be the optimal solution obtained from the oracle, and let $z_v := (w_1 x_v, \ldots, w_d x_v) \in Q$ be its projection. Since $P = \mathrm{conv}(S)$, we have that x_v is also a maximizer of g_v over P. Since for every $x \in P$ and its projection $z := (w_1 x, \ldots, w_d x) \in Q$ we have $h_v z = g_v x$, we conclude that z_v is a maximizer of h_v over Q. Now we claim that each vertex u of Q equals some z_v. Indeed, since Z is a refinement of Q by Lemma 1, it follows that there is some vertex v of Z such that h_v is maximized over Q uniquely at u, and therefore $u = z_v$. Since $c(w_1 x, \ldots, w_d x)$ is convex on $\mathbb{R}^n$ and c is convex on $\mathbb{R}^d$, we find that

$$\begin{aligned}
&\max_{x \in S} c(w_1 x, \ldots, w_d x)\\
&= \max_{x \in P} c(w_1 x, \ldots, w_d x)\\
&= \max_{z \in Q} c(z)\\
&= \max\{c(u) : u \text{ vertex of } Q\}\\
&= \max\{c(z_v) : v \text{ vertex of } Z\}\,.
\end{aligned}$$

Using the comparison oracle of c, find a vertex v of Z attaining maximum value $c(z_v)$, and output $x_v \in S$, an optimal solution to the convex discrete optimization problem. □

Pseudo Polynomial Reduction when Edge-Directions Are not Available

Theorem 6 reduces convex discrete optimization to polynomially many linear discrete optimization counterparts when a set covering all edge-directions of the underlying polytope is available. However, often such a set is not available (see e. g. [8] for the important case of bipartite matching). We now show how to reduce convex discrete optimization to many linear discrete optimization counterparts when a set covering all edge-directions is not offhand available. In the absence of such a set, the problem is much harder, and the algorithm below is polynomially bounded only in the unary length of the radius $\rho(S)$ and of the linear functionals $w_1, \ldots, w_d$, rather than in their binary length $\langle \rho(S), w_1, \ldots, w_d \rangle$ as in the algorithm of Theorem 6 Moreover, an upper bound $\rho \geq \rho(S)$ on the radius of S is required to be given explicitly in advance as part of the input.

Theorem 7 *For every fixed d there is a polynomial time algorithm that, given finite set $S \subseteq \mathbb{Z}^n$ presented by a linear discrete optimization oracle, integer $\rho \geq \rho(S)$, vectors $w_1, \ldots, w_d \in \mathbb{Z}^n$, and convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[\rho, w_1, \ldots, w_d]$, solves the convex discrete optimization problem*

$$\max\{c(w_1 x, \ldots, w_d x) : x \in S\}.$$

Proof Let $P := \mathrm{conv}(S) \subset \mathbb{R}^n$, let $T := \{(w_1 x, \ldots, w_d x) : x \in S\}$ be the projection of S by $w_1, \ldots, w_d$, and let $Q := \mathrm{conv}(T) \subset \mathbb{R}^d$ be the corresponding projection of P. Let $r := n\rho \max_{i=1}^d \|w_i\|_\infty$ and let $G := \{-r, \ldots, -1, 0, 1, \ldots, r\}^d$. Then $T \subseteq G$ and the number $(2r+1)^d$ of points of G is polynomially bounded in the input as encoded.

Let $D := \{u - v : u, v \in G, u \neq v\}$ be the set of differences of pairs of distinct point of G. It covers all edge-directions of Q since $\mathrm{vert}(Q) \subseteq T \subseteq G$. Moreover, the number of points of D is less than $(2r+1)^{2d}$ and hence polynomial in the input. Now invoke the algorithm of Theorem 6: while the algorithm requires a set E covering all edge-directions of P, it needs E only to compute a set D covering all edge-directions of the projection Q (see proof of Theorem 6, which here is computed directly. □

Convex Combinatorial Optimization and More

In this section we discuss convex combinatorial optimization. The main result is that convex combinatorial optimization over a set $S \subseteq \{0,1\}^n$ presented by a membership oracle can be solved in strongly polynomial time provided it is endowed with a set covering all edge-directions of $\mathrm{conv}(S)$. In particular, the standard linear combinatorial optimization problem over S can be solved in strongly polynomial time as well. In Sect. "From Membership to Linear Optimization" we provide some preparatory statements involving various oracle presentation of the feasible set S. In Sect. "Linear and Convex Combinatorial Optimization in Strongly Polynomial Time" we combine these preparatory statements with Theorem 6 and prove the main result of this section. An extension to arbitrary finite sets $S \subset \mathbb{Z}^n$ endowed with edge-directions is established in Sect. "Linear and Convex Discrete Optimization over any Set in Pseudo Polynomial Time". We conclude with some applications in Sect. "Some Applications".

As noted in the introduction, when S is contained in $\{0,1\}^n$ we refer to discrete optimization over S also as *combinatorial optimization* over S, to emphasize that S typically represents a family $\mathcal{F} \subseteq 2^N$ of subsets of a ground set $N := \{1, \ldots, n\}$ possessing some combinatorial property of interest (for instance, the family of bases of a matroid over N, see Sect. "Matroids and Maximum Norm Spanning Trees". The convex combinatorial optimization problem then also has the following interpretation (taken in [47,49]). We are given a weighting $\omega : N \longrightarrow \mathbb{Z}^d$ of elements of the ground set by d-dimensional integer vectors. We interpret the weight vector $\omega(j) \in \mathbb{Z}^d$ of element j as representing its value under d criteria (e. g., if N is the set of edges in a network then such criteria may include profit, reliability, flow velocity, etc.). The weight of a subset $F \subseteq N$ is the sum $\omega(F) := \sum_{j \in F} \omega(j)$ of weights of its elements, representing the total value of F under the d criteria. Now, given a convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$, the objective function value of $F \subseteq N$ is the "convex balancing" $c(\omega(F))$ of the values of the weight vector of F. The convex combinatorial optimization problem is to find a family member $F \in \mathcal{F}$ maximizing $c(\omega(F))$. The usual linear combinatorial optimization problem over $\mathcal{F}$ is the special case of $d = 1$ and c the identity on

$\mathbb{R}$. To cast a problem of that form in our usual setup just let $S := \{\mathbf{1}_F : F \in \mathcal{F}\} \subseteq \{0,1\}^n$ be the set of indicators of members of $\mathcal{F}$ and define weight vectors $w_1, \ldots, w_d \in \mathbb{Z}^n$ by $w_{i,j} := \omega(j)_i$ for $i = 1, \ldots, d$ and $j = 1, \ldots, n$.

From Membership to Linear Optimization

A *membership oracle* for a set $S \subseteq \mathbb{Z}^n$ is one that, queried on $x \in \mathbb{Z}^n$, asserts whether or not $x \in S$. An *augmentation oracle* for S is one that, queried on $x \in S$ and $w \in \mathbb{Z}^n$, either returns an $\hat{x} \in S$ with $w\hat{x} > wx$, i. e. a better point of S, or asserts that none exists, i. e. x is optimal for the linear discrete optimization problem over S.

A membership oracle presentation of S is very broad and available in all reasonable applications, but reveals little information on S, making it hard to use. However, as we now show, the edge-directions of $\mathrm{conv}(S)$ allow to convert membership to augmentation.

Lemma 4 *There is a strongly polynomial time algorithm that, given set $S \subseteq \{0,1\}^n$ presented by a membership oracle, $x \in S$, $w \in \mathbb{Z}^n$, and set $E \subset \mathbb{Z}^n$ covering all edge-directions of the polytope $\mathrm{conv}(S)$, encoded as $[n, |E|; \langle x, w, E\rangle]$, either returns a better point $\hat{x} \in S$, that is, one satisfying $w\hat{x} > wx$, or asserts that none exists.*

Proof Each edge of $P := \mathrm{conv}(S)$ is the difference of two $\{0,1\}$-vectors. Therefore, each edge direction of P is, up to scaling, a $\{-1,0,1\}$-vector. Thus, scaling $e := \frac{1}{\|e\|_\infty} e$ and $e := -e$ if necessary, we may and will assume that $e \in \{-1,0,1\}^n$ and $we \geq 0$ for all $e \in E$. Now, using the membership oracle, check if there is an $e \in E$ such that $x + e \in S$ and $we > 0$. If there is such an e then output $\hat{x} := x + e$ which is a better point, whereas if there is no such e then terminate asserting that no better point exists.

Clearly, if the algorithm outputs an $\hat{x}$ then it is indeed a better point. Conversely, suppose x is not a maximizer of w over S. Since $S \subseteq \{0,1\}^n$, the point x is a vertex of P. Since x is not a maximizer of w, there is an edge $[x, \hat{x}]$ of P with $\hat{x}$ a vertex satisfying $w\hat{x} > wx$. But then $e := \hat{x} - x$ is the one $\{-1,0,1\}$ edge-direction of $[x, \hat{x}]$ with $we \geq 0$ and hence $e \in E$. Thus, the algorithm will find and output $\hat{x} = x + e$ as it should. □

An augmentation oracle presentation of a finite S allows to solve the linear discrete optimization problem $\max\{wx : x \in S\}$ over S by starting from any feasible $x \in S$ and repeatedly augmenting it until an optimal solution $x^* \in S$ is reached. The next lemma bounds the running time needed to reach optimality using this procedure. While the running time is polynomial in the binary length of the linear functional w and the initial point x, it is more sensitive to the radius $\rho(S)$ of the feasible set S, and is polynomial only in its unary length. The lemma is an adaptation of a result of [30,57] (stated therein for $\{0,1\}$-sets), which makes use of bit-scaling ideas going back to [23].

Lemma 5 *There is a polynomial time algorithm that, given finite set $S \subset \mathbb{Z}^n$ presented by an augmentation oracle, $x \in S$, and $w \in \mathbb{Z}^n$, encoded as $[\rho(S), \langle x, w\rangle]$, provides an optimal solution $x^* \in S$ to the linear discrete optimization problem $\max\{wz : z \in S\}$.*

Proof Let $k := \max_{j=1}^n \lceil \log_2(|w_j| + 1)\rceil$ and note that $k \leq \langle w\rangle$. For $i = 0, \ldots, k$ define a linear functional $u_i = (u_{i,1}, \ldots, u_{i,n}) \in \mathbb{Z}^n$ by $u_{i,j} := \mathrm{sign}(w_j) \lfloor 2^{i-k}|w_j|\rfloor$ for $j = 1, \ldots, n$. Then $u_0 = 0$, $u_k = w$, and $u_i - 2u_{i-1} \in \{-1,0,1\}^n$ for all $i = 1, \ldots, k$.

We now describe how to construct a sequence of points $y_0, y_1, \ldots, y_k \in S$ such that y_i is an optimal solution to $\max\{u_i y : y \in S\}$ for all i. First note that all points of S are optimal for $u_0 = 0$ and hence we can take $y_0 := x$ to be the point of S given as part of the input. We now explain how to determine y_i from y_{i-1} for $i = 1, \ldots, k$. Suppose y_{i-1} has been determined. Set $\tilde{y} := y_{i-1}$. Query the augmentation oracle on $\tilde{y} \in S$ and u_i; if the oracle returns a better point $\hat{y}$ then set $\tilde{y} := \hat{y}$ and repeat, whereas if it asserts that there is no better point then the optimal solution for u_i is read off to be $y_i := \tilde{y}$. We now bound the number of calls to the oracle. Each time the oracle is queried on $\tilde{y}$ and u_i and returns a better point $\hat{y}$, the improvement is by at least one, i. e. $u_i(\hat{y} - \tilde{y}) \geq 1$; this is so because u_i, $\tilde{y}$ and $\hat{y}$ are integer. Thus, the number of necessary augmentations from y_{i-1} to y_i is at most the total improvement, which we claim satisfies

$$u_i(y_i - y_{i-1}) = (u_i - 2u_{i-1})(y_i - y_{i-1}) + 2u_{i-1}(y_i - y_{i-1}) \leq 2n\rho + 0 = 2n\rho\,,$$

where $\rho := \rho(S)$. Indeed, $u_i - 2u_{i-1} \in \{-1,0,1\}^n$ and $y_i, y_{i-1} \in S \subset [-\rho, \rho]^n$ imply $(u_i - 2u_{i-1})(y_i -$

$y_{i-1}) \leq 2n\rho$; and y_{i-1} optimal for u_{i-1} gives $u_{i-1}(y_i - y_{i-1}) \leq 0$.

Thus, after a total number of at most $2n\rho k$ calls to the oracle we obtain y_k which is optimal for u_k. Since $w = u_k$ we can output $x^* := y_k$ as the desired optimal solution to the linear discrete optimization problem. Clearly the number $2n\rho k$ of calls to the oracle, as well as the number of arithmetic operations and binary length of numbers occurring during the algorithm, are polynomial in $\rho(S), \langle x, w\rangle$. This completes the proof.□

We conclude this preparatory subsection by recording the following result of [24] which incorporates the heavy simultaneous Diophantine approximation of [44].

Proposition 1 *There is a strongly polynomial time algorithm that, given $w \in \mathbb{Z}^n$, encoded as $[n; \langle w\rangle]$, produces $\hat{w} \in \mathbb{Z}^n$, whose binary length $\langle \hat{w}\rangle$ is polynomially bounded in n and independent of w, and with $sign(\hat{w}z) = sign(wz)$ for every $z \in \{-1, 0, 1\}^n$.*

Linear and Convex Combinatorial Optimization in Strongly Polynomial Time

Combining the preparatory statements of Sect. "From Membership to Linear Optimization" with Theorem 6, we can now solve the convex combinatorial optimization over a set $S \subseteq \{0, 1\}^n$ presented by a membership oracle and endowed with a set covering all edge-directions of conv(S) in strongly polynomial time. We start with the special case of linear combinatorial optimization.

Theorem 8 *There is a strongly polynomial time algorithm that, given set $S \subseteq \{0, 1\}^n$ presented by a membership oracle, $x \in S$, $w \in \mathbb{Z}^n$, and set $E \subset \mathbb{Z}^n$ covering all edge-directions of the polytope* conv(S)*, encoded as $[n, |E|; \langle x, w, E\rangle]$, provides an optimal solution $x^* \in S$ to the linear combinatorial optimization problem* $\max\{wz : z \in S\}$.

Proof First, an augmentation oracle for S can be simulated using the membership oracle, in strongly polynomial time, by applying the algorithm of Lemma 4.

Next, using the simulated augmentation oracle for S, we can now do linear optimization over S in strongly polynomial time as follows. First, apply to w the algorithm of Proposition 1 and obtain $\hat{w} \in \mathbb{Z}^n$ whose binary length $\langle \hat{w}\rangle$ is polynomially bounded in n, which satisfies $\text{sign}(\hat{w}z) = \text{sign}(wz)$ for every $z \in \{-1, 0, 1\}^n$. Since $S \subseteq \{0, 1\}^n$, it is finite and has radius $\rho(S) = 1$. Now apply the algorithm of Lemma 5 to S, x and $\hat{w}$, and obtain a maximizer x^* of $\hat{w}$ over S. For every $y \in \{0, 1\}^n$ we then have $x^* - y \in \{-1, 0, 1\}^n$ and hence $\text{sign}(w(x^* - y)) = \text{sign}(\hat{w}(x^* - y))$. So x^* is also a maximizer of w over S and hence an optimal solution to the given linear combinatorial optimization problem. Now, $\rho(S) = 1$, $\langle \hat{w}\rangle$ is polynomial in n, and $x \in \{0, 1\}^n$ and hence $\langle x\rangle$ is linear in n. Thus, the entire length of the input $[\rho(S), \langle x, \hat{w}\rangle]$ to the polynomial-time algorithm of Lemma 5 is polynomial in n, and so its running time is in fact strongly polynomial on that input. □

Combining Theorems 6 and 8 we recover at once the following result of [49].

Theorem 9 *For every fixed d there is a strongly polynomial time algorithm that, given set $S \subseteq \{0, 1\}^n$ presented by a membership oracle, $x \in S$, vectors $w_1, \ldots, w_d \in \mathbb{Z}^n$, set $E \subset \mathbb{Z}^n$ covering all edge-directions of the polytope* conv(S)*, and convex functional $c : \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[n, |E|; \langle x, w_1, \ldots, w_d, E\rangle]$, provides an optimal solution $x^* \in S$ to the convex combinatorial optimization problem*

$$\max\{c(w_1 z, \ldots, w_d z) : z \in S\}.$$

Proof Since S is nonempty, a linear discrete optimization oracle for S can be simulated in strongly polynomial time by the algorithm of Theorem 8. Using this simulated oracle, we can apply the algorithm of Theorem 6 and solve the given convex combinatorial optimization problem in strongly polynomial time. □

Linear and Convex Discrete Optimization over any Set in Pseudo Polynomial Time

In Sect. "Linear and Convex Combinatorial Optimization in Strongly Polynomial Time" above we developed strongly polynomial time algorithms for linear and convex discrete optimization over $\{0, 1\}$-sets. We now provide extensions of these algorithms to arbitrary finite sets $S \subset \mathbb{Z}^n$. As can be expected, the algorithms become slower.

We start by recording the following fundamental result of Khachiyan [40] asserting that linear programming is polynomial time solvable via the ellipsoid

method [63]. This result will be used below as well as several more times later in the monograph.

Proposition 2 *There is a polynomial time algorithm that, given $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$, and $w \in \mathbb{Z}^n$, encoded as $[\langle A, b, w\rangle]$, either asserts that $P := \{x \in \mathbb{R}^n : Ax \le b\}$ is empty, or asserts that the linear functional wx is unbounded over P, or provides a vertex $v \in vert(P)$ which is an optimal solution to the linear program* $\max\{wx : x \in P\}$.

The following analog of Lemma 4 shows how to covert membership to augmentation in polynomial time, albeit, no longer in strongly polynomial time. Here, both the given initial point x and the returned better point $\hat{x}$ if any, are *vertices* of conv(S).

Lemma 6 *There is a polynomial time algorithm that, given finite set $S \subset \mathbb{Z}^n$ presented by a membership oracle, vertex x of the polytope* conv(S), $w \in \mathbb{Z}^n$, *and set $E \subset \mathbb{Z}^n$ covering all edge-directions of* conv(S), *encoded as $[\rho(S), \langle x, w, E\rangle]$, either returns a better vertex $\hat{x}$ of* conv(S), *that is, one satisfying $w\hat{x} > wx$, or asserts that none exists.*

Proof Dividing each vector $e \in E$ by the greatest common divisor of its entries and setting $e := -e$ if necessary, we can and will assume that each e is *primitive*, that is, its entries are relatively prime integers, and $we \ge 0$. Using the membership oracle, construct the subset $F \subseteq E$ of those $e \in E$ for which $x + re \in S$ for some $r \in \{1, \ldots, 2\rho(S)\}$. Let $G \subseteq F$ be the subset of those $f \in F$ for which $wf > 0$. If G is empty then terminate asserting that there is no better vertex. Otherwise, consider the convex cone cone(F) generated by F. It is clear that x is incident on an edge of conv(S) in direction f if and only if f is an extreme ray of cone(F). Moreover, since $G = \{f \in F : wf > 0\}$ is nonempty, there must be an extreme ray of cone(F) which lies in G. Now $f \in F$ is an extreme ray of cone(F) if and only if there do not exist nonnegative λ_e, $e \in F \setminus \{f\}$, such that $f = \sum_{e\neq f} \lambda_e e$; this can be checked in polynomial time using linear programming. Applying this procedure to each $f \in G$, identify an extreme ray $g \in G$. Now, using the membership oracle, determine the largest $r \in \{1, \ldots, 2\rho(S)\}$ for which $x + rg \in S$. Output $\hat{x} := x + rg$ which is a better vertex of conv(S). □

We now prove the extensions of Theorems 8 and 9 to arbitrary, not necessarily $\{0, 1\}$-valued, finite sets. While the running time remains polynomial in the binary length of the weights $w_1, \ldots, w_d$ and the set of edge-directions E, it is more sensitive to the radius $\rho(S)$ of the feasible set S, and is polynomial only in its unary length. Here, the initial feasible point and the optimal solution output by the algorithms are vertices of conv(S). Again, we start with the special case of linear combinatorial optimization.

Theorem 10 *There is a polynomial time algorithm that, given finite $S \subset \mathbb{Z}^n$ presented by a membership oracle, vertex x of the polytope* conv(S), $w \in \mathbb{Z}^n$, *and set $E \subset \mathbb{Z}^n$ covering all edge-directions of* conv(S), *encoded as $[\rho(S), \langle x, w, E\rangle]$, provides an optimal solution $x^* \in S$ to the linear discrete optimization problem* $\max\{wz : z \in S\}$.

Proof Apply the algorithm of Lemma 5 to the given data. Consider any query $x' \in S$, $w' \in \mathbb{Z}^n$ made by that algorithm to an augmentation oracle for S. To answer it, apply the algorithm of Lemma 6 to x' and w'. Since the first query made by the algorithm of Lemma 5 is on the given input vertex $x' := x$, and any consequent query is on a point $x' := \hat{x}$ which was the reply of the augmentation oracle to the previous query (see proof of Lemma 5), we see that the algorithm of Lemma 6 will always be asked on a vertex of S and reply with another. Thus, the algorithm of Lemma 6 can answer all augmentation queries and enables the polynomial time solution of the given problem. □

Theorem 11 *For every fixed d there is a polynomial time algorithm that, given finite set $S \subseteq \mathbb{Z}^n$ presented by membership oracle, vertex x of* conv(S), *vectors $w_1, \ldots, w_d \in \mathbb{Z}^n$, set $E \subset \mathbb{Z}^n$ covering all edge-directions of the polytope* conv(S), *and convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[\rho(S), \langle x, w_1, \ldots, w_d, E\rangle]$, provides an optimal solution $x^* \in S$ to the convex combinatorial optimization problem*

$$\max\{c(w_1 z, \ldots, w_d z) : z \in S\}.$$

Proof Since S is nonempty, a linear discrete optimization oracle for S can be simulated in polynomial time by the algorithm of Theorem 10 . Using this simulated oracle, we can apply the algorithm of Theorem 6 and solve the given problem in polynomial time. □

Some Applications

Positive Semidefinite Quadratic Binary Programming The quadratic binary programming problem is the following: given an $n \times n$ matrix M, find a vector $x \in \{0,1\}^n$ maximizing the quadratic form $x^T Mx$ induced by M. We consider here the instance where M is positive semidefinite, in which case it can be assumed to be presented as $M = W^T W$ with W a given $d \times n$ matrix. Already this restricted version is very broad: if the rank d of W and M is variable then, as mentioned in the introduction, the problem is NP-hard. We now show that, for fixed d, Theorem 9 implies at once that the problem is strongly polynomial time solvable (see also [2]).

Corollary 6 *For every fixed d there is a strongly polynomial time algorithm that given $W \in \mathbb{Z}^{d\times n}$, encoded as $[n; \langle W\rangle]$, finds $x^* \in \{0,1\}^n$ maximizing the form $x^T W^T Wx$.*

Proof Let $S := \{0,1\}^n$ and let $E := \{\mathbf{1}_1, \ldots, \mathbf{1}_n\}$ be the set of unit vectors in $\mathbb{R}^n$. Then $P := \text{conv}(S)$ is just the n-cube $[0,1]^n$ and hence E covers all edge-directions of P. A membership oracle for S is easily and efficiently realizable and $x := 0 \in S$ is an initial point. Also, $|E|$ and $\langle E\rangle$ are polynomial in n, and E is easily and efficiently computable.

Now, for $i = 1, \ldots, d$ define $w_i \in \mathbb{Z}^n$ to be the ith row of the matrix W, that is, $w_{i,j} := W_{i,j}$ for all i, j. Finally, let $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ be the squared l_2 norm given by $c(y) := \|y\|_2^2 := \sum_{i=1}^d y_i^2$, and note that the comparison of $c(y)$ and $c(z)$ can be done for $y, z \in \mathbb{Z}^d$ in time polynomial in $\langle y, z\rangle$ using a constant number of arithmetic operations, providing a strongly polynomial time realization of a comparison oracle for c.

This translates the given quadratic programming problem into a convex combinatorial optimization problem over S, which can be solved in strongly polynomial time by applying the algorithm of Theorem 9 to S, $x = 0$, $w_1, \ldots, w_d$, E, and c. □

Matroids and Maximum Norm Spanning Trees Optimization problems over matroids form a fundamental class of combinatorial optimization problems. Here we discuss matroid bases, but everything works for independent sets as well. Recall that a family $\mathcal{B}$ of subsets of $\{1, \ldots, n\}$ is the family of *bases* of a *matroid* if all members of $\mathcal{B}$ have the same cardinality, called the *rank* of the matroid, and for every $B, B' \in \mathcal{B}$ and $i \in B \setminus B'$ there is a $j \in B'$ such that $B \setminus \{i\} \cup \{j\} \in \mathcal{B}$. Useful models include the *graphic matroid* of a graph G with edge set $\{1, \ldots, n\}$ and $\mathcal{B}$ the family of spanning forests of G, and the *linear matroid* of an $m \times n$ matrix A with $\mathcal{B}$ the family of sets of indices of maximal linearly independent subsets of columns of A.

It is well known that linear combinatorial optimization over matroids can be solved by the fast greedy algorithm [22]. We now show that, as a consequence of Theorem 9, convex combinatorial optimization over a matroid presented by a membership oracle can be solved in strongly polynomial time as well (see also [34,47]). We state the result for bases, but the analogous statement for independent sets hold as well. We say that $S \subseteq \{0,1\}^n$ is the *set of bases of a matroid* if it is the set of indicators of the family $\mathcal{B}$ of bases of some matroid, in which case we call $\text{conv}(S)$ the *matroid base polytope*.

Corollary 7 *For every fixed d there is a strongly polynomial time algorithm that, given set $S \subseteq \{0,1\}^n$ of bases of a matroid presented by a membership oracle, $x \in S$, $w_1, \ldots, w_d \in \mathbb{Z}^n$, and convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[n; \langle x, w_1, \ldots, w_d\rangle]$, solves the convex matroid optimization problem*

$$\max\{c(w_1 z, \ldots, w_d z) : z \in S\}\,.$$

Proof Let $E := \{\mathbf{1}_i - \mathbf{1}_j : 1 \le i < j \le n\}$ be the set of differences of pairs of unit vectors in $\mathbb{R}^n$. We claim that E covers all edge-directions of the matroid base polytope $P := \text{conv}(S)$. Consider any edge $e = [y, y']$ of P with $y, y' \in S$ and let $B := \text{supp(y)}$ and $B' := \text{supp(y}')$ be the corresponding bases. Let $h \in \mathbb{R}^n$ be a linear functional uniquely maximized over P at e. If $B \setminus B' = \{i\}$ is a singleton then $B' \setminus B = \{j\}$ is a singleton as well in which case $y - y' = \mathbf{1}_i - \mathbf{1}_j$ and we are done. Suppose then, indirectly, that it is not, and pick an element i in the symmetric difference $B \Delta B' := (B \setminus B') \cup (B' \setminus B)$ of minimum value h_i. Without loss of generality assume $i \in B \setminus B'$. Then there is a $j \in B' \setminus B$ such that $B'' := B \setminus \{i\} \cup \{j\}$ is also a basis. Let $y'' \in S$ be the indicator of B''. Now $|B \Delta B' q| > 2$ implies that B'' is neither B nor B'. By the choice of i we have $hy'' = hy - h_i + h_j \ge hy$. So y'' is also a max-

imizer of h over P and hence $y'' \in e$. But no $\{0,1\}$-vector is a convex combination of others, a contradiction.

Now, $|E| = \binom{n}{2}$ and $E \subset \{-1,0,1\}^n$ imply that $|E|$ and $\langle E \rangle$ are polynomial in n. Moreover, E can be easily computed in strongly polynomial time. Therefore, applying the algorithm of Theorem 9 to the given data and the set E, the convex discrete optimization problem over S can be solved in strongly polynomial time. □

One important application of Corollary 7 is a polynomial time algorithm for computing the *universal Gröbner basis* of any system of polynomials having a finite set of common zeros in fixed (but arbitrary) number of variables, as well as the construction of the *state polyhedron* of any member of the *Hilbert scheme*, see [5,51]. Other important applications are in the field of *algebraic statistics* [52], in particular for *optimal experimental design*. These applications are beyond our scope here and will be discussed elsewhere.

Here is another concrete example of a convex matroid optimization application.

Example 1 (MAXIMUM NORM SPANNING TREE). Fix any positive integer d. Let $\|\cdot\|_p \colon \mathbb{R}^d \longrightarrow \mathbb{R}$ be the l_p norm given by $\|x\|_p := (\sum_{i=1}^d |x_i|^p)^{\frac{1}{p}}$ for $1 \le p < \infty$ and $\|x\|_\infty := \max_{i=1}^d |x_i|$. Let G be a connected graph with edge set $N := \{1,\ldots,n\}$. For $j = 1,\ldots,n$ let $u_j \in \mathbb{Z}^d$ be a weight vector representing the values of edge j under some d criteria. The weight of a subset $T \subseteq N$ is the sum $\sum_{j\in T} u_j$ representing the total values of T under the d criteria. The problem is to find a spanning tree T of G whose weight has maximum l_p norm, that is, a spanning tree T maximizing $\|\sum_{j\in T} u_j\|_p$.

Define $w_1,\ldots,w_d \in \mathbb{Z}^n$ by $w_{i,j} := u_{j,i}$ for $i = 1,\ldots,d$, $j = 1,\ldots,n$. Let $S \subseteq \{0,1\}^n$ be the set of indicators of spanning trees of G. Then, in time polynomial in n, a membership oracle for S is realizable, and an initial $x \in S$ is obtainable as the indicator of any greedily constructible spanning tree T. Finally, define the convex functional $c := \|\cdot\|_p$. Then for most common values $p = 1, 2, \infty$, and in fact for any $p \in \mathbb{N}$, the comparison of $c(y)$ and $c(z)$ can be done for $y, z \in \mathbb{Z}^d$ in time polynomial in $\langle y, z, p\rangle$ by computing and comparing the integer valued pth powers $\|y\|_p^p$ and $\|z\|_p^p$. Thus, by Corollary 7, this problem is solvable in time polynomial in $\langle u_1,\ldots,u_n,p\rangle$.

Linear N-fold Integer Programming

In this section we develop a theory of linear *n-fold integer programming*, which leads to the polynomial time solution of broad classes of linear integer programming problems in variable dimension. This will be extended to convex n-fold integer programming in Sect. "Convex Integer Programming".

In Sect. "Oriented Augmentation and Linear Optimization" we describe an adaptation of a result of [56] involving an oriented version of the augmentation oracle of Sect. "From Membership to Linear Optimization". In Sect. "Graver Bases and Linear Integer Programming" we discuss Graver bases and their application to linear integer programming. In Sect. "Graver Bases of N-fold Matrices" we show that Graver bases of n-fold matrices can be computed efficiently. In Sect. "Linear N-fold Integer Programming in Polynomial Time" we combine the preparatory statements from Sect. "Oriented Augmentation and Linear Optimization", Sect. "Graver Bases and Linear Integer Programming", and Sect. "Graver Bases of N-fold Matrices", and prove the main result of this section, asserting that linear n-fold integer programming is polynomial time solvable. We conclude with some applications in Sect. "Some Applications".

Here and in Sect. "Convex Integer Programming" we concentrate on discrete optimization problems over a set S presented as the set of integer points satisfying an explicitly given system of linear inequalities. Without loss of generality we may and will assume that S is given either in standard form $S := \{x \in \mathbb{N}^n : Ax = b\}$ where $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$, or in the form

$$S := \{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\}$$

where $l, u \in \mathbb{Z}_\infty^n$ and $\mathbb{Z}_\infty = \mathbb{Z} \uplus \{\pm\infty\}$, where some of the variables are bounded below or above and some are unbounded. Thus, S is no longer presented by an oracle, but by the explicit data A, b and possibly l, u. In this setup we refer to discrete optimization over S also as *integer programming* over S. As usual, an algorithm solving the problem must either provide an $x \in S$ maximizing wx over S, or assert that none exists (either because S is empty or because the objective function is unbounded over the underlying polyhedron). We will sometimes assume that an initial point $x \in S$ is given,

in which case b will be computed as $b := Ax$ and not be part of the input.

Oriented Augmentation and Linear Optimization

We have seen in Sect. "From Membership to Linear Optimization" that an augmentation oracle presentation of a finite set $S \subset \mathbb{Z}^n$ enables to solve the linear discrete optimization problem over S. However, the running time of the algorithm of Lemma 5 which demonstrated this, was polynomial in the unary length of the radius $\rho(S)$ of the feasible set rather than in its binary length.

In this subsection we discuss a recent result of [56] and show that, when S is presented by a suitable stronger oriented version of the augmentation oracle, the linear optimization problem can be solved by a much faster algorithm, whose running time is in fact polynomial in the binary length $\langle\rho(S)\rangle$. The key idea behind this algorithm is that it gives preference to augmentations along interior points of conv(S) staying far off its boundary. It is inspired by and extends the combinatorial interior point algorithm of [61].

For any vector $g \in \mathbb{R}^n$, let $g^+, g^- \in \mathbb{R}^n_+$ denote its *positive* and *negative* parts, defined by $g^+_j := \max\{g_j, 0\}$ and $g^-_j := -\min\{g_j, 0\}$ for $j = 1, \ldots, n$. Note that both g^+, g^- are nonnegative, $\mathrm{supp(g)} = \mathrm{supp(g^+)} \biguplus \mathrm{supp(g^-)}$, and $g = g^+ - g^-$.

An *oriented augmentation oracle* for a set $S \subset \mathbb{Z}^n$ is one that, queried on $x \in S$ and $w_+, w_- \in \mathbb{Z}^n$, either returns an *augmenting vector* $g \in \mathbb{Z}^n$, defined to be one satisfying $x + g \in S$ and $w_+g^+ - w_-g^- > 0$, or asserts that none exists.

Note that this oracle involves *two* linear functionals $w_+, w_- \in \mathbb{Z}^n$ rather than one (w_+, w_- are two distinct independent vectors and *not* the positive and negative parts of one vector). The conditions on an augmenting vector g indicate that it is a feasible direction and has positive value under the nonlinear objective function determined by w_+, w_-. Note that this oracle is indeed stronger than the augmentation oracle of Sect. "From Membership to Linear Optimization": to answer a query $x \in S$, $w \in \mathbb{Z}^n$ to the latter, set $w_+ := w_- := w$, thereby obtaining $w_+g^+ - w_-g^- = wg$ for all g, and query the former on x, w_+, w_-; if it replies with an augmenting vector g then reply with the better point $\hat{x} := x + g$, whereas if it asserts that no g exists then assert that no better point exists.

The following lemma is an adaptation of the result of [56] concerning sets of the form $S := \{x \in \mathbb{Z}^n : Ax = b,\ 0 \le x \le u\}$ of nonnegative integer points satisfying equations and upper bounds. However, the pair A, b is neither explicitly needed nor does it affect the running time of the algorithm underlying the lemma. It suffices that S is of that form. Moreover, an arbitrary lower bound vector l rather than 0 can be included. So it suffices to assume that S coincides with the intersection of its affine hull and the set of integer points in a box, that is, $S = \mathrm{aff}(S) \cap \{x \in \mathbb{Z}^n : l \le x \le u\}$ where $l, u \in \mathbb{Z}^n$. We now describe and prove the algorithm of [56] adjusted to any lower and upper bounds l, u.

Lemma 7 *There is a polynomial time algorithm that, given vectors $l, u \in \mathbb{Z}^n$, set $S \subset \mathbb{Z}^n$ satisfying $S = aff(S) \cap \{z \in \mathbb{Z}^n : l \le z \le u\}$ and presented by an oriented augmentation oracle, $x \in S$, and $w \in \mathbb{Z}^n$, encoded as $[\langle l, u, x, w\rangle]$, provides an optimal solution $x^* \in S$ to the linear discrete optimization problem* $\max\{wz : z \in S\}$.

Proof We start with some strengthening adjustments to the oriented augmentation oracle. Let $\rho := \max\{\|l\|_\infty, \|u\|_\infty\}$ be an upper bound on the radius of S. Then any augmenting vector g obtained from the oriented augmentation oracle when queried on $y \in S$ and $w_+, w_- \in \mathbb{Z}^n$, can be made in polynomial time to be *exhaustive*, that is, to satisfy $y + 2g \notin S$ (which means that no longer augmenting step in direction g can be taken). Indeed, using binary search, find the largest $r \in \{1, \ldots, 2\rho\}$ for which $l \le y + rg \le u$; then $S = \mathrm{aff}(S) \cap \{z \in \mathbb{Z}^n : l \le z \le u\}$ implies $y + rg \in S$ and hence we can replace $g := rg$. So from here on we will assume that if there is an augmenting vector then the oracle returns an exhaustive one. Second, let $\mathbb{R}_\infty := \mathbb{R} \uplus \{\pm\infty\}$ and for any vector $v \in \mathbb{R}^n$ let $v^{-1} \in \mathbb{R}^n_\infty$ denote its entry-wise reciprocal defined by $v^{-1}_i := \frac{1}{v_i}$ if $v_i \neq 0$ and $v^{-1}_i := \infty$ if $v_i = 0$. For any $y \in S$, the vectors $(y-l)^{-1}$ and $(u-y)^{-1}$ are the reciprocals of the "entry-wise distance" of y from the given lower and upper bounds. The algorithm will query the oracle on triples y, w_+, w_- with $w_+ := w - \mu(u-y)^{-1}$ and $w_- := w + \mu(y-l)^{-1}$ where μ is a suitable positive scalar and w is the input linear functional. The fact that such w_+, w_- may have infinite entries does not cause any problem: indeed, if g is an augmenting vector then $y + g \in S$ implies that $g^+_i = 0$ whenever $y_i = u_i$

and $g_i^- = 0$ whenever $l_i = y_i$, so each infinite entry in w_+ or w_- occurring in the expression $w_+g^+ - w_-g^-$ is multiplied by 0 and hence zeroed out.

The algorithm proceeds in phases. Each phase i starts with a feasible point $y_{i-1} \in S$ and performs repeated augmentations using the oriented augmentation oracle, terminating with a new feasible point $y_i \in S$ when no further augmentations are possible. The queries to the oracle make use of a positive scalar parameters μ_i fixed throughout the phase. The first phase (i=1) starts with the input point $y_0 := x$ and sets $\mu_1 := \rho \|w\|_\infty$. Each further phase $i \geq 2$ starts with the point y_{i-1} obtained from the previous phase and sets the parameter value $\mu_i := \frac{1}{2}\mu_{i-1}$ to be half its value in the previous phase. The algorithm terminates at the end of the first phase i for which $\mu_i < \frac{1}{n}$, and outputs $x^* := y_i$. Thus, the number of phases is at most $\lceil \log_2(2n\rho\|w\|_\infty) \rceil$ and hence polynomial in $\langle l, u, w \rangle$.

We now describe the ith phase which determines y_i from y_{i-1}. Set $\mu_i := \frac{1}{2}\mu_{i-1}$ and $\hat{y} := y_{i-1}$. Iterate the following: query the strengthened oriented augmentation oracle on $\hat{y}$, $w_+ := w - \mu_i(u - \hat{y})^{-1}$, and $w_- := w + \mu_i(\hat{y} - l)^{-1}$; if the oracle returns an exhaustive augmenting vector g then set $\hat{y} := \hat{y} + g$ and repeat, whereas if it asserts that there is no augmenting vector then set $y_i := \hat{y}$ and complete the phase. If $\mu_i \geq \frac{1}{n}$ then proceed to the $(i+1)$th phase, else output $x^* := y_i$ and terminate the algorithm.

It remains to show that the output of the algorithm is indeed an optimal solution and that the number of iterations (and hence calls to the oracle) in each phase is polynomial in the input. For this we need the following facts, the easy proofs of which are omitted:

1. For every feasible $y \in S$ and direction g with $y + g \in S$ also feasible, we have

$$(u - y)^{-1}g^+ + (y - l)^{-1}g^- \leq n .$$

2. For every $y \in S$ and direction g with $y + g \in S$ but $y + 2g \notin S$, we have

$$(u - y)^{-1}g^+ + (y - l)^{-1}g^- > \frac{1}{2} .$$

3. For every feasible $y \in S$, direction g with $y + g \in S$ also feasible, and $\mu > 0$, setting $w_+ := w - \mu(u - y)^{-1}$ and $w_- := w + \mu(y - l)^{-1}$ we have

$$w_+g^+ - w_-g^- = wg - \mu\left((u - y)^{-1}g^+ + (y - l)^{-1}g^-\right) .$$

Now, consider the last phase i with $\mu_i < \frac{1}{n}$, let $x^* := y_i := \hat{y}$ be the output of the algorithm at the end of this phase, and let $\hat{x} \in S$ be any optimal solution. Now, the phase is completed when the oracle, queried on the triple $\hat{y}$, $w_+ = w - \mu_i(u - \hat{y})^{-1}$, and $w_- = w + \mu_i(\hat{y} - l)^{-1}$, asserts that there is no augmenting vector. In particular, setting $g := \hat{x} - \hat{y}$, we find $w_+g^+ - w_-g^- \leq 0$ and hence, by facts 1 and 3 above,

$$w\hat{x} - wx^* = wg \leq \mu_i\left((u - \hat{y})^{-1}g^+ + (\hat{y} - l)^{-1}g^-\right) < \frac{1}{n}n = 1 .$$

Since $w\hat{x}$ and wx^* are integer, this implies that in fact $w\hat{x} - wx^* \leq 0$ and hence the output x^* of the algorithm is indeed an optimal solution to the given optimization problem.

Next we bound the number of iterations in each phase i starting from $y_{i-1} \in S$. Let again $\hat{x} \in S$ be any optimal solution. Consider any iteration in that phase, where the oracle is queried on $\hat{y}$, $w_+ = w - \mu_i(u - \hat{y})^{-1}$, and $w_- = w + \mu_i(\hat{y} - l)^{-1}$, and returns an exhaustive augmenting vector g. We will now show that

$$w(\hat{y} + g) - w\hat{y} \geq \frac{1}{4n}(w\hat{x} - wy_{i-1}) , \qquad (1)$$

that is, the increment in the objective value from $\hat{y}$ to the augmented point $\hat{y} + g$ is at least $\frac{1}{4n}$ times the difference between the optimal objective value $w\hat{x}$ and the objective value wy_{i-1} of the point y_{i-1} at the beginning of phase i. This shows that at most $4n$ such increments (and hence iterations) can occur in the phase before it is completed.

To establish (1), we show that $wg \geq \frac{1}{2}\mu_i$ and $w\hat{x} - wy_{i-1} \leq 2n\mu_i$. For the first inequality, note that g is an exhaustive augmenting vector and so $w_+g^+ - w_-g^- > 0$ and $\hat{y} + 2g \notin S$ and hence, by facts 2 and 3, $wg > \mu_i((u - \hat{y})^{-1}g^+ + (\hat{y} - l)^{-1}g^-) > \frac{1}{2}\mu_i$. We proceed with the second inequality. If $i = 1$ (first phase) then this indeed holds since $w\hat{x} - wy_0 \leq 2n\rho\|w\|_\infty = 2n\mu_1$. If $i \geq 2$, let $\tilde{w}_+ := w - \mu_{i-1}(u - y_{i-1})^{-1}$ and $\tilde{w}_- :=$

$w + \mu_{i-1}(y_{i-1} - l)^{-1}$. The $(i-1)$th phase was completed when the oracle, queried on the triple y_{i-1}, $\tilde{w}_+$, and $\tilde{w}_-$, asserted that there is no augmenting vector. In particular, for $\tilde{g} := \hat{x} - y_{i-1}$, we find $\tilde{w}_+\tilde{g}^+ - \tilde{w}_-\tilde{g}^- \le 0$ and so, by facts 1 and 3,

$$\begin{aligned} & w\hat{x} - wy_{i-1} \\ &= w\tilde{g} \le \mu_{i-1}\left((u - y_{i-1})^{-1}\tilde{g}^+ + (y_{i-1} - l)^{-1}\tilde{g}^-\right) \\ &\le \mu_{i-1} n = 2n\mu_i \,. \end{aligned}$$

□

Graver Bases and Linear Integer Programming

We now come to the definition of a fundamental object introduced by Graver in [28]. The *Graver basis* of an integer matrix A is a canonical finite set $\mathcal{G}(A)$ that can be defined as follows. Define a partial order $\sqsubseteq$ on $\mathbb{Z}^n$ which extends the coordinate-wise order $\le$ on $\mathbb{N}^n$ as follows: for two vectors $u, v \in \mathbb{Z}^n$ put $u \sqsubseteq v$ and say that u is *conformal* to v if $|u_i| \le |v_i|$ and $u_i v_i \ge 0$ for $i = 1, \ldots, n$, that is, u and v lie in the same orthant of $\mathbb{R}^n$ and each component of u is bounded by the corresponding component of v in absolute value. It is not hard to see that $\sqsubseteq$ is a *well* partial ordering (this is basically Dickson's lemma) and hence every subset of $\mathbb{Z}^n$ has finitely-many $\sqsubseteq$-minimal elements. Let $\mathcal{L}(A) := \{x \in \mathbb{Z}^n : Ax = 0\}$ be the lattice of linear integer dependencies on A. The *Graver basis* of A is defined to be the set $\mathcal{G}(A)$ of all $\sqsubseteq$-minimal vectors in $\mathcal{L}(A) \setminus \{0\}$.

Note that if A is an $m \times n$ matrix then its Graver basis consist of vectors in $\mathbb{Z}^n$. We sometimes write $\mathcal{G}(A)$ as a suitable $|\mathcal{G}(A)| \times n$ matrix whose rows are the Graver basis elements. The Graver basis is centrally symmetric ($g \in \mathcal{G}(A)$ implies $-g \in \mathcal{G}(A)$); thus, when listing a Graver basis we will typically give one of each antipodal pair and prefix the set (or matrix) by $\pm$. Any element of the Graver basis is primitive (its entries are relatively prime integers). Every circuit of A (nonzero primitive minimal support element of $\mathcal{L}(A)$) is in $\mathcal{G}(A)$; in fact, if A is totally unimodular then $\mathcal{G}(A)$ coincides with the set of circuits (see Sect. "Convex Integer Programming over Totally Unimodular Systems" in the sequel for more details on this). However, in general $\mathcal{G}(A)$ is much larger. For more details on Graver bases and their connection to Gröbner bases see Sturmfels [58] and for the currently fastest procedure for computing them see [35,36].

Here is a quick simple example; we will see more structured and complex examples later on. Consider the 1×3 matrix $A := (1, 2, 1)$. Then its Graver basis can be shown to be the set $\mathcal{G}(A) = \pm\{(2, -1, 0), (0, -1, 2), (1, 0, -1), (1, -1, 1)\}$. The first three elements (and their antipodes) are the circuits of A; already in this small example non-circuits appear as well: the fourth element (and its antipode) is a primitive linear integer dependency whose support is not minimal.

We now show that when we do have access to the Graver basis, it can be used to solve linear integer programming. We will extend this in Sect. "Convex Integer Programming", where we show that the Graver basis enables to solve convex integer programming as well. In Sect. "Graver Bases of N-fold Matrices" we will show that there are important classes of matrices for which the Graver basis is indeed accessible.

First, we need a simple property of Graver bases. A finite sum $u := \sum_i v_i$ of vectors $v_i \in \mathbb{R}^n$ is *conformal* if each summand is conformal to the sum, that is, $v_i \sqsubseteq u$ for all i.

Lemma 8 *Let A be any integer matrix. Then any $h \in \mathcal{L}(A) \setminus \{0\}$ can be written as a conformal sum $h := \sum g_i$ of (not necessarily distinct) Graver basis elements $g_i \in \mathcal{G}(A)$.*

Proof By induction on the well partial order $\sqsubseteq$. Recall that $\mathcal{G}(A)$ is the set of $\sqsubseteq$-minimal elements in $\mathcal{L}(A) \setminus \{0\}$. Consider any $h \in \mathcal{L}(A) \setminus \{0\}$. If it is $\sqsubseteq$-minimal then $h \in \mathcal{G}(A)$ and we are done. Otherwise, there is a $h' \in \mathcal{G}(A)$ such that $h' \sqsubset h$. Set $h'' := h - h'$. Then $h'' \in \mathcal{L}(A) \setminus \{0\}$ and $h'' \sqsubset h$, so by induction there is a conformal sum $h'' = \sum_i g_i$ with $g_i \in \mathcal{G}(A)$ for all i. Now $h = h' + \sum_i g_i$ is the desired conformal sum of h. □

The next lemma shows the usefulness of Graver bases for oriented augmentation.

Lemma 9 *Let A be an $m \times n$ integer matrix with Graver basis $\mathcal{G}(A)$ and let $l, u \in \mathbb{Z}^n_\infty$, $w_+, w_- \in \mathbb{Z}^n$, and $b \in \mathbb{Z}^m$. Suppose $x \in T := \{y \in \mathbb{Z}^n : Ay = b, l \le y \le u\}$. Then for every $g \in \mathbb{Z}^n$ which satisfies $x + g \in T$ and $w_+g^+ - w_-g^- > 0$ there exists an element $\hat{g} \in \mathcal{G}(A)$ with $\hat{g} \sqsubseteq g$ which also satisfies $x + \hat{g} \in T$ and $w_+\hat{g}^+ - w_-\hat{g}^- > 0$.*

Proof Suppose $g \in \mathbb{Z}^n$ satisfies the requirements. Then $Ag = A(x+g) - Ax = b - b = 0$ since $x, x+g \in$

T. Thus, $g \in \mathcal{L}(A) \setminus \{0\}$ and hence, by Lemma 8, there is a conformal sum $g = \sum_i h_i$ with $h_i \in \mathcal{G}(A)$ for all i. Now, $h_i \sqsubseteq g$ is equivalent to $h_i^+ \leq g^+$ and $h_i^- \leq g^-$, so the conformal sum $g = \sum_i h_i$ gives corresponding sums of the positive and negative parts $g^+ = \sum_i h_i^+$ and $g^- = \sum_i h_i^-$. Therefore we obtain

$$0 < w_+ g^+ - w_- g^- = w_+ \sum_i h_i^+ - w_- \sum_i h_i^- = \sum_i (w_+ h_i^+ - w_- h_i^-)$$

which implies that there is some h_i in this sum with $w_+ h_i^+ - w_- h_i^- > 0$. Now, $h_i \in \mathcal{G}(A)$ implies $A(x + h_i) = Ax = b$. Also, $l \leq x, x + g \leq u$ and $h_i \sqsubseteq g$ imply that $l \leq x + h_i \leq u$. So $x + h_i \in T$. Therefore the vector $\hat{g} := h_i$ satisfies the claim. □

We can now show that the Graver basis enables to solve linear integer programming in polynomial time provided an initial feasible point is available.

Theorem 12 *There is a polynomial time algorithm that, given $A \in \mathbb{Z}^{m \times n}$, its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}^n_\infty$, $x, w \in \mathbb{Z}^n$ with $l \leq x \leq u$, encoded as $[\langle A, \mathcal{G}(A), l, u, x, w\rangle]$, solves the linear integer program $\max\{wz : z \in \mathbb{Z}^n, Az = b, l \leq z \leq u\}$ with $b := Ax$.*

Proof First, note that the objective function of the integer program is unbounded if and only if the objective function of its relaxation $\max\{wy : y \in \mathbb{R}^n, Ay = b, l \leq y \leq u\}$ is unbounded, which can be checked in polynomial time using linear programming. If it is unbounded then assert that there is no optimal solution and terminate the algorithm.

Assume then that the objective is bounded. Then, since the program is feasible, it has an optimal solution. Furthermore, (as basically follows from Cramer's rule, see e. g. [13, Theorem 17.1]) it has an optimal x^* satisfying $|x_j^*| \leq \rho$ for all j, where ρ is an easily computable integer upper bound whose binary length $\langle \rho \rangle$ is polynomially bounded in $\langle A, l, u, x\rangle$. For instance, $\rho := (n+1)(n+1)! r^{n+1}$ will do, with r the maximum among $\max_i |\sum_j A_{i,j} x_j|$, $\max_{i,j} |A_{i,j}|$, $\max\{|l_j| : |l_j| < \infty\}$, and $\max\{|u_j| : |u_j| < \infty\}$.

Let $T := \{y \in \mathbb{Z}^n : Ay = b,\ l \leq y \leq u\}$ and $S := T \cap [-\rho, \rho]^n$. Then our linear integer programming problem now reduces to linear discrete optimization over S. Now, an oriented augmentation oracle for S can be simulated in polynomial time using the given Graver basis $\mathcal{G}(A)$ as follows: given a query $y \in S$ and $w_+, w_- \in \mathbb{Z}^n$, search for $g \in \mathcal{G}(A)$ which satisfies $w_+ g^+ - w_- g^- > 0$ and $y + g \in S$; if there is such a g then return it as an augmenting vector, whereas if there is no such g then assert that no augmenting vector exists. Clearly, if this simulated oracle returns a vector g then it is an augmenting vector. On the other hand, if there exists an augmenting vector g then $y + g \in S \subseteq T$ and $w_+ g^+ - w_- g^- > 0$ imply by Lemma 9 that there is also a $\hat{g} \in \mathcal{G}(A)$ with $\hat{g} \sqsubseteq g$ such that $w_+ \hat{g}^+ - w_- \hat{g}^- > 0$ and $y + \hat{g} \in T$. Since $y, y + g \in S$ and $\hat{g} \sqsubseteq g$, we find that $y + \hat{g} \in S$ as well. Therefore the Graver basis contains an augmenting vector and hence the simulated oracle will find and output one.

Define $\hat{l}, \hat{u} \in \mathbb{Z}^n$ by $\hat{l}_j := \max(l_j, -\rho), \hat{u}_j := \min(u_j, \rho)$, $j = 1, \ldots, n$. Then it is easy to see that $S = \mathrm{aff}(S) \cap \{y \in \mathbb{Z}^n : \hat{l} \leq y \leq \hat{u}\}$. Now apply the algorithm of Lemma 7 to $\hat{l}, \hat{u}, S, x$, and w, using the above simulated oriented augmentation oracle for S, and obtain in polynomial time a vector $x^* \in S$ which is optimal to the linear discrete optimization problem over S and hence to the given linear integer program. □

As a special case of Theorem 12 we recover the following result of [55] concerning linear integer programming in standard form when the Graver basis is available.

Theorem 13 *There is a polynomial time algorithm that, given matrix $A \in \mathbb{Z}^{m \times n}$, its Graver basis $\mathcal{G}(A)$, $x \in \mathbb{N}^n$, and $w \in \mathbb{Z}^n$, encoded as $[\langle A, \mathcal{G}(A), x, w\rangle]$, solves the linear integer programming problem $\max\{wz : z \in \mathbb{N}^n, Az = b\}$ where $b := Ax$.*

Graver Bases of N-fold Matrices

As mentioned above, the Graver basis $\mathcal{G}(A)$ of an integer matrix A contains all circuits of A and typically many more elements. While the number of circuits is already typically exponential and can be as large as $\binom{n}{m+1}$, the number of Graver basis elements is usually even larger and depends also on the entries of A and not only on its dimensions m, n. So unfortunately it is typically very hard to compute $\mathcal{G}(A)$. However, we now show that for the important and useful broad class of n-fold matrices, the Graver basis is better behaved and can be computed in polynomial time. Recall the following definition from the introduction. Given an $(r + s) \times t$

matrix A, let A_1 be its $r \times t$ sub-matrix consisting of the first r rows and let A_2 be its $s \times t$ sub-matrix consisting of the last s rows. We refer to A explicitly as $(r+s) \times t$ *matrix*, since the definition below depends also on r and s and not only on the entries of A. The *n-fold matrix* of an $(r+s) \times t$ matrix A is then defined to be the following $(r+ns) \times nt$ matrix,

$$A^{(n)} := (\mathbf{1}_n \otimes A_1) \oplus (I_n \otimes A_2) = \begin{pmatrix} A_1 & A_1 & A_1 & \cdots & A_1 \\ A_2 & 0 & 0 & \cdots & 0 \\ 0 & A_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & A_2 \end{pmatrix}.$$

We now discuss a recent result of [54], which originates in [4], and its extension in [38], on the stabilization of Graver bases of n-fold matrices. Consider vectors $x = (x^1, \ldots, x^n)$ with $x^k \in \mathbb{Z}^t$ for $k = 1, \ldots, n$. The *type* of x is the number $|\{k : x^k \neq 0\}|$ of nonzero components $x^k \in \mathbb{Z}^t$ of x. The *Graver complexity* of an $(r+s) \times t$ matrix, denoted $c(A)$, is defined to be the smallest $c \in \mathbb{N} \uplus \{\infty\}$ such that for all n, the Graver basis of $A^{(n)}$ consists of vectors of type at most $c(A)$. We provide the proof of the following result of [38,54] stating that the Graver complexity is always finite.

Lemma 10 *The Graver complexity $c(A)$ of any $(r+s) \times t$ integer matrix A is finite.*

Proof Call an element $x = (x^1, \ldots, x^n)$ in the Graver basis of some $A^{(n)}$ *pure* if $x^i \in \mathcal{G}(A_2)$ for all i. Note that the type of a pure $x \in \mathcal{G}(A^{(n)})$ is n. First, we claim that if there is an element of type m in some $\mathcal{G}(A^{(l)})$ then for some $n \geq m$ there is a pure element in $\mathcal{G}(A^{(n)})$, and so it will suffice to bound the type of pure elements. Suppose there is an element of type m in some $\mathcal{G}(A^{(l)})$. Then its restriction to its m nonzero components is an element $x = (x^1, \ldots, x^m)$ in $\mathcal{G}(A^{(m)})$. Let $x^i = \sum_{j=1}^{k_i} g_{i,j}$ be a conformal decomposition of x^i with $g_{i,j} \in \mathcal{G}(A_2)$ for all i, j, and let $n := k_1 + \cdots + k_m \geq m$. Then $g := (g_{1,1}, \ldots, g_{m,k_m})$ is in $\mathcal{G}(A^{(n)})$, else there would be $\hat{g} \sqsubset g$ in $\mathcal{G}(A^{(n)})$ in which case the nonzero $\hat{x}$ with $\hat{x}^i := \sum_{j=1}^{k_i} \hat{g}_{i,j}$ for all i would satisfy $\hat{x} \sqsubset x$ and $\hat{x} \in \mathcal{L}(A^{(m)})$, contradicting $x \in \mathcal{G}(A^{(m)})$. Thus g is a pure element of type $n \geq m$, proving the claim.

We proceed to bound the type of pure elements. Let $\mathcal{G}(A_2) = \{g_1, \ldots, g_m\}$ be the Graver basis of A_2 and let G_2 be the $t \times m$ matrix whose columns are the g_i. Suppose $x = (x^1, \ldots, x^n) \in \mathcal{G}(A^{(n)})$ is pure for some n. Let $v \in \mathbb{N}^m$ be the vector with $v_i := |\{k : x^k = g_i\}|$ counting the number of g_i components of x for each i. Then $\sum_{i=1}^m v_i$ is equal to the type n of x. Next, note that $A_1 G_2 v = A_1(\sum_{k=1}^n x^k) = 0$ and hence $v \in \mathcal{L}(A_1 G_2)$. We claim that, moreover, $v \in \mathcal{G}(A_1 G_2)$. Suppose indirectly not. Then there is $\hat{v} \in \mathcal{G}(A_1 G_2)$ with $\hat{v} \sqsubset v$, and it is easy to obtain a nonzero $\hat{x} \sqsubset x$ from x by zeroing out some components so that $\hat{v}_i = |\{k : \hat{x}^k = g_i\}|$ for all i. Then $A_1(\sum_{k=1}^n \hat{x}^k) = A_1 G_2 \hat{v} = 0$ and hence $\hat{x} \in \mathcal{L}(A^{(n)})$, contradicting $x \in \mathcal{G}(A^{(n)})$.

So the type of any pure element, and hence the Graver complexity of A, is at most the largest value $\sum_{i=1}^m v_i$ of any nonnegative element v of the Graver basis $\mathcal{G}(A_1 G_2)$. □

Using Lemma 10 we now show how to compute $\mathcal{G}(A^{(n)})$ in polynomial time.

Theorem 14 *For every fixed $(r+s) \times t$ integer matrix A there is a strongly polynomial time algorithm that, given $n \in \mathbb{N}$, encoded as $[n; n]$, computes the Graver basis $\mathcal{G}(A^{(n)})$ of the n-fold matrix $A^{(n)}$. In particular, the cardinality $|\mathcal{G}(A^{(n)})|$ and binary length $\langle \mathcal{G}(A^{(n)}) \rangle$ of the Graver basis of the n-fold matrix are polynomially bounded in n.*

Proof Let $c := c(A)$ be the Graver complexity of A and consider any $n \geq c$. We show that the Graver basis of $A^{(n)}$ is the union of $\binom{n}{c}$ suitably embedded copies of the Graver basis of $A^{(c)}$. For every c indices $1 \leq k_1 < \cdots < k_c \leq n$ define a map $\phi_{k_1,\ldots,k_c}$ from $\mathbb{Z}^{ct}$ to $\mathbb{Z}^{nt}$ sending $x = (x^1, \ldots, x^c)$ to $y = (y^1, \ldots, y^n)$ with $y^{k_i} := x^i$ for $i = 1, \ldots, c$ and $y^k := 0$ for $k \notin \{k_1, \ldots, k_c\}$. We claim that $\mathcal{G}(A^{(n)})$ is the union of the images of $\mathcal{G}(A^{(c)})$ under the $\binom{n}{c}$ maps $\phi_{k_1,\ldots,k_c}$ for all $1 \leq k_1 < \cdots < k_c \leq n$, that is,

$$\mathcal{G}(A^{(n)}) = \bigcup_{1 \leq k_1 < \cdots < k_c \leq n} \phi_{k_1,\ldots,k_c}(\mathcal{G}(A^{(c)})) . \quad (2)$$

If $x = (x^1, \ldots, x^c) \in \mathcal{G}(A^{(c)})$ then x is a $\sqsubseteq$-minimal nonzero element of $\mathcal{L}(A^{(c)})$, implying that $\phi_{k_1,\ldots,k_c}(x)$ is a $\sqsubseteq$-minimal nonzero element of $\mathcal{L}(A^{(n)})$ and therefore we have $\phi_{k_1,\ldots,k_c}(x) \in \mathcal{G}(A^{(n)})$. So the right-hand side of (2) is contained in the left-hand side. Conversely, consider any $y \in \mathcal{G}(A^{(n)})$. Then, by Lemma 10, the type of y is at most c, so there are indices $1 \leq k_1 < \cdots < k_c \leq n$

such that all nonzero components of y are among those of the reduced vector $x := (y^{k_1}, \ldots, y^{k_c})$ and therefore $y = \phi_{k_1,\ldots,k_c}(x)$. Now, $y \in \mathcal{G}(A^{(n)})$ implies that y is a $\sqsubseteq$-minimal nonzero element of $\mathcal{L}(A^{(n)})$ and hence x is a $\sqsubseteq$-minimal nonzero element of $\mathcal{L}(A^{(c)})$. Therefore $x \in \mathcal{G}(A^{(c)})$ and $y \in \phi_{k_1,\ldots,k_g}(\mathcal{G}(A^{(c)}))$. So the left-hand side of (2) is contained in the right-hand side.

Since A is fixed we have that $c = c(A)$ and $\mathcal{G}(A^{(c)})$ are constant. Then (2) implies that $|\mathcal{G}(A^{(n)})| \leq \binom{n}{c}|\mathcal{G}(A^{(c)})| = O(n^c)$. Moreover, every element of $\mathcal{G}(A^{(n)})$ is an nt-dimensional vector $\phi_{k_1,\ldots,k_c}(x)$ obtained by appending zero components to some $x \in \mathcal{G}(A^{(c)})$ and hence has linear binary length $O(n)$. So the binary length of the entire Graver basis $\mathcal{G}(A^{(n)})$ is $O(n^{c+1})$. Thus, the $\binom{n}{c} = O(n^c)$ images $\phi_{k_1,\ldots,k_c}(\mathcal{G}(A^{(c)}))$ and their union $\mathcal{G}(A^{(n)})$ can be computed in strongly polynomial time, as claimed. □

Example 2 Consider the $(2+1)\times 2$ matrix A with $A_1 := I_2$ the 2×2 identity and $A_2 := (1,1)$. Then $\mathcal{G}(A_2) = \pm(1,-1)$ and $\mathcal{G}(A_1 G_2) = \pm(1,1)$ from which the Graver complexity of A can be concluded to be $c(A) = 2$ (see the proof of Lemma 10). The 2-fold matrix of A and its Graver basis, consisting of two antipodal vectors only, are

$$A^{(2)} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix},$$

$$\mathcal{G}(A^{(2)}) = \pm \begin{pmatrix} 1 & -1 & -1 & 1 \end{pmatrix}.$$

By Theorem 14, the Graver basis of the 4-fold matrix $A^{(4)}$ is computed to be the union of the images of the $6 = \binom{4}{2}$ maps $\phi_{k_1,k_2} : \mathbb{Z}^{2\cdot 2} \longrightarrow \mathbb{Z}^{4\cdot 2}$ for $1 \leq k_1 < k_2 \leq 4$, getting

$$A^{(4)} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

$$\mathcal{G}(A^{(4)}) = \pm \begin{pmatrix} 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \end{pmatrix}.$$

Linear N-fold Integer Programming in Polynomial Time

We now proceed to provide a polynomial time algorithm for linear integer programming over n-fold matrices. First, combining the results of Sect. "Graver Bases and Linear Integer Programming" and Sect. "Graver Bases of N-fold Matrices", we get at once the following polynomial time algorithm for converting any feasible point to an optimal one.

Lemma 11 *For every fixed $(r+s)\times t$ integer matrix A there is a polynomial time algorithm that, given $n \in \mathbb{N}$, $l, u \in \mathbb{Z}^{nt}_{\infty}$, $x, w \in \mathbb{Z}^{nt}$ satisfying $l \leq x \leq u$, encoded as $[\langle l, u, x, w\rangle]$, solves the linear n-fold integer programming problem with $b := A^{(n)}x$,*

$$\max\{wz \,:\, z \in \mathbb{Z}^{nt},\ A^{(n)}z = b,\ l \leq z \leq u\}.$$

Proof First, apply the polynomial time algorithm of Theorem 14 and compute the Graver basis $\mathcal{G}(A^{(n)})$ of the n-fold matrix $A^{(n)}$. Then apply the polynomial time algorithm of Theorem 12 to the data $A^{(n)}$, $\mathcal{G}(A^{(n)})$, l, u, x and w. □

Next we show that an initial feasible point can also be found in polynomial time.

Lemma 12 *For every fixed $(r+s)\times t$ integer matrix A there is a polynomial time algorithm that, given $n \in \mathbb{N}$, $l, u \in \mathbb{Z}^{nt}_{\infty}$, and $b \in \mathbb{Z}^{r+ns}$, encoded as $[\langle l, u, b\rangle]$, either finds an $x \in \mathbb{Z}^{nt}$ satisfying $l \leq x \leq u$ and $A^{(n)}x = b$ or asserts that none exists.*

Proof If $l \not\leq u$ then assert that there is no feasible point and terminate the algorithm. Assume then that $l \leq u$ and determine some $x \in \mathbb{Z}^{nt}$ with $l \leq x \leq u$ and $\langle x\rangle \leq \langle l, u\rangle$. Now, introduce $n(2r+2s)$ auxiliary variables to the given n-fold integer program and denote by $\hat{x}$ the resulting vector of $n(t+2r+2s)$ variables. Suitably extend the lower and upper bound vectors to $\hat{l}, \hat{u}$ by setting $\hat{l}_j := 0$ and $\hat{u}_j := \infty$ for each auxiliary variable $\hat{x}_j$. Consider the auxiliary integer program of finding an integer vector $\hat{x}$ that minimizes the sum of auxiliary variables subject to the lower and upper bounds $\hat{l} \leq \hat{x} \leq \hat{u}$ and the following system of equations, with I_r and I_s the

$r \times r$ and $s \times s$ identity matrices,

$$\begin{pmatrix} A_1 & I_r & -I_r & 0 & 0 & A_1 & I_r & -I_r & 0 \\ A_2 & 0 & 0 & I_s & -I_s & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A_2 & 0 & 0 & I_s \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & \cdots & A_1 & I_r & -I_r & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ -I_s & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & A_2 & 0 & 0 & I_s & -I_s \end{pmatrix} \hat{x} = b.$$

This is again an n-fold integer program, with an $(r+s)\times(t+2r+2s)$ matrix $\hat{A}$, where $\hat{A}_1 = (A_1, I_r, -I_r, 0, 0)$ and $\hat{A}_2 = (A_2, 0, 0, I_s, -I_s)$. Since A is fixed, so is $\hat{A}$. It is now easy to extend the vector $x \in \mathbb{Z}^{nt}$ determined above to a feasible point $\hat{x}$ of the auxiliary program. Indeed, put $\hat{b} := b - A^{(n)}x \in \mathbb{Z}^{r+ns}$; now, for $i = 1, \ldots, r+ns$, simply choose an auxiliary variable $\hat{x}_j$ appearing only in the ith equation, whose coefficient equals the sign $\text{sign}(\hat{b}_i)$ of the corresponding entry of $\hat{b}$, and set $\hat{x}_j := |\hat{b}_i|$. Define $\hat{w} \in \mathbb{Z}^{n(t+2r+2s)}$ by setting $\hat{w} := 0$ for each original variable and $\hat{w} := -1$ for each auxiliary variable, so that maximizing $\hat{w}\hat{x}$ is equivalent to minimizing the sum of auxiliary variables. Now solve the auxiliary linear integer program in polynomial time by applying the algorithm of Lemma 11 corresponding to $\hat{A}$ to the data $n, \hat{l}, \hat{u}, \hat{x}$, and $\hat{w}$. Since the auxiliary objective $\hat{w}\hat{x}$ is bounded above by zero, the algorithm will output an optimal solution $\hat{x}^*$. If the optimal objective value is negative, then the original n-fold program is infeasible, whereas if the optimal value is zero, then the restriction of $\hat{x}^*$ to the original variables is a feasible point x^* of the original integer program. □

Combining Lemmas 11 and 12 we get at once the main result of this section.

Theorem 15 *For every fixed $(r+s)\times t$ integer matrix A there is a polynomial time algorithm that, given n, lower and upper bounds $l, u \in \mathbb{Z}^{nt}_{\infty}$, $w \in \mathbb{Z}^{nt}$, and $b \in \mathbb{Z}^{r+ns}$, encoded as $[\langle l, u, w, b\rangle]$, solves the following linear n-fold integer programming problem,*

$$\max\{wx : x \in \mathbb{Z}^{nt},\ A^{(n)}x = b,\ l \le x \le u\}.$$

Again, as a special case of Theorem 15 we recover the following result of [13] concerning linear integer programming in standard form over n-fold matrices.

Theorem 16 *For every fixed $(r+s)\times t$ integer matrix A there is a polynomial time algorithm that, given n, linear functional $w \in \mathbb{Z}^{nt}$, and right-hand side $b \in \mathbb{Z}^{r+ns}$, encoded as $[\langle w, b\rangle]$, solves the following linear n-fold integer program in standard form,*

$$\max\left\{wx : x \in \mathbb{N}^{nt},\ A^{(n)}x = b\right\}.$$

Some Applications

Three-Way Line-Sum Transportation Problems Transportation problems form a very important class of discrete optimization problems studied extensively in the operations research and mathematical programming literature, see e. g. [6,42,43,53,60,62] and the references therein. We will discuss this class of problem and its applications to secure statistical data disclosure in more detail in Sect. "Multiway Transportation Problems and Privacy in Statistical Databases".

It is well known that 2-way transportation problems are polynomial time solvable, since they can be encoded as linear integer programs over totally unimodular systems. However, already 3-way transportation problem are much more complicated. Consider the following 3-way transportation problem over $p\times q\times n$ tables with all line-sums fixed,

$$\max\left\{wx : x \in \mathbb{N}^{p\times q\times n},\ \sum_i x_{i,j,k} = z_{j,k},\ \sum_j x_{i,j,k} = v_{i,k},\ \sum_k x_{i,j,k} = u_{i,j}\right\}.$$

The data for the problem consist of given integer numbers (lines-sums) $u_{i,j}$, $v_{i,k}$, $z_{j,k}$ for $i = 1, \ldots, p$, $j = 1, \ldots, q$, $k = 1, \ldots, n$, and a linear functional given by a $p \times q \times n$ integer array w representing the transportation profit per unit on each cell. The problem is to find a transportation, that is, a $p \times q \times n$ nonnegative integer table x satisfying the line sum constraints, which attains maximum profit $wx = \sum_{i=1}^{p}\sum_{j=1}^{q}\sum_{k=1}^{n} w_{i,j,k}x_{i,j,k}$.

When at least two of the table sides, say p, q, are variable part of the input, and even when the third side is fixed and as small as $n = 3$, this problem is already *universal* for integer programming in a very

strong sense [14,16], and in particular is NP-hard [15]; this will be discussed in detail and proved in Sect. "Multiway Transportation Problems and Privacy in Statistical Databases". We now show that in contrast, when two sides, say p, q, are fixed (but arbitrary), and one side n is variable, then the 3-way transportation problem over such *long* tables is an n-fold integer programming problem and therefore, as a consequence of Theorem 16, can be solved is polynomial time.

Corollary 8 *For every fixed p and q there is a polynomial time algorithm that, given n, integer profit array $w \in \mathbb{Z}^{p\times q\times n}$, and line-sums $u \in \mathbb{Z}^{p\times q}$, $v \in \mathbb{Z}^{p\times n}$ and $z \in \mathbb{Z}^{q\times n}$, encoded as $[\langle w, u, v, z\rangle]$, solves the integer 3-way line-sum transportation problem*

$$\max\left\{ wx \,:\, x \in \mathbb{N}^{p\times q\times n}\,,\ \sum_i x_{i,j,k} = z_{j,k}\,,\ \sum_j x_{i,j,k} = v_{i,k}\,,\ \sum_k x_{i,j,k} = u_{i,j} \right\}.$$

Proof Re-index $p \times q \times n$ arrays as $x = (x^1, \ldots, x^n)$ with each component indexed as $x^k := (x^k_{i,j}) := (x_{1,1,k}, \ldots, x_{p,q,k})$ suitably indexed as a pq vector representing the kth layer of x. Put $r := t := pq$ and $s := p + q$, and let A be the $(r + s) \times t$ matrix with $A_1 := I_{pq}$ the $pq \times pq$ identity and with A_2 the $(p+q)\times pq$ matrix of equations of the usual 2-way transportation problem for $p\times q$ arrays. Re-arrange the given line-sums in a vector $b := (b^0, b^1, \ldots, b^n) \in \mathbb{Z}^{r+ns}$ with $b^0 := (u_{i,j})$ and $b^k := ((v_{i,k}), (z_{j,k}))$ for $k = 1, \ldots, n$.

This translates the given 3-way transportation problem into an n-fold integer programming problem in standard form,

$$\max\{wx \,:\, x \in \mathbb{N}^{nt},\ A^{(n)}x = b\}\,,$$

where the equations $A_1(\sum_{k=1}^n x^k) = b^0$ represent the constraints $\sum_k x_{i,j,k} = u_{i,j}$ of all line-sums where summation over layers occurs, and the equations $A_2 x^k = b^k$ for $k = 1, \ldots, n$ represent the constraints $\sum_i x_{i,j,k} = z_{j,k}$ and $\sum_j x_{i,j,k} = v_{i,k}$ of all line-sums where summations are within a single layer at a time.

Using the algorithm of Theorem 16, this n-fold integer program, and hence the given 3-way transportation problem, can be solved in polynomial time. □

Example 3 We demonstrate the encoding of the $p\times q\times n$ transportation problem as an n-fold integer program as in the proof of Corollary 8 for $p = q = 3$ (smallest case where the problem is genuinely 3-dimensional). Here we put $r := t := 9$, $s := 6$, write

$$x^k := (x_{1,1,k}, x_{1,2,k}, x_{1,3,k}, x_{2,1,k}, x_{2,2,k}, x_{2,3,k}, x_{3,1,k}, x_{3,2,k}, x_{3,3,k})\,,\quad k = 1, \ldots, n\,,$$

and let the $(9 + 6) \times 9$ matrix A consist of $A_1 = I_9$ the 9×9 identity matrix and

$$A_2 := \begin{pmatrix} 1&1&1&0&0&0&0&0&0\\ 0&0&0&1&1&1&0&0&0\\ 0&0&0&0&0&0&1&1&1\\ 1&0&0&1&0&0&1&0&0\\ 0&1&0&0&1&0&0&1&0\\ 0&0&1&0&0&1&0&0&1 \end{pmatrix}.$$

Then the corresponding n-fold integer program encodes the $3 \times 3 \times n$ transportation problem as desired. Already for this case, of $3\times 3\times n$ tables, the only known polynomial time algorithm for the transportation problem is the one underlying Corollary 8 .

Corollary 8 has a very broad generalization to multiway transportation problems over long k-way tables of any dimension k; this will be discussed in detail in Sect. "Multiway Transportation Problems and Privacy in Statistical Databases".

Packing Problems and Cutting-Stock We consider the following rather general class of packing problems which concern maximum utility packing of many items of several types in various bins subject to weight constraints. More precisely, the data is as follows. There are t types of items. Each item of type j has integer weight v_j. There are n_j items of type j to be packed. There are n bins. The weight capacity of bin k is an integer u_k. Finally, there is a utility matrix $w \in \mathbb{Z}^{t\times n}$ where $w_{j,k}$ is the utility of packing one item of type j in bin k. The problem is to find a feasible packing of maximum total utility. By incrementing the number t of types by 1 and suitably augmenting the data, we may assume that the last type t represents "slack items" which occupy the unused capacity in each bin, where the weight of each slack item is 1, the utility of packing any slack item in any bin is 0, and the number of slack items is the total residual weight capacity $n_t := \sum_{k=1}^n u_k - \sum_{j=1}^{t-1} n_j v_j$. Let $x \in \mathbb{N}^{t\times n}$ be a variable matrix where $x_{j,k}$ represents

the number of items of type j to be packed in bin k. Then the packing problem becomes the following linear integer program,

$$\max\left\{ wx \,:\, x \in \mathbb{N}^{t\times n},\ \sum_j v_j x_{j,k} = u_k,\ \sum_k x_{j,k} = n_j \right\}.$$

We now show that this is in fact an n-fold integer programming problem and therefore, as a consequence of Theorem 16, can be solved is polynomial time. While the number t of types and type weights v_j are fixed, which is natural in many bin packing applications, the numbers n_j of items of each type and the bin capacities u_k may be very large.

Corollary 9 *For every fixed number t of types and integer type weights $v_1, \ldots, v_t$, there is a polynomial time algorithm that, given n bins, integer item numbers $n_1, \ldots, n_t$, integer bin capacities $u_1, \ldots, u_n$, and $t \times n$ integer utility matrix w, encoded as $[\langle n_1, \ldots, n_t, u_1, \ldots, u_n, w\rangle]$, solves the following integer bin packing problem,*

$$\max\left\{ wx \,:\, x \in \mathbb{N}^{t\times n},\ \sum_j v_j x_{j,k} = u_k,\ \sum_k x_{j,k} = n_j \right\}.$$

Proof Re-index the variable matrix as $x = (x^1, \ldots, x^n)$ with $x^k := (x_1^k, \ldots, x_t^k)$ where x_j^k represents the number of items of type j to be packed in bin k for all j and k. Let A be the $(t+1) \times t$ matrix with $A_1 := I_t$ the $t \times t$ identity and with $A_2 := (v_1, \ldots, v_t)$ a single row. Re-arrange the given item numbers and bin capacities in a vector $b := (b^0, b^1, \ldots, b^n) \in \mathbb{Z}^{t+n}$ with $b^0 := (n_1, \ldots, n_t)$ and $b^k := u_k$ for all k. This translates the bin packing problem into an n-fold integer programming problem in standard form,

$$\max\{wx \,:\, x \in \mathbb{N}^{nt},\ A^{(n)}x = b\},$$

where the equations $A_1(\sum_{k=1}^n x^k) = b^0$ represent the constraints $\sum_k x_{j,k} = n_j$ assuring that all items of each type are packed, and the equations $A_2 x^k = b^k$ for $k = 1, \ldots, n$ represent the constraints $\sum_j v_j x_{j,k} = u_k$ assuring that the weight capacity of each bin is not exceeded (in fact, the slack items make sure each bin is perfectly packed).

Using the algorithm of Theorem 16, this n-fold integer program, and hence the given integer bin packing problem, can be solved in polynomial time. □

Example 5 (cutting-stock problem). This is a classical manufacturing problem [27], where the usual setup is as follows: a manufacturer produces rolls of material (such as scotch-tape or band-aid) in one of t different widths $v_1, \ldots, v_t$. The rolls are cut out from standard rolls of common large width u. Given orders by customers for n_j rolls of width v_j, the problem facing the manufacturer is to meet the orders using the smallest possible number of standard rolls. This can be cast as a bin packing problem as follows. Rolls of width v_j become items of type j to be packed. Standard rolls become identical bins, of capacity $u_k := u$ each, where the number of bins is set to be $n := \sum_{j=1}^t \lceil n_j / \lfloor u/v_j \rfloor \rceil$ which is sufficient to accommodate all orders. The utility of each roll of width v_j is set to be its width negated $w_{j,k} := -v_j$ regardless of the standard roll k from which it is cut (paying for the width it takes). Introduce a new roll width $v_0 := 1$, where rolls of that width represent "slack rolls" which occupy the unused width of each standard roll, with utility $w_{0,k} := -1$ regardless of the standard roll k from which it is cut (paying for the unused width it represents), with the number of slack rolls set to be the total residual width $n_0 := nu - \sum_{j=1}^t n_j v_j$. Then the cutting-stock problem becomes a bin packing problem and therefore, by Corollary 9, for every fixed t and fixed roll widths $v_1, \ldots, v_t$, it is solvable in time polynomial in $\sum_{j=1}^t \lceil n_j / \lfloor u/v_j \rfloor \rceil$ and $\langle n_1, \ldots, n_t, u\rangle$.

One common approach to the cutting-stock problem uses so-called *cutting patterns*, which are feasible solutions of the knapsack problem $\{y \in \mathbb{N}^t : \sum_{j=1}^t v_j y_j \le u\}$. This is useful when the common width u of the standard rolls is of the same order of magnitude as the demand role widths v_j. However, when u is much larger than the v_j, the number of cutting patterns becomes prohibitively large to handle. But then the values $\lfloor u/v_j \rfloor$ are large and hence $n := \sum_{j=1}^t \lceil n_j / \lfloor u/v_j \rfloor \rceil$ is small, in which case the solution through the algorithm of Corollary 9 becomes particularly appealing.

Convex Integer Programming

In this section we discuss convex integer programming. In particular, we extend the theory of Sect. "Linear N-fold Integer Programming" and show that convex n-fold integer programming is polynomial time solvable as well. In Sect. "Convex Integer Programming over Totally Unimodular Systems" we discuss convex integer programming over totally unimodular matrices. In Sect. "Graver Bases and Convex Integer Programming" we show the applicability of Graver bases to convex integer programming. In Sect. "Convex N-fold Integer Programming in Polynomial Time" we combine Theorem 6, the results of Sect. "Linear N-fold Integer Programming", and the preparatory facts from Sect. "Graver Bases and Convex Integer Programming", and prove the main result of this section, asserting that convex n-fold integer programming is polynomial time solvable. We conclude with some applications in Sect. "Some Applications".

As in Sect. "Linear N-fold Integer Programming", the feasible set S is presented as the set of integer points satisfying an explicitly given system of linear inequalities, given in one of the forms

$$S := \{x \in \mathbb{N}^n : Ax = b\} \quad \text{or}$$
$$S := \{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\}\ ,$$

with matrix $A \in \mathbb{Z}^{m\times n}$, right-hand side $b \in \mathbb{Z}^m$, and lower and upper bounds $l, u \in \mathbb{Z}^n_\infty$.

As demonstrated in Sect. "Limitations", if the polyhedron $P := \{x \in \mathbb{R}^n : Ax = b,\ l \le x \le u\}$ is unbounded then the convex integer programming problem with an oracle presented convex functional is rather hopeless. Therefore, an algorithm that solves the convex integer programming problem should either return an optimal solution, or assert that the program is infeasible, or assert that the underlying polyhedron is unbounded.

Nonetheless, we do allow the lower and upper bounds l, u to lie in $\mathbb{Z}^n_\infty$ rather than $\mathbb{Z}^n$, since often the polyhedron is bounded even though the variables are not bounded explicitly (for instance, if each variable is bounded below only, and appears in some equation all of whose coefficients are positive). This results in broader formulation flexibility. Furthermore, in the next subsections we prove auxiliary lemmas asserting that certain sets cover all edge-directions of relevant polyhedra, which do hold also in the unbounded case. So we now extend the notion of edge-directions, defined in Sect. "Edge-Directions and Zonotopes" for polytopes, to polyhedra. A *direction* of an edge (1-dimensional face) e of a polyhedron P is any nonzero scalar multiple of $y - x$ where x, y are any two distinct points in e. As before, a set *covers all edge-directions of* P if it contains a direction of each edge of P.

Convex Integer Programming over Totally Unimodular Systems

A matrix A is *totally unimodular* if the determinant of every square submatrix of A lies in $\{-1, 0, 1\}$. Such matrices arise naturally in network flows, ordinary (2-way) transportation problems, and many other situations. A fundamental result in integer programming [37] asserts that polyhedra defined by totally unimodular matrices are integer. More precisely, if A is an $m \times n$ totally unimodular matrix, $l, u \in \mathbb{Z}^n_\infty$, and $b \in \mathbb{Z}^m$, then

$$\begin{aligned} P_I &:= \operatorname{conv}\{x \in \mathbb{Z}^n : Ax = b,\ l \le x \le u\} \\ &= \{x \in \mathbb{R}^n : Ax = b,\ l \le x \le u\} := P\ , \end{aligned}$$

that is, the underlying polyhedron P coincides with its integer hull P_I. This has two consequences useful in facilitating the solution of the corresponding convex integer programming problem via the algorithm of Theorem 6. First, the corresponding linear integer programming problem can be solved by linear programming over P in polynomial time. Second, a set covering all edge-directions of the implicitly given integer hull P_I, which is typically very hard to determine, is obtained here as a set covering all edge-directions of P which is explicitly given and hence easier to determine.

We now describe a well known property of polyhedra of the above form. A *circuit* of a matrix $A \in \mathbb{Z}^{m\times n}$ is a nonzero primitive minimal support element of $\mathcal{L}(A)$. So a circuit is a nonzero $c \in \mathbb{Z}^n$ satisfying $Ac = 0$, whose entries are relatively prime integers, such that no nonzero c' with $Ac' = 0$ has support strictly contained in the support of c.

Lemma 13 *For every $A \in \mathbb{Z}^{m\times n}$, $l, u \in \mathbb{Z}^n_\infty$, and $b \in \mathbb{Z}^m$, the set of circuits of A covers all edge-directions of the polyhedron $P := \{x \in \mathbb{R}^n : Ax = b,\ l \le x \le u\}$.*

Proof Consider any edge e of P. Pick two distinct points $x, y \in e$ and set $g := y - x$. Then $Ag = 0$

and therefore, as can be easily proved by induction on $|\mathrm{supp}(g)|$, there is a finite decomposition $g = \sum_i \alpha_i c_i$ with α_i positive real number and c_i circuit of A such that $\alpha_i c_i \sqsubseteq g$ for all i, where $\sqsubseteq$ is the natural extension from $\mathbb{Z}^n$ to $\mathbb{R}^n$ of the partial order defined in Sect. "Graver Bases and Linear Integer Programming". We claim that $x + \alpha_i c_i \in P$ for all i. Indeed, c_i being a circuit implies $A(x + \alpha_i c_i) = Ax = b$; and $l \le x, x + g \le u$ and $\alpha_i c_i \sqsubseteq g$ imply $l \le x + \alpha_i c_i \le u$.

Now let $w \in \mathbb{R}^n$ be a linear functional uniquely maximized over P at the edge e. Then $w\alpha_i c_i = w(x + \alpha_i c_i) - wx \le 0$ for all i. But $\sum(w\alpha_i c_i) = wg = wy - wx = 0$, implying that in fact $w\alpha_i c_i = 0$ and hence $x + \alpha_i c_i \in e$ for all i. This implies that each c_i is a direction of e (in fact, all c_i are the same and g is a multiple of some circuit). □

Combining Theorem 6 and Lemma 13 we obtain the following statement.

Theorem 17 *For every fixed d there is a polynomial time algorithm that, given $m \times n$ totally unimodular matrix A, set $C \subset \mathbb{Z}^n$ containing all circuits of A, vectors $l, u \in \mathbb{Z}^n_\infty$, $b \in \mathbb{Z}^m$, and $w_1, \ldots, w_d \in \mathbb{Z}^n$, and convex $c : \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle A, C, l, u, b, w_1, \ldots, w_d\rangle]$, solves the convex integer program*

$$\max\{c(w_1 x, \ldots, w_d x) \,:\, x \in \mathbb{Z}^n, \; Ax = b, \; l \le x \le u\}.$$

Proof First, check in polynomial time using linear programming whether the objective function of any of the following $2n$ linear programs is unbounded,

$$\max\{\pm y_i : y \in P\}, \; i = 1, \ldots, n, \qquad P := \{y \in \mathbb{R}^n \,:\, Ay = b, \; l \le y \le u\}.$$

If any is unbounded then terminate, asserting that P is unbounded. Otherwise, let ρ be the least integer upper bound on the absolute value of all optimal objective values. Then $P \subseteq [-\rho, \rho]^n$ and $S := \{y \in \mathbb{Z}^n : Ay = b, \; l \le y \le u\} \subset P$ is finite of radius $\rho(S) \le \rho$. In fact, since A is totally unimodular, $P_I = P = \mathrm{conv}(S)$ and hence $\rho(S) = \rho$. Moreover, by Cramer's rule, $\langle\rho\rangle$ is polynomially bounded in $\langle A, l, u, x\rangle$.

Now, since A is totally unimodular, using linear programming over $P_I = P$ we can simulate in polynomial time a linear discrete optimization oracle for S. By Lemma 13, the given set C, which contains all circuits of A, also covers all edge-directions of $\mathrm{conv}(S) = P_I = P$. Therefore we can apply the algorithm of Theorem 6 and solve the given convex n-fold integer programming problem in polynomial time. □

While the number of circuits of an $m \times n$ matrix A can be as large as $2\binom{n}{m+1}$ and hence exponential in general, it is nonetheless relatively small in that it is bounded in terms of m and n only and is independent of the matrix A itself. Furthermore, it may happen that the number of circuits is much smaller than the upper bound $2\binom{n}{m+1}$. Also, if in a class of matrices, m grows slowly in terms of n, say $m = O(\log n)$, then this bound is subexponential. In such situations, the above theorem may provide a good strategy for solving convex integer programming over totally unimodular systems.

Graver Bases and Convex Integer Programming

We now extend the statements of Sect. "Convex Integer Programming over Totally Unimodular Systems" about totally unimodular matrices to arbitrary integer matrices. The next lemma shows that the Graver basis of any integer matrix covers all edge-directions of the integer hulls of polyhedra defined by that matrix.

Lemma 14 *For every $A \in \mathbb{Z}^{m\times n}$, $l, u \in \mathbb{Z}^n_\infty$, and $b \in \mathbb{Z}^m$, the Graver basis $\mathcal{G}(A)$ of A covers all edge-directions of the polyhedron $P_I := \mathrm{conv}\{x \in \mathbb{Z}^n : Ax = b, \; l \le x \le u\}$.*

Proof Consider any edge e of P_I and pick two distinct points $x, y \in e \cap \mathbb{Z}^n$. Then $g := y - x$ is in $\mathcal{L}(A) \setminus \{0\}$. Therefore, by Lemma 8, there is a conformal sum $g = \sum_i h_i$ with $h_i \in \mathcal{G}(A)$ for all i. We claim that $x + h_i \in P_I$ for all i. Indeed, first note that $h_i \in \mathcal{G}(A) \subset \mathcal{L}(A)$ implies $Ah_i = 0$ and hence $A(x + h_i) = Ax = b$; and second note that $l \le x, x + g \le u$ and $h_i \sqsubseteq g$ imply that $l \le x + h_i \le u$.

Now let $w \in \mathbb{Z}^n$ be a linear functional uniquely maximized over P_I at the edge e. Then $wh_i = w(x + h_i) - wx \le 0$ for all i. But $\sum(wh_i) = wg = wy - wx = 0$, implying that in fact $wh_i = 0$ and hence $x + h_i \in e$ for all i. Therefore each h_i is a direction of e (in fact, all h_i are the same and g is a multiple of some Graver basis element). □

Combining Theorems 6 and 12 and Lemma 14 we obtain the following statement.

Theorem 18 *For every fixed d there is a polynomial time algorithm that, given integer $m \times n$ matrix A, its Graver basis $\mathcal{G}(A)$, $l, u \in \mathbb{Z}^n_\infty$, $x \in \mathbb{Z}^n$ with $l \le x \le u$, $w_1, \ldots, w_d \in \mathbb{Z}^n$, and convex $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle A, \mathcal{G}(A), l, u, x, w_1, \ldots, w_d\rangle]$, solves the convex integer program with $b := Ax$,*

$$\max\{c(w_1 z, \ldots, w_d z) \,:\, z \in \mathbb{Z}^n, \; Az = b,\; l \le z \le u\}\,.$$

Proof First, check in polynomial time using linear programming whether the objective function of any of the following $2n$ linear programs is unbounded,

$$\max\{\pm y_i : y \in P\},\; i = 1, \ldots, n,$$
$$P := \{y \in \mathbb{R}^n \,:\, Ay = b,\; l \le y \le u\}\,.$$

If any is unbounded then terminate, asserting that P is unbounded. Otherwise, let ρ be the least integer upper bound on the absolute value of all optimal objective values. Then $P \subseteq [-\rho, \rho]^n$ and $S := \{y \in \mathbb{Z}^n : Ay = b,\; l \le y \le u\} \subset P$ is finite of radius $\rho(S) \le \rho$. Moreover, by Cramer's rule, $\langle\rho\rangle$ is polynomially bounded in $\langle A, l, u, x\rangle$.

Using the given Graver basis and applying the algorithm of Theorem 12 we can simulate in polynomial time a linear discrete optimization oracle for S. Furthermore, by Lemma 14, the given Graver basis covers all edge-directions of the integer hull $P_I := \mathrm{conv}\{y \in \mathbb{Z}^n : Ay = b, l \le y \le u\} = \mathrm{conv}(S)$. Therefore we can apply the algorithm of Theorem 6 and solve the given convex program in polynomial time. □

Convex N-fold Integer Programming in Polynomial Time

We now extend the result of Theorem 15 and show that convex integer programming problems over n-fold systems can be solved in polynomial time as well. As explained in the beginning of this section, the algorithm either returns an optimal solution, or asserts that the program is infeasible, or asserts that the underlying polyhedron is unbounded.

Theorem 19 *For every fixed d and fixed $(r+s) \times t$ integer matrix A there is a polynomial time algorithm that, given n, lower and upper bounds $l, u \in \mathbb{Z}^{nt}_\infty$, $w_1, \ldots, w_d \in \mathbb{Z}^{nt}$, $b \in \mathbb{Z}^{r+ns}$, and convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle l, u, w_1, \ldots, w_d, b\rangle]$, solves the convex n-fold integer programming problem*

$$\max\{c(w_1 x, \ldots, w_d x) \,:\, x \in \mathbb{Z}^{nt}, \; A^{(n)}x = b,\; l \le x \le u\}\,.$$

Proof First, check in polynomial time using linear programming whether the objective function of any of the following $2nt$ linear programs is unbounded,

$$\max\{\pm y_i : y \in P\},\; i = 1, \ldots, nt,$$
$$P := \{y \in \mathbb{R}^{nt} \,:\, A^{(n)}y = b,\; l \le y \le u\}\,.$$

If any is unbounded then terminate, asserting that P is unbounded. Otherwise, let ρ be the least integer upper bound on the absolute value of all optimal objective values. Then $P \subseteq [-\rho, \rho]^{nt}$ and $S := \{y \in \mathbb{Z}^{nt} : A^{(n)}y = b,\; l \le y \le u\} \subset P$ is finite of radius $\rho(S) \le \rho$. Moreover, by Cramer's rule, $\langle\rho\rangle$ is polynomially bounded in n and $\langle l, u, b\rangle$.

Using the algorithm of Theorem 15 we can simulate in polynomial time a linear discrete optimization oracle for S. Also, using the algorithm of Theorem 14 we can compute in polynomial time the Graver basis $\mathcal{G}(A^{(n)})$ which, by Lemma 14, covers all edge-directions of $P_I := \mathrm{conv}\{y \in \mathbb{Z}^{nt} : A^{(n)}y = b, l \le y \le u\} = \mathrm{conv}(S)$. Therefore we can apply the algorithm of Theorem 6 and solve the given convex n-fold integer programming problem in polynomial time. □

Again, as a special case of Theorem 19 we recover the following result of [12] concerning convex integer programming in standard form over n-fold matrices.

Theorem 20 *For every fixed d and fixed $(r+s) \times t$ integer matrix A there is a polynomial time algorithm that, given n, linear functionals $w_1, \ldots, w_d \in \mathbb{Z}^{nt}$, right-hand side $b \in \mathbb{Z}^{r+ns}$, and convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle w_1, \ldots, w_d, b\rangle]$, solves the convex n-fold integer program in standard form*

$$\max\{c(w_1 x, \ldots, w_d x) \,:\, x \in \mathbb{N}^{nt}, \; A^{(n)}x = b\}\,.$$

Some Applications

Transportation Problems and Packing Problems Theorems 19 and 20 generalize Theorems 15 and 16 by broadly extending the class of objective functions that can be maximized in polynomial time over n-fold systems. Therefore all applications discussed in Sect. "Some Applications" automatically extend accordingly.

First, we have the following analog of Corollary 8 for the *convex integer transportation problem* over long 3-way tables. This has a very broad further generalization to multiway transportation problems over long k-way tables of any dimension k, see Sect. "Multiway Transportation Problems and Privacy in Statistical Databases".

Corollary 10 *For every fixed d, p, q there is a polynomial time algorithm that, given n, arrays $w_1, \ldots, w_d \in \mathbb{Z}^{p\times q\times n}$, line-sums $u \in \mathbb{Z}^{p\times q}$, $v \in \mathbb{Z}^{p\times n}$ and $z \in \mathbb{Z}^{q\times n}$, and convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle w_1, \ldots, w_d, u, v, z\rangle]$, solves the convex integer 3-way line-sum transportation problem*

$$\max\Big\{ c(w_1 x, \ldots, w_d x) \;:\; x \in \mathbb{N}^{p\times q\times n}\,, \quad \sum_i x_{i,j,k} = z_{j,k}\,, \quad \sum_j x_{i,j,k} = v_{i,k}\,, \quad \sum_k x_{i,j,k} = u_{i,j} \Big\}\,.$$

Second, we have the following analog of Corollary 9 for convex bin packing.

Corollary 11 *For every fixed d, number of types t, and type weights $v_1, \ldots, v_t \in \mathbb{Z}$, there is a polynomial time algorithm that, given n bins, item numbers $n_1, \ldots, n_t \in \mathbb{Z}$, bin capacities $u_1, \ldots, u_n \in \mathbb{Z}$, utility matrices $w_1, \ldots, w_d \in \mathbb{Z}^{t\times n}$, and convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle n_1, \ldots, n_t, u_1, \ldots, u_n, w_1, \ldots, w_d\rangle]$, solves the convex integer bin packing problem,*

$$\max\Big\{ c(w_1 x, \ldots, w_d x) \;:\; x \in \mathbb{N}^{t\times n}\,, \quad \sum_j v_j x_{j,k} = u_k\,, \quad \sum_k x_{j,k} = n_j \Big\}\,.$$

Vector Partitioning and Clustering The vector partition problem concerns the partitioning of n items among p players to maximize social value subject to constraints on the number of items each player can receive. More precisely, the data is as follows. With each item i is associated a vector $v_i \in \mathbb{Z}^k$ representing its utility under k criteria. The utility of player h under ordered partition $\pi = (\pi_1, \ldots, \pi_p)$ of the set of items $\{1, \ldots, n\}$ is the sum $v_h^\pi := \sum_{i\in\pi_h} v_i$ of utility vectors of items assigned to h under π. The social value of π is the balancing $c(v_{1,1}^\pi, \ldots, v_{1,k}^\pi, \ldots, v_{p,1}^\pi, \ldots, v_{p,k}^\pi)$ of the player utilities, where c is a convex functional on $\mathbb{R}^{pk}$. In the constrained version, the partition must be of a given *shape*, i. e. the number $|\pi_h|$ of items that player h gets is required to be a given number λ_h (with $\sum \lambda_h = n$). In the unconstrained version, there is no restriction on the number of items per player.

Vector partition problems have applications in diverse areas such as load balancing, circuit layout, ranking, cluster analysis, inventory, and reliability, see e. g. [7,9,25,39,50] and the references therein. Here is a typical example.

Example 6 (minimal variance clustering). This problem has numerous applications in the analysis of statistical data: given n observed points $v_1, \ldots, v_n$ in k-space, group them into p clusters $\pi_1, \ldots, \pi_p$ that minimize the sum of cluster variances given by

$$\sum_{h=1}^{p} \frac{1}{|\pi_h|} \sum_{i\in\pi_h} \left|\left| v_i - \left(\frac{1}{|\pi_h|} \sum_{i\in\pi_h} v_i \right) \right|\right|^2 .$$

Consider instances where there are $n = pm$ points and the desired clustering is balanced, that is, the clusters should have equal size m. Suitable manipulation of the sum of variances expression above shows that the problem is equivalent to a constrained vector partition problem, where $\lambda_h = m$ for all h, and where the convex functional $c\colon \mathbb{R}^{pk} \longrightarrow \mathbb{R}$ (to be maximized) is the Euclidean norm squared, given by

$$c(z) = ||z||^2 = \sum_{h=1}^{p} \sum_{i=1}^{k} |z_{h,i}|^2 .$$

If either the number of criteria k or the number of players p is variable, the partition problem is intractable since it instantly captures NP-hard problems [39].

When both k, p are fixed, both the constrained and unconstrained versions of the vector partition problem are polynomial time solvable [39,50]. We now show that vector partition problems (either constrained or unconstrained) are in fact convex n-fold integer programming problems and therefore, as a consequence of Theorem 20, can be solved is polynomial time.

Corollary 12 *For every fixed number p of players and number k of criteria, there is a polynomial time algorithm that, given n, item vectors $v_1, \ldots, v_n \in \mathbb{Z}^k$, $\lambda_1, \ldots, \lambda_p \in \mathbb{N}$, and convex functional $c\colon \mathbb{R}^{pk} \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle v_1, \ldots, v_n, \lambda_1, \ldots, \lambda_p \rangle]$, solves the constrained and unconstrained partitioning problems.*

Proof There is an obvious one-to-one correspondence between partitions and matrices $x \in \{0,1\}^{p \times n}$ with all column-sums equal to one, where partition π corresponds to the matrix x with $x_{h,i} = 1$ if $i \in \pi_h$ and $x_{h,i} = 0$ otherwise. Let $d := pk$ and define d matrices $w_{h,j} \in \mathbb{Z}^{p \times n}$ by setting $(w_{h,j})_{h,i} := v_{i,j}$ for all $h = 1, \ldots, p$, $i = 1, \ldots, n$ and $j = 1, \ldots, k$, and setting all other entries to zero. Then for any partition π and its corresponding matrix x we have $v^{\pi}_{h,j} = w_{h,j}x$ for all $h = 1, \ldots, p$ and $j = 1, \ldots, k$. Therefore, the unconstrained vector partition problem is the convex integer program

$$\max\left\{ c(w_{1,1}x, \ldots, w_{p,k}x) \,:\, x \in \mathbb{N}^{p \times n}\,, \sum_h x_{h,i} = 1 \right\}.$$

Suitably arranging the variables in a vector, this becomes a convex n-fold integer program with a $(0+1) \times p$ defining matrix A, where A_1 is empty and $A_2 := (1, \ldots, 1)$.

Similarly, the constrained vector partition problem is the convex integer program

$$\max\left\{ c(w_{1,1}x, \ldots, w_{p,k}x) \,:\, x \in \mathbb{N}^{p \times n}\,, \sum_h x_{h,i} = 1\,,\ \sum_i x_{h,i} = \lambda_h \right\}.$$

This again is a convex n-fold integer program, now with a $(p+1) \times p$ defining matrix A, where now $A_1 := I_p$ is the $p \times p$ identity matrix and $A_2 := (1, \ldots, 1)$ as before.

Using the algorithm of Theorem 20, this convex n-fold integer program, and hence the given vector partition problem, can be solved in polynomial time. □

Multiway Transportation Problems and Privacy in Statistical Databases

Transportation problems form a very important class of discrete optimization problems. The feasible points in a transportation problem are the multiway tables ("contingency tables" in statistics) such that the sums of entries over some of their lower dimensional subtables such as lines or planes ("margins" in statistics) are specified. Transportation problems and their corresponding transportation polytopes have been used and studied extensively in the operations research and mathematical programming literature, as well as in the statistics literature in the context of secure statistical data disclosure and management by public agencies, see [4,6,11,18,19,42,43,53,60,62] and references therein.

In this section we completely settle the algorithmic complexity of treating multiway tables and discuss the applications to transportation problems and secure statistical data disclosure, as follows. After introducing some terminology in Sect. "Tables and Margins", we go on to describe, in Sect. "The Universality Theorem", a universality result that shows that "short" 3-way $r \times c \times 3$ tables, with variable number r of rows and variable number c of columns but fixed small number 3 of layers (hence "short"), are *universal* in a very strong sense. In Sect. "The Complexity of the Multiway Transportation Problem" we discuss the general multiway transportation problem. Using the results of Sect. "The Universality Theorem" and the results on linear and convex n-fold integer programming from Sect. "Linear N-fold Integer Programming" and Sect. "Convex Integer Programming", we show that the transportation problem is intractable for short 3-way $r \times c \times 3$ tables but polynomial time treatable for "long" $(k+1)$-way $m_1 \times \cdots \times m_k \times n$ tables, with k and the sides $m_1, \ldots, m_k$ fixed (but arbitrary), and the number n of layers variable (hence "long"). In Sect. "Privacy and Entry-Uniqueness" we turn to discuss data privacy and security and consider the central problem of detecting entry uniqueness in tables with disclosed margins. We show that as a consequence of the results of Sect. "The Universality Theorem" and Sect. "The Complexity of

the Multiway Transportation Problem", and in analogy to the complexity of the transportation problem established in Sect. "The Complexity of the Multiway Transportation Problem", the entry uniqueness problem is intractable for short 3-way $r \times c \times 3$ tables but polynomial time decidable for long $(k+1)$-way $m_1 \times \cdots \times m_k \times n$ tables.

Tables and Margins

We start with some terminology on tables, margins and transportation polytopes. A *k-way table* is an $m_1 \times \cdots \times m_k$ array $x = (x_{i_1,\ldots,i_k})$ of nonnegative integers. A *k-way transportation polytope* (or simply *k-way polytope* for brevity) is the set of all $m_1 \times \cdots \times m_k$ nonnegative arrays $x = (x_{i_1,\ldots,i_k})$ such that the sums of the entries over some of their lower dimensional sub-arrays (margins) are specified. More precisely, for any tuple $(i_1, \ldots, i_k)$ with $i_j \in \{1, \ldots, m_j\} \cup \{+\}$, the corresponding *margin* $x_{i_1,\ldots,i_k}$ is the sum of entries of x over all coordinates j with $i_j = +$. The *support* of $(i_1, \ldots, i_k)$ and of $x_{i_1,\ldots,i_k}$ is the set $\mathrm{supp}(\mathrm{i}_1, \ldots, \mathrm{i}_k) := \{\mathrm{j} : \mathrm{i}_\mathrm{j} \neq +\}$ of non-summed coordinates. For instance, if x is a $4 \times 5 \times 3 \times 2$ array then it has 12 margins with support $F = \{1, 3\}$ such as $x_{3,+,2,+} = \sum_{i_2=1}^{5} \sum_{i_4=1}^{2} x_{3,i_2,2,i_4}$. A collection of margins is *hierarchical* if, for some family $\mathcal{F}$ of subsets of $\{1, \ldots, k\}$, it consists of all margins $u_{i_1,\ldots,i_k}$ with support in $\mathcal{F}$. In particular, for any $0 \leq h \leq k$, the collection of *all h-margins* of k-tables is the hierarchical collection with $\mathcal{F}$ the family of all h-subsets of $\{1, \ldots, k\}$. Given a hierarchical collection of margins $u_{i_1,\ldots,i_k}$ supported on a family $\mathcal{F}$ of subsets of $\{1, \ldots, k\}$, the corresponding *k-way polytope* is the set of nonnegative arrays with these margins,

$$T_{\mathcal{F}} := \{\, x \in \mathbb{R}_+^{m_1 \times \cdots \times m_k} : x_{i_1,\ldots,i_k} = u_{i_1,\ldots,i_k}, \ \mathrm{supp}(\mathrm{i}_1, \ldots, \mathrm{i}_k) \in \mathcal{F} \,\}.$$

The integer points in this polytope are precisely the k-way tables with the given margins.

The Universality Theorem

We now describe the following *universality* result of [14,16] which shows that, quite remarkably, *any* rational polytope is a short 3-way $r \times c \times 3$ polytope with all line-sums specified. (In the terminology of Sect. "Tables and Margins" this is the $r \times c \times 3$ polytope $T_{\mathcal{F}}$ of all 2-margins fixed, supported on the family $\mathcal{F} = \{\{1,2\},\{1,3\},\{2,3\}\}$.) By saying that a polytope $P \subset \mathbb{R}^p$ is *representable* as a polytope $Q \subset \mathbb{R}^q$ we mean in the strong sense that there is an injection $\sigma : \{1, \ldots, p\} \longrightarrow \{1, \ldots, q\}$ such that the coordinate-erasing projection

$$\pi : \mathbb{R}^q \longrightarrow \mathbb{R}^p : x = (x_1, \ldots, x_q) \mapsto \pi(x) = (x_{\sigma(1)}, \ldots, x_{\sigma(p)})$$

provides a bijection between Q and P and between the sets of integer points $Q \cap \mathbb{Z}^q$ and $P \cap \mathbb{Z}^p$. In particular, if P is representable as Q then P and Q are isomorphic in any reasonable sense: they are linearly equivalent and hence all linear programming related problems over the two are polynomial time equivalent; they are combinatorially equivalent and hence they have the same face numbers and facial structure; and they are integer equivalent and therefore all integer programming and integer counting related problems over the two are polynomial time equivalent as well.

We provide only an outline of the proof of the following statement; complete details and more consequences of this theorem can be found in [14,16].

Theorem 21 *There is a polynomial time algorithm that, given $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$, encoded as $[\langle A, b\rangle]$, produces r, c and line-sums $u \in \mathbb{Z}^{r \times c}$, $v \in \mathbb{Z}^{r \times 3}$ and $z \in \mathbb{Z}^{c \times 3}$ such that the polytope $P := \{y \in \mathbb{R}_+^n : Ay = b\}$ is representable as the 3-way polytope*

$$T := \left\{ x \in \mathbb{R}_+^{r \times c \times 3} : \sum_i x_{i,j,k} = z_{j,k}, \ \sum_j x_{i,j,k} = v_{i,k}, \ \sum_k x_{i,j,k} = u_{i,j} \right\}.$$

Proof The construction proving the theorem consists of three polynomial time steps, each representing a polytope of a given format as a polytope of another given format.

First, we show that any $P := \{y \geq 0 : Ay = b\}$ with A, b integer can be represented in polynomial time as $Q := \{x \geq 0 : Cx = d\}$ with C matrix all entries of which are in $\{-1, 0, 1, 2\}$. This reduction of coefficients will enable the rest of the steps to run in polynomial time. For each variable y_j let $k_j := \max\{\lfloor \log_2 |a_{i,j}| \rfloor :$

$i = 1, \ldots m\}$ be the maximum number of bits in the binary representation of the absolute value of any entry $a_{i,j}$ of A. Introduce variables $x_{j,0}, \ldots, x_{j,k_j}$, and relate them by the equations $2x_{j,i} - x_{j,i+1} = 0$. The representing injection σ is defined by $\sigma(j) := (j, 0)$, embedding y_j as $x_{j,0}$. Consider any term $a_{i,j}\, y_j$ of the original system. Using the binary expansion $|a_{i,j}| = \sum_{s=0}^{k_j} t_s 2^s$ with all $t_s \in \{0, 1\}$, we rewrite this term as $\pm \sum_{s=0}^{k_j} t_s x_{j,s}$. It is not hard to verify that this represents P as Q with defining $\{-1, 0, 1, 2\}$-matrix.

Second, we show that any $Q := \{y \geq 0 : Ay = b\}$ with A, b integer can be represented as a face F of a 3-way polytope with all plane-sums fixed, that is, a face of a 3-way polytope $T_{\mathcal{F}}$ of all 1-margins fixed, supported on the family $\mathcal{F} = \{\{1\}, \{2\}, \{3\}\}$.

Since Q is a polytope and hence bounded, we can compute (using Cramer's rule) an integer upper bound U on the value of any coordinate y_j of any $y \in Q$. Note also that a face of a 3-way polytope $T_{\mathcal{F}}$ is the set of all $x = (x_{i,j,k})$ with some entries forced to zero; these entries are termed "forbidden", and the other entries are termed "enabled".

For each variable y_j, let r_j be the largest between the sum of positive coefficients of y_j and the sum of absolute values of negative coefficients of y_j over all equations,

$$r_j := \max\left(\sum_k \{a_{k,j} : a_{k,j} > 0\}, \sum_k \{|a_{k,j}| : a_{k,j} < 0\} \right).$$

Assume that A is of size $m \times n$. Let $r := \sum_{j=1}^{n} r_j$, $R := \{1, \ldots, r\}$, $h := m + 1$ and $H := \{1, \ldots, h\}$. We now describe how to construct vectors $u, v \in \mathbb{Z}^r$, $z \in \mathbb{Z}^h$, and a set $E \subset R \times R \times H$ of triples – the enabled, non-forbidden, entries – such that the polytope Q is represented as the face F of the corresponding 3-way polytope of $r \times r \times h$ arrays with plane-sums u, v, z and only entries indexed by E enabled,

$$F := \left\{ x \in \mathbb{R}_+^{r \times r \times h} : x_{i,j,k} = 0 \text{ for all } (i,j,k) \notin E, \text{ and } \sum_{i,j} x_{i,j,k} = z_k, \quad \sum_{i,k} x_{i,j,k} = v_j, \quad \sum_{j,k} x_{i,j,k} = u_i \right\}.$$

We also indicate the injection $\sigma : \{1, \ldots, n\} \longrightarrow R \times R \times H$ giving the desired embedding of coordinates y_j as coordinates $x_{i,j,k}$ and the representation of Q as F.

Roughly, each equation $k = 1, \ldots, m$ is encoded in a "horizontal plane" $R \times R \times \{k\}$ (the last plane $R \times R \times \{h\}$ is included for consistency with its entries being "slacks"); and each variable y_j, $j = 1, \ldots, n$ is encoded in a "vertical box" $R_j \times R_j \times H$, where $R = \biguplus_{j=1}^{n} R_j$ is the natural partition of R with $|R_j| = r_j$ for all $j = 1, \ldots, n$, that is, with $R_j := \{1 + \sum_{l<j} r_l, \ldots, \sum_{l \leq j} r_l\}$.

Now, all "vertical" plane-sums are set to the same value U, that is, $u_j := v_j := U$ for $j = 1, \ldots, r$. All entries not in the union $\biguplus_{j=1}^{n} R_j \times R_j \times H$ of the variable boxes will be forbidden. We now describe the enabled entries in the boxes; for simplicity we discuss the box $R_1 \times R_1 \times H$, the others being similar. We distinguish between the two cases $r_1 = 1$ and $r_1 \geq 2$. In the first case, $R_1 = \{1\}$; the box, which is just the single line $\{1\} \times \{1\} \times H$, will have exactly two enabled entries $(1, 1, k^+), (1, 1, k^-)$ for suitable k^+, k^- to be defined later. We set $\sigma(1) := (1, 1, k^+)$, namely embed $y_1 = x_{1,1,k^+}$. We define the *complement* of the variable y_1 to be $\bar{y}_1 := U - y_1$ (and likewise for the other variables). The vertical sums u, v then force $\bar{y}_1 = U - y_1 = U - x_{1,1,k^+} = x_{1,1,k^-}$, so the complement of y_1 is also embedded. Next, consider the case $r_1 \geq 2$. For each $s = 1, \ldots, r_1$, the line $\{s\} \times \{s\} \times H$ (respectively, $\{s\} \times \{1 + (s \mod r_1)\} \times H$) will contain one enabled entry $(s, s, k^+(s))$ (respectively, $(s, 1 + (s \mod r_1), k^-(s))$. All other entries of $R_1 \times R_1 \times H$ will be forbidden. Again, we set $\sigma(1) := (1, 1, k^+(1))$, namely embed $y_1 = x_{1,1,k^+(1)}$; it is then not hard to see that, again, the vertical sums u, v force $x_{s,s,k^+(s)} = x_{1,1,k^+(1)} = y_1$ and $x_{s,1+(s \mod r_1),k^-(s)} = U - x_{1,1,k^+(1)} = \bar{y}_1$ for each $s = 1, \ldots, r_1$. Therefore, both y_1 and $\bar{y}_1$ are each embedded in r_1 distinct entries.

We now encode the equations by defining the horizontal plane-sums z and the indices $k^+(s), k^-(s)$ above as follows. For $k = 1, \ldots, m$, consider the kth equation $\sum_j a_{k,j} y_j = b_k$. Define the index sets $J^+ := \{j : a_{k,j} > 0\}$ and $J^- := \{j : a_{k,j} < 0\}$, and set $z_k := b_k + U \cdot \sum_{j \in J^-} |a_{k,j}|$. The last coordinate of z is set for consistency with u, v to be $z_h = z_{m+1} := r \cdot U - \sum_{k=1}^{m} z_k$. Now, with $\bar{y}_j := U - y_j$ the complement of variable y_j as above, the kth equation can be

rewritten as

$$\begin{aligned}&\sum_{j\in J^+} a_{k,j}y_j + \sum_{j\in J^-} |a_{k,j}|\bar{y}_j\\ &= \sum_{j=1}^{n} a_{k,j}y_j + U\cdot\sum_{j\in J^-}|a_{k,j}|\\ &= b_k + U\cdot\sum_{j\in J^-}|a_{k,j}| = z_k\ .\end{aligned}$$

To encode this equation, we simply "pull down" to the corresponding kth horizontal plane as many copies of each variable y_j or $\bar{y}_j$ by suitably setting $k^+(s) := k$ or $k^-(s) := k$. By the choice of r_j there are sufficiently many, possibly with a few redundant copies which are absorbed in the last hyperplane by setting $k^+(s) := m+1$ or $k^-(s) := m+1$. This completes the encoding and provides the desired representation.

Third, we show that any 3-way polytope with plane-sums fixed and entry bounds,

$$\begin{aligned}F := \Bigg\{ y\in\mathbb{R}_+^{l\times m\times n} : &\sum_{i,j} y_{i,j,k} = c_k\,,\\ &\sum_{i,k} y_{i,j,k} = b_j\,,\\ &\sum_{j,k} y_{i,j,k} = a_i\,,\\ &y_{i,j,k}\le e_{i,j,k}\Bigg\}\,,\end{aligned}$$

can be represented as a 3-way polytope with line-sums fixed (and no entry bounds),

$$\begin{aligned}T := \Bigg\{ x\in\mathbb{R}_+^{r\times c\times 3} : &\sum_I x_{I,J,K} = z_{J,K}\,,\\ &\sum_J x_{I,J,K} = v_{I,K}\,,\ \sum_K x_{I,J,K} = u_{I,J}\Bigg\}\ .\end{aligned}$$

In particular, this implies that any face F of a 3-way polytope with plane-sums fixed can be represented as a 3-way polytope T with line-sums fixed: forbidden entries are encoded by setting a "forbidding" upper-bound $e_{i,j,k} := 0$ on all forbidden entries $(i,j,k)\notin E$ and an "enabling" upper-bound $e_{i,j,k} := U$ on all enabled entries $(i,j,k)\in E$. We describe the presentation, but omit the proof that it is indeed valid; further details on this step can be found in [14,15,16]. We give explicit formulas for $u_{I,J}, v_{I,K}, z_{J,K}$ in terms of a_i, b_j, c_k and $e_{i,j,k}$ as follows. Put $r := l\cdot m$ and $c := n+l+m$. The first index I of each entry $x_{I,J,K}$ will be a pair $I = (i,j)$ in the r-set

$$\begin{aligned}\{&(1,1),\ldots,(1,m),(2,1),\ldots,\\ &(2,m),\ldots,(l,1),\ldots,(l,m)\}\,.\end{aligned}$$

The second index J of each entry $x_{I,J,K}$ will be a pair $J = (s,t)$ in the c-set

$$\{(1,1),\ldots,(1,n),(2,1),\ldots,(2,l),(3,1),\ldots,(3,m)\}\,.$$

The last index K will simply range in the 3-set $\{1,2,3\}$. We represent F as T via the injection σ given explicitly by $\sigma(i,j,k) := ((i,j),(1,k),1)$, embedding each variable $y_{i,j,k}$ as the entry $x_{(i,j),(1,k),1}$. Let U now denote the minimal between the two values $\max\{a_1,\ldots,a_l\}$ and $\max\{b_1,\ldots,b_m\}$. The line-sums (2-margins) are set to be

$$\begin{aligned}u_{(i,j),(1,t)} &= e_{i,j,t},\\ u_{(i,j),(2,t)} &= \begin{cases} U & \text{if } t=i,\\ 0 & \text{otherwise.}\end{cases},\\ u_{(i,j),(3,t)} &= \begin{cases} U & \text{if } t=j,\\ 0 & \text{otherwise.}\end{cases}\end{aligned}$$

$$v_{(i,j),t} = \begin{cases} U & \text{if } t=1,\\ e_{i,j,+} & \text{if } t=2,\\ U & \text{if } t=3.\end{cases},$$

$$z_{(i,j),1} = \begin{cases} c_j & \text{if } i=1,\\ m\cdot U - a_j & \text{if } i=2,\\ 0 & \text{if } i=3.\end{cases}$$

$$z_{(i,j),2} = \begin{cases} e_{+,+,j} - c_j & \text{if } i=1,\\ 0 & \text{if } i=2,\\ b_j & \text{if } i=3.\end{cases},$$

$$z_{(i,j),3} = \begin{cases} 0 & \text{if } i=1,\\ a_j & \text{if } i=2,\\ l\cdot U - b_j & \text{if } i=3.\end{cases}.$$

Applying the first step to the given rational polytope P, applying the second step to the resulting Q, and applying the third step to the resulting F, we get in polynomial time a 3-way $r\times c\times 3$ polytope T of all line-sums fixed representing P as claimed. □

The Complexity of the Multiway Transportation Problem

We are now finally in position to settle the complexity of the general multiway transportation problem. The data for the problem consists of: positive integers k (table dimension) and $m_1, \ldots, m_k$ (table sides); family $\mathcal{F}$ of subsets of $\{1, \ldots, k\}$ (supporting the hierarchical collection of margins to be fixed); integer values $u_{i_1,\ldots,i_k}$ for all margins supported on $\mathcal{F}$; and integer "profit" $m_1 \times \cdots \times m_k$ array w. The transportation problem is to find an $m_1 \times \cdots \times m_k$ table having the given margins and attaining maximum profit, or assert than none exists. Equivalently, it is the linear integer programming problem of maximizing the linear functional defined by w over the transportation polytope $T_{\mathcal{F}}$,

$$\max\{\, wx \,:\, x \in \mathbb{N}^{m_1 \times \cdots \times m_k} \,:\, x_{i_1,\ldots,i_k} = u_{i_1,\ldots,i_k}\,, \quad \mathrm{supp}(\mathrm{i}_1, \ldots, \mathrm{i}_k) \in \mathcal{F} \,\}\,.$$

The following result of [15] is an immediate consequence of Theorem 21 . It asserts that if two sides of the table are variable part of the input then the transportation problem is intractable already for short 3-way tables with $\mathcal{F} = \{\{1,2\},\{1,3\},\{2,3\}\}$ supporting all 2-margins (line-sums). This result can be easily extended to k-way tables of any dimension $k \geq 3$ and $\mathcal{F}$ the collection of all h-subsets of $\{1, \ldots, k\}$ for any $1 < h < k$ as long as two sides of the table are variable; we omit the proof of this extended result.

Corollary 13 *It is NP-complete to decide, given r, c, and line-sums $u \in \mathbb{Z}^{r\times c}$, $v \in \mathbb{Z}^{r\times 3}$, and $z \in \mathbb{Z}^{c\times 3}$, encoded as $[\langle u, v, z\rangle]$, if the following set of tables is nonempty,*

$$S := \left\{ x \in \mathbb{N}^{r\times c\times 3} \,:\, \sum_i x_{i,j,k} = z_{j,k}\,, \quad \sum_j x_{i,j,k} = v_{i,k}\,,\ \sum_k x_{i,j,k} = u_{i,j} \right\}.$$

Proof The integer programming feasibility problem is to decide, given $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$, if $\{y \in \mathbb{N}^n : Ay = b\}$ is nonempty. Given such A and b, the polynomial time algorithm of Theorem 21 produces r, c and $u \in \mathbb{Z}^{r\times c}$, $v \in \mathbb{Z}^{r\times 3}$, and $z \in \mathbb{Z}^{c\times 3}$, such that $\{y \in \mathbb{N}^n : Ay = b\}$ is nonempty if and only if the set S above is nonempty. This reduces integer programming feasibility to short 3-way line-sum transportation feasibility. Since the former is NP-complete (see e. g. [55]), so turns out to be the latter. □

We now show that in contrast, when all sides but one are fixed (but arbitrary), and one side n is variable, then the corresponding long k-way transportation problem for any hierarchical collection of margins is an n-fold integer programming problem and therefore, as a consequence of Theorem 16, can be solved is polynomial time. This extends Corollary 8 established in Sect. "Three-Way Line-Sum Transportation Problems" for 3-way line-sum transportation.

Corollary 14 *For every fixed k, table sides $m_1, \ldots, m_k$, and family $\mathcal{F}$ of subsets of $\{1, \ldots, k+1\}$, there is a polynomial time algorithm that, given n, integer values $u = (u_{i_1,\ldots,i_{k+1}})$ for all margins supported on $\mathcal{F}$, and integer $m_1 \times \cdots \times m_k \times n$ array w, encoded as $[\langle u, w\rangle]$, solves the linear integer multiway transportation problem*

$$\max\{\, wx \,:\, x \in \mathbb{N}^{m_1\times\cdots\times m_k\times n},\ x_{i_1,\ldots,i_{k+1}} = u_{i_1,\ldots,i_{k+1}},\ \mathrm{supp}(\mathrm{i}_1, \ldots, \mathrm{i}_{k+1}) \in \mathcal{F} \,\}\,.$$

Proof Re-index the arrays as $x = (x^1, \ldots, x^n)$ with each $x^j = (x_{i_1,\ldots,i_k,j})$ a suitably indexed $m_1 m_2 \cdots m_k$ vector representing the jth layer of x. Then the transportation problem can be encoded as an n-fold integer programming problem in standard form,

$$\max\{wx \,:\, x \in \mathbb{N}^{nt},\ A^{(n)}x = b\}\,,$$

with an $(r+s)\times t$ defining matrix A where $t := m_1 m_2 \cdots m_k$ and r, s, A_1 and A_2 are determined from $\mathcal{F}$, and with right-hand side $b := (b^0, b^1, \ldots, b^n) \in \mathbb{Z}^{r+ns}$ determined from the margins $u = (u_{i_1,\ldots,i_{k+1}})$, in such a way that the equations $A_1(\sum_{j=1}^n x^j) = b^0$ represent the constraints of all margins $x_{i_1,\ldots,i_k,+}$ (where summation over layers occurs), whereas the equations $A_2 x^j = b^j$ for $j = 1, \ldots, n$ represent the constraints of all margins $x_{i_1,\ldots,i_k,j}$ with $j \neq +$ (where summations are within a single layer at a time).

Using the algorithm of Theorem 16, this n-fold integer program, and hence the given multiway transportation problem, can be solved in polynomial time. □

The proof of Corollary 14 shows that the set of feasible points of any long k-way transportation problem, with all sides but one fixed and one side n variable,

for any hierarchical collection of margins, is an n-fold integer programming problem. Therefore, as a consequence of Theorem 20, we also have the following extension of Corollary 14 for the convex integer multiway transportation problem over long k-way tables.

Corollary 15 *For every fixed d, k, table sides $m_1, \ldots, m_k$, and family $\mathcal{F}$ of subsets of $\{1, \ldots, k+1\}$, there is a polynomial time algorithm that, given n, integer values $u = (u_{i_1,\ldots,i_{k+1}})$ for all margins supported on $\mathcal{F}$, integer $m_1 \times \cdots \times m_k \times n$ arrays $w_1, \ldots, w_d$, and convex functional $c\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ presented by a comparison oracle, encoded as $[\langle u, w_1, \ldots, w_d\rangle]$, solves the convex integer multiway transportation problem*

$$\max\{\, c(w_1 x, \ldots, w_d x) \;:\; x \in \mathbb{N}^{m_1 \times \cdots \times m_k \times n},$$
$$x_{i_1,\ldots,i_{k+1}} = u_{i_1,\ldots,i_{k+1}},\ \mathrm{supp}(\mathrm{i}_1, \ldots, \mathrm{i}_{k+1}) \in \mathcal{F}\}\,.$$

Privacy and Entry-Uniqueness

A common practice in the disclosure of a multiway table containing sensitive data is to release some of the table margins rather than the table itself, see e. g. [11,18,19] and the references therein. Once the margins are released, the security of any specific entry of the table is related to the set of possible values that can occur in that entry in any table having the same margins as those of the source table in the data base. In particular, if this set consists of a unique value, that of the source table, then this entry can be exposed and privacy can be violated. This raises the following fundamental *entry-uniqueness problem*: given a consistent disclosed (hierarchical) collection of margin values, and a specific entry index, is the value that can occur in that entry in any table having these margins unique? We now describe the results of [48] that settle the complexity of this problem, and interpret the consequences for secure statistical data disclosure.

First, we show that if two sides of the table are variable part of the input then the entry-uniqueness problem is intractable already for short 3-way tables with all 2-margins (line-sums) disclosed (corresponding to $\mathcal{F} = \{\{1,2\},\{1,3\},\{2,3\}\}$). This can be easily extended to k-way tables of any dimension $k \geq 3$ and $\mathcal{F}$ the collection of all h-subsets of $\{1, \ldots, k\}$ for any $1 < h < k$ as long as two sides of the table are variable; we omit the proof of this extended result. While this result indicates that the disclosing agency may not be able to check for uniqueness, in this situation, some consolation is in that an adversary will be computationally unable to identify and retrieve a unique entry either.

Corollary 16 *It is coNP-complete to decide, given r, c, and line-sums $u \in \mathbb{Z}^{r\times c}$, $v \in \mathbb{Z}^{r\times 3}$, $z \in \mathbb{Z}^{c\times 3}$, encoded as $[\langle u, v, z\rangle]$, if the entry $x_{1,1,1}$ is the same in all tables in*

$$\left\{ x \in \mathbb{N}^{r\times c\times 3} \;:\; \sum_i x_{i,j,k} = z_{j,k}\,,\ \sum_j x_{i,j,k} = v_{i,k}\,,\ \sum_k x_{i,j,k} = u_{i,j} \right\}.$$

Proof The *subset-sum problem*, well known to be NP-complete, is the following: given positive integers $a_0, a_1, \ldots, a_m$, decide if there is an $I \subseteq \{1, \ldots, m\}$ with $a_0 = \sum_{i\in I} a_i$. We reduce the complement of subset-sum to entry-uniqueness. Given $a_0, a_1, \ldots, a_m$, consider the polytope in $2(m+1)$ variables $y_0, y_1 \ldots, y_m, z_0, z_1, \ldots, z_m$,

$$P := \left\{ (y,z) \in \mathbb{R}^{2(m+1)}_+ \;:\; a_0 y_0 - \sum_{i=1}^m a_i y_i = 0\,,\ y_i + z_i = 1\,,\ i = 0, 1 \ldots, m \right\}.$$

First, note that it always has one integer point with $y_0 = 0$, given by $y_i = 0$ and $z_i = 1$ for all i. Second, note that it has an integer point with $y_0 \neq 0$ if and only if there is an $I \subseteq \{1, \ldots, m\}$ with $a_0 = \sum_{i\in I} a_i$, given by $y_0 = 1$, $y_i = 1$ for $i \in I$, $y_i = 0$ for $i \in \{1, \ldots, m\} \setminus I$, and $z_i = 1 - y_i$ for all i. Lifting P to a suitable $r \times c \times 3$ line-sum polytope T with the coordinate y_0 embedded in the entry $x_{1,1,1}$ using Theorem 21, we find that T has a table with $x_{1,1,1} = 0$, and this value is unique among the tables in T if and only if there is *no* solution to the subset-sum problem with $a_0, a_1, \ldots, a_m$. □

Next we show that, in contrast, when all table sides but one are fixed (but arbitrary), and one side n is variable, then, as a consequence of Corollary 14, the corresponding long k-way entry-uniqueness problem for any hierarchical collection of margins can be solved is polynomial time. In this situation, the algorithm of Corollary 17 below allows disclosing agencies to efficiently check possible collections of margins before disclosure: if an entry value is not unique then disclosure

may be assumed secure, whereas if the value is unique then disclosure may be risky and fewer margins should be released. Note that this situation, of long multiway tables, where one category is significantly richer than the others, that is, when each sample point can take many values in one category and only few values in the other categories, occurs often in practical applications, e. g., when one category is the individuals age and the other categories are binary ("yes-no"). In such situations, our polynomial time algorithm below allows disclosing agencies to check entry-uniqueness and make learned decisions on secure disclosure.

Corollary 17 *For every fixed k, table sides $m_1, \ldots, m_k$, and family $\mathcal{F}$ of subsets of $\{1, \ldots, k+1\}$, there is a polynomial time algorithm that, given n, integer values $u = (u_{j_1,\ldots,j_{k+1}})$ for all margins supported on $\mathcal{F}$, and entry index $(i_1, \ldots, i_{k+1})$, encoded as $[n, \langle u \rangle]$, decides if the entry $x_{i_1,\ldots,i_{k+1}}$ is the same in all tables in the set*

$$\{x \in \mathbb{N}^{m_1 \times \cdots \times m_k \times n} \,:\, x_{j_1,\ldots,j_{k+1}} = u_{j_1,\ldots,j_{k+1}}\,, \quad \mathrm{supp}(\mathrm{j}_1, \ldots, \mathrm{j}_{k+1}) \in \mathcal{F}\}\,.$$

Proof By Corollary 14 we can solve in polynomial time both transportation problems

$$l := \min\{\, x_{i_1,\ldots,i_{k+1}} \,:\, x \in \mathbb{N}^{m_1 \times \cdots \times m_k \times n}\,, \quad x \in T_{\mathcal{F}}\,\}\,,$$

$$u := \max\{\, x_{i_1,\ldots,i_{k+1}} \,:\, x \in \mathbb{N}^{m_1 \times \cdots \times m_k \times n}\,, \quad x \in T_{\mathcal{F}}\,\}\,,$$

over the corresponding k-way transportation polytope

$$T_{\mathcal{F}} := \{\, x \in \mathbb{R}_+^{m_1 \times \cdots \times m_k \times n} \,:\, x_{j_1,\ldots,j_{k+1}} = u_{j_1,\ldots,j_{k+1}}\,, \quad \mathrm{supp}(\mathrm{j}_1, \ldots, \mathrm{j}_{k+1}) \in \mathcal{F}\,\}\,.$$

Clearly, entry $x_{i_1,\ldots,i_{k+1}}$ has the same value in all tables with the given (disclosed) margins if and only if $l = u$, completing the description of the algorithm and the proof. □

References

1. Aho AV, Hopcroft JE, Ullman JD (1975) The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading
2. Allemand K, Fukuda K, Liebling TM, Steiner E (2001) A polynomial case of unconstrained zero-one quadratic optimization. Math Prog Ser A 91:49–52
3. Alon N, Onn S (1999) Separable partitions. Discret Appl Math 91:39–51
4. Aoki S, Takemura A (2003) Minimal basis for connected Markov chain over $3 \times 3 \times K$ contingency tables with fixed two-dimensional marginals. Austr New Zeal J Stat 45:229–249
5. Babson E, Onn S, Thomas R (2003) The Hilbert zonotope and a polynomial time algorithm for universal Gröbner bases. Adv Appl Math 30:529–544
6. Balinski ML, Rispoli FJ (1993) Signature classes of transportation polytopes. Math Prog Ser A 60:127–144
7. Barnes ER, Hoffman AJ, Rothblum UG (1992) Optimal partitions having disjoint convex and conic hulls. Math Prog 54:69–86
8. Berstein Y, Onn S: Nonlinear bipartite matching. Disc Optim (to appear)
9. Boros E, Hammer PL (1989) On clustering problems with connected optima in Euclidean spaces. Discret Math 75: 81–88
10. Chvátal V (1973) Edmonds polytopes and a hierarchy of combinatorial problems. Discret Math 4:305–337
11. Cox LH (2003) On properties of multi-dimensional statistical tables. J Stat Plan Infer 117:251–273
12. De Loera J, Hemmecke R, Onn S, Rothblum UG, Weismantel R: Integer convex maximization via Graver bases. E-print: arXiv:math.CO/0609019. (submitted)
13. De Loera J, Hemmecke R, Onn S, Weismantel R: N-fold integer programming. Disc Optim (to appear)
14. De Loera J, Onn S (2004) All rational polytopes are transportation polytopes and all polytopal integer sets are contingency tables. In: Proc IPCO 10 – Symp on Integer Programming and Combinatoral Optimization, Columbia University, New York. Lec Not Comp Sci. Springer, 3064, pp 338–351
15. De Loera J, Onn S (2004) The complexity of three-way statistical tables. SIAM J Comput 33:819–836
16. De Loera J, Onn S (2006) All linear and integer programs are slim 3-way transportation programs. SIAM J Optim 17:806–821
17. De Loera J, Onn S (2006) Markov bases of three-way tables are arbitrarily complicated. J Symb Comput 41:173–181
18. Domingo-Ferrer J, Torra V (eds) (2004) Privacy in Statistical Databases. Proc. PSD 2004 – Int Symp Privacy in Statistical Databases, Barcelona, Spain. Lec Not Comp Sci. Springer, 3050
19. Doyle P, Lane J, Theeuwes J, Zayatz L (eds) (2001) Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies. North-Holland, Amsterdam
20. Edelsbrunner H, O'Rourke J, Seidel R (1986) Constructing arrangements of lines and hyperplanes with applications. SIAM J Comput 15:341–363
21. Edelsbrunner H, Seidel R, Sharir M (1991) On the zone theorem for hyperplane arrangements. In: New Results and

Trends in Computer Science. Lec Not Comp Sci. Springer, 555, pp 108–123
22. Edmonds J (1971) Matroids and the greedy algorithm. Math Prog 1:127–136
23. Edmonds J, Karp RM (1972) Theoretical improvements in algorithmic efficiency of network flow problems. J Ass Comput Mach 19:248–264
24. Frank A, Tardos E (1987) An application of simultaneous Diophantine approximation in combinatorial optimization. Combinatorica 7:49–65
25. Fukuda K, Onn S, Rosta V (2003) An adaptive algorithm for vector partitioning. J Global Optim 25:305–319
26. Garey MR, Johnson DS (1979) Computers and Intractability. Freeman, San Francisco
27. Gilmore PC, Gomory RE (1961) A linear programming approach to the cutting-stock problem. Oper Res 9:849–859
28. Graver JE (1975) On the foundation of linear and integer programming I. Math Prog 9:207–226
29. Gritzmann P, Sturmfels B (1993) Minkowski addition of polytopes: complexity and applications to Gröbner bases. SIAM J Discret Math 6:246–269
30. Grötschel M, Lovász L (1995) Combinatorial optimization. In: Handbook of Combinatorics. North-Holland, Amsterdam, pp 1541–1597
31. Grötschel M, Lovász L, Schrijver A (1993) Geometric Algorithms and Combinatorial Optimization, 2nd edn. Springer, Berlin
32. Grünbaum B (2003) Convex Polytopes, 2nd edn. Springer, New York
33. Harding EF (1967) The number of partitions of a set of *n* points in *k* dimensions induced by hyperplanes. Proc Edinburgh Math Soc 15:285–289
34. Hassin R, Tamir A (1989) Maximizing classes of two-parameter objectives over matroids. Math Oper Res 14:362–375
35. Hemmecke R (2003) On the positive sum property and the computation of Graver test sets. Math Prog 96:247–269
36. Hemmecke R, Hemmecke R, Malkin P (2005) 4ti2 Version 1.2–Computation of Hilbert bases, Graver bases, toric Gröbner bases, and more. http://www.4ti2.de/. Accessed Sept 2005
37. Hoffman AJ, Kruskal JB (1956) Integral boundary points of convex polyhedra. In: Linear inequalities and Related Systems, Ann Math Stud 38. Princeton University Press, Princeton, pp 223–246
38. Hoşten S, Sullivant S (2007) A finiteness theorem for Markov bases of hierarchical models. J Comb Theory Ser A 114:311–321
39. Hwang FK, Onn S, Rothblum UG (1999) A polynomial time algorithm for shaped partition problems. SIAM J Optim 10:70–81
40. Khachiyan LG (1979) A polynomial algorithm in linear programming. Sov Math Dok 20:191–194
41. Klee V, Kleinschmidt P (1987) The *d*-step conjecture and its relatives. Math Oper Res 12:718–755
42. Klee V, Witzgall C (1968) Facets and vertices of transportation polytopes. In: Mathematics of the Decision Sciences, Part I, Stanford, CA, 1967. AMS, Providence, pp 257–282
43. Kleinschmidt P, Lee CW, Schannath H (1987) Transportation problems which can be solved by the use of Hirsch-paths for the dual problems. Math Prog 37:153–168
44. Lenstra AK, Lenstra HW Jr, Lovász L (1982) Factoring polynomials with rational coefficients. Math Ann 261:515–534
45. Lovàsz L (1986) An Algorithmic Theory of Numbers, Graphs, and Convexity. CBMS-NSF Ser App Math, SIAM 50:iv+91
46. Onn S (2006) Convex discrete optimization. Lecture Series, Le Séminaire de Mathématiques Supérieures, Combinatorial Optimization: Methods and Applications, Université de Montréal, Canada, June 2006. http://ie.technion.ac.il/~onn/Talks/Lecture_Series.pdf and at http://www.dms.umontreal.ca/sms/ONN_Lecture_Series.pdf. Accessed 19–30 June 2006
47. Onn S (2003) Convex matroid optimization. SIAM J Discret Math 17:249–253
48. Onn S (2006) Entry uniqueness in margined tables. In: Proc. PSD 2006 – Symp. on Privacy in Statistical Databases, Rome, Italy. Lec Not Comp Sci. Springer, 4302, pp 94–101
49. Onn S, Rothblum UG (2004) Convex combinatorial optimization. Disc Comp Geom 32:549–566
50. Onn S, Schulman LJ (2001) The vector partition problem for convex objective functions. Math Oper Res 26:583–590
51. Onn S, Sturmfels B (1999) Cutting Corners. Adv Appl Math 23:29–48
52. Pistone G, Riccomagno EM, Wynn HP (2001) Algebraic Statistics. Chapman and Hall, London
53. Queyranne M, Spieksma FCR (1997) Approximation algorithms for multi-index transportation problems with decomposable costs. Disc Appl Math 76:239–253
54. Santos F, Sturmfels B (2003) Higher Lawrence configurations. J Comb Theory Ser A 103:151–164
55. Schrijver A (1986) Theory of Linear and Integer Programming. Wiley, New York
56. Schulz A, Weismantel R (2002) The complexity of generic primal algorithms for solving general integral programs. Math Oper Res 27:681–692
57. Schulz A, Weismantel R, Ziegler GM (1995) (0, 1)-integer programming: optimization and augmentation are equivalent. In: Proc 3rd Ann Euro Symp Alg. Lec Not Comp Sci. Springer, 979, pp 473–483
58. Sturmfels B (1996) Gröbner Bases and Convex Polytopes. Univ Lec Ser 8. AMS, Providence
59. Tardos E (1986) A strongly polynomial algorithm to solve combinatorial linear programs. Oper Res 34:250–256
60. Vlach M (1986) Conditions for the existence of solutions of the three-dimensional planar transportation problem. Discret Appl Math 13:61–78
61. Wallacher C, Zimmermann U (1992) A combinatorial interior point method for network flow problems. Math Prog 56:321–335

62. Yemelichev VA, Kovalev MM, Kravtsov MK (1984) Polytopes, Graphs and Optimisation. Cambridge University Press, Cambridge
63. Yudin DB, Nemirovskii AS (1977) Informational complexity and efficient methods for the solution of convex extremal problems. Matekon 13:25–45
64. Zaslavsky T (1975) Facing up to arrangements: face count formulas for partitions of space by hyperplanes. Memoirs Amer Math Soc 154:vii+102
65. Ziegler GM (1995) Lectures on Polytopes. Springer, New York

Convex Envelopes in Optimization Problems

YASUTOSHI YAJIMA
Tokyo Institute Technol., Tokyo, Japan

MSC2000: 90C26

Article Outline

Keywords
See also
References

Keywords

Convex underestimator; Nonconvex optimization

Let $f\colon S \to \mathbf{R}$ be a *lower semicontinuous function*, where $S \subseteq \mathbf{R}^n$ is a nonempty convex subset. The convex envelope taken over S is a function $f_S\colon S \to \mathbf{R}$ such that

- f_S is a convex function defined over the set S;
- $f_S(x) \le f(x)$ for all $x \in S$;
- if h is any other convex function such that $h(x) \le f(x)$ for all $x \in S$, then $h(x) \le f_S(x)$ for all $x \in S$.

In other words, f_S is the pointwise supremum among any *convex underestimators* of f over S, and is uniquely determined. The following demonstrates the most fundamental properties shown by [3,6]. Suppose that the minimum of f over S exists. Then,

$$\min\{f(x)\colon\ x \in S\} = \min\{f_S(x)\colon\ x \in S\}$$

and

$$\{x^*\colon\ f(x^*) \le f(x),\ \forall x \in S\} \subseteq \{x^*\colon\ f_S(x^*) \le f_S(x),\ \forall x \in S\}.$$

The properties indicate that an optimal solution of a nonconvex minimization problem could be obtained by minimizing the associated convex envelope. In general, however, finding the convex envelope is at least as difficult as solving the original one.

Several practical results have been proposed for special classes of objective functions and constraints. Suppose that the function f is concave and S is a polytope with vertices $v^0, \dots, v^K$. Then, the convex envelope f_S over S can be expressed as:

$$\begin{cases} f_S(x) = \min & \displaystyle\sum_{i=0}^{K} \alpha_i f(v^i) \\ \text{s.t.} & \displaystyle\sum_{i=0}^{K} \alpha_i v^i = x, \\ & \displaystyle\sum_{i=0}^{K} \alpha_i = 1, \\ & \alpha_i \ge 0, \quad i = 0, \dots, K. \end{cases}$$

Especially, if S is an n-simplex with vertices $v^0, \dots, v^n$, f_S is the affine function

$$f_S(x) = a^\top x + b,$$

which is uniquely determined by solving the following linear system

$$a^\top v^i + b = f(v^i), \quad i = 0, \dots, n.$$

The properties above have been used to solve concave minimization problems with linear constraints [4,6].

The following property shown in [1,5] is frequently used in the literature. For each $i = 1, \dots, p$, let $f^i\colon S_i \to \mathbf{R}$ be a continuous function, where $S_i \subseteq \mathbf{R}^{n_i}$, and let $n = n_1 + \cdots + n_p$. If

$$f(x) = \sum_{i=1}^{p} f^i(x^i)$$

and

$$S = S_1 \times \cdots \times S_p,$$

where $x^i \in \mathbf{R}^{n_i}$, $i = 1, \dots, p$, and $x = (x^1, \dots, x^p) \in \mathbf{R}^n$, then the convex envelope $f_S(x)$ can be expressed as:

$$f_S(x) = \sum_{i=1}^{p} f_S^i(x^i).$$

In particular, let $f(x) = \sum_{i=1}^{n} f_i(x_i)$ be a separable function, where $x = (x_1, \ldots, x_n) \in \mathbf{R}^n$, and let $f_i(x_i)$ be concave for each $i = 1, \ldots, n$. Then the convex envelope of $f(x)$ over the rectangle $R = \{x \in \mathbf{R}^n: a_i \le x_i \le b_i, i = 1, \ldots, n\}$ can be the affine function, which is given by the sum of the linear functions below:

$$f_R(x) = \sum_{i=1}^{n} l_i(x_i),$$

where $l_i(x_i)$ meets $f_i(x_i)$ at both ends of the interval $a_i \le x_i \le b_i$ for each $i = 1, \ldots, n$. B. Kalantari and J.B. Rosen [7] show an algorithm for the global minimization of a quadratic concave function over a polytope. They exploit convex envelopes of separable functions over rectangles to generate lower bounds in a *branch and bound scheme*.

Also, convex envelopes of *bilinear functions* over rectangles have been proposed in [2]. Consider the following rectangles:

$$\Omega_i = \left\{(x_i, y_i): \begin{array}{c} l_i \le x_i \le L_i, \\ m_i \le y_i \le M_i \end{array}\right\}, \qquad i = 1, \ldots, n,$$

and let

$$f^i(x_i, y_i) = x_i y_i, \quad i = 1, \ldots, n,$$

be bilinear functions with two variables. It has been shown that for each $i = 1, \ldots, n$, the convex envelope of $f^i(x_i, y_i)$ over Ω_i is expressed as:

$$f^i_{\Omega_i}(x_i, y_i) = \max\{m_i x_i + l_i y_i - l_i m_i, M_i x_i + L_i y_i - L_i M_i\}.$$

Moreover, it can be verified that $f^i_{\Omega_i}(x_i, y_i)$ agrees with $f^i(x_i, y_i)$ at the four extreme points of Ω_i. Thus, the convex envelope of the general bilinear function

$$f(x, y) = x^\top y = \sum_{i=1}^{n} f^i(x_i, y_i),$$

where $x^\top = (x_1, \ldots, x_n)$ and $y^\top = (y_1, \ldots, y_n)$ over $\Omega = \Omega_1 \times \cdots \times \Omega_n$ can be expressed as

$$f_\Omega(x, y) = \sum_{i=1}^{n} f^i_{\Omega_i}(x_i, y_i).$$

Another characterization of convex envelopes of bilinear functions over a special type of polytope, which includes a rectangle as a special case, is derived in [8].

See also

- ▶ αBB Algorithm
- ▶ Global Optimization in Generalized Geometric Programming
- ▶ MINLP: Global Optimization with αBB

References

1. Al-Khayyal FA (1990) Jointly constrained bilinear programs and related problems: An overview. Comput Math Appl 19(11):53–62
2. Al-Khayyal FA, Falk JE (1983) Jointly constrained biconvex programming. Math Oper Res 8(2):273–286
3. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: Theory and algorithms. Wiley, New York
4. Benson HP, Sayin S (1994) A finite concave minimization algorithm using branch and bound and neighbor generation. J Global Optim 5(1):1–14
5. Falk JE (1969) Lagrange multipliers and nonconvex programs. SIAM J Control 7:534–545
6. Falk JE, Hoffman K (1976) A successive underestimation method for concave minimization problems. Math Oper Res 1(3):251–259
7. Kalantari B, Rosen JB (1987) An algorithm for global minimization of linearly constrained concave quadratic functions. Math Oper Res 12(3):544–561
8. Sherali HD, Alameddine A (1990) An explicit characterization of the convex envelope of a bivariate bilinear function over special polytopes. Ann Oper Res 25(1–4):197–209

Convexifiable Functions, Characterization of[1]

SANJO ZLOBEC
Department of Mathematics and Statistics, McGill University, Montreal, Canada

MSC2000: 90C25, 90C26, 90C30, 90C31, 25A15, 34A05

Article Outline

Introduction
Definitions
Characterizations of a Convexifiable Function
Canonical Form of Smooth Programs
Other Applications
Conclusions
References

[1]Research partly supported by NSERC of Canada.

Introduction

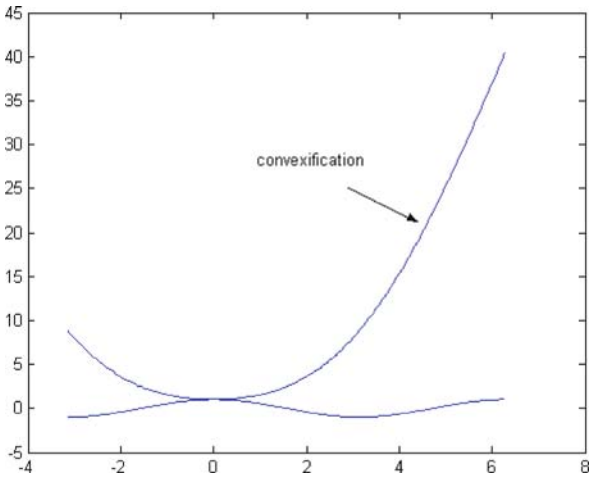
A twice continuously differentiable function in several variables, when considered on a compact convex set C, becomes convex if an appropriate convex quadratic is added to it, e. g. [2]. Equivalently, a twice continuously differentiable function is the difference of a convex function and a convex quadratic on C. This decomposition is valid also for smooth functions with Lipschitz derivatives [8]. Here we recall three conditions that are both necessary and sufficient for the decomposition [9,10]. We also list several implications of the convexification in optimization and applied mathematics [10,11]. A different notion of convexification is studied in, e. g. [6]; see also [3,5].

Definitions

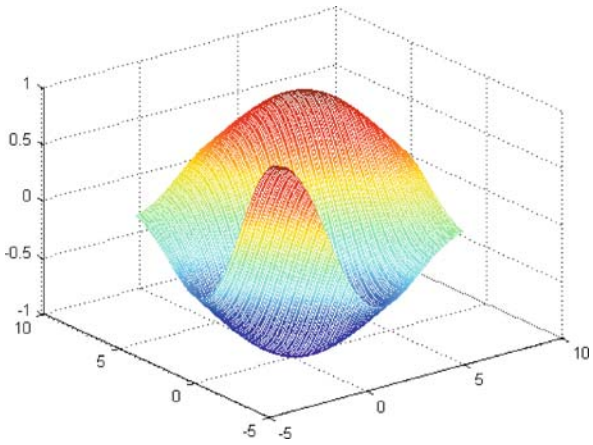
Definition 1. ([7,10]) Given a continuous function $f\colon \mathbb{R}^n \to \mathbb{R}$ on a compact convex set C of the Euclidean space $\mathbb{R}^n$, consider $\phi\colon \mathbb{R}^{n+1} \to \mathbb{R}$ defined by $\phi(x,\alpha) = f(x) - 1/2\alpha x^{\mathrm{T}}x$ where x^{T} is the transpose of x. If $\phi(x,\alpha)$ is convex on C for some $\alpha = \alpha^*$, then $\phi(x,\alpha)$ is said to be a *convexification* of f and α^* is its *convexifier* on C. Function f is *convexifiable* if it has a convexification.

Observation. If α^* is a convexifier of f on a compact convex set C, then so is every $\alpha < \alpha^*$.

Illustration 1. Consider $f(t) = \cos t$ on, say, $-\pi \le t \le 2\pi$. This function is convexifiable, its convexifier is any $\alpha \le -1$. For, e. g., $\alpha^* = -2$, its convexification is $\phi(t,-2) = \cos t + t^2$. Note that $f(t)$ is the difference of (strictly) convex $\phi(t,\alpha) = \cos t - 1/2\alpha t^2$ and (strictly) convex quadratic $-1/2\alpha x^{\mathrm{T}}x$ for every *sufficiently small* α. The graphs of $f(t)$ and its convexification $\phi(t,-2)$ are depicted in Fig. 1.



Convexifiable Functions, Characterization of, Figure 1
Function $f(t) = \cos t$ and its convexification



Convexifiable Functions, Characterization of, Figure 2
Mid-point acceleration function of $f(t) = \cos t$

Characterizations of a Convexifiable Function

One can characterize convexifiable functions using the fact that a continuous $f\colon \mathbb{R}^n \to \mathbb{R}$ is convex if, and only if, f is mid-point convex, i. e., $f((x+y)/2) \le 1/2(f(x)+f(y))$, $x, y \in C$, e. g. [4]. Denote the norm of $u \in \mathbb{R}^n$ by $||u|| = (u^{\mathrm{T}}u)^{1/2}$. With a continuous $f\colon \mathbb{R}^n \to \mathbb{R}$ one can associate $\Psi\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$:

Definition 2. ([10]) Consider a continuous $f\colon \mathbb{R}^n \to \mathbb{R}$ on a compact convex set C in $\mathbb{R}^n$. *The mid-point acceleration function* of f on C is the function

$$\Psi(x,y) = (4/||x-y||^2)[f(x) + f(y) - 2f((x+y)/2)], \quad x, y \in C, x \neq y .$$

Function Ψ describes a mid-point "displacement of the displacement" (i. e., the "acceleration") of f at x between x and y along the direction $y-x$. The graph of Ψ for the scalar function $f(t) = \cos t$ is depicted in Fig. 2.

Using Ψ one can characterize a convexifiable function:

Theorem 1. ([10]) *Consider a continuous $f\colon \mathbb{R}^n \to \mathbb{R}$ on a compact convex set C in $\mathbb{R}^n$. Function f is convexifi-*

able on C if, and only if, its mid-point acceleration function Ψ is bounded from below on C.

For scalar functions one can also use a determinant:

Theorem 2. ([9] Determinant Characterization of Scalar Convexifiable Function) *A continuous scalar function $f: \mathbb{R} \to \mathbb{R}$ is convexifiable on a compact convex interval I if, and only if, there exists a number α such that for every three points $s < t < \xi$ in I*

$$\det \begin{pmatrix} 1 & 1 & 1 \\ s & t & \xi \\ f(s) & f(t) & f(\xi) \end{pmatrix} \geq \frac{1}{2}\alpha(s-t)(t-\xi)(\xi-s)\,.$$

Illustration 2. Function $f(t) = -|t|^{3/2}$ on $C = [-1, 1]$ is continuously differentiable but it is not convexifiable. Indeed, for $s = 0, \Psi(0,t) = 2^{5/2}(1 - 2^{1/2})/t^{-1/2} \to -\infty$ as $t > 0, t \to 0$. Also, using Theorem 2 at $s = -\epsilon$, $t = 0, \xi = \varepsilon > 0$, we find that there is no α such that $\alpha \leq -2/\epsilon$ as $\varepsilon \to 0$. Function $g(t) = -|t|$ is not convexifiable around the origin.

Scalar convexifiable functions can be represented explicitly on a compact interval I:

Theorem 3. ([9] Explicit Representation of Scalar Convexifiable Functions) *A continuous scalar function $f: I \to \mathbb{R}$ is convexifiable if, and only if, there exists a number α such that*

$$f(t) = f(c) + \frac{1}{2}\alpha(t^2 - c^2) + \int_c^t g(\xi, \alpha)\,\mathrm{d}\xi\,.$$

Here $c, t \in I, c < t$, and $g = g(\cdot, \alpha): I \to \mathbb{R}$ is a non-decreasing right-continuous function.

An implication of this result is that every smooth function with a Lipschitz derivative, in particular every analytic function and every trajectory of an object governed by Newton's Second Law, is of this form.

Two important classes of functions are convexifiable and a convexifier α can be given explicitly. First, if f is twice continuously differentiable, then the second derivative of f at x is represented by the Hessian matrix $H(x) = (\partial^2 f(x)/\partial x_i \partial x_j)$, $i, j = 1, \ldots, n$. This is a symmetric matrix with real eigenvalues. Denote its smallest eigenvalue at x by $\lambda(x)$ and its "globally" smallest eigenvalue over a compact convex set C by

$$\lambda^* = \min_{x \in C} \lambda(x)\,.$$

Corollary 1. ([7]) *A twice continuously differentiable function $f: \mathbb{R}^n \to \mathbb{R}$ is convexifiable on a compact convex set C in $\mathbb{R}^n$ and $\alpha = \lambda^*$ is a convexifier.*

Suppose that f is a continuously differentiable (smooth) function with the derivative satisfying the Lipschitz property, i. e., $||\nabla f(x) - \nabla f(y)|| \leq L||x - y||$ for every $x, y \in C$ and some constant L. Here $\nabla f(u)$ is the (Frechet) derivative of f at u. We represent the derivative of f at x by a column n-tuple gradient $\nabla f(x) = (\partial f(x)/\partial x_i)$.

Corollary 2. ([8]) *A continuously differentiable function $f: \mathbb{R}^n \to \mathbb{R}$, with the derivative having the Lipschitz property with a constant L on a compact convex set C in $\mathbb{R}^n$, is convexifiable on C and $\alpha = -L$ is a convexifier.*

A Lipschitz function may not be convexifiable. For example, $f(t) = t^2 \sin(1/t)$ for $t \neq 0$ and $f(0) = 0$ is a Lipschitz function and it is also differentiable (not continuously differentiable). Its derivative is uniformly bounded, but the function is not convexifiable, e. g. [7,11].

Canonical Form of Smooth Programs

Every mathematical program (NP)

$$\mathrm{Min} f(x),\ f^i(x) \leq 0, \quad i \in P = \{1, \ldots, m\}, x \in C\,,$$

where the functions $f, f^i: \mathbb{R}^n \to \mathbb{R}, i \in P$ are continuous and convexifiable on a compact convex set C can be reduced to a canonical form. First one considers some convexifications of these functions: $\phi(x, \alpha) = f(x) - 1/2\alpha x^{\mathrm{T}} x$ and $\phi^i(x, \alpha_i) = f^i(x) - 1/2\alpha_i x^{\mathrm{T}} x$, where α, α_i are, respectively, arbitrary convexifiers of $f, f^i, i \in P$. Then one associates with (NP) the following program with partly linear convexifications

$$(\mathrm{LF}; \theta, \varepsilon):$$
$$\mathrm{Min}_{(x,\theta)} \phi(x, \alpha) + \frac{1}{2}\alpha x^{\mathrm{T}}\theta\,,$$
$$\phi^i(x, \alpha_i) + \frac{1}{2}\alpha_i x^{\mathrm{T}}\theta \leq 0, \quad i \in P\,,$$
$$x \in C, \|x - \theta\| \leq \varepsilon\,.$$

Here $\varepsilon \geq 0$ is a scalar parameter. This parameter was fixed at zero value in [2]. For the sake of "numerical stability" it was extended to $\varepsilon \geq 0$ in [7]. If the norm is chosen to be uniform, i. e., $\|u\|_\infty = \max_{i=1,\ldots,n} |u_i|$

then (LF; θ, ε) is a convex program in x for every fixed (θ, ε) and linear in (θ, ε) for every x. Such programs are called partly linear-convex. Theory of optimality and stability for such programs and related models is well studied, e. g. [1,8].

Remark Since one can construct the program (LF; θ, ε) for every (NP) with convexifiable functions, we refer to (LF; θ, ε) as the *parametric Liu–Floudas canonical form* of (NP).

Let us relate an optimal solution of (NP) to optimal solutions $x^0(\varepsilon)$, $\theta^0(\varepsilon)$ of (LF; θ, ε).

Theorem 4. ([8,9]) *Consider (NP) with a unique optimal solution, where all functions are assumed to be convexifiable, and its partly linear-convex program (LF; θ, ε). Then a feasible x^* is an optimal solution of (NP) if, and only if, $x^* = \lim_{\epsilon\to 0} x^0(\varepsilon)$ and $\theta^* = \lim_{\varepsilon\to 0} \theta^0(\varepsilon)$, with $x^* = \theta^*$. Moreover, the feasible set mapping of (LF; θ, ε) is lower semi-continuous at θ^* and $\varepsilon^* = 0$, relative to all feasible perturbations of (θ, ε).*

Other Applications

There are many other areas of applications of convexifiable functions:

(i) Every convexifiable function f is the difference of a convex function $\phi(x, \alpha)$ and a convex quadratic $1/2\alpha x^{\mathrm{T}} x$ for every sufficiently small α on a compact convex set. Hence it follows that the results for convex functions can be applied to $\phi(x, \alpha)$. With minor adjustments, pertaining to the quadratic term, such results can be extended to convexifiable (generally non-convex) functions. Here is an illustration of how this works for the mean value. The result is well known for convex functions (the case $\alpha = 0$).

Theorem 5. ([9]) *Consider a continuous scalar convexifiable function $f\colon \mathbb{R} \to \mathbb{R}$ on an open interval (a, b) with a convexifier α. Then*

$$1/(d-c)\int_c^d f(\xi)\,\mathrm{d}\xi \le \frac{1}{2}[f(c)+f(d)] - \frac{1}{12}\alpha(d-c)^2\,, \quad \text{for every} \quad a < c < d < b\,.$$

A composite version of this result follows.

Theorem 6. ([9]) *Integral Mean-Value for Composite Convexifiable Function) Let $f\colon (a, b) \to \mathbb{R}$ be continous and convexifiable with a convexifier α and let $g\colon [c, d] \to (a, b)$ be continuous. Then*

$$f\left(1/(d-c)\cdot\int_c^d g(t)\,\mathrm{d}t\right) \le 1/(d-c)\cdot\int_c^d (f\circ g)(t)\,\mathrm{d}t + \frac{1}{2}\alpha\cdot R(c,d;g)$$

where

$$R(c,d;g) = [1/(d-c)\cdot\int_c^d g(t)\,\mathrm{d}t]^2 - 1/(d-c)\cdot\int_c^d [g(t)]^2\,\mathrm{d}t\,.$$

Remarks If $f\colon (a, b) \to \mathbb{R}$ and $g\colon [c, d] \to (a, b)$ are continuous on (a, b), then, in Theorem 5 and 6, one can specify $\alpha = 0$, if f is convex on (a, b). Also $\alpha = \lambda^* = \min_{t\in I} f''(t)$, if f is twice continuously differentiable or analytic on (a, b), and $\alpha = -L$ a negative Lipschitz constant of the derivative of f on (a, b), if f is continuously differentiable.

(ii) Convexification of differential equations: A solution $y(t)$ of an ordinary differential equation of second or higher order, over a compact interval, is continuously differentiable with a Lipschitz derivative. Such y is convexifiable. Using $y(t) = \phi(t, \alpha) + 1/2\alpha t^2$, the problems in differential equations can be "convexified", i. e., transformed to equivalent differential equations with convex solutions $\phi(t, \alpha)$. After back-substitution, the true solution $y(t)$ is recovered. In particular, the problems of theoretical mechanics based on the Second Newton Law, can be "convexified", e. g. [9,11].

(iii) Convexification in linear algebra: Some of the basic eigenvalue inequalities for symmetric matrices follow from inequalities for convex functions, e. g. [4]. Using a convexification, one can extend these results to non-convexity. For example, after finding a convexifier of the product function $f(x) = x_1 x_2 \cdots x_n$, the following follows:

Corollary 3. (Bounds for the Determinant of an Arbitrary Symmetric Matrix [11]) *Let $A = (a_{ij})$ be an*

$n \times n$ real symmetric matrix where $n \geq 2$ and let ρ be its spectral radius. Then

$$\prod_{i=1,\ldots,n} a_{ii} - (n-1)\rho^{n-2} \sum_{i,j=1,\ldots,n;i<j} a_{ij}^2 \leq \det A \leq \prod_{i=1,\ldots,n} a_{ii} + (n-1)\rho^{n-2} \sum_{i,j=1,\ldots,n;i<j} a_{ij}^2 .$$

If the left hand-side in Corollary 3 is positive, then the matrix A is non-singular.

From the two-sided Jensen inequalities, obtained by convexification, new estimates follow for the absolute value of the inner product function in arbitrary inner product spaces. In some situations the new estimates are sharper than the Cauchy–Schwarz inequality. Many results for convexifiable functions, including the canonical form (LF; θ, ε), can be extended to non-smooth Lipschitz functions. This can be done using the fact that every Lipschitz function, when considered on a compact convex set, is only a linear function away from the set of all coordinate-wise monotone functions [11].

Conclusions

A necessary and sufficient condition for convexifiability on a compact convex set is given using the mid-point acceleration function. For scalar functions there are also characterizations given in terms of determinants and integrals. In particular, every smooth function with a Lipschitz derivative is convexifiable. Such function is only a convex quadratic away from the set of all convex functions. Using this "closeness", many results for convex functions can be extended to smooth (generally non-convex) functions with Lipschitz derivatives. On the other hand, mathematical programs with convexifiable functions can be reduced to the parametric Liu-Floudas partly linear-convex canonical form.

References

1. Floudas CA (2000) Deterministic Global Optimization. Kluwer, Dordrecht
2. Liu WB, Floudas CA (1993) A remark on the GOP algorithm for global optimization. J Global Optim 3:519–521
3. Pardalos PM (1988) Enumerative techniques in global optimization. Oper Res Spektr 10:29–35
4. Roberts AW, Varberg DE (1973) Convex Functions. Academic Press, New York
5. Vial JP (1983) Strong and weak convexity of sets and functions. Math Oper Res 8(2):231–259
6. Wu ZY, Lee HWJ, Yang XM (2005) A class of convexification and concavification methods for non-monotone optimization problems. Optimization 54:605–625
7. Zlobec S (2003) Estimating convexifiers in continuous optimization. Math Commun 8:129–137
8. Zlobec S (2005) On the Liu-Floudas convexification of smooth programs. J Glob Optim 32:401–407
9. Zlobec S (2005) Convexifiable functions in integral calculus. Glasnik Matematički 40(60):241–247, Erratum (2006) Ibid 41(61):187–188
10. Zlobec S (2006) Characterization of convexifiable functions. Optimization 55:251–262
11. Zlobec S (2008) On two simple decompositions of continuous functions. Optimization 57(2):249–261

Convex Max-Functions

CLAUDIA SAGASTIZÁBAL
IMPA, Jardim Botânico, Brazil

MSC2000: 49K35, 49M27, 65K10, 90C25

Article Outline

Keywords
Synonyms
Examples of the Problem
Continuity and Optimality Conditions
Algorithms of Minimization
  Nonlinear Programming
  Nonsmooth Optimization
  Other Methods of Resolution
See also
References

Keywords

Minimax problem; Convex optimization; Max-function

Synonyms

Minimax

Examples of the Problem

In a classic nonlinear program (NLP) a smooth objective function is minimized on a feasible set defined by finitely many smooth constraints. However, many optimization problems have an objective function that is

not smooth but is of the max type, that is, defined as

$$f(x) := \max\{f_\ell(x): \ \ell \in \mathcal{L}\}, \tag{1}$$

where the functions $f_\ell: \mathbf{R}^n \to \mathbf{R}$ are themselves smooth.

When $\mathcal{L}$ is finite, the minimization of f is called a *finite minimax problem*. In a general *minimax problem*, the indices ℓ can range over an infinite compact set $\mathcal{L}$, see [6]. Those f_ℓ's realizing the maximum in (1) are called *active functions*. The corresponding indices form the *active index set*, defined as $\mathcal{L}_a(x) := \{l \in \mathcal{L}: f(x) = f_l(x)\}$.

There are numerous examples of optimization problems dealing with the minimization of convex max-functions:

- When processing empirical data $\{(t_\ell, p_\ell): \ell = 1, \ldots, m\}$, consider the problem of selecting the coefficients x of a polynomial $p_x(t) = \sum_{j=0}^{n} x_{j+1} t^j$ that fits well the data. The quality of the approximation can be measured through the deviation $f_\ell(x) := |p_\ell - p_x(t_\ell)|$, defined for all ℓ. Depending on the nature of the problem, it can be interesting to minimize either the sum of squared deviations or the maximum deviation. The first case is a *least squares problem*, while the second is known as *Chebyshev best approximation*, and is a particular instance of (1), with index set $\ell = \{\ell: \ell = 1, \ldots, m\}$.
- A more general case is finding the best approximation of a continuous function φ_0 on a compact interval $\mathcal{L}$ Instead of n powers t^j, n linearly independent functions $\varphi_j: \mathcal{L} \to \mathbf{R}$ are given. To find the linear combination $\sum_j x_j \varphi_j$ which best approximates φ_0 in the max-norm comes to solve (1), with $f_\ell(x) = |\sum_j x_j \varphi_j(\ell) - \varphi_0(\ell)|$. Because the problem has infinitely many constraints, it is also an example of semi-infinite programming.
- A basic problem in structural optimization, see [2], is to find the stiffest structure of a given volume that is able to carry loads varying on a given set. Optimization can be performed through the variation of sizing variables, like the thickness of bars in a truss; shape variables, as the splines defining the boundary of the body; or even the distribution and properties of the composite material used to make the structure itself. After discretization by finite elements, the design problem has an objective function as in (1). The f_ℓ's therein have usually the form $f_\ell(x) = (1/2)x^\mathsf{T} A_\ell x - b^\mathsf{T} x$, with A_ℓ denoting the stiffness matrix of the ℓth element of the grid.
- Some large scale mixed integer problems can be solved by *decomposition techniques* using Lagrangian relaxation. The idea is to coordinate, by means of a master program, the iterative resolution of problems of smaller dimension or complexity, called local problems. When applying price decomposition, the master is led to maximize a dual function, say $\theta(\lambda)$, on the space of multipliers Λ, using some iterative method. In production planning problems, the iterate λ^k sent to the local solvers can be interpreted as prices paid by the master. Let $\ell = \prod_i \mathcal{L}^i$ denote the (decomposed) primal space and let $L(x, \lambda) = \sum_i L^i(x^i, \lambda)$ be the (decomposed) Lagrangian of the original problem. Each local unit i decides its corresponding optimal level of production by solving $\min_{x^i \in \mathcal{L}^i} L^i(x^i, \lambda^k)$. Having those local optimal levels, the master adjusts prices by updating λ^k in order to maximize the dual function θ, defined as the pointwise minimum of the Lagrangian: $\theta(\lambda) = \min_{x \in \mathcal{L}} L(x, \lambda) = \sum_i \min_{x^i \in \mathcal{L}^i} L^i(x^i, \lambda)$. An equivalent problem for the master is to minimize $f(\lambda) := -\theta(\lambda)$. This last formulation has an objective f that is a max-function as in (1), letting $f_\ell(\lambda) = -\sum_i L^i(x^i_\ell, \lambda)$, for each $x_\ell \in \mathcal{L}$.
- Solving a nonlinear program using *exact penalties* leads to the iterative minimization of penalized objective which are max-functions, with the max-operation involving the constraints.
- In game theory, consider a zero sum game, with two players whose strategy is to optimize their individual choice against the worst possible selection by the other player. The first player can choose his action over n possible moves, with probability distribution $x = (x_1, \ldots, x_n)$. To every possible move $j = 1, \ldots, m$ of the second player, corresponds a loss $a_{i,j}$ player I pays to player II. Let ℓ be a continuous index counting elements in the set of probability distributions of the second player: $\mathcal{L} = \{z \in \mathbf{R}^m: \sum_{j=1}^m z_j = 1, z_j \geq 0\}$. Calling $A = [a_{i,j}]$ the $n \times m$ loss-matrix, the average amount to be paid by player I is $f_\ell(x) = x^\mathsf{T} A z_\ell$. It follows that the first player needs to solve a problem like (1). The mini-maximization of the bivariate function $F(x, z) = x^\mathsf{T} A z$ is also called a *saddle-point problem*.

Continuity and Optimality Conditions

When taking the pointwise maximum in (1), some properties of the functions f_ℓ are transmitted to f. Such is the case of continuity and convexity, but not of differentiability.

More precisely, f is continuous when both the f_ℓ-s and its gradients ∇f_ℓ are continuous. When the underlying functions f_ℓ are convex, so is f.

Convexity implies that the max-function f is differentiable almost everywhere. Nevertheless, at those points where more than one underlying function f_l realizes the maximum, the gradient fails to exist. Typically, such is the case at $\overline{x}$, a minimizer of f. For instance, suppose that $\mathcal{L}_a(\overline{x}) = \{1, 2\}$, i. e., there are two active functions at $\overline{x}$: $f(\overline{x}) = f_1(\overline{x}) = f_2(\overline{x})$. Clearly, for $\nabla f(\overline{x})$ to exist, the unlikely equality $\nabla f(\overline{x}) = \nabla f_1(\overline{x}) = \nabla f_2(\overline{x})$ needs to hold. Moreover, the optimality condition for minimizing f on $\mathbf{R}^n$ would require all the involved gradients to be null.

Rather than a single gradient, it is possible to define a whole set of *subgradients* by making convex combinations of $\nabla f_1(\overline{x})$ and $\nabla f_2(\overline{x})$. For an arbitrary convex max-function f, at any given x, the set of subgradients is the so-called *subdifferential* of f at x. Its expression is given by the formula

$$\partial f(x) = \left\{ \sum_{l \in \mathcal{L}_a(x)} \alpha_l \nabla f_l(x) \colon \alpha \in \Delta \right\}, \tag{2}$$

where Δ is the unit simplex

$$\Delta := \left\{ \alpha \in \mathbb{R}^{|\mathcal{L}_a(x)|} \colon \sum_{l \in \mathcal{L}_a(x)} \alpha_l = 1,\ \alpha_l \geq 0 \right\}. \tag{3}$$

When $\mathcal{L}$ in (1) is an infinite compact set, (2) still holds, provided the application $\ell \longmapsto f_\ell(x)$ is upper semicontinuous for each x, see [8, Chap. VI, §4.4].

Consider the constrained problem

$$\min_{x \in \Omega} f(x), \tag{4}$$

where $\Omega \subset \mathbf{R}^n$ is a closed convex set and f is the function defined in (1). Assume the index set $\mathcal{L}$ is infinite and suppose (4) has a solution $\overline{x}$ such that $f(\overline{x}) = \max\{f_\ell(\overline{x}) \colon \ell \in \mathcal{L}\}$. Then it can be proved (see [6, Chap. VI, Thm. 3.3]) that (4) is equivalent to the finite minimax

$$\min_{x \in \Omega} \max_{i=1,\ldots,r} \left\{ f_{\ell_i}(x) \colon \ell_i \in \mathcal{L} \right\}, \tag{5}$$

with $r \leq n + 1$. The set $\{\ell_1, \ldots, \ell_r\}$ is called an *extremal basis* of (4).

When Ω satisfies a constraint qualification condition of Slater type, see, for instance, [7, Chap. III], a necessary optimality condition (OC) characterizing a minimizer $\overline{x}$ of (4) is

$$0 \in \partial f(\overline{x}) + N_\Omega(\overline{x}), \tag{6}$$

where N is the normal cone of convex analysis. Because f is convex, the optimality condition is also sufficient.

The optimality condition (6) can be further specified when Ω is represented by a set of convex inequalities:

$$\Omega := \left\{ x \in \mathbb{R}^n \colon c_j(x) \leq 0,\ j \in \mathcal{J} \right\}. \tag{7}$$

Observe that Ω may contain an infinite number of constraints c_j, assumed to be smooth and convex. Using (1) together with (7), the following characterization of $\overline{x}$ results:

Lemma 1 *There exist $r \leq n + 1$ and $s \leq n$ such that for the index sets $\mathcal{L}_a(\overline{x}) := \{l_1, \ldots, l_r\} \subset \mathcal{L}$ and $\{j_1, \ldots, j_s\} \subset \mathcal{J}$ it holds*

$$\sum_{i=1}^{r} \alpha_i \nabla f_{l_i}(\overline{x}) + \sum_{i=1}^{s} \mu_i \nabla c_{j_i}(\overline{x}) = 0, \tag{8}$$

where the multipliers α and μ are positive and α is an element of the simplex Δ in $\mathbb{R}^{|\mathcal{L}|_a(\overline{x})}$ from (3).

This characterization ensures the existence of an extremal basis of (4) near $\overline{x}$.

Algorithms of Minimization

Depending on the nature of the problem, several approaches have been proposed to solve (4):

- Reformulation as a NLP.
- Minimization of the nonsmooth max-function.
- Determination of a saddle point.
- Search of an extremal basis.

For example, the amount of available information can determine the method of resolution: if for any given x, all the active indices in (1) are known, then the full subdifferential (2) is available and a nonlinear programming technique can be applied.

Nonlinear Programming

An important feature of this approach is that smooth NLP techniques have a superlinear rate of convergence. The essential idea is first to write (4) as an NLP with an additional variable:

$$\begin{cases} \min\limits_{\substack{r\in\mathbb{R}\\ x\in\Omega}} & r \\ \text{s.t.} & r \geq f_\ell(x), \quad \ell \in \mathcal{L}, \end{cases} \tag{9}$$

and then solve the associated optimality conditions by using a Newton-like method, such as sequential quadratic programming (cf. also ► Successive quadratic programming) or interior point schemes, see [4, Parts III–IV] and [3, §§4.3–4.4].

Nonsmooth Optimization

Sometimes the explicit knowledge of all the active constraints in (9) can be difficult, if not impossible, to obtain; such is the case for structural optimization problems.

On the other hand, it is often possible to obtain a single subgradient almost for free when computing $f(x)$. Indeed, suppose that just one active index l in $\ell_a(x)$ is known: $f(x) = f_l(x)$. Then, because of (2), $\nabla f_l(x) \in \partial f(x)$.

Algorithms from nonsmooth optimization, such as bundle methods [8, Chaps. XIV–XV], are designed to minimize a general convex function, possibly nondifferentiable, with the information furnished by an oracle that gives $f(x)$ and only one subgradient at x. Nonsmooth optimization techniques, specialized to a max-function like f in (1), have been successfully used in [12] and [9] to solve general minimax problems.

Although bundle methods are essentially first order methods, in recent years some proposals have been given that aim at obtaining better than linear convergence. They consist of a combination of bundle, proximal and quasi-Newton techniques [5,10,11].

Other Methods of Resolution

V.F. Demyanov and V.N. Malozemov treat (4) in an indirect way, by solving an infinite sequence of finite minimax problems. Keeping (5) in mind, the idea is to asymptotically identify an extremal basis by making successive approximations on a finite grid of the index set $\mathcal{L}$.

In game theory, rather than solving (4) by some 'mini-maximization' procedure, it can be more convenient to find a *saddle point*. That is, an equilibrium point satisfying $\min_x \max_{z_\ell} F(x, z_\ell) = \max_{z_\ell} \min_x F(x, z_\ell)$, with $F(x, z_\ell) := f_\ell(x)$. The determination of saddle points of F can be performed taking advantage of the extra structure of the problem. Some popular methods are Arrow–Hurwicz's and Uzawa's, see [1].

See also

► Lagrangian Multipliers Methods for Convex Programming

References

1. Arrow KJ, Hurwicz L, Uzawa H (1958) Studies in linear and nonlinear programming. Stanford Univ Press, Palo Alto, CA
2. Bendsøe MP (1995) Optimization of structural topology, shape and material. Springer, Berlin
3. Bertsekas DP (1995) Nonlinear programming. Athena Sci, Belmont, MA
4. Bonnans JF, Gilbert JCh, Lemaréchal C, Sagastizábal C (1997) Optimisation numérique: Aspects théoriques et pratiques. Springer, Berlin
5. Chen X, Fukushima M (1999) Proximal quasi-Newton methods for nondifferentiable convex optimization. Math Program 85(2):313–334
6. Dem'yanov VF, Malozemov VN (1974) Introduction to minimax. Wiley, New York
7. Hiriart-Urruty J-B (1996) L'optimisation. Que sais-je?, vol 3184. Press. Univ. France , Paris
8. Hiriart-Urruty J-B, Lemaréchal C (1993) Convex analysis and minimization algorithms. Grundl Math Wiss, vol 305-306. Springer, Berlin
9. Kiwiel KC (1987) A direct method of linearization for continuous minimax problems. J Optim Th Appl 55:271–287
10. Lemaréchal C, Sagastizábal C (1997) Variable metric bundle methods: from conceptual to implementable forms. Math Program 76:393–410
11. Mifflin R, Sun DF, Qi LQ (1998) Quasi-Newton bundle-type methods for nondifferentiable convex optimization. SIAM J Optim 8(2):583–603
12. Panin VM (1981) Linearization method for continuous minmax problems. Kibernetika 2:75–78

Convex-Simplex Algorithm

SIRIPHONG LAWPHONGPANICH
Naval Postgraduate School, Monterey, USA

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Basic component; Convex-simplex algorithm; Extreme point; Feasible direction methods; First order Taylor series expansion; Generalized networks; Generalized upper bounding structure; Golden section method; Improving feasible direction; Inexact line search technique; Karush–Kuhn–Tucker conditions; Linearly independent; Line search problem; Linear program; Nonbasic component; Nonlinear network flow problems; Nonlinear programming; Nonsingular; Polyhedron; Pseudoconvex function; Reduced gradient algorithm; Second order approximation; Simplex algorithm; Superbasic variables

W.I. Zangwill [9] first proposed the *convex-simplex algorithm* (CSA) for the following problem:

$$\min_{x \in S} f(x), \tag{1}$$

where $f(x)$ is a *pseudoconvex function* on $\mathbf{R}^n$. The set S is a nonempty *polyhedron*, i. e., $S = \{x \in \mathbf{R}^n : Ax = b, x \geq 0\}$, A is a $m \times n$ matrix, and b is a vector in $\mathbf{R}^m$. For simplicity, S is assumed to be bounded also.

CSA belongs to a class of algorithms called *feasible direction methods*. Given an initial feasible solution, algorithms in this class solve problem (1) by iteratively generating an improving feasible direction that leads to another feasible solution with an improved objective value. The name 'convex-simplex' is to indicate that the algorithm generates improving feasible directions in manner similar to the simplex algorithm for linear programs. When $f(x)$ is linear, the algorithm is identical to the simplex algorithm. In general, a vector d is an *improving feasible direction* at a point, x, feasible to problem (1) if the following (sufficient) conditions hold

a) $\nabla f(x)^{\intercal} d < 0$,
b) $Ad = 0$, and
c) $d_j \geq 0$ if $x_j = 0$.

It follows from the first order Taylor series expansion of $f(x)$ that

$$f(x + \lambda d) = f(x) + \lambda \nabla f(x)^{\top} d + \lambda \|d\| \alpha(x; \lambda d),$$

where $\lim_{\lambda \to 0} \alpha(x; \lambda d) = 0$. Via the above expansion, condition a) implies that $f(x + \lambda d) < f(x)$ for a sufficiently small $\lambda > 0$, i. e., d leads to an improvement in the objective function. The remaining two conditions guarantee that d can produce a point in S. In particular, condition b) yields the following:

$$A(x + \lambda d) = Ax + \lambda Ad = Ax = b.$$

This shows that $x + \lambda d$ is always feasible with respect to the equality constraint. Next, each component of $x + \lambda d$ can be written as

$$x_i + \lambda d_i = \begin{cases} x_i + \lambda d_i & \text{if } x_i > 0, \\ \lambda d_i & \text{if } x_i = 0. \end{cases}$$

When λ is a sufficiently small positive number, $x_i + \lambda d_i$ remains nonnegative in the first case. For the second case, it follows directly from condition c) that $\lambda d_i \geq 0$ for all $\lambda > 0$. Thus, $x + \lambda d \in S$ when λ is sufficiently small.

To describe how CSA generates an improving feasible direction, let a_j denote the jth column of A. Also, assume that every m columns of A are linearly independent and every extreme point of S has m strictly positive components. Under these assumptions, every feasible solution has at least m positive components and at most $(n - m)$ zero components. Given a feasible solution x, let $I(x)$ be the set of indices for the m largest components of x. Then, A can be partitioned into $[B, N]$, where $B = [a_j : j \in I(x)]$ and $N = [a_j : j \notin I(x)]$. Similarly, $x^{\top}$ can be partitioned into $[x_B^{\top}, x_N^{\top}]$ where $x_B^{\top}$, the *basic component*, corresponds to components of x belonging to $I(x)$, and $x_N^{\top}$, the *nonbasic component*, corresponds to components not in $I(x)$. By the above assumptions, $x_B^{\top} > 0$ and B is nonsingular.

Partitioning the direction vector, $d^{\top}$, into its basic and nonbasic components, i. e., $[d_B^{\top}, d_N^{\top}]$, produces the following sequence of relationships:

$$\begin{aligned} &Ad = 0, \\ &Bd_B + Nd_N = 0, \\ &d_B = -B^{-1} N d_N. \end{aligned}$$

The last equality yields the following:

$$\begin{aligned} \nabla f(x)^{\top} d &= \nabla_B f(x)^{\top} d_B + \nabla_N f(x)^{\top} d_N \\ &= [\nabla_N f(x)^{\top} - \nabla_B f(x)^{\top} B^{-1} N] d_N \\ &= r_N^{\top} d_N = \sum_{j \notin I(x)} r_j d_j, \end{aligned}$$

where $r_N^\top \equiv \nabla_N f(x)^\top - \nabla_B f(x)^\top B^{-1} N$. In order for d to be an improving direction, $\nabla f(x)^\top d < 0$. There are several approaches to make this inner product negative. Each approach generates a different algorithm and all of which can be viewed as an extension or variant of the *reduced gradient algorithm* first proposed by P. Wolfe [8].

Like the simplex algorithm, CSA allows only one nonbasic component of d to be nonzero. In particular, let

$$j^+ = \operatorname*{argmax}_{j \notin I(x)} \{-r_j : r_j < 0\}$$

and

$$j^- = \operatorname*{argmax}_{j \notin I(x)} \{x_j r_j : x_j > 0, r_j > 0\}.$$

Then, CSA chooses d_N as follows. If $-r_{j^+} \geq x_{j^-} r_{j^-}$, then $d_{j^+} = 1$ and $d_j = 0$ for the remaining nonbasic components. This makes $\nabla f(x)^\top d = r_{j^+} < 0$. Otherwise, $d_{j^-} = -1$ instead and $\nabla f(x)^\top d = -r_{j^-} < 0$. Given d_N, the basic component can be computed using the relationship $d_B = -B^{-1} N d_N$.

When $r_N \geq 0$ and $r_N^\top x_N = 0$, the indices j^+ and j^- are undefined in the above construction. When this occurs, x is globally optimal and d_N is usually set to zero to indicate that there is no improving feasible direction. To demonstrate, it is sufficient to show that there exist vectors $\mu^\top = [\mu_B^\top, \mu_N^\top] \geq 0$ and ν (unrestricted) satisfying the following equations:

$$\nabla_B f(x) + B^\top \nu - \mu_B = 0,$$
$$\nabla_N f(x) + N^\top \nu - \mu_N = 0,$$
$$\mu_B^\top x_B = 0,$$
$$\mu_B^\top x_N = 0.$$

These equations are known as the *Karush–Kuhn–Tucker conditions* (see, e. g., [2]) and they are sufficient optimality conditions for problem (1). Letting $\mu_B = 0$, $\mu_N = \nabla_N f(x) - N(B^\top)^{-1} \nabla_N f(x)$, and $\nu = -(B^\top)^{-1} \nabla_B f(x)$ satisfies the first three conditions. Since $\mu_N = r_N$, the above assumptions concerning r_N imply that $\mu_N \geq 0$ and $\mu_N^\top x_N = 0$. Thus, the Karush–Kuhn–Tucker conditions hold and x must be globally optimal.

When $d_N \neq 0$, a better feasible point can be obtained from a solution to the following problem, typically called the *line search problem*:

$$\min_{0 \leq \lambda \leq \lambda_{\max}} f(x + \lambda d),$$

where $\lambda_{\max} = \min_j \{-x_j/d_j : d_j < 0\}$. This prevents components of $x + \lambda d$ from being negative. (If S is unbounded, then every component of d may be nonnegative and $\lambda_{\max} = \infty$.) Algorithms such as the bisection search, the golden section method, and an inexact line search technique (e. g., the *Armijo rule*, [1]) can efficiently solve the line search problem.

To summarize, CSA can be stated as follows:

0 Select $x^1 \in S$ and set $k = 1$.

1 Identify $I(x^k)$ and form the submatrices B and N.
Compute

$$r_N^\top = \nabla_N f(x^k)^\top - \nabla_B f(x^k)^\top B^{-1} N.$$

2 IF $r_N \geq 0$ and $r_N^\top x_N^\top = 0$,
THEN stop and x^k is an optimal solution.
ELSE, let

$$j^+ = \operatorname*{argmax}_{j \notin I(x^k)} \{-r_j : r_j < 0\},$$

$$j^- = \operatorname*{argmax}_{j \notin I(x^k)} \{x_j^k r_j : x_j^k > 0, r_j > 0\}.$$

Set $d_N^k = 0$.
IF $-r_{j^+} \geq x_{j^-}^k r_{j^-}$,
THEN set $d_{j^+}^k = 1$.
ELSE, set $d_{j^-}^k = -1$ instead.
Set $d_B^k = B^{-1} N d_N^k$. Go to Step 3.

3 Set $\lambda_{\max} = \min_j \{-x_j^k/d_j^k : d_j^k < 0\}$ and compute

$$\lambda^k = \operatorname*{argmin}_{0 \leq \lambda \leq \lambda_{\max}} f(x^k + \lambda d^k).$$

Then, set $x^{k+1} = x^k + \lambda^k d^k$ and $k = k + 1$.
Return to Step 1.

The convex-simplex algorithm (CSA)

The convergence proof for CSA is the same as that of the reduced gradient algorithm and follows standard arguments in nonlinear programming. Although CSA behaves like the simplex algorithm, CSA converges slowly when compared to other algorithms. This is due in part to the restriction that only one nonbasic component of the improving feasible direction can be nonzero. To accelerate CSA, B.A. Murtagh and M.A.

Saunder [5] used a second order approximation for the objective function and allowed several nonbasic components to be nonzero. The latter is often referred to as *superbasic variables*.

For other developments, S. Nguyen [6] and R.V. Helgason and J.L. Kennington [4] specialized CSA to nonlinear network flow problems. D.P. Rutenberg [7] (see also [3]) demonstrated that special techniques for solving linear programs with generalized network and generalized upper bounding structure also extend to CSA.

See also

- ▶ Lemke Method
- ▶ Linear Complementarity Problem
- ▶ Linear Programming
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Sequential Simplex Method

References

1. Armijo L (1966) Minimization of functions having Lipschitz continuous first-partial derivatives. Pacific J Math 16:1–3
2. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: Theory and algorithm. Wiley, New York
3. Hsia WS (1974) On Rutenberg's decomposition method. Managem Sci 21:10–12
4. Kennington JL, Helgason RV (1980) Algorithms for network programming. Wiley, New York
5. Murtagh BA, Saunder MA (1978) Large-scale linearly constrained optimization. Math Program 14:14–72
6. Nguyen S (1974) An algorithm for the traffic assignment problem. Transport Sci 8:203–216
7. Rutenberg DP (1970) Generalized networks, generalized upper bounding, and decomposition of the convex simplex method. Managem Sci 16:388–401
8. Wolfe P (1963) Methods of nonlinear programming. In: Graves RL, Wolfe P (eds) Recent Advances in Mathematical Programming. McGraw-Hill, New York
9. Zangwill WI (1967) The convex simplex method. Managem Sci 14:221–283

Copositive Optimization

Immanuel M. Bomze
University of Vienna, Vienna, Austria

MSC2000: 90C20, 90C22, 90C26

Article Outline

References

Copositive optimization (or copositive programming, coined in [4]) is a special case of conic optimization that consists in extremizing a linear function over a (convex) cone subject to additional (inhomogeneous) linear (inequality or equality) constraints.

It is well known that the simplest class of hard problems in continuous optimization is that of quadratic optimization problems [20] – to extremize a (possibly indefinite) quadratic form $x^\top Qx$ over a polyhedron $\{x \in \mathbb{R}^n_+ : Ax = b\}$. Note that a linear term in the objective function can be removed by an affine transformation of the polyhedron. The number of local, nonglobal solutions to this problem may be exponential in the number of variables and/or constraints.

This class has a close connection to copositive optimization. The idea here is to linearize the quadratic form

$$x^\top Qx = \operatorname{trace}(x^\top Qx) = \operatorname{trace}(Q, xx^\top) = \langle Q, xx^\top \rangle$$

by introducing the new symmetric matrix variable $X = xx^\top$ and Frobenius duality $\langle X, Y \rangle = \operatorname{trace}(X, Y)$. If $Ax \in \mathbb{R}^m_+$ for all $x \in \mathbb{R}^n_+$ and $b \in \mathbb{R}^m_+$, then the linear constraints can be squared, to arrive in a similar way at constraints of the form $\langle A_i, X \rangle = b_i^2$.

Now the set of all these X generated by feasible x is nonconvex since $\operatorname{rank}(xx^\top) = 1$. The convex hull

$$\mathcal{K} = \operatorname{conv}\{xx^\top : x \in \mathbb{R}^n_+\} ,$$

results in a convex matrix cone known as the cone of completely positive matrices since [14]; see [1]. Note that a similar construction dropping nonnegativity constraints leads to

$$\mathcal{P} = \operatorname{conv}\{xx^\top : x \in \mathbb{R}^n\} ,$$

the cone of positive-semidefinite matrices, the basic set in semidefinite optimization (or semidefinite programming, SDP).

The first account of copositive optimization goes back to [4], who established a copositive representation of a subclass of particular interest, namely, in standard quadratic optimization (StQP). Here the feasible polyhedron is the standard simplex $\Delta = \{x \in \mathbb{R}^n_+ :$

$\sum_i x_i = 1\}$: this subclass is also NP-hard from the worst-case complexity but allows for a polynomial-time approximation scheme [3]. There can be up to $\approx 2^n/(1.25\sqrt{n})$ local nonglobal solutions. Now, with J the $n \times n$ all-ones matrix, we have

$$\min\left\{x^\top Qx \colon x \in \Delta\right\} = \min\left\{\langle Q, X\rangle \colon \langle J, X\rangle = 1\,,\ X \in \mathcal{K}\right\}\,.$$

Note that the right-hand problem is convex, so there are no more local, nonglobal solutions. In addition, the objective function is now linear, and there is just one linear equality constraint. The complexity has been completely pushed into the feasibility condition $X \in \mathcal{K}$, which also shows that there are indeed convex minimization problems that cannot be solved easily.

The duality theory for conic optimization problems requires the dual cone $\mathcal{K}^*$ of $\mathcal{K}$ w.r.t. the Frobenius inner product $\langle \ldots, \rangle$, which is

$$\mathcal{K}^* = \{S \text{ symmetric } n \times n \colon \langle S, X\rangle \geq 0 \text{ for all } X \in \mathcal{K}\}\,.$$

Here it can easily be shown that $\mathcal{K}^*$ coincides with the cone of copositive matrices, which justifies the terminology:

$$\mathcal{K}^* = \left\{S \text{ symmetric } n \times n \colon x^\top Sx \geq 0 \text{ if } x \in \mathbb{R}^n_+\right\}\,,$$

i. e., a matrix S is copositive [18] (most probably abbreviating "*co*nditionally *positive*-semidefinite"), if S generates a quadratic form $x^\top Sx$ taking no negative values over the positive orthant. The dual of the special program over $\mathcal{K}$ above is then

$$\max\{y \colon S = Q - yJ \in \mathcal{K}^*\}\,,$$

a linear objective in just one variable y with the innocent-looking feasibility constraint $S \in \mathcal{K}^*$. This shows that checking membership of $\mathcal{K}^*$ (and, similarly, of $\mathcal{K}$) is already NP-hard, and there are many approaches to algorithmic copositivity detection; for recent developments see, e. g., [7] and references therein.

More generally, a typical primal-dual pair in copositive optimization (COP) is of the following form:

$$\inf\{\langle C, X\rangle \colon \langle A_i, X\rangle = b_i\,,\quad i = 1\colon m\,,\ X \in \mathcal{K}\} \geq \sup\left\{\sum_i b_i y_i \colon y \in \mathbb{R}^m\,,\quad S = C - y_i A_i \in \mathcal{K}^*\right\}\,.$$

The inequality above is just standard weak duality, but observe we have to use inf and sup since – as in general conic optimization – there may be problems with the attainability of either or both problems above, and likewise there could be a (finite or infinite) positive duality gap without any further conditions such as strict feasibility (Slater's condition). For the above representation of standard quadratic optimization problems, this is not the case:

$$\min\{\langle Q, X\rangle \colon \langle J, X\rangle = 1\,,\quad X \in \mathcal{K}\} = \max\{y \colon S = Q - yJ \in \mathcal{K}^*\}\,.$$

But for a similar class arising in many applications, the multi-standard quadratic optimization problems [6], dual attainability is not guaranteed while the duality gap is zero – an intermediate form between weak and strong duality [25].

Recently Burer [8] showed a more general result: any mixed-binary quadratic optimization problem

$$\min\left\{\frac{1}{2}x^\top Qx + c^\top x \colon Ax = b\,,\ x \in \mathbb{R}^n_+\,,\ x_j \in \{0,\,1\}\,,\ \text{all } j \in B\right\}$$

can (under mild conditions) be represented as COP:

$$\min\left\{\frac{1}{2}\langle \hat{Q}, \hat{X}\rangle \colon \mathcal{A}(\hat{X}) = \hat{b},\ X \in \mathcal{K}\right\}\,,$$

where $\hat{X}$ and $\hat{Q}$ are $(n+1) \times (n+1)$ matrices, and the size of $(\mathcal{A}, \hat{b})$ is polynomial in the size of (A, b).

Denote by $\mathcal{N} = \{N \text{ symmetric } n \times n \colon N_{ij} \geq 0 \text{ for all } i,\ j = 1\colon n\}$ the cone of nonnegative matrices. Then evidently

$$\mathcal{K} \subseteq \mathcal{P} \cap \mathcal{N} \subset \mathcal{P} + \mathcal{N} \subseteq \mathcal{K}^*\,,$$

which also shows that $\mathcal{K}$ never can be self-dual (note $(\mathcal{P} \cap \mathcal{N})^* = \mathcal{P}^* + \mathcal{N}^* = \mathcal{P} + \mathcal{N}$), unlike $\mathcal{P} = \mathcal{P}^*$ and $\mathcal{N} = \mathcal{N}^*$. For $n \geq 5$, A. Horn noted that the leftmost and the rightmost inclusion above is strict [10,14], so the middle sets $\mathcal{P} \cap \mathcal{N}$ and $\mathcal{P} + \mathcal{N}$ can only be used as tractable approximations for the intractable cones $\mathcal{K}$ and $\mathcal{K}^*$, respectively.

Copositive approximation hierarchies [3,15,21,22] start with $\mathcal{K}^{(0)} = \mathcal{P} + \mathcal{N}$ and consist of an increasing sequence $\mathcal{K}^{(r)}$ of cones satisfying $\cup_{r\geq 0}\mathcal{K}^{(r)} = \operatorname{int}\mathcal{K}^*$, the cone of strictly copositive matrices, i. e., those that

generate quadratic forms strictly positive over Δ. For instance, a higher-order approximation due to [21] squares the variables to get rid of sign constraints: $S \in \mathcal{K}^*$ if and only if $y^\top S y \geq 0$ for all y s.t. $y_i = x_i^2$, some $x \in \mathbb{R}^n$, and this is guaranteed if the n-variable polynomial of degree $2(r+2)$ in x,

$$p_S^{(r)}(x) = \big(\sum x_i^2\big)^r y^\top S y = \big(\sum x_i^2\big)^r \sum_{j,\,k} S_{jk} x_j^2 x_k^2 \,,$$

is nonnegative for all $x \in \mathbb{R}^n$. But this holds in particular if

(a) $p_S^{(r)}$ has no negative coefficients; or if

(b) $p_S^{(r)}$ is a sum-of-squares (s.o.s.):

$$p_S^{(r)}(x) = \sum_i [f_i(x)]^2 \,, \quad f_i \text{ some polynomials.}$$

This gives the approximation cones

$$C^{(r)} = \{S \text{ symmetric } n \times n \colon S \text{ satisfies (a)}\}$$

and

$$\mathcal{K}^{(r)} = \{S \text{ symmetric } n \times n \colon S \text{ satisfies (b)}\} \,.$$

While $C^{(r)}$ can be described by linear constraints on the entries of S, leading to LP formulations, the cones $\mathcal{K}^{(r)}$ need for their description linear matrix inequalities (LMIs), leading to SDP formulations. However, for large r both are also intractable as they generate problems on matrices of order $\mathcal{O}(n^{r+1} \times n^{r+1})$, see [3].

Copositive optimization has been receiving increasing attention also because many NP-hard combinatorial problems have a representation in this domain; we start with the historically first such representation, the maximum (weight) clique problem, which amounts to finding a largest (or heaviest) clique in an undirected graph G (with weights on the vertices). Using an StQP formulation going back to [19] and applying some regularization [2], the following copositive formulation was introduced in [4]:

$$\begin{aligned} 1/\omega(G) &= \min \big\{x^\top Q_G x \colon x \in \Delta\big\} \\ &= \min \big\{\langle Q_G, X\rangle \colon \langle J, X\rangle = 1 \,,\ X \in \mathcal{K}\big\} \,, \end{aligned}$$

where Q_G is a matrix derived from the adjacency matrix of G (and the weights). Taking the inverse $t = 1/y$ in the dual of the last problem above, we also arrive at the formulation of [9] (for the complementary graph):

$$\omega(G) = \min \{t \colon tQ_G - J \in \mathcal{K}^*\} \,.$$

Here $\omega(G)$ is the clique number of G, i. e., the size (weight) of a maximum (weight) clique in G. Replacing $\mathcal{K}^*$ with its zero-order approximation, we get a strengthening $\theta'(G)$ of the well-known Lovász bound $\theta(G)$ [16,17,26]:

$$\theta'(G) = \min \{t \colon tQ_G - J \in \mathcal{P} + \mathcal{N}\} \geq \omega(G) \,,$$

while shrinking further the feasible set to $\mathcal{P}$, we finally arrive at the Lovász number $\theta(G)$ which – as $\theta'(G)$ – can be computed in polynomial time:

$$\theta(G) = \min \{t \colon tQ_G - J \in \mathcal{P}\} \geq \omega(G) \,.$$

Strong duality yields, as above,

$$1/\theta'(G) = \min \{\langle Q_G, X\rangle \colon \langle J, X\rangle = 1,\ X \in \mathcal{P} \cap \mathcal{N}\} \,,$$

and a recent improvement over $\theta'(G)$ adding a single valid linear cut motivated by the COP representation is

$$\begin{aligned} 1/\theta^C(G) = \min \big\{&\langle Q_G, X\rangle \colon \langle J, X\rangle = 1 \,, \\ &\langle C, X\rangle \geq 0 \,,\ X \in \mathcal{P} \cap \mathcal{N}\big\} \geq 1/\theta'(G) \,, \end{aligned}$$

where $C \in \mathcal{K}^*$ is arbitrary: indeed, for any $X \in \mathcal{K}$ we then have $\langle C, X\rangle \geq 0$. See [5] for appropriate choices of C and results, and [9,13,15,22] for higher-order approximation alternatives, with a particular emphasis on SDP-based bounds on the clique number. Similar copositivity optimization approaches, among many others, were employed to obtain bounds on the (fractional) chromatic number of a graph [11,12], and graph partitioning and quadratic assignment problems [23,24].

References

1. Berman A, Shaked-Monderer N (2003) Completely positive matrices. World Scientific, Singapore
2. Bomze IM (1998) On standard quadratic optimization problems. J Glob Optim 13:369–387
3. Bomze IM, de Klerk E (2002) Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. J Glob Optim 24:163–185
4. Bomze IM, Dür M, de Klerk E, Quist A, Roos C, Terlaky T (2000) On copositive programming and standard quadratic optimization problems. J Glob Optim 18:301–320
5. Bomze IM, Frommlet F, Locatelli M (2007) The first cut is the cheapest: improving SDP bounds for the clique number via copositivity. Technical Report TR-ISDS 2007-05,

University of Vienna, submitted. Available at http://www.optimization-online.org/DB_HTML/2006/08/1443.html. Accessed Nov 2007
6. Bomze IM, Schachinger W (2007) Multi-standard quadratic optimization problems. Technical Report TR-ISDS 2007-12, University of Vienna, submitted. Available at http://www.optimization-online.org/DB_HTML/2007/11/1843.html. Accessed Mar 2008
7. Bundfuss S, Dür M (2008) Algorithmic copositivity detection by simplicial partition. Linear Algebr Appl 428:1511–1523
8. Burer S (2006) On the copositive representation of binary and continuous nonconvex quadratic programs. Manuscript, Department of Management Sciences, University of Iowa, October (2006). Available at http://www.optimization-online.org/DB_HTML/2006/10/1501.html. Accessed Nov 2007
9. de Klerk E, Pasechnik DV (2002) Approximation of the stability number of a graph via copositive programming. SIAM J Optim 12:875–892
10. Diananda PH (1967) On non-negative forms in real variables some or all of which are non-negative. Proc Cambridge Philos Soc 58:17–25
11. Dukanović I, Rendl F (2007) Semidefinite programming relaxations for graph coloring and maximal clique problems. Math Programm 109:345–365
12. Gvozdenović N, Laurent M (2008) The operator Ψ for the chromatic number of a graph. To appear in SIAM J Optim. Available at http://www.optimization-online.org/DB_HTML/2007/02/1592.html. Accessed Mar 2008
13. Gvozdenović N, Laurent M (2006) Semidefinite bounds for the stability number of a graph via sums of squares of polynomials. Lecture Notes in Computer Science, vol 3509/2005, pp 136–151. Integer Programming and Combinatorial Optimization: 11th International IPCO Conference
14. Hall M, Newman M (1963) Copositive and completely positive quadratic forms. Proc Cambridge Philos Soc 59:329–339
15. Lasserre JB (2001) Global optimization with polynomials and the problem of moments. SIAM J Optim 11:796–817
16. Lovász L (1979) On the Shannon capacity of a graph. IEEE Trans Inf Theory IT 25:1–7
17. McEliece RJ, Rodemich ER, Rumsey HC (1978) The Lovász' bound and some generalizations. J Combinat Inf Syst Sci 3:134–152
18. Motzkin TS (1952) Copositive quadratic forms. Natl Bur Stand Rep 1818 11–22
19. Motzkin TS, Straus EG (1965) Maxima for graphs and a new proof of a theorem of Turán. Can J Math 17:533–540
20. Murty KG, Kabadi SN (1987) Some NP-complete problems in quadratic and linear programming. Math Programm 39:117–129
21. Parrilo P (2003) Semidefinite programming relaxations for semi-algebraic problems. Math Programm 696B:293–320
22. Peña J, Vera J, Zuluaga L (2007) Computing the stability number of a graph via linear and semidefinite programming. SIAM J Optim 18:87–105
23. Povh J, Rendl F (2006) Copositive and semidefinite relaxations of the quadratic assignment problem. Univ. of Klagenfurt, submitted. Available at http://www.optimization-online.org/DB_HTML/2006/10/1502.html. Accessed Nov 2007
24. Povh J, Rendl F (2007) A copositive programming approach to graph partitioning. SIAM J Optim 18:223–241
25. Schachinger W, Bomze IM (2007) A conic duality Frank–Wolfe type theorem via exact penalization in quadratic optimization. Technical Report 2006-10, ISDS, University of Vienna. to appear in Math Oper Res. Available at http://www.optimization-online.org/DB_HTML/2007/02/1596.html. Accessed Nov 2007
26. Schrijver A (1979) A comparison of the Delsarte and Lovasz bounds. IEEE Trans Inf Theory IT 25:425–429

Copositive Programming

Stanislav Busygin
Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA

MSC2000: 90C25, 90C22

Article Outline

Introduction
Applications
  Complexity of Copositive Programming
Models
  Approximating C_n with Linear Matrix Inequalities
References

Introduction

Let us denote the set of $n \times n$ real symmetric matrices by

$$S_n = \{X \in \mathbb{R}^{n\times n},\ X = X^T\}\,.$$

We will be considering the following subsets of S_n:

- The $n \times n$ symmetric *positive semidefinite* matrices

$$S_n^+ = \{X \in S_n,\ y^T X y \geq 0\ \forall y \in \mathbb{R}^n\}\,;$$

- The $n \times n$ symmetric *copositive* matrices

$$C_n = \{X \in S_n,\ y^T X y \geq 0\ \forall y \in \mathbb{R}^n,\ y \geq 0\}\,;$$

- The $n \times n$ symmetric *completely positive* matrices

$$C_n^* = \{X = \sum_{i=1}^{k} y_i y_i^T,\ y_i \in \mathbb{R}^n,\ y_i \geq 0\,(i = 1, \ldots, k)\};$$

- The $n \times n$ symmetric nonnegative matrices

$$\mathcal{N}_n = \{X \in S_n, X \geq 0\}\,.$$

It is easy to see that all these sets are *convex cones* (that is, if X and Y belong to one of these sets, then cX, for any $c \geq 0$, and $\alpha X + (1-\alpha)Y$, for any $0 \leq \alpha \leq 1$, also do so).

We will denote the Eucledian inner product of $A \in S_n$ and $B \in S_n$ by

$$A \bullet B = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ij} = \text{trace}(AB)\,.$$

For an arbitrary cone of matrices $\mathcal{K}$, we define its *dual cone* $\mathcal{K}^*$ as

$$\mathcal{K}^* = \{Y | X \bullet Y \geq 0,\ \forall X \in \mathcal{K}\}\,.$$

The cone of positive semidefinite matrices S_n^+ is dual to itself, and it is easy to see that the cone of copositive matrices C_n and the cone of completely positive matrices C_n^* are dual to each other (generally, $\mathcal{K}^{**} = \mathcal{K}$).

For a given cone of $n \times n$ matrices $\mathcal{K}_n$, and its dual cone $\mathcal{K}_n^*$, we define a pair of conic linear programs called, correspondingly, *primal* and *dual*:

$$p^* = \inf_X\{C \bullet X :\ A_i \bullet X = b_i\ (i = 1, \ldots, m),\ X \in \mathcal{K}_n\}\,; \quad (1)$$

$$d^* = \sup_y\{b^T y :\ C - \sum_{i=1}^{m} y_i A_i + S = C,\ S \in \mathcal{K}_n^*\}. \quad (2)$$

When $\mathcal{K}_n = \mathcal{N}_n$, we refer to *linear programming*, when $\mathcal{K}_n = S_n^+$, we refer to *semidefinite programming*, and when $\mathcal{K}_n = C_n$, we refer to *copositive programming*.

The *conic duality theorem* (see, e. g., [9]) establishes the duality relation between (1) and (2):

Theorem 1 (Conic Duality Theorem) *If there exists an interior feasible solution $X^0 \in int(\mathcal{K})$ of (1) and a feasible solution of (2), then $p^* = d^*$ and the supremum in (2) is attained. Similarly, if there exist feasible y^0 and S^0 for (2), where $S^0 \in int(\mathcal{K}^*)$, and a feasible solution for (1), then $p^* = d^*$ and the infimum in (1) is attained.*

It is well-known that optimization over the cones S_n^+ and $\mathcal{N}_n$ can be done in polynomial time (in sense of computing ε-optimal solution), but copositive programming is NP-hard as we will see below.

Applications

Copositive matrices have a great variety of applications in mathematics and, especially, in optimization. They play an essential part in characterization of local solutions of constrained optimization problems [5], including the linear complementarity problem. In [1,2,8] the authors used copositivity to improve convex relaxation bounds for quadratic programming problems. Generally, convex relaxations are the underlying basis of many crucial results in robustness analysis. For example, copositive matrices have been used in the stability analysis of piecewise linear control systems (in context of using piecewise quadratic Lyapunov functions [4]).

Complexity of Copositive Programming

Copositive programming can be easily shown to be NP-hard by reduction from the maximum independent set problem. In [3], the authors established the following theorem:

Theorem 2 *Let $G(V, E)$ be a given graph with $V = \{1 \ldots n\}$. Then the maximum independent set size of G is the optimum value of the following program:*

$$\alpha(G) = \max\ O_n \bullet X \quad (3)$$

subject to

$$\begin{cases} x_{ij} = 0,\ (i,j) \in E \\ trace(X) = 1 \\ X \in C_n^* \end{cases} \quad (4)$$

where O_n is the all-one $n \times n$ matrix.

Proof. Extreme rays of C_n^* are rank-one matrices of the form xx^T for nonnegative $x \in \mathbb{R}^n$. Then, considering the convex cone

$$C_G = \{X \in C_n^* : x_{ij} = 0,\ (i,j) \in E\}\,,$$

we can conclude that its extreme rays are of the form xx^T, where the nonnegative $x \in \mathbb{R}^n$ supports an independent set (i. e., the set $\{i : x_i > 0\}$ is independent). Therefore, the extreme points of the set defined by (4) are given by the intersection of the extreme rays of C_G with the hyperplane $O_n \bullet X = 1$.

Since the optimum value of a linear function over a convex set is attained at an extreme point, there is an optimum solution of the form

$$X^* = x^* x^{*T},\ x^* \in \mathbb{R}^n,\ x^* \geq 0, \|x^*\| = 1,$$

and where x^* supports an independent set. Therefore, we can reformulate the program (3) as

$$\max_x \left(\sum_{i=1}^n x_i \right)^2,\ \|x\| = 1,$$
$$x \geq 0,\ x_i x_j > 0 \Rightarrow (i,j) \notin E.$$

Then, it is easy to see that the maximum is attained when x supports a maximum independent set and all $x_i > 0$ are equal to $1/\alpha(G)$. This provides the optimum value to the program (3) equal to $\alpha(G)$. QED.

Since $X \in C_n^*$ is always nonnegative, we can reduce the set of constraints $x_{ij} = 0, (i,j) \in E$ in (4) to a single constraint $A \bullet X = 0$. Thus, the following copositive program is dual to (3), (4):

$$\alpha(G) = \min_{\lambda, y \in \mathbb{R}} \{\lambda : \lambda I + yA - O_n = Q,\ Q \in C_n\}.$$

Therefore, the maximum independent set problem is reducible to copositive programming. See also [1,2] for reduction of the standard quadratic optimization problem to copositive programming.

Furthermore, it can be shown that checking if a given matrix is not copositve is *NP*-complete [5] and, hence, checking matrix copositivity is *co-NP*-complete.

Models

Approximating C_n with Linear Matrix Inequalities

While, in general, there is no polynomial-time verifiable certificate of copositivity, unless *co-NP* = *NP*, in many cases it is still possible to show by a short argument that a matrix is copositive. For instance, if the matrix M can be represented as sum of a positive semidefinite matrix $S \in S_n^+$ and a nonnegative matrix $N \in \mathcal{N}_n$, then it follows that $M \in C_n$. Hence, we can obtain a semidefinite relaxation of a copositive program over M introducing the linear matrix constraints:

$$\begin{cases} M = S + N \\ N \geq 0,\ S \in S_n^+ \end{cases}$$

Parrilo showed in [6] that using sufficiently large systems of linear matrix inequalities, one can approximate the copositive cone C_n to any desired accuracy.

Obviously, copositivity of the matrix M is equivalent to (global) nonnegativity of the fourth-degree form:

$$P(x) = (x \circ x)^T M (x \circ x) = \sum_{i=1}^n \sum_{j=1}^n M_{ij} x_i^2 x_j^2 \geq 0,\ x \in \mathbb{R}^n, \quad (5)$$

where "$\circ$" denotes the componentwise (Hadamard) product. It is shown in [6] that the mentioned decomposition into positive semidefinite and nonnegative matrices exists if and only if $P(x)$ can be represented as sum of squares. Higher-order sufficient conditions for copositivity proposed by Parrilo in [6] correspond to checking whether the polynomial

$$P^{(r)}(x) = \left(\sum_{i=1}^n x_i^2 \right) P(x) \quad (6)$$

has a sum-of-squares decomposition (or – a weaker condition – whether $P^{(r)}(x)$ has only nonnegative coefficients). These conditions can be expressed via linear matrix inequalities over $n^r \times n^r$ symmetric matrices. In particular, for $r = 1$, Parrilo showed that the existence of a sum-of-squares decomposition of $P^{(1)}(x)$ is equivalent to feasibility of the following system (see also [3]):

$$\begin{cases} M - M^{(i)} \in S_n^+, & i = 1, \ldots, n, \\ M_{ii}^{(i)} = 0, & i = 1, \ldots, n, \\ M_{jj}^{(i)} + 2M_{ij}^{(j)} = 0, & i \neq j, \\ M_{jk}^{(i)} + M_{ik}^{(j)} + M_{ij}^{(k)} \geq 0, & i < j < k, \end{cases}$$

where $M^{(i)}$ $(i = 1, \ldots, n)$ are symmetric matrices.

With sufficiently large r, the convergence to the copositivity constraint on M is guaranteed by the famous theorem of Pólya [7]:

Theorem 3 (Pólya) *Let f be a homogeneous polynomial which is positive on the simplex*

$$\Delta = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1,\ x \geq 0 \right\}.$$

Then, for a sufficiently large N, all the coefficients of the polynomial

$$\left(\sum_{i=1}^{n} x_i \right)^N f(x)$$

are positive.

References

1. Bomze IM, de Klerk E (2002) Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. J Glob Optim 24:163–185
2. Bomze IM, Dür M, de Klerk E, Roos C, Quist AJ, Terlaky T (2000) On copositive programming and standard quadratic optimization problems. J Glob Optim 18:301–320
3. de Klerk E, Pasechnik D (2001) Approximation of the stability number of a graph via copositive programming. SIAM J Optim 12(4):875–892
4. Johansson M (1999) Piecewise linear control systems. PhD thesis, Lund Institute of Technology, Lund
5. Murty KG, Kabadi SN (1987) Some *NP*-complete problems in quadratic and nonlinear programming. Math Prog 39:117–129
6. Parrilo PA (2000) Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. PhD thesis, California Institute of Technology, Pasadena, CA, http://www.mit.edu/~parrilo/pubs
7. Pólya G (1928) Über positive Darstellung von Polynomen, Vierteljschr. Naturforsch Ges Zürich 73:141–145 (Collected Papers, vol 2, MIT Press, Cambridge, MA, London, 1974, 309–313)
8. Quist AJ, de Klerk E, Roos C, Terlaky T (1998) Copositive relaxation for general quadratic programming. Optim Method Softw 9:185–208
9. Renegar J (2001) A Mathematical View of Interior-Point Methods in Convex Optimization. SIAM, Philadelphia

Cost Approximation Algorithms

CA Algorithms

MICHAEL PATRIKSSON
Department Math., Chalmers University Technol., Göteborg, Sweden

MSC2000: 90C30

Article Outline

Keywords
Instances of the CA Algorithm
  Linearization Methods
  Regularization, Splitting and Proximal Point Methods
  Perturbation Methods
Variational Inequality Problems
Descent Properties
  Optimization
  Variational Inequality Problems
Steplength Rules
Convergence Properties
Decomposition CA Algorithms
  Sequential Decomposition
  Synchronized Parallel Decomposition
  Asynchronous Parallel Decomposition
See also
References

Keywords

Linearization methods; Gradient projection; Quasi-Newton; Frank–Wolfe; Sequential quadratic programming; Operator splitting; Proximal point; Levenberg–Marquardt; Auxiliary problem principle; Subgradient optimization; Variational inequality problem; Merit function; Cartesian product; Gauss–Seidel; Jacobi; Asynchronous computation

The notion of cost approximation (CA) was created in the thesis [39], to describe the construction of the subproblem of a class of iterative methods in mathematical programming. In order to explain the notion of CA, we will consider the following conceptual problem (the full generality of the algorithm is explained in detail in [45] and in [37,38,40,42,43,44,46]):

$$\begin{cases} \min & T(x) := f(x) + u(x), \\ \text{s.t.} & x \in X, \end{cases} \tag{1}$$

where $X \subseteq \mathbf{R}^n$ is nonempty, closed and convex, u: $\mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ is lower semicontinuous (l.s.c.), proper and convex, and f: $\mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ is continuously differentiable (for short, in C^1) on dom $u \cap X$, where dom denotes 'effective domain'. This problem is general enough to cover convex optimization ($f = 0$), unconstrained optimization ($f = 0$ and $X = \mathbf{R}^n$), and differentiable constrained optimization ($u = 0$). We note

that if int (dom u) $\cap X$ is nonempty, then any locally optimal solution x^* satisfies the inclusion

$$-\nabla f(x^*) \in \partial u(x^*) + N_X(x^*), \tag{2}$$

where N_X is the normal cone operator for X and ∂u is the subdifferential mapping for u. Equivalently, by the definitions of these two operators,

$$\nabla f(x^*)^\top (x - x^*) + u(x) - u(x^*) \geq 0, \quad x \in X.$$

The CA algorithm was devised in order to place a number of existing algorithms for (1) in a common framework, thereby facilitating comparisons, for example, between their convergence properties. In short, the method works iteratively as follows. Note that, from (2), we seek a zero of the mapping $[\nabla f + \partial u + N_X]$. Given an iterate, $x^t \in \text{dom } u \cap X$, this mapping is approximated by a monotone mapping, constructed so that a zero of which is easier to find. Such a point, y^t, is then utilized in the search for a new iterate, x^{t+1}, having the property that the value of some merit function for (1) is reduced sufficiently, for example through a line search in T along the direction of $d^t := y^t - x^t$.

Instances of the CA Algorithm

To obtain a monotone approximating mapping, we introduce a monotone mapping Φ^t: dom $u \cap X \to \mathbf{R}^n$, which replaces the (possibly nonmonotone) mapping ∇f; by subtracting off the error at x^t, $[\Phi^t - \nabla f](x^t)$, from Φ^t, so that the resulting mapping becomes $[\Phi^t + \partial u + N_X] + [\nabla f - \Phi^t](x^t)$, the CA subproblem becomes the inclusion

$$[\Phi^t + \partial u + N_X](y^t) + [\nabla f - \Phi^t](x^t) \ni 0^n. \tag{3}$$

We immediately reach an interesting fixed-point characterization of the solutions to (2):

Theorem 1 *(Fixed-point, [45]) The point x^t solves (2) if and only if $y^t = x^t$ solves (3).*

This result is a natural starting point for devising stopping criteria for an algorithm.

Assume now that $\Phi^t \equiv \nabla \varphi^t$ for a convex function φ^t. We may then derive the inclusion equivalently as follows. At x^t, we replace f with the function φ^t, and subtract off the linearization of the error at x^t; the subproblem objective function then becomes

$$T_{\varphi^t}(y) := \varphi^t(y) + u(y) + [\nabla f(x^t) - \nabla \varphi^t(x^t)]^\top y.$$

It is straightforward to establish that (3) is the optimality conditions for the convex problem of minimizing T_{φ^t} over X.

Linearization Methods

Our first example instances utilize Taylor expansions of f to construct the approximations.

Let $u = 0$ and $X = \mathbf{R}^n$. Let $\Phi^t(y) := (1/\gamma_t)\, Q^t\, y$, where $\gamma_t > 0$ and Q^t is a symmetric and positive definite mapping in $\mathbf{R}^{n \times n}$. The inclusion (3) reduces to

$$\nabla f(x^t) + \frac{1}{\gamma_t} Q^t (y^t - x^t) = 0^n,$$

that is, $y^t = x^t - \gamma_t (Q^t)^{-1} \nabla f(x^t)$. The direction of $y^t - x^t$, $d^t := -\gamma_t (Q^t)^{-1} \nabla f(x^t)$, is the search direction of the class of *deflected gradient methods*, which includes the *steepest descent method* ($Q^t := I^n$, the identity matrix) and *quasi-Newton methods* (Q^t equals (an approximation of) $\nabla^2 f(x^t)$, if positive definite). (See further [5,35,47,50].)

In the presence of constraints, this choice of Φ^t leads to $y^t = P_X^{Q^t}[x^t - \gamma_t (Q^t)^{-1} \nabla f(x^t)]$, where $P_X^{Q^t}[\cdot]$ denotes the projection onto X with respect to the norm $\|z\|_{Q^t} := \sqrt{z^\top Q^t z}$. Among the algorithms in this class we find the *gradient projection algorithm* ($Q^t := I^n$) and *Newton's method* ($Q^t := \nabla^2 f(x^t)$, $\gamma_t := 1$). (See [5,19,27,50].)

A first order Taylor expansion of f is obtained from choosing $\varphi^t(y) := 0$; this results in $T_{\varphi^t}(y) = \nabla f(x^t)^\intercal\, y$ (if $u = 0$ is still assumed), which is the subproblem objective in the Frank–Wolfe algorithm ([5,17]; cf. also ► Frank–Wolfe algorithm).

We next provide the first example of the very useful fact that the result of the cost approximation (in the above examples a linearization), leads to different approximations of the original problem, and ultimately to different algorithms, depending on which *representation* of the problem to one applies the cost approximation.

Consider the problem

$$\begin{cases} \min & f(x) \\ \text{s.t.} & g_i(x) = 0, \quad i = 1, \ldots, \ell, \end{cases} \tag{4}$$

where f and g_i, $i = 1, \ldots, \ell$, are functions in C^2. We may associate this problem with its first order optimal-

ity conditions, which in this special case is

$$F(x^*, \lambda^*) := \begin{pmatrix} \nabla_x L(x^*, \lambda^*) \\ -\nabla_\lambda L(x^*, \lambda^*) \end{pmatrix} = \begin{pmatrix} 0^n \\ 0^\ell \end{pmatrix}, \qquad (5)$$

where $\lambda \in \mathbf{R}^\ell$ is the vector of Lagrange multipliers for the constraints in (4), and $L(x, \lambda) := f(x) + \lambda^\intercal g(x)$ is the associated Lagrangian function. We consider using Newton's method for this system, and therefore introduce a (primal-dual) mapping $\Phi: \mathbf{R}^{2(n+\ell)} \to \mathbf{R}^{n+\ell}$ of the form

$$\Phi((y, p), (x, \lambda)) := \nabla F(x, \lambda) \begin{pmatrix} y \\ p \end{pmatrix} = \begin{pmatrix} \nabla_x^2 L(x, \lambda) & \nabla g(x)^\top \\ -\nabla g(x) & 0 \end{pmatrix} \begin{pmatrix} y \\ p \end{pmatrix}.$$

The resulting CA subproblem in (y, p) can be written as the following linear system:

$$\nabla_x^2 L(x, \lambda)(y - x) + \nabla g(x)^\top p = 0^n,$$
$$\nabla g(x)(y - x) = -g(x);$$

this system constitutes the first order optimality conditions for (e. g., [4, Sec. 10.4])

$$\begin{cases} \min & f(x) + \nabla f(x)^\top (y - x) \\ & \quad + \frac{1}{2}(y - x)^\top \nabla_x^2 L(x, \lambda)(y - x) \\ \text{s.t.} & g(x) + \nabla g(x)^\top (y - x) = 0^\ell, \end{cases}$$

where we have added some fixed terms in the objective function for clarity. This is the generic subproblem of *sequential quadratic programming* (SQP) methods for the solution of (4); see, for example, [5,16].

Regularization, Splitting and Proximal Point Methods

We assume now that $f := f_1 + f_2$, where f_1 is convex on dom $u \cap X$, and rewrite the cost mapping as

$$[\nabla f + \partial u + N_X] = [\nabla \varphi^t + \nabla f_1 + \partial u + N_X] - [\nabla \varphi^t - \nabla f_2].$$

The CA subproblem is, as usual, derived by fixing the second term at x^t; the difference to the original setup is that we have here performed an *operator splitting* in the mapping ∇f to keep an additive part from being approximated. (Note that such a splitting can always be found by first choosing f_1 as a convex function, and then define $f_2 := f - f_1$. Note also that we can derive this subproblem from the original derivation by simply redefining $\varphi^t := \varphi^t + f_1$.) We shall proceed to derive a few algorithms from the literature.

Consider choosing $\varphi^t(y) = 1/(2\gamma_t) \parallel y - x^t \parallel^2$, $\gamma_t > 0$. If $f_2 = 0$, then we obtain the subproblem objective $T_{\varphi^t}(y) = T(y) + 1/(2\gamma_t) \parallel y - x^t \parallel^2$, which is the subproblem in the *proximal point algorithm* (e. g., [32,33,34,51,52]). This is the most classical algorithm among the regularization methods. More general choices of strictly convex functions φ^t are of course possible, leading for example to the class of regularization methods based on Bregman functions ([9,14,22]) and ψ-divergence functions ([23,54]). If, on the other hand, $f_1 = 0$, then we obtain the gradient projection algorithm if also $u = 0$.

We can also construct algorithms in between these two extremes, yielding a true operator splitting. If both f_1 and f_2 are nonzero, choosing $\varphi^t = 0$ defines a *partial linearization* ([25]) of the original objective, wherein only f_2 is linearized. Letting $x = (x_1^\top, x_2^\top)^\intercal$, the choice $\varphi^t(y) = 1/(2\gamma_t) \parallel y_1 - x_1^t \parallel^2$ leads to the *partial proximal point algorithm* ([7,20]); choosing $\varphi^t(y) = f(y_1, x_2^t)$ leads to a linearization of f in the variables x_2.

Several well-known methods can be derived either directly as CA algorithms, or as *inexact proximal point algorithms*. For example, the *Levenberg–Marquardt algorithm* ([5,49]), which is a Newton-like algorithm wherein a scaled diagonal matrix is added to the Hessian matrix in order to make the resulting matrix positive definite, is the result of solving the proximal point subproblem with one iteration of a Newton algorithm. Further, the *extra-gradient algorithm* of [24] is the result of instead applying one iteration of the gradient projection algorithm to the proximal point subproblem.

The perhaps most well-known splitting algorithm is otherwise the class of *matrix splitting methods in quadratic programming* (e. g., [28,29,35,36]). In a quadratic programming problem, we have

$$f(x) = \frac{1}{2} x^\top A x + q^\top x,$$

where $A \in \mathbf{R}^{n \times n}$. A splitting (A_1^t, A_2^t) of this matrix is one for which $A = A_1^t + A_2^t$, and it is further termed regular if $A_1^t - A_2^t$ is positive definite. Matrix splitting

methods correspond to choosing

$$f_1(x) = \frac{1}{2}x^\top A_1^t x,$$

and results in the CA subproblem mapping $y \longmapsto A_1^t\, y + [A_2^t\, x^t + q]$, which obviously is monotone whenever A_1^t was chosen positive semidefinite.

Due to the fact that proximal point and splitting methods have dual interpretations as augmented Lagrangian algorithms ([51]), a large class of multiplier methods is included among the CA algorithms. See [45, Chapt. 3.2–3.3] for more details.

Perturbation Methods

All the above algorithms assume that

i) the mappings ∂u and N_X are left intact; and
ii) the CA subproblem has the fixed-point property of Theorem 1.

We here relax these assumptions, and are then able to derive subgradient algorithms as well as perturbed CA algorithms which include both regularization algorithms and exterior penalty algorithms.

Let $[\Phi^t + N_X] + [\nabla f + \partial u + \Phi^t]$ represent the original mapping, having moved ∂u to the second term. Then by letting any element $\xi_u(x^t) \in \partial u(x^t)$ represent this point-to-set mapping at x^t, we reach the subproblem mapping of the *auxiliary problem principle* of [12]. Further letting $\Phi^t(y) = (1/\gamma_t)[y - x^t]$ yields the subproblem in the classical *subgradient optimization* scheme ([48,53]), where, assuming further that $f = 0$, $y^t := P_X[x^t - \gamma_t\, \xi_u(x^t)]$. (Typically, $\ell_t := 1$ is taken.)

Let again $[\Phi^t + \partial u + N_X] + [\nabla f + \Phi^t]$ represent the original problem mapping, but further let u be replaced by an *epiconvergent sequence* $\{u^t\}$ of l.s.c., proper and convex functions. An example of an epiconvergent sequence of convex functions is provided by convex exterior penalty functions. In this way, we can construct CA algorithm that approximate the objective function and simultaneously replace some of the constraints of the problem with exterior penalties. See [3,13] for example methods of this type.

One important class of regularization methods takes as the subproblem mapping $[\Phi^t + \nabla f + \partial u + N_X]$, where Φ^t is usually taken to be strongly monotone (cf. (12)). This subproblem mapping evidently does not have the fixed-point property, as it is not identical to the original one at x^t unless $\Phi^t(x^t) = 0^n$ holds. In order to ensure convergence, we must therefore force the sequence $\{\Phi^t\}$ of mappings to tend to zero; this is typically done by constructing the sequence as $\Phi^t := (1/\gamma_t)\Phi$ for a fixed mapping Φ and for a sequence of $\gamma_t > 0$ constructed so that $\{\gamma_t\} \to \infty$ holds. For this class of algorithms, F. Browder [10] has established convergence to a unique limit point x^* which satisfies $-\Phi(x^*) \in N_{X^*}(x^*)$, where X^* is the solution set of (2). The origin of this class of methods is the work of A.N. Tikhonov [55] for *ill-posed problems*, that is, problems with multiple solutions. The classical regularization mapping is the scaled identity mapping, $\Phi^t(y) := (1/\gamma_t)[y]$, which leads to least squares (least norm) solutions. See further [49,56].

Variational Inequality Problems

Consider the following extension of (2):

$$-F(x^*) \in \partial u(x^*) + N_X(x^*), \tag{6}$$

where $F\colon X \to \mathbf{R}^n$ is a continuous mapping on X. When $F = \nabla f$ we have the situation in (2), and also in the case when $F(x, y) = (\nabla_x \Pi(x, y)^\top, -\nabla_y \Pi(x, y)^\top)^\top$ holds for some saddle function Π on some convex product set $X \times Y$ (cf. (5)), the *variational inequality problem* (6) has a direct interpretation as the necessary optimality conditions for an optimization problem. In other cases, however, a merit function (or, objective function), for the problem (6) is not immediately available. We will derive a class of suitable merit functions below.

Given the convex function $\varphi\colon \operatorname{dom} u \cap X \to \mathbf{R}$ in C^1 on $\operatorname{dom} u \cap X$, we introduce the function

$$\psi(x) := \sup_{y \in X} L(y, x), \quad x \in \operatorname{dom} u \cap X, \tag{7}$$

where

$$L(y, x) := u(x) - u(y) + \varphi(x) - \varphi(y) + [F(x) - \nabla\varphi(x)]^\top (x - y). \tag{8}$$

We introduce the optimization problem

$$\min_{x \in X} \psi(x). \tag{9}$$

Theorem 2 *(Gap function, [45]) For any $x \in X$, $\psi(x) \geq 0$ holds. Further, $\psi(x) = 0$ if and only if x solves (6). Hence, the solution set of (6) (if nonempty) is identical*

to that of the optimization problem (9), and the optimal value is zero.

The Theorem shows that the CA subproblem defines an auxiliary function ψ which measures the violation of (6), and which can be used (directly or indirectly) as a merit function in an algorithm.

To immediately illustrate the possible use of this result, let us consider the extension of Newton's method to the solution of (6). Let $x \in \text{dom } u \cap X$, and consider the following cost approximating mapping: $y \longmapsto \Phi(y, x) := \nabla F(x)(y - x)$. The CA subproblem then is the linearized variational inequality problem of finding $y \in \text{dom } u \cap X$ such that

$$[F(x) + \nabla F(x)^\top (y - x)]^\top (z - y) + u(z) - u(y) \geq 0, \quad \forall z \in X. \tag{10}$$

Assuming that x is not a solution to (6), we are interested in utilizing the direction $d := y - x$ in a line search based on a merit function. We will utilize the *primal gap function* ([2,62]) for this purpose, which corresponds to the choice $\varphi := 0$ in the definition of ψ. We denote the primal gap function by ψ_p. Let w be an arbitrary solution to its inner problem, that is, $\psi_p(x) = u(x) - u(w) + F(x)^\top (x - w)$. The steplength is chosen such that the value of ψ_p decreases sufficiently; to show that this is possible, we use Danskin's theorem and the variational inequality (10) with $z = w$ to obtain (the maximum is taken over all w defining $\psi_p(x)$)

$$\begin{aligned} &\psi_p'(x; d) \\ &:= \max_w \left\{ \left[F(x) + \nabla F(x)^\top (x - w) \right]^\top d + u'(x; d) \right\} \\ &\leq -\psi_p(x) - d^\top \nabla F(x)^\top d, \end{aligned}$$

which shows that d defines a direction of descent with respect to the merit function ψ_p at all points outside the solution set, whenever F is monotone and in C^1 on dom $u \cap X$. (See also [30] for convergence rate results.) So, if Newton's method is supplied with a line search with respect to the primal gap function, it is globally convergent for the solution of variational inequality problems.

The merit function ψ and the optimization problem (9) cover several examples previously considered for the solution of (6).

The primal gap function, as typically all other gap functions, is nonconvex, and further also nondifferentiable in general. In order to utilize methods from differentiable optimization, we consider letting φ be strictly convex, whence the solution y^t to the inner problem (7) is unique. Under the additional assumption that dom $u \cap X$ is bounded and that u is in C^1 on this set, ψ is in C^1 on dom $u \cap X$. Among the known differentiable gap functions that are covered by this class of merit functions we find those of [1,18,26,40], and [31,59,60,61].

Descent Properties

Optimization

Assume that x^t is not a solution to (2). We are interested in the conditions under which the direction of $d^t := y^t - x^t$ provides a descent direction for the merit function T. Let $d^t := \overline{y}^t - x^t$, where $\overline{y}^t$ is a possibly inexact solution to (3). Then, if $\Phi^t = \nabla \varphi^t$, the requirement is that

$$T_{\varphi^t}(\overline{y}^t) < T_{\varphi^t}(x^t), \tag{11}$$

that is, any improvement in the value of the subproblem objective over that at the current iterate is enough to provide a descent direction. To establish this result, one simply utilizes the convexity of φ^t and u and the formula for the directional derivative of T in the direction of d^t (see [45, Prop. 2.14.b]). We further note that (11) is possible to satisfy if and only if x^t is not a solution to (2); this result is in fact a special case of Theorem 1.

If Φ^t has stronger monotonicity properties, descent is also obtained when Φ^t is not necessarily a gradient mapping, and, further, if it is Lipschitz continuous then we can establish measures of the steepness of the search directions, extending the gradient relatedness conditions of unconstrained optimization. Let Φ^t be *strongly monotone* on dom $u \cap X$, that is, for $x, y \in \text{dom } u \cap X$,

$$[\Phi^t(x) - \Phi^t(y)]^\top (x - y) \geq m_{\Phi^t} \|x - y\|^2, \tag{12}$$

for some $m_{\Phi^t} > 0$. This can be used to establish that

$$T'(x^t; d^t) \leq -m_{\Phi^t} \left\| d^t \right\|^2.$$

If y^t is not an exact solution to (3), in the sense that for a vector $\overline{y}^t$, we satisfy a perturbation of (3) where its right-hand side 0^n is replaced by $r^t \neq 0^n$, then $d^t := \overline{y}^t - x^t$ is a descent direction for T at x^t if $\| r^t \| < m_{\Phi^t} \| d^t \|$.

Variational Inequality Problems

The requirements for obtaining a descent direction in the problem (6) are necessarily much stronger than in the problem (2), the reason being the much more complex form that the merit functions for (6) takes. (For example, the directional derivative of T at x in any direction d depends only on those quantities, while the directional derivative of ψ depends also on the argument y which defines its value at x.) Typically, monotonicity of the mapping F is required, as is evidenced in the above example of the Newton method. If further a differentiable merit function is used, the requirements are slightly strengthened, as the following example result shows.

Theorem 3 *(Descent properties, [45,60]) Assume that X is bounded, u is finite on X and F is monotone and in C^1 on X. Let $\varphi: X \times X \to \boldsymbol{R}$ be a continuously differentiable function on $X \times X$ of the form $\varphi(y, x)$, strictly convex in y for each $x \in X$. Let $\alpha > 0$. Let $x \in X$, y be the unique vector in X satisfying*

$$\psi_\alpha(x) := \max_{y \in X} L_\alpha(y, x),$$

where

$$\begin{aligned} L_\alpha(y, x) &:= u(x) - u(y) \\ &+ \frac{1}{\alpha}[\varphi(x, x) - \varphi(y, x)] \\ &+ \left[F(x) - \frac{1}{\alpha}\nabla_y \varphi(x, x)\right]^\top (x - y). \end{aligned}$$

Then, with $d := y - x$, either d satisfies

$$\psi'_\alpha(x; d) \le -\gamma \psi_\alpha(x), \quad \gamma \in (0, 1),$$

or

$$\psi_\alpha(x) \le -\frac{1}{\alpha(1-\gamma)}(\varphi(y, x) + \nabla_x \varphi(y, x)^\top d).$$

A descent algorithm is devised from this result as follows. For a given $x \in X$ and choice of $\alpha > 0$, the CA subproblem is solved with the scaled cost approximating, continuous and iteration-dependent function φ. If the resulting direction does not have the descent property, then the value of α is increased and the CA subproblem rescaled and resolved. Theorem 3 shows that a sufficient increase in the value of α will produce a descent direction unless x solves (6).

Steplength Rules

In order to establish convergence of the algorithm, the steplength taken in the direction of d^t must be such that the value of the merit function decreases sufficiently. An exact line search obviously works, but we will introduce simpler steplength rules that do not require a one-dimensional minimization to be performed.

The first is the *Armijo rule*. We assume temporarily that $u = 0$. Let $\alpha, \beta \in (0, 1)$, and $\ell := \beta^{\bar{\imath}}$, where $\bar{\imath}$ is the smallest nonnegative integer i such that

$$f(x^t + \beta^i d^t) - f(x^t) \le \alpha \beta^i \nabla f(x^t)^\top d^t. \tag{13}$$

There exists a finite integer such that (13) is satisfied for any search direction $\overline{d}^t := \overline{y}^t - x^t$ satisfying (11), by the descent property and Taylor's formula (see [45, Lemma 2.24.b]).

In the case where $u \neq 0$, however, the situation becomes quite different, since $T := f + u$ is nondifferentiable. Simply replacing $\nabla f(x^t)^\intercal\ d^t$ with $T'(x^t; d^t)$ does not work. We can however use an overestimate of the predicted decrease $T'(x^t; d^t)$. Let $\alpha, \beta \in (0, 1)$, and $\ell := \beta^{\bar{\imath}}$, where $\bar{\imath}$ is the smallest nonnegative integer i such that

$$\begin{aligned} &T(x^t + \beta^i d^t) - T(x^t) \\ &\qquad \le \alpha \beta^i [\nabla \varphi^t(x^t) - \nabla \varphi^t(y^t)]^\top d^t, \end{aligned}$$

where now y^t necessarily is an exact solution to (3), and φ^t must further be strictly convex. We note that $T'(x^t; d^t) \le [\nabla \varphi^t(x^t) - \nabla \varphi^t(y^t)]^\intercal\ d^t$ indeed holds, with equality in the case where $u = 0$ and $X = \mathbf{R}^n$ (see [45, Remark 2.28]).

To develop still simpler steplength rules, we further assume that ∇f is Lipschitz continuous, that is, that for $x, y \in \operatorname{dom} u \cap X$,

$$\|\nabla f(x) - \nabla f(y)\| \le M_{\nabla f} \|x - y\|,$$

for some $M_\nabla f > 0$. The Lipschitz continuity assumption implies that for every $\ell \in [0, 1]$,

$$\begin{aligned} &T(x^t + \ell d^t) - T(x^t) \\ &\quad \le \ell \left[\nabla \varphi^t(x^t) - \nabla \varphi^t(y^t)\right]^\top d^t \\ &\quad + \frac{M_{\nabla f}}{2} \ell^2 \left\| d^t \right\|^2; \end{aligned}$$

adding a strong convexity assumption on φ^t yields that

$$T(x^t + \ell d^t) - T(x^t) \le \ell \left(-m_{\varphi^t} + \frac{M_{\nabla f} \ell}{2}\right) \left\| d^t \right\|^2.$$

This inequality can be used to validate the *relaxation step*, which takes

$$\ell_t \in \left(0, \frac{2m_{\varphi^t}}{M_{\nabla f}}\right) \cap [0,1], \tag{14}$$

and the *divergent series steplength rule*,

$$[0,1] \supset \{\ell_t\} \to 0, \quad \sum_{t=0}^{\infty} \ell_t = \infty. \tag{15}$$

In the case of (14), descent is guaranteed in each step, while in the case of (15), descent is guaranteed after a finite number of iterations.

Convergence Properties

Convergence of the CA algorithm can be established under many combinations of

i) the properties of the original problem mappings;
ii) the choice of forms and convexity properties of the cost approximating mappings;
iii) the choice of accuracy in the computations of the CA subproblem solutions;
iv) the choice of merit function; and
v) the choice of steplength rule.

A subset of the possible results is found in [45, Chapt. 5–9]. Evident from these results is that convergence relies on reaching a critical mass in the properties of the problem and algorithm, and that, given that this critical mass is reached, there is a very large freedom-of-choice how this mass is distributed. So, for example, weaker properties in the monotonicity of the subproblem must be compensated both by stronger coercivity conditions on the merit function and by the use of more accurate subproblem solutions and steplength rules.

Decomposition CA Algorithms

Assume that dom $u \cap X$ is a *Cartesian product set*, that is, for some finite index set $\mathcal{C}$ and positive integers n_i with $\sum_{i\in\mathcal{C}} n_i = n$,

$$X = \prod_{i\in\mathcal{C}} X_i, \quad X_i \subseteq \mathbb{R}^{n_i};$$
$$u(x) = \sum_{i\in\mathcal{C}} u_i(x_i), \quad u_i\colon \mathbb{R}^{n_i} \to \mathbb{R} \cup \{+\infty\}.$$

Such problems arise in applications of equilibrium programming, for example in traffic ([41]) and Nash equilibrium problems ([21]); of course, box constrained and unconstrained problems fit into this framework as well.

The main advantage of this problem structure is that one can devise several decomposition versions of the CA algorithm, wherein components of the original problem are updated upon in parallel or sequentially, independently of each other. With the right computer environment at hand, this can mean a dramatic increase in computing efficiency. We will look at three computing models for decomposition CA algorithm, and compare their convergence characteristics. In all three cases, decomposition is achieved by choosing the cost approximating mapping separable with respect to the partition of $\mathbf{R}^n$ defined by $\mathcal{C}$:

$$\Phi(x)^\top = [\Phi_1(x_1)^\top, \ldots, \Phi_{|\mathcal{C}|}(x_{|\mathcal{C}|})^\top]. \tag{16}$$

The individual subproblems, given x, then are to find y_i, $i \in \mathcal{C}$, such that

$$\Phi_i(y_i) + \partial u_i(y_i) + N_{X_i}(y_i) + F_i(x) - \Phi_i(x_i) \ni 0^{n_i};$$

if $\Phi_i \equiv \nabla\varphi_i$ for some convex function φ_i: dom $u_i \cap X_i \to \mathbf{R}$ in C^1 on dom $u_i \cap X_i$, then this is the optimality conditions for

$$\min_{y_i\in X_i} T_{\varphi_i}(y_i) := \varphi_i(y_i) + u_i(y_i) + [F_i(x) - \nabla\varphi_i(x_i)]^\top y_i .$$

Sequential Decomposition

The *sequential CA algorithm* proceeds as follows. Given an iterate $x^t \in \text{dom } u \cap X$ at iteration t, choose an index $i_t \in \mathcal{C}$ and a cost approximating mapping $\Phi_{i_t}^t$, and solve the problem of finding $y_{i_t}^t \in \mathbf{R}^{n_{i_t}}$ such that $(i = i_t)$

$$\Phi_i^t(y_i^t) + \partial u_i(y_i^t) + N_{X_i}(y_i^t) + F_i(x^t) - \Phi_i^t(x_i^t) \ni 0^{n_i}.$$

Let $y_j^t := x_j^t$ for all $j \in \mathcal{C} \setminus \{i_t\}$ and $d^t := y^t - x^t$. The next iterate, x^{t+1}, is then defined by $x^{t+1} := x^t + \ell_t d^t$, that is,

$$x_j^{t+1} := \begin{cases} x_j^t + \ell_t(y_j^t - x_j^t), & j = i_t, \\ x_j^t, & j \neq i_t, \end{cases}$$

for some value of ℓ_t such that $x_{i_t}^t + \ell_t(y_{i_t}^t - x_{i_t}^t) \in \text{dom } u_{i_t} \cap X_{i_t}$ and the value of a merit function ψ is reduced sufficiently.

Assume that F is the gradient of a function f: dom $u \cap X \to \mathbf{R}$. Let the sequence $\{i_t\}$ be chosen according to the *cyclic rule*, that is, in iteration t,

$$i_t := t \pmod{|\mathcal{C}|} + 1.$$

Choose the cost approximating mapping ($i = i_t$)

$$y_i \mapsto \Phi_i^t(y_i) := \nabla_i f(x_{\neq i}^t, y_i), \qquad y_i \in \operatorname{dom} u_i \cap X_i.$$

Note that this mapping is monotone whenever f is convex in x_i. Since $\Phi_i^t(x_i^t) = \nabla_i f(x^t)$, the CA subproblem is equivalent (under this convexity assumption) to finding

$$y_i^t \in \arg\min_{y_i \in X_i} \{f(x_{\neq i}^t, y_i) + u_i(y_i)\}.$$

An exact line search would produce $\ell_t := 1$, since y_i^t minimizes $f(x_{\neq i}, \cdot) + u_i$ over dom $u_i \cap X_i$ (the remaining components of x kept fixed), and so $x_i^{t+1} := y_i^t$. The iteration described is that of the classic *Gauss–Seidel algorithm* ([35]) (also known as the *relaxation algorithm*, the *coordinate descent method*, and the *method of successive displacements*), originally proposed for the solution of unconstrained problems. The Gauss–Seidel algorithm is hence a special case of the sequential CA algorithm.

In order to compare the three decomposition approaches, we last provide the steplength requirement in the relaxation steplength rule (cf. (14)). The following interval is valid under the assumptions that for each $i \in \mathcal{C}$, $\nabla_i f$ is Lipschitz continuous on dom $u_i \cap X_i$ and each mapping Φ_i^t is strongly monotone:

$$\ell_{i,t} \in \left(0, \frac{2m_{\Phi_i^t}}{M_{\nabla_i f}}\right) \cap [0, 1]. \tag{17}$$

Synchronized Parallel Decomposition

The synchronized parallel CA algorithm is identical to the original scheme, where the CA subproblems are constructed to match the separability structure in the constraints.

We presume the existence of a multiprocessor powerful enough to solve the $|\mathcal{C}|$ CA subproblems in parallel. (If fewer than $|\mathcal{C}|$ processors are available, then either some of the subproblems are solved in sequence or, if possible, the number of components is decreased; in either case, the convergence analysis will be the same, with the exception that the value of $|\mathcal{C}|$ may change.)

In the sequential decomposition CA algorithm, the steplengths are chosen individually for the different variable components, whereas the original CA algorithm uses a uniform steplength, ℓ_t. If the relative scaling of the variable components is poor, in the sense that F or u changes disproportionally to unit changes in the different variables x_i, $i \in \mathcal{C}$, then this ill-conditioning may result in a poor performance of the parallel algorithm. Being forced to use the same steplength in all the components can also have an unwanted effect due to the fact that the values of some variable components are close to their optimal ones while others may be far from optimal, in which case one might for example wish to use longer steps for the latter components. These two factors lead us to introduce the possibility to scale the component directions in the synchronized parallel CA algorithm. We stress that such effects cannot in general be accommodated into the original algorithm through a scaling of the mappings Φ_i^t. The scaling factors $s_{i,t}$ introduced are assumed to satisfy

$$0 < \underline{s}_i \le s_{i,t} \le 1, \quad i \in \mathcal{C}.$$

Note that the upper bound of one is without any loss of generality.

Assume that F is the gradient of a function f: dom $u \cap X \to \mathbf{R}$. In the parallel algorithm, choose the cost approximating mapping of the form (16), where for each $i \in \mathcal{C}$,

$$y_i \mapsto \Phi_i^t(y_i) := \nabla_i f(x_{\neq i}^t, y_i), \qquad y_i \in \operatorname{dom} u_i \cap X_i.$$

This mapping is monotone on dom $u \cap X$ whenever f is convex in each component x_i. Since $\Phi_i^t(x_i^t) = \nabla_i f(x^t)$, $i \in \mathcal{C}$, it follows that the CA subproblem is equivalent (under the above convexity assumption on f) to finding

$$y_i^t \in \arg\min_{y_i \in X_i} \{f(x_{\neq i}^t, y_i) + u_i(y_i)\}.$$

Choosing $\ell_t := 1$ and $s_{i,t} := 1$, $i \in \mathcal{C}$, yields $x^{t+1} := y^t$, and the resulting iteration is that of the *Jacobi algorithm* [8,35] (also known as the *method of simultaneous displacements*). The Jacobi algorithm, which was originally proposed for the solution of systems of equations,

is therefore a parallel CA algorithm where the cost approximating mapping is (18) and unit steps are taken.

The admissible step in component i is $\ell s_{i,t} \in [0, 1]$, where

$$\ell \in \left(0, \min_{i \in \mathcal{C}} \left\{ \frac{2m_{\Phi_i^t}}{s_{i,t} M_{\nabla f}} \right\}\right). \tag{18}$$

The maximal step is clearly smaller than in the sequential approach. To this conclusion contributes both the minimum operation and that $M_{\nabla_i f} \leq M_{\nabla f}$; both of these requirements are introduced here because the update is made over all variable components simultaneously. (An intuitive explanation is that the sequential algorithm utilizes more recent information when it constructs the subproblems.) One may therefore expect the sequential algorithm to converge to a solution with a given accuracy in less iterations, although the parallel algorithm may be more efficient in terms of solution time; the scaling introduced by $s_{i,t}$ may also improve the performance of the parallel algorithm to some extent.

Although the parallel version of the algorithm may speed-up the practical convergence rate compared to the sequential one, the need for synchronization in carrying out the updating step will generally deteriorate performance, since faster processors must wait for slower ones. In the next section, we therefore introduce an asynchronous version of the parallel algorithm, in which processors do not wait to receive the latest information available.

Asynchronous Parallel Decomposition

In the algorithms considered in this Section, the synchronization step among the processors is removed. Because the speed of computations and communications can vary among the processors, and communication delays can be substantial, processors will perform the calculations out of phase with each other. Thus, the advantage of reduced synchronization is paid for by an increase in interprocessor communications, the use of outdated information, and a more difficult convergence detection (see [8]). (Certainly, the convergence analysis also becomes more complicated.) Recent numerical experiments indicate, however, that the introduction of such *asynchronous computations* can substantially enhance the efficiency of parallel iterative methods (e. g., [6,11,15]).

The model of partial asynchronism that we use is as follows. For each processor (or, variable component) $i \in \mathcal{C}$, we introduce

a) initial conditions, $x_i(t) := x_i^0 \in X_i$, for all $t \leq 0$;
b) a set $\mathcal{T}^i$ of times at which x_i is updated; and
c) a variable $\tau_j^i(t)$ for each $j \in \mathcal{C}$ and $t \in \mathcal{T}^i$, denoting the time at which the value of x_j used by processor i at time t is generated by processor j, satisfying $0 \leq \tau_j^i(t) \leq t$ for all $j \in \mathcal{C}$ and $t \geq 0$.

We note that the sequential CA algorithm and the synchronized parallel CA algorithm can both be expressed as asynchronous algorithms: the cyclic sequential algorithm model is obtained from the choices $\mathcal{T}^i := \cup_{k \geq 0}\{|\mathcal{C}|\ k + i - 1\}$ and $\tau_j^i(t) := t$, while the synchronous parallel model is obtained by choosing $\mathcal{T}^i := \{1, 2, \ldots\}$ and $\tau_j^i(t) := t$, for all i, j and t.

The communication delay from processor j to processor i at time t is $t - \tau_j^i(t)$. The convergence of the *partially asynchronous parallel decomposition CA algorithm* is based on the assumption that this delay is upper bounded: there exists a positive integer P such that

i) for every $i \in \mathcal{C}$ and $t \geq 0$, at least one element of $\{t, \ldots, t + P - 1\}$ belongs to $\mathcal{T}^i$;
ii) $0 \leq t - \tau_j^i(t) \leq P - 1$ holds for all $i, j \in \mathcal{C}$ and all $t \geq 0$; and
iii) $\tau_i^i(t) = t$ holds for all $i \in \mathcal{C}$ and all $t \geq 0$.

In short, parts i) and ii) of the assumption state that no processor waits for an arbitrarily long time to compute a subproblem solution or to receive a message from another processor. (Note that a synchronized model satisfies $P = 1$.) Part iii) of the assumption states that processor i always uses the most recent value of its own component x_i of x, and is in [58] referred to as a *computational nonredundancy condition*. This condition holds in general when no variable component is updated simultaneously by more than one processor, as, for example, in message passing systems. For further discussions on the assumptions, we refer the reader to [8,57]; we only remark that they are easily enforced in practical implementations.

The iterate $x(t)$ is defined by the vector of $x_i(t)$, $i \in \mathcal{C}$. At a given time t, processor i has knowledge of a possibly outdated version of $x(t)$; we let

$$x^i(t)^\top := \left[x_1(\tau_1^i(t))^\top, \ldots, x_{|\mathcal{C}|}(\tau_{|\mathcal{C}|}^i(t))^\top\right]$$

denote this vector. (Note that iii) above implies the relation $x_i^i(t) := x_i(\tau_i^i(t)) = x_i(t)$.)

To describe the (partially) asynchronous parallel CA algorithm, processor i updates $x_i(t)$ according to

$$x_i(t+1) := x_i(t) + \ell s_i(y_i(t) - x_i^i(t)), \qquad t \in \mathcal{T}^i,$$

where $y_i(t)$ solves the CA subproblem defined at $x^i(t)$, and $s_i \in (0, 1]$ is a scaling parameter. (We define $d_i(t) := y_i(t) - x_i^i(t)$ to be zero at each $t \notin \mathcal{T}^i$.)

The admissible steplength for $i \in \mathcal{C}$ is $\ell s_i \in [0, 1]$, where

$$\ell \in \left(0, \frac{2 \min_{i \in \mathcal{C}} \{\frac{m_{\Phi_i}}{s_i}\}}{M_{\nabla f}[1 + (|\mathcal{C}| + 1)P]}\right). \tag{19}$$

If further for some $M \geq 0$ and every $i \in \mathcal{C}$, all vectors x, y in dom u $\cap X$ with $x_i = y_i$ satisfy

$$\|\nabla_i f(x) - \nabla_i f(y)\| \leq M \|x - y\|, \tag{20}$$

then, in the above result, the steplength restrictions are adjusted to

$$\ell \in \left(0, \frac{2 \min_{i \in \mathcal{C}} \{\frac{m_{\Phi_i}}{s_i}\}}{M_{\nabla f} + (|\mathcal{C}| + 1)MP}\right).$$

(We interpret the property (20) as a quantitative measure of the coupling between the variables.)

Most important to note is that the upper bound on ℓ is (essentially) inversely proportional to the maximal allowed asynchronism P; this is very intuitive, since if processors take longer steps then they should exchange information more often. Conversely, the more outdated the information is, the less reliable it is, hence the shorter step.

The relations among the steplengths in the three approaches (cf. (17), (18), and (19)) quantify the intuitive result that utilizing an increasing degree of parallelism and asynchronism results in a decreasing quality of the step directions, due to the usage of more outdated information; subsequently, smaller steplengths must be used. More detailed discussions about this topic is found in [45, Sect. 8.7.2].

See also

► Dynamic Traffic Networks

References

1. Auchmuty G (1989) Variational principles for variational inequalities. Numer Funct Anal Optim 10:863–874
2. Auslender A (1976) Optimisation: Méthodes numériques. Masson, Paris
3. Auslender A, Crouzeix JP, Fedit P (1987) Penalty-proximal methods in convex programming. J Optim Th Appl 55:1–21
4. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: Theory and algorithms, 2nd edn. Wiley, New York
5. Bertsekas DP (1999) Nonlinear programming, 2nd edn. Athena Sci., Belmont, MA
6. Bertsekas DP, Castanon DA (1991) Parallel synchronous and asynchronous implementations of the auction algorithm. Parallel Comput 17:707–732
7. Bertsekas DP, Tseng P (1994) Partial proximal minimization algorithms for convex programming. SIAM J Optim 4: 551–572
8. Bertsekas DP, Tsitsiklis JN (1989) Parallel and distributed computation: Numerical methods. Prentice-Hall, Englewood Cliffs, NJ
9. Bregman LM (1966) A relaxation method of finding a common point of convex sets and its application to problems of optimization. Soviet Math Dokl 7:1578–1581
10. Browder FE (1966) Existence and approximation of solutions of nonlinear variational inequalities. Proc Nat Acad Sci USA 56:1080–1086
11. Chajakis ED, Zenios SA (1991) Synchronous and asynchronous implementations of relaxation algorithms for nonlinear network optimization. Parallel Comput 17: 873–894
12. Cohen G (1978) Optimization by decomposition and coordination: A unified approach. IEEE Trans Autom Control AC-23:222–232
13. Cominetti R (1997) Coupling the proximal point algorithm with approximation methods. J Optim Th Appl 95: 581–600
14. Eckstein J (1993) Nonlinear proximal point algorithms using Bregman functions, with applications to convex programming. Math Oper Res 18:202–226
15. El Baz D (1989) A computational experience with distributed asynchronous iterative methods for convex network flow problems. Proc. 28th IEEE Conf. Decision and Control, pp 590–591
16. Fletcher R (1987) Practical methods of optimization, 2nd edn. Wiley, New York
17. Frank M, Wolfe P (1956) An algorithm for quadratic programming. Naval Res Logist Quart 3:95–110
18. Fukushima M (1992) Equivalent differentiable optimization problems and descent methods for asymmetric variational inequality problems. Math Program 53:99–110
19. Goldstein AA (1964) Convex programming in Hilbert space. Bull Amer Math Soc 70:709–710

20. Ha CD (1990) A generalization of the proximal point algorithm. SIAM J Control Optim 28:503–512
21. Harker PT, Pang J-S (1990) Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications. Math Program 48:161–220
22. Kiwiel KC (1998) Generalized Bregman projections in convex feasibility problems. J Optim Th Appl 96:139–157
23. Kiwiel KC (1998) Subgradient method with entropic projections for convex nondifferentiable minimization. J Optim Th Appl 96:159–173
24. Korpelevich GM (1977) The extragradient method for finding saddle points and other problems. Matekon 13:35–49
25. Larsson T, Migdalas A (1990) An algorithm for nonlinear programs over Cartesian product sets. Optim 21:535–542
26. Larsson T, Patriksson M (1994) A class of gap functions for variational inequalities. Math Program 64:53–79
27. Levitin ES, Polyak BT (1966) Constrained minimization methods. USSR Comput Math Math Phys 6:1–50
28. Luo Z-Q, Tseng P (1991) On the convergence of a matrix splitting algorithm for the symmetric monotone linear complementarity problem. SIAM J Control Optim 29:1037–1060
29. Mangasarian OL (1991) Convergence of iterates of an inexact matrix splitting algorithm for the symmetric monotone linear complementarity problem. SIAM J Optim 1:114–122
30. Marcotte P, Dussault J-P (1989) A sequential linear programming algorithm for solving monotone variational inequalities. SIAM J Control Optim 27:1260–1278
31. Marcotte P, Zhu D (1995) Global convergence of descent processes for solving non strictly monotone variational inequalities. Comput Optim Appl 4:127–138
32. Martinet B (1972) Détermination approchée d'un point fixe d'une application pseudo-contractante. CR Hebdom Séances de l'Acad Sci (Paris), Sér A 274:163–165
33. Minty GJ (1962) Monotone (nonlinear) operators in Hilbert space. Duke Math J 29:341–346
34. Moreau J-J (1965) Proximité et dualité dans un espace Hilbertien. Bull Soc Math France 93:273–299
35. Ortega JM, Rheinboldt WC (1970) Iterative solution of nonlinear equations in several variables. Acad. Press, New York
36. Pang J-S (1982) On the convergence of a basic iterative method for the implicit complementarity problem. J Optim Th Appl 37:149–162
37. Patriksson M (1993) Partial linearization methods in nonlinear programming. J Optim Th Appl 78:227–246
38. Patriksson M (1993) A unified description of iterative algorithms for traffic equilibria. Europ J Oper Res 71:154–176
39. Patriksson M (1993) A unified framework of descent algorithms for nonlinear programs and variational inequalities. PhD Thesis Dept. Math. Linköping Inst. Techn.
40. Patriksson M (1994) On the convergence of descent methods for monotone variational inequalities. Oper Res Lett 16:265–269
41. Patriksson M (1994) The traffic assignment problem – Models and methods. Topics in Transportation. VSP, Utrecht
42. Patriksson M (1997) Merit functions and descent algorithms for a class of variational inequality problems. Optim 41:37–55
43. Patriksson M (1998) Cost approximation: A unified framework of descent algorithms for nonlinear programs. SIAM J Optim 8:561–582
44. Patriksson M (1998) Decomposition methods for differentiable optimization problems over Cartesian product sets. Comput Optim Appl 9:5–42
45. Patriksson M (1998) Nonlinear programming and variational inequality problems: A unified approach. Applied Optim, vol 23. Kluwer, Dordrecht
46. Patriksson M (1999) Cost approximation algorithms with nonmonotone line searches for a general class of nonlinear programs. Optim 44:199–217
47. Polyak BT (1963) Gradient methods for the minimisation of functionals. USSR Comput Math Math Phys 3:864–878
48. Polyak BT (1967) A general method of solving extremum problems. Soviet Math Dokl 8:593–597
49. Polyak BT (1987) Introduction to optimization. Optim. Software, New York
50. Pshenichny BN, Danilin Yu M. (1978) Numerical methods in extremal problems. MIR, Moscow
51. Rockafellar RT (1976) Augmented Lagrangians and applications of the proximal point algorithm in convex programming. Math Oper Res 1:97–116
52. Rockafellar RT (1976) Monotone operators and the proximal point algorithm. SIAM J Control Optim 14:877–898
53. Shor NZ (1985) Minimization methods for non-differentiable functions. Springer, Berlin
54. Teboulle M (1997) Convergence of proximal-like algorithms. SIAM J Optim 7:1069–1083
55. Tikhonov AN (1963) Solution of incorrectly formulated problems and the regularization method. Soviet Math Dokl 4:1035–1038
56. Tikhonov AN, Arsenin VYa (1977) Solutions of ill-posed problems. Wiley, New York (translated from Russian)
57. Tseng P, Bertsekas DP, Tsitsiklis JN (1990) Partially asynchronous, parallel algorithms for network flow and other problems. SIAM J Control Optim 28:678–710
58. Üresin A, Dubois M (1992) Asynchronous iterative algorithms: Models and convergence. In: Kronsjö L, Shumsheruddin D (eds) Advances in Parallel Algorithms. Blackwell, Oxford, pp 302–342
59. Wu JH, Florian M, Marcotte P (1993) A general descent framework for the monotone variational inequality problem. Math Program 61:281–300
60. Zhu DL, Marcotte P (1993) Modified descent methods for solving the monotone variational inequality problem. Oper Res Lett 14:111–120
61. Zhu DL, Marcotte P (1994) An extended descent framework for variational inequalities. J Optim Th Appl 80: 349–366

62. Zuhovickiĭ SI, Polyak RA, Primak ME (1969) Two methods of search for equilibrium points of n-person concave games. Soviet Math Dokl 10:279–282

Credit Rating and Optimization Methods

Constantin Zopounidis, Michael Doumpos
Department of Production Engineering and Management, Financial Engineering Laboratory, Technical University of Crete, Chania, Greece

MSC2000: 91B28 90C90 90C05 90C20 90C30

Article Outline

Synonyms
Introduction/Background
Definitions
Formulation
Methods/Applications
  Logistic Regression
  Neural Networks
  Support Vector Machines
  Multicriteria Value Models and Linear Programming Techniques
  Evolutionary Optimization
Conclusions
References

Synonyms

Credit scoring; Credit granting; Financial risk management; Optimization

Introduction/Background

Financial risk management has evolved over the past two decades in terms of both its theory and its practices. Economic uncertainties, changes in the business environment and the introduction of new complex financial products (e. g., financial derivatives) led financial institutions and regulatory authorities to the development of a new framework for financial risk management, focusing mainly on the capital adequacy of banks and credit institutions.

Banks and other financial institutions are exposed to many different forms of financial risks. Usually these are categorized as [14]:

- Market risk that arises from the changes in the prices of financial securities and currencies.
- Credit risk originating from the inability of firms and individuals to meet their debt obligations to their creditors.
- Liquidity risk that arises when a transaction cannot be conducted at the existing market prices or when early liquidation is required in order to meet payments obligations.
- Operational risk that originate from human and technical errors or accidents.
- Legal risk which is due to legislative restrictions on financial transactions.

Among these types of risk, credit risk is considered as the primary financial risk in the banking system and exists in virtually all income-producing activities [7]. How a bank selects and manages its credit risk is critically important to its performance over time.

In this context credit risk management defines the whole range of activities that are implemented in order to measure, monitor and minimize credit risk. Credit risk management has evolved dramatically over the last 20 years. Among others, some factors that have increased the importance of credit risk management include [2]: (i) the worldwide increase in the number of bankruptcies, (ii) the trend towards disintermediation by the highest quality and largest borrowers, (iii) the increased competition among credit institutions, (iv) the declining value of real assets and collateral in many markets, and (v) the growth of new financial instruments with inherent default risk exposure, such as credit derivatives.

Early credit risk management was primarily based on empirical evaluation systems of the creditworthiness of a client. CAMEL has been the most widely used system in this context, which is based on the empirical combination of several factors related to capital, assets, management, earnings and liquidity.

It was soon realized, however, that such empirical systems cannot provide a solid and objective basis for credit risk management. This led to an outgrowth of studies from academics and practitioners on the development of new credit risk assessment systems. These efforts were also motivated by the changing regulatory framework that now requires banks to implement specific methodologies for managing and monitoring their credit portfolios [4].

The existing practices are based on sophisticated statistical and optimization methods, which are used to develop a complete framework for measuring and monitoring credit risk. Credit rating models are in the core of this framework and are used to assess the creditworthiness of firms and individuals. The following sections describe the functionality of credit rating systems and the type of optimization methods that are used in some popular techniques for developing rating systems.

Definitions

As already noted, credit risk is defined as the likelihood that an obligor (firm or individual) will be unable or unwilling to fulfill debt obligations towards the creditors. In such a case, the creditors will suffer losses that have to be measured as accurately as possible.

The expected loss L_{it} over a period t from granting credit to a given obligor i can be measured as follows:

$$L_{it} = PD_{it} LGD_i EAD_i$$

where PD_{it} is the probability of default for the obligor i in the time period t, LGD_i is the percentage of exposure the bank might lose in case the borrower defaults and EAD_i is the amount outstanding in case the borrower defaults. The time period t is usually taken equal to one year.

In the new regulatory framework default is considered to have occurred with regard to a particular obligor when one or more of the following events has taken place [4,11]:

- it is determined that the obligor is unlikely to pay its debt obligations in full;
- a credit loss event associated with any obligation of the obligor;
- the obligor is past due more than 90 days on any credit obligation; or
- the obligor has filed for bankruptcy or similar protection from creditors.

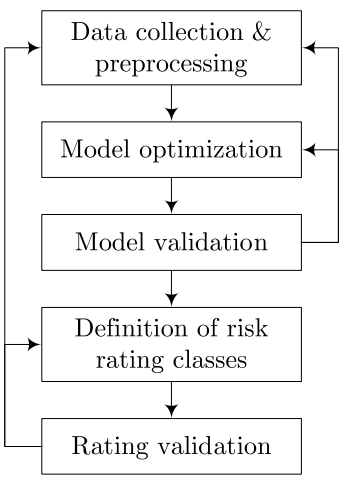
The aim of credit rating models is to assess the probability of default for an obligor, whereas other models are used to estimate LGD and EAD. Rating systems measure credit risk and differentiate individual credits and groups of credits by the risk they pose. This allows bank management and examiners to monitor changes and trends in risk levels thus promoting safety and soundness in the credit granting process. Credit rating systems are also used for credit approval and underwriting, loan pricing, relationship management and credit administration, allowance for loan and lease losses and capital adequacy, credit portfolio management and reporting [7].

Formulation

Generally, a credit rating model can be considered as a mapping function $f : \mathbb{R}^n \longrightarrow G$ that estimates the probability of default of an obligor described by a vector $\boldsymbol{x} \in \mathbb{R}^n$ of input features and maps the result to a set G of risk categories. The feature vector $\boldsymbol{x}$ represents all the relevant information that describes the obligor, including financial and nonfinancial data.

The development of a rating model is based on the process of Fig. 1.

The process begins with the collection of appropriate data regarding known cases in default and nondefault cases. These data can be taken from the historical database of a bank, or from external resources. At this data selection stage, some preprocessing of the data is necessary in order to transform the obtained data into useful features, to clean out the data from possible outliers and to select the appropriate set of features for the analysis. These steps lead to the final data $\{\boldsymbol{x}_i, y_i\}_{i=1}^m$, where $\boldsymbol{x}_i$ is the input feature vector for obligor i, y_i in the known status of the obligor (e. g. $y_i = -1$ for cases



Credit Rating and Optimization Methods, Figure 1
The process for developing credit rating models

in default and $y_i = 1$ for nondefault cases), and m in the number of observations in the data set. These data, which are used for model development, are usually referred to as *training data*.

The second stage involves the optimization process, which refers to the identification of the model's parameters that best fit the training data. In the simplest case, the model can be expressed as a linear function of the form:

$$f(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{\beta} + \beta_0$$

where $\boldsymbol{\beta} \in \mathbb{R}^n$ is the vector with the coefficients of the selected features in the model and β_0 is a constant term. Other types of nonlinear models are also applicable.

In the above linear case, the objective of the optimization process is to identify the optimal parameter vector $\boldsymbol{\alpha} = (\boldsymbol{\beta}, \beta_0)$ that best fit the training data. This can be expressed as an optimization problem of the following general form:

$$\min_{\boldsymbol{\alpha} \in S} \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{X}) \tag{1}$$

where S is a set of constraints that define the feasible (acceptable) values for the parameter vector $\boldsymbol{\alpha}$, $\boldsymbol{X}$ is the training data set and $\mathcal{L}$ is a loss function measuring the differences between the model's output and the given classification of the training observations.

The result of the model optimization process are validated using another sample of obligors with known status. This is referred to as the *validation sample*. Typically it consists of cases different than the ones of the training sample and for a future time period. The optimal model is applied to these new observations and its predictive ability is measured. If this is acceptable, then the model's outputs are used to define a set of risk rating classes (usually 10 classes are used). Each rating class is associated with a probability of default and it includes borrowers with similar credit risk levels. The defined rating needs also to be validated in terms of its stability over time, the distribution of the borrowers in the rating groups, and the consistency between the estimated probabilities of default in each group and the empirical ones which are taken from the population of rated borrowers.

Methods/Applications

The optimization problem (1) is expressed in different forms depending on the method used to develop the rating model. The characteristics of some popular methods are outlined below.

Logistic Regression

Logistic regression is the most widely used method in financial decision-making problems, with numerous applications in credit risk rating. Logistic regression assumes that the log of the probability odds is a linear function:

$$\log \frac{p}{1-p} = \beta_0 + \boldsymbol{x}\boldsymbol{\beta}$$

where $p = \Pr(1 \mid \boldsymbol{x})$ is the probability that an obligor $\boldsymbol{x}$ is a member of class 1, which is then expressed as

$$p = \left[1 + \exp^{-(\beta_0 + \boldsymbol{x}\boldsymbol{\beta})}\right]^{-1}$$

The parameters of the model (constant term β_0 and coefficient vector $\boldsymbol{\beta}$) are estimated to maximize the conditional likelihood of the classification given the training data. This is expressed as

$$\max_{\beta_0, \boldsymbol{\beta} \in \mathbb{R}} \prod_{i=1}^{m} \Pr(y_i \mid \boldsymbol{x}_i)$$

which can be equivalently written as

$$\max_{\beta_0, \boldsymbol{\beta} \in \mathbb{R}} \sum_{i=1}^{m} \left[\frac{y_i + 1}{2} \ln(p_i) + \frac{1 - y_i}{2} \ln(1 - p_i)\right]$$

where $y_i = 1$ if obligor i is in the nondefault group and $y_i = -1$ otherwise.

Nonlinear optimization techniques such as the Newton algorithm are used to perform this optimization.

Logistic regression has been widely applied in credit risk rating both by academics and by practitioners [1]. Its advantages are mainly related to its simplicity and transparency: it provides direct estimates of the probabilities of default as well as estimates for the significance of the predictor variables and it is computationally feasible even for large data sets.

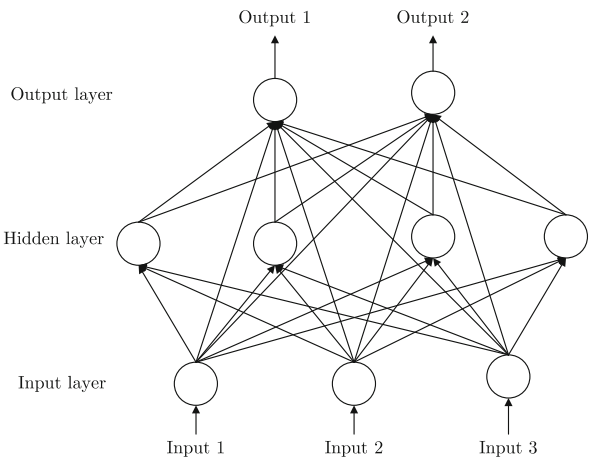
Neural Networks

Neural networks is a popular methodology for developing decision-making models in complex domains. A neural network is a network of parallel processing units (neurons) organized into layers. A typical structure of a neural network (Fig. 2) includes the following structural elements:

1. An input layer consisting of a set of nodes (processing units – neurons); one for each input to the network.
2. An output layer consisting of one or more nodes depending on the form of the desired output of the network. In classification problems, the number of nodes of the output layer is determined in accordance with the the number of groups.
3. A series of intermediate layers referred to as hidden layers. The nodes of each hidden layer are fully connected with the nodes of the subsequent and the proceeding layer.

Each connection between two nodes of the network is assigned a weight representing the strength of the connection. On the basis of the connections' weights, the input to each node is determined as the weighted average of the outputs from all the incoming connections. Thus, the input in_{ir} to node i of the hidden layer r is defined as follows:

$$in_{ir} = \sum_{j=0}^{r-1} \sum_{k=1}^{n_j} w_{ik}^{j} o_{ik} + \phi_{ir}$$



Credit Rating and Optimization Methods, Figure 2
A typical architecture of a neural network

where n_j is the number of nodes at the hidden layer j, w_{ik} is the weight of the connection between node i at layer r and node k at layer j, o_{kj} is the output of node k at layer j and ϕ_{ir} an bias term.

The output of each node is specified through a transformation function. The most common form of this function is the logistic function:

$$o_{ir} = (1 + \exp^{-in_{ij}})^{-1}$$

The determination of the optimal neural network model requires the estimation of the connection weights and the bias terms of the nodes. The most widely used network training methodology is the backpropagation approach [18]. Nonlinear optimization techniques are used for this purpose [10,13,16].

Neural networks have become increasingly popular in recent years for the development of credit rating models [3]. Their main advantages include their ability to model complex nonlinear relationships in credit data, but they have also been criticized for their lack of transparency, the difficulty of specifying a proper architecture and the increased computational resources that are needed for large data sets.

Support Vector Machines

Support vector machines (SVMs) have become an increasingly popular nonparametric methodology for developing classification models. In a dichotomous classification setting, SVMs can be used to develop a linear decision function $f(\boldsymbol{x}) = \text{sgn}(\boldsymbol{x}\boldsymbol{\beta} + \beta_0)$.

The optimal decision function f should maximize the margin induced in the separation of the classes [24], which is defined as $2/\|\boldsymbol{\beta}\|$. Thus, the estimation of the optimal model is expressed as a quadratic programming problem of the following from:

$$\begin{aligned} &\min \ \tfrac{1}{2}\boldsymbol{\beta}^{\top}\boldsymbol{\beta} + C\boldsymbol{e}^{\top}\boldsymbol{d} \\ &\text{subject to } \boldsymbol{Y}(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{e}\beta_0) + \boldsymbol{d} \geq \boldsymbol{e} \\ &\qquad \boldsymbol{\beta}, \beta_0 \in \mathbb{R}, \boldsymbol{d} \geq \boldsymbol{0} \end{aligned} \tag{2}$$

where $\boldsymbol{X}$ is an $m \times n$ matrix with the training data, $\boldsymbol{Y}$ is an $m \times m$ matrix such that $Y_{ii} = y_i$ and $Y_{ij} = 0$ for all $i \neq j$, $\boldsymbol{d}$ is $m \times 1$ vector with nonnegative error (slack) variables defined such that $d_i > 0$ iff $y_i(\boldsymbol{x}_i\boldsymbol{\beta} + \beta_0) < 1$, $\boldsymbol{e}$ is a $m \times 1$ vector of ones, and $C > 0$ is a user-defined constant representing the trade-off between the two con-

flicting objectives (maximization of the separating margin and minimization of the training errors).

SVMs can also be used to develop nonlinear models. This is achieved by mapping the problem data to a higher-dimensional space H (feature space) through a transformation of the form $\boldsymbol{x}_i\boldsymbol{x}_j^\top = \phi(\boldsymbol{x}_i)\phi^\top(\boldsymbol{x}_j)$. The mapping function ϕ is implicitly defined through a symmetric positive definite kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)\phi^\top(\boldsymbol{x}_j)$ [22]. The representation of the data using the kernel function enables the development of a linear model in the feature space H.

For large training sets several computational procedures have been proposed to enable the fast training of SVM models. Most of these procedures are based on a decomposition scheme. The optimization problem (2) is decomposed into smaller subproblems taking advantage of the sparse nature of SVM models, since only a small part of the data (the support vectors), contribute to the final form of the model. A review of the algorithms for training SVMs can be found in [6].

SVMs seem to be a promising methodology for developing credit rating models. The algorithmic optimization advances enable their application to large credit data sets and they provide a unified framework for developing both linear and nonlinear models. Recent application of SVMs in credit rating can be found in [9,12,21].

Multicriteria Value Models and Linear Programming Techniques

The aforementioned classification methods assume that the groups are defined in a nominal way (i. e., the grouping provides a simple description of the cases). However, in credit risk modeling the groups are defined in an ordinal way, in the sense that an obligor classified in a low risk group is preferred to an obligor classified in a high risk group (in terms of its probability of default). Multicriteria methods are well-suited to the study of ordinal classification problems [26].

A typical multicriteria method that is well-suited for the development of credit rating models is the UTADIS method. The method leads to the development of an multiattribute additive value function:

$$V(\boldsymbol{x}) = \sum_{j=1}^{n} w_j v_j(x_j)$$

where w_j is the weight of attribute j, and $v_j(x_j)$ is the corresponding marginal value function. Each marginal value function provides a monotone mapping of the performance of the obligors on the corresponding attribute in a scale between 0 (high risk) and 1 (low risk). According to [15], such an additive value function model is well-suited for credit scoring and is widely used by banks in their internal rating systems.

Using a piece-wise linear modeling approach, the estimation of the value function is performed based on a set of training data using linear programming techniques. For a two-class problem, the general form of the linear programming formulation is as follows [8]:

$$\begin{aligned} \min\ & d_1 + d_2 + \cdots + d_m \\ \text{subject to: } & y_i[V(\boldsymbol{x}_i) - \beta] + d_i \geq \delta, \quad 1, 2, \ldots, m \\ & w_1 + w_2 + \cdots + w_n = 1 \\ & w_j,\ d_i,\ \beta \geq 0 \end{aligned}$$

where β is a value threshold that distinguishes the two classes, δ is a small positive user-defined constant and $d_i = \max\{0,\ \delta - y_i[V(\boldsymbol{x}_i) - \beta]\}$ denotes the classification error for obligor i.

Extensions of this framework and alternative linear programming formulations with applications to credit risk rating have been presented by [5,17,19]. The main advantages of these methodologies involve their computational efficiency and the simplicity and transparency of the resulting models.

Evolutionary Optimization

Evolutionary algorithms (EA) are stochastic search and optimization heuristics inspired from the theory of natural evolution. In an EA, different possible solutions of an optimization problem constitute the individuals of a population. The quality of each individual is assessed with a fitness (objective) function. Better solutions are assigned higher fitness values than worse performing solutions. The key idea of EAs is that the optimal solution can be found if an initial population is evolved using a set of stochastic genetic operators, similar to the "survival of the fittest" mechanism of natural evolution. The fitness values of the individuals in a population are used to define how they will be propagated to subsequent generations of populations. Most EAs include operators that select individuals for reproduction, pro-

duce new individuals based on those selected, and determine the composition of the population at the subsequent generation.

Well-known EAs and similar metaheuristic techniques include, among others, genetic algorithms, genetic programming, tabu search, simulated annealing, ant colony optimization and particle swarm optimization. EAs have been used to facilitate the development of credit rating systems addressing some important issues such as feature selection, rule extraction, neural network development, etc. Some recent applications can be found in the works of Varetto [20], Salcedo-Sanza et al. [25] and Tsakonas et al. [23].

Conclusions

Credit rating systems are in the core of the new regulatory framework for the supervision of financial institutions. Such systems support the credit granting process and enable the measurement and monitoring of credit risk exposure.

The increasing volume of credit data which are available for developing rating systems highlight the importance of implementing efficient optimization techniques for the construction of rating models. The existing optimization methods used in this field, are mainly based on nonlinear optimization, linear programming and evolutionary algorithms.

Future research is expected to take advantage of the advances in computer science, algorithmic developments regarding new forms of decision models, the analysis of the combination of different models, the comparative investigation on the performance of the existing methods and the implementation into decision support system that can be used by credit analysts in their daily practice.

References

1. Altman EI, Avery R, Eisenbeis R, Stinkey J (1981) Application of Classification Techniques in Business, Banking and Finance. JAI Press, Greenwich
2. Altman EI, Saunders A (1998) Credit risk measurement: Developments over the last 20 years. J Banking Finance 21:1721–1742
3. Atiya AF (2001) Bankruptcy prediction for credit risk using neural networks: A survey and new results. IEEE Trans Neural Netw 12:929–935
4. Basel Committee on Banking Supervision (2004) International Convergence of Capital Measurement and Capital Standards: A Revised Framework. Bank for International Settlements, Basel, Switzerland
5. Bugera V, Konno H, Uryasev S (2002) Credit cards scoring with quadratic utility functions. J Multi-Criteria Decis Anal 11:197–211
6. Campbell C (2002) Kernel methods: A survey of current techniques. Neurocomput 48:63–84
7. Comptroller of the Currency Administrator of National Banks (2001) Rating Credit Risk: Comptrollers Handbook. Comptroller of the Currency Administrator of National Banks, Washington, DC
8. Doumpos M, Zopounidis C (2002) Multicriteria Decision Aid Classification Methods. Kluwer, Dordrecht
9. Friedman C (2002) CreditModel technical white paper, Techical Report. Standard and Poor's, New York
10. Hagan MT, Menhaj M (1994) Training feedforward networks with the Marquardt algorithm. IEEE Trans Neural Netw 5:989–993
11. Hayden E (2003) Are credit scoring models sensitive with respect to default definitions? Evidence from the Austrian market. EFMA 2003 Helsinki Meetings (Available at SSRN: http://ssrn.com/abstract=407709)
12. Huang Z, Chen H, Hsu CJ, Chen WH, Wu S (2004) Credit rating analysis with support vector machines and neural networks: A market comparative study. Decis Support Syst 37:543–558
13. Hung MS, Denton JW (1993) Training neural networks with the GRG2 nonlinear optimizer. Eur J Oper Res 69:83–91
14. Jorion P (2000) Value at Risk: The New Benchmark for Managing Financial Risk, 2nd edn. McGraw-Hill, New York
15. Krahnen JP, Weber M (2001) Generally accepted rating principles: A primer. J Banking Finance 25:3–23
16. Moller MF (1993) A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw 6:525–533
17. Mou TY, Zhou ZF, Shi Y (2006) Credit risk evaluation based on LINMAP. Lecture Notes Comput Sci 3994:452–459
18. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representation by error propagation. In: Rumelhart DE, Williams JL (eds) Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, Cambridge, pp 318–362
19. Ryu YU, Yue WT (2005) Firm bankruptcy prediction: Experimental comparison of isotonic separation and other classification approaches. IEEE Trans Syst, Man Cybern – Part A 35:727–737
20. Salcedo-Sanza S, Fernández-Villacañasa JL, Segovia-Vargas MJ, Bousoño-Calzón C (2005) Genetic programming for the prediction of insolvency in non-life insurance companies. Comput Oper Res 32:749–765
21. Schebesch KB, Stecking R (2005) Support vector machines for classifying and describing credit applicants: detecting typical and critical regions. J Oper Res Soc 56: 1082–1088

22. Schölkopf B, Smola A (2002) Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond. MIT Press, Cambridge
23. Tsakonas A, Dounias G, Doumpos M, Zopounidis C (2006) Bankruptcy prediction with neural logic networks by means of grammar-guided genetic programming. Expert Syst Appl 30:449–461
24. Vapnik VN (1998) Statistical Learning Theory. Wiley, New York
25. Varetto F (1998) Genetic algorithms applications in the analysis of insolvency risk. J Banking Finance 22:1421–1439
26. Zopounidis C, Doumpos M (2002) Multicriteria classification and sorting methods: A literature review. Eur J Oper Res 138:229–246

Criss-Cross Pivoting Rules

Tamás Terlaky
Department Comput. & Software,
McMaster University, Hamilton, Canada

MSC2000: 90C05, 90C33, 90C20, 05B35, 65K05

Article Outline

Keywords
Synonyms
Introduction
  Ziont's Criss-Cross Method
The Least-Index Criss-Cross Method
  Other Interpretations
  Recursive Interpretation
  Lexicographically Increasing List
  Other Finite Criss-Cross Methods
  First-in Last-out Rule (FILO)
  Most Often Selected Variable Rule
  Exponential and Average Behavior
  Best-Case Analysis of Admissible Pivot Methods
Generalizations
  Fractional Linear Optimization
  Linear Complementarity Problems
  Convex Quadratic Optimization
  Oriented Matroids
See also
References

Keywords

Pivot rules; Criss-cross method; Cycling; Recursion; Linear optimization; Oriented matroids

Synonyms

Criss-cross

Introduction

From the early days of linear optimization (LO) (or linear programming), many people have been looking for a *pivot algorithm* that avoids the *two-phase procedure* needed in the simplex method when solving the general LO problem in standard primal form

$$\min\left\{c^\top x\colon\ Ax = b,\ x \geq 0\right\},$$

and its dual

$$\max\left\{b^\top y\colon\ A^\top y \leq c\right\}.$$

Such a method was assumed to rely on the intrinsic symmetry behind the *primal and dual problems* (i. e. it hoped to be *selfdual*), and it should be able to start with any *basic solution*.

There were several attempts made to relax the feasibility requirement in the simplex method. It is important to mention Dantzig's [7] parametric selfdual simplex algorithm. This algorithm can be interpreted as *Lemke's algorithm* [22] for the corresponding linear complementarity problem (cf. ► Linear complementarity problem) [23]. In the 1960s people realized that pivot sequences through possibly infeasible basic solutions might result in significantly shorter paths to the optimum. Moreover a selfdual one phase procedure was expected to make linear programming more easily accessible for broader public. Probably these advantages stimulated the introduction of the *criss-cross method* by S. Ziont [39,40].

Ziont's Criss-Cross Method

Assuming that the reader is familiar with both the primal and dual simplex methods, Ziont's criss-cross method can easily be explained.

- It can be initialized by any, possibly both primal and dual infeasible *basis*.

  If the basis is optimal, we are done.

  If the basis is not *optimal*, then there are some primal or dual infeasible variables. One might choose any of these.

  It is advised to choose once a primal and then a dual infeasible variable, if possible.

- If the selected variable is dual infeasible, then it enters the basis and the leaving variable is chosen among the primal feasible variables in such a way that primal feasibility of the currently primal feasible variables is preserved.

  If no such basis exchange is possible another infeasible variable is selected.
- If the selected variable is primal infeasible, then it leaves the basis and the entering variable is chosen among the dual feasible variables in such a way that dual feasibility of the currently dual feasible variables is preserved.

  If no such basis exchange is possible another infeasible variable is selected.

If the current basis is infeasible, but none of the infeasible variables allows a pivot fulfilling the above requirements then it is proved that the LO problem has no optimal solution.

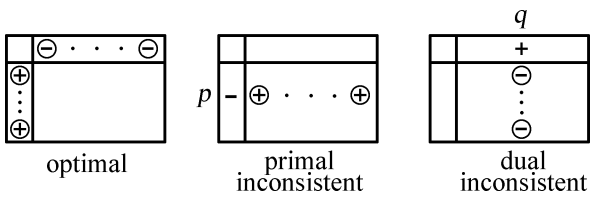
Once a primal or dual feasible solution is reached then Ziont's method reduces to the primal or dual simplex method, respectively.

One attractive character of Ziont's criss-cross method is primal-dual symmetry (selfduality), and this alone differentiates itself from the simplex method. However it is not clear how one can design a finite version (i. e. a finite pivot rule) of this method. Both lexicographic perturbation and minimal index resolution seem not to be sufficient to prove finiteness in the general case when the initial basis is both primal and dual infeasible. Nevertheless, although not finite, this is the first published criss-cross method in the literature.

The other thread, that lead to finite criss-cross methods, was the intellectual effort to find finite, other than the lexicographic rule [4,8], variants of the simplex method. These efforts were also stimulated by studying the combinatorial structures behind linear programming. From the early 1970s in several branches of the optimization theory, finitely convergent algorithms were published. In particular A.W. Tucker [32] introduced the *consistent labeling* technique in the Ford–Fulkerson maximal flow algorithm; pivot selection rules based on least-index ordering, such as the Bard-type scheme for the P-matrix linear complementarity problem (K.G. Murty, [24]) and the celebrated least-index rule in linear and oriented matroid programming (R.G. Bland, [2]). A thorough survey of pivot algorithms can be found in [29].

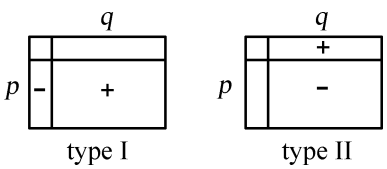
It is remarkable that almost at the same time, in different parts of the world (China, Hungary, USA) essentially the same result was obtained independently by approaching the problem from quite different directions.

Below we will refer to the standard simplex (basis) tableau. A tableau is called *terminal* if it gives a primal and dual optimal solution or evidence of primal or dual infeasibility/inconsistency of the problem. Terminal tableaus have the following sign structure.



Terminal tableaus

The pivot operations at all known pivot methods, including all variants of the primal and dual simplex method and Ziont's criss-cross method have the following properties. When a primal infeasible variable is selected to leave the basis, the entering variable is selected so that after the pivot both variables involved in the pivot will be primal feasible. Analogously, when a dual infeasible variable is selected to enter the basis, then the leaving variable is selected in such a way that after the pivot both variables involved in the pivot will be dual feasible. Such pivots are called *admissible*. The sign structure of tableaus at an admissible pivot of 'type I' and 'type II' are demonstrated by the following figure.



Admissible pivot situations

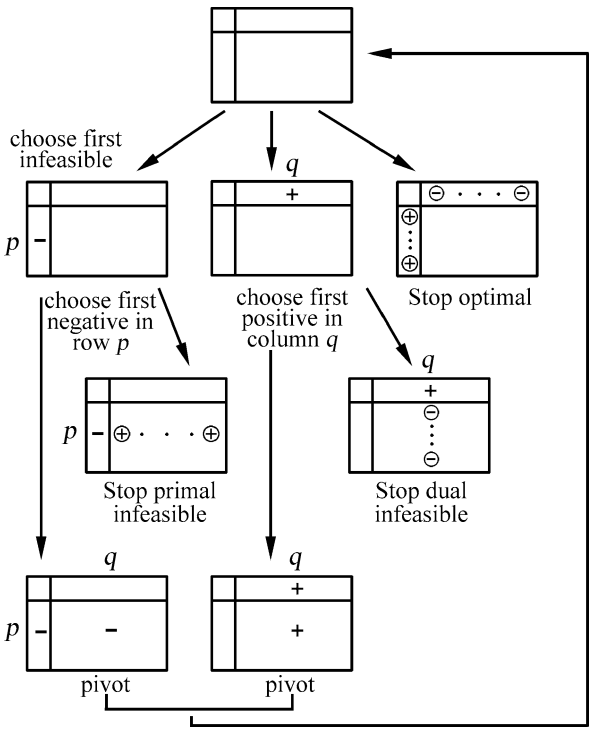
Observe that, while dual(primal) simplex pivots preserve dual(primal) feasibility of the basic solution, admissible pivots do not in general. Admissible pivots extract the greedy nature of pivot selection, i. e. 'repair primal/dual infeasibility' of the pivot variables.

The Least-Index Criss-Cross Method

The first finite criss-cross algorithm, which we call the *least-index criss-cross method*, was discovered independently by Y.Y. Chang [26], T. Terlaky [26,27,28] and Zh. Wang [34]; further, a strongly related general recursion by D. Jensen [18]. Chang presented the algorithm for positive semidefinite linear complementarity problems, Terlaky for linear optimization, for oriented matroids and with coauthors for QP, LCP and for oriented matroid LCP [9,16,19], while Wang primarily for the case of oriented matroids.

The least-index criss-cross method is perhaps the simplest finite pivoting method to LO problems. This criss-cross method is a purely combinatorial pivoting method, it uses admissible pivots and traverses through different (possibly both primal and dual infeasible) bases until the associated basic solution is optimal, or an evidence of primal or dual infeasibility is found.

To ease the understanding a figure is included that shows the scheme of the least-index criss-cross method.



Scheme of the least-index criss-cross method

Observe the simplicity of the algorithm:

- It can be initiated with any basis.
- No two phases are needed.
- No ratio test is used to preserve feasibility, only the signs of components in a basis tableau and a prefixed ordering of variables determine the pivot selection.

Several finiteness proofs for the least-index criss-cross method can be found in the literature. The proofs are quite elementary, they are based on the orthogonality of the primal and dual spaces [14,26,28,29,34]; on recursive argumentation [11,18,33] or on lexicographically increasing lists [11,14].

0	Let an ordering of the variables be fixed. Let $T(B)$ be an arbitrary basis tableau (it can be neither primal nor dual feasible);
1	Let r be the minimal i such that either x_i is primal infeasible or x_i has a negative reduced cost. IF there is no r, THEN stop; the first terminal tableau is obtained, thus $T(B)$ is optimal.
2	IF x_r is primal infeasible THEN let $p := r$; $q := \min\{\ell : t_{p\ell} < 0\}$. IF there is no q, THEN stop; the second terminal tableau is obtained, thus the primal problem is infeasible. Go to Step 3. IF x_r is dual infeasible, THEN let $q := r$: $p :=\min\{\ell : t_{lq} > 0\}$. IF there is no q, THEN stop: the third terminal tableau is obtained, thus the dual problem is infeasible. Go to Step 3.
3	Pivot on (p, q). Go to Step 1.

The least-index criss-cross rule

One of the most important consequences of the finiteness of the least-index criss-cross method is the *strong duality theorem* of linear optimization. This gives probably the simplest algorithmic proof of this fundamental result:

Theorem 1 (*Strong duality theorem*) *Exactly one of the following two cases occurs.*

- *At least one of the primal problem and the dual problem is infeasible.*
- *Both problems have an optimal solution and the optimal objective values are equal.*

Other Interpretations

The least-index criss-cross method can be interpreted as a *recursive algorithm*. This recursive interpretation and the finiteness proof based on it can be derived from the results in [2,3,18] and can be found in [33].

Recursive Interpretation

As performing the least-index criss-cross method at each pivot one can make a note of the larger of the two indices $r = \max\{p, q\}$ that entered or left the basis. In this list, an index must be followed by another larger one before the same index occurs anew.

The recursive interpretation is becoming apparent when one notes that it is based on the observation that the size of the solved subproblem (the subproblem for which a terminal tableau is obtained) is monotonically increasing.

The third interpretation is based on the proof technique developed by J. Edmonds and K. Fukuda [9] and adapted by Fukuda and T. Matsui [11] to the case of the least-index criss-cross method.

Lexicographically Increasing List

Let u be an binary vector with appropriate dimension, set initially to be the zero vector. In applying the algorithm let $r = \max\{p, q\}$ be the larger of the two indices p that entered or q that left the basis.

At each pivot update u as follows: let $u_r = 1$ and $u_i = 0$, $\forall i < r$. The remaining components of u stay unchanged. Then at each step of the least-index criss-cross method the vector u strictly increases lexicographically, thus the method terminates in a finite number of steps.

Other Finite Criss-Cross Methods

Both the recursive interpretation and the hidden flexibility of pivot selection in the least-index criss-cross method make it possible to develop other finite variants. Such finite criss-cross methods, which do not rely on a fixed minimal index ordering, were developed on the basis of the finite simplex rules presented by S. Zhang [37]. These finite criss-cross rules [38] are as follows.

First-in Last-out Rule (FILO)

First, choose a primal or dual infeasible variable that has changed its basis-nonbasis status most recently.

Then choose a variable in the selected row or column so that the pivot entry fulfills the sign requirement of the admissible pivot selection and which has changed its basis/nonbasis status most recently.

When more than one candidates occur with the same pivot age then one break tie as you like (e. g. randomly).

This rule can easily be realized by assigning an 'age' vector u to the vector of the variables and using a pivot counter k. Initially we set $k = 0$ and $u = 0$. Increase k by one at each pivot and we set the pivot coordinates of u equal to k. Then the pivot selections are made by choosing the variable with the highest possible u_i value satisfying the sign requirements.

Most Often Selected Variable Rule

First, choose a primal or dual infeasible variable that has changed its basis-nonbasis status most frequently.

Then choose a variable in the selected row or column so that the pivot entry fulfills the sign requirement of the admissible pivot selection and which has changed its basis/nonbasis status most frequently.

When more than one candidates occur with the same pivot age then one break tie as you like (e. g. randomly).

The most often selected rule can also be realized by assigning another 'age' vector u to the vector of the variables. Initially we set $u = 0$. At each pivot we increase the pivot-variable components of u by one. Then the pivot selections are made by choosing the variable with the highest possible u_i value satisfying the sign requirement.

Exponential and Average Behavior

The worst-case exponential behavior of the least-index criss-cross method was studied by C. Roos [25]. Roos'

exponential example is a variant of the cube of V. Klee and G.J. Minty [21]. In this example the starting solution is the origin defined by a feasible basis, the variables are ordered so that the least-index criss-cross method follows a simplex path, i. e. without making any ratio test feasibility of the starting basis is preserved. Another exponential example was presented by Fukuda and Namiki [12] for linear complementarity problems.

Contrary to the clear result on the worst-case behavior, to date not much is known about the expected or average number of pivot steps required by finite criss-cross methods.

Best-Case Analysis of Admissible Pivot Methods

As it was discussed above, and it is the case for many simplex algorithms, the least-index criss-cross method is not a polynomial time algorithm. A question naturally arises: whether there exists a polynomial criss-cross method? Unfortunately no answer to this question is available at this moment. However some weaker variants of this question can be answered positively. The problem is stated as follows: An arbitrary basis is given. What is the shortest admissible pivot path from this given basis to an optimal basis?

For *nondegenerate problems*, [10] shows the existence of such an admissible pivot sequence of length at most m. The nondegeneracy assumption is removed in [15]. This result solves a relaxation of the d-step conjecture.

Observe, that we do not know of any such result for feasibility preserving, i. e. simplex algorithms. In fact, the maximum length of feasibility-preserving pivot sequences between two feasible bases is not known to be bounded by a polynomial in the size of the given LO problem.

Generalizations

Finite criss-cross methods were generalized to solve fractional linear optimization problems, to large classes of linear complementarity problems (LCPs; cf. ► Linear complementarity problem) and to oriented matroid programming problems (OMPs).

Fractional Linear Optimization

Fractional linear or, as it is frequently referred to, *hyperbolic programming*, can be reformulated as a linear optimization problem. Thus it goes without surprise that the least-index criss-cross method is generalized to this class of optimization problems as well [17].

Linear Complementarity Problems

The largest solvable class of LCPs is the class of LCPs with a *sufficient matrix* [5,6]. The LCP least-index criss-cross method is a proper generalization of the LO criss-cross method. When the LCP arises from a LO problem, the LO criss-cross method is obtained.

Convex Quadratic Optimization

Convex quadratic optimization problems give an LCP with a bisymmetric coefficient matrix. Because a bisymmetric matrix is semidefinite and semidefinite matrices form a subclass of sufficient matrices, one obtain a finite criss-cross algorithm for convex quadratic optimization problems as well. Such criss-cross algorithms were published e. g. in [20]. The least-index criss-cross method is extremely simple for the *P-matrix* LCP. Starting from an arbitrary complementary basis, here the least-indexed infeasible variable leaves the basis and it is replaced by its complementary pair. This algorithm was originally proposed in [24], and studied in [12]. The general case of sufficient LCPs was treated in [4,13,16].

Oriented Matroids

The intense research in the 1970s on *oriented matroids* and oriented matroid programming [2,9] gave a new insight in pivot algorithms. It became clear that although the simplex method has rich combinatorial structures, some essential results like the finiteness of Bland's least-index simplex rule [2] does not hold in the oriented matroid context. Edmonds and Fukuda [9] showed that it might cycle in the oriented matroid case due to the possibility of nondegenerate cycling which is impossible in the linear case.

The predecessors of finite criss-cross rules are: Bland's recursive algorithm [2,3], the Edmonds–Fukuda algorithm [9], its variants and generalizations [1,35,36,37]. All these are variants of the simplex method in the linear case, i. e. they preserve the feasibility of the basis, but not in the oriented matroid case. In the case of oriented matroid programming only Todd's finite lexicographic method [30,31] preserves feasibility

of the basis and therefore yields a finite simplex algorithm for oriented matroids.

The least-index criss-cross method is a finite criss-cross method for oriented matroids [28,34]. A general recursive scheme of finite criss-cross methods is given in [18]. Finite criss-cross rules are also presented for oriented matroid quadratic programming and for oriented matroid linear complementarity problems [13,19].

See also

- ► Least-Index Anticycling Rules
- ► Lexicographic Pivoting Rules
- ► Linear Programming
- ► Pivoting Algorithms for Linear Programming Generating Two Paths
- ► Principal Pivoting Methods for Linear Complementarity Problems
- ► Probabilistic Analysis of Simplex Algorithms
- ► Simplicial Pivoting Algorithms for Integer Programming

References

1. Bjorner A, Vergnas M LAS, Sturmfels B, White N, Ziegler G (1993) Oriented matroids. Cambridge Univ Press, Cambridge
2. Bland RG (1977) A combinatorial abstraction of linear programming. J Combin Th B 23:33–57
3. Bland RG (1977) New finite pivoting rules for the simplex method. Math Oper Res 2:103–107
4. Chang YY (1979) Least index resolution of degeneracy in linear complementarity problems. Techn Report Dept Oper Res Stanford Univ 14
5. Cottle R, Pang JS, Stone RE (1992) The linear complementarity problem. Acad Press, New York
6. Cottle RW, Pang J-S, Venkateswaran V (1987) Sufficient matrices and the linear complementarity problem. Linear Alg Appl 114/115:235–249
7. Dantzig GB (1963) Linear programming and extensions. Princeton Univ Press, Princeton
8. Dantzig GB, Orden A, Wolfe P (1955) Notes on linear programming: Part I – The generalized simplex method for minimizing a linear form under linear inequality restrictions. Pacific J Math 5(2):183–195
9. Fukuda K (1982) Oriented matroid programming. PhD Thesis Waterloo Univ
10. Fukuda K, Luethi H-J, Namiki M (1997) The existence of a short sequence of admissible pivots to an optimal basis in LP and LCP. ITOR 4:273–284
11. Fukuda K, Matsui T (1991) On the finiteness of the criss-cross method. Europ J Oper Res 52:119–124
12. Fukuda K, Namiki M (1994) On extremal behaviors of Murty's least index method. Math Program 64:365–370
13. Fukuda K, Terlaky T (1992) Linear complementarity and oriented matroids. J Oper Res Soc Japan 35:45–61
14. Fukuda K, Terlaky T (1997) Criss-cross methods: A fresh view on pivot algorithms. Math Program (B) In: Lectures on Math Program, vol 79. ISMP97, Lausanne, pp 369–396
15. Fukuda K, Terlaky T (1999) On the existence of short admissible pivot sequences for feasibility and linear optimization problems. Techn Report Swiss Federal Inst Technol
16. Hertog D Den, Roos C, Terlaky T (1993) The linear complementarity problem, sufficient matrices and the criss-cross method. Linear Alg Appl 187:1–14
17. Illés T, Szirmai Á, Terlaky T (1999) A finite criss-cross method for hyperbolic programming. Europ J Oper Res 114:198–214
18. Jensen D (1985) Coloring and duality: Combinatorial augmentation methods. PhD Thesis School OR and IE, Cornell Univ
19. Klafszky E, Terlaky T (1989) Some generalizations of the criss-cross method for the linear complementarity problem of oriented matroids. Combinatorica 9:189–198
20. Klafszky E, Terlaky T (1992) Some generalizations of the criss-cross method for quadratic programming. Math Oper Statist Ser Optim 24:127–139
21. Klee V, Minty GJ (1972) How good is the simplex algorithm? In: Shisha O (ed) Inequalities-III. Acad Press, New York, pp 1159–175
22. Lemke CE (1968) On complementary pivot theory. In: Dantzig GB, Veinott AF (eds) Mathematics of the Decision Sci Part I. Lect Appl Math 11. Amer Math Soc. Providence, RI, pp 95–114
23. Lustig I (1987) The equivalence of Dantzig's self-dual parametric algorithm for linear programs to Lemke's algorithm for linear complementarity problems applied to linear programming. SOL Techn Report Dept Oper Res Stanford Univ 87(4)
24. Murty KG (1974) A note on a Bard type scheme for solving the complementarity problem. Opsearch 11(2–3): 123–130
25. Roos C (1990) An exponential example for Terlaky's pivoting rule for the criss-cross simplex method. Math Program 46:78–94
26. Terlaky T (1984) Egy új, véges criss-cross módszer lineáris programozási feladatok megoldására. Alkalmazott Mat Lapok 10:289–296 English title: A new, finite criss-cross method for solving linear programming problems. (In Hungarian)
27. Terlaky T (1985) A convergent criss-cross method. Math Oper Statist Ser Optim 16(5):683–690
28. Terlaky T (1987) A finite criss-cross method for oriented matroids. J Combin Th B 42(3):319–327

29. Terlaky T, Zhang S (1993) Pivot rules for linear programming: A survey on recent theoretical developments. Ann Oper Res 46:203–233
30. Todd MJ (1984) Complementarity in oriented matroids. SIAM J Alg Discrete Meth 5:467–485
31. Todd MJ (1985) Linear and quadratic programming in oriented matroids. J Combin Th B 39:105–133
32. Tucker A (1977) A note on convergence of the Ford–Fulkerson flow algorithm. Math Oper Res 2(2):143–144
33. Valiaho H (1992) A new proof of the finiteness of the criss-cross method. Math Oper Statist Ser Optim 25:391–400
34. Wang Zh (1985) A conformal elimination free algorithm for oriented matroid programming. Chinese Ann Math 8(B1)
35. Wang Zh (1991) A modified version of the Edmonds–Fukuda algorithm for LP in the general form. Asia–Pacific J Oper Res 8(1)
36. Wang Zh (1992) A general deterministic pivot method for oriented matroid programming. Chinese Ann Math B 13(2)
37. Zhang S (1991) On anti-cycling pivoting rules for the simplex method. Oper Res Lett 10:189–192
38. Zhang S (1999) New variants of finite criss-cross pivot algorithms for linear programming. Europ J Oper Res 116: 607–614
39. Zionts S (1969) The criss-cross method for solving linear programming problems. Managem Sci 15(7):426–445
40. Zionts S (1972) Some empirical test of the criss-cross method. Managem Sci 19:406–410

Cutting Plane Methods for Global Optimization

HOANG TUY
Institute Math., Hanoi, Vietnam

MSC2000: 90C26

Article Outline

Keywords
Outer Approximation
Inner Approximation
Concavity Cut
Nonlinear Cuts
References

Keywords

Cutting plane method; Outer approximation; Inner approximation; Polyhedral annexation; Concavity cut; Intersection cut; Convexity cut; Nonlinear cut; Polyblock approximation; Monotonic optimization

In solving global and combinatorial optimization problems cuts are used as a device to discard portions of the feasible set where it is known that no optimal solution can be found. Specifically, given the optimization problem

$$\min\{f(x):\ x \in D \subset \mathbb{R}^n\}, \tag{1}$$

if x^0 is an unfit solution and there exists a function $l(x)$ satisfying $l(x^0) > 0$, while $l(x) \le 0$ for every optimal solution x, then by adding the inequality $l(x) \le 0$ to the constraint set we exclude x^0 without excluding any optimal solution. The inequality $l(x) \le 0$ is called a *valid cut*, or briefly, a *cut*. Most often the function $l(x)$ is affine: the cut is then said to be linear, and the hyperplane $l(x) = 0$ is called a cutting plane. However, nonlinear cuts have proved to be useful, too, for a wide class of problems.

Cuts may be employed in different contexts: outer and inner approximation (conjunctive cuts), branch and bound (disjunctive cuts), or in combined form.

Outer Approximation

Let $\Omega \subset \mathbf{R}^n$ be the set of optimal solutions of problem (2). Suppose there exists a family $\mathcal{P}$ of polytopes $P \supset \Omega$ such that for each $P \in \mathcal{P}$ a *distinguished point* $z(P) \in P$ (conceived of as some approximate solution) can be defined satisfying the following conditions:

A1) $z(P)$ always exists (unless $\Omega = \emptyset$) and can be computed by an efficient procedure;

A2) given any $P \in \mathcal{P}$ and the associated distinguished point $z = z(P)$, we can recognize when $z \in \Omega$ and if $z \notin \Omega$, we can construct an affine function $l(x)$ such that $P' = P \cap \{x: l(x) \le 0\} \in \mathcal{P}$, and $l(z) > 0$, while $l(x) \le 0$, $\forall\, x \in \Omega$, i. e. $\Omega \subset P' \subset P \setminus \{z\}$.

Under these conditions, one can attempt to solve problem (2) by the following *outer approximation method (OA method)* [8]:

Prototype OA (outer approximation) procedure

0	Start with an initial polytope $P_1 \in \mathcal{P}$. Set $k = 1$.
1	Compute the distinguished point $z^k = z(P_k)$ (by A1)). If $z(P_k)$ does not exist, terminate: the problem is infeasible. If $z(P_k) \in \Omega$, terminate. Otherwise, continue.
2	Using A2), construct an affine function $l_k(x)$ such that $P_{k+1} = P_k \cap \{x: l_k(x) \le 0\} \in \mathcal{P}$ and $l_k(x)$ strictly separates z^k from Ω, i. e. satisfies $l_k(z^k) > 0, \quad l_k(x) \le 0\ \forall x \in \Omega.$ (2) Set $k \leftarrow k + 1$ and return to Step 1.

The algorithm is said to be *convergent* if it is either finite or generates an infinite sequence $\{z^k\}$ every cluster point of which is an optimal solution of problem (2).

Usually the distinguished point z^k is defined as a vertex of the polytope P_k satisfying some criterion (e. g., minimizing a given concave function). In these cases, the implementation of the above algorithm requires a procedure for computing, at iteration k, the vertex set V_k of the current polytope P_k. At the beginning, V_1 is supposed to be known, while P_{k+1} is obtained from P_k simply by adding one more linear constraint $l_k(x) \leq 0$. Using this information V_{k+1} can be derived from V_k by an on-line vertex enumeration procedure [1].

Example 1 (Concave minimization.) Consider the problem (1) where $f(x)$ is concave and D is a convex compact set with int $D \neq \emptyset$.

Assume that D is defined by a convex inequality $g(x) \leq 0$ and let $w \in$ int D. Take $\mathcal{P}$ to be the collection of all polytopes containing D. For every $P \in \mathcal{P}$ define $z := z(P)$ to be a minimizer of $f(x)$ over the vertex set V of P (hence, by concavity of $f(x)$, a minimizer of $f(x)$ over P). Clearly, if $z \in D$, it solves the problem. Otherwise, the line segment joining z to w meets the boundary of D at a unique point y and the affine function $l(x) = \langle p, x - y\rangle + g(y)$ with $p \in \partial\, g(y)$ strictly separates D from z (indeed, $l(z) = g(z) > 0$ while $l(x) \leq g(x) - g(z) + g(z) \leq 0$ for all $x \in D$. Obviously $P' = P \cap \{x : l(x) \leq 0\} \in \mathcal{P}$, so Assumptions A1) and A2) are fulfilled and the OA algorithm can be applied. The convergence of the algorithm is easy to establish.

Example 2 (Reverse convex programming.) Consider the problem (1) where $f(x) = \langle c, x\rangle$, while $D = \{x \in \mathbf{R}^n : h(x) \leq 0 \leq g(x)\}$ with $g(x)$, $h(x)$ continuous convex functions. Assume that the problem is *stable*, i. e. that $D = \mathrm{cl}(\mathrm{int}\, D)$, so a feasible solution $\overline{x} \in D$ is optimal if and only if

$$\{x \in D\colon\ \langle c, x - \overline{x}\rangle \leq 0\} \subset \{x\colon\ g(x) \leq 0\}\,. \qquad (3)$$

Also for simplicity assume a point w is available satisfying $\max\{h(w), g(w)\} < 0$ and $\langle c, w\rangle < \min\{\langle c, x\rangle : h(x) \leq 0 \leq g(x)\}$ (the latter assumption amounts to assuming that the constraint $g(x) \geq 0$ is essential).

Let Ω be the set of optimal solutions, $\mathcal{P}$ the collection of all polytopes containing Ω. For every $P \in \mathcal{P}$ let $z = z(P)$ be a maximizer of $g(x)$ over the vertex set V of the polyhedron $P \cap \{x : \langle c, x\rangle \leq \gamma\}$, where γ is the value of the objective function at the best feasible solution currently available (set $\gamma = +\infty$ if no feasible solution is known yet). By (3), if $g(z) \leq 0$, then γ is the optimal value (for $\gamma < +\infty$), or the problem is infeasible (for $\gamma = +\infty$). Otherwise, $g(z) > 0$, and we can construct an affine function $l(x)$ strictly separating z from Ω as follows. Since $\max\{h(w), g(w)\} < 0$ while $\max\{h(z), g(z)\} > 0$ the line segment joining z, w meets the surface $\max\{h(x), g(x)\} = 0$ at a unique point y.

1) If $g(y) = 0$ (while $h(y) \leq 0$), then y is a feasible solution and since $y = \lambda\, w + (1 - \lambda)\, z$ for some $\lambda \in (0, 1)$ we must have $\langle c, y\rangle = \lambda\, \langle c, w\rangle + (1 - \lambda)\langle c, z\rangle < \gamma$, so the cut $l(x) = \langle c, x - y\rangle \leq 0$ strictly separates z from Ω.
2) If $h(y) = 0$, then the cut $l(x) = \langle p, x - y + h(y) \leq 0$, where $p \in \partial\, h(y)$, strictly separates z from Ω (indeed, $l(x) \leq h(x) - h(y) + h(y) = h(x) \leq 0$ for all $x \in \Omega$ while $l(z) > 0$ because $l(w) < 0$, $l(y) = 0$).

Thus assumptions A1), A2) are satisfied, and again the OA algorithm can be applied. The convergence of the OA algorithm for this problem is established by a more elaborate argument than for the concave minimization problem (see [3,8]).

Various variants of OA method have been developed for a wide class of optimization problems, since any optimization problem described by means of differences of convex functions can be reduced to a reverse convex program of the above form [3]. However, a difficulty with this method when solving large scale problems is that the size of the vertex set V_k of P_k may grow exponentially with k, creating serious storage problems and making the computation of V_k almost impracticable.

Inner Approximation

Consider the concave minimization problem under linear constraints, i. e. the problem (2) when $f(x)$ is a concave function and D is a polytope in $\mathbf{R}^n$.

Without loss of generality we may assume that 0 is a vertex of D. For any real number $\gamma \leq f(0)$, the set $C_\gamma = \{x \in \mathbf{R}^n\}\{f(x) \geq \gamma\}$ is convex and $0 \in D \cap C_\gamma$. Of course, $D \subset C_\gamma$ if and only if $f(x) \geq \gamma$ for all $x \in D$.

The idea of the *inner approximation* method (IA method), also called the *polyhedral annexation* method (or PA method)[3], is to construct a sequence of expanding polytopes $P_1 \subset P_2 \subset \cdots$ together with a nonin-

creasing sequence of real numbers $\gamma_1 \geq \gamma_2 \geq \cdots$, such that $\gamma_k \in f(D)$, $P_k \subset C_{\gamma k}$, $k = 1, 2, \ldots$, and eventually $D \subset P_h$ for some h: then $\gamma_h \leq f(x)$ for all $x \in D$, i. e. γ_h will be the optimal value.

For every set $P \subset \mathbf{R}^n$ let P° be the *polar* of P, i. e. $P^\circ = \{y \in \mathbf{R}^n : \langle y, x\rangle \leq 1, \forall\, x \in P\}$. As is well known P° is a closed convex set containing 0 (in fact a polyhedron if P is a polyhedron), and $P \subset Q$ only if $P^\circ \supset Q^\circ$; moreover, if C is a closed convex set containing 0, then $(C^\circ)^\circ = C$. Therefore, setting $S_k = (P^k)^\circ$, the IA method amounts to constructing a sequence of nested polyhedra $S_1 \supset \cdots \supset S_h$ satisfying $S_k^\circ \subset C_{\gamma k}$, $k = 1, \ldots, h$ and $S_h \subset D^\circ$. The key point in this scheme is: Given $\gamma_k \in f(D)$ and a polyhedron S_k such that $S_k^\circ \subset C_{\gamma k}$, check whether $S_k \subset D^\circ$ and if there is $y^k \in S_k \backslash D^\circ$, then construct a cut $l_k(y) \leq 1$ to exclude y^k and to form a smaller polyhedron S_{k+1} such that $S_{k+1}^\circ \subset C_{\gamma k+1}$ for some $\gamma_{k+1} \in f(D)$ satisfying $\gamma_{k+1} \leq \gamma_k$.

To deal with this point, define $s(y) = \max\{\langle y, x\rangle : x \in D\}$. Since $y \in D^\circ$ whenever $s(y) \leq 1$ we will have $S_k \subset D^\circ$ whenever

$$\max\{s(y) :\ y \in S_k\} \leq 1. \tag{4}$$

But clearly the function $s(y)$ is convex as the pointwise maximum of a family of linear functions. Therefore, denoting the vertex set and the extreme direction set of S_k by V_k, U_k, respectively, we will have (4) (i. e. $S_k \subset D^\circ$) whenever

$$\begin{cases} \max\{s(y) :\ y \in V_k\} \leq 1, \\ \max\{s(y) :\ y \in U_k\} \leq 0. \end{cases} \tag{5}$$

Thus, checking the inclusion $S_k \subset D^\circ$ amounts to checking (5), a condition that fails to hold in either of the following cases:

$$s(y^k) > 1 \quad \text{for some } y^k \in V_k \tag{6}$$

$$s(y^k) > 0 \quad \text{for some } y^k \in U_k. \tag{7}$$

In each case, it can be verified that if x^k maximizes $\langle y^k, x\rangle$ over D, and $\gamma_{k+1} = \min\{\gamma_k, f(x^k)\}$ while

$$\theta_k = \sup\left\{\theta :\ f(\theta x^k) \geq \gamma_{k+1}\right\},$$

then $S_{k+1} = S_k \cap \{y : \langle x^k, y\rangle \leq 1/\theta_k\}$ satisfies

$$P_{k+1} := S_{k+1}^\circ = \operatorname{conv}(P_k \cup \{\theta_k x^k\}) \subset C_{\gamma_{k+1}}.$$

In the case (6), S_{k+1} no longer contains y^k while in the case (7), y^k is no longer an extreme direction of S_{k+1}. In this sense, the cut $\langle x^k, y \leq 1/\theta_k$ excludes y^k. We can thus state the following algorithm.

IA Algorithm (for concave minimization)

0	By translating if necessary, make sure that 0 is a vertex of D. Let $\bar{x}^1$ be the best basic feasible solution available, $\gamma_1 = f(\bar{x}^1)$. Take a simplex $P_1 \subset C_{\gamma_1}$ and let $S_1 = P_1^\circ$, $V_1 =$ vertex set of S_1, $U_1 =$ extreme direction set of S_1. Set $k = 1$.
1	Compute $s(y)$ for every new $y \in (V_k \cup U_k)\backslash\{0\}$. If (5) holds, then terminate: $S_k \subset D^\circ$ so $\bar{x}^k$ is a global optimal solution.
2	If (6) or (7) holds, then let $x^k \in \arg\max\{\langle y^k, x\rangle :\ x \in D\}$. Update the current best feasible solution by comparing x^k and $\bar{x}^k$. Set $\gamma_{k+1} = f(\bar{x}^{k+1})$.
3	Compute $\theta_k = \max\{\theta \geq 1 : f(\theta x^k) \geq \gamma_{k+1}\}$ and let $S_{k+1} = S_k \cap \left\{y :\ \langle x^k, y\rangle \leq \frac{1}{\theta_k}\right\}$. From V_k and U_k derive the vertex set V_{k+1} and the extreme direction set U_{k+1} of S_{k+1}. Set $k \leftarrow k+1$ and go to Step 1.

It can be shown that the IA algorithm is finite [3]. Though this algorithm can be interpreted as dual to the OA algorithm, its advantage over the OA method is that it can be started at any vertex of D, so that each time the set V_k has reached a certain critical size, it can be stopped and 'restarted' at a new vertex of D, using the last obtained best value of $f(x)$ as the initial γ_1. In that way the set V_k can be kept within manageable size.

Note that if D is contained in a cone M and $P_1 = \{x \in M : \langle v^1, x\rangle \leq 1\} \subset C_{\gamma 1}$, then it can be shown that (7) automatically holds, and only (6) must be checked [6].

Concavity Cut

The cuts mentioned above are used to separate an unfit solution from some convex set containing at least one optimal solution. They were first introduced in convex programming [2,4]. Another type of cuts originally devised for concave minimization [7] is the following.

Suppose that a feasible solution $\bar{x}$ has already been known with $f(\bar{x}) = \gamma$ and we would like to check whether there exists a better feasible solution. One way to do that is to take a vertex x^0 of D with $f(x^0) > \gamma$ and to construct a cone M, as small as possible, vertexed at x^0, containing D and having exactly n edges. Since x^0 is interior to the convex set $C_\gamma = \{x : f(x) \geq \gamma\}$, each

ith edge of M, for $i = 1, \ldots, n$, meets the boundary of C_γ at a uniquely defined point y^i (assuming that C_γ is bounded). Through these n points $y^1, \ldots, y^n$ (which are affinely independent) one can draw a unique hyperplane, of equation $\pi\ (x - x^0) = 1$ such that $\pi\ (y^i - x^0) = 1\ (i = 1, \ldots, n)$, hence $\pi = e^\top\ U^{-1}$, where U is the matrix of columns $y^1 - x^0, \ldots, y^n - x^0$ and e denotes a vector of n ones. Since the linear inequality

$$e^\top U^{-1}(x - x^0) \geq 1 \tag{8}$$

excludes x^0 without excluding any feasible solution x better than $\overline{x}$, this inequality defines a valid cut. In particular, if it so happens that the whole polytope D is cut off, i. e. if

$$D \subset \left\{x:\ e^\top U^{-1}(x - x^0) \leq 1\right\}, \tag{9}$$

then $\overline{x}$ is a global optimal solution.

This cut is often referred to as a γ-valid *concavity cut* for (f, D) at x^0 [3]. Its construction requires the availability of a cone $M \supset S$ vertexed at x^0 and having exactly n edges. In particular, if the vertex x^0 of D has exactly n neighboring vertices then M can be taken to be the cone generated by the n halflines from x^0 through each of these neighbors of x^0. Note, however, that the definition of the concavity cut can be extended so that its construction is possible even when the cone M has more than n edges (as e. g., when x^0 is a degenerated vertex of D).

Condition (9), sufficient for optimality, suggests a cutting method for solving the linearly constrained concave minimization problem by using concavity cuts to iteratively reduce the feasible polyhedron. Unfortunately, experience has shown that concavity cuts, when applied repeatedly, tend to become shallower and shallower. Though these cuts can be significantly strengthened by exploiting additional structure of the problem (e. g., in concave quadratic minimization, bilinear programming [5] and also in low rank nonconvex problems [6]), pure cutting methods are often outperformed by *branch and cut* methods where cutting is combined with successive partition of the space [8].

Concavity cuts have also been used in combinatorial optimization ('intersection cuts', or in a slightly extended form, 'convexity cuts').

Nonlinear Cuts

In many problems, *nonlinear cuts* arise in a quite natural way.

For example, consider the following problem of *monotonic optimization* [10]:

$$\max\left\{f(x):\ g(x) \leq 1,\ h(x) \geq 1,\ x \in \mathbb{R}^n_+\right\}, \tag{10}$$

where f, g, h are continuous increasing functions on $\mathbf{R}^n_+$ (a function $f(x)$ is said to be *increasing* on $\mathbf{R}^n_+$ if $0 \leq x \leq x' \Rightarrow f(x) \leq f(x')$; the notation $x \leq x'$ means $x_i \leq x'_i$ for all i while $x < x'$ means $x_i < x'_i$ for all i). As argued in [10], a very broad class of optimization problems can be cast in the form (10). Define $G = \{x \in \mathbf{R}^n_+ g(x) \leq 1\}$, $H = \{x \in \mathbf{R}^n_+ : h(x) \geq 1\}$, so that the problem is to maximize $f(x)$ over the feasible set $G \cap H$. Clearly

$$0 \leq x \leq x' \in G \quad \Rightarrow \quad x \in G, \tag{11}$$

$$0 \leq x \leq x' \notin H \quad \Rightarrow \quad x \notin H. \tag{12}$$

Assume that $g(0) < 1$ and $0 < a \leq x \leq b$ for all $x \in G \cap H$ (so $0 \in$ int G, $b \in H$). From (11) it follows that if $z \in \mathbf{R}^n_+ \setminus G$ and $\pi(z)$ is the last point of G on the halfline from 0 through z, then the cone $K\pi(z)\} = \{x \in \mathbf{R}^n_+ : x > \pi(z)\}$ separates z from G, i. e. $G \cap K_\pi(z) = \emptyset$, while $z \in K_\pi(z)$.

A set of the form $P = \cup_{y \in V}\{x: 0 \leq y\}$, where V is a finite subset of $\mathbf{R}^n_+$, is called a *polyblock* of vertex set V [9]. A vertex v is said to be *improper* if $v \leq v'$ for some $v' \in V \setminus \{v\}$. Of course, improper vertices can be dropped without changing P. Also if $P \supset G \cap H$ then the polyblock of vertex set $V' = V \cap H$ still contains $G \cap H$ because $v \notin H$ implies that $[0, v] \cap H = \emptyset$. With these properties in mind we can now describe the *polyblock approximation* procedure for solving (10).

Start with the polyblock $P_1 = [0, b] \supset G \cap H$ and its vertex set $V_1 = \{b\} \subset H$. At iteration k we have a polyblock $P_k \supset G \cap H$ with vertex set $V_k \subset H$. Let $y^k \in \arg\max\{f(x) : x \in V_k\}$. Clearly y^k maximizes $f(x)$ over P_k, and $y^k \in H$, so if $y^k \in G$ then y^k is an optimal solution. If $y^k \notin G$ then the point $x^k = \pi\ (y^k)$ determines a cone K_{x^k} such that the set $P_{k+1} = P_k \setminus K_{x^k}$ excludes y^k but still contains $G \cap H$. It turns out that P_{k+1} is a polyblock whose vertex set V_{k+1} is obtained from V_k by adding n points $v^{k,1}, \ldots, v^{k,n}$ (which are the n vertices of the hyperrectangle $[x^k, y^k]$ adjacent to y^k) and then dropping

all those which do not belong to H. With this polyblock P_{k+1}, we pass to iteration k+1.

In that way we generate a nested sequence of polyblocks $P_1 \supset P_2 \supset \cdots \supset G \cap H$. It can be proved that either y^k is an optimal solution at some iteration k or $f(y^k) \searrow \gamma := \max\{f(x) : x \in G \cap H\}$.

A similar method can be developed for solving the problem

$$\min\left\{f(x):\ g(x) \leq 1, h(x) \geq 1, x \in \mathbb{R}^n_+\right\}$$

by interchanging the roles of g, h and a, b. In contrast with what happens in OA methods, the vertex set V_k of the polyblock P_k in the polyblock approximation algorithm is extremely easy to determine. Furthermore this method admits restarts, which provide a way to prevent stall and overcome storage difficulties when solving large scale problems [10].

References

1. Chen P, Hansen P, Jaumard B (1991) On-line and off-line vertex enumeration by adjacent lists. Oper Res Lett 10:403–409
2. Cheney EW, Goldstein AA (1959) Newton's method for convex programming and Tchebycheff approximation. Numerische Math 1:253–268
3. Horst R, Tuy H (1996) Global optimization: deterministic approaches, 3rd edn. Springer, Berlin
4. Kelley JE (1960) The cutting plane method for solving convex programs. J SIAM 8:703–712
5. Konno H (1976) A cutting plane algorithm for solving bilinear programs. Math Program 11:14–27
6. Konno H, Thach PT, Tuy H (1997) Optimization on low rank nonconvex structures. Kluwer, Dordrecht
7. Tuy H (1964) Concave programming under linear constraints. Soviet Math 5:1437–1440
8. Tuy H (1998) Convex analysis and global optimization. Kluwer, Dordrecht
9. Tuy H (1999) Normal sets, polyblocks and monotonic optimization. Vietnam J Math 27(4):277–300
10. Tuy H (2000) Monotonic optimization: Problems and solution approaches. SIAM J Optim 11(2):464–494

Cutting-Stock Problem

András Prékopa[1], Csaba I. Fábián[2]

[1] RUTCOR, Rutgers Center for Operations Research, Piscataway, USA

[2] Eötvös Loránd University, Budapest, Hungary

MSC2000: 90B90, 90C59

Article Outline

Keywords
See also
References

Keywords

Cutting-stock problem; Cutting pattern; Column generation; Knapsack problem

A company that produces large rolls of paper, textile, steel, etc., usually faces the problem of how to cut the large rolls into smaller rolls, called finished rolls, in such a way that the demands for all finished rolls be satisfied. Any large roll is cut according to some cutting pattern and the problem is to find the cutting patterns to be used and to how many large rolls they should be applied. We assume, for the sake of simplicity, that each large roll has width W, an integer multiple of some unit and the finished roll widths are also specified by some integers $w_1, \ldots, w_m$. Let a_{ij} designate the number of rolls of width w_i produced by the use of the jth pattern, $i = 1, \ldots, m$, $j = 1, \ldots, n$. Let further b_i designate the demand for roll i, $i = 1, \ldots, m$, and $c_j = 1$, $j = 1, \ldots, n$. If $A = (a_{ij})$, $\mathbf{b} = (b_1, \ldots, b_m)^\mathsf{T}$, $\mathbf{c} = (c_1, \ldots, c_n)^\mathsf{T}$, then the problem is:

$$\begin{cases} \min & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{cases}$$

Here x_j means the number of jth cutting patterns to be used and as such, an all integer solution would be required to the problem. However, one is usually satisfied with an optimal solution of the above problem without the integrality restriction and, having that, a simple round-off procedure provides us with the solution to the problem.

In the above problem, however, the matrix A is huge, therefore we do not, and in most cases cannot, create it, by enumerating the cutting patterns. P.C. Gilmore and R.E. Gomory [3,4] resolved this difficulty by an ingenious column generation technique. It works in such a way that we generate column j, in the course of the simplex algorithm, whenever needed. Assume

that B is the current basis and designate by $\boldsymbol{\pi}$ the corresponding dual vector, i. e., the solution of the linear equation: $\boldsymbol{\pi}^{\mathsf{T}} B = \mathbf{c}_B^{\top}$. Now, if $\mathbf{a} = (a_1, \ldots, a_m)^{\mathsf{T}} \in \mathbf{Z}_+^m$ satisfies the inequality $\mathbf{w}^{\mathsf{T}}\mathbf{a} \leq W$, then, by definition, $\mathbf{a}$ represents a cutting pattern, a column of the matrix A. Since the cutting-stock problem is a minimization problem, the basis B is optimal if $\boldsymbol{\pi}^{\mathsf{T}}\mathbf{a} \leq 1$ for any $\mathbf{a}$ that satisfies $\mathbf{w}^{\mathsf{T}}\mathbf{a} \leq W$. We can check it by solving the linear program:

$$\begin{cases} \min & \boldsymbol{\pi}^{\top}\mathbf{a} \\ \text{s.t.} & \mathbf{w}^{\top}\mathbf{a} \leq W \\ & \mathbf{a} \in \mathbb{Z}_+^m. \end{cases}$$

If the optimum value is greater than 1, then the optimal $\mathbf{a}$ vector may enter the basis, otherwise B is an optimal basis and $\mathbf{x}_B$ is an optimal solution to the problem. The problem to find the vector $\mathbf{a}$ is a knapsack problem for which efficient solution methods exist.

In practice, however, frequently more complicated cutting-stock problems come up, due to special customer requirements depending on quality and other characteristics. In addition, we frequently need to include set up costs, capacity constraints and costs due to delay in manufacturing. These lead to the development of special algorithms as described in [1,4,5,6,7]. Recently Cs.I. Fábián [2] formulated stochastic variants of the cutting-stock problem, for use in fiber manufacturing.

See also

► Integer Programming

References

1. Dyckhoff H, Kruse HJ, Abel D, Gal T (1985) Trim loss and related problems. OMEGA Internat J Management Sci 13: 59–72
2. Fábián CsI (1998) Stochastic programming model for optical fiber manufacturing. RUTCOR Res Report 34–98
3. Gilmore PC, Gomory RE (1961) A linear programming approach to the cutting stock problem. Oper Res 9:849–859
4. Gilmore PC, Gomory RE (1963) A linear programming approach to the cutting stock problem, Part II. Oper Res 11:863–888
5. Gilmore PC, Gomory RE (1965) Multistage cutting stock problems of two and more dimensions. Oper Res 13:94–120
6. Johnson MP, Rennick C, Zak E (1998) Skiving addition to the cutting stock problem in the paper industry. SIAM Rev 39(3):472–483
7. Nickels W (1988) A knowledge-based system for integrated solving cutting stock problems and production control in the paper industry. In: Mitra G (ed) Mathematical Models for Decision Support. Springer, Berlin

Cyclic Coordinate Method

CCM

Vassilios S. Vassiliadis, Raúl Conejeros
Chemical Engineering Department,
University Cambridge, Cambridge, UK

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Cyclic coordinate search; Line search methods; Pattern search; Aitken double sweep method; Gauss–Southwell method; Nondifferentiable optimization

Often the solution of multivariable optimization problems it is desired to be done with a *gradient-free algorithm*. This may be the case when gradient evaluations are difficult, or in fact gradients of the underlying optimization method do not exist. Such a method that offers this feature is the method of the cyclic coordinate search and its variants.

The minimization problem considered is:

$$\min_{\mathbf{x}} f(\mathbf{x}).$$

The method in its basic form uses the coordinate axes as the search directions. In particular, the search directions $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n)}$, where the $\mathbf{d}^{(i)}$ are vectors of zeros, except for a 1 in the ith position. Therefore along each search direction $\mathbf{d}^{(i)}$ the corresponding variable x_i is changed only, with all remaining variables being kept constant to their previous values.

It is assumed here that the minimization is carried out in order over all variables with indices $1, \ldots, n$ at each iteration of the algorithm. However there are

variants. The first of these is the *Aitken double sweep method*, which processes first the variables in the order mentioned above, and then in the second sweep returns in reverse order, that is $n-1, \ldots, 1$. The second variant is termed the *Gauss–Southwell method* [2], according to which the component (variable) with largest partial derivative magnitude in the gradient vector is selected for line searching. The latter requires the availability of first derivatives of the objective function.

The algorithm of the cyclic coordinate method can be summarized as follows:

1. Initialization
Select a tolerance $\epsilon > 0$, to be used in the termination criterion of the algorithm. Select an initial point $\mathbf{x}^{(0)}$ and initialize by setting $\mathbf{z}^{(1)} = \mathbf{x}^{(0)}$. Set $k = 0$ and $i = 1$.

2. Main iteration
Let α_i^* (scalar variable) be the optional solution to the line search problem of minimizing $f(\mathbf{z}^{(i)} + \alpha \mathbf{d}_i)$. Set $\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} + \alpha_i^* \mathbf{d}^{(i)}$. If $j < n$, then increase i to $i+1$ and repeat step 2. Otherwise, if $j = n$, then go to step 3.

3. Termination check
Set $\mathbf{x}^{k+1} = \mathbf{z}^{(n)}$. If the termination criterion is satisfied, for example $||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}|| \leq \epsilon$, then stop. Else, set $\mathbf{z}^{(1)} = \mathbf{x}^{(k+1)}$. Increase k to $k+1$, set $i = 1$ and repeat step 2.

The steps above outline the basic cyclic coordinate method, the Aitken and Gauss–Southwell variants can be easily included by modifying the main algorithm.

In terms of convergence rate comparisons, D.G. Luenberger [3] remarks that such comparisons are not easy. However, an interesting analysis presented there indicates that roughly $n-1$ coordinate searches can be as effective as a single gradient search. Unless the variables are practically uncoupled from one another then coordinate search seems to require approximately n line searches to bring about the same effect as one step of steepest descent.

It can generally be proved that the cyclic coordinate method, when applied to a differentiable function, will converge to a stationary point [1,3]. However, when differentiability is not present then the method can stall at a suboptimal point. Interestingly there are ways to overcome such difficulties, such as by applying at every pth iteration (a heuristic number, user specified) the search direction $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$. This is even applied in practice for differentiable functions, as it is found to be helpful in accelerating convergence. These modifications are referred to as *acceleration steps* or *pattern searches*.

See also

- ► Powell Method
- ► Rosenbrock Method
- ► Sequential Simplex Method

References

1. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming, theory and algorithms. Wiley, New York
2. Forsythe GE (1960) Finite difference methods for partial differential equations. Wiley, New York
3. Luenberger DG (1984) Linear and nonlinear programming, 2nd edn. Addison-Wesley, Reading, MA

D

Data Envelopment Analysis

DEA

R. De Leone
Dip. Mat. e Fisica, University degli Studi di Camerino, Camerino, Italy

MSC2000: 90B50, 90B30, 91B82, 90C05

Article Outline

Keywords
See also
References

Keywords

DEA; Comparative efficiency assessment; Linear programming

Data envelopment analysis (DEA) is a novel technique based on linear programming for evaluating the relative *performance* of similar units, referred to as decision making units (DMUs). The system under evaluation consists of n DMUs: each DMU consumes varying amount of m_1 different inputs (resources) to produce m_2 different outputs (products). Specifically, the jth DMU is characterized by the input vector $x^j > 0$ and the output vector $y^j > 0$. The aim of DEA is to discern, for each DMU, whether or not is operating in an efficient way, given its inputs and outputs, relative to all remaining DMUs under consideration. The measure of *efficiency* is the ratio of a weighted sum of the outputs to a weighted sum of the inputs. For each DMU, the weights are different and obtained by solving a linear programming problem with the objective of showing the DMU in the best possible light.

The ability to deal directly with incommensurable inputs and outputs, the possibility of each DMU of adopting a different set of weights and the focus on individual observation in contrast to averages are among the most appealing features of model based on DEA.

A process is defined *output-efficient* if there is no other process that, using the same or smaller amount of inputs, produces higher level of outputs. A process is defined *input-efficient* if there is no other process that produces the same or higher level of outputs, using smaller amount of inputs. For each orientation there are four possible models:

1) the 'constant returns' model;
2) the 'variable returns' model;
3) the 'increasing returns' model;
4) the 'decreasing returns' model.

Each model is defined by a specific set of economic assumptions regarding the relation between inputs and outputs [10,11]. Associated with each of the four DEA models, independent of the orientation, there is a *production possibility set*, that is, the set of all possible inputs and outputs for the entire system. This set consists of the n DMUs and of 'virtual' DMUs obtained as linear combination of the original data. The efficient frontier is a subset of the boundary points of this production set. The objective of DEA is to determine if the DMU under evaluation lies on the efficient frontier and to assign a score based on the distance from this frontier [6].

The production set for the 'constant returns' model is

$$T_1 = \left\{ (x,y) \geq 0 : \begin{array}{c} x \geq \mu \sum_{j=1}^n x^j \lambda_j, \\ y \leq \mu \sum_{j=1}^n y^j \lambda_j, \\ \forall \lambda_j \geq 0, \quad \sum_{j=1}^n \lambda_j = 1, \\ \mu > 0 \end{array} \right\},$$

while for the 'variable returns' model we have

$$T_2 = \left\{ (x,y) \geq 0 : \begin{array}{c} x \geq \mu \sum_{j=1}^n x^j \lambda_j, \\ y \leq \mu \sum_{j=1}^n y^j \lambda_j, \\ \forall \lambda_j \geq 0, \quad \sum_{j=1}^n \lambda_j = 1 \end{array} \right\}.$$

The production sets for the 'increasing' (resp. 'decreasing') returns models are similar to the set T_2 above with the equality constraint $\sum_{j=1}^n \lambda_j = 1$ replaced by the inequality $\sum_{j=1}^n \lambda_j \leq 1$ (resp. $\sum_{j=1}^n \lambda_j \geq 1$).

The 'constant returns, input oriented' envelopment LP is given next:

$$\begin{cases} \min\limits_{\theta,\lambda \geq 0} & \theta \\ \text{s.t.} & \sum\limits_{j=1}^n x^j \lambda_j - x^{j^*} \theta \leq 0 \\ & \sum\limits_{j=1}^n y^j \lambda_j \geq y^{j^*}. \end{cases} \tag{1}$$

For the 'constant returns, output oriented' case we have, instead, the following LP problem [4]:

$$\begin{cases} \max\limits_{\psi,\lambda \geq 0} & \psi \\ \text{s.t.} & \sum\limits_{j=1}^n x^j \lambda_j \leq x^{j^*} \\ & \sum\limits_{j=1}^n y^j \lambda_j - y^{j^*} \psi \geq 0. \end{cases} \tag{2}$$

In both cases the additional constraint

$$\sum_{j=1}^n \lambda_j \begin{pmatrix} = \\ \leq \\ \geq \end{pmatrix} 1$$

defines the LP for the variable, increasing and decreasing returns DEA models, respectively.

The corresponding dual problem for the input-oriented case is

$$\begin{cases} \max\limits_{\pi,\sigma,\beta} & \sigma^\top y^{j^*} + \beta \\ \text{s.t.} & -\pi^\top x^j + \sigma^\top y^j + \beta \leq 0 \\ & j = 1, \ldots, n \\ & \pi^\top x^{j^*} \leq 1 \\ & \pi \geq 0, \quad \sigma \geq 0, \end{cases} \tag{3}$$

where $\beta = 0$, β unrestricted, $\beta \leq 0$ and $\beta \geq 0$ for the constant, variable, increasing and decreasing return DEA models.

For the output-oriented case the dual is:

$$\begin{cases} \min\limits_{\pi,\sigma,\beta} & \pi^\top x^{j^*} + \beta \\ \text{s.t.} & \pi^\top x^j - \sigma^\top y^j + \beta \geq 0 \\ & j = 1, \ldots, n \\ & \sigma^\top y^{j^*} \geq 1 \\ & \pi \geq 0, \quad \sigma \geq 0 \end{cases} \tag{4}$$

with $\beta = 0$, β unrestricted $\beta \geq 0$ and $\beta \leq 0$ for the constant, variable, increasing and decreasing returns DEA models.

For the 'input-oriented, constant returns' case, the reference DMU j^* is

- *inefficient* if
  - the optimal value of problem (1) is different from 1, or
  - the optimal value of Problem (1) is equal to 1 but there exists an optimal solution with at least one slack variable strictly positive;
- *efficient* in the remaining cases.

Moreover the efficient DMU j^* can be

- *extreme-efficient* if Problem (1) has the unique solution $\lambda^*_{j^*} = 1, \lambda^*_j = 0, j = 1, \ldots, n, j \neq j^*$;
- *nonextreme efficient* when Problem (1) has alternate optimal solutions.

The efficiency for the other models is defined in a similar manner.

The conditions $\theta \geq 0$ and $\psi \geq 0$ can be introduced without loss of generality in (1) and (2) since only non-negative values for these variables are possible given our assumption on the data. Since $\lambda_{j*} = 1, \lambda_j = 0$ for $j \neq j^*$, $\theta^* = 1$, and $\lambda_{j*} = 1, \lambda_j = 0$ for $j \neq j^*$, $\psi^* = 1$ are always feasible for (1) and (2), respectively, the optimal objective function value lies in the interval $(0, 1]$ for the input orientation case and $[1, \infty)$ for the output orientation case.

The linear programs (1) and (3) above can be interpreted in the following way. In the input-oriented case, we compare the reference DMU j^* with a 'virtual' DMU obtained as linear combination of the original DMUs. Each input and output of this virtual DMU is a linear combination of the corresponding component of the inputs and outputs of all the DMUs. The optimal value is, in this case, always less than or equal to 1. If the optimal value is strictly less than 1, then it is possible to construct a virtual DMU that produces at least the

same amount of outputs as the reference DMU using an amount of inputs that is strictly smaller than amount used by the j^*th DMU. When this happens we declare the DMU j^* inefficient. Instead, when the optimal value is equal to 1 there are three possible cases:

- there exists an optimal solution with at least one slack variable strictly positive;
- the optimal solution is unique;
- there exists multiple optimal solutions.

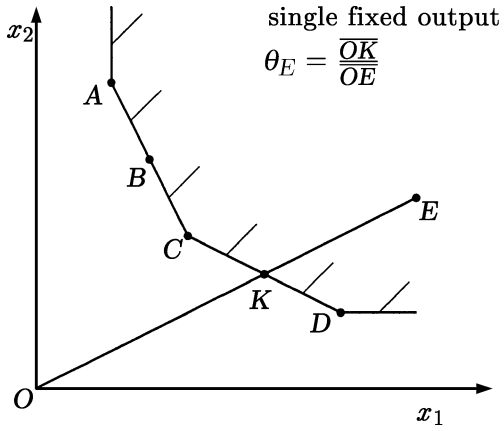
In the first case we declare the reference DMU inefficient. In the last two cases the j^*th DMU is efficient (extreme-efficient, respectively nonextreme efficient).

For Problem (3), the optimal solution π^* and σ^* represent the weights that are the most favorable for the reference DMU, i. e., the weights that produce the highest efficiency score under the hypothesis that, using the same weights for the other DMUs, the efficiency remains always below 1.

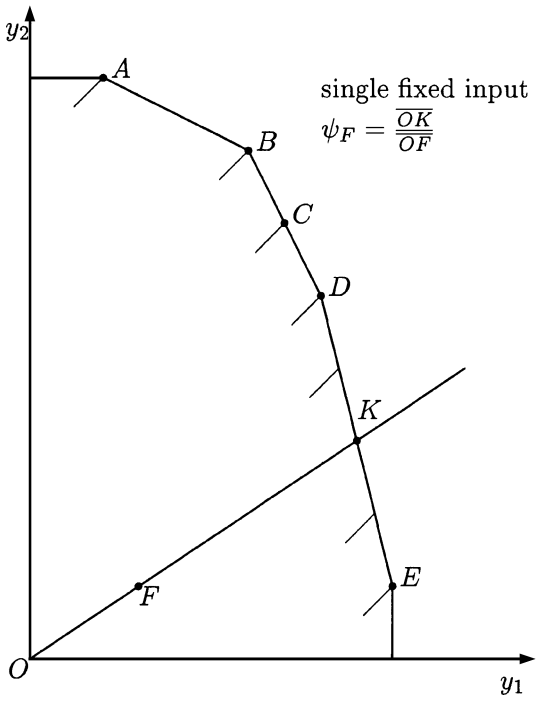
Similar interpretations can be given for the output-oriented case for Problems (2) and (4).

In Fig. 1 it is represented the production possibility set and the efficient frontier for the five DMUs 'A' to 'E'. These DMUs are characterized by two different inputs and a single output value set to some fixed value.

All the DMUs are efficient but the DMU 'E'. The DMU 'B' is efficient but nonextreme. The virtual DMU 'K', obtained as convex combination of the DMUs 'C' and 'D', is more efficient than the DMU 'E'. The optimal value θ^* for the linear programming problem (1) for the DMU 'E' is exactly the ratio of the lengths of the segments OE and OK.



Data Envelopment Analysis, Figure 1
Two-input, single output DMUs



Data Envelopment Analysis, Figure 2
Two-output, single input DMUs

Figure 2 shows the case of DMUs characterized by two distinct outputs and a single input set to a fixed value. All the DMUs are efficient except the DMU 'F' that is dominated by the virtual DMU 'K'. The optimal value ψ^* for the linear programming problem (2) for the DMU 'F' is the ratio of the lengths of the segments OE and OK.

The original 'constant returns' model was proposed in [4]. In [2] the *variable returns* model was proposed with the objective of discriminating between technical efficiency and scale efficiency. The bibliography published in [7] (part of [3]) contains more than 500 references to published article in the period 1978–1992 and many more articles appeared since.

In all the DEA models discussed above, all efficient DMUs receive an equal score of 1. An important modification proposed in [1] allows to rank efficient units. The main idea is to exclude the column being scored from the DEA envelopment LP technology matrix. The efficiency score is now a value between $(0, +\infty]$ in both orientations. In [5] are discussed in detail the issues (infeasibility, relationship between modified and standard formulation, degeneracy, interpretation of the optimal solutions) related to these DEA models.

In [8] and [9] the properties of 'unit invariance' (independence of the units in which inputs and outputs are measured) and 'translation invariance' (independence of an affine translation of the inputs and the outputs) of an efficiency DEA measure are discussed. The translation invariance property is particularly important when data contain zero or negative values. Standard DEA models are not unit invariant and translation invariant. In [8] it is proposed a weighted additive DEA model that satisfies these properties:

$$
\begin{cases}
\min\limits_{\lambda,s^+,s^-} & \sum\limits_{i=1}^{m_1} w_i^+ s_i^+ + \sum\limits_{r=1}^{m_2} w_r^- s_r^- \\
\text{s.t.} & \sum\limits_{j=1}^{n} x_i^j \lambda_j + s_i^+ = x_i^{j^*} \\
& i = 1, \ldots, m_1 \\
& \sum\limits_{j=1}^{n} y_r^j \lambda_j - s_r^- = y_r^{j^*} \\
& r = 1, \ldots, m_2 \\
& \sum\limits_{j=1n}^{n} \lambda_j = 1 \\
& \lambda_j \geq 0, \quad j = 1, \ldots, n, \\
& s_i^+ \geq 0, \quad i = 1, \ldots, m_1, \\
& s_r^- \geq 0, \quad r = 1, \ldots, m_2.
\end{cases}
\tag{5}
$$

where w_i^+ and w_r^- are the sample standard deviation of the inputs and outputs variables respectively.

Models based on data envelopment analysis have been widely used in order to evaluate efficiency in both public and private sectors. [3, Part II] contains 15 application of DEA showing the 'range, power, elegance and insight obtainable via DEA analysis'. Banks, hospitals, and universities are among the most challenging sectors where models based on DEA have been able to assess efficiency and determine strength and weakness of the various units.

See also

► Optimization and Decision Support Systems

References

1. Andersen P, Petersen NC (1993) A procedure for ranking efficient units in data envelopment analysis. mansci 10:1261–1264
2. Banker RD, Charnes A, Cooper WW (1984) Some models for estimating technological and scale inefficiencies in data envelopment analysis. mansci 30:1078–1092
3. Charnes A, Cooper W, Lewin AY, Seiford LM (1994) Data envelopment analysis: Theory, methodology and applicastions. Kluwer, Dordrecht
4. Charnes A, Cooper WW, Rhodes E (1978) Measuring the efficiency of decision making units. ejor 2:429–444
5. Dulá JH, Hickman BL (1997) Effects of excluding the column being scored from the DEA envelopment LP technology matrix. jors 48:1001–1012
6. Farrell MJ (1957) The measurement of productive efficiency. J Royal Statist Soc A120:253–281
7. Seiford LM (1994) A DEA bibliography (1978–1992). In: Charnes A, Cooper W, Lewin AY, Seiford LM (eds) Data Envelopment Analysis: Theory, Methodology and Applicastions. Kluwer, Dordrecht, pp 437–469
8. Pastor JT (1994) New DEA additive model for handling zero and negative data. Techn Report Dept Estadistica e Invest. Oper Univ Alicante
9. Pastor JT, Knox Lovell CA (1995) Units invariant and translation invariant DEA models. orl 18:147–151
10. Shepard RW (1953) Cost and production functions. Princeton Univ. Press, Princeton
11. Shepard RW (1970) The theory of cost and production functions. Princeton Univ Press, Princeton

Data Mining

DAVID L. OLSON
Department of Management, University of Nebraska, Lincoln, USA

Article Outline

Synonyms
Introduction
Definitions
Example Applications
  Customer Relationship Management (CRM)
  Credit Scoring
  Bankruptcy Prediction
  Fraud Detection
Ethical Issues in Data Mining
Conclusions
References

Synonyms

Data mining; Large-scale data analysis; Pattern recognition

Introduction

Data mining has proven valuable in almost every aspect of life involving large data sets. Data mining is made possible by the generation of masses of data from computer information systems. In engineering, satellites stream masses of data down to storage systems, yielding a mountain of data that needs some sort of data mining to enable humans to gain knowledge. Data mining has been applied in engineering applications such as quality [4], manufacturing and service [13], labor scheduling [17], and many other places. Medicine has been an extensive user of data mining, both in the technical area [21] and in health policy [6]. Pardalos [16] provide recent research in this area. Governmental operations have received support from data mining, primarily in the form of fraud detection [9].

In business, data mining has been instrumental in customer relationship management [5,8], financial analysis [3,12], credit card management [1], health service debt management [22], banking [19], insurance [18], and many other areas of business involving services. Kusiak [13] reviewed data mining applications to include service applications of operations. Recent reports of data mining applications in web service and technology include Tseng and Lin [20] and Hou and Yang [11]. In addition to Tseng and Lin, Lee et al. [14] discuss issues involving mobile technology and data mining. Data mining support is required to make sense of the masses of business data generated by computer technology. Understanding this information-generation system and tools available leading to analysis is fundamental for business in the 21st century. The major applications have been in customer segmentation (by banks and retail establishments wishing to focus on profitable customers) and in fraud and rare event detection (especially by insurance and government, as well as by banks for credit scoring). Data mining has been used by casinos in customer management, and by organizations evaluating personnel.

We will discuss data mining functions, data mining process, data systems often used in conjunction with data mining, and provide a quick review of software tools. Four prototypical applications are given to demonstrate data mining use in business. Ethical issues will also be discussed.

Definitions

There are a few basic functions that have been applied in business. Bose and Mahapatra [2] provided an extensive list of applications by area, technique, and problem type.

- **Classification** uses a training data set to identify classes or clusters, which then are used to categorize data. Typical applications include categorizing risk and return characteristics of investments, and credit risk of loan applicants. The Adams [1] case, for example, involved classification of loan applications into groups of expected repayment and expected problems.
- **Prediction** identifies key attributes from data to develop a formula for prediction of future cases, as in regression models. The Sung et al. [19] case predicted bankruptcy while the Drew et al. [5]) case and the customer retention part of the Smith et al. [18] case predicted churn.
- **Association** identifies rules that determine the relationships among entities, such as in market basket analysis, or the association of symptoms with diseases. IF–THEN rules were shown in the Sung et al. [19] case.
- **Detection** determines anomalies and irregularities, valuable in fraud detection. This was used in claims analysis by Smith et al. [18].

To provide analysis, data mining relies on some fundamental analytic approaches. Regression and neural network approaches are alternative ways to identify the best fit in a given set of data. Regression tends to have advantages with linear data, while neural network models do very well with irregular data. Software usually allows the user to apply variants of each, and lets the analyst select the model that fits best. Cluster analysis, discriminant analysis, and case-based reasoning seek to assign new cases to the closest cluster of past observations. Rule induction is the basis of decision tree methods of data mining. Genetic algorithms apply to special forms of data, and are often used to boost or improve the operation of other techniques.

In order to conduct data mining analyzes, a **data mining process** is useful. The Cross-Industry Standard Process for Data Mining (CRISP-DM) is widely used by industry members [15]. This model consists of six phases intended as a cyclical process:

- **Business understanding:** Business understanding includes determining business objectives, assessing the current situation, establishing data mining goals, and developing a project plan.
- **Data understanding:** Once business objectives and the project plan are established, data understanding considers data requirements. This step can include initial data collection, data description, data exploration, and the verification of data quality. Data exploration such as viewing summary statistics (which includes the visual display of categorical variables) can occur at the end of this phase. Models such as cluster analysis can also be applied during this phase, with the intent of identifying patterns in the data.
- **Data preparation:** Once the data resources available are identified, they need to be selected, cleaned, built into the form desired, and formatted. Data cleaning and data transformation in preparation for data modeling needs to occur in this phase. Data exploration at a greater depth can be applied during this phase, and additional models utilized, again providing the opportunity to see patterns based on business understanding.
- **Modeling:** Data mining software tools such as visualization (plotting data and establishing relationships) and cluster analysis (to identify which variables go well together) are useful for initial analysis. Tools such as generalized rule induction can develop initial association rules. Once greater data understanding is gained (often through pattern recognition triggered by viewing model output), more detailed models appropriate to the data type can be applied. The division of data into training and test sets is also needed for modeling (sometimes even more sets are needed for model refinement).
- **Evaluation:** Model results should be evaluated in the context of the business objectives established in the first phase (business understanding). This will lead to the identification of other needs (often through pattern recognition), frequently reverting to prior phases of CRISP-DM. Gaining business understanding is an iterative procedure in data mining, where the results of various visualization, statistical, and artificial intelligence tools show the user new relationships that provide a deeper understanding of organizational operations.
- **Deployment:** Data mining can be used both to verify previously held hypotheses, and for knowledge discovery (identification of unexpected and useful relationships). Through the knowledge discovered in the earlier phases of the CRISP-DM process, sound models can be obtained that may then be applied to business operations for many purposes, including prediction or identification of key situations. These models need to be monitored for changes in operating conditions, because what might be true today may not be true a year from now. If significant changes do occur, the model should be redone. It is also wise to record the results of data mining projects so documented evidence is available for future studies.

This six-phase process is not a rigid, by-the-numbers procedure. There is usually a great deal of backtracking. Additionally, experienced analysts may not need to apply each phase for every study. But CRISP-DM provides a useful framework for data mining.

There are many **database systems** that provide content needed for data mining. Database software is available to support individuals, allowing them to record information that they consider personally important. They can extract information provided by repetitive organizational reports, such as sales by region within their area of responsibility, and regularly add external data such as industry-wide sales, as well as keep records of detailed information such as sales representative expense account expenditure.

- **Data warehousing** is an orderly and accessible repository of known facts and related data that is used as a basis for making better management decisions. Data warehouses provide ready access to information about a company's business, products, and customers. This data can be from both internal and external sources. Data warehouses are used to store massive quantities of data in a manner that can be easily updated and allow quick retrieval of specific types of data. Data warehouses often integrate information from a variety of sources. Data needs to be identified and obtained, cleaned, catalogued, and stored in a fashion that expedites organizational decision making. Three general data warehouse processes exist. (1) Warehouse generation is the process of designing the warehouse and loading data. (2) Data management is the process of storing the

data. (3) Information analysis is the process of using the data to support organizational decision making.

- **Data marts** are sometimes used to extract specific items of information for data mining analysis. Terminology in this field is dynamic, and definitions have evolved as new products have entered the market. Originally, many data marts were marketed as preliminary data warehouses. Currently, many data marts are used in conjunction with data warehouses rather than as competitive products. But also many data marts are being used independently in order to take advantage of lower-priced software and hardware. Data marts are usually used as repositories of data gathered to serve a particular set of users, providing data extracted from data warehouses and/or other sources. Designing a data mart tends to begin with the analysis of user needs. The information that pertains to the issue at hand is relevant. This may involve a specific time-frame and specific products, people, and locations. Data marts are available for data miners to transform information to create new variables (such as ratios, or coded data suitable for a specific application). In addition, only that information expected to be pertinent to the specific data mining analysis is extracted. This vastly reduces the computer time required to process the data, as data marts are expected to contain small subsets of the data warehouse's contents. Data marts are also expected to have ample space available to generate additional data by transformation.
- **Online analytical processing** (OLAP) is a multidimensional spreadsheet approach to shared data storage designed to allow users to extract data and generate reports on the dimensions important to them. Data is segregated into different dimensions and organized in a hierarchical manner. Many variants and extensions are generated by the OLAP vendor industry. A typical procedure is for OLAP products to take data from relational databases and store them in multidimensional form, often called a **hypercube**, to reflect the OLAP ability to access data on these multiple dimensions. Data can be analyzed locally within this structure. One function of OLAP is standard report generation, including financial performance analysis on selected dimensions (such as by department, geographical region, product, salesperson, time, or other dimensions desired by the analyst). Planning and forecasting are supported through spreadsheet analytic tools. Budgeting calculations can also be included through spreadsheet tools. Usually, pattern analysis tools are available.

There are many statistical and analytic **software tools** marketed to provide data mining. Many good data mining software products are being used, including the well-established (and expensive) Enterprise Miner by SAS and Intelligent Miner by IBM, CLEMENTINE by SPSS (a little more accessible by students), PolyAnalyst by Megaputer, and many others in a growing and dynamic industry. For instance, SQL Server 2005 has recently been vastly improved by Microsoft, making a more usable system focused on the database perspective.

These products use one or more of a number of analytic approaches, often as complementary tools that might involve initial cluster analysis to identify relationships and visual analysis to try to understand why data clustered as it did, followed by various prediction models. The major categories of methods applied are regression, decision trees, neural networks, cluster detection, and market basket analysis. The Web site www.KDnuggets.com gives information on many products, classified by function. In the category of overall data mining suites, they list 56 products in addition to 16 free or shareware products. Specialized software products were those using multiple approaches (15 commercial plus 3 free), decision tree (15 plus 10 free), rule-based (7 plus 4 free), neural network (12 plus 3 free), Bayesian (13 plus 11 free), support vector machines (3 plus 8 free), cluster analysis (8 plus 10 free), text mining (50 plus 4 free), and other software for functions such as statistical analysis, visualization, and Web usage analysis.

Example Applications

There are many applications of data mining. Here we present four short examples in the business world.

Customer Relationship Management (CRM)

The idea of customer relationship management is to target customers for special treatment based on their anticipated future value to the firm. This requires estimation of where in the customer life-cycle each subject is, as well as lifetime customer value, based on expected

tenure with the company, monthly transactions by that customer, and the cost of providing service. Lifetime value of a customer is the discounted expected stream of cash flow generated by the customer.

Many companies applying CRM score each individual customer by their estimated lifetime value (LTV), stored in the firm's customer database [5]. This concept has been widely used in catalog marketing, newspaper publishing, retailing, insurance, and credit cards. LTV has been the basis for many marketing programs offering special treatment such as favorable pricing, better customer service, and equipment upgrades.

While CRM is very promising, it has often been found to be less effective than hoped [10]. CRM systems can cost up to $70 million to develop, with additional expenses incurred during implementation. Many of the problems in CRM expectations have been blamed on over-zealous sales pitches. CRM offers a lot of opportunities to operate more efficiently. However, they are not silver bullets, and benefits are not unlimited. As with any system, prior evaluation of benefits is very difficult, and investments in CRM systems need to be based on sound analysis and judgment.

Credit Scoring

Data mining can involve model building (extension of conventional statistical model building to very large data sets) and pattern recognition. Pattern recognition aims to identify groups of interesting observations. Interesting is defined as discovery of knowledge that is important and unexpected. Often experts are used to assist in pattern recognition. Adams et al. [1] compared data mining used for model building and pattern recognition on the behavior of customers over a one-year period. The data set involved bank accounts at a large British credit card company observed monthly. These accounts were revolving loans with credit limits. Borrowers were required to repay at least some minimum amount each month. Account holders who paid in full were charged no interest, and thus not attractive to the lender.

We have seen that clustering and pattern search are typically the first activities in data analysis. Then appropriate models are built. Credit scoring is a means to use the results of data mining modeling for two purposes. Application scoring was applied in the Adams et al. example to new cases, continuing an activity that had been done manually for half a century in this organization. Behavioral scoring monitors revolving credit accounts with the intent of gaining early warnings of accounts facing difficulties.

Bankruptcy Prediction

Corporate bankruptcy prediction is very important to management, stockholders, employees, customers, and other stakeholders. A number of data mining techniques have been applied to this problem, including multivariate discriminant analysis, logistical regression, probit, genetic algorithms, neural networks, and decision trees.

Sung et al. [19] applied decision analysis and decision tree models to a bankruptcy prediction case. Decision tree models provide a series of IF–THEN rules to predict bankruptcy. Pruning (raising the proportion of accurate fit required to keep a specific IF–THEN relationship) significantly increased overall prediction accuracy in the crisis period, indicating that data collected in the crisis period was more influenced by noise than data from the period with normal conditions. Example rules obtained were as shown in Table 1, giving an idea of how decision tree rules work.

For instance, in normal conditions, if the variable Productivity of capital (E6) was greater than 19.65, the model would predict firm survival with 86 percent confidence. Conversely, if Productivity of capital (E6) was less than or equal to 19.65, and if the Ratio of cash flow to total assets (C9) was less than or equal to 5.64, the model would predict bankruptcy with 84 percent confidence. These IF–THEN rules are stated in ways that are easy for management to see and use. Here the rules are quite simple, a desirable feature. With large data sets, it is common to generate hundreds of clauses in decision tree rules, making it difficult to implement (although gaining greater accuracy). The number of rules can be controlled through pruning rates within the software.

Fraud Detection

Data mining has successfully supported many aspects of the insurance business, to include fraud detection, underwriting, insolvency prediction, and customer segmentation. An insurance firm had a large data warehouse system recording details on every transac-

Data Mining, Table 1
Bankruptcy Prediction Rules

Condition	Rule	Prediction	Confidence level
Normal	E6>19.65	Nonbankrupt	0.86
Normal	C9>5.64	Nonbankrupt	0.95
Normal	C9≤5.64 & E6≤19.65	Bankrupt	0.84
Crisis	E6>20.61	Nonbankrupt	0.91
Crisis	C8>2.64	Nonbankrupt	0.85
Crisis	C3>87.23	Nonbankrupt	0.86
Crisis	C8≤2.64, E6≤20.61, & C3≤87.23	Bankrupt	0.82

Where C3 = Ratio of fixed assets to equity & long-term liabilities. C8 = Ratio of cash flow to liabilities. C9 = Ratio of cash flow to total assets. E6 = Productivity of capital. Based on Sung et al. [19]

tion and claim [18]. An aim of the analysis was to accurately predict average claim costs and frequency, and to examine the impact of pricing on profitability.

In evaluating claims, data analysis for hidden trends and patterns is needed. In this case, recent growth in the number of policy holders led to lower profitability for the company. Understanding the relationships between cause and effect is fundamental to understanding what business decisions would be appropriate.

Policy rates are based on statistical analysis assuming various distributions for claims and claim size. In this case, clustering was used to better model the performance of specific groups of insured.

Profitability in insurance is often expressed by the cost ratio, or sum of claim costs divided by sum of premiums. Claim frequency ratio is the number of claims divided by the number of policy units of risk (possible claims). Profitability would be improved by lowering the frequency of claims, or the costs of claims relative to premiums.

Data was extracted from the data warehouse for policies for which premiums were paid in the first quarter over a three-year period. This meant that policies were followed over the period, augmented by new policies, and diminished by terminations. Data on each policy holder was available as well as claim behavior over the preceding year. The key variables of cost ratio and claim frequency ratio were calculated for each observation. Sample sizes for each quarter were well above 100,000.

Descriptive statistics found exceptional growth in policies over the past two years for young people (under 22), and with cars insured for over $40,000. Clustering analysis led to the conclusion that the claim cost of each individual policy holder would be pointless, as the vast majority of claims could not be predicted. Af-

Data Mining, Table 2
General Ability of Data Mining Techniques to Deal with Data Features

Data characteristic	Rule induction	Neural networks	Case-based reasoning	Genetic algorithms
Handle noisy data	Good	Very good	Good	Very good
Handle missing data	Good	Good	Very good	Good
Process large data sets	Very good	Poor	Good	Good
Process different data types	Good	Transform to numerical	Very good	Transforma-tion needed
Predictive accuracy	High	Very high	High	High
Explanation capability	Very good	Poor	Very good	Good
Ease of integration	Good	Good	Good	Very good
Ease of operation	Easy	Difficult	Easy	Difficult

Extracted from Bose and Mahapatra [2]

ter experimentation, the study was based on 50 clusters. A basic k-means algorithm was used. This identified several clusters as having abnormal cost ratios or frequency sizes. By testing over a two-year gap, stability for each group was determined. Table 2 compares data mining techniques.

Ethical Issues in Data Mining

Data mining is a potentially useful tool, capable of doing a lot of good, not only for business but also for the medical field and for government. It does, however, bring with it some dangers. So, how can we best protect ourselves, especially in the area of business data mining?

A number of options exist. Strict control of data usage through governmental regulation was proposed by Garfinkel [7]. A number of large database projects that made a great deal of practical sense have ultimately been stopped. Those involving government agencies were successfully stopped due to public exposure, the negative outcry leading to cancellation of the National Data Center and the Social Security Administration projects. A system with closely held information by credit bureaus in the 1960s was only stopped after governmental intervention, which included the passage of new laws. Times have changed, with business adopting a more responsive attitude toward consumers. Innovative data mining efforts by Lotus/Equifax and by LexisNexis were quickly stopped by public pressure alone.

Public pressure seems to be quite effective in providing some control over potential data mining abuses. If that fails, litigation is available (although slow in effect). It is necessary for us to realize what businesses can do with data. There will never be a perfect system to protect us, and we need to be vigilant. However, too much control can also be dangerous, inhibiting the ability of business to provide better products and services through data mining. Garfinkel prefers more governmental intervention, while we would prefer less governmental intervention and more reliance on publicity and, if necessary, the legal system.

Control would be best accomplished if it were naturally encouraged by systemic relationships. The first systemic means of control is publicity. Should those adopting questionable practices persist, litigation is a slow, costly, but ultimately effective means of system correction. However, before taking drastic action, a good rule is that if the system works, it is best not to fix it. The best measure that electronic retailers can take is to not do anything that will cause customers to suspect that their rights are being violated.

Conclusions

Data mining has evolved into a useful analytic tool in all aspects of human study, to include medicine, engineering, and science. It is a necessary means to cope with the masses of data that are produced in contemporary society. Within business, data mining has been especially useful in applications such as fraud detection, loan analysis, and customer segmentation. Such applications heavily impact the service industry. Data mining provides a way to quickly gain new understanding based upon large-scale data analysis.

This paper reviewed some of the applications that have been applied in services. It also briefly reviewed the data mining process, some of the analytic tools available, and some of the major software vendors of general data mining products. Specific tools for particular applications are appearing with astonishing rapidity.

References

1. Adams NM, Hand DJ, Till RJ (2001) Mining for classes and patterns in behavioural data. J Oper Res Soc 52(9):1017–1024
2. Bose I, Mahapatra RK (2001) Business data mining – a machine learning perspective. Inf Manage 39(3):211–225
3. Cowan AM (2002) Data mining in finance: Advances in relational and hybrid methods. Int J Forecasting 18(1):155–156
4. Da Cunha C, Agard B, Kusiak A (2006) Data mining for improvement of product quality. Int J Prod Res 44(18/19): 4027–4041
5. Drew JH, Mani DR, Betz AL, Datta P (2001) Targeting customers with statistical and data-mining techniques. J Serv Res 3(3):205–219
6. Garfinkel MS, Sarewitz D, Porter AL (2006) A societal outcomes map for health research and policy. Am J Public Health 96 (3):441–446
7. Garfinkel S (2000) Database Nation: The Death of Privacy in the 21st Century. O'Reilly & Associates, Sebastopol CA
8. Garver MS (2002) Using data mining for customer satisfaction research. Mark Res 14(1):8–12
9. Government Accounting Office (2006) Hurricanes Katrina and Rita: Unprecedented challenges exposed the individ-

uals and households program to fraud and abuse: Actions needed to reduce such problems in future: GAO-06–1013, 9/27/2006, pp 1–110
10. Hart ML (2006) Customer relationship management: Are software applications aligned with business objectives? South African J Bus Manage 37(2):17–32
11. Hou J-L, Yang S-T (2006) Technology-mining model concerning operation characteristics of technology and service providers. Int J Prod Res 44(16):3345–3365
12. Hui W, Weigend AS (2004) Data mining for financial decision making. Decis Support Syst 37(4):457–460
13. Kusiak A (2006) Data mining: Manufacturing and service applications. Int J Prod Res 44(18/19):4175–4191
14. Lee S, Hwang C-S, Kitsuregawa M (2006) Efficient, energy conserving transaction processing in wireless data' broadcast. IEEE Trans Knowl Data Eng 18(9):1225–1238
15. Olson DL, Shi Y (2007) Introduction to Business Data Mining. McGraw-Hill/Irwin, Englewood Cliffs, NJ
16. Pardalos PM, Boginski VL, Vazacopoulos A (eds) (2007) Data Mining in Biomedicine. Springer, Heidelberg
17. Qi X, Bard JF (2006) Generating labor requirements and rosters for mail handlers. Comput Oper Res 33(9):2645–2666
18. Smith KA, Willis RJ, Brooks M (2000) An analysis of customer retention and insurance claim patterns using data mining: A case study. J Oper Res Soc 51(5):532–541
19. Sung TK, Chang N, Lee G (1999) Dynamics of modeling in data mining: Interpretive approach to bankruptcy prediction. J Manage Inf Sys 16(1):63–85
20. Tseng VS, Lin KW (2006) Efficient mining and prediction of user behavior patterns in mobile web systems. Inf Softw Technol 48(6):357–369
21. Yamaguchi M, Kaseda C, Yamazaki K, Kobayashi M (2006) Prediction of blood glucose level of type 1 diabetics using response surface methodology and data mining. Med Biol Eng Comput 44(6):451–457
22. Zurada J, Lonial S (2005) Comparison of the performance of several data mining methods for bad debt recovery in the healthcare industry. J Appl Bus Res 21(2):37–54

D.C. Programming

HOANG TUY
Institute of Mathematics, VAST, Hanoi, Vietnam

MSC2000: 90C26

Article Outline

DC Structure in Optimization
Recognizing dc Functions
Global Optimality Criterion
Solution Methods
An OA Method for (CDC)
A BB Method for General DC Optimization
DCA–A Local Optimization Approach to (DC)
Applications and Extensions
References

As optimization techniques become widely used in engineering, economics, and other sciences, an increasing number of nonconvex optimization problems are encountered that can be described in terms of *dc functions* (differences of convex functions). These problems are called dc optimization problems, and the theory dealing with these problems is referred to as dc programming, or dc optimization ([3,4,5,6,13]; see also [1,8]).

Historically, the first dc optimization problem that was seriously studied is the concave minimization problem [11]. Subsequently, reverse convex programming and some other special dc optimization problems such as quadratic and, more generally, polynomial programming problems appeared before a unified theory was developed and the term dc optimization was introduced [12]. In fact, most global optimization problems of interest that have been studied so far can be identified as dc optimization problems, despite the diversity of the approaches used.

DC Structure in Optimization

Let Ω be a convex set in $\mathbb{R}^n$. A function $f\colon \Omega \to \mathbb{R}$ is said to be *dc on* Ω if it can be expressed as the difference of two convex functions on Ω: $f(x) = p(x) - q(x)$, where $p(x), q(x)\colon \Omega \to \mathbb{R}$ are convex. Denote the set of dc functions on Ω by $DC(\Omega)$.

Proposition 1 *$DC(\Omega)$ is a vector lattice with respect to the two operations of pointwise maximum and pointwise minimum.*

In other words, if $f_i(x) \in DC(\Omega)$, $i = 1, \ldots, m$, then:
1. $\sum_{i=1}^{m} \alpha_i f_i(x) \in DC(\Omega)$, for any real numbers α_i;
2. $g(x) = \max\{f_1(x), \ldots, f_m(x)\} \in DC(\Omega)$;
3. $h(x) = \min\{f_1(x), \ldots, f_m(x)\} \in DC(\Omega)$.

From this property it follows in particular that if $f \in DC(\Omega)$, then $|f| \in DC(\Omega)$, and if $g, h \in DC(\Omega)$, then $gh \in DC(\Omega)$. But for the purpose of optimization

the most important consequence is that

$$g_i(x) \le 0,\ \forall i = 1, \dots, m \quad \Leftrightarrow \quad g(x) := \max\{g_1(x), \dots, g_m(x)\} \le 0\,,$$
$$g_i(x) \le 0 \text{ for at least one } i = 1, \dots, m \quad \Leftrightarrow \quad g(x) := \min\{g_1(x), \dots, g_m(x)\} \le 0\,.$$

Therefore, any finite system of dc inequalities, whether conjunctive or disjunctive, can be rewritten as a single dc inequality.

By easy manipulations it is then possible to reduce any dc optimization problem to the following *canonical form*:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && g(x) \le 0 \le h(x)\,, \end{aligned} \tag{CDC}$$

where all functions f, g, h are convex.

Thus dc functions allow a very compact description of a wide class of nonconvex optimization problems.

Recognizing dc Functions

To exploit the dc structure in optimization problems, it is essential to be able to recognize dc functions that are still in hidden form (i. e., not yet presented as differences of convex functions). The next proposition addresses this question.

Proposition 2 *Every function $f \in C^2$ is dc on any compact convex set Ω.*

It follows that any polynomial function is dc, and hence, by the Weierstrass theorem, $DC(\Omega)$ is dense in the Banach space $C(\Omega)$ of continuous functions on Ω with the supnorm topology. In other words, any continuous function can be approximated as closely as desired by a dc function.

More surprisingly, any closed set S in $\mathbb{R}^n$ can be shown to be a *dc set*, i. e., a set that is the solution set of a dc inequality. Namely, given any closed set $S \subset \mathbb{R}^n$ and any strictly convex function $h\colon \mathbb{R}^n \to \mathbb{R}$, there exists a continuous convex function $g_S\colon \mathbb{R}^n \to \mathbb{R}$ such that $S = \{x \in \mathbb{R}^n\colon\ g_S(x) - h(x) \le 0\}$ [10].

In many situations we not only need to recognize a dc function but also to know how to represent it effectively as a difference of two convex functions. While several classes of functions have been recognized as dc functions [2], there are still few results about effective dc representations of these functions. For composite functions a useful result about dc representation is the following [13].

Proposition 3 *Let $h(x) = u(x) - v(x)$, where $u, v\colon \Omega \to \mathbb{R}_+$ are convex functions on a compact convex set $\Omega \subset \mathbb{R}^m$ such that $0 \le h(x) \le a\ \forall x \in \Omega$. If $q\colon [0, a] \to \mathbb{R}$ is a convex nondecreasing function such that $q'_-(a) < \infty$ ($q'_-(a)$ being the left derivative of $q(t)$ at a), then $q(h(x))$ is a dc function on Ω:*

$$q(h(x)) = g(x) - K[a + v(x) - u(x)]\,,$$

where $g(x) = q(h(x)) + K[a + v(x) - v(x)]$ is a convex function and K is any constant satisfying $K \ge q'_-(a)$.

For example, by writing $x^\alpha = e^{h(x)}$ with $h(x) = \sum_{i=1,\dots,n} \alpha_i \log x_i$ and applying the above proposition, it is easy to see that $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, with $\alpha \in \mathbb{R}^n_+$, is dc on any box $\Omega = [r, s] \subset \mathbb{R}^n_{++}$. Hence, any synomial function $f(x) = \sum_\alpha c_\alpha x^\alpha$, with $c_\alpha \in \mathbb{R}$, $\alpha \in \mathbb{R}^n_+$, is also dc on Ω.

Global Optimality Criterion

A key question in the theoretical as well as computational study of a global optimization problem is how to test a given feasible solution for global optimality.

Consider a pair of problems in some sense mutually obverse:

$$\inf\{f(x)\colon x \in \Omega\,,\ h(x) \ge \alpha\}\,, \tag{P_α}$$

$$\sup\{h(x)\colon x \in \Omega\,,\ f(x) \le \gamma\}\,, \tag{Q_γ}$$

where $\alpha, \gamma \in \mathbb{R}$, Ω is a closed set in $\mathbb{R}^n$, and $f, g\colon \mathbb{R}^n \to \mathbb{R}$ are two arbitrary functions.

We say that problem (P_α) is *regular* if

$$\inf P_\alpha = \inf\{f(x)\colon x \in \Omega, h(x) > \alpha\}\,. \tag{1}$$

Analogously, problem (Q_γ) is regular if $\sup Q_\gamma = \sup\{h(x)\colon x \in \Omega, f(x) < \gamma\}$.

Proposition 4 *Let $\bar{x}$ be a feasible solution of problem (P_α). If $\bar{x}$ is optimal to problem (P_α) and if problem (Q_γ) is regular for $\gamma = f(\bar{x})$, then*

$$\sup\{h(x)\colon\ x \in \Omega,\ f(x) \le \gamma\} = \alpha\,. \tag{2}$$

Conversely, if (2) holds and if problem (P_α) is regular, then $\bar{x}$ is optimal to (P_α).

Turning now to the canonical dc optimization problem (CDC), let us set $\Omega = \{x : g(x) \le 0\}$ and without losing generality assume that the reverse convex constraint $h(x) \ge 0$ is essential, i. e.,

$$\inf\{f(x) : x \in \Omega\} < \inf\{f(x) : x \in \Omega, h(x) \ge 0\}. \quad (3)$$

Since CDC is a problem (P_α) with $\alpha = 0$, if $\bar{x}$ is a feasible solution to CDC, then condition (3) ensures the regularity of the associated problem (Q_γ) for $\gamma = f(\bar{x})$. Define

$$C = \{x : h(x) \le 0\}, \quad D(\gamma) = \{x \in \Omega : f(x) \le \gamma\}, \quad (4)$$

and for any set E denote its polar by E^*. As specialized for CDC, Proposition 4 yields:

Proposition 5 *In order that a feasible solution $\bar{x}$ of CDC may be a global minimizer, it is necessary that the following equivalent conditions hold for $\gamma = f(\bar{x})$:*

$$D(\gamma) \subset C, \quad (5)$$

$$0 = \max\{h(x) : x \in D(\gamma)\}, \quad (6)$$

$$C^* \subset [D(\gamma)]^*. \quad (7)$$

If the problem is regular, then any one of the above conditions is also sufficient.

An important special dc program is the following problem:

$$\text{minimize} \quad g(x) - h(x) \quad \text{subject to} \quad x \in \mathbb{R}^n, \quad \text{(DC)}$$

where $g, h : \mathbb{R}^n \to \bar{\mathbb{R}}$ are convex functions ($\bar{R}$ denotes the set of extended real numbers). Writing this problem as $\min\{g(x) - t : x \in D, h(x) \ge t\}$ with $D = \text{dom}g \cap \text{dom}h$ and using (7), one can derive the following:

Proposition 6 *Let $g, h : \mathbb{R}^n \to \bar{\mathbb{R}}$ be two convex functions such that $h(x)$ is proper and lsc. Let $\bar{x}$ be a point where $g(\bar{x})$ and $h(\bar{x})$ are finite. In order for $\bar{x}$ to be a global minimizer of $g(x) - h(x)$ over $\mathbb{R}^n$, it is necessary and sufficient that*

$$\partial_\varepsilon h(\bar{x}) \subset \partial_\varepsilon g(\bar{x}) \quad \forall \varepsilon > 0, \quad (8)$$

where $\partial_\varepsilon f(a) = \{p \in \mathbb{R}^n : \langle p, x - a\rangle - \varepsilon \le f(x) - f(a) \quad \forall x \in \mathbb{R}^n\}$ is the ε-subdifferential of $f(x)$ at point a.

Solution Methods

Numerous solution methods have been proposed for different classes of dc optimization. Each of them proceeds either by outer approximation (OA) of the feasible set or by branch and bound (BB) or is of a hybrid type, combining OA with BB. Following are some typical dc algorithms.

An OA Method for (CDC)

Without losing generality, assume (3), i. e.,

$$\exists w \quad \text{s.t. } g(w) \le 0, \quad f(w) < \min\{f(x) : x \in \Omega, h(x) \ge 0\}. \quad (9)$$

where, as was defined above, $\Omega = \{x : g(x) \le 0\}$. In most cases checking the regularity of a problem is not easy while regularity is needed for the sufficiency of the optimality criteria in Proposition 5. Therefore the method to be presented below only makes use of the necessity part of this proposition and is independent of any regularity assumption.

In practice, what we usually need is not an exact solution but just an approximate solution of the problem. Given tolerances $\varepsilon > 0, \eta > 0$, we are interested in *$\varepsilon$-approximate solutions*, i. e., solutions $x \in \Omega$ satisfying $h(x) \ge -\varepsilon$. An ε-approximate solution x^* is then said to be *η-optimal* if $f(x^*) - \eta \le \min\{f(x) : x \in \Omega, h(x) \ge 0\}$.

With $\bar{x}$ now being a given ε-approximate solution and $\gamma = f(\bar{x}) - \eta$, consider the subproblem

$$\max\{h(x) : x \in \Omega, f(x) \le \gamma\}. \quad (Q_\gamma)$$

For simplicity assume that the set $D(\gamma) = \{x \in \Omega, f(x) \le \gamma\}$ is bounded. Then (Q_γ) is a convex maximization problem over a compact convex set and can be solved by an OA algorithm (see [13] or [3]) generating a sequence $\{x^k, y^k\}$ such that

$$x^k \in \Omega, f(x^k) \le \gamma, h(x^k) \le \max(Q_\gamma) \le h(y^k) \quad (10)$$

and, furthermore, $\|x^k - y^k\| \to 0$ as $k \to +\infty$. These relations imply that we must either have $h(y^k) < 0$ for some k (which implies that $\max(Q_\gamma) < 0$), or else $h(x^k) \ge -\varepsilon$ for some k. In the former case,

this means there is no $x \in \Omega$ with $h(x) \geq 0$ and $f(x) \leq f(\bar{x}) - \eta$, i. e., $\bar{x}$ is η-optimal to CDC, and we are done. In the latter case, x^k is an ε-approximate solution with $f(x^k) \leq f(\bar{x}) - \eta$. Using then a local search (or any inexpensive way available) one can improve x^k to $x' \in \Omega \cap \{x : h(x) = -\varepsilon\}$, and, after resetting $\gamma \leftarrow f(x') - \eta$ in (Q_γ), one can repeat the procedure with the new (Q_γ). And so on.

As is easily seen, the method consists essentially of a number of consecutive cycles in each of which, say the lth cycle, a convex maximization subproblem (Q_γ) is solved with $\gamma = f(x^l) - \eta$ for some ε-approximate solution x^l. This sequence of cycles can be organized into a unified procedure. For this, it suffices to start each new cycle from the result of the previous cycle: after resetting $\gamma \leftarrow \gamma' := f(x') - \eta$ in (Q_γ), we have $D(\gamma') \subset D(\gamma)$, with a point $x' \notin D(\gamma')$, so the algorithm can be continued by using a hyperplane separating x' from $D(\gamma')$ to form with the current polytope outer approximating $D(\gamma)$ a smaller polytope outer approximating $D(\gamma')$. Since each cycle decreases the objective function value by at least a quantity $\eta > 0$, and the objective function is bounded from below, the whole procedure must terminate after finitely many cycles, yielding an ε-approximate solution that is η-optimal to (CDC).

It is also possible to use a BB algorithm for solving the subproblem (Q_γ) in each cycle. The method then proceeds exactly as in the BB method for GDC to be presented next.

A BB Method for General DC Optimization

A general dc optimization problem can be formulated as

$$\min\{f(x) : g_i(x) \geq 0, \quad i = 1, \ldots, m, x \in \Omega\}, \quad \text{(GDC)}$$

where Ω is a compact convex subset of $\mathbb{R}^n$, and $f, g_1, \ldots, g_m$ are dc functions on Ω. Although in principle GDC can be reduced to the canonical form and solved as a CDC problem, this may not be an efficient method as it does not take account of specific features of GDC. For instance, if the feasible set of GDC is highly nonconvex, computing a single feasible solution may be as hard as solving the problem itself. Under these conditions, a direct application of the OA or the BB strategies to GDC is fraught with pitfalls. Without adequate precautions, such approaches may lead to grossly incorrect results or to an unstable solution that may change drastically upon a small change of the data or the tolerances [15,16].

A safer approach is to reduce GDC to a sequence of problems with a convex feasible set in the following way. By simple manipulations it is always possible to arrange that the objective function $f(x)$ is convex. Let $g(x) = \min_{i=1,\ldots,m} g_i(x)$, and for every $\gamma \in \mathbb{R} \cup \{+\infty\}$ consider the subproblem

$$\max\{g(x) : x \in \Omega, f(x) \leq \gamma\}. \quad (R_\gamma)$$

Assuming the set $D(\gamma) := \{x \in \Omega, f(x) \leq \gamma\}$ to be bounded, we have in (R_γ) a dc optimization over a compact convex set. Using a *BB procedure* to solve (R_γ) we generate a nested sequence of partition sets M_k (boxes, e. g., using a rectangular subdivision), together with a sequence $\alpha(M_k) \in \mathbb{R} \cup \{-\infty\}$, and $x^k \in \mathbb{R}^n$, $k = 1, 2, \ldots$, such that

$$\operatorname{diam} M_k \to 0 \text{ as } k \to +\infty, \quad (11)$$

$$\alpha(M_k) \searrow \max\{g(x) : x \in M_k \cap D(\gamma)\} (k \to +\infty), \quad (12)$$

$$\alpha(M_k) \geq \max(R_\gamma), x^k \in M_k \cap D(\gamma), \quad (13)$$

where max(P) denotes, as usual, the optimal value of problem P. Condition (11) means that the subdivision rule used must be exhaustive, while (12) indicates that $\alpha(M_k)$ is an upper bound over the feasible solutions in M_k, and (13) follows from the fact that M_k is the partition set with the largest upper bound among all partition sets currently of interest.

As before, we say that x is an *ε-approximate solution* of GDC if $x \in \Omega$, $g(x) \geq -\varepsilon$ and x^* is η-optimal if $f(x^*) - \eta \leq \min\{f(x) : g(x) \geq 0, x \in \Omega\}$. From (11)–(13) it follows that $\alpha(M_k) - g(x^k) \to 0$ as $k \to +\infty$, and hence, for any given $\varepsilon > 0$, either $\alpha(M_k) < 0$ for some k or $g(x^k) \geq -\varepsilon$ for some k. In the former case, $\max(R_\gamma) < 0$, hence $\max(\text{GDC}) > \gamma$; in the latter case, x^k is an ε-approximate solution of GDC with $f(x^k) \leq \gamma$. So, given any ε-approximate solution $\bar{x}$ with $\gamma = f(\bar{x}) - \eta$, a finite number of iterations of this BB

procedure will help to determine whether there is no feasible solution x to GDC with $f(x) \le f(\bar{x}) - \eta$, i. e., $\bar{x}$ is η-*optimal* to GDC, or else there exists an ε-approximate solution x' to GDC with $f(x') \le f(\bar{x}) - \eta$. In the latter case, we can reset $f(x') - \eta \leftarrow \gamma$ and repeat the procedure with the new γ, and so on. In this way the whole solution process consists of a number of cycles, each involving a finite BB procedure and giving a decrease in the incumbent value of $f(x)$ by at least $\eta > 0$. By starting each cycle right from the result of the previous one, the sequence of cycles forms a unified procedure. Since η is a positive constant, the number of cycles is finite and the procedure terminates with an ε-approximate solution that is η-optimal to GDC.

The efficiency of such a BB procedure depends on two basic operations: branching and bounding. Usually, branching is performed by means of an exhaustive subdivision rule, so as to satisfy condition (11). For rectangular partition, this condition can be achieved by the standard bisection rule: bisect the current box M into two equal subboxes by means of a hyperplane perpendicular to a longest edge of M at its midpoint. However, it has been observed that the convergence guaranteed by an exhaustive subdivision rule is rather slow, especially in high dimensions. To improve the situation, the idea is to use, instead of the standard bisection, an *adaptive subdivision rule* defined as follows. Let the upper bound $\alpha(M_k)$ in (12) be obtained as $\alpha(M_k) = \max\{\Gamma(x) \colon x \in M_k \cap D(\gamma)\}$, where $\Gamma(x)$ is some concave overestimator of $g(x)$ over M_k that is tight at some point $y^k \in M_k$, i. e., satisfies $\Gamma(y^k) = g(y^k)$. If $x^k \in \operatorname{argmax}\{\Gamma(x) | x \in M_k \cap D(\gamma)\}$, then the subdivision rule is to bisect M_k by means of the hyperplane $x_s = x_s^k + y_s^k/2$, where $s \in \operatorname{argmax}_{i=1,\dots,n} |y_i^k - x_i^k|$. As has been proved in [13], such an adaptive bisection rule ensures the existence of an infinite subsequence $\{k_\nu\}$ such that $y^{k_\nu} - x^{k_\nu} \to 0$ as $\nu \to +\infty$. The common limit x^* of x^{k_ν} and y^{k_ν} then yields an optimal solution of the problem (R_γ). Computational experience has effectively confirmed that convergence achieved with an adaptive subdivision rule is usually much faster than with the standard bisection. For such an adaptive subdivision to be possible, the constraint set $D(\gamma)$ of (12) must be convex, so that for each partition set M_k two points $x^k \in M_k \cap D(\gamma)$ and $y^k \in M_k$ can be defined such that $\alpha(M_k) - g(y^k) = o(\|x^k - y^k\|)$.

DCA–A Local Optimization Approach to (DC)

By rewriting DC as a canonical dc optimization problem

$$\min\{t - h(x) \colon x \in \mathbb{R}^n, t \in \mathbb{R}, g(x) - t \le 0\},$$

we see that DC can be solved by the same method as CDC. Since, however, for some large-scale problems we are not so much interested in a global optimal solution as in a sufficiently good feasible solution, a local optimization approach to DC has been developed [9] that seems to perform quite satisfactorily in a number of applications. This method, referred to as DCA, is based on the well-known Toland equality:

$$\inf_{x \in \operatorname{dom} g} \{g(x) - h(x)\} = \inf_{y \in \operatorname{dom} h^*} \{g^*(y) - g^*(y)\}, \quad (14)$$

where $g, h \colon \mathbb{R}^n \to \mathbb{R}$ are lower semicontinuous proper convex functions, and the star denotes the conjugate, e. g., $g^*(y) = \sup\{\langle x, y\rangle - g(x) \colon x \in \operatorname{dom} g\}$. Taking account of this equality, DCA starts with $x^0 \in \operatorname{dom} g$ and for $k = 1, 2, \dots$, computes $y^k \in \partial h(x^k); x^{k+1} \in \partial g^*(y^k)$. As has been proved in [9], the thus generated sequence x^k, y^k satisfies the following conditions:

1. The sequences $g(x^k) - h(x^k)$ and $h^*(x^k) - g(^*(x^k)$ are decreasing.
2. Every accumulation point x^* (resp. y^*) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of the function $g(x) - h(x)$ (resp. $h^*(y) - g^*(y)$).

Though global optimality cannot be guaranteed by this method, it has been observed that in many cases of interest it yields a local minimizer that is also global.

Applications and Extensions

The above described dc methods are of a general-purpose type. For many special dc problems more efficient algorithms are needed to take full advantage of additional structures. Along this line, dc methods have been adapted to solve problems with separated nonconvexity, bilinear programming, multilevel programming, multiobjective programming, optimization problems over efficient sets, polynomial and synomial programming, fractional programming, continuous location problems, clustering and datamining problems, etc. [4]. In particular, quite efficient methods have been developed for a class of dc optimization problems important for applications called multiplicative program-

ming [4,5]. Also, techniques for bounding, branching, and decomposition have been refined that have very much widened the range of applicability of dc methods. Most recently, *monotonic optimization*, also called *DM optimization*, has emerged as a new promising field of research dealing with a class of optimization problems important for applications whose structure, though different from the dc structure, shares many common features with the latter. To be specific, let C be a family of real valued functions on $\mathbb{R}^n$ such that (i) $g_1, g_2 \in C, \alpha_1, \alpha_2 \in \mathbb{R}_+ \Rightarrow \alpha_1 g_1 + \alpha_2 g_2 \in C$; (ii) $g_1, g_2 \in C \Rightarrow g(x) := \max\{g_1(x), g_2(x)\} \in C$. Then the family $\mathcal{D}(C) = C - C$ is a vector lattice with respect to the two operations of pointwise maximum and pointwise minimum. When C is the set of convex functions, $\mathcal{D}(C)$ is nothing but the vector lattice of dc functions. When C is the set of increasing functions on $\mathbb{R}^n$, i. e., the set of functions $f: \mathbb{R}^n \to \mathbb{R}$ such that $x' \geq x \Rightarrow f(x') \geq f(x)$, the vector lattice $\mathcal{D}(C)$ consists of *DM functions*, i. e., functions representable as the difference of two increasing functions. For the theory, methods, and algorihms of DM optimization, we refer the reader to [7,14,18].

References

1. Floudas CA (2000) Deterministic Global Optimization. Kluwer, Dordrecht
2. Hartman P (1959) On Functions Representable as a Difference of Convex Functions. Pacific J Math 9:707–713
3. Horst R, Tuy H (1996) Global Optimization (Deterministic Approaches), 3rd edn. Springer, Berlin
4. Horst R, Pardalos PM (eds) (1995) Handbook of Global Optimization. Kluwer, Dordrecht
5. Konno H, Thach PT, Tuy H (1997) Optimization on Low Rank Nonconvex Structures. Kluwer, Dordrecht
6. Pardalos PM, Rosen JB (1987) Constrained Global Optimization: Algorithms and Applications. Lecture Notes in Computer Sciences 268. Springer, Berlin
7. Rubinov A (1999) Abstract Convexity and Global Optimization. Kluwer, Dordrecht
8. Sherali HD, Adams WP (1999) A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Kluwer, Dordrecht
9. Tao PD, An LTH (1997) Convex analysis approach to D.C. Programmng: Theory, algorithms and applications. Acta Mathematica Vietnamica 22:289–356
10. Thach PT (1993) D.c. sets, dc functions and nonlinear equations. Math Programm 58:415–428
11. Tuy H (1964) Concave programming under linear constraints. Soviet Math 5:1437–1440
12. Tuy H (1985) A general deterministic approach to global optimization via dc programming. In: Hiriart-Urruty JB (ed) Fermat Days 1985: Mathematics for Optimization. North-Holland, Amsterdam, pp 137–162
13. Tuy H (1998) Convex Analysis and Global Optimization. Kluwer, Dordrecht
14. Tuy H (2000) Monotonic Optimization: Problems and Solution Approaches. SIAM J Optim 11(2):464–494
15. Tuy H (2005) Robust Solution of Nonconvex Global Optimization Problems. J Global Optim 32:307–323
16. Tuy H (2005) Polynomial Optimization: A Robust Approach. Pacific J Optim 1:357–373
17. Tuy H, Al-Khayyal FA, Thach PT (2005) Monotonic Optimization: Branch and Cuts Methods. In: Audet C, Hansen P, Savard G (eds) Essays and Surveys on Global Optimization. GERAD. Springer, Berlin, pp 39–78
18. Tuy H, Minoux M, NTH Phuong (2006) Discrete Monotonic Optimization with Application to A Discrete Location Problem. SIAM J Optim 1778–97

Decision Support Systems with Multiple Criteria

CONSTANTIN ZOPOUNIDIS, MICHAEL DOUMPOS
Department Production Engineering and Management Financial Engineering Lab. Techn., University Crete, Chania, Greece

MSC2000: 90C29

Article Outline

Keywords
Multicriteria Decision Aid
Multicriteria Decision Support Systems
  Multicriteria Group Decision Support Systems
  Intelligent Multicriteria Decision Support Systems
Conclusions
See also
References

Keywords

Decision support system; Multicriteria analysis; Multicriteria group decision support system; Intelligent multicriteria decision support systems

In practical real-world situations the available time for making decisions is often limited, while the cost of investigation is increasing with time. Therefore, it would

be enviable to exploit the increasing processing power provided by the modern computer technology, to save significant amounts of time and cost in decision making problems. Computationally intensive, but routine tasks, such as data management and calculations can be performed with remarkable speed by a common personal computer, compared to the time that a human would need to perform the same tasks. On the other hand, computers are unable to perform cognitive tasks, while their inference and reasoning capabilities are still very limited compared to the capabilities of the human brain. Thus, in decision making problems, computers can support decision makers by managing the data of the problem and performing computationally intensive calculations, based on a selected decision model, which could help in the analysis, while the decision makers themselves have to examine the obtained results of the models and conclude to the most appropriate decision.

This merging of human judgment and intuition together with computer systems constitutes the underlying philosophy, methodological framework and basic goal of *decision support systems* [17]. The term 'decision support system' (DSS) is already consolidated and it is used to describe any computer system that provides information on a specific decision problem using analytical decision models and access to databases, in order to support a decision maker in making decisions effectively in complex and ill-structured problems where no straightforward, algorithmic procedure can be employed [28].

The development of DSSs kept pace with the advances in computer and information technologies, and since the 1970s numerous DSSs have been designed by academic researchers and practitioners for the examination and analysis of several decision problems including finance and accounting, production management, marketing, transportation, human resources management, agriculture, education, etc. [17,19].

Except for the specific decision problems that DSSs address, these systems are also characterized by the type of decision models and techniques that they incorporate (i. e. statistical analysis tools, mathematical programming and optimization techniques, multicriteria decision aid methods, etc.). Some of these methodologies (optimization, statistical analysis, etc.) which have already been implemented in several DSSs, are based on the classical monocriterion approach. However, real world decision problems can be hardly considered through the examination of a single criterion, attribute or point of view that will lead to the 'optimum' decision. In fact such a monocriterion approach is merely an oversimplification of the actual nature of the problem at hand, that can lead into unrealistic decisions.

On the other hand, a more realistic and flexible approach would be the simultaneous consideration of all pertinent factors that may affect a decision. However, through this appealing approach a very essential issue emerges: how can several and often conflicting factors can be aggregated to make rational decisions? This issue constitutes the focal point of interest for all the *multicriteria decision aid* methods. The incorporation of multicriteria decision aid methods in DSSs provides the decision makers with a highly efficient tool to study complex real world decision problems where multiple criteria of conflicting nature are involved. Therefore, the subsequent sections of this paper will concentrate on this specific category of DSSs (*multicriteria DSSs*, MCDSSs).

The article is organized as follows. In section 2 some basic concepts, notions and principles of multicriteria decision aid are discussed. Section 3 presents the main features and characteristics of MCDSSs, along with a review of the research that has been conducted in this field, while some extensions of the classical MCDSSs framework in group decision making and intelligent decision support are also discussed. Finally, section 4 concludes the paper and outlines some possible future research directions in the design, development and implementation of MCDSSs.

Multicriteria Decision Aid

Multicriteria decision aid (MCDA, the European School) or multicriteria decision making (MCDM, the American School) [49,64] constitutes an advanced field of operations research which is devoted to the development and implementation of decision support methodologies to confront complex decision problems involving multiple criteria, goals or objectives of conflicting nature. The foundations of MCDA can be traced back in the works of J. von Neumann and O. Morgenstern [43], and P.C. Fishburn [20] on utility theory, A. Charnes and W.W. Cooper [10] on goal program-

ming, and B. Roy [47] on the concept of outranking relations and the foundations of the ELECTRE methods. These pioneering works have affected the subsequent research in the field of MCDA that can be distinguished in two major groups: discrete and continuous MCDA. The former is involved with decision problems where there is a finite set of alternatives which should be considered in order to select the most appropriate one, to rank them from the best to the worst, or to classify them in predefined homogeneous classes. On the contrary in continuous MCDA problems the alternatives are not defined a priori, but instead one seeks to construct an alternative that meets his/her goals or objectives (for instance the construction of a portfolio of stocks).

There are different ways to address these two classes of problems in MCDA. Usually, a continuous MCDA problem is addressed through *multi-objective* or *goal programming* approaches. In the former case, the objectives of the decision maker are expressed as a set of linear or non linear functions which have to be 'optimized', whereas in the latter case the decision maker expresses his/her goals in the form of a reference or ideal point which should be achieved as close as possible. These two approaches extend the classical single-objective optimization framework, through the simultaneous consideration of more than one objectives or goals. Of course in this new context it seems illusory to speak of optimality, but instead the aim is initially to determine the set of efficient solutions (solutions which are not dominated by any other solution) and then to identify interactively a specific solution which is consistent with the preference structure of the decision maker. The books [54,57] and [63] provide an excellent and extensive discussion of both multi-objective and goal programming.

On the other hand, discrete MCDA problems are usually addressed through the *multi-attribute utility theory* (MAUT) [26], the *outranking relations approach* [48] or the *preference disaggregation approach* ([23,44]). These three approaches are mainly focused on the determination and modeling of the decision makers' preferences, in order to develop a global preference model which can be used in decision making. Their differences concern mainly the form of the global preference model that is developed, as well as the procedure that is used to estimate the parameters of the model. The developed preference model in both MAUT and preference disaggregation is a utility or value function either additive or multiplicative, whereas the outranking relations approach is based on pairwise comparisons of the form 'alternative a is at least as good as alternative b'. Concerning the procedure that is used to estimate the parameters of the global preference model, both in MAUT and outranking relations there is a direct interrogation of the decision maker. More precisely, in MAUT the decision maker is asked to determine the trade-offs among the several attributes or criteria, while in outranking relations the decision maker has to determine several parameters, such as the weights of the evaluation criteria, indifference, strict preference and veto thresholds for each criterion. On the contrary, in preference disaggregation, an ordinal regression procedure is used to estimate the global preference model. Based on a reference set of alternatives, which may consist either of past decisions or by a small subset of the alternatives under consideration, the decision maker is asked to provide a ranking or a classification of the alternatives according to his/her decision policy (global preferences). Then, using an ordinal regression procedure the global preference model is estimated so that the original ranking or classification (and consequently the global preference system of the decision maker) can be reproduced as consistently as possible.

Multicriteria Decision Support Systems

From the above brief discussion of the basic concepts and approaches of MCDA, it is clear that in any case the decision maker and his/her preferences constitute the focal point of the methodological framework of MCDA. This special characteristic of MCDA implies that a comprehensive model of a decision situation cannot be developed, but instead the model should be developed to meet the requirements of the decision maker [46]. The development of such a model can be only achieved through an iterative and interactive process, until the decision maker's preferences are consistently represented in the model. Both interactivity and iterative operation are two of the key characteristics of DSSs. Consequently, a DSS incorporating MCDA methods could provide essential support in structuring the decision problem, analyzing the preferences of the decision maker, and supporting the model building process.

The support provided by multicriteria DSSs (MCDSSs) is essential for the decision maker as well as for the decision analyst.

- The decision maker through the use of MCDSSs becomes familiar with sophisticated operations research techniques, he is supported in structuring the decision problem considering all possible points of view, attributes or criteria, and furthermore, he is able to analyze the conflicts between these points of view and consider the existing trade-offs. All these capabilities provided by MCDSSs serve the learning process of decision makers in resolving complex decision problems in a realistic context, and constitute a solid scientific basis for arguing upon the decisions taken.
- On the other hand, from the decision analyst point of view, MCDSSs provide a supportive tool which is necessary throughout the decision making process, enabling the decision analyst who usually acts as an intermediate between the system and the decision maker, to highlight the essential features of the problem to the decision maker, to introduce the preferences of the decision maker in the system, and to develop the corresponding model. Furthermore, through sensitivity and robustness analyses the decision analyst is able to examine several scenarios, concerning both the significance of the evaluation criteria as well as the changes in the decision environment.

The supportive operation of MCDSSs in making decisions in ill-structured complex decision problems was the basic motivation for computer scientists, management scientists and operations researchers in the development of such systems. Actually, MCDSSs are one of the major areas of DSSs research since the 1970s [19] and significant progress has been made both on the theoretical and the practical/implementation viewpoints.

The first MCDSSs to be developed in the 1970s where mainly oriented towards the study of multiobjective mathematical programming problems ([16,61]). These early pioneer systems, mainly due to the limited capabilities of computer technology during that period, were primarily developed for academic purposes, they were implemented in mainframe computers, with no documentation available, while they had no visual representation capabilities [31]. Today, after more than twenty years of research and advances in MCDA, DSSs, and computer science, most MCDSSs provide many advanced capabilities to decision makers including among others [46]:

1) Enhanced data management capabilities including interactive addition, deletion or modification of criteria.
2) Assessment and management of weights.
3) User-friendly interfaces based on visual representations of both alternatives and criteria to assist the interaction between the system and the decision maker.
4) Sensitivity analysis (what-if analysis) to determine how the changes in the weights of the evaluation criteria can affect the actual decision.

These capabilities are in accordance with the general characteristics of DSSs, that is interactivity, flexibility and adaptability to the changes of the decision environment, user oriented design and development, and combination of data base management with decision models. Although the aforementioned capabilities are common to most of the existing MCDSSs, one could provide a distinction of the MCDSSs according to the MCDA approaches that they employ:

- MCDSSs based on the multi-objective programming approach:
  - the TOMMIX system [2],
  - the TRIMAP system [11],
  - the VIG system ([29,32]),
  - the VIDMA system [30],
  - the DIDAS system [36],
  - the AIM system [37],
  - the ADBASE system [58], and
  - the STRANGE system [59].
- MCDSSs based on the MAUT approach:
  - the MACBETH system [5],
  - the VISA system [6], and
  - the EXPERT CHOICE system [21].
- MCDSSs based on the outranking relations approach:
  - the PROMCALC and GAIA systems [7],
  - the ELECCALC system [27],
  - the PRIAM system [34], and
  - the ELECTRE TRI system [62].
- MCDSSs based on the preference disaggregation approach:
  - the PEFCALC system [22],
  - the MINORA system [51],

– the MIIDAS system [52], and
– the PREFDIS system [66].

Most of the existing MCDSSs are designed for the study of general multicriteria decision problems. Although they provide advanced capabilities for modeling the decision makers' preferences in order to make a specific decision regarding the choice of an alternative and the ranking or the classification of the alternatives, MCDSSs do not consider the specific characteristics, as well as the nature of the decision that should be taken according to the specific decision problem that is considered.

To address the unique nature of some significant decision problems, where except for the application of MCDA methodology, some other type of analyses are necessary to consider the environment in which the decision is taken, several authors proposed domain specific MCDSSs. Some decision problems for which specific MCDSSs have been developed include the assessment of corporate performance and viability (the BANKADVISER system [39], the FINCLAS system [65], the FINEVA system [68], and the system proposed in [53]), bank evaluation (the BANKS system [40]), bank asset liability management [33], financial planning [18], portfolio selection [67], new product design (the MARKEX system [42]), urban planning (the system proposed in [1]), strategic planning [9], and computer system design [15].

Multicriteria Group Decision Support Systems

A common characteristic of all the aforementioned MCDSSs is that they refer to decisions that are taken by individual decision makers. However, in many cases the actual decision is not the responsibility of an individual, but instead there is a team of negotiating or cooperative participants who must conclude to a consensus decision. In this case, although the decision process and consequently the required decision support, remains the same, as far as each individual decision maker is concerned, the process that will lead the cooperative team or the negotiating parties to a consensus decision is completely different from the individual decision making process. Therefore, the type of support needed also differs.

Group DSSs (GDSSs) aim at supporting such decision processes, and since the tools provided by MCDA can be extended to generalized group decision process, several attempts have been made to design and develop such multicriteria systems. Some examples of *multicriteria GDSSs* include the Co-oP system [8], the JUDGES system [12], the WINGDSS system [13], the MEDIATOR system [24], and the SCDAS system [35].

Intelligent Multicriteria Decision Support Systems

Except for the extension of the MCDSSs framework in supporting group decision making, recently researchers have also investigated the extension of MCDSSs through the exploitation of the advances in the field of artificial intelligence. Scientific fields such as those of neural networks, expert systems, fuzzy sets, genetic algorithms, etc., provide promising features and new capabilities regarding the representation of expert knowledge, the development of intelligent and more friendly user interfaces, the reasoning and explanation abilities, as well as the handling of incomplete, uncertain and imprecise information.

These appealing new capabilities provided by artificial intelligence techniques can be incorporated in the existing MCDSSs framework to provide expert advice on the problem under consideration, assistance to the use of the several modules of the system, explanations concerning the results MCDA, models, support on structuring the decision making process, as well as recommendations and further guidance for the future actions that the decision maker should take in order to implement successfully his/her decisions. The terms '*intelligent multicriteria decision support systems*' or 'knowledge-based multicriteria decision support systems' have been used by several authors to describe MCDSSs which take advantage of artificial intelligence techniques in combination with MCDA methods.

Some representative examples of intelligent MCDSSs are, the system proposed in [3] for multiobjective linear programming, the MARKEX system for new product design [42], the CREDEX system [45] and the CGX system [55] for credit granting problems, the MIIDAS system for estimating additive utility functions based on the preference disaggregation approach [52], the INVEX system for investment analysis [60] based on the PROMETHEE method, as well as the FINEVA system [68] for the assessment of corporate performance and viability. All these systems incor-

porate in their structure one or more expert system components either to derive estimations regarding the problem under consideration (FINEVA, MARKEX, CREDEX, CGX, INVEX systems) or to support the use of the MCDA models which are incorporated in the system and generally support and improve the communication between the user and the system (MIIDAS and MARKEX systems). Furthermore, the INVEX system incorporates fuzzy sets to provide an initial distinction between good and bad investment projects, so that the number of alternatives to be considered latter on in the multicriteria analysis module is reduced.

The ongoing research on the integration of artificial intelligence with MCDA regarding the theoretical foundations of this integration and the related implementation issues ([4,25]), the construction of fuzzy outranking relations ([14,41,50]), and the applications of neural networks in preference modeling and utility assessment ([38,56]) constitutes a significant basis for the design and development of intelligent MCDSSs implementing the theoretical findings of this research.

Conclusions

This article investigated the potentials provided by MCDSSs in the decision making process. MCDSSs during the last two decades have consolidated their position within the operations research, information systems and management science communities as an efficient tool for supporting the whole decision making process beginning from problem structuring until the implementation of the final decision, in complex ill-structured problems.

The review which was presented in this paper reveals that recent advances in MCDSSs include systems for general use to solve both discrete and continuous MCDA problems, systems designed to study some specific real world decisions, as well as systems designed to support negotiation and group decision making.

As the computer science and technology progresses rapidly, new areas of applications of MCDSSs can be explored including their operation over the Internet to provide computer support to co-operative work of dispersed and asynchronous decision units. The incorporation of artificial intelligence techniques in the existing framework of MCDSSs also constitutes another significant area of future research. Although, as its has been illustrated in this paper, researchers have already tried to integrate these two approach in an integrated intelligent system, there is a lot of work to be done in order to take the most out of the capabilities of neural networks, fuzzy sets and expert systems to provide user-friendly support in decision problems where multiple criteria are involved.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Anselin L, Arias EG (1983) A multi-criteria framework as decision support system for urban growth management applications: Central city redevelopment. Europ J Oper Res 13:300–309
2. Antunes CH, Alves MJ, Silva AL, Climaco J (1992) An integrated MOLP method base package-A guided tour of TOMMIX. Comput Oper Res 1(4):609–625
3. Antunes CH, Melo MP, Climaco JN (1992) On the integration of an interactive MOLP procedure base and expert system techniques. Europ J Oper Res 61:135–144

4. Balestra G, Tsoukiàs A (1990) Multicriteria analysis represented by artificial intelligence techniques. J Oper Res Soc 41(5):419–430
5. Bana e Costa CA, Vansnick JC (1994) MACBETH-An interactive path towards the construction of cardinal value functions. Internat Trans Oper Res 1:489–500
6. Belton V, Vickers SP (1989) V.I.S.A.-VIM for MCDA. In: Lockett AG, Islei G (eds) Improving Decision Making in Organizations. Springer, Berlin, pp 319–334
7. Brans JP, Mareschal B (1994) The PROMCALC and GAIA decision support system for multicriteria decision aid. Decision Support Systems 12:297–310
8. Bui T (1994) Software architectures for negotiation support: Co-oP and Negotiator. Computer-Assisted Negotiation and Mediation Symposium, Program of Negotiation (May 26-27 1994), Harvard Law School, Cambridge, MA, pp 216–227
9. Chandrasekaran G, Ramesh R (1987) Microcomputer based multiple criteria decision support system for strategic planning. Inform and Management 12:163–172
10. Charnes A, Cooper WW (1961) Managem. models and industrial applications of linear programming. Wiley, New York
11. Climaco J, Antunes CH (1989) Implementation of a user friendly software package-A guided tour of TRIMAP. Math Comput Modelling 12(10–11):1299–1309
12. Colson G, Mareschal B (1994) JUDGES: A descriptive group decision support system for the ranking of items. Decision Support Systems 12:391–404
13. Csaki P, Rapcsak T, Turchanyi P, Vermes M (1995) R and D for group decision aid in Hungary by WINGDSS, a Microsoft Windows based group decision support system. Decision Support Systems 14:205–217
14. Czyzak P, Slowinski R (1996) Possibilistic construction of fuzzy outranking relation for multiple-criteria ranking. Fuzzy Sets and Systems 81:123–131
15. Dutta A, Jain HK (1985) A DSS for distributed computer system design in the presence of multiple conflicting objectives. Decision Support Systems 1:233–246
16. Dyer J (1973) A time-sharing computer program for the solution of the multiple criteria problem. Managem Sci 19:1379–1383
17. Eom HB, Lee SM (1990) Decision support systems applications research: A bibliography 1971–1988. Europ J Oper Res 46:333–342
18. Eom HB, Lee SM, Snyder CA, Ford FN (1987/8) A multiple criteria decision support system for global financial planning. J Management Information Systems 4(3):94–113
19. Eom SB, Lee SM, Kim JK (1993) The intellectual structure of decision support systems: 1971–1989. Decision Support Systems 10:19–35
20. Fishburn PC (1965) Independence in utility theory with whole product sets. Oper Res 13:28–45
21. Forman EH, Selly MA (2000) Decisions by objectives: How to convince others that you are right. World Sci., Singapore
22. Jacquet-Lagrèze E (1990) Interactive assessment of preferences using holistic judgments: The PREFCALC system. In: Bana e Costa CA (ed) Readings in Multiple Criteria Decision Making. Springer, Berlin, pp 335–350
23. Jacquet-Lagrèze E, Siskos J (1982) Assessing a set of additive utility functions for multicriteria decision-making: The UTA method. Europ J Oper Res 10:151–164
24. Jarke M, Jelassi MT, Shakun MF (1987) MEDIATOR: Toward a negotiation support system. Europ J Oper Res 31:314–334
25. Jelassi MT (1987) MCDM: From stand-alone methods to integrated and intelligent DSS. In: Sawaragi Y, Inoue K, Nakayama H (eds) Towards Interactive and Intelligent Decision Support Systems. Springer, Berlin, pp 575–584
26. Keeney RL, Raiffa H (1976) Decisions with multiple objectives: Preferences and value trade-offs. Wiley, New York
27. Kiss LN, Martel JM, Nadeau R (1994) ELECCALC-An interactive software for modelling the decision maker's preferences. Decision Support Systems 12:311–326
28. Klein MR, Methlie LB (1995) Knowledge based decision support systems with application in business. Wiley, New York
29. Korhonen P (1987) VIG-A visual interactive support system for multiple criteria decision making. Belgian J Oper Res Statist Computer Sci 27:3–15
30. Korhonen P (1988) A visual reference direction approach to solving discrete multiple criteria problems. Europ J Oper Res 34:152–159
31. Korhonen P, Moskowitz H, Wallenius J (1992) Multiple criteria decision support-A review. Europ J Oper Res 63:361–375
32. Korhonen P, Wallenius J (1988) A Pareto race. Naval Res Logist 35:615–623
33. Langen D (1989) An (interactive) decision support system for bank asset liability management. Decision Support Systems 5:389–401
34. Levine P, Pomerol JCh (1986) PRIAM, an interactive program for chosing among multiple attribute alternatives. Europ J Oper Res 25:272–280
35. Lewandowski A (1989) SCDAS-Decision support system for group decision making: Decision theoretic framework. Decision Support Systems 5:403–423
36. Lewandowski A, Kreglewski T, Rogowski T, Wierzbicki A (1989) Decision support systems of DIDAS family (Dynamic Interactive Decision Analysis & Support. In: Lewandowski A and Wierzbicki A (eds) Aspiration Based Decision Support Systems. Springer, Berlin, pp 21–27
37. Lofti V, Stewart TJ, Zionts S (1992) An aspiration-level interactive model for multiple criteria decision making. Comput Oper Res 19:677–681
38. Malakooti B, Zhou YQ (1994) Feedforward artificial neural networks for solving discrete multiple criteria decision making problems. Managem Sci 40(11):1542–1561
39. Mareschal B, Brans JP (1991) BANKADVISER: An industrial evaluation system. Europ J Oper Res 54:318–324

40. Mareschal B, Mertens D (1992) BANKS a multicriteria, PROMETHEE-based decision support system for the evaluation of the international banking sector. Revue des Systèmes de Décision 1(2):175–189
41. Martel JM, D'Avignon CR, Couillard J (1986) A fuzzy outranking relation in multicriteria decision making. Europ J Oper Res 25:258–271
42. Matsatsinis NF, Siskos Y (1999) MARKEX: An intelligent decision support system for product development decisions. Europ J Oper Res 113:336–354
43. Neumann J Von, Morgenstern O (1944) Theory of games and economic behavior. Princeton Univ. Press, Princeton
44. Pardalos PM, Siskos Y, Zopounidis C (1995) Advances in multicriteria analysis. Kluwer, Dordrecht
45. Pinson S (1992) A multi-expert architecture for credit risk assessment: The CREDEX system. In: O'Leary DE, Watkins PR (eds) Expert Systems in Finance. North-Holland, Amsterdam, pp 27–64
46. Pomerol JCh (1993) Multicriteria DSSs: State of the art and problems. Central Europ J Oper Res Econ 3(2):197–211
47. Roy B (1968) Classement et choix en présence de points de vue multiples: La méthode ELECTRE. RIRO 8:57–75
48. Roy B (1991) The outranking approach and the foundations of ELECTRE methods. Theory and Decision 31:49–73
49. Roy B, Vanderpooten D (1997) An overview on the European school of MCDA: Emergence, basic features and current works. Europ J Oper Res 99:26–27
50. Siskos J (1982) A way to deal with fuzzy preferences in multiple-criteria decision problems. Europ J Oper Res 10:614–324
51. Siskos Y, Spiridakos A, Yannacopoulos D (1993) MINORA: A multicriteria decision aiding system for discrete alternatives. J Inf Sci Techn 2:136–149
52. Siskos Y, Spiridakos A, Yannacopoulos D (1999) Using artificial intelligence and visual techniques into preference disaggregation analysis: The MIIDAS system. Europ J Oper Res 113:281–299
53. Siskos Y, Zopounidis C, Pouliezos A (1994) An integrated DSS for financing firms by an industrial development bank in Greece. Decision Support Systems 12:151–168
54. Spronk J (1981) Interactive multiple goal programming application to financial planning. Martinus Nijhoff, Boston, MA
55. Srinivasan V, Ruparel B (1990) CGX: An expert support system for credit granting. Europ J Oper Res 45:293–308
56. Stam A, Sun M, Haines M (1996) Artificial neural network representations for hierarchical preference structures. Comput Oper Res 23(12):1191–1201
57. Steuer RE (1986) Multiple criteria optimization: Theory, computation and application. Wiley, New York
58. Steuer RE (1992) Manual for the ADBASE multiple objective linear programming package. Dept Management Sci and Inform. Technol. Univ. Georgia, Athens, GA
59. Teghem J, Dufrane D, Thauvoye M, Kunsch P (1986) STRANGE: An interactive method for multi-objective linear programming under uncertainty. Europ J Oper Res 26:65–82
60. Vranes S, Stanojevic M, Stevanovic V, Lucin M (1996) INVEX: Investment advisory expert system. Expert Systems 13, no 2:105–119
61. Wallenius J, Zionts S (1976) Some tests of an interactive programming method for multicriteria optimization and an attempt at implementation. In: Thiriez H, Zionts S (eds) Multiple Criteria Decision Making. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 319–331
62. Yu W (1992) ELECTRE TRI: Aspects methodologiques et manuel d'utilisation. Document du Lamsade (Univ Paris-Dauphine) 74
63. Zeleny M (1982) Multiple criteria decision making. McGraw-Hill, New York
64. Zopounidis C (1997) The European school of MCDA: Some recent trends. In: Climaco J (ed) Multicriteria Analysis. Springer, Berlin, pp 608–616
65. Zopounidis C, Doumpos M (1998) Developing a multicriteria decision support system for financial classification problems: The FINCLAS system. Optim Methods Softw 8(3-4)
66. Zopounidis C, Doumpos M (2000) PREFDIS: A multicriteria decision support system for sorting decision problems. Comput Oper Res 27:779–797
67. Zopounidis C, Godefroid M, Hurson Ch (1995) Designing a multicriteria DSS for portfolio selection and management. In: Janssen J, Skiadas CH, Zopounidis C (eds) Advances in Stochastic Modeling and Data Analysis. Kluwer, Dordrecht, pp 261–292
68. Zopounidis C, Matsatsinis NF, Doumpos M (1996) Developing a multicriteria knowledge-based decision support system for the assessment of corporate performance and viability: The FINEVA system. Fuzzy Economic Rev 1(2):35–53

Decomposition Algorithms for the Solution of Multistage Mean-Variance Optimization Problems

Panos Parpas, Berç Rustem
Department of Computing, Imperial College, London, UK

MSC2000: 90C15, 90C90

Article Outline

Abstract
Background
Problem Statement

Methods
Nested Benders Decomposition (NBD)
Augmented Lagrangian Decomposition (ALD)
Numerical Experiments
References

Abstract

Stochastic multistage mean-variance optimization problems represent one of the most frequently used modeling tools for planning problems, especially financial. Decomposition algorithms represent a powerful tool for the solution of problems belonging to this class. The first aim of this article is to introduce multi-stage mean-variance models, explain their applications and structure. The second aim is the discussion of efficient solution methods of such problems using decomposition algorithms.

Background

Stochastic programming (SP) is becoming an increasingly popular tool for modeling decisions under uncertainty because of the flexible way uncertain events can be modeled, and real-world constraints can be imposed with relative ease. SP also injects robustness to the optimization process. Consider the following standard "deterministic" quadratic program:

$$\begin{aligned} \min_x \quad & \frac{1}{2}x'Hx + c'x \\ s.t \quad & Ax = b \\ & x^l \le x \le x^u . \end{aligned} \tag{1}$$

It is not always possible to know the exact values of the problem data of (1) given by H, A, c, and b. Instead, we may have some estimations in the form of data gathered either empirically or known to be approximated well by a probability distribution. The SP framework allows us to solve problems where the data of the problem are represented as functions of the randomness, yielding results that are more robust to deviations.

The power and flexibility of SP does, however, come at a cost. Realistic models include many possible events distributed across several periods, and the end result is a large-scale optimization problem with hundreds of thousands of variables and constraints. Models of this scale cannot be handled by general-purpose optimization algorithms, so special-purpose algorithms attempt to take advantage of the specific structure of SP models. We examine two decomposition algorithms that had encouraging results reported in linear SP; the first is based on the regularized version of Benders decomposition developed by [21], and the second on an augmented-lagrangian-based scheme developed by [4].

Others [9,24,27] formulated multistage SP as a problem in optimal control, where the current stage variables depend on the parent node variables, and used techniques from optimal control theory to solve the resulting problem. Another related method is the approximation algorithm by [11] where a sequence of scenario trees is generated whose solution produces lower and upper bounds on the solution of the true problem. Decomposition algorithms are not, however, the only approach to tackle the state explosion from which SPs suffer; approximation algorithms and stochastic methods are just two examples of other methods where research is very active [5]. In this study, we are concerned only with decomposition methods.

Problem Statement

We consider a quadratic multistage SP. In the linear case, SP was first proposed independently by [10] and [1]; for a more recent description see [7] and [13]. For two stages, the problem is:

$$\min_x \quad \frac{1}{2}x'Hx + c'x + \mathcal{Q}(x) \tag{2a}$$

$$s.t \quad Ax = b \tag{2b}$$

$$x^l \le x \le x^u . \tag{2c}$$

We use $'$ to denote the transpose of a vector or a matrix. c and $x^{u,l}$ are known vectors in $\Re^{n_1}$. Let A and H be known matrices in $\Re^{m_1 \times n_1}$ and $\Re^{n_1 \times n_1}$. These quantities represent the state of the world that is known. We assume that H is positive semidefinite. The first two terms in the objective function (2a) model the goals of the decision maker that do not depend on uncertain events. $\mathcal{Q}(x)$ represents the expected value of the second-stage objective function:

$$\mathcal{Q}(x) = E_\xi[Q(x, \xi(\omega))] ,$$

where

$$Q(x, \xi(\omega)) = \min_y \frac{1}{2}\alpha y'(\omega)H(\omega)y(\omega) - (1-\alpha)c'(\omega)y(\omega) \tag{3}$$

$$s.t \quad W(\omega)y(\omega) = h(\omega) - T(\omega)x \tag{4}$$

$$y(\omega)^l \le y(\omega) \le y(\omega)^u . \tag{5}$$

Let Ω be the set of all random events, and $\omega \in \Omega$ be the particular realization of an event so that when ω is known the random events are aggregated in the vector $\xi(\omega) = [y(\omega), H(\omega), W(\omega), h(\omega), T(\omega), y^{u,l}(\omega)]$, and let Ξ be the support of ξ. The uncertainty of the second stage is represented by the random data $H(\omega)$, $W(\omega)$, and $T(\omega)$, which are matrices in $\Re^{n_2 \times n_2}$, $\Re^{m_2 \times n_2}$, and $\Re^{m_2 \times n_1}$ respectively. The vectors $c(\omega)$, $h(\omega)$, and $y^{l,u}(\omega)$ are random vectors in $\Re^{n_2}$, $\Re^{m_2}$, and $\Re^{n_2}$ respectively. We assume that the number of possible realizations of ω is finite. Under this assumption, $\xi(\omega)$ is taken to mean that for different ω's the data of the problem change. The dependence of y on uncertainty is depicted as $y(\omega) \in \Re^{n_2}$. The vector $y(\omega)$ is still the decision variable but this notation is used to stress the point that for different realizations of ω we must have a different y. In the objective function (3), the quadratic term represents the risk of the decision measured by variance, while the linear term represents the expected outcome. The scalar $\alpha \in [0, 1]$ is used in (3) to describe the trade-off between risk expectation.

Deriving the multi-stage problem from the two-stage formulation is just a matter of applying the ideas described above recursively to attain the required number of stages. For the multistage problem with T_s periods, the first-stage decision remains the same but for $t = 2 \ldots T_s$ we have

$$\mathcal{Q}_t(x_{t-1}) = E_{\xi_t}\Big[Q_t\Big(x_{t-1}, \xi_t(\omega)\Big)\Big], \tag{6}$$

where

$$\begin{aligned} Q_t(x_{t-1}, \xi_t(\omega)) = \min_y \alpha \frac{1}{2} y_t'(\omega) H_t(\omega) y_t(\omega) \\ - (1-\alpha) c_t' y_t(\omega) + \mathcal{Q}_{t+1}(y_t(\omega)) \\ s.t \; W_t(\omega) y_t(\omega) = h_t(\omega) - T_{t-1}(\omega) x_{t-1} \\ y_t(\omega)^l \le y_t(\omega) \le y_t(\omega)^u . \end{aligned} \tag{7}$$

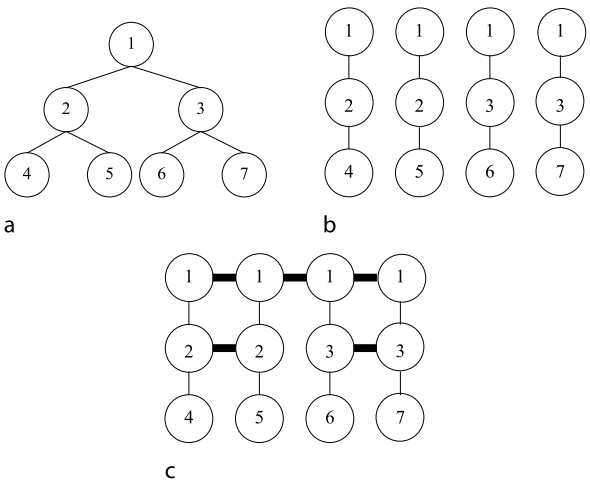
For the last time period $t = T_s$, the recourse function $\mathcal{Q}_{T_s+1}$ is zero.

Our principal concern involves decomposition algorithms for (7). For more insight into the properties of stochastic quadratic problems the reader is referred to [14], and [15]. Before we delve into decomposition algorithms, we introduce some terminology that will be used in the next section.

The dynamic programming model (7) is usually referred to as *non-anticipative*. This property means that decisions are based on the past and not the future. There are two ways this concept can be represented, namely compact and split-view formulations [20].

The compact variable formulation can be mapped directly onto a tree structure known as the *scenario tree*; see Fig. 1a. The root of the tree represents the state of the world that is deterministic. As we move down the scenario tree, different events represent different realizations of ω, each level of the tree represents a different time period, and the path from the root to a leaf node is known as a *scenario*. We use $v = (t, k)$ to denote the kth node in period t, $a(v)$ the ancestor node, and $d(v)$ the descendant nodes. Benders decomposition, to be introduced in the next section, assumes such a structure and the result is a decomposition of the large scale problem into several subproblems, each representing a node in the tree.

In a split-variable formulation for each scenario, from the set of possible scenarios, new decision variables are introduced so that the large-scale problem is decomposed into n subproblems, where n is the number of scenarios. Conceptually, using this approach, the non-anticipative constraints are completely relaxed; see



Decomposition Algorithms for the Solution of Multistage Mean-Variance Optimization Problems, Figure 1
Different views on non-anticipativity

Fig. 1b. To enforce these constraints, new constraints are introduced that "rebuild" the links between subproblems, usually through some penalty function (see Fig. 1c).

Methods

The importance of decomposition algorithms in SP was recognized early on, as results in the theory of stochastic programs are closely linked with their solution algorithms. The two algorithms described in this section represent two very promising approaches in decomposition of SPs.

Nested Benders Decomposition (NBD)

Benders decomposition was first proposed in [2], and it has been applied to SP by [26]; it is usually referred to as the *L-shaped method* due to the structure of the constraint matrix. The extension to the non-linear convex case has been done in [12], and the extension to the general convex SP appears in [8]. The algorithm has also been widely studied for multistage problems in a parallel environment[6]. More recent studies appear in [18]. In [15] the quadratic case is also studied.

It can easily be seen that (2a) is equivalent to:

$$\begin{aligned} \min_{x,\theta} \quad & \frac{1}{2}x'Hx + c'x + e'\theta \\ s.t \quad & Ax = b \\ & \theta \le p_\omega Q(x,\xi(\omega)) \\ & x^l \le x \le x^u \end{aligned} \tag{8}$$

where e is a vector of ones. The dimension of the latter vector is equal to the number of nodes in the next period. The expression $p_\omega Q(x,\xi(\omega))$ represents the value of the next stage decision if event ω occurs (with probability p_ω). The dimensions of the rest of the data are the same as in (2a). Even though it is possible to aggregate the θ vector to a single variable, computational studies [5,6] have shown that the reduction of variables did not enhance performance, possibly due to loss of information.

To represent the recourse function in (8), we construct an approximation using outer linearizations. This is achieved by computing cuts (cutting planes). There are two types of cuts: optimality and feasibility. Instead of solving the large-scale problem (8) we solve the relaxed version

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x'Hx + c'x + e'\theta \\ s.t \quad & Ax = b \\ & Dx \ge d \qquad (9a) \\ & \theta \ge Gx + g \qquad (9b) \\ & x^l \le x \le x^u \end{aligned}$$

where (9a) and (9b) represent feasibility and optimality cuts, respectively. The aim of these constraints is to approximate the feasible region of (8). Feasibility cuts are constructed as follows: Assuming that $t = T$ and for a fixed $\hat{\omega}$ the νth problem in (7) takes the following form:

$$\begin{aligned} Q(x) = \min_{y} \ & \frac{\alpha}{2}y'Hy - (1-\alpha)c'y \\ & s.t \ Wy = h - Tx_{a(\nu)} \\ & y^l \le y \le y^u \end{aligned} \tag{10}$$

Assume that this problem is infeasible due to the vector $x_{a(\nu)}$ generated in a subproblem of a previous stage. Consider the following problem:

$$P(y, x_{a(\nu)}) = \min_{y} e'y^+ + e'y^-$$

$$s.t \ Wy + y^+ - y^- = h - Tx_{a(\nu)} \tag{11}$$

$$y^l \le y \le y^u \tag{12}$$

$$y^{+,-} \ge 0$$

Then since the original problem was infeasible due to $x_{a(\nu)}$ we must have that $P(\cdot, x_{a(\nu)}) > 0$. Let λ be the Lagrange multiplier of the constraint in (11), then by duality we must also have that $\lambda'(h - Tx_{a(\nu)}) \le 0$. Set $D = \lambda'T$ and $d = \lambda'h$ to obtain (9a), a supporting hyperplane to $Q(x)$. To apply this result when $t \ne T$ just note that the same procedure is recursively applied by taking under consideration the additional constraints from cuts of other subproblems.

For optimality cuts, one proceeds as follows: again we start with the problem in (10) and let x_k be the solution vector of a subproblem in the previous stage. By the gradient inequality we must have that (ω is dropped since it is clear from context):

$$Q(x) \ge Q(x_k) + \nabla Q(x_k)(x - x_k)$$

$$Q(x_k) = \frac{\alpha}{2}y'Hy - (1-\alpha)c'y + \lambda'(Wy - h + Tx_k)$$

$$\nabla Q(x_k) = \lambda'T$$

Thus

$$Q(x) \geq \lambda' Tx + \frac{a}{2} y' Hy - (1-\alpha)c'y + \lambda'(Wy - h)$$

Set

$$\begin{aligned} \theta &= Q(x) \\ G &= \lambda' T \\ g &= \frac{\alpha}{2} y' Hy - (1-\alpha)c'y + \lambda(Wy - h) \end{aligned}$$

to obtain (9b). Since we require a lower support for the expected value, we then multiply G and g by the probability of ω taking the particular realization of $\hat{\omega}$ for two-stage problems and the conditional probability for multistage problems. The application of optimality cuts when $t \neq T$ is again developed recursively just by taking into account the additional variables and constraints.

The algorithm proceeds by solving the relaxed problem (7) to obtain a solution vector, known as the *proposal vector*. The latter is then used to solve the subproblems in (10). If a subproblem is feasible then an optimality cut is appended to the constraint set of the ancestor problem (also called the *master problem*). Otherwise, only a feasibility cut is appended.

In the linear case, there are some well known drawbacks to the algorithmic framework developed above [5,21]. We expect issues similar to the following to manifest themselves in the quadratic case:

- The algorithm tends to be inefficient in early iterations due to the poor description of the original objective function provided by the cuts. Moreover, if a good warm-start is used, the algorithm may deviate significantly from this point, so any efficiency achieved by a good starting point is lost.
- The number of cuts for master problems may increase substantially, adding considerable computational burden to their solution.

For these reasons a regularized version of the algorithm was proposed in [21]; see also [22,23] for the multistage version. Ruszczynski's results, as well as a study performed in [28], indicate that the regularized version outperforms the original algorithm.

The basic idea is to add a quadratic term $\rho \parallel x - \hat{x} \parallel_2^2$ in the objective function, where $\hat{x}$ is chosen as the "best" current point, in a way to be made precise, and ρ is a penalty parameter. For a high value of ρ the algorithm is penalized from deviating from the current point. In [21], the convergence of the algorithm for $\rho = \frac{1}{2}$ was established for the convex case. The regularizing term stabilizes the behavior of the algorithm between iterations, enables valid deletion schemes of the cuts, and avoids degenerate iterations that would otherwise be possible.

The original problem is now decomposed into three types of subproblems. The first type is for the root node. The following problem is solved at each iteration:

$$\begin{aligned} \min_{x,\theta} \quad & \frac{1}{2}\alpha x' Hx - (1-\alpha)c'x + e'\theta + \frac{\rho}{2} \parallel x - \hat{x} \parallel_2^2 \\ s.t \quad & Ax = b \\ & Gx \geq \theta + g \\ & Dx \geq d \\ & x^l \leq x \leq x^u . \end{aligned}$$

The second type is for non-terminal nodes. The following subproblem needs to be considered:

$$\begin{aligned} \min_{y_v,\theta_v} \quad & \frac{a}{2} y_v' H_v y_v - (1-\alpha)c_v' y_v \\ & + e'\theta_v + \frac{\rho_v}{2} \parallel y_v - \hat{y}_v \parallel_2^2 \\ s.t \quad & W y_v = h_v - T_{a(v)} x_{a(v)} \\ & G_v y_v \geq \theta_v + g_v \\ & D_v y_v \geq d_v \\ & y_v^l \leq y_v \leq y_v^u . \end{aligned} \tag{13}$$

The third type is for terminal nodes. This type of subproblem is identical to (13) without, of course, the cuts in the constraint set and the regularizing term in the objective function.

The way cuts are recursively defined and the way subproblems are nested in each other has led this to be referred to as *nested Benders decomposition* (NBD). The algorithm can now be stated as follows:

Step 1: Set the iteration counter $i = 0$ and $t = k = 0$, and let $\hat{x}$ be a feasible point.

Step 2: Construct and solve $v(t,k)$ to find the solution vector x_v^i.

Step 2.1: If the problem is infeasible and $t = 0$ then STOP: the problem is infeasible.

Step 2.2: If the problem is infeasible and $t > 0$, generate an optimality cut (9a) and append it to the constraint set of $a(v)$.

Step 2.3: If the problem was optimal and $t > 0$, generate an optimality cut (9b) and append it to the constraint set of $a(v)$.

Step 3: Compute

$$\hat{F}(x_v^i) = \frac{1}{2}x'Hx + c'x + \sum_{j=d(v)} \theta_j$$

$$F(\hat{x}_v^i) = \frac{1}{2}x'Hx + c'x + \sum_{j=d(v)} Q_j(x)\,.$$

If $\hat{F} = F$ and $t = k = 0$ then STOP: $\hat{x}$ is optimal; Else go to step 4

Step 4: Update the regularizing term:

4.1 If a subproblem returned a feasibility cut then $\hat{x}_v^{i+1} = \hat{x}_v^i$.

4.2 If $F(x_v^i) > F(\hat{x}_v^i)$ or $F(x_v^i) > \tau F(\hat{x}_v^i) + (1-\tau)\hat{F}$, then set $\hat{x}_v^{i+1} = \hat{x}_v^i$, and increase ρ.

4.3 If $F(x_v^i) < F(\hat{x}_v^i)$ or $F(x_v^i) < \tau F(\hat{x}_v^i) + (1-\tau)\hat{F}$, then set $\hat{x}_v^{i+1} = x_v^i$ and decrease ρ.

4.4 If $F(x_v^i) = \hat{F}$, then set $\hat{x}_v^{i+1} = x_v^i$, and decrease ρ.

Step 5: Set $i = i + 1$, find the next subproblem to solve (see below), and go to step 2.

Augmented Lagrangian Decomposition (ALD)

An alternative algorithm to Benders decomposition described in the previous section is based on the augmented lagrangian and the method of multipliers [3]. The fundamental difference between NBD and ALD is the way the two algorithms attack non-anticipativity constraints. NBD handles these constraints by having a master problem generating proposals to the subproblems further down the event tree; proposal vectors are affected by "future" nodes by feasibility and optimality cuts. In ALD a different approach is taken: non-anticipativity constraints are relaxed by expressing the large-scale problem in terms of smaller subproblems that are discouraged from violating the original constraints. The algorithm we use was developed in [4], so here we only sketch the main idea. ALD was developed and applied to the stochastic quadratic programming setting in [25] with encouraging results. Similar algorithms to ALD have been developed for linear stochastic programs [16,17].

The expectation in (6) for a given time period can also be written as

$$\begin{aligned} &\min_y \sum_{i=1}^m p_i\Big(\frac{\alpha}{2}y_i'H_iy_i - (1-\alpha)c_i'y_i\Big) \\ &s.t\, W_jy_i = h_j - T_jx_{a(i)} \qquad j = 1\ldots r \\ &y_i^l \le y_i \le y_i^u\,. \end{aligned} \tag{14}$$

The problem in (14) is to be interpreted as follows: at the current time period there are m scenarios, each having different realizations for H, W, c, etc. There are r linking constraints (14) that are linked by the vector $x_{a(i)}$. In [4] the problem is decomposed by introducing a new variable z as follows

$$\begin{aligned} &\min_{y,z} \sum_{i=1}^m p_i\Big(\frac{\alpha}{2}y_i'H_iy_i - (1-\alpha)c_i'y_i\Big) \\ &s.t\, W_{ji}y_i = z_{ij} \qquad j = 1\ldots r; i \in I(j) \\ &z_{ij} = h_j - T_jx_{a(i)} \qquad j = 1\ldots r; i \in I(j) \\ &y_i^l \le y_i \le y_i^u\,, \end{aligned} \tag{15}$$

where $I(j)$ contains the indices of the subproblems that the jth constraint "crosses", i. e., $I(j) = \{i | w_{ji} \neq 0\}$. "Crosses" means that a constraint contains data from more than one subproblem. It is obvious that (14) and (15) are exactly the same problem, but the structure of (15) facilitates a decomposition algorithm via the relaxation of the constraints of (15). In [4] the method of multipliers is used for the general problem $\min\{f(x)|Ax = b\}$. Let $L_c(x,\lambda)$ denote the associated augmented lagrangian defined by $L_c(x,\lambda) = f(x) + \lambda'(Ax-b) + \frac{c}{2} \parallel Ax - b \parallel_2^2$, where λ is the vector of multipliers. The general algorithmic framework of the method of multipliers can be described as follows:

Step 1: Initialization: Set the iteration counter $k = 0$, and set $c(0) > 0$. Set $x(0)$, and $\lambda(0)$ as the starting point for the decision variables, and lagrange multipliers, respectively.

Step 2: Compute the next point $x(k+1) = \arg\min L_c(x, \lambda(k))$.

Step3 Update the Lagrange multiplier vector $\lambda(k+1) = \lambda(k) + c(k)\big(Ax(k+1) - b\big)$.

Step 4: Update the penalty parameter $c(k)$, and set $k = k + 1$. If some convergence criterion is not satisfied go to step 2.

Applying this general algorithmic framework to (15), the problem is decomposed into m subproblems and the non-anticipativity constraints are enforced through the penalty term in the augmented lagrangian.

The computation for the solution of (15) involves keeping z fixed in order to compute the next incumbent for y, and then keeping y fixed in order to compute the next incumbent for z. Thus, at the kth iteration the following subproblems are solved:

$$\begin{aligned} y_i(k+1) = \arg\min_{\xi} \Bigg\{ & p_i \left(\frac{\alpha}{2} \xi_i' H_i \xi_i - (1-\alpha) c_i' \xi_i \right) \\ & + \sum_{\{j \mid i \in I(j)\}} \left(\lambda_{ji}'(k) W_{ji} \xi_i \right. \\ & \left. + \frac{c(k)}{2} (W_{ji}\xi_i - z_{ji})^2 \right) \Bigg\} \\ & \forall i = 1, \ldots, n \end{aligned}$$

$$\begin{aligned} z_{ji}(k+1) = \arg\min_{\zeta_{ji}} \Bigg\{ & - \sum_{i \in I(j)} \lambda_{ji}'(k) \zeta_{ji} + \frac{c(k)}{2} \\ & \cdot \sum_{i \in I(j)} (W_{ji}\xi_i - \zeta_{ji})^2 \Bigg\} \quad \forall j = 1, \ldots, r \\ & \text{s.t } \zeta_{ji} = h_j - T_j x_{a(i)} \quad i \in I(j) \end{aligned}$$

followed by an update of the lagrange-multiplier vector $\lambda_{ji}(k+1) = \lambda_{ji}(k) + c(k)(W_{ji} y_i(k+1) - z_{ji}(k+1))$. From a computational point of view the above iterative framework is inefficient because of the alternate minimizations required, making this algorithm unsuitable for a parallel environment. In our implementation we used the more efficient iteration proposed in [4]:

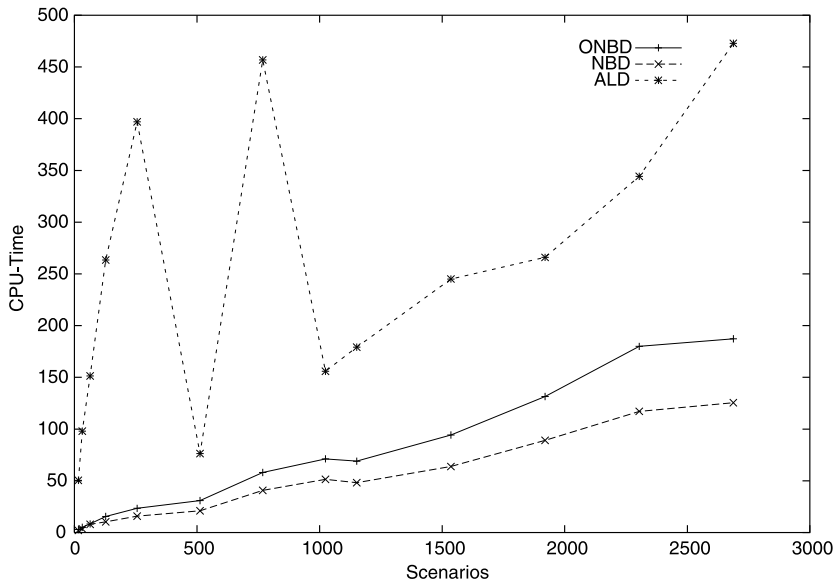
$$\begin{aligned} y_i(k+1) = \arg\min_{\xi} \Bigg\{ & p_i \left(\frac{\alpha}{2} \xi_i' H_i \xi_i - (1-\alpha) c_i' \xi_i \right) \\ & + \sum_{\{j \mid i \in I(j)\}} \left(\lambda_j'(k) W_{ji} \xi_i \right. \\ & \left. + \frac{c(k)}{2} \left(W_{ji}(\xi_i - y_i(k)) + w_j \right)^2 \right) \Bigg\} \end{aligned} \tag{16}$$

where $w_j = \frac{1}{m_j}(W_j y_i - h_j + T_j x_{a(i)})$, $\lambda_j(k+1) = \lambda_j(k) + \frac{c(k)}{m_j}(W_j y_i - h_j + T_j x_{a(i)})$, and m_j denotes the cardinality of $I(j)$. The derivation of this iteration is discussed in Bertsekas and Tsitsiklis ([4], p. 249). The expression in (16) forms the main iteration of the ALD algorithm. In order to have a complete description of the algorithm we need to specify how one can perform the updates of the penalty parameter $c(k)$ and how we tested for convergence.

The obvious convergence criteria for ALD are a test for feasibility and small changes in the objective function. However, it is possible, due to a poor selection of updates for $c(k)$, to reach a suboptimal solution. For this reason, it is vital to check the KKT conditions of the problem in addition to any other stopping criteria. If the KKT conditions are not satisfied while the change in the objective function is small (10^{-6} in our implementation), the update strategy for the penalty parameter appears to have been inappropriate. We performed various experiments with different update strategies for this penalty parameter and found that the strategy that works best on most problems is to start with a small value (0.001) and increase it at every iteration by another small factor (1.05); being more aggressive with the update of this parameter caused the algorithm to terminate prematurely. Note that an arbitrary starting point can be used to start the algorithm. If a feasible solution or the solution from a previous run is available it may be beneficial to start with a higher penalty term.

Numerical Experiments

The two algorithms were implemented and tested on a multistage financial planning problem. The detailed results can be found in [19]. Figure 2 summarizes the numerical performance (in terms of CPU time) as the number of scenario increases. ALD, and NBD stand for Augmented Lagrangian Decomposition, and Nested Benders Decomposition respectively. ONBD refers to Ordinary Nested Benders Decomposition, i. e. NBD without the regularizing term. From Fig. 2 it is clear that the regularized version of Benders decomposition is the most efficient of the algorithms we considered in this article. This result is in line with similar studies performed in the linear setting. One possible explanation is that the NBD algorithm takes advantage of the constraint structure of multistage stochastic programming problems more effectively. Note that the ALD algorithm can be applied to separable convex problems with more general constraint structure while NBD will need to be modified in order to be applicable to other types of separable problems. SP problems are one of the



Decomposition Algorithms for the Solution of Multistage Mean-Variance Optimization Problems, Figure 2
Solution times vs. number of scenarios

most frequently occurring class of large scale problems, so it is important to know whether cutting plane type algorithms or Lagrangian based algorithms take advantage of this structure more effectively. Based on the results of our experiments it seems that the NBD algorithm appears to be substantially better. Furthermore, we found that the penalty parameter often caused notable changes to the convergence times of both NBD and ALD. Finding an update scheme that works for all problems is a difficult task. In ALD the penalty parameter has two goals, one is forcing feasibility and the other of keeping iterations close to each other, thus a 'suboptimal' penalty update scheme may be more damaging than in NBD, this may give some insight to the difference in performance of the two algorithms. More detailed numerical experiments can be found in [19].

References

1. Beale EML (1955) On minimizing a convex function subject to linear inequalities. J R Stat Soc 17:173–184
2. Benders JF (1962) Partitioning procedures for solving mixed-variables problems. Numerische Mathematik 4: 238–252
3. Bertsekas DP (1999) Nonlinear Programming, 2nd edn. Athena Scientific, Belmont
4. Bertsekas DP, Tsitsiklis JN (1989) Parallel and Distributed Computation. Prentice-Hall, Englewood Cliffs
5. Birge JR (1997) Stochastic programming computation and applications. INFORMS J Comput 9:111–133
6. Birge JR, Donohue CJ, Holmes DF, Svintsitski OG (1996) A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. Math Program 75(2):327–352
7. Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer, New York
8. Birge JR, Rosa CH (1996) Parallel decomposition of large-scale stochastic nonlinear programs. Ann Oper Res 64: 39–65
9. Blomvall J, Lindberg P (2002) A Riccati-based primal interior point solver for multistage stochastic programming. Eur J Oper Res 143(2):452–461
10. Dantzig GB (1955) Linear programming under uncertainty. Manag Sci 1:197–206
11. Frauendorfer K (1996) Barycentric scenario trees in convex multistage stochastic programming. Math Program 75(2B):277–293
12. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Theory Appl 10(4):237–260
13. Kall P, Wallace SW (1994) Stochastic Programming. Wiley, Chichester
14. Lau K, Womersley RS (2001) Multistage quadratic stochastic programming. J Comput Appl Math 129(1-2):105–138
15. Louveaux F (1978) Piecewise convex programs. Math Program 15:53–62
16. Mulvey JM, Ruszczynski A (1992) A diagonal quadratic approximation method for linear multistage stochastic programming problems. In: System modeling and optimiza-

tion, Lecture Notes in Control and Inform Sci, vol 180. Springer, Berlin, pp 588–597
17. Mulvey JM, Ruszczynski A (1995) A new scenario decomposition method for large-scale stochastic optimization. Oper Res 43(3):477–490
18. Nielsen SS, Zenios SA (1997) Scalable parallel Benders decomposition for stochastic linear programming. Parallel Comput 23(8):1069–1088
19. Parpas P, Rustem B (2005) Computational assessment of nested benders and augmented lagrangian decomposition for mean-variance multistage stochastic problems. INFORMS J Comput 19(2):239–247
20. Rockafellar T, Wets R (1991) Scenarios and policy aggregation in optimization under uncertainty. Math Oper Res 16(1):119–147
21. Ruszczynski A (1986) A regularized decomposition method for minimizing a sum of polyhedral functions. Math Program 35:309–333
22. Ruszczynski A (1993) Parallel decomposition of multistage stochastic programming problems. Math Program 58(2A):201–228
23. Ruszczynski A (1995) On the regularized decomposition method for stochastic programming problems. In: Stochastic programming, Lecture Notes in Econom and Math Systems, vol 423. Springer, Berlin, pp 93–108
24. Salinger DH, Rockafellar T (2003) Dynamic splitting: an algorithm for deterministic and stochastic multiperiod optimization. Stochastic Programming E-Print Series (SPEPS). http://www.speps.info/
25. Settergren R (2001) Decomposition of financial engineering problems. Technical report. Department of Computing, Imperial College, London
26. Van Slyke R, Wets RJ-B (1969) L-shaped linear programs with applications to control and stochastic programming. SIAM J Appl Math 17:638–663
27. Steinbach MC (1998) Recursive direct algorithms for multistage stochastic programs in financial engineering. In: Kall P, Luthi HJ (eds) Papers of the International Conference on Operations Research. Springer, New York, pp 241–250
28. Vladimirou H (1998) Computational assessment of distributed decomposition methods for stochastic linear programs. Eur J Oper Res 108:653–670

Decomposition in Global Optimization

HOANG TUY
VAST, Institute of Mathematics,
Hanoi, Vietnam

MSC2000: 90C26, 90C25

Article Outline

BB Procedure for Decomposition
Convergence Achieved with Lagrangian Bounds
Partly Convex Optimization Problems
Partly Linear Optimization
Extensions
Partly Monotonic Optimization
Monotonic/Convex Optimization
References

In many nonconvex optimization problems the set of variables is partitioned into two groups such that the problem becomes much easier to solve when the variables in one group are held temporarily fixed. To exploit this structure, a method which has proved to be efficient is to *decompose* the problem into a sequence of easier subproblems involving only variables of the other group. The basic tool for this decomposition is the branch and bound (BB) concept.

BB Procedure for Decomposition

Consider the nonconvex global optimization problem

$$\min\{F(x,y)\colon G(x,y)\preceq_K 0,\ x\in X,\ y\in Y\}\ , \quad \text{(P)}$$

where X is a compact convex subset of $\mathbb{R}^n$, Y is a closed convex subset of $\mathbb{R}^p$, $F\colon X\times Y\to\mathbb{R}$, $G\colon X\times Y\to\mathbb{R}^m$, K is a closed convex cone in $\mathbb{R}^m$ and $\preceq_K$ is the partial ordering in $\mathbb{R}^m$ induced by the cone K, i. e., such that $y\preceq_K y'\Leftrightarrow y'-y\in K$.

Problems of this form abound in applications such as pooling and blending in oil refining, optimal design of water distribution, structural design, signal processing, robust stability analysis and design of chips.

Suppose that by fixing $x\in X$ problem (P) becomes an easier problem in $y\in Y$. Then, to take advantage of this property on can solve (P) by a BB algorithm with branching performed in the x-space.

Specifically, at iteration k of of the BB procedure a collection S_k of partition sets in the x-space is considered, where for each partition set $M\in S_k$ a number (lower bound) $\beta(M)\in\mathbb{R}\cup\{+\infty\}$ has been computed such that

$$\beta(M)\le\inf\{F(x,y)\colon G(x,y)\preceq_K 0, x\in M\cap X, y\in Y\}. \quad (1)$$

A partition set $M_k \in \operatorname{argmin}\{\beta(M)\colon M \in S_k\}$ is then further subdivided according to an exhaustive subdivision rule (e. g., the standard bisection rule, if rectangular subdivision is used), while the best feasible solution available $(\bar{x}^k, \bar{y}^k)$ is recorded. By removing every M such that $\beta(M) \geq F(\bar{x}^k, \bar{y}^k)$ the new collection S_{k+1} is formed. If $S_{k+1} = \emptyset$, the procedure terminates, concluding that the problem is infeasible if no feasible solution is available, or else that the current best feasible solution is actually an optimal one. Otherwise, the next iteration is started.

A key operation in this procedure is bounding: $M \mapsto \beta(M)$. It is assumed that this operation satisfies the following natural conditions:

$$\begin{aligned} &\text{(a)} \quad M' \subset M \Rightarrow \beta(M') \geq \beta(M)\,; \\ &\text{(b)} \quad \beta(M) < +\infty \Rightarrow M \cap X \neq \emptyset\,. \end{aligned} \tag{2}$$

When the BB procedure is infinite it generates a filter (an infinite nested sequence of partition sets) M_{k_ν}, $\nu = 1, 2, \ldots$, such that

$$\begin{aligned} &\beta(M_{k_\nu}) \leq \min(\mathrm{P}) \quad \forall \nu, \\ &M_{k_\nu} \cap X \neq \emptyset \quad \forall \nu, \\ &\bigcap_{\nu=1}^{+\infty} M_{k_\nu} = \{x^*\}\,. \end{aligned} \tag{3}$$

The algorithm is said to be *convergent* if $x^* \in X$ and

$$\min(\mathrm{P}) = \min\{F(x^*, y)\colon G(x^*, y) \preceq_K 0,\ y \in Y\}, \tag{4}$$

so any optimal solution y^* of this problem yields an optimal solution (x^*, y^*) of (P).

The basic issue of this decomposition scheme is under which conditions the BB procedure described above is guaranteed to converge in sense (4).

First observe that, since $M_{k_{\nu+1}} \subset M_{k_\nu}$ and hence, $\beta(M_{k_{\nu+1}}) \geq \beta(M_{k_\nu})$, we have from (3)

$$\beta(M_{k_\nu}) \nearrow \beta^* \leq \min(\mathrm{P})\,. \tag{5}$$

Theorem 1 *If $\beta(M_k) = +\infty$ for some k then (P) is infeasible and the algorithm terminates. If $\beta(M_k) < +\infty\ \forall k$, then there is an infinite subsequence M_{k_ν}, $\nu = 1, 2, \ldots$, satisfying (3) and such that $x^* \in X$. If in addition*

$$\lim_{\nu \to +\infty} \beta(M_{k_\nu}) = \min\{F(x^*, y)\colon G(x^*, y) \preceq_K 0, y \in Y\}, \tag{6}$$

then the BB decomposition algorithm is convergent.

Condition (6) simply says that the lower bound must be *eventually exact* as $k \to +\infty$. Also note that for ensuring that $x^* \in X$ the condition $x \in M \cap X$ in (1) is essential and cannot be omitted.

Convergence Achieved with Lagrangian Bounds

In many important cases *Lagrangian bounds* can be used throughout the decomposition algorithm, so that for every partition set M:

$$\beta(M) = \sup_{\lambda \in K^*} \inf\{F(x, y) + \langle \lambda, G(x, y)\rangle\colon x \in M \cap X,\ y \in Y\}\,, \tag{7}$$

where $K^* = \{\lambda \in \mathbb{R}^m\colon \langle \lambda, u\rangle \geq 0\ \forall u \in K\}$ is the dual cone of K.

For every $t \geq 0$ define

$$v(t) = \sup_{\lambda \in K^*} \inf_{\substack{y \in Y \\ \|x - x^*\| \leq t, x \in X}} \{F(x, y) + \langle \lambda, G(x, y)\rangle\}, \tag{8}$$

where, as throughout in what follows, x^* denotes the limit point of an exhaustive filter of partition sets generated by the BB algorithm, i. e., an infinite nested sequence $\{M_{k_\nu}\}$ such that $\cap_{\nu=1}^{+\infty} M_{k_\nu} = \{x^*\}$.

Theorem 2 *Assume Lagrangian bounds are used throughout the BB decomposition algorithm, and:*
(A1) $v(t) \to v(0)$ *as* $t \searrow 0$.
(A2) $\sup_{\lambda \in K^*} \inf_{y \in Y}\{F(x^*, y) + \langle \lambda, G(x^*, y)\rangle\} = \min_{y \in Y} \sup_{\lambda \in K^*}\{F(x^*, y) + \langle \lambda, G(x^*, y)\rangle\}$.
Then the BB decomposition algorithm is convergent.

Condition A1 expresses the continuity of $v(t)$ at $t = 0$. Condition A2 requires that the duality gap be zero for the subproblem $\min_{y \in Y}\{F(x^*, y)\colon G(x^*, y) \preceq_K 0\}$.

Theorem 3 *Assume that $F(x, y)$, $G_i(x, y)$, $i = 1, \ldots, m$, are lower semi-continuous and:*
(i) There exists a compact set $Y^0 \subset Y$ such that

$$\begin{aligned} &(\forall x \in X)(\forall \lambda \in K^*) \\ &Y^0 \cap \operatorname{argmin}_{y \in Y}\{F(x, y) + \langle \lambda, G(x, y)\rangle\} \neq \emptyset; \end{aligned} \tag{9}$$

(ii) $\sup_{\lambda\in K^*}\inf_{y\in Y}\{F(x^*,y) + \langle\lambda, G(x^*,y)\rangle\} = \min_{y\in Y}\sup_{\lambda\in K^*}\{F(x^*,y)+\langle\lambda, G(x^*,y)\rangle\}$.

Then the BB decomposition algorithm using Lagrangian bounds is convergent. Furthermore, the function $x\mapsto\sigma(x) := \min\{F(x,y)\colon G(x,y)\preceq_K 0,\ y\in Y\}$ *is lower semicontinuous at* x^* *and satisfies*

$$\lim_{\substack{x\in X\\ x\to x^*}}\ \sigma(x) = \min(\mathrm{P})\,. \tag{10}$$

Remark 1 Condition (i) cannot be replaced by the following weaker one

(*) *There exists a compact set* $Y^0\subset Y$ *such that for each* $\lambda\in\mathbb{R}^m_+$ *and for each* $x\in X$ *either the set of optimal solutions of the problem*

$$\min\{F(x,y)+\sum_{i=1}^m\lambda_i G_i(x,y)\colon y\in Y\} \tag{11}$$

is empty or it has a nonempty intersection with Y^0.

Partly Convex Optimization Problems

An important class of problems (P) is constituted by *partly convex problems*, i. e., problems (P) with the following assumption:

(PCA) *For every fixed* $x\in X$ *the function* $y\mapsto F(x,y)$ *is convex, while the mapping* $y\mapsto G(x,y)$ *is K-convex.* The latter means that $G(x,\alpha y^1+(1-\alpha(y^2\preceq_K \alpha G(x,y^1)+(1-\alpha)G(x,y^2)$ whenever $y^1,y^2\in\mathbb{R}^p, 0\le\alpha\le 1$.

Owing to (PCA), for every $\lambda\in K^*$ and fixed $x\in X$ the function $\langle\lambda, G(x,y)\rangle$ is convex and the problem $\min\{F(x,y)+\langle\lambda, G(x,y)\rangle : y\in Y\}$ is a convex optimization problem. Specific decomposition methods for this class of problems were developed earlier in [3,4], and more recently in [1]. Within the present framework, the convergence conditions can be specialized as follows.

A function $f\colon Y\to\mathbb{R}$ is said to be *coercive* on Y if $\lim_{y\in Y,\ y\to+\infty} f(y) = +\infty$. Clearly this is equivalent to saying that for any $\eta\in\mathbb{R}$ the set $\{y\in Y\colon f(y)\le\eta\}$ is bounded.

Theorem 4 *Assume (PCA) with* $F(x,y), G_i(x,y),\ i=1,\ldots,m$, *is lower semi-continuous on* $X\times Y$ *and continuous in x for fixed* $y\in Y$. *Assume further that:*

(S) For some $\lambda^*\in K^*$ *the function* $y\mapsto F(x^*,y)+\langle\lambda^*, G(x^*,y)\rangle$ *is coercive on Y.*

Then the BB decomposition algorithm using Lagrangian bounds is convergent and the function $\sigma(x) := \min\{F(x,y)\colon G(x,y)\preceq_K 0,\ y\in Y\}$ *is lower semicontinuous at* x^* *and satisfies*

$$\lim_{\substack{x\in X\\ x\to x^*}}\ \sigma(x) = \min(\mathrm{P})\,.$$

Remark 2 Condition A2, sometimes referred to as *dual properness* at x^*, means that the subproblem $\min\{F(x^*,y)\colon G(x^*,y)\preceq_K 0,\ y\in Y\}$ has zero duality gap. When Y is bounded, condition (S) obviously holds, so by Theorem 4, both conditions A1 and A2 follow from (PCA) and the lower semicontinuity of $F(x,y), G_i(x,y), i=1,\ldots,m$. On the other hand, when Y is unbounded, dual properness (i. e., condition A2), even coupled with continuity of the functions involved, is not sufficient to guarantee condition A1. These results suggest that several methods developed in the literature for problems of the form (P) should be revised for validity.

Partly Linear Optimization

A subclass of the class of partly convex optimization problems is formed by partly linear optimization problems which have the general formulation

$$\min\{\langle c(x),y\rangle+\langle c^0,x\rangle : A(x)y+B(x)\le b, r\le x\le s,\ y\ge 0\}\,, \tag{GPL}$$

where $x\in\mathbb{R}^n, y\in\mathbb{R}^p, c:\mathbb{R}^n\to\mathbb{R}^p,\ c^0\in\mathbb{R}^n,\ A := \mathbb{R}^n\to\mathbb{R}^{m\times p},\ B\in\mathbb{R}^{m\times n},\ b\in\mathbb{R}^m,\ r,s\in\mathbb{R}^n_+$.

A special case of interest is the "pooling and blending problem" from the petrochemical industry which can be stated as

$$\min\{c^T y : A(x)y\le b,\ y\ge 0,\ x\in X\}\,,$$

where X is a box in $\mathbb{R}^n$ and $A(x)$ is an $m\times p$ matrix whose elements $a_{ij}(x)$ are continuous functions of x. Condition (S) in Theorem 4 now reads

$$\text{For some } \lambda^*\in\mathbb{R}^m_+ \quad \text{we have } c^T y+\langle\lambda^*, A(x^*)y-b\rangle\to+\infty \quad \text{as } y\to+\infty\,,$$

which clearly holds if and only if $\langle A(x^*),\lambda^*\rangle + c > 0$. For example this condition is fulfilled by the partly linear problems considered in [1], and also by the bilinear matrix inequalities problem studied in [5].

Extensions

The above decomposition method can be extended to a number of important nonconvex global optimization problems.

Partly Monotonic Optimization

A function $f(x)\colon \mathbb{R}^n \to \mathbb{R}$ is said to be increasing (decreasing, respectively) if $f(x) \le f(x')$ ($f(x) \ge f(x')$, respectively) whenever $x \le x'$ [7].

Theorem 5 *In problem (P) assume that $X = [a,b] \subset \mathbb{R}^m_+$, $> F(x,y), G(x,y)$ are continuous, and*

(PMA) $F(x,y), G_i(x,y), i = 1, \ldots, m,$
are increasing in $x \in [a,b]$ for every fixed $y \in Y$.

Assume further that the set $\{y \in Y\colon G(b,y) \preceq_K 0\}$ is contained in some box Y^0. Then, with lower bounds defined as

$$M = [r,s] \subset [a,b] \mapsto \beta(M) = \min\{F(r,y)\colon G(s,y) \preceq_K 0,\ y \in Y\}, \quad (12)$$

the BB decomposition algorithm is convergent.

If $F(x,y), G_i(x,y), i = 1, \ldots, m,$ are monotonic in $y \in Y^0$ (or more generally, dm functions in $y \in Y^0$ [7]) then the subproblems in (12) are standard monotonic (or dm) optimization problems and can be solved by currently available algorithms [7,10].

Remark 3 Theorem 5 still holds if $F(x,y), G_i(x,y), i = 1 \ldots, m$ are decreasing in $x \in [a,b]$ for fixed $y \in Y$ and we define

$$\beta(M) = \min\{F(s,y)\colon G(r,y) \preceq_K 0\}.$$

Monotonic/Convex Optimization

Theorem 6 *In problem (P) assume $X \subset [a,b]$, $F(x,y), G_i(x,y), i = 1, \ldots, m,$ are continuous in (x,y), increasing in $x \in [a,b]$ for fixed $y \in Y$, and convex (affine, respectively) in y for fixed $x \in [a,b]$. Assume, in addition, that*

(ST) *For some $\lambda^* \in K^*$*
the function $F(a,y) + \langle \lambda^*, G(a,y)\rangle$ is coercive on Y.

Then for every $M = [r,s] \subset [a,b]$, the Lagrangian bound problem

$$\sup_{\lambda \in K^*} \inf_{\substack{x \in M \cap X \\ y \in Y}} F(x,y) + \langle \lambda, G(x,y)\rangle \quad (13)$$

is a convex (linear, respectively) program and the associated BB decomposition algorithm is convergent.

References

1. Ben-Tal A et al (1994) Global minimization by reducing the duality gap. Math Programm 63:193–212
2. Ekeland I, Temam R (1976) Convex analysis and variational problems. North-Holland, Amsterdam, American Elsevier, New York
3. Floudas CA, Visweswaran V (1993) A primal-relaxed dual global optimization approach. J Optim Theory Appl 78: 187–225
4. Geoffrion AM (1972) Generalized Benders Decomposition. J Optim Theory Appl 10:237–260
5. Tuan HD, Apkarian P, Nakashima Y (2000) A new Lagrangian dual global optimization algorithm for solving bilinear matrix inequalities. Int J Robust Nonlin Control 10:561–578
6. Tuy H (1998) Convex Analysis and Global Optimization. Kluwer, Dordrecht
7. Tuy H (2000) Monotonic Optimization: Problems and Solution Methods. SIAM J Optim 11(2):464–494
8. Tuy H (2005) On solving nonconvex global optimization by reducing the duality gap. J Global Optim 32:349–365
9. Tuy H (2007) On a Decomposition Method for Nonconvex Global Optimization. Optim Lett 1:245–258, doi:10.1007/s11590-006-0025-2
10. Tuy H, Minoux M, Hoai Phuong NT (2006) Discrete Monotonic Optimization With Application to a Discrete Location Problem. SIAM J Optim 17:78–97

Decomposition Principle of Linear Programming

JØRGEN TIND
University Copenhagen, Copenhagen, Denmark

MSC2000: 90C06

Article Outline

Keywords
See also
References

Keywords

Optimization; Decomposition

A frequently applied approach in the history of optimization is that of decomposition, by which a large problem is decomposed into smaller problems. The principle of decomposition goes back to the seminal paper by G.B. Dantzig and P. Wolfe [2].

The basic model is a linear programming problem with two sets of constraints to be stated as follows:

$$\begin{cases} \max & cx \\ \text{s.t.} & A_1 x \le b_1 \\ & A_2 x \le b_2 \\ & x \ge 0. \end{cases} \tag{1}$$

where $c \in \mathbf{R}^n$, $A_1 \in \mathbf{R}^{m \times n}$, $b_1 \in \mathbf{R}^m$, $A_2 \in \mathbf{R}^{q \times n}$ and $b_2 \in \mathbf{R}^q$ are given constants and $x \in \mathbf{R}^n$ is a vector of variables.

The fundamental idea is to solve (1) by interaction between two optimization problems, one of which is subject to the first set of constraints and the other subject to the second set of constraints. Denote the second set by

$$X = \{x \ge 0 \colon\ A_2 x \le b_2\}.$$

For simplicity we assume that X is bounded and nonempty. Hence X is a polytope. Let x^i denote an extreme point of X for $i \in P$ where P is the index set of all extreme points. According to the Minkowski representation theorem (see [1]), the polytope X can alternatively be represented as the convex hull of the extreme points, i. e.

$$X = \left\{ x = \sum_{i \in P} \lambda_i x^i \colon \quad \begin{array}{c} \sum_{i \in P} \lambda_i = 1, \\ \lambda_i \ge 0 \text{ for } i \in P \end{array} \right\}.$$

If X is unbounded extreme rays are introduced in the representation of X leading to a straightforward extension of the subsequent considerations.

Hence (1) is equivalent to

$$\begin{cases} \max & \displaystyle\sum_{i \in P} cx^i \lambda_i \\ \text{s.t.} & \displaystyle\sum_{i \in P} A_1 x^i \lambda_i \le b_1 \\ & \displaystyle\sum_{i \in P} \lambda_i = 1 \\ & \lambda_i \ge 0. \end{cases} \tag{2}$$

Problem (2) operates with fewer rows than the original formulation (1). The variable x has been substituted by the variables λ_i. However, since the number of extreme points is usually very large in comparison with the dimension n of the problem, the number of λ-variables may also be very large, and it requires a big effort to enumerate and calculate all extreme points. Fortunately, this is unnecessary. In fact, by the Caratheodory theorem, at most $n + 1$ extreme points need to be considered, see for example [1]. The trouble is to find the correct ones.

Problem (2) is called the *full master problem* since all extreme points are introduced in the formulation. As already indicated we shall consider formulations dealing with only a subset of extreme points. For this purpose let $\overline{P}$ denote a subset of the index set P leading to a tightening of (2), called the *restricted master problem*.

$$\begin{cases} \max & \displaystyle\sum_{i \in \overline{P}} cx^i \lambda_i \\ \text{s.t.} & \displaystyle\sum_{i \in \overline{P}} A_1 x^i \lambda_i \le b_1 \\ & \displaystyle\sum_{i \in \overline{P}} \lambda_i = 1 \\ & \lambda_i \ge 0 \text{ for } i \in \overline{P}. \end{cases} \tag{3}$$

Assume here for simplicity that (3) is feasible. If not, additional techniques exist and may be applied to make the problem feasible. So an optimal basic solution exists together with optimal dual variables to be denoted by $y \in \mathbf{R}^m$ and $v \in \mathbf{R}$ according to the $m + 1$ rows of (3). By linear programming duality there exists a dual linear programming problem of the full master problem (2) with the variables (y, v) and with constraints

$$yA_1 x^i + v \ge cx^i \quad \text{for all } i \in P. \tag{4}$$

Also by linear programming we know that an optimal solution has been found for the full master problem (2) if and only if the dual solution (y, v) satisfies (4). This may of course be checked through examination of all extreme points x^i. Fortunately, this is not necessary and here comes the major idea behind the decomposition principle. Instead we consider the following linear programming problem, the so-called *subproblem*.

$$\begin{cases} u = \max & (c - yA_1)x \\ \text{s.t.} & A_2 x \le b_2 \\ & x \ge 0. \end{cases} \tag{5}$$

Decomposition Principle of Linear Programming, Table 1

Step 1	Calculate an optimal dual solution (y, v) of the restricted master problem (3).
Step 2	Determine an extreme point by solving the subproblem (5). If (6) is violated expand the index set $\bar{P}$ by including the extreme point and go to Step 1.
Step 3	An optimal solution has been obtained by the solution of the last master problem as $x = \sum_{i\in\bar{P}} x^i \lambda_i$

By assumption an optimal solution exists among the extreme points of X. Let $i^* \in P$ denote the index of an optimal extreme point. Observe that the objective function calculates the maximal value u of $cx^i - yA_1x^i$ among all extreme points x^i in X. Hence by (4) it remains to check if

$$u \leq v. \tag{6}$$

If so, then all constraints of (4) are satisfied and we may stop. Otherwise introduce the elements $(cx^{i^*}, A_1x^{i^*})$ as a new column in the restricted masterproblem (3) and continue by solving it.

The above discussion can be summarized into the *algorithm* in Table 1.

The number of extreme points in X is finite. Hence only a finite number of mutually different columns may be introduced in the restricted master problem. This implies that the algorithm must terminate in a finite number of steps.

The decomposition principle is suited to solve large scale problems. Moreover it has a nice economic interpretation. Consider a central level and a sublevel of a decentralized organization. The central level operates on the first set of constraints $A_1x \leq b_1$ and the sublevel on the second set of constraints $A_2x \leq b_2$. The right hand sides b_1, b_2 may be interpreted as resources for the central level and sublevel, respectively. During the course of the algorithm information is communicated from one level to the other. The central level solves the restricted master problem and as a result marginal prices y on central resources are communicated to the sublevel. The sublevel solves the subproblem in which the objective function incorporates the costs for utilization of the central resources. The sublevel then suggests activities x^i to be incorporated at central level. During the iterations no direct information about the coefficients in the constraints is communicated between the central level and the sublevel. Instead price information is communicated from the central level to the sublevel and the algorithm is the fundamental method among the so-called price-directive procedures.

In most applications the last set of constraints, $A_2x \leq b_2$ have a so-called *block-angular structure*, in which the variables are grouped into independent blocks. This implies that the subproblem separates into multiple independent problems. In an economic context the block-angular structure reflects a division of the sublevel into multiple independent sublevels, each of which communicates directly with the central level.

A counterpart of the present Dantzig–Wolfe decomposition procedure exists in the form of Benders decomposition. In linear programming they are dual in the sense that application of Benders decomposition on the dual program of the original problem (1) is equivalent to the direct application of the present procedure on (1).

See also

- ▶ Generalized Benders Decomposition
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Logic-Based Methods
- ▶ Simplicial Decomposition
- ▶ Simplicial Decomposition Algorithms
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Successive Quadratic Programming: Decomposition Methods

References

1. Bazaraa MS, Jarvis JJ, Sherali HD (1990) Linear programming and network flows. Wiley, New York
2. Dantzig GB, Wolfe P (1961) The decomposition algorithm for linear programs. Econometrica 29:767–778

Decomposition Techniques for MILP: Lagrangian Relaxation

Viswanathan Visweswaran
SCA Technologies LLC, Pittsburgh, USA

MSC2000: 90C90, 90C30

Article Outline

Keywords
Lagrangian decomposition
Aggregation Schemes
Practical Issues
Choosing Among Alternate Relaxations
Choice of Multipliers
Applications
See also
References

Keywords

Decomposition; Lagrangian relaxation; Lagrangian decomposition

A large number of combinatorial optimization problems can be viewed as potentially 'easy' problems to solve that are complicated by a set of side constraints. If the complicating constraints were removed, the resulting problem would have constraints possessing a high degree of structure, for which many efficient algorithms exist. One of the most attractive methods to exploit this property is the Lagrangian relaxation technique, in which the complicating constraints are dualized and then removed from the constraint set. This class of methods, originally proposed by various authors for a variety of problems, and later generalized in [3], has proven highly successful in solving otherwise difficult combinatorial problems. For an excellent introduction to the approach and its applications, see [1,2].

Consider the following problem (P):

$$\max f^\top x \tag{1}$$

such that

$$Ax \le b \tag{2}$$

$$Cx \le d \tag{3}$$

$$x \in X, \tag{4}$$

where x is an n-vector, b is an m-vector and d is a k-vector, and f, A and C have conformable dimensions. Some or all of the x variables can be integers (i. e. $X \subseteq Z^n$). It is assumed that there is a finite and nonempty set of solutions to the constraints in the problem (2)–(4). Let (LP) represent problem (P) with any integrality constraints in X removed.

The following notation is used in the sequel. For any problem $(\cdot)$, OS$(\cdot)$ is its optimal set, and $V(\cdot)$ represents its optimal value. For any set S, Co(S) represents the convex hull of the set.

The *Lagrangian relaxation* (LR$_u$) of (P) relative to the constraint set (2) and a conformable nonnegative vector u is defined as

$$\begin{cases} \max\limits_x & f^\top x + u(b - Ax) \\ \text{s.t.} & Cx \le d \\ & x \in X. \end{cases}$$

The problem

$$(\text{LR}) = \min_{u \ge 0} (\text{LR}_u)$$

is called the *Lagrangian dual* relative to (2). The constraints (2) are referred to as the 'dualized constraints', and u is the corresponding multiplier or dual vector. The constraints should be chosen so that the remaining set $C_x \le d$ possesses desirable structure. For example, (3) might only specify up per bounds on the variable, or might be a single 'knapsack' constraint of the form $\sum_{i=1}^n x_i \le 1$.

The first point to note about (LR$_u$) is that it always provides an upper bound for (P), i. e.

$$V(\text{LR}_u) \ge V(P).$$

This can easily be seen from the fact that $u \ge 0$ and $Ax \le b$ for any solution x which is optimal for (P). In practice, it is desirable to have $V(LR_u)$ as close to $V(P)$ as possible. Moreover, there is already an LP relaxation of (P), obtained by dropping the integrality requirement on x. How does $V(\text{LR}_u)$ relate to $V(\text{LP})$? To answer this question, consider the following relaxation of (P), denoted by (P^*):

$$\begin{cases} \max & f^\top x \\ \text{s.t.} & Ax \le b \\ & x \in \text{Co}\{Cx \le d\colon\ x \in X\}. \end{cases}$$

It can be shown ([3]) that

$$V(P) \le V(P^*) = V(\text{LR}) \le V(\text{LP}).$$

The equality for the optimal values of problems (P^*) and (LR) follows from the fact that they are duals of each other. Moreover, it can be shown that if the multipliers for the constraints obtained from solving the LP relaxation were used, the resulting Lagrangian relaxation provides a bound at least as tight as the bound from (LP). Also, if $\overline{u}$ is an optimal solution for (LR), with $A\overline{x} \leq b$ and $u(A\overline{x} - b) = 0$, then $\overline{x}$ is optimal for (P).

When are the inequalities above strict? This can be shown through the following *integrality property*, again due to [3]: The optimal value of (LR_u) is not changed by dropping the integrality condition on the x variables.

If the integrality property (also referred to as the *complementary slackness* property) holds, then

$$V(P) = V(P^*) = V(\mathrm{LR}) = V(\mathrm{LP}).$$

In this case, therefore, Lagrangian relaxation can do no better than the standard LP relaxation for (P). For a large number of practical problems, however, this property does not hold. This fact allows (LR_u) to be used in place of (LP) to provide lower bounds in a branch and bound algorithm.

Lagrangian decomposition

A drawback of the Lagrangian relaxation (LR) described above is that only one of the possibly many special structured constraint sets embedded in the problem can be exploited. This results in the loss of structure of all the dualized constraints. One way to avoid this is to use Lagrangian decomposition ([4,5,6,7]).

Introducing a new set of 'copy' constraints $y = x$, problem (P) is equivalent to

$$\max_{x,y} f^\top x$$

such that

$$Ay \leq b \tag{5}$$

$$Cx \leq d \tag{6}$$

$$y = x \tag{7}$$

$$x \in X, \qquad y \in Y, \tag{8}$$

where $X \subseteq Y$. Dualizing the 'copy' constraints (7) results in

$$\max f^\top x + v(y - x)$$

such that

$$Ay \leq b \tag{9}$$

$$Cx \leq d \tag{10}$$

$$x \in X, y \in Y, \tag{11}$$

which can be decomposed to the following problem (LD_v):

$$\begin{cases} \max\limits_x & F_1(x) \\ \text{s.t.} & Cx \leq d \\ & x \in X \end{cases} + \begin{cases} \max\limits_y & F_2(y) \\ \text{s.t.} & Ay \leq b \\ & y \in Y \end{cases},$$

where $F_1(x) = (f - v)^\mathsf{T} x$ and $F_2(y) = v^\mathsf{T} y$. The Lagrangian decomposition dual (LD) can then be defined to be

$$\min_{v \geq 0} V(\mathrm{LD}_v)$$

If $\overline{u}$ is an optimal solution to (LR), then, with $\overline{v} = \overline{u} \cdot A$, it can be shown that

$$V(\mathrm{LD}_{\overline{v}}) = V(\mathrm{LR}_{\overline{u}}) - \overline{u}(b - A\overline{y})$$

and therefore

$$V(\mathrm{LD}) \leq V(\mathrm{LR}).$$

It is possible to define an integrality property ([5]) such that if either the x- or the y-problem has the property, then V(LD) will be equal to the stronger of the bounds obtained from the two Lagrangian relaxations corresponding to each set of constraints.

Lagrangian decomposition (LD) has several advantages over (LR). Every constraint in the original problem appears in one of the subproblems. It thus avoids having to choose between the various sets of structured constraints. Secondly, as shown above, the bounds from (LD) can be tighter than those from (LR). Furthermore, the bound can be tightened by adding surrogate constraints (for example, a surrogate constraint from $Ax \leq b$ can be added to the x-problem in (LD). Thirdly, analogous to (LR), it can be shown that (LD) is really the

dual of a primal problem involving the optimization of the original objective function over the intersection of the convex hulls of the two constraint sets. Finally, empirical results suggest that when using heuristics based on Lagrangian decomposition, any intermediate solutions found in the solution of (LD) lead to better solutions for problem (P) as compared to solutions found by Lagrangian relaxation.

Aggregation Schemes

The main drawback of the Lagrangian decomposition method is that a large number of multipliers (v) are introduced, one for each of the copy variables. The calculation of v can be time consuming at each step. Moreover, the convergence of the scheme can be slowed significantly by the larger number of directions (for the multipliers) to search. In order to avoid this, an alternate approach that has been suggested [8] is to aggregate some or all of the variables using a simplified linear function, and then to dualize the resulting copy constraints. The purpose of the aggregation is to substantially reduce the number of dual variables, while still maintaining the constraint structure as in the standard decomposition.

Let $A \equiv [A_1 \mid A_2]$, and $x \equiv \binom{x^1}{x^2}$, with $x^1 \in \mathbf{R}^{n_1}$ and $x^2 \in \mathbf{R}^{n-n_1}$. Introduce the copy variables $y \equiv \binom{y^1}{y^2} = x$ and the constraints

$$x^1 = y^1 \quad \text{and} \quad A_2 x^2 = g(y^2),$$

where $g(\cdot)$ is the *aggregation function* (for example, $g(y^2) = A_2\, y^2$, or $g(y^2) = y^2$). Then, the problem (P) can be written as

$$\max f^\top x$$

such that

$$A_1 y^1 + g(y^2) \le b \tag{12}$$

$$Cx \le d \tag{13}$$

$$x^1 = y^1 \tag{14}$$

$$A_2 x^2 = g(y^2) \tag{15}$$

$$x \in X, \quad y \in Y, \tag{16}$$

where $X \subseteq Y$. Dualizing the constraints (14) and (15) leads to

$$\begin{cases} \max & f^\top x + w^1(y^1 - x^1) + w^2(g(y^2) - A_2 x^2) \\ \text{s.t.} & A_1 y^1 + g(y^2) \le b \\ & Cx \le d \\ & x \in X, \quad y \in Y, \end{cases}$$

which can be decomposed to the problem (LDA_w), given by

$$\begin{cases} \max\limits_x & F_1(x) \\ & Cx \le d \\ & x \in X \end{cases} + \begin{cases} \max\limits_y & F_2(y) \\ & A_1 y^1 + g(y^2) \le b \\ & y \in Y, \end{cases}$$

where $F_1(x) = f^\top x - w^1 x^1 - w^2 A_2 x^2$ and $F_2(y) = w^1 y^1 + w^2 g(y^2)$. The corresponding dual problem (LDA) is then defined by

$$\min_{w \ge 0} \mathrm{LDA}_w$$

It can then be proved that for any optimal solution of (LR) defined by $\overline{u} \in \mathrm{OS(LR)}$, with $\overline{w_1} = \overline{u} A_1$ and $\overline{w^2} = \overline{u}$,

$$V(\mathrm{LDA}_{\overline{w}}) \le V(\mathrm{LR}_{\overline{u}})$$

and therefore

$$V(\mathrm{LDA}) \le V(\mathrm{LR}).$$

This inequality is strict only if the second subproblem in (LDA_w) (i. e. the problem of maximizing $F_2(y)$) does not satisfy the integrality (*complementary slackness*) property. Moreover, this inequality holds only for the Lagrangian relaxation with those constraints that have been aggregated. Any other Lagrangian relaxation defined for problem (P) by dualizing other sets of constraints will not necessarily satisfy this inequality.

Similarly, defining $\overline{v} \equiv (\overline{w^1}, \overline{w^2})^\top$, it can also be easily shown that

$$V(\mathrm{LD}_{\overline{v}}) \le V(\mathrm{LDA}_{\overline{w}})$$

and therefore

$$V(\mathrm{LD}) \le V(\mathrm{LDA}).$$

In general, therefore, the bound obtained by aggregating some or all of the variables is stronger than the

bound obtained from Lagrangian relaxation but weaker than the bounds from standard Lagrangian decomposition. However, while the standard (LD) introduces n multipliers, (LDA) has $n_1 + m$, which can be considerably less depending on the number of dualized constraints. Moreover, any inherent problem structure is still maintained in (LDA). The aggregate formulation can thus be viewed as a reasonable compromise between tightness of the bounds and speed of solution. As in the case of standard decomposition, (LDA) can be defined by aggregating different subsets of variables. Unfortunately, it is not always apparent a priori which is the best choice for obtaining the tightest bound, and various alternatives may have to be tried in practice.

One possible method of exploiting the potential inequalities in these various bounds is as follows:

1) solve (LR) to obtain $\overline{u}$;
2) if the integrality property does not hold, set $\overline{w^1} = \overline{u}A_1$ and $\overline{w^2} = \overline{u}$, and solve ($\mathrm{LDA}_{\overline{u}}$); this problem is guaranteed to give a tighter bound than (LR);
3) set $\overline{v} \equiv \left(\frac{\overline{w^1}}{\overline{w^2}}\right)$, and solve ($\mathrm{LD}_{\overline{v}}$). Note that if the aggregate function $g(y^2)$ is of the form $A_2 y$, this step will not yield any improvement.

Practical Issues

Because (LR), (LD) and (LDA) can all provide tighter bounds than (LP), any one of the relaxations can be used in place of (LP) to provide upper bounds in a classical branch and bound algorithm to solve (P). Consequently, the choice of the relaxation scheme used, as well as the quality of the bounds obtained, is of considerable importance. These are discussed in some detail below.

Choosing Among Alternate Relaxations

Often, there are several choices for the constraints to be dualized. For example, consider the generalized assignment problem

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

such that

$$\sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, \ldots, n, \tag{17}$$

$$\sum_{j=1}^{n} a_{ij} x_{ij} \le b_i, \quad i = 1, \ldots, m, x_{ij} = 0 \text{ or } 1 \quad \forall i, j. \tag{18}$$

By dualizing the first set of constraints (17), this problem reduces to m knapsack problems. Conversely, dualizing (18) results in a generalized upper bound (GUB) problem in 0–1 variables. Which of the two relaxations should be used? There are two conflicting factors involved here, namely the tightness of the bounds and the ease of solution of the problem. It would be wise to select a relaxation that yields a problem that is fairly easy to solve, but not so easy that the bounds are very loose. In general, it is difficult to know this a priori. In some instances, however, the test of the integrality property can be useful. For example, in the generalized assignment problem, the integrality property holds for the second relaxation but not the first, suggesting that the second relaxation will yield a tighter bound.

For the case of the Lagrangian decomposition, this issue is not important since all constraints are maintained in one of the subproblems. However, if aggregation is being used, then again, it is in general hard to know a priori which variables to aggregate and which ones to copy. Often, the best solution is to try various alternatives and use the computational results to guide the choice.

Choice of Multipliers

It is clear that for (LR), the best choice for u is an optimal solution to the problem

$$\min_{u \ge 0} V(\mathrm{LR}_u)$$

since this will yield the tightest bound V(LR). Similarly, the best dual vectors v and w for (LD) and (LDA) are those from the optimal solutions to the respective dual problems. Unfortunately, these optimal values for the dual variables cannot be determined a priori, and therefore, an interactive procedure is the only viable approach to improv ing the value of u, v or w. Below, a couple of techniques for updating u for (LR) are discussed, but the methods are just as relevant for updating v and w for (LD) and (LDA).

In general, the function $V(\mathrm{LR}_u)$ is piecewise linear, convex and differentiable at all points except where

the Lagrangian problem has multiple optimal solutions. This observation has led to the development of *subgradient techniques* for the determining the u that minimizes $V(\mathrm{LR}_u)$. This method is similar to traditional gradient methods, except that at the nondifferentiable points, it chooses randomly from the set of optimal Lagrange solutions. Given an initial value u^0 (typically $u^0 = 0$), a sequence $\{u^k\}$ is generated by the formula

$$u^{k+1} = \max\{u^k + t^k(Ax^k - b), 0\},$$

where x^k is an optimal solution to (LR_{u^k}) and t^k is a scalar stepsize, generally designed to be a decreasing sequence converging to zero. It is not possible to prove optimality in this method, so usually it is terminated upon reaching a specified number of iterations. Because of its simplicity, the subgradient technique is generally the method of first choice when solving (LR).

An alternate way to update u is to use *dual descent* algorithms, also referred to as *multiplier adjustment* methods. In these methods, the sequence u^k is generated by

$$u^{k+1} = u^k + t^k d^k,$$

where d^k is an *descent direction*, determined from the directional derivative of $V(\mathrm{LR}_{u^k})$ using a finite set of directions. Typically, the direction of steepest descent is chosen, and the stepsize t^k is the one that minimizes $V(\mathrm{LR}_{u^k + td^k})$. Unlike subgradient optimization, this procedure guarantees monotonic bound improvement. Moreover, it may only adjust a few multipliers at each iteration, resulting in improved computational performance. However, for general problems, the set of directions to choose from can be very large, resulting in very poor descent. It is therefore essential to tailor these methods to particular problems to exploit their structure in determining the set of directions.

Applications

Lagrangian relaxation and decomposition have been successfully applied to solve a large number of practical combinatorial problems. These include the generalized assignment problem, capacitated facility location problem, the traveling salesman problem and instances of the general mixed integer programming problem. For each of these problems, the constraints contain well-understood structures such as knapsack, spanning tree and generalized upper bound constraints, thus facilitating the dualization of the other complicating constraints. For a number of problems, these techniques represent the best available solution method. Computational results for these and other problems indicate that the bounds provided by (LR) and (LD) can be extremely sharp. These results have led to (LD) and (LDA) being considered among the best available solution methods for solving these problems.

See also

- Branch and Price: Integer Programming with Column Generation
- Integer Linear Complementary Problem
- Integer Programming
- Integer Programming: Algebraic Methods
- Integer Programming: Branch and Bound Methods
- Integer Programming: Branch and Cut Algorithms
- Integer Programming: Cutting Plane Algorithms
- Integer Programming Duality
- Integer Programming: Lagrangian Relaxation
- Lagrange, Joseph-Louis
- Lagrangian Multipliers Methods for Convex Programming
- LCP: Pardalos–Rosen Mixed Integer Formulation
- Mixed Integer Classification Problems
- Multi-objective Integer Linear Programming
- Multi-objective Mixed Integer Programming
- Multi-objective Optimization: Lagrange Duality
- Multiparametric Mixed Integer Linear Programming
- Parametric Mixed Integer Nonlinear Optimization
- Set Covering, Packing and Partitioning Problems
- Simplicial Pivoting Algorithms for Integer Programming
- Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- Stochastic Integer Programs
- Time-Dependent Traveling Salesman Problem

References

1. Fisher ML (1981) The Lagrangean relaxation method for solving integer programming problems. Managem Sci, 27(1):1–18
2. Fisher ML (1985) An applications oriented guide to Lagrangean relaxation. Interfaces 15(2):10–21

3. Geoffrion AM (1974) Lagrangian relaxation and its uses in integer programming. Math Program Stud 2:82–114
4. Glover F, Klingman D (1988) Layering strategies for creating exploitable structure in linear and integer programs. Math Program 40:165–182
5. Guignard M, Kim S (1987) Lagrangean decomposition: A model yielding stronger Lagrangean bounds. Math Program 39:215–228
6. Guignard M, Kim S (1987) Lagrangean decomposition for integer programming: Theory and applications. RAIRO Oper Res 21(4):307–323
7. Jörsten K, Näsberg M, Smeds P (1985) Variable splitting–a new Lagrangean relaxation approach to some mathematical programming models. Techn Report Dept Math Linkoping Inst Technol, Sweden MAT-R-85-04
8. Reinoso H, Maculan N (1992) Lagrangean decomposition in integer linear programming: A new scheme. INFOR 30:1–5

De Novo Protein Design Using Flexible Templates

Ho Ki Fung, Christodoulos A. Floudas
Department of Chemical Engineering,
Princeton University, Princeton, USA

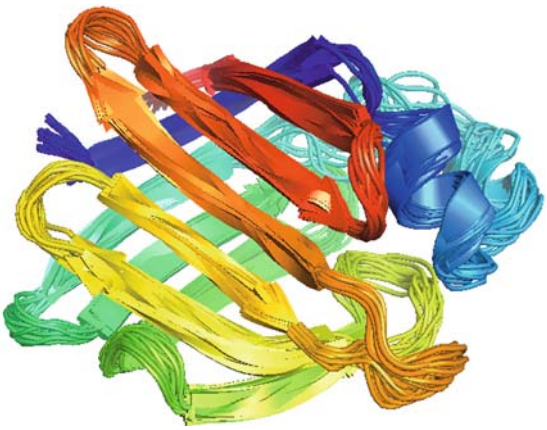
MSC2000: 92D20, 46N10, 90C10

Article Outline

Introduction
Flexible Template via Multiple Discrete Templates and Discrete Rotamers
Approaches Which Separate Sequence Selection and Backbone Movement
Approaches Which Iterate Between Sequence Space and Structure Space
Flexible Template via Continuum Template and Discrete Rotamers
Successes
Flexible Template via Continuum Template and NMR Structure Refinement
Successes
References

Introduction

The assumption of a fixed template for de novo peptide and protein design is highly questionable [41], as protein is commonly known to exhibit backbone flexibility, as illustrated by the superposition of NMR structures in Fig. 1. De novo design templates were observed to allow residues that would not have been permissible had the backbone been fixed [34]. The Mayo group claimed that their ORBIT protein design program was robust against 15% change in the backbone. Nevertheless, they found in a later case study on T4 lysozyme that core repacking to stabilize the fold was difficult to achieve without considering a flexible template [37]. The secondary structures of α-helices and β-sheets actually display twisting and bending in the fold, and Emberly et al. [6,7] applied principal component analysis of database protein structures to quantify the degree and modes of their flexibilities.



De Novo Protein Design Using Flexible Templates, Figure 1
Template flexibility as illustrated by the superposition of the 20 NMR structures of apo intestinal fatty acid binding protein (Protein Data Bank code 1AEL)

In this chapter we classify the various methods of incorporating backbone flexibility into the design template into three main types according to their treatment of the backbone and side-chain conformations. The first type involves considering a set of multiple discrete templates and performing de novo design with discrete rotamers on each of the templates under the fixed-backbone assumption. The second type considers a continuum template by means of algebraic parameterization of the backbone and variation of the parameters to allow for backbone movement during sequence selection. However, it still employs rotamer libraries to simplify the side-chain conformations. Through novel sequence selection formulations [14] and pairwise contact potentials which are discretized over distance bins [35,44,45], the third type considers a continuum design template in which the Cα–Cα distances and dihedral angles assume

continuous values between upper and lower bounds observed from the template structures [9], and confirms sequence specificity to the target fold based on these bounded continuous distances and angles via NMR structure refinement methods [16,17] rather than the discrete rotamer approach. For each category we will quote some examples of successes of de novo peptide and protein design.

Flexible Template via Multiple Discrete Templates and Discrete Rotamers

By incorporating protein backbone flexibility via discrete templates and discrete rotamers, de novo protein design frameworks either separate sequence selection and backbone movement explicitly or iterate between sequence space and structure space [3]. Notice that in both cases, the sequence search methods outlined in the previous section are all applicable, as fixed backbones and discrete rotamers are still assumed.

Approaches Which Separate Sequence Selection and Backbone Movement

These approaches consider an ensemble of fixed backbones, searches for sequences for each of them assuming a fixed template, and finally identify the best solutions from all the results. Successes using different kinds of search algorithms include the ones described next.

Successes Using Dead-End Elimination By varying the supersecondary structure parameters, Ross et al. [46] and Su and Mayo [48] generated several sets of perturbed backbones from the native structure and redesigned the core of the $\beta 1$ domain of the streptococcal protein using the DEE algorithm under the fixed-template assumption for each backbone. Confirmed by NMR experiments, six of the seven sequences tested folded into nativelike structure.

Successes Using the Self-Consistent Mean Field Method Kono and Saven [29] applied their self-consistent mean field based protein combinatorial library design strategy to a set of similar backbone structures to obtain new sequences that are robust to distance changes in the template for the immunoglobulin light chain-binding domain of protein L.

Successes of Monte Carlo Methods/Genetic Algorithms The Pande group generated families of 100 fixed templates within 1 Å root-mean-square deviation (rmsd) from the initial backbone using a Monte Carlo method. With these fixed-template ensembles, they performed de novo design, which was based on genetic algorithms, on their Genome@home distributed grid system for 253 naturally occurring proteins. They obtained sequences that exhibited higher diversity than the corresponding natural sequence alignments, as well as good agreement on the sequence entropies of the designed sequences from the same template family [32,33].

In order to incorporate protein flexibility, Kraemer-Pecore et al. [30] executed a Monte Carlo simulation to generate 30 fixed backbones that were within 0.3 Å rmsd of the initial template. A genetic-algorithm-based sequence prediction algorithm [43] which combines filtering and sampling rotamers and energy minimization was then employed for sequence search on each template under the fixed backbone assumption. The work led to the identification of a sequence that folded into the WW domain.

In designing protein conformational switches, Ambroggio and Kuhlman [1,2] also used the Monte Carlo based RosettaDesign to search for sequences for multiple fixed-template structures.

Approaches Which Iterate Between Sequence Space and Structure Space

There are two good examples which belong to this class. The first example is a genetic algorithm/Monte Carlo based framework used by Desjarlais and Handel [5], in which a starting population of backbones is generated by small angle perturbations to the template, rotamers are randomly selected on each backbone, and a genetic algorithm is subsequently used which exchanges not only rotamers but also backbone torsional information in recombination. The framework is ended with a Monte Carlo stage which refines the backbone structures. Using this novel approach, Desjarlais and Handel [5] designed three new core variants of the protein 434 cro. They also compared results on 434 cro and T4 lysozyme with those obtained earlier using fixed-template models and found that they were similar, given that the fixed-

template models scan over a much larger rotamer space.

The second one was proposed by Kuhlman et al. [31] and Saunders and Baker [31,47]. Their method starts with a set of initial backbones, searches by a Monte Carlo method for the sequence with the lowest energy for each of them, performs atomic-resolution structure prediction for the sequences to allow shifts in the structure space, and continues until the number of iterations hits a predetermined number. They successfully designed a new sequence for Top 7, a 93-residue α/β protein with a novel fold [31]. They also claimed that the new method better captures sequence variation than approaches that separate sequence selection and backbone movement explicitly.

Flexible Template via Continuum Template and Discrete Rotamers

This method of constituting a continuum template via backbone parameterization and performing sequence search from rotamer libraries was proposed by Harbury et al. [18,19,40]. On the basis of the algebraic parameterization equations developed for coiled-coils by Crick [4], they allowed backbone movement by treating the parameters as variables during sequence search for energy minimization, which was in turn done by the local optimization methods of steepest descent minimization and adopted-basis Newton–Raphson minimization.

Successes

Harbury et al. [18,19,40] adopted this approach to design a family of α-helical bundle proteins with right-handed superhelical twist. The crystal structure of the designed sequences with the optimal specificity was experimentally validated to match the design template.

Flexible Template via Continuum Template and NMR Structure Refinement

Considering discrete rotamers is certainly not the best approach to adopt in de novo design, as about 15% of side-chain conformations are not represented by common rotamer libraries [8]. A recent two-stage de novo design approach proposed by the Floudas group [10,11,26,27] considers a continuum design template without using discrete rotamers for the possible side-chain conformations. The first stage selects a rank-ordered list of low-energy sequences using novel quadratic assignment-like models [13,14] driven by pairwise residue contact potentials, which were developed by the group by solving a linear programming parameter estimation problem, requiring that the native conformations for a large training set of 1250 proteins be ranked energetically more favorably than their high-resolution decoys [35,44,45]. The forcefields developed were found to produce very good Z scores in recognizing the native folds for a large test set of proteins [35,44,45]. Rather than being continuous, the dependence of contact potential on distance is discretized into bins. This designed feature serves to make the energy objective function insensitive to a limited degree of backbone movement. For example, in the high resolution C^{α}-C^{α} forcefield [44], if the pair of amino acids selected at two positions i and k, which are 3.5 Å apart in the template, are Arg and Glu, respectively, their energy contribution to the objective function is Minus 7.77 kcal/mol. Despite small distance variations, this energy value is constant for all Arg–Glu interactions as long as the C^{α} positions of the two residues are 3–4 Å (bin 1) apart. To perform sequence selection based on a flexible template of multiple structures, Fung et al. [14] also developed two novel formulations: a weighted model which considers the distance between any two positions as the weighted average of their distances in all structures, and a binary distance bin model that decides which bin the distance falls into during energy optimization. The latter approach is in a sense similar to the backbone parameterization approach of Harbury et al. [18,19,40] in which there are distance variables associated with the backbone.

The second stage of the approach confirms fold specificity of the sequences generated in the first stage based on a full-atomistic forcefield. The group used to perform the task via ASTRO-FOLD [20,21,22,23,24,25,28,36], a protein structure prediction method via global optimization. Conformational ensembles are generated for each sequence under two sets of conditions. In the first circumstance, the structure is constrained to vary, with some imposed fluctuations, around the template structure. In the second condition, a free folding calculation is performed for which only a limited number of restraints (e. g., disulfide bridges), but not the un-

derlying template structure, are enforced. The relative fold specificity of the sequence, f_{spec}, can be found by summing the statistical weights for those conformers from the free folding simulation that resemble the template structure (denoted as set *temp*), and dividing this sum by the summation of statistical weights for all conformers from the free folding simulation (denoted as set *total*):

$$f_{\text{spec}} = \frac{\sum_{i \in temp} \exp[-\beta E_i]}{\sum_{i \in total} \exp[-\beta E_i]}$$

where $\exp[-\beta E_i]$ is the statistical weight for conformer *i*.

Note that in this nonrotamer approach, in both the template-constrained and the free folding calculations, all continuous C^{α}–C^{α} and angle values between upper and lower bounds input by the user are considered in sampling the conformers. True backbone flexibility [9] is thus conserved.

Lately the Floudas group developed an approximate fold validation method which is computationally less expensive than ASTRO-FOLD. Through the CYANA 2.1 software for NMR structure refinement [16,17], an ensemble of several hundred conformers is generated for both a new sequence from the first stage and the native sequence. The energies of the conformers are then minimized using TINKER [42], and the fold specificity of the new sequence is calculated using the formula

$$f_{\text{spec}} = \frac{\sum_{i \in \text{conformers for new sequence}} \exp[-\beta E_i]}{\sum_{i \in \text{conformers for native sequence}} \exp[-\beta E_i]}$$

based on the assumption that the fold specificity to the flexible template is unity for the native sequence.

Like the fold-validation method via ASTRO-FOLD, all continuous distance and dihedral angle values between their upper and lower bounds, which are input into CYANA on the basis of observations about the template structures, are considered in generating the conformers. This distinguishes the method from the common rotamer approach in which only discrete side-chain conformations are allowed.

Successes

The novel two-stage de novo strategy was applied to (1) the design of new sequences for compstatin, a synthetic 13-residue cyclic peptide that binds to complement protein 3 (C3) and inhibits the activation of the complement system (part of innate immunity) [10,26,27,38,39], (2) the design of a potential peptide-drug candidate derived from the C-terminal sequence of the C3a fragment of C3 [15], and (3) the full sequence of human β-defensin-2, a 41-residue cationic peptide in the immune system [12]. In the case of the compstatin redesign, sequences with 16-fold and 45-fold improvement in specificity over the native sequence were confirmed in experiments [26,27]. For the design of the peptide drug from C3a, the best sequence identified corresponds to 15-fold improvement [15].

References

1. Ambroggio XI, Kuhlman B (2006) Computational design of a single amino acid sequence that can switch between two distinct protein folds. J Am Chem Soc 128:1154–1161
2. Ambroggio XI, Kuhlman B (2006) Design of protein conformational switches. Curr Opin Struct Biol 16:525–530
3. Butterfoss GL, Kuhlman B (2006) Computer-based design of novel protein structures. Annu Rev Biophys Biomol Struct 35:49–65
4. Crick FHC (1953) The fourier transform of a coiled-coil. Acta Crystallogr 6:685–689
5. Desjarlais J, Handel T (1999) Side chain and backbone flexibility in protein core design. J Mol Biol 290:305–318
6. Emberly E, Mukhopadhyay R, Tang C, Wingreen N (2003) Flexibility of α-helices: Results of a statistical analysis of database protein structures. J Mol Biol 327:229–237
7. Emberly E, Mukhopadhyay R, Tang C, Wingreen N (2004) Flexibility of β-sheets: Principal component analysis of database protein structures. Proteins: Struct Funct Genet 55:91–98
8. Filippis VD, Sander C, Vriend G (1994) Predicting local structural-changes that result from point mutations. Prot Eng 7:1203–1208
9. Floudas CA (2005) Research challenges, opportunities and synergism in systems engineering and computational biology. AIChE J 51:1872–1884
10. Floudas CA, Fung HK (2006) Mathematical modeling and optimization methods for de novo protein design. In: Rigoutsos I, Stephanopoulos G (eds) Systems Biology, vol II, Oxford University Press, pp 42–66
11. Floudas CA, Fung HK, Morikis D, Taylor MS, Zhang L (2007) Overcoming the key challenges in de novo protein design: Enhancing computational efficiency and incorporating true backbone flexibility. In: Mondaini R (ed) Modeling of Biosystems: An Interdisciplinary Approach. Springer, Berlin
12. Fung HK, Floudas CA, Morikis D, Taylor MS, Zhang L (2007) Toward full-sequence de novo protein design with flexible templates for human beta-defensin-2. Biophys J (in print)

13. Fung HK, Rao S, Floudas CA, Prokopyev O, Pardalos PM, Rendl F (2005) Computational comparison studies of quadratic assignment like formulations for the in silico sequence selection problem in de novo protein design. J Comb Optim 10:41–60
14. Fung HK, Taylor MS, Floudas CA (2007) Novel formulations for the sequence selection problem in de novo protein design with flexible templates. Optim Methods Softw 22:51–71
15. Fung HK, Taylor MS, Floudas CA, Morikis D, Lambris JD (2007) Redesigning complement 3a based on flexible templates from both x-ray crystallography and molecular dynamics simulation. in preparation
16. Guntert P (2004) Automated nmr structure calculation with cyana. methods mol biol. J Mol Biol 278:353–378
17. Guntert P, Mumenthaler C, Wuthrich K (1997) Torsion angle dynamics for nmr structure calculation with the new program dyana. J Mol Biol 273:283–298
18. Harbury P, Plecs J, Tidor B, Alber T, Kim P (1998) High-resolution protein design with backbone freedom. Science 282:1462–1467
19. Harbury P, Tidor B, Alber T, Kim P (1995) Repacking protein cores with backbone freedom: Structure prediction for coiled coils. Proc Natl Acad Sci USA 92:8408–8412
20. Klepeis JL, Floudas CA (1999) Free energy calculations for peptides via deterministic global optimization. J Chem Phys 110:7491–7512
21. Klepeis JL, Floudas CA (2002) Ab initio prediction of helical segments in polypeptides. J Comput Chem 23:245–266
22. Klepeis J, Floudas CA (2003) Astro-fold: A combinatorial and global optimization framework for ab initio prediction of three-dimensional structures of proteins from the amino acid sequence. Biophys J 85:2119–2146
23. Klepeis JL, Floudas CA (2003) Prediction of β-sheet topology and disulfide bridges in polypeptides. J Comput Chem 24:191–208
24. Klepeis JL, Floudas CA (2003) Ab initio tertiary structure prediction of proteins. J Global Optim 25:113–140
25. Klepeis JL, Floudas CA, Morikis D, Lambris J (1999) Predicting peptide structures using nmr data and deterministic global optimization. J Comput Chem 20:1354–1370
26. Klepeis JL, Floudas CA, Morikis D, Tsokos CG, Argyropoulos E, Spruce L, Lambris JD (2003) Integrated structural, computational and experimental approach for lead optimization: Deisgn of compstatin variants with improved activity. J Am Chem Soc 125:8422–8423
27. Klepeis JL, Floudas CA, Morikis D, Tsokos CG, Lambris JD (2004) Design of peptide analogs with improved activity using a novel de novo protein design approach. Ind Eng Chem Res 43:3817–3826
28. Klepeis J, Wei Y, Hecht M, Floudas C (2005) Ab initio prediction of the three-dimensional structure of a de novo designed protein: A double-blind case study. Proteins 58:560–570
29. Kono H, Saven J (2001) Statistical theory of protein combinatorial libraries: Packing interactions, backbone flexibility, and the sequence variability of a main-chain structure. J Mol Biol 306:607–628
30. Kraemer-Pecore C, Lecomte J, Desjarlais J (2003) A de novo redesign of the ww domain. Protein Sci 12:2194–2205
31. Kuhlman B, Dantae G, Ireton G, Verani G, Stoddard B, Baker D (2003) Design of a novel globular protein fold with atomic-level accuracy. Science 302:1364–1368
32. Larson SM, England JL, Desjarlais JR, Pande VS (2002) Thoroughly sampling sequence space: Large-scale protein design of structural ensembles. Protein Sci 11:2804–2813
33. Larson SM, Garg A, Desjarlais JR, Pande VS (2003) Increased detection of structural templates using alignments of designed sequences. Proteins 51:390–396
34. Lim W, Hodel A, Sauer R, Richards F (1994) The crystal structure of a mutant protein with altered but improved hydrophobic core packing. Proc Natl Acad Sci USA 91:423–427
35. Loose C, Klepeis J, Floudas C (2004) A new pairwise folding potential based on improved decoy generation and side chain packing. Proteins: Struct Funct Bioinformatics 54:303–314
36. McAllister S, Mickus B, Klepeis J, Floudas C (2006) Novel approach for alpha-helical topology prediction in globular proteins: Generation of interhelical restraints. Proteins 65:930–952
37. Mooers B, Datta D, Baase W, Zollars E, Mayo S, Matthews B (2003) Repacking the core of t4 lysozyme by automated design. J Mol Biol 332:741–756
38. Morikis D, Floudas CA, Lambris J (2005) Structure-based integrative computational and experimental approach for the optimization of drug design. Lect Notes Comput Sci 3515:680–688
39. Morikis D, Soulika A, Mallik B, Klepeis J, Floudas C, Lambris J (2004) Improvement of the anti-c3 activity of compstatin using rational and combinatorial approaches. Biochem Soc Trans 32:28–32
40. Plecs J, Harbury PB, Kim P, Alber T (2004) Structural test of the parameterized-backbone method for protein design. J Mol Biol 342:289–297
41. Pokala N, Handel T (2001) Review: Protein design-where we were, where we are, where we're going. J Struct Biol 134:269–281
42. Ponder J (1998) TINKER, software tools for molecular design. Department of Biochemistry and Molecular Biophysics, Washington University School of Medicine: St. Louis
43. Raha K, Wollacott A, Italia M, Desjarlais J (2000) Prediction of amino acid sequence from structure. Protein Sci 9:1106–1119
44. Rajgaria R, McAllister SR, Floudas CA (2006) A novel high resolution c^{α}-c^{α} distance dependent force field based on a high quality decoy set. Proteins: Struct, Funct, Bioinformatics 65:726–741

45. Rajgaria R, McAllister SR, Floudas CA (2007) Improving the performance of a high resolution distance dependent force field by including protein side chains. Proteins: Struct Funct Bioinformatics (in print)
46. Ross S, Sarisky C, Su A, Mayo S (2001) Designed protein g core variants fold to native-like structures: Sequence selection by orbit tolerates variation in backbone specification. Protein Sci 10:450–454
47. Saunders CT, Baker D (2005) Recapitulation of protein family divergence using flexible backbone protein design. J Mol Biol 346:631–644
48. Su A, Mayo S (1997) Coupling backbone flexibility and amino acid sequence selection in protein design. Protein Sci 6:1701–1707

De Novo Protein Design Using Rigid Templates

HO KI FUNG, CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering,
Princeton University, Princeton, USA

Article Outline

Introduction
Sequence Search Methods
  Deterministic Methods
  Stochastic Methods
References

Introduction

Computational protein design efforts were first initiated with the premise that the three-dimensional coordinates of the design template or backbone were fixed. This simplification was first proposed in [39], and was appealing because it greatly reduced the combinatorial complexity of the search. Together with consideration of only a limited set of most frequently observed side-chain conformations called rotamers [29,40], the assumption enhanced the efficiency of the initial de novo design efforts, most of which focused on protein cores [5,16,32,41,42], in exploring search spaces. The reason why protein cores were selected instead of the boundary or surface regions was based on the thesis that protein folding is primarily driven by hydrophobic collapse, and thus a good core tends to provide a well-folded and stable structure for the de novo designed protein [10]. The scope of the de novo design encompassed intermediate and surface residues in subsequent years, and obviously the problem became more challenging. In this chapter, we outline the different deterministic and stochastic methods that search for sequences specific to the fixed rigid design template. It should be noted that they all discretize the side-chain conformational space into rotamers for tractability of the search problem. After the introduction of each method we also review examples of successes.

Sequence Search Methods

De novo design algorithms can be classified into two main categories, namely, deterministic and stochastic [8]. The two main methods that fall into the deterministic category are dead-end elimination (DEE) and self-consistent mean field (SCMF), whereas the two major stochastic type frameworks are Monte Carlo and genetic algorithms. Some methods search for low-energy sequences, whereas others assign probability to each of the 20 amino acids for each design position in a sequence in order to maximize the conformational entropy.

Deterministic Methods

The Dead-End Elimination Criteria DEE, which is arguably the most popular rotamer search algorithm, operates on the basis of the systematic elimination of rotamers that cannot be parts of the sequence with the lowest energy. The energy function in DEE is written in the form of the sum of an individual term (rotamer–template) and a pairwise term (rotamer–rotamer):

$$E = \sum_{i=1}^{N} E(i_a) + \sum_{i=1}^{N-1} \sum_{j>i}^{N} E(i_a, j_b) , \qquad (1)$$

where $E(i_a)$ is the rotamer–template energy for rotamer i_a of amino acid i, $E(i_a, j_b)$ is the rotamer–rotamer energy of rotamer i_a and rotatmer j_b of amino acids i and j, respectively, and N is the total number of positions. The original DEE pruning criterion is based on the concept that if the pairwise energy between rotamer i_a and rotamer j_b is higher than that between rotamer i_c and rotamer j_b for all rotamer j_b in a certain rotamer set $\{B\}$, then rotamer i_a cannot be in the global energy minimum conformation and thus can be eliminated. It was

proposed in [9] and can be expressed in the following mathematical form:

$$E(i_a) + \sum_{j\neq i}^{N} E(i_a, j_b) \;>\; E(i_c) + \sum_{j\neq i}^{N} E(i_c, j_b) \;\; \forall\{B\}. \tag{2}$$

Rotamer i_a can be pruned if the above holds true. Bounds implied by (1) can be utilized to generate the following computationally more tractable inequality [9]:

$$E(i_a) \;+\; \sum_{j\neq i}^{N} \min_b E(i_a, j_b) \;>\; E(i_c) \;+\; \sum_{j\neq i}^{N} \max_b E(i_c, j_b)\,. \tag{3}$$

The above equations for eliminating rotamers at a single position (or singles) can be extended to eliminating rotamer pairs at two distinct positions (doubles), rotamer triplets at three distinct positions (triples), or above [9,37]. In the case of doubles, the equation becomes

$$\varepsilon(i_a, j_b) \;+\; \sum_{k\neq i,j}^{N} \min_c \varepsilon(i_a, j_b, k_c) \;>\; \varepsilon(i_{a'}, j_{b'}) \;+\; \sum_{k\neq i,j}^{N} \max_c \varepsilon(i_{a'}, j_{b'}, k_c)\,, \tag{4}$$

where ε is the total energy of rotamer pairs:

$$\varepsilon(i_a, j_b) \;=\; E(i_a) \;+\; E(j_b) \;+\; E(i_a, j_b)\,, \tag{5}$$

$$\varepsilon(i_a, j_b, k_c) \;=\; E(i_a, k_c) \;+\; E(j_b, k_c)\,. \tag{6}$$

It determines a rotamer pair i_a and j_b which always contributes higher energies than rotamer pair $i_{a'}$ and $j_{b'}$ for all possible rotamer combinations. Goldstein [14] improved the original DEE criterion by stating that rotamer i_a can be pruned if the energy contribution is always reduced by an alternative rotamer i_c:

$$E(i_a) \;-\; E(i_c) \;+\; \sum_{j\neq i}^{N} \min_b [E(i_a, j_b) - E(i_c, j_b)] \;>\; 0\,. \tag{7}$$

This can be generalized to the use of a weighted average of C rotamers i_c to eliminate i_a [14]:

$$E(i_a) - \sum_{c=1,\ldots,C} w_c E(i_c) \;+\; \sum_{j\neq i}^{N} \min_b [E(i_a, j_b) \;-\; \sum_{c=1,\ldots,C} w_c E(i_c, j_b)] \;>\; 0\,. \tag{8}$$

Lasters et al. [25] proposed that the most suitable weights w_c can be determined by solving a linear programming problem.

In addition to these criteria proposed by Goldstein [14], Pierce et al. [38] introduced the split DEE, which splits the conformational space into partitions and thus eliminated the dead-ending rotamers more efficiently:

$$E(i_a) \;-\; E(i_c) \;+\; \sum_{j,j\neq k\neq i}^{N} \{\min_{a'} [E(i_a, j_{a'}) - E(i_c, j_{a'})]\} \;+\; [E(i_a, k_{b'}) - E(i_c, k_{b'})] \;>\; 0\,. \tag{9}$$

In general, n splitting positions can be assigned for more efficient but computationally expensive rotamer elimination:

$$E(i_a) \;-\; E(i_c) \;+\; \sum_{j,j\neq k_1,\ldots,kn\neq i}^{N} \{\min_{a'} [E(i_a, j_{a'}) - E(i_c, j_{a'})]\} \;+\; \sum_{k=k_1,\ldots,kn} [E(i_a, k_{b'}) - E(i_c, k_{b'})] \;>\; 0\,. \tag{10}$$

Looger and Hellinga [27] also introduced the generalized DEE by ranking the energy of rotamer clusters instead of that of individual rotamers and increased the ability of the algorithm to deal with higher levels of combinatorial complexity. Further revisions and improvements on DEE were performed by Wernisch et al. [47] and Gordon et al. [15].

Being deterministic in nature, the different forms of DEE reviewed above all yield the same globally optimal solution upon convergence.

Successes Using Dead-End Elimination: Based on operating the DEE algorithm on a fixed template, the Mayo group devised their optimization of rotamers

by an iterative technique (ORBIT) program and applied it to numerous de novo protein designs. Examples are the full-sequence design of the $\beta\beta\alpha$ fold of a zinc finger domain [6], improvement of calmodulin binding affinity [45], full core design of the variable domains of the light and heavy chains of catalytic antibody 48G7 FAB, full core/boundary design, full surface design, and full-sequence design of the $\beta 1$ domain of protein G [15], as well as the redesign of the core of T4 lysozyme [32]. They also adjusted secondary structure parameters to build the "idealized backbone" and used it as a fixed template to design an α/β-barrel protein [33]. The Hellinga group applied DEE with a fixed backbone structure to introduce iron and oxygen binding sites into thioredoxin [2,3], design receptor and sensor proteins with novel ligand-binding functions [28], and confer novel enzymatic properties onto ribose-binding protein [11].

The Self-Consistent Mean-Field Method The SCMF optimization method is an iterative procedure that predicts the values of the elements of a conformational matrix $P(i,a)$ for the probability of a design position i adopting the conformation of rotamer a. Note that $P(i,a)$ sums to unity over all rotamers a for each position i. Koehl and Delarue [19] were among those who introduced such a method for protein design. They started the iteration with an initial guess for the conformational matrix, which assigns equal probability to all rotamers:

$$P(i,a) = \frac{1}{A} \quad a = 1, 2, \ldots, A\,. \tag{11}$$

Most importantly, they applied the mean-field potential, $E(i,a)$, which depends on the conformational matrix $P(i,a)$:

$$E(i,a) = U(x_{ia}) + U(x_{ia}, x_0) + \sum_{j=1, j\neq i}^{N} \sum_{b=1}^{B} P(j,b) U(x_{i_a}, x_{j_b})\,, \tag{12}$$

where x_0 corresponds to the coordinates of atoms in the fixed template, and x_{i_a} and x_{j_b} correspond to the coordinates of the atoms of position i assuming the conformation of rotamer a and those of position j assuming the conformation of rotamer b, respectively. The classical Lennard-Jones (12-6) potential can be used to describe potential energy U [19]. The conformational matrix can be subsequently updated using the mean-field potential and the Boltzmann law:

$$P_1(i,a) = \frac{e^{\frac{-E(i,a)}{RT}}}{\sum_{a=1}^{A} e^{\frac{-E(i,a)}{RT}}}\,. \tag{13}$$

The update on $P(i,a)$, namely, $P_1(i,a)$, can then be used to repeat the calculation of the mean-field potential and another update until convergence is attained. Koehl and Delarue [19] set the convergence criterion to be 10^{-4} to define self-consistency. They also proposed the introduction of memory of the previous step to minimize oscillations during convergence:

$$P(i,a) = \lambda P_1(i,a) + (1-\lambda) P(i,a)\,, \tag{14}$$

with the optimal step size λ to be 0.9 [19].

The Saven group [12,24,44,48] extended the SCMF theory and formulated de novo design as an optimization problem maximizing the sequence entropy subject to composition constraints and mean-field energy constraints. In addition to the site probabilities, their method also predicts the number of sequences for a combinatorial library of arbitrary size for the fixed template as a function of energy.

It should be highlighted that though deterministic in nature, the SCMF method does not guarantee convergence to the global optimal solution [26].

Successes Using the Self-Consistent Mean-Field Method Koehl and Delarue [20] applied the SCMF approach to design protein loops. In their optimization procedure, they first selected the loop fragment from a database with the highest site probabilities. Then they placed side chains on the fixed loop backbone from a rotamer library. Kono and Doi [23] also used an energy minimization with an automata network, which bears some resemblance to the SCMF method, to design the cores of the globular proteins of cytochrome b_{562}, triosephosphate isomerase, and barnase. The SCMF method is related to the design of combinatorial libraries of new sequences with good folding properties, which was reviewed in several papers [17,34,35,43].

Stochastic Methods

The fact that de novo design is nondeterministic polynomial-time hard [13,36] means that in the worst

case the time required to solve the problem scales non-polynomially with the number of design positions. As the problem complexity exceeds a certain level, deterministic methods may reach their limits and in such instances we may have to resort to stochastic methods, which perform searches for only locally optimal solutions. Monte Carlo methods and genetic algorithms are the two most commonly used types of stochastic methods for de novo protein design.

Monte Carlo Methods Different variants of the Monte Carlo methods have been applied for sequence design. In the classic Monte Carlo method, mutation is performed at a certain position in the sequence and energies of the sequence in the fixed template are calculated before and after the mutation. This usually involves the use of discrete rotamer libraries to simplify the consideration of possible side-chain conformations. The new sequence after mutation is accepted if the energy becomes lower. If the energy is higher, the Metropolis acceptance criterion [30] is used

$$p_{\text{accept}} = \min(1, \exp(-\beta \Delta E)) \quad \beta = \frac{1}{kT}, \qquad (15)$$

and the sequence is updated if p_{accept} is larger than a random number uniformly distributed between 0 and 1.

In the configurational bias Monte Carlo method, at each step a local energy is used which does not include those positions where a mutation has not been attempted [49]. Cootes et al. [4] reported that the method was more efficient at finding good solutions than the conventional Monte Carlo method, especially for complex systems. Zoz and Savan [49] also devised the mean-field biased Monte Carlo method which biases the sequence search with predetermined site probabilities, which are in turn calculated using SCMF theory. They claimed their new method converges to low-energy sequences faster than classic Monte Carlo and configurational bias Monte Carlo methods.

Successes of Monte Carlo Methods Imposing sequence specificity by keeping the amino acid composition fixed, which reduced significantly the complexity, Koehl and Levitt [21,22] designed new sequences for the fixed backbones of the $\beta 1$ domain of protein G, λ repressor, and sperm whale myoglobin using the conventional Monte Carlo method. The Baker group also utilized the classic Monte Carlo algorithm in their computational protein design program RosettaDesign. Examples of applications of the program include the redesign of nine globular proteins: the src SH3 domain, λ repressor, U1A, protein L, tenascin, procarboxypeptidase, acylphosphatase, S6, and FKBP12 using fixed templates [7].

Genetic Algorithms Originating in genetics and evolution, genetic algorithms generate a multitude of random amino acid sequences and exchange them for a fixed template. Sequences with low energies form hybrids with other sequences, while those with high energies are eliminated in an iterative process which only terminates when a converged solution is attained [46].

Successes of Genetic Algorithms With fixed backbones, Belda et al. [1] applied genetic algorithms to the design of ligands for prolyl oligopeptidase, p53, and DNA gyrase. In addition, with a cubic lattice and empiricial contact potentials Hohm et al. [18] and Miyazawa and Jernigan [31] also employed evolutionary methods to design short peptides that resemble the antibody epitopes of thrombin and blood coagulation factor VIII with high stability.

References

1. Belda I, Madurga S, Llorà X, Martinell M, Tarragó T, Piqueras MG, Nicolás E, Giralt E (2005) ENPDA: An evolutionary structure-based de novo peptide design algorithm. J Computer-Aided Mol Des 19:585–601
2. Benson D, Wisz M, Hellinga H (1998) The development of new biotechnologies using metalloprotein design. Curr Opin Biotechnol 9:370–376
3. Benson D, Wisz M, Hellinga H (2000) Rational design of nascent metalloenzymes. Proc Natl Acad Sci USA 97:6292–6297
4. Cootes AP, Curmi PMG, Torda AE (2000) Biased monte carlo optimization of protein sequences. J Chem Phys 113:2489–2496
5. Dahiyat B, Mayo S (1996) Protein design automation. Protein Sci 5:895–903
6. Dahiyat B, Mayo S (1997) De novo protein design: Fully automated sequence selection. Science 278:82–87
7. Dantas G, Kuhlman B, Callender D, Wong M, Baker D (2003) A large scale test of computational protein design: Folding

and stability of nine completely redesigned globular proteins. J Mol Biol 332:449–460
8. Desjarlais JR, Clarke ND (1998) Computer search algorithms in protein modification and design. Curr Opin Struct Biol 8:471–475
9. Desmet J, Maeyer MD, Hazes B, Lasters I (1992) The dead-end elimination theorem and its use in side-chain positioning. Nature 356:539–542
10. Dill K (1990) Dominant forces in protein folding. Biochemistry 29:7133–7155
11. Dwyer MA, Looger LL, Hellinga H (2004) Computational design of a biologically active enzyme. Science 304:1967–1971
12. Fu X, Kono H, Saven J (2003) Probabilistic approach to the design of symmetric protein quaternary structures. Protein Eng 16:971–977
13. Fung HK, Rao S, Floudas CA, Prokopyev O, Pardalos PM, Rendl F (2005) Computational comparison studies of quadratic assignment like formulations for the in silico sequence selection problem in de novo protein design. J Comb Optim 10:41–60
14. Goldstein R (1994) Efficient rotamer elimination applied to protein side-chains and related spin glasses. Biophys J 66:1335–1340
15. Gordon B, Hom G, Mayo S, Pierce N (2003) Exact rotamer optimization for protein design. J Comput Chem 24:232–243
16. Handel T, Desjarlais J (1995) De novo design of the hydrophobic cores of proteins. Protein Sci 4:2006–2018
17. Hecht M, Das A, Go A, Bradley L, Wei Y (2004) De novo proteins from designed combinatorial libraries. Protein Sci 13:1711–1723
18. Hohm T, Limbourg P, Hoffmann D (2006) A multiobjective evolutionary method for the design of peptidic mimotopes. J Comput Biol 13:113–125
19. Koehl P, Delarue M (1994) Application of a self-consistent mean field theory to predict protein side-chains conformation and estimate their conformational entropy. J Mol Biol 239:249–275
20. Koehl P, Delarue M (1995) A self consistent mean field approach to simultaneouos gap closure and side-chain positioning in homology modeling. Nat Struct Biol 2:163–170
21. Koehl P, Levitt M (1999) De novo protein design. i. in search of stability and specificity. J Mol Biol 293:1161–1181
22. Koehl P, Levitt M (1999) De novo protein design. ii. plasticity in sequence space. J Mol Biol 293:1183–1193
23. Kono H, Doi J (1994) Energy minimization method using automata network for sequence and side-chain conformation prediction from given backbone geometry. Proteins 19:244–255
24. Kono H, Saven J (2001) Statistical theory of protein combinatorial libraries: Packing interactions, backbone flexibility, and the sequence variability of a main-chain structure. J Mol Biol 306:607–628
25. Lasters I, Maeyer MD, Desmet J (1995) Enhanced dead-end elimination in the search for the global minimum energy conformation of a collection of protein side chains. Protein Eng 8:815–822
26. Lee C (1994) Predicting protein mutant energetics by self-consistent ensemble optimization. J Mol Biol 236:918–939
27. Looger L, Hellinga H (2001) Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: Implications for protein design and structural genomics. J Mol Biol 307:429–445
28. Looger L, Dwyer M, Smith J, Hellinga H (2003) Computational design of receptor and sensor proteins with novel functions. Nature 423:185–190
29. Lovell SC, Word JM, Richardson JS, Richardson DC (2000) The penultimate rotamer library. Proteins 40:389–408
30. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:389–408
31. Miyazawa S, Jernigan RL (1996) Residue-residue potentials with a favorable contact pair term and an unfavorable high packing density term for simulation and threading. J Mol Biol 256:623–644
32. Mooers B, Datta D, Baase W, Zollars E, Mayo S, Matthews B (2003) Repacking the core of t4 lysozyme by automated design. J Mol Biol 332:741–756
33. Offredi F, Dubail F, Kischel P, Sarinski K, Stern AS, de Weerdt CV, Hoch JC, Prosperi C, François JM, Mayo SL, Martial JA (2003) De novo backbone and sequence design of an idealized α/β-barrel protein: Evidence of stable tertiary structure. J Mol Biol 325:163–174
34. Park S, Stowell XF, Wang W, Yang X, Saven J (2004) Computational protein design and discovery. Annu Rep Prog Chem Sect C 100:195–236
35. Park S, Yang X, Saven J (2004) Advances in computational protein design. Curr Opin Struct Biol 14:487–494
36. Pierce N, Winfree E (2002) Protein design is np-hard. Protein Eng 15:779–782
37. Pierce N, Spriet J, Desmet J, Mayo S (2000) Conformational splitting: A more powerful criterion for dead-end elimination. J Comput Chem 21:999–1009
38. Pierce N, Spriet J, Desmet J, Mayo S (2000) Conformational splitting: A more powerful criterion for dead-end elimination. J Comput Chem 21:999–1009
39. Ponder J, Richards F (1987) Tertiary templates for proteins. J Mol Biol 193:775–791
40. Dunbrack L Jr, Cohen FE (1997) Bayesian statistical analysis of protein side-chain rotamer preferences. Protein Sci 6:1661–81
41. Richards F, Hellinga H (1994) Optimal sequence selection in proteins of known structure by simulated evolution. Proc Natl Acad Sci USA 91:5803–5807
42. Rosenberg M, Goldblum A (2006) Computational protein design: A novel path to future protein drugs. Curr Pharm Des 12:3973–3997

43. Saven J (2002) Combinatorial protein design. Curr Opin Struct Biol 12:453–458
44. Saven J, Wolynes PG (1997) Statistical mechanics of the combinatorial synthesis and analysis of folding macromolecules. J Phys Chem B 101:8375–8389
45. Shifman J, Mayo S (2002) Modulating calmodulin binding specificity through computational protein design. J Mol Biol 323:417–423
46. Tuffery P, Etchebest C, Hazout S, Lavery R (1991) A new approach to the rapid determination of protein side chain conformations. J Biomol Struct Dyn 8:1267–1289
47. Wernisch L, Hery S, Wodak S (2000) Automatic protein design with all atom force-fields by exact and heuristic optimization. J Mol Biol 301:713–736
48. Zou J, Saven J (2000) Statistical theory of combinatorial libraries of folding proteins: Energetic discrimination of a target structure. J Mol Biol 296:281–294
49. Zou J, Saven J (2003) Using self-consistent fields to bias monte carlo methods with applications to designing and sampling protein sequences. J Chem Phys 118:3843–3854

Derivative-Free Methods for Non-smooth Optimization

ADIL BAGIROV
Centre for Informatics and Applied Optimization,
University of Ballarat,
Ballarat, Australia

MSC2000: 65K05, 90C56

Article Outline

Introduction
Definitions
The Clarke Subdifferential
Semismooth Functions
Quasidifferentiable Functions
Methods
Approximation of Subgradients
Computation of Subgradients
Computation of Subdifferentials and Discrete Gradients
A Necessary Condition for a Minimum
Computation of Descent Directions
The Discrete Gradient Method
Applications
Conclusions
References

Introduction

Consider the following unconstrained minimization problem:

$$\text{minimize} f(x) \text{ subject to} x \in \mathbb{R}^n , \tag{1}$$

where the objective function f is assumed to be Lipschitz continuous.

Nonsmooth unconstrained optimization problems appear in many applications and in particular in data mining. Over more than four decades different methods have been developed to solve problem (1). We mention among them the bundle method and its different variations (see, for example, [11,12,13,14,17,20]), algorithms based on smoothing techniques [18], and the gradient sampling algorithm [8].

In most of these algorithms at each iteration the computation of at least one subgradient or approximating gradient is required. However, there are many practical problems where the computation of even one subgradient is a difficult task. In such situations derivative-free methods seem to be a better choice since they do not use the explicit computation of subgradients.

Among derivative-free methods, the generalized pattern search methods are well suited for nonsmooth optimization [1,19]. However their convergence are proved under quite restrictive differentiability assumptions. It was shown in [19] that when the objective function f is continuously differentiable in $\mathbb{R}^n$, then the lower limit of the norm of the gradient of the sequence of points generated by the generalized pattern search algorithm goes to zero. The paper [1] provides convergence analysis under less restrictive differentiability assumptions. It was shown that if f is strictly differentiable near the limit of any refining subsequence, then the gradient at that point is zero. However, in many practically important problems this condition is not satisfied, because in such problems the objective functions are not differentiable at local minimizers.

In the paper [15] a derivative-free algorithm for a linearly constrained finite minimax problem was proposed. The original problem was converted into a smooth one using a smoothing technique. This algorithm is globally convergent toward stationary points of the finite minimax problem.

In this paper we describe a derivative-free method based on the notion of a discrete gradient for solving

unconstrained nonsmooth optimization problems. Its convergence is proved for a broad class of nonsmooth functions.

Definitions

We use the following notation: $\mathbb{R}^n$ is an n-dimensional space, where the scalar product will be denoted by $\langle x, y\rangle$:

$$\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i$$

and $\|\cdot\|$ will denote the associated norm. The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}^1$ will be denoted by ∇f and the closed δ-ball at $x \in \mathbb{R}^n$ by $S_\delta(x)$ (by S_δ if $x = 0$): $S_\delta(x) = \{y \in \mathbb{R}^n : \|x - y\| \le \delta\}, \delta > 0$.

The Clarke Subdifferential

Let f be a function defined on $\mathbb{R}^n$. Function f is called locally Lipschitz continuous if for any bounded subset $X \subset \mathbb{R}^n$ there exists an $L > 0$ such that

$$|f(x) - f(y)| \le L\|x - y\| \forall x, y \in X.$$

We recall that a locally Lipschitz function f is differentiable almost everywhere and that we can define for it a Clarke subdifferential [9] by

$$\partial f(x) = \operatorname{co}\left\{v \in \mathbb{R}^n : \exists (x^k \in D(f), \; x^k \to x, k \to +\infty) : v = \lim_{k\to+\infty} \nabla f(x^k)\right\},$$

where $D(f)$ denotes the set where f is differentiable and co denotes the convex hull of a set. It is shown in [9] that the mapping $\partial f(x)$ is upper semicontinuous and bounded on bounded sets.

The generalized directional derivative of f at x in the direction g is defined as

$$f^0(x, g) = \limsup_{y\to x, \alpha\downarrow 0} \alpha^{-1}[f(y + \alpha g) - f(y)].$$

If function f is locally Lipschitz continuous, then the generalized directional derivative exists and

$$f^0(x, g) = \max\{\langle v, g\rangle : v \in \partial f(x)\}.$$

f is called a Clarke regular function on $\mathbb{R}^n$ if it is differentiable with respect to any direction $g \in \mathbb{R}^n$ and $f'(x, g) = f^0(x, g)$ for all $x, g \in \mathbb{R}^n$, where $f'(x, g)$ is a derivative of function f at point x with respect to direction g:

$$f'(x, g) = \lim_{\alpha\downarrow 0} \alpha^{-1}[f(x + \alpha g) - f(x)].$$

It is clear that the directional derivative $f'(x, g)$ of the Clarke regular function f is upper semicontinuous with respect to x for all $g \in \mathbb{R}^n$.

Let f be a locally Lipschitz continuous function defined on $\mathbb{R}^n$. For point x to be a minimum point of function f on $\mathbb{R}^n$, it is necessary that $0 \in \partial f(x)$.

Semismooth Functions

The function $f : \mathbb{R}^n \to \mathbb{R}^1$ is called semismooth at $x \in \mathbb{R}^n$, if it is locally Lipschitz continuous at x and for every $g \in \mathbb{R}^n$, the limit

$$\lim_{g'\to g, \alpha\downarrow 0, v\in\partial f(x+\alpha g')} \langle v, g\rangle$$

exists. It should be noted that the class of semismooth functions is fairly wide and it contains convex, concave, max- and min-type functions [16]. The semismooth function f is directionally differentiable and

$$f'(x, g) = \lim_{g'\to g, \alpha\downarrow 0, v\in\partial f(x+\alpha g')} \langle v, g\rangle.$$

Quasidifferentiable Functions

A function f is called quasidifferentiable at a point x if it is locally Lipschitz continuous and directionally differentiable at this point and there exist convex, compact sets $\underline{\partial} f(x)$ and $\overline{\partial} f(x)$ such that

$$f'(x, g) = \max_{u\in\underline{\partial} f(x)} \langle u, g\rangle + \min_{v\in\overline{\partial} f(x)} \langle v, g\rangle.$$

The set $\underline{\partial} f(x)$ is called a subdifferential, the set $\overline{\partial} f(x)$ is called a superdifferential, and the pair of sets $[\underline{\partial} f(x), \overline{\partial} f(x)]$ is called a quasidifferential of function f at a point x [10].

Methods

Approximation of Subgradients

We consider a locally Lipschitz continuous function f defined on $\mathbb{R}^n$ and assume that this function is quasidifferentiable. We also assume that both sets $\underline{\partial} f(x)$ and

$\overline{\partial} f(x)$ at any $x \in \mathbb{R}^n$ are polytopes, that is, at a point $x \in \mathbb{R}^n$ there exist sets

$$A = \{a^1, \ldots, a^m\}, a^i \in \mathbb{R}^n, i = 1, \ldots, m, m \geq 1$$

and

$$B = \{b^1, \ldots, b^p\}, b^j \in \mathbb{R}^n, j = 1, \ldots, p, p \geq 1$$

such that

$$\underline{\partial} f(x) = \operatorname{co} A, \overline{\partial} f(x) = \operatorname{co} B.$$

This assumption is true, for example, for functions represented as a maximum, minimum, or max-min of a finite number of smooth functions.

We take a direction $g \in \mathbb{R}^n$ such that

$$g = (g_1, \ldots, g_n), |g_i| = 1, i = 1, \ldots, n$$

and consider the sequence of n vectors $e^j = e^j(\alpha), j = 1, \ldots, n$ with $\alpha \in (0, 1]$:

$$\begin{array}{lcl} e^1 & = & (\alpha g_1, 0, \ldots, 0), \\ e^2 & = & (\alpha g_1, \alpha^2 g_2, 0, \ldots, 0), \\ \ldots & = & \ldots\ldots\ldots \\ e^n & = & (\alpha g_1, \alpha^2 g_2, \ldots, \alpha^n g_n). \end{array}$$

We introduce the following sets:

$$\underline{R}_0 = A, \overline{R}_0 = B,$$

$$\underline{R}_j = \left\{ v \in \underline{R}_{j-1} : v_j g_j = \max\{w_j g_j : w \in \underline{R}_{j-1}\} \right\},$$

$$\overline{R}_j = \left\{ v \in \overline{R}_{j-1} : v_j g_j = \min\{w_j g_j : w \in \overline{R}_{j-1}\} \right\}.$$

$$j = 1, \ldots, n.$$

It is clear that

$$\underline{R}_j \neq \emptyset, \forall j \in \{0, \ldots, n\}, \underline{R}_j \subseteq \underline{R}_{j-1}, \forall j \in \{1, \ldots, n\}$$

and

$$\overline{R}_j \neq \emptyset, \forall j \in \{0, \ldots, n\}, \overline{R}_j \subseteq \overline{R}_{j-1}, \forall j \in \{1, \ldots, n\}.$$

Moreover,

$$v_r = w_r \ \forall v, w \in \underline{R}_j, r = 1, \ldots, j \tag{2}$$

and

$$v_r = w_r \ \forall v, w \in \overline{R}_j, r = 1, \ldots, j. \tag{3}$$

Consider the following two sets:

$$\underline{R}(x, e^j(\alpha)) = \left\{ v \in A : \langle v, e^j \rangle = \max_{u \in A} \langle u, e^j \rangle \right\},$$

$$\overline{R}(x, e^j(\alpha)) = \left\{ w \in B : \langle w, e^j \rangle = \min_{u \in B} \langle u, e^j \rangle \right\}.$$

Proposition 1 *Assume that function f is quasidifferentiable and its subdifferential and superdifferential are polytopes at a point x. Then there exists $\alpha_0 > 0$ such that*

$$\underline{R}(x, e^j(\alpha)) \subset \underline{R}_j, \overline{R}(x, e^j(\alpha)) \subset \overline{R}_j, j = 1, \ldots, n$$

for all $\alpha \in (0, \alpha_0)$.

Corollary 1 *Assume that function f is quasidifferentiable and its subdifferential and superdifferential are polytopes at a point x. Then there exists $\alpha_0 > 0$ such that*

$$f'(x, e^j(\alpha)) = f'(x, e^{j-1}(\alpha)) + v_j \alpha^j g_j + w_j \alpha^j g_j, \quad \forall v \in \underline{R}_j, \ w \in \overline{R}_j, \ j = 1, \ldots, n$$

for all $\alpha \in (0, \alpha_0]$

Proposition 2 *Assume that function f is quasidifferentiable and its subdifferential and superdifferential are polytopes at a point x. Then the sets $\underline{R}_n$ and $\overline{R}_n$ are singletons.*

Remark 1 In the next subsection we propose an algorithm to approximate subgradients. This algorithm finds a subgradient that can be represented as a sum of elements of the sets $\underline{R}_n$ and $\overline{R}_n$.

Computation of Subgradients

Let $g \in \mathbb{R}^n, |g_i| = 1, i = 1, \ldots, n$ be a given vector and $\lambda > 0, \alpha > 0$ be given numbers. We define the following points:

$$x^0 = x, \ x^j = x^0 + \lambda e^j(\alpha), \ j = 1, \ldots, n.$$

It is clear that

$$x^j = x^{j-1} + (0, \ldots, 0, \lambda \alpha^j g_j, 0, \ldots, 0), \ j = 1, \ldots, n.$$

Let $v = v(\alpha, \lambda) \in \mathbb{R}^n$ be a vector with the following coordinates:

$$v_j = (\lambda \alpha^j g_j)^{-1} \left[f(x^j) - f(x^{j-1}) \right], j = 1, \ldots, n. \tag{4}$$

For any fixed $g \in \mathbb{R}^n$, $|g_i| = 1$, $i = 1, \ldots, n$ and $\alpha > 0$ we introduce the following set:

$$V(g,\alpha) = \left\{ w \in \mathbb{R}^n : \exists (\lambda_k \to +0, k \to +\infty), \; w = \lim_{k \to +\infty} v(\alpha, \lambda_k) \right\}.$$

Proposition 3 *Assume that f is a quasidifferentiable function and its subdifferential and superdifferential are polytopes at x. Then there exists $\alpha_0 > 0$ such that*

$$V(g,\alpha) \subset \partial f(x)$$

for all $\alpha \in (0, \alpha_0]$.

Remark 2 It follows from Proposition 3 that in order to approximate subgradients of quasidifferentiable functions one can choose a vector $g \in \mathbb{R}^n$ such that $|g_i| = 1$, $i = 1, \ldots, n$, sufficiently small $\alpha > 0, \lambda > 0$, and apply (4) to compute a vector $v(\alpha, \lambda)$. This vector is an approximation to a certain subgradient.

Computation of Subdifferentials and Discrete Gradients

In the previous subsection we demonstrated an algorithm for the computation of subgradients. In this subsection we consider an algorithm for the computation of subdifferentials. This algorithm is based on the notion of a discrete gradient. We start with the definition of the discrete gradient, which was introduced in [2] (for more details, see also [3,4]).

Let f be a locally Lipschitz continuous function defined on $\mathbb{R}^n$. Let

$$S_1 = \{g \in \mathbb{R}^n : \|g\| = 1\}, G = \{e \in \mathbb{R}^n : e = (e_1, \ldots, e_n), |e_j| = 1, j = 1, \ldots, n\},$$

$$P = \{z(\lambda) : z(\lambda) \in \mathbb{R}^1, z(\lambda) > 0, \lambda > 0, \lambda^{-1} z(\lambda) \to 0, \lambda \to 0\}.$$

Here S_1 is the unit sphere, G is the set of vertices of the unit hypercube in $\mathbb{R}^n$, and P is the set of univariate positive infinitesimal functions.

We take any $g \in S_1$ and define $|g_i| = \max\{|g_k|, k = 1, \ldots, n\}$. We also take any $e = (e_1, \ldots, e_n) \in G$, a positive number $\alpha \in (0, 1]$, and define the sequence of n vectors $e^j(\alpha), j = 1, \ldots, n$ as in Sect. "Approximation of Subgradients." Then for given $x \in \mathbb{R}^n$ and $z \in P$ we define a sequence of $n+1$ points as follows:

$$\begin{aligned} x^0 &= x + \lambda g, \\ x^1 &= x^0 + z(\lambda) e^1(\alpha), \\ x^2 &= x^0 + z(\lambda) e^2(\alpha), \\ \ldots &= \ldots \quad \ldots \\ x^n &= x^0 + z(\lambda) e^n(\alpha). \end{aligned}$$

Definition 1 The discrete gradient of function f at point $x \in \mathbb{R}^n$ is the vector $\Gamma^i(x, g, e, z, \lambda, \alpha) = (\Gamma_1^i, \ldots, \Gamma_n^i) \in \mathbb{R}^n, g \in S_1$ with the following coordinates:

$$\Gamma_j^i = [z(\lambda)\alpha^j e_j)]^{-1} \left[f(x^j) - f(x^{j-1}) \right], \quad j = 1, \ldots, n, j \neq i,$$

$$\Gamma_i^i = (\lambda g_i)^{-1} \left[f(x + \lambda g) - f(x) - \lambda \sum_{j=1, j \neq i}^{n} \Gamma_j^i g_j \right].$$

It follows from the definition that

$$f(x + \lambda g) - f(x) = \lambda \langle \Gamma^i(x, g, e, z, \lambda, \alpha), g \rangle \quad (5)$$

for all $g \in S_1, e \in G, z \in P, \lambda > 0, \alpha > 0$.

Remark 3 One can see that the discrete gradient is defined with respect to a given direction $g \in S_1$, and in order to compute the discrete gradient $\Gamma^i(x, g, e, z, \lambda, \alpha)$, first we define a sequence of points $x^0, \ldots, x^n$ and compute the values of function f at these points; that is, we compute $n+2$ values of this function including point x. $n-1$ coordinates of the discrete gradient are defined similarly to those of the vector $v(\alpha, \lambda)$ from the Sect. "Approximation of Subgradients," and the ith coordinate is defined so as to satisfy equality (5), which can be considered as as version of the mean value theorem.

Proposition 4 *Let f be a locally Lipschitz continuous function defined on $\mathbb{R}^n$ and $L > 0$ its Lipschitz constant. Then for any $x \in \mathbb{R}^n, g \in S_1, e \in G, \lambda > 0, z \in P$, $\alpha > 0$*

$$\|\Gamma^i\| \leq C(n)L, C(n) = (n^2 + 2n^{3/2} - 2n^{1/2})^{1/2}.$$

For a given $\alpha > 0$ we define the following set:

$$B(x,\alpha) = \{v \in \mathbb{R}^n : \exists (g \in S_1, e \in G, z_k \in P, z_k \to +0, \lambda_k \to +0, k \to +\infty), v = \lim_{k \to +\infty} \Gamma^i(x, g, e, z_k, \lambda_k, \alpha)\}. \quad (6)$$

Proposition 5 *Assume that f is a semismooth, quasidifferentiable function and its subdifferential and superdifferential are polytopes at a point x. Then there exists $\alpha_0 > 0$ such that*

$$co\, B(x,\alpha) \subset \partial f(x)$$

for all $\alpha \in (0, \alpha_0]$.

Remark 4 Proposition 5 implies that discrete gradients can be applied to approximate subdifferentials of a broad class of semismooth, quasidifferentiable functions.

Remark 5 One can see that the discrete gradient contains three parameters: $\lambda > 0$, $z \in P$, and $\alpha > 0$. $z \in P$ is used to exploit the semismoothness of function f, and it can be chosen sufficiently small. If f is a semismooth quasidifferentiable function and its subdifferential and superdifferential are polytopes at any $x \in \mathbb{R}^n$, then for any $\delta > 0$ there exists $\alpha_0 > 0$ such that $\alpha \in (0, \alpha_0]$ for all $y \in S_\delta(x)$. The most important parameter is $\lambda > 0$. In the sequel we assume that $z \in P$ and $\alpha > 0$ are sufficiently small.

Consider the following set:

$$D_0(x,\lambda) = \mathrm{cl\,co}\,\{v \in \mathbb{R}^n : \exists (g \in S_1, e \in G, z \in P) : v = \Gamma^i(x, g, e, \lambda, z, \alpha)\}.$$

Proposition 4 implies that the set $D_0(x,\lambda)$ is compact and it is also convex for any $x \in \mathbb{R}^n$.

Corollary 2 *Let f be a quasidifferentiable semismooth function. Assume that in the equality*

$$f(x + \lambda g) - f(x) = \lambda f'(x, g) + o(\lambda, g), g \in S_1$$

$\lambda^{-1} o(\lambda, g) \to 0$ as $\lambda \to +0$ uniformly with respect to $g \in S_1$. Then for any $\varepsilon > 0$ there exists $\lambda_0 > 0$ such that

$$D_0(x,\lambda) \subset \partial f(x) + S_\varepsilon$$

for all $\lambda \in (0, \lambda_0)$.

Corollary 2 shows that the set $D_0(x,\lambda)$ is an approximation to the subdifferential $\partial f(x)$ for sufficiently small $\lambda > 0$. However, it is true at a given point. To get convergence results for a minimization algorithm based on discrete gradients, we need some relationship between the set $D_0(x,\lambda)$ and $\partial f(x)$ in some neighborhood of a given point x. We will consider functions satisfying the following assumption.

Assumption 1 *Let $x \in \mathbb{R}^n$ be a given point. For any $\varepsilon > 0$ there exist $\delta > 0$ and $\lambda_0 > 0$ such that*

$$D_0(y,\lambda) \subset \partial f(x + \bar{S}_\varepsilon) + S_\varepsilon \quad (7)$$

for all $y \in S_\delta(x)$ and $\lambda \in (0, \lambda_0)$. Here

$$\partial f(x + \bar{S}_\varepsilon) = \bigcup_{y \in \bar{S}_\varepsilon(x)} \partial f(y), \bar{S}_\varepsilon(x) = \{y \in \mathbb{R}^n : \|x - y\| \le \varepsilon\}.$$

A Necessary Condition for a Minimum

Consider problem (1), where $f : \mathbb{R}^n \to \mathbb{R}^1$ is an arbitrary function.

Proposition 6 *Let $x^* \in \mathbb{R}^n$ be a local minimizer of function f. Then there exists $\lambda_0 > 0$ such that*

$$0 \in D_0(x,\lambda)$$

for all $\lambda \in (0, \lambda_0)$.

Proposition 7 *Let $0 \notin D_0(x,\lambda)$ for a given $\lambda > 0$ and $v^0 \in \mathbb{R}^n$ be a solution to the following problem:*

$$minimize \|v\|^2 subject\ to\ v \in D_0(x,\lambda).$$

Then the direction $g^0 = -\|v^0\|^{-1} v^0$ is a descent direction.

Proposition 7 shows how the set $D_0(x,\lambda)$ can be used to compute descent directions. However, in many cases the computation of the set $D_0(x,\lambda)$ is not possible. In the next section we propose an algorithm for the computation of descent directions using a few discrete gradients from $D_0(x,\lambda)$.

Computation of Descent Directions

In this subsection we describe an algorithm for the computation of descent directions of the objective function f of Problem (1).

Let $z \in P, \lambda > 0, \alpha \in (0, 1]$, the number $c \in (0, 1)$, and a tolerance $\delta > 0$ be given.

Algorithm 1 An algorithm for the computation of the descent direction.

Step 1. Choose any $g^1 \in S_1, e \in G$; compute $i = \text{argmax}\{|g_j|, j = 1, \ldots, n\}$ and a discrete gradient $v^1 = \Gamma^i(x, g^1, e, z, \lambda, \alpha)$. Set $\overline{D}_1(x) = \{v^1\}$ and $k = 1$.

Step 2. Compute the vector $\|w^k\|^2 = \min\{\|w\|^2 : w \in \overline{D}_k(x)\}$. If

$$\|w^k\| \leq \delta, \tag{8}$$

then stop. Otherwise go to Step 3.

Step 3. Compute the search direction by $g^{k+1} = -\|w^k\|^{-1} w^k$.

Step 4. If

$$f(x + \lambda g^{k+1}) - f(x) \leq -c\lambda\|w^k\|, \tag{9}$$

then stop. Otherwise go to Step 5.

Step 5. Compute $i = \text{argmax}\{|g_j^{k+1}| : j = 1, \ldots, n\}$ and a discrete gradient

$$v^{k+1} = \Gamma^i(x, g^{k+1}, e, z, \lambda, \alpha),$$

construct the set $\overline{D}_{k+1}(x) = \text{co}\{\overline{D}_k(x) \bigcup \{v^{k+1}\}\}$, set $k = k + 1$, and go to Step 2.

In what follows we provide some explanations of Algorithm 1. In Step 1 we compute the discrete gradient with respect to an initial direction $g^1 \in \mathbb{R}^n$. The distance between the convex hull $\overline{D}_k(x)$ of all computed discrete gradients and the origin is computed in Step 2. This problem is solved using the algorithm from [21]. If this distance is less than the tolerance $\delta > 0$, then we accept point x as an approximate stationary point (Step 2); otherwise we compute another search direction in Step 3. In Step 4 we check whether this direction is a descent direction. If it is, we stop and the descent direction has been computed; otherwise we compute another discrete gradient with respect to this direction in Step 5 and update the set $\overline{D}_k(x)$. At each iteration k we improve the approximation of the subdifferential of function f.

The next proposition shows that Algorithm 1 is terminating.

Proposition 8 *Let f be a locally Lipschitz function defined on $\mathbb{R}^n$. Then, for $\delta \in (0, \bar{C})$, either condition (8) or condition (9) satisfies after m computations of the discrete gradients, where*

$$m \leq 2(\log_2(\delta/\bar{C})/\log_2 r + 1), r = 1 - [(1-c)(2\bar{C})^{-1}\delta]^2,$$

$\bar{C} = C(n)L$, and $C(n)$ is a constant from Proposition 4.

Remark 6 Proposition 4 and equality (5) are true for any $\lambda > 0$ and for any locally Lipschitz continuous functions. This means that Algorithm 1 can compute descent directions for any $\lambda > 0$ and for any locally Lipschitz continuous functions in a finite number of iterations. Sufficiently small values of λ give an approximation to the subdifferential, and in this case Algorithm 1 computes local descent directions. However, larger values of λ do not give an approximation to the subdifferential and in this case descent directions computed by Algorithm 1 can be considered global descent directions.

The Discrete Gradient Method

In this section we describe the discrete gradient method. Let sequences $\delta_k > 0, z_k \in P, \lambda_k > 0, \delta_k \to +0, z_k \to +0, \lambda_k \to +0, k \to +\infty$, sufficiently small number $\alpha > 0$, and numbers $c_1 \in (0, 1), c_2 \in (0, c_1]$ be given.

Algorithm 2 Discrete gradient method

Step 1. Choose any starting point $x^0 \in \mathbb{R}^n$ and set $k = 0$.

Step 2. Set $s = 0$ and $x_s^k = x^k$.

Step 3. Apply Algorithm 1 for the computation of the descent direction at $x = x_s^k, \delta = \delta_k, z = z_k, \lambda = \lambda_k, c = c_1$. This algorithm terminates after a finite number of iterations $l > 0$. As a result we get the set $\overline{D}_l(x_s^k)$ and an element v_s^k such that

$$\|v_s^k\|^2 = \min\{\|v\|^2 : v \in \overline{D}_l(x_s^k)\}.$$

Furthermore, either $\|v_s^k\| \leq \delta_k$ or for the search direction $g_s^k = -\|v_s^k\|^{-1} v_s^k$

$$f(x_s^k + \lambda_k g_s^k) - f(x_s^k) \leq -c_1\lambda_k\|v_s^k\|. \tag{10}$$

Step 4. If

$$\|v_s^k\| \leq \delta_k, \tag{11}$$

then set $x^{k+1} = x_s^k, k = k + 1$ and go to Step 2. Otherwise go to Step 5.

Step 5. Construct the following iteration $x_{s+1}^k = x_s^k + \sigma_s g_s^k$, where σ_s is defined as follows:

$$\sigma_s = \text{argmax}\left\{\sigma \geq 0 : f(x_s^k + \sigma g_s^k) - f(x_s^k) \leq -c_2\sigma\|v_s^k\|\right\}.$$

Step 6. Set $s = s + 1$ and go to Step 3.

For the point $x^0 \in \mathbb{R}^n$ we consider the set $M(x^0) = \left\{x \in \mathbb{R}^n : f(x) \leq f(x^0)\right\}$.

Proposition 9 *Assume that function f is semismooth quasidifferentiable, its subdifferential and superdifferential are polytopes at any $x \in \mathbb{R}^n$, Assumption 1 is satisfied, and the set $M(x^0)$ is bounded for starting points $x^0 \in \mathbb{R}^n$. Then every accumulation point of $\{x^k\}$ belongs to the set $X^0 = \{x \in \mathbb{R}^n : 0 \in \partial f(x)\}$.*

Remark 7 Since Algorithm 1 can compute descent directions for any values of $\lambda > 0$, we take $\lambda_0 \in (0, 1)$, some $\beta \in (0, 1)$, and update $\lambda_k, k \geq 1$ as follows:

$$\lambda_k = \beta^k \lambda_0, k \geq 1.$$

Thus in the discrete gradient method we use approximations to subgradients only at the final stage of the method, which guarantees convergence. In most iterations we do not use explicit approximations of subgradients. Therefore it is a derivative-free method.

Remark 8 It follows from (10) and $c_2 \leq c_1$ that always $\sigma_s \geq \lambda_k$ and therefore $\lambda_k > 0$ is a lower bound for σ_s. This leads to the following rule for the computation of σ_s. We define a sequence:

$$\theta_m = m\lambda_k, \ m \geq 1,$$

and σ_s is defined as the largest θ_m satisfying the inequality in Step 5.

Applications

There are many problems from applications where the objective and/or constraint functions are not regular. We will consider one of them, the cluster analysis problem, which is an important application area in data mining.

Clustering is also known as the unsupervised classification of patterns; it deals with problems of organizing a collection of patterns into clusters based on similarity. Clustering has many applications in information retrieval, medicine, etc.

In cluster analysis we assume that we have been given a finite set C of points in the n-dimensional space $\mathbb{R}^n$, that is,

$$C = \{c^1, \ldots, c^m\}, \text{ where } c^i \in \mathbb{R}^n, \ i = 1, \ldots, m.$$

We consider here partition clustering, that is, the distribution of the points of set C into a given number q of disjoint subsets C^i, $i = 1, \ldots, q$ with respect to predefined criteria such that:

(1) $C^i \neq \emptyset, i = 1, \ldots, q$;
(2) $C^i \bigcap C^j = \emptyset, i, j = 1, \ldots, q, i \neq j$;
(3) $C = \bigcup_{i=1}^{q} C^i$.

The sets $C^i, i = 1, \ldots, q$ are called clusters. The strict application of these rules is called *hard clustering*, unlike *fuzzy clustering*, where the clusters are allowed to overlap. We assume that no constraints are imposed on the clusters $C^i, i = 1, \ldots, q$, that is, we consider the hard unconstrained clustering problem.

We also assume that each cluster $C^i, i = 1, \ldots, q$ can be identified by its center (or centroid). There are different formulations of clustering as an optimization problem. In [5,6,7] the cluster analysis problem is reduced to the following nonsmooth optimization problem:

$$\begin{aligned} &\text{minimize} f(x^1, \ldots, x^q) \\ &\text{subject to } (x^1, \ldots, x^q) \in \mathbb{R}^{n \times q}, \end{aligned} \tag{12}$$

where

$$f(x^1, \ldots, x^q) = \frac{1}{m}\sum_{i=1}^{m} \min_{s=1,\ldots,q} \|x^s - c^i\|^2. \tag{13}$$

Here $\|\cdot\|$ is the Euclidean norm and $x^s \in \mathbb{R}^n$ stands for the sth cluster center. If $q > 1$, then the objective function (13) in problem (12) is nonconvex and nonsmooth. Moreover, function f is a nonregular function, and the computation of even one subgradient of this function is quite a difficult task. This function can be represented as the difference of two convex functions as follows:

$$f(x) = f_1(x) - f_2(x),$$

where

$$f_1(x) = \frac{1}{m}\sum_{i=1}^{m}\sum_{s=1}^{q} \|x^s - c^i\|^2,$$

$$f_2(x) = \frac{1}{m}\sum_{i=1}^{m} \max_{s=1,\ldots,q} \sum_{k=1,k\neq s}^{q} \|x^k - c^i\|^2.$$

It is clear that function f is quasidifferentiable and its subdifferential and are polytopes at any point.

Thus, the discrete gradient method can be applied to solve clustering problem.

Conclusions

We have discussed a derivative-free discrete gradient method for solving unconstrained nonsmooth optimization problems. This algorithm can be applied to a broad class of optimization problems including problems with nonregular objective functions. It is globally convergent toward stationary points of semismooth, quasidifferentiable functions whose subdifferential and superdifferential are polytopes.

References

1. Audet C, Dennis JE Jr (2003) Analysis of generalized pattern searches. SIAM J Optim 13:889–903
2. Bagirov AM, Gasanov AA (1995) A method of approximating a quasidifferential. J Comput Math Math Phys 35(4):403–409
3. Bagirov AM (1999) Minimization methods for one class of nonsmooth functions and calculation of semi-equilibrium prices. In: Eberhard A et al (eds) Progress in Optimization: Contributions from Australasia. Kluwer, Dordrecht, pp 147–175
4. Bagirov AM (2003) Continuous subdifferential approximations and their applications. J Math Sci 115(5):2567–2609
5. Bagirov AM, Rubinov AM, Soukhoroukova AV, Yearwood J (2003) Supervised and unsupervised data classification via nonsmooth and global optimisation. TOP: Span Oper Res J 11(1):1–93
6. Bagirov AM, Ugon J (2005) An algorithm for minimizing clustering functions. Optim 54(4–5):351–368
7. Bagirov AM, Yearwood J (2006) A new nonsmooth optimisation algorithm for minimum sum-of-squares clustering problems. Eur J Oper Res 170(2):578–596
8. Burke JV, Lewis AS, Overton ML (2005) A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J Optim 15(3):751–779
9. Clarke FH (1983) Optimization and Nonsmooth Analysis. Wiley, New York
10. Demyanov VF, Rubinov AM (1995) Constructive Nonsmooth Analysis. Lang, Frankfurt am Main
11. Hiriart-Urruty JB, Lemarechal C (1993) Convex Analysis and Minimization Algorithms, vols 1 and 2. Springer, Berlin
12. Kiwiel KC (1985) Methods of Descent for Nondifferentiable Optimization. In: Lecture Notes in Mathematics. Springer, Berlin
13. Lemarechal C (1975) An extension of Davidon methods to nondifferentiable problems. In: Balinski ML, Wolfe P (eds) Nondifferentiable Optimization. Mathematical Programming Study, vol 3, North-Holland, Amsterdam, pp 95–109
14. Lemarechal C, Zowe J (1994) A condensed introduction to bundle methods in nonsmooth optimization. In: Spedicato E (ed) Algorithms for Continuous Optimization. Kluwer, Dordrecht, pp 357–482
15. Liuzzi G, Lucidi S, Sciandrone M (2006) A derivative free algorithm for linearly constrained finite minimax problems. SIAM J Optim 16(4):1054–1075
16. Mifflin R (1977) Semismooth and semiconvex functions in constrained optimization. SIAM J Control Optim 15(6):959–972
17. Mifflin R (1977) An algorithm for constrained optimization with semismooth functions. Math Oper Res 2:191–207
18. Polak E, Royset JO (2003) Algorithms for finite and semiinfinite min-max-min problems using adaptive smoothing techniques. J Optim Theory Appl 119(3):421–457
19. Torczon V (1997) On the convergence of pattern search algorithms. SIAM J Optim 7:1–25
20. Wolfe PH (1975) A method of conjugate subgradients of minimizing nondifferentiable convex functions. Math Program Stud 3:145–173
21. Wolfe PH (1976) Finding the nearest point in a polytope. Math Program 11(2):128–149

Derivatives of Markov Processes and Their Simulation

GEORG PFLUG
University Vienna, Vienna, Austria

MSC2000: 90C15, 60J05

Article Outline

Keywords
Introduction
  Process Derivatives
  Distributional Derivatives
Regenerative Processes
See also
References

Keywords

Derivatives; Stochastic optimization

Introduction

The optimal design of stochastic systems like queueing or inventory systems is a specific stochastic optimization problem.

Let $Y_x(t)$ be an ergodic Markov process with discrete time $t = 1, 2, \ldots$ and values in $\mathbf{R}^m$, depending on a control parameter $x \in \mathbf{R}^d$. Let $H(x, \cdot)$ be some cost function. The problem is to find the control x which minimizes the expectedcosts of the system either under the transient or under the stationary regime:

1) Under the *transient regime*, the process is started at time 0 in a specific starting state y_0 and observed until time T. The optimality problem reads

$$\begin{cases} \min & F(x) = \sum_{t=1}^{T} \mathsf{E}[H(x, Y_x(t)] \\ \text{s.t.} & x \in S \subseteq \mathbb{R}^d. \end{cases} \tag{1}$$

2) Under the *stationary regime* it is assumed that $Y_x(\infty)$ is distributed according to the stationary distribution of the process, which is — by ergodicity — the asymptotic distribution of $Y_x(t)$ as t tends to infinity. The optimality problem reads

$$\begin{cases} \min & F(x) = \mathsf{E}[H(x, Y_x(\infty)] \\ \text{s.t.} & x \in S \subseteq \mathbb{R}^d. \end{cases} \tag{2}$$

As an example, consider the problem of optimally determining the decision limits (x_1, x_2) in an inventory policy (if the inventory at hand has fallen below x_1 order the amount needed to bring the inventory up to x_2). Assuming that the sales are of random size, the inventory system can be modeled as a Markov process depending on control parameters (x_1 and x_2).

The solution method for such an optimization problem is a version of the stochastic quasigradient method (cf. also ► Stochastic quasigradient methods). The solution is stepwise improved by moving it into the direction of an estimate of the negative gradientof the objective function.

The basic problem is therefore to find good estimates for the gradient $\nabla_x F(x)$. This problem is a generalization of the problem of finding derivatives of probability measures (cf. ► Derivatives of probability measures). The general notions of *distributional derivatives* (direct differentiability) and *process derivatives* (inverse differentiability) are applicable here.

Process Derivatives

Suppose that the process $Y_x(\cdot)$ has a representation of the form

$$Y_x(t+1) = K_t(x, Y_x(t), \xi_t),$$

where ξ_t is a sequence of random variables, the distribution of which does not depend on x. If the derivatives $\nabla_x K_t(x, y, \xi)$, $\nabla_y K_t(x, y, \xi)$ and $\nabla_y H(y)$ exist, we get by elementary calculus

$$\nabla_x H(Y_x(t)) = \nabla_y H(Y_x(t)) \times \left[\sum_{i=0}^{t-1} \left(\prod_{j=i+1}^{t-1} \nabla_y K_j(x, Y_x(j), \xi_j) \right) \nabla_x K_i(x, z_x(i), \xi_i) \right], \tag{3}$$

where the order of multiplication in the product is here and in the following from left to right by *descending index*. Formula (3) may be computed recursively, as follows.

Define the $[m \times d]$ (random) matrices N_t by

$$N_t = \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t-1} \nabla_y K_j(x, Y_x(j), \xi_j) \right) \times \nabla_x K_i(x, Y_x(i), \xi_i).$$

This sequence follows the forward recursion

$$N_0 = 0,$$
$$N_{t+1} = \nabla_x K_t(x, Y_x(t), \xi_t) + \nabla_y K_t(x, Y_x(t), \xi_t) \cdot N_t.$$

After having found N_t by this recursion, one may calculate

$$\nabla_x H(Y_x(t)) = \nabla_y H(Y_x(t)) \cdot N_t.$$

This pointwise calculation carries over to the expectation under the standard assumptions of dominated convergence, yielding

$$\nabla_x \mathsf{E}[H(Y_x(t))] = \mathsf{E}[\nabla_y H(Y_x(t)) \cdot N_t].$$

Now, the estimate for the problem in transient regime is

$$\widehat{\nabla_x F}(x) = \sum_{t=1}^{T} H(Y_x(t)) \cdot N_t,$$

whereas for the stationary regime one uses

$$\widehat{\nabla_x F}(x) = \frac{1}{T-\tau} \sum_{t=\tau+1}^{T} H(Y_x(t)) \cdot N_t,$$

where T is large and τ stands for the warmup-phase of the process, which is skipped for the estimation. Of course, the latter estimate is biased, it bias decreases with increasing T and τ.

Distributional Derivatives

Suppose that the Markov transition has transition density $p_x(y_1 \mid y_0)$, i. e.

$$\mathsf{P}(Y_x(t+1) \in A | Y_x(t) = y_0) = \int_A p_x(y_1|y_0)\, dy_1$$

and starts in state y_0. The expectation of $H(Y_x(t)$ is

$$\mathsf{E}[H(Y_x(t))] = \int \cdots \int H(y_t) \prod_{i=1}^{t} p_x(y_i|y_{i-1})\, dy_t \cdots dy_1.$$

Introduce the *score function*

$$s_x(y_0, \ldots, y_t) = \sum_{i=1}^{t} \frac{\nabla_x p_x(y_i|y_{i-1})}{p_x(y_i|y_{i-1})}.$$

By the product rule we get the formula

$$\nabla_x \mathsf{E}[H(Y_x(t))] = \int \cdots \int H(y_t) s_x(y_0, \ldots, y_t) \times \prod_{i=1}^{t} p_x(y_i|y_{i-1}) \cdots dy_t \cdots dy_1.$$

An estimate for $\nabla_x \mathsf{E}[H(Y_x(t))]$ is

$$H(Y_x(t)) \cdot W_x(t),$$

where $W_x(t) = s_x(Y_x(0), \ldots, Y_x(t))$ is called the *score function martingale*. As before, the estimate for the problem in transient regime is

$$\widehat{\nabla_x F}(x) = \sum_{t=1}^{T} H(Y_x(t)) \cdot W_x(t),$$

whereas the estimate for the stationary regime is

$$\widehat{\nabla_x F}(x) = \frac{1}{T-\tau} \sum_{t=\tau}^{T} H(Y_x(t)) \cdot W_x(t).$$

It is asymptotically unbiased for $T, \tau \to \infty$ (see [2]).

There is also the a way of attacking directly the derivative of the stationary distribution: Let P_x represent the transition matrix (transition operator) of the Markov process. The stationary distribution π_x satisfies

$$\pi_x = \pi_x \cdot P_x$$

and therefore

$$\nabla_x \pi_x = [\nabla_x \pi_x] \cdot P_x + \pi_x \cdot [\nabla_x P_x],$$

i. e.

$$\nabla_x \pi_x = \pi_x [\nabla_x P_x] S_x, \tag{4}$$

with

$$S_x = \sum_{k=0}^{\infty} (P_x^k - 1 \cdot \pi_x).$$

Here $1 \cdot \pi_x$ is the transition with rows being identical to π_x. The operator S_x solves the *Poisson equation*

$$S_x(I - P_x) = I,$$

where I is the identity operator. There is a method, to use equation(4)) as the basis for estimating $\nabla_x \mathsf{E}[Y_x(\cdot)]$, see [1, Chapt. 3].

Regenerative Processes

Recall that a set A is a *regenerative set* of the ergodic Markov transition P if

i) $u \to P(u, B)$ is independent of $u \in A$, for all B; and
ii) $\pi(A) > 0$, where π is the unique stationary probability measure pertaining to P.

Suppose that A is a regenerative set for all transitions P_x. The sequence of *regenerative stopping times* of $Y_x(t)$is

$$T_1^{(A)} = \min\{t\colon\ Y_x(t) \in A\},$$

$$T_{i+1}^{(A)} = \min\left\{t > T_i^{(A)}\colon\ Y_x(t) \in A\right\}.$$

These stopping times cut the process into independent pieces. For a process Y_x started in A, the following fundamental equation relates the finite time behavior to the stationary, i. e. long run behavior:

$$\mathsf{E}[H(Y_x(\infty))] = \frac{\mathsf{E}\left[\sum_{t=1}^{T^{(A)}} H(Y_x(t))\right]}{\mathsf{E}(T^{(A)})}. \tag{5}$$

The score method for derivative estimation gives

$$\nabla_x \mathsf{E}\left[\sum_{t=1}^{T^{(A)}} H(Y_x(t))\right] = \mathsf{E}\left[\sum_{t=1}^{T^{(A)}} H(Y_x(t)) W_x(t)\right]$$

and

$$\nabla_x \mathsf{E}[T^{(A)}] = \mathsf{E}\left[\sum_{t=1}^{T^{(A)}} W_x(t)\right]$$

and — by the quotient rule —

$$\begin{aligned}&\nabla_x \mathsf{E}[H(Y_x(\infty))]\\ &= \frac{\mathsf{E}(T^{(A)}) \cdot \nabla_x \mathsf{E}\left[\sum_{t=1}^{T^{(A)}} H(Y_x(t))\right]}{[\mathsf{E}(T^{(A)})]^2}\\ &- \frac{\mathsf{E}\left[\sum_{t=1}^{T^{(A)}} H(Y_x(t))\right] \cdot \nabla_x \mathsf{E}(T^{(A)})}{[\mathsf{E}(T^{(A)})]^2}\end{aligned}$$

(see [2]). For the estimation of $\nabla_x \mathsf{E}[H(Y_x(\infty))]$, all expectations of the right-hand side have to be replaced by estimates.

See also

- ▶ Derivatives of Probability and Integral Functions: General Theory and Examples
- ▶ Derivatives of Probability Measures
- ▶ Discrete Stochastic Optimization
- ▶ Optimization in Operation of Electric and Energy Power Systems

References

1. Pflug GCh (1996) Optimization of stochastic models: the interface between simulation and optimization. Kluwer, Dordrecht
2. Rubinstein RY, Shapiro A (1993) Discrete event systems: Sensitivity and stochastic optimization by the score function method. Wiley, New York

Derivatives of Probability and Integral Functions: General Theory and Examples

S. URYASEV
Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 90C15

Article Outline

Keywords
Notations and Definitions
Integral Over the Surface Formula
Integral Over the Volume Formula
General Formula
See also
References

Keywords

Probability function; Derivative of an integral; Gradient of an integral; Derivative of a probability function; Gradient of a probability function

Probability functions are commonly used for the analysis of models with uncertainties or variabilities in parameters. For instance, in risk and reliability analysis, performance functions, characterizing the operation of systems, are formulated as probabilities of successful or unsuccessful accomplishment of their missions (core damage probability of a nuclear power plant, probability of successful landing of an aircraft, probability of profitable transactions in a stock market, or percentiles of the risks in public risk assessments). Sensitivity analysis of such performance functions involves evaluating of their derivatives with respect to the parameters. Also, the derivatives of the probability function can be used to solve *stochastic optimization* problems [1].

A probability function can be formally presented as an expectation of a discontinuous indicator function of a set, or as an integral over a domain — depending upon parameters. Nevertheless, differentiability conditions of the probability function do not follow from similar conditions of the expectations of continuous (smooth or convex) functions.

The derivative of the probability function has many equivalent representations. It can be represented as an integral over the surface, an integral over the volume, or a sum of integrals over the volume and over the surface. Also, it can be calculated using weak derivatives of the probability measures or conditional expectations.

The first general result on the differentiability of the probability function was obtained by E. Raik [8]. He represented the gradient of the probability function with one constraint in the form of the surface integral. S. Uryasev [10] extended Raik's formula for probability functions with many constraints. A.I. Kibzun and G.L. Tretyakov [3] extended it to the piecewise smooth constraint and probability density function. Special cases of probability function with normal and gamma distributions were investigated by A. Prékopa [6]. G.Ch. Pflug [5] represented the gradient of probability function in the form of an expectation using weak probability measures.

Uryasev [9] expressed the gradient of the probability function as a volume integral. Also, using a change of variables, K. Marti [4] derived the probability function gradient in the form of the volume integral.

A general analytical formula for the *derivative of probability functions* with many constraints was obtained by Uryasev [10]; it calculates the gradient as an integral over the surface, an integral over the volume, or the sum of integrals over the surface and the volume. Special cases of this formula correspond to the Raik formula [8], the Uryasev formula[9], and the change-of-variables approach [4].

The gradient of the quantile function was obtained in [2].

Notations and Definitions

Let an *integral over the volume*

$$F(x) = \int_{f(x,y)\le 0} p(x,y)\,dy \tag{1}$$

be defined on the Euclidean space $\mathbf{R}^n$, where $f\colon \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^k$ and $p\colon \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}$ are some functions. The inequality $f(x,y) \le 0$ in the integral is a system of inequalities

$$f_i(x,y) \le 0, \quad i = 1,\dots,k.$$

Both the kernel function $p(x,y)$ and the function $f(x,y)$ defining the integration set depend upon the parameter x. For example, let

$$F(x) = \mathsf{P}\{f(x,\zeta(\omega)) \le 0\} \tag{2}$$

be a *probability function*, where $\zeta(\omega)$ is a random vector in $\mathbf{R}^m$. The random vector $\zeta(\omega)$ is assumed to have a probability density $p(x,y)$ that depends on a parameter $x \in \mathbf{R}^n$. The probability function can be represented as an *expectation of an indicator function*, which equals one on the integration set, and equals zero outside of it. For example, let

$$\begin{aligned} F(x) &= \mathsf{E}\left[I_{\{f(x,\zeta)\le 0\}} g(x,\zeta)\right] \\ &= \int_{f(x,y)\le 0} g(x,y)\rho(x,y)\,dy \\ &= \int_{f(x,y)\le 0} p(x,y)\,dy, \end{aligned} \tag{3}$$

where $I_{\{\cdot\}}$ is an indicator function, andthe random vector ζ in $\mathbf{R}^m$ has a probability density $\rho(x,y)$ that depends on a parameter $x \in \mathbf{R}$.

Integral Over the Surface Formula

The following formula calculates the gradient of an integral (1) over the set given by nonlinear inequalities as sum of integral over the volume plus integral over the surface of the integration set. We call this the *integral over the surface formula* because if the density $p(x,y)$ does not depend upon x the gradient of the integral (1) equals an integral over the surface. This formula for the case of one inequality was obtained by Raik [8] and generalized for the case with many inequalities by Uryasev [10].

Let us denote by $\mu(x)$ the integration set

$$\begin{aligned} \mu(x) &= \{y \in \mathbb{R}^m :\ f(x,y) \le 0\} \\ &:= \{y \in \mathbb{R}^m :\ f_l(x,y) \le 0,\ 1 \le l \le k\} \end{aligned}$$

and by $\partial\mu(x)$ the surface of this set $\mu(x)$. Also, let us denote by $\partial_i\mu(x)$ a part of the surface which corresponds to the function $f_i(x,y)$, i. e.,

$$\partial_i\mu(x) = \mu(x) \cap \{y \in \mathbb{R}^m :\ f_i(x,y) = 0\}.$$

If the constraint functions are differentiable and the following integral exists, then gradient of integral (1)

equals

$$\nabla_x F(x) = \int_{\mu(x)} \nabla_x p(x,y)\, dy - \sum_{i=1}^{k} \int_{\partial_i \mu(x)} \frac{p(x,y)}{\|\nabla_y f_i(x,y)\|} \nabla_x f_i(x,y)\, dS. \quad (4)$$

A potential disadvantage of this formula is that in multidimensional case it is difficult to calculate the integral over the nonlinear surface. Most well known numerical techniques, such as Monte-Carlo algorithms, are applicable to volume integrals. Nevertheless, this formula can be quite useful in various special cases, such as the linear case.

Example 1 (Linear case: Integral over the surface formula [10].)

Let $A(\omega)$, be a random $l \times n$ matrix with the joint density $p(A)$. Suppose that $x \in \mathbf{R}^n$ and $x_j \neq 0, j = 1, \ldots, n$. Let us define

$$F(x) = \mathsf{P}\{A(\omega)x \le b,\ A(\omega) \ge 0\}, \quad b = (b_1, \ldots, b_l) \in \mathbb{R}^l, \quad x \in \mathbb{R}^n, \quad (5)$$

i. e. $F(x)$ is the probability that the linear constraints $A(\omega)x \le b, A(\omega) \ge 0$ are satisfied. The constraint, $A(\omega) \ge 0$, means that all elements $a_{ij}(\omega)$ of the matrix $A(\omega)$ are nonnegative. Let us denote by A_i and A^i the ith row and column of the matrix A

$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_l \end{pmatrix} = \left(A^1, \ldots, A^n\right);$$

then

$$f(x,A) = \begin{pmatrix} f_1(x,A) \\ \vdots \\ f_k(x,A) \end{pmatrix} = \begin{pmatrix} A_1 x - b_1 \\ \vdots \\ A_l x - b_l \\ -A^1 \\ \vdots \\ -A^n \end{pmatrix},$$

$k = l + l \times n$.

The function $F(x)$ equals

$$F(x) = \int_{f(x,A)\le 0} p(A)\, dA. \quad (6)$$

We use formula (4) to calculate the gradient $\nabla_x F(x)$ as an integral over the surface. The function $p(A)$ does not depend upon x and $\nabla_x p(A) = 0$. Formula (4) implies that $\nabla_x F(x)$ equals

$$-\sum_{i=1}^{k} \int_{\partial_i \mu(x)} \frac{p(A)}{\|\nabla_A f_i(x,A)\|} \nabla_x f_i(x,A)\, dS.$$

Since $\nabla_x f_i(x, A) = 0$ for $i = l + 1, \ldots, k$, then $\nabla_x F(x)$ equals

$$-\sum_{i=1}^{l} \int_{\partial_i \mu(x)} \frac{p(A)}{\|\nabla_A f_i(x,A)\|} \nabla_x f_i(x,A)\, dS = -\sum_{i=1}^{l} \int_{\partial_i \mu(x)} \frac{p(A)}{\|x\|} A_i^{\top}\, dS = -\|x\|^{-1} \sum_{i=1}^{l} \int_{\substack{Ax\le b,\ A\ge 0 \\ A_i x = b_i}} p(A) A_i^{\top}\, dS.$$

Integral Over the Volume Formula

This section presents gradient of the function (1) in the form of volume integral. Let us introduce the following shorthand notations

$$f_{1l}(x,y) = \begin{pmatrix} f_1(x,y) \\ \vdots \\ f_l(x,y) \end{pmatrix}, \quad f(x,y) = f_{1k}(x,y),$$

$$\nabla_y f(x,y) = \begin{pmatrix} \frac{\partial f_1(x,y)}{\partial y_1} & \cdots & \frac{\partial f_k(x,y)}{\partial y_1} \\ & \vdots & \\ \frac{\partial f_1(x,y)}{\partial y_m} & \cdots & \frac{\partial f_k(x,y)}{\partial y_m} \end{pmatrix}.$$

Divergence for the $n \times m$ matrix H consisting of the elements h_{ji} is denoted by

$$\operatorname{div}_y H = \begin{pmatrix} \sum_{i=1}^{m} \frac{\partial h_{1i}}{\partial y_i} \\ \vdots \\ \sum_{i=1}^{m} \frac{\partial h_{ni}}{\partial y_i} \end{pmatrix}.$$

Following [10], the derivative of the function (1) is represented as an integral over the volume

$$\nabla_x F(x) = \int_{\mu(x)} \nabla_x p(x,y)\, dy + \int_{\mu(x)} \operatorname{div}_y \left(p(x,y) H(x,y)\right)\, dy, \quad (7)$$

where a matrix function H: $\mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^{n\times m}$ satisfies the equation

$$H(x, y)\nabla_y f(x, y) + \nabla_x f(x, y) = 0. \tag{8}$$

The last system of equations may have many solutions, therefore formula (7) provides a number of equivalent expressions for the gradient. The following section gives analytical solutions of this system of equations. In some cases, this system does not have any solution, and formula (7) is not valid. The following section deals with such cases and provides a general formula where system of equations can be solved only for some of the functions defining the integration set.

Example 2 (Linear case: Integral over the volume formula [10].)

With formula (7), the gradient of the probability function (5) with linear constrains considered in Example 1 can be represented as the integral over the volume. It can be shown that equation (8) does not have a solution in this case. Nevertheless, we can slightly modify the constraints, such that integration set is not changed and equation (8) has a solution. In the vector function $f(x, A)$ we multiply column A^i on x^i if x^i is positive or multiply it on $-x^i$ if x^i is negative. Therefore, we have the following constraint function

$$f(x, A) = \begin{pmatrix} A_1 x - b_1 \\ \vdots \\ A_l x - b_l \\ -(+)x_1 A^1 \\ \vdots \\ -(+)x_n A^n \end{pmatrix}, \tag{9}$$

where $-(+)$ means that we take an appropriate sign. It can be directly checked that, the matrix $H_l^*(x, A)$

$$H^*(x, A) = \left(h^1(x, A_1), \ldots, h^l(x, A_l)\right),$$
$$h^i(x, A_i) = -\begin{pmatrix} a_{i1}x_1^{-1} & & 0 \\ & \ddots & \\ 0 & & a_{in}x_n^{-1} \end{pmatrix}$$

is a solution of system (8). As it will be shown in the next section, this analytical solution follows from the fact that change of the variables $Y^i = x_i A^i$, $i = 1, \ldots, n$, eliminates variables x^i, $i = 1, \ldots, n$, from the constraints (9).

Since $\nabla_x p(A) = 0$ and $\mathrm{div}_A(p(A)H^*(x, A))$ equals

$$-\begin{pmatrix} x_1^{-1}\left(lp(A) + \sum_{i=1}^{l} a_{i1}\frac{\partial}{\partial a_{i1}}p(A)\right) \\ \vdots \\ x_n^{-1}\left(lp(A) + \sum_{i=1}^{l} a_{in}\frac{\partial}{\partial a_{in}}p(A)\right) \end{pmatrix},$$

formula (7) implies that $\partial F(x)/\partial x_j\}$ equals

$$-x_j^{-1}\int_{\substack{Ax\le b \\ A\ge 0}} \left(lp(A) + \sum_{i=1}^{l} a_{ij}\frac{\partial}{\partial a_{ij}}p(A)\right) dA.$$

General Formula

Further, we give a general formula [9,10] for the differentiation of integral (1). A gradient of the integral is represented as a sum of integrals taken over a volume and over a surface. This formula is useful when system of equations (8) does not have a solution. We split the set of constraints $K := = \{1, \ldots, k\}$ into two subsets K_1 and K_2.Without loss of generality we suppose that

$$K_1 = \{1, \ldots, l\}, \quad K_2 = \{l + 1, \ldots, k\}.$$

The derivative of integral (1) can be represented as the sum of the volume and surface integrals

$$\begin{aligned} \nabla_x F(x) &= \int_{\mu(x)} \nabla_x p(x, y)\, dy \\ &+ \int_{\mu(x)} \mathrm{div}_y\left(p(x, y)H_l(x, y)\right)\, dy \\ &- \sum_{i=l+1}^{k} \int_{\partial_i\mu(x)} \frac{p(x, y)}{\left\|\nabla_y f_i(x, y)\right\|} \\ &\times \left[\nabla_x f_i(+x, y) + H_l(x, y)\nabla_y f_i(x, y)\right]\, dS, \end{aligned} \tag{10}$$

where the matrix H_l: $\mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^{n\times m}$ satisfies the equation

$$H_l(x, y)\nabla_y f_{1l}(x, y) + \nabla_x f_{1l}(x, y) = 0. \tag{11}$$

The last equation can have a lot of solutions and we can choose an arbitrary one, differentiable with respect to the variable y.

The general formula contains as a special cases the integral over the surface formula (4) and integral over the volume formula (7). When the set K_1 is empty, the

matrix H_l is absent and the general formula is reduced to the integral over the surface. Also, when the set K_2 is empty we have integral over the volume formula (7). Except these extreme cases, the general formula provides number of intermediate expressions for the gradient in the form of the sum of an integral over the surface and an integral over the volume. Thus, we have a number of equivalent representations of the gradient corresponding to the various sets K_1 and K_2 and solutions of equation (11).

Equation (11) (and equation (8) which is a partial case of equation (11)) can be solved explicitly. Usually, this equation has many solutions. The matrix

$$-\nabla_x f_{1l}(x,y)\times\left(\nabla_y^{\top} f_{1l}(x,y)\nabla_y f_{1l}(x,y)\right)^{-1}\nabla_y^{\top} f_{1l}(x,y) \quad (12)$$

is a solution of equation (11). Also, in many cases there is another way to solve equation (11) using change of variables. Suppose that there is a change of variables

$$y=\gamma(x,z)$$

which eliminates vector x from the function $f(x, y)$ defining integration set, i. e., function $f(x, \gamma(x, z))$ does not depend upon the variable x. Denote by $\gamma^{-1}(x, y)$ the inverse function, defined by the equation

$$\gamma^{-1}(x,\gamma(x,z))=z.$$

Let us show that the following matrix

$$H(x,y)=\nabla_x\gamma(x,z)|_{z=\gamma^{-1}(x,y)} \quad (13)$$

is a solution of (11). Indeed, the gradient of the function $\gamma(x, y(x, z))$ with respect to x equals zero, therefore

$$\begin{aligned}0&=\nabla_x f_{1l}(x,\gamma(x,z))\\&=\nabla_x\gamma(x,z)\nabla_y f_{1l}(x,y)|_{y=\gamma(x,z)}\\&\quad+\nabla_x f_{1l}(x,y)|_{y=\gamma(x,z)},\end{aligned}$$

and function $\nabla_x\gamma(x,z)|_{z=\gamma-1}(x,y)$ is a solution of (11).

Formula (7) with matrix (13) gives the derivative formulas which can be obtained with change of variables in the integration set [4].

Example 3 While investigating the operational strategies for inspected components (see [7]) the following integral was considered

$$F(x)=\int_{\substack{b(y)\le x,\\ y_i\ge\theta,\\ i=1,\dots,m}} p(y)\,dy, \quad (14)$$

where $x\in\mathbf{R}^1$, $y\in\mathbf{R}^m$, p: $\mathbf{R}^m\to\mathbf{R}^1$, $\theta>0$, $b(y)=\sum_{i=1}^m y_i^{\alpha}$. In this case

$$f(x,y)=\begin{pmatrix}b(y)-x\\ \theta-y_1\\ \vdots\\ \theta-y_m\end{pmatrix},$$

and

$$F(x)=\int_{f(x,y)\le 0} p(y)\,dy=\int_{\mu(x)} p(y)\,dy.$$

Let us consider that $l=1$, i. e. $K_1=\{1\}$ and $K_2=\{2,\dots,m+1\}$. The gradient $\nabla_x F(x)$ equals

$$\begin{aligned}&\int_{\mu(x)}\left[\nabla_x p(y)+\operatorname{div}_y\left(p(y)H_1(x,y)\right)\right]dy\\&-\sum_{i=2}^{m+1}\int_{\partial_i\mu(x)}\frac{p(y)}{\left\|\nabla_y f_i(x,y)\right\|}\\&\times\left[\nabla_x f_i(x,y)+H_1(x,y)\nabla_y f_i(x,y)\right]dS,\end{aligned} \quad (15)$$

Where the matrix $H_1(x, y)$ satisfies (11). In view of

$$\nabla_y f_1(x,y)=\alpha\begin{pmatrix}y_1^{\alpha-1}\\ \vdots\\ y_m^{\alpha-1}\end{pmatrix},\quad \nabla_x f_1(x,y)=-1.$$

a solution $H_1^*(x, y)$ of (11) equals

$$\begin{aligned}H_1^*(x,y)=h(y)&:=\left(h_1(y_1),\dots,h_m(y_m)\right)\\&=\frac{1}{\alpha m}\left(y_1^{1-\alpha},\dots,y_m^{1-\alpha}\right).\end{aligned} \quad (16)$$

Let us denote

$$\begin{aligned}&(\theta_i|y)=(y_1,\dots,y_{i-1},\theta,y_{i+1},\dots,y_m),\\&y^{-i}=(y_1,\dots,y_{i-1},y_{i+1},\dots,y_m),\\&b(\theta_i|y)=\theta^{\alpha}+\sum_{\substack{j=1\\ j\ne i}}^{m} y_j^{\alpha}.\end{aligned}$$

We denote by $y^{-i}\ge\theta$ the set of inequalities

$$y_j\ge\theta,\quad j=1,\dots,i-1,\quad i+1,\dots,m.$$

The sets $\partial_i \mu(x)$, $i = 2, \ldots, m+1$, have a simple structure

$$\partial_i \mu(x) = \mu(x) \bigcap \{y \in \mathbb{R}^m : \ y_i = \theta\}$$
$$= \left\{y^{-i} \in \mathbb{R}^{m-1} : \ b(\theta_i|y) \le x, \ y^{-i} \ge 0\right\}.$$

For $i = 2, \ldots, m+1$, we have

$$\left(\nabla_y f_i(y)\right)_j = 0, \quad j = 1, \ldots, m, \quad j \ne i-1, \tag{17}$$

$$\left(\nabla_y f_i(y)\right)_{i-1} = -1, \quad \left\|\nabla_y f_i(y)\right\| = 1. \tag{18}$$

The function $p(y)$ and the functions $f_i(y)$, $i = 2, \ldots, m+1$, do not depend on x, consequently

$$\nabla_x p(y) = 0, \tag{19}$$

$$\nabla_x f_i(y) = 0, \quad i = 2, \ldots, m+1. \tag{20}$$

Equations (15)–(20) imply

$$\nabla_x F(x) = \int_{\mu(x)} \operatorname{div}_y \left(p(y)h(y)\right) \, dy$$
$$- \sum_{i=2}^{m+1} \int_{\partial_i \mu(x)} \frac{p(y)}{\left\|\nabla_y f_i(y)\right\|} h(y) \nabla_y f_i(y) \, dS$$
$$= \int_{\mu(x)} \operatorname{div}_y \left(p(y)h(y)\right) \, dy$$
$$+ \sum_{i=2}^{m+1} h_{i-1}(\theta) \int_{\partial_i \mu(x)} p(y) \, dS$$
$$= \int_{\substack{b(y) \le x, \\ y_i \ge \theta, \\ i=1,\ldots,m}} \operatorname{div}_y \left(p(y)h(y)\right) \, dy$$
$$+ \sum_{i=1}^{m} \frac{\theta^{1-\alpha}}{\alpha m} \int_{\substack{b(\theta_i|y) \le x, \\ y^{-i} \ge \theta}} p(\theta_i|y) \, dy^{-i}.$$

Since

$$\operatorname{div}_y \left(p(y)h(y)\right)$$
$$= h(y) \nabla_y p(y) + p(y) \operatorname{div}_y h(y)$$
$$= \frac{1}{\alpha m} \sum_{i=1}^{m} \frac{\partial p(y)}{\partial y_i} y_i^{1-\alpha} + p(y) \frac{1-\alpha}{\alpha m} \sum_{i=1}^{m} y_i^{-\alpha},$$

we, finally, obtain that the gradient $\nabla_x F(x)$ equals

$$\int_{\substack{b(y) \le x, \\ y_i \ge \theta, \\ i=1,\ldots,m}} \sum_{i=1}^{m} \frac{y_i^{-\alpha}}{\alpha m} \left[y_i \frac{\partial p(y)}{\partial y_i} + (1-\alpha) p(y) \right] dy$$
$$+ \frac{\theta^{1-\alpha}}{\alpha m} \sum_{i=1}^{m} \int_{\substack{b(\theta_i|y) \le x, \\ y^{-i} \ge \theta}} p(\theta_i|y) \, dy^{-i}.$$

The formula for $\nabla_x F(x)$ is valid for an arbitrary sufficiently smooth function $p(y)$.

See also

- ▶ Derivatives of Markov Processes and Their Simulation
- ▶ Derivatives of Probability Measures
- ▶ Discrete Stochastic Optimization
- ▶ Optimization in Operation of Electric and Energy Power Systems

References

1. Ermoliev Y, Wets RJ-B (eds) (1988) Numerical techniques for stochastic optimization. Ser Comput Math. Springer, Berlin
2. Kibzun AI, Malyshev VV, Chernov DE (1988) Two approaches to solutions of probabilistic optimization problems. Soviet J Automaton Inform Sci 20(3):20–25
3. Kibzun AI, Tretyakov GL (1996) Onprobability function differentiability. Theory and Control System 2:53–63. (In Russian)
4. Marti K (1996) Differentiation formulas for probability functions: The transformation method. Math Program B 75(2)
5. Pflug GCh (1996) Optimization of stochastic models: the interface between simulation and optimization. Kluwer, Dordrecht
6. Prékopa A (1970) On probabilistic constrained programming, Proc. Princeton Symp. Math. Program. Princeton University Press, Princeton, 113–138
7. Pulkkinen A, Uryasev S (1991) Optimal operational strategies for an inspected component: Solution techniques. Collaborative Paper Internat Inst Appl Systems Anal, Laxenburg, Austria CP-91-13
8. Raik E (1975) The differentiability in theparameter of the probability function and optimization of the probability function via the stochastic pseudogradient method. Eesti NSV Teaduste Akad Toimetised Füüs Mat. 24(1):3–6 (In Russian)
9. Uryasev S (1989) A differentiation formula for integrals over sets given by inclusion. Numer Funct Anal Optim 10(7–8):827–841
10. Uryasev S (1994) Derivatives of probability functions and integrals over sets given by inequalities. J Comput Appl Math 56:197–223
11. Uryasev S (1995) Derivatives of probability functions and some applications. Ann Oper Res 56:287–311

Derivatives of Probability Measures

GEORG PFLUG
University Vienna, Vienna, Austria

MSC2000: 90C15

Article Outline

Keywords
Direct Differentiability
Inverse Differentiability
Simulation of Derivatives
  Process Derivatives
  Distributional Derivatives
See also
References

Keywords

Derivatives; Stochastic optimization

For stochastic optimization problems of the form

$$\begin{cases} \min & F(x) = \int H(x,v)\, d\mu_x(v) \\ \text{s.t.} & x \in S \subseteq \mathbb{R}^d \end{cases} \tag{1}$$

where $H(x, v)$ is a cost function, μ_x a family of probability measures indexed by x and $F(x)$ the objective value function (OVF), the necessary condition $\nabla_x F(x) = 0$ must be expressed in terms of the derivatives of $H(x, \cdot)$ and μ_x w.r.t. x. In particular, concepts of differentiability of probability measures are needed.

Direct Differentiability

Suppose that the family (μ_x) is *dominated*, i. e. there is a Borel measure ν such that the densities

$$g_x(v) = \frac{d\mu_x}{d\nu}(v)$$

exist for all x. Then the differentiability of the measures may be defined by the differentiability of the densities.

Definition 1 The family of densities $(g_x(v))$ is called *strongly $L_1(\nu)$-differentiable* if there is a vector of integrable functions $\nabla_x g_x = (g'_{x,1}, \ldots, g'_{x,d})^\intercal$ such that

$$\int \left| g_{x+h}(v) - g_x(v) - h^\top \cdot \nabla_x g_x(v) \right| \, d\nu(v) \\ b = o(\|h\|) \quad \text{as } \|h\| \downarrow 0. \tag{2}$$

The family of densities $(g_x(v))$ is called *weakly $L_1(\nu)$-differentiable* if there is a vector of $L_1(\nu)$ functions $\nabla_x g_x = (g'_{x,1}, \ldots, g'_{x,d})^\intercal$ such that for every bounded measurable function H

$$\int [g_{x+h}(v) - g_x(v) - h^\top \cdot \nabla_x g_x(v)] H(v)\, d\nu(v) \\ = o(\|h\|) \quad \text{as } \|h\| \downarrow 0\,. \tag{3}$$

Weak differentiability implies strong differentiability but not vice versa.

There is also a notion of differentiability for families (μ_x), which do not possess densities (see [3]).

If the densities (g_x) are differentiable and $H(x, v)$ is boundedly differentiable in x and bounded and continuous in v, then the gradient of $F(x) = \int H(x, v) g_x(v) d\nu(v)$ is

$$\int \nabla_x H(x,v) g_x(v)\, d\nu(v) \\ + \int H(x,v) \nabla_x g_x(v)\, d\nu(v).$$

Inverse Differentiability

The family (μ_x) is called *process differentiable* if there exists a family of random variables $V_x(\omega)$ — the *process representation* — defined on some probability space $(\Omega, \mathcal{A}, \mathsf{P})$, such that:

a) $V_x(\cdot)$ has distribution μ_x for all x; and
b) $x \longmapsto V_x(\omega)$ is differentiable a.s.

As an example, let μ_x be exponential distributions with densities $g_x(v) = x \exp(-x \cdot u)$. Then $V_x(\omega) = (1/x) U$ for $U \sim$ Uniform $[0, 1]$ is a process representation in the sense of a) and differentiable in the sense of b) with derivative $\nabla_x V_x(\omega) = -(1/x^2) U$.

Process differentiability does not imply and is not implied by weak differentiability. If $G_x(u) = \int_{-\infty}^{u} g_x(v) dv$ is the distribution function, then process differentiability is equivalent to the differentiability of $x \longmapsto G_x^{-1}(u)$, whereas the weak differentiability is connected to the differentiablity of $x \longmapsto G_x(u)$.

If $V_x(\cdot)$ is a process representation of (μ_x), then the objective function

$$F(x) = \int H(x,v)\, d\mu_x(v) = \mathsf{E}[H(x, V_x)]$$

has derivative

$$\nabla_x F(x) = \mathsf{E}[H_x(x, V_x) + H_v(x, V_x) \cdot \nabla_x V_x].$$

where $H_x(x, v) = \nabla_x H(x, v)$ and $H_v(x, v) = \nabla_v H(x, v)$.

Simulation of Derivatives

If the objective function F in (1) is easily calculated, then the stochastic optimization problem reduces to a standard nonlinear deterministic optimization problem. This is however the exception. In the majority of applications, the objective function value has to be approximated either by a numeric integration technique or a Monte-Carlo (MC) estimate. In the same manner, the gradient $\nabla_x F(x)$ may be approximated either by numerical integration or by Monte-Carlo simulation. We discuss here the construction of MC estimates for the gradient $\nabla_x F(x)$. For simplicity, we treat only the univariate case $x \in \mathbf{R}^1$.

We begin with recalling the Monte-Carlo (MC) method for estimating $F(x)$. If $(V_x^{(i)})$ is a sequence of independent identically distributed random variables with distribution function G_x, then the MC estimate

$$\widehat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} H(x, V_x^{(i)})$$

is an unbiased estimate of $F(x)$.

Process Derivatives

If the family (μ_x) has differentiable process representation (V_x), then

$$\widehat{\nabla_x F_n}(x) = \frac{1}{n}\sum_{i=1}^{n}\Big[H_x(x, V_x^{(i)}) + H_v(x, V_x^{(i)}) \cdot \nabla_x V_x^{(i)}\Big] \quad (4)$$

is a MC estimate of $\nabla_x F(x)$. The method of using the process derivative (4) is also called *perturbation analysis* ([1,2]).

Distributional Derivatives

If the densities g_x are differentiable, there are two possibilities to construct estimates. First, one may define the *score function* $s_x(v) = [\nabla_x g_x(v)]/g_x(v)$ and construct the score function estimate [4]

$$\widehat{\nabla_x F_n}(x) = \frac{1}{n}\sum_{i=1}^{n}\Big[H_x(x, V_x^{(i)}) + H(x, V_x^{(i)}) s_x(V_x^{(i)})\Big],$$

which is unbiased.

Alternatively, one may write the function $\nabla_x g_x(v)$ in the form

$$\nabla_x g_x(v) = c_x[\dot{g}_x(v) - \ddot{g}_x(v)], \quad (5)$$

where $\dot{g}_x$ and $\ddot{g}_x$ are probability densities w.r.t. v, and c_x is a nonnegative constant. One possibility is to set $\dot{g}_x$ resp. $\ddot{g}_x$ as the appropriately scaled positive, resp. negative, part of $\nabla_x g_x$, but other representations are possible as well. Let now $\dot{V}_x^{(i)}$, resp. $\ddot{V}_x^{(i)}$, be random variables with distributions $\dot{g}_x dv$, resp. $\ddot{g}_x dv$. The *difference estimate* is

$$\widehat{\nabla_x F_n}(x) = \frac{1}{n} \times \sum_{i=1}^{n}\Big\{H_x(x, V_x^{(i)}) + c_x[H(x, \dot{V}_x^{(i)}) - H(x, \ddot{V}_x^{(i)}]\Big\},$$

which is unbiased (see [3]).

Example 2 Assume again that (μ_x) are exponential distributions with expectation x. The probability μ_x has density

$$g_x(y) = x \cdot \exp(-xy).$$

Let V_x be distributed according to μ_x. For simplicity, assume that the cost function H does not depend explicitly on x. We need estimates for $\nabla_x E(H(V_x))$. The three methods are:

1) Score derivative: The score function is

$$\frac{\nabla_x g_x(v)}{g_x(v)} = \frac{1}{x} - v$$

and the score function estimate is

$$\widehat{\nabla_x F^{(1)}} = H\left(V_x\right)\left(\frac{1}{x} - V_x\right).$$

2) Difference derivative: There are several representations in the sense of (5). One could use the decomposition of $\nabla_x g_x(\cdot)$ into its positive and negative part (*Jordan–Hahn decomposition*) and get the estimate

$$\widehat{\nabla_x F^{(2a)}} = \frac{1}{x}(H(\dot{V}_x) - H(\ddot{V}_x)),$$

where $\dot{V}_x$ has density

$$xe(1 - xv)e^{-xv} \cdot 1_{v \le \frac{1}{x}}$$

and $\ddot{V}_x$ has density

$$xe(xv-1)e^{-xv}\cdot 1_{v>\frac{1}{x}}$$

and both are independent.
Another possibility is to set

$$\dot{V}_x = -\frac{1}{x}\log U_1,$$
$$\ddot{V}_x = -\frac{1}{x}(\log U_1 + \log U_2),$$

where U_1, U_2 are independent Uniform [0, 1] variates. The final difference estimate is

$$\widehat{\nabla_x F^{(2b)}} = \frac{1}{x}(H(\dot{V}_x) - H(\ddot{V}_x)).$$

3) Process derivative: A process representation of (μ_x) is

$$V_x = -\frac{1}{x}\log(1-U), \quad U \sim \text{Uniform}[0,1].$$

A process derivative of $H(V_x)$ is

$$\widehat{\nabla_x F_x^{(3)}} := H_x(V_x)(-\frac{1}{x}V_x).$$

Notice that in methods 1) and 2) the function H need not to be differentiable and may be an indicator function – as is required in some applications. In method 3), the function H must be differentiable.

Whenever a MC estimate $\widehat{\nabla_x F(x)}$ has been defined, it can be used in a stochastic quasigradient method (SQG; cf. also ► Stochastic quasigradient methods) for optimization

$$X_{s+1} = \text{pr}_S[X_s - \rho_s \widehat{\nabla_x F_n}(X_s)]$$

where pr_S is the projection on the set S and (ρ_s) are the stepsizes. The important feature of such algorithms is the fact that they work with stochastic estimates. In particular, the sample size n per step can be set to 1 and still convergence holds under regularity assumptions. To put it differently, the SQG allows to approach quickly a neighborhood of the solution even with much noise corrupted estimates.

See also

- ► Derivatives of Markov Processes and Their Simulation
- ► Derivatives of Probability and Integral Functions: General Theory and Examples
- ► Discrete Stochastic Optimization
- ► Optimization in Operation of Electric and Energy Power Systems

References

1. Glasserman P (1991) Gradient estimation via perturbation analysis. Kluwer, Dordrecht
2. Ho YC, Cao X (1983) Perturbation analysis and optimization of queueing networks. J Optim Th Appl 20:559–589
3. Pflug GC (1996) Optimization of stochastic models. Kluwer, Dordrecht
4. Rubinstein RY, Shapiro A (1993) Discrete event systems: Sensitivity analysis and stochastic optimization by the score function method. Wiley, New York

Design Optimization in Computational Fluid Dynamics

Doyle Knight
Department Mechanical and Aerospace Engineering, Rutgers University, New Brunswick, USA

MSC2000: 90C90

Article Outline

Keywords
Synonyms
Focus
Framework
Levels of Simulation
The Stages of Design
Emergence of Automated Design Optimization Using CFD
Problem Definition
Algorithms for Optimization
  Gradient Optimizers
  Stochastic Optimizers
Examples
  Sequential Quadratic Programming
  Variational Sensitivity
  Response Surface
  Simulated Annealing
  Genetic Algorithms
Conclusion
See also
References

Keywords

Optimization; Computational fluid dynamics

Synonyms

Design Optimization in CFD

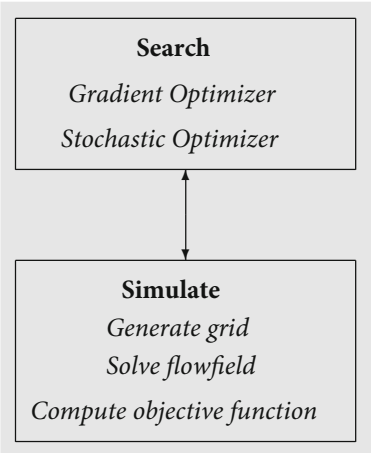
Focus

The article focuses on design optimization using computational fluid dynamics (CFD). *Design* implies the creation of an engineering prototype (e. g., a pump) or engineering process (e. g., particle separator). *Optimization* indicates the selection of a 'best' design. *Computational fluid dynamics* (CFD) represents a family of models of fluid motion implemented on a digital computer. In recent years, efforts have focused on merging elements of these three disciplines to improve design effectiveness and efficiency.

Framework

Consider the design of a prototype or process with n *design variables* $\{x_i: i = 1, \ldots, n\}$ denoted by $\mathbf{x}$. It is assumed that n is finite, although infinite-dimensional design spaces also exist (e. g., the shape of a civilian transport aircraft). The domain of $\mathbf{x}$ constitutes the *design space*. A scalar *objective function* $f(\mathbf{x})$ is assumed to be defined for some (or possibly all) points in the design space. This is the simplest design optimization problem. Oftentimes, however, the optimization cannot be easily cast into this form, and other methods (e. g., Pareto optimality) are employed. The purpose of the design optimization is to find the design point $\mathbf{x}^*$ which minimizes f. Note that there is no loss of generality in assuming the objective is to minimize f, since the maximization of an objective function $\widetilde{f}(\mathbf{x})$ is equivalent to the minimization of $f = -\widetilde{f}$.

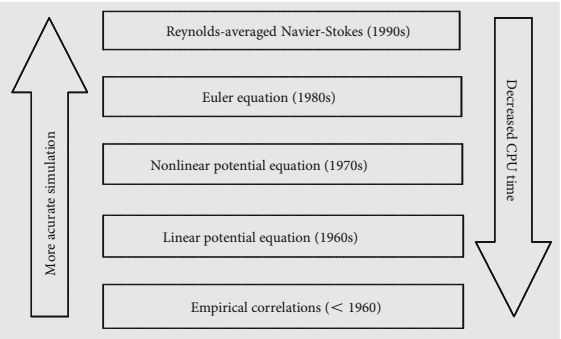
The design optimization is typically an iterative process involving two principal elements. The first element is the *simulation* which evaluates the objective function by (in the case of computational fluid dynamics) a fluid flow code (*flow solver*). The second element is the *search* which determines the direction for traversing the design space. The search engine is the *optimizer* of which they are several different types as described later. The design optimization process is an iterative procedure involving repetitive simulation and search steps until a predefined convergence criteria is met. This is illustrated in Fig. 1.



Design Optimization in Computational Fluid Dynamics, Figure 1
Elements of design optimization

Levels of Simulation

There are five levels of complexity for CFD simulation Fig. 2. *Empirical methods* represent correlations of experimental data and possibly simple one-dimensional analytical models. An example is the NIDA code [15] employed for analysis of two-dimensional and axisymmetric inlets. The code is restricted to a limited family of geometries and flow conditions (e. g., no sideslip). Codes based on the *linear potential* equations (e. g., PANAIR [6]; see also [17]) and *nonlinear potential* equations (e. g., [8]; see also [7]) incorporate increased geometric flexibility while implementing a simplified model of the flow physics (i. e., it is assumed that the shock waves are weak and there is no significant flow separation). Codes employing the *Euler equations* (e. g., [22]) allow for strong shocks and vorticity although neglect viscous effects. *Reynolds-averaged Navier–Stokes codes* (RANS codes) (e. g., GASP [31]) employ a model for the effects of turbulence. The range of execution time between the lowest and highest levels is roughly three orders of magnitude, e. g., on a conventional workstation the NIDA code requires only a few seconds execution time while a 2-dimensional RANS simulation would typically require a few hours.



Design Optimization in Computational Fluid Dynamics, Figure 2
Levels of CFD simulation

The Stages of Design

There are typically three stages of design: *conceptual*, *preliminary* and *detailed*. As the names suggest, the design specification becomes more precise at successive design stages. Thus, for example, a conceptual design of a civilian transport aircraft may consider a (discrete) design space with the possibility of two, three or four engines, while the preliminary design space assumes a fixed number of engines and considers the details of the engine (e. g., nacelle shapes). It is important to note that the CFD algorithms employed in each of these three stages are likely to be different. Typically, the conceptual design stage employs empirical formulae, while the preliminary design stage may also include simplified CFD codes (e. g., linearized and nonlinear potential methods, and Euler codes), and the detailed design stage may utilize full Reynolds-averaged Navier–Stokes methods. Additionally, experiment is oftentimes essential to verify key features of the design.

Emergence of Automated Design Optimization Using CFD

Although the first numerical simulation of viscous fluid flow was published in 1933 by A. Thom [51], CFD as a discipline emerged with the development of digital mainframe computers in the 1960s. With the principal exception of the work on inverse design methods for airfoils (see, for example, the review [30] and [48]), CFD has mainly been employed in *design analysis* as a cost-effective replacement for some types of experiments. However, CFD can now be employed as part of an *automated design optimization process*. This opportunity has arisen for five reasons. First, the continued rapid improvements in computer performance (e. g., doubling of microprocessor performance every 18 to 24 months [3]) enable routine numerical simulations of increasing sophistication and complexity. Second, improve- ments in the accuracy, efficiency and robustness of CFD algorithms (see, for example, [18]) likewise contribute to the capability for simulation of more complex flows. Third, the development of more accurate turbulence models provides increased confidence in the quality of the flow simulations [16]. Fourth, the development of efficient and robust optimizers enable automated search of design spaces [33]. Finally, the development of sophisticated shell languages (e. g., Perl [43]) provide effective control of pathological events which may occur in an automated design cycle using CFD (e. g., square root of a negative number, failure to converge within a predetermined number of iterations, etc.).

Problem Definition

The general scalar *nonlinear optimization problem* (also known as the *nonlinear programming problem*) is [11,33,52]

$$\text{minimize } f(\mathbf{x}), \tag{1}$$

where $f(\mathbf{x})$ is the scalar *objective function* and $\mathbf{x}$ is the vector of design variables. Typically there are limits on the allowable values of $\mathbf{x}$:

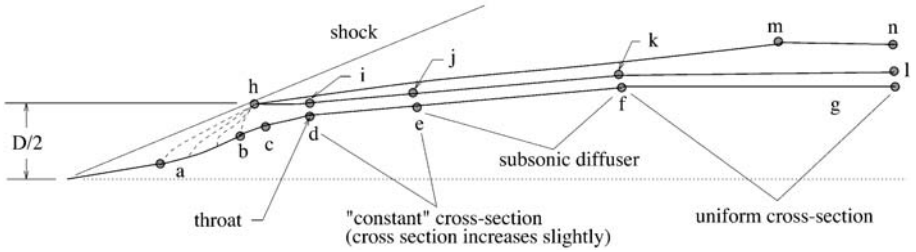
$$\mathbf{a} \le \mathbf{x} \le \mathbf{b}, \tag{2}$$

and m additional *linear* and/or *nonlinear constraints*

$$\begin{cases} c_i(\mathbf{x}) = 0, & i = 1, \ldots, m', \\ c_i(\mathbf{x}) \le 0, & i = m' + 1, \ldots, m. \end{cases} \tag{3}$$

If f and c_i are linear functions, then the optimization problem is denoted the *linear programming problem*, while if f is quadratic and the c_i are linear, then the optimization problem is denoted the *quadratic programming problem*.

An example of a nonlinear optimization problem using CFD is the design of the shape of an inlet for a supersonic missile. The geometry model of an axisymmetric inlet [53] is shown in Fig. 3.



Design Optimization in Computational Fluid Dynamics, Figure 3
Geometry of high speed inlet

The eight design variables are listed below.

Item	Definition
θ_1	initial cone angle
θ_2	final cone angle
x_d	*x*-coordinate of throat
r_d	*r*-coordinate of throat
x_e	*x*-coordinate of end of 'constant' cross section
θ_3	internal cowl lip angle
H_{ej}	height at end of 'constant' cross section
H_{fk}	height at beginning of 'constant' cross section

There are no general methods for guaranteeing that the *global* minimum of an *arbitrary* objective function $f(\mathbf{x})$ can be found in a finite number of steps [4,11]. Typically, methods focus on determining a *local* minimum with additional (often heuristic) techniques to avoid convergence to a local minimum which is not the global minimum.

A point $\mathbf{x}^*$ is a (strong) *local minimum* [11] if there is a region surrounding $\mathbf{x}^*$ wherein the objective function is defined and $f(\mathbf{x}) > f(\mathbf{x}^*)$ for $\mathbf{x} \neq \mathbf{x}^*$. Provided $f(\mathbf{x})$ is twice continuously differentiable (this is not always true; see, for example, [53]), necessary and sufficient conditions for the existence of a solution to (1) subject to (3) may be obtained [11]. In the one-dimensional case with no constraints the sufficient conditions for a minimum at x^* are

$$g = 0 \text{ and } H > 0 \quad \text{at } x = x^*,$$

where $g = df/dx$ and $H = d^2f / dx^2$. For the multidimensional case with no constraints

$$|g_i| = 0 \quad \text{and} \quad H \text{ is positive definite at } x = x^*, \quad (4)$$

where $g_i = \partial f / \partial x_i$, $|g_i|$ is the norm of the vector g_i, and $H = H_{ij}$ is the *Hessian matrix*

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}.$$

The matrix H is positive definite if all of the eigenvalues of H are positive.

Algorithms for Optimization

The efficacy of an optimization algorithm depends strongly on the nature of the design space. In engineering problems, the design space can manifest pathological characteristics. The objective function f may pos sess multiple local optima [36] arising from physical and/or numerical reasons. Examples of the latter include noise introduced in the objective function by grid refinement between successive flow simulations, and incomplete convergence of the flow simulator. Also, the objective function f and/or its gradient g_i may exhibit near discontinuities for physical reasons. For example, a small change in the the design state $\mathbf{x}$ of a mixed compression supersonic inlet operating at critical conditions can cause the terminal shock to be expelled, leading to a rapid decrease in total pressure recovery [44]. Moreover, the objective function f may not be evaluable at certain points. This may be due to constraints in the flow simulator such as a limited range of applicability for empirical data tables.

A brief description of some different classes of general optimizers is presented. These methods are described for the unconstrained optimization problem for reasons of brevity. See [33] for an overview of opti-

mization algorithms and software packages, and [11] for a comprehensive discussion of the constrained optimization problem. Detailed mathematical exposition of optimization problems is presented in [19].

Gradient Optimizers

If the objective function f can be approximated in the vicinity of a point $\widetilde{\mathbf{x}}$ by a quadratic form, then

$$f \approx \widetilde{f} + \widetilde{g}_i(x_i - \widetilde{x}_i) + \frac{1}{2}(x_i - \widetilde{x}_i)\widetilde{H}_{ij}(x_j - \widetilde{x}_j), \quad (5)$$

where $\widetilde{f}$, $\widetilde{g}$ and $\widetilde{H}_{ij}$ imply evaluation at $\widetilde{\mathbf{x}}$ and the Einstein summation convention is implied. In the relatively simple *method of steepest descent* [40], the quadratic term in (5) is ignored, and a line minimization is performed along the direction of $-g_i$, i. e., a sequence of values of the design variable $\mathbf{x} =^{(\nu)}$, $\nu = 1, \ldots$, are formed according to

$$\mathbf{x}^{(\nu)} = \widetilde{\mathbf{x}} + \delta\mathbf{x}^{(\nu)}$$

where

$$\delta x_i^{(\nu)} = -\lambda^{(\nu)}\widetilde{g}_i\,|\widetilde{g}_i|^{-1}$$

and $\lambda^{(\nu)}$, $\nu = 1, \ldots$, are an increasing sequence of displacements. The estimated decrease in the objective function f is $-\lambda^{(\nu)}\,|\widetilde{g}_i|$. The objective function f is evaluated at each iteration ν and the search is terminated when f begins to increase. At this location, the gradient g_i is computed and the procedure is repeated. This method, albeit straightforward to implement, is inefficient for design spaces which are characterized by long, narrow 'valleys' [40].

The *conjugate gradient methods* [40] are more efficient than the method of steepest descent, since they perform a sequence of line minimizations along specific directions in the design space which are mutually orthogonal in the context of the objective function. Consider a line minimization of f along a direction $\mathbf{u} = \{u_i : i = 1, \ldots, n\}$. At any point on the line, the gradient of f in the direction of $\mathbf{u}$ is $u_i\widetilde{g}_i$ by definition. At the minimum point $\widetilde{\mathbf{x}}$ in the line search,

$$u_i\widetilde{g}_i = 0$$

by definition. Consider a second line minimization of f along a direction $\mathbf{v}$. From (5) and noting that H_{ij} is symmetric, the change in g_i along the direction $\mathbf{v}$ is $\widetilde{H}_{ij}v_j$. Thus, the condition that the second line minimization also remain a minimization along the first direction $\mathbf{u}$ is

$$u_i\widetilde{H}_{ij}v_j = 0$$

When this condition is satisfied, $\mathbf{u}$ and $\mathbf{v}$ are denoted *conjugate pairs*. Conjugate gradient methods (CGM) generate a sequence of directions $\mathbf{u}, \mathbf{v}, \ldots$ which are mutually conjugate. If f is exactly quadratic, then CGM yield an n-step sequence to the minimum.

Sequential quadratic programming methods employ the Hessian H which may be computed directly when economical or may be approximated from the sequence of gradients g_i generated during the line search (the *quasi-Hessian* [33]). Given the gradient and Hessian, the location x_i^* of the minimum value of f may be found from (5) as

$$\widetilde{H}_{ij}(x_j^* - \widetilde{x}_j) = -\widetilde{g}_i.$$

For the general case where f is not precisely quadratic, a line minimization is typically performed in the direction $(x_i^* - \widetilde{x}_i)$, and the process is repeated.

Variational sensitivity employs the concept of direct differentiation of the optimization function f and governing fluid dynamic equations (in continuous or discrete form) to obtain the gradient g_i, and optimization using a gradient-based method. It is related to the theory of the control of systems governed by partial differential equations [29,39]. For example, the boundary shape (e. g., airfoil surface) is viewed as the (theoretically infinite-dimensional) design space which controls the objective function f. Several different formulations have been developed depending on the stage at which the numerical discretization is performed, and the use of direct or adjoint (costate) equations. Detailed descriptions are provided in [23] and [24]. Additional references include [2,5,20,21,37,38,50].

The following summary follows the presentation in [24] which employs the adjoint formulation. The objective function f is considered to be a function of the flowfield variables w and the physical shape S. The differential change in the objective function is therefore

$$\delta f = \frac{\partial f}{\partial w}\delta w + \frac{\partial f}{\partial S}\delta S. \quad (6)$$

The discretized governing equations of the fluid motion are represented by the vector of equations

$$R(w; S) = 0$$

and therefore

$$\delta R = \frac{\partial R}{\partial w}\delta w + \frac{\partial R}{\partial S}\delta S = 0, \tag{7}$$

where δR is a vector. Assume a vector Lagrange multiplier ψ and combining (6) and (7)

$$\delta f = \left\{\frac{\partial f}{\partial w} - \psi^{\top}\frac{\partial R}{\partial w}\right\}\delta w + \left\{\frac{\partial f}{\partial S} - \psi^{\top}\frac{\partial R}{\partial S}\right\}\delta S,$$

where $^{\top}$ indicates vector transpose. If ψ is chosen to satisfy the adjoint (costate) equation

$$\psi^{\top}\frac{\partial R}{\partial w} = \frac{\partial f}{\partial w}, \tag{8}$$

then

$$\delta f = G\delta S,$$

where

$$G = \frac{\partial f}{\partial S} - \psi^{\top}\frac{\partial R}{\partial S}.$$

This yields a straightforward method for optimization using, for example, the method of steepest descent. The increment in the shape is

$$\delta S = -\lambda G,$$

where λ is a positive scalar. The variational sensitivity approach is particularly advantageous when the dimension n of the design space (which defines S) is large, since the gradient of S is obtained from a single flowfield solution (7) plus a single adjoint solution (8) which is comparable to the flowfield solution in cost. Constraints can be implemented by projecting the gradient onto an allowable subspace in which the constraints are satisfied.

Response surface methods employ an approximate representation of the objective function using smooth functions which are typically quadratic polynomials [25]. For example, the objective function may be approximated by

$$f \approx \widehat{f} = \alpha + \sum_{1\le i\le n}\beta_i x_i + \sum_{1\le i\le j\le n}\gamma_{ij}x_i x_j$$

where α, β_i and γ_{ij} are coefficients which are determined by fitting $\widehat{f}$ to a discrete set of data using the method of least squares. The minimum of $\widehat{f}$ can then be found by any of the gradient optimizers, with optional recalibration of the coefficients of $\widehat{f}$ as needed. There are many different implementations of the response surface method (see, for example, [12,34] and [46]).

Stochastic Optimizers

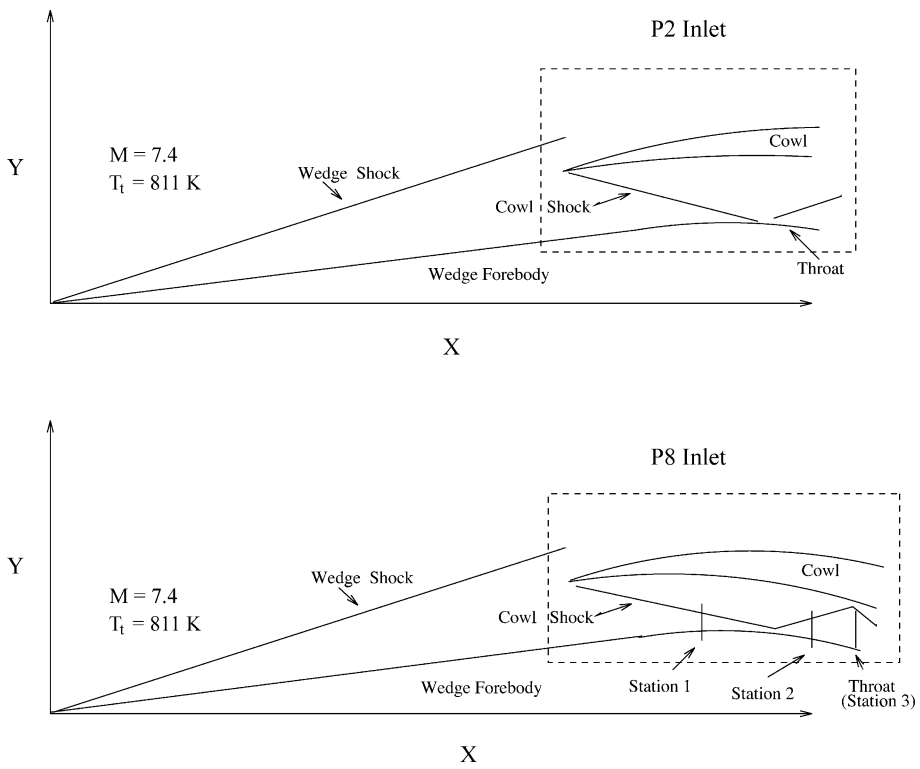
Often the objective function is not well behaved in a portion or all of the design space as discussed above. In such situations, gradient methods can stop without achieving the global optimum (e. g., at an infeasible point, or a local minimum). Stochastic optimizers seek to avoid these problems by incorporating a measure of randomness in the optimization process, albeit oftentimes at a cost of a significant increase in the number of evaluations of the objective function f.

Simulated annealing [26,27,32] mimics the process of crystalization of liquids or annealing of metals by minimizing a function E which is analogous to the energy of a thermodynamic system. Consider a current point (state) in the design space $\widetilde{\mathbf{x}}$ and its associated 'energy' $\widetilde{E}$. A candidate for the next state $\mathbf{x}^*$ is selected by randomly perturbing typically one of the components $\widetilde{x}_j$, $1 \le j \le n$, of, $\widetilde{\mathbf{x}}$ and its energy E^* is evaluated (typically, each component of $\mathbf{x}$ is perturbed in sequence). If $E^* < \widetilde{E}$ then $\widetilde{\mathbf{x}} = \mathbf{x}^*$, i. e., the next state is $\mathbf{x}^*$. If $E^* > \widetilde{E}$ then the probability of selecting $\mathbf{x}^*$ as the next design state is

$$p = \exp\left(-\frac{E^* - \widetilde{E}}{kT}\right),$$

where k is the 'Boltzman constant' (by analogy to statistical mechanics) and T is the 'temperature' which is successively reduced during the optimization according to an assumed schedule [27]. (Of course, only the value of the product kT is important.) The stochastic nature can be implemented by simply calling a random number generator to obtain a value r between zero and one. Then the state $\mathbf{x}^*$ is selected if $r < p$. Therefore, during the sequence of design states, the algorithm permits the selection of a design state with $E > \widetilde{E}$, but the probability of selecting such a state decreases with increasing $E - \widetilde{E}$. This feature tends to enable (but does not guarantee) the optimizer to 'jump out' of a local minimum.

Genetic algorithms (GAs) mimic the process of biological evolution by means of random changes (mutations) in a set of designs denoted the *population* [14]. At each step, the 'least fit' member(s) of the population (i. e., those designs with the highest value of f) are typically removed, and new members are generated by a recombination of some (or all) of the remaining members. There are numerous GA variants. In the approach of [41], an initial population P of designs is generated



Design Optimization in Computational Fluid Dynamics, Figure 4
P2 and P8 inlets

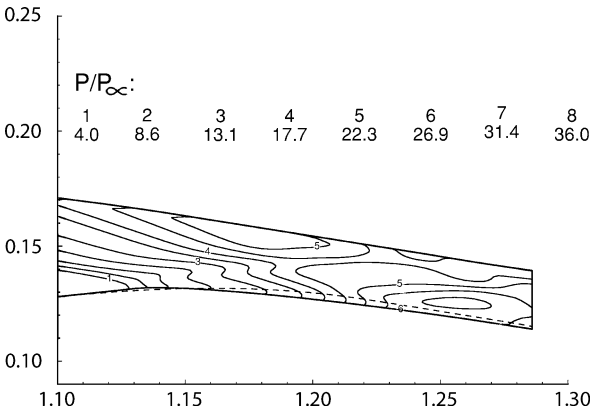
by randomly selecting points $\mathbf{x}_i$, $i = 1, \ldots, p$, satisfying (2). The two best designs (i. e., with the lowest values of f) are joined by a straight line in the design space. A random point $\mathbf{x}'$ is chosen on the line connecting the two best designs. A mutation is performed by randomly selecting a point $\mathbf{x}_{p+1}$ within a specified distance of $\mathbf{x}'$. This new point is added to the population. A member of the population is then removed according to a heuristic criterion, e. g., among the k members with the highest f, remove the member closest to $\mathbf{x}_{p+1}$, thus maintaining a constant number of designs in the population. The removal of the closest member tends to prevent clustering of the population (i. e., maintains diversity). The process is repeated until convergence.

Examples

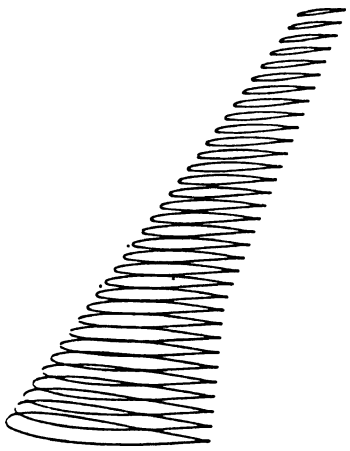
Examples of the above algorithms for optimization using CFD are presented. All of the examples are single discipline involving CFD only. It is emphasized that multidisciplinary optimization (MDO) involving computational fluid dynamics, structural dynamics, electromagnetics, materials and other disciplines is a very active and growing field, and many of the optimization algorithms described herein are appropriate to MDO also. A recent review is presented in [49].

Sequential Quadratic Programming

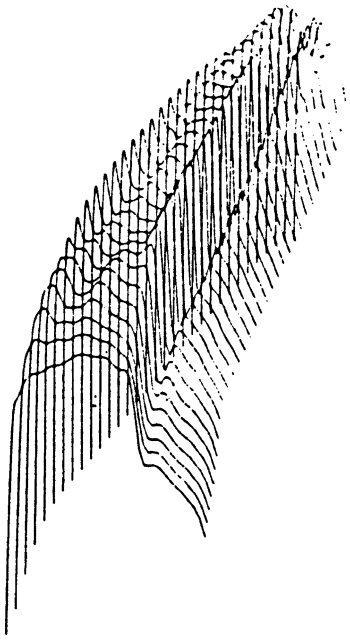
V. Shukla et al. applied a sequential quadratic programming algorithm CFSQP [28] to the optimal design of two hypersonic inlets (denoted P2 and P8) at Mach 7.4. The geometric model is shown in Fig. 4. The optimization criteria was the minimization of the strength of the shock wave which reflected from the centerbody (lower) surface. This is the same criteria as originally posed in the design of the P2 and P8 inlets [13]. The NPARC flow solver [47] was employed for the P2 optimization, and the GASP flow solver [31] for the P8 optimization.



Design Optimization in Computational Fluid Dynamics, Figure 5
Static pressure contours for optimal P8 inlet (the original centerbody contour is shown by the dotted line)



Design Optimization in Computational Fluid Dynamics, Figure 6
Initial shape of wing



Design Optimization in Computational Fluid Dynamics, Figure 7
Initial surface pressure distribution

The optimization criteria was met for both inlets. In Fig. 5, the static pressure contours for the optimized P8 inlet are shown. The strength of the reflected shock is negligible.

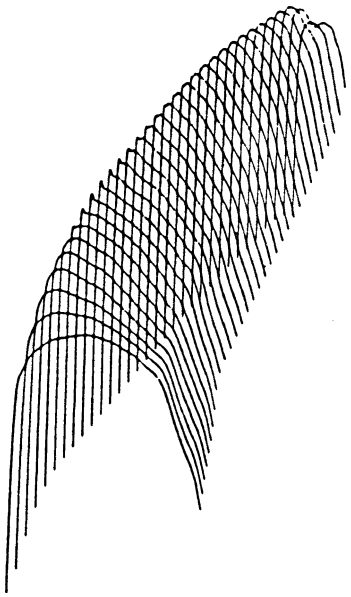
Variational Sensitivity

A. Jameson et al. [24] applied the methodology of variational sensitivity (control theory) to the optimization of a three-dimensional wing section for a subsonic wide-body commercial transport. The design objective was to minimize the drag at a given lift coefficient C_L = 0.55 at Mach 0.83 while maintaining a fixed planform. A two stage procedure was implemented. The first stage employed the Euler equations, while the second stage used the full Reynolds-averaged Navier–Stokes equations. In the second stage, the pressure distribution obtained from the Euler optimization is used as the target pressure distribution.

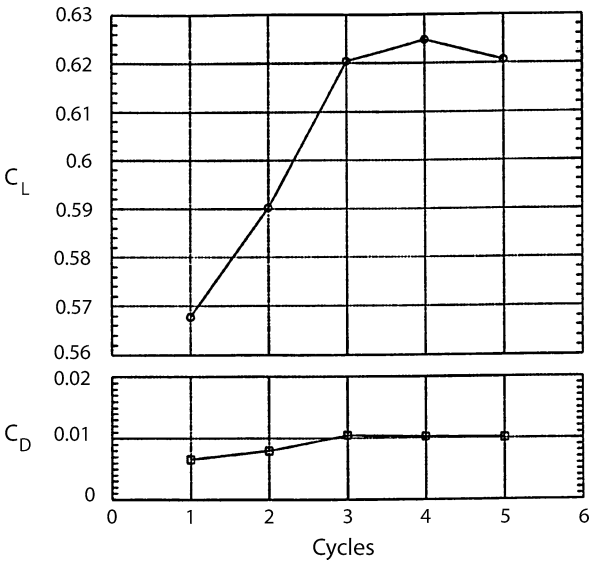
The initial starboard wing shape is shown in Fig. 6 as a sequence of sections in the spanwise direction. The initial pressure distribution on the upper surface, shown as the pressure coefficient c_p plotted with negative values upward, is presented in Fig. 7. A moderately strong shock wave is evident, as indicated by the sharp drop in $-c_p$ at roughly the mid-chord line. After sixty design cycles of the first stage, the drag coefficient was reduced by 15 counts from 0.0196 to 0.0181, and the shock wave eliminated as indicated in the c_p distribution in Fig. 8. A subsequent second stage optimization using the Reynolds–averaged Navier–Stokes equations yielded only slight modifications.

Response Surface

R. Narducci et al. [35] applied a response surface method to the optimal design of a two-dimensional



Design Optimization in Computational Fluid Dynamics, Figure 8
Optimized surface pressure distribution



Design Optimization in Computational Fluid Dynamics, Figure 9
Convergence history for transonic airfoil

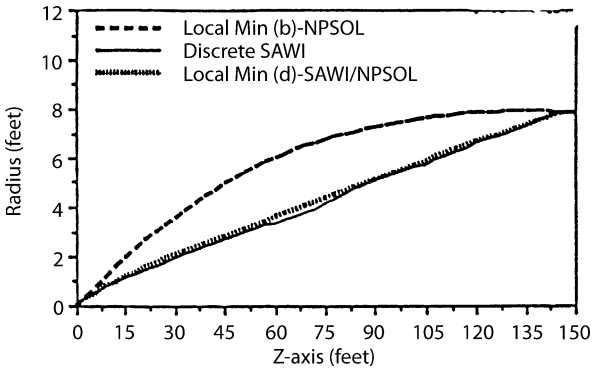
transonic airfoil. The design objective was to maximize the lift coefficient C_L at Mach 0.75 and zero degrees angle of attack, while satisfying the constraints that the drag coefficient $C_D \leq 0.01$ and the thickness ratio $0.075 \leq t \leq 0.15$ where t is the ratio of the maximum airfoil thickness to the airfoil chord. The airfoil surface was represented by a weighted sum of six different shapes which included four known airfoils (a different set of basis functions were employed in [9] for airfoil optimization using a conjugate gradient method). The objective function f was represented by a quadratic polynomial. An inviscid flow solver was employed.

A successful optimization was achieved in five response surface cycles. The history of the convergence of C_L and C_D is shown in Fig. 9. A total of twenty three flow solutions were required for each response surface.

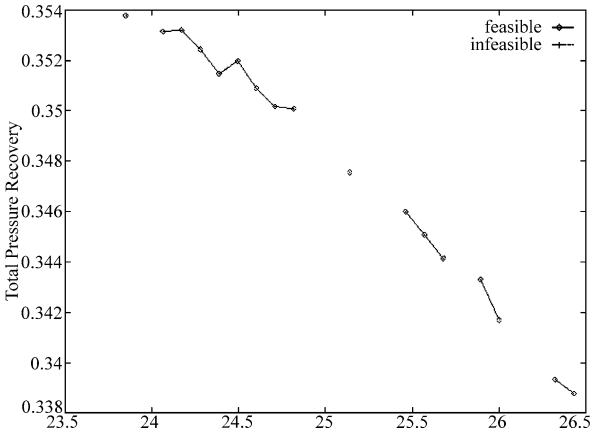
Simulated Annealing

S. Aly et al. [1] applied a modified simulated annealing algorithm to the optimal design of an axisymmetric forebody in supersonic flow. The design objective was to minimize the pressure drag on the forebody of a vehicle at Mach 2.4 and zero angle of attack, subject to constraints on the allowable range of the body radius as a function of axial position. Two different variants of SA were employed, and compared to a gradient optimizer NPSOL [10] which is based on a sequential quadratic programming algorithm. All optimizers employed the same initial design which satisfied the constraints but was otherwise a clearly nonoptimal shape. Optimizations were performed for two different initial shapes. The flow solver was a hybrid finite volume implicit Euler marching method [45].

The first method, denoted simulated annealing with iterative improvement (SAWI), employed SA for the initial phase of the optimization, and then switched to a random search iterative improvement method when close to the optimum. This method achieved from 8% to 31% reduction in the pressure drag, compared to optimal solution obtained NPSOL alone, while requiring fewer number of flowfield simulations (which constitute the principal computational cost). The second method employed SA for the initial phase of the optimization, followed by NPSOL. This approach achieved from 31% to 39% reduction in the pressure drag, compared to the optimal solution obtained by NPSOL alone, while requiring comparable (or less) cputime. The forebody shapes obtained using SA, SA with NPSOL and NPSOL alone are shown in Fig. 10.



Design Optimization in Computational Fluid Dynamics, Figure 10
Forebody shapes obtained using SA, SA with NPSOL and NPSOL. Copyright 1996 AIAA - Reprinted with permission

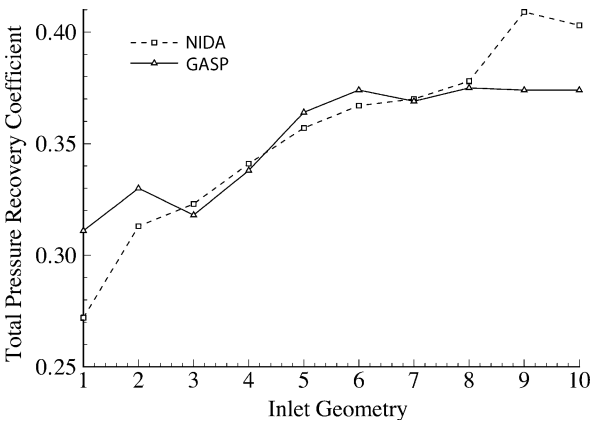


Design Optimization in Computational Fluid Dynamics, Figure 11
Total pressure recovery coefficient versus axial location of throat

Genetic Algorithms

G. Zha et al. applied a modified genetic algorithm (GADO [42]) to the optimal design of an axisymmetric supersonic mixed compression inlet at Mach 4 and 60 kft altitude cruise conditions (see above). The geometric model included eight degrees of freedom (see above), and the optimization criteria was maximization of the inlet total pressure recovery coefficient. The constraints included the requirement for the inlet to start at Mach 2.6, plus additional constraints on the inlet geometry including a minimum cowl thickness and leading edge angle. The constraints were incorporated into the GA using a penalty function. The flow solver was the empirical inlet analysis code NIDA [15]. This code is very efficient, requiring only a few seconds cputime on a workstation, but is limited to 2-dimensional or axisymmetric geometries. Moreover, the design space generated by NIDA (i. e., the total pressure recovery coefficient as a function of the eight degrees of freedom) is nonsmooth with numerous local minima and gaps attributable to the use of empirical data Fig. 11.

The GA achieved a 32% improvement in total pressure recovery coefficient compared to a trial-and-error method [53]. A total of 50 hours on a DEC-2100 workstation was employed. A series of designs generated during the optimization were selected for evaluation by a full Reynolds-averaged Navier–Stokes code (GASP [31]). A close correlation was observed between the predictions of NIDA and GASP Fig. 12.



Design Optimization in Computational Fluid Dynamics, Figure 12
Total pressure recovery coefficient from NIDA and GASP for several different inlet designs

Conclusion

Computational fluid dynamics has emerged as a vital tool in design optimization. The five levels of CFD analysis are utilized in various optimization methodologies. Complex design optimizations have become commonplace. A significant effort is focused on multidisciplinary optimization involving fluid dynamics, solid mechanics, materials and other disciplines.

See also

- ▶ Bilevel Programming: Applications in Engineering
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Multidisciplinary Design Optimization
- ▶ Multilevel Methods for Optimal Design
- ▶ Optimal Design of Composite Structures
- ▶ Optimal Design in Nonlinear Optics
- ▶ Structural Optimization: History

References

1. Aly S, Ogot M, Pelz R (Sept.–Oct.1996) Stochastic approach to optimal aerodynamic shape design. J Aircraft 33(5):945–961
2. Anderson W, Venkatakrishnan V (1997) Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. AIAA Paper 97–0643, (Amer. Inst. Aeronautics and Astronautics, Reston, VA)
3. Berkowitz B (1996) Information age intelligence. Foreign Policy, 103:35–50
4. Boender C, Romeijn H (1995) Stochastic methods. In: Handbook of Global Optimization. Kluwer, Dordrecht, pp 829–869
5. Cabuk H, Modi V (1992) Optimal plane diffusers in laminar flow. J Fluid Mechanics 237:373–393
6. Carmichael R, Erickson L (1981) PAN AIR – A higher order panel method for predicting subsonic or supersonic linear potential flows about arbitrary configurations. AIAA Paper 81–1255, (Amer. Inst. Aeronautics and Astronautics, Reston, VA)
7. Caughey D (1982) The computation of transonic potential flows. In: Annual Rev. Fluid Mechanics, 14, pp 261–283
8. Caughey D, Jameson A (Feb. 1979) Numerical calculation of transonic potential flow about wing–body combinations. AIAA J 17(2):175–181
9. Eyi S, Hager J, Lee K (Dec. 1994) Airfoil design optimization using the Navier–Stokes equations. J Optim Th Appl 83(3):447–461
10. Gill P, Murray W, Saunders M, Wright M (1986) User's guide for NPSOL: A FORTRAN package for nonlinear programming. SOL Techn Report Dept Oper Res Stanford Univ 86(2)
11. Gill P, Murray W, Wright M (1981) Practical optimization. Acad. Press, New York
12. Giunta A, Balabanov V, Haim D, Grossman B, Mason W, Watson L (1996) Wing design for a high speed civil transport using a design of experiments methodology. AIAA Paper 96–4001-CP, (Amer. Inst. Aeronautics and Astronautics, Reston, VA)
13. Gnos A, Watson E, Seebaugh W, Sanator R, DeCarlo J (Apr. 1973) Investigation of flow fields within large–scale hypersonic inlet models. Techn Note NASA D–7150
14. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA
15. Haas M, Elmquist R, Sobel D (Apr. 1992) NAWC inlet design and analysis (NIDA) code. UTRC Report, R92–970037–1, (United Technologies Res. Center)
16. Haase W, Chaput E, Elsholz E, Leschziner M, Müller U (eds) (1997) ECARP – European computational aerodynamics research project: Validation of CFD codes and assessment of turbulence models. Notes on Numerical Fluid Mechanics. Vieweg, Braunschweig/Wiesbaden
17. Hess J (1990) Panel methods in computational fluid dynamics. In: Annual Rev. Fluid Mechanics, 22, pp 255–274
18. Hirsch C (1988) Numerical computation of internal and external flows, vol I–II. Wiley, New York
19. Horst R, Pardalos PM (eds) (1995) Handbook of global optimization. Kluwer, Dordrecht
20. Ibrahim A, Baysal O (1994) Design optimization using variational methods and CFD. AIAA Paper 94–0093, (Amer. Inst. Aeronautics and Astronautics, Reston, VA)
21. Iollo A, Salas M (1995) Contribution to the optimal shape design of two–dimensional internal flows with embedded shocks. ICASE Report 95–20, (NASA Langley Res. Center, Hampton, VA)
22. Jameson A (1982) Steady–state solution of the Euler equations for transonic flow. In: Transonic, Shock and Multidimensional Flows. Acad. Press, New York, pp 37–70
23. Jameson A (1988) Aerodynamic design via control theory. J Sci Comput 3:33–260
24. Jameson A, Pierce N, Martinelli L (1997) Optimum aerodynamic design using the Navier–Stokes equations. AIAA Paper 97–0101, (Amer. Inst. Aeronautics and Astronautics, Reston, VA)
25. Khuri A, Cornell J (1987) Response surfaces: Designs and analyses. M. Dekker, New York
26. Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing. Science 220(4598):671–680
27. van Laarhoven P, Aarts E (1987) Simulated annealing: Theory and Acad. Pressplications. Reidel, London
28. Lawrence AHGC, Zhou J, Tits A (Nov. 1994) User's guide for CFSQP version 2.3: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Techn Report Inst Systems Res Univ Maryland 94–16r1
29. Lions JL (1971) Optimal control of systems governed by partial differential equations. Springer, Berlin (translated from the French)
30. Lores M, Hinson B (1982) Transonic design using computational aerodynamics. In: Progress in Astronautics and Aeronautics, 81. Am Inst Aeronautics and Astronautics, Reston, VA, pp 377–402
31. McGrory W, Slack D, Pressplebaum M, Walters R (1993) GASP version 2.2: The general aerodynamic simulation program. Aerosoft, Blacksburg, VA

32. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equations of state calculations by fast computing machines. J Chem Phys 21:1087–1092
33. Moré J, Wright S (1993) Optimization software guide. SIAM, Philadelphia
34. Myers R, Montgomery D (1995) Response surface methodology: Process and product optimization using design experiments. Wiley, New York
35. Narducci R, Grossman B, Valorani M, Dadone A, Haftka R (1995) Optimization methods for non–smooth or noisy objective functions in fluid dynamic design problems. AIAA Paper 95–1648–CP. Am Inst Aeronautics and Astronautics, Reston, VA
36. Obayashi S, Tsukahara T (1996) Comparison of optimization algorithms for aerodynamic shape design. AIAA Paper 96–2394–CP. Am Inst Aeronautics and Astronautics, Reston, VA
37. Pironneau O (1973) On optimal profiles in Stokes flow. J Fluid Mechanics 59(1):117–128
38. Pironneau O (1974) On optimal design in fluid mechanics. J Fluid Mechanics 64(1):97–110
39. Pironneau O (1984) Optimal shape design for elliptic systems. Springer, Berlin
40. Press W, Flannery B, Teukolsky S, Vetterling W (1986) Numerical recipes. Cambridge Univ. Press, Cambridge
41. Rasheed K, Gelsey A (1996) Adaption of genetic algorithms for continuous design space search. In: Fourth Internat. Conf. Artificial Intelligence in Design: Evolutionary Systems in Design Workshop,
42. Rasheed K, Hirsh H, Gelsey A (1997) A genetic algorithm for continuous design space search. Artif Intell in Eng 11(3):295–305
43. Schwartz R (1993) Learning Perl. O'Reilly, Sebastopol, CA
44. Seddon J, Goldsmith E (eds) (1985) Intake aerodynamics. AIAA Education Ser Amer. Inst. Aeronautics and Astronautics, Reston, VA
45. Siclari M, Del Guidice P (Jan 1990) Hybrid finite volume Approach to Euler solutions for supersonic flows. AIAA J 28(1):66–74
46. Simpson T, Peplinski J, Koch P, Allen J (1997) On the use of statistics in design and the implications for deterministic computer experiments. ASME Paper DETC 97/DTM–3881. Am Soc Mech Engin, New York
47. Sirbaugh J, Smith C, Towne C, Cooper G, Jones R, Power G (Nov. 1994) A users guide to NPARC version 2.0. NASA Lewis Res. Center and Arnold Engin. Developm. Center, Cleveland, OH/Arnold, TN
48. Sobieczky H, Seebass A (1984) Supercritical airfoil and wing design. In: Annual Rev. Fluid Mechanics, 16, pp 337–363
49. Sobieszczanski–Sobieski J, Haftka R (1996) Multidisciplinary aerospace design optimization: Survey of recent developments. AIAA Paper 96–0711. Am Inst Aeronautics and Astronautics, Reston, VA
50. Ta'asan S, Kuruvilla K, Salas M (1992) Aerodyamic design and optimization in one shot. AIAA Paper 92–0025. Am Inst Aeronautics and Astronautics, Reston, VA
51. Thom A (1933) The flow past circular cylinders at low speeds. Proc Royal Soc London A141:651–666
52. Vanderplaats G (1984) Numerical optimization techniques for engineering design: With Applications. McGraw-Hill, New York
53. Zha G, Smith D, Schwabacher M, Rasheed K, Gelsey A, Knight D (Nov.–Dec. 1997) High–performance supersonic missile inlet design using automated optimization. J Aircraft 34(6):697–705

Design of Robust Model-Based Controllers via Parametric Programming

K.I. Kouramas[1], V. Sakizlis[2], Efstratios N. Pistikopoulos[1]
[1] Centre for Process Systems Engineering, Imperial College London, London, UK
[2] Bechtel Co. Ltd., London, UK

Article Outline

Introduction/Background
Definitions
Formulation
  Open-Loop Robust Parametric Model Predictive Controller
  Closed-Loop Robust Parametric Model-Based Control
Methods/Applications
  Parametric Solution of the Inner Maximization Problem of the Open-Loop Robust pMPC Problem
  Solution of the Closed-Loop RpMPC Problem
Cases
  Robust Counterpart (RC) Problem
  Interval Robust Counterpart Problem
Conclusions
References

Introduction/Background

Model predictive control (MPC) is very popular for its capacity to deal with multivariable, constraints-model-based control problems for a variety of complex linear or non-linear processes [13]. MPC is based on the receding-time-horizon philosophy where an open-loop, constrained optimal control problem is solved online at each sampling time to obtain the optimal control actions. The optimal control problem is solved repet-

itively at each time when a new measurement or estimate of the state is available, thus establishing an implicit feedback control method [14,15]. The main reasons for the popularity of MPC are its optimal performance, its capability to handle constraints and its inherent robustness due to feedback control properties.

Despite the widely acknowledged capabilities of MPC, there are two main shortcomings that have been a major concern for the industrial and academic communities. The first shortcoming is that MPC implementation is limited to slowly varying processes due to the demanding online computational effort for solving the online optimal control problem. The second is that, despite its inherent robustness due to the implicit feedback, MPC cannot guarantee the satisfaction of constraints and optimal performance in the presence of uncertainties and input disturbances, since usually it relies on nominal models (uncertainty-free models) for the prediction of future states and control actions [14,20,22].

The first shortcoming of MPC can be overcome by employing the so-called parametric MPC (pMPC) or multiparametric MPC (mp-MPC) [4,16,20]. Parametric MPC controllers are based on the well-known *parametric optimization* techniques [9,18] for solving the open-loop optimal control problem offline and obtain the complete map of the optimal control actions as functions of the states. Thus, a feedback control law is obtained offline and the online computational effort is reduced to simple function evaluations of the feedback control. The inevitable presence of uncertainties and disturbances have been ignored by the pMPC community, and only recently has the research started focusing on control problems with uncertainty [2,20]. In traditional MPC the issue of robustness under uncertainty has been dealt with using various methods such as robust model predictive control [3,8], model predictive tubes [6,12] and min-max MPC [21,22]. However, this is still an unexplored area for pMPC, apart from the recent work presented in [2,20].

In this manuscript we discuss the challenges of robust parametric model predictive control (RpMPC) and we present a method for RpMPC for linear, discrete-time dynamic systems with exogenous disturbances (input uncertainty) and a method for RpMPC for systems with model uncertainty. In both cases the uncertainty is described by the realistic scenario where no uncertainty model (stochastic or deterministic) is known but it is assumed that the uncertainty variables satisfy a set of inequalities.

Definitions

Consider the following linear, discrete-time system:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + W\theta_t \\ y_t &= Bx_t + Du_t + F\theta_t \,, \end{aligned} \tag{1}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$, $u \in \mathcal{U} \subset \mathbb{R}^m$, $y \in \mathcal{Y} \subset \mathbb{R}^q$ and $\theta \in \Theta \subset \mathbb{R}^w$ are the state, input, output and disturbance (or uncertain) input vectors respectively and A, B, C, D, W and F are matrices of appropriate dimensions. The disturbance input θ is assumed to be bounded in the set $\Theta = \{\theta \in \mathbb{R}^w | \theta_i^L \le \theta_i \le \theta_i^U,\ i = 1, \ldots, w\}$. This type of uncertainty is used to characterize a broad variety of input disturbances and modeling uncertainties including non-linearities or hidden dynamics [7,11]. This type of uncertainty in general may result in infeasibilities and performance degradation.

Definition 1 The robust controller is defined as the controller that provides a single control sequence that steers the plant into the feasible operating region for a specific range of variations in the uncertain variables.

The general robust parametric MPC (RpMPC) problem is defined as [20]

$$\phi(x_{t|t}) = \min_{u^N \in V^N} \left\{ x_{t+N|t}^T P x_{t+N|t} + \sum_{k=0}^{N-1} \left[y_{t+k|t}^T Q y_{t+k|t} + u_{t+k}^T R u_{t+k} \right] \right\} \tag{2}$$

$$\text{s.t.} \quad x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k} + W\theta_{t+k} \,, \quad k \ge 0 \tag{3}$$

$$y_{t+k|t} = Cx_{t+k|t} + Du_{t+k} + F\theta_{t+k} \,, \quad k \ge 0 \tag{4}$$

$$g(x_{t+k|t}, u_{t+k}) = C_1 x_{t+k|t} + C_2 u_{t+k} + C_3 \le 0, \quad k = 0, 1, \ldots, N-1 \tag{5}$$

$$h(x_{t+N|t}) = D_1 x_{t+N|t} + D_2 \le 0 \tag{6}$$

$$u_{t+k} = K x_{t+k|t} \,, \quad k \ge N \tag{7}$$

$$x_{t|t} = x^* \tag{8}$$

where $g : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_g}$ and $h : \mathcal{X} \to \mathbb{R}^{n_h}$ are the path and terminal constraints respectively, x^* is the initial state, $u^N = \{u_t, \ldots, u_{t+N-1}\} \in \mathcal{U} \times \cdots \times \mathcal{U} = \mathcal{U}^N$ are the predicted future inputs and $\theta^N = \{\theta_t, \ldots, \theta_{N-1}\} \in \Theta^N$ are the current and future values of the disturbance.

Formulation

The design of a robust control scheme is obtained by solving a receding horizon constrained optimal control problem where the objective is the deviations expected over the entire uncertainty set, or the nominal value of the output and input deviations. In order to ensure feasibility of (2)–(8) for every possible uncertainty scenario $\theta_{t+k} \in \Theta$, $k = 0, \ldots, N-1$, the set of constraints of (2)–(8) is usually augmented with an extra set of feasibility constraints. The type of these constraints, as will be described later, will determine if the RpMPC is an open-loop or closed-loop controller.

Open-Loop Robust Parametric Model Predictive Controller

To define the set of extra feasibility constraints the future state prediction

$$x_{t+k|t} = A^k x^* + \sum_{j=0}^{k-1} (A^j B u_{t+k-1-j} + A^j W \theta_{t+k-1-j}) \tag{9}$$

is substituted into the inequality constraints (5)–(6), which then become

$$\begin{aligned} &\tilde{g}_j(x^*, u^N, \theta^N) \le 0\,, \quad j = 1, \ldots, J \quad \Leftrightarrow \\ &\sum_{i=1}^{n} \gamma 1_{i,j} x_i^* + \sum_{k=0}^{N-1} \sum_{i=1}^{q} \gamma 2_{i,k,j} u_{t+k,i} \\ &\quad + \sum_{k=0}^{N-1} \sum_{i=1}^{w} \gamma 3_{i,k,j} \theta + t + k, i + \gamma 4_j \le 0 \end{aligned} \tag{10}$$

where $\gamma 1$, $\gamma 2$, $\gamma 3$ are coefficients that are explicit functions of the elements of matrices A, B, C, D, W, F, C_1, C_2, C_3, D_1, D_2, Q, R, P. The set of feasibility constraints is defined as

$$\begin{aligned} &\psi(x^*, u^N) \le 0 \Leftrightarrow \forall \theta^N \in \Theta^N \left(\forall j = 1, \ldots, J \right. \\ &\left. \left[\tilde{g}_j(x^*, u^N, \theta^N) \le 0, u^N \in \mathcal{U}^N, x^* \in \mathcal{X}\,.\right]\right) \end{aligned} \tag{11}$$

The constraints $\psi \le 0$ ensure that, given a particular state realization x^*, the single control action u^N satisfies all the constraints for all possible bounded disturbance scenarios over the time horizon. However, this feasibility constraint represents an infinite set of constraints since the inequalities are defined for every possible value of $\theta^N \in \Theta^N$. In order to overcome this problem one has to notice that (11) is equivalent to

$$\begin{aligned} &\max_{\theta^N} \max_j \left\{\tilde{g}(x^*, u^N, \theta^N)\right| \\ &\left. j = 1, \ldots, J, u^N \in \mathcal{U}^N, x^* \in \mathcal{X}, \theta^N \in \Theta^N\right\} \le 0 \end{aligned} \tag{12}$$

Adding (12) into (2)–(8) and minimizing the expectation of the objective function (2) over all uncertain realizations θ_{t+k} one obtains the following robust model predictive control problem:

$$\begin{aligned} \phi(x_{t|t}) = \min_{u^N \in V^N} E_{\theta^N \in \Theta^N} \Bigg\{ & x_{t+N|t}^T P x_{t+N|t} + \\ & \sum_{k=0}^{N-1} \left[y_{t+k|t}^T Q y_{t+k|t} + u_{t+k}^T R u_{t+k} \right] \Bigg\} \end{aligned} \tag{13}$$

$$\text{s.t.} \quad (3)\text{–}(8) \quad \text{and} \quad (11) \tag{14}$$

Problem (13)–(14) is a bilevel program that has as constraint a maximization problem, which, as will be shown later, can be solved parametrically and then replaced by a set of linear inequalities of u^N, x^*. The solution to this problem corresponds to a robust control law as it is defined in Definition 1. Problem (13)–(14) is an open-loop robust control formulation in that it obtains the optimal control actions u^N for the worst-case realization of the uncertainty only, as expressed by inequality (12), and does not take into account the information of the past uncertainty values in the future measurements, thus losing the benefit of the prediction property. This implies that the future control actions can be readjusted to compensate for any variation in the past uncertainty realizations, thereby obtaining more "realistic" and less conservative values for the optimal control actions. This problem can be overcome if we consider the following closed-loop formulation of the problem (2)–(8).

Closed-Loop Robust Parametric Model-Based Control

To acquire a closed-loop formulation of the general RpMPC problem, a dynamic programming approach is used to formulate the worst-case closed-loop MPC problem, which requires the solution of a number of embedded optimization problems that in the case of a quadratic objective are non-linear and non-differentiable. Feasibility analysis is used to directly address the problem and a set of constraints is again incorporated in the optimization problem to preserve feasibility and performance for all uncertainty realizations. Future measurements of the state contain information about the past uncertainty values. This implies that the future control actions can be readjusted to compensate for the past disturbance realizations by deriving a closed-loop MPC problem as shown next. The main idea is to introduce constraints into the control optimization problem (2)–(8) that preserve feasibility and performance for all disturbance realizations. These constraints are given as

$$
\begin{aligned}
&\psi^{\theta_{t+\ell}}(x^*, [u_{t+k}]_{k=0,\ldots,\ell}) \Leftrightarrow \\
&\forall \theta_{t+\ell} \in \Theta \{\exists u_{t+\ell+1} \in \mathcal{U} \{\forall \theta_{t+\ell+1} \\
&\in \Theta \{\exists u_{t+\ell+2} \in \mathcal{U} \ldots \{\forall \theta_{t+N-2} \\
&\in \Theta \{\exists u_{t+N-1} \in \mathcal{U} \{\forall \theta_{t+N-1} \in \Theta \\
&\{\forall j = 1, \ldots, J \left[\tilde{g}_j(x^*, [u_{t+k}]_{k=0,\ldots,N-1}, \right. \\
&\left. [\theta_{t+k}]_{k=0,\ldots,N-1}) \leq 0\right]\}\}\} \ldots \}\}\}, \\
&u_{t+k} \in \mathcal{U}, \quad k = 0, \ldots, \ell, \quad x^* \in \mathcal{X}, \quad \theta_{t+k} \in \Theta, \\
&\qquad k = 0, \ldots, \ell - 1, \quad \ell = 0, \ldots, N - 1. \qquad (15)
\end{aligned}
$$

The constraints of (15) are incorporated into (2)–(8) and give rise to a semi-infinite dimensional program that can be posed as a min–max bilevel optimization problem:

$$
\begin{aligned}
\phi(x^*) = \min_{u^N \in V^N} &\left\{ x_{t+N|t}^T P x_{t+N|t} + \right. \\
&\left. \sum_{k=0}^{N-1} \left[y_{t+k|t}^T Q y_{t+k|t} + u_{t+k}^T R u_{t+k} \right] \right\} \qquad (16)
\end{aligned}
$$

$$
\begin{aligned}
\text{s.t.} \quad &\max_{\theta_{t+N-1,j}} \tilde{g}_j(x^*, u^N, \theta^N) \leq 0 \qquad (17) \\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
&\max_{\theta_{t+1}} \min_{u_{t+2}} \ldots \max_{\theta_{t+N-2}} \min_{u_{t+N-1}} \max_{\theta_{t+N-1}} \max_{j} \\
&\qquad \tilde{g}_j(x^*, u^N, \theta^N) \leq 0 \qquad (18)
\end{aligned}
$$

$$
\begin{aligned}
&\max_{\theta_t} \min_{u_{t+1}} \max_{\theta_{t+1}} \min_{u_{t+2}} \ldots \max_{\theta_{t+N-2}} \min_{u_{t+N-1}} \max_{\theta_{t+N-1}} \max_{j} \\
&\qquad \tilde{g}_j(x^*, u^N, \theta^N) \leq 0 \qquad (19)
\end{aligned}
$$

$$
u^N \in \mathcal{U}^N, \quad x^* \in \mathcal{X}, \quad \theta^N \in \Theta^N. \qquad (20)
$$

The difference between the above formulation and formulation (13)–(14) is that at every time instant $t + k$ the future control actions $\{u_{t+k+1}, \ldots, u_{t+N-1}\}$ are readily adjusted to offset the effect of the past uncertainty $\{\theta_t, \ldots, \theta_{t+k}\}$ to satisfy the constraints. In contrast, in formulation (13)–(14) the control sequence has to ensure constraint satisfaction for all possible disturbance scenarios. The main issue for solving the above optimization problem is how to solve parametrically each of (17)–(19) and replace them with a set of inequalities of u^N, x^* suitable to formulate a multiparametric programming problem. This is shown in the following section.

Methods/Applications

Parametric Solution of the Inner Maximization Problem of the Open-Loop Robust pMPC Problem

An algorithm for solving parametrically the maximization problem of (12), which forms the inner maximization problem of the open-loop RpMPC (13)–(14), comprises the following steps:

Step 1. Solve $G_j(x^*, u^N) = \max_{\theta^N} \{\tilde{g}_j(x^*, u^N, \theta^N) | \theta^{N,L} \leq \theta^N \leq \theta^{N,U}\}$, $j = 1, \ldots, J$ as a parametric program with respect to θ^N and by recasting the control elements and future states as parameters. The parametric solution can be obtained by following the method in [19], where the critical disturbance points for each maximization are identified as follows:

1. If $\frac{\partial \tilde{g}_j}{\partial \theta_{t+k,i}} = \gamma 3_{i,k} > 0 \Rightarrow \theta_{t+k,i}^{cr} = \theta_{t+k,i}^{U}$, $j = 1, \ldots, J$, then $k = 0, \ldots, N-1$, $i = 1, \ldots, w$;
2. If $\frac{\partial \tilde{g}_j}{\partial \theta_{t+k,i}} = \gamma 3_{i,k} < 0 \Rightarrow \theta_{t+k,i}^{cr} = \theta_{t+k,i}^{L}$, $j = 1, \ldots, J$, then $k = 0, \ldots, N-1$, $i = 1, \ldots, w$.

Substituting $\theta_{t+k,i}^{cr}$ in the constraints $\tilde{g} \leq 0$ we obtain $G_j(x^*, u^N) = \tilde{g}_j(x^*, u^N, \theta^{N,cr})$, where $\theta^{N,cr}$ is the sequence of the critical values of the uncertainty vector θ_t^{cr} over the horizon N.

Step 2. Compare the parametric profiles $G_j(x^*, u^N)$ over the joint space of u^N and x^* and retain the upper bounds. A multiparametric linear program is formulated:

$$\psi(x^*, u^N) = \max_j G_j$$
$$\Leftrightarrow \psi(x^*, u^N) = \min_{\varepsilon}\{\varepsilon | \varepsilon \geq G_j, j = 1, \ldots, J\}\,,$$
$$u^N \in \mathcal{U}^N\,,\quad x^* \in \mathcal{X}\,, \quad (21)$$

which is equivalent to the comparison procedure of [1].

Step 3. Problem (21) is a multiparametric linear programming problem; hence the solution consists of a set of piece-wise linear expressions for ψ_i in terms of the parameters u^N and x^* and a set of regions Ψ_i, $i = 1, \ldots, \hat{N}_{\text{reg}}$ where these expressions are valid. This statement was proven in [20], sect. 2.2, theorem 2.1, and in [10]. Note that no region Ψ_s exists such that $\psi_s \leq \psi_i$, $\forall\{x^*, u^N\} \in \Psi_s$ and $\forall i \neq s$ since ψ is convex. Thus, inequality (11) can be replaced by the inequalities $\psi_i(x^*, u^N) \leq 0$. In this way problem (13)–(14) can be recast as a single-level stochastic program:

$$\phi(x^*) = \min_{u^N \in \mathcal{U}^N}\{\Phi(x^*, u^N, \theta^{N,n}) |$$
$$\tilde{g}_j(x^*, u^N, \theta^{N,n}) \leq 0,\ j = 1, \ldots, J,$$
$$\psi(x^*, u^N) \leq 0, i = 1, \ldots, \hat{N}_{\text{reg}}\}\,, \quad (22)$$

where Φ is the quadratic objective (13) after substituting (9). The superscript n in $\theta^{N,n}$ denotes the nominal value of θ^N, which is usually zero. An approximate solution to the above stochastic problem can be obtained by discretizing the uncertainty space into a finite set of scenarios $\theta^{N,i}$, $i = 1, \ldots, ns$ with associated objective weights ([20]), thus leading to a multiperiod optimization problem where each period corresponds to a particular uncertainty scenario. By treating the control variables u^N as the optimization variables and the current state x^* as parameters, (22) is recast as multiparametric quadratic program.

Theorem 1 *The solution of (22) is a piece-wise linear control law $u_t(x^*) = \mathcal{A}_c x^* + b_c$ and $CR_c x^* + cr_c$, $c = 1, \ldots, N_c$ is the polyhedral critical region where this control law is valid and guarantees that (5) and (6) are feasible for all $\theta_{t+k} \in \Theta$, $k = 0, \ldots, N-1$.*

The proof of the theorem is straightforward from (21) and [20] and is omitted for brevity's sake. It shows that the solution to (22), and hence (13)–(14), can be obtained as an explicit multiparametric solution [9].

Solution of the Closed-Loop RpMPC Problem

In order to solve the problem (16)–(20), the inner max–min–max problem in (17)–(19) have to be solved parametrically and replaced by simpler linear inequalities, so the resulting problem is a simple multiparametric quadratic program. For simplicity, we only present an algorithm for solving the most difficult problem (19). The same thought process can be performed for the remaining constraints. The algorithm consists of the following steps:

Step 1. Solve

$$G_j^{\theta_{t+N-1}}(x^*, u^N, [\theta_{t+k}]_{k=0,\ldots,N-2})$$
$$= \max_{\theta_{t+N-1}}\{\tilde{g}_j(x^*, u^N, \theta^N), \theta^{N,L} \leq \theta^N \leq \theta^{N,U}\}\,,$$
$$j = 1, \ldots, J\,, \quad (23)$$

as a multiparametric optimization problem by recasting x^* and u^N as parameters and by following again the method of [19] or [20], sect. 2.2.

Step 2. Compare the parametric profiles $G_j^{\theta_{t+N-1}}(x^*, u^N, [\theta_{t+k}]_{k=0,\ldots,N-2})$ over the joint space of u^N, $[\theta_{t+k}]_{k=0,\ldots,N-2}$ and x^* to retain the upper bounds. For this comparison a multiparametric program is formulated and then solved by following the comparison procedure in [1]:

$$\psi^{\theta_{t+N-1}}(x^*, u^N, [\theta_{t+k}]_{k=0,\ldots,N-2}) = \max_j G_j^{\theta_{t+N-1}}$$
$$\Leftrightarrow \psi^{\theta_{t+N-1}}(x^*, u^N, [\theta_{t+k}]_{k=0,\ldots,N-2})$$
$$= \min_{\varepsilon}\{\varepsilon \mid \text{s.t. } G_j^{\theta_{t+N-1}} \leq \varepsilon,\ j = 1, \ldots, J\}\,. \quad (24)$$

The solution of the above optimization consists of a set of linear expressions for $\psi_i^{\theta_{t+N-1}}$ in terms of the parameters x^*, u^N, $[\theta_{t+k}]_{k=0,\ldots,N-2}$ and a set of polyhedral regions $\Psi_i^{\theta_{t+N-1}}$, $i = 1, \ldots, \hat{N}_{\text{reg}}^{\theta_{t+N-1}}$, where these expressions are valid.

Step 3. Set $\ell = N - 1$.

Step 4. Solve the following multiparametric optimization problem over u^{ℓ}

$$\psi^{u_{t+\ell}}(x^*, u^{\ell}, \theta^{\ell}) = \min_{u_{t+\ell} \in \mathcal{U}} \{\psi_i^{\theta_{t+\ell}}(x^*, [u_{t+k}]_{k=0,\ldots,\ell}, [\theta_{t+k}]_{k=0,\ldots,\ell-1}), \text{ if } \Psi_i^{\theta_{t+\ell}} \leq 0,\ i = 1, \ldots, \hat{N}_{\text{reg}}^{\theta_{t+\ell}}\}. \quad (25)$$

The above problem can be solved parametrically by following the procedure in [20], appendix A, or [17], chap. 3, sect. 3.2. The solution to (25) is a convex piecewise affine function of $\psi^{u_{t+\ell}}$ in terms of the parameters $x^*, u^N, [\theta_{t+k}]_{k=0,\ldots,N-2}$ that is defined over a set of polyhedral regions $\Psi_i^{u_{t+\ell}}$, $i = 1, \ldots, \hat{N}_{\text{reg}}^{u_{t+\ell}}$.

Step 5. Set $\ell = \ell - 1$ and solve the following maximization problem over $\theta^{\ell-1}$:

$$\psi^{\theta_{t+\ell}}(x^*, [u_{t+k}]_{k=0,\ldots,\ell}, [\theta_{t+k}]_{k=0,\ldots,\ell-1}) = \max_{\theta_{t+\ell}} \{\psi_i^{\theta_{t+\ell}}(x^*, [u_{t+k}]_{k=0,\ldots,\ell}, [\theta_{t+k}]_{k=0,\ldots,\ell-1}), \text{ if } \Psi_i^{u_{t+\ell+1}} \leq 0,\ i = 1, \ldots, \hat{N}_{\text{reg}}^{u_{t+\ell+1}}\}. \quad (26)$$

Since the function on the left-hand side of the above equality is a convex piecewise affine function, its maximization with respect to $[\theta_{t+k}]_{k=0,\ldots,\ell-1}$ reduces to the method of [19] followed by a comparison procedure as described in step 2.

Step 6. If $\ell > 0$, then go to step 4, else terminate the procedure and store the affine functions $\psi_i^{\theta_t}$, $i = 1, \ldots, \hat{N}_{\text{reg}}^{\theta_t}$.

Step 7. The expressions $\psi_i^{\theta_t}(u_t, x^*)$ are the max–min–max constraint (19). Similarly, the remaining max–min–max constraints are replaced by the set of inequalities

$$\begin{aligned}
&\psi_i^{\theta_{t+1}}(x^*, [u_t^T, u_{t+1}^T]^T, \theta_t) \leq 0\,,\\
&\ldots\,,\\
&\psi_i^{\theta_{t+N-2}}(x^*, [u_t^T, u_{t+1}^T, \ldots, u_{t+N-2}^T]^T, [\theta_t^T, \theta_{t+1}^T, \ldots, \theta_{t+N-3}^T]^T) \leq 0\,,\\
&\psi_i^{\theta_{t+N-1}}(x^*, [u_t^T, u_{t+1}^T, \ldots, u_{t+N-1}^T]^T, [\theta_t^T, \theta_{t+1}^T, \ldots, \theta_{t+N-2}^T]^T) \leq 0\,.
\end{aligned}$$

Substituting the inequalities in step 7 into the max–min–max constraints of (16)–(20) we obtain the following stochastic multparametric program:

$$\begin{aligned}
\phi(x^*) = \min_{u^N \in \mathcal{U}^N}\ & E_{\theta^N \in \Theta^N}\{\Phi(x^*, u^N, \theta^{N,n})\}\\
\text{s.t.}\ \ & \bar{g}_j(x^*, u^N, \theta^{N,n}) \leq 0\\
& \psi_i^{\theta_t}(x^*, u_t) \leq 0,\ i = 1, \ldots, \hat{N}_{\text{reg}}^{\theta_0}\\
& \psi_i^{\theta_{t+1}}(x^*, [u_t^T, u_{t+1}^T]^T, \theta_t^n),\ i = 1, \ldots, \hat{N}_{\text{reg}}^{\theta_1}\\
& \vdots\\
& \psi_i^{\theta_{t+N-2}}(x^*, [u_{t+k}]_{k=0,\ldots,N-2}, [\theta_{t+k}^n]_{k=0,\ldots,N-3}),\\
& \qquad i = 1, \ldots, \hat{N}_{\text{reg}}^{\theta_{t+N-2}}\\
& \psi_i^{\theta_{t+N-1}}(x^*, [u_{t+k}]_{k=0,\ldots,N-1}, [\theta_{t+k}^n]_{k=0,\ldots,N-2}),\\
& \qquad i = 1, \ldots, \hat{N}_{\text{reg}}^{\theta_{t+N-1}}\\
& x_{t|t} = x^*,\ j = 1, \ldots, J\,,
\end{aligned} \quad (27)$$

where Φ is again the quadratic objective function in (16). By discretizing the expectation of the value function to a set of discrete uncertainty scenarios and by treating the current state x^* as parameter and the control actions as optimization variables and the problem is recast as a parametric quadratic program. The solution is a complete map of the control variables in terms of the current state. The results for the closed-loop RpMPC controller are summarized in the following theorem.

Theorem 2 *The solution of (27) is obtained as a linear piecewise control law $u_t(x^*) = \mathcal{A}_c x^* + b_c$ and a set of polyhedral regions $CR_c = \{x^* \in X | CR_c x^* + cr_c \leq 0\}$ in the state space for which system (1) satisfies constraints (5)–(6) for all $\theta^N \in \Theta^N$.*

Cases

A special case of the RpMPC problem (2)–(8) arises when the system matrices in the first equation in (1) are uncertain in that their entries are unknown but bounded within specific bounds. For simplicity we will consider the simpler case where $W, F = 0$ and the entries a_{ij} and b_{ij} of matrices A and B are not known but satisfy

$$\begin{aligned}
&a_{ij} = \bar{a}_{ij} + \delta a_{ij},\ \ b_{i\ell} = \bar{b}_{i\ell} + \delta b_{i\ell}\\
&\delta a_{ij} \in \mathcal{A}_{ij} = \{\delta a_{ij} \in \mathbb{R} | -\varepsilon|\bar{a}_{ij}| \leq \delta a_{ij} \leq \varepsilon|\bar{a}_{ij}|\}\,,
\end{aligned} \quad (28)$$

$$\delta b_{i\ell} \in \mathcal{B}_{i\ell} = \{\delta b_{i\ell} \in \mathbb{R} | -\varepsilon|\bar{b}_{i\ell}| \leq \delta b_{i\ell} \leq \varepsilon|\bar{b}_{i\ell}|\}\,, \tag{29}$$

where $\bar{a}_{ij}$, $\bar{b}_{i\ell}$ are the nominal values of the entries of A, B respectively and δa_{ij}, $\delta b_{i\ell}$ denote the uncertainty in the matrix entries, which is assumed to be bounded as in (28)–(29). The general RpMPC formulation (2)–(8) must be redefined to include the introduced model uncertainty by adding the extra constraints

$$\begin{aligned} &a_{ij} = \bar{a}_{ij} + \delta a_{ij}, b_{i\ell} = \bar{b}_{i\ell} + \delta b_{i\ell} \\ &\forall \delta a_{ij} \in \mathcal{A}_{ij}, \forall \delta b_{i\ell} \in \mathcal{B}_{i\ell}\,, \end{aligned} \tag{30}$$

The new formulation of the RpMPC (2)–(8) and (30) gives rise to a semi-infinite dimensional problem with a rather high computational complexity.

Definition 2 A feasible solution u^N for problem (2)–(8) and (30), for a given initial state x^*, is called a *robust* or *reliable* solution.

Obviously, a robust solution for a given x^* is a control sequence u^N (future prediction vector) for which constraints (5)–(6) are satisfied for all admissible values of the uncertainty. Since it is difficult to solve this MPC formulation by the known parametric optimization methods, the problem must be reformulated in a multi-parametric quadratic programming (mpQP) form. Our objective in this section is to obtain such a form by considering the worst-case values of the uncertainty, i. e. those values of the uncertain parameters for which the linear inequalities of (5)–(6) are critically satisfied. Usually, the objective function (2) is formulated to penalize the nominal system behavior; thus one must substitute $x_{t+k|t} = \bar{A}^k x^* + \sum_{j=0}^{k-1} \bar{A}^j \bar{B} u_{t+k-1-j}$ in (2). In this way the objective function is a quadratic function of u^N and x^*. Finally, the uncertain evolution of the system $x_{t+k|t} = A^k x^* + \sum_{j=0}^{k-1} A^j B u_{t+k-1-j}$ is replaced in the constraints (5)–(6) to formulate a set of linear inequalities. Thus the following formulation of the RpMPC is obtained:

$$\phi(x^*) = \min_{u^N \in U^N} \left\{ \frac{1}{2}(u^N)^T H u^N + x^{*T} F u^N + \frac{1}{2}(x^*)^T Y x^* \right\}\,, \tag{31}$$

$$\text{s.t. } C_{1i}^T A^k x^* + \sum_{j=0}^{k-1} C_{1i}^T A^j B u_{t+k-1-j} + C_{2i}^T u_{t+k} + C_{3i} \leq 0\,, \quad k = 1, \ldots, N-1\,, \quad i = 1, \ldots, n_g\,, \tag{32}$$

$$D_{1\ell}^T A^N x^* + \sum_{j=0}^{N-1} D_{1\ell}^T A^j B u_{t+k-1-j} + D_{2\ell} \leq 0\,, \quad \ell = 1, \ldots, n_h\,, \tag{33}$$

$$\forall \delta a_{ij} \in \mathcal{A}_{ij}\,, \quad \forall \delta b_{i\ell} \in \mathcal{B}_{i\ell}\,, \quad i, j = 1, \ldots, n\,, \quad \ell = 1, \ldots, m\,. \tag{34}$$

It is evident that the new formulation of the RpMPC problem (31)–(34) is also a semi-infinite dimensional problem. This formulation can be further simplified if one considers that for any uncertain matrices A and B, the entries of the matrices A^k and $A^k B$ for all $k \geq 0$ are given respectively by [17]

$$a_{i\ell}^k = \bar{a}_{i\ell}^k + \delta a_{i\ell}^k, -\epsilon|\delta a_{i\ell,\min}^k| \leq \delta a_{i\ell}^k \leq \epsilon|\delta a_{i\ell,\max}^k|\,, \tag{35}$$

$$ab_{i\ell}^k = \bar{ab}_{i\ell}^k + \delta ab_{i\ell}^k, \quad -\epsilon|\delta ab_{i\ell,\min}^k| \leq \delta ab_{i\ell}^k \leq \epsilon|\delta ab_{i\ell,\max}^k|\,. \tag{36}$$

The analysis on (35)–(36) follows from [17], chap. 3, and is omitted for brevity's sake.

Robust Counterpart (RC) Problem

Using the basic properties of matrix multiplication and (35)–(36), problem (31)–(34) reformulates into

$$\phi(x^*) = \min_{u^N \in U^N} \left\{ \frac{1}{2}(u^N)^T H u^N + x^{*T} F u^N + \frac{1}{2}(x^*)^T Y x^* \right\}\,, \tag{37}$$

$$\sum_{j=1}^{k-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} C_{1iq} ab_{q\ell}^j u_{t+k-1-j,\ell} + \sum_{\ell} C_{2i\ell} u_{t+k,\ell} + \sum_{q=1}^{n} \sum_{\ell=1}^{n} C_{1iq} a_{q\ell}^k x_\ell^* + C_{3i} \leq 0\,, \tag{38}$$

$$k = 1, \ldots, N-1, i = 1, \ldots, n_g$$
$$\sum_{j=1}^{N-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} D_{1iq} a b_{q\ell}^{j} u_{t+k-1-j,\ell} + \sum_{q=1}^{n} \sum_{\ell=1}^{n} D_{1iq} a_{q\ell}^{k} x_{\ell}^{*} + D_{2i} \leq 0 \quad (39)$$

$$i = 1, \ldots, n_h \forall \delta a_{ij} \in \mathcal{A}_{ij},$$
$$\forall \delta b_{i\ell} \in \mathcal{B}_{i\ell}, i, j = 1, \ldots, n, \ell = 1, \ldots, m\,. \quad (40)$$

This is a robust multiparametric QP problem (robust mp-QP) where the coefficients of the linear inequalities in the constraints are uncertain, the vector u^N is the optimization variable and the initial states x^* are the parameters. A similar robust LP problem was studied in [5] where the coefficients of the linear constraints are uncertain, similar to (35)–(36); however, no multiparametric programming problems were considered.

In a similar fashion to the analysis in [5] we construct the *robust counterpart* of the robust mp-QP problem (37)–(40):

$$\phi(x^*) = \min_{u^N \in U^N} \left\{ \frac{1}{2} (u^N)^T H u^N + x^{*T} F u^N + \frac{1}{2} (x^*)^T Y x^* \right\}, \quad (41)$$

s.t.

$$\sum_{j=1}^{k-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} C_{1iq} \bar{ab}_{q\ell}^{j} u_{t+k-1-j,\ell} + \sum_{j=1}^{k-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} \epsilon \max\{|C_{1iq}||\delta a b_{q\ell,\min}^{k}|, |C_{1iq}||\delta a_{q\ell,\max}^{k}|\} |u_{t+k-1-j,\ell}| + \sum_{\ell} C_{2i\ell} u_{t+k,\ell}$$
$$+ \sum_{q=1}^{n} \sum_{\ell=1}^{n} C_{1iq} \bar{a}_{q\ell}^{k} x_{\ell}^{*} + \sum_{q=1}^{n} \sum_{\ell=1}^{n} \epsilon \max\{|C_{1iq}||\delta a_{q\ell,\min}^{k}|, |C_{1iq}||\delta a_{q\ell,\max}^{k}|\} |x_{\ell}^{*}| + C_{3i} \leq 0$$
$$k = 1, \ldots, N-1\,, \quad i = 1, \ldots, n_g\,, \quad (42)$$

$$\sum_{j=1}^{N-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} D_{1iq} \bar{ab}_{q\ell}^{j} u_{t+k-1-j,\ell} + \sum_{j=1}^{N-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} \max\{|D_{1iq}||\delta a b_{q\ell,\min}^{k}|, |D_{1iq}||\delta a b_{q\ell,\max}^{k}|\} |u_{t+k-1-j,\ell}|$$
$$+ \sum_{q=1}^{n} \sum_{\ell=1}^{n} D_{1iq} \bar{a}_{q\ell}^{k} x_{\ell}^{*} + \sum_{q=1}^{n} \sum_{\ell=1}^{n} \max\{|D_{1iq}||\delta a_{q\ell,\min}^{k}|, |D_{1iq}||\delta a_{q\ell,\max}^{k}|\} |x_{\ell}^{*}| + D_{2i} \leq 0$$
$$i = 1, \ldots, n_h\,, \quad (43)$$

$$u^N \in \mathcal{U}^N\,, \quad x^* \in \mathcal{X}\,. \quad (44)$$

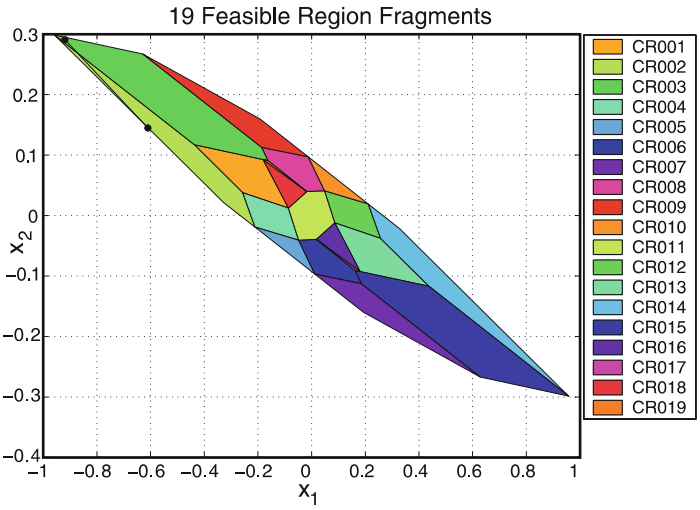
In this way the initial semi-infinite dimensional problem (37)–(40) becomes the above multiparametric non-linear program (mp-NLP). However, the parametric solution of this mp-NLP problem is still very difficult.

Interval Robust Counterpart Problem

The interval robust counterpart (IRC) problem can then be formulated as follows:

$$\phi(x^*) = \min_{u^N \in U^N} \left\{ \frac{1}{2} (u^N)^T H u^N + x^{*T} F u^N + \frac{1}{2} (x^*)^T Y x^* \right\}, \quad (45)$$

s.t.
$$\sum_{j=1}^{k-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} C_{1iq} \bar{ab}_{q\ell}^{j} u_{t+k-1-j,\ell} + \sum_{j=1}^{k-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} \epsilon \max\{|C_{1iq}||\delta a b_{q\ell,\min}^{k}|, |C_{1iq}||\delta a_{q\ell,\max}^{k}|\} z_{t+k-1-j,\ell} + \sum_{\ell} C_{2i\ell} u_{t+k,\ell}$$
$$+ \sum_{q=1}^{n} \sum_{\ell=1}^{n} C_{1iq} \bar{a}_{q\ell}^{k} x_{\ell}^{*} + \sum_{q=1}^{n} \sum_{\ell=1}^{n} \epsilon \max\{|C_{1iq}||\delta a_{q\ell,\min}^{k}|, |C_{1iq}||\delta a_{q\ell,\max}^{k}|\} w_{\ell} + C_{3i} \leq 0$$
$$k = 1, \ldots, N-1\,, \quad i = 1, \ldots, n_g\,, \quad (46)$$



Design of Robust Model-Based Controllers via Parametric Programming, Figure 1
Critical regions for the nominal parametric MPC and state trajectory

$$\sum_{j=1}^{N-1}\sum_{q=1}^{n}\sum_{\ell=1}^{m} D_{1iq}\bar{ab}^{j}_{q\ell}u_{t+k-1-j,\ell} + \sum_{j=1}^{N-1}\sum_{q=1}^{n}\sum_{\ell=1}^{m} \max\{|D_{1iq}||\delta ab^{k}_{q\ell,\min}|, |D_{1iq}||\delta ab^{k}_{q\ell,\max}|\}z_{t+k-1-j,\ell} + \sum_{q=1}^{n}\sum_{\ell=1}^{n} D_{1iq}\bar{a}^{k}_{q\ell}x^{*}_{\ell} + \sum_{q=1}^{n}\sum_{\ell=1}^{n} \max\{|D_{1iq}||\delta a^{k}_{q\ell,\min}|, |D_{1iq}||\delta a^{k}_{q\ell,\max}|\}w_{\ell} + D_{2i} \le 0$$

$$i = 1,\ldots,n_h\,, \tag{47}$$

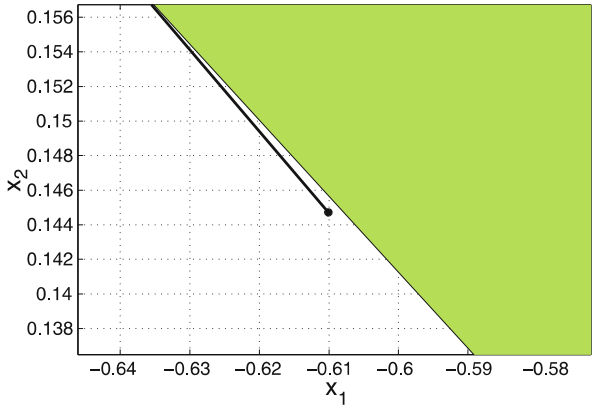
$$-z_{t+k-1-j,\ell} \le u_{t+k-1-j,\ell} \le z_{t+k-1-j,\ell}\,, \tag{48}$$

$$-w_{\ell} \le x^{*}_{\ell} \le w_{\ell}\,, \tag{49}$$

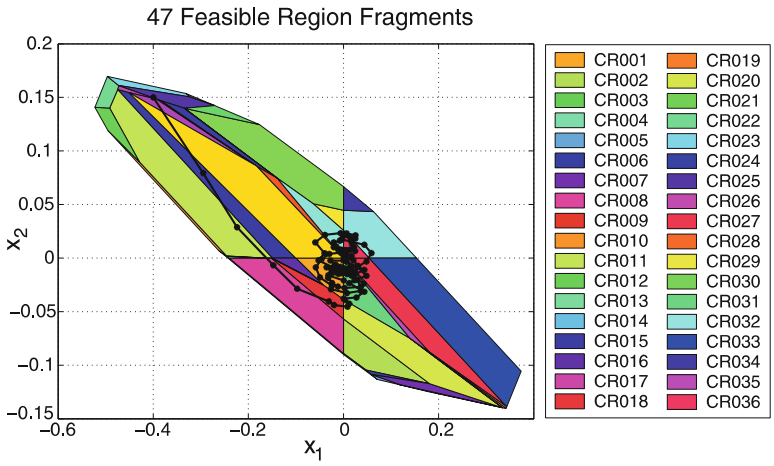
$$u^{N} \in \mathcal{U}^{N}\,, \quad x^{*} \in \mathcal{X}\,, \tag{50}$$

where the non-linear inequalities (42)–(43) have been replaced by four new linear inequalities. Two new variables have been introduced to replace the absolute values of the $u_{t+k-1-j,\ell}$ and x^{*}_{ℓ}, thus leading to the relaxed IRC problem.

The IRC is a mpQP problem with a quadratic index and linear inequalities, where the optimization variables now are the vectors $u_{t+k-1-j}$, $z_{t+k-1-j}$ and w and the parameters are the states x^{*}. The IRC problem can be solved with the known parametric optimization methods [4,9,16] since the objective function is strictly convex by assumption. The optimal control inputs u^{N}, optimization variables z and w and hence the optimal control u_t can then be obtained as explicit functions $u^{N}(x^{*})$, $z(x^{*})$ and $w(x^{*})$ of the initial state x^{*}. Furthermore, the control input u_t is obtained as the explicit, optimal control



Design of Robust Model-Based Controllers via Parametric Programming, Figure 2
Magnification of Fig. 1 around the state trajectory at the second time instant



Design of Robust Model-Based Controllers via Parametric Programming, Figure 3
Critical regions for the nominal parametric MPC and state trajectory

law [9] $u_t(x^*) = \mathcal{A}_c x^* + b_c$ which is valid in the polyhedral region $C\mathcal{R}_c = \{x^* \in \mathcal{X} | CR_c x^* + cr \leq 0\}$, $c = 1, \ldots, N_c$, where N_c is the number of critical regions obtained from the parametric programming algorithm.

The general RpMPC problem obtained from the case where the dynamic system (1) pertains to model uncertainties have now been transformed into the IRC problem and can be solved as a mp-QP problem. It is obvious that a feasible solution for the IRC problem is also a feasible solution for the RC and hence the initial RpMPC problem (2)–(8) and (30). Hence:

Lemma 1 *If u^N is a feasible solution for the IRC problem, then it is also a feasible solution for the RC problem, and hence it is a robust solution for the initial RpMPC problem (2)–(8), (30).*

Example 2 Consider a two-dimensional, discrete-time linear system (1) where $W = F = 0$ and

$$A = \begin{bmatrix} 0.7326 + \delta a & -0.0861 \\ 0.1722 & 0.0064 \end{bmatrix}, \quad B = \begin{bmatrix} 0.0609 + \delta b \\ 0.0064 \end{bmatrix}, \tag{51}$$

where the entries a_{11} and b_1 of the A and B matrices are uncertain, where δa and δb are bounded as in (28)–(29) with $\epsilon = 10\%$ and the nominal values are $\bar{a}_{11} = 0.7326$ and $\bar{b}_1 = 0.0609$. The state and control constraints are $-3 \leq [0 \;\; 1.4142]^T x \leq 3$, $-2 \leq u \leq 2$, and the terminal constraint is

$$\begin{bmatrix} 0.070251 & 1 \\ -0.070251 & -1 \\ 0.21863 & 1 \\ -0.21863 & -1 \end{bmatrix} x \leq \begin{bmatrix} 0.02743 \\ 0.02743 \\ 0.022154 \\ 0.022154 \end{bmatrix}. \tag{52}$$

Moreover,

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} R = 0.01, \quad P = \begin{bmatrix} 1.8588 & 1.2899 \\ 1.2899 & 6.7864 \end{bmatrix}. \tag{53}$$

Initially, the MPC problem (2)–(8) is formulated and solved only for the nominal values of A and B, thus solving a multiparametric quadratic programming problem as described in [4,16]. Then the IRC problem is formulated as in (45)–(50) by using POP software [9]. The resulting regions for both cases are shown in Figs. 1 and 3 respectively. A simulation of the state trajectories of the nominal and the uncertain system are shown in Figs. 1 and 3 respectively. In these simulations the uncertain parameters δa and δb were simulated as a sequence of random numbers that take their values on the upper or lower bounds of δa, δb i. e. a time-varying uncertainty. It is clear from Fig. 1 (and Fig. 2, which displays the magnified area around the state trajectory at the second

time instant) that the nominal solution to problem (2)–(8) cannot guarantee robustness in the presence of the uncertainty and the nominal system trajectory results in constraint violation. On the other hand, the controller obtained with the method discussed here manages to retain the trajectory in the set of feasible initial states (obtained by the critical regions of the parametric solution) and drives the trajectory close to the origin. One should notice that the space of feasible initial states (Fig. 3) given by the critical regions of the parametric solution is smaller than the one given in the nominal system's case (Fig. 1).

Conclusions

In this chapter two robust parametric MPC problems were analyzed. In the first problem two methods for robust parametric MPC are discussed, an open-loop and a closed-loop method, for treating robustness issues arising from the presence of input disturbances/uncertainties. In the second problem, a robust parametric MPC procedure was discussed for the control of dynamic systems with uncertainty in the system matrices by employing robust parametric optimization methods.

References

1. Acevedo J, Pistikopoulos EN (1999) An algorithm for multiparametric mixed-integer linear programming problems. Oper Res Lett 24:139–148
2. Bemporad A, Borelli F, Morari M (2003) Min–max control of constrained uncertain discrete-time linear systems. IEEE Trans Autom Contr 48(9):1600–1606
3. Bemporad A, Morari M (1999) Robust model predictive control: a survey Robustness in indentification and control. Springer, Berlin, pp 207–226
4. Bemporad A, Morari M, Dua V, Pistikopoulos EN (2002) The explicit linear quadratic regulator for constrained systems. Automatica 38:3–20
5. Ben-Tal A, Nemirovski A (2000) Robust solutions of linear programming problems contaminated with uncertain data. Math Program 88:411–424
6. Bertsekas DP, Rhodes IB (1971) On the minimax reachability of target sets and target tubes. Automatica 7:233–247
7. Camacho E, Bordons C (1999) Model Predictive Control. Springer, Berlin
8. Chisci L, Rossiter JA, Zappa G (2001) Systems with persistent disturbances: predictive control with restricted constraints. Automatica 37:1019–1028
9. Dua V, Bozinis NA, Pistikopoulos EN (2002) A multiparametric programming approach for mixed integer and quadratic engineering problems. Comput Chem Eng 26:715–733
10. Fiacco A (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Academic, New York
11. Kothare MV, Balakrishnan V, Morari M (1996) Robust constrained model predictive control using linear matrix inequalities. Automatica 32(10):1361–1379
12. Langson W, Chryssochoos I, Raković SV, Mayne DQ (2004) Robust model predictive control using tubes. Automatica 40:125–133
13. Lee J, Cooley B (1997) Recent advances in model predictive control and other related areas. In: Carnahan B, Kantor J, Garcia C (eds) Proceedings of chemical process control – V: Assesment and new directions for research, vol 93 of AIChE Symposium Series No. 316, AIChE and CACHE, pp 201–216
14. Mayne D, Rawlings J, Rao C, Scokaert PO (2000) Constrained model predictive control: stability and optimality. Automatica 36:789–814
15. Morari M, Lee J (1999) Model predictive control: past, present and future. Comput Chem Eng 23:667–682
16. Pistikopoulos EN, Dua V, Bozinis NA, Bemporad A, Morari M (2002) On-line optimization via off-line parametric optimization tools. Automatica 26:175–185
17. Pistikopoulos EN, Georgiadis M, Dua V (eds) (2007) Multiparametric Model-Based Control. In: Process Systems Engineering, vol 2. Wiley-VCH, Weinheim
18. Pistikopoulos EN, Georgiadis M, Dua V (eds) (2007) Multiparametric Programming. In: Process Systems Engineering, vol 1. Wiley-VCH, Weinheim
19. Pistikopoulos EN, Grossmann I (1988) Optimal retrofit design for improving process flexibility in linear systems. Comput Chem Eng 12(7):719–731
20. Sakizlis V, Kakalis NMP, Dua V, Perkins JD, Pistikopoulos EN (2004) Design of robust model-based controllers via parametric programming. Automatica 40:189–201
21. Scokaert P, Mayne D (1998) Min–max feedback model predictive control for constrained linear systems. IEEE Trans Autom Contr 43(8):1136–1142
22. Wang YJ, Rawlings JB (2004) A new robust model predictive control method I: theory and computation. J Process Control 14:231–247

Determining the Optimal Number of Clusters

Meng Piao Tan,
Christodoulos A. Floudas
Department of Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C26, 91C20, 68T20, 68W10, 90C11, 92-08, 92C05, 92D10

Article Outline

Introduction
Methods
Dunn's Validity Index
Davies–Bouldin Validity Index
Measure of Krzanowski and Lai
Measure of Calinski and Harabasz
Applications
A Novel Clustering Approach with Optimal Cluster Determination
Extension for Biological Coherence Refinement
References

Introduction

Clustering is probably the most important unsupervised learning problem and involves finding coherent structures within a collection of unlabeled data. As such it gives rise to data groupings so that the patterns are similar within each group and remote between different groups. Besides having been extensively applied in areas such as image processing and pattern recognition, clustering also sees rich applications in biology, market research, social network analysis, and geology. For instance, in marketing and finance, cluster analysis is used to segment and determine target markets, position new products, and identify clients in a banking database having a heavy real estate asset base. In libraries, clustering is used to aid in book ordering and in insurance, clustering helps to identify groups of motor insurance policy holders with high average claim costs. Given its broad utility, it is unsurprising that a substantial number of clustering methods and approaches have been proposed.

On the other hand, fewer solutions to systematically evaluate the quality or validity of clusters have been presented [1]. Indeed, the prediction of the optimal number of groupings for any clustering algorithm remains a fundamental problem in unsupervised classification. To address this issue, numerous cluster indices have been proposed to assess the quality and the results of cluster analysis. These criteria may then be used to compare the adequacy of clustering algorithms and different dissimilarity measures, or to choose the optimal number of clusters. Some of these measures are introduced in the following section.

Methods

Dunn's Validity Index

This technique [2,5] is based on the idea of identifying the cluster sets that are compact and well separated. For any partition of clusters, where c_i represent the ith cluster of such a partition, Dunn's validation index, D, can be calculated as

$$D = \min_{1\le i\le n}\left\{\min_{\substack{1\le j\le n\\ i\neq j}}\left\{\frac{d(c_1,c_j)}{\max_{1\le k\le n} d'(c_k)}\right\}\right\}.$$

Here, $d(c_i,c_j)$ is the distance between clusters c_i, and c_j (intercluster distance), $d'(c_k)$ is the intracluster distance of cluster c_k, and n is the number of clusters. The goal of this measure is to maximize the intercluster distances and minimize the intracluster distances. Therefore, the number of cluster that maximizes D is taken as the optimal number of clusters to be used.

Davies–Bouldin Validity Index

This index [4] is a function of the ratio of the sum of within-cluster scatter to between-cluster separation:

$$DB = \frac{1}{n}\sum_{i=1}^{n}\max_{i\neq j}\left\{\frac{S_n(Q_i)+S_n(Q_j)}{S(Q_i,Q_j)}\right\}.$$

In this expression, DB is the Davies–Bouldin index, n is the number of clusters, S_n is the average distance of all objects from the cluster to their cluster center, and $S(Q_iQ_j)$ is the distance between cluster centers. Hence, the ratio is small if the clusters are compact and far from each other. Consequently, the Davies–Bouldin index will have a small value for a good clustering.

The silhouette validation technique [22] calculates the silhouette width for each sample, the average silhouette width for each cluster, and the overall average silhouette width for a total data set. With use of this approach each cluster can be represented by a so-called silhouette, which is based on the comparison of its tightness and separation. The average silhouette width can be applied for the evaluation of clustering validity and can also be used to decide how good are the number of selected clusters. To construct the silhouettes S(i) the

following formula is used:

$$S(i) = \frac{(b(i) - a(i))}{\max\{a(i), b(i)\}} .$$

Here, $a(i)$ is the average dissimilarity of the ith object to all other objects in the same cluster and $b(i)$ is the minimum average dissimilarity of the ith object to all objects in the other clusters.

It follows from the formula that s(i) lies between -1 and 1. If the silhouette value is close to 1, it means that sample is "well clustered" and has been assigned to a very appropriate cluster. If the silhouette value is close to 0, it means that that sample could be assigned to another "closest" cluster as well, and the sample lies equally far away from both clusters. If the silhouette value is close to -1, it means that sample is "misclassified" and is merely somewhere in-between the clusters. The overall average silhouette width for the entire plot is simply the average of the $S(i)$ for all objects in the whole dataset and the largest overall average silhouette indicates the best clustering (number of clusters). Therefore, the number of clusters with the maximum overall average silhouette width is taken as the optimal number of the clusters.

Measure of Krzanowski and Lai

This index is based on the decrease of the within-cluster sum of squares (WSS) [15] and is given by

$$\mathrm{KL}(k) = \left| \frac{\mathrm{DIFF}(k)}{\mathrm{DIFF}(k+1)} \right|, \quad \text{where}$$
$$\mathrm{DIFF}(k) = (k-1)^{\frac{2}{p}}\,\mathrm{WSS}(k-1) - k^{\frac{2}{p}}\mathrm{WSS}(k) .$$

Assuming that g is the ideal cluster number for a given dataset, and k is a particular number of clusters, then WSS(k) is assumed to decrease rapidly for $k \le g$ and decreases only slightly for $k > g$. Thus, it is expected that KL(k) will be maximized for the optimal number of clusters.

Measure of Calinski and Harabasz

This method [3] assesses the quality of k clusters via the index

$$\mathrm{CH}(k) = \frac{\mathrm{BSS}(k-1)/(k-1)}{\mathrm{WSS}(k)/(n-k)} .$$

Here, WSS(k) and BSS(k) are the WSS and the between-cluster sums of squares, for a dataset of n members. The measure seeks to choose clusters that are well isolated from one another and coherent, but at the same time keep the number of clusters as small as possible, thus maximizing the criterion at the optimal cluster number. Incidentally, a separate study comparing 28 validation criteria [18] found this measure to perform the best.

In addition, some other measures to determine the optimal number of clusters are (i) the C index [10], (ii) the Goodman–Kruskal index [8]), (iii) the isolation index [19], (iv) the Jaccard index [11], and (v) the Rand index [20].

Applications

As can be seen, while it is relatively easy to propose indices of cluster validity, it is difficult to incorporate these measures into clustering algorithms and to appoint suitable thresholds on which to define key decision values [9,12]. Most clustering algorithms do not contain built-in screening functions to determine the optimal number of clusters. This implies that for a given clustering algorithm, the most typical means of determining the optimal cluster number is to repeat the clustering numerous times, each with a different number of groupings, and hope to catch a maximum or minimum turning point for the cluster validity index in play.

Nonetheless, there have been attempts to incorporate measures of cluster validity into clustering algorithms. One such method [21] introduces a validity index:

$$\mathit{Validity} = \frac{\mathit{Intra} - \mathit{Cluster}}{\mathit{Inter} - \mathit{Cluster}} .$$

Since it is desirable for the intracluster distance and the intercluster distance to be minimized and maximized, respectively, the above validity measure should be as small as possible. Using the K-means algorithm, Ray and Turi [21] proposed running the process for two up to a predetermined maximum number of clusters. At each stage, the cluster with the maximum variance is split into two and clustering is repeated with these updated centers, until the desired turning point for the validity measure is observed. Another approach [16] is based on simulated annealing, which was originally formulated to simulate a collection of atoms in equilibrium at a given temperature [14,17]. It assumes two

given parameters D, which is the cutoff cluster diameter, and P, a P-value statistic, as well as $p(d)$, the distribution function of the Euclidean distances between the members in a dataset. Then, the upper boundary for the fraction of incorrect vector pairs is given by

$$f(D, K = 1) = \int_D^{\infty} p(x)\,\mathrm{d}x\ .$$

On the other hand, it is possible to define a lower boundary for $f(D,K)$ with a preassigned P-value cutoff. The clustering algorithm then sequentially increases the cluster number until the two indicators converge.

A Novel Clustering Approach with Optimal Cluster Determination

See also the article on "Gene Clustering: A Novel Optimization-Based Approach".

Recently, we proposed a novel clustering approach [23,24] that expeditiously contains a method to predict the optimal cluster number. The clustering seeks to minimize the Euclidean distances between the data and the assigned cluster centers as

$$\underset{w_{ij},z_{jk}}{\mathrm{MIN}} \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} w_{ij}\left(a_{ik} - z_{jk}\right)^2\ .$$

To make the nonlinear problem tractable, we apply a variant of the generalized benders decomposition algorithm [6,7], the global optimum search. The global optimum search decomposes the problem into a primal problem and the master problem. The former solves the continuous variables while fixing the integer variables and provides an upper-bound solution, while the latter finds the integer variables and the associated Lagrange multipliers while fixing the continuous variables and provides a lower-bound solution. The two sequences are iteratively updated until they converge at an optimal solution in a finite number of steps.

In determining the optimal cluster number, we note that the optimal number of clusters occurs when the intercluster distance is maximized and the intracluster distance is minimized. We adapt the novel work of Jung et al. [13] in defining a clustering balance, which has been shown to have a minimum value when intracluster similarity is maximized and intercluster similarity is minimized. This provides a measure of how optimal is a certain number of clusters used for a particular clustering algorithm. Given n data points, each having k feature points, j clusters, and a binary decision variable for cluster membership w_{ij}, we introduce the following:

$$\text{Global center, } z_k^o = \frac{1}{n}\sum_{i=1}^{n} a_{ik}, \quad \forall k\ ,$$

$$\text{Intracluster error sum, } \Lambda = \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} w_{ij}\left\|a_{ik} - z_{jk}\right\|_2^2\ ,$$

$$\text{Intercluster error sum, } \Gamma = \sum_{j=1}^{c}\sum_{k=1}^{s}\left\|z_{jk} - z_k^o\right\|_2^2\ .$$

Jung et al. [13] next proposed a clustering balance parameter, which is the α-weighted sum of the two error sums:

$$\text{Clustering balance, } \quad \varepsilon = \alpha\Lambda + (1-\alpha)\,\Gamma\ .$$

We note here that the right α ratio is 0.5. There are two ways to come to this conclusion. We note that the factor α should balance the contributive weights of the two error sums to the clustering balance. At extreme cluster numbers, that is, the largest and smallest numbers possible, the sum of the intracluster and intercluster error sums at both cluster numbers should be balanced. In the minimal case, all the data points can be placed into a single cluster, in which case the intercluster error sum is zero and the intracluster error sum can be calculated with ease. In the maximal case, each data point forms its own cluster, in which case the intracluster error sum is zero and the intercluster error sum can be easily found. Obviously the intracluster error sum in the minimal case and the intercluster error sum in the maximal case are equal, suggesting that the most appropriate weighting factor to use is in fact 0.5. The second approach uses a clustering gain parameter proposed by Jung et al. [13]. This gain parameter is the difference between the decreased intercluster error sum γ_j compared with the value at the initial stage and the increased intracluster error sum λ_j compared with the value at the initial stage, and is given by

$$\gamma_{jk} = \sum_{i=1}^{n} w_{ij}\left\|a_{ik} - z_k^o\right\|_2^2 - \left\|z_{jk} - z_k^o\right\|_2^2\ , \quad \forall j, \forall k\ ,$$

$$\lambda_{jk} = \sum_{i=1}^{n} w_{ij}\left\|a_{ik} - z_{jk}\right\|_2^2, \ \forall j, \forall k\ ,$$

$$Gain,\ \Delta_{jk} = \sum_{i=1}^{n} w_{ij} \left\| a_{ik} - z_k^o \right\|_2^2 - \left\| z_{jk} - z_k^o \right\|_2^2 - \sum_{i=1}^{n} w_{ij} \left\| a_{ij} - z_{jk} \right\|_2^2,\ \forall \mathrm{j}, \forall \mathrm{k}\,.$$

With the identities

$$\sum_{i=1}^{n} w_{ij} a_{ik} = n_j z_{jk},\ \forall j,\ \forall k\,,$$
$$\sum_{i=1}^{n} w_{ij} = n_j,\ \forall j\,,$$

where n_j denotes the number of data points in cluster j, the gain can be simplified to

$$\Delta_{\mathrm{jk}} = \left(n_j - 1\right) \left\| z_k^o - z_{jk} \right\|_2^2, \forall \mathrm{j}, \forall \mathrm{k}\,,$$
$$\Delta = \sum_{\mathrm{j}=1}^{c} \sum_{k=1}^{s} \left(n_j - 1\right) \left\| z_k^o - z_{jk} \right\|_2^2.$$

Jung et al. [13] showed the clustering gain to have a maximum value at the optimal number of clusters, and demonstrated that the sum total of the clustering gain and balance parameters is a constant. As can be seen from the following derivation, this is only possible if the α ratio is 0.5:

$$\begin{aligned}
&\text{Sum of clustering balance and clustering gain, } \Omega \\
&= \varepsilon + \Delta \\
&= \Lambda + \Gamma + \Delta \\
&= \left[\sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{s} w_{ij} \left\| a_{ik} - z_{jk} \right\|_2^2 \right] \\
&\quad + \left[\sum_{j=1}^{c} \left\| z_{jk} - z_k^o \right\|_2^2 \right] + \ldots \\
&\quad \left[\begin{array}{l} \sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{s} w_{ij} \left\| a_{ik} - z_k^o \right\|_2^2 - \sum_{j=1}^{c} \left\| z_{jk} - z_k^o \right\|_2^2 \\ - \sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{s} w_{ij} \left\| a_{ik} - z_{jk} \right\|_2^2 \end{array} \right] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{s} w_{ij} \left\| a_{ik} - z_k^o \right\|_2^2 \\
&= \sum_{i=1}^{n} \sum_{k=1}^{s} \left\| a_{ik} - z_k^o \right\|_2^2,
\end{aligned}$$

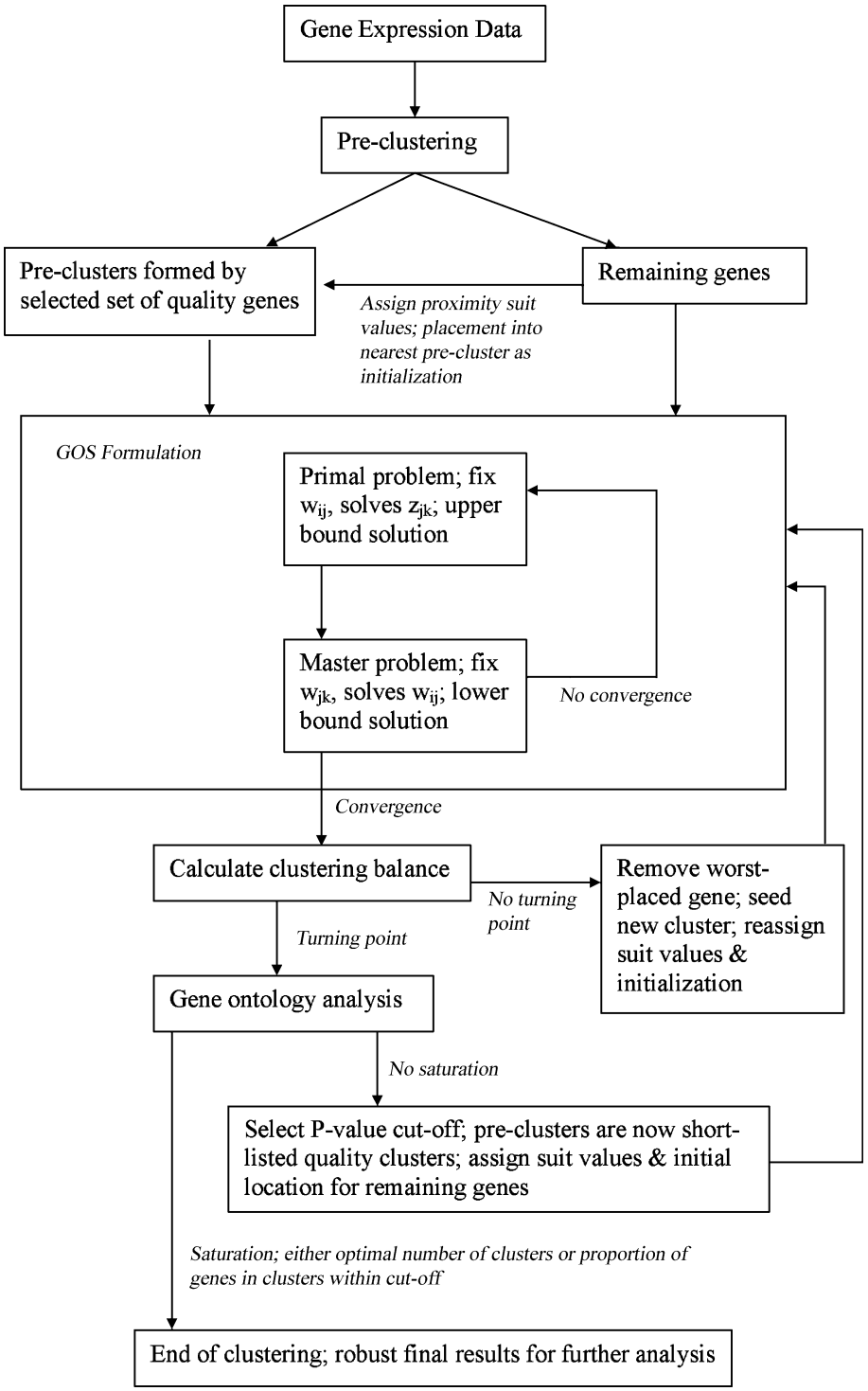
which is a constant for any given dataset.

Extension for Biological Coherence Refinement

Today, the advent of DNA microarray technology has made possible the large-scale monitoring of genomic behavior. In working with gene expression data, it is often useful to utilize external validation in evaluating clusters of gene expression data. Besides assessing the biological meaning of a cluster through the functional annotations of its constituent genes using gene ontology resources, other indications of strong biological coherence [25] are (i) the proportion of genes that reside in clusters with good P-value scores, (ii) cluster correlation, since closely related genes are expected to exhibit very similar patterns of expression, and (iii) cluster specificity, which is the proportion of genes within a cluster that annotates for the same function. A novel extension of the previously described work [25] allows not just for the determination of the optimal cluster number within the framework of a robust yet intuitive clustering method, but also for an iterative refinement of biological validation for the clusters. The algorithm is as follows.

Gene Preclustering We precluster the original data by proximity studies to reduce the computational demands by (i) identifying genes with very similar responses and (ii) removing outliers deemed to be insignificant to the clustering process. To provide just adequate discriminatory characteristics, preclustering can be done by reducing the expression vectors into a set of representative variables $\{+, o, -\}$, or by pregrouping genes that are close to one another by correlation or some other distance function.

Iterative Clustering We let the initial clusters be defined by the genes preclustered previously, and find the distance between each of the remaining genes and these initial clusters and as a good initialization point place these genes into the nearest cluster. For each gene, we allow its suitability in a limited number of clusters on the basis of the proximity study. In the primal problem of the global optimum search algorithm, we solve for z_{jk}. These, together with the Lagrange multipliers, are used in the master problem to solve for w_{ij}. The primal problem gives an upper-bound solution and the master problem gives a lower bound. The optimal solution is obtained when both bounds converge. Then, the worst-



Determining the Optimal Number of Clusters, Figure 1
Iterative clustering procedure. *GOS* global optimum search

placed gene is removed and used as a seed for a new cluster. This gene has already been subjected to a membership search, so there is no reason for it to belong to any of the older clusters. The primal and master problems are iterated and the number of clusters builds up gradually until the optimal number is attained.

Iterative Extension Indication of strong biological coherence is characterized by good P values based on gene ontology resources and the proportion of genes that reside in such clusters. As an extension, we would like to mine for the maximal amount of relevant information from the gene expression data and sieve out the least relevant data. This is important because information such as biological function annotation drawn from the cluster content is often used in the further study of coregulated gene members, common reading frames, and gene regulatory networks. From the clustered genes, we impose a coherence floor, based on some or all of the possible performance factors such as functional annotation, cluster specificity, and correlation, to demarcate genes that have already been well clustered. We then iterate to offer the poorly placed genes an opportunity to either find relevant membership in one of the strongly coherent clusters, or regroup amongst themselves to form quality clusters. Through this process, a saturation point will be reached eventually whereby the optimal number of clusters becomes constant as the proportion of genes distributed within clusters of high biological coherence levels off. Figure 1 shows a schematic of the entire clustering algorithm.

References

1. Azuaje F (2002) A Cluster Validity Framework for Genome Expression Data. Bioformatics 18:319–320
2. Bezdek JC, Pal NR (1998) Some New Indexed of Cluster Validity. IEEE Trans Syst Man Cybern 28:301–315
3. Calinski RB, Harabasz J (1974) A Dendrite Method for Cluster Analysis. Commun Stat 3:1–27
4. Davis DL, Bouldin DW (1979) A Cluster Separation Measure. IEEE Trans Pattern Anal Machine Intell 1(4):224–227
5. Dunn JC (1974) Well Separated Clusters and Optimal Fuzzy Partitions. J Cyber 4:95–104
6. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, New York
7. Floudas CA, Aggarwal A, Ciric AR (1989) Global Optimum Search for Non Convex NLP and MINLP Problems. Comp Chem Eng 13(10):1117–1132
8. Goodman L, Kruskal W (1954) Measures of Associations for Cross-Validations. J Am Stat Assoc 49:732–764
9. Halkidi M, Batistakis Y, Vazirgiannis M (2002) Cluster Validity Methods: Part 1. SIGMOD Record 31(2):40–45
10. Hubert L, Schultz J (1976) Quadratic Assignment as a General Data-Analysis Strategy. Brit J Math Stat Psych 29: 190–241
11. Jaccard P (1912) The Distribution of Flora in the Alpine Zone. New Phytol 11:37–50
12. Jain AK, Dubes RC (1988) Algorithms for Clustering Data. Prentice-Hall Advanced Reference Series, Prentice-Hall, New Jersey
13. Jung Y, Park H, Du D, Drake BL (2003) A Decision Criterion for the Optimal Number of Clusters in Hierarchical Clustering. J Global Optim 25:91–111
14. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by Simulated Annealing. Science 220(4598):671–680
15. Krzanowski WJ, Lai YT (1985) A Criterion for Determining the Number of Groups in a Data Set using Sum of Squares Clustering. Biometrics 44:23–44
16. Lukashin AV, Fuchs R (2001) Analysis of Temporal Gene Expression Profiles: Clustering by Simulated Annealing and Determining the Optimal Number of Clusters. Bioinformatics 17(5):405–414
17. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller EJ (1953) Equations of State Calculations by Fast Computing Machines. J Chem Phys 21:1087
18. Milligan GW, Cooper MC (1985) An Examination of Procedures for Determining the Number of Clusters in a Data Set. Psychometrika 50:159–179
19. Pauwels EJ, Fregerix G (1999) Finding Salient Regions in Images: Non-parametric Clustering for Image Segmentation and Grouping. Comput Vis Image Underst 75:73–85
20. Rand WM (1971) Objective Criteria for the Evaluation of Clustering Methods. J Am Stat Assoc 66(336):I846–850
21. Ray S, Turi R (1999) Determination of Number of Clusters in K-Means Clustering and Application in Color Image Segmentation. In: Proceed 4th Int Conf Advances in Pattern Recognition and Digital Techniques, 137–143
22. Rousseeuw PJ (1987) Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. J Comp Appl Math 20:53–65
23. Tan MP, Broach JR, Floudas CA (2007) A Novel Clustering Approach and Prediction of Optimal Number of Clusters: Global Optimum Search with Enhanced Positioning. J Global Optim 39:323–346
24. Tan MP, Broach JR, Floudas CA (2008) Evaluation of Normalization and Pre-Clustering Issues in a Novel Clustering Approach: Global Optimum Search with Enhanced Positioning. J Bioinf Comput Biol 5(4):895–913
25. Tan MP, Broach JR, Floudas CA (2008) Microarray Data Mining: A Novel Optimization-Based Iterative Clustering Approach to Uncover Biologically Coherent Structures (submitted for publication)

Deterministic and Probabilistic Optimization Models for Data Classification

YA-JU FAN, W. ART CHAOVALITWONGSE
Department of Industrial and Systems Engineering, Rutgers University, Piscataway, USA

MSC2000: 65K05, 55R15, 55R35, 90C11

Article Outline

Deterministic Optimization Models
- Support Vector Machines
- Robust LP for SVM
- Feature Selection with SVM
- Hybrid LP Discriminant Model
- MIP Discriminant Model
- Multi-hyperplane Classification
- Support Feature Machines

Probabilistic Optimization Models
- Bayesian-Based Mathematical Program
- Probabilistic Models for Classification
- MIP Formulation for Anderson's Model
- LP Formulation for Anderson's Model

References

A classification problem is concerned with categorizing a data point (entity) into one of G ($G \geq 2$) mutually exclusive groups based upon m (positive integer) specific measurable features of the entity. A classification rule is typically constructed from a sample of entities, where the group classifications are known or labeled (training or supervised learning). Then it can be used to classify new unlabeled entities. Many classification methods are based on distance measures. A common approach is to find a hyperplane to classify two groups ($G = G_1 \bigcup G_2$). The hyperplane can be represented in a form of $A\omega = \gamma$, where A denotes an $n \times m$ input data matrix, n is the total number of input data points, and m is the total number of data features/attributes. The classification rule is then made by the weight vector ω to map data points onto a hyperplane, and the scalar γ, which are best selected by solving a mathematical programming model. The goal is to have entities of Group 1 (G_1) lie on one side of the hyperplane and entities of Group 2 (G_2) lie on the other side. Support Vector Machines (SVM) is the most studied hyperplane construction method. The SVM concept is to construct a hyperplane that minimizes the upper bound on the out-of-sample error. The critical step of SVM is to transform (or map) data points on to a high dimensional space, known as *kernel transformation*, and classify data points by a separating plane [9]. Subsequently, the hybrid linear programming discriminant model is proposed by [12,13,20]. The hybrid model does not depend on data transformation, where the objective is to find a plane that minimizes violations and maximizes satisfactions of the classified groups. Glover [19] proposed a mixed integer programming (MIP) formulation for the hybrid model by adding binary variables for misclassified entities. Other MIP formulations that are subsequently developed include [1,15,16]. Recently, a new technique that use multiple hyperplanes for classification has been proposed by [17]. This technique constructs a piecewise-linear model that gives convex separating planes. Subsequently, Better, Glover and Samorani [6] proposed multi-hyperplane formulations that generate multiple linear hyperplanes simultaneously with the consequence of forming a binary decision tree.

In classification, the selection of data's features/attributes is also very critical. Many mathematical programming methods have been proposed for selecting well represented features/attributes. Bennett and Mangasarian [5,23] gives a feature selection formulation such that the model not only separates entities into two groups, but also tries to suppress nonsignificant features. In a more recent study, Chaovalitwongse et al. (2006) proposed Support Feature Machine (SFM) formulations can be used to find a set of features that gives the highest classification performance [10].

Baysian decision method has also been widely studied in classification. However, there are only few studies incorporating the Baysian model with mathematical programming approaches. Among those studies, Asparouhov and Danchev [4] formulates a MIP model with binary variables, which are conformed with the Bayesian decision theory. In the case of multi-group classification, Anderson [2] developed a mathematical formulation that incorporates the population densities and prior probabilities of training data. This model yields classification rules for multi-groups with a reject option, (a set having entities that does not belong to any group) [22].

Deterministic Optimization Models

Support Vector Machines

Support Vector Machines (SVM) is aimed at finding a hyperplane that separates the labeled input data into two groups, G_1 and G_2. Then the optimal plane can be used for classifying new data point. The hyperplane can be mathematically expressed by $A\omega = \gamma$, where $\omega \in \Re^m$ is an m-dimensional vector of real numbers, m is total number of attributes/features used to represent a data entity, and $\gamma \in \Re^n$ is a scalar vector. All elements from G_1 and G_2 will be separated by this hyperplane under the assumption that the sets G_1 and G_2 are separable. Define margin as the minimum distance from the plane to elements in a group, G_1 or G_2. The objective function of SVM is to find a separating hyperplane with the largest margin. The data set G_1 can be represented by the matrix $A_i \in \Re^{k\times m}$, $i \in G_1$ and the set G_2 can be represented by the matrix $A_j \in \Re^{(n-k)\times m}$, $j \in G_2$, where k are number of data points (entities) of group G_1. Two open half spaces defined by the hyperplane are $\{A_i\omega < \gamma\}$ and $\{A_j\omega > \gamma\}$. One contains elements of G_1 and the other contains elements of G_2. Therefore, a linear programming (LP) problem can be formulated to determine the optimal values of vectors ω and γ. To construct valid inequalities for linear programming, we rescale the variables (ω, γ), by dividing them by the positive value $\min_{i\in G_1, j\in G_2}\{A_i\omega-\gamma, -A_j\omega+\gamma\}$. Let e denote a vector of ones, and the resulting inequalities become

$$A_i\omega \geq e\gamma + e, \quad A_j\omega \leq e\gamma - e\,. \tag{1}$$

The performance of the SVM relies heavily on the kernel transformation, the data mapping to a high dimension. SVM can also incorporates nonlinear mapping $\phi(\cdot)$. If the new dimension is sufficiently high enough, the data from two classes can always be separated by a hyperplane [9,11]. Examples of SVM kernel functions include linear, polynomial, radial basis function (RBF) and sigmoid. Recently, Shimodaira et al. [24] has proposed the Dynamic Time-Alignment Kernel for time series data.

Robust LP for SVM

It is important to note that the above LP model assumes that A_i and A_j are perfectly separable, which is usually not a case in practice. In other words, it is possible that the inequalities in Eq. (1) provide no solution as the data are not perfectly separable. Bennett and Mangasarian [5] proposed an improved formulation that minimizes an average misclassifications given by

$$\begin{aligned}
&\min_{\omega,\gamma,y,z} \frac{e^T y}{m} + \frac{e^T z}{k}\\
&\text{s.t. } A_i\omega - e\gamma - e \geq y,\\
&\quad -A_j\omega + e\gamma - e \geq z,\\
&\quad y \geq 0, z \geq 0\,.
\end{aligned}$$

It is easy to see that the variables y and z are, in fact, the vectors representing the violations of inequalities in Eq. (2) and minimizing the objective function would lead to the minimum average violation.

Feature Selection with SVM

We note that an extension of the robust LP formulation can be used for feature selection [5,23]. A new term is added in the objective function in the robust LP model to suppress the components of ω. This would try to eliminate all unnecessary features. Let v denote the absolute value of the weight vector ω, log is the base of the natural logarithm, and $\lambda \in (0, 1)$. The mathematical program with a concave objective function and linear constraints for feature selection is given by

$$\begin{aligned}
\min_{\omega,\gamma,y,z,v}\quad & (1-\lambda)\left(\tfrac{e^T y}{m} + \tfrac{e^T z}{k}\right) + \lambda e^T(e - \log^{-\alpha v})\\
\text{s.t.}\quad & A_i\omega - e\gamma - e \geq y,\\
& -A_j\omega + e\gamma - e \geq z,\\
& y \geq 0, z \geq 0,\\
& -v \leq \omega \leq v\,.
\end{aligned}$$

Note that when $\lambda = 0$, the model gives a plane that separates A_i and A_j without considering feature suppression. On the other hand, when $\lambda > 0$, the objective not only tries to separate A_i and A_j, but also tries to eliminate as many of ω components as possible. Specifically, for each v_i $(i = 1, \ldots, n)$, we minimize an exponential smoothing of the step function $(1 - \log^{-\alpha v_i})$. This step function enables the deletion of irrelevant components of ω. There also exists a finitely-terminating algorithm that solves this problem using successive linear programming [8].

Hybrid LP Discriminant Model

A hybrid LP discriminant model is proposed in [12,13,20]. This model is guaranteed to give the optimal solution regardless of the nature of the data. Therefore, the solution is invariant to transformations. This model is improved to overcome many shortcomings of contemporary linear discriminant formulations, which are reviewed and discussed in [21]. Recall that m is the number of attributes, all data points in G_1 are represented by an $k \times m$ matrix A_i, $i \in G_1$, and all data points in G_1 are represented by an $(n-k) \times m$ matrix A_j, $j \in G_2$. For the simplicity of mathematical representation, the membership in G_1 or G_2 can be represented by $i \in G_1$ or $i \in G_2$, respectively. This model will give a hyperplane of the form $A^T\omega = \gamma$, where the model seeks for the optimal weight vector ω, and a scalar γ, where data points of Group 1 lie on one side of the hyperplane and data points of Group 2 lie on the other side (i. e., $A_i\omega < \gamma, i \in G_1$ and $A_i\omega > \gamma, i \in G_2$). Let y_i and z_i represent external and internal deviation variables referring to the point violations and satisfactions of the classification rule. More specifically, they are the magnitudes of the data points lying outside or inside their targeted half spaces. The objective is to minimize violations and maximize the satisfactions of the classified groups. Thus, in the objective function, variable h_i's discourage external deviations and variable k_i's encourage internal deviations. Then $h_i \le k_i$ for $i = 0$ and $i \in G$, must be satisfied. The hybrid model is given by

$$
\begin{aligned}
\min\ & h_0 y_0 + \sum_{i\in G} h_i y_i - k_0 z_0 - \sum_{i\in G} k_i z_i \\
\text{s.t. } & \mathbf{A}_i\omega - y_0 - y_i + z_0 + z_i = \gamma, \quad i \in G_1 \\
& \mathbf{A}_i\omega + y_0 + y_i - z_0 - z_i = \gamma, \quad i \in G_2 \\
& z_0 + \sum_i z_i = 1, \quad i \in G \\
& y_0, z_0 \ge 0 \\
& y_i, z_i \ge 0, \quad i \in G \\
& \omega, \gamma \quad \text{unrestricted.}
\end{aligned}
\tag{2}
$$

We note that Eq. (2) is a normalization constraint that is necessary for avoiding a trivial solution where all $\omega_j = 0$ and $\gamma = 0$. Glover [18] identifies more normalization methods to conquer the problem with null weighting.

MIP Discriminant Model

There are several related mixed integer formulations in the literature [1,15,16]. In general, due to the computational requirements, these standard MIP formulations can only be applied to classification problems with a relatively small number of observations. Glover [19] proposed a compact mathematical program for discriminant model, which is a variant of the above-mentioned hybrid LP model. This objective of this model is to minimize the number of misclassified entities. The MIP discriminant model is given by

$$
\begin{aligned}
\min\ & \sum_{i\in G} z_i \\
\text{s.t. } & A_i x - M z_i + \beta_i = b, \quad i \in G_1 \\
& A_i x + M z_i - \beta_i = b, \quad i \in G_2 \\
& \beta_i \ge 0, \quad i \in G \\
& z_i \in \{0,1\}, \quad i \in G \\
& x, b \quad \text{unrestricted,}
\end{aligned}
$$

where β_i are slack variables, and M is a large constant chosen so that when $z_i = 1$, $A_i x \le b + M z_i$ will be redundant for $i \in G_1$ and $A_i x \ge b - M z_i$ will be redundant for $i \in G_2$. This model can incorporate a normalization constraint, $(-n_2 \sum_{i\in G_1} A_i + n_1 \sum_{i\in G_2} A_i)x = 1$, where n_1 and n_2 are the number of entities in G_1 and G_2, respectively.

Multi-hyperplane Classification

Multi-hyperplane formulations, given by Better et al. [6], generate multiple linear hyperplanes simultaneously with the consequence of forming a decision tree. The hyperplanes are generated from an extension of the Discriminant Model proposed by Glover [18]. Instead of using kernel transformation that projects data into a high dimensional space to improve the performance of SVM, the multi-hyperplane approach approximates a nonlinear separation by constructing multiple hyperplanes. Let $d = 0$ when we are at a root node of a binary tree, where none of the classifications have been done. Let $d = D$ when the tree has two leaf nodes corresponding to the final separation step. In order to explain the model, we define the following terms.

- Successive Perfect Separation (SPS) is a procedure that forces all elements of Group 1 (G_1) and Group 2 (G_2) to lie on one side of the hyperplane at each node

for any depth $d \in \{0, \ldots, D-1\}$. SPS is a special use of a variant based on a proposal of Glover [18].

- SPS decision tree is a tree that results from the two-group classification iteratively applying the SPS procedure. The root node ($d = 0$) contains all the entities in the data set, and at $d = D$ the two leaf nodes correspond to the final separation step.

For a given maximum depth D, an initial multi-hyperplane model considers each possible SPS tree type of depth d, for $d = 0, \ldots, D-1$. A root node is viewed as a "problem" node where all data points from both groups need to be separated. A leaf node, on the other hand, is considered to be a "decision" node where data points are classified into two groups. Define slicing variables sl_i for $i \in \{1, \ldots, D-1\}$. There are total of $D-1$ slicing variables needed for a tree having maximum depth D. Specifically, at depth $d = 1$, $sl_1 = 0$ if the "left" node constitutes a leaf node while the "right" node constitutes a root (or problem) node. Without loss of generality, we herein consider $D = 3$ for the initial multi-hyperplane model. The mathematical model for multi-hyperplane SVM can be formally defined as follows.

Let M and ε denote large and small positive constants, respectively, and G denote a set of the union of entities in G_1 and G_2. Suppose there are n entities in the training data set. Define a binary variable $z_i^* = 0$ if object i is correctly classified by the "tree", otherwise $z_i^* = 1$. Define a binary variable and $z_{hi}^* = 0$ if object i is correctly classified by "hyperplane h", otherwise $z_{hi}^* = 1$. The multi-hyperplane SVM model also includes traditional hyperplane constraints for each depth d of the tree and the normalization constraint, which is similar to the mixed integer programming model in [18]. Then, ε is added to prevent data points from lying on the hyperplane. Tree-type constraints are included to identify the optimal tree structure for the data set, which will be in part of the optimal classification rule. Binary variables y_i are used for tree types (0,1) and (1,0) to activate or deactivate either-or constraints. The SPS decision tree formulation for the depth $D = 3$ is given by

$$\min \sum_{i=1}^{n} z_i^*$$

$$\text{s.t.} A_i x_d - M z_{di} + \beta_i = b_d - \varepsilon \quad i \in G_1,\ d = 1, 2, 3 \tag{3}$$

$$A_i x_d + M z_{di} - \beta_i = b_d + \varepsilon \quad i \in G_2,\ d = 1, 2, 3 \tag{4}$$

$$M(sl_1 + sl_2) + z_i^* \geq z_{1i} + z_{2i} + z_{3i} - 2 \quad i \in G_1 \tag{5}$$

$$M(sl_1 + sl_2) + M z_i^* \geq z_{1i} + z_{2i} + z_{3i} \quad i \in G_2 \tag{6}$$

$$M(2 - sl_1 - sl_2) + M z_i^* \geq z_{1i} + z_{2i} + z_{3i} \quad i \in G_1 \tag{7}$$

$$M(2 - sl_1 - sl_2) + z_i^* \geq z_{1i} + z_{2i} + z_{3i} - 2 \quad i \in G_2 \tag{8}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{1i} - M y_i \quad i \in G_1 \tag{9}$$

$$M(1 + sl_1 - sl_2) + M z_i^* \geq z_{2i} + z_{3i} - M[1 - y_i] \quad i \in G_1 \tag{10}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{1i} \quad i \in G_2 \tag{11}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{2i} + z_{3i} - 1 \quad i \in G_2 \tag{12}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{1i} \quad i \in G_1 \tag{13}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{2i} + z_{3i} - 1 \quad i \in G_1 \tag{14}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{1i} - M y_i \quad i \in G_2 \tag{15}$$

$$M(1 + sl_1 - sl_2) + M z_i^* \geq z_{2i} + z_{3i} - M[1 - y_i] \quad i \in G_2 \tag{16}$$

$$\sum_{j=1}^{n}\sum_{d=1}^{3} x_{jd} = 1 \tag{17}$$

$$z_i^* \in \{0,1\},\ z_{di} \in \{0,1\},\ y_i \in \{0,1\},$$
$$i \in G, d = 1,2,3$$
$$sl_k \in \{0,1\}\ k = 1,2,\ x, b \text{ unrestricted},$$

where the constraints in Eqs. (3)-(4) are the hyperplane constraints, Eqs. (5)-(6) are the constraints for tree type (0,0), Eqs. (7)-(8) are the constraints for tree type (1,1), Eqs. (9)-(12) are the constraints for tree type (0,1), Eqs. (13)-(16) are the constraints for tree type (1,0), and Eq. (17) is the normalization constraint. This small model with $D = 3$ performs well for small depths and has computational limitations. The reader should refer to [6] for a greater detail of an improved and generalized structure model for all types of SPS trees.

Support Feature Machines

Support Feature Machines (SFM) proposed in [10] is a mathematical programming technique used to identify a set of features that gives the highest performance in classification using the nearest neighbor rule. SFM can be formally defined as follows. Assume there are n data points, each with m features, we define the decision variables $x_j \in \{0,1\}\,(j = 1,\ldots,m)$ indicating if feature j is selected by SFM and $y_i \in \{0,1\}\,(i = 1,\ldots,n)$ indicating if sample i can be correctly classified by SFM. There are two versions of SFM, *voting* and *averaging*. Each version uses different weight matrices, which are provided by user's classification rule.

The objective function of voting SFM is to maximize the total correct classification as in Eq. (18). There are two sets of constraints used to ensure that the training samples are classified based on the voting nearest neighbor rule as in Eqs. (19)-(20). There is a set of logical constraints in Eq. (21) used to ensure that at least one feature is used in the voting nearest neighbor rule. The mixed-integer program for voting SFM is given by:

$$\max \sum_{i=1}^{n} y_i \tag{18}$$
$$\text{s.t.} \sum_{j=1}^{m} a_{ij}x_j - \sum_{j=1}^{m} \frac{x_j}{2} \le M y_i$$
$$\text{for } i = 1,\ldots,n \tag{19}$$
$$\sum_{j=1}^{m} \frac{x_j}{2} - \sum_{j=1}^{m} a_{ij}x_j + \epsilon \le M(1-y_i)$$
$$\text{for } i = 1,\ldots,n \tag{20}$$
$$\sum_{j=1}^{m} x_j \ge 1 \tag{21}$$
$$x \in \{0,1\}^m,\ \ y \in \{0,1\}^n,$$

where $a_{ij} = 1$ if the nearest neighbor rule correctly classified sample i at electrode j, 0 otherwise, n is total number of training samples, m is total number of features, $M = m/2$, and ϵ is a small positive number used to break a tie during the voting ($0 < \epsilon < 1/2$).

The objective function of averaging SFM is to maximize the total correct classification as in Eq. (22). There are two sets of constraints used to ensure that the training samples are classified based on the distance averaging nearest neighbor rule as in Eqs. (23)-(24). There is a set of logical constraints in Eq. (25) used to ensure that at least one feature is used in the distance averaging nearest neighbor rule. The mixed-integer program for averaging SFM is given by:

$$\max \sum_{i=1}^{n} y_i \tag{22}$$
$$\text{s.t.} \sum_{j=1}^{m} \bar{d}_{ij}x_j - \sum_{j=1}^{m} d_{ij}x_j \le M_{1i} y_i$$
$$\text{for } i = 1,\ldots,n \tag{23}$$
$$\sum_{j=1}^{m} d_{ij}x_j - \sum_{j=1}^{m} \bar{d}_{ij}x_j \le M_{2i}(1-y_i)$$
$$\text{for } i = 1,\ldots,n \tag{24}$$
$$\sum_{j=1}^{m} x_j \ge 1 \tag{25}$$
$$x \in \{0,1\}^m,\ \ y \in \{0,1\}^n,$$

where d_{ij} is the average statistical distance between sample i and all other samples from the same class at feature j (*intra-class distance*), $\bar{d}_{ij}$ is the average statistical distance between sample i and all other samples from different class at feature j (*inter-class distance*), $M_{1i} = \sum_{j=1}^{m} \overline{d_{ij}}$, and $M_{2i} = \sum_{j=1}^{m} d_{ij}$.

Probabilistic Optimization Models

The deterministic classification models in the previous section make a strong assumption that the data are separable. In the case that the data may not be well separated, using the deterministic models may lead to a high misclassification rate. The classification models that incorporate probabilities may be a better option for such noisy data. When the population densities and prior probabilities are known, there are probabilistic models that consider constrained rules with a reject option [2] as well as a Baysian-based model [4].

Bayesian-Based Mathematical Program

The Baysian-based mathematical program that are conformed with the Bayesian decision theoretic approach is proposed by Asparouhov and Danchev [4]. The model can be formally defined as follows. Denote $c \in \Re$ as a cut-off value, $x \in B^m$ as a vector of m binary values, and $\omega \in \Re^m$ is a decision variable having m-dimensional vector of real numbers. A preprocessing needs to be performed so that if $x^T\omega \le c$, the entity x belong to class 1; otherwise it belongs to class 2. Suppose we have a set of n data points, n_1 data points are in G_1 and n_2 data points are in G_2, $(n = n_1 + n_2)$. Let s be a non empty multinomial cell. Denote n_{is} as the number of design set observation from the class i, where $i = 1, 2$, falling in this cell s. There are 2^m number of multinomial cells. Each cell is unique and all observations that belongs to it have exactly the same values of the m binary variables. Denote M as a sufficiently large positive real number, and ϵ as a small positive number. In addition to having a geometric interpretation, this formulation is inspired from Bayesian decision theoretic approach and having prior probabilities, $n_i/n > \forall i$, incorporated. Experimental studies in [4] suggest this Baysian-based model can give better performance than other contemporary linear discriminant models. The Baysian-based classification formulation is given by

$$\begin{aligned}
\min_{\omega, z_s, c} \ & \sum_s \left(|n_{1s} - n_{2s}|\, z_s + \min(n_{1s}, n_{2s})\right) \\
\text{s.t. } & \mathbf{x}_s^T\omega - Mz_s \le c \text{ if } n_{1s} \ge n_{2s} \\
& \mathbf{x}_s^T\omega + Mz_s \ge c + \varepsilon \text{ if } n_{1s} < n_{2s} \\
& n_{1s} + n_{2s} \ne 0 \\
& z_s \in \{0, 1\},\ \omega \in \Re^m,\ c \in \Re\,.
\end{aligned}$$

Probabilistic Models for Classification

An optimization model proposed by Anderson [2] incorporates population densities, prior probabilities from all groups, and misclassification probabilities. This method is aimed to find a partition $\{R_0, R_1, \ldots, R_G\}$ of Re^m where m is the number of features. This method naturally forms a multi-group classification. The objective is to maximize the probability of correct allocation subject to constraints on the misclassification probabilities. The mathematical model can be formally defined as follows. Let f_h, $h = 1, \ldots, G$, denote the group conditional density functions. Let π_g denote the prior probability that a randomly selected entity is from group g, $g = 1, \ldots, G$, and α_{hg}, $h \ne g$, are constants between 0 and 1. The probabilistic classification model is then given by

$$\begin{aligned}
\min \ & \sum_{g=1}^{G} \pi_g \int_{R_g} f_g(w)\, \mathrm{d}w \\
\text{s.t. } & \int_{R_g} f_h(w)\, \mathrm{d}w \le \alpha_{hg} \\
& \text{for } h, g = 1, \ldots, G,\ h \ne g\,.
\end{aligned}$$

The optimal rule that can be used as a classification method is given by

$$R_g = \left\{ x \in \Re^k : L_g(x) = \max_{h \in 0,1,\ldots,G} L_h(x) \right\}, \quad (26)$$

where $g = 0, \ldots, G$, $L_0(x) = 0$, and $L_h(x) = \pi_h f_h(x) - \sum_{i=1, i\ne h}^{G} \lambda_{ih} f_i(x)$, for $h = 1, \ldots, G$. In general, there exist nonnegative constants $\lambda_{ih}, i, h \in 1, \ldots, G, i \ne h$, such that this optimal rule holds. The procedure for deriving a discriminant rule is composed of two stages. The first stage is to compute $\hat{f}_h$, which are estimated density functions f_h, and $\hat{\pi}_h$, which are estimated prior probabilities π_h, for $h = 1, \ldots, G$. There are many methods proposed for density estimation. The second stage is to estimate the optimal $\lambda'_{jh}s$, given the estimates $\hat{f}'_h s$ and $\hat{\pi}'_h s$. For estimating the $\lambda'_{jh}s$, there is a MIP approach proposed in [14], and a LP approach proposed in [22].

The MIP approach uses binary variables to record whether each entity was allocated to each region. This approach measures the probabilities of correct classification and misclassification for any candidate set of

$\lambda'_{ih}s$, which are calculated as the proportion of training samples that fall into each of the regions. The objective is to maximize a linear combination of variables representing correct allocation. The proportions of training samples misclassified were incorporated in constraints on misclassification probabilities. On the other hand, the LP approach does not have binary variables to incorporate proportions of misclassified training data points, and to provide a mechanism for modeling a priori bounds on misclassification probabilities. Instead, the LP approach provides a mechanism for estimating $\lambda'_{ih}s$ that balances the minimization of misclassifications and the maximization of correct classifications. This can be demonstrated as follows. Redefine the function $L_h, h = 1, \ldots, G$, as

$$L_h(x) = \pi_h p_h(x) - \sum_{i=1, i\neq h}^{G} \lambda_{ih} p_i(x)\,, \tag{27}$$

where $p_i(x) = f_i(x)/\sum_{t=1}^{G} f_t(x)$. This is analogous to the definition of original p_i in Eq. (26) since R_g can be expressed as $R_g = \{x \in \Re^k : L_g(x) \leq L_h(x),\ h = 0, \ldots, G\}$, if and only if, $\left(\left(1/\sum_{t=1}^{G}\right) f_t(x)\right) L_g(x) \leq \left(\left(1/\sum_{t=1}^{G}\right) f_t(x)\right) L_h(x)$. Note that this new definition of L_h is just an assumption. In addition, we also assume that we have a training sample of n data points whose group classifications are known. There are n_g data points in group g and $\sum_{g=1}^{G} n_g = n$. For notational convenience, let $\gamma = 1, \ldots, G$ and $N_g = 1, \ldots, n_g$. Each data point x has k attributes, denoted as $x^{gj} \in \Re^k$ for $g = 1, \ldots, G$ and $j = 1, \ldots, n_g$.

MIP Formulation for Anderson's Model

In order to find the optimal estimation of the second stage for solving Anderson's formula in Eq. (27), after the estimates $\hat{f}'_h s$ and $\hat{\pi}'_h s$ are given, the optimal $\lambda'_{jh}s$ is the final goal. For estimating the $\lambda'_{jh}s$, Gallagher et al. [14] proposed a MIP formulation. Same notation used in Anderson's formula in last section is applied here. The model ensures that the proportion of training data points and total data points n_g of group g in region R_h is less than or equal to a pre-specified percentage, $\alpha_{hg} > (0 < \alpha_{hg} < 1)$, for $h, g \in \gamma$ and $h \neq g$. The original formulation of the approach is a nonlinear MIP model given by

$$\min_{L_{hgj}, y_{gj}, \lambda_{ih}, u_{gj}} \sum_{g\in\gamma} \sum_{j\in N_g} u_{ggj}$$

s.t.

$$L_{hgj} = \pi_h \hat{p}_h(x^{gj}) - \sum_{i\in\gamma\setminus\{h\}} \lambda_{ih} \hat{p}_i(x^{gj}) \tag{28}$$

for $h, g \in \gamma, j \in N_g$

$$y_{gj} = \max\left\{0, L_{hgj} : h = 1, \ldots, G\right\} \tag{29}$$

for $g \in \gamma, j \in N_g$

$$y_{gj} - L_{ggj} \leq M\left(1 - u_{ggj}\right) \tag{30}$$

for $g \in \gamma, j \in N_g$

$$y_{gj} - L_{hgj} \geq \varepsilon(1 - u_{hgj}) \tag{31}$$

for $h, g \in \gamma, h \neq g, j \in N_g$

$$\sum_{j\in N_g} u_{hgj} \leq \lfloor \alpha_{hg} n_g \rfloor \tag{32}$$

for $h, g \in \gamma, h \neq g$

$$-\infty < L_{hgj} < \infty \quad \text{for} \quad h, g \in \gamma, j \in N_g$$

$$y_{gj} \geq 0 \quad \text{for} \quad g \in \gamma, j \in N_g$$

$$\lambda_{ih} \geq 0 \quad \text{for} \quad i \in N_g, h \in \gamma$$

$$u_{gj} \in \{0, 1\} \quad \text{for} \quad g \in \gamma, j \in N_g$$

The above nonlinear mixed integer programming model can be transformed to an equivalent linear mixed integer model. The transformation is made by replacing the constraint in Eq. (29) with the following constraints:

$$\begin{aligned}
&y_{gj} \geq L_{hgj} \ \ h, g \in \gamma,\ j \in N_g \\
&\tilde{y}_{hgj} - L_{hgj} \leq M(1 - v_{ghj}) \ \ h, g \in \gamma,\ j \in N_g \\
&\tilde{y}_{hgj} \leq \pi_h \hat{p}_h(x^{gj}) v_{hgj} \ \ h, g \in \gamma,\ j \in N_g \\
&\sum_{h\in G} v_{hgj} \leq 1 \ \ g \in \gamma,\ j \in N_g \\
&\sum_{h\in G} \tilde{y}_{hgj} = y_{gj} \ \ g \in \gamma,\ j \in N_g\,,
\end{aligned}$$

where $\tilde{y}_{hgj} \geq 0$ and $v_{ghj} \in \{0, 1\}$, for $h, g \in \gamma,\ j \in N_g$. The constraints in Eq. (28) define the decision variable L_{hgj} as a function value of L_h at x^{gj}. The variable y_{gj} in Eq. (29) gives a result that x^{gj} lies in region R_h, if and only if, $y_{gj} = L_{hgj}$. The binary variable u_{hgj} is used to indicate whether or not x^{gj} lies in region R_h. The constraints in Eq. (30) together with the objective function ensure that $u_{ggj} = 1$, if and only if, the jth entity from group g is correctly allocated to group g. The constraints in Eqs. (31)-(32) ensure that at most $\lfloor \alpha_{hg} n_g \rfloor$ data points of group g are allocated to group h, $h \neq g$. Note that the condition of indicator variables, $u_{hgj} = 0$, $h \neq g$, implies that $x^{gj} \notin R_h$ by Eq. (31), but the converse need not hold. As a result, the number of misclassifications may be overcounted. To force the converse hold, (that is $u_{hgj} = 1$, if and only if, $x^{gj} \in R_h, \forall h, g \in \gamma$), one can include the following constraints: $y_{gj} - L_{hgj} \leq M\left(1 - u_{hgj}\right)$ for $h, g \in \gamma, j \in N_g$. However, the addition of such constraints substantially increases the solution times and the actual amount of overcounting is minimal. M and ε are large and small positive constants, respectively. Since this MIP formulation is very difficult to solve, especially it involves $2GN$ binary variables. There is a preprocessing strategy suggested in [14] by aggregating variables and constraints. Special branching strategies for solving the MIP model is also suggested in [14]. Those strategies include branching on the smallest indexed fractional-valued binary variable, branching on the most infeasible fractional-valued binary variable, pseudo reduced-cost branching schemes, and strong branching [3,7].

LP Formulation for Anderson's Model

In order to estimate the $\lambda'_{jh}s$ for solving Anderson's formula in the second stage in Eq. (27), Lee et al. [22] proposed the Linear Programming (LP) model that minimizes a penalty function in order to allocate each training entity to its correct group or to the reserved-judgment region. Note that same notation used in the MIP approach and Anderson's formula is consistent here. The method is given by

$$\min_{L_{hgj}, \omega_{gj}, y_{gj}, \lambda_{ih}} \sum_{g \in \gamma} \sum_{j \in N_g} \left(c_1 \omega_{gj} + c_2 y_{gj}\right)$$

s.t.

$$L_{hgj} = \pi_h \hat{p}_h(x^{gj}) - \sum_{i \in \gamma \setminus \{h\}} \lambda_{ih} \hat{p}_i(x^{gj}) \tag{33}$$

$$\text{for} \quad h, g \in \gamma, j \in N_g$$

$$L_{ggj} - L_{hgj} + \omega_{gj} \geq 0 \tag{34}$$

$$\text{for} \quad h, g \in \gamma, h \neq g, j \in N_g$$

$$L_{ggj} + \omega_{gj} \geq 0 \tag{35}$$

$$\text{for} \quad g \in \gamma, j \in N_g$$

$$-L_{hgj} + y_{gj} \geq 0 \tag{36}$$

$$\text{for} \quad h, g \in \gamma, j \in N_g$$

$$-\infty < L_{hgj} < \infty \quad \text{for} \quad h, g \in \gamma, j \in N_g$$

$$\omega_{gj} \geq 0 \quad \text{for} \quad g \in \gamma, j \in N_g$$

$$y_{gj} \geq 0 \quad \text{for} \quad g \in \gamma, j \in N_g$$

$$\lambda_{ih} \geq 0 \quad \text{for} \quad i \in N_g, h \in \gamma\ .$$

The constraints in Eq. (33) define the decision variable L_{hgj} as a function value of L_h for x^{gj}. If the optimal solution yields $\omega_{gj} = 0$, for some (g, j) pair, the constraints in Eqs. (34)-(35) imply that $L_{ggj} = \max\{0, L_{hgj} : h \in \gamma\}$. Thus, when $\omega_{gj} = 0$, it means that the jth entity from group g is correctly classified. If $y_{gj} = 0$ is the case for some (g, j) pair, then the constraints in Eq. (36) implies that $L_{ggj} = \max\{0, L_{hgj} : h \in \gamma\} = 0$. Hence, the jth entity from group g is placed in the reserved-judgment region. If both ω_{gj} and y_{gj} are positive, the jth entity from group g is misclassified. The optimization solver is attempting either to correctly classify training data points ($\omega_{gj} = 0$), or to place them in the reserved-judgment region ($y_{gj} = 0$). The optimizer's emphasis can be realized by varying the weights c_1 and c_2. It is possible for both ω_{gj} and y_{gj} to be zero. One should decide how to interpret in such situation. Recall the optimal rule in Eq. (26), which constrains that if x belongs to the reserved judgment region ($h = 0$) then it gives the function value $L_0(x) = 0$.

References

1. Abad PL, Banks WJ (1993) New LP based heuristics for the classification problem. Eur J Oper Res 67:88–100
2. Anderson JA (1969) Constrained discrimination between k populations. J Royal Stat Soc Series B 31:123–139
3. Applegate D, Bixby RE, Chvatal V, Cook W (1994) The traveling salesman problem. Technical Report, Dept Comput Appl Math, Rice University, Houston, TX
4. Asparouhov O, Danchev S (1997) Discrimination and classification in the presence of binary variables. Biocybern Biomed Eng 17(1–2):25–39
5. Bennett KP, Mangasarian OL (1992) Robust linear programming discrimination of two linearly inseparable sets. Optim Meth Soft 1:23–34
6. Better M, Glover F, Samorani M (2006) Multi-Hyperplane Formulations for Classification and Discrimination Analysis. submitted for the Student Paper Award of the Decision Analysis Society, working paper, University of Colorado, Boulder, Colorado
7. Bixby RE, Cook W, Cox A, Lee EK (1995) Computational experience with parallel mixed integer programming in a distributed environment. Research Monograph CRPC-TR95554, Center for Research on Parallel Computation, Rice University, Houston, TX
8. Bradley PS, Mangasarian OL, Street WN (1998) Feature selection via mathematical programming. INFORMS J Comput 10:209–217
9. Burges C (1998) Tutorial on Support Vector Machines for Pattern Recognition. Data Min Know Discov 2:121–167
10. Chaovalitwongse WA, Fan YJ, Sachdeo RC (2006) Novel Optimization Models for Multidimensional Time Series Classification: Application to the Identification of Abnormal Brain Activity. Submitted to Oper Res
11. Duda RO, Hart PE, Stork DG Pattern Classification, 2nd edn. Wiley-Interscience, New York
12. Freed E, Glover F (1981) Simple but powerful goal programming models for discriminant problems. Eur J Oper Res 7(1):44–60
13. Freed E, Glover F (1986) Resolving certain difficulties and improving the classification power of the LP discriminant analysis procedure. Decis Sci 17:589–595
14. Gallagher RJ, Lee EK, Patterson DA (1997) Constrained discriminant analysis via 0/1 mixed integer programming. Annals Oper Res 74:65–88
15. Glen JJ (1999) Integer programming methods for normalization and variable selection in mathematical programming discriminant analysis models. J Oper Res Soc 50: 1043–1053
16. Glen JJ (2003) An iterative mixed integer programming method for classification accuracy maximizing discriminant analysis. Comput Oper Res 30:181–198
17. Glen JJ (2005) Mathematical programmming models for piecewise-linear discriminant analysis. J Oper Res Soc 56:331–341
18. Glover F (1990) Improved Linear Programming Models for Discriminant Analysis. Decis Sci 21(4):771–785
19. Glover F (1993) Improved Linear and Integer Programming Models for Discriminant Analysis. Creative and Innovative Approaches to the Science of Management. RGK Foundation Press, Austin, pp 187–215
20. Glover F, Keene S, Duea B (1988) A new class of models for the discriminant problem. Decis Sci 19:269–280
21. Joachimsthaler EA, Stam A (1990) Mathematical Programming Approaches for the Classification Problem in Two-Group Discriminant Analysis. Multi Behav Res 25(4):427–454
22. Lee EK, Gallagher RJ, Patterson DA (2003) A Linear Programming Approach to Discriminant Analysis with a Reserved-Judgment Region. INFORMS J Comput 15(1): 23–41
23. Mangasarian OL (1997) Mathematical Programming in Data Mining. Data Min Know Discov 1:183–201
24. Shimodaira H, Noma KI, Nakai M, Sagayama S (2001) Dynamic Time-Alignment Kernel in Support Vector Machine. Adv Neural Inf Process Sys 14(2):921–928

Differential Equations and Global Optimization

Maurizio Bruglieri[1], Pierluigi Maponi[1], Maria Cristina Recchioni[2], Francesco Zirilli[3]

[1] Dip. Mat. e Fisica, University Camerino, Camerino, Italy

[2] Ist. Mat. e Stat., University Ancona, Ancona, Italy

[3] Dip. Mat. G. Castelnuovo, University Roma La Sapienza, Roma, Italy

MSC2000: 60G35, 65K05

Article Outline

Keywords
Mathematical Background
Global Unconstrained Optimization
Global Constrained Optimization
Miscellaneous Results
  Clique Problem
  Quasivariational Inequalities
See also
References

Keywords

Global optimization; Stochastic differential equation

Let $\mathbf{N}$ be the set of natural numbers. Let $N \in \mathbf{N}$ and $\mathbf{R}^N$ be the N-dimensional real Euclidean space, let $\mathbf{C}^N$ be the N-dimensional complex Euclidean space, $\mathbf{R}$, $\mathbf{C}$ are used in place of $\mathbf{R}^1$, $\mathbf{C}^1$ respectively. Let $\mathbf{R}^+ = \{x \in \mathbf{R}: x \geq 0\}$. Let $x \in \mathbf{R}$; then $|x|$ denotes the absolute value of x. Let $\mathbf{x}, \mathbf{y} \in \mathbf{R}^N$; then $(\mathbf{x}, \mathbf{y})$ denotes the scalar product in $\mathbf{R}^N$ and $\| \mathbf{x} \|$ denotes the euclidean norm in $\mathbf{R}^N$. Let $\mathbf{S}^N = \{\mathbf{x} \in \mathbf{R}^{N+1}: \| \mathbf{x} \| = 1\}$.

Let $D \subseteq \mathbf{R}^N$ be a connected domain, let $F: D \to \mathbf{R}$ be a given function. The following problem is considered:

$$\min_{\mathbf{x} \in D} F(\mathbf{x}). \tag{1}$$

When $D = \mathbf{R}^N$ problem (1) is called the *global unconstrained optimization problem*. When $D \subset \mathbf{R}^N$ it is called the *global constrained optimization problem*.

Without loss of generality one considers only the minimization problem, that is, problem (1), since the maximization problem can be easily reduced to a minimization problem.

To solve problem (1) means to find a point $\mathbf{x}^* \in D$ such that $F(\mathbf{x}^*) \leq F(\mathbf{x})$, $\forall \mathbf{x} \in D$.

A large number of problems with great theoretical and practical interest can be formulated as global optimization problems, that is, as problem (1).

In this article the global optimization problems are studied only from the point of view of numerical optimization and in particular of numerical methods based on differential equations. Many other fruitful points of view are possible to study that include the set of global minimizers of F on D or in general the set of critical points of F on D depending on the hypotheses made on F and D.

A method to solve problem (1) in the sense of numerical optimization is usually an iterative scheme that from a given initial guess $\mathbf{x}^0 \in D$ is able to compute a sequence $\{\mathbf{x}^n \in D: n \in \mathbf{N}\}$ such that $\mathbf{x}^n \to \mathbf{x}^*$ when $n \to \infty$.

Problem (1) can be easily solved in some special cases, that is, when the function F and the domain D have special forms, for example one can recall the following two important cases:

- *linear programming problem*: F linear function, D convex polyhedron, i. e., $D \subset \mathbf{R}^N$ is defined implicitly by means of equalities and inequalities between linear functions;
- *convex programming problem*: F convex function, $D \subset \mathbf{R}^N$ convex region.

One notes that the linear programming problem can be considered as a special case of the convex programming problem. For both cases effective methods to solve problem (1) are known, e. g., for the linear programming problem the simplex method, see [7], and for the convex programming problem the Newton method coupled with some strategy to treat the constraints that define D, for example active set strategy, see [9].

In general, problem (1) is a difficult one since the property of being a global minimizer is not a local property. That is, a global minimizer $\mathbf{x}^*$ cannot be recognized from local properties of the function F at $\mathbf{x}^*$, such as the value of F and its derivatives at $\mathbf{x}^*$. Numerical algorithms to recognize global properties are unusual and in general computationally expensive.

For example, let $D = \mathbf{R}$, $m, \alpha \in \mathbf{R}$, $\delta > 0$, one considers the following two functions:

$$F_1(x) = -\frac{1}{1+x^2},$$

$$F_2(x) = -\frac{1}{1+x^2} + m \begin{cases} e^{\frac{1}{(x-\alpha)^2-\delta^2}} e^{\frac{1}{\delta^2}}, & x \in (\alpha-\delta, \alpha+\delta), \\ 0, & x \notin (\alpha-\delta, \alpha+\delta). \end{cases}$$

Function F_1 has in $x = 0$ the unique local minimizer which is also the global minimizer, i. e. $x^* = 0$. Let $m < -1$, $0 < \delta < |\alpha|$; then function F_2 has several critical points including two local minimizers, one is $x = 0$ and the other is $x = x_2 \in (\alpha - \delta, \alpha + \delta)$. Moreover the global minimizer of F_2 is $x = x^* = x_2$. One notes that F_1, F_2 are smooth functions and that they coincide, for every $x \in \mathbf{R} \setminus (\alpha - \delta, \alpha + \delta)$ where $\delta > 0$ is arbitrary.

Let $D = \mathbf{R}^N$, let $F: D \to \mathbf{R}$ be a continuously differentiable function, let ∇F be the gradient of F, let $\mathbf{x} \in D$ be such that $(\nabla F)(\mathbf{x}) \neq 0$ then the vector $-(\nabla F)(\mathbf{x})$ gives the direction of steepest descent for the function F at the point $\mathbf{x}$. One can consider the following system of differential equations:

$$\frac{d\mathbf{x}}{dt}(t) = -(\nabla F)(\mathbf{x}(t)), \quad t > 0, \tag{2}$$

$$\mathbf{x}(0) = \mathbf{x}^0. \tag{3}$$

Under some hypotheses on F, the solution of problem (2), (3) is a trajectory in $\mathbf{R}^N$ starting from $\mathbf{x}^0$ and

ending in the critical point $\mathbf{x}^*_{\mathrm{loc}}(\mathbf{x}^0)$ of F whose attraction region contains $\mathbf{x}^0$. Using a numerical integration scheme for (2), (3) one can obtain a numerical optimization method, for example choosing the Euler integration scheme with variable stepsize from (2), (3) one obtains the so-called *steepest descent algorithm*. Let $\xi^k \in \mathbf{R}^N$ be the approximation of $\mathbf{x}(t_k)$, $k \in \mathbf{N}$, where $t_0 = 0$, $0 < t_k < t_{k+1} < +\infty$, $k = 1, 2, \ldots$, and $t_k \to +\infty$ when $k \to \infty$, obtained with a numerical optimization method coming from (2), (3). Suppose $\{\boldsymbol{\xi}^k\, k \in \mathbf{N}\}$ is a sufficiently good approximation of the solution $\mathbf{x}(t)$, $t > 0$ of (2), (3) one has $\lim_{k \to \infty} \boldsymbol{\xi}^k = \lim_{t \to +\infty} \mathbf{x}(t) = \mathbf{x}^*_{\mathrm{loc}}(\mathbf{x}^0)$, thus the numerical optimization methods obtained from (2), (3) compute critical points that depend on the initial guess $\mathbf{x}^0$. So that these critical points usually are not global minimizers of F.

One can consider numerical optimization methods due to other differential equations instead of (2), that is differential equations taking in account higher order derivatives of F or of $\mathbf{x}(t)$. However the minimizers computed with these numerical optimization methods depend only on local properties of the function F, thus in general they will not be global minimizers of F. So that methods based on ordinary differential equations are inadequate to deal with problem (1).

In this article it is described how to use stochastic differential equations to avoid this difficulty. In fact one wants to destabilize the trajectories generated by problem (2), (3) using a stochastic perturbation in order to be able to reach global minimizers. This must be an appropriate perturbation, that is the corresponding perturbed trajectories must be able to leave the attraction region of a local minimizer of F to go in an attraction region of another minimizer of F obtaining as $t \to +\infty$ the solution of problem (1). This is done by adding a stochastic term, i. e., a Brownian motion on the right-hand side of equation (2). Moreover this stochastic term takes into account the domain D, when $D \subset \mathbf{R}^N$. This is done introducing the solution of the Skorokhod reflection problem.

In the second section one gives some mathematical background about stochastic differential equations that is necessary to state the results of the third and fourth sections. In the third section, the unconstrained version of problem (1) is treated, i. e., $D = \mathbf{R}^N$. In the fourth section, the constrained version of problem (1) is treated, i. e., $D \subset \mathbf{R}^N$. In both these sections one gives methods, convergence analysis and discussion when possible of a relevant software library. In the last section one gives some information about new application areas of global optimization such as graph theory and game theory.

Mathematical Background

Let $\Omega \subseteq \mathbf{R}$, Σ be a σ-field of subsets of Ω and P be a probability measure on Σ. The triple $(\Omega, \Sigma, \mathsf{P})$ is called a *probability measure space*, see [5] for a detailed introduction to probability theory. Let $\Omega' \subseteq \mathbf{R}$, Υ be a topology of subsets of Ω'. Then $X : \Omega \to \Omega'$ is a random variable if $\{X \in A\} \in \Sigma$ for every $A \in \Upsilon$.

The *distribution function* $G_X : \mathbf{R} \to [0, 1]$ of X is defined by $G_X(x) = \mathsf{P}\{X \leq x\}$, $x \in \mathbf{R}$ and one denotes with g_X its density. The expected value or the mean value of X is defined as follows:

$$m(X) = \int_{\mathbb{R}} x G_X(\, dx) = \int_{\mathbb{R}} x g_X(x)\, dx \tag{4}$$

and the variance of X is given by:

$$v(X) = m((X - m(X))^2). \tag{5}$$

For example, a random variable X has discrete distribution, or is concentrated on $x_1, \ldots, x_n$, when $g_X(x) = \sum_{i=1}^n p_i \delta(x - x_i)$, where $p_i > 0$, $x_i \in \Omega'$, $i = 1, \ldots, n$, $\sum_{i=1}^n p_i = 1$ and δ is the Dirac delta. Given $m \in \mathbf{R}$, $v > 0$ a random variable has normal distribution when

$$g_X(x) = \frac{1}{\sqrt{2\pi v}} e^{-\frac{(x-m)^2}{2v}},$$

one notes that $m(X) = m$ and $v(X) = v$.

A *stochastic process* is a family of random variables depending on a parameter t, that is, $\{X(t)\colon \Omega \to \Omega', t \geq 0\}$. A *Brownian motion* is a stochastic process $\{w(t)\colon t \geq 0\}$ having the following properties:

- $\mathsf{P}\{w(0) = 0\} = 1$;
- for every choice of t_i, $i = 1, \ldots, k$, $0 \leq t_i < t_{i+1} < +\infty$, $i = 1, \ldots, k-1$, the increments $w(t_{i+1}) - w(t_i)$, $i = 1, \ldots, k-1$, are independent and normally distributed random variables with mean value equal to zero and variance equal to $t_{i+1} - t_i$.

An *N-dimensional Brownian motion* is a N-dimensional process

$$\{\mathbf{w}(t) = (w_1(t), \ldots, w_N(t))\colon\ t \geq 0\}$$

where its components $\{w_i(t)\colon \Omega \to \Omega', t \geq 0\}$, $i = 1, \ldots, N$, are independent Brownian motions. The Brownian

motion is a good mathematical model to describe phenomena that are the superposition of a large number of chaotic elementary independent events. The most famous example of Brownian motion is the motion of pollen grains immersed in a fluid, the grains have a chaotic perpetual motion due to the collisions with the molecules of the fluid, see [15, p. 39].

Let $\Pi = \Omega' \times \cdots \times \Omega' \subset \mathbf{R}^N$, where $\times$ denotes the Cartesian product of sets. Let Υ' be a topology of subsets of Π. Let s, t, be such that $0 \leq s \leq t$, let $\mathbf{x} \in \Pi$, $A \in \Upsilon'$, then the transition distribution function of a N-dimensional stochastic process $\{\mathbf{X}(t) : t \geq 0\}$ is defined as follows:

$$T(s, \mathbf{x}, t, A) = \mathsf{P}\{\mathbf{X}(t) \in A \text{ and } \mathbf{X}(s) = \mathbf{x}\}. \tag{6}$$

When T can be written as:

$$T(s, \mathbf{x}, t, A) = \int_A p(s, \mathbf{x}, t, \mathbf{y})\ d\mathbf{y} \tag{7}$$

for every $0 \leq s \leq t$, $\mathbf{x} \in \Pi$, $A \in \Upsilon'$ then the function p is called the *transition probability density* of the process $\{\mathbf{X}(t): t \geq 0\}$.

Finally, if there exists a density distribution function π that depends only on $\mathbf{x} \in \Pi$ such that:

$$\pi(\mathbf{x}) = \lim_{t \to +\infty} p(s, \mathbf{u}, t, \mathbf{x}), \tag{8}$$

then π is called the *steady-state distribution density* of the process $\{\mathbf{X}(t): t \geq 0\}$.

One considers the following stochastic differential equation:

$$d\mathbf{Z}(t) = \boldsymbol{\alpha}(\mathbf{Z}(t), t)\, dt + \beta(\mathbf{Z}(t), t)\, d\mathbf{w}(t), \quad t > 0, \tag{9}$$

$$\mathbf{Z}(0) = \mathbf{x}^0, \tag{10}$$

where $\mathbf{w}$ is the N-dimensional Brownian motion, α is the drift coefficient and β is the diffusion coefficient, see [8, p. 98] or [8, p. 196] for a detailed discussion. One notes that $d\mathbf{w}$ cannot be considered as a differential in the elementary sense and must be understood as a stochastic differential, see [8, p. 59]. Under regularity assumptions on α and β there exists a unique solution $\{\mathbf{Z}(t): t > 0\}$ of (9), (10), see [8, p. 98].

When $\boldsymbol{\alpha}$ is minus the gradient of a potential function equation (9) is called the *Smoluchowski-Kramers equation*. The Smoluchowski-Kramers equation is a singular limit of the Langevin equation. The Langevin equation expresses Newton principle for a particle subject to a random force field, see [15, p. 40].

Let $\mathrm{div}_{\mathbf{y}}$ be the divergence operator with respect to the variables $\mathbf{y}$, $\Delta_{\mathbf{y}}$ be the Laplace operator with respect to the variables $\mathbf{y}$ and $L_{\beta}, \alpha(\cdot) = \mathrm{div}_{\mathbf{y}}(\cdot\alpha) - (1/2)\Delta_{\mathbf{y}}(\cdot\beta^2)$. Under regularity assumptions on α and β, the transition probability density $p(s, \mathbf{x}, t, \mathbf{y})$, $0 \leq s < t$, $\mathbf{x}, \mathbf{y} \in \mathbf{R}^N$, associated to the solution $\{\mathbf{Z}(t): t \geq 0\}$ of problem (9), (10) exists and satisfies the Fokker–Planck equation, (see 8, p. 149]) that is, given $\mathbf{x} \in \mathbf{R}^N$, $s \geq 0$ one has:

$$\frac{\partial p}{\partial t} + L_{\beta,\boldsymbol{\alpha}}(p) = 0, \quad \mathbf{y} \in \mathbb{R}^N,\ t > s, \tag{11}$$

$$\lim_{t \to s, t > s} p(s, \mathbf{x}, t, \mathbf{y}) = \delta(\mathbf{x} - \mathbf{y}), \quad \mathbf{y} \in \mathbb{R}^N. \tag{12}$$

For the treatment of the constrained global optimization, that is, problem (1) with $D \subset \mathbf{R}^N$, a stochastic process depending on the domain D must be considered. Let $\nu(\mathbf{x}) \subset \mathbf{S}^{N-1}$ be the set-valued function that gives the outward unit normals of the boundary ∂D of D at the point $\mathbf{x} \in \partial D$. One notes that when $\mathbf{x}$ is a regular point of ∂D, $\nu(\mathbf{x})$ is a singleton. Let η: $[0, T] \to \mathbf{R}^N$, with possibly $[0, T] = \mathbf{R}^+$, let $|\eta|(t)$ be the total variation of η in the interval [0, t], where $t < T$. The *Skorokhod problem* is defined as follows: let ϕ, ψ, η: $[0, T] \to \mathbf{R}^N$, then the triple (ϕ, ψ, η) satisfies the Skorokhod problem, on $[0, T]$ with respect to D, if $|\eta|(T) < +\infty$, $\phi(0) = \psi(0)$ and for $t \in [0, T]$ the following relations hold:

$$\boldsymbol{\phi}(t) = \boldsymbol{\psi}(t) + \boldsymbol{\eta}(t), \tag{13}$$

$$\boldsymbol{\phi}(t) \in D, \tag{14}$$

$$|\boldsymbol{\eta}|(t) = \int_0^t \chi_{\{r \in \mathbb{R}:\ \boldsymbol{\phi}(r) \in \partial D\}}(s)\, d\,|\boldsymbol{\eta}|(s), \tag{15}$$

$$\boldsymbol{\eta}(t) = -\int_0^t \boldsymbol{\gamma}(s)\, d\,|\boldsymbol{\eta}|(s), \tag{16}$$

where χ_S is the characteristic function of the set S and $\boldsymbol{\gamma}(s) \in \nu(\phi(s))$, when $s \in [0, T]$ and $\boldsymbol{\phi}(s) \in \partial D$ and $\gamma(s) = \mathbf{O}$ elsewhere. Viewing $\psi(t)$, $t \in [0, T]$, as the trajectory of a point $\mathbf{A} \in \mathbf{R}^N$, one has that at time zero $\mathbf{A}$ is inside D, since $\psi(0) \in D$. Moreover the trajectory of $\mathbf{A}$ is reflected from the boundary of D and the reflected trajectory can be viewed as $\phi(t)$, $t \in [0, T]$. That is, ϕ is equal to ψ until $\mathbf{A} \in D$, when $\mathbf{A}$ goes out of D it is brought back on ∂D in the normal direction to ∂D. One

notes that the function $\boldsymbol{\eta}$ gives the reflection rule with respect to the boundary of D of the function ψ. In [16] it is proved that under suitable assumptions on D and F there exists a unique solution of the Skorokhod problem.

One considers the following stochastic differential equation with reflection term, that is:

$$d\mathbf{Z}(t) = \boldsymbol{\alpha}(\mathbf{Z}(t), t)\, dt + \beta(\mathbf{Z}(t), t)\, d\mathbf{w}(t) + d\boldsymbol{\eta}(t), \quad t > 0\,, \tag{17}$$

$$\mathbf{Z}(0) = \mathbf{x}^0, \tag{18}$$

where

$$(\mathbf{Z}, \mathbf{Z} - \boldsymbol{\eta}, \boldsymbol{\eta}) \tag{19}$$

is the solution of the Skorokhod problem. One notes that relations (14), (19) imply that the solution of (17), (18) verifies $\mathbf{Z}(t) \in D$, $t > 0$. In [16] it is proved that under some hypotheses there exists a unique solution $\{\mathbf{Z}(t)\colon t \geq 0\}$ of (17), (18), (19) for every $\mathbf{x}^0 \in D$.

Global Unconstrained Optimization

Given problem (1) with $D = \mathbf{R}^N$ one considers the following stochastic differential equation:

$$d\mathbf{Z}(t) = -(\nabla F)(\mathbf{Z}(t))\, dt + \sigma(t) d\mathbf{w}(t), \quad t > 0, \tag{20}$$

$$\mathbf{Z}(0) = \mathbf{x}^0, \tag{21}$$

where $\{\mathbf{w}(t)\colon t \geq 0\}$ is the N-dimensional Brownian motion and $\sigma(t)$ is a suitable decreasing function that guarantees the convergence of the stochastic process $\{\mathbf{Z}(t) : t \geq 0\}$ to a random variable with density concentrated on the global minimizers of F. Under some assumptions on F, the transition probability density $p(0, \mathbf{x}^0, t, \mathbf{x})$, $\mathbf{x}^0$, $\mathbf{x} \in \mathbf{R}^N$, $t > 0$, of the process $\{\mathbf{Z}(t) : t \geq 0\}$ exists and verifies equations (11), (12); moreover, when $\sigma \equiv \epsilon$, $\epsilon > 0$, for the steady-state distribution density $\pi_\epsilon(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^N$, the following equation holds:

$$L_{\epsilon, -\nabla F}(\pi_\epsilon) = 0, \quad \mathbf{x} \in \mathbb{R}^N, \tag{22}$$

one has:

$$\pi_\epsilon(\mathbf{x}) = C_\epsilon e^{-\frac{2F(\mathbf{x})}{\epsilon^2}}, \quad \mathbf{x} \in \mathbb{R}^N, \quad \epsilon > 0, \tag{23}$$

where:

$$C_\epsilon = \left(\int_{\mathbb{R}^N} e^{-\frac{2F(\mathbf{y})}{\epsilon^2}}\, d\mathbf{y} \right)^{-1}, \quad \epsilon > 0. \tag{24}$$

One assumes $C_\epsilon < +\infty$ for $\epsilon > 0$. Moreover, one has:

$$p(0, \mathbf{x}^0, t, \mathbf{x}) = \pi_\epsilon(\mathbf{x}) + e^{-\frac{2F(\mathbf{x}^0)}{\epsilon^2}} \sum_{n=1}^{\infty} \pi_\epsilon^n(\mathbf{x}) \pi_\epsilon^n(\mathbf{x}^0) e^{\lambda_\epsilon^n t}, \tag{25}$$

where π_ϵ^n is the eigenfunction of $L_{\epsilon, -\nabla F}$ corresponding to the eigenvalue λ_ϵ^n, $n = 1, 2, \ldots$, and $0 = \lambda_\epsilon^0 > \lambda_\epsilon^1 > \cdots$. One notes that the eigenfunctions π_ϵ^n, $n = 1, 2 \ldots$, are appropriately normalized and π_ϵ is the eigenfunction of $L_{\epsilon, -\nabla F}$ corresponding to the eigenvalue $\lambda_\epsilon^0 = 0$. Consider $N = 1$, the function F smooth and with three extrema in $x^-, x^0, x^+ \in \mathbf{R}$ such that $x^- < x^0 < x^+$. Moreover, F increases in (x^-, x^0) and in $(x^+, +\infty)$ and decreases in $(-\infty, x^-)$ and in (x^0, x^+). Let:

$$\begin{cases} c^- = \dfrac{d^2F}{dx^2}(x^-), \\ c^0 = \dfrac{d^2F}{dx^2}(x^0), \\ c^+ = \dfrac{d^2F}{dx^2}(x^+). \end{cases} \tag{26}$$

One assumes $c^\pm$, c^0 to be nonzero. In [1] it is shown that when $F(x^-) < F(x^+)$, one has $\pi_\epsilon(x) \to \delta(x - x^-)$ as $\epsilon \to 0$ while when $F(x^-) = F(x^+)$ one has $\pi_\epsilon(x) \to \gamma\delta(x - x^-) + (1 - \gamma)\delta(x - x^+)$, where $\gamma = [1 + \sqrt{\frac{c^-}{c^+}}]^{-1}$ as $\epsilon \to 0$, and the limits are taken in distribution sense. That is in [1] it is shown that the steady-state distribution density tends to Dirac deltas concentrated on the global minimizers of F when $\epsilon \to 0$.

In [12] it is shown that:

$$\lambda_\epsilon^1 \approx -\frac{\sqrt{c^+ c^0}}{2\pi} e^{-\frac{2}{\epsilon^2}\delta F} \quad \text{as} \quad \epsilon \to 0, \tag{27}$$

where $\delta F = \max\{F(x^0) - F(x^-), F(x^0) - F(x^+)\}$. Formula (25) shows that p converges to π_ϵ when $t \to +\infty$, but the rate of convergence becomes slow when ϵ is small. Replacing ϵ with $\sigma(t)$ a slowly decreasing function such that $\sigma(t) \to 0$ when $t \to +\infty$, using elementary adiabatic perturbation theory one can expect that

the condition:

$$\int_0^{+\infty} e^{-\frac{2}{\sigma^2(t)}\delta F}\, dt = +\infty \tag{28}$$

guarantees that $\{\mathbf{Z}(t) : t > 0\}$ is a solution of (20), (21) when $t \to +\infty$ converges to a random variable concentrated on the global minimizers of F.

In [6] the following result is proved:

Theorem 1 (*convergence theorem*) *Let $F : \boldsymbol{R}^N \to \boldsymbol{R}$ be a twice continuously differentiable function satisfying the following properties:*

$$\min_{\mathbf{x}\in\mathbb{R}^N} F(\mathbf{x}) = 0, \tag{29}$$

$$\lim_{\|\mathbf{x}\|\to+\infty} F(\mathbf{x}) = \lim_{\|\mathbf{x}\|\to+\infty} \|(\nabla F)(\mathbf{x})\| = +\infty, \tag{30}$$

$$\lim_{\|\mathbf{x}\|\to+\infty} \|(\nabla F)(\mathbf{x})\|^2 - (\Delta F)(\mathbf{x}) > -\infty, \tag{31}$$

let $\sigma(t) = \sqrt{(c)/(\log t)}$ for $t \to +\infty$, where $c > c_F > 0$ and c_F is a constant depending on the function F. Then the transition probability density p of the process $\{\mathbf{Z}(t) : t \geq 0\}$, solution of (20), (21), converges weakly to a stationary distribution π, that is:

$$p(0, \mathbf{x}, t, \cdot) \to \pi \quad \text{when } t \to +\infty. \tag{32}$$

Moreover the distribution π is the weak limit of π_ϵ, given by (23), as $\epsilon \to 0$.

One notes that (20), (21) is obtained perturbing the trajectories given by the steepest descent equation for F with the Brownian motion and σ is a factor that controls the amplitude of this perturbation. The fact that $\sigma(t) \to 0$ when $t \to +\infty$ makes possible the stabilization of the perturbed trajectories at the minimizers of F. With the assumptions of the convergence theorem it is possible to conclude that π is concentrated on the global minimizers of F, so that the random variable $\mathbf{Z}(t) = (Z_1(t), \ldots, Z_N(t))$ 'converges' to $\mathbf{x}^*$, solution of problem (1), as $t \to +\infty$. That is, when $\mathbf{x}^*$ is the unique global minimizer of F, then $\mathrm{P}\{Z_i(t) = x_i^*\} \to 1$ when $t \to +\infty$ for $i = 1, \ldots, \mathrm{N}$.

The stochastic differential equation (20) can be integrated numerically to obtain an algorithm for the solution of problem (1). Let $t_0 = 0$, $t_k = \sum_{l=0}^{k-1} h_l$, where $h_l > 0$, $l = 0, 1, \ldots$, are such that $t_k \to +\infty$ when $k \to \infty$ then using the Euler method one has:

$$\boldsymbol{\zeta}^0 = \mathbf{x}^0, \tag{33}$$

$$\boldsymbol{\zeta}^{k+1} = \boldsymbol{\zeta}^k - h_k(\nabla F)(\boldsymbol{\zeta}^k) + \sigma(t_k)(\mathbf{w}(t_k + h_k) - \mathbf{w}(t_k)), \tag{34}$$

where $k = 0, 1, \ldots$ and $\zeta^k \in \mathbf{R}^N$ is the approximation of $\mathbf{Z}(t_k)$, $k = 1, 2, \ldots$, see [2,3].

In (34) due to the presence of the stochastic term, one can substitute the gradient of F with a kind of 'stochastic gradient' of F in order to save computational work, see [2,3] for details.

One notes that the sequence $\{\zeta^k : k \in \mathbf{N}\}$ depends on the particular realization of the Brownian motion $\{\mathbf{w}(t_k) : k = 0, 1, \ldots\}$. That is, solving several times problem (20), (21), by means of (33), (34), the solutions obtained are not necessarily the same. However, the convergence theorem states that 'all' the solutions $\{\zeta^k : k \in \mathbf{N}\}$ obtained by (33), (34) tend to $\mathbf{x}^*$ as $k \to +\infty$.

So that in the numerical algorithm derived from (20), (21) using (33), (34) one can approximate by means of n_T independent realizations (i. e., trajectories) of the stochastic process $\{\mathbf{Z}(t) : t \geq 0\}$, solution of (20), (21). A possible strategy for a numerical algorithm is the following: after an 'observation period' the various trajectories are compared, one of them is discarded and is not considered any more, another one is branched. The new set of trajectories are computed throughout the next observation period. The following stopping conditions are used:

- uniform stop: the final values of the function F at the end of the various trajectories are numerically equal;
- maximum trial duration: a maximum number of observation periods has been reached.

One notes that the algorithms based on the discretization of the stochastic differential equations have sound mathematical basis, that is for a wide class of functions F some convergence results such as the convergence theorem given above are available. These algorithms usually have a slow convergence rate, this can be seen from the kind of function σ which is required in the convergence theorem. This implies that the algorithms based on stochastic differential equations have an high computational cost, so that their use is usually restricted to low-dimensional problems. However these algorithms can be parallelized with a significant computational advantage, for example in the algorithm described above each trajectory can be computed independently from the others until the end of an observation period. One notes that the algorithms derived

from (20), (21) are in some sense similar to the simulated annealing algorithm (cf. also ▶ Simulated annealing methods in protein folding) introduced in combinatorial optimization in [11].

Global Constrained Optimization

Given problem (1) with $D \subset \mathbf{R}^N$ the following stochastic differential equation with reflection term is considered:

$$d\mathbf{Z}(t) = -(\nabla F)(\mathbf{Z}(t))\, dt + \sigma(t)\, d\mathbf{w}(t) + d\boldsymbol{\eta}(t), \quad t > 0, \tag{35}$$

$$\mathbf{Z}(0) = \mathbf{x}^0, \tag{36}$$

where $\mathbf{x}^0 \in D$, $\{\mathbf{w}(t) : t \geq 0\}$ is the N-dimensional Brownian motion, $\sigma(t)$ is a suitable decreasing function that guarantees the convergence of the stochastic process $\{\mathbf{Z}(t) : t > 0\}$ to a random variable with density concentrated on the global minimizers of F on D when $t \to +\infty$ and $\eta(t)$ is a suitable function to assure $\mathbf{Z}(t) \in D$, $t > 0$, that is, $(\mathbf{Z}, \mathbf{Z} - \eta, \eta)$ is the solution of the Skorokhod problem in $\mathbf{R}^+$ respect to D.

Let int(D) be the set of the interior points of D. One assumes that D is the closure of int(D). Let $p(0, \mathbf{x}^0, \mathrm{t}, \mathbf{x})$, $\mathbf{x}^0, \mathbf{x} \in \mathrm{int}(D)$, $t > 0$, be the transition probability density of the process $\{\mathbf{Z}(t)$: $t > 0\}$, solution of (35), (36), when $\sigma \equiv \epsilon$, $\epsilon > 0$. Then p satisfies the Fokker–Planck equation:

$$\frac{\partial p}{\partial t} + L_{\epsilon, -\nabla F}(p) = 0, \quad \mathbf{x} \in \mathrm{int}(D), \tag{37}$$

$$\lim_{t \to 0^+} p(0, \mathbf{x}^0, t, \mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}^0), \quad \mathbf{x} \in \mathrm{int}(D), \tag{38}$$

$$\left(\frac{\epsilon^2}{2} \nabla_{\mathbf{x}} p + p \nabla F, \mathbf{n}(\mathbf{x})\right) = 0, \quad \mathbf{x} \in \partial D, \quad t > 0, \tag{39}$$

where $L_{\epsilon, -\nabla F}$ is defined in (11) (12) and $\mathbf{n}(\mathbf{x}) \in \nu(\mathbf{x})$ is the outward unit normal to ∂D in $\mathbf{x} \in \partial D$. One notes that boundary condition (39) assures that $\mathsf{P}\{\mathbf{Z}(t) \in D\} = 1$ for every $t > 0$. This boundary condition follows from the requirement that $(\mathbf{Z}, \mathbf{Z} - \eta, \eta)$ is the solution of the Skorokhod problem.

One assumes the following properties of F and D:

- $F : D \to \mathbf{R}$ is twice continuously differentiable;
- $D \subset \mathbf{R}^N$ is a bounded convex domain such that exists p satisfying (37), (38), (39) and exists the steady-state distribution density π of the process solution of (35), (36);
- let π_ϵ be the steady-state distribution density of the process solution of (35), (36) when $\sigma \equiv \epsilon$, $\epsilon > 0$, that is:

$$\pi_\epsilon(\mathbf{x}) = C_\epsilon e^{-\frac{2F(\mathbf{x})}{\epsilon^2}}, \quad \mathbf{x} \in D, \tag{40}$$

$$C_\epsilon = \left(\int_D e^{-\frac{2F(\mathbf{y})}{\epsilon^2}}\, d\mathbf{y}\right)^{-1} \tag{41}$$

and π is the weak limit of π_ϵ as $\epsilon \to 0$.

In analogy with the unconstrained case one can conjecture that when $D \subset \mathbf{R}^N$ and $F : D$ rarr; $\mathbf{R}$ satisfy the properties listed above and when $\sigma(t) = \sqrt{(c)/(\log t)}$ for $t \to +\infty$, where $c > c_F > 0$ and c_F is a constant depending on F, then the transition probability density $p(0, \mathbf{x}^0, \mathrm{t}, \mathbf{y})$, $\mathbf{x}^0, \mathbf{x} \in D$, $t > 0$ of the process $\{\mathbf{Z}(t)$: $t \geq 0\}$, solution of (35), (36) converges to a steady-state distribution density π when $t \to +\infty$ and π is the distribution density obtained as weak limit of π_ϵ when $\epsilon \to 0$. That is, the process $\{\mathbf{Z}(\mathrm{t})$: $t \geq 0\}$ converges in law to a random variable concentrated at the points $\mathbf{x}^* \in D$ that solve problem (1).

A numerical algorithm to solve problem (1), with $D \subset \mathbf{R}^N$, can be obtained using a numerical method to integrate problem (35), (36). This is done integrating numerically problem (20), (21) and 'adding' the constraints given by D. In the numerical algorithm the trajectories can be computed using formulas (33), (34) when the trajectories are in D, when a trajectory violates the constraints, it is brought back on ∂D putting to zero its normal component with respect to the violated constraints. Finally the stopping conditions are the same ones considered in the previous section.

Analogously to the unconstrained problem, the algorithms based on the stochastic differential equations for the constrained case have slow convergence rate. However these algorithms have a high rate of parallelism.

Miscellaneous Results

In this section are shown two mathematical problems that are somewhat unusual as optimization problems.

Clique Problem

Let $I = \{1, \ldots, N\} \subset \mathbf{N}$ be a finite set, let $I * I$ be the set of unordered pairs of elements of I. Let $E \subseteq I * I$. Then a *graph* G is a pair $G = (I, E)$, where I is the set of the nodes of G and E is the set of the edges of G, i. e. $\{i, j\} \in E$ implies that G has an edge joining nodes $i, j \in I$. A graph $G = (I, E)$ is said to be *complete* or to be a *clique* when $E = I * I$. A graph $G' = (I', E')$ is a subgraph of $G = (I, E)$ when $I' \subseteq I$ and $E' \subseteq E \cap (I' * I')$.

The *maximum clique problem* can be defined as follows: Given $G = (I, E)$, find the largest subgraph G' of G which is complete. Let $k(G)$ be the number of nodes of the graph G'.

Several algorithms exist to obtain a numerical solution of the maximum clique problem see, for example, [14] where the branch and bound algorithm is described.

One considers here the maximum clique problem as a continuous optimization problem. The adjacency matrix A of the graph $G = (I, E)$ is a square matrix of order equal to the number of nodes of G and its generic entry $A_{i,j}$, at row i and at column j, is defined equal to 1 if $\{i, j\} \in E$ and is equal to 0 otherwise. Then in [13] it is shown that:

$$1 - \frac{1}{k(G)} = \max_{\mathbf{x} \in S} \mathbf{x}^t A \mathbf{x}, \tag{42}$$

where

$$S = \left\{ \mathbf{x} = (x_1, \ldots, x_N)^t \in \mathbb{R}^N : \sum_{i=1}^{N} x_i = 1,\ x_i \geq 0,\ i = 1, \ldots, N \right\}.$$

One notes that many maximizers of (42) can exist, however there exists always a maximizer $\mathbf{x}^* = (x_1^*, \ldots, x_N^*)^t$ of problem (42) such that for $i = 1, \ldots, N$ one has $x_i^* = 1/k(G)$ if $i \in G'$ and $x_i^* = 0$ if $i \notin G'$. That is the maximum clique problem is reduced to a continuous global optimization problem that can be treated with the algorithms described above. Several other problems in graph theory can be reformulated as continuous optimization problems.

Quasivariational Inequalities

Let $X \subset \mathbf{R}^N$ be a nonempty set, let $\Omega(\mathbf{x}) \subset X$, $\mathbf{x} \in X$, be a set-valued function and let $\mathbf{F} : \mathbf{R}^N \to \mathbf{R}^N$. The *quasivariational inequality* problem, is defined as follows: Find a vector $\mathbf{x}^* \in \Omega(\mathbf{x}^*)$ such that:

$$\left(\mathbf{F}(\mathbf{x}^*), \mathbf{y} - \mathbf{x}^*\right) \geq 0, \quad \forall \mathbf{y} \in \Omega(\mathbf{x}^*), \tag{43}$$

see [4] for a detailed introduction to quasivariational inequalities. This problem can be reduced to the search of a fixed-point of a function defined implicitly by a variational inequality.

The quasivariational inequalities have many applications such as for example the study of the generalized Nash equilibrium points of an N-player noncooperative game. See [10] for a detailed discussion on N-player noncooperative games.

See also

- ▶ αBB Algorithm
- ▶ Continuous Global Optimization: Applications
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ DIRECT Global Optimization Algorithm
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Binary Star Astronomy
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization Using Space Filling
- ▶ Topology of Global Optimization

References

1. Aluffi-Pentini F, Parisi V, Zirilli F (1985) Global optimization and stochastic differential equations. J Optim Th Appl, 47:1–17
2. Aluffi-Pentini F, Parisi V, Zirilli F (1988) A global optimization algorithm using stochastic differential equations. ACM Trans Math Software, 14:345–365
3. Aluffi-Pentini F, Parisi V, Zirilli F (1988) SIGMA - A stochastic integration global minimization algorithm. ACM Trans Math Softw 14:366–380
4. Baiocchi C, Capelo A (1984) Variational and quasi-variational inequalities: Application to Free-boundary problems. Wiley, New York
5. Billingsley P (1995) Probability and measure. Wiley, New York
6. Chiang TS, Hwang CR, Sheu SJ (1987) Diffusion for global optimization in Rn. SIAM J Control Optim 25:737–753
7. Dantzing GB (1963) Linear programming and extensions. Princeton Univ Press, Princeton
8. Friedman A (1975) Stochastic differential equations and applications, vol 1. Acad Press, New York

9. Gill PE, Murray W, Wright MH (1981) Practical optimization. Acad Press, New York
10. Harker P, Pang J (1990) Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications. Math Program 48:161–220
11. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
12. Matkowsky BJ, Schuss Z (1981) Eigenvalues of the Fokker-Planck operator and the equilibrium for the diffusions in potential fields. SIAM J Appl Math 40:242–254
13. Motzkin TS, Straus EG (1964) Maxima for graphs and a new proof of a theorem of Turán. Notices Amer Math Soc 11:533–540
14. Pardalos PM, Rodgers GP (1990) Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing 45:131–144
15. Schuss Z (1980) Theory and applications of stochastic differential equations. Wiley, New York
16. Tanaka H (1979) Stochastic differential equations with reflecting boundary conditions in convex regions. Hiroshima Math J 9:163–177

Dini and Hadamard Derivatives in Optimization

VLADIMIR F. DEMYANOV
St. Petersburg State University, St. Petersburg, Russia

MSC2000: 90Cxx, 65K05

Article Outline

Keywords
Directional Derivatives
First Order Necessary and Sufficient Conditions for an Unconstrained Optimum
Conditions for a Constrained Optimum
See also
References

Keywords

Dini directional derivatives; Hadamard directional derivatives; Necessary and sufficient optimality conditions; Bouligand cone; Composition theorem; Lipschitz function; Quasidifferentiable function; Concave function; Convex function; First order approximation of a function; Nondifferentiable optimization; Nonsmooth analysis; Numerical methods; Maximizer; Maximum function; Minimum function; Minimizer; Steepest ascent direction; Steepest descent direction; Unconstrained optimum

Directional Derivatives

Let f be a function defined on some open set $X \subset \mathbf{R}^n$ and taking its values in $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. The set $\operatorname{dom} f = \{x \in X : |f(x)| < +\infty\}$ is called the *effective set* (or *domain*) of the function f. Take $x \in \operatorname{dom} f$, $g \in \mathbf{R}^n$. Put

$$f_{\mathrm{D}}^{\uparrow}(x,g) := \limsup_{\alpha \downarrow 0} \frac{1}{\alpha}\big[f(x+\alpha g) - f(x)\big], \tag{1}$$

$$f_{\mathrm{D}}^{\downarrow}(x,g) := \liminf_{\alpha \downarrow 0} \frac{1}{\alpha}\big[f(x+\alpha g) - f(x)\big]. \tag{2}$$

Here $\alpha \downarrow 0$ means that $\alpha \to +0$.

The quantity $f_{\mathrm{D}}^{\uparrow}(x,g)$ (respectively, $f_{\mathrm{D}}^{\downarrow}(x,g)$) is called the *Dini upper* (respectively, *lower*) derivative of the function f at the point x in the direction g.

The limit

$$f'(x,g) = f'_{\mathrm{D}}(x,g) := \lim_{\alpha \downarrow 0} \frac{1}{\alpha}\big[f(x+\alpha g) - f(x)\big], \tag{3}$$

is called the *Dini derivative* of f at the point x in the direction g. If the limit in (3) exists, then $f_{\mathrm{D}}^{\uparrow}(x,g) = f_{\mathrm{D}}^{\downarrow}(x,g) = f'(x,g)$.

The quantity

$$f_{\mathrm{H}}^{\uparrow}(x,g) := \limsup_{[\alpha,g'] \to [+0,g]} \frac{1}{\alpha}\big[f(x+\alpha g') - f(x)\big] \tag{4}$$

(respectively,

$$f_{\mathrm{H}}^{\downarrow}(x,g) := \liminf_{[\alpha,g'] \to [+0,g]} \frac{1}{\alpha}\big[f(x+\alpha g') - f(x)\big]), \tag{5}$$

is called the *Hadamard upper* (respectively, *lower*) derivative of the function f at the point x in the direction g.

The limit

$$f'_{\mathrm{H}}(x,g) := \lim_{[\alpha,g'] \to [+0,g]} \frac{1}{\alpha}\big[f(x+\alpha g') - f(x)\big] \tag{6}$$

is called the *Hadamard derivative* of f at x in the direction g.

If the limit in (6) exists, then $f_{\mathrm{H}}^{\uparrow}(x,g) = f_{\mathrm{H}}^{\downarrow}(x,g)$.

Note that the limits in (1), (2), (4) and (5) always exist but are not necessarily finite.

Remark 1 In the one-dimensional case ($\mathbf{R}^n = \mathbf{R}$) the Hadamard directional derivatives coincide with the corresponding Dini directional derivatives:

$$f_{\mathrm{H}}^{\uparrow}(x,g) = f_{\mathrm{D}}^{\uparrow}(x,g),$$
$$f_{\mathrm{H}}^{\downarrow}(x,g) = f_{\mathrm{D}}^{\downarrow}(x,g),$$
$$f'_{\mathrm{H}}(x,g) = f'_{\mathrm{D}}(x,g).$$

If the limit in (3) exists and is finite, then the function f is called *differentiable* (or *Dini differentiable*) at x in the direction g. The function f is called *Dini directionally differentiable* (Dini d.d.) at the point x if it is Dini differentiable at x for every $g \in \mathbf{R}^n$. Analogously, if the limit in (6) exists and is finite, the function f is called *Hadamard differentiable* at x in the direction g. The function f is called *Hadamard directionally differentiable* (Hadamard d.d.) at the point x if it is Hadamard differentiable at x for every $g \in \mathbf{R}^n$.

If the limit in (6) exists and is finite, then the limit in (3) also exists and $f'_H(x,g) = f'(x,g)$. The converse is not necessarily true.

All these derivatives are *positively homogeneous* (of degree one) functions of direction:

$$f_Q^*(x,\lambda g) = \lambda f_Q^*(x,g), \quad \forall \lambda \geq 0. \tag{7}$$

(Here $*$ is either $\uparrow$, or $\downarrow$, and Q is either D, or H.)

A function f defined on an open set X is called *Dini uniformly directionally differentiable* at a point $x \in X$ if it is directionally differentiable at x and for every $\varepsilon > 0$ there exists a real number $\alpha_0 > 0$ such that

$$\frac{1}{\alpha}\left[f(x+\alpha g) - f(x) - \alpha f'(x,g)\right] < \varepsilon,$$
$$\forall \alpha \in (0,\alpha_0), \quad \forall g \in S,$$

where $S = \{g \in \mathbf{R}^n : \|g\| = 1\}$ is the unit sphere.

Proposition 2 (see [2, Thm. I.3.2]) *A function f is Hadamard d.d. at a point $x \in X$ if and only if it is Dini uniformly differentiably at x and its directional derivative $f'(x,g)$ is continuous as a function of direction.*

Remark 3 If f is locally Lipschitz and Dini directionally differentiable at $x \in X$, then it is Hadamard d.d. at x, too.

For Dini and Hadamard derivatives (see (3) and (6)) there exists a calculus:

Proposition 4 *Let functions f_1 and f_2 be Dini (Hadamard) directionally differentiable at a point $x \in X$. Then their sum, difference, product and quotient (if $f_2(x) \neq 0$) are also Dini (Hadamard) d.d. at this point and the following formulas hold:*

$$(f_1 \pm f_2)'_Q(x,g) = f'_{1Q}(x,g) \pm f'_{2Q}(x,g), \tag{8}$$

$$(f_1 f_2)'_Q(x,g) = f_1(x) f'_{2Q}(x,g) + f_2(x) f'_{1Q}(x,g), \tag{9}$$

$$\left(\frac{f_1}{f_2}\right)'_Q(x,g) = -\frac{1}{(f_2(x))^2}\left(f_1(x) f'_{2Q}(x,g) - f_2(x) f'_{1Q}(x,g)\right). \tag{10}$$

Here Q is either D, or H.

These formulas follow from the classical theorems of differential calculus.

Proposition 5 *Let*

$$\varphi(x) = \max_{i \in 1:N} f_i(x), \tag{11}$$

where the functions f_i are defined and continuous on an open set $X \subset \mathbf{R}^n$ and Dini (Hadamard) d.d. at a point $x \in X$ in a direction g. Then the function φ is also Dini (Hadamard) d.d. at x and

$$\varphi'_Q(x,g) = \max_{i \in R(x)} f'_i(x,g), \tag{12}$$

where $R(x) = \{i \in 1:N : f_i(x) = \varphi(x)\}$ (see [2, Cor. I.3.2]).

If φ is defined by

$$\varphi(x) = \max_{y \in Y} f_i(x,y),$$

where Y is some set, then under some additional conditions a formula, analogous to (12), also holds (see [2, Chap. I, Sec. 3]).

A theorem on the differentiability of a composition can also be stated.

Unfortunately, formulas similar to (8)–(10) and (12) are not valid for Dini (Hadamard) upper and lower derivatives.

The Dini and Hadamard upper and lower directional derivatives are widely used in nonsmooth analysis and nondifferentiable optimization. For example, the following *mean value theorem* holds.

Proposition 6 (see [2, Thm. I.3.1]) *Let f be defined and continuous on the interval $\{y: y = x + \alpha g, \alpha \in [0, \alpha_0], \alpha_0 > 0\}$. Put*

$$m = \inf_{\alpha\in[0,\alpha_0]} f_D^{\downarrow}(x+\alpha g, g),$$

$$M = \sup_{\alpha\in[0,\alpha_0]} f_D^{\uparrow}(x+\alpha g, g).$$

Then [1]

$$m\alpha_0 \le f(x+\alpha_0 g) - f(x) \le M\alpha_0.$$

The following first order approximations may be constructed via the Dini and Hadamard derivatives.

Proposition 7 *Let f be defined on an open set $X \subset \boldsymbol{R}^n$, and Dini d.d. at a point $x \in X$. Then*

$$f(x+\Delta) = f(x) + f'_D(x,\Delta) + o_D(x,\Delta). \tag{13}$$

If f is Hadamard d.d. at x, then

$$f(x+\Delta) = f(x) + f'_H(x,\Delta) + o_H(x,\Delta). \tag{14}$$

Let f be defined on an open set $X \subset \mathbf{R}^n$ and finite at $x \in X$. Then

$$f(x+\Delta) = f(x) + f_D^{\uparrow}(x,\Delta) + \overline{o}_D(x,\Delta), \tag{15}$$

$$f(x+\Delta) = f(x) + f_D^{\downarrow}(x,\Delta) + \underline{o}_D(x,\Delta), \tag{16}$$

$$f(x+\Delta) = f(x) + f_H^{\uparrow}(x,\Delta) + \overline{o}_H(x,\Delta), \tag{17}$$

$$f(x+\Delta) = f(x) + f_H^{\downarrow}(x,\Delta) + \underline{o}_H(x,\Delta), \tag{18}$$

where

$$\frac{o_D(x,\alpha\Delta)}{\alpha} \overset{\alpha\downarrow 0}{\to} 0, \quad \forall \Delta \in \mathbb{R}^n, \tag{19}$$

$$\frac{o_H(x,\alpha\Delta)}{\|\Delta\|} \overset{\|\Delta\|\to 0}{\to} 0, \tag{20}$$

$$\limsup_{\alpha\downarrow 0} \frac{\overline{o}_D(x,\alpha\Delta)}{\alpha} = 0, \quad \forall \Delta \in \mathbb{R}^n, \tag{21}$$

$$\liminf_{\alpha\downarrow 0} \frac{\underline{o}_D(x,\alpha\Delta)}{\alpha} = 0, \quad \forall \Delta \in \mathbb{R}^n, \tag{22}$$

$$\limsup_{[\alpha,\Delta']\to[+0,\Delta]} \frac{\overline{o}_H(x,\alpha\Delta')}{\alpha} \ge 0, \quad \forall \Delta \in \mathbb{R}^n, \tag{23}$$

$$\liminf_{[\alpha,\Delta']\to[+0,\Delta]} \frac{\underline{o}_H(x,\alpha\Delta')}{\alpha} \le 0, \quad \forall \Delta \in \mathbb{R}^n. \tag{24}$$

First Order Necessary and Sufficient Conditions for an Unconstrained Optimum

Let a function f be defined on an open set $X \subset \mathbf{R}^n$, Ω be a subset of X. A point $x^* \in \Omega$ is called a *local minimum point* (*local minimizer*) of the function f on the set Ω if there exists $\delta > 0$ such that

$$f(x) \ge f(x^*), \quad \forall x \in \Omega \cap B_\delta(x^*),$$

where $B_\delta(x^*) = \{x \in \mathbf{R}^n : \| x - x^* \| \le \delta\}$. If $\delta = +\infty$, then the point x^* is called a *global minimum point* (*global minimizer*) of f on Ω. A point $x^* \in \Omega$ is called a *strict local minimum point* (*strict local minimizer*) of f on Ω if there exists $\delta > 0$ such that

$$f(x) > f(x^*), \quad \forall x \in \Omega \cap B_\delta(x^*), \quad x \ne x^*.$$

Analogously one can define local, global and strict *local maximum points* (*maximizers*) of f on Ω.

It may happen that the set of local (global, strict local) minimizers (maximizers) is empty.

If $\Omega = X$ then the problem of finding a minimum or a maximum of f on X is called an *unconstrained optimization problem.*

Proposition 8 *Let a function f be Dini (Hadamard) directionally differentiable on X. For a point $x^* \in \operatorname{dom} f$ to be a local or global minimizer of f on X it is necessary that*

$$f'_D(x^*, g) \ge 0, \quad \forall g \in \mathbb{R}^n, \tag{25}$$

$$\left(f'_H(x^*, g) \ge 0 \quad \forall g \in \mathbb{R}^n \right). \tag{26}$$

If f is Hadamard d.d. at x^* and

$$f'_H(x^*, g) > 0, \quad \forall g \in \mathbb{R}^n, \quad g \ne 0_n, \tag{27}$$

then x^* is a strict local minimizer of f.

Here $0_n = (0, \ldots, 0)$ is the zero element of $\mathbf{R}^n$.

Proposition 9 *Let f be Dini (Hadamard) d.d. on X. For a point $x^{**} \in \operatorname{dom} f$ to be a local or global maximizer of f on X it is necessary that*

$$f'_D(x^{**}, g) \le 0, \quad \forall g \in \mathbb{R}^n, \tag{28}$$

$$(f'_H(x^{**}, g) \le 0, \quad \forall g \in \mathbb{R}^n). \tag{29}$$

If f is Hadamard d.d. at x^{**} and

$$f'_H(x^{**}, g) < 0, \quad \forall g \in \mathbb{R}^n, \quad g \neq 0_n, \tag{30}$$

then x^{**} is a strict local maximizer of f.

Note that (26) implies (25), and (29) implies (28). In the smooth case $f'_H(x, g) = (f'(x), g)$ ($f'(x)$ being the gradient of f at x) and the conditions (27) and (30) are impossible. It means that the sufficient conditions (27) and (30) are essentially nonsmooth.

Proposition 10 *Let f be defined on an open set on $X \subset \boldsymbol{R}^n$. For a point $x^* \in \operatorname{dom} f$ (i. e., $|f(x)| < +\infty$) to be a local or global minimizer of f on X it is necessary that*

$$f_D^{\downarrow}(x^*, g) \geq 0, \quad \forall g \in \mathbb{R}^n, \tag{31}$$

$$f_H^{\downarrow}(x^*, g) \geq 0, \quad \forall g \in \mathbb{R}^n. \tag{32}$$

If

$$f_H^{\downarrow}(x^*, g) > 0, \quad \forall g \in \mathbb{R}^n, \quad g \neq 0_n, \tag{33}$$

then x^* is a strict local minimizer of f.

Note that (32) implies (31) but (31) does not necessarily imply (32).

Proposition *Let f be defined on an open set on $X \subset \boldsymbol{R}^n$. For a point $x^{**} \in \operatorname{dom} f$ to be a local or global maximizer of f on X it is necessary that*

$$f_D^{\uparrow}(x^{**}, g) \leq 0, \quad \forall g \in \mathbb{R}^n \tag{34}$$

and

$$f_H^{\uparrow}(x^{**}, g) \leq 0, \quad \forall g \in \mathbb{R}^n. \tag{35}$$

If

$$f_H^{\uparrow}(x^{**}, g) < 0, \quad \forall g \in \mathbb{R}^n, \quad g \neq 0_n, \tag{36}$$

then x^{**} is a strict local maximizer of f.

The condition (35) implies (34) but (34) does not necessarily imply (35).

Remark 12 Observe that the conditions for a minimum are different from the conditions for a maximum.

A point x^* satisfying the conditions (25) or (31) is called a *Dini* inf-*stationary point* of f, while a point x^* satisfying (26) or (32) is called an *Hadamard* inf-*stationary point*. A point x^{**} satisfying the conditions (28) or (34) is called a *Dini* sup-*stationary point* of f, while a point x^{**} satisfying (28) or (35) is called an *Hadamard* sup-*stationary point*.

Remark 13 Note that the function f is not assumed to be continuous or even finite-valued.

Let $x_0 \in \operatorname{dom} f$ and assume that the condition (31) does not hold, i. e. x_0 is not a Dini inf-stationary point. If $g_0 \in \mathbf{R}^n$, $\|g_0\| = 1$,

$$f_{\mathrm{D}}^{\downarrow}(x_0, g_0) = \inf_{\|g\|=1} f_{\mathrm{D}}^{\downarrow}(x_0, g),$$

then g_0 is called a *Dini steepest descent direction* of f at x_0 ($\|g\|$ is the Euclidean norm).

If (32) does not hold and if $g_0 \in \mathbf{R}^n$, $\|g_0\| = 1$,

$$f_{\mathrm{H}}^{\downarrow}(x_0, g_0) = \inf_{\|g\|=1} f_{\mathrm{H}}^{\downarrow}(x_0, g),$$

then g_0 is called an *Hadamard steepest descent direction* of f at x_0.

Analogously if x_0 is not a Dini sup-stationary point and if $g^0 \in \mathbf{R}^n$, $\|g^0\| = 1$,

$$f_{\mathrm{D}}^{\uparrow}(x_0, g^0) = \sup_{\|g\|=1} f_{\mathrm{D}}^{\uparrow}(x_0, g),$$

then g^0 is called a *Dini steepest ascent direction* of f at x_0.

If x_0 is not an Hadamard sup-stationary point of f (i. e. (35) does not hold) and if $g^0 \in \mathbf{R}^n$, $\| g^0 \| = 1$,

$$f_{\mathrm{H}}^{\uparrow}(x_0, g^0) = \sup_{\|g\|=1} f_{\mathrm{H}}^{\uparrow}(x_0, g),$$

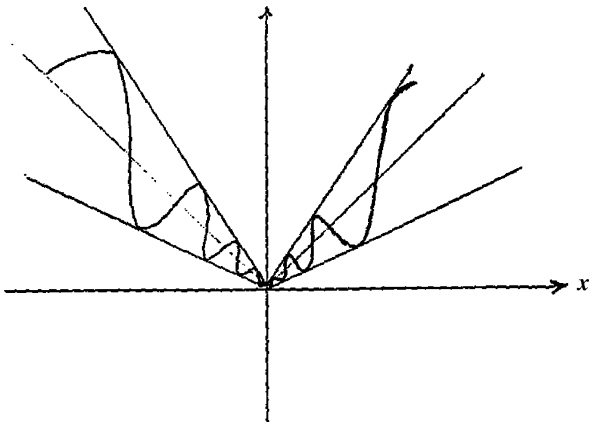
then g^0 is called an *Hadamard steepest ascent direction* of f at x_0.

Of course it is possible that there exist many steepest descent or/and steepest ascent directions of f at x_0.

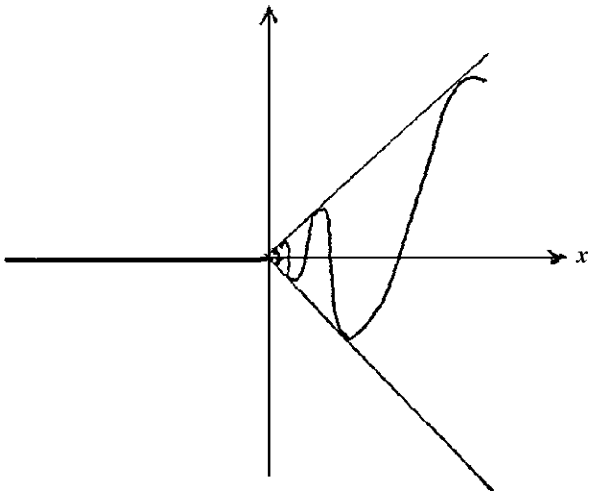
It may also happen that some direction is a direction of steepest ascent and, at the same time, a direction of steepest ascent as well (which is impossible in the smooth case).

Example 14 Let $X = \mathbf{R}$,

$$f(x) = \begin{cases} |x| + \frac{1}{2} x \sin \frac{1}{x}, & x \neq 0, \\ 0, & x = 0. \end{cases}$$



Dini and Hadamard Derivatives in Optimization, Figure 1



Dini and Hadamard Derivatives in Optimization, Figure 2

Take $x_0 = 0$. It is clear that (see Fig. 1):

$$f_{\mathrm{D}}^{\uparrow}(x_0, g) = |g| + \frac{1}{2}|g| = \frac{3}{2}|g|,$$
$$f_{\mathrm{D}}^{\downarrow}(x_0, g) = |g| - \frac{1}{2}|g| = \frac{1}{2}|g|.$$

As $X = \mathbf{R}$, the Hadamard derivatives coincide with the Dini ones (see Remark 1).

$$f_{\mathrm{D}}^{\downarrow}(x_0, g) > 0, \quad \forall g \neq 0,$$

we may conclude (see (32)) that x_0 is a strict local minimizer (in fact it is a global minimizer but our theory does not allow us to claim this).

Note that $f_{\mathrm{D}}^{\uparrow}$ and $f_{\mathrm{D}}^{\downarrow}$ are positively homogeneous (see (7)), therefore it is sufficient to consider (in $\mathbf{R}$) only two directions: $g_1 = 1$ and $g_2 = -1$.

Example 15 Let $X = \mathbf{R}$, $x_0 = 0$,

$$f(x) = \begin{cases} x \sin \frac{1}{x}, & x > 0, \\ 0, & x \leq 0. \end{cases}$$

It is clear that (see Fig. 2) that

$$f_{\mathrm{D}}^{\uparrow}(x_0, g) = \begin{cases} |g|, & g > 0, \\ 0, & g \leq 0, \end{cases}$$
$$f_{\mathrm{D}}^{\downarrow}(x_0, g) = \begin{cases} -|g|, & g > 0, \\ 0, & g \leq 0. \end{cases}$$

Neither the condition (25) nor the condition (31) holds, therefore we conclude that x_0 is neither a local minimizer nor a local maximizer. Since

$$\max_{\|g\|=1} f_{\mathrm{D}}^{\uparrow}(x_0, g)$$
$$= \max\{f_{\mathrm{D}}^{\uparrow}(x_0, +1), f_{\mathrm{D}}^{\uparrow}(x_0, -1)\}$$
$$= \max\{1, 0\} = f_{\mathrm{D}}^{\uparrow}(x_0, +1) = +1,$$

then $g_1 = +1$ is a steepest ascent direction.

Since

$$\min_{\|g\|=1} f_{\mathrm{D}}^{\downarrow}(x_0, g)$$
$$= \min\{f_{\mathrm{D}}^{\downarrow}(x_0, +1), f_{\mathrm{D}}^{\downarrow}(x_0, -1)\}$$
$$= \min\{-1, 0\} = f_{\mathrm{D}}^{\downarrow}(x_0, +1) = -1,$$

then $g_1 = +1$ is a steepest descent direction as well.

Conditions for a Constrained Optimum

Let a function f be defined on an open set $X \subset \mathbf{R}^n$, Ω be a subset of X. Let $x \in \Omega$, $|f(x)| < +\infty$, $g \in \mathbf{R}^n$. The limit

$$f_{\mathrm{D}}^{\uparrow}(x, g; \Omega) = \limsup_{\substack{\alpha \downarrow 0 \\ x + \alpha g \in \Omega}} \frac{f(x + \alpha g) - f(x)}{\alpha} \tag{37}$$

is called the *Dini conditional upper derivative* of the function f at the point x in the direction g with respect to Ω. If no sequence $\{\alpha_k\}$ exists such that $\alpha_k \downarrow 0$, $x + \alpha_k g \in \Omega$ for all k, then, by definition, we set $f_{\mathrm{D}}^{\uparrow}(x, g; \Omega) = -\infty$.

The limit

$$f_{\mathrm{D}}^{\downarrow}(x,g;\Omega)=\liminf_{\substack{\alpha\downarrow 0\\ x+\alpha g\in\Omega}}\frac{f(x+\alpha g)-f(x)}{\alpha} \tag{38}$$

is called the *Dini conditional lower derivative* of the function f at the point x in the direction g with respect to Ω. If no sequence $\{\alpha_k\}$ exists such that $\alpha_k\downarrow 0$, $x+\alpha_k g\in\Omega$ for all k, then, by definition, we set $f_{\mathrm{D}}^{\downarrow}(x,g;\Omega)=+\infty$.

The limit

$$f_{\mathrm{H}}^{\uparrow}(x,g;\Omega)=\limsup_{\substack{[\alpha,g']\to[+0,g]\\ x+\alpha g'\in\Omega}}\frac{f(x+\alpha g')-f(x)}{\alpha} \tag{39}$$

is called the *Hadamard conditional upper derivative* of the function f at the point x in the direction g with respect to Ω. If no sequences $\{\alpha_k\}$, $\{g_k\}$ exist such that $[\alpha_k, g_k]\to[+0,g]$, $x+\alpha_k g_k\in\Omega$ for all k, then, by definition, we set $f_{\mathrm{H}}^{\uparrow}(x,g;\Omega)=-\infty$.

The limit

$$f_{\mathrm{H}}^{\downarrow}(x,g;\Omega)=\liminf_{\substack{[\alpha,g']\to[+0,g]\\ x+\alpha g'\in\Omega}}\frac{f(x+\alpha g')-f(x)}{\alpha} \tag{40}$$

is called the *Hadamard conditional lower derivative* of f at x in the direction g with respect to Ω. If no sequences $\{\alpha_k\}$, $\{g_k\}$ exist such that $[\alpha_k, g_k]\to[+0,g]$, $x+\alpha_k g_k\in\Omega$ for all k, then, by definition, we set $f_{\mathrm{H}}^{\downarrow}(x,g;\Omega)=+\infty$.

Proposition 16 (see [1]) *For a point $x^*\in\Omega$ and such that $|f(x^*)|<\infty$ to be a local or global minimizer of f on Ω it is necessary that*

$$f_{D}^{\downarrow}(x^*,g;\Omega)\geq 0,\quad \forall g\in\mathbb{R}^n, \tag{41}$$

$$f_{H}^{\downarrow}(x^*,g;\Omega)\geq 0,\quad \forall g\in\mathbb{R}^n. \tag{42}$$

Furthermore, if

$$f_{H}^{\downarrow}(x^*,g;\Omega)>0,\quad \forall g\in\mathbb{R}^n,\quad g\neq 0_n, \tag{43}$$

then x^* is a strict local minimizer of f on Ω.

A point $x^*\in\Omega$ satisfying (41) ((42)) is called a *Dini* (*Hadamard*) inf-*stationary point* of f on Ω.

Proposition 17 *For a point $x^{**}\in\Omega$ and such that $|f(x^{**})|<\infty$ to be a local or global minimizer of f on Ω it is necessary that*

$$f_{D}^{\uparrow}(x^{**},g;\Omega)\leq 0,\quad \forall g\in\mathbb{R}^n, \tag{44}$$

$$f_{H}^{\uparrow}(x^{**},g;\Omega)\leq 0,\quad \forall g\in\mathbb{R}^n. \tag{45}$$

If

$$f_{H}^{\uparrow}(x^{**},g;\Omega)<0,\quad \forall g\in\mathbb{R}^n,\quad g\neq 0, \tag{46}$$

then x^{**} is a strict local maximizer of f on Ω.

A point $x^{**}\in\Omega$ satisfying (44) ((45)) is called a *Dini* (*Hadamard*) sup-*stationary point* of f on Ω.

The condition (41) is equivalent to

$$f_{\mathrm{D}}^{\downarrow}(x^*,g;\Omega)\geq 0,\quad \forall g\in K(x^*,\Omega), \tag{47}$$

where

$$K(x^*,\Omega)=\left\{g\in\mathbb{R}^n:\ \exists\alpha_k:\ \begin{array}{c}\alpha_k\downarrow 0,\\ x^*+\alpha_k g\in\Omega,\\ \forall k\end{array}\right\}. \tag{48}$$

Analogously, the condition (44) is equivalent to

$$f_{\mathrm{D}}^{\uparrow}(x^{**},g;\Omega)\leq 0,\quad \forall g\in K(x^{**},\Omega). \tag{49}$$

The condition (42) is equivalent to

$$f_{\mathrm{H}}^{\downarrow}(x^*,g;\Omega)\geq 0,\quad \forall g\in\Gamma(x^*,\Omega), \tag{50}$$

where

$$\Gamma(x^*,\Omega)=\left\{g\in\mathbb{R}^n:\ \exists\{[\alpha_k,g_k]\}:\ \begin{array}{c}[\alpha_k,g_k]\to[+0,g],\\ x^*+\alpha_k g_k\in\Omega,\\ \forall k\end{array}\right\}. \tag{51}$$

Analogously, the condition (45) is equivalent to

$$f_{\mathrm{H}}^{\uparrow}(x^{**},g;\Omega)\leq 0,\quad \forall g\in\Gamma(x^{**},\Omega). \tag{52}$$

Note that the cones $K(x^*,\Omega)$ and $K(x^{**},\Omega)$ are not necessarily closed, while the cones $\Gamma(x^*,\Omega)$ and $\Gamma(x^{**},\Omega)$ are the Bouligand cones to Ω at x^* and x^{**}, respectively, and therefore always closed.

Now it is possible to define conditional steepest ascent and descent directions.

Remark 18 It is also possible (see [3, p. 156]) to define the Dini and Hadamard conditional directional derivatives as follows:

$$f'_{\mathrm{D}}(x,g;\Omega)=\lim_{\substack{\alpha\downarrow 0\\ x+\alpha g\in\Omega}}\frac{f(x+\alpha g)-f(x)}{\alpha}, \tag{53}$$

$$f'_{\mathrm{H}}(x,g;\Omega) = \lim_{\substack{[\alpha,g']\to[+0,g]\\ x+\alpha g'\in\Omega}} \frac{f(x+\alpha g')-f(x)}{\alpha}. \quad (54)$$

A function f is called *Dini* (*Hadamard*) *conditionally differentiable* at x in a direction g if the limit in (53) ((54)) exists and is finite.

Remark 19 The conditional directional derivatives defined by (37)–(40) essentially depend on the set Ω.

In some cases it is possible to 'separate' the function f and the set Ω in the necessary conditions (47), (49), (50) and (52). For example, if f is Lipschitz and directionally differentiable at x, then

$$\begin{aligned} &f^{\uparrow}_{\mathrm{D}}(x,g;\Omega) = f^{\downarrow}_{\mathrm{D}}(x,g;\Omega)\\ &= f^{\uparrow}_{\mathrm{H}}(x,g;\Omega) = f^{\downarrow}_{\mathrm{H}}(x,g;\Omega)\\ &= f'(x,g) \quad \forall g\in K(x,\Omega). \end{aligned}$$

In this case the derivatives at the left-hand sides of (47), (49) and (50), (52) should be replaced by $f'(x^*,g)$ or $f'(x^{**},g)$ respectively.

Note that if $g\in\Gamma(x,g)$ but $g\notin K(x,g)$ then $f^{\uparrow}_{\mathrm{D}}(x,g;\Omega)$ and $f^{\downarrow}_{\mathrm{D}}(x,g;\Omega)$ are not finite, by definition, while

$$f^{\uparrow}_{\mathrm{D}}(x,g;\Omega) = f^{\downarrow}_{\mathrm{D}}(x,g;\Omega) = f'(x,g).$$

Remark 20 The necessary optimality conditions for unconstrained and constrained optimization problems described above can be used to construct numerical methods for finding corresponding (inf- or sup-stationary) points.

For special classes of functions (e. g., convex, concave, max-type, minmax-type, quasidifferentiable functions), the derivative (3) has a more 'constructive' form and therefore the conditions (25)–(36) and (41)–(46) take also more 'constructive' forms (see, e. g., [2]).

Remark 22 The limits in (4), (5), (6), (39) and (40) are taken if

$$[\alpha,g']\to[+0,g]. \quad (55)$$

Sometimes, in the literature instead of this relation one can see two relations

$$\alpha\to+0, \quad g'\to g. \quad (56)$$

It was demonstrated in [4] that the limits resulting from (55) and (56) do not necessarily coincide. This warning should be taken into account.

See also

- Global Optimization: Envelope Representation
- Nondifferentiable Optimization
- Nondifferentiable Optimization: Cutting Plane Methods
- Nondifferentiable Optimization: Minimax Problems
- Nondifferentiable Optimization: Newton Method
- Nondifferentiable Optimization: Parametric Programming
- Nondifferentiable Optimization: Relaxation Methods
- Nondifferentiable Optimization: Subgradient Optimization Methods

References

1. Demyanov VF, Di Pillo G, Facchinei F (1997) Exact penalization via Dini and Hadamard conditional derivatives. Optim Methods Softw 9:19–36
2. Demyanov VF, Rubinov AM (1995) Constructive nonsmooth analysis. P Lang, Frankfurt am Main
3. Dieudonne J (1969) Foundations of modern analysis. Acad Press, New York
4. Giannessi F (1995) A common understanding or a common misuderstanding. Numer Funct Anal Optim 16(9–10):1359–1363

Directed Tree Networks

THOMAS ERLEBACH[1], KLAUS JANSEN[2], CHRISTOS KAKLAMANIS[3], GIUSEPPE PERSIANO[4]
[1] Department of Computer Science, University of Leicester, Leicester, UK
[2] Institut für Informatik, Univerisität Kiel, Kiel, Germany
[3] RA CTI, University of Patras, Patras, Greece
[4] Dipartimento di Informatica ed Appl., Università di Salerno, Fisciano, Italy

MSC2000: 05C85

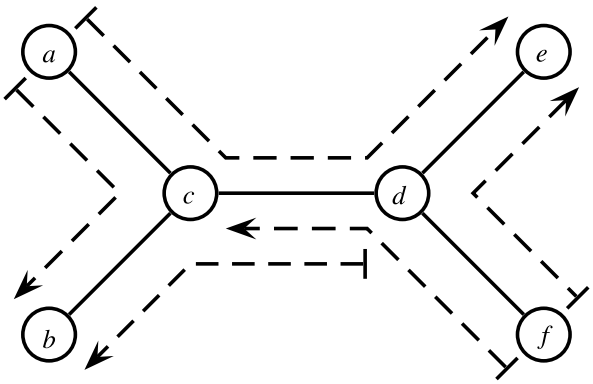
Article Outline

Keywords and Phrases
Complexity Results
Reduction from Edge Coloring
Reduction from Arc Coloring
Approximation Algorithms
Reduction to Constrained Bipartite Edge Coloring

Partition Into Matchings
Coloring of Triplets
Selection of Triplets
Lower Bounds
Lower Bound for Greedy Algorithms
Lower Bounds for Optimal Colorings
Randomized Algorithms
Related Topics
References

Keywords and Phrases

Optical networks; Path coloring; Bipartite edge coloring; Approximation algorithms



Directed Tree Networks, Figure 1
Example path coloring instance

Technological developments in the field of optical communication networks using *wavelength-division multiplexing* have triggered intensive research in an optimization problem concerning the assignment of colors to paths in a directed tree. Here, the term *directed tree* refers to the graph obtained from an undirected tree by replacing each undirected edge by two directed edges with opposite directions. This *path coloring problem* was first studied by M. Mihail, C. Kaklamanis and S. Rao [19]. An instance of it is given by a directed tree $T = (V, E)$ and a set $P = \{p_1, \ldots, p_t\}$ of directed simple (i. e., not visiting any vertex twice) paths in T, where each path is specified by an ordered pair of vertices (start vertex and end vertex). The task is to assign colors to the given paths such that paths receive different colors if they share a directed edge. The goal is to minimize the number of different colors used. For given $T = (V, E)$ and P, let $L(e)$ denote the load on directed edge $e \in E$, i. e., the number of paths containing e. Obviously, the maximum load $L = \max_{e \in E} L(e)$ is a lower bound on the number of colors in an optimal coloring. Consider Fig. 1 for an example of a tree with six vertices and paths from a to e, from f to e, from f to c, from d to b, and from a to b. A possible valid coloring is to assign these paths the colors 1, 2, 1, 2, and 3, respectively. The maximum load of the paths is 2, because 2 paths use the edge (d, c). It is not possible to color these paths with 2 colors, because the *conflict graph* of the paths (a graph with a vertex for each path and an edge between vertices if the corresponding paths share an edge) is a cycle of length 5. Hence, the coloring with three colors is an optimal coloring.

The path coloring problem models the *assignment of wavelengths* to directed connection requests in *all-optical networks* with *tree topology*. In such networks data is transmitted in optical form via laser beams [13]. Two adjacent nodes of the network are connected by a pair of optical fiber links, one for each direction. When wavelength-division multiplexing is used, multiple signals can be transmitted over the same link if they use different wavelengths, and the nodes are capable of switching an incoming signal onto any outgoing link depending on the wavelength of the signal. However, the wavelength of a signal cannot be changed, and every connection uses the same wavelength on the whole transmitter-receiver path. If two signals using the same wavelength are transmitted over the same directed link, the data is lost due to interference. The number of available wavelengths is called *optical bandwidth*, and it is a scarce resource. Therefore, one is interested in minimizing the number of wavelengths necessary to route a given set of requests. This optimization problem corresponds to the path coloring problem defined above: paths correspond to connection requests, and colors correspond to wavelengths.

Complexity Results

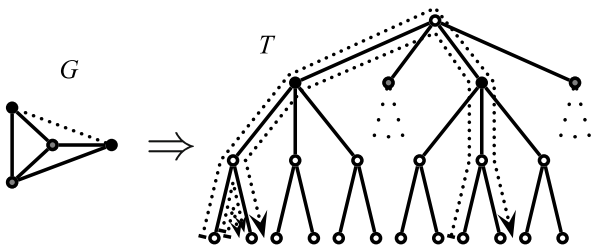
Whereas the path coloring problem can be solved in linear time in *chain networks* as it is equivalent to *interval graph* coloring, it is *NP*-hard in directed tree networks. More precisely, it is *NP*-complete to decide whether a set of paths in a directed tree of arbitrary degree can be colored using at most 3 colors [8,9], and it is *NP*-

complete to decide whether a set of paths in a directed binary tree can be colored using at most k colors if k is part of the input [7,9,17]. The respective *reductions* will be outlined in the following. It should be remarked that the special case in which both the maximum degree of the tree as well as the maximum load of the paths are bounded by a constant can be solved optimally in polynomial time [7,9].

Reduction from Edge Coloring

It is an *NP*-complete problem to decide whether the edges of a given 3-regular undirected graph G can be colored with three colors such that edges receive different colors if they are incident to the same vertex [14]. Let G be a 3-regular undirected graph with n vertices and m edges. A directed tree T with $n' = 10n + 1$ vertices and a set P of $t = 4m$ paths in T can be constructed in polynomial time such that the paths in T can be colored using three colors if and only if the edges of G can be colored with three colors. T consists of a root r, one child c_v of the root for every vertex v of G, three children c_v^1, c_v^2 and c_v^3 of every c_v, and two children $c_v^{i,1}$ and $c_v^{i,2}$ of every c_v^i. For each edge $e = \{v, w\}$ of G, four paths in T are created: one from $c_v^{i,1}$ to $c_w^{j,2}$ and one from $c_w^{j,1}$ to $c_v^{i,2}$, called real paths, and two copies of the path from $c_v^{i,1}$ to $c_v^{i,2}$, called blockers. Here i and j are chosen such that the subtree rooted at c_v^i resp. c_w^j is not used by any paths other than the paths created for this particular edge e. Figure 2 shows an example of a 3-regular graph G, the constructed tree T (two of the four subtrees are represented by dotted triangles), and the paths created for the edge between the black vertices of G.

If the paths in T are to be colored with three colors, the blockers ensure that the two real paths corresponding to e receive the same color and, therefore, this color cannot be used by any other real path corresponding to an edge incident to v or w. If there exists a 3-coloring of the paths in T, a 3-coloring of the edges of G can be obtained by assigning each edge the color of its corresponding real paths. On the other hand, if there exists a 3-coloring of the edges of G, a 3-coloring of the paths in T can be obtained by assigning the real paths corresponding to edge e the same color as e and coloring the blockers with the remaining two colors. Hence, a solution to the path coloring problem in T would also solve the *edge coloring* problem in G.



Directed Tree Networks, Figure 2
Reduction from edge coloring

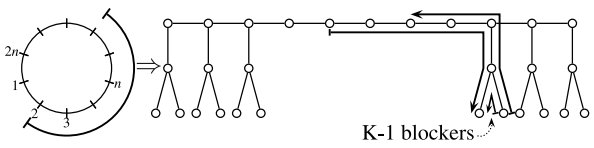
Since it has just been proved *NP*-complete to decide whether paths in a directed tree can be colored with three colors, it follows that there cannot be an approximation algorithm for path coloring with absolute *approximation ratio* $< 4/3$ unless $P = NP$.

Reduction from Arc Coloring

The *NP*-complete *arc coloring* problem [12] is to decide for a given set of n arcs $A_1, \ldots, A_n$ on a circle and a given integer k whether the arcs can be colored with k colors such that arcs receive different colors if they intersect. Without loss of generality, assume that each arc is specified by a pair (a_i, b_i) with $a_i \neq b_i$ and $1 \leq a_i, b_i \leq 2n$. The span of arc A_i is $sp(A_i) = \{a_i + 1, a_i + 2, \ldots, b_i\}$ if $a_i < b_i$ and $sp(A_i) = \{a_i + 1, \ldots, 2n, 1, 2, \ldots, b_i\}$ if $a_i > b_i$. Two arcs A_i and A_j intersect iff $sp(A_i) \cap sp(A_j) \neq \emptyset$. Note that one can view the arc coloring problem as a path coloring problem on a cycle.

If a number is contained in the span of more than k arcs, then the arcs can surely not be colored with k colors and the answer to this instance of arc coloring is *no*. Otherwise, one can assume that every number i, $1 \leq i \leq 2n$ is contained in the span of exactly k arcs; if this were not the case, one could simply add arcs of the form $(i, i + 1)$ until the condition holds, without changing the answer of the coloring problem.

Now consider a chain of n vertices $v_1, v_2, \ldots, v_n$. Imagine the chain drawn from left to right, with v_1 the start vertex at its left end. The directed edges from left to right followed by the directed edges from right to left make up a cycle of length $2n$. The given circular arcs can be translated into directed paths on this cycle such that two paths share a directed edge iff the corre-



Directed Tree Networks, Figure 3
Reduction from arc coloring

sponding arcs intersect, but these paths do not yet constitute a valid path coloring problem because some of the paths are not simple: an arc $(1, 2n)$ would correspond to a path running from v_1 to v_n and back to v_1, for example. Nevertheless, it is possible to obtain a valid instance of the path coloring problem by splitting paths that are not simple into two or three simple paths and by using blockers to make sure that the paths derived from one non-simple path must receive the same color in any valid k-coloring.

For this purpose, extend the chain by adding k vertices on both ends, resulting in a chain of length $n + 2k$. Connect each of the newly added vertices to a distinct subtree consisting of a new vertex with two leaf children. The resulting network is a binary tree T. If a path arrives at vertex v_n coming from the left (i. e., from v_{n-1}) and "turns around" to revisit v_{n-1}, divide the path into two: one coming from the left, passing through v_n and ending at the left leaf of one of the subtrees added on the right side of the chain; the other one starting at the right leaf of that subtree, passing through v_n and continuing left. In addition, add $k - 1$ blockers in that subtree, i. e., paths from the right leaf to the left leaf. Observe that there are no more than k paths containing v_n as an inner vertex, and a different subtree can be chosen for each of these paths. A symmetric splitting procedure is applied to the paths that contain v_1 as an inner vertex, i. e., the paths that arrive at v_1 coming from the right (i. e., from v_2) and "turn around" to revisit v_2. This way, all non-simple paths are split into two or three simple paths, and a number of blockers are added.

The resulting set of paths in T can be colored with k colors if and only if the original arc coloring instance is a *yes*-instance. The blockers ensure that all paths corresponding to the same arc receive the same color in any k-coloring. Hence, a k-coloring of the paths can be used to obtain a k-coloring of the arcs by assigning each arc the color of its corresponding paths. Also, a k-coloring of the arcs can be turned into a k-coloring of the paths by assigning all paths corresponding to an arc the same color as the arc and by coloring the blockers with the remaining $k - 1$ colors. This shows that the decision version of the path coloring problem is *NP*-complete already for binary trees.

Approximation Algorithms

Since the path coloring problem in directed tree networks is *NP*-hard, one is interested in polynomial-time *approximation algorithms* with provable *performance guarantee*. All such approximation algorithms that have been developed so far belong to the class of *greedy algorithms*. A greedy algorithm picks a start vertex s in the tree T and assigns colors to the paths touching (starting at, ending at, or passing through) s first. Then it visits the remaining vertices of the tree in some order that ensures that the current vertex is adjacent to a previously visited vertex; for example, a *depth-first search* can be used to obtain such an order. When the algorithm processes vertex v, it assigns colors to all paths touching v without changing the color of paths that have been colored at a previous vertex. Each such step is referred to as *coloring extension*. Furthermore, the only information about the paths touching the current vertex that the algorithm considers is which edges incident to the current vertex they use. To emphasize this latter property, greedy algorithms are sometimes referred to as *local greedy algorithms*.

Whereas all greedy algorithms follow this general strategy, individual variants differ with respect to the solution to the coloring extension substep. The best known algorithm was presented by T. Erlebach, K. Jansen, C. Kaklamanis, and P. Persiano in [11,16] (see also [10]). It colors a set of paths with maximum load L in a directed tree network of arbitrary degree with at most $\lceil 5L/3 \rceil$ colors. In the next section this will be shown to be best possible in the class of greedy algorithms.

For the sake of clarity, assume that the load on all edges is exactly L and that L is divisible by 3. The algorithm maintains two *invariants*: (a) the number of colors used is at most $5L/3$, and (b) for each pair of directed edges with opposite directions the number of colors used to color paths going through either of these edges is at most $4L/3$. First, the algorithm picks a leaf s

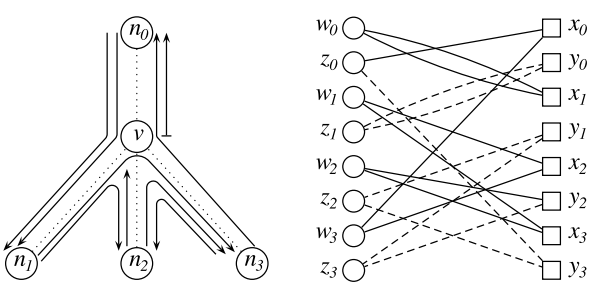
of T as the start vertex and colors all paths starting or ending at s using at most L colors. Therefore, the invariants are satisfied initially. It remains to show that they still hold after a coloring extension step if they were satisfied at the beginning of this step.

Reduction to Constrained Bipartite Edge Coloring

The coloring extension problem at a current vertex v is reduced to a *constrained edge coloring* problem in a *bipartite graph* G_v with left vertex set V_1 and right vertex set V_2. This reduction was introduced by M. Mihail, C. Kaklamanis and S. Rao in [19]. Let $n_0, n_1, \ldots, n_k$ be the neighbors of v in T, and let n_0 be the unique neighbor that was processed before v. For every neighbor n_i of v the graph G_v contains four vertices: vertices w_i and z_i in V_1, and vertices x_i and y_i in V_2. Vertex w_i is said to be opposite x_i, and z_i is opposite y_i. A pair of opposite vertices is called a *line* of G_v. A line *sees* a color if it appears on an edge incident to a vertex of that line. For every path touching v there is one edge in G_v: an edge (w_i, x_j) for each path coming from n_i, passing through v and going to n_j; an edge (w_i, y_i) for each path coming from n_i and ending at v; and an edge (z_i, x_i) for each path starting at v and going to n_i.

It is easy to see that coloring the paths touching v is equivalent to coloring the edges of G_v. Note that the vertices w_i and x_i have degree L in G_v, while the other vertices may have smaller degree. If this is the case, the algorithm adds dummy edges (shown dashed in Fig. 4) in order to make the graph L-regular.

As the paths that contain the edges (n_0, v) or (v, n_0) have been colored at a previous vertex, the edges incident to w_0 and x_0 are already colored with at most $4L/3$ colors by invariant (b). These edges are called *color-forced* edges. A color that appears on exactly one color-forced edge is a *single color*. A color that appears on two color-forced edges is a *double color*. Since there are at most $4L/3$ colors on $2L$ color-forced edges, there must be at least $2L/3$ double colors. Furthermore, one can assume that there are exactly $2L/3$ double colors and $2L/3$ single colors, because if there are too many double colors then it is possible to split an appropriate number of double colors into two single colors for the duration of the current coloring extension step. In order to maintain invariant (a), the algorithm must color the uncolored edges of G_v using at most $L/3$ new colors (colors not used on the color-forced edges). Invariant (b) is satisfied by ensuring that no line of G_v sees more than $4L/3$ colors.



Directed Tree Networks, Figure 4
Construction of the bipartite graph

Partition Into Matchings

G_v is an L-regular bipartite graph and its edges can thus be partitioned into L *perfect matchings* efficiently. Each *matching* is classified according to the colors on its two color-forced edges: SS-matchings contain two single colors, ST-matchings contain one single color and one double color, PP-matchings contain the same (preserved) double color on both color-forced edges, and TT-matchings contain two different double colors. Next, the L matchings are grouped into chains and cycles: a chain of length $\ell \geq 2$ is a sequence of ℓ matchings $M_1, \ldots, M_\ell$ such that M_1 and M_ℓ are ST-matchings, $M_2, \ldots, M_{\ell-1}$ are TT-matchings, and two consecutive matchings share a double color; a cycle of length $\ell \geq 2$ is a sequence of ℓ TT-matchings such that consecutive matchings as well as the first and the last matching share a double color. Obviously, the set of L matchings is in this way entirely partitioned into SS-matchings, chains, cycles, and PP-matchings. In addition, if a chain or cycle contains parallel color-forced edges, then the algorithm exchanges these edges in the respective matchings, thus dividing the original chain or cycle into a shorter sequence of the same type and an extra cycle.

Now the algorithm chooses *triplets*, i. e., groups of three matchings, and colors the uncolored edges of each triplet using at most one new color and at most four *active* colors. The active colors are selected among the colors on color-forced edges of that triplet, and a color is active in at most one triplet. The algorithm ensures

that a line that sees the new color does not see one of the active colors of that triplet. This implies that no line of G_v sees more than $4L/3$ colors altogether, as required to maintain invariant (b).

Coloring of Triplets

The rules for choosing triplets ensure that each triplet contains two color-forced edges with single colors and four color-forced edges with double colors. Furthermore, most triplets are chosen such that one double color appears twice, and this double color as well as the two single colors can be reused without considering conflicts outside the triplet. V. Kumar and E.J. Schwabe proved in [18] that such triplets can be colored as required using three active colors and one new color. This coloring procedure can be sketched as follows. Partition the edges of the triplet into a matching on all vertices except w_0 and x_0 and a *gadget*, i. e., a subgraph in which w_0 and x_0 have degree 3 while all other vertices have degree 2. A gadget consists of a number of cycles of even length not containing w_0 or x_0 and either three disjoint paths from w_0 to x_0 or one path from w_0 to x_0, one path from w_0 to w_0, and one path from x_0 to x_0. A careful case analysis shows that the triplet can be colored by reusing the single colors and the double color to color the gadget and using a new color for the matching. If a partitioning into gadget and matching does not exist, the triplet contains a PP-matching and can be colored using the double color of the PP-matching for the uncolored edges of the PP-matching and a single color and a new color for the uncolored edges of the cycle cover consisting of the other two matchings.

In the following, the terms *even sequence* and *odd sequence* refer to sequences of TT-matchings of even resp. odd length such that consecutive matchings share a double color. Note that an even sequence can be grouped into triplets by combining two consecutive matchings of the sequence with an SS-matching as long as SS-matchings are available and combining each remaining TT-matching with a chain of length 2. There are always enough SS-matchings or chains of length 2 because the ratio between color-forced edges with double colors and color-forced edges with single colors is $2:1$ in G_v initially and remains the same after extracting triplets. Similarly, an odd sequence can be grouped into triplets if there is at least one chain of length 2, which can be used to form a triplet with the first matching of the sequence, leaving an even sequence behind.

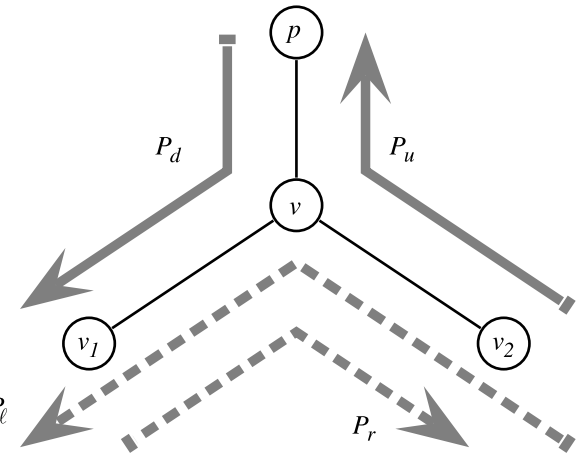
Selection of Triplets

Now the rules for selecting triplets are as follows. From chains of odd length, combine the first two matchings and the last matching to form a triplet. The remainder of the chain (if non-empty) is an even sequence and can be handled as described above. Cycles of even length are even sequences and can be handled the same way. As long as there is a chain of length 2 left, chains of even length ≥ 4 and odd cycles can be handled, too. Pairs of PP-matchings can be combined with an SS-matching, single PP-matchings can be combined with chains of length 2. If there are two chains of even length ≥ 4, combine the first two matchings of one chain with the last matching of the other and the last two matchings of the first chain with the first matching of the other, leaving two even sequences behind. So far, all triplets contained a double color twice and could be colored as outlined above. What remains is a number of cycles of odd length, at most one chain of even length, at most one PP-matching, and some SS-matchings. To deal with these, it is necessary to form some triplets that contain four distinct double colors. However, it is possible to ensure that the set of color-forced edges of G_v (inside and outside the triplet) colored with one of these double colors does not contain parallel edges; T. Erlebach, K. Jansen, C. Kaklamanis and P. Persiano showed in [11] that such a triplet can be colored as required using its single colors, two of its double colors, and one new color.

In the end, the entire graph G_v has been partitioned into triplets, and each triplet has been colored using at most one new color and such that a line that sees a new color in a triplet does not see one of the active colors of that triplet. Hence, invariants (a) and (b) hold at the end of the coloring extension step, and once the coloring extension step has been performed for all vertices of T all paths have received one of $\lceil 5L/3 \rceil$ colors. Since the number OPT of colors necessary in an optimal coloring is at least L, this implies that the algorithm uses at most $\lceil 5OPT/3 \rceil$ colors to color the paths. From the lower bound in the next section it will be clear that the algorithm (and any other greedy algorithm) is not better than $5OPT/3$ in the worst case.

Note that greedy algorithms are well-suited for practical distributed implementation in optical networks: one node of the network initiates the wavelength assignment by assigning wavelengths to all connections going through that node; then it transfers control to its neighbors who can extend the assignment independently and in parallel, transferring control to their neighbors in turn once they are done.

It should be mentioned that simpler variants of greedy algorithms are known that are restricted to *binary trees* and color a given set of paths with load L using $\lceil 5L/3 \rceil$ colors. These algorithms do not make use of the reduction to constrained bipartite edge coloring [6,15].



Directed Tree Networks, Figure 5
Lower bound for greedy algorithms

Lower Bounds

Two kinds of *lower bounds* have been investigated for path coloring in directed tree networks. First, one wants to determine the best *worst-case performance guarantee* achievable by any greedy algorithm. Second, it is interesting to know how many colors are required even in an optimal coloring for a given set of paths with load L in the worst case.

Lower Bound for Greedy Algorithms

For a given local greedy algorithm A and positive integer L, an *adversary* can construct an instance of path coloring in a directed binary tree network such that A uses at least $\lfloor 5L/3 \rfloor$ colors while an optimal solution uses only L colors [15]. The construction proceeds inductively. As A considers only the edges incident to a vertex v when it colors the paths touching v, the adversary can determine how these paths should continue and introduce new paths not touching v depending on the coloring A produces at vertex v.

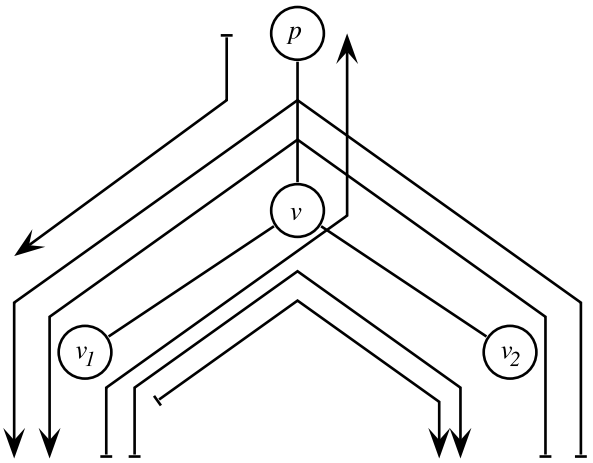
Assume that there are $\alpha_i L/2$ paths going through each of the directed edges between vertex v and its parent, and that these paths have been colored with $\alpha_i L$ different colors. Initially, this assumption can be satisfied for $\alpha_0 = 1$ by introducing L paths in either direction on the link between the start vertex picked by algorithm A and one of its neighbors and letting appropriately chosen $L/2$ of these paths start resp. end at that neighbor. Denote the set of paths coming down from the parent by P_d and let them continue to (pass through) the left child v_1 of v. Denote the set of paths going up to the parent by P_u and let them pass through the right child v_2 of v. Introduce a set P_ℓ of $(1 - \alpha_i/2)L$ paths coming from v_2 and going left to v_1, and a set P_r of L paths coming from v_1 and going right to v_2.

Algorithm A must use $(1 - \alpha_i/2)L$ new colors to color the paths in P_ℓ. No matter which colors it chooses for the paths in P_r, it will use at least $(1 + \alpha_i/4)L$ different colors on the connection between v and v_1 or on the connection between v and v_2. The best it can do with respect to minimizing the number of colors appearing between v and v_1 and between v and v_2 is to color $(1 - \alpha_i/2)L$ paths of P_r with colors used for P_ℓ, $\alpha_i L/4$ paths of P_r with colors used for P_d, and $\alpha_i L/4$ paths of P_r with colors used for P_u. In that case, it uses $(1 + \alpha_i/4)L$ colors on each of the downward connections of v. Any other assignment uses more colors on one of the downward connections.

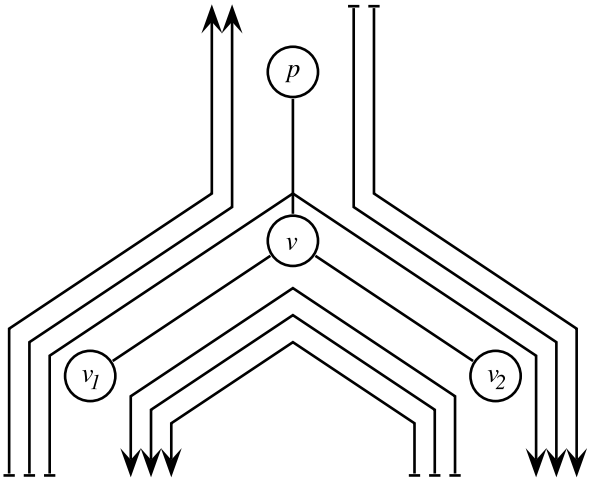
If the algorithm uses at least $(1 + \alpha_i/4)L$ different colors for paths on, say, the connection between v and v_1, let $(1 + \alpha_i/4)L/2$ of the downward paths and equally many of the upward paths extend to the left child of v_1, such that all of these paths use different colors, and let the remaining paths terminate or begin at v_1. Now the inductive assumption holds for the left child of v_1 with $\alpha_{i+1} = 1 + \alpha_i/4$. Hence, the number of colors on a pair of directed edges can be increased as long as $\alpha_i < 4/3$. When $\alpha_i = 4/3$, $4L/3$ colors are used for the paths touching v and its parent, and algorithm A must use $L/3$ new colors to color the paths in P_ℓ, using $5L/3$ colors altogether.

The previous calculations have assumed that all occurring terms like $(1+\alpha_i/4)L/2$ are integers. If one takes the possibility of non-integral values into account and carries out the respective calculations for all cases, one can show that, for every L, every greedy algorithm can be forced to use $\lfloor 5L/3 \rfloor$ colors on a set of paths with maximum load L [15].

Furthermore, it is not difficult to show that the paths resulting from this worst-case construction for greedy algorithms can be colored optimally using only L colors. Hence, this yields also a lower bound of $\lfloor 5OPT/3 \rfloor$ colors for any greedy algorithm.



Directed Tree Networks, Figure 6
Root component



Directed Tree Networks, Figure 7
Type A component

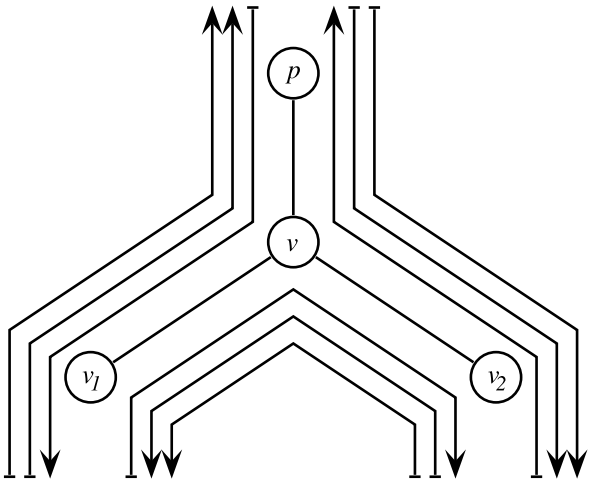
Lower Bounds for Optimal Colorings

The instance of path coloring depicted in Fig. 1 consists of 5 paths in a binary tree with maximum load $L = 2$ such that even an optimal coloring requires 3 colors. Consider the instances of path coloring obtained from this instance by replacing each path by ℓ identical copies. Such an instance consists of 5ℓ paths with maximum load $L = 2\ell$, and an optimal coloring requires at least $\lceil 5\ell/2 \rceil = \lceil 5L/4 \rceil$ colors because no more than two of the given paths can be assigned the same color. Furthermore, $\lceil 5\ell/2 \rceil$ colors are also sufficient to color these instances: for example, if ℓ is even, use colors $1, \ldots, \ell$ for paths from a to e, colors $\ell+1, \ldots, 2\ell$ for paths from f to e, colors $1, \ldots, \ell/2$ and $2\ell + 1, \ldots, 5\ell/2$ for paths from f to c, colors $\ell/2 + 1, \ldots, 3\ell/2$ for paths from d to b, and colors $3\ell/2 + 1, \ldots, 5\ell/2$ for paths from a to b. Hence, for every even L there is a set of paths in a binary tree with load L such that an optimal coloring requires $\lceil 5L/4 \rceil$ colors [4,18].

While the path coloring instance with $L = 2$ and $OPT = 3$ could be specified easily, K. Jansen used a more involved construction to obtain an instance with $L = 3$ and $OPT = 5$ [15]. It makes use of three components as building blocks. Each component consists of a vertex v with its parent and two children and a specification of the usage of edges incident to v by paths touching v.

The root component ensures that at least 3 colors are used either on the left downward connection (extending below v_1) or on the right downward connection (extending below v_2). Each child of the root component is connected to a type A component, i. e., the child is identified with the parent vertex of a type A component and the corresponding paths are identified as well.

Type A components have the property that, if the paths touching v and its parent are colored with 3 colors, at least 4 colors must be used either for the paths touching v and v_1 or for those touching v and v_2. (If the paths touching v and its parent are colored with 4 colors, the remaining paths of the type A component require even 5 colors.) Hence, there is at least one child in one of the two type A components below the root component such that the paths touching this child and its parent are colored with four colors.



Directed Tree Networks, Figure 8
Type B component

The final component used is of type B. It has the property that, if the paths touching v and its parent are colored with 4 colors, at least 4 colors must be used either for the paths touching v and v_1 or for those touching v and v_2. For certain arrangements of colors on the paths touching v and its parent, 5 colors are necessary. It is possible to arrange a number of type B components in a binary tree such that for any combination of four colors on paths entering the tree of type B components at its root, 5 colors are necessary to complete the coloring. Hence, if one attaches a copy of this tree of type B components to each of the children of a type A component, it is ensured that at least one of the trees will be entered by paths with four colors and consequently 5 colors are necessary to color all paths. Since the load on every directed edge is at most 3, this gives a worst-case example for path coloring in binary trees with $L = 3$ and $OPT = 5$.

Randomized Algorithms

In [1,2], V. Auletta, I. Caragiannis, C. Kaklamanis and G. Persiano presented a class of *randomized algorithms* for path coloring in directed tree networks. They gave a randomized algorithm that, with high probability, uses at most $7/5L + o(L)$ colors for coloring any set of paths of maximum load L on binary trees of height $o(L^{1/3})$. The analysis of the algorithm uses tail inequalities for hypergeometrical probability distributions such as *Azuma's inequality*. Moreover, they proved that no randomized greedy algorithm can achieve, with high probability, a performance ratio better than 3/2 for trees of height $\Omega(L)$ and better than $1.293 - o(1)$ for trees of constant height.

These results have been improved in [5] by I. Caragiannis, A. Ferreira, C. Kaklamanis, S. Pérennes, and H. Rivano, who gave a randomized approximation algorithm for bounded-degree trees that has approximation ratio $1.61 + o(1)$. The algorithm first computes in polynomial time an optimal solution for the fractional path coloring problem and then applies *randomized rounding* to obtain an integral solution.

Related Topics

A number of further results related to the path coloring problem in directed tree networks or in networks with different topology are known. The number of colors required for sets of paths that have a special form have been investigated, e. g., *one-to-all instances*, *all-to-all instances*, *permutations*, and *k-relations*. A survey of many of these results can be found in [4]. The undirected version of the path coloring problem has been studied by P. Raghavan and E. Upfal in [20]; here, the network is represented by an undirected graph and paths must receive different colors if they share an undirected edge. Approximation results for directed and undirected path coloring problems in *ring networks*, *mesh networks*, and arbitrary networks (all of these are *NP*-hard no matter whether the paths are fixed or can be chosen by the algorithm [7]) have been derived.

An on-line variant of path coloring was studied by Y. Bartal and S. Leonardo in [3]. Here, the algorithm is given connection requests one by one and must determine a path connecting the corresponding vertices and a color for this path without any knowledge of future requests. The worst-case ratio between the number of colors used by the *on-line algorithm* and that used by an optimal off-line algorithm with complete advance knowledge is the *competitive ratio*. In [3] on-line algorithms with competitive ratio $O(\log n)$ are presented for trees, trees of rings, and meshes with n vertices.

References

1. Auletta V, Caragiannis I, Kaklamanis C, Persiano P (2000) Randomized Path Coloring on Binary Trees. In: Jansen K,

Khuller S (eds) Approximation Algorithms for Combinatorial Optimization (APPROX 2000), LNCS, vol 1913. Springer, Berlin, pp 60–71
2. Auletta V, Caragiannis I, Kaklamanis C, Persiano P (2002) Randomized Path Coloring on Binary Trees. Theoret Comput Sci 289:355–399
3. Bartal Y, Leonardi S (1997) On-Line routing in all-optical networks. Proceedings of the 24th International Colloquium on Automata, Languages and Programming ICALP 97, LNCS, vol 1256. Springer, Berlin, pp 516–526
4. Beauquier B, Bermond J-C, Gargano L, Hell P, Perennes S, Vaccaro U (1997) Graph problems arising from wavelength-routing in all-optical networks. Proceedings of IPPS 97, Second Workshop on Optics and Computer Science (WOCS), Geneva
5. Caragiannis I, Ferreira A, Kaklamanis C, Pérennes S, Rivano H (2001) Fractional Path Coloring with Applications to WDM Networks. Proceedings of the 28th International Colloquium on Automata, Languages and Programming ICALP 01, LNCS, vol 2076. Springer, Berlin, pp 732–743
6. Caragiannis I, Kaklamanis C, Persiano P (1997) Bounds on optical bandwidth allocation on directed fiber tree topologies. Proceedings of IPPS 97, Second Workshop on Optics and Computer Science (WOCS), Geneva
7. Erlebach T, Jansen K (1997) Call scheduling in trees, rings and meshes. Proceedings of the 30th Hawaii International Conference on System Sciences HICSS-30, vol 1, IEEE Computer Society Press, Maui, pp 221–222
8. Erlebach T, Jansen K (1997) Scheduling of virtual connections in fast networks. Proceedings of the 4th Parallel Systems and Algorithms Workshop PASA 96, World Scientific Publishing, Jülich, pp 13–32
9. Erlebach T, Jansen K (2001) The complexity of path coloring and call scheduling. Theoret Comput Sci 255(1–2): 33–50
10. Erlebach T, Jansen K, Kaklamanis C, Mihail M, Persiano P (1999) Optimal Wavelength Routing on Directed Fiber Trees. Theoret Comput Sci 221(1–2):119–137
11. Erlebach T, Jansen K, Kaklamanis C, Persiano P (1998) An optimal greedy algorithm for wavelength allocation in directed tree networks. Proceedings of the DIMACS Workshop on Network Design: Connectivity and Facilities Location. DIMACS Series Disc Math Theoret Comput Sci AMS 40:117–129
12. Garey MR, Johnson DS, Miller GL, Papadimitriou CH (1980) The complexity of coloring circular arcs and chords. SIAM J Algebraic Discrete Methods 1(2):216–227
13. Green PE (1991) The future of fiber-optic computer networks. IEEE Comput 24(9):78–87
14. Holyer I (1981) The NP-completeness of edge-coloring. SIAM J Comput 10(4):718–720
15. Jansen K (1997) Approximation results for wavelength routing in directed trees. Proceedings of IPPS 97, Second Workshop on Optics and Computer Science (WOCS), Geneva
16. Kaklamanis C, Persiano P, Erlebach T, Jansen K (1997) Constrained bipartite edge coloring with applications to wavelength routing. Proceedings of the 24th International Colloquium on Automata, Languages and Programming ICALP 97, LNCS, vol 1256, Bologna. Springer, Berlin, pp 493–504
17. Kumar SR, Panigrahy R, Russel A, Sundaram R (1997) A note on optical routing on trees. Inf. Process. Lett 62:295–300
18. Kumar V, Schwabe EJ (1997) Improved access to optical bandwidth in trees. Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms SODA 97, New Orleans. pp 437–444
19. Mihail M, Kaklamanis C, Rao S (1995) Efficient access to optical bandwidth. Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp 548–557
20. Raghavan P, Upfal E (1994) Efficient routing in all-optical networks. Proceedings of the 26th Annual ACM Symposium on Theory of Computing STOC 94, ACM SIGACT, Monteal. ACM Press, New York, pp 134–143

Direct Global Optimization Algorithm

Donald R. Jones
General Motors Corp., Warren, USA

MSC2000: 65K05

Article Outline

Keywords
See also
References

Keywords

Global optimization; Black-box optimization; Nonsmooth optimization; Constraints; Deterministic optimization

For a *black-box global optimization* algorithm to be truly global, some effort must be allocated to global search, that is, search done primarily to ensure that potentially good parts of the space are not overlooked. On the other hand, to be efficient, some effort must also be placed on local search near the current best solution. Most algorithms either move progressively from global to local search (e. g., simulated annealing) or combine a fundamentally global method with a fundamentally local method (e. g., multistart, tunneling).

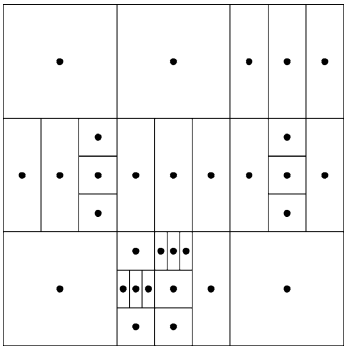
DIRECT introduces a new approach: in each iteration several search points are computed using all possible weights on local versus global search (how this is done will be made clear shortly). This approach eliminates the need for *'tuning parameters'* that set the balance between local and global search, resulting in an algorithm that is robust and easy-to-use.

DIRECT is especially valuable for *engineering optimization* problems. In these problems, the objective and constraint functions are often computed using time-consuming computer simulations, so there is a need to be efficient in the use of function evaluations. The problems may contain both continuous and integer variables, and the functions may be nonlinear, nonsmooth, and multimodal. While many algorithms address these problem features individually, DIRECT is one of the few that addresses them collectively. However, the versatility of DIRECT comes at a cost: the algorithm suffers from a curse of dimensionality that limits it to low-dimensional problems (say, no more than 20 variables).

The general problem solved by DIRECT can be written as follows:

$$\begin{cases} \min & f(x_1, \ldots, x_n) \\ \text{s.t.} & g_1(x_1, \ldots, x_n) \le 0, \\ & \quad \vdots \\ & g_m(x_1, \ldots, x_n) \le 0, \\ & \ell_i \le x_i \le u_i, \\ & x_i \in I \text{ integer}. \end{cases}$$

To prove convergence, we must assume that the objective and constraint functions are continuous in the neighborhood of the optimum, but the functions can otherwise be nonlinear, nondifferentiable, nonconvex, and multimodal. While DIRECT does not explicitly handle equality constraints, problems with equalities can often be rewritten as problems with *inequality constraints* (either by replacing the equality with an inequality that becomes binding in the solution, or by using the equalities to eliminate variables). The set I in the above problem is the set of variables that are restricted to *integer* values. DIRECT works best when the integer variables describe an ordered quantity, such as the number of teeth on a gear. It is less effective when the integer variables are categorical.



Direct Global Optimization Algorithm, Figure 1

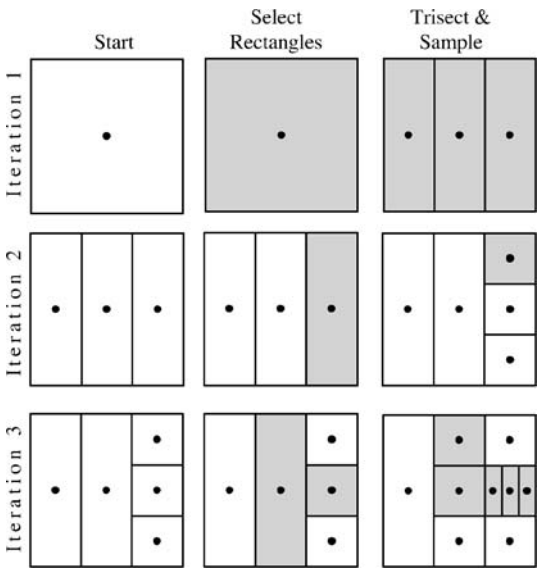
In what follows, we begin by describing how DIRECT works when there are no inequality and integer constraints. This basic version corresponds, with minor differences, to the originally published algorithm [2]. After describing the basic version, we then introduce extensions to handle inequality and integer constraints (this article is the first publication to document these extensions). We conclude with a step-by-step description of the algorithm.

The bounds on the variables limit the search to an n-dimensional hyper-rectangle. DIRECT proceeds by *partitioning* this rectangle into smaller rectangles, each of which has a 'sampled point' at its center, that is, a point where the functions have been evaluated. An example of such a partition for $n = 2$ is shown in Fig. 1.

We have drawn the rectangle as a square because later, whenever we measure distances or lengths, we will weight each dimension so that the original range $(u_i - \ell_i)$ has a weighted distance of one. Drawing the hyper-rectangle as a hyper-cube allows us to visualize relative lengths as they will be used in the algorithm.

Figure 2 shows the first three iterations of DIRECT on a hypothetical two-variable problem. At the start of each iteration, the space is partitioned into rectangles. DIRECT then selects one or more of these rectangles for further search using a technique described later. Finally, each selected rectangle is *trisected* along one of its long sides, after which the center points of the outer thirds are sampled. In this way, we sample two new points in the rectangle and maintain the property that every sampled point is at the center of a rectangle (this property would not be preserved if the rectangle were bisected).

At the beginning of iteration 1, there is only one rectangle (the entire space). The process of selecting



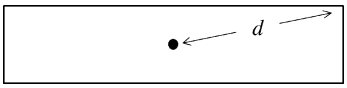
Direct Global Optimization Algorithm, Figure 2

rectangles is therefore trivial, and this rectangle is trisected as shown. At the start of iteration 2, the selection process is no longer trivial because there are three rectangles. In the example, we select just one rectangle, which is then trisected and sampled. At the start of iteration 3, there are 5 rectangles; in this example, two of them are selected and trisected.

The key step in the algorithm is the selection of rectangles, since this determines how search effort is allocated across the space. The trisection process and other details are less important, and we will defer discussion of them until later.

To motivate how DIRECT selects of rectangles, let us begin by considering the extremes of pure global search and pure local search. A pure global search strategy would select one of the biggest rectangles in each iteration. If this were done, all the rectangles would become small at about the same rate. In fact, if we always trisected one of the biggest rectangles, then after 3^{kn} function evaluations every rectangle would be a cube with side length 3^{-k}, and the sampled points would form a uniform grid. By looking everywhere, this pure global strategy avoids overlooking good parts of the space.

A pure local strategy, on the other hand, would sample the rectangle whose center point has the best objective function value. This strategy is likely to find good



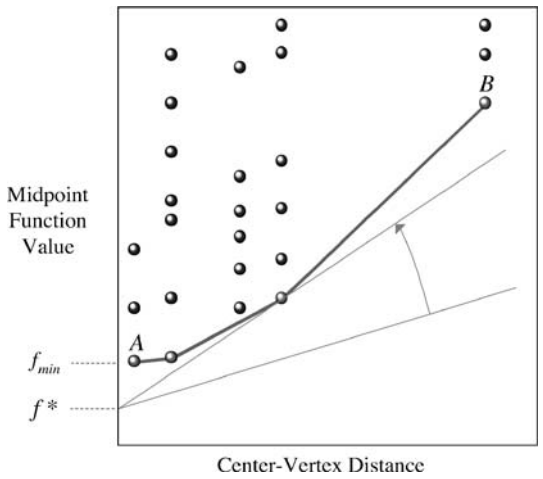
Direct Global Optimization Algorithm, Figure 3

solutions quickly, but it could overlook the rectangle that contains the global optimum (this would happen if the rectangle containing the global optimum had a poor objective function value at the center).

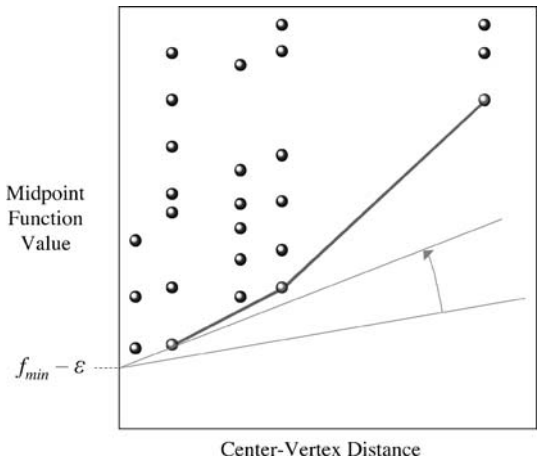
To select just one 'best' rectangle, we would have to introduce a tuning parameter that controlled the local/global balance. Unfortunately, the algorithm would then be extremely sensitive to this parameter, since the proper setting would depend on the (unknown) difficulty of the problem at hand.

DIRECT avoids tuning parameters by rejecting the idea of selecting just one rectangle. Instead, several rectangles are selected using all possible relative weightings of local versus global search. The idea of using all possible weightings may seem impractical, but with the help of a simple diagram this idea can actually be made quite intuitive. For this diagram, we will need a way to measure of the size of a rectangle. We will measure size using the distance between the center point and the vertices, as shown in Fig. 3.

With this measure of rectangle size, we can now turn our attention to Fig. 4 which shows how rectangles are selected. In the figure, each rectangle in the partition is represented by a dot. The horizontal coordinate of a dot is the size of the rectangle, measured by the center-vertex distance. The vertical coordinate is the function value at the midpoint of the rectangle. The dot labeled *A* represents the rectangle with the lowest function value, and so this would be the rectangle selected by a pure local strategy. Similarly, the dot labeled *B* represents one of the biggest rectangles, and so it would be selected by a pure global strategy. DIRECT selects not only these two extremes but also all the rectangles on the lower-right convex hull of the cloud of dots (the dots connected by the line). These rectangles represent 'efficient trade-offs' between local versus global search, in the sense that each of them is best for some relative weighting of midpoint function value and center-vertex distance. (We will explain the other lines in Fig. 4. shortly.)



Direct Global Optimization Algorithm, Figure 4



Direct Global Optimization Algorithm, Figure 5

One might think that the idea illustrated in Fig. 4 would extend naturally to the constrained case; that is, we would simply select any rectangle that was best for some weighting of objective value, center-vertex distance, and constraint values. Unfortunately, this does not work because it leads to excessive sampling in the infeasible region. However, as we explain next, there is an alternative way of thinking about the lower-right convex hull that does extend to the constrained case.

For the sake of the exposition, let us suppose for the moment that we know the optimal function value f^*. For the function to reach f^* within rectangle r, it would have to undergo a rate of change of at least $(f_r - f^*)/d_r$, where f_r is the function value at the midpoint of rectangle r and d_r is the center-vertex distance. This follows because the function value at the center is f_r and the maximum distance over which the function can fall to f^* is the center-vertex distance d_r. Intuitively, it seems 'more reasonable' to assume that the function will undergo a gradual change than to assume it will make a steep descent to f^*. Therefore, if only we knew the value f^*, a reasonable criterion for selecting a rectangle would be to choose the one that minimizes $(f_r - f^*)/d_r$.

Figure 4 shows a graphical way to find the rectangle that minimizes $(f_r - f^*)/d_r$. Along the vertical axis we show the current best function value, $f_{\min}$, as well as the supposed global minimum f^*. Now suppose we anchor a line at the point $(0, f^*)$ and slowly swing it upwards. When we first encounter a dot, the slope of the line will be precisely the ratio $(f_r - f^*)/d_r$, where r is the index of the rectangle corresponding to the encountered dot. Moreover, since this is the first dot touched by the line, rectangle r must be the rectangle that minimizes $(f_r - f^*)/d_r$.

Of course, in general we will not know the value of f^*. But we do know that, whatever f^* is, it satisfies $f^* \leq f_{\min}$. So imagine that we repeat the line-sweep exercise in Fig. 4 for all values of f^* ranging from $f_{\min}$ to $-\infty$. How many rectangles could be selected? Well, with a little thought, it should be clear that the set of dots that can be selected via these line sweeps is precisely the lower-right convex hull of the dots.

This alternative approach to deriving the lower-right convex hull suggests a small but important modification to the selection rule. In particular, to prevent DIRECT from wasting function evaluations in pursuit of very small improvements, we will insist that the value of f^* satisfy $f^* \leq f_{\min} - \epsilon$. That is, we are only interested in selecting rectangles where it is reasonable that we can find a 'significantly better' solution. A natural value of ϵ would be the desired accuracy of the solution. In our implementation, we have set $\epsilon = \max(10^{-4}|f_{\min}|, 10^{-8})$.

As shown in Fig. 5, the implication of this modification is that some of the smaller rectangles on the lower-right convex hull may be skipped. In fact, the smallest rectangle that will be selected is the one chosen when $f^* = f_{\min} - \epsilon$.

The version of DIRECT described so far corresponds closely to the originally published version [2].

The only difference is that, in the original version, a selected rectangle was trisected not just on a single long side, but rather on all long sides. This approach eliminated the need to arbitrarily select a single long side when there were more than one and, as a result, it added an element of robustness to the algorithm. Experience has since shown, however, that the robustness benefit is small and that trisecting on a single long side (as here) accelerates convergence in higher dimensions.

Let us now consider how the rectangle selection procedure can be extended to handle inequality constraints. The key to handling constraints in DIRECT is to work with an *auxiliary function* that combines information on the objective and constraint functions in a special manner. To express this auxiliary function, we will need some additional notation. Let g_{rj} denote the value of constraint j at the midpoint of rectangle r. In addition, let $c_1, \ldots, c_m$ be positive weighting coefficients for the inequality constraints (we will discuss how these coefficients are computed later). Finally, for the sake of the exposition, let us again suppose that we know the optimal function value f^*. The auxiliary function, evaluated at the center of rectangle r, is then as follows:

$$\max(f_r - f^*, 0) + \sum_{j=1}^{m} c_j \max(g_{rj}, 0)$$

The first term of the auxiliary function exacts a penalty for any deviation of the function value f_r above the global minimum value f^*. Note that, in a constrained problem, it is possible for f_r to be less than f^* by violating the constraints; due to the maximum operator, the auxiliary function gives no credit for values of f_r below f^*. The second term in the auxiliary function is a sum of weighted constraint violations. Clearly, the lowest possible value of the auxiliary function is zero and occurs only at the global minimum. At any other point, the auxiliary function is positive either due to suboptimality or infeasibility.

This auxiliary function is not a *penalty function* in the standard sense. A standard penalty function would be a weighted sum of the objective function and constraint violations; it would not include the value f^* since this value is generally unknown. Moreover, in the standard approach, it is critical that the penalty coefficients be sufficiently large to prevent the penalty function from being minimized in the infeasible region. This is not true for our auxiliary function: as long as f^* is the optimal function value, the auxiliary function is minimized at the global optimum for *any* positive constraint coefficients.

For the global minimum to occur in rectangle r, the auxiliary function must fall to zero starting from its (positive) value at the center point. Moreover, the maximum distance over which this change can occur is the center-vertex distance d_r. Thus, to reach the global minimum in rectangle r, the auxiliary function must undergo a minimum rate of change, denoted $h_r(f^*)$, given by

$$h_r(f^*) = \frac{\max(f_r - f^*, 0) + \sum_{j=1}^{m} c_j \max(g_{rj}, 0)}{d_r}.$$

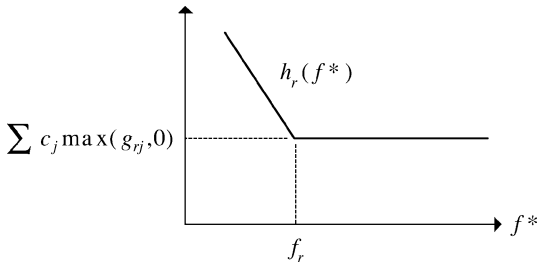
Since it is more reasonable to expect gradual changes than abrupt ones, a reasonable way to select a rectangle would be to select rectangle that minimizes the rate of change $h_r(f^*)$. Of course, this is impractical because we generally will not know the value f^*. Nevertheless, it is possible to select the set of rectangles that minimize $h_r(f^*)$ for *some* $f^* \leq f_{\min} - \epsilon$. This is how we select rectangles with constraints—assuming a feasible point has been found so that $f_{\min}$ is well-defined (we will show how this is implemented shortly). If no feasible point has been found, we simply select the rectangle that minimizes

$$\frac{\sum_{j=1}^{m} c_j \max(g_{rj}, 0)}{d_r}.$$

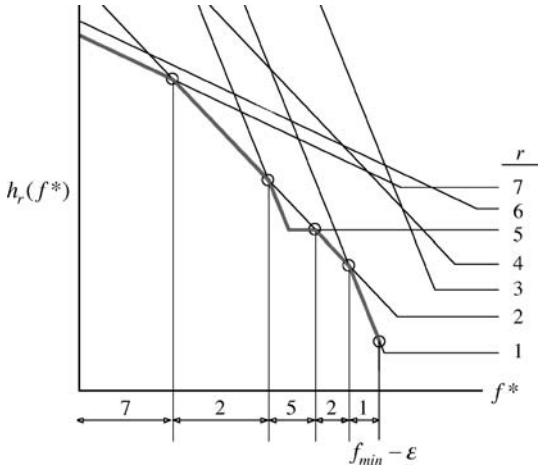
That is, we select the rectangle where the weighted constraint violations can be brought to zero with the least rate of change.

To implement this selection rule, it is again helpful to draw a diagram. This new selection diagram is based on plotting the rate-of-change function $h_r(f^*)$ as a function of f^*. Figure 6 illustrates this function. For values of $f^* \geq f_r$, the first term in the numerator of $h_r(f^*)$ is zero, and so $h_r(f^*)$ is constant. As f^* falls below f_r, however, the $h_r(f^*)$ increases, because we now exact a penalty for f_r being above the supposed global minimum f^*. The slope of $h_r(f^*)$ function to the left of f_r is $-1/d_r$.

Figure 7 superimposes, in one diagram, the rate-of-change functions for a hypothetical set of seven rectangles. For a particular value of f^*, we can visually find the rectangle that minimizes $h_r(f^*)$ by starting at the point



Direct Global Optimization Algorithm, Figure 6



Direct Global Optimization Algorithm, Figure 7

(f^*, 0) along the horizontal axis and moving vertically until we first encounter a curve. What we want, however, is the set of *all* rectangles that can be selected in this way using *any* $f^* \leq f_{\min} - \epsilon$. This set can be found as follows (see Fig. 7). We start with $f^* = f_{\min} - \epsilon$ and move upwards until we first encounter a curve for some rectangle. We note this rectangle and follow its curve to the left until it intersects the curve for another rectangle (these intersections are circled in Figure 7). When this happens, we note this other rectangle and follow its curve to the left. We continue in this way until we find a curve that is never intersected by another one. This procedure will identify all the $h_r(f^*)$ functions that participate in the lower envelope of the curves to the left of $f_{\min} - \epsilon$. The set of rectangles found in this way is the set selected by DIRECT.

Along the horizontal axis in Fig. 7, we identify ranges of f^* values for which different rectangles have the lowest value of $h_r(f^*)$. As we scan from $f_{\min} - \epsilon$ to the left, the rectangles that participate in the lower envelope are 1, 2, 5, 2, and 7. This example illustrates that it is possible to encounter a curve more than once (here rectangle 2), and care must be taken not to double count such rectangles. It is also possible for some curves to coincide along the lower envelope, and so be 'tied' for the least rate of change (this does not happen in Fig. 7). In such cases, we select all the tied rectangles.

Tracing the lower envelope in Fig. 7 is not computationally intense. To see this, note that each selected rectangle corresponds to a curve on the lower envelope, and for each such curve the work we must do is to find the intersection with the next curve along the lower envelope. Finding this next intersection requires computing the intersection of the current curve with all the other curves. It follows that the work required for each selected rectangle (and hence for every two sampled points) is only on the order of the total number of rectangles in the partition.

The tracing of the lower envelope can also be accelerated by some pre-processing. In particular, it is possible to quickly identify rectangles whose curves lie completely above other curves. For example, in Fig. 7, curve 3 lies above curve 1, and curve 4 lies above curve 2. These curves cannot possibly participate in the lower envelope, and so they can be deleted from consideration before the lower envelope is traced.

It remains to explain how the constraint coefficients $c_1, \ldots, c_m$ are computed, as well as a few other details about trisection and the handling of integer variables. We will cover these details in turn, and then bring everything together into a step-by-step description of the algorithm.

To understand how we compute the constraint coefficient c_j, suppose for the moment that we knew the average rate of change of the objective function, denoted a_0, and the average rate of change of constraint j, denoted a_j. Furthermore, suppose that at the center of a rectangle we have $g_j > 0$. At the average rate of change of constraint j, we would have to move a distance equal to g_j/a_j to get rid of the constraint violation. If during this motion the objective function got worse at its average rate of change, it would get worse by a_0 times the distance, or $a_0(g_j/a_j) = (a_0/a_j)\, g_j$. Thus we see that the ratio a_0/a_j provides a way of converting units of constraint violation into potential increases in the objective function. For this reason, we will set $c_j = a_0/a_j$.

The average rates of change are estimated in a very straightforward manner. We maintain a variable s_0 for the sum of observed rates of change of the objective function. Similarly we maintain variables $s_1, \ldots, s_m$ for the sum of observed rates of change for each of the m constraints. All of these variables are initialized to zero at the start of the algorithm and updated each time a rectangle is trisected. Let x^{mid} denote the midpoint of the parent rectangle and let x^{left} and x^{right} denote the midpoints of the left and right child rectangles after trisection. The variables are updated as follows:

$$s_0 = s_0 + \sum_{\text{child=left}}^{\text{right}} \frac{\left|f(x^{\text{child}}) - f(x^{\text{mid}})\right|}{\left\|x^{\text{child}} - x^{\text{mid}}\right\|}$$

$$s_j = s_j + \sum_{\text{child=left}}^{\text{right}} \frac{\left|g_j(x^{\text{child}}) - g_j(x^{\text{mid}})\right|}{\left\|x^{\text{child}} - x^{\text{mid}}\right\|}.$$

Now the average rates of change are $a_0 = s_0/N$ and $a_j = s_j/N$, where N is the number of rates of change accumulated into the sums. It follows that

$$\frac{a_0}{a_j} = \frac{\frac{s_0}{N}}{\frac{s_j}{N}} = \frac{s_0}{s_j}.$$

We may therefore compute c_j using

$$c_j = \frac{s_0}{\max(s_j, 10^{-30})},$$

where we use the maximum operator in the denominator to prevent division by zero.

So far we have said that we will always trisect a rectangle along one of its long sides. However, as shown in Fig. 2, several sides may be tied for longest, and so we need some way to break these ties. Our tie breaking mechanism is as follows. We maintain counters t_i ($i = 1, \ldots, n$) for how many times we have split along dimension i over the course of the entire search. These counters are initialized to zero at the beginning of the algorithm, and counter t_i is incremented each time a rectangle is trisected along dimension i. If we select a rectangle that has several sides tied for being longest, we break the tie in favor of the side with the lowest t_i value. If several long sides are also tied for the lowest t_i value, we break the tie arbitrarily in favor of the lowest-indexed dimension. This tie breaking strategy has the effect of equalizing the number of times we split on the different dimensions.

Let us now turn to the calculation of the center-vertex distance. Recall that we measure distance using a weighted metric that assigns a length of one to the initial range of each variable ($u_i - \ell_i$). Each time a rectangle is split, the length of that side is then reduced by a factor of 1/3. Now consider a rectangle that has been trisected T times. Let $j = \text{mod}(T, n)$, so that we may write $T = kn + j$ where $k = (T - j)/n$. After the first kn trisections, all of the n sides will have been trisected k times and will therefore have length 3^{-k}. The remaining j trisections will make j of the sides have length $3^{-(k+1)}$, leaving $n - j$ sides with length 3^{-k}. Simple algebra then shows that the distance d from the center to the vertices is given by

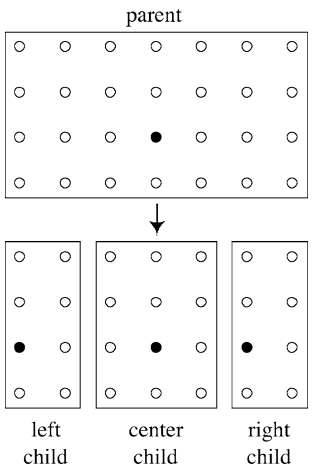
$$d = \frac{3^{-k}}{2}\left(\frac{j}{9} + n - j\right)^{0.5}.$$

(This is not obvious, but can be easily verified.)

The handling of integer variables is amazingly simple, involving only minor changes to the trisection routine and to the way the midpoint of a rectangle is defined. For example, consider an integer variable with range [1, 8]. We could not define the midpoint to be 4.5 because this is not an integer. Instead, we will use the following procedure. Suppose the range of a rectangle along an integer dimension is $[a, b]$, with both a and b being integers. We will define the 'midpoint' as $\lfloor (a + b)/2 \rfloor$, that is, it is the floor of algebraic average (the *floor* of z, denoted $\lfloor z \rfloor$, is the greatest integer less than or equal to z).

To trisect along the integer dimension, we first compute $\Delta = \lfloor (b - a + 1)/3 \rfloor$. If $\Delta \geq 1$, then after trisection the left child will have the range $[a, a + \Delta - 1]$, the center child will have the range $[a + \Delta, b - \Delta]$, and the right child will have range $[b - \Delta + 1, b]$. If $\Delta = 0$, then the integer side must have a range of two (i. e., $b = a + 1$). In this case, the center child will have the range $[a, a]$ the right child will have the range $[b, b]$, and there will be no left child. This procedure maintains the property that the midpoint of the parent rectangle always becomes the midpoint of the center child. As an example, Fig. 8 shows how a rectangle would be trisected when there are two integer dimensions. In the figure, the circles represent possible integer combinations, and the filled circles represent the midpoints.

Integer variables introduce three other complications. The first, which may be seen in Fig. 8, is that the



Direct Global Optimization Algorithm, Figure 8

sampled point may not be in the true geometric center of the rectangle. As a result, the center-vertex distance will not be unique but will vary from vertex to vertex. We ignore this detail and simply use the formula given above for the continuous case, which only depends upon the number of times a rectangle has been trisected.

The second complication concerns how we define a 'long' side. In the continuous case, the length of a side is directly related to the number of times it has been trisected along that dimension. Specifically, if a rectangle has been split k times along some side, then the side length will be 3^{-k} (recall that we measure distance relative to the original range of each variable). In the continuous case, therefore, the set of long sides is the same as the set of sides that have been split upon the least. When there are integers, however, the side lengths will no longer be multiples of 1/3. To keep things simple, however, we ignore this and continue to define a 'long' side as one that has been split upon the least. However, if an integer side has been split so many times that its side length is zero (i. e., the range contains a single integer), then this side will not be considered long.

The third and final complication is that, if *all* the variables are integer, then it is possible for a rectangle to be reduced to a single point. If this happens, the rectangle would be fathomed; hence, it should be ignored in the rectangle selection process in all subsequent iterations.

DIRECT stops when it reaches a user-defined limit on function evaluations. It would be preferable, of course, to stop when we have achieved some desired accuracy in the solution. However, for black-box problems where we only assume continuity, better stopping rules are hard to develop.

As for convergence, it is easy to show that, as f^* moves to $-\infty$, DIRECT will select one of the largest rectangles. Because we always select one of the largest rectangles, and because we always subdivide on a long side, every rectangle will eventually become very small and the sampled points will be dense in the space. Since we also assume the functions are continuous in the neighborhood of the optimum, this insures that we will get within any positive tolerance of the optimum after a sufficiently large number of iterations.

Although we have now described all the elements of DIRECT, our discussion has covered several pages, and so it will be helpful to bring everything together in a step-by-step description of the algorithm.

1) Initialization.
   Sample the center point of the entire space. If the center is feasible, set $x_{\min}$ equal to the center point and $f_{\min}$ equal to the objective function value at this point. Set $s_j = 0$ for $j = 0, \ldots, m$; $t_i = 0$ for $i = 1, \ldots, n$; and neval = 1 (function evaluation counter). Set maxeval equal to the limit on the number of function evaluations (stopping criterion).
2) Select rectangles.
   Compute the c_j values using the current values of s_0 and s_j, $j = 1, \ldots, m$. If a feasible point has *not* been found, select the rectangle that minimizes the rate of change required to bring the weighted constraint violations to zero. On the other hand, if a feasible point has been found, identify the set of rectangles that participate in the lower envelope of the $h_r(f^*)$ functions for some $f^* \leq f_{\min} - \epsilon$. A good value for ϵ is $\epsilon = \max(10^{-4}\ |f_{\min}|, 10^{-8})$. Let S be the set of selected rectangles.
3) Choose any rectangle $r \in S$.
4) Trisect and sample rectangle r.
   Choose a splitting dimension by identifying the set of long sides of rectangle r and then choosing the long side with the smallest t_i value. If more than one side is tied for the lowest t_i value, choose the one with the lowest-dimensional index. Let i be the resulting splitting dimension. Note that a 'long side'

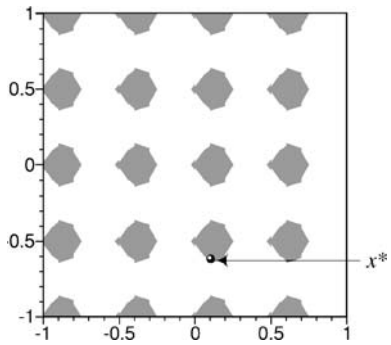
is defined as a side that has been split upon the least and, if integer, has a positive range. Trisect rectangle r along dimension i and increment t_i by one. Sample the midpoint of the left third, increment neval by one, and update $x_{\min}$ and $f_{\min}$. If neval = maxeval, go to Step 7. Otherwise, sample the midpoint of the right third, increment *neval* by one, and update $x_{\min}$ and $f_{\min}$ (note that there might not be a right child when trisecting on an integer variable). Update the s_j $j = 0, \ldots, m$. If all n variables are integer, check whether a child rectangle has been reduced to a single point and, if so, delete it from further consideration. Go to Step 5.

5) Update S.
   Set $S = S - \{r\}$. If S is not empty, go to Step 3. Otherwise go to Step 6.
6) Iterate.
   Report the results of this iteration, and then go to Step 2.
7) Terminate.
   The search is complete. Report $x_{\min}$ and $f_{\min}$ and stop.

The results of DIRECT are slightly sensitive to the order in which the selected rectangles are trisected and sampled because this order affects the t_i values and, hence, the choice of splitting dimensions for other selected rectangles. In our current implementation, we select the rectangles in Step 3 in the same order that they are found as we scan the lower envelope in Fig. 7 from $f^* = f_{\min} - \epsilon$ towards $f^* = -\infty$.

On the first iteration, all the s_j will be zero in Step 2 and, hence, all the c_j will be zero when computed using $c_j = s_0/\max(s_j, 10^{-30})$. Thus, in the beginning the constants c_j will not be very meaningful. This is not important, however, because on the first iteration there is only one rectangle eligible for selection (the entire space), and so the selection process is trivial. As the iterations proceed, the s_j will be based on more observations, leading to more meaningful c_j constants and better rectangle selections.

When there are no inequality constraints, the above step-by-step procedure reduces to the basic version of DIRECT described earlier. To see this, note that, when there are no constraints, every point is feasible and so $f_r \geq f_{\min} \geq f^*$ for all rectangles r. This fact, combined with the lack of any constraint violations, means that the $h_r(f^*)$ function given earlier reduces to $(f_r - f^*)/d_r$,



Direct Global Optimization Algorithm, Figure 9

which is precisely the rate-of-change function we minimized in the unconstrained version. Thus, in the unconstrained case, tracing the lower envelope in Fig. 7 identifies the same rectangles as tracing the lower-right convex hull in Fig. 5.

We will illustrate DIRECT on the following two-dimensional test function:

$$\begin{cases} \min & f(x_1, x_2) \\ \text{s.t.} & g(x_1, x_2) \leq 0 \\ & -1 \leq x_1, x_2 \leq +1, \end{cases}$$

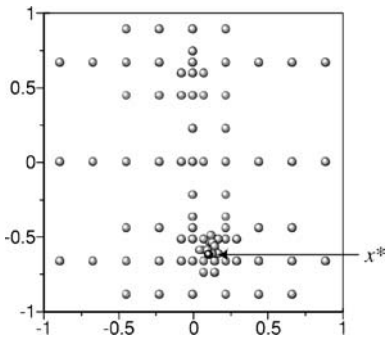
where

$$f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + \left(-4 + 4x_2^2\right) x_2^2,$$

$$g(x_1, x_2) = -\sin(4\pi x_1) + 2\sin^2(2\pi x_2).$$

We call this problem the *Gomez #3 problem* since it was listed as the third test problem in an article by S. Gomez and A. Levy [1]. The global minimum of the Gomez #3 problem occurs at the point (0.109, − 0.623) where the function value is − 0.9711. The problem is difficult because the feasible region consists of many disconnected, approximately circular parts, giving rise to many local optima (see Fig. 9).

For this test function, DIRECT gets within 1% of the optimum after 89 function evaluations and within 0.01% after 513 function evaluations. The first 89 sampled points are shown in Fig. 10. For comparison, the tunneling algorithm of Gomez and Levy [1] converged using an average of 1053 objective function evaluations and 1873 constraint evaluations (averaged over 20 random starting points). One reason DIRECT converges



Direct Global Optimization Algorithm, Figure 10

quickly is that it searches both globally and locally during each iteration; as a result, as soon as the global part of the algorithm finds the basin of convergence of the optimum, the local part of the algorithm automatically exploits it.

In this example, DIRECT quickly gets close to the optimum but takes longer to achieve a high degree of accuracy. This suggests that the best performance would be obtained by combining DIRECT with a good local optimizer. The simplest way to do this is to run DIRECT for a predetermined number of function evaluations and then use the resulting solution as a starting point for a local optimization. While straightforward, this approach is highly sensitive to the number of function evaluations used in the global phase with DIRECT. If one uses too few function evaluations, DIRECT might not discover the basin of convergence of the global minimum.

To ensure that the global optimum is eventually found, we must somehow return to the global phase after we have performed a local search. One way of doing this is as follows. We start the local optimizer the very first time a feasible point is found (or perhaps after a minimum initial phase of 50–100 evaluations). After the local finishes, we return to DIRECT. However, DIRECT does not proceed the same as it would have without the local optimizer. Instead, the search will be more global, because the local optimizer will have reduced the value of $f_{\min}$ (which affects rectangle selection). DIRECT will now be looking for a point that improves upon the local solution—in effect, it will be looking for the basin of convergence of a better minimum. If DIRECT finds such an improving point, then we run a local search from this point and again return to DIRECT. This process continues until we reach a predetermined limit on the total number of function evaluations (for both DIRECT and the local optimizer). Used in this way, DIRECT becomes an intelligent routine for selecting starting points for the local optimizer.

While DIRECT works well on the Gomez #3 problem and on test functions reported in [2], the algorithm is not without its disadvantages. For example, DIRECT's use of a space-partitioning approach requires the user to have relatively tight lower and upper bounds on all the variables. DIRECT will perform miserably if one specifies wide bounds such as $[-10^{30}, +10^{30}]$. The space-partitioning approach also limits the algorithm to low-dimensional problems (say, less than 20). While integer variables are handled, they must be ordered, such as the number of gear teeth, since only then can we expect the function value at a rectangle's midpoint to be indicative of what the function is like in the rest of the rectangle. Another limitation is that equality constraints are not handled. Finally, the stopping criterion—a limit on function evaluations—is weak.

The advantages of DIRECT, however, are considerable. The algorithm can handle nonsmooth, nonlinear, multimodal, and even discontinuous functions (as long as the discontinuity is not close to the global optimum). The algorithm works well in the presence of noise, since a small amount of noise usually has little impact on the set of selected rectangles until late in the search. Computational overhead is low, and the algorithm can exploit parallel processing because it generates several new points per iteration. Based on the comparisons in [2], the algorithm also appears to be efficient in terms of the number of function evaluations required to get close to the global minimum. But the most important advantage of DIRECT stems from its unique approach to balancing local and global search—the simple idea of not sampling just one point per iteration, but rather sampling several points using all possible weightings of local versus global search. This approach leads to an algorithm with no tuning parameters, making the algorithm easy-to-use and robust.

See also

► αBB Algorithm
► Continuous Global Optimization: Applications

- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Differential Equations and Global Optimization
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Binary Star Astronomy
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization Using Space Filling
- ▶ Topology of Global Optimization

References

1. Gomez S, Levy A (1982) The tunneling method for solving the constrained global optimization problem with several non-connected feasible regions. In: Lecture Notes Math, vol 909. Springer, Berlin, 34–47
2. Jones DR, Perttunen CD, Stuckman BE (1993) Lipschitzian optimization without the Lipschitz constant. J Optim Th Appl 73(1):157–181

Direct Search Luus–Jaakola Optimization Procedure

LJ Optimization Procedure

REIN LUUS
Department Chemical Engineering,
University Toronto, Toronto, Canada

MSC2000: 93-XX

Article Outline

Keywords
Optimization Problem
Handling Equality Constraints
Use of LJ Optimization Procedure for High-Dimensional Problems
Determination of Region Size
See also
References

Keywords

Optimization; Direct search; Nonseparable problem optimization; Sensitivity; Global optimization

Direct search optimization procedures are attractive because of the ease with which they can be used. Optimization procedures where auxiliary functions such as gradients are not calculated are desirable for problems where discontinuous functions are encountered, or where numerous constraints make the calculation and use of gradients very difficult in searching for the global optimum. The reliability of getting to the vicinity of the global optimum is an additional feature that makes the use of direct search optimization an attractive means of optimization.

The need for an efficient and easy to use optimization procedure was illustrated in [1], in attempting to obtain the best weighting factors in a Liapunov function used for time suboptimal control of a linear gas absorber. Although at that time the best optimization procedure for that problem was the hill-climbing procedure due to H.H. Rosenbrock [35], the method encountered difficulties in establishing the global optimum. In the 1970s a large number of direct search optimization procedures were introduced. One such method is due to R. Luus and T.H.I. Jaakola [29], which has been called in the literature by numerous authors as the *LJ optimization procedure*. The method is based on using a number of randomly chosen test points over some region and contracting the region after every iteration, always starting the iteration with the best point found from the previous iteration as the center of the region. The ease of programming and the ease with which inequality constraints can be handled make this direct search procedure attractive.

Optimization Problem

We consider the problem of minimizing the *performance index* or *cost function*

$$I = f(x_1, \ldots, x_n) \tag{1}$$

subject to p inequality constraints

$$g_j(x_1, \ldots, x_n) \geq 0, \quad j = 1, \ldots, p, \tag{2}$$

through the appropriate choice of $x_1, \ldots, x_n$. The direct search optimization procedure suggested in [29] involves only three steps:

- Choose a number of points in the n-dimensional space through the equation

$$\mathbf{x} = \mathbf{x}^* + \mathbf{D}\mathbf{r}, \tag{3}$$

where $\mathbf{D}$ is a diagonal matrix with diagonal elements chosen at random between -1 and $+1$, and $\mathbf{r}$ is the region size vector.

- Check the feasibility of each point with respect to (2), and for each feasible point evaluate the performance index I in (1).
- At the end of each iteration, $\mathbf{x}^*$ is replaced by the best feasible value of $\mathbf{x}$ obtained in Step 2, and the region size vector $\mathbf{r}$ is reduced in size by

$$\mathbf{r}^{(j+1)} = \gamma \mathbf{r}^{(j)}, \tag{4}$$

where γ is a contraction factor such as 0.95. This procedure is continued for a number of iterations and the results are examined.

If adequate convergence is not obtained, then the procedure can be repeated by carrying out another pass, using the information obtained from the previous pass. This optimization procedure enabled several difficult optimization problems to be solved in the original paper [29], and provided a means to solve a wide variety of problems in optimal control, such as time suboptimal control [9,8,7] and gave good approximation to optimal control by providing a means of obtaining the elements for the feedback gain matrix. The LJ optimization procedure was found very useful for stabilizing systems through shifting of poles [30] and testing stabilizability of linear systems [13]. Research was done to improve the likelihood of getting the global optimum for nonunimodal systems [37], but even without any modification, the reliability of the LJ procedure was found to be very good [38], even for the difficult bifunctional catalyst blend problem [26]. Therefore, the LJ optimization procedure could be used effectively for optimization of complex systems such as heat exchanger networks [17], a transformer design problem [36], design of structural columns in such a way that the amount of material would be minimized [3], and problems dealing with metallurgical processes [34]. The simplicity of the method was illustrated by the computer program given in its entirety in reference [17].

When the variables are restricted to be integers, special procedures may be necessary [12], since we cannot simply search on integer values to get the global optimum. Thus the scope of problems where LJ optimization procedure has been successfully applied is quite wide. In parameter estimation, N. Kalogerakis and Luus [6] found that by LJ optimization reliable estimates could be obtained for parameters in very few iterations, so that these estimates could then be used as starting values for quadratically convergent Gauss–Newton method, without having to worry about nonconvergence. In model reduction the LJ method has been found useful to match the reduced system's Nyquist plot to that of the original system [15], or used directly in time domain [40]. LJ optimization procedure has also been used successfully in model reduction in sampled-data systems [39] and is illustrated with several examples in [23].

Handling Equality Constraints

Suppose that in addition to the inequality constraints in (2), we also have m equality constraints

$$h_i(x_1, \ldots, x_n) = 0, \quad i = 1, \ldots, m, \tag{5}$$

where these equality constraints are 'difficult' in the sense that they can not be used to solve for some particular variable.

Although a two-pass method to deal with equality constraints [10] was effective to solve optimization problems involving recycle streams [11], the general approach for handling *equality constraints* with LJ optimization procedure was not solved satisfactorily, until it was shown [4] that *penalty functions* can be used very effectively in direct search optimization. The work was extended in [33], and now it appears that the use of a quadratic penalty function incorporating a shifting term is the best way of dealing with difficult equality constraints [19]. We consider the *augmented performance index*

$$J = I + \theta \sum_{i=1}^{m} (h_i - s_i)^2, \tag{6}$$

where a shifting term s_i is introduced for each equality constraint. To solve the optimization problem, LJ optimization procedure is used in a multipass fashion, where at the beginning of each pass consisting of a number of iterations, the region sizes are restored to a fraction of the sizes used at the beginning of the previous pass. The shifting terms s_i are updated after every pass simply by adjusting the values at the beginning of pass $(q+1)$ based on the deviation of the left-hand side

in (5) from zero; i. e.,

$$s_i^{(q+1)} = s_i^{(q)} - h_i, \quad i = 1, \ldots, m, \tag{7}$$

where q is the pass number. Upon optimization the product $-2\,\theta\,s_i$ gives the Lagrange multiplier associated with the ith equality constraint, yielding useful sensitivity information. Full details are given in [19]. This approach to dealing with equality constraints was used in [31] in using *iterative dynamic programming* (IDP) to solve an optimal control problem where the final state was specified, and in [28] where the volume of the fed-batch reactor was specified at the final time.

Another approach to deal with equality constraints is to solve the algebraic equations at each iteration by grouping the equations [21]. For several optimization problems this approach yielded very rapid convergence to the global optimum [22].

Use of LJ Optimization Procedure for High-Dimensional Problems

In [2] it was found that LJ optimization can be used quite effectively to solve optimal control problems, where the system is divided into a number of time stages. This approach was used in [27] to solve a very difficult optimal control problem involving the determination of the optimum drug delivery schedule to minimize the tumor size at the end of 12 weeks. The problem was broken into 84 time stages, each consisting of a single day. In spite of the *state constraints* and *discontinuous functions*, this 84-dimensional optimization problem was solved successfully on a personal computer in reasonable computation time. Especially now that the personal computers are much faster, such a problem is considerably easier to solve. To solve high-dimensional problems a multipass method was used for LJ optimization where after a pass, the region would be restored to a value smaller than used at the beginning of the previous pass and the procedure was repeated. In the case of the *cancer chemotherapy* problem, the problem required a number of runs for successful solution [27].

Determination of Region Size

One of the problems that was outstanding for the LJ optimization procedure was how to choose the region size vector $\mathbf{r}$ effectively at the beginning of the iterations, especially when a multipass procedure was used. This problem was recently solved in [20], by suggesting that the initial region size be determined by the extent of the variation of the variable during the previous pass. With the use of reliable values for the region size at the beginning of each pass in a multipass run, the computational effort is decreased quite substantially. For example, when we consider the *nonseparable optimization problem* introduced in [32], where we have a system described by three difference equations:

$$x_1(k+1) = \frac{x_1(k)}{1 + 0.01u_1(k)(3 + u_2(k))},$$

$$x_2(k+1) = \frac{x_2(k) + u_1(k)x_1(k+1)}{1 + u_1(k)(1 + u_2(k))},$$

$$x_3(k+1) = \frac{x_3(k)}{1 + 0.01u_2(k)(1 + u_3(k))},$$

with the initial condition

$$\mathbf{x}(0) = [2 \quad 5 \quad 7]^\top.$$

The control variables are constrained by

$$0 \le u_1(k) \le 4,$$

$$0 \le u_2(k) \le 4,$$

$$0 \le u_3(k) \le 0.5.$$

The performance index to be minimized is

$$I = x_1^2(P) + x_2^2(P) + x_3^2(P) + \left[\left(\sum_{k=1}^{P} x_1^2(k-1) + x_2^2(k-1) + 2u_3^2(k-1)\right) \cdot \left(\sum_{k=1}^{P} x_3^2(k-1) + 2u_1^2(k-1) + 2u_2^2(k-1)\right)\right]^{\frac{1}{2}}$$

where P is the number of stages. When P is taken as 100, then we have a 300 variable optimization problem, because at each stage there are three control variables

to be determined. Without the use of a reliable way of determining the region sizes over which to take the control variables, the problem is very difficult, but with the method suggested in [20] the problem was solved quite readily by the LJ optimization procedure by using 100 random points per iteration and 60 passes, each consisting of 201 iterations, to yield $I = 258.3393$. Although the computational requirements appear enormous, the actual computation time was less than 20 minutes on a Pentium-120 personal computer [20], which corresponds to less than one minute on the Pentium4/2.4 GHz personal computer. This value of the performance index is very close to the value $I = 258.3392$ obtained by use of iterative dynamic programming [18]. To solve this problem, IDP is much more efficient in spite of the nonseparability of the problem, because in IDP the problem is solved as a 3 variable problem over 100 stages, rather than a 300 variable optimization problem. Therefore, the LJ procedure is useful in checking the optimal control policy obtained by some other method. Here, the control policies obtained by IDP and LJ optimization procedure are almost identical, where a sudden change at around stage 70 occurs in the control variables u_1 and u_2. Therefore, LJ optimization procedure is ideally suited for checking results obtained by other methods, especially when the optimal control policy differs from what is expected, as is the case with this particular example.

Recently it was shown that the convergence of the LJ optimization procedure in the vicinity of the optimum can be improved substantially by incorporating a simple line search to choose the best center point for a subsequent pass [24]. For a typical model reduction problem, to reach the global optimum the computation time was reduced by a factor of four when the line search was incorporated. Due to its simplicity, the LJ optimization procedure can be programmed very easily. Computational experience with numerous optimization problems has shown that the method has high reliability of obtaining the *global optimum*, so the LJ optimization procedure provides a very good means of obtaining the optimum for very complex problems.

See also

► Interval Analysis: Unconstrained and Constrained Optimization

References

1. Bennett HW, Luus R (1971) Application of numerical hill-climbing in control of systems via Liapunov's direct method. Canad J Chem Eng 49:685–690
2. Bojkov B, Hansel R, Luus R (1993) Application of direct search optimization to optimal control problems. Hungarian J Industr Chem 21:177–185
3. Dinkoff B, Levine M, Luus R (1979) Optimum linear tapering in the design of columns. Trans ASME 46:956–958
4. Hartig F, Keil FJ, Luus R (1995) Comparison of optimization methods for a fed-batch reactor. Hungarian J Industr Chem 23:141–148
5. Jaakola THI, Luus R (1974) A note on the application of nonlinear programming to chemical-process optimization. Oper Res 22:415–417
6. Kalogerakis N, Luus R (1982) Increasing the size of region of convergence for parameter estimation. Proc. 1982 Amer. Control Conf., 358–362
7. Luus R (1974) Optimal control by direct search on feedback gain matrix. Chem Eng Sci 29:1013–1017
8. Luus R (1974) A practical approach to time-optimal control of nonlinear systems. Industr Eng Chem Process Des Developm 13:405–408
9. Luus R (1974) Time-optimal control of linear systems. Canad J Chem Eng 52:98–102
10. Luus R (1974) Two-pass method for handling difficult equality constraints in optimization. AIChE J 20:608–610
11. Luus R (1975) Optimization of multistage recycle systems by direct search. Canad J Chem Eng 53:217–220
12. Luus R (1975) Optimization of system reliability by a new nonlinear integer programming procedure. IEEE Trans Reliabil 24:14–16
13. Luus R (1975) Solution of output feedback stabilization and related problems by stochastic optimization. IEEE Trans Autom Control 20:820–821
14. Luus R (1976) A discussion on optimization of an alkylation process. Internat J Numer Methods in Eng 10:1187–1190
15. Luus R (1980) Optimization in model reduction. Internat J Control 32:741–747
16. Luus R (1990) Optimal control by dynamic programming using systematic reduction in grid size. Internat J Control 19:995–1013
17. Luus R (1993) Optimization of heat exchanger networks. Industr Eng Chem Res 32:2633–2635
18. Luus R (1996) Application of iterative dynamic programming to optimal control of nonseparable problems. Hungarian J Industr Chem 25:293–297
19. Luus R (1996) Handling difficult equality constraints in direct search optimization. Hungarian J Industr Chem 24:285–290
20. Luus R (1998) Determination of the region sizes for LJ optimization procedure. Hungarian J Industr Chem 26:281–286

21. Luus R (1999) Effective solution procedure for systems of nonlinear algebraic equations. Hungarian J Industr Chem 27:307–310
22. Luus R (2000) Handling difficult equality constraints in direct search optimization. Part 2. Hungarian J Industr Chem 28:211–215
23. Luus R (2000) Iterative dynamic programming. Chapman and Hall/CRC, London, pp 44–66
24. Luus R (2007) Use of line search in the Luus–Jaakola optmization procedure. Proc IASTED Internat Conf on Computational Intelligence. Banff, Alberta, Canada, July 2-4, 2007, pp 128–135
25. Luus R, Brenek P (1989) Incorporation of gradient into random search optimization. Chem Eng Techn 12:309–318
26. Luus R, Dittrich J, Keil FJ (1992) Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor. Canad J Chem Eng 70:780–785
27. Luus R, Hartig F, Keil FJ (1995) Optimal drug scheduling of cancer chemotherapy by direct search optimization. Hungarian J Industr Chem 23:55–58
28. Luus R, Hennessy D (1999) Optimization of fed-batch reactors by the Luus–Jaakola optimization procedure. Industr Eng Chem Res 38
29. Luus R, Jaakola THI (1973) Optimization by direct search and systematic reduction in the size of search region. AIChE J 19:760–766
30. Luus R, Mutharasan R (1974) Stabilization of linear system behaviour by pole shifting. Internat J Control 20:395–405
31. Luus R, Storey C (1997) Optimal control of final state constrained systems. Proc. IASTED Intl. Conf. Modelling, Simulation and Optimization, pp 245–249
32. Luus R, Tassone V (1992) Optimal control of nonseparable problems by iterative dynamic programming. Proc. 42nd Canad. Chemical Engin. Conf., Toronto, Canada, pp 81–82
33. Luus R, Wyrwicz R (1996) Use of penalty functions in direct search optimization. Hungarian J Industr Chem 24:273–278
34. Papangelakis VG, Luus R (1993) Reactor optimization in the pressure oxidation process. Proc. Internat. Symp. Modelling, Simulation and Control of Metall. Processes, 159–171
35. Rosenbrock HH (1960) An automatic way of finding the greatest or least value of a function. Computer J 3:175–184
36. Spaans R, Luus R (1992) Importance of search-domain reduction in random optimization. JOTA 75:635–638
37. Wang BC, Luus R (1977) Optimization of non-unimodal systems. Internat J Numer Methods in Eng 11:1235–1250
38. Wang BC, Luus R (1978) Reliability of optimization procedures for obtaining global optimum. AIChE J 19:619–626
39. Yang SM, Luus R (1983) A note on model reduction of digital control systems. Internat J Control 37:437–439
40. Yang SM, Luus R (1983) Optimization in linear system reduction. Electronics Lett 19:635–637

Discontinuous Optimization

MARCEL MONGEAU
Laboratoire MIP, University Paul Sabatier, Toulouse, France

MSC2000: 90Cxx

Article Outline

Keywords
See also
References

Keywords

Discontinuous optimization; Nondifferentiable optimization; Piecewise linear programming; Active set methods; Exact penalty method

Continuous optimization refers to optimization involving objective functions whose domain of definition is a continuum, as opposed to a set of discrete points in combinatorial (or discrete) optimization. Discontinuous optimization is the special case of continuous optimization in which the objective function, although defined over a continuum (let us suppose over $\mathbf{R}^n$), is not necessarily a continuous function.

We define the discontinuous optimization problem as:

$$\begin{cases} \inf & \widetilde{f}(x) \\ \text{s.t.} & f_i(x) = 0, \quad i \in E, \\ & f_i(x) \geq 0, \quad i \in I, \end{cases} \tag{1}$$

where the index sets E and I are finite and disjoint and $\widetilde{f}$ and f_i, $i \in E \cup I$ are a collection of (possibly discontinuous) piecewise differentiable functions that map $\mathbf{R}^n$ to $\mathbf{R}$. A *piecewise differentiable function* f: $\mathbf{R}^n \to \mathbf{R}$ is a function whose derivative is defined everywhere except over a subset of a finite number of sets, called *ridges*, of the form $\{x \in \mathbf{R}^n : r(x) = 0\}$, where r is a differentiable function, and these ridges partition the domain into subdomains over each of which f is differentiable. By abuse of language, we shall call $r(x)$ a ridge of f.

Without loss of generality, we can restrict our attention to the *unconstrained optimization problem*: $\inf_x f(x)$, where f is a (possibly discontinuous) piecewise differentiable function. Indeed, in order to solve problem (1), one can consider the unconstrained l_1 *exact penalty function*

$$f_\gamma(x) := \gamma \widetilde{f}(x) + \sum_{i \in E} |f_i(x)| - \sum_{i \in I} \min[0, f_i(x)]$$

for a succession of decreasing positive values of the *penalty parameter* γ (f_γ is clearly a piecewise differentiable function). Notice however that using the l_1 penalty function (and dealing with the decrease of a penalty parameter) is only one approach to handling the constrained problem and may not be the best way.

Given a (possibly discontinuous) piecewise differentiable function f defined over $\mathbf{R}^n$ and the finite set $\{r_i(x)\}_{i \in \mathcal{R}}$ of its ridges, we define a *cell* of f to be a nonempty set $C \subseteq \mathbf{R}^n$ such that for all $x, y \in C$ we have $\text{sign}(r_i(x)) = \text{sign}(r_i(y)) \neq 0$, for all $i \in \mathcal{R}$, where the function sign is either 1, −1 or 0, according to whether its argument is positive, negative or zero. Thus, f is differentiable over a cell.

Considering the optimization of functions which are nonsmooth and even discontinuous is motivated by applications in VLSI and floor-planning problems, plant layout, batch production, switching regression, discharge allocation for hydro-electric generating stations, fixed-charge problems, for example (see [4, Introd.] for references). Note that most of these problems can alternatively be modeled within the context of mixed integer programming, a field straddling combinatorial optimization and continuous optimization.

The inescapable nonconvexity nature of discontinuous functions gives rise to the existence of several local optima in discontinuous optimization problems. We do not address here the difficult issue of global optimization. We are concerned with finding a *local* infimum of the above optimization problem. An algorithm looking for local optima can however be used as an adjunct to some heuristic or global optimization method for discontinuous optimization problems but the inherent combinatorial nature of such an approach is often ultimately dominant. More importantly, it provides a framework allowing the optimizer to deal directly with the nonsmoothnesses and discontinuities involved, and thereby, improve solutions found by heuristic methods, when this is possible.

Leaving aside the heuristic methods (which many people facing practical discontinuous optimization problems rely upon in order to solve mixed integer programming formulation of discontinuous optimization problems), previous work on discontinuous optimization includes smoothing algorithms. The *smoothing algorithms* express discontinuities by means of a step function, and then they approximate the step function by a function which is not only continuous but moreover smooth, so that the resulting problem can be solved by a gradient technique (cf. also ► Conjugate–gradient methods). Both I.I. Imo and D.J. Leech [7] and I. Zang [9] developed methods in which the objective function is replaced only in the neighborhood of the discontinuities. Two drawbacks of these methods are the potential numerical instability when we want this neighborhood to be small, and the cost of evaluating the smoothed functions. In many instances the discontinuities of the first derivative are exactly the regions of interest and smoothing has the effect of making such regions less discernible.

Another approach, which deals explicitly with the discontinuities within the framework of continuous optimization, is the following *active set method* (introduced in [4]). Recall the following definitions relevant to active set methods: the *null space* of M, denoted by $\mathcal{N}(M)$, is defined by

$$\mathcal{N}(M) \equiv \left\{ x \in \mathbb{R}^n : \; Mx = \vec{0} \right\}.$$

We say that a ridge r is *active* at $\widehat{x}$ if $r(\widehat{x}) = 0$. Let $\mathcal{A}(\widehat{x}) \subseteq \mathcal{R}$ be the (finite) index set of the ridges that are active at the current point $\widehat{x}$, and let $A(\widehat{x})$ be the *matrix of activities*, having as columns the gradients of the ridges that are active at $\widehat{x}$. In the case of linear ridges, $r_i(x) := a_i^\mathsf{T} x - b_i$, a direction $d \in \mathcal{N}(A^\top(\widehat{x}))$ is said to *preserve* each activity $i \in \mathcal{A}(\widehat{x})$ since for each $i \in \mathcal{A}(\widehat{x})$ we have $r_i(\widehat{x} + \alpha d) = r_i(\widehat{x}) = 0$.

If $\mathcal{A}(\widehat{x}) \neq \emptyset$, then $\nabla f(\widehat{x})$ is not necessarily defined. This is because we cannot talk about the gradient of the

function at $\widehat{x}$ since there is no vector $g \in \mathbf{R}^n$ such that $g^\intercal d$ is the first order change of f in direction d, for any $d \in \mathbf{R}^n$. Thus, we cannot use, as in the smooth situation, the negative gradient direction as a descent direction. We term any $(n \times 1)$-vector $g_{\hat{x}}$ such that

$$f'(\widehat{x}; d) = g_{\hat{x}}^\top d, \quad \text{for all } d \in \mathcal{N}(A^\top),$$

a *restricted gradient* of f at $\widehat{x}$, because it is the gradient of the restriction of f to the space $\mathcal{N}(A^\top(\widehat{x}))$.

Let us first consider the continuous piecewise linear case. We assume that the ridges of f are given, and also we assume that the restriction of f to any cell is known. Hence, we are assuming that more information on the structure of the objective function is available than, for example, in a bundle method [8], which assumes that only one element of the subdifferential is known at any point.

It is shown in [4] that, under some nondegeneracy assumptions (e. g. the gradients of the ridges which are active at $\overline{x}$ are linearly independent), any continuous piecewise linear function f can be *decomposed* in a neighborhood of $\widehat{x}$ into a smooth function and a sum of continuous functions having a single ridge as follows:

$$f(x) = f(\widehat{x}) + g_{\hat{x}}^\top (x - \widehat{x}) + \sum_{i \in \mathcal{A}(\hat{x})} v_{\hat{x}}^i \min(0, a_i^\top (x - \widehat{x})) \ ,$$

for some scalars $\{v_{\hat{x}}^i\}_{i \in \mathcal{A}(\hat{x})}$, and some vector $g_{\hat{x}} \in \mathbb{R}^n$. We term $g_{\hat{x}}$ *the* restricted gradient of f at $\widehat{x}$. Note that if m ridges of f are active at $\widehat{x}$, it means that there are 2^m cells in any small neighborhood of $\widehat{x}$. The vector $g_{\hat{x}}$ and the m scalars $\{v_{\hat{x}}^i\}_{i \in \mathcal{A}(\widehat{x})}$, together with the m gradients of the activities, $\{a_i\}_{i \in \mathcal{A}(\widehat{x})}$, thus completely characterize the behavior of f over the 2^m cells in the neighborhood of $\widehat{x}$!

With such a decomposition at any point of $\mathbf{R}^n$, an algorithm for finding a local minimum of a continuous piecewise linear function f is readily obtained, as long as we assume no degeneracy at any iterate and at any breakpoint encountered in the line search (we shall discuss later the degenerate situation):

1	Choose any $x^1 \in \mathbb{R}^n$ and set $k \leftarrow 1$.
BEGIN	REPEAT
2	Identify the activities, $\mathcal{A}(x^k)$, and compute $d^k \equiv -P(g_{x^k})$, the projection of the restricted gradient onto the space orthogonal to the gradients of the activities.
	IF $d^k = \overrightarrow{0}$ (x^k is a *dead point*; compute a single-dropping descent direction or establish optimality), THEN
3	Compute $\{u_i\}_{i \in \mathcal{A}(x^k)}$, the coefficients of $\{a_i\}_{i \in \mathcal{A}(x^k)}$ in the linear combination of g_{x^k} in terms of the columns of $\mathcal{A}(x^k)$.
4	IF $u_i < 0$ or $u_i > -v_{x^k}^i$, for some $i \in \mathcal{A}(x^k)$ (violated optimality condition), THEN
5	(Drop activity i) Redefine $d^k = P_{-i}(a_i)$, if the violated inequality found corresponds to $u_i \geq 0$; otherwise $d^k = -P_{-i}(a_i)$ if it is $u_i \leq -v_{x^k}^i$, where P_{-i} is the orthogonal projector onto the space orthogonal to the gradients of all the activities but activity i. ELSE stop: x^k is a local minimum of f. ENDIF ENDIF
6	(Line search) Determine the step size α^k by solving $\min_{\alpha > 0} f(x^k + \alpha d^k)$. This line search can be done from x^k, moving from one break-point of f to the next, in the direction d^k, until either we establish unboundedness of the objective function or the value of f starts increasing.
7	Update $x^{k+1} = x^k + \alpha^k d^k$, $k \leftarrow k + 1$.
END	REPEAT

Continuous piecewise linear minimization algorithm

Remark that in step 6, the directional derivative of the objective function in the direction d^k can easily be updated from one breakpoint to the other in terms of the scalar $v_{\overline{x}}^i$, where i is the index of the ridge crossed at breakpoint $\overline{x}$.

Let us now consider the case where f is still piecewise linear but with possibly discontinuities across some ridges. We term such ridges: *faults*, and $\mathcal{F}(\widehat{x})$ denotes the faults that are active at $\widehat{x}$.

Note first that a (local) minimum does not always exist in the discontinuous case. Consider for example the following univariate function, having $x = 0$ as a fault:

$$f(x) = \begin{cases} x + 1 & \text{if } x \geq 0, \\ -x & \text{otherwise.} \end{cases}$$

Hence, we rather look for a *local infimum*. In order to find such a local infimum of a function f having some faults, we shall simply generalize the algorithm for the continuous problem by implicitly considering any discontinuity or *jump* across a fault i in f as the limiting case of a continuous situation. Since we are looking for a local infimum, without loss of generality we shall henceforth only consider functions f such that

$$f(x) = \liminf_{\overline{x} \to x} f(\overline{x}),$$

in other words, we consider the lower semicontinuous envelope of f.

The algorithm for the discontinuous case is essentially the same as in the continuous case except that we consider dropping an active fault from a dead point, x, only if we do so along a direction d such that

$$\lim_{\delta \to 0^+} f(x + \delta d) = f(x)$$

(i. e. as $\delta > 0$ is small, the value of f does not jump up from x to $x + \delta d$). Thus, virtually only step 4 must be adapted from the continuous problem algorithm in order to solve the discontinuous case. To make more carefully the intuitive concept of directions jumping up or down, we define the set of *soaring directions* from a point $\widehat{x}$ to be:

$$S(\widehat{x}) := \left\{ d \in \mathbb{R}^n : \begin{array}{c} \exists \epsilon > 0, \overline{\delta} > 0 : \\ \forall\, 0 < \delta < \overline{\delta} : \\ f(\widehat{x} + \delta d) - f(\widehat{x}) > \epsilon \end{array} \right\}.$$

If we define, for a nondegenerate point $\widehat{x}$,

$$S^+(\widehat{x}) := \left\{ i \in \mathcal{A}(\widehat{x}) : \begin{array}{c} \text{if } d^{i^+} \in \mathcal{N}(A_{-i}^\top) \\ \text{and } a_i^\top d^{i^+} > 0 \\ \text{then } d^{i^+} \in S(\widehat{x}) \end{array} \right\},$$

$$S^-(\widehat{x}) := \left\{ i \in \mathcal{A}(\widehat{x}) : \begin{array}{c} \text{if } d^{i^-} \in \mathcal{N}(A_{-i}^\top) \\ \text{and } a_i^\top d^{i^-} < 0 \\ \text{then } d^{i^-} \in S(\widehat{x}) \end{array} \right\},$$

then the set of soaring single-dropping directions from $\widehat{x}$ are simply the directions dropping an activity $i \in S^+(\widehat{x})$ positively and the directions dropping an $i \in S^-(\widehat{x})$ negatively (we say that activity i is *dropped positively* (*negatively*) if all current activities, except for the ith, are preserved and if, moreover, $a_i^\top d$ is positive (negative)). A fault can now be defined more rigorously: a *positive* (*negative*) fault of f at a point $\widehat{x}$ is a ridge $i \in \mathcal{R}$ such that for any neighborhood, $B(\widehat{x})$, of $\widehat{x}$, there exists a nondegenerate point $x' \in B(\widehat{x})$ with $i \in \mathcal{S}^+(x')$ (with $i \in \mathcal{S}^-(x')$). The set of all positive (negative) faults at $\widehat{x}$ is denoted by $\mathcal{F}^+(\widehat{x})$ $\mathcal{F}^-(\widehat{x})$). The *set of faults* of f at a point $\widehat{x}$ is denoted by

$$\mathcal{F}(\widehat{x}) := \mathcal{F}^+(\widehat{x}) \cup \mathcal{F}^-(\widehat{x}).$$

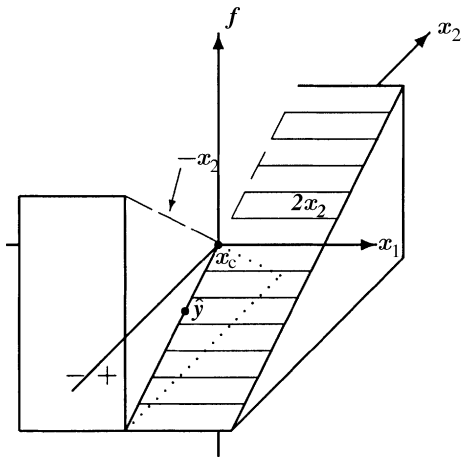
We modify the continuous problem algorithm in such a way that, at a nondegenerate dead point, x^k, we do not need to verify the optimality conditions corresponding to soaring single-dropping directions ($u_i \geq 0, i \in \mathcal{S}^+(x^k)$ and $u_i \leq -\nu^i, i \in \mathcal{S}^-(x^k)$), so that we never consider such single-dropping directions in order to establish whether x^k is optimal. This is reasonable since we are looking for a local minimum. The line-search step (step 6) is modified similarly: when we encounter a breakpoint $\overline{x}$ on a fault along a direction $d \in S(\overline{x})$ (jump up), we stop; while if d is such that $-d \in S(\overline{x})$, (jump down), we carry on to the next breakpoint, and update properly the directional derivative along d.

Note that one has to be careful at a 'contact' point $x_c \in \mathbf{R}$ (defined below). At x_c, contrary to at other points of a fault, we can drop activity i *both* positively and negatively.

The function $f\colon \mathbf{R}^2 \to \mathbf{R}$, given by

$$f(x) = \begin{cases} 2x_2 & \text{if } x_1 > 0 \text{ or} \\ & (x_1 = 0 \text{ and } x_2 \leq 0), \\ -x_2 & \text{otherwise,} \end{cases} \tag{2}$$

illustrates well the situation. Figure 1 shows the graph of f in a neighborhood of $x_c := (0, 0)^\top$ (the dotted lines are simply lines that could be seen if the hatched surface were transparent). The point x_c is a contact point with respect to the fault $x_1 = 0$.



Formally, we define $x_c \in \mathbf{R}^n$ to be a *contact point* of f with respect to $i \in \mathcal{A}(x_c)$, when $i \in \mathcal{F}(x_c)$ such that either

1) $i \in \mathcal{F}^+(x_c) \cap \mathcal{F}^-(x_c)$, or
2) there exist $\sigma^+, \sigma^- \in 3^{|\mathcal{R}|}$ such that $\sigma_i^+ = 1, \sigma_i^- = -1$ and

$$\lim_{\substack{x \to x_c, \\ \sigma(x)=\sigma^+}} f(x) = \lim_{\substack{x \to x_c, \\ \sigma(x)=\sigma^-}} f(x)$$

(continuity when crossing ridge i, which is a fault, at x_c), where $\sigma(x)$ is the vector whose kth component is $\operatorname{sign}(r_k(x))$.

Note that the fault $x_1 = 0$ and the point $x_c = (0, 0)^\top$ satisfy both conditions 1) and 2) in the above definition of a contact point for the function f defined by (2). They however satisfy only condition 1) for the function $f: \mathbf{R}^2 \to \mathbf{R}$, defined by

$$f(x) = \begin{cases} 1 & \text{if } x_1 \geq 0 \text{ and } x_2 \geq 0, \\ 2 & \text{if } x_1 \leq 0 \text{ and } x_2 \leq 0 \\ & \quad \text{and } x \neq (0,0)^\top, \\ 3 & \text{if } x_1 > 0 \text{ and } x_2 < 0, \\ 4 & \text{otherwise,} \end{cases}$$

with faults $x_1 = 0$ and $x_2 = 0$. For the function $f: \mathbf{R}^2 \to \mathbf{R}$ given by

$$f(x) = \begin{cases} -x_2 & \text{if } x_1 > 0 \text{ and } x_2 < 0, \\ 0 & \text{otherwise,} \end{cases}$$

with fault $x_1 = 0$, we satisfy only condition 2) ($\mathcal{F}^-(x_c)$ is empty).

An algorithm similar to the one introduced in the continuous case, but which does not consider soaring single-dropping directions, will encounter no difficulty with the discontinuity in f at any noncontact point (e. g. for (2), at any point other than x_c). Let us assume, without loss of generality, that at the kth iterate, x^k, $\mathcal{F}(x^k) = \mathcal{F}^-(x^k)$. The only step of the continuous algorithm which need to be modified is (assuming moreover that all points encountered in the algorithm are noncontact points):

4	IF $u_i < 0$ for some $i \in \mathcal{A}(x^k)$, or $u_i > -v^i_{x^k}$ for some $i \in \mathcal{A}(x^k) \backslash \mathcal{F}(x^k)$ (violated optimality condition), THEN

The paper [4] describes techniques (including perturbation) to cope with problems that occur in certain cases where the hypothesis of nondegeneracy is not satisfied at points encountered in the course of the algorithm. One cannot however extend this algorithm to deal with *dead-point iterates* (i. e. not encountered as breakpoint along the line search) without considering carefully the combinatorial nature of the problem of degeneracy. Nevertheless, no difficulties were encountered in the computational experiments reported in [4], although serious problems can still arise at certain singular points (contact points and dead-point iterates, at which the objective function is not decomposable). Indeed, in the discontinuous case, there is no straightforward extension of this approach to the cases where the algorithm encounters a contact point. In the continuous case, the behavior of f over two juxtaposed cells are linked. At contact points however, there is coincidence of the values of restrictions of f to subdomains not otherwise linked to each other.

Let us now discuss the extension to the nonlinear case. An advantage of the active set approach for the continuous piecewise linear optimization problem, over, for example, the simplex-format algorithm of R. Fourer [6], is that it generalizes it not only to the discontinuous situation but also to the nonseparable and certain (decomposable) nonconvex cases. Above all, the active set approach is readily extendable to the nonlinear case, by adapting conventional techniques for nonlinear programming, as was done above with the projected gradient method for the (possibly discontinuous)

piecewise linear case. The definition of decomposition must first be generalized so that it expresses the first order behavior of a piecewise differentiable function in the neighborhood of a point. The piecewise linear algorithm described above used descent directions attempting to decrease the smooth part of the function while maintaining the value of its nonsmooth part (when preserving the current activities). A first order algorithm for the nonlinear case could obtain these two objectives *up to first order changes*, as in the approach of A.R. Conn and T. Pietrzykowski to nonlinear optimization, via an l_1 exact penalty function [5]. In order to develop a second order algorithm, assuming now that f is (possibly discontinuous) *piecewise twice-differentiable* (i. e. twice differentiable everywhere except over a finite number of ridges), one must first extend the definition of first order decomposition to that of *second order decomposition*. One could then consider extending the strategies used by T.F. Coleman and Conn [2] on the exact penalty function approach to nonlinear programming (although the l_1 exact penalty function involves only first order types of nondifferentiabilities – ridges). The main idea is to attempt to find a direction which minimizes the change in f (up to second order terms) subject to preserving the activities (up to second order terms). Specifically, second order conditions must be derived (which are the first order conditions plus a condition on the 'definiteness' of the *reduced Hessian* of the *twice-differentiable part* of f (in the second order decomposition of f)). An analog of the Newton step (or of a modification of the Newton method; cf. also ► Gauss-Newton method: Least squares, relation to Newton's method) using a nonorthogonal projection [3] is then taken (or a single-dropping direction is used). An algorithm following these lines would be expected to possess global convergence properties (regardless of starting point) and a fast (2-step superlinear) asymptotic convergence rate as in [1].

See also

► Nondifferentiable Optimization

References

1. Coleman TF, Conn AR (1982) Nonlinear programming via an exact penalty function: Asymptotic analysis. Math Program 24:123–136
2. Coleman TF, Conn AR (1982) Nonlinear programming via an exact penalty function: Global analysis. Math Program 24:137–161
3. Conn AR (1976) Projection matrices – A fundamental concept in optimization. In: Vogt WG, Mickle MH (eds) 7th Annual Conf. in Modelling and Simulation, Ed. April 26–27, pp 599–605
4. Conn AR, Mongeau M (1998) Discontinuous piecewise linear optimization. Math Program 80(3):315–380
5. Conn AR, Pietrzykowski T (Apr. 1977) A penalty function method converging directly to a constrained optimum. SIAM J Numer Anal 14(2):348–375
6. Fourer R (1985) A simplex algorithm for piecewise-linear programming I: Derivation and proof. Math Program, 33:204–233
7. Imo II, Leech DJ (1984) Discontinuous optimization in batch production using SUMT. Internat J Production Res 22(2):313–321
8. Lemaréchal C (1978) Bundle methods in nonsmooth optimization. In: Lemaréechal C, Mifflin R (eds) Proc. IIASA Workshop, Nonsmooth Optimization, March 28–April 8, 1977, vol 3, Pergamon, Oxford, pp 79–102
9. Zang I (1981) Discontinuous optimization by smoothing. Math Oper Res 6(1):140–152

Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points

KURT MARTI
Federal Armed Forces, University Munich, Neubiberg, Germany

MSC2000: 90C15, 90C29

Article Outline

Keywords
Introduction
  Discretely Distributed Stochastic Programs
Example: Scenario Analysis
A System of Linear Relations for the Construction of Descent Directions
Efficient Solutions of (1), (2)
  Comparison of Definitions 7 and 3
  Further Characterization of $E_{D,J}$
  Necessary Optimality Conditions Without Using (Sub)Gradients
Parametric Representation of $E_{D,J}$
See also
References

Keywords

Stochastic program; Stochastic optimization; Mean value function; Uncertainty; Efficient point; Efficient solution; Admissible solution; Pareto optimal solution; Partially monotonous; Partial monotonicity; Robustness; Stochastic matrix; Markov kernel; Descent direction; Necessary optimality condition without using (sub)gradients parametric representations; Scenario analysis

Introduction

Many problems in stochastic optimization, as for instance optimal stochastic structural design problems, stochastic control problems, problems of scenario analysis, etc., can be described [3,5] by mean value minimization problems of the type

$$\begin{cases} \min & F(x) \\ \text{s.t.} & x \in D, \end{cases} \tag{1}$$

where the objective function $F = F_u$ is the *mean value function*, defined by

$$F(x) = \mathsf{E}u(A(\omega)x - b(\omega)), \quad x \in \mathbb{R}. \tag{2}$$

Here, $(A(\omega), b(\omega))$ is a random $m \times (n+1)$ matrix, E denotes the expectation operator, D is a convex subset of R^n and $u : \mathbf{R}^m \to \mathbf{R}$ designates a convex loss function measuring the loss arising from the deviation $z = A(\omega)x - b(\omega)$ between the output $A(\omega)\,x$ of the stochastic linear system $x \to A(\omega)\,x$ and the random target $b(\omega)$.

Solving (1), (2), the loss function u should be exactly known. However, in practice mostly there is some uncertainty about the appropriate selection of u, for instance due to difficulties in assigning appropriate penalty costs to the deviation $z = A(\omega)x - b(\omega)$ between the output $A(\omega)x$ and the target $b(\omega)$. We suppose that $u \in U$, where U is a given set of convex loss functions containing the true, but unknown loss function u_0. A possible way out in this situation of uncertainty about u is either to construct (feasible) descent directions h of F at a (iteration) point x being valid for a large class U of loss functions, or to provide the decision maker with a certain set $E = E_{\mathrm{D,\,U}}$ $(\subset D)$ of *efficient points* or solutions, being substitutes for an optimal solution x^* of (1), (2); hence, this set $E_{D,\,U}$ or at least its closed hull $\overline{E_{D,U}}$ should contain an optimal solution $x^* = x_u^*$ of (1), (2) for each $u \in U$. An important class $U = C_m^J$ of loss functions u is the set of partially monotonous increasing convex loss functions on $\mathbf{R}^m$ defined asfollows:

Definition 1 Let J be a given subset of $\{1, \ldots, m\}$. For $J = \emptyset$ we put $C_m^{\emptyset} = C_m$, where C_m is the set of all convex functions u on $\mathbf{R}^m$. If $J \neq \emptyset$, then $\mathcal{C}_m^J$ denotes the set of all convex functions $u : \mathbf{R}^m \to \mathbf{R}$ having the following property:

$$z_I \leq w_I,\ z_{II} \leq w_{II} \implies u(z) \leq u(w). \tag{3}$$

Here, $z_I \in \mathbf{R}^{|J|}$, $z_{II} \in \mathbf{R}^{m-|J|}$ is the partition of any $z \in \mathbf{R}^m$ into the subvectors $z_I = (z_i)_{i \in J}$, $z_{II} = (z_i)_{i \notin J}$. Moreover, $z_I \leq w_I$ means that $z_j \leq w_j$ for all $j \in J$.

Remark 2 In many cases one has loss functions $u \in C_m^J$ with one of the following additional strict partial monotonicity property:

$$\begin{cases} z_I \leq w_I, \\ z_{II} = w_{II}, \\ z_i < w_i \text{ for some } i \in J \end{cases} \implies u(z) < u(w), \tag{4}$$

$$z_I < w_I,\ z_{II} = w_{II} \implies u(z) < u(w), \tag{5}$$

where $z_I < w_I$ means that $z_j < w_j$ for all $j \in J$.

For a given set U of convex loss functions u containing the true, but unknown loss function u_0, a first definition of efficient solutions can be given as follows:

Definition 3 A point $x \in D$ is called a *nondominated*, *admissible* or *Pareto optimal solution* of (1), (2) if there is no vector $\widetilde{x} \in D, \widetilde{x} \neq x$, such that

$$F_u(\tilde{x}) \leq F_u(x) \quad \text{for all } u \in U, \tag{6}$$

$$F_{\tilde{u}}(\tilde{x}) < F_u(x) \quad \text{for some } \tilde{u} \in U, \tag{7}$$

where $F_u(x) := \mathsf{E}u(A(\omega)x - b(\omega))$. Let $E_{D,u}^0$ denote the set of all nondominated solutions of (1), (2).

Discretely Distributed Stochastic Programs

In the following we consider the construction of descent directions and efficient solutions for (1), (2) in the case that $(A(\omega), b(\omega))$ has a discrete distribution

$$\mathsf{P}_{(A(\cdot),b(\cdot))} = \sum_{i=1}^{r} \alpha_i \epsilon_{(A^i,b^i)}, \tag{8}$$

where $r > 1$ is an integer, $\alpha_i > 0$, $i = 1, \ldots, r$, $\sum_{i=1}^{r} \alpha_i = 1$, and $\epsilon_{(A^i, b^i)}$ denotes the one-point measure in the given $m \times (n+1)$ matrix (A^i, b^i), $i = 1, \ldots, r$.

Example: Scenario Analysis

Given a certain planning problem, in *scenario analysis* [1,2,6,7,8] the future evolution or development of the system to be considered is anticipated or explored by means of a (usually small) number r (e. g., $r = 3, 4, 5, 6$) of so-called scenarios $s^1, \ldots, s^r$. Scenarios s^i, $i = 1, \ldots, r$, are plausible alternative models of the future development given by 'extreme points' of a certain set of basic or key variables. An individual scenario or a certain mixture of the scenarios $s^1, \ldots, s^r$ is assumed then to be revealed in the considered future time period. We assume now that the planningproblem can be described mathematically by the optimization problem

$$\begin{cases} \min & c'x \\ \text{s.t.} & Tx = (\leq)\, h, \quad x \in D. \end{cases} \tag{9}$$

Here, D is a given convex subset of $\mathbf{R}^n$, and the data (c, T, h) are given by $(c, T, h) = (c^i, T^i, h^i)$ for scenario s^i, $i = 1, \ldots, r$, where c^i is an n-vector, T^i an $m \times n$ matrix and h^i an m-vector. Having written here the scenarios $s^1, \ldots, s^r$ by means of (9) and the data (c^i, T^i, h^i), $i = 1, \ldots, r$, and facing therefore the subproblems

$$\begin{cases} \min & c^{i\prime}x \\ \text{s.t.} & T^i x = (\leq)\, h^i, \quad x \in D, \end{cases} \tag{10}$$

for $i = 1, \ldots, r$, the decision maker has then to select an appropriate decision $x \in D$. Since one is unable in general to predict with certainty which scenario s^i will occur, scenario analysts are looking for decisions x^0 which are 'robust' with respect to the different scenarios or 'scenario-independent', cf. [6,7,8]. Obviously, this robustness concept is closely related to the idea of detecting 'similarities' within the family of optimal solutions $x^*(s^i)$, $i = 1, \ldots, r$, of the individual subproblems $(10)_{(i)}$, $i = 1, \ldots, r$. Let $\alpha_1, \ldots, \alpha_r$ with $\alpha_i > 0$, $i = 1, \ldots, r$, $\sum_{i=1}^{r} \alpha_i = 1$, be (subjective) probabilities for the occurrence of $s^1, \ldots, s^r$, or weights reflecting the relative importance of $s^1, \ldots, s^r$. Considering loss functions $u \in C_m^J$ for evaluating the violations $z^i = T^i x - h^i$ of the constraint $T^i x = h^i$, $T^i x \leq h^i$, resp., in $(10)_{(i)}$, a class of robust or scenario-independent decisions are obviously the efficient solutions of

$$\min_{x \in D} \sum_{i=1}^{r} \alpha_i \left(c^{i\prime} x + u(T^i x - h^i) \right), \tag{11}$$

which is a discretely distributed stochastic optimization problem of the type (1), (2).

A System of Linear Relations for the Construction of Descent Directions

Fundamental for the computation of the set $E_{D,U}$ of efficient solutions of (1), (2) is the following construction method for descent directions of the objective function F of (1), (2), cf. [3,4]. We suppose that the true, but unknown loss function u in (1) is, see Definition 1, an element of $\mathcal{C}_m^J$ for some known index set $J \subset \{1, \ldots, m\}$. We recall that for any vector $z \in \mathbf{R}^m$ the subvectors z_I, z_{II} are defined by $z_I = (z_i)_{i \in J}$, $z_{II} = (z_i)_{i \notin J}$;see (3). Of course, if $J = \emptyset$, then $z = z_{II}$ and z_I does not exist. For any $m \times (n+1)$ matrix (A, b), let (A_I, b_I), (A_{II}, b_{II}), resp., denote the submatrices of (A, b) having the rows (A_i, b_i) with $i \in J$, $i \in \{1, \ldots, \mathrm{m}\} \setminus J$, respectively.

Given an n-vector x (e. g., the tth iteration point of an algorithm for solving (1), (2)), we consider, in extension of [3, system (3.1)–(3.4b)], the following system of linear relationsfor the unknowns (y, Π), where $y \in \mathbf{R}^n$ and $\Pi = (\pi_{ij})$ is an auxiliary $r \times r$ matrix:

$$\sum_{j=1}^{r} \pi_{ij} = 1, \qquad \pi_{ij} \geq 0, \quad i, j = 1, \ldots, r, \tag{12}$$

$$\alpha_j = \sum_{i=1}^{r} \alpha_i \pi_{ij}, \quad j = 1, \ldots, r, \tag{13}$$

$$A_I^j y - b_I^j \leq \sum_{i=1}^{r} \frac{\alpha_i \pi_{ij}}{\alpha_j} (A_I^i x - b_I^i), \quad j = 1, \ldots, r, \tag{14}$$

$$A_{II}^j y - b_{II}^j = \sum_{i=1}^{r} \frac{\alpha_i \pi_{ij}}{\alpha_j} (A_{II}^i x - b_{II}^i), \quad j = 1, \ldots, r. \tag{15}$$

The transition probability measure

$$K^j = \sum_{i=1}^{r} \beta_{ij} \epsilon_{z^i}, \quad \beta_{ij} = \frac{\alpha_i \pi_{ij}}{\alpha_j}, \quad z^i = A^i x - b^i, \tag{16}$$

is not a one-point measure for at least one j, $1 \leq j \leq r$.

There exists at least one j, $1 \le j \le r$, such that for all $i = 1, \ldots, r$

$$K^j \text{ is not a one-point measure and } \pi_{ij} > 0. \quad (17)$$

$$\text{At least one inequality in (14) holds with } <. \quad (18)$$

The constraint $x \in D$ in (1) can be handled by adding the condition

$$y \in D. \quad (19)$$

Remark 4

a) By ϵ_z we denote the one-point measure in a point $z \in \mathbf{R}^m$.

b) According to (12), Π is a stochastic matrix. System (12)–(15) has always the trivial solution $(y, \Pi) = (x, \mathrm{Id})$, where Id is the $r \times r$ identity matrix.

c) If (y, Π) solves (12)–(15), then

$$\overline{A}_I y \le \overline{B}_I x, \quad \overline{A}_{II} y = \overline{B}_{II} x, \quad (20)$$

where $\overline{A}_I = \mathsf{E}A_I(\omega)$, $\overline{A}_{II} = \mathsf{E}A_{II}(\omega)$.

d) If $\mathsf{P}_{A_{II}(\cdot)y - b_{II}(\cdot)}$ denotes the probability distribution of the random $(m - |\mathrm{J}|)$-vector $A_{II}(\omega)x - b_{II}(\omega)$, then (12), (13) and (15) mean that the distributions $\mathsf{P}_{A_{II}(\cdot)y - bII(\cdot)}$ and $\mathsf{P}_{A_{II}(\cdot)x - bII(\cdot)}$ corresponding to y, x, resp., are related by

$$\mathsf{P}_{A_{II}(\cdot)x - b_{II}(\cdot)} = K\mathsf{P}_{A_{II}(\cdot)y - b_{II}(\cdot)} = \int K(w, \cdot)\mathsf{P}_{A_{II}(\cdot)y - b_{II}(\cdot)}(dw), \quad (21)$$

where $K(w, \cdot)$ is the *Markov kernel* defined by

$$K(w^j, \cdot) := K^j = \sum_{i=1}^{r} \frac{\alpha_i \pi_{ij}}{\alpha_j} \epsilon_{z^i}, \quad (22)$$

with $w^j = A^j y - b^j$, $z^i = A^i x - b^i$, $i, j = 1, \ldots, r$. Since $\int zK(w, dz) = w$, the Markov kernel K is also calleda *dilatation*.

e) If n-vectors x, y are related by (21), (22), then for every convex subset $B \subset \mathbf{R}^m - |\mathrm{J}|$ we have that

$$\mathsf{P}_{A_{II}(\cdot)x - b_{II}(\cdot)}(B) = 1 \implies \mathsf{P}_{A_{II}(\cdot)x - b_{II}(\cdot)}(B) = 1;$$

hence, the distribution of $A_{II}(\cdot)\, y - b_{II}(\cdot)$ is concentrated to the convex hull of the support of $\mathcal{P}_{AII}(\cdot)x - b_{II}(\cdot)$.

f) If $J = \emptyset$, then (14) vanishes and (15) reads

$$A^j y - b^j = \sum_{i=1}^{r} \frac{\alpha_i \pi_{ij}}{\alpha_j}(A^i x - b^i), \quad j = 1, \ldots, r. \quad (23)$$

In the special case

$$(A_I^j, b_I^j) = (\overline{A}_I, \overline{b}_I) \quad \text{for all } j = 1, \ldots, r, \quad (24)$$

i. e., if $(A_I(\omega),\ b_I(\omega))$ is constant with probability one, then (14) is reduced, cf. (20), to

$$\overline{A}_I y \le \overline{A}_I x. \quad (25)$$

The meaning of (12)–(15) and the additional conditions (16)–(18) for the basic mean value minimization problem (1), (2) with objective function F is summarized in the next result.

Theorem 5 *Let J be any fixed subset of $\{1, \ldots, m\}$.*

a) If (y, II) is a solution of (12)–(15), then $F(y) \le F(x)$ for every $u \in \mathcal{C}_m^J$. For $J = \emptyset$ also the converse holds: If there is a vector y such that $F(y) \le F(x)$ for all $u \in \mathcal{C}_m$ $(\mathcal{C}_m^{\emptyset})$, then there exists an $r \times r$ matrix II such that (y, II) satisfies (12), (13) and (23).

b) If (y, II) is a solution of (12)–(15) and (16), then $F(y) < F(x)$ for every $u \in \mathcal{C}_m^J$ which is strictly convex on $\mathrm{conv}\{z^i : 1 \le i \le r\}$.

c) If (y, II) is a solution of (12)–(15) and (17), then $F(y) < F(x)$ for every $u \in \mathcal{C}_m^J$ which is not affine-linear on $\mathrm{conv}\{z^i : 1 \le i \le r\}$.

d) If (y, II) fulfills (12)–(15) and (18), then $F(y) < F(x)$ for every $u \in \mathcal{C}_m^J$ satisfying (4).

Proof If x and (y, II) are related by (12)–(15), then $F(y) \le \sum_{j=1}^{r} \alpha_j u(\sum_{i=1}^{r} \beta_{ij} z^i)$ for every $u \in \mathcal{C}_m^J$.

If x, (y, II) are related by (12)–(15) and (18), then $F(y) < \sum_{j=1}^{r} \alpha_j u(\sum_{i=1}^{r} \beta_{ij} z^i)$ for every $u \in \mathcal{C}_m^J$ fulfilling (4). The rest can then be shown as in [3, Thm. 2.2].

A simple, but important consequence of the above theorem is stated next:

Corollary 6 *For given $x \in \boldsymbol{R}^n$ or $x \in D$ let (y, II) be any solution of (12)–(15) such that $y \ne x$, $y \in D \setminus \{x\}$, respectively.*

a) Then $h = y - x$ is a descent direction, a feasible descent direction, resp., of F at x for every $u \in \mathcal{C}_m^J$ such that F is not constant on the line segment xy joining x and y.

b) If (y, II) fulfills also (16), (17), (18), resp., then $h = y - x$ is a (feasible) descent direction of F at x for every $u \in \mathcal{C}_m^J$ which is strictly convex on conv $\{z^i : 1 \le i \le r\}$, is not affine-linear on conv $\{z^i : 1 \le i \le r\}$, fulfills (4), respectively.

Efficient Solutions of (1), (2)

In the following we suppose that the unknown loss function u is an element of $\mathcal{C}_m^J$, where J is a given subset of $\{1, \ldots, m\}$. For a given point $x \in D$, the descent direction-finding procedure described in the previous section only can fail completely if for each solution (y, II) of (12)–(15) with $x \in D$ we have that

$$A^j y = A^j x \quad \text{for each } j = 1, \ldots, r. \tag{26}$$

Indeed, in this case we either have $y = x$, or, for arbitrary loss functions u, the objective function F of (1), (2) is constant on the whole line through the points x, y. This observation suggests the following basic efficiency concept.

Definition 7 A point $x \in D$ is called a $(\mathcal{C}_m^J)$-*efficient point* or a $(\mathcal{C}_m^J)$-*efficient solution* of (1), (2) if and only if for each solution (y, II) of (12)–(15) with $y \in D$ we have that $A^j y = A^j x$ for each $j = 1, \ldots, r$, i. e., $A(\omega)x = A(\omega)y$ with probability 1. Let $E_{D,J}$ denote the set of all efficient points of (1), (2).

For deriving parametric representations of $E_{D,J}$, we need the following definitions and lemmas.

For a given n-vector x and $z^i = A^i x - b^i$, $i = 1, \ldots, r$, let $S = S_x \subset \{1, \ldots, r\}$ with $|S| = s$ be an index set such that $\{z^i : 1 \le i \le r\} = \{z^i : i \in S\}$, where $z_i \ne z_j$ for $i, j \in S$, $i \ne j$. Defining for $i \in S$, $j = 1, \ldots, r$, the quantities

$$\widetilde{\alpha}_i := \sum_{z^t = z^i} \alpha_t,$$

$$\tau_{ij} := \frac{1}{\widetilde{\alpha}_i} \sum_{z^t = z^i} \alpha_t \pi_{tj}, \quad \widetilde{\beta}_{ij} := \frac{\widetilde{\alpha}_i \tau_{ij}}{\alpha_j}, \tag{27}$$

we find that relations (12)–(15) can also be represented by

$$\sum_{j=1}^{r} \tau_{ij} = 1, \quad \tau_{ij} \ge 0, \quad j = 1, \ldots, r, \quad i \in S, \tag{28}$$

$$\alpha_j = \sum_{i \in S} \widetilde{\alpha}_i \tau_{ij}, \quad j = 1, \ldots, r, \tag{29}$$

$$A_I^j y - b_I^j \le \sum_{i \in S} \widetilde{\beta}_{ij} z_I^i, \quad j = 1, \ldots, r, \tag{30}$$

$$A_{II}^j y - b_{II}^j = \sum_{i \in S} \widetilde{\beta}_{ij} z_{II}^i, \quad j = 1, \ldots, r. \tag{31}$$

For the next lemma we still need the $s \times r$ matrix $T^0 = (\tau_{ij}^0)$ defined by

$$\tau_{ij}^0 = \begin{cases} 0 & \text{if } z^i \ne z^j, \\ \dfrac{\alpha_j}{\widetilde{\alpha}_i} & \text{if } z^j = z^i, \end{cases} \quad \text{for } i \in S,\ j = 1, \ldots, r. \tag{32}$$

Lemma 8 *Let (y, II) be a solution of (12)–(15), and let $T = T(II) = (\tau_{ij})$ be the $s \times r$ matrix having the elements τ_{ij} given by (27). If (26) holds, then $T(II) = T^0$ and (14) holds with '='.*

Lemma 8 implies the following important property of efficient solutions:

Corollary 9 *Let $x \in D$ be an efficient solution of (1), (2). If (y, II) is any solution of (21)–(22) with $y \in D$, then $T(II) = T^0$ and (14) holds with '='.*

For $J = \emptyset$ we obtain the set $E_D := E_{D,\emptyset}$ of all C_m-efficient solutions of (1), (2). This set is studied in [3]. An important relationship between E_D and $E_{D,J}$ for any $J \subset \{1, \ldots, m\}$ is given next:

Lemma 10 *$E_{D,J} \subset E_D$ for every $J \subset \{1, \ldots, m\}$.*

Comparison of Definitions 7 and 3

Comparing the efficient solutions according to Definition 7 and the nondominated solutions according to Definition 3, first for $J = \emptyset$, i. e., $U = C_m$, we find the following correspondence:

Theorem 11 $E_{D,\emptyset} = E_{D,C_m}^{(0)}$.

The next corollary follows immediately from the above theorem and Lemma 10.

Corollary 12 *$E_{D,J} \subset E_{D,\emptyset} = E_{D,C_m}^{(0)}$ for $J \subset \{1, \ldots, m\}$.*

Considering now $U = C_m^J$ we have this inclusion:

Theorem 13 *$E_{D,J} \supset E_D$, $C_m^{J\,(0)}$ for $J \subset \{1, \ldots, m\}$.*

The following inclusion follows from Corollary 12 and Theorem 13.

Corollary 14 *$E_{D,C_m^J}^{(0)} \subset E_{D,J} \subset E_{D,C_m}^{(0)}$ for $J \subset \{1, \ldots, m\}$.*

A converse statement to Theorem 13 can be obtained for (24):

Theorem 15 *If* $(A_I(\omega), b_I(\omega)) = (\overline{A}_I, \overline{b}_I)$ *with probability 1, then* $E_{D,J} = E^{(0)}_{D,C^J_m}$ *for each* $J \subset \{1, \ldots, m\}$.

Further Characterization of $E_{D,J}$

The C^J_m-efficiency of a point $x \in D$ can also be described in the following way.

Theorem 16 *A point* $x \in D$ *is* (C^J_m)*-efficient if and only if for every solution (y, II) of (12)-(15) we have that* $A^j y = A^j x$ *for all* $j = 1, \ldots, r$, *or* $h = y - x$ *is not a feasible direction for D at x.*

Necessary Optimality Conditions Without Using (Sub)Gradients

If $x \in D$ is efficient, then, cf. Theorem 16, the descent direction-finding method described in in the previous Section fails at x. Since especially in any optimal solution x^* of (1), (2) no feasible descent direction may exist, efficient points are candidates for optimal solutions:

Theorem 17 *Suppose that for every* $x \in D$ *and every solution (y, II) of (12)-(15) with* $y \in D$ *the objective function F of (1), (2) with a loss function* $u \in C^J_m$ *is constant on the line segment xy if and only if* $A^j y = A^j x$ *for every* $j = 1, \ldots, r$. *If* x^* *is an optimal solution of (1), (2), then* $x^* \in E_{D,J}$.

Remark 18 The assumption in Theorem 17 concerning F is fulfilled, e. g., if $u \in C^J_m$ is strictly convex on the convex hull conv $\{(A^j y - b^j)(A^j x - b^j)$: $x, y \in D$, $1 \le j \le r\}$ generated by the line segments $(A^j y - b^j)(A^j x - b^j)$ joining $(A^j y - b^j)$ and $(A^j x - b^j)$.

If the assumption in Theorem 17 concerning F does not hold, then it may happen that F is constant on a certain line segment xy though $A^j y \neq A^j x$ for at least one index j, $1 \le j \le r$. Hence, Theorem 17 can not be applied then directly. However, in this case the following modification of Theorem 17 holds true.

Theorem 19 *Let u be an arbitrary loss function from* C^J_m *for some* $J \subset \{1, \ldots, m\}$. *If D is a compact convex subset of* $\mathbf{R}^n$, *then there exists at least one optimal solution* x^* *of (1), (2) lying in the closure* $\overline{E}_{D,J}$ *of the set* $E_{D,J}$ *of efficient solutions of (1), (2).*

Parametric Representation of $E_{D,J}$

Suppose that $u \in C^J_m$ for some index set $J \subset \{1, \ldots, m\}$. For solving the descent direction-generating relations (12)–(15) and (16)–(18), resp., we may use, see Theorem 5 and Corollary 6, the quadratic program, cf. [3,4],

$$
\begin{cases}
\min & \eta'(\overline{A}_I y - \overline{A}_I x) + \sum_{j=1}^{r} \alpha_j \sum_{i \in S} \widetilde{\beta}^2_{ij} \\
\text{s.t.} & \sum_{j=1}^{r} \tau_{ij} = 1, \quad \tau_{ij} \ge 0, \\
& j = 1, \ldots, r, \quad i \in S, \\
& \alpha_j = \sum_{i \in S} \widetilde{\alpha}_i \tau_{ij}, \quad j = 1, \ldots, r, \\
& A^j_I y - b^j_I \le \sum_{i \in S} \widetilde{\beta}_{ij} z^i_I, \quad j = 1, \ldots, r, \\
& A^j_{II} y - b^j_{II} = \sum_{i \in S} \widetilde{\beta}_{ij} z^i_{II}, \quad j = 1, \ldots, r, \\
& y \in D,
\end{cases}
\tag{33}
$$

where $\eta = (\eta_l)$ is a $|J|$-vector having fixed positive components η_l, $l \in J$. Efficient solutions of (1), (2) can be characterized as follows:

Lemma 20 *A vector* $x \in D$ *is an efficient solution of (1), (2) if and only if (33) has an optimal solution* (y^*, T^*) *such that* $A^j y^* = A^j x$ *for all* $j = 1, \ldots, r$.

Remark 21 According to Lemma 8 we have then also that $T^* = T^0$ and (14) holds with ' = '.

We suppose now that the feasible domain D of (1), (2) is given by

$$
D = \{x \in \mathbb{R}^n : \; g_k(x) \le 0, \; k = 1, \ldots, \kappa\}. \tag{34}
$$

Here, $g_1, \ldots, g_\kappa$ are differentiable, convex functions. Moreover, we suppose that (33) has a feasible solution (y, T) such that for each nonaffine linear function g_k

$$
g_k(y) < 0. \tag{35}
$$

No constraint qualifications are needed in the important special case $D = \{x \in \mathbf{R}^n : G_x \le g\}$, where (G, g) is a given$\kappa \times (n+1)$ matrix.

By means of the Kuhn-Tucker conditions of (33), the following *parametric representation* of $E_{D,J}$ can be derived [3,4]:

Theorem 22 *Let D be given by (34), and assume that the constraint qualification (35) holds for every* $x \in D$. *An n-vector x is an efficient solution of (1), (2) if and*

only if x satisfies the linear relations

$$\lambda_j - \lambda_i - \left(\frac{\widehat{\gamma}_j}{\alpha_j} - \frac{\widehat{\gamma}_i}{\alpha_i}\right)' z^i = 2\left(\frac{1}{\alpha_i} - \frac{1}{\alpha_j}\right), \quad \text{if } z^i = z^j, \tag{36}$$

$$\lambda_j - \lambda_i - \left(\frac{\widehat{\gamma}_j}{\alpha_j} - \frac{\widehat{\gamma}_i}{\alpha_i}\right)' z^i \geq \frac{2}{\alpha_i}, \quad \text{if } z^i \neq z^j, \tag{37}$$

where $\lambda_1, \ldots, \lambda_r$ *are arbitrary real parameters, and the parameter m-vectors* $\gamma_1, \ldots, \gamma_r$ *and further parameter vectors* $\rho \in \boldsymbol{R}^\kappa$, $y \in \boldsymbol{R}^n$ *are selected such that*

$$\sum_{j=1}^{r} A^{j'} \widehat{\gamma}_j + \sum_{k=1}^{\kappa} \rho_k \nabla g_k(y) = 0, \tag{38}$$

$$\gamma_{jI} \geq 0, \quad j = 1, \ldots, r, \tag{39}$$

$$g_k(x) \leq 0, \quad k = 1, \ldots, \kappa, \tag{40}$$

$$g_k(y) \leq 0, \quad \rho_k g_k(y) = 0, \quad \rho_k \geq 0, \quad k = 1, \ldots, \kappa, \tag{41}$$

$$A^j y = A^j x, \quad j = 1, \ldots, r, \tag{42}$$

and the vectors $\widehat{\gamma}_j$ *are defined by* $\widehat{\gamma}_j = \binom{\alpha_j \eta + \gamma_{jI}}{\gamma_{jII}}$, $j = 1, \ldots, r$.

See also

- Approximation of Extremum Problems with Probability Functionals
- Approximation of Multivariate Probability Integrals
- Extremum Problems with Probability Functions: Kernel Type Solution Methods
- General Moment Optimization Problems
- Logconcave Measures, Logconvexity
- Logconcavity of Discrete Distributions
- L-shaped Method for Two-Stage Stochastic Programs with Recourse
- Multistage Stochastic Programming: Barycentric Approximation
- Preprocessing in Stochastic Programming
- Probabilistic Constrained Linear Programming: Duality Theory
- Probabilistic Constrained Problems: Convexity Theory
- Simple Recourse Problem: Dual Method
- Simple Recourse Problem: Primal Method
- Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- Static Stochastic Programming Models
- Static Stochastic Programming Models: Conditional Expectations
- Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- Stochastic Integer Programs
- Stochastic Linear Programming: Decomposition and Cutting Planes
- Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- Stochastic Network Problems: Massively Parallel Solution
- Stochastic Programming: Minimax Approach
- Stochastic Programming Models: Random Objective
- Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- Stochastic Programming with Simple Integer Recourse
- Stochastic Programs with Recourse: Upper Bounds
- Stochastic Quasigradient Methods in Minimax Problems
- Stochastic Vehicle Routing Problems
- Two-Stage Stochastic Programming: Quasigradient Method
- Two-Stage Stochastic Programs with Recourse

References

1. Carroll JM (ed) (1995) Scenario-based design. WileyNew York , New York
2. Chandler J, Cockle P (1982) Techniques of scenario planning. McGraw-Hill, New York
3. Marti K (1988) Descent directions and efficient solutions in discretely distributed stochastic programs. Lecture Notes Economics and Math Systems, vol 299. Springer, Berlin
4. Marti K (1992) Computation of efficient solutions of discretely distributed stochastic optimization problems. ZOR 36:259–294
5. Marti K (1996) Stochastic optimization methodsin engineering. In: Dolezal J, Fidler J (eds) System Modelling and Optimization. Chapman and Hall, London, pp 75–87
6. Reibnitz Uvon (1988) Scenario techniques. McGraw-Hill, New York
7. Steinsiek E, Knauer P (1981) Szenarien als Instrument der Umweltplanung. Angewandte Systemanalyse 2:10–19
8. Zentner RD (1975) Scenarios: A new tool for corporate planners. Chem and Eng News, Internat Ed 53:22–34

Discrete Stochastic Optimization

GEORG PFLUG
University Vienna, Vienna, Austria

MSC2000: 90C15, 90C27

Article Outline

Keywords
Stochastic Simulated Annealing
Stochastic Branch and Bound
See also
References

Keywords

Derivatives; Stochastic optimization

A *stochastic combinatorial optimization* problem is of the form

$$\begin{cases} \min & F(x) = \int H(x,v)\, d\mu_x(v) \\ \text{s.t.} & x \in S \end{cases} \tag{1}$$

where $S = \{x_1, \ldots, x_N\}$ is a finite discrete feasible set. If the value of the objective function F is easily obtainable, the problem is just a deterministic combinatorial optimization problem. In most applications however, the value of the objective function F has to be approximated by numerical integration or Monte-Carlo simulation.

Some problems of type (1) exhibit a special structure, which can be exploited for solution methods, like the *stochastic linear optimization problems*, where S are the integer points of a convex polyhedron and H is piecewise linear (see ► Stochastic integer programming: Continuity, stability, rates of convergence). In this contribution, we discuss problems with an arbitrary and unstructured feasible set S.

An example is the stochastic single machine tardiness problem (SSMTP): The optimal sequence of m jobs, which are processed on a single machine has to be found. Each job has a random processing time, which is distributed according to the distribution function G_i, $i = 1, \ldots, m$ (independent of all others), and a fixed due date d_i. The feasible set S is the set of all $m!$ permutations π of $\{1, \ldots, m\}$. If π is the solution found, we process job $\pi(1)$ as the first, $\pi(2)$ as the second and so on. Let $c_i(u)$ be the costs for job i being late u time units ($c_i(u) = 0$ for $u \le 0$). The SSMTP is

$$\begin{cases} \min & \sum_{i=1}^{m} \mathsf{E}[c_i(V_{\pi(1)} + \cdots + V_{\pi(i)} - d_{\pi(i)})] \\ \text{s.t.} & \pi \in S \end{cases} \tag{2}$$

where V_i are random variables distributed independently according to G_i. The analytic calculation of the objective function (OF) in (2) involves multiple integrals (the convolution of up to m distribution functions). A simple way of approximating the OF is by Monte-Carlo simulation. Let $V_i^{(1)}, \ldots, V_i^{(n)}$ be independent random (pseudorandom) variables, each with distribution G_i. The true expectation $F(\pi) = \mathrm{E}[c_i(V_{\pi(1)} + \cdots + V_{\pi(\mathrm{i})} - d\pi(\mathrm{i}))]$ is approximated by the estimate

$$\widehat{F}_n(\pi) = \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{m} [c_i(V_{\pi(1)}^{(j)} + \cdots + V_{\pi(i)}^{(j)} - d_{\pi(i)})].$$

In principle, all exact (branch and bound) and heuristic (evolutionary algorithms, tabu search, ant systems, random search, simulated annealing, genetic algorithms) methods for combinatorial optimization may be applied to stochastic combinatorial optimization — just that the exact values $F(x)$ have to be replaced by stochastic estimates $\widehat{F}_n(x)$, which are based on sample size n.

The main difficulty in stochastic combinatorial optimization is the fact that even if $F(x_1) \le F(x_2) - \delta$, it may happen with positive probability that $\widehat{F}_n(x_1) > \widehat{F}_n(x_2)$, that is we may wrongly conclude that x_2 is better than x_1. The probability of this error decreases to zero with sample size n increasing to infinity. A compromise between the quality of the solution and the costs of very large samples has to be found in stochastic optimization.

If the random distribution μ_x in (1) does not depend on x, *common random numbers* may be taken. To be more precise, let $V^{(1)}, \ldots, V^{(n)}$ be a sample from μ and let

$$\widehat{F}_n(x_i) = \frac{1}{n} \sum_{j=1}^{n} H(x_i, V^{(j)}).$$

The estimates $\widehat{F}$ are now correlated, and the probability that $\widehat{F}_n(x_1) > \widehat{F}_n(x_2)$ although $\widehat{F}_n(x_1) \le \widehat{F}_n(x_2) - \delta$

is typically much smaller for such a choice than with samples taken independently for each x_j (see [6]).

If the estimates $\widehat{F}$ are difficult to get (e. g. they need real observation or expensive simulation) allocation rules decide, which estimate or which set of estimates has to be taken next. These rules try to exclude quickly subsets of the feasible set, which – with high statistical evidence – do not contain optimal solutions. The effort is then concentrated on the (shrinking) set of not yet excluded points.Allocation rules may be based on *subset selection* (see [3]) or *ordinal optimization* (see [5]). There is also a connection to experimental design, in particular to *sequential experimental design*: In experimental design one has to choose the next point(s) for sampling, which – based on the information gathered so far – will give the best additional information which we need to solve the underlying estimation or optimization problem (for experimental design literature see [1] and the references therein).

For large sets S, which have graph-neighborhood or partition structures, 'stochastic' variants of neighbor search or branch and bound methods may be used. In particular, stochastic simulated annealing and stochastic branch and bound have been studied in literature.

Stochastic Simulated Annealing

This is a variant of ordinary simulated annealing (cf. ► Simulated annealing): The Metropolis rule for the acceptance probability is calculated on the basis of the current stochastic estimates of the objective function, i. e. the new state x_j is preferred to the current state x_i with probability

$$\min\left(\exp\left(-\frac{\widehat{F}_n(x_j)-\widehat{F}_n(x_i)}{k_B T}\right),1\right)$$

where k_B is the Boltzmann constant and T is the temperature. The estimates $\widehat{F}$ are improved in each step by taking additional observations, i. e. increasing the sample size n. For an analysis of this algorithm see [4].

Stochastic Branch and Bound

For the implementation of a stochastic branch and bound method (cf. also ► Integer programming: Branch and bound methods), an estimate of a lower bound function is needed. Recall that a function $\mathbf{F}$, defined on the subsets of S, is called a *lower bound function* if

$$\inf\{F(x):\ x\in T\}\geq \mathbf{F}(T)$$

for all $T\subseteq S$.

In stochastic branch and bound an estimate $\widehat{\mathbf{F}}$ of $\mathbf{F}$ can be found for instance by sampling $\widehat{F}_n(x_i)$ for each x_i in T with $\mathsf{E}(\widehat{F}_n(x_i))=F(x_i)$ and setting

$$\widehat{\mathbf{F}}(T)=\inf\left\{\widehat{F}_n(x_i):\ x_i\in T\right\}.$$

The bound-step of the branch and bound method is replaced by a statistical test, whether the lower bound estimate of a branch is significantly larger than the estimate of an intermediate solution. After each step, all estimates are improved by taking additional observations. For details see [2] and [7].

In all these algorithms, common random numbers may decrease the variance.

See also

- ► Derivatives of Markov Processes and Their Simulation
- ► Derivatives of Probability and Integral Functions: General Theory and Examples
- ► Derivatives of Probability Measures
- ► Optimization in Operation of Electric and Energy Power Systems

References

1. Chernoff H (1989) Sequential analysis and optimal design. SIAM, Philadelphia
2. Ermoliev YM, Norkin VI, Ruszczyński A (1998) On optimal allocation of indivisibles under uncertainty. Oper Res 46:381–395
3. Futschik A, Pflug GCh (1997) Optimal allocation of simulation experiments in discrete stochastic optimization and approximative algorithms. Europ J Oper Res 101:245–260
4. Gutjahr W, Pflug G (1996) Simulated annealing for noisy cost functions. J Global Optim 8:1–13
5. Ho YC, Sreenivas RS, Vakili P (1992) Ordinal optimization of DEDS. J Discret Event Dynamical Systems 2:61–88
6. Kleywegt AJ, Shapiro A (1999) The sample average approximation method for stochastic discrete optimization. Georgia Inst Technol, Atlanta, GA
7. Norkin VI, Pflug GCh, Ruszczyński A (1998) A branch and bound method for stochastic global optimization. Math Program 83:425–450

Disease Diagnosis: Optimization-Based Methods

EVA K. LEE, TSUNG-LIN WU
Center for Operations Research in Medicine and HealthCare,
School of Industrial and Systems Engineering,
Georgia Institute of Technology,
Atlanta, USA

MSC2000: 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90, 90-08, 65K05

Article Outline

Abstract
Introduction
  Pattern Recognition, Discriminant Analysis, and Statistical Pattern Classification
  Supervised Learning, Training, and Cross-Validation
  Bayesian Inference and Classification
  Discriminant Functions
Mathematical Programming Approaches
  Linear Programming Classification Models
  Mixed Integer Programming Classification Models
  Nonlinear Programming Classification Models
  Support Vector Machine
Mixed Integer Programming Based Multigroup Classification Models and Applications to Medicine and Biology
  Discrete Support Vector Machine Predictive Models
  Classification Results for Real-World Biological and Medical Applications
  Further Advances
Progress and Challenges
Other Methods
Summary and Conclusion
References

Abstract

In this chapter, we present classification models based on mathematical programming approaches. We first provide an overview of various mathematical programming approaches, including linear programming, mixed integer programming, nonlinear programming, and support vector machines. Next, we present our effort of novel optimization-based classification models that are general purpose and suitable for developing predictive rules for large heterogeneous biological and medical data sets. Our predictive model simultaneously incorporates (1) the ability to classify any number of distinct groups; (2) the ability to incorporate heterogeneous types of attributes as input; (3) a high-dimensional data transformation that eliminates noise and errors in biological data; (4) the ability to incorporate constraints to limit the rate of misclassification, and a reserved-judgment region that provides a safeguard against overtraining (which tends to lead to high misclassification rates from the resulting predictive rule); and (5) successive multistage classification capability to handle data points placed in the reserved-judgment region. To illustrate the power and flexibility of the classification model and solution engine, and its multigroup prediction capability, application of the predictive model to a broad class of biological and medical problems is described. Applications include the differential diagnosis of the type of erythemato-squamous diseases; predicting presence/absence of heart disease; genomic analysis and prediction of aberrant CpG island methylation in human cancer; discriminant analysis of motility and morphology data in human lung carcinoma; prediction of ultrasonic cell disruption for drug delivery; identification of tumor shape and volume in treatment of sarcoma; multistage discriminant analysis of biomarkers for prediction of early atherosclerois; fingerprinting of native and angiogenic microvascular networks for early diagnosis of diabetes, aging, macular degeneracy, and tumor metastasis; prediction of protein localization sites; and pattern recognition of satellite images in classification of soil types. In all these applications, the predictive model yields correct classification rates ranging from 80 to 100%. This provides motivation for pursuing its use as a medical diagnostic, monitoring and decision-making tool.

Introduction

Classification is a fundamental machine learning task whereby rules are developed for the allocation of independent observations to groups. Classic examples of applications include medical diagnosis – the allocation of patients to disease classes on the basis of symptoms and laboratory tests – and credit screening – the acceptance or rejection of credit applications on the basis of applicant data. Data are collected concerning observa-

tions with known group membership. These *training data* are used to develop rules for the classification of future observations with unknown group membership.

In this introduction, we briefly describe some terminologies related to classification, and provide a brief description of the organization of this chapter.

Pattern Recognition, Discriminant Analysis, and Statistical Pattern Classification

Cognitive science is the science of learning, knowing, and reasoning. *Pattern recognition* is a broad field within *cognitive science*, which is concerned with the process of recognizing, identifying, and categorizing input information. These areas intersect with computer science, particularly in the closely related areas of *artificial intelligence*, *machine learning*, and *statistical pattern recognition*. Artificial intelligence is associated with constructing machines and systems that reflect human abilities in cognition. Machine learning refers to how these machines and systems replicate the learning process, which is often achieved by seeking and discovering patterns in data, or statistical pattern recognition.

Discriminant analysis is the process of discriminating between categories or populations. Associated with discriminant analysis as a statistical tool are the tasks of determining the features that best discriminate between populations, and the process of classifying new objects on the basis of these features. The former is often called *feature selection* and the latter is referred to as *statistical pattern classification*. This work will be largely concerned with the development of a viable statistical pattern classifier.

As with many computationally intensive tasks, recent advances in computing power have led to a sharp increase in the interest and application of discriminant analysis techniques. The reader is referred to Duda et al. [25] for an introduction to various techniques for pattern classification, and to Zopounidis and Doumpos [121] for examples of applications of pattern classification.

Supervised Learning, Training, and Cross-Validation

An *entity* or *observation* is essentially a data point as commonly understood in statistics. In the framework of statistical pattern classification, an entity is a set of quantitative measurements (or qualitative measurements expressed quantitatively) of *attributes* for a particular object. As an example, in medical diagnosis an entity could be the various blood chemistry levels of a patient. With each entity is associated one or more *groups* (or *populations*, *classes*, *categories*) to which it belongs. Continuing with the medical diagnosis example, the groups could be the various classes of heart disease. Statistical classification seeks to determine rules for associating entities with the groups to which they belong. Ideally, these associations align with the associations that human reasoning would produce on the basis of information gathered on objects and their apparent categories.

Supervised learning is the process of developing classification rules based on entities for which the classification is already known. Note that the process implies that the populations are already well defined. *Unsupervised learning* is the process of discovering patterns from unlabeled entities and thereby discovering and describing the underlying populations. Models derived using supervised learning can be used for both functions of discriminant analysis – feature selection and classification. The model that we consider is a method for supervised learning, so we assume that populations are previously defined.

The set of entities with known classification that is used to develop classification rules is the *training set*. The training set may be partitioned so that some entities are withheld during the model-development process, also known as the *training* of the model. The withheld entities form a *test set* that is used to determine the validity of the model, a process known as *cross-validation*. Entities from the test set are subjected to the rules of classification to measure the performance of the rules on entities with unknown group membership.

Validation of classification models is often performed using m-fold cross-validation where the data with known classification are partitioned into m *folds* (subsets) of approximately equal size. The classification model is trained m times, with the mth fold withheld during each run for testing. The performance of the model is evaluated by the classification accuracy on the m test folds, and can be represented using a *classification matrix* or *confusion matrix*.

The classification matrix is a square matrix with the number of rows and columns equal to the number of

groups. The *ij*th entry of the classification matrix contains the number or proportion of test entities from group i that were classified by the model as belonging to group j. Therefore, the number or proportion of correctly classified entities is contained in the diagonal elements of the classification matrix, and the number or proportion of misclassified entities is in the off-diagonal entries.

Bayesian Inference and Classification

The popularity of *Bayesian inference* has risen drastically over the past several decades, perhaps in part due to its suitability for statistical learning. The reader is referred to O'Hagan [92] for a thorough treatment of Bayesian inference. Bayesian inference is usually contrasted against *classical inference*, though in practice they often imply the same methodology.

The Bayesian method relies on a *subjective* view of probability, as opposed to the *frequentist* view upon which classical inference is based [92]. A subjective probability describes a degree of belief in a *proposition* held by the investigator based on some information. A frequency probability describes the likelihood of an *event* given an infinite number of trials.

In Bayesian statistics, inferences are based on the *posterior distribution*. The posterior distribution is the product of the *prior probability* and the *likelihood function*. The prior probability distribution represents the initial degree of belief in a proposition, often before empirical data are considered. The likelihood function describes the likelihood that the behavior is exhibited, given that the proposition is true. The posterior distribution describes the likelihood that the proposition is true, given the observed behavior.

Suppose we have a proposition or random variable θ about which we would like to make inferences, and data x. Application of Bayes's theorem gives

$$\mathrm{d}F(\theta|x) = \frac{\mathrm{d}F(\theta)\,\mathrm{d}F(x|\theta)}{\mathrm{d}F(x)}\,.$$

Here, F denotes the (cumulative) distribution function. For ease of conceptualization, assume that F is differentiable, then $\mathrm{d}F = f$, and the above equality can be rewritten as

$$f(\theta|x) = \frac{f(\theta)f(x|\theta)}{f(x)}\,.$$

For classification, a prior probability function $\pi(g)$ describes the likelihood that an entity is allocated to group g regardless of its exhibited feature values x. A group density function $f(x|g)$ describes the likelihood that an entity exhibits certain measurable attribute values, given that it belongs to population g. The posterior distribution for a group $P(g|x)$ is given by the product of the prior probability and group density function, normalized over the groups to obtain a unit probability over all groups. The observation x is allocated to group h if $h = \arg\max_{g\in\mathcal{G}} P(g|x) = \arg\max_{g\in\mathcal{G}} \frac{\pi(g)f(x|g)}{\sum_{j\in\mathcal{G}}\pi(j)f(x|j)}$, where $\mathcal{G}$ denotes the set of groups.

Discriminant Functions

Most classification methods can be described in terms of *discriminant functions*. A discriminant function takes as input an observation and returns information about the classification of the observation. For data from a set of groups $\mathcal{G}$, an observation x is assigned to group h if $h = \arg\max_{g\in\mathcal{G}} l_g(x)$, where the functions l_g are the discriminant functions. Classification methods restrict the form of the discriminant functions, and training data are used to determine the values of the parameters that define the functions.

The optimal classifier in the Bayesian framework can be described in terms of discriminant functions. Let $\pi_g = \pi(g)$ be the prior probability that an observation is allocated to group g and let $f_g(x) = f(x|g)$ be the likelihood that data x are drawn from population g. If we wish to minimize the probability of misclassification given x, then the optimal allocation for an entity is to the group $h = \arg\max_{g\in\mathcal{G}} P(g|x) = \arg\max_{g\in\mathcal{G}} \frac{\pi_g f_g(x)}{\sum_{j\in\mathcal{G}}\pi_j f_j(x)}$. Under the Bayesian framework,

$$P(g|x) = \frac{\pi_g f(x|g)}{f(x)} = \frac{\pi_g f(x|g)}{\sum\limits_{j\in\mathcal{G}} \pi_j f(x|j)}\,.$$

The discriminant functions can be $l_g(x) = P(g|x)$ for $g \in \mathcal{G}$. The same classification rule is given by $l_g(x) = \pi_g f(x|g)$ and $l_g(x) = \log f(x|g) + \log \pi_g$. The problem then becomes finding the form of the prior functions and likelihood functions that match the data.

If the data are multivariate normal with equal covariance matrices ($f(x|g) \sim N(\mu_g, \Sigma)$), then a linear discriminant function (LDF) is optimal:

$$\begin{aligned} l_g(x) &= \log f(x|g) + \log \pi_g \\ &= -1/2(x-\mu_g)^T \Sigma^{-1}(x-\mu_g) - 1/2 \log|\Sigma_g| \\ &\quad - d/2 \log 2\pi + \log \pi_g \\ &= w_g^T x + w_{g0}\,, \end{aligned}$$

where d is the number of attributes, $w_g = \Sigma^{-1}\mu_g$, and $w_{g0} = -1/2\mu_g^T \Sigma^{-1}\mu_g + \log \pi_g + x^T \Sigma^{-1} x - d/2 \log 2\pi$. Note that the last two terms of w_{g0} are constant for all g and need not be calculated. When there are two groups ($\mathcal{G} = \{1, 2\}$) and the priors are equal ($\pi_1 = \pi_2$), the discriminant rule is equivalent to Fisher's linear discriminant rule [30]. Fisher's rule can also be derived, as it was by Fisher, by choosing w so that $(w^T\mu_1 - w^T\mu_2)^2/(w^T \Sigma w)$ is maximized.

These LDFs and quadratic discriminant functions (QDFs) are often applied to data sets that are not multivariate normal or continuous (see pp. 234–235 in [98]) by using approximations for the means and covariances. Regardless, these models are *parametric* in that they incorporate assumptions about the distribution of the data. Fisher's LDF is *nonparametric* because no assumptions are made about the underlying distribution of the data. Thus, for a special case, a parametric and a nonparametric model coincide to produce the same discriminant rule. The LDF derived above is also called the *homoscedastic model*, and the QDF is called the *heteroscedastic model*. The exact form of discriminant functions in the Bayesian framework can be derived for other distributions [25].

Some classification methods are essentially methods for finding coefficients for LDFs. In other words, they seek coefficients w_g and constants w_{g0} such that $l_g(x) = w_g x + w_{g0}$, $g \in \mathcal{G}$ is an optimal set of discriminant functions. The criteria for optimality are different for different methods. LDFs project the data onto a linear subspace and then discriminate between entities in that subspace. For example, Fisher's LDF projects two-group data on an optimal line, and discriminates on that line. A good linear subspace may not exist for data with overlapping distributions between groups and therefore the data will not be classified accurately using these methods. The hyperplanes defined by the discriminant functions form boundaries between the group regions. A large portion of the literature concerning the use of mathematical programming models for classification describes methods for finding coefficients of LDFs [121].

Other classification methods seek to determine parameters to establish QDFs. The general form of a QDF is $l_g(x) = x^T W_g x + w_g^T x + w_{g0}$. The boundaries defining the group regions can assume any hyperquadric form, as can the Bayes decision rules for arbitrary multivariate normal distributions [25].

In this paper, we survey the development and advances of classification models via the mathematical programming techniques, and summarize our experience in classification models applied to prediction in biological and medical applications. The rest of this chapter is organized as follows. Section "Mathematical Programming Approaches" first provides a detailed overview of the development and advances of mathematical programming based classification models, including linear programming (LP), mixed integer programming (MIP), nonlinear programming, and support vector machine (SVM) approaches. In Sect. "Mixed Integer Programming Based Multigroup Classification Models and Applications to Medicine and Biology", we describe our effort in developing optimization-based multigroup multistage discriminant analysis predictive models for classification. The use of the predictive models for various biological and medical problems is presented. Section "Progress and Challenges" provides several tables to summarize the progress of mathematical programming based classification models and their characteristics. This is followed by a brief description of other classification methods in Sect. "Other Methods", and by a summary and concluding remarks in Sect. "Summary and Conclusion".

Mathematical Programming Approaches

Mathematical programming methods for statistical pattern classification emerged in the 1960s, gained popularity in the 1980s, and have grown drastically since. Most of the mathematical programming approaches are nonparametric, which has been cited as an advantage when analyzing contaminated data sets over methods that require assumptions about the distribution of the

data [107]. Most of the literature about mathematical programming methods is concerned with either using mathematical programming to determine the coefficients of LDFs or *support vector machines* (SVMs).

The following notation will be used. The subscripts i, j, and k are used for the observation, attribute, and group, respectively. Let x_{ij} be the value of attribute j of observation i. Let m be the number of attributes, K be the number of groups, G_k represent the set of data from group k, M be a big positive number, and ϵ be a small positive number. The abbreviation "urs" is used in reference to a variable to denote "unrestricted in sign."

Linear Programming Classification Models

The use of linear programs to determine the coefficients of LDFs has been widely studied [31,46,50,74]. The methods determine the coefficients for different objectives, including minimizing the sum of the distances to the separating hyperplane, minimizing the maximum distance of an observation to the hyperplane, and minimizing other measures of badness of fit or maximizing measures of goodness of fit.

Two-Group Classification One of the earliest LP classification models was proposed by Mangasarian [74] to construct a hyperplane to separate two groups of data. Separation by a nonlinear surface using LP was also proposed when the surface parameters appear linearly. Two sets of points may be inseparable by one hyperplane or surface through a single-step LP approach, but they can be strictly separated by more planes or surfaces via a multistep LP approach [75]. In [75] real problems with up to 117 data points, ten attributes, and three groups were solved. The three-group separation was achieved by separating group 1 from groups 2 and 3, and then group 2 from group 3.

Studies of LP models for the discriminant problem in the early 1980s were carried out by Hand [47], Freed and Glover [31,32], and Bajgier and Hill [5]. Three LP models for the two-group classification problem, including minimizing the sum of deviations (MSD), minimizing the maximum deviation (MMD), and minimizing the sum of interior distances (MSID) were proposed. Freed and Glover [33] provided computational studies of these models where the test conditions involved normal and nonnormal populations.

MSD:

$$\begin{aligned}
\text{Minimize} \quad & \sum_i d_i \\
\text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d_i \le 0 \quad \forall i \in G_1 , \\
& w_0 + \sum_j x_{ij} w_j + d_i \ge 0 \quad \forall i \in G_2 , \\
& w_j \text{ urs} \quad \forall j , \\
& d_i \ge 0 \quad \forall i .
\end{aligned}$$

MMD:

$$\begin{aligned}
\text{Minimize} \quad & d \\
\text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d \le 0 \quad \forall i \in G_1 , \\
& w_0 + \sum_j x_{ij} w_j + d \ge 0 \quad \forall i \in G_2 , \\
& w_j \text{ urs} \quad \forall j , \\
& d \ge 0 .
\end{aligned}$$

MSID:

$$\begin{aligned}
\text{Minimize} \quad & pd - \sum_i e_i \\
\text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d + e_i \le 0 \quad \forall i \in G_1 , \\
& w_0 + \sum_j x_{ij} w_j + d - e_i \ge 0 \quad \forall i \in G_2 , \\
& w_j \text{ urs} \quad \forall j , \\
& d \ge 0 , \\
& e_i \ge 0 \quad \forall i ,
\end{aligned}$$

where p is a weight constant.

The objective function of the MSD model is the L_1-norm distance, while the objective function of MMD is the L_∞-norm distance. They are special cases of L_p-norm classification [50,108].

In some models the constant term of the hyperplane is a fixed number instead of a decision variable. The model minimize the sum of deviations with constant cutoff score MSD^0 shown below is an example where the cutoff score b replaces w_0 in the formulation. The same replacement could be used in other formulations.

MSD^0:

$$\begin{aligned}
\text{Minimize} \quad & \sum_i d_i \\
\text{subject to} \quad & \sum_j x_{ij} w_j - d_i \le b \quad \forall i \in G_1 , \\
& \sum_j x_{ij} w_j + d_i \ge b \quad \forall i \in G_2 , \\
& w_j \text{ urs} \quad \forall j , \\
& d_i \ge 0 \quad \forall i .
\end{aligned}$$

A gap can be introduced between the two regions determined by the separating hyperplane to prevent degenerate solutions. Take MSD as an example; the separation constraints become

$$\begin{aligned}
& w_0 + \sum_j x_{ij} w_j - d_i \le -\epsilon \quad \forall i \in G_1 , \\
& w_0 + \sum_j x_{ij} w_j + d_i \ge \epsilon \quad \forall i \in G_2 .
\end{aligned}$$

The small number ϵ can be normalized to 1.

Besides introducing a gap, another normalization approach is to include constraints such as $\sum_{j=0}^{m} w_j = 1$ or $\sum_{j=1}^{m} w_j = 1$ in the LP models to avoid unbounded or trivial solutions.

Specifically, Glover et al. [45] gave the hybrid model, as follows.

Hybrid model:

$$\begin{aligned}
\text{Minimize} \quad & pd + \sum_i p_i d_i - qe - \sum_i q_i e_i \\
\text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d - d_i + e + e_i = 0 \\
& \quad \forall i \in G_1 , \\
& w_0 + \sum_j x_{ij} w_j + d + d_i - e - e_i = 0 \\
& \quad \forall i \in G_2 , \\
& w_j \text{ urs} \quad \forall j , \\
& d, e \ge 0 , \\
& d_i, e_i \ge 0 \quad \forall i ,
\end{aligned}$$

where p, p_i, q, q_i are the costs for different deviations. Including different combinations of deviation terms in the objective function then leads to variant models.

Joachimsthaler and Stam [50] reviewed and summarized LP formulations applied to two-group classification problems in discriminant analysis, including MSD, MMD, MSID, and MIP models, and the hybrid model. They summarized the performance of the LP methods together with the traditional classification methods such as Fisher's LDF [30], Smith's QDF [106], and a logistic discriminant method. In their review, MSD sometimes but not uniformly improves classification accuracy, compared with traditional methods. On the other hand, MMD is found to be inferior to MSD. Erenguc and Koehler [27] presented a unified survey of LP models and their experimental results, in which the LP models include several versions of MSD, MMD, MSID, and hybrid models. Rubin [99] provided experimental results comparing these LP models with Fisher's LDF and Smith's QDF. He concluded that QDF performs best when the data follow normal distributions and that QDF could be the benchmark when seeking situations for advantageous LP methods. In summary, the above mentioned review papers [27,50,99] describe previous work on LP classification models and their comparison with traditional methods. However, it is difficult to make definitive statements about the conditions under which one LP model is superior to others, as stated in [107].

Stam and Ungar [110] introduced the software package RAGNU, a utility program in conjunction with the LINDO optimization software, for solving two-group classification problems using LP-based methods. LP formulations such as MSD, MMD, MSID, hybrid models, and their variants are contained in the package.

There are some difficulties in LP-based formulations, in that some models could result in unbounded, trivial, or unacceptable solutions [34,87], but possible remedies are proposed. Koehler [51,52,53] and Xiao [114,115] characterized the conditions of unacceptable solutions in two-group LP discriminant models, including MSD, MMD, MISD, the hybrid model, and their variants. Glover [44] proposed the normalization constraint $\sum_{j=1}^{m}(-|G_2| \sum_{i \in G_1} x_{ij} + |G_1| \sum_{i \in G_2} x_{ij}) w_j = 1$, which is more effective and reliable. Rubin [100] examined the separation failure for two-group models and suggested applying the models twice, reversing the group designations the second time. Xiao and Feng [116] proposed a regularization method to avoid multiple solutions in LP discriminant analysis by adding the term $\epsilon \sum_{j=1}^{m} w_j^2$ in the objective functions.

Bennett and Mangasarian [9] proposed the following model which minimizes the average of the deviations, which is called robust LP (RLP):

$$\begin{aligned}
\text{Minimize} \quad & \frac{1}{|G_1|}\sum_{i\in G_1} d_i + \frac{1}{|G_2|}\sum_{i\in G_2} d_i \\
\text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d_i \le -1 \quad \forall i \in G_1\,, \\
& w_0 + \sum_j x_{ij} w_j + d_i \ge 1 \quad \forall i \in G_2\,, \\
& w_j \text{ urs} \quad \forall j\,, \\
& d_i \ge 0 \quad \forall i\,.
\end{aligned}$$

It is shown that this model gives the null solution $w_1 = \cdots = w_m = 0$ if and only if $\frac{1}{|G_1|}\sum_{i\in G_1} x_{ij} = \frac{1}{|G_2|}\sum_{i\in G_2} x_{ij}$ for all j, in which case the solution $w_1 = \cdots = w_m = 0$ is guaranteed to be not unique. Data of different diseases have been tested by the proposed classification methods, as in most of Mangasarian's papers.

Mangasarian et al. [86] described two applications of LP models in the field of breast cancer research, one in diagnosis and the other in prognosis. The first application is to discriminate benign from malignant breast lumps, while the second one is to predict when breast cancer is likely to recur. Both of them work successfully in clinical practice. The RLP model [9] together with the multisurface method tree algorithm [8] is used in the diagnostic system.

Duarte Silva and Stam [104] included the second-order (i. e., quadratic and cross-product) terms of the attribute values in the LP-based models such as MSD and hybrid models and compared them with linear models, Fisher's LDF, and Smith's QDF. The results of the simulation experiments show that the methods which include second-order terms perform much better than first-order methods, given that the data substantially violate the multivariate normality assumption. Wanarat and Pavur [113] investigated the effect of the inclusion of the second-order terms in the MSD, MIP, and hybrid models when the sample size is small to moderate. However, the simulation study shows that second-order terms may not always improve the performance of a first-order LP model even with data configurations that are more appropriately classified by Smith's QDF. Another result of the simulation study is that inclusion of the cross-product terms may hurt the model's accuracy, while omission of these terms causes the model to be not invariant with respect to a nonsingular transformation of the data.

Pavur [94] studied the effect of the position of the contaminated normal data in the two-group classification problem. The methods for comparison in that study included MSD, minimizing the number of misclassifications (MM; (described in the "Mixed Integer Programming Classification Models" section), Fisher's LDF, Smith's QDF, and nearest -neighbor models. The nontraditional methods such as LP models have potential for outperforming the standard parametric procedures when nonnormality is present, but this study shows that no one model is consistently superior in all cases.

Asparoukhov and Stam [3] proposed LP and MIP models to solve the two-group classification problem where the attributes are binary. In this case the training data can be partitioned into multinomial cells, allowing for a substantial reduction in the number of variables and constraints. The proposed models not only have the usual geometric interpretation, but also possess a strong probabilistic foundation. Let s be the index of the cells, n_{1s}, n_{2s} be the number of data points in cell s from groups 1 and 2, respectively, and $(b_{s1}, \ldots, b_{sm})$ be the binary digits representing cell s. The model shown below is the LP model of minimizing the sum of deviations for two-group classification with binary attributes.

Cell conventional MSD:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{s:\, n_{1s}+n_{2s}>0} (n_{1s} d_{1s} + n_{2s} d_{2s}) \\
\text{subject to} \quad & w_0 + \sum_j b_{sj} w_j - d_{1s} \le 0 \quad \forall s : n_{1s} > 0\,, \\
& w_0 + \sum_j b_{sj} w_j + d_{2s} > 0 \quad \forall s : n_{2s} > 0\,, \\
& w_j \text{ urs} \quad \forall j\,, \\
& d_{1s}, d_{2s} \ge 0 \quad \forall s\,.
\end{aligned}$$

Binary attributes are usually found in medical diagnoses data. In this study three real data sets of disease discrimination were tested: developing postoperative pulmonary embolism or not, having dissecting aneurysm or other diseases, and suffering from post-

traumatic epilepsy or not. In these data sets the MIP model for binary attributes (BMIP), which will be described later, performs better than other LP models or traditional methods.

Multigroup Classification Freed and Glover [32] extended the LP classification models from two-group to multigroup problems. One formulation which uses a single discriminant function is given below:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{k=1}^{K-1} c_k \alpha_k \\
\text{subject to} \quad & \sum_j x_{ij} w_j \le U_k \quad \forall i \in G_k \ \forall k \,, \\
& \sum_j x_{ij} w_j \ge L_k \quad \forall i \in G_k \ \forall k \,, \\
& U_k + \epsilon \le L_{k+1} + \alpha_k \\
& \quad \forall k = 1, \ \ldots \,, K-1 \,, \ w_j \text{ urs} \quad \forall j \,, \\
& U_k, L_k \text{ urs} \quad \forall k \,, \\
& \alpha_k \text{ urs} \quad \forall k = 1, \ \ldots \,, K-1 \,,
\end{aligned}
$$

where the number ϵ could be normalized to be 1, and c_k is the misclassification cost. However, single-function classification is not as flexible and general as multiple-function classification. Another extension from the two-group case to the multigroup case in [32] is to solve two-group LP models for all pairs of groups and determine classification rules based on these solutions. However, in some cases the group assignment is not clear and the resulting classification scheme may be suboptimal [107].

For the multigroup discrimination problem, Bennett and Mangasarian [10] defined the piecewise-linear separability of data from K groups as the following: The data from K groups are piecewise-linear-separable if and only if there exist $(w_0^k, w_1^k, \ \ldots \,, w_m^k) \in R^{m+1}, \ k = 1, \ \ldots \,, K$, such that $w_0^h + \sum_j x_{ij} w_j^h \ge w_0^k + \sum_j x_{ij} w_j^k + 1, \quad \forall i \in G_h \ \forall h, k \ne h$. The following LP will generate a piecewise-linear separation for the K groups if one exists, otherwise it will generate an error-minimizing separation:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_h \sum_{k \ne h} \frac{1}{|G_h|} \sum_{i \in G_h} d_i^{hk} \\
\text{subject to} \quad & d_i^{hk} \ge -(w_0^h + \sum_j x_{ij} w_j^h) \\
& \qquad + (w_0^k + \sum_j x_{ij} w_j^k) + 1 \\
& \qquad \forall i \in G_h \ \forall h, k \ne h \,, \\
& w_j^k \text{ urs} \quad \forall j, k \,, \\
& d_i^{hk} \ge 0 \quad \forall i \in G_h \ \forall h, \ k \ne h \,.
\end{aligned}
$$

The method was tested in three data sets. It performs pretty well in two of the data sets which are totally (or almost totally) piecewise-linear separable. The classification result is not good in the third data set, which is inherently more difficult. However, combining the multisurface method tree algorithm [8] results in an improvement in performance.

Gochet et al. [46] introduced an LP model for the general multigroup classification problem. The method separates the data with several hyperplanes by sequentially solving LPs. The vectors w^k, $k = 1, \ \ldots \,, K$, are estimated for the classification decision rule. The rule is to classify an observation i into group s, where $s = \arg\max_k \{w_0^k + \sum_j x_{ij} w_j^k\}$.

Suppose observation i is from group h. Denote the goodness of fit for observation i with respect to group k as

$$
\begin{aligned}
& G_{hk}^i(w^h, w^k) \\
& \quad = \left[\left(w_0^h + \sum_j x_{ij} w_j^h \right) - \left(w_0^k + \sum_j x_{ij} w_j^k \right) \right]^+ , \\
& \quad \text{where } [a]^+ = \max\{0, a\} \,.
\end{aligned}
$$

Likewise, denote the badness of fit for observation i with respect to group k as

$$
\begin{aligned}
& B_{hk}^i(w^h, w^k) \\
& \quad = \left[\left(w_0^h + \sum_j x_{ij} w_h^h \right) - \left(w_0^k + \sum_j x_{ij} w_j^k \right) \right]^- , \\
& \quad \text{where } [a]^- = -\min\{0, a\} \,.
\end{aligned}
$$

The total goodness of fit and total badness of fit are then defined as

$$
G(w) = G(w^1, \ \ldots \,, w^K) = \sum_h \sum_{k \ne h} \sum_{i \in G_h} G_{hk}^i(w^h, w^k) \,,
$$

$$
B(w) = B(w^1, \ \ldots \,, w^K) = \sum_h \sum_{k \ne h} \sum_{i \in G_h} B_{hk}^i(w^h, w^k) \,.
$$

The LP is to minimize the total badness of fit, subject to a normalization equation, in which $q > 0$:

$$\begin{aligned} &\text{Minimize} && B(w)\,, \\ &\text{subject to} && G(w) - B(w) = q\,, \\ &&& w \text{ urs}\,. \end{aligned}$$

Expanding $G(w)$ and $B(w)$ and substituting $G^i_{hk}(w^h, w^k)$ and $B^i_{hk}(w^h, w^k)$ by γ^i_{hk} and β^i_{hk} respectively, the LP becomes

$$\begin{aligned} &\text{Minimize} && \sum_h \sum_{k\neq h} \sum_{i\in G_h} \beta^i_{hk} \\ &\text{subject to} && \left(w^h_0 + \sum_j x_{ij} w^h_j\right) - \left(w^k_0 + \sum_j x_{ij} w^k_j\right) \\ &&& \quad = \gamma^i_{hk} - \beta^i_{hk} \quad \forall i \in G_h\ \forall h, k \neq h\,, \\ &&& \sum_h \sum_{k\neq h} \sum_{i\in G_h} (\gamma^i_{hk} - \beta^i_{hk}) = q\,, \\ &&& w^k_j \text{ urs} \quad \forall j, k\,, \\ &&& \gamma^i_{hk}, \beta^i_{hk} \geq 0 \quad \forall i \in G_h\ \forall h, k \neq h\,. \end{aligned}$$

The classification results for two real data sets show that this model can compete with Fisher's LDF and the nonparametric k-nearest-neighbor method.

The LP-based models for classification problems highlighted above are all nonparametric models. In Sect. "Mixed Integer Programming Based Multigroup Classification Models and Applications to Medicine and Biology", we describe LP-based and MIP-based classification models that utilize a parametric multigroup discriminant analysis approach [39,40,60,63]. These latter models have been employed successfully in various multigroup disease diagnosis and biological/medical prediction problems [16,28,29,56,57,59,60,64,65].

Mixed Integer Programming Classification Models

While LP offers a polynomial-time computational guarantee, MIP allows more flexibility in (among other things) modeling misclassified observations and/or misclassification costs.

Two-Group Classification In the two-group classification problem, binary variables can be used in the formulation to track and minimize the exact number of misclassifications. Such an objective function is also considered as the L_0-norm criterion [107].

MM:

$$\begin{aligned} &\text{Minimize} && \sum_i z_i \\ &\text{subject to} && w_0 + \sum_j x_{ij} w_j \leq M z_i \quad \forall i \in G_1\,, \\ &&& w_0 + \sum_j x_{ij} w_j \geq -M z_i \quad \forall i \in G_2\,, \\ &&& w_j \text{ urs} \quad \forall j\,, \\ &&& z_i \in \{0, 1\} \quad \forall i\,. \end{aligned}$$

The vector w is required to be a nonzero vector to prevent the trivial solution.

In the MIP formulation the objective function could include the deviation terms, such as those in the hybrid models, as well as the number of misclassifications [5]; or it could represent expected cost of misclassification [1,6,101,105]. In particular, there are some variant versions of the basic model.

Stam and Joachimsthaler [109] studied the classification performance of MM and compared it with that of MSD, Fisher's LDF, and Smith's QDF. In some cases the MM model performs better, but in some cases it does not. MIP formulations are in the review studies of Joachimsthaler and Stam [50] and Erenguc and Koehler [27], and are contained in the software developed by Stam and Ungar [110]. Computational experiments show that the MIP model performs better when the group overlap is higher [50,109], although it is still not easy to reach general conclusions [107].

Since the MIP model is $\mathcal{NP}$-hard, exact algorithms and heuristics are proposed to solve it efficiently. Koehler and Erenguc [54] developed a procedure to solve MM in which the condition of nonzero w is replaced by the requirement of at least one violation of the constraints $w_0 + \sum_j x_{ij} w_j \leq 0$ for $i \in G_1$ or $w_0 + \sum_j x_{ij} w_j \geq 0$ for $i \in G_2$. Banks and Abad [6] solved the MIP of minimizing the expected cost of misclassification by an LP-based algorithm. Abad and Banks [1] developed three heuristic procedures for the problem of minimizing the expected cost of misclas-

sification. They also included the interaction terms of the attributes in the data and applied the heuristics [7]. Duarte Silva and Stam [105] introduced the divide and conquer algorithm for the classification problem of minimizing the misclassification cost by solving MIP and LP subproblems. Rubin [101] solved the same problem by using a decomposition approach, and tested this procedure on some data sets, including two breast cancer data sets. Yanev and Balev [119] proposed exact and heuristic algorithms for solving MM, which are based on some specific properties of the vertices of a polyhedral set neatly connected with the model.

For the two-group classification problem where the attributes are binary, Asparoukhov and Stam [3] proposed LP and MIP models which partition the data into multinomial cells and result in fewer variables and constraints. Let s be the index of the cells, n_{1s}, n_{2s} be the number of data points in cell s from groups 1 and 2, respectively, and $(b_{s1}, \ldots, b_{sm})$ be the binary digits representing cell s. Below is the BMIP, which performs best in the three real data sets in [3]:

BMIP

$$\text{Minimize} \quad \sum_{s:\, n_{1s}+n_{2s}>0} \{|n_{1s} - n_{2s}| z_s + \min(n_{1s}, n_{2s})\}$$

$$\text{subject to} \quad w_0 + \sum_j b_{sj} w_j \le M z_s \quad \forall s : n_{1s} \ge n_{2s};$$

$$n_{1s} > 0\,,$$

$$w_0 + \sum_j b_{sj} w_j > -M z_s \quad \forall s : n_{1s} < n_{2s}\,,$$

$$w_j \text{ urs} \quad \forall j\,,$$

$$z_s \in \{0, 1\} \quad \forall s : \; n_{1s} + n_{2s} > 0\,.$$

Pavur et al. [96] included different secondary goals in model MM and compared their misclassification rates. A new secondary goal was proposed, which maximizes the difference between the means of the discriminant scores of the two groups. In this model the term $-\delta$ is added to the minimization objective function as a secondary goal with a constant multiplier, while the constraint $\sum_j \tilde{x}_j^{(2)} w_j - \sum_j \tilde{x}_j^{(1)} w_j \ge \delta$ is included, where $\tilde{x}_j^{(k)} = 1/|G_k| \; \sum_{i \in G_k} x_{ij} \; \forall j$, for $k = 1, 2$. The results of simulation study show that an MIP model with the proposed secondary goal has better performance than the other models studied.

Glen [42] proposed integer programming (IP) techniques for normalization in the two-group discriminant analysis models. One technique is to add the constraint $\sum_{j=1}^{m} |w_j| = 1$. In the proposed model, w_j for $j = 1, \ldots, m$ is represented by $w_j = w_j^+ - w_j^-$, where $w_j^+, w_j^- \ge 0$, and binary variables δ_j and γ_j are defined such that $\delta_j = 1 \Leftrightarrow w_j^+ \ge \epsilon$ and $\gamma_j = 1 \Leftrightarrow w_j^- \ge \epsilon$. The IP normalization technique is applied to MSD and MMD, and the MSD version is presented below.

MSD – with IP normalization:

$$\text{Minimize} \quad \sum_i d_i$$

$$\text{subject to} \quad w_0 + \sum_{j=1}^{m} x_{ij}(w_j^+ - w_j^-) - d_i \le 0$$

$$\forall i \in G_1\,,$$

$$w_0 + \sum_{j=1}^{m} x_{ij}(w_j^+ - w_j^-) + d_i \ge 0$$

$$\forall i \in G_2\,,$$

$$\sum_{j=1}^{m} (w_j^+ + w_j^-) = 1\,,$$

$$w_j^+ - \epsilon \delta_j \ge 0 \quad \forall j = 1, \ldots, m\,,$$

$$w_j^+ - \delta_j \le 0 \quad \forall j = 1, \ldots, m\,,$$

$$w_j^- - \epsilon \gamma_j \ge 0 \quad \forall j = 1, \ldots, m\,,$$

$$w_j^- - \gamma_j \le 0 \quad \forall j = 1, \ldots, m\,,$$

$$\delta_j + \gamma_j \le 1 \quad \forall j = 1, \ldots, m\,,$$

$$w_0 \text{ urs}\,,$$

$$w_j^+, w_j^- \ge 0 \quad \forall j = 1, \ldots, m\,,$$

$$d_i \ge 0 \quad \forall i\,,$$

$$\delta_j, \gamma_j \in \{0, 1\} \quad \forall j = 1, \ldots, m\,.$$

The variable coefficients of the discriminant function generated by the models are invariant under origin shifts. The proposed models were validated using two data sets from [45,87]. The models were also extended for attribute selection by adding the constraint $\sum_{j=1}^{m} (\delta_j + \gamma_j) = p$, which allows only a constant number, p, of attributes to be used for classification.

Glen [43] developed MIP models which determine the thresholds for forming dichotomous variables as well as the discriminant function coefficients, w_j. For each continuous attribute to be formed as a dichotomous attribute, the model finds the threshold among possible thresholds while determining the separating hyperplane and optimizing the objective function such as minimizing the sum of deviations or minimizing the number of misclassifications. Computational results of a real data set and some simulated data sets show that the MSD model with dichotomous categorical variable formation can improve classification performance. The reason for the potential of this technique is that the LDF generated is a nonlinear function of the original variables.

Multigroup Classification Gehrlein [41] proposed MIP formulations of minimizing the total number of misclassifications in the multigroup classification problem. He gave both a single-function classification scheme and a multiple-function classification scheme, as follows.

General single-function classification (GSFC) – minimizing the number of misclassifications:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_i z_i \\
\text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - M z_i \le U_k \quad \forall i \in G_k \ , \\
& w_0 + \sum_j x_{ij} w_j + M z_i \ge L_k \quad \forall i \in G_k \ , \\
& U_k - L_k \ge \delta' \quad \forall k \ , \\
& \left.\begin{array}{l} L_g - U_k + M y_{gk} \ge \delta \\ L_k - U_g + M y_{kg} \ge \delta \\ y_{gk} + y_{kg} = 1 \end{array}\right\} \forall g, k, \ g \ne k \ , \\
& w_j \text{ urs} \quad \forall j \ , \\
& U_k, L_k \text{ urs} \quad \forall k \ , \\
& z_i \in \{0, 1\} \quad \forall i \ , \\
& y_{gk} \in \{0, 1\} \quad \forall g, k, \ g \ne k \ ,
\end{aligned}
$$

where U_k, L_k denote the upper and lower endpoints of the interval assigned to group k, and $y_{gk} = 1$ if the interval associated with group g precedes that with group k and $y_{gk} = 0$ otherwise. The constant δ' is the minimum width of an interval of a group and the constant δ is the minimum gap between adjacent intervals.

General multiple-function classification (GMFC) – minimizing the number of misclassifications:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_i z_i \\
\text{subject to} \quad & w_0^h + \sum_j x_{ij} w_j^h - w_0^k \\
& \quad - \sum_j x_{ij} w_j^k + M z_i \ge \epsilon \\
& \quad \forall i \in G_h, \ \forall h, k \ne h \ , \\
& w_j^k \text{ urs} \quad \forall j, k \ , \\
& z_i \in \{0, 1\} \quad \forall i \ .
\end{aligned}
$$

Both models work successfully on the iris data set provided by Fisher [30].

Pavur [93] solved the multigroup classification problem by sequentially solving the GSFC in one dimension each time. LDFs were generated by successively solving the GSFC with the added constraints that all linear discriminants are uncorrelated to each other for the total data set. This procedure could be repeated for the number of dimensions that is believed to be enough. According to the simulation results, this procedure substantially improves the GSFC model and sometimes outperforms GMFC, Fisher's LDF, or Smith's QDF.

To solve the three-group classification problem more efficiently, Loucopoulos and Pavur [71] made a slight modification to the GSFC and proposed the model MIP3G, which also minimizes the number of misclassifications. Compared with GSFC, MIP3G is also a single-function classification model, but it reduces the possible group orderings from six to three in the formulation and thus becomes more efficient. Loucopoulos and Pavur [72] reported the results of a simulation experiment on the performance of GMFC, MIG3G, Fisher's LDF, and Smith's QDF for a three-group classification problem with small training samples. Second-order terms were also considered in the experiment. Simulation results show that GMFC and MIP3G can outperform the parametric procedures in some nonnormal data sets and that the inclusion of second-order terms can improve the performance

of MIP3G in some data sets. Pavur and Loucopoulos [95] investigated the effect of the gap size in the MIP3G model for the three-group classification problem. A simulation study illustrates that for fairly separable data, or data with small sample sizes, a non-zero-gap model can improve the performance. A possible reason for this result is that the zero-gap model may be overfitting the data.

Gallagher et al. [39,40,63] and Lee [59,60] proposed MIP models, both heuristic and exact, as a computational approach to solving the constrained discriminant method described by Anderson [2]. These models are described in detail in Sect. "Mixed Integer Programming Based Multigroup Classification Models and Applications to Medicine and Biology".

Nonlinear Programming Classification Models

Nonlinear programming approaches are natural extensions for some of the LP-based models. Thus far, nonlinear programming approaches have been developed for two-group classification.

Stam and Joachimsthaler [108] proposed a class of nonlinear programming methods to solve the two-group classification problem under the L_p-norm objective criterion. This is an extension of MSD and MMD, for which the objectives are the L_1-norm and L_∞-norm, respectively.

Minimize the general L_p-norm distance:

$$\begin{aligned}
\text{Minimize} \quad & \Big(\sum_i d_i^p\Big)^{1/p} \\
\text{subject to} \quad & \sum_j x_{ij} w_j - d_i \le b \quad \forall i \in G_1 , \\
& \sum_j x_{ij} w_j + d_i \ge b \quad \forall i \in G_2 , \\
& w_j \text{ urs} \quad \forall j , \\
& d_i \ge 0 \quad \forall i .
\end{aligned}$$

The simulation results show that, in addition to the L_1-norm and the L_∞-norm, it is worth the effort to compute other L_p-norm objectives. Restricting the analysis to $1 \le p \le 3$, plus $p = \infty$, is recommended. This method was reviewed by Joachimsthaler and Stam [50] and Erenguc and Koehler [27].

Mangasarian et al. [85] proposed a nonconvex model for the two-group classification problem:

$$\begin{aligned}
\text{Minimize} \quad & d^1 + d^2 \\
\text{subject to} \quad & \sum_j x_{ij} w_j - d^1 \le 0 \quad \forall i \in G_1 , \\
& \sum_j x_{ij} w_j + d^2 \ge 0 \quad \forall i \in G_2 , \\
& \max_{j=1,\dots,m} |w_j| = 1 , \\
& w_j \text{ urs} \quad \forall j , \\
& d^1, d^2 \text{ urs} ,
\end{aligned}$$

This model can be solved in polynomial-time by solving $2m$ linear programs, which generate a sequence of parallel planes, resulting in a piecewise-linear nonconvex discriminant function. The model works successfully in clinical practice for the diagnosis of breast cancer.

Further, Mangasarian [76] also formulated the problem of minimizing the number of misclassifications as a linear program with equilibrium constraints (LPEC) instead of the MIP model MM described previously:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{i \in G_1 \cup G_2} z_i \\
\text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d_i \le -1 \quad \forall i \in G_1 , \\
& z_i (w_0 + \sum_j x_{ij} w_j - d_i + 1) = 0 \\
& \qquad \forall i \in G_1 , \\
& w_0 + \sum_j x_{ij} w_j + d_i \ge 1 \quad \forall i \in G_2 , \\
& z_i (w_0 + \sum_j x_{ij} w_j + d_i - 1) = 0 \\
& \qquad \forall i \in G_2 , \\
& d_i (1 - z_i) = 0 \quad \forall i \in G_1 \cup G_2 , \\
& 0 \le z_i \le 1 \quad \forall i \in G_1 \cup G_2 , \\
& d_i \ge 0 \quad \forall i \in G_1 \cup G_2 , \\
& w_j \text{ urs} \quad \forall j .
\end{aligned}$$

The general LPEC can be converted to an exact penalty problem with a quadratic objective and linear

constraints. A stepless Frank–Wolfe-type algorithm is proposed for the penalty problem, terminating at a stationary point or a global solution. This method is called the parametric misclassification minimization (PMM) procedure, and numerical testing is included in [77].

To illustrate the next model, we first define the step function $s\colon R \to \{0, 1\}$ as

$$s(u) = \begin{cases} 1 & \text{if } u > 0\ , \\ 0 & \text{if } u \le 0\ . \end{cases}$$

The problem of minimizing the number of misclassifications is equivalent to

$$\begin{aligned} \text{Minimize} \quad & \sum_{i \in G_1 \cup G_2} s(d_i) \\ \text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d_i \le -1 \quad \forall i \in G_1\ , \\ & w_0 + \sum_j x_{ij} w_j + d_i \ge 1 \quad \forall i \in G_2\ , \\ & d_i \ge 0 \quad \forall i \in G_1 \cup G_2\ , \\ & w_j \text{ urs} \quad \forall j\ . \end{aligned}$$

Mangasarian [77] proposed a simple concave approximation of the step function for nonnegative variables: $t(u, \alpha) = 1 - e^{-\alpha u}$, where $\alpha > 0, u \ge 0$. Let $\alpha > 0$ and approximate $s(d_i)$ by $t(d_i, \alpha)$. The problem then reduces to minimizing a smooth concave function bounded below on a nonempty polyhedron, which has a minimum at a vertex of the feasible region. A finite successive linearization algorithm (SLA) was proposed, terminating at a stationary point or a global solution. Numerical tests of SLA were done and compared with the PMM procedure described above. The results show that the much simpler SLA obtains a separation that is almost as good as PMM in considerably less computing time.

Chen and Mangasarian [21] proposed an algorithm on a defined hybrid misclassification minimization problem, which is more computationally tractable than the $\mathcal{NP}$-hard misclassification minimization problem. The basic idea of the hybrid approach is to obtain iteratively w_0 and $(w_1, \ldots, w_m)$ of the separating hyperplane:

1. For a fixed w_0, solve RLP [9] to determine $(w_1, \ldots, w_m)$.
2. For this $(w_1, \ldots, w_m)$, solve the one-dimensional misclassification minimization problem to determine w_0.

Comparison of the hybrid method is made with the RLP method and the PMM procedure. The hybrid method performs better in the testing sets of the ten-fold cross-validation and is much faster than PMM.

Mangasarian [78] proposed the model of minimizing the sum of arbitrary-norm distances of misclassified points to the separating hyperplane. For a general norm $||\cdot||$ on R^m, the dual norm $||\cdot||'$ on R^m is defined as $||x||' = \max_{||y||=1} x^T y$. Define $[a]^+ = \max\{0, a\}$ and let $w = (w_1, \ldots, w_m)$. The formulation can then be written as

$$\begin{aligned} \text{Minimize} \quad & \sum_{i \in G_1} \Big[w_0 + \sum_j x_{ij} w_j \Big]^+ \\ & + \sum_{i \in G_2} \Big[-w_0 - \sum_j x_{ij} w_j \Big]^+ \\ \text{subject to} \quad & ||w||' = 1\ , \\ & w_0, w \text{ urs}\ . \end{aligned}$$

The problem is to minimize a convex function on a unit sphere. A decision problem related to this minimization problem is shown to be $\mathcal{NP}$-complete, except for $p = 1$. For a general p-norm, the minimization problem can be transformed via an exact penalty formulation to minimizing the sum of a convex function and a bilinear function on a convex set.

Support Vector Machine

A support vector machine (SVM) is a type of mathematical programming approach [112]. It has been widely studied, and has become popular in many application fields in recent years. The introductory description of SVMs given here is summarized from the tutorial by Burges [20]. In order to maintain consistency with SVM studies in published literature, the notation used below is slightly different from the notation used to describe the mathematical programming methods in earlier sections.

In the two-group separable case, the objective function is to maximize the margin of a separating hyper-

plane, $2/||w||$, which is equivalent to minimizing $||w||^2$:

$$\begin{aligned} \text{Minimize} \quad & w^T w \,, \\ \text{subject to} \quad & x_i^T w + b \geq +1 \quad \text{for } y_i = +1 \,, \\ & x_i^T w + b \leq -1 \quad \text{for } y_i = -1 \,, \\ & w, b \text{ urs} \,, \end{aligned}$$

where $x_i \in R^m$ represents the values of attributes of observation i and $y_i \in \{-1, 1\}$ represents the group of observation i.

This problem can be solved by solving its Wolfe dual problem:

$$\begin{aligned} \text{Maximize} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \,, \\ \text{subject to} \quad & \sum_i \alpha_i y_i = 0 \,, \\ & \alpha_i \geq 0 \quad \forall i \,. \end{aligned}$$

Here, α_i is the Lagrange multiplier for the training point i, and the points with $\alpha_i > 0$ are called the support vectors (analogous to the support of a hyperplane, and thus the introduction of the name "support vector"). The primal solution w is given by $w = \sum_i \alpha_i y_i x_i$. b can be computed by solving $y_i(w^T x_i + b) - 1 = 0$ for any i with $\alpha_i > 0$.

For the nonseparable case, slack variables ξ_i are introduced to handle the errors. Let C be the penalty for the errors. The problem becomes

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2} w^T w + C(\sum_i \xi_i)^k \,, \\ \text{subject to} \quad & x_i^T w + b \geq +1 - \xi_i \quad \text{for } y_i = +1 \,, \\ & x_i^T w + b \leq -1 + \xi_i \quad \text{for } y_i = -1 \,, \\ & w, b \text{ urs} \,, \\ & \xi_i \geq 0 \quad \forall i \,. \end{aligned}$$

When k is chosen to be 1, neither the ξ_i nor their Lagrange multipliers appear in the Wolfe dual problem:

$$\begin{aligned} \text{Maximize} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \,, \\ \text{subject to} \quad & \sum_i \alpha_i y_i = 0 \,, \\ & 0 \leq \alpha_i \leq C \quad \forall i \,. \end{aligned}$$

The data points can be separated nonlinearly by mapping the data into some higher-dimensional space and applying linear SVM to the mapped data. Instead of knowing explicitly the mapping Φ, SVM needs only the dot products of two transformed data points $\Phi(x_i) \cdot \Phi(x_j)$. The kernel function K is introduced such that $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$. Replacing $x_i^T x_j$ by $K(x_i, x_j)$ in the above problem, the separation becomes nonlinear, while the problem to be solved remains a quadratic program. In testing a new data point x after training, the sign of the function $f(x)$ is computed to determine the group of x:

$$f(x) = \sum_{i=1}^{N_s} \alpha_i y_i \Phi(s_i) \cdot \Phi(x) + b = \sum_{i=1}^{N_s} \alpha_i y_i K(s_i, x) + b,$$

where the s_i are the support vectors and N_s is the number of support vectors. Again the explicit form of $\Phi(x)$ is avoided.

Mangasarian provided a general mathematical programming framework for SVM, called generalized SVM or GSVM [79,83]. Special cases can be derived from GSVM, including the standard SVM.

Many SVM-type methods have been developed by Mangasarian and others to solve huge classification problems more efficiently. These methods include successive overrelaxation for SVM [82], proximal SVM [36,38], smooth SVM [68], reduced SVM [67], Lagrangian SVM [84], incremental SVMs [37], and other methods [13,81]. Mangasarian [80] summarized some of the developments. Examples of applications of SVM include breast cancer studies [69,70] and genome research [73].

Hsu and Lin [49] compared different methods for multigroup classification using SVMs. Three methods studied were based on several binary classifiers: one against one, one against all, and directed acyclic graph (DAG) SVM. The other two methods studied are methods with decomposition implementation. The experimental results show that the one-against-one and DAG methods are more suitable for practical use than the other methods. Lee et al. [66] proposed a generic approach to multigroup problems with some theoretical properties, and the proposed method was well applied to microarray data for cancer classification and satellite radiance profiles for cloud classification.

Gallagher et al. [39,40,63] offered the first discrete SVM for multigroup classification with reserved judgement. The approach has been successfully applied to

a diverse variety of biological and medical applications (see Sect. "Mixed Integer Programming Based Multigroup Classification Models and Applications to Medicine and Biology").

Mixed Integer Programming Based Multigroup Classification Models and Applications to Medicine and Biology

Commonly used methods for classification, such as LDFs, decision trees, mathematical programming approaches, SVMs, and artificial neural networks, can be viewed as attempts at approximating a *Bayes optimal rule* for classification; that is, a rule that maximizes (minimizes) the total probability of correct classification (misclassification). Even if a Bayes optimal rule is known, intergroup misclassification rates may be higher than desired. For example, in a population that is mostly healthy, a Bayes optimal rule for medical diagnosis might misdiagnose sick patients as healthy in order to maximize total probability of correct diagnosis. As a remedy, a constrained discriminant rule that limits the misclassification rate is appealing.

Assuming that the group density functions and prior probabilities are known, Anderson [2] showed that an optimal rule for the problem of maximizing the probability of correct classification subject to constraints on the misclassification probabilities must be of a specific form when discriminating among multiple groups with a simplified model. The formulae in Anderson's result depend on a set of parameters satisfying a complex relationship between the density functions, the prior probabilities, and the bounds on the misclassification probabilities. Establishing a viable mathematical model to describe Anderson's result, and finding values for these parameters that yield an optimal rule are challenging tasks. The first computational models utilizing Anderson's formulae were proposed in [39,40].

Discrete Support Vector Machine Predictive Models

As part of the work carried out at Georgia Institute of Technology's Center for Operations Research in Medicine, we have developed a general-purpose discriminant analysis modeling framework and computational engine that are applicable to a wide variety of applications, including biological, biomedical, and logistics problems. Utilizing the technology of large-scale discrete optimization and SVMs, we have developed novel classification models that simultaneously include the following features: (1) the ability to classify any number of distinct groups; (2) the ability to incorporate heterogeneous types of attributes as input; (3) a high-dimensional data transformation that eliminates noise and errors in biological data; (4) constraints to limit the rate of misclassification, and a reserved-judgment region that provides a safeguard against overtraining (which tends to lead to high misclassification rates from the resulting predictive rule); and (5) successive multistage classification capability to handle data points placed in the reserved-judgment region. Studies involving tumor volume identification, ultrasonic cell disruption in drug delivery, lung tumor cell motility analysis, CpG island aberrant methylation in human cancer, predicting early atherosclerosis using biomarkers, and fingerprinting native and angiogenic microvascular networks using functional perfusion data indicate that our approach is adaptable and can produce effective and reliable predictive rules for various biomedical and biobehavior phenomena [16,28,29,56,57,59,60,64,65].

Based on the description in [39,40,59,60,63], we summarize below some of the classification models we have developed.

Modeling of Reserved-Judgment Region for General Groups When the population densities and prior probabilities are known, the constrained rules with a reject option (reserved judgment), based on Anderson's results, call for finding a partition $\{R_0, \ldots, R_G\}$ of $\mathbb{R}^k$ that maximizes the probability of correct allocation subject to constraints on the misclassification probabilities; i. e.,

$$\text{Maximize} \quad \sum_{g=1}^{G} \pi_g \int_{R_g} f_g(w)\, dw \tag{1}$$

$$\text{subject to} \quad \int_{R_g} f_h(w) dw \le \alpha_{hg}, h, g = 1, \ldots, G, \quad h \ne g\,, \tag{2}$$

where $f_h, h \in \{1, \ldots, G\}$, are the group conditional density functions, π_g denotes the prior probability that a randomly selected entity is from group $g, g \in \{1, \ldots, G\}$, and $\alpha_{hg}, h \ne g$, are constants between 0 and 1. Under quite general assumptions, it was shown

that there exist unique (up to a set of measure zero) nonnegative constants $\lambda_{ih}, i, h \in \{1, \ldots, G\}, i \neq h$, such that the optimal rule is given by

$$R_g = \{x \in \mathbb{R}^k : L_g(x) = \max_{h \in \{0,1,\ldots,G\}} L_h(x)\}, \quad g = 0, \ldots, G, \tag{3}$$

where

$$L_0(x) = 0, \tag{4}$$

$$L_h(x) = \pi_h f_h(x) - \sum_{i=1, i \neq h}^{G} \lambda_{ih} f_i(x), h = 1, \ldots, G. \tag{5}$$

For $G = 2$ the optimal solution can be modeled rather straightforwardly. However, finding optimal λ_{ih}'s for the general case, $G \geq 3$, is a difficult problem, with the difficulty increasing as G increases. Our model offers an avenue for modeling and finding the optimal solution in the general case. It is the first such model to be computationally viable [39,40].

Before proceeding, we note that R_g can be written as $R_g = \{x \in \mathbb{R}^k : L_g(x) \geq L_h(x) \text{ for all } h = 0, \ldots, G\}$. So, since $L_g(x) \geq L_h(x)$ if, and only if, $(1/\sum_{t=1}^{G} f_t(x)) L_g(x) \geq (1/\sum_{t=1}^{G} f_t(x)) L_h(x)$, the functions $L_h, h = 1, \ldots, G$, can be redefined as

$$L_h(x) = \pi_h p_h(x) - \sum_{i=1, i \neq h}^{G} \lambda_{ih} p_i(x), h = 1, \ldots, G, \tag{6}$$

where $p_i(x) = f_i(x)/\sum_{t=1}^{G} f_t(x)$. We assume that L_h is defined as in (6) in our model.

Mixed Integer Programming Formulations Assume that we are given a training sample of N entities whose group classifications are known; say, n_g entities are in group g, where $\sum_{g=1}^{G} n_g = N$. Let the k-dimensional vectors x^{gj}, $g = 1, \ldots, G, j = 1, \ldots, n_g$, contain the measurements on k available characteristics of the entities. Our procedure for deriving a discriminant rule proceeds in two stages. The first stage is to use the training sample to compute estimates, $\hat{f}_h$, either parametrically or nonparametrically, of the density functions f_h [89] and estimates, $\hat{\pi}_h$, of the prior probabilities $\pi_h, h = 1, \ldots, G$. The second stage is to determine the optimal λ_{ih}'s given these estimates. This stage requires being able to estimate the probabilities of correct classification and misclassification for any candidate set of λ_{ih}'s. One could, in theory, substitute the estimated densities and prior probabilities into (5), and directly use the resulting regions R_g in the integral expressions given in (1) and (2). This would involve, even in simple cases such as normally distributed groups, the numerical evaluation of k-dimensional integrals at each step of a search for the optimal λ_{ih}'s. Therefore, we have designed an alternative approach. After substituting the $\hat{f}_h$'s and $\hat{\pi}_h$'s into (5), we simply calculate the proportion of training sample points which fall in each of the regions $R_1, \ldots, R_G$. The MIP models discussed below attempt to maximize the proportion of training sample points correctly classified while satisfying constraints on the proportions of training sample points misclassified. This approach has two advantages. First, it avoids having to evaluate the potentially difficult integrals in (1) and (2). Second, it is nonparametric in controlling the training sample misclassification probabilities. That is, even if the densities are poorly estimated (by assuming, for example, normal densities for non-normal data), the constraints are still satisfied for the training sample. Better estimates of the densities may allow a higher correct classification rate to be achieved, but the constraints will be satisfied even if poor estimates are used. Unlike most SVM models that minimize the sum of errors, our objective is driven by the number of correct classifications, and will not be biased by the distance of the entities from the supporting hyperplane.

A word of caution is in order. In traditional unconstrained discriminant analysis, the true probability of correct classification of a given discriminant rule tends to be smaller than the rate of correct classification for the training sample from which it was derived. One would expect to observe such an effect for the method described herein as well. In addition, one would expect to observe an analogous effect with regard to constraints on misclassification probabilities – the true probabilities are likely to be greater than any limits imposed on the proportions of training sample misclassifications. Hence, the α_{hg} parameters should be carefully chosen for the application in hand.

Our first model is a nonlinear 0/1 MIP model with the nonlinearity appearing in the constraints. Model 1

maximizes the number of correct classifications of the given N training entities. Similarly, the constraints on the misclassification probabilities are modeled by ensuring that the number of group g training entities in region R_h is less than or equal to a prespecified percentage, $\alpha_{hg}(0 < \alpha_{hg} < 1)$, of the total number, n_g, of group g entities, $h, g \in \{1, \ldots, G\}, h \neq g$.

For notational convenience, let $\mathbf{G} = \{1, \ldots, G\}$ and $\mathbf{N}_g = \{1, \ldots, n_g\}$, for $g \in \mathbf{G}$. Also, analogous to the definition of p_i, define $\hat{p}_i$ by $\hat{p}_i = \hat{f}_i(x)/\sum_{t=1}^{G} \hat{f}_t(x)$. In our model, we use binary indicator variables to denote the group classification of entities. Mathematically, let u_{hgj} be a binary variable indicating whether or not x^{gj} lies in region R_h; i. e., whether or not the jth entity from group g is allocated to group h. Then model 1 can be written as follows.

Discriminant analysis MIP (DAMIP):

$$\begin{aligned} \text{Maximize} \quad & \sum_{g\in G}\sum_{j\in N_g} u_{ggj} \\ \text{subject to} \quad & L_{hgj} = \hat{\pi}_h \hat{p}_h(x^{gj}) - \sum_{i\in G\setminus h} \lambda_{ih}\hat{p}_i(x^{gj}), \\ & h, g \in \mathbf{G}, j \in \mathbf{N}_g, \end{aligned} \tag{7}$$

$$y_{gj} = \max\{0, L_{hgj} : h = 1, \ldots, G\}, \quad g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{8}$$

$$y_{gj} - L_{ggj} \leq M(1 - u_{ggj}), \quad g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{9}$$

$$y_{gj} - L_{hgj} \geq \varepsilon(1 - u_{hgj}), \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g, h \neq g, \tag{10}$$

$$\sum_{j\in N_g} u_{hgj} \leq \lfloor \alpha_{hg} n_g \rfloor, \quad h, g \in \mathbf{G}, \ h \neq g, \tag{11}$$

$$-\infty < L_{hgj} < \infty, y_{gj} \geq 0, \lambda_{ih} \geq 0, u_{hgj} \in \{0, 1\}.$$

Constraint (7) defines the variable L_{hgj} as the value of the function L_h evaluated at x^{gj}. Therefore, the continuous variable y_{gj}, defined in constraint (8), represents $\max\{L_h(x^{gj}) : h = 0, \ldots, G\}$; and consequently, x^{gj} lies in region R_h if, and only if, $y_{gj} = L_{hgj}$. The binary variable u_{hgj} is used to indicate whether or not x^{gj} lies in region R_h; i. e., whether or not the jth entity from group g is allocated to group h. In particular, constraint (9), together with the objective, forces u_{ggj} to be 1 if, and only if, the jth entity from group g is correctly allocated to group g; and constraints (10) and (11) ensure that at most $\lfloor \alpha_{hg} n_g \rfloor$ (i. e., the greatest integer less than or equal to $\alpha_{hg} n_g$) group g entities are allocated to group $h, h \neq g$. One caveat regarding the indicator variables u_{hgj} is that although the condition $u_{hgj} = 0, h \neq g$, implies (by constraint (10)) that $x^{gj} \notin R_h$, the converse need not hold. As a consequence, the number of misclassifications may be overcounted. However, in our preliminary numerical study we found that the actual amount of overcounting is minimal. One could force the converse (thus, $u_{hgj} = 1$ if and only if $x^{gj} \in R_h$) by adding constraints $y_{gj} - L_{hgj} \leq M(1 - u_{hgj})$, for example. Finally, we note that the parameters M and ϵ are extraneous to the discriminant analysis problem itself, but are needed in the model to control the indicator variables u_{hgj}. The intention is for M and ϵ to be, respectively, large and small positive constants.

Model Variations We explore different variations in the model to grasp the quality of the solution and the associated computational effort.

A first variation involves transforming model 1 to an equivalent linear mixed integer model. In particular, model 2 replaces the N constraints defined in (8) with the following system of $3GN + 2N$ constraints:

$$y_{gj} \geq L_{hgj}, \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{12}$$

$$\tilde{y}_{hgj} - L_{hgj} \leq M(1 - v_{hgj}), \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{13}$$

$$\tilde{y}_{hgj} \leq \hat{\pi}_h \hat{p}_h(x^{gj}) v_{hgj}, \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{14}$$

$$\sum_{h\in G} v_{hgj} \leq 1, \quad g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{15}$$

$$\sum_{h\in G} \tilde{y}_{hgj} = y_{gj}, \quad g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{16}$$

where $\tilde{y}_{hgj} \geq 0$ and $v_{hgj} \in \{0, 1\}, h, g \in \mathbf{G}, j \in \mathbf{N}_g$. These constraints, together with the nonnegativity of y_{gj} force $y_{gj} = \max\{0, L_{hgj} : h = 1, \ldots, G\}$.

The second variation involves transforming model 1 to a heuristic linear MIP model. This is done by replacing the nonlinear constraint (8) with $y_{gj} \geq L_{hgj}, h, g \in \mathbf{G}, j \in \mathbf{N}_g$, and including penalty terms in the objective function. In particular, model 3

has the objective

$$\text{Maximize} \sum_{g\in G}\sum_{j\in N_g} \beta u_{ggj} - \sum_{g\in G}\sum_{j\in N_g} \gamma y_{gj} ,$$

where β and γ are positive constants. This model is heuristic in that there is nothing to force $y_{gj} = \max\{0, L_{hgj}: h = 1, \ldots, G\}$. However, since in addition to trying to force as many u_{ggj}'s to 1 as possible, the objective in model 3 also tries to make the y_{gj}'s as small as possible, and the optimizer tends to drive y_{gj} towards $\max\{0, L_{hgj}: h = 1, \ldots, G\}$. We remark that β and γ could be stratified by group (i. e., introduce possibly distinct $\beta_g, \gamma_g, g \in \mathbf{G}$) to model the relative importance of certain groups to be correctly classified.

A reasonable modification to models 1, 2, and 3 involves relaxing the constraints specified by (11). Rather than placing restrictions on the number of type g training entities classified into group h, for all $h, g \in \mathbf{G}, h \neq g$, one could simply place an upper bound on the *total* number of misclassified training entities. In this case, the $G(G-1)$ constraints specified by (11) would be replaced by the single constraint

$$\sum_{g\in G}\sum_{h\in G\setminus\{g\}}\sum_{j\in N_g} u_{hgj} \leq \lfloor \alpha N \rfloor , \tag{17}$$

where α is a constant between 0 and 1. We will refer to models 1, 2, and 3 modified in this way as models 1T, 2T, and 3T, respectively. Of course, other modifications are also possible. For instance, one could place restrictions on the total number of type g points misclassified for each $g \in \mathbf{G}$. Thus, in place of the constraints specified in (17), one would include the constraints $\sum_{h\in G\setminus\{g\}}\sum_{j\in N_g} u_{hgj} \leq \lfloor \alpha_g N \rfloor$, $g \in \mathbf{G}$, where $0 < \alpha_g < 1$.

We also explore a heuristic linear model of model 1. In particular, consider the linear program (DALP):

$$\text{Maximize} \quad \sum_{g\in G}\sum_{j\in N_g} (c_1 w_{gj} + c_2 y_{gj}) \tag{18}$$

$$\text{subject to} \quad L_{hgj} = \pi_h \hat{p}_h(x^{gj}) - \sum_{i\in G\setminus h} \lambda_{ih}\hat{p}_i(x^{gj}) ,$$

$$h, g \in \mathbf{G}, j \in \mathbf{N}_g , \tag{19}$$

$$L_{ggj} - L_{hgj} + w_{gj} \geq 0, \quad h, g \in \mathbf{G}, h \neq g, j \in \mathbf{N}_g, \tag{20}$$

$$L_{ggj} + w_{gj} \geq 0, \quad g \in \mathbf{G}, j \in \mathbf{N}_g , \tag{21}$$

$$-L_{hgj} + y_{gj} \geq 0, \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g , \tag{22}$$

$$-\infty < L_{hgj} < \infty, w_{gj}, y_{gj}, \lambda_{ih} \geq 0 .$$

Constraint (19) defines the variable L_{hgj} as the value of the function L_h evaluated at x^{gj}. As the optimization solver searches through the set of feasible solutions, the λ_{ih} variables will vary, causing the L_{hgj} variables to assume different values. Constraints (20), (21), and (22) link the objective-function variables with the L_{hgj} variables in such a way that correct classification of training entities and allocation of training entities into the reserved-judgment region are captured by the objective-function variables. In particular, if the optimization solver drives w_{gj} to zero for some g,j pair, then constraints (20) and (21) imply that $L_{ggj} = \max\{0, L_{hgj}: h \in \mathbf{G}\}$. Hence, the jth entity from group g is correctly classified. If, on the other hand, the optimal solution yields $y_{gj} = 0$ for some g,j pair, then constraint (22) implies that $\max\{0, L_{hgj}: h \in \mathbf{G}\} = 0$. Thus, the jth entity from group g is placed in the reserved-judgment region. (Of course, it is possible for both w_{gj} and y_{gj} to be zero. One should decide prior to solving the linear program how to interpret the classification in such cases.) If both w_{gj} and y_{gj} are positive, the jth entity from group g is misclassified.

The optimal solution yields a set of λ_{ih}'s that best allocates the training entities (i. e., "best" in terms of minimizing the penalty objective function). The optimal λ_{ih}'s can then be used to define the functions L_h, $h \in G$, which in turn can be used to classify a new entity with feature vector $x \in \mathbb{R}^k$ by simply computing the index at which $\max\{L_h(x): h \in \{0, 1, \ldots, G\}\}$ is achieved.

Note that model DALP places no a priori bound on the number of misclassified training entities. However, since the objective is to minimize a weighted combination of the variables w_{gj} and y_{gj}, the optimizer will attempt to drive these variables to zero. Thus, the optimizer is, in essence, attempting either to correctly classify training entities ($w_{gj} = 0$), or to place them in the reserved-judgment region ($y_{gj} = 0$). By varying the weights c_1 and c_2, one has a means of controlling the optimizer's emphasis for correctly classifying training entities versus placing them in the reserved-

Disease Diagnosis: Optimization-Based Methods, Table 1
Model size

Model	Type	Constraints	Total variables	0/1 Variables
1	Nonlinear MIP	$2GN + N + G(G-1)$	$2GN + N + G(G-1)$	GN
2	Linear MIP	$5GN + 2N + G(G-1)$	$4GN + N + G(G-1)$	$2GN$
3	Linear MIP	$3GN + G(G-1)$	$2GN + N + G(G-1)$	GN
1T	Nonlinear MIP	$2GN + N + 1$	$2GN + N + G(G-1)$	GN
2T	Linear MIP	$5GN + 2N + 1$	$4GN + N + G(G-1)$	$2GN$
3T	Linear MIP	$3GN + 1$	$2GN + N + G(G-1)$	GN
DALP	Linear program	$3GN$	$NG + N + G(G-1)$	0

judgment region. If $c_2/c_1 < 1$, the optimizer will tend to place a greater emphasis on driving the w_{gj} variables to zero than driving the y_{gj} variables to zero (conversely, if $c_2/c_1 > 1$). Hence, when $c_2/c_1 < 1$, one should expect to get relatively more entities correctly classified, fewer placed in the reserved-judgment region, and more misclassified, than when $c_2/c_1 > 1$. An extreme case is when $c_2 = 0$. In this case, there is no emphasis on driving y_{gj} to zero (the reserved-judgment region is thus ignored), and the full emphasis of the optimizer is to drive w_{gj} to zero.

Table 1 summarizes the number of constraints, the total number of variables, and the number of 0/1 variables in each of the discrete SVM models, and in the heuristic LP model (DALP). Clearly, even for moderately sized discriminant analysis problems, the MIP instances are relatively large. Also, note that model 2 is larger than model 3, in terms of both the number of constraints and the number of variables. However, it is important to keep in mind that the difficulty of solving an MIP problem cannot, in general, be predicted solely by its size; problem structure has a direct and substantial bearing on the effort required to find optimal solutions. The LP relaxation of these MIP models poses computational challenges as commercial LP solvers return (optimal) LP solutions that are infeasible, owing to the equality constraints, and the use of big M and small ϵ in the formulation.

It is interesting to note that the set of feasible solutions for model 2 is "tighter" than that for model 3. In particular, if F_i denotes the set of feasible solutions of model i, then

$$F_1 = \{(L, \lambda, u, y): \text{ there exists } \tilde{y}, v \text{ such that } (L, \lambda, u, y, \tilde{y}, v) \in F_2\} \subsetneq F_3 . \quad (23)$$

The novelties of the classification models developed herein include the following: (1) they are suitable for discriminant analysis given any number of groups, (2) they accept heterogeneous types of attributes as input, (3) they use a parametric approach to reduce high-dimensional attribute spaces, and (4) they allow constraints on the number of misclassifications, and utilize a reserved judgment to facilitate the reduction of misclassifications. The lattermost point opens the possibility of performing multistage analysis.

Clearly, the advantage of an LP model over an MIP model is that the associated problem instances are computationally much easier to solve. However, the most important criterion in judging a method for obtaining discriminant rules is how the rules perform in correctly classifying new unseen entities. Once the rule has been developed, applying it to a new entity to determine its group is trivial. Extensive computational experiments have been performed to gauge the qualities of solutions of different models [17,19,40,59,60,63].

Validation of Model and Computational Effort We performed tenfold cross-validation, and designed simulation and comparison studies on our models. The results reported in [40,63] demonstrate that our approach works well when applied to both simulated data and data sets from the machine learning database repository [91]. In particular, our methods compare favorably and at times superior to other mathematical programming methods, including the GSFC model by Gehrlein [41], and the LP model by Gochet et al. [46], as well as Fisher's LDF, artificial neural networks, quadratic discriminant analysis, tree classification, and other SVMs, on real biological and medical data.

Classification Results for Real-World Biological and Medical Applications

The main objective in discriminant analysis is to derive rules that can be used to classify entities into groups. Computationally, the challenge lies in the effort expended to develop such a rule. Once the rule has been developed, applying it to a new entity to determine its group is trivial. Feasible solutions obtained from our classification models correspond to predictive rules. Empirical results [40,63] indicate that the resulting classification model instances are computationally very challenging, and even intractable by competitive commercial MIP solvers. However, the resulting predictive rules prove to be very promising, offering correct classification rates on new unknown data ranging from 80 to 100% for various types of biological/medical problems. Our results indicate that the general-purpose classification framework that we have designed has the potential to be a very powerful predictive method for clinical settings.

The choice of MIP as the underlying modeling and optimization technology for our SVM classification model is guided by the desire to simultaneously incorporate a variety of important and desirable properties of predictive models within a general framework. MIP itself allows for incorporation of continuous and discrete variables, and linear and nonlinear constraints, providing a flexible and powerful modeling environment.

Our mathematical modeling and computational algorithm design shows great promise as the resulting predictive rules are able to produce higher rates of correct classification for new biological data (with unknown group status) compared with existing classification methods. This is partly due to the transformation of raw data via the set of constraints in (7). While most mathematical programming approaches directly determine the hyperplanes of separation using raw data, our approach transforms the raw data via a probabilistic model, before the determination of the supporting hyperplanes. Further, the separation is driven by maximizing the sum of binary variables (representing correct classification or not of entities), instead of maximizing the margins between groups, or minimizing a sum of errors (representing distances of entities from hyperplanes), as in other SVMs. The combination of these two strategies offers better classification capability. Noise in the transformed data is not as profound as in raw data. And the magnitudes of the errors do not skew the determination of the separating hyperplanes, as all entities have *equal* importance when correct classification is being counted.

To highlight the broad applicability of our approach, below we briefly summarize the application of our predictive models and solution algorithms to ten different biological problems. Each of the projects was carried out in close partnership with experimental biologists and/or clinicians. Applications to finance and other industry applications are described elsewhere [17,40,63].

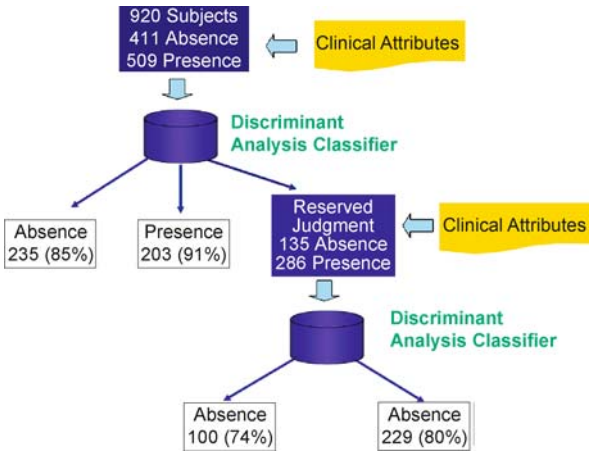
Determining the Type of Erythemato-Squamous Disease The differential diagnosis of erythemato-squamous diseases is an important problem in dermatology [60]. They all share the clinical features of erythema and scaling, with very little differences. The six groups are psoriasis, seboreic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis, and pityriasis rubra pilaris. Usually a biopsy is necessary for the diagnosis but unfortunately these diseases share many histopathological features as well. Another difficulty for the differential diagnosis is that a disease may show the features of another disease at the beginning stage and may have the characteristic features at the following stages [91].

The six groups consisted of 366 subjects (112, 61, 72, 49, 52, and 20 respectively) with 34 clinical attributes. Patients were first evaluated clinically with 12 features. Afterwards, skin samples were taken for the evaluation of 22 histopathological features. The values of the histopathological features were determined by an analysis of the samples under a microscope. The 34 attributes include (1) clinical attributes (erythema, scaling, definite borders, itching, koebner phenomenon, polygonal papules, follicular papules, oral mucosal involvement, knee and elbow involvement, scalp involvement, family history, age) and (2) histopathological attributes (melanin incontinence, eosinophils in the infiltrate, polymorphonuclear leukocyte infiltrate, fibrosis of the papillary dermis, exocytosis, acanthosis, hyperkeratosis, parakeratosis, clubbing of the rete ridges, elongation of the rete ridges, thinning of the suprapapillary epidermis, spongiform pustule, Munro microabscess, focal hypergranulosis, disappearance of the granular

layer, vacuolization and damage of basal layer, spongiosis, sawtooth appearance of retes, follicular horn plug, perifollicular parakeratosis, inflammatory monoluclear infiltrate, band-like infiltrate).

Our multigroup classification model selected 27 discriminatory attributes, and successfully classified the patients into six groups, each with an unbiased correct classification of greater than 93% (with 100% correct rate for groups 1, 3, 5, and 6) with an average overall accuracy of 98%. Using 250 subjects to develop the rule, and testing the remaining 116 patients, we obtained a prediction accuracy of 91%.

Predicting Presence/Absence of Heart Disease The four databases concerning heart disease diagnosis were collected by Dr. Andras Janosi of the Hungarian Institute of Cardiology, Budapest; Dr. William Steinbrunn of University Hospital, Zurich; Dr. Matthias Pfisterer of University Hospital, Basel; and Dr. Robert Detrano of V.A. Medical Center, Long Beach, and Cleveland Clinic Foundation [60]. Each database contains the same 76 attributes. The "goal" field refers to the presence of heart disease in the patient. The classification attempts to distinguish presence (values 1, 2, 3, 4, involving a total of 509 subjects) from absence (value 0, involving 411 subjects) [91]. The attributes include demographics, physiocardiovascular conditions, traditional risk factors, family history, personal lifestyle, and cardiovascular exercise measurements. This data set has posed some challenges to past analysis via various classification approaches, resulting in less than 80% correct classification. Applying our classification model without reserved judgment, we obtained 79 and 85% correct classification for each group respectively. To determine the usefulness of multistage analysis, we applied two-stage classification. In the first stage, 14 attributes were selected as discriminatory. One hundred and thirty-five group absence subjects were placed into the reserved-judgment region, with 85% of the remaining being classified as group absence correctly; while 286 group presence subjects were placed into the reserved-judgment region, and 91% of the remaining were classified correctly into the group presence. In the second stage, 11 attributes were selected with 100 and 229 classified into group absence and presence respectively. Combining the two stages, we obtained a correct classification of 82 and 85%, respectively, for diagnosis of absence or presence of heart disease. Figure 1 illustrates the two-stage classification.



Disease Diagnosis: Optimization-Based Methods, Figure 1
A tree diagram for two-stage classification and prediction of heart disease

Predicting Aberrant CpG Island Methylation in Human Cancer More details of this work can be found in [28,29]. Epigenetic silencing associated with aberrant methylation of promoter-region CpG islands is one mechanism leading to loss of tumor suppressor function in human cancer. Profiling of CpG island methylation indicates that some genes are more frequently methylated than others, and that each tumor type is associated with a unique set of methylated genes. However, little is known about why certain genes succumb to this aberrant event. To address this question, we used restriction landmark genome scanning (RLGS) to analyze the susceptibility of 1749 unselected CpG islands to de novo methylation driven by overexpression of DNMT1. We found that whereas the overall incidence of CpG island methylation was increased in cells overexpressing DNMT1, not all loci were equally affected. The majority of CpG islands (69.9%) were resistant to de novo methylation, regardless of DNMT1 overexpression. In contrast, we identified a subset of methylation-prone CpG islands (3.8%) that were consistently hypermethylated in multiple DNMT1 overexpressing clones. Methylation-prone and methylation-resistant CpG islands were not significantly different with respect to size, C+G content, CpG frequency, chromosomal location, or gene association or pro-

moter association. To discriminate methylation-prone from methylation-resistant CpG islands, we developed a novel DNA pattern recognition model and algorithm [61], and coupled our predictive model described herein with the patterns found. We were able to derive a classification function based on the frequency of seven novel sequence patterns that was capable of discriminating methylation-prone from methylation-resistant CpG islands with 90% correctness upon cross-validation, and 85% accuracy when tested against blind CpG islands unknown to us regarding the methylation status. The data indicate that CpG islands differ in their intrinsic susceptibility to de novo methylation, and suggest that the propensity for a CpG island to become aberrantly methylated can be predicted on the basis of its sequence context.

The significance of this research is twofold. First, the identification of sequence patterns/attributes that distinguish methylation-prone CpG islands will lead to a better understanding of the basic mechanisms underlying aberrant CpG island methylation. Because genes that are silenced by methylation are otherwise structurally sound, the potential for reactivating these genes by blocking or reversing the methylation process represents an exciting new molecular target for chemotherapeutic intervention. A better understanding of the factors that contribute to aberrant methylation, including the identification of sequence elements that may act to target aberrant methylation, will be an important step in achieving this long-term goal. Secondly, the classification of the more than 29,000 known (but as yet unclassified) CpG islands in human chromosomes will provide an important resource for the identification of novel gene targets for further study as potential molecular markers that could impact on both cancer prevention and treatment. Extensive RLGS fingerprint information (and thus potential training sets of methylated CpG islands) already exists for a number of human tumor types, including breast, brain, lung, leukemias, hepatocellular carcinomas, and primitive neuroectodermal tumor [23,24,35,102]. Thus, the methods and tools developed are directly applicable to CpG island methylation data derived from human tumors. Moreover, new microarray-based techniques capable of "profiling" more than 7000 CpG islands have been developed and applied to human breast cancers [15,117,118]. We are uniquely poised to take advantage of the tumor CpG island methylation profile information that will likely be generated using these techniques over the next several years. Thus, our general-predictive modeling framework has the potential to lead to improved diagnosis and prognosis and treatment planning for cancer patients.

Discriminant Analysis of Cell Motility and Morphology Data in Human Lung Carcinoma Refer to [16] for more details of this work. This study focuses on the differential effects of extracellular matrix proteins on the motility and morphology of human lung epidermoid carcinoma cells. The behavior of carcinoma cells is contrasted with that of normal L-132 cells, resulting in a method for the prediction of metastatic potential. Data collected from time-lapsed videomicroscopy were used to simultaneously produce quantitative measures of motility and morphology. The data were subsequently analyzed using our discriminant analysis model and algorithm to discover relationships between motility, morphology, and substratum. Our discriminant analysis tools enabled the consideration of many more cell attributes than is customary in cell motility studies. The observations correlate with behaviors seen in vivo and suggest specific roles for the extracellular matrix proteins and their integrin receptors in metastasis. Cell translocation in vitro has been associated with malignancy, as has an elongated phenotype [120] and a rounded phenotype [97]. Our study suggests that extracellular matrix proteins contribute in different ways to the malignancy of cancer cells, and that multiple malignant phenotypes exist.

Ultrasound-Assisted Cell Disruption for Drug Delivery Reference [57] discusses this in detail. Although biological effects of ultrasound must be avoided for safe diagnostic applications, ultrasound's ability to disrupt cell membranes has attracted interest as a method to facilitate drug and gene delivery. This preliminary study seeks to develop rules for predicting the degree of cell membrane disruption based on specified ultrasound parameters and measured acoustic signals. Too much ultrasound destroys cells, while cell membranes will not open up for absorption of macromolecules when too little ultrasound is applied. The key is to increase cell permeability to allow absorption of macromolecules, and to apply ultrasound transiently to disrupt viable cells so

as to enable exogenous material to enter without cell damage. Thus our task is to uncover a "predictive rule" of ultrasound-mediated disruption of red blood cells using acoustic spectrums and measurements of cell permeability recorded in experiments.

Our predictive model and solver for generating prediction rules were applied to data obtained from a sequence of experiments on bovine red blood cells. For each experiment, the attributes consisted of four ultrasound parameters, acoustic measurements at 400 frequencies, and a measure of cell membrane disruption. To avoid overtraining, various feature combinations of the 404 predictor variables were selected when developing the classification rule. The results indicate that the variable combination consisting of ultrasound exposure time and acoustic signals measured at the driving frequency and its higher harmonics yields the best rule, and our method compares favorably with classification tree and other ad hoc approaches, with a correct classification rate of 80% upon cross-validation and 85% when classifying new unknown entities. Our methods used for deriving the prediction rules are broadly applicable, and could be used to develop prediction rules in other scenarios involving different cell types or tissues. These rules and the methods used to derive them could be used for real-time feedback about ultrasound's biological effects. For example, it could assist clinicians during a drug delivery process, or could be imported into an implantable device inside the body for automatic drug delivery and monitoring.

Identification of Tumor Shape and Volume in Treatment of Sarcoma Reference [56] includes the detailed analysis. This project involves the determination of tumor shape for adjuvant brachytherapy treatment of sarcoma, based on catheter images taken after surgery. In this application, the entities are overlapping consecutive triplets of catheter markings, each of which is used for determining the shape of the tumor contour. The triplets are to be classified into one of two groups: group 1 (triplets for which the middle catheter marking should be bypassed) and group 2 (triplets for which the middle marking should not be bypassed). To develop and validate a classification rule, we used clinical data collected from 15 soft-tissue sarcoma patients. Cumulatively, this comprised 620 triplets of catheter markings. By careful (and tedious) clinical analysis of the geometry of these triplets, 65 were determined to belong to group 1, the "bypass" group, and 555 were determined to belong to group 2, the "do-not-bypass" group.

A set of measurements associated with each triplet was then determined. The choice of what attributes to measure to best distinguish triplets as belonging to group 1 or group 2 is nontrivial. The attributes involved the distance between each pair of markings, angles, and the curvature formed by the three triplet markings. On the basis of the attributes selected, our predictive model was used to develop a classification rule. The resulting rule provides 98% correct classification on cross-validation, and was capable of correctly determining/predicting 95% of the shape of the tumor with new patients' data. We remark that the current clinical procedure requires manual outline based on markers in films of the tumor volume. This study was the first to use automatic construction of tumor shape for sarcoma adjuvant brachytherapy [56,62].

Discriminant Analysis of Biomarkers for Prediction of Early Atherosclerosis More detail on this work can be found in [65]. Oxidative stress is an important etiologic factor in the pathogenesis of vascular disease. Oxidative stress results from an imbalance between injurious oxidant and protective antioxidant events, of which the former predominate [88,103]. This results in the modification of proteins and DNA, alteration in gene expression, promotion of inflammation, and deterioration in endothelial function in the vessel wall, all processes that ultimately trigger or exacerbate the atherosclerotic process [22,111]. It was hypothesized that novel biomarkers of oxidative stress would predict early atherosclerosis in a relatively healthy nonsmoking population free from cardiovascular disease. One hundred and twenty-seven healthy nonsmokers, without known clinical atherosclerosis had carotid intima media thickness (IMT) measured using ultrasound. Plasma oxidative stress was estimated by measuring plasma lipid hydroperoxides using the determination of reactive oxygen metabolites (d-ROMs) test. Clinical measurements include traditional risk factors, including age, sex, low-density lipoprotein (LDL), high-density lipoprotein (HDL), triglycerides, cholesterol, body-mass index (BMI), hypertension, diabetes mellitus, smoking history, family history of coronary artery

disease, Framingham risk score, and high-sensitivity C-reactive protein.

For this prediction, the patients were first clustered into two groups: (group 1, IMT $\geq$ 0.68; group 2, IMT < 0.68). On the basis of this separator, 30 patients belonged to group 1, and 97 belonged to group 2. Through each iteration, the classification method trains and learns from the input training set and returns the most discriminatory patterns among the 14 clinical measurements; ultimately resulting in the development of a prediction rule based on observed values of these discriminatory patterns among the patient data. Using all 127 patients as a training set, the predictive model identified age, sex, BMI, HDL cholesterol, family history of coronary artery disease under 60, high-sensitivity C-reactive protein, and d-ROM as discriminatory attributes that together provide unbiased correct classification of 90 and 93%, respectively, for group 1 (IMT $\geq$ 0.68) and group 2 (IMT < 0.68) patients. To further test the power of the classification method for correctly predicting the IMT status of new/unseen patients, we randomly selected a smaller patient training set of size 90. The predictive rule from this training set yielded 80 and 89% correct rates for predicting the remaining 37 patients as group 1 and group 2 patients, respectively. The importance of d-ROM as a discriminatory predictor for IMT status was confirmed during the machine learning process. This biomarker was selected in every iteration as the "machine" learned and was trained to develop a predictive rule to correctly classify patients in the training set. We also performed predictive analysis using Framingham risk score and d-ROM; in this case the unbiased correct classification rates (for the 127 individuals) for groups 1 and 2 were 77 and 84%, respectively. This is the first study to illustrate that this measure of oxidative stress can be effectively used along with traditional risk factors to generate a predictive rule that can potentially serve as an inexpensive clinical diagnostic tool for prediction of early atherosclerosis.

Fingerprinting Native and Angiogenic Microvascular Networks Through Pattern Recognition and Discriminant Analysis of Functional Perfusion Data

The analysis and findings are described in [64]. The cardiovascular system provides oxygen and nutrients to the entire body. Pathological conditions that impair normal microvascular perfusion can result in tissue ischemia, with potentially serious clinical effects. Conversely, development of new vascular structures fuels the progression of cancer, macular degeneration, and atherosclerosis. Fluorescence microangiography offers superb imaging of the functional perfusion of new and existent microvasculature, but quantitative analysis of the complex capillary patterns is challenging. We developed an automated pattern-recognition algorithm to systematically analyze the microvascular networks, and then applied our classification model described herein to generate a predictive rule. The pattern-recognition algorithm identifies the complex vascular branching patterns, and the predictive rule demonstrates, respectively, 100 and 91% correct classification for perturbed (diseased) and normal tissue perfusion. We confirmed that transplantation of normal bone marrow to mice in which genetic deficiency resulted in impaired angiogenesis eliminated predicted differences and restored normal-tissue perfusion patterns (with 100% correctness). The pattern-recognition and classification method offers an elegant solution for the automated fingerprinting of microvascular networks that could contribute to better understanding of angiogenic mechanisms and be utilized to diagnose and monitor microvascular deficiencies. Such information would be valuable for early detection and monitoring of functional abnormalities before they produce obvious and lasting effects, which may include improper perfusion of tissue, or support of tumor development.

The algorithm can be used to discriminate between the angiogenic response in a native healthy specimen compared with groups with impairment due to age or chemical or other genetic deficiency. Similarly, it can be applied to analyze angiogenic responses as a result of various treatments. This will serve two important goals. First, the identification of discriminatory patterns/attributes that distinguish angiogenesis status will lead to a better understanding of the basic mechanisms underlying this process. Because therapeutic control of angiogenesis could influence physiological and pathological processes such as wound and tissue repairing, cancer progression and metastasis, or macular degeneration, the ability to understand it under different conditions will offer new insight into developing novel therapeutic interventions, monitoring and treatment, especially in aging, and heart disease. Thus, our study

and the results form the foundation of a valuable diagnostic tool for changes in the functionality of the microvasculature and for discovery of drugs that alter the angiogenic response. The methods can be applied to tumor diagnosis, monitoring, and prognosis. In particular, it will be possible to derive microangiographic fingerprints to acquire specific microvascular patterns associated with early stages of tumor development. Such "angioprinting" could become an extremely helpful early diagnostic modality, especially for easily accessible tumors such as skin cancer.

Prediction of Protein Localization Sites The protein localization database consists of eight groups with a total of 336 instances (143, 77, 52, 35, 20, 5, 2, and 2, respectively) with seven attributes [91]. The eight groups are eight localization sites of protein, including cytoplasm (cp), inner membrane without signal sequence (im), perisplasm (pp), inner membrane, uncleavable signal sequence (imU), outer membrane (om), outer membrane lipoprotein (omL), inner membrane lipoprotein (imL), inner membrane, and cleavable signal sequence (imS). However, the last four groups were taken out of our classification experiment since the population sizes are too small to ensure significance.

The seven attributes include McGeoch's method for signal sequence recognition (mcg), von Heijne's method for signal sequence recognition (gvh), von Heijne's signal peptidase II consensus sequence score (lip), presence of charge on N-terminus of predicted lipoproteins (chg), score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins (aac), score of the ALOM membrane spanning region prediction program (alm1), and score of the ALOM program after excluding putative cleavable signal regions from the sequence (alm2).

In the classification we use four groups, 307 instances, with seven attributes. Our classification model selected the discriminatory patterns mcg, gvh, alm1, and alm2 to form the predictive rule with unbiased correct classification rates of 89%, compared with 81% by other classification models [48].

Pattern Recognition in Satellite Images for Determining Types of Soil The satellite database consists of the multispectral values of pixels in 3×3 neighborhoods in a satellite image, and the classification associated with the central pixel in each neighborhood. The aim is to predict this classification, given the multispectral values. In the sample database, the class of a pixel is coded as a number. There are six groups with 4435 samples in the training data set and 2000 samples in the testing data set; and each sample entity has 36 attributes describing the spectral bands of the image [91].

The original Landsat Multi-Spectral Scanner (MSS) image data for this database were generated from data purchased from NASA by the Australian Centre for Remote Sensing. The Landsat satellite data are one of the many sources of information available for a scene. The interpretation of a scene by integrating spatial data of diverse types and resolutions including multispectral and radar data, maps indicating topography, land use, etc. is expected to assume significant importance with the onset of an era characterized by integrative approaches to remote sensing (for example, NASA's Earth Observing System commencing this decade).

One frame of Landsat MSS imagery consists of four digital images of the same scene in different spectral bands. Two of these are in the visible region (corresponding approximately to green and red regions of the visible spectrum) and two are in the (near) infrared. Each pixel is an 8-bit binary word, with 0 corresponding to black and 255 to white. The spatial resolution of a pixel is about 80 m $\times$ 80 m. Each image contains 2340×3380 such pixels.

The database is a (tiny) subarea of a scene, consisting of 82×100 pixels. Each line of data corresponds to a 3×3 square neighborhood of pixels completely contained within the 82×100 subarea. Each line contains the pixel values in the four spectral bands (converted to ASCII) of each of the nine pixels in the 3×3 neighborhood and a number indicating the classification label of the central pixel. The number is a code for the following six groups: red soil, cotton crop, gray soil, damp gray soil, soil with vegetation stubble, and very damp gray soil. Running our classification model, we selected 17 discriminatory attributes to form the classification rule, producing an unbiased prediction with 85% accuracy.

Further Advances

Brooks and Lee [17,18] devised other variations of the basic DAMIP model. They also showed that

DAMIP is strongly universally consistent (in some sense) with very good rates of convergence from Vapnik and Chervonenkis theory. A polynomial-time algorithm for discriminating between two populations with the DAMIP model was developed, and DAMIP was shown to be $\mathcal{NP}$-complete for a general number of groups. The proof demonstrating $\mathcal{NP}$-completeness employs results used in generating edges of the conflict graph [4,11,12,55]. Exploiting the necessary and sufficient conditions that identify edges in the conflict graph is the central contribution to the improvement in solution performance over industry-standard software. The conflict graph is the basis for various valid inequalities, a branching scheme, and for conditions under which integer variables are fixed for all solutions. Additional solution methods are identified which include a heuristic for finding solutions at nodes in the branch-and-bound tree, upper bounds for model parameters, and necessary conditions for edges in the conflict hypergraph [26,58]. Further, we have concluded that DAMIP is a computationally feasible, consistent, stable, robust, and accurate classifier.

Progress and Challenges

We summarize in Table 2 the mathematical programming techniques used in classification problems as reviewed in this chapter.

As noted by current research efforts, multigroup classification remains $\mathcal{NP}$-complete and much work is needed to design effective models as well as to derive novel and efficient computational algorithms to solve these multigroup instances.

Other Methods

While most classification methods can be described in terms of discriminant functions, some methods are not trained in the paradigm of determining coefficients or parameters for functions of a predefined form. These methods include *classification and regression trees*, *nearest-neighbor* methods, and *neural networks*.

Classification and regression trees [14] are nonparametric approaches to prediction. Classification trees seek to develop classification rules based on successive binary partitions of observations based on attribute values. Regression trees also employ rules consisting of binary partitions, but are used to predict continuous responses.

The rules generated by classification trees are easily viewable by plotting them in a treelike structure, from which the name arises. A test entity may be classified using rules in a tree plot by first comparing the entity's data with the root node of the tree. If the root node condition is satisfied by the data for a particular entity, the left branch is followed to another node; otherwise, the right branch is followed to another node. The data from the observation are compared with conditions at subsequent nodes until a leaf node is reached.

Nearest-neighbor methods begin by establishing a set of labeled prototype observations. The nearest-neighbor classification rule assigns test entities to groups according to the group membership of the nearest prototype. Different measures of distance may be used. The k-nearest-neighbor rule assigns entities to groups according to the group membership of the k nearest prototypes.

Neural networks are classification models that can also be interpreted in terms of discriminant functions, though they are used in a way that does not require finding an analytic form for the functions [25]. Neural networks are trained by considering one observation at a time, modifying the classification procedure slightly with each iteration.

Summary and Conclusion

In this chapter, we presented an overview of mathematical programming based classification models, and analyzed their development and advances in recent years. Many mathematical programming methods are geared toward two-group analysis only, and their performance is often compared with Fisher's LDF or Smith's QDF. It has been noted that these methods can be used for multiple group analysis by finding $G(G-1)/2$ discriminants for each pair of groups ("one against one") or by finding G discriminants for each group versus the remaining data ("one against all"), but these approaches can lead to ambiguous classification rules [25].

Mathematical programming methods developed for multiple group analysis have been described [10,32,39,40,41,46,59,60,63,93]. Multiple group formulations for SVMs have been proposed and tested [17,36,40,49,59,60,66], but are still considered computationally in-

Disease Diagnosis: Optimization-Based Methods, Table 2
Progress in mathematical programming-based classification models

Mathematical programming methods	References
Linear programming	
Two-group classification	
Separate data by hyperplanes	[74,75]
Minimizing the sum of deviations, minimizing the maximum deviation, and minimizing the sum of interior distances	[5,31,32,33,47,99]
Hybrid model	[45,99]
Review	[27,50,107]
Software	[110]
Issues about normalization	[34,44,51,52,53,87,100,114,115,116]
Robust linear programming	[9,86]
Inclusion of second-order terms	[104,113]
Effect of the position of outliers	[94]
Binary attributes	[3]
Multigroup classification	
Single function classification	[32]
Multiple function classification	[10,46]
Classification with reserved-judgment region using linear programming	[39,40,60,63]
Mixed integer programming	
Two-group classification	
Minimizing the number of misclassifications	[1,5,6,7,54,101,105,109,119]
Review	[27,50,107]
Software	[110]
Secondary goals	[96]
Binary attributes	[3]
Normalization and attribute selection	[42]
Dichotomous categorical variable formation	[43]
Multigroup classification	
Multigroup classification	[41,93]
Three-group classification	[71,72,95]
Classification with reserved-judgment region using mixed integer programming	[17,39,40,59,60]
Nonlinear programming	
Two-group classification	
L_p-norm criterion	[108]
Review	[27,50,107]
Piecewise-linear nonconvex discriminant function	[85]
Minimizing the number of misclassifications	[21,76,77]
Minimizing the sum of arbitrary-norm distances	[78]
Support vector machine	
Introduction and tutorial	[20,112]
Generalized support vector machine	[79,83]
Methods for huge-size problems	[13,36,37,38,67,68,80,81,82,84]
Multigroup support vector machine	[17,38,39,40,49,59,60,63,66]

tensive [49]. The "one-against-one" and "one-against-all" methods with SVMs have been successfully applied [49,90].

We also discussed a class of multigroup general-purpose predictive models that we have developed based on the technology of large-scale optimization and SVMs [17,19,39,40,59,60,63]. Our models seek to maximize the correct classification rate while constraining the number of misclassifications in each group. The models incorporate the following features: (1) the ability to classify any number of distinct groups; (2) allow incorporation of heterogeneous types of attributes as input; (3) a high-dimensional data transformation that eliminates noise and errors in biological data; (4) constrain the misclassification in each group and a reserved-judgment region that provides a safeguard against overtraining (which tends to lead to high misclassification rates from the resulting predictive rule); and (5) successive multistage classification capability to handle data points placed in the reserved-judgment region. The performance and predictive power of the classification models is validated through a broad class of biological and medical applications.

Classification models are critical to medical advances as they can be used in genomic, cell, molecular, and system-level analyses to assist in early prediction, diagnosis and detection of disease, as well as for intervention and monitoring. As shown in the CpG

island study for human cancer, such prediction and diagnosis opens up novel therapeutic sites for early intervention. The ultrasound application illustrates its application to a novel drug delivery mechanism, assisting clinicians during a drug delivery process, or in devising devices that can be implanted into the body for automated drug delivery and monitoring. The lung cancer cell motility study offers an understanding of how cancer cells behave in different protein media, thus assisting in the identification of potential gene therapy and target treatment. Prediction of the shape of a cancer tumor bed provides a personalized treatment design, replacing manual estimates by sophisticated computer predictive models. Prediction of early atherosclerosis through inexpensive biomarker measurements and traditional risk factors can serve as a potential clinical diagnostic tool for routine physical and health maintenance, alerting physicians and patients to the need for early intervention to prevent serious vascular disease. Fingerprinting of microvascular networks opens up the possibility for early diagnosis of perturbed systems in the body that may trigger disease (e. g., genetic deficiency, diabetes, aging, obesity, macular degeneracy, tumor formation), identification of target sites for treatment, and monitoring prognosis and success of treatment. Determining the type of erythemato-squamous disease and the presence/absence of heart disease helps clinicians to correctly diagnose and effectively treat patients. Thus, classification models serve as a basis for predictive medicine where the desire is to diagnose early and provide personalized target intervention. This has the potential to reduce healthcare costs, improve success of treatment, and improve quality of life of patients.

References

1. Abad PL, Banks WJ (1993) New LP based heuristics for the classification problem. Eur J Oper Res 67:88–100
2. Anderson JA (1969) Constrained discrimination between *k* populations. J Roy Statist Soc Ser B (Methodological) 31(1):123–139
3. Asparoukhov OK, Stam A (1997) Mathematical programming formulations for two-group classification with binary variables. Ann Oper Res 74:89–112
4. Atamturk A (1998) Conflict graphs and flow models for mixed-integer linear optimization problems. PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta
5. Bajgier SM, Hill AV (1982) An experimental comparison of statistical and linear programming approaches to the discriminant problem. Decis Sci 13:604–618
6. Banks WJ, Abad PL (1991) An efficient optimal solution algorithm for the classification problem. Decis Sci 22:1008–1023
7. Banks WJ, Abad PL (1994) On the performance of linear programming heuristics applied on a quadratic transformation in the classification problem. Eur J Oper Res 74:23–28
8. Bennett KP (1992) Decision tree construction via linear programming. In: Evans M (ed) Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference, pp 97–101
9. Bennett KP, Mangasarian OL (1992) Robust linear programming discrimination of two linearly inseparable sets. Optim Methods Softw 1:23–34
10. Bennett KP, Mangasarian OL (1994) Multicategory discrimination via linear programming. Optim Methods Softw 3:27–39
11. Bixby RE, Lee EK (1998) Solving a truck dispatching scheduling problem using branch-and-cut. Oper Res 46:355–367
12. Borndörfer R (1997) Aspects of set packing, partitioning and covering. PhD thesis, Technischen Universität Berlin, Berlin
13. Bradley PS, Mangasarian OL (2000) Massive data discrimination via linear support vector machines. Optim Methods Softw 13(1):1–10
14. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and Regression Trees. Wadsworth and Brooks/Cole Advanced Books and Software, Pacific Grove
15. Brock GJ, Huang TH, Chen CM, Johnson KJ (2001) A novel technique for the identification of CpG islands exhibiting altered methylation patterns (ICEAMP). Nucleic Acids Res 29:e123
16. Brooks JP, Wright A, Zhu C, Lee EK (2007) Discriminant analysis of motility and morphology data from human lung carcinoma cells placed on purified extracellular matrix proteins. Ann Biomed Eng, Submitted
17. Brooks JP, Lee EK (2006) Solving a mixed-integer programming formulation of a multi-category constrained discrimination model. In: Proceedings of the 2006 INFORMS Workshop on Artificial Intelligence and Data Mining, Pittsburgh
18. Brooks JP, Lee EK (2007) Analysis of the consistency of a mixed integer programming-based multi-category constrained discriminant model. Submitted
19. Brooks JP, Lee EK (2007) Mixed integer programming constrained discrimination model for credit screening. In: Proceedings of the 2007 Spring Simulation Multiconference, Business and Industry Symposium, pp 1–6, Norfolk, VA, March ACM Digital Library

20. Burges CJC (1998) A tutorial on support vector machines for pattern recognition. Data Mining Knowl Discov 2: 121–167
21. Chen C, Mangasarian OL (1996) Hybrid misclassification minimization. Adv Comput Math 5:127–136
22. Chevion M, Berenshtein E, Stadtman ER (2000) Human studies related to protein oxidation: protein carbonyl content as a marker of damage. Free Radical Res 33(Suppl):S99–S108
23. Costello JF, Fruhwald MC, Smiraglia DJ, Rush LJ, Robertson GP, Gao X, Wright FA, Feramisco JD, Peltomaki P, Lang JC, Schuller DE, Yu L, Bloomfield CD, Caligiuri MA, Yates A, Nishikawa R, Su HH, Petrelli NJ, Zhang X, O'Dorisio MS, Held WA, Cavenee WK, Plass C (2000) Aberrant CpG-island methylation has non-random and tumour-type-specific patterns. Nat Genet 24:132–138
24. Costello JF, Plass C, Cavenee WK (2000) Aberrant methylation of genes in low-grade astrocytomas. Brain Tumor Pathol 17:49–56
25. Duda RO, Hart PE, Stork DG (2001) Pattern Classification. Wiley, New York
26. Easton T, Hooker K, Lee EK (2003) Facets of the independent set plytope. Math Program Ser B 98:177–199
27. Erenguc SS, Koehler GJ (1990) Survey of mathematical programming models and experimental results for linear discriminant analysis. Managerial Decis Econ 11:215–225
28. Feltus FA, Lee EK, Costello JF, Plass C, Vertino PM (2003) Predicting aberrant CpG island methylation. Proc Natl Acad Sci USA 100:12253–12258
29. Feltus FA, Lee EK, Costello JF, Plass C, Vertino PM (2006) DNA signatures associated with CpG island methylation states. Genomics 87:572–579
30. Fisher RA (1936) The use of multiple measurements in taxonomic problems. Ann Eugenics 7:179–188
31. Freed N, Glover F (1981) A linear programming approach to the discriminant problem. Decis Sci 12:68–74
32. Freed N, Glover F (1981) Simple but powerful goal programming models for discriminant problems. Eur J Oper Res 7:44–60
33. Freed N, Glover F (1986) Evaluating alternative linear programming models to solve the two-group discriminant problem. Decis Sci 17:151–162
34. Freed N, Glover F (1986) Resolving certain difficulties and improving the classification power of LP discriminant analysis formulations. Decis Sci 17:589–595
35. Fruhwald MC, O'Dorisio MS, Rush LJ, Reiter JL, Smiraglia DJ, Wenger G, Costello JF, White PS, Krahe R, Brodeur GM, Plass C (2000) Gene amplification in NETs/medulloblastomas: mapping of a novel amplified gene within the MYCN amplicon. J Med Genet 37:501–509
36. Fung GM, Mangasarian OL (2001) Proximal support vector machine classifiers. In Proceedings KDD-2001, San Francisco
37. Fung GM, Mangasarian OL (2002) Incremental support vector machine classification. In: Grossman R, Mannila H, Motwani R (eds) Proceedings of the Second SIAM International Conference on Data Mining. SIAM, Philadelphia, pp 247–260
38. Fung GM, Mangasarian OL (2005) Multicategory proximal support vector machine classifiers. Mach Learn 59:77–97
39. Gallagher RJ, Lee EK, Patterson DA (1996) An optimization model for constrained discriminant analysis and numerical experiments with iris, thyroid, and heart disease datasets. In: Proceedings of the 1996 American Medical Informatics Association
40. Gallagher RJ, Lee EK, Patterson DA (1997) Constrained discriminant analysis via 0/1 mixed integer programming. Ann Oper Res 74:65–88
41. Gehrlein WV (1986) General mathematical programming formulations for the statistical classification problem. Oper Res Lett 5(6):299–304
42. Glen JJ (1999) Integer programming methods for normalisation and variable selection in mathematical programming discriminant analysis models. J Oper Res Soc 50:1043–1053
43. Glen JJ (2004) Dichotomous categorical variable formation in mathematical programming discriminant analysis models. Naval Res Logist 51:575–596
44. Glover F (1990) Improved linear programming models for discriminant analysis. Decis Sci 21:771–785
45. Glover F, Keene S, Duea B (1988) A new class of models for the discriminant problem. Decis Sci 19:269–280
46. Gochet W, Stam A, Srinivasan V, Chen S (1997) Multigroup discriminant analysis using linear programming. Oper Res 45(2):213–225
47. Hand DJ (1981) Discrimination and classification. Wiley, New York
48. Horton P, Nakai K (1996) A probablistic classification system for predicting the cellular localization sites of proteins. In: Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology, St. Louis, USA, pp 109–115
49. Hsu CW, Lin CJ (2002) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Networks 13(2):415–425
50. Joachimsthaler EA, Stam A (1990) Mathematical programming approaches for the classification problem in two-group discriminant analysis. Multivariate Behavior Res 25(4):427–454
51. Koehler GJ (1989) Characterization of unacceptable solutions in LP discriminant analysis. Decis Sci 20:239–257
52. Koehler GJ (1989) Unacceptable solutions and the hybrid discriminant model. Decis Sci 20:844–848
53. Koehler GJ (1994) A response to Xiao's "necessary and sufficient conditions of unacceptable solutions in LP discriminant analysls": Something is amiss. Decis Sci 25: 331–333
54. Koehler GJ, Erenguc SS (1990) Minimizing misclassifications in linear discriminant analysis. Decis Sci 21: 63–85

55. Lee EK (1993) Solving a truck dispatching scheduling problem using branch-and-cut. PhD thesis, Computational and Applied Mathematics, Rice University, Houston
56. Lee EK, Fung AYC, Brooks JP, Zaider M (2002) Automated planning volume definition in soft-tissue sarcoma adjuvant brachytherapy. Biol Phys Med 47:1891–1910
57. Lee EK, Gallagher RJ, Campbell AM, Prausnitz MR (2004) Prediction of ultrasound-mediated disruption of cell membranes using machine learning techniques and statistial analysis of acoustic spectra. IEEE Trans Biomed Eng 51:1–9
58. Lee EK, Maheshwary S (2006) Conflict hypergraphs in integer programming. Technical report, Georgia Institute of Technology, submitted
59. Lee EK (2007) Optimization-based predictive models in medicine and biology. Optimization in Medicine. Springer Netherlands. Springer Series in Optimization and Its Application 12:127–151
60. Lee EK (2007) Large-scale optimization-based classification models in medicine and biology. Ann Biomed Eng Syst Biol Bioinformat 35(6):1095–1109
61. Lee EK, Easton T, Gupta K (2006) Novel evolutionary models and applications to sequence alignment problems. Ann Oper Res Oper Res Medic – Comput Optim Medic Life Sci 148:167–187
62. Lee EK, Fung AYC, Zaider M (2001) Automated planning volume contouring in soft-tissue sarcoma adjuvant brachytherapy treatment. Int J Radiat Oncol Biol Phys 51:391
63. Lee EK, Gallagher RJ, Patterson DA (2003) A linear programming approach to discriminant analysis with a reserved-judgment region. INFORMS J Comput 15(1):23–41
64. Lee EK, Jagannathan S, Johnson C, Galis ZS (2006) Fingerprinting native and angiogenic microvascular networks through pattern recognition and discriminant analysis of functional perfusion data. submitted
65. Lee EK, Wu TL, Ashfaq S, Jones DP, Rhodes SD, Weintrau WS, Hopper CH, Vaccarino V, Harrison DG, Quyyumi AA (2007) Prediction of early atherosclerosis in healthy adults via novel markers of oxidative stress and d-ROMs. Working paper
66. Lee Y, Lin Y, Wahba G (2004) Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. J Am Stat Assoc 99:67–81
67. Lee YJ, Mangasarian OL (2001) RSVM: Reduced support vector machines. In: Proceedings of the SIAM International Conference on Data Mining, Chicago, April 5–7
68. Lee YJ, Mangasarian OL (2001) SSVM: A smooth support vector machine for classification. Comput Optim Appl 20(1):5–22
69. Lee YJ, Mangasarian OL, Wolberg WH (2000) Breast cancer survival and chemotherapy: A support vector machine analysis. In: DIMACS Series in Discrete Mathematical and Theoretical Computer Science, vol 55. American Mathematical Society, Providence, pp 1–10
70. Lee YJ, Mangasarian OL, Wolberg WH (2003) Survival-time classification of breast cancer patients. Comput Optim Appl 25:151–166
71. Loucopoulos C, Pavur R (1997) Computational characteristics of a new mathematical programming model for the three-group discriminant problem. Comput Oper Res 24(2):179–191
72. Loucopoulos C, Pavur R (1997) Experimental evaluation of the classificatory performance of mathematical programming approaches to the three-group discriminant problem: The case of small samples. Ann Oper Res 74:191–209
73. Luedi PP, Hartemink AJ, Jirtle RL (2005) Genome-wide prediction of imprinted murine genes. Genome Res 15:875–884
74. Mangasarian OL (1965) Linear and nonlinear separation of patterns by linear programming. Oper Res 13:444–452
75. Mangasarian OL (1968) Multi-surface method of pattern separation. IEEE Trans Inform Theory 14(6):801–807
76. Mangasarian OL (1994) Misclassification minimization. J Global Optim 5:309–323
77. Mangasarian OL (1996) Machine learning via polyhedral concave minimization. In: Fischer H, Riedmueller B, Schaeffler S (eds) Applied Mathematics and Parallel computing – Festschrift for Klaus Ritter. Physica-Verlag, Heidelberg, pp 175–188
78. Mangasarian OL (1999) Arbitrary-norm separating plane. Oper Res Lett 24:15–23
79. Mangasarian OL (2000) Generalized support vector machines. In: Smola AJ, Bartlett P, Schökopf B, Schuurmans D (eds) Advances in Large Margin Classifiers. MIT Press, Cambridge, pp 135–146
80. Mangasarian OL (2003) Data mining via support vector machines. In: Sachs EW, Tichatschke R (eds) System Modeling and Optimization XX. Kluwer, Boston, pp 91–112
81. Mangasarian OL (2005) Support vector machine classification via parameterless robust linear programming. Optim Methods Softw 20:115–125
82. Mangasarian OL, Musicant DR (1999) Successive overrelaxation for support vector machines. IEEE Trans Neural Networks 10:1032–1037
83. Mangasarian OL, Musicant DR (2001) Data discrimination via nonlinear generalized support vector machines. In: Ferris MC, Mangasarian OL, Pang JS (eds) Complementarity: Applications, Algorithms and Extensions. Kluwer, Boston, pp 233–251
84. Mangasarian OL, Musicant DR (2001) Lagrangian support vector machines. J Mach Learn Res 1:161–177
85. Mangasarian OL, Setiono R, Wolberg WH (1990) Pattern recognition via linear programming: Theory and application to medical diagnosis. In: Coleman TF, Li Y (eds) Large-Scale Numerical Optimization. SIAM, Philadelphia, pp 22–31

86. Mangasarian OL, Street WN, Wolberg WH (1995) Breast cancer diagnosis and prognosis via linear programming. Oper Res 43(4):570–577
87. Markowski EP, Markowski CA (1985) Some difficulties and improvements in applying linear programming formulations to the discriminant problem. Decis Sci 16:237–247
88. McCord JM (2000) The evolution of free radicals and oxidative stress. Am J Med 108:652–659
89. McLachlan GJ (1992) Discriminant analysis and statistical pattern recognition. Wiley, New York
90. Müller KR, Mika S, Rätsch G, Tsuda K, Schölkopf B (2001) An introduction to kernel-based learning algorithms. IEEE Trans Neural Networks 12(2):181–201
91. Murphy PM, Aha DW (1994) UCI Repository of machine learning databases http://www.ics.uci.edu/~mlearn/MLRepository.html. Department of Information and Computer Science, University of California, Irvine
92. O'Hagan A (1994) Kendall's Advanced Theory of Statistics: Bayesian Inference, vol 2B. Halsted Press, New York
93. Pavur R (1997) Dimensionality representation of linear discriminant function space for the multiple-group problem: An MIP approach. Ann Oper Res 74:37–50
94. Pavur R (2002) A comparative study of the effect of the position of outliers on classical and nontraditional approaches to the two-group classification problem. Eur J Oper Res 136:603–615
95. Pavur R, Loucopoulos C (2001) Evaluating the effect of gap size in a single function mathematical programming model for the three-group classification problem. J Oper Res Soc 52:896–904
96. Pavur R, Wanarat P, Loucopoulos C (1997) Examination of the classificatory performance of MIP models with secondary goals for the two-group discriminant problem. Ann Oper Res 74:173–189
97. Raz A, Ben-Zéev A (1987) Cell contact and architecture of malignant cells and their relationship to metastasis. Cancer Metastasis Rev 6:3–21
98. Rencher AC (1998) Multivariate Statistical Inference and Application. Wiley, New York
99. Rubin PA (1990) A comparison of linear programming and parametric approaches to the two-group discriminant problem. Decis Sci 21:373–386
100. Rubin PA (1991) Separation failure in linear programming discriminant models. Decis Sci 22:519–535
101. Rubin PA (1997) Solving mixed integer classification problems by decomposition. Ann Oper Res 74:51–64
102. Rush LJ, Dai Z, Smiraglia DJ, Gao X, Wright FA, Fruhwald M, Costello JF, Held WA, Yu L, Krahe R, Kolitz JE, Bloomfield CD, Caligiuri MA, Plass C (2001) Novel methylation targets in de novo acute myeloid leukemia with prevalence of chromosome 11 loci. Blood 97:3226–3233
103. Sies H (1985) Oxidative stress: introductory comments. In: Sies H (ed) Oxidative Stress. Academic Press, London, pp 1–8
104. Duarte Silva AP, Stam A (1994) Second order mathematical programming formulations for discriminant analysis. Eur J Oper Res 72:4–22
105. Duarte Silva AP, Stam A (1997) A mixed integer programming algorithm for minimizing the training sample misclassification cost in two-group classification. Ann Oper Res 74:129–157
106. Smith CAB (1947) Some examples of discrimination. Ann Eugenics 13:272–282
107. Stam A (1997) Nontraditional approaches to statistical classification: Some perspectives on l_p-norm methods. Ann Oper Res 74:1–36
108. Stam A, Joachimsthaler EA (1989) Solving the classification problem in discriminant analysis via linear and nonlinear programming methods. Decis Sci 20:285–293
109. Stam A, Joachimsthaler EA (1990) A comparison of a robust mixed-integer approach to existing methods for establishing classification rules for the discriminant problem. Eur J Oper Res 46:113–122
110. Stam A, Ungar DR (1995) RAGNU: A microcomputer package for two-group mathematical programming-based nonparametric classification. Eur J Oper Res 86:374–388
111. Tahara S, Matsuo M, Kaneko T (2001) Age-related changes in oxidative damage to lipids and DNA in rat skin. Mechan Ageing Develop 122:415–426
112. Vapnik V (1995) The Nature of Statistical Learning Theory. Springer, New York
113. Wanarat P, Pavur R (1996) Examining the effect of second-order terms in mathematical programming approaches to the classification problem. Eur J Oper Res 93: 582–601
114. Xiao B (1993) Necessary and sufficient conditions of unacceptable solutions in LP discriminant analysis. Decis Sci 24:699–712
115. Xiao B (1994) Decision power and solutions of LP discriminant models: Rejoinder. Decis Sci 25:335–336
116. Xiao B, Feng Y (1997) Alternative discriminant vectors in LP models and a regularization method. Ann Oper Res 74:113–127
117. Yan PS, Chen CM, Shi H, Rahmatpanah F, Wei SH, Caldwell CW, Huang TH (2001) Dissecting complex epigenetic alterations in breast cancer using CpG island microarrays. Cancer Res 61:8375–8380
118. Yan PS, Perry MR, Laux DE, Asare AL, Caldwell CW, Huang TH (2000) CpG island arrays: an application toward deciphering epigenetic signatures of breast cancer. Clin Cancer Res 6:1432–1438
119. Yanev N, Balev S (1999) A combinatorial approach to the classification problem. Eur J Oper Res 115:339–350
120. Zimmermann A, Keller HU (1987) Locomotion of tumor cells as an element of invasion and metastasis. Biomed Pharmacotherapy 41:337–344
121. Zopounidis C, Doumpos M (2002) Multicriteria classification and sorting methods: A literature review. Eur J Oper Res 138:229–246

Disjunctive Programming

DP

HANIF D. SHERALI
Virginia Polytechnic Institute and State University, Blacksburg, USA

MSC2000: 90C09, 90C10, 90C11

Article Outline

Keywords
See also
References

Keywords

Disjunctive programming; Polyhedral annexation; Facial disjunctive program; Cutting planes; Nondominated cuts; Facet; Valid inequalities; Reformulation-linearization technique; Lift-and-project; Tight relaxations; Model reformulation; Convex hull; Mixed integer 0–1 programs; Polynomial programs; Nonconvex programs

Disjunctive programming (DP) problems can be stated in the form

$$\text{(DP)} \qquad \text{Minimize } \{f(x)\colon\ x \in X, x \in \cup_{h\in H} S_h\}\,,$$

where $f\colon \mathbf{R}^n \to \mathbf{R}$ is a lower semicontinuous function, X is a closed convex subset of the nonnegative orthant of $\mathbf{R}^n$, and H is an index set for the collection of nonempty polyhedra

$$S_h = \left\{x\colon\ A^h x \geq b^h, x \geq 0\right\}, \quad h \in H. \tag{1}$$

The name for this class of problems arises from the feature that the constraints in (1) include the *disjunction* that at least one of the (linear) sets of constraints defining S_h, for $h \in H$, must be satisfied. Problems including other logical conditions such as conjunctions, negations, and implications can be cast in the framework of this problem. Problem (DP) subsumes the classes of 0–1 *mixed integer problems*, the generalized lattice point problem, the cardinality constrained linear program, the extreme point optimization problem, the linear complementarity problem, among numerous others, and finds application in several related problems such as orthogonal production scheduling, scheduling on identical machines, multistage assignment, location-allocation problems, load balancing problems, the segregated storage problem, the fixed-charge problem, project/portfolio selection problems, goal programming problems, and many other game theory and decision theory problems (see [35] for a detailed discussion of such problems and applications).

The theory and algorithms for disjunctive programming problems are mainly supported by the fundamental *disjunctive cut principle*. The forward part of this result due to E. Balas [4,5] states that for any nonnegative surrogate multiplier vectors λ^h, $h \in H$, the inequality

$$\sup_{h\in H}\{\lambda^h A^h\}x \geq \inf_{h\in H}\{\lambda^h b^h\} \tag{2}$$

is valid for (or is implied by) the disjunction $x \in \cup_{h\in H} S_h$, where the $\sup\{\cdot\}$ and $\inf\{\cdot\}$ are taken componentwise in (2). More importantly, the converse part of this result due to R.G. Jeroslow [16] states that for any given valid inequality $\pi x \geq \pi_0$ for the disjunction $x \in \cup_{h\in H} S_h$, there exist nonnegative surrogate multipliers λ^h, $h \in H$, such that the disjunctive cut (2) implies this given valid inequality, or uniformly dominates it, over the nonnegative orthant. This disjunctive cut principle also arises from the setting of convexity cuts and polyhedral annexation methods as propounded by F. Glover [11,12], and it subsumes as well as can improve upon many types of classical cutting planes such as Gomory's mixed integer cuts, intersection cuts, and reverse outer polar cuts for 0–1 programs (see [4,5,11,12,35]). H.P. Williams [39] provides some additional insights into disjunctive formulations.

The generation of particular types of 'deep cuts' to delete a given solution (say, the origin, without loss of generality) based on the criteria of maximizing the Euclidean distance or the rectilinear distance between the origin and the nonnegative region feasible to the cutting plane, or maximizing the surplus in the cut with respect to the origin subject to suitable normalization constraints has also been explored in [34,37]. The intent behind such cutting plane methods is to generate *nondominated valid inequalities* that are supports (and hopefully, facets) of the closure convex hull of solutions

feasible to the disjunction. H.D. Sherali and C.M. Shetty [35,37] discuss how different alternate formulations of the disjunctive statement can influence the strength of the cut derived therefrom, and demonstrate how a sequence of augmented formulations can be used to sequentially tighten a given valid inequality. This process turns out to be precisely the Glover polyhedral annexation scheme in [12]. In contrast with this sequence dependent *'lifting' procedure*, Sherali and Shetty [37] propose a 'simultaneous lifting' variant of this approach. Other types of disjunctive cutting planes for special problems include the cuts of [4,5,10,11,12,20], and [32] for linear knapsack, multiple choice and combinatorial disjunctions, [29] for linear complementarity problems, and the facet cuts of [25] based on the convex hull of certain types of disjunctions.

Balas [3] also provides an algebraic characterization for the closure *convex hull* of a union of polyhedra. This characterization is particularly useful in the study of the important class of facial disjunctive programs, that subsumes mixed integer 0–1 problems and linear complementarity problems, for example. A *facial disjunctive program* (FDP) can be stated as follows.

$$\text{(FDP)} \qquad \text{Minimize}\,\{cx\colon\ x \in X \cap Y\}\,,$$

where X is a nonempty polytope in $\mathbf{R}^n$, and where Y is a conjunction of some $\widehat{h}$ disjunctions given in the so-called *conjunctive normal form* (conjunction of disjunctions)

$$Y = \cap_{h\in H}\left[\cup_{i\in Q_h}\left\{x\colon\ a_i^h x \geq b_i^h\right\}\right]. \tag{3}$$

Here, $H = \{1,\ldots,\widehat{h}\}$ and for each $h \in H$ we have specified a disjunction that requires at least one of the inequalities $a_i^h x \geq b_i^h$, for $i \in Q_h$, to be satisfied. The terminology 'facial' conveys the feature that $X \cap \{x\colon a_i^h x \geq b_i^h\}$ defines a face of X for each $i \in Q_h$, $h \in H$. For example, in the context of 0–1 mixed integer problems, the set X represents the linear programming relaxation of the problem, and for each binary variable x_h, $h \in H$, the corresponding disjunction in (3) states that $x_h \leq 0$ or $x_h \geq 1$ should hold true (where $0 \leq x_h \leq 1$ is included within X). Balas [3] shows that for facial disjunctive programs, the convex hull of feasible solutions can be constructed inductively by starting with $K_0 = X$ and then determining

$$K_h = \text{conv}\left[\cup_{i\in Q_h}\left(K_{h-1}\cap\left\{x\colon\ a_i^h x \geq b_i^h\right\}\right)\right]$$
$$\text{for}\quad h = 1,\ldots,\widehat{h}\,, \tag{4}$$

where $K_{\widehat{h}}$ produces $\text{conv}(X \cap Y)$. Based on this, a hierarchy of relaxations $K_0,\ldots,K_{\widehat{h}}$ is generated for (FDP) that spans the spectrum from the linear programming to the convex hull representation [6]. Each member in this hierarchy can also be viewed as being obtained by representing the feasible region of the original problem as the intersection of the union of certain polyhedra, and then taking a hull-relaxation of this representation. Here, for a set $D = \cap_j D_j$, where each D_j is the union of certain polyhedra, the hull-relaxation of D [3] is defined as $h - \text{rel}(D) = \cap_j \text{conv}(D_j) \supseteq \text{conv}(D)$.

In the context of 0–1 mixed integer problems (MIP), Sherali and W.P. Adams [27,28] develop a *reformulation-linearization technique* (RLT) for generating a hierarchy of such relaxations, introducing the notion of multiplying constraints using factors composed of x_h and $(1 - x_h)$, $h \in H$, to reformulate the problem, followed by a variable substitution to linearize the resulting problem. Approaches based on such constraint product and linearization strategies were used by these authors earlier in the context of several special applications [1,2,26]. Later, L. Lovász and A. Schrijver [17] independently used more general constraint factors to generate a similar hierarchy for 0–1 problems. The foregoing RLT construct can be specialized to derive K_h defined by (4) for 0–1 MIPs, where in this case,

$$K_h \equiv \text{conv}\left[(K_{h-1}\cap\{x\colon\ x_h \leq 0\})\right.$$
$$\left.\cup\,(K_{h-1}\cap\{x\colon\ x_h \geq 1\})\right]$$

can be obtained by multiplying the (implicitly defined) constraints of K_{h-1} by x_h and $(1 - x_h)$ and then linearizing the resulting problem. This RLT approach is used in [8] in the *'lift-and-project' hierarchy* of relaxations. However, the RLT process of [27,28] generates tighter relaxations at each level which can be viewed as hull relaxations produced by the intersection of the convex hull of the union of certain specially constructed polyhedra. No direct realization of (4) can produce these relaxations. For a survey on RLT approaches and for further enhancements, see [29,30].

In the context of general facial disjunctive programs, Jeroslow [15] presented a cutting plane algorithm that generates suitable facetial inequalities at each stage of the procedure such that an overall finite convergence is guaranteed via (4). This is accomplished by showing that in the worst case, the hierarchy $K_0, \ldots, K_{\widehat{h}}$ would be generated. The lift-and-project algorithm of [8] employs this cutting plane procedure based on the foregoing hierarchy of relaxations. Balas [7] also addresses an enhanced procedure that considers two variables at a time to define the disjunctions. The RLT process is used to construct partial convex hulls, and the resulting relaxations are embedded in a branch and cut algorithm.

Furthermore, for general facial disjunctive programs, Sherali and Shetty [36] present another finitely convergent cutting plane algorithm. At each step, this procedure searches for *extreme faces* of X relative to the cuts generated thus far (these are faces that do not contain any feasible points lying in a lower-dimensional face of X, see [18]), and based on the dimension of this extreme face and its feasibility to Y, either a disjunctive face cut or a disjunctive intersection cut is generated. This procedure was specialized for *bilinear programming* problems in [33] to derive a first nonenumerative finitely convergent algorithm for this class of problems.

Other disjunctive cutting plane algorithms include the Sherali–Sen procedures [31] for solving the general class of *extreme point mathematical programs*, the Baptiste–LePape procedures [9], and the Pinto–Grossmann procedures [21] for solving certain scheduling problems having disjunctive logic constraints. S. Sen and Sherali [24] also discuss issues related to designing convergent cutting plane algorithms, and present examples to show nonconvergence of certain iterative disjunctive cutting plane methods. Sensitivity and stability issues related to feasible and optimal sets of disjunctive programs have been addressed in [14]; [13] deals with the problem of solving algebraic systems of disjunctive equations. For other applications of disjunctive methods to process systems engineering, and to logic programming, see [19,23,38].

See also

- ► MINLP: Branch and Bound Global Optimization Algorithm
- ► MINLP: Branch and Bound Methods
- ► MINLP: Global Optimization with αBB
- ► MINLP: Logic-Based Methods
- ► Reformulation-linearization Methods for Global Optimization

References

1. Adams WP, Sherali HD (1986) A tight linearization and an algorithm for zero-one quadratic programming problems. Managem Sci 32(10):1274–1290
2. Adams WP, Sherali HD (1990) Linearization strategies for a class of zero-one mixed integer programming problems. Oper Res 38(2):217–226
3. Balas E (1974) Disjunctive programming: Properties of the convex hull of feasible points. Managem Sci Res Report GSIA Carnegie-Mellon Univ 348, no. July
4. Balas E (1974) Intersection cuts from disjunctive constraints. Managem Sci Res Report Carnegie-Mellon Univ 330, no. Feb
5. Balas E (1975) Disjunctive programming: Cutting planes from logical conditions. In: Mangasarian OL, Meyer RR, Robinson SM (eds) Nonlinear Programming. Acad. Press, New York
6. Balas E (1985) Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM J Alg Discrete Meth 6:466–485
7. Balas E (1997) A modified lift-and-project procedure. Math Program 79(1–3):19–32
8. Balas E, Ceria S, Cornuejols G (1993) A lift-and-project cutting plane algorithm for mixed 0-1 programs. Math Program 58:295–324
9. Baptiste P, Lepape C (1996) Disjunctive constraints for manufacturing scheduling: Principles and extensions. Internat J Comput Integrated Manufacturing 9(4):306–310
10. Glover F (1973) Convexity cuts and cut search. Oper Res 21:123–134
11. Glover F (1974) Polyhedral convexity cuts and negative edge extensions. Z Oper Res 18:181–186
12. Glover F (1975) Polyhedral annexation in mixed integer and combinatorial programming. Math Program 8:161–188 (See also MSRS Report 73-9, Univ. Colorado, August, 1973).
13. Grossmann IE, Turkay M (1996) Solution of algebraic systems of disjunctive equations. Comput Chem Eng 20, Suppl.:S339–S344
14. Helbig S (1994) Stability in disjunctive optimization II. Continuity of the feasible and optimal set. Optim 31(1): 63–93
15. Jeroslow RG (1977) A cutting plane game and its algorithms. Discussion Paper Center Oper Res and Econometrics Univ Catholique de Louvain 7724, no. June
16. Jeroslow RG (1977) Cutting plane theory: Disjunctive methods. Ann Discret Math 1:293–330

17. Lovasz L, Schrijver A (1991) Cones of matrices and set functions and 0-1 optimization. SIAM J Optim 1:166–190
18. Majthay A, Whinston A (1974) Quasi-concave minimization subject to linear constraints. Discret Math 9:35–59
19. Mcaloon K, Tretkoff C (1997) Logic, modeling, and programming. Ann Oper Res 71:335–372
20. Owen G (1973) Cutting planes for programs with disjunctive constraints. Optim Theory Appl 11:49–55
21. Pinto JM, Grossmann IE (1997) A logic based approach to scheduling problems with resource constraints. Comput Chem Eng 21(8):801–818
22. Ramarao B, Shetty CM (1984) Application of disjunctive programming to the linear complementarity problem. Naval Res Logist Quart 31:589–600
23. Sakama C, Seki H (1997) Partial deduction in disjunctive logic programming. J Logic Programming 32(3):229–245
24. Sen S, Sherali HD (1985) On the convergence of cutting plane algorithms for a class of nonconvex mathematical programs. Math Program 31(1):42–56
25. Sen S, Sherali HD (1986) Facet inequalities from simple disjunctions in cutting plane theory. Math Program 34(1): 72–83
26. Sherali HD, Adams WP (1984) A decomposition algorithm for a discrete location-allocation problem. Oper Res 32(878–900
27. Sherali HD, Adams WP (1990) A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J Discret Math 3(3):411–430
28. Sherali H, Adams WP (1994) A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. Discrete Appl Math 52:83–106 Manuscript, Virginia Polytechnic Inst. State Univ., 1989.
29. Sherali HD, Adams WP (1996) Computational advances using the reformulation-linearization technique (RLT) to solve discrete and continuous nonconvex problems. OPTIMA 49:1–6
30. Sherali HD, Adams WP, Driscoll P (1998) Exploiting special structures in constructing a hierarchy of relaxations for 0-1 mixed integer problems. Oper Res 46(3):396–405
31. Sherali HD, Sen S (1985) A disjunctive cutting plane algorithm for the extreme point mathematical programming problem. Opsearch (Theory) 22(2):83–94
32. Sherali HD, Sen S (1985) On generating cutting planes from combinatorial disjunctions. Oper Res 33(4):928–933
33. Sherali HD, Shetty CM (1980) A finitely convergent algorithm for bilinear programming problems using polar cuts and disjunctive face cuts. Math Program 19:14–31
34. Sherali HD, Shetty CM (1980) On the generation of deep disjunctive cutting planes. Naval Res Logist Quart 27(3):453–475
35. Sherali HD, Shetty CM (1980) Optimization with disjunctive constraints. Lecture Notes Economics and Math Systems, vol 181. Springer, Berlin
36. Sherali HD, Shetty CM (1982) A finitely convergent procedure for facial disjunctive programs. Discrete Appl Math 4:135–148
37. Sherali HD, Shetty CM (1983) Nondominated cuts for disjunctive programs and polyhedral annexation methods. Opsearch (Theory) 20(3):129–144
38. Vecchietti A, Grossmann IE (1997) LOGMIP: A disjunctive 0-1 nonlinear optimizer for process systems models. Comput Chem Eng 21, Suppl.:S427–S432
39. Williams HP (1994) An alternative explanation of disjunctive formulations. Europ J Oper Res 72(1):200–203

Distance Dependent Protein Force Field via Linear Optimization

R. Rajgaria, S. R. McAllister, Christodoulos A. Floudas
Department of Chemical Engineering, Princeton University, Princeton, USA

Article Outline

Abstract
Keywords
Introduction
Theory and Modeling
Physical Constraints
Database Selection and Decoy Generation
Training and Test Set
Results and Discussion
Conclusions
References

Abstract

Protein force fields play an important role in protein structure prediction. Knowledge based force fields use the database information to derive the interaction energy between different residues or atoms of a protein. These simplified force fields require less computational effort and are relatively easy to use. A C^{α}–C^{α} distance dependent high resolution force field has been developed using a set of high quality (low rmsd) decoys. A linear programming based formulation was used in which non-native "decoy" conformers are forced to take a higher energy compared to the corresponding native structure. This force field was tested on an independent test set and was found to excel on all the metrics that are widely used to measure the effectiveness of a force field.

Keywords

Force field; Potential model; High resolution decoys; Protein structure prediction; Linear optimization; Protein design potential

Introduction

Predicting the structure of a protein from its amino acid sequence is one of the biggest and yet most fundamental problems in computational structural biology. Anfinsen's hypothesis [1] is one of the main approaches used to solve this problem, which says that for a given physiological set of conditions the native structure of a protein corresponds to the global Gibbs free energy minimum. Thus, one needs a force field to calculate the energy of different conformers and pick the one with the lowest energy.

Physics-based force fields consider various types of interactions (for example, van der Waals interactions, hydrogen bonding, electrostatic interactions etc.) occurring at the atomic level of a protein to calculate the energy of a conformer. CHARMM [19], AMBER [5], ECEPP [20], ECEPP/3 [21] and GROMOS [24] are a few examples of the physics-based force fields. On the other hand, knowledge-based force fields use information from databases. Researchers have used the Boltzmann distribution [4,7,26,], optimization based techniques [17,27] and many other approaches [6,12,13, 14,15,16,18,23,25] to calculate these parameters. A recent review on such potentials can be found in Floudas et al. [8].

This work presents a novel C^{α}–C^{α} distance dependent high resolution force field that has been generated using linear optimization based framework [22]. The emphasis is on the high resolution, which would enable us to differentiate between native and non-native structures that are very similar to each other (rmsd < 2 Å). The force field is called high resolution because it has been trained on a large set of high resolution decoys (small rmsd with respect to the native) and it intends to effectively distinguish high resolution decoys structures from the native structure.

The basic framework used in this work is similar to the one developed by Loose et al. [17]. However, it has been improved and applied to a diverse and enhanced (both in terms of quantity and quantity) set of high resolution decoys. The new proposed model has resulted in remarkable improvements over the LKF potential. These high resolution decoys were generated using torsion angle dynamics in combination with restricted variations of the hydrophobic core within the native structure. This decoy set highly improves the quality of training and testing. The force field developed in this paper was tested by comparing the energy of the native fold to the energies of decoy structures for proteins separate from those used to train the model. Other leading force fields were also tested on this high quality decoy set and the results were compared with the results of our high resolution potential. The comparison is presented in the Results section.

Theory and Modeling

In this model, amino acids are represented by the location of its C^{α} atom on the amino acid backbone. The conformation of a protein is represented by a coordinate vector, X, which includes the location of the C^{α} of each amino acid. The native conformation is denoted as X_n, while the set $i = 1, \ldots, N$ is used to denote the decoy conformations X_i. Non-native decoys are generated for each of $p = 1, \ldots, P$ proteins and the energy of the native fold for each protein is forced to be lower than those of the decoy conformations (Anfinsen's hypothesis). This constraint is shown in the following equation:

$$E(X_{p,i}) - E(X_{p,n}) > \varepsilon \quad p = 1, \ldots, P \quad i = 1, \ldots, N \tag{1}$$

Equation (1) requires the native conformer to be always lower in energy than its decoy. A small positive parameter ε is used to avoid the trivial solution in which all energies are set to zero. An additional constraint (Eq. 2) is used to produce a nontrivial solution by constraining the sum of the differences in energies between decoy and native folds to be greater than a positive constant [28]. For the model presented in this paper, the values of ε and Γ were set to 0.01 and 1000, respectively.

$$\sum_{p} \sum_{i} [E(X_{p,i}) - E(X_{p,n})] > \Gamma \tag{2}$$

The energy of each conformation is taken as the arithmetic sum of pairwise interactions corresponding to each amino acid combination at a particular "contact" distance. A contact exists when the C^{α} carbons of two

Distance Dependent Protein Force Field via Linear Optimization, Table 1
Distance dependent bin definition [17]

Bin ID	C^{α} Distance [Å]
1	3–4
2	4–5
3	5–5.5
4	5.5–6
5	6–6.5
6	6.5–7
7	7–8
8	8–9

amino acids are within 9 Å of each other. The energy of each interaction is a function of the C^{α}–C^{α} distances and the identity of the interacting amino acids. To formulate the model, the energy of an interaction between a pair of amino acids, IC, within a distance bin, ID, was defined as $\theta^{\mathrm{IC,ID}}$. The eight distance bins defined for the formulation are shown in Table 1. The energy for any fold X, of decoy i, for a protein p, is given by Eq. (3).

$$E(X_{p,i}) = \sum_{\mathrm{IC}} \sum_{\mathrm{ID}} N_{p,i,\mathrm{IC,ID}} \theta_{\mathrm{IC,ID}} \tag{3}$$

In this equation, $N_{p,i,\mathrm{IC,ID}}$ is the number of interactions between an amino acid pair IC, at a C^{α}–C^{α} distance ID. The set IC ranges from 1 to 210 to account for the 210 unique combinations of the 20 naturally occurring amino acids. These bin definitions yield a total of 1680 interaction parameters to be determined by this model. To determine these parameters, a linear programming formulation is used in which the energy of a native protein is compared with a large number of its decoys. The violations, in which a non-native fold has a lower energy than the natural conformation, are minimized by optimizing with respect to these interaction parameters.

Equation (1) can be rewritten in terms of $N_{p,i,\mathrm{IC,ID}}$ as Eq. (4), where the slack parameters, S_p, are positive variables (Eq. 5) that represent the difference between the energies of the decoys and the native conformation of a given protein.

$$\sum_{\mathrm{IC}} \sum_{\mathrm{ID}} [N_{p,i,\mathrm{IC,ID}} - N_{p,n,\mathrm{IC,ID}}]\theta_{\mathrm{IC,ID}} + S_p \geq \varepsilon$$
$$p = 1, \ldots, P \quad i = 1, \ldots, N \tag{4}$$

$$S_p \geq 0 \quad p = 1, \ldots, P \tag{5}$$

$$\min_{\theta(\mathrm{IC,ID})} \sum_{p} S_p \tag{6}$$

The objective function for this formulation is to minimize the sum of the slack variables, S_p, written in the form of Eq. (6). The relative magnitude of $\theta_{\mathrm{IC,ID}}$ is meaningless because if all $\theta_{\mathrm{IC,ID}}$ parameters are multiplied by a common factor then Eqs. (4) and (5) are still valid. In this formulation, $\theta_{\mathrm{IC,ID}}$ values were bound between −25 and 25.

Physical Constraints

The above mentioned equations constitute the basic constraints needed to solve this model. However, this set does not guarantee a physically realistic solution. It is possible to come up with a set of parameters that can satisfy Eqs. (2,3,4,5,6) but would not reflect the actual interaction occurring between amino acids in a real system. To prohibit these unrealistic cases, another set of constraints based on the physical properties of the amino acids was imposed. Statistical results presented in Bahar and Jernigan [2] were also incorporated through the introduction of hydrophilic and hydrophobic constraints. The details of these physical constraints are given elsewhere [22].

Database Selection and Decoy Generation

The protein database selection is critical to force field training. This set should adequately represent the PDB set [3]. At the same time, it should not be too large, as the training becomes difficult with an increase in the size of the training set. Zhang and Skolinck [29] developed a set of 1,489 nonhomologous single domain proteins. High resolution decoys were generated for these proteins and used for training and testing purposes. High quality decoy generation was based on the hypothesis that high-quality decoy structures should preserve information about the distances within the hydrophobic core of the native structure of each protein. For each of the proteins in the database, a number of distance constraints are introduced based on the hydrophobic-hydrophobic distances within the native structure. Using a set of proximity parameters, a large number of decoy structures are generated using DYANA [9]. The rmsd distribution of decoy structures can be found elsewhere [22].

Training and Test Set

Of the 1400 proteins used for decoy generation, 1250 were randomly selected for training and the rest were used for testing purposes. For every protein in the set, 500–1600 decoys were generated depending on the fraction of secondary structure present in the native structure of the protein. These decoys were sorted based on their C^{α} rmsd to the native structure and then 500 decoys were randomly selected to represent the whole rmsd range. This creates a training set of $500 \times 1250 = 625,000$ decoys. However, because of computer memory limitations, it is not possible to include all of these decoys at the same time for training. An iterative scheme, "Rank and Drop", was employed to overcome the memory problem while effectively using all the high quality structures. In this scheme, a subset of decoys is used to generate a force field. This force field is then used to rank all the decoys and a set of most challenging decoys (based on their energy value) is selected for the next round of force field generation. This process of force field generation and decoy ranking is repeated until there is no improvement in the ranking of the decoys [22]. This force field model was solved using the GAMS modeling language coupled with the CPLEX linear programming package [11].

It is equally important to test a force field on a difficult and rigorous testing set to confirm its effectiveness. The test set was comprised of 150 randomly selected proteins (41–200 amino acids in length). For each of the 150 test proteins, 500 high resolution decoys were generated using the same technique that was used to generate training decoys. The minimum C^{α} based rmsds for these non-native structures were in the range of 0–2 Å. This HR force field was also tested on another set of medium resolution decoys [17]. This set has 200 decoys for 151 proteins. The minimum RMSD of the decoys of this set ranged from 3–16 Å. This set, along with the high resolution decoy set, spans the practical range of possible protein structures that one might encounter during protein structure prediction.

Results and Discussion

A linear optimization problem was solved using information from 625,000 decoy structures and the values of all the energy parameters were obtained. The ability to distinguish between the native structure and native-like conformers is the most significant test for any force field. The HR force field was tested on 500 decoys of the 150 test proteins. In this testing, the relative position, or rank, of the native conformation among its decoys was calculated. An ideal force field should be able to assign rank 1 to the native structures of all the test proteins. Other force fields like LKF [17], TE13 [27], and HL [10] were also tested on this set of high resolution decoys. All these force fields are fundamentally different from each other in their methods of energy estimation. Comparing the results obtained with these force fields aims to assess the fundamental utility of the HR force field. The comparison of the energy rankings obtained using different force fields is presented in Table 2. From this table it is evident that the HR force field is the most effective in identifying the native structures by rank. The HR force field correctly identified the native folds of 113 proteins out of a set of 150 proteins, which compares favorably to a maximum of 92 (out of 148) by the TE13 force field.

Distance Dependent Protein Force Field via Linear Optimization, Table 2
Testing force fields on 150 proteins of the high resolution decoy set. TE13 force field was only tested on 148 cases

FF-Name	Average Rank	No of Firsts	Average rmsd
HR	1.87	113 (75.33%)	0.451
LKF	39.45	17 (11.33%)	1.721
TE13	19.94	92 (62.16%)	0.813
HL	44.93	70 (46.67%)	1.092

Another analysis was carried out to evaluate the discrimination ability of these potentials. In this evaluation, all the decoys of the test set were ranked using these potentials. For each test protein, the C^{α} rmsd of the rank 1 conformer was calculated with respect to the native structure of that protein. The C^{α} rmsd would be zero for the cases in which a force field selects the native structure as rank 1. However, it will not be zero for all other cases in which a non-native conformer is assigned the top rank. The average of these rmsds represents the spatial separation of the decoys with respect to the native structure. The average rmsd value obtained for each of the force fields is shown in Table 2. It can be seen that the average C^{α} rmsd value is least for the HR force field. The average C^{α} rmsd value for the HR force field is 0.451 Å, which is much less compared to 1.721 Å by the LKF, and 0.813 Å by TE13 force field. This means

that the structures predicted by the HR force fields have the least spatial deviation from their corresponding native structures.

The HR force field was also tested on the test set published by Loose et al. [17] and was found to do better than other force fields. The comparison results for this test can be found elsewhere [22]. The effectiveness of the HR force field is further reinforced by its success on the medium resolution decoy test set. On the test set of 110 medium resolution decoys, it was capable of correctly identifying 78.2 % of the native structures, significantly more than other force fields.

Conclusions

The HR force field was developed using an optimization based linear programming formulation, in which the model is trained using a diverse set of high quality decoys. Physically observed interactions between certain amino acids were written in the form of mathematical constraints and included in the formulation.

The decoys were generated based on the premise that high quality decoy structures should preserve information about the distance within the hydrophobic core of the native structure of each protein. The set of interaction energy parameters obtained after solving the model were found to be of very good discriminatory capacity. This force field performed well on a set of independent, non-homologous high resolution decoys. This force field can become a powerful tool for fold recognition and de novo protein design.

References

1. Anfinsen CB (1973) Principles that govern the folding of protein chains. Science 181:223–230
2. Bahar I, Jernigan RL (1997) Inter-residue potential in globular proteins and the dominance of highly specific hydrophillic interactions at close separation. J Molec Biol 266:195–214
3. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) The protein data bank. Nucleic Acids Res 28:235–242
4. Bryant SH, Lawrence CE (1991) The frequency of ion-pair substructures in proteins is quantitaively related to electrostatic potential, a statistical model for nonbonded interactions. Proteins: Structure, Function, Bioinformatics 9:108–119
5. Cornell WD, Cieplak P, Bayly CI, Gould IR, Merz Jr KM, Ferguson DM, Spellmeyer DC, Fox T, Caldwell JW, Kollman PA (1995) A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. J Am Chem Soc 117:5179–5197
6. DeBolt S, Skolnick J (1996) Evaluation of atomic level mean force potentials via inverse folding and inverse refinement of proteins structures: atomic burial position and pairwise non-bonded interactions. Protein Eng 9:637–655
7. Finkelstein AV, Badretdinov AY, Gutin AM (1995) Why do proteins architectures have Boltzmann-like statistics? Proteins: Structure, Function, Bioinformatics 23:142–150
8. Floudas CA, Fung HK, McAllister SR, Mönnigmann M, Rajgaria R (2006) Advances in protein structure prediction and de novo protein design: A review. Chem Eng Sci 61:966–988
9. Güntert P, Mumenthaler C, Wüthrich K (1997) Torsion angle dynamics for NMR structure calculation with the new program DYANA. J Molec Biol 273:283–298
10. Hinds DA, Levitt M (1994) Exploring conformational space with a simple lattice model for protein structure. J Molec Biol 243:668–682
11. ILOG CPLEX (2003) User's Manual 9.0
12. Jernigan RL, Bahar I (1996) Structure-derived potentials and protein simulations. Curr Opin Struct Biol 6:195–209
13. Liwo A, Kazmierkiewicz R, Czaplewski C, Groth M, Oldziej S, Wawak RJ, Rackovsky S, Pincus MR, Scheraga HA (1998) A united-residue force field for off-lattice protein structure simulations. III. Origin of backbone hydrogen bonding cooperativity in united residue potential. J Comput Chem 19:259–276
14. Liwo A, Odziej S, Czaplewski C, Kozlowska U, Scheraga HA (2004) Parametrization of backbone-electrostatic and multibody contributions to the UNRES force field for protein-structure prediction from ab initio energy surfaces of model systems. J Phys Chem B 108:9421–9438
15. Liwo A, Oldziej S, Pincus MR, Wawak RJ, Rackovsky S, Scheraga HA (1997) A united-residue force field for off-lattice protein structure simulations. I. Functional forms and parameters of long-range side-chain interaction potentials from protein crystal data. J Comput Chem 18:849–873
16. Liwo A, Pincus MR, Wawak RJ, Rackovsky S, Oldziej S, Scheraga HA (1997) A united-residue force field for off-lattice protein structure simulations. II. Parameterization of short-range interactions and determination of weights of energy terms by z-score optimization. J Comput Chem 18:874–887
17. Loose C, Klepeis JL, Floudas CA (2004) A new pairwise folding potential based on improved decoy generation and side-chain packing. Proteins: Structure, Function, Bioinformatics 54:303–314
18. Lu H, Skolnick J (2001) A distance-dependent knowledge-based potential for improved protein structure selection. Proteins: Structure, Function, Bioinformatics 44:223–232
19. MacKerell Jr AD, Bashford D, Bellott M, Dunbrack Jr RL, Evanseck JD, Field MJ, Fischer S, Gao J, Guo H, Ha S, Joseph-McCarthy D, Kuchnir L, Kuczera K, Lau FTK, Mattos C, Michnick S, Ngo T, Nguyen DT, Prodhom B, Reiher III WE, Roux

B, Schlenkrich M, Smith JC, Stote R, Straub J, Watanabe M, Wiórkiewicz-Kuczera J, Yin D, Karplus M (1998) All-atom empirical potential for molecular modeling and dynamics studies of proteins. J Phys Chem B 102:3586–3616
20. Momany FA, McGuire RF, Burgess AW, Scheraga HA (1975) Energy parameters in polypeptides. VII. Geometric parameters, partial atomic charges, nonbonded interactions, hydrogen bond interactions, and intrinsic torsional potentials for the naturally occurring amino acids. J Phys Chem 79:2361–2381
21. Némethy G, Gibson KD, Palmer KA, Yoon CN, Paterlini G, Zagari A, Rumsey S, Scheraga HA (1992) Energy parameters in polypeptides. 10. Improved geometrical parameters and nonbonded interactions for use in the ECEPP/3 algorithm, with application to proline-containing peptides. J Phys Chem 96:6472–6484
22. Rajgaria R, McAllister SR, Floudas CA (2006) Development of a novel high resolution C^{α}–C^{α} distance dependent force field using a high quality decoy set. Proteins: Structure, Function, Bioinformatics 65:726–741
23. Samudrala R, Moult J (1998) An all-atom distance-dependent conditional probability discriminatory function for protein structure prediction. J Molec Biol 275:895–916
24. Scott WRP, Hunenberger PH, Trioni IG, Mark AE, Billeter SR, Fennen J, Torda AE, Huber T, Kruger P, VanGunsteren WF (1997) The GROMOS biomolecular simulation program package. J Phys Chem A 103:3596–3607
25. Subramaniam S, Tcheng DK, Fenton J (1996) A knowledge-based method for protein structure refinement and prediction. In: States D, Agarwal P, Gaasterland T, Hunter L, Smith R (eds) Proceedings of the 4th International Conference on Intelligent Systems in Molecular Biology. AAAI Press, Boston, pp 218–229
26. Tanaka S, Scheraga HA (1976) Medium- and long-range interaction parameters between amino acids for predicting three-dimensional structures of proteins. Macromolecules 9:945–950
27. Tobi D, Elber R (2000) Distance-dependent, pair potential for protein folding: Results from linear optimization. Proteins: Structure, Function, Bioinformatics 41:40–46
28. Tobi D, Shafran G, Linial N, Elber R (2000) On the design and analysis of protein folding potentials. Proteins: Structure, Function, Bioinformatics 40:71–85
29. Zhang Y, Skolnick J (2004) Automated structure prediction of weakly homologous proteins on a genomic scale. Proc Natl Acad Sci USA 101:7594–7599

Domination Analysis in Combinatorial Optimization

GREGORY GUTIN
Department of Computer Science, Royal Holloway, University of London, Egham, UK

MSC2000: 90C27, 90C59, 68Q25, 68W40, 68R10

Article Outline

Keywords and Phrases
Introduction
Definitions
Methods
  Greedy-Type Algorithms
  Better-Than-Average Heuristics
Cases
  Traveling Salesman Problem
  Upper Bounds for Domination Numbers of ATSP Heuristics
  Multidimensional Assignment Problem (MAP)
  Minimum Partition
    and Multiprocessor Scheduling Problems
  Max Cut Problem
  Constraint Satisfaction Problems
  Vertex Cover, Independent Set and Clique Problems
  Quadratic Assignment Problem
See also
References

Keywords and Phrases

Domination analysis; Domination number; Domination ratio

Introduction

Exact algorithms allow one to find optimal solutions to NP-hard combinatorial optimization (CO) problems. Many research papers report on solving large instances of some NP-hard problems (see, e. g., [25,27]). The running time of exact algorithms is often very high for large instances (many hours or even days), and very large instances remain beyond the capabilities of exact algorithms. Even for instances of moderate size, if we wish to remain within seconds or minutes rather than hours or days of running time, only heuristics can be used. Certainly, with heuristics, we are not guaranteed to find optimum, but good heuristics normally produce near-optimal solutions. This is enough in most applications since very often the data and/or mathematical model are not exact anyway.

Research on CO heuristics has produced a large variety of heuristics especially for well-known CO problems. Thus, we need to choose the best ones among them. In most of the literature, heuristics are compared in computational experiments. While experi-

mental analysis is of definite importance, it cannot cover all possible families of instances of the CO problem at hand and, in particular, it practically never covers the hardest instances.

Approximation Analysis [3] is a frequently used tool for theoretical evaluation of CO heuristics. Let $\mathcal{H}$ be a heuristic for a combinatorial minimization problem P and let $\mathcal{I}_n$ be the set of instances of P of size n. In approximation analysis, we use the performance ratio $r_H(n) = \max\{f(I)/f^*(I) : I \in \mathcal{I}_n\}$, where $f(I)(f^*(I))$ is the value of the heuristic (optimal) solution of I. Unfortunately, for many CO problems, estimates for $r_{\mathcal{H}}(n)$ are not constants and provide only a vague picture of the quality of heuristics. Moreover, even constant performance ratio does not guarantee that the heuristic often outputs good-quality solutions, see, e. g., the discussion of the DMST heuristic below.

Domination Analysis (DA) (for surveys, see [22,24]) provides an alternative and a complement to approximation analysis. In DA, we are interested in the domination number or domination ratio of heuristics. *Domination number* (*ratio*) of a heuristic H for a combinatorial optimization problem P is the maximum number (fraction) of all solutions that are not better than the solution found by H for any instance of P of size n. In many cases, DA is very useful. For example, we will see later that the greedy algorithm has domination number 1 for many CO problems. In other words, the greedy algorithm, in the worst case, produces the unique worst possible solution. This is in line with latest computational experiments with the greedy algorithm, see, e. g., [25], where the authors came to the conclusion that the greedy algorithm 'might be said to self-destruct' and that it should not be used even as 'a general-purpose starting tour generator'.

The *Asymmetric Traveling Salesman Problem (ATSP)* is the problem of computing a minimum weight tour (Hamilton cycle) passing through every vertex in a weighted complete digraph K_n^* on n vertices. The *Symmetric TSP (STSP)* is the same problem, but on a complete undirected graph. When a certain fact holds for both ATSP and STSP, we will simply speak of *TSP*. Sometimes, the maximizing version of TSP is of interest, we denote it by *Max TSP*.

APX is the class of CO problems that admit polynomial time approximation algorithms with a constant performance ratio [3]. It is well known that while Max TSP belongs to APX, TSP does not. This is at odds with the simple fact that a 'good' approximation algorithm for Max TSP can be easily transformed into an algorithm for TSP. Thus, it seems that both Max TSP and TSP should be in the same class of CO problems. The above asymmetry was already viewed as a drawback of performance ratio already in the 1970's, see, e. g. [11,28,33]. Notice that from the DA point view Max TSP and TSP are equivalent problems.

Zemel [33] was the first to characterize measures of quality of approximate solutions (of binary integer programming problems) that satisfy a few basic and natural properties: the measure becomes smaller for better solutions, it equals 0 for optimal solutions and it is the same for corresponding solutions of equivalent instances. While the performance ratio and even the relative error (see [3]) do not satisfy the last property, the parameter *1-r*, where *r* is the domination ratio, does satisfy all of the properties.

Local Search (LS) is one of the most successful approaches in constructing heuristics for CO problems. Recently, several researchers investigated LS with Very Large Scale Neighborhoods (see, e. g., [1,12,24]). The hypothesis behind this approach is that the larger the neighborhood the better quality solution are expected to be found [1]. However, some computational experiments do not support this hypothesis; sometimes an LS with small neighborhoods proves to be superior to that with large neighborhoods. This means that some other parameters are responsible for the relative power of neighborhoods. Theoretical and experimental results on TSP indicate that one such parameter may well be the domination number of the corresponding LS.

In our view, it is advantageous to have bounds for both performance ratio and domination number (or, domination ratio) of a heuristic whenever it is possible. Roughly speaking this will enable us to see a 2D rather than 1D picture. For example, consider the double minimum spanning tree heuristic (DMST) for the Metric STSP (i. e., STSP with triangle inequality). DMST starts from constructing a minimum weight spanning tree T in the complete graph of the STSP, doubles every edge in T, finds a closed Euler trail E in the 'double' T, and cancels any repetition of vertices in E to obtain a TSP tour H. It is well-known and easy to prove that the weight of H is at most twice the weight

of the optimal tour. Thus, the performance ratio for DMST is bounded by 2. However, Punnen, Margot and Kabadi [29] proved that the domination number of DMST is 1. Interestingly, in practice DMST often performs much worse than the well-known 2-Opt LS heuristic. For 2-Opt LS we cannot give any constant approximation guarantee, but the heuristic is of very large domination number [29].

The above example indicates that it makes sense to use DA to rank heuristics for the CO problem under consideration. If the domination number of a heuristic $\mathcal{H}$ is larger than the domination of a heuristic $\mathcal{H}'$ (for all or 'almost all' sizes n), we may say that $\mathcal{H}$ is better than $\mathcal{H}'$ in the worst case (from the DA point of view). Berend, Skiena and Twitto [10] used DA to rank some well-known heuristics for the Vertex Cover problem (and, thus, the Independent Set and Clique problems). The three problems and the heuristics will be defined in the corresponding subsection of the Cases section. Ben-Arieh et al. [7] studied three heuristics for the *Generalized TSP*: the vertices of the complete digraph are partitioned into subsets and the goal is to find a minimum weight cycle containing exactly one vertex from each subset. In the computational experiment in [7] one of the heuristics was clearly inferior to the other two. The best two behaved very similarly. Nevertheless, the authors of [7] managed to 'separate' the two heuristics by showing that one of the heuristics was of much larger domination number.

One might wonder whether a heuristic $\mathcal{A}$, which is significantly better that another heuristic $\mathcal{B}$ from the DA point of view, is better that $\mathcal{B}$ in computational experiments. In particular, whether the ATSP greedy algorithm, which is of domination number 1, is worse, in computational experiments, than any ATSP heuristic of domination number at least $(n-2)!$? Generally speaking the answer to this natural question is negative. This is because computational experiments and DA indicate different aspects of quality of heuristics. Nevertheless, it seems that many heuristics of very small domination number such as the ATSP greedy algorithm perform poorly also in computational experiments and, thus, cannot be recommended to be widely used in computational practice.

The rest of the entry is organized as follows. We give additional terminology and notation in the section Definitions. In the section Methods, we describe two powerful methods in DA. In the section Cases, we consider DA results for some well-known CO problems.

Definitions

Let $\mathcal{P}$ be a CO problem and let $\mathcal{H}$ be a heuristic for $\mathcal{P}$. The *domination number* $\mathrm{domn}(\mathcal{H}, \mathcal{I})$ of $\mathcal{H}$ *for an instance* $\mathcal{I}$ of $\mathcal{P}$ is the number of solutions of $\mathcal{I}$ that are not better than the solution s produced by $\mathcal{H}$ including s itself. For example, consider an instance $\mathcal{T}$ of the STSP on 5 vertices. Suppose that the weights of tours in $\mathcal{T}$ are 2,5,5,6,6,9,9,11,11,12,12,15 (every instance of STSP on 5 vertices has 12 tours) and suppose that the greedy algorithm computes the tour T of weight 6. Then $\mathrm{domn}(\texttt{greedy}, \mathcal{T}) = 9$. In general, if $\mathrm{domn}(\mathcal{H}, \mathcal{I})$ equals the number of solutions in $\mathcal{I}$, then $\mathcal{H}$ finds an optimal solution for $\mathcal{I}$. If $\mathrm{domn}(\mathcal{H}, \mathcal{I}) = 1$, then the solution found by $\mathcal{H}$ for $\mathcal{I}$ is the unique worst possible one.

The *domination number* $\mathrm{domn}(\mathcal{H}, n)$ of $\mathcal{H}$ is the minimum of $\mathrm{domn}(\mathcal{H}, \mathcal{I})$ over all instances $\mathcal{I}$ of size n. Since the ATSP on n vertices has $(n-1)!$ tours, an algorithm for the ATSP with domination number $(n-1)!$ is exact. The domination number of an exact algorithm for the STSP is $(n-1)!/2$. If an ATSP heuristic $\mathcal{A}$ has domination number equal 1, then there is an assignment of weights to the arcs of each complete digraph K_n^*, $n \geq 2$, such that $\mathcal{A}$ finds the unique worst possible tour in K_n^*.

While studying TSP we normally consider only feasible solutions (tours), for several other problems some authors take into consideration also infeasible solutions [10]. One example is the *Maximum Independent Set* problem, where given a graph G, the aim is to find an independent set in G of maximum cardinality. Every non-empty set of vertices is considered to be a solution by Berend, Skiena and Twitto [10]. To avoid dealing with infeasible solutions (and, thus, reserving the term 'solution' only for feasible solutions) we also use the notion of the *blackball number* introduced in [10]. The *blackball number* $\mathrm{bbn}(\mathcal{H}, \mathcal{I})$ of $\mathcal{H}$ *for a an instance* $\mathcal{I}$ of $\mathcal{P}$ is the number of solutions of $\mathcal{I}$ that are better than the solution produced by $\mathcal{H}$. The *blackball number* $\mathrm{bbn}(\mathcal{H}, n)$ of $\mathcal{H}$ is the maximum of $\mathrm{domn}(\mathcal{H}, \mathcal{I})$ over all instances $\mathcal{I}$ of size n.

When the number of solutions depends not only on the size of the instance of the CO problem at hand (for

example, the number of independent sets of vertices in a graph G on n vertices depends on the structure of G), the domination ratio of an algorithm $\mathcal{A}$ is of interest: the *domination ratio* of $\mathcal{A}$, $\mathrm{domr}(\mathcal{A}, n)$, is the minimum of $\mathrm{domn}(\mathcal{A}, \mathcal{I})/\mathrm{sol}(\mathcal{I})$, where $\mathrm{sol}(\mathcal{I})$ is the number of solutions of $\mathcal{I}$, taken over all instances $\mathcal{I}$ of size n. Clearly, domination ratio belongs to the interval $(0, 1]$ and exact algorithms are of domination ratio 1.

Methods

Currently, there are two powerful methods in DA. One is used to prove that the heuristic under consideration is of domination number 1. For this method to be useful, the heuristic has to be a greedy-type algorithm for a CO problem on independence systems. We describe the method and its applications in the subsection Greedy-Type Algorithms. The other method is used prove that the heuristic under consideration is of very large domination number. For many problems this follows from the fact that the heuristic always finds a solution that is not worse than the average solution. This method is described in the subsection Better-Than-Average Heuristics.

Greedy-Type Algorithms

The main practical message of this subsection is that one should be careful while using the classical greedy algorithm and its variations in combinatorial optimization (CO): there are many instances of CO problems for which such algorithms will produce the unique worst possible solution. Moreover, this is true for several well-known optimization problems and the corresponding instances are not exotic, in a sense. This means that not always the paradigm of greedy optimization provides any meaningful optimization at all.

An *independence system* is a pair consisting of a finite set E and a family $\mathcal{F}$ of subsets (called *independent sets*) of E such that (I1) and (I2) are satisfied.

(I1) the empty set is in $\mathcal{F}$;
(I2) If $X \in \mathcal{F}$ and Y is a subset of X, then $Y \in \mathcal{F}$.

All maximal sets of $\mathcal{F}$ are called *bases*. An independence system is *uniform* if all its bases are of the same cardinality.

Many combinatorial optimization problems can be formulated as follows. We are given an independence system $(E,\mathcal{F})$, a set $W \subseteq \mathbb{Z}_+$ and a weight function w that assigns a weight $w(e) \in W$ to every element of E ($\mathbb{Z}_+$ is the set of non-negative integers). The weight $w(S)$ of $S \in \mathcal{F}$ is defined as the sum of the weights of the elements of S. It is required to find a base $B \in \mathcal{F}$ of minimum weight. We will consider only such problems and call them the *(E,F,W)-optimization problems*.

If $S \in \mathcal{F}$, then let $I(S) = \{x : S \cup \{x\} \in \mathcal{F}\} - S$. This means that $I(S)$ consists of those elements from E-S, which can be added to S, in order to have an independent set of size $|S| + 1$. Note that by (I2) $I(S) \neq \emptyset$ for every independent set S which is not a base.

The greedy algorithm tries to construct a minimum weight base as follows: it starts from an empty set X, and at every step it takes the current set X and adds to it a minimum weight element $e \in I(X)$, the algorithm stops when a base is built. We assume that the greedy algorithm may choose any element among equally weighted elements in $I(X)$. Thus, when we say that the greedy algorithm *may construct* a base B, we mean that B is built provided the appropriate choices between elements of the same weight are made.

An *ordered partitioning* of an ordered set $Z = \{z_1, z_2, \ldots, z_k\}$ is a collection of subsets $A_1, A_2, \ldots, A_q$ of Z such that if $z_r \in A_i$ and $z_s \in A_j$ where $1 \leq i < j \leq q$ then $r < s$. Some of the sets A_i may be empty and $\cup_{i=1}^{q} A_i = Z$.

The following theorem by Bang-Jensen, Gutin and Yeo [6] characterizes all uniform independence systems $(E,\mathcal{F})$ for which there is an assignment of weights to the elements of E such that the greedy algorithm solving the $(E, \mathcal{F}, \{1, 2, \ldots, r\})$-optimization problem may construct the unique worst possible solution.

Theorem 1 *Let $(E,\mathcal{F})$ be a uniform independence system and let $r \geq 2$ be a natural number. There exists a weight assignment $w : E \to \{1, 2, \ldots, r\}$ such that the greedy algorithm may produce the unique worst possible base if and only if $\mathcal{F}$ contains some base B with the property that for some ordering $x_1, \ldots, x_k$ of the elements of B and some ordered partitioning $A_1, A_2, \ldots, A_r$ of $x_1, \ldots, x_k$ the following holds for every base $B' \neq B$ of $\mathcal{F}$:*

$$\sum_{j=0}^{r-1} |I(A_{0,j}) \cap B'| < \sum_{j=1}^{r} j \cdot |A_j| , \qquad (1)$$

where $A_{0,j} = A_0 \cup \cdots \cup A_j$ and $A_0 = \emptyset$.

The special case $r = 2$ has an 'easier' characterization also proved in [6].

Theorem 2 *Let $(E,\mathcal{F})$ be a uniform independence system. For every choice of distinct natural numbers a,b there exists a weight function $w\colon E \to \{a,b\}$ such that the greedy algorithm may produce the unique worst base if and only if $\mathcal{F}$ contains a base $B = \{x_1, x_2, \ldots, x_k\}$ such that for some $1 \le i < k$ the following holds:*

(a) *If B' is a base such that $\{x_1, \ldots, x_i\} \subseteq B'$ then $B' = B$.*
(b) *If B' is a base such that $\{x_{i+1}, \ldots, x_k\} \subseteq B'$ then $B' = B$.*

Using Theorem 1, the authors of [6] proved the following two corollaries.

Corollary 3 *Consider STSP as an $(E,\mathcal{H},W)$-optimization problem. Let $n \ge 3$.*

(a) *If $n \ge 4$ and $|W| \le \lfloor \frac{n-1}{2} \rfloor$, then the greedy algorithm never produces the unique worst possible base (i. e., tour).*
(b) *If $n \ge 3$, $r \ge n-1$ and $W = \{1,2,\ldots,r\}$, then there exists a weight function $w\colon E \to \{1,2,\ldots,r\}$ such that the greedy algorithm may produce the unique worst possible base (i. e., tour).*

Corollary 4 *Consider ATSP as an $(E,\mathcal{H},W)$-optimization problems. Let $n \ge 3$.*

(a) *If $|W| \le \lfloor \frac{n-1}{2} \rfloor$, then the greedy algorithm never produces the unique worst possible base (i. e., tour).*
(b) *For every $r \ge \lceil \frac{n+1}{2} \rceil$ there exists a weight function $w\colon E(K_n^*) \to \{1,2,\ldots,r\}$ such that the greedy algorithm may produce the unique worst possible base (i. e., tour).*

Let $\mathcal{F}$ be the sets of those subsets X of $E(K_{2n})$ which induce a bipartite graph with at most n vertices in each partite set. Then $(E(K_{2n}),\mathcal{F})$ is a uniform independence system and the bases of $(E(K_{2n}),\mathcal{F})$ correspond to copies of the complete balanced bipartite graph $K_{n,n}$ in K_{2n}. The $(E(K_{2n}),\mathcal{F},\mathbb{Z}_+)$-optimization problem is called the *Minimum Bisection Problem*. Theorem 2 implies the following:

Corollary 5 [6] *Let $n \ge 4$. The greedy algorithm for the $(E(K_{2n}),\mathcal{F},W)$-optimization problem may produce the unique worst solution even if $|W| = 2$.*

For $W = \mathbb{Z}_+$, the following sufficient condition can often be used:

Theorem 6 [21] *Let $(E,\mathcal{F})$ be an independence system which has a base $B' = \{x_1, x_2, \ldots, x_k\}$ such that the following holds for every base $B \in \mathcal{F}$, $B \ne B'$,*

$$\sum_{j=0}^{k-1} |I(x_1, x_2, \ldots, x_j) \cap B| < k(k+1)/2 \,.$$

Then the greedy algorithm for the $(E,\mathcal{F},\mathbb{Z}_+)$-optimization problem may produce the unique worst solution.

Gutin, Yeo and Zverovich [23] considered the well-known *nearest neighbor (NN)* TSP heuristic: the tour starts at any vertex x of the complete directed or undirected graph; we repeat the following loop until all vertices have been included in the tour: add to the tour a vertex (among vertices not yet in the tour) closest to the vertex last added to the tour. It was proved in [23] that the domination number of NN is 1 for any $n \ge 3$.

Bendall and Margot [8] studied greedy-type algorithms for many CO problems. Greedy-type algorithms were introduced in [18]. They include NN and were defined as follows. A *greedy-type* algorithm $\mathcal{H}$ is similar to the greedy algorithm: start with the partial solution $X = \emptyset$; and then repeatedly add to X an element of minimum weight in $I_{\mathcal{H}}(X)$ (ties are broken arbitrarily) until X is a base of $\mathcal{F}$, where $I_{\mathcal{H}}(X)$ is a subset of $I(X)$ that does not depend on the cost function c, but only on the independence system $(E,\mathcal{F})$ and the set X. Moreover, $I_{\mathcal{H}}(X)$ is non-empty if $I(X) \ne \emptyset$, a condition that guarantees that $\mathcal{H}$ always outputs a base. Bendall and Margot [8] obtained complicated sufficient conditions for an independent system $(E,\mathcal{F})$ that ensure that every greedy-type algorithm is of domination number 1 for the $(E,\mathcal{F},\mathbb{Z}_+)$-optimization problem.

The conditions imply that every greedy-type algorithm is of domination number 1 for the following classical CO problems [8]: (1) The Minimum Bisection Problem; (2) The *k-Clique Problem*: find a set of k vertices in a complete graph so that the sum of the weights of the edges between them is minimum; (3) ATSP; (4) STSP; (5) The *MinMax Matching Subgraph Problem*:

find a maximal (with respect to inclusion) matching so that the sum of the weights of the edges in the matching is minimum; (6) The *Assignment Problem*: find a perfect matching in a weighted complete bipartite graph so that the sum of the weights of the edges in the matching is minimum.

Better-Than-Average Heuristics

The idea of this method is to show that a heuristic is of very large domination number if it always produces a solution that is not worse than the average solution. The first such result was proved by Rublineckii [31] for the STSP.

Theorem 7 *Every STSP heuristic that always produces a tour not worse than the average tour (of the instance) is of domination number at least* $(n-2)!$ *when n is odd and* $(n-2)!/2$ *when n is even.*

The following similar theorem was proved by Sarvanov [32] for n odd and Gutin and Yeo [20] for n even.

Theorem 8 *Every ATSP heuristic that always produces a tour not worse than the average tour (of the instance) is of domination number at least* $(n-2)!$ *for each* $n \neq 6$.

The two theorems has been used to prove that a wide variety of TSP heuristics have domination number at least $\Omega((n-2)!)$. We discuss two families of such heuristics.

Consider an instance of the ATSP (STSP). Order the vertices $x_1, x_2, \ldots, x_n$ of K_n^* (K_n) using some rule. The *generic vertex insertion algorithm* proceeds as follows. Start with the cycle $C_2 = x_1x_2x_1$. Construct the cycle C_j from C_{j-1} ($j = 3, 4, 5, \ldots, n$), by inserting the vertex x_j into $C_{j\text{-}1}$ at the optimum place. This means that for each arc $e = xy$ which lies on the cycle $C_{j\text{-}1}$ we compute $w(xx_j) + w(x_jy) - w(xy)$, and insert x_j into the arc $e = xy$, which obtains the minimum such value. Here *w(uv)* denotes the weight of an arc *uv*. E.M. Lifshitz (see [31]) was the first to prove that the generic vertex insertion algorithm always produces a tour not worse than the average tour. Thus, we have the following:

Corollary 9 *The generic vertex insertion algorithm has domination number at least* $(n-2)!$ $(n \neq 6)$.

In TSP local search (LS) heuristics, a neighborhood *N(T)* is assigned to every tour *T*; *N(T)* is a set of tours in some sense close to *T*. The *best improvement* LS proceeds as follows. We start from a tour T_0. In the *i*'th iteration ($i \geq 1$), we search in the neighborhood $N(T_{i-1})$ for the best tour T_i. If the weights of $T_{i\text{-}1}$ and T_i do not coincide, we carry out the next iteration. Otherwise, we output T_i.

The *k-Opt*, $k \geq 2$, neighborhood of a tour T consists of all tours that can be obtained by replacing a collection of k edges (arcs) by a collection of k edges (arcs). It is easy to see that one iteration of the best improvement k-Opt LS can be completed in time $O(n^k)$. Rublineckii [31] showed that every local optimum for the best improvement 2-Opt or 3-Opt LS for STSP is of weight at least the average weight of a tour and, thus, by Theorem 7 is of domination number at least $(n-2)!/2$ when n is even and $(n-2)!$ when n is odd. Observe that this result is of restricted interest since to reach a k-Opt local optimum one may need exponential time (cf. Section 3 in [26]). However, Punnen, Margot and Kabadi [29] managed to prove that, in polynomial time, the best improvement 2-Opt and 3-Opt LS's for STSP produce a tour of weight at least the average weight of a tour. Thus, we have the following:

Corollary 10 *For the STSP the best improvement 2-Opt LS produces a tour, which is not worse than at least* $\Omega((n-2)!)$ *other tours, in at most* $O(n^3 \log n)$ *iterations.*

Corollary 10 is also valid for the best improvement 3-Opt LS and some other LS heuristics for TSP, see [24,29]. In the next section, we will give further examples of better-than-average heuristics for problems other than TSP.

Cases

Traveling Salesman Problem

In the previous sections, we discussed several TSP heuristics. However, there are many more TSP heuristics and, in this subsection, we consider some of them. In the next subsection, some general upper bounds are given on the domination number of TSP heuristics.

Gutin, Yeo and Zverovich [23] considered the *repeated nearest neighbor (RNN)* heuristic, which is the following variation of the NN heuristic: construct n tours by starting NN from each vertex of the complete (di)graph and choose the best tour among the n tours. The authors of [23] proved that for the ATSP, RNN al-

ways produces a tour, which is not worse than at least $n/2-1$ other tours, but for some instance it finds a tour, which is not worse than at most *n-2* other tours, $n \geq 4$. We also show that, for some instance of the STSP on $n \geq 4$ vertices, RNN produces a tour not worse than at most 2^{n-3} tours.

Another ATSP heuristic, max-regret-fc (fc abbreviates First Coordinate), was first introduced by Ghosh et al. [13]. Extensive computational experiments in [13] demonstrated a clear superiority of max-regret-fc over the greedy algorithm and several other construction heuristics from [14]. Therefore, the result of Theorem 11 obtained by Gutin, Goldengorin and Huang [15] was somewhat unexpected.

Let K_n^* be a complete digraph with vertices $V = \{1, 2, \ldots, n\}$. The weight of an arc *(i,j)* is denoted by w_{ij}. Let Q be a collection of disjoint paths in K_n^*. An arc *a=(i,j)* is a *feasible addition* to Q if $Q+a$ is either a collection of disjoint paths or a tour in K_n^*. Consider the following two ATSP heuristics: max-regret-fc and max-regret.

The heuristic *max-regret-fc* proceeds as follows. Set $W = T = \emptyset$. While $V \neq W$ do the following: For each $i \in V \setminus W$, compute two lightest arcs *(i,j)* and *(i,k)* that are feasible additions to T, and compute the difference $\Delta_i = |w_{ij} - w_{ik}|$. For $i \in V \setminus W$ with maximum Δ_i choose the lightest arc *(i,j)*, which is a feasible addition to T and add *(i,j)* to M and i to W.

The heuristic *max-regret* proceeds as follows. Set $W^+ = W^- = T = \emptyset$. While $V \neq W^+$ do the following: For each $i \in V \setminus W^+$, compute two lightest arcs *(i,j)* and *(i,k)* that are feasible additions to T, and compute the difference $\Delta_i^+ = |w_{ij} - w_{ik}|$; for each $i \in V \setminus W^-$, compute two lightest arcs *(j,i)* and *(k,i)* that are feasible additions to T, and compute the difference $\Delta_i^- = |w_{ji} - w_{ki}|$. Compute $i' \in V \setminus W^+$ with maximum $\Delta_{i'}^+$ and $i'' \in V \setminus W^-$ with maximum $\Delta_{i''}^-$. If $\Delta_{i'}^+ \geq \Delta_{i''}^-$ choose the lightest arc (i', j'), which is a feasible addition to T and add (i', j') to M, i' to W^+ and j' to W^-. Otherwise, choose the lightest arc (j'', i''), which is a feasible addition to T and add (j'', i'') to M, i'' to W^- and j'' to W^+.

Notice that in max-regret-fc, if $|V \setminus W| = 1$ we set $\Delta_i = 0$. A similar remark applies to max-regret.

Theorem 11 *The domination number of both max-regret-fc and max-regret equals 1 for each $n \geq 2$.*

Upper Bounds for Domination Numbers of ATSP Heuristics

It is realistic to assume that any ATSP algorithm spends at least one unit of time on every arc of K_n^* that it considers. We use this assumption in this subsection.

Theorem 12 [17] *Let $\mathcal{A}$ be an ATSP heuristic of running time t(n). Then the domination number of $\mathcal{A}$ does not exceed* $\max_{1 \leq n' \leq n}(t(n)/n')^{n'}$.

Corollary 13 [17] *Let $\mathcal{A}$ be an ATSP heuristic of complexity t(n). Then the domination number of $\mathcal{A}$ does not exceed* $\max\{e^{t(n)/e}, (t(n)/n)^n\}$, *where e is the basis of natural logarithms.*

The next assertion follows directly from the proof of Corollary 13.

Corollary 14 [17] *Let $\mathcal{A}$ be an ATSP heuristic of complexity t(n). For $t(n) \geq en$, the domination number of $\mathcal{A}$ does not exceed* $(t(n)/n)^n$.

We finish this subsection with a result from [17] that improves (and somewhat clarifies) Theorem 20 in [29].

Theorem 15 *Unless P = NP, there is no polynomial time ATSP algorithm of domination number at least* $(n-1)! - \lfloor n - n^\alpha \rfloor !$ *for any constant* $\alpha < 1$.

Multidimensional Assignment Problem (MAP)

In case of s dimensions, MAP is abbreviated by s-AP and defined as follows. Let $X_1 = X_2 = \cdots = X_s = \{1, 2, \ldots, n\}$. We will consider only vectors that belong to the Cartesian product $X = X_1 \times X_2 \times \cdots \times X_s$. Each vector e is assigned a weight $w(e)$. For a vector e, e_j denotes its jth coordinate, i. e., $e_j \in X_j$. A *partial assignment* is a collection $e^1, e^2, \ldots, e^t$ of $t \leq n$ vectors such that $e_j^i \neq e_j^k$ for each $i \neq k$ and $j \in \{1, 2, \ldots, s\}$. An *assignment* is a partial assignment with n vectors. The *weight* of a partial assignment $A = \{e^1, e^2, \ldots, e^t\}$ is $w(A) = \sum_{i=1}^t w(e^i)$. The objective is to find an assignment of minimum weight. Notice that s-AP has $(n!)^{s-1}$ solutions (assignments).

s-AP can be considered as the $(X, \mathcal{F}, \mathbb{Z}_+)$-optimization problem. ($\mathcal{F}$ consists of partial assignments including the empty one.) This allows us to define the greedy algorithm for s-AP and to conclude from Theorem 6 that the greedy algorithm is of domination number 1 (for every fixed $s \geq 3$).

In the subsection Traveling Salesman Problem, we considered the *max-regret-fc and max-regret heuristics*. In fact, max-regret was first introduced for 3-AP by Balas and Saltzman [4]. (See [15] for detailed description of the s-AP max-regret-fc and max-regret heuristics for each $s \geq 2$.) In computational experiments, Balas and Saltzman [4] compared the greedy algorithm with max-regret and concluded that max-regret is superior to the greedy algorithm with respect to the quality of solutions. However, after conducting wider computational experiments, Robertson [30] came to a different conclusion: the greedy algorithm and max-regret are of similar quality for 3-AP. Gutin, Goldengorin and Huang [15] share the conclusion of Robertson: both max-regret and max-regret-fc are of domination number 1 (similarly to the greedy algorithm) for s-AP for each $s \geq 3$. Moreover, there exists a family of s-AP instances for which all three heuristics will find the unique worst assignment [15] (for each $s \geq 3$).

Similarly to TSP, we may obtain MAP heuristics of factorial domination number if we consider better-than-average heuristics. This follows from the next theorem:

Theorem 16 [15] *Let $\mathcal{H}$ be a heuristic that for each instance of s-AP constructs an assignment of weight at most the average weight of an assignment. Then the domination number of $\mathcal{H}$ is at least* $((n-1)!)^{s-1}$.

Balas and Saltzman [4] introduced a *3-Opt heuristic* for 3-AP which is similar to the 3-Opt TSP heuristic. The *3-Opt neighborhood* of an assignment $A = \{e^1, e^2, \ldots, e^n\}$ is the set of all assignments that can be obtained from A by replacing a triple of vectors with another triple of vectors. The 3-Opt is a local search heuristic that uses the 3-Opt neighborhood. It is proved in [15] that an assignment, that is the best in its 3-Opt neighborhood, is at least as good as the average assignment. This implies that 3-Opt is of domination number at least $((n-1)!)^2$. We cannot guarantee that 3-Opt local search will stop after polynomial number of iterations. Moreover, 3-Opt is only for 3-AP. Thus, the following heuristic introduced and studied in [15] is of interest.

Recursive Opt Matching (ROM) proceeds as follows. Compute a new weight $\bar{w}(i,j) = w(X_{ij})/n^{s-2}$, where X_{ij} is the set of all vectors with last two coordinates equal i and j, respectively. Solving the 2-AP with the new weights to optimality, find an optimal assignment $\{(i, \pi_s(i)) : i = 1, 2, \ldots, n\}$, where π_s is a permutation on X_s. While $s \neq 1$, introduce *(s-1)*-AP with weights given as follows: $w'(f^i) = w(f^i, \pi_s(i))$ for each vector $f^i \in X'$, where $X'n = X_1 \times X_2 \times \cdots \times X_{s-1}$, with last coordinate equal i and apply ROM recursively. As a result we have obtained permutations $\pi_s, \pi_{s-1}, \ldots, \pi_2$. The output is the assignment $\{(i, \pi_2(i), \ldots, \pi_s(\pi_{s-1}(\ldots(\pi_2(i)))\ldots)) : i = 1, 2, \ldots, n\}$.

Clearly, ROM is of running time $O(n^3)$ for every fixed $s \geq 3$. Using Theorem 16, it is proved in [15] that ROM is of domination number at least $((n-1)!)^{s-1}$.

Minimum Partition and Multiprocessor Scheduling Problems

In this subsection, N always denotes the set $\{1, 2, \ldots, n\}$ and each $i \in N$ is assigned a positive integral weight $\sigma(i)$. $\mathcal{A} = (A_1, A_2, \ldots, A_p)$ is a *p-partition* of N if each $A_i \subseteq N$, $A_i \cap A_j = \emptyset$ for each $i \neq j$ and the union of all sets in $\mathcal{A}$ equals N. For a subset A of N, $\sigma(A) = \sum_{i \in A} \sigma(i)$. The *Minimum Multiprocessor Scheduling Problem* (MMS) [3] can be stated as follows. We are given a triple (N, σ, p), where p is an integer, $p \geq 2$. We are required to find a p-partition C of N that minimizes $\sigma(\mathcal{A}) = \max_{1 \leq i \leq p} \sigma(A_i)$ over all p-partitions $\mathcal{A} = (A_1, A_2, \ldots, A_p)$ of N.

Clearly, if $p \geq n$, then MMS becomes trivial. Thus, in what follows, $p < n$. The size s of MMS is $\Theta(n + \sum_{i=1}^{n} \log \sigma(i))$. Consider the following heuristic $\mathcal{H}$ for MMS. If $s \geq p^n$, then we simply solve the problem optimally. This takes $O(s^2)$ time, as there are at most $O(s)$ solutions, and each one can be evaluated and compared to the current best in $O(s)$ time. If $s < p^n$, then we sort the elements of the sequence $\sigma(1), \sigma(2), \ldots, \sigma(n)$. For simplicity of notation, assume that $\sigma(1) \geq \sigma(2) \geq \cdots \geq \sigma(n)$. Compute $r = \lceil \log n / \log p \rceil$ and solve MMS for $(\{1, 2, \ldots, r\}, \sigma, p)$ to optimality. Suppose we have obtained a p-partition $\mathcal{A}$ of $\{1, 2, \ldots, r\}$. Now for i from *r+1* to n add i to the set A_j of the current p-partition $\mathcal{A}$ with smallest $\sigma(A_j)$. The following result was proved by Gutin, Jensen and Yeo [16].

Theorem 17 *The heuristic $\mathcal{H}$ runs in time $O(s^2 \log s)$ and* $\lim_{s \to \infty} \mathrm{domr}(\mathcal{H}, s) = 1$.

The *Minimum Partition Problem (MP)* is MMS with *p=2*. Alon, Gutin and Krivelevich [2] proved Theorem 17 for MP with s replaced by n.

Max Cut Problem

The *Max Cut (MC)* is the following problem: given a weighted graph $G=(V,E)$, find a bipartition (a *cut*) (X,Y) of V such that the sum of weights of the edges with one end vertex in X and the other in Y, called the *weight of the cut* (X,Y), is maximum. For this problem, there are some better-than-average heuristics. The simplest is probably the following greedy-like heuristic C; order the vertices arbitrarily and put each vertex in its turn either in X or in Y in order to maximize in each step the total weight of crossing edges.

Using an advanced probabilistic approach Alon, Gutin and Krivelevich [2] proved that the heuristic C is of domination ratio larger than 0.025. For the unweighted MC (all weights are equal), a better quality algorithm can be designed as described in [2]. Its domination ratio is at least $1/3 - o(1)$.

Constraint Satisfaction Problems

Let r be a fixed positive integer, and let $\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$ be a collection of Boolean functions, each involving at most r of the n variables, and each having a positive weight $w(f_i)$. The *Max-r-Constraint Satisfaction Problem* (or *Max-r-CSP*, for short), is the problem of finding a truth assignment to the variables so as to maximize the total weight of the functions satisfied. Note that this includes, as a special case, the Max Cut problem. Another interesting special case is the Max-r-SAT problem, in which each of the Boolean functions f_i is a clause of at most r literals.

Alon, Gutin and Krivelevich [2] proved the following:

Theorem 18 *For each fixed integer $r \geq 1$ there exists a linear time algorithm for the Max-r-CSP problem, whose domination ratio exceeds $\frac{1}{2^{4/3} \cdot 2^{6r}}$.*

Vertex Cover, Independent Set and Clique Problems

A *clique* in a graph G is a set of vertices in G such that every pair of vertices in the set are connected by an edge. The *Maximum Clique Problem* (*MCl*) is the problem of finding a clique of maximum cardinality in a graph. A *vertex cover* in a graph G is a set S of vertices in G such that every edge is incident to a vertex in S. The *Minimum Vertex Cover Problem* (*MVC*) is the problem of finding a minimum cardinality vertex cover. An *independent set* in a graph is a set S of vertices such that no edge joins two vertices in S. The *Maximum Independent Set Problem* (*MIS*) is the problem of finding a minimum cardinality independent set in a graph. It is easy to see that the number of cliques and independent sets in a graph depends on its structure, and not only on the number of vertices. The same holds for vertex covers.

Notice that if C is a vertex cover of a graph G, then $V(G) \setminus C$ is an independent set in G; if Q is a clique in G, then Q is an independent set in the complement of G. These well-known facts imply that if there is a heuristic for one of the problem of domination ratio at least $r(n)$, all other problems admit a heuristic of domination ratio at least $r(n)$.

MCl, MIS and MVC are somewhat different from the previous problems we have considered. Firstly, the number of feasible solutions, for an input of size n, depends on the actual input, and not just its size. The second difference is that the three problems do not admit polynomial-time heuristics of domination ratio at least $1/p(n)$ for any polynomial $p(n)$ in n unless P=NP. This was proved by Gutin, Vainshtein and Yeo [19].

Because of the first difference, it is better to compare heuristics for the problems using the blackball number rather than domination number. Since a heuristic for MVC can be easy transformed into a heuristic for the other two problems, we restrict ourselves only to MVC heuristics.

The *incremental deletion heuristic* starts with an arbitrary permutation π of vertices of G and an initial solution $S=V(G)$. We consider each vertex of G in turn (according to π), deleting it from S if the resulting subset remains a (feasible) solution. A seemingly better heuristic for MVC is obtained by ordering the vertices by degree (lower degrees first), and then applying the incremental deletion heuristic. We call it the *increasing-degree deletion heuristic*. The well-known *maximal matching heuristic* constructs a maximal matching M and outputs both end-vertices of all edges in M as a solution. Berend, Skiena and Twitto [10] proved that the incremental deletion heuristic (increasing-degree deletion heuristic, maximal matching heuristic) is of blackball number $2^{n-1} - n$ (of blackball number larger than $2 - \epsilon)^n$ for each $\epsilon > 0$, of blackball number approximately 1.839^n). Clearly, the maximal matching heuris-

tic is the best among the three heuristics from the DA point of view.

Quadratic Assignment Problem

The *Quadratic Assignment Problem* (*QAP*) can be formulated as follows. We are given two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ of integers. Our aim is to find a permutation π of $\{1, 2, \ldots, n\}$ that minimizes the sum

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)} .$$

Gutin and Yeo [23] described a better-than-average heuristic for QAP and proved that the heuristic is of domination number at least $n!/\beta^n$ for each $\beta > 1$. Moreover, the domination number of the heuristic is at least $(n-2)!$ for every prime power n. These results were obtained using a group-theoretical approach.

See also

► Traveling Salesman Problem

References

1. Ahuja RK, Ergun Ö, Orlin JB, Punnen AP (2002) A survey of very large-scale neighborhood search techniques. Discret Appl Math 123:75–102
2. Alon N, Gutin G, Krivelevich M (2004) Algorithms with large domination ratio. J Algorithms 50:118–131
3. Ausiello G, Crescenzi P, Gambosi G, Kann V, Marchetti-Spaccamela A, Protasi M (1999) Complexity and Approximation. Springer, Berlin
4. Balas E, Saltzman MJ (1991) An algorithm for the three-index assignment problem. Oper Res 39:150–161
5. Bang-Jensen J, Gutin G (2000) Digraphs: Theory, Algorithms and Applications. Springer, London
6. Bang-Jensen J, Gutin G, Yeo A (2004) When the greedy algorithm fails. Discret Optim 1:121–127
7. Ben-Arieh D, Gutin G, Penn M, Yeo A, Zverovitch A (2003) Transformations of Generalized ATSP into ATSP: experimental and theoretical study. Oper Res Lett 31:357–365
8. Bendall G, Margot F (2006) Greedy Type Resistance of Combinatorial Problems. Discret Optim 3:288–298
9. Berge C (1958) The Theory of Graphs. Methuen, London
10. Berend B, Skiena SS, Twitto Y, Combinatorial dominance guarantees for heuristic algorithms. ACM Trans Algorithm (to appear)
11. Cornuejols G, Fisher ML, Nemhauser GL (1977) Location of bank accounts to optimize float; an analytic study of exact and approximate algorithms. Manag Sci 23:789–810
12. Deineko VG, Woeginger GJ (2000) A study of exponential neighbourhoods for the traveling salesman problem and the quadratic assignment problem. Math Prog Ser A 87:519–542
13. Ghosh D, Goldengorin B, Gutin G, Jäger G (2007) Tolerance-based greedy algorithms for the traveling salesman problem. Commun DQM 41:521–538
14. Glover F, Gutin G, Yeo A, Zverovich A (2001) Construction heuristics for the asymmetric TSP. Eur J Oper Res 129: 555–568
15. Gutin G, Goldengorin B, Huang J (2006) Worst Case Analysis of Max-Regret, Greedy and Other Heuristics for Multidimensional Assignment and Traveling Salesman Problems. Lect Notes Comput Sci 4368:214–225
16. Gutin G, Jensen T, Yeo A (2006) Domination analysis for minimum multiprocessor scheduling. Discret Appl Math 154:2613–2619
17. Gutin G, Koller A, Yeo A (2006) Note on Upper Bounds for TSP Domination Number. Algorithm Oper Res 1:52–54
18. Gutin G, Vainshtein A, Yeo A (2002) When greedy-type algorithms fail. Unpublished manuscript
19. Gutin G, Vainshtein A, Yeo A (2003) Domination Analysis of Combinatorial Optimization Problems. Discret Appl Math 129:513–520
20. Gutin G, Yeo A (2002) Polynomial approximation algorithms for the TSP and the QAP with a factorial domination number. Discret Appl Math 119:107–116
21. Gutin G, Yeo A (2002) Anti-matroids. Oper Res Lett 30: 97–99
22. Gutin G, Yeo A (2005) Domination Analysis of Combinatorial Optimization Algorithms and Problems. In: Golumbic M, Hartman I (eds) Graph Theory, Combinatorics and Algorithms: Interdisciplinary Applications. Springer, New York, pp 152–176
23. Gutin G, Yeo A, Zverovitch A (2002) Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. Discret Appl Math 117:81–86
24. Gutin G, Yeo A, Zverovitch A (2002) Exponential Neighborhoods and Domination Analysis for the TSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 223–256
25. Johnson DS, Gutin G, McGeoch LA, Yeo A, Zhang X, Zverovitch A (2002) Experimental Analysis of Heuristics for ATSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 445–487
26. Johnson DS, McGeoch LA (1997) The traveling salesman problem: A case study in local optimization. In: Aarts EHL, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, Chichester, pp 251–310
27. Johnson DS, McGeoch LA (2002) Experimental Analysis of Heuristics for STSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 369–443
28. Kise H, Ibaraki T, Mine H (1979) Performance analysis of six approximation algorithms for the one-machine maximum

lateness scheduling problem with ready times. J Oper Res Soc Japan 22:205–223
29. Punnen AP, Margot F, Kabadi SN (2003) TSP heuristics: domination analysis and complexity. Algorithmica 35:111–127
30. Robertson AJ (2001) A set of greedy randomized adaptive local search procedure implementations for the multidimentional assignment problem. Comput Optim Appl 19:145–164
31. Rublineckii VI (1973) Estimates of the accuracy of procedures in the Traveling Salesman Problem. Numer Math Comput Tech 4:18–23 (in Russian)
32. Sarvanov VI (1976) The mean value of the functional of the assignment problem. Vestsi Akad Navuk BSSR, Ser Fiz-Mat Navuk 2:111–114 (in Russian)
33. Zemel E (1981) Measuring the quality of approximate solutions to zero-one programming problems. Math Oper Res 6:319–332

Duality Gaps in Nonconvex Optimization

Panos Parpas, Berç Rustem
Department of Computing, Imperial College, London, UK

MSC2000: 90B50, 78M50

Article Outline

Abstract
Background
Game Theory Interpretation
Methods
Randomization
Functional Lagrange Multipliers
Conclusions
References

Abstract

Duality gaps in optimization problems arise because of the nonconvexities involved in the objective function or constraints. The Lagrangian dual of a nonconvex optimization problem can also be viewed as a two-person zero-sum game. From this viewpoint, the occurrence of duality gaps originates from the order in which the two players select their strategies. Therefore, duality theory can be analyzed as a zero-sum game where the order of play generates an asymmetry. One can conjecture that this asymmetry can be eliminated by allowing one of the players to select strategies from a larger space than that of the finite-dimensional Euclidean space. Once the asymmetry is removed, then there is zero duality gap. The aim of this article is to review two methods by which this process can be carried out. The first is based on randomization of the primal problem. The second extends the space from which the dual variables can be selected. Duality gaps are important in mathematical programming and some of the results reviewed here are more than 50 years old, but only recently methods have been discovered to take advantage of them. The theory is elegant and helps appreciate the game-theoretic origins of the dual problem and the role of Lagrange multipliers.

Background

We discuss how duality gaps arise, and how they can be eliminated in nonconvex optimization problems. A standard optimization problem is stated as follows:

$$\begin{aligned} \min \quad & f(x)\,, \\ & g(x) \leq 0\,, \\ & x \in X\,, \end{aligned} \tag{1}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ and $g\colon \mathbb{R}^n \to \mathbb{R}^m$ are assumed to be smooth and nonconvex. The feasible region of (1) is denoted by $\mathcal{F}$, and it is assumed to be nonempty and compact. X is some compact convex set.

In order to understand the origins of duality in mathematical programming, consider devising a strategy to determine whether a point, say y, is the globally optimal solution of (1). Such a strategy can be concocted as follows: if $f(y)$ is the global solution of (1) then the following system of inequalities

$$\begin{aligned} & f(x) < f(y)\,, \\ & g(x) \leq 0\,, \\ & x \in X \end{aligned} \tag{2}$$

will not have a solution. We can reformulate (2) in a slightly more convenient framework. Indeed, suppose that there exist m positive scalars λ_i, $i = 1, \ldots, m$, such that

$$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) < f(y) \tag{3}$$

has no solution. Then (2) does not have a solution either. The left-hand side of (3) is called the Lagrangian function associated with (1). It is clear from the discussion above that the Lagrangian can be used to answer questions about the optimal solutions of (1). The usefulness of the dual function emanates from the following duality observation: Let f^* be the optimal objective function value of (1), and let $L\colon \mathbb{R}^m \to \mathbb{R}$ be defined as follows:

$$L(\lambda) = \inf_{x\in X} L(x,\lambda)\,.$$

Then it is easy to prove that:

$$\sup_{\lambda\geq 0} L(\lambda) \leq f^*\,. \tag{4}$$

This result is known as the weak duality theorem, and it is valid with a quite general set of assumptions. The strong duality theorem asserts that if f and g are convex, $f^* > -\infty$, and the interior of $\mathcal{F}$ is not empty, then

$$\sup_{\lambda\geq 0} L(\lambda) = f^*\,.$$

Proofs of the weak and strong duality theorems can be found in [1,10].

Game Theory Interpretation

There is an interesting relationship between (1) and the following optimization problem:

$$\sup_{\lambda\geq 0}\inf_{x\in X}\; f(x) + \sum_{i=1}^{m}\lambda_i g_i(x)\,. \tag{5}$$

We refer to (1) as the primal problem, while (5) is referred to as the Lagrangian dual. The λ's that appear in (5) are called the Lagrange multipliers (or dual variables).

It is interesting to note that (1) can equivalently be restated as follows:

$$\inf_{x\in X}\sup_{\lambda\geq 0}\; f(x) + \sum_{i=1}^{m}\lambda_i g_i(x)\,. \tag{6}$$

The relationship between (6) (or (1)) and (5) can be analyzed as a two-person zero-sum game. In this game player A chooses the x variables, and player B chooses the λ variables. If player A chooses x', and player B chooses λ', then player A pays $L(x',\lambda')$ to player B. Naturally, player A wishes to minimize this quantity, while player B attempts to maximize it.

In game theory equilibria play an important role. An equilibrium, in the present context means a point from which no player will gain by a unilateral change of strategy. For the game outlined above an equilibrium point (x^*, λ^*) must satisfy

$$\begin{aligned} L(x,\lambda^*) &\geq L(x^*,\lambda^*) \\ &\geq L(x^*,\lambda)\quad \forall x\in X,\quad \forall\lambda\in\mathbb{R}^m_+\,. \end{aligned} \tag{7}$$

A point satisfying the preceding equation is also known as a saddle point of L. To see that (7) is an equilibrium point we argue as follows: Given that player A wishes to minimize the amount paid to player B, then it is obvious that if player B chooses λ^* and player A selects anything other than x^*, player A will be worse off. Similarly, if player A chooses x^*, then if player B chooses anything other than λ^*, then player B will be worse off.

By the strong duality theorem, we know that the game has an equilibrium point under convexity assumptions. For the general case, insight can be obtained by interpreting (5) and (6) as two different games. A saddle point will exist if the optimal values of the two games are equal.

Our next task is to interpret (5) and (6) as games. Indeed consider the following situation: Player A chooses a strategy first, and then player B chooses a strategy. Thus, player B already knows the strategy that player A has chosen. As a result player B will have an advantage. Player A will argue as follows: "If I choose x, then player B will choose $\sup_{\lambda\geq 0} L(x,\lambda)$, therefore I had better choose the strategy that will minimize my losses." In other words player A will choose the optimal strategy given by solving (6).

Now consider the same game, but with the order of play reversed, i. e., player B chooses first, and player A second. Then applying the rules of rational behavior (as above), we see that player B will select the λ that solves (5).

Consequently, duality gaps originate from the order in which the two players select their strategies. In the next section we see how this asymmetry can be eliminated by allowing one of the players to select strategies from a larger space than that of the finite-dimensional

Euclidean space. Once the asymmetry is eliminated, then there is zero duality gap.

Methods

As argued above, the player that chooses first is disadvantaged, since the other player can adjust. In this section we discuss two methods in which this asymmetry in the order of play can be eliminated. Both methods were proposed early in the history of mathematical programming. The first method proceeds by randomization (increasing the powers of player A). It is difficult to say who suggested this strategy first. Since the origins of the idea emanate from mixed strategies in game theory, one could argue that the idea was first suggested by Borel in the 1920s [14]. A modern proof can be found in [2]. The second method allows player B to select the dual variables from a larger space. This idea seems to have been suggested by Everett [3], and then by Gould [6]. A review can be found in [11]. Algorithms that attempt to reduce the duality gap appeared in [4,5,7,8,9,12].

Randomization

Assume that player A chooses first, then the game can be described by

$$P^* = \inf_{x\in X} \sup_{\lambda\geq 0} L(x,\lambda)\,,$$

and in general by

$$P^* \geq D^* = \sup_{\lambda\geq 0} \inf_{x\in X} L(x,\lambda)\,.$$

Player A has a handicap since player B will choose a strategy knowing what player A will do. In order to avoid having a duality gap, we consider giving more flexibility to player A. We thus allow player A to choose strategies from $\mathcal{M}(X)$, where $\mathcal{M}(X)$ denotes the space of probability measures on $\mathcal{B}$ (the σ-field generated by X). Player A will therefore choose a strategy by solving

$$\begin{aligned} P^* = \inf_{\mu\in\mathcal{M}(X)} & \int_X f(x)\,\mathrm{d}\mu(x) \\ & \int_X g(x)\,\mathrm{d}\mu(x) \leq 0 \\ & \int_X \mathrm{d}\mu(x) = 1\,. \end{aligned} \tag{8}$$

Equivalently:

$$\begin{aligned} P^* = \inf_{\mu\in\mathcal{M}(X)} \sup_{\lambda\geq 0} & \int_X f(x)\,\mathrm{d}\mu(x) \\ & + \sum_{i=1}^m \lambda_i \int_X g(x)\,\mathrm{d}\mu(x) + \lambda_0 \left(\int_X \mathrm{d}\mu(x) - 1 \right)\,. \end{aligned}$$

The dual of (8) is given by

$$\begin{aligned} D^* = \sup_{\lambda\geq 0} \inf_{\mu\in\mathcal{M}(X)} & \int_X f(x)\,\mathrm{d}\mu(x) \\ & + \sum_{i=1}^m \lambda_i \int_X g(x)\,\mathrm{d}\mu(x) + \lambda_0 \left(\int_X \mathrm{d}\mu(x) - 1 \right)\,. \end{aligned}$$

Then it can be shown that $P^* = D^*$. The proof is beyond the scope of this article; it can be found in [2].

Functional Lagrange Multipliers

We now consider the case where player B chooses first. From the previous section, it follows that player B will choose a strategy according to:

$$D^* = \sup_{\lambda\geq 0} \inf_{x\in X} L(x,\lambda)\,. \tag{9}$$

We have already pointed out that the following holds:

$$D^* \leq \inf_{x\in X} \sup_{\lambda\geq 0} L(x,\lambda)\,.$$

In order for the preceding equation to hold as an equality, without any convexity assumptions, we consider increasing the space of available strategies of B. This was suggested in [3,6]. The exposition here is based on [11]. Let $\mathcal{H}$ denote all the feasible right-hand sides for (1):

$$\mathcal{H} = \{b \in \mathbb{R}^m \mid \exists x \in X\colon g(x) \leq b\}\,.$$

Let $\mathcal{D}$ denote the following set of functions:

$$\begin{aligned} \mathcal{D} = \{z\colon \mathbb{R}^m \to \mathbb{R} \mid & z(d_1) \leq z(d_2),\ \text{if } d_1 \leq d_2, \\ & \forall d_1, d_2 \in \mathcal{H}\}\,. \end{aligned}$$

The following dual can be defined using the concepts above:

$$\begin{aligned} D^* = \sup_z\ & z(0) \\ & z(g(x)) \leq f(x) \quad \forall x \in X \quad z \in \mathcal{D}\,. \end{aligned} \tag{10}$$

The dual in (10) is different from the type of duals that we have been discussing in this article. If, however, we

assume that $c + \mathcal{D} \subset \mathcal{D}$, then it was shown in [11] that (10) is equivalent to the following:

$$D^* = \sup_{z \in \mathcal{D}} \inf_{x \in X} f(x) + z(g(x))$$

dual problem. A proof that the duality gap between (10) and (1) is zero can be found in [11].

Conclusions

We have discussed Lagrangian duality, and the existence of duality gaps from a game-theoretic viewpoint. We have discussed two ways in which duality gaps can be eliminated. The first is randomization and the second is the use of functional Lagrange multipliers. Unfortunately none of the two methods are immediately applicable to real-world problems. However, for certain classes of problems the functional Lagrange multiplier approach can be useful. It was shown in [13] that if the original problem involves the optimization of polynomial functions, and if the Lagrange multipliers are allowed to be themselves polynomials then there will be no duality gap. Unlike the general case discussed in this article, polynomial Lagrange multipliers can be manipulated numerically. This approach can potentially help develop efficient algorithms for a large class of problems.

References

1. Bertsekas DP (1999) Nonlinear Programming, 2nd edn. Athena Scientific, Belmont
2. Ermoliev Y, Gaivoronski A, Nedeva C (1985) Stochastic optimization problems with incomplete information on distribution functions. SIAM J Control Optim 23(5):697–716
3. Everett H (1963) Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Oper Res 11:399–417
4. Floudas CA, Visweswaran V (1990) A global optimization algorithm (gop) for certain classes of nonconvex nlps: I. theory. Comput Chem Eng 14(12):697–716
5. Floudas CA, Visweswaran V (1993) A primal-relaxed dual global optimization approach. J Optim Theory Appl 78(2): 187–225
6. Gould FJ (1972) Nonlinear duality theorems. Cahiers Centre Études Recherche Opér 14:196–212
7. Liu WB, Floudas CA (1993) A remark on the GOP algorithm for global optimization. J Global Optim 3(4):519–521
8. Liu WB, Floudas CA (1996) A generalized primal-relaxed dual approach for global optimization. J Optim Theory Appl 90(2):417–434
9. Rajasekaran S, Pardalos PM, Reif JH, Rolim J (eds) (2001) Combinatorial Optimization. In: Handbook of randomized computing, vol I, II, vol 9. Kluwer, Dordrecht
10. Rockafellar RT (1997) Convex analysis. Princeton Landmarks in Mathematics. Princeton Univ. Press, Princeton
11. Tind J, Wolsey LA (1981) An elementary survey of general duality theory in mathematical programming. Math Programm 21(3):241–261
12. Visweswaran V, Floudas CA (1990) A global optimization algorithm (gop) for certain classes of nonconvex nlps: II. applications of theory and test problems. Comput Chem Eng 14(12):1417–1434
13. Waki H, Kim S, Kojima M, Muramatsu M (2006) Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. SIAM J Optim 17(1):218–242
14. Weintraub ER (ed) (1992) Toward a history of game theory. Duke University Press, Durham, Annual supplement to History of Political Economy, vol 24

Duality in Optimal Control with First Order Differential Equations

Sabine Pickenhain
Brandenburg Technical University Cottbus, Cottbus, Germany

MSC2000: 49K05, 49K10, 49K15, 49K20

Article Outline

Keywords
Construction of a Dual Problem
Fenchel–Rockafellar Duality
Duality in the Sense of Klötzler
Bidual Problems, Generalized Flows, Relaxed Controls
Strong Duality Results
  Case A
  Case B
Sufficient Optimality Conditions
Duality and Maximum Principle
  Case A
  Case B
See also
References

Keywords

Control problems; First order partial differential equations; Duality theory; Necessary and sufficient optimality conditions

Consider the following optimal control problem with first order ordinary or partial differential equations:

$$\text{(P)} \quad \min J(x,u) = \int_{\Omega} r(t, x(t), u(t))\, dt$$

subject to functions $(x, u) \in W_p^{1,n}(\Omega) \times L_p^{\nu}(\Omega)$, fulfilling

- the *state equations*

$$x^i_{t_\alpha}(t) = g^i_\alpha(t, x(t), u(t)) \quad \text{a.e. on } \Omega,$$
$$(\alpha = 1, \ldots, m; \quad i = 1, \ldots, n);$$

- the *control restrictions*

$$u(t) \in U \subseteq \mathbb{R}^{\nu} \quad \text{a.e. on } \Omega;$$

- the *state constraints*

$$x(t) \in \overline{G(t)} \subseteq \mathbb{R}^n \quad \text{on } \overline{\Omega};$$

- and the *boundary conditions*

$$x(s) = \varphi(s) \quad \text{on } \partial\Omega.$$

The data of problem (P) satisfies the following hypothesis:

H1) For $m = 1$ we have $1 \le p \le \infty$, for $m \ge 2$ we have $m < p < \infty$.

H2) The sets Ω and

$$X := \{(t, \xi) \in \mathbb{R}^m \times \mathbb{R}^n : \ t \in \Omega,\ \xi \in G(t)\}$$

are strongly Lipschitz domains in the sense of C.B. Morrey and S. Hildebrandt [6]; the set U is closed.

H3) The functions r, r_ξ, g^i_α, $(g^i_\alpha)_\xi$, φ are continuous with respect to all arguments.

H4) The set of all admissible pairs (x, u), denoted by $\mathcal{Z}$, is nonempty.

The characterization of optimal solutions of special variational problems by dual or complementary problems has been well known in physics for a long time, e. g.,

- in elasticity theory, the principle of the minimum of potential energy (Dirichlet's principle) and the principle of tension (Castigliano's principle) are dual or complementary to each other.
- in the theory of electrostatic fields, the principle of the minimum of potential energy and the Thomson (Lord Kelvin) principle are dual problems.

A first systematic approach to duality for special problems in calculus of variations was given by K. Friedrichs ([4], 1928). In the 1950s and 1960s, this concept was extended by W. Fenchel [3], J.-J. Moreau, R.T. Rockafellar [19,20] and I. Ekeland and R. Temam [2] to larger classes of variational and control problems. Basing on Legendre transformation (or Fenchel conjugation), it was proved to be a suitable tool to handle convex problems.

Nonconvex problems (P) require an extended concept of duality. The construction of R. Klötzler, given in 1977 [7], can be regarded as a further development of Hamilton–Jacobi field theory.

Construction of a Dual Problem

In a very general setting, a problem (D) of maximization of an (extended real-valued) functional L over an arbitrary set $S \neq \emptyset$ is said to be a *dual problem* to (P) if the *weak duality relation*

$$\sup \text{(D)} \le \inf \text{(P)}$$

is satisfied.

The different notions of duality given in the introduction can be embedded into the following construction scheme:

1	The set of admissible pairs $(x, u) = z \in \mathcal{Z}$ is represented by the intersection of two suitable nonempty sets $\mathcal{Z}_0$ and $\mathcal{Z}_1$.
2	For an (extended real-valued) functional $\Phi : \mathcal{Z}_0 \times S_0 \to \overline{\mathbf{R}}$ the *equivalence relation* $$\inf_{z \in \mathcal{Z}} J(z) = \inf_{z \in \mathcal{Z}_0} \sup_{S \in S_0} \Phi(z, S),$$ holds.
3	Assuming $L_0(S) := \inf_{z \in \mathcal{Z}_0} \Phi(z, S)$, each problem $$\text{(D)} \quad \begin{cases} \max & L(S) \\ \text{s.t.} & S \in S_1 \subseteq S_0 \end{cases}$$ is a (weak) dual problem to (P) if $L(S) \le L_0(S)$ for all $S \in S_1$.

The proof of the *weak duality relation* results from the well-known inequality

$$\inf_{z\in\mathcal{Z}_0}\sup_{S\in S_0}\Phi(z,S)\geq\sup_{S\in S_0}\inf_{z\in\mathcal{Z}_0}\Phi(z,S).$$

Fenchel–Rockafellar Duality

In accordance with [2], we transform (P) into a general variational problem:

$$(\mathrm{V})\qquad\begin{cases}\min & \displaystyle\int_\Omega l(t,x(t),x_{t_\alpha}(t))\,dt\\ \text{s.t.} & x\in\mathcal{X}\end{cases}$$

where $l\colon\ \mathbb{R}^n\times\mathbb{R}^n\times\mathbb{R}^{nm}\to\overline{\mathbb{R}}$ is given by

$$l(t,\xi,w):=\begin{cases}\inf\{r(t,\xi,v):\\ \quad v\in U\text{ with}\\ \quad w=g(t,\xi,v)\}, & (t,\xi)\in\overline{X},\\ \infty & \text{else.}\end{cases}$$

and $\mathcal{X}=\{x\in W_p^{1,n}(\Omega):x(s)=\varphi(s)\text{ on }\partial\Omega\}$.

Then (P) is called *convex* if (V) is convex in the sense of [2, p. 113]. In this case both problems are equivalent [15]. The Fenchel-dual problem is obtained by the following settings in the above construction scheme:

1) $\mathcal{Z}_0=\{z=(x,u)\in W_p^{1,n}(\Omega)\times L_p^{1,\nu}(\Omega):$
$u(t)\in U\text{ a.e. on }\Omega,$
$x(t)\in\overline{G(t)}\text{ on }\Omega,$
$x(s)=\varphi(s)\text{ on }\partial\Omega\},$
$\mathcal{Z}_1=\{z=(x,u)\in W_p^{1,n}(\Omega)\times L_p^{1,\nu}(\Omega):$
$x^i_{t_\alpha}(t)=g^i_\alpha(t,x(t),u(t))\text{ a.e. on }\Omega,$
$\alpha=1,\dots,m;\ i=1,\dots,n\}.$

2) $\mathcal{S}_0=L_q^{n(1+m)}(\Omega)$ $(p^{-1}+q^{-1}=1)$, Φ is the classical *Lagrange functional*,

$$\Phi(z,S)=J(z)+\sum_{i,\alpha}\left\langle x^i_{t_\alpha}-g(\cdot,x,u),\ y_i^\alpha\right\rangle,$$

where $\langle\cdot,\cdot\rangle$ is the bilinear canonical pairing over $L_p(\Omega)\times L_p^*(\Omega)$, [15]. By use of the *Hamiltonian* of (P),

$$\mathcal{H}\colon\ \mathbb{R}^1\times\mathbb{R}^n\times\mathbb{R}^{nm}\to\overline{\mathbb{R}},$$
$$\mathcal{H}(t,\xi,\eta)=\sup\{H(t,\xi,v,\eta)\colon\ v\in U\}$$

with

$$H(t,\xi,v,\eta)=-r(t,\xi,v)+\sum_{i,\alpha}\eta_i^\alpha g_\alpha^i(t,\xi,v)$$

it can be formulated as follows [15]:

$$(\mathrm{D_R})_q\qquad\begin{cases}\max\\ \quad\Big[-\displaystyle\int_\Omega\Big(\sup_{\xi\in G(t)}[\mathcal{H}(t,\xi,-y(t))\\ \quad -y_0(t)^\top\xi]\Big)\,dt\\ \quad-\sup_{\zeta\in\mathcal{X}}\Big[\displaystyle\int_\Omega\big(y_0(t)^\top\zeta(t)\\ \quad+\sum_{i,\alpha}y_i^\alpha(t)\zeta^i_{t_\alpha}(t)\Big)\,dt\Big]\Big]\\ \text{s.t.}\quad(y_0,y)\in L_q^{n(1+m)}(\Omega).\end{cases}$$

Duality in the Sense of Klötzler

The duality in the sense of Klötzler is realized by the following settings in the general construction scheme [14]:

1) $\mathcal{Z}_0$ and $\mathcal{Z}_1$ are chosen as before.
2) $S_0=W_q^{1,n}(\overline{X})$, and Φ is an extended Lagrange functional,

$$\Phi(z,S)=J(z)+\sum_{i,\alpha}\left\langle x^i_{t_\alpha}-g^i_\alpha(\cdot,x,u),\,S^i_{\xi_\alpha}(\cdot,x)\right\rangle,$$

where $\langle\cdot,\cdot\rangle$ is again the bilinear canonical pairing over $L_p(\Omega)\times L_p^*(\Omega)$.

By use of Gauss' theorem, the dual problem reads as follows [8]:

$$(\mathrm{D_K})_q\qquad\begin{cases}\max & \displaystyle\int_{\partial\Omega}S(s,\zeta(s))\mathfrak{n}(s)\,do(s)\\ \text{s.t.} & S\in S_1,\end{cases}$$

where

$$S_1:=\left\{S\in S_0\colon\ \begin{array}{c}\sum_{\alpha=1}^m S^\alpha_{t_\alpha}(t,\xi)\\ +\mathcal{H}(t,\xi,S(t,\xi))\le 0\\ \text{a.e. on }\overline{X}\end{array}\right\},$$

$\mathfrak{n}(\cdot)$ is the exterior unit normal vector to $\partial\Omega$.

In this way we can characterize minimizers of (P) in terms of solutions of the *Hamilton–Jacobi inequality* or of the *Hamilton–Jacobi equation*. Since classical solutions of the latter equation may fail to exist, on the one hand techniques were developed to construct generalized solutions of this equation (viscosity solutions [11], generalized solutions involving the Clarke generalized gradient [1] or lower Dini derivatives [24]). On

the other hand, optimization techniques for parametric problems in finite-dimensional spaces are used to minimize the defect in the Hamilton–Jacobi inequality and to get sufficient conditions for (local) optimality [16,17,27,28,29].

Bidual Problems, Generalized Flows, Relaxed Controls

Duality allows to associate the bidual problem with a *flow problem* or a *relaxed control problem*. $(\mathrm{D_K})_q$, interpreted as an infinite linear programming problem, [13], has a dual problem again, which can be identified as a generalized flow problem in the sense of L.C. Young [26]: Assuming compactness of the control set U, one obtains the bidual problem $(\mathrm{D_K})_q^* = (\mathrm{F})$:

$$(\mathrm{F}) \quad \begin{cases} \min & \displaystyle\int_D r(t,\xi,v)\, dv(t,\xi,v) \\ \text{s.t.} & v \in \mathfrak{N}_D, \end{cases}$$

with

$$\mathfrak{N}_D := \Big\{ v \in \mathcal{R}_D : \int_{\partial\Omega} \sum_\alpha \psi^\alpha(s,\zeta(s))\mathfrak{n}_\alpha(s)\, do(s) = \int_D \left(\psi^\alpha_{t_\alpha}(t,\xi) + \sum_{i,\alpha} \psi^\alpha_{\xi_i}(t,\xi) g^i_\alpha(t,\xi,v) \right) dv(t,\xi,v),\ \forall \psi \in C^{1,m}(\overline{X}) \Big\},$$

where $\mathcal{R}_D$ is the set of all nonnegative Radon measures on $D := \overline{X} \times U$. $\mathfrak{N}_D$ contains special measures

$$dv(t,\xi,v) = d\delta_{x(t)}\, d\mu_t(v)\, dt,$$

with $\mu = \{\mu_t : t \in \Omega\} \in \mathfrak{M}_U$, $\mathfrak{M}_U$ is a *regular family of probability measures*, concentrated on U, [5], and δ_ξ are Dirac measures concentrated on the point $\xi \in \overline{G(t)}$.

Thus the relaxed control problem

$$\overline{(\mathrm{P})} \quad \begin{cases} \min & \displaystyle\int_\Omega \int_U r(t,x(t),v)\, d\mu_t(v)\, dt \\ \text{s.t.} & (x,\mu) \in W_p^{1,n}(\Omega) \times \mathfrak{M}_U, \end{cases}$$

satisfying $x(t) \in \overline{G(t)}$ and fulfilling the following variational equation for all $\psi \in C^{1,m}(\overline{X})$,

$$\int_{\partial\Omega} \sum_\alpha \psi^\alpha(s,\zeta(s))\mathfrak{n}_\alpha(s)\, do(s) = \int_\Omega \left[\sum_\alpha \psi^\alpha_{t_\alpha}(t,x(t)) + \sum_{i,\alpha} \psi^\alpha_{\xi_i}(t,x(t)) \int_U g^i_\alpha(t,x(t),v)\, d\mu_t(v) \right] dt$$

has the embedding (F), and

$$\sup(\mathrm{D_K})_\infty = \inf(\mathrm{F}) \le \inf \overline{(\mathrm{P})}$$

holds, [13].

Strong Duality Results

The property of strong duality between (P) and (D) is defined by the equation

$$\sup(\mathrm{D}) = \inf(\mathrm{P}),$$

and this common value is called *settle-value* of Φ.

Case A

Control problems with single integrals and ordinary differential equations, $\Omega = [0, T]$.

For convex problems (P),

$$\sup(\mathrm{D_R})_q = \min(\mathrm{P})$$

holds, [2]. Moreover, in $(\mathrm{D_R})_q$ the optimal solution (y_0^*, y^*) exists and fulfills

$$y^* \in W_p^{1,n}(\Omega)$$

and

$$\frac{d}{dt} y^*(t) = y_0^*(t) \text{ a.e. on } (0,T),$$

[15]. Both dual problems, $(\mathrm{D_R})_q$ and $(\mathrm{D_K})_q$, coincide, if in $(\mathrm{D_K})_q$ a linear setting

$$S(t,\xi) = a(t) + \xi^\top y(t)$$

is chosen, [14].

For nonconvex control problems with compact U is was shown by different techniques, that (P) as well as

$\overline{(\mathrm{P})}$ possess a same dual problem, [12,22,23], and strong duality

$$\sup\,(\mathrm{D_K})_\infty = \min\,\overline{(\mathrm{P})}$$

holds. The variational equation appearing in $\overline{(\mathrm{P})}$ is in this case equivalent to the *generalized state equations*

$$\frac{d}{dt}x^i(t) = \int_U g^i(t, x(t), v)\, d\mu_t(v) \text{ a.e. on } (0, T)$$

with the *boundary conditions*

$$x(s) = \varphi(s) \quad \text{for } s = 0,\ s = T.$$

The question of existence of a solution of the dual problem $(\mathrm{D_K})_\infty$ was discussed in [21].

Case B

Control problems with multiple integrals and first order partial differential equations.

As before, for convex problems

$$\sup\,(\mathrm{D_R})_q = \min\,(\mathrm{P})$$

holds. The equivalence of $(\mathrm{D_R})_q$ and $(\mathrm{D_K})_q$ is lost in general. Results concerning strong duality between (D) and $\overline{(\mathrm{P})}$ in the nonconvex case are largely missing.

Sufficient Optimality Conditions

First- and second order sufficient optimality conditions for global minimizers can be derived by means of duality. In the general concept, $(x^*,\, u^*) \in \mathcal{Z}$ is a global minimizer of (P) if

$$\begin{aligned} J(x^*, u^*) &= \inf_{z\in\mathcal{Z}_0} \sup_{S\in\mathcal{S}_0} \Phi(z, S) \\ &= \max_{S\in\mathcal{S}_0} \inf_{z\in\mathcal{Z}_0} \Phi(z, S) \end{aligned}$$

and it exists an $S^* \in \mathcal{S}_1$ with

$$L_0(S^*) = \max_{S\in\mathcal{S}_1} L(S) = \max_{S\in\mathcal{S}_0} L_0(S).$$

Following the concept of Klötzler, these equations are satisfied if and only if for $S^* \in W^{1,n}_\infty(\overline{X})$ the following conditions are fulfilled:

a) the *Hamilton–Jacobi inequality*

$$\begin{aligned} \Lambda(t, \xi) := &\sum_\alpha S^{*\alpha}_{t_\alpha}(t, \xi) \\ &+ \mathcal{H}(t, \xi, S^*_\xi(t, \xi)) \le 0 \text{ on } \overline{X}; \end{aligned}$$

b) the *Hamilton–Jacobi equation*

$$\begin{aligned} &\sum_\alpha S^{*\alpha}_{t_\alpha}(t, x^*(t)) \\ &\quad + \mathcal{H}(t, x^*(t), S^*_\xi(t, x^*(t))) = 0 \text{ on } \overline{\Omega}; \end{aligned}$$

c) the *maximum condition*

$$\begin{aligned} &\mathcal{H}(t, x^*(t), S^*_\xi(t, x^*(t))) \\ &\quad = \mathcal{H}(t, x^*(t), u^*(t), S^*_\xi(t, x^*(t))) \text{ a.e. on } \Omega. \end{aligned}$$

From conditions a) and b) follows that $x^*(t)$ must be a global minimizer of the parametric optimization problem

$$(\mathrm{P})_t \quad \begin{cases} \max & \Lambda(t, \xi) \\ \text{s.t.} & \xi \in \overline{G(t)} \end{cases}$$

with parameter $t \in \overline{\Omega}$. For this last problem $(\mathrm{P})_t$ first- and second order sufficient optimality conditions can be derived with the quadratic setting

$$\begin{aligned} S^{*\alpha}(t, \xi) &= a^\alpha(t) + y^{\alpha\top}(t)(\xi - x(t)) \\ &\quad + \frac{1}{2}(\xi - x^*(t))Q^\alpha(t)(\xi - x^*(t)) \end{aligned}$$

in the dual problem $(\mathrm{D_K})_\infty$, where $y^\alpha \in W^{1,n}_\infty(\Omega)$ and $Q^\alpha \in W^{1,nn}_\infty(\Omega)$ symmetric.

The ideas, mentioned above, can be used for identifying *strong local minimizers* of (P) too. In this case $\overline{X}$ is to be replaced by

$$\begin{aligned} \overline{X}_\varepsilon = &\overline{X} \\ &\cap \left\{(t, \xi) \in \mathbb{R}^{n+1}:\ \|\xi - x(t)\| < \varepsilon,\ t \in \overline{\Omega}\right\}, \end{aligned}$$

[16,17,27,28,29]. The second order condition for $(\mathrm{P})_t$ yields a definiteness condition for a Riccati-type expression which generalizes the known theory of conjugated points in the calculus of variations in one independent variable.

Duality and Maximum Principle

Case A

Control problems with single integrals, $\Omega = [0, T]$. For convex problems (P) it can be shown that the *Pontryagin maximum principle* is not only a necessary but also

a sufficient optimality condition. In this case the canonical variables in the Maximum principle solve at the same time the dual problem $(D_K)_\infty$, [15].

Case B

Control problems with multiple integrals, $\Omega \subseteq \mathbf{R}^m$, $m \geq 2$. For convex problems (P) or relaxed problems $(\overline{P})$ a maximum principle was proved in the beginning of the 1990s, [10,18,25]. It turns out, that the canonical variables in this principle are not necessarily functions but contents or measures from $L^*_\infty(\Omega)$ or $C^*(\Omega)$. A corresponding duality theory with dual variables in these measure spaces was developed by Klötzler [9] and strong duality was shown. As before, in the convex case the canonical variables of the maximum principle solve the dual problem.

See also

- ▶ Control Vector Iteration
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Optimal Control of a Flexible Arm
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-Infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Clarke FH, Vinter RB (1983) Local optimality conditions and Lipschitz solutions to the Hamilton–Jacobi equation. SIAM J Control Optim 21:856–870
2. Ekeland I, Temam R (1976) Convex analysis and variational problems. North-Holland, Amsterdam
3. Fenchel W (1953) Convex cones, sets and functions. Lecture Notes Dept Math. Princeton Univ. Press, Princeton
4. Friedrichs K (1929) Ein Verfahren der Variationsrechnung, das Minimum eines Integrals als das Maximum eines anderen Ausdrucks darzustellen. Göttinger Nachrichten 13–20
5. Gamkrelidze RV (1978) Principles of optimal control theory. Plenum, New York
6. Hildebrandt S (1962) Über die Identität der Sobolewschen und Calkin-Morreyschen Räume. Math Ann 148:226–237
7. Klötzler R (1977) On a general conception of duality in optimal control. Proc Equadiff 4:189–196
8. Klötzler R (1980) Starke Dualität in der Steuerungstheorie. Math Nachrichten 95:253–263
9. Klötzler R (1995) Optimal transportation flows. J Anal Appl 14:391–401
10. Klötzler R, Pickenhain S (1993) Pontryagin's maximum principle for multidimensional control problems. Internat Ser Numer Math, vol 111. Birkhäuser, Basel pp 21–30
11. Lions P-L (1982) Generalized solutions of Hamilton-Jacobi equations. Res Notes Math, vol 69. Pitman, Boston, MA
12. Pickenhain S (1982) Starke Dualität bei verallgemeinerten Steuerungsproblemen. Z Anal Anwend 1(4):15–24
13. Pickenhain S (1987) Dualität bei Steuerungsproblemen mehrfacher Integrale. Z Anal Anwend 6(2):159–174
14. Pickenhain S (1988) Zum Vergleich verschiedener Dualitätsbegriffe aus der Sicht der Steuerungstheorie. Z Anal Anwend 7(3):277–285
15. Pickenhain S (1992) Beiträge zur Theorie mehrdimensionaler Steuerungsprobleme. Habilitationsschrift Univ. Leipzig
16. Pickenhain S (1992) Sufficiency conditions for weak local minima in multidimensional optimal control problems with mixed control-state restrictions. Z Anal Anwend 11(4):559–568
17. Pickenhain S, Tammer K (1991) Sufficient conditions for local optimality in multidimensional control problems with state restrictions. Z Anal Anwend 10(3):397–405
18. Pickenhain S, Wagner M (1999) Critical points in relaxed deposit problems. In: Ioffe A, Reich S, Shafrir I (eds) Calculus of Variations and Optimal Control, Technion '98. Pitman Res Notes Math Chapman and Hall/CRC, London/Boca Raton, FL, pp 217–236
19. Rockafellar RT (1970) Conjugate convex functions in optimal control and the calculus of variations. J Math Anal Appl 32:174–222
20. Rockafellar RT (1974) Conjugate duality and optimization. Regional Conf Ser, vol 16. SIAM, Philadelphia
21. Vinter RB (1983) Weakest conditions for existence of Lipschitz continous Krotov functions in optimal control theory. SIAM J Control Optim 21(2):215–234
22. Vinter RB, Levis RM (1978) The equivalence of the strong and weak formulation for certain problems in optimal control. SIAM J Control Optim 16(4):546–570

23. Vinter RB, Levis RM (1978) A nessesary and sufficient condition for optimality of dynamic programming type, making no a priori assumptions on the control. SIAM J Control Optim 16(4):571–583
24. Vinter RB, Wolenski P (1990) Hamilton-Jacobi theory for problems with data measurable in time. SIAM J Control Optim 28:1404–1419
25. Wagner M (1999) Pontryagin's maximum principle for Dieudonné-Rashevsky type problems involving Lipschitz functions. Optim:165–184 46
26. Young LC (1968) Calculus of variations and optimal control theory. W.B. Saunders, Philadelphia
27. Zeidan V (1983) Sufficient conditions for the generalized problem of Bolza. Trans Amer Math Soc 275:561–586
28. Zeidan V (1984) Extendend Jacobi sufficiency criterion for optimal control. SIAM J Control Optim 22:294–301
29. Zeidan V (1984) First- and second-order sufficient conditions for optimal control and the Calculus of Variations. Appl Math Optim 11:209–226

Duality for Semidefinite Programming

HENRY WOLKOWICZ
Department Combinatorics and Optimization, University Waterloo, Waterloo, Canada

MSC2000: 90C30

Article Outline

Keywords
Synonyms
Basic Properties
Strong Duality
Strict Complementarity
Closing the Duality Gap
Extensions
See also
References

Keywords

Semidefinite programming; Convex programming; Lagrangian duality; Strong and weak duality; Constraint qualification; Complementarity

Synonyms

SDP duality

Basic Properties

Consider the primal *semidefinite program*

$$\text{SDP} \quad \mu^* := \begin{cases} \min & C \bullet X \\ \text{s.t.} & AX = b \\ & X \succeq 0, \end{cases}$$

where $C \bullet X = \operatorname{trace} CX$ denotes the inner product of the symmetric matrices C, X; $X \succeq Y$ denotes that the symmetric matrix $X - Y$ is positive semidefinite; and $\mathcal{A}: \mathcal{S}^n \to \mathbf{R}^m$ is a linear operator on the space of symmetric matrices, with adjoint $\mathcal{A}^*$. Equivalently, the linear constraint can be written using symmetric matrices A_i, $i = 1, \ldots, m$, as

$$\mathcal{A}_i \bullet X = b_i \quad \text{for all } i = 1, \ldots, m;$$

while the adjoint operation on $y \in \mathbf{R}^m$ is

$$\mathcal{A}^* y := \sum_{i=1}^{m} y_i A_i.$$

The *Lagrangian* function is

$$\mathcal{L}(X, y) := C \bullet X + y^\top (b - \mathcal{A}X).$$

The primal problem is equivalent to

$$\mu^* = \min_{X \succeq 0} \max_{y} \mathcal{L}(X, y) = C \bullet X + y^\top (b - \mathcal{A}X).$$

The equivalence can be seen by using the *hidden constraint* in the outer minimization problem $b - \mathcal{A}X = 0$, i. e. if this constraint does not hold then the inner maximum value is $+\infty$.

By interchanging the maximum and minimum and rewriting the order of terms in the Lagrangian, we get the dual problem and *weak duality*:

$$\mu^* \geq \nu^* := \max_{y} \min_{X \succeq 0} b^\top y + (C - \mathcal{A}^* y) \bullet X.$$

Using the hidden constraint in the outer maximization problem $C - \mathcal{A}^* y \succeq 0$, this becomes equivalent to

$$\text{(D)} \quad \nu^* = \begin{cases} \max & b^\top y \\ \text{s.t.} & \mathcal{A}^* y \preceq C. \end{cases}$$

The dual pair SDP and (D) look very much like a dual pair of linear programs (denoted LP) where the

adjoint operator replaces the transpose and positive semidefiniteness of matrix variables replaces nonnegativity of vector variables. In fact, duality theory for SDP has a lot of similarities with that of LP: *weak duality* $\mu^* \geq \nu^*$ follows from the interchange of maximum and minimum; from this we get that unboundedness of SDP (respectively, (D)) implies infeasibility of (D) (respectively, (D)).

Weak duality illustrates one of the powerful uses of the dual program, i. e. it provides lower bounds on the optimal value of the primal program.

Other formulations of SDP provide similar duals. In fact, SDP is a special case of cone programming. Let K, L be two *convex cones*, i. e. K (and L) satisfy: the Minkowski sum $K + K \subset K$ and $\alpha\, K \subset K$ for all $\alpha \in \mathbf{R}$. Define the primal cone program as

$$\text{(PC)} \quad \nu^* = \begin{cases} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}X \succeq_K b \\ & X \succeq_L 0, \end{cases}$$

where $X \succeq_L Y$ denotes $X - Y \in L$ (and similarly for K), and $\langle \cdot, \cdot \rangle$ denotes the appropriate inner product. Then the above min-max argument yields the dual cone program

$$\text{(DC)} \quad \nu^* = \begin{cases} \max & \langle b, y\rangle \\ \text{s.t.} & \mathcal{A}^* y \preceq_{L^+} C \\ & y \succeq_{K^+} 0, \end{cases}$$

where $\cdot^+$ denotes taking the *polar cone*.

It is an interesting exercise to see that this elegant dual formulation works for linear programs that have mixtures of inequality and equality constraints with mixtures of free and nonnegative variables.

Strong Duality

However, unlike linear programming, *strong duality* for SDP needs a constraint qualification, e. g. strict primal feasibility (called *Slater's condition*),

there exists a $\widehat{X} \succ 0$ with $\mathcal{A}\widehat{X} = b$.

This constraint qualification implies strong duality holds, i. e. that $\mu^* = \nu^*$ and ν^* is attained. Conversely,

$$\mathcal{A}^* y \prec_{L^+} C$$

also implies that $\mu^* = \nu^*$ but with μ^* attained. If Slater's condition does not hold, then a duality gap $\mu^* > \nu^*$ can exist, and/or the dual (or primal) optimal value may not be attained, see e. g. [10].

Example 1 If the dual is

$$\text{(D)} \quad \nu^* = \begin{cases} \sup & x_2 \\ \text{s.t.} & \begin{pmatrix} x_2 & 0 & 0 \\ 0 & x_1 & x_2 \\ 0 & x_2 & 0 \end{pmatrix} \preceq \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \end{cases}$$

then the primal is

$$\text{(P)} \quad \mu^* = \begin{cases} \inf & U_{11} \\ \text{s.t.} & U_{22} = 0 \\ & U_{11} + 2U_{23} = 1 \\ & U \succeq 0 \end{cases}$$

and we have the duality gap

$$\nu^* = 0 < \mu^* = 1.$$

If strong duality holds, we get the following primal-dual characterization of optimality for the dual pair X, y, with $X \succeq 0$:

- $\mathcal{A}\, X = b$ (primal feasibility);
- $\mathcal{A}^* y \preceq C$ (dual feasibility);
- $X\,(\mathcal{A}^* y - C) = 0$ (complementary slackness).

These optimality conditions provide the basis for:

i) the primal simplex method (maintain primal feasibility and complementary slackness while striving for dual feasibility);
ii) the dual simplex method (maintain dual feasibility and complementary slackness while striving for primal feasibility); and
iii) the interior point methods (maintain primal and dual feasibility while striving for complementary slackness).

Unlike the LP case, there are currently no efficient algorithms for primal or dual simplex methods for SDP; however, interior point methods have proven to be very successful. Thus we see the importance of duality for both theoretical and algorithmic purposes.

Strict Complementarity

Another example of the difference between LP and SDP arises in the complementary slackness conditions. If an optimal pair X, y exist, then in the LP case there also exists an optimal pair that satisfies strict complementarity, i. e.

$$X + (C - \mathcal{A}^* y) \succ 0,$$

where in the LP case this is a sum of nonnegative diagonal matrices, see [4,5]. However, in the SDP case, there may not exist such a strict complementary optimal pair, though the existence is generic, see [8].

Closing the Duality Gap

Both the strict complementarity and strict feasiblity, or Slater's constraint qualification, are generic, see [8]. But there are classes of problems where strong duality fails, e. g. relaxations that arise from hard combinatorial problems, e. g. [13].

One can regularize semidefinite programs and guarantee that Slater's constraint qualification holds, e. g. [2,3,12]. This involves finding the minimal face of the semidefinite cone that contains the feasible set, i. e. the so-called *minimal cone*. A numerical procedure for regularization is presented in [3]. However, this process is not computationally tractable. An equivalent approach is the extended Lagrange–Slater dual program of M. Ramana [9,10]. This provides a means of writing down a regularized program that is of polynomial size. Thus strong duality can be attained theoretically using the above techniques and exploiting the structure of specific problems. However, lack of regularity (Slater's condition) is an indication of an ill-posed problem. Thus, the question of whether regularization can be done computationally for general problems is still an open question, see e. g. [7].

Extensions

The SDPs considered above have all contained linear objectives and constraints. There is no reason to restrict SDPs to this special case. Duality for general cone programs with possible nonlinear objectives and constraints is considered in [2,3,11]. Applications for quadratic objectives SDP appear in, e. g., [1,6].

See also

- ▶ Interior Point Methods for Semidefinite Programming
- ▶ Semidefinite Programming and Determinant Maximization
- ▶ Semidefinite Programming: Optimality Conditions and Stability
- ▶ Semidefinite Programming and Structural Optimization
- ▶ Semi-infinite Programming, Semidefinite Programming and Perfect Duality
- ▶ Solving Large Scale and Sparse Semidefinite Programs

References

1. Alfakih A, Khandani A, Wolkowicz H (1997) Solving Euclidean distance matrix completion problems via semidefinite programming. Techn Report CORR Univ Waterloo 98-9, in Computational Optim and Appl
2. Borwein JM, Wolkowicz H (1981) Characterizations of optimality for the abstract convex program with finite dimensional range. J Austral Math Soc (Ser A) 30:390–411
3. Borwein JM, Wolkowicz H (1981) Regularizing the abstract convex program. J Math Anal Appl 83:495–530
4. Goldman AJ, Tucker AW (1956) Polyhedral convex cones. Linear equalities and related systems. In: Ann of Math Stud, vol 38. Princeton Univ. Press, Princeton, pp 19–40
5. Goldman AJ, Tucker AW (1956) Theory of linear programming. Linear inequalities and related systems. In: Ann of math Stud, vol 38. Princeton Univ. Press, Princeton, pp 53–97
6. Johnson CR, Kroschel B, Wolkowicz H (1998) An interior-point method for approximate positive semidefinite completions. Comput Optim Appl 9(2):175–190
7. Klerk Ede (1997) Interior point methods for semidefinite programming. PhD Thesis Delft Univ
8. Pataki G, Tuncel L (1997) On the generic properties of convex optimization problems in conic form. Techn Report CORR Dept Combinatorics and Optim Waterloo, Ont 97-16
9. Ramana MV (1993) An algorithmic analysis of multiquadratic and semidefinite programming problems. PhD Thesis Johns Hopkins Univ, Baltimore, Md
10. Ramana M, Tuncel L, Wolkowicz H (1997) Strong duality for semidefinite programming. SIAM J Optim 7(3):641–662
11. Shapiro A (1997) First and second order analysis of nonlinear semidefinite programs. Math Program 77:301–320
12. Wolkowicz H (1981) Some applications of optimization in matrix theory. Linear Alg Appl 40:101–118
13. Zhao Q, Karisch SE, Rendl F, Wolkowicz H (1998) Semidefinite programming relaxations for the quadratic assignment problem. J Combin Optim 2:71–109

Duality Theory: Biduality in Nonconvex Optimization

David Yang Gao
Virginia Polytechnic Institute and State University, Blacksburg, USA

MSC2000: 49-XX, 90-XX, 93-XX

Article Outline

Keywords
SuperLagrangian Duality
Nonconvex Primal and Dual Problems
D.C. Programming and Hamiltonian
See also
References

Keywords

Duality; Nonconvex optimization; d.c. programming; SuperLagrangian; Biduality; Clarke duality; Hamiltonian system

It is known that in convex optimization, the Lagrangian associated with a constrained problem is usually a saddle function, which leads to the classical *saddle Lagrange duality* (i. e. the *monoduality*) theory. In nonconvex optimization, a so-called superLagrangian was introduced in [1], which leads to a nice biduality theory in convex *Hamiltonian systems* and in the so-called *d.c. programming*.

SuperLagrangian Duality

Definition 1 Let $L(x, y^*)$ be an arbitrary given real-valued function on $\mathcal{X} \times \mathcal{Y}^*$.

A function $L: \mathcal{X} \times \mathcal{Y}^* \to \mathbf{R}$ is said to be a *supercritical function* (or a ∂^+*-function*) on $\mathcal{X} \times \mathcal{Y}^*$ if it is concave in each of its arguments.

A function $L: \mathcal{X} \times \mathcal{Y}^* \to \mathbf{R}$ is said to be a *subcritical function* (or a ∂^-*-function*) on $\mathcal{X} \times \mathcal{Y}^*$ if $-L$ is a supercritical function on $\mathcal{X} \times \mathcal{Y}^*$.

A point $(\overline{x}, \overline{y}^*)$ is said to be a *supercritical point* (or a ∂^+*-critical point*) of L on $\mathcal{X} \times \mathcal{Y}^*$ if

$$L(\overline{x}, y^*) \le L(\overline{x}, \overline{y}^*) \ge L(x, \overline{y}^*) \tag{1}$$

holds for all $(x, y^*) \in \mathcal{X} \times \mathcal{Y}^*$.

A point $(\overline{x}, \overline{y}^*)$ is said to be a *subcritical point* (or a ∂^-*-critical point*) of L on $\mathcal{X} \times \mathcal{Y}^*$ if

$$L(\overline{x}, y^*) \ge L(\overline{x}, \overline{y}^*) \le L(x, \overline{y}^*) \tag{2}$$

holds for all $(x, y^*) \in \mathcal{X} \times \mathcal{Y}^*$.

Clearly, a point $(\overline{x}, \overline{y}^*)$ is a subcritical point of L on $\mathcal{X} \times \mathcal{Y}^*$ if and only if it is a supercritical point of $-L$ on $\mathcal{X} \times \mathcal{Y}^*$. A supercritical function $L(x, y^*)$ is called the *superLagrangian* if it is a *Lagrange form* associated with a constrained optimization problem. $L(x, y^*)$ is called the *subLagrangian* if $-L(x, y^*)$ is a superLagrangian.

For example, the quadratic function

$$L(x, y) = axy - \frac{1}{2}bx^2 - \frac{1}{2}cy^2, \quad b, c > 0,$$

is concave for each x and y, and hence is a supercritical point function on $\mathbf{R} \times \mathbf{R}$. But $L(x, y)$ is not concave on the vector (x, y) since the Hessian matrix of L

$$D^2L(x, y) = \begin{pmatrix} -b & a \\ a & -c \end{pmatrix}$$

is not necessarily to be negative-definite for any $a \in \mathbf{R}$ and $b, c > 0$. L is a subcritical function if $b, c < 0$. But L may not be convex on (x, y) for the same reason.

Since L is a subLagrangian if and only if $-L$ is a superLagrangian, here we only consider the duality theory for the superLagrangian.

Theorem 2 (*Supercritical point*) *Let $L(x, y^*)$ be an arbitrary given function, partially Gâteaux differentiable on an open subset $\mathcal{X}_a \times \mathcal{Y}_a^* \subset \mathcal{X} \times \mathcal{Y}^*$. If $(\overline{x}, \overline{y}^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$ is either a supercritical or subcritical point of L, then $(\overline{x}, \overline{y}^*)$ is a critical point of L on $\mathcal{X}_a \times \mathcal{Y}_a^*$.*

Any critical point of a Gâteaux differentiable superLagrangian is a supercritical point. However, if $(\overline{x}, \overline{y}^*)$ is a supercritical point of L, it does not follows that L is a superLagrangian. In the d.c. programming or variational analysis of convex Hamiltonian systems, the following statements are of important theoretical value.

S1) Under certain necessary and sufficient conditions we have

$$\inf_{x \in \mathcal{X}_a} \sup_{y^* \in \mathcal{Y}_a^*} L(x, y^*) = \inf_{y^* \in \mathcal{Y}_a^*} \sup_{x \in \mathcal{X}_a} L(x, y^*). \tag{3}$$

A statement of this type is called a *superminimax theorem* and the pair $(\overline{x}, \overline{y}^*)$ is called a *superminimax point* of L on $\mathcal{X}_a \times \mathcal{Y}_a^*$.

S2) Under certain conditions, a pair $(\overline{x}, \overline{y}^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$ exists such that

$$L(x, \overline{y}^*) \le L(\overline{x}, \overline{y}^*) \ge L(\overline{x}, y^*) \tag{4}$$

holds for all $(x, y^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$. A statement of this type is called a *supercritical point theorem*.

By the fact that the maxima of $L(x, y^*)$ can be taken in either order on $\mathcal{X}_a \times \mathcal{Y}_a^*$, the equality

$$\sup_{x\in\mathcal{X}_a}\sup_{y^*\in\mathcal{Y}_a^*} L(x, y^*) = \sup_{y^*\in\mathcal{Y}_a^*}\sup_{x\in\mathcal{X}_a} L(x, y^*) \tag{5}$$

always holds. A pair $(\overline{x}, \overline{y}^*)$ which maximizes L on $\mathcal{X}_a \times \mathcal{Y}_a^*$ is called a *supermaximum point* of L on $\mathcal{X}_a \times \mathcal{Y}_a^*$.

For a given superLagrangian L: $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$, we let $\mathcal{X}_k \subseteq \mathcal{X}_a$ and $\mathcal{Y}_s^* \subseteq \mathcal{Y}_a^*$ be such that

$$\sup_{y^*\in\mathcal{Y}_a^*} L(x, y^*) < +\infty, \quad \forall x \in \mathcal{X}_k,$$

$$\sup_{x\in\mathcal{X}_a} L(x, y^*) < +\infty, \quad \forall y^* \in \mathcal{Y}_s^*.$$

Theorem 3 (*superLagrangian duality*) *Let the Lagrangian L: $\mathcal{X} \times \mathcal{Y}^* \to \mathbf{R}$ be an arbitrary given function. If there exists either a supermaximum point $(\overline{x}, \overline{y}^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$ such that*

$$L(\overline{x}, \overline{y}^*) = \max_{x\in\mathcal{X}_a}\max_{y^*\in\mathcal{Y}_a^*} L(x, y^*) = \max_{y^*\in\mathcal{Y}_a^*}\max_{x\in\mathcal{X}_a} L(x, y^*), \tag{6}$$

or a superminimax point $(\overline{x}, \overline{y}^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$ such that

$$L(\overline{x}, \overline{y}^*) = \min_{x\in\mathcal{X}_a}\max_{y^*\in\mathcal{Y}_a^*} L(x, y^*) = \min_{y^*\in\mathcal{Y}_a^*}\max_{x\in\mathcal{X}_a} L(x, y^*), \tag{7}$$

then $(\overline{x}, \overline{y}^*)$ is a supercritical point of L on $\mathcal{X}_a \times \mathcal{Y}_a^*$.

Conversely, if L is partially Gâteaux differentiable on an open subset $\mathcal{X}_a \times \mathcal{Y}_a^* \subset \mathcal{X} \times \mathcal{Y}^*$, and $(\overline{x}, \overline{y}^*)$ is a supercritical point of L on $\in \mathcal{X}_a \times \mathcal{Y}_a^*$, then either the supermaximum theorem in the form

$$L(\overline{x}, \overline{y}^*) = \max_{x\in\mathcal{X}_k}\max_{p\in\mathcal{Y}_a^*} L(x, y^*) = \max_{y^*\in\mathcal{Y}_s^*}\max_{x\in\mathcal{X}_a} L(x, y^*), \tag{8}$$

holds, or the superminimax theorem in the form

$$L(\overline{x}, \overline{y}^*) = \min_{x\in\mathcal{X}_k}\max_{y^*\in\mathcal{Y}_a^*} L(x, y^*) = \min_{y^*\in\mathcal{Y}_s^*}\max_{x\in\mathcal{X}_a} L(x, y^*) \tag{9}$$

holds.

This superLagrangian duality theorem shows a very important fact in Hamiltonian systems, i. e. the critical points of the Lagrangian L either maximize or minimaximize L on $\mathcal{X}_k \times \mathcal{Y}_s^*$ in either order.

Nonconvex Primal and Dual Problems

Let L: $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ be an arbitrary given supercritical function. For any fixed $x \in \mathcal{X}_a$, let

$$P(x) = \sup_{y^*\in\mathcal{Y}_a^*} L(x, y^*). \tag{10}$$

Clearly, the function $P(x)$ need not be either convex or concave. Let $\mathcal{X}_k \subset \mathcal{X}_a$ be the *primal feasible set* such that P: $\mathcal{X}_k \to \mathbf{R}$ is finite and Gâteaux differentiable. Then for a nonconvex function P, two primal problems can be proposed as:

$$(\mathcal{P}_{\inf}): \quad P(x) \to \min, \quad \forall u \in \mathcal{X}_k, \tag{11}$$

$$(\mathcal{P}_{\sup}): \quad P(x) \to \max, \quad \forall u \in \mathcal{X}_k. \tag{12}$$

The problems $(\mathcal{P}_{\inf})$ and $(\mathcal{P}_{\sup})$ are realisable if the primal feasible set $\mathcal{X}_k$ is not empty.

Dually, for any fixed $y^* \in \mathcal{Y}_a^*$, let

$$P^d(y^*) = \sup_{x\in\mathcal{X}_a} L(x, y^*) \tag{13}$$

with the *dual feasible set* $\mathcal{Y}_s^* \subset \mathcal{Y}_a^*$ such that P^d: $\mathcal{Y}_s^* \to \mathbf{R}$ is finite and Gâteaux differentiable. The two nonconvex dual problems are:

$$(\mathcal{P}_{\inf}^d): \quad P^d(y^*) \to \min, \quad \forall y^* \in \mathcal{Y}_s^*, \tag{14}$$

$$(\mathcal{P}_{\sup}^d): \quad P^d(y^*) \to \max, \quad \forall y^* \in \mathcal{Y}_s^*. \tag{15}$$

These two dual problems are realisable if the dual feasible set $\mathcal{Y}_s^*$ is not empty.

Theorem 4 (*Biduality theorem*) *Let L: $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ be a given arbitrary function such that P and P^d are well-defined by (10) and (13) on the open subsets $\mathcal{X}_k$ and $\mathcal{Y}_s^*$, respectively.*

1) *If* $(\overline{x}, \overline{y}^*)$ *is a supercritical point of* L *on* $\mathcal{X}_k \times \mathcal{Y}_s^*$, *then* $DP(\overline{x}) = 0$, $DP^d(\overline{y}^*) = 0$, *and*

$$P(\overline{x}) = L(\overline{x}, \overline{y}^*) = P^d(\overline{y}^*). \tag{16}$$

2) *If* $(\overline{x}, \overline{y}^*)$ *is a supercritical point of* L *on* $\mathcal{X}_k \times \mathcal{Y}_s^*$, *then* $\overline{x}$ *is a minimizer of* P *on* $\mathcal{X}_k$ *if and only if* $\overline{y}^*$ *is a minimizer of* P^d *on* $\mathcal{Y}_s^*$, *i. e. the double-min duality*

$$P(\overline{x}) = \inf_{x \in \mathcal{X}_k} P(x) \Leftrightarrow \inf_{y^* \in \mathcal{Y}_s^*} P^d(y^*) = P^d(\overline{y}^*) \tag{17}$$

holds.

3) *If* $(\overline{x}, \overline{y}^*)$ *is a supercritical point of* L *on* $\mathcal{X}_k \times \mathcal{Y}_s^*$, *then* $\overline{x}$ *is a maximizer of* P *on* $\mathcal{X}_k$ *if and only if* $\overline{y}^*$ *is a maximizer of* P^d *on* $\mathcal{Y}_s^*$, *i. e. the double-max duality*

$$P(\overline{x}) = \sup_{x \in \mathcal{X}_k} P(x) \Leftrightarrow \sup_{y^* \in \mathcal{Y}_s^*} P^d(y^*) = P^d(\overline{y}^*) \tag{18}$$

holds.

D.C. Programming and Hamiltonian

In d.c. programming, the primal function $P\colon \mathcal{X}_k \to \mathbf{R}$ can be written as

$$P(x) = W(\Lambda x) - F(x),$$

where $\Lambda\colon \mathcal{X} \to \mathcal{Y}$ is a linear operator, $W\colon \mathcal{Y}_a \to \mathbf{R}$ and $F\colon \mathcal{X}_a \to \mathbf{R}$ are two convex, Gâteaux differentiable real-valued functions, satisfying the *Legendre duality relations*

$$x^* = DF(x) \Leftrightarrow x = DF^*(x^*)$$
$$\Leftrightarrow \langle x, x^* \rangle = F(x) + F^*(x^*)$$

on $\mathcal{X}_a \times \mathcal{X}_a^*$, and

$$y^* = DW(y) \Leftrightarrow y = DW^*(y^*)$$
$$\Leftrightarrow \langle y; y^* \rangle = W(y) + W^*(y^*)$$

on $\mathcal{Y}_a \times \mathcal{Y}_a^*$, where $F^*\colon \mathcal{X}_a^* \to \mathbf{R}$ and $W^*\colon \mathcal{Y}_a^* \to \mathbf{R}$ are the *Legendre conjugates* of F and W, respectively.

In dynamical systems, if $\Lambda = d/dt$ is a differential operator, its adjoint associated with the standard bilinear forms in $\mathcal{L}^2$ is $\Lambda^* = -d/dt$. If W denotes the kinetic energy, F stands for potential energy, then the primal function $P(x) = W(\Lambda x) - F(x)$ is the *total action* of the system. The primal feasible set $\mathcal{X}_k \subset \mathcal{X}$, defined by

$$\mathcal{X}_k = \{x \in \mathcal{X}_a \colon \Lambda x \in \mathcal{Y}_a\},$$

is called the *kinetically admissible space*. Clearly, $P\colon \mathcal{X}_k \to \mathbf{R}$ is nonconvex.

The *Lagrangian form* associated with the nonconvex primal problems is defined by

$$L(x, y^*) = \langle \Lambda x; y^* \rangle - W^*(y^*) - F(x), \tag{19}$$

which is G âteaux differentiable on $\mathcal{X}_a \times \mathcal{Y}_a^*$. The critical condition $DL(\overline{x}, \overline{y}^*) = 0$ leads to the Lagrange equations:

$$\Lambda \overline{x} = DW^*(\overline{y}^*), \quad \Lambda^* \overline{y}^* = DF(\overline{x}).$$

Clearly, $L\colon \mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ is a supercritical function, and

$$P(x) = \sup_{y^* \in \mathcal{Y}_a^*} L(x, y^*), \quad \forall x \in \mathcal{X}_k.$$

Dually, for any given dual feasible $y^* \in \mathcal{Y}_s^*$,

$$P^d(y^*) = \sup_{x \in \mathcal{X}_a} L(x, y^*) = F^*(\Lambda^* y^*) - W^*(y^*),$$

where the dual feasible set $\mathcal{Y}_s^* \subset \mathcal{Y}_a^*$ is defined by

$$\mathcal{Y}_s^* = \{y^* \in \mathcal{Y}_a^* \colon \Lambda^* y^* \in \mathcal{X}_a^*\}.$$

The criticality conditions $DL(\overline{x}, \overline{y}^*) = 0$, $DP(\overline{x}) = 0$ and $DP^d(\overline{y}^*) = 0$ are equivalent to each other.

The *Hamiltonian* $H\colon \mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ associated with the Lagrangian L is defined by

$$H(x, y^*) = \langle \Lambda x; y^* \rangle - L(x, y^*)$$
$$= W^*(y^*) + F(x). \tag{20}$$

For d.c. programming, $H(x, y^*)$ is a convex function on $\mathcal{X}_a \times \mathcal{Y}_a^*$. The critical point $(\overline{x}, \overline{y}^*)$ of L satisfies the Hamiltonian *canonical form*

$$\Lambda \overline{x} = D_{y^*} H(\overline{x}, \overline{y}^*), \quad \Lambda^* \overline{y}^* = D_x H(\overline{x}, \overline{y}^*).$$

Particularly, if $W(\Lambda x) = 1/2 \langle \Lambda x, C \Lambda x \rangle$ is a quadratic function, $C\colon \mathcal{Y}_a \to \mathcal{Y}_a^*$ is a symmetric operator such that the composite operator $A = \Lambda^* C \Lambda = A^*$ is selfadjoint, then the total action can be written as

$$P(x) = \frac{1}{2} \langle x, Ax \rangle - F(x).$$

Let $P^c(x) = -P^d(C \Lambda x)$; then the function $P^c\colon \mathcal{X}_a \to \mathbf{R}$

$$P^c(x) = \frac{1}{2} \langle x, Ax \rangle - F^*(Ax)$$

is the so-called *Clarke dual action* (see [1]).

Theorem 5 (*Clarke duality theorem*) *Let A: $\mathcal{X}_k \subset \mathcal{X}_a \to \mathcal{X}_a^*$ be a closed selfadjoint operator, and Ker $A = \{x \in \mathcal{X}: A\,x = 0 \in \mathcal{X}^*\}$ the null space of A. If $\overline{x} \in \mathcal{X}_k$ is a critical point of P, then any vector $x \in \operatorname{Ker} A + \overline{x}$ is a critical point of P^c.*

Conversely, if there exists a $x_o \in \mathcal{X}_k$ such that $A\,x_o \in \mathcal{X}_a^*$, then for a given critical point $\overline{x}$ of P^c, any vector $x \in \operatorname{Ker} A + \overline{x}$ is a critical point of P.

Example 6 Let us consider a very simple one-dimensional optimization problem with constraint

$$\begin{cases} F(x) = & \frac{1}{2}kx^2 - fx \to \max \\ \text{s.t.} & a \le x \le b, \end{cases} \tag{21}$$

where $k > 0$ and $f \in \mathbf{R}$ are given constants. We assume that $-\infty < a < 0 < b < \infty$. Since $F(x)$ is strictly convex on the closed set $[a, b]$, the maximum is attained only on the boundary, i. e.

$$\sup_{x\in[a,b]} F(x) = \max\{F(a), F(b)\} < \infty.$$

The classical Lagrange multiplier method cannot be used for this nonconvex problem. To set this problem within our framework, we need only set $\mathcal{X} = \mathbf{R}$, $\mathcal{X}_a = [a, b]$ and let $\Lambda = 1$, so that

$$y = \Lambda x = x \in \mathcal{Y} = \mathbb{R}.$$

Thus, the range of the mapping Λ: $\mathcal{X}_a \to \mathcal{Y} = \mathbf{R}$ is $\mathcal{Y}_a = [a, b]$. Let

$$W(y) = \begin{cases} 0 & \text{if } y \in \mathcal{Y}_a, \\ +\infty & \text{if } y \notin \mathcal{Y}_a. \end{cases}$$

It is not difficult to check that W: $\mathcal{Y} \to \mathbf{R} \cup \{+\infty\}$ is convex. On $\mathcal{Y}_a$, W is finite and differentiable. Thus, the primal feasible space can be defined by

$$\mathcal{X}_k = \{x \in \mathcal{X}_a:\ \Lambda x = x \in \mathcal{Y}_a\} = [a, b].$$

Clearly, on $\mathcal{X}_k$ $P(x) = W(\Lambda\, x) - F(x) = F(x)$. The constrained maximization problem (21) is then equivalent to the standard nonconvex minimization problem ($\mathcal{P}_{\text{inf}}$): $P(x) \to \min$, $\forall x \in \mathcal{X}_k$.

Since $F(x)$ is strictly convex and differentiable on $\mathcal{X}_a = [a, b]$, and

$$x^* = DF(x) = kx - f \in \mathcal{X}_a^*$$

is invertible, where

$$\mathcal{X}_a^* = [ak - f, bk - f] \subset \mathcal{X}^* = \mathbb{R},$$

the Legendre conjugate P^c: $\mathcal{X}_a^* \to \mathbf{R}$ can easily be obtained as

$$F^*(x^*) = \max_{x\in\mathcal{X}_a}\{xx^* - F(x)\} = \frac{1}{2k}(x^* + f)^2.$$

By the *Fenchel transformation*, the conjugate of the nonsmooth function W can be obtained as

$$W^*(y^*) = \sup_{y\in\mathcal{Y}}\{yy^* - W(y)\} = \max_{y\in\mathcal{Y}_a} yy^*$$

$$= \begin{cases} by^* & \text{if } y^* > 0, \\ 0 & \text{if } y^* = 0, \\ ay^* & \text{if } y^* < 0. \end{cases}$$

It is convex and differentiable on $\mathcal{Y}_a^* = \mathcal{Y}^* = \mathbf{R}$.

On $\mathcal{X}_a \times \mathcal{Y}_a^* = [a, b] \times \mathbf{R}$, the Lagrange form for this nonconvex programming is well-defined by

$$L(x, y^*) = y^*\Lambda x - W^*(y^*) - F(x)$$

$$= \begin{cases} xy^* - by^* - \frac{1}{2}kx^2 + fx & \text{if } y^* \ge 0, \\ \\ xy^* - ay^* - \frac{1}{2}kx^2 + fx & \text{if } y^* < 0. \end{cases}$$

Since both W^* and F are convex, $L(x, y^*)$ is a supercritical point function. If $x \in \mathcal{X}_k = [a, b]$, then

$$P(x) = \sup_{y^*\in\mathcal{Y}_a^*} L(x, y^*).$$

On the other hand, for any y^* in the dual feasible space

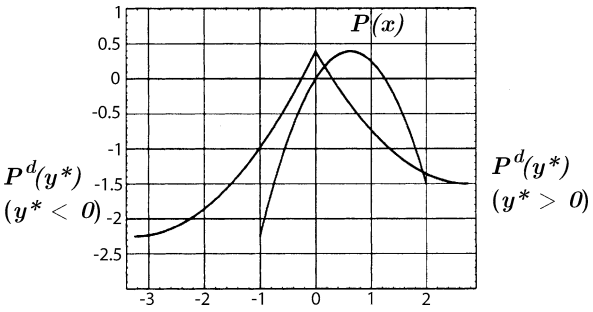
$$\mathcal{Y}_s^* = \{y^* \in \mathcal{Y}_a^* = \mathbb{R}:\ \Lambda^* y^* = y^* \in \mathcal{X}_a^*\}$$
$$= [ak - f, bk - f],$$

the dual function is obtained by

$$P^d(y^*) = \sup_{x\in\mathcal{X}_a} L(x, y^*)$$
$$= \sup_{x\in\mathbb{R}}\{(\Lambda x)y^* - F(x)\} - W^*(y^*)$$
$$= F^*(\Lambda^* y^*) - W^*(y^*),$$

where

$$F^*(\Lambda^* y^*) = \sup_{x\in\mathcal{X}_a}\{(\Lambda x)y^* - F(x)\}$$
$$= \sup_{x\in\mathbb{R}}\{x(y + f) - \frac{1}{2}kx^2\}$$
$$= \frac{1}{2k}(y^* + f)^2.$$



Duality Theory: Biduality in Nonconvex Optimization, Figure 1
Biduality in constrained nonconvex optimization

Thus, the dual action P^d is well defined on $\mathcal{Y}_s^*$ by

$$P^d(y^*) = \begin{cases} \frac{1}{2k}(y^* + f)^2 - by^* & \text{if } y^* > 0, \\ \frac{1}{2k} f^2 & \text{if } y^* = 0, \\ \frac{1}{2k}(y^* + f)^2 - ay^* & \text{if } y^* < 0. \end{cases} \quad (22)$$

This is a double-well function on $\mathbf{R}$ (see Fig. 1.). The dual problem

$$(P_{\text{inf}}^d): \quad P^d(y^*) \to \min, \quad \forall y^* \in \mathcal{Y}_s^*,$$

is a convex optimization problem on either

$$\mathcal{Y}_s^{*+} = \{y^* \in \mathcal{Y}_s^* : \ y^* > 0\}$$

or

$$\mathcal{Y}_s^{*-} = \{y^* \in \mathcal{Y}_s^* : \ y^* < 0\}.$$

In n-dimensional problems, this dual problem is much easier than the primal problem. The criticality condition leads to

$$\overline{y}^* = \begin{cases} bk - f & \text{if } \overline{y}^* > 0, \\ 0 & \text{if } \overline{y}^* = 0 \\ ak - f & \text{if } \overline{y}^* < 0. \end{cases}$$

It is easy to check that the following duality theorems hold:

$$\min_{x \in X_k} P(x) = \min_{y^* \in \mathcal{Y}_s^*} P^d(y^*),$$

$$\max_{x \in X_k} P(x) = \max_{y^* \in \mathcal{Y}_s^*} P^d(y^*).$$

The graphs of $P(x)$ and $P^d(y^*)$ are shown in Fig. 1.

See also

- ▶ Duality Theory: Monoduality in Convex Optimization
- ▶ Duality Theory: Triduality in Global Optimization
- ▶ History of Optimization
- ▶ Von Neumann, John

References

1. Gao DY (1999) Duality principles in nonconvex systems: Theory, methods and applications. Kluwer, Dordrecht

Duality Theory: Monoduality in Convex Optimization

DAVID YANG GAO
Virginia Polytechnic Institute and State University, Blacksburg, USA

MSC2000: 49-XX, 90-XX, 93-XX

Article Outline

Keywords
Saddle Lagrange Duality
Fenchel–Rockafellar Duality
Linear Programming and Central Path
See also
References

Keywords

Duality; Convex optimization; Saddle Lagrangian; Legendre duality; Fenchel–Rockafellar duality; Complementarity; Linear programming; Center path

The concept of *duality* is one of the most successful ideas in modern mathematics and science. Inner beauty in natural phenomena is bound up with duality, which has always been a rich source of inspiration in human knowledge through the centuries. Duality in mathematics, roughly speaking, is a fundamental concept that underlies many aspects of *extremum principles* in natural systems. Eigenvectors, geodesics, minimal surfaces, KKT conditions, harmonic maps, Hamiltonian canonical equations and equilibrium states of many field equations are all critical points of certain functions on some appropriate constraint sets or manifolds. Considerable

attention has been attracted on this fascinating research subject during the last years. A comprehensive study on duality theory in general nonconvex and nonsmooth systems is given in [1]. In global optimization problems, duality falls principally into three categories:

1) the classical *saddle Lagrange duality* (i. e. monoduality) in convex optimization;
2) the nice *biduality* in convex Hamilton systems or the *d.c. programming* (difference of convex functions); and
3) the interesting *triduality* in general nonconvex systems.

Saddle Lagrange Duality

Let $(\mathcal{X}, \mathcal{X}^*)$ and $(\mathcal{Y}, \mathcal{Y}^*)$ be two pairs real vector spaces, finite- or infinite-dimensional, and let $\langle *, * \rangle: \mathcal{X} \times \mathcal{X}^* \to \mathbf{R}$ and $\langle *; * \rangle: \mathcal{Y} \times \mathcal{Y}^* \to \mathbf{R}$ be certain *bilinear forms* which put the paired spaces $(\mathcal{X}, \mathcal{X}^*)$ and $(\mathcal{Y}, \mathcal{Y}^*)$ in *duality*, respectively. In classical convex optimization, a real-valued function $L: \mathcal{X} \times \mathcal{Y}^* \to \mathbf{R}$ is said to be a *saddle function* if it is convex in one variable and concave in the other one.

A pair $(\overline{x}, \overline{y}^*)$ is called a *right saddle point* of L on a subspace $\mathcal{X}_a \times \mathcal{Y}_{a*} \subset \mathcal{X} \times \mathcal{Y}^*$ if

$$L(\overline{x}, y^*) \le L(\overline{x}, \overline{y}^*) \le L(x, \overline{y}^*)$$

holds for any $(x, y^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$.

A pair $(\overline{x}, \overline{y}^*)$ is called a *left saddle point* of L on a subspace $\mathcal{X}_a \times \mathcal{Y}_a^* \subset \mathcal{X} \times \mathcal{Y}^*$ if it is a right saddle point of $-L$ on the subspace $\mathcal{X}_a \times \mathcal{Y}_a^*$.

A pair $(\overline{x}, \overline{y}^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$ is called a *critical point* of L if L is partially Gâteaux differentiable at $(\overline{x}, \overline{y}^*)$ and

$$D_x L(\overline{x}, \overline{y}^*) = 0, \quad D_{y^*} L(\overline{x}, \overline{y}^*) = 0,$$

where $D_x L: \mathcal{X}_a \to \mathcal{X}^*$ and $D_{y^*} L: \mathcal{Y}_a^* \to \mathcal{Y}$ denote, respectively, the partial Gâteaux derivatives of L with respect to x and y^*.

Any critical point of a Gâteaux differentiable saddle function is a saddle point. However, if $(\overline{x}, \overline{y}^*)$ is a saddle point of L it does not follow that L is a saddle function. In convex optimization problems, the following statements are of important theoretical value.

S1) Under certain necessary and sufficient conditions, suppose that

$$\inf_{x \in \mathcal{X}_a} \sup_{y^* \in \mathcal{Y}_a^*} L(x, y^*) = \sup_{y^* \in \mathcal{Y}_a^*} \inf_{x \in \mathcal{X}_a} L(x, y^*). \quad (1)$$

A statement of this type is called a *saddle-minimax theorem* and the pair $(\overline{x}, \overline{y}^*)$ is called a *saddle-minimax point* of L on $\mathcal{X}_a \times \mathcal{Y}_a^*$.

S2) Under certain conditions, suppose that a pair $(\overline{x}, \overline{y}^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$ exists such that

$$L(x, \overline{y}^*) \ge L(\overline{x}, \overline{y}^*) \ge L(\overline{x}, y^*) \quad (2)$$

holds for any $(x, y^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$. A statement of this type is called a *right saddle-point theorem*.

Let $\mathcal{X}_k$ be a subset of $\mathcal{X}_a$ such that $\mathcal{X}_k$ contains all point $u \in \mathcal{X}_a$ for which the supremum $\sup_{y^*} L(x, y^*)$ is finite, i. e.

$$\sup_{y^* \in \mathcal{Y}_a^*} L(x, y^*) < +\infty, \quad \forall x \in \mathcal{X}_k.$$

Dually, let $\mathcal{Y}_s^*$ be a subset of $\mathcal{Y}_a^*$ such that $\mathcal{Y}_s^*$ contains all points $y^* \in \mathcal{Y}_a^*$ for which the infimum $\inf_x L(x, y^*)$ is finite, i. e.

$$\inf_{x \in \mathcal{X}_a} L(x, y^*) > -\infty, \quad \forall y^* \in \mathcal{Y}_s^*.$$

The sets $\mathcal{X}_k$ and $\mathcal{Y}_s^*$ may be either empty or $\mathcal{X}_k = \mathcal{X}_a$ and $\mathcal{Y}_s^* = \mathcal{Y}_a^*$. The connection between the minimax theorem and the saddle-point theorem is given by the following results.

Theorem 1 (*Saddle-minimax theorem*) *Let $L: \mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ be a given arbitrary function. If there exists a saddle-minimax point $(\overline{x}, \overline{y}^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$ such that*

$$\begin{aligned} L(\overline{x}, \overline{y}^*) &= \min_{x \in \mathcal{X}_a} \max_{y^* \in \mathcal{Y}_a^*} L(x, y^*) \\ &= \max_{y^* \in \mathcal{Y}_a^*} \min_{x \in \mathcal{X}_a} L(x, y^*), \end{aligned} \quad (3)$$

then $(\overline{x}, \overline{y}^*)$ is a saddle point of L on $\mathcal{X}_a \times \mathcal{Y}_a^*$.

Conversely, if $L(x, y^*)$ possesses a saddle point $(\overline{x}, \overline{y}^*)$ on $\mathcal{X}_a \times \mathcal{Y}_a^*$, then the saddle-minimax theorem in the form

$$\begin{aligned} L(\overline{x}, \overline{y}^*) &= \min_{x \in \mathcal{X}_k} \max_{y^* \in \mathcal{Y}_a^*} L(x, y^*) \\ &= \max_{y^* \in \mathcal{Y}_s^*} \min_{x \in \mathcal{X}_a} L(x, y^*) \end{aligned} \quad (4)$$

holds.

Let $L: \mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ be a given arbitrary right saddle function. For any fixed $x \in \mathcal{X}_a$, let

$$P(x) = \sup_{y^* \in \mathcal{Y}_a^*} L(x, y^*). \quad (5)$$

Let $\mathcal{X}_k \subset \mathcal{X}_a$ be the domain of P such that P: $\mathcal{X}_k \to \mathbf{R}$ is finite and Gâteaux differentiable. Then the inf-problem

$$(\mathcal{P}_{\inf}): \quad P(x) \to \min, \quad \forall u \in \mathcal{X}_k, \tag{6}$$

is called the *primal problem*.

Dually, for any fixed $y^* \in \mathcal{Y}_a^*$, let

$$P^d(y^*) = \sup_{x \in \mathcal{X}_a} L(x, y^*) \tag{7}$$

with domain $\mathcal{Y}_s^* \subset \mathcal{Y}_a^*$, on which, P^d:$\mathcal{Y}_s^* \to \mathbf{R}$ is finite and Gâteaux differentiable. Thus, the sup-problem

$$(\mathcal{P}_{\sup}^d): \quad P^d(y^*) \to \max, \quad \forall y^* \in \mathcal{Y}_s^*, \tag{8}$$

is referred as the *dual problem*. The problems $(\mathcal{P}_{\inf})$ and $(\mathcal{P}_{\sup}^d)$ are *realisable* if $\mathcal{X}_k$ and $\mathcal{Y}_s^*$ are not empty, i. e., there exists a pair $(\overline{x}, \overline{y}^*) \in \mathcal{X}_k \times \mathcal{Y}_s^*$ such that

$$P(\overline{x}) = \min_{x \in \mathcal{X}_k} P(x) = \inf_{x \in \mathcal{X}_k} P(x),$$
$$P^d(\overline{y}^*) = \max_{y^* \in \mathcal{Y}_s^*} P^d(y^*) = \sup_{y^* \in \mathcal{Y}_s^*} P^d(y^*).$$

Theorem 2 (*Saddle duality theorem*) *Let L: $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ be a given arbitrary function such that P and P^d are well-defined by (5) and (7) on the open subsets $\mathcal{X}_k$ and $\mathcal{Y}_s^*$, respectively. If $(\overline{x}, \overline{y}^*)$ is a saddle point of L on $\mathcal{X}_k \times \mathcal{Y}_s^*$, P is Gâteaux differentiable at $\overline{x}$, and P^d is Gâteaux differentiable at $\overline{y}^*$, then $DP(\overline{x}) = 0$, $DP^d(\overline{y}^*) = 0$, and*

$$P(\overline{x}) = L(\overline{x}, \overline{y}^*) = P^d(\overline{y}^*). \tag{9}$$

Theorem 3 (*Weak duality theorem*) *The inequality*

$$P(x) \geq P^d(y^*) \tag{10}$$

holds for all $(x, y^*) \in \mathcal{X}_k \times \mathcal{Y}_s^*$.

Theorem 4 (*Strong duality theorem*) *$(\overline{x}, \overline{y}^*)$ is a saddle-point of L on $\mathcal{X}_k \times \mathcal{Y}_s^* \subseteq \mathcal{X}_a \times \mathcal{Y}_a^*$ if and only if the equality*

$$P(\overline{x}) = \inf_{x \in \mathcal{X}_k} P(x) = \sup_{y^* \in \mathcal{Y}_s^*} P^d(y^*) = P^d(\overline{y}^*) \tag{11}$$

holds.

Fenchel–Rockafellar Duality

Very often, the primal function P: $\mathcal{X}_k \to \mathbf{R}$ can be written as

$$P(x) = W(\Lambda x) - F(x),$$

where Λ: $\mathcal{X} \to \mathcal{Y}$ is a linear operator, W: $\mathcal{Y}_a \to \mathbf{R}$ and F: $\mathcal{X}_a \to \mathbf{R}$ are Gâteaux differentiable real-valued functions. The feasible set $\mathcal{X}_k \subset \mathcal{X}$ is then defined by

$$\mathcal{X}_k = \{x \in \mathcal{X}_a : \ \Lambda x \in \mathcal{Y}_a\}.$$

Clearly, P: $\mathcal{X}_k \to \mathbf{R}$ is convex if W is convex on $\mathcal{Y}_a$ and F is concave on $\mathcal{X}_a$.

The *conjugate function* W^*: $\mathcal{Y}_a^* \to \mathbf{R}$ of $W(y)$ is defined by the *Fenchel transformation*, i. e.

$$W^*(y^*) = \sup_{y \in \mathcal{Y}_a} \{\langle y; y^* \rangle - W(y)\}, \tag{12}$$

which is always l.s.c. and convex on $\mathcal{Y}^*$. The following *Fenchel–Young inequality*

$$W(y) \geq \langle y; y^* \rangle - W^*(y^*) \tag{13}$$

holds on $\mathcal{Y}_a \times \mathcal{Y}_a^*$. If W is strictly convex, and Gâteaux differentiable on $\mathcal{Y}_a \subset \mathcal{Y}$, then the following *Legendre duality relations*

$$y^* = DW(y) \Leftrightarrow y = DW^*(y^*)$$
$$\Leftrightarrow \langle y; y^* \rangle = W(y) + W^*(y^*)$$

hold on $\mathcal{Y}_a \times \mathcal{Y}_a^*$. In this case, we have $W(y) = W^{**}(y)$, the biconjugate of W, and the Fenchel transformation (12) is equivalent to the classical *Legendre transformation*

$$W^*(y^*) = \langle y(y^*); y^* \rangle - W(y(y^*)).$$

The *Lagrangian form* associated with $(\mathcal{P}_{\inf})$ is defined by

$$L(x, y^*) = \langle \Lambda x; y^* \rangle - W^*(y^*) - F(x), \tag{14}$$

which is Gâteaux differentiable on $\mathcal{X}_a \times \mathcal{Y}_a^*$. The critical condition $DL(\overline{x}, \overline{y}^*) = 0$ leads to the *Lagrange equations*:

$$\Lambda \overline{x} = DW^*(\overline{y}^*), \quad \Lambda^* \overline{y}^* = DF(\overline{x}), \tag{15}$$

where Λ^*: $\mathcal{Y}_a^* \to \mathcal{X}_a^*$ is the adjoint operator of Λ. Clearly, L: $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ is a right saddle function if

$F(x)$ is concave on $\mathcal{X}_a$. For convex function $W(y)$, we have

$$P(x) = \sup_{y^* \in \mathcal{Y}_a^*} L(x, y^*), \quad \forall x \in \mathcal{X}_k.$$

The Fenchel conjugate function of a concave function F: $\mathcal{X}_a \to \mathbf{R}$ is defined by

$$F^*(x^*) = \inf_{x \in \mathcal{X}_a} \{\langle x, x^* \rangle - F(x)\}. \tag{16}$$

Thus, for any given dual admissible $y^* \in \mathcal{Y}_s^*$ with

$$\mathcal{Y}_s^* = \left\{ y^* \in \mathcal{Y}_a^* : \ \Lambda^* y^* \in \mathcal{X}_a^* \right\},$$

the Fenchel–Rockafellar dual function P^d: $\mathcal{Y}_k^* \to \mathbf{R}$ can be obtained as

$$P^d(y^*) = \inf_{x \in \mathcal{X}_a} L(x, y^*) = F^*(\Lambda^* y^*) - W^*(y^*).$$

If P is Gâteaux differentiable on $\mathcal{X}_k$, the critical condition $DP(\overline{x}) = 0$ leads to the *Euler–Lagrange equation* of the primal problem $(\mathcal{P}_{\inf})$:

$$\Lambda^* DW(\Lambda \overline{x}) - DF(\overline{x}) = 0. \tag{17}$$

Similarly, the critical condition $DP^d(\overline{y}^*) = 0$ gives the *dual Euler–Lagrange equation* of $(\mathcal{P}_{\sup}^d)$:

$$\Lambda DF^*(\Lambda^* \overline{y}^*) - DW^*(\overline{y}^*) = 0. \tag{18}$$

Clearly, the critical point theorem (9) holds if the Lagrange equation (15), Euler–Lagrange equation (17) and its dual equation (18) are equivalent to each others.

For any given F and W, the weak duality theorem (10) always holds on $\mathcal{X}_k \times \mathcal{F}_s^*$. The difference inf P − sup P^d is the so-called *duality gap*. For convex primal problem, the duality gap is zero and the strong Lagrange duality theorem (11) holds, which is also referred as the *Fenchel–Rockafellar duality theory*.

Linear Programming and Central Path

Let us now demonstrate how the above scheme fits in with finite-dimensional linear programming. Let $\mathcal{X} = \mathcal{X}^* = \mathbf{R}^n$, $\mathcal{Y} = \mathcal{F}^* = \mathbf{R}^m$, with the standard inner products $\langle x, x^* \rangle = x^{\mathsf{T}} x^*$ in $\mathbf{R}^n$, and $\langle y; y^* \rangle = y^{\mathsf{T}} y^*$ in $\mathbf{R}^m$. For fixed $\overline{x}^* = c \in \mathbb{R}^n$ and $\overline{y} = b \in \mathbb{R}^m$, the primal problem is a constrained linear optimization problem:

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} & \langle c, x \rangle \\ \text{s.t.} & \Lambda x = b, \quad x \geq 0, \end{cases} \tag{19}$$

where $\Lambda \in \mathbf{R}^{m \times n}$ is a matrix, and its adjoint is simply $\Lambda^* = \Lambda^{\mathsf{T}} \in \mathbf{R}^{n \times m}$. To reformulate this linear constrained optimization problem in the model form $(\mathcal{P}_{\inf})$, we need to set $\mathcal{X}_a = \{x \in \mathbf{R}^n x \geq 0\}$, which is a convex cone in $\mathbf{R}^n$, $\mathcal{Y}_a = \{y \in \mathbf{R}^m : y = b\}$, a hyperplane in $\mathbf{R}^m$, and let

$$F(x) = -\langle c, x \rangle, \quad \forall x \in \mathcal{X}_a,$$
$$W(y) = 0, \quad \forall y \in \mathcal{Y}_a.$$

Thus on the primal feasible set

$$\mathcal{X}_k = \{x \in \mathbb{R}^n : \ \Lambda x = b, \ x \geq 0\}$$

we have $P(x) = W(\Lambda x) - F(x) = \langle c, x \rangle$. The conjugate functions in this elementary case may be calculated at once as

$$W^*(y^*) = \sup_{y \in \mathcal{Y}_a} \langle y; y^* \rangle = \langle b; y^* \rangle,$$
$$\forall y^* \in \mathcal{Y}_a^* = \mathbb{R}^m,$$
$$F^*(x^*) = \inf_{x \in \mathcal{X}_a} \langle x, x^* + c \rangle = 0, \quad \forall x^* \in \mathcal{X}_a^*,$$

where $\mathcal{X}_a^* = \{x^* \in \mathbf{R}^n : x^* + c \geq 0\}$ is a polar cone of $\mathcal{X}_a$. Thus, on the dual feasible space

$$\mathcal{Y}_s^* = \{y^* \in \mathbb{R}^m : \ \Lambda^* y^* + c \geq 0\},$$

the problem dual to the linear programming (19) reads

$$\max_{p \in \mathbb{R}^m} P^d(y^*) = -\langle b; y^* \rangle, \quad \forall y^* \in \mathcal{Y}_s^*. \tag{20}$$

The Lagrangian L: $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ associated with this constrained linear programming is

$$L(x, y^*) = \langle \Lambda x; y^* \rangle - \langle b; y^* \rangle + \langle c, x \rangle$$
$$= \langle x, \Lambda^* y + c \rangle - \langle b; y^* \rangle.$$

But for inequality constraints in $\mathcal{X}_a$, the Lagrange multiplier $x^* = \Lambda^* y^* \in \mathbf{R}^n$ has to satisfy the following *KKT optimality conditions*

$$\begin{aligned} &\Lambda x = b, \quad s = c + \Lambda^* y^*, \\ &x \geq 0, \quad s \geq 0, \quad s^{\top} x, = 0, \end{aligned} \tag{21}$$

where the vector $s \in \mathbf{R}^n$ is called the *dual slacks*. The problem of finding $(\overline{x}, \overline{y}^*, \overline{s})$ satisfying (21) is also known as the *mixed linear complementarity problem*.

By using the vector of dual slacks $s \in \mathbf{R}^n$, the dual problem can be rewritten as

$$\begin{cases} \max\limits_{p \in \mathbb{R}^m} & \langle b; p \rangle \\ \text{s.t.} & \Lambda^* y^* + c - s = 0, \quad s \geq 0. \end{cases} \tag{22}$$

We can see that the primal variable x is the Lagrange multiplier for the constraint $\Lambda^* y^* + c \geq 0$ in the dual problem. However, the dual variables y^* and s are respectively Lagrange multipliers for the constraints $\Lambda x = b$ and $x \geq 0$ in the primal problem. These choices are not accidents.

Theorem 5 *The vector $\overline{x} \in \mathbb{R}^n$ is a solution of (19) if and only if there exists a Lagrange multipliers $(\overline{y}^*, \overline{s}) \in \mathbb{R}^m \times \mathbb{R}^n$ for which the KKT optimality conditions (21) hold for $(\overline{x}, \overline{y}^*, \overline{s})$. Dually, the vector $(\overline{y}^*, \overline{s}) \in \mathbb{R}^m \times \mathbb{R}^n$ is a solution of (22) if and only if there exists a Lagrange multiplier $\overline{x} \in \mathbb{R}^n$ such that the KKT conditions (21) hold for $(\overline{x}, \overline{y}^*, \overline{s})$.*

The vector $(\overline{x}, \overline{y}^*, \overline{s})$ is called a *primal-dual solution* of (19). The so-called *primal-dual methods* in mathematical programming are those methods to find primal-dual solutions $(\overline{x}, \overline{y}^*, \overline{s})$ by applying variants of Newton's method to the three equations in (21) and modifying the search directions and steplengths so that the inequalities in (21) are satisfied at every iteration. If the inequalities are strictly satisfied, the methods are called *primal-dual interior-point methods*. In these methods, the so called *central path* $\mathcal{C}_{\text{path}}$ plays a vital role in the theory of primal-dual algorithms. It is a parametrical curve of strictly feasible points defined by

$$C_{\text{path}} = \left\{(x_\tau, y_\tau^*, s_\tau)^\top \in \mathbb{R}^{2n+m} : \ \tau > 0\right\}, \qquad (23)$$

where each point $(x_\tau, y_\tau^*, s_\tau)$ solves the following system:

$$\begin{aligned} &\Lambda x = b, \quad \Lambda^* y^* + c = s, \\ &x > 0, \quad s > 0, \quad u_i s_i = \tau, \quad i = 1, \ldots, n. \end{aligned} \qquad (24)$$

This problem has a unique solution $(x_\tau, y_\tau^*, s_\tau)$ for each $\tau > 0$ if and only if the *strictly feasible set*

$$\mathcal{F}_o = \left\{(x, p, s): \quad \begin{array}{c} \Lambda x = b, \ \Lambda^* y^* + c = s, \\ x > 0, \ s > 0 \end{array} \right\}$$

is nonempty. A comprehensive study of the primal-dual interior-point methods in mathematical programming has been given in [3] and [2].

See also

- ► Duality Theory: Biduality in Nonconvex Optimization
- ► Duality Theory: Triduality in Global Optimization
- ► History of Optimization

References

1. Gao DY (1999) Duality principles in nonconvex systems: Theory, methods and applications. Kluwer, Dordrecht
2. Wright SJ (1996) Primal-dual interior-point methods. SIAM, Philadelphia
3. Ye Y (1997) Interior point algorithms: Theory and analysis. Discrete Math and Optim. Wiley/Interscience, New York

Duality Theory: Triduality in Global Optimization

DAVID YANG GAO
Virginia Polytechnic Institute and State University, Blacksburg, USA

MSC2000: 49-XX, 90-XX, 93-XX

Article Outline

Keywords
Canonical Dual Transformation
Triality Theory
See also
References

Keywords

Duality; Nonconvexity; Global minimization; SuperLagrangian; Triduality; Triality

Consider the following general nonconvex extremum problem $(\mathcal{P})$:

$$P(x) = \Phi(x, \Lambda(x)) \to \text{extremum}, \quad \forall x \in \mathcal{X}, \quad (1)$$

where $\mathcal{X}$ is a locally convex topological vector space (l.c.s.), $P: \ \mathcal{X} \to \overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty\} \cup \{+\infty\}$ is a nonconvex and nonsmooth extended function, whose effective domain

$$\mathcal{X}_k = \text{dom}\, P = \{x \in \mathcal{X} : \ |P(x)| < +\infty\}$$

is a nonempty convex subset of $\mathcal{X}$; the operator $\Lambda : \mathcal{X} \to \mathcal{Y}$ is a continuous, generally nonlinear, mapping from $\mathcal{X}$ to another l.c.s. $\mathcal{Y}$, and $\Phi : \ \mathcal{X} \times \mathcal{Y} \to \overline{\mathbb{R}}$ is an associated extended function. Since the cost function $P(x)$ is usually nonconvex, the problem $(\mathcal{P})$ may possess many locally extremum (either minimum or maximum) solutions. The goal of *global optimization* is to find all the

local extrema of $P(x)$ over the feasible set $\mathcal{X}_k$. Generally speaking, traditional direct approaches and algorithms for solving nonconvex, nonsmooth global optimization problems are usually very difficult. The classical saddle Lagrange duality methods as well as the well-known Fenchel-Rockafellar duality theory can be used mainly for solving convex problems. For nonconvex problems, there exists a so-called *duality gap* between the primal and the classical dual problems.

The *canonical dual transformation method* andassociated *triduality theory* were proposed originally in finite deformation theory [1]. The key idea of this method is to choose a suitable nonlinear operator Λ: $\mathcal{X} \to \mathcal{Y}$ such that $\Phi(x, y)$ is either convex or concave in each of its variables. This method can be used to solve many nonconvex, nonsmooth global optimization problems.

Canonical Dual Transformation

Let $(\mathcal{X}, \mathcal{X}^*)$ be a pair of real linear spaces, placed in duality by a bilinear form $\langle \cdot, \cdot \rangle : \mathcal{X} \times \mathcal{X}^* \to \mathbf{R}$. For a given extended real-valued function $P\colon \mathcal{X} \to \overline{\mathbf{R}}$, the *subdifferential* of P at $\overline{x} \in \mathcal{X}$ is a convex subset $\partial^- P(\overline{x}) \subset \mathcal{X}^*$ such that for each $\overline{x}^* \in \partial^- P(\overline{x})$, we have

$$\left\langle \overline{x}^*, x - \overline{x} \right\rangle \le P(x) - P(\overline{x}), \quad \forall x \in \mathcal{X}.$$

Dually, the *superdifferential* of P at $\overline{x} \in \mathcal{X}$ is a convex subset $\partial^+ P(\overline{x}) \subset \mathcal{X}^*$ such that for each $\overline{x}^* \in \partial^+ P(\overline{x})$, we have

$$\left\langle \overline{x}^*, x - \overline{x} \right\rangle \ge P(x) - P(\overline{x}), \quad \forall x \in \mathcal{X}.$$

Clearly, we always have $\partial^+ P = -\partial^-(-P)$. In convex analysis, it is convention that ∂^- is simply written as ∂. In nonconvex analysis, ∂stands for either ∂^- or ∂^+, i. e.

$$\partial = \{\partial^-, \partial^+\}.$$

If P is smooth, Gâteaux-differentiable at $\overline{x} \in \mathcal{X}_a \subset \mathcal{X}$, then

$$\partial P(\overline{x}) = \partial^- P(\overline{x}) = \partial^+ P(\overline{x}) = \{DP(\overline{x})\},$$

where $DP : \mathcal{X}_a \to \mathcal{X}^*$ denotes the Gâteaux derivative of P at $\overline{x}$.

Definition 1 The set of functions $P\colon \mathcal{X} \to \overline{\mathbb{R}}$ which are either convex or concave is denoted by $\Gamma(\mathcal{X})$. In particular, let $\check{\Gamma}(\mathcal{X})$ denote the subset of functions $P \in \Gamma(\mathcal{X})$ which are convex and $\widehat{\Gamma}(\mathcal{X})$ the subset of $P \in \Gamma(\mathcal{X})$ which are concave.

The *canonical function space* $\Gamma_G(\mathcal{X}_a)$ is a subset of functions $P \in \Gamma(\mathcal{X}_a)$ which are Gâteaux differentiable on $\mathcal{X}_a \subset \mathcal{X}$ and the duality mapping $DP : \mathcal{X}_a \to \mathcal{X}_a^* \subset \mathcal{X}^*$ is invertible.

The *extended canonical function space* $\Gamma_0(\mathcal{X})$ is a subset of functions $P \in \Gamma(\mathcal{X})$which are either convex, lower semicontinuous or concave, upper semicontinuous, and if P takes the values $\pm\infty$, then P is identically equal to $\pm\infty$.

By the Legendre–Fenchel transformation, the *supconjugate function* of an extended function $P\colon \mathcal{X} \to \overline{\mathbb{R}}$ is defined by

$$P^\sharp(x^*) = \sup_{x \in \mathcal{X}} \{\langle x, x^* \rangle - P(x)\}.$$

By the theory of convex analysis, $P^\sharp\colon \mathcal{X}^* \to \vec{\mathbb{R}} := \mathbb{R} \cup \{+\infty\}$ is always convex and lower semicontinuous, i. e. $P^\sharp \in \check{\Gamma}_0(\mathcal{X}^*)$. Dually, the *subconjugate function* of P, defined by

$$P^\flat(x^*) = \inf_{x \in \mathcal{X}} \{\langle x, x^* \rangle - P(x)\},$$

is always concave and upper semicontinuous, i. e. $P^\flat \in \widehat{\Gamma}_0(\mathcal{X}^*)$, and $P^\flat = -P^\sharp$. Both the super- and subconjugates are called *Fenchel conjugate functions* and we write $P^* = \{P^\flat, P^\sharp\}$. Thus the extended Fenchel transformation can be written as

$$P^*(x^*) = \operatorname{ext}\{\langle x, x^* \rangle - P(x)\colon \forall x \in \mathcal{X}\}, \tag{2}$$

where ext stands for extremum. Clearly, if $P \in \Gamma_0(\mathcal{X})$, we have the Fenchel equivalent relations, namely,

$$\begin{aligned} x^* \in \partial P(x) &\Leftrightarrow x \in \partial P^*(x^*) \\ &\Leftrightarrow P(x) + P^*(x^*) = \langle x, x^* \rangle . \end{aligned} \tag{3}$$

The pair (x, x^*) is called the *Fenchel duality pair* on $\mathcal{X} \times \mathcal{X}^*$ if and only if equation (3) holds on $\mathcal{X} \times \mathcal{X}^*$.

The conjugate pair (x, x^*) is said to be a *Legendre duality pair* on $\mathcal{X}_a \times \mathcal{X}_a^* \subset \mathcal{X} \times \mathcal{X}^*$ if and only if the equivalent relations

$$\begin{aligned} x^* = DP(x) &\Leftrightarrow x = DP^*(x^*) \\ &\Leftrightarrow P(x) + P^*(x^*) = \langle x, x^* \rangle \end{aligned} \tag{4}$$

hold on $\mathcal{X}_a \times \mathcal{X}_a^*$.

Let $(\mathcal{Y}, \mathcal{Y}^*)$ be an another pair of locally convex topological real linear spaces paired in separating duality by the second bilinear form $\langle\cdot;\cdot\rangle : \mathcal{Y} \times \mathcal{Y}^* \to \mathbf{R}$. The so-called *geometrical operator* $\Lambda : \mathcal{X} \to \mathcal{Y}$ is a continuous, Gâteaux differentiable operator such that for any given $x \in \mathcal{X}_a \subset \mathcal{X}$, there exists a $y \in \mathcal{Y}_a \subset \mathcal{Y}$ satisfying the *geometrical equation*

$$y = \Lambda(x).$$

The directional derivative of y at $\overline{x}$ in the direction $x \in \mathcal{X}$ is then defined by

$$\delta y(\overline{x}; x) := \lim_{\theta \to 0^+} \frac{y(\overline{x} + \theta x) - y(\overline{x})}{\theta} = \Lambda_t(\overline{x})x,$$

where $\Lambda_t(\overline{x}) = D\Lambda(\overline{x})$ denotes the Gâteaux derivative of the operator Λ at $\overline{x}$. For a given $y^* \in \mathcal{Y}^*$, $G^*_y(x) = \langle \Lambda(x); y^* \rangle$ is a real-valued function of x on $\mathcal{X}$. Its Gâteaux derivative at $\overline{x} \in \mathcal{X}_a$ in the direction $x \in \mathcal{X}$ is

$$\delta G_{y^*}(\overline{x}; x) = \langle \Lambda_t(\overline{x})x; y^* \rangle = \left\langle x, \Lambda^*_t(\overline{x})y^* \right\rangle,$$

where $\Lambda^*_t(\overline{x}) : \mathcal{Y}^* \to \mathcal{X}^*$ is the adjoint operator of Λ_t associated with the two bilinear forms.

Let $\Phi : \mathcal{X} \times \mathcal{Y} \to \overline{\mathbb{R}}$ be an extended function such that $P(x) = \Phi(x, \Lambda(x))$. If $\Phi : \mathcal{X} \times \mathcal{Y} \to \overline{\mathbb{R}}$ is an extended canonical function, i. e. $\Phi \in \Gamma_0(\mathcal{X}) \times \Gamma_0(\mathcal{Y})$, the duality relations between the paired spaces $(\mathcal{X}, \mathcal{X}^*)$ and $(\mathcal{Y}, \mathcal{Y}^*)$ can be written as

$$x^* \in \partial_x \Phi(x, y), \quad y^* \in \partial_y \Phi(x, y). \tag{5}$$

On the product space $\mathcal{X}_a \times \mathcal{Y}_a \subset \mathcal{X} \times \mathcal{Y}$, if the canonical function $\Phi(x, y)$ is finite and Gâteaux differentiable such that the feasible space $\mathcal{X}_k$ can be written as

$$\mathcal{X}_k = \{x \in \mathcal{X}_a : \Lambda(x) \in \mathcal{Y}_a\}, \tag{6}$$

then on $\mathcal{X}_k$, the critical condition $\delta P(\overline{x}; x) = \langle x, DP(\overline{x}) \rangle = 0, \forall x \in \mathcal{X}_k$, leads to the Euler equation

$$D_x \Phi(\overline{x}, \Lambda(\overline{x})) + \Lambda^*_t(\overline{x}) D_y \Phi(\overline{x}, \Lambda(\overline{x})) = 0, \tag{7}$$

where $D_x\Phi$ and $D_y\Phi$ denote the partial Gâteaux derivatives of Φ with respect to x and y, respectively. Since $\Phi \in \Gamma_G(\mathcal{X}_a) \times \Gamma_G(\mathcal{Y}_a)$ is a canonical function, the Gâteaux derivative $D\Phi : \mathcal{X}_a \times \mathcal{Y}_a \to \mathcal{X}^*_a \times \mathcal{Y}^*_a \subset \mathcal{X}^* \times \mathcal{Y}^*$ is a monotone mapping, i. e. there exists a pair $(\overline{x}^*, \overline{y}^*) \in \mathcal{X}^* \times \mathcal{Y}^*$ such that

$$-\overline{x}^* = D_x \Phi(\overline{x}, \Lambda(\overline{x})), \quad \overline{y}^* = D_y \Phi(\overline{x}, \Lambda(\overline{x})).$$

Thus, in terms of canonical dual variables $\overline{x}^*$ and $\overline{y}^*$, the Euler equation (7) can be written in the so-called *balance* (or *equilibrium*) equilibrium

$$\overline{x}^* = \Lambda^*_t(\overline{x})\overline{y}^*, \tag{8}$$

which linearly depends on the dual variable $\overline{y}^*$.

Definition 2 Suppose that for a given problem $(\mathcal{P})$, the geometrical operator $\Lambda : \mathcal{X} \to \mathcal{Y}$ can be chosen in such a way that $P(x) = \Phi(x, \Lambda(x))$, $\Phi \in \Gamma_G(\mathcal{X}_a) \times \Gamma_G(\mathcal{Y}_a)$ and $\mathcal{X}_k = \{x \in \mathcal{X}_a : \Lambda(x) \in \mathcal{Y}_a\}$. Then

1) the transformation $\{\mathrm{P}; \mathcal{X}_k\} \to \{\Phi; \mathcal{X}_a \times \mathcal{Y}_a\}$ is called the *canonical transformation*, and $\Phi : \mathcal{X}_a \times \mathcal{Y}_a \to \mathbf{R}$ iscalled the *canonical function associated with* Λ;
2) the problem $(\mathcal{P})$ is called *geometrically nonlinear* (respectively, *geometrically linear*) if $\Lambda : \mathcal{X} \to \mathcal{Y}$ is nonlinear (respectively, linear); it is called *physically nonlinear* (respectively, *physically linear*) if the duality mapping $D\Phi : \mathcal{X}_a \times \mathcal{Y}_a \to \mathcal{X}^*_a \times \mathcal{Y}^*_a$ is nonlinear (respectively, linear); it is called *fully nonlinear* if it is both geometrically and physically nonlinear.

The canonical transformation plays a fundamental role in duality theory of global optimization. By this definition, the governing equation (7) for fully nonlinear problems canbe written in the *tricanonical forms*, namely,

1) geometrical equation: $y = \Lambda(x)$;
2) physical relations: $(-x^*, y^*) \in \partial\Phi(x, y)$;
3) balance equation: $x^* = \Lambda^*_t(x)\, y^*$.

Since $\Lambda : \mathcal{X} \to \mathcal{Y}$ is Gâteaux differentiable, for any given $x \in \mathcal{X}$ we have the *operator decomposition*

$$\Lambda(x) = \Lambda_t(x)x + \Lambda_c(x), \tag{9}$$

where $\Lambda_c = \Lambda - \Lambda_t$ is the *complementary operator* of Λ_t. By this operator decomposition, the relation between the two bilinear forms reads

$$\langle \Lambda(x); y^* \rangle = \left\langle x, \Lambda^*_t(x)y^* \right\rangle - G(x, y^*),$$

where $G(x, y^*) = \langle -\Lambda_c(x); y^* \rangle$ is the so-called *complementary gap function*, introduced in [2]. This gap plays an important role in the canonical dual transformation methods. A framework for the fully nonlinear system is

$$\begin{array}{ccc} x \in \mathcal{X} & \leftarrow \langle x, x^* \rangle \rightarrow & \mathcal{X}^* \ni x^* \\ {\scriptstyle \Lambda_t + \Lambda_c = \Lambda} \downarrow & & \uparrow {\scriptstyle \Lambda^*_t = (\Lambda - \Lambda_c)^*} \\ y \in \mathcal{Y} & \leftarrow \langle y; y^* \rangle \rightarrow & \mathcal{Y}^* \ni y^* \end{array}$$

Extensive illustrations of the canonical transformation and the tricanonical forms in mathematical physics and variational analysis can be found in [1].

Very often, the extended canonical function Φ can be written in the form

$$\Phi(x, y) = W(y) - F(x),$$

where $F \in \Gamma(\mathcal{X})$ and $W \in \Gamma(\mathcal{Y})$ are extended canonical functions. The duality relations (5) in this special case take the forms

$$x^* \in \partial F(x), \quad y^* \in \partial W(y).$$

If $F \in \Gamma_G(\mathcal{X}_a)$ and $W \in \Gamma_G(\mathcal{Y}_a)$ are Gâteaux differentiable, the Euler equation (7) reads

$$\Lambda_t^*(\overline{x}) DW(\Lambda(\overline{x})) - DF(\overline{x}) = 0.$$

If $\Lambda : \mathcal{X} \to \mathcal{Y}$ is linear, and $W\ \mathcal{Y} \to \mathbf{R}$ is quadratic such that $DW = Cy$, where $C : \mathcal{Y} \to \mathcal{Y}^*$ is a linear operator, then the governing equations for linear system can be written as

$$\Lambda^* C \Lambda x = Ax = x^*.$$

For conservative systems, the operator $\mathrm{A} = \Lambda^*\ C\Lambda$ is usually symmetric. In static systems, C is usually positive definite and the associated total potential P is convex. However, in dynamical systems, C is indefinite and P is called the total action, which is usually a d.c. function in convex Hamilton systems.

Triality Theory

We assume that for any given nonconvex extended function $P\colon\ \mathcal{X} \to \overline{\mathbb{R}}$, there exists a general nonlinear operator $\Lambda : \mathcal{X} \to \mathcal{Y}$ and a canonical function $W \in \Gamma(\mathcal{Y})$ suchthat the canonical transformation can be written as

$$P(x) = W(\Lambda(x)) - \langle x, c \rangle, \tag{10}$$

where $c \in \mathcal{X}^*$ is a given source variable. Since $F(x) = x$, c is a linear function, the Hamiltonian $H(x, y^*) = W^*(y^*) + x$, c is a canonical function on $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}^*$ and the extended Lagrangian reads

$$L(x, y^*) = \langle \Lambda(x); y^* \rangle - W^*(y^*) - \langle x, c \rangle. \tag{11}$$

For a fixed $y^* \in \mathcal{Y}^*$, the convexity of $L(\cdot, y^*)\colon\ \mathcal{X} \to \overline{\mathbb{R}}$ depends on $\Lambda(x)$ and $y^* \in \mathcal{Y}^*$.

Let $\mathcal{Z}_a = \mathcal{X}_a \times \mathcal{Y}_a^* \subset \mathcal{Z}$ be the effectivedomain of L, and let $\mathcal{L}_c \subset \mathcal{Z}_a$ be a critical point set of L, i. e.

$$\mathcal{L}_c = \left\{(\overline{x}, \overline{y}) \in \mathcal{X}_a \times \mathcal{Y}_a^*\colon\ DL(\overline{x}, \overline{y}^*) = 0\right\}.$$

For any given critical point $(\overline{x}, \overline{y}^*) \in \mathcal{L}_c$, we let $\mathcal{X}_r \times \mathcal{Y}_r^*$ be its neighborhood such that on $\mathcal{X}_r \times \mathcal{Y}_r^*$, the pair $(\overline{x}, \overline{y}^*)$ is the only critical point of L. The following resultis of fundamental importance in global optimization.

Theorem 3 (*Triality theorem*) *Suppose that* $W \in \check{\Gamma}(\mathcal{Y}_a)$ *is convex,* $(\overline{x}, \overline{y}^*) \in \mathcal{L}_c$ *is a critical point of* L *and* $\mathcal{X}_r \times \mathcal{Y}_r^*$ *is a neighborhood of* $(\overline{x}, \overline{y}^*)$.

If $\left\langle \Lambda(x); \overline{y}^* \right\rangle$ *is convex on* $\mathcal{X}_r$, *then*

$$\begin{aligned} L(\overline{x}, \overline{y}^*) &= \min_{x \in \mathcal{X}_r} \max_{y^* \in \mathcal{Y}_r^*} L(x, y^*) \\ &= \max_{y^* \in \mathcal{Y}_r^*} \min_{x \in \mathcal{X}_r} L(x, y^*). \end{aligned} \tag{12}$$

However, if $\left\langle \Lambda(x); \overline{y}^* \right\rangle$ *is concave on* $\mathcal{X}_r$, *then either*

$$\begin{aligned} L(\overline{x}, \overline{y}^*) &= \min_{x \in \mathcal{X}_r} \max_{y^* \in \mathcal{Y}_r^*} L(x, y^*) \\ &= \min_{y^* \in \mathcal{Y}_r^*} \max_{x \in \mathcal{X}_r} L(x, y^*), \end{aligned} \tag{13}$$

or

$$\begin{aligned} L(\overline{x}, \overline{y}^*) &= \max_{x \in \mathcal{X}_r} \max_{y^* \in \mathcal{Y}_r^*} L(x, y^*) \\ &= \max_{y^* \in \mathcal{Y}_r^*} \max_{x \in \mathcal{X}_r} L(x, y^*). \end{aligned} \tag{14}$$

Since $W \in \Gamma(\mathcal{Y}_a)$ is a canonical function, we always have

$$P(x) = \operatorname{ext}\{L(x, y^*)\colon\ y^* \in \mathcal{Y}^*\}, \quad \forall x \in \mathcal{X}_k. \tag{15}$$

On the other hand, for a given Gâteaux differentiable geometrical mapping $\Lambda : \mathcal{X}_a \to \mathcal{Y}_a$, the criticality condition $D_x L(\overline{x}, y^*) = 0$ leads to the equilibrium equation

$$\Lambda_t^*(\overline{x}) y^* = c. \tag{16}$$

If there exists a subspace $\mathcal{Y}_s^* \subset \mathcal{Y}_a^*$ such that for any $y^* \in \mathcal{Y}_s^*$ and a given source variable $c \in \mathcal{X}^*$, the equation (16) can be solved for $\overline{x} = \overline{x}(y^*)$, then by the operator decomposition (9), the dual function $P^d : \mathcal{Y}_s^* \to \mathbf{R}$ can be written explicitly in the form

$$\begin{aligned} P^d(y^*) &= \operatorname{sta}\{L(x, y^*)\colon\ x \in \mathcal{X}\} \\ &= -G^d(y^*) - W^*(y^*), \quad \forall y^* \in \mathcal{Y}_s^*, \end{aligned}$$

where $G^d : \mathcal{Y}^* \to \mathbf{R}$ is the so-called *pure complementary gap function*, defined by

$$G^d(y^*) = G(\overline{x}(y^*), y^*) = -\langle \Lambda_c(\overline{x}(y^*)); y^* \rangle .$$

For any given critical point $(\overline{x}, \overline{y}^*) \in \mathcal{L}_c$, we have $G^d(\overline{y}^*) = \langle \overline{x}, c \rangle - \langle \Lambda(\overline{x}(\overline{y}^*)); \overline{y}^* \rangle$. Thus, the Legendre duality relations among the canonical functions W and W^* lead to

$$P(\overline{x}) - P^d(\overline{y}^*) = 0, \quad \forall (\overline{x}, \overline{y}^*) \in \mathcal{L}_c. \tag{17}$$

This identity shows that there is no duality gap between the nonconvex function P and its canonical dual function P^d. Actually the duality gap, which exists in classical duality theories, is now recovered by the complementary gap function $G(\overline{x}, \overline{y}^*)$.

Theorem 4 (*Triduality theorem*) *Suppose that $W \in \check{\Gamma}(\mathcal{Y}_a)$ is a critical point of L and $\mathcal{X}_r \times \mathcal{Y}_r^*$ is a neighborhood of $(\overline{x}, \overline{y}^*)$. If $\langle \Lambda(x); \overline{y}^* \rangle$ is convex on $\mathcal{X}_r$, then*

$$P(\overline{x}) = \min_{x \in \mathcal{X}_r} P(x) \Leftrightarrow P^d(\overline{y}^*) = \max_{y^* \in \mathcal{Y}_r^*} P^d(y^*).$$

However, if $\langle \Lambda(x); \overline{y}^* \rangle$ is concave on $\mathcal{X}_r$, then

$$P(\overline{x}) = \min_{x \in \mathcal{X}_r} P(x) \Leftrightarrow P^d(\overline{y}^*) = \min_{y^* \in \mathcal{Y}_r^*} P^d(y^*);$$
$$P(\overline{x}) = \max_{x \in \mathcal{X}_r} P(x) \Leftrightarrow P^d(\overline{y}^*) = \max_{y^* \in \mathcal{Y}_r^*} P^d(y^*).$$

Example 5 We now illustrate the application of the interesting triduality theory for solving the following nonconvex optimization problem in $\mathcal{X} = \mathbf{R}^n$,

$$P(x) = \frac{a}{2}\left(\frac{1}{2}\|Ax\|^2 - \mu\right)^2 - x^\top c \to \text{sta}, \quad \forall x,$$

where a, $\mu > 0$ are given parameters, $c \in \mathbf{R}^n$ is a given vector, and $A : \mathbf{R}^n \to \mathbf{R}^m$ is a matrix. The Euler equation associated with this nonconvex stationary problem is a nonlinear algebraic equation in $\mathbf{R}^n$

$$a\left(\frac{1}{2}\|A\overline{x}\|^2 - \mu\right)C\overline{x} = c,$$

where $C = A^\intercal A = C^\intercal \in \mathbf{R}^{nn}$. We are interested in finding all the critical points of P. To set this nonconvex problem in our framework, we let $\mathcal{X} = \mathbf{R}^n = \mathcal{X}^*$, and $\Lambda : \mathbf{R}^n \to \mathcal{Y} = \mathrm{R}$ a quadratic operator

$$y = \Lambda(x) = \frac{1}{2}\|Ax\|^2 - \mu = \frac{1}{2}x^\top Cx - \mu.$$

Since $F(x) = \langle x, c \rangle = x^\intercal c$ is a linear function on $\mathbf{R}^n$, the admissible space $\mathcal{X}_a = \mathcal{X} = \mathbf{R}^n$. By the fact that $x^* = DF(x) = c$, the range for the canonical mapping $DF : \mathcal{X} \to \mathcal{X}^* = \mathbf{R}$ is a hyperplane in $\mathbf{R}^n$, i. e.

$$\mathcal{X}_a^* = \{x^* \in \mathbb{R}^n : \ x^* = c\}.$$

The feasible set for the primal problem is $\mathcal{X}_k = \{x \in \mathcal{X}_a : \Lambda(x) \in \mathcal{Y}_a\} = \mathbf{R}^n$.

By the fact that $x^\intercal Cx \geq 0$, $\forall x \in \mathcal{X}_a = \mathcal{X} = \mathbf{R}^n$, the range for the geometrical mapping Λ: $\mathcal{X}_a \to \mathbf{R}$ is a closed convex set in $\mathbf{R}$

$$\mathcal{Y}_a = \{y \in \mathbb{R} : \ y \geq -\mu\} \subset \mathcal{Y} = \mathbb{R}.$$

On the admissible subset $\mathcal{Y}_a \subset \mathcal{Y} = \mathbf{R}$, the canonical function $W(y) = (1/2)ay^2$ is quadratic. The range for the constitutive mapping DW: $\mathcal{Y}_a \to \mathcal{Y}^* = \mathbf{R}$ is also a closed convex set in $\mathbf{R}$,

$$\mathcal{Y}_a^* = \{y^* \in \mathbb{R} : \ y^* \geq -a\mu\}.$$

On $\mathcal{Y}_a^*$, the Legendre conjugate of W isalso strictly convex

$$W^*(y^*) = \frac{1}{2}a^{-1}y^{*2}, \tag{18}$$

and the Legendre duality relations hold on $\mathcal{Y}_a \times \mathcal{Y}_a^*$.

On $\mathcal{X}_a \times \mathcal{Y}_a^* = \mathbf{R}^n \times \mathbf{R}$, the extended Lagrangian in this case reads

$$L(x, y^*) = \frac{1}{2}y^* x^\top Cx - \mu y^* - \frac{1}{2}a^{-1}y^{*2} - x^\top c.$$

It is easy to check that the dual function associated with L is

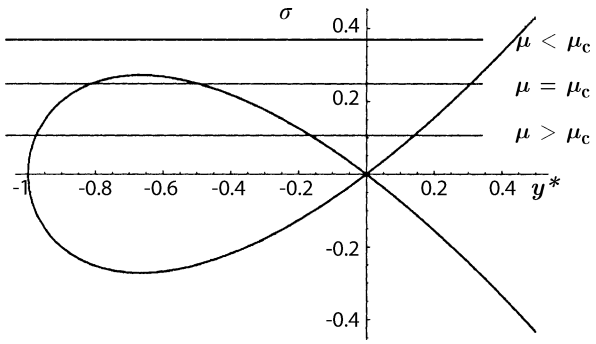
$$P^d(y^*) = \frac{1}{2}(y^*)^{-1}c^\top Cc - \mu y^* - \frac{1}{2a}y^{*2}.$$

The dual Euler–Lagrange equation is an algebraic equation in $\mathbf{R}$:

$$(\mu + a^{-1}y^*)y^{*2} = \frac{1}{2}\sigma^2, \tag{19}$$

where $\sigma^2 = c^\intercal Cc$ is a constant. Since $C \in \mathbf{R}^{nn}$ is positive definite, this equation holds only on $\mathcal{Y}_a^*$.

In algebraic geometry, the dual Euler–Lagrange equation (19) is the so-called singular algebraic curve in (y^*, σ)-space (see Fig. 1). For a given parameter μ and c



Duality Theory: Triduality in Global Optimization, Figure 1
Singular algebraic curve

$\in \mathbf{R}^n$, this dual equation has at most three real roots $y_k^* \in \mathcal{Y}_a^*$, $k = 1, 2, 3$, which leads to the primal solution

$$x_k = y_k^* C^+ c, \quad k = 1, 2, 3,$$

where C^+ stands for the generalized inverse of C. We know that each (x_k, y_k^*) is a critical point of L and

$$P(x_k) = L(x_k, y_k^*) = P^d(y_k^*), \quad k = 1, 2, 3.$$

In the case of $n = 1$, the cost function

$$P(x) = \frac{1}{2} a \left(\frac{1}{2} x^2 - \mu \right)^2 - cx$$

is a *double-well function* (see Fig. 2, solid line), which appears in many physical systems. The graph of the canonical dual function

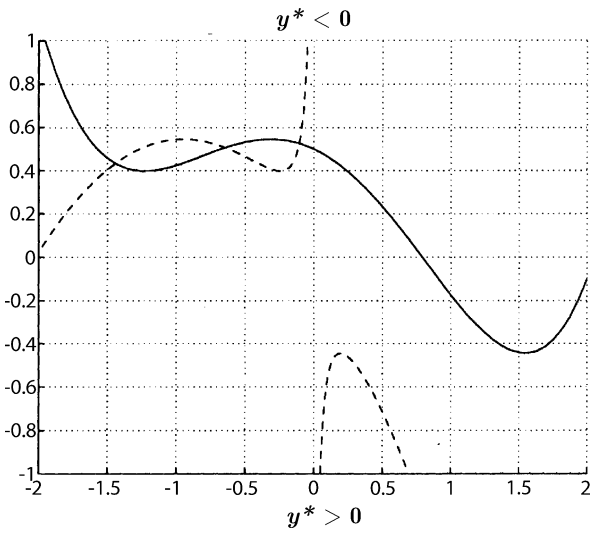
$$P^d(y^*) = \frac{1}{2} \frac{c^2}{y^*} - \mu y^* - \frac{y^{*2}}{2a}$$

has two branches (Fig. 2, dashed line). It is easy to prove (see [1]) that if $\mu > \mu_c = 1.5\ (\sigma/a)^{2/3}$, the dual Euler–Lagrange equation (19) has three roots $y_1^* > 0 > y_2^* > y_3^*$, corresponding to three critical points of P^d (see Fig. 2). Then, y_1^* is a global maximizer of P^d, $x_1 = \sigma/y_1^*$ is a global minimizer of P, P^d takes local minimum and local maximum values at y_2^* and y_3^*, respectively, $x_2 = \sigma/y_2^*$ is a local maximizer of P, while $x_3 = \sigma/y_3^*$ is a local minimizer.

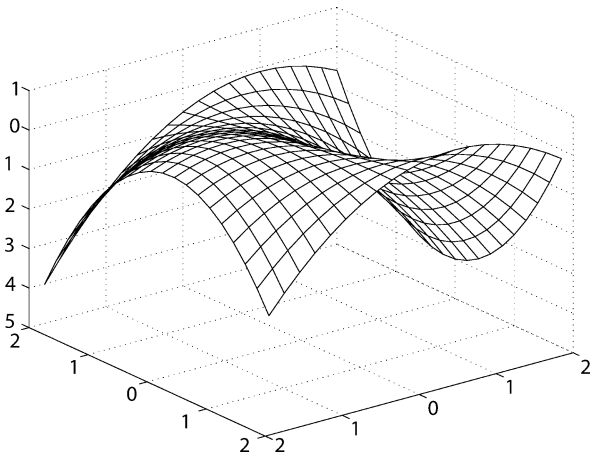
The Lagrangian associated with this double-well energy is

$$L(x, y^*) = \frac{1}{2} x^2 y^* - \left(\frac{1}{2a} y^{*2} + \mu y^* \right) - y^* x.$$

It is a saddle function for $y^* > 0$. If $y^* < 0$, it is a supercritical point function (see Fig. 3).



Duality Theory: Triduality in Global Optimization, Figure 2
Graphs of $P(u)$ and itsdual $P^d(y^*)$



Duality Theory: Triduality in Global Optimization, Figure 3
Lagrangian for the double-well energy

See also

- ▶ Duality Theory: Biduality in Nonconvex Optimization
- ▶ Duality Theory: Monoduality in Convex Optimization
- ▶ History of Optimization
- ▶ Von Neumann, John

References

1. Gao DY (1999) Duality principles in non-convex systems: Theory, methods and applications. Kluwer, Dordrecht

2. Gao DY, Strang G (1989) Geometrical nonlinearity: Potential energy, complementary energy, and the gap functional. Quart Appl Math XLVII(3):487–504

Dykstra's Algorithm and Robust Stopping Criteria

Ernesto G. Birgin[1], Marcos Raydan[2]
[1] Department of Computer Science IME-USP, University of São Paulo, São Paulo, Brazil
[2] Departamento de Computación, Facultad de Ciencias, Universidad Central de Venezuela, Caracas, Venezuela

MSC2000: 90C25, 65K05, 65G505

Article Outline

Keywords
Introduction
Formulations
Dykstra's Algorithm
Difficulties with some Commonly Used Stopping Criteria
Robust Stopping Criteria
References

Keywords

Convex optimization; Alternating projection methods; Dykstra's algorithm; Stopping criteria

Introduction

We consider the application of Dykstra's algorithm for solving the following optimization problem

$$\min_{x\in\Omega} \|x^0 - x\| \,, \tag{1}$$

where x^0 is a given point, Ω is a closed and convex set, and $\|z\|^2 = \langle z, z\rangle$ defines a real inner product in the space. The solution x^* is called the projection of x^0 onto Ω and is denoted by $P_\Omega(x^0)$. Dykstra's algorithm for solving (1) has been extensively studied since it fits in many different applications (see [5,21,22,23,27,28,29,32,24,42,42,45]).

For simplicity, we consider the case

$$\Omega = \cap_{i=1}^{p} \Omega_i \,, \tag{2}$$

where Ω_i are closed and convex sets in $\mathbb{R}^n$, for $i = 1, 2, \ldots, p$, and $\Omega \neq \emptyset$. Moreover, we assume that for any $z \in \mathbb{R}^n$ the calculation of $P_\Omega(z)$ is not trivial; whereas, for each Ω_i, $P_{\Omega_i}(z)$ is easy to obtain as in the case of a box, an affine subspace, or a sphere. For the not feasible case (i. e., when $\Omega = \emptyset$) the behavior of Dykstra's algorithm is treated in [2,6,37].

Dykstra's alternating projection algorithm is a cyclic scheme for finding asymptotically the projection of a given point onto the intersection of a finite number of closed convex sets. Roughly speaking, it iterates by projecting in a clever way onto each of the convex sets individually. The algorithm was originally proposed by Dykstra [20] for closed and convex cones in the Euclidean space $\mathbb{R}^n$, and later extended by Boyle and Dykstra [7] for closed and convex sets in a Hilbert space. It was rediscovered by Han [30] using duality theory, and the linear rate of convergence was established by Deutsch and Hundal [18] for the polyhedral case (see also [19,43,44]).

Dykstra's algorithm belongs to the general family of alternating projection methods, that dates back to von Neumann [46] who treated the problem of finding the projection of a given point in a Hilbert space onto the intersection of two closed subspaces. Later, Cheney and Goldstein [15] extended the analysis of von Neumann's alternating projection scheme to the case of two closed and convex sets. In particular, they established convergence under mild assumptions. However, the limit point need not be the closest in the intersection. Therefore, the alternating projection method, proposed by von Neumann, is not useful for problem (1). Fortunately, Dykstra [20] found the clever modification of von Neumann's scheme for which convergence to the solution point is guaranteed. For a complete discussion on alternating projection methods see Deutsch [17].

Dykstra's algorithm has been extended in several different ways. Gaffke and Mathar [24] proposed, via duality, a family of simultaneous Dykstra's algorithm in Hilbert space. Later Iusem and De Pierro [37] established the convergence of the simultaneous version considering also the inconsistent case in the Euclidean space $\mathbb{R}^n$. Bauschke and Borwein [2] further analyzed Dykstra's algorithm for two sets, that appears frequently in applications and in particular generalized the results in [37]. In [36] it was established that for linear inequality constraints the method of Dykstra re-

duces to the method proposed by Hildreth [33] in his pioneer work on dual alternating projections. See also [40] for further analysis and extensions.

Dykstra's algorithm has also been generalized by Deutsch and Hundal [35] to an infinite family of sets, and also to allow a random ordering, instead of cyclic, of the projections onto the closed convex sets. More recently, it has also been generalized by Bregman et al. [9] to avoid the projection onto each one of the convex sets in every cycle. Instead, projections onto either a suitable half space of the intersection of two half spaces are used. Further results concerning the connection between Bregman distances and Dykstra's algorithm can be found in [3,4,8,14]. For the advantages of projecting cyclically onto suitable half spaces, see the previous work by Iusem and Svaiter [38,39].

A computational experiment comparing Dykstra's algorithm and the Halpern-Lions-Wittmann-Bauschke algorithm [1] on linear best approximation test problems can be found in [12].

Formulations

Dykstra's Algorithm

Dykstra's algorithm solves (1), (2) by generating two sequences: the iterates $\{x_i^k\}$ and the increments $\{y_i^k\}$. These sequences are defined by the following recursive formulae:

$$\begin{aligned} x_0^k &= x_p^{k-1}, \\ x_i^k &= P_{\Omega_i}(x_{i-1}^k - y_i^{k-1}), \qquad i = 1, 2, \ldots, p\,, \\ y_i^k &= x_i^k - (x_{i-1}^k - y_i^{k-1}), \qquad i = 1, 2, \ldots, p\,, \end{aligned} \tag{3}$$

for $k = 1, 2, \ldots$ with initial values $x_p^0 = x^0$ and $y_i^0 = 0$ for $i = 1, 2, \ldots, p$.

Remarks

1. For the sake of simplicity, the projecting control index $i(k)$ used in (3) is the most common one: $i(k) = k \bmod p + 1$, for all $k \geq 0$. However, more advanced control indices can also be used, as long as they satisfy some minimal theoretical requirements (see e. g., [35]).
2. The increment y_i^{k-1} associated with Ω_i in the previous cycle is always subtracted before projecting onto Ω_i. Only one increment (the last one) for each Ω_i needs to be stored.
3. If Ω_i is a closed affine subspace, then the operator P_{Ω_i} is linear and it is not required, in the kth cycle, to subtract the increment y_i^{k-1} before projecting onto Ω_i. Thus, for affine subspaces, Dykstra's procedure reduces to the alternating projection method of von Neumann [46].
4. For $k = 1, 2, \ldots$ and $i = 1, 2, \ldots, p$, it is clear from (3) that the following relations hold

$$x_p^{k-1} - x_1^k = y_1^{k-1} - y_1^k\,, \tag{4}$$

$$x_{i-1}^k - x_i^k = y_i^{k-1} - y_i^k\,, \tag{5}$$

where $x_p^0 = x^0$ and $y_i^0 = 0$, for all $i = 1, 2, \ldots, p$. For the sake of completeness we now present the key theorem associated with Dykstra's algorithm.

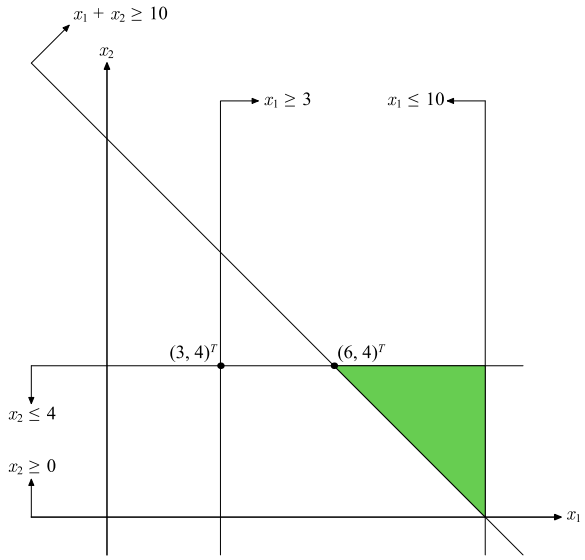
Theorem 1 Boyle and Dykstra, 1986 [7] *Let $\Omega_1, \ldots, \Omega_p$ be closed and convex sets of $\mathbb{R}^n$ such that $\Omega = \cap_{i=1}^p \Omega_i \neq \emptyset$. For any $i = 1, 2, \ldots, p$ and any $x^0 \in \mathbb{R}^n$, the sequence $\{x_i^k\}$ generated by (3) converges to $x^* = P_\Omega(x^0)$ (i. e., $\|x_i^k - x^*\| \to 0$ as $k \to \infty$).*

We now discuss the delicate issue of stopping Dykstra's algorithm within a certain previously established tolerance that indicates the distance of the current iterate to the unique solution.

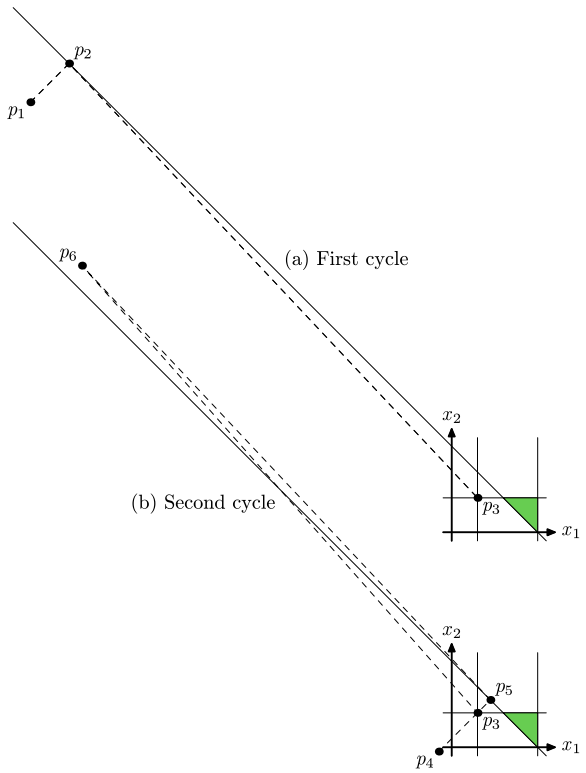
Difficulties with some Commonly Used Stopping Criteria

In some applications it is possible to obtain a *somehow* natural stopping rule, associated with the problem at hand. For example, when solving a linear system, $Ax = b$, by alternating projection methods [10,25], the residual vector ($r(x) = b - Ax$) is usually available and yields some interesting and robust stopping rules. Another example appears in image reconstruction for which a *good and feasible* image tells the user that it is time to stop the process [13,16]. Similar circumstances are present in some other specific applications (e. g. saddle point problems [31], and molecular biology [28,29]).

However, in general, this is not the case, and we are left with the information produced only by the internal computations, i. e., the sequence of iterates and perhaps the sequence of increments, and some inner products. For this general case, a popular stopping rule



Dykstra's Algorithm and Robust Stopping Criteria, Figure 1
Feasible set $\Omega = \Omega_1 \cap \Omega_2$ in $\mathbb{R}^2$



Dykstra's Algorithm and Robust Stopping Criteria, Figure 2
First two cycles of Dykstra's algorithm to find the projection of $x^0 = (-49, 50)^T$ onto $\Omega = \Omega_1 \cap \Omega_2$

is to monitor the subsequence of projections onto one particular convex set, Ω_i, and stop the process when the distance, in norm, of two consecutive projections is less than or equal to a previously established tolerance [26,27,32,41].

Another commonly used criterion, that is claimed to improve the previous one (e. g. [7,22,28,45]) is to somehow compute an average of all the projections at each cycle of projections, and then stop the process when the distance, in norm, of two consecutive of those average projections is less than or equal to a previously established tolerance.

Finally, we would like to mention that another criterion, that is also designed to improve any of the two criteria above, is to check any of the previously described rules during N consecutive cycles, where N is a fixed positive integer.

None of these stopping rules is a trustable choice. In [6], Birgin and Raydan presented the example below to establish that they can fail even for a two dimensional problem. (see Figs. 1 and 2).

Consider the closed and convex set $\Omega = \Omega_1 \cap \Omega_2$, where $\Omega_1 = \{x \in \mathbb{R}^2 \mid x_1 + x_2 \geq 10\}$ is a half space and $\Omega_2 = \{x \in \mathbb{R}^2 \mid 3 \leq x_1 \leq 10, 0 \leq x_2 \leq 4\}$ is a box. This closed and convex set in $\mathbb{R}^2$ is shown in Fig. 1.

Let $x^0 = (-49, 50)^T$ and let us use Dykstra's algorithm to find the closest point to x^0 in Ω. In Fig. 2 we can see the first two cycles of this convergent process. Since $y_1^0 = y_2^0 = 0$ (null initial increments) then for the first cycle we project x^0 onto Ω_1 to obtain $p_2 = x_1^1 = (-44.5, 54.5)^T$ and then we project p_2 onto Ω_2 to obtain $p_3 = x_2^1 = (3, 4)^T$. For the second cycle, the increments are not null ($y_1^1 = (4.5, 4.5)^T$ and $y_2^1 = (47.5, -50.5)^T$), and we start from p_3. First we project $p_4 = p_3 - y_1^1$ onto Ω_1 to obtain $p_5 = x_1^2$. Then we project $p_6 = p_5 - y_2^1$ onto Ω_2 to obtain p_3 again. Hence $x_2^2 = x_2^1$. The increment associated with Ω_2 is large enough to take the iterate back to the quadrant where the projection onto the box is again p_3. As discussed in [6], this phenomenon will occur until cycle 32, i. e., $p_3 = x_2^1 = x_2^2 = \cdots = x_2^{32}$.

Moreover, by choosing x^0 far enough, this misleading event can be repeated for as many cycles as any previously established positive integer N. Eventually the size of the increments will be reduced and convergence to x^* will be observed.

Robust Stopping Criteria

After a close inspection of the proof of the Boyle and Dykstra's theorem, Birgin and Raydan [6] proposed some robust stopping criteria for Dykstra's algorithm. For that they first established the following result.

Theorem 2 *Let x^0 be any element of $\mathbb{R}^n$. Consider the sequences $\{x_i^k\}$ and $\{y_i^k\}$ generated by (3) and define c^k as*

$$c^k = \sum_{m=1}^{k}\sum_{i=1}^{p} \|y_i^{m-1} - y_i^m\|^2 + 2\sum_{m=1}^{k-1}\sum_{i=1}^{p}\langle y_i^m, x_i^{m+1} - x_i^m\rangle . \quad (6)$$

Then, in the kth cycle of Dykstra's algorithm,

$$\|x^0 - x^*\|^2 \geq c^k . \quad (7)$$

Moreover, at the limit when k goes to infinity, equality is attained in (7).

Based on the previous theorem, let us now write c^k as follows:

$$c^k = c_L^k + c_S^k ,$$

where

$$c_L^k = \sum_{m=1}^{k} c_I^m , \quad (8)$$

$$c_I^m = \sum_{i=1}^{p} \|y_i^{m-1} - y_i^m\|^2 \quad (9)$$

and

$$c_S^k = 2\sum_{m=1}^{k-1}\sum_{i=1}^{p}\langle y_i^m, x_i^{m+1} - x_i^m\rangle .$$

Both c_L^k and c_S^k are monotonically nondecreasing by definition. Moreover in [6], the following theorem is also established.

Theorem 3 *Consider the sequences $\{x_i^k\}$ and $\{y_i^k\}$ generated by (3), and c^k, c_L^k and c_I^k as defined in (6), (8) and (9), respectively. For any $k \in \mathbb{N}$, if $x^k \neq x^*$ then $c_I^{k+1} > 0$ and, hence, $c_L^k < c_L^{k+1}$ and $c^k < c^{k+1}$.*

The results established in Theorems 2 and 3 are combined in [6] to propose robust stopping criteria. Notice that $\{c_L^k\}$ and $\{c^k\}$ are monotonically increasing and convergent, and also that $\{c_I^k\}$ converges to zero. Therefore we can stop the process when

$$c_I^k = \sum_{i=1}^{p} \|y_i^{k-1} - y_i^k\|^2 \leq \varepsilon$$

or, similarly, when

$$c^k - c^{k-1} = c_I^k + 2\sum_{i=1}^{p}\langle y_i^{k-1}, x_i^k - x_i^{k-1}\rangle \leq \varepsilon , \quad (10)$$

where $\varepsilon > 0$ is a sufficiently small tolerance. As c^k may grow fast, computing $c^k - c^{k-1}$ may give inaccurate results due to loss of accuracy in floating point representation and, hence, cancellation. So, for the criterion in (10), it is recommendable to test convergence with the second expression.

The computation of c_I^k involves the squared-norm $\|y_i^{k-1} - y_i^k\|^2$, for $i = 1, 2, \ldots, p$. By (5), $y_i^k = y_i^{k-1} + v$, where $v = x_i^k - x_{i-1}^k$ is a temporary n-dimensional array needed in the computation of Dykstra's algorithm. So, the computational cost involved in the calculation of c_I^k is just the cost of the extra inner product $\langle v, v\rangle$ at each iteration.

The computation of c^k involves the calculation of c_I^k plus an extra term. The computational of this extra term is also small and involves an inner product and the difference of two vectors per iteration. But, in contrast with the computation of c_I^k which does not require additional savings, the computation of the extra term requires to save p extra n-dimensional arrays (the same amount of memory required in Dykstra's algorithm to save the increments). So, the computation of c^k requires some additional calculations and memory savings, and hence it is more expensive. However, it also has the advantage of revealing the optimal distance: $\|x^0 - x^*\|^2$, that could be of interest in some applications.

We close this section with some comments concerning the behavior of the stopping criteria when the problem is not feasible. In this case ($\Omega = \emptyset$), there is no solution and we know from Theorem 3 that the sequences $\{c_L^k\}$ and $\{c^k\}$ are monotonically increasing. Moreover, under some mild assumptions on the sets Ω_i, the sequences $\{x_i^k\}$ converge for $1 \leq i \leq p$, and there exists a real constant $\delta > 0$ such that $\sum_{i=1}^{p} \|x_{i-1}^k - x_i^k\|^2 \geq \delta$ for all k. A discussion on this topic is presented in [2, Section 6], including a notion

of *distance* between all the sets Ω_i (see also [37]). Now using (5), we obtain

$$\sum_{i=1}^{p} \|x_{i-1}^k - x_i^k\|^2 = \sum_{i=1}^{p} \|y_i^{k-1} - y_i^k\|^2 = c_I^k \,.$$

Therefore, the sequence $\{c_I^k\}$ remains bounded away from zero, whereas $\{c_L^k\}$ and $\{c^k\}$ tend to infinity. Consequently, none of the proposed stopping criteria will be satisfied for any k, as expected.

References

1. Bauschke HH (1996) The approximation of fixed points of compositions of nonexpansive mappings in Hilbert space. J Math Anal Appl 202:150–159
2. Bauschke HH, Borwein JM (1994) Dykstra's alternating projection algorithm for two sets. J Approx Theory 79:418–443
3. Bauschke HH, Borwein JM (1997) Legendre functions and the method of random Bregman projections. J Convex Anal 4:27–67
4. Bauschke HH, Lewis AS (2000) Dykstra's algorithm with Bregman projections: a convergence proof. Optim 48:409–427
5. Birgin EG, Martínez JM, Raydan M (2003) Inexact spectral gradient method for convex-constrained optimization. IMA J Numer Anal 23:539–559
6. Birgin EG, Raydan M (2005) Robust stopping criteria for Dykstra's algorithm. SIAM J Sci Comput 26:1405–1414
7. Boyle JP, Dykstra RL (1986) A method for finding projections onto the intersection of convex sets in Hilbert spaces. Lect Notes Stat 37:28–47
8. Bregman LM, Censor Y, Reich S (1999) Dykstra's algorithm as the nonlinear extension of Bregman's optimization method. J Convex Anal 6:319–333
9. Bregman LM, Censor Y, Reich S, Zepkowitz-Malachi Y (2003) Finding the projection of a point onto the intersection of convex sets via projections onto halfspaces. J Approx Theory 124:194–218
10. Bramley R, Sameh A (1992) Row projection methods for large non symmetric linear systems. J Sci Statist Comp 13:168–193
11. Bregman LM, Censor Y, Reich S (1999) Dykstra's algorithm as the nonlinear extension of Bregman's optimization method. J Convex Anal 6:319–333
12. Censor Y (2006) Computational acceleration of projection algorithms for the linear best approximation problem. Linear Algebr Appl 416:111–123
13. Censor Y, Herman GT (1987) On some optimization techniques in image reconstruction from projections. Appl Numer Math 3:365–391
14. Censor Y, Reich S (1998) The Dykstra's algorithm with Bregman projections. Commun Appl Anal 2:407–419
15. Cheney W, Goldstein A (1959) Proximity maps for convex sets. Proc Am Math Soc 10:448–450
16. Combettes PL (1996) The convex feasibility problem in image recovery. Adv Imag Electron Phys 95:155–270
17. Deutsch F (2001) Best Approximation in Inner Product Spaces. Springer, New York
18. Deutsch F, Hundal H (1994) The rate of convergence of Dykstra's cyclic projections algorithm: the polyhedral case. Numer Funct Anal Optim 15:537–565
19. Deutsch F (1995) Dykstra's cyclic projections algorithm: the rate of convergence. In: Singh SP (ed) Approximation Theory, Wavelets and Applications. Kluwer, Dordrecht, pp 87–94
20. Dykstra RL (1983) An algorithm for restricted least-squares regression. J Am Stat Assoc 78:837–842
21. Eberle MG, Maciel MC (2003) Finding the Closest Toeplitz Matrix. Comput Appl Math 22:1–18
22. Escalante R, Raydan M (1996) Dykstra's Algorithm for a Constrained Least-Squares Matrix Problem. Numer Linear Algebr Appl 3:459–471
23. Escalante R, Raydan M (1998) On Dykstra's algorithm for constrained least-squares rectangular matrix problems. Comput Math Appl 35:73–79
24. Gaffke N, Mathar R (1986) A cyclic projection algorithm via duality. Metr 36:29–54
25. Garcia-Palomares UM (1999) Preconditioning projection methods for solving algebraic linear systems. Numer Algorithm 21:157–164
26. Glunt W (1995) An alternating projection method for certain linear problems in a Hilbert space. IMA J Numer Anal 15:291–305
27. Glunt W, Hayden TL, Hong S, Wells J (1990) An alternating projection algorithm for computing the nearest Euclidean distance matrix. SIAM J Matrix Anal Appl 11: 589–600
28. Glunt W, Hayden TL, Raydan M (1993) Molecular conformations from distance matrices. J Comput Chem 14:114–120
29. Glunt W, Hayden TL, Liu WM (1991) The embedding problem for predistance matrices. Bull Math Biol 53:769–796
30. Han SP (1988) A successive projection method. Math Program 40:1–14
31. Hernández-Ramos LM (2005) Alternating oblique projections for coupled linear systems. Numer Algorithm 38:285–303
32. Higham N (2002) Computing the nearest correlation matrix – a problem from finance. IMA J Numer Anal 22:329–343
33. Hildreth C (1957) A quadratic programming procedure. Naval Res Log Quart 4:79–85
34. Hu H, Olkin I (1991) A numerical procedure for finding the positive definite matrix closest to a patterned matrix. Stat Probab Lett 12:511–515
35. Hundal H, Deutsch F (1997) Two generalizations of Dykstra's cyclic projections algorithm. Math Program 77:335–355

36. Iusem A, De Pierro A (1990) On the convergence properties of Hildreth's quadratic programming algorithm. Math Prog 47:37–51
37. Iusem A, De Pierro A (1991) On the convergence of Han's method for convex programming with quadratic objective. Math Prog 52:265–284
38. Iusem A, Svaiter BF (1995) A row-action method for convex programming. J Math Prog 64:149–171
39. Iusem A, Svaiter BF (1995) Primal-dual row-action method for convex programming. J Optim Theor Appl 86:73–112
40. Lent A, Censor Y (1980) Extensions of Hildreth's row-action method for quadratic programming. SIAM J Control Optim 18:444–454
41. Mendoza M, Raydan M, Tarazaga P (1998) Computing the nearest diagonally dominant matrix. Numer Linear Algebr Appl 5:461–474
42. Monsalve M, Moreno J, Escalante R, Raydan M (2003) Selective alternating projections to find the nearest SSD+ matrix. Appl Math Comput 145:205–220
43. Morillas PM (2005) Dykstra's algorithm with strategies for projecting onto certain polyhedral cones. Appl Math Comput 167:635–649
44. Perkins C (2002) A Convergence Analysis of Dykstra's Algorithm for Polyhedral Sets. SIAM J Numer Anal 40:792–804
45. Raydan M, Tarazaga P (2002) Primal and polar approach for computing the symmetric diagonally dominant projection. Numer Linear Algebr Appl 9:333–345
46. von Neumann J (1950) Functional operators vol. II. The geometry of orthogonal spaces. Annals of Mathematical Studies 22, Princeton University Press. This is a reprint of mimeographed lecture notes first distributed in 1933

Dynamic Programming: Average Cost Per Stage Problems

Ioannis P. Androulakis
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L99

Article Outline

Keywords
See also
References

Keywords

Dynamic programming; Infinite horizon problems; Average cost per stage

Dynamic programming deals with optimal decision making problems in the presence of uncertainty that addresses systems in which events occur sequential. In general the state transitions are described by stationary dynamic systems of the form:

$$x_{k+1} = f(x_k, u_k, \omega_k), \quad k = 0, \ldots,$$

where for each time instance (stage) k, the state of the system is an element of the space S, the control action u that is to be implemented so as to achieve optimality belong to a space C, and finally the uncertainty is modeled through a set of random disturbances ω that belong to a *countable set* D. Furthermore, it is assumed that the control u_k is constrained to take values in a given nonempty set $U(x_k) \in C$, which depends of the current state x_k. The random disturbances ω_k, $k = 0, \ldots,$ have identical statistics and the probability distributions $\mathsf{P}(\cdot|x_k, u_k)$ are defined on D. These may depend explicitly on x_k and u_k but not on prior disturbances. Given an initial state x_0, we seek a policy π such that $\pi = \{\mu_0, \mu_1, \ldots\}$ for which:

$$\mu_k : S \to C \to, \quad \mu_k(x_k) \in U(x_k), \quad \forall x_k \in S,$$

that minimizes a cost function defined as:

$$J_\pi(x_0) = \lim_{N\to\infty} \mathsf{E}\left\{ \sum_{k=1}^{N-1} \alpha^k g(x_k, \mu_k(x_k), \omega_k) \right\}.$$

The function $g()$ is the cost per stage such that: g: $S \times C \times D \to \mathbf{R}$ and is assumed to be given. Finally, the parameter α is termed *discount factor* and it holds that: $0 < \alpha \leq 1$. We denote by Π the set of all admissible policies $\pi = \{\mu_0, \mu_1, \ldots\}$, that is the set of all sequences of such functions for which:

$$\mu_k : S \to C, \quad \mu_k(x_k) \in U(x_k), \quad \forall x_k \in S.$$

The optimal cost function J^* is then defined as:

$$J^* = \min_{\pi\in\Pi} J_\pi(x), \quad x \in S.$$

An admissible policy of the form $\pi = \{\mu, \mu, \ldots\}$ is termed *stationary* and its corresponding cost is J_μ.

When studying problems of this kind the assumption is made that either the discount factor is $\alpha < 1$ (*discounted problems*, [3]) or that naturally there exists a special cost-free absorbing state (*stochastic shortest path problems*, [3]). In either of these two cases the

total expected cost is finite, and the minimization, the way it was previously stated, is well defined. In situations where either the discount factor is 1, or a terminal state does not exist it is more meaningful to optimize an average expected cost as:

$$J_\pi(x_0) = \lim_{N\to\infty} \frac{1}{N} \mathsf{E}\left\{\sum_{k=0}^{N-1} g(x_k, u_k, \omega_k)\right\}.$$

The problems are known as *average cost per stage problems* and although they bare various similarities with both discounted and stochastic shortest path problems, they have some distinct characteristics. One of the earliest studies was that of D. Blackwell, [7], which made the connection between the optimal average return and the optimal return for values of $\alpha \to 1$. Precisely these characteristics make the analysis of average cost per stage problems the target of intense research, [1]. Connections are made, for developing the associated theory, with both the associated discounted problem, but also recently with an associated stochastic shortest path problem, [4].

Since the theory for analyzing average cost dynamic programming problems has been largely based on the associated theory for discounted and stochastic shortest path problems, most of the results and computational methods bare major similarities. As a prelude to what follows, it should be pointed out that for the average cost per stage problems:

1) the optimal average cost per stage is independent of the initial state for most problems;
2) *Bellman's equation* will take a slightly modified form that would include *differential cost* for each state;
3) there exist computational analogues of all methods developed for either discounted or stochastic shortest path problems.

The cost function of average cost per stage problems are closely related the associated α-discounted problem for a given stationary policy as follows:

- For any stationary policy μ and for any $\alpha \in (0, 1)$ we have:

  $$J_{\alpha,\mu} = (1-\alpha)^{-1} J_\mu + h_\mu + O(|1-\alpha|),$$

  where

  $$J_\mu = \left(\lim_{n\to\infty} \frac{1}{N} \sum_{k=1}^{N-1} P_\mu^k\right) g_\mu$$

  is the average cost corresponding to policy μ, for a process with a transition probability matrix P_μ and costs g_μ. The matrix O is such that: $\lim_{\alpha\to 1} O(|1-\alpha|) = 0$, and the vector h_μ satisfies: $J_\mu + h_\mu = g_\mu + P_\mu h_\mu$.

In the above, the matrix P_μ is the *transition probability matrix* for a given stationary policy μ, given by:

$$P_\mu = \begin{pmatrix} p_{11}(\mu(1)) & \cdots & p_{1n}(\mu(1)) \\ \cdots & \cdots & \cdots \\ p_{n1}(\mu(1)) & \cdots & p_{nn}(\mu(n)) \end{pmatrix}$$

and g_μ the associated cost vector:

$$g_\mu = \begin{pmatrix} g(1, \mu(1)) \\ \cdots \\ g(n, \mu(n)) \end{pmatrix}.$$

The vector h_μ is termed *differential cost vector*, and it represents the difference in N-stage expected optimal cost due to starting at stage i rather than starting at stage j. The key optimality results irrespective of initial states is based on ideas first formulated in [6]. An important element of this analysis is that of a *unichain policy*. Given a *stationary-state Markov chain*, [10], the subset of states that communicate, i. e., there exist transitions k_1 and k_2 for which state transitions probabilities $p_{ij}^{k_1}$ and $p_{ji}^{k_2}$ are positive, is termed a *recurrent class of states*. States that do not belong to a recurrent class are termed *transient*. A stationary policy whose associated Markov chain has a single recurrent class and a possibly empty set of transient states is called *unichain*. In view of the above, the form of the Bellman's equation for characterizing an optimal policy, [9], for the average cost per stage problem takes the following from:

- Assume that any of the following conditions hold:

  1) Every policy that is optimal within the class of stationary policies is unichain.
  2) For every two states i and j, there exists a stationary policy $\pi(i, j)$, such that for some k:

     $$\mathsf{P}(x_k = j | x_0 = i,\ \pi) > 0.$$

  3) There exist a state t, a constant $L > 0$, and $\overline{\alpha} \in (0, 1)$ such that:

     $$|J_\alpha(i) - J_\alpha(t)| \le L,$$
     $$i = 1, \ldots, n,$$
     $$\alpha \in (\overline{\alpha}, 1),$$

     where J_α is the α-discounted optimal cost vector.

Then:

1) The optimal average cost per stage cost has the same value, λ, for all intimal states, and it satisfies:

$$\lambda = \lim_{\alpha \to 1}(1-\alpha)J_\alpha(i),$$
$$i = 1, \ldots, n.$$

2) For any state t, the vector of differential cost, h is given by:

$$h(i) = \lim_{\alpha \to 1}(J_\alpha(i) - J_\alpha(t)),$$
$$i = 1, \ldots, n,$$

together with λ, satisfies Bellman's equation:

$$\lambda + h(i) = \lim_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u)h(j) \right],$$
$$i = 1, \ldots, n,$$

and λ is the optimal average cost per stage for all states i, i. e.,

$$\lambda = J^*(x) = \min_{\pi} J_\pi(i),$$
$$i = 1, \ldots, n.$$

The above result is also discussed in [2] where the minimization of an expected cost without discounting is considered. All classical methods for computing optimal policies and costs in dynamic programming have their counterparts for addressing average cost per stage problems. Certain alterations are nevertheless necessary. Let us consider first the *value iteration* method exhaustively analyzed in [11,12]. This is a method based on the premise that the limit of steps of the basic dynamic programming algorithm:

$$\lim_{k \to \infty} \frac{1}{k} T^k J = J^*.$$

Two issues arise with average cost per stage problems. First, some elements of the sequence $T^k J$ may diverge to $+\infty$ or $-\infty$ making the numerical calculation troublesome. Furthermore, since we found that the quantity described as the differential cost is important it would be appropriate to develop methods that allow the parallel computation of h as well. [14] developed the fundamentals based on which a *relative value iteration* of the form:

$$h^{k+1}(i) = (Th^k)(i) - (Th^k)(t),$$
$$i = 1, \ldots, n,$$

for some fixed state t, converges to vector h such that $(Th)(t)$ is equal to the optimal average cost per stage for all initial states, and h is the associated differential cost vector. [3] discusses various technical details required for proving convergence. Tight bounds that could improve the computational behavior of the value iteration method were proposed by [8] which modified the approach set forth in [14] to prove that upper and lower bounds on the maximal gain could be readily obtained. These are given according to:

$$\underline{c}_k \leq \underline{c}_{k+1} \leq \lambda \leq \overline{c}_{k+1} \leq \overline{c}_k,$$

where λ is the optimal average cost per stage for all initial states and

$$\underline{c}_k = \min_i [(Th^k)(i) - h^k(i)],$$
$$\overline{c}_k = \max_i [(Th^k)(i) - h^k(i)].$$

Recently, [4], by exploiting the connection between the average cost and the stochastic shortest path problem developed a new value iteration method by making use of *weighter sup-norm contraction* arising in the stochastic shortest path problem. One of the key advantages of this approach is that it admits a *Gauss–Seidel* implementation, thus it is amenable to a distributed implementation. *Policy iteration* methods can also be developed. The policy iteration algorithms generate sequences of stationary policies, each with improved cost over the preceding one. These methods are comprised of two basic steps, a policy evaluation and a policy improvement step. During the first step, for a given stationary policy, μ^k, we obtain the corresponding average and differential costs via the solution of the following system of equations which solution provides the kth iterate of λ, and h:

$$\lambda^k + h^k(i) = g(i, \mu^k(i)) + \sum_{j=1}^{n} p_{ij}(\mu^k(i))h^k(j),$$
$$j = 1, \ldots, n.$$

The policy improvement step, consists of finding a policy μ^{k+1}, where for all state i, is such that:

$$g(i,\mu^{k+1}(i)) + \sum_{j=1}^{n} p_{ij}(\mu^{k+1}(i))h^k(j)$$
$$= \min_{u\in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u)h^k(j) \right].$$

In [7], the scope of policy iteration is expanded so as to address problems in which the optimal average cost per stage is not the same for every initial state. It can also be shown, [3], that the optimal vector (λ^*, h^*) is equivalent to the optimal solution of the following linear program:

$$\begin{cases} \max & \lambda \\ \text{s.t.} & \lambda + h(i) \le g(i,u) + \sum_{j=1}^{n} p_{ij}(u)h(j) \\ & u \in U(i) \\ & i = 1,\dots,n. \end{cases}$$

In [9] the *dual problem* of the above-mentioned formulation is considered, whose optimal value is the optimal value of the *primal problem*. The form of the dual is:

$$\begin{cases} \min & \sum_{i=1}^{n} \sum_{u\in U(i)} q(i,u)g(i,u) \\ \text{s.t.} & \sum_{u\in U(i)} q(j,u) = \sum_{i=1}^{n} \sum_{u\in U(i)} q(i,u)p_{ij}(u) \\ & \quad j = 1,\dots,n \\ & \sum_{i=1}^{n} \sum_{u\in U(i)} q(i,u) = 1 \\ & q(i,u) \ge 0, \quad i = 1,\dots,n \\ & u \in U(i). \end{cases}$$

Simulation-based methods are presented in [3,5] that use the basic concepts of *Monte-Carlo simulation* as well as ideas of *reinforcement learning*, [13].

See also

- Dynamic Programming in Clustering
- Dynamic Programming: Continuous-time Optimal Control
- Dynamic Programming: Discounted Problems
- Dynamic Programming: Infinite Horizon Problems, Overview
- Dynamic Programming: Inventory Control
- Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- Dynamic Programming: Optimal Control Applications
- Dynamic Programming: Stochastic Shortest Path Problems
- Dynamic Programming: Undiscounted Problems
- Hamilton–Jacobi–Bellman Equation
- Multiple Objective Dynamic Programming
- Neuro-dynamic Programming

References

1. Arapostathis A, Borkar VK, Fernández-Gaucherand E, Ghosh ML, Markus SI (1993) Discrete-time controlled markov processes with average cost criterion: A survey. SIAM J Control Optim 31:282–344
2. Bather J (1973) Optimal decision procedures for finite Markov chains. Part I: Example. Adv Appl Probab 5:328–339
3. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
4. Bertsekas DP (1998) A new value iteration method for the average cost dynamic programming problem. SIAM J Optim 36:742–759
5. Bertsekas DP, Tsitsiklis JN (1997) Neuro-dynamic programming. Athena Sci., Belmont, MA
6. Blackwell D (1962) Discounted dynamic programming. Ann Math Statist 33:719–726
7. Blackwell D (1962) Discrete dynamic programming. Ann Math Statist 33:719–726
8. Odoni AR (1969) On finding the maximal gain for markov decision processes. Oper Res 17:857–860
9. Ross SM (1970) Applied probability models with optimization applications. Dover, Mineola, NY
10. Ross SM (1985) Probability models. Acad. Press, New York
11. Schweitzer PJ, Federgruen A (1977) The assymptotic behavior of undiscounted value iteration in Markov decision problems. Math Oper Res 2:360–381
12. Schweitzer PJ, Federgruen A (1978) The functional equation of undiscounted Markov renewal programming. Math Oper Res 3:308–321
13. Singh SP (1994) Reinforcement learning algorithms for average–payoff Markovian decision processes. Proc. 12th Nat. Conf. Artificial Intelligence
14. White DJ (1963) Dynamic programming, Markov chains and the method of successive approximation. J Math Anal Appl 6:373–376

Dynamic Programming in Clustering

L. J. HUBERT[1], P. ARABIE[2], J. MEULMAN[3]
[1] University Illinois, Champaign, USA
[2] Rutgers University, Newark, USA
[3] Leiden University, Leiden, The Netherlands

MSC2000: 90C39, 62H30

Article Outline

Keywords
The GDPP
Partitioning
  Admissibility Restrictions on Partitions
Hierarchical Clustering
  Constrained Hierarchical Clustering
Optimal Sequencing of an Object Set
  Optimal Sequencing Based on the Construction of Optimal Paths
Optimal Ordered Partitions
Heuristic Applications of the GDPP
See also
References

Keywords

Combinatorial optimization; Dynamic programming; Clustering; Classification; Ultrametric; Unidimensional scaling; Seriation

Many of the data analysis tasks that arise in the field of *classification* and *clustering* can be given some type of combinatorial characterization that involves the identification of object groupings, *partitions*, or *sequences*. These combinatorial structures are generally defined by certain properties of optimality for some loss (or merit) criterion based on data given in the form of an $n \times n$ *symmetric proximity* matrix **P** between distinct pairs of objects from a set $S = \{O_1, \ldots, O_n\}$. The focus of this entry will be solely in this context and the use of a general optimization strategy referred to as the *General Dynamic Programming Paradigm* (GDPP), which allows the construction of *recursive procedures* to solve a range of *combinatorial optimization* tasks encountered in the field of classification and clustering. The GDPP will be presented in a general form below with later sections indicating how it can be operationalized for a number of specific problem types. For a more extensive presentation of the topics introduced in this entry and for generalizations to proximity matrices that may not be symmetric or that are defined between objects from (two) distinct sets, the monograph [12] should be consulted. This latter source also provides numerical illustrations for the topics introduced here plus instructions on how to obtain a collection of programs (available on the World Wide Web) to carry out the various optimization tasks presented in this entry and [12]. For a recent and comprehensive review of cluster analysis and the use of mathematical programming techniques in general, see [7].

The GDPP

To present the GDPP, a collection of K sets of entities is first defined, $\Omega_1, \ldots, \Omega_K$, where it is possible by some operation to transform entities in Ω_{k-1} to certain entities in Ω_k for $2 \leq k \leq K$. Each such transformation can be assigned a *merit* (or *cost*) value based *only* on the entity in Ω_{k-1} and the transformed entity in Ω_k. An entity in Ω_k is denoted by A_k, and $\mathcal{F}(A_k)$ is the optimal value that can be assigned to A_k based on the sum of the merit (or cost) increments necessary to transform an entity in Ω_1, step-by-step, to $A_k \in \Omega_k$. If $A_{k-1} \in \Omega_{k-1}$ can be transformed into $A_k \in \Omega_k$, the merit (or cost) of that single transition will be denoted by $M(A_{k-1}, A_k)$ (or $C(A_{k-1}, A_k)$), and where the latter *does not depend* on how A_{k-1} may have been arrived at starting from an entity in Ω_1. Given these conditions, and assuming the values $\mathcal{F}(A_1)$ for $A_1 \in \Omega_1$ are available to initialize the recursive system, $\mathcal{F}(A_k)$ may be constructed for $k = 2, \ldots, K$ (when merit is to be maximized) as

$$\mathcal{F}(A_k) = \max[\mathcal{F}(A_{k-1}) + M(A_{k-1}, A_k)],$$

where $A_k \in \Omega_k$, $A_{k-1} \in \Omega_{k-1}$, and the maximum is taken over all A_{k-1} that can be transformed into A_k. Or, if cost is to be minimized,

$$\mathcal{F}(A_k) = \min[\mathcal{F}(A_{k-1}) + C(A_{k-1}, A_k)].$$

In addition, both max/min and min/max forms could be considered as:

$$\mathcal{F}(A_k) = \max[\min(\mathcal{F}(A_{k-1}), M(A_{k-1}, A_k))],$$
$$\mathcal{F}(A_k) = \min[\max(\mathcal{F}(A_{k-1}), C(A_{k-1}, A_k))].$$

The leading maximization or minimization is over all $A_{k-1} \in \Omega_{k-1}$ that can be transformed into A_k. In all instances, an optimal solution is identified by some value for $\mathcal{F}(A_K)$ for a specific $A_K \in \Omega_K$, and the actual optimal solution obtained by working backwards through the recursion to see how $\mathcal{F}(A_K)$ was constructed.

Partitioning

The most direct characterization of the partitioning task can be stated as follows: given $S = \{O_1, \ldots, O_n\}$ and $\mathbf{P} = \{p_{ij}\}$, find a collection of M mutually exclusive and exhaustive subsets (or clusters) of S, say, $S_1, \ldots, S_M$, such that for some measure of *heterogeneity* $H(\cdot)$ that attaches a value to each possible subset of S, either the sum $\sum_{m=1}^{M} H(S_m)$, or alternatively, $\max[H(S_1), \ldots, H(S_M)]$, is minimized. This stipulation assumes that heterogeneity has a cost interpretation and that smaller values of the heterogeneity indices represent the'better' subsets (or clusters). If $H(S_m)$ for some $S_m \subseteq S$ depends only on those proximities from $\mathbf{P}$ that are within S_m and/or between S_m and $S - S_m$, an application of the GDPP is possible. Define K to be M, and let each of the sets $\Omega_1, \ldots, \Omega_M$ contain *all* of the $2^n - 1$ nonempty subsets of the n object subscripts; $\mathcal{F}(A_k)$ is the optimal value for a partitioning into k classes of the object subscripts present in A_k. A transformation of an entity $A_{k-1} \in \Omega_{k-1}$ to $A_k \in \Omega_k$ is possible if $A_{k-1} \subset A_k$, with cost $C(A_{k-1}, A_k) \equiv H(A_k - A_{k-1})$. Thus, beginning with the heterogeneity indices $H(A_1)$ for *every* subset $A_1 \subseteq S$, the recursion can be carried out, with the optimal solution represented by $\mathcal{F}(A_M)$ when $A_M = S$.

The first discussion of this general type of recursive solution for the partitioning task was in [14] but limited to one specific measure of subset heterogeneity defined by the sum of proximities within a subset divided by twice the number of objects in the subset. If the original proximities in $\mathbf{P}$ happened to be squared *Euclidean distances* between numerically given vectors (or profiles) for the n objects over some set of variables, then this subset heterogeneity measure is equivalent to the sum of squared Euclidean distances between each profile and the mean profile for the subset (this quantity is usually called the *sum of squared error* or the *k-means criterion*, e. g., see [16], p. 52) A major advantage of the GDPP formulation is that a variety of heterogeneity measures can be considered under a common rubric, with the sole requirement that the measure chosen be dependent only on the proximities within a subset and/or between the subset and its complement. For example, in [12] some twelve different alternatives are illustrated using a program implementation that can effectively deal with object set sizes in their lower 20's with the type of computational equipment and storage capacity now commonly available. As noted in a later section, it is also possible to extend the GDPP heuristically to allow for much larger object set sizes, although an absolute guarantee of globally optimality for the identified structures is sacrificed.

Admissibility Restrictions on Partitions

A specific restriction discussed at some length in the literature (see [8, Chapt. 5], [16, pp. 61–64], [6]) that would permit the construction of optimal partitions (subject to the restriction) for very large object sets is when there is an a priori assumed object *ordering* along a continuum that can be taken without loss of generality as $O_1 \prec \cdots \prec O_n$, and the only *admissible clusters* are those for which the objects in the cluster form a consecutive sequence or segment. Thus, an optimal partition will consist of M clusters, each of which defines a consecutive segment along the given object ordering. To tailor the GDPP to a consecutive-ordering admissibility criterion, each of the sets $\Omega_1, \ldots, \Omega_M$ is now defined by the n subsets of S that contain the objects $\{O_1, \ldots, O_i\}$ for $1 \leq i \leq n$; $\mathcal{F}(A_k)$ is the optimal value for a partitioning of A_k into k classes; a transformation of an entity $A_{k-1} \in \Omega_{k-1}$ to $A_k \in \Omega_k$ is possible if $A_{k-1} \subset A_k$; and the cost of the transition is $H(A_k - A_{k-1})$, where $A_k - A_{k-1}$ must contain a consecutive sequence of objects. Again, $\mathcal{F}(A_M)$ for $A_M = S$ identifies an optimal solution.

The selection of some prespecified ordering that constrains admissible clusters in a partition obviously does not lead necessarily to the same unconstrained optimal partitions, even though the identical subset heterogeneity measure and optimization criterion are being used. There are, however, several special instances where the original proximity matrix $\mathbf{P}$ is appropriately defined and/or patterned so that the imposition of a particular order constraint does invariably lead to partitions that would also be optimal even when no such order constraint was imposed. One such result dates back to W.D. Fisher [6] who showed that when proxim-

ities are squared differences between the values on some (unidimensional) variable, and the order constraint is derived from the ordering of the objects on this variable, then the selection of the sum of squared error as the subset heterogeneity measure, and minimizing this sum as an optimization criterion, leads to partitions that are not only optimal under the order constraint but also optimal when unconstrained, i. e., an unconstrained optimal partition will include only those subsets defined by objects consecutive in the given order. (The subset heterogeneity measure in this unidimensional case reduces to the sum of squared deviations of the univariate values for the objects from their mean value within the subset.) A more general result appears in [3] where the special case is discussed when a proximity matrix **P** can be row- and column-reordered to display an anti-Robinson form (a matrix pattern first introduced in [15]). As stated more formally below, a matrix has an *anti-Robinson* form if the entries within each row and column of **P** never decrease when moving away from a main diagonal entry in any direction. For certain subset heterogeneity measures and optimization criteria, imposing the order constraint that displays the anti-Robinson pattern in the row- and column-reordered proximity matrix leads to partitions that are also optimal when unconstrained.

The choice of an ordering that can be imposed to constrain the search domain for optimal partitions could be directly tied to a task, discussed later, of finding an (optimal) sequencing of the objects along a continuum. Somewhat more generally, one possible data analysis strategy for seeking partitions as close to optimal as possible, would be to construct an object ordering through an initial optimization process, and possibly one based on another analysis method that could then constrain the domain of search for an optimal partition. Obviously, if one were successful in generating an appropriate object ordering, partitions that would be optimal when constrained would also be optimal without the constraint. The obvious key here is to have some mechanism for identifying an appropriate order to give this possible equivalence (between an optimal constrained partition and one that is optimal without constraint) a chance to succeed. As one example of how such a process might be developed for constructing partitions based on an empirically generated ordering for the objects, a three-stage process is proposed in [1] and [2]. First, the objects to be partitioned are embedded in a *Euclidean representation* with a specific *multidimensional scaling* strategy. Second, by heuristic methods, a path among the n objects in the Euclidean representation is identified (hopefully, with close to minimal length) and used to define a prior ordering for the objects and to constrain the subsets present in a partition. Finally, a recursive strategy of the same general form just described is carried out to obtain a partitioning of S.

Hierarchical Clustering

The problem of hierarchical clustering will be characterized by the search for an optimal collection of partitions of S, which are denoted generically as $\mathcal{P}_1, \ldots, \mathcal{P}_n$. Here, $\mathcal{P}_1$ is the (trivial) partition where all n objects from S are placed into n separate classes, $\mathcal{P}_n$ is the (also trivial) partition where a single subset contains all n objects, and $\mathcal{P}_k$ is obtained from $\mathcal{P}_{k-1}$ by uniting some pair of classes present in $\mathcal{P}_{k-1}$. As an optimization criterion the sum of transition costs is minimized, irrespective of how the costs might be defined, between successive partitions in a hierarchy. Specifically, suppose $T(\mathcal{P}_{k-1}, \mathcal{P}_k)$ denotes some measure of transition cost between two partitions $\mathcal{P}_{k-1}$ and $\mathcal{P}_k$, where $\mathcal{P}_k$ is constructed from $\mathcal{P}_{k-1}$ by uniting two classes in the latter partition. An optimal *partition hierarchy* $\mathcal{P}_1, \ldots, \mathcal{P}_n$ will be one for which the sum of the transition costs, $\sum_{k>2} T(\mathcal{P}_{k-1}, \mathcal{P}_k)$, is minimized. To apply the GDPP, first define n sets $\Omega_1, \ldots, \Omega_n$, where Ω_k contains all partitions of the n objects in S into $n- k+ 1$ classes. The value $\mathcal{F}(A_k)$ for $A_k \in \Omega_k$ is the optimal sum of transition costs up to the partition A_k; a transformation of an entity $A_{k-1} \in \Omega_{k-1}$ to $A_k \in \Omega_k$ is possible if A_k is obtainable from A_{k-1} by uniting two classes in A_{k-1}, and has cost $C(\mathrm{A}_{k-1}, A_k) \equiv T(A_{k-1}, A_k)$. Beginning with an assumed value for $\mathcal{F}(A_1)$ of 0 for the single entity $A_1 \in \Omega_1$ (which is the partition of S into n subsets each containing a single object), and constructing $\mathcal{F}(A_k)$ recursively for $2, \ldots, n$, an optimal solution is identified by $\mathcal{F}(A_n)$ for the single entity $A_n \in \Omega_n$ defined by the partition containing all n objects in a single class.

A concept routinely encountered in discussions of hierarchical clustering is that of an *ultrametric*, which can be characterized by any nonnegative $n \times n$ symmetric dissimilarity matrix for distinct pairs of the objects in S, denoted generically as $\mathbf{U} = \{u_{ij}\}$, where $u_{ij} = 0$ if and

only if $i = j$ and the entries in $\mathbf{U}$ satisfy the ultrametric inequality: $u_{ij} \leq \max\{u_{ik}, u_{jk}\}$ for $1 \leq i, j, k \leq n$. Any ultrametric identifies a specific partition hierarchy, $\mathcal{P}_1, \ldots, \mathcal{P}_n$, where those object pairs defined between subsets united in $\mathcal{P}_{t-1}$ to form $\mathcal{P}_t$ all have a common ultrametric value; moreover, this latter value is not smaller than those for object pairs defined within these same subsets. One approach to the development of hierarchical clustering methods is by directly fitting an ultrametric to $\mathbf{P}$ minimizing a loss criterion defined by an L_p-norm between $\{p_{ij}\}$ and a (to be identified) ultrametric matrix $\{u_{ij}\}$. To be specific, for a given partition hierarchy, $\mathcal{P}_1, \ldots, \mathcal{P}_n$, let $C_{t-1}^{(u)}$ and $C_{t-1}^{(v)}$ denote the two classes united in $\mathcal{P}_{t-1}$ to form $\mathcal{P}_t$, and specify b_{t-1} to be some appropriate aggregate (or 'average') value of the proximities for object pairs between $C_{t-1}^{(u)}$ and $C_{t-1}^{(v)}$. The loss functions used to index the adequacy of a given partition hierarchy in producing an ultrametric fitted to $\mathbf{P}$ are for the *L_1-norm*:

$$\sum_{t=2}^{n} \sum_{\substack{O_{i'} \in C_{t-1}^{(u)}, \\ O_{j'} \in C_{t-1}^{(v)}}} \left| p_{i'j'} - b_{t-1} \right|,$$

where b_{t-1} is the median proximity between $C_{t-1}^{(u)}$ and $C_{t-1}^{(v)}$; for the *L_2-norm*:

$$\sum_{t=2}^{n} \sum_{\substack{O_{i'} \in C_{t-1}^{(u)}, \\ O_{j'} \in C_{t-1}^{(v)}}} (p_{i'j'} - b_{t-1})^2,$$

where b_{t-1} is the mean proximity between $C_{t-1}^{(u)}$ and $C_{t-1}^{(v)}$; and for the *L_∞-norm*:

$$\sum_{t=2}^{n} \max_{\substack{O_{i'} \in C_{t-1}^{(u)}, \\ O_{j'} \in C_{t-1}^{(v)}}} \left| p_{i'j'} - b_{t-1} \right|,$$

where b_{t-1} is the average of the minimum and maximum proximities between $C_{t-1}^{(u)}$ and $C_{t-1}^{(v)}$. For all three L_p-norms, an optimal ultrametric will be one for which the order constraint on the between-subset aggregate values holds: $b_1 \leq \cdots \leq b_{n-1}$, and the norm is minimized. For such an optimal solution, $b_1, \ldots, b_{n-1}$ define the distinct entries in an (optimal) fitted ultrametric. To implement a *dynamic programming* approach for locating an optimal ultrametric, $C(A_{k-1}, A_k)$ is the incremental cost of transforming A_{k-1} to A_k characterized by the appropriate L_p-norm when that pair of subsets in A_{k-1} is united to form A_k. As developed in detail in [11], an explicit admissibility criterion must also be imposed for defining a permissible transition from A_{k-1} to A_k that could ensure a nondecreasing sequence of between-subset aggregate values.

Constrained Hierarchical Clustering

Analogously to the admissibility conditions for partitions, one constraint that might be imposed on each partition in Ω_k is for the constituent subsets to contain objects consecutive in some given *ordering* (which could be taken as $O_1 \prec \cdots \prec O_n$ without loss of any generality). Thus, Ω_k will be redefined to contain those partitions that include $n - k + 1$ classes, and where each class is a segment in the given object ordering.

Optimal Sequencing of an Object Set

A combinatorial optimization task closely related to both partitioning and hierarchical clustering is the search for an optimal *sequencing* of the object set S based on the proximity matrix $\mathbf{P}$. A best reordering is sought for the rows and columns of $\mathbf{P}$ that will optimize, over all possible row/column reorderings, some specified measure of patterning for the entries of the reordered matrix. Irrespective of the particular measure chosen, the GDPP is specialized as follows: A collection of sets $\Omega_1, \ldots, \Omega_n$ is defined, where Ω_k includes all the subsets that have k members from the integer set $\{1, \ldots, n\}$. The value $\mathcal{F}(A_k)$ is the optimal contribution to the total measure of matrix patterning for the objects in A_k when they occupy the first k positions in the (re)ordering. A transformation is now possible between $A_{k-1} \in \Omega_{k-1}$ and $A_k \in \Omega_k$ if $A_{k-1} \subset A_k$ (i. e., A_{k-1} and A_k differ by one integer). The contribution to the total measure of patterning generated by placing the single integer in $A_{k-1} - A_k$ at the kth order position is $M(A_{k-1}, A_k)$. As always, the validity of the recursive process will require the incremental merit index, $M(A_{k-1}, A_k)$, to depend only on the unordered sets A_{k-1} and A_k, and the complement $S - A_k$, and specifically *not* on how A_{k-1} may have been obtained beginning with Ω_1. Assuming $\mathcal{F}(A_1)$ for all $A_1 \in \Omega_1$ are available, the recursive process can be carried out from Ω_1 to Ω_n, with $\mathcal{F}(A_n)$ for the single set $A_n = \{1, \ldots, n\} \in \Omega_n$ defining the optimal value for the specified measure of matrix patterning.

Two general classes of measures of matrix patterning are mentioned here. The first is a row and column *gradient index* motivated by the ideal of reordering a symmetric proximity matrix to have an anti-Robinson form (and which is the same structure noted briefly in the clustering context when an optimal order-constrained partition might also be optimal when unconstrained). Specifically, suppose $\rho(\cdot)$ is some permutation of the first n integers that reorders both the rows and columns of $\mathbf{P}$ (i. e., $\mathbf{P}_\rho \equiv \{p_{\rho(i)\rho(j)}\}$). As noted earlier, the reordered matrix $\mathbf{P}_\rho$ is said to have an anti-Robinson form if the entries within the rows and within the columns of $\mathbf{P}_\rho$ moving away from the main diagonal in any direction never decrease; or formally, two gradient conditions must be satisfied:

- within rows: $p_{\rho(i)\rho(k)} \leq p_{\rho(i)\rho(j)}$ for $1 \leq i < k < j \leq n$;
- within columns: $p_{\rho(k)\rho(j)} \leq p_{\rho(i)\rho(j)}$ for $1 \leq i < k < j \leq n$.

It might be noted that whenever $\mathbf{P}$ is an ultrametric, or if $\mathbf{P}$ has an exact Euclidean representation in a single dimension (i. e., $\mathbf{P} = \{|x_j - x_i|\}$, for some collection of coordinate values, $x_1, \ldots, x_n$), then $\mathbf{P}$ can be row/column reordered to display a perfect anti-Robinson pattern. Thus, the notion of an anti-Robinson form can be interpreted as generalizing either a perfect discrete classificatory structure induced by a partition hierarchy (through an ultrametric) or as the pattern expected in $\mathbf{P}$ if there exists an exact *unidimensional Euclidean representation* for the objects in S. In any case, if a matrix can be row/column reordered to display an anti-Robinson form, then the objects are orderable along a continuum so that the degree of separation between objects in the ordering is reflected perfectly by the dissimilarity information in $\mathbf{P}$, i. e., for the object ordering, $O_{\rho(i)} \prec O_{\rho(k)} \prec O_{\rho(j)}$ (for $i < k < j$), $p_{\rho(i)\rho(k)} \leq p_{\rho(i)\rho(j)}$ and $p_{\rho(k)\rho(j)} \leq p_{\{\rho(i)\rho(j)}$.

A natural (merit) measure of how well a reordered proximity matrix $\mathbf{P}_\rho$ satisfies these two gradient conditions would rely on an aggregate index of the violations/nonviolations over all distinct object triples, as given by the expression:

$$\sum_{i<k<j} f(p_{\rho(i)\rho(k)}, p_{\rho(i)\rho(j)}) + \sum_{i<k<j} f(p_{\rho(k)\rho(j)}, p_{\rho(i)\rho(j)}), \quad (1)$$

where $f(\cdot, \cdot)$ is some function indicating how a violation/nonviolation of a particular gradient condition for an object triple within a row or within a column (and defined above the main diagonal of $\mathbf{P}_\rho$) is to be counted in the total measure of merit. The one option concentrated on here will be $f(z, y) = \text{sign}(z - y) = +1$ if $z > y$; 0 if $z = y$; and -1 if $z < y$; thus, the (raw) number of satisfactions minus the number of dissatisfactions of the gradient conditions *within rows* above the main diagonal of $\mathbf{P}_\rho$ is given by the first term in (1), and the (raw) number of satisfactions minus dissatisfactions of the gradient conditions *within columns* above the main diagonal of $\mathbf{P}_\rho$ is given by the second term. To carry out the GDPP based on the measure in (1), an explicit form must be given for the incremental contribution, $M(A_{k-1}, A_k)$, to the total merit measure of patterning generated by placing the single integer $A_k - A_{k-1}$ at the kth order position. For any ordering $\rho(\cdot)$ of the rows and columns of $\mathbf{P}$, the merit increment for placing an integer, say, k' ($\equiv \rho(k)$) (i. e., $\{k'\} = A_k - A_{k-1}$) at the kth order position can be defined as $\sum_{k=1}^{n} I_{\text{row}}(\rho(k)) + \sum_{k=1}^{n} I_{\text{col}}\rho(k))$, where

$$I_{\text{row}}(\rho(k)) = \sum_{i' \in A_{k-1}} \sum_{j' \in S - A_k} f(p_{i'k'}, p_{i'j'}),$$

$$I_{\text{col}}(\rho(k)) = \sum_{i' \in A_{k-1}} \sum_{j' \in S - A_k} f(p_{k'j'}, p_{i'j'}),$$

and $A_{k-1} = \{\rho(1), \ldots, \rho(k-1)\}$, $S - A_k = \{\rho(k+1), \ldots, \rho(n)\}$. Thus, letting $\mathcal{F}(A_1) = 0$ for all $A_1 \in \Omega_1$, and using the specification for $f(\cdot, \cdot)$ suggested above, the recursion can be carried out to identify an optimal row/column reordering of the given proximity matrix $\mathbf{P}$ to maximize this gradient measure over all row/column reorderings of $\mathbf{P}$.

A second class of measures of matrix patterning can be derived indirectly from the auxiliary problem of attempting to fit a given proximity matrix $\mathbf{P}$ by some type of unidimensional scaling representation (i. e., a *seriation*). Suppose the search is for a set of n ordered coordinate values, $x_1 \leq \cdots \leq x_n$ (such that $\sum_k x_k = 0$), and a permutation $\rho(\cdot)$ to minimize the *least squares criterion*

$$\sum_{i<j} (p_{\rho(i)\rho(j)} - |x_j - x_i|)^2.$$

After some algebraic reduction (see [5]), this latter least squares criterion can be rewritten as

$$\sum_{i<j} p_{ij}^2 + n\sum_k [x_k - \left(\frac{1}{n}\right) G(\rho(k))]^2 - \frac{1}{n}\sum_k [G(\rho(k))]^2,$$

where

$$G(\rho(k)) = \sum_{i=1}^{k-1} p_{\rho(k)\rho(i)} - \sum_{i=k+1}^{n} p_{\rho(k)\rho(i)}.$$

If the measure

$$\sum_{k=1}^{n} [G(\rho(k))]^2 \tag{2}$$

is maximized over all row/column reorderings of **P**, and denoting the optimal permutation by $\rho^*(\cdot)$, then $G(\rho^*(1)) \leq \cdots \leq G(\rho^*(n))$, and the optimal coordinates can be retrieved as $x_k = (1/n)G(\rho^*(k))$, for $1 \leq k \leq n$. To execute the GDPP recursion using (2), the merit increment for placing the integer, say k' ($\equiv \rho(k)$) (i. e., $\{k'\} = A_k - A_{k-1}$) in the kth order position can be written as $[G(\rho(k))]^2$, where

$$G(\rho(k)) = \sum_{i' \in A_{k-1}} p_{k'i'} - \sum_{j' \in S - A_k} p_{k'j'},$$

with $A_{k-1} = \{\rho(1), \ldots, \rho(k-1)\}$, $S - A_k = \{\rho(k+1), \ldots, \rho(n)\}$, and $\mathcal{F}(A_1)$ for $A_1 = \{k'\} \in \Omega_1$ defined by

$$\left(\sum_{j' \in S - \{k'\}} p_{k'j'}\right)^2.$$

Optimal Sequencing Based on the Construction of Optimal Paths

To tailor the GDPP (and for the moment emphasizing the minimization of the sum of adjacent object proximities in constructing a path among the objects in S), a collection of sets $\Omega_1, \ldots, \Omega_n$ is defined so that each entity in Ω_k, $1 \leq k \leq n$, is now an ordered pair (A_k, j_k). Here, A_k is a k element subset of the n subscripts on the objects in S, and j_k is one subscript in A_k (to be interpreted as the subscript for the last-placed object in a sequencing of the objects contained within A_k). The function value $\mathcal{F}((A_k, j_k))$ is the optimal contribution to the total measure of matrix patterning for the objects in A_k when they are placed in the first k positions in the (re)ordering, and the object with subscript j_k occupies the kth. A transformation is possible between $(A_{k-1}, j_{k-1}) \in \Omega_{k-1}$ and $(A_k, j_k) \in \Omega_k$ if $A_{k-1} \subset A_k$ and $A_k - A_{k-1} = \{j_k\}$ (i. e., A_{k-1} and A_k differ by the one integer j_k). The cost increment $C((A_{k-1}, j_{k-1}), (A_k, j_k))$ is simply $p_{(j_{k-1})j_k}$ for the contribution to the total measure of patterning generated by placing the object with the single integer subscript in $A_k - A_{k-1}$ at the kth order position (i. e., the proximity between the adjacently-placed objects with subscripts j_{k-1} and j_k). The type of GDPP recursion used for the construction of optimal *linear paths* can be modified easily for the construction of optimal *circular paths*: choose object O_1 as an (arbitrary) origin and force the construction of the optimal linear paths to include O_1 as the initial object by defining $\mathcal{F}((A_1, j_1)) = 0$ for $j_1 = 1$ and $A_1 = \{1\}$, and otherwise by a very large positive or negative value (depending on whether the task is a minimization or a maximization, respectively). The function values $\mathcal{F}((A_n, j_n))$ for all j_n, $1 \leq j_n \leq n$ for $(A_n, j_n) \in \Omega_n$ and $A_n = \{1, \ldots, n\}$ can then be used to obtain the optimal circular paths depending on the chosen optimization criteria as follows:

- *minimum path length*:

  $$\min[\mathcal{F}((A_n, j_n)) + p_{j_n 1}];$$

- *maximum path length*:

  $$\max[\mathcal{F}((A_n, j_n)) + p_{j_n 1}];$$

- *minimax path length*:

  $$\min[\max(\mathcal{F}((A_n, j_n)), p_{j_n 1})];$$

- *maximin path length*:

  $$\max[\min(\mathcal{F}((A_n, j_n)), p_{j_n 1})].$$

For the first discussions in the literature on constructing optimal paths through DP, see [4,9]; for applications to a variety of data analysis tasks, see [13].

Optimal Ordered Partitions

The task of constructing an *ordered partition* of an object set $S = \{O_1, \ldots, O_n\}$ into M ordered classes, $S_1 \prec \cdots \prec S_M$, using some (merit) measure of matrix patterning

and a proximity matrix **P**, can be approached through the GDPP recursive process applied to the partitioning task but with appropriate variation in defining the merit increments. Explicitly, the sets $\Omega_1, \ldots, \Omega_M$ will each contain all $2^n - 1$ nonempty subsets of the n object subscripts; $\mathcal{F}(A_k)$ for $A_k \in \Omega_k$ is the optimal value for placing k classes in the first k positions, and the subset A_k is the union of these k classes. A transformation from $A_{k-1} \in \Omega_{k-1}$ to $A_k \in \Omega_k$ is possible if $A_{k-1} \subset A_k$; the merit increment $M(A_{k-1}, A_k)$ is based on placing the class $A_{k-1} - A_k$ at the kth position (which will depend on A_{k-1}, A_k, and $S - A_k$). Beginning with $\mathcal{F}(A_1)$ for all $A_1 \in \Omega_1$ (i. e., the merit of placing the class A_1 at the first position), the recursion proceeds from Ω_1 to Ω_M, with $\mathcal{F}(A_M)$ for $A_M = S \in \Omega_M$ defining the optimal merit value for an ordered partition into M classes.

To generalize the gradient measure given in (1), the merit increment for placing the class $A_k - A_{k-1}$ at the kth order position is $I_{\text{row}}(A_k - A_{k-1}) + I_{col(A_k - A_{k-1})}$, where

$$I_{\text{row}}(A_k - A_{k-1}) = \sum_{i' \in A_{k-1}} \sum_{k' \in A_k - A_{k-1}} \sum_{j' \in S - A_k} f(p_{i'k'}, p_{i'j'}),$$

and

$$I_{\text{col}}(A_k - A_{k-1}) = \sum_{i' \in A_{k-1}} \sum_{k' \in A_k - A_{k-1}} \sum_{j' \in S - A_k} f(p_{k'j'}, p_{i'j'}).$$

To initialize the recursion, let $\mathcal{F}(A_1) = 0$ for all $A_1 \in \Omega_1$.

A merit measure based on a coordinate representation for each of the M ordered classes, $S_1 \prec \cdots \prec S_M$, that generalizes (2) can also be developed directly. Here, M coordinates, $x_1 \leq \cdots \leq x_M$, are to be identified so that the residual sum-of-squares

$$\sum_{k \leq k'} \sum_{\substack{i_k \in S_k, \\ j_{k'} \in S_{k'}}} (p_{i_k j_{k'}} - |x_{k'} - x_k|)^2,$$

is minimized (the notation $pi_kj_{k'}$) indicates those proximities in **P** defined between objects with subscripts $i_k \in S_k$ and $j_{k'M} \in S_{k'}$). A direct extension of the argument that led to optimal coordinate representation for single objects would require the maximization of

$$\sum_{k=1}^{M} \left(\frac{1}{n_k}\right) (G(A_k - A_{k-1}))^2, \tag{3}$$

where

$$G(A_k - A_{k-1}) = \sum_{k' \in A_k - A_{k-1}} \left(\sum_{i' \in A_{k-1}} p_{k'i'} - \sum_{i' \in S - A_k} p_{k'i'} \right),$$

and n_k denotes the number of objects in $A_k - A_{k-1}$. The merit increment for placing the subset $A_k - A_{k-1}$ at the kth order position would be $(1/n_k)(G(A_k - A_{k-1}))^2$, with the recursion initialized by

$$\mathcal{F}(A_1) = \left(\frac{1}{n_1}\right) \left(\sum_{k' \in A_1} \sum_{i' \in S - A_1} p_{k'i'} \right)^2,$$

for all $A_1 \in \Omega_1$. If an optimal ordered partition that maximizes (3) is denoted by $S_1^* \prec \cdots \prec S_M^*$, the optimal coordinates for each of the M classes can be given as

$$x_k^* = \left(\frac{1}{n}\right) \left(\frac{G(S_k^*)}{n_k}\right),$$

where $x_1^* \leq \cdots \leq x_M^*$, and $\sum_k n_k x_k^* = 0$. A more complete discussion of constructing optimal ordered partitions appears in [10].

Heuristic Applications of the GDPP

When faced with the task of finding a single optimal partition for a (large) object set S, if one had knowledge that for an optimal M-class partition the classes could be allocated to two (or more) groups, then the aggregate collections of the objects within these latter groups could be separately and optimally partitioned and an optimal M-class partition for the complete object set identified directly. Or, if it were known that certain elemental subsets of the objects in S had to appear within the classes of an optimal M-class partition, one could begin with these elemental subsets as the objects to be analyzed, and an optimal M-class partition could again be retrieved. The obvious difficulty is to identify either the larger aggregate groups that might be dealt with separately, or an appropriate collection of elemental subsets, and in a size and number that might be handled by the recursive optimization strategy. For the latter task of identifying elemental subsets, one possible approach would be to begin with a partition of S into several classes (possibly obtained through another heuristic process), and where each class con-

tained a number of objects that could be optimally analyzed. Based on these separate subset analyses, a (tentative) collection of elemental subsets would be identified. These could then be used to obtain a subdivision of S, and again within each group of this subdivision, the objects could be optimally partitioned to generate a possibly better collection of elemental subsets. This process could be continued until no change occurred in the particular elemental subsets identified. As an alternative, one could start with some collection of tentative *elemental subsets* obtained through another (*heuristic*) optimization strategy and try, if possible, to improve upon these through the same type of procedure. This latter approach is illustrated in [12]. Similarly, the tasks of constructing a (hopefully optimal) partition hierarchy or object order for a (large) set could be approached through the identification of a collection of elemental subsets, which would then be operated on as the basic entities for the generation of a partition hierarchy or an object sequence.

See also

- ► Dynamic Programming: Average Cost Per Stage Problems
- ► Dynamic Programming: Continuous-time Optimal Control
- ► Dynamic Programming: Discounted Problems
- ► Dynamic Programming: Infinite Horizon Problems, Overview
- ► Dynamic Programming: Inventory Control
- ► Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ► Dynamic Programming: Optimal Control Applications
- ► Dynamic Programming: Stochastic Shortest Path Problems
- ► Dynamic Programming: Undiscounted Problems
- ► Hamilton–Jacobi–Bellman Equation
- ► Multiple Objective Dynamic Programming
- ► Neuro-dynamic Programming

References

1. Alpert CJ, Kahng AB (1995) Multiway partitioning via geometric embeddings, orderings, and dynamic programming. IEEE Trans Computer-Aided Design Integr Circuits and Syst 14:1342–1357
2. Alpert CJ, Kahng AB (1997) Splitting an ordering into a partition to minimize diameter. J Class 14:51–74
3. Batagelj V, Korenjak-Černe S, Klavžar S (1994) Dynamic programming and convex clustering. Algorithmica 11:93–103
4. Bellman R (1962) Dynamic programming treatment of the traveling salesman problem. J ACM 9:61–63
5. Defays D (1978) A short note on a method of seriation. British J Math Statist Psych 31:49–53
6. Fisher WD (1958) On grouping for maximum heterogeneity. J Amer Statist Assoc 53:789–798
7. Hansen P, Jaumard B (1997) Cluster analysis and mathematical programming. Math Program 79:191–215
8. Hartigan JA (1975) Clustering algorithms. Wiley, New York
9. Held M, Karp RM (1962) A dynamic programming approach to sequencing problems. J Soc Indus Appl Math 10:196–210
10. Hubert LJ, Arabie P, Meulman J (1997) The construction of globally optimal ordered partitions. In: Mirkin B, McMorris FR, Roberts FS, Rzhetsky A (eds) Mathematical hierarchies and biology. DIMACS, Amer. Math. Soc., Providence, RI, pp 299–312
11. Hubert LJ, Arabie P, Meulman J (1997) Hierarchical clustering and the construction of (optimal) ultrametrics using L_p-norms. In: Dodge Y (ed) L_1-statistical procedures and related topics. vol 31 of Lecture Notes. Inst. Math. Statist., Berkley, U.S., pp 457–472
12. Hubert LJ, Arabie P, Meulman J (2001) Combinatorial data analysis: Optimization by dynamic programming. SIAM, Philadelphia
13. Hubert LJ, Baker FB (1978) Applications of combinatorial programming to data analysis: The traveling salesman and related problems. Psychometrika 43:81–91
14. Jensen RE (1969) A dynamic programming algorithm for cluster analysis. J Oper Res Soc Amer 7:1034–1057
15. Robinson WS (1951) A method for chronologically ordering archaeological deposits. Amer Antiq 16:293–301
16. Späth H (1980) Cluster analysis algorithms. Horwood, Westergate

Dynamic Programming: Continuous-time Optimal Control

WILLIAM R. ESPOSITO
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 49L20, 34H05, 90C39

Article Outline

Keywords
Problem Formulation

Example
Hamilton–Jacobi–Bellman Equation
Pontryagin Minimum Principle
See also
References

Keywords

Dynamic programming; Continuous-time optimal control

Even though *dynamic programming* [1] was originally developed for systems with discrete types of decisions, it can be applied to continuous problems as well. In this article the application of dynamic programming to the solution of continuous time optimal control problems is discussed.

Problem Formulation

Consider the following continuous time dynamical system:

$$\begin{cases} \dot{z}(t) = f(z(t), u(t)), \\ z(0) = z_0, \qquad\qquad 0 \le t \le T, \end{cases} \tag{1}$$

where $z(t) \in \mathbf{R}^n$ is the state vector at time t with time derivative given by $\dot{z}(t)$, $u(t) \in U \subset \mathbf{R}^m$ is the control vector at time t, U is the set of control constraints, and T is the terminal time. The function $f(z(t), u(t))$ is continuously differentiable with respect to z and continuous with respect to u. The set of admissible control trajectories are given by the piecewise constant functions, $\{u(t): u(t) \in U, \forall t \in [0, T]\}$. It is assumed that for any admissible control trajectory, that a state trajectory $z^u(t)$ exists and is unique. For a full treatment of existence and uniqueness, see [4].

The objective is to determine a control trajectory and the corresponding state trajectory which minimizes a cost function of the form:

$$h(z^u(T)) + \int_0^T g(z^u(t), u(t))\, dt, \tag{2}$$

where the functions g, and h are continuously differentiable with respect to both z and u.

Example

As a simple example, consider the problem of moving a unit mass from an initial point to a given final point. The position of the mass along a line is given by the state $z_1(t)$ and its velocity by $z_2(t)$. The control $u(t)$ is the force applied to the mass, and is bounded $u(t) \in [-1, 1]$. This system is described by:

$$\begin{aligned} &\dot{z}_1(t) = z_2(t), \quad \dot{z}_2(t) = u(t), \\ &z(0) = [z_1(0), z_2(0)], \quad t \in [0, T], \\ &u(t) \in [-1, 1]. \end{aligned}$$

The objective is to move this mass as near to the final state point, $[\bar{z}_1, \bar{z}_2]$, as possible. This can be formulated as the minimization of the square error at the final time point.

$$\min_{u(t)} \sum_{i=1}^{2} (z_i(T) - \bar{z}_i(T))^2 .$$

Converting this cost function into the form given by (2)) results in:

$$\begin{aligned} &h(z(T)) = \sum_{i=1}^{2} (z_i(T) - \bar{z}_i(T))^2 , \\ &g(z^u(t), u(t)) = 0, \quad \forall t \in [0, T]. \end{aligned}$$

Hamilton–Jacobi–Bellman Equation

The time horizon is divided into N equally spaced intervals with δ = T/N. This converts the problem into the discrete-time domain and the dynamic programming approach can be applied. Once the approach is applied, the result is converted back into the continuous-time domain by taking the limit as $\delta \to 0$. The result is the following partial differential equation,

$$\begin{aligned} &0 = \min_{u \in U} \\ &\left[g(z, u) + \nabla_t J^*(t, z) + \nabla_x J^*(t, z)^\top f(z, u) \right], \\ &J^*(T, z) = h(z), \quad \forall z, \end{aligned} \tag{3}$$

where $J^*(t, z)$ is the optimal cost-to-go function. This equation is called the *Hamilton–Jacobi–Bellman equation*. It is also referred to as the *continuous-time analog of the dynamic programming equation*.

Pontryagin Minimum Principle

It is possible to derive the *Pontryagin minimum principle* using the Hamilton–Jacobi–Bellman equation given

above. Using the system given by (1), the classic principle results:

$$\dot{p}(t) = -\nabla_z H(z^*(t), u*(t), p(t)),$$
$$p(T) = \nabla h(z^*(T)),$$
$$H(z^*(t), u^*(t), p(t)) = g(z^*(t), u^*(t)) + p^\top(t) f(z^*(t), u^*(t)),$$
$$u^*(t) = \arg\min_{u \in U} H(z^*(t), u(t), p(t)),$$

where $z^*(t)$ and $u^*(t)$ are the optimal state and control trajectories, respectively.

A more detailed description of these two results are given in the following sections. For dynamic programming and optimal control problems, see [2] as well as the classic optimal control text [3].

See also

- Control Vector Iteration
- Duality in Optimal Control with First Order Differential Equations
- Dynamic Programming: Average Cost Per Stage Problems
- Dynamic Programming in Clustering
- Dynamic Programming: Discounted Problems
- Dynamic Programming: Infinite Horizon Problems, Overview
- Dynamic Programming: Inventory Control
- Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- Dynamic Programming: Optimal Control Applications
- Dynamic Programming: Stochastic Shortest Path Problems
- Dynamic Programming: Undiscounted Problems
- Hamilton–Jacobi–Bellman Equation
- High-order Maximum Principle for Abnormal Extremals
- Infinite Horizon Control and Dynamic Games
- MINLP: Applications in the Interaction of Design and Control
- Multi-objective Optimization: Interaction of Design and Control
- Multiple Objective Dynamic Programming
- Neuro-dynamic Programming
- Optimal Control of a Flexible Arm
- Optimization Strategies for Dynamic Systems
- Pontryagin Maximum Principle
- Robust Control
- Robust Control: Schur Stability of Polytopes of Polynomials
- Semi-infinite Programming and Control Problems
- Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- Suboptimal Control

References

1. Bellman R (1957) Dynamic programming. Princeton Univ. Press, Princeton
2. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
3. Bryson AE, Ho Y (1975) Applied optimal control. Hemisphere, Washington, DC
4. Pontryagin LS (1962) Ordinary differential equations. Addison-Wesley, Reading, MA

Dynamic Programming: Discounted Problems

Ioannis P. Androulakis
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L20, 90C39

Article Outline

Keywords
See also
References

Keywords

Dynamic programming; Infinite horizon problems; Discounted problem

Dynamic programming addresses models of decision making systems of an inherent sequential character. The problem of interest is defined as follows. We consider a discrete-time dynamic system:

$$x_{k+1} = f(x_k, u_k, \omega_k), \quad k = 0, 1, \ldots.$$

The state transitions, f, that define the evolution of the system from time k to time $k+1$ depend on the current state of the system, x_k, external disturbances, ω_k, which are considered to be random variables, and finally on a set of control, or policy, actions, u_k. The state of the system, x_k, $k = 0, 1, \ldots$, is an element of a space S, the control variables, u_k, $k = 0, 1, \ldots$, belong to space C, and the random external disturbance belongs to a countable space D. The control variables are such that: $u_k \in U(x_k) \subset C$, $k = 0, 1, \ldots$, and depend on the current state x_k, $k = 0, 1, \ldots$. The random disturbances, ω_k, $k = 0, 1, \ldots$, have identical, known, distributions which depend on the current state and control, $P(\omega_K \mid x_k, u_k)$. Note that ω_k does not depend on previous values of the disturbances, but may depend explicitly on the values of x_k, and u_k. Given an initial state x_0, the problem is to find a control law $\pi = \{\mu_0, \mu_1, \ldots\}$, belonging to the set of *admissible policies*, Π, which is the set of all sequences of functions $\pi = \{\mu_0, \mu_1, \ldots\}$ with:

$$\mu_k \colon S \to C, \quad \mu_x(x_k) \in U(x_k),$$
$$\forall x_k \in S, \quad k = 0, 1, \ldots,$$

that minimizes the cost functional:

$$J_\pi(x_0) = \lim_{N\to\infty} \mathsf{E}\left\{ \sum_{k=0}^{k=N-1} \alpha^k g_k(x_k, u_k, \omega_k) \right\}.$$

The optimal cost function J^* is thus defined as:

$$J^*(x) = \min_{\pi\in\Pi} J_\pi(x), \quad x \in S.$$

The cost, $J_\pi(x_0)$, for any $x_0 \in S$ and a given policy π, represents the limit of the expected finite horizon costs and these are well defined. The *discounted problems with bounded cost per stage* are such that the following assumption holds:

Assumption 1

1) $\forall(x, u, \omega) \in S \times C \times D$ the functions defining the cost per stage g are *uniformly* bounded:

$$0 \le |g_k(x_k, u_k, \omega_k)| \le M;$$

2) $M \in \mathbf{R}$, and $0 < \alpha < 1$.

This type of problem was first address through the pioneering work of D. Blackwell, [6]. The scalar, α, is the *discount factor*, and the range of its admissible values implies that future costs matter less that costs incurring at the present time, particularly when the cost per stage has a monetary interpretation. Mathematically, the presence of the discount factor guarantees the finiteness of the cost functional provided that the per stage costs are bounded uniformly. Furthermore, although the assumption of an infinite number of stages may never be satisfied in practice, it constitutes a reasonable approximation for problems involving a large number of stages. A rather typical example of a dis counted infinite horizon dynamic problem is the so-called *asset selling problem* where the reward for selling a particular asset at a given time k diminishes as time progresses.

For any function $J\colon S \to \mathbf{R}$ we define the operator $(T(\cdot))$ as:

$$(TJ)(x) = \min_{u\in U(x)} \mathsf{E}\{g(x, u, \omega) + \alpha J(f(x, u, \omega))\}.$$

This is in essence the function obtained when applying the standard dynamic programming mapping to J. Note that (TJ) represents essentially the optimal cost for a one-stage problem that has stage cost g and terminal cost αJ. For this operator, it can be shown, [4], that:

- For any bounded function J, the optimal cost function satisfies:

$$J^*(x) = \lim_{N\to\infty} (T^N J)(x), \quad \forall x \in S.$$

In other words, the *dynamic programming algorithm* converges to the optimal cost function. The above result relies on Assumption 1. It should be noted that the operator (TJ) can be shown to be:

1) *monotonic*:

$$J(x) \le J'(x) \;\Rightarrow\; (T^k J)(x) \le (T^k J')(x)$$

for any functions $J\colon S \to \mathbf{R}$ and $J'\colon S \to \mathbf{R}$ and,

2) *contractive*:

$$\max_{x\in S} \left| (T^k J)(x) - (T^k J'(x) \right| \le \alpha^k \max_{x\in S} \left| J(x) - J'(x) \right|$$

for any bounded functions $J\colon S \to \mathbf{R}$ and $J'\colon S \to \mathbf{R}$.

Both of these properties are important so as to show not only theoretical convergence of the dynamic programming algorithm but also to construct numerical solution schemes. Furthermore, the optimal cost function J^* can be shown to satisfy *Bellman's equation*, i. e., $\forall x \in S$:

$$J^*(x) = \min_{u \in U(x)} \mathsf{E}\{g(x,u,\omega) + \alpha J^*(f(x,u,\omega))\};$$

in other words: $J^* = TJ^*$. This proposition essentially defines the necessary and sufficient condition for the optimality of a policy μ, i. e. $\mu(x)$ is optimal if and only if it attains the minimum in Bellman's equation for every $x \in S$.

For the case where the state, control and disturbance space are finite, i. e., each set has a definite number of elements which can be found in principle, [9], several approaches exist for numerically solving the discounted problem with bounded cost per stage. It should be pointed out that under these conditions the problem is equivalent to a *finite-state Markov chain*. The first, *value iteration*, is based on a successive computation of TJ, T^2J, ..., since we know that $\lim_k \to \infty (T^kJ) = J^*$. Recall that the operator (TJ) is defined as the minimum over all possible disturbances with respect to the controls. Therefore, asymptotically we approach the optimal cost as well as the optima policy. Tight upper and lower bounds on the iterations can be derived, [3,4], which substantially improve the convergence rate of the successive approximations. More specifically, it can shown that for every vector J, state i, and time k:

$$(T^kJ)(i) + \underline{c}_k \le J^*(i) \le (T^kJ)(i) + \overline{c}_k,$$

where:

$$\underline{c}_k = \frac{\alpha}{1-\alpha} \min_{i=1,\dots,n} [(T^kJ)(i) - (T^{k-1}J)(i)],$$
$$\overline{c}_k = \frac{\alpha}{1-\alpha} \max_{i=1,\dots,n} [(T^kJ)(i) - (T^{k-1}J)(i)].$$

In fact, these error bounds can be used so as to further prove the finite convergence of the value iteration after $k < k'$ steps, $k' \in \mathbf{N}$. It can also be observed that instead of performing the value iteration simultaneously for all policies, one can perform the iteration in a *Gauss–Seidel* fashion, [10]. The contractive characteristics of the operator (FJ) make it possible to develop similar schemes. Instead of iterating on the operator (TJ), we define a new sequence based on the operator (FJ):

$$(FJ)(1) = \min_{u \in U(1)} \left[g(1,u) + \alpha \sum_{j=1}^{N} p_{1j}(u) J(j) \right],$$

$$(FJ)(i) = \lim_{u \in U(i)} \Bigg[g(i,u) + \alpha \sum_{j=1}^{i-1} p_{ij}(FJ)(j) + \alpha \sum_{j=i}^{n} p_{ij}(u) J(j) \Bigg],$$

$$i = 1, \dots, n\ .$$

In fact, when the error bounds are not used, a very interesting property can be shown, [4]:

- If J satisfies:

$$J(i) \le (TJ)(i) \le J^*(i), \quad i = 1, \dots, n,$$

then:

$$(T^kJ)(i) \le (F^kJ)(i) \le J^*(i),$$
$$i = 1, \dots, n; \quad k = 1, 2, \dots.$$

In other words, the Gauss–Seidel iteration converges faster than the ordinary, i. e., *Jacobi*, value iteration. An excellent treatment of the comparisons between Gauss–Seidel and Jacobi iterations and their parallel implementation can be found in [14]. Although the value iteration can be shown to be convergent even when the state and control spaces are infinite, the actual implementation can only proceed via approximations. In other words, instead of actually computing TJ we can only compute J', such that: $\max_{x \in S} |J'(x) - (TJ)(x)| \le \epsilon$. For such approximate methods to be in order, we do not necessarily need infinite spaces but even spaces with a very large number of states in which the actual computation is deemed inappropriate. Any function J' that satisfies the above criterion can in principle be used. Details regarding discretization approaches and computational techniques for addressing infinite state spaces can be found in [2,11].

The value iteration, thus far presented, is based on successive evaluations of the cost functions. Early on, [1], it was suggested that an alternate approach is to iterate on policies so as to generate sequences of stationary policies with improved, over the preceding one, costs. This method is know as the *policy iteration*. The method proceeds in three steps:

1) initialize control policy, u^0.
2) given a stationary policy, μ^k, evaluate the cost function J_{μ^k} by solving:

$$(I - \alpha P_{\mu^k}) J_{\mu^k} = g_{\mu^k}.$$

3) Obtain a new policy such that it satisfies: $T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}$.

In the above, the matrix P_μ is the *transition probability matrix* for a given stationary policy μ, given by:

$$P_\mu = \begin{pmatrix} p_{11}(\mu(1)) & \cdots & p_{1n}(\mu(1)) \\ \cdots & \cdots & \cdots \\ p_{n1}(\mu(1)) & \cdots & p_{nn}(\mu(n)) \end{pmatrix}$$

and g_μ the associated cost vector:

$$g_\mu = \begin{pmatrix} g(1, \mu(1)) \\ \cdots \\ g(n, \mu(n)) \end{pmatrix}.$$

Termination is detected once $J_{\mu^k} = T J_{\mu^k}$, i. e., a *fixed point* of the operator TJ has been identified. Notice that because of the assumption that the policy space is finite, the algorithm will terminate in a finite number of steps. Similarly to the value iteration, infinite state and control spaces pause problems when implementing policy iterations. Specifically, the policy evaluation and policy improvement steps can only be performed via approximations.

In [5] an *adaptive aggregation method* is proposed so as to address the issue of occasional slow convergence. The fundamental premise is to lump states of the original problem so as to generate a smaller dimension problem. In other words, the state space S is partitioned into smaller-dimensional spaces as: $S = S_1 \cup \cdots \cup S_m$. Given such a partitioning one can further define the transition probabilities for the aggregate states as:

$$r_{ij} = \sum_{s \in S_i} q_{is} \sum_{t \in S_j} p_{st}(\mu(s)),$$

which is the probability that the next state will belong to S_j given that the current state is S_i. q_{ij} are the elements of an $m \times n$ matrix Q, such that $q_{is} \neq 0$, if $s \in S$.

Finally, [7], noticed that since in the limit $J \leq J^* = TJ^*$, the optimal policy can be derived as the solution of the following *linear programming problem*:

$$\begin{cases} \max & \sum_{i \in S} \lambda_i \\ \text{s.t.} & \lambda_i \leq g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) \lambda_i, \\ & u \in U(i), \\ & i = 1, \ldots, n. \end{cases}$$

In the above formulation $p_{ij}(u)$ denote the transition probabilities: $p_{ij}(u) = \{ \mathrm{P}(x_{k+1} = j | \; x_k = i, \; u_k = u)\}$, i, $j \in S$, $u \in U(i)$. These can either be given or derived based on the discrete dynamic system, $x_{k+1} = f(x_k, u_k, \omega_k)$, and the known probability distribution $\mathrm{P}(\cdot | x, u)$ of the input disturbance ω_k. Linear programming formulations can also be used to derived cost and policy evaluation approximations. One possibility is to approximate J^* by a set of known basis functions as: $J'(x, r) = \sum_{k=1}^{m} r_k \omega_k(x)$. The vector r is an m-dimensional vector of known parameters, and for each state x we have chose a set of known scalars $\omega_k(x)$. The vector r can be determined as the solution of:

$$\begin{cases} \max & \sum_{x \in S'} J'(x, r) \\ \text{s.t.} & J'(x,r) \leq g(x,u) + \alpha \sum_{y \in S} p_{xy} J'(y, r), \\ & x \in S' \subset S, \\ & u \in U'(x) \subset U(x). \end{cases}$$

Furthermore, the cost function J_μ for a given policy μ can be approximated via linear programming formulations by identifying a vector r so as to minimize: $\max_{x \in S} | J'(x, r) - J_\mu(x)|$. This can be shown, [4], to be equivalent to solving:

$$\begin{cases} \min & z \\ \text{s.t.} & \left| J'(x,r) - g(x, \mu(x)) \right. \\ & \qquad \left. - \alpha \sum_{y \in S} p_{xy}(\mu(x)) J'(y, r) \right| \leq z, \\ & x \in S' \subset S. \end{cases}$$

Extensions of the general ideas are discussed in [12] where work on including constraints in the general formulation of the discounted dynamic programming problem is presented. Furthermore, [8] expanded the

scope of these models so as to address dynamic programming optimization problems involving multiple criteria by identifying the set of non-inferior, *Pareto optimal*, solutions.

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming

References

1. Bellman R (1957) Applied dynamic programming. Princeton Univ. Press, Princeton
2. Bertsekas DP (1975) Converence of discretization procedures in dynamic programming. IEEE Trans Autom Control AC-20:415–419
3. Bertsekas DP (1976) On error bounds for successive approximation methods. IEEE Trans Autom Control AC-21:396–396
4. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
5. Bertsekas DP, Castanon DA (1989) Adaptive aggregation methods for infinite horizon dynamic programming. IEEE Trans Autom Control AC-34:589–598
6. Blackwell D (1965) Discounted dynamic programming. Ann Math Statist 33:719–726
7. D'Epenoux F (1963) Sur un probleme de production et de stockage dans l'aleatoire. Managem Sci 10:98–108English transl.
8. Ghosh MK (1990) Markov decision processes with multiple costs. Oper Res Lett 9:257–260
9. Kolmogorov AN, Fomin SV (1970) Introductory real analysis. Dover, Mineola, NY
10. Kushner HJ (1971) Introduction to stochastic control. Holt, Rinehart and Winston, New York
11. Puterman ML (1978) Dynamic programming and its applications. Acad. Press New York, New York
12. Ross KW (1989) Randomized and past–dependent policies for Markov decision processes with multiple constraints. Oper Res 37:474–477
13. Shapley LS (1953) Stochastic games. Proc Nat Acad Sci USA 39
14. Tsitsiklis JN (1989) A comparison of Jacobi and Gauss–Seidel parallel iterations. Appl Math Lett 2:167–170

Dynamic Programming: Infinite Horizon Problems, Overview

Ioannis P. Androulakis
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L20, 90C39, 90C40

Article Outline

Keywords
See also
References

Keywords

Dynamic programming; Infinite horizon problems

Dynamic programming deals with situations where optimal decisions are being sought in systems operating in stages. Events occur in a specific order, such that the decision at time $k+1$ depends on the state of the system at time k. In general the key variables of the basic formulation are as follows:

- k represents discrete time;
- x_k represents the state of the system at time k;
- $\mu(x_k)$ represents the control, or decision, variable to be selected at time k;
- ω_k represents a random disturbance occurring at time k;
- N represents the time horizon.

Given the aforementioned variables, the basic dynamic programming formulation requires the following ingredients:

- a discrete-time dynamic system:

$$x_{k+1} = f_k(x_k, \mu_k, \omega_k);$$

- an additive cost function of the form:

$$g_N(x_N) + \sum_{k=1}^{k=N-1} g_k(x_k, \mu_k, \omega_k),$$

where g_k corresponds to the cost incurred at time k. One is therefore wishing to identify that control policy, $\pi = \{\mu_0, \ldots, \mu_{N-1}$, which minimizes the expected cost:

$$J^*(x_0) = \min_{\pi \in \Pi} J_\pi(x_0) = \mathsf{E}\left\{ g_N(x_N) + \sum_{k=1}^{k=N-1} g_k(x_k, \mu_k, \omega_k) \right\}.$$

The expectancy operator is needed since the presence of the random parameters ω_k the cost function becomes itself a random variable. As further complication, one might also minimize the expected cost not only for a given initial state of the system, x_0, but also with respect to all possible initial states.

Infinite horizon problems are further characterized by the fact that the number of stages N is infinite. In such a case, the cost functional over an infinite number of stages for a given control policy $\pi = \{\mu_0, \mu_1, \ldots\}$, and initial state x_0, is given by:

$$J_\pi(x_0) = \lim_{n \to \infty} \mathsf{E}\left\{ \sum_{k=1}^{k=N-1} \alpha^k g_k(x_k, \mu_k, \omega_k) \right\}.$$

The factor α is termed *discount factor* and is a positive scalar $0 < \alpha \leq 1$ which simply implies that future costs matter less than similar costs incurred at the present time. Infinite horizon problems are by definition the limit of the corresponding N-stage problem, as $N \to \infty$. Three points are pivotal in the analysis of infinite-dimensional dynamic programming problems:

- The optimal cost for the infinite horizon is the limit of the corresponding N-stage optimal cost, i. e., $J^* = \lim_{N \to \infty} J_N$.
- The optimal costs satisfy *Bellman's equation*, i. e.,

$$J^*(x) = \min_{u \in U(x)} \mathsf{E}\{g(x, \mu, w) + J^*(f(x, \mu, w))\}.$$

- If the optimal policy that correspond to the minimum of Bellman's equation is $\mu(x)$, then the policy $\pi = \{\mu, \mu, \ldots\}$ should be optimal.

The assumption of an infinite number of stages may not be satisfied in practice but is a very important one in terms of analyzing the asymptotic behavior of systems involving a finite but large number of stages. Depending on the nature of the cost per stage and the discount factor, the following categories of infinite horizon dynamic programming problems can be identified, [1]:

- *stochastic shortest path problems*: this problem is actually a generalization of the *deterministic shortest path problem* in the sense that we select not a successor but rather a probability distribution $p_{ij}(\mu)$. Obviously, if the probability $p_{ij}(\mu) = 1$ for a unique state j, then we recover the deterministic shortest path problem. One key feature of the stochastic shortest path problem is that the termination state t is cost-free termination state such that once the system reaches that state it never leaves from it. In other words, $p_{tt}(\mu) = 1$ and $g(t, \mu) = 0$, for all policies μ. In effect, the horizon is finite but the actual length is random. Furthermore, there exists at least one policy for which the destination state will be reached inevitably. A key assumption required for guaranteeing eventual termination states that there exists an integer m such that for every initial state and policy, there is a positive probability that the termination state will be reached after no more than m stages.
- *discounted problems with bounded cost per stage*: this type of infinite horizon dynamic programming encompasses problems for which:

$$|g(x, \mu, \omega)| \leq M, \ \forall (x, \mu, \omega) \in S \times C \times D,$$

i. e., there exists a finite scalar, M, that bounds the per stage cost. Furthermore, the discount factor is such that $0 < \alpha < 1$.
Both of these conditions are important so as to show that:

$$\lim_{K \to \infty} \mathsf{E}\left\{ \sum_{k=N}^{K} \alpha^k g(x_k, \mu(x_k), \omega_k) \right\} \to 0.$$

Boundedness and discounting results in successive approximation mappings which are *contraction mappings*, [2], thus proving the convergence of such schemes to the optimal solution of the discounted with bounded costs infinite horizon dynamic programming problems.
- *undiscounted problems*: this type of infinite horizon problems covers situations in which the discount

factor $\alpha = 1$, which greatly complicates the analysis. The key distinction is that the lack of a discount factor may result in infinite costs even when the per cost stage is bounded.

- *average cost per stage problems*: In cases where neither discounting nor a cost-free termination state exists, it is often meaningful to optimize the average per stage cost starting from state i.

$$J(i) = \lim_{N\to\infty} \frac{1}{N} \mathsf{E}\left\{ \sum_{k=1}^{N-1} g(x_k, \mu_k(x_k), \omega_k) \colon \; x_0 = i \right\}.$$

In essence what is assumed is that for most problems of interest the average and the optimal per stage cost are independent of the initial state. As a result, costs that incurred in the early stages do not matter since their contributions vanishes, i. e.,

$$\lim_{N\to\infty} \frac{1}{N} \mathsf{E}\left\{ \sum_{k=0}^{K} g(x_k, \mu_k(x_k), \omega_k) \right\} = 0.$$

For discrete state and transition spaces, it is helpful to consider the associated finite-state Markov chain. Let the state space S consist of n states, denoted by $1, \ldots, n$:

$$S = \{1, \ldots, n\}.$$

The transitions probabilities from state i to state j are:

$$p_{ij}(u) = \mathsf{P}(x_{k+1} = j | x_k = i, \; u_k = u), \qquad i, j \in S, \quad u \in U(i).$$

The dynamics of the state transitions $x_{k+1} = f(x_k, u_k, \omega_k)$ can actually be used to compute the state transitions. Given the above, the per stage expected cost can be expressed as: $g(i, u) = \sum_{j=1}^{n} p_{ij}(u)\, g'(i, u, j)$ Given the above definitions, a very important mapping can now be defined:

$$(TJ)(i) = \min_{u\in U(i)} \left[g(i, u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J(j) \right], \qquad i = 1, 2, \ldots,$$

and also

$$(T_\mu J)(i) = g(i, \mu(i)) + \alpha \sum_{j=1}^{n} p_{ij}(\mu(i)) J(j), \qquad i = 1, 2, \ldots,$$

This operator can actually be written as:

$$T_\mu J = g_\mu + \alpha P_\mu J.$$

Therefore, a stationary policy has a corresponding cost, J_μ, which is the solution to the equation:

$$(I - \alpha P_\mu) J_\mu = g_\mu.$$

Computationally, two major families of approaches exists for determining the optimal additive costs and the optimal policies. The first one, *value iteration*, is based on the idea of successive approximations. It can be shown, under conditions depending of the specific type of infinite horizon problem, that:

$$\lim_{k\to\infty} (T^k J)(i) = J^*(i).$$

This property essentially implies that the successive application of the mapping (TJ) will in the limit provide the optimal cost.

On the other hand *policy iteration* operates on the policy space and tries to identify a converging sequence of stationary policies converging to the optimal one. In all cases, the following basic three steps define the iteration:

- *Initialization*: guess an initial stationary policy, mu^0.
- *Policy evaluation*: given a stationary policy, μ^k, compute the corresponding cost function, J_{μ^k} from the system:

$$(I - \alpha P_{\mu^k}) J_{\mu^k} = g_{\mu^k}.$$

- Policy improvement: obtain a new stationary policy satisfying:

$$T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}.$$

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control

- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming
- ▶ Optimization Strategies for Dynamic Systems

References

1. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
2. Shapley LS (1953) Stochastic games. Proc Nat Acad Sci USA 39

Dynamic Programming: Inventory Control

Ioannis P. Androulakis
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L20

Article Outline

Keywords
See also
References

Keywords

Dynamic programming; Inventory control

Consider the problem of ordering a quantity of a certain item at each of the N periods so as to meet some stochastic demand. In mathematical terms the problem is defined as follows:

- x_k, the stock of a particular commodity available at the beginning of the kth period.
- u_k the stock to be ordered and immediately delivered at the beginning of the kth period.
- ω_k the demand during the kth period, whose probability distribution is assumed to be known.

The demand distributions are assumed to be independent random variables for each time period k. A simple stock balance at the beginning of each time period provides the description of the discrete-time evolution equation as:

$$x_{k+1} = x_k + u_k - \omega_k.$$

In other words, the state of the system (stock) at the beginning of period $k + 1$ was the state of the system (stock) at period k plus the ordered stock minus the demand at period k. The form of the replenishment of policy is very important and sits at the hart of the analysis of similar problems. The one just presented is, as will be seen, one of the two major assumptions regarding the stock balance equations. Given the above definitions, the cost incurred at period k has two components:

- a cost $r(x_k)$ representing either a penalty for positive stock, storage, or negative stocks, shortage for unfilled demand.
- a surcharging cost, cu_k, where c is the per unit surcharged cost.

The problem just described is known as the *inventory control problem*, one of the most important ones in the area of operations research. The preceding formulation illustrates the main characteristics of the inventory control problem:

- a discrete-time system that defines the system evolution in time of the form:

$$x_k = f_k(x_k, u_k, \omega_k);$$

- a set of independent random disturbances, representing commodity demands;
- a set of control constraints that depend on the state of the system at time k, x_k, that is $u_k \in U(x_k)$;
- a period of N time intervals over which the operating cost has an additive form as:

$$\mathsf{E}\left(g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, \omega_k)\right);$$

and, finally,

- we wish to optimally select the control actions at every time interval k, so as to optimize over all possible control policies the cost of operating the inventory system.

Clearly, the above definition of the inventory control problem, formulates the problem as *dynamic programming* problem in which we try to minimize an expected additive cost function.

Stochastic inventory problems were first considered by [6,10], were an abstract stochastic inventory model that allowed for possible constraints on the inventory after ordering were considered. In the literature of stochastic inventory models, there are two different assumptions about the excess demand unfilled from existing inventories: the *backlog assumption* and the *lost sales assumption*. These affect the form of the stock balance equation. The backlog assumption is historically more popular in the literature because of the inventory studies with spare parts inventory management problems. This assumption essentially states that an unfilled demand is being accumulated and satisfied at later times. The lost sales assumption states that unfilled demand is lost, which is the situation arising in retail establishments. Under either assumption, an important issue has been to establish the optimality of the (s, S)-type policy, [7]. It defines a very simple replenishment rule:

$$\mu^*(x_k) = \begin{cases} S_k - x_k, & x_k < s_k, \\ 0, & x_k \geq s_k. \end{cases}$$

The above rule is referred to as the (s, S) *policy*, implying that when the current level is less than the reorder point, s, an order up to the reorder level, S, has to be placed. Under an (s, S) policy if the inventory level at the beginning of a period is less than the reorder point s, then a sufficient quantity must be re-ordered to achieve an inventory level S upon replenishment. The key concept of *K-convexity*, [1], was instrumental in proving the optimality of the (s, S) policies. A real-valued function g is *K-convex*, where $K \geq 0$, if:

$$K + g(z + y) \geq g(y) + z\left(\frac{g(y) - g(y - b)}{b}\right),$$

$$\forall z \geq 0,\ b > 0,\ y.$$

The parameter K is the fixed cost associated with a positive inventory order:

$$C(u) = \begin{cases} K + cu, & u > 0, \\ 0, & u = 0. \end{cases}$$

The concept of K-convexity is one of the most important tools for the analysis of inventory control problem. It essentially expands the concept of convexity and is instrumental in proving the optimality of policies in inventory control problems. Regarding K-convexity, [2], the following hold true:

1) A real-valued convex function g is also 0-convex and hence also K-convex for all $K \geq 0$.
2) If $g_1(y)$ and $g_2(y)$ are K-convex and L-convex ($K \geq 0$, $L \geq 0$), respectively, then $\alpha g_1(y) + \beta g_2(y)$ is $(\alpha K + \beta L)$-convex for all $\alpha > 0$ and $\beta > 0$.
3) If $g(y)$ is K-convex and ω is a random variable, then $\mathsf{E}_\omega\{g(y-\omega)\}$ is also K-convex, provided $\mathsf{E}_\omega\{|g(y - \omega)|\} < \infty$, for all y.
4) If g is a continuous K-convex function and $g(y) \to \infty$ as $|y| \to \infty$, then there exist scalars s and S with $s \leq S$ such that:
   a) $g(S) \leq g(y)$, for all scalars y;
   b) $g(S) + K = g(s) < g(y)$, for all $y < s$;
   c) $g(y)$ is a decreasing function on $(-\infty, s)$;
   d) $g(y) \leq g(z) + K$, for all y, z with $z \leq y \leq z$.

If we further define a holding/storage cost as:

$$r(x) = p \max(0, -x) + h \max(0, x),$$

the function H as:

$$H(y) = p\mathsf{E}\left(\max(0, \omega_k - y)\right) + \mathsf{E}\left(\max(0, y - \omega_k)\right).$$

Application of the *dynamic programming algorithm* for zero final cost gives:

$$J_k(x_k) = \min \left\{ G_k(x_k), \min_{u_k > 0}\{K + cu_k + G_k(x_k + u_k)\} \right\},$$

with $G_k(x_k)$ defined as:

$$G_k(y) = cyH(y) + \mathsf{E}(J_{k+}1(y - \omega))$$

and $y_k = x_k + u_k$. Because of the K-convexity of G, it can actually be shown, [7], that the (s, S) policy is optimal. See [11] for the optimality of the (s, S) policy in the case of lost sales. For the case where the unfilled demand is not backlogged but rather lost, the system dynamic equation is defined as:

$$x_{k+1} = \max(0, x_k + u_k - \omega_k).$$

Additional K-convexity results and the optimality of the (s, S) for the case of lost-sales is presented and analyzed in [4]. Finite storage capacity in most real-life situations imposes an upper bound on theory that can be kept. The recent analysis of [3] considers the multi-product inventory model with stochastic demands and

warehousing constraints. This is a fairly general model in that it does not allow for surplus disposal, $u_k \geq 0$, and imposes constraints on the stored stock, $x_k + u_k \in \Gamma$.

Similar ideas pertain the analysis of inventory control problems over an infinite horizon. *Infinite horizon problems* need not necessarily correspond to physically realistic situations, but nevertheless, they define the vehicle for a thorough analysis of the asymptotic response of the inventory system. A discounted version of the backlogged problem can be stated as:

$$J_\pi(x_0) = \lim_{N\to\infty} \mathsf{E}\left(\sum_{k=1}^{N-1} \alpha^k (c\mu_k(x_k) + H(x_k + \mu(x_k) - \omega_k))\right),$$

where:

$$H(y) = p\max(0, -y) + h\max(0, y).$$

The case of $\alpha < 1$, i. e., a *discounted infinite horizon problem*, has also been analyzed, [8], and the existence of an optimal state-dependent (s, S)-type policy for problems with discounted costs was rigorously established.

The classical papers [5,10] were also devoted to stochastic inventory problems with the criterion of long-run average cost. In other words, one is interested in minimizing an average expected cost within an infinite horizon

$$J_\pi(x_0) = \lim_{N\to\infty} \frac{1}{N} \times \mathsf{E}\left(\sum_{k=1}^{N-1} \alpha^k (c\mu_k(x_k) + H(x_k + \mu(x_k) - \omega_k))\right).$$

This analysis also concludes that (s, S)-type policies are as well optimal for the long time average-return problem. The (s, S)-type of optimal policies are very important and they have be shown to be optimal for a wide variety of inventory problems including systems with continuous demands and discrete order sizes, [9], in other words for the cases where the orders u_k are assumed to be nonnegative integers, as well as the case where special structure in the form of periodicity of various components of the formulation such as demands, prices, and cost, [2].

Undoubtably, one of the most appealing features of inventory theory has been the fact that (s, S) policies are optimal for the class of dynamic inventory problems with random demands. However, real-life inventory problems impose constraints that make the assumption imposed on the analysis apparently too restrictive. The nature of demand, for instance, is an important factor in determining optimal policies. Classical models have assumed demand in each period to be a random variable independent of demands in other periods and of environmental factors at other times. Nevertheless, fluctuating economic conditions and uncertain market conditions can have a major effect. Furthermore, various constraints are observed in real life that limit the nature of ordering decisions nd inventory levels. The recent work of [8] addresses similar issues so as to incorporate cyclic or seasonal demand, as well as constraints imposed on the ordering periods, storage and service level constraints. Nevertheless, it is still shown that (s, S) policies are also optimal for these types of generalized models.

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming

References

1. Bertsekas DP (1976) Dynamic programming and stochastic control. Acad. Press, New York
2. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
3. Beyer D, Sethi SP, Sridhar R (1997) Stochastic multi–product inventory models with limited storage. Working Paper The Univ. Texas at Dallas, Richardson, TX

4. Cheng F, Sethi SP (1997) Optimality of state-dependent (*s*, *S*) policies with Markovian demand and lost sales. Production and Operations Management
5. Iglehart D (1963) Optimality of (*s*, *S*) policies in the infinite horizon dynamic inventory problem. Managem Sci 9: 259–267
6. Ignall EJ, Veinott A (1969) Optimality of myopic inventory policies for several substitue products. Managem Sci 18:284–204
7. Scarf H (1960) The optimality of (*s*, *S*) policies in the dynamic inventory problem. In: Arrow J, Karlin S, Suppes P (eds) Math. Methods in Social Sciences. Stanford Univ. Press, Palo Alto, CA
8. Sethi SP, Cheng F (1997) Optimality of (*s*, *S*) policies in invetory models with markovian demand. Oper Res 45: 931–939
9. Tsitsiklis JN (1984) Periodic review inventory systems with continuous demand and discrete order sizes. Managem Sci 10:1250–1254
10. Veinott A (1965) Optimal policy for a multi-product, dynamic nonstationary inventory problem. Managem Sci 12:206–222
11. Veinott A (1966) On the optimality of (*s*, *S*) inventory policies: New conditions and a new proof. SIAM J Appl Math 14:1067–1083

Dynamic Programming and Newton's Method in Unconstrained Optimal Control

JOSEPH C. DUNN
Math. Department, North Carolina State University, Raleigh, USA

MSC2000: 49M29, 65K10, 90C06

Article Outline

Keywords
Discrete-Time Optimal Control
Newton's Method
The Accessory Minimum Problem
Dynamic Programming Recursions
See also
References

Keywords

Dynamic programming; Newton method; Unconstrained optimal control

Discrete-Time Optimal Control

The cost function in the standard k-stage discrete-time optimal control problem is defined by

$$J = l_{k+1}(x_{k+1}) + \sum_{i=1}^{k} l_i(u_i, x_i) \tag{1}$$

and the recursion

$$\begin{cases} x_1 = a, \\ x_{i+1} = f_i(u_i, x_i), \quad i = 1, \ldots, k. \end{cases} \tag{2}$$

In these equations, u_i is a *control vector* in $\mathbf{R}^m$ and x_i is a *state vector* in $\mathbf{R}^n$. For each vector-valued k-tuple $u = (u_1, \ldots, u_k)$ in the direct sum $\mathbf{R}^{km} = \oplus_{i=1}^{k} \mathbf{R}^m$, there is a unique state vector-valued $(k+1)$-tuple $x(u) = (x_1(u), \ldots, x_{k+1}(u)) \in \mathbf{R}^{(k+1)n}$ satisfying (2), and a corresponding unique value $J(u)$ in (1). For present purposes, the state transition functions f_i, the terminal loss function l_{k+1} and the stage-wise loss functions l_i, $i = 1, \ldots, k$, are assumed to be twice continuously differentiable. The functions $x(\cdot)$ and $J(\cdot)$ are then also twice continuously differentiable, and Newton's method is formally applicable to the problem of minimizing $J(\cdot)$ over $\mathbf{R}^{km}$. Moreover, for fixed m and n the $km \times km$ linear system associated with the Newton iteration map for (1), (2) can be solved efficiently with *dynamic programming recursions* in $O(k)$ floating point operations as k increases without bound. In contrast, it requires $O(k^3)$ floating point operations to assemble and solve the Newtonian linear system for a general cost function J on $\mathbf{R}^{km}$ by standard Gaussian elimination methods.

The following discussion conforms to [4]. See [7] and [8] for an alternative development with connections to *differential dynamic programming*, and for a related but nonequivalent treatment of discrete-time optimal control problems based on the Riccati transformation. For analogous constructions in the setting of continuous-time optimal control problems, see [4] and the original papers [5] and [6]. For extensions to Newtonian projection methods and input-constrained optimal control problems, see [2] and [3].

Newton's Method

If J is any continuously differentiable real-valued function on $\mathbf{R}^N$ with global or local minimizing vectors $\overline{u}$,

then all such vectors must satisfy the *first order necessary condition*,

$$\nabla J(u) = 0, \tag{3}$$

where

$$\nabla J(u) = \left(\frac{\partial J}{\partial u_1}(u), \ldots, \frac{\partial J}{\partial u_N}(u) \right).$$

A solution of (3) is called a *stationary point*.

Condition (3) comprises N scalar equations in N scalar unknowns u_i. If J is a quadratic function, then (3) is a linear system which can be treated with standard elimination algorithms or other techniques capable of exploiting whatever structure may exist in the coefficient matrix for (3). On the other hand, if J is a nonquadratic nonlinear function, then (3) is a nonlinear system and iterative methods are generally needed to generate successive approximations to a solution of (3). One such method is Newton's recursive linearization scheme,

$$\begin{aligned} &u \to u + \overline{v}, \\ &\nabla J(u) + \nabla^2 J(u)\overline{v} = 0. \end{aligned} \tag{4}$$

When J is twice continuously differentiable, the vector-valued map $\nabla J(\cdot)$: $\mathbf{R}^N \to \mathbf{R}^N$ is continuously differentiable and its first differential at u is the Hessian operator $\nabla^2 J(u)$: $\mathbf{R}^N \to \mathbf{R}^N$ defined by

$$(\nabla^2 J(u)v)_i = \sum_{i=1}^{N} \frac{\partial^2 J}{\partial u_i \partial u_j}(u) v_j$$

for $i = 1, \ldots, N$. In such cases, (4) is formally applicable to the nonlinear system (3). Furthermore, if $\nabla^2 J(\overline{u})$ is invertible at a solution $\overline{u}$ for (3), then in some neighborhood N of $\overline{u}$, $\nabla^2 J(u)$ is also invertible and for each starting point $u^0 \in N$, the iteration (4) generates a sequence of vectors, $u^0, u^1, \ldots$, which remain in N and converge rapidly to $\overline{u}$. More precisely, either $u^i = \overline{u}$ eventually, or the errors $\|u^i - \overline{u}\| = (\langle u^i - \overline{u}, u^i - \overline{u}\rangle)^{\frac{1}{2}}$ satisfy the *superlinear convergence condition*,

$$\lim_{i \to \infty} \frac{\|u^{i+1} - \overline{u}\|}{\|u^i - \overline{u}\|} = 0. \tag{5}$$

A solution of (3) at which $\nabla^2 J(u)$ is invertible is said to be a *regular stationary point*. Note that solutions of (3) can be local maximizers or saddle points of J, and that regular points of this kind can also attract the Newton iterates. Hence for minimization problems, a simple steepest descent iteration is often employed at the outset to seek out likely starting points u^0 for (4) near some *regular local minimizer* for J.

If J is twice continuously differentiable, then every global or local minimizer $\overline{u}$ must also satisfy the *second order necessary condition*,

$$\forall v \in \mathbb{R}^N, \quad \langle v, \nabla^2 J(\overline{u})v \rangle \geq 0, \tag{6}$$

where $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner product,

$$\langle v, w \rangle = \sum_{i=1}^{N} v_i w_i.$$

In $\mathbf{R}^N$, $\overline{u}$ is therefore a regular local minimizer if and only if $\nabla^2 J(\overline{u})$ is *positive definite*, i. e.,

$$\forall v \in \mathbb{R}^N, \quad v \neq 0 \Rightarrow \langle v, \nabla^2 J(\overline{u})v \rangle > 0. \tag{7}$$

The gap between (6) and (7) is not large, hence regular local minimizers are commonly encountered in $\mathbf{R}^N$.

By continuity, property (7) extends to $\nabla^2 J(u)$ in some neighborhood of $\overline{u}$. At each fixed u in this neighborhood, the linear system in Newton's iteration (4) is equivalent to a corresponding unconstrained *accessory minimum problem*

$$\overline{v} \in \arg \min_{v \in \mathbb{R}^N} \phi(v) \tag{8}$$

with a strictly convex quadratic cost function

$$\phi(v) = \langle \nabla J(u), v \rangle + \frac{1}{2} \langle v, \nabla^2 J(u)v \rangle, \tag{9}$$

and a unique global minimizer $\overline{v}$. This equivalence is computationally significant for unconstrained minimization problems in general and discrete-time optimal control problems in particular.

The Accessory Minimum Problem

If J is the cost function of a discrete-time optimal control problem, then it can be shown that the accessory minimum problem (8)–(9) is also a discrete-time control problem with quadratic loss functions and linear state transition functions. Near a regular minimizer for J, this linear-quadratic (LQ) problem has a strictly

convex cost function that can be minimized with dynamic programming recursions. The required control-theoretic construction for ϕ and the related dynamic programming algorithms are outlined below.

In the following development, the symbol u may denote a vector in $\mathbf{R}^m$ or a vector-valued k-tuple in $\mathbf{R}^{km}$. Similarly, x may indicate a vector in $\mathbf{R}^n$ or a vector-valued $(k+1)$-tuple in $\mathbf{R}^{(k+1)n}$, and the bracket $\langle \cdot, \cdot \rangle$ may denote the Euclidean inner product in any of the spaces $\mathbf{R}^m$, $\mathbf{R}^n$, $\mathbf{R}^{km}$, or $\mathbf{R}^{(k+1)n}$. In each case, the correct interpretation is always clear from the context. Now suppose that $J(\cdot)$ is defined by (1)–(2) on $\mathbf{R}^{km}$, and fix $u \in \mathbf{R}^{km}$. Then for all $v \in \mathbf{R}^{km}$ the chain rule gives,

$$\begin{aligned}\langle \nabla J(u), v \rangle &= \frac{d}{ds} J(u + sv)|_{s=0} \\ &= \sum_{i=1}^{k+1} \langle \nabla_x l_i, y_i \rangle + \sum_{i=1}^{k} \langle \nabla_u l_i, v_i \rangle \end{aligned} \tag{10}$$

and

$$\begin{aligned}\left\langle v, \nabla^2 J(u) v \right\rangle &= \frac{d^2}{ds^2} J(u + sv)|_{s=0} \\ &= \sum_{i=1}^{k+1} \langle \nabla_x l_i, z_i \rangle + \sum_{i=1}^{k+1} \left\langle y_i, \nabla^2_{xx} l_i y_i \right\rangle \\ &\quad + 2 \sum_{i=1}^{k} \left\langle y_i, \nabla^2_{xu} l_i v_i \right\rangle + \sum_{i=1}^{k} \left\langle v_i, \nabla^2_{uu} l_i v_i \right\rangle, \end{aligned} \tag{11}$$

where

$$\begin{aligned} y_i &= \frac{d}{ds} x_i(u + sv)|_{s=0}, \\ z_i &= \frac{d^2}{ds^2} x_i(u + sv)|_{s=0}, \end{aligned} \tag{12}$$

and where all partial gradients and Hessians of l_{k+1} and l_i, $i = 1, \ldots, k$, are evaluated at $x_{k+1}(u) \in \mathbf{R}^n$ and $(u_i, x_i(u)) \in \mathbf{R}^m \oplus \mathbf{R}^n$, respectively.

Equations (2) and (12) and the chain rule also establish that y_i and z_i are recursively generated by the equations of variation,

$$\begin{cases} y_1 = 0, \\ y_{i+1} = A_i y_i + B_i v_i, \quad i = 1, \ldots, k, \end{cases} \tag{13}$$

and

$$\begin{cases} z_1 = 0, \\ z_{i+1} = A_i z_i \\ \quad + (C_i y_i) y_i + 2(D_i y_i) v_i + (E_i v_i) v_i \end{cases} \tag{14}$$

for $i = 1, \ldots, k$, with linear differential maps,

$$A_i = \frac{\partial f_i}{\partial x}, \qquad B_i = \frac{\partial f_i}{\partial u}$$

and

$$C_i = \frac{\partial^2 f_i}{\partial x \partial x}, \qquad D_i = \frac{\partial^2 f_i}{\partial x \partial u}, \qquad E_i = \frac{\partial^2 f_i}{\partial u \partial u}$$

evaluated at $(u_i, x_i(u))$. Useful control-theoretic representations for $\nabla J(u)$ and ϕ can now be constructed by removing y_i from formula (10) and z_i from formula (11) with the aid of an *adjoint recursion* for (13) and (14).

Equations (13) and (14) are special instances of

$$\begin{cases} w_1 = 0, \\ w_{i+1} = A_i w_i + \xi_i, \quad i = 1, \ldots, k, \end{cases} \tag{15}$$

with $w = (w_1, \ldots, w_{k+1}) \in \mathbf{R}^{(k+1)n}$ and $\xi = (\xi_1, \ldots, \xi_k) \in \mathbf{R}^{kn}$. For each ξ, there is a unique $w = \Phi\xi$ satisfying (15), and the resulting correspondence defines a linear map $\Phi: \mathbf{R}^{kn} \to \mathbf{R}^{(k+1)n}$. The map Φ has an associated *adjoint linear map* $\Phi^*: \mathbf{R}^{(k+1)n} \to \mathbf{R}^{kn}\}$ which, in principle, is uniquely determined by the requirement,

$$\langle \Phi^* \eta, \xi \rangle = \langle \eta, \Phi\xi \rangle, \tag{16}$$

imposed for all $\xi \in \mathbf{R}^{kn}$ and $\eta \in \mathbf{R}^{(k+1)n}$. The matrix representor for Φ^* in the standard basis for $\mathbf{R}^{(k+1)n}$ is obtained by transposing the analogous matrix representor for Φ; however, the adjoint map can also be computed directly with recursions derived from (15), *without prior construction of* Φ. More precisely, for each $\eta \in \mathbf{R}^{(k+1)n}$,

$$(\Phi^* \eta)_i = \psi_{i+1} \tag{17}$$

for $i = 1, \ldots, k$, where ψ is the unique solution of the adjoint recursion,

$$\begin{cases} \psi_{k+1} = \eta_{k+1}, \\ \psi_i = A_i^* \psi_{i+1} + \eta_i, \quad i = 1, \ldots, k. \end{cases} \tag{18}$$

To see this, note that if w and ψ are solutions of (15) and (18) respectively, then

$$\begin{aligned}\langle \eta_{k+1}, w_{k+1}\rangle &= \langle \psi_{k+1}, w_{k+1}\rangle - \langle \psi_1, w_1\rangle \\ &= \sum_{i=1}^{k}(\langle \psi_{i+1}, w_{i+1}\rangle - \langle \psi_i, w_i\rangle) \\ &= \sum_{i=1}^{k}\langle \psi_{i+1}, A_i w_i + \xi_i\rangle \\ &\quad - \sum_{i=1}^{k}\left\langle A_i^* \psi_{i+1} + \eta_i, w_i\right\rangle \\ &= \sum_{i=1}^{k}\langle \psi_{i+1}, \xi_i\rangle - \sum_{i=1}^{k}\langle \eta_i, w_i\rangle .\end{aligned}$$

Hence for all $\xi \in \mathbf{R}^{kn}$ and $\eta \in \mathbf{R}^{(k+1)n}$ condition (16) gives,

$$\langle \Phi^* \eta, \xi\rangle = \langle \eta, \Phi\xi\rangle = \langle \eta, w\rangle = \sum_{i=1}^{k}\langle \psi_{i+1}, \xi_i\rangle ,$$

and this establishes (17).

With the preceding formulas, it is now possible to write ϕ as a sum of linear and quadratic terms in the variables $v_1, \ldots, v_k$ and $y_1, \ldots, y_{k+1}$, with coefficients derived from the partial gradients and Hessians of *Hamiltonian functions*,

$$\begin{aligned}&H_i(u_i, x_i, \psi_{i+1}) \\ &\qquad = l_i(u_i, x_i) + \langle \psi_{i+1}, f_i(u_i, x_i)\rangle . \quad (19)\end{aligned}$$

Fix u and v in $\mathbf{R}^{km}$, let $\eta_i(u) = \nabla_x l_i$ for $i = 1, \ldots, k+1$, and let $\psi(u) \in \mathbf{R}^{(k+1)n}$ be the corresponding solution of the adjoint recursion,

$$\begin{cases}\psi_{k+1} = \nabla_x l_{k+1}, \\ \psi_i = A_i^* \psi_{i+1} + \nabla_x l_i, \quad i = 1, \ldots, k.\end{cases} \quad (20)$$

In addition, let y and z be the unique solutions of (13) and (14) respectively. Then with reference to (15)–(17) and (20),

$$\begin{aligned}\sum_{i=1}^{k+1}\langle \nabla_x l_i, y_i\rangle &= \sum_{i=1}^{k}\langle \psi_{i+1}, B_i v_i\rangle \\ &= \sum_{i=1}^{k}\left\langle B_i^* \psi_{i+1}, v_i\right\rangle\end{aligned}$$

and

$$\begin{aligned}\sum_{i=1}^{k+1}\langle \nabla_x l_i, z_i\rangle &= \sum_{i=1}^{k}\langle \psi_{i+1}, (C_i y_i) y_i\rangle \\ &\quad + 2\sum_{i=1}^{k}\langle \psi_{i+1}, (D_i y_i) v_i\rangle + \sum_{i=1}^{k}\langle \psi_{i+1}, (E_i v_i) v_i\rangle .\end{aligned}$$

When these expressions are substituted into (9)–(11), it follows from (19) that ϕ is prescribed by

$$\phi(v) = q_{k+1}(y_{k+1}) + \sum_{i=1}^{k} q_i(v_i, y_i) \quad (21)$$

and the recursions,

$$\begin{cases} y_1 = 0, \\ y_{i+1} = A_i y_i + B_i v_i, \quad i = 1, \ldots, k,\end{cases} \quad (22)$$

where A_i and B_i are the differential maps

$$A_i = \frac{\partial f_i}{\partial x}, \qquad B_i = \frac{\partial f_i}{\partial u}$$

as before, and the loss functions q are given by

$$q_{k+1}(y) = \frac{1}{2}\langle y, Q_{k+1} y\rangle$$

and

$$\begin{aligned}q_i(v, y) = \langle r_i, v\rangle + \frac{1}{2}\langle y, Q_i y\rangle \\ + \langle y, R_i v\rangle + \frac{1}{2}\langle v, S_i v\rangle ,\end{aligned}$$

for $i = 1, \cdots, k$, with

$$\begin{aligned}Q_{k+1} &= \nabla_{xx}^2 l_{k+1}, \\ r_i &= \nabla_u H_i,\end{aligned}$$

and

$$Q_i = \nabla_{xx}^2 H_i, \quad R_i = \nabla_{xu}^2 H_i, \quad S_i = \nabla_{uu}^2 H_i$$

for $i = 1, \ldots, k$. Moreover, the cost gradient $\nabla J(u)$ is separately recoverable from

$$\begin{aligned}\nabla J(u) &= (r_1, \ldots, r_k) \\ &= (\nabla_u H_1, \ldots, \nabla_u H_k) . \quad (23)\end{aligned}$$

In these equations, the Hessian of l_{k+1} is evaluated at $x_{k+1}(u)$ and the Hamiltonian gradients and Hessians are evaluated at $(u_i, x_i(u), \psi_{i+1}(u))$.

Dynamic Programming Recursions

Recall that $\nabla^2 J(u)$ is positive definite in some neighborhood of a regular local minimizer for J. When $\nabla^2 J(u)$ is positive definite, the quadratic accessory minimum problem with cost function (21)–(22) can be solved by dynamic programming techniques, which rest on a simple embedding scheme and a few elementary theorems stated below without proof. For a fuller discussion of dynamic programming, see [1].

For $j = 1, \ldots, k$ and $y \in \mathbf{R}^n$, define the family of cost functions $\phi_j(\cdot;y)$: $\mathbf{R}^{(k+1-j)m} \to \mathbf{R}^1$ by

$$\phi_j(v_j, \ldots, v_k; y) = q_{k+1}(y_{k+1}) + \sum_{i=j}^{k} q_i(v_i, y_i) \quad (24)$$

and the recursions,

$$\begin{cases} y_j = y \\ y_{i+1} = A_i y_i + B_i v_i, \quad i = j, \ldots, k, \end{cases} \quad (25)$$

where q_i, A_i and B_i are u-dependent entities defined as before. Evidently, the cost function ϕ in (21)–(22) is recovered from the equation,

$$\phi(v) = \phi_1(v_1, \ldots, v_k; 0). \quad (26)$$

Moreover, the cost functions ϕ_j are recursively generated by

$$\phi_k(v_k; y) = q_k(v_k; y) + q_{k+1}(A_k y + B_k v_k)$$

and

$$\phi_j(v_j, \ldots, v_k; y) = q_j(v_j; y) + \phi_{j+1}(v_{j+1}, \ldots, v_k; A_j y + B_j v_j),$$

for $j = k-1, \ldots, 1$. It is likewise readily seen that

$$\phi_j(v_j, \ldots, v_k; 0) = \phi(0, \ldots, 0, v_j, \ldots, v_k),$$
$$\nabla^2_{vv} \phi_j(v_j, \ldots, v_k; y) = \nabla^2_{vv} \phi_j(v_j, \ldots, v_k; 0),$$

and

$$\nabla^2_{vv} \phi(v) = \nabla^2 J(u)$$

for $j = 1, \ldots, k$, $v \in \mathbf{R}^{km}$ and $y \in \mathbf{R}^n$. Note also that since $\phi(\cdot)$ and $\phi_j(\cdot;y)$ are quadratic functions, their corresponding v-Hessians are independent of v as well as y. These facts and the basic principles of dynamic programming yield the following theorems.

Theorem 1 *The following statements are equivalent:*

1) *The quadratic function $\phi(\cdot)$ has a unique global minimizer $\overline{v} \in \mathbb{R}^{km}$.*
2) *$\nabla^2 J(u)$ is positive definite.*
3) *For all $j = 1, \ldots, k$, $\nabla^2_{vv} \phi_j$ is positive definite.*
4) *For all $j = 1, \ldots, k$ and all $y \in \mathbf{R}^n$, the quadratic function $\phi_j(\cdot;y)$ has a unique global minimizer $(\overline{v}_j, \ldots, \overline{v}_k) \in \mathbb{R}^{(k+1-j)m}$*

Theorem 2 *The following statements are equivalent:*

1) *For all $j = 1, \ldots, k$ and $y \in \mathbf{R}^n$,*

$$\phi^0_j(y) := \inf_{v_j, \ldots, v_k} \phi_j(v_j, \ldots, v_k; y) > -\infty, \quad (27)$$

2) *The real-valued functions $\phi^0_1(\cdot), \ldots, \phi^0_k(\cdot)$ satisfy the backward functional recursion,*

$$\phi^0_j(y) = \inf_{v \in \mathbb{R}^m} [q_j(v, y) + \phi^0_{j+1}(A_j y + B_j v)] \quad (28)$$

for $j = k, \ldots, 1$, with

$$\phi^0_{k+1}(y) := q_{k+1}(y).$$

Theorem 3 *Let r_i, A_i, B_i, Q_i, R_i, and S_i be the vectors and linear maps appearing in the representations (21)–(22) and (24)–(25) for the functions $\phi(\cdot)$ and $\phi_j(\cdot;y)$. Then the following statements are equivalent for all $\overline{v} = (\overline{v}_1, \ldots, \overline{v}_k) \in \mathbb{R}^{km}$:*

1) *$\overline{v}$ is the unique global minimizer for ϕ in $\mathbf{R}^{km}$, i. e.,*

$$\arg \min_{v \in \mathbb{R}^{km}} \phi(v) = \{\overline{v}\}.$$

2) *The vector $\overline{v} \in \mathbb{R}^{km}$ is generated by the forward recursions*

$$\begin{cases} y_1 = 0, \\ y_{j+1} = A_j y_j + B_j \overline{v}_j, \end{cases} \quad (29)$$

$$\{\overline{v}_j\} = \arg \min_{v \in \mathbb{R}^m} \left[q_j(v, y_j) + \phi^0_{j+1}(A_j y_j + B_j v) \right] = \{\gamma_j + \Gamma_j y_j\}, \quad (30)$$

for $j = 1, \ldots, k$, where

$$q_j(v, y) = \langle r_j, v \rangle + \frac{1}{2} \langle y, Q_j y \rangle + \langle y, R_j v \rangle + \frac{1}{2} \langle v, S_i v \rangle,$$
$$\phi^0_j(y) = \alpha_j + \langle \beta_j, y \rangle + \frac{1}{2} \langle y, \Theta_j y \rangle,$$

$S_j + B_j^* \Theta_{j+1} B_j$ is positive definite,

$$\gamma_j = -(S_j + B_j^* \Theta_{j+1} B_j)^{-1}(r_j + B_j^* \beta_{j+1}), \quad (31)$$

$$\Gamma_j = -(S_j + B_j^* \Theta_{j+1} B_j)^{-1}(R_j^* + B_j^* \Theta_{j+1} A_j), \quad (32)$$

and the linear maps Θ_j, vectors β_j, and numbers α_j satisfy the backward recursions,

$$\begin{cases} \Theta_{k+1} = 0, \\ \Theta_j = Q_j + A_j^* \Theta_{j+1} A_j \\ \quad + (R_j^* + B_j^* \Theta_{j+1} A_j)^* \Gamma_j, \end{cases} \quad (33)$$

$$\begin{cases} \beta_{k+1} = 0, \\ \beta_j = (A_j + B_j \Gamma_j)^* \beta_{j+1} + \Gamma_j^* r_j, \end{cases} \quad (34)$$

and

$$\begin{cases} \alpha_{k+1} = 0, \\ \alpha_j = \alpha_{j+1} - \frac{1}{2} \left\langle \gamma_j, (S_j + B_j^* \Theta_{j+1} B_j) \gamma_j \right\rangle \end{cases} \quad (35)$$

for $j = k, \ldots, 1$.

These theorems support the following efficient scheme for computing the Newton increment $\overline{v}$ in (4).

1. Given $u \in \mathbf{R}^{km}$, solve the forward recursion (2) for $x(u)$, and construct the corresponding linear maps A_j and B_j, and vectors $\eta_j = \nabla_x l_j$.
2. Solve the backward adjoint recursion (20) for $\psi(u)$ and construct the corresponding vectors r_j.
3. Construct the linear maps Q_i, R_i and S_i, solve the backward dynamic programming recursions (33) and (34) for Θ_j and β_j, and compute γ_j and Γ_j in (31) and (32).
4. Solve the forward recursions (29)–(30) for y and $\overline{v}$.

Algorithm

Stages 1 and 2 in the foregoing algorithm are always well-posed, and yield the cost gradient $\nabla J(u)$ (see (23)). The calculation for the Newton increment $\overline{v}$ is well-posed if and only if stage 3 produces invertible linear maps $S_j + B_j^* \Theta_{j+1} B_j$ for $j = k, \ldots, 1$. The calculation for $\overline{v}$ is well-posed *and* stage 3 concludes with k positive definite linear maps $S_j + B_j^* \Theta_{j+1} B_j$ if and only if $\nabla^2 J(u)$ is positive definite. If a positive semidefinite, indefinite or singular linear map $S_j + B_j^* \Theta_{j+1} B_j$ is encountered at some point in stage 3, it follows that $\nabla^2 J(u)$ is not positive definite and the accessory minimum problem may have no global minimizers or stationary points, or infinitely many such points. In such cases, it may be advantageous or even necessary to abort stage 3 and abandon Newton's method temporarily in favor of a descent iteration that employs the negative gradient $-\nabla J(u)$ computed in stages 1 and 2, or some other descent direction. Alternative *quasi-Newtonian* descent directions can be obtained by replacing S_j in stage 3 with $S_j + \lambda_j I$ where $\lambda_j I$ is a positive shift added where necessary to maintain positive definiteness of $S_j + \lambda_j I + B_j^* \Theta_{j+1} B_j$. This variant of stage 3 is automatically well-posed and produces the unique global minimizer $\overline{v}$ of the perturbed cost function,

$$\phi(v) + \frac{1}{2} \langle v, \Lambda(u) v \rangle ,$$

where

$$\Lambda(u) v = (\lambda_1 v_1, \ldots, \lambda_k v_k).$$

By construction, $\nabla^2 J(u) + \Lambda(u)$ is positive definite and

$$\overline{v} = -(\nabla^2 J(u) + \Lambda(u))^{-1} \nabla J(u).$$

Hence $\overline{v}$ is a descent vector for J. On the other hand, the simple steepest descent direction $-\nabla J(u)$ may be more cost-efficient, particularly when u is far from a local minimizer for J.

If the work required to compute the differentials for f_j and l_j in each time step j is uniformly bounded in j, with m and n fixed, then the number of arithmetic operations required to execute the foregoing algorithm (or its shifted variants) is directly proportional to k. This compares very favorably with the standard $O(k^3)$ estimate for general Newtonian calculations in $\mathbf{R}^{km}$.

Finally, references [9] and [10] revise the basic serial dynamic programming algorithm for parallel computation, and thereby achieve significant reductions in the *time* needed to calculate each Newton iteration.

See also

- ► Automatic Differentiation: Calculation of Newton Steps

- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ Interval Newton Methods
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming
- ▶ Nondifferentiable Optimization: Newton Method
- ▶ Optimal Control of a Flexible Arm
- ▶ Optimization Strategies for Dynamic Systems
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control
- ▶ Unconstrained Nonlinear Optimization: Newton–Cauchy Framework

References

1. Bellman R (1957) Dynamic programming. Princeton Univ. Press, Princeton
2. Bertsekas DP (1982) Projected Newton methods for optimization methods with simple constraints. SIAM J Control Optim 20:221–246
3. Dunn JC (1988) A projected Newton method for minimization problems with nonlinear inequality constraints. Numerische Math 53:377–409
4. Dunn JC, Bertsekas DP (1989) Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems. J Optim Th Appl 63: 23–38
5. Merriam CW (1964) An algorithm for the iterative solution of a class of two-point boundary value problems. SIAM J Control Optim Ser A 2:1–10
6. Mitter SK (1966) Successive approximation methods for the solution of optimal control problems. Automatica 3:135–149
7. Pantoja J (1988) Differential dynamic programming and Newton's method. Internat J Control 47:1539–1553
8. Polak E (1971) Computational methods in optimization. Acad. Press, New York
9. Wright SJ (1990) Solution of discrete-time optimal control problems on parallel computers. Parallel Comput 16:221–238
10. Wright SJ (1991) Partitioned dynamic programming for optimal control. SIAM J Optim 1:620–642

Dynamic Programming: Optimal Control Applications

Rein Luus
Department Chemical Engineering, University Toronto, Toronto, Canada

MSC2000: 93-XX

Article Outline

Keywords
Optimal Control Problem
Iterative Dynamic Programming
Early Applications of IDP
Choice of Candidates for Control
Piecewise Linear Continuous Control
Algorithm for IDP
Time-Delay Systems
State Constraints
Singular Control Problems
Sensitivity of Control Policy
Use of Variable Stage-Lengths
Nonseparable Problems
Future Directions
See also
References

Keywords

Optimal control; Optimization; Approximation algorithms; Singular control; Time-delay; Nonseparable problem; Sensitivity

A very powerful method for optimization of a system that can be separated into stages is *dynamic programming* developed by R. Bellman [1]. The main concept of this technique lies in the *principle of optimality* which can be stated as follows:

> An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Many engineering systems are in the form of individual stages, or can be broken into stages, so the idea of breaking up a complex problem into simpler subproblems so that optimization could be carried out systematically by optimizing the subproblems was received enthusiastically. Numerous applications of the principle of optimality in dynamic programming were given in [2], and there was a great deal of interest in applying dynamic programming to optimal control problems. In the 1960s many books and numerous papers were written to explore the use of dynamic programming as a means of optimization for optimal control problems. Since an optimal control problem, involving *optimization over a trajectory*, can be broken into a sequence of time stages, it appeared that dynamic programming would be ideally suited for such problems.

Although dynamic programming could be successfully applied to some simple optimal control problems, one of the greatest problems in using dynamic programming, however, was the *interpolation problem* encountered when the trajectory from a grid point did not reach exactly the *grid point* at the next stage [12]. This interpolation difficulty coupled with the dimensionality restriction and the requirement of a very large number of grid points limited the use of dynamic programming to only very simple optimal control problems. The limitations imposed by the '*curse of dimensionality*' and the '*menace of the expanding grid*' for solving optimal control problems kept dynamic programming from being used for practical types of optimal control problems, until R. Luus [14] suggested effective means of overcoming both the interpolation and the dimensionality problems.

Optimal Control Problem

We consider the continuous dynamic system described by the vector differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{1}$$

with the initial state $\mathbf{x}(0)$ given, where $\mathbf{x}$ is an $(n \times 1)$ state vector and $\mathbf{u}$ is an $(m \times 1)$ control vector bounded by

$$\alpha_j \le u_j(t) \le \beta_j, \quad j = 1, \ldots, m. \tag{2}$$

The *performance index* associated with this system is a scalar function of the state at the given final time t_f; i. e.,

$$I = \Phi(\mathbf{x}(t_f)). \tag{3}$$

We may have also state constraints, but for simplicity we shall leave these for later. The optimal control problem is to find the control $\mathbf{u}$ in the time interval $0 \le t < t_f$, so that the performance index in (3) is either minimized or maximized. To set up the problem into a staged form, we may approximate the optimal control problem by requiring a *piecewise constant control* policy instead of a continuous control policy, over P stages, each of length L, so that

$$L = \frac{t_f}{P}, \tag{4}$$

and we can consider the system at the grid points set up at these P stages. We may also use a piecewise linear approximation and the stages do not necessarily have to be of equal length.

Iterative Dynamic Programming

M. DeTremblay and Luus [10] suggested that instead of interpolation, an approximation can be used when the trajectory from a grid point does not reach a grid point at the next stage. They suggested that the control policy that was found to be optimal for the closest grid point be used to continue the integration to the next stage. If a large number of grid points are taken at each stage then a reasonable approximation may be ob-

tained, but the resulting control policy can still be quite far from the optimum. Therefore, this simplification by itself gives only a crude approximation.

However, by making a small change to the procedure, the *accuracy* with which the optimum is obtained can be improved substantially. This change requires the use of the procedure repeatedly in an iterative fashion [16], so that after every iteration, where the best value is used as the center point, the regions for the allowable values for control and for the grid points are reduced in size. The idea of *region reduction* in optimization was successfully used by Luus and T.H.I. Jaakola [37] in *direct search optimization*. As was shown by Luus [16], dynamic programming can be used in this fashion to give a sufficiently accurate optimal control policy. Although easy to program, the method was not computationally attractive until the idea of using *accessible states* as grid points [14]. With this latter change, the method was recognized as a *feasible approach* to solving optimal control problems.

The advantage of generating the state grid points is also that the dimensionality of the state vector then does not matter. The application of the method to a nonlinear system described by 7 differential equations and having 4 control variables was solved quite easily [15]. Also the method was used for system of *difference equations*, which is actually easier, since no discretization is necessary [40]. In essence, the 'curse of dimensionality' was eliminated and the new computational procedure became known as *iterative dynamic programming* (*IDP*).

Early Applications of IDP

Iterative dynamic programming provided a very convenient way of investigating the effect of the choice of the final time in optimal control problems [18]. However, by generating the grid points, it was no longer possible to guarantee a *global optimum*. This was illustrated by Luus and M. Galli [36]. Even the use of a very large number of grid points does not guarantee getting the global optimum. In fact, the number of grid points can be quite small in many cases and the global optimum is still obtained with good accuracy [4].

A very challenging problem is the bifunctional catalyst problem, where it is necessary to determine the blend of the catalyst along a tubular reactor to maximize the yield of a desired component [35]. By using *successive quadratic programming* (*SQP*) and starting from 100 randomly chosen starting points, 26 local optima were located, but the global optimum was not obtained. With IDP, however, the global optimum was readily obtained with the use of a single grid point [34]. To avoid the numerous local optima, all that was required for this system was to take a sufficiently large initial region size for the control.

Although the optimal control of *fed-batch reactors* was very difficult to obtain by methods based on *Pontryagin's maximum principle*, iterative dynamic programming provided a reliable means of obtaining the global optimum, and the results were even marginally better than had been previously reported [20,22]. The additional advantage of IDP is that the computations are straightforward and the algorithm can be easily programmed to run on a personal computer.

Choice of Candidates for Control

In the early work with IDP, the test values for the control variables were chosen over a uniform distribution. This was easy to program and was easy to visualize. For each control variable we could have a minimum of 3 values, namely $-r$, 0, and r, where r is the region size. For m control variables we must examine then 3^m candidates at each grid point. This is fine if m is less than 4, but if m is large, this number becomes excessively large.

An alternative method for choosing candidates for control was suggested by V. Tassone and Luus [47], but a better approach as shown by B. Bojkov and Luus [3] was to choose such candidates at random inside the allowable range. This meant that in theory there was no upper limit on m. Conceptually m could be greater than 100. In fact, IDP was used successfully on a system with 130 differential equations and 130 control variables [21] and later with 250 differential equations with 250 control variables [26].

Piecewise Linear Continuous Control

In the early work with IDP, the given time interval was divided into P time stages of equal length and at each time-stage we would have constant control. In many cases the optimal control policy is quite smooth, and therefore it may be beneficial to approximate the control policy by linear sections. This, indeed, gives a bet-

ter result with a smaller number of time stages as was shown by Luus [23], and allowed an optimal control policy for very high-dimensional systems to be determined accurately [15,26]. For a *piecewise linear control* we calculate the control policy in the time interval (t_k, t_{k+1}) by the expression

$$\mathbf{u}(t) = \mathbf{u}(k) + \frac{\mathbf{u}(k+1) - \mathbf{u}(k)}{L}(t - t_k), \tag{5}$$

where $\mathbf{u}(k)$ is the value of $\mathbf{u}$ at the time t_k and $\mathbf{u}(k+1)$ is the value of $\mathbf{u}$ at time t_{k+1}.

Algorithm for IDP

To illustrate the underlying logic in IDP, an algorithm is given to solve the optimal control problem as outlined in (1)-(4), where it is required to minimize the performance index in (3) with the use of piecewise constant control over P stages, each of same length:

1) Divide the time interval [0, t_f] into P time stages, each of length L.
2) Choose the number of test values for $\mathbf{u}$, denoted by R, an initial control policy and the initial region size $\mathbf{r}_{in}$; also choose the region contraction factor γ used after every iteration and the number of grid points N.
3) Choose the total number of iterations to be used in every pass and set the iteration number index to $j = 1$.
4) Set the region size vector $\mathbf{r}^{(j)} = \mathbf{r}_{in}$.
5) By using the best control policy (the initial control policy for the first iteration) as reference, integrate (1) from $t = 0$ to t_f N times with different values for control inside the allowable region to generate $N\mathbf{x}$-trajectories and store the values of $\mathbf{x}$ at the beginning of each time stage as grid points, so that $\mathbf{x}(k-1)$ corresponds to the value of $\mathbf{x}$ at beginning of stage k.
6) Starting at stage P, corresponding to time $t_f - L$, for each of the N stored values for $\mathbf{x}(P-1)$ from step 5 (grid points) integrate (1) from $t_f - L$ to t_f, with each of the R allowable values for the control vector calculated from

$$\mathbf{u}(P-1) = \mathbf{u}(P-1)^{*(j)} + \mathbf{D}\mathbf{r}^{(j)}, \tag{6}$$

where $\mathbf{u}(P-1)^{*(j)}$ is the best value obtained in the previous iteration and $\mathbf{D}$ is a diagonal matrix of different random numbers between -1 and 1. Out of the R values for the performance index, choose the control values that give the minimum value, and store these values as $\mathbf{u}(P-1)$. We now have the best control for each of these N grid points.
7) Step back to stage $P-1$, corresponding to time $t_f - 2L$, and for each of the N grid points do the following calculations. Choose R values for $\mathbf{u}(P-2)$ as in the previous step, and by taking as the initial state $\mathbf{x}(P-2)$ integrate (1) over one stage length. Continue integration over the last time stage by using the stored value of $\mathbf{u}(P-1)$ from step 6 by choosing the control policy corresponding to the grid point that is closest to the values of the state vector that has been reached. Compare the R values of the performance index and store the $\mathbf{u}(P-2)$ that gives the minimum value for the performance index.
8) Continue the procedure until stage 1, corresponding to the initial time $t = 0$ and the given initial state, is reached. This stage has only a single grid point, since the initial state is specified. As before, integrate (1) and compare the R values of the performance index and store the control $\mathbf{u}(0)$ that gives the minimum performance index. Store also the corresponding $\mathbf{x}$-trajectory.
9) Reduce the region for allowable control

$$\mathbf{r}^{(j+1)} = \gamma \mathbf{r}^{(j)}, \tag{7}$$

where j is the iteration number index. Use the best control policy from step 8 as the midpoint for the allowable values for the control denoted by the superscript $*$.
10) Increment the iteration index j by 1 and go to step 5 and continue the procedure for the specified number of iterations and interpret the results.

The application of this algorithm is illustrated with several examples in [33], where also the computer program in FORTRAN is given for IDP.

Time-Delay Systems

The great advantage of IDP over Pontryagin's maximum principle is that no auxiliary variables have to be calculated and no derivatives are required. The state equation is integrated forward and there is no need to integrate any equations backward. Therefore, the method is applicable to more complex systems, such as

time-delay systems. The initial attempt to apply IDP to time-delay systems was made by S.A. Dadebo and Luus [8]. By using piecewise linear continuous control very good results for a difficult nonlinear time-delay CSTR system were obtained by Luus et al. [43]. The method is further illustrated in [32].

State Constraints

Control constraints actually simplify the problem by decreasing the range over which the admissible values of control are to be taken. Research in how to handle *state constraints* is still continuing, but already very useful results have been obtained. As was shown in [39] and [17], the use of penalty functions appears to be the best way to deal with state constraints. The best type of penalty function has not yet been firmly established. Although Dadebo and K.B. McAuley [9] suggested the use of absolute value type of penalty function for state equality constraints, the recent work of Luus [27], and Luus and C. Storey [41] show that a quadratic penalty function with shifting terms also works very well. The advantage of using the quadratic penalty function with shifting terms is that, at the optimum, the shifting terms yield useful sensitivity information with respect to the constraints. Handling of state inequality constraints can be achieved by introducing through differential equations auxiliary variables that are increased in value whenever the constraint is violated and then including these auxiliary variables at the final time as penalty functions in the augmented performance index [46]. The use of differential equations is better than the use of difference equations as was used by Luus [17], because this will prevent violation of the constraint inside a time stage. The auxiliary variables when incorporated into the augmented performance index through a penalty function with a sufficiently large penalty function factor thus prevent a violation of the state constraint anywhere in the time interval.

Singular Control Problems

When Pontryagin's maximum principle is used, computational difficulties arise if the Hamiltonian is not an explicit function of the control for a portion of the trajectory. Such problems do not arise when IDP is used, and therefore this area was investigated by using IDP. The early work [19] showed that IDP can be used without much difficulty for such problems, and Luus [29] was able to obtain solutions to *singular control problems* that had eluded many investigators. For these problems the main difficulty is the very low *sensitivity* of the performance index on control.

Sensitivity of Control Policy

Especially for batch reactors, it is found that the cause of computational difficulties lies in the sensitivity of control policy with respect to the yield that is to be maximized [24]. Whereas we are not concerned with more than four figure accuracy in the yield, we would nevertheless like to know what the optimal control policy is. The very low sensitivity was brought out by Luus [25] where in the optimal control of a fed-batch reactor, it was shown that the optimal control policy is relatively smooth.

Use of Variable Stage-Lengths

Bojkov and Luus [5,6] suggested the use of flexible stage-lengths in IDP for *time optimal control problems* where the time of *switching* is very important. For general type of optimal control problems the use of *variable stage-lengths* enabled the optimum to be more accurately obtained, and in some instances the local optima encountered with the use of stages of fixed length could be avoided [7]. The problem of applying this idea to problems where the final time was specified was overcome by the use of shifting terms in a quadratic penalty function [27]. The use of flexible stage-lengths provides a means of obtaining accurate *switching times* and allowed some optimal control problems, that had gone by unsolved for several decades, to be readily solved [29]. The use of variable stage lengths and a quadratic penalty function with shifting terms enables time optimal control problems to be solved directly [31], so that a difficult boundary value problem is avoided. Further illustration of the usefulness of variable stage lengths is given in [30] and [33].

Nonseparable Problems

Problems where the performance index is a function of all the control variables and states, and where separation into stages as required for dynamic programming is not possible, constitutes and interesting class

of problems. D. Li and Y.Y. Haimes [13] suggested a method of tackling such problems, and Luus and Tassone [42] considered the application of IDP to *nonseparable problems*. Luus [28] showed that even for complex nonseparable problems a large number of grid points is not necessary, and the optimum can be obtained quite readily. If the number of stages is large, then IDP has a great advantage over direct search optimization where optimization is carried out simultaneously over all the stages. The best means for the application of IDP to nonseparable problems are still to be determined. The strategy of using the values for some of the variables from previous iteration appears to work very well, however.

Future Directions

Iterative dynamic programming has been developed into a useful optimization procedure. As has been shown in [11], IDP has certain advantages over other optimization procedures for the optimization of a fed-batch reactor. The *reliability* of getting the *global optimum* is very high. Now that the personal computers have become very powerful, the method can be easily used on very complex optimal control problems. When G. Marroquin and W.L. Luyben [44] suggested operating a batch reactor at its best isothermal temperature as the set point, the computational power of the existing computers was relatively low and the cost of computation was very high. It appeared then that optimal control could not be used for realistic systems. Now, however, we can, in effect, have a *feedback control* if the measurements of the pertinent state variables can be done sufficiently fast, by solving the optimal control problem many times during the time of operation of the batch reactor. If the trend in the enhancement of computer speed continues, we can use realistic models and carry out optimization 'on-line', so that optimal control calculations can be carried out during the operation and the required changes in the control can be immediately implemented. Then the optimal control's application will not be only for investigation of design possibilities, but will constitute an important part of the actual operation of the process.

The viability of using IDP for on-line optimal control has been illustrated for reactor control by Luus and O.N. Okongwu [38].

Since derivatives are not required in the use of IDP, the method is applicable to more general types of optimal control problems. Also, since no *auxiliary variables* are necessary, except to handle state inequality constraints, the method is easier to use than variational methods based on Pontryagin's maximum principle. As convergence properties of IDP are studied in greater detail, further improvements will inevitably be introduced, to make IDP even more useful. Luus [30] showed that variable stage lengths can be incorporated into optimal control problems where *state inequality constraints* are also present, by combining the approach of Bojkov and Luus [7] along with that of W. Mekarapiruk and Luus [46]. Although the best choice for the *penalty function* to be used in IDP has not yet been established, good progress has been made in this field [45] and further research in this area is continuing. Furthermore, since no *derivatives* are required for IDP, the method should have important applications where *nondifferentiable functions* are encountered.

See also

- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Average Cost per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-Time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control

- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-Dynamic Programming
- ▶ Optimal Control of a Flexible Arm
- ▶ Optimization Strategies for Dynamic Systems
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-Infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Bellman R (1957) Dynamic programming. Princeton Univ. Press, Princeton
2. Bellman R, Dreyfus S (1962) Applied dynamic programming. Princeton Univ. Press, Princeton
3. Bojkov B, Luus R (1992) Use of random admissible values for control in iterative dynamic programming. Ind Eng Chem Res 31:1308–1314
4. Bojkov B, Luus R (1993) Evaluation of the parameters used in iterative dynamic programming. Canad J Chem Eng 71:451–459
5. Bojkov B, Luus R (1994) Time-optimal control by iterative dynamic programming. Ind Eng Chem Res 33:1486–1492
6. Bojkov B, Luus R (1995) Time optimal control of high dimensional systems by iterative dynamic programming. Canad J Chem Eng 73:380–390
7. Bojkov B, Luus R (1996) Optimal control of nonlinear systems with unspecified final times. Chem Eng Sci 51:905–919
8. Dadebo S, Luus R (1992) Optimal control of time-delay systems by dynamic programming. Optimal Control Appl Meth 13:29–41
9. Dadebo SA, McAuley KB (1995) Dynamic optimization of constrained chemical engineering problems using dynamic programming. Comput Chem Eng 19:513–525
10. DeTremblay M, Luus R (1989) Optimization of non-steady-state operation of reactors. Canad J Chem Eng 67:494–502
11. Hartig F, Keil FJ, Luus R (1995) Comparison of optimization methods for a fed-batch reactor. Hungarian J Ind Chem 23:141–148
12. Lapidus L, Luus R (1967) Optimal control of engineering processes. Blaisdell, Waltham, pp 84–86
13. Li D, Haimes YY (1990) New approach for nonseparable dynamic programming problems. JOTA 66:311–330
14. Luus R (1989) Optimal control by dynamic programming using accessible grid points and region reduction. Hungarian J Ind Chem 17:523–543
15. Luus R (1990) Application of dynamic programming to high-dimensional nonlinear optimal control problems. Internat J Control 52:239–250
16. Luus R (1990) Optimal control by dynamic programming using systematic reduction in grid size. Internat J Control 19:995–1013
17. Luus R (1991) Application of iterative dynamic programming to state constrained optimal control problems. Hungarian J Ind Chem 19:245–254
18. Luus R (1991) Effect of the choice of final time in optimal control of nonlinear systems. Canad J Chem Eng 69: 144–151
19. Luus R (1992) On the application of iterative dynamic programming to singular optimal control problems. IEEE Trans Autom Control 37:1802–1806
20. Luus R (1993) Application of dynamic programming to differential-algebraic process systems. Comput Chem Eng 17:373–377
21. Luus R (1993) Application of iterative dynamic programming to very high-dimensional systems. Hungarian J Ind Chem 21:243–250
22. Luus R (1993) Optimization of fed-batch fermentors by iterative dynamic programming. Biotechnol and Bioengin 41:599–602
23. Luus R (1993) Piecewise linear continuous control by iterative dynamic programming. Ind Eng Chem Res 32:859–865
24. Luus R (1994) Optimal control of batch reactors by iterative dynamic programming. J Process Control 4:218–226
25. Luus R (1995) Sensitivity of control policy on yield of a fed-batch reactor. Proc. IASTED Internat. Conf. on Modelling and Simulation, Pittsburgh, PA, April 27-29, 1995, pp 224–226
26. Luus R (1996) Numerical convergence properties of iterative dynamic programming when applied to high dimensional systems. Chem Eng Res Des 74:55–62
27. Luus R (1996) Use of iterative dynamic programming with variable stage lengths and fixed final time. Hungarian J Ind Chem 24:279–284
28. Luus R (1997) Application of iterative dynamic programming to optimal control of nonseparable problems. Hungarian J Ind Chem 25:293–297
29. Luus R (1997) Use of iterative dynamic programming for optimal singular control problems. Proc. IASTED Internat. Conf. on Control, Cancun, Mexico, May 28-31, 1997, pp 286–289
30. Luus R (1997) Use of variable stage-lengths for constrained optimal control problems. Hungarian J Ind Chem 25: 299–304
31. Luus R (1998) Direct approach to time optimal control by iterative dynamic programming. Proc. IASTED Internat. Conf. on Intelligent Systems and Control, Halifax, Nova Scotia, Canada, June 1-4, 1998, pp 121–125
32. Luus R (1998) Iterative dynamic programming: from curiosity to a practical optimization procedure. Control and Intelligent Systems 26:1–8

33. Luus R (2000) Iterative dynamic programming. Chapman and Hall/CRC, London
34. Luus R, Bojkov B (1994) Global optimization of the bifunctional catalyst problem. Canad J Chem Eng 72:160–163
35. Luus R, Dittrich J, Keil FJ (1992) Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor. Canad J Chem Eng 70:780–785
36. Luus R, Galli M (1991) Multiplicity of solutions in using dynamic programming for optimal control. Hungarian J Ind Chem 19:55–62
37. Luus R, Jaakola THI (1973) Optimization by direct search and systematic reduction of the size of search region. AIChE J 19:760–766
38. Luus R, Okongwu ON (1999) Towards practical optimal control of batch reactors. Chem Eng 75:1–9
39. Luus R, Rosen O (1991) Application of iterative dynamic programming to final state constrained optimal control problems. Ind Eng Chem Res 30:1525–1530
40. Luus R, Smith SG (1991) Application of dynamic programming to high-dimensional systems described by difference equations. Chem Eng Techn 14:122–126
41. Luus R, Storey C (1997) Optimal control of final state constrained systems. Proc. IASTED Internat. Conf. on Modelling, Simulation and Control, Singapore, Aug. 11-13, 1997, pp 245–249
42. Luus R, Tassone V (1992) Optimal control of nonseparable problems by iterative dynamic programming. Proc. 42nd Canad. Chemical Engin. Conf., Toronto, Canada, October, 18-21, 1992, pp 81–82
43. Luus R, Zhang X, Hartig F, Keil FJ (1995) Use of piecewise linear continuous control for time-delay systems. Ind Eng Chem Res 34:4136–4139
44. Marroquin G, Luyben WL (1973) Practical control studies of batch reactors using realistic mathematical models. Chem Eng Sci 28:993–1003
45. Mekarapiruk W, Luus R (1997) Optimal control of final state constrained systems. Canad J Chem Eng 25:806–811
46. Mekarapiruk W, Luus R (1997) Optimal control of inequality state constrained systems. Ind Eng Chem Res 36:1686–1694
47. Tassone V, Luus R (1993) Reduction of allowable values for control in iterative dynamic programming. Chem Eng Sci 48:3864–3867

Dynamic Programming: Stochastic Shortest Path Problems

Ioannis P. Androulakis
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L20, 90C40

Article Outline

Keywords
See also
References

Keywords

Dynamic programming; Infinite horizon problems; Stochastic shortest path

The *shortest path problem* is considered to be one of the classical and most important combinatorial optimization problems. Given a directed graph and a length α_{ij} for each arc (i, j), the problem is to find a path of minimum length that leads from any node i to a node t, called the destination node. So, for each node i, we need to optimally identify a successor node $u(i)$ so as to reach the destination at the minimum sum of arc lengths over all paths that start at i and terminate at t. Of particular relevance is, in the area of distributed computation, the problem of data routing within a computer communication network. In such a case, the cost associated with a particular link (i, j) is related to an average delay. The *stochastic shortest path problem* is a generalization whereby for each node i we must select a probability distribution over all possible successor nodes j out of a given set of probability distributions $p_{ij}(u)$, parameterized by a control $u \in U(i)$. Clearly, the path traversed and its length are random variables, but the optimal path should lead to the destination with probability 1 and have the minimum expected length. Furthermore, if the probability distributions are such that they assign a probability of 1 to a single successor we then recover the deterministic shortest path problem. Clearly, sequential decisions have to be made optimally so as to determine the sequence of controls that would produce for any current state, i. e. node, a successor state, i. e. node, so as to minimize the expected length for reaching the terminal state. If we were to assume that a particular policy π, i. e., set of control actions, has been selected the total expected cost starting from an initial state i, using this policy would be:

$$J^{\pi}(i) = \lim_{N\to\infty} \mathsf{E}\left\{\sum_{k=0}^{N-1} \alpha^k g(i_k, \mu_k(i), i_{k+1}) : \; i_0 = i\right\} .$$

The optimal cost-to-go starting from state i is denoted by

$$J^* = \min_{\pi} J^{\pi}(i).$$

This problem is defined as s special case of the *total cost infinite horizon problem*. The need to assume an infinite horizon is not required by the actual description of the problem, but rather it is a necessity since the actual length is random as well as unknown. The following are the key characteristics of the stochastic shortest path problem description within the infinite horizon dynamic programming framework:

1) There is no discounting, $\alpha = 1$.
2) The state space is $S = \{1, \ldots, n, t\}$.
3) The transitions probabilities are:

$$p_{ij}(u) = = \mathsf{P}(x_{k+1} = j | x_k = i,\ u_k = u),$$
$$i, j \in S, \quad u \in U(i).$$

4) The state t is absorbing, that is,

$$p_{tt}(u) = 1, \quad \forall u \in U(t);$$

the state t, the termination state, is special in the sense that reaching it is inevitable.

5) The control set $U(i)$ is finite.
6) The destination is cost-free, i. e.,

$$g(t, u, t) = 0, \quad \forall u \in U(t).$$

If we denote by $g'(i, u, j)$ the cost of moving from i to j using control u, then the expected cost per stage will be defined as:

$$g'(i, u) = \sum_{j=1}^{n} p_{ij}(u) g(i, u, j).$$

The concept of the absorbing state implies that this state will either be reached inevitably, or there is an incentive to reach it with the minimum expected cost. The stochastic shortest path problem was first formulated in [3] while addressing a fundamental problem in control theory, namely finding the input that would take a given system to a specified terminal state at minimum cost. A fundamental assumption regarding the types of stochastic shortest path problems that can be analyzed states that:

Assumption 1 There exists at least one proper policy.

A *proper policy* μ is a *stationary policy* which, when used, results in a positive probability that the destination state will be reached after at most n stages, regardless of the initial state. That is:

$$\rho_{\mu} = \max_{i=1,\ldots,n} \mathsf{P}\{x_n \neq t :\ x_0 = i,\ \mu\} < 1.$$

A stationary policy is a policy of the form $\pi = \{\mu, \mu, \ldots\}$.

For analysis purposes, the following operator for any vector J is defined:

$$(TJ)(i) = \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^{n} p_{ij}(u) J(j) \right], \quad i = 1, \ldots, n,$$

which is obtained by applying one iteration of the basic *dynamic programming algorithm* to the cost function J, by realizing that the expectancy operator can be reformulated based on the functional form of the state transition probabilities p_{ij}. It can actually be shown, [1], that T is a *contraction mapping* with respect to a *weighter sup norm*. In other words there exist positive constants $\nu_1, \ldots, \nu_n$, and some γ with $0 < \gamma < 1$, such that for all J_1, J_2:

$$\max_{i=1,\ldots,n} \frac{1}{\nu_i} |(TJ_1)(i) - (TJ_2)(i)| \leq \gamma \max_{i=1,\ldots,n} \frac{1}{\nu_i} |J_1(i) - J_2(i)|.$$

Furthermore, the operator T is *monotone*, that is: for any vector J and $\bar{J}$ such that $J(i) \leq \bar{J}(i)$, $i = 1, \ldots, n$, and for any stationary policy μ we have:

$$(T^k J)(i) \leq (T^k \bar{J})(i),$$
$$(T_{\mu}^k J)(i) \leq (T_{\mu}^k \bar{J})(i),$$
$$i = 1, \ldots, n, \quad k = 1, 2, \ldots.$$

The main results of the theoretical analysis of stochastic shortest path problems are analogous to those for *discounted problems*:

i) The optimal cost vector is a solution to *Bellman's equation*: $J^* = TJ^*$.
ii) For every proper policy the cost vector J satisfies:

$$\lim_{k \to \infty} (T^k J)(i) = J^*(i), \quad i = 1, \ldots, n.$$

iii) A stationary policy μ is optimal if and only if $T_\mu J^* = TJ^*$.

iv) For every proper policy μ:

$$\lim_{k\to\infty} T_\mu^k J = J^\mu,$$
$$J^\mu = T_\mu J^\mu.$$

A thorough analysis of the computational complexity of stochastic shortest path problems has been presented in [5].

In order to address computationally stochastic shortest path problems the general methods, i. e., *value iteration*, and *policy iteration*, as well as approximation schemes along the lines presented in [1] as developed for discounted problems. A detailed account can also be found in [6]. Value iteration is a principal method for calculating optimal cost J^* by generating sequences $T^k J$ starting from some J. Issues related to the *Gauss–Seidel* implementation of the value iteration are discussed in [2]. Although in principle an infinite number of iterations will be required, under certain conditions finite convergence can be achieved. An alternative way is to perform policy iterations, in the sense that starting with a proper policy μ_0, a sequence of policies converging to the optimal one is constructed. According to property iv), for any given policy μ, the cost vector can be evaluate as the solution of a system of linear equations:

$$J(i) = \sum_{j=1}^{n} p_{ij}(\mu_k(i))(g(i,\mu_k(i),j) + J(j)), \qquad i = 1,\ldots,n.$$

A policy improvement can be know determined as in:

$$\mu_{k+1}(i) = \arg\min_{u\in U(i)} \sum_{j=0}^{n} p_{ij}(u)(g(i,u,j) + J^{\mu_k}(j)).$$

These approaches assume that mathematical models for the cost structure and the transition probabilities of the system exist. In may cases however, such information is not available and methods based on simulation have been developed. This information can be derived by simulating, for given control and state spaces, the system's response so as to derive the associated transition costs $g(i,u,j)$. The ideas of *Monte-Carlo simulation* can be utilized so as to use simulation for policy evaluations. A straightforward way of computing the corresponding cost vector J_μ for a given policy μ, is to generate many sample trajectories starting at i, average the corresponding costs, therefore obtaining an estimate for $J_\mu(i)$. An alternative way, is to perform an infinite (large) number of simulation runs from various initial states up to the destination state, and any time that state i is encountered we record the corresponding cost of reaching state t:

$$c(i,m) = g(i,i_1) + g(i_1,i_2) + \cdots + g(i_N,t).$$

By averaging the simulations we obtain:

$$J_\mu(i) = \lim_{M\to\infty} \frac{1}{M} \sum_{m=1}^{M} c(i,m).$$

The iterative implementation of the update process results in:

$$\begin{aligned} J_\mu(i_k) = J_\mu(i_k) &+ \gamma_k \left(g(i_k,i_{k+1}) + g(i_{k+1},i_{k+2})\right. \\ &\left. + \cdots + g(i_N,t) - J_\mu(i_k)\right), \\ k &= 1,\ldots,N, \\ \gamma_k &= \frac{1}{m}, \quad m = 1,2,\ldots \end{aligned}$$

Using simulation to perform the policy evaluation as just described, can be utilized so as to improve on the actual policies in order to achieve optimality. The concept of temporal differences, [7], was proposed recently as an alternative way so as to develop policy iterations, [1,2]. This concept originated in the field of reinforcement learning, and the key premise is to adjust the estimations appropriately so as to modify prior predictions when a temporal difference if observed, by essentially looking back in time and correcting previous predictions. The *temporal difference* is defined as the quantity:

$$d_k = g(i_k,i_{k+1}) + J_{\mu_{k+1}} - J_{\mu_k}, \qquad k = 1,\ldots,N.$$

The temporal difference represents the difference between the current estimate $J_\mu(i_k)$ of expected *cost-to-go* to the termination state and the predicted cost-to-go to the termination state $g(i_k, i_k{+}1){+}\ J_{\mu_{k+1}}$. The key idea of the Monte-Carlo simulation using temporal differences is to update the individual cost-to-go as soon as

the cost to go of the successor has been estimated. In other words $J_\mu(i_1)$ is update as soon as $g(i_1, i_2)$ and i_2 are generated during the simulation runs. Then update both $J_\mu(i_1)$ and $J_\mu(i_2)$ immediately after $g(i_2, i_3)$ and i_3 are generated, etc.

For the cases where the number of stages becomes prohibitively large, approximations schemes can be used so as to derive accurate estimates of either the optimal cost, J^*, or the optimal policy, μ. By approximating the optimal cost, J^*, we need to generate for a given state i and approximation $J'(i, r)$ of $J^*(i)$, where r a parameter vector that is to be determined by using some type of least squares minimization. Once the cost is know, it can then be used so as to generate suboptimal policies as:

$$\mu'(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)(g(i, u, j) + J'(j, r)) .$$

The type of the approximation is nonunique but usually the approximations are of the form:

$$J'(i, r) = \sum_{k=1}^{m} r_k w_k(i).$$

In essence the approximation is a linear combination of a set of basis functions.

Recently (1994), [8], presented an approximation scheme referred to as *feature-based aggregation*. The idea is to develop an approximation by making use of the fact that several states may share some common characteristics (features). For a stochastic shortest path problem with n states, one can identify m disjoint subsets S_k, $k = 1, \ldots, m$, such that:

$$S = S_1 \cup \cdots \cup S_m.$$

The basis functions $\omega_k(i)$ can therefore be defined as:

$$\omega_k(i) = \begin{cases} 1 & \text{if } i \in S_k, \\ 0 & \text{if } i \in S_k. \end{cases}$$

The approximate cost can thus be defined as:

$$J'(i, r) = \sum_{k=1}^{m} r_k \omega_k(i).$$

The optimal vector r can be determined as the solution of the aggregate stochastic shortest path problem, for which the aggregate aggregate transition probabilities q_{ki} express the probability of moving from any state in S_k to state i. The vector r solves the corresponding Bellman's equation of the aggregate problem:

$$r_k = \sum_{i=1}^{n} q_{ki} \times \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i, u, j) + \sum_{s=1}^{m} r_s \omega_s(j) \right), \quad k = 1, \ldots, m.$$

For the aggregate problem, the simulation ideas previously developed can be utilized so as to obtain the optimal vector r and therefore obtain the required approximation.

Approximation and simulation schemes can also be combined so as to provide alternatives to performing policy iterations, [1]. For a given stationary policy μ, a number of simulations, M, can be performed so as to obtain the estimates $c(i, m)$. subsequently, a least squares optimization can be solved to provide an approximation to the costs $J_\mu{}'(i, r$, and the coefficients r are derived by solving the following optimization problem:

$$\min_r \sum_{i \in S} \sum_{m=1}^{M} \left| J'(i, r) - c(i, m) \right|^2 .$$

Once the costs function have been determined, and improved policy, $\overline{\mu}(i)$ is identified as:

$$\overline{\mu}(i) = \arg \min_{u \in U(i)} \sum_{j} p_{ij}(u)(g(i, u, j) + J'(i, r)).$$

In essence, the method iterates between a policy evaluation and a policy improvement step, using both simulation techniques for obtaining, for a given state i and policy μ, sample costs and approximation techniques for obtaining representation of these costs. There are subsequently used so as to estimate improved policies and the iterations continue. The concept of *Q-learning*, [9], was recently proposed as an alternative way of implementing the concept of re-enforcement learning in the solution of dynamic programming, [4].

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming
- ▶ Optimization Strategies for Dynamic Systems

References

1. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
2. Bertsekas DP, Tsitsiklis JN (1997) Neuro-dynamic programming. Athena Sci., Belmont, MA
3. Eaton JH, Zadeh LA (1962) Optimal pursuit strategies in discrete state probabilistic systems. Trans ASME Ser DJ Basic Eng 84:23–29
4. Littman ML (1996) Algorithms for sequential decision making. PhD Thesis Brown Univ.
5. Psaraftis HN, Tsitsiklis JN (1993) Dynamic shortest paths in acyclic networks with markovian arc costs. Oper Res 41:91–101
6. Puterman ML (1994) Markov decision processes - Discrete stochastoc dynamic programming. Wiley, New York
7. Sutton RS (1988) Learning to predict by the method of temporal differences. Machine Learning 3:9–44
8. Tsitsiklis JN (1994) Asynchronous stochastic aggregation and Q-learning. Machine Learning 16:185–202
9. Watkins CJ (1989) Learning from delayed rewards. PhD Thesis Cambridge Univ.

Dynamic Programming: Undiscounted Problems

IOANNIS P. ANDROULAKIS
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L20, 90C40

Article Outline

Keywords
See also
References

Keywords

Dynamic programming; Infinite horizon problems; Discounted problem

Total cost *infinite horizon problems* deal with optimal decision making problems in the presence of uncertainty of systems in which events occur sequential. In general, the state transitions are described by a stationary dynamic system of the form:

$$x_{k+1} = f(x_k, u_k, \omega_k), \quad k = 0, 1, \ldots,$$

where for each time instance (stage) k, the state of the system is an element of the space S, the control action u that is to be implemented so as to achieve optimality belong to a space C, and finally the uncertainty is modeled through a set of random disturbances ω that belong to a *countable set* D. Furthermore, it is assumed that the control u_k is constrained to take values in a given nonempty set $U(x_k) \in C$, which depends of the current state x_k. The random disturbances ω_k, $k = 0, 1, \ldots$, have identical statistics and the probability distributions $\mathcal{P}(\cdot|x_k, u_k)$ are defined on D. These may depend explicitly on x_k and u_k but not on prior disturbances. Given an initial state x_0, we seek a policy π such that $\pi = \{\mu_0, \mu_1, \ldots\}$ for which:

$$\mu_k : \ S \to C,$$
$$\mu_k(x_k) \in U(x_k), \quad \forall x_k \in S,$$

that minimizes a cost function defined as:

$$J_\pi(x_0) = \lim_{N\to\infty} \mathsf{E}\left\{\sum_{k=1}^{N-1} \alpha^k g(x_k, \mu_k(x_k), \omega_k)\right\}.$$

The function $g()$ is the cost per stage such that: g: $S \times C \times D \to \mathbf{R}$ and is assumed to be given. Finally, the parameter α is termed *discount factor* and it holds that: $0 < \alpha \leq 1$. We denote by Π the set of all admissible policies $\pi = \{\mu_0, \mu_1, \ldots\}$, that is, the set of all sequences of

such functions for which:

$$\mu_k : S \to C,$$
$$\mu_k(x_k) \in U(x_k), \quad \forall x_k \in S.$$

The optimal cost function J^* is then defined as:

$$J^* = \min_{\pi \in \Pi} J_\pi(x), \quad x \in S.$$

An admissible policy of the form $\pi = \{\mu, \mu, \ldots\}$ is termed *stationary* and its corresponding cost is $J\mu$.

Nevertheless, it is often the case that either the discount factor, α, does not have to be less than one, or even the cost per stage does not have to be bounded from either above or bellow. If that is the case, then it is quite possible that for some initial states x_0, the cost functional $J_\pi(x_0)$ may become infinite.

D. Blackwell [4] was among the first to analyze the case in which the discount factor α becomes 1. His approach was based on the idea of studying the behavior as the discount factor approaches 1. Based on the ideas introduced in [7], undiscounted problems are analyzed under either of the following assumptions:

- Positivity assumption:

$$0 \le g(x, u, \omega), \quad \forall (x, u, \omega) \in S \times C \times D;$$

- Negativity assumption:

$$g(x, u, \omega) \le 0, \quad \forall (x, u, \omega) \in S \times C \times D.$$

Having costs per stages being bounded from either above or below may result in the complication of having unbounded costs for some initial states. Therefore, the assumption will be made that $\infty, (-\infty)$, are admissible costs J_π, under the positivity (negativity) assumption. Defining the following two mappings, greatly simplifies the analysis. For any function J defined in S that takes the values $[0, +\infty]$ under the positivity assumption, or the values $[-\infty, 0]$ under the negativity assumption, the mappings T and T_μ are defined as:

$$(TJ)(x) = \min_{u \in U(x)} \mathsf{E}\{g(x, u, \omega) + J(f(x, u, \omega)\}.$$

Furthermore, for any admissible *stationary policy*, the mapping T_μ is defined as:

$$(T_\mu J)(x) = \mathsf{E}\{g(x, \mu(x), \omega) + J(f(x, \mu(x), \omega)\}.$$

Under both the positivity or negativity assumptions, it can be shown, [1], that *Bellman's equation* is satisfied:

- Under either the positivity or the negativity assumption, the optimal cost function, J^* satisfies:

$$J^*(x) = \min_{u \in U(x)} \mathsf{E}\{g(x, u, \omega) + J^*(f(x, u, \omega)\}$$

Clearly, the optimality conditions requires that:

$$J^* = TJ^*.$$

Equivalently, for any stationary policy, it holds true that:

$$J_\mu = T_\mu(J_\mu).$$

It is to be noted though that for undiscounted problems, $\alpha = 1$, the function J^* need not be the unique function minimizes Bellman's equation. In other words, the mapping T does not have a unique fixed point. Nevertheless, the optimal cost vector, J^*, is the smallest fixed point, under the positivity assumption, or the largest fixed point, under the negativity assumption.

- Under the positivity assumption, if $J' : S \to [0, +\infty]$ satisfies $J' = TJ'$, then:

$$J^* \le J'.$$

- Under the negativity assumption, if $J' : S \to [-\infty, 0]$ satisfies $J' = TJ'$, then:

$$J' \le J^*.$$

It should be pointed out, that in the analysis of undiscounted problems the concept of monotonicity plays a key role. The following *monotone convergence theorem* summarizes the key properties, [3]:

Theorem *Let $P = (p_1, p_2, \ldots)$ be a probability distribution over $S = \{1, 2, \ldots\}$. Let $\{h_N\}$ be a sequence of extended real-valued functions on S such that for all $i \in S$ and $N = 1, 2, \ldots$:*

$$0 \le h_N(i) \le h_{N+1}(i).$$

Let $h : S \to [0, \infty]$ be the limit function:

$$h(i) = \lim_{N \to \infty} h_N(i).$$

Then:

$$\lim_{N \to \infty} \sum_{i=1}^{\infty} p_i h_N(i) = \sum_{i=1}^{\infty} p_i \lim_{N \to \infty} h_N(i) = \sum_{i=1}^{\infty} p_i h(i).$$

As examples of undiscounted problems, let us consider two types of problems that define interesting classes that can be cast as undiscounted dynamic programming problems, namely the *optimal stopping* and the *optimal gambling* strategy problem. The former defines a situation in which at each state x of the state space there are two possible actions that are available. One may either decide to stop, by selecting control u_1, and pay a terminal cost $t(x)$, or, select control u_2, pay a cost $c(x)$ and continue the process, with a new state be given by:

$$x_{k+1} = f(x_k, \omega_k).$$

For completeness purposes one also defines a termination state s that is entered once the stopping decision is made. In other words,

$$x_{k+1} = s \quad \text{if } u_k = u_1 \text{ or } x_k = s.$$

If it is further assumed that both the termination and the continuation costs are positive (or negative), then the problem satisfies the positivity (negativity) assumption. The mapping T defined earlier takes now the form:

$$(TJ)(x) = \min\{t(x), c(x) + \mathsf{E}\{J(f(x,\omega))\}\}, \quad \forall x \in S.$$

The objective is to find the optimal stopping policy that minimizes the total expected costs over an infinite number of stages. Insofar regarding external disturbances, ω, it is assumed that they have the same probability distribution for all time instances and depend only on the current state x_k.

The early work [5] details the gambling problem, but was also one of the early works on undiscounted problems. The problem is defined as one in which a player may stake at any time k any amount $u_k \geq 0$ that does not exceed his/her current fortune, x_k. The stake is won back with probability p, and lost with probability $1-p$. The discrete-time state evolution is described by:

$$x_{k+1} = x_k + \omega_k u_k, \quad k = 1, 2, \ldots.$$

The disturbance ω_k is considered to be 1 with probability p, and -1 with probability $1-p$. The gambling is continued until reaching given fortune or loosing the entire initial capital. The problem is to determine that optimal gambling strategy that maximizes the probability of reaching the target fortune. As gambling strategy is defined the specific rule that specifies what the stakes should be at time k. It can be shown that the *bold strategy* is an optimal policy. The bold strategy is defined as:

$$\mu^*(x) = \begin{cases} x, & 0 < x \leq \frac{1}{2}, \\ 1-x, & \frac{1}{2} \leq x < 1. \end{cases}$$

As suggested by the theory of undiscounted problems, the bold strategy is simply an optimal strategy and others can also be derived, [5].

From a computational standpoint it is important to know whether the method of *successive approximations*, i. e., *value iteration*, converges to the optimal cost function. In other words, it is important to know whether the basic dynamic programming algorithm converges. Under either of the two basic assumptions we have:

- Positivity assumption:

$$J_0 \leq T(J_0) \leq \cdots \leq T^k(J_0) \leq \cdots.$$

- Negativity assumption:

$$J_0 \geq T(J_0) \geq \cdots \geq T^k(J_0) \geq \cdots.$$

In either case

$$J_\infty(x) = \lim_{k\to\infty} T^k(J_0)(x), \quad x \in S.$$

In other words, the sequence generated by successive approximation, i. e., by successively applying the mapping T, converges and the limit is well defined, including the values of $+\infty$ and $-\infty$. For the value iteration method though to be valid we also need to have that $J_\infty = J^*$. In order for the above to be true under the positivity assumptions, an additional condition needs to be satisfied, [2]:

- Let the positivity assumption be satisfied and assume that the sets:

$$U_k(x,\lambda) = \Big\{u \in U(x): \mathsf{E}\{g(x,u,\omega) + T^k(J_0)(f(x,u,\omega))\} \leq \lambda\Big\}$$

are *compact* subsets of a Euclidean space for every $x \in S$, $\lambda \in \mathbf{R}$, and for all k greater than some integer $\bar{k}$. Then:

$$J_\infty = T(J_\infty) = J^*.$$

It should be noted that since $U(x)$ is assumed to be finite, the above condition is satisfied. A detailed account of the value iteration method of undiscounted Markov decision problems can also be found in [6].

It can also be shown, [3], that it is possible to devise computational methods based on mathematical programming when the state, control, and disturbance spaces are finite. Under the negativity assumption, the vector J solves the following linear programming problem:

$$\begin{cases} \max & \sum_{i=1}^{n} \lambda_i \\ \text{s.t.} & \lambda_i \le g(i,u) + \sum_{j=1}^{n} p_{ij}(u)\lambda_j \\ & i = 1, 2, \ldots . \end{cases}$$

Under the positivity assumption, the corresponding program takes the form:

$$\begin{cases} \min & \sum_{i=1}^{n} \lambda_i \\ \text{s.t.} & \lambda_i \ge \min_{u \in U(x)} \left\{ g(i,u) + \sum_{j=1}^{n} p_{ij}(u)\lambda_j \right\}, \\ & i = 1, 2, \ldots . \end{cases}$$

Unfortunately, this two-level optimization problem is neither linear nor convex, and therefore its solution highly nontrivial.

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming

References

1. Bertsekas DP (1976) Dynamic programming and stochastic control. Acad. Press, New York
2. Bertsekas DP (1977) Monotone mappings with applications in dynamic programing. SIAM J Control Optim 15 , 438–464
3. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
4. Blackwell D (1962) Discrete dynamic programming. Ann Math Statist 33:719–726
5. Dubins L, Savage LM (1966) How to gabmble if you must. McGraw-Hill, New York
6. Schweitzer PJ, Federgruen A (1977) The asymptotic behavior of undiscounted value iteration in Markov decision problems. Math Oper Res 2:360–381
7. Strauch R (1966) Negative dynamic programming. Ann Math Statist 37:871–890

Dynamic Traffic Networks

Anna Nagurney
University Massachusetts, Amherst, USA

MSC2000: 90B15

Article Outline

Keywords
A Dynamic Traffic Network Model
The Trip-Route Choice Adjustment Process
Stability Analysis
Discrete Time Algorithms
See also
References

Keywords

Projected dynamical system; Trip-route choice adjustment process; Day-to-day dynamic travel behavior; System stability; Asymptotical system stability; Discrete-time algorithms; Regular link cost function

Congested urban transportation networks represent complex systems in which travelers interact so as to determine unilaterally their cost-minimizing routes of

travel between their points of origin and their destinations. The governing concept here is that of 'user-optimization', which dates to J.G. Wardrop [19] (and was so termed by S.C. Dafermos and F.T. Sparrow [4]), and states that, in equilibrium, all used paths connecting an origin/destination pair of nodes will have travel costs that are equal and minimal.

The complexity of user-optimized transportation networks, sometimes also referred to as the 'traffic assignment' problem, has stimulated much research in the past several decades, both from methodological perspectives, as well as in terms of practical application. Notable developments include: the proof by M.J. Beckmann, C.B. McGuire, and C.B. Winsten [1] that, under certain symmetry assumptions on the link travel cost functions (and travel disutility functions), the traffic network equilibrium solution also satisfies the Kuhn–Tucker conditions of an appropriately constructed optimization problem, and the identification by Dafermos [2] that the traffic network equilibrium conditions, as formulated by M.J. Smith [17] (without any imposition of a symmetry assumption), satisfy a variational inequality problem. Books that discuss methodological approaches to static traffic equilibrium problems include [9,14,16] (see also [6]).

The study of dynamic travel route choice models on general transportation networks, where time is explicitly incorporated into the framework, was initiated by D.K. Merchant and G.L. Nemhauser [8], who focused on dynamic system-optimal networks with the characteristic of many origins and a single destination. In system-optimal networks, in contrast to user-optimal networks, one seeks to determine the path flow and link load patterns that minimize the total cost in the network, rather than the individual path travel costs.

M.J. Smith [18], in turn, proposed a dynamic traffic user-optimized model with fixed demands. H. Mahmassani [7] also proposed dynamic traffic models and investigated them experimentally. The recent book [15] provides an overview of the history of dynamic traffic network models and discusses distinct approaches for their analysis and computation.

Here we present a dynamic traffic model with elastic demands proposed by P. Dupuis and A. Nagurney [5], who, along with D. Zhang and Nagurney [22], established the foundations for a new methodology, that of 'projected dynamical systems' theory. The notable feature of a projected dynamical system is that its set of stationary points coincides with the set of solutions of the corresponding variational inequality problem. There, thus, exists a fundamental linkage between the static world of finite-dimensional variational inequality problems and the dynamic world exhibited by a new class of dynamical system.

The dynamic adjustment process that is presented here models the travelers' *day-to-day* dynamic behavior of making trip decisions and route choices associated with a travel disutility perspective. Subsequently, some of the stability results of this travel-route choice adjustment process obtained by Zhang and Nagurney [21] are reviewed, which address whether and how the travelers' dynamic behavior in attempting to avoid congestion leads to a traffic equilibrium pattern. Finally, we recall the discrete-time algorithms devised for the computation of traffic network equilibria with elastic demands and with known travel disutility functions. The convergence of these discrete-time algorithms was established by Zhang and Nagurney [10,13]. Additional dynamic traffic network models, as well as qualitative and numerical results, using this methodology can be found in [11,12], and [22]. For alternative dynamic traffic network models and approaches, see [15], and the references therein.

A Dynamic Traffic Network Model

The model that we present is due to Dupuis and Nagurney [5]. It is a dynamic counterpart to the static traffic network equilibrium model with elastic travel demands developed by Dafermos [3].

We consider a network $[N, L]$ consisting of nodes $[N]$ and directed links $[L]$. Let a denote a link of the network connecting a pair of nodes, and let p denote a path (assumed to be acyclic) consisting of a sequence of links connecting an origin/destination (O/D) pair w. P_w denotes the set of paths connecting the O/D pair w with n_{P_w} paths. We let W denote the set of O/D pairs and P the set of paths in the network.

Let x_p represent the flow on path p and let f_a denote the load on link a. The following conservation of flow equation must hold for each link a:

$$f_a = \sum_p x_p \delta_{ap},$$

where $\delta_{ap} = 1$, if link a is contained in path p, and 0 otherwise. The expression states that the load on a link a is equal to the sum of all the path flows on paths that contain the link a.

Moreover, if we let d_w denote the demand associated with an O/D pair w, then we must have that for each O/D pair w

$$d_w = \sum_{p \in P_w} x_p,$$

where $x_p \geq 0$, for all p, that is, the sum of all the path flows on paths connecting the O/D pair w must be equal to the demand d_w. Let x denote the column vector of path flows with dimension n_P.

Let c_a denote the user cost associated with traversing link a, and let C_p denote the user cost associated with traversing path p. Then

$$C_p = \sum_a c_a \delta_{ap}.$$

In other words, the cost of a path is equal to the sum of the costs on the links comprising that path. We group the link costs into the column vector c with n_A components, and the path costs into the column vector C with n_P components. We also assume that we are given a travel disutility function λ_w for each O/D pair w. We group the travel disutilities into the column vector λ with J components.

We assume that, in general, the cost associated with a link may depend upon the entire link load pattern, that is,

$$c_a = c_a(f)$$

and that the travel disutility associated with an O/D pair may depend upon the entire demand pattern, that is,

$$\lambda_w = \lambda_w(d),$$

where f is the n_A-dimensional column vector of link loads and d is the J-dimensional column vector of travel demands.

We now, for completeness, recall the traffic network equilibrium conditions.

Definition 1 (*traffic network equilibrium*, [1,3]) A vector $x^* \in \mathbf{R}_+^{n_p}$, which induces a vector d^* through the demand equations, is a traffic network equilibrium if for each path $p \in P_w$ and every O/D pair w:

$$C_p(x^*) \begin{cases} = \lambda_w(d^*), & \text{if} \quad x_p^* > 0 \\ \geq \lambda_w(d^*), & \text{if} \quad x_p^* = 0. \end{cases}$$

In equilibrium, only those paths connecting an O/D pair that have minimal user costs are used, and their costs are equal to the travel disutility associated with traveling between the O/D pair.

The equilibrium conditions have been formulated as a variational inequality problem by Dafermos (cf. [2,3]). In particular, we have:

Theorem 2 [3] *$(x^*, d^*) \in K^1$ is a traffic network equilibrium pattern, that is, satisfies the equilibrium conditions if and only if it satisfies the variational inequality problem (path flow formulation):*

$$\langle C(x^*)^\top, x - x^* \rangle - \langle \lambda(d^*)^\top, d - d^* \rangle \geq 0, \qquad \forall (x, d) \in K^1,$$

where $K^1 \equiv \{(x, d) : x \geq 0$, and the demand constraints hold$\}$.

Note that, in view of the demand constraints, one may define $\widehat{\lambda}(x) \equiv \lambda(d)$, in which case one may rewrite the variational inequality in the path flow variables x only, that is, we seek to determine $x^* \in \mathbf{R}_+^{n_p}$, such that

$$\langle (C(x^*) - \overline{\lambda}(x^*))^\top, x - x^* \rangle \geq 0, \qquad \forall x \in \mathbb{R}_+^{n_P},$$

where $\overline{\lambda}(x)$ is the $n_{P_{w_1}} \times \ldots n_{P_{w_J}}$-dimensional column vector with components:

$$(\widehat{\lambda}_{w_1}(x), \ldots, \widehat{\lambda}_{w_1}(x), \ldots, \widehat{\lambda}_{w_J}(x), \ldots, \widehat{\lambda}_{w_J}(x)),$$

where J is the number of O/D pairs. If we now let $F(x) \equiv (C(x) - \overline{\lambda}(x))$ and $K \equiv \{x : x \in \mathbf{R}_+^{n_p}\}$, then, clearly, this inequality can be placed into standard variational inequality form.

The Trip-Route Choice Adjustment Process

The dynamical system, first presented in [5], whose stationary points correspond to solutions of the latter variational inequality problem above, is given by:

$$\dot{x} = \Pi_K(x, \overline{\lambda}(x) - C(x)), \qquad x(0) = x_0 \in K,$$

where, assuming that the feasible set K is a convex polyhedron (as is the case here), and given $x \in K$ and $v \in \mathbf{R}^n$, we define the projection of the vector v at x (with respect to K) by

$$\Pi_K(x, v) = \lim_{\delta \to 0} \frac{P_K(x + \delta v) - x}{\delta},$$

where P_K is defined as:

$$P_K(x) = \arg\min_{z \in K} \|x - z\|,$$

and $\|\cdot\|$ denotes the Euclidean norm.

This dynamical system is a *projected dynamical system* (cf. [10,22]), since the right-hand side, which is a projection operator, is discontinuous.

The adjustment process interpretation of the dynamical system, as discussed in [5], is as follows: Users of a transportation network choose, at the greatest rate, those paths whose differences between the travel disutilities (demand prices) and path costs are maximal; in other words, those paths whose costs are minimal relative to the travel disutilities. If the travel cost on a path exceeds the travel disutility associated with the O/D pair, then the flow on that path will decrease; if the travel disutility exceeds the cost on a path, then the flow on that path will increase. If the difference between the travel disutility and the path cost drives the path flow to be negative, then the projection operator guarantees that the path flow will be zero. The process continues until there is no change in path flows, that is, until all used paths have path costs equal to the travel disutilities, whereas unused paths will have costs which exceed the disutilities. Specifically, the travelers adjust their route choices until an equilibrium is reached.

The following example, given in a certain discrete-time realization, shows how the dynamic mechanism of the above trip-route choice adjustment would reallocate the traffic flow among the paths and would react to changes in the travel disutilities.

Example 3 Consider a simple transportation network consisting of two nodes, with a single O/D pair w, and two links a and b representing the two disjoint paths connecting the O/D pair. Suppose that the link costs are:

$$c_a(f_a) = f_a + 2, \qquad c_b(f_b) = 2f_b,$$

and the travel disutility function is given by:

$$\lambda_w(d_w) = -d_w + 5.$$

Note that here a path consists of a single link and, hence, we can use x and f interchangeably. Suppose that, at time $t = 0$, the flow on link a is 0.7, the flow on link b is 1.5; hence, the demand is 2.2, and the travel disutility is 2.8, that is,

$$x_a(0) = 0.7, \qquad x_b(0) = 1.5,$$
$$d_w(0) = 2.2, \qquad \lambda_w(0) = 2.8,$$

which yields travel costs: $c_a(0) = 2.7$ and $c_b(0) = 3.0$.

According to the above trip-route choice adjustment process, the flow changing rates at time $t = 0$ are:

$$\dot{x}_a(0) = \lambda_w(0) - c_a(0) = 0.1,$$
$$\dot{x}_b(0) = \lambda_w(0) - c_b(0) = -0.2.$$

If a time increment of 0.5 is used, then at the next moment $t = 0.5$, the flows on link a and link b are:

$$x_a(0.5) = x_a(0) + 0.5\dot{x}_a(0)$$
$$= 0.7 + 0.5 \times 0.1 = 0.75,$$
$$x_b(0.5) = x_b(0) + 0.5\dot{x}_b(0)$$
$$= 1.5 - 0.5 \times 0.2 = 1.4,$$

which yields travel costs: $c_a(0.5) = 2.75$ and $c_b(0.5) = 2.8$, a travel demand $d_w(0.5) = 2.15$, and a travel disutility $\lambda_w(0.5) = 2.85$. Now, the flow changing rates are given by:

$$\dot{x}_a(0.5) = \lambda_w(0.5) - c_a(0.5)$$
$$= 2.85 - 2.75 = 0.1,$$
$$\dot{x}_b(0.5) = \lambda_w(0.5) - c_b(0.5)$$
$$= 2.85 - 2.8 = 0.05.$$

The flows on link a and link b at time $t = 1.0$ would, hence, then be:

$$x_a(1.0) = x_a(0.5) + 0.5\dot{x}_a(0.5)$$
$$= 0.75 + 0.5 \times 0.1 = 0.80,$$
$$x_b(1.0) = x_b(0.5) + 0.5\dot{x}_b(0.5)$$
$$= 1.4 + 0.5 \times 0.05 = 1.425,$$

which yields travel costs: $c_a(1.0) = 2.80$ and $c_b(1.0) = 2.85$, a travel demand $d_w(1.0) = 2.225$, and a travel disutility $\lambda_w(1.0) = 2.775$. Now, the flow changing rates are

given by:

$$\dot{x}_a(1.0) = \lambda_w(1.0) - c_a(1.0)$$
$$= 2.775 - 2.800 = 0.025,$$
$$\dot{x}_b(1.0) = \lambda_w(1.0) - c_b(1.0)$$
$$= 2.775 - 2.850 = -0.075.$$

The flows on link a and link b at time $t = 1.5$ would be:

$$x_a(1.5) = x_a(1.0) + 0.5\dot{x}_a(1.0)$$
$$= 0.8 - 0.5 \times 0.025 = 0.7875,$$
$$x_b(1.5) = x_b(1.0) + 0.5\dot{x}_b(1.0)$$
$$= 1.425 - 0.5 \times 0.075 = 1.3875,$$

which yields travel costs: $c_a(1.5) = 2.7875$ and $c_b(1.5) = 2.775$, a travel demand $d_w(1.5) = 2.175$, and a travel disutility $\lambda_w(1.0) = 2.82$.

In this example, hence, as time elapses, the trip-route choice adjustment process adjusts the flow volume on the two links so that the difference between the travel costs of link a and link b is being reduced, from 0.3, to 0.05, and, finally, to 0.0125; and, the difference between the disutility and the travel costs on the used links is also being reduced from 0.2, to 0.1, and to 0.045. In fact, the traffic equilibrium with: $x_a^* = 0.8$ and $x_b^* = 1.4$, which induces the demand $d_w^* = 2.2$, is almost attained in only 1.5 time units.

Stability Analysis

We now present the stability results of the trip route choice adjustment process. The results described herein are due to Zhang and Nagurney [21]. For example, the questions that motivate transportation planners and analysts to study the stability of a transportation system include: Will any initial flow pattern be driven to an equilibrium by the adjustment process? In addition, will a flow pattern near an equilibrium always stay close to it? These concerns of *system stability* are important in traffic assignment and form, indeed, a critical base for the very concept of an equilibrium flow pattern.

For the specific application of transportation network problems, the following definitions of stability of the transportation system and the local stability of an equilibrium are adapted from the general stability concepts of projected dynamical systems (cf. [22]).

Definition 4 (*stability at an equilibrium*) An equilibrium flow pattern x^* is stable if it is a global monotone attractor for the corresponding route choice adjustment process.

Definition 5 (*asymptotical stability at an equilibrium*) An equilibrium flow pattern x^* is asymptotically stable if it is a strictly global monotone attractor for the corresponding route choice adjustment process.

Definition 6 (*stability of the system*) A route choice adjustment process is stable if all its equilibrium flow patterns are stable.

Definition 7 (*asymptotical stability of the system*) A route choice adjustment process is asymptotically stable if all its equilibrium flow patterns are asymptotically stable.

We now present the stability results in [21] for the trip-route choice adjustment process.

Theorem 8 ([21]) *Suppose that the link cost functions c are monotone increasing in the link load pattern f and that the travel disutility functions λ are monotone decreasing in the travel demand d. Then the trip-route choice adjustment process is stable.*

Theorem 9 ([21]) *Assume that there exists some equilibrium path flow pattern. Suppose that the link cost functions c and negative disutility functions $-\lambda$ are strictly monotone in the link load f and the travel demand d, respectively. Then the trip-route choice adjustment process is asymptotically stable.*

The first theorem states that, provided that monotonicity of the link cost functions and the travel disutility functions holds true, then any flow pattern near an equilibrium will stay close to it forever. Under the strict monotonicity assumption, on the other hand, the second theorem can be interpreted as saying that any initial flow pattern will eventually be driven to an equilibrium by the route choice adjustment process.

Discrete Time Algorithms

The Euler method and the Heun method were employed in [13] and [10] for the computation of solutions to dynamic elastic demand traffic network problems with known travel disutility functions, and their convergence was also established therein. We refer the

reader to these references for numerical results, including traffic network examples that are solved on a massively parallel computer architecture.

In particular, at iteration τ, the *Euler method* computes

$$x^{\tau+1} = P_K(x^\tau - a_\tau F(x^\tau)),$$

whereas, according to the *Heun method*, at iteration τ one computes

$$x^{\tau+1} = P_K\left(x^\tau - a_\tau \frac{1}{2}\left[F(x^\tau) + F(P(x^\tau - a_\tau F(x^\tau)))\right]\right).$$

In the case that the sequence $\{a_\tau\}$ in the Euler method is fixed, say, $\{a_\tau\} = \rho$, for all iterations τ, then the Euler method collapses to a projection method (cf. [2,6,9], and [14]).

In the context of the dynamic traffic network problem with known travel disutility functions, the projection operation in the above discrete-time algorithms can be evaluated explicitly and in closed form. Indeed, each iteration τ of Euler method takes the form: For each path $p \in P$ in the transportation network, compute the path flow $x_p^{\tau+1}$ according to:

$$x_p^{\tau+1} = \max\{0, x_p^\tau + a_\tau(\lambda_w(d^\tau) - C_p(x^\tau))\}.$$

Each iteration of the Heun method, in turn, consists of two steps. First, at iteration τ one computes the approximate path flows:

$$\overline{x}_p^\tau = \max\{0, x_p^\tau + a_\tau(\lambda_w(d^\tau) - C_p(x^\tau))\}, \qquad \forall p \in P,$$

and updates the approximate travel demands:

$$\overline{d}_w^\tau = \sum_{p \in P_w} \overline{x}_p^\tau, \qquad \forall w \in W.$$

Let

$$\overline{x}^\tau = \{\overline{x}_p^\tau, p \in P\}$$

and

$$\overline{d}^\tau = \{\overline{d}_w^\tau, w \in W\}.$$

Then, for each path $p \in P$ in the transportation network one computes the updated path flows $x_p^{\tau+1}$ according to:

$$x_p^{\tau+1} = \max\left\{0, x_p^\tau + \frac{a_\tau}{2}[\lambda_w(d^\tau) - C_p(x^\tau) + \lambda_w(\overline{d}^\tau) - C_p(\overline{x}^\tau)]\right\}, \quad \forall p \in P,$$

and updates the travel demands $d_w^{\tau+1}$ according to:

$$d_w^{\tau+1} = \sum_{p \in P_w} x_p^{\tau+1}, \qquad \forall w \in W.$$

It is worth noting that both the Euler method and the Heun method at each iteration yield subproblems in the path flow variables, each of which can be solved not only in closed form, but also, simultaneously. Hence, these algorithms in the context of this model can be interpreted as massively parallel algorithms and can be implemented on massively parallel architectures. Indeed, this has been done so by Nagurney and Zhang [13] (see also [11] for the case where the demand functions are given, rather than the travel disutility functions).

In order to establish the convergence of the Euler method and the Heun method, one regularizes the link cost structures.

Definition 10 (*regular cost function*) The link cost function c is called regular if, for every link $a \in L$,

$$c_a(f) \to \infty, \qquad \text{as} \quad f_a \to \infty,$$

holds uniformly true for all link flow patterns.

We note that the above regularity condition on the link cost functions is natural from a practical point of view and it does not impose any substantial restrictions. In reality, any link has an upper bound in the form of a capacity. Therefore, letting $f_a \to \infty$ is an artificial device under which one can reasonably deduce that $c_a(f) \to \infty$, due to the congestion effect. Consequently, any practical link cost structure can be theoretically extended to a regular link cost structure to allow for an infinite load.

The theorem below shows that both the Euler method and the Heun method converge to the traffic network equilibrium under reasonable assumptions.

Theorem 11 ([10,13]) *Suppose that the link cost function c is regular and strictly monotone increasing, and that the travel disutility function λ is strictly monotone decreasing. Let $\{a_\tau\}$ be a sequence of positive real numbers that satisfies*

$$\lim_{\tau\to\infty} a_\tau = 0$$

and

$$\sum_{\tau=0}^{\infty} a_\tau = \infty.$$

Then both the Euler method and the Heun method produce sequences $\{x^\tau\}$ that converge to some traffic network equilibrium path flow pattern.

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Cost Approximation Algorithms
- ▶ Directed Tree Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Beckmann MJ, McGuire CB, Winsten CB (1956) Studies in the economics of transportation. Yale Univ. Press, New Haven, CT
2. Dafermos S (1980) Traffic equilibrium and variational inequalities. Transport Sci 14:42–54
3. Dafermos S (1982) The general multimodal traffic equilibrium problem with elastic demand. Networks 12:57–72
4. Dafermos SC, Sparrow FT (1969) The traffic assignment problem for a general network. J Res Nat Bureau Standards 73B:91–118
5. Dupuis P, Nagurney A (1993) Dynamical systems and variational inequalities. Ann Oper Res 44:9–42
6. Florian M, Hearn D (1995) Network equilibrium models and algorithms. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing. Handbook Oper Res and Management Sci. Elsevier, Amsterdam, pp 485–550
7. Mahmassani H (1991) Dynamic models of commuter behavior: Experimental investigation and application to the analysis of planned traffic disruptions. Transport Res 24A:465–484
8. Merchant DK, Nemhauser GL (1978) A model and an algorithm for the dynamic traffic assignment problems. Transport Sci 12:183–199
9. Nagurney A (1999) Network economics: A variational inequality approach, 2nd edn. Kluwer, Dordrecht
10. Nagurney A, Zhang D (1996) Projected dynamical systems and variational inequalities with applications. Kluwer, Dordrecht
11. Nagurney A, Zhang D (1997) Massively parallel computation of dynamic traffic networks modeled as projected dynamical systems. In: Pardalos PM, Hearn DW, Hager WW (eds) Network Optimization. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 374–396
12. Nagurney A, Zhang D (1997) Projected dynamical systems in the formulation, stability analysis, and computation of fixed demand traffic network equilibria. Transport Sci 31:147–158
13. Nagurney A, Zhang D (1998) Massively parallel implementation of a discrete time algorithm for the computation of dynamic elastic demand traffic problems modeled as projected dynamical systems. J Econom Dynam Control 22:1467–1485
14. Patriksson M (1994) The traffic assignment problem. VSP, Utrecht
15. Ran B, Boyce DE (1996) Modeling dynamic transportation network, 2nd revised edn. Springer, Berlin
16. Sheffi Y (1985) Urban transportation networks. Prentice-Hall, Englewood Cliffs, NJ
17. Smith MJ (1979) Existence, uniqueness, and stability of traffic equilibria. Transport Res 13B:295–304
18. Smith MJ (1984) The stability of a dynamic model of traffic assignment-An application of a method of Lyapunov. Transport Sci 18:245–252
19. Wardrop JG (1982) Some theoretical aspects of road traffic research. Proc Inst Civil Engineers II:325–278
20. Zhang D, Nagurney A (1995) On the stability of projected dynamical systems. J Optim Th Appl 85:97–124
21. Zhang D, Nagurney A (1996) On the local and global stability of a travel route choice adjustment process. Transport Res 30B:245–262
22. Zhang D, Nagurney A (1997) Formulation, stability, and computation of traffic network equilibria as projected dynamical systems. J Optim Th Appl 93:417–444

E

Economic Lot-Sizing Problem

SANDRA DUNI EKSIOGLU, BURAK EKSIOGLU
Department of Industrial and Systems Engineering, Mississippi State University, Starkville, USA

MSC2000: 90C11, 90C39, 90C90, 90B10, 90B05, 55M05

Article Outline

Keywords and Phrases
Introduction
Formulation
  Dynamic Programming Algorithm
  Shortest Path Algorithm
  Primal-Dual Algorithm
  Cutting Plane Algorithm
Cases
  Capacitated Economic Lot-Sizing Problem
  Multi-commodity Economic Lot-Sizing Problem
References

Keywords and Phrases

Economic lot-sizing problem; Dynamic programming; Primal-dual algorithm; Valid inequality

Introduction

A considerable amount of effort has been spent on studying and developing efficient solution procedures for the economic lot-sizing problem. This problem had been solved in 1958, but there is still continuing interest in the problem. The main reason for the continuing interest in this problem is its practical applications. For example, economic lot-sizing is the core problem in aggregate production planning in MRP systems (Nahmias [25]). For an extensive review, see Aggarwal and Park [1], Bahl et al. [2], Belvaux and Wolsey [6,7,8], Nemhauser and Wolsey [26], and Wolsey [36].

The economic lot-sizing problem can be defined as follows. Given the demand, the unit production cost, the unit inventory holding cost for a commodity, the production capacities, and the setup costs for each time period over a finite, discrete-time horizon, find a production schedule that would satisfy demand at minimum cost.

This model assumes a fixed and a variable component of production costs. The fixed cost consists of manpower and materials to start up the machines. To reduce the fixed cost per unit, large lot sizes are desired. On the other hand, for every unit produced there are associated production and inventory holding costs, and the total variable cost (production plus inventory) increases with the number of units produced. Solving the lot-sizing problem means finding a production schedule that would satisfy demand at every period and minimize the total of fixed and variable costs.

The work by Harris [18] in 1913 has been cited as the first study of the economic lot-sizing problem that assumes deterministic demands. This model, known as the Economic Order Quantity (EOQ) model, proposes a production schedule to satisfy the demand for a single commodity with a constant demand rate. Production takes place continuously over time, and the model does not incorporate capacity limits.

A major limitation of the above model is that the demand is continuous over time and has a constant rate. Manne [23] and Wagner and Whitin [35] studied the lot-sizing problem with a finite time horizon consisting of a number of discrete periods, each with its own deterministic and independent demand.

Formulation

Let T be the length of the planning horizon, and c_t, h_t, s_t, b_t denote the unit production cost, the unit inventory holding cost, the setup cost, and the demand in period $t(1 \le t \le T)$, respectively. The following are the decision variables for this problem:

q_t: amount of production in period t

I_t: inventory level at the end of period t

$$y_t = \begin{cases} 1 & \text{if production occurs in period } t \\ 0 & \text{otherwise} \end{cases}$$

A mixed-integer programming formulation of the classical economic lot-sizing problem is

$$\text{minimize} \sum_{t=1}^{T} (c_t q_t + s_t y_t + h_t I_t)$$

subject to (C-ELS)

$$q_t + I_{t-1} = b_t + I_t, \quad 1 \le t \le T, \tag{1}$$

$$q_t \le b_{tT} y_t, \quad 1 \le t \le T, \tag{2}$$

$$I_T = 0, \tag{3}$$

$$y_t \in \{0, 1\}, \quad 1 \le t \le T, \tag{4}$$

$$q_t, I_t \ge 0, \quad 1 \le t \le T, \tag{5}$$

where b_{tT} is the total demand in periods $t, \ldots, T$.

Below, some of the approaches that have been used to solve the economic lot-sizing problem are summarized.

Dynamic Programming Algorithm

In 1958, Wagner and Whitin [35] developed a dynamic programming algorithm to solve the economic lot-sizing problem. This algorithm runs in $O(T^2)$.

The special structure of the optimal solutions to the economic lot-sizing problem contributed to developing an efficient dynamic programming algorithm. For this purpose, it is useful to view the problem as a fixed charge network flow problem. The optimal solution for this network flow problem will be a tree since the objective function is concave and the arcs do not have capacity restrictions. This property of an optimal solution implies that:

Theorem 1 *An optimal solution to the economic lot-sizing problem satisfies the following:*
(i.) $q_t I_{t-1} = 0$ *for* $1 \le t \le T$,
(ii.) If $q_t > 0$, *then* $q_t = \sum_{\tau=t}^{\tau'} b_\tau$ *for* $t \le \tau' \le T$.

Property (*i.*) shows that production takes place when the inventory carried forward is zero. This property is known as the *Zero-Inventory Ordering* (ZIO) property. Property (*ii.*) shows that if production takes place in period t, the amount produced will be exactly equal to the total demand for a number of periods (t to τ'). Property (*ii.*) is used to develop the dynamic programming algorithm for the economic lot-sizing problem.

Let $v(t)$ be the minimum cost of a solution for periods $1, \ldots, t$. If $\tau \le t$ is the last period in which production occurs, then $q_\tau = b_{\tau t}$ and $I_{\tau-1} = 0$. This implies that the problem can be divided into two smaller subproblems, and the least-cost solution $v(\tau - 1)$ is optimal for the first subproblem (periods $1, \ldots, \tau - 1$). This leads to the following recursive function:

$$v(t) = \min_{1 \le \tau \le t} \{v(\tau - 1) + s_\tau + c_\tau b_{\tau t} + \sum_{s=\tau}^{t-1} h_s b_{s+1,t}\}$$

with $v(0)$=0.

Calculating $v(t)$ for $t = 1, \ldots, T$ leads to the optimal solution $v(T)$ of the economic lot-sizing problem.

Shortest Path Algorithm

The economic lot-sizing problem has also been solved as a shortest-path algorithm in an acyclic network (Eppen and Martin [13], Martin [24], Zangwill [38]). The acyclic network, G, is built in the following way: let the total number of nodes in G be equal to $T + 1$, one for each time period along with a dummy node. The traversing arc $(t, t') \in G(1 \le t \le t' \le T + 1)$ represents the choice of producing in period t to satisfy the demand in periods $t, \ldots, t' - 1$. Thus, the cost of arc (t, t') is calculated using the following cost function:

$$g_{t,t'} = s_t + c_t b_{t,t'-1} + \sum_{\tau=t}^{t'-2} h_\tau b_{\tau+1,t'-1}.$$

The shortest path from node "1" to node "$T + 1$" provides the set of production intervals of minimum cost and therefore solves the economic lot-sizing problem. The shortest-path algorithm is solved in order $O(m)$,

where m is the number of arcs in the directed, acyclic network. The total number of arcs in G is in the order of T^2; therefore, this shortest-path algorithm solves the economic lot-sizing problem in $O(T^2)$.

Primal-Dual Algorithm

Almost 30 years later (after Wagner and Whitin developed their dynamic programming algorithm), Wagelmans et al. [34], Aggarwal and Park [1], and Federgruen and Tzur [15] showed that the running time of the dynamic programming algorithm could be reduced to $O(T \log T)$ in the general case and to $O(T)$ when the costs have a special structure. This special structure ($h_t + p_t \geq p_{t+1}$) is also referred to as the absence of speculative motives. They developed a primal-dual algorithm to get these results.

Below, we present a reformulation of the economic lot-sizing problem. In this formulation, $q_{t\tau}(\tau = t, \ldots, T)$ is defined as the amount produced in period t to satisfy demand in period τ (Krarup and Bilde [20], van Hoesel [31]). The reformulation is

$$\text{minimize} \sum_{t=1}^{T} \left[\bar{c}_t \sum_{\tau=t}^{T} q_{t\tau} + s_t y_t \right]$$

subject to (Ex-ELS)

$$\sum_{t=1}^{\tau} q_{t\tau} = b_\tau, \quad 1 \leq \tau \leq T, \tag{7}$$

$$q_{t\tau} - b_\tau y_t \leq 0, \quad 1 \leq t \leq \tau \leq T, \tag{8}$$

$$q_{t\tau} \geq 0, \quad 1 \leq t \leq \tau \leq T, \tag{9}$$

$$y_t \in \{0, 1\}, \quad 1 \leq t \leq T, \tag{10}$$

where $\bar{c}_t = p_t + \sum_{s=t+1}^{\tau} h_s$.

Krarup and Bilde show that the linear programming relaxation of (Ex-ELS) (obtained by relaxing constraints $y \in \{0, 1\}$) always has an integer solution. The corresponding dual problem has a special structure that enables developing a primal-dual-based algorithm. The following is the formulation of the dual problem:

$$\text{maximize} \sum_{t=1}^{T} b_t v_t$$

subject to (D-ELS)

$$\sum_{\tau=t}^{T} b_\tau \max(0, v_\tau - \bar{c}_t) \leq s_t \quad 1 \leq t \leq T. \tag{11}$$

The dual variables have the following property: $v_t \geq v_{t+1}$for$1 \leq t \leq T - 1$. This property of the dual variables is used to show that the dual-ascent algorithm gives the optimal solution to the economic lot-sizing problem. Note that formulation (Ex-ELS) of the economic lot-sizing problem is a special case of the facility location problem. The primal-dual algorithm, in principle, is similar to the primal-dual scheme proposed by Erlenkotter [14] for the facility location problem. The primal-dual algorithm for the economic lot-sizing problem developed by Wagelmans et al. [34] runs in $O(T \log T)$.

Cutting Plane Algorithm

Many researchers studying lot-sizing problems also focused on determining a partial polyhedral description of the set of the feasible solutions and applying branch-and-cut methods (Pochet et al. [28], Leung et al. [21], Barany et al. [5]). The main motivation for studying the polyhedral structure of the single item lot-sizing problem is to use the results from these studies to develop efficient algorithms for problems such as the multi-commodity economic lot-sizing problem. However, the branch-and-cut approach has not yet resulted in competitive algorithms for the single-item lot-sizing problem itself. The reason is that generating a single cut could be as time-consuming as solving the whole problem.

Barany et al. [4,5] provide a set of valid inequalities for the single-commodity lot-sizing problem and show that these inequalities are facets of the convex hull of the feasible region. Furthermore, they show that the inequalities fully describe the convex hull of the feasible region. Their separation algorithm runs in order $O(T^2)$.

Pereira and Wolsey [27] study a family of unbounded polyhedra arising in an uncapacitated lot-sizing problem with Wagner-Whitin type costs ($h_t + p_t - p_{t+1} \geq 0$). They characterize the bounded faces of maximal dimension completely and show that they are integral. For a problem with T periods they derive an $O(T^2)$ algorithm to express any point within the polyhedron as a convex combination of the extreme points and the extreme rays of the polyhedron. They observe that for a given objective function, the face of optimal solutions can be found in $O(T^2)$.

Cases

Capacitated Economic Lot-Sizing Problem

The capacitated lot-sizing problem is *NP*-hard even for many special cases (Florian et al. [16] and Bitran and Yanasse [9]). In 1971, Florian and Klein presented a remarkable result. They developed an $O(T^4)$ algorithm for solving the capacitated lot-sizing problem with equal capacities in all periods. This result uses a dynamic programming approach combined with some important properties of optimal solutions to these problems. Recently, van Hoesel and Wagelmans [32] showed that this algorithm can be improved to $O(T^3)$ if backlogging is not allowed and the holding cost functions are linear.

Several solution approaches have been proposed for *NP*-hard special cases of the capacitated lot-sizing problem. These methods are typically based on branch-and-bound (see, for instance, Baker et al. [3] and Erengüç and Aksoy [12]), dynamic programming (Kirca [19] and Chen and Lee [10]) or a combination of the two (Chung and Lin [11] and Lofti and Yoon [22]).

Shaw and Wagelmans [29] considered the capacitated lot-sizing problem with piecewise linear production costs and general holding costs. They showed that this is an *NP*-hard problem and presented an algorithm that runs in pseudopolynomial time.

Multi-commodity Economic Lot-Sizing Problem

The multicommodity version of the problem has attracted much attention. Manne [23] uses the ZIO property to develop a column generation approach to solve this problem. Barany et al. [5] solve the multicommodity capacitated lot-sizing problem without setup times optimally using a cutting-plane procedure followed by branch-and-bound. Moreover, Manne introduces upper bounds on the production capacities.

Other extensions to the classic economic lot-sizing problem consider setup times, backorders, and other factors. Zangwill [37] extends Wagner and Whitin's model by allowing backlogging and introducing general concave cost functions. Veinott [33] studies an uncapacitated model with convex cost structures. Trigeiro et al. [30] show that a capacitated lot-sizing problem with setup times is much harder to solve than a capacitated lot-sizing problem without setup times. It is easy to check if the capacitated lot-sizing problem without setup times has a feasible solution or not. This can be done by computing cumulative demand and cumulative capacity. When setup times are considered, the feasibility problem is *NP*-complete. Also, the bin-packing problem is a special case of capacitated lot-sizing problem with setup times (Garey and Johnson [17, p. 226]).

References

1. Aggarwal A, Park J (1993) Improved algorithms for economic lot size problems. Oper Res 41:549–571
2. Bahl HC, Ritzman LP, Gupta JND (1987) Determining lot sizes and resource requirements: a review. Oper Res 35(3):329–345
3. Baker KR, Dixon P, Magazine MJ, Silver EA (1978) An algorithm for the dynamic lot-size problem with time-varying production capacity constraints. Manage Sci 24:1710–1720
4. Barany I, van Roy J, Wolsey L (1983) Uncapacitated lot-sizing: the convex hull of solutions. Core discussion paper 8314, Université Catholique de Louvain, Louvain-la-Neuve, Belgium
5. Barany I, van Roy J, Wolsey L (1984) Strong formulations for multi-item capacitated lot sizing. Manage Sci 30:1255–1261
6. Belvaux G, Wolsey LA (1998) Lotsizelib: a library of models and matrices for lot-sizing problems. Core discussion paper, Université Catholique de Louvain, Leuven
7. Belvaux G, Wolsey LA (1999) Lot-sizing problems: modeling issues and a specialized branch-and-cut system bc-prod. Core discussion paper dp9849, Université Catholique de Louvain, Leuven
8. Belvaux G, Wolsey LA (2001) Modelling practical lot-sizing problems as mixed-integer programs. Manage Sci 47(7):993–1007
9. Bitran GR, Yanasse HH (1982) Computational complexity of the capacitated lot size problem. Manage Sci 28:1174–1186
10. Chen HD, Lee CY (1991) A simple algorithm for the error bound of the dynamic lot size model allowing speculative motive. Research Report 91-5, Department of Industrial and Systems Engineering, University of Florida, Gainesville
11. Chung CS, Lin CMH (1988) An $O(T^2)$ algorithm for the ni/g/ni/nd capacitated lot size problem. Manage Sci 34:420–426
12. Erengüç ŞS, Aksoy Y (1990) A branch and bound algorithm for a single item nonconvex dynamic lot sizing problem with capacity constraints. Comput Oper Res 17(2):199–210
13. Eppen GD, Martin RK (1987) Solving multi-item capacitated lot-sizing problems using variable redefinition. Oper Res 35(6):832–848
14. Erlenkotter D (1978) A dual-based procedure for uncapacitated facility location. Oper Res 26(6):992–1009

15. Federgruen A, Tzur M (1991) A simple forward algorithm to solve general dynamic lot sizing models with n periods in $O(n \log n)$ or $O(n)$ time. Manage Sci 37:909–925
16. Florian M, Klein M (1971) Deterministic production planning with concave costs and capacity constraints. Manage Sci 18(1):12–20
17. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. WH Freeman and Company, New York
18. Harris FW (1913) How many parts to make at once, in a factory. Manage Sci 10:135–136
19. Kirca O (1990) An efficient algorithm for the capacitated single item dynamic lot size problem. Eur J Oper Res 45:15–24
20. Krarup J, Bilde O (1977) Plant location, set covering, an economic lot-size: an $O(mn)$ algorithm for structured problems. In: Numerische Methoden bei Optimierungsaufgaben, Band 3: Optimierung bei Graphentheoretischen und Ganzzahligen Problemen. Birkhäuser, Basel, pp 155–186
21. Leung JMY, Magnanti TL, Vachani R (1989) Facets and algorithms for capacitated lot sizing. Math Program 45:331–359
22. Lofti V, Yoon Y-S (1994) An algorithm for the single item capacitated lot-sizing problem with concave production and holding costs. J Oper Res Soc 45:934–941
23. Manne AS (1958) Programming of economic lotsizes. Manage Sci 4:115–135
24. Martin RK (1987) Generating alternative mixed-integer programming models using variable redefinition. Oper Res 35:331–359
25. Nahmias S (1997) Production and operations analysis, 3rd edn. Irwin Book Team, Chicago
26. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
27. Pereira O, Wolsey LA (2001) On the wagner-whitin lot-sizing polyhedron. Math Oper Res 26(3):591–600
28. Pochet Y, Wolsey LA (1991) Solving multi-item lot-sizing problems using strong cutting planes. Manage Sci 37:53–67
29. Shaw DX, Wagelmans APM (1998) An algorithm for single-item capacitated economic lot sizing with piecewise linear production costs and general holding costs. Manage Sci 44:831–838
30. Trigeiro W, Thomas LJ, McLain JO (1989) Capacitated lot-sizing with setup times. Manage Sci 35:353–366
31. van Hoesel CPM (1991) Models and algorithms for single-item lot sizing problems, Phd thesis. Erasmus University Rotterdam, Rotterdam
32. van Hoesel CPM, Wagelmans APM (1996) An $O(T^3)$ algorithm for the economic lot-sizing problem with constant capacities. Manage Sci 42:142–150
33. Veinott AF (1964) Production planning with convex costs: a parametric study. Manage Sci 10:441–460
34. Wagelmans A, van Hoesel CPM, Kolen A (1992) Economic lot sizing: an $O(n \log n)$ algorithm that runs in linear time in the wagner-whitin case. Oper Res 40(1):145–156
35. Wagner HM, Whitin TM (1958) Dynamic version of the economic lot size model. Manage Sci 5:89–96
36. Wolsey LA (1995) Progress with single-item lot-sizing. Eur J Oper Res 86:395–401
37. Zangwill WI (1966) A deterministic multiperiod production scheduling model with backlogging. Manage Sci 13:105–119
38. Zangwill WI (1968) Minimum concave cost flows in certain networks. Manage Sci 14:429–450

Eigenvalue Enclosures for Ordinary Differential Equations

Henning Behnke
Institute Math. TU Clausthal, Clausthal, Germany

MSC2000: 49R50, 65L15, 65L60, 65G20, 65G30, 65G40

Article Outline

Keywords
Inclusion Method
Numerical Example
See also
References

Keywords

Upper and lower bounds to eigenvalues; Rayleigh–Ritz method; Lehmann–Maehly method

Selfadjoint eigenvalue problems for ordinary differential equations are very important in the sciences and in engineering. The characterization of eigenvalues by a minimum-maximum principle for the *Rayleigh quotient* forms the basis for the famous *Rayleigh–Ritz method*. This method allows for an efficient computation of nonincreasing upper eigenvalue bounds. N.J. Lehmann and H.J. Maehly [6,7,8] independently developed complementary characterizations that can be used to compute lower bounds. These methods are based on extremal principles for the *Temple quotient*. In general, however, an application of the *Lehmann–Maehly method* requires that certain quantities can be determined explicitly. This may be difficult or even impossible when dealing with partial differential equations. Of great importance is therefore a generalization, the

Goerisch method [3,4,5], that may be used to overcome these problems. Nevertheless, the original Lehmann–Maehly method can easily be applied to a large class of ordinary differential equations; in [10] it is shown, that the method can be interpreted as a special application of the Rayleigh–Ritz method.

Inclusion Method

Let $(H, (\cdot|\cdot))$ be an infinite-dimensional Hilbert space with the inner product $(\cdot|\cdot)$ and the norm $\|\cdot\|$. Suppose that V is a dense subspace of H and that one has the inner product $[\cdot|\cdot]$ in V such that $(V, [\cdot|\cdot])$ is a Hilbert space (the norm in V is denoted by $\|\cdot\|_V$). The embedding $V \hookrightarrow H$ is assumed to be compact.

One can consider the right-definite eigenvalue problem

$$\begin{cases} \text{Find} & \lambda \in \mathbb{R} \text{ and } \varphi \in V,\ \varphi \neq 0, \\ \text{s.t.} & [\varphi|v] = \lambda(\varphi|v) \text{ for all } v \in V. \end{cases} \tag{1}$$

Problem (1) has a countable spectrum of eigenvalues, and the eigenvalues can be ordered by magnitude:

$$0 < \lambda_1 \leq \lambda_2 \leq \cdots, \quad \lim_{j\to\infty} \lambda_j = \infty.$$

The Rayleigh–Ritz procedure for calculating upper bounds is a discretization of the Poincaré principle (cf. [9, Chapt. 22])

$$\lambda_j = \min_{\substack{E\subset V \\ \dim E = j}} \max_{\substack{u\in E \\ u\neq 0}} \frac{[u|u]}{(u|u)}, \quad j \in \mathbb{N}. \tag{2}$$

If the linearly independent trial functions

$$u_1, \ldots, u_n \in V, \quad n \in \mathbb{N},$$

are chosen, one can reduce (2) to the n-dimensional subspace V_n (the span of the chosen functions $\{u_1, \ldots, u_n\}$) and obtains the values

$$\Lambda_1^{[n]} \leq \cdots \leq \Lambda_n^{[n]},$$

which are upper bounds to the following λ_j:

$$\lambda_j \leq \Lambda_j^{[n]}, \quad j = 1, \ldots, n.$$

$\Lambda_j^{[n]}$ is called a *Rayleigh–Ritz bound* for λ_j. Now one forms the real $n \times n$-matrices

$$\begin{cases} A_0 := ((u_i|u_k))_{i,k=1,\ldots,n}, \\ A_1 := ([u_i|u_k])_{i,k=1,\ldots,n}, \end{cases} \tag{3}$$

the Rayleigh–Ritz bounds are the eigenvalues of the matrix eigenvalue problem

$$A_1 x = \Lambda^{[n]} A_0 x, \quad (\Lambda^{[n]}, x) \in \mathbb{R} \times \mathbb{R}^n. \tag{4}$$

The Rayleigh–Ritz bounds are monotonically decreasing in $n \in \mathbf{N}$.

The Lehmann–Goerisch procedure (see [3,4,5,6,7]) for calculating lower bounds can be understood as the discretization of a variational principle for characterizing the eigenvalues as well. This principle and a proof of the method is due to S. Zimmermann and U. Mertins [10].

Let $\rho \in \mathbf{R}$ be a spectral parameter such that for an $N \in \mathbf{N}$ the inequality

$$\lambda_N < \rho < \lambda_{N+1} \tag{5}$$

holds true. One expresses the first N eigenvalues in the form

$$\lambda_{N+1-i} = \rho + \frac{1}{\sigma_i}, \quad i = 1, \ldots, N$$

(assuming $\sigma_i < 0$). For $u \in \mathbf{V}$, $w_u \in H$ denotes the uniquely determined solution of the equation

$$[u|v] = (w_u|v) \quad \text{for all } v \in V,$$

the following σ_i therefore are characterized by

$$\sigma_i = \inf_{\substack{E\subset V \\ \dim E = i}} \max_{\substack{u\in E \\ u\neq 0}} \frac{[u|u] - \rho(u|u)}{(w_u|w_u) - 2\rho[u|u] + \rho^2(u,u)}, \tag{6}$$

$i = 1, \ldots, N$. A negative upper bound for σ_i results in a lower bound for λ_{N+1-i}. In order to discretize (6), one determines $w_1, \ldots, w_n \in H$ such that

$$[u_i|v] = (w_i|v) \quad \text{for all } v \in V, \tag{7}$$

then one defines the matrix

$$A_2 := ((w_i|w_k))_{i,k=1,\ldots,n}, \tag{8}$$

and solves the matrix eigenvalue problem

$$(A_1 - \rho A_0)\, x = \tau \left(A_2 - 2\rho A_1 + \rho^2 A_0\right) x, \quad (\tau,\ x) \in \mathbb{R} \times \mathbb{R}^n. \tag{9}$$

If for $n \in \mathbf{N}$ the condition $\Lambda_N^{[n]} < \rho$ is fulfilled, then (9) has exactly N negative eigenvalues $\tau_1 \le \cdots \le \tau_N < 0 \le \cdots \le \tau_n$. These τ_i are upper bounds for our σ_i ($\sigma_i \le \tau_i$, $i = 1, \ldots, N$). One obtains the lower bounds

$$\Lambda_j^{\rho[n]} := \rho + \frac{1}{\tau_{N+1-j}} \le \lambda_j, j = 1, \ldots, N. \quad (10)$$

This discretization (9), (10) is the Lehmann–Goerisch procedure. $\Lambda_j^{\rho[n]}$ is called a *Lehmann–Goerisch bound* for λ_j.

Numerical Example

The numerical example is the well known Mathieu equation. This equation has been considered by several authors, bounds for eigenvalues of the Mathieu equation can be found in [1,9] and [3]. The eigenvalue problem reads as follows

$$-\Phi''(x) + s\cos^2(x)\Phi(x) = \lambda\Phi(x), \quad x \in \left[0, \frac{\pi}{2}\right],$$
$$\Phi'(0) = \Phi'\left(\frac{\pi}{2}\right) = 0,$$

where $s \in \mathbf{R}$, $s > 0$, is a parameter.

In order to treat this problem, the required quantities can be defined as follows: $I := (0, \pi/2)$,

$$H := L_2(I), \quad V := H^1(I).$$

The inner products $(\cdot,\cdot)$ and $[\cdot,\cdot]$ are given by

$$(f, g) := \int_0^{\frac{\pi}{2}} f(x)\, g(x)\, dx \quad \text{for all } f, g \in H,$$
$$[f, g] := \int_0^{\frac{\pi}{2}} \left(f'(x)g'(x) + s\cos^2(x)f(x)g(x)\right)\, dx$$

for all $f, g \in V$.

With this definition the inner product $[\cdot,\cdot]$ and the usual H^1 inner product are equivalent; the embedding $(V, [\cdot,\cdot]) \hookrightarrow (H, (\cdot,\cdot))$ is compact.

Now the eigenvalue problem

$$\begin{cases} \text{Find} & \lambda \in \mathbb{R} \text{ and } \varphi \in V,\ \varphi \neq 0 \\ \text{s.t.} & [\varphi|v] = \lambda(\varphi|v) \text{ for all } v \in V. \end{cases}$$

is equivalent to the Mathieu equation. The trial functions $v_k \in V$ are defined by

$$\begin{aligned} &v_1(x) := 1, \\ &v_k(x) := \cos(2(k-1)x) \quad (11) \\ &\text{for } x \in I, \quad k = 2, \ldots, n. \end{aligned}$$

With these trial functions the Rayleigh–Ritz upper bounds $\Lambda_i^{[n]}$ (cf. (3), (4)) can be computed. For $n = 5$ one obtains

i	$\Lambda_i^{[5]}$
1	2.28404873592
2	8.4560567005
3	19.606719005
4	39.5439779
5	67.609198

The quality of these upper bounds can be increased by increasing n.

An application of the Lehmann–Goerisch procedure requires a spectral parameter ρ which is a rough eigenvalue bound (cf. (5)). For this aim the Mathieu equation is considered for $s = 0$. This is a second order problem with constant coefficients and can be solved in closed form. Its eigenvalues are $\widetilde{\lambda}_i = 4(i-1)^2$, $i \in \mathbf{N}$. From the comparison theorem (see [3]) one can see that the $\widetilde{\lambda}_i$ are lower bounds for the eigenvalues of the Mathieu equation with $s > 0$; this can be used to verify the left hand side inequality of (5), the right-hand side inequality can be examined by means of the Rayleigh–Ritz bounds. For $N = 4$ one obtains

$$\lambda_3 \le \Lambda_3^{[n]} \le 19.607 < \rho := \widetilde{\lambda}_4 = 36 < \lambda_4.$$

If s is increased dramatically, it may be impossible to satisfy (5). If this happens, one can link the eigenvalue problem under consideration and the comparison problem by a homotopy method (cf. [3]).

The next task is the determination of $w_i \in H$ such that (7) holds true. In general this is a problem, but for differential equations, where the right-hand side is the identity, one can proceed as follows: The operator on the left-hand side of the differential equation is denoted by M; then the trial functions v_i are chosen from $D(M)$ (that means sufficiently smooth) such that all essential and natural boundary conditions are satisfied. Now $w_i := M\, v_i$ fulfills (7). For the Mathieu equation one can define

$$(Mf)(x) := -f''(x) + s\cos^2(x)f(x)$$

and

$$\widetilde{V} := \left\{f \in H^2(I):\ f'(0) = f'\left(\frac{\pi}{2}\right) = 0\right\};$$

now it is easy to see that the v_i from (11) fulfill $v_i \in \widetilde{V}$ and $w_i := M\, v_i$ can be used in (7), (8).

From the eigenvalues of the matrix eigenvalue problem (9) one obtains the following bounds:

i	$\Lambda_i^{\rho[5]}$	$\Lambda_i^{[5]}$
1	2.28404873561	2.28404873592
2	8.4560566942	8.4560567005
3	19.6067171	19.6067191

For an example with a system of ordinary differential equations see [2].

See also

- ▶ αBB Algorithm
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Semidefinite Programming and Determinant Maximization

References

1. Albrecht J (1964) Iterationsverfahren zur Berechnung der Eigenwerte der Mathieuschen Differentialgleichung. Z Angew Math Mechanics 44:453–458
2. Behnke H (1996) A numerically rigorous proof of curve veering in an eigenvalue problem for differential equations. Z Anal Anwend 15:181–200
3. Behnke H, Goerisch F (1994) Inclusions for eigenvalues of selfadjoint problems. In: Herzberger J (ed) Topics in validated computations. Elsevier, Amsterdam, pp 277–322
4. Goerisch F (1980) Eine Verallgemeinerung eines Verfahrens von N.J. Lehmann zur Einschließung von Eigenwerten. Wiss Z Techn Univ Dresden 29:429–431
5. Goerisch F, Haunhorst H (1985) Eigenwertschranken für Eigenwertaufgaben mit partiellen Differentialgleichungen. Z Angew Math Mechanics 65(3):129–135
6. Lehmann NJ (1949) Beiträge zur Lösung linearer Eigenwertprobleme I. Z Angew Math Mechanics 29:341–356
7. Lehmann NJ (1950) Beiträge zur Lösung linearer Eigenwertprobleme II. Z Angew Math Mechanics 30:1–16
8. Maehly HJ (1952) Ein neues Verfahren zur genäherten Berechnung der Eigenwerte hermitescher Operatoren. Helv Phys Acta 25:547–568
9. Weinstein A, Stenger W (1972) Methods of intermediate problems for eigenvalues. Acad. Press, New York
10. Zimmermann S, Mertins U (1995) Variational bounds to eigenvalues of self-adjoint problems with arbitrary spectrum. Z Anal Anwend 14:327–345

Ellipsoid Method

Steffen Rebennack
Center of Applied Optimization,
University of Florida,
Gainesville, USA

MSC2000: 90C05

Article Outline

Keywords
Abstract
Introduction
Method
  The Basic Ellipsoid Algorithm
  Polynomially Running Time: Avoiding the Assumptions
  Modifications
Applications
  Linear Programming
  Separation and Optimization
Conclusion
See also
References

Keywords

Ellipsoid method; Linear programming; Polynomially solvable; Linear inequalities; Separation problem

Abstract

In this article, we give an overview of the Ellipsoid Method. We start with a historic introduction and provide a basic algorithm in Sect. "Method". Techniques to avoid two important assumptions required by this algorithm are considered in Sect. "Polynomially Running Time: Avoiding The Assumptions". After the discussion of some implementation aspects, we are able to show the polynomial running time of the Ellipsoid Method. The second section is closed with some modifications in order to speed up the running time of the ellipsoid algorithm. In Sect. "Applications", we discuss some theoretical implications of the Ellipsoid Method to linear programming and combinatorial optimization.

Introduction

In 1979, the Russian mathematician Leonid G. Khachiyan published his famous paper with the title "A Polynomial Algorithm in Linear Programming", [11]. He was able to show that linear programs (LPs) can be solved efficiently; more precisely that LP belongs to the class of polynomially solvable problems. Khachiyan's approach was based on ideas similar to the Ellipsoid Method arising from convex optimization. These methods were developed by David Yudin and Arkadi Nemirovski, [24,25,26], and independently by Naum Shor, [20], preceded by other methods as for instance the Relaxation Method, Subgradient Method or the Method of Central Sections, [2]. Khachiyan's effort was to modify existing methods enabling him to prove the polynomial running time of his proposed algorithm. For his work, he was awarded with the Fulkerson Prize of the American Mathematical Society and the Mathematical Programming Society, [16,21].

Khachiyan's four-page note did not contain proofs and was published in the journal Soviet Mathematics Doklady in February 1979 in Russian language. At this time he was 27 years young and quite unknown. So it is not surprising that it took until the Montreal Mathematical Programming Symposium in August 1979 until Khachiyan's breakthrough was discovered by the mathematical world and a real flood of publications followed in the next months, [23]. In the same year, *The New York Times* made it front-page news with the title "A Soviet Discovery Rocks World of Mathematics". In October 1979, the *Guardian* titled "Soviet Answer to Traveling Salesmen" claiming that the Traveling Salesman problem has been solved – based on a fatal misinterpretation of a previous article. For an amusing outline of the interpretation of Khachiyan's work in the world press, refer to [15].

Method

The Ellipsoid Method is designed to solve decision problems rather than optimization problems. Therefore, we first consider the decision problem of finding a feasible point to a system of linear inequalities

$$A^{\mathrm{T}}x \le b \tag{1}$$

where A is a $n \times m$ matrix and b is an n-dimensional vector. From now on, we assume all data to be integral and n to be greater or equal than 2. The goal is to find a vector $x \in \mathbb{R}^n$ satisfying (1) or to prove that no such x exists. We see in Sect. "Linear Programming" that this problem is equivalent to a linear programming optimization problem of the form

$$\min_x \; c^{\mathrm{T}}x \quad s.t. \quad A^{\mathrm{T}}x \le b\,, \quad x \ge 0\,,$$

in the sense that any algorithm solving one of the two problems in polynomial time can be modified to solve the other problem in polynomial time.

The Basic Ellipsoid Algorithm

Roughly speaking, the basic idea of the Ellipsoid Method is to start with an initial ellipsoid containing the solution set of (1). The center of the ellipsoid is in each step a candidate for a feasible point of the problem. After checking whether this point satisfies all linear inequalities, one either produced a feasible point and the algorithm terminates, or one found a violated inequality. This is used to construct a new ellipsoid of smaller volume and with a different center. Now the procedure is repeated until either a feasible point is found or a maximum number of iterations is reached. In the latter case, this implies that the inequality set has no feasible point.

Let us now consider the presentation of ellipsoids. It is well known, that the n-dimensional ellipsoid with center x^0 and semi-axis g_i along the coordinate axis is defined as the set of vectors satisfying the equality

$$\sum_{i=1}^{n} \frac{(x_i - x_i^0)^2}{g_i^2} = 1. \tag{2}$$

More general, we can formulate an ellipsoid algebraically as the set

$$E := \{x \in \mathbb{R}^n \mid (x - x^0)^{\mathrm{T}} B^{-1} (x - x^0) = 1\}, \tag{3}$$

with symmetric, positive definite, real-valued $n \times n$ matrix B. This can be seen with the following argument. As matrix B is symmetric and real-valued, it can be diagonalized with a quadratic matrix Q, giving $D = Q^{-1}BQ$, or equivalently, $B = QDQ^{-1}$. The entries of D are the eigenvalues of matrix B, which are positive and real-valued. They will be the quadratic, reciprocal values of the semi-axis g_i. Inserting the relationship for B into (3)

yields to $(x-x^0)^{\mathrm{T}} Q D^{-1} Q^{-1} (x-x^0) = 1$ which is equivalent to

$$\left((x-x^0)^{\mathrm{T}} Q\right) D^{-1} \left((x-x^0)^{\mathrm{T}} Q\right)^{\mathrm{T}} = 1.$$

Hence, we can interpret matrix Q as a coordinate transform to the canonical case where the semi-axis of the ellipse are along the coordinate axis. Recognize that in the case when matrix B is a multiple of the unit matrix, $B = r^2 \cdot I$, then the ellipsoid gets a sphere with radius $r > 0$ and center x^0. We abbreviate this in the following by $S(x^0, r)$.

We start with a somewhat basic version of the ellipsoid algorithm. This method requires two important assumptions on the polyhedron

$$P := \{x \in \mathbb{R}^n \mid Ax \leq b\}.$$

We assume that

1. the polyhedron P is bounded and that
2. P is either empty or full-dimensional.

In Sect. "Polynomially Running Time: Avoiding The Assumptions", we will see how this algorithm can be modified not needing these assumptions. This will allow us to conclude that a system of linear inequalities can be solved in polynomial running time with the Ellipsoid Method.

Let us now discuss some consequences of these two assumptions. The first assumption allows us to construct a sphere $S(c_0, R)$ with center c_0 and radius R containing P completely: $P \subseteq S(c_0, R)$. The sphere $S(c_0, R)$ can be constructed, for instance, in the following two ways. If we know the bounds on all variables x, e. g. $L_i \leq x_i \leq U_i$, one can use a geometric argument to see that with

$$R := \sqrt{\sum_{i=1}^{n} \max\{|L_i|, |U_i|\}^2},$$

the sphere $S(0, R)$ will contain the polytope P completely. In general, when such bounds are not given explicitly, one can use the integrality of the data and proof, see for instance [6], that the sphere with center 0 and radius

$$R := \sqrt{n} 2^{\langle A\rangle + \langle b\rangle - n^2} \tag{4}$$

contains P completely, where $\langle\cdot\rangle$ denotes the encoding length of some integral data. For an integer number b_i, we define

$$\langle b_i\rangle := 1 + \lceil \log_2(|b_i| + 1)\rceil,$$

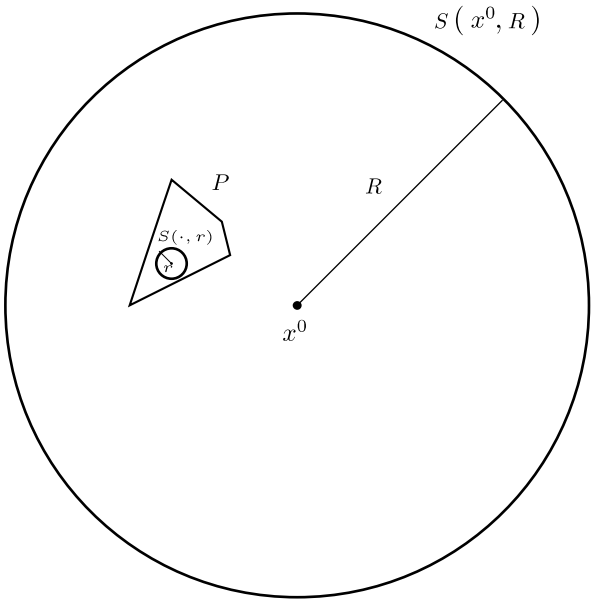
which is the number of bits needed to encode integer b_i in binary form; one bit for the sign and $\lceil \log_2(|b_i| + 1)\rceil$ bits to encode $|b_i|$. With this, the encoding length of a vector b is the sum of the encoding lengths of its components. Similarly, for a matrix A, the encoding length is given by $\langle A\rangle := \sum_{i=1}^{m} \langle a_i\rangle$.

The second assumption implies that if P is non-empty, its volume is strictly positive, meaning that there is an n-dimensional sphere of radius $r > 0$ which is contained in P. More precisely, it is possible to show that

$$vol(P) \geq 2^{-(n+1)\langle A\rangle + n^3}, \tag{5}$$

in the case that P is not empty, see [5]. This will help us to bound the number of iterations of the basic ellipsoid algorithm. The graphical interpretation of the positive volume of polytope P is that the solution set of system (1) is not allowed to have mass zero, for instance, not to be a hyperplane.

Figure 1 illustrates the effect of the two assumptions in the case that P is non-empty. As this is a two-dimensional example, the second assumption implies that



Ellipsoid Method, Figure 1
P is bounded and full-dimensional

polytope P is not just a line segment but instead contains a sphere of positive radius r.

Now, we are ready to discuss the main steps of the basic ellipsoid algorithm. Consider therefore Algorithm 1. In the first six steps the algorithm is initialized. The first ellipsoid is defined in step three. The meaning of the parameters for the ellipsoid and especially number k^* will be discussed later in this section. For now, let us also ignore step seven. Then, for each iteration k, it is checked in step eight if center x^k satisfies the linear inequality system (1). This can be done for instance by checking each of the m inequalities explicitly. In the case that all inequalities are satisfied, x^k is a feasible point and the algorithm terminates. In the other case there is an inequality $a_j^{\mathrm{T}} x \leq b_j$ which is violated by x^k, step nine. In the next two steps, a new ellipsoid E_{k+1} is constructed. This ellipsoid has the following properties. It contains the half ellipsoid

$$H := E_k \cap \{x \in \mathbb{R}^n \mid a_j^{\mathrm{T}} x \leq a_j^{\mathrm{T}} x^k\} \tag{6}$$

which insures that the new ellipsoid E_{k+1} also contains polytope P completely, assuming that the initial ellipsoid $S(0, R)$ contained P. Furthermore, the new ellipsoid has the smallest volume of all ellipsoids satisfying (6), see [2]. The central key for the proof of polynomial running time of the basic ellipsoid Algorithm 1 is another property, the so called *Ellipsoid Property*. It provides the following formula about the ratio of the volumes of the ellipsoids

$$\frac{volE_{k+1}}{volE_k} = \sqrt{\left(\frac{n}{n+1}\right)^{n+1}\left(\frac{n}{n-1}\right)^{n-1}} < \exp\left(-\frac{1}{2n}\right), \tag{7}$$

for a proof see, for instance, [5,17]. As $\exp(-1/(2n)) < 1$ for all natural numbers n, the new ellipsoid has a strict smaller volume than the previous one. We also notice that $\exp(-1/(2n))$ is a strictly increasing function in n which has the consequence that the ratio of the volumes of the ellipsoids is closer to one when the dimension of the problem increases.

Now, let us discuss the situation when P is empty. In this case, we want Algorithm 1 to terminate in step seven. To do so, we will derive an upper bound k^* on the number of iterations needed to find a center x^k satisfying the given system of linear inequalities for the case that P is not empty. Clearly, if the algorithm would need more than k^* iterations, the polytope P must be empty. Therefore, let us assume again that $P \neq \emptyset$. In this case (5) provides a lower bound for the volume of P and an upper bound of its volume is given, for instance, by (4). In addition, according to the construction of the ellipsoids, we know that each of the E_{k+1} contains P completely. Together with the Ellipsoid Property (7) we get the relation

$$vol(E_{k^*}) < \exp\left(\frac{-k^*}{2n}\right) vol(E_0) < 2^{\frac{k^*}{2n}+n+n\log(R)} \stackrel{!}{=} 2^{-(n+1)\langle C\rangle+n^3} \leq vol(P).$$

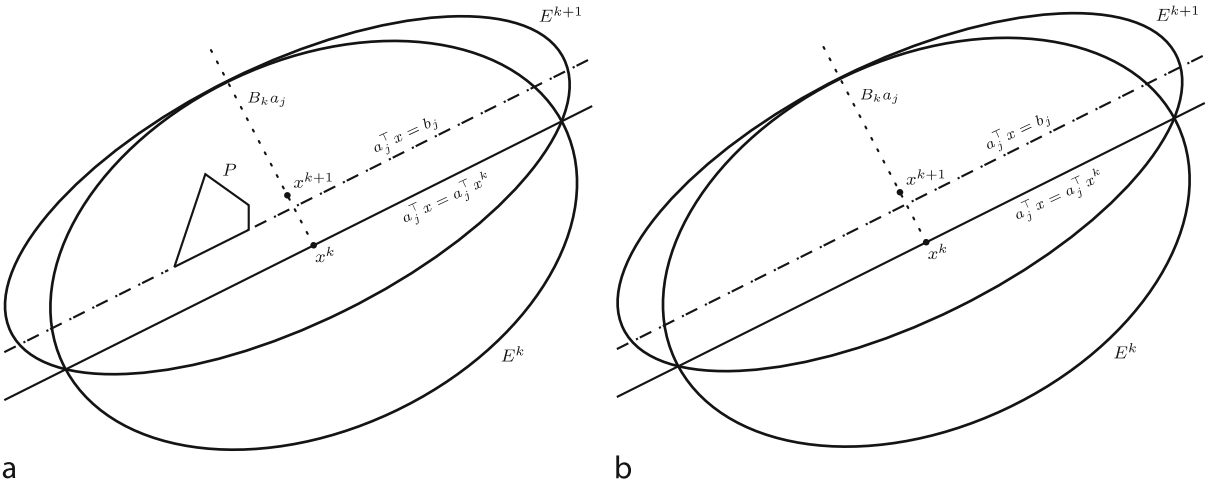
This chain provides an equation defining the maximum number of iterations

$$k^* := 2n(2n+1)\langle C\rangle + 2n^2(\log(R) - n^2 + 1). \tag{8}$$

Input: Matrix A, vector b; sphere $S(c_0, R)$ containing P
Output: feasible $\overline{x}$ or proof that $P = \emptyset$

// Initialize
1: $k := 0$
2: $k^* = 2n(2n+1) < C > +2n^2(\log(R) - n^2 + 1)$ // Max number of iterations
3: $x^0 = c_0$, $B_0 = R^2 \cdot I$ // Initial ellipsoid
4: $\tau := \frac{1}{n+1}$ // Parameter for ellipsoid: step
5: $\sigma := \frac{2}{n+1}$ // Parameter for ellipsoid: dilation
6: $\delta := \frac{n^2}{n^2-1}$ // Parameter for ellipsoid: expansion
// Check if polytope P is empty
7: **if** $k = k^*$ **return** $P = \emptyset$
// Check feasibility
8: **if** $A x^k \leq b$ **return** $\overline{x} := x^k$ // $\overline{x}$ is a feasible point
9: **else** let $a_j^{\mathrm{T}} x^k > b_j$
// Construct new ellipsoid
10: $B_{k+1} := \delta(B_k - \sigma \frac{B_k a_j (B_k a_j)^{\mathrm{T}}}{a_j^{\mathrm{T}} B_k a_j})$
11: $x^{k+1} := x^k - \tau \frac{B_k a_j}{\sqrt{a_j^{\mathrm{T}} B_k a_j}}$
// Loop
12: $k \leftarrow k+1$ and goto step 7

Ellipsoid Method, Algorithm 1
Basic Ellipsoid Algorithm



Ellipsoid Method, Figure 2
Basic ellipsoid algorithm. a $P6 \neq \emptyset$. **b** $P = \emptyset$

A geometric interpretation is that the volume of the ellipsoid in the k^*th iteration would be too small to contain the polytope P. Obviously, this implies that P has to be empty. With this, we have shown that the presented basic ellipsoid algorithm works correctly.

One iteration of the basic ellipsoid algorithm, for the case of a non-empty polytope P, is illustrated in Fig. 2a. We recognize that P is contained completely in E_k. The dashed line shows equality $a_j^{\mathrm{T}} x \leq b_j$ corresponding to one of the inequalities which are violated by x^k. Geometrically, this equality is moved in parallel until it contains center x^k. Recognize that the new ellipsoid E_{k+1} contains the half ellipsoid (6). In this case, the new center x_{k+1} is again not contained in polytope P and at least one more step is required. The case that polytope P is empty is illustrated in Fig. 2b which is mainly the same as Fig. 2a.

In the case that B_k is a multiple of the identity, the ellipsoid is an n-dimensional sphere. According to the initialization of the basic ellipsoid algorithm in step three, this is the case for the first iteration when $k = 0$. This gives us an interpretation of the values of δ and τ. The new ellipsoid E_{k+1} is shrunk by the factor $\sqrt{\delta(1-\sigma)} = n/(n+1)$ in the direction of vector a_j and expanded in all orthogonal directions by factor $\sqrt{\delta} = n/\sqrt{n^2-1}$. Hence, we see that in the next step we do no longer get a sphere if we start with one. The third parameter, the step, gives intuitively the measure how far we go from point x^k in the direction of vector $B_k a_j$, multiplied by factor $1/\sqrt{a_j^{\mathrm{T}} B_k a_j}$. For more details we refer to the survey of Bland, Goldfarb and Todd, [2].

After all the discussions above, we are now able to conclude the polynomial running time of the basic ellipsoid Algorithm 1 for the case that the polyhedron P is bounded and either empty or full-dimensional. For an algorithm to have polynomial running time in the deterministic Turing-machine concept, there has to be a polynomial in the encoding length of the input data describing the number of elementary steps needed to solve an arbitrary instance of the problem. Therefore, let $L := \langle A, b \rangle$ be the encoding length of the input data. Obviously, each of the steps of Algorithm 1 can be done in polynomial many steps in L. The maximum number of iterations is given through k^* which is also a polynomial in L; consider therefore equations (8) and (4). Hence, we conclude that the ellipsoid Algorithm 1 has a polynomial running time. During the reasoning above, we required implicitly one more important assumption, namely to have exact arithmetic. Normally, this is of more interest in numerics rather than in theoretical running time analysis. However, it turns out that for the Ellipsoid Method this is crucial as the Turing-machine concept also requires finite precision. Let us postpone this topic to the end of the next subsection and rather discuss methods to avoid the two major assumptions on polyhedron P.

Polynomially Running Time: Avoiding the Assumptions

In order to prove that the Ellipsoid Method can solve the system of linear inequalities (1) in polynomial time, one has to generalize the basic ellipsoid Algorithm 1 to need not the assumptions that 1. polyhedron P is bounded, 2. P is either empty or full-dimensional and 3. that exact arithmetic is necessary.

In 1980, Khachiyan published a paper, [12], discussing all the details and proofs about the Ellipsoid Method which were neglected in his paper from 1979. In Lemma 1, he showed that

$$P \cap S(0, 2^L) \neq \emptyset, \tag{9}$$

in the case that $P \neq \emptyset$. With this, one can use for R of (4) value 2^L. However, we cannot just adopt the algorithm above to the new situation. Instead of a lower bound for $vol(P)$, as given in (5), we would need a lower bound for $vol(P \cap S(0, 2^L))$. To achieve this, we follow a trick introduced by Khachiyan and consider the perturbed system of linear inequalities

$$2^L a_i^{\mathrm{T}} x \leq 2^L \beta_i + 1 \quad i = 1, \ldots, m. \tag{10}$$

Let us abbreviate the corresponding solution set with P', which is in general a polyhedron. Khachiyan was able to proof a one-to-one correspondence of the original system (1) and the perturbed one (10). This means that P is empty if and only if P' is empty, it is possible to construct a feasible point for (1) out of (10) in polynomial time and the formulation of the perturbed system is polynomial in L. Furthermore, the new inequality system has the additional property that if $P' \neq \emptyset$ it is full-dimensional. Hence, it is possible to find a (non-empty) sphere included in P'. It can be shown, that

$$S(\overline{x}, 2^{-2L}) \subseteq P' \cap S(0, 2^L),$$

where $\overline{x}$ is any feasible point of the original system (1) and hence $\overline{x} \in P$. With this argument at hand, it is possible to derive an upper bound for the number of iterations for the Ellipsoid Method by solving the perturbed system (10). It can be shown that a feasible point can be found in at most $6n(n+1)L$ iterations, [2].

With the perturbation of the original system and property (9), we do no longer require that P is bounded. As a byproduct, polyhedron P has not to be of full-dimension in the case that it is non empty; as system (10) is of full-dimension independent of whether P is or not, assuming that $P \neq \emptyset$. As a consequence, the basic ellipsoid algorithm can be generalized to apply for any polyhedron P and the two major assumptions are no longer necessary.

During all of the reasoning, we assumed to have exact arithmetic, meaning that no rounding errors during the computation are allowed. This implies that all data have to be stored in a mathematically correct way. As we use the Turing-machine concept for the running time analysis, we require that all computations have to be done in finite precision. Let us now have a closer look for the reason why this is crucial for ellipsoid Algorithm 1.

The presentation of the ellipsoid with the matrix B_k in (3) yields to the convenient update formulas for the new ellipsoid, parameterized by B_{k+1} and x^{k+1}. However, to obtain the new center x^{k+1} one has to divide by factor $\sqrt{a_j^{\mathrm{T}} B_k a_j}$. If we work with finite precision, rounding errors are the consequence, and it is likely that matrix B_k is no longer positive definite. This may cause that $a_j^{\mathrm{T}} B_k a_j$ becomes zero or negative, implying that the ellipsoid algorithm fails.

Hence, to implement the ellipsoid method, one has to use some modifications to make it numerically stable. One basic idea is to use factorization

$$B_k = L_k D_k L_k^{\mathrm{T}}$$

for the positive definite matrix B_k, with L being a lower triangular matrix with unit diagonal and diagonal matrix D_k with positive diagonal entries. Obtaining such a factorization is quite expensive as it is of order n^3. But there are update formulae applying for the case of the ellipsoid algorithm which have only quadratic complexity. Already in 1975, for such a type of factorization, numerically stable algorithms have been developed, insuring that D_k remains positive definite, see [7]. We skip the technical details here and refer instead to Golfarb and Todd, [9].

With a method at hand which can handle the ellipsoid algorithm in finite precision numerically stable, the proof of its polynomial running time is complete.

Modifications

In this subsection, we briefly discuss some straightforward modifications of the presented ellipsoid method in order to improve its convergence rate. Therefore, let us consider Fig. 2 once more. From this figure it is intuitively clear that an ellipsoid containing set $\{x \in E_k \mid a_j^T x \leq b\}$ has a smaller volume than the one containing the complete half-ellipsoid (6). In the survey paper of Bland et al. it is shown that the smallest ellipsoid, arising from the so called deep cut $a_j^T x \leq b$, can be obtained by choosing the following parameters for Algorithm 1

$$\tau := \frac{1 + n\alpha}{n+1},$$
$$\sigma := \frac{2 + 2n\alpha}{(n+1)(1+\alpha)},$$
$$\delta := \frac{n^2}{(n^2-1)(1-\alpha^2)},$$

with

$$\alpha := \frac{a_j^T - b_j}{\sqrt{a_j^T B_k a_j}}.$$

The parameter α gives an additional advantage of this deep cut, as it is possible to check infeasibility or for redundant constraints, [19].

Another idea could be to use a whole system of violated inequalities as a cut instead of only one. Such type of cuts are called surrogate cuts and were discussed by Goldfarb and Todd. An iterative procedure to generate these cuts was described by Krol and Mirman, [14].

Consider now the case that the inequality system (1) contains two parallel constraints which means that they differ only in the right hand side. With this it is possible to generate a new ellipsoid containing the information of both inequalities. These cuts are called *parallel cuts*. Update formulas for B_k and x^k were discovered independently by several authors. For more details, we refer to [2,19].

However, all modifications which have been found so far do not allow to reduce the worst case running time significantly – they especially do not allow to avoid the presence of L. This implies that the running time does not only depend on the size of the problem but also on the magnitude of the data.

At the end of the second chapter, we point out that the Ellipsoid Method can also be generalized to use other convex structures as ellipsoids. Methods working for instance only with spheres, or triangles, are possible. The only crucial point is that one has to make sure that its polynomial running time can be proven. Furthermore, the underlying polytope can be generalized to any convex set; for which the separation problem can be solved in polynomial time, see Sect. "Separation and Optimization".

Applications

In 1981, Grötschel, Lovász and Schrijver used the Ellipsoid Method to solve many open problems in combinatorial optimization. They developed polynomial algorithms, for instance, for the vertex packing in perfect graphs, and could show that the weighted fractional chromatic number is $\mathcal{NP}$-hard, [5]. Their proofs were mainly based on the relation of separation and optimization, which could be established with the help of the Ellipsoid Method. We discuss this topic in Sect. "Separation and Optimization" and give one application for the maximum stable set problem. For all other interesting results, we refer to [5]. But first, we consider another important application of the Ellipsoid Method. We examine two concepts showing the equivalence of solving a system of linear inequalities and to find an optimal solution to a LP. This will prove that LP is polynomial solvable.

Linear Programming

As we have seen in the last section, the Ellipsoid Method solves the problem of finding a feasible point of a system of linear inequalities. This problem is closely related to the problem of solving the linear program

$$\max_x \; c^T x \quad s.t. \quad A^T x \leq b\,, \quad x \geq 0\,. \tag{11}$$

Again, we assume that all data are integral. In the following we briefly discuss two methods of how the optimization problem (11) can be solved in polynomial time via the Ellipsoid Method. This will show that LP is in the class of polynomially solvable problems.

From duality theory, it is well known that solving the linear optimization problem (11) is equivalent to

finding a feasible point of the following system of linear inequalities

$$\begin{aligned} A^{\mathrm{T}}x &\leq b, \\ -x &\leq 0, \\ -Ay &\leq -c, \\ -y &\leq 0, \\ -c^{\mathrm{T}}x + b^{\mathrm{T}}y &\leq 0. \end{aligned} \tag{12}$$

The third and fourth inequality come from the dual problem of (11), insuring primal and dual feasibility of x and y, respectively. The last inequality results from the Strong Duality Theorem, implying that this inequality always has zero slack. The equivalence of the two problems means in this case that vector $\overline{x}$ of each solution pair $(\overline{x}, \overline{y})$ of (12) is an optimal solution of problem (11) and $\overline{y}$ is an optimum for the dual problem of (11). In addition, to each solution of the optimization problem exists a vector such that this pair is feasible for problem (12).

From the equivalence of the two problems (11) and (12) we immediately conclude that the linear programming problem can be solved in polynomial time; as the input data of (12) are polynomially bounded in the length of the input data of (11). This argument was used by Gács and Lovász in their accomplishment to Khachiyan's work, see [4]. The advantage of this method is that the primal and dual optimization problem are solved simultaneously. However, note that with this method, one has no idea whether the optimization problem is infeasible or unbounded in the case when the Ellipsoid Method proves that problem (12) is infeasible. Another disadvantage is that the dimension of the problem increases from n to $n + m$.

Next we discuss the so called bisection method which is also known as binary search or sliding objective hyperplane method. Starting with an upper and lower bound of an optimal solution, the basic idea is to make the difference between the bounds smaller until they are zero or small enough. Solving system $A^{\mathrm{T}}x \leq b$, $x \geq 0$ with the Ellipsoid Method gives us either a vector $\overline{x}$ providing the lower bound $l := c^{\mathrm{T}}\overline{x}$ for problem (11) or in the case that the polyhedron is empty, we know that the optimization problem is infeasible. An upper bound can be obtained, for instance, by finding a feasible vector to the dual problem $Ay \geq c$, $y \geq 0$. If the Ellipsoid Method proves that the polytope of the dual problem is empty, we can use the duality theory (as we already know that problem (11) is not infeasible) to conclude that the optimization problem (11) is unbounded. In the other case we obtain vector $\overline{y}$ yielding to the upper bound $u := b^{\mathrm{T}}\overline{y}$ of problem (11), according to the Weak Duality Theorem. Once bounds are obtained, one can iteratively use the Ellipsoid Method to solve the modified problem $A^{\mathrm{T}}x \leq b$, $x \geq 0$ with the additional constraint

$$-c^{\mathrm{T}}x \leq -\frac{u+l}{2},$$

which is a constraint on the objective function value of the optimization problem. If the new problem is infeasible, one can update the upper bound to $\frac{u+l}{2}$, and in the case that the ellipsoid algorithm computes a vector $\overline{x}$, the lower bound can be increased to $c^{\mathrm{T}}\overline{x}$ which is greater or equal to $\frac{u+l}{2}$. In doing so, one at least bisects the gap in each step. However, this method does not immediately provide a dual solution. Note that only one inequality is added during the process, keeping the problem size small. More details and especially the polynomial running time of this method are discussed by Padberg and Rao [1].

Separation and Optimization

An interesting property of the ellipsoid algorithm is that it does not require an explicit list of all inequalities. In fact, it is enough to have a routine which solves the so called Separation Problem for a convex body K:

> *Given $z \in \mathbb{R}^n$, either conclude that $z \in K$ or give a vector $\pi \in \mathbb{R}^n$ such that inequality $\pi^{\mathrm{T}}x < \pi^{\mathrm{T}}z$ holds for all $x \in K$.*

In the latter case we say that vector π separates z from K. In the following, we restrict the discussion to the case when K is a polytope meeting the two assumptions of Sect. "The Basic Ellipsoid Algorithm". To be consistent with the notation, we write P for K. From the basic ellipsoid Algorithm 1 follows immediately that if one can solve the separation problem for polytope P polynomially in L and n, then the corresponding optimization problem

$$\max_x c^{\mathrm{T}}x \quad s.t. \quad x \in P$$

can also be solved in polynomial time in L and n; L is again the encoding length of the input data, see Sect. "The Basic Ellipsoid Algorithm". The converse statement was proven by Groetschel et al., yielding to the following equivalence of separation and optimization:

> *The separation problem and the optimization problem over the same family of polytopes are polynomially equivalent.*

Consider now an example to see how powerful the concept of separation is. Given a graph $G = (V, E)$ with node set V and edges $e \in E$. A stable set S of graph G is defined as a subset of V with the property that any two nodes of S are not adjacent; which means that no edge between them exists in E. To look for a maximum one is the maximum stable set problem. This is a well known optimization problem and proven to be $\mathcal{NP}$-hard, see [3]. It can be modeled, for instance, as the integer program

$$\max_x \; c^{\mathrm{T}} x$$
$$s.t. \quad x_i + x_j \leq 1 \quad \forall (i,j) \in E \quad x \in \{0,1\} \qquad (13)$$

with incidence vector x, meaning that $x_i = 1$ if node i is in a maximum stable set, otherwise it is zero. Constraints (13) are called edge inequalities. Relaxing the binary constraints for x gives the so-called trivial inequalities

$$0 \leq x \leq 1 \qquad (14)$$

yielding to a linear program. However, this relaxation is very weak; consider therefore a complete graph. To improve it, one can consider the odd-cycle inequalities

$$\sum_{i \in C} x_i \leq \frac{|C| - 1}{2} \qquad (15)$$

for each odd cycle C in G. Recognize that there are in general exponentially many such inequalities in the size of graph G. Obviously, every stable set satisfies them and hence they are valid inequalities for the stable set polytope. The polytope satisfying the trivial-, edge- and odd-cycle inequalities

$$P := \{x \in \mathbb{R}^{|V|} \mid x \text{ satisfies (14), (15) and (16)}\} \qquad (16)$$

is called the cycle-constraint stable set polytope. Notice that this polytope is contained strictly in the stable set polytope. It can be shown that the separation problem for polytope (16) can be solved in polynomial time. One idea is based on a construction of an auxiliary graph H with a double number of nodes. Solving a sequence of n shortest path problems on H solves then the separation problem with a total running time of order $|V| \cdot |E| \cdot \log(|V|)$. With the equivalence of optimization and separation, the stable set problem over the cycle-constraint stable set polytope can be solved in polynomial time. This is quite a remarkable conclusion as the number of odd-cycle inequalities may be exponential. However, note that it does not imply that the solution will be integral and hence, we cannot conclude that the stable set problem can be solved in polynomial time. But we can conclude that the stable set problem for t-perfect graphs can be solved in polynomial time; where a graph is called t-perfect, if the stable set polytope is equal to the cycle-constraint stable set polytope (16). For more details about this topic see, for instance [8,18].

Conclusion

In 1980, the Ellipsoid Method seemed to be a promising algorithm to solve problems practically [23]. However, even though many modifications to the basic ellipsoid algorithm have been made, the worst case running time still remains a function in n, m and especially L. This raises two main questions. First, is it possible to modify the ellipsoid algorithm to have a running time which is independent of the magnitude of the data, but instead depends only on n and m – or at least any other algorithm with this property solving LPs? (This concept is known as strongly polynomial running time.) The answer to this question is still not known and it remains an open problem. In 1984, Karmarkar introduced another polynomial running time algorithm for LP which was the start of the Interior Point Methods, [10]. But also his ideas could not be used to solve this question. For more details about this topic see [17,22]. The second question, coming into mind, is how the algorithm performs in practical problems. Unfortunately, it turns out that the ellipsoid algorithm tends to have a running time close to its worst-case bound and is inefficient compared to other methods. The Simplex Method

developed by George B. Dantzig in 1947, was proven by Klee and Minty to have exponential running time in the worst case, [13]. In contrast, its practical performance is much better and it normally requires only a linear number of iterations in the number of constraints.

Until now, the Ellipsoid Method has not played a role for solving linear programming problems in practice. However, the property that the inequalities themselves have not to be explicitly known, distinguishes the Ellipsoid Method from others, for instance from the Simplex Method and the Interior Point Methods. This makes it a theoretically powerful tool, for instance attacking various combinatorial optimization problems which was impressively shown by Grötschel, Lovász and Schrijver.

See also

- ► Linear Programming
- ► Linear Programming: Interior Point Methods
- ► Linear Programming: Karmarkar Projective Algorithm
- ► Linear Programming: Klee–Minty Examples
- ► Volume Computation for Polytopes: Strategies and Performances

References

1. Padberg MW, Rao MR (1980) The Russian Method for Linear Inequalities. Graduate School of Business Administration, New York University, New York
2. Bland RG, Goldfarb D, Todd MJ (1981) The Ellipsoid Method: A Survey. Operat Res 29(6):1039–1091
3. Garey MR, Johnson DS (1979) Computers and Intractability, A guide to the Theory of NP-Completeness. In: Klee V (ed) A series of books in the mathematical sciences. WH Freeman and Company, New York
4. Gács P, Lovász L (1981) Khachiyan's Algorithm for Linear Programming. Math Program Stud 14:61–68
5. Grötschel M, Lovász L, Schrijver A (1981) The Ellipsoid Method and Its Consequences in Combinatorial Optimization. Comb 1:169–197
6. Grötschel M, Lovász L, Schrijver A (1988) Geometric Algorithms and Combinatorial Optimization. In: Algorithms and Combinatorics 2. Springer, Berlin
7. Gill PE, Murray W, Saunders MA (1975) Methods for Computing and Modifying the LDV Factors of a Matrix. Math Comput 29(132):1051–1077
8. Gerards AMH, Schrijver A (1986) Matrices with the Edmonds-Johnson property. Comb 6(4):365–379
9. Goldfarb D, Todd MJ (1982) Modifications and Implementation of The Ellipsoid Algorithm for Linear Programming. Math Program 23:1–19
10. Karmarkar N (1984) A New Polynomial-Time Algorithm for Linear Programming. Comb 4(4):373–395
11. Khachiyan LG (1979) A Polynomial Algorithm in Linear Programming. Dokl Akad Nauk SSSR 244:1093–1096 (English translation: Sov Math Dokl, 20(1):191–194, 1979)
12. Khachiyan LG (1980) Polynomial Algorithms in Linear Programming. Zhurnal Vychislit Math Math Fiz 20:51–68 (English translation: USSR Comput Math Math Phys, 20(1):53–72, 1980)
13. Klee V, Minty GL (1972) How good is the Simplex Algorithm? Inequalities III. In: Shisha O (ed). Academic Press, New York, pp 159–175
14. Krol J, Mirman B (1980) Some Practical Modifications of The Ellipsoid Method for LP Problems. Arcon Inc, Boston
15. Lawler EL (1980) The Great Mathematical Sputnik of 1979. Science 20(7):12–15, 34–35
16. Murray W, Todd M (2005) Tributes to George Dantzig and Leonid Khachiyan. SIAM Act Group Optim – Views News 16(1–2):1–6
17. Nemhauser GL, Wolsey LA (1999) Integer and Combinatorial Optimization. In: Wiley-Interscience Series in Discret Mathematics and Optimization. Wiley, New York
18. Rebennack S (2006) Maximum Stable Set Problem: A Branch & Cut Solver. Diplomarbeit, Ruprecht–Karls Universität Heidelberg
19. Schrader R (1983) The Ellipsoid Method and Its Implications. OR Spektrum 5(1):1–13
20. Shor NZ (1977) Cut-off Method with Space Extension in Convex Programming Problems. Kibern 13(1):94–95 (English translation: Cybern 13(1):94–96, 1977)
21. Khachiyan L (2005) 1952–2005: An Appreciation. SIAM News 38(10)
22. Tardos E (1986) A Strongly Polynomial Algorithm to Solve Combinatorial Linear Programs. Oper Res 34(2):250–256
23. Tee GJ (1980) Khachian's efficient algorithm for linear inequalities and linear programming. SIGNUM Newsl 15(1):13–15
24. Yudin DB, Nemirovski AS (1976) Evaluation of the Informational Complexity of Mathematical Programming Problems. Èkon Math Metod 12:128–142 (English translation: Matekon, 13(2):3–25, 1976–7)
25. Yudin DB, Nemirovski AS (1976) Informational Complexity and Efficient Methods for the Solution of Convex Extremal Problems. Èkon Math Metod 12:357–369 (English translation: Matekon, 13(3):25–45, 1977)
26. Yudin DB, Nemirovski AS (1977) Optimization Methods Adapting to the "Significant" Dimension of the Problem. Autom Telemekhanika 38(4):75–87 (English translation: Autom Remote Control, 38(4):513–524, 1977)

Emergency Evacuation, Optimization Modeling

FATEMEH SAYYADY[2], SANDRA DUNI EKSIOGLU[1]
[1] Department of Industrial & Systems Engineering, Mississippi State University, Starkvilee, USA
[2] Operations Research Graduate Program, North Carolina State University, Raleigh, USA

Article Outline

Keywords and Phrases
Introduction
Applications
Community Evacuation
Community Evacuation: Trip Generation
Community Evacuation: Trip Departure Time
Community Evacuation: Trip Destination Selection
Community Evacuation: Trip Route Selection
Building Evacuation
Small-Area Evacuation
Real-Time Traffic Management
Agent-Based Modeling
Conclusions
References

Keywords and Phrases

Emergency evacuation; Optimization modeling; Simulation modeling; Real-time traffic management; Agent-based simulation

Introduction

Natural disasters (such as fires, hurricanes, tornadoes, flash floods, tsunamis, earthquakes, etc.) or man-made disasters (such as nuclear power plant explosions, chemical plant explosions, hazmat releases, dirty bomb threats, etc.) affect millions of people every year. Evacuation is an emergency management strategy used to ensure a population's safety in these type of situations. *Emergency evacuation is defined as the relocation of a threatened population to a safer area due to an immediate or predictable life-threatening danger* [25].

Prior to 1979, the models developed for evacuating people and vehicles from dangerous locations were mainly qualitative. However, the accident at the Three Mile Island nuclear power plant near Middletown, Pennsylvania, in 1979 provided a major motive for quantifying emergency response plans [36]. Since then, a number of optimization- and simulation-based models have been developed to identify evacuation strategies for communities (urban and rural areas), buildings and industrial plants, and residential areas. The most recent and challenging developments in this area are real-time traffic management and agent-based models.

Applications

Community Evacuation

Southworth [35] models a community evacuation plan using a five-step procedure that involves trip generation, trip departure time, trip destination, trip route selection, and evacuation plan set-up and analysis. The factors that affect any of these steps are the distribution of the population that is at risk, human behavior, transportation infrastructure, road capacity, vehicle utilization, accessibility of warning technologies, time available before the occurrence of the hazard, evacuees' route and destination selection, promptness in cleaning and preparing to operate the affected highways and roads, traffic management actions and the availability of non-evacuation based protective actions, such as in site sheltering [6,7,10,35,36]. The next sections of this article provide a summary of the research related to the above mentioned steps.

Community Evacuation: Trip Generation

The trip generation step determines the number of vehicles loaded to the traffic network during the evacuation. The number of vehicles loaded in the network depends on the population of the evacuation zone (which is space and time dependent), number of vehicles per household and vehicle utilization rate. The population of an evacuation area consists of the *permanent residents*, the *transients* (tourists and daily workers), and the *residents of special facilities* such as students, prisoners, patients, customers in shopping malls, and members at recreational facilities [35,36]. For a given residential area with a population size equal to N, Southworth [35] estimates the daytime population using the following equation:

$$D = H + W + P + S\,,$$

where W, P, and S denote the number of workers, students, and residents of special facilities, respectively. H denotes the number of people who stay at home during the day and is estimated by

$$H = [N - (W + P)] * (1 - s),$$

where s is the probability that a non-working adult (or a child) is not being engaged in shopping, recreational or social activities.

Based on Southworth [35], the vehicle utilization rate depends on the time of the evacuation, the household size, the average number of commuters per vehicle and the average number of workers and licensed drivers per household. Estimating vehicle utilization rate is challenging. This is why some post evacuation surveys report significantly different utilization rates. For example, Baker [1] estimates the vehicle utilization rate to be 52% and Lindell and Perry [23] 75%.

Community Evacuation: Trip Departure Time

This is the time it takes one to evacuate once the evacuation warning is released. The trip departure time consists of the time required to receive the official evacuation warning, the time required to leave the current location to get home, the time required to arrive home, and the time to prepare to leave home. Next, we give a summary of the approaches used to calculate trip departure time.

In 1984, Jamei [18] introduced the *mobilization curve* to estimate the percent of evacuees that enter the traffic network in specific time intervals. The mobilization curve is represented using the following equation:

$$P_t = \frac{1}{(1 + exp[-z(t - h)])},$$

where P_t is the cumulative percentage of traffic volume loaded in the network by time t, z is the response rate of the public to the disaster and is known as the slope of the mobilization curve, and h is the "half loading time". The loading time depends on the incident and its relative severity. Radwan et al. [29] and Hobeika and Kim [17] have incorporated the mobilization curve in mass evacuation computer programs (MASSVAC 3.0 and MASSVAC 4.0) to determine the loading rate of evacuees. This approach relies on the planner's judgment in calibrating the model parameters z and h.

In 2000, Urbanik [36] developed a *probability distribution* of the trip departure time. He defined the probability distribution of an activity time based on the percentage of the population that completed the activity within a given time span. To simplify, he assumed that the probability distribution of trip departure sub-activity times were independent. Then, he derived the probability distributions of trip departure time as the join probability distribution of sub-activities involved.

None of the above mentioned approaches consider the impact of human behavior on trip departure time. The work by Murray et al. [27,28] addresses the tendency of households to gather and then evacuate as a single unit. They believe that this type of behavior increases the departure time and, as a consequence, the evacuation time. Their evacuation model is based on a network flow formulation of the problem. In this network, the nodes represent residential and other possible meeting locations. The arcs represent the shortest path between nodes. Two linear integer programming formulations of the problem are given that consider a realistic presentation of human behavior in emergency situations. The first formulation determines the household meeting location while minimizing the maximum travel time of family members. The second formulation determines the route assignment along with the non-drivers' pickup schedule by minimizing the linear trade-off between waiting time and travel time. For a more extensive review of the impact of human behavior in trip departure time, see [20,22,33,40].

Community Evacuation: Trip Destination Selection

In emergency situations, the most straightforward approach that evacuees follow in choosing a destination is the shortest evacuation plan (SEP) [38]. Based on SEP, the evacuees seek the closest exit that flows them away from the danger area. In 1996, Yamada [38] presented an emergency evacuation plan for a city using two network flow optimization models. In these models, the residential areas (RA) and places of refuge (PR) are the nodes of the network, and the roads between them are the arcs. Yamada assumes that the roads are bi-directional with the same travel time in both directions and that the evacuees traverse roads on foot at

a uniform speed. He introduces a dummy node, v^+. v^+ is then connected to each RA node. The new network is denoted by $G^+(V^+, E^+)$, and $V^+ = V \cup \{v^+\}$, $E^+ = E \cup \{(v^+, d)|d \in D\}$. V is the original set of nodes (RA and PR nodes), E is the original set of arcs and D is the set of RA nodes only.

The first model does not consider node and arc capacities. The model minimizes the individual and total travel distance. Yamada applied the Dijkstra algorithm with v^+ as the source node and PR as the demand nodes. This algorithm runs in $O(|V^+|^2)$. The optimal solution for this network flow problem is a forest of trees with exactly one PR in each tree. The solution determines the best possible destination for each RA.

The second model considers capacity constraints on PR nodes. Yamada uses a minimum cost network flow formulation to model this problem. He modified the original graph by adding a dummy source node (v_*), a dummy sink node (v^*), and a set of arcs connecting (v_*) to RA and v^* to PR nodes. The new network is denoted by $G^*(V^*, E^*)$, where $V^* = V \cup \{v_*, v^*\}$ and $E^* = E \cup \{(v_*, r)|r \in R\} \cup \{(d, v^*)|d \in D\}$.

The following is the problem formulation:

$$\text{minimize} \sum_{(u,v)\in E} k(u, v)x(u, v)$$

subject to

$$\sum_{u\in V^*} x(u, v) = \sum_{w\in V^*} x(v, w) \quad \forall v \in V$$

$$\sum_{d\in D} x(d, v^*) = P$$

$$0 \le x(u, v) \le c(u, v) \quad \forall (u, v) \in E^*,$$

where $k(u, v)$, $c(u, v)$, and $x(u, v)$ are the cost coefficient, the capacity, and the number of evacuees traversing arc (u, v), respectively. P is the size of the population, and R is the set of RA nodes. Due to the capacity constraints, the solution may not be a forest of trees and, as a result, evacuees of an RA node may be assigned to multiple PR nodes that may not necessary be the closest.

The SEP minimizes the total travel distance by routing evacuees to the closest exit. This approach causes congestion in certain exits, which in turn increases the total evacuation time. Cova and Johnson [10] overcame this difficulty by developing an optimal lane-based evacuation routing plan. They formulated the problem as an integer extension of the minimum cost network flow problem. The objective, again, is to minimize the total travel distance. However, the model generates routing plans that trade total vehicle travel distance against merging conflicts while preventing traffic-crossing conflicts at intersections. They use a microscopic traffic simulation to compare the relative efficiency of the plans. The model is then used to identify evacuation routing plans for Salt Lake City, Utah.

The selection of a specific destination limits the route choices of evacuees and increases congestion of the roads that lead to safety. To avoid congestion, Hobeika et al. [17] have developed a model that routes the evacuees to the outside boundary of the risk area and lets them seek a safe place afterwards. They have extended the traffic network by adding dummy links that connect the final destinations to the network at the boundary areas. The dummy links have infinite capacity and short travel time. The objective is to minimize the total evacuation time.

Similarly to the trip departure step, human behavior significantly affects the destination choice of evacuees in emergency situations. Evacuees may change their intended destination if they notice considerable traffic backed up ahead of them [35]. In situations when the household members are scattered throughout the evacuation area, the individuals' tendency to meet before evacuating affects the destination selection choice. Murray and Mahmassani [27] point out that depending on the current location of family members, evacuees may decide to meet in a place that is close to the danger rather than far from it.

Community Evacuation: Trip Route Selection

The trip route selection, also known as trip route assignment, identifies the movement of evacuees during the evacuation process. Numerous *optimization*, *simulation* and combinatorial *optimization-simulation* approaches have been used to model route selection procedures in the last four decades. The most common objectives of these models are to minimize the total travel time, to minimize the total evacuation time, or to maximize the flow of evacuees from the risk area to safety [3,10,27,28,38]. The travel time depends on the speed of a vehicle on a highway segment. The average speed is a non-increasing function of the traffic vol-

ume [15]. The following equation demonstrates the relationship between speed and volume and is referred to as the BPR (Bureau of Public Roads) equation.

$$SP_{it} = \frac{SP_i}{[1 + \alpha(\frac{V_{it}}{C_i})^{\beta}]}$$

where, SP_i is the speed limit on segment i, SP_{it} is the average speed of a vehicle on segment i at time t, C_i is the capacity of segment i, V_{it} vehicle flow entering segment i at time t, α and β are constants.

Evacuation time depends not only on the traffic density, but also on traffic delays at intersections. The limited capacity, merging conflicts and crossing conflicts at intersections create unavoidable bottlenecks. The lane-base routing approach, presented by Cova and Johnson [10] in 2003, speeds up the evacuating process by increasing the intersection capacity and alleviating conflicts. The formulate the problem as a minimum cost network flow problem that minimizes the total travel distance, considering intersection conflicts, lane changing, and left-hand turns, simultaneously. Murray et al. [27,28] have formulated the evacuation routing as a vehicle routing problem (VRP). Unlike the classic VRP, they assume that vehicles have different capacities and are not located in a single depot but scattered throughout the network. In addition, the objective function minimizes not only the total travel time, but also the waiting time of evacuees at the meeting locations.

Besides *optimization* approaches, *simulation* based approaches have also been used to model evacuation routing. The employed route selection logic in simulation models might be simple, static or dynamic [35]. In a simple routing approach, the drivers either select the least congested route based on their myopic perception or follow some pre-determined set of routes. This approach has been used in microscopic simulation models such as CLEAR and NETSIM to simulate evacuation routing in small urban and rural areas [26,30].

The static route assignment models assume that traffic conditions remain unchanged during the simulation period. A mesoscopic simulation package, DYNEV, developed by KLD Associates Inc., uses such models to create evacuation routing plans for large urban areas [21].

Considering the dynamic nature of emergency evacuation, the dynamic traffic route assignment models are superior to simple and static approaches [16,17,31,32,36]. To route the evacuees to safety, the dynamic routing approach does not follow a pre-determined set of turning movements at intersections; instead, turning movements are function of dynamic traffic flow and evacuee behavioral considerations. These behavioral considerations address a driver's prior knowledge of the best direction leading to safety and her/his myopic perception of traffic conditions.

In combinatorial *optimization-simulation* approaches, an optimal route assignment model is integrated with a traffic simulation model. MASSVAC [17,29] and Dynasmart-P [4] are examples of macroscopic simulation packages that rely on combinatorial approaches. The objective of the optimization model in MASSVAC is to minimize the number of casualties. This model generates an optimal set of routes along with an optimal evacuation schedule. Dynasmart-P is a dynamic traffic network analysis and evaluation tool that determines a time-dependent assignment of vehicles to different network paths. Thus, the assignment of a driver to a path is made not only based on the length of the path, but also evacuation time. The objective is to minimize the travel time for each individual traveler. A set of outflow constraints limits the total number of vehicles leaving the link at an intersection approach. Additionally, a set of inflow constraints limits the maximum number of vehicles allowed to enter a link from all approaches [19].

Building Evacuation

The issues discussed above are applicable in many emergency scenarios. However, the inevitable differences of some cases demand special considerations. Evacuating a building due to a disaster, such as the threat of smoke, fire, earthquake, bomb or toxic gas leak, requires a different approach. Lindell and Prater [24] have identified major differences between building and community evacuation. First, the social units within a building are not as clear as residential units within a community. Second, the employers can exercise more control strategies than public agencies. Finally, the departure time for building evacuation is shorter because of limited required preparation activities.

A number of studies have been devoted to describing the behavior of building inhabitants in emergency situations. For an extensive review, see Bryan [1]. However, fewer studies have been conducted to model evacuation procedures. One of the earliest attempts by Chalmet et al. [3] uses a capacitated network flow problem as the basis for modeling a building. Workplaces, halls, stairwells, and elevators represent the nodes of the network, and movement paths between them represent the arcs. The static capacity of nodes is the maximum number of individuals who are allowed to be in building components simultaneously. And the dynamic capacity of arcs is an upper bound on the number of individuals who can traverse the pathways in each time interval.

Chalmet et al. have used three different optimization models to solve the problem: a dynamic model, a graphical model and an intermediate model. The dynamic model is a multi objective optimization model that represents the evacuation as it evolves over time. In contrast, the graphical and intermediate models are not time dependent; they treat time as a parameter. They are simpler than the dynamic model; however, they provide almost the same insight about the building evacuation process.

The objectives of the dynamic model are to minimize the average number of time periods spent by each individual to evacuate the building, to maximize the total number of people saved, and to minimize the total evacuation time. The dynamic model is formulated as a minimum cost flow problem and efficiently solved using the GNET Algorithm [2].

The graphical approach to model building evacuation was originally presented by Francis [13,14]. The model assigns people to evacuation routes with the objective of minimizing the evacuation time. Two implicit assumptions of this model are that all evacuees have a uniform accessibility to the exit routes and that route clearance time depends on the number of people using the route. Given k to be the number of individuals in a building that has n exit routes, the formulation is

$$\text{minimize } \max[t_j(x_j) | 1 \leq j \leq n]$$

subject to

$$x_1 + \ldots + x_n = k$$
$$x_1, \ldots, x_n \geq 0$$

where $t_j(x_j)$ is the time required to clear route j if the total number of evacuees on this route is x_j. t_j is a continuous function and is strictly increasing with respect to x_j. Note that $t_j(0) = 0$. Considering the assumptions made by this model, the minimum evacuation time happens when all routes are cleared in the same time.

The intermediate model uses the same network structure as the dynamic model. However, it is superior to the dynamic model in view of the required input data and computational time. Similar to the dynamic model, the arcs are capacitated, but there is no traverse time on arcs. For a given subset A of arcs, called *critical arcs*, there is no capacity constraint; instead, the function $t_{ij}(x_{ij})$ estimates the time it takes to traverse arc $(i, j) \in A$ when the flow of evacuees on this arc is x_{ij}. Clearly, $t_{ij}(0) = 0$. The objective is to minimize the building evacuation time which is explained as minimizing the traverse time on critical arcs. A heuristic bisection search algorithm and an exact minimax algorithm were used to solve the problem.

Small-Area Evacuation

The standard approach for developing an evacuation plan for regions, buildings, ships, etc starts with determining the evacuation zone around a known hazard and then exploring some important factors that affect the evacuation plan (e. g., population distribution, road capacities and human behavior). To delimit the evacuation zone, a boundary is established around the affected area. In nuclear power plant evacuations, the boundary of the evacuation area is defined to X miles of radius from the plant, and X depends on the type of plant and the type of accident. In building evacuations such a boundary is defined by the shell of the building. However, for some emergency situations, such as urban firestorms or toxic spills on highways, the spatial impact of the hazard is unknown. In these kind of situations, defining the evacuation zone and its boundary can not be done in advance. This is usually the case for small urban and rural areas that could be subject to different hazardous events with uncertain spatial effects. Thus, the focus has been on general planning and mock drills rather than attempting to develop neighborhood specific evacuation plans. Cova and Church [8] were the

first to analyze the potential for evacuation difficulty at the neighborhood scale.

Little is known about small area evacuation as it is nearly impossible to measure accurately during an emergency. But, there has been an interest in looking for those areas that might be difficult to evacuate safely in an emergency. Church and Cova [6] introduce a network-based model to search for small contiguous areas or neighborhoods, within a urban/rural area, that may face difficulties in a sudden evacuation scenario. Their model classifies a neighborhood based on the degree of evacuation difficulty. The evacuation difficulty is measured by the evacuation risk factor which is defined as the number of vehicles per exit road. Church and Cova formulate the problem as a nonlinear network partitioning problem. The objective is to identify a critical neighborhood (critical cluster) that has the highest evacuation risk factor. In their network the nodes represent the households and the arcs represent the road segments connecting them. The demand of each node is estimated by multiplying the number of people per household with the average number of vehicles per person. The problem is transformed to an integer linear program whose objective is to identify an evacuation area that has a risk factor greater than a specific minimum threshold. This problem is solved for each node of the network. As a result, each node is labelled by the risk factor of its corresponding critical neighborhood. Finally, for each node the critical risk value is the highest value related to the critical clusters that this node has been part of. An exact and a heuristic approach are proposed to solve the integer-linear programming problem. Since the exact approach is time consuming, the heuristic approach is used to find a contiguous critical area around a given node. The heuristic approach follows a region growing basis. The base node is selected arbitrarily from the network. The area around the node is expanded iteratively by selecting a node randomly from a list of candidate nodes within a specific distance from the base node that most improves the objective function.

Cove and Church [8] have applied a similar methodology to generate an evacuation vulnerability map which classifies a local area based on the evacuation difficulty.

Real-Time Traffic Management

Real-time traffic management for emergency evacuation dynamically controls the traffic flow to achieve certain system objectives such as maximum utilization of transportation system and minimum fatalities and property losses. This approach considers the evolution of the traffic flow in a traffic network to generate a real-time feedback traffic management system by using in-vehicle and on-route surveillance systems [5,25]. Briefly stated, the current condition of dynamic traffic flow is *monitored* by surveillance systems and a reference model that generates the desired traffic status and the "safest evacuation strategy" is developed to satisfy the designated objectives. The objectives are defined based on the nature of the hazard and the involvement of the emergency authorities. Possible objectives are minimizing the total travel time, minimizing the network clearance time or minimizing the number of casualties. The generated real-time *control* strategies include routing assignments, split rates at intersections, or traffic control advisories that are passed on to evacuees cyclically. In fact, the control strategies are not necessarily practiced by all evacuees in emergency situations. Therefore, these strategies are *modified* based on the differences between the current traffic status and the desired traffic status defined by the reference model. The "monitor, control, and modify" framework is repeated frequently in a closed feedback loop to decrease discrepancies between the the original plan and the current traffic status [25]. Some evacuation models include aspects of human behavior to provide more realistic control strategies that alleviate the deviations. The evacuation route choice model developed by Chiu et al. [5] is an example. This model replicates the route-selection procedure of evacuees when they are provided with safe evacuation routes. The probability that an evacuee will select a particular route depends on his familiarity with the route, the degree of overlap between the routes and his preference of using freeways.

Agent-Based Modeling

Agent-based modeling is also known as individual-oriented modeling. This is an increasingly powerful modeling technique to simulate individual interactions in dynamic routing situations such as emergency evacuations. Agent-based modeling treats the individual

vehicles as intelligent decision-making entities [11]. A model of agents and a model of their environment are two basic components of agent-based modeling [39]. The behavior of an agent and its interaction with other agents is modeled by a set of rules such as accelerating, decelerating, and lane-changing rules. The traffic environment is modeled using a traffic network topology, road category, traffic lights, and traffic signs [12,37].

In emergency evacuations, the agent-based simulation captures the collective behavior of agents, which greatly affects the evacuation plan. As a result, more realistic strategies are developed by including the individual behavior of evacuees and their interactions in panic situations. In 1993 Sinuany-Stern and Stern [34] used agent-based simulation for spontaneous urban evacuation. They examined the sensitivity of network clearance time to several traffic factors (such as interaction with pedestrians, intersection traversing time, and car ownership), and route choice mechanisms (such as shortest path selection or myopic-based selection). Cova and Johnson [9] assessed the spatial affect of a proposed second access road on household evacuation time using an agent-based microsimulation model. Church and Sexton [7] used Paramics, an agent-based microsimulation software, to simulate evacuation scenarios in a small neighborhood. They estimated the impact of different evacuation scenarios, such as opening an alternative exit, invoking traffic control plans, and changing the number of vehicles leaving a household, on evacuation time.

Conclusions

Emergency evacuation is a management strategy to ensure population safety in emergency situations. Communities, buildings, and residential areas are prone to disasters, thus detailed evacuation planing is necessary. Evacuation planning models consist of a five-step procedure that involves trip generation, trip departure time, trip destination, trip route selection, and evacuation plan set-up and analysis. We have presented here a summary of some noteworthy research on each of the above mentioned steps of the planning process. This review also focuses on the special features of community, building and small area evacuation planing. In addition, real-time traffic management and agent-based models are discussed. The real-time traffic management models consider the dynamic nature of traffic flow and generate a real-time feedback traffic management system in emergency situation. The agent-based models provide realistic emergency evacuation strategies by considering the individual behavior of evacuees (agents) and their interactions.

References

1. Baker EJ (1979) Predicting Response to Hurricane Warnings: A Reanalysis from Four Studies. Int J Mass Emerg Disast 4:9–24
2. Bradley GH, Brown GG, Graves GW (1977) Design and Implementation of Large Scale Primal Transshipment Algorithms. Management Sci 24(1):1–34
3. Chalmet LG, Francis RL, Saunders PB (1982) Network Models for Building Evacuation. Management Sci 28(1):86–105
4. Chiu YC (2004) Traffic Scheduling Simulation and Assignment for Area-Wide Evacuation. Proc IEEE Intell Transp Syst Conference, pp 537–542
5. Chiu YC, Korada P, Mirchandani PB (2005) Dynamic Traffic Management for Evacuation. The 84th Annual Meeting of Transportation Research Board, Washington
6. Church RL, Cova TJ (2000) Mapping Evacuation Risk on Transportation Networks Using a Spatial Optimization Model. Transp Res Part C: Emerg Technol 8(1):321–336
7. Church RL, Sexton RM (2002) Modeling Small Area Evacuation: Can Existing Transportation Infrastructure Impede Public Safety? Tech Rep, Vehicle Intelligence & Transportation Analysis Laboratory. University of California at Santa Barbara, http://www.ncgia.ucsb.edu/vital/research/pubs/200204-Evacuation.pdf
8. Cova TJ, Church RL (1997) Modeling Community Evacuation Vulnerability Using GIS. Int J Geograph Inform Sci 11(8):763–784
9. Cova TJ, Johnson JP (2002) Microsimulation of Neighborhood Evacuations in the Urban-Wildland Interface. Environ Plann 34:2211–2229
10. Cova TJ, Johnson JP (2003) A Network Flow Model for Lane-based Evacuation Routing. Transport Res Part A: Policy Practice 37(7):579–604
11. Deadman PJ (1999) Modeling Individual Behavior and Group Performance in an Intelligent Agent-Based Simulation of the Tragedy of the Commons. J Environ Manage 56(3):159–172
12. Dia H, Purchase H (1999) Modelling the Impacts of Advanced Traveler Information Systems Using Intelligent Agents. Road Transport Res J 8(3):68–73
13. Francis RL (1979) A Simple Graphical Procedure to Estimate the Minimum Time to Evacuate a Building. Tech Rep. Society of Fire Protection Engineers, Boston, MA

14. Francis RL (1981) A Uniformity Principle for Evacuation Route Allocation. J Res Nat Bureau Standard 86(5):509–513
15. Garber NJ, Hoel LA (2001) Traffic & Highway Engineering, 3rd edn. Thomson Learning, Pacific Grove
16. Hobeika AG, Jamie B (1985) MASSVAC: A Model for Calculating Evacuation Times under Natural Disaster. In: Carroll JM (ed) Proceedings of the Conference on Computer Simulation in Emergency Planning, vol 15. Soc Comput Simul, pp 23–28
17. Hobeika AG, Kim C (1998) Comparison of Traffic Assignments in Evacuation Modeling. IEEE Trans Eng Managem 45(2):192–198
18. Jamei B (1984) Transportation Actions to Reduce Highway Evacuation Times under Natural Disasters Ph.D Thesis, Virginia Polytechnic Institute and State University
19. Jayakrishnan R, Mahmassani HS, Hu TY (1994) An Evaluation Tool For Advanced Traffic Information and Management System in Urban Networks. Transp Res Part C: Emerging Technol 2(3):129–147
20. Johnson NR (1988) Fire in a Crowded Theatre: A Descriptive Investigation of the Emergence of Panic. Int J Mass Emerg Disasters 6(1):7–26
21. KLD Associates Inc. (1984) DYNEV: Traffic Simulation Models Prepared for the Federal Emergency Management Agency, http://www.kldassociates.com/st_.htm
22. Lindell MK, Lu J-C, Prater CS (2005) Household Decision Making and Evacuation in Response to Hurricane Lili. Nat Hazards Rev 6(4):171–179
23. Lindell MK, Perry RW (1992) Behavioral Foundation of Community Emergency Planning. Hemisphere Press, Washington, DC
24. Lindell MK, Prater CS (2005) Estimating Evacuation Time Components: Lessons from Nuclear Power Plants, Hurricanes, and FirstWorld Trade Center Bombing. In: Peacock RD, Kuligowski ED (eds) NIST Special Publication 1032. Workshop on Building Occupant Movement During Fire Emergencies. pp 91–95.S
25. Liu HX, Ban JX, Ma W, Mirchandani P (2007) Model Reference Adaptive Control Framework for Realtime Traffic Management under Emergency Evacuation. ASCE J Urban Plann Developm 133(1):43–50
26. Moeller M, Urbanik T, Desrosiers A (1981) CLEAR (Calculated Logical Evacuation and Response): A Generic Transportation Network Model for the Calculation of Evacuation Time Estimates. Prepared for the US Nuclear Regulatory Commission by Pacific Northwest Laboratory, NUREGICR-2504
27. Murray PM, Mahmassani HS (2003) Model of Household Trip Chain Sequencing in an Emergency Evacuation. Transp Res Rec 1831:21–29
28. Murray PM, Pamela M, Mahmassani HS (2004) Transportation Network Evacuation Planning with Household Activity Interactions. Transp Res Rec 1894:150–159
29. Radwan AE, Hobeika AG, Sivasailam D (1985) A Computer Simulation Model for Rural Network Evacuation under Natural Disasters. Inst Transp Eng J 55(9):25–30
30. Rathi A, Santiago A (1990) The new NETSIM Simulation Program. Traffic Eng Control 31(5):317–320
31. Sheffi Y, Mahmassani HS, Powell W (1981) Evacuation Studies at Nuclear Power Plants: A New Challenge for Transportation Engineers. Inst Transport Eng J 51(6):25–28
32. Sheffi Y, Mahmassani H, Powell WB (1982) A Transportation Network Evacuation Model. Transport Res Part A: Policy Practice 16(3):209–218
33. Sime JD (1995) Crowd Psychology and Engineering. Safety Sci 21(1):1–14
34. Sinuany-Stern Z, Stern E (1993) Simulating the Evacuation of a Small City: The Effects of Traffic Factors. Soc-Econ Plann Sci 27(2):97–108
35. Southworth F (1991) Regional Evacuation Modeling: A State-of-the-art Review. Center for Transportation Analysis, Oak Ridge National Laboratory, Oak Ridge
36. Urbanik T (2000) Evacuation Time Estimate for Nuclear Power Plant. J Hazard Mater 75(2):165–180
37. Wahle J, Neubert L, Esser J, Scherckenberg M (2001) A Cellular Automaton Traffic Flow Model for Online Simulation of Traffic. Parall Comput 27(5):719–735
38. Yamada T (1996) A Network Flow Approach to a City Emergency Evacuation Planning. Int J Syst Sci 27(10):931–936
39. Zeigler BP (1976) Theory of Modeling and Simulation. Wiley, New York
40. Zelinsky W, Kosinski LA (1991) The Emergency Evacuation of Cities. Rowman & Littlefield Publishers, Savage, MD

Entropy Optimization: Interior Point Methods

Interior Point Algorithms for Entropy Optimization

SHU-CHERNG FANG[1], JACOB H.-S. TSAO[2]
[1] North Carolina State University, Raleigh, USA
[2] San Jose State University, San Jose, USA

MSC2000: 94A17, 90C51, 90C25

Article Outline

Keywords
See also
References

Keywords

Entropy optimization; Interior point methods; Primal-dual algorithm; Polynomial time convergence

This section introduces the interior point approach to solving entropy optimization problems with linear constraints. In particular, we consider the following problem:

Program EL:

$$\begin{cases} \min & f(\mathbf{x}) \equiv \mathbf{c}^\top \mathbf{x} + \sum_{j=1}^{n} d_j x_j \ln x_j \\ \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{O}, \end{cases} \tag{1}$$

where $\mathbf{c} \in \mathbf{R}^n$, $\mathbf{d} \in \mathbf{R}^n$, $\mathbf{d} > \mathbf{O}$, $\mathbf{b} \in \mathbf{R}^m$, $\mathbf{A}$ is an $(m \times n)$-matrix, $\mathbf{O}$ is an n-dimensional zero vector, and $0 \ln 0 \equiv 0$. When $\mathbf{c} = \mathbf{O}$ and $d_j = 1$, $j = 1, \ldots, n$, Program EL becomes a pure entropy optimization problem.

Denote the feasible region of Program EL by $F_p \equiv \{\mathbf{x} \in \mathbf{R}^n: \mathbf{A}\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{O}\}$ and the (relative) interior of F_p by $F_p^0 \equiv \{\mathbf{x} \in \mathbf{R}^n: \mathbf{A}\,\mathbf{x} = \mathbf{b}; \mathbf{x} > \mathbf{O}\}$. An n-vector $\mathbf{x}$ is called an *interior solution* of Program EL if $\mathbf{x} \in F_p^0$. With these definitions, we have the following verifiable result:

Lemma 1 *If F_p is nonempty, then Program EL has a unique optimal solution. Moreover, if F_p has a nonempty interior, then the unique optimal solution is strictly positive.*

All *interior point methods*, including those to be discussed in this section, require the fundamental assumption that F_p has a nonempty interior, i. e., $F_p^0 \neq \emptyset$. A Lagrangian dual can be derived in the following manner. For all $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{y} \in \mathbf{R}^m$, and $\mathbf{z} \in \mathbf{R}^n_+ \equiv \mathbf{x}: \mathbf{x} \in \mathbf{R}^n, \mathbf{x} \geq \mathbf{O}\}$, define the following Lagrangian function:

$$L(\mathbf{x}, \mathbf{y}, \mathbf{z}) \equiv \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} d_j e(x_j) - \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j - b_i \right) y_i - \sum_{j=1}^{n} z_j x_j, \tag{2}$$

where

$$e(x) \equiv \begin{cases} x \ln x & \text{if} \quad x \geq 0, \\ \infty & \text{if} \quad x < 0, \end{cases}$$

is a proper convex function with the set $\{x: x \in \mathbf{R}, x \geq 0\}$ being its effective domain [6]. The concept of proper convex function has often been used to simplify convex analysis. For details about the theory of using Lagrange multipliers for solving constrained optimization problems defined in terms of proper convex functions, the reader is referred to [6, Chap. 28].

Rearranging terms in (2) results in

$$L(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} d_j e(x_j) + \sum_{i=1}^{m} b_i y_i - \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i + z_j \right) x_j.$$

Considering the fact that $d_j > 0$ and the shape of the entropic function $x \ln x$, we know that, for any given $\mathbf{y} \in \mathbf{R}^m$ and $\mathbf{z} \in \mathbf{R}^n_+$, $L(\mathbf{x}, \mathbf{y}, \mathbf{z})$ achieves its unique minimum at $\mathbf{x}^* > \mathbf{O}$. Also, its first derivative at $\mathbf{x}^*$ vanishes. This implies

$$d_j \ln x_j^* - \sum_{i=1}^{m} a_{ij} y_i + c_j + d_j = z_j \geq 0. \tag{3}$$

Multiplying both sides of (3) by x_j^* and summing over j produces

$$\sum_{j=1}^{n} c_j x_j^* + \sum_{j=1}^{n} d_j x_j^* \ln x_j^* - \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i + z_j \right) x_j^* = -\sum_{j=1}^{n} d_j x_j^*.$$

Consequently, for any $\mathbf{y} \in \mathbf{R}^m$ and $\mathbf{z} \in \mathbf{R}^n_+$,

$$L(\mathbf{x}^*, \mathbf{y}, \mathbf{z}) = \sum_{i=1}^{m} b_i y_i - \sum_{j=1}^{n} d_j x_j^*,$$

where $\mathbf{x}^*$ satisfies (3). Therefore, a Lagrangian *dual* of Program EL becomes

$$\begin{cases} \max\limits_{\substack{\mathbf{y} \in \mathbb{R}^m \\ \mathbf{z} \in \mathbb{R}^n_+}} & L(\mathbf{y}, \mathbf{z}) \equiv \sum_{i=1}^{m} b_i y_i - \sum_{j=1}^{n} d_j x_j^* \\ \text{s.t.} & d_j \ln x_j^* - \sum_{i=1}^{m} a_{ij} y_i + c_j + d_j = z_j, \\ & j = 1, \ldots, n. \end{cases}$$

This dual is equivalent to

Program DEL:

$$\begin{cases} \max\limits_{\substack{\mathbf{y}\in\mathbb{R}^m \\ \mathbf{O}<\mathbf{x}\in\mathbb{R}^n}} & L(\mathbf{x},\mathbf{y}) \equiv \sum\limits_{i=1}^{m} b_i y_i - \sum\limits_{j=1}^{n} d_j x_j \\ \text{s.t.} & d_j \ln x_j + c_j + d_j - \sum\limits_{i=1}^{m} a_{ij} y_i \geq 0, \\ & \qquad j = 1, \ldots, n. \end{cases} \tag{4}$$

Note that $\mathbf{x}$ is strictly positive because ln 0 is not well-defined. However, if we define ln 0 = $-\infty$, the domain of $\mathbf{x}$ in Program DEL can be replaced by $\{\mathbf{x}:\mathbf{x} \in \mathbf{R}^n, \mathbf{x} \geq \mathbf{O}\}$. Denote the excess vector $\nabla f(\mathbf{x}) - \mathbf{A}^\intercal \mathbf{y}$ by $\mathbf{s}$. The jth component of $\mathbf{s}$ is simply $d_j \ln x_j + c_j + d_j - \sum_{i=1}^{m} a_{ij} y_i$, which is the left-hand side of (4). Denote the feasible region of Program DEL by $F_d \equiv \{(\mathbf{x}, \mathbf{y}): \nabla f(\mathbf{x}^*) - \mathbf{A}^\intercal \mathbf{y}^* \geq \mathbf{O}\}$, and assume that F_d has a nonempty interior.

We now derive the Karush–Kuhn–Tucker conditions for Program DEL. First, define, for all $\mathbf{u} \geq \mathbf{O}$, the following Lagrangian:

$$L'(\mathbf{x},\mathbf{y},\mathbf{u}) \equiv \sum_{i=1}^{m} b_i y_i - \sum_{j=1}^{n} d_j x_j + \sum_{j=1}^{n} u_j \left(d_j \ln x_j + c_j + d_j - \sum_{i=1}^{m} a_{ij} y_i \right).$$

Setting the partial derivatives with respect to y_i and x_j to zero gives

$$\begin{aligned} & b_i - \sum_{j=1}^{n} a_{ij} u_j = 0, \quad i = 1, \ldots, m, \\ & -d_j + \frac{u_j d_j}{x_j} = 0, \quad j = 1, \ldots, n. \end{aligned} \tag{5}$$

Note that (5) is equivalent to $u_j = x_j$. Therefore, the KKT conditions for Program DEL become

a) There exists $\mathbf{x} \in \mathbf{R}^n$ such that $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{x} \geq \mathbf{O}$. This can be viewed as the 'primal feasibility condition'.
b) There exists $\mathbf{y} \in \mathbf{R}^m$ such that, together with $\mathbf{x}$, $d_j \ln x_j + c_j + d_j - \sum_{i=1}^{m} a_{ij} y_i \geq 0$ or $\nabla f(\mathbf{x}) - \mathbf{A}^\intercal \mathbf{y} \geq \mathbf{O}$. Similarly, this can be viewed as the 'dual feasibility condition'.
c) For all $j = 1, \ldots, n$, $(d_j \ln x_j + c_j + d_j - \sum_{i=1}^{m} a_{ij} y_i) x_j = 0$. This can be viewed as the 'complementary slackness condition'.

Note that, by (5), the Lagrange multipliers associated with the constraints of Program DEL at its optimal solution happen to coincide with the $\mathbf{x}$-component of the optimal solution of Program DEL. This, together with the fact that the dual of Program DEL is Program EL, imply that the optimal solution of Program DEL contains the optimal solution of Program EL.

Also note that an alternative *dual* program can be defined by considering the following Lagrangian:

$$L''(\mathbf{x},\mathbf{y}) \equiv \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} d_j x_j \ln x_j - \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j - b_i \right) y_i,$$

for $\mathbf{x} \geq \mathbf{O}$ and $\mathbf{y} \in \mathbf{R}^m$. In this expression, no Lagrange multipliers are defined for the constraints $\mathbf{x} \geq \mathbf{O}$, and it leads to the following dual program:

$$\max_{\mathbf{y}\in\mathbb{R}^m} \sum_{i=1}^{m} b_i y_i - \sum_{j=1}^{n} d_j \exp\left\{ \frac{\sum_{i=1}^{m} a_{ij} y_i - c_j}{d_j} - 1 \right\}.$$

Since this dual program is *unconstrained*, any solution algorithm can be viewed as an interior point algorithm. For details about this approach and companion efficient solution *algorithms*, see [2].

In the rest of this section, we focus on the development of a *primal-dual* interior point algorithm [5]. Note that, to obtain the algorithm, Program DEL, rather than the unconstrained dual program, was used in [5]. The primal-dual interior point algorithm starts with an initial primal feasible solution $\mathbf{x}^0$ and an initial dual feasible solution $\mathbf{y}^0$. While the algorithm iterates, it maintains the primal and dual feasibility conditions and reduces the complementary slackness. In other words, the algorithm iterates from a pair of interior solutions $(\mathbf{x}^k, \mathbf{y}^k)$, with $\mathbf{A}\,\mathbf{x}^k = \mathbf{b}$, $\mathbf{x}^k > \mathbf{O}$ and $\mathbf{s}^k = \nabla f(\mathbf{x}^k) - \mathbf{A}^\intercal \mathbf{y}^k > \mathbf{O}$, to a new interior solution pair $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1})$ such that the complementary slackness is reduced from $\delta_k \equiv (\mathbf{x}^k)^\intercal \mathbf{s}^k$ to $\delta_{k+1} \equiv (\mathbf{x}^{k+1})^\intercal \mathbf{s}^{k+1}$. The algorithm terminates when $\delta_k \leq \epsilon$, for some given $\epsilon > 0$ (or when the difference between $f(\mathbf{x}^k)$ and the optimum is sufficiently small).

To describe the algorithm, we use the boldface upper-case letters **X**, **S**, and **W** to denote the diagonal matrices formed by the components of vectors **x**, **s**, and **w**, respectively. We also denote the vectors of all ones of appropriate dimensions by **e**, the l_2 norm by $\|\cdot\|$, and the vector whose components are $\ln(x_j)$'s, $j = 1, \ldots, n$, by $\ln \mathbf{x}$.

Rather than dealing with the complementary slackness δ_k directly, the following *primal-dual potential function* [8]

$$\psi(\mathbf{x}, \mathbf{s}) = \rho \ln(\mathbf{x}^\top \mathbf{s}) - \sum_{j=1}^{n} \ln(x_j s_j),$$

where $\rho \geq n + \sqrt{n}$, can be used as a surrogate measure [5].

Given the initial solution pair, the potential of the associated complementary slackness can be calculated. Given the inaccuracy tolerance ϵ, a target potential can be calculated. Therefore, the amount of required potential reduction can be calculated. The primal-dual interior point algorithm, under proper conditions, will reduce the potential by a constant amount in each iteration.

Note that two different pairs of $(\mathbf{x}, \mathbf{s})$ that have the same complementary slackness measure may have different potentials. Therefore, to ensure that the target potential is sufficiently small, we need to find the minimum potential among all those $(\mathbf{x}, \mathbf{s})$ pairs such that $\mathbf{x}^\top \mathbf{s} = \epsilon$, or a lower bound of this minimum potential.

Rewrite the potential function as

$$\psi(\mathbf{x}, \mathbf{s}) = (\rho - n) \ln(\mathbf{x}^\top \mathbf{s}) - \sum_{j=1}^{n} \ln\left(\frac{x_j s_j}{\mathbf{x}^\top \mathbf{s}}\right).$$

Applying the geometric-arithmetic inequality results in

$$\prod_{j=1}^{n} \left(\frac{x_j s_j}{\mathbf{x}^\top \mathbf{s}}\right)^{\frac{1}{n}} \leq \frac{1}{n} \sum_{j=1}^{n} \left(\frac{x_j s_j}{\mathbf{x}^\top \mathbf{s}}\right) = \frac{1}{n}.$$

Taking the natural logarithm leads to

$$\frac{1}{n} \sum_{j=1}^{n} \ln\left(\frac{x_j s_j}{\mathbf{x}^\top \mathbf{s}}\right) \leq \ln\left(\frac{1}{n}\right) = -\ln n.$$

Consequently,

$$\sum_{j=1}^{n} \ln\left(\frac{x_j s_j}{\mathbf{x}^\top \mathbf{s}}\right) \leq -n \ln n.$$

Therefore, the target potential should be $(\rho - n) \ln \epsilon + n \ln n$. Given the potential associated with the initial solution, the exact amount of potential reduction is $\psi(\mathbf{x}^0, \mathbf{s}^0) - (\rho - n) \ln \epsilon - n \ln n$. Note that for a given inaccuracy tolerance ϵ, the target potential is indeed the minimum of all the potentials associated with all $(\mathbf{x}, \mathbf{s})$ pairs such that $\mathbf{x}^\top \mathbf{s} = \epsilon$. This is indicated by the tight geometric-arithmetic inequality.

Given the knowledge of how much potential reduction needs to be, if an algorithm reduces the potential by a constant amount in each iteration, then the complexity of the algorithm is $O(\psi(\mathbf{x}^0, \mathbf{s}^0) - (\rho - n) \ln \epsilon - n \ln n)$.

Assume that, in iteration k, we have a primal-dual feasible solution pair $(\mathbf{x}^k, \mathbf{y}^k)$ and the slack vector $\mathbf{s}^k \equiv \nabla f(\mathbf{x}^k) - \mathbf{A}^\top \mathbf{y}^k > \mathbf{O}$. Ideally, one would like to find $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1})$ such that the KKT conditions are met, i. e.,

$$\begin{aligned}
&\mathbf{A}\mathbf{x}^{k+1} = \mathbf{b}, \quad \mathbf{x}^{k+1} \geq \mathbf{O}, \\
&\nabla f(\mathbf{x}^{k+1}) - \mathbf{A}^\top \mathbf{y}^{k+1} \geq \mathbf{O}, \\
&\mathbf{X}^{k+1}(\nabla f(\mathbf{x}^{k+1}) - \mathbf{A}^\top \mathbf{y}^{k+1}) = \mathbf{O}.
\end{aligned}$$

Define

$$\begin{aligned}
\Delta\mathbf{x} &\equiv \mathbf{x}^{k+1} - \mathbf{x}^k, \\
\Delta\mathbf{y} &\equiv \mathbf{y}^{k+1} - \mathbf{y}^k, \\
\Delta\mathbf{s} &\equiv \mathbf{s}^{k+1} - \mathbf{s}^k, \\
\Delta\mathbf{X} &\equiv \mathbf{X}^{k+1} - \mathbf{X}^k.
\end{aligned}$$

With these definitions, the conditions stated above become

$$\begin{aligned}
&\mathbf{A}(\mathbf{x}^k + \Delta\mathbf{x}) = \mathbf{b} \quad , \quad \mathbf{x}^k + \Delta\mathbf{x} \geq \mathbf{O}, \\
&\nabla f(\mathbf{x}^k + \Delta\mathbf{x}) - \mathbf{A}^\top(\mathbf{y}^k + \Delta\mathbf{y}) \geq \mathbf{O}, \\
&(\mathbf{X}^k + \Delta\mathbf{X}) \\
&\times [\nabla f(\mathbf{x}^k + \Delta\mathbf{x}) - \mathbf{A}^\top(\mathbf{y}^k + \Delta\mathbf{y})] = \mathbf{O}.
\end{aligned} \tag{6}$$

Note that quantity in the bracket of (6) is simply $\mathbf{s}^{k+1} = \mathbf{s}^k + \Delta\,\mathbf{s}$, where

$$\Delta\mathbf{s} \equiv \nabla f(\mathbf{x}^k + \Delta\mathbf{x}) - \nabla f(\mathbf{x}^k) - \mathbf{A}^\top \Delta\mathbf{y}.$$

Therefore, we have

$$(\mathbf{X}^k + \Delta\mathbf{X})(\mathbf{s}^k + \Delta\mathbf{s}) = \mathbf{O},$$

or

$$\mathbf{X}^k\mathbf{s}^k + \mathbf{X}^k\Delta\mathbf{s} + \Delta\mathbf{X}\mathbf{s}^k + \Delta\mathbf{X}\Delta\mathbf{s} = \mathbf{O},$$

or

$$\mathbf{X}^k\Delta\mathbf{s} + \mathbf{S}^k\Delta\mathbf{x} = -\Delta\mathbf{X}\Delta\mathbf{s} - \mathbf{X}^k\mathbf{s}^k.$$

Solving the equations

$$\mathbf{A}(\mathbf{x}^k + \Delta\mathbf{x}) = \mathbf{b},$$
$$\mathbf{X}^k\Delta\mathbf{s} + \mathbf{S}^k\Delta\mathbf{x} = -\Delta\mathbf{X}\Delta\mathbf{s} - \mathbf{X}^k\mathbf{s}^k,$$

subject to the condition

$$\nabla f(\mathbf{x}^k + \Delta\mathbf{x}) - \mathbf{A}^\top(\mathbf{y}^k + \Delta\mathbf{y}) \geq \mathbf{O}$$

is in general difficult.

Given $\mathbf{O} < \mathbf{x}^k \in F_p$, $\mathbf{s}^k = \nabla f(\mathbf{x}^k) - \mathbf{A}^\top\mathbf{y}^k > 0$, and $\delta^k = (\mathbf{x}^k)^\top\mathbf{s}^k$, the algorithm proposed in [4] solves the following system of nonlinear equations for $\Delta\mathbf{x}$ and $\Delta\mathbf{y}$:

$$\mathbf{X}^k\Delta\mathbf{s} + \mathbf{S}^k\Delta\mathbf{x} = \theta\mathbf{p}^k, \tag{7}$$

$$\mathbf{A}\Delta\mathbf{x} = \mathbf{O}, \tag{8}$$

where $\theta > 0$ is a constant to be specified later and

$$\mathbf{p}^k \equiv \frac{\delta^k}{\rho}\mathbf{e} - \mathbf{X}^k\mathbf{S}^k\mathbf{e},$$

and $n + \sqrt{n} \leq \rho < 2n$.

By choosing

$$\theta = \frac{\beta \min_j(\sqrt{x_j^k s_j^k})}{\left\|(\mathbf{X}^k\mathbf{S}^k)^{-0.5}\mathbf{p}^k\right\|}$$

for some $0 < \beta < 1$ yet to be determined, we obtain

$$\psi(\mathbf{x}^{k+1}, \mathbf{s}^{k+1}) \leq \psi(\mathbf{x}^k, \mathbf{s}^k) - \gamma$$

for a constant $\gamma > 0$. Let $C > 0$ be a real number. Choose β such that

$$0 < \beta < 1, \tag{9}$$

$$\beta(1 + C\beta) \leq \frac{1}{2}, \qquad 1 - C\beta \geq 0. \tag{10}$$

It can be shown [5] that, to reduce the potential by a constant amount in each iteration, solving a linear approximation of equations (7) and (8) can achieve the required accuracy.

Suppose that $n + \sqrt{n} \leq \rho < 2n$ and that $\Delta\,\mathbf{x}$ and $\Delta\,\mathbf{y}$ satisfy

$$\mathbf{A}\Delta\mathbf{x} = \mathbf{O}, \qquad \mathbf{X}^k\Delta\mathbf{s} + \mathbf{S}^k\Delta\mathbf{x} = \theta\mathbf{p}^k + \mathbf{z}^k,$$
$$\left\|\mathbf{z}^k\right\| \leq C\beta^2 \min(x_j^k s_j^k), \tag{11}$$

then

$$\psi(\mathbf{x}^k, \mathbf{s}^k) - \psi(\mathbf{x}^{k+1}, \mathbf{s}^{k+1}) > \gamma,$$

where $\gamma = (\frac{\sqrt{3}}{2})\beta(1 - C\beta) - \beta^2(1 + C\beta)^2$.

Condition (11) can be achieved by solving the following set of linear equations:

$$\mathbf{X}^k(\nabla^2 f(\mathbf{x}^k)\Delta\mathbf{x} - \mathbf{A}^\top\Delta\mathbf{y}) + \mathbf{S}^k\Delta x = \theta\mathbf{p}^k, \tag{12}$$

$$\mathbf{A}\Delta\mathbf{x} = \mathbf{O}. \tag{13}$$

Note that the vector $\nabla^2 f(\mathbf{x}^k)\Delta\,\mathbf{x}$ replaces $\nabla f(\mathbf{x}^k + \Delta\,\mathbf{x}) - \nabla f(\mathbf{x}^k)$ of (7) and serves as a simple linear approximation. Equations (12) and (13) are key to the 'potential-reduction' primal-dual interior point algorithm.

Given an initial interior point solution, an interior point algorithm can be stated as follows.

Initialization:
Given an initial primal interior point solution $\mathbf{x}^0$ and an initial dual solution $\mathbf{y}^0$ such that $\mathbf{A}\mathbf{x}^0 = \mathbf{b}$, $\mathbf{x}^0 > 0$, and $\mathbf{s}^0 = \nabla f(\mathbf{x}^0) - \mathbf{A}^\top\mathbf{y}^0 > 0$, calculate $\delta^0 = (\mathbf{x}^0)^\top\mathbf{s}^0$;
set $k \leftarrow 0$.
Iteration:
IF $\delta^k < \epsilon$, THEN STOP
ELSE
solve (12), (13) for $\Delta\mathbf{x}$ and $\Delta\mathbf{y}$;
set
$\mathbf{x}^{k+1} \equiv \mathbf{x}^k + \Delta\mathbf{x}$;
$\mathbf{y}^{k+1} \equiv \mathbf{y}^k + \Delta\mathbf{y}$;
$\mathbf{s}^{k+1} \equiv \nabla f(\mathbf{x}^{k+1}) - \mathbf{A}^\top\mathbf{y}^{k+1}$;
$\delta^{k+1} \equiv (\mathbf{x}^{k+1})^\top\mathbf{s}^{k+1}$;
reset $k \leftarrow k + 1$ for the next iteration.
END IF

With a standard procedure for obtaining an initial solution [1], the following theorem of *polynomial time convergence* was shown in [5].

Theorem 2 *Suppose that $\epsilon > 0$ and $2n \geq \rho \geq n + \sqrt{n}$. Then, in the kth iteration, $\mathbf{x}^k > \mathbf{O}$, $\mathbf{s}^k > \mathbf{O}$, and $\mathbf{x}^k$ and $\mathbf{y}^k$ are feasible for Programs EL and DEL. Moreover, the interior point algorithm terminates in at most $O(\psi(\mathbf{x}^0, \mathbf{s}^0) - (\rho - n)\ln\epsilon - n\ln n)$ iterations.*

It was also suggested that, in practical implementation, the stepsize can be set to $\overline{\eta}$ based on a line search such

that $\overline{\eta} \equiv \arg\min_{\eta \geq 0} \psi(\mathbf{x}^k + \eta \Delta \mathbf{x}, \mathbf{s}^k + \eta \Delta \mathbf{s})$. With this stepsize, one can set $\mathbf{x}^{k+1} \equiv \mathbf{x}^k + \overline{\eta} \Delta \mathbf{x}$, $\mathbf{y}^{k+1} \equiv \mathbf{y}^k + \overline{\eta} \Delta \mathbf{y}$.

The search direction is a combination of a decent direction and a centering direction. To enable local quadratic convergence, a computable criterion was developed under which a pure Newton method for solving $\nabla f(\mathbf{x}) - \mathbf{A}^\top \mathbf{y} = \mathbf{O}$, $\mathbf{A}\mathbf{x} = \mathbf{b}$ (by solving the linear system of $\nabla^2 f(\mathbf{x}^k)\Delta \mathbf{x} - \mathbf{A}^\top \Delta \mathbf{y} = -\mathbf{s}^k$ and $\mathbf{A}\Delta \mathbf{x} = \mathbf{O}$) can be applied for the rest of the search process. Note that when $\mathbf{x}^k$ is close to the optimal solution, we have $\mathbf{x}^k$ being strictly positive, and therefore $\nabla f(\mathbf{x}) - \mathbf{A}^\top \mathbf{y}$ should be close to $\mathbf{O}$. Implementation of primaldual interior point algorithms proposed in [5] is discussed in [3].

In addition to the 'potential-reduction' interior point method described above, the *'path following' interior point method*, which follows an ideal interior trajectory to reach an optimal solution, was proposed in [7,9]. The convergence of the path following interior point method has been established. However, to the best of our knowledge, possible polynomial time convergence behavior remains an open issue.

See also

- ▶ Entropy Optimization: Parameter Estimation
- ▶ Entropy Optimization: Shannon Measure of Entropy and its Properties
- ▶ Homogeneous Selfdual Methods for Linear Programming
- ▶ Interior Point Methods for Semidefinite Programming
- ▶ Jaynes' Maximum Entropy Principle
- ▶ Linear Programming: Interior Point Methods
- ▶ Linear Programming: Karmarkar Projective Algorithm
- ▶ Maximum Entropy Principle: Image Reconstruction
- ▶ Potential Reduction Methods for Linear Programming
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Fang S-C, Puthenpura S (1993) Linear optimization and extensions: theory and algorithms. Prentice-Hall, Englewood Cliffs, NJ
2. Fang S-C, Rajasekera JR, Tsao H-SJ (1997) Entropy optimization and mathematical programming. Kluwer, Dordrecht
3. Han C-G, Pardalos PM, Ye Y (1992) Implementation of interior-point algorithms for some entropy optimization problems. Optim Softw 1:71–80
4. Kortanek KO, Potra F, Ye Y (1991) On some efficient interior point methods for nonlinear convex programming. Linear Alg & Its Appl 152:169–189
5. Potra F, Ye Y (1993) A quadratically convergent polynomial algorithm for solving entropy optimization problems. SIAM J Optim 3:843–860
6. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton
7. Sheu RL, Fang S-C (1994) On the generalized path-following methods for linear programming. Optim 30:235–249
8. Todd MJ, Ye Y (1990) A centered projective algorithm for linear programming. Math Oper Res 15:508–529
9. Zhu J, Ye Y (1990) A path-following algorithm for a class of convex programming problems. Working Paper College of Business Administration, Univ Iowa, no. 90-14

Entropy Optimization: Parameter Estimation

WILLIAM R. ESPOSITO,
CHRISTODOULOS A. FLOUDAS
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 94A17, 62F10

Article Outline

Keywords
Introduction
Entropy Measures
Jaynes' Maximum Entropy
  for Continuous Distributions
MaxEnt Estimation Method
Maximum Likelihood From MaxEnt
Recursive Parameter Estimation
Parameter Estimation and Model Selection
See also
References

Keywords

Maximum entropy; Parameter estimation; Model identification

Introduction

Entropy optimization has been applied to problems in various fields of interest from thermodynamics to financial planning. In this context 'entropy' refers to the amount of uncertainty in a system, rather than the amount of disorder. A detailed definition of entropy can be found in [4].

One area of application, which has not received much attention in recent years, is that of parameter estimation. The estimation of parameters in semi-empirical mathematical models is a process which is important in many disciplines in the sciences and engineering. This article will focus on a few different areas of the parameter estimation problem which have been approached from an entropy perspective. *Jaynes' maximum entropy* principle allows for the estimation of parameters in a statistical distribution function by specification of the characteristic moments. This method can also be used to derive the principle of *maximum likelihood*, one of the most widely used parameter estimation approaches. Entropy principles have also been used to derive theoretical 'best estimators' for recursive parameter estimation schemes. These results can then be used to gauge the performance of various nonoptimal approaches. A final application involves the development of a measure which not only allows for the estimation of model parameters, but also simultaneously choosing the best mathematical form of the model.

Entropy Measures

In order to optimize entropy, one must possess some quantitative measure of the entropy of a given distribution. One such measure was developed by C.E. Shannon [8]. Shannon arrived at the function by postulating a set of properties which the measure should have, and then deriving a form which possesses those properties. For a probability distribution $\mathbf{p} = (p_1, \ldots, p_n)$, the function takes the form of:

$$S = -\sum_{i=1}^{n} p_i \ln p_i. \tag{1}$$

Shannon also proved that this function was unique for the postulated set of properties. Other researchers have postulated different sets of properties, but arrived at the same result [4].

Another measure of entropy, in this case the cross entropy or distance between two distributions, was presented by S. Kullback and R.A. Leibler [5]. For two given distributions, $\mathbf{p} = (p_1, \ldots, p_n)$, and $\mathbf{q} = (q_1, \ldots, q_n)$, the function takes the form:

$$I = \sum_{i=1}^{n} p_i \ln \frac{p_i}{q_i}. \tag{2}$$

It is assumed that when $q_i = 0$, the associated p_i also is zero and $0 \ln \frac{0}{0} \equiv 0$. This function is referred to as the *Kullback–Leibler measure of cross-entropy*.

Jaynes' Maximum Entropy for Continuous Distributions

Since most distributions encountered in practice are continuous in nature, Jaynes' principle of maximum entropy (*MaxEnt*), must first be extended to continuous distributions. This extension is straight forward and results in:

$$\begin{cases} \max & -\int_a^b f(x) \ln f(x)\, dx \\ \text{s.t.} & \int_a^b f(x)\, dx = 1 \\ & \int_a^b f(x) g_r(x)\, dx = a_r, \\ & \quad r = 1, \ldots, m, \end{cases} \tag{3}$$

where $f(x)$ is a continuous probability density function from a to b. The Lagrange function takes the form of:

$$\begin{aligned} L \equiv & -\int_a^b f(x) \ln f(x)\, dx \\ & - (\lambda_0 - 1) \left[\int_a^b f(x) dx - 1 \right] \\ & - \sum_{r=1}^{m} \lambda_r \left[\int_a^b f(x) g_r(x)\, dx - a_r \right]. \end{aligned} \tag{4}$$

Using the Euler–Lagrange equation the following expression results:

$$f(x) = \exp\left[-\lambda_0 - \lambda_1 g_1(x) \cdots - \lambda_m g_m(x) \right]. \tag{5}$$

A detailed discussion can be found in [4].

MaxEnt Estimation Method

The estimation of parameters in a statistical distribution using MaxEnt follows these steps:

1) Specify m characterizing functions, $g_1(x), \ldots, g_m(x)$.
2) Use MaxEnt to find $f(x)$, which is given by (5).
3) Find estimates of the values of the moment equations from the observed data set $\mathbf{x} = \{x_1, \ldots, x_n\}$ through the relationship:

$$\widehat{a}_r = \frac{1}{n}\left[g_r(x_1) + \cdots + g_r(x_n)\right]. \tag{6}$$

4) Determine estimates of the Lagrange multipliers, $\widehat{\lambda}_0, \ldots, \widehat{\lambda}_m$, from:

$$a_r = \frac{\int_a^b g_r(x) e^{-\widehat{\lambda}_1 g_1(x) - \cdots - \widehat{\lambda}_m g_m(x)}\, dx}{\int_a^b e^{-\widehat{\lambda}_1 g_1(x) - \cdots - \widehat{\lambda}_m g_m(x)}\, dx} \tag{7}$$

and

$$e^{\widehat{\lambda}_0} = \int_a^b e^{-\widehat{\lambda}_1 g_1(x) - \cdots - \widehat{\lambda}_m g_m(x)}\, dx. \tag{8}$$

5) The estimated function then takes the form:

$$f(x) = \exp\left[-\widehat{\lambda}_0 - \cdots - \widehat{\lambda}_m g_m(x)\right]. \tag{9}$$

Maximum Likelihood From MaxEnt

The principle of maximum likelihood has been widely used to estimate the parameters of both statistical distributions and semi-empirical models. Maximum likelihood assumes that information exists about a random variable in the form of an observation, $x_1, \ldots, x_n$, and a density function, $f(x;\theta_1, \ldots, \theta_m)$, unlike in MaxEnt where the forms of the characterizing moments are known. The approach seeks to maximize the likelihood that the given observations will occur given a set of parameters. If each observation is independent, then this 'likelihood' is defined as:

$$L(\mathbf{X}; \Theta) = \prod_{i=1}^{n} f(x_i|\Theta). \tag{10}$$

The log likelihood function is most often used:

$$\ln L(\mathbf{X}; \Theta) = \sum_{i=1}^{n} \ln f(x_i|\Theta). \tag{11}$$

The $\ln L$ is maximized to determine the optimal parameter estimates $\widehat{\Theta}$.

The same objective can also be derived using the concept of MaxEnt, even though the former predates the latter. The parameters need to be chosen such that the entropy which remains after the observed values are known is large as possible. This implies that the entropy of the observation itself has to be a minimum. The entropy is given by:

$$-\int_a^b f(x, \Theta) \ln f(x, \Theta)\, dx = \int_a^b \ln f(x, \Theta)\, dF. \tag{12}$$

The knowledge which is given by the observation is:

$$\begin{cases} F(x, \Theta) = 0 & \text{when } x < x_1 \\ F(x, \Theta) = \frac{1}{n} & \text{when } x_1 \le x < x_2 \\ \vdots & \vdots \\ F(x, \Theta) = \frac{r}{n} & \text{when } x_r \le x < x_{r+1} \\ \vdots & \vdots \\ F(x, \Theta) = 1 & \text{when } x_n \le x \end{cases}$$

where $F(x, \Theta)$ is the cumulative density. Thus the entropy of the sample is then written as:

$$-\frac{1}{n}\left[\ln f(x_1, \Theta) + \cdots + \ln f(x_n, \Theta)\right], \tag{13}$$

which is equal to:

$$-\frac{1}{n}\left[L(x_1, \ldots, x_n; \theta_1, \ldots, \theta_m)\right], \tag{14}$$

where L is the same as described by (11). Therefore to minimize the entropy of the sample the likelihood function must be maximized.

Recursive Parameter Estimation

The determination of the parameters of a dynamical system on-line is a key step in the implementation of a wide range of control schemes. The estimation procedure is conducted in a recursive fashion in which the estimates from the previous time step are combined with the current state observations to calculate a new set of parameter estimates. The analysis of the estimation process used is typically approached from a mean square error criterion. This method requires some assumptions about the error to be made and the

form of the data processor to be restricted. H.L. Weidemann and E.B. Stear [9] presented an approach based on entropy concepts which has various benefits over the mean squared error method:

- The form of the optimal data processor is not constrained nor does it have to be known.
- Errors are not restricted to have a normal probability distribution.
- None of the operators in the system are required to be linear.

Before continuing with the analysis, various measures need to be defined. The entropy of a K-dimensional random vector $\mathbf{X}$ with the joint probability density function, $p_x(x_1, \ldots, x_k)$ is defined as:

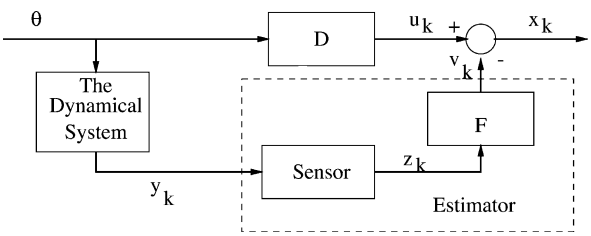
$$H(\mathbf{X}) = -\int_{-\infty}^{\infty} p_x(\mathbf{X}) \ln p_x(\mathbf{X})\, d\mathbf{X}. \tag{15}$$

If R_x is the covariance matrix of the vector $\mathbf{X}$ then the following holds:

$$H(\mathbf{X}) \leq \frac{1}{2} \ln \left\{ (2\pi e)^K \det[R_x] \right\}. \tag{16}$$

When $\mathbf{X}$ is a Gaussian random vector then (16) holds as an equality. Another quantity which will be used in the analysis is referred to as the *mutual information* between $\mathbf{X}$ and $\mathbf{Y}$.

$$I(\mathbf{X};\mathbf{Y}) = \int \int_{-\infty}^{\infty} p_{xy}(\mathbf{X},\mathbf{Y}) \ln \frac{p_{xy}(\mathbf{X},\mathbf{Y})}{p_y(\mathbf{Y}) p_y(\mathbf{Y})}\, d\mathbf{X}\, d\mathbf{Y}. \tag{17}$$



Entropy Optimization: Parameter Estimation, Figure 1
Typical parameter estimator

The object is to estimate a vector Θ of unknown parameters with the joint probability density function, $p_\theta(\theta_1, \ldots, \theta_m)$. The output of the dynamical model as a function of these parameters is expressed as $y_k(\theta_1, \ldots, \theta_m, k)$. These outputs are then measured by a sensor to produce $\{z_k\}$. These measurements are then used by the data processor F to produce an r-dimensional vector $\mathbf{V}$ which is an estimate of $D(\Theta)$. The estimation error is given by:

$$X = D(\Theta) - \mathbf{V} = D(\Theta) - F(\mathbf{Z}) = \mathbf{U} - \mathbf{V}. \tag{18}$$

Also, under certain conditions, the transform $D(\Theta)$ will possess a property that for any given random vector ξ, the following holds:

$$I(\xi;\Theta) = I(\xi; D(\Theta)), \tag{19}$$

or, in words, that $D(\Theta)$ preserves energy. When this does not hold, $I(\xi\Theta) > I(\xi D(\Theta))$.

The problem now is to determine the function $\hat{F}$ which will produce an optimal estimator. The theoretically best function results in a minimum of the error entropy, defined to be $\hat{H}_0$. The only constraint on the approach is that the mutual information, $I(\Theta;\mathbf{Z})$, must be known. With that the following can be stated:

- The minimum entropy of the error vector is given by:
$$H_0 = H(\mathbf{U}) - I(\mathbf{U};\mathbf{Z}). \tag{20}$$
- Minimizing the mutual information, $I(\mathbf{X};\mathbf{Z})$, is equivalent to the minimization of the error vector. This is achieved be choosing $F(\mathbf{Z})$ such that $\mathbf{Z}$ and $\mathbf{X}$ are independent.
- Whether or not $D(\Theta)$ preserves energy, the reduction in the processed parameter entropy, $H(\mathbf{U})$, is bounded above by $I(\Theta;\mathbf{Z})$, that is,
$$H(D(\Theta)) - H(\mathbf{X}) \leq I(\Theta;\mathbf{Z}), \tag{21}$$
and the equality holds when $D(\Theta)$ preserves energy and the optimal processor, $\hat{F}$, is used.

These three statements now make it possible to determine the best possible performance an estimator can achieve for a given system. The proofs of these statements and a simple example can be found in [9]. The extension of the theorems to the continuous time case is given in [6], and to the similar problem of state estimation in [7].

Parameter Estimation and Model Selection

For most problems of any physical significance the form of the model equations are not known with absolute certainty. In this lies the problem of not only estimated unknown parameters, but also determining the

best fitting model. Given a set of N independent observations, $x_1, \ldots, x_N$, of a random variable from an unknown true distribution $g(x)$, the objective is to estimate this true distribution by choosing a member of a family of distributions given by $f(x|\Theta)$ where Θ is a vector of parameters. In order to accomplish this, the distance between the two distributions needs to be minimized. The entropy of the true distribution is given by:

$$S(g;g) = \int g(x) \ln g(x)\, dx \tag{22}$$

while a measure of the *cross-entropy* is given by:

$$S(g; f(x|\Theta) = \int g(x) \ln f(x|\Theta)\, dx. \tag{23}$$

The Kullback–Leibler (K-L) measure is defined as:

$$I = S(g;g) - S(g; f(x|\Theta) = \int g(x) \ln \frac{g(x)}{f(x|\Theta)} dx. \tag{24}$$

Therefore the solution involves the minimization of the K-L measure [3].

Take the example of a family of possible distributions each one having a different number, k, of unknown parameters, Θ_k. These are denoted by $f(x|\Theta_k)$. The resulting form of the measure to choose the correct distribution is referred to as *Akaike's information criterion* (AIC) [1]:

$$\mathrm{AIC}(k) = -2 \ln L(\widehat{\Theta}_k) + 2k, \tag{25}$$

where $\ln L(\widehat{\Theta}_k)$ is the value of the log likelihood function with optimally determined parameters $\widehat{\Theta}_k$. It is proven in [3] that this result is obtained by the minimization of the K-L measure given by (24).

A secondary problem in the area of model selection, is sequential design of experiments. The concept of entropy has been applied to this problem in [2]. A total entropy criterion is developed which includes the uncertainty in the model selected as well as the uncertainty in the parameter values in each model. The use of this measure leads to a choice of an experiment for which the outcome is the most uncertain.

See also

- ► Entropy Optimization: Interior Point Methods
- ► Entropy Optimization: Shannon Measure of Entropy and its Properties
- ► Jaynes' Maximum Entropy Principle
- ► Maximum Entropy Principle: Image Reconstruction

References

1. Akaike H (1974) A new look at the statistical model identification. IEEE Trans Autom Control 19(6):716–723
2. Borth DM (1975) A total entropy criterion for the dual problem of model discrimination and parameter estimation. J Royal Statist Soc B 37:77–87
3. Bozdogan H (1987) Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions. Psychometrika 52(3):345–370
4. Kapur JN, Kesavan HK (1992) Entropy optimization principles and applications. Acad. Press, New York
5. Kullback S, Leibler RA (1951) On information and sufficiency. Ann Math Statist 22:79–86
6. Minamide N (1982) An extension of the entropy theorem for parameter estimation. Inform and Control 53:81–90
7. Minamide N, Nikiforuk PN (1993) Conditional entropy theorem for recursive parameter estimation and its application to state estimation problems. Internat J Syst Sci 24(1): 53–63
8. Shannon CE (1948) A mathematical theory of communication. Bell System Techn J 27:379–423, 623–659
9. Weidemann HL, Stear EB (1969) Entropy analysis of parameter estimation. Inform and Control 14:493–506

Entropy Optimization: Shannon Measure of Entropy and its Properties

Shu-Cherng Fang[1], Jacob H.-S. Tsao[2]

[1] North Carolina State University, Raleigh, USA

[2] San Jose State University, San Jose, USA

MSC2000: 94A17, 90C25

Article Outline

Keywords
See also
References

Keywords

Entropy; Cross-entropy; Maximum entropy principle; Minimum cross-entropy principle

The word *entropy* originated in the literature on thermodynamics around 1865 in Germany and was coined by R. Clausius [4] to represent a measure of the amount of energy in a thermodynamic system as a function of

the temperature of the system and the heat that enters the system. Clausius wanted a word similar to the German word *energie* (i. e., energy) and found it in the Greek word $\eta\tau\rho o\pi\eta$, which means transformation [1]. The word entropy had belonged to the domain of physics until 1948 when C.E. Shannon, while developing his theory of communication at Bell Laboratories, used the term to represent a measure of information after a suggestion made by J. von Neumann. Shannon wanted a word to describe his newly found *measure of uncertainty* and sought Von Neumann's advice. Von Neumann's reasoning to Shannon [25] was that: 'No one really understands entropy. Therefore, if you know what you mean by it and you use it when you are in an argument, you will win every time.'

Whatever the reason for the name is, the concept of Shannon's entropy has penetrated a wide range of disciplines, including statistical mechanics [12], thermodynamics [12], statistical inference [24], business and finance [5], nonlinear spectral analysis [21], image reconstruction [3], transportation and regional planning [26], queueing theory [10], information theory [9,20], statistics [17], econometrics [8], and linear and nonlinear programming [6,7].

The concept of entropy is closely tied to the concept of *uncertainty embedded in a probability distribution*. In fact, entropy can be defined as a measure of probabilistic uncertainty. For example, suppose the probability distribution for the outcome of a coin-toss experiment is (0.0001, 0.9999), with 0.0001 being the probability of having a tail. One is likely to notice that there is much more 'certainty' than 'uncertainty' about the outcome of this experiment and hence about the probability distribution. In fact, one is almost certain that the outcome will be a head. If, on the other hand, the probability distribution governing that same experiment were (0.5, 0.5), one would realize that there is much less 'certainty' and much more 'uncertainty,' when compared to the previous distribution. Generalizing this observation to the case of n possible outcomes, we conclude that the uniform distribution has the highest uncertainty out of all possible probability distributions. This implies that, if one had to choose a probability distribution for a chance experiment without any prior knowledge about that distribution, it would seem reasonable to pick the uniform distribution. This is because one would have no reason to choose any other and because that distribution maximizes the 'uncertainty' of the outcome. This is called *Laplace's principle of insufficient reasoning* [15]. Note that we are able to justify this principle without resorting to a rigorous definition of 'uncertainty.' However, this principle is inadequate when one has some prior knowledge about the distribution. Suppose, for example, that one knows some particular moments of the distribution, e. g., the expected value. In this case, a mathematical definition of 'uncertainty' is crucial. This is the case where Shannon's measure of uncertainty, or Shannon's entropy, plays an indispensable role [20].

To define *entropy*, Shannon proposed some axioms that he thought any measure of uncertainty should satisfy and deduced a unique function, up to a multiplicative constant, that satisfies them. It turned out that this function actually possesses many more desirable properties. In later years, many researchers modified and replaced some of his axioms in an attempt to simplify the reasoning. However, they all deduced that same function.

We first focus on finite-dimensional entropy, i. e., Shannon's entropy defined on discrete probability distributions that have a finite number of outcomes (or states). Let $\mathbf{p} \equiv (p_1, \ldots, p_n)^\mathsf{T}$ be a probability distribution associated with n possible outcomes, denoted by $\mathbf{x} \equiv (x_1, \ldots, x_n)^\mathsf{T}$, of an experiment. Denote its entropy by $S_n(\mathbf{p})$. Among those defining axioms, J.N. Kapur and H.K. Kesavan stated the following [15]:

1) $S_n(\mathbf{p})$ should depend on all the p_j's, $j = 1, \ldots, n$.
2) $S_n(\mathbf{p})$ should be a continuous function of p_j, $j = 1, \ldots, n$.
3) $S_n(\mathbf{p})$ should be permutationally symmetric. In other words, if the p_j's are merely permuted, then $S_n(\mathbf{p})$ should remain the same.
4) $S_n(1/n, \ldots, 1/n)$ should be a monotonically increasing function of n.
5) $S_n(p_1, \ldots, p_n) = S_{n-1}(p_1 + p_2, p_3, \ldots, p_n) + (p_1 + p_2) S_2(p_1/(p_1 + p_2), p_2/(p_1 + p_2))$.

Properties 1, 2 and 3 are obvious. Property 4 states that the maximum uncertainty of a probability distribution should increase as the number of possible outcomes increases. Property 5 is the least obvious but states that the uncertainty of a probability distribution is the sum of the uncertainty of the probability distribution that combines two of the outcomes and the uncertainty of the probability distribution consisting of only

those two outcomes adjusted by the combined probabilities of the two outcomes.

It turns out that the unique family of functions that satisfy the defining axioms has the form $S_n(\mathbf{p}) = -k \sum_{j=1}^{n} p_j \ln p_j$, where k is a positive constant, ln represents the natural logarithmic function, and $0 \ln 0 \equiv 0$ [15]. Shannon chose $-\sum_{j=1}^{n} p_j \ln p_j$ to represent his concept of entropy [20]. Among its many other desirable properties, we state the following:

6) Shannon's measure is nonnegative and concave in $p_1, \ldots, p_n$.
7) The measure does not change with the inclusion of a zero-probability outcome.
8) The entropy of a probability distribution representing a completely certain outcome is 0, and the entropy of any probability distribution representing uncertain outcomes is positive.
9) Given any fixed number of outcomes, the maximum possible entropy is that of the uniform distribution.
10) The entropy of the joint distribution of two independent distributions is the sum of the individual entropies.
11) The entropy of the joint distribution of two dependent distributions is no greater than the sum of the two individual entropies.

Property 6 is desirable because it is much easier to maximize a concave function than a nonconcave one. Properties 7 and 8 are appealing because a zero-probability outcome contributes nothing to uncertainty, and neither does a completely certain outcome. Property 9 was discussed earlier. Properties 10 and 11 state that joining two distributions does not affect the entropy, if they are independent, and may actually reduce the entropy, if they are dependent.

Shannon's entropy was originally defined for a probability distribution over a finite sample space, i. e., a finite number of possible outcomes, and can be interpreted as a measure of uncertainty of the probability distribution. It has subsequently been defined for general discrete and continuous random vectors. It has been rigorously proved that Shannon's entropy is the unique measure of uncertainty (up to a multiplicative constant) of a finite probability distribution that satisfies a set of axioms considered necessary for any reasonable measure of uncertainty [16,19,20]. The concept of entropy, when extended for probability distributions defined on a countably infinite sample space, takes the form of $-\sum_{j=1}^{\infty} p_j \ln p_j$. It can still be viewed as a measure of uncertainty but such an interpretation does not enjoy the same degree of mathematical rigor as its finite-sample-space counterpart. When the concept is extended for continuous probability distributions, it is defined to be $-\int p(x) \ln p(x)\, dx$. However, it can no longer be interpreted as a measure of uncertainty at all [9,11]. Rather, it can only be viewed as a measure of relative uncertainty [15].

Note that, with Shannon's entropy as the measure of uncertainty, in the absence of any prior information about the underlying probability distribution, the best course of action suggested by the principle of insufficient reasoning is to choose the uniform distribution because it possesses maximum uncertainty. Given the knowledge of some moments of the underlying distribution, the same reasoning leads to the following principle:

- Out of all possible distributions that are consistent with the moment constraints, choose the one that has maximum entropy.

This principle was proposed by E.T. Jaynes ([5, Chapter 2]), and has been known as the *principle of maximum entropy* or Jaynes' *maximum entropy principle*. It has often been abbreviated as *MaxEnt* in literature.

Let X be a random variable with n possible outcomes $\{x_1, \ldots, x_n\}$ and $\mathbf{p} \equiv (p_1, \ldots, p_n)^\intercal$ be a vector consisting of corresponding probabilities. Suppose that $g_1(X), \ldots, g_m(X)$ are m functions of X with known expected values $a_1, \ldots, a_m$, respectively. The principle of maximum entropy leads to the following mathematical *optimization* problem:

$$\begin{cases} \max & H_1(\mathbf{p}) = -\sum_{j=1}^{n} p_j \ln p_j \\ \text{s.t.} & \sum_{j=1}^{n} p_j\, g_i(x_j) = a_i, \quad i = 1, \ldots, m, \\ & \sum_{j=1}^{n} p_j = 1, \\ & p_j \geq 0, \quad j = 1, \ldots, n. \end{cases}$$

This is a convex programming problem with linear constraints. The nonnegativity constraints are not binding for the optimal solution $\mathbf{p}^*$ because each p_j^* can be expressed as an exponential function in terms

of the Lagrange multipliers associated with the equality constraints. Note that, in the absence of the moment constraints, the solution to the problem is the uniform probability distribution, whose entropy is ln n. As such, the maximum entropy principle can be viewed as an extension of the Laplace's principle of insufficient reasoning. The distribution selected under the maximum entropy principle has also been interpreted as one that is the 'most probable' in the sense that the maximum entropy distribution coincides with the frequency distribution that can be realized in the greatest number of ways [3]. An explanation of this linkage in the context of the well-known application of entropy maximization in transportation planning can be found in [7].

Recall that the above discussion was originally motivated by the task of choosing a probability distribution among those that are consistent with some given moments. Now, in addition to the moment constraints, suppose that we have an *a priori* probability distribution $\mathbf{p}^0$ that we think our probability distribution $\mathbf{p}$ should be close to. In fact, in the absence of the moment constraints, we would like to choose $\mathbf{p}^0$ for $\mathbf{p}$ because it is clearly the closest to $\mathbf{p}^0$. However, in the presence of some moment constraints which $\mathbf{p}^0$ does not satisfy, we need a precise definition of 'closeness' or 'deviation'. In other words, we need to define some sort of deviation or, more precisely, '*directed divergence*' [15] on the space of discrete probability distributions where the distribution is chosen from. Note that we deliberately avoid calling this measure a 'distance'. This is because a distance measure should be symmetric and should satisfy the triangular inequality, but these two properties are not important in this context. In fact, we can be content with a 'one-way (asymmetric) deviation measure', $D(\mathbf{p}, \mathbf{p}^0)$, from $\mathbf{p}$ to $\mathbf{p}^0$. If a 'one-way deviation measure' from $\mathbf{p}$ to $\mathbf{p}^0$ is not satisfactory, one can consider using a symmetric measure defined as the sum of $D(\mathbf{p}, \mathbf{p}^0)$ and $D(\mathbf{p}^0, \mathbf{p})$. What is desirable for this 'directed divergence' measure includes the following properties:

1) $D(\mathbf{p}, \mathbf{p}^0)$ should be nonnegative for all $\mathbf{p}$ and $\mathbf{p}^0$.
2) $D(\mathbf{p}, \mathbf{p}^0) = 0$ if and only if $\mathbf{p} = \mathbf{p}^0$.
3) $D(\mathbf{p}, \mathbf{p}^0)$ should be a convex function of $p_1, \ldots, p_n$.
4) When $D(\mathbf{p}, \mathbf{p}^0)$ is minimized subject to moment constraints but without the explicit presence of the nonnegativity constraints, the resulting p_j's should be nonnegative.

Property 1 is desirable for any such measure of deviation. If property 2 were not satisfied, then it would be possible to choose a vector $\mathbf{p}$ that has a zero directed divergence from $\mathbf{p}^0$, i. e., one that is as 'close' to $\mathbf{p}^0$ as $\mathbf{p}^0$ itself, but differs from $\mathbf{p}^0$. Property 3 makes minimizing the measure much simpler, and property 4 spares us from explicitly considering n nonnegativity constraints. Fortunately, there are many measures that satisfy these properties. We may even be able to find one that satisfies the triangular inequality. But, simplicity of the measure is also desirable. The simplest and most important of those measures is the Kullback–Leibler measure ([5, Chapt. 4]), defined as $D(\mathbf{p}, \mathbf{p}^0) = \sum_{j=1}^{n} p_j \ln (p_j/p_j^0)$, with the convention that, whenever p_j^0 is 0, p_j is set to 0 and 0 ln (0/0) is defined to be 0. This measure is also known as the *cross-entropy*, *relative entropy*, *directed divergence* or *expected weight of evidence* of $\mathbf{p}$ with respect to $\mathbf{p}^0$. A. Hobson [1] provided an axiomatic characterization of *cross-entropy*. He interpreted $D(\mathbf{p}, \mathbf{p}^0)$ as the 'information in $\mathbf{p}$ relative to $\mathbf{p}^0$', and showed that the only function $I(\mathbf{p}, \mathbf{p}^0)$ satisfying the following five properties has the form of $k \sum_{j=1}^{n} p_j \ln (p_j/p_j^0)$, where k is a positive constant:

5) $I(\mathbf{p}, \mathbf{p}^0)$ is a continuous function of $\mathbf{p}$ and $\mathbf{p}^0$.
6) $I(\mathbf{p}, \mathbf{p}^0)$ is permutationally symmetric, i. e., the measure does not change if the pairs of (p_j, p_j^0) are permuted among themselves.
7) $I(\mathbf{p}, \mathbf{p}) = 0$.
8) For any pair of integers n and n_0 such that $n_0 \geq n > 0$, $I(1/n, \ldots, 1/n, 0, \ldots, 0; 1/n_0, \ldots, 1/n_0)$ is an increasing function of n_0 and a decreasing function of n, where $I(1/n, \ldots, 1/n, 0, \ldots, 0; 1/n_0, \ldots, 1/n_0)$ denotes the information obtained when the number of equally likely possibilities is reduced from n_0 to n.
9)
$$I(p_1, \ldots, p_n; p_1^0, \ldots, p_n^0) = I(q_1, q_2; q_1^0, q_2^0) + q_1 I\left(\frac{p_1}{q_1}, \ldots, \frac{p_r}{q_1}; \frac{p_1^0}{q_1^0}, \ldots, \frac{p_r^0}{q_1^0}\right) + q_2 I\left(\frac{p_{r+1}}{q_2}, \ldots, \frac{p_n}{q_2}; \frac{p_{r+1}^0}{q_2^0}, \ldots, \frac{p_n^0}{q_2^0}\right),$$
where $1 \leq r \leq n$, $q_1 \equiv p_1 + \cdots + p_r$, $q_2 \equiv p_r + 1 + \cdots + p_n$, $q_1^0 \equiv p_1^0 + \cdots + p_r^0$, $q_2^0 \equiv p_{r+1}^0 + \cdots + p_n^0$.

Property 8 says, for example, that the information obtained upon reducing the number of equally likely sides on a die from 6 to 3 is greater than the information obtained upon reducing the number from 6 to 4. Prop-

erty 9 says that one may give information about the outcome associated with the random event either by specifying the probabilities $p_1, \ldots, p_n$ directly, or by specifying the probabilities q_1 and q_2 first and then specifying the conditional probabilities p_i/q_1 and p_i/q_2.

In addition to the nine properties discussed above, we state the following desirable properties for cross-entropy:

10) $D(\mathbf{p}, \mathbf{p}^0)$ is convex in both $\mathbf{p}$ and $\mathbf{p}^0$.
11) $D(\mathbf{p}, \mathbf{p}^0)$ is not symmetric.
12) If $\mathbf{p}$ and $\mathbf{q}$ are independent and $\mathbf{r}$ and $\mathbf{s}$ are also independent, then $D(\mathbf{p} * \mathbf{q}, \mathbf{r} * \mathbf{s}) = D(\mathbf{p}, \mathbf{r}) + D(\mathbf{q}, \mathbf{s})$, where $*$ denotes the convolution operation between two independent distributions.
13) In general, the triangular inequality does not hold. But, if distribution $\mathbf{p}$ minimizes $D(\mathbf{p}, \mathbf{p}^0)$ subject to some moment constraints and $\mathbf{q}$ is any other distribution that satisfies those same constraints, then $D(\mathbf{q}, \mathbf{p}^0) = D(\mathbf{q}, \mathbf{p}) + D(\mathbf{p}, \mathbf{p}^0)$. Thus, in this special case, the triangular inequality holds, but as an equality.

Kullback and Leibler's cross-entropy was also originally defined for probability distributions with a finite sample space and can be interpreted as a measure of deviation of one probability distribution from another. It has been extended subsequently for distributions defined on countably infinite and continuous sample spaces. The corresponding forms become $\sum_{j=1}^{\infty} p_j \ln (p_j/p_j^0)$ and $\int p(x) \ln (p(x)/p^0(x))\, dx$, respectively. It has also been derived rigorously as the unique measure of deviation of one probability distribution from another that satisfies a set of axioms considered as necessity for any reasonable measure of deviation, for both finite probability distributions [11] and continuous distributions [14]. Cross-entropy for probability distributions with countably infinite sample space can be viewed and has been used as a measure of deviation, although the justification is not as strong as their finite-sample-space and continuous counterparts.

With cross-entropy interpreted as a measure of 'deviation', the Kullback–Leibler's *principle of minimum cross-entropy*, or *MinxEnt*, can be stated as follows [15]:

> Out of all possible distributions that are consistent with the moment constraints, choose the one that minimizes the cross-entropy with respect to the given *a priori* distribution.

Mathematically, we consider the following *optimization* problem:

$$\begin{cases} \min & H_2(\mathbf{p}) = \sum_{j=1}^{n} p_j \ln \dfrac{p_j}{p_j^0} \\ \text{s.t.} & \sum_{j=1}^{n} p_j\, g_i(x_j) = a_i, \quad i = 1, \ldots, m, \\ & \sum_{j=1}^{n} p_j = 1, \\ & p_j \geq 0, \quad j = 1, \ldots, n. \end{cases}$$

Note that the nonnegativity constraints are not binding, for the same reason as in the MaxEnt problem. For a detailed discussion of the properties of MinxEnt, the reader is referred to [23].

Note that, if there is no a priori information, then one may use the uniform distribution, denoted by $\mathbf{u}$, as the a priori distribution. In this case, $D(\mathbf{p}, \mathbf{p}^0) = D(\mathbf{p}, \mathbf{u}) = \sum_{j=1}^{n} p_j \ln (p_j/(1/n)) = \ln n + \sum_{j=1}^{n} p_j \ln p_j$. Since minimizing $\sum_{j=1}^{n} p_j \ln p_j$ is equivalent to maximizing $-\sum_{j=1}^{n} p_j \ln p_j$, minimizing the cross-entropy with respect to the uniform distribution is equivalent to maximizing entropy and, therefore, MaxEnt is a special case of MinxEnt. These two principles can now be combined into a general principle:

> Out of all probability distributions satisfying the given moment constraints, choose the distribution that minimizes the cross-entropy with respect to the given a priori distribution and, in the absence of it, choose the distribution that minimizes the cross-entropy with respect to the uniform distribution.

Both the MaxEnt and MinxEnt principles for selecting finite-sample-space probability distributions and the MinxEnt principle for selecting continuous probability distributions can be *axiomatically derived* [22]. Under four consistency axioms, it was shown that the two principles are uniquely correct methods for inductive inference when new information is given in the form of expected values. Many well-known and widely used distributions, including the normal, gamma and geometric distributions, can actually be derived as solutions to some MaxEnt or MinxEnt problems [15].

The maximum entropy principle has also been shown to be a dual principle of the *maximum likelihood*

principle for the exponential family of probability distributions in the sense that a dual problem to the linearly constrained entropy maximization problem is equivalent to the problem of maximizing a likelihood function with respect to the parameters of an exponential family [2]. This principle has also been shown to be related to the *Bayesian parameter estimation* problem [7]. *Duality theory* and major mathematical *algorithms* for solving finite-dimensional MaxEnt or MinxEnt problems can be found in [7] and the references therein.

See also

- ► Entropy Optimization: Interior Point Methods
- ► Entropy Optimization: Parameter Estimation
- ► Jaynes' Maximum Entropy Principle
- ► Maximum Entropy Principle: Image Reconstruction
- ► Optimization in Medical Imaging

References

1. Baierlein R (1992) How entropy got its name. Amer J Phys 60:1151
2. Ben-Tal A, Teboulle M, Charnes A (1988) The role of duality in optimization problems involving entropy functionals with applications to information theory. J Optim Th Appl 58:209–223
3. Burch SF, Gull SF, Skilling JK (1983) Image restoration by a powerful maximum entropy method. Computer Vision, Graphics, and Image Processing 23:113–128
4. Clausius R (1865) Ueber Verschiedene fur die Anwendung Bequeme Formen der Hauptgleichungen der Mechanischen Warmetheorie. Ann Physik und Chemie 125:353–400
5. Cozzolino JM, Zahner MJ (1973) The maximum entropy distribution of the future market price of a stock. Oper Res 21:1200–1211
6. Erlander S (1981) Entropy in linear programming. Math Program 21:137–151
7. Fang S-C, Rajasekera JR, Tsao H-SJ (1997) Entropy optimization and mathematical programming. Kluwer, Dordrecht
8. Golan A, Judge G, Miller D (1996) Maximum entropy econometrics: robust estimation with limited data. Wiley, New York
9. Guiasu S (1977) Information theory with applications. McGraw-Hill, New York
10. Guiasu S (1986) Maximum entropy condition in queueing theory. J Oper Res Soc 37:293–301
11. Hobson A (1987) Concepts in statistical mechanics. Gordon and Breach, New York
12. Jaynes ET (1957) Information theory and statistical mechanics II. Phys Rev 108:171–190
13. Jaynes ET (1968) Prior probabilities. IEEE Trans Syst, Sci Cybern SSC-4:227–241
14. Johnson RW (1979) Axiomatic characterization of the directed divergence and their linear combinations. IEEE Trans Inform Theory 25:709–716
15. Kapur JN, Kesavan HK (1992) Entropy optimization principles with applications. Acad. Press, New York
16. Khinchin AI (1957) Mathematical foundations of information theory. Dover, Mineola, NY
17. Kullback S (1968) Information theory and statistics. Dover, Mineola, NY
18. Scott CH, Jefferson TR (1977) Entropy maximizing models of residential location via geometric programming. Geographical Anal 9:181–187
19. Shannon CE (1948) A mathematical theory of communication. Bell System Techn J 27:379–423; 623–656
20. Shannon CE, Weaver W (1962) The mathematical theory of communication. Univ. Illinois Press, Champaign
21. Shore JE (1981) Minimum cross-entropy spectral analysis. IEEE Trans Acoustics, Speech and Signal Processing 29:230–237
22. Shore JE, Johnson RW (1980) Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. IEEE Trans Inform Theory 26:26–37
23. Shore JE, Johnson RW (1981) Properties of cross-entropy minimization. IEEE Trans Inform Theory 27:472–482
24. Tribus M (1969) Rational descriptions, decisions, and designs. Pergamon, Oxford
25. Tribus M (1988) An engineer looks at Bayes. In: Erickson GJ, Smith CR (eds) Maximum-Entropy and Bayesian Methods in Sci. and Engineering: Foundations. 1, Kluwer, Dordrecht, pp 31–52
26. Wilson AG (1970) Entropy in urban and regional modeling. Pion, London

Equality-Constrained Nonlinear Programming: KKT Necessary Optimality Conditions

EQNLP

ANDERS FORSGREN
Royal Institute Technol. (KTH), Stockholm, Sweden

MSC2000: 49M37, 65K05, 90C30

Article Outline

Keywords
See also
References

Keywords

Equality-constrained optimization; KKT necessary optimality conditions

An equality-constrained nonlinear programming problem may be posed in the form

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & f(x) \\ \text{subject to} & c(x) = 0, \end{cases} \tag{1}$$

where f is a real-valued nonlinear function and c is an m-vector of real-valued nonlinear functions with ith component $c_i(x)$, $i = 1, \ldots, m$. Normally, with the term *equality-constrained nonlinear programming problem* is meant a problem of the form (1) where f and c are sufficiently smooth, at least continuously differentiable. This will be assumed throughout this discussion, with the gradient of $f(x)$ denoted by $g(x)$ and the $m \times n$ Jacobian of $c(x)$ denoted by $J(x)$.

Of fundamental importance for equality-constrained optimization problems are the *first order necessary optimality conditions*. These conditions are often referred to as the *KKT necessary optimality conditions*, or more briefly, the *KKT conditions*. The KKT conditions state that if x^* is a local minimizer to (1) that satisfies a certain *constraint qualification*, then there exists an m-dimensional vector λ^* such that

$$g(x^*) - J(x^*)^\top \lambda^* = 0,$$
$$c(x^*) = 0.$$

The vector λ^* is usually referred to as the vector of *Lagrange multipliers*. For equality-constrained problems, the KKT conditions are attributed to J.L. Lagrange, and hence 'classical'. The acronym KKT arises from the more general results on inequality-constrained problems provided by W. Karush [3], H.W. Kuhn and A.W. Tucker [4,5].

For an equality-constrained problem, the KKT conditions state that x^* must be feasible, i. e., $c(x^*) = 0$; and that the gradient must have zero projection onto the null space of the constraint gradients, i. e., there exists a λ^* such that $g(x^*) = J(x^*)^\top \lambda^*$. In the case of linear equality constraints, i. e., $c(x) = Ax - b$ for some $(m \times n)$-matrix A and m-vector b, it follows that if x^* is feasible, then $x^* + p$ is feasible if and only if $Ap = 0$. Hence, in this situation, if x^* is a local minimizer, it must hold that $g(x^*)^\top p = 0$ for all p such that $Ap = 0$. But this is equivalent to the existence of a λ^* such that $g(x^*) = A^\top \lambda^*$. Consequently, in the case of linear constraints, the KKT conditions are necessary for x^* to be a local minimizer to problem (1). Constraint qualifications essentially ensure that the linearization of c at x^* provided by $J(x^*)$ adequately describes c in a neighborhood of x^*. A constraint qualification which is frequently used is that $J(x^*)$ has rank m, i. e., that the gradients of the constraints are linearly independent at x^*. The related *Fritz John necessary optimality conditions* are valid without any constraint qualification.

The KKT conditions are of fundamental importance, not only from a theoretical point of view, but also algorithms for solving equality-constrained nonlinear programming problems are often based on finding a solution to the KKT conditions. In general, the KKT conditions are not sufficient for x^* to be a local minimizer, but second order optimality conditions need be considered. However, if c is affine and f is a convex function on the feasible region, then the KKT conditions are sufficient for x^* to be a global minimizer. Detailed discussions on optimality conditions can be found in textbooks on nonlinear programming, e. g., [1,2,6].

As a simple example, consider the two-dimensional problem where $f(x) = x_1$ and $c(x) = (x_1^2 + x_2^2 - 1)/2$. Then, the KKT conditions have two solutions: $\tilde{x} = (1, 0)^\top$ together with $\tilde{\lambda} = 1$, and $\hat{x} = (-1, 0)^\top$ together with $\hat{\lambda} = -1$ However, only $\hat{x}$ is a local minimizer (and in fact also a global minimizer).

See also

- First Order Constraint Qualifications
- Inequality-Constrained Nonlinear Optimization
- Kuhn–Tucker Optimality Conditions
- Lagrangian Duality: Basics
- Relaxation in Projection Methods
- Rosen's Method, Global Convergence, and Powell's Conjecture
- Saddle Point Theory and Optimality Conditions
- Second Order Constraint Qualifications
- Second Order Optimality Conditions for Nonlinear Optimization
- SSC Minimization Algorithms
- SSC Minimization Algorithms for Nonsmooth and Stochastic Optimization

References

1. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: Theory and algorithms, 2nd edn. Wiley, New York
2. Bertsekas DP (1995) Nonlinear programming. Athena Sci., Belmont, MA
3. Karush W (1939) Minima of functions of several variables with inequalities as side constraints. Master's Thesis, Dept Math Univ Chicago
4. Kuhn HW (1991) Nonlinear programming: A historical note. In: Lenstra JK, Rinnooy Kan AHG, Schrijver A (eds) History of Mathematical Programming: A Collection of Personal Reminiscences. Elsevier, Amsterdam, pp 82–96
5. Kuhn HW, Tucker AW (1951) Nonlinear programming. In: Neyman J (ed) Proc. Second Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, Berkeley, pp 481–492
6. Nash SG, Sofer A (1996) Linear and nonlinear programming. McGraw-Hill, New York

Equilibrium Networks

Anna Nagurney
University Massachusetts, Amherst, USA

MSC2000: 90C30

Article Outline

Keywords
A Multimodal Traffic Network Equilibrium Model
A Migration Network Equilibrium Model
See also
References

Keywords

Traffic network equilibrium; Spatial price equilibrium; Migration equilibrium; Multimodal networks; Multiclass migration

Many complex systems in which agents compete for scarce resources on a network, be it a physical one, as in the case of congested urban transportation systems, or an abstract one, as in the case of certain economic and financial problems, can be formulated and studied as *network equilibrium* problems. Applications of network equilibrium problems are common in many disciplines, in particular, in operations research and management science and in economics and engineering (cf. [10,17]).

Network equilibrium problems as opposed to network optimization problems involve competition among the agents or users of the network system. Moreover, network equilibrium problems are governed by an underlying behavioral principle as to the behavior of the agents as well as the equilibrium conditions. For example, in congested urban transportation systems in which users seek to determine their cost minimizing routes of travel, the equilibrium conditions, due to J.G. Wardrop [23] (see also [2] and [8]), state that, in equilibrium all used paths connecting an origin/destination pair will have minimal and equal user travel costs. On the other hand, in the case of spatial price equilibrium patterns one seeks to determine the commodity production, trade, and consumption pattern satisfying the equilibrium condition, due to S. Enke [9] and P.A. Samuelson [20], that expresses that there will be trade between a pair of spatially separated supply and demand markets provided the supply price of the commodity at the supply market plus the unit cost of transportation associated with shipping the commodity is equal to the demand price of the commodity at the demand market; if the supply price plus the transportation cost exceed the demand price, then there will be no trade between this pair of supply and demand markets.

M.J. Beckmann, C.B. McGuire, and C.B. Winsten [2] initiated the systematic study of network equilibrium problems in the general setting of traffic networks and demonstrated that the equilibrium flow pattern satisfying the traffic network equilibrium conditions (see also [23]), under certain symmetry assumptions on the underlying functions, could be reformulated as the solution to an optimization problem. Samuelson [20], following [9], had made a similar connection but in the more specialized context of spatial price equilibrium problems on networks that were bipartite.

M.J. Smith [22] later proposed an alternative formulation of traffic network equilibrium conditions which were then identified by S.C. Dafermos [3] to satisfy a finite-dimensional variational inequality problem. This connection allowed for the relaxation of the symmetry assumption and, consequently, for the construction of more realistic models (cf. [17,21], and the references therein).

Other network equilibrium applications whose study and understanding have benefited from this methodology (cf. [10,14,17,19]), include: spatial price equilibrium problems (see, e. g., [11,15]), oligopolistic market equilibrium problems ([7,12,13]), migration equilibrium problems (cf. [16,18]), and general economic equilibrium problems (cf. [5]).

Here we present two examples of network equilibrium problems for illustrative purposes with the first example being a multimodal/multiclass transportation network equilibrium problem in which the network is a physical one whereas the second problem is a multiclass migration equilibrium problem which is isomorphic to a specially structure multiclass traffic network equilibrium problem.

Additional background, models and applications, qualitative results, as well as computational procedures and references can be found in [17] and [10].

A Multimodal Traffic Network Equilibrium Model

We now present a multimodal traffic network equilibrium model (cf. [3,4,6]). The model is a fixed demand model in that the demands associated with traveling between the origin/destination pairs are assumed known. See [17] for additional background, as well as elastic demand traffic network equilibrium models and other network equilibrium problems.

Consider a general network $N = [G, A]$, where N denotes the set of nodes and A the set of directed links. Let $a, b, c, \ldots$ denote the links, $p, q, \ldots$ the paths. Assume that there are J origin/destination (O/D) pairs, with a typical O/D pair denoted by w, and n modes of transportation on the network with typical modes denoted by $i, j, \ldots$.

The flow on a link a generated by mode i is denoted by f_a^i, and the user cost associated with traveling by mode i on link a is denoted by c_a^i. Group the link flows into a column vector $f \in \mathbf{R}^{nL}$, where L is the number of links in the network. Group the link costs into a row vector $c \in \mathbf{R}^{nL}$. Assume that the user cost on a link and a particular mode may, in general, depend upon the flows of every mode on every link in the network, that is,

$$c = c(f),$$

where c is a known smooth function.

The travel demand of users of mode i traveling between O/D pair w is denoted by d_w^i and the travel disutility associated with traveling between this O/D pair using the mode is denoted by λ_w^i. Group the demands into a vector $d \in \mathbf{R}^{nJ}$.

The flow on path p due to mode i is denoted by x_p^i. Group the path flows into a column vector $x \in \mathbf{R}^{nQ}$, where Q denotes the number of paths in the network.

The conservation of flow equations are as follows. The demand for a mode and O/D pair must be equal to the sum of the flows of the mode on the paths joining the O/D pair, that is,

$$d_w^i = \sum_{p \in P_w} x_p^i, \qquad \forall i, \quad \forall w,$$

where P_w denotes the set of paths connecting w.

A nonnegative path flow vector x which satisfies the demand constraint is termed feasible. Moreover, we must have that

$$f_a^i = \sum_p x_p^i \delta_{ap},$$

that is, for each mode, the link load associated with a mode is equal to the sum of the path flows of that mode on paths that utilize that link.

A user traveling on path p using mode i incurs a user (or personal) travel cost C_p^i satisfying

$$C_p^i = \sum_a c_a^i \delta_{ap},$$

in other words, the cost on a path p due to mode i is equal to the sum of the link costs of links comprising that path and using that mode.

The traffic network equilibrium conditions are given below.

Definition 1 (***multimodal traffic network equilibrium***) ([2,3,4]) A link load pattern f^* satisfying the feasibility conditions is an equilibrium pattern, if, once established, no user has any incentive to alter his travel arrangements. This state is characterized by the following equilibrium conditions, which must hold for every mode i, every O/D pair w, and every path $p \in P_w$:

$$C_p^i \begin{cases} = \lambda_w^i & \text{if} \quad x_p^{i\,*} > 0, \\ \geq \lambda_w^i & \text{if} \quad x_p^{i\,*} = 0, \end{cases}$$

where λ_w^i is the equilibrium travel disutility associated with the O/D pair w and mode i.

We now define the feasible set K as

$$K \equiv \left\{ f: \begin{array}{c} \exists x \geq 0, \\ \text{the demand constraints and} \\ \text{the link load constraints hold} \end{array} \right\}.$$

One can verify (see [3]) that the variational inequality governing equilibrium conditions for this model would be given as in the subsequent theorem.

Theorem 2 (*variational inequality formulation*) *A vector $f^* \in K$ is an equilibrium pattern, if and only if, it satisfies the variational inequality problem*

$$\langle c(f^*), f - f^* \rangle \geq 0, \qquad \forall f \in K.$$

Note that this variational inequality is in link loads. One can also derive a variational inequality problem in path flows (see also [1,4,17]). Existence of an equilibrium f^* follows from the standard theory of variational inequalities (cf. [14]) solely from the assumption that c is continuous, since the feasible set K is now compact.

In the special case where the symmetry condition

$$\left[\frac{\partial c_a^i}{\partial f_b^j} = \frac{\partial c_b^j}{\partial f_a^i} \right], \qquad \forall i, j; a, b,$$

holds, then the variational inequality problem can be reformulated as the solution to an optimization problem. This symmetry assumption, however, is not expected to hold in most applications. Consequently, the variational inequality problem which is the more general problem formulation is needed. For example, the symmetry condition essentially says that the flow on link b due to mode j should affect the cost of mode i on link a in the same manner that the flow of mode i on link a affects the cost on link b and mode j. In the case of a single mode problem, the symmetry condition would imply that the cost on link a is affected by the flow on link b in the same manner as the cost on link b is affected by the flow on link a.

A Migration Network Equilibrium Model

Human migration is a topic that has been studied not only by economists, but also by demographers, sociologists, and geographers. Here a model of human migration is described, which is shown to have a simple, abstract network structure in which the links correspond to locations and the flows on the links to populations of a particular class at the particular location. Hence, the model is isomorphic to the traffic network equilibrium problem just described on a network with special structure. For additional details, see [16,17,18].

Assume a closed economy in which there are n locations, typically denoted by i, and J classes, typically denoted by k. Assume further that the attractiveness of any location i as perceived by class k is represented by a utility u_i^k. Let $\overline{p}^k$ denote the fixed and known population of class k in the economy, and let p_i^k denote the population of class k at location i. Group the utilities into a row vector $u \in \mathbf{R}^{Jn}$ and the populations into a column vector $p \in \mathbf{R}^{Jn}$. Assume no births and no deaths in the economy.

The conservation of flow equation for each class k is given by

$$\overline{p}^k = \sum_{i=1}^{n} p_i^k,$$

where $p_i^k \geq 0$, $k = 1, \ldots, J$; $i = 1, \ldots, n$. Let

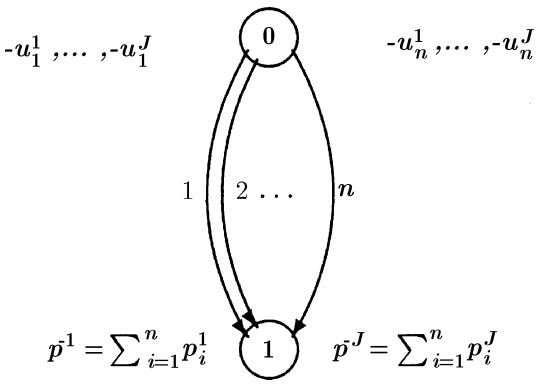
$$K \equiv \left\{ p: \begin{array}{c} p \geq 0 \text{ and satisfy the} \\ \text{conservation of flow equation} \end{array} \right\}.$$

The *conservation of flow equation* expresses that the population of each class k must be conserved in the economy.

Definition 3 (*migration equilibrium*) Assume that the migrants are rational and that migration will continue until no individual of any class has any incentive to move since a unilateral decision will no longer yield an increase in the utility. Mathematically, hence, a multiclass population vector $p^* \in K$ is said to be in equilibrium if for each class k, $k = 1, \ldots, J$:

$$u_i^k \begin{cases} = \lambda^k & \text{if } p_i^{k*} > 0 \\ \leq \lambda^k & \text{if } p_i^{k*} = 0. \end{cases}$$

The equilibrium conditions express that for a given class k only those locations i with maximal utility will have a positive population volume of the class. Moreover, the utilities for a given class are equilibrated across the locations.



Equilibrium Networks, Figure 1
Network equilibrium formulation of a multiclass migration equilibrium model

We now discuss the utility functions. Assume that, in general, the utility associated with a particular location as perceived by a particular class, may depend upon the population associated with every class and every location, that is, assume that

$$u = u(p).$$

Note that in allowing the utility to depend upon the populations of the classes, we are using populations as a proxy for amenities associated with a particular location. Such a utility function can also model the negative externalities associated with overpopulation, such as congestion, increased crime, competition for scarce resources, etc.

As illustrated in [17], the above migration model is equivalent to a network equilibrium model with a single origin/destination pair and fixed demands. Indeed, one can make the identification as follows. Construct a network consisting of two nodes, an origin node 0 and a destination node 1, and n links connecting the origin node to the destination node. Associate with each link i, J costs: $-u_i^1, \ldots, u_i^J$, and link flows represented by $p_i^1, \ldots, p_i^J$. This model is, hence, equivalent to a multimodal traffic network equilibrium model with fixed demand for each mode, consisting of a single origin/destination pair, and J paths connecting the O/D pair. Note that one can make J copies of the network, in which case, each ith network will correspond to class i with the cost functions on the links defined accordingly. This identification enables us to immediately write down the following:

Theorem 4 (*variational inequality formulation*) *A population pattern $p^* \in K$ is in equilibrium, if and only if it satisfies the variational inequality problem:*

$$\langle -u(p^*), p - p^* \rangle \geq 0, \qquad \forall p \in K.$$

Existence of an equilibrium then follows from the standard theory of variational inequalities, since the feasible set K is compact, assuming that the utility functions are continuous. Uniqueness of the equilibrium population pattern also follows from the standard theory provided that the $-u$ function is strictly monotone. The interpretation of this monotonicity condition in the context of applications is that condition implies that the utility associated with a given class and location is expected to be a decreasing function of the population of that class at that location.

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Evacuation Networks
- ▶ Financial Equilibrium
- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Generalized Networks
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Oligopolistic Market Equilibrium
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Spatial Price Equilibrium
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium
- ▶ Walrasian Price Equilibrium

References

1. Aashtiani HZ, Magnanti TL (1981) Equilibria on a congested transportation network. SIAM J Alg Discrete Meth 2:213–226
2. Beckmann MJ, McGuire CB, Winsten CB (1956) Studies in the economics of transportation. Yale Univ. Press, New Haven, CT
3. Dafermos S (1980) Traffic equilibrium and variational inequalities. Transport Sci 14:43–54
4. Dafermos S (1982) The general multimodal network equilibrium problem with elastic demand. Networks 14:43–54
5. Dafermos S (1990) Exchange price equilibria and variational inequalities. Math Program 46:391–402
6. Dafermos S, Nagurney A (1984) Stability and sensitivity analysis for the general network equilibrium-travel choice model. In: Volmuller J, Hamerslag R (eds) Proc. 9th Internat. Symp. Transportation and Traffic Theory, VNU Sci. Press, Utrecht, pp 217–234
7. Dafermos S, Nagurney A (1987) Oligopolistic and competitive behavior of spatially separated markets. Regional Sci and Urban Economics 17:245–254
8. Dafermos S, Sparrow FT (1969) The traffic assignment problem for a general network. J Res Nat Bureau Standards 73B:91–118
9. Enke S (1951) Equilibrium among spatially separated markets: solution by electronic analogue. Econometrica 10: 40–47
10. Florian M, Hearn D (1995) Network equilibrium models and algorithms. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing. Handbook Oper Res and Management Sci. Elsevier, Amsterdam, pp 485–550
11. Florian M, Los M (1982) A new look at static spatial price equilibrium models. Regional Sci and Urban Economics 12:579–597
12. Gabay D, Moulin H (1980) On the uniqueness and stability of Nash-equilibria in noncooperative games. In: Bensoussan A, Kleindorfer P, Tapiero CS (eds) Applied Stochastic Control in Econometrics and Management Science. North-Holland, Amsterdam, 271–294
13. Haurie A, Marcotte P (1985) On the relationship between Nash-Cournot and Wardrop equilibria. Networks 15:295–308
14. Kinderleher D, Stampacchia G (1980) An introduction to variational inequalities and their applications. Acad. Press, New York
15. Nagurney A (1987) Computational comparisons of spatial price equilibrium methods. J Reg Sci 27:55–76
16. Nagurney A (1989) Migration equilibrium and variational inequalities. Economics Lett 31:109–112
17. Nagurney A (1999) Network economics: A variational inequality approach, 2nd edn. Kluwer, Dordrecht
18. Nagurney A, Pan J, Zhao L (1991) Human migration networks. Europ J Oper Res
19. Patriksson M (1994) The traffic assignment problem. VSP, Utrecht
20. Samuelson PA (1952) A spatial price equilibrium and linear programming. Amer Economic Rev 42:283–303
21. Sheffi Y (1985) Urban transportation networks. Prentice-Hall, Englewood Cliffs, NJ
22. Smith MJ (1979) The existence, uniqueness, and stability of traffic equilibria. Transport Res 13B:259–304
23. Wardrop JG (1952) Some theoretical aspects of road traffic research. Proc Inst Civil Engineers II:325–378

Equivalence Between Nonlinear Complementarity Problem and Fixed Point Problem

GEORGE ISAC
Royal Military College of Canada, Kingston, Canada

MSC2000: 90C33

Article Outline

Keywords
Preliminaries
Nonlinear Complementarity Problem
Solvability by Fixed Points Theorems
The Nonlinear Complementarity Problem as a Mathematical Tool in Fixed Point Theory
Conclusions
See also
References

Keywords

Nonlinear complementarity problem; Fixed point problem

Complementarity theory is a new domain of applied mathematics strongly related to Linear Analysis, Nonlinear Analysis, Topology, Variational Inequalities Theory, Ordered Topological Vector Spaces, Numerical Analysis etc. The main goal in this theory is the study of complementarity problems. It is well known that complementarity problems encompass a variety of practical problems arising in: Optimization, Structural Mechanics, Elasticity, Economics etc. [8]. The relation between the general nonlinear complementarity problem and the fixed point problem it seems to be remarkable. The main aim of this article is the study of this relation.

Preliminaries

Let E, E^* be a pair of real locally convex spaces. The space E^* can be the topological dual of E. Let $\langle \cdot, \cdot \rangle$ be a bilinear form on $E \times E^*$ satisfying the separation axioms:

s_1) $\langle x_0, y \rangle = 0$ for all $y \in E^*$ implies $x_0 = 0$;
s_2) $\langle x, y_0 \rangle = 0$ for all $x \in E$ implies $y_0 = 0$.

The triplet $(E, E^*, \langle \cdot, \cdot \rangle)$ is called a *dual system* or a *duality* (denoted by $\langle E, E^* \rangle$). In practical problems, the space E can be a Banach space and E^* its topological dual and $\langle x, y \rangle = y(x)$ for all $x \in E$ and $y \in E^*$. When E is a Hilbert space $(H, \langle \cdot, \cdot \rangle)$ or the Euclidean space $(\mathbf{R}^n, \langle \cdot, \cdot \rangle)$ we have that H^* (respectively, $(\mathbf{R}^n)^*$) is isomorphic to H (respectively, to $\mathbf{R}^n$). Let $\langle E, E^* \rangle$ be a dual system of locally convex spaces. Denote by $\mathbf{K}$ a *pointed convex cone* in E, i. e., a subset of E satisfying the following properties:

1) $\mathbf{K} + \mathbf{K} \subseteq \mathbf{K}$;
2) $\lambda \mathbf{K} \subseteq \mathbf{K}$ for all $\lambda \in \mathbf{R}_+$ (the set of nonnegative real numbers); and
3) $\mathbf{K} \cap (-\mathbf{K}) = \{0\}$.

The closed convex cone

$$K^* = \{y \in E^* : \ \langle x, y \rangle \geq 0 \text{ for all } x \in \mathbf{K}\}$$

is called the *dual* of $\mathbf{K}$. The polar of $\mathbf{K}$ is $\mathbf{K}^0 = -\mathbf{K}^*$. Given the pointed convex cone $\mathbf{K} \subset E$ we denote by $\leq$ the ordering defined on E by $\mathbf{K}$, i. e., $x \leq y$ if and only if $y - x$ $\mathbf{K}$. In some situations, E is a *vector lattice* with respect to this ordering, i. e., for every pair $x, y \in E$ there exist $\inf(x, y)$ (denoted by $x \wedge y$) and $\sup(x, y)$ (denoted by $x \vee y$). We say that the bilinear form $\langle \cdot, \cdot \rangle$ is $\mathbf{K}$-local if $\langle x, y \rangle = 0$, whenever $x, y \in \mathbf{K}$ and $x \wedge y = 0$.

Let $(H, \langle \cdot, \cdot \rangle)$ be a Hilbert space and $\mathbf{K} \subset H$ a closed pointed convex cone. It is known that the projection operator onto $\mathbf{K}$, denoted by $P_{\mathbf{K}}$ is well defined [20] and for every $x \in H$, $P_{\mathbf{K}}(x)$ is the unique element of $\mathbf{K}$ satisfying $\| x - P_{\mathbf{K}}(x) \| = \min_{y \in \mathbf{K}} \| x - y \|$.

Theorem 1 *For every $x \in H$, $P_{\mathbf{K}}(x)$ is characterized by the following property:*
1) $\langle P_{\mathbf{K}}(x) - x, y \rangle \geq 0$ for all $y \in \mathbf{K}$;
2) $\langle P_{\mathbf{K}}(x) - x, x \rangle = 0$.

Proof A proof of this theorem is in [20]. □

Very useful is also the following classical *Moreau's theorem*:

Theorem 2 *If $\mathbf{K} \subset H$ is a closed convex cone and $x, y, z \in H$, then the following statements are equivalent:*
i) $z = x + y$, $x \in \mathbf{K}$, $y \in \mathbf{K}^0$ and $\langle x, y \rangle = 0$;
ii) $x = P_{\mathbf{K}}(z)$ and $y = P_{\mathbf{K}^0}(z)$.

Proof For the proof the reader is referred to [16]. □

We say that the closed pointed convex cone $\mathbf{K} \subset H$ is *isotone projection* if and only if, for every $x, y \in H$ such that $y - x \in \mathbf{K}$ we have $P_{\mathbf{K}}(y) - P_{\mathbf{K}}(x) \in \mathbf{K}$. This remarkable class of cones has been studied in several papers (see for example [13]). We say that a closed pointed convex cone $\mathbf{K} \subset H$ is a *Galerkin cone* if there exists a family of convex subcones $\{\mathbf{K}_n\}_{n \in \mathbf{N}}$ of $\mathbf{K}$ such that:
1) $\mathbf{K}_n$ is a locally compact cone, for every $n \in \mathbf{N}$;
2) if $n \leq m$, then $\mathbf{K}_n \subseteq \mathbf{K}_m$;
3) $\mathbf{K} = \overline{\cup_{n \in \mathbb{N}} \mathbf{K}_n}$.
We denote a Galerkin cone by $\mathbf{K}(\mathbf{K}_n)_{n \in \mathbf{N}}$. For more information about the application of Galerkin cones in complementarity theory, we indicate the papers [7,8,10,11,12,13] and [14].

Nonlinear Complementarity Problem

Let $\langle E, E^* \rangle$ be a dual system of locally convex spaces and $\mathbf{K} \subset E$ a pointed convex cone. Given the mapping f:$\mathbf{K} \to E^*$, the *nonlinear complementarity problem* associated to f and $\mathbf{K}$ is:

$$\text{NLCP}(f, \mathbf{K}) \quad \begin{cases} \text{find} & x_0 \in \mathbf{K} \\ \text{s.t.} & f(x_0) \in \mathbf{K}^* \\ & \text{and } \langle x_0, f(x_0) \rangle = 0. \end{cases}$$

Given two mappings f: $\mathbf{K} \to E^*$ and g: $\mathbf{K} \to E$ the *implicit complementarity problem* is:

$$\text{ICP}(f, g, \mathbf{K}) \quad \begin{cases} \text{find} & x_0 \in \mathbf{K} \\ \text{s.t.} & g(x_0) \in \mathbf{K}, \quad f(x_0) \in \mathbf{K}^* \\ & \text{and } \langle g(x_0), f(x_0) \rangle = 0. \end{cases}$$

The problem NLCP(f, $\mathbf{K}$) is important in optimization, Economics, mechanics, engineering, game theory, etc. [8]. The problem ICP(f, g, $\mathbf{K}$) was defined in relation with the study of some problems in stochastic optimal control [8]. The problems NLCP(f, $\mathbf{K}$), ICP(f, g, $\mathbf{K}$) can be solvable or unsolvable.

Solvability by Fixed Points Theorems

Given a topological space X and a mapping $f\colon X \to X$, the *fixed point problem* is to know under what conditions there exists a point $x_* \in X$ such that $f(x_*) = x_*$. This problem is studied in the Fixed Point Theory, which is a very popular domain in Nonlinear Analysis. In particular the Fixed Point Theory has been used by several authors in the study of solvability of the problem NLCP(f, $\mathbf{K}$). The results obtained in this sense, are based on some equivalences between NLCP(f, $\mathbf{K}$) and the fixed point problem. Let $(H, \langle\cdot,\cdot\rangle)$ be a Hilbert space, $\mathbf{K} \subset H$ a pointed closed convex cone and $f\colon \mathbf{K} \to H$ a mapping.

Theorem 3 *The element $x_* \in \mathbf{K}$ is a solution of the problem NLCP(f, $\mathbf{K}$) if and only if x_* is a fixed point in $\mathbf{K}$ for the mapping $T(x) = P_{\mathbf{K}}(x - f(x))$.*

Proof Suppose that $x_* \in \mathbf{K}$ is a solution of the problem NLCP(f, $\mathbf{K}$). We can show that x_* satisfies properties 1), 2), of Theorem 1 for $x = x_* - f(x_*)$.

Conversely, if $x_* \in \mathbf{K}$ and $x_* = P_{\mathbf{K}}(x_* - f(x_*))$, then since $P_{\mathbf{K}}(x_* - f(x_*))$ satisfies properties 1), 2) of Theorem 1 we deduce that x_* is a solution of the problem NLCP(f, $\mathbf{K}$). □

Theorem 4 *The problem NLCP(f, $\mathbf{K}$) has a solution if and only if the mapping $\Phi(x) = P_{\mathbf{K}}(x) - f(P_{\mathbf{K}}(x))$, defined for every $x \in H$, has a fixed point in H. Moreover, if x_0 is a fixed point of Φ, then $x_* = P_{\mathbf{K}}(x_0)$ is a solution of the problem NLCP(f, $\mathbf{K}$).*

Proof Suppose that x_0 is a fixed point for the mapping Φ, i. e.,

$$x_0 = P_{\mathbf{K}}(x_0) - f\left(P_{\mathbf{K}}(x_0)\right).$$

If we denote by $x_* = P_{\mathbf{K}}(x_0)$, we have that $x_* \in \mathbf{K}$ and $x_0 = x_* - f(x_*)$, or $x_* - x_0 = f(x_*)$. Applying Theorem 1 we can show that $f(x_*) \in \mathbf{K}^*$ and $\langle x_*, f(x_*)\rangle = 0$, i. e., x_* is a solution of the problem NLCP(f, $\mathbf{K}$).

Conversely, if $x_* \in \mathbf{K}$ is a solution of the problem NLCP(f, $\mathbf{K}$), then denoting by $x_0 = x_* - f(x_*)$ and applying Theorem 2 we deduce that $P_{\mathbf{K}}(x_0) = x_*$ and finally,

$$\begin{aligned}\Phi(x_0) &= P_{\mathbf{K}}(x_0) - f(P_{\mathbf{K}}(x_0))\\ &= x_* - f(x_*) = x_0,\end{aligned}$$

i. e., x_0 is a fixed point of Φ. □

The mapping, Φ defined in Theorem 4 was applied in complementarity theory in 1988, [7], while the mapping $\Psi(x) = x - \Phi(x)$ was used in 1992 [19]. The mapping Ψ is known as the *normal map*. By Theorem 3 the NLCP(f, $\mathbf{K}$) is transformed in a fixed point problem for the mapping T with respect to the cone $\mathbf{K}$ while, by Theorem 4 the problem NLCP(f, $\mathbf{K}$) is transformed in a fixed point problem with respect to the whole space H. Several existence results for the problem NLCP(f, $\mathbf{K}$) have been obtained by several authors using the fixed point theory and the mappings T and Φ, [3,6,7,8,10,13]. The fixed point problem associated to the mappings T and Φ has been also used in several iterative methods for solving numerically the problem NLCP (f, $\mathbf{K}$) [1,8,13,17,18] etc.

In [15] and also in [2] it is shown that the problem NLCP (f, $\mathbf{K}$) is equivalent to the following variational inequality

$$\mathrm{VI}(f, \mathbf{K}) \quad \begin{cases} \text{find} & x \in \mathbf{K} \\ \text{s.t.} & \langle f(x), y - x\rangle \geq 0 \\ & \text{for all } y \in \mathbf{K}. \end{cases}$$

Because, the fixed point theory is systematically applied to the study of variational inequalities, we have by this way another possibility to use the fixed point theory in the study of the problem NLCP(f, $\mathbf{K}$). In this sense are relevant the results obtained in [5,7,8,12] and in many other papers dedicated to the study of variational inequalities. In the study of some economical problems, we are interested to find a solution of the problem NLCP(f, $\mathbf{K}$) which is also the least element of the feasible set

$$F = \{x \in \mathbf{K}\colon\ f(x) \in \mathbf{K}^*\}.$$

This particular problem can be also studied by the fixed point theory [5,8]. If the cone $\mathbf{K}$ is an *isotone projection cone* in a Hilbert space H and if the mapping $f\colon H \to H$ satisfies some properties with respect to the ordering defined by $\mathbf{K}$, we obtain that the mappings T and Φ are monotone increasing or the difference of two monotone increasing mappings. In this case, we can apply some fixed point theorems based on the ordering, to study of the problem NLCP(f, $\mathbf{K}$). Several results in this sense are presented in [13].

The Nonlinear Complementarity Problem as a Mathematical Tool in Fixed Point Theory

The fixed point theorems on cones attracted the attention of many mathematicians. The applications of such kind of fixed point theorems are very important. We will show now how the problem NLCP(f, $\mathbf{K}$) can be used to obtain new fixed point theorems on cones.

Let H be a Hilbert space, $\mathbf{K} \subset H$ a closed pointed convex cone and h: $\mathbf{K} \to \mathbf{K}$ a mapping. The fixed point problem associated to h and $\mathbf{K}$ is:

$$\mathrm{FP}(h, \mathbf{K}) \quad \begin{cases} \text{find} & x_0 \in \mathbf{K} \\ \text{s.t.} & h(x_0) = x_0. \end{cases}$$

Consider the mapping f: $\mathbf{K} \to H$ defined by $f(x) = x - h(x)$ for all $x \in \mathbf{K}$.

Theorem 5 *The problems NLCP(f, K) and FP(h, K) are equivalent.*

Proof Suppose that x_* is a solution of the problem FP(h, $\mathbf{K}$). In this case we have $h(x_*) = x_*$, which implies that $f(x_*) = 0$. It is evident that x_* is a solution of the problem NLCP(f, $\mathbf{K}$). Conversely, if x_* is a solution of the problem NLCP(f, $\mathbf{K}$) we have that x_* is a solution of the problem VI(f, $\mathbf{K}$), i. e., $x_* \in \mathbf{K}$ and $\langle f(x_*), y - x_* \rangle \geq 0$ for all $y \in \mathbf{K}$. But $f(x_*) = x_* - h(x_*)$ and $h(x_*) \in \mathbf{K}$ (by hypothesis). This means that

$$0 \leq \langle x_* - h(x_*), x_* - h(x_*) \rangle \leq 0,$$

which implies that $h(x_*) = x_*$. □

We note that Theorem 5 was applied to obtain new fixed point theorems [7,10,11]. We cite only the following two fixed point theorems.

Theorem 6 *Let $(H, \langle \cdot, \cdot \rangle)$ be a Hilbert space ordered by a Galerkin cone $\mathbf{K}(\mathbf{K})_{n \in \mathbf{N}}$. Let T: $\mathbf{K} \to \mathbf{K}$ be a mapping satisfying the following assumptions:*
1) $T(0) \neq 0$;
2) T is a (ws)-compact operator;
3) T is ϕ-asymptotically bounded, with $\lim_{t \to \infty} \phi(t) \neq +\infty$.
Then, T has a fixed point $x_* \in \mathbf{K} \setminus \{0\}$. Moreover, x_* is the limit of a sequence $\{x_m\}_{m \in \mathbf{N}}$ where for every $m \in \mathbf{N}$, x_m is a solution of the problem NLCP(T, $\mathbf{K}_m$).

Proof The terminology and the proof is in [7]. □

Recently, a new proof for this theorem was proposed in [14].

Theorem 7 *Let $(H, \langle \cdot, \cdot \rangle)$ be a Hilbert space ordered by a Galerkin cone $\mathbf{K}(\mathbf{K})_{n \in \mathbf{N}} \subset H$. Suppose, given two continuous operators S, T: $\mathbf{K} \to H$ such that S is bounded, T is compact and $(S + T)(\mathbf{K}) \subseteq \mathbf{K}$. If the following assumptions are satisfied:*
1) $I - S$ satisfies condition $(S)_+$;
2) $I - S - T$ satisfies condition (GM),
then $S + T$ has a fixed point in $\mathbf{K}$.

Proof The terminology and the proof is in [11]. □

We note that Theorem 7 has several interesting corollaries. In [10] the reader can find other fixed point theorems for set-valued operators.

Conclusions

This interesting double relation between the *nonlinear complementarity problem* and the *fixed point theory*, can be exploited to obtain new results in complementarity theory and also in fixed point theory.

See also

- Convex-Simplex Algorithm
- Generalized Nonlinear Complementarity Problem
- Integer Linear Complementary Problem
- LCP: Pardalos–Rosen Mixed Integer Formulation
- Lemke Method
- Linear Complementarity Problem
- Linear Programming
- Order Complementarity
- Parametric Linear Programming: Cost Simplex Algorithm
- Principal Pivoting Methods for Linear Complementarity Problems
- Sequential Simplex Method
- Topological Methods in Complementarity Theory

References

1. Ahn BH (1981) Solution of nonsymmetric linear complementarity problems by iterative methods. J Optim Th Appl 33(2):175–185
2. Cottle RW (1976) Complementarity and variational problems. J Am Math Soc 19:177–208
3. Hyers DH, Isac G, Rassias TM (1997) Topics in non-linear analysis and applications. World Sci., Singapore
4. Isac G (1985) On the implicit complementarity problem in Hilbert spaces. Bull Austral Math Soc 32(2):251–260
5. Isac G (1986) Complementarity problem and coincidence equations o convex cones. Boll Unione Mat Ital Ser B 6:925–943

6. Isac G (1987) Fixed point theory and complementarity problems in Hilbert spaces. Bull Austral Math Soc 36(2):295–310
7. Isac G (1988) Fixed point theory, coincidence equations on convex cones and complementarity problem. Contemp Math 72:139–155
8. Isac G (1992) Complementarity problems. Lecture Notes Math, vol 1528. Springer, Berlin
9. Isac G (1993) Tihonov's regularization and the complementarity problem in Hilbert spaces. J Math Anal Appl 174(1):53–66
10. Isac G (1995) Fixed point theorems on convex cones, generalized pseudo-contractive mappings and the complementarity problem. Bull Inst Math Acad Sinica 23(1):21–35
11. Isac G (1995) On an Altman type fixed point theorem on convex cones. Rocky Mountain J Math 25(2):701–714
12. Isac G, Goeleven D (1993) Existence theorems for the implicit complementarity problem. Internat J Math and Math Sci 16(1):67–74
13. Isac G, Neméth AB (1990) Projection methods, isotone projection cones and the complementarity problem. J Math Anal Appl 153(1):258–275
14. Jachymski J (1994) On Isac's fixed point theorem for self-maps of a Galerkin cone. Ann Sci Math Québec 18(2):169–171
15. Karamardian S (1971) Generalized complementarity problem. J Optim Th Appl 8:161–168
16. Moreau J (1962) Décomposition orthogonale d'un espace hilbertien selon deux cones mutuellement polaires. C R Acad Sci Paris 225:238–240
17. Noor MA (1988) Fixed point approach for complementarity problems. J Math Anal Appl 133:437–448
18. Noor MA (1988) Iterative methods for a class of complementarity problems. J Math Anal Appl 133:366–382
19. Robinson SM (1992) Normal maps induced by linear transformations. Math Oper Res 17(3):691–714
20. Zarantonello EH (1971) Projection on convex sets in Hilbert space and spectral theory. In: Zarantonello EH (ed) Contributions to Nonlinear Functional Analysis. Acad. Press, New York, pp 237–424

Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques

QING CHEN, EVANGELOS TRIANTAPHYLLOU
Department Industrial and Manufacturing Systems Engineering, Louisiana State University, Baton Rouge, USA

MSC2000: 90C29

Article Outline

Keywords
Extraction of Relative Priorities from Complete Pairwise Matrices
The Eigenvalue Approach
Optimization Approaches
Considering the Human Rationality Factor
Matrices with Missing Comparisons
Estimating Missing Comparisons
Using Connecting Paths
Revised Geometric Mean Method (RGM)
Least Squares Formulation
Determining the Comparison to Elicit Next
Conclusions
See also
References

Keywords

Pairwise comparisons; Data elicitation; Multicriteria decision making; MCDM; Scale; Analytic hierarchy process; AHP; Consistent judgment matrix; Eigenvalue; Eigenvector; Least squares problem; Incomplete judgments

One of the most crucial steps in many multicriteria decision making methods (MCDM) is the accurate estimation of the pertinent data [18]. Very often these data cannot be known in terms of absolute values. For instance, what is the worth of the ith alternative in terms of a political impact criterion? Although information about questions like the previous one is vital in making the correct decision, it is very difficult, if not impossible, to quantify it correctly. Therefore, many decision making methods attempt to determine the *relative* importance, or weight, of the alternatives in terms of each criterion involved in a given decision making problem.

Consider the case of having a single decision criterion and a set of n alternatives, denoted as A_i (for $i = 1, \ldots, n$). The decision maker wants to determine the relative performance of these alternatives in terms of a single criterion. An approach based on *pairwise comparisons* which was proposed by T.L. Saaty [11], and [12] has long attracted the interest of many researchers, because both of its easy applicability and interesting mathematical properties. Pairwise comparisons are used to determine the relative im-

portance of each alternative in terms of each criterion.

In that approach the decision maker has to express his/her opinion about the value of one single pairwise comparison at a time. Usually, the decision maker has to choose his/her answer among 10–17 discrete choices. Each choice is a linguistic phrase. Some examples of such linguistic phrases when two concepts, **A** and **B** are considered might be: '**A** is more important than **B**', or '**A** is of the same importance as **B**', or '**A** is a little more important than **B**', and so on. When one focuses directly on the *data elicitation* issue one may use linguistic statements such as 'How much more does alternative **A** belong to the set **S** than alternative **B**'?

The main problem with the pairwise comparisons is how to quantify the *linguistic choices* selected by the decision maker during the evaluation of the pairwise comparisons. All the methods which use the pairwise comparisons approach eventually express the qualitative answers of a decision maker into some numbers.

Pairwise comparisons are quantified by using a *scale*. Such a scale is nothing but an one-to-one mapping between the set of discrete linguistic choices available to the decision maker and a discrete set of numbers which represent the importance, or weight, of the previous linguistic choices. There are two major approaches in developing such scales. The first approach is based on the *linear scale* proposed by Saaty [12] as part of the *analytic hierarchy process* AHP. The second approach was proposed by F. Lootsma [8,9,10] and determines *exponential scales*. Both approaches depart from some psychological theories and develop the numbers to be used based on these psychological theories. For an extensive study of the scale issue, see [18] and [19].

In this article we examine three problems related to the use of pairwise comparisons for data elicitation in MCDM. The first problem is how to combine the $n(n-1)/2$ comparisons needed to compare n entities (alternatives or *criteria*) under a given goal and extract their relative preferences. This subject was extensively studied in [21] and it is briefly discussed in the second section. The second problem in this article is how to estimate *missing comparisons*. The third problem is how to select the order for eliciting the comparisons and determine whether all comparisons are needed. These problems are examined in detail in the following sections.

Extraction of Relative Priorities from Complete Pairwise Matrices

Let $A_1, \ldots, A_n$ be n alternatives (or criteria or, in general, concepts) to be compared. We are interested in evaluating the relative preference values of the above concepts. Saaty [11,12,14] proposed to use a matrix A of rational numbers taken from the set {1/9, 1/8, 1/7, …, 1, …, 9}. Each entry of the above matrix A represents a *pairwise judgment*. Specifically, the entry a_{ij} denotes the number that estimates the relative preference of element A_i when it is compared with element A_j. Obviously, $a_{ij} = 1/a_{ji}$ and $a_{ii} = 1$. That is, the matrix is reciprocal.

The Eigenvalue Approach

Let us first examine the case in which it is possible to have perfect values a_{ij}. In this case it is $a_{ij} = W_i/W_j$ (W_s denotes the actual value of element s) and the previous reciprocal matrix A is *consistent*. That is:

$$a_{ij} = a_{ik} \times a_{kj} \quad \text{for } i, j, k = 1, \ldots, n, \tag{1}$$

where n is the number of elements in the comparison set. It can be proved [12] that the matrix A has rank 1 with n to be its nonzero eigenvalue. Thus, we have:

$$Ax = nx, \tag{2}$$

where x is an eigenvector. From the fact that $a_{ij} = W_i/W_j$, the following are obtained:

$$\sum_{j=1}^{n} a_{ij} W_j = \sum_{j=1}^{n} W_i = nW_i, \quad i = 1, \ldots, n, \tag{3}$$

or

$$AW = nW. \tag{4}$$

Equation (4) states that n is an eigenvalue of A with W being a corresponding eigenvector. The same equation also states that in the *perfectly consistent case* (i. e., when $a_{ij} = a_{ik} \times a_{kj}$ for all possible triplets), the vector W, with the relative preferences of the elements $A_1, \ldots, A_n$, is the principal right eigenvector (after normalization) of A.

In the nonconsistent case (which is the most common) the pairwise comparisons are not perfect, that is, the entry a_{ij} might deviate from the real ratio W_i/W_j (i. e., from the ratio of the real relative preference values W_i and W_j). In this case, the previous expression

(1) does not hold for all possible combinations. Now the new matrix A can be considered as a perturbation of the previous consistent case. When the entries a_{ij} change slightly, then the eigenvalues change in a similar fashion [12]. Moreover, the maximum eigenvalue is close to n (actually greater than n) while the remaining eigenvalues are close to zero. Thus, in order to find the relative preferences in the nonconsistent cases, one should find an eigenvector that corresponds to the maximum eigenvalue $\lambda_{\max}$. That is to say, to find the principal right eigenvector W that satisfies:

$$AW = \lambda_{\max} W \quad \text{where } \lambda_{\max} = n.$$

Saaty estimates the principal right eigenvector W by multiplying the entries in each row of A together and taking the nth root (n being the number of the elements in the comparison set). Since we desire to have values that add up to 1, we normalize the previously found vector by the sum of the above values. If we want to have the element with the highest value to have a relative preference value equal to 1, we divide the previously found vector by the highest value.

Under the assumption of *total consistency*, if the judgments are gamma distributed (something that Saaty claims to be the case), the principal right eigenvector of the resultant reciprocal matrix A is Dirichlet distributed. If the assumption of total consistency is relaxed, then L.G. Vargas [23] proved that the hypothesis that the principal right eigenvector follows a Dirichlet distribution is accepted if the consistency ratio is 10% or less.

The *consistency ratio* (CR) is obtained by first estimating $\lambda_{\max}$. Saaty estimates $\lambda_{\max}$ by adding the columns of matrix A and then multiplying the resulting vector with the vector W. Then, he uses what he calls the consistency index (CI) of the matrix A. He defined CI as follows:

$$\mathrm{CI} = \frac{\lambda_{\max} - n}{n-1}.$$

Then, the consistency ratio CR is obtained by dividing the CI by the random consistency index (RCI) as given in Table 1. Each RCI is an average random consistency index derived from a sample of size 500 of randomly generated reciprocal matrices with entries from the set $\{1/9, 1/8, 1/7, \ldots, 1, \ldots, 9\}$ to see if its CI is 10% or less. If the previous approach yields a CR greater than 10%, then a reexamination of the pairwise judgments is recommended until a CR less than or equal to 10% is achieved.

Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques, Table 1
RCI values for sets of different order n [12]

n	1	2	3	4	5	6	7	8	9
RCI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45

Optimization Approaches

A.T.W. Chu, R.E. Kalaba and K. Spingarn [2] claimed that given the data a_{ij}, the values W_i to be estimated are desired to have the property:

$$a_{ij} \approx \frac{W_i}{W_j}. \tag{5}$$

This is reasonable since a_{ij} is meant to be the estimation of the ratio W_i/W_j. Then, in order to get the estimates for the W_i given the data a_{ij}, they proposed the following constrained optimization problem:

$$\begin{cases} \min & S = \sum_{i=j}^{n} \sum_{j=i}^{n} (a_{ij} w_j - w_i)^2, \\ \text{s.t.} & \sum_{i=j}^{n} W_i = 1, \\ & W_i > 0 \quad \text{for } i = 1, \ldots, n. \end{cases} \tag{6}$$

They also provide an alternative expression S_1 that is more difficult to solve numerically. That is,

$$S_1 = \sum_{i=j}^{n} \sum_{j=i}^{n} \left(a_{ij} - \frac{W_j}{W_i} \right)^2. \tag{7}$$

In [3] a variation of the above *least squares* formulation is proposed. For the case of only one decision maker it recommends the following models:

$$\log a_{ij} = \log W_i - \log W_j + \psi_2(W_i, W_j)\varepsilon_{ij}, \tag{8}$$

$$a_{ij} = \frac{W_i}{W_j} + \psi_2(W_i, W_j)\varepsilon_{ij}, \tag{9}$$

where W_i and W_j are the true (and hence unknown) relative preferences; $\psi_1(X, Z)$ and $\psi_2(X, Z)$ are given positive functions (where $X, Z > 0$). The random errors ε_{ij} are assumed independent with zero mean and unit variance. Using these two assumptions one is able to calculate the variance of each individual estimated relative preference. However, is fails to give a way of selecting the appropriate positive functions. In the second example, presented later, a sample problem which originates in [11] and later in [3] is solved for different functions ψ_1, ψ_2 using this method.

Considering the Human Rationality Factor

According to the *human rationality assumption* [21] the decision maker is a rational person. Rational persons are defined here as individuals who try to minimize their regret [15], to minimize losses, or to maximize profit [24]. In the relative preference evaluation problem, *minimization of regret*, losses, or maximization of profit could be interpreted as the effort of the decision maker to minimize the errors involved in the pairwise comparisons.

As it is stated in previous paragraphs, in the inconsistent case the entry a_{ij} of the matrix A is an estimation of the real ratio W_i/W_j. Since it is an estimation, the following is true:

$$a_{ij} = \left(\frac{W_i}{W_j}\right) d_{ij}, \quad i, j = 1, \ldots, n. \tag{10}$$

In the above relation d_{ij} denotes the deviation of a_{ij} from being an accurate judgment. Obviously, if $d_{ij} = 1$, then the a_{ij} was perfectly estimated. From the previous formulation we conclude that the errors involved in these pairwise comparisons are given by:

$$\varepsilon_{ij} = d_{ij} - 1.00,$$

or after using (10), above:

$$\varepsilon_{ij} = a_{ij}\left(\frac{W_j}{W_i}\right) - 1.00. \tag{11}$$

When a comparison set contains n elements, then Saaty's method requires the estimation of the following $n(n-1)/2$ pairwise comparisons:

$$\begin{aligned} &\frac{W_2}{W_1}, \ldots, \frac{W_n}{W_1}, \\ &\frac{W_3}{W_2}, \ldots, \frac{W_n}{W_2}, \\ &\vdots \\ &\frac{W_{n-1}}{W_n}. \end{aligned} \tag{12}$$

The corresponding $n(n-1)/2$ errors are (after using relations (11) and (12)):

$$\varepsilon_{ij} = a_{ij}\left(\frac{W_j}{W_i}\right) - 1.00, \quad i, j = 1, \ldots, n, \text{ and } j > 1. \tag{13}$$

Since the W_i are relative preferences that add up to 1, the following relation (14) should also be satisfied:

$$\sum_{i=1}^{n} W_i = 1.00. \tag{14}$$

Apparently, since the W_i represent relative preferences we also have:

$$W_i > 0, \quad i = 1, \ldots, n. \tag{15}$$

Relations (13) and (14), when the data are consistent (i. e., all the errors are equal to zero), can be written as follows:

$$BW = b. \tag{16}$$

The vector b has zero entries everywhere except the last one that is equal to 1, and the matrix B has the following form (blank entries represent zeros):

$$B = \begin{bmatrix} 1 & 2 & 3 & \cdots & n & \\ -1 & a_{1,2} & & & & 1 \\ -1 & & a_{1,3} & & & 2 \\ \vdots & & & \ddots & & \vdots \\ -1 & & & & a_{1,n} & n-1 \\ & -1 & a_{2,3} & & & 1 \\ & \vdots & & \ddots & & \vdots \\ & -1 & & & a_{2,n} & n-2 \\ & & \ddots & & & \vdots \\ & & & & a_{n-1,n} & 1 \\ 1 & 1 & 1 & \cdots & 1 & \end{bmatrix}.$$

The *error minimization* issue is interpreted in many cases (regression analysis, linear least squares problem) as the minimization of the *sum of squares* of the residual vector: $r = b - BW$ [16]. In terms of formulation (15) this means that in a real life situation (i. e., when errors are not zero any more) the real intention of the decision maker is to minimize the expression:

$$f^2(x) = \|b - BW\|, \tag{17}$$

which, apparently, expresses a typical linear least squares problem.

If we use the notation described previously, then the quantity (6) which is minimized in [2] becomes:

$$S = \sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij}W_j - W_i)^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}(\varepsilon_{ij}W_i)^2$$

and the alternative expression (7) becomes:

$$S_1 = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij}\frac{W_j}{W_i}\right)^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\varepsilon_{ij}\frac{W_i}{W_j}\right)^2.$$

Clearly, both expressions are too complicated to reflect, in a reasonable way, the intentions of the decision maker.

The models proposed in [3] are closer to the one developed under the human rationality assumption. The only difference is that instead of the relations:

$$\log a_{ij} = \log w_i - \log W_j + \psi_1(W_i, W_j)\varepsilon_{ij}$$

and

$$a_{ij} = \frac{W_i}{W_j} + \psi_2(W_i, W_j)\varepsilon_i j,$$

the following simpler expression is used:

$$a_{ij} = \frac{W_i}{W_j}d_{ij}, \tag{18}$$

or

$$a_{ij} = \frac{W_i}{W_j} \times (\varepsilon_{ij} + 1.00).$$

However, as the second example illustrates, the performance of this method is greatly dependent on the selection of the $\psi_1(X, Z)$ or $\psi_2(X, Z)$ functions. Now, however, these functions are further modified by (17).

Example 1

Let us assume that the following is the matrix with the pairwise comparisons for a set of four elements:

$$A = \begin{bmatrix} 1 & 2/1 & 1/5 & 1/9 \\ 1/2 & 1 & 1/8 & 1/9 \\ 5/1 & 8/1 & 1 & 1/4 \\ 9/1 & 9/1 & 4/1 & 1 \end{bmatrix}.$$

Using the methods presented in previous sections we can see that

$$\lambda_{\max} = 4.226,$$
$$\mathrm{CI} = \frac{4.226 - 4}{4 - 1} = 0.053,$$
$$\mathrm{CR} = \frac{\mathrm{CI}}{0.90} = 0.0837 < 0.10.$$

The formulation (15) that corresponds to this example is as follows:

$$\begin{bmatrix} -1 & 2/1 & 0.0 & 0 \\ -1 & 0.0 & 1/5 & 0 \\ 1 & 0.0 & 0 & 1/9 \\ 0.0 & -1 & 1/8 & 0 \\ 0.0 & -1 & 0 & 1/9 \\ 0.0 & 0.0 & -1 & 1/4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1.0 \end{bmatrix}.$$

The vector V that solves the above least squares problem is calculated to be:

$$V = (0.065841\ 0.039398\ 0.186926\ 0.704808).$$

Hence, the sum of squares of the residual vector components is 0.003030. The average squared residual for

Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques, Table 2
Data for the second example

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
(1)	1	4	9	6	6	5	5
(2)	1/4	1	7	5	5	3	5
(3)	1/9	1/7	1	1/5	1/5	1/7	1/5
(4)	1/6	1/5	5	1	1	1/3	1/3
(5)	1/6	1/5	5	1	1	1/3	1/3
(6)	1/5	1/3	7	3	3	1	2
(7)	1/5	1/4	5	3	3	1/2	1

Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques, Table 3
Comparison of the relative preferences for the data in Table 2

	elements in set							
method used	(1)	(2)	(3)	(4)	(5)	(6)	(7)	Ave. residual
Saaty eigenvector method	0.429	0.231	0.021	0.053	0.053	0.119	0.095	0.134
Power method eigenvector	0.427	0.230	0.021	0.052	0.052	0.123	0.094	0.135
Chu's method	0.487	0.175	0.030	0.059	0.059	0.104	0.085	0.097
Federov Model 1 with $\psi_1 = 1$	0.422	0.232	0.021	0.052	0.052	0.127	0.094	0.138
Federov Model 2 with $\psi_2 = 1$	0.386	0.287	0.042	0.061	0.061	0.088	0.075	0.161
Federov Model 2 with $\psi_2 = \lvert W_i - W_j \rvert$	0.383	0.262	0.032	0.059	0.059	0.122	0.083	0.152
Federov Model 2 with $\psi_2 = W_i/W_j$	0.047	0.229	0.021	0.051	0.051	0.120	0.081	0.130
Least squares method under the HR assumption	0.408	0.147	0.037	0.054	0.054	0.080	0.066	0.082

this problem is $0.003030/(4(4-1)/2) + 1 = 0.000433$; that is, the average residual is $\sqrt{0.000433} = 0.020806$.

Example 2 The second example uses the same data used originally in [11], and later in [2] and [3]. These data are presented in Table 2.

Table 3 presents a summary of the results (as found in the corresponding references) when the methods described in the subsections above are used. The *power method* for deriving the eigenvector was applied as presented in [7]. In the last row of Table 2 are the results obtained by using the least square method under the human rationality assumption (HR).

As it is shown in the last column of Table 3, the performance of each method is very different as far the mean residual is concerned. The results also illustrate how critical is the role of the functions $\psi_1(X, Z)$ and $\psi_2(X, Z)$ in the method of [3]. The mean residual obtained by using the least squares method under the human rationality assumption is the smallest one by 16%.

Matrices with Missing Comparisons

For one to evaluate n concepts, normally all the required $n(n-1)/2$ pairwise comparisons are needed. However, for large numbers of concepts to be compared, the decision maker may become quite bored, tired and inattentive with assigning the values to the comparisons as time is going on, which may easily lead to erroneous judgments. Moreover, the time spent to elicit all the comparisons for a judgment matrix may be unaffordable. Also the decision maker may not be sure about the values of some comparisons and thus may not want to make a direct evaluation of them. In cases like the previous ones, the decision maker may wish to stop the process and then try to derive the relative preferences from an incomplete pairwise comparison (judgment) matrix.

Given an incomplete pairwise comparison matrix, there are two central and closely interrelated problems. The first problem is how to estimate the missing comparisons. The second problem is which comparison to evaluate next. In other words, if the decision maker wishes to estimate a few extra comparisons (from the remaining undetermined ones) how should the next comparison be selected? Should it be selected randomly or according to some rule (to be determined)? Next, we study the first of these two closely related problems.

Estimating Missing Comparisons

Using Connecting Paths

Suppose that $X_{i,j}$ is a missing comparison to be estimated. Next, also assume that there are two known comparisons $a_{i,k}$ and $a_{k,j}$ for some index k. In the perfectly consistent case the following relationship should be true:

$$X_{i,j} = a_{i,k} \times a_{k,j}.$$

In the more general inconsistent case, the $X_{i,j}$ value can be approximated by the product $a_{i,k} \times a_{k,j}$. In [5], and [6] the pair $a_{i,k}$ and $a_{k,j}$ is called an *elementary connecting path* connecting the missing comparison $X_{i,j}$. Obviously, given a missing comparison, more than one such connecting path may exist (i. e., if there are more than one k indexes which satisfy the above relationship). Moreover, it is also possible to have connecting paths comprised by more than two known comparisons (i. e., paths of size larger than 2). The general structure of a connecting path of size r, denoted as CP_r, has the following form:

$$\mathrm{CP}_r: \quad X_{i,j} = a_{i,k1} \times a_{k1,k2} \times \cdots \times a_{kr,j},$$

for $i, j, k1, \ldots, kr = 1, \ldots, n$, $1 \le r \le n-2$.

According to P.T. Harker [5,6] the value of the missing comparison $X_{i,j}$ should be equal to the geometric mean of all connecting paths related to this missing comparison. That is, the following should be true:

$$X_{ij} = \sqrt[q]{\prod_{r=1}^{q} \mathrm{CP}_r}.$$

In the previous expression it is assumed that there are q such connecting paths. For the above reasons, this method is known as the *geometric mean method* for estimating missing comparisons.

A method alternative to the geometric means method is to express the missing comparisons in terms of the arithmetic averages of all related connecting paths and some error terms. In this way, one can also introduce error terms on consistency relations which are defined on pairs of missing comparisons (for more details, please see [1]). A natural objective then, could be to minimize the sum of the absolute terms of all these error terms (which can be of any sign). That is, the above consideration leads to the formulation of a linear programming (LP) problem. A similar approach is presented in [17] (in which the path problem does not occur).

However, there is a serious drawback with any method which attempts to use connecting paths. The number of connecting paths may be astronomically large, rendering any such method computationally intractable. For instance, for a comparison matrix of dimension of six, the number of possible connecting paths to be considered might be equal to 64, while in a case of dimension equal to ten, the number of paths may become equal to 109,600. As a result, some alternative approaches have been developed. The revised geometric means method (or RGM) method and a least squares formulation are two such methods and are discussed next.

Revised Geometric Mean Method (RGM)

An alternative approach to the use of connecting paths, is to convert the incomplete judgement matrix into a transformed matrix and then determine its principal right eigenvector. This was proposed by Harker [4] and it is best illustrated by means of an example.

Suppose that the following is an incomplete judgement matrix of order 3 (taken from [4]).

$$A_0 = \begin{bmatrix} 1 & 2 & - \\ 1/2 & 1 & 2 \\ - & 1/2 & 1 \end{bmatrix}.$$

One can replace the missing elements (denoted by $-$) by the corresponding ratios of weights. Therefore, the previous matrix becomes:

$$A_1 = \begin{bmatrix} 1 & 2 & w_1/w_3 \\ 1/2 & 1 & 2 \\ w_3/w_1 & 1/2 & 1 \end{bmatrix}.$$

That is, the missing comparison $X_{1,3}$ was replaced by the ratio w_1/w_3 (similar for the reciprocal entry $X_{3,1}$). Next observe that the product $A_1\ W$ is equal to:

$$A_1 W = \begin{bmatrix} 1 & 2 & w_1/w_3 \\ 1/2 & 1 & 2 \\ w_3/w_1 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 2w_1 + 2w_2 \\ w_1/2 + w_2 + 2w_3 \\ w_2/2 + 2w_3 \end{bmatrix}$$

The same result can also be obtained if one considers the matrix C, given as follows:

$$C = \begin{bmatrix} 2 & 2 & 0 \\ 1/2 & 1 & 2 \\ 0 & 1/2 & 2 \end{bmatrix},$$

that is, matrix C satisfies the relationship

$$A_1 W = CW.$$

Therefore, the desired relative preferences (i. e., the entries of vector W) can be determined as the principal right eigenvector of the new matrix C. This is true because:

$$A_1 W = CW = \lambda W.$$

In general, the entries of matrix C can be determined from the entries of an incomplete judgement matrix A_0 as follows (where $c_{i,j}$ and $a_{i,j}$ are the elements of the matrices C and A_0, respectively):

$$c_{i,i} = 1 + m_i$$

and for $i \neq j$:

$$c_{i,j} = \begin{cases} a_{i,j} & \text{if } a_{i,j} \text{ is a positive number,} \\ 0 & \text{otherwise,} \end{cases}$$

where m_i is the number of unanswered questions in the ith row of the incomplete comparison matrix.

Next, the elements of the W vector can be determined by using one of the methods presented in the second section.

Least Squares Formulation

This formulation is a natural extension of the formulation discussed earlier in the section on the HR factor. The only difference is that in relations (12) one should only consider known comparisons. This, as a result, implies that the new matrix B (as defined earlier) should not have rows which would correspond to missing comparisons. Finally, observe that in order to solve the least squares problem given as (16), one has to calculate the vector W as follows:

$$W = (B^\top B)^{-1} B^\top b,$$

where $B^\intercal$ stands for the transpose of B.

In [1] the revised geometric means and the previous least squares method were tested on random problems. First, a complete judgment matrix was determined. These matrices, in general, were slightly inconsistent. They were derived according to the procedures used in [20,22], and [19]. Then, some comparisons were randomly removed and set as missing. Then, the previous two methods were applied on the incomplete judgment matrix and the missing comparisons were estimated. The estimated matrix was used to derive a ranking of the compared entities. This ranking was compared with the ranking derived when the original complete judgment matrix is used. In these computational experiments it was found that the two estimation methods for missing comparisons performed almost in a similar manner. This manner was different for matrices of different order and various percentages of missing comparisons. More details on these issues can be found in [1].

Determining the Comparison to Elicit Next

Suppose that the decision maker has determined some of the $n(n-1)/2$ comparisons when a set of n entities is considered for extracting relative preferences. Next assume that the decision maker wishes to proceed with only a few additional comparisons and not determine the entire judgment matrix. The question we examine at this point is which ones the additional comparisons should be. To be more specific, the question we consider is best stated as follows: Given an incomplete judgment matrix, and the option to elicit just some additional comparisons, then which one should be the comparison to elicit next?

One obvious approach is to select the next comparison just randomly among the missing ones. This problem was examined by Harker in [5] and [6]. Harker focused his attention on how to determine which comparison, among the missing ones, is the most *critical* one. He determined as the most critical one, to be the comparison which would have the largest impact (when the appropriate derivatives are considered) on the vector W.

He observed that the largest absolute gradient (i. e., the largest partial derivative) means that a unit change of the specific missing comparison brings out the biggest change on the vector W. Therefore, he asserted, that the missing comparison related to the largest absolute gradient should be the most critical one and therefore, the one to evaluate next. Then, the following formula calculating the largest absolute gradient can be used to choose the most critical comparison index (i, j):

$$(i, j) = \arg \max_{(k,l)\epsilon Q} \left\| \frac{\partial x(A)}{\partial_{k,l}} \right\|_{\infty},$$

where Q is the set of missing comparisons and $\| \cdot \|_{\infty}$ is the Tchebyshev norm. The most critical comparison index (i, j) is determined by the maximum norm of the vector of $\partial x(A)/\ \partial_{k,1}$ which corresponds to all missing comparisons.

The previous approach is intuitively plausible but computationally non trivial. Moreover, its effectiveness had not been addressed until recently. In [1] Harker's derivatives approach was tested versus a method which randomly selects the next comparison to elicit. The test problems were generated similarly to the ones described at the end of the previous section. The two methods were also tested in a similar manner as before. To our surprise, the two methods performed in a similar manner. Therefore, the obvious conclusion is that one does not have to implement the more complex derivatives method. It is sufficient to select the next comparison just randomly. Of course, the more comparisons are selected, the better is for the accuracy of the final results. Since the order of comparisons seems not to have an impact, the best strategy is to select as the next comparison the one which is easier for the decision maker to elicit.

Conclusions

Deriving the data for MCDM problems is an approach which requires trade-offs. Thus, it should not come as a surprise that optimization can be used at various stages of this crucial phase in solving many MCDM problems. The previous analysis of some key problems signifies that optimization becomes more critical as the size of the decision problem increases.

Finally, it should be stated here that an in depth analysis of many key issues in multicriteria decision making theory and practice is provided in [18].

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Chen Q, Triantaphyllou E, Zanakis S (2001) Estimating missing comparisons and selecting the next comparison to elicit in MCDM. Working Paper Dept Industrial Engin Louisiana State Univ., http://www.imse.lsu.edu/vangelis
2. Chu ATW, Kalaba RE, Spingarn K (1979) A comparison of two methods for determining the weights of belonging to fuzzy sets. J Optim Th Appl 27(4):321–538
3. Federov VV, Kuzmin VB, Vereskov AI (1982) Membership degrees determination from Saaty matrix totalities. In: Gupta MM, Sanchez E (eds) Approximate Reasoning in Decision Analysis. North-Holland, Amsterdam, 23–30
4. Harker PT (1987) Alternative modes of questioning in the analytic hierarchy process. Math Model 9(3–5):353–360
5. Harker PT (1987) Derivatives of the Perron root of a positive reciprocal matrix: With application to the analytic hierarchy process. Appl Math Comput 22:217–232

6. Harker PT (1987) Incomplete pairwise comparisons in the analytic hierarchy process. Math Model 9(11):837–848
7. Kalaba R, Spingarn K (1979) Numerical approaches to the eigenvalues of Saaty's matrices for fuzzy sets. Comput Math Appl 4
8. Lootsma FA (1988) Numerical scaling of human judgment in pairwise-comparison methods for fuzzy multi-criteria decision analysis: Mathematical Models for Decision Support. In: NATO ASI F: Computer and System Sci, vol 48. Springer, Berlin, pp 57–88
9. Lootsma FA (1990) The French and the American school in multi-criteria decision analysis. Rech Oper Res 24(3):263–285
10. Lootsma FA (1991) Scale sensitivity and rank preservation in a multiplicative variant of the AHP and SMART. Techn Report Fac Techn Math and Informatics Delft Univ Techn, no. 91–67
11. Saaty TL (1977) A scaling method for priorities in hierarchical structures. J Math Psych 15(3):234–281
12. Saaty TL (1980) The analytic hierarchy process. McGraw-Hill, New York
13. Saaty TL (1983) Priority setting in complex problems. IEEE Trans Engin Management EM–30(3):140–155
14. Saaty TL (1994) Fundamentals of decision making and priority theory with the analytic hierarchy process. VI, RWS Publ., Pittsburgh, PA
15. Simon HA (1961) Models of man, 2nd edn. Wiley, New York
16. Stewart SM (1973) Introduction to matrix computations. Acad. Press, New York
17. Triantaphyllou E (1995) Linear programming based decomposition approach in evaluating priorities from pairwise comparisons and error analysis. J Optim Th Appl 84(1):207–234
18. Triantaphyllou E (2000) Multi-criteria decision making methods: A comparative study. Kluwer, Dordrecht
19. Triantaphyllou E, Lootsma FA, Pardalos PM, Mann SH (1994) On the evaluation and application of different scales for quantifying pairwise comparisons in fuzzy sets. J Multi-Criteria Decision Anal, 3:133–155
20. Triantaphyllou E, Mann SH (1994) A computational evaluation of the AHP and the revised AHP when the eigenvalue method is used under a continuity assumption. Comput and Industrial Eng 26(3):609–618
21. Triantaphyllou E, Pardalos PM, Mann SH (1990) A minimization approach to membership evaluation in fuzzy sets and error analysis. J Optim Th Appl 66(2):275–287
22. Triantaphyllou E, Sanchez A (1997) A sensitivity analysis approach for some deterministic multi-criteria decision-making methods. Decision Sci 28(1):151–194
23. Vargas LG (1982) Reciprocal matrices with random coefficients. Math Model 3:69–81
24. Write C, Tate MD (1973) Economics and systems analysis: Introduction for public managers. Addison-Wesley, Reading, MA

Evacuation Networks

J. MacGregor Smith
Department of Mechanical and Industrial Engineering,
University of Massachusetts,
Amherst, Massachusetts

MSC2000: 90B15

Article Outline

Abstract
Keywords
Introduction
  Purpose of Chapter
  Outline of Chapter
Modelling Fundamentals
  Representation Stage
  Analysis Step
  Synthesis Step
Mathematical Models
  Set Partitioning Model
  Congestion Models
  Erlang Loss/Delay Networks
Algorithms
  K-shortest Paths
  Other Algorithms
Summary and Conclusion
References

Abstract

Planning and design of evacuation networks is both a complex and critically important optimization problem for a number of emergency situations. One particularly critical class of examples concerns the emergency evacuation of chemical plants, high-rise buildings, and naval vessels due to fire, explosion or other emergencies. The problem is compounded because the solution must take into account the fact that human occupants may panic during the evacuation, therefore, there must be a well-defined set of evacuation routes in order to minimize the sense of panic and at the same time create safe, effective routes for evacuation. The problem is a highly transient, stochastic, nonlinear, combinatorial optimization programming problem. We focus on evacuation networks where congestion is a significant problem.

Keywords

Combinatorial optimization; Evacuation networks; Congestion

Introduction

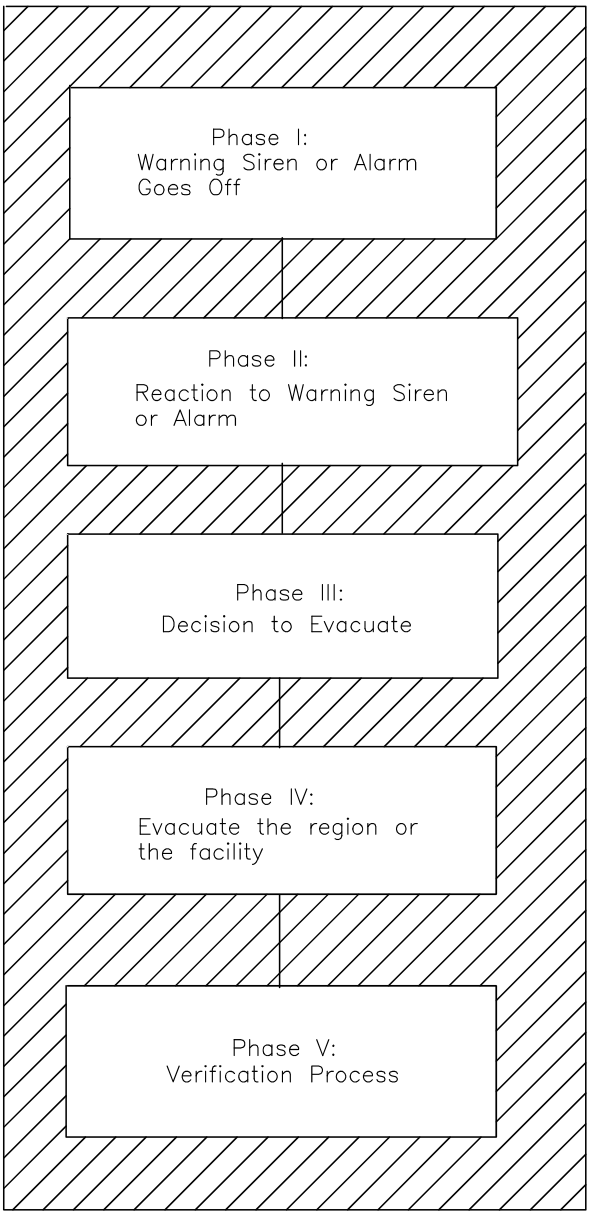
Evacuation is one of the most perilous, pernicious, and persistent problems faced by humanity. Hurricanes, fires, earthquakes, explosions and other natural and man-made disasters happen on almost a daily basis throughout the world. How can we safely evacuate a collection of occupants within an affected region or facility is the fundamental problem faced in evacuation.

Purpose of Chapter

The purpose of this chapter is to both introduce to the reader the problem of evacuation and its manifest nature, and also suggest some alternative approaches to optimize this process. That life-threatening evacuations happen as often as they do is somewhat surprising. That people often do not know how to safely evacuate in time of need is a sad reality. That people must help people plan for evacuation is one of the most important activities of a research scientist.

Outline of Chapter

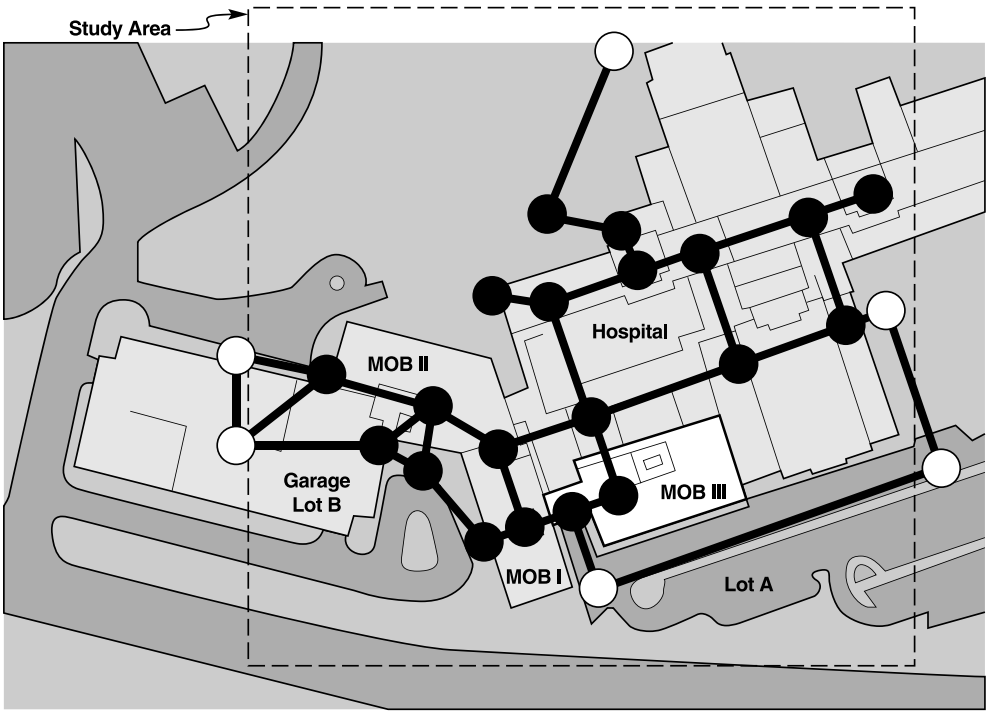
In this chapter we first introduce the problem in Sect. "Modelling Fundamentals" and then also describe our fundamental modelling 3-step methodology. In Sect. "Mathematical Models", we array the number of different of static and dynamic approaches to this problem and present our general approach which has guided our research on the problem. Finally, in Sect. "Algorithms" we discuss the algorithmic approaches to the problem where we capture the congested flow of occupants in the network and attempt to define the safest evacuation routes trading off the different objective performance measures in the network.



Evacuation Networks, Figure 1
Processes for an evacuation

Modelling Fundamentals

The process of an evacuation is captured in the simplified flow chart of Fig. 1. There are essentially five phases which underly the evacuation process. The first and foremost is a warning bell or siren signaling the occupant population to leave. Unfortunately, one must react to the warning and recognize the problem at hand, so there is often a great deal of uncertainty associated with the second phase. Thirdly, after the warning is taken seriously, the occupants must decide to evacuate. The first three phases are highly uncertain and transient. Once the occupants decide to evacuate, the general evacuation process gets underway and this is where the evacuation plans should be followed. Finally, there is a verification phase, were one must account for all the occupants to ensure their safe arrival at the destina-

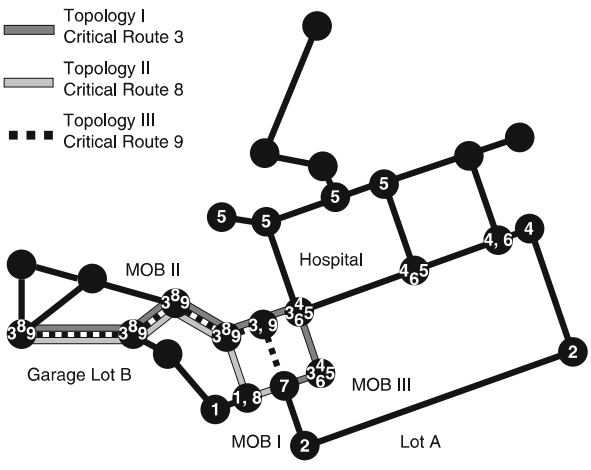


Evacuation Networks, Figure 2
Evacuation plan for a hospital complex

tion. As a constructive framework for this chapter on evacuation networks, we establish that the modelling of evacuation problems has three fundamental steps: **Step 1.0 Representation:** *How should a region, e. g. Fig. 2 or facility be represented or modelled?* **Step 2.0 Analysis:** *Given the model, how should analyze the evacuation of the occupants, i. e. a deterministic or stochastic evacuation process? What performance measures are crucial to measuring performance of the evacuation? and* **Step 3.0 Synthesis:** *How should one synthesize the results of the analysis step so as to best evacuate the occupants in light of the performance measures?*

Representation Stage

Figure 2 depicts a large hospital campus with many inter-connected buildings, many different levels, and a complex array of circulation passages, and illustrates that the evacuation problem is a difficult one to represent. However, one can begin to accurately model the evacuation process through a network as depicted in Fig. 3. By definition, an evacuation network (graph) $G(V, E)^{\ell}$ is comprised of a finite set V of nodes (vertices) of size N where $V = \{V_1, V_2, \ldots, V_n\}$ together with a finite set E of arcs $e_k = (v_i, v_j) \quad \forall (i, j)$ nodal pairs and an indication of the level at which the network is defined ℓ. The levels actually correspond to the degree of aggregation inherent in modelling large com-



Evacuation Networks, Figure 3
Route site plan

plex networks. V can further be partitioned into three sets of nodes:

V_1 := *which represents the occupant source nodes during the evacuation,*
V_2 := *which represents the intermediate nodes during the evacuation;*
V_3 := *which represents the sink or destination nodes of the occupants.*

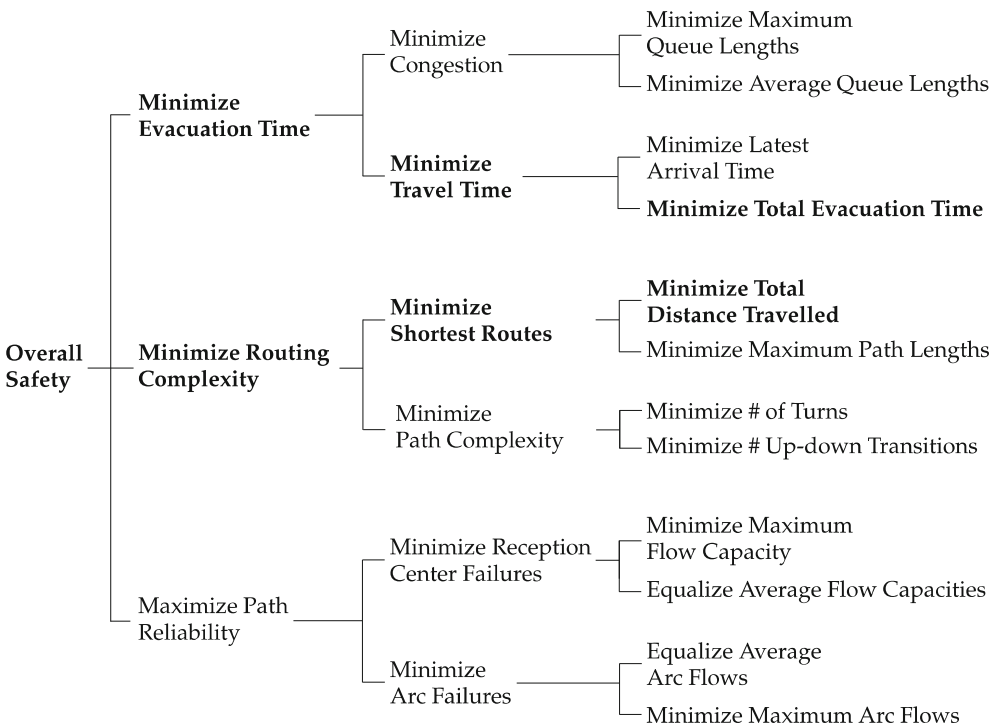
The set of arcs represent the different streets, passageways, or routes from V_1 to V_3. Associated with each node $\ell \in V$ and each arc $(v_i, v_j) \in E$ are variables and parameters which represent node and arc processing times, node and arc capacities, arrival times to the network, distances, and occupant population sizes at the source nodes.

Figure 3 illustrates the example evacuation network with the key congested routes in the evacuation planning problem embedded in the network model.

The Representation Step is often defined in terms of the size and composition of the customer population: infinite, finite, or mixed and how the facility under study should be decomposed by V, E, and ℓ. The crucial link between the Representation and Analysis Steps is the complexity (i. e., number of nodes and arcs) of G^{ℓ}, which governs the number of equations used in the mathematical model in the Analysis Step. The Representation Step presents an interesting and challenging problem because of the many possible ways of representing regions, facilities, ships, vehicles, and building components.

Analysis Step

The *Analysis Step* is the point at which the methodology and mathematical models underlying the flow processes, and the algorithmic structure for computing the performance characteristics of $G^{\ell}(Z, E)$ come together. Mathematically, we have a network $G(V, E)$, with a finite set of nodes V, and edges(arcs) E, over which multiple classes of customers (occupants) flow from source(s) to sink(s) while a vector of objective functions $\Omega = \{f_1(\bar{x}), f_2(\bar{x}), \ldots, f_p(\bar{x})\}$ is simultaneously extremized subject to a set of constraints on the occupants flowing through the network. Figure 4 cap-



Evacuation Networks, Figure 4
Morphological diagram of multi-objective approaches

tures many of the recognized criteria appropriate in analyzing a network evacuation problem. In our studies, we have often used Minimum Total Evacuation Time and Minimum Total Distance Travelled to capture the evacuation problem. The Total Distance travelled is a suitable surrogate objective for approaching the route complexity, since reducing the evacuation path length will often begin to capture the path complexity and, hopefully, minimizing this measure will abate the occupants sense of panic. Other objectives might be appropriate given the particular context or decision situation.

Synthesis Step

Given the performance characteristics determined during the Analysis Step, we can begin to optimize the network topology itself, routing and resource allocation problems within:

Topological Network Design (TND): Determination of the number, type, and subset of nodes and arcs as well as the particular node and arc topology to be used for the evacuation.

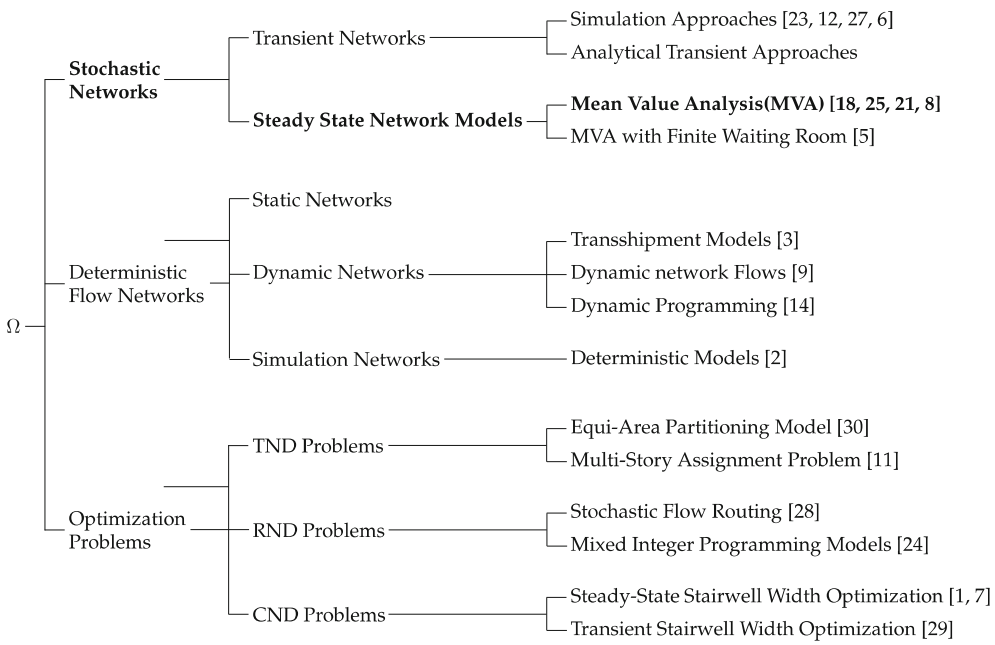
Routing Network Design (RND): Determination of the routing scheme in both steady-state and real time.

Capacitated Network Design (CND): Determination of the Network Resources: Number of highway lanes, corridor length, widths, areas, landing shape, reception center capacity, configuration etc.

Mathematical Models

There are many possible mathematical modelling approaches once our network is constructed and Fig. 5 represents the range of approaches many research scientists have followed. References are provided for further details. The *boldface* text along the morphological tree represents the approach suggested in this chapter which we have applied in many different contexts.

Many mathematical models which have appeared in the literature for generating and evaluating evacuation paths for an occupant population [3,9,13,24,31]. Besides the models for estimating flows, many newer works are just becoming available for the optimization of the evacuation networks, i. e. the TND, RND, and CND problems, and these are illustrated in the third branch of the morphological tree.



Evacuation Networks, Figure 5
Morphological diagram of EEP approaches

Set Partitioning Model

The model which is presented below is a variation of one model appearing in [13]. It was one of the first to account for the critical features of the stochastic evacuation problem. Another class of models that one might utilize to formulate the problem are those of the class of multi-commodity network flow models. Unfortunately, these models will not control the Bernoulli splitting of the occupant population along the different evacuation paths which is problematic since splitting the different source populations will engender confusion and create a potential sense of panic among the evacuating occupants. The integer set partitioning programming model presented below has the desired property to control splitting of the flows. The multi-objective model of our routing problem is:

$$Minimize\{f_1(\bar{x}); f_2(\bar{x})\}$$

where:

$$(Evacuation\ Time) : f_1(\bar{x}) = \sum_i \sum_j \sum_k q_{ijk}\lambda_{ijk}x_{ijk}$$

$$(Distance\ Travelled) : f_2(\bar{x}) = \sum_i \sum_j \sum_k d_{ijk}x_{ijk}$$

subject to:

$$(V_2\ Arcs) : \sum_i \sum_j \sum_k \alpha_{\ell ijk}\lambda_{ijk}x_{ijk} \leq \mu_\ell \quad \forall \ell \tag{1}$$

$$(V_3\ Sinks) : \sum_i \sum_j \sum_k p_{ijk}x_{ijk} \leq C_q \quad \forall q \tag{2}$$

$$(Occupant\ Classes) : \sum_k x_{ijk} = 1 \quad \forall ij \tag{3}$$

$$(Routes) : x_{ijk} = 0, 1\ \forall ijk \tag{4}$$

and where:

x_{ijk} := 1 if the ith occupant class from the jth source is assigned the kth route alternative.

λ_{ijk} := the arrival rate of the ijth occupant population into the kth routing alternative.

$\alpha_{\ell ijk}$:= a data coefficient which equals 1 if the ℓth arc is included in the ijkth route assignment and equals 0 otherwise.

μ_ℓ := maximum allowable traffic service rate along arc ℓ.

C_q : capacity of sink (destination) node q.

p_{ijk} := occupant population of source ij on the kth route alternative.

q_{ijk} := expected evacuation (sojourn) time of the ijkth occupant class. These values must be calculated from the particular stochastic model used in the evacuation study, see discussion below.

d_{ijk} := average distance travelled for the ijkth occupant class.

Since we have two objectives in our model, it makes sense to talk of the Noninferior (NI) set of route alternatives, since the tradeoffs between f_1 and f_2 naturally underlie the optimal set of solutions we seek. Because of the complexity of solving this model directly, an alternative approach which systematically generates feasible routing alternatives to a relaxed version of our mathematical model but at the same time measures the critical objectives of evacuation time and distance travelled is proposed and demonstrated in the next two sections of the chapter.

Congestion Models

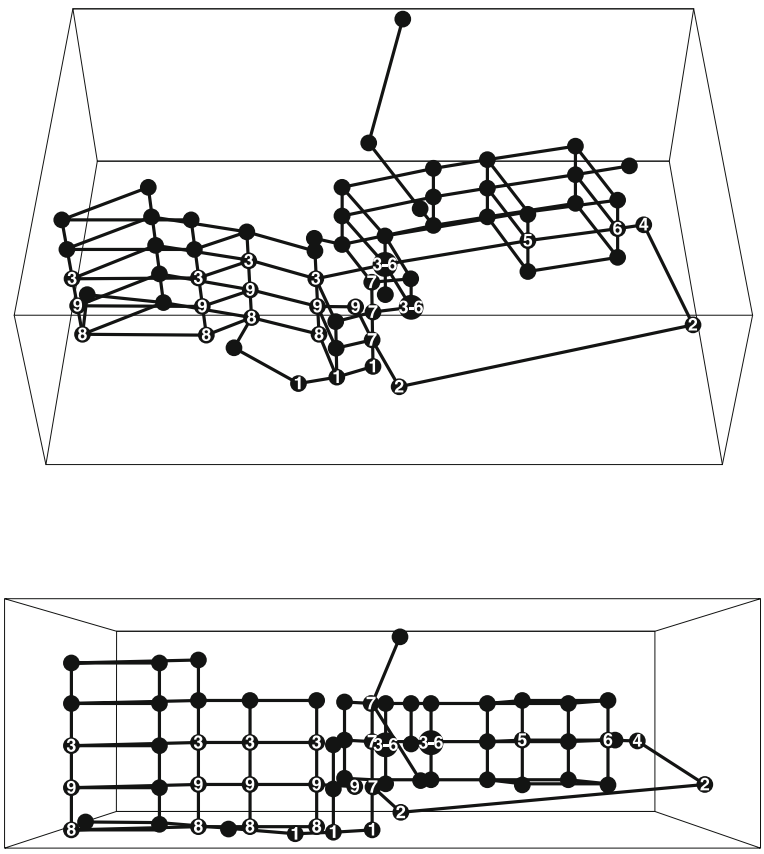
The real crux of the evacuation problem is to capture the congestion that naturally occurs when occupants choose the shortest routes to evacuate. There are some deterministic measures possible for measuring congestion, yet stochastic ones are the most accurate, because queueing is a nonlinear complex phenomenon.

Erlang Loss/Delay Networks

Fundamentally, each S_j node in the circulation network is an $M/G/c/c$ queue, i. e. there is no waiting room and C depends on the square footage area of the circulation segment or the number of vehicles which can maximally occupy a highway segment [33]. Let's for the sake of the argument, focus on pedestrian evacuation. Later on we will show how our model extends to vehicular congestion. Each occupant in the circulation system consumes approximately 0.2 m^2 of floorspace, and, therefore, the capacity of a circulation system element is:

$$C = 5LW$$

where L(Length) and W(Width) are given in meters. Each circulation segment is a representative "building



Evacuation Networks, Figure 6
Three-dimensional network models

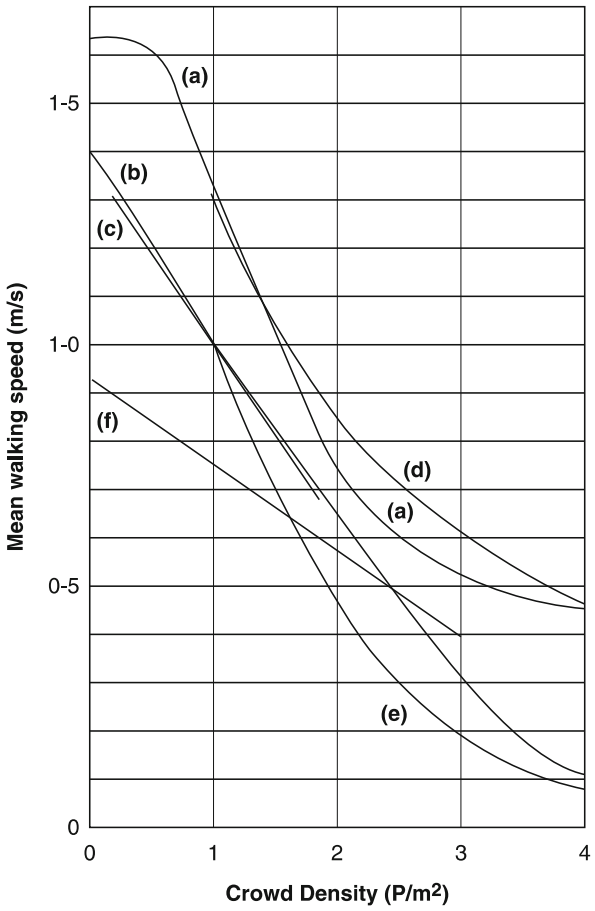
block" for modelling pedestrian movements through the facility. Corridor segments, intersections, landings, stairwells, ramps, and so on represent a network of interconnected $M/G/c/c$ queues, see Fig. 6. The separations of the circulation blocks are due to changes in flow direction, level, or merging and splitting decisions. Further, the cardinality of S depends on the configuration and complexity of movement patterns within the facility. Flows through the nodes of S, the circulation system of a building are largely state dependent, in that a customer receives service in the circulation node S_j and this service rate decays with increasing amounts of customer traffic.

Figure 7 shows a family of curves which represent the variety of empirical studies (curves a-f in Fig. 7) that document the decay rate of the customer service rate as a function of population density in a corridor. Empirical models are also available showing distributions for stairs and other circulation elements with bi-, and multi-directional pedestrian flows [10,26]. Finally, there are a set of classical linear and exponential curves which relate vehicle speed and vehicle density captured in Fig. 8. We have utilized these type of vehicular speed/density relations to develop state dependent models for vehicular traffic analysis [12]. In general, the service rate μ is a function of velocity v_i, which is a constant for each individual in the corridor. Thus, it takes t_i (seconds)

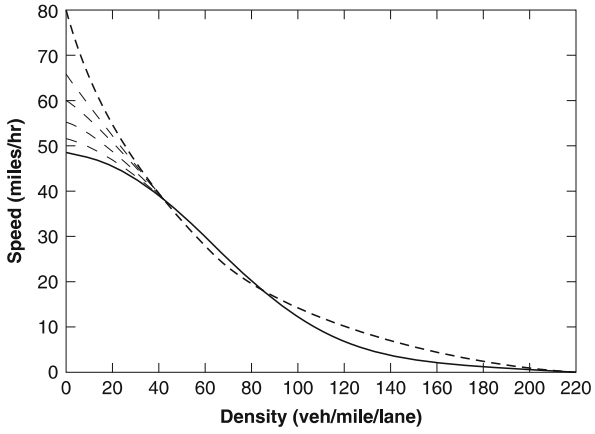
$$t_i = \frac{L}{v_i}$$

for each person to traverse the corridor, where i is the number of occupants in the circulation system when an individual enters.

Because of the complexity of dynamically updating the service rate as a function of the number of customers within a corridor segment, it becomes extremely difficult to utilize digital simulation models in



Evacuation Networks, Figure 7
Empirical distributions of pedestrian traffic flows



Evacuation Networks, Figure 8
Empirical distributions of vehicular traffic flows

the design of circulation systems within buildings. Our computational experience in digital simulation of access and egress networks underscores this defect in simulation models. We must, therefore, look to analytical models to aid the network design process if state dependent models are to be effectively utilized. Also, since we are examining the pedestrian/vehicular network as a design problem rather than as a control problem, it makes most sense to look at steady state measures rather than transient ones.

We have recently developed a generalized model of the $M/G/c/c$ Erlang loss queueing model for service rate decay which can model any service rate distribution (linear, exponential, ...) [4,5,22]. It is a special case of an Erlang Loss model. Kelly [14] has treated M/G/c/c state dependent models in his book, but only ones with a *linear, increasing* function of the number of customers in the queue, whereas, we treat the queue with an *nonlinear, decreasing* service rate, see Fig. 3.

Our M/G/c/c state dependent model dynamically models the flow rate of pedestrians within a corridor as a function of the population within the corridor. Suppose that G is a continuous distribution having density g and failure rate $\mu(t) = g(t)/G(t)$. Loosely speaking, $\mu(t)$ is the instantaneous probability intensity that a service t units old will end. The service rate depends on the number of customers in the system: given that there are n people in the system, each server processes work at rate $f(n)$. In other words, if there is an arrival, the service rate will change to $f(n+1)$ and if there is a departure, the service rate will change to $f(n-1)$.

In particular, the probability distribution of the number of occupants in the corridor is given by:

$$P(n \text{ in system}) = \frac{[\lambda E(S)]^n P_0}{n! f(n) \dots f(2) f(1)} \quad n = 1, 2 \dots C$$

where

$$P_0 = \frac{1}{1 + \sum_{i=1}^{C} \frac{[\lambda E(S)]^i}{i! f(i) \dots f(2) f(1)}}$$

$$E(S) = \frac{L}{1.5}$$

$$f(n) = \frac{v_n}{v_1}$$

and $E(S)$ is the mean service time of a lone occupant flowing through a corridor of length L, with service rate

1.5 m/s (see Fig. 3). The term v_n is defined as the average walking speed when n people are in the corridor. For the $M/G/c/c$ state dependent model, we have also shown that the departure process (including customers completing service and those that are lost) is a Poisson process with rate λ [4,5].

Algorithms

The problem we face in our evacuation planning problem is that we do not know a priori which paths are NI without assessing the congestion in $G(V, E)$. We must iteratively generate candidate paths, assess the congestion in $G(V, E)$, and then iterate again until the desired tradeoffs between distance travelled and evacuation time is acceptable to the planner. This iterative process leads to the algorithm described below. For product form networks where the estimate of time delays in the *Expected Savings* calculation for re-routing among the alternative Noninferior paths can be computed exactly, then the algorithm will guarantee finding a Noninferior path for re-routing the occupant classes. For non-product form networks, which are typically the case, we can only approximate these time delays, therefore, the algorithm can only guarantee an approximate Noninferior solution. Considering the complexity of the underlying stochastic-integer programming problem, this is a reasonable and practical strategy.

K-shortest Paths

The algorithm to facilitate the design methodology can be incorporated into any appropriate discrete-event simulation model [6] or analytical model [8] to estimate f_1, f_2, and carry out the evacuation planning/routing analysis. To summarize and focus the efforts in this chapter, an algorithmic description of *Steps 1.0, 2.0, 3.0* and it substeps are presented.

Step 1.0: Representation Step Represent the underlying facility or region as a network $G(V, E)$ where $V :=$ is a finite set of nodes and $E :=$ is a finite set of arcs or nodal pairs.

Step 2.0: Analysis Step Analyze $G(V, E)$ as a queueing network either with a transient or steady-state model and compute the total evacuation time of the occupant population along with total distance travelled to evacuate given a set of evacuation paths.

Step 3.0 *Synthesis Step*

***Step 3.1*:** Analyze the queueing output from the evacuation model and compute the set of Noninferior evacuation paths which simultaneously minimize time and distance travelled in $G(V, E)$ for each occupant population.

3.1.1 If the set on NI paths are uniquely optimal then

$$E_{ij}^k = q_{ijk} - \left[\left(d_{ij}^k/\omega\right) + q_{ij}^k\right] \leq 0 \quad \forall ijk$$

go to Step 3.2 where:

$E_{ijk} :=$ is the net increase or decrease in the average egress time per person caused by re-routing occupants to the (kth + 1) Noninferior route.

$q_{ijk} :=$ the sum of the average queue times per person on the original route.

$d_{ij}^k :=$ the increased distance travelled on the (kth + 1) Noninferior route (e. g. if the kth Noninferior route is 100 feet and the kth + 1 Noninferior route is 120 feet, d_{ij}^k is equal to 20 feet i. e. 120 minus 100).

$\omega :=$ is the average travel speed for d_{ij}^k.

$q_{ij}^k :=$ the sum of the expected queue times per person on the (kth + 1) Noninferior route, otherwise:

3.1.2 Significant queueing (congestion) exists on one or more routes then go to Step 3.3.

***Step 3.2*:** STOP! The NI shortest time/distance routes are optimal and identical and total evacuation time, distance and congestion are minimized.

***Step 3.3*:** Determine the total number of occupants who pass through the queueing area(s) and trace them back to their origins.

***Step 3.4*:** Select the total number of occupants to be re-routed from each source node. The total number of occupants re-routed is correlated to both the size of the queues and the number of occupants on each route. In selecting the population, the analyst should strive to achieve uniformity of occupants and queues on each egress route.

***Step 3.5*:** Re-route the population to the kth route of the NI set of paths where k is selected by em-

ploying the following formula:

$$E_{ij}^{k} = q_{ijk} - \left[\left(d_{ij}^{k}/\omega\right) + q_{ij}^{k}\right] \quad \forall ijk$$

***Step 3.6*:** Select the largest positive E^{*} for each set of populations to be re-routed, where:

$$E^{*} = \max_{\forall i\, j \text{sources}} \{E_{11}, E_{22}, \ldots, E_{IJ}\}$$

for all possible savings, and then re-run the computer evacuation planning model with the new set of routes, by returning to *Step 2.0* of the *General Algorithm*. If all $E_i's$ are negative, stop! The current set of NI shortest routes used on the previous iteration are selected.

Other Algorithms

Besides the *k*-shortest path approach, one might utilize a turn-penalty algorithm to guide the process of determining the evacuation paths. This is probably very appropriate in vehicular evacuation schemes. Also, another approach which seems quite viable, would be to define the set of arc disjoint paths, since this would tend to completely separate the occupant congestion along the paths. We have not experimented with these approaches to define the evacuation routes, but their use might be quite appropriate in the future.

Summary and Conclusion

We have given you some insights into the performance modelling and optimization problems associated with evacuation networks. As the maturity of this application area grows, the more research that is devoted to the area, the more theoretical and algorithmic issues and progress that will emerge.

References

1. Bakuli DL, MacGregor Smith J (1996) Resource Allocation in state-dependent emergency evacuatin networks. Eur J Oper Res 89:543–555
2. Berlin GN (1982) A simulation model for assessing building firesafety. Fire Technol 18(1):66–76
3. Chalmet LG, Francis RL, Saunders PB (1982) Network models for building evacuation. Manag Sci 28(1):86–105
4. Cheah J (1990) State dependent queueing models. Masters Thesis, Department of Industrial Engineering and Operations Research, University of Massachusetts, Amherst
5. Cheah J, MacGregor Smith J (1994) Generalized M/G/c/c state dependent queueing models and pedestrian traffic flows. Queueing Syst Appl 15:365–386
6. Cruz FRB, MacGregor Smith J, Medeiros RO (2005) An M/G/c/c State Dependent Network Simulation Model. Comput Oper Res 32 (4):919–941
7. Cruz FRB, MacGregor Smith J, Queiroz DC (2005) Service and Capacity Allocation in M/G/c/c State Dependent Queueing Networks. Comput Oper Res 32(4):919–941
8. Cruz FRB, MacGregor Smith J (2007) Approximate Analysis of M/G/c/c State Dependent Queueing Networks. Comput Oper Res 34:2332–2344
9. Francis RL, Chalmet LG (1980) Network models for building evacuation: a prototype primer. Unpublished Paper, Department of Industrial and Systems Engineering, University of Florida, Gainesville
10. Fruin JJ (1971) Pedestrian Planning and Design. Metropolitan Association of Urban Designers and Environmental Planners, New York
11. Hahn, Peter, MacGregor Smith J, Yi-Rong Zhu (2007) The Multi-Story Assignment Problem, in review
12. Jain R, MacGregor Smith J (1997) Modeling Vehicular Traffic Flow using M/G/c/c State Dependent Queueing Models. Transp Sci 31(4):324–336
13. Karbowicz CJ, MacGregor Smith J (1984) A k-shortest path routing heuristic for stochastic evacuation networks. Eng Optim 7:253–280
14. Kelly FP (1979) Reversibility and Stochastic Networks. Wiley, Chicester
15. Kostreva M, Wiecek MW (1993) Time Dependency in Multiple Objective Dynamic Programming. J Math Anal Appl 173:289–307
16. MacGregor Smith J, Rouse WB (1979) Application of Queueing Network Models to Optimization of Resource Allocation within Libraries. JASIS 30(5):250–263
17. MacGregor Smith J (1981) The Use of Queueing Networks and Mixed Integer Programming to Optimally Allocate Resources within a Library Layout. JASIS 32(1):33–42
18. MacGregor Smith J, Towsley D (1981) The Use of Queueing Networks in the Evaluation of Egress from Buildings. Environ Plan B 8:125–139
19. MacGregor Smith J (1982) An Analytical Queueing Network Computer Program for the Optimal Egress Problem. Fire Technol 18(1):18–37
20. MacGregor Smith J (1982) Queueing Networks and Facility Planning. Build Environ 17(1):33–45
21. MacGregor Smith J (1987) QNET-C: an interactive graphics computer program for evacuation planning. In: Newkirk R (ed) Proceeding of the Society for Computer Simulation Emergency Planning Session, SCS Multi-conference, Orlando. pp 19–24
22. MacGregor Smith J (1991) State Dependent Queueing Models in Emergency Evacuation Networks. Transp Sci: Part B 25B(6):373–389

23. Stahl FI (1982) BFIRES-II: A Behavior Based Computer Simulation of Emergency Egress during Fires. Fire Technol 18(1):49–65
24. Stepanov A (2008) An Integrated Methodology for Optimal Egress Route Assignments during Population Evacuation under an Evolving Emergency Event. PhD Dissertaion. Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst
25. Talebi K, MacGregor Smith J (1984) Stochastic network evacuation models. Comput Oper Res 12(6):559–577
26. Tregenza P (1976) The Design of Interior Circulation. Van Norstand Reinhold Company, New York
27. Watts JM (1986) Computer models for evacuation analysis. Paper presented at the SFPE Symposium: Quantitative Methods for Life Safety Analysis College Park Maryland. Available from the Fire Safety Institute, Middlebury
28. Wen Y, MacGregor Smith J (2007) Stochastic Flow Analysis of the Evacuation of Multi-Story Facilities. in review
29. Wen Y, MacGregor Smith J (2007) Stairwell Width Optimizationi in the Evacuation of Multi-Story Facilities. in review
30. Wen Y, MacGregor Smith J (2007) On the Equi-Area Partitioning Problem for Rectilinear Simple Polygons. in review
31. Wen Y (2007) Design, Evaluation and Optimization of the Evacuation Problem of Multi-Story Facilities. PhD Dissertaion. Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst
32. Woodside CM, Hunt RE (1977) Medical Facilities Planning Using General Queueing Network Analysis. IEEE Trans SMC 7(11):793–799
33. Yuhaski S, MacGregor Smith J (1989) Modeling circulation systems in buildings using state dependent queueing models. Queueing Syst Appl 4:319–338

Evolutionary Algorithms in Combinatorial Optimization

EACO

Daniel Kobler
Department Math., Swiss Federal Institute Technol., Lausanne, Switzerland

MSC2000: 90C27, 05-04

Article Outline

Keywords
The Traveling Salesman Problem
The Vehicle Routing Problem
The Quadratic Assignment Problem
The Satisfiability Problem (SAT)
The Set Covering and Set Partitioning Problems
The Knapsack Problem
The Bin Packing Problem
Graph Coloring
Other Graph Problems
  Maximum Clique
  Graph Partitioning
Miscellaneous
  Sequencing and Scheduling
  Steiner Trees
Conclusion
See also
References

Keywords

Evolutionary algorithm; Combinatorial optimization; Heuristics

Most of the *NP*-hard combinatorial optimization problems cannot be solved to optimality in practice. Therefore heuristic techniques have to be used to obtain solutions of high quality. There exists different approaches to design a heuristic algorithm, such as tabu search and genetic algorithm for example. The latter solution method belongs to a wider class of algorithms, called *evolutionary algorithms*, that handle a set of several solutions. Within this class, the best known algorithms that are applied to combinatorial optimization problems are genetic algorithms (cf. ► Genetic algorithms) and ant systems. For a general presentation, one can mention [22,72] for genetic algorithms and [12,23] for ant systems.

In this article, a review of the evolutionary algorithms used up to 1998 in combinatorial optimization is being made. For a certain number of combinatorial problems, the main papers that present an evolutionary algorithm for that problem are referenced, and some short remarks are given. While it is difficult to provide a very precise definition of an evolutionary algorithm, this term will be used here as a synonym of population-based algorithm: an algorithm that makes evolve several solutions, in particular by exchanging some kind of information between them. Algorithms that iteratively modify a solution in order to obtain a good one (like tabu search or genetic algorithms with a 'population' of size 1) will not be considered as evolutionary algorithms.

The Traveling Salesman Problem

The traveling salesman problem (or TSP) is probably the problem on which the largest number of evolutionary algorithms have been applied. It consists in determining a shortest tour visiting all of the given cities exactly once. A very complete survey of local search approaches to this problem has been provided by D.S. Johnson and L.A. McGeoch [51], while J.-Y. Potvin [70] compared several genetic algorithms for TSP. In [51], the authors recommend different solving techniques depending on the quality of the solution desired and the time available. Genetic algorithms or ant systems are a good choice if enough running time is allowed and good solutions are needed. With similar running times, the iterated Lin–Kernighan algorithm (or ILK) yields better results but is more complex to implement. In ILK, a single solution instead of a population of individuals is considered and this method will therefore not be referred to as an evolutionary algorithm. If there is no restriction on the running time, the best results can be obtained by genetic algorithms based on ILK.

An important breakthrough in the field of evolutionary algorithms for the TSP was the paper [67] by H. Mühlenbein, M. Gorges-Schleuter and O. Krämer. In their algorithm, implemented on a parallel machine, a solution was allowed to mate only with certain other solutions and some optimization technique was applied to the offsprings. Indeed, the use of a local search algorithm to improve created offsprings is a necessary condition for an evolutionary algorithm to be efficient. Moreover, they designed a crossover specific to the TSP, called MPX (maximum preservative crossover). It consists in copying a segment of a certain length from a first parent into the offspring and adding cities consecutively from the second parent according to some rules. This crossover is very suitable for the TSP, as shown in [66]. Further researches studied the impact of the different elements on the results and improved the quality of the solutions obtained [7,44,89]. Several other crossovers, most of them using two parents, have been suggested by various authors. In particular, B. Freisleben and P. Merz proposed [37,38] the distance preserving crossover (or DPX): An offspring is created by keeping the edges that are found in both parents, and greedily reconnecting the different pieces without using the edges contained in only one parent. They obtain a very efficient algorithm, that won both the ATSP (asymmetric TSP) and the TSP competitions at the First International Contest in Evolutionary Optimization [6]. They further improved their algorithm, in terms of speed and quality of solutions, in [39]. Their use of an edge-preserving crossover and of a hill-climbing algorithm illustrates important elements necessary to obtain an efficient genetic algorithm for TSP. These elements have been put forward in different comparisons between various genetic algorithms for TSP [70,78], together with the necessity to split the population into several subpopulations for solving large instances (more than a few hundred cities).

The first presentation of ant colony optimization (ACO) [12] was made with the TSP as illustration and this problem remains the most often used application problem of works on ant colony optimization. The initial ACO system, named ant system, has been extended to what is called ant colony system (ACS). A description of this algorithm can be found in [23] by M. Dorigo and L.M. Gambardella. In the same paper, local search has been added to ACS and the resulting algorithm has been applied to ATSP and TSP. The results reported are better in [39] for TSP, but are better in [23] for ATSP. Another proposed extension of ant system, called *MAX-MIN ant system* [79], consists in introducing explicit maximum and minimum values for the trail factors on the arcs. Good results are obtained with such an algorithm when local search is added.

The Vehicle Routing Problem

The most studied extension of the *vehicle routing problem* (VRP) is the one with time windows (VRPTW). In order to solve this problem, a two-phase heuristic, called *GIDEON*, has been proposed in [84]. The first phase uses a genetic algorithm to cluster the customers, and the solutions obtained are improved by local optimization techniques in the second phase. This procedure has first been improved in [83], and then extended in [85]. In this last paper, S.R. Thangiah, I.H. Osman and T. Sun present several metaheuristics, all having a first phase similar to the one in GIDEON. These algorithms have been compared to several other heuristics and showed very good results on test problems taken

from the literature. Some improvements have still to be brought for solving problems with large time windows. For such problems, a heuristic based on *simulated annealing* and a population-based algorithm called *GENEROUS* [71] are shown to be a little more efficient. The latter is not a standard genetic algorithm since it does not represent solutions by chromosomes, but it nevertheless handles several solutions and uses a recombination operator. An *adaptive memory* procedure, in conjunction with tabu search, has also been applied to this problem [75].

Improvements of the GIDEON approach with local post-optimization procedures have also been used for the VRP with time deadlines. A comparison done in [86,87] with two other heuristics shows that the cluster-first route-second algorithm with a genetic algorithm in the first phase performs well for problems in which the customers are distributed uniformly and/or with short time deadlines.

The Quadratic Assignment Problem

The *quadratic assignment problem* (or QAP) allows the modelization of many practical problems in location science, but can be solved optimally only for very small instances. Therefore different heuristics have been proposed for this problem. Several of them are compared in [13,81]. For real-world problems (irregular and structured), the genetic hybrid by C. Fleurent and J.A. Ferland in [33] appears to be one of the most efficient algorithms [81]. Based on a standard genetic algorithm with solutions encoded as permutations [82], this genetic hybrid applies a robust tabu search on the offsprings and was able to find several new best solutions on some benchmark problems.

The ant colony optimization approach has also been considered, first in [64]. This ant system algorithm, hybridized with a local search, has been improved in [62,63] and provides very good results. A different ACO approach, where at each iteration the solutions are modified instead of newly constructed, has been proposed in [40]. This algorithm, also hybridized with a local search procedure, yields better results on real-world problems than the genetic hybrid of [33], but is not competitive on random problems. A further promising method, based on *scatter search*, has been presented in [19].

The Satisfiability Problem (SAT)

The problem of finding a truth assignment for variables to make a propositional formula true is probably the best known, and historically the first, *NP*-complete problem. But only few evolutionary algorithms for SAT can be found in the literature. After a straightforward approach in [52], a rather different solution representation has been proposed in [45]. But the drawback of this method, despite adapted operators, is that it increases the size of the individuals in an important way, compared to the coding 'one gene for one variable'. This last coding has been used in [35], together with a SAT-adapted crossover (the objective function being simply the number of satisfied clauses). But the evolutionary algorithm thus obtained was not able to compete with a tabu search (also presented in [35]). The tabu search-genetic hybrid (where some iterations of tabu search is used for mutation) is computationally expensive, but is able to solve large instances that a tabu search alone cannot solve. For smaller instances, the hybridization is not useful.

Another heuristic approach to SAT consists in assigning weights to the different clauses and minimizing the sum of the weights of the unsatisfied clauses. These weights are adapted during the algorithm depending on the 'difficulty' of each constraint. This mechanism has been used in evolutionary algorithms in [25] and [90], but in both cases it came out that the best results are obtained with a 'population' of size 1. Such an algorithm is therefore no longer considered as an evolutionary algorithm.

The Set Covering and Set Partitioning Problems

The *set covering problem* (SCP) is a zero-one integer programming problem where the constraints are all of the type $\sum_j a_{ij}x_j \geq 1$ with zero-one coefficients. It is a well-known problem, that has also been used to study penalty functions in genetic algorithms [3,74].

Different genetic algorithms approaches have been proposed in the literature (see for example [50,60,61]), and a very efficient one has been presented by J.E. Beasley and P.C. Chu in [5]. This algorithm uses binary representation of the solutions, and a repair operator to preserve the feasibility of the individuals and to improve the solutions. Moreover, a variable mutation rate has been introduced. Results on

standard test problems up to 1000 constraints and 10,000 variables show the efficiency of this algorithm that was able to improve the best-known result on some of the larger instances. The same paper shows no significant difference between various crossovers.

The *set partitioning problem* (SPP) is also a zero-one integer programming problem, the difference with SCP being that the constraints are equalities instead of inequalities. Relatively few heuristics have been developed for this problem. D. Levine investigated sequential and parallel genetic algorithms for SPP [59]. His best algorithm was a genetic algorithm in an island model, hybridized with a local search heuristic. But this algorithm remained less efficient, both in terms of quality of the solutions and in terms of running time, than the branch and cut approach of [49]. Some problems met by his algorithm were due to the penalty term for infeasible solutions in the fitness function. In order to overcome these problems, other authors decomposed the single fitness measure in two distinct parts (the objective function and a measure of 'infeasibility') [10]. Adapting the parent selection method to this modification, and also using an improvement operator, they obtained a better genetic algorithm, but that is still not able, for the problems they considered, to compete with a commercial mixed integer solver.

The Knapsack Problem

The multidimensional (zero-one) *knapsack problem* is equivalent to the zero-one integer programming problem with nonnegative coefficients. Only few papers tried to solve this problem with evolutionary algorithms. While the first such algorithms did not give high-quality results and were not competitive with other heuristics [56,88], the quality has improved. Genetic algorithms as presented in [11,48], both working only with feasible solutions, are able to obtain optimal solutions on standard test problems (instances with at most 105 variables and 30 constraints). In [11], Chu and Beasley proposed some larger test problems (up to 500 variables and 30 constraints), without known optimal solution, and used them for a comparison with other heuristics. Their genetic algorithm uses a 'repair' operator specific to this problem to ensure good feasible offsprings and obtained high-quality results, but needed also more computation time (on a same machine, about one hour for the genetic algorithm against a few seconds for the other heuristics).

The Bin Packing Problem

The standard one-dimensional *bin packing problem* consists in putting items of given sizes in bins of given capacity. Many evolutionary algorithms proposed for this problem (genetic algorithms and evolution strategy, see for example [16,57,77]) performed worse than a simple heuristic like first fit decreasing. E. Falkenauer and A. Delchambre then suggested in [30] a genetic algorithm designed for grouping problems: the *grouping genetic algorithm* (GGA). In this algorithm, solutions are represented by chromosomes having two parts: the item part encodes for each item its bin and the group part, of variable length, encodes the bin identifiers used. The crossover, mutation and inversion operators have been adapted to this encoding. Instead of simply using the number of bins, the authors designed a fitness function that also takes into account the proportion to which each bin is filled. With this approach, they obtained very satisfactory results. The arguments presented for this new encoding are discussed by C. Reeves in [73]. In the same paper, a hybrid genetic algorithm is presented, where solutions are represented by permutations and decoded using heuristics like first fit and best fit. The results obtained are more or less similar to those in [30]. A problem size reduction heuristic, similar to the reduction process used in [16], has also been introduced in this genetic algorithm. According to Falkenauer [29], this reduction violates the search strategy of the genetic algorithm and he therefore prefers the GGA's crossover, that has the same goal of propagating promising bins. In the same paper, the GGA is improved by the introduction of local optimization inspired by the dominance criterion of [65]. The new algorithm is compared with an efficient branch and bound algorithm and gives excellent results.

Extensions of the standard bin packing problem, like the two-dimensional bin packing problem, have also been considered with evolutionary algorithms [15,69,77]. An overview of these variations is presented in [43].

Graph Coloring

The *graph coloring problem* is a well-known problem in graph theory; it consists in determining the smallest number of colors that must be used to color the vertices of a graph such that two adjacent vertices do not have the same color. L. Davis is the first author who proposed an evolutionary algorithm for this problem [22]. In fact, he considered a graph with weights on the vertices and an integer k. He then designed a hybrid genetic algorithm for finding a partial k-coloring such that the colored vertices have maximum total weight. In this algorithm, individuals are represented as permutations of the vertices of the graph. This order-based encoding is not very efficient, as shown by Fleurent and Ferland in [34]. In this paper, they also present hybrid genetic algorithms that use string-based encodings of the solutions for finding a coloring in k colors with as few conflicting edges (edges with both ends of the same color) as possible. They consider different crossovers, including a graph-adapted one, and hybridize the genetic algorithm with a simple local search or with tabu search (a modified version of [46]). The results on random graphs G_n, 0.5 improve the previous best results. For graphs up to 300 vertices, their tabu search-genetic hybrid and their tabu search give similar results, but in much less time for the latter. For larger graphs (500 or 1000 vertices), the running time becomes prohibitive, and both the evolutionary algorithm and the tabu search must be used within a different approach (determining large stable sets and coloring the residual graph). The tests on 450-vertices Leighton graphs (with known chromatic numbers) showed that the tabu search-genetic hybrid outperforms the tabu search on about half of the instances, while the opposite is true for the remaining instances. The hybrid algorithm was able to find an optimal solution for two instances (out of twelve) that could not be solved by the tabu search alone.

Another evolutionary algorithm has been proposed in [18], with a graph-adapted crossover that takes into account how 'close' a vertex is to conflicting edges. The improving algorithm applied to offsprings is a steepest descent method, instead of a tabu search like in [34]. Despite this less sophisticated method, their algorithm gives similar results to those obtained by the hybrid algorithm in [34]. Moreover, the latter gives worser results when the tabu search is replaced by a simple descent method.

Concerning ant colony optimization, a first approach to graph coloring has been proposed in [17], but the results obtained need improvements.

Other Graph Problems

Maximum Clique

The problem of determining the *maximum clique* (complete subgraph) in a graph is equivalent to the problem of determining the minimum vertex cover or the maximum stable set in the complementary graph. A first genetic algorithm, hybridized with a tabu search, has been proposed by Fleurent and Ferland in [35], but they show that their tabu search alone gives similar results in a shorter time. In these algorithms, a solution is a set of vertices of given size and the objective function measures how many edges are missing for a set to be a clique. Improving an algorithm of [2], E. Balas and W. Niehaus [4] proposed a genetic algorithm (without improving algorithm applied to the offsprings) for both the maximum cardinality and maximum weight clique problems where an individual is a clique. In this algorithm, the recombination operation ('crossover') used is designed specifically for this problem and taken from another heuristic. The results obtained on the DIMACS benchmark graphs are very good, similar to those obtained in [35] from the point of view of the solutions' quality. A different fitness function has been suggested in [8] and included in a hybrid genetic algorithm using a local optimization step. The fitness value associated to a set of vertices is a weighted combination of the size of the set and the number of edges missing to have a clique, but the weights are modified during the run of the algorithm according to a simple rule. Despite the introduction of a preprocessing step that determines the order of the vertices on the chromosome, this algorithm is less efficient (but this may be due to the use of the 2-point crossover).

Graph Partitioning

Evolutionary algorithms are rather seldom used to tackle the *k-way graph partitioning problem* (partitioning a (weighted) graph in k equal-sized parts), even if the graph bisectioning problem (the case $k = 2$) is some-

times taken to illustrate various ingredients in genetic algorithms ([9,54]). For the general k-way graph partitioning problem, different problem-oriented operators are introduced and studied in a parallel genetic algorithm in [58]. In this algorithm, the population is only composed of feasible solutions. Another approach has been proposed in [76] where the population is split in two halves: one containing only feasible solutions and the other only infeasible ones. This algorithm uses the same encoding scheme and crossover operator as [58], but has not been applied on similar instances of the problem. In a general way, genetic algorithms give good results on partitioning problems, but at a very high computational cost.

Miscellaneous

Sequencing and Scheduling

The best-known *sequencing and scheduling problems* are the flow-shop, job-shop and open shop problems. The first paper applying an evolutionary algorithm to such a problem is [21]. Later, several other genetic algorithms have been proposed ([36,80] for example). One of the first efficient evolutionary algorithm for *job-shop* problems has been presented in [68] and improved in [20,91]. Comparisons done with other heuristics on benchmark problems show that sophisticated genetic algorithms (with the use of problem-adapted crossovers and hybridization) yield the best results for *flow-shop* and job-shop problems [1,24,42]. The *open shop problems* have less attracted researchers of the evolutionary algorithms' field, but a genetic algorithm has been proposed in [31,32]. An ant colony approach of job-shop problems has also been tested, in [14], but gave worse results than known genetic algorithms.

Steiner Trees

Only very few works deal with *Steiner trees* and evolutionary algorithms. Moreover, they consider different variants of this problem. The first paper [47] proposes a genetic algorithm with local optimization for determining minimum Steiner trees in the Euclidean plane. A solution is represented by the coordinates of the Steiner points. A comparison with *simulated annealing* and the Rayward-Smith–Care algorithm shows no significant differences. The problem of the rectilinear Steiner problem has been addressed in [53] with a specific coding and an adapted crossover. The minimal Steiner tree problem in graphs has attracted a little more interest. A standard genetic algorithm (with bit strings as chromosomes) that gave good results on the sparse graphs tested has been proposed in [55]. Later, H. Esbensen and P. Mazumder [28] designed a genetic algorithm in which the encoding method is based on the distance network heuristic. Improvements have been brought in [26] and [27], where there is also a comparison between different algorithms. But this genetic algorithm is not competitive with an efficient tabu search as presented in [41].

Conclusion

In this paper, some references on the evolutionary approaches that have been proposed up to 1998 for different combinatorial problems have been given. A general remark that can be made on these solution methods is that evolutionary algorithms in general, and genetic algorithms in particular, are not efficient for such problems if implemented too naively. To obtain an algorithm with good performances, it is necessary to make adjustments of the basic method. Moreover, knowledge about the problem considered is very often also needed, in order to design adapted operators.

Another remark concerns their competitivity compared to other heuristic methods. While evolutionary algorithms can quite easily be adapted to (almost) any problem, their running time is often quite high. Local search algorithms, like tabu search or simulated annealing, can also be adapted to the different combinatorial problems quite easily. If they are designed in an intelligent way, they are very often able to obtain better results than evolutionary algorithms. Moreover, they are usually faster. For some problems, specifically designed heuristics can use theoretical results about this problem, allowing them to obtain good results. In general, evolutionary algorithms are not competitive against (extended) local search or specific algorithms for small to medium size instances of combinatorial problems.

But this does not mean that population-based algorithms are not useful. In fact, the different approaches have various (dis)advantages, and the efficient algorithms that will be developed in the future will proba-

bly mix these different approaches. Such algorithms are usually called *'hybrid algorithms'* and have already been proposed for example for the traveling salesman problem [39] or the quadratic assignment problem [33], demonstrating their potentials.

See also

- Combinatorial Matrix Analysis
- Combinatorial Optimization Games
- Fractional Combinatorial Optimization
- Multi-objective Combinatorial Optimization
- Neural Networks for Combinatorial Optimization
- Replicator Dynamics in Combinatorial Optimization

References

1. Aarts EHL, van Laarhoven PJM, Lenstra JK, Ulder NLJ (1994) A computational study of local search algorithms for job shop scheduling. ORSA J Comput 6:118–125
2. Aggarwal CC, Orlin JB, Tai RP (1995) An optimized crossover for maximum independent set. Oper Res 45:226–234
3. Bäck T, Schütz M, Khuri S (1996) A comparative study of a penalty function, a repair heuristic, and stochastic operators with the set-covering problem. In: Alliot JM, Lutton E, Ronald E, Schoenhauer M, Snyers D (eds) Artificial Evolution: European Conf., Lecture Notes Computer Sci. Springer, Berlin, pp 3–20
4. Balas E, Niehaus W (1998) Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems. J Heuristics 4:107–122
5. Beasley J, Chu P (1996) A genetic algorithm for the set covering problem. Europ J Oper Res 94:392–404
6. Bersini H, Dorigo M, Langerman S, Seront G, Gambardella LM (1996) Results of the first international contest on evolutionary optimisation (1st ICEO). In: Proc. 1996 IEEE Internat. Conf. Evolutionary Computation, IEEE Press, pp 611–615
7. Braun H (1991) On solving travelling salesman problems by genetic algorithms. In: Schwefel H-P, Männer R (eds) Parallel Problem Solving from Nature. Lecture Notes Computer Sci. Springer, Berlin, pp 129–133
8. Bui TN, Eppley PH (1995) A hybrid genetic algorithm for the maximum clique problem. In: Eshelman LJ (ed) Proc. 6th Internat. Conf. Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 478–484
9. Bui TN, Moon BR (1995) On multi-dimensional encoding/crossover. In: Eshelman LJ (ed) Proc. 6th Internat. Conf. Genetic Algorithms, Morgan Kaufmann. San Mateo, CA, pp 49–56
10. Chu PC, Beasley JE (1998) Constraint handling in genetic algorithms: the set partitioning problem. J Heuristics 4:323–357
11. Chu PC, Beasley JE (1998) A genetic algorithm for the multidimensional knapsack problem. J Heuristics 4:63–86
12. Colorni A, Dorigo M, Maniezzo V (1991) Distributed optimization by ant coloniess. In: Varela F, Bourgine P (eds) Proc. ECAL91 - European Conf. Artificial Life. Elsevier, Amsterdam, pp 134–142
13. Colorni A, Dorigo M, Maniezzo V (1995) Algodesk: An experimental comparison of eight evolutionary heuristics applied to the quadratic assignment problem. Europ J Oper Res 81:188–205
14. Colorni A, Dorigo M, Maniezzo V, Trubian M (1994) Ant system for job-shop scheduling. JORBEL - Belgian J Oper Res Statist Comput Sci 34(1):39–53
15. Corcoran AL, Wainwright RL (1992) A genetic algorithm for packing in three dimensions. In: Proc. 1992 ACM/SIGAPP Symposium on Applied Computing SAC'92. ACM, New York, pp 1021–1030
16. Corcoran AL, Wainwright RL (1993) A heuristic for improved genetic bin packing. Techn Report UTULSA-MCS-93-08, Univ Tulsa, USA
17. Costa D, Hertz A (1997) Ants can color graphs. J Oper Res Soc 48:295–305
18. Costa D, Hertz A, Dubuis O (1995) Embedding a sequential procedure within an evolutionary algorithm for coloring problems in graphs. J Heuristics 1:105–128
19. Cung V-D, Mautor Th, Michelon Ph, Tavares A (1997) A scatter search based approach for the quadratic assignment problem. In: Proc. 1997 IEEE Internat. Conf. Evolutionary Computation. IEEE Press, pp 190–206
20. Davidor Y, Yamada T, Nakano R (1993) The ecological framework II: Improving GA performance with virtually zero cost. In: Forrest S (ed) Proc. 5th Internat. Conf. Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 171–176
21. Davis L (1985) Job shop scheduling with genetic algorithms. In: Grefenstette JJ (ed) Proc. 1st Internat. Conf. on Genetic Algorithms. Lawrence Erlbaum Ass., pp 136–140
22. Davis L (1991) Handbook of genetic algorithms. v. Nostrand Reinhold, Princeton, NJ
23. Dorigo M, Gambardella LM (1997) Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Trans Evolutionary Computation 1:53–66
24. Duvivier D, Preux Ph, Talbi E-G (1995) Stochastic algorithms for optimization and application to job-shop-scheduling. Techn Report LIL-95-5, Univ du Littoral, France
25. Eiben AE, van der Hauw JK (1997) Solving 3-SAT with adaptive genetic algorithms. In: Proc. 4th IEEE Conf. Evolutionary Computation, IEEE Press, pp 81–86
26. Esbensen H (1995) Computing near-optimal solutions to the Steiner problem in a graph using a genetic algorithm. Networks 26:173–185

27. Esbensen H (1995) Finding (near-)optimal Steiner trees in large graphs. In: Eshelman LJ (ed) Proc. 6th Internat. Conf. on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 485–491
28. Esbensen H, Mazumder P (1993) A genetic algorithm for the Steiner problem in a graph. Techn Report Univ Michigan, Ann Arbor
29. Falkenauer E (1996) A hybrid grouping genetic algorithm for bin packing. J Heuristics 2:5–30
30. Falkenauer E, Delchambre A (1992) A genetic algorithm for bin packing and line balancing. In: Proc. 1992 IEEE Internat. Conf. on Robotics and Automation. IEEE Computer Soc Press, New York, pp 1186–1192
31. Fang H-L (1994) Genetic algorithms in timetabling and scheduling. PhD Thesis, Univ. Edinburgh,).
32. Fang H-L, Ross P, Corne D (1993) A promising genetic algorithm approach to job-shop scheduling, re-scheduling, and open-shop scheduling problems. In: Forrest S (ed) Proc. 5th Internat. Conf. Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 375–382
33. Fleurent C, Ferland JA (1994) Genetic hybrids for the quadratic assignement problem. In: Pardalos PM, Wolkowicz H (eds) Quadratic assignment and related problems. DIMACS 16. Amer. Math. Soc., Providence, RI, pp 190–206
34. Fleurent C, Ferland JA (1996) Genetic and hybrid algorithms for graph coloring. In: Laporte G, Osman IH (eds) Metaheuristics in combinatorial optimization. Ann Oper Res. Baltzer, Basel, pp 437–461
35. Fleurent C, Ferland JA (1996) Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In: Johnson DS, Trick MA (eds) Cliques, coloring, and satisfiability. Amer. Math. Soc., Providence, RI p 619
36. Fox BR, McMahon MB (1991) Genetic operators for sequencing problems. In: Rawlins GJE (ed) Foundations of Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 284–300
37. Freisleben B, Merz P (1996) A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In: Proc. 1996 IEEE Internat. Conf. on Evolutionary Computation, IEEE Press, pp 616–621
38. Freisleben B, Merz P (1996) New genetic local search operators for the traveling salesman problem. In: Voigt H-M, Ebeling W, Rechenberg I, Schwefel H-P (eds) Proc. 4th Conf. on Parallel Problem Solving from Nature, Lecture Notes Computer Sci. Springer, Berlin, pp 890–899
39. Freisleben B, Merz P (1997) Genetic local search for the TSP: new results. In: Proc. 1997 IEEE Internat. Conf. on Evolutionary Computation, IEEE Press, pp 159–164
40. Gambardella L-M, Taillard ED, Dorigo M (1999) Ant colonies for the quadratic assignment problems. J Oper Res Soc 50:167–176
41. Gendreau M, Larochelle J-F, Sansó B (1998) A tabu search heuristic for the Steiner tree problem. GERAD G-98-01, Univ Montréal, Canada
42. Glass CA, Potts CN (1996) A comparison of local search methods for flow shop. In: Laporte G, Osman IH (eds) Metaheuristics in combinatorial optimization. Ann Oper Res. Baltzer, Basel, pp 489–509
43. Goodman ED, Tetelbaum AY, Kureichik VM (1994) A genetic algorithm approach to compaction, bin packing and nesting problems. Techn Report GARAGe94-4, Michigan State Univ
44. Gorges-Schleuter M (1989) Asparagos: An asynchronous parallel genetic optimization strategy. In: Schaffer JD (ed) Proc. 3rd Internat. Conf. on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 422–427
45. Hao JK (1995) A clausal genetic representation and its related evolutionary procedures for satisfiability problems. In: Pearson DW, Steele NC, Albrecht RF (eds) Proc. 2nd Internat. Conf. on Artificial Neural Networks and Genetic Algorithms. Springer, Berlin, pp 289–292
46. Hertz A, de Werra D (1987) Using tabu search techniques for graph coloring. Computing 39:345–351
47. Hesser J, Männer R, Stucky O (1991) On Steiner trees and genetic algorithms. In: Becker JD, Eisele I, Mündemann FW (eds) Parallelism, Learning, Evolution. Lecture Notes Artificial Intelligence. Springer, Berlin, pp 509–525
48. Hoff A, Lökketangen A, Mittet I (1996) Genetic algorithms for 0/1 multidimensional knapsack problems. Proc. Norsk Informatik Konferanse, NIK '96
49. Hoffman K, Padberg M (1993) Solving airline crew-scheduling problems by branch-and-cut. Managem Sci 39:657–682
50. Huang W-C, Kao C-Y, Horng J-T (1994) A genetic algorithm approach for set covering problems. Proc. First IEEE Internat. Conf. on Evolutionary Computation, IEEE Press, pp 569–574
51. Johnson DS, McGeoch LA (1997) The traveling salesman problem: A case study in local optimization. In: Aarts EHL, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, New York, pp 215–310
52. Jong KA De, Spears WM (1989) Using genetic algorithms to solve NP-complete problems. In: Schaffer JD (ed) Proc. 3rd Internat. Conf. on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 123–132
53. Julstrom BA (1993) A genetic algorithm for the rectilinear Steiner problem. In: Forrest S (ed) Proc. 5th Internat. Conf. Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 474–480
54. Kahng AB, Moon BR (1995) Toward more powerful recombinations. In: Eshelman LJ (ed) Proc. 6th Internat. Conf. on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 96–103
55. Kapsalis A, Rayward-Smith VJ, Smith GD (1993) Solving the graphical Steiner tree problem using genetic algorithms. J Oper Res Soc 44:397–406
56. Khuri S, Bäck T, Heitkötter J (1994) The zero/one multiple knapsack problem and genetic algorithms. Proc. 1994

ACM Symposium on Applied Computing. ACM, New York, pp 188–193
57. Khuri S, Schütz M, Heitkötter J (1995) Evolutionary heuristics for the bin packing problem. In: Pearson DW, Steele NC, Albrecht RF (eds) Proc. 2nd Internat. Conf. on Artificial Neural Networks and Genetic Algorithms. Springer, Berlin, pp 285–288
58. Laszewski G von (1991) Intelligent structural operators for the k-way graph partitioning problem. In: Belew R, Booker L (eds) Proc. 4th Internat. Conf. on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 45–52
59. Levine D (1994) A parallel genetic algorithm for the set partitioning problem. PhD Thesis Illinois Inst. Techn.,).
60. Liepins GE, Hilliard MR, Palmer MR, Morrow M (1987) Greedy genetics. In: Grefenstette JJ (ed) Proc. 2nd Internat. Conf. on Genetic Algorithms, Lawrence Erlbaum Ass.
61. Liepins GE, Hilliard MR, Richardson JT, Palmer M (1990) Genetic algorithms applications to set covering and traveling salesman problems. In: Brown DE, White CC (eds) Oper. Res. and Artificial Intelligence: The Integration of Problem-Solving Strategies. Kluwer, Dordrecht, pp 29–57
62. Maniezzo V (1998) Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. Techn Report Univ Bologna CSR 98-1
63. Maniezzo V, Colorni A (1998) The ant system applied to the quadratic assignment problem. IEEE Trans Knowledge and Data Engin
64. Maniezzo V, Colorni A, Dorigo M (1994) The ant system applied to the quadratic assignment problem. Techn Report IRIDIA/94-28, Univ Libre de Bruxelles, Belgium
65. Martello S, Toth P (1990) Lower bounds and reduction procedures for the bin packing problem. Discrete Appl Math 22:59–70
66. Mathias K, Whitley D (1992) Genetic operators, the fitness landscape and the traveling salesman problem. In: Männer R, Manderick B (eds) Parallel Problem Solving from Nature. Elsevier, Amsterdam, pp 219–228
67. Mühlenbein H, Gorges-Schleuter M, Krämer O (1988) Evolution algorithms in combinatorial optimization. Parallel Comput 7:65–85
68. Nakano R, Yamada T (1991) Conventional genetic algorithm for job shop problems. In: Belew R, Booker L (eds) Proc. 4th Internat. Conf. on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 474–479
69. Pargas RP, Jain R (1993) A parallel stochastic optimization algorithm for solving 2D bin packing problems. In: Proc. 9th Conf. on Artificial Intelligence for Applications, pp 18–25
70. Potvin J-Y (1996) Genetic algorithms for the traveling salesman problem. In: Laporte G, Osman IH (eds) Metaheuristics in combinatorial optimization. Ann Oper Res. Baltzer, Basel, pp 339–370
71. Potvin J-Y, Bengio S (1993) A genetic approach to the vehicle routing problem with time windows. Techn Report CRT-953, Univ Montréal
72. Reeves CR (ed) (1993) Modern heuristic techniques for combinatorial problems. Blackwell, Oxford
73. Reeves C (1996) Hybrid genetic algorithms for bin-packing and related problems. In: Laporte G, Osman IH (eds) Metaheuristics in combinatorial optimization. Ann Oper Res. Baltzer, Basel, pp 371–396
74. Richardson JT, Palmer MR, Liepins GE, Hilliard M (1989) Some guidelines for genetic algorithms with penalty functions. In: Schaffer JD (ed) Proc. 3rd Internat. Conf. on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 191–197
75. Rochat Y, Taillard ED (1995) Probabilistic diversification and intensification in local search for vehicle routing. J Heuristics 1:147–167
76. Sekharan DA, Wainwright RL (1993) Manipulating subpopulations in genetic algorithms for solving the k-way graph partitioning problem. In: Proc. 7th Oklahoma Symposium on Artificial Intelligence, pp 215–225
77. Smith D (1985) Bin packing with adaptive search. In: Grefenstette JJ (ed) Proc. 1st Internat. Conf. on Genetic Algorithms, Lawrence Erlbaum Ass., pp 202–207
78. Starkweather T, McDaniel S, Mathias K, Whitley D, Whitley C (1991) A comparison of genetic sequencing operators. In: Belew R, Booker L (eds) Proc. 4th Internat. Conf. on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 69–76
79. Stützle T, Hoos H (1997) The MAX-MIN ant system and local search for the traveling salesman problem. In: Proc. 1997 IEEE Internat. Conf. on Evolutionary Computation, IEEE Press, pp 308–313
80. Syswerda G (1991) Schedule optimization using genetic algorithms. In: Davis L (ed) Handbook Genetic Algorithms. v. Nostrand Reinhold, Princeton, NJ, p 333
81. Taillard E (1995) Comparison of iterative searches for the quadratic assignment problem. Location Sci 3:87–105
82. Tate DM, Smith AE (1995) A genetic approach to the quadratic assignment problem. Comput Oper Res 22:73–83
83. Thangiah SR (1995) Vehicle routing with time windows using genetic algorithms. In: Chambers L (ed) Applications handbook of genetic algorithms: new frontiers. CRC Press, Boca Raton, FL
84. Thangiah SR, Nygard KE, Juell PL (1991) GIDEON: A genetic algorithm system for vehicle routing with time windows. Proc. 7th IEEE Conf. Artificial Intelligence Applications. IEEE Computer Soc Press, New York, pp 322–328
85. Thangiah SR, Osman IH, Sun T (1995) Metaheuristics for vehicle routing problems with time windows. Techn Report Slippery Rock Univ
86. Thangiah SR, Osman IH, Vinayagamoorthy R, Sun T (1994) Algorithms for the vehicule routing problems with time deadlines. American J Math Management Sci 13:323–355
87. Thangiah SR, Vinayagamoorthy R, Gubbi A (1993) Vehicle routing with time deadlines using genetic and local algorithms. In: Forrest S (ed) Proc. 5th Internat. Conf. Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 506–513

88. Thiel J, Voss S (1994) Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithm. INFOR special issue: Knapsack, packing and cutting, Part II 32:226–242
89. Ulder NLJ, Aarts EHL, Bandelt HJ, van Laarhoven PJM, Pesch E (1991) Genetic local search algorithms for the traveling salesman problems. In: Schwefel H-P, Männer R (eds) Parallel Problem Solving from Nature. Lecture Notes Computer Sci. Springer, Berlin, pp 109–116
90. Vink M (1997) Solving combinatorial problems using evolutionary algorithms. Techn Report Leiden Univ, Netherlands
91. Yamada T, Nakano R (1992) A genetic algorithm applicable to large-scale job-shop problems. In: Männer R, Manderick B (eds) Parallel Problem Solving from Nature, 2. Elsevier, Amsterdam, pp 281–290

Extended Cutting Plane Algorithm

Claus Still, Tapio Westerlund
Process Design Laboratory, Åbo Akademi University, Åbo, Finland

MSC2000: 90C11, 90C26

Article Outline

Keywords and Phrases
Introduction
Formulation
Methods
  Calculating Sufficiently Large α-values
  Handling Infeasible MILP Problems
Convergence
  Cycling
  Convergence to a Feasible Point
  Convergence to the Optimal Solution
Cases
Conclusions
References

Keywords and Phrases

Mixed integer nonlinear programming; Extended cutting plane; Quasi-convex functions; Feasible underestimator

Introduction

The α-ECP (Extended Cutting Plane) algorithm [13,14] is an algorithm for solving quasi-convex MINLP (Mixed Integer Nonlinear Programming) problems. The algorithm approximates the feasible region with linear approximations and solves a sequence of MILP problems based on these approximations. There are several other similar methods, for instance the Generalized Benders Decomposition method [7], the Outer Approximation method [4], the LP/NLP Based Branch-and-Bound method [8], the Linear Outer Approximation method [5] and the Sequential Cutting Plane method [10]. A good overview of MINLP algorithms and applications is given in [6]. Most of the other methods iteratively solve both NLP and MILP problems, while the α-ECP method only solves MILP problems. The size of the MILP problems grow in each iteration, so efficient algorithms of this type require efficient MILP solvers.

Most of the MINLP methods can only ensure global convergence for convex MINLP problems. Different heuristic procedures for some of the above algorithms have been introduced for the non-convex case, e. g. [11]. Although these methods perform quite well in different applications, convergence towards the optimal solution cannot generally be ensured by these algorithms for non-convex problems.

There are also some MINLP global optimization methods [1,2,9,12]. In these algorithms the function space is separated for the continuous and discrete variables and the discrete variables can only occur in linear space. The α-ECP method can solve quasi-convex problems where the discrete variables are involved in nonlinear equations as well.

The α-ECP method has also been further extended to global optimization problems through the use of transformation techniques for problems containing signomial terms, see [15].

Formulation

The α-ECP algorithm can be used to solve problems of the form

$$\begin{aligned} &\min\ c^T z, \\ &\text{s.t. } g(z) \le 0, \\ &\quad Az \le a, \\ &\quad Bz = b, \\ &\quad z \in X \times Y, \end{aligned} \tag{1}$$

where c is a vector of constants, $z = (x, y)$ consists of a vector x of continuous variables in $\mathbf{R}^n$ and a vector y of integer variables in $\mathbf{Z}^m$ and $g(z)\mathbf{R}^n \times \mathbf{Z}^m\mathbf{R}^p$ is a vector of continuous differentiable quasi-convex functions defined on the set $X \times Y$ having non-zero gradients in the infeasible region of (1). The feasible region of (1) is assumed to be nonempty. Furthermore, X is a compact convex set $X \subset \mathbf{R}^n$ and Y is a finite discrete set $Y \subset \mathbf{Z}^m$.

The matrices A and B and vectors a and b are used to define the linear constraints of the problem and are of suitable dimensions.

The α-ECP method guarantees global optimal solutions for MINLP problems having a linear objective function and differentiable quasi-convex constraints. The linear objective function is not too restrictive since most optimization problems having a nonlinear objective $f(z)$ can be rewritten as a problem involving an additional variable u and an additional constraint

$$f(z) - u \leq 0. \tag{2}$$

The new problem, then, will be to minimize u subject to the original constraints and the additional constraint (2). Note, however, that this is not, in general, possible for quasi-convex objectives since $f(z) - u$ is not necessarily quasi-convex when $f(z)$ is quasi-convex. An extension to handle quasi-convex objective functions, rigorously, is given in [13]

Methods

The algorithm solves the problem (1) by approximating the maximal violated nonlinear function with a linear function

$$l(z) = g_i(z^k) + \alpha \cdot \nabla g_i(z^k)^T(z - z^k) \tag{3}$$

in the current iterate z^k, where $i = \arg\max_i\{g_i(z^k)\}$. To simplify notation, let $g_k = g_i(z^k)$. Furthermore, if the linearization added to the MILP problem is the jth linearization, let $\bar{g}_j = g_i(z^k)$, $\bar{g}_j(z) = g_i(z)$, $\nabla\bar{g}_j = \nabla g_i(z^k)$ and $\bar{z}^j = z^k$ where i is defined as above. The α values change from iteration to iteration so to be able to reference the value of the jth constant in iteration k the α constants are replaced with $\alpha_j^{(k)}$. Thus the linearization (3) is redefined so that in iteration k the jth linear approximation $l_j^{(k)}$ will be

$$l_j^{(k)}(z) = \bar{g}_j + \alpha_j^{(k)} \cdot (\nabla\bar{g}_j)^T(z - \bar{z}^j)$$

and the algorithm adds the linear constraint

$$l_j^{(k)}(z) \leq 0 \tag{4}$$

to the MILP problem. The α constants initially have the value $\alpha_j^{(k)} = 1$ and they are either left unchanged or increased by a factor in each iteration. The algorithm then iteratively adds more and more constraints to an MILP problem originally consisting of only the linear constraints $Az \leq a$ and $Bz = b$ from (1). In iteration k it thus solves the MILP problem

$$\begin{aligned} \min\ c^T z\,,\ \text{s.t.}\ & l_j^{(k)}(z) \leq 0\,,\ \ j = 1, \ldots, L_k\,, \\ & Az \leq a\,,\ \ Bz = b\,,\ \ z \in X \times Y\,, \end{aligned} \tag{5}$$

where L^k is the number of linearizations in iteration k. The solution to this MILP problem will be the new iteration point. Using this point a new linearization is added to the MILP problem or one or several of the α constants are updated. The procedure is then repeated until a feasible point of (1) is found. A point is considered feasible if

$$g_i(z) \leq \epsilon_g\,,\ \ i = 1, \ldots, p, \tag{6}$$

for some prespecified tolerance ϵ_g. Note that the constraints $Az \leq a$ and $Bz = b$ are automatically satisfied since the current iteration point is the solution to (5). The idea of finding a feasible and optimal point by solving a sequence of MILP problems is the same as in the classical Kelley's cutting plane method for NLP problems. However, Kelley considered only the continuous case using LP subsolutions. Furthermore, Kelley's cutting plane algorithm assumes that the linearizations will always be valid underestimators of the corresponding nonlinear functions. This is true if the functions are convex, since for convex functions it holds that

$$g_i(z^k) + \nabla g_i(z^k)^T(z - z^k) \leq g_i(z) \tag{7}$$

for all $z, z^k \in X \times Y$. Thus, $l_j^{(k)}(z) \leq 0$ whenever $\bar{g}_j(z) \leq 0$ even when $\alpha_j^{(k)} = 1$.

Unfortunately (7) does not generally hold for quasi-convex functions. It is possible that the linear approximations are not valid underestimators of the corresponding nonlinear function and thus the constraint $l_j^{(k)}(z) \leq 0$ may cut away parts of the feasible region. To avoid this problem the α constants have been introduced. By using sufficiently large α values it is ensured that $l_j^{(k)}(z) \leq 0$ whenever $\bar{g}_j(z) \leq 0$ holds. The

linearizations will then be valid outer approximations of the feasible region of (1).

Generally it is not known how large the α constants should be. Instead an updating strategy is used. The α values are checked and updated in each iteration if they turn out to be too small. The updated value is obtained by multiplying the current value with a constant greater than one. When the current MILP solution is a feasible solution in (1) and all α constants are large enough, the optimal solution to (1) has been found and the algorithm terminates.

Calculating Sufficiently Large α-values

Since it is not known beforehand how large α values to use, it is shown below how to obtain large enough values to ensure ϵ-optimality. As previously mentioned, parts of the feasible region may be cut out when linearizing the quasi-convex functions, if the value of the α constant is not increased.

If a sufficiently large α value can be found so that the linearization is a *global underestimator* of the corresponding nonlinear function in the entire feasible region, the linearization should satisfy

$$\bar{g}_j + \alpha_j^{(k)} \cdot (\nabla \bar{g}_j)^T (z - \bar{z}^j) \le \bar{g}_j(z)\ , \quad \forall z \in \{z \in X \times Y : \bar{g}_j(z) \le 0\}. \tag{8}$$

A weaker condition is that the inequality (8) is satisfied only for all current iteration points. If this condition is satisfied, the linearization is called a *local underestimator*. Thus, the linearization is a local underestimator if it satisfies the following inequality in iteration k

$$\bar{g}_j + \alpha_j^{(k)} \cdot (\nabla \bar{g}_j)^T (z^k - \bar{z}^j) \le \bar{g}_j(z^k)\ , \quad j = 1, \ldots, L_k. \tag{9}$$

This inequality is easy to check in each iteration. If there is some α constant $\alpha_j^{(k)}$ that does not satisfy (9) then it is updated by multiplying the constant with β. The update formula is thus

$$\alpha_j^{(k+1)} = \begin{cases} \beta \cdot \alpha_j^{(k)}, & l_j^{(k)}(z^k) > \bar{g}_j(z^k)\ , \\ \alpha_j^{(k)}, & \text{otherwise.} \end{cases} \tag{10}$$

The β constant is a prespecified constant ($\beta > 1$). The concept of local underestimators is now extended to *feasible underestimators*. A linearization is called a feasible underestimator if it approximates the entire feasible region. Thus, for such linearizations, it holds that

$$\bar{g}_j + \alpha_j^{(k)} \cdot (\nabla \bar{g}_j)^T (z - \bar{z}^j) \le 0, \quad \forall z \in \{z \in X \times Y : \bar{g}_j(z) \le 0\}. \tag{11}$$

This is a much more strict requirement since a local underestimator need only underestimate the nonlinear function in a finite set of infeasible points. But condition (11) is weaker than the condition for global underestimators (8) since a feasible underestimator does not necessarily have to underestimate all points in the feasible region of the corresponding nonlinear function. It is only required that $l_j^{(k)} \le 0$ in this region. In practice, a feasible underestimator needs to underestimate the entire boundary or, more precisely, the convex hull of the feasible region.

To see how to get a feasible underestimator, a new constant $h_j^{(k)}$ is introduced where, as previously with the α constants, the constant will be used in the jth linearization and k stands for the value of the constant in the kth iteration. The constant is defined as

$$h_j^{(k)} = \frac{\bar{g}_j}{\alpha_j^{(k)}}. \tag{12}$$

Since (11) can be divided by $\alpha_j^{(k)}$, the inequality becomes

$$h_j^{(k)} + (\nabla \bar{g}_j)^T (z - \bar{z}^j) \le 0 \tag{13}$$

and moreover, because $\alpha_j^{(k)} \ge 1$, it holds that

$$h_j^{(k)} \le \bar{g}_j. \tag{14}$$

The level sets of quasi-convex functions are convex, which means that if the constant parameter $h_j^{(k)}$ is replaced with zero, then the linearization (13) is always an outer approximation of the feasible region. In fact, the linearization is then an approximation of an even larger region

$$\{z \in X \times Y : \bar{g}_j(z) \le \bar{g}_j(z^k)\}$$

containing the feasible region. Thus, if $h_j^{(k)}$ is sufficiently small, (13) is an approximation of the feasible region. In practice the h constants should satisfy

$$h_j^{(k)} \le \epsilon_h\ , \quad \forall j = 1, \ldots, L_k.$$

This is the same as requiring that

$$\alpha_j^{(k)} \geq \frac{\bar{g}_j}{\epsilon_h}, \quad \forall j = 1, \ldots, L_k, \tag{15}$$

which can easily be seen from (12). In (15) it can be seen that there is an important connection between *sufficiently large* α values and the value of the nonlinear function in the linearization point ($\bar{g}_j$). The larger the term $\bar{g}_j$ is, the larger the constant α has to be, to be sufficiently large. One could use the same updating scheme (10) as was used for obtaining a local underestimator, but to speed up the process a new updating factor $\gamma > 1$ (and $\gamma \geq \beta$) is introduced. This constant is used to update the α values if the corresponding linearizations are not feasible underestimators.

Whenever the algorithm finds a feasible point it checks that all linearizations are feasible underestimators, i. e. that (15) holds. If there is some $\alpha_j^{(k)}$ constant that violates this inequality, the value of that constant is updated by multiplying it with γ. Thus, the α constants will be updated according to

$$\alpha_j^{(k+1)} = \begin{cases} \gamma \cdot \alpha_j^{(k)}, & \alpha_j^{(k)} < \bar{g}_j/\epsilon_h, \\ \alpha_j^{(k)}, & \text{otherwise.} \end{cases} \tag{16}$$

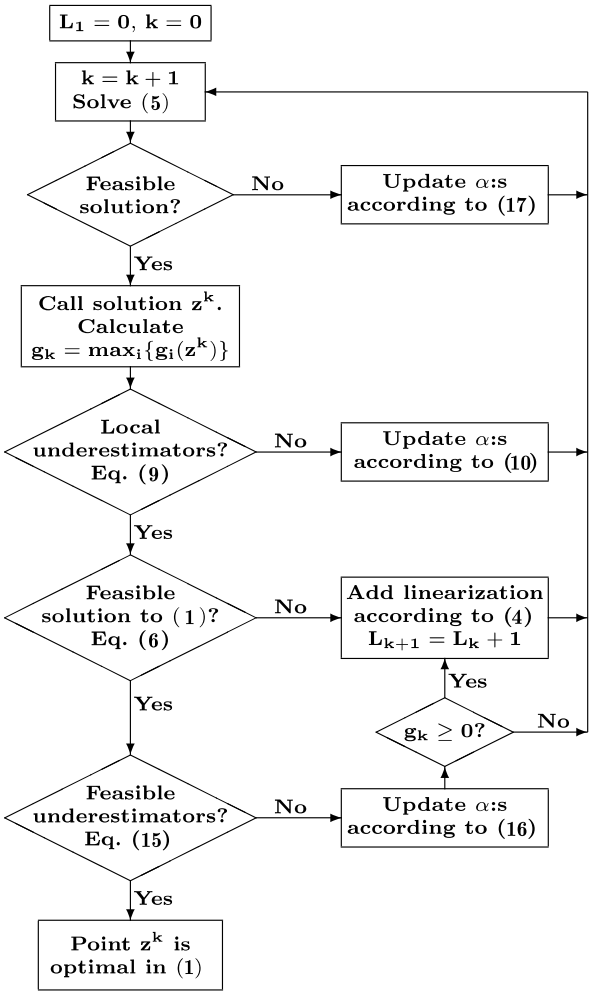
In fact it would be sufficient to require that the linear underestimators should not cut away the optimal point z^*, i. e. that $l_j^{(k)}(z^*) \leq 0$. The algorithm would then terminate in considerably fewer iterations, but since the optimal solution z^* is not known it is very difficult to check this requirement. The same difficulty also appears if the algorithm would be based on global underestimators of the type (8). However, as will follow, global convergence of the algorithm towards the optimal solution can be guaranteed by using local and feasible underestimators. That is why the concepts of local and feasible underestimators have been introduced.

Handling Infeasible MILP Problems

It is possible that the linearizations cut out enough of the feasible region of (1) to make the corresponding MILP problem infeasible. Then there would be no new iteration point and the algorithm would not be able to continue. The solution to this problem is to update all α values and solve the MILP problem again, after updating the values. If there is still no feasible point, this process is repeated until a feasible point is obtained. There exist large enough α values to make the MILP problem feasible, since the nonlinear problem (1) was assumed to be feasible. Thus, if the MILP problem is infeasible, the α update will be

$$\alpha_j^{(k+1)} = \beta \cdot \alpha_j^{(k)}, \quad j = 1, \ldots, L_k. \tag{17}$$

To illustrate the algorithm, a flowsheet of the algorithm is given below.



Extended Cutting Plane Algorithm, Figure 1

Convergence

Convergence properties of the algorithm are now studied. Below it is proven that the algorithm converges towards the optimal solution for the quasi-convex problem (1). There are three important properties which

are needed to prove convergence. First, the algorithm will never return to the same point if it is infeasible, secondly the generated points will converge to a feasible solution and finally this feasible solution will be the global optimal solution to the original quasi-convex problem (1).

Cycling

First it is shown that the algorithm never returns to the same point if it is infeasible, i. e., that cycling is not possible. Note that compactness or quasi-convexity of the constraint functions are unnecessary to prove this theorem.

Theorem 1 *If, in iteration k, the current point z^k is not feasible, then all new points generated by the algorithm will be different from z^k.*

Proof If z^k is infeasible, then $g_k > 0$ and a linearization is added to the MILP problem. If this linearization was the jth one added, then all new points z^l generated by the algorithm will satisfy

$$\bar{g}_j + \alpha_j^{(l)} \cdot (\nabla \bar{g}_j)^T (z^l - \bar{z}^j) \leq 0\,, \quad l > k. \tag{18}$$

Since $z^l = z^k\ (= \bar{z}^j)$ does not satisfy the inequality (18), all new points will be different from z^k. □

It immediately follows that all previous points generated by the algorithm are different from z^k as well.

Corollary 1 *If the current point z^k is infeasible, then z^k is different from all previous points.*

Proof If there is a z^j, $j < k$ such that $z^j = z^k$ then z^j would be a point not satisfying the previous theorem.□

Convergence to a Feasible Point

Convergence to a feasible point for discrete problems is directly ensured by the above cycling theorem. By assumption, there are only a finite number of points in Y, and there is at least one feasible point. Consequently, if the algorithm does not find any of the feasible points in finite time, it would have to repeat an infeasible point after generating at most $|Y|$ iteration points, which is not possible under the cycling theorem.

Convergence in the mixed integer case can be proven by utilizing the fact that the points x^k are taken on a compact set X, and the set Y is finite. This implies that any infinite sequence of points $\{z^k = (x^k, y^k), k \in \mathcal{K}\}$ taken on the set $X \times Y$ has a subsequence with a limit point. The following theorem shows that any limit point will be a feasible point which is a property required for convergence. Note that the quasi-convex property of the nonlinear functions is not required to prove convergence of the algorithm. Quasi-convexity is only required to ensure a global optimal solution.

The algorithm ensures that $\alpha_j^{(k)} \geq \bar{g}_j/\epsilon_h$, but for simplicity assume that equality holds for those j where $\bar{g}_j \geq \epsilon_h$. Then the constant $h_j^{(k)}$ satisfies

$$\min\,(\epsilon_h, \bar{g}_j) \leq h_j^{(k)} \leq \bar{g}_j. \tag{19}$$

This follows directly from (14) and the fact that (15) is already satisfied for $\alpha_j^{(k)} = 1$ if $\bar{g}_j < \epsilon_h$.

Below it is proven that any limit point is a feasible point.

Theorem 2 *Suppose that the α-ECP algorithm generates an infinite sequence of points $\{z^k,\ k \in \mathcal{K}\}$. Then the limit point of any convergent subsequence $\bar{\mathcal{K}} \subset \mathcal{K}$ is feasible.*

Proof Assume there is a convergent subsequence $\{z^k, k \in \bar{\mathcal{K}}\}$ with a limit point that is not feasible. Then $\lim_{k\in\bar{\mathcal{K}}} g_k = \epsilon > 0$ and one can find a constant M such that

$$h_j^{(k)} \geq \min\,(\epsilon_h, \frac{\epsilon}{2})\,,\ \forall j > L_M\,,\ \forall k > M,$$

by (19). Since subsequent points z^k are solutions to a linear program containing the linearization (13) it holds for all k that

$$\begin{aligned} 0 &\geq h_j^{(k)} + (\nabla \bar{g}_j)^T (z^k - \bar{z}^j) \\ &\geq h_j^{(k)} - ||\nabla \bar{g}_j|| \cdot ||z^k - \bar{z}^j|| \end{aligned}$$

when $j = 1, \ldots, L_k$. Define G as the maximal norm of the gradient of $g(z)$ in $X \times Y$. That is, $G = \max\{||\nabla g_i(z)||, z \in X \times Y, i = 1, \ldots, p\}$. Then

$$||z^k - \bar{z}^j|| \geq \frac{h_j^{(k)}}{||\nabla \bar{g}_j||} \geq \frac{\min\,(\epsilon_h, \epsilon/2)}{G} > 0$$

when $k > M$ and $j > L_M$. This implies that the sequence is not a Cauchy sequence and thus not convergent, which is a contradiction since it was assumed that the sequence $\{z^k, k \in \bar{\mathcal{K}}\}$ was convergent. □

Convergence to the Optimal Solution

Finally convergence of the algorithm to the global optimal solution of (1) is shown.

First note that the algorithm will terminate in finite time at a point where all underestimators are ϵ-feasible underestimators, i. e. (15) is satisfied. This follows from the convergence theorem. Since any convergent subsequence has a limit point that is feasible, it means that the entire sequence of points will also converge to a feasible point. Thus, there is a tail of the sequence, say $\{\bar{z}^j, j = M, \dots\}$, where the initial α values of the corresponding linearizations directly satisfy (15). This is true for those M values that satisfy $\bar{g}_j \leq \epsilon_h,\ \forall j > M$. These α values will remain constant in subsequent iterations. On the other hand, after reaching a feasible point ($\bar{g}_j \leq \epsilon_g$), the old constants $\alpha_j^{(k)},\ j = 1, \dots, M$ can only be updated a finite number of times before being sufficiently large to satisfy (15). Therefore the algorithm will eventually reach a feasible point where all linearizations are ϵ-feasible underestimators and the algorithm terminates. It remains to see if this point is also the optimal solution.

To prove that the obtained solution is the optimal solution one needs to assume that all linear constraints are feasible underestimators according to (11). This is in general true if $h_j^{(k)} = 0$. However, in the actual algorithm it was only required that $h_j^{(k)} \leq \epsilon_h$. Thus, the actual solution obtained by the algorithm can only be ensured to be ϵ-optimal.

Theorem 3 *Assume that the α-ECP algorithm converges to a feasible solution z^∞ and that all linearizations are feasible underestimators according to (11). Then z^∞ is an optimal point in (1) and $Z(z^\infty)$, where $Z(z) = c^T z$, is the optimal solution of (1).*

Proof Denote the feasible region of (1) with Ω, the feasible region of the MILP problem that was solved to obtain z^∞ with Ω^∞ and an optimal point of (1) with z^*. By (11) it holds that $\Omega \subset \Omega^\infty$ and thus

$$Z(z^\infty) \leq Z(z^*). \tag{20}$$

On the other hand z^∞ was feasible in (1) and thus

$$Z(z^*) \leq Z(z^\infty)\ . \tag{21}$$

From (20) and (21) one gets that $Z(z^*) = Z(z^\infty)$ and thus $Z(z^\infty)$ is the optimal solution to (1) and z^∞ is an optimal point in (1). □

Cases

The algorithm is demonstrated on a quasi-convex integer problem. In these, as well as in other test runs, it has turned out that a suitable choice of β and γ is $\beta = 1.3$ and $\gamma = 10$. The ϵ-tolerances in these examples are $\epsilon_g = \epsilon_h = 0.001$.

Consider the problem

$$\begin{aligned} &\min\ 3y_1 + 2y_2, \\ &\text{s.t. } 3.5 - y_1 y_2 \leq 0, \\ &\quad y \in \{1, \dots, 5\}^2. \end{aligned} \tag{22}$$

The optimal solution to this problem is $y = (2, 2)$, which can be seen from the Fig. 2.

The steps executed by the α-ECP algorithm are:

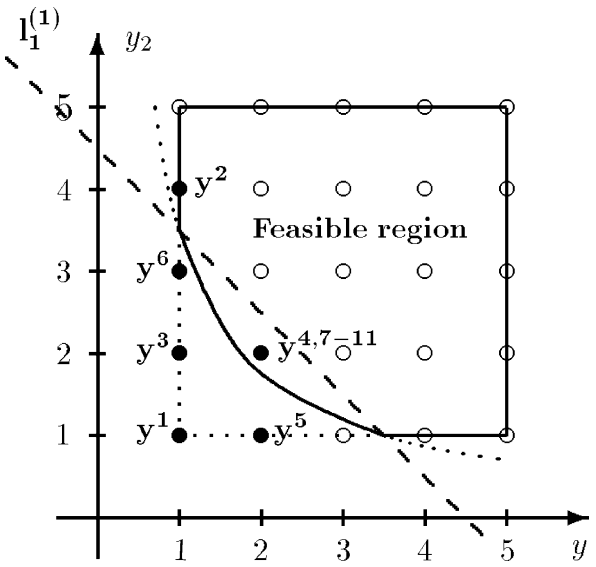
Iteration 1. Solve problem

$$\begin{aligned} &\min\ 3y_1 + 2y_2, \\ &\text{s.t. } y \in \{1, \dots, 5\}^2. \end{aligned}$$

The solution is $y^1 = (1, 1)$. A linearization in this point

$$2.5 + \alpha_1^{(1)} \begin{pmatrix} -1 & -1 \end{pmatrix} \begin{pmatrix} y_1 - 1 \\ y_2 - 1 \end{pmatrix} \leq 0$$

is added to the MILP problem according to (4). Set $\alpha_1^{(1)} = 1$. The linearization $l_1^{(1)}$ is shown in the



Extended Cutting Plane Algorithm, Figure 2
Feasible region of (22)

Fig. 2. As can be seen from the figure, the linearization cuts away the optimal solution to the problem.

Iteration 2. The solution to the new MILP problem is $y^2 = (1, 4)$. This point is a feasible solution to the INLP problem. The linearization satisfies the requirements of a local underestimator but is not a feasible underestimator. Observe, that without the concept of feasible underestimators the algorithm would stop here at a nonoptimal point. However, in order to ensure the linear function be a feasible underestimator, the α constant is updated according to (16) and $\alpha_1^{(3)} = 10$. Since $\max_i\{g_i(z^k)\} < 0$, no additional linearization is added.

Iteration 3. The solution to the new MILP problem is $y^3 = (1, 2)$. A new linearization at this point is added to the MILP problem ($\alpha_2^{(3)} = 1$)

$$1.5 + \alpha_2^{(3)} \begin{pmatrix} -2 & -1 \end{pmatrix} \begin{pmatrix} y_1 - 1 \\ y_2 - 2 \end{pmatrix} \leq 0.$$

Iteration 4. The MILP solution is $y^4 = (2, 2)$ which is feasible, however, neither linearization is a feasible underestimator, so the α values are updated using (16). The new values are $\alpha_1^{(5)} = 100$ and $\alpha_2^{(5)} = 10$.

Iteration 5. The solution of the modified MILP problem is $y^5 = (2, 1)$. Since it is infeasible, a new linearization

$$1.5 + \alpha_3^{(5)} \begin{pmatrix} -1 & -2 \end{pmatrix} \begin{pmatrix} y_1 - 2 \\ y_2 - 1 \end{pmatrix} \leq 0$$

is added, where $\alpha_3^{(5)} = 1$.

Iteration 6. The MILP solution is $y^6 = (1, 3)$ which is also infeasible. A new linearization

$$0.5 + \alpha_4^{(6)} \begin{pmatrix} -3 & -1 \end{pmatrix} \begin{pmatrix} y_1 - 1 \\ y_2 - 3 \end{pmatrix} \leq 0$$

is added ($\alpha_4^{(6)} = 1$).

Iteration 7. The MILP solution is again the feasible solution $y^7 = (2, 2)$. The linearizations are not feasible underestimators and thus the α values are updated. The new α values are $\alpha_1^{(8)} = 1000$, $\alpha_2^{(8)} = 100$ and $\alpha_3^{(8)} = \alpha_4^{(8)} = 10$.

Iterations 8–10. The new solutions to the MILP problems are still $y^{8,9,10} = (2, 2)$ but the α values are not large enough to guarantee that the linearizations are feasible underestimators. Therefore the α constants are updated.

Iteration 11. The solution is $y^{11} = (2, 2)$ and all linearizations are feasible underestimators. The algorithm terminates with $y^* = (2, 2)$.

The algorithm thus returns the global solution $y^* = (2, 2)$ to (22) with the optimal value $Z(2, 2) = 10$. The final MILP problem solved in iteration 11 is

$$\begin{aligned} \min\ & 3y_1 + 2y_2, \\ \text{s.t. } & 2.5 + 10{,}000(2 - y_1 - y_2) \leq 0, \\ & 1.5 + 10{,}000(4 - 2y_1 - y_2) \leq 0, \\ & 1.5 + 10{,}000(4 - y_1 - 2y_2) \leq 0, \\ & 0.5 + 1000(6 - 3y_1 - y_2) \leq 0, \\ & y \in \{1, \ldots, 5\}^2. \end{aligned}$$

Conclusions

The above algorithm has several advantages when compared to other similar algorithms for solving MINLP problems. At each iteration, the procedure only solves MILP subproblems and is thus a competitive alternative to algorithms where only NLP problems or both NLP and MILP problems are solved in each iteration.

One consequence is that since only MILP problems are solved in each iteration, the nonlinear constraints need not be calculated at relaxed values of the integer variables. It can be very difficult to calculate the value in a relaxed point if, for instance, there are binary variables that represent the existence of units in a process and the constraints are evaluated by simulating the result of having those units present or not. Then it may sometimes be impossible to evaluate the constraints if the integer variables are relaxed.

The α-ECP algorithm also solves MINLP problems that have general integer variables, not only binary variables. Also, no integer cuts are needed to ensure convergence. This is not the case with all outer approximation MINLP methods. In addition, the proposed algorithm ensures global convergence for quasi-convex MINLP problems.

Cutting plane methods are claimed to have slow convergence. This, generally, is not the case if the convergence rate is measured as the number of nonlinear function evaluations. Numerical experience with the algorithm indicates that there are many cases where the

number of function evaluations are even magnitudes lower than for competing algorithms that solve both MINLP and NLP subproblems. This is a significant advantage if evaluation of the constraints is the most time-consuming part of the problem.

Very good results for the algorithm for a set of difficult block optimization problems is reported in [3]. Therefore, the algorithm also appears to work very well on problems where the problem complexity is dominated by the integer part.

References

1. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP problems in process synthesis and design. Comput Chem Eng 21:445–450
2. Androulakis IP, Maranas CD, Floudas CA (1995) α-BB: A global optimization method for general constrained nonconvex problems. J Glob Optim 7:337–363
3. Castillo I, Westerlund J, Emet S, Westerlund T (2005) Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods. Comput Chem Eng 30:54–69
4. Duran MA, Grossmann IE (1986) An outer approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
5. Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Math Program 66:327–349
6. Floudas CA (1995) Nonlinear and mixed-integer optimization, fundamentals and applications. Oxford University Press, Oxford
7. Geoffrion AM (1972) Generalized Benders Decomposition. J Optim Theory Appl 10:237–260
8. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput Chem Eng 16:937–947
9. Ryoo HS, Sahinidis NV (1995) Global optimization of nonconvex NLPs and MINLPs with applications in process design. Comput Chem Eng 19:551–566
10. Still C, Westerlund T (2006) Solving convex MINLP optimization problems using a sequential cutting plane algorithm. Comput Optim Appl 34:63–83
11. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer-approximation method for MINLP optimization. Comput Chem Eng 14:769–782
12. Visweswaran V, Floudas CA (1996) New formulations and branching strategies for the GOP algorithm. In: Grossmann IE (ed) Global Optimization in Engineering Design. Kluwer, Dordrecht, pp 75–109
13. Westerlund T, Pörn R (2002) Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. Optim Eng 3:253–280
14. Westerlund T, Skrifvars H, Harjunkoski I, Pörn R (1998) An extended cutting plane method for a class of non-convex MINLP problems. Comput Chem Eng 22:357–365
15. Westerlund T (2006) Some transformation techniques in global optimization. In: Liberti L, Maculan N (eds) Global Optimization, From Theory to Implementation. Springer, Berlin, pp 45–74

Extension of the Fundamental Theorem of Linear Programming

ALLEN HOLDER[1,2]

[1] Department of Mathematics, Trinity University, San Antonio, USA

[2] Departments Radiological Sciences, University of Texas, Health Science Center in San Antonio, San Antonio, USA

MSC2000: 90C05

Article Outline

Introduction
Definitions
Methods
  Extension of the Fundamental Theorem
Conclusions
References

Introduction

The roots of contemporary optimization are found in the works of some of the greatest minds in mathematics: Lagrange, Weierstrass, Caratheodory and von Neumann to name a few. However, it was the work of George Dantzig in the late 1940s that catalyzed much of the research comprising the core of optimization, mathematical programming and operations research. Dantzig developed the simplex algorithm and proved the discipline's fundamental theorem, and in doing so he became known as the father of linear programming. The simplex algorithm is arguably one of the most important discoveries of the 20th century. Applications and subsequent theoretical developments have flourished ever since.

The Fundamental Theorem of Linear Programming states that every well-posed linear program has a basic optimal solution. Some related theory shows that this

result can be interpreted algebraically (as stated), geometrically (replace basic with vertex), or in terms of convex analysis (replace basic with extreme point). This multi-faceted perspective is a byproduct of the linearity since the supporting polyhedral analysis shows that all three constructs are one and the same. However, a natural question is whether or not the algebraic interpretation extends beyond the confines of linearity. The answer is yes, and the goal of this article is to explain how a broader perspective is achieved.

Definitions

We consider the following multiple objective extension of the standard form linear program

$$\text{(MOP)}\ \min\{F(x)\colon Ax = b\}\,,$$

where each component function F_i of $F\colon \mathbb{R}^n \to \mathbb{R}^p$ has the form $F_i(x) = \sum_{j=1}^{n} f_{(i,j)}(x_j)$. Notice that each F_i is separable in the sense that $f_{(i,j)}$ only depends on the component x_j. The codomain of F is ordered lexicographically so that the minimization problem is well defined. Unlike a standard form linear program, (MOP) does not have inequality constraints. However, the lexicographic ordering allows us to model inequalities through the objective function. Let $D\colon \mathbb{R}^n \to \mathbb{R}$ be defined by

$$D(x) = \sum_{j=1}^{n} \max\{-x_j, 0\}\,,$$

so that

$$\begin{aligned}\arg\min\left\{c^{\mathrm{T}}x\colon Ax = b, x \geq 0\right\} \\ = \arg\min\left\{\begin{pmatrix} D(x) \\ c^{\mathrm{T}}x \end{pmatrix} : Ax = b\right\}\end{aligned} \quad (1)$$

and

$$\begin{aligned}\text{(LP)}\ \min\left\{c^{\mathrm{T}}x\colon Ax = b, x \geq 0\right\} \\ = \min\left\{\begin{pmatrix} D(x) \\ c^{\mathrm{T}}x \end{pmatrix} : Ax = b\right\}\,,\end{aligned} \quad (2)$$

provided that (LP) is well-posed. Hence (MOP) is an extension of (LP). We make the tacit assumption that A has full row rank.

For $x \in \mathbb{R}^n_+ \equiv \{x \in \mathbb{R}\colon x \geq 0\}$, we define $B(x) = \{i\colon x_i > 0\}$ and $N(x) = \{i\colon x_i = 0\}$. The argument is assumed when it is clear. A set subscripts on a vector (matrix) indicates the subvector (submatrix) whose components (columns) are indexed by the set. For example,

$$\begin{aligned} Ax &= \begin{bmatrix} 1 & 1 & 0 \\ 2 & -1 & -1 \end{bmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 2 & -1 \end{bmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix}\begin{pmatrix} 0 \end{pmatrix} \\ &= A_B x_B + A_N x_N \\ &= A_B x_B\,. \end{aligned}$$

A basic feasible solution is an $x \in \mathbb{R}^n_+$ satisfying $Ax = b$ so that the columns of A_B are linearly independent. A basic feasible solution within the optimal set of (LP) is called a basic optimal solution, and the Fundamental Theorem states that such a solution exists as long as (LP) is well-posed – i. e. not infeasible or unbounded.

Observe that the definition of a basic feasible solution is independent of the objective function, after all, it is only concerned with feasibility. However, the (MOP) framework blurs the division between the constraints and the objective, and the Fundamental Theorem's extension relies on a broader definition of B and N that includes information from the objective. The idea is to replace the separation of zero and nonzero expressed by B and N with the distinction of whether or not the objective is monotonic. Notice that each term of D has a natural change in monotonicity when its argument is 0. It is precisely this observation that permits the broader perspective.

We say that a function f from $\mathbb{R}$ into $\mathbb{R}$ is *locally strongly monotonic at* x if there is a neighborhood about x over which f is strictly increasing, strictly decreasing or constant. The objective F is not a single variable mapping, making the concept of monotonicity opaque. However, the terms of each component functions are single variable mappings, and we define

$$\begin{aligned} H_j = \{x_j\colon f_{(i,j)} \quad &\text{is not strongly monotonic at } x_j \\ &\text{for some} \quad i \in \{1, 2, \ldots, p\}\}\,. \end{aligned}$$

Each H_j is the collection of values on the jth axis at which the monotonicity of at least one of the objective's components changes. In the case of D, we have for each j that $H_j = \{0\}$. The jth component of x is

cornered if $x_j \in H_j$, and the analogs to B and N are

$$\beta(x) = \{j\colon x_j \quad \text{is not cornered}\} \quad \text{and}$$
$$\nu(x) = \{j\colon x_j \quad \text{is cornered}\}\,.$$

As before, we drop the argument when it is clear. The corresponding definition of a basic feasible solution is a *corner solution*, which is any $x \in \mathbb{R}^n$ satisfying $Ax = b$ so that the columns of A_β are linearly independent.

We are not guaranteed that (MOP) has a corner solution, and to circumvent this undesirable possibility, we assume that H_j is non-empty for each j. Then, from the full rank assumption of A we know that the columns of A can be rearranged so that $A = [A'|A'']$, where A' is invertible. Partitioning x appropriately and cornering each component of x'', we have that $((A')^{-1}(b - Ax'')^{\mathrm{T}}, (x'')^{\mathrm{T}})^{\mathrm{T}}$ is a corner solution.

The monotonicity discussion above permits an extended definition of a basic optimal solution, but it is not enough to extend the Fundamental Theorem. For this we require F to have an additional monotonicity property. We say that a real valued function f is *strongly linearly monotonic over* Ω if it is strongly monotonic, which again means strictly increasing, strictly decreasing or constant, on each line segment within Ω. The monotonicity property we impose on F to extend the Fundamental Theorem is that each component function F_i be strongly linearly monotonic on the closure of

$$\Omega(x) = \{x + \alpha q\colon \alpha \in \mathbb{R}, Aq = 0, \nu(x + \alpha q) = \nu(x), x_{\nu(x)} = (x + \alpha q)_{\nu(x)}\}\,,$$

for each non-cornered optimal x. This assumption guarantees the component functions, and in turn F, are well behaved as we move from a non-corner optimal solution in the affine plane $\{x\colon Ax = b\}$.

Methods

The following result extends the Fundamental Theorem of Linear Programming.

Extension of the Fundamental Theorem

Under the assumptions of the previous section, we have that if $\min\{F(x)\colon Ax = b\}$ has a solution, then it has a corner optimal solution.

The fact that this result includes the original Fundamental Theorem follows directly from (1) and (2). Although the monotonicity properties required by the result are technical, they are not overly restrictive. Indeed, no assumption of continuity or differentiability is needed, and the extension permits functions that are standard counter examples to other analytical results. To highlight this fact, we consider an example with $A = [0, 0, 1]$ and $b = [0]$, which makes the feasible region $\mathbb{R}^2 \times \{0\}$. We let g be the standard Cantor function on $[0, 1]$, which is continuous, piecewise constant, and differentiable almost everywhere with $f'(x) = 0$. This function fails to be differentiable on the Cantor set, denoted by $C = \{\sum_{i=1}^{\infty} \theta_i/3^i\colon \theta_i \in \{0, 2\}\}$. If $f_{(i,j)}$ is the Cantor function, then H_j contains each of these points. We additionally let h be the Dirichlet function on $[1, 2]$ defined by

$$h(x) = \begin{cases} 0, & x \in [1, 2] \cap \mathbb{Q} \\ 1, & x \notin [1, 2] \cap \mathbb{Q} \end{cases}\,.$$

The Dirichlet function is discontinuous over its entire domain. These two functions have played a critical role in the development of analysis since they highlight errors in previous mathematical convention. The point of discussing them here is to show that the extension of the Fundamental Theorem does not suffer from similar hindrances.

We let $F_1 = D$, which guarantees each H_j contains 0. The first two component functions of F_2 are

$$f_{(2,1)}(x_1) = \begin{cases} 1 - x, & x_1 < 1 \\ h(x), & 1 \le x \le 2 \\ \sin(x - (4 + \pi)/2) + 1 & x_1 > 2\,, \end{cases}$$

and

$$f_{(2,2)}(x_2) = \begin{cases} \arctan(1 - x), & x_2 \le 1 \\ 1 - 2g(x - 1), & 1 < x_2 < 4/3 \\ 2g(x - 1) - 1, & 4/3 \le x_2 < 2 \\ \ln(x - 1), & x_2 \ge 2\,. \end{cases}$$

Since every feasible element has $x_3 = 0$ and $0 \in H_3$, this element is always cornered. Notice that $f_{(2,3)}$ only needs to be defined over the singleton $\{0\}$, and we set $f_{(2,3)}(0) = 0$. Each of the functions just defined are non-negative over their domains, and since $F_1(1, 1, 0) = 0$ and $F_2(1, 1, 0) = 0$, the minimum values of F_1 and F_2 over $\{x\colon Ax = b\} = \{x \in \mathbb{R}^3\colon x_3 = 0\}$ are simultaneously zero.

From the definition of F we have that

$$H_1 = \{0\} \cup [1,2] \cup \{2 + k\pi : k = 0, 1, 2, \dots\}$$
$$H_2 = \{0\} \cup (C + 1), \quad \text{and} \quad H_3 = \{0\}.$$

Since x_3 is cornered for every feasible element, we have that $\beta(x) \subseteq \{1, 2\}$, and hence, $A_{\beta(x)}$ is a submatrix of $[0, 0]$. No subcollection of these columns has linearly independent columns, and we conclude that every corner satisfies $\beta(x) = \emptyset$. This means the collection of corners is $H_1 \times H_2 \times H_3$. Since $(1, 1, 0)$ is in this set and $F(1, 1, 0) = (0, 0)^{\mathrm{T}}$, we have that a corner optimal solution exists. To see that there are non-corner optimal solutions, notice that $F(1, 3/2, 0)$ is also $(0, 0)^{\mathrm{T}}$ but that $(1, 3/2, 0)$ is not a corner.

Conclusions

The Fundamental Theorem of Linear Programming was one of the most important results of the 20th century, and we have discussed how the algebraic insights of this result extend beyond the assumption of linearity. The extended presentation leads to simplex type procedures for many problems in which the basic idea is to move from corner to corner until optimality is achieved. Unfortunately, the number of corners can be uncountable as demonstrated by our example, so we lose the finite convergence of linear programming.

References

1. Brown A, Gedlaman A, Holder A, Martinez S (2002) An Extension of the Fundamental Theorem of Linear Programming. Oper Res Lett 30(5):281–288
2. Dantzig G (1963) Linear Programming and Extensions. Princeton University Press, Princeton, NJ
3. Ukovish W, Pastore S, Premoli A (2001) An Algorithm for Linearly Constrained Piecewise Lexicographic Programming Problems. J Optim Theory Appl 111:195–226

Extremum Problems with Probability Functions: Kernel Type Solution Methods

KSM

Riho Lepp
Tallinn Technical University, Tallinn, Estonia

MSC2000: 90C15

Article Outline

Keywords
See also
References

Keywords

Probability function; Kernel estimates; Stochastic approximation

Two types of stochastic programs are widely known: two-stage and chance constrained problems. The last ones were introduced to stochastic programming by A. Charnes and W.W. Cooper in the 1950s [1] and are formally described defining a nonlinear probability function $v(x, t)$ of the form:

$$v(x, t) = \mathsf{P}\{\xi : \ f(x, \xi) \le t\}. \tag{1}$$

Here $f(x, \xi)$ is a real valued function, defined on $\mathbf{R}^r \times \mathbf{R}^\nu$, t is a fixed level of reliability, $\xi = \xi(\omega)$ is a random parameter and P denotes probability. Note that for a fixed x the function $v(x, t)$ as a function of t is the distribution function of the random variable $f(x, s)$.

Various examples of extremum problems with probability function $v(x, t)$ can be found in [3, Chap. 1], where among others also the so-called 'stock exchange' paradox is analyzed. To overcome a paradoxical situation being caused by an unsuccessful choice of the objective expected return, the strategy which maximizes the expected growth of return (Kelly strategy), was applied in [2]. In [3] it was demonstrated that a risky (i. e. probabilistic) strategy is better than the Kelly one.

In the approximate maximization of $v(x, t)$ over the constraint set $X \subset \mathbf{R}^r$ we should apply some (quasi-) gradient type method. This in turn needs the presentation of $v(x, t)$ as an integral, which we can realize via the Heaviside zero-one function $\chi(\cdot)$:

$$\chi(t - f(x, \xi)) = \begin{cases} 1, & \text{if } f(x, \xi) \le t, \\ 0, & \text{if } f(x, \xi) > t. \end{cases}$$

Then

$$v(x, t) = \int_S \chi(t - f(x, \xi))\sigma(d\xi), \tag{2}$$

where $\sigma(\cdot)$ is the distribution function of a random vector ξ and the integral in (2) is understood in the Lebesgue–Stieltjes sense.

Integral representation (2) of the probability function $v(x, t)$ demonstrates us expressively difficulties which arise in approximate maximization of its value: integrand $\chi(\cdot)$ itself is a discontinuous zero-one function and integral (2) over $\chi(\cdot)$ is never convex. Only in some cases, e. g., if function $f(x, \xi)$ is jointly convex and continuous in (x, ξ) and $\sigma(\cdot)$ as a measure is quasiconcave, then function $v(x, t)$ is quasiconcave in x, see [12].

In this survey we at first will solve iteratively, using stochastic analogues of linearization and gradient projection methods, the following probability maximization problem:

$$\max_{x\in X} v(x,t) = \max_{x\in X} \mathsf{P}\{\xi:\ f(x,\xi) \le t\}, \tag{3}$$

where the constraint set X is assumed to be simple, i. e. on X we can effectively solve auxiliary problems of maximization of linear or quadratic functions. At second, we will exploit the introduced technique for minimization of a smooth function over probabilistic equality-inequality type constraints, using a stochastic analogue of the modified Lagrange method.

Gradient type methods require differentiability of a cost function. A lot of papers have been devoted to differentiability conditions of $v(x, t)$ in x, starting from [13] where $v'_x(x, t)$ was presented via surface integral. The gradient of $v(x, t)$ via volume integral was presented in [16]; see also the survey paper [4]. All these formulas are quite uncomfortable to handle, especially for numerical methods. In the following we will assume differentiability of $v(x, t)$ in x and in (x, t), i. e. there exist $v'_x(x, t)$ and $v''_{xt}(x, t)$.

Define solution sets X^* for the problem (3) as follows:

$$X^* = \left\{x^*:\ (v'_x(x^*,t), x - x^*) \le 0, \quad \forall x \in X\right\}, \tag{4}$$

or

$$X^* = \left\{x^*:\ x^* = \pi[x^* + \rho v'_x(x,t)], \quad \forall \rho > 0\right\}, \tag{5}$$

where $\pi[y]$ means the projection of a vector y to the set X. Then we can interpret linearization and gradient projection methods as iteration ways for testing conditions (4) and (5), respectively.

Following [17, Chap. IV], method for solution of a problem is said to be convergent, if limit points of the sequence $\{x_n\}$, generated by the algorithm, belong to the solution set X^*.

Denote n independent realizations $\xi_1, \ldots, \xi_n$ of a random vector ξ by ξ^n, i. e., $\xi^n = (\xi_1, \ldots, \xi_n)$. Then, following [14] and [10], the smoothed approximation of $v(x, t)$ looks as follows:

$$v_n(x,t,\xi^n) = v_n(x,t,\xi_1 \ldots, \xi_n) = \frac{1}{nh_n}\sum_{i=1}^{n}\int_{-\infty}^{t} K\left(\frac{\tau - f(x,\xi_i)}{h_n}\right)d\tau, \tag{6}$$

where the sequence $\{h_n\}$ is connected with the sequence $\mathbf{N} = \{1, 2, \ldots\}$ as

$$\lim h_n = 0, \quad \lim nh_n = \infty, \quad n \in \mathbb{N}, \tag{7}$$

and the continuous kernel function $K(y)$ satisfies conditions [14]:

$$\int_{-\infty}^{\infty} K(y)\,dy = 1, \quad \sup_{-\infty<y<\infty} |K(y)| < \infty, \tag{8}$$

$$\int_{-\infty}^{\infty} yK(y)\,dy = 0, \quad \int_{-\infty}^{\infty} |yK(y)|\,dy < \infty. \tag{9}$$

Gradient of the smoothed approximate probability function $v_n(x, t, \xi^n)$ from (6) looks now as follows:

$$v'_{nx}(x,t,\xi^n) = -\frac{1}{nh_n}\sum_{i=1}^{n} f'_x(x,\xi_i)K\left(\frac{t - f(x,\xi_i)}{h_n}\right). \tag{10}$$

Even estimates (6) and (10) are biased, i. e.,

$$\mathsf{E}v'_{nx}(x,t,\xi^n) \ne v'_x(x,t),$$

we still have

$$\mathsf{E}v'_{nx}(x,t,\xi^n) = v'_x(x,t) - h_n\int_{-\infty}^{\infty} yK(y)v'_{x,t}(x, t - \theta h_n y)\,dy,$$

where $0 \le \theta \le 1$, see [15], and consequently,

$$\lim_{n\to\infty}\sup_{x\in X}\left|\mathsf{E}v'_{nx}(x,t,\xi^n) - v'_x(x,t)\right| = 0.$$

For approximate solution of (3) consider the stochastic analogue of the linearization method:

$$x_{n+1} = x_n + \gamma_n(\overline{x}_n - x_n), \tag{11}$$

where $\overline{x}_n$ is a solution of the linear problem:

$$\max_{x\in X}(v'_{nx}(x_n, t, \xi^n), x) = v'_{nx}(x_n, t, \xi^n, \overline{x}_n)$$

and $x_0 \in X$.

Explain the stochastic nature of the sequence $\{x_n\}$, generated by the algorithm (11). For each n the random vector x_n is defined on the sigma-algebra F_{n-1}, generated by random vectors $\xi_1, \ldots, \xi_{n-1}$. Union of the sequence of sub-sigma-algebras $\cup_{i=1}^{\infty} F_i$ is equal to the sigma-algebra F of the initial probability space $(\Omega, F, \mathbf{P})$, where the random vector ξ was defined. Note that in each iteration step we should generate new (independent) realizations of the random vector ξ.

Assume that function $f(x, \xi)$ is differentiable in x and that for all $t \in \mathbf{R}^1$ and all $x \in X$ its gradient is bounded with a σ-integrable function $K(\xi)$:

$$|f_x(x, \xi)| \le K(\xi), \quad \int_{\mathbb{R}^v} K(\xi)\sigma(\,d\xi) < \infty. \tag{12}$$

Let the sequence $\{\gamma_n\}$ of steplength satisfy conditions:

$$0 \le \gamma_n \le 1, \quad \gamma_n \to 0, \quad \sum_{n=1}^{\infty} \gamma_n = \infty. \tag{13}$$

Then the following convergence theorem holds [5]

Theorem 1 *Let differentiable in x function $f(x, \xi)$ satisfy conditions (12), smoothing continuous kernel $K(y)$ conditions (8), (9), sequence $\{\gamma_n\}$ of steplength conditions (13), and let the solution set X^* be finite. Then all limit points of the sequence $\{x_n\}$, generated by the algorithm (11), belong almost surely to the solution set X^*.*

Remark 2 Proof of the theorem relies on the stochastic analogue of [17, Thm. A], see [9, Chap. II, Thm. 8], and was verified in [5].

Remark 3 Statements of the theorem are valid also for the stochastic analogue of the gradient projection method, see [5]:

$$x_{n+1} = \pi[x_n + \gamma_n v'_{nx}(x_n, t, \xi_n)], \quad x_0 \in X. \tag{14}$$

As it was described earlier, algorithms (11) and (14) need in nth iteration step n independent realizations of the random vector ξ. In [11] it was verified that in asymptotic sense statistical estimation type methods, as algorithms (11) and (14) are, have no advantages compared with methods of random search, but need more calculating efforts.

As an example of the last statement consider the free maximum problem:

$$\max_{x\in\mathbb{R}^r} = \max_{x\in\mathbb{R}^r} \mathsf{P}\{\xi\colon\ f(x, \xi) \le t\}. \tag{15}$$

Let ξ_n be the nth realization of the random variable ξ. Consider the algorithm:

$$x_{n+1} = x_n - \frac{\gamma_n}{h_n} f'_x(x_n, \xi_n) K\left(\frac{t - f(x_n, \xi_n)}{h_n}\right). \tag{16}$$

Assume, in addition to assumptions (7)–(9) and (12), (13) to $\{h_n\}$, $\{\gamma_n\}$, $K(y)$ and $f(x, \xi)$, that

$$\sum_{n=1}^{\infty} \gamma_n^2 < \infty; \quad \sum_{n=1}^{\infty} \gamma_n h_n < \infty; \quad \sum_{n=1}^{\infty} \frac{\gamma_n^2}{h_n^2} < \infty. \tag{17}$$

Then, if $\int_{\mathbb{R}^v} |f'_x(x, \xi)|\sigma(\mathrm{d}\xi)$ is bounded for bounded x, the limit points of the sequence $\{x_n\}$ belong almost surely to the set X^* of stationary points,

$$X^* = \left\{x^*\colon\ v'_x(x^*, t) = 0\right\},$$

see [7].

Remark 4 Even algorithms (11) and (14) take more calculating efforts compared with random search method (16), the last one is very unstable, and converges only 'in probability' sense.

Consider the following nonlinear programming problem with a smooth cost function $f(x)$ and with probabilistic constraints of inequality type with a fixed level of reliability α, $0 < \alpha < 1$, i. e.,

$$\min_{x\in\mathbb{R}^r} \{f(x)\colon\ v(x, t) \ge \alpha\} \tag{18}$$

(for sake of simplicity consider only the case with one inequality constraint).

Define the solution set X^* for the problem (18) as follows [8]:

$$X^* = \{x^*\colon\ F \cap G\}, \tag{19}$$

where

$$F = \left\{x^*\colon\ \left|f'_x(x^*) + v'_x(x^*, t)\lambda^*\right|^2 = 0\right\}, \tag{20}$$

with

$$\lambda^* = \arg\min_{\lambda \geq 0} \left| f'_x(x^*) + v'_x(x^*, t)\lambda \right|^2, \tag{21}$$

and

$$G = \{x^* : \ v(x^*, t) \geq \alpha\}, \tag{22}$$

where λ^* is the optimal Lagrange multiplier of the Lagrangian.

Replacing $v(x, t)$ and $v'_x(x, t)$ with their estimates (6) and (10), we should regularize the estimated analogue of (21) since the approximated subproblem (21) could be ill-posed.

Denote by

$$w_n(x, t, \xi_n) = \min\{0, v_n(x, t, \xi^n) - \alpha\}.$$

Then the stochastic analogue of modified Lagrange method looks as follows:

$$\begin{aligned} x_{n+1} &= x_n \\ &- \gamma_n [f'_x(x_n) + v'_{nx}(x_n, t, \xi^n)\lambda_n(\xi^n) \\ &+ M v'_{nx}(x_n, t, \xi^n) w_n(x_n, t, \xi_n)], \end{aligned} \tag{23}$$

where $\lambda_n(\xi_n)$ is a solution of the regularized auxiliary subproblem of quadratic programming

$$\min_{\lambda \geq 0} \left[\left| f'_x(x_n) + v'_{nx}(x_n, t, \xi^n)\lambda \right|^2 + \alpha_n |\lambda|^2 \right]$$

with $\alpha_n > 0$, $\alpha_n \to 0$, $n \to \infty$ and $M > 0$. The following convergence theorem is valid, see [6]:

Theorem 5 *Let conditions of the previous theorem be satisfied, let the cost function $f(x)$ be continuously differentiable and let*

$$\sum_{n=1}^{\infty} \alpha_n \gamma_n < \infty.$$

Then limit points of the sequence, generated by the algorithm (23), belong almost surely to the solution set X^*, defined by (19).

See also

- Approximation of Extremum Problems with Probability Functionals
- Approximation of Multivariate Probability Integrals
- Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- General Moment Optimization Problems
- Logconcave Measures, Logconvexity
- Logconcavity of Discrete Distributions
- L-shaped Method for Two-stage Stochastic Programs with Recourse
- Multistage Stochastic Programming: Barycentric Approximation
- Preprocessing in Stochastic Programming
- Probabilistic Constrained Linear Programming: Duality Theory
- Probabilistic Constrained Problems: Convexity Theory
- Simple Recourse Problem: Dual Method
- Simple Recourse Problem: Primal Method
- Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- Static Stochastic Programming Models
- Static Stochastic Programming Models: Conditional Expectations
- Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- Stochastic Integer Programs
- Stochastic Linear Programming: Decomposition and Cutting Planes
- Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- Stochastic Network Problems: Massively Parallel Solution
- Stochastic Programming: Minimax Approach
- Stochastic Programming Models: Random Objective
- Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- Stochastic Programming with Simple Integer Recourse
- Stochastic Programs with Recourse: Upper Bounds
- Stochastic Quasigradient Methods in Minimax Problems
- Stochastic Vehicle Routing Problems
- Two-stage Stochastic Programming: Quasigradient Method
- Two-Stage Stochastic Programs with Recourse

References

1. Charnes A, Cooper WW (1959) Chance-constrained programming. Managem Sci 5:73–79
2. Kelly J (1956) A new interpretation of information rate. Bell System Techn J 35:917–926

3. Kibzun AI, Kan YS (1995) Stochastic programming problems with probability and quantile functions. Wiley, New York
4. Kibzun A, Uryasev S (1998) Differentiability of probability functions. Stochastic Anal Appl 16:1101–1128
5. Lepp R (1979) Maximization of a probability function over simple sets (in Russian). Proc Acad Sci Estonian SSR Phys Math 28:303–308
6. Lepp R (1980) Minimization of a smooth function over probabilistic constraints (in Russian). Proc Acad Sci Estonian SSR Phys Math 29:140–144
7. Lepp R (1983) Stochastic approximation type algorithm for the maximization of the probability function. Proc Acad Sci Estonian SSR Phys Math 32:150–156
8. Miele A, Cragg EG, Iver RR, Levy AV (1971) Use of the augmented penalty functions in mathematical programming problems. Part I. J Optim Th Appl 8:115–130
9. Nurminskii EA (1979) Numerical methods for solution of deterministic and stochastic Minimax Problems. Nauk. Dumka, Kiev
10. Parzen E (1962) On the estimation of a probability density and the mode. Ann Math Statist 33:1065–1076
11. Polyak BT, Tsypkin YZ (1979) Adaptive algorithms of estimation (convergence, optimality, stability) (in Russian). Avtomatika i Telemekhanika (Automatics and Remote Control):74–84
12. Prékopa A (1980) Logarithmic concave measures and related topics. In: Dempster MAH (ed) Stochastic Programming. Acad. Press, New York
13. Raik E (1975) Differentiability in the parameter of the probability function and optimization of the probability function via the stochastic pseudogradient method. Proc Acad Sci Estonian SSR Phys Math 24:3–6 (In Russian.)
14. Rosenblatt M (1957) Remarks on some nonparametric estimates of a density function. Ann Math Statist 27:832–837
15. Tamm E (1979) On the minimization of the probability function (in Russian). Proc Acad Sci Estonian SSR Phys Math 28:17–24
16. Uryasev S (1989) A differentiation formula for integrals over sets given by inclusion. Numer Funct Anal Optim 10:827–841
17. Zangwill WI (1969) Nonlinear programming. A unified approach. Prentice-Hall, Englewood Cliffs, NJ

F

Facilities Layout Problems

FLP

L. R. Foulds[1], Horst W. Hamacher[2]
[1] Department Management Systems, University Waikato, Waikato, New Zealand
[2] Fachbereich Math., University Kaiserslautern, Kaiserslautern, Germany

MSC2000: 90B80

Article Outline

Keywords
See also
References

Keywords

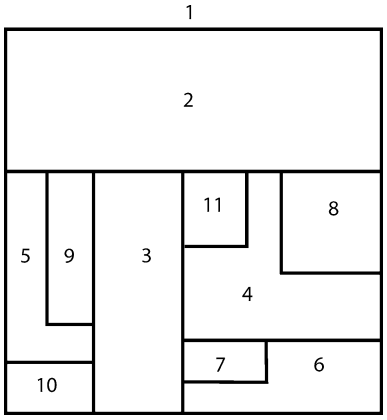
Multifacilities location; REL chart scores; NP-hard; Heuristics; Improved procedure; Construction procedures; Quadratic assignment problem; Location; Graph theory; Graph; Planar subgraph; Adjacency graph; Integer programming; Lagrangian relaxation; Bounds; Branch and bound; Simulated annealing; Genetic algorithms; Multicriteria objective function; Transportation cost; Decision support system; Layout manager

Facilities layout (FL) is concerned with the placement, relative to one another, of the facilities of some physical system. This area differs from planar multifacilities location (MFL; cf. also ► Multifacility and Restricted Location Problems) in that in FL the facilities are all assumed to have a significant physical area and are to be placed in a finite total area which represents their physical system. In MFL the facilities are assumed to be dimensionless points.

The aim of FL is to produce a scale plan (in some scenarios called a *block plan*) of the physical system to be designed. The plan depicts the facilities of the system (each one having its given area and shape) laid out relative to each other. An example of a simple block plan is shown in Fig. 1.

The identification of effective plans depends upon interfacility relationships, which may be quantitative (e. g. transportation costs) or qualitative (e. g. utility scores, called *REL chart scores*, based on facility adjacency). Each FL problem involves optimizing one or more objective functions based on the given interfacility relationship.

FL is an important application area of optimization. This is partly because increased global competition in manufacturing has spurred renewed efforts to



Facilities Layout Problems, Figure 1
A block plan with 11 facilities, including the exterior region, indicated as facility 1

reduce production costs. Efficient physical layout design of manufacturing plants is critical in the quest to achieve and maintain competitive productivity. Indeed, up to 70% of the operating costs of a manufacturing system are related to materials handling and layout. This is because improved layout design often brings about reductions in materials handling, transportation, congestion, and work-in-process.

There are applications of FL techniques in areas other than manufacturing plant design. Examples include the design of office blocks and other commercial buildings, hospitals and other public services, and university campuses, government agencies, and sports complexes. As will become evident in the following discussion, most FL models are *NP*-hard in the strong sense (cf. also ► Complexity Theory; ► Complexity Classes in Optimization). This has reinforced the search for effective heuristics for them.

One of the earliest and best-known FL heuristics is termed CRAFT (coordinate relative allocation of facilities technique) [1]. It requires an initial block plan as input, which it attempts to improve by exchanging the positions of two or three facilities at a time. In contrast to this improved procedure, many other early FL heuristics are construction procedures which build up the final block plan iteratively, by placing facilities sequentially. The serial decision process requires, at each step:

i) a selection of which facility is to be placed next in the block plan being constructed, and
ii) a decision as to where this facility is going to be placed.

Early construction procedures include: CORELLAP [23] and ALDEP [32].

One of the major FL optimization models is based on the quadratic assignment problem (QAP; cf. also ► Quadratic Assignment Problem). For overviews on this subject see [4,5,29]. Formulations of various FL problems based on the QAP involve minimizing the total transportation cost between all pairs of facilities. This total cost comprises a sum of components calculated according to the distance and the amount of work flow between each pair of facilities. The constraints of the QAP model are based on the assumption that the block plan is tessellated into a grid of unit squares (called *locations*) and that no two facilities are to be assigned the same location. Many of these models assume that all the facilities are of equal area. However, when facilities have unequal areas or irregular shapes, additional constraints must be added. The facilities are partitioned into a number of subfacilities of unit area. The problem then is to locate the subfacilities so that all the subfacilities of each facility are assigned adjacent locations in an appropriate configuration. As the QAP is *NP*-hard, most FL applications of it are concerned with heuristics. A QAP model of a common FL problem is:

$$\begin{cases} \min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{r=1}^{n} a_{ijkr}x_{ij}x_{kr} \\ \text{s.t.} & \displaystyle\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,\ldots,n, \\ & \displaystyle\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,\ldots,n, \\ & x_{ij} = 0 \text{ or } 1, \quad i,j = 1,\ldots,n, \end{cases}$$

where

n = the number of subfacilities,
c_{ij} = the cost per unit time period of assigning subfacility i to location j. (These costs are usually one-time relocation costs which are converted to an annual equivalent.)
d_{jr} = the cost per movement or interaction over the distance from location i to location r,
f_{ik} = the number of moves per time period in the workflow from subfacility i to subfacility k,
S_i = the set of locations to which subfacility i may be feasibly assigned,

$$a_{ijkr} = \begin{cases} f_{ik}d_{jr} & \text{if } i \neq k \text{ or } j \neq r, \\ c_{ij} & \text{if } i = k \text{ and } j = r, \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if subfacility } i \\ & \quad \text{is assigned to location } j, \\ 0 & \text{otherwise.} \end{cases}$$

If there are more locations than subfacilities, a number of dummy facilities can be introduced with zero c_{ij} and f_{ik} values. The f_{ik} values are set to relatively high levels if subfacilities i and k belong to the same facility, thereby ensuring their adjacency. The c_{ij} values are set to relatively high values when $j \notin S_i$.

A second major FL optimization model is based on graph theory (GT) and involves maximizing the sum

of the REL chart scores corresponding to the pairs of facilities that are adjacent in the block plan. The formulations can accommodate specifications that the region exterior to the block plan is one of the facilities, and that the facilities are of unequal areas and various shapes. GT models represent facilities and the possible adjacency of pairs of facilities in the block plan by the vertices and edges of a graph, respectively. The REL chart scores are used to weight the edges of the graph. The objective is to identify the planar subgraph (termed an *adjacency graph*) of this graph with the largest total weight in terms of its REL chart scores. The optimal adjacency graph specifies which pairs of facilities are to be placed adjacent to each other in the block plan. As this model was shown to be *NP*-hard in [13], most research concentrates on heuristics. However, some GT algorithms for FL problems guaranteeing optimality, do exist. The algorithm in [12] involves a series of tests for determining whether a proposed adjacency graph being constructed is planar or not. In [6] an integer programming formulation based on the GT approach is discussed. It employs a Lagrangian relaxation procedure (cf. also ► Integer Programming: Lagrangian Relaxation) for the derivation of bounds to be used in a branch and bound algorithm (cf. also ► Integer Programming: Branch and Bound Methods). Approaches to enforce connectivity of subgraphs corresponding to facilities are taken from k-cardinality tree models ([14] and [7]) which can also incorporate forbidden areas [10,11].

Early GT heuristics first identify the adjacency graph and then attempt to construct a block plan corresponding to the information provided by the graph. Examples include the heuristics of [3,9,24] and [27]. The comparisons in [28] show that the results of [27] are invariably so close to optimality that the quest for heuristics which find good quality adjacency graphs can now be considered essentially solved. More recent GT heuristics build up the adjacency graph and its corresponding block plan simultaneously, such as the heuristics of [37].

It has been observed that many of the previously mentioned techniques are not computationally feasible for some of the large scale numerical instances of FL problems encountered in industry and often identify local optima which are clearly far from globally optimal. This has given rise to many investigations into whether the more recently developed random search procedures (such as simulated annealing (SA; cf. also ► Simulated Annealing Methods in Protein Folding) and genetic algorithms (GA; cf. also ► Genetic Algorithms)) could be used to devise useful FL heuristics. There is a fundamental difference between SA and GA. That is, GA must, of necessity, deal with a set of possible solutions to the problem in hand, while SA considers only one possible solution at a time. Because GA explores the set of all feasible solutions by combining the characteristics of various single feasible solutions, it sometimes covers a larger portion of the solution space than SA, within the same computational time. Thus, it appears to be the more successful of the two for FL problems.

SA can be applied to FL problems in a variety of ways. There exist SA improvement heuristics for FL problems with

i) multiple floors, (based on the improvement approach) [26],
ii) multiple objectives based on both transportation costs and REL chart scores [33].

For further information see [16] and [22]. However, it appears that the logarithmic cooling schedule of SA causes its FL heuristics to perform relatively slowly. For this reason it seems that GA heuristics are more effective for FL problems. For instance the GA approach to solve the QAP, devised in [35], can be applied to QAP models of FL problems, such as the one given earlier. However, this GA heuristic has only a single solution giving rise to a mutant, which means that parallelism is lost to a certain extent.

To overcome this deficiency, it is possible to design more effective GA heuristics for FL problems by adopting a small mutation rate and a large crossover rate. A heuristic with efficient crossover operators with low level mutation has been devised in [34]. Further heuristic attempts to tackle the QAP include tabu search (see e. g. [2]) or the reverse elimination method [36].

The approaches to FL described so far have been classical in the sense that they have nearly all embraced single objective functions. In contrast, there have been developments in FL models with multiple criteria. Examples include: a multifactor plant layout methodology devised in [15], a layout planning system with multiple criteria and a variable domain representation in [18], an expert system using priorities for solving multiple criteria facilities layout problems in [25], and a multi-

attribute decision theoretic approach for layout design in [31].

There are numerous computer programs in existence which implement FL heuristics. Three early ones from the 1960s: CRAFT, CORELAP, and ALDEP have already been discussed. In the 1970s two improvement-style heuristics, both based on CRAFT, appeared to be among the best of those proposed then. FRAT (facilities relative allocation technique) [21] assumes that all the facilities have equal areas. TSP (terminal sampling procedure) [17] carries out the interchange of the placement, in the block plan, of pairs of facilities on a selective basis. The program has the ability to use improved block plans as input and to fix the placement of certain facilities. Three of the large number of FL programs written in the 1980s will be mentioned. SPACE-CRAFT [20] is an extension of CRAFT to multifloor FL problems. See [17] for a perturbation scheme, and [18] for a new FL system which accommodates a variety of types of spaces, including solid, circulation, and empty. A multicriteria objective function involves transportation cost, REL chart scores, the percentage of unused area, and block plan structure.

The 1990s saw a different type of FL program emerging: the *decision support system* (DSS). One such example, called *layout manager* [8] is a user-friendly menu-driven DSS which provides for the choice between a number of optimality criteria including, among others, transportation cost and REL chart scores. The system is written in Pascal, within the Microsoft Windows environment.

See also

- ▶ Combinatorial Optimization Algorithms in Resource Allocation Problems
- ▶ Competitive Facility Location
- ▶ Facility Location with Externalities
- ▶ Facility Location Problems with Spatial Interaction
- ▶ Facility Location with Staircase Costs
- ▶ Global Optimization in Weber's Problem with Attraction and Repulsion
- ▶ MINLP: Application in Facility Location-allocation
- ▶ Multifacility and Restricted Location Problems
- ▶ Network Location: Covering Problems
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Production-distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Stochastic Transportation and Location Problems
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Armour G, Buffa E (1963) A heuristic algorithm and simulation approach to relative location of facilities. Managem Sci A:294–309
2. Battiti R, Tecciolli G (1992) Parallel biased search for combinatorial optimization: Genetic algorithms and tabu search. Microprocessors and Microsystems 16:351–367
3. Boswell SG (1992) A new greedy heuristic for facilities planning. Internat J Production Res 30:1957–1968
4. Burkard RE (1990) Locations with spatial interactions: The quadratic assignment problem. In: Francis RL, Mirchandani P (eds) Discrete Location Theory. Wiley, New York, pp 387–437
5. Cela E (1998) The quadratic assignment problem: Theory and algorithms. Kluwer, Dordrecht
6. Christofides N, Galliani G, Stefanini L. An algorithm for the maximal planar graph problem based on Lagrangean relaxation. Math Program
7. Fishetti M, Hamacher HW, Jörnsten K, Maffioli F (1994) Weighted k-cardinality trees: Complexity and polyhedral structure. NETWORKS 24:11–21
8. Foulds LR (1997) Layout manager: A microcomputer-based decision support system for facilities layout. Decision Support Systems 20:199–213
9. Foulds LR, Gibbons PB, Giffin JW (1985) Facilities layout adjacency determination: An experimental comparison of three graph theoretic heuristics. Oper Res 33:1091–1106
10. Foulds LR, Hamacher HW (1992) A new integer programming approach to (resticted) facilities layout problems allowing flexible facility shapes. Res Report Dept Management Systems Univ Waikato
11. Foulds LR, Hamacher HW, Wilson JM (1998) Integer programming approaches to facilities layout models with forbidden areas. Ann Oper Res 81:405–417
12. Foulds LR, Robinson DF (1976) A strategy for solving the plant layout problem. Oper Res Quart 27:845–855
13. Giffin JW (1985) Graph theoretic techniques for facilities layout. PhD Thesis, Dept Economics, Univ Canterbury, Christchurch, New Zealand

14. Hamacher HW, Joernsten K, Maffioli F (1993) Weighted k-cardinality trees. Techn Report, Dipt Elettronica, Politecnico Milano, 91.023
15. Harmonosku CM, Tothero GK (1992) A multi-factor plant layout methodology. Internat J Production Res 30:1773–1790
16. Heregu SS, Alfa AS (1992) Experimental analysis of simulated annealing-based algorithms for the layout problem. Europ J Oper Res 57:190–202
17. Hitchings GG, Cottam M (1976) An efficient heuristic procedure for solving the layout design problem. OMEGA Internat J Management Sci 4:205–214
18. Jacobs FR (1987) A layout planning system with multiple criteria and a variable domain representation. Managem Sci 33:1020–1034
19. Jajodia S, Minis I, Harhalakis G, Proth JM (1992) CLASS: Computerised layout solutions using simulated annealing. Internat J Production Res 30:95–108
20. Johnson RV (1982) SPACECRAFT for multi-floor layout planning. Managem Sci 28:407–417
21. Khalil TM (1973) Facilities relative allocation technique (FRAT). Internat J Production Res 11:183–194
22. Kouvelis P, Chiang WC, Fitzsimmons J (1992) Simulated annealing for machine layout problems in the presence of zoning constraints. Europ J Oper Res 57:203–223
23. Lee R, Moore JM (1967) CORELAP – Computerized relationship layout planning. J Industr Eng 18:195–200
24. Leung J (1992) A new graph theoretic heuristic for facility layout. Managem Sci 38:594–605
25. Malakooti B, Tsirisjo A (1989) An expert system using priorities for solving multiple criteria facility layout problems. Internat J Production Res 27:793–808
26. Meller RD, Bozer YA (1986) A new simulated annealing algorithm for the facility layout problem. Internat J Production Res 34(6):1675–1692
27. Merker J, Waescher G (1987) Two new heuristic algorithms for the maximal planar layout problem. OR Spektrum 19:131–137
28. Merker J, Waescher G (1997) A comparative evaluation of heuristics for the adjacency problem in facility layout planning. Internat J Production Res 33:447–466
29. Pardalos PM, Wolkowicz H (1994) Quadratic assignment and related problems. DIMACS, vol 16. Amer Math Soc, Providence
30. Picone CJ, Wilhelm WE (1984) A perturbation scheme to improve Hillier's solution to the facilities layout problem. Managem Sci 30:1238–1249
31. Sarin SC, Loharjun P, Malmborg CJ, Krishnakumar A (1992) A multiattribute decision-theoretic approach for the layout design problem. Europ J Oper Res 57:231–242
32. Seehof JF, Evans WO (1967) Automated layout design problems. J Industr Eng 18:690–695
33. Suresh G, Sahu S (1993) Multi-objective facility layout using simulated annealing. Internat J Production Economics 32:239–254
34. Suresh G, Vinod VV, Sahu S (1995) A genetic algorithm for facility layout. Internat J Production Res 33:3411–3423
35. Tate DM, Smith AE (1995) A genetic approach to the quadratic assignment problem. Comput Oper Res 32:73–83
36. Voss S (1995) Solving quadratic assignment problems using the reverse elimination method. In: Nash SG, Sofer A (eds) The Impact of Emerging Technologies on Computer Sci and Operations Res. Kluwer, Dordrecht
37. Watson K, Giffin JW, Foulds LR (1995) Orthogonal layouts using the deltahedron heuristic. J Australasian Combin Soc 12:127–144

Facility Location with Externalities

MARGARET BRANDEAU
Stanford University, Stanford, USA

MSC2000: 90B80, 90B85, 91Bxx, 90Cxx, 91Axx

Article Outline

Keywords
Location of Mobile Servers
Location of Fixed Service Facilities
System-Optimizing Environment
User-Optimizing Environment
Resource Allocation with Externalities
See also
References

Keywords

Discrete location and assignment; Continuous location; Mathematical economics; Mathematical programming; Game theory

A typical assumption in *facility location* models is that the cost customers face in patronizing facilities is independent of the actions of other customers (with the possible exception of capacity restrictions). For example, many classical facility location models assume that customers patronize the facility (or are served by the facility) that minimizes the cost of travel between the facility and the customer (e. g., see, [12,13]). Other facility location models incorporate marketing considerations, and assume that customers patronize the facility that is

'most attractive' to them, where attractiveness depends not only on travel cost, but also on attributes of each facility such as size, goods offered, and number of servers (e. g., see [15]).

However, in many situations, the cost customers face in patronizing a facility is a function of the actions of other customers. For example, waiting time for service may be longer in a store that is patronized by many customers than in a store with fewer customers. An ambulance that serves a large, heavily-populated area is likely to incur longer delays in providing service than an ambulance serving a smaller, less-populated area. These are examples of negative externalities associated with the market share of the facility. Conversely, in some cases the *externalities* could be positive: for example, a crowded nightclub is likely to be more popular than one that attracts fewer patrons.

If facilities provide essential services (e. g., gasoline, drivers' licenses), customer demand may be constant, regardless of the costs customers face in obtaining services. However, for facilities that provide nonessential services (e. g., fast-food restaurants, retail stores), customer demand might be a function of the total cost of receiving service.

This chapter discusses models for the location of facilities that incorporate not only travel cost but also negative externalities associated with the market share of the facility. Various problem formulations are discussed, and selected references are provided. A more comprehensive discussion is given in [10]. The case of positive externalities is not discussed because, for such problems, degenerate solutions tend to occur (e. g., the optimal solution may be to locate all facilities at the same point, with any point in the region being optimal).

One can consider two different situations regarding the allocation of customer demands to facilities. In a *user-optimizing environment*, customers patronize the facility that minimizes their total cost, in this case travel cost plus externality cost. Such a situation occurs, for example, in customers' selection of grocery stores and bank branches. In a *system-optimizing environment*, customers are assigned to facilities by a central agent. An example is the assignment of voters to polling places.

In the system-optimizing environment, allocation of customer demands can be considered as part of the location optimization problem, similar to many models of facility location that do not incorporate externalities. In the user-optimizing environment, however, models of facility location have at their core a customer-choice *equilibrium* problem: equilibrium occurs when each customer frequents the facility that minimizes his total travel cost plus externality cost. For purely negative externalities, the equilibrium utilization of facilities (total demand satisfied by each facility) is unique, although the equilibrium user-choice pattern (allocation of individual customer demands to facilities) may not be unique ([8,18]). This result holds whether demands are inelastic or elastic with respect to total customer cost. Determination of the user-choice equilibrium can be written as a nonlinear complementarity problem (analogous to [1]), and also as a network flow problem [21] which can be solved using network optimization techniques (e. g., [20]).

This article discusses models for the location of facilities in both types of customer choice environments. A distinction is made between facilities with mobile servers (e. g., ambulances) that travel to fixed customers and return to their home location between calls and facilities that house fixed servers (e. g., postal clerks).

Location of Mobile Servers

Some of the first location models to incorporate externalities were developed in the context of emergency service vehicle location. In such models, the servers (the emergency service vehicles) travel to customers, and the externality cost is the servers' queueing delay. A system-optimizing environment is assumed: customers are assigned to service regions of the servers. Models for determining the home location of such mobile servers have considered a variety of location objectives, including minimization of mean response time to customers (travel time plus queue delay), minimization of the maximum response time to any customer, equalization of server workloads, and other objectives. Examples of such models can be found in [3,4,5,7,11], and [19].

Location of Fixed Service Facilities

Most other facility location models that incorporate externality costs have assumed fixed service facilities.

System-Optimizing Environment

In the system-optimizing environment, since customers are assigned centrally to facilities, it is natural to think only of noncompeting facilities. For the case of fixed customer demands, a natural objective in locating facilities (and allocating customers to facilities) is to minimize total customer cost. This problem is a generalized p-median problem. Such a model might be appropriate for the location of certain public facilities such as voters' polling places. O. Berman and R.C. Larson [2] presented a p-median problem that includes queueing-like congestion of the facilities. In the system-optimizing environment with elastic demands, a natural objective is to locate facilities to maximize the total demand served by facilities (i. e., maximum facility utilization). Such a model might be relevant for the location of fast-food franchises or clinics for preventive childcare (e. g., inoculations). This facility location problem is a generalized p-median problem with an embedded demand equilibrium [18]. For the case of discrete customer demands on a network, S. Kumar [18] proved a nodal optimality theorem and showed that the problem can be formulated as a nonlinear integer convex programming problem and solved using branch and bound (see also [10]).

User-Optimizing Environment

In the user-optimizing environment with fixed service facilities, one can distinguish between noncompeting and competing facilities. For the case of noncompeting facilities in the user-optimizing environment with inelastic demand, a natural location objective is to minimize total customer cost. This framework might be appropriate for the location of public facilities such as Social Security Offices. Assuming discrete customer demands, the problem can be written as a mixed integer bilevel program [10] (given a set of fixed facility locations, one can then determine the user-choice equilibrium utilization of facilities). M.L. Brandeau and S.S. Chiu [8] considered the case of two such facilities on a tree network with nodal demands. They characterized the optimal facility locations, and presented an algorithm for finding those locations.

A typical location objective for the case of competing facilities (whether or not externalities are considered) is maximization of market share. When externalities are not considered, problems of competitive facility location involve a locational equilibrium; when negative externality costs and user-optimizing customer choice are considered, such problems also involve a customer-choice equilibrium. E. Kohlberg [17] considered the location of competing identical facilities on a line with uniformly distributed, inelastic demands where customers select a facility based on the sum of travel time plus waiting time for service. For the case of two facilities, the optimal locations occur at the midpoint of the line, and for the case of more than two facilities, Kohlberg [17] showed that a locational equilibrium does not occur. R.M. Braid [6] analyzed the locational equilibrium for two congested public facilities located by competing governmental jurisdictions in an inelastic-demand environment. Brandeau and Chiu [9] analyzed the case of two competing facilities on a tree network with inelastic demands and a general negative externality function. Such a model might be appropriate for the location of similar competing grocery stores. They assumed a Stackelberg game (with a leader and a follower). They characterized the optimal locations of the leader and the follower, and presented an algorithm for finding those locations.

Kumar [18] considered the location decision of a profit-maximizing firm that locates one facility in a region where a number of competitors are already located and in which customer demand is elastic. An example application is the location of competing retail outlets. The problem is a bilevel programming problem which can be heuristically solved using a gradient projection ascent approach (e. g., [14]).

Resource Allocation with Externalities

If facilities are already located, changing facility locations may be expensive. An alternative is to allocate resources to change the characteristics of the facility (e. g., through training or technological improvements). The question is how to balance the cost of change with the associated benefits (e. g., increased market share, lowered total customer cost). *Resource allocation* problems of this type are discussed in [10] and [16].

See also

► Combinatorial Optimization Algorithms in Resource Allocation Problems

- ▶ Competitive Facility Location
- ▶ Facility Location Problems with Spatial Interaction
- ▶ Facility Location with Staircase Costs
- ▶ Global Optimization in Weber's Problem with Attraction and Repulsion
- ▶ MINLP: Application in Facility Location-allocation
- ▶ Multifacility and Restricted Location Problems
- ▶ Network Location: Covering Problems
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Production-distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Stochastic Transportation and Location Problems
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Aashtiani HZ, Magnanti TL (1981) Equilibria on a congested transportation network. SIAM J Alg Discrete Meth 2:213–226
2. Berman O, Larson RC (1982) The median problem with congestion. Comput Oper Res 4:119–126
3. Berman O, Larson RC, Chiu SS (1985) Optimal server location on a network operating as an M/G/1 queue. Oper Res 33:746–771
4. Berman O, Larson RC, Parkan C (1987) The stochastic queue p-median problem. Transport Sci 21:207–216
5. Berman O, Mandowski RR (1986) Location-allocation on congested networks. Europ J Oper Res 26:238–250
6. Braid RM (1991) The locations of congestible facilities in adjacent jurisdictions. Regional Sci and Urban Economics 21:617–626
7. Brandeau ML, Chiu SS (1992) A center location problem with congestion. Ann Oper Res 40:17–32
8. Brandeau ML, Chiu SS (1994) Facility location in a user-optimizing environment with market externalities: Analysis of customer equilibria and optimal facility locations. Location Sci 2:129–147
9. Brandeau ML, Chiu SS (1994) Location of competing private facilities in a user-optimizing environment with market externalities. Transport Sci 28:125–140
10. Brandeau ML, Chiu SS, Kumar S, Grossman TA (1995) Location with market externalities. In: Drezner Z (ed) Facility Location: A Survey of Applications and Methods. Springer, Berlin, pp 121–150
11. Chiu SS, Larson RC (1985) Locating an n-server facility in a stochastic environment. Comput Oper Res 12:509–516
12. Daskin MS (1995) Network and discrete location. Wiley/Interscience, New York
13. Drezner Z (eds) (1995) Facility location: A survey of applications and methods. Springer, Berlin
14. Friesz TL, Tobin RL, Cho HJ, Mehta NJ (1990) Sensitivity analysis based heuristic algorithms for mathematical Programs with variational inequality constraints. Math Program 48:265–284
15. Ghosh A, Craig CS (1983) Formulating retail location strategy in a changing environment. J Marketing 47:56–68
16. Grossman TA (1993) Models for optimal facility utilization decisions. Unpublished PhD Diss., Stanford University
17. Kohlberg E (1983) Equilibrium store locations when consumers minimize travel time plus waiting time. Economics Lett 11:211–216
18. Kumar S (1993) Location decisions for private facilities in the presence of elastic demand and externalities. Unpublished PhD Diss., Stanford University
19. Larson RC (1974) A hypercube queueing model for facility location and redistricting in urban emergency services. Comput Oper Res 1:67–95
20. Tomlin JA (1971) A mathematical programming model for the combined distribution-assignment of traffic. Transport Sci 5:120–140
21. Wardrop JG (1952) Some theoretical aspects of road traffic research. Proc Inst Civil Eng Part II:325–378

Facility Location Problems with Spatial Interaction

KAJ HOLMBERG
Department Math., Linköping Institute Technol., Linköping, Sweden

MSC2000: 90B80, 90C10

Article Outline

Keywords
Model
Solution Methods
The Dual Ascent and Adjustment Method
Dual Ascent and Branch-And-Bound
Lagrangian Relaxation
  with Subgradient Optimization
Benders Decomposition
Mean Value Cross Decomposition

Comparisons and the Role of Γ
Conclusion
See also
References

Keywords

Location; Spatial interaction; Integer programming; Dual ascent

Facility location problems deal with the question of where to locate certain facilities, so that they can satisfy some kind of demand of a certain set of customers, and so that the total cost is minimized. If the facilities are factories or warehouses and the goods will be shipped from the facilities to the customers, one can assume that the shipments will be made so as to minimize the transportation costs. (See also ► Facility Location with Staircase Costs and ► Stochastic Transportation and Location Problems.) However, if the facilities are hospitals or supermarkets, transportation will consist of customers traveling by their own means to/from the facility, and in such a case, it is not certain that each customer will behave exactly so as to minimize the transportation costs.

So in public facility location problems where the clients are free to make their own choice of facility, one should probably expect different results than those minimizing the transportation costs. Modeling such situations, the objective cannot only be to minimize the total transportation and facility costs. The effect of *spatial interaction* has been used to improve location models of this type. Simple plant location problems with spatial interaction between the travelers have been treated in [3,4,6,19,20,22,23], modeled as a nonlinear, mixed integer programming problem.

In [15] a different model is derived, in a similar way as used in [14], that does not use the approximation yielding entropy terms. The model is called the 'exact' formulation of the simple plant location problem with spatial interaction, because of the usage of the classical way of deriving the gravity model, without doing any approximation.

Assuming integer requirements on the transported amounts enables an exact linearization of the nonlinear costs. This yields a linear, pure zero-one model, to the price of a significantly increased number of variables. Luckily the model has a special structure that can be exploited by several different solution methods.

Model

We now describe a public facility location model, with m possible locations for supply points (plants) and n demand points (client zones). The fixed cost for opening plant i is a_i. At demand point j the demand (the number of clients in zone j) is w_j. Trips will be made between the demand points and the opened plants so that the demand is satisfied. The transportation costs for one trip between plant i and demand point j (i. e. the cost for a client at zone j to get service at plant i) is c_{ij}.

The following variables are introduced.

$$z_i = \begin{cases} 1 & \text{if a plant at location } i \text{ is opened,} \\ 0 & \text{if not,} \end{cases}$$

$$x_{ij} = \begin{cases} \text{the number of trips between} \\ \text{plant } i \text{ and demand point } j \\ \text{(i. e. the number of clients} \\ \text{in zone } j \text{ getting service at plant } i). \end{cases}$$

The total cost for transportation and opening plants is

$$v_1 = \min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} a_i z_i.$$

As for the spatial interaction, one may note that several *microstates* (obtained by identifying every single client's trip) may yield the same *macrostate* (the x-solution). The macrostate given by the largest number of microstates is the most probable solution, according to [29]. Maximizing the number of microstates yielding x yields another objective function for finding the most likely x-solution.

$$v_2 = \min \sum_{i=1}^{m} \sum_{j=1}^{n} \ln(x_{ij}!).$$

A suitable objective function is now obtained by combining these two parts, $v* = v_2 + \gamma v - 1$, where the weight γ reflects the sensitivity of the system to the costs. For large values of γ, it is very important to minimize the costs, while for smaller values of γ, the costs are not very important.

The best value of the parameter γ, being the weight of how much the clients take the costs into account, must be found by calibrating the model against a real life situation. Considering a certain situation, one can assume that γ is fixed and given.

The following model (SPLPS) is obtained:

$$v^* = \min \sum_{i=1}^{m}\sum_{j=1}^{n} \ln(x_{ij}!) + \gamma \left(\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} a_i z_i \right)$$

such that

$$\sum_{i=1}^{m} x_{ij} = w_j, \quad \forall j \tag{1}$$

$$x_{ij} - w_j z_i \le 0, \quad \forall i, j \tag{2}$$

$$x_{ij} \ge 0, \quad \text{integer}, \quad \forall i, j \tag{3}$$

$$z_i \in \{0, 1\}, \quad \forall i. \tag{4}$$

SPLPS is a pure integer problem, with a nonlinear objective function that actually is defined only in the integer points.

If γ is so large that the logarithmic part is negligible, we get pure cost minimization. The model is then identical to the simple plant location problem, SPLP, and can be efficiently solved by for example a dual ascent method, [5].

In previous work, a continuous relaxation of x together with Stirling's approximation, $\ln(x_{ij}!) \approx x_{ij} \ln(x_{ij}) - x_{ij}$, have been used, yielding a nonlinear, mixed integer programming problem.

Now we linearize the cost function for each variable x_{ij} in the interval $0 \le x_{ij} \le w_j$, with break points at each integer point. This does not introduce any error (as Stirling's approximation would). The number of variables then depends on the values of the demands.

We get $\bar{c}_{ijk} = \ln(k!) - \ln((k-1)!) + \gamma c_{ij} = \ln(k) + \gamma c_{ij}$. Note that $\bar{c}_{ijk} > \bar{c}_{ijk-1}$, [18], which indicates convexity of the resulting cost functions.

Then we do the substitution $x_{ij} = \sum_k x_{ijk}$, where x_{ijk} is the amount of x_{ij} that falls in the interval $(k-1, k)$. The following model (SPLPE) is obtained:

$$v^* = \min \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{w_j} \bar{c}_{ijk}x_{ijk} + \sum_{i=1}^{m} \gamma a_i z_i$$

such that

$$\sum_{i=1}^{m}\sum_{k=1}^{w_j} x_{ijk} = w_j, \quad \forall j, \tag{5}$$

$$x_{ijk} - z_i \le 0, \quad \forall i, j, k, \tag{6}$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i, j, k, \tag{7}$$

$$z_i \in \{0, 1\}, \quad \forall i. \tag{8}$$

This is a large linear integer programming problem with $m(1 + \sum_{j=1}^{n} w_j)$ binary variables. The fact that it is a pure 0–1-problem is favorable when it comes to solution methods. The coefficients in the constraints (6) are all reduced to one, so the formulation is probably quite strong.

Solution Methods

It is in principle possible to solve SPLPE with a standard integer programming code, but the size of the model prohibits this for all instances but very small ones. As the model is fairly new, one cannot find many solution methods proposed in the literature.

A dual ascent procedure for this problem has been developed, see [17]. Another method, based on the same dual, is Lagrangian relaxation and subgradient optimization, investigated in [18]. Solution methods based on primal and dual decomposition techniques can also be used, see [16], where one conclusion is that Benders decomposition seems to be an efficient solution method. In [13], the dual ascent approach is inserted in a branch and bound framework, and applied to a somewhat more general problem.

We will briefly describe these methods below.

The Dual Ascent and Adjustment Method

A dual ascent procedure can be used to, in principle, solve the LP-dual of the LP relaxation of SPLPE, by increasing the dual variables in small steps, in such a way that an ascent of the dual function is obtained in each step. Furthermore, a dual adjustment procedure can be used to temporarily decrease dual variables that block further improvement.

Let α_j denote the dual variables corresponding to constraint set 1 of the LP relaxation of SPLPE, β_{ijk} the

dual variables corresponding to constraint set 2 and δ_i the dual variables corresponding to the constraints $z_i \leq 1$. The LP-dual will be as follows:

$$v^* = \max \sum_{j=1}^{n} w_j \alpha_j + \sum_{i=1}^{m} \delta_i$$

such that

$$\alpha_j - \beta_{ijk} \leq \bar{c}_{ijk}, \quad \forall i, j, k, \tag{9}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{w_j} \beta_{ijk} - \delta_i \leq \gamma a_i, \quad \forall i, \tag{10}$$

$$\beta_{ijk} \geq 0, \quad \forall i, j, k, \tag{11}$$

$$\delta_i \geq 0, \quad \forall i. \tag{12}$$

The basic steps are to make moves in the dual variables α_j. For fixed $\alpha = \overline{\alpha}$, the LP-dual is trivially solvable, yielding

$$\beta_{ijk} = \max(0, \overline{\alpha}_j - \bar{c}_{ijk}), \quad \forall i, j, k,$$

and

$$\delta_i = \max \left(0, \sum_{j=1}^{n} \sum_{k=1}^{w_j} \beta_{ijk} - \gamma a_i \right), \quad \forall i.$$

Now let k_{ij} be such that

$$\overline{\alpha}_j \geq c_{ijk}, \quad \forall k \leq k_{ij}, \qquad \overline{\alpha}_j < c_{ijk}, \quad \forall k > k_{ij},$$

and

$$q_{ij} = \begin{cases} 1 & \text{if } \overline{\alpha}_j = c_{ijk_{ij}} \\ 0 & \text{if not.} \end{cases}$$

Also, let

$$s_i = \sum_{j=1}^{n} \sum_{k=1}^{w_j} \max(0, \overline{\alpha}_j - c_{ijk})$$

and define $I^{>}$={i: $s_i > \gamma\ a_i$}, $I^{=}$={i: $s_i = \gamma a_i$}, $I^{<}$ = {i: $s_i < \gamma a_i$ and $I^{\geq} = I^{>} \cup I^{=}$.

The complementary slackness conditions are

$$x_{ijk} = z_i, \quad \forall k \leq k_{ij} - q_{ij}, \quad \forall i, j,$$
$$x_{ijk} = 0, \quad \forall k > k_{ij}, \quad \forall i, j,$$
$$z_i = 1, \quad \forall i \in I^{>},$$
$$z_i = 0, \quad \forall i \in I^{<}.$$

Now define

$$w_j^l(\overline{\alpha}) = \sum_{i \in I^{>}} (k_{ij} - q_{ij})$$

and

$$w_j^u(\overline{\alpha}) = \sum_{i \in I^{\geq}} k_{ij}.$$

Then it can be shown, [17], that

$$w_j^l(\overline{\alpha}) \leq \sum_{i=1}^{m} \sum_{k=1}^{w_j} x_{ijk} \leq w_j^u(\overline{\alpha}).$$

This means that w_j^l and w_j^u are lower and upper bounds on the left-hand sides of constraints (9). In order to obtain feasibility (optimality in the dual) the intervals between these bounds should contain the right-hand sides w_j. The following is proved in [17]: If $w_j^l(\overline{\alpha}) \leq w_j \leq w_j^u(\overline{\alpha})$, $\forall j$, then α is optimal in the LP relaxation of SPLPE.

The dual ascent method is now to increase α_j in small steps, so that $w_j^l(\overline{\alpha})$ and $w_j^u(\overline{\alpha})$ increase. The increase of a certain α_j is bounded by the closest breakpoint, induced by the dual constraints of either set 1 (corresponding to enabling or forcing the increase of yet another x_{ijk}) or set 2 (corresponding to enabling or forcing the increase of yet another z_i).

The bounds w_j^l and w_j^u will approach w_j from below, and w_j^l will not be allowed to exceed w_j. The increase of α is repeated, in each step for the j which yields the largest distance between w_j^u and w_j, until optimum is found or improvement is blocked (i. e. a further increase of any α_j would result in $w_j^l > w_j$). In the last case we use an adjustment procedure, which decreases some α_j, in order to allow the increase of other α_j's. Then the ascent phase above is repeated. More details can be found in [17].

Dual Ascent and Branch-And-Bound

The dual ascent and adjustment procedure only solves the LP relaxation of the problem, so to find the exact integer optimum, the procedure must be used within a branch and bound framework. The subproblem in each node of the branch and bound tree is then solved with the dual ascent procedure, in the sense that lower bounds on the optimal objective function value

and sometimes feasible primal solutions are obtained. Branches are cut off when the lower bound exceeds the best upper bound known.

One can note that if all the z-variables are fixed in SPLPE, then the problem is trivially solvable, and the x-variables will attain integer values even if the constraints $x_{ijk} \in \{0, 1\}$ are replaced by $0 \le x_{ijk} \le 1$, so it may be regarded as an LP-problem. Furthermore it is proved, in [13], that the dual ascent procedure accurately solves the problem when all z-variables are fixed, within a finite number of steps.

Therefore, it is natural to do the branching over the z-variables. Fixed z-variables are handled as follows in the dual ascent phase. Let $I_0 = \{I: z_i \text{ is fixed to } 0\}$ and $I_1 = \{I: z_i \text{ is fixed to } 1\}$. For all $i \in I_0 \cup I_1$, the dual variables δ_i are removed, and the corresponding dual constraints in set 2 are removed. For all $i \in I_1$ the corresponding primal constraints in set 2 are redundant, so we can assume that $\beta_{ijk} = 0$, $\forall\ i \in I_1$, $\forall j, k$. Also, $x_{ijk} = 0$, $\forall\ i \in I_0$, $\forall\ j, k$.

All elements of $I_0 \cup I_1$ must be removed from $I^>$, $I^=$, $I^<$ and $I^\ge$. It is not necessary to calculate s_i for $i \in I_0 \cup I_1$. After these changes, the bounds w_j^l and w_j^u are calculated as above.

Some supporting hyperplanes, and breakpoints, are removed from the dual function, as a result of the fixations, so in the dual ascent procedure, fewer steps often need to be taken. (Sometimes the increase of some α_j is limited by the breakpoint where a facility is opened. This will not occur if the facility is fixed open or closed.)

In the worst case, the branch and bound method will enumerate all z-solutions. Thus we have the following result: The dual ascent method within a branch and bound framework will find the exact optimum of SPLPE within a finite number of steps.

In practice branching is done when the dual ascent and adjustment procedure stops, which not necessarily means that the LP-optimum is found. In many cases unnecessary branching is done, and we must expect the branch and bound tree to be larger than it would be for an LP-based branch and bound method.

Branching is done over any z_i, $i \in I^=$, since any value between 0 and 1 is optimal for such a z_i, i. e. the complementary slackness conditions allow for nonintegral values of z_i.

The original dual ascent method starts from zero (no facilities opened and nothing sent). However, for very small values of γ in SPLPS many facilities will be opened, while for very large values of γ the z-solution obtained for the ordinary uncapacitated facility location problem, SPLP, by for examples the dual ascent method DUALOC, [5], might be optimal or close to optimal in SPLPS. In such cases one can use these solutions as starting solutions.

The choice of which dual variable, α_j, to increase first in the dual ascent procedure, could be done cyclically in j, but it seems better to choose the j which exhibits the maximal residual, i. e. the largest gap between w_j^l and w_j.

Lagrangian Relaxation with Subgradient Optimization

Lagrangian relaxation is a well known and often used approach for approximate solution of integer and mixed integer problems, see for example [8] and [7]. The Lagrangian relaxation of SPLPE is obtained by relaxing the demand constraints, using multipliers α_j. We obtain the following Lagrangian dual:

$$\text{(LD)} \quad v_L = \max \varphi(\alpha),$$

where, for fixed multipliers, $\alpha = \overline{\alpha}$, the Lagrangian relaxation takes the following form:

$$\text{(DS)} \quad \begin{cases} \varphi(\overline{\alpha}) & = \min \displaystyle\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{w_j} \overline{c}_{ijk} x_{ijk} \\ & \quad + \displaystyle\sum_{i=1}^{m} \gamma a_i z_i \\ & \quad + \displaystyle\sum_{j=1}^{n} \overline{\alpha}_j \left(\sum_{i=1}^{m}\sum_{k=1}^{w_j} x_{ijk} - w_j \right) \\ \text{s.t.} & \quad x_{ijk} - z_i \le 0, \quad \forall i, j, k, \\ & \quad x_{ijk} \in \{0, 1\}, \quad \forall i, j, k, \\ & \quad z_i \in \{0, 1\}, \quad \forall i. \end{cases}$$

(DS) separates into m problems, one for each facility, containing one binary variable and a number of continuous variables, and is trivially solvable. It has the 'integrality property', i. e. the y-solution will obtain integral values even if the integrality constraints are removed. This property implies that the optimal value of LD is the same as that of the LP relaxation.

In order to solve the Lagrangian dual, we use the well known technique subgradient optimization, see [25,26] and [9]. We use a subgradient, ξ, of the dual function $\varphi(\alpha)$ to find a search direction for updating the multipliers, α:

$$\xi_j = \sum_{i=1}^{m}\sum_{k=1}^{w_j} \overline{x}_{ijk} - w_j, \quad \forall j,$$

where $\overline{x}$ denotes the optimal solution of (DS) at $\overline{\alpha}$. Letting $\overline{\alpha}^{(l)}$ denote the multiplier values in iteration l, we obtain the multipliers in the next iteration by setting

$$\overline{\alpha}^{(l+1)} = \overline{\alpha}^{(l)} + t^{(l)}\xi^{(l)},$$

where $t^{(l)}$ and $\xi^{(l)}$ are the stepsize and the search direction. Several ways of choosing the stepsize, $t^{(l)}$, have been suggested. Here we use the one that is suggested by [26]:

$$t^{(l)} = \lambda_l \frac{\widetilde{\nu} - \varphi(\overline{\alpha}^{(l)})}{\left\|\xi^{(l)}\right\|^2},$$

where $\widetilde{\nu}$ is an upper bound of ν_L and λ_l should be assigned a value in the interval $(\varepsilon_1, 2-\varepsilon_1)$, where $\varepsilon_1 > 0$, in order to ensure convergence.

Termination of the subgradient search procedure occurs when $\|d^{(l)}\| < \epsilon$, $t^{(l)} < \epsilon$, $l > M$, $\overline{\nu} - \varphi(\overline{\alpha}^{(l)}) \le \epsilon$ or $\overline{\nu} - \underline{\nu} < 1$. The last criterion indicates optimality, since all feasible solutions are integral, i. e. ν^* is integral.

Benders Decomposition

We have noted that if all z-variables were fixed, the solution would not be changed if the constraints $x_{ijk} \in \{0,1\}$ were replaced by $0 \le x_{ijk} \le 1$. Therefore one might regard SPLPE as a mixed integer programming problem. This opens up the possibility of solving the problem with Benders decomposition, [1]. Below we give a short description of how the method can be applied to SPLPE, as done in [16].

In the Benders subproblem, (PS), we fix z to $\overline{z}$, which makes the subproblem separable into several trivial knapsack problems:

$$\text{(PS)} \quad h(\overline{z}) = \sum_{j=1}^{n} h_j(\overline{z}) + \sum_{i=1}^{m} \gamma a_i \overline{z}_i,$$

where, $\forall\, j$,

$$\begin{cases} h_j(\overline{z}) & = \min \displaystyle\sum_{i=1}^{m}\sum_{k=1}^{w_j} \overline{c}_{ijk} x_{ijk} \\ \text{s.t.} & \displaystyle\sum_{i=1}^{m}\sum_{k=1}^{w_j} x_{ijk} = w_j, \\ & x_{ijk} \le \overline{z}_i, \quad \forall i, \\ & x_{ijk} \in \{0,1\}, \quad \forall i,k. \end{cases}$$

(PS) is feasible if and only if $\sum_i \overline{z}_i \ge 1$. The dual solution (α, β) is also easy to calculate.

The Benders master problem is given below.

$$\text{(PM)} \quad \begin{cases} \nu_{\text{PM}} & = \min \displaystyle\sum_{j=1}^{n} q_j + \sum_{i=1}^{m} \gamma a_i z_i \\ \text{s.t.} & q_j \ge w_j \alpha_j^{(l)} - \displaystyle\sum_{i=1}^{m}\sum_{k=1}^{w_j} \beta_{ijk}^{(l)} z_i, \\ & \quad \forall l, j, \\ & \displaystyle\sum_{i=1}^{m} z_i \ge 1, \\ & z_i \in \{0,1\}, \quad \forall i. \end{cases}$$

The Benders decomposition method is to iterate between the master problem, (PM), and the subproblem, (PS). (PM) yields a lower bound on ν^*, and $\overline{z}$ to be used in (PS). (PM) yields an upper bound on ν^* (for integral $\overline{z}$) and a new dual solution, $(\alpha_j^{(l)}, \beta_{ijk}^{(l)})$, which is used to form a new cut for the master problem. The method has exact finite convergence.

The proportion of z-variables is much smaller in SPLPE than in SPLP, which is promising for the Benders decomposition approach. However, as shown computationally in [16], (PM) often is very difficult to solve. A suggested modification, [24], is to use the *LP relaxation* of (PM), by replacing $z_i \in \{0,1\}$ with $0 \le z_i \le 1$, in initial iterations (for example until the LP-bounds are within 1% of each other). A good set of Benders cuts is thus generated before the integer master problem is solved. It is possible since any dual feasible solution of (PS) yields a valid Benders cut, and $\overline{z}$ only appears in the dual objective function.

If $\overline{z}$ is not integer, (PS) might not yield integer x-solutions, but is still easily solvable. The bounds obtained from the master problem and subproblem are not valid for the integer problem, but for the LP relaxation of SPLPE.

Mean Value Cross Decomposition

An alternate way of solving the LP relaxation of the problem is to use the method mean value cross decomposition, [10,11,12]. This method is a modification of the subproblem phase of ordinary cross decomposition, [28], but also a generalization of the Kornai–Liptak method, [21], and a generalization of the Brown–Robinson methods for polyhedral games, [2,27].

The method uses the Lagrangian relaxation and the subproblem of Benders decomposition, both described in previous sections, but no master problems. The input to one of the problems consists of the mean value of all the previous solutions of the other subproblem. The method has asymptotic convergence.

Comparisons and the Role of Γ

The parameter γ reflects the relation between the transportation costs and the effects of the spatial interaction in the objective function, and its value should be chosen specifically for each real life situation.

For very small values of γ, the optimal solution is $z_i = 1$, $\forall\ i$, while for large values of γ, the optimal z-solution can be obtained by DUALOC. In these cases the problem is then completely solved by simply solving the primal subproblem, (PS), once. In computational tests in [16,17,18] and [13], this occurs when γ is smaller than 0.0001 or larger than 0.1, while for $\gamma = 0.01$ the differences to the solutions mentioned above are the largest.

The conclusions of the computational tests in [16,17,18] and [13] are the following. Ordinary Benders decomposition seems to be more efficient than direct solution with a general integer programming code. However, direct solution with a standard IP-code can only solve small problems, due to memory requirements, and the ordinary Benders decomposition method also fails for many of the problems. The integer master problem is simply too hard.

The modified Benders decomposition method (starting with the LP relaxation of the master problem) eliminates the weaknesses of the Benders approach, and is a very efficient method.

The approximate methods mean value cross decomposition and Lagrangian relaxation with dual subgradient optimization are much quicker than ordinary Benders, but not better than modified Benders decomposition. For some large problems, these methods give large gaps between the upper and lower bounds.

The dual ascent method is also quite quick, but leaves gaps between the upper and lower bounds of varying size. In [18] it is noted that the dual ascent method and the Lagrangian method complement each other in an interesting way.

The best methods seems to be the modified Benders decomposition method and the dual ascent method with branch and bound. These methods are capable of solving quite large problems (up to almost 3,000,000 variables) optimally.

Finally we wish to point out that none of these methods explicitly store the whole x-matrix, and that this is what enables the solving of large problems.

Conclusion

We have described the simple plant location problem with spatial interaction, applied an exact linearization to the problem, and described a couple of solution methods for the resulting large integer programming problem. Although the model has a large number of variables, the methods are able to solve it quite efficiently. The problem is very well suited for the approaches of Benders decomposition, Lagrangian relaxation, and dual ascent. These methods actually manage to solve the problem without storing all of the variables, and especially the dual ascent method uses relatively small amounts of computer memory.

We conclude that the model is solvable and useful in practice.

See also

- ► Combinatorial Optimization Algorithms in Resource Allocation Problems
- ► Competitive Facility Location
- ► Facility Location with Externalities
- ► Facility Location with Staircase Costs
- ► Global Optimization in Weber's Problem with Attraction and Repulsion
- ► MINLP: Application in Facility Location-allocation
- ► Multifacility and Restricted Location Problems
- ► Network Location: Covering Problems
- ► Optimizing Facility Location with Rectilinear Distances

- ▶ Production-distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Stochastic Transportation and Location Problems
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Math 4:238–252
2. Brown GW (1949) Some notes on computation of games solutions. RAND Report P-78
3. Coelho JD (1980) A locational surplus maximization approach to public facility location. Nota Centro de Estasica e Aplic Dept Mat Aplic Fac Ciences de Lisboa, Portugal, no. 15
4. Coelho JD, Wilson AG (1976) The optimum location and size of shopping centres. Regional Studies 4:413–421
5. Erlenkotter D (1978) A dual-based procedure for uncapacitated facility location. Oper Res 26:992–1009
6. Erlenkotter D, Leonardi G (1985) Facility location with spatially interactive travel behaviour. Sistemi Urbani 1:29–41
7. Fisher ML (1981) The Lagrangean relaxation method for solving integer programming problems. Managem Sci 27(1):1–18
8. Geoffrion AM (1974) Lagrangean relaxation for integer programming. Math Program Stud 2:82–114
9. Held M, Wolfe P, Crowder HP (1974) Validation of subgradient optimization. Math Program 6:62–88
10. Holmberg K (1992) Linear mean value cross decomposition: A generalization of the Kornai–Liptak method. Europ J Oper Res 62:55–73
11. Holmberg K (1994) A convergence proof for linear mean value cross decomposition. Z Oper Res 39(2):157–186
12. Holmberg K (1997) Mean value cross decomposition applied to integer programming problems. Europ J Oper Res 97:124–138
13. Holmberg K (1999) Exact solution methods for uncapacitated location problems with convex transportation costs. Europ J Oper Res 114:127–140
14. Holmberg K, Jörnsten K (1989) Exact methods for gravity trip distribution models. Environm Plan A 21:81–97
15. Holmberg K, Jörnsten K (1990) An exact formulation of the simple plant location problem with spatial interaction. In: Orban-Ferauge Fr, Rasson JP (eds) Proc Meeting V of the EURO Working Group on Locational Analysis, Fac Univ Notre-Dame de-la-Paix, Namur
16. Holmberg K, Jörnsten K (1995) Decomposition methods for the exact formulation of the simple plant location problem with spatial interaction. Res Report, Dept Math, Linköping Inst Techn, LiTH-MAT-R-1995–13
17. Holmberg K, Jörnsten K (1996) A dual ascent procedure for the exact formulation of the simple plant problem with spatial interaction. Optim 36:139–152
18. Holmberg K, Jörnsten K (1996) Dual search procedures for the exact formulation of the simple plant location problem with spatial interaction. Location Sci 4:83–100
19. Jacobsen SK (1986) Plant location by entropy maximization. Res Report IMSOR, DTH, Lyngby, Denmark 10
20. Jacobsen SK (1988) On the solution of an entropy maximizing location model by submodularity. Res Report, IMSOR, DTH, Lyngby, Denmark 6
21. Kornai J, Liptak Th (1965) Two-level planning. Econometrica 33:141–169
22. Leonardi G (1978) Optimum facility location by acessibility maximizing. Environm Plan A 11:1287–1305
23. Leonardi G (1983) The use of random-utility theory in building location-allocation models. In: Thisse JF, Zoller HG (eds) Location Analysis of Public Facilities. North-Holland, Amsterdam, pp 357–383
24. McDaniel D, Devine M (1977) A modified Benders partitioning algorithm for mixed integer programming. Managem Sci 24:312–319
25. Poljak BT (1967) A general method of solving extremum problems. Soviet Math Dokl 8:593–397
26. Poljak BT (1969) Minimization of unsmooth functionals. USSR Comput Math Math Phys 9:14–29
27. Robinson J (1951) An iterative method of solving a game. Ann of MATH 54:296–301
28. Roy TJ Van (1983) Cross decomposition for mixed integer programming. Math Program 25:46–63
29. Wilson AG (1967) A statistical theory of spatial distribution models. Transport Res 1:253–269

Facility Location with Staircase Costs

KAJ HOLMBERG
Department Math., Linköping Institute Technol., Linköping, Sweden

MSC2000: 90B80, 90C11

Article Outline

Keywords
Mathematical Model
Solution Methods
Primal Heuristics
Linearization

Convex Piecewise Linearization
Benders Decomposition
Lagrangian Relaxation
and Subgradient Optimization
Computational Results
Conclusion
See also
References

Keywords

Facility location; Mixed integer programming; Heuristics

Location of facilities, plants, or other units for production or distribution, is an important problem in many different situations. The same type of problem can occur when one is installing equipment in for example telecommunication networks, or when installing machines in a factory.

The common circumstances in these situations are the following. A number of units, 'facilities', producing a certain service, may be located at certain possible points. The service commodity is then to be sent from the facilities to certain 'customer' points, which have a certain demand for the service. The main complication is that the costs for production of the service is not linear, instead there is a fixed cost for placing a facility at a certain location. In addition there may be linear costs for producing and shipping the commodity to the customers.

In the literature, see for example [3,4,6,8,13] and [1], one can find the traditional capacitated plant location model, where the total cost for satisfying demand consists of two parts, namely linear transportation costs and fixed costs for opening/installing the facilities. In this model there is one fixed cost for each facility. (Other variants can be found in ► Stochastic transportation and location problems and ► Facility location problems with spatial interaction.)

However, in practice, there is often a need for considering several different possible sizes of each facility. This leads to a *facility location problem* with *staircase* shaped costs. This approach will not only allow different sizes, but also different production costs at different levels of production at a facility.

For example, in telecommunications there are almost always several different sizes for the fibers, cables, switches, controllers and other connections that must be dimensioned when installing a new network. In such problems staircase shaped costs will occur at several different levels, both for the activities at nodes as well as along links. One situation where the specific location model discussed here is quite appropriate is the installation of video servers for a video-on-demand service on a telecommunication network.

Mathematical Model

We define a *staircase cost function* as a finite piecewise linear nondecreasing function with a finite set of discontinuities, each corresponding to a certain size of a facility. Let m be the number of possible location sites, n the number of customers and q_i the number of possible sizes at location site i. Furthermore, D_j is the demand of customer j, p_{ik} is the unit cost of production at a facility at location site i and size k, S_{ik} is the capacity of a facility of size k at location site i, f_{ik} is the fixed cost for a facility of size k at location site i, and c_{ij} is the cost for sending one unit from location site i to customer j.

The following variables are used: t_{ik} is the production within level k at facility i (where level k of the staircase corresponds to an operating facility of size k), x_{ij} is the amount shipped from location i to customer j, and y_{ik} is set to 1 if the facility at site i is of size k or larger and 0 otherwise.

The capacities and costs for increasing the size of a facility are $\Delta S_{ik} = S_{ik} - S_{ik-1}$ and $\Delta f_{ik} = f_{ik} - f_{ik-1} - (p_{ik} - p_{ik-1})\, S_{ik-1}$, where $S_{i0} = 0$ and $f_{i0}, = 0$, see Fig. 1. Note that $0 \le t_{ik} \le \Delta S_{ik}$, $\forall$i, k, and if the total production at facility i requires more than size k, then $t_{ik} = \Delta S_{ik}$.

$$v^* = \min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} \sum_{k=1}^{q_i} (p_{ik} t_{ik} + \Delta f_{ik} y_{ik}),$$

such that

$$\sum_{i=1}^{m} x_{ij} = D_j, \quad \forall j, \tag{1}$$

$$\sum_{j=1}^{n} x_{ij} = \sum_{k=1}^{q_i} t_{ik}, \quad \forall i, \tag{2}$$

$$t_{ik} \leq \Delta S_{ik} y_{ik}, \quad \forall i, k, \tag{3}$$

$$t_{ik-1} \geq \Delta S_{ik-1} y_{ik}, \quad \forall i, k > 1, \tag{4}$$

$$x_{ij} \geq 0, \quad \forall i, k, \tag{5}$$

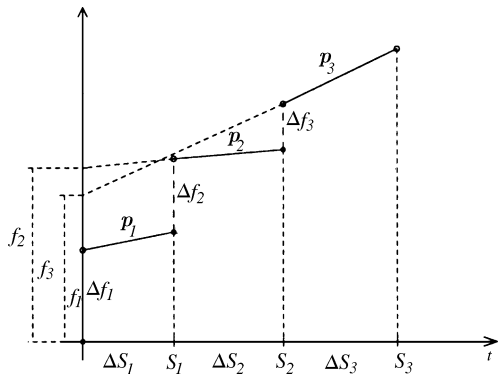
$$y_{ik} = 0 \text{ or } 1, \quad \forall i, k. \tag{6}$$

It is natural to assume that $p_{ik} \geq 0$, $\Delta f_{ik} \geq 0$, $\Delta S_{ik} > 0$ for all i, k and $D_j > 0$ for all j. The constraints (1) ensure that all the demand must be met for each customer, while (2) ensure that, for each location, the amount shipped also is produced. Constraint sets (3) and (4) ensure that the level of production corresponds to the correct level on the staircase cost function for each facility. One might note that from constraints (3) and (4) follows that $y_{ik+1} \leq y_{ik}$.

This is a linear mixed integer programming problem with $mn + \sum_{i=1}^{m} q_i$ continuous variables and $\sum_{i=1}^{m} q_i$ integer variables. The proportion of integer variables is higher than in the ordinary facility location problem. Because of this, and because of the structure of the problem, solving the problem with a general code for mixed integer programming problems is probably not very efficient for large (real life) instances.

One aspect of the structure of the problem is that if y is kept fixed (i. e. the sizes of the facilities are given), the remaining problem is simply a standard network flow problem, and hence x and t will attain integer values.

Another important aspect of the structure of the problem is the potential separability. There are several possibilities of making the model separable by relaxing different sets of constraints.



Facility Location with Staircase Costs, Figure 1

It is also possible to use a problem formulation with f and S instead of Δf and ΔS. This yields constraints of SOS1-type (one must ensure that only one of the possible sizes is used at a facility), and a somewhat smaller problem (less constraints). The LP relaxation is quicker to solve and the optimal objective function value is the same as that of the model above (i. e. the duality gaps of the two formulations are the same). However, solving the model with general mixed integer codes, the alternate model seems to produce larger branch and bound trees. Concerning the methods discussed below, the two models in most cases behave in identical manners.

Solution Methods

Methods for models with staircase cost functions or for models capable of modeling such functions can be found in for example [2,11,14,15] and [12]. We will below describe some possibilities.

If the exact solution is to be found (and verified), the only reasonable way seems to be to resort to branch and bound, in some sense. This matter in general is extensively discussed in the literature, and although there might be some considerations for the staircase cost case that differ from the single fixed cost case, when it comes to branching and search strategies, we will not dwell on it here.

Assuming a standard branch and bound framework, the main question is how to solve the subproblems, i. e. how to get the bounds, especially the lower bounds. This will be discussed more below.

However, an alternative is to move the branch and bound procedure into a Benders master problem, i. e. use a Benders decomposition framework in order to obtain the exact solution. This will also be briefly described below.

We will start with procedures for obtaining upper and lower bounds on the optimal objective function value. The upper bounds correspond to feasible solutions obtained, while the lower bounds are used to get estimates of the quality of the feasible solutions. If all cost coefficients are integral, we note that a lower bound that is within one unit from the upper bound indicates that the upper bound is optimal.

Primal Heuristics

There is a well-known ADD heuristic for the capacitated plant location problem, [13], which can be modified to suit the staircase cost facility location problem, see [12]. This heuristic can be improved by combining it with certain priority rules, [6].

If for each plant it is decided to which level of production it can be used (i. e. the y-variables are fixed), the resulting problem is an unbalanced transportation problem. Let L_i denote the level (size) of plant i, and initiate the heuristic by setting $L_i = 0$, $\forall i$. Let $I = \{i\colon L_i < q_i\}$. The ADD heuristic consists of the repeated use of the following step: Increase the size (set $L_i = L_i + 1$) of the location site $i \in I$ that provides the largest reduction of the total cost. Terminate the procedure when no more reduction is possible.

In order to avoid ADD increasing the level of production in the order of 'decreasing' capacity until a feasible solution is found, we apply a generalization of one of the priority rules discussed in [6]. These priority rules provides a better phase-1 solution than the ADD heuristic itself. Two examples of priority rules, PR1 and PR2, for choosing the location site $i \in I$ where the size is to be increased ($L_i = L_i + 1$), are given below. (They correspond to P1 and P3 in the notation of [6]).

PR1) Choose site $i \in I$ in the order of decreasing quotients $\Delta S_{i,L_i+1}/\Delta f_{i,L_i+1}$, until the location sites are able to serve the entire demand.

PR2) Choose site $i \in I$ in the order of increasing values of

$$\frac{1}{\lfloor n/3 \rfloor} \sum_{j=1}^{\lfloor n/3 \rfloor} \overline{c}_{ij} + \frac{\Delta f_{i,L_i+1}}{\Delta S_{i,L_i+1}},$$

until the location sites are able to serve the entire demand. ($\overline{c}$ is c sorted according to increasing values.)

In [13] the ADD heuristic is outperformed by the heuristic DROP but [6] show that ADD with priority rules produce solutions with equally good objective function values as DROP, in less computational time.

Linearization

A widely used way of obtaining a lower bound is direct LP relaxation. The integer requirements (6) are replaced with the constraints $0 \le y_{ik} \le 1$, $\forall i, k$. We also include the redundant constraints $t_{ik} \le \Delta S_{ik}$, $\forall i, k$, and possibly $y_{ik} \le y_{ik-1}$ for all i, $k > 1$. The optimal objective function value of the LP relaxation is denoted by v_{LP}, and $v_{\mathrm{LP}} \le v^*$. The duality gap, the difference between v^* and v_{LP}, is in most cases larger than zero. The LP-problem is large, but sparse, and can be solved with a standard LP-code.

Convex Piecewise Linearization

Since the binary variables y_{ik} are only included to give the correct cost for the production, they can be eliminated if we use an approximation of the costs. If the staircase cost function is underestimated with a piecewise linear and convex function, we get a problem, much easier to solve, which gives a lower bound on v^*, denoted by v_{CPL}, see [14] and [11]. For explicit expressions of how to construct the convex piecewise linearization see [11].

The resulting problem is a linear minimal cost network flow problem with parallel arcs, which is quite easily solvable by a standard network code. The x- and t-part of the solution is feasible in the original problem, so we can generate an upper bound by evaluating this solution in the correct cost function, which is done by finding the correct values of y.

In [10] it is proved that the convex piecewise linearization and the LP relaxation are equivalent, in the sense that $v_{\mathrm{CPL}} = v_{\mathrm{LP}}$ and an x-solution that is optimal in one of the problems is also optimal in the other problem. Utilizing the network structure, we thus get a quicker way of solving the LP relaxation.

Benders Decomposition

In [11] a Benders decomposition approach is used, and combined with the convex piecewise linearization described above.

The Benders subproblem is simply obtained by fixing the integer variables, i. e. fixing the sizes of the facilities. The resulting problem is minimal cost network flow problem, similar to a transportation problem, but with certain intervals (given by the facility sizes) for the supplies.

However, the Benders master problem obtained by a standard application of the Benders approach, is much too hard to solve. The number of integer variables

is much larger than in an ordinary location problem with the same numbers of facilities and customers. One way around this is to combine the Benders approach with the convex piecewise linearization.

An *improved piecewise linearization* is obtained by branching at certain production levels. A staircase cost function is divided into two parts by the branching, and a binary variable is introduced, indicating which of the parts that is to be used. In each of the two parts, convex piecewise linearization is used. In this manner, one could design a branch and bound method for solving the problem, similar to what is described in [14].

Considering the model after a number of branchings, we have an approximation (a relaxation) of the original problem, with a much smaller number of integer variables. On this problem we then apply Benders decomposition.

In principle one could let each subproblem in the branch and bound method be solved exactly with Benders decomposition, thereby obtaining basically a branch and bound method, which employs Benders decomposition to solve the branch and bound subproblems. This is however very inefficient.

The other extreme is standard application of Benders decomposition to the original problem, in which case the Benders approach employs branch and bound to solve the Benders subproblems. This is also quite inefficient in practice.

A more efficient method is to combine the two approaches, Benders decomposition and branch and bound on a more equal level. This can be done the following way.

1) Solve the initial convex piecewise linearization (with a network code).
2) Do one or more branchings, where the error of the approximation at the obtained solution is largest.
3) Solve the obtained problem with Benders decomposition (to a certain accuracy).
4) Repeat 2) and 3), until optimality.

There are two very important comments to the above algorithm.

A) When one returns to step 3) after having done branchings, one can recalculate and reuse all the Benders cuts obtained before the branchings. (This is described in detail in [11].)
B) The stopping criterion for the Benders method, i. e. the required accuracy in step 3), is a very important control parameter. One should in initial iterations require a low accuracy, and gradually, as the method approaches the optimal solution, require higher and higher accuracies.

The effect of combining comments A) and B) is that one should only do a few Benders iteration in each main iteration, since the number of Benders cuts will automatically increase, as the old cuts are recalculated and kept.

The main conclusion of the computational tests done in [11] is that only a small part of all the integer variables (in average 4%) need to be included by the improving piecewise linearization technique, when solving a problem to reasonable accuracy. In other words, only a small subset of the possible sizes need to be investigated.

Lagrangian Relaxation and Subgradient Optimization

Now we will describe a Lagrangian heuristic, found in [12], in more detail. Lagrangian relaxation and subgradient optimization are used to obtain a near-optimal dual solution, and act as a base for an efficient primal heuristic. Based on the solution of the Lagrangian relaxation one can construct a transportation problem which yields primal feasible solutions, and can be used during the subgradient process.

An important aspect of the Lagrangian approach is that a method yielding good feasible *primal solutions* can be based on *dual techniques*.

Lagrangian relaxation, [7], in combination with subgradient optimization, [9] is a commonly used technique for generating lower bounds on the optimal objective function value of mixed integer programming problems. Here we apply Lagrangian relaxation to constraint set (2), and denote the Lagrangian multipliers by u_i. For fixed values of u, the subproblem separates into several smaller problems:

$$\begin{cases} \theta_{1j}(\overline{u}) & = \min \sum_{i=1}^{m} (c_{ij} + \overline{u}_i) x_{ij} \\ \text{s.t.} & \sum_{i=1}^{m} x_{ij} = D_j, \\ & x_{ij} \geq 0, \end{cases}$$

$$\begin{cases} \theta_{2i}(\overline{u}) & = \min \sum_{k=1}^{q_i}((p_{ik} - \overline{u}_i)t_{ik} + \Delta f_{ik} y_{ik}) \\ \text{s.t.} & t_{ik} \le \Delta S_{ik} y_{ik}, \quad \forall k, \\ & t_{ik-1} \ge \Delta S_{ik-1} y_{ik}, \quad \forall k > 1, \\ & t_{ik} \ge 0, \\ & y_{ik} = 0 \text{ or } 1 \end{cases}$$

The first set of subproblems consists of n continuous knapsack problems, which are trivially solvable. The second set of subproblems consists of m one-dimensional staircase cost problems. The solution can be found by calculating the minimizer $\overline{k}_i$ for each i, as follows:

$$\theta_{2i}(\overline{u}_i) = \min_{k_i=0,\ldots,q_i} \sum_{k=1}^{k_i}((p_{ik} - \overline{u}_i)t_{ik} + \Delta f_{ik} y_{ik}).$$

The resulting solution is

$$y'_{ik} = \begin{cases} 1, & \forall k \le \overline{k}_i, \\ 0, & \forall k > \overline{k}_i, \end{cases}$$

$$t'_{ik} = \begin{cases} \Delta S_{ik}, & \forall k \le \overline{k}_i, \\ 0, & \forall k > \overline{k}_i. \end{cases}$$

Note that the subproblem has the integrality property, [7], so $\max \theta(u) = v_{\text{LP}}$.

The Lagrangian dual,

$$\max \theta(u) = \sum_{j=1}^{n} \theta_{1j}(u) + \sum_{i=1}^{m} \theta_{2i}(u)$$

can be solved by standard subgradient optimization, [9], in order to get the best lower bound. One can use enhancements such as modified directions, [5], $d^r = \xi^r + \alpha\, d^{r-1}$, where ξ^r is the subgradient generated in iteration r and d^r is the direction used in iteration r.

A steplength shortening is obtained by setting $\lambda = \lambda \backslash 2$ when there has not been any improvement of $\underline{v}$ for $\overline{N}_1$ consecutive iterations. When there has not been any improvement of $\underline{v}$ for $\overline{N}_2$ iterations, the subgradient optimization procedure is terminated. The subgradient is given by $\xi_i^r = \sum_{j=1}^{n} x_{ij}' - \sum_{k=1}^{q_i} t_{ik}'$ for all i, where x_{ij}' and t_{ik}' are the optimal solutions to the subproblems. Reasonable choices for the parameters are $\overline{N}_1 = 6$, $\overline{N}_2 = 25$, and $\alpha = 0.7$.

One can use a heuristic based on the solution of the Lagrangian relaxation to try to get a feasible solution. The obtained values of y_{ik}' are used to calculate the supply at each location and a transportation problem is solved. The solution to the transportation problem is feasible in the original problem if constraint sets (3) and (4) are satisfied, which easily can be achieved. The values of the flow variables x_{ij} are taken directly from the solution to the transportation problem. The total production t_i is then calculated as $t_i = \sum_{j=1}^{n} x_{ij}$. One can then easily find t_{ik} as the part of t_i that lies within level k, and the y_{ik} solution is simply $y_{ik} = 1$ if $t_{ik} > 0$ and 0 if not. Finally all unnecessary production capacity at each location i is removed.

The complete Lagrangian heuristic, LH, also includes the following. The convex piecewise linearization, CPL, is solved with an efficient network code. The Lagrangian multipliers are initiated with a convex combination of the appropriate node prices obtained by solving CPL and $\min_j c_{ij}$, with the largest weight on the former. The primal procedure to generate feasible solutions is used every third iteration in the subgradient procedure.

Note that CPL yields $v_{\text{CPL}} = v_{\text{LP}}$, so the subgradient procedure cannot improve the lower bound, which is quite unusual in methods of this kind. The motivation behind using the subgradient procedure is not to get lower bounds, but to get primal solutions (upper bounds).

Computational Results

In [12] the heuristic procedures are tested by solving a number of randomly generated test problems, with up to 50 locations, 100 destinations and 20 sizes of each location (yielding 6000 continuous variables and 1000 integer variables). The conclusions of the tests are the following.

A standard mixed integer programming code (in this case LAMPS) needs extremely long solution times for finding the exact optimum. The ADD heuristics produce solutions with relative errors in the range of 1%–20% (in average 11%), but also requires quite long solution times (although not as long as the MIP-code).

The convex piecewise linearization CPL, combined with exact integer evaluation of the solutions obtained, yields solutions that all are better than those obtained by the ADD heuristics, with relative errors between 0.8% and 10% (in average 4%), in a much shorter time

(in average 1000 times quicker than the ADD heuristics). So CPL dominates the ADD heuristics completely, both with respect to solution time and solution quality.

The Lagrangian heuristic, LH, produces solutions with relative errors between 0.4% and 3.2% (in average 1.5%), with solution times in average 20 times shorter than the ADD heuristics, but of course significantly longer than CPL.

Comparison to other tests is difficult, since other computers and codes are used. The Benders approach in [11] seems to be slower than the Lagrangian approach. However, on modern computers and with modern MIP-codes, its performance may well improve.

Conclusion

The capacitated facility location problem with staircase costs has many important applications. Computational results indicate that it is possible to find near-optimal solutions to such problems of reasonable size in a reasonable time, i. e. that this better model can be used instead of, for example, the ordinary capacitated facility location problem in appropriate situations.

See also

- ▶ Combinatorial Optimization Algorithms in Resource Allocation Problems
- ▶ Competitive Facility Location
- ▶ Facility Location with Externalities
- ▶ Facility Location Problems with Spatial Interaction
- ▶ Global Optimization in Weber's Problem with Attraction and Repulsion
- ▶ MINLP: Application in Facility Location-allocation
- ▶ Multifacility and Restricted Location Problems
- ▶ Network Location: Covering Problems
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Production-distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Stochastic Transportation and Location Problems
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Beasley JE (1993) Lagrangean heuristics for location problems. Europ J Oper Res 65:383–399, Testproblems available at http://mscmga.ms.ic.ac.uk
2. Bornstein CT, Rust R (1988) Minimizing a sum of staircase functions under linear constraints. Optim 19:181–190
3. Christofides N, Beasley JE (1983) Extensions to a Lagrangean relaxation approach for the capacitated warehouse location problem. Europ J Oper Res 12:19–28
4. Cornuejols G, Sridharan R, Thizy JM (1991) A comparison of heuristics and relaxations for the capacitated plant location problem. Europ J Oper Res 50:280–297
5. Crowder H (1976) Computational improvements for subgradient optimization. Symp Math, vol XIX. Acad. Press, pp 357–372
6. Domschke W, Drexl A (1985) ADD-heuristics' starting procedures for capacitated plant location models. Europ J Oper Res 21:47–53
7. Geoffrion AM (1974) Lagrangean relaxation for integer programming. Math Program Stud 2:82–114
8. Geoffrion A, McBride R (1978) Lagrangean relaxation applied to capacitated facility location problems. AIIE Trans 10:40–47
9. Held M, Wolfe P, Crowder HP (1974) Validation of subgradient optimization. Math Program 6:62–88
10. Holmberg K (1991) Linearizations of the staircase cost facility location problem. Res Report Dept Math Linköping Inst Techn, no. LiTH-MAT-R-1991-19
11. Holmberg K (1994) Solving the staircase cost facility location problem with decomposition and piecewise linearization. Europ J Oper Res 75:41–61
12. Holmberg K, Ling J (1997) A Lagrangean heuristic for the facility location problem with staircase costs. Europ J Oper Res 97:63–74
13. Jacobsen SK (1983) Heuristics for the capacitated plant location model. Europ J Oper Res 12:253–261
14. Rech P, Barton LG (1970) A non-convex transportation algorithm. In: Beale EM (ed) Applications of Mathematical Programming Techniques, pp 250–260
15. Sridharan R (1991) A Lagrangean heuristic for the capacitated plant location problem with side constraints. J Oper Res Soc 42:579–585

Farkas Lemma

FI

Kees Roos
Department ITS/TWI/SSOR,
Delft University Technol., Delft, The Netherlands

MSC2000: 15A39, 90C05

Article Outline

Keywords
See also
References

Keywords

Inequality systems; Certificate; Theorem of the alternative; Skew-symmetric matrix; Orthogonal matrix

Farkas' lemma is the most well-known *theorem of the alternative* or *transposition theorem* (cf. ▶ Linear optimization: Theorems of the alternative). Given an $m \times n$ matrix A and a vector b (of dimension m) it states that either the set

$$S := \{y\colon\ y^\top A \geq 0, y^\top b < 0\}$$

or the set

$$T := \{x\colon\ Ax = b,\ x \geq 0\}$$

is empty but not both sets are empty. This result has a long history and it has had a tremendous impact on the development of the duality theory of linear and nonlinear optimization.

J. Farkas (1847–1930) was professor of Theoretical Physics at the Univ. of Kolozsvár in Hungary. His interest in the subject is explained in the first two sentences of his paper [5]:

> The natural and systematic treatment of analytic mechanics has to have as its background the inequality principle of virtual displacements first formulated by Fourier and later by Gauss. The possibility of such a treatment requires, however, some knowledge of homogeneous linear inequalities that may be said to have been entirely missing up to now.

J.B.J. Fourier [7] seems to have been the first who established that a mechanical system has a stable equilibrium state if and only if some homogeneous system of inequalities, like in the definition of the above set S, has no solution. This observation became known as the *mechanical principle of Fourier*. By Farkas'lemma this happens if and only if the set T is nonempty.

It is almost obvious that if the set T is not empty, then the set S will be empty and we have equilibrium. This follows easily by noting that the sets S and T cannot be both nonempty: if $y \in S$ and $x \in T$ then the contradiction

$$y^\top b = y^\top (Ax) = (y^\top A)x \geq 0$$

follows, because $y^\top A \geq 0$ and $x \geq 0$. This shows that the condition 'T is not empty' is certainly a sufficient condition for equilibrium. The hard part is to prove that this is also a necessary condition for equilibrium. The proof has a long history. First, the condition without proof for special cases was given by A. Cournot in 1827 and for the general case by M. Ostrogradsky in 1834. Farkas published his condition first in 1894 and 1895, but the proof contains a gap. A second attempt, in 1896, is also incomplete. The first complete proof was published in Hungarian, in 1898 [3], and in German in 1899 [4]. This proof is included in Farkas' best known paper [5]. For more details and references, see the historical overviews [9] and [10].

Nowadays (1998) many different proofs of Farkas' lemma are known. For quite recent proofs, see, e. g., [1,2,8]. An interesting derivation has been given by A.W. Tucker [11], based on a result that will be referred to as *Tucker's theorem*. (See ▶ Tucker homogeneous systems of linear relations.) The theorem states that for any *skew-symmetric matrix* K (i. e., $K = -K^\top$) there exists a vector x such that

$$Kx \geq 0, \quad x \geq 0, \quad x + Kx > 0.$$

By taking

$$K = \begin{pmatrix} 0 & 0 & A & -b \\ 0 & 0 & -A & b \\ -A^\top & A^\top & 0 & 0 \\ b^\top & -b^\top & 0 & 0 \end{pmatrix},$$

Tucker's theorem implies the existence of nonnegative vectors z_1, z_2 and x and a nonnegative scalar t such that

$$Ax - tb \geq 0, \tag{1}$$

$$-Ax + tb \geq 0, \tag{2}$$

$$\begin{aligned} -A^\top z_1 + A^\top z_2 &\geq 0, \\ b^\top z_1 - b^\top z_2 &\geq 0, \end{aligned} \tag{3}$$

and

$$z_1 + Ax - tb > 0,$$
$$z_2 - Ax + tb > 0,$$
$$x - A^\top z_1 + A^\top z_2 > 0,$$
$$t + b^\top z_1 - b^\top z_2 > 0. \tag{4}$$

If $t = 0$, then, putting $y = z_2 - z_1$, (3) and (4) yield a vector in the set S. If $t > 0$, since the above inequalities are all homogeneous, one may take $t = 1$ and then (1) and (2) give a vector in the set T. This shows that at least one of the two sets S and T is nonempty, proving the hard part of Farkas' lemma.

It is worth mentioning a *result* of C.G. Broyden [1] who showed that Tucker's theorem, and hence also Farkas' lemma, follows from a simple property of orthogonal matrices. The result states that for any orthogonal matrix Q (so $QQ^\top = Q^\top Q = I$) there exists a unique sign matrix D and a positive vector x such that $Qx = Dx$; a *sign matrix* is a diagonal matrix whose diagonal elements are equal to either plus one or minus one.

The key observation here is that if K is a skew-symmetric matrix, then

$$Q = (I + K)^{-1}(I - K)$$

is an orthogonal matrix, where I denotes the identity matrix; Q is known as the *Cayley transform* of K [6]. The proof of this fact is straightforward. First, for each vector x one has

$$x^\top (I + K)x = x^\top x,$$

whence $I + K$ is an invertible matrix. Furthermore, using $K^\top = -K$, one may write

$$\begin{aligned} Q^\top Q &= (I + K)(I - K)^{-1}(I + K)^{-1}(I - K) \\ &= (I + K)(I - K^2)^{-1}(I - K). \end{aligned}$$

Multiplying both sides from the left with $(I - K)$ one gets

$$\begin{aligned} (I - K)QQ^\top &= (I - K^2)(I - K^2)^{-1}(I - K) \\ &= (I - K), \end{aligned}$$

and multiplying both sides with $(I - K)^{-1}$ one finds $QQ^\top = I$, showing that Q is orthogonal indeed.

Therefore, by Broyden's theorem, there exists a sign matrix D and a positive vector z such that

$$(I + K)^{-1}(I - K)z = Dz.$$

This can be rewritten as

$$(I - K)z = (I + K)Dz,$$

whence

$$z - Kz = Dz + KDz,$$

or

$$z - Dz = K(z + Dz).$$

Defining $x = z + Dz$ one has $x \geq 0$, $Kx \geq 0$ and $x + Kx = 2z > 0$, proving Tucker's theorem.

See also

- ▶ Farkas Lemma: Generalizations
- ▶ Linear Optimization: Theorems of the Alternative
- ▶ Linear Programming
- ▶ Motzkin Transposition Theorem
- ▶ Theorems of the Alternative and Optimization
- ▶ Tucker Homogeneous Systems of Linear Relations

References

1. Broyden CG (1998) A simple algebraic proof of Farkas' lemma and related theorems. Optim Methods Softw 8:185–199
2. Dax A (1997) An elementary proof of Farkas' lemma. SIAM Rev 39:503–507
3. Farkas Gy (1898) A Fourier-féle mechanikai elv alkalmazásának algebrai alapja. Math Természettudományi Értesitö 16:361–364
4. Farkas J (1899) Die algebraischen Grundlage der Anwendungen des mechanischen Princips von Fourier. Math Naturwissenschaftl Bericht Ungarn 16:154–157
5. Farkas J (1902) Theorie der Einfachen Ungleichungen. J Reine Angew Math 124:1–27
6. Fekete A (1985) Real linear algebra. M. Dekker, New York
7. Fourier JBJ (1826) Solution d'une question particulière du calcul des inégalités. Nouveau Bull Sci Soc Philomath Paris, pp 99–100
8. Klafsky E, Terlaky T (1987) Remarks on the feasibility problem of oriented matroids. Ann Univ Sci Budapest R Eötvös 7:155–157
9. Prékopa A (1980) On the development of optimization theory. Amer Math Monthly 87:527–542

10. Schrijver A (1986) Theory of linear and integer programming. Wiley, New York
11. Tucker AW (1956) Dual systems of homogeneous linear relations. In: Kuhn HW, Tucker AW (eds) Linear Inequalities and Related Systems. Ann Math Stud., vol 38. Princeton Univ Press, Princeton, pp 3–18

Farkas Lemma: Generalizations

V. Jeyakumar
School of Math., University New South Wales, Sydney, Australia

MSC2000: 46A20, 90C30, 52A01

Article Outline

Keywords
Infinite-Dimensional Optimization
Nonsmooth Optimization
Global Nonlinear Optimization
Nonconvex Optimization
Semidefinite Programming
See also
References

Keywords

Inequality systems; $\in$-subdifferential; D.c. function; Global optimization; Convex inequality systems; Convex-like systems; Nonsmooth optimization; Semidefinite programming; Alternative theorem

The key to identifying optimal solutions of constrained nonlinear optimization problems is the Lagrange multiplier conditions. One of the main approaches to establishing such multiplier conditions for inequality constrained problems is based on the dual solvability characterizations of systems involving inequalities. J. Farkas [7] initially established such a dual characterization for linear inequalities which was used in [23] to derive necessary conditions for optimality for nonlinear programming problems. This dual characterization is popularly known as Farkas' lemma, which states that given any vectors $a_1, \ldots, a_m$ and c in $\mathbf{R}^n$, the linear inequality $c^\top x \geq 0$ is a consequence of the linear system $a_i^\top x \geq 0$, $i = 1, \ldots, m$, if and only if there exist multipliers $\lambda_i \geq 0$ such that $c = \sum_{i=1}^m \lambda_i a_i$. This result can also be expressed as a so-called *alternative theorem*: Exactly one of the following alternatives is true:

i) $\exists x \in \mathbf{R}^n$, $a_i^\top x \geq 0$, $c^\top x < 0$,

ii) $\exists \lambda_i \geq 0$, $c = \sum_{i=1}^m \lambda_i a_i$.

This lemma is the key result underpinning the linear programming duality and has played a central role in the development of nonlinear optimization theory. A large variety of proofs of the lemma can be found in the literature (see [5,25,26]). The proof [3,5] that relies on the separation theorems has led to various extensions. These extensions cover wide range of systems including systems involving infinite-dimensional linear inequalities, *convex inequalities* and matrix inequalities. Applications range from classical nonlinear programming to modern areas of optimization such as nonsmooth optimization and semidefinite programming. Let us now describe certain main generalizations of Farkas' lemma and their applications to problems in various areas of optimization.

Infinite-Dimensional Optimization

The Farkas lemma for a finite system of linear inequalities has been generalized to systems involving arbitrary convex cones and continuous linear mappings between spaces of arbitrary dimensions. In this case the lemma holds under a crucial closure condition. In symbolic terms, the main version of such extension to arbitrary dual pairs of vector spaces states that the following equivalence holds [6]:

$$\left[A(x) \in S \Rightarrow c(x) \geq 0\right] \Leftrightarrow c \in A^\top(S^*), \qquad (1)$$

provided the cone $A^\top(S^*)$ is closed in some appropriate topology. Here A is a continuous linear mapping between two Banach spaces, S is a closed convex cone having the dual cone S^* [5]. The closure condition holds when S is a polyhedral cone in some finite-dimensional space. For simple examples of nonpolyhedral convex cones in finite dimensions where the closure condition does not hold, see [1,5]. However, the following asymptotic version of Farkas' lemma holds without a closure condition:

$$\left[A(x) \in S \Rightarrow c(x) \geq 0\right] \Leftrightarrow c \in \mathrm{cl}(A^\top(S^*)), \qquad (2)$$

where $\mathrm{cl}(A^\top(S^*))$ is the closure of $A^\top(S^*)$ in the appropriate topology. These extensions resulted in the development of asymptotic and nonasymptotic first

order necessary optimality conditions for infinite-dimensional smooth constrained optimization problems involving convex cones and duality theory for infinite-dimensional linear programming problems (see e. g. [12]). Smooth optimization refers to the optimization of a differentiable function. A nonasymptotic form of an extension of Farkas' lemma that is different from the one in (1) is given in [24] without the usual closure condition. For related results see [4]. An approach to the study of semi-infinite programming, which is based on generalized Farkas' lemma for infinite linear inequalities is given in [12].

Nonsmooth Optimization

The success of linear programming duality and the practical nature of the Lagrange multiplier conditions for smooth optimization have led to extensions of Farkas' lemma to systems involving nonlinear functions. Convex analysis allowed to obtain extensions in terms of *subdifferentials* replacing the linear systems by sublinear (convex and positively homogeneous) systems [8,31]. A simple form of such an extension states that the following statements are equivalent:

$$-g(x) \in S \Rightarrow f(x) \geq 0 \tag{3}$$

$$0 \in \mathrm{cl}\left[\partial f(0) + \bigcup_{\lambda \in S^*} \partial(\lambda g)(0)\right], \tag{4}$$

where the real valued function f is sublinear and lower semicontinuous, and the vector function g is sublinear with respect to the cone S and vg is lower semicontinuous for each $v \in S^*$. When f is continuous the statement (4) collapses to the condition

$$0 \in \partial f(0) + \mathrm{cl}\left[\bigcup_{\lambda \in S^*} \partial(\lambda g)(0)\right]. \tag{5}$$

This extension was used to obtain optimality conditions for convex optimization problems and *quasidifferentiable problems* in the sense of B.N. Pshenichnyi [27]. A review of results of Farkas type for systems involving sublinear functions is given in [13,14].

Difference of sublinear (DSL) functions which arise frequently in nonsmooth optimization provide useful approximations for many classes of nonconvex *nonsmooth functions*. This has led to the investigation of results of Farkas type for systems involving DSL functions.

A mapping $g: X \to Y$ is said to be *difference sublinear* (DSL) (with respect to S) if, for each $v \in S^*$, there are (weak $*$) compact convex sets, here denoted $\underline{\partial}(\mathrm{vg})(0)$ and $\overline{\partial}(vg)(0)$, such that, for each $x \in X$,

$$vg(x) = \max_{u \in \underline{\partial}(vg)(0)} u(x) - \max_{w \in \overline{\partial}(vg)(0)} w(x),$$

where X and Y are Banach spaces. If $Y = \mathbf{R}$ and $S = \mathbf{R}_+$ then this definition coincides with the usual notion of a difference sublinear real-valued function. Thus a mapping g is DSL if and only if vg is a DSL function for each $v \in S^*$. The sets $\underline{\partial}(\mathrm{vg})(0)$ and $\overline{\partial}(vg)(0)$ are the *subdifferential* and *superdifferential* of vg, respectively. For a DSL mapping $g: X \to Y$ we shall often require a *selection* from the class of sets $\left\{\overline{\partial}(vg)(0):\ v \in S^*\right\}$. This is a set, denoted (w_v), in which we select a single element $\overline{\partial}(vg)(0)$ for each $v \in S^*$. An extension of the Farkas lemma for DSL systems states that the following statements are equivalent [10,20]:

i) $-g(x) \in S \Rightarrow f(x) \geq 0$;

ii) for each selection (w_v) with $w_v \in \overline{\partial}(vg)(0)$, $v \in S^*$, $\overline{\partial} f(0) \subseteq \underline{\partial} f(0) + B$,

where $B = \mathrm{cl\,cone\,co}\left[\bigcup_{v \in S^*}(\underline{\partial}(vg)(0) - w_v)\right]$. A unified approach to generalizing the Farkas lemma for sublinear systems which uses multivalued functions and convex process is given [2,17,18].

Global Nonlinear Optimization

Given that the optimality of a constrained *global optimization* problem can be viewed as the solvability of appropriate inequality systems, it is easy to see that an extension of Farkas' lemma again provides a mechanism for characterizing *global optimality* of a range of nonlinear optimization problems. The *ϵ-subdifferential* analysis here allowed to obtain a new version of the Farkas lemma replacing the linear inequality $c(x) \geq 0$ by a *reverse convex inequality* $h(x) \leq 0$, where h is a convex function with $h(0) = 0$. This extension for systems involving DSL functions states that the following conditions are equivalent.

i) $-g(x) \in S \Rightarrow h(x) \leq 0$;

ii) for each selection (w_v) with $w_v \in \overline{\partial}(vg)(0)$, $v \in S^*$ and for each $\epsilon \geq 0$,

$$\partial_\epsilon h(0) \subseteq \text{cl cone co}\left[\bigcup_{v\in S^*}(\underline{\partial}(vg)(0) - w_v)\right].$$

Such an extension has led to the development of conditions which characterize optimal solutions of various classes of global optimization problems such as convex maximization problems and fractional programming problems (see [19,20]).

However, simple examples show that the asymptotic forms of the above results of Farkas type do not hold if we replace the DSL (or sublinear) system by a convex system. Ch.-W. Ha [15] established a version of the Farkas lemma for convex systems in terms of *epigraphs* of *conjugate functions*. A simple form of such a result [29] states that the following statements are equivalent:

i) $(\forall i \in I)\, g_i(x) \leq 0 \Rightarrow h(x) \leq 0$;

ii) $\text{epi}\, h^* \subseteq \text{cl cone co}\,[\cup_{i\in I} \text{epi}\, g_i^*]$,

provided the system

$$i \in I, \quad g_i(x) \leq 0$$

has a solution. Here h and, for each $i \in I$, g_i are continuous convex functions, I is an arbitrary index set, and h^* and g_i^* are conjugate functions of h and g_i respectively. This result has also been employed to study infinite-dimensional nonsmooth nonconvex problems [30]. A basic general form of the Farkas lemma for convex system with application to *multiobjective convex optimization* problems is given in [11]. Extensions to systems involving the difference of convex functions are given in [21,29]. A more general result involving *H-convex functions* [29] with application to global nonlinear optimization is given in [29].

Nonconvex Optimization

The convexity requirement of the functions involved in the extended Farkas lemma above can be relaxed to obtain a form of Farkas' lemma for convex-like system. Let $F: X \times Y \to \mathbf{R}$ and let $f: X \to \mathbf{R}$, where X and Y are arbitrary nonempty sets. The pair (f, F) is *convex-like* on X if

$$(\exists \alpha \in (0,1))(\forall x_1, x_2 \in X)(\exists x_3 \in X),$$
$$f(x_3) \leq \alpha f(x_1) + (1-\alpha) f(x_2)$$

and $(\forall y \in Y)$:

$$F(x_3, y) \leq \alpha F(x_1, y) + (1-\alpha) F(x_2, y).$$

If the pair (f, F) is convex-like on X, there is $x_0 \in X$ with $(\forall y \in Y)\, F(x_0, y) \leq 0$ and if a regularity condition holds then the following statements are equivalent [21]:

$$\forall y \in Y,\ F(x, y) \leq 0 \Longrightarrow f(x) \geq 0,$$
$$(\forall \theta < 0)(\exists \lambda \in \Lambda)(\forall x \in X)$$
$$f(x) + \sum_{y\in Y} \lambda_y F(x, y) > \theta,$$

where Λ is the dual cone of the convex cone of all nonnegative functions on Y. An asymptotic version of the above result holds if the regularity hypothesis is not fulfilled. This extension has been applied to develop Lagrange multiplier type results for minimax problems and constrained optimization problems involving convex-like functions. For related results see [16].

Semidefinite Programming

A useful corollary of the Farkas lemma, which is often used to characterize the *feasibility problem* for linear inequalities, states that exactly one of the following alternatives is true:

i) $\exists x \in \mathbf{R}^n\ a_i^\top x \leq b_i,\ i = 1, \ldots, m$,

ii) $\exists \lambda_i \geq 0\ \sum_{i=1}^m \lambda_i a_i = 0,\ \sum_{i=1}^m b_i \lambda_i = -1$.

This form of the Farkas lemma has also attracted various extensions to nonlinear systems, including *sublinear and DSL systems* [20] with the view to characterize the feasibility of such systems. The feasibility problem, which has been of great interest in *semidefinite programming*, is the problem of determining whether there exists an $x \in \mathbf{R}^n$ such that $Q(x) \succeq 0$, for real symmetric matrices Q_i, $i = 0, \ldots, m$, where $\succeq$ denotes the partial order, i. e. $B \succeq A$ if and only if $A - B$ is positive semidefinite, and $Q(x) = Q_0 - \sum_{i=1}^m x_i Q_i$. However, simple examples show that a direct analog of the alternative does not hold for the semidefinite inequality systems $Q(x) \succeq 0$ without additional hypothesis on Q. A modified dual conditions which characterize solvability of the system $Q(x) \succeq 0$ is given in [28].

See also

► Farkas Lemma

References

1. Ben-Israel A (1969) Linear inequalities and inequalities on finite dimensional real or complex vector spaces: A unified theory. J Math Anal Appl 27:367–389
2. Borwein JM (1983) Adjoint process duality. Math Oper Res 8:403–437
3. Borwein JM (1983) A note on the Farkas lemma. Utilitas Math 24:235–241
4. Borwein JM, Wolkowicz H (1982) Characterizations of optimality without constraint qualification for the abstract convex program. Math Program Stud 19:77–100
5. Craven BD (1978) Mathematical programming and control theory. Chapman and Hall, London
6. Craven BD, Koliha JJ (1977) Generalizations of Farkas' theorem. SIAM J Math Anal 8:983–997
7. Farkas J (1901) Theorie der einfachen Ungleichungen. J Reine Angew Math 124:1–27
8. Glover BM (1982) A generalized Farkas lemma with applications to quasidifferentiable programming. Z Oper Res 26:125–141
9. Glover BM, Ishizuka Y, Jeyakumar V, Tuan HD (1996) Complete characterization of global optimality for problems involving the pointwise minimum of sublinear functions. SIAM J Optim 6:362–372
10. Glover BM, Jeyakumar V, Oettli W (1994) Farkas lemma for difference sublinear systems and quasidifferentiable programming. Math Program 63:333–349
11. Glover BM, Jeyakumar V, Rubinov AM (1999) Dual conditions characterizing optimality for convex multi-objective programs. Math Program 84:201–217
12. Goberna MA, Lopez MA, Pastor J (1981) Farkas-Minkowski systems in semi-infinite programming. Appl Math Optim 7:295–308
13. Gwinner J (1987) Corrigendum and addendum to 'Results of Farkas type'. Numer Funct Anal Optim 10:415–418
14. Gwinner J (1987) Results of Farkas type. Numer Funct Anal Optim 9:471–520
15. Ha Ch-W (1979) On systems of convex inequalities. J Math Anal Appl 68:25–34
16. Ills T, Kassay G (1994) Farkas type theorems for generalized convexities. Pure Math Appl 5:225–239
17. Jeyakumar V (1987) A general Farkas lemma and characterization of optimality for a nonsmooth program involving convex processes. J Optim Th Appl 55:449–461
18. Jeyakumar V (1990) Duality and infinite dimensional optimization. Nonlinear Anal Th Methods Appl 15:1111–1122
19. Jeyakumar V, Glover BM (1993) A new version of Farkas' lemma and global convex maximization. Appl Math Lett 6(5):39–43
20. Jeyakumar V, Glover BM (1995) Nonlinear extensions of Farkas' lemma with applications to global optimization and least squares. Math Oper Res 20:818–837
21. Jeyakumar V, Gwinner J (1991) Inequality systems and optimization. J Math Anal Appl 159:51–71
22. Jeyakumar V, Rubinov AM, Glover BM, Ishizuka Y (1996) Inequality systems and global optimization. J Math Anal Appl 202:900–919
23. Kuhn HW, Tucker AW Nonlinear programming, Proc. Second Berkeley Symp. Math. Statist. and Probab., Univ. Calif. Press, Berkeley, CA, pp 481–492
24. Lasserre JB (1997) A Farkas lemma without a standard closure condition. SIAM J Control Optim 35:265–272
25. Mangasarian OL (1969) Nonlinear programming. McGraw-Hill, New York
26. Prékopa A (1980) On the development of optimization theory. Amer Math Monthly 87:527–542
27. Pshenichnyi BN (1971) Necessary conditions for an extremum. M. Dekker, New York
28. Ramana MV (1977) An exact duality theory for semidefinite programming and its complexity implications. Math Program 77:129–162
29. Rubinov AM, Glover BM, Jeyakumar V (1995) A general approach to dual characterizations of solvability of inequality systems with applications. J Convex Anal 2(2):309–344
30. Schirotzek W (1985) On a theorem of Ky Fan and its application to nondifferentiable optimization. Optim 16:353–366
31. Zalinescu C (1978) A generalization of the Farkas lemma applications to convex programming. J Math Anal Appl 66:651–678

Feasible Sequential Quadratic Programming

FSQP

André L. Tits
University Maryland, College Park, USA

MSC2000: 65K05, 65K10, 90C06, 90C30, 90C34

Article Outline

Keywords
Main Ideas
Algorithms
Applications
See also
References

Keywords

Nonlinear programming; Sequential quadratic programming; Successive quadratic programming; Feasible iterates

Feasible sequential quadratic programming (FSQP) refers to a class of *sequential quadratic programming* (SQP) methods that have the additional property that all iterates they construct satisfy the inequality constraints. Thus, for the problem

$$\begin{cases} \min\limits_{x\in\mathbf{R}^n} & f(x) \\ \text{s.t.} & g_j(x) \le 0, \quad j = 1,\ldots,m_i, \\ & h_j(x) = 0, \quad j = 1,\ldots,m_e, \end{cases} \tag{1}$$

where f, the g_js, and the h_js are smooth, FSQP methods generate a sequence $\{x_k\}$ such that $g_j(x_k) \le 0$ for all j and all k.

From the application's point of view, enforcing *feasibility of the iterates* with respect to inequality constraints is often an important attribute. First, it may be the case that the objective function is simply not defined when certain constraints are violated, for example, with problems involving dynamical systems, in which stability is needed in order for, say, certain steady state errors to be well defined. Second, it may be crucial that a (suboptimal) solution satisfying certain 'hard' constraints be available after a prescribed amount of time has elapsed, too short to allow convergence to the optimal solution. This is the case, for instance in certain real-time control applications. A third situation where feasibility of successive iterates is desirable is in the context of trade-off exploration for design problems. Indeed, trade-offs between 'soft' design specifications cannot be meaningfully explored unless 'hard' specifications are satisfied. From the point of view of numerical algorithms, while maintaining feasibility of successive iterates obviously requires special attention, it also has important beneficial side effects. Namely,

i) the objective function can be forced to decrease at each iteration, and thus can serve as *merit function* in the line search, thereby avoiding the complex issue of choice of an appropriate surrogate merit function; and
ii) as pointed out below, in the context of SQP type methods, the quadratic programs successively constructed all have a nonempty feasible set, which is not the case in general for 'infeasible' methods.

Methods that generate feasible iterates have regained much popularity in recent years with the in-depth investigation of barrier-based *interior point methods*, successively in the context of linear, convex-quadratic, general convex, and nonconvex problems, the class of problems of interest here. Contributions to the latter can be found in the classical book [4] as well as, e. g., in [11] (see [14] for a 'modern' presentation) and [3], and in many recent reports. In those methods, each search direction is typically obtained via the solution of a linear system of equations. FSQP algorithms, on the other hand, being of the SQP type, involve the solution of quadratic programs as subproblems. While they are often impractical for problems with large numbers of variables, SQP-type algorithms are particularly suited to various classes of engineering applications where the number of variables is not too large but evaluations of objective/constraint functions and of their gradients are highly time consuming. Indeed, because these algorithms use quadratic programs as successive models, progress between (expensive) function evaluations is typically significantly better than with algorithms making use of mere linear systems of equations as models.

FSQP algorithms are of the *feasible direction* type in that, while they allow iterates to lie on constraint boundaries, small enough displacements along the search directions they generate always yield feasible points. Indeed, whenever the current iterate lies on or near a nonlinear constraint boundary, the search direction tends to point toward the interior of the feasible set. In that respect FSQP algorithms are analogous to interior point methods. Early feasible direction algorithms (see, e. g., [12,16]) were first order methods, i. e., only first derivatives were used and no attempt was made to accumulate and use second order information. As a consequence, such algorithms converged linearly at best. E. Polak proposed several extensions to these algorithms which take second order information into account when computing the search direction (see [12, Sect. 4.4]). Some of the search directions proposed by Polak can be viewed as modified SQP directions but the fast local convergence rate usually associated with SQP is not preserved. In [1], a feasible SQP algorithm is proposed with emphasis on avoiding costly line searches by making it likely that a full step along the constructed direction is acceptable as a next iterate, even early on in the optimization process. The price paid however is again the loss of fast local convergence.

In this article, we focus on feasible SQP methods which, under appropriate assumptions, preserve the fast rate of convergence of standard SQP methods. Such

methods have been considered early on by J.N. Herskovits and L.A.V. Carvalho [5] and in [2,6,8,9,10], and recently also by L. Qi and Z. Wei [13].

Main Ideas

For simplicity, consider the case where only inequality constraints are present, i. e., $m_e = 0$. Suppose that the current estimate x_k for the solution of (1) is feasible, i. e., $g_j(x_k) \leq 0$ for all j. The basic SQP direction, d_k^0, is obtained by solving the quadratic programming problem

$$\begin{cases} \min\limits_{d^0} & \frac{1}{2}\left\langle d^0, H_k d^0\right\rangle + \left\langle \nabla f(x_k), d^0\right\rangle \\ \text{s.t.} & g_j(x_k) + \left\langle \nabla g_j(x_k), d^0\right\rangle \leq 0, \quad \forall j, \end{cases} \tag{2}$$

where H_k is the Hessian of the Lagrangian, or an estimate thereof. While, in general, QP (2) may be inconsistent, feasibility of x_k, which we seek to enforce, guarantees that it admits a feasible point. Indeed, in particular, $d_k^0 = 0$ is always feasible. Assume that H_k is symmetric positive definite. Then QP (2) has a unique solution d_k^0. It is a simple exercise to show that, in addition, d_k^0 has the interesting property of being a first order descent direction for f at x_k, i. e., $\langle \nabla f(x_k), d_k^0\rangle < 0$.

Suppose now that some constraint, say g_{j_0}, is active at x_k, i. e., $g_{j_0}(x_k) = 0$. Then, if the j_0th constraint is also active in QP (2), then $\langle \nabla g_{j_0}(x_k), d_k^0\rangle = 0$, so that d_k^0 is tangent to the feasible set. Quite possibly, as a result, $g_{j_0}(x_k + td_k^0)$ may be positive for small t, making it difficult, or impossible, to locate a next feasible iterate in direction d_k^0. Thus d_k^0 is not an appropriate search direction for FSQP. However any, however small, amount of tilting of d_k^0 towards the interior of the feasible set makes it a feasible direction. The challenge in FSQP type methods is to tilt d_k^0 enough that a sizable step can be made within the feasible set, but little enough that the fast local convergence properties of sequential quadratic programming are preserved.

With appropriate titling of the basic SQP search direction, and an appropriate line search along the resulting direction d (yielding a next iterate $x_k + t_k d_k$ for some $t_k \in (0, 1]$) a globally convergent feasible SQP algorithm can be constructed. However, the result would be unsatisfactory if the algorithm thus obtained did not exhibit a fast local convergence rate, a property that is generally expected from SQP-type methods. For such rate (in particular, a superlinear rate) to take place, it is critical that a full step of one be eventually taken, i. e., that, when x_k is close to the solution, t_k be equal to one. Here a difficulty already arises in the context of classical (nonfeasible) SQP methods, where it may happen that the line search rule prevents the full step from being taken. This possible conflict between global convergence and fast local convergence is known as the *Maratos effect*. In the context of FSQP methods, this difficulty is compounded by the fact that, in order to be acceptable, in addition to satisfy an appropriate descent criterion, the next iterate must be feasible. This imposes further demands on the Maratos-effect avoidance scheme. Two schemes have been proposed in the literature: second order correction with arc search, and nonmonotone line search.

Algorithms

Following is a simple example of an FSQP algorithm, taken from [10].

Parameters: $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$.

Data: $x_0 \in X$, $H_0 = H_0^\top > 0$.
Step 0. Initialization: Set $k = 0$.
Step 1. Computation of a search arc.
  Compute d_k^0. If $d_k^0 = 0$, stop.
Compute d_k^1 and ρ_k and set
  $d_k = (1 - \rho_k)d_k^0 + \rho_k d_k^1$.
  Compute correction $\tilde{d}_k$.
Step 2. Arc search.
  Compute t_k, the first number t in the sequence $\{1, \beta, \beta^2, \ldots\}$ satisfying $f(x_k + td_k + t^2\tilde{d}_k) \leq f(x_k) + \alpha t\langle \nabla f(x_k), d_k\rangle$, $g_j(x_k + td_k + t^2\tilde{d}_k) \leq 0$, $j = 1, \ldots, m_i$.
Step 3. Updates.
  Compute $H_{k+1} = H_{k+1}^\top > 0$.
  Set $x_{k+1} = x_k + t_k d_k + t_k^2\tilde{d}_k$.
  Set $k = k + 1$.
  Go back to Step 1.

Algorithm: Simple FSQP

Here, d_k^1 is a feasible direction and a direction of first order descent for f, $\rho \in (0, 1]$ goes to zero fast enough (like $\| d_k^0 \|^2$) when d_k^0 goes to zero, and $\widetilde{d}_k$ is a correction that aims at insuring that the full step of one will be accepted when x_k is close enough to a solution; compu-

tation of $\widetilde{d}_k$ involves constraint values at $x_k + d_k$. Under standard assumptions this algorithm is known to generate sequences whose limit points are Karush–Kuhn–Tucker points. Under strengthened assumptions, including the assumption that H_k is updated in such a way that it approximates well, in a certain sense, the Hessian of the Lagrangian as a solution is approached, convergence can be shown to be *Q-superlinear* or *2-step superlinear*. See [10] for details. A refined version of the algorithm of [10] is implemented in the CFSQP/FFSQP software (see [15]). Refinements include the capability to handle equality constraints [6], *minimax* and *constrained minimax problems* and to efficiently handle problems with large numbers of inequality constraints and minimax problems with large numbers of objective functions [8]. Also note that an FSQP method with drastically reduced amount of work per iteration has been recently proposed [7].

Applications

Applications abound where FSQP-type algorithms are of special interest. In particular, as stressed above, such algorithms are particularly appropriate for problems where the number of variables is not too large but functions evaluations are expensive, and feasibility of iterates is desirable (or imperative). Furthermore, problems with a large number of inequality constraints (or minimax problems with large numbers of objective functions), such as finely discretized *semi-infinite optimization* problems, can be handled effectively, making FSQP especially well-suited for problems involving, e. g., time or frequency responses of dynamical systems. Pointers to a large number of applications can be found on the web, at the URL listed above. Application areas include all branches of engineering, medicine, physics, astronomy, economics and finances, to mention but a few.

See also

- ▶ Optimization with Equilibrium Constraints: A Piecewise SQP Approach
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Successive Quadratic Programming
- ▶ Successive Quadratic Programming: Applications in Distillation Systems
- ▶ Successive Quadratic Programming: Applications in the Process Industry
- ▶ Successive Quadratic Programming: Decomposition Methods
- ▶ Successive Quadratic Programming: Full Space Methods
- ▶ Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Birge J, Qi L, Wei Z (2000) A variant of the Topkis–Veinott method for solving inequality constrained optimization problems. J Appl Math Optim 41:309–330
2. Bonnans JF, Panier ER, Tits AL, Zhou JL (Aug. 1992) Avoiding the Maratos effect by means of a nonmonotone line search II. Inequality constrained problems – feasible iterates. SIAM J Numer Anal 29(4):1187–1202
3. El-Bakry AS, Tapia RA, Tsuchiya T, Zhang Y (1996) On the formulation and theory of the Newton interior-point method for nonlinear programming. J Optim Th Appl 89:507–541
4. Fiacco AV, McCormick GP (1968) Nonlinear programming: Sequential unconstrained minimization techniques. Wiley New York, New York
5. Herskovits JN, Carvalho LAV (1986) A successive quadratic programming based feasible directions algorithm. In: Bensoussan A, Lions JL (eds) Proc. Seventh Internat. Conf. Analysis and Optimization of Systems – Antibes, June 25-27, 1986, Lecture Notes Control Inform Sci. Springer, Berlin, pp 93–101
6. Lawrence CT, Tits AL (1996) Nonlinear equality constraints in feasible sequential quadratic programming. Optim Methods Softw 6:265–282
7. Lawrence CT, Tits AL (1998) A computationally efficient feasible sequential quadratic programming algorithm. Techn Report Inst Systems Res, Univ Maryland, no. TR 98-46
8. Lawrence CT, Tits AL (1998) Feasible sequential quadratic programming for finely discretized problems from SIP. In: Reemtsen R, Rückmann J-J (eds) Semi-infinite programming. Nonconvex Optim Appl. Kluwer, Dordrecht, pp 159–193
9. Panier ER, Tits AL (1987) A superlinearly convergent feasible method for the solution of inequality constrained optimization problems. SIAM J Control Optim 25(4):934–950
10. Panier ER, Tits AL (1993) On combining feasibility, descent and superlinear convergence in inequality constrained optimization. Math Program 59:261–276
11. Panier ER, Tits AL, Herskovits JN (July 1988) A QP-free, globally convergent, locally superlinearly convergent algorithm for inequality constrained optimization. SIAM J Control Optim 26(4):788–811

12. Polak E (1971) Computational methods in optimization. Acad. Press, New York
13. Qi L, Wei Z (2000) On the constant positive linear independence condition and its application to SQP methods. SIAM J Optim 10:963–981
14. Urban T, Tits AL, Lawrence CT (1998) A primal-dual interior-point method for nonconvex optimization with multiple logarithmic barrier parameters and with strong convergence properties. Techn Report Inst Systems Res Univ Maryland no. TR 98-27
15. WEB: 'www.isr.umd.edu/Labs/CACSE/FSQP/fsqp.html'.
16. Zoutendijk G (1960) Methods of feasible directions. Elsevier, Amsterdam

Feedback Set Problems

FSP

PAOLA FESTA[1], PANOS M. PARDALOS[2], MAURICIO G.C. RESENDE[3]
[1] Dip. Mat. e Inform., University Salerno, Salerno, Italy
[2] Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA
[3] Information Sci. Res., AT&T Labs Res., Florham Park, USA

MSC2000: 90C35

Article Outline

Keywords
Notation and Graph Representation
The Feedback Vertex Set Problem
  Mathematical Model of the Feedback Vertex Set Problem
  Polynomially Solvable Cases
  Approximation Algorithms and Provable Bounds on Undirected Graphs
  Approximation Algorithms and Provable Bounds on Directed Graphs
  Exact Algorithms
The Feedback Arc Set Problem
  Mathematical Model of the Feedback Arc Set Problem
  State of the Art of Feedback Arc Set Problems
A GRASP for Feedback Set Problems
Future Research
Conclusions
See also
References

Keywords

Combinatorial optimization; Feedback set problem; Graph bipartization; Local search; GRASP; FORTRAN subroutines

In recent years (1990) feedback set problems have been the subject of growing interest. They have found applications in many fields, including deadlock prevention [90], program verification [79], and Bayesian inference [2]. Therefore, it is natural that in the past few years there have been intensive efforts on exact and approximation algorithms for these kinds of problems. Exact algorithms have been proposed for solving the problems restricted to special classes of graphs as well as several approximation algorithms with provable bounds for the cases that are not known to be polynomially solvable. The most general feedback set problem consists in finding a minimum-weight (or minimum cardinality) set of vertices (arcs) that meets all cycles in a collection C of cycles in a graph (G, w), where w is a nonnegative function defined on the set of vertices $V(G)$ (on the set of edges $E(G)$). This kind of problem is also known as the *hitting cycle problem*, since one must hit every cycle in C. It generalizes a number of problems, including the *minimum feedback vertex (arc) set problem* in both directed and undirected graphs, the *subset minimum feedback vertex (arc) set problem* and the *graph bipartization problem*, in which one must remove a minimum-weight set of vertices so that the remaining graph is bipartite. In fact, if C is the set of all cycles in G, then the hitting cycle problem is equivalent to the problem of finding the minimum feedback vertex (arc) set in a graph. If we are given a set of special vertices and C is the set of all cycles of an undirected graph G that contains some special vertex, then we have the *subset feedback vertex (arc) set problem* and, finally, if C contains all odd cycles of G, then we have the *graph bipartization problem*. All these problems are also special cases of *vertex (arc) deletion problems*, where one seeks a minimum-weight (or minimum cardinality) set of vertices (arcs) whose deletion gives a graph satisfying a given property. There are different versions of *feedback set problems*, depending on whether the graph is directed or undirected and/or the vertices (arcs) are weighted or unweighted. See [30] for a complete survey, and [91] for a general *NP*-hardness proof for almost all vertex and arc deletion problems restricted to

planar graphs. These results apply to the planar bipartization problem, the planar (directed, undirected, or subset) feedback vertex set problems, already proved to be *NP*-hard [33,46]. Furthermore, it is *NP*-complete for planar graphs with no indegree or outdegree exceeding three [46], general graphs with no indegree or outdegree exceeding two [46], and edge-directed graphs [46].

The scope of this article is to give a complete state-of-art survey of exact and approximation algorithms and to analyze a new practical heuristic method called GRASP for solving both feedback vertex and feedback arc set problems.

Notation and Graph Representation

Throughout this paper, we use the following notation and definitions.

A *graph* $G = (V, E)$ consists of a finite set of vertices $V(G)$, and a set of arcs $E(G) \subseteq V(G) \times V(G)$.

An arc (or edge) $e = (v_1, v_2)$ of a directed graph (digraph) $G = (V, E)$ is an incoming arc to v_2 and an outgoing arc from v_1 and it is incident to both v_1 and v_2. If G is undirected, then e is said to be only incident to v_1 and v_2.

For each vertex $i \in V(G)$, let in(i) and out(i) denote the set of incoming and outgoing edges of i, respectively. They are defined only in case of a digraph G. If G is undirected, we will take into account only the degree $\Delta_G(i)$ of i as the number of edges that are incident to i in G.

$\Delta(G)$ denotes the maximum degree among all vertices of a graph G and it is called the *graph degree*.

A vertex $v \in G$ is called an *endpoint* if it has degree one, a *linkpoint* if it has degree two, while a vertex having degree higher than two is called a *branchpoint*.

A *path* P in G connecting vertex u to vertex v is a sequence of arcs $e_1, \ldots, e_r$ in $E(G)$, such that $e_i = (v_i, v_{i+1})$, $i = 1, \ldots, r$, with $v_1 = u$ and $v_{r+1} = v$. A *cycle* C in G is a path $C = (v_1, \ldots, v_r)$, with $v_1 = v_r$.

A *subgraph* $G' = (V', E')$ of $G = (V, E)$ induced by V' is a graph such that $E' = E \cap (V' \times V')$. A graph G is said to be a *singleton*, if $|V(G)| = 1$. Any graph G can be partitioned into isolated connected components $G_1, \ldots, G_k$ and the partition is unique. Similarly, every feedback vertex set V' of G can be partitioned into feedback vertex sets $F_1, \ldots, F_k$ such that F_i is a feedback vertex set of G_i. Therefore, following the additive property and denoting by $\mu(G, w)$ the weight of a minimum feedback vertex (arc) set for (G, w), we have:

$$\mu(G, w) = \sum_{i=1}^{k} \mu(G_i, w).$$

The Feedback Vertex Set Problem

Formally, the feedback vertex set problem can be described as follows. Let $G = (V, E)$ be a graph and let w: $V(G) \to \mathbf{R}^+$ be a weight function defined on the vertices of G. A *feedback vertex set* of G is a subset of vertices $V' \subseteq V(G)$ such that each cycle in G contains at least one vertex in V'. In other words, a feedback vertex set V' is a set of vertices of G such that by removing V' from G along with all the edges incident to V', results in a forest. The weight of a feedback vertex set is the sum of the weights of its vertices, and a *minimum feedback vertex set* of a *weighted graph* (G, w) is a feedback vertex set of G of minimum weight. The weight of a minimum feedback vertex set will be denoted by $\mu(G, w)$. The *minimum weighted feedback vertex set problem* (MWFVS) is to find a minimum feedback vertex set of a given weighted graph (G, w). The special case of identical weights is called the *unweighted feedback vertex set problem* (UFVS).

Mathematical Model of the Feedback Vertex Set Problem

As a covering-type problem, the feedback vertex set problem admits an integer zero-one programming formulation. Given a feedback vertex set V' for a graph (G, w), $G = (V, E)$, and a set of weights $w = \{w(v)\}_{v \in V(G)}$, let $x = \{x_v\}_{v \in V(G)}$ be a binary vector such that $x_v = 1$ if $v \in V'$, and $x_v = 0$ otherwise. Let C be the set of cycles in (G, w). The problem of finding the minimum feedback vertex set of G can be formulated as an integer programming problem as follows:

$$\begin{cases} \min & \displaystyle\sum_{v \in V(G)} w(v) x_v \\ \text{s.t.} & \displaystyle\sum_{v \in V(\Gamma)} x_v \geq 1, \quad \forall \Gamma \in C, \\ & 0 \leq x_v \leq 1 \text{ integer}, \quad v \in V(G). \end{cases}$$

If one denotes by C_v the set of cycles passing through vertex $v \in V(G)$, then the dual of the corresponding lin-

ear programming relaxation is a *packing problem*:

$$\begin{cases} \max & \sum_{\Gamma \in C} y_\Gamma \\ \text{s.t.} & \sum_{\Gamma \in C_v} y_\Gamma \leq w(v), \quad \forall v \in V(G), \\ & y_\Gamma \geq 0, \quad \forall \Gamma \in C. \end{cases}$$

Polynomially Solvable Cases

Given the *NP*-completeness of the feedback vertex set problem, a recent line of research has focused on identifying the largest class of specially structured graphs on which such problems remain polynomially solvable. A pioneering work is due to A. Shamir [79], who proposed a linear time algorithm to find a feedback vertex set for a reducible flow graph. C. Wang, E. Lloyd, and M. Soffa [90] developed an $O(|E(G)|\cdot|V(G)|^2)$ algorithm for finding a feedback vertex set in the class of graphs known as *cyclically reducible graphs*, which is shown to be unrelated to the class of quasireducible graphs. Although the exact algorithm proposed by G.W. Smith and R.B. Walford [83] has exponential running time in general, it returns an optimal solution in polynomial time for certain types of graphs. A variant of the algorithm, called the *Smith–Walford-one algorithm*, selects only candidate sets F of size one and runs in $O(|E(G)|\cdot|V(G)|^2)$ time. The class of graphs for which it finds a feedback vertex set is called *Smith–Walford one-reducible*. In the study of feedback vertex set problems a set of operations called *contraction operations* has had significant impact. They contract the graph $G(V, E)$, while preserving all the important properties relevant to the minimum feedback vertex set. See [56] for a detailed analysis of these reduction procedures which are important for the following two reasons. First, a class of graphs of increasing size is computed, where the feedback vertex set of each graph can be found exactly. Second, most proposed heuristics and approximation algorithms use the reduction schemes in order to reduce the size of the problem. Another line of research on polynomially solvable cases focuses on other special classes, including *chordal* and *interval graphs*, *permutation graphs*, *convex bipartite graphs*, *cocomparability graphs* and on *meshes* and *toroidal meshes*, *butterflies*, and *toroidal butterflies*. The feedback vertex set on chordal and interval graphs can be viewed as a special instance of the generalized clique cover problem, which is solved in polynomial time on chordal graphs [20,93] and interval graphs [65]. For permutation graphs, an algorithm due to A. Brandstädt and D. Kratsch [8] was improved by Brandstädt [7] to run in $O(|V(G)|^6)$ time. More recently (1994), Y.D. Liang [58] presented an $O(|V(G)|\cdot|E(G)|)$ algorithm for permutation graphs that can be easily extended to *trapezoid graphs* while keeping the same time complexity. On interval graphs, C.L. Lu and C.Y. Tang [61] developed a linear-time algorithm to solve the minimum weighted feedback vertex set problem using dynamic programming. S.R. Coorg and C.P. Rangan [19] present an $O(|V(G)|^4)$ time and $O(|V(G)|^4)$ space exact algorithm for cocomparability graphs, which are a superclass of permutation graphs. More recently, Liang and M.S. Chang [13] developed a polynomial time algorithm, that by exploring the structural properties of a cocomparability graph uses dynamic programming to get a minimum feedback vertex set in $O(|V(G)^2|\ |E(G)|)$ time. A recent (1998) line of research [63] on polynomially solvable cases focuses on special undirected graphs having bounded degree and that are widely used as connection networks, namely mesh, butterfly and *k-dimensional cube connected cycle* (CCC_k).

Approximation Algorithms and Provable Bounds on Undirected Graphs

A $2\log_2|V(G)|$-approximation algorithm for the unweighted minimum feedback vertex set problem on undirected graphs is contained in a lemma due to P. Erdös and L. Posa [25]. This result was improved in [66] to obtain a performance ratio of $O(\sqrt{\log|V(G)|})$. R. Bar-Yeruda, D. Geiger, J. Naor, and R.M. Roth [2] gave an approximation algorithm for the unweighted undirected case having ratio less than or equal to 4 and two approximation algorithms for the weighted undirected case having ratios $4\log_2|V(G)|$ and $2\Delta^2(G)$, respectively. To speedup the algorithm, they show how to preprocess the input valid graph by applying the corresponding undirected versions of the Levy–Lowe reduction transformations. For the feedback vertex set problem in general undirected graphs, two slightly different 2-approximation algorithms are described in [3] and [1]. These algorithms improve the approximation algorithms of [2]. They also can find a loop cutset which, under specific conditions, is guaranteed in the

worst case to contain less than four times the number of variables contained in a minimum loop cutset. Subsequently, A. Becker and Geiger [4] applied the same reduction procedure from the loop cutset problem to the minimum weighted feedback vertex set problem of [2], but their result is independent of any condition and is guaranteed in the worst case to contain less than twice the number of variables contained in a minimum loop cutset. They [4] propose two greedy approximation algorithms for finding the minimum feedback vertex set V' in a vertex-weighted undirected graph (G, w), one of them having performance ratio bounded by the constant 2 and complexity $O(m+n \log n)$, where $m = |E(G)|$ and $n = |V(G)|$. In [17], F.A. Chudak, M.X. Goemans, D. Hochbaum, and D.P. Williamson showed how the algorithms due to Becker and Geiger [3] and V. Bafna, P. Berman, and T. Fujito [1] can be explained in terms of the primal-dual method for approximation algorithms that are used to obtain approximation algorithms for network design problems. The primal-dual method starts with an integer programming formulation of the problem under consideration. It then simultaneously builds a feasible integral solution and a feasible solution to the dual of the linear programming relaxation. If it can be shown that the value of these two solutions is within a factor of α, then an α-approximation algorithm is found. The *integrality gap* of an integer program is the worst-case ratio between the optimum value of the integer program and the optimum value of its linear relaxation. Therefore, by applying the primal-dual method it is possible to proof that the integrality gap of the integer program under consideration is bounded. In fact, Chudak et al., after giving a new integer programming formulation of the feedback vertex set problem, provided a proof that its integrality gap is at most 2. They also gave the proofs of some key inequalities needed to prove the correctness of their new integer programming formulation.

Theorem 1 *Let V' denote any feedback vertex set of a graph $G = (V, E)$, $E \neq \emptyset$, let τ denote the cardinality of the smallest feedback vertex set for G, and let E(S) denote the subset of edges that have both endpoints in $S \subseteq V(G)$, $b(S) = |E(S)| - |S|+1$. Then*

$$\sum_{v \in V'} [\Delta_G(v) - 1] \geq b(V(G)), \tag{1}$$

$$\sum_{v \in V'} \Delta_G(v) \geq b(V(G)) + \tau. \tag{2}$$

If every vertex in G has degree at least two, and V'_M is any minimal feedback vertex set (i. e. $\forall\ v \in V'_M$, $V'_M \setminus \{v\}$ is not a feedback vertex set), then

$$\sum_{v \in V'_M} \Delta_G(v) \leq 2(b(V(G)) + \tau) - 2. \tag{3}$$

G. Even, Naor, B. Schieber, and L. Zosin [28] showed that the integrality gap of that integer program for the standard cycle formulation of the feedback vertex set problem is $\Omega(\log n)$. The new integer programming formulation given in [17] is as follows:

$$\begin{cases} \min & \displaystyle\sum_{v \in V(G)} w(v) x_v \\ s.t. & \displaystyle\sum_{v \in S} (\Delta_S(v) - 1) x_v \geq b(S), \\ & S \subseteq V(G) :\ E(S) \neq \emptyset, \\ & x_v \in \{0, 1\}, \quad v \in V(G). \end{cases}$$

The linear programming relaxation is:

$$\begin{cases} \min & \displaystyle\sum_{v \in V(G)} w(v) x_v \\ s.t. & \displaystyle\sum_{v \in S} (\Delta_S(v) - 1) x_v \geq b(S), \\ & S \subseteq V(G) :\ E(S) \neq \emptyset, \\ & x_v \geq 0, \quad v \in V(, G), \end{cases}$$

and its dual is:

$$\begin{cases} \max & \displaystyle\sum_{S} b(S) y_S \\ s.t. & \displaystyle\sum_{S: v \in S} (\Delta_S(v) - 1) y_S \leq w_v, \quad v \in V(G), \\ & y_S \geq 0, \quad S \subseteq V(G) :\ E(S) \neq \emptyset. \end{cases}$$

For the subset feedback vertex problem, the authors of [28] showed that it can be approximated in polynomial time by a factor of $\min\{2\ \Delta(G), 8 \log(|V'|+1), O(\log \tau^*)\}$, where τ^* denotes the value of the optimal fractional solution. In [28] the authors also proposed a technique, called bootstrapping, that enhances the $O(\log |V'|)$ to a factor of $O(\log \tau^*/\beta)$, where β denotes the minimum weight of a vertex. The bootstrapping technique iteratively uses a graph partition algorithm. The output of each iteration is by itself a subset

feedback vertex set and is used as part of the input of the next iteration. After $O(\log |V'|)$ iterations the algorithm gives as output a subset feedback vertex set having weight at most $O(\tau^* \log \tau^*)$. Even, Naor and Zosin [26] improved this result proposing an 8-approximation algorithm. The main tool that they used in developing their approximation algorithm and its analysis is a new version of multicommodity flow, called relaxed multicommodity flow, a hybrid of multicommodity flow and multiterminal flow, in which there are additional constraints, called intercommodity constraints. For each arc, the authors considered the maximum flow among all the commodities, which is shipped along it. They required that for each vertex $v \in V(G)$ the sum of the maximum flows shipped along its incident arcs be bounded by four times the capacity of v. By considering the multicommodity flow, the vertices for which the intercommodity constraints are tight play an important role from the point of view of the connectivity of the graph. They are called intersatured vertices. The main result of [26] is a theorem that bounds the weight of the vertices that must be intersatured, so as to satisfy a given demand vector by the sum of demands.

Approximation Algorithms and Provable Bounds on Directed Graphs

In general, problems on undirected graphs are relatively easier to handle than problems on directed graphs, since more graph theory can be utilized. Not surprisingly, the approximation results obtained so far for the undirected version are stronger than those for the directed version. In fact, none of the algorithms referred to in the previous subsections apply to the feedback vertex set problem in directed graphs and, in contrast with the undirected version, no analytical results are known for the directed case. A very recent direction of research on approximation algorithms in the directed version focuses on the complete equivalence among all feedback set (and/or feedback subset) problems and among these and the directed minimum capacity multicut problem in circular networks. An exhaustive description of the procedures that reduce any feedback set problem to any other or any of them to the directed minimum capacity multicut problem and vice versa are formalized and used in [27] to obtain an approximation algorithm for the subset feedback arc set problem of a weighted directed graph $G = (V, E)$, where the interesting cycles to be hit are contained in a set of special vertices $X \subseteq V(G)$, where $|X| = k$. The weight of the feedback arc set found by their approximation algorithm is $O(\tau^* \log^2 |X|)$, where τ^* is the weight of an optimal fractional feedback set. Nevertheless, their approach can be used to solve any other feedback set problem as well as the directed minimum capacity multicut problem. Even et al. [27] also proposed an algorithm for approximating the minimum weighted subset vertex set problem in the weighted and directed case, leading to a result that holds for any other feedback set problem as well. This approach is an algorithmic adaptation of a theoretical result due to P.D. Seymour [78], who proved that the integrality gap in the case of the unweighted feedback vertex set problem can be at most $O(\log \tau^* \log \log \tau^*)$, where τ^* is defined as above. Even et al. observe that all existence arguments contained in the proof of Seymour's statement can be made constructive and thus, with some additional operations, an algorithm for the unweighted feedback vertex set problem having an approximation factor of $O(\log \tau^* \log \log \tau^*)$ can be obtained. Further modifications of the algorithm lead to a polynomial time approximation scheme applicable to the weighted problem. In $O(|E(G)| \cdot |V(G)|^2)$ time the algorithm finds a feedback vertex set having weight

$$O\left(\min\{\tau^* \log \tau^* \log\log \tau^*, \tau^* \log |V(G)| \log\log |V(G)|\}\right).$$

All the observations contained in [27] improve the $O(\log^2 |V(G)|)$-approximation algorithm for this case [54]. In the case of directed planar graphs, H. Stamm [86] presented an $O(|V(G)| \log |V(G)|)$-approximation algorithm, whose performance guarantee is bounded by the maximum degree of the graph and an $O(|V(G)|^2)$ time approximation algorithm with performance guarantee no more than the number of cyclic faces in the planar embedding of the graph minus 1. M. Cai, X. Deng, and W. Zang [10] obtained a 2.5-approximation algorithm for the minimum feedback vertex set problem on tournaments, improving the previously known algorithm with performance guarantee of 3 by E. Speckenmeyer [85]. Let H be the triangle-vertex incidence matrix of a tournament T and let e be the all-one vector. In [10], necessary and sufficient con-

ditions are established for the linear system $\{x: Hx \geq e, x \geq 0\}$ to be a totally dual integral system (TDI).

Definition 2 A rational linear system $\{x: Ax \geq b, x \geq 0\}$ is called *totally dual integral*, if the optimization problem $\max \{y^{\mathsf{T}} b: y^{\mathsf{T}} A \leq c^{\mathsf{T}}, y \geq 0\}$ has an integral optimum solution y for every integral vector c for which the maximum is finite.

It has been shown that any rational polyhedron P has a TDI system $P = \{x: Ax \leq b\}$ representation with A integral, and that, if P is full-dimension, there is a unique minimal TDI system $P = \{x: Ax \leq b\}$ with A and b integral if and only if P is integral. In [11] the authors have extended this approach to the feedback vertex set problems and the cycle packing problem in *bipartite tournaments*, where a bipartite tournament is an orientation of a complete bipartite graph. For the aforementioned problems they have found strongly polynomial time algorithms, which are a consequence of a min-max relaxation on packing and covering directed cycles.

Exact Algorithms

In contrast to the numerous approximation schemes that have been studied, relatively few exact algorithms for the feedback vertex set problem have been proposed. To our knowledge, the first algorithm to find an exact minimal cardinality FVS is due to Smith and Walford [83], who proposed a particular graph partition technique. Although their algorithm solves the problem in an arbitrary directed graph in exponential running time, it returns an optimal solution in polynomial time for certain types of graphs. Later, exact algorithms of enumerative nature often used the graph reduction procedures to speed up the process. One study, [16], essentially used direct enumeration plus reduction and reported satisfactory computational results for a set of partial scan design test problems. T. Orenstein, Z. Kohavi, and I. Pomeranz [67] proposed a somewhat more involved exact enumerative procedure based on graph reduction and efficient graph partitioning methods. Their algorithm has been designed for identifying a minimum feedback vertex set in a digital circuit and it is efficient in random graphs, even though in cliques or graphs that are 'almost' cliques it has an exponential behavior, since the reduction and partition techniques cannot be applied.

Somewhat surprising, exact algorithms for feedback vertex set based on mathematical programming formulation are quite few. Recently (1996), M. Funke and G. Reinelt [32] considered a special variant of feedback problems, namely the problem of finding a maximum weight node induced acyclic subdigraph. They discussed valid and facet defining inequalities for the associated polytope and developed a polyhedral-based exact algorithm presenting computational results obtained by applying a branch and cut algorithm.

The Feedback Arc Set Problem

Given a graph $G = (V, E)$ and a nonnegative weight function w: $E(G) \rightarrow \mathbf{R}^+$ defined on the arcs of G, the feedback arc set problem consists of finding a minimum-weight subset of arcs $E' \subseteq E(G)$ that meets every cycle in a given collection C of cycles in (G, w). As in the vertex case, this leads to the *minimum feedback arc set problem* (MWFAS) in both directed and undirected graphs, the *minimum weighted graph bipartization problem* via arc removals, and so on.

Mathematical Model of the Feedback Arc Set Problem

Given an arc weighted graph (G, w), $G = (V, E)$ and the set C of all cycles in G, the minimum weighted feedback arc set problem can be formulated as the following integer programming problem:

$$\begin{cases} \min & \displaystyle\sum_{e \in E(G)} w(e) x_e \\ \text{s.t.} & \displaystyle\sum_{e \in \Gamma} x_e \geq 1, \quad \forall \Gamma \in C, \\ & x_e \in \{0, 1\}, \quad \forall e \in E(G). \end{cases}$$

In its relaxation, the constraints $x_e \in \{0, 1\}$, $\forall\, e \in E(G)$ are replaced by $x_e \geq 0$, $\forall\, e \in E(G)$, obtaining a fractional feedback arc set. As with the feedback vertex set problem, the feedback arc set problem is a *covering problem* and its (linear programming) dual is called a *packing problem*. In the case of the feedback arc set problem this means assigning a dual variable to all interesting cycles to be hit in the given graph, such that for each arc the sum of the variables corresponding to the interesting cycles passing through that arc is at most the weight of the arc itself.

State of the Art of Feedback Arc Set Problems

Feedback arc set problems tend to be easier than their vertex counterparts, especially for planar graphs. In the directed case feedback vertex and feedback arc set problems are each reducible to one another. Even, Naor, Schieber, and Sudan [27] showed how to perform reductions among feedback set problems and feedback subset problems and vice versa, preserving feasible solutions and their costs. In all reductions, there is a one-to-one correspondence between feasible solutions and their corresponding costs. Therefore, an approximate solution to one problem can be translated to an approximate solution of the other problem reducible to this problem. Because most of the reduction procedures can be performed in linear time, these problems can be viewed as different representations of the same problem. Hence, as feedback vertex sets are reduced into feedback arc sets with the same weight and vice versa, all of these problems are equally hard to approximate. In the literature of feedback set problems most of the proposed algorithms are designed to solve the problem in vertex-weighted graphs. One of the pioneering papers on feedback arc set problems is [76], where it is proved that finding a minimum feedback arc set in an arc-weighted reducible flow graph is as difficult as finding a minimum cut in a flow network. The proposed algorithm has complexity $O(mn^2 \log (n^2/m))$, where $m = |E(G)|$ and $n = |V(G)|$. The algorithm was adapted to solve the problem in the vertex-weighted case. Shamir's linear time algorithm [79], used for the unit-weighted case, cannot be applied to solve the arc-weighted problem, because any reduction between arc and vertex set problems does not preserve the reducibility property. Given a directed graph $G = (V, E)$, a *dijoin* $E' \subseteq E(G)$ is a set of arcs such that the graph $G' = (V, B)$, $B = E \cup \{(v, u): (u, v) \in E'\}$ is strongly connected. Given nonnegative weights w_e, $e \in E(G)$, the minimum-weight dijoin problem is to find the dijoin with minimum weight. The feedback arc set problem in planar digraphs is reducible to the problem of finding a minimum-weight dijoin in the dual graph, which is solvable in polynomial time [39]. Stamm [86] proposed a simple 2-approximation algorithm for the minimum weight dijoin problem by superposing two arborescences. It is interesting to observe that, when translated to the dual graph, all these problems lead to problems of hitting certain cutsets of the dual graph, problems which can be approximated within a ratio of 2 by the primal-dual method. Goemans and Williamson [37] proposed a primal-dual algorithm that finds a 9/4-approximate solution to feedback set problems in planar graphs. The first approximation algorithm for the feedback arc set problem was given in [54]. The approximation factor is $O(\log^2 n)$ in the unweighted case, where n is the number of vertices of the input graph. This bound was obtained by using a $O(\log n)$ approximation algorithm for a directed separator that splits the graph into two approximately equally-sized components, S and $\overline{S}$. This separator can be found by approximating special cuts called *quotient cuts*. This result was improved by Seymour [78], who gave a $O(\log n \log \log n)$-approximation algorithm that solves the linear relaxation of the feedback arc set mathematical model and then interprets the optimal fractional solution x^* as a length function defined on the arcs. Systematically, in a recursive fashion, it uses this length function to delete from the graph G all arcs between S and $\overline{S}$. Note that the linear program can be solved in polynomial time by using the ellipsoid or an interior point algorithm. Hence, the quality of the bound in this approach depends on the way the graph is partitioned. Seymour [78] proved the following lemma:

Lemma 3 *For a given strongly connected digraph $G = (V, E)$, suppose there exists a feasible solution x to the feedback arc set problem. If ϕ is the value of the optimal fractional solution x^*, then there exists a partition $(S, \overline{S})$ such that, for some ϵ, $0 < \epsilon < 1$, the following conditions hold: If $\delta^+(S) = \{(u, v): (u, v) \in E(G), u \in S, v \in \overline{S}\}$ and $\delta^-(S) = \{(v, u): (v, u) \in E(G), u \in S, v \in \overline{S}\}$, then the following is true:*

$$\sum_{e \in E(S)} w(e)x(e) \leq \epsilon\phi, \tag{4}$$

$$\sum_{e \in E(\overline{S})} w(e)x(e) \leq (1 - \epsilon)\phi, \tag{5}$$

and either

$$\sum_{e \in \delta^+(S)} w(e) \leq 20\epsilon\phi \log\left(\frac{1}{\epsilon}\right) \log\log\phi \tag{6}$$

or

$$\sum_{e \in \delta^-(S)} w(e) \leq 20\epsilon\phi \log\left(\frac{1}{\epsilon}\right) \log\log\phi. \tag{7}$$

Furthermore, the partition $(S, \overline{S})$ *can be found in polynomial time.*

This Lemma admits a constructive proof, [27]. The algorithm in this proof finds a feedback arc set having weight $O(\tau^* \log^2|X|)$, where X is a special set of vertices defining the cycles to be hit and τ^* is the weight of an optimal fractional feedback set. The idea is to reduce the problem to the directed minimum capacity multicut problem in circular networks and of adapting the undirected *sphere growing technique* described in [35] to directed circular networks. Then the graph is decomposed in the following way. A fractional and optimal solution to the directed feedback set problem induces a distance metric on the set of arcs (on the set of vertices) $E(G)$. The approximation algorithm arbitrarily picks a vertex $v \in X$ and solves the shortest path tree problem rooted at v with respect to the metric induced by the fractional solution. The procedure that finds the shortest path tree defines layers with respect to the source v. Each layer is a directed cut that partitions the graph into two parts. The next step of the approximation algorithm is to choose a directed cut and to add the cut to the feedback set constructed so far. The algorithm continues recursively in each part and ends when the graph does not contain any interesting cycles. The key of the algorithm is the choice of the criterion to select the directed cut that partitions the graph. Even et al. decided to relate the weight of the cut to the cost of the fractional solution. More recently (1996), Even, Naor, Schieber, and Zosin [28] showed that, for any weight function defined on the arcs, the subset feedback arc set problem can be approximated in polynomial time by a factor of two. The approximation algorithm consists of successive computations of minimum cuts. Its approximation factor is estimated by considering the capacities of minimum cuts as flow paths. When new minimum cuts are computed, previous flow paths are updated according to the decomposition of the graph induced by an optimal solution.

A GRASP for Feedback Set Problems

Although the approximation algorithms guarantee a solution of a certain quality, for many practical real world cases, heuristic methods can lead to better solutions in a reasonable amount of CPU time. Metaheuristics, such as genetic algorithms, simulated annealing, greedy randomized adaptive search procedures (GRASP), Lagrangian relaxation, and others have been developed with successful computational performance on a wide range of combinatorial optimization problems. Interestingly, however, feedback vertex set problems seem to be an exception. For this family of problems relatively few practical heuristics have been developed. Furthermore, most of the heuristics that seem to be quite successful computationally are greedy type heuristics or generalized greedy type heuristics (e. g. GRASP). Almost all the efficient heuristics developed so far employ the solution-preserved reduction rules studied in [56]. It has been observed in practice that this group of heuristics greatly reduces the cardinality of the graph not only at the beginning of the algorithm, but also dynamically during the execution of node deletion type heuristics. A recent line of research on heuristic approaches is due to P.M. Pardalos, T. Qian, and M.G.C. Resende [70] where three variants of the so-called *greedy randomized adaptive search procedure* (*GRASP*) metaheuristic are proposed for finding approximate solutions of large instances of the feedback vertex set problem in a digraph. GRASP is a multistart method characterized by two phases: a construction phase and a local search phase, also known as a local improvement phase. During the construction phase a feasible solution is iteratively constructed. One element at time is randomly chosen from a *restricted candidate list* (RCL), whose elements are sorted according to some greedy criterion, and is added to the building feedback vertex set and removed from the graph with all its incident arcs. Since the computed solution, in general, may not be locally optimal with respect to the adopted neighborhood definition, the local search phase tries to improve it. These two phases are iterated and the best solution found is kept as an approximation of the optimal solution. To improve the efficiency of the method, Pardalos et al. incorporated in each iteration of their algorithm solution-preserving graph reduction techniques in their directed version and that can be used also to check if a digraph is acyclic, returning an empty reduced graph in case of positive answer. The authors employed the following three greedy functions used to select the node with the maximum $G(i)$ values:

- $G_A\,(i) = \text{in}(i) + \text{out}(i)$;
- $G_B\,(i) = \text{in}(i) * \text{out}(i)$;
- $G_C\,(i) = \max\{\text{in}(i), \text{out}(i)\}$.

Greedy function G_A assigns equal weight to in- and out-degrees. G_B favors the balance between in- and out-degrees. G_C only considers the largest value of the degrees. As demonstrated in [70], G_B produced the best computational results. GRASP was tested on two randomly generated problem sets, finding the optimal solutions to all the problems in the first set, where the optimal values are known (computed in [32]). Furthermore, this GRASP dominates the pure greedy heuristics in all the test instances with comparable running time. In [31], Fortran subroutines are given for finding approximate solutions of both the directed feedback vertex set problem and the directed feedback arc set problem using GRASP. The subroutines for solving approximately the feedback vertex set problem corresponds to the pseudocode algorithm proposed in [70]. The subroutines for solving approximately the feedback arc set problem uses a linear-time procedure proposed in [27] in order to reduce the given feedback arc set problem instance to an equivalent feedback vertex set problem instance, and then the reduced vertex version problem is solved.

Future Research

As has been pointed out in [38], fast construction heuristics combined with local improvement techniques tailored for special applications have been the 'workhorse' of combinatorial optimization in practice. As the design of efficient construction heuristics and local search procedures will be a key to the effective computational procedure for feedback set problems, new approaches are considered that will lead to higher quality solution. New variants of the classical GRASP approach are considered, called *Reactive GRASP* techniques. The first idea along this line has been due to M. Prais and C.C. Ribeiro [74], who used reactive GRASP to a matrix decomposition problem arising in the context of traffic scheduling in satellite-division-multiple-access systems (SS/TDMA). In the reactive GRASP, the restricted candidate list parameter α is not fixed, but selfadjusted according to the quality of the solution previously found during the search. In more detail, the parameter α is randomly chosen from a set of m predeterminated acceptable values $A = \{\alpha_1, \ldots, \alpha_m\}$. Associated with the choice of α_i there is a probability p_i, initially corresponding to a uniform distribution. During the search phase some information is collected in order to change the discrete set of probabilities $\{p_i\}_{i=1,\ldots,m}$. Several possible strategies can be explored for this update operation. One among them has been proposed by Prais and Ribeiro. It is an *absolute qualification rule*, based on the average value of the solutions obtained with each value of $\alpha = \alpha_i$. Once chosen the updating criterion of the probabilities $\{p_i\}_{i=1,\ldots,m}$, it is possible to use different values of α at different iterations. Therefore, different restricted candidate lists can be built and eventually different solutions can be constructed, which would never be built by using a single, fixed value of α.

T.A. Feo and Resende have discussed in [29] the effects the parameter α can have on the quality of the solution and, at least analyzing the results obtained by Prais and Ribeiro, it seems that α can have an evident impact on the outcome of a GRASP procedure.

Conclusions

Despite the large body of work on approximation algorithms, computational studies of feedback set problems seem to be still in their embryonic stage. No modern metaheuristics, except the GRASP procedure recently (1996) developed in [70] have ever been applied to the feedback vertex set problem. The size of the general problem that can be handled is still quite limited. It seems that this area of computational research has the greatest potential for progress and impact in the coming years. It has to be also underlined that, since detecting cycles is a relatively expensive operation, the local search of feedback vertex set appears to be even more difficult than other notorious combinatorial problems like the traveling salesperson or set covering problems. Therefore, the design of efficient local search procedures and fast construction heuristics will be a key to the effective computational procedure for feedback set problems.

See also

- ► Generalized Assignment Problem
- ► Graph Coloring
- ► Graph Planarization
- ► Greedy Randomized Adaptive Search Procedures
- ► Quadratic Assignment Problem
- ► Quadratic Semi-assignment Problem

References

1. Bafna V, Berman P, Fujito T (1995) Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs. In: Staples J, Eades P, Katoh N, Moffat A (eds) ISAAC95, Algorithms and Computation. Lecture Notes Computedr Sci. Springer, Berlin, pp 142–151
2. Bar-Yehuda R, Geiger D, Naor J, Roth RM (1998) Approximation algorithms for the vertex feedback set problem with applications to constraint satisfaction and Bayesian inference. SIAM J Comput 27(4):942–959
3. Becker A, Geiger D (1994) Approximation algorithm for the loop cutset problem. 10th Conf. Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Mateo, pp 60–68
4. Becker A, Geiger D (1996) Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. Artif Intell 83:167–188
5. Bondy JA, Hopkins G, Staton W (1987) Lower bounds for induced forests in cubic graphs. Canad Math Bull 30:193–199
6. Bovet DP, de Agostino S, Petreschi R (1988) Parallelism and the feedback vertex set problem. Inform Process Lett 28:81–85
7. Brandstädt A (1993) On improved time bounds for permutation graph problems. In: 18th Workshop on Graph-theoretic concepts in computer science, vol 657. In: Lecture Notes Computer Sci, vol 657. Wiesbaden-Naurod and Springer, Berlin, pp 1–10
8. Brandstädt A, Kratsch D (1985) On the restriction of some NP-complete graph problems to permutation graphs. In: Budach L (ed) Fundamentals of Computing Theory. Lecture Notes Computer Sci. Springer, Berlin, pp 53–62
9. Breuer MA, Gupta R (1989) BALLAST: A methodology for partial scan design. 19th Internat Symposium on Fault-Tolerant Computing, pp 118–125
10. Cai M, Deng X, Zang W (1998) A TDI system and its application to approximation algorithm. 39th Annual Symposium on Foundations of Computer Sci
11. Cai M, Deng X, Zang W (1999) A min-max theorem on feedback vertex sets. Integer Programming and Combinatorial Optimization. Proc 7th Internat IPCO Conf. In: Lecture Notes Computer Sci. Springer, Berlin
12. Chakradhar S, Balakrishnan A, Agrawal V (1994) An exact algorithm for selecting partial scan flip-flops. unpublished
13. Chang MS, Liang YD (1997) Minimum feedback vertex sets in cocomparability graphs and convex bipartite graphs. Acta Informatica 34:337–346
14. Charon I, Guenoche A, Hudry O, Wairgard F (1997) New results on the computation of median orders. Discret Math 165/166:139–153
15. Chen R, Guo X, Zhang F (1988) The z-transformation graphs of perfect matchings of hexagonal system. Discret Math 72:405–415
16. Cheng KT, Agrawal VD (1990) A partial scan method for sequential circuits with feedback. IEEE Trans Comput 39(4):544–548
17. Chudak FA, Goemans MX, Hochbaum D, Williamson DP (1998) A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs. Oper Res Lett 22:111–118
18. Chvátal V (1979) A greedy heuristic for the set covering problem. Math Oper Res 4:233–235
19. Coorg SR, Rangan CP (1995) Feedback vertex set on cocomparability graphs. Networks 26:101–111
20. Corneil DG, Fonlupt J (1988) The complexity of generalized clique covering. Discrete Appl Math 22:109–118
21. Dechter R (1990) Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. Artif Intell 41:273–312
22. Dechter R, Pearl J (1987) The cycle cutset method for improving search performance in AI. In: 3rd IEEE on AI Applications
23. Donald J, Elwin J, Hager R, Salamon P (1995) A bad example for the minimum feedback vertex set problem. IEEE Trans Circuits and Systems 32:491–493
24. Downey RG, Fellows MR (1995) Fixed-parameter tractability and completeness I: Basic results. SIAM J Comput 24:873–921
25. Erdös P, Posa L (1962) On the maximal number of disjoint circiuts of a graph. Publ Math Debrecen 9:3–12
26. Even G, Naor JS, Zosin L. An 8-approximation algorithm for the subset feedback vertex problem proposed a 8-approximation algorithm
27. Even G, Naor S, Schieber B, Sudan M (1998) Approximating minimum feedback sets and multicuts in directed graphs. Algorithmica 20:151–174
28. Even G, Naor S, Schieber B, Zosin L (1996) Approximating minimum subset feedback sets in undirected graphs, with applications. 4th Israel Symposium on Theory of Computing and Systems, pp 78–88
29. Feo TA, Resende MG (1995) Greedy randomized adaptive search procedures. J Global Optim 6:109–133
30. Festa P, Pardalos PM, Resende MGC (1999) Feedback set problems. In: Du D-Z, Pardalos PM (eds) Handbook Combinatorial Optim, vol 4, pp 209–258
31. Festa P, Pardalos PM, Resende MGC (1999) Fortran subroutines for approximate solution of feedback vertex set problems using GRASP. AT&T Lab Res, Florham Park
32. Funke M, Reinelt G (1996) A polyhedral approach to the feedback vertex set problem. unpublished
33. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, New York
34. Garey MR, Tarjan RE (1978) A linear-time algorithm for finding all feedback vertices. Inform Process Lett 7:274–276
35. Garg N, Vazirani VV, Yannakakis M (1996) Approximate max-flow min-(multi) cut theorems and their applications. SIAM J Comput 25(2):235–251

36. Gavril F (1977) Some NP-complete problems on graphs. 11th Conf Inform Sci and Systems, Johns Hopkins Univ Press, Baltimore, pp 91–95
37. Goemans MX, Williamson DP (1996) Primal-dual approximation algorithms for feedback problems in planar graphs. 5th MPS Conf Integer Programming and Combinatorial Optimization (IPCO), pp 147–161
38. Grötschel M, Lovász L (1993) Combinatorial optimization: A survey. Techn Report, DIMACS Rutgers Univ 29
39. Grötschel M, Lovász L, Schrijver A (1988) Geometric algorithms and combinatorial optimization. Springer, Berlin, pp 253–254
40. Harary F, Klein DJ, Zivkovic TP (1991) Graphical properties of polyhexes: Perfect matching vector and forcing. J Math Chem 6:295–306
41. Hochbaum D (1982) Approximation algorithms for set covering and vertex cover problem. SIAM J Comput 11(3):555–556
42. Hu TC (1963) Multi-commodity network flows. Oper Res 11:344–360
43. Isaak G (1995) Tournaments as feedback arc sets. Electronic J Combin 20(2):1–19
44. Johnson DS (1974) Approximation algorithms for combinatorial problems. J Comput Syst Sci 9:256–278
45. Johnson DB (1975) Finding all the elementary circuits of a directed graph. SIAM J Comput 4(1):77–84
46. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity Of Computer Computations. Plenum, New York, pp 85–103
47. Kevorkian AK (1980) General topological results on the construction of a minimum essential set of a directed graph. IEEE Trans Circuits and Systems 27:293–304
48. Kim H, Perl J. A computational model for combined causal and diagnostic reasoning in inference systems. 8th IJCAI, Morgan Kaufmann, San Mateo, pp 190–193
49. Klein DJ, Randić M (1987) Innate degree of freedom of a graph. J Comput Chem 8:516–521
50. Klein DJ, Zivković TP, Valenti R (1991) Topological long-range order for resonating-valance-bond structures. Phys Rev B 43A:723–727
51. Kunzmann A, Wunderlich HJ (1990) An analytical approach to the partial scan problem. J Electronic Testing: Th Appl 1:163–174
52. Lauritzen SL, Spiegelhalter DJ (1988) Local computations with probabilities on graphical structures and their application to expert systems (with discussion). J Royal Statist Soc B 50:157–224
53. Lee D, Reedy S (1990) On determining scan flip-flops in partial scan designs. Internat Conf Computer Aided Design, pp 322–325
54. Leighton T, Rao S (1988) An approximate max-flow mincut theorem for uniform multicommodity flow problems with applications to approximation algorithms. 29th Annual Symposium on Fundations of Computer Sci, pp 422–431
55. Lempel A, Cederbaum I (1966) Minimum feedback arc and vertex sets of a directed graph. IEEE Trans Circuit Theory CT-13:399–403
56. Levy H, Lowe L (1988) A contraction algorithm for finding small cycle cutsets. J Algorithms 9:470–493
57. Li X, Zhang F (1995) Hexagonal systems with forcing edges. Discret Math 140:253–263
58. Liang YD (1994) On the feedback vertex set problem in permutation graphs. Inform Process Lett 52:123–129
59. Liu J, Zhao C (1996) A new bound on the feedback vertex sets in cubic graphs. Discret Math 148:119–131
60. Lloyd EL, Soffa ML, Wang CC (1988) On locating minimum feedback vertex sets. J Comput Syst Sci 37:292–311
61. LuChin Lung, Tang Chuan Yi (1997) A linear-time algorithm for the weighted feedback vertex problem on interval graphs. Inform Process Lett 61:107–111
62. Lucchesi CL, Younger DH (1978) A minimax theorem for directed graphs. J London Math Soc 17:369–374
63. Luccio FL (1998) Almost exact minimum feedback vertex set in meshes and butterflies. Inform Process Lett 66:59–64
64. Lund C, Yannakakis M (1993) On the hardness of approximating minimization problems. 25th ACM Symp on Theory Of Computing, pp 286–293
65. Marathe MV, Pandu Rangan C, Ravi R (1992) Efficient algorithms for generalized clique covering on interval graphs. Discrete Appl Math 39:87–93
66. Monien B, Schultz R (1981) Four approximation algorithms for the feedback vertex set problems. 7th Conf Graph Theoretic Concepts of Computer Sci. Hauser, pp 315–326
67. Orenstein T, Kohavi Z, Pomeranz I (1995) An optimal algorithm for cycle breaking in directed graphs. J Electronic Testing: Th Appl 7:71–81
68. Pachter L, Kim P (1998) Forcing matchings on square grids. Discret Math 190:287–294
69. Papadimitriou C, Yannakakis M (1988) Optimization, approximation and complexity classes. 20th Annual ACM Symp on Theory of Computing, pp 251–277
70. Pardalos PM, Qian T, Resende MGC (1999) A greedy randomized adaptive search procedure for feedback vertex set. J Combin Optim 2:399–412
71. Peleg D (1996) Local majority voting, small coalitions, and controlling monopolies in graphs: A review. 3rd Colloq Structural Information and Communication Complexity, pp 152–169
72. Peleg D (1997) Size bounds for dynamic monopolies. 4th Colloquium on Structural Information and Communication Complexity, Carleton Univ Press, Ottawa, pp 165–175
73. Perl J (1986) Fusion, propagation and structuring in belief networks. Artif Intell 29:241–288
74. Prais M, Ribeiro CC. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment
75. Qian T, Ye Y, Pardalos PM (1995) A pseudo-ϵ approximation algorithm for feedback vertex set. In: Floudas CA,

Pardalos PM (eds) Recent Advances in Global Optimization. Kluwer, Dordrecht, pp 341–351
76. Ramachandran V (1988) Finding a minimum feedback arc set in reducible flow graphs. J Algorithms 9:299–313
77. Rosen B (1982) Robust linear algorithms for cutsets. J Algorithms 3:205–217
78. Seymour PD (1995) Packing directed circuits fractionally. Combinatorica 15:281–288
79. Shamir A (1979) A linear time algorithm for finding minimum cutsets in reduced graphs. SIAM J Comput 8(4):645–655
80. Shatcher RD, Andersen SK, Szolovits P (1994) Global conditioning for probabilistic inference in belief networks. In: 10 Conf Uncertainty in AI, pp 514–522
81. Shaw AC (1974) The logical design of operating systems. Prentice-Hall, Upper Saddle River
82. Simovici DA, Grigoras G (1979) Even initial feedback vertex set problem is NP-complete. Inform Process Lett 8:64–66
83. Smith GW, Walford RB (1975) The identification of a minimal feedback vertex set of a directed graph. IEEE Trans Circuits and Systems CAS-22(1):9–14
84. Speckenmeyer E (1988) On feedback vertex sets and nonseparating independent sets in cubic graphs. J Graph Theory 12:405–412
85. Speckenmeyer E (1989) On feedback problems in digraphs. Lecture Notes Computer Sci, vol 411. Springer, Berlin, pp 218–231
86. Stamm H (1990) On feedback problems in a planar digraph. In: Möhring R (ed) Graph-Theoretic Concepts in Computer Sci. Lecture Notes Computer Sci, vol 484. Springer, Berlin, pp 79–89
87. Tarjan RE (1972) Depth first search and linear graph algorithms. SIAM J Comput 1:146–160
88. Ueno S, Kajitani Y, Gotoh S (1988) On the nonseparating independent set problem and feedback set problem for graphs with no vertex degree exceeding three. Discret Math 72:355–360
89. Vazirani V. Approximation algorithms. Manuscript College of Computing, Georgia Inst Techn
90. Wang C, Lloyd E, Soffa M (1985) Feedback vertex sets and cyclically reducible graphs. J ACM 32(2):296–313
91. Yannakakis M (1978) Node and edge-deletion NP-complete problems. 10th Annual ACM Symp Theory of Computing, pp 253–264
92. Yannakakis M (Feb. 1994) Some open problems in approximation. Second Italian Conf Algorithm and Complexity, CIAC'94, pp 33–39
93. Yannakakis M, Gavril F (1987) The maximum k-colorable subgraph problem for chordal graphs. Inform Process Lett 24:133–137
94. Younger DH (1963) Minimum feedback arc set for a directed graph. IEEE Trans Circuit Theory CT-10:238–245
95. Zheng M, Lu X (1990) On the maximum induced forests of a connected cubic graph without triangles. Discret Math 85:89–96

Fejér Monotonicity in Convex Optimization

PATRICK L. COMBETTES[1,2]
[1] Lab. d'Anal. Numér., University Pierre et Marie Curie, Paris, France
[2] City College and Graduate School City, University New York, New York, USA

MSC2000: 47H05, 90C25, 90C55, 65J15, 90C25

Article Outline

Keywords
Notation and Assumptions
Basic Convergence Properties
  Weak Convergence
  Strong Convergence
  Linear Convergence
Geometric Construction
Applications
  Fixed Points of Nonlinear Operators
  Zeros of Monotone Maps
  Zeros of the Sum of Two Monotone Maps
  Variational Inequalities
  Differentiable Optimization
  Convex Feasibility Problems
  Nondifferentiable Optimization
  Inconsistent Convex Feasibility Problems
Historical Notes and Comments
See also
References

Keywords

Fejér monotonicity; Convex programming; Convergence

Let S be a nonempty closed and convex set in a real Hilbert space $\mathcal{H}$ with norm $\|\cdot\|$. A sequence $(x_n)_{n\geq 0}$ of points in $\mathcal{H}$ is said to be *Fejér monotone* with respect to S (or simply *S-Fejérian*) if

$$\forall \overline{x} \in S \; \forall n \in \mathbb{N}: \quad \|x_{n+1} - \overline{x}\| \leq \|x_n - \overline{x}\|. \tag{1}$$

In words, each point in the sequence is not further from any point in S than its predecessor. Given $x_0 \in \mathcal{H}$, a typical example of S-Fejérian sequence is that generated by the algorithm

$$\forall n \in \mathbb{N}: \quad x_{n+1} = Tx_n,$$

where $T:\mathcal{H} \to \mathcal{H}$ is a nonexpansive operator, i. e.,

$$\forall (x, y) \in \mathcal{H}^2 : \ \|Tx - Ty\| \leq \|x - y\| , \tag{2}$$

with nonempty fixed point set S. Under suitable assumptions, the sequence of successive approximations $(x_n)_{n \geq 0}$ converges to a point in S [20].

In convex optimization, one frequently encounters algorithms whose orbits $(x_n)_{n \geq 0}$ are Fejér monotone with respect to the solution set. In order to simplify and standardize the convergence proofs of this broad class of algorithms, it is important to investigate the notion of Fejér monotonicity and to bring out some general convergence principles. These are precisely the objectives of the present article.

Notation and Assumptions

Throughout, the sequence $(x_n)_{n \geq 0}$ is Fejér monotone with respect to a nonempty closed and convex set S in a real Hilbert space $\mathcal{H}$ with scalar product $\langle \cdot | \cdot \rangle$, norm $\|\cdot\|$, and distance d. For every $n \in \mathbf{N}$, p_n denotes be the projection of x_n onto S, i. e., the unique point $p_n \in S$ such that $\|x_n - p_n\| = d(x_n, S)$. Recall that p_n is characterized by the variational inequality

$$\forall \overline{x} \in S : \ \langle \overline{x} - p_n | x_n - p_n \rangle \leq 0. \tag{3}$$

The expressions $x_n \rightharpoonup x$ and $x_n \to x$ denote respectively the weak and strong convergence of $(x_n)_{n \geq 0}$ to x. $\mathfrak{W}$ and $\mathfrak{S}$ denotes respectively the sets of weak and strong cluster points of $(x_n)_{n \geq 0}$. Finally, Id denotes the identity operator on $\mathcal{H}$.

Basic Convergence Properties

By way of preamble, some immediate consequences of (1) are stated below.

Proposition 1 *The following assertions hold.*

i) $(x_n)_{n \geq 0}$ *is bounded.*

ii) $\forall \overline{x} \in S : \ (\|x_n - \overline{x}\|)_{n \geq 0}$ *converges.*

iii) $(d(x_n, S))_{n \geq 0}$ *is nonincreasing.*

iv) $\forall \overline{x} \in S : \ x_n \to \overline{x}$ *if and only if* $\underline{\lim} \|x_n - \overline{x}\| = 0$ *if and only if* $S \cap \mathfrak{S} \neq \emptyset$.

Weak Convergence

In general, Fejér monotone sequences do not converge, even weakly (consider for instance the $\{0\}$-Fejérian sequence $((-1)^n x_0)_{n \geq 0}$ with $x_0 \neq 0$). By virtue of Proposition 1i), $\mathfrak{W} \neq \emptyset$ and a necessary condition for $(x_n)_{n \geq 0}$ to converge weakly to a point in S is $\mathfrak{W} \subset S$. A remarkable consequence of Fejér monotonicity is that this condition is also sufficient. To see this, take y_1 and y_2 in $\mathfrak{W}$, say $x_{k_n} \rightharpoonup y_1$ and $x_{l_n} \rightharpoonup y_2$, and $\overline{x} \in S$. By Proposition 1ii),

$$\lim \|x_{k_n} - \overline{x}\|^2 = \lim \|x_{l_n} - \overline{x}\|^2 .$$

Therefore, by expanding,

$$\lim \|x_{k_n}\|^2 - \lim \|x_{l_n}\|^2 = 2 \langle \overline{x} | y_1 - y_2 \rangle .$$

It follows that

$$S \subset \{x \in \mathcal{H} : \ \langle x | y_1 - y_2 \rangle = \alpha\} , \tag{4}$$

where $\alpha = (\lim \|x_{k_n}\|^2 - \|x_{l_n}\|^2)/2$. Thus, $(y_1, y_2) \in S^2 \Rightarrow \alpha = \langle y_1 | \ y_1 - y_2 \rangle = \langle y_2 \ | \ y_1 - y_2 \rangle \Rightarrow y_1 = y_2$. Consequently, the bounded sequence $(x_n)_{n \geq 0}$ cannot have more than one weak cluster point in S. This fundamental property will be recorded as:

Proposition 2 *$(x_n)_{n \geq 0}$ converges weakly to a point in S if and only if $\mathfrak{W} \subset S$.*

Two additional properties are worth mentioning in connection with weak convergence.

- Let $\overline{\text{aff}} S$ be the closed affine hull of S. If $y_1 \neq y_2$, then (4) asserts that S is contained in a closed affine hyperlane. If $\overline{\text{aff}} S = \mathcal{H}$, $\mathfrak{W}$ reduces to a singleton and $(x_n)_{n \geq 0}$ therefore converges weakly.
- Suppose that $x_n \rightharpoonup \overline{x} \in S$ and let $x \in \mathcal{H}$. Then the identities

$$\begin{aligned} &\forall n \in \mathbb{N} : \ \|x_n - x\|^2 \\ &= \|x_n - \overline{x}\|^2 + 2 \langle x_n - \overline{x} | \overline{x} - x \rangle + \|\overline{x} - x\|^2 \end{aligned}$$

together with Proposition 1ii) imply that $(\|x_n - x\|)_{n \geq 0}$ converges.

Strong Convergence

As evidenced by the classical counterexample of [13], $x_n \rightharpoonup \overline{x} \in S \not\Rightarrow x_n \to \overline{x} \in S$. Accordingly, strong convergence conditions for Fejér monotone sequences must be identified.

First, consider the projected sequence $(p_n)_{n\geq 0}$. It follows from (1) and (3) that for every $(m, n) \in \mathbf{N}^2$

$$\begin{aligned}
&\|p_n - p_{n+m}\|^2 \\
&= \|p_n - x_{n+m}\|^2 + 2\langle p_n - x_{n+m} | x_{n+m} - p_{n+m}\rangle \\
&\quad + \|x_{n+m} - p_{n+m}\|^2 \\
&\leq d(x_n, S)^2 - d(x_{n+m}, S)^2 \\
&\quad + 2\langle p_n - p_{n+m} | x_{n+m} - p_{n+m}\rangle \\
&\leq d(x_n, S)^2 - d(x_{n+m}, S)^2.
\end{aligned}$$

Consequently, since $(d(x_n, S))_{n\geq 0}$ converges by Proposition 1iii), $(p_n)_{n\geq 0}$ is a Cauchy sequence. This establishes:

Proposition 3 *$(p_n)_{n\geq 0}$ converges strongly.*

This result, which is of interest in its own right, also leads to a simple criterion for the strong convergence of $(x_n)_{n\geq 0}$ to a point in S. Indeed, suppose that $\underline{\lim}\, d(x_n, S) = 0$. Then, thanks to Proposition 1iii), $d(x_n, S) \to 0$, i. e., $x_n - p_n \to 0$. On the other hand, by Proposition 3, $p_n \to \overline{x}$ with $\overline{x} \in S$ since S is closed. One thus obtains:

Proposition 4 *$(x_n)_{n\geq 0}$ converges strongly to a point in S if and only if $\underline{\lim}\, d(x_n, S) = 0$.*

Going back to (4), assume now that $(y_1, y_2) \in \mathfrak{S}^2$. Then $\alpha = (\|y_1\|^2 - \|y_2\|^2)/2$ and (4) therefore becomes

$$\begin{aligned}
S &\subset \left\{ x \in \mathcal{H} : \left\langle x - \frac{y_1 + y_2}{2} \,\middle|\, y_1 - y_2 \right\rangle = 0 \right\} \\
&= \{ x \in \mathcal{H} : \|x - y_1\| = \|x - y_2\| \}. \qquad (5)
\end{aligned}$$

In words, if $(x_n)_{n\geq 0}$ possesses two distinct strong cluster points y_1 and y_2, S is contained in the closed affine hyperplane whose elements are equidistant from y_1 and y_2. If $\overline{\mathrm{aff}} S = \mathcal{H}$, it results from (5) that $(x_n)_{n\geq 0}$ possesses at most one strong cluster point. This happens in particular when the interior of S is nonempty (Slater condition). In this case, however, a sharper result holds, namely $(x_n)_{n\geq 0}$ converges strongly [22].

Linear Convergence

Proposition 1iii) asserts that $(d(x_n, S))_{n\geq 0}$ is nonincreasing. Assume now that it decreases at a linear rate, say

$$\exists \kappa \in]0, 1[\ \forall n \in \mathbb{N} : \ d(x_{n+1}, S) \leq \kappa d(x_n, S). \qquad (6)$$

Then, in view of Proposition 4, $x_n \to \overline{x} \in S$. On the other hand, for every $(m, n) \in \mathbf{N}^2$, (1) yields

$$\begin{aligned}
&\|x_n - x_{n+m}\| \\
&\leq \|x_n - p_n\| + \|x_{n+m} - p_n\| \\
&\leq 2d(x_n, S).
\end{aligned}$$

Thus $\|x_n - \overline{x}\| \leq 2d(x_n, S)$ and one reaches the following conclusion.

Proposition 5 *Suppose that (6) holds. Then $(x_n)_{n\geq 0}$ converges linearly to a point $\overline{x} \in S$: $\forall n \in \mathbb{N} : \ \|x_n - \overline{x}\| \leq 2\kappa^n d(x_0, S)$*

Geometric Construction

In order to make the above theoretical convergence results more readily applicable in concrete problems, it will henceforth be assumed that $(x_n)_{n\geq 0}$ has been generated by the following algorithm.

0	Take $x_0 \in \mathcal{H}$ and set $n = 0$.
1	Generate a closed affine half-space H_n such that $S \subset H_n$.
2	Compute the projection $P_n x_n$ of x_n onto H_n and take $\lambda_n \in [0, 2]$.
3	Set $x_{n+1} = x_n + \lambda_n(P_n x_n - x_n)$.
4	Set $n = n + 1$ and go to step 1.

Fejér Monotonicity in Convex Optimization, Algorithm 1
General Fejérian scheme

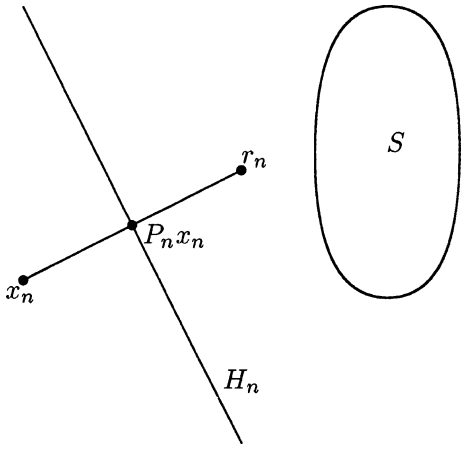
The relaxation parameter λ_n determines the position of the update x_{n+1} on the closed segment between the current iterate x_n and its reflection $r_n = 2P_n x_n - x_n$ with respect to H_n (see Fig. 1.). In some problems, it is possible to significantly accelerate the progression of the iterates towards a solution by proper choice of the relaxation sequence $(\lambda_n)_{n\geq 0}$ [5].

Hereafter, two properties of the relaxation sequence will be considered, namely

$$\sum_{n\geq 0} \lambda_n(2 - \lambda_n) = +\infty \qquad (7)$$

and

$$(\lambda_n)_{n\geq 0} \text{ lies in } [\varepsilon, 2 - \varepsilon], \text{ where } \varepsilon \in]0, 1[. \qquad (8)$$



Fejér Monotonicity in Convex Optimization, Figure 1
A Fejérian iteration

Now fix $\overline{x} \in S$. Then, for every $n \in \mathbf{N}$,

$$\begin{aligned}
&\|x_{n+1} - \overline{x}\|^2 \\
&= \|x_n - \overline{x}\|^2 + \lambda_n^2 \|P_n x_n - x_n\|^2 \\
&\quad + 2\lambda_n \langle x_n - \overline{x} | P_n x_n - x_n \rangle \\
&\leq \|x_n - \overline{x}\|^2 - \lambda_n (2 - \lambda_n) d(x_n, H_n)^2. \qquad (9)
\end{aligned}$$

Consequently, $(x_n)_{n \geq 0}$ is S-Fejérian and

$$\sum_{n \geq 0} \lambda_n (2 - \lambda_n) d(x_n, H_n)^2 < +\infty. \qquad (10)$$

Furthermore, if $(\lambda_n)_{n \geq 0}$ lies in $[0, 2 - \varepsilon]$ for some $\varepsilon \in]0, 1[$, then the series $\sum_{n \geq 0} \|x_{n+1} - x_n\|^2$ and $\sum_{n \geq 0} \langle \overline{x} - x_n | x_{n+1} - x_n \rangle$ converge [6,15].

In view of (10), the next two convergence results are immediate consequences of Proposition 2 and 4, respectively.

Proposition 6 *$(x_n)_{n \geq 0}$ converges weakly to a point in S if one of the conditions below is fulfilled.*

i) *(10)* $\Rightarrow \mathfrak{W} \subset S$.
ii) *(7) is in force and* $\underline{\lim}\, d(x_n, H_n) = 0 \Rightarrow \mathfrak{W} \subset S$.
iii) *(8) is in force and* $\sum_{n \geq 0} d(x_n, H_n)^2 < +\infty \Rightarrow \mathfrak{W} \subset S$.

Proposition 7 *$(x_n)_{n \geq 0}$ converges strongly to a point in S if one of the conditions below is fulfilled.*

i) *(10)* $\Rightarrow \underline{\lim}\, d(x_n, S) = 0$.
ii) *(7) is in force and* $\underline{\lim}\, d(x_n, H_n) = 0 \Rightarrow \underline{\lim}\, d(x_n, S) = 0$.
iii) *(8) is in force and* $\sum_{n \geq 0} d(x_n, H_n)^2 < +\infty \Rightarrow \underline{\lim}\, d(x_n, S) = 0$.

To investigate linear convergence, assume that

$$\exists \eta \in]0, 1[\ \forall n \in \mathbb{N}: \ d(x_n, H_n) \geq \eta d(x_n, S) \qquad (11)$$

and that (8) holds. Then $\overline{x} = p_n$ in (11) supplies

$$\begin{aligned}
d(x_{n+1}, S)^2 &\leq \|x_{n+1} - p_n\|^2 \\
&\leq d(x_n, S)^2 - \varepsilon^2 d(x_n, H_n)^2 \\
&\leq (1 - \varepsilon^2 \eta^2) d(x_n, S)^2.
\end{aligned}$$

Whence, Proposition 5 yields:

Proposition 8 *Suppose that (8) and (11) hold. Then $(x_n)_{n \geq 0}$ converges linearly to a point $\overline{x} \in S$: $\forall n \in \mathbb{N}$: $\|x_n - \overline{x}\| \leq 2\kappa^n d(x_0, S)$ with $\kappa = (1 - \varepsilon^2 \eta^2)^{1/2}$.*

Applications

Several convex optimization methods are now presented. They are shown to be Fejér monotone and their convergence is established on the basis of the general results stated above. For brevity, only weak convergence is considered; however, strong and linear convergence results can be derived in a like manner under suitable assumptions. In each problem, the solution set S is assumed to be nonempty.

Fixed Points of Nonlinear Operators

For every $n \in \mathbf{N}$, let $T_n: \mathcal{H} \to \mathcal{H}$ be a *firmly nonexpansive* operator, i. e.,

$$\begin{aligned}
&\forall (x, y) \in \mathcal{H}^2: \\
&\qquad \langle T_n x - T_n y | x - y \rangle \geq \|T_n x - T_n y\|^2, \qquad (12)
\end{aligned}$$

and let $\operatorname{Fix} T_n = \{x \in \mathcal{H}: T_n x = x\}$ be its fixed point set. The problem under consideration is to find a common *fixed point* of the family $(T_n)_{n \geq 0}$, i. e.,

$$\begin{cases} \text{Find} & \overline{x} \in \mathcal{H} \\ \text{s.t.} & \forall n \in \mathbb{N}: \ T_n \overline{x} = \overline{x}. \end{cases} \qquad (13)$$

Let $S = \bigcap_{n \geq 0} \operatorname{Fix} T_n$ and

$$H_n = \{x \in \mathcal{H}: \ \langle x - T_n x_n | x_n - T_n x_n \rangle \leq 0\}.$$

It then follows from (12) that $S \subset \operatorname{Fix} T_n \subset H_n$. Thus, Algorithm 1 takes the following form.

0	Take $x_0 \in \mathcal{H}$ and set $n = 0$.
1	Take $\lambda_n \in [0, 2]$.
2	Set $x_{n+1} = x_n + \lambda_n(T_n x_n - x_n)$.
3	Set $n = n + 1$ and go to step 1.

Fejér Monotonicity in Convex Optimization, Algorithm 2
Common fixed point

Noting that $d(x_n, \mathrm{H}_n) = \|(\mathrm{Id} - T_n)\, x_n\|$, several convergence results can be derived by direct application of Propositions 6–8. In particular, in the case of a single nonexpansive operator T (see (2)), the algorithm below is pertinent.

0	Take $x_0 \in \mathcal{H}$ and set $n = 0$.
1	Take $\lambda_n \in [0, 1]$.
2	Set $x_{n+1} = x_n + \lambda_n(Tx_n - x_n)$.
3	Set $n = n + 1$ and go to step 1.

Fejér Monotonicity in Convex Optimization, Algorithm 3
Fixed point

Proposition 9 *If $\sum_{n\geq 0} \lambda_n(1 - \lambda_n) = +\infty$, any sequence generated by Algorithm 3 converges weakly to a fixed point of T.*

Indeed, the assignments $T_n \leftarrow (\mathrm{Id} + T)/2$ and $\lambda_n \leftarrow 2\lambda_n$ in Algorithm 2 yield Algorithm 3 as T_n is firmly nonexpansive [3,5] and $\mathrm{Fix}\, T_n = \mathrm{Fix}\, T$. Next, observe that $(d(x_n, H_n))_{n\geq 0} = (\|(\mathrm{Id} - T)\, x_n\|/2)_{n\geq 0}$ is nonincreasing by (2). Hence, $\underline{\lim}\, d(x_n, H_n) = 0 \Rightarrow (\mathrm{Id} - T)\, x_n \to 0$ and it results from the demiclosedness of $\mathrm{Id} - T$ [20] that $x_{k_n} \rightharpoonup x \Rightarrow (\mathrm{Id} - T)\, x = 0$. Thus, Proposition 9 follows from Proposition 6ii).

Zeros of Monotone Maps

In connection with set-valued maps $A, B: \mathcal{H} \rightrightarrows \mathcal{H}$ a few definitions and facts need to be recalled [2,27]. First, A is characterized by its graph $\mathrm{gr}\, A = \{(x, u) \in \mathcal{H}^2 : u \in Ax\}$. The inverse A^{-1} of A has graph $\{(u, x) \in \mathcal{H}^2 : (x, u) \in \mathrm{gr}A\}$ and the linear combination $A + \gamma B$ ($\gamma \in \mathbf{R}$) has graph

$$\{(x, u + \gamma v) :\ (x, u) \in \mathrm{gr}\, A, (x, v) \in \mathrm{gr}\, B\}\,.$$

A is monotone if

$$\forall (x, u) \in \mathrm{gr}\, A\ \forall (y, v) \in \mathrm{gr}\, A: \qquad \langle x - y | u - v\rangle \geq 0.$$

If A is monotone and if there exists no monotone map $B \neq A$ such that $\mathrm{gr}\, A \subset \mathrm{gr}B$ then A is *maximal monotone*. In this case

- $\mathrm{gr}\, A$ is weakly-strongly closed: for every sequence $((y_n, v_n))_{n\geq 0}$ in $\mathcal{H}^2$

$$\begin{cases} ((y_n, v_n))_{n\geq 0} \text{ in } \mathrm{gr}\, A \\ y_n \overset{n}{\rightharpoonup} y \\ v_n \overset{n}{\to} v \end{cases} \Rightarrow\ (y, v) \in \mathrm{gr}\, A. \quad (14)$$

- For every $\gamma \in]0, +\infty[$, the resolvent of A, $J_\gamma^A = (\mathrm{Id} + \gamma A)^{-1}$, is a single-valued firmly nonexpansive operator defined on $\mathcal{H}$ [17,23].

Of broad interest is the problem of finding a zero of a maximal monotone map $A: \mathcal{H} \rightrightarrows \mathcal{H}$ [23], i. e.,

$$\begin{cases} \text{Find} & \overline{x} \in \mathcal{H} \\ \text{s.t.} & 0 \in A\overline{x}. \end{cases} \quad (15)$$

For every $\gamma \in]0, +\infty[$, the solution set $S = A^{-1}\, 0$ can be written as $S = \{x \in \mathcal{H} : x \in x + \gamma Ax\} = \mathrm{Fix}\, J_\gamma^A$. Thus, given $(\gamma_n)_{n\geq 0}$ in $]0, +\infty[$, the equilibrium problem (15) can be cast in the form of the common fixed point problem (13) with $(T_n)_{n\geq 0} = (J_{\gamma_n}^A)_{n\geq 0}$. Algorithm 2 is then known as the (relaxed) *proximal point algorithm* [17,23].

0	Take $x_0 \in \mathcal{H}$ and set $n = 0$.
1	Take $\gamma_n \in]0, +\infty[$ and $\lambda_n \in [0, 2]$.
2	Set $x_{n+1} = x_n + \lambda_n(J_{\gamma_n}^A x_n - x_n)$.
3	Set $n = n + 1$ and go to step 1.

Fejér Monotonicity in Convex Optimization, Algorithm 4
Proximal point

Proposition 10 *Suppose that*

$$\begin{cases} (\gamma_n)_{n\geq 0} & \text{is in } [\varepsilon, +\infty[\\ (\lambda_n)_{n\geq 0} & \text{is in } [\varepsilon, 2 - \varepsilon] \end{cases} \text{where } \varepsilon \in]0, 1[\,. \quad (16)$$

Then any sequence generated by Algorithm 4 converges weakly to a zero of A.

This result is a consequence of Proposition 6iii). Indeed, for every $n \in \mathbf{N}$, define $y_n = x_n + (x_{n+1} - x_n)/\lambda_n$, $v_n = (x_n - x_{n+1})/(\gamma_n \lambda_n)$ and note that $v_n \in Ay_n$. Now suppose $d(x_n, H_n) \to 0$. Then, thanks to (16), $x_{n+1} - x_n$

$\to 0$ and, in turn, $v_n \to 0$ and $y_n - x_n \to 0$. Hence, $x_{k_n} \rightharpoonup x \Rightarrow y_{k_n} \rightharpoonup x \Rightarrow 0 \in Ax$ by (14).

Weak convergence can also be achieved under variants of (16), e. g., $\sum_{n\geq 0} \gamma_n^2 = +\infty$ and $\forall\, n \in \mathbf{N}$: $\lambda_n = 1$ [2]. Such results can be deduced from Proposition 6 as well.

Zeros of the Sum of Two Monotone Maps

Take two maximal monotone maps $A, B\colon \mathcal{H} \rightrightarrows \mathcal{H}$. An extension of (15) that captures a wide body of optimization and applied mathematics problems is [27]

$$\begin{cases} \text{Find} & \overline{x} \in \mathcal{H} \\ \text{s.t.} & 0 \in A\overline{x} + B\overline{x}. \end{cases} \tag{17}$$

In instances when $A + B$ is maximal monotone, one can approach this problem via Algorithm 4. Naturally, for this approach to be numerically viable, the resolvents of $A + B$ should be computable relatively easily. A more widely applicable alternative is to devise an *operator splitting algorithm*, in which the operators A and B are employed in separate steps [16]. Two Fejérian splitting algorithms are described below.

First, suppose that B is (single-valued and) *cocoercive* in the sense that $B^{-1} - \alpha\,\mathrm{Id}$ is monotone for some $\alpha \in]0, +\infty[$, i. e.,

$$\forall (x, y) \in \mathcal{H}^2\colon \quad \langle Bx - By | x - y\rangle \geq \alpha \|Bx - By\|^2. \tag{18}$$

Given $\gamma \in]0, 2\alpha]$, it follows from (18) that $\mathrm{Id} - \gamma B$ is nonexpansive. Moreover, the solution set $S = (A + B)^{-1}0$ can be written as $S = \{x \in \mathcal{H}\colon x - \gamma\, Bx \in x + \gamma Ax\} = \mathrm{Fix}\, T$ where $T = J_\gamma^A \circ (\mathrm{Id} - \gamma\, B)$ is nonexpansive as the composition of two nonexpansive operators. Algorithm 3 can then be implemented by alternating a forward step involving B with a backward (proximal) step involving A.

0	Take $\gamma \in]0, 2\alpha]$, $x_0 \in \mathcal{H}$, and set $n = 0$.
1	Set $x_{n+1/2} = x_n - \gamma B x_n$ and take $\lambda_n \in [0, 1]$.
2	Set $x_{n+1} = x_n + \lambda_n (J_\gamma^A x_{n+1/2} - x_n)$.
3	Set $n = n + 1$ and go to step 1.

Fejér Monotonicity in Convex Optimization, Algorithm 5
Forward-backward method

As a corollary of Proposition 9 we obtain:

Proposition 11 *If $\sum_{n\geq 0} \lambda_n(1 - \lambda_n) = +\infty$, any sequence generated by Algorithm 5 converges weakly to a zero of $A + B$.*

The second algorithm is centered around the operator $T = J_\gamma^A \circ (2J_\gamma^B - \mathrm{Id}) + \mathrm{Id} - J_\gamma^B$, where $\gamma \in]0, +\infty[$. This operator possesses two nice properties: it is firmly nonexpansive and $y \in \mathrm{Fix}\, T \Leftrightarrow J_\gamma^B\, y \in (A + B)^{-1}\, 0$ [16]. Whence, by putting $T_n \leftarrow T$ in Algorithm 2, one obtains the *Douglas–Rachford method* [8,16].

0	Take $\gamma \in]0, +\infty[$, $x_0 \in \mathcal{H}$, and set $n = 0$.
1	Set $x_{n+1/2} = J_\gamma^B x_n$ and take $\lambda_n \in [0, 2]$.
2	Set $x_{n+1} = x_n + \lambda_n \big(J_\gamma^A (2x_{n+1/2} - x_n) - x_{n+1/2}\big)$.
3	Set $n = n + 1$ and go to step 1.

Fejér Monotonicity in Convex Optimization, Algorithm 6
Douglas–Rachford method

As in Algorithm 5, B is activated at step 1 and A at step 2. Convergence is established as in Proposition 9

Proposition 12 *If $\sum_{n\geq 0} \lambda_n(2 - \lambda_n) = +\infty$, any sequence generated by Algorithm 6 converges weakly and the image of the weak limit under J_γ^B is a zero of $A + B$.*

Variational Inequalities

Let $B\colon \mathcal{H} \to \mathcal{H}$ be a single-valued maximal monotone operator, let $\varphi\colon \mathcal{H} \to]-\infty, +\infty]$ be a proper, lower-semicontinuous, convex function, and let $\partial\varphi\colon \mathcal{H} \rightrightarrows \mathcal{H}$ be its subdifferential, i. e.,

$$\partial\varphi(x) = \bigcap_{y\in\mathcal{H}} \{u \in \mathcal{H}\colon\ \langle y - x | u\rangle + \varphi(x) \leq \varphi(y)\}.$$

Then $\partial\varphi$ is maximal monotone [2] and, upon taking $A = \partial\varphi$ in (17), one arrives at the *variational inequality* problem

$$\begin{cases} \text{Find} & \overline{x} \in \mathcal{H} \\ \text{s.t.} & \forall x \in \mathcal{H}\colon \\ & \langle \overline{x} - x | B\overline{x}\rangle + \varphi(\overline{x}) \leq \varphi(x). \end{cases} \tag{19}$$

In this context, the resolvent J_γ^A reduces to Moreau's prox mapping [18]

$$\mathrm{prox}_\gamma^\varphi\colon x \mapsto \arg\min_{y\in\mathcal{H}} \varphi(y) + \frac{1}{2\gamma}\|y - x\|^2.$$

As a special instance of (17), the variational inequality problem (19) can be solved via the forward-backward method (Algorithm 5) and Proposition 11 then yields:

Proposition 13 *Suppose that (18) is in force. Take $\gamma \in]0, 2\alpha[$, $x_0 \in \mathcal{H}$, and let*

$$\forall n \in \mathbb{N}: \quad x_{n+1} = x_n + \lambda_n \left(\operatorname{prox}_{\gamma}^{\varphi}(x_n - \gamma B x_n) - x_n\right), \tag{20}$$

where $(\lambda_n)_{n \geq 0}$ is in [0, 1] and $\sum_{n \geq 0} \lambda_n(1 - \lambda_n) = +\infty$. Then $(x_n)_{n \geq 0}$ converges weakly to a solution of (19).

A noteworthy situation is when $\varphi = \iota_Q$, where ι_Q is the indicator function of a nonempty closed convex set Q, i. e.,

$$\iota_Q : x \mapsto \begin{cases} 0 & \text{if } x \in Q, \\ +\infty & \text{if } x \notin Q. \end{cases} \tag{21}$$

It follows that $\partial \iota_Q = \mathrm{N}_Q$, where N_Q is the normal cone to Q, i. e.,

$$N_Q x = \bigcap_{y \in Q} \{u \in \mathcal{H} : \ \langle y - x | u \rangle \leq 0\},$$

if $x \in Q$, and $N_Q x = \emptyset$ otherwise. In addition, (19) reads

$$\begin{cases} \text{Find} & \overline{x} \in Q \\ \text{s.t.} & \forall x \in Q : \ \langle \overline{x} - x | B\overline{x} \rangle \leq 0, \end{cases} \tag{22}$$

and $\operatorname{prox}_{\gamma}^{\iota_Q} = P_Q$ is the projector onto Q.

Differentiable Optimization

A standard convex programming problem is to minimize a proper, lower-semicontinuous, convex function $f: \mathcal{H} \to]-\infty, +\infty]$ over a nonempty closed convex set $Q \subset \mathcal{H}$, i. e.,

$$\text{Find} \quad \overline{x} = \arg\min_{x \in Q} f(x). \tag{23}$$

In view of (21), (23) is equivalent to finding a global minimizer of $\iota_Q + f$, i. e., by Fermat's rule, to finding a zero of $\partial(\iota_Q + f)$. If 0 lies in the interior of $Q - \{x \in \mathcal{H} : f(x) < +\infty\}$, then $\partial(\iota_Q + f) = \partial \iota_Q + \partial f$ [2] and (23) is therefore of the form (17) with $A = N_Q$ and $B = \partial f$. This occurs in particular when f is finite and continuous at a point in Q.

Now suppose that f is differentiable. Then $\partial f = \{\nabla f\}$ is single-valued and (23) can further be reduced to (22) with $B = \nabla f$. The forward-backward scheme (20) then becomes the *projected gradient algorithm*

$$\forall n \in \mathbb{N}: \quad x_{n+1} = x_n + \lambda_n \left(P_Q\left(x_n - \gamma \nabla f(x_n)\right) - x_n\right).$$

Proposition 13 provides conditions for weak convergence to a minimizer of f over Q.

Convex Feasibility Problems

Given a family $(S_i)_{i \in I}$ of intersecting nonempty closed and convex subsets of $\mathcal{H}$, the *convex feasibility problem* reads [3,5,6,15]

$$\text{Find} \quad \overline{x} \in S = \bigcap_{i \in I} S_i. \tag{24}$$

At iteration n, select a nonempty finite index set $I_n \subset I$ and, for every $i \in I_n$, let $p_{i,n}$ be an approximate projection of x_n onto S_i, i. e., the projection of x_n onto a closed affine half-space $H_{i,n}$ containing S_i. Then

$$H_{i,n} = \{x \in \mathcal{H} : \ \langle x - p_{i,n} | x_n - p_{i,n} \rangle \leq 0\}.$$

Let

$$H_n = \left\{ x \in \mathcal{H} : \sum_{i \in I_n} w_{i,n} \langle x - p_{i,n} | x_n - p_{i,n} \rangle \leq 0 \right\}$$

where the weights $(w_{i,n})_{i \in I_n}$ are in $]0, 1]$ and satisfy $\sum_{i \in I_n} w_{i,n} = 1$. Then $S \subset \cap_{i \in I_n} S_i \subset \cap_{i \in I_n} H_{i,n} \subset H_n$ and $P_n x_n = x_n + L_n(x_{n+1/2} - x_n)$, where $x_{n+1/2} = \sum_{i \in I_n} w_{i,n} p_{i,n}$ and

$$L_n = \begin{cases} \dfrac{\sum_{i \in I_n} w_{i,n} \|p_{i,n} - x_n\|^2}{\left\|x_{n+\frac{1}{2}} - x_n\right\|^2} & \text{if } \ x_{n+\frac{1}{2}} \neq x_n \\ 1 & \text{else.} \end{cases} \tag{25}$$

Algorithm 1 then turns into Algorithm 7.

0 | Take $x_0 \in \mathcal{H}$ and set $n = 0$.
1 | Take a nonempty finite set $I_n \subset I$.
2 | Compute approximate projections $(p_{i,n})_{i\in I_n}$ of x_n onto $(S_i)_{i\in I_n}$.
3 | Take $(w_{i,n})_{i\in I_n}$ in $]0, 1]$ such that $\sum_{i\in I_n} w_{i,n} = 1$.
4 | Set $x_{n+1/2} = \sum_{i\in I_n} w_{i,n} p_{i,n}$, L_n as in (25).
5 | Take $\lambda_n \in [0, 2L_n]$.
6 | Set $x_{n+1} = x_n + \lambda_n(x_{n+1/2} - x_n)$.
7 | Set $n = n + 1$ and go to step 1.

Fejér Monotonicity in Convex Optimization, Algorithm 7
Convex feasibility

Weak convergence to a point in S follows from Proposition 6 under various assumptions on the control sequence $(I_n)_{n\geq 0}$ and the approximate projections $((p_{i,n})_{i\in I_n})_{n\geq 0}$ [5,6,15].

Nondifferentiable Optimization

Suppose that f is subdifferentiable in (23), i. e.,

$$\forall x \in \mathcal{H}\colon\ \partial f(x) \neq \emptyset,$$

and that its minimum value $\overline{f}$ over Q is known. Then (23) can be viewed as a special case of (24) with two sets, namely $S_1 = Q$ and $S_2 = \{x \in \mathcal{H}\colon\ f(x) \leq \overline{f}\}$. Now take

$$H_{2,n} = \left\{x \in \mathcal{H}\colon\ \langle x - x_n | u_n\rangle \leq \overline{f} - f(x_n)\right\}$$

where $u_n \in \partial f(x_n)$. Then $S_2 \subset H_{2,n}$ and

$$p_{2,n} = \begin{cases} x_n + \dfrac{\overline{f} - f(x_n)}{\|u_n\|^2}\, u_n & \text{if } x_n \notin S_2 \\ x_n & \text{otherwise} \end{cases}$$

is called a *subgradient projection* of x_n onto S_2 [3,5]. If Algorithm 7 is implemented by alternating a relaxed subgradient projection onto S_2 with an exact projection onto S_1, i. e.,

$$\forall n \in \mathbb{N}\colon\ x_{n+1} = P_Q\left(x_n + \lambda_n\left(p_{2,n} - x_n\right)\right),$$

one obtains the subgradient projection method of [21]. Weak convergence to a solution of (23) under the assumptions of uniform boundedness of ∂f on bounded sets, $(\lambda_n)_{n\geq 0}$ is in $[0, 2]$, and (8), follows from Proposition 6iii [3,5].

Inconsistent Convex Feasibility Problems

When $\cap_{i\in I} S_i = \emptyset$ and I is finite, (24) can be replaced by the minimization problem

$$\text{Find}\quad \overline{x} = \arg\min_{x\in\mathcal{H}} \frac{1}{2}\sum_{i\in I} w_i d(x, S_i)^2 \tag{26}$$

where $(w_i)_{i\in I}$ is in $]0, 1]$ and $\sum_{i\in I} w_i = 1$. Let $(P_i)_{i\in I}$ be the projectors onto $(S_i)_{i\in I}$, let $T = \sum_{i\in I} w_i P_i$, and let S be the solution set of (26). Then T is firmly nonexpansive and $S = \text{Fix } T$ [5]. By reiterating a previous argument, one obtains:

Proposition 14 *Take $x_0 \in \mathcal{H}$, $(\lambda_n)_{n\geq 0}$ in $[0, 2]$ such that $\sum_{n\geq 0} \lambda_n(2 - \lambda_n) = +\infty$, and let*

$$\forall n \in \mathbb{N}\colon\ x_{n+1} = x_n + \lambda_n\left(\sum_{i\in I} w_i P_i x_n - x_n\right).$$

Then $(x_n)_{n\geq 0}$ converges weakly to a solution of (26).

Historical Notes and Comments

In 1922, L. Fejér considered the following problem [12]: given a closed subset $S \subset \mathbf{R}^p$ and a point $y \notin S$ can one find a point $x \in \mathbf{R}^p$ such that

$$\forall \overline{x} \in S\colon\ \|x - \overline{x}\| < \|y - \overline{x}\|.$$

Inspired by this work, T.S. Motzkin and I.J. Schoenberg adopted in their 1954 paper [19] the term Fejér monotone to describe sequences satisfying (1). In this paper (see also [1]), an algorithm was developed to solve systems of linear inequalities in $\mathbf{R}^p$ by successive projections onto the half-spaces defining the polyhedral solution set S. The concept of Fejér monotonicity was shown to be an adequate tool to study convergence of this algorithm. Basic facts such as (5) and (9) can already be found in [19] and [1], respectively.

In the 1960s, I.I. Eremin extended the use of Fejér monotonicity to more general convex problems in Hilbert spaces. A summary of his publications covering the period 1961–1967 is given in [9]. By the end of the 1960s, most results on Fejér monotonicity in Hilbert spaces were essentially known and one can find them scattered in the Soviet literature in the context of specific convex programming problems. Thus, (4) appears in [10], Proposition 2 in [4], Propositions 4 and 5 in [14], and Proposition 8 in [14] and [21]. It should be

noted that Proposition 2 has been implicitly rediscovered many times and that it seems to originate in [24].

Recently, Fejér monotonicity has been reserved a featured role in several convex optimization papers [3,6,15,25,26]. It has also proven a valuable tool in more applied disciplines such as biology, economics, and engineering [5,11]. Some extensions of the notion of Féjer monotonicity are discussed in [7].

See also

- ► Generalized Monotone Multivalued Maps
- ► Generalized Monotone Single Valued Maps
- ► Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems

References

1. Agmon S (1954) The relaxation method for linear inequalities. Canad J Math 6:382–392
2. Aubin J-P, Cellina A (1984) Differential inclusions. Springer, Berlin
3. Bauschke HH, Borwein JM (1996) On projection algorithms for solving convex feasibility problems. SIAM Rev 38:367–426
4. Brègman LM (1965) The method of successive projection for finding a common point of convex sets. Soviet Math Dokl 6:688–692
5. Combettes PL (1996) The convex feasibility problem in image recovery. In: Advances in Imaging and Electron Physics, vol 95. Acad, pp 155–270
6. Combettes PL (1997) Hilbertian convex feasibility problem: Convergence of projection methods. Appl Math Optim 35:311–330
7. Combettes PL (2001) Quasi-Fejérian analysis of some optimization algorithms. In: Butnariu D, Censor Y, Reich S (eds) Inherently Parallel Algorithms for Feasibility and Optimization and Their Applications. Elsevier, Amsterdam
8. Eckstein J, Bertsekas DP (1992) On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math Program 55:293–318
9. Eremin II (1968) Methods of Fejér approximations in convex programming. Math Notes 3:139–149
10. Eremin II (1968) On the speed of convergence in the method of Fejér approximations. Math Notes 4:522–527
11. Eremin II, Mazurov VD (1979) Nonstationary processes of mathematical programming. Nauka, Moscow
12. Fejér L (1922) Uber die Lage der Nullstellen von Polynomen, die aus Minimumforderungen gewisser Art entspringen. Math Ann 85:41–48
13. Genel A, Lindenstrauss J (1975) An example concerning fixed points. Israel J Math 22:81–86
14. Gubin LG, Polyak BT, Raik EV (1967) The method of projections for finding the common point of convex sets. USSR Comput Math Math Phys 7:1–24
15. Kiwiel KC, Łopuch B (1997) Surrogate projection methods for finding fixed points of firmly nonexpansive mappings. SIAM J Optim 7:1084–1102
16. Lions PL, Mercier B (1979) Splitting algorithms for the sum of two nonlinear operators. SIAM J Numer Anal 16:964–979
17. Martinet B (1972) Détermination approchée d'un point fixe d'une application pseudo-contractante. Cas de l'application prox. CR Acad Sci Paris Sér A Math 274:163–165
18. Moreau J-J (1962) Fonctions convexes duales et points proximaux dans un espace Hilbertien. CR Acad Sci Paris Sér A Math 255:2897–2899
19. Motzkin TS, Schoenberg IJ (1954) The relaxation method for linear inequalities. Canad J Math 6:393–404
20. Opial Z (1967) Weak convergence of the sequence of successive approximations for nonexpansive mappings. Bull Amer Math Soc 73:591–597
21. Polyak BT (1969) Minimization of unsmooth functionals. USSR Comput Math Math Phys 9:14–29
22. Raik E (1969) A class of iterative methods with Fejér-monotone sequences. Eesti NSV Tead Akad Toimetised Füü-Mat 18:22–26
23. Rockafellar RT (1976) Monotone operators and the proximal point algorithm. SIAM J Control Optim 14:877–898
24. Schaefer H (1957) Uber die Methode sukzessiver Approximationen. Jahresber Deutsch Math-Verein 59:131–140
25. Schott D (1991) A general iterative scheme with applications to convex optimization and related fields. Optim 22:885–902
26. Schott D (1995) Iterative solution of convex problems by Fejér-monotone methods. Numer Funct Anal Optim 16:1323–1357
27. Zeidler E Nonlinear functional analysis and its applications II/B – Nonlinear monotone operators. Springer, Berlin

Financial Applications of Multicriteria Analysis

CONSTANTIN ZOPOUNIDIS
Department Production Engineering and Management Financial Engineering Lab. Techn., University Crete University Campus, Chania, Greece

MSC2000: 91B06, 91B60

Article Outline

Keywords
Basic Principles of Multicriteria Analysis
Methods
Decision Aid Activity
Multicriteria Character of Financial Problems and Some Real-World Applications
Investment Decision
Portfolio Management
Some Real-World Applications
Venture Capital Investment
Study Context
Multicriteria Method and Results
The Business Failure Risk
Study Context
Multicriteria Method and Results
Concluding Remarks
See also
References

Keywords

Finance; Investment analysis; Multicriteria analysis; Decision support; Applications

The financial decisions of an organization (i. e. firm, bank, insurance company, etc.) are usually considered in the context of optimization. Concerning the case of a firm and for a long term period, one meets two types of decisions: decisions related to the optimal allocation of funds, and decisions related to the optimal financial structure. In the short term, one considers decisions related to the management of working capital, and refers to the optimization of stocks, cash, accounts receivable and short term debts. The financial theory analyzes these decisions (short and long terms), but always from an optimization perspective (for example, theory of cost of capital, portfolio theory, options theory, etc.). This perspective has led some researchers to propose techniques of operations research to solve financial decision problems. The classical modeling of decision problems in operations research consists in formulating an optimization (maximization or minimization) problem under specific constraints. In fact, it is a best choice problem.

However, recently, these financial problems have been examined from a more comprehensive and more realistic perspective which overcomes the restrictive framework of optimization [80,84]. For example, in capital budgeting decision making, K. Bhaskar and P. McNamee [6] pose the following questions:

a) In assessing investment proposals, do the decision makers have a single objective or multiple objectives?
b) If decision makers do have multiple objectives, which are those and what is the priority structure of the objectives?

In another similar study, Bhaskar [5] refers that microeconomic theory has largely adopted a single objective function which is the principle of utility maximization for the consumer unit and profit maximization for firms. To attack the single objective function principle for firms, Bhaskar [5] addresses three categories of criticism:

a) there exist alternatives to the profit maximization approach which are based on equally simple hypotheses and which can better explain reality;
b) the profit maximization or any other equally simple hypothesis is too naive to explain the complex process of decision making;
c) the real-world firms do not have suitable information to enable them to maximize their profits. Furthermore, several other theories of the firm have been postulated which have proposed different objectives than that of the traditional microeconomic theory.

One can cite the revenue maximizing model [3], the manager's utility model [74], the satisficing model [64] and the behavioral models [13].

On the basis of the above remarks it is possible to distinguish three main reasons which have motivated a change of view in the modeling of the financial problems:

1) Formulating the problem in terms of seeking the optimum, financial decision makers (i. e. financial analysts, portfolio managers, investors, etc.) get involved in a very narrow problematic, often irrelevant to the real decision problem.
2) The different decisions (financial ones) are taken by the people (i. e. financial managers) and not by the models; the decision makers get more and more deeply involved in the decision making process and, in order to solve problems, it becomes necessary to take into consideration their preferences, their experiences and their knowledge.

3) For financial decision problems such as the choice of investment projects, the portfolio selection, the evaluation of business failure risk, etc., it seems illusory to speak of optimality, since multiple criteria must be taken into consideration.

In this article, our basic aim is to examine the contribution of the multicriteria analysis to the study and to the solution of some financial decision problems. Section 2 presents the basic principles of multicriteria analysis. The multicriteria character of some financial problems and some real world applications of the multicriteria analysis in the field of financial management are given in section 3. Finally, some discussion and the advantages which resulted by the application of multicriteria analysis in the field of financial management, are given in section 4.

Basic Principles of Multicriteria Analysis

Multicriteria analysis, often called *multiple criteria decision making* (MCDM) by the American School and *multicriteria decision aid* (MCDA) by the European School, is a set of methods which allow the aggregation of several evaluation criteria in order to choose one or more alternatives (i. e. investment projects, financial assets at variable revenue, financial assets at fixed revenue, dynamic firms, etc.). It also deals with the study of the activity of decision aid to a well-identified decision maker (i. e. individual, firm, organization, etc.).

The development of multicriteria decision aid (hence we use this term in the text) began in 1971. Its principal objective is to provide the decision maker with tools in order to enable him to advance in solving a decision problem (for example, the selection of investment projects for a firm), where several, often conflicting multiple criteria must be taken into consideration.

Methods

The specialists in the field distinguish several categories of methods in MCDA. The boundaries between these categories are, of course, rather fuzzy. B. Roy [58] proposes the following three categories of methods:

1) unique synthesis criterion approach disregarding any incomparability;
2) outranking synthesis approach, accepting incomparability; and
3) interactive local judgement approach with trial-error iterations.

In this paper, the classification proposed in [53] is adopted. It distinguishes four categories:

1) multi-objective mathematical programming;
2) multi-attribute utility theory;
3) outranking relations approach; and
4) preference disaggregation approach.

Multi-objective mathematical programming is characterized by the fact that an action (or alternative) a is represented by a vector of real variables $(x_1, \ldots, x_l)$. The set A of the feasible solutions is defined by a set of linear constraints: $A = \{x \in \mathbf{R}^l: A \cdot X \leq b, x \geq 0\}$ with A a matrix of dimensions $m \times l$ and b a vector-matrix $m \times 1$. The chosen vector must give satisfaction to relatively several numerical criteria, m in number, and noted as $C^1, \ldots, C^m$, which are linear functions of x. It is possible to distinguish three different methods inside this approach:

1) the efficient solutions procedure;
2) the goal programming;
3) the compromise programming.

A synthesis of the studies realized on this category of methods can be found in [69,72] and [77].

Multi-attribute utility theory (MAUT) is an extension of the classical utility theory. It seeks to give a representation of the preferences of a decision maker by means of a utility function, aggregating several evaluation criteria: $u(\mathbf{g}) = u(g_1, \ldots, g_n)$. In other words, the problem is to choose the action a which maximizes the utility function of the decision maker: $u[\mathbf{g}(a)] = \max u[\mathbf{g}(a)]$.

The criteria (attributes) can be certain or probabilistic (each $g_i(a)$ is associated with a probability distribution). In general, one can decompose a multicriteria utility function in real functions $u_1, \ldots, u_n$ concerning the independence of criteria. Thus, different utility function models are obtained. The most studied form of utility function, from a theoretical point of view, is the additive form:

$$u(g_1, \ldots, g_n) = u_1(g_1) + \cdots + u_n(g_n),$$

where $u_1, \ldots, u_n$ are the marginal utilities defined on the scales of criteria. For the study of the condition of independence in utility between criteria (substitution rate), one can refer to [34]. The latter and [77] present syntheses of works on the construction of multicriteria

utility functions both, under certainty and under uncertainty.

The *outranking relations* approach was developed in Europe with the elaboration of the ELECTRE methods (ELimination Et Choix Traduisant la REalité). The concept of outranking in ELECTRE methods is due to Roy, who is the founder of these methods. The outranking relation allows to conclude that an action $a \in A$ (discrete set) outranks an action $b \in A$ if there are enough arguments to confirm that a is at least as good as b, while there is no essential reason to refute this statement. In the ELECTRE methods the aggregation of criteria requires to define the threshold notions of preference and indifference, concordance and discordance. In fact, a outranks b if there exists a sufficient majority of criteria for which a is better classified than b (concordance) and if the unfavorable deviations for the rest of the criteria (discordance) are not too high. Thus, this modeling can bring into evidence the cases of incomparability when the multicriteria evaluation of two actions is very differentiated. A detailed presentation of all outranking methods can be found in [61,63] and [72].

The approach of the *disaggregation of preferences* is often used in MCDA as a mean for the modeling of the preferences of a decision maker or a group of decision makers. This approach uses the regression methods. The introduction of regression methods in MCDA is effected because of the development of the social judgement theory. Multiple regression can, in general, detect, identify or 'capture' the judgement policy of a decision maker (i. e. disaggregation of the preferences). This one, particularly if it is in relation with a certain number of past decisions, might be the expression of a global preference. The approach by multiple regression is quite close to the MAUT; their differences are placed at the level of obtaining the marginal utilities $u_i(g_i)$ and the weights p_i. For example, for the additive utility function:

$$u(\mathbf{g}) = \sum_i p_i u_i(g_i),$$

the marginal utilities $u_i(g_i)$ and the weights p_i are obtained by direct interrogation of the decision maker (aggregation methods) as far as it concerns the MAUT approach, and by indirect interrogation of the decision maker (disaggregation methods) as far as it concerns the multiple regression approach. The principal drawback which prevents the closeness of the two approaches is related to the linearity of the models proposed by multiple linear regression. A rather exhaustive bibliography of the methods of the disaggregation of preferences can be found in [32] and [53].

Decision Aid Activity

Concerning the activity of decision aid, Roy [58,60] proposes a methodology of systematic intervention of multicriteria analysis in the decision process. In brief, this methodology comprises four levels:

I) Object of the decision and spirit of recommendation or participation.
II) Analyzing consequences and developing criteria.
III) Modeling comprehensive preferences and operationally aggregating performances.
IV) Investigating and developing the recommendation.

It is important to emphasize that these four levels do not necessarily follow one another in the above mentioned order. The activity of decision aid does not necessarily constitute a sequential process; interactions between the decision maker and the analyst can occur. This general methodology has contributed to the development of several multicriteria methods which have been applied successfully to real cases. Among these methods the well-known are the ELECTRE methods developed by Roy and his collaborators.

Multicriteria Character of Financial Problems and Some Real-World Applications

The operational research techniques were the first to be used in the solution of some financial problems. I. Ekeland [19] wonders

> why finance, rather curiously, has remained so long away from the techniques of operational research (i. e. optimization techniques), except for those concerning portfolio selection models.

According to the same author, the Capital Asset Pricing Model (CAPM) is a static optimization model based on the principle according which, the best portfolio (i. e. optimal portfolio) is the one which maximizes the expected return for a given level of risk, in the period of time considered. For R.W. Ashford et al. [2], the techniques of operational research can be applied to

working capital management as well as to the evaluation of investment projects. Among the techniques used for the management of working capital, one could mention:

- inventory control for the management of stocks;
- dynamic programming, linear programming, stochastic programming and visual and interactive techniques of simulation for the management of cash;
- the Markov process and the discriminant analysis for the management of accounts receivable;
- dynamic programming, linear programming, and stochastic programming for the management of short-term debts (current liabilities).

Among the techniques used in the evaluation of investment projects, one could mention the simulation methods [23] and those of mathematical statistics [24] which take into consideration the risk factor. Simulation methods and linear programming (i. e. the LONGER program, [51]) are also used in financial planning (i. e. elaboration of investment and financing plans). Under these circumstances, the solution of financial problems is easy to obtain. It is based on the fact that the problem is well posed, well-formulated regarding the reality involved and on an evaluation criterion (i. e. monocriteria paradigm). But in reality, the modeling of financial problems is based on a different kind of logic. In that case, their solution should take into consideration the following elements (i. e. multicriteria paradigm, cf. [59]):

- multiple criteria;
- conflict situation between the criteria;
- complex evaluation process, subjective and ill-structured;
- introduction of financial decision makers in the evaluation process.

MCDA has already contributed in a significant manner to the solving of several financial problems such as venture capital investment, business failure risk, credit granting, bond rating, country risk, political risk, evaluation of the performance and viability of organizations, choice of investments, financial planning and portfolio management.

The multicriteria character of these financial problems can be easily demonstrated. We will limit here the analysis on the choice of investment projects and portfolio management. International literature could actually provide very important case studies for the rest of the financial problems [36,80,84].

Investment Decision

The choice of investment projects entails an important decision for every firm, public or private, large or small one. In fact, considering its duration, its amount and its irreversible character an investment decision is regarded as a major and strategic one. Therefore, the process of an *investment decision* should be conveniently modeled. If one considers that, in principle, the investment decision process consists of four main stages: perception, formulation, evaluation and choice, the financial theory intervenes only in the stages of evaluation and choice [8]. With its empirical financial criteria (i. e. the payback method, the accounting rate of return) and sophisticated ones, based on discount techniques (i. e. the net present value, the internal rate of return, the index of profitability, the discounted payback method, etc.), the financial theory proposes either a ranking from the better to worst when there are many investment projects in competition or an acceptance or refusal if there is only one investment project. Although the tools of the financial theory should be improved so that they could take into account time, inflation and risk (i. e. analytical methods, simulation methods, games theory, CAPM, etc.), there are still problems concerning the evaluation and selection of investment projects. Among the most important ones, one could mention the reduction of the investment notion in a time series of monetary flows (i. e. inflows, outflows), the choice of the discount rate, the conflicts between financial criteria (i. e. net present value versus internal rate of return), etc. According to the financial theory, the discount rate (sometimes rate of return) plays the role of acceptance or rejection rate (a cut off rate) of an investment project in the case where the criterion of internal rate of return is used. Thus, one can see that the investment decision of a firm depends on one variable only, which is the discount rate. As far as the conflicts between criteria are concerned, one often ascertain that the criteria which are supposed to express the goal of the profitability of projects, could lead to divergent rankings (for example, the net present value and the index of profitability or even the net present value and the internal rate of return). In consequence,

the financial approach of investment decision seems limited and unrealistic. It is limited because it remains in the stages of evaluation and choice, and it is unrealistic because it is based only on financial criteria.

MCDA, on the other hand, contributes in a very original way to the investment decision process. Initially, it intervenes in the whole process of investment, from the stages of perception and formulation to the stages evaluation and choice. Concerning the stages of perception and formulation, MCDA contributes to the identification of possible actions (i. e. investment opportunities) and to the definition of a set of potential actions (i. e. possible variants, each variant constituting an investment project in competition with others). This set of projects can be global, fragmented, stable or evolutionary. Then, it is necessary to choose a reference problematic which is well-adapted to the investment decision problem (i. e. choice, sorting, ranking).

- Choice problematic $P.\alpha$: help in choosing the best investment project or in developing a selection procedure for investment projects.
- Sorting problematic $P.\beta$: help in sorting investment projects according to norms or in building an assignment procedure for investment projects.
- Ranking problematic $P.\gamma$: help in ranking the investment projects according to a decreasing preference order or in building an ordering procedure for investment projects.

Concerning the stages of evaluation and choice, MCDA offers a methodological framework much more realistic than the financial theory, by introducing in the study of investment projects both quantitative and qualitative criteria. Criteria such as the urgency of the project, the coherence of the objectives of the projects with those of the general policy of the firm [21], the social and environmental aspects should be taken into consideration in an investment decision. Therefore, MCDA contributes to show the best investment projects according to the problematic chosen, to solve the conflicts between criteria satisfactorily, to set up the relative importance of criteria in the decision making process and to make known the preferences and the investors' system of values. It is very interesting to mention that many authors have already used MCDA methods in the evaluation of investment projects (list non exhaustive): ELECTRE II and ORESTE methods [14]; MAUT methods [21]; multi-objective mathematical programming [5,35,41]; the Analytic Hierarchy Process (AHP) method [38]; PROMETHEE method [55,81].

Finally, in order to examine if the firms apply in reality multiple criteria in their investment decisions, we present the results of the empirical study of Bhaskar and McNamee [6]. The two authors, by studying large United Kingdom companies, have shown that most companies appear to have more than one objective when an investment is being appraised (96%). The most common number of objectives that companies had was eight. Concerning the objectives priority, most companies (77%) had profitability as the primary objective. The next most important objective was company's growth. Other criteria less important than the two above but, which play a role in the investment decisions are the risk, the liquidity, the environment, the age of assets, the flexibility, the depth of skills, etc. With these empirical results an answer has been given to the questions posed in the introduction by the two authors.

Portfolio Management

In the field of *portfolio management* it is possible to cite the pioneering work of H.M. Markowitz [46] who, by developing the optimization model mean-variance (M-V), is the founder of the classical approach of the portfolio management. According to [19], the problem of portfolio choice in the model (M-V) is a multicriteria one, because the investor will try simultaneously to maximize the return and minimize the risk; but determining the acceptance level of risk, one comes back to maximize the return, which is a classical monocriteria problem. After this bicriteria, and even more the monocriteria (i. e. market model, CAPM) portfolio choice consideration, the development of multifactor models has been started where there are more types of risk and not only market risk [57]. Thus, the problem of portfolio selection becomes multidimensional. The necessity of having multidimensional methods (i. e. statistics and econometrics) for the selection of stocks has been presented by specialist researchers in finance [33]. The multidimensional nature of risk in portfolio management has also been demonstrated by specialist researchers in multicriteria analysis. See [76,77] and [10] on the 'Prospect Ranking Vector (PRV)' method. Today an arsenal of multidimensional and multicriteria methods such as factor analysis,

goal programming, AHP, ELECTRE, MINORA, ADELAIS, etc. have been already applied in the field of portfolio management [9,25,27,28,29,37,40,47,48,62,82,86].

The multicriteria nature of the problem of portfolio selection is well presented in [37]. The authors study the problem of the international portfolio selection. According to them, the classical optimization model of portfolio selection used in a national context can have even more chance of being sub-optimal in a situation of international diversification. In fact, in an international context, the M-V model does not always constitute a suitable method because, it does not incorporate all the criteria that the portfolio managers and investors use in their stock investment decisions. For such decisions, the authors propose new criteria such as: the return of the last five years on a monthly basis, the standard deviation of the return calculated on the last five years, the total cost of transactions, the country risk (or political risk), the direct available coverage for foreign currencies and the exchange risk. The multicriteria methodology used (i. e. ELECTRE IS, ELECTRE III) has the advantage of offering the portfolio manager a large set of investment opportunities, and also gives him the flexibility of choosing the relative importance of the different criteria during the process of portfolio selection. Finally, the authors believe that the use of an optimization model under constraints changes the nature of the portfolio selection problem because a constraint does not play the same role as a criterion in all decision problems. To show this new direction of research in portfolio management, it is convenient to mention the special issue of the Canadian journal 'L' Actualité Economique}, which is dedicated on the contribution of multicriteria analysis in the study of financial markets [36].

Some Real-World Applications

In this paragraph two applications of MCDA are briefly presented. The first one concerns the evaluation of the venture capital investment and, the second one the evaluation of the business failure risk.

Venture Capital Investment

Venture capital constitutes today an important source of financing for small and medium size firms. It plays, also, an interesting role in the development of the business' spirit. The crucial problem for *venture capital investment* is the choice of evaluation criteria and their aggregation in a global operational model, which will serve as a basis for the rational and automatic selection of viable firms. The earlier evaluation models (i. e. descriptive and statistical) can not explain the investment decisions in venture capital, since the latter relies much more on subjective and qualitative elements than on objectives and quantitative ones [83]. Moreover, the complexity of the evaluation of venture capital investment problem has been mentioned in the evaluation procedures of projects by two French venture capital firms [80].

Study Context

The data sample coming from two French venture capital firms, IDI and SIPAREX, was used as the application object of MCDA. Although these two firms use project evaluation procedures, their problem remains that of the absence of a model able of supporting their decisions in venture capital investment. In fact, the variables used in the evaluation procedure are both financial variables (i. e. profitability ratios, solvency ratios, liquidity ratios, etc.), and qualitative variables (i. e. market trend, information security, quality of management, market niche/position, etc.). But, although there are, in both venture capital firms, techniques for the treatment of financial variables, there is no explicit model for the elaboration and modeling of the qualitative variables. Therefore, it is at this stage of analysis that the evaluation problem becomes complex. Moreover, the complexity of the evaluation of venture capital investment problem is also underlined in other studies [18,26,54,71,83] among others). The role of the venture capitalist goes beyond that of the simple contributor to the funds of the firm.

Multicriteria Method and Results

The multicriteria system MINORA (Multicriteria INteractive Ordinal Regression Analysis) was proposed for the evaluation of firms to the two venture capital firms. It belongs to the fourth category of MCDA methods, which is the approach of the disaggregation of preferences. The MINORA system is both based on the iterative utilization of an ordinal regression method and on an appropriate man-machine dialogue. Its aim

is to construct multicriteria decision models which are as consistent as possible with the judgement policy of a decision maker. The decision maker (here the venture capitalist) expresses his judgement policy by ranking some firms, among those he knows well on the basis of previous decisions. The system, then, by the use of the ordinal regression method UTA (UTilitś Additives [31], estimates optimally the additive utility function(s), on multiple criteria, which is (are) as consistent as possible with the decision maker's ranking. The utility function model is estimated iteratively and interactively. It allows, first, the aggregation of all the criteria (i. e. financial and/or qualitative) by giving their relative importance, and second, the automatic and global evaluation of each firm. With the help of the decision makers of the two venture capital firms, two evaluation models were elaborated (one for each venture capital firm). This paper presents only the global model of IDI.

- The evaluation model for IDI

IDI evaluates firms for financing according to twelve criteria. The utility function model was then estimated in the fourth stage of interaction and appeared perfectly consistent with the objectives of IDI. The equation for the global model is the following:

$$\begin{aligned} u(\mathbf{g}) = {} & 0.008u_1(g_1) \\ & + 0.072u_2(g_2) + 0.006u_3(g_3) + 0.197u_4(g_4) \\ & + 0.105u_5(g_5) + 0.232u_6(g_6) + 0.009u_7(g_7) \\ & + 0.094u_8(g_8) + 0.047u_9(g_9) + 0.071u_{10}(g_{10}) \\ & + 0.097u_{11}(g_{11}) + 0.062u_{12}(g_{12}), \end{aligned}$$

where the evaluation criteria are the following:

g_1) the sensitivity of sales to the inflation rate;
g_2) the sensitivity of value added to the sales variations;
g_3) the sensitivity of labor productivity (value added per capita) to wage cost increase (wage per capita);
g_4) the supplier credit in days;
g_5) the available net income;
g_6) the quality of management;
g_7) the research and development effort;
g_8) the extent of diversification;
g_9) the market trend;
g_{10}) the market niche/position;
g_{11}) the cash-out method (opportunities for exit);
g_{12}) the world market share.

The model described above is the best adapted to express the preferences, the knowledge and the experiences of the venture capitalist concerning the quality of the firms and their final evaluation. A detailed presentation of the multicriteria method and the results of the application in the two venture capital firms IDI and SIPAREX can be found in [80].

The Business Failure Risk

According to a general definition, failure is the situation that a firm cannot pay lenders, preferred stock shareholders, suppliers, etc., or a bill is overdrawn, or the firm is bankrupt according to law. Today, there is a complete arsenal of evaluation methods for the *business failure risk* [16]. Since the late 1980s, methods close to a qualitative definition of business failure have been developed. These are multicriteria methods which present undeniable advantages in matter of evaluation for the business failure risk [84].

Study Context

The study concerns the evaluation of failure risk of firms financed by a Greek bank of industrial development. This bank finances with stock equity and long term credit the development of Greek firms and contributes to the renovation of industrial and commercial firms on a national and regional level. As in the previous case of the venture capital investment, there is no model able to provide help to the bank credit managers in the financing of firms.

Multicriteria Method and Results

ELECTRE TRI method was proposed for the evaluation of business failure risk, which is particularly suitable for multicriteria sorting problems. It belongs to the third category of MCDA methods, which is the approach of outranking relations [61,75]. From a finite set of actions (i. e. firms) evaluated by quantitative and/or qualitative criteria and from a set of categories previously defined (i. e. reference actions or reference profiles), ELECTRE TRI proposes two different procedures of assignment which allow the classification of all the actions in these categories. In consequence, ELECTRE TRI consists of establishing an outranking relation between the actions to be assigned and the reference profiles. The eventual

differences between the two assignment procedures, the pessimistic and the optimistic one, come from the incomparability situations between an action and one or several reference profiles [17,75].

For the case of the evaluation of the business failure risk, three categories of risk were determined by the credit managers of the Greek bank:

C_1) the failed firms (9 in number);
C_2) the risky firms; uncertain category of firms to be studied further (10 in number);
C_3) the healthy firms (20 in number).

These 39 firms were evaluated by seven criteria, five financial ratios and two strategic criteria. The criteria are the following:

x_1) Earnings before interests and taxes/Total assets,
x_2) Net income/Stockholder's equity,
x_3) Total debts/Total assets,
x_4) Financial expenses/Sales,
x_5) Administrative and general expenses/Sales,
x_6) Managers work experience,
x_7) Market niche/position.

From the reference profiles and the thresholds of discrimination (preference model established by the credit managers of the bank), ELECTRE TRI provided good percentages of classification, which were of the order of 82.05% and 89.74% for the optimistic and the pessimistic procedures respectively. The pessimistic procedure gave better results and proved more adaptable to the problem of evaluation of business failure risk (it did not give serious classification errors of the type $C_1 \to C_3$ or $C_3 \to C_1$). For a detailed presentation of the multicriteria method and the results, see [84].

Concerning other financial problems which present a multicriteria character and on which a MCDA method has been applied, it is possible to provide a list of published works (non exhaustive).

- Acquisitions of firms: [68].
- Bankruptcy risk: [1,17,67,78,79].
- Country risk: [7,11,12,49,52,70].
- Evaluation of performance of organizations
  - Insurance: [45].
  - Banks: [43,44,85].
  - Firms: [4,15,30,39,42,66,87,88].
- Financial planning: [20,22,73].
- Venture capital: [50,56,65].

Concluding Remarks

This article has shown the contribution of the MCDA to the solution of some financial decision problems (i. e. venture capital, business failure risk, investment choice, portfolio management, etc.). In the past, all these problems were approached with the use of financial theory in a very narrow framework, that of optimization. Some researchers took advantage of the optimal character of these problems in order to propose operational research techniques (i. e. classical or monocriteria modeling) for their solution. The use of MCDA methods provides many advantages in financial management, among which one could mention the following:

- the possibility of structuring complex evaluation problems;
- the introduction of both quantitative (i. e. financial ratios) and qualitative criteria in the evaluation process;
- the transparency in the evaluation, allowing good argumentation in financial decisions;
- the introduction of sophisticated scientific methods in the field of financial management.

In conclusion, MCDA methods seem to have a promoting future because they offer a highly methodological and realistic framework to decision problems.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Competitive Ratio for Portfolio Management
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Optimization
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties

- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling
- ▶ Robust Optimization
- ▶ Semi-infinite Programming and Applications in Finance

References

1. Andenmatten A (1995) Evaluation du risque de défaillance des emetteurs d'obligations: Une approche par l'aide multicritère à la décision. Press. Polytechniques Univ. Romandes
2. Ashford RW, Berry RH, Dyson RG (1988) Operational research and financial management. Europ J Oper Res 36:143–152
3. Baumol WJ (1959) Economic theory and operations analysis. Prentice-Hall, Englewood Cliffs, NJ
4. Bergeron M, Martel JM, Twarabimenye P (1996) The evaluation of corporate loan applications based on the MCDA. J Euro–Asian Management 2(2):16–46
5. Bhaskar K (1979) A multiple objective approach to capital budgeting. Accounting and Business Res Winter, pp 25–46
6. Bhaskar K, Mcnamee P (1983) Multiple objectives in accounting and finance. J Business Finance and Accounting 10(4):595–621
7. Clei J (1994) La méthodologie d'analyse de la Coface. Banque Stratégie 109:5–6
8. Colasse B (1993) Gestion financière de l'entreprise. Press. Univ. France, Paris
9. Colson G, De Bruyn Ch (1989) An integrated multiobjective portfolio management system. Math Comput Modelling 12(10-11):1359–1381
10. Colson G, Zeleny M (1979) Uncertain prospects ranking and portfolio analysis under the condition of partial information. Math Syst in Economics 44
11. Cook WD, Hebner KJ (1993) A multicriteria approach to country risk evaluation: With an example employing Japanese data. Internat Rev Economics and Finance 2(4):327–438
12. Cosset JC, Siskos Y, Zopounidis C (1995) Evaluating country risk: A decision support approach. Global Finance J 3(1):79–95
13. Cyert RM, March JG (1963) A behavioral theory of the firm. Prentice-Hall, Englewood Cliffs, NJ
14. Danila N (1980) Méthodologie d'aide à la décision dans le cas d'investissements fort dépendants. Thèse de Doctorat de 3e Cycle, UER Sci des Organisations, Univ Paris-Dauphine
15. Diakoulaki D, Mavrotas G, Papagyannakis L (1992) A multicriteria approach for evaluating the performance of industrial firms. OMEGA Internat J Management Sci 20(4):467–474
16. Dimitras AI, Zanakis SH, Zopounidis C (1996) A survey of business failures with an emphasis on prediction methods and industrial applications. Europ J Oper Res 90:487–513
17. Dimitras AI, Zopounidis C, Hurson CH (1995) A multicriteria decision aid method for the assessment of business failure risk. Found Computing and Decision Sci 20(2):99–112
18. Dixon R (1991) Venture capitalists and the appraisal of investments. OMEGA Internat J Management Sci 19(5):333–344
19. Ekeland I (1993) Finance: Un nouveau domaine des mathématiques appliquées. Revue Franc de Gestion, pp 44–48
20. Eom HB, Lee SM, Snyder ChA, Ford FN (1987/8) A multiple criteria decision support system for global financial planning. J Management Information Systems 4(3):94–113
21. Evrard Y, Zisswiller R (1982) Une analyse des décisions d'investissement fondée sur les modèles de choix multiattributs. Finance 3(1):51–68
22. Goedhart MH, Spronk J (1995) Financial planning with fractional goals. Europ J Oper Res 82:111–124
23. Hertz DB (1964) Risk analysis in capital investment. Harvard Business Rev 42:95–106
24. Hillier FS (1963) The derivation of probabilistic information for the evaluation of risky investments. Managem Sci 9:443–457
25. Ho PC, Paulson AS (1980) Portfolio selection via factor analysis. J Portfolio Management, pp 27–30
26. Hoban JP (1976) Characteristics of venture capital investments. Unpublished Doctoral Diss, Univ Utah
27. Hui TK, Kwan E (1994) Internat. portfolio diversification: A factor analysis approach. OMEGA Internat J Management Sci 22(3):263–267
28. Hurson CH, Zopounidis C (1995) On the use of multicriteria decision aid methods to portfolio selection. J Euro–Asian Management 1(2):69–94
29. Hurson CH, Zopounidis C (1997) Gestion de portefeuille et analyse multicritère. Economica, Paris
30. Jablonsky J (1993) Multicriteria evaluation of clients in financial houses. Central Europ J Oper Res and Economics 3(2):257–264
31. Jacquet-Lagrèze E, Siskos J (1982) Assessing a set of additive utility functions for multicriteria decision making: The UTA method. Europ J Oper Res 10:151–164
32. Jacquet-Lagrèze E, Siskos J (1983) Méthodes de décision multicritère. Ed. Hommes et Techniques
33. Jacquillat B (1972) Les modèles d'évaluation et de sélection des valeurs mobilières: Panorama des recherches américaines. Analyse Financière 11(4):68–88
34. Keeney RL, Raiffa H (1976) Decisions with multiple objectives: Preferences and value tradeoffs. Wiley, New York

35. Khorramshahgol R, Okoruwa AA (1994) A goal programming approach to investment decisions: A case study of fund allocation among different shopping malls. Europ J Oper Res 73:17–22
36. Khoury NT, Martel JM (1993) Nouvelles orientations dans l'étude des marchés financiers: Asymétrie d'information et analyse multicritère. L'Actualité Economique, Revue d'Analyse Economique 69(1):5–7
37. Khoury NT, Martel JM, Veilleux M (1993) Méthode multicritère de sélection de portefeuilles indiciels internationaux. L'Actualité Economique, Revue d'Analyse Economique 69(1):171–190
38. Kivijärvi H, Tuominen M (1992) A decision support system for semistructured strategic decisions: A multi-tool method for evaluating intangible investments. Revue des Systèmes de Décision 1:353–376
39. Lee H, Kwak W, Han I (1995) Developing a business performance evaluation system: An analytic hierarchical model. The Engineering Economist 30(4):343–357
40. Lee SM, Chesser DL (1980) Goal programming for portfolio selection. J Portfolio Management, pp 22–26
41. Lin WT (1978) Multiple objective budgeting models: A simulation. Accounting Rev LIII(1):61–76
42. Mareschal B, Brans JP (1993) BANKADVISER: Un système interactif multicritère pour l'évaluation financière des entreprises à l'aide des méthodes PROMETHEE. L'Actualité Economique, Revue d'Analyse Economique 69(1):191–205
43. Mareschal B, Mertens D (1990) Evaluation financière par la méthode multicritère GAIA: Application au secteur bancaire belge. Revue de la Banque 6:317–329
44. Mareschal B, Mertens D (1992) BANKS: A multicriteria, PROMETHEE-based, decision support system for the evaluation of the international banking sector. Revue des Systèmes de Décision 1(2-3):175–189
45. Mareschal B, Mertens D (1993) Evaluation multicritère par la méthode multicritère GAIA: Application au secteur de l'assurance en Belgique. L'Actualité Economique, Revue d'Analyse Economique 69(1):206–228
46. Markowitz H (1952) Portfolio selection. J Finance, pp 77–91
47. Martel JM, Khoury NT, Bergeron M (1988) An application of a multicriteria approach to portfolio comparisons. J Oper Res Soc 39(7):617–628
48. Martel JM, Khoury NT, M'zali B (1991) Comparaison performance-taille des fonds mutuels par une analyse multicritère. L'Actualité Economique, Revue d'Analyse Economique 67(3):306–324
49. Mondt K, Despontin M (1986) Evaluation of country risk using multicriteria analysis. Paper presented at the EURO VIII Conf
50. Muzyka D, Birley S, Leleux B (1996) Trade-offs in the investment decisions of Europ. venture capitalists. J Business Venturing 11:273–287
51. Myers SC, Pogue GA (1974) A programming approach to corporate financial management. J Finance 29:579–599
52. Oral M, Kettani O, Cosset JC, Daouas M (1992) An estimation model for country risk rating. Internat J Forecasting 8:583–593
53. Pardalos PM, Siskos Y, Zopounidis C (1995) Advances in multicriteria analysis. Kluwer, Dordrecht
54. Poindexter JB (1976) The efficiency of financial markets: The venture capital case. Unpublished Doctoral Diss New York Univ
55. Ribarovic Z, Mladineo N (1987) Application of multicriterional analysis to the ranking and evaluation of the investment programmes in the ready mixed concrete industry. Engin Costs and Production Economics 12:367–374
56. Riquelme H, Rickards T (1992) Hybrid conjoint analysis: An estimation probe in new venture decisions. J Business Venturing 7:505–518
57. Ross SA (1976) The arbitrage theory of capital asset pricing. J Econom Theory, pp 343–362
58. Roy B (1985) Méthodologie multicritère d'aide à la décision. Economica, Paris
59. Roy B (1988) Des critères multiples en recherche opérationnelle: Pourquoi? In: Rand GK (ed) Operational Research '87. Elsevier, Amsterdam, pp 829–842
60. Roy B (1996) Multicriteria methodology for decision aiding. Kluwer, Dordrecht
61. Roy B, Bouyssou D (1993) Aide multicritère à la décision: Méthodes et cas. Economica, Paris
62. Saaty TL, Rogers PC, Pell R (1980) Portfolio selection through hierarchies. J Portfolio Management Spring:16–21
63. Schärlig A (1996) Pratiquer ELECTRE et PROMETHEE. Press. Polytechniques Univ. Romandes
64. Simon HA (1957) Models of man. Wiley, New York
65. Siskos J, Zopounidis C (1987) The evaluation criteria of the venture capital investment activity: An interactive assessment. Europ J Oper Res 31:304–313
66. Siskos Y, Zopounidis C, Pouliezos A (1994) An integrated DSS for financing firms by an industrial development bank in Greece. Decision Support Systems 12:151–168
67. Slowinski R, Zopounidis C (1995) Application of the rough set approach to evaluation of bankruptcy risk. Internat J Intelligent Systems in Accounting, Finance, and Management 4:27–41
68. Slowinski R, Zopounidis C, Dimitras AI (1997) Prediction of company acquisition in Greece by means of the rough set approach. Europ J Oper Res 100:1–15
69. Spronk J (1981) Interactive multiple goal programming application to financial planning. Martinus Nijhoff, Boston
70. Tang JCS, Espinal CG (1989) A model to assess country risk. OMEGA Internat J Management Sci 17(4):363–367
71. Tyebjee TT, Bruno AV (1984) A model of venture capitalist investment activity. Managem Sci 30(9):1051–1066
72. Vincke Ph (1992) Multicriteria decision-aid. Wiley, New York
73. Vinso JD (1982) Financial planning for the multinational corporation with multiple goals. J Internat Business Stud, pp 43–58

74. Williamson OE (1964) The economics of discretionary behavior: Managerial objectives in a theory of the firm. Prentice-Hall, Englewood Cliffs, NJ
75. Yu W (1992) Aide multicritère à la décision dans le cadre de la problématique du tri: Concepts, méthodes et applications. Thèse de Doctorat, Univ Paris-Dauphine
76. Zeleny M (1977) Multidimensional measure of risk: The prospect ranking vector. In: Zionts S (ed) Multiple Criteria Problem Solving. Springer, Berlin, pp 529–548
77. Zeleny M (1982) Multiple criteria decision making. McGraw-Hill, New York
78. Zollinger M (1982) L'analyse multicritère et le risque de crédit aux entreprises. Revue Franc de Gestion Jan.-Fév.:56–66
79. Zopounidis C (1987) A multicriteria decision-making methodology for the evaluation of the risk of failure and an application. Found Control Eng 12(1):45–67
80. Zopounidis C (1990) La gestion du capital-risque. Economica, Paris
81. Zopounidis C (1993) A multicriteria methodology for the evaluation and ranking of investment projects. Bull Greek Banks Assoc 10(39-40):22–28. (In Greek)
82. Zopounidis C (1993) On the use of the MINORA multicriteria decision aiding system to portfolio selection and management. J Inform Sci and Techn 2(2):150–156
83. Zopounidis C (1994) Venture capital modelling: Evaluation criteria for the appraisal investments. The Financier: ACMT 1(2):54–64
84. Zopounidis C (1995) Evaluation du risque de défaillance de l'entreprise: Méthodes et cas d'application. Economica, Paris
85. Zopounidis C, Despotis DK, Stavropoulou E (1995) Multiattribute evaluation of Greek banking performance. Applied Stochastic Models and Data Analysis 11(1):97–107
86. Zopounidis C, Godefroid M, Hurson Ch (1995) Designing a multicriteria decision support system for portfolio selection and management. In: Janssen J, Skiadas CH, Zopounidis C (eds) Advances in Stochastic Modelling and Data Analysis. Kluwer, Dordrecht, pp 261–292
87. Zopounidis C, Matsatsinis NF, Doumpos M (1996) Developing a multicriteria knowledge-based decision support system for the assessment of corporate performance and viability: The FINEVA system. Fuzzy Economic Rev 1(2):35–53
88. Zopounidis C, Pouliezos A, Yannacopoulos D (1992) Designing a DSS for the assessment of company performance and viability. Comput Sci in Economics and Management 5:41–56

Financial Equilibrium

Anna Nagurney
University Massachusetts, Amherst, USA

MSC2000: 91B50

Article Outline

Keywords
A Multi-Sector, Multi-Instrument Financial Equilibrium Model
Optimality Conditions
Economic System Conditions
See also
References

Keywords

Efficient frontier; Risk-free asset; Market portfolio; Option; Perfect competition; Portfolio optimization; Variational inequality formulation

Finance is concerned with the study of capital flows over space and time. The theory of financial economics is a combination of many different theories among which the theories of finance and economics, mathematical programming, and utility theory are credited with the biggest contributions.

The current state of modern financial economic theory is based upon the fundamental contributions of economists in the decade of the 1950s. Here we review some of the major developments. For a more complete historical breakdown, see [32].

The first major breakthrough was by K. Arrow and G. Debreu, who, in a series of publications (cf. [1,2,4,12,13]), introduced an important extension to the existing economic theory. Their contributions brought competitive equilibrium theory to a new level and allowed for the development of modern economic and finance theory. Specifically, Arrow and Debreu applied the techniques of convexity and fixed point theory to a model that followed the neoclassical economic foundations of: market clearing, uncertainty, and individual rationality and then they derived new fundamental economic properties from these models (e. g., [3,14]).

F. Modigliani and M. Miller [28], in turn, showed that the capital structure of a firm, that is, the financial framework of the firm, usually measured by the debt to equity ratio, does not affect the value of a firm. In their work, for the first time, the idea of financial arbitrage was used by stating that any investor can use risk-

less arbitrage in order to avoid the financial structure of a firm. Their work serves as the base for most of the research on capital structure.

The other theoretical breakthrough was by H.M. Markowitz in 1952, the founder of modern portfolio theory. Markowitz [25] proposed that one of the principal objectives of investors, in addition to the maximization of the returns of their portfolios, is to diversify away as much risk as possible. He claimed that investors choose assets in a manner so that the risk of their portfolio matches their risk preferences. He suggested that individuals who cannot bear risk will invest in assets with low risk, whereas people more comfortable with risk will accept investments of higher risk. His work suggested that the trade-off between risk and return is distinct for each investor; however, the preferences of all people lie upon a fictitious curve which is usually called the *'frontier of efficient portfolios'*. Along this curve lie all the diversified portfolios that have the highest return for a given risk, or the lowest risk for a given return. Markowitz's model was based on mean-variance portfolio selection, where the average and the variability of portfolio returns were determined in terms of the mean and covariance of the corresponding investments.

Many versions and extensions of Markowitz's model have appeared in the literature (cf. [19], and the references therein). The first important simplification of Markowitz's model was suggested by W.F. Sharpe [35], through a model known as the *diagonal model*, in which 'the individual covariances between all securities are assumed to be zero'. According to this model, the variance-covariance matrix has zeros in all positions other than the diagonal.

The most significant extension of the models by Markowitz [25] and Sharpe [35], was the *Capital Asset Pricing Model* (CAPM), which was based on the work of Sharpe [36], J. Lintner [24], and J. Mossin [29]. In this model the concept of a risk-free asset and market portfolio were introduced. A *risk-free asset* is an asset with a positive expected rate of return and a zero standard deviation. A *market portfolio*, on the other hand, is a portfolio on the efficient frontier of the Markowitz model which is considered to be desirable by all investors. The CAPM assumes that all investors will select a portfolio that will be a linear combination of the risk-free asset and the market portfolio, and, hence, the equilibrium prices of all assets can be expressed as a linear combination of the risk-free price and the price of the market portfolio. Since some of the assumptions governing the CAPM were not realistic (such as the absence of transaction costs), the model was extended and improved several times in the years that followed. It is, nevertheless, one of the major breakthroughs in modern economic and finance theory and forms the basis for most of the financial models.

Most of the major extensions of the CAPM occurred in the decade of the seventies, where a series of papers either relaxed some of its assumptions, or derived empirical results by applying it to a series of problems. Among the most significant contributions of that time were: the extension to a multiperiod economy by R.C. Merton [27] and the consumption CAPM by D.T. Breeden [6] (which, however, failed empirically due to the difficulty in observing and computing consumption).

The dissatisfaction with the empirical tests of the CAPM led to more advanced models, such as the *Arbitrage Pricing Theory* (APT) by S.A. Ross [34]. The APT's main contribution was the inclusion of multiple risk factors and the generalization of the CAPM, which was considered to be a special case of APT with only a single risk factor. In particular, Ross assumed that the rate of return of every security can be expressed as a linear combination of some 'basic' risk factors.

Another major development in modern financial economic theory was the derivation of an accurate option pricing model by F. Black and M. Scholes [5], which revolutionized the pricing of financial instruments and the entire financial industry. Note that an *option* is, in general, the right to trade an asset for a preagreed amount of capital. If the right is not exercised after a predetermined period of time, the option expires and the holder loses the money paid for holding that right. A major part of the subsequent literature focused on different approaches to, simplifications of, and variations of the *Black–Scholes Model* (BSM). A significant simplification of the BSM was done by J.C. Cox, Ross, and M. Rubinstein [11] (see [27]).

Furthermore, the *mean-variance portfolio analysis* that was introduced and mathematically formulated by Markowitz [25,26] and later simplified by the diagonal model of Sharpe [35], was further extended by G.A. Pogue [33] and J.C. Francis [18], with the introduction of variance-covariance matrices for both assets and li-

abilities, applied to the asset-liability management of banks.

Most of the aforementioned models and theories were subsequently extended and improved. The APT of Ross was refined by G. Chamberlain [7] and G. Connor [8], and the Black–Scholes model was further explored and significantly generalized (see, e. g., [10,15,17]).

The majority of the literature in financial economics has been based on the assumption that investors cannot affect the prices at which they buy or sell. Each investor is considered to be an isolated case, who tries to maximize his utility function, subject to the prices that the market provides him. All the participants in the economy, be they buyers or sellers, have as a goal the maximization of their profits and the minimization of their losses. The prices are derived through the market where investors constantly buy and sell commodities. The analysis of market equilibrium tries to determine the prices at which different products will be bought and sold, and also the amount of each product that each participant in the economy will hold in an equilibrium state.

Market equilibrium analysis has its roots in the last half of the nineteenth century. The work of H. Gossen [21], W. Jevons [23], and L. Walras [39] initiated the analysis of equilibrium theory. Subsequently, in the 1930s the study of market equilibrium became more formal and solid. The work of A. Wald [37,38] and J.R. Hicks [22] provided, for the first time, proofs of different qualitative properties of the equilibrium, along with a detailed study of the conditions under which an equilibrium could be modeled and derived. Furthermore, the work of Arrow [1] and G. Debreu [12] started a new era in equilibrium analysis by bringing uncertainty into equilibrium theory, which led to the current status of market equilibrium theory.

The basic assumption that governs most of the existing models that address the theory of market equilibrium is that of *perfect competition*. Perfect competition prohibits any participant in the economy (buyer or seller) from having control over the prices of different products or over the actions of other participants. The price of a product is considered to be a variable, the value of which is determined by the combined actions of all the buyers and sellers. Buyers are, hence, *'price takers'*, in that they modify their holdings of a product according to the price, ignoring the effects that their behavior may have on that price. Moreover, perfect competition assumes that all participants in the economy have perfect information about the products available, the current price, and the bids of a specific product. Furthermore, the number of the participants in the economy is assumed to be large enough so that the market activity regarding a specific product will be small compared to the transactions in the overall market.

For definiteness, we present a financial equilibrium model due to A. Nagurney [30] (see, also, [32], and the references therein). The model relaxes the CAPM assumptions of homogeneous expectations (cf. [24,29,36]), without imposing restrictions as to the nature of different sectors (e. g., [20]).

The mathematical framework that is utilized to develop the multi-sector, multi-instrument financial equilibrium model is finite-dimensional variational inequality theory. The methodology of finite-dimensional variational inequalities was first suggested for the modeling, analysis, and computation of multi-sector, multi-instrument financial equilibrium problems by Nagurney, J. Dong, and M. Hughes [31] and was further explored by Nagurney [30]. For complete references, qualitative results, as well as a plethora of financial equilibrium models and computational approaches, see [32].

A Multi-Sector, Multi-Instrument Financial Equilibrium Model

Consider a single country economy with multiple instruments and with multiple sectors. We let i denote a typical instrument, with the total number of instruments available in the economy, denoted by I. We let j denote a typical sector in the economy, with the number of sectors denoted by J.

Let r_i denote the (nonnegative) price of instrument i, and group the prices of all the instruments into the column vector $r \in \mathbf{R}^I_+$. Denote the volume of instrument i that sector j holds as an asset, by X^j_i, and group the (nonnegative) assets in the portfolio of sector j into the column vector $X^j \in \mathbf{R}^I_+$. Further, group the assets of all sectors in the economy into the column vector $X \in \mathbf{R}^{JI}_+$. Similarly, denote the volume of instrument i that sector j holds as a liability, by Y^j_i, and group the (nonnegative) liabilities in the portfolio of sector j into

the column vector $Y^j \in \mathbf{R}^I_+$. Finally, group the liabilities of all sectors in the economy into the column vector $Y \in \mathbf{R}^{JI}_+$.

Assume that the total volume of each balance sheet side of each sector is exogenous. Recall that a *balance sheet* is a financial report that demonstrates the status of a company's assets, liabilities, and the owner's equity at a specific point of time. The left-hand side of a balance sheet contains the assets that a sector holds at a particular point of time, whereas the right-hand side accommodates the liabilities and owner's equity held by that sector at the same point of time. According to accounting principles, the sum of all assets is equal to the sum of all the liabilities and the owner's equity. Moreover, we assume that the sectors under consideration act in a perfectly competitive environment.

Since each sector's expectations are formed by reference to current market activity, a sector's expected utility maximization can be written in terms of optimizing the current portfolio. Sectors may trade, issue, or liquidate holdings in order to optimize their portfolio compositions.

We assume that each sector j tries to maximize his utility function, which we denote as $U^j(X^j, Y^j, r)$. We also assume that the utility function of every sector is concave, continuous, and twice continuously differentiable. Furthermore, the accounts of each sector must balance. We denote the total financial volume held by sector j by S^j. Therefore, the optimization problem that each sector j faces is given by:

$$\begin{cases} \max & U^j(X^j, Y^j, r) \\ \text{s.t.} & \sum_{i=1}^{I} X_i^j = S^j, \\ & \sum_{i=1}^{I} Y_i^j = S^j, \\ & X_i^j \geq 0,\ Y_i^j \geq 0, \\ & \quad i = 1, \ldots, I, \end{cases}$$

where the price vector r is an exogenous vector in the optimization problem of every sector j; $j = 1, \ldots, J$.

We now discuss the feasible set of the sectors. For each sector j; $j = 1, \ldots, J$, we let

$$\widetilde{X}^j \equiv \left\{ X^j \in \mathbf{R}^I_+ : \ \sum_{i=1}^{I} X_i^j = S^j \right\}$$

denote the constraint set of his assets. Similarly, we let

$$\widetilde{Y}^j \equiv \left\{ Y^j \in \mathbf{R}^I_+ : \ \sum_{i=1}^{I} Y_i^j = S^j \right\}$$

denote the constraint set for his liabilities. Then, the feasible set for a sector j is a Cartesian product, denoted by κ^j, where

$$\kappa^j \equiv \{\widetilde{X}^j \times \widetilde{Y}^j\}.$$

Let $\widetilde{X}$ denote the feasible set for the assets of all the sectors, where:

$$\widetilde{X} \equiv \widetilde{X}^1 \times \ldots \times \widetilde{X}^j \times \ldots \times \widetilde{X}^J.$$

Similarly, for the liabilities, let $\widetilde{Y}$ denote the feasible set of the liabilities of all the sectors, that is,

$$\widetilde{Y} \equiv \widetilde{Y}^1 \times \ldots \times \widetilde{Y}^j \times \ldots \times \widetilde{Y}^J.$$

Also, define $\kappa \equiv \{\widetilde{X} \times \widetilde{Y}\}$.

We now present the optimality conditions for a sector's utility maximization problem, given above. We then give the economic conditions determining the instrument prices (in equilibrium).

Optimality Conditions

The necessary and sufficient conditions for an optimal portfolio for sector j are that the vector of assets and liabilities, $(X^{j*}, Y^{j*}) \in \kappa^j$, satisfies the following system of equalities and inequalities: For each instrument i, $i = 1, \ldots, I$, we must have the following *Kuhn–Tucker conditions* being satisfied, at an equilibrium price vector r^*:

$$-\frac{\partial U^j(X^{j*}, Y^{j*}, r^*)}{\partial X_i^j} - \mu_j^1 \geq 0,$$

$$-\frac{\partial U^j(X^{j*}, Y^{j*}, r^*)}{\partial Y_i^j} - \mu_j^2 \geq 0,$$

$$X_i^{j*}\left(-\frac{\partial U^j(X^{j*}, Y^{j*}, r^*)}{\partial X_i^j} - \mu_j^1\right) = 0,$$

$$Y_i^{j*}\left(-\frac{\partial U^j(X^{j*}, Y^{j*}, r^*)}{\partial Y_i^j} - \mu_j^2\right) = 0,$$

where μ_j^1, μ_j^2 are the Lagrange multipliers associated with the constraints. Obviously, a similar set of equalities and inequalities holds for every other sector in the single country economy.

Economic System Conditions

Moreover, the economic system conditions that ensure market clearance at a positive instrument price (and a possible excess supply of the instrument at a zero price) are: For each instrument i, $i = 1, \ldots, I$, we must have that:

$$\sum_{j=1}^{J}(X_i^{j*} - Y_i^{j*}) \begin{cases} = 0 & \text{if } r_i^* > 0, \\ \geq 0 & \text{if } r_i^* = 0. \end{cases}$$

This system of equalities and inequalities states that if the price of a financial instrument is positive, then the market must clear for that instrument and if the price is zero, then either there is an excess supply of that instrument in the economy or the market clears.

Let K be the feasible set for all the asset and liability holdings of all the sectors, and all the prices of all the instruments where $K \equiv \{\kappa \times \mathbf{R}_+^I\}$.

Combining the above, we present the following definition of equilibrium.

Definition 1 (*financial equilibrium*) A vector $(X^*, Y^*, r^*) \in K$ is an equilibrium of the single country, multi-sector, multi-instrument financial model if and only if it satisfies the system of equalities and inequalities above, for all sectors j, $j = 1, \ldots, J$, and for all instruments i, $i = 1, \ldots, I$, simultaneously.

The necessary and sufficient conditions for optimal portfolios, along with the economic conditions for the instrument prices, are now utilized in obtaining the variational inequality formulation of the financial equilibrium conditions.

Theorem 2 (***variational inequality formulation***) *A vector of assets and liabilities of the sectors, and instrument prices, $(X^*, Y^*, r^*) \in K$, is a financial equilibrium if and only if it satisfies the variational inequality problem:*

$$\sum_{j=1}^{J}\sum_{i=1}^{I}\left[-\frac{\partial U^j(X^{j*}, Y^{j*}, r^*)}{\partial X_i^j}\right] \times [X_i^j - X_i^{j*}]$$
$$+ \sum_{j=1}^{J}\sum_{i=1}^{I}\left[-\frac{\partial U^j(X^{j*}, Y^{j*}, r^*)}{\partial Y_i^j}\right] \times [Y_i^j - Y_i^{j*}]$$
$$+ \sum_{i=1}^{I}\sum_{j=1}^{J}[X_i^{j*} - Y_i^{j*}] \times [r_i - r_i^*] \geq 0,$$
$$\forall (X, Y, r) \in K.$$

We now put the variational inequality into standard form. We first define the J-dimensional column vector U with components: $\{U^1, \ldots, U^J\}$ and let $\nabla_X U$ denote the JI-dimensional vector with components: $\{\nabla_{X^1} U^1, \ldots, \nabla_{X^J} U^J\}$ with $\nabla_{X^j} U^j$ denoting the gradient of U^j with respect to the vector X^j. The expression $\nabla_Y U$ is defined accordingly. We let $n = 2JI + I$. We define the n-dimensional column vector $x \equiv (X, Y, r) \in K$, and the n-dimensional column vector $F(x)$ with components:

$$F(x) = \begin{pmatrix} F_1(x) \\ \vdots \\ F_b(x) \\ \vdots \\ F_n(x) \end{pmatrix} = \begin{pmatrix} -\nabla_X U(X, Y, r) \\ -\nabla_Y U(X, Y, r) \\ \sum_{j=1}^{J}(X_1^j - Y_1^j) \\ \vdots \\ \sum_{j=1}^{J}(X_I^j - Y_I^j) \end{pmatrix}_{n \times 1}.$$

Consequently, the variational inequality may be rewritten as:

- Determine $x^* \in K$ satisfying:

$$\langle F(x^*)^\top, x - x^* \rangle \geq 0, \qquad \forall x \in K.$$

Other financial equilibrium models, including models with hedging instruments such as futures and options, as well as, international financial equilibrium models can be found in [32], and the references therein.

See also

- ► Equilibrium Networks
- ► Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ► Oligopolistic Market Equilibrium
- ► Spatial Price Equilibrium
- ► Traffic Network Equilibrium
- ► Walrasian Price Equilibrium

References

1. Arrow KJ (1951) An extension of the basic theorems of classical welfare economics. Econometrica 51:1305–1323
2. Arrow KJ (1953) Les rôle des valeurs boursières pour la répartition la meilleure des risques. Econométrie Centre Nat Réch Sci, pp 41–48
3. Arrow KJ (1969) Collected Papers of Kenneth Arrow. Belknap Press
4. Arrow KJ, Debreu G (1954) Existence of an equilibrium for a competitive economy. Econometrica 22:265–290
5. Black F, Scholes M (1973) The pricing of options and corporate liabilities. J Political Economy 3:637–654

6. Breeden DT (1978) An intertemporal asset pricing model with stochastic consumption and investment opportunities. J Financial Economics 7:265–296
7. Chamberlain G (1983) Funds, factors and diversification in arbitrage pricing models. Econometrica 51:1305–1323
8. Connor G (1984) A unified beta pricing theory. J Econom Theory 34:13–31
9. Cox JC, Huang C (1987) Option pricing theory and its applications. In: Constantinides G, Bhattacharya S (eds) Frontiers of Financial Theory. Rowman & Littlefield
10. Cox JC, Ingersoll J, Ross SA (1985) An intertemporal general equilibrium model of asset prices. Econometrica 53:363–384
11. Cox JC, Ross SA, Rubinstein M (1979) Option pricing: A simplified approach. J Financial Economics 7:229–263
12. Debreu G (1951) The coefficient of resource utilization. Econometrica 19:273–292
13. Debreu G (1959) Theory of value. Yale Univ. Press
14. Debreu G (1970) Economies with a finite set of equilibria. Econometrica 38:387–392
15. Duffie D (1986) Stochastic equilibria: Existence, spanning number and the 'No expected financial gain from trade' hypothesis. Econometrica 54:1161–1184
16. Duffie D (1988) Security markets. Stochastic models. Acad. Press, New York
17. Duffie D, Huang C (1985) Implementing Arrow-Debreu equilibria by continuous trading of few long-lived securities. Econometrica 53:1337–1356
18. Francis JC (1978) Portfolio analysis of asset and liability management in small-medium-and large-sized banks. J Monetary Economics 3:112–134
19. Francis JC, Archer SH (1979) Portfolio analysis. Prentice-Hall, Englewood Cliffs, NJ
20. Gonedes NJ (1976) Capital market equilibrium for a class of heterogeneous expectations in a two-parameter world. J Finance 31:1–15
21. Gossen H (1854) Entwicklung der Gesetze des Menschlichen Verkehrs. Prager
22. Hicks JR (1939) Value and capital. Clarendon Press, Oxford
23. Jevons W (1871) The theory of political economy. MacMillan, New York
24. Lintner J (1965) The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. Rev Economic Stud 47:13–37
25. Markowitz HM (1952) Portfolio selection. J Finance 7:77–91
26. Markowitz HM (1959) Portfolio selection: Efficient diversification of investments. Wiley, New York
27. Merton RC (1973) An intertemporal capital asset pricing model. Econometrica 41:867–887
28. Modigliani F, Miller M (1958) The cost of capital, corporate cinance and the theory of corporation finance. Amer Economic Rev 48:261–297
29. Mossin J (1966) Equilibrium in a capital asset market. Econometrica 34:768–783
30. Nagurney A (1994) Variational inequalities in the analysis and computation of multi-sector, multi-instrument financial equilibria. J Econom Dynam Control 18:161–184
31. Nagurney A, Dong J, Hughes M (1992) Formulation and computation of general financial equilibrium. Optim 26:339–354
32. Nagurney A, Siokos S (1997) Financial networks: Statics and dynamics. Springer, Berlin
33. Pogue GA (1970) An extension of the Markowitz portfolio selection model to include transaction costs, short sales, leverage policies, and taxes. J Finance 25:1005–1027
34. Ross SA (1976) Arbitrage theory of capital asset pricing. J Econom Theory 13:341–360
35. Sharpe WF (1963) A simplified model for portfolio analysis. Managem Sci 9:277–293
36. Sharpe WF (1964) Capital asset prices: A theory of market equilibrium under conditions of risk. J Finance 19:425–443
37. Wald A (1935) Über die eindeutige positive Losbarkeit der neuen Productionsgleichungen. Ergebn Math Kolloq 6:12–20
38. Wald A (1936) Über die productionsgleichungen der Ökonomischen wertlehre. Ergebn Math Kolloq 7:1–6
39. Walras L (1874) Eléments d'economie politique pure. Guillaumin

Financial Optimization

JOHN M. MULVEY[1], BALA SHETTY[2]

[1] Department Operations Research & Financial Engineering, Princeton University, Princeton, USA

[2] Department Information & Operations Management, Texas A&M University, College Station, USA

MSC2000: 91B28

Article Outline

Keywords
Single-Period Models
Multiperiod Models
  Parameters
  Decision Variables
  Model SP
Scenario Generation
Solution Techniques
  Direct Solvers
  Decomposition Algorithms
Conclusions and Future Directions
See also
References

Keywords

Stochastic programming; Nonlinear optimization; Network programming

There is great need for an integrative approach to financial analysis and planning. The globalization of financial markets and the introduction of complex products such as exotic derivatives have increased volatility and risks. Strides in computers and information technology has eliminated any delays between the occurrence of an event and the impact on the markets — within the home country and internationally. The domain of *financial planning* provides a rich source of applications for optimization models and related tools. Such tools as simulation, estimation, stochastic processes, decision support, and artificial intelligence have become indispensable in several domains of financial operations [36]. With the continued growth of complex financial instruments and an increased acceptance of operations research tools by practitioners, optimization models are positioned to play a significant role in financial planning. There is a wealth of literature available regarding the role of optimization models in financial planning. See [12,16,23,32,35,37,38].

The primary purpose of this article is to present an overview of an integrative optimization-based financial planning model. In financial applications, the planner must provide recommendations from among a large of number of alternatives in which there is considerable uncertainty. The financial planner must therefore model the decisional environment as well as the stochastic elements in a dynamic fashion. The model presented here encompasses several popular approaches to the problem of investment strategies, including stochastic programs and dynamic stochastic control [4]. The financial planning model results in large stochastic optimization problems and efficient algorithms are now available for solving these nonlinear programs. A brief review of the various algorithms is also presented.

Single-Period Models

The most widely used methods for portfolio selection are based on the mean/variance approach [20]. Mean-variance optimization is a mathematical tool that creates a portfolio of assets with the maximum expected return for a given level of risk or with the minimum risk for a given expected return. Over the years, a number of researchers have extended and refined the original model to include transactions costs, trading size and turnover constraints and other practical requirements [30]. Several researchers have provided efficient procedures for estimating the variance/covariance matrix of returns required by the model, based on factor, index or scenario analysis [10].

While mean-variance analysis provides a powerful framework for asset allocation, it suffers from several limitations. The Markowitz model treats expected returns, standard deviations, and correlations as population parameters. These population parameters are not available, and therefore statistical estimates are used. The estimation errors thus introduced can distort the optimization results and could result in major errors in asset allocation.

Single-period models cannot capture long-term investment goals. They do not have the ability to consider opportunity costs that should influence decisions on strategic placement of funds; investment opportunities with maturities exceeding a single period cannot be included; neither can the impact of anticipated exogenous supply/demand for funds be properly assessed [21]. Single-period models tend to produce high portfolio turnovers and opportunistic asset trades. They cannot accurately account for the effect of transaction costs. Purchases of asset categories with high transaction costs are disfavored unless they promise high immediate returns. Multiperiod models, properly formulated, can overcome many of these limitations. This is the focus of our discussion in the next section.

Multiperiod Models

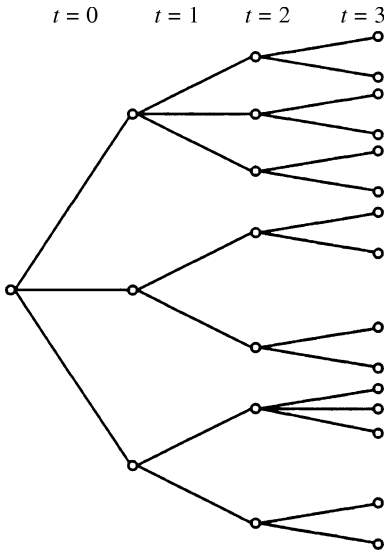
We address financial planning over long horizons via multistage stochastic programming. The stochastic program brings together all major financial-related decisions in a single and consistent structure. It integrates investment strategies (also known as asset allocation strategies), liability decisions (e. g., borrowings) and savings strategies (or re-investment decisions) in a comprehensive fashion. As such, the system forms the basis for assessing and managing risks for large institutional organizations, including banks, savings and loans, insurance companies, pension plans, and gov-

ernment entities. Several noteworthy applications of stochastic programming for financial planning include the Russell–Yasuda investment system for insurance companies [6], the Towers Perrin investment system for pension plans [22], the integrated simulation and optimization system for the Metropolitan Life Insurance Company [35], and the integrated product management system [12]. In each case, asset investment decisions are combined with liability choices in order to maximize the investor's wealth over time.

We describe a generalized network model for multiperiod investment planning [23]. While some real-world issues are difficult to accommodate within the network context and must be handled as general linear constraints, the network provides a visual reference for the financial planning system. We divide the entire planning horizon T into two discrete time intervals T_1 and T_2 where $T_1 = \{0, \ldots, \tau\}$ and $T_2 = \{\tau + 1, \ldots, T\}$. The former corresponds to periods in which investment decisions are made. Period τ defines the date of the planning horizon; we focus on the investor's position at the beginning of period τ. Decisions occur at the beginning of each time stage. Much flexibility exists. An active trader might see his time interval as short as minutes, whereas a pension plan advisor will be more concerned with much longer planning periods such as the dates between the annual Board of Director's meeting. It is possible for the steps to vary over time — short intervals at the beginning of the planning period and longer intervals towards the end. T_2 handles the horizon at time τ by calculating economic and other factors beyond period τ up to period T. The investor cannot render any active decisions after the end of period τ.

Asset investment categories are defined by set $A = \{1, \ldots, I\}$, with category 1 representing cash. The remaining categories can include broad investment groupings such as stocks, bonds, and real estate. The categories should track well-defined market segment. Ideally, the co-movements between pairs of asset returns would be relatively low so that diversification can be done across the asset categories.

In our approach, uncertainty is represented by a number of distinct realizations. Each complete realization of all uncertain parameters gives rise to a *scenario*; we denote by S the discrete set of all scenarios. Several scenarios may reveal identical values for the uncertain quantities up to a certain period – i. e., they share common information history up to that period (see Fig. 1). Scenarios that share common information up to a specific period must yield the same decisions up to that period. We will address the representation of the information structure through a condition known as *nonanticipativity*.



Financial Optimization, Figure 1
Scenario tree

We assume that the portfolio is rebalanced at the beginning of each period. Alternatively, we could simply make no transaction except reinvest any dividend and interest – a buy and hold strategy. For convenience, we also assume that the cashflows are reinvested in the generating asset category and all the borrowing is done on a single period basis.

For each $i \in A$, $t \in T_1$, and $s \in S$, we define the following parameters and decision variables.

Parameters

- $r^s_{i,t} = 1 + \rho^s_{i,t}$, where $\rho^s_{i,t}$ is the percent return for asset i, time period t, under scenario s (projected by the stochastic modeling subsystem). π_s is the probability that scenario s occurs, $\sum_{s=1}^{S} \pi_s = 1$.
- w_0 is the wealth in the beginning of time period 0.
- $\sigma_{i,t}$ are the transaction costs incurred in rebalancing asset i at the beginning of time period t (symmetric transaction costs are assumed, i. e., cost of selling equals cost of buying).
- β^s_t is the borrowing rate in period t under scenario s.

Decision Variables

- $x^s_{i,t}$ is the amount of money in asset category i, in time period t, under scenario s, after rebalancing.
- $v^s_{i,t}$ is the amount of money in asset category i, in the beginning of time period t, under scenario s, before rebalancing.
- w^s_t is the wealth at the beginning of time period t, under scenario s.
- $p^s_{i,t}$ is the amount of asset purchased for rebalancing in period t, under scenario s.
- $d^s_{i,t}$ is the amount of asset i sold for rebalancing in period t, under scenario s.
- b^s_t is the amount borrowed in period t, under scenario s.

With these definitions in place, we can present the deterministic equivalent of the stochastic asset allocation problem.

Model SP

$$\max Z = \sum_{s=1}^{S} \pi_s f(w^s_\tau) \tag{1}$$

such that

$$\sum_i x^s_{i,0} = w_0, \quad \forall s \in S, \tag{2}$$

$$\sum_i x^s_{i,\tau} = w^s_\tau, \quad \forall s \in S, \tag{3}$$

$$v^s_{i,t} = r^s_{i,t-1} x^s_{i,t-1}, \quad \forall s \in S, \quad t = 1, \ldots, \tau, \quad i \in A, \tag{4}$$

$$x^s_{i,t} = v^s_{i,t} + p^s_{i,t}(1 - \sigma_{i,t}) - d^s_{i,t}, \quad \forall s \in S, \quad i \neq 1, \quad t = 1, \ldots, \tau \tag{5}$$

$$x^s_{1,t} = v^s_{1,t} + \sum_{i \neq 1} d^s_{i,t}(1 - \sigma_{i,t}) - \sum_{i \neq 1} p^s_{i,t} - b^s_{t-1}(1 + \beta^s_{t-1}) + b^s_t, \quad \forall s \in S, \quad t = 1, \ldots, \tau, \tag{6}$$

$$x^s_{i,t} = x^{s'}_{i,t} \quad \text{for all scenarios } s, s' \text{ with identical past up to time } t. \tag{7}$$

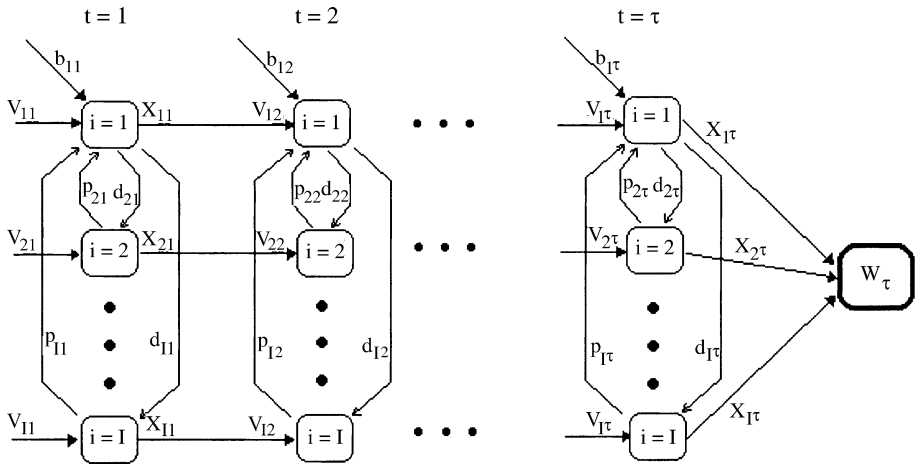
The generalized network model is presented in Fig. 2. The nonlinear objective function (1) can take several different forms. If the classical mean-variance function is employed, then (1) becomes $\max Z = \eta \text{Mean}(w_\tau) - (1 - \eta) \text{Var}(w_\tau)$, where $\text{Mean}(w_\tau)$ is the average total wealth and $\text{Var}(w_\tau)$ is the variance of the total wealth across the scenarios at the end of period τ. Parameter η indicates the relative importance of variance as compared with the expected value. This objective leads to an efficient frontier of wealth at period τ. An alternative to mean-variance is the von Neumann–Morgenstern expected utility of wealth at period τ. Here, the objective becomes

$$\max Z = \sum_{s=1}^{S} \pi_s \text{Utility}(w^s_\tau),$$

where $\text{Utility}(W)$ is the von Neumann–Morgenstern utility function [15]. The two objective functions are equivalent under certain conditions on the distribution of returns and the shape of the utility function [17].

Constraint (2) guarantees that the total initial investment equals the initial wealth. Constraint (3) represents the total wealth in the beginning of period τ. This constraint can be modified to include assets, liabilities, and investment goals. The modified result is called the *surplus wealth* [21]. Most investors make allocation decisions without reference to liabilities or investment goals. J.M. Mulvey employs the notion of surplus wealth to the mean-variance and the expected utility models to address liabilities in the context of asset allocation strategies. Constraint (4) depicts the wealth $v^s_{i,t}$ accumulated at the beginning of period t before rebalancing in asset i. The flow balance constraint for all assets except cash for all periods is given by constraint (5). This constraint guarantees that the amount invested in period t equals the net wealth for asset. Constraint (6) represents flow balancing constraint for cash. Nonanticipativity constraints are represented by (7). These constraints ensure that the scenarios with the same past will have identical decisions up to that period. While these constraints are numerous, solution algorithms take advantage of their simple structure.

Model SP is a split variable formulation of the stochastic asset allocation problem. This formulation has proven successful for solving the model using techniques such as progressive hedging algorithm of [27] and quadratic diagonal approximation of [25]. Split variable formulation is also found beneficial by direct solvers that use the interior point method [19]. By substituting constraint (7) back in constraint (2) to (6), we



Financial Optimization, Figure 2
Generalized network model for each scenario $s \in S$

obtain a compact formulation of the stochastic allocation problem. Constraints for this formulation exhibit a dual block diagonal structure. This formulation may be better for some direct solvers [19].

Scenario Generation

Scenario analysis offers an effective, and easily understood tool for addressing the stochastic elements in a multistage financial model. We define a scenario as a single deterministic realization of all uncertainties over the planning horizon. Ideally, the process constructs scenarios that represent the universe of possible outcomes. This objective differs from generation of a single scenario, say for forecasting and trading strategies. We are interested in constructing a 'representative' set of scenarios that are both optimistic and pessimistic within a risk analysis framework. Such an effort was undertaken by Towers Perrin (one of the largest actuarial firms in the world) using a system called CAP:Link [22]. The system entails a cascading of a set of submodels, starting with the interest rate component. Towers Perrin employs a version of the Brennan–Schwartz [5] two-factor interest rate model. The other submodels are driven by the interest rates and other economic factors. Towers Perrin has implemented the system in over 14 countries in Europe, Asia, and North America.

Scenario generation requires the estimation of the input parameters for the modeling of the economic factors. The ability to choose the 'correct' or 'best' set of parameters is essential if such models are to have practical value. A variety of techniques are available for estimating the economic factors required for projected returns and liabilities. See [24] and [1] for a discussion of some these techniques. See also [8,9] for a treatment of the robustness of scenario generation.

Solution Techniques

In this section we review a number of algorithms available to solve the asset allocation models. We focus on solutions to multistage stochastic programs possessing discrete-time decisions with a modest number of scenarios – typically under 1000 to 3000 – and nonlinear objective functions addressing risk aversion. The model's size depends on the number of decision variables and the form of the nonanticipativity rules. If Model SP is selected, the model becomes a convex program whose size hinges on the number of scenarios that are placed in S.

Direct Solvers

The simplest approach when the objective is linear is to use an efficient linear programming solver. Although simpler to handle, LP does not represent risk aversion well. See [19] for a solution of the multistage asset allocation problem with a linear objective function using OB1 and MINOS. OB1 is a primal-dual interior point algorithm for solving linear programs [18]. MINOS is a nonlinear programming code that can also

solve LP [28]. On a compact formulation, MINOS outperformed OB1 on several test problems. The split formulation, however, significantly reduced the time required by OB1 to yield the fastest solution times.

When the objective is nonlinear, a general purpose nonlinear programming code can be used for solution. However, the nonlinear interior point methods have advantages over these codes. For example, in mean-variance applications, the covariance matrix can be factored to convert the mean-variance function into a separable function. This is achieved by a modest increase in the number of constraints. R.J. Vanderbei and T.J. Carpenter [34] show that nonlinear interior point methods can take advantage of the separable structure despite the increase in the number of constraints. A similar transformation is possible with the expected utility objectives as discussed in [3].

Primal-dual interior point algorithms can be specialized to solve nonlinear stochastic optimization problems. See [7] for an extension of a primal-dual interior point procedure for linear programs to the case of convex separable quadratic objectives. The extension is tested on the asset allocation problems of [26] and compared to MINOS. The primal-dual interior point method compared favorably with MINOS, especially for the larger test problems. In the direct solution of nonlinear programs via interior point methods, the primary computational step is the factorization of the normal equations ADA^{T}, where A is the coefficient matrix and D is a diagonal matrix [18]. This factorization is typically done by means of the Cholesky (LL^{T}) method. A major difficulty when applying these algorithms to stochastic optimization problems has to do with the sparsity structure of A. Several efficient approaches are now (2000) available to address the sparsity issue, the most recent being the tree dissection method [2].

Ideas of using parallel computing for stochastic programs have been around for quite some time [29]; [14]. More recently (1993), E. Rothberg [31] developed an extremely efficient method for carrying out sparse matrix factorization in a parallel environment. Rothberg's factorization coupled with tree dissection concepts provides some very encouraging results for stochastic programs. Initial evidence indicates that parallel direct solvers will be able handle stochastic programs with over 10,000 scenarios within several minutes of runtime in a parallel environment.

Decomposition Algorithms

Considerable progress has been made in the design of efficient decomposition algorithms for solving multistage stochastic programs. A number of decomposition algorithms are based on the augmented Lagrangian function, such as the progressive hedging algorithm (PHA) and the diagonal quadratic approximation (DQA). PHA applies to the variable split form of the multistage stochastic program. The nonanticipativity constraints are placed in the objective function as penalty and multiplier terms, and are progressively enforced by an iterative procedure. Mulvey and H. Vladimirou [27] compare the performance of the progressive hedging algorithm to alternative solution strategies on a set of linear and nonlinear portfolio management problems. The general purpose optimizer MINOS [28] solve these test problems in their compact form. This is the most efficient program formulation for MINOS because it results in the smallest constraint matrix, i. e. the size of the basis is minimized. The linear problems were also solved using the primal-dual interior code (OB1) of [18]. For nonlinear test cases, they employ an extension of the primal-dual interior point method to convex, separable optimization programs [7]. The staircase formulation obtained by partial variable splitting is employed in these terms. On linear problems the progressive hedging algorithm was faster than MINOS. It was also faster than OB1 when the compact form was used. Interior point outperformed PHA for staircase structures. On nonlinear problems, PHA maintains its superiority over MINOS, particularly on large test problems. The progressive hedging algorithm also fares well against interior point algorithm on nonlinear problems, outperforming it in several cases.

DQA forms an augmented Lagrangian function by dualizing nonanticipativity constraints. The DQA algorithm approximates the Lagrangian at the current iterate by a quadratic and separable term [25]. The outer loop revises the dual variable by the method of multipliers, whereas the inner loop consists of separable quadratic or convex terms. DQA is a flexible scheme which can be implemented in many ways, in particular, in a parallel distributed environment. Mulvey and A. Ruszczynski [25] compare the performance of DQA with highly specialized meth-

ods for linear two-stage problems. The most successful methods found so far are based on Benders decomposition, suggested for stochastic programming in [33]. MSLiP [11] is a recent (1990) implementation of this idea, which allows for solving linear multistage problems in a nested formulation. Mulvey and Ruszczynski [25] show that the specialized decomposition techniques MSLiP and DQA outperform MINOS.

Conclusions and Future Directions

The proposed multistage financial planning model provides a general framework for integrating all asset and liability decisions for a large financial entity – such as an insurance company, bank, or pension plan – as well as for individual investors. This comprehensive approach measures the risk and rewards of alternative investment strategies. Without an integrative asset-liability model, investors are unable to properly measure risks to their wealth. The usual asset-only approach inadequately evaluates the impact of investments on wealth and achieving investment goals. The main lesson is that investment models must be tailored to individual circumstances. The multistage stochastic program provides an ideal vehicle for developing a financial plan that fits the investor's needs.

Future research should continue along several dimensions. First, we must increase the size of solvable stochastic programs so that additional scenarios can be handled in a practical fashion. There is no fundamental reason why we cannot address 10,000 to 100,000 scenarios using parallel and distributed computers. Certainly, the raw computing power will be available. Whether or not direct solvers or decomposition algorithms are best is a matter for future research.There are a number of algorithmic items to explore. One is to take further advantage of the structure of the multistage stochastic program within a parallel interior-point algorithm. For instance, we can conduct the Cholesky factorization using modern sparse matrix calculations on parallel or distributed computers. Rothberg's approach [31] seems to be a potential winner. Solving the stochastic program as quickly as possible will increase the chances that individual investors and institutions will apply the models. In the case of decomposition methods, the sparse matrix calculations are key for techniques such as DQA which use an interior-point algorithm for solving subproblems. Any substantial progress on this issue leads to immediate gains in the decomposition algorithm. Also, the restarting issue for interior-point algorithms remains.

Another computational issue involves generating scenarios. In particular, out-of-sample testing will be critical in order to compute valid bounds on the model recommendations. When it applies, dynamic stochastic control can be useful. The control system assists in the selection of the scenarios – for instance, by generating importance estimates for adding (or deleting) scenarios as they affect the solution to the control problem. These scenarios should be linked to the stochastic program. Of course, embedding a stochastic program within a simulation system such as carried out in [35] to evaluate the precision of the recommendations is possible. The approach requires large computational resources and may be impractical. Linking simulation and optimization models, however, will be increasingly important, as multistage stochastic programs become more widespread in practice.

A third issue deals with the automatic calibration of scenario generation systems using a nonlinear program. For example, the two-factor interest rate model possesses seven parameters, including the correlation coefficient for the Weiner terms. Setting these parameters requires considerable effort. There are several competing objectives: minimizing deviations on the summary statistics with respect to historical values; meeting expectations regarding future asset returns such as stocks and bonds; and avoiding trends that are clearly unrealistic. The estimation approaches developed in [24] and [1] address some these issues, but more work is needed to fully understand both modeling and computational issues of automatic calibration of scenarios.

See also

- ► Competitive Ratio for Portfolio Management
- ► Financial Applications of Multicriteria Analysis
- ► Portfolio Selection and Multicriteria Analysis
- ► Robust Optimization
- ► Semi-infinite Programming and Applications in Finance

References

1. Berger AJ, Mitchell JE, Mulvey JM, Rush R (1997) A tabu search procedure for target-matching in financial scenario generation. Techn. Report Dept. Civil Engin. Oper. Res. Princeton Univ. SOR-97-16
2. Berger AJ, Mulvey JM, Rothberg E, Vanderbei R (1996) Solving multi-stage stochastic programs using tree dissection. Techn. Report Dept. Civil Engin. Oper. Res. Princeton Univ. SOR-95-07
3. Berger AJ, Mulvey JM, Ruszczyński A (1994) An extension of the DQA algorithm to convex stochastic programs. SIAM J Optim 4:735–753
4. Birge JR (1997) Stochastic programming computation and applications. INFORMS J Comput 9:111–133
5. Brennan MJ, Schwartz ES (1982) An equilibrium model of bond pricing and a test of market efficiency. J Financial and Quantitative Anal 17:75–100
6. Cariño DR, Kent T, Myers DH, Stacy C, Sylvanus M, Turner A, Watanabe K, Ziemba WT (1994) The Russell–Yasuda Kasai model: An asset liability model for a Japanese insurance company using multi-stage stochastic programming. Interfaces 24:29–49
7. Carpenter TJ, Lustig IJ, Mulvey JM, Shanno DF (1993) Separable quadratic programming via a primal-dual interior point method and its use in a sequential procedure. ORSA J Comput 5:182–191
8. Dupacova J (1987) Stochastic programming with incomplete information: A survey of results on postoptimization and sensitivity analysis. Optim 18:507–532
9. Dupacova J (1996) Scenario based stochastic programs: Resistance with respect to sample. Ann Oper Res 64:21–38
10. Elton EJ, Gruber MJ (1995) Modern portfolio theory and investment analysis, 5th edn. Wiley, New York
11. Gassmann HI (1990) A computer code for the multi-stage stochastic linear programming problem. Math Program 47:407–423
12. Golub B, Holmer M, McKendall R, Pohlman L, Zenios SA (1995) A stochastic programming model for money management. Europ J Oper Res 85:282–296
13. Jarrow R, Maksimovic V, Ziemba W (1995) Asset and liability allocation in a global environment. In: Mulvey JM, Ziemba WT (eds) Handbook Oper. Res. and Management Sci.: Finance. Elsevier, Amsterdam
14. Jessup ER, Yang D, Zenios SA (1994) Parallel factorization of structured matrices arising in stochastic programming. SIAM J Optim 4:833–846
15. Keeney R, Raiffa H (1993) Decisions with multiple objectives: Preferences and value Tradeoffs. Cambridge Univ. Press, Cambridge. First printed by Wiley, 1976
16. Konno H, Luenberger DG, Mulvey JM (eds) (1993) Ann. Oper. Res. Baltzer, Basel
17. Kroll Y, Levy H, Markowitz H (1984) Mean variance versus direct utility Maximization. J Finance 39:47–62
18. Lustig IJ, Marsten RE, Shanno DF (1994) Interior point methods for linear programming: Computational state of the art. ORSA J Comput 6:1–14
19. Lustig IJ, Mulvey JM, Carpenter TJ (1991) Formulating stochastic programs for interior point methods. Oper Res 39:757–770
20. Markowitz HM (1991) Portfolio selection: Efficient diversification of investments, 2nd edn. Blackwell, Oxford
21. Mulvey JM (1989) A surplus optimization perspective. Investment Management Rev 3:31–39
22. Mulvey JM (1995) Generating scenarios for the Towers Perrin investment systems. Interfaces 26:1–15
23. Mulvey JM, Rosenbaum DP, Shetty B (1997) Strategic financial risk management and operations research. Europ J Oper Res 97:1–16
24. Mulvey JM, Rosenbaum DP, Shetty B (1999) Stochastic scenario generation systems. Europ J Oper Res 118:563–577
25. Mulvey JM, Ruszczyński A (1995) A new scenario decomposition method for large-scale stochastic optimization. Oper Res 43:477–490
26. Mulvey JM, Vladimirou H (1989) Stochastic network optimization models for investment planning. Ann Oper Res 20:187–217
27. Mulvey JM, Vladimirou H (1992) Stochastic network programming for financial planning problems. Managem Sci 38:1642–1664
28. Murtagh BA, Saunders MA (1987) MINOS 5.1 user's guide. SOL Techn. Report Systems Optim. Lib. Stanford Univ. 83-20R
29. Nielsen S, Zenios SA (1993) A massively parallel algorithm for nonlinear stochastic network problems. Oper Res 41:319–337
30. Perold AF (1984) Large-scale portfolio optimization. Managem Sci 30:1143–1160
31. Rothberg E (1993) Performance of panel and block approaches to sparse Cholesky factorization on the iPSC/860 and Paragon multicomputers. Techn. Report Intel Supercomputer Systems Div., Beaverton, OR
32. Speranza MG (1993) Linear programming models for portfolio optimization. J Finance 14:107–123
33. Van Slyke R, Wets RJ-B (1969) L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J Appl Math 17:638–663
34. Vanderbei RJ, Carpenter TJ (1993) Symmetric indefinite systems for interior-point methods. Math Program 58:1–32
35. Worzel K, Vassiadou-Zeniou C, Zenios SA (1995) Integrated simulation and optimization models for tracking fixed-income indices. Oper Res 42:223–233
36. Zenios SA (ed) (1993) Financial optimization. Cambridge Univ. Press, Cambridge
37. Zenios SA, Ziemba WT (1992) Theme issue on financial modeling. Managem Sci 38:1525–1528
38. Ziemba WT, Mulvey JM (eds) (1998) Worldwide asset and liability modeling. Cambridge Univ. Press, Cambridge

Finite Complete Systems of Many-valued Logic Algebras

LADISLAV J. KOHOUT
Department Computer Sci., Florida State University, Tallahassee, USA

MSC2000: 03B50, 68T15, 68T30

Article Outline

Keywords
Boolean 2-Valued Logic Algebras
A Repertory of Complete Many-Valued Logic Normal Forms
Many-Valued Families of the Pinkava Logic Algebras
General Pi-Algebras
Families of Pi-Algebras and their Functionally Complete Normal Forms
The Taxonomy of the PI-Algebras of Many-Valued Logics
Special Subfamilies of n-Valued PI-Systems
PI-Algebras and a New Variety of 2-Valued Normal Forms
Theoretical and Practical Importance of PI-algebras
The Requirement of Functional Completeness
Satisfiability Problem in Computational and Descriptive Complexity Theory
Central Importance of the Satisfiability Problem of Boolean Formulas in Complexity Theory
Open Problems in the Complexity Theory of PI-algebras (1999)
Applications of PI-Algebras
See also
References

Keywords

Many-valued logics; PI-logic algebras; Discrete functions; Normal form; Functional completeness; Satisfiability; Computational complexity; Descriptive complexity; Scientific applications; Medical applications

First we review briefly some facts about 2-valued discrete functions (two-atom Boolean algebras). Then we proceed with various n-valued extensions and generalizations which are not necessarily always Boolean. The most general class of systems to be discussed are Pi-algebras. The logic connectives of these algebras are partial nonassociative noncommutative general algebraic *groupoids* [13,30].

Associative connectives, such as various families of t-norms and t-conorms [10] which are widely used in fuzzy logics are the special instances of PI-algebra groupoid connectives. References to some applications of Pi-logic algebras conclude this entry.

Although the primitives of PI-algebras are only partially defined they are *functionally complete*. Hence one can represent any finite discrete function by PI-normal forms.

Definition 1 A finite *discrete function* of k arguments $f(x_1, \ldots, x_k)$ is a mapping from the k-fold Cartesian product of a set A to itself. In symbols: $f\colon A \times \cdots \times A \to A$, where A is a finite set containing n elements, $A = \{a_0, \ldots, a_{n-1}\}$.

For typographic convenience, we shall map the elements of A into the finite subset **N** of nonnegative numbers by the assignment $a_i = i$, namely $A = \{0, \ldots, n-1\}$. This does not imply that the ordering of natural numbers is always relevant to our algebraic considerations. These numbers should just be considered as more convenient labels than, say, a_i for the elements of a finite set A.

Boolean 2-Valued Logic Algebras

A *Boolean 2-valued function* is a discrete function that takes its values from the two-element set $\{0, 1\}$. We can form 16 different two-argument functions on the set $\{0, 1\}$. The ten nontrivial of these are shown below.

Finite Complete Systems of Many-valued Logic Algebras, Table 1
Two-argument connectives of the 2-valued logic

x	y	$\wedge$	$\vee$	$\rightarrow$	$\leftarrow$	$\equiv$	$\oplus$	$\nrightarrow$	$\nleftarrow$	$\downarrow$	$\mid$
0	0	0	0	1	1	1	0	0	0	1	1
0	1	0	1	1	0	0	1	1	0	0	1
1	0	0	1	0	1	0	1	0	1	0	1
1	1	1	1	1	1	1	0	0	0	0	0

Some of these operations, also called *logic algebra connectives*, play an important role in logic. Hence they are given special names to stress their meaning and significance.

When '1' is interpreted as 'True' and '0' as 'False', then $\vee$ represents logical AND, $\wedge$ (nonexclusive) logical OR, $\oplus$ exclusive logical OR (i. e. XOR). The connective $\equiv$ is *logical equivalence* which captures the equivalence of two propositions, and $\rightarrow$ is an *implication operator* which captures the validity (truth-value) of the *conditional* 'If __ then __'.

The connectives, together with some inference rules (e. g. modus ponens, see ► Checklist paradigm semantics for fuzzy logics), make a system of classical propositional logic.

Let us recall that any Boolean function can be expressed by the conjunctive normal form (CNF) or disjunctive normal form (DNF). Let us introduce the notation $x^\sigma = (x \wedge \sigma) \vee (\overline{x} \wedge \overline{\sigma})$, where $\overline{x}$ denotes the *negation* of x, and σ is a parameter equal either to 0 or 1. Then it is obvious that

$$x^\sigma = \begin{cases} \overline{x} & \text{when } \sigma = 1, \\ x & \text{when } \sigma = 0. \end{cases}$$

x^σ is called *literal*.

Theorem 2 (Normal forms theorem) *Every Boolean function $f(x_1, \ldots, x_n)$ can be represented by their canonical (full) CNF and DNF normal forms.*

i) The disjunctive normal form:

$$\bigvee_{f(\sigma_1,\ldots,\sigma_n)=1} x_1^{\sigma_1} \wedge \cdots \wedge x_n^{\sigma_n}.$$

ii) The conjunctive normal form:

$$\bigwedge_{f(\sigma_1,\ldots,\sigma_n)=0} x_1^{\overline{\sigma_1}} \vee \cdots \vee x_n^{\overline{\sigma_n}}.$$

A *clause* in a DNF consists either of a literal or of a conjunction of literals. In a CNF, on the other hand, a clause consists either of a literal or of a disjunction of literals.

Because we can express any Boolean function by formulas formed by means of the sets of connectives CNF-Cset (= $\{\wedge, \vee, \neg\}$), DNF-Cset (= $\{\vee, \wedge, \neg\}$), we call these sets *complete sets* of connectives.

A Repertory of Complete Many-Valued Logic Normal Forms

Important structural relationships that provide the algebraic backbone of various logics are contained in their normal forms. It is possible to generalize from two-valued normal forms to *many-valued normal forms* in various ways. We shall discuss here one such generalization, namely partial functionally complete Pinkava algebras (Pi-algebras) which offer some interesting insights and have also a significant practical value. The systems were discovered in 1971 by V. Pinkava [24,25] as a significant generalization of the systems used by Pinkava in his previous work [21].

Many-Valued Families of the Pinkava Logic Algebras

Definition 3 ([30,35]) The *Pinkava n-valued family of logical calculi* Pi = {A, $\Box$, $\diamond$, $\odot$, $\hookrightarrow$} consists of the partially defined connectives operating on the value-set {0, …, n −1}:

$$v_i \diamond v_j = \begin{cases} 0 & \text{if } v_i = 0, \\ 1 & \text{if } v_i \neq 0 \ \& \ v_j = 1, \\ \text{undefined} & \text{otherwise}, \end{cases}$$

$$v_i \Box v_j = \begin{cases} 0 & \text{if } v_i = 0, \\ v_j & \text{if } v_i \neq 0 \ \& \ v_j = 1, \\ \text{undefined} & \text{otherwise}, \end{cases}$$

$$v_1 \odot v_2 = \begin{cases} v_j & \text{if } v_i = 0, \\ \text{undefined} & \text{otherwise}, \end{cases}$$

$$v^{\hookrightarrow} = v + 1 \pmod{n},$$

$$\Psi_\kappa = \begin{cases} 1 & \text{if } v = \kappa, \\ 0 & \text{if } v \neq \kappa, \end{cases}$$

where $i, j \in \{1, 2\}$.

Note that the carrier and the *characteristic functions* can also be generated in the Pinkava logic calculi by the connectives. For example, the characteristic function Ψ can be generated by $\diamond$, [13].

Theorem 4 (Complete normal forms) *Any n-valued logic function that is obtained by a completion of the partially defined Pinkava connectives of the type {$\odot$ $\Box$, $\diamond$, $\hookrightarrow$} defined above is functionally complete and can be*

expressed in the following canonical normal form:

$$f(v_1,\ldots,v_n) = \bigodot_{f(v_1,\ldots,v_n)\neq 0} \left[c_\gamma \square \left(\diamond_{s=1}^{n} \psi_{\alpha_s}(v_s)\right)\right].$$

General Pi-Algebras

Families of Pi-Algebras and their Functionally Complete Normal Forms

The Pinkava logic calculi can be further generalized [11,13,14]. These generalizations are called *Pi-algebras*. The connectives involved are partial nonassociative noncommutative general (algebraic) groupoids in their most general form. *Associative connectives*, such as the *t-norms* and *t-conorms* [10,44] are special instances of them (see also ► Boolean and fuzzy relations).

Definition 5 (*Families of Pi-algebras*) Let Pi be an algebra with carrier P such that PI = $\langle P, \diamond, \square, \odot, \Phi\rangle$, where [13]:

1) $\langle P, \diamond\rangle$ is an arbitrary groupoid with zero $z_\diamond$, without divisors of zero, and with the almost absorbing element $a_\diamond$ such that $a_\diamond \diamond p = p \diamond a_\diamond = a_\diamond$ for every $p \in P, p \neq z_\diamond$.
2) $\langle P, \odot\rangle$ is an arbitrary groupoid with unit $e_\odot$.
3) $\langle P, \square\rangle$ is an arbitrary groupoid with a right zero $z_r\square$ and a right unit $e_r\square$.
4) Φ is a discrete *cyclic shift function* $\Phi{:}P \to P$ satisfying the following conditions: Given a discrete cyclic order of P, then for every $p \in P$ it holds that $p \preccurlyeq \Phi(p)$ and $\Phi^0(p) = p$, $\Phi^{k+1}(p) = \Phi(\Phi^k(p))$.

In the above definition $a \preccurlyeq b$ means that a is a direct predecessor of b.

Definition 6 Let $p_1, p_2 \in P$, and Φ be a cyclic shift function. Then the *advance* δ from p_1 to p_2 with respect to Φ is the least ordinal such that $\Phi^\delta(p_1) = p_2$. We write $\delta_\Phi(p_1) = p_2$. The advance δ^* denotes the inverse of δ.

Theorem 7 (Canonical normal forms) *Let the advances δ_1, δ_2 be defined by the formulas $\delta_1 := (a_\diamond, e_r\square)$, $\delta_2 := (z_\square, e_\odot)$. Then any function f on the carrier P in a Pi-algebra can be expressed in its canonical normal form:*

$$f(v_1, v_2, \ldots) = \bigodot_{f\neq e_\odot} \Phi^{\delta_2}\left\{\Phi^{\delta_2^*}(c_\gamma)\square\left(\diamond_{s=1}^{\operatorname{card} P}\Phi^{\delta_1}[\Psi_{\alpha_s}(v_s)]\right)\right\}.$$

The argument scope of the outer connective of the normal form is $\odot_{\{\forall f(v_1,v_2,\ldots): f\neq(e_\odot)\}}$. This means that the values $e_\odot$ are omitted.

Theorem 8 (Functional completeness) *Any Pi-algebra is functionally complete if, given the advance $\delta_1 = \delta(a_\diamond, e_r\square)$, it also holds that $\delta_1 = \delta(z_\diamond, z_\square)$.*

Theorem 9 *If the right zero $z_r\square$ is also the zero and the right unit $e_r\square$ is also the unit of the groupoid $\langle P, \square\rangle$, then the following normal form is also functionally complete:*

$$f(v_1, v_2, \ldots) = \bigodot_{f\neq e_\odot} \Phi^{\delta_2}\left\{\Phi^{\delta_2^*}(c_\gamma)\square\left(\square_{s=1}^{n}\Phi^{\delta_1}[\Psi_{\alpha_s}(v_s)]\right)\right\}.$$

The Taxonomy of the PI-Algebras of Many-Valued Logics

The main theoretical question that the PI-algebras answer is: 'Which features of two-valued Boolean logic structures disappear and which are preserved and carried over into the extensions and generalizations to many-valued logics'? The Pi-logic algebras are partial systems that put under one roof a wide variety of families of functionally complete many-valued logical systems. Thus they offer a useful framework in which various generalizations and extensions can be carried out. They also provide a sound base for a useful *classification of many-valued logics* by their various properties. This approach, based on *PI-normal forms*, usefully complements another way of classifying the many-valued logic connectives by groups of logic transformations (see ► Checklist paradigm semantics for fuzzy logics).

Special Subfamilies of *n*-Valued PI-Systems

Because the Pinkava connectives are only partially defined, it is possible by imposing further restrictions on these connectives to define subclasses of the functionally complete Pinkava systems. For example, consider the restrictions

a) $(v_1^{\hookrightarrow}\square v_2^{\hookrightarrow})^{\hookrightarrow n-1} = v_1 \odot v_2$;
b) $(v_1^{\hookrightarrow n-1} \odot v_2^{\hookrightarrow n-1})^{\hookrightarrow} = v_1 \square v_2$.

They make the subclass $\{\diamond, \square, \hookrightarrow\}$ functionally complete. Imposing some other restrictions we can ob-

tain other subclasses of functionally complete connectives. For example, the *Post system*, the *Aizenberg–Rabinovich system*, the *Zhegalkin algebra*, and *lattice-type many-valued logic systems* can be so obtained. For further details see e. g. [13,38]. A partial taxonomy of various subclasses of the Pinkava systems can be found in [13, Fig. 8.1] or in [11 p. 279, Fig. 1]. Several important subclasses of the Pinkava systems are presented in [26]. For the subclasses that are generalizations of the *Sheffer function*, see [32]. The systems particularly suitable for *minimizations* appear in [33].

PI-Algebras and a New Variety of 2-Valued Normal Forms

It is illuminating to look at two-valued well-known special instances of logic connectives and classify them in terms of Pi-algebra connective types. This reveals that there are other canonical normal forms in addition to DNF and CNF. For instance, the $\odot$, which is partial, offers two distinct completions: either Boolean (inclusive) OR $\vee$ or exclusive-OR (nonequivalence) $\oplus$. Although the connectives $\Box$ and $\diamond$ are identical in the two-valued case, both forming Boolean $\wedge$, they extend each to a distinct partial connective for $n > 2$. This is because each of these connectives plays a different role in the normal form, serving a different purpose.

In order to explore more fully the richness of Pi-algebras, one has to look at their *taxonomy* in the general many-valued case. For a more detailed taxonomy see [13, Fig. 8.1.2] or [11].

Two highlights emerge from this approach:

1) Even in the simple two-valued case, the normal forms of generalized Pi-algebras subsume not only the conjunctive and disjunctive normal form but also the implication, equivalence, exclusive-or and other normal forms in one unifying pattern.
2) Two distinct general n-valued connectives may 'collapse' into a single connective when one sets $n = 2$. Viewed the other way, a two-valued connective may 'bifurcate' into two distinct types of connectives when more than two values are used. This bifurcation of structures and concepts is an interesting phenomenon that accompanies *fuzzification* of two-valued structures.

Theoretical and Practical Importance of PI-algebras

The Requirement of Functional Completeness

The functional completeness is of primary interest to a scientist or an engineer engaged in practical applications of many-valued logic. In such applications it is often desirable to have the means for generating all possible finite discrete functions by means of a complete set of many valued logic connectives.

For example, it is desirable to have a set of *logic gates* that can generate any *combinatorial switching circuit*. In pattern recognizers implemented by many-valued logic networks, the set of basic *'cognitive elements'* has to be complete, otherwise some patterns may be misclassified. The completeness is necessary in order to have the means for representation of all the possible discrete functions over a finite set of elements.

Similarly, in biological or psychological and medical models based on abstract classification of patterns by logic nets the choice of an incomplete set of connectives as the representational base of the model might yield a bias towards assumptions that are not contained in the experimental data. For example, in ethological models of instincts the representation using an incomplete set of connectives would represent the a priori assumption that certain forms of instincts do not exist. Yet the data might contain the evidence for these, but this evidence is not representable and will be discarded by an unfortunate choice of the incomplete set of connectives. In models of neuro-psychological disorders this might cause a priori exclusion of some impairments of the substratum structures, diminishing the predictive usefulness of such models.

The complexity of the normal forms as well as the complexity of the minimized many-valued logic (MVL) expressions depends on the character of the discrete function (i. e. the data) to be represented, the choice of an appropriate many-valued logic system of connectives, and the number of the discrete values of the value set A. Hence, only by the choice of an appropriate MVL system may we achieve an optimal representation in each specific application domain.

The choice of a suitable system is usually an iterative process, which requires a comparative evaluation of several systems, performed in order to optimize the choice. In order to assess whether a chosen system is

functionally complete, a set of conditions sufficient to determine the completeness is required. Alternatively, a set of rules has to be given that would make it possible to generate complete systems of required additional properties directly. The constructive conditions for completeness given above for the normal forms of PI-algebras are such rules.

Other conditions for completeness of the same of greater generality are not so suitable for this purpose because the number of conditions necessary to test for completeness increases rapidly with increasing number of values n of a many-valued logic system. E. Post was first to give the general conditions for completeness of 2-valued logics ($n = 2$). These were later generalized by S.W. Jablonskij (S.V. Yablonskii) [9], J. Slupecki [45], A. Salomaa [43] and others. The most general conditions known at present are those given by I. Rosenberg [41,42] which are the generalization of the *Post conditions* for any n-valued finite case.

In all these later cases (unlike for PI-algebras initiated by Pinkava), the number of conditions increases astronomically with increasing n. For $n = 2$ (Post) there are 5 conditions that the logic system has to satisfy. For $n = 3$ (Jablonskij) there are 18 conditions. For a seven-valued MVL system ($n = 7$), there are 7,848,984 conditions to be tested. The general formula for any finite $n \leq 2$ is given by Rosenberg in [42]. This formula shows that, for large n, the number is rather prohibitive, hence of no practical value. On the other hand, PI-algebras can generate an infinite number of finite functionally complete systems of connectives for any finite n. This is so because the Pinkava complete sets of connectives are only partially specified and the completion of the 'blanks' by any values does not invalidate their completeness.

Satisfiability Problem in Computational and Descriptive Complexity Theory

Central Importance of the Satisfiability Problem of Boolean Formulas in Complexity Theory

The main goal in the *complexity theory of algorithms* is to distinguish problems that can be solved efficiently from those that cannot be. A computational solution to a problem is *practically feasible* if it belongs to the *complexity class P*, that can be computed by a deterministic algorithm in time bounded by a polynomial function of the size of the input data.

The central problem of complexity theory in computer science and a major problem of contemporary logic and mathematics is whether the class P is equal to the class NP. Problems solvable by nondeterministic algorithms in polynomial time belong to the class NP.

Problems in the class NP are computationally tractable only if they are of *polynomial complexity*, that is if $P = NP$.

A successful proof of the conjecture that $P \neq NP$ would on the other hand indicate that the NP class is of computationally not tractable *exponential complexity*.

The class NP contains many practical problems that can be characterised by the following property: There is no known way to compute a solution in polynomial time, but there is a known way to check in polynomial time whether a potential (e. g. guessed) solution is an actual solution.

The *satisfiability problem* [19,20] that concerns Boolean formulas [5] is closely related to the question about computational complexity of many other computational problems [6,7].

We say that a Boolean formula is *satisfiable* if there exists at least one way of assigning values to its variables so as to make it true. Finding the answer 'yes' or 'no' to this question is called the [2]. If the Boolean formula of our concern is written solely in the CNF we have the *SAT-CNF problem*. *SAT-k-CNF* is obtained by restricting SAT-CNF to Boolean formulas in k-CNF, where *k-CNF* is composed of clauses, each of which contains at most k literals [2].

It follows from *Cook's theorem* [3] that the question whether or not $P = NP$ is equivalent to asking whether there is a *polynomial time deterministic algorithm* (PTDA) recognizing the set of satisfiable Boolean propositional formulas (the SAT problem), or equivalently, a PTDA recognizing the set of propositional tautologies *TAUT* [19]. This demonstrates the central importance of the SAT problem for computational complexity theory.

In 1971 S.A. Cook [3] proved that every problem $X \in NP$ is *polynomially Turing reducible* [1,2] to the question about the complexity of *TAUT-DNF*, i. e. the set of propositional tautologies coded in DNF. In symbols: $X \leq_T^p$ TAUT-DNF.

This is related to one of the open (as of 1999) fundamental problems of logic [19]: Is there a *propositional proof system P* in which every tautology has a polynomial size proof? At present it is known only [4] that there exists a propositional proof system in which every tautology has a polynomial size of proof if and only if $NP = \text{co}\,NP$, i. e. the class NP is closed under complementation.

The relation $\leq_T^p$ is a pre-order (see ► Boolean and fuzzy relations) hence it provides the means for comparing the relative computational difficulty of problems [1]. Because it is a pre-order, it may contain various equivalence classes (see ► Boolean and fuzzy relations) of problems.

The statement: 'The SAT problem is *NP*-complete' is referred to as the *Cook–Levin theorem* in the literature. Using the reducibility relation $\leq_T^p$ together with this theorem yields a useful technique for providing proofs of the *NP*-completeness of other problems. We say that a problem X is *NP-complete* [1,2] if

- $X \in NP$; and
- $Y \leq_T^p X$ for every problem $Y \in NP$.

There is a great number of computational problems in the graph and set theories, the *NP*-completeness of which can be proven by reducing the SAT problem directly or indirectly to each of them. For example *dominating set*, *vertex cover*, *clique*, *3-SAT*, *3-colorability* [20]. SAT can be reduced to the clique problem, which in turn is reducible to the vertex cover problem. The vertex cover problem is reducible to the dominating set problem. Similarly, there is another chain of reductions: SAT to 3-SAT to 3-colorability. These reduction chains form a part of the semilattice generated by the reducibility relation.

Open Problems in the Complexity Theory of PI-algebras (1999)

Computational complexity of PI-logic algebras and normal forms is an uncharted territory. There is an infinite number of ways in which the partially specified but functionally complete PI-logic normal forms can be made fully specified, and a large variety of algebraic restrictions that can be placed upon them to generate particular fully defined systems.

Despite of their partial nature, the PI-normal form have well defined length even before their algebraic properties are completely specified. Hence one may expect that they will play some role in placing the upper bound on descriptive complexity [8] of propositional systems. This may be a promising direction of research in the future. It should also be noted that the generalized PI-normal forms allow for description of transformations from lattice based connectives to ring based connectives. Indeed, both are special instances of PI-logic algebras (see Theorem 7 and [13]). That might help to build a bridge between methods for analysis of algebraic propositional systems [40] with notions of descriptive complexity [8].

Applications of PI-Algebras

In addition to their theoretical significance, the functionally complete PI-systems have found a number of practical applications in various fields: in medicine, clinical behavioral sciences and neurology [12,22,23,34,36]; in data analysis and classification [37,39]; analysis of logical paradoxes [29,38]; automata theory and systems science [17,31,38]; design of MVL-switching circuits [11]; in dynamic computer protection also applicable to distributed an parallel systems [13,15,16]; logic and theorem proving [18,35]; optimization of discrete functions [27,33] and fuzzy logics [28].

See also

- ► Alternative Set Theory
- ► Boolean and Fuzzy Relations
- ► Checklist Paradigm Semantics for Fuzzy Logics
- ► Inference of Monotone Boolean Functions
- ► Optimization in Boolean Classification Problems
- ► Optimization in Classifying Text Documents

References

1. Balcázar JL, Díaz J, Gabarró J (1988) Structural complexity, vol I. Springer, Berlin
2. Brassard G, Bratley P (1996) Fundamentals of algorithmics. Prentice-Hall, Englewood Cliffs, NJ
3. Cook SA (1971) The complexity of theorem proving procedures. In: Proc. 3rd Annual ACM Symp. Theory of Computing. ACM, New York, pp 151–158

4. Cook SA, Reckhow AR (1979) The relative efficiency of propositional proof systems. J Symbolic Logic 44(1):36–50
5. Delgrande JP, Gupta A (1996) The complexity of minimum partial truth assignments and implication in negation-free formulae. Ann Math Artificial Intelligence 18(1):51–67
6. Du D-Z, Gu J, Pardalos PM (1997) Satisfiability problem: Theory and applications. Amer. Math. Soc., Providence, RI
7. Hochbaum DS (1997) Approximation algorithms for NP-hard problems. PWS, Boston, MA
8. Immerman N, Kolaitis PG (eds) (1997) Descriptive complexity and finite models. Amer. Math. Soc., Providence, RI
9. Jablonskij SV (1958) Funkcional'nye postrojenija v k-značnoj logike. Trudy Mat Inst Steklov 51:5–142
10. Klement EP, Mesiar R (1997) Triangular norms. In: Mesiar R, Riečan B (eds) Fuzzy Structures – Current Trends, Tatra Mountains Math. Publ. (Special Issue), vol 13. Math. Inst. Slovak Acad. Sci., Bratislava, pp 169–194
11. Kohout L (1974) The Pinkava many-valued complete logic systems and their application to the design of many-valued switching circuits. In: Rine DC (ed) Proc. 1974 Internat. Symp. Multiple-Valued Logic (West Virginia Univ., May, 1974). IEEE, New York, pp 261–284
12. Kohout LJ (1976) Application of multi-valued logics to the study of human movement disorders. In: Proc. Sixth Internat. Symp. Multiple-Valued Logic. IEEE, New York, pp 224–232
13. Kohout LJ (1990) A perspective on intelligent systems: A framework for analysis and design. Chapman and Hall and v. Nostrand, London–New York
14. Kohout LJ (1995) Epistemological aspects of many-valued logics and fuzzy structures. In: Höhle U, Klement EP (eds) Non-Classical Logics and their Applications to Fuzzy Subsets: A Handbook of Mathematical Foundations of Fuzzy Set Theory. Kluwer, Dordrecht, pp 291–339
15. Kohout LJ, Gaines BR (1975) The logic of protection. In: Lecture Notes Computer Sci, vol 34. Springer, Berlin, pp 736–751
16. Kohout LJ, Gaines BR (1976) Protection as a general systems problem. Internat J General Syst 3:1–21
17. Kohout LJ, Pinkava V (1976) The functional completness of Pi-algebras and its relevance to technological applications and biological modelling. In: Mamdani EH, Gaines BR (eds) Discrete Systems and Fuzzy Reasoning (Workshop Proc.). Queen Mary College Univ. London, London, pp 110–124
18. Kohout LJ, Pinkava V (1980) The algebraic structure of the Spencer–Brown and Varela calculus. Internat J General Syst 6(3):155–171
19. Krajíček J (1995) Bounded arithmetic, propositional logic, and complexity theory. Cambridge Univ. Press, Cambridge
20. Manber U (1989) Introduction to algorithms. Addison-Wesley, Reading, MA
21. Pinkava V (1966) Logické modely sexualních deviací v objektu (Models of sexual deviations based on many-valued logics). PhD Thesis Charles Univ. Prague
22. Pinkava V (1971) A logical model of some schizophrenic thought impairments. Internat J Man-Machine Studies 3:81–97
23. Pinkava V (1971) Logical models of sexual deviations. Internat J Man-Machine Studies 3:351–374
24. Pinkava V (1971) On a set of functionally complete systems of functors for k-valued logical calculi. Preprint Severalls Hospital, Colchester, UK
25. Pinkava V (1974) A family of complete sets of functions in k-valued logics allowing easy generating. IEEE Comput Repos R74:1–24
26. Pinkava V (1975) Some further properties of the Pi-logics. Proc. 1975 Internat. Symp. Multiple-Valued Logic. IEEE, New York20–26
27. Pinkava V (1976) Arrangements of formulas and minimisation in Pi-algebras. In: Mamdani EH, Gaines BR (eds) Discrete Systems and Fuzzy Reasoning (Workshop Proc.). Queen Mary College Univ. London, London
28. Pinkava V (1976) Fuzzification of binary and finite multivalued logical calculi. Internat J Man-Machine Studies 8:717–730
29. Pinkava V (1977) On the nature of some logical paradoxes. Internat J Man-Machine Studies 9:383–398
30. Pinkava V (1978) On a class of functionally complete multivalued logical calculi. Studia Logica 2:201–212
31. Pinkava V (1978) Variable structure automata. In: Internat. Symp. General Systems Res., Cleveland State Univ., pp 1–26
32. Pinkava V (1979) On Sheffer functions in k-valued logical calculi. Colloq Math Soc János Bolyai, vol 28. North-Holland, Amsterdam, pp 537–545
33. Pinkava V (1979) On some manipulative properties of PI-algebras. In: Proc. Ninth Internat. Symp. Multiple-Valued Logic (Bath, England, 29-31 May 1979). IEEE, New York, pp 139–142
34. Pinkava V (1980) A multivalued logical net modelling conditioning. Behavioral and Brain Sci 3:461–462. Open Peer Commentary on Eysinck HJ (1979) The conditioning model of neurosis, ibid 2:459–482.
35. Pinkava V (1980) On potential tautologies in k-valued calculi. In: Wang PP (ed) Fuzzy Sets: Theory and Applications to Policy Analysis and Information systems. Plenum, New York, pp 77–86
36. Pinkava V (1981) The dichotomous predicament of contemporary psychology. Behavioral and Brain Sci 4:546–547. Open Peer Commentary on Colby KM: Modeling a paranoid mind, 534–550.
37. Pinkava V (1986) Classification and diagnostics. In: Kohout LJ, Bandler W (eds) Knowledge Representation in Medicine and Clinical Behavioural Sci. Abacus Book. Gordon and Breach, New York, pp 69–94
38. Pinkava V (1988) Introduction to logic for system modelling. Gordon and Breach, New York
39. Pinkava V (1993) Multiple-valued logics and classification criteria. J Fuzzy Logic and Intelligent Systems 3(1):90–96,

special issue: Festschrift in Wyllis Bandler's Honor, Zadeh LA, et al (eds)
40. Pitassi T (1997) Algebrais propositional proof systems. In: Immerman N, Kolaitis PG (eds) Descriptive Complexity and Finite Models. Amer. Math. Soc., Providence, RI, pp 215–244
41. Rosenberg I (1965) La structure des functions de plusieurs variables sur un ensemble fini. C R Acad Sci Paris 260:3817–3819
42. Rosenberg I (1970) Über die functionale Vollstandigkeit in den mehrvertigen Logiken. Rozpravy ČSAV, Natural Sci Ser (Prague) 80(4):1–93
43. Salomaa A (1959) On many-valued systems of logic. Ajatus 22:115–159
44. Schweizer B, Sklar A (1983) Probabilistic metric spaces. North-Holland, Amsterdam
45. Slupecki J (1939) A criterion of completness of many-valued systems of propositional logic. CR Soc Sci Letters Varsovie 32:102–110. (In Polish.)

First Order Constraint Qualifications

Diethard Klatte
Institute OR, University Zurich, Zurich, Switzerland

MSC2000: 90C30, 49K27, 90C31, 49K40

Article Outline

Keywords
Optimality Conditions
Duality
Stability
Metric Regularity
Error Bounds
See also
References

Keywords

Constraint qualification; Optimality conditions; Duality; Regularity; Stability

A *constraint qualification* (CQ) is a condition imposed on the analytical description of a given set which can be the constraint set of an optimization problem. CQs are essential in order to establish optimality conditions, but they play also a crucial role in duality theory and perturbation analysis for optimization problems, and in the study of error bounds and stability for algebraic systems like systems of equations or/and inequalities. The notion *first order constraint qualification* is used if a CQ is formulated in terms of first order derivatives or generalized derivatives of the data functions defining the (constraint) set, or if it is related to optimality or stability conditions involving first order terms of the original data. Roughly speaking, first order constraint qualifications establish a link between the geometry of the given set and certain kinds of first order approximations of the analytical data.

A canonical form of constraints for which constraint qualifications have been studied is, for example, the constraint system of a mathematical programming problem, i. e.,

$$\begin{cases} g_i(x) \le 0, & i \in I = \{1, \dots, m\}, \\ g_j(x) = 0, & j \in J = \{m+1, \dots, r\}, \end{cases} \tag{1}$$

where $g_i : \mathbf{R}^n \to \mathbf{R}$ $(i = 1, \dots, r)$ are given functions, possibly restricted to some subset $X \subset \mathbf{R}^n$.

Another canonical form are *abstract constraints*,

$$G(x) \in C, \tag{2}$$

where G maps a Banach space X into a Banach space Y, and C is a nonempty closed convex cone in Y. Many of the results reported below similarly hold (with some technical modifications) under the weaker assumption that $C \subset Y$ is an arbitrary closed convex set [4,5,8,40,42]. The inclusion (2) is suitable to represent also constraints of abstract optimal control problems, semi-infinite programs, semidefinite optimization problems, and others, see, e. g., [5]. Obviously, (1) is a special case of (2), put $X = \mathbf{R}^n$, $Y = \mathbf{R}^r$, $C = \{y : y_i \le 0, i \in I; y_j = 0, j \in J\}$ and $G = (g_1, \dots, g_r)$.

The notion 'constraint qualification' was introduced by H.W. Kuhn and A.W. Tucker [22] in developing the theory of nonlinear programming. However, under the name *regularity conditions*, description-depending assumptions were known already in classical variational and extremum problems. To illustrate the meaning of first order CQs, let us consider a simple example:

Example 1

$$\begin{cases} \min & \frac{1}{2}x^2 + y \\ \text{s.t.} & x - y = 0. \end{cases}$$

The classical Lagrange conditions $x + u = 0$, $1 - u = 0$, $x - y = 0$ are necessary (and sufficient, in this example) for the optimality of the point $(\overline{x}, \overline{y}) = (-1, -1)$ with associated multiplier $\overline{u} = 1$. On the other hand, if the constraint is equivalently written as

$$\frac{1}{2}(x-y)^2 = 0,$$

then the corresponding Lagrange conditions become $x + u(x-y) = 0$, $1 - u(x-y) = 0$, $(x-y)^2/2 = 0$, which are contradictory. Trivially, in the first description of the feasible set, the linearization adequately represents the possibilities for variation near $(\overline{x}, \overline{y})$, in the second description, the linearization is inadequate in this respect.

Optimality Conditions

First order necessary optimality conditions in dual form require certain CQs to hold. Consider the optimization problem

$$(\mathrm{P}) \quad \begin{cases} \min & f(x) \\ \text{s.t.} & x \in M, \end{cases}$$

where M is the solution set of (1). First suppose that $X = \mathbf{R}^n$, $f\colon \mathbf{R}^n \to \mathbf{R}$, and g_i $(\forall i)$ are continuously differentiable. For $\overline{x} \in M$ define $I_{\overline{x}} := \{i \in I\colon g_i(\overline{x}) = 0\}$, write $h \in T_M(\overline{x})$ (tangent cone) if $h = \lim_{k\to\infty} \lambda_k(x^k - \overline{x}$, where $\lambda_k > 0$, $x^k \in M$ $(\forall k)$ and $x^k \to \overline{x}$, and write $h \in K_M(\overline{x})$ (linearization cone) if $\langle h, Dg_i(\overline{x})\rangle \le 0$ for $i \in I_{\overline{x}}$ and $\langle h, Dg_j(\overline{x})\rangle = 0$ for $j \in J$.

Then the *Karush–Kuhn–Tucker conditions* (KKT),

$$\begin{cases} \exists u \in \mathbb{R}^r\colon & Df(\overline{x}) + \sum\limits_{i\in I\cup J} u_i Dg_i(\overline{x}) = 0, \\ & \overline{x} \in M, \\ & u_i \ge 0, \quad u_i g_i(\overline{x}) = 0, \quad i \in I, \end{cases}$$

are necessary for $\overline{x}$ being a local minimizer of (P), provided that, for example, one of the following CQs is satisfied (see, e. g., [2]):

- *Abadie CQ*: $T_M(\overline{x}) = K_M(\overline{x})$.
- *Kuhn–Tucker CQ*: For every $h \in K_M(\overline{x})$ there is a continuously differentiable function $y\colon [0, \delta) \to M$, $\delta > 0$, such that $y(0) = \overline{x}$ and $\dot{y}(0) = h$.
- *Mangasarian–Fromovitz CQ* (MFCQ, [28]): $Dg_j(\overline{x})$, $j \in J$, are linearly independent, and for some $h \ne 0$, there holds $\langle h, Dg_i(\overline{x})\rangle < 0$, $i \in I_{\overline{x}}$, and $\langle h, Dg_j(\overline{x})\rangle = 0, j \in J$.
- *Linear Independence CQ* (LICQ): $Dg_i(\overline{x})$, $i \in I \cup J$, are linearly independent.

There holds (see, e. g., [2]): LICQ $\Rightarrow$ MFCQ $\Rightarrow$ Kuhn–Tucker CQ $\Rightarrow$ Abadie CQ; the converse implications are not true, in general. For further CQs in this respect, see [2,38]. If no inequalities appear (i. e., $I = \emptyset$), the above CQs are classical for optimality conditions in Euler–Lagrange form. Note that Abadie's CQ is automatically satisfied at each point of M if g_i are affine-linear for all indices $i \in I \cup J$.

Now suppose that (P) is a convex program, i. e., g_i, $i \in I$, are convex (but not necessarily differentiable) functions and g_j are affine-linear functions with gradients a_j, $j \in J$. Then the following CQs are often used for optimality conditions of Karush–Kuhn–Tucker type (in subdifferential terms) and saddle-point conditions ([16,36,41]):

- *Basic CQ* at $\overline{x} \in M$: Each normal direction h, i. e., $\langle h, x - \overline{x}\rangle \le 0$ for all $x \in M$, has a representation $h = \sum_{i\in I_{\overline{x}}} \mu_i d_i + \sum_{j\in J} \lambda_j a_j$ for some $\lambda \in \mathbf{R}^m$, $\mu_i \ge 0$, $d_i \in \partial g_i(\overline{x})$ (for $i \in I_{\overline{x}}$), where $\partial g_i(\overline{x})$ denotes the *Moreau–Rockafellar subdifferential* of g_i at $\overline{x}$.
- *Weak Slater CQ*: $\exists x^0 \in M$ such that $g_i(x^0) < 0$, $i \in I_N$, where $I_N := \{i \in I\colon g_i$ is not affine-linear$\}$.
- *Strong Slater CQ*: $\exists x^0 \in M$ such that $g_i(x^0) < 0$, $i \in I$, is satisfied, and a_j, $j \in J$, are linearly independent.

The latter naming of CQs is taken from [16]. If no equations appear, the strong Slater CQ becomes the well-known and classical Slater CQ [2,36,41]. There holds: weak Slater CQ $\Rightarrow$ basic CQ; and for a given $\overline{x} \in M$, the basic CQ is equivalent to a nonsmooth form of the Abadie CQ [16]. If the g_i, $i \in I$, are differentiable, then the strong Slater CQ is equivalent to the MFCQ being satisfied at any $x \in M$ [33,34]. There are certain forms of first order optimality conditions which do not require a CQ, see, e. g., [2,3,32,38].

Next, consider

$$(\widehat{\mathrm{P}}) \quad \begin{cases} \min & f(x) \\ \text{s.t.} & x \in M, \end{cases}$$

where M is the solution set of (2), and f is defined on the Banach space X. Let f, G be continuously differentiable.

Denote by Y^* the dual space of Y. Then the conditions

$$\begin{cases} \exists u \in Y^* : & Df(\overline{x}) + \langle u, DG(\overline{x})\rangle = 0, \\ & \langle u, y\rangle \le 0, \quad \forall y \in C, \\ & G(\overline{x}) \in C, \\ & \langle u, G(\overline{x})\rangle = 0, \end{cases}$$

are necessary for $\overline{x}$ being a local minimizer of $(\widehat{\mathrm{P}})$, provided that, for example, the following CQ is satisfied (see, e. g., [5,34,42]):

- *Robinson CQ*: $0 \in \mathrm{int}\{G(\overline{x}) + DG(\overline{x})X - C\}$, where 'int' denotes the topological interior.

Because the core of a convex set includes its interior, $0 \in \mathrm{core}\{G(\overline{x}) + DG(\overline{x})X - C\}$ is a consequence of the Robinson CQ. In fact, the latter is also sufficient for Robinson's CQ to hold, and both CQs are also equivalent to $\mathbb{R}_+[G(\overline{x}) + DG(\overline{x})X - C] = Y$, for details one may consult [5,33,42]. If $(\widehat{\mathrm{P}})$ is specialized to (P), then the Robinson CQ and MFCQ are equivalent [34]. Under convexity assumptions on f and G in $(\widehat{\mathrm{P}})$, an extension of the strong Slater CQ plays a crucial role for first order optimality characterizations [37] (see also [40]): $0 \in \mathrm{int}(G(x) - C)$, which becomes $G(x) \in \mathrm{int}C$ if $\mathrm{int}C \neq \emptyset$. In the case of differentiable data, the latter CQ is equivalent to the Robinson CQ [33,40].

For many other classes of optimization problems, first order CQs in connection with optimality conditions have been intensively studied. Among them we refer to CQs in composite optimization [38], optimal control problems [7,17,31], nonsmooth (nonconvex) programs [7,38], mathematical programs with equilibrium constraints [27], semidefinite programs [39], and semi-infinite programs [5,15,31,32]. Certain first order CQs, in particular, Robinson's CQ and the MFCQ play an important role in the theory of second order optimality conditions (and second order stability analysis), see ► Second order constraint qualifications and, e. g., [3,4,8,39,40].

Duality

If (P) is a convex program, then first order CQs are closely related to the existence of optimal solutions of the Lagrange dual problem (D) associated with (P) and to properties of the perturbation function $v(u) := \inf\{f(x)\colon g_i(x) \le u_i,\ i \in I,\ g_j(x) = u_j,\ j \in J\}$, like continuity or subdifferentiability [10,36,37]. An important CQ is

- *Calmness*: $v(0)$ is finite and the Moreau–Rockafellar subdifferential $\partial v(0)$ of $v(\cdot)$ is nonempty.

Under calmness, the dual problem (D) is solvable and $v(0)$ coincides with the optimal value of (D) [10,36,37]. The strong Slater CQ implies calmness. If $v(0)$ is finite, then the following three conditions are mutually equivalent:

i) For (1) the strong Slater CQ holds;
ii) $v(\cdot)$ is continuous at 0;
iii) the set of solutions of the dual problem (D) is nonempty and bounded.

For more details see, e. g., [1,33,36]; for generalizations to convex problems $(\widehat{\mathrm{P}})$ with abstract constraints of the type (2) see, e. g., [33,37,40].

Now suppose that (P) has continuously differentiable data f, g_i, and $\overline{x}$ is a stationary solution solution of (D), i. e., $\overline{x}$ satisfies together with some multiplier u the KKT condition. Then, obviously, LICQ implies that the multiplier u associated with $\overline{x}$ is unique. In [25] is shown that a strengthened form of MFCQ, the so-called strict MFCQ, is necessary and sufficient for the uniqueness of the Lagrange multiplier. Another basic result is the following: MFCQ holds at $\overline{x}$ if and only if the set of all multipliers associated with $\overline{x}$ is bounded (*Gauvin's theorem* [13]).

Extensions of Gauvin's theorem to the general problem $(\widehat{\mathrm{P}})$ with smooth data can be found, e. g., in [8,34,42]. For recent surveys of several aspects of CQs and duality, see [5,40].

The above relations are also important theoretical tools for establishing solution techniques which use Lagrangians or dual schemes (see, e. g., [16,29,38]), for convergence analysis of path following methods (see, e. g., [14]), for regularity properties guaranteeing finite termination of algorithms [6,11], and for several stability subjects (see the next section).

Stability

If the data couples (f, g) of (P) or (f, G) of $(\widehat{\mathrm{P}})$, respectively, are embedded in a family **F** of data, where a 'distance' between two elements of **F** should be available, then the question arises how changes of the data in **F** affect existence of solutions (local or global optimizers, stationary solutions, critical points), and whether 'small' data perturbations lead to 'small' changes of the

optimal value and solution set, or not. A 'good' stability behavior is often sensitive to the description of the constraint set and needs a CQ.

For example, consider a parametric smooth program with finite-dimensional variables x and canonical perturbations (t, a, b) in a finite-dimensional space, namely,

$$(\mathrm{P}(t,a,b)) \quad \begin{cases} \min & f(t,x) - \langle a, x\rangle \\ \text{s.t.} & g_i(t,x) \le b_i, \quad i \in I, \\ & g_j(t,x) = b_j, \quad j \in J, \end{cases} \tag{3}$$

with respect to x, where I, J are as above and f, g_i are twice continuously differentiable with respect to (t, x). Given an initial parameter triple $(\bar{t}, 0, 0)$ and a KKT point $\bar{z} = (\bar{x}, \bar{u})$ of the initial problem, then strong stability of $\bar{z}$ (i. e., the existence of a locally unique and Lipschitzian solution $z(t, a, b)$ of the perturbed KKT system near $\bar{z}$) necessarily requires LICQ to hold at $\bar{x}$ [23], while LICQ together with some strong second order optimality condition characterizes strong stability [9,21,23,35].

MFCQ and the strong Slater CQ are very important to get other stability properties like strong stability, pseudoregularity, upper Lipschitz (or Hölder) continuity, or upper semicontinuity of the optimal and/or stationary solution maps under perturbations (see also the next section). LICQ and MFCQ, respectively, play an essential role for existence, representations and estimates of directional derivatives (studied in different forms: standard one-sided directional derivative, semiderivative, Dini type, Hadamard type, and others) of the optimal value function. For an introduction into these interrelations, see [1,5,9,12,19,24,38], while [5,40] also survey extensions to the class $(\widehat{\mathrm{P}})$, under the Robinson CQ.

In the study of structural (or global) stability of feasible sets and nonlinear finite/semi-infinite programs, including one-parametric deformations, MFCQ and its extensions turn out to be fundamental in these settings, in particular, they characterize the global stability of compact feasible sets, for surveys see [14,18].

If one is interested in directional stability of optimal values or optimal solutions under data perturbations, another type of CQs often comes into play: directional regularity conditions which are imposed on the constraint set of the initial problem $\mathrm{P}((\bar{t}, 0, 0))$. A typical example is the

- *Gollan CQ* at a feasible $\bar{x}$ in direction d: $Dg_j(\bar{t}, \bar{x})$, $j \in J$, are linearly independent, and for some $h \neq 0$, there holds $\langle (h, d), Dg_i(\bar{t}, \bar{x})\rangle < 0$, $i \in I_{(\bar{t},\bar{x})}$, and $\langle (h, d), Dg_j(\bar{t}, \bar{x})\rangle = 0, j \in J$.

For this CQ and a natural extension to abstract constraints in Banach spaces, see [4,5,40], in which directional differentiability and second order expansion of the optimal-value function as well as Lipschitz/Hölder stability and first order expansion of optimal solutions under directional regularity conditions are studied.

Metric Regularity

Metric regularity of a parametric constraint system refers to a certain local error bound for the distance of some point x to the solution set in terms of the residuum of the data at x and is closely related to first order CQs. Consider for example system (1) with $X = \mathbf{R}^n$ under right-hand side perturbations b,

$$\begin{cases} g_i(x) \le b_i, \quad i \in I, \\ g_j(x) = b_j, \quad j \in J, \end{cases} \tag{3}$$

where I and J are as above. Denote by $S(b)$ the solution set mapping of this system. If g_i, $\forall i \in I$, are continuously differentiable, and if $\bar{x} \in S(0)$, then MFCQ at $\bar{x}$ implies that (4) is *metrically regular* at $(\bar{x}, 0)$, i. e., there exist a neighborhood U of $(\bar{x}, 0)$ and a constant $L = L(\bar{x})$ such that for $(x, b) \in U$,

$$\operatorname{dist}(x, S(b)) \le L \left\| \begin{pmatrix} (g_i(x) - b_i)_+, & i \in I \\ g_j(x) - b_j, & j \in J \end{pmatrix} \right\|, \tag{5}$$

where $c_+ := \max\{c, 0\}$ for $c \in \mathbf{R}$, and $\|\cdot\|$ is any norm in $\mathbf{R}^r$. This was shown in [34], the converse assertion is also true [8,34]. In the Banach space context of the system (2) with right-hand side perturbations, the same equivalence holds when replacing MFCQ by the Robinson CQ, see again [8,34].

If g_i, $i \in I$, are convex (not necessarily differentiable), and g_j, $j \in J$, are affine-linear, then the strong Slater CQ implies that (4) is metrically regular at each $(\bar{x}, 0)$, $\bar{x} \in S(0)$, see [33]. The converse direction is also true [8]. In fact, in both [8] and [33], the authors prove these results for convex inclusions in the Banach space setting (2) using a suitable generalized form of the Slater CQ. Moreover, under the strong Slater CQ, the estimate

(5) even holds for all x in X and all b near 0, where the number $L = L(\overline{x})$ is bounded on bounded sets [33].

Note that under mild assumptions on the parameter-dependence, the same CQs imply metric regularity under more general perturbations (like in (3), for example), for details see again [8,24,33,34].

Error Bounds

The role of (first order) CQs for deriving local and global error bounds will be now discussed for a system of convex inequalities, i. e., suppose in (1) that $J = \emptyset$, and g_i, $i \in I$, are convex (not necessarily differentiable) functions. Denote the solution set again by M.

Given $\overline{x} \in M$, the system (1) is called *metrically regular** at $\overline{x}$ (or simply *metrically regular* at $\overline{x}$ [26,30]; the asterisk is used to avoid confusions with the above notion for parametric systems), if there exist a neighborhood U of $\overline{x}$ and a constant $L = L(\overline{x})$ such that for $x \in U$,

$$\operatorname{dist}(x, M) \leq L \max \{g_i(x)_+ : \ i \in I\}. \tag{6}$$

In [26] was shown that for differentiable functions g_i, $i \in I$, the Abadie CQ holds at $\overline{x} \in M$ if and only if (1) is metrically regular* at $\overline{x}$ [26]. For the nondifferentiable case, it follows by a similar idea of proof that the basic CQ is equivalent to metric regularity*.

If M is bounded and the strong Slater CQ holds, then (6) is satisfied for all $x \in \mathbf{R}^n$, with some uniform constant L [33]. This property is called a *global error bound*, or, an 'error bound in Hoffman's sense' [20,26,30]. If M is unbounded then additional *asymptotic CQs* are required to guarantee the existence of a global error bound. For a survey of asymptotic CQs and their interrelations, see [20].

CQs like Abadie's CQ, MFCQ and the (strong) Slater CQ are also essential in deriving local and global error bounds for approximate solutions of convex and nonconvex mathematical programs and other variational problems. These questions are for some classes of programs closely related to the existence of so-called weak sharp minima introduced in [6,11]. For a general survey of error bounds in the sense just discussed see [30], for the special case of quadratic convex programs see [26].

In contrast to other applications of CQs, the relations between CQs and error bounds are still not clarified completely and require strong additional effort in research.

See also

- ▶ Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- ▶ Inequality-constrained Nonlinear Optimization
- ▶ Kuhn–Tucker Optimality Conditions
- ▶ Lagrangian Duality: Basics
- ▶ Rosen's Method, Global Convergence, and Powell's Conjecture
- ▶ Saddle Point Theory and Optimality Conditions
- ▶ Second Order Constraint Qualifications
- ▶ Second Order Optimality Conditions for Nonlinear Optimization

References

1. Bank B, Guddat J, Klatte D, Kummer B, Tammer K (1982) Non-linear parametric optimization. Akad. Verlag, Berlin
2. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: Theory and algorithms, 2nd edn. Wiley, New York
3. Ben-Israel A, Ben-Tal A, Zlobec S (1981) Optimality in nonlinear programming: A feasible direction approach. Wiley, New York
4. Bonnans JF, Cominetti R (1996) Perturbed optimization in Banach spaces I, II, III. SIAM J Control Optim 34:1151–1171; 1172–1189; 1555–1567
5. Bonnans JF, Shapiro A (2000) Perturbation analysis of optimization problems. Springer, Berlin
6. Burke JV, Ferris MC (1993) Weak sharp minima in mathematical programming. SIAM J Control Optim 31:1340–1359
7. Clarke F (1983) Optimization and nonsmooth analysis. Wiley, New York
8. Cominetti R (1990) Metric regularity, tangent sets and second-order optimality conditions. Appl Math Optim 21:265–287
9. Dontchev A, Rockafellar RT (1997) Characterizations of Lipschitz stability in nonlinear programming. In: Fiacco AV (ed) Mathematical Programming with Data Perturbations. M. Dekker, New York, pp 65–82
10. Ekeland I, Temam R (1974) Analyse convexe et problèmes variationnels. Dunod, Paris
11. Ferris MC, Mangasarian OL (1992) Minimum principle sufficiency. Math Program 57:1–14
12. Fiacco AV (1983) Introduction to sensitivity and stability analysis. Acad. Press, New York
13. Gauvin J (1977) A necessary and sufficient regularity condition to have bounded multipliers in nonconvex programming. Math Program 12:136–138

14. Guddat J, Guerra F, Jongen HTh (1990) Parametric optimization: Singularities, pathfollowing and jumps. Wiley, New York
15. Hettich R, Kortanek K (1993) Semi-infinite programming: Theory, methods, and applications. SIAM Rev 35:380–429
16. Hiriart-Urruty J-B, Lemaréchal C (1993) Convex analysis and minimization algorithms I, II. Springer, Berlin
17. Ioffe AD, Tichomirov VM (1974) Theory of extremal problems. Nauka, Moscow
18. Jongen HTh, Rückmann Jan-J (1998) On stability and deformation in semi-infinite optimization. In: Reemtsen R, Rückmann J-J (eds) Semi-Infinite Programming. Kluwer, Dordrecht, pp 29–67
19. Klatte D (1994) On quantitative stability for non-isolated minima. Control and Cybernetics 23:183–200
20. Klatte D, Li W (1999) Asymptotic constraint qualifications and global error bounds for convex inequalities. Math Program 84:137–160
21. Kojima M (1980) Strongly stable stationary solutions on nonlinear programs. In: Robinson SM (ed) Analysis and Computation of Fixed Points. Acad. Press, New York, pp 93–138
22. Kuhn HW, Tucker AW (1950) Nonlinear programming. In: Neyman J (ed) Proc. Second Berkeley Symp. Math. Stat. Probab. Univ. Calif. Press, Berkeley, CA, pp 481–492
23. Kummer B (1991) Lipschitzian inverse functions, directional derivatives and application in $C^{1,1}$ optimization. J Optim Th Appl 70:559–580
24. Kummer B (1997) Lipschitzian and pseudo-Lipschitzian inverse functions and applications to nonlinear programming. In: Fiacco AV (eds) Mathematical Programming with Data Perturbations. M. Dekker, New York, pp 201–222
25. Kyparisis J (1985) On uniqueness of Kuhn–Tucker multipliers in nonlinear programming. Math Program 32:242–246
26. Li W (1997) Abadie's constraint qualification, metric regularity, and error bounds for differentiable convex inequalities. SIAM J Optim 7:966–978
27. Luo Z-Q, Pang J-S, Ralph D (1996) Mathematical programs with equlibrium constraints. Cambridge Univ. Press, Cambridge
28. Mangasarian OL, Fromovitz S (1967) The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. J Math Anal Appl 17:37–47
29. Nesterov Y, Nemirovskii A (1994) Interior-point polynomial algorithms in convex programming. SIAM Stud Appl Math. SIAM, Philadelphia
30. Pang J-S (1997) Error bounds in mathematical programming. Math Program 79:299–332
31. Polak E (1997) Optimization: Algorithms and consistent approximations. Springer, Berlin
32. Pshenichnyi BN (1969) Necessary conditions for an extremum. Nauka, Moscow
33. Robinson SM (1976) Regularity and stability for convex multivalued functions. Math Oper Res 1:130–143
34. Robinson SM (1976) Stability theorems for systems of inequalities. Part II: Differentiable nonlinear systems. SIAM J Numer Anal 13:497–513
35. Robinson SM (1980) Strongly regular generalized equations. Math Oper Res 5:43–62
36. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton
37. Rockafellar RT (1974) Conjugate duality and optimization. Regional Conf Ser Applied Mat. SIAM, Philadelphia
38. Rockafellar RT, Wets RJ-B (1997) Variational analysis. Springer, Berlin
39. Shapiro A (1997) First and second order analysis of nonlinear semidefinite programs. Math Program 77:301–320
40. Shapiro A (1998) First and second order optimality conditions and perturbation analysis of semi-infinite programming problems. In: Reemtsen R, Rückmann J-J (eds) Semi-Infinite Programming. Kluwer, Dordrecht, pp 103–133
41. Stoer J, Witzgall C (1970) Convexity and optimization in finite dimensions I. Springer, Berlin
42. Zowe J, Kurcyusz S (1979) Regularity and stability for the mathematical programming problem in Banach spaces. Appl Math Optim 5:42–62

Flow Shop Scheduling Problem

MAGDALENE MARINAKI
Department of Production Engineering and Management, Industrial Systems Control Laboratory, Technical University of Crete, Chania, Greece

MSC2000: 68M20, 90B35

Article Outline

Introduction
Variations
Exact Algorithms for the Flow Shop Scheduling Problem
Heuristic Algorithms for the Flow Shop Scheduling Problem
Metaheuristic Algorithms for the Flow Shop Scheduling Problem
References

Introduction

The general flow-shop problem [4,60,68], denoted as $n/m/C_{\max}$ in the literature, involves n jobs, each re-

quiring operations on m machines, in the same machine sequence. The processing time for each operation is p_{ij}, where $i \in \{1, 2, \ldots, n\}$ denotes a job and $j \in \{1, 2, \ldots, m\}$ a machine. The problem is to determine the sequence of these n jobs that produces the smallest makespan assuming no preemption of operations. In the simplest case, all jobs are available and ready to start at time zero and the setup times on machines are assumed to be sequence-independent and included in p_{ij}. In more realistic situations, however, jobs are released at different times, thus requiring dynamic scheduling and the setup times are sequence-dependent.

The makespan problem for flow shops has been the most studied by far in the literature. (The makespan $C_{\max}$ is equivalent to the completion time of the last job to leave the system.) This is partly because:

- Makespan is a simple and useful criterion for heavily loaded shops when long-term utilization should be maximized.
- Makespan is the only objective function simple enough to have available some analytic results for multimachine problems and to make some branch-and-bound methods practical for medium-sized problems.

The minimization of the makespan objective is to a certain extent equivalent to the maximization of the utilization of the machines. The models, however, tend to be of such complexity that makespan results are already relatively hard to obtain. Even harder to analyze are the flow time and the due-date-related objectives.

Variations

There are a number of variations for the flow shop scheduling problem [60,68]. Some of them are presented in the following.

The **permutation flow shop problem** (PFSP). A constraint that may appear in the flow-shop environment is that the queues in front of each machine operate according to the first in, first out discipline, which implies that the order (or permutation) in which the jobs go through the first machine is maintained throughout the system. This problem can be formulated as follows. Each of n jobs from the job set $j = \{1, 2, \ldots, n\}$, for $n > 1$, has to be processed on m machines $1, 2, \ldots, m$ in the order given by the indexing of the machines. Thus, job j, $j \in J$, consists of a sequence of m operations; each of them corresponding to the processing of job j on machine i during an uninterrupted processing time $p_{ij} \geq 0$. (It is assumed that a zero processing time on a machine corresponds to a job performed by that machine in an infinitesimal time.) Machine i, $i = 1, 2, \ldots, m$, can execute at most one job at a time, and it is assumed that each machine processes the jobs in the same order. We represent the job processing order by the permutation $\pi = (\pi(1), \ldots, \pi(n))$ on the set j, and we let P denote the set of all permutations on j. We wish to find the optimal processing order $\pi^* \in P$ of jobs minimizing the maximum completion time $C_{\max}(\pi)$ (makespan) [65,68].

The **flow shop scheduling problem with limited machine availability**. In such a problem, n jobs have to be scheduled on m machines under the makespan criterion and under the assumption that the machines are not available during the whole planning horizon [6].

No-wait or no-idle flow shop scheduling problems with deteriorating jobs. Deterioration of a job means that its processing time is a function of its execution start time. A simple linear deterioration function is assumed and some dominating relationships between machines can be satisfied. No-wait requirement is another phenomenon which may occur in flow shops and implies that the starting time of a job at the first machine has to be delayed to ensure that the job can go through the flow shop without having to wait for any machine. The "no-idle" constraint means that each machine, once it commences its work, has to process all operations assigned to it without any interruption. In [102] it is shown that for the problems to minimize makespan or weighted sum of completion time, polynomial algorithms still exist, although these problems are more complicated than the classical ones. In [101] the general, no-wait and no-idle flow shop scheduling problem with decreasing linear deterioration under dominant machines is considered.

A **hybrid flow shop** consists of a series of production stages, each of which has several machines operating in parallel. Some stages may have only one machine, but at least one stage must have multiple machines. The flow of jobs through the shop is unidirectional. Each job is processed by one machine in each stage and it must go through one or more stage. Machines in each stage

can be identical, uniform or unrelated. An extended survey of the problem is presented in [57].

In [45] the **stochastic flow shop problem** is presented and analyzed.

Exact Algorithms for the Flow Shop Scheduling Problem

Branch and bound is a general method for solving many types of combinatorial problems. The basic idea of branching is to conceptualize the problem as a decision tree. From each decision choice point, called a node, for a partially completed solution there grow a number of new branches, one for each possible decision. These in turn become new nodes for branching again and so on. Leaf nodes, which cannot branch any further, represent complete solutions or dead ends. A number of branch-and-bound procedures have been proposed for the solution of the flow shop scheduling problem and its variations [16,19,36,83,95,106,107]. Dynamic programming approaches for the solution of the flow shop scheduling problem have been proposed in [92,105].

Heuristic Algorithms for the Flow Shop Scheduling Problem

Since the last few decades, pure flow shop scheduling problems have been largely studied. Since the flow shop minimization problem is NP-hard [87], a number of heuristic and metaheuristic algorithms have been proposed for the solution of the problem. High-performance heuristics have been proposed to minimize the makespan [15,21,61] or the maximum tardiness [96]. Some additional characteristics have been studied for the makespan criterion: non-sequence-dependent setup and removal times [69,91], minimum time lags [91], and more recently job-precedence constraints [14]. Studies on hybrid flow shop scheduling problems are relatively recent. The main results deal with the makespan criterion, and are often limited to two stages; nevertheless, some work has been done on lateness criteria [38,43]. A number of heuristics algorithms were proposed in [13]. A worst-case analysis of heuristics is presented in [85].

In [6] a heuristic approach is proposed to approximately solve the problem that consists in scheduling the jobs two by two according to an input sequence, and using a polynomial algorithm. This algorithm is an extension of the geometric approach developed for the two-job shop scheduling problem. An algorithm that constructs heuristics that use a lower bound to find a feasible solution for the general m-stage flow shop scheduling problem with multiple operations and time lags is described in [75]. A greedy algorithm for the solution of the permutation flow shop model with variable processing times is presented in [28]. A two-stage heuristic algorithm for the flow-shop problem with multiple processors is presented in [90]. A bilevel programming heuristic is presented in [48]. A simple heuristic is presented in [55].

A two-phase heuristic is presented in [89]. In the first phase, an initial job sequence is generated using one of the available well-known and efficient heuristics, while in the second phase the sequence generated is improved in terms of the makespan using a pair exchange mechanism with directionality constraint. The n-job two-machine flow shop scheduling problem is studied in [99] with the criterion of minimizing the sum of job completion times. The scheduling problem is first formulated mathematically. Three heuristic methods are then invented to find near optimal schedules. Three lower bound generation schemata are designed to compute three different lower bounds, of which the tightest one is used. To further reduce the search space, some dominance properties are proved. Then a branch-and-bound algorithm is developed to obtain an optimal schedule. In [100] the flow shop scheduling problem, with the criterion of minimizing the sum of job completion times is addressed. Two heuristic approaches are proposed to deal with this problem. The first approach focuses on reducing machine idle times and the second one places efforts on reducing both the machine idle times and the job queue times.

Complete reviews of the heuristic and metaheuristic algorithms for the solution of the flow-shop problem and some of its variations are presented in [11,20,39,60,68,78,97].

Metaheuristic Algorithms for the Flow Shop Scheduling Problem

Several metaheuristic algorithms have been proposed for the solution of the flow shop scheduling problem. In the following an analytical presentation of these algorithms is given:

- **Simulated annealing** [1,3,50] plays a special role within local search for two reasons. First, it appears to be quite successful when applied to a broad range of practical problems. Second, some threshold-accepting algorithms such as simulated annealing have a stochastic component, which facilitates a theoretical analysis of their asymptotic convergence. Simulated annealing [2] is a stochastic algorithm that allows random uphill jumps in a controlled fashion in order to provide possible escapes from poor local optima. Gradually the probability allowing the objective function value to increase is lowered until no more transformations are possible. Simulated annealing owes its name to an analogy with the annealing process in condensed-matter physics, where a solid is heated to a maximum temperature at which all particles of the solid randomly arrange themselves in the liquid phase, followed by cooling through careful and slow reduction of the temperature until the liquid is frozen with the particles arranged in a highly structured lattice and minimal system energy. This ground state is reachable only if the maximum temperature is sufficiently high and the cooling sufficiently slow. Otherwise a metastable state is reached. The metastable state is also reached with a process known as quenching, in which the temperature is instantaneously lowered. Its predecessor is the so-called Metropolis filter. Simulated annealing algorithms for the flow shop scheduling problem are presented in [27,35,40,46,47,52,58,62,63,82,88,94,98,103].
- **Tabu search** (TS) was introduced by Glover [30,31] as a general iterative metaheuristic for solving combinatorial optimization problems. Computational experience has shown that TS is a well-established approximation technique, which can compete with almost all known techniques and which, by its flexibility, can beat many classic procedures. It is a form of local neighbor search. Each solution S has an associated set of neighbors $N(S)$. A solution $S' \in N(S)$ can be reached from S by an operation called a *move*. TS can be viewed as an iterative technique which explores a set of problem solutions, by repeatedly making moves from one solution S to another solution S' located in the neighborhood $N(S)$ of S [32]. TS moves from a solution to its best admissible neighbor, even if this causes the objective function to deteriorate. To avoid cycling, solutions that have been recently explored are declared *forbidden or tabu* for a number of iterations. The tabu status of a solution is overridden when certain criteria (*aspiration criteria*) are satisfied. Sometimes, *intensification* and *diversification* strategies are used to improve the search. In the first case, the search is accentuated in the promising regions of the feasible domain. In the second case, an attempt is made to consider solutions in a broad area of the search space. TS algorithms for the flow shop scheduling problem are presented in [5,7,8,12,26,27,37,40,46,63,66,86,104].
- **Genetic algorithms** (GAs) are search procedures based on the mechanics of natural selection and natural genetics. The first GA was developed by John H. Holland [42] in the 1960s to allow computers to evolve solutions to difficult search and combinatorial problems, such as function optimization and machine learning. GAs offer a particularly attractive approach for problems like the flow shop scheduling problem since they are generally quite effective for rapid global search of large, nonlinear and poorly understood spaces. Moreover, GAs are very effective in solving large-scale problems. GAs [34] mimic the evolution process in nature. GAs are based on an imitation of the biological process in which new and better populations among different species are developed during evolution. Thus, unlike most standard heuristics, GAs use information about a population of solutions, called individuals, when they search for better solutions. A GA is a stochastic iterative procedure that maintains the population size constant in each iteration, called a generation. The basic operation is the mating of two solutions in order to form a new solution. To form a new population, a binary operator, called crossover, and a unary operator, called mutation, are applied [72,73]. Crossover takes two individuals, called parents, and produces two new individuals, called offspring, by swapping parts of the parents. GAs for the flow shop scheduling problem are presented in [5,9,10,17,18,53,59,64,79,80,81,82].
- **Scatter search** [33] may be viewed as an evolutionary (population-based) algorithm that constructs solutions by combining others. It derives its foundations from strategies originally proposed for combining decision rules and constraints in the context

of integer programming. The goal of this method is to enable the implementation of solution procedures that can derive new solutions from combined elements in order to yield better solutions than those procedures that base their combinations only on a set of original elements. Scatter search algorithms for the flow shop scheduling problem are presented in [44,65].

- **Variable neighborhood search** is a metaheuristic for solving combinatorial optimization problems whose basic idea is systematic change of the neighborhood within a local search [41]. Variable neighborhood search algorithms for the flow shop scheduling problem are presented in [47,63].
- The use of **artificial neural networks** to find good solutions to combinatorial optimization problems has recently attracted some attention. A neural network consists of a network [67] of elementary nodes (neurons) that are linked through weighted connections. The nodes represent computational units, which are capable of performing a simple computation, consisting of a summation of the weighted inputs, followed by the addition of a constant called the threshold or bias, and the application of a nonlinear response (activation) function. The result of the computation of a unit constitutes its output. This output is used as an input for the nodes to which it is linked through an outgoing connection. The overall task of the network is to achieve a certain network configuration, for instance, a required input–output relation, by means of the collective computation of the nodes. This process is often called *self-organization*. A neural networks algorithm for the flow shop scheduling problem is presented in [86].
- An improvement heuristic based on an **adaptive learning approach** is proposed and applied to the general flow-shop problem. The approach uses a single-pass or a constructive heuristic and tries to find improvements iteratively by perturbing the data using a weight factor, allowing a nondeterministic local neighborhood search. The weights are modified using strategies similar to neural-networks training, i. e., weights are reinforced if the solution improves [4].
- **Artificial immune system** (AIS) is an intelligent problem-solving technique that has been used in scheduling problems for about 10 years. AISs are computational systems inspired by theoretical immunology, observed immune functions, principles and mechanisms in order to solve problems. Nature and in particular biological systems have always been fascinating to the human expert owing to their complexity, flexibility and sophistication. The nervous system inspired the evolution of an artificial neural network, in the very similar manner immune system motivated the emergence of the AIS. The AIS can be defined as an abstract or metamorphic computational system using ideas gleaned from the theories and component of immunology [22,23]. AIS algorithms for the flow shop scheduling problem are presented in [25,51].
- **Particle swarm optimization** (PSO) is a population-based swarm intelligence algorithm. It was originally proposed by Kennedy and Eberhart [49] as a simulation of the social behavior of social organisms such as bird flocking and fish schooling. PSO uses the physical movements of the individuals in the swarm and has a flexible and well-balanced mechanism to enhance and adapt to the global and local exploration abilities. PSO algorithms for the flow shop scheduling problem are presented in [7,8,54,56,93].
- The **ant colony optimization** (ACO) metaheuristic is a relatively new technique for solving combinatorial optimization problems. Based strongly on the ant system metaheuristic developed by Dorigo, Maniezzo and Colorni [24], ant colony optimization is derived from the foraging behavior of real ants in nature. The main idea of ACO is to model the problem as the search for a minimum cost path in a graph. Artificial ants walk through this graph, looking for good paths. Each ant has a rather simple behavior so that it will typically only find rather poor-quality paths on its own. Better paths are found as the emergent result of the global cooperation among ants in the colony. An ACO algorithm consists of a number of cycles (iterations) of solution construction. During each iteration a number of ants (which is a parameter) construct complete solutions using heuristic information and the collected experiences of previous groups of ants. These collected experiences are represented by a digital analogue of trail pheromone which is deposited on the constituent elements of a solution. Small quantities

are deposited during the construction phase, while larger amounts are deposited at the end of each iteration in proportion to solution quality. Pheromone can be deposited on the components and/or the connections used in a solution depending on the problem. ACO algorithms for the flow shop scheduling problem are presented in [29,70,71,84].

- **Greedy randomized adaptive search procedure** (GRASP) [74] is an iterative two-phase search method which has gained considerable popularity in combinatorial optimization. Each iteration consists of two phases, a construction phase and a local search procedure. In the construction phase, a randomized greedy function is used to build up an initial solution. This randomized technique provides a feasible solution within each iteration. This solution is then exposed for improvement attempts in the local search phase. The final result is simply the best solution found over all iterations. GRASP algorithms for the flow shop scheduling problem are presented in [76,77].

References

1. Aarts E, Korst J (1989) Simulated Annealing and Boltzmann Machines – A Stochastic Approach to Combinatorial Optimization and Neural Computing. Wiley, Chichester
2. Aarts E, Korst J, Van Laarhoven P (1997) Simulated Annealing. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, Chichester, pp 91–120
3. Aarts E, Ten Eikelder HMM (2002) Simulated Annealing. In: Pardalos PM, Resende MGC (eds) Handbook of Applied Optimization. Oxford University Press, pp 209–221
4. Agarwal A, Colak S, Eryarsoy E (2006) Improvement Heuristic for the Flow-Shop Scheduling Problem: An Adaptive-Learning Approach. Eur J Oper Res 169:801–815
5. Aggoune R (2004) Minimizing the Makespan for the Flow Shop Scheduling Problem with Availability Constraints. Eur J Oper Res 153:534–543
6. Aggoune R, Portmann M-C (2006) Flow Shop Scheduling Problem with Limited Machine Availability: A Heuristic Approach. Int J Prod Econ 99:4–15
7. Al-Anzi FS, Allahverdi A (2007) A Self-Adaptive Differential Evolution Heuristic for Two-Stage Assembly Scheduling Problem to Minimize Maximum Lateness With Setup Times. Eur J Oper Res 182(1):80–94
8. Allahverdi A, Al-Anzi FS (2006) A PSO and a Tabu Search Heuristics for the Assembly Scheduling Problem of the Two-Stage Distributed Database Application. Comput Oper Res 33(4):1056–1080
9. Arroyo JEC, Armentano VA (2005) Genetic Local Search for Multi-Objective Flowshop Scheduling Problems. Eur J Oper Res 167(3):717–738
10. Bagchi TP (1999) Multiobjective Scheduling by Genetic Algorithms. Kluwer, Boston
11. Bagchi TP, Gupta JND, Sriskandarajah C (2006) A Review of TSP Based Approaches for Flowshop Scheduling. Eur J Oper Res 169(3):816–854
12. Ben-Daya M, Al-Fawzan M (1998) A Tabu Search Approach for the Flow Shop Scheduling Problem. Eur J Oper Res 109:88–95
13. Botta-Genoulaz V (2000) Hybrid Flow Shop Scheduling with Precedence Constraints and Time Lags to Minimize Maximum Lateness. Int J Prod Econ 64:101–111
14. Botta V, Guinet A (1996) Scheduling flowshops with precedence constraints and time lags. Proceedings of the Workshop on Production Planning and Control, Mons, Belgium, 16–19
15. Campbell HG, Dudek RA, Smith ML (1970) A Heuristic Algorithm for the n-job, m-Machine Sequencing problem. Manag Sci 16:B630–B637
16. Carlier J, and Rebaï, I (1996) Two Branch and Bound Algorithms for the Permutation Flow Shop Problem. Eur J Oper Res 90:238–251
17. Chang PC, Chen SH, Liu CH (2007) Sub-Population Genetic Algorithm with Mining Gene Structures for Multiobjective Flowshop Scheduling Problems. Exp Syst Appl 33(3):762–771
18. Cheng BW, Chang CL (2007) A Study on Flowshop Scheduling Problem Combining Taguchi Experimental Design and Genetic Algorithm. Exp Syst Appl 32(2):415–421
19. Chung CS, Flynn J, Kirca O (2006) A Branch and Bound Algorithm to Minimize the Total Tardiness for M-Machine Permutation Flowshop Problems. Eur J Oper Res 174(1):1–10
20. Conway RW, Maxwell WL, Miller LW (2003) Theory of Scheduling. Dover Publications INC., Mineola
21. Dannenbring DG (1977) An Evaluation of Flowshop Sequencing Heuristics. Manag Sci 23(11):1174–1182
22. De Castro LN, Von Zuben FJ (1999) Artificial Immune Systems, Part 1, Basic Theory and Applications. Technical Report, TR-DCA 01/99
23. De Castro LN, Von Zuben FJ (2001) Learning and Optimization Using the Clonal Selection Principle. Trans IEEE Evol Comput 6(3):239–251
24. Dorigo M, Maniezzo V, Colorni A (1996) Ant System: Optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B 26(1):29–41
25. Engin O, Döyen A (2004) A New Approach to Solve Hybrid Flow Shop Scheduling Problems by Artificial Immune System. Future Generat Comput Syst 20:1083–1095

26. Eren T, Guner E (2006) A Bicriteria Flowshop Scheduling Problem with Setup Times. Appl Math Comput 183(2):1292–1300
27. Fink A, Voßb S (2003) Solving the Continuous Flow-Shop Scheduling Problem by Metaheuristics. Eur J Oper Res 151:400–414
28. Finke G, Jiang H (1997) A Variant of the Permutation Flow Shop Model with Variable Processing Times. Discret Appl Math 76:123–140
29. Gajpal Y, Rajendran C (2006) An Ant-Colony Optimization Algorithm for Minimizing the Completion-Time Variance of Jobs in Flowshops. Int J Prod Econ 101(2):259–272
30. Glover F (1989) Tabu Search I. ORSA J Comput 1(3):190–206
31. Glover F (1990) Tabu Search II. ORSA J Comput 2(1):4–32
32. Glover F, Laguna M, Taillard E, de Werra D (eds) (1993) Tabu Search. JC Baltzer AG, Science Publishers, Basel
33. Glover F, Laguna M, Marti R (2003) Scatter Search and Path Relinking. In: Glover F, Kochenberger GA (eds) Advances and Applications Handbook of Metaheuristics. Kluwer, Boston, pp 1–36
34. Goldberg DE (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company INC, Massachussets
35. Gourgand M, Grangeon N, Norre S (2003) A Contribution to the Stochastic Flow Shop Scheduling Problem. Eur J Oper Res 151:415–433
36. Gowrishankar K, Rajendran C, Srinivasan G (2001) Flow Shop Scheduling Algorithms for Minimizing the Completion Time Variance and the Sum of Squares of Completion Time Deviations from a Common Due Date. Eur J Oper Res 132:643–665
37. Grabowski J, Pempera J (2005) Some Local Search Algorithms for No-Wait Flow-Shop Problem with Makespan Criterion. Comput Oper Res 32:2197–2212
38. Guinet A, Solomon M (1996) Scheduling Hybrid Flowshops to Minimize Maximum Tardiness or Maximum Completion Time. Int J Prod Res 34(6):1643–1654
39. Gupta JND, Stafford EF (2006) Flowshop Scheduling Research After Five Decades. Eur J Oper Res 169(3):699–711
40. Gupta JND, Henning K, Werner F (2002) Local Search Heuristics for Two-Stage Flow Shop Problems with Secondary Criterion. Comput Oper Res 29:123–149
41. Hansen P, Mladenovic N (2001) Variable Neighborhood Search: Principles and Applications. Eur J Oper Res 130:449–467
42. Holland JH (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor
43. Hunsucker JL, Shah JR (1992) Performance of priority rules in a due-date flowshop. OMEGA Int J Manag Sci 20(1):73–89
44. Jain AS, Meeran S (2002) A Multi-Level Hybrid Framework Applied to the General Flow-Shop Scheduling Problem. Comput Oper Res 29:1873–1901
45. Jia C (1998) Minimizing Variation in Stochastic Flow Shop. Oper Res Lett 23:109–111
46. Janiak A, Kozan E, Lichtenstein M, Oğuz C (2005) Metaheuristic Approaches to the Hybrid Flow Shop Scheduling Problem with a Cost-Related Criterion. Int J Prod Econ 31(3):504–514
47. Jin Z, Yang Z, Ito T (2006) Metaheuristic Algorithms for the Multistage Hybrid Flowshop Scheduling Problem. Int J Prod Econ 100(2):322–334
48. Karlor JK, Wang W (1996) Bilevel Programming Applied to the Flow Shop Scheduling Problem. Comput Oper Res 23(5):443–451
49. Kennedy J, Eberhart R (1995) Particle Swarm Optimization. In: Proceedings of 1995 IEEE International Conference on Neural Networks 4:1942–1948
50. Kirkpatrick S, Gelatt CD, Vecchi MP (1982) Optimization by Simulated Annealing. Science 220:671–680
51. Kumar A, Prakash A, Shankar R, Tiwari MK (2005) Psycho-Clonal Algorithm Based Approach to Solve Continuous Flow Shop Scheduling Problem. Exp Syst Appl, in press
52. Laha D, Chakraborty UK (2007) An Efficient Stochastic Hybrid Heuristic for Flowshop Scheduling. Eng Appl Artif Intell 20(6):851–856
53. Leu S-S, Hwang S-T (2002) GA-Based Resource-Constrained Flow-Shop Scheduling Model for Mixed Precast Production. Automat Construct 11:439–452
54. Lian Z, Gu X, Jiao B (2006) A Similar Particle Swarm Optimization Algorithm for Permutation Flowshop Scheduling to Minimize Makespan. Appl Math Comput 175(1):773–785
55. Liao C-J, Sun C-L, You W-C (1995) Flow-Shop Scheduling with Flexible Processors. Comput Oper Res 22(3):297–301
56. Liao CJ, Tseng CT, Luarn P (2007) A Discrete Version of Particle Swarm Optimization for Flowshop Scheduling Problems. Comput Oper Res 34(10):3099–3111
57. Linn R, Zhang W (1999) Hybrid Flow Shop Scheduling: A Survey. Comput Indust Eng 37:57–61
58. Low C (2005) Simulated Annealing Heuristic for Flow Shop Scheduling Problems with Unrelated Parallel Machines. Comput Oper Res 32:2013–2025
59. Martin CH (2006) A Hybrid Genetic Algorithm/Mathematical Programming Approach to the Multi-Family Flowshop Scheduling Problem with Lot Streaming. Omega, In Press, Available online 26 December 2006. doi: 10.1016/j.omega.2006.11.002
60. Morton TE, Pentico DW (1993) Heuristic Scheduling Systems. With Applications to Production Systems and Project Management. Wiley, New York
61. Nawaz M, Enscore E, Ham I (1983) A Heuristic Algorithm for the m-Machine, n-Job Flowshop Sequencing Problem. Omega 11:91–95
62. Nearchou A (2004) A Novel Metaheuristic Approach for the Flow Shop Scheduling Problem. Eng Appl Artif Intell 17:289–300

63. Negenman EG (2001) Local Search Algorithms for the Multiprocessor Flow Shop Scheduling Problem. Eur J Oper Res 128:147–158
64. Neppali VR, Chen C-L, Gupta JND (1996) Genetic Algorithms for the Two-Stage Bicriteria Flow Shop Problem. Eur J Oper Res 95:356–373
65. Nowicki E, Smutnicki C (2006) Some Aspects of Scatter Search in the Flow-Shop Problem. Eur J Oper Res 169:654–666
66. Oğuz C, Zinder Y, Do VH, Janiak A, Lichtenstein M (2004) Hybrid Flow-Shop Scheduling Problems with Multiprocessor Task Systems. Eur J Oper Res 152:115–131
67. Sodererg B, Peterson C (1997) Artificial Neural Networks. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, Chichester, pp 173–214
68. Pinedo M (1995) Scheduling. Theory, Algorithms, and Systems. Prentice Hall, Englewood Cliffs
69. Proust C , Gupta JND, Deschamps V (1991) Flowshop Scheduling with Set-Up, Processing and Removal Times Separated. Int J Prod Res 29:479–493
70. Rajendran C, Ziegler H (2004) Ant-Colony Algorithms for Permutation Flowshop Scheduling to Minimize Makespan/Total Flowtime of Jobs. Eur J Oper Res 155(2): 426–438
71. Rajendran C, Ziegler H (2005) Two Ant-Colony Algorithms for Minimizing Total Flowtime in Permutation Flowshops Computers and Industrial Engineering. Comput Indust Eng 48(4):789–797
72. Reeves CR (1995) Genetic Algorithms. In: Reeves CR (ed) Modern Heuristic Techniques for Combinatorial Problems. McGraw-Hill, London, pp 151–196
73. Reeves CR (2003) Genetic Algorithms. In: Glover F, Kochenberger GA (eds) Handbooks of Metaheuristics. Kluwer, Dordrecht, pp 55–82
74. Resende MGC, Ribeiro CC (2003) Greedy Randomized Adaptive Search Procedures. In: Glover F, Kochenberger GA (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 219–249
75. Riezebos J, Gaalman GJC (1998) Time Lag Size in Multiple Operations Flow Shop Scheduling Heuristics. Eur J Oper Res 105:72–90
76. Rios-Mercado R, Bard J (1998) Heuristics for the Flow Line Problem with Setup Costs. Eur J Oper Res 110:76–98
77. Rios-Mercado R, Bard J (1999) An Enhanced TSP-based Heuristic for Makespan Minimization in a Flow Shop with Setup Costs. J Heuristic 5:57–74
78. Ruiz R, Maroto C (2005) A Comprehensive Review and Evaluation of Permutation Flowshop Heuristics. Eur J Oper Res 165(2):479–494
79. Ruiz R, Maroto C (2006) A Genetic Algorithm for Hybrid Flowshops with Sequence Dependent Setup Times and Machine Eligibility. Eur J Oper Res 169(3):781–800
80. Ruiz R, Maroto C, Alcaraz J (2006) Solving the Flowshop Scheduling Problem with Sequence Dependent Setup Times Using Advanced Metaheuristics. Eur J Oper Res 165(1):34–54
81. Ruiz R, Maroto C, Alcaraz J (2006) Two New Robust Genetic Algorithms for the Flowshop Scheduling Problem. Omega 34(5):461–476
82. Sadegheih A (2006) Scheduling Problem Using Genetic Algorithm, Simulated Annealing and the Effects of Parameter Values on GA Performance. Appl Math Modell 30(2):147–154
83. Sayin S, Karabati S (1999) A Bicriteria Approach to the Two-Machine Flow Shop Scheduling Problem. Eur J Oper Res 113:435–449
84. Shyu SJ, Lin BMT, Yin PY (2004) Application Of Ant Colony Optimization For No-Wait Flowshop Scheduling Problem To Minimize The Total Completion Time. Comput Indust Eng 47(2–3):181–193
85. Smutnicki C (1998) Some Results of the Worst-Case Analysis for Flow Shop Scheduling. Eur J Oper Res 109:66–87
86. Solimanpur M, Vrat P, Shankar R (2004) A Neuro-Tabu Search Heuristic for the Flow Shop Scheduling Problem. Comput Oper Res 31:2151–2164
87. Soukhal A, Oulamara A, Martineau P (2005) Complexity of Flow Shop Scheduling Problems with Transportation Constraints. Eur J Oper Res 161:32–41
88. Steinhöfel K, Albrecht A, Wong CK (2002) The Convergence of Stochastic Algorithms Solving Flow Shop Scheduling. Theoretic Comput Sci 285:101–117
89. Suliman SMA (2000) A Two-Phase Heuristic Approach to the Permutation Flow-Shop Scheduling Problem. Int J Prod Econ 64:143–152
90. Suresh V (1997) A Note on Scheduling of Two-Stage Flow Shop with Multiple Processors. Int J Prod Econ 49:77–82
91. Szwarc W (1983) Flowshop problems with time lags. Manag Sci 29:477–481
92. Tang L, Xuan H, Liu J (2006) A New Lagrangian Relaxation Algorithm for Hybrid Flowshop Scheduling to Minimize Total Weighted Completion Time. Comput Oper Res 33(11):3344–3359
93. Tasgetiren MF, Liang YC, Sevkli M, Gencyilmaz G (2007) A Particle Swarm Optimization Algorithm for Makespan and Total Flowtime Minimization in the Permutation Flowshop Sequencing Problem. Eur J Oper Res 177(3):1930–1947
94. Tian P, Ma J, Zhang D-M (1999) Application of the Simulated Annealing Algorithm to the Combinatorial Optimisation Problem with Permutation Property: An Investigation of Generation Mechanism. Eur J Oper Res 118:81–94
95. Toktaç B, Azizoğlu M, Kööksalan SK (2004) Two-Machine Flow Shop Scheduling with Two Criteria: Maximum Earliness and Makespan. Eur J Oper Res 157:286–295
96. Townsend W (1977) Sequencing n-Jobs on m-Machines to Minimize Maximum Tardiness: A Branch-and-Bound Solution. Manag Sci 23:1016–1019
97. Vallada E, Ruiz R, Minella G (2008) Minimising Total Tardiness in the M-Machine Flowshop Problem: A Review

and Evaluation of Heuristics and Metaheuristics. Comput Oper Res 35(4):1350–1373
98. Varadharajan TK, Rajendran C (2005) A Multi-Objective Simulated-Annealing Algorithm for Scheduling in Flowshops to Minimize the Makespan and Total Flowtime of Jobs. Eur J Oper Res 167(3):772–795
99. Wang C, Chu C, Proth J-M (1996) Efficient Heuristic and Optimal Approaches for n/2/F/$\sum C_i$ Scheduling Problems. Int J Prod Econ 44:225–237
100. Wang C, Chu C, Proth J-M (1997) Heuristic Approaches for n/m/F/$\sum C_i$ Scheduling Problems. Eur J Oper Res 96:636–644
101. Wang J-B (2007) Flow Shop Scheduling Problems with Decreasing Linear Deterioration Under Dominant Machines. Comput Oper Res 34(7):2043–2058
102. Wang J-B, Xia Z-Q (2006) Flow Shop Scheduling with Deteriorating Jobs under Dominating Machines. Omega 34(4):327–336
103. Wang L, Zhang L (2006) Stochastic Optimization Using Simulated Annealing with Hypothesis Test. Appl Math Comput 174(2):1329–1342
104. Wardono B, Fathi Y (2004) A Tabu Search Algorithm for the Multi-Stage Parallel Machine Problem with Limited Buffer Capacities. Eur J Oper Res 155(2):380–401
105. Xuan H, Tang L (2007) Scheduling a Hybrid Flowshop with Batch Production at the Last Stage. Comput Oper Res 34(9):2718–2733
106. Yokoyama M (2001) Hybrid Flow-Shop Scheduling with Assembly Operations. Int J Prod Econ 73:103–116
107. Yokoyama M, Santos DL (2005) Three-Stage Flow-Shop Scheduling with Assembly Operations to Minimize the Weighted Sum of Product Completion Times. Eur J Oper Res 161:754–770

Forecasting

Ulrike Leopold-Wildburger[1], Otwin Becker[2], Martin Kuehrer[3]
[1] Department of Statistics and OR, Karl-Franzens University of Graz, Graz, Austria
[2] Institute for Alfred Weber for Economics and Social Sciences, Faculty of Economics and Social Studies, University of Heidelberg, Heidelberg, Germany
[3] Siemens AG (NYSE: SI), Wien, Austria

MSC2000: 90C30, 90C26

Article Outline

Keywords
Introduction
Forecasting Methods and Models
Expert Systems
Subjective Curve Fitting
Technological Comparison
Expectations and Decision
Statistical Procedures
Moving Average Model
Regression Analysis
Econometric Models
Modern Computer Aided Techniques
Neural Networks
Fuzzy Logic
Genetic Algorithms
See also
References

Keywords

Artificial intelligence; Curve fitting; Econometrics; Expert systems; Exponential smoothing; Extrapolation; Fuzzy logic; Genetic algorithms; Inductive inference; Moving average model; Neural networks; Regression analysis; Statistics; Time series; Time series analysis; Visual inference

This article consists of three parts: first, a historical introduction to the topic, next an overview of the most frequent forecasting methods and finally a short description of modern computer-aided techniques as they are used nowadays (2000) for instance for forecasts on financial markets.

Introduction

Prediction ideas and information about uncertain future events in general are as old as humanity. Scientific forecasts are based on predetermined patterns, regularities or conformities with a (natural) law. A theoretical basis is made up of its components, the parameters of the model and the conditions for the system.

Predictions of future events are called forecasts and are concerned with the question of what the world 'will' look like [6]. Any organization must be able to make forecasts concerning their work which aim to reduce the uncertainty of the environment. For example, business firms, in particular, require forecasts for a large number of events and conditions in all phases of their operation and forecasts are indispensable for planning and strategy in everyday life.

For centuries, the nature of forecasting was the field of philosophers, who studied problems of *inductive inference* analytically in order to obtain instruments for qualitative and quantitative forecasting.

When asking the question of whether the future can be predicted, or whether it is arbitrary and random, we first noticed certain regularities in the behavior of nature. These regularities were most obvious in the fields of physics and astronomy in which it was possible to make conditional forecasts. They were given their firm mathematical basis by I. Newton, in the seventeenth century. We still use his theory of gravity which is able to predict the motion of (celestial) bodies.

At the turn into the twentieth century, psychology began to use experimental methods to investigate learning in humans and other organisms. In doing so, psychologists acquired knowledge about which behavior could be forecasted and how to reduce uncertainty as much as possible. During the twentieth century, the topic of forecasting in general became increasingly important, especially after quantitative methods had been developed. Various forecasting methods were given priority in economics and even more recently the computer has provided research tools, engendering the field of machine learning. Systematic research on trade cycles and on crises management are the first economic forecasts, at the same time as early psychological investigations.

One of the first aims of economics was to become a science which could make forecasts with the help of induction. The true measure of the value of economists is often seen as the accuracy of the forecasts they make [14].

J.H. Holland, K. Holyoak, R. Nisbett and P. Thagard in 1986 [18] gave an excellent overview of the various insights of researchers in psychology, philosophy and *artificial intelligence*. Also borrowing from several other disciplines such as engineering, *statistics*, biology and game theory, including experimental economics [20] they systematically developed principles providing coherence of a diverse set of findings on the nature of inductive processes for prospective events in the future.

Forecasting Methods and Models

Obviously there are several possibilities of classification because of various methodological approaches. Using a fundamental division, we will generate two groups of forecasting methods:

i) qualitative methods and
ii) quantitative methods.

Another main distinction consists of a generalization of similar situations which can be

i) data based (usually given in the form of a time series, a chronological sequence of observed data with respect to a certain variable) expecting that history repeats itself in a certain way and
ii) theory based, where we assume that external factors determine events.

It is natural to start with *qualitative forecasting methods* predicting future events with a certain subjective probability: on the one hand we tend to make forecasts for similar events on the basis of a certain generalization, on the other hand we try to predict new events for those situations where little or no historical data are available and for events where we expect changes within the data patterns. Generalization ideas - in a logical and methodological sense - are made on the basis that events will have properties in a certain analogy to the past and tend towards the direction of objective processes.

Here we want to recommend a well known classification by S. Makridakis and S.C. Wheelwright [21]. Our aim is to discuss a selected subset of these frequently used methods.

Expert Systems

In questions about future events, a systematic discussion of a group of five to twelve experts (*expert systems*) usually yields forecasts with a better hit-rate than individual predictions. This belongs to the class of *judgemental forecast*. Using this method, credibility is one of the most desirable features of a forecast [10].

The *Delphi method*, developed by members of the RAND Corporation in the 1960s [11], avoids face-to-face effects by using a procedure based on a questionnaire technique. Delphi therefore guarantees three basic characteristics: anonymity, interaction with controlled feedback, and statistical group responses.

Subjective Curve Fitting

A frequently used method is *subjective curve fitting and extrapolation*, which is used in economics, for exam-

ple, for forecasts on the development of products with certain life cycles or seasonal fluctuations. Experimental findings show that there are several gestaltsoriented rules dominating expectation formations [4,5]. Subjective curve fitting differs to some extent between the different subjects; frequently we are not only interested in individual expectations of a single forecasting subject but also interested in the so-called average opinion of a whole group as a good predictor for future events. For example it is a well known hypothesis that the average expected rate of inflation has an essential influence on various economic variables.

Technological Comparison

A sensible method is *technological comparison* [2] and [15]. Obviously, we should enlarge, compare or combine these qualitative methods with quantitative ones (combined forecasts), knowing that even models which best fit the available data are not necessarily the most accurate ones in predicting beyond this data [23].

Expectations and Decision

The simplest way of modeling expectations of future events, which is used frequently in economic theory, is to assume that conditions prevailing today will be maintained in all subsequent periods analogously. In cases where no causal explanation from other variables seems to be appropriate we could simply use *extrapolation methods* with the given data base to enable at least a short term forecast. These methods are successfully used

i) for seasonally adjusted data; and
ii) for cases where a continuation of the historical trend is to be expected.

Statistical Procedures

The next step is to use statistical procedures which are in some sense learning and in another sense adaptive methods. *Quantitative forecasting methods*, theory and data based, using knowledge from mathematical statistics started to be successful in the early 1960s beginning with ideas of R.G. Brown on *smoothing methods*.

In particular, *exponential smoothing* is frequently used for producing short term forecasts [8,9]. Brown suggested estimating the average of a time series and used it as an *extrapolation* for the forecast. With each new data set and observation respectively, we are able to revise the mean square error (MSE) applying exponential smoothing to the squares of the error in the most recent forecast. Several techniques have been proposed using *exponential smoothing* but it is evident that all the history of a process cannot be described by one and the same simple model.

Moving Average Model

In 1970 G.E. Box and G.M. Jenkins introduced more sophisticated forecasting models which were the first to take into account the nature of the data and the manner of the stochastic process to be forecasted. They asked not only the question of what to forecast and what data to collect, but also what data to analyze and in what context to embed the forecast. Their *moving average models* [7], enable a successful application. They popularized an approach that combines the moving average and the autoregressive approaches in [7]. The classes of autoregressive (integrated) moving averages (ARMA and ARIMA) processes have been successfully introduced by them and their models are some of the most frequently used tools for stochastic analysis.

Several ways to model multiple time series are described in [16]. Further ARIMA models are given in [23,24]; [13] gives an excellent comparison of these models using performance methods.

When enlarging the statistical methods with sensible associations and connections, *econometric methods* should be considered, if causal relationship and changes in causal variables are expected and can be estimated.

Regression Analysis

Usually it would be sensible to figure out a certain a priori relationship between the given data sets such that statistical methods of regression analysis can be used. In its simplest form, the *classical linear regression model* is used to determine an equation relating two sets of data with each other:

i) the set of observations of the explanatory or independent variables (the predictors); and
ii) the set of the associated responses, the observations of the dependent variables.

This task often seems to be easy at first sight, but when all details are concerned it becomes a high leveled task.

Obviously there are some future values which can easily be forecasted, e. g. the fuel consumption for a certain period as a relation of velocity.

As our example from the field of finance will demonstrate, there are, however, enough reasons to assume more complicated situations caused by complex systems and/or error terms. For example, demand as the variable of interest can be seen as a function of the price which takes the role of the predictor.

Econometric Models

Obviously, the standard model using the single regression equation has been extended in various ways. For example,

i) disturbances are allowed to be autocorrelated and to have different variances (heteroskedastic);
ii) by regressors measured with errors or in dynamic situations (with dependences on lagged values) stochastic regressors arise.

As proposed already at the foundation of the Econometric Society in Cleveland in 1930 we also use nowadays (2000) econometric models which are not only data based, but also theory oriented.

In econometrics, the single equation regression model is enlarged and complex systems of *simultaneous equation models* are used, including several equations and also several dependent variables. These models are implemented as applied econometrics software and build, for instance, the basis for national budget calculations, usually containing several hundred equations.

Modern Computer Aided Techniques

To predict future movements of financial markets, technical analysts use time series and apply the statistical and econometrical methods described above. We enlarge the methods by new techniques which are able to recognize certain relationships from examples by generalization with the help of new computer technologies. The methods used in this application are a composition of artificial neural networks, genetic algorithms (see ► Genetic algorithms) and fuzzy logic. Obviously we are not able to go into details, but we try to give a short characterization for our application.

Neural Networks

These are inspired by the functionality of nerve cells in the brain. Like humans, they can learn to recognize patterns by repeated exposure to many different examples. They can be used to detect salient characteristics whether they are handwritten characters, profitable loans or good trading decisions. Neural networks learn to recognize even regularities in data that are inexact or incomplete. A neural network finds this relationship by means of a learning cycle where a large amount of samples are presented repeatedly to the network. Neural networks cannot guarantee an optimal solution to a problem. However, properly configured and trained neural networks can often make consistently good classifications, generalizations or decisions in a statistical sense. Neural networks are widely used to identify patterns in the data of financial markets.

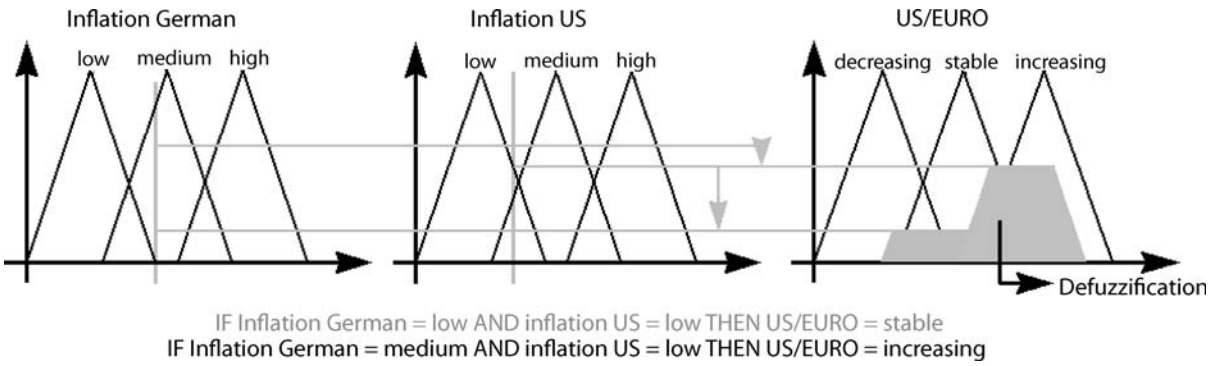
Fuzzy Logic

This is a strategy that is not based on a mathematical description of a special system or market but is intended to model the behavior of a human investor. The expert's knowledge is specified in terms of linguistic rules in which linguistic expressions are associated with fuzzy sets. Fuzzy set methods tend to overcome the vagueness of causality. They can be used to explain financial markets' developments using fundamental rules as shown in Fig. 1.

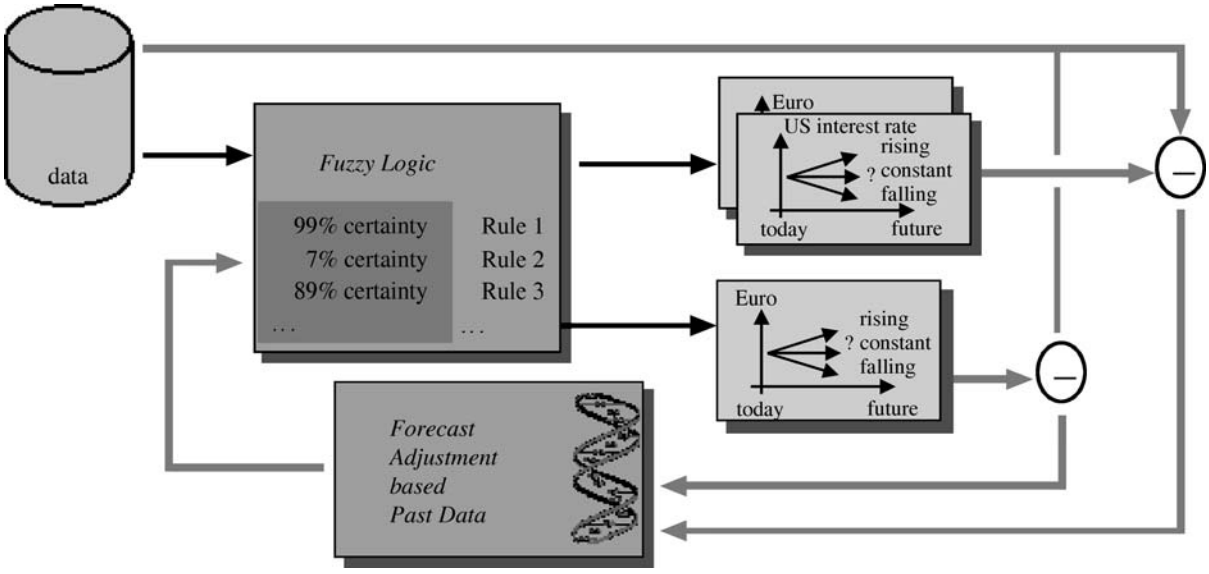
Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth – truth values between 'completely true' and 'completely false'. [25,26]. In other words, a fuzzy system is a collection of 'membership functions' and rules that are used to reason about data. In our example the ranges of the interest rates of Germany and the USA gives forecasts for the exchange rate between these currencies. Fuzzy logic enables us to model and predict market developments on the basis of the experience of financial experts.

Genetic Algorithms

A genetic algorithm allows us to optimize any given function. Genetic algorithms are search procedures based on the mechanisms of natural selection, mutation



Forecasting, Figure 1
Fuzzy Logic rules to predict the US Dollar/Euro fixingj



Forecasting, Figure 2
Genetic algorithm used for optimizing a rulebasis to forecast the US Dollar/Euro fixing

and recombination. A population consisting of chromosomes (i. e., solutions for a function) is created randomly. In the next step each chromosome is evaluated and given a certain fitness value. The fitness value represents the feasibility and the optimality of a given solution. Depending on their fitness value a certain percentage of the population is selected and deleted. The surviving individuals are recombined and mutated. After the population has been evaluated and the forecast adjustment based on past data has been decided, the selection process starts again.

The genetic algorithm is used to optimize the certainty of each rule in the *fuzzy logic* rulebase. As shown in Fig. 2 the fitness function of the genetic algorithm consists of a fuzzy logic rulebase and several mathematical objects to calculate the forecast error. The forecast error is used to evaluate the individuals. All rules are applied to historic financial data and their forecast error is summed up over the whole horizon.

Rules which are only partly true get lower certainty values until their certainty corresponds to their actual influence. Currency markets tend to follow certain reg-

ularities, detected by expert knowledge or, for example, the purchasing power theorem.

A similar procedure is applied to neural networks in order to optimize the topology of the neural network itself as well as a data mining approach to identify the input parameters. The genetic algorithm allows us to cancel out useless time series. When forecasting financial markets an appropriate and adaptive input parameter selection is necessary.

In our case the inputs are knowledge of economic data to receive forecasts for future developments of financial markets.

One goal in system theory is, in order to integrate the ideas of several disciplines, to have a successful instrument for analyzing forecasting processes including learning and discovery in direction optimality. This process of searching for the best value that can be realized or attained is based on the events of subjects whose actions are not able to be forecasted with total certainty.

Finally, we are able to summarize this as follows, using the different stages we took into account:

- historical comparison based on repeatedly observed similar events and on statistical data, e. g. business fluctuations, Harvard's barometers, chart extrapolations;
- time series analysis, based on proceedings on mathematical statistics;
- econometric forecasting models, including stepwise regression models, as well as vector autoregressive models (VAR);
- modern techniques, mainly computer aided, and software which is available as adaptive models, learning models, artificial neural networks (ANN), fuzzy set models and evolutionary algorithms.

See also

► Continuous Global Optimization: Applications

References

1. Armstrong JS (1985) Long range forecasting: From crystal ball to computer, 2nd edn. Wiley, New York
2. Armstrong JS (1996) Forecasting. In: Gass S, Harris C (eds) Encycl. Oper. Res. and Management Sci. Kluwer, Dordrecht, pp 229–233
3. Armstrong JS, Collopy F (1992) Error measures for generalizing about forecasting methods: Empirical comparison. Internat J Forecasting 8:69–80
4. Becker O, Leopold-Wildburger U (1996) The bounds & likelihood-procedure - A simulation study concerning the efficiency of visual forecasting techniques. Central Europ J Oper Res and Economics 4(2/3):223–229
5. Becker O, Leopold-Wildburger U (2000) Erwartungsbildung und Prognose. Austrian J Statist:29(1):1–76
6. Bowerman B, O'Connell R (1993) Forecasting and time series: An applied approach, 3rd edn. Duxbury, Belmont, CA
7. Box GE, Jenkins GM (1970) Time series analysis for forecasting and control. Holden Day, San Francisco
8. Brown RG (1959) Statistical forecasting for inventory control. McGraw-Hill, New York
9. Brown RG (1962) Smoothing, forecasting and prediction of discrete time series. Prentice-Hall, Englewood Cliffs, NJ
10. Bunn DW (1996) Non-traditional methods of forecasting. Europ J Oper Res 92:528–536
11. Dalkey N, Helmer O (1963) An experimental application of the Delphi method to the use of experts. Managem Sci 9:458–467
12. Davis L (ed) (1991) Handbook of genetic algorithms. v. Nostrand, Princeton, NJ
13. Fildes R, Hibon M, Makridakis S, Meade N Generalizing about univariate forecasting models, WP Imperial College,
14. Friedman M (1963) Individual behaviour in oligopolistic markets: An experimental study. Yale Economic Essays 3:359–417
15. Gerstenfeld A (1944) Technological forecasting. J Business 44:10–18
16. Granger GWC, Newbold P (1977) Forecasting economic time series. Acad. Press, New York
17. Holland J (1975) Adaptation in natural and artificial systems. Univ. Michigan Press, Ann Arbor, MI
18. Holland J, Holyoak K, Nisbett R, Thagard P (1986) Induction, processes of inference, learning and discovery. MIT, Cambridge, MA
19. Kohonen T (1997) Self organizing maps, 2nd edn. Springer, Berlin
20. Leopold-Wildburger U (1997) Some selected topics in experimental economics. Internat Trans Oper Res 4(3):165–174
21. Makridakis S, Wheelwright SC (1987) The handbook of forecasting: A manager's guide, 2nd edn. Wiley, New York
22. Makridakis S, Wheelwright SC (1989) Forecasting methods for management, 3rd edn. Wiley, New York
23. Makridakis S, Winkler RL (1983) Averages of forecasts: Some empirical results. Managem Sci 29:987–996
24. Montgomery DC, Johnson LA, Gardiner JS (1990) Forecasting and time series analysis, 2nd edn. McGraw-Hill, New York
25. Yager RR, Zadeh LA (1995) An introduction to fuzzy logic applications in intelligent systems, 3rd edn. Kluwer, Dordrecht
26. Zimmermann H-J (1996) Fuzzy set theory, 3rd edn. Kluwer, Dordrecht

Fourier–Motzkin Elimination Method

LEONID KHACHIYAN
Department Computer Sci., Rutgers University, Piscataway, USA

MSC2000: 52B12, 68Q25

Article Outline

Keywords
Solution of Systems of Linear Inequalities and Linear Programming Problems
Complexity of the Fourier–Motzkin Method
See also
References

Keywords

Fourier–Motzkin elimination; Linear inequalities; Linear programming; First order theory of real addition with order; Semilinear set; Projection; Output-polynomial

Consider a system of m linear inequalities in n real variables

$$Ax \le b, \tag{1}$$

where $x = (x_1, \dots, x_n)^\intercal \in \mathbf{R}^n$ is the vector of unknowns and A, b are a given real matrix and vector. Let $X = \{x \in \mathbf{R}^n : Ax \le b\}$ be the solution set of the system, and let $X^{[k]}$ denote the projection of X onto the linear space spanned by the last $n-k$ coordinates:

$$X^{[k]} = \{(x_{k+1}, \dots, x_n) \in \mathbb{R}^{n-k}: \exists (x_1, \dots, x_k) \in \mathbb{R}^k \text{ s.t. } (x_1, \dots, x_n) \in X\}.$$

The *Fourier–Motzkin method* [3,4,5,8,10,12,14,15] successively eliminates variables $x_1, \dots, x_{n-1}$ from (1) and computes matrices $A^{[k]}$ and vectors $b^{[k]}$ such that

$$X^{[k]} = \left\{x^{[k]} \in \mathbb{R}^{n-k}: \ A^{[k]}x^{[k]} \le b^{[k]}\right\}, \qquad k = 1, \dots, n-1,$$

where $x^{[k]} = (x_{k+1}, \dots, x_n)^\intercal$.

In order to eliminate variable x_1, we first multiply each of the m inequalities of (1) by an appropriate positive scalar to make each entry in the first column of A equal to ± 1 or 0. We can thus assume without loss of generality that the original system of inequalities has the form

$$\begin{aligned} +1 \cdot x_1 + \alpha_i(x^{[1]}) &\le 0, \quad i \in M_+, \\ -1 \cdot x_1 + \alpha_i(x^{[1]}) &\le 0, \quad i \in M_-, \\ 0 \cdot x_1 + \alpha_i(x^{[1]}) &\le 0, \quad i \in M_0, \end{aligned}$$

where $\alpha_i(x^{[1]}) = \alpha_{i2}x_2 + \cdots + \alpha_{in}x_n + \beta_i$ are given affine forms of $x^{[1]} = (x_2, \dots, x_n)^\intercal \in \mathbf{R}^{n-1}$ and M_+, M_-, M_0 are disjoint sets of (indices of) inequalities partitioning the entire set of inequalities in (1):

$$M_+ \cup M_- \cup M_0 = \{1, \dots, m\}.$$

It is easy to see that for each fixed $x^{[1]}$, the inequalities with indices $i \in M_+ \cup M_-$ can be satisfied by some real x_1 if and only if each upper bound $-\alpha_i(x^{[1]})$, $i \in M_+$ on x_1 exceeds each lower bound $\alpha_j(x^{[1]})$, $j \in M_-$ on the same variable, i. e., $-\alpha_i(x^{[1]}) \ge \alpha_j(x^{[1]})$ for all $i \in M_+$ and $j \in M_-$. Combining these $|M_+|\ |M_-|$ inequalities with the remaining $|M_0|$ inequalities of (1) that do not depend on x_1, we arrive at the system of $|M_+|\ |M_-| + |M_0|$ linear inequalities

$$\begin{aligned} \alpha_i(x^{[1]}) + \alpha_j(x^{[1]}) &\le 0, \quad (i,j) \in M_+ \times M_-, \\ \alpha_i(x^{[1]}) &\le 0, \quad i \in M_0, \end{aligned}$$

whose solutions set is $X^{[1]}$. The above system can be written as $A^{[1]}x^{[1]} \le b^{[1]}$ with appropriate matrix $A^{[1]}$ and vector $b^{[1]}$. This gives $X^{[1]} = \{x^{[1]} \in \mathbf{R}^{n-1} : A^{[1]}x^{[1]} \le b^{[1]}\}$. Eliminating variable x_2 from $A^{[1]}x^{[1]} \le b^{[1]}$ we obtain a similar description $X^{[2]} = \{x^{[2]} \in \mathbf{R}^{n-2} : A^{[2]}x^{[2]} \le b^{[2]}\}$ for the second projection and so on. After $n-1$ steps of the above procedure we have $n-1$ matrices $A^{[k]}$ and vectors $b^{[k]}$ such that $X^{[k]} = \{x^{[k]} \in \mathbf{R}^{n-k} : A^{[k]}x^{[k]} \le b^{[k]}\}$, $k = 1, \dots, n-1$.

Solution of Systems of Linear Inequalities and Linear Programming Problems

If the solution set $X = \{x \in \mathbf{R}^n: Ax \le b\}$ is nonempty, then so are all the projections $X^{[k]} \subseteq \mathbf{R}^{n-k}$, $k = 1, \dots, n-1$, and vice versa. In particular, if $Ax \le b$ is feasible, then

$$X^{[n-1]} = \left\{x^{[n-1]} \in \mathbb{R}: \ A^{[n-1]}x^{[n-1]} \le b^{[n-1]}\right\}$$

is a nonempty interval on the scalar variable $x^{[n-1]} = x_n$. Given $A^{[n-1]}$ and $b^{[n-1]}$, we can easily find a point $\overline{x}_n \in X^{[n-1]}$. Then, substituting $x_n = \overline{x}_n$ into $A^{[n-2]}x^{[n-2]} \le b^{[n-2]}$, we obtain a new feasible system of linear inequalities whose solution set is the interval $\{x_{n-2} \in \mathbb{R}\colon\ (x_{n-1}, \overline{x}_n) \in X^{[n-2]}\}$. Solving this one-variable system yields a point $\overline{x}^{[n-2]} = (\overline{x}_{n-1}, \overline{x}_n) \in X^{[n-2]}$, which can be substituted in $A^{[n-3]}x^{[x-3]} \le b^{[n-3]}$ etc. By repeating such backward substitutions, the Fourier–Motzkin method can compute a solution $(\overline{x}_1, \ldots, \overline{x}_n)$ to any feasible system of linear inequalities $Ax \le b$. 'Historically, it is the 'pre-linear programming' method to solve linear inequalities' [14].

Now suppose that the input system is infeasible, i. e. $X = \{x \in \mathbf{R}^n\colon Ax \le b = \emptyset\}$. As was pointed out in [10], the Fourier–Motzkin method can then find nonnegative real multipliers $p_1, \ldots, p_m$ such that

$$pA = 0, \quad pb = -1, \quad p = (p_1, \ldots, p_m) \ge 0. \quad (2)$$

To see this, observe that each inequality in $A^{[1]}x^{[1]} \le b^{[1]}$ is a positive combination of at most two inequalities of the original system. Since a nonnegative combination of nonnegative combinations of some inequalities is a nonnegative combination of the same inequalities, we conclude that each inequality in each system $A^{[k]}x^{[k]} \le b^{[k]}$, $k = 1, \ldots, n-1$, is a nonnegative combination of the input inequalities. Considering that $A^{[n-1]}x^{[n-1]} \le b^{[n-1]}$ is an infeasible system of linear inequalities in one variable, $A^{[n-1]}x^{[n-1]} \le b^{[n-1]}$ is easily seen to contain one or two inequalities whose positive combination yields the infeasible inequality $0 \cdot x_n \le -1$. This is equivalent to (2). In particular, the Fourier–Motzkin method provides a simple algorithmic proof of the Farkas lemma (cf. ▶ Farkas lemma; ▶ Farkas lemma: Generalizations): (1) is feasible if and only if (2) is infeasible.

The Fourier–Motzkin method can also be used to solve the general linear programming problem

$$\xi^* = \max\{c^\top x\colon\ Ax \le b,\ x \in \mathbb{R}^n\}. \quad (3)$$

For instance, we can eliminate n variables $x = (x_1, \ldots, x_n)$ from $Ax \le b, x_{n+1} - c^\top x \le 0$ to determine the interval $X^{[n]} = \{x_{n+1} : x_{n+1} \le \xi^*\}$. Then, letting $x_{n+1} = \xi^*$ and solving the resulting system yields an optimal solution.

It should be mentioned that there are far more efficient linear programming algorithms. Note, however, that (3) calls for projecting $X = \{x \in \mathbf{R}^n\colon Ax \le b\}$ on a one-dimensional subspace. After an appropriate linear transformation, the Fourier–Motzkin method can project $X = \{x \in \mathbf{R}^n\colon Ax \le b\}$ on any given subspace in $\mathbf{R}^n$.

Complexity of the Fourier–Motzkin Method

Let m_k denote the number of inequalities in the kth system $A^{[k]}x^{[k]} \le b^{[k]}$ generated by the Fourier–Motzkin method. Since $m_1 = |M_+|\,|M_-| + |M_0| \le m^2$, we have $m_k \le m_{k-1}^2$ for all k. So the number of inequalities is at most squared at each step of the method, which implies that m_k is bounded by a doubly exponential function in k, say $m_k \le m^{2^k}$. The following example shows that with sufficiently many variables, the kth step of the method can produce

$$m_k = m^{2^{k(1-o(1))}}$$

inequalities.

Example 1 [14] Let $n = 2^k + k + 2$ and consider a system of linear inequalities $Ax \le b$ which contains as left-hand sides $m = 8\binom{n}{3}$ linear forms $\pm x_{i_1} \pm x_{i_2} \pm x_{i_3}$ for all $1 \le i_1 < i_2 < i_3 \le n$. By induction on $j = 1, \ldots, k$ it is easy to show that after eliminating the first j variables, the resulting system includes among its left-hand sides all the forms $\pm x_{i_1} \pm \cdots \pm x_{i_s}$ with $k+1 \le i_1 < \cdots < i_s \le n$ and $s = 2^j + 2$. In particular, for $j = k$ we have at least $2^{2^k+2} = m^{2^{k(1-o(1))}}$ inequalities in $A^{[k]}x^{[k]} \le b^{[k]}$.

Let us now return to the first step of the algorithm where we replace $Ax \le b$ by the $|M_+|\,|M_-| + |M_0|$ new inequalities $A^{[1]}x^{[1]} \le b^{[1]}$. As was pointed out already by J.B.J. Fourier, 'it nearly always happens that a rather large number of these new inequalities are redundant' and 'their removal greatly simplifies the problem' [8]. If the redundant inequalities are systematically removed at each step of the algorithm, the number m_k of inequalities generated by kth step of the Fourier–Motzkin method is bounded by an exponential function in k. Assume without loss of generality that $X = \{x \in \mathbf{R}^n\colon Ax \le b\}$ is full-dimensional, then each projection $X^{[k]}$ is also full-dimensional and m_k is the number of facets of $X^{[k]}$. Therefore m_k is bounded by the total number of i-faces of X for $i \ge n - k - 1$. Hence

$$m_k \le \sum_{i=1}^{k+1} \binom{m}{i} \sim \frac{m^{k+1}}{(k+1)!} \quad \text{for } m \to \infty.$$

(This rough estimate can be improved by using the upper bound theorem [11]; in particular, m_k cannot grow faster than $m^{\lfloor n/2 \rfloor}$.) In the example below, $X^{[k]}$ has

$$m_k \geq \frac{m^{k+1}}{(k+1)^{k+1}}$$

facets.

Example 2 Let $s \geq 2$ be a natural number. Consider the system of $m = (k+1)\,s$ linear inequalities

$$y_{ij} \geq x_i, \quad i = 1, \ldots, k, \quad j = 1, \ldots, s,$$
$$x_1 + \cdots + x_k \geq z_l, \quad l = 1, \ldots, s,$$

where x_i, y_{ij}, and z_l are real variables. The elimination of $x_1, \ldots, x_k$ results in $s^{k+1} = (m/(k+1))^{k+1}$ inequalities

$$y_{1f(1)} + \cdots + y_{kf(k)} \geq z_l, \quad l = 1, \ldots, s,$$

where f ranges over the set of all s^k mappings from $\{1, \ldots, k\}$ to $\{1, \ldots, s\}$. None of the inequalities above is redundant. For instance,

$$y_{11} + \cdots + y_{k1} \geq z_1$$

is violated by $y_{11} = \cdots = y_{k1} = 0$ and $z_1 = \cdots = z_s = 1$, whereas all the other inequalities can be satisfied by giving the remaining variables y_{ij} a high value.

Since detecting the redundancy of an inequality can be done via linear programming (or by maintaining a list of vertices and extreme directions of $X^{[k]}$ with the *double description method* [4,13], see also [9,15] and references herein), the Fourier–Motzkin method runs in exponential space and time. It is natural to ask whether given $X = \{x \in \mathbf{R}^n : Ax \leq b\}$ and a number $k \in \{1, \ldots, n-1\}$, an irredundant description for $X^{[k]} = \{x^{[k]} \in \mathbf{R}^{n-k} : A^{[k]}x^{[k]} \leq b^{[k]}\}$ can be computed in *output-polynomial time*, i. e. by an algorithm that runs in time polynomial in the total input and output size. This question is open even in the bit model of computation for rational A and b, when redundant inequalities can be detected in polynomial time. A related problem is the generation of all vertices for $X = \{x \in \mathbf{R}^n : Ax \leq b\}$. The vertex generation problem (or its dual, the *convex hull problem*) can also be solved by the double description method, see e. g. [1], but the question as to whether there is an output-polynomial vertex generation algorithm remains open.

Finally, we mention that the Fourier–Motzkin method can be modified to a quantifier-elimination method for arbitrary *semilinear sets*

$$\mathcal{X}^{[k]} = \{(x_{k+1}, \ldots, x_n) \in \mathbb{R}^{n-k} : (Q_1 x_1 \in \mathbb{R}) \cdots (Q_k x_k \in \mathbb{R}) \; \mathcal{F}(x_1, \ldots, x_n) \text{ true}\}, \quad (4)$$

where $Q_1, \ldots, Q_k \in \{\exists, \forall\}$ are existential and/or universal quantifiers and $\mathcal{F}(x_1, \ldots, x_n)$ is a given Boolean function of m threshold predicates

$$\mathcal{F}_i(x) = \begin{cases} \text{true} & \text{if } a_i^\top x \leq b_i, \\ \text{false} & \text{otherwise,} \end{cases}$$

with given coefficients $a_i \in \mathbf{R}^n$ and $b_i \in \mathbf{R}$, $i = 1, \ldots, m$. In particular, if $Q_1, \ldots, Q_k$ are all existential quantifies and $\mathcal{F} = \mathcal{F}_1 \wedge \cdots \wedge \mathcal{F}_m$, we obtain the previously considered problem of projecting the polyhedral set $X = \{x \in \mathbf{R}^n : a_i^\top x \leq b_i, i = 1, \ldots, m\}$ onto the space spanned by the last $n - k$ coordinates. In general, (4) can be transformed into an equivalent quantifier-free representation $\mathcal{X}^{[k]} = \{(x_{k+1}, \ldots, x_n) : \mathcal{G}(x_{k+1}, \ldots, x_n) \text{ true}\}$, where $\mathcal{G}$ is some Boolean formula whose atoms are new threshold predicates of $(x_{k+1}, \ldots, x_n) \in \mathbf{R}^{n-k}$. This can be done, for instance, as follows [6,7]. To eliminate the rightmost quantifier $Q_k x_k \in \mathbf{R}$, write each threshold inequality involving x_k in the form $x_k \leq \alpha_i(x^{(k)})$ or $x_k \geq \alpha_i(x^{(k)})$, where the α_i' are given affine forms of the remaining variables $x^{(k)} = (x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n)$. Replace the infinite range $x_k \in \mathbf{R}$ by the finite set S of sample points $x_k = (\alpha_i(x^{(k)}) + \alpha_j(x^{(k)}))/2$ and $x_k = \pm\infty$. Now it is easy to see that the expression $(\exists x_k \in \mathbf{R})\mathcal{F}(x_1, \ldots, x_n)$ is equivalent to the quantifier-free disjunction $\vee_{x_k \in S} \mathcal{F}(x_1, \ldots, x_n)$ and that $(\forall x_k \in \mathbf{R})\mathcal{F}(x_1, \ldots, x_n)$ can be replaced by the equivalent conjunction $\wedge_{x_k \in S}\mathcal{F}(x_1, \ldots, x_n)$. Quantifies $Q_{k-1}x_{k-1}, \ldots, Q_1 x_1$ can be eliminated in the same way. For a discussion of faster algorithms that eliminate *blocks* of consecutive identically quantified variables see [2].

See also

► Linear Programming

References

1. Avis D, Bremner B, Seidel R (1997) How good are convex hull algorithms. Comput Geom Th Appl 7:265–302

2. Basu S (1997) An improved algorithm for quantifier elimination over real closed fields. In: Proc. 38th IEEE Symp. Foundations of Computer Sci., pp 56–65
3. Chvátal V (1983) Linear programming. Freeman, New York
4. Dantzig GB, Eaves BC (1975) Fourier–Motzkin elimination and its dual. J Combin Th A 14:288–297
5. Dines LL (1918/9) Systems of linear inequalities. Ann of Math 2(20):191–199
6. Eaves BC, Rothblum UG (1992) Dines–Fourier–Motzkin quantifier-elimination and applications of corresponding transfer principles over ordered fields. Math Program 53:307–321
7. Ferrante J, Rackoff C (1975) A decision procedure for the first order theory of real addition with order. SIAM J Comput 1:69–76
8. Fourier JBJ (1973) Analyse de travaux de l'Académie Royale de Sci., pendant l'anné 1824. In: Kohler DA (ed) Oper Res, 10, pp 38–42, Transl. of a Report by Fouries on his work on linear inequalities.
9. Fukuda K, Prodon A (1996) Double description method revisited. Lecture Notes Computer Sci 1120:91–111
10. Kuhn HW (1956) Solvability and consistency for linear equations and inequalities. Amer Math Monthly 63:217–232
11. McMullen P (1970) The maximal number of faces of a convex polytope. Mathematika 17:179–184
12. Motzkin TS (1936) Beiträge zur Theorie der linearen Ungleichungen. Doktorarbeit, Univ. Basel. Transl. in Cantor D, Gordon B, Rothschild B (eds) (1983): Contribution to the theory of linear inequalities: Selected papers. Birkhäuser, pp 81–103
13. Motzkin TS, Raifa H, Thompson GL, Thrall RM (1953) The double description method. In: Kuhn HW, Tucker AW (eds) Contributions to the Theory of Games, vol II. Ann Math Stud. 28 Princeton Univ. Press, Princeton, pp 81–103
14. Schrijver A (1986) Theory of linear and integer programming. Wiley/Interscience, New York
15. Ziegler GM (1994) Lectures on polytopes. Springer, Berlin

Fractional Combinatorial Optimization

FCO

Tomasz Radzik
King's College London, London, UK

MSC2000: 90-08, 90C27, 90C32, 68Q25, 68R05

Article Outline

Keywords
The Binary Search Method (BSM)
The Newton Method (NM)
Megiddo's parametric search (MPS)
See also
References

Keywords

Fractional optimization; Cost-to-time ratio; Newton method; Dinkelbach method; Megiddo parametric search

A *fractional combinatorial optimization problem* (FCOP) is a combinatorial optimization problem with an objective function which is a ratio of two (nontrivial) functions. Instances of a FCOP can be expressed in the general form:

$$\begin{cases} \max & \frac{f(\mathbf{x})}{g(\mathbf{x})}, \\ \text{for} & \mathbf{x} \in \mathcal{X}, \end{cases} \tag{1}$$

where $\mathcal{X} \subseteq \{0, 1\}^p$ is a set of (vectors representing) certain combinatorial structures, and f and g are real-valued functions defined on $\mathcal{X}$. Numbers $f(\mathbf{x})$, $g(\mathbf{x})$, and $f(\mathbf{x})/g(\mathbf{x})$ are usually called the *cost*, the *weight*, and the *mean-weight cost* of structure $\mathbf{x}$. A minimization FCOP is equivalent to the corresponding maximization problem, if the cost function f can be replaced with function $-f$. The FCOPs which appear in the literature on combinatorial optimization include: the *minimum ratio spanning-tree* problem [2,13,14]; the *maximum profit-to-time ratio cycle* problem and the equivalent *minimum cost-to-time ratio cycle* problem [1,3,6,11,12,13,14]; the *minimum mean cycle* problem [1,10,11]; the *maximum mean-weight cut* problem [16]; the *maximum mean cut* problem [5,9]; and the *fractional 0–1 knapsack* problem [7,8].

Consider, as an example, the minimum cost-to-time ratio cycle problem (MRCP). An instance of this problem consists of a directed graph $G = (V, E)$, where $E = \{e_1, \ldots, e_m\}$ is the set of edges, and numbers c_i and t_i associated with each edge e_i, for $i = 1, \ldots, m$. The objective is to find a simple cycle Γ in G which minimizes the ratio of $\sum\{c_i : e_i \in \Gamma\}$ to $\sum\{t_i : e_i \in \Gamma\}$. To express this instance of the MRCP in the form (1), let $\mathcal{X} \subseteq \{0, 1\}^m$ be the set of the characteristic vectors of the simple cycles in G, and for $\mathbf{x} = (x_1, \ldots, x_m) \in \{0, 1\}^m$, let $f(\mathbf{x}) = -(c_1 x_1 + \cdots + c_m x_m)$ and $g(\mathbf{x}) = t_1 x_1 + \cdots + t_m x_m$. The MRCP models the following *tramp steamer*

problem [1,12]: V is the set of ports which can be visited by our cargo ship; $E \subseteq V \times V$ is the set of possible direct port-to-port trips; numbers c_i and t_i are the cost and the transit time of trip $e_i \in E$, respectively; and the objective is to find a closed tour for the ship which minimizes the daily cost (or, equivalently, maximizes the daily profit).

A FCOP such that the denominator of the objective function $g(x_1, \ldots, x_p) = x_1 + \cdots + x_p$ is commonly called a *uniform fractional combinatorial optimization* problem. The minimum mean cycle problem (which is a special case of the MRCP with all numbers t_i equal to 1) is a uniform FCOP. A FCOP such that $f(x_1, \ldots, x_p) = a_1x_1 + \cdots + a_px_p$ and $g(x_1, \ldots, x_p) = b_1x_1 + \cdots + b_px_p$ is called a *linear fractional combinatorial optimization* problem. All FCOPs mentioned above are linear.

Generic methods for FCO usually follow the *parametric approach to fractional optimization*. Let $\delta \in \mathbf{R}$ be a parameter. Problem:

$$\begin{cases} \max & f(\mathbf{x}) - \delta \cdot g(\mathbf{x}), \\ \text{for} & \mathbf{x} \in \mathcal{X}, \end{cases} \tag{2}$$

is called the *parametric problem* corresponding to the fractional problem (1). Let $h(\delta)$ denote the optimum objective value of problem (2). From now on assume that $f(\mathbf{x}) > 0$, for some $\mathbf{x} \in \mathcal{X}$, and $g(\mathbf{x}) > 0$, for all $\mathbf{x} \in \mathcal{X}$. Function h is continuous, convex, piecewise linear and strictly decreasing on $(-\infty, +\infty)$. It has exactly one root δ^* and this root is the optimum objective value of problem (1). The main generic methods for FCO are the *binary search method*, the *Newton method*, and *Megiddo's parametric search*. They all can be viewed as methods for finding the root of function h.

The Binary Search Method (BSM)

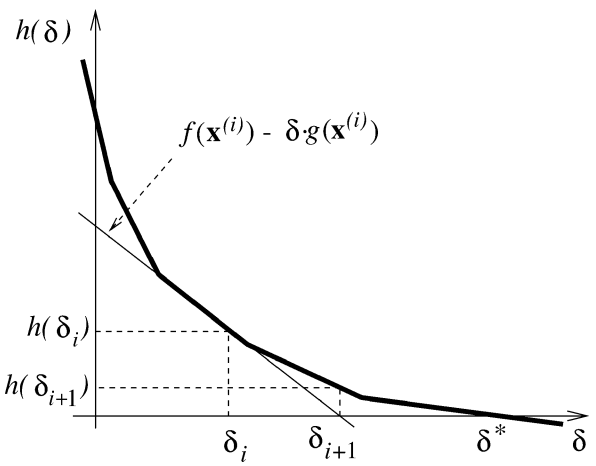
This method maintains an interval $[\alpha, \beta]$ containing the root δ^* of function h, and reduces this interval by half in each iteration by checking the sign of $h((\alpha+\beta)/2)$. Thus to apply the BSM, one needs an algorithm $\mathcal{A}_0$ which for a given $\delta \in \mathbf{R}$ calculates the sign of the optimum objective value of problem (2). For a linear FCOP such that all numbers $|a_i|$ and $|b_i|$ are integers not greater than U (an *integral linear FCOP*), the BSM finds an optimum solution in $O(\log(pU))$ iterations. This follows form the fact that if numbers $f(\mathbf{x}')/g(\mathbf{x}')$ and $f(\mathbf{x}'')/g(\mathbf{x}'')$ are not equal, they must differ by at least $1/(pU)^2$. Hence, as soon as the length of the search interval $[\alpha, \beta]$ becomes less than $1/(pU)^2$, it contains only one value $f(\mathbf{x})/g(\mathbf{x})$, which must be equal to δ^*.

The Newton Method (NM)

This generic method for *fractional optimization*, also called the *Newton–Raphson method* or the *Dinkelbach method* [4], is an application of the classical Newton method to the problem of finding the root δ^* of function h. The NM computes an increasing sequence $\delta_1, \delta_2, \ldots$ of lower bounds on δ^*. During iteration i, a solution $\mathbf{x}^{(i)}$ of problem (2) for $\delta = \delta_i$ is computed, and δ_{i+1} is set to $f(\mathbf{x}^{(i)})/g(\mathbf{x}^{(i)})$ (see Fig. 1). The NM finds an optimum solution of a FCOP in a finite number of iterations, because function h consists of a finite number ($\leq |\mathcal{X}|$) of linear segments. The NM solves a uniform FCOP in at most $p + 1$ iterations, because function h corresponding to such a problem consists of at most p linear segments (since function g yields at most p different values). Other bounds on the number of iterations of the NM for FCO can be derived from the fact that for each iteration i, except the last one,

$$\frac{h(\delta_{i+1})}{h(\delta_i)} + \frac{g(\mathbf{x}^{(i+1)})}{g(\mathbf{x}^{(i)})} \leq 1, \tag{3}$$

which implies that sequence $(h(\delta_i) \cdot g(\mathbf{x}^{(i)}))$ decreases to 0 at a geometric rate. Using this fact one can show that the NM solves an integral linear FCOP in $O(\log(pU))$ iterations, and any linear FCOP in $O(p^2 \log^2 p)$ itera-



Fractional Combinatorial Optimization, Figure 1
The Newton method for FCO

tions [16,17]. The NM gives the asymptotically fastest known algorithm for the maximum mean-weight cut problem [16,17]. Its running time is $O(nm^2 \log n)$ for a graph with n nodes and m edges.

Megiddo's parametric search (MPS)

Let $\mathcal{A}_1$ be an algorithm as $\mathcal{A}_0$ above but with the following additional property: The value of each computed arithmetic expression on each possible execution path of algorithm $\mathcal{A}_1$ is a linear function of parameter δ. Such an algorithm $\mathcal{A}_1$ is called a *linear algorithm* for a parametric problem. MPS [13,14] solves a FCOP by following the computation of algorithm $\mathcal{A}_1$ for $\delta = \delta^*$. All values calculated during this computation are linear functions of (unknown) δ^* and are stored as such. Thus each comparison amounts to calculating the sign of the value of an expression $s - t\,\delta^*$, where s and t are known numbers, and can be resolved by running algorithm $\mathcal{A}_0$ for $\delta = s/t$ ($s/t \leq \delta^* \Leftrightarrow h(s/t) \geq 0$). If the running times of both $\mathcal{A}_1$ and $\mathcal{A}_0$ are at most T, then the overall running time of MPS is $O(T^2)$. If algorithm $\mathcal{A}_0$ runs in time T_0 and algorithm $\mathcal{A}_1$ is parallel and runs in time T_1 on P processors, then MPS can be implemented in such a way that the overall (sequential) running time is $O(T_1 P + T_0 T_1 \log P)$: At the kth (parallel) step of the computation of $\mathcal{A}_1$ for $\delta = \delta^*$, the required signs of P_k ($\leq P$) expressions $s_{k,j} - t_{k,j}\,\delta^*$, $j = 1, \ldots, P_k$, can be found by at most $\log P_k + 1$ executions of algorithm $\mathcal{A}_0$. The first execution is for δ equal to the median of the numbers $s_{k,j}/t_{k,j}$, and its result gives the signs of half of the expressions. MPS gives, for example, the asymptotically fastest known algorithms for the minimum ratio spanning-tree problem and the minimum cost-to-time ratio cycle problem [14]. Their running times are $O(m \log^2 n \log\log n)$ and $O(n^3 \log n \log\log n)$, respectively, for a graph with n nodes and m edges. An extension of MPS to cases when only approximate algorithms $\mathcal{A}_0$ are practical is proposed in [7] and applied there to the fractional 0–1 knapsack problem.

For some FCOPs, there are specialized algorithms which do not follow any of the above three generic methods. The most prominent examples are the $O(mn)$ [10] and $O(m\sqrt{n}\log(nU))$ [15] algorithms for the maximum mean cycle problem (the latter one is for the integral case). A detailed treatment of methods for FCO can be found in [17].

See also

- Bilevel Fractional Programming
- Combinatorial Matrix Analysis
- Combinatorial Optimization Algorithms in Resource Allocation Problems
- Combinatorial Optimization Games
- Complexity Classes in Optimization
- Complexity of Degeneracy
- Complexity of Gradients, Jacobians, and Hessians
- Complexity Theory
- Complexity Theory: Quadratic Programming
- Computational Complexity Theory
- Evolutionary Algorithms in Combinatorial Optimization
- Fractional Programming
- Information-based Complexity and Information-based Optimization
- Kolmogorov Complexity
- Mixed Integer Nonlinear Programming
- Multi-objective Combinatorial Optimization
- Neural Networks for Combinatorial Optimization
- NP-complete Problems and Proof Methodology
- Parallel Computing: Complexity Classes
- Quadratic Fractional Programming: Dinkelbach Method
- Replicator Dynamics in Combinatorial Optimization

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs, NJ
2. Chandrasekaran R (1977) Minimum ratio spanning trees. Networks 7:335–342
3. Dantzig GB, Blattner WO, Rao MR (1967) Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem. In: Rosentiehl P (ed) Theory of Graphs. Gordon and Breach, New York, pp 77–83
4. Dinkelbach W (1967) On nonlinear fractional programming. Managem Sci 13:492–498
5. Ervolina TR, McCormick ST (1993) Two strongly polynomial cut cancelling algorithms for minimum cost network flow. Discrete Appl Math 46:133–165
6. Fox B (1969) Finding minimal cost-time ratio circuits. Oper Res 17:546–551
7. Hashizume S, Fukushima M, Katoh N, Ibaraki T (1987) Approximation algorithms for combinatorial fractional programming problems. Math Program 37:255–267

8. Ishii H, Ibaraki T, Mine H (1976) Fractional knapsack problems. Math Program 13:255–271
9. Iwano K, Misono S, Tezuka S, Fujishige S (1994) A new scaling algorithm for the maximum mean cut problem. Algorithmica 11:243–255
10. Karp RM (1978) A characterization of the minimum cycle mean in a digraph. Discret Math 23:309–311
11. Lawler EL (1967) Optimal cycles in doubly weighted directed linear graphs. In: Rosentiehl P (ed) Theory of Graphs. Gordon and Breach, New York, pp 209–213
12. Lawler EL (1976) Combinatorial optimization: networks and matroids. Holt, Rinehart and Winston, New York
13. Megiddo N (1979) Combinatorial optimization with rational objective functions. Math Oper Res 4:414–424
14. Megiddo N (1983) Applying parallel computation algorithms in the design of serial algorithms. J ACM 30:852–865
15. Orlin JB, Ahuja RK (1992) New scaling algorithms for assignment and minimum cycle mean problems. Math Program 54:41–56
16. Radzik T (1993) Parametric flows, weighted means of cuts, and fractional combinatorial optimization. In: Pardalos PM (ed) Complexity in Numerical Optimization. World Sci., Singapore, pp 351–386
17. Radzik T (1998) Fractional combinatorial optimization. In: Du D-Z, Pardalos PM (eds) Handbook Combinatorial Optim., vol 1. Kluwer, Dordrecht, pp 429–478

Fractional Programming

HANS FRENK[1], SIEGFRIED SCHAIBLE[2]
[1] Econometric Institute, Erasmus University, Rotterdam, The Netherlands
[2] A.G. Anderson Graduate School of Management, University California, Riverside, USA

MSC2000: 90C32

Article Outline

Keywords
Classification
Single-ratio Fractional Programs
  Applications
  Economic Applications
  Maximization of Productivity
  Maximization of Return on Investment
  Maximization of Return/Risk
  Minimization of Cost/Time
  Maximization of Output/Input
  Noneconomic applications
  Indirect Applications
  Theoretical and Algorithmic Results
  Solving Problem (P) Directly
  Solving an Equivalent Problem (P_{eq})
  Solving a Dual Problem (D)
  Solving a Parametric Problem (P_q)
  Interior Point Algorithms
  Integer Fractional Programming
Maximization of the Smallest of Several Ratios
  Applications
  Theoretical and Algorithmic Results
Maximizing a Sum of Ratios
  Applications
  Theoretical and Algorithmic Results
Multi-objective Fractional Programming
Conclusion
See also
References

Keywords

Fractional programming; Single-ratio programs; Multiratio programs

In various applications of nonlinear programming a ratio of two functions is to be maximized or minimized. In other applications the objective function involves more than one ratio of functions. Ratio optimization problems are commonly called *fractional programs*.

One of the earliest fractional programs (though not called so) is an equilibrium model for an expanding economy introduced by J. von Neumann [50] in 1937. The model determines the growth rate as the maximum of the smallest of several output-input ratios. At a time when linear programming hardly existed, the author already proposed a duality theory for this nonconcave program.

However, apart from a few isolated papers like von Neumann's, a systematic study of fractional programming began much later. In 1962 A. Charnes and W.W. Cooper published their classical paper [19] in which they show that a linear fractional program can be reduced to a linear program with help of a nonlinear variable transformation.

The study of fractional programs with only one ratio has largely dominated the literature in this field until about 1980. Many of the results then known are presented in the first monograph on fractional programming by S. Schaible [57] (1978). Since then two other monographs solely devoted to fractional pro-

gramming appeared, authored by B.D. Craven [23] and I.M. Stancu-Minasian [68]. Each of these includes a chapter on multi-ratio fractional programs.

Fractional programs have often been studied in the broader context of generalized convex programming [4]. Ratios of convex and concave functions as well as composites of such ratios are not convex in general, even in the case of linear ratios. But often they are generalized convex in some sense. From the beginning, fractional programming has benefited from advances in generalized convexity, and vice versa; see B. Martos [45]. This is demonstrated by the fact that the proceedings of each of the five international symposia on generalized convexity contain contributions on fractional programming; see [16,27,41,63,66]. Fractional programming overlaps also with global optimization. Several types of ratio optimization problems have local, nonglobal optima. For an extensive survey of fractional programming, see [60].

The survey [60] also contains the largest bibliography on fractional programming so far (1999). It has almost twelve-hundred entries. For a separate, rich bibliography see [68].

Clearly, fractional programming is a dynamic, growing area of research. It has been encouraging to observe that over the years research on theory and solution methods has increasingly more focused on those ratios which are of particular interest in applications. Since these are spread over a wide range of fields, surveys on fractional programming applications have been much needed. In the single-ratio case, a first detailed survey appeared in [57] and became a basis for [58,62] and for surveys by others. A more recent, detailed survey of single-ratio fractional programming applications is found in [68]. For the multi-ratio case, the surveys in [60,61,62] may be consulted. As various classes of fractional programs are presented below, the relevance of each class will be indicated.

Classification

Let f, g, h_k $(k = 1, \ldots, m)$ denote real-valued functions which are defined on a set C in the n-dimensional Euclidean space $\mathbf{R}^n$. Consider

$$q(x) = \frac{f(x)}{g(x)} \tag{1}$$

over the set

$$S = \{x \in C\colon\ h_k(x) \le 0,\ k = 1, \ldots, m\}\,, \tag{2}$$

assuming that $g(x) > 0$ on C. The nonlinear program

$$\text{(P)} \quad \sup\{q(x)\colon\ x \in S\} \tag{3}$$

is called a (single-ratio) *fractional program*. In some applications more than one ratio appears in the objective function. Examples discussed in this article are

$$\sup\left\{\min_{1\le i\le p} q_i(x)\colon\ x \in S\right\} \tag{4}$$

and

$$\sup\left\{\sum_{i=1}^{p} q_i(x)\colon\ x \in S\right\}\,, \tag{5}$$

where $q_i(x)$ equals the ratio of the numerator $f_i(x)$ and the denominator $g_i(x)$ satisfying $g_i(x) > 0$ on C. Problem (4) is sometimes referred to as a *generalized fractional program* [62] while (5) is called a *sum-of-ratios fractional program*. Both problems (4) and (5) are related to the *multi-objective fractional program*

$$\max\{(q_1(x), \ldots, q_p(x))\colon\ x \in S\}\,. \tag{6}$$

So far, the functions in the numerator and denominator were not specified. If f, g and h_k are affine functions (linear plus a constant) and C is the nonnegative orthant of $\mathbf{R}^n$, then (P) is called a *linear fractional program*. It is of the following form:

$$\sup\left\{\frac{c^\top x + \alpha}{d^\top x + \beta}\colon\ Ax \le b, x \ge 0\right\}\,, \tag{7}$$

where $c, d \in \mathbf{R}^n$, $\alpha, \beta \in \mathbf{R}$, the superscript $\top$ denotes the transpose, A is an $m \times n$ matrix and $b \in \mathbf{R}^m$.

In generalization of a linear fractional program, we call (P) a *quadratic fractional program* if C is the nonnegative orthant, f, g are quadratic and the h_k are affine.

Problem (P) is said to be a *concave fractional program* if the numerator f is concave on C and g, h_k are convex on C, where C is a convex set. In addition, it is assumed that f is nonnegative on S if g is not affine. Note that the objective function of a concave fractional program (3) is generally not a concave function. Instead, it is composed of a concave and a convex function. Even under these restrictive concavity/convexity

assumptions fractional programs are generally nonconcave programs.

The focus in fractional programming is the objective function and its ratio-structure. The feasible region is generally assumed to be convex or a convex polyhedron.

Single-ratio Fractional Programs

Consider the problem

$$(\mathrm{P}) \quad \sup\{q(x): \ x \in S\}, \tag{8}$$

where $q(x)$ equals the ratio of the numerator $f(x)$ and the denominator $g(x)$ with $g(x) > 0$ on C.

Applications

Fractional programs arise in management decision making as well as outside of it. They also occur sometimes indirectly in modeling where initially no ratio is involved. The purpose of the following overview is to demonstrate the diversity of problems which can be cast in the form of a single-ratio fractional program. A more comprehensive coverage which also includes the references for the models below is given in [60]. For other surveys of applications of (8) see [23,57,58,62,68].

Economic Applications

The efficiency of a system is sometimes characterized by a ratio of technical and/or economical terms. Maximizing the efficiency then leads to a fractional program. Some applications are given below.

Maximization of Productivity

P.C. Gilmore and R.E. Gomory [35] discuss a stock cutting problem in the paper industry for which under the given circumstances it is more appropriate to minimize the ratio of wasted and used amount of raw material rather than just minimizing the amount of wasted material. This stock cutting problem is formulated as a linear fractional program. In a case study, J.A. Hoskins and R. Blom [38] use fractional programming to optimize the allocation of warehouse personnel. The objective is to minimize the ratio of labor cost to the volume entering and leaving the warehouse.

Maximization of Return on Investment

In some resource allocation problems the ratio profit/capital or profit/revenue is to be maximized. A related objective is return per cost maximization. Resource allocation problems with this objective are discussed in more detail in [47]. In these models the term 'cost' may either be related to actual expenditure or may stand, for example, for the amount of pollution or the probability of disaster in nuclear energy production. Depending on the nature of the functions describing return, profit, cost or capital, different types of fractional programs are encountered. For example, if the price per unit depends linearly on the output and cost and capital are affine functions, then maximization of the return on investment gives rise to a concave quadratic fractional program (assuming linear constraints). In location analysis maximizing the profitability index (rate of return) is in certain situations preferred to maximizing the net present value, according to [5] and [8] and the cited references.

Maximization of Return/Risk

Some portfolio selection problems give rise to a concave nonquadratic fractional program of the form (11) below which expresses the maximization of the ratio of expected return and risk. For related concave and nonconcave fractional programs arising in financial planning see [60]. Markov decision processes may also lead to the maximization of the ratio of mean and standard deviation.

Minimization of Cost/Time

In several routing problems a cycle in a network is to be determined which minimizes the cost-to-time ratio or maximizes the profit-to-time ratio. Also the average cost objective used within the theory of stochastic regenerative processes [3] leads to the minimization of cost per unit time. A particular example occurring within this framework is the determination of the optimal ordering policy of classical periodic and continuous review single item inventory models, e. g., [31]. Another example of this framework are maintenance and replacement models. Here the ratio of the expected cost for inspection, maintenance and replacement and the expected time between two inspections is to be minimized, e. g., [6,30].

Maximization of Output/Input

Charnes, Cooper and E. Rhodes [22] use a linear fractional program as a model to evaluate the efficiency of decision making units (data envelopment analysis (DEA)). Given a collection of decision making units, the efficiency of each unit is obtained from the maximization of a ratio of weighted outputs and inputs subject to the condition that similar ratios for every decision making unit are less than or equal to unity. The variable weights are then the efficiency of each member relative to that of the others. For an extensive treatment of DEA see [21].

In the management literature there has been an increasing interest in optimizing relative terms such as relative profit. No longer are these terms merely used to monitor past economic behavior. Instead the optimization of rates is getting more attention in decision making processes for future projects; e. g., [5,37].

Noneconomic applications

In information theory the capacity of a communication channel can be defined as the maximal transmission rate over all probabilities. This is a concave nonquadratic fractional program. The eigenvalue problem in numerical mathematics can be reduced to the maximization of the Rayleigh quotient, and hence gives rise to a quadratic fractional program which is generally not concave. An example of a fractional program in physics is given by J.E. Falk [29]. He maximizes the signal-to-noise ratio of a spectral filter which is a concave quadratic fractional program.

Indirect Applications

There are a number of management science problems that indirectly give rise to a concave fractional program. A concave quadratic fractional program arises in location theory as the dual of a Euclidean multifacility minimax problem. In large scale mathematical programming, decomposition methods reduce the given linear program to a sequence of smaller problems. In some of these methods the subproblems are linear fractional programs. The ratio originates in the minimum-ratio rule of the simplex method.

Fractional programs are also met indirectly in stochastic programming, as first shown in [20] and [13]. This will be illustrated by two models below [57,68]. First, consider the stochastic mathematical program:

$$\max\left\{a^\top x\colon\ x\in S\right\}, \tag{9}$$

where the coefficient vector a is a random vector with a multivariate normal distribution and S is a (deterministic) convex feasible region. It is assumed that the decision maker replaces (9) by a decision problem

$$\max\left\{\mathsf{P}\{a^\top x\geq k\}\colon\ x\in S\right\}, \tag{10}$$

i. e., he wants to maximize the probability that the random variable $a^\top x$ attains at least a prescribed level k. Then (9) reduces to

$$\max\left\{\frac{e^\top x-k}{\sqrt{x^\top Vx}}\colon\ x\in S\right\}, \tag{11}$$

where e is the mean vector of the random vector a and V its variance-covariance matrix. Hence the maximum probability model of the concave program (9) gives rise to a concave fractional program. If in (9) the linear objective function is replaced by other types of nonlinear functions, then the maxi- mum probability model leads to various other concave fractional programs as demonstrated in [57,68].

Consider a second stochastic program

$$\max\left\{f_0(x)+\theta f_1(x)\colon\ x\in S\right\}, \tag{12}$$

where f_0, f_1 are concave functions on the convex feasible region S, $f_1>0$ and θ is a random variable with a continuous cumulative distribution function. Then the maximum probability model for (12) gives rise to the fractional program

$$\max\left\{\frac{f_0(x)-k}{f_1(x)}\colon\ x\in S\right\}. \tag{13}$$

For a linear program (12) the deterministic equivalent (13) becomes a linear fractional program. If f_0 is concave and f_1 linear, then (13) is still a concave fractional program. However, if f_1 is also a (nonlinear) concave function, then (13) is no longer a concave fractional program. Obviously a quadratic program (12) reduces to a quadratic fractional program. For more details on (12), (13) see [57,68].

Stochastic programs (9) and (12) are met in a wide variety of planning problems. Whenever the maximum

probability model is used as a deterministic equivalent, such decision problems lead to a fractional program of one type or another. Hence, fractional programs are encountered indirectly in many different applications of mathematical programming, although initially the objective function is not a ratio.

With the recent advent of various interior point methods for linear programming problems fractional programming has been given more attention as well. For instance, K.M. Anstreicher [2] showed that Karmarkar's projective algorithm is fundamentally an algorithm for linear fractional programming on a simplex.

M. Gaudioso and M.F. Monaco [34] use quadratic fractional programs as subproblems in an algorithm for convex nondifferentiable programs. These arise as duals of search direction subproblems.

Theoretical and Algorithmic Results

Most of the algorithms known so far solve linear, or more generally, concave fractional programs (8). At least five different strategies are found in the literature and are reviewed below.

Solving Problem (P) Directly

Concave (linear) fractional programs share some important properties with concave (linear) programs, due to the generalized concavity (and in addition, generalized convexity in the linear case) of the objective function $q(x) = f(x)/g(x)$ [4,45]:

1) a local maximum is a global maximum;
2) a maximum is unique if either the numerator is strictly concave or the denominator is strictly convex;
3) a solution of the Karush–Kuhn–Tucker optimality conditions is a maximum, assuming f, g, h_k are differentiable on the open set C;
4) a maximum is attained at an extreme point of the convex polyhedron S of a linear fractional program (provided an optimal solution exists).

Because of the properties 1) and 3), it is possible to solve concave fractional programs by several of the standard concave programming algorithms. Indeed, it was shown that certain concave programming methods can be applied to programs with a quasiconcave objective function [45]; for example, the Frank–Wolfe linearization method [23,45]. M. Boncompte and J.E. Martinez-Legaz [14] proposed a cutting plane method for concave fractional programs, based on the upper subdifferentiability of the objective function. If (P) is a linear fractional program, then property 4) can be used to calculate a maximum $\overline{x}$ by determining a finite sequence of extreme points x_i of S with increasing values $q(x_i)$ converging to $\overline{x}$. Thus a simple simplex-like procedure can solve linear fractional programs [45].

Solving an Equivalent Problem (P_{eq})

Some of the concave programming algorithms are not suitable for generalized concave programs [45]. Thus the choice of concave programming algorithms to solve concave fractional programs directly is limited. However, it can be shown that every concave fractional program is transformable into a concave program: the variable transformation

$$y = \frac{x}{g(x)} \quad \text{and} \quad t = \frac{1}{g(x)} \tag{14}$$

reduces (P) to

$$(P_{eq}) \quad \sup \left\{ tf\left(\frac{y}{t}\right) : \ (y, t) \in \widetilde{S} \right\} \tag{15}$$

with the region $\widetilde{S}$ represented by the relations

$$th_k\left(\frac{y}{t}\right) \le 0, \quad tg\left(\frac{y}{t}\right) \le 1, \quad \frac{y}{t} \in C, \quad t > 0, \tag{16}$$

and this is a concave program [55]. If $(\overline{y}, \overline{t})$ is an optimal solution of (P_{eq}), then $\overline{x} = \overline{y}/\overline{t}$ is an optimal solution of (P). Such a transformation was originally suggested by Charnes and Cooper [19] who showed that with help of (14) a linear fractional program can be reduced to a linear program. Because of the transformability into a concave program, concave fractional programs can indirectly be solved by *any* concave programming method, applying the algorithm to the equivalent program (P_{eq}). Hence through transformation (14) one gains access to *all* convave programming algorithms.

To solve (P_{eq}) rather than (P) may be particularly appropriate when the numerator f and the denominator g have a certain algebraic structure. For example, the maximum probability model (11) or certain portfolio selection models have an affine numerator and a denominator which is the square root of a convex quadratic form. In this case (P_{eq}) reduces to a concave quadratic program, and hence (P) can directly be solved

by one of the standard quadratic programming techniques [59]. In the special case of a linear fractional program (7) transformation (14) yields the linear program

$$\sup\left\{c^\top y + \alpha t\colon\ (y,t) \in \widehat{S}\right\} \tag{17}$$

with the feasible region $\widehat{S}$ represented by the relations

$$Ay - bt \le 0, \quad d^\top y + \beta t = 1, \quad y \ge 0, \quad t > 0.$$

Hence (7) can be solved by the simplex method [19]. For a comparison with other linear fractional programming methods see [60].

Solving a Dual Problem (D)

One of the disadvantages of solving (P) directly is that duality concepts of concave programming cannot be used since basic duality relations are no longer valid for these nonconcave programs. However, transformation (14) enables us to gain access to concave programming duality. Thus a dual fractional program can be defined as one of the classical duals of the equivalent concave program (P_{eq}) [55]. For instance, the Lagrangian dual of (P_{eq}) gives rise to the dual fractional program

$$\text{(D)}\quad \inf\left\{\sup_{x\in C}\left\{\frac{f(x)-u^\top h(x)}{g(x)}\colon\ u \ge 0\right\}\right\}, \tag{18}$$

where $h = (h_1, \ldots, h_m)^\top$. As in concave programming, several duality relationships can be established between (P) and (D) [55].

Various duals have been suggested in different approaches [57,59]. However, not much effort has been devoted to algorithmically using duality. In [56] the dual is used to calculate bounds in an iterative procedure for concave fractional programs. Much remains to be done to take full advantage of fractional programming duality in algorithms.

For the dual (D) to be a computationally attractive alternative to (P) or (P_{eq}), the fractional program (P) should have a certain amount of algebraic structure in f, g and h_k. Otherwise it may well be easier to solve (P) rather than a dual of (P). If (P) is a concave quadratic fractional program with an affine denominator, then the dual can be written as a linear program with one additional concave quadratic constraint [59,65].

One advantage of a dual method is that in addition to an optimal solution of (P) also the sensitivity of the maximal value of $q(x)$ with regard to right-hand side changes can be calculated. The dual variables $\overline{u}_i$ in an optimal solution turn out to be propertional to the marginal values of $q(x)$ at $\overline{x}$ [57,58,59]. Sensitivity and parametric analysis for fractional programming has been extensively discussed; see [18,23,57,58] and the cited references.

Solving a Parametric Problem (P_q)

There is a rich class of algorithms based on the following parametric problem associated with (P):

$$(P_q)\quad \max\{f(x) - qg(x)\colon\ x \in S\}, \tag{19}$$

where $q \in \mathbf{R}$ is a parameter. The program (P_q) is sometimes numerically more tractable than the program (P). For example, (P_q) is a parametric quadratic (linear) program if (P) is a quadratic (linear) fractional program, and (P_q) is a parametric concave program if (P) is a concave fractional program. M. Sniedovich [67] analyzed the relationship between (P_q) and classical optimization techniques applied to (P).

In the following it is assumed that S is compact and f, g are continuous on S. Let $F(q)$ denote the optimal value of the objective function of (P_q). $F(q)$ is a strictly decreasing, convex function which has a unique zero $q = \overline{q}$. An optimal solution $\overline{x}$ of ($P_{\overline{q}}$)) is also an optimal solution of (P) with $\overline{q} = f(\overline{x})/g(\overline{x})$. Thus solving (P) is equivalent to finding the unique root of the nonlinear equation $F(q) = 0$. With the properties of $F(q)$ in view, T. Ibaraki [39] applied various classical search techniques to calculate the zero $q = \overline{q}$. These interval-type algorithms generate a succession of intervals with decreasing amplitude containing $q = \overline{q}$. Computational results are reported in [39,62]. The application of Newton's method is commonly referred to as the algorithm by W. Dinkelbach, who first proposed such a procedure [28]. Its equivalence to Newton's method was seen later by Ibaraki. A very efficient version of Dinkelbach's method was suggested in [51] improving an earlier variant in [56].

Interior Point Algorithms

In addition to the four more classical strategies above, recently new techniques have emerged which are of the interior point type. The first such method, developed for linear fractional programs, is due to Anstre-

icher [2]. In 1994, R.W. Freund and F. Jarre [32] proposed a method for concave fractional programs. A polynomial convergence is established and some numerical results are reported.

Most of the computational work in single-ratio fractional programming compares and tests algorithms that use the parametric program ($\mathbf{P}_q$). Much more work is needed to compare computationally the various four approaches above with each other and with the very recent polynomial time interior point methods. Also new methods need to be developed for certain nonconcave fractional programs arising in applications; e. g., [59].

This section on single-ratio fractional programming concludes with a brief discussion of integer fractional programming. In some of the economic applications above the variables are restricted to be integers, if indivisible goods are involved. Also, a number of combinatorial fractional programs with 0–1 variables are of interest; for instance fractional location problems [5].

Integer Fractional Programming

This is an important, but somewhat neglected field within fractional programming. In [5] A.I. Barros gives an overview of some of the advances. Here the parametric procedure by N. Megiddo [46] stands out particularly. T. Radzik [53] provided a detailed survey of the advances in *combinatorial fractional programming*. The survey includes many of his own complexity results on Dinkelbach's and Megiddo's parametric procedures. Among others, Radzik shows that Dinkelbach's algorithm solves a combinatorial linear fractional program in a strongly polynomial number of iterations, regardless of the constraint structure. Some of the results in [53] are specialized to cases such as the problem of profit-to-time cycles and maximum mean-weight cuts. In the same survey also open problems in combinatorial fractional programming are identified.

Leaving the single-ratio case now, the three multi-ratio fractional programs in (4), (5) and (6) will be addressed below. Among these, so far best researched is the following.

Maximization of the Smallest of Several Ratios

Consider the Problem

$$\sup\left\{\min_{1\le i\le p} q_i(x)\colon\ x\in S\right\}, \tag{20}$$

where $q_i(x)=f_i(x)/g_i(x)$ and

$$S=\{x\in C\colon\ h_k(x)\le 0,\ k=1,\dots,m\}.$$

It is assumed that $C\subseteq \mathbf{R}^n$ is nonempty, convex and h_k are real-valued convex functions on C. Before analyzing (20), some applications of this model are outlined.

Applications

In mathematical economics problem (20) may arise when the growth rate of an expanding economy is determined [50]:

$$\text{growth rate}=\max_x\left(\min_{1\le i\le p}\frac{\text{output}_i(x)}{\text{input}_i(x)}\right), \tag{21}$$

where x denotes a feasible production plan of the economy. In management science simultaneous maximization of rates such as those discussed earlier can lead to (20). This is so if either in a worst-case approach the model

$$\min_{1\le i\le p}\frac{f_i(x)}{g_i(x)}\to\sup \tag{22}$$

is used or with the help of prescribed ratio goals r_i

$$\max_{1\le i\le p}\left|\frac{f_i(x)}{g_i(x)}-r_i\right|\to\inf \tag{23}$$

is employed. In both cases essentially a *max-min fractional program* (20) is to be solved. Examples of the second approach are found in financial planning with different financial ratios or in the allocation of funds under equity considerations. Furthermore (20) was recently used in location analysis; see [5] for details. A third area of application of model (20) is numerical mathematics. Given the values F_i of a function $F(t)$ in finitely many points t_i of an interval for which an approximating ratio of two polynomials $N(t,x_1)$ and $D(t,x_2)$ with coefficient vectors x_1, x_2 is sought. If the best approximation is defined in the sense of the L_∞-norm, then the following problem is to be solved:

$$\max_i\left|\frac{N(t_i,x_1)}{D(t_i,x_2)}-F_i\right|\to\inf \tag{24}$$

with variables x_1, x_2. Like (23), this problem can be reduced to a max-min fractional program (20).

Theoretical and Algorithmic Results

Several authors, including von Neumann [50], have introduced dual programs for problem (20) employing different approaches; see [60]. In most duality approaches the following assumptions are made: C is nonempty, convex and compact, $-f_i$, g_i, h_k are lower semicontinuous, $-f_i$, g_i, h_k are convex, g_i are positive on C, f_i are nonnegative on S, if at least one g_i is not affine, and the feasible region S is nonempty. Let $F = (f_1, \ldots, f_p)^\top$, $G = (g_1, \ldots, g_p)^\top$ and $h = (h_1, \ldots, h_m)^\top$. The following dual is derived in [40] with help of the Farkas lemma (cf. ► Farkas lemma; ► Farkas lemma: Generalizations):

$$\inf_{\substack{u\geq 0,\\ v\geq 0, v\neq 0}} \left\{ \sup_{x\in C} \left\{ \frac{v^\top F(x) - u^\top h(x)}{v^\top G(x)} \right\} \right\}. \tag{25}$$

Under the assumptions above the optimal values in (20) and (25) coincide, and duality relations much like those in concave and linear programming hold [40]; see also [25].

The primal max-min program (20) is associated with a dual *min-max fractional program* (25). Such a symmetry is not obvious in single-ratio fractional programming duality theory; see (18). Symmetry between the primal and dual exists also in the following sense: in both problems a local optimum is a global optimum. This follows from the fact that the primal objective function is semistrictly quasiconcave and the dual objective function is semistrictly quasiconvex [4]. The dual objective function usually involves infinitely many ratios in contrast to the primal one. However, this asymmetry disappears in case of a linear problem (20) where f_i, g_i, h_k are affine and C is the (unbounded) nonnegative orthant of $\mathbf{R}^n$. Then only finitely many ratios need to be considered in the dual objective function. In the linear case it can further be shown that in addition to the usual complementary slackness between variables in one problem and constraints in the other one, complementary slackness also exists between certain variables in one and ratios in the other one [25]. Hence in the linear case of (20) there exist complete symmetry as well as a close relationship between the primal and the dual fractional program.

Regarding solution methods for (20), an extension of Dinkelbach's algorithm to (20) was introduced by J.P. Crouzeix, J.A. Ferland and Schaible in [26]. It proved to have attractive convergence properties and became the starting point for the design of similar methods surveyed in [24]. Several of these interval-type methods have been compared and tested. M. Gugat [36] proposed a fast interval-type algorithm for (20) which always converges superlinearly. Boncompte and Martinez–Legaz [14] used a cutting plane approach, originally suggested in [52] for a more general class of quasiconcave problems, employing upper subdifferentiability of the objective function in (20). A computational comparison with the Dinkelbach-type method in [26] is carried out too. A different cutting plane method incorporating the ideas of [52] and [67], again for a more general class of problems than generalized fractional programs, is discussed in [7]. In case of problem (20) the method in [7] reduces to the Dinkelbach-type method in [26]. Thus the latter can also be viewed as a cutting plane method.

Most of the algorithms above solve the primal problem (20). In the linear case the Dinkelbach-type algorithm in [26] can also be applied to the dual because of symmetry between (P) and (D). Recently a 'dual' algorithm for (20) was proposed in the nonlinear case [10]. It can be viewed as an extension of the Dinkelbach-type algorithm in [26] applied to the dual of a generalized linear fractional program. In [9] a new dual of (20) was proposed as well as an efficient method to solve it. Less restrictive assumptions ensure superlinear convergence of this new 'dual' algorithm. An extensive computational comparison of the Dinkelbach-type method in [26] with the two dual methods was performed by Barros, J.B.G. Frenk, Schaible and S. Zhang; see [5,9,10]. The test problems involve quadratic ratios.

Freund and Jarre [33] proposed an interior point method for solving (20) which extends their method in [32] for single-ratio problems. Furthermore, A.S. Nemirovsky and Yu.E. Nesterov [48,49] developed several interior point algorithms for (20) which converge in polynomial time. The studies above contain thorough complexity analyses. Summarizing, one can say that max-min fractional programs have been researched quite extensively.

Maximizing a Sum of Ratios

Consider the multi-ratio fractional program

$$\sup\left\{\sum_{i=1}^{p} q_i(x):\ x\in S\right\}, \tag{26}$$

where $q_i(x)=f_i(x)/g_i(x)$, $g_i(x)>0$.

Applications

Model (26) arises naturally in decision making when several rates are to be optimized simultaneously and a compromise is sought which optimizes a weighted sum of these rates. In light of the applications in the single-ratio case, numerators and denominators may represent profit, cost, capital, risk or time, for example. Model (26) also includes the case where some ratios are not proper quotients, i. e., $g_i(x)=1$. This describes situations where a compromise is sought between absolute and relative terms like profit and return on investment (profit/capital) or return and return/risk. Additional applications of (26) are surveyed in [61]. To mention a few, Y. Almogy and O. Levin [1] analyze a multistage stochastic shipping problem and show that a deterministic equivalent of this stochastic problem leads to (26). M.R. Rao [54] discusses various models in cluster analysis. The problem of optimal partitioning of a given set of entities into a number of mutually exclusive and exhaustive groups (clusters) gives rise to various mathematical programming problems, depending on which optimality criterion is used. If the objective is to minimize the sum of the average squared distances within groups, then a minimum of a sum of ratios is to be determined. H. Konno and M. Inori [42] formulated a bond portfolio optimization problem in the form (26).

Theoretical and Algorithmic Results

As seen earlier, the case of ratios of concave and convex functions is of particular interest in applications. Fortunately, it lends itself to a relatively easy analysis of models (8) and (20). A local maximum is a global one, duality relations can be established and several efficient solution techniques are available. Unfortunately, for the sum-of-ratios problem none of this is true any longer if in (26) all ratios $f_i(x)/g_i(x)$ are quotients of concave and convex functions. In particular, a local maximum is usually not a global one, even in the case of linear ratios. More often the objective function is not quasiconcave. I.A. Bykadorov [15] studied certain generalized concavity properties of sums of linear ratios and, more generally, of sums of ratios of polynomials. Only some limited theoretical results are known for the sum of concave ratios; see [23] and the surveys [60,61]. In the case of linear ratios, C.H. Scott and T.R. Jefferson [64] proposed a duality concept for (26) using geometric programming duality.

Given the small theoretical basis, it is not surprising that algorithmic advances have been rather limited too. Several strategies have been proposed and are surveyed in [61]. The best tested method can be found in [43]. Separating numerators and denominators with help of additional variables, problem (26) is embedded into a higher-dimensional space with a concave objective function. A global minimum is then found through approximation techniques. Computational experience with the related multiplicative program in [43] shows that the method works quite well for up to about four terms. However, for more terms in the sum it looses its efficiency fast. Much work is still necessary to develop efficient algorithms for (26), even in the case of linear ratios.

Multi-objective Fractional Programming

The problem of simultaneously maximizing several ratios leads to a *multi-objective fractional program*

$$\max\left\{(q_1(x),\ldots,q_p(x)):\ x\in S\right\}, \tag{27}$$

where $q_i(x)=f_i(x)/g_i(x)$, $g_i(x)>0$. Such a model arises when in contrast to the previous two models (20) and (26) a unifying objective is not considered. Instead, the decision-maker is to be provided with some or all efficient (Pareto optimal) alternatives. These are feasible solutions such that no ratio can be further increased without decreasing at least one of the other ratios. Applications, for instance in financial planning or production planning, can easily be envisioned in light of the applications of fractional programming described earlier; see also [44,60,68] and references therein. In case of concave ratios problem (27) can be seen as a special case of a semistrictly quasiconcave multi-objective programming problem; e. g., [17] and articles in [16,27,41,63,66]. Duality for multi-objective fractional programs has been studied by several authors,

usually for concave or linear ratios; see [11,68]. For such problems also equivalences to multi-objective programs without ratios have been established [68]. These are formed with help of the numerator and denominator functions.

Another topic, important from a theoretical and algorithmic point of view, is the question whether the set E of efficient (Pareto optimal) solutions is connected. Only partial answers were available until very recently [60]. Meanwhile connectedness has been shown for continuous concave fractions over a compact convex feasible region. This is a consequence of a more general result in [12] for semistrictly quasiconcave objective functions. Several solution methods for the calculation of (weakly, proper) efficient solutions have been proposed for linear and concave ratios; see [44,68] and cited references. It is noted that the calculation of E may simplify the solution of the difficult sum-of-ratios problem [60] since an optimal solution of (26) is an efficient solution. Such an approach seems to be particularly promising in case of two ratios. In summary, some good progress has been made in the analysis of concave multi-objective fractional programs. However more work is needed.

Conclusion

Many interesting problems inside and outside management decision making gives rise to the optimization of one or several ratios. Much effort has been devoted to the analysis of such nonconcave programs. However, the theoretical basis is still not broad enough, especially for sum-of-ratios problems and, to a lesser extend, for multi-objective fractional programs. The computational experience with fractional programs is also quite limited. Major progress has been made for concave single-ratio and max-min fractional programs. But much more work is necessary for the other fractional programs of interest in applications.

See also

- ► Bilevel Fractional Programming
- ► Fractional Combinatorial Optimization
- ► Quadratic Fractional Programming: Dinkelbach Method

References

1. Almogy Y, Levin O (1970) Parametric analysis of a multistage stochastic shipping problem. In: Lawrence J (ed) Operational Research '69. Tavistock Publ., London, pp 359–370
2. Anstreicher KM (1986) A monotonic projective algorithm for fractional linear programming. Algorithmica, 1(4):483–498
3. Asmussen S (1987) Applied probability and queues. Wiley, New York
4. Avriel M, Diewert WE, Schaible S, Zang I (1988) Generalized concavity. Plenum, New York
5. Barros AI (1998) Discrete and fractional programming. Kluwer, Dordrecht
6. Barros AI, Dekker R, Frenk JBG, Weeren S van (1997) Optimizing a general optimal replacement model by fractional programming techniques. J Global Optim 10:405–423
7. Barros AI, Frenk JBG (1995) Generalized fractional programming and cutting plane algorithms. J Optim Th Appl 87(1):103–120
8. Barros AI, Frenk JBG, Gromicho J (1997) Fractional location problems. Location Sci 5(1):47–58
9. Barros AI, Frenk JBG, Schaible S, Zhang S (1996) A new algorithm for generalized fractional programs. Math Program 72(2):147–175
10. Barros AI, Frenk JBG, Schaible S, Zhang S (1996) Using duality to solve generalized fractional programming problems. J Global Optim 8:139–170
11. Bector CR, Chandra S, Singh C (1990) Duality in multiobjective fractional programming. In: Cambini A, et al (eds) Generalized Convexity and Fractional Programming with Economic Applications. Lecturer Notes Economics and Math Systems, vol 345. Springer, Berlin, pp 232–241
12. Benoist J (1998) Connectedness of the efficient set for strictly quasiconcave sets for strictly quasiconcave sets. J Optim Th Appl, 96:627–654
13. Bereanu B (1965) Decision regions and minimum risk solutions in linear programming. In: Prekopa A (ed) Coll. Applications of Mathematics to Economics. Hungarian Acad. Sci., Budapest, pp 37–42
14. Boncompte M, Martinez-Legaz JE (1991) Fractional programming by lower subdifferentiability techniques. J Optim Th Appl 68(1):95–116
15. Bykadorov IA (1994) On quasiconvexity in fractional programming. In: Komlosi S, Rapcsak T, Schaible S (eds) Generalized Convexity, Proc. Pécs/Hungary, 1992. Lecture Notes Economics and Math Systems, vol 405. Springer, Berlin, pp 281–293
16. Cambini A, Castagnoli E, Martein L, Mazzoleni P, Schaible S (eds) (1990) Generalized convexity and fractional programming with economic applications, Proc. 3rd Internat. Workshop at Pisa, Italy, 1988. Lecture Notes Economics and Math Systems, vol 345. Springer, Berlin

17. Cambini A, Martein L (1998) Generalized concavity in multiobjective programming. In: Crouzeix JP, Martinez-Legaz JE, Volle M (eds) Generalized Convexity, Generalized Monotonicity. Kluwer, Dordrecht, pp 453–467
18. Cambini A, Schaible S, Sodini C (1993) Parametric linear fractional programming for an unbounded feasible region. J Global Optim 3:157–169
19. Charnes A, Cooper WW (1962) Programming with linear fractional functionals. Naval Res Logist Quart, 9:181–186
20. Charnes A, Cooper WW (1963) Deterministic equivalents for optimizing and satisficing under chance constraints. Oper Res, 11:18–39
21. Charnes A, Cooper WW, Lewin AY, Seiford LM (eds) (1994) Data envelopment analysis: Theory, methodology and applications. Kluwer, Dordrecht
22. Charnes A, Cooper WW, Rhodes E (1978) Measuring the efficiency of decision making units. Europ J Oper Res 2:429–444
23. Craven BD (1988) Fractional programming. Sigma Ser Appl Math, vol 4. Heldermann, Berlin
24. Crouzeix JP, Ferland JA (1991) Algorithms for generalized fractional programming. Math Program 52:191–207
25. Crouzeix JP, Ferland JA, Schaible S (1983) Duality in generalized linear fractional programming. Math Program 27(3):342–354
26. Crouzeix JP, Ferland JA, Schaible S (1985) An algorithm for generalized fractional programs. J Optim Th Appl 47(1):35–49
27. Crouzeix JP, Martinez-Legaz JE, Volle M (eds) (1998) Generalized convexity, generalized monotonicity: Recent results, Proc. 5th Internat. Symp. Kluwer, Dordrecht
28. Dinkelbach W (1967) On nonlinear fractional programming. Managem Sci 13:492–498
29. Falk JE (1969) Maximization of signal-to-noise ratio in an optical filter. SIAM J Appl Math 7:582–592
30. Frenk JBG, Dekker R, Kleijn M (1997) On the marginal cost approach in maintenance. J Optim Th Appl 94(3):771–781
31. Frenk JBG, Kleijn MJ (1998) On regenerative processes and inventory control. Pure Appl Math 9:61–94
32. Freund RW, Jarre F (1994) An interior-point method for fractional programs with convex constraints. Math Program 67(3):407–440
33. Freund RW, Jarre F (1995) An interior-point method for multifractional programs with convex constraints. J Optim Th Appl 85:125–161
34. Gaudioso M, Monaco MF (1991) Quadratic approximations in convex non-differentiable optimization. SIAM J Control Optim 29:58–70
35. Gilmore PC, Gomory RE (1963) A linear programming approach to the stock cutting problem – Part II. Oper Res 11:863–888
36. Gugat M (1996) A fast algorithm for a class of generalized fractional programs. Managem Sci 42(10):1493–1499
37. Heinen E (1971) Grundlagen betriebswirtschaftlicher Entscheidungen, Das Zielsystem der Unternehmung, 2nd edn. Gabler, Wiesbaden
38. Hoskins JA, Blom R (1984) Optimal allocation of warehouse personnel: A case study using fractional programming. Focus (UK) 3(2):13–21
39. Ibaraki T (1983) Parametric approaches in fractional programs. Math Program 26(3):345–362
40. Jagannathan R, Schaible S (1983) Duality in generalised fractional programming via Farkas' lemma. J Optim Th Appl 41(3):417–424
41. Komlosi S, Rapcsak T, Schaible S (eds) (1994) Generalized convexity, Proc. 4th Internat. Workshop at Pécs/Hungary, 1992. Lecture Notes Economics and Math Systems, vol 405. Springer, Berlin
42. Konno H, Inori M (1989) Bond porfolio optimiza tion by bilinear fractional programming. J Oper Res Soc Japan 32(2):143–158
43. Konno H, Kuno T, Yajima Y (1994) Global minimization of a generalized convex multiplicative function. J Global Optim 4:47–62
44. Kornbluth JSH, Steuer RE (1981) Multiple objective linear fractional programming. Managem Sci 27:1024–1039
45. Martos B (1975) Nonlinear programming, theory and methods. North-Holland, Amsterdam
46. Megiddo N (1979) Combinatorial optimization with rational objective functions. Math Oper Res 4:414–424
47. Mjelde KM (1983) Methods of the allocation of limited resources. Wiley, New York
48. Nemirovski AS (1997) A long-step method of analytical centers for fractional programs. Math Program 77:191–224
49. Nesterov YuE, Nemirovski AS (1995) An interior point method for generalized linear-fractional problems. Math Program 69:117–204
50. Von Neumann J (1937) "Über ein ökonomisches Gleichungssystem und eine Verallgemeinerung des Brouwerschen Fixpunktsatzes. In: Menger K (ed) Ergebn. eines math. Kolloqiums, vol 8, pp 73–83
51. Pardalos PM, Phillips AT (1991) Global optimization of fractional programs. J Global Optim 1:173–182
52. Plastria F (1985) Lower subdifferentiable functions and their minimization by cutting planes. J Optim Th Appl Appl. 46:37–53
53. Radzik T (1998) Fractional combinatorial optimization. In: Du D-Z, Pardalos PM (eds) Handbook Combinatorial Optim., vol 1. Kluwer, Dordrecht, pp 429–478
54. Rao MR (1971) Cluster analysis and mathematical programming. J Amer Statist Assoc 66:622–626
55. Schaible S (1976) Fractional programming: I, Duality. Managem Sci 22(8):858–867
56. Schaible S (1976) Fractional programming: II, On Dinkelbach's algorithm. Managem Sci 22(8):868–873
57. Schaible S (1978) Analyse und Anwendungen von Quotientenprogrammen, Ein Beitrag zur Planung mit Hilfe der

nichtlinearen Programmierung, Mathematical Systems in Economics, vol 42. Anton Hain Verlag, Meisenheim
58. Schaible S (1981) Fractional programming: Applications and algorithms. Europ J Oper Res 7:111–120
59. Schaible S (1983) Fractional programming. Z Oper Res, Ser A 27(1):39–54
60. Schaible S (1995) Fractional programming. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, 495–608
61. Schaible S (1996) Fractional programming with sums of ratios. In: Castagnoli E, Giorgi G (eds) Scalar and Vector Optimization in Economic and Financial Problems, Proc. Workshop Milan, 1995. Elioprint, Perugia, 163–175
62. Schaible S, Ibaraki T (1983) Fractional programming (invited review). Europ J Oper Res 12(4):325–338
63. Schaible S, Ziemba WT (eds) (1981) Generalized concavity in optimization and economics. Proc NATO Advanced Study Inst Vancouver, Canada 1980. Acad. Press, New York
64. Scott CH, Jefferson TR (1998) Duality for a nonconvex sum of ratios. J Optim Th Appl 98:151–159
65. Scott CH, Jefferson TR, Frenk JBG (1998) A duality theory for a class of generalized fractional programs. J Global Optim 12:239–245
66. Singh C, Dass BK (eds) (1989) Continuous-time, fractional and multiobjective programming. Proc. Internat. Conf. St. Lawrence Univ., Canton, NY, 1986, Analytic Publ., Delhi, also: J. Inform. Optim. Sci. 10(1)
67. Sniedovich M (1988) Fractional programming revisited. Europ J Oper Res 33(3):334–341
68. Stancu-Minasian IM (1997) Fractional programming: Theory, methods and applications. Kluwer, Dordrecht

Fractional Zero-One Programming

OLEG PROKOPYEV
Department of Industrial Engineering,
University of Pittsburgh, Pittsburgh, USA

Article Outline

Introduction
Applications
Complexity Issues
Mixed Integer Reformulation
Solution Techniques
References

Introduction

One of the classes of 0-1 optimization problems is the maximization (or minimization) of a sum of ratios of linear 0-1 functions:

$$\max_{x\in\{0,1\}^n} f(x) = \sum_{j=1}^{m} \frac{a_{j0} + a_j^T x}{b_{j0} + b_j^T x}\,, \tag{1}$$

$$\text{s.t. } Dx \le c\,, \tag{2}$$

where $a_j \in \mathbb{R}^n$, $b_j \in \mathbb{R}^n$, $a_{j0} \in \mathbb{R}$, $b_{j0} \in \mathbb{R}$, $D \in \mathbb{R}^{k\times n}$ and $c \in \mathbb{R}^k$. Problem (1)–(2) is referred to as *fractional 0-1 programming problem* [21], or *hyperbolic 0-1 programming problem* [1,20].

Note that if for some j and x in the feasible region (2) the term $b_{j0} + b_j^T x$ is equal to zero, then problem (1)–(2) may not have a finite optimum. Therefore, it is usually assumed that

$$b_{j0} + b_j^T x \neq 0\,, \text{ for all } x \in \{0,1\}^n \text{ and } j = 1,\ldots,m\,. \tag{3}$$

Furthermore, sometimes we can make a stricter assumption and require that all denominators in (1) are positive, i. e.,

$$b_{j0} + b_j^T x > 0\,, \text{ for all } x \in \{0,1\}^n \text{ and } j = 1,\ldots,m\,. \tag{4}$$

A special simplified class of (1)–(2) is the so-called *single-ratio fractional (hyperbolic) 0-1 programming problem*:

$$\max_{x\in\{0,1\}^n} f(x) = \frac{a_0 + \sum_{i=1}^n a_i x_i}{b_0 + \sum_{i=1}^n b_i x_i}\,. \tag{5}$$

Problem (1) can be generalized if instead of linear 0-1 functions we consider 0-1 polynomials:

$$\max_{x\in\{0,1\}^n} f(x) = \sum_j \frac{a_{j0} + \sum_{S\in A_j} a_{jS} \prod_{i\in S} x_i}{b_{j0} + \sum_{T\in B_j} b_{jT} \prod_{i\in T} x_i}\,, \tag{6}$$

where A_j and B_j are families of subsets of $\{1,2,\ldots,n\}$.

In general case, problems of type (1), (5) and (6) can be considered subject to various 0-1 linear and nonlinear constraints. A specific class of fractional 0-1 programming problems, where fractional terms appear not in the objective function, but in the set of constraints, is discussed in [2]:

$$\max_{x\in\{0,1\}^n} g(x) = \sum_{i=1}^{m} w_i x_i\,, \tag{7}$$

$$\text{s.t.} \sum_{j=1}^{m_s} \frac{\alpha_{j0}^s + \sum_{i=1}^n \alpha_{ji}^s x_i}{\beta_{j0}^s + \sum_{i=1}^n \beta_{ji}^s x_i} \geq p_s\,,\ s = 1,\ldots,K\,, \quad (8)$$

where K is the number of fractional constraints.

Finally, we should note here that in contrast to (1)–(2), (6) and (7)–(8), problem (5) received most of the attention in the literature. Detailed surveys on single-ratio fractional combinatorial optimization can be found in [14,19].

Applications

Applications of constrained and unconstrained versions of problems (1)–(2), (5), (6), (7)–(8) arise in scheduling [16], query optimization in data bases and information retrieval [7], service systems design and facility location [3,20], graph theory [11], data mining [2] and other areas [19].

Consider, for example, a problem discussed in [3]. We have a set of customers' regions with Poisson demand rates $a_i (i = 1,\ldots,n)$. These regions can be assigned to a service facility with an exponential service rate b. If we define a 0-1 variable x_i corresponding to each region i such that $x_i = 1$ if region i is serviced by the service facility (and $x_i = 0$, otherwise) then the service facility can be described as an $M/M/1$ queue with arrival rate $\lambda = \sum_{i=1}^n a_i x_i$ and service rate b. If we assume steady-state conditions ($\lambda < b$) then the average waiting time for each customer is equal to

$$\frac{1}{b-\lambda} = \frac{1}{b - \sum_{i=1}^n a_i x_i}\,, \quad (9)$$

and the total average waiting time is given by

$$\frac{\sum_{i=1}^n a_i x_i}{b - \sum_{i=1}^n a_i x_i}\,. \quad (10)$$

Next suppose that the customers' region i contributes profit p_i and the penalty for delay per unit time/per customer is t. Then in order to maximize the profit we need to solve the following nonlinear knapsack problem

$$\max_{x\in\{0,1\}^n} \sum_{i=1}^n p_i x_i - t \cdot \frac{\sum_{i=1}^n a_i x_i}{b - \sum_{i=1}^n a_i x_i}\,, \quad (11)$$

$$\text{s.t.} \sum_{i=1}^n a_i x_i \leq b\,. \quad (12)$$

Another interesting application of fractional 0-1 programming can be found in graph theory [11]. Let $G = (V, E)$ be an undirected graph. The *density* $d(G)$ of G is defined as the maximum ratio of the number of edges e_H to the number of nodes n_H over all subgraphs $H \subseteq G$, i. e.

$$d(G) = \max_{H\subseteq G} \frac{e_H}{n_H}\,, \quad (13)$$

where e_H and n_H are the number of edges and nodes in the subgraph H. Obviously, the problem of finding $d(G)$ can be formulated as the following fractional 0-1 programming problem:

$$d(G) = \max_{x\in\{0,1\}^n,\ x\neq 0} \frac{\sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i x_j}{2\sum_{j=1}^n x_j}\,, \quad (14)$$

where a_{ij} are the elements of the adjacency matrix of G and n is the number of nodes in G. A similar formulation can also be given for the *arboricity* $\Gamma(G)$ which is defined as the minimum number of edge-disjoint forests into which G can be decomposed [11].

Complexity Issues

Constrained problems (1) and (5) where we optimize a single- or multiple-ratio fractional 0-1 function subject to linear 0-1 constraints, as well as problem (7)–(8) are obviously *NP*-hard since general linear 0-1 programming is their special case if we set $b_{ji} = 0$ and $b_{j0} = 1$ for $j = 1,\ldots,m$ and $i = 1,\ldots,n$.

An unconstrained single-ratio fractional 0-1 programming problem (5), can be solved in polynomial time, see [7], if condition (4) holds. If the denominator can take both negative and positive values, i. e., only (3) holds, single-ratio problem (5) is known to be *NP*-hard [7]. In other words, the sign of the denominator is "the borderline between polynomial and *NP*-hard classes" [7]. Another simple proof of this fact is given in [1]. Recall the classical **SUBSET SUM** problem: Given a set of positive integers $S = \{s_1,\ldots,s_n\}$ and a positive integer K, does there exist a vector $x \in \{0,1\}^n$, such that

$$\sum_{i=1}^n s_i x_i = K\,? \quad (15)$$

This problem is known to be *NP*-complete [4]. With the instance of the **SUBSET SUM** problem we associate the following unconstrained single-ratio fractional 0-1 programming problem:

$$\max_{x\in\{0,1\}^n} \frac{1}{1-2(\sum_{i=1}^n s_i x_i - K)}\,. \tag{16}$$

It is easy to observe that (3) holds and the solution of (16) is equal to 1 if and only if the **SUBSET SUM** has a solution, which implies the necessary result. Furthermore, it can be easily shown that finding an approximate solution of (5) within any positive multiple of the (negative) optimal value is *NP*-hard [7].

For multiple-ratio problem (1) with (4) satisfied, the number of ratios ($m = 1$, or $m \geq 2$) defines complexity of the problem. For $m = 1$ we have a classical polynomially solvable single-ratio case, while for $m = 2$, that is the 2-ratio case, the problem becomes *NP*-hard (see [18] or [13]).

Some other aspects of the complexity of unconstrained single- and multiple-ratio fractional 0-1 programming problems (1) and (5), including complexity of uniqueness, approximability and local search, are addressed in [12,13].

Mixed Integer Reformulation

Li [9] and Wu [21] suggested a straightforward linearization technique for (1) based on a simple well-known idea: a polynomial mixed 0-1 term $z = xy$, where x is a 0-1 variable, and y is a continuous variable taking any positive value, can be represented by the following linear inequalities: (1) $y - z \leq K - Kx$; (2) $z \leq y$; (3) $z \leq Kx$; (4) $z \geq 0$, where K is an upper bound on y.

Assume that (4) is satisfied. Define a new variable y_j for each ratio in (1) that is

$$y_j = \frac{1}{b_{j0} + \sum_{i=1}^n b_{ji} x_i}\,. \tag{17}$$

Then fractional 0-1 programming problem (1) can be equivalently expressed as:

$$\max_{x\in\{0,1\}^n} \sum_{j=1}^m a_{j0} y_j + \sum_{j=1}^m \sum_{i=1}^n a_{ji} u_{ji}$$

$$\text{s.t. } Dx \leq c$$

$$b_{j0} y_j + \sum_{i=1}^n b_{ji} u_{ji} = 1 \quad j = 1,\ldots,m$$

$$y_j - K_j(1 - x_i) \leq u_{ji} \leq K_j x_i \quad j = 1,\ldots,m;\ i = 1,\ldots,n$$

$$0 \leq u_{ji} \leq y_j \quad j = 1,\ldots,m;\ i = 1,\ldots,n\,, \tag{18}$$

where a new variable u_{ij} is introduced for each nonlinear term $y_j x_i$, and K_j is an upper bound on y_j.

Additional, though similar in spirit to (18), linear mixed 0-1 reformulations as well as other related issues are carefully discussed in [20].

Solution Techniques

Most of the research efforts have been focused on solving various classes of single-ratio problem (5). Among developed solution techniques we should mention branch-and-bound [15], cutting plane [5], enumeration [6] and approximation algorithms [8]. However, most popular methods for solving single-ratio fractional 0-1 programming (and general fractional combinatorial) problems are based on the *parametric approach* [10,11,14].

For some classes of multiple-ratio fractional 0-1 programming problems, there are developed specialized algorithms [2,3,16,17,20]. More recent examples include a highly efficient cutting-plane algorithm for solving problem (11)–(12) [3] and a heuristic for solving special classes of fractionally constrained problems of type (7)–(8) [2]. Reported computational experiments involved test instances with the size of up to 10,000 variables.

Unfortunately, the fractional programming problem becomes substantially more difficult if we introduce additional ratios in the objective function. General multiple-ratio problem (1)–(2) can be solved utilizing standard branch-and-bound methods after reformulation into linear mixed 0-1 programming problem via techniques discussed in [9,20,21]. An improved branch-and-bound algorithm based on *node tightening* is developed in [20].

References

1. Boros E, Hammer P (2002) Pseudo-Boolean Optimization. Discret Appl Math 123(1-3):155–225

2. Busygin S, Prokopyev OA, Pardalos PM (2005) Feature Selection for Consistent Biclustering via Fractional 0-1 Programming. J Comb Optim 10(1):7–21
3. Elhedhli S (2005) Exact solution of a class of nonlinear knapsack problems. Oper Res Lett 33:615–624
4. Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of *NP*-Completeness. WH Freeman, San Francisco
5. Granot D, Granot F (1976) On solving fractional (0-1) programs by implicit enumeration. INFOR 14:241–249
6. Granot D, Granot F (1977) On integer and mixed integer fractional programming problems. Ann Discret Math 1:221–231
7. Hansen P, Poggi de Aragão M, Ribeiro CC (1991) Hyperbolic 0-1 programming and query optimization in information retrieval. Math Program 52:256–263
8. Hashizume S, Fukushima M, Katoh N, Ibaraki T (1987) Approximation algorithms for combinatorial fractional programming problems. Math Program 37:255–267
9. Li H (1994) A global approach for general 0-1 fractional programming. Eur J Oper Res 73:590–596
10. Megiddo N (1979) Combinatorial optimization with rational objective functions. Math Oper Res 4:414–424
11. Picard J-C, Queyranne M (1982) A network flow solution to some nonlinear 0-1 programming problems, with applications to graph theory. Networks 12:141–159
12. Prokopyev OA, Huang HX, Pardalos PM (2005) On complexity of unconstrained hyperbolic 0-1 programming problems. Oper Res Lett 33:312–318
13. Prokopyev OA, Meneses C, Oliveira CAS, Pardalos PM (2005) On Multiple-ratio Hyperbolic 0-1 Programming Problems. Pac J Optim 1/2:327–345
14. Radzik T (1998) Fractional Combinatorial Optimization. In: Du D-Z, Pardalos PM (eds) Handbook of Combinatorial Optimization, vol 1. Kluwer, Dordrecht, pp 429–478
15. Robillard P (1971) (0,1) Hyperbolic programming problems. Nav Res Logist Q 18:47–57
16. Saipe S (1975) Solving a (0,1) hyperbolic program by branch and bound. Nav Res Logist Q 22:497–515
17. Skiscim CC, Palocsay SW (2001) Minimum Spanning Trees with Sums of Ratios. J Glob Optim 19:103–120
18. Skiscim CC, Palocsay SW (2004) The Complexity of Minimum Ratio Spanning Tree Problems. J Glob Optim 30: 335–346
19. Stancu-Minasian IM (1987) Fractional Programming. Kluwer, Dordrecht
20. Tawarmalani M, Ahmed S, Sahinidis N (2002) Global Optimization of 0-1 Hyperbolic Programs. J Glob Optim 24:385–416
21. Wu T-H (1997) A note on a global approach for general 0-1 fractional programming. Eur J Oper Res 101:1997

Frank–Wolfe Algorithm

SIRIPHONG LAWPHONGPANICH
Naval Postgraduate School, Monterey, USA

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Away direction; Bisection search; Bounded; Column generation; Convergence rate; Convex function; Convex hull; Convexity; Dantzig–Wolfe decomposition; Differentiable function; Direction finding problem; Extreme point; Feasible direction; Feasible direction methods; First order Taylor series expansion; Geometrically; Globally optimal; Golden section method; Heuristics; Inexact line search line; Search algorithms; Line search problem; Linear program; Matrix; Network; Nonempty; Nonlinear programming; Parallel tangents; PARTAN; Polyhedron; Positive definite matrix; Pseudoconvex; Quadratic programming; Regularized direction finding problem; Regularized Frank–Wolfe decomposition; Simplex algorithm; Simplicial decomposition; Stepsize; Stopping criterion; Strongly convex

In 1956, M. Frank and P. Wolfe [5] published an article proposing an algorithm for solving quadratic programming problems. In the same article, they extended their algorithm to the following problem:

$$\min_{x \in S} f(x), \tag{1}$$

where $f(x)$ is a convex and continuously differentiable function on $\mathbf{R}^n$. The set S is a nonempty and bounded polyhedron of the form $S = \{x \in \mathbf{R}^n : Ax \leq b, x \geq 0\}$, where A is a $m \times n$ matrix and $b \in \mathbf{R}^m$. The algorithm belongs to the class of *feasible direction methods for nonlinear programming problems*. Starting from a feasible solution, algorithms in this class solve (1) by iteratively generating a feasible direction that leads to another feasible solution with an improved objective

function value. The Frank–Wolfe (FW) algorithm for (1) can be stated as follows:

0	Select $x^1 \in S$ and set $k = 1$.
1	Let $y_k = \arg\min_{y \in S} \nabla f(x^k)^\top y$.
	IF $\nabla f(x^k)^\top (y^k - x^k) \geq 0$
	THEN stop and x^k is an optimal solution
	ELSE go to Step 2.
2	Let $\lambda^k = \arg\min_{0 \leq \lambda \leq 1} f(x^k + \lambda(y^k - x^k))$.
	Set $x^{k+1} = x^k + \lambda^k (y^k - x^k)$ and $k = k + 1$;
	Go to Step 1.

The Frank–Wolfe algorithm

The problem in Step 1 is generally referred to as the *direction finding problem*, for the vector $d^k = (y^k - x^k)$ is a feasible direction at x^k. Since $\nabla f(x^k)$ is a constant vector with respect to y, the direction finding problem is a linear program and can be solved using the simplex algorithm. Doing so implies that d^k always points toward an extreme point since y^k is always an extreme point of S. When x^k satisfies the stopping criterion, it must be globally optimal because the following holds for all $x \in S$:

$$f(x) \geq f(x^k) + \nabla f(x^k)^\top (x - x^k) \geq f(x^k) + \nabla f(x^k)^\top (y^k - x^k) \geq f(x^k).$$

The three inequalities follow from the convexity of $f(x)$, the fact that y^k solves the direction finding problem, and the stopping criterion, respectively.

When x^k does not satisfy the stopping criterion, $\nabla f(x^k)^\top (y^k - x^k) < 0$ and the algorithm proceeds to Step 2. In this step, λ^k is a solution to a line search problem which has only one a decision variable and can be solved by a number of algorithms such as bisection search, golden section method and an inexact line search technique using, e. g., Armijo's rule [1]. It is important to note that the new solution, x^{k+1}, has a better objective value. To demonstrate, consider the first order Taylor series expansion of $f(x)$ around the point x^k, i. e.,

$$f(x^k + \lambda(y^k - x^k)) = f(x^k) + \lambda \nabla f(x^k)^\top (y^k - x^k) + \lambda \left\| y^k - x^k \right\| \alpha(x^k; \lambda(y^k - x^k)),$$

where $\lim_{\lambda \to 0} \alpha(x^k; \lambda(y^k - x^k)) = 0$. Since $\nabla f(x^k)^\top (y^k - x^k) < 0$, the above expansion implies that there exists a sufficiently small $\widehat{\lambda} \in (0, 1)$ such that $f(x^k + \widehat{\lambda}(y^k - x^k)) < f(x^k)$. Using the fact that λ^k solves the line search problem, the following must hold:

$$f(x^{k+1}) = f(x^k + \lambda^k (y^k - x^k)) \leq f(x^k + \widehat{\lambda}(y^k - x^k)) < f(x^k).$$

Thus, x^{k+1} has a better objective value.

Using standard techniques in nonlinear programming, it can be shown that the sequence of FW iterates, x^k, converges to an optimal solution. This also holds under a weaker assumption that $f(x)$ is pseudoconvex. In [14], W.B. Powell and Y. Sheffi eliminate the line search problem in Step 2 and show that the FW algorithm still converges to an optimal solution as long as λ^k satisfies the following conditions:

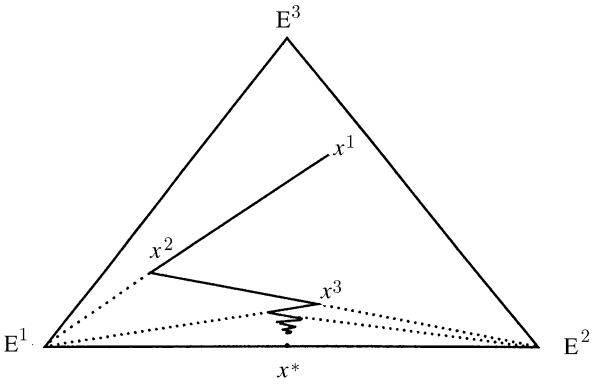
$$\sum_{k=1}^{\infty} \lambda^k \quad \text{and} \quad \lim_{k \to \infty} \lambda^k = 0.$$

For example, one suitable choice is $\lambda^k = 1/k$.

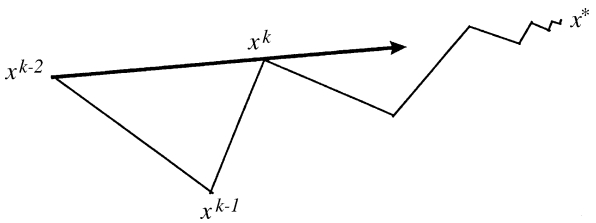
The main advantage of the FW algorithm is in its simplicity. It is easy to understand and implement on a computer. Computer programs for the simplex and the line search algorithms already exist and are generally available. When the constraint matrix A has a network structure (see, e. g., [7,11], and [2]), more efficient network algorithms can be used to solve the direction finding problem and the overall computational time can be reduced. In addition, the FW algorithm does not require much computer storage or memory. However, this feature may be less important as the computer memory becomes available in abundance and at a cheaper price.

The main disadvantage of the FW algorithm is its slow convergence rate. (See Fig. 1.) During the early iterations, the algorithm tends to decrease the objective function rather dramatically. However, the FW iterates tend to zigzag as they slowly approach an optimal solution. In [17], Wolfe shows that the sequence x^k converges geometrically to the optimal solution, if it is in the relative interior of S and $f(x)$ is strongly convex. On the other hand, if the optimal solution is on the boundary of S, the convergence may be slower.

In practice, there are several modifications that can accelerate the convergence of the FW algorithm. The



Frank–Wolfe Algorithm, Figure 1
The problem is: min $\{\|w - x\|^2 : x \in S\}$ where S is the convex hull of E^1, E^2, and E^3. The Frank–Wolfe algorithm generates feasible directions that point toward either E^1 or E^2. It dramatically reduces the objective function during the first two iterations and zigzags toward the optimal solution, x^*, afterward



Frank–Wolfe Algorithm, Figure 2
The PARTAN direction, $p^k = (x^k - x^{k-2})$, points toward an optimal solution

first modification is due to Wolfe [17]. It involves generating in Step 1 an additional feasible direction, $\widehat{d}^k = z^k - x^k$, where $z^k = \arg\max_{z \in S} \nabla f(x^k)^\top z$. The direction $\widehat{d}^k$ is generally referred to as the'away' direction since it is constructed from the worst extreme point with respect to minimizing the objective function. Between the original and the away directions, only one is selected for the line search problem in Step 2. Although the away direction generally leads to a faster convergence in practical applications (see, e. g., [3]), J. Guélat and P. Marcotte [8] showed that the resulting algorithm still converges geometrically to an optimal solution under appropriate assumptions. The second modification is based on the parallel tangents (PARTAN) method introduced in [15]. During the kth iteration, the PARTAN direction, p^k, is defined to be $(x^k - x^{k-2})$ when $k \geq 3$. When the FW algorithm zigzags, p^k intuitively points toward an optimal solution. (See Fig. 2.)

When integrated together, the PARTAN variant (see [4] and [10]) of the FW algorithm alternates between the original and the PARTAN directions when performing line searches. More formally, the original Step 2 of the FW algorithm is replaced with the following steps:

2 Let $\lambda^k = \arg\min_{0 \leq \lambda \leq 1} f(x^k + \lambda(y^k - x^k))$.
Set $z^k = x^k + \lambda^k(y^k - x^k)$.
go to Step 3.

3 (PARTAN step)
IF $k = 1$
THEN set $x^{k+1} = z^k$
ELSE let

$$\alpha^k = \arg\min_{0 \leq \alpha \leq \alpha^k_{\max}} f(x^{k-1} + \alpha(z^k - x^{k-1})),$$

where $\alpha^k_{\max}$ is the maximal stepsize in the direction $(z^k - x^{k-1})$,
set $x^{k+1} = x^{k-1} + \alpha^k(z^k - x^{k-1})$;
set $k = k + 1$;
return to Step 1.

Finally, the last modification for accelerating the FW algorithm involves using some or all of the extreme points generated during the current and prior iterations. Instead of performing a line search in Step 2, a typical modification (see, e. g., [6,9,16] and [12]) either requires a heuristic, approximate, or exact solution to the following problem:

$$\begin{cases} \min & f\left(\sum_{i=1}^{k} \beta_i y^i\right) \\ \text{s.t.} & \sum_{i=1}^{k} \beta_i = 1, \\ & \beta_i \geq 0, \quad i = 1, \ldots, k. \end{cases} \tag{2}$$

The feasible region of (2) is the convex hull of $\{y^1, \ldots, y^k\}$, each of which is an extreme point of S. Thus, (2) is an approximation to (1) and this approximation improves as more extreme points are added to (2). Since the number of extreme points of S is finite, an optimal

solution to (2) should also solve (1) after a finite number of iterations. When (2) is solved exactly (or nearly so), the resulting algorithm is generally known as the *simplicial decomposition* or *column generation* technique and is related to the *Dantzig–Wolfe decomposition*.

In the above three modifications, the direction finding problems are linear programs with the same structure. In 1994, A. Migdalas [13] introduced an extension called the regularized Frank–Wolfe algorithm in which the direction finding problem has a nonlinear term in the objective function to control the distance between y^k and x^k. For example, one version of the regularized direction finding problem is:

$$y^k = \arg\min_{y \in S} \nabla f(x^k)^\top y + \frac{1}{2}(y - x^k)^\top D^k (y - x^k),$$

where D^k is a positive definite matrix.

See also

- ► Rosen's Method, Global Convergence, and Powell's Conjecture

References

1. Armijo L (1966) Minimization of functions having Lipschitz continuous first-partial derivatives. Pacific J Math 16:1–3
2. Collins M, Cooper L, Helgason R, Kennington J, LeBlanc L (1978) Solving the pipe network analysis problem using optimization techniques. Managem Sci 24:747–760
3. Florian M (1977) An improved linear approximation algorithm for the network equilibrium (packet switching) problem. Proc. 1977 IEEE Conf. on Decision and Control, New Orleans, Louisiana, pp 167–196
4. Florian M, Guéelat J, Spiess H (1987) An efficient implementation of the PARTAN variant of the linear approximation method of the network equilibrium problem. Networks 17:319–339
5. Frank M, Wolfe P (1956) An algorithm for quadratic programming. Naval Res Logist Quart 3:95–110
6. Fukushima M (1984) A modified Frank–Wolfe algorithm for solving the traffic assignment problem. Transport Res 18B:169–177
7. Golden BL (1975) A minimum-cost network flow problem concerning imports and exports. Networks 5:331–356
8. Guélat J, Marcotte P (1986) Some comments on Wolfe's away step. Math Program 35:110–119
9. Holloway C (1974) An extension of the Frank–Wolfe method of feasible directions. Math Program 6:14–27
10. LeBlanc LJ, Helgason RV, Boyce DE (1985) Improved efficiency of the Frank-Wolfe algorithm for convex network programs. Transport Sci 19:445–462
11. LeBlanc LJ, Morlok EK, Pierskalla WP (1975) An efficient approach to solving the road network equilibrium traffic assignment problem. Transport Res 9:308–318
12. Meyer G (1974) Accelerated Frank–Wolfe algorithms. SIAM J Control 12:655–663
13. Migdalas A (1994) A regularization of the Frank–Wolfe method and unification of certain nonlinear programming method. Math Program 65:331–346
14. Powell WB, Sheffi Y (1982) The convergence of equilibrium algorithms with predetermined step size. Transport Sci 16:45–55
15. Shah B, Buehler R, Kempthorne O (1964) Some algorithms for minimizing a function of several variables. SIAM J Appl Math 12:74–92
16. Von Hohenbalken B (1977) Simplicial decomposition in nonlinear programming algorithms. Math Program 13:49–68
17. Wolfe P (1970) Convergence theory in nonlinear programming. In: Abadie J (ed) Integer and Nonlinear Programming. North-Holland, Amsterdam, pp 1–36

Frequency Assignment Problem

FAP

Robert Murphey
US Air Force Research Labor., Eglin AFB, USA

MSC2000: 05-XX

Article Outline

Keywords
See also
References

Keywords

Optimization; Graph coloring

The ever growing number of wireless communications systems deployed around the globe has made the optimal assignment of a limited radio frequency spectrum a problem of primary importance. At issue are planning models for permanent spectrum allocation, licensing, regulation [20] and network design to include; aeronautical mobile, land mobile, maritime mobile, broadcast, land fixed (point-to-point) and satellite. Further at issue are on-line algorithms for dynamically assigning frequencies to users within an established network. In particular, land cellular mobile systems have been well

studied (I. Katzela and M. Naghshineh [9] reference nearly 100 works in cellular dynamic channel assignment).

Frequency assignment problems are typically modeled in *graph theoretical* terms. That is, a graph $G(V, E)$ is considered with vertices $V(G) = \{v_1, \ldots, v_n\}$ and edges $E(G)$. Each vertex in $V(G)$ represents a transmitter and two vertices (v_i, v_j) are adjacent (have an edge between them) if the corresponding transmitters are not permitted to share the same frequency. The frequency continuum is partitioned into channels (frequencies) of even width and numbered consecutive integer values. A frequency assignment is then a mapping f of the nonzero positive integers $\mathbf{Z}_+$ to the vertices of the graph such that no two adjacent vertices receive the same value:

$$f\colon\ V \to \mathbb{Z}_+$$
$$\text{s.t. } (v_i, v_j) \in E(G) \Leftrightarrow f(v_i) \neq f(v_j).$$

This formulation, where adjacent vertices cannot share the same frequency is termed *co-channel constrained* and was shown by B.H. Metzger [12] to be equivalent to the well-studied *graph coloring problem*. Typically, the objective is to find an assignment of frequencies (colors) to the transmitters (vertices) that minimizes the number of frequencies (colors) used. The minimum number $\chi(G)$ for which a $\chi(G)$-coloring exists for G is called the *chromatic number*. Since graph K-colorability for arbitrary K is known to be an *NP*-complete problem [6], co-channel constrained frequency assignment is also *NP*-complete.

Consider the restriction that two adjacent vertices may not receive frequencies that are the same or differ by exactly k. This FAP is said to be *adjacent channel constrained* and when $k = 0$ is simply the co-channel problem. Adjacent channel constraints model harmonic interference (signals that are integer multiples of the fundamental or carrier frequency). In general, a set T may be defined which contains zero and a subset of the positive integers such that no two adjacent vertices may receive assignments whose absolute difference is contained in T,

$$f\colon\ V \to \mathbb{Z}_+$$
$$\text{s.t. } (v_i, v_j) \in E(G) \Leftrightarrow \left|f(v_i) - f(v_j)\right| \notin T.$$

This FAP formulation was introduced by W.K. Hale [7] and is termed *T-coloring*. When $T = \{0\}$, the co-channel constrained FAP or graph coloring problem results. M.B. Cozzens and F.S. Roberts [4] define the number of unique colors used in a T-coloring as the *order* and the total bandwidth used (maximum color minus the minimum color) as the *span*. Hence for any T-coloring, two optimality criteria exist: minimum order, denoted by $\chi_T(G)$, and minimum span, denoted by $\mathrm{sp}_T(G)$. For the co-channel constrained FAP $\chi_T(G) = \mathrm{sp}_T(G)$ however, in general, this is not true. Cozzens and Roberts show that for any graph and any T the minimum order is equivalent to the chromatic number; $\chi_T(G) = \chi(G)$. Hence, T-coloring research has primarily been focused on characterizing the minimum span using numerous assumptions about the structure of G and value of T [2,4,5,11,13,16], and [17].

In many situations, the potential for interference between transmitters may occur on several different levels, where each level is defined by a separate set of edges on the common set of vertices. The kth edge set is denoted by the graph G_k, $k = 0, \ldots, K$. The family of graphs thus defined and which share an identical vertex set are sometimes referred to, in unison, as a *multigraph* and denoted by $G(V, G_0, \ldots, G_K)$. Since each level represents a unique interference mechanism, a family of T-sets must be also defined as $T(0), \ldots, T(K)$. Interference occurs on the kth level when any 2 vertices adjacent in the kth edge set receive frequencies that differ by a value in $T(k)$. In graph coloring nomenclature, the multilevel FAP is denoted by

$$f\colon\ V \to \mathbb{Z}_+,$$
$$(x, y) \in E(G_k) \Leftrightarrow |f(x) - f(y)| \notin T(k),$$
$$\forall (x, y) \in V, \quad x \neq y, \quad k = 0, \ldots, K,$$

where the family of graphs are nested such that $G_0 \supseteq \cdots \supseteq G_K$ and the T-sets are reverse nested, as $0 \in T(0) \subseteq \cdots \subseteq T(K)$. Cozzens and D.I. Wang develop bounds on the minimum span for general multigraph T-colorings in [5]. Excellent reviews on T-coloring and frequency assignment for single graphs and multigraphs may be found in [14] and [15].

Since the simplest FAP has been shown to be *NP*-complete, it is generally hopeless to pursue exact solution methods. Approximate heuristic techniques have been the focus of most research and most of these techniques fall under the scope of *sequential heuristics*. There are three fundamental approaches to sequentially

coloring the vertices of a graph:

- (*frequency exhaustive*) Given an ordering of the vertices, attempt to color each vertex, sequentially, the smallest feasible color. This approach is also called a *greedy coloring*.
- (*requirement exhaustive*) Given an ordering of the vertices, attempt to assign the first color to each vertex, sequentially. When all vertices have been considered, attempt to assign the second color to the unassigned vertices, then the third and so on, following the same vertex ordering.
- (*uniform*) Given an ordering of the vertices, attempt to color each vertex, sequentially, the color that has been least used.

The efficiency of each approach is quite dependent upon what ordering the vertices are placed in. There are many rules by which the vertices of a graph can be ordered. In a smallest-last ordering, the vertex of smallest degree in V is denoted v_1. This vertex is then deleted from the graph and the next smallest degree vertex v_2 is found and deleted, and so on until all vertices have been deleted. The smallest-last vertex order is then $\{v_n, v_{n-1}, \ldots, v_1\}$. The largest-first vertex order sorts the vertices of the graph according to their degree in G: largest to smallest. D. Brelaz [3] introduced a vertex ordering specified by the saturation degree of the vertices, from highest saturation degree to lowest. The saturation degree of a vertex is defined to be the number of different colors that exist on the vertices that are adjacent. The vertex with the highest saturation degree is 'most denied' since it has fewer colors to choose from. J.A. Zoellner and C.L. Beall [21] compared the three sequential approaches with several different vertex ordering rules and found that, all else being equal, frequency exhaustive methods typically yields smaller spans. Hale [8] expanded upon these results by defining a generalized structure for all sequential coloring algorithms which consists of three fundamental steps:

1) order the vertices;
2) select the next vertex to color;
3) select the color.

Hale's procedure is general. It cover all types of vertex orderings in step 1 and allows for each of the three sequential techniques in step 3. Step 2 is added to allow the coloring sequence to adapt during the process. Hale introduced new sequential techniques for step 2 that are adaptive variants of the saturation degree. A very good review of frequency assignment heuristics can be found in [10].

Approximate solutions may also be obtained by using more traditional *polyhedral methods* on relaxation problems of the *integer program* (IP) formulation of the FAP. A. Wisse [19] developed a minimum order IP formulation for the FAP which relies on a *list coloring* model, that is, the frequencies which may be assigned are restricted to a finite list (set), designated by F, of cardinality m. Furthermore, I is designated as the index set for all transmitters (vertices) and n the cardinality of I. Define two binary decision variables as

$$x_{if} = \begin{cases} 1 & \text{if transmitter } i \text{ assigned freq } f, \\ 0 & \text{else,} \end{cases}$$

$$y_f = \begin{cases} 1 & \text{if freq } f \text{ used at least once,} \\ 0 & \text{else.} \end{cases}$$

The IP which results is

$$\begin{cases} \min & z = \sum_{f \in F} y_f \\ \text{s.t.} & \sum_{f \in F} x_{if} = 1, \quad \forall i \in I, \\ & \sum_{i \in I} x_{if} \le n y_f, \quad \forall f \in F, \\ & \sum_{g:\, |f-g| \notin T} x_{jg} \le 1 - x_{if}, \\ & \quad \forall f \in F, \quad \forall (v_i, v_j) \in E(G), \\ & x_{if} \in \{0,1\}, \quad \forall i \in I, \ \forall f \in F \\ & y_f \in \{0,1\}, \quad \forall f \in F. \end{cases}$$

A minimum span FAP IP formulation may be had by deleting variable y_f and adding μ, defined to be the maximum frequency assigned,

$$\begin{cases} \min & \mu \\ \text{s.t.} & \sum_{f \in F} x_{if} = 1, \quad \forall i \in I, \\ & \sum_{g:\, |f-g| \notin T} x_{jg} \le 1 - x_{if}, \\ & \quad \forall f \in F, \quad \forall (v_i, v_j) \in E(G), \\ & \sum_{f \in F} f x_{if} \le \mu, \quad \forall i \in I, \\ & x_{if} \in \{0,1\}, \quad \forall i \in I, \quad \forall f \in F, \\ & \mu \in F. \end{cases}$$

Of course a solution to either IP formulation would be exact, however efficient methods for finding solutions to this formulation do not yet exist for problems of large dimension. Linear relaxations of these formulations where $0 \le x_{if} \le 1$, have been successfully developed and yield fairly good solutions for some moderate size problems [1]. A *potential reduction* method [18] has also been developed that utilizes the transformation $x_{if} = 2x_{if} - 1$, that is, $x_{if} \in \{-1, +1\}$. As a result, any feasible solution to the transformed IPs must satisfy $x^{\mathsf{T}}x = mn$ where x is a vector of x_{if} in $\mathbf{Z}_{+}^{mn}$. The polyhedron $\mathcal{P}$ formed from the linear relaxation of x and y or μ in the constraints of the IPs is then incorporated into the problem by minimizing a logarithmic potential function over the polyhedron as

$$\min_{\mathcal{P}} \left[nm - x^{\top}x - \frac{1}{N}\sum_{k=1}^{N} w_k \log s_k \right],$$

where N is the number of constraints, w_k are nonnegative real valued weights, and s_k is the slack of constraint k. A sequence of iterative solutions are obtained in a three step process which begins with a nonoptimal feasible solution x^0. An interior point method is applied to a quadratic approximation of the potential function within an ellipsoid centered on the current feasible point x^i. This yields a decent direction Δx. The potential function is then minimized within the ellipsoid along the line $x^i + \alpha\Delta x$ and yields the next iterate x^{i+1}. The iterate solution is then rounded to an integer value. The algorithm stops when the rounded solution is feasible to the original problem. This algorithm was tested and was found to suffer from slow convergence. As a result, an alternate *quadratic assignment* formulation was developed which proved to be much faster. Define a new binary valued decision variable

$$q_{ifjg} = \begin{cases} 1 & \text{if } x_{if} = 1,\ x_{jg} = 1, \text{ and} \\ & \quad (v_i, v_j) \in E(G), \quad |f - g| \in T, \\ 0 & \text{else.} \end{cases}$$

Then the assignment $F \to x$ has no interference if

$$\sum_{i=1}^{n}\sum_{f=1}^{m}\sum_{j=1}^{n}\sum_{g=1}^{m} x_{if}x_{jg}q_{ifjg} = 0,$$

which is equivalent to

$$\tfrac{1}{2}x^{\top}Qx = 0,$$

where Q is a $mn \times mn$ matrix containing q_{ifjg}. Thus the new potential function, with the added quadratic term, is minimized over the polyhedron as

$$\min_{\mathcal{P}} \left[\frac{1}{2}x^{\top}Qx - \frac{1}{N}\sum_{k=1}^{N} w_k \log s_k \right].$$

Interior point solutions of this potential function converged much more quickly than those of the first formulation.

See also

- Assignment and Matching
- Assignment Methods in Clustering
- Bi-objective Assignment Problem
- Communication Network Assignment Problem
- Graph Coloring
- Maximum Constraint Satisfaction: Relaxations and Upper Bounds
- Maximum Partition Matching
- Quadratic Assignment Problem

References

1. Aardal KI, Hipolito A, Van Hoesel CPM, Jansen B, Roos C, Terlaky T (1995) A branch-and-cut algorithm for the frequency assignment problem. Techn. Report EUCLID CALMA Project, Delft and Eindhoven Univ. Techn., The Netherlands
2. Bonias I (1991) T-colorings of complete graphs. PhD Thesis Dept. Math. Northeastern Univ.
3. Brelaz D (1979) New methods to color the vertices of a graph. Comm ACM 22:251–256
4. Cozzens MB, Roberts FS (1982) T-colorings of graphs and the channel assignment problem. Congressus Numerantium 35:191–208
5. Cozzens MB, Wang DI (1984) The general channel assignment problem. Congressus Numerantium 41:115–129
6. Garey MR, Johnson DS (1979) Computers and intractability. A guide to the theory of NP-completeness. Freeman, New York
7. Hale WK (1980) Frequency assignment: theory and applications. Proc IEEE 68(12):1497–1514
8. Hale WK (1981) New spectrum management tools. IEEE Internat. Symp. Electromagnetic Compatibility, pp 47–53
9. Katzela I, Naghshineh M (1996) Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey. IEEE Personal Comm 33:10–31

10. Lanfear TA (1989) Graph theory and radio frequency assignment. Techn. Report Allied Radio Frequency Agency NATO
11. Liu DD (1991) Graph homomorphisms and the channel assignment problem. PhD Thesis Dept. Math. Univ. South Carolina
12. Metzger BH (1970) 38th Nat. ORSA Meeting. In: Spectrum management techniques
13. Raychaudhuri A (1985) Intersection assignments, T-coloring, and powers of graphs. PhD Thesis Dept. Math. Rutgers Univ.
14. Roberts FS (1991) From garbage to rainbows: generalizations of graph coloring and their applications. In: Alavi Y, Chartrand G, Oellerman OR, Schwenk AJ (eds) Graph Theory, Combinatories, and Applications, vol 2. Wiley, New York, pp 1031–1052
15. Roberts FS (1991) T-colorings of graphs: recent results and open problems. Discret Math 93:229–245
16. Tesman BA (1989) T-colorings, list T-colorings, and set T-colorings of graphs. PhD Thesis Dept. Math. Rutgers Univ.
17. Wang DI (1984) The channel assignment problem and closed neighborhood containment graphs. PhD Thesis Northeastern Univ.
18. Warners JP, Terlaky T, Roos C, Jansen B (1995) A potential reduction approach to the frequency assignment problem. Fac. Techn. Math. and Inform. Delft Univ., The Netherlands, 95–98
19. Wisse A (1994) Mathematical model for the radio link frequency assignment problem. EUCLID CALMA Project, Delft Univ. Techn., The Netherlands
20. Zoellner JA (1973) Frequency assignment games and strategies. IEEE Trans Electromagnetic Compatibility EMC-15:191–196
21. Zoellner JA, Beall CL (1977) A breakthrough in spectrum conserving frequency assignment technology. IEEE Trans Electromagnetic Compatibility EMC-19:313–319

Fundamental Theorem of Algebra

Helmut Maier
Abt. Math. III, University Ulm, Ulm, Germany

MSC2000: 01A55, 01A50, 01A60

Article Outline

Keywords
Analytic Proofs
Topological Proofs
Algebraic Proofs
See also
References

Keywords

Argument principle; Homotopy; Splitting field

Some of the main indicators of progress in the mathematical sciences have been the occurrences of new types of numbers. One of the more recent cases are the complex numbers. Much of modern science cannot be imagined without their use.

Their introduction into mathematics first had been motivated by the wish to solve the equation

$$P(z) = 0, \tag{1}$$

where $P(z)$ is a polynomial.

If one considers only real numbers, such simple equations like $P(z) = z^2 + 1 = 0$ have no solutions. In the field of complex numbers however (1) always at least one solution, if P is a nonconstant polynomial with complex coefficients. This fact is known as the fundamental theorem of algebra. It was first proved rigorously by C.F. Gauss in 1799. Since then a large number of proofs have been found. In this article I give some examples for the main types of proofs: analytic, topological and algebraic.

Analytic Proofs

Possibly the simplest proof, being based on the *Liouville theorem* [3]: Every bounded entire function is a constant.

Assume now that the nonconstant polynomial $P(z)$ has no zero. Since $|P(z)| \to \infty$ for $|z| \to \infty$, the function $f(z) = 1/P(z)$ is bounded and thus a constant by Liouville's theorem. But then also $P(z)$ is constant, a contradiction.

Another, still simple, proof is based on the *argument principle*: The number of zeros of a holomorphic function f inside a simple closed curve γ can be expressed by the integral

$$\frac{1}{2\pi i} \int_\gamma \frac{f'(z)}{f(z)}\, \mathrm{d}z.$$

Let $P(z)$ be a polynomial of degree $n \geq 1$. Choosing for γ the circle around the origin with radius $R > 0$, we obtain for the number N of zeros of $P(z)$:

$$N = \frac{1}{2\pi i} \int_\gamma \frac{P'(z)}{P(z)}\, \mathrm{d}z\,.$$

Since

$$\frac{P'(z)}{P(z)} = \frac{n}{z} + O\left(\frac{1}{|z|^2}\right), \quad |z| \to \infty,$$

we obtain $N = n$ for $R \geq R_0$. Thus $P(z)$ has n zeros (counted with multiplicity).

Topological Proofs

Closely related to the second analytic proof, presented above, is the proof by the concept of homotopy [2].

If X and Y are two topological spaces, then two continuous maps $\varphi_0, \varphi_1 \colon X \to Y$ are called *homotopic* if there exists a continuous map

$$\varphi \colon X \times [0,1] \to Y$$

such that

$$\varphi(x,0) = \varphi_0(x),$$
$$\varphi(x,1) = \varphi_1(x).$$

We choose $X = \{z \in \mathrm{C} \colon |z| = 1\}$, $\mathrm{Y} = \mathrm{C} - 0$. Let $P(z) = z^n + a_{n-1}z^{n-1} + \cdots + a_0 = z^n + Q(z)$, say $n \geq 1$, $a_0 \neq 0$).

For sufficiently large R the two maps

$$\varphi_0 \colon X \to Y, \quad z \longmapsto (Rz)^n,$$
$$\varphi_1 \colon X \to Y, \quad z \longmapsto P(Rz)$$

are homotopic. A homotopy is in fact given by $\varphi(z,t) = (Rz)^n + t\,Q(Rz)$, $z \in X$, $t \in [0,1]$. If there is no zero of $P(z)$ inside the circle $|z| = R$, φ_1 and thus also φ_0 would be homotopic with the constant map

$$\varphi_c \colon X \to Y, \quad z \longmapsto a_0;$$

which can be shown to be false by topological means.

Algebraic Proofs

Since there is no purely algebraic system of axioms for the field of complex numbers there cannot be a purely algebraic proof. However there is a proof which as the only result from analysis uses the intermediate value theorem [1], which we reproduce here.

A statement equivalent to the fundamental theorem is that **C** is the algebraic closure of **R**. We start by showing that every nonconstant polynomial $P(z)$ with *real coefficients* has a complex zero. We proceed by induction.

Let n be the degree of $P(z)$.

i) If n is odd, the claim is an immediate consequence of the intermediate value theorem.

ii) Let $n = 2^t u$ with odd u, $t > 0$, and assume the claim has been proven for $t - 1$.
We select a splitting field S for $P(z)$ over **C**. Then we have a decomposition

$$P(z) = (z - a_1) \cdots (z - a_n) \quad \text{in } S[z].$$

For an arbitrary real number c we form the expressions $b_{ij}(c) = a_i a_j + c(a_i + a_j)$ and the polynomial $Q(z) = \prod_{1 \leq i < j \leq n} (z - b_{ij}(c))$. The coefficients of $Q(z)$ are symmetric polynomials in $a_1, \ldots, a_n$ over **R** and thus real. The degree of $Q(z)$ is $n(n-1)/2 = 2^{t-1}u(2^t u - 1) = 2^{t-1}v$ for an odd number v. By the induction hypothesis $Q(z)$ has at least one zero in **C**. Thus $b_{ij}(c)$ is in **C** for a pair of subscripts (i, j) that may depend on c. If this construction is carried out for all natural numbers c with $1 \leq c \leq 1 + n(n-1)/2$ one finds c and c' belonging to the same pair of subscripts, i. e. there is a pair (i, j) with $b_{ij}(c) \in \mathbf{C}$ and $b_{ij}(c') \in \mathbf{C}$. If one solves the system of equations

$$b_{ij}(c) = a_i a_j + c(a_i + a_j),$$
$$b_{ij}(c') = a_i a_j + c'(a_i + a_j)$$

one obtains $a_i = a/2 \pm \sqrt{a^2 - 4b^2}/2 \in \mathbb{C}$. Thus $P(z)$ has a complex zero.

Let now $P(z) \in \mathbf{C}[z]$ be irreducible and t a zero of $P(z)$ in a splitting field of $P(z)$ over **C**. Then $P(z)$ is the irreducible polynomial of t over **C**. Since t is algebraic over **C** and **C** is algebraic over **R**, t is algebraic over **R**. We denote the irreducible polynomial of t over **R** by $U(z)$. Then $P(z)/U(z)$ in $\mathbf{C}[z]$.

$U(z)$ has at least one zero in **C**. Since **C** is normal over **R**, $U(z)$ splits into linear factors in $\mathbf{C}[z]$. Thus $P(z)$ is linear and $t \in \mathbf{C}$.

See also

► Gröbner Bases for Polynomial Equations

References

1. Körner O (1990) Algebra. Aula-Verlag, Wiesbaden
2. Massey WS (1967) Algebraic topology: An introduction. Springer, Berlin
3. Titchmarsh EC (1939) The theory of functions. Oxford Univ. Press, Oxford

Fuzzy Multi-objective Linear Programming

FMOLP

ROMAN SŁOWIŃSKI
Institute Computing Sci., Poznań University Technol., Poznań, Poland

MSC2000: 90C70, 90C29

Article Outline

Keywords
Flexible Programming
MOLP with Fuzzy Coefficients
Flexible MOLP with Fuzzy Coefficients
Conclusions
See also
References

Keywords

Multi-objective linear programming under uncertainty; Fuzzy sets; Uncertainty modeling; Multicriteria decision making; Interactive procedures

Fuzzy multi-objective linear programming extends the linear programming model (LP) in two important aspects:

- multiple objective functions representing different points of view (criteria) used for evaluation of feasible solutions,
- *uncertainty* inherent to information used in the modeling and solving stage.

A general model of the FMOLP problem can be presented as the following system:

$$[\widetilde{\mathbf{c}}_1\mathbf{x}, \ldots, \widetilde{\mathbf{c}}_k\mathbf{x}] \to \widetilde{\min} \tag{1}$$

such that

$$\widetilde{\mathbf{a}}_i\mathbf{x}\,\widetilde{\leq}\,\widetilde{b}_i, \quad i = 1, \ldots, m, \tag{2}$$

$$\mathbf{x} \geq 0, \tag{3}$$

where $\widetilde{\mathbf{c}}_l = [\widetilde{c}_{l1}, \ldots, \widetilde{c}_{ln}$ $(l = 1, \ldots, k)$, $\mathbf{x} = [x_1, \ldots, x_n]^\mathsf{T}$, $\widetilde{\mathbf{a}}_i = [\widetilde{a}_{i1}, \ldots, \widetilde{a}_{in}$ $(i = 1, \ldots, m)$. The coefficients with the sign of wave are, in general, *fuzzy numbers*, i. e. convex continuous fuzzy subsets of the real line. The wave over min and relation $\leq$ 'fuzzifies' their meaning. Conditions (2) and (3) define a set of feasible solutions (decisions) X. An additional information completing (1) is a set of fuzzy aspiration levels on particular objectives, thought of as goals, denoted by $\widetilde{g}_1, \ldots, \widetilde{g}_k$.

There are three important special cases of the above problem that gave birth to the following classes of problems:

- flexible programming;
- multi-objective linear programming (MOLP) with fuzzy coefficients;
- flexible MOLP with fuzzy coefficients.

In *flexible programming*, coefficients are crisp but there is a fuzzified relation $\widetilde{\leq}$ between objective functions and goals, and between left- and right-hand sides of the constraints. This means that the goals and constraints are fuzzy ('soft') and the key question is the degree of satisfaction. In *MOLP with fuzzy coefficients* all the coefficientsare, in general, fuzzy numbers and the key question is a representation of relation $\leq$ between fuzzy left- and right-hand sides of the constraints. Flexible MOLP with fuzzy coefficients concerns the most general form (1)–(3) and combines the two key questions of the previous problems.

The two first classes of FMOLP problems use different semantics of *fuzzy sets* while the third class combines the two semantics. In flexible programming, fuzzy sets are used to express *preferences* concerning satisfaction of flexible constraints and/or attainment of goals. This semantics is especially important for exploiting information in decision making. The gradedness introduced by fuzzy sets refines the simple binary distinction made by ordinary constraints. It also refines the crisp specification of goals and 'all-or-nothing' decisions. Constraint satisfaction algorithms, optimization techniques and multicriteria decision analysis are typically involving flexible requirements which can be represented by fuzzy relations.

In MOLP with fuzzy coefficients, the semantics of fuzzy sets is related to the representation of incomplete or vague states of information under the form of possibility distributions. This view of fuzzy sets enables representation of *imprecise* or *uncertain information* in mathematical models of decision problems considered in operations research. In models formulated in terms of mathematical programming, the imprecision and uncertainty of information (data) is taken into ac-

count through the use of fuzzy numbers or fuzzy intervals instead of crisp coefficients. It involves fuzzy arithmetic and other mathematical operations on fuzzy numbers that are defined with respect to the famous Zadeh's extension principle.

In flexible MOLP with fuzzy coefficients, the uncertainty and the preference semantics are encountered together. This is typical for decision analysis and operations research where, in order to deal with both uncertain data and flexible requirements, one can use a fuzzy set representation.

Below, we make a tutorial characterization of the three classes of problems and solution methods. For more detailed surveys see, e. g., [16,18,20,27,30,32,36,37].

Flexible Programming

Flexible programming has been considered for the first time in [41] with respect to single-objective linear programming. It is based on a general Bellman–Zadeh principle [2] defining the concept of *fuzzy decision* as an intersection of *fuzzy goals* and *fuzzy constraints*. A fuzzy goal corresponding to objective $\mathbf{c}_l\mathbf{x}$ is defined as a fuzzy set in X; its membership function $\mu_l : X \to [0, 1]$ characterizes the decision maker's aspiration of making $\mathbf{c}_l\mathbf{x}$ 'essentially smaller or equal to g_l'. A fuzzy constraint corresponding to $\mathbf{a}_i\mathbf{x} \tilde{\leq} b_i$ is also defined as a fuzzy set in X; its membership function $\mu_i \to [0, 1]$ characterizes the degree of satisfaction of the ith constraint.

In order to define the membership function $\mu_i(\mathbf{x})$ for the ith fuzzy constraint, one has to know the tolerance margin $d_i \geq 0$ for the right-hand side b_i ($i = 1, \ldots, m$);

$$\mu_i(\mathbf{x}) = \begin{cases} 1 & \text{for } \mathbf{a}_i\mathbf{x} \leq b_i, \\ \text{strictly} & \text{decreasing from 1 to 0} \\ & \text{for } b_i < \mathbf{a}_i\mathbf{x} < b_i + d_i, \\ 0 & \text{for } \mathbf{a}_i\mathbf{x} \geq b_i + d_i. \end{cases} \tag{4}$$

Specifying a membership level α, $\alpha \in [0, 1]$, in [41] the set of feasible solutions ofeach fuzzy constraint has been restricted to the crisp set

$$X_\alpha^i = \{\mathbf{x}\colon\ \mu_i(\mathbf{x}) \geq \alpha\}, \quad i = l, \ldots, m.$$

Then, the set of feasible solutions of a flexible programming problem is $X_\alpha = \cap_{i=1}^m X_\alpha^i$. The single objective function is replaced by the fuzzy goal

$$\mu_G(\mathbf{x}) = \frac{\min_{\mathbf{x}\in X_0}\{\mathbf{cx}\}}{\mathbf{cx}}.$$

To get an optimal solution one has to determine the optimal pair $(\alpha^*, \mathbf{x}^*)$ such that

$$\min\{\alpha^*, \mu(\mathbf{x}^*)\} = \sup \min_\alpha \left(\alpha, \max_{\mathbf{x}\in X_\alpha}\{\mu_G(\mathbf{x})\}\right). \tag{5}$$

If the optimal α^* was determined a priori, the problem(5) could be reduced to a crisp mathematical programming problem where the objective was to find $\mathbf{x}^*$ that maximizes $\mu_G(\mathbf{x})$ on the set X_α^*. In the general case an iterative algorithm is necessary when beginning with any $\alpha_1 \in [0, 1]$, the values α_k and $\max_{\mathbf{x}\in X_{\infty k}}\{\mu_G(\mathbf{x})\}$ converge to the optimum step by step.

H.J. Zimmermann [46] has proposed a more integrative approach to flexible programming allowing consideration of multiple goals and constraints on a common ground. An aspiration level g_l and a tolerance margin $d_l \geq 0$ have to be assumed for the lth goal ($l = 1, \ldots, k$) when assessing the membership function $\mu_l(\mathbf{x})$ as:

$$\mu_l(\mathbf{x}) = \begin{cases} 1 & \text{for } \mathbf{c}_l\mathbf{x} \leq g_l, \\ \text{strictly} & \text{decreasing from 1 to 0} \\ & \text{for } g_l < \mathbf{c}_l\mathbf{x} < g_l + d_l, \\ 0 & \text{for } \mathbf{c}_l\mathbf{x} \geq g_l + d_l. \end{cases} \tag{6}$$

According to the Bellman–Zadeh principle, the set of fuzzy decisions is characterized by an aggregation of the component membership functions. If a conjunctive minimum operator were used for the aggregation, the membership function would be:

$$\mu_D(\mathbf{x}) = \min_{l,i}\{\mu_l(\mathbf{x}), \mu_i(\mathbf{x})\}. \tag{7}$$

Then, the problem of finding the best decision (solution) boils down to the following optimization problem:

$$\begin{cases} \mu_D(\mathbf{x}) & \to \max \\ \text{s.t.} & \mathbf{x} \geq 0. \end{cases} \tag{8}$$

The value of the aggregated function $\mu_D(\mathbf{x})$ can be interpreted as the overall degree of satisfaction of the decision maker with k fuzzy goals and m fuzzy constraints.

In case of minimum operator (7), problem (8) becomes:

$$\begin{cases} v & \to \max \\ \text{s.t.} & v \le \mu_l(\mathbf{x}), \quad l = 1, \ldots, k, \\ & v \le \mu_i(\mathbf{x}), \quad i = 1, \ldots, m, \\ & \mathbf{x} \ge 0. \end{cases} \tag{9}$$

In [46,47], Zimmermann has applied linear membership functions (4), (6) in problem (9) thus getting an ordinary LP problem. He also proposed to use the product operator instead of minimum, however, then (8) becomes nonlinear even if linear membership functions are used. A comprehensive review of various propositions for modeling the functions $\mu_D(\mathbf{x})$ can be found in [39,48].

Knowing the membership functions $\mu_l(\mathbf{x})(l = 1,\ldots, k)$ for fuzzy goals, one can define a Pareto optimal solution in the space of membership values, calledan M-Pareto optimal solution [32]. Some other refinements of the Zimmermann's approach have been proposed in [1,11].

Definition 1 A solution $\mathbf{x}^*$ is said to be *M-Pareto optimal* if and only if there does not exist another $\mathbf{x} \in X$ such that $\mu_l(\mathbf{x}) \ge \mu_l(\mathbf{x}^*), l = 1, \ldots, k$, with strict inequality holding for at least one l.

The concept of M-Pareto optimal solutions was at the origin of several *interactive methods* proposed for flexible programming (see [30,32]). In these methods, the decision maker determines membership functions for fuzzy goals and then specifies reference levels for the membership functions, denoted by $\overline{\mu}_l$ $(l = 1, \ldots, k)$. Assuming some minimum levels for membershipfunctions of fuzzy constraints, denoted by t_i $(i = 1, \ldots, m)$, one gets the following optimization problem:

$$\begin{cases} \max\limits_l \{\overline{\mu}_l - \mu_l(\mathbf{x})\} & \to \min \\ \text{s.t.} & \mu_i(\mathbf{x}) \ge t_i, \ i = 1, \ldots, m, \\ & \mathbf{x} \ge 0, \end{cases}$$

which is equivalent to

$$\begin{cases} v & \to \min \\ \text{s.t.} & v \ge \overline{\mu}_l - \mu_l(\mathbf{x}), \quad l = 1, \ldots, k, \\ & \mu_i(\mathbf{x}) \ge t_i, \quad i = 1, \ldots, m, \\ & \mathbf{x} \ge 0. \end{cases} \tag{10}$$

Again, problem (10) becomes an ordinary LP problem when all membership functions are linear. This approach is interactive in the sense that the reference levels can be changed from one iteration to another, as well as the membership functionsof fuzzy goals.

MOLP with Fuzzy Coefficients

All fuzzy coefficients of the FMOLP problem are given in a convenient form of *L-R* fuzzy numbers [13]. An *L-R* (*flat*) *fuzzy number* $\widetilde{a} = (a^L, a^R, \alpha^L, \alpha^R)_{LR}$ is defined by the membership function:

$$\mu_{\tilde{a}}(r) = \begin{cases} L\left(\frac{a^L - r}{\alpha^L}\right) & \text{for } r \le a^L, \\ 1 & \text{for } a^L \le r \le a^R, \\ R\left(\frac{r - a^R}{\alpha^R}\right) & \text{for } r \ge a^R, \end{cases}$$

where L and R are symmetric bell-shaped reference functions which are strictly decreasing in [0, 1] and such that $L(0) = \mathrm{R}(0) = 1$, $L(1) = \mathrm{R}(1) = 0$; $[a^L, a^R]$ is an interval of the most possible values, and α^L and α^R are nonnegative left and right 'spreads' of $\widetilde{a}$, respectively.

Experience indicates that an expert can describe the precise form of a fuzzy number only rarely. Therefore, as a practical way of getting suitable membership functions of fuzzy coefficients, H. Rommelfanger [26] has proposed that the expert begins with the specification of some prominent membership levels α and associates them with special meanings. After that the expert is expected tospecify values which belong to the selected membership levels.

$\alpha = 1$: $\mu_{\tilde{a}}(r) = 1$ means that value r certainly belongs to the set of possible values;

$\alpha = \lambda$: $\mu_{\tilde{a}}(r) \ge \lambda$ means that the expert estimates that value r with $\mu_{\tilde{a}}(r) \ge \lambda$ has a good chance of belonging to the set of possible values;

$\alpha = \varepsilon$: $\mu_{\tilde{a}}(r) < \varepsilon$ means that value r with $\mu_{\tilde{a}}(r) < \varepsilon$ has only a very little chance of belonging to the set of possible values, i. e. the expert is willing to neglect the corresponding values of r with $\mu_{\tilde{a}}(r) < \varepsilon$.

For example, it is reasonable to assume that $\lambda = 0.6$, $\varepsilon = 0.1$.

For the sake of clarity, let us assume that the reference functions of all fuzzy coefficients are of two kinds only: L and R. It should be specified, moreover, that all

arithmetic operations on fuzzy numbers taking place in (1), (2) are extended operations in the sense of Zadeh's extension principle [45]:

$$f_{\tilde{a}*\tilde{b}}(r) = \sup_{r=y*z} T(f_{\tilde{a}}(y), f_{\tilde{b}}(z)), \quad r \in \mathbb{R}, \tag{11}$$

where $*$ is a real operation $*: \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ and T: [0, 1] $\times$ [0, 1] $\to$ [0, 1] is any given t-norm.

For any $\mathbf{x} \geq 0$, the left-hand side of the ith constraint and the value of the lth objective function can be summarized to the following fuzzy numbers:

$$\widetilde{\mathbf{a}}_i\mathbf{x} = \left(\mathbf{a}_i^L\mathbf{x}, \mathbf{a}_i^R\mathbf{x}, \alpha_i^L\mathbf{x}, \alpha_i^R\mathbf{x}\right)_{LR}, \quad i = 1, \ldots, m,$$
$$\widetilde{\mathbf{c}}_l\mathbf{x} = \left(\mathbf{c}_l^L\mathbf{x}, \mathbf{c}_l^R\mathbf{x}, \gamma_l^L\mathbf{x}, \gamma_l^R\mathbf{x}\right)_{LR}, \quad l = 1, \ldots, k.$$

In the literature the min t-norm is generally applied. Then,

$$\mathbf{a}_i^L\mathbf{x} = \sum_{j=1}^{n} a_{ij}^L x_j, \quad \mathbf{c}_l^L\mathbf{x} = \sum_{j=1}^{n} c_{lj}^L x_j, \tag{12}$$

$$\mathbf{a}_i^R\mathbf{x} = \sum_{j=1}^{n} a_{ij}^R x_j, \quad \mathbf{c}_l^R\mathbf{x} = \sum_{j=1}^{n} c_{lj}^R x_j, \tag{13}$$

$$\alpha_i^L\mathbf{x} = \sum_{j=1}^{n} \alpha_{ij}^L x_j, \quad \gamma_l^L\mathbf{x} = \sum_{j=1}^{n} \gamma_{lj}^L x_j, \tag{14}$$

$$\alpha_i^R\mathbf{x} = \sum_{j=1}^{n} \alpha_{ij}^R x_j, \quad \gamma_l^R\mathbf{x} = \sum_{j=1}^{n} \gamma_{lj}^R x_j. \tag{15}$$

Obviously, the spreads of these fuzzy numbers extend when number and values of variables increase. The simple addition of the spreads of fuzzy coefficients corresponds to the assumption that their uncertainty comes fromindependent sources. This is not realistic in many practical situations. For getting a more realistic extended addition of the left-hand sides of fuzzy constraints and of fuzzy objectives, Rommelfanger and T. Keresztfalvi [29] recommend the use of Yager's parameterized t-norm:

$$T_p(t_1, \ldots, t_s) = \max\left\{0, 1 - \left(\sum_{i=1}^{s}(1 - t_i)^p\right)^{1/p}\right\},$$
$$t_1, \ldots, t_s \in [0, 1], \quad p > 0. \tag{16}$$

Then, $\mathbf{a}_i^L\mathbf{x}, \mathbf{a}_i^R\mathbf{x}, \mathbf{c}_l^L\mathbf{x}, \mathbf{c}_l^R\mathbf{x}$ are calculated according to (12) and (13), however, the spreads $\alpha_i^L\,\mathbf{x}, \alpha_i^R\mathbf{x}, \gamma_l^L\mathbf{x}, \gamma_l^R\,\mathbf{x}$ are calculated according to a new, less cumulative formula:

$$\alpha_i^L\mathbf{x} = \left(\sum_{j=1}^{n}\left(\alpha_{ij}^L x_j\right)^q\right)^{1/q},$$

$$\alpha_i^R\mathbf{x} = \left(\sum_{j=1}^{n}\left(\alpha_{ij}^R x_j\right)^q\right)^{1/q},$$

$$\gamma_i^L\mathbf{x} = \left(\sum_{j=1}^{n}\left(\gamma_{ij}^L x_j\right)^q\right)^{1/q},$$

$$\gamma_i^R\mathbf{x} = \left(\sum_{j=1}^{n}\left(\gamma_{ij}^R x_j\right)^q\right)^{1/q},$$

where $q = p/(p-1) \geq 1$.

Coming back to MOLP problem with fuzzy coefficients, we haveto answer the question how to interpret the relation between fuzzy left- and right-hand side of the constraints. If constraints (2) were transformed to equality constraints (by addition of slack variables on the left) thenthe equality relation could be interpreted in terms of weak inclusion of fuzzy sets [12,21]:

$$\widetilde{\mathbf{a}}_i\mathbf{x} \subseteq \widetilde{b}_i, \quad i = 1, \ldots, m. \tag{17}$$

It says that the region of possible values of the left-hand side should be contained in the tolerance region of the right-hand side. The LP problem with constraints (17) is called *robust programming problem*.

Each constraint (18) is then reduced to four deterministic constraints:

$$\begin{aligned} &\mathbf{a}_i^L\mathbf{x} \geq b_i^L, \quad \mathbf{a}_i^R\mathbf{x} \leq b_i^R, \\ &\mathbf{a}_i^L\mathbf{x} - \alpha_i^L\mathbf{x} \geq b_i^L - \beta_i^L, \\ &\mathbf{a}_i^R\mathbf{x} + \alpha_i^L\mathbf{x} \leq b_i^R + \beta_i^R, \\ &\text{for } i = 1, \ldots, m, \end{aligned} \tag{18}$$

where $\widetilde{b}_i = (b_i^L, b_i^R, \beta_i^L, \beta_i^R)_{LL}$ or $\widetilde{b}_i = (b_i^L, b_i^R, \beta_i^L, \beta_i^R)_{RR}$, $i = 1, \ldots, m$.

In order to transform fuzzy objectives into deterministic equivalents, one can consider a 'middle' value of $\widetilde{\mathbf{c}}_l\mathbf{x}$ at some level $\xi \in [0, 1]$, $l = 1, \ldots, k$. The 'middle' can be understood [8] as a weighted combination of the most possible values $\mathbf{c}_l^L\mathbf{x}$ and $\mathbf{c}_l^R\mathbf{x}$, and of the smallest and the greatest (extreme) values at possibility level ξ. Thus, the objectives (1) become:

$$[z_1(\mathbf{x}), \ldots, z_k(\mathbf{x})] \to \min, \tag{19}$$

where $z_l(\mathbf{x}) = w_1\mathbf{c}_l^L - w_2\gamma_l^L\mathbf{x}L^{-1}(\xi) + w_3\mathbf{c}_l^R\mathbf{x} + w_4\gamma_l^R\mathbf{x}\mathbf{R}^{-1}(\xi)$, $l = 1, \ldots, k$; w_1, w_2, w_3, w_4 are nonnegative weights, e. g. $w_1 = w_3 = 0.3$, $w_2 = w_4 = 0.2$. The deterministicobjectives (19) are linear even if reference functions L and R are nonlinear.

There exist approaches proposing a substitution of each objective by several deterministic objectives corresponding to extreme values of several ξ-level sets [9,28].

Finally, let us mention a comparison technique of fuzzy numbers, which is based on the compensation of area determined by the membership functions of two fuzzy numbers being compared. This technique, which has been characterized in [17] and [5], and then in [31] and [15], can be used directly to transform the comparison of fuzzy left- and right-hand side of the constraints, and of the fuzzy objectives and fuzzy goals into nonparametric deterministic equivalents. Although this technique seems intuitive, it has a convincing theoretical foundation.

Indeed, the semantics of fuzzy numbers considered in the MOLP problem with fuzzy coefficients is related to the representation of incomplete or vague states of information under the form of possibility distributions. This view of fuzzy numbers is concordant with the Dempster interpretation of fuzzy numbers as imprecise probability distributions [10]. In this perspective, the comparison of two fuzzy numbers can be substituted by the comparison of their mean values defined consistently with the well-known definition of expectation in probability theory. The idea exploited in [14] relies on the mathematical fact that, with respect to a fuzzy number, the possibility measure corresponds to an upper probability distribution, while the necessity measure, to a lower probability distribution of the corresponding random variable. Then it is reasonable to define the mean value of a fuzzy number as a closed interval whose bounds are expectations of upper and lower probability distributions. The comparison of two fuzzy numbers boils down to the comparison if arithmetic means of these bounds, which is computationally equivalent to the above mentioned technique based on area compensation, as shown in [15].

In consequence of application of all these comparison techniques, the MOLP problem with fuzzy coefficients is transformed to an associate deterministic MOLP problem, as (19), (18), (3) above, which should, preferably, be solved by one of existing interactive procedures (see, e. g., [43]).

Flexible MOLP with Fuzzy Coefficients

This problem combines the two semantics of fuzzy sets considered separately in flexible programming and in MOLP with fuzzy coefficients. This means that in addition to fuzzy coefficients in the objective functions and on the both sides of the constraints, the degree of satisfaction of fuzzy constraints and fuzzy goals is considered in fuzzy set terms.

A crucial question which has to be answered while solving a flexible MOLP problem with fuzzy coefficients is how to express the minimal conditions on the satisfaction of fuzzy constraints in deterministic terms.

In most of existing approaches, the minimal conditions on the satisfaction of fuzzy constraints (2) are expressed by one or two deterministic linear constraints which substitute the original fuzzy constraints. To give an idea of these crisp surrogates, let us present them in common terms from the most pessimistic to the most optimistic attitude. We assume the following form of the fuzzy left- and right-hand side of the ith constraint:

$$\widetilde{\mathbf{a}}_i\mathbf{x} = (\mathbf{a}_i^L\mathbf{x}, \mathbf{a}_i^R\mathbf{x}, \alpha_i^L\mathbf{x}, \alpha_i^R\mathbf{x})_{LR},$$
$$\widetilde{b}_i = (b_i, 0, \beta_i)_{LR},$$

a) (see [3,40])

$$\mathbf{a}_i^R\mathbf{x} + \alpha_i^R\mathbf{x}R^{-1}(\rho) \le b_i, \quad \rho \in [0,1];$$

b) (see [22,25,44])

$$\begin{cases} \mathbf{a}_i^R\mathbf{x} \le b_i \\ \mathbf{a}_i^R\mathbf{x} + \alpha_i^R\mathbf{x}R^{-1}(\varepsilon) \le b_i + \beta_i R^{-1}(\varepsilon), \\ \quad \varepsilon \in [0,1]; \end{cases}$$

c) (see [4])

$$\mathbf{a}_i^R\mathbf{x} + \alpha_i^R\mathbf{x}R^{-1}(\sigma) \le b_i + \beta_i R^{-1}(\sigma),$$
$$\sigma \in [0,1];$$

d) (see [8,34,35])

$$\begin{cases} \mathbf{a}_i^L\mathbf{x} - b_i \le \alpha_i^L\mathbf{x}L^{-1}(\tau) + \beta_i R^{-1}(\tau), \\ \tau \in [0,1], \text{ optimistic}, \\ \mathbf{a}_i^R\mathbf{x} + \alpha_i^R\mathbf{x}R^{-1}(\eta) \le b_i + \beta_i R^{-1}(\eta), \\ \eta \in [0,1], \text{ pessimistic}; \end{cases}$$

e) (see [23])

$$\begin{cases} \mathbf{a}_i^R \mathbf{x} \le b_i + \delta\beta_i, \\ \delta + \varepsilon \in [0,1],\ \delta \ge 0,\ \varepsilon \ge 0, \\ \mathbf{a}_i^R \mathbf{x} + (1-\varepsilon-\delta)\alpha_i^R \mathbf{x} \le b_i + (1-\varepsilon)\beta_i; \end{cases}$$

f) (see [33,19])

$$\mathbf{a}_i^L \mathbf{x} - \alpha_i^L \mathbf{x} L^{-1}(\alpha) \le b_i + \beta_i R^{-1}(\alpha),$$
$$\alpha \in [0,1].$$

In all these approaches, the parameters α, δ, ε, η, τ, ρ, σ can be used by the decision maker to control the degree of satisfaction of fuzzy constraints in an interactive way.

Figure 1 shows results of conditions a)–f) applied on a common fuzzy constraint. Although it is the case in Fig. 1, the reference functions L and R need not be linear in the above conditions.

Another interpretation of fuzzy constraints has been given in [24]. The ith fuzzy constraint is replaced by the pessimistic condition proposed in [34] and by a new objective:

$$\mathbf{a}_i^R \mathbf{x} + \alpha_i^{R\varepsilon} \mathbf{x} \le b_i + \beta_i^{\varepsilon}, \tag{20}$$

$$\mu_i(\mathbf{x}) \to \max, \tag{21}$$

where membership function $\mu_i(\mathbf{x})$ is defined according to (4). More detailed discussion of the interpretation of fuzzy constraints can be found in [30].

If fuzzy goals are specified as L-R fuzzy numbers $\widetilde{g}_l = (g_l, 0, \nu_l)_{LL}$ $(l = 1, \ldots, k)$, then the satisfying conditions

$$\widetilde{\mathbf{c}}_l \mathbf{x} \widetilde{\le} \widetilde{g}_l, \quad l = 1, \ldots, k, \tag{22}$$

can be treated as additional fuzzy constraints. In accordance to the chosen interpretation of the fuzzy inequality relation, (22) can be substituted by one or two crisp inequalities listed above or by (20) and (21). Another proposal has been made by R. Slowinski in [34,35]; the degree of satisfaction of fuzzy goals is represented there by the levels of intersection of left reference functions of $\widetilde{\mathbf{c}}_l \mathbf{x}$ with right reference functions of g_l $(l = 1, \ldots, k)$:

$$L\left(\frac{\mathbf{c}_l^L \mathbf{x} - g_l}{\gamma_l^L \mathbf{x} + \nu_l}\right) \to \max, \quad l = 1, \ldots, k. \tag{23}$$

These crisp objectives substitute the fuzzy ones. In the case of linear reference functions L, functions (23) become linear fractional:

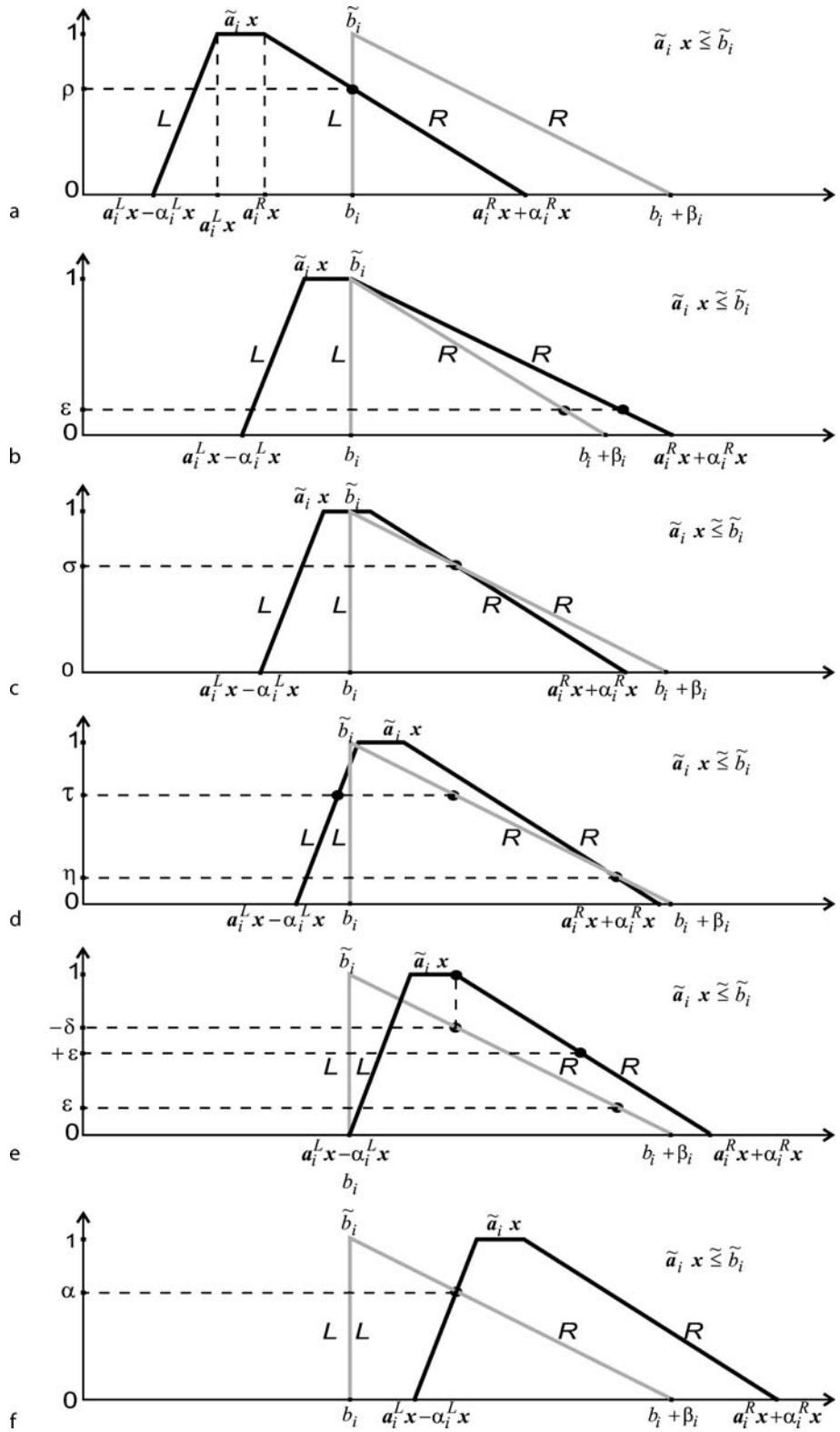
$$\frac{\mathbf{c}_l^L \mathbf{x} - g_l}{\gamma_l^L \mathbf{x} + \nu_l} \to \min, \quad l = 1, \ldots, k. \tag{24}$$

The crisp objectives (24) and the optimistic and pessimistic conditions d) on the satisfaction of fuzzy constraints have been used in the FLIP method presented in [8,34,35,39]. They constitute an associate deterministic multi-objective *linear-fractional programming* (MOLFP) problem. In FLIP, the MOLFP problem is solved using an *interactive sampling procedure*. In each calculation step of this procedure, a sample of nondominated points (*Pareto optimal solutions*) of the MOLFP problem is generated and then shown to the decision maker who is asked to select the one that fits best his/her preferences. If the selected point is not the final compromise, it becomes a central point of a nondominated region that is sampled in the next calculation step. In this way, the sampled part of the nondominated set is successively reduced (focusing phenomenon) until the most satisfactory efficient point (compromise solution) is reached. An important advantage of the method presented above is that the only optimization procedure to be used is a linear programming one. Moreover, it has a simple scheme and allows retractions to the points abandoned in previous iterations.

The interaction with the decision maker takes place at two levels: first when fixing the safety parameters and then in the course of the guided generation and evaluation of the nondominated points of the MOLFP problem.

Let us precise that the fuzzy goals g_l $(l = 1, \ldots, k)$ do not influence the set of nondominated points of the MOLFP problem; they rather play the role of a visual reference than that of a preferential information influencing the set of generated proposals for the compromise solution.

An important feature of any software implementing a fuzzy multi-objective programming method is the presentation of candidate solutions in the interactive process. In the FLIP software, the Pareto optimal solutions of the MOLFP problem are shown not only numerically but also graphically, in terms of mutual positions of fuzzy numbers corresponding to original objectives and aspiration levels on the one hand, and to



Fuzzy Multi-objective Linear Programming, Figure 1
Results of conditions a)–f) applied on a common fuzzy constraint

left- and right-hand side of original constraints on the other hand [6]. In this way, the decision maker gets quite a complete idea of the quality of each proposed solution.

The quality is evaluated taking into account the following characteristics:

- scores of fuzzy objectives in relation to the goals;
- dispersion of values of the fuzzy objectives due to uncertainty;
- safety of the solution or, using a complementary term, the risk of violation of the constraints.

So, the definition of the best compromise involves not only the scores on particular objectives but also the safety of the corresponding solution. It is possible due to *visual interaction* that needs graphical display of objectives and constraints for any analyzed solution. The comparison of fuzzy left- and right-hand side of the constraints, as well as evaluation of dispersion of the values of objectives, is practically infeasible on the basis of numerals only. The graphical presentation of proposed solutions is not only a 'user friendly' interface but the best way for a complete characterization of these solutions.

There exists an implementation of FLIP in Visual Basic in the MS-Excel environment; it allows a user to define all safety parameters and the parameter p of the Yager's formula (16) for the aggregation of fuzzy objectives and of fuzzy left-hand sides of fuzzy constraints. The candidates for the best compromise solution are displayed there both numerically and graphically.

Conclusions

Fuzzy multi-objective linear programming methods have often been proposed in view of specific applications (see, e. g., [6,18,30,34,39,44]). This means that the many proposals described in this article are based on different assumptions that are verified in different practical situations. The choice of a procedure for an actual decision problem should take into account these assumptions. In any case, the interactive process should enable the best use of the decision maker's knowledge of the problem. Fuzzy multi-objective linear programming can also be seen as a tool for an interactive *robustness analysis* of MOLP problems. It gives an insight into sensitivity of proposed solutions on changes of particular coefficients within some intervals and on changes of preferences as to degrees of satisfaction of the constraints.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Behringer FA (1981) A simplex-based algorithm for the lexicographically extended linear maxmin problem. Europ J Oper Res 7:274–283
2. Bellman RE, Zadeh LA (1970) Decision making in a fuzzy environment. Managem Sci 17(4):141–164
3. Buckley JJ (1988) Possibilistic linear programming with triangular fuzzy numbers. Fuzzy Sets and Systems 26:135–138
4. Carlsson C, Korhonen P (1986) A parametric approach to fuzzy linear programming. Fuzzy Sets and Systems 20:17–30
5. Chanas S (1987) Fuzzy optimization in networks. In: Kacprzyk J, Orlowski SA (eds) Optimization Models Using Fuzzy Sets and Possibility Theory. Reidel, London, pp 303–327
6. Czyzak P (1990) Application of "FLIP" method to farm structure optimization under uncertainty. In: Slowinski R,

Teghem J (eds) Stochastic versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty. Kluwer, Dordrecht, pp 263–278
7. Czyzak P, Slowinski R (1991) "FLIP"-multiobjective fuzzy linear programming software with graphical facilities. In: Fedrizzi M, Kacprzyk J, Roubens M (eds) Interactive Fuzzy Optimization. Springer, Berlin, pp 168–187
8. Czyzak P, Slowinski R (1993) A visual interactive method for MOLP problems with fuzzy coefficients. In: Lowen R, Roubens M (eds) Fuzzy Logic-State of the Art. Kluwer, Dordrecht, pp 321–332
9. Delgado M, Verdegay JL, Vila MA (1989) A general model for fuzzy linear programming. Fuzzy Sets and Systems 29:21–30
10. Dempster AP (1967) Upper and lower probabilities induced by a multivalued mapping. Ann Math Statist 38:325–339
11. Dubois D, Fortemps Ph (1999) Computing improved optimal solutions to max-min flexible constraint satisfaction problems. Europ J Oper Res 118:95–126
12. Dubois D, Prade H (1980) Systems of linear fuzzy constraints. Fuzzy Sets and Systems 3:37–48
13. Dubois D, Prade H (1987) Fuzzy numbers-an overview. In: Bezdek JC (ed) Analysis of Fuzzy Information: vol 1, Mathematics and Logic. CRC Press, Boca Raton, FL, 3–39
14. Dubois D, Prade H (1987) The mean value of a fuzzy number. Fuzzy Sets and Systems 24:279–300
15. Fortemps Ph, Roubens M (1996) Ranking and defuzzification methods based on area compensation. Fuzzy Sets and Systems 82:319–330
16. Inuiguchi M, Ichihashi H, Tanaka H (1990) Fuzzy programming: A survey of recent developments. In: Slowinski R, Teghem J (eds) Stochastic versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty. Kluwer, Dordrecht, pp 45–68
17. Kolodziejczyk W (1986) Orlovsky's concept of decision-making with fuzzy preference relation-further results. Fuzzy Sets and Systems 19:11–20
18. Lai Y-J, Hwang C-L (1994) Fuzzy multiple objective decision making: Methods and applications. Springer, Berlin
19. Luhandjula MK (1987) Multiple objective programming with possibilistic constraints. Fuzzy Sets and Systems 21:135–146
20. Luhandjula MK (1989) Fuzzy optimization: An appraisal. Fuzzy Sets and Systems 30:257–282
21. Negoita CV (1981) The current interest in fuzzy optimization. Fuzzy Sets and Systems 6:261–269
22. Ramik J, Rimanek J (1985) Inequality between fuzzy numbers and its use in fuzzy optimization. Fuzzy Sets and Systems 16:123–138
23. Ramik J, Rommelfanger H (1993) A single-and a multivalued order on fuzzy numbers and its use in linear programming with fuzzy coefficients. Fuzzy Sets and Systems 57:203–208
24. Rommelfanger H (1988, 1994) Entscheiden bei Unschaerfe-Fuzzy Decision Support Systeme. Springer, Berlin
25. Rommelfanger H (1989) Inequality relations in fuzzy constraints and their use in linear fuzzyoptimization. In: Verdegay JL, Delgado M (eds) The Interface between Artificial Intelligence and Operational Research in Fuzzy Environment. TÜV Rheinland, Köln, 195–211
26. Rommelfanger H (1990) "FULPAL"-an interactive method for solving (multi-objective) fuzzy linear programming problems. In: Slowinski R, Teghem J (eds) Stochastic versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty. Kluwer, Dordrecht, pp 279–299
27. Rommelfanger H (1996) Fuzzy linear programming and applications. Europ J Oper Res 92:512–527
28. Rommelfanger H, Hanuscheck R, Wolf J (1989) Linear programming with fuzzy objectives. Fuzzy Sets and Systems 29:31–48
29. Rommelfanger H, Keresztfalvi T (1991) Multi-criteria fuzzy optimization based on Yager's parametrized t-norm. Found Computing and Decision Sci 16:99–110
30. Rommelfanger H, Slowinski R (1998) Fuzzy linear programming with single or multiple objective functions. In: Slowinski R (ed) Fuzzy Sets in Decision Analysis: Operations Research and Statistics. Kluwer, Dordrecht, pp 179–213
31. Roubens M (1990) Inequality constraints between fuzzy numbers and their use in mathematical programming. In: Slowinski R, Teghem J (eds) Stochastic versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty. Kluwer, Dordrecht, pp 321–330
32. Sakawa M (1993) Fuzzy sets and interactive multiobjective optimization. Plenum, New York
33. Sakawa M, Yano H (1989) Interactive fuzzy satisficing method for multiobjective nonlinear programming problems with fuzzy parameters. Fuzzy Sets and Systems 30:221–238
34. Slowinski R (1986) A multicriteria fuzzy linear programming method for water supply system development planning. Fuzzy Sets and Systems 19:217–237
35. Slowinski R (1990) "FLIP"-an interactive method for multiobjective linear programming with fuzzy coefficients. In: Slowinski R, Teghem J (eds) Stochastic versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty. Kluwer, Dordrecht, pp 249–262
36. Slowinski R (1997) Fuzzy multi-objective linear programming. Belgian J Oper Res Statist Comput Sci 37:83–98
37. Slowinski R (1997) Interactive fuzzy multiobjective programming. In: Climaco J (ed) Multicriteria analysis. Springer, Berlin, 202–212
38. Slowinski R, Teghem J (1988) Fuzzy versus stochastic approaches to multicriteria linear programming under uncertainty. Naval Res Logist 35:673–695. Reprinted in: Dubois D, Prade H, Yager R (eds) (1993) Readings in Fuzzy Sets

for Intelligent Systems. Morgan Kaufmann, San Mateo, CA, 810–821
39. Slowinski R, Teghem J (1990) Stochastic versus fuzzy approaches to multiobjective mathematical programming under uncertainty. Kluwer, Dordrecht
40. Tanaka H, Asai K (1984) Fuzzy linear programming with fuzzy numbers. Fuzzy Sets and Systems 13:1–10
41. Tanaka H, Okuda T, Asai K (1974) On fuzzy-mathematical programming. J Cybernetics 3(4):37–46
42. Vanderpooten D, Vincke Ph (1989) Description and analysis of some representative interactive multicriteria procedures. Math Comput Modelling, 12:1221–1238
43. Verdegay JL (1984) Application of fuzzy optimization in operational research. Control and Cybernetics 13:229–239
44. Wolf J (1988) Lineare Fuzzy Modelle zur Unterstuetzung der Investitionsentscheidung. Lang-Verlag, Frankfurt am Main
45. Zadeh LA (1965) Fuzzy sets. Inform and Control 8:338–353
46. Zimmermann HJ (1976) Description and optimization of fuzzy systems. Internat J General Syst 2:209–216
47. Zimmermann HJ (1978) Fuzzy programming and linear programming with several objective functions. Fuzzy Sets and Systems 1:45–55
48. Zimmermann HJ (1985) Applications of fuzzy sets theory to mathematical programming. Inform Sci 36:29–58

G

Gasoline Blending and Distribution Scheduling: An MILP Model

ZHENYA JIA, MARIANTHI IERAPETRITOU
Department of Chemical and Biochemical Engineering, Rutgers University, Piscataway, USA

MSC2000: 90B35, 93A30

Article Outline

Synonyms
- Indices
- Sets
- Parameters
- Variables

Introduction
Definition
Formulation
- Material Balance Constraints for Product-Stock Tank *j*
- Capacity Constraints
- Allocation Constraints
- Demand Constraints
- Sequence Constraints
- Duration Constraints

Blending Stage Consideration
- Material Balance Constraints for the Blender
- Material Balance Constraints for Component Tank *l*
- Allocation Constraints for Product-Stock Tank *j*
- Allocation Constraints for Blender
- Sequence Constraints
- Duration Constraints
- Objective Function

Case
Conclusions
References

Synonyms

Indices

i = orders
j = product-stock tanks
s = products
k = components
l = component tanks
n = event points

Sets

I = orders
I_j = orders which can be performed in product-stock tank j
I_s = orders which order product s
J = product-stock tanks
J_i = product-stock tanks which are suitable for performing order i
J_s = product-stock tanks which can store product s
N = event points within the time horizon
S = products
S_j = products which can be stored in product-stock tank j
K = components
K_l = components which can be stored in component-stock tank l
L = component stock tanks
L_k = component-stock tanks which can store component k

Parameters

$Vmax(j)$ = maximum capacity of product-stock tank j
$Vmin(j)$ = minimum amount of product stored in tank j if tank j is utilized

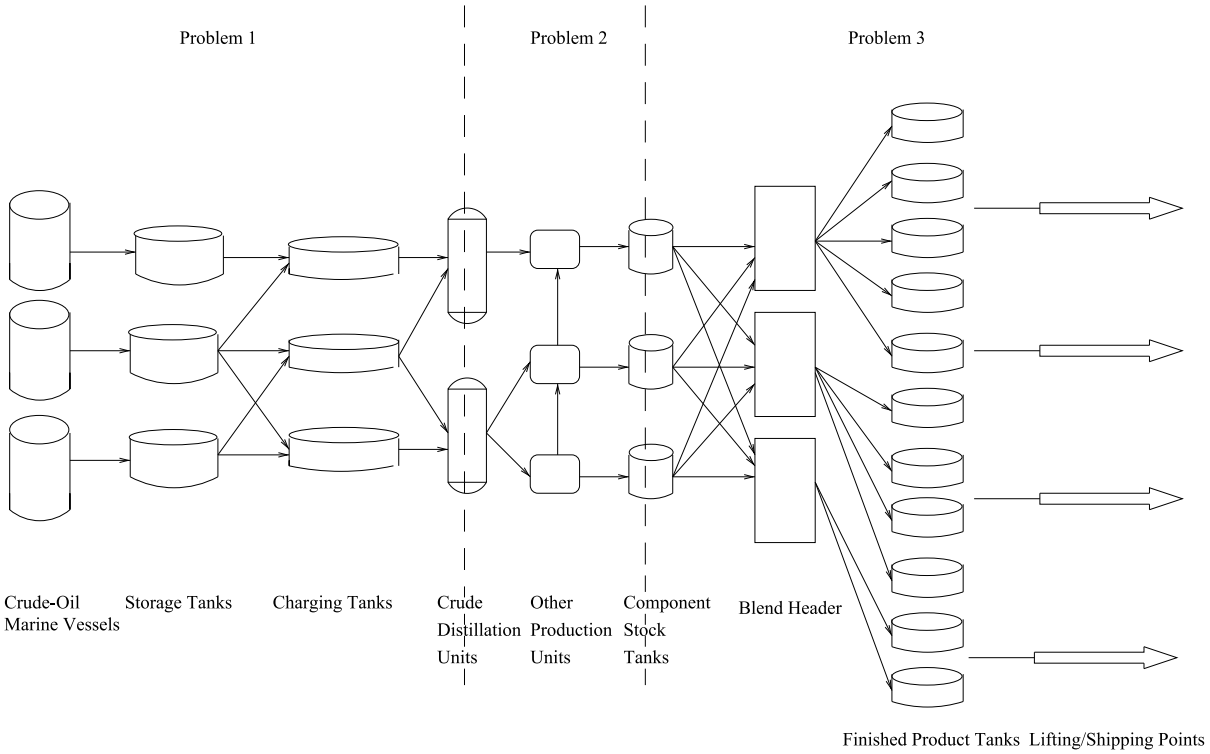
Vinitial(j,s) = amount of product *s* stored in tank *j* initially
Vin(l,k) = amount of component *k* stored in component tank *l* initially
Vcomp(l) = maximum capacity of component tank *l*
Recipe(s,k) = the proportion of component *k* to in product *s*
l(i) = lifting rate of order *i*
Bflow = flow rate of product being produced and transferred to product-stock tanks
Prod_srt(i) = time by which order *i* can start
Prod_end(i) = time by which order *i* is due
U1 = lower bound on the amount of product lifted
U2 = upper bound on the amount of product lifted
U3 = upper bound of a small-sized order
U4 = upper bound of a medium-sized order
U5 = lower bound of a large-sized order
flowmin = minimum flow rate of component tanks
flowmax = maximum flow rate of component tanks
H = time horizon

Variables

uv(i,j,n) = binary variables that assign the beginning of order *i* in tank *j* at event point *n*
y(s,j,n) = binary variables that assign product *s* being stored in tank *j* at event point *n*
sv(s,j,n) = binary variables that assign product *s* being produced and transferred to tank *j* at event point *n*
xv(s,n) = 0-1 continuous variables that assign product *s* being produced at event point *n*
yv(k,l,n) = binary variables that assign component *k* being extracted from component-stock tank *l* at event point *n*
Ts(i,j,n) = starting time of order *i* in tank *j* at event point *n*
Te(i,j,n) = finishing time of order *i* in tank *j* while it starts at event point *n*
lift(i,j,n) = amount of product being lifted for order *i* from tank *j* at event point *n*
Pst(s,j,n) = amount of product *s* in tank *j* at event point *n* before new product is transferred from the blender
Tbs(s,j,n) = starting time of product *s* being produced and transferred to product-stock tank *j* at event point *n*
Tbf(s,j,n) = finishing time of product *s* being produced and transferred to product-stock tank *j* at event point *n*
Blnd(s,j,n) = amount of product *s* being transferred from blender to tank *j* at event point *n*
comp(k,l,n) = amount of component *k* being transferred to the blender at event point *n*
bc(k,l,n) = amount of component *k* in component tank *l* at event point *n*
cracking(k,l,n) = amount of component *k* being transferred from separation units to component tank *l* at event point *n*

Introduction

Gasoline blending is a crucial step in refinery operation as gasoline can yield 60–70% of a refinery's profit. The process involves mixing various stocks, which are the intermediate products from the refinery, along with some additives, such as antioxidants and corrosion inhibitors, to produce blends with certain qualities [1]. In the past few decades, a substantial amount of work has been dedicated to process operations [3,4,7,8,9]. A variety of support systems have been developed to address planning and scheduling of blending operations. StarBlend [13], for example, which is developed by Texaco, uses a multiperiod blending model written in GAMS that facilitates the incorporation of future requirements into current blending decisions. Glismann and Gruhn [5,6] proposed a mixed-integer linear model (MILP), which is based on a resource-task network representation, to solve the task of short-term scheduling of blending processes. The recipe optimization problem is then formulated as a nonlinear program and the results are returned to the scheduling problem, so that an overall optimization can be achieved. A fuzzy linear formulation was applied to the blending facilities by Djukanovic et al. [2], in order to address the problem of uncertainty of input information within the fuel scheduling optimization. Singh et al. [14] addressed the



Gasoline Blending and Distribution Scheduling: An MILP Model, Figure 1
Graphic overview of the gasoline blending and distribution system

problem of blending optimization for in-line blending for the case of stochastic disturbances in feedstock qualities. They presented a real-time optimization method that can provide significantly improved profitability.

The objective of this work is to propose a new mathematical model that addresses the simultaneous optimization of the short-term scheduling problem of gasoline blending and distribution as described in the following section.

Definition

The overall oil-refinery system is decomposed into three parts as depicted in Fig. 1. The first part (problem 1, Fig. 1) involves the crude-oil unloading, mixing and inventory control (Jia et al. [10]), the second part (problem 2, Fig. 1) consists of the production unit scheduling, which includes both fractionation and reaction processes, and the third part (problem 3, Fig. 1), which is addressed in this work, depicts the finished product blending and shipping end of the refinery. The gasoline blending system consists of four pieces of equipment all linked together through various piping segments, flow meters and valves. They are component-stock tanks, blend header, product-stock tanks and lifting ports. Components from the component-stock tanks are fed to the blend header according to the recipes. Thus, different products can be produced and then stored in their suitable product-stock tanks. The final step is to lift those products during the specified time periods in order to satisfy all the orders. The objective is to determine the following variables: (1) starting and finishing time of orders taking place in each product-stock tank; (2) the amount and type of product being lifted for each order from tanks; (3) starting and finishing times of the product being transferred from the blender to the tanks; (4) the amount and type of component being transferred from component tanks to the blender, so as to process all the orders in specific time periods.

The scheduling problem as described above is modeled in the next section following a continuous-time representation. It gives rise to an MILP formulation that can be efficiently solved using commercially available solvers.

Formulation

It is assumed that perfect mixing is achieved at the blend header and that the changeover time between different products in the storage tanks is negligible.

Material Balance Constraints for Product-Stock Tank *j*

Constraint (1a) expresses that the amount of product *s* in tank *j* at event point *n+1* (*Pst(s,j,n+1)*) is equal to that at event point n adjusted by any amounts transferred from the blender (*Blnd(s,j,n)*) or lifted at event point *n* ($\sum_{i\in I_s} lift(i,j,n)$). Constraint (1b) states that the amount of product *s* being lifted from tank *j* at the last event point *N* should not exceed the amount of product *s* stored in tank *j*.

$$Pst(s,j,n+1) = Pst(s,j,n) + Blnd(s,j,n) - \sum_{i\in I_s} lift(i,j,n), \quad \forall s\in S, j\in J_s, n\in N, n\neq N \tag{1a}$$

$$Pst(s,j,n) + Blnd(s,j,n) \geq \sum_{i\in I_s} lift(i,j,n)\,, \quad \forall s\in S, j\in J_s, n=N \tag{1b}$$

Capacity Constraints

Constraint (2) imposes a volume capacity limitation of product *s* in tank *j* at event point *n*.

$$Vmin(j) * y(s,j,n) \leq Pst(s,j,n) + Blnd(s,j,n) \leq Vmax(j) * y(s,j,n)\,, \quad \forall s\in S, j\in J_s, n\in N \tag{2}$$

Allocation Constraints

According to constraint (3a), *uv(i,j,n)* is equal to 1 if the amount of product being lifted from tank *j* for order *i* is not zero at event point *n*, that is, $lift(i,j,n)\neq 0$; *uv(i,j,n)* equals 0 otherwise. U1 and U2 correspond to lower and upper bounds on the amount of product lifted, respectively, and are chosen according to the smallest order and the maximum capacities of the tanks.

$$U1 * uv(i,j,n) \leq lift(i,j,n) \leq U2 * uv(i,j,n)\,, \quad \forall i\in I, j\in J_i, n\in N \tag{3a}$$

To avoid task splitting, constraints (3b)–(3d) state that order *i* should be processed only once if it is a small order and at most twice if it is a medium-sized order. Otherwise, it can be processed at most three times. For different problems, U3 and U4 are chosen accordingly to define small and medium-sized orders. Constraint (3e) expresses that for large orders which are defined as greater than or equal to U5, the minimum order splitting is 25 Mbbl.

$$\sum_n \sum_{j\in J_i} uv(i,j,n) = 1\,, \quad \forall \sum_s Prod_ord(i,s) \leq U3,\ i\in I,\ n\in N \tag{3b}$$

$$\sum_n \sum_{j\in J_i} uv(i,j,n) \leq 2\,, \quad \forall \sum_s Prod_ord(i,s) \leq U4,\ i\in I,\ n\in N \tag{3c}$$

$$\sum_n \sum_{j\in J_i} uv(i,j,n) \leq 3\,, \forall i\in I,\ n\in N \tag{3d}$$

$$25 * uv(i,j,n) \leq lift(i,j,n)\,, \quad \forall \sum_s Prod_ord(i,s) \geq U5, i\in I, j\in J_i, n\in N \tag{3e}$$

Constraint (4) forces *sv(s,j,n)* to be equal to 1 when Blnd(s,j,n) is not zero; otherwise sv(s,j,n) equals 0.

$$Vmin(j) * sv(s,j,n) \leq Blnd(s,j,n) \leq Vmax(j) * sv(s,j,n)\,, \quad \forall s\in S, j\in J_s, n\in N \tag{4}$$

Demand Constraints

Constraints (5a) and (5b) state that order *i* can be processed at most once in one tank during the time horizon under consideration and that the amount of product being lifted from all the product-stock tanks should be equal to the amount ordered ($\sum_s Prod_ord(i,s)$).

$$\sum_n uv(i,j,n) \leq 1\,, \quad \forall i\in I, j\in J_i, n\in N \tag{5a}$$

$$\sum_n \sum_{j\in J_i} lift(i,j,n) = \sum_s Prod_ord(i,s)\,, \quad \forall s\in S, i\in I, n\in N \tag{5b}$$

Sequence Constraints

Constraints (6a)–(6c) state that order i starting in tank j at event point $n+1$ should start after the finishing time of the same order processed in the same tank which has started at event point n. Constraints (6d) and (6e) express that order i should start and finish during the specific time period based on the order requirement. These constraints are relaxed if $uv(i,j,n)$ is zero, which means order i is not executed in tank j at event point n.

$$Ts(i,j,n+1) \geq Te(i,j,n) - H * (1 - uv(i,j,n)) , \quad \forall\, i \in I_j, j \in J, n \in N, n \neq N \quad (6a)$$

$$Ts(i,j,n+1) \geq Ts(i,j,n) , \quad \forall i \in I_j, j \in J, n \in N, n \neq N \quad (6b)$$

$$Te(i,j,n+1) \geq Te(i,j,n) , \quad \forall i \in I_j, j \in J, n \in N, n \neq N \quad (6c)$$

$$Ts(i,j,n) \geq Prod_srt(i) * uv(i,j,n) , \quad \forall i \in I_j, j \in J, n \in N \quad (6d)$$

$$Te(i,j,n) \leq Prod_end(i) + H * (1 - uv(i,j,n)) , \quad \forall i \in I_j, j \in J, n \in N \quad (6e)$$

Duration Constraints

If order i is processed in tank j at event point n, that is, $uv(i,j,n) = 1$, then both ends of constraint (7a) are equal, so the duration is given by $lift(i,j,n)/l(i)$, where $l(i)$ is the lifting rate of order i. If $uv(i,j,n) = 0$, then the duration is zero according to constraint (7b).

$$\frac{lift(i,j,n) - \sum_s Prod_ord(i,s) * (1 - uv(i,j,n))}{l(i)} \leq Te(i,j,n) - Ts(i,j,n) \leq \frac{lift(i,j,n)}{l(i)} , \quad \forall i \in I_j, j \in J, n \in N \quad (7a)$$

$$Te(i,j,n) - Ts(i,j,n) \leq \frac{\sum_{s \in S_j} Prod_ord(i,s) * uv(i,j,n)}{l(i)} , \quad \forall i \in I_j, j \in J, n \in N \quad (7b)$$

Blending Stage Consideration

The consideration of the blending stage requires the incorporation of the constraints described in the following constraints.

Material Balance Constraints for the Blender

To avoid the introduction of bilinear terms in the mass-balance equations and to keep the model linear, the idea of component mixing used by Quesada and Grossmann [12] together with the assumption of constant production recipe is used. On the basis of these assumptions, constraint (8) is introduced to express that the required amount of component k to produce product s at event point n $(\sum_s (Recipe(s,k) * \sum_{j \in J_s} Blnd(s,j,n)))$ should be equal to the total amount of component k being transferred from all the component tanks at that event point $(\sum_{l \in L_k} comp(k,l,n))$.

$$\sum_s (Recipe(s,k) * \sum_{j \in J_s} Blnd(s,j,n)) = \sum_{l \in L_k} comp(k,l,n) , \quad \forall s \in S, k \in K, n \in N \quad (8)$$

Material Balance Constraints for Component Tank *l*

The amount of component k in tank l at event point $n+1$ $(bc(k,l,n+1))$ is equal to that at event point n $(bc(k,l,n))$ adjusted by any amounts transferred from separation units $(cracking(k,l,n))$ or delivered to the blender at event point $n(comp(k,l,n))$. This relation is expressed by constraint (9a). Constraint (9b) imposes the upper and the lower bounds on the flow rates of component k transferred from tank l to the blender.

$$bc(k,l,n+1) = bc(k,l,n) + cracking(k,l,n) - comp(k,l,n), \forall k \in K_l, n \in N \quad (9a)$$

$$flowmin * yv(k,l,n) \leq comp(k,l,n) \leq flowmax * yv(k,l,n) , \quad \forall k \in K, l \in L_k, n \in N \quad (9b)$$

Allocation Constraints for Product-Stock Tank *j*

Constraint (10) states that product s cannot be transferred to product-stock tank j and distributed at the same event point n.

$$\sum_{s \in S_j} sv(s,j,n) + uv(i,j,n) \leq 1, \quad \forall i \in I_j, j \in J, n \in N \quad (10)$$

Gasoline Blending and Distribution Scheduling: An MILP Model, Table 1
Distribution data for an example with ten orders

Order	o1	o2	o3	o4	o5	o6	o7	o8	o9	o10
Product and amount (Mbbl)	N411	W43	W43	N411	W43	N411	W43	N4132	W43	N5175
Time by which an order can start (hr)	0	0	24	24	48	48	96	118	144	150.5
Due date (hr)	24	24	48	48	72	72	120	190	168	185.5
Lifting rate (Mbbl/hr)	50	50	50	50	50	50	50	8	50	5

Time horizon (hr)	192										
Product-stock tank	pt1	pt2	pt3	pt4	pt5	pt6	pt7	pt8	pt9	pt10	pt11
Products that can be stored	E4W4	E4W4	E4W4	W4 E4N5	E4W4	E4W4	N4N5	N4N5	N4N5	N4N5	N4N5
Initial product and amount (Mbbl)	E490.20	–	W414.08	N587.51	W428.49	W457.59	N413.79	N412.36	N523.96	N485.11	N412.36
Maximum capacity (Mbbl)	92	92	94	91	92	84	94	92	92	91	82
Minimum capacity (Mbbl)	0.92	0.92	0.94	0.91	0.92	0.84	0.94	092	0.92	0.91	0.82

Allocation Constraints for Blender

According to constraint (11a), *xv(s,n)* equals 1 if product s is produced and transferred to at least one tank at event point n, whereas *xv(s,n)* equals 0 if product s is not transferred to any of the tanks at event point n. Constraint (11b) expresses that only one product can be produced in the blender at the same event point n.

$$sv(s,j,n) \le xv(s,n) \le \sum_{j \in J_s} sv(s,j,n)\,, \quad \forall s \in S, n \in N \qquad (11a)$$

$$\sum_{s} xv(s,n) \le 1\,, \quad \forall s \in S, n \in N \qquad (11b)$$

Sequence Constraints

Similar to constraints (6a)–(6c), constraints (12a)–(12c) state that product s should start being transferred to tank j at event point *(n+1)* after the finishing time for the same product transferred to the same tank which started at event point n, whereas constraints (12d) and (12e) represent the requirement of all the transfers to happen within the time horizon H.

$$Tbs(s,j,n+1) \ge Tbe(s,j,n) - H*(1-sv(s,j,n))\,, \quad \forall s \in S_j, j \in J, n \in N, n \ne N \qquad (12a)$$

$$Tbs(s,j,n+1) \ge Tbs(s,j,n)\,, \quad \forall s \in S_j, j \in J, n \in N, n \ne N \qquad (12b)$$

$$Tbe(s,j,n+1) \ge Tbe(s,j,n)\,, \quad \forall s \in S_j, j \in J, n \in N, n \ne N \qquad (12c)$$

$$Tbs(s,j,n) \le H\,, \quad \forall s \in S_j, j \in J, n \in N \qquad (12d)$$

$$Tbe(s,j,n) \le H\,, \quad \forall s \in S_j, j \in J, n \in N \qquad (12e)$$

If the blender provides product s for more than one product-stock tank at event point n, then the starting and finishing times for all the tanks should be the same.

$$Tbs(s,j,n) + H*(1-sv(s,j,n)) \ge Tbs(s,j',n) - H*(1-sv(s,j',n))\,, \quad \forall s \in S, j \in J_s, j' \in J_s, j \ne j', n \in N \qquad (13a)$$

Gasoline Blending and Distribution Scheduling: An MILP Model, Table 2
Blending data for an example with ten orders

Component		A	C7	C6	M	C4	C5	CR	AR	CG
Tanks that can be stored in		ct10	ct9	ct8	ct53,54 ct15,52	ct51	ct57,58 ct60	ct4 ct13	ct55 ct11	ct7,12,17 ct56,59
Recipe of products	N4	0	0.0767	0	0	0.14	0.2742	0.4018	0	0.1073
	N5	0	0	0.0419	0	0.0121	0.5178	0	0.0443	0.384
	E4	0	0	0	0	0.2729	0	0.3897	0	0.3078
	W4	0.6527	0	0	0	0.1591	0	0.1882	0	0
Amount of component (Mbbl) and tank that it is and initially stored in		26.46 ct10	67.90 ct9	59.44 ct8	7.30 ct15 5.75 ct52 3.10 ct53 28.29 ct54	0.59 ct51	0.29 ct57 8.90 ct58 1.64 ct60	19.35 ct13 27.38 ct4	13.84 ct55 25.63 ct11	4.25 ct59 53.41 ct56 49.34 ct51 34.58 ct7
Blending rate (Mbbl/hr)		50								

$$Tbs(s,j,n) - H * (1 - sv(s,j,n)) \leq Tbs(s,j',n) + H * (1 - sv(s,j',n)), \quad \forall s \in S, j \in J_s, j' \in J_s, j \neq j', n \in N \quad (13b)$$

$$Tbe(s,j,n) + H * (1 - sv(s,j,n)) \geq Tbe(s,j',n) - H * (1 - sv(s,j',n)), \quad \forall s \in S, j \in J_s, j' \in J_s, j \neq j', n \in N \quad (13c)$$

$$Tbe(s,j,n) - H * (1 - sv(s,j,n)) \leq Tbe(s,j',n) + H * (1 - sv(s,j',n)), \quad \forall s \in S, j \in J_s, j' \in J_s, j \neq j', n \in N \quad (13d)$$

Constraints (14a) and (14b) express that product transfer and distribution should be performed consecutively in the same product-stock tank j.

$$Ts(i,j,n+1) \geq Tbe(s,j,n) - H * (1 - sv(s,j,n)), \quad \forall i \in I_j, s \in S_j, j \in J, n \in N, n \neq N \quad (14a)$$

$$Tbs(s,j,n+1) \geq Te(i,j,n) - H * (1 - uv(i,j,n)), \quad \forall i \in I_j, s \in S_j, j \in J, n \in N, n \neq N \quad (14b)$$

According to constraint (15), two different products s and s' being transferred to the same or different product-stock tanks have to be transferred consecutively according to the allocation constraint for the blender.

$$Tbs(s,j,n+1) \geq Tbe(s',j',n) - H * (1 - sv(s',j',n)), \quad \forall s \in S_j, s' \in S_j, s \neq s', j \in J, j' \in J, n \in N, n \neq N \quad (15)$$

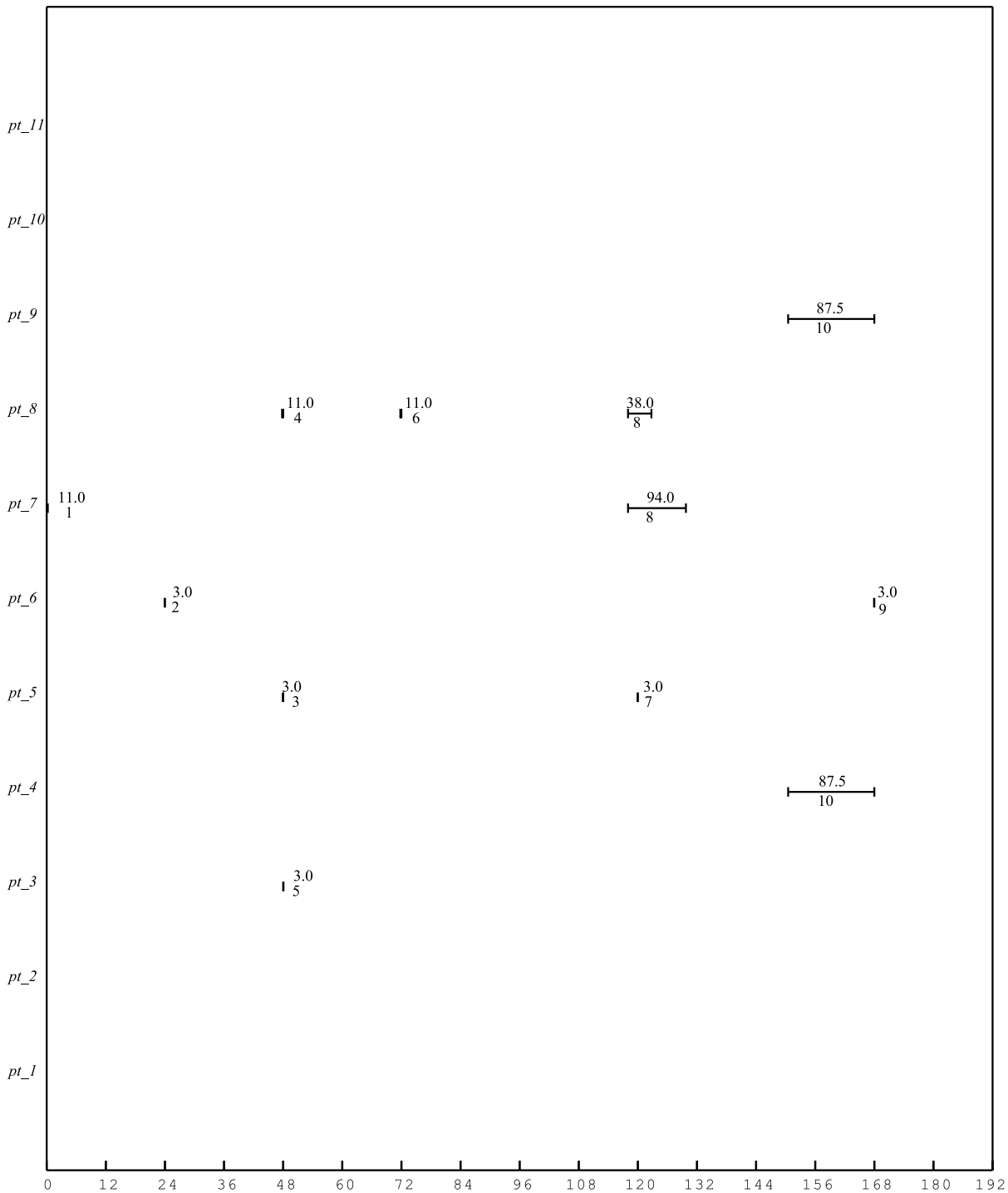
Duration Constraints

The minimum run length of 6h is imposed on the blender by constraint (16a):

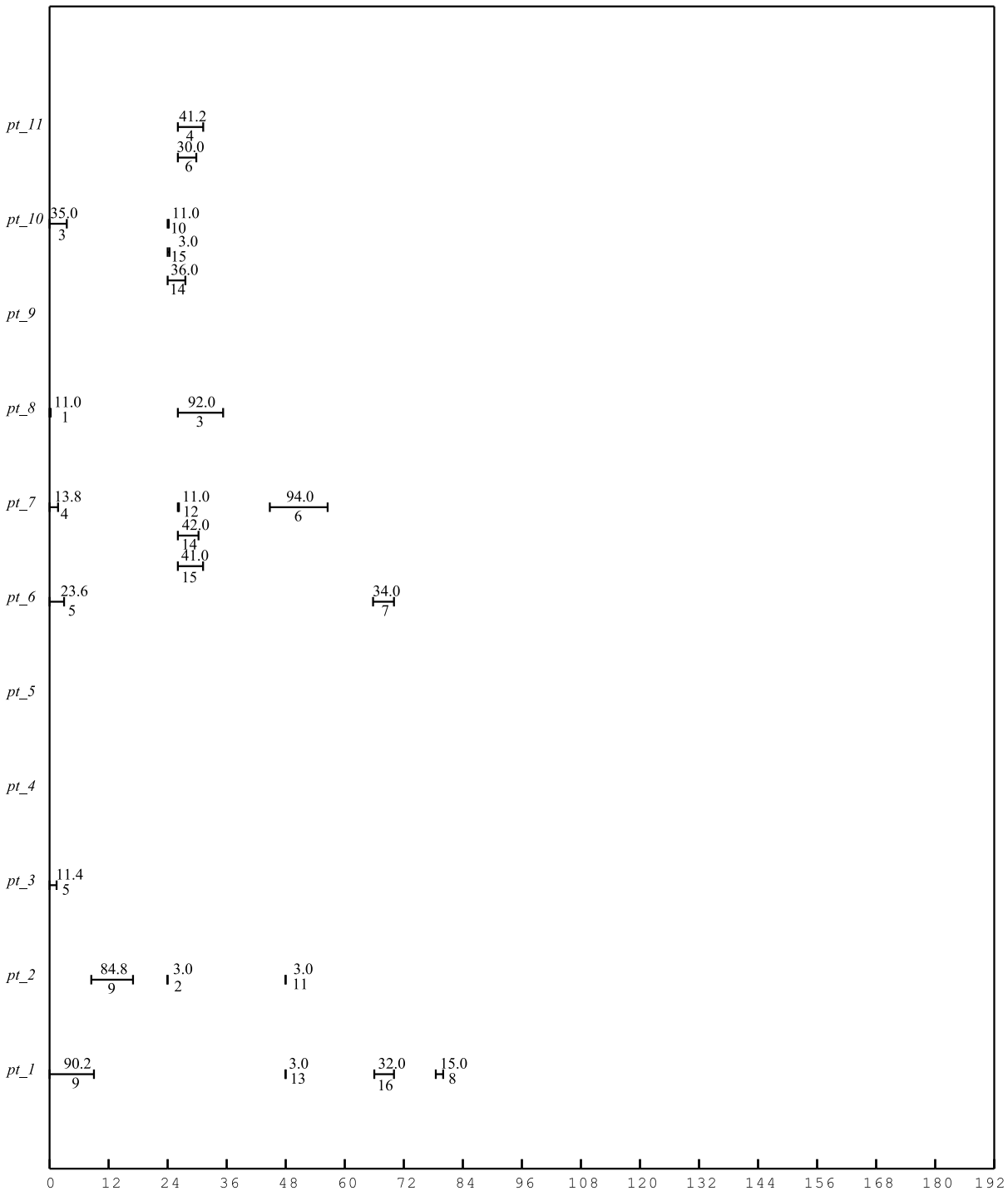
$$\sum_{j \in J_s} Blnd(s,j,n) \geq 6 * Bflow, \quad \forall s \in S, n \in N \quad (16a)$$

Constraint (16b) defines the duration of product s being transferred to the tanks at event point n as the difference between the finishing time ($Tbe(s,j,n)$) and the starting time ($Tbs(s,j,n)$), if it takes place in tank j. Constraint (16c) expresses that the duration of transferring product s from the blender to tank j corresponds to the amount of product s being transferred divided by the flow rate. The purpose of having an artificial variable ($arti(s,n)$) is to find a feasible solution in case a larger flow rate is required.

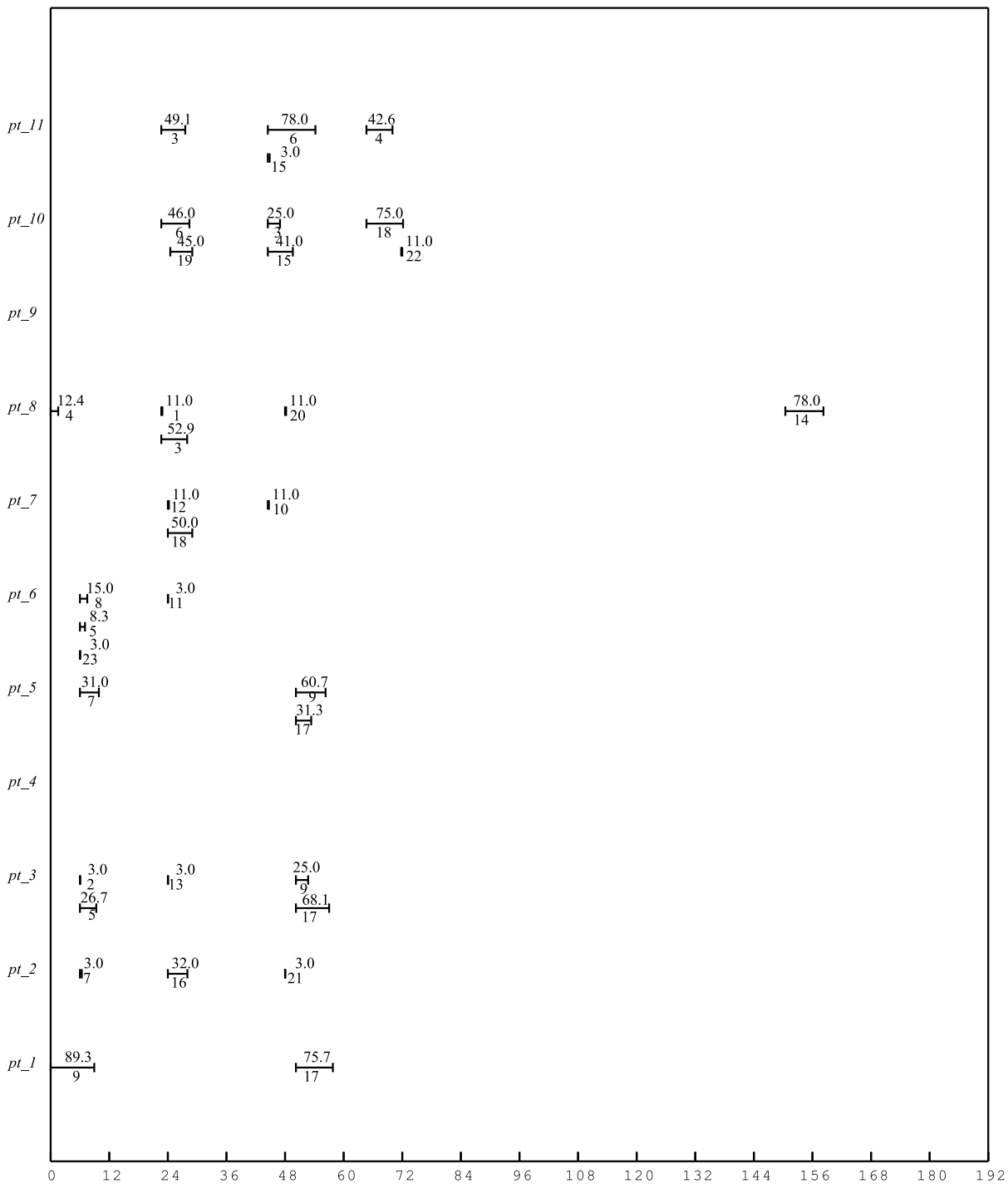
$$(Tbe(s,j,n) - Tbs(s,j,n)) - H * (1 - sv(s,j,n)) \leq duration(s,n) \leq (Tbe(s,j,n) - Tbs(s,j,n)) + H * (1 - sv(s,j,n)), \quad \forall s \in S_j, j \in J, n \in N \quad (16b)$$



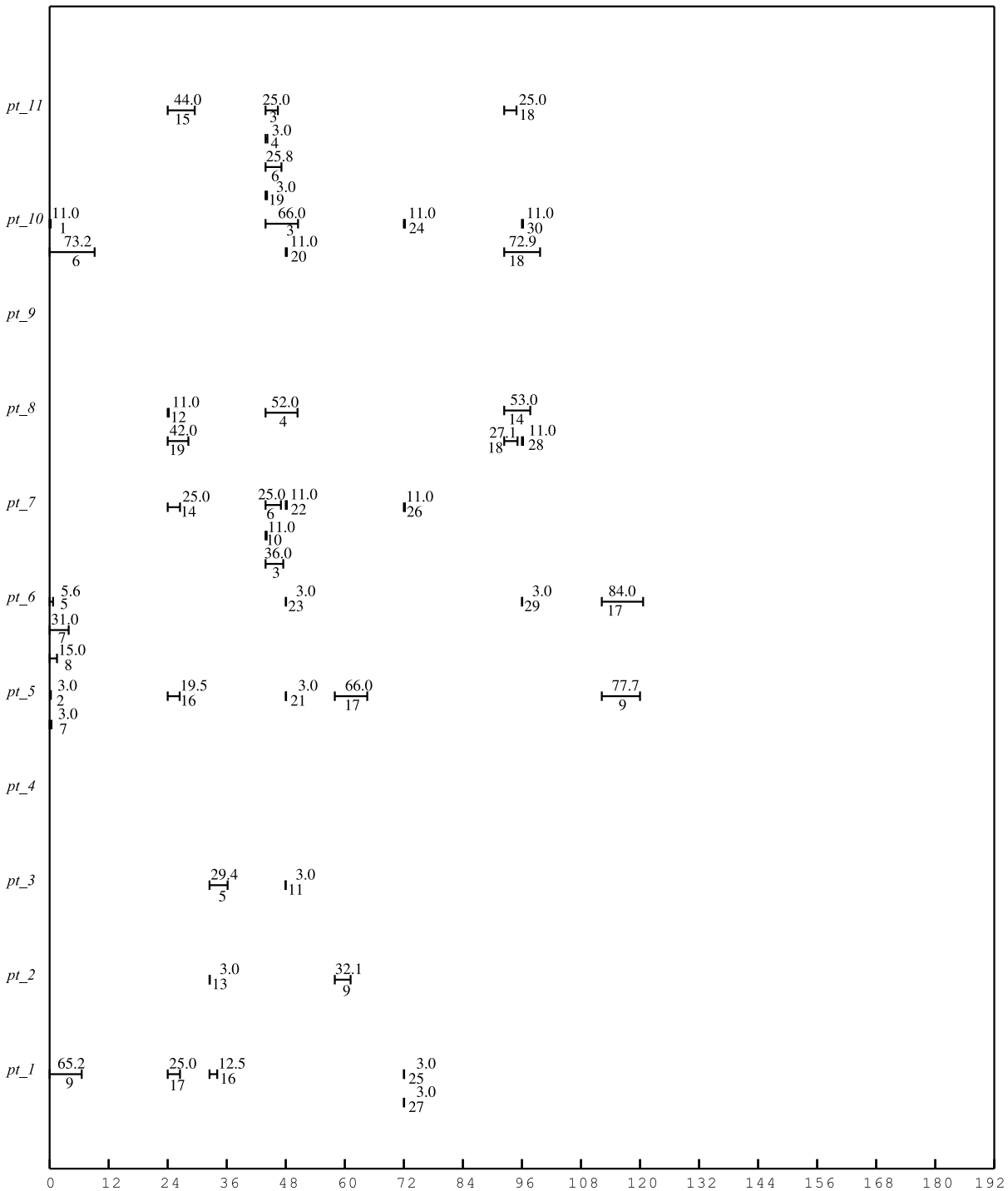
Gasoline Blending and Distribution Scheduling: An MILP Model, Figure 2
Gantt chart for the example with ten orders



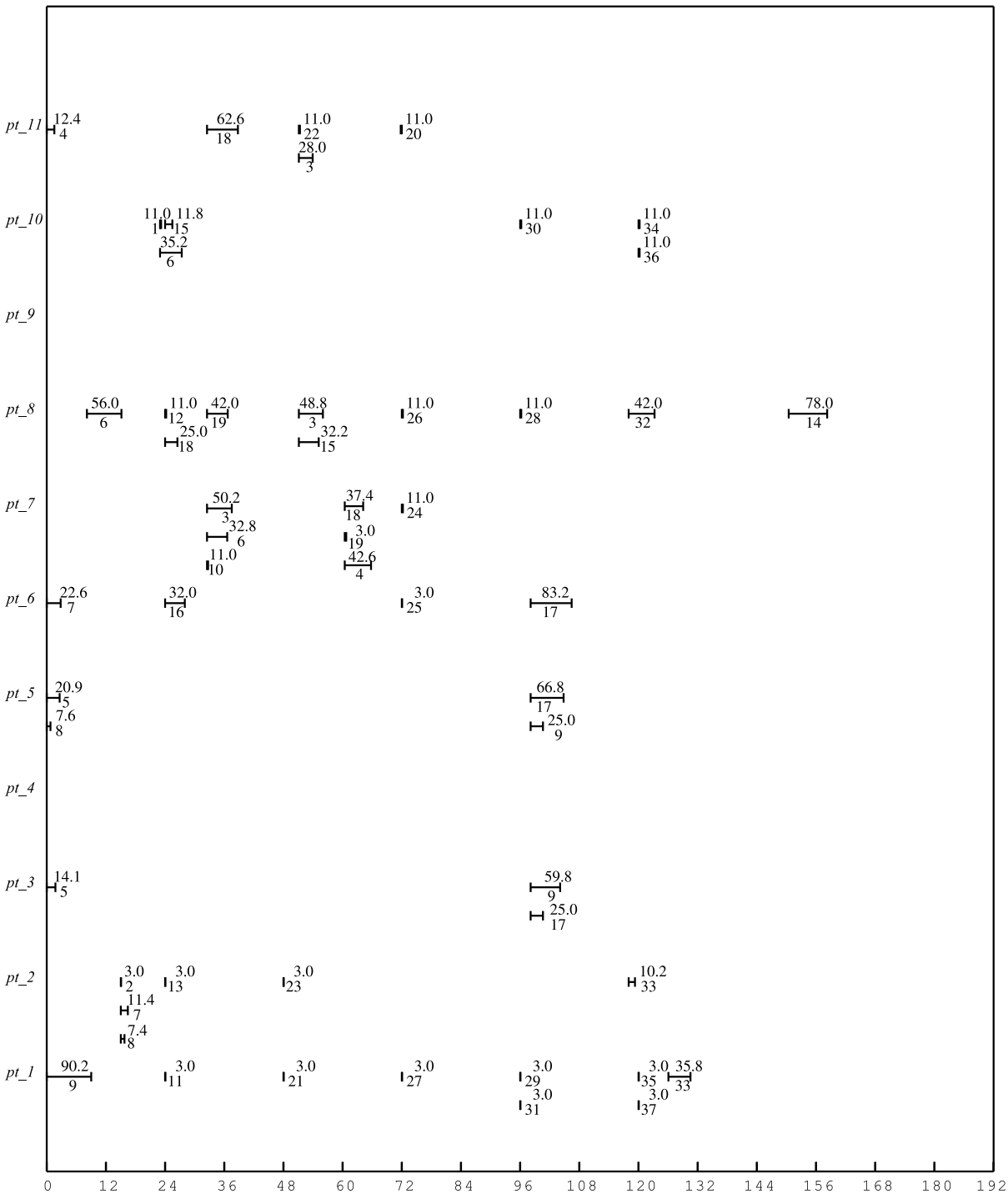
Gasoline Blending and Distribution Scheduling: An MILP Model, Figure 3
Gantt chart for the example with 16 orders



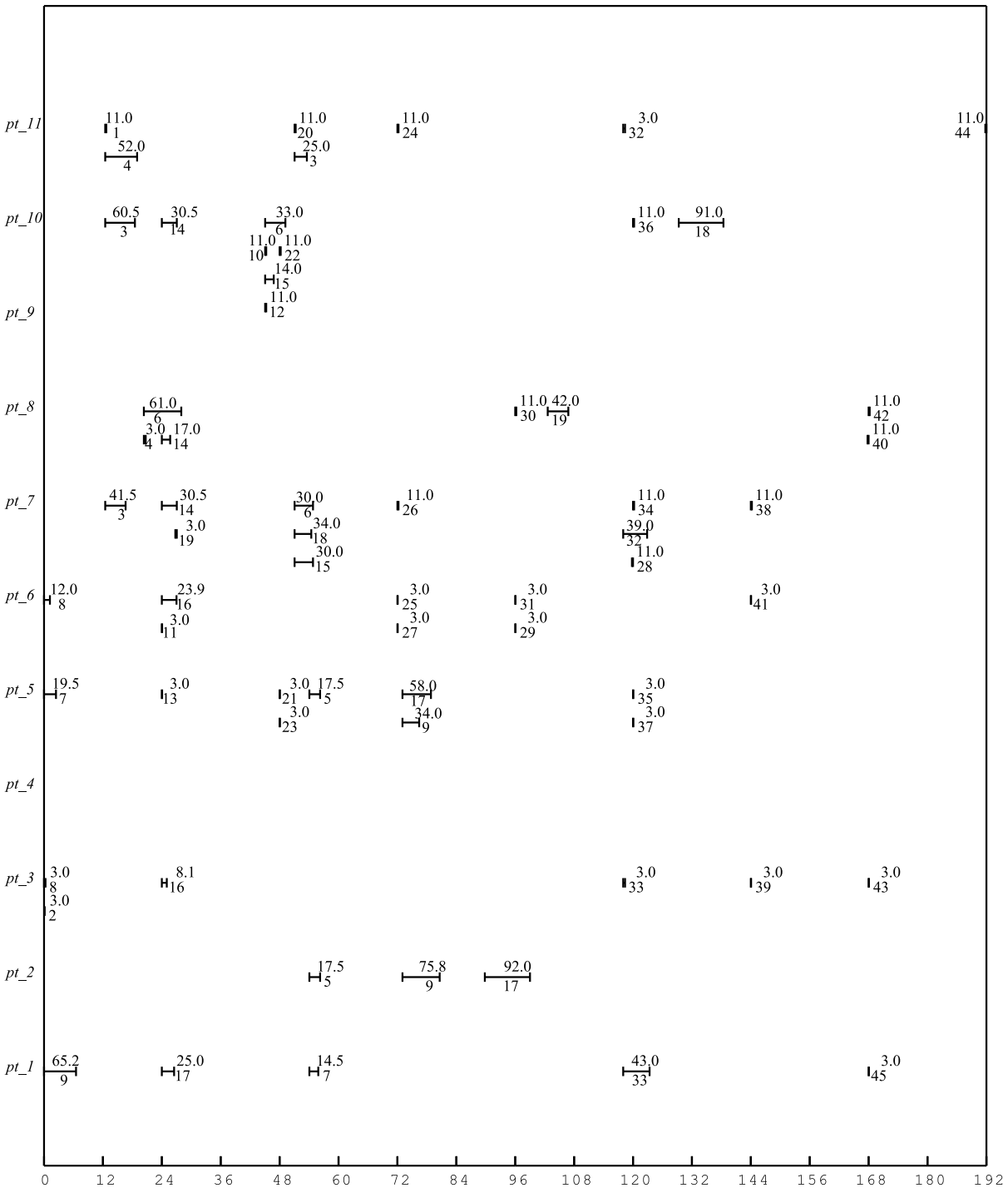
Gasoline Blending and Distribution Scheduling: An MILP Model, Figure 4
Gantt chart for the example with 23 orders



Gasoline Blending and Distribution Scheduling: An MILP Model, Figure 5
Gantt chart for the example with 30 orders



Gasoline Blending and Distribution Scheduling: An MILP Model, Figure 6
Gantt chart for the example with 37 orders



Gasoline Blending and Distribution Scheduling: An MILP Model, Figure 7
Gantt chart for the example with 45 orders

Gasoline Blending and Distribution Scheduling: An MILP Model, Table 3
Computational results for the blending and distribution system

Orders	Continuous variables	0-1 variables	Constraints	1st integer solution				2nd integer solution			Optimal solution		
				Nodes	Iterations	CPU time (s)	Objective value	Nodes	Iterations	Objective value	Nodes	Iterations	CPU time (s)
10	1706	420	7130	21	1495	6.15	0	N/A	N/A	N/A	21	1495	6.15
16	4205	1032	18737	20	3614	29.03	0	N/A	N/A	N/A	20	3614	29.03
23	5974	1470	26746	40	13474	210.24	0	N/A	N/A	N/A	40	13474	210.24
30	9056	2232	40793	80	24906	627.13	0	N/A	N/A	N/A	80	24906	627.13
37	13955	3444	63308	828	177746	4081.49	4.934	1338	244258	0.793	1353	246176	5016.51
45	25454	6289	116452	361	194954	7406.48	6.645	4138	838195	5.094	4280	874051	20351.18

$$duration(s,n) = \frac{\sum_{s\in S_j} Blnd(s,j,n)}{Bflow} - arti(s,n)\,, \quad \forall s \in S_j, j \in J, n \in N \quad (16c)$$

Objective Function

The objective of the scheduling problem is to minimize the sum of artificial variables in the duration constraints on the blender so as to determine a feasible solution with a flow rate as close to Bflow as possible. The formulation, however, is general to accommodate different objective functions targeting the optimization of production. However, in most realistic cases [11] the objective of this stage of refinery operation is to satisfy all the orders without any delays.

$$objective = \sum_s \sum_n arti(s,n)\,, \quad \forall s \in S, n \in N \quad (17)$$

Case

The case study considered here is based on realistic data provided by Honeywell Hi-Spec Solutions. The distribution problem consists of 45 orders of four different products that are stored in 11 product-stock tanks. The incorporation of the blending stage adds the consideration of nine components and 20 component tanks. Smaller-scale instances of the problem are constructed to test the proposed formulation involving the consideration of 10, 16, 23, 30, and 37 orders. The detailed data for the case often orders are presented in Tables 1 and 2. GAMS/CPLEX 7.0 was used for the solution of the resulting MILP formulation. The computational characteristics of the models are tabulated in Table 3. The optimal solution with zero integrality gap as well as the first and second integer solutions are shown. Note that since the objective corresponds to the summation of artificial variables used to relax the flow-rate constraints, if a solution has a nonzero objective this indicates that one of these constraints has been violated at the cost of the objective function. For the case study examined, however, as shown in Table 3, even the full-scale problem involving 45 orders converged to a feasible solution requiring 4280 nodes in approximately 5h CPU time which is a reasonable time for the solution of the integrated scheduling of blending and distribution problem with a time horizon of 8 days. The resulting Gantt–charts of the six cases examined are shown in Figs. 2–7. Compared with the commonly used Gantt chart for scheduling purposes, the difference here is that the number below the line corresponds to the order number, whereas the number above the line corresponds to the amount of product lifted from this particular tank. Note that different orders can be performed in the same tank at the same time as shown in Figs. 3–7.

Conclusions

In this work, a continuous-time formulation was presented for the short-term scheduling of a gasoline blending and distribution system. It was shown that the resulting model can be solved efficiently even for realistic large-scale problems. The main advantage of the proposed approach is the full utilization of the time continuity. This results in smaller models in terms of variables and constraints since only the real events have to be modeled.

References

1. DeWitt CW, Lasdon LS, Waren AD, Brenner DA, Melhem SA (1989) Omega: an improved gasoline blending system for Texaco. Interfaces 19:85
2. Dujkanovic M, Babic B, Milosevic B, Sobajic DJ, Pao YH (1996) Fuzzy linear programming based optimal fuel scheduling incorporating blending/transloading facilities. IEEE Trans Power Syst 11:1017
3. Floudas CA, Lin X (2004) Continuous-Time versus Discrete-Time Approaches for Scheduling of Chemical Processes: A Review. Comput Chem Eng 28:2109
4. Floudas CA, Lin X (2005) Mixed Integer Linear Programming in Process Scheduling: Modeling, Algorithms, and Applications. Ann Oper Res 139:131
5. Glismann K, Gruhn G (2001) Short-term planning of blending processes: scheduling and nonlinear optimization of recipes. Chem Eng Tech 24:246
6. Glismann K, Gruhn G (2001) Short-term scheduling and recipe optimization of blending processes. Comput Chem Eng 25:627
7. Ierapetritou MG, Floudas CA (1998) Effective Continuous-Time Formulation for Short-Term Scheduling. 1. Multipurpose Batch Processes. Ind Eng Chem Res 37:4341
8. Ierapetritou MG, Floudas CA (1998) Effective Continuous-Time Formulation for Short-Term Scheduling. 2. Continuous and Semicontinuous Processes. Ind Eng Chem Res 37:4360
9. Ierapetritou MG, Hene TS, Floudas CA (1999) Effective Continuous-Time Formulation for Short Term Scheduling. 3. Multiple Intermediate Due Dates. Ind Eng Chem Res 38:3446
10. Jia Z, Ierapetritou MG, Kelly JD (2003) Refinery short-term scheduling using continuous time formulation – crude oil operations. Ind Eng Chem Res 42:3085
11. Kelly JD. Honeywell Hi-Spec Solutions. Personal communication
12. Quesada I, Grossmann IE (1995) Global optimization of bilinear process network with multicomponent flows. Comput Chem Eng 19:1219
13. Rigby B, Lasdon LS, Waren AD (1995) The evolution of Texaco blending systems - from omega to starblend. Interfaces 25:64
14. Singh A, Forbes JF, Vermeer PJ, Woo SS (2000) Model-based real-time optimization of automotive gasoline blending operations. J Process Control 10:43

Gauss, Carl Friedrich

DUKWON KIM
University Florida, Gainesville, USA

MSC2000: 01A99

Article Outline

Keywords
See also
References

Keywords

Fundamental theorem of algebra; Method of least squares; Gaussian elimination

C.F. Gauss (1777–1855) worked in a wide variety of fields in both mathematics and physics including number theory, group theory, analysis, differential geometry, geodesy, magnetism, astronomy, and optics. His work has had an immense influence in many areas.

In 1788, Gauss began his education at the Gymnasium with the help of L. Büttner and R. Bartels, where he learned High German and Latin. After receiving a stipend from the Duke of Brunswick–Wolfenbüttel, Gauss entered Brunswick Collegium Carolinum in 1792. At the academy, Gauss independently discovered Bode's law, the binomial theorem and the arithmetic-geometric mean, as well as the law of quadratic reciprocity and the prime number theorem [1,4].

Gauss left Göttingen in 1798 without a diploma, but by this time he had made one of his most important discoveries: the construction of a regular 17-gon by ruler and compasses [2,3]. This was the most major advance in this field since the time of Greek mathematics and was published in his famous work 'Disquisitiones Arithmeticae' [1, Sect. VII].

On July 16, 1799, in his absence, he was awarded his Doctor of Philosophy degree at the university in Helmstedt. His dissertation is a proof of the *fundamental theorem of algebra* (FTA) [2,3]. The fundamental theorem of algebra states that

Theorem 1 *Every polynomial equation of degree n has n roots in the complex numbers.*

Gauss is usually credited with the first proof of the FTA. He is undoubtedly the first to spot the fundamental flaw in earlier proofs, namely the fact that they were assuming the existence of roots and then trying to deduce properties of them. His proof of 1799 is topological in nature and has some rather serious gaps. It does not meet our present-day standards required for a rigor-

ous proof. He published the book 'Disquisitiones Arithmeticae' in the summer of 1801. There were seven sections, all but the last section, referred to above, being devoted to number theory.

In 1814, the Swiss accountant J.R. Argand published a proof of the FTA which may be the simplest of all the proofs. His proof is based on d'Alembert's idea in 1746. Argand simplifies d'Alembert's idea using a general theorem on the existence of a minimum of a continuous function.

Two years after Argand's proof appeared Gauss published in 1816 a second proof of the FTA. Gauss uses Euler's approach but instead of operating with roots which may not exist, Gauss operates with indeterminates. This proof is complete and correct. A third proof by Gauss also in 1816 is, like the first, topological in nature. Gauss introduced in 1831 the term 'complex number'.

In 1849 Gauss produced the first proof that a polynomial equation of degree n with complex coefficients has n complex roots. The proof is similar to the first proof given by Gauss. However it adds little since it is straightforward to deduce the result for complex coefficients from the result about polynomials with real coefficients.

It is worth noting that despite Gauss's insistence that one could not assume the existence of roots which were then to be proved reals he did believe, as did everyone at that time, that there existed a whole hierarchy of imaginary quantities of which complex numbers were the simplest. Gauss called them a *shadow of shadows*.

The different proofs of the FTA are Gauss's most important contributions as a rigorist, that is to say, as a representative of logical strictness in method of proof [1]. Since this theorem has great significance in both algebra and function theory, it influenced many other related areas, including mathematical optimization.

Gauss used infinite sequences and series in his daily work, not only in mathematics but in astronomy, geodesy, and physics. As an eleven-year-old, Gauss was already studying Newton's binomial theorem, which includes the infinite geometric series as a special case. He investigated the conditions under which an infinite binomial series has a logical meaning. He also thought about the theoretical formulation of the notion of limiting value [3]. In an unfinished article written around 1800, 'Fundamental concepts in the principles of series', he formulated the notion of the limit of a sequence in a fashion far ahead of the times.

Gauss introduced there the notions of upper bound and least upper bound G; he also introduced the notions of lower bound and greatest lower bound g. Furthermore he introduced the 'final upper bound' H and the 'final lower bound' h. If $H = h$, then their common value was called the *absolute limit* (limiting value) of the sequence. His definitions nearly agree with the present-day definitions of upper bound G, lower bound g, limit superior H, limit inferior h, and the condition $H = h$ for the existence of the *limiting value* [3,4].

Gauss's great interest in astronomy, and his later interest in geodesy, compelled him to seek a rational method for determining the magnitude of observational errors. In turn, the theory of observational errors forced him to deal with the modes of thought and concepts of the calculus of probabilities. This work had great significance in the development of numerous areas in both the calculus of probabilities and mathematical statistics. Furthermore this theory forced researchers to make clear the conditions under which the *law of the normal distribution* is applicable. This law is often called *Gauss's distribution law*.

In 1823 Gauss published his great work 'Theoria combinationis observationum erroribus minimus obnoxiae' ('A theory for the combination of observations, which is connected with least possible error'). It is a systematic and generalized presentation of his earlier theory of observational errors. Here he develops the *method of least squares* [3,4] with mathematical rigor as, in general, the most suitable way of combining observations, independent of any hypothetical law concerning the probability of error.

The term 'determinant' was first introduced by Gauss in 'Disquisitiones Arithmeticae' (1801) while discussing quadratic forms [3]. He used the term because the determinant determines the properties of the quadratic form. However the concept is not the same as that of our determinant. In the same work Gauss lays out the coefficients of his quadratic forms in rectangular arrays. He describes matrix multiplication (which he thinks of as composition so he has not yet reached the concept of matrix algebra) and the inverse of a matrix in the particular context of the arrays of coefficients of quadratic forms.

Gaussian elimination, which first appeared in the text 'Nine Chapters of the Mathematical Art' written in 200 BC, was used by Gauss in his work which studied the orbit of the asteroid Pallas. Using observations of Pallas taken between 1803 and 1809, Gauss obtained a system of six linear equations in six unknowns. Gauss gave a systematic method for solving such equations which is precisely Gaussian elimination on the coefficient matrix [1].

Gauss's career was marked by distinct periods during which he immersed himself first in astronomy, then in geodesy, and then in physics. Yet he regarded himself first and last as 'entirely a mathematician'. More Gauss was an outstanding example of the few creative thinkers who were equally at home in both pure mathematics and applied mathematics. Gauss was always trying to find new applications of mathematics. He kept many little notebooks in which he wrote down ideas and suggestions as they occurred to him. Always alert to possibilities of applying mathematical theories to practical problems, he foresaw the use of mathematics not only in science and technology, but also in such fields as economics, statistics, finance, and so on.

During his long and active career, Gauss published a considerable number of books and articles in journals. But upon his death in 1855, many unpublished articles, notes, and manuscripts were found in his desk. When his complete 'Collected Works' were finally published later, it had taken a group of German scientists nearly seventy years to edit his writings. Even today the name of Gauss occurs throughout mathematics and related areas over and over again. We have the Gaussian equations in spherical trigonometry; the hypergeometric series is also called the Gaussian series; the normal probability curve is known as the Gaussian curve; Gaussian period is a period of congruent roots in the division of the circle; addition and subtraction logarithms are also known as Gaussian logarithm; in higher geometry we speak of Gauss's theorem and Gauss curvature; certain formulas for approximations are known as *Gaussian approximation methods*.

To appreciate the genius of a man like Gauss we must also see him in perspective, through the eyes of his colleagues, his students, his friends, and in terms of posterity's verdict. No other mathematician of the nineteenth century ever received as much acclaim and recognition as that given to Gauss.

See also

- ► Gauss–Newton Method: Least Squares, Relation to Newton's Method
- ► Least Squares Problems
- ► Linear Programming
- ► Symmetric Systems of Linear Equations

References

1. Bühler WK (1981) Gauss: A biographical study. Springer, Berlin
2. Dunnington GW (1955) Carl Friedrich Gauss: Titan of science. Exposition Press, New York
3. Hall T (1970) Carl Friedrich Gauss. MIT, Cambridge, MA
4. Schaaf WL (1964) Carl Friedrich Gauss: Prince of mathematicians. Franklin Watts, London

Gauss–Newton Method: Least Squares, Relation to Newton's Method

WILLIAM R. ESPOSITO,
CHRISTODOULOS A. FLOUDAS
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C30, 90C30, 90C52, 90C53, 90C55

Article Outline

Keywords
Newton's Method
Properties of Newton's Method
Gauss–Newton Method
Comparisons Between Newton and Gauss–Newton Method
Variable Steplength
Gauss–Newton Example
Modifications and Applications
See also
References

Keywords

Gauss–Newton method; Least squares; Gradient methods

Least squares optimization appears most often in *parameter estimation* problems involving nonlinear models. In this problem the object is to minimize the squared distance between an observed and a fitted value from a model with adjustable parameters. For a single equation model the formulation becomes

$$\min_{\theta} S(\theta) = \sum_{\mu=1}^{n} \left[y_\mu - f(\theta, \mathbf{x}_\mu) \right]^2 , \tag{1}$$

where θ are the adjustable model parameters, y_μ is the observed value of the the dependent variable (assumed to contain error) at the μ data point, $\boldsymbol{x}_\mu$ are the observed values of the independent variables (assumed error free) at the μ data point, and n is the total number of data points observed.

This is a very common and well studied problem. As a result many different solution methods exist. In particular two of the earlier developed methods, Newton's method and the Gauss–Newton approach will be discussed and the relationship between the two will be presented.

Newton's Method

Newton's method is derived based on a second order *Taylor series* expansion of the objective function around the current 'guess' of the solution θ_i:

$$\begin{aligned} Q_i(\theta) = S(\theta_i) &+ \mathbf{q}^\top (\theta - \theta_i) \\ &+ \frac{1}{2} (\theta - \theta_i)^\top \mathbf{H} (\theta - \theta_i) \end{aligned} \tag{2}$$

with

$$\mathbf{q}_l = \frac{\partial S}{\partial \theta_l} = -2 \sum_{\mu=1}^{n} e_\mu \frac{\partial f_\mu}{\partial \theta_l} , \tag{3}$$

$$\begin{aligned} \mathbf{H}_{lk} &= \frac{\partial^2 S}{\partial \theta_l \partial \theta_k} \\ &= -2 \sum_{\mu=1}^{n} e_\mu \frac{\partial^2 f_\mu}{\partial \theta_l \partial \theta_k} + 2 \sum_{\mu=1}^{n} \frac{\partial f_\mu}{\partial \theta_l} \frac{\partial f_\mu}{\partial \theta_k} , \end{aligned} \tag{4}$$

where $e_\mu = y_\mu - f_\mu$ and $f_\mu = f(\mathbf{x}_\mu, \theta)$. In order to find a stationary point of (2) the first order derivatives are equated to zero:

$$\frac{\partial Q_i}{\partial \theta} = \mathbf{q}_i + \mathbf{H}_i (\theta - \theta_i) = 0. \tag{5}$$

If $\mathbf{H}$ is nonsingular, then the solution of (5) for θ can be written as:

$$\theta = \theta_i - \mathbf{H}_i^{-1} \mathbf{q}_i . \tag{6}$$

The method is implemented in a iterative fashion where the value of θ from (6) is used as the next 'guess' of the solution. The iterations continue until a convergence criterion is reached. Theoretically this should be based on the first order derivatives being equal to zero. But for practically purposes and numerical reasons the criterion is most often based on the change in the parameter values. For example:

$$\frac{|\theta_{i+1} - \theta_i|}{|\theta_i| + \epsilon_1} \leq \epsilon_2 , \tag{7}$$

where ϵ_1 and ϵ_2 are arbitrary small constants.

Properties of Newton's Method

Newton's method has the following properties [11]:

- Converges in one iteration if $S(\theta)$ is quadratic, as is the case when the model $f(\theta, \mathbf{x})$ is linear in the parameters.
- Requires that both the first and second derivatives of $S(\theta)$ are computed.
- Inversion of the Hessian matrix of $S(\theta)$ is required at each iteration ($O(n^3)$ operation).
- The iteration is undefined when $\mathbf{H}$ is singular.
- $\mathbf{H}$ is required to be positive definite for the step to reduce the value of the objective function.
- Outside the neighborhood of the minimum, convergence is not guaranteed.

Many of these properties, especially the requirement of second derivatives, makes this method impractical for most physically significant problems.

Gauss–Newton Method

The method developed by C.F. Gauss [7] attempts to overcome some of the drawbacks to the original Newton approach. A closer look at (4) shows that for small errors (e_μ) the first term in the equation is approximately zero:

$$-2 \sum_{\mu=1}^{n} e_\mu \frac{\partial^2 f_\mu}{\partial \theta_l \partial \theta_k} \approx 0 \quad \text{for } e_\mu \ll 1. \tag{8}$$

Therefore the Hessian matrix $\mathbf{H}_i$ can be approximated as:

$$\mathbf{H}_i \approx \mathbf{H}_i^* = 2\sum_{\mu=1}^{n} \frac{\partial f_\mu}{\partial \theta_l}\frac{\partial f_\mu}{\partial \theta_k}. \tag{9}$$

A step in the solution method then takes the form:

$$\theta_{i+1} = \theta_i - \mathbf{H}_i^{*-1}\mathbf{q}_i. \tag{10}$$

This method can be viewed as linearizing the nonlinear model, and then solving the resulting linear regression to determine the starting point for the next iteration [4]. The Gauss–Newton method has the following properties [12]:

- Only first derivatives of $S(\theta)$ need to be computed at each iteration.
- The approximated Hessian matrix $\mathbf{H}^*$ is intrinsically positive definite and due to the structure, inversion is much easier.
- The approximation is exact if the errors e_μ tend to zero at the minimum.
- Outside the neighborhood of the minimum, convergence is not guaranteed.

These properties offer improvements over the Newton method especially in the computational effort required.

Comparisons Between Newton and Gauss–Newton Method

Various comparisons have been made between these two methods:

1) If the model fits the data well (i. e., all e_μ are small at the solution), then the Gauss–Newton method often requires no more iterations than the Newton method [1].
2) If the model does not fit the data well (i. e., some e_μ do not tend to zero at the solution), then the Newton method will require fewer iterations than the Gauss–Newton, but the computation times will be similar [6].

Both of these methods are similar in that they fall under the category of *gradient based approaches*. In general, a gradient method is iterative in which the step at each iteration is defined as:

$$\theta_{i+1} = \theta_i - \rho_i \mathbf{R}_i \mathbf{q}_i, \tag{11}$$

where $\mathbf{q}_i$ is defined earlier, ρ_i is the steplength, and $\mathbf{R}_i$ is a matrix which should be positive definite. In the Newton method $\mathbf{R}_i$ is the inverse Hessian $\mathbf{H}^{-1}$, while Gauss–Newton uses the approximation $\mathbf{H}^{*-1}$. As mentioned earlier, the inverse Hessian is not always positive definite, while the approximation is, except in the case that the Jacobian matrix, $\mathbf{q}$, is rank deficient. In the implementation of both methods, the steplength ρ_i is taken as 1.

Variable Steplength

One of the obvious extensions of the method involves a selection of the steplength other than one. At each iteration, the search direction given by the Gauss–Newton step is downhill due to the positive definiteness of the approximate Hessian. But the step does not necessarily result in a reduction of the objective function S, since overshooting the minimum is possible. Therefore a steplength ρ should be chosen such that at least:

$$S(\theta_{i+1}) \leq S(\theta_i). \tag{12}$$

One such method can be found in [3]. First define the function $\Psi_i(\rho)$ as:

$$\Psi_i(\rho) \equiv S(\theta_i - \rho \mathbf{R}_i \mathbf{q}_i). \tag{13}$$

The value of $\Psi_i(0)$ is defined as $S(\theta_i)$. An initial value of ρ^o is chosen and the value of $\Psi_i(\rho^o)$ is calculated. If $\Psi_i(\rho^o)$ is greater than $\Psi_i(0)$, then obviously this value of ρ is not acceptable. Even if the value of ρ is acceptable, the following process may still offer an improvement.

The function $\Psi_i(\rho)$ can be approximated by a quadratic function which matches at $\rho = 0$, $\rho = \rho^0$, and the slope at $\rho = 0$. The function takes the form:

$$\Psi_i(\rho) \approx a + b\rho + c\rho^2 \tag{14}$$

with the coefficients defined as:

$$a = \Psi_i(0) = S(\theta_i),$$
$$b = \left.\frac{d\Psi_i}{d\rho}\right|_{\rho=0} = -\mathbf{q}_i^\top \mathbf{R}_i \mathbf{q}_i,$$
$$c = \frac{\Psi_i(\rho^o) - a - b\rho^o}{(\rho^o)^2}.$$

The object is to minimize this approximation over ρ. A stationary point occurs at:

$$\rho^* = \frac{-b}{2c}. \tag{15}$$

This calculation can be used in an iterative fashion until an acceptable value of ρ is found which reduces the objective function. Reference [3] contains a detailed implementation of this iterative calculation.

Gauss–Newton Example

This example of the Gauss–Newton approach with a variable steplength is found in [3]. This example consists of a two parameter single equation model of the form:

$$y = \exp\left\{-\theta_1 x_1 \exp\left[-\frac{\theta_2}{x_2}\right]\right\}. \tag{16}$$

The parameters, θ, represent the *Arrhenius constants* for a first order irreversible reaction:

$$A \xrightarrow{k} B$$

with x_1 representing the reaction time, x_2 the reaction temperature, and y the fraction of A remaining. The data for the example can be found in the table below.

μ	x_1(hr)	x_2(K)	y
1	0.10	100	0.980
2	0.20	100	0.983
3	0.30	100	0.955
4	0.40	100	0.979
5	0.50	100	0.993
6	0.05	200	0.626
7	0.10	200	0.544
8	0.15	200	0.455
9	0.20	200	0.255
10	0.25	200	0.167
11	0.02	300	0.566
12	0.04	300	0.317
13	0.06	300	0.034
14	0.08	300	0.016
15	0.10	300	0.066

The objective is to minimize the least squares function:

$$\min_{\theta} S(\theta) = \sum_{\mu=1}^{15} \left[y - f_\mu(\theta)\right]^2. \tag{17}$$

The gradients, $\mathbf{q}$, of the objective function take the form:

$$q_1 = 2\sum_{\mu=1}^{15} e_\mu f_\mu \exp\left[-\frac{\theta_2}{x_{\mu 2}}\right] x_{\mu 1}, \tag{18}$$

$$q_2 = -2\sum_{\mu=1}^{15} e_\mu f_\mu \frac{\theta_1 x_{\mu 1}}{x_{\mu 2}} \exp\left[-\frac{\theta_2}{x_{\mu 2}}\right], \tag{19}$$

and the approximate Hessian matrix is given by:

$$H^*_{lk} = 2\sum_{\mu=1}^{15} \frac{\partial f_\mu}{\partial \theta_l}\frac{\partial f_\mu}{\partial \theta_k}, \quad l,k = 1,2, \tag{20}$$

where:

$$\frac{\partial f_\mu}{\partial \theta_1} = f_\mu \exp\left[-\frac{\theta_2}{x_{\mu 2}}\right] x_{\mu 1}, \tag{21}$$

$$\frac{\partial f_\mu}{\partial \theta_2} = f_\mu \frac{\theta_1 x_{\mu 1}}{x_{\mu 2}} \exp\left[-\frac{\theta_2}{x_{\mu 2}}\right]. \tag{22}$$

The initial guess for the parameter values is taken as:

$$\theta_1 = \begin{pmatrix}\theta_{1,1}\\ \theta_{1,2}\end{pmatrix} = \begin{pmatrix}750\\ 1200\end{pmatrix}.$$

Using this initial guess the value of the objective function, gradients, and approximated Hessian were calculated.

$$S(\theta_1) = 1.090441,$$

$$\mathbf{q}_1 = \begin{pmatrix}-0.002230450\\ 0.006863795\end{pmatrix},$$

$$\mathbf{H}^*_1 = \begin{pmatrix}0.2689478 & -0.7730614\\ -0.7730614 & 2.310325\end{pmatrix} \times 10^{-5}.$$

The search step direction ν_1, is calculated from $-\mathbf{H}^{*-1}_1 \mathbf{q}_1$. This is generally accomplished by solving the linear system:

$$-\mathbf{H}^*_1 \nu_1 = \mathbf{q}_1. \tag{23}$$

Many different numerical techniques exist for the solution of (23), see [5] or [13] for examples. The calculation results in:

$$\nu_1 = \begin{pmatrix}-644.9785\\ -512.9099\end{pmatrix}.$$

Initially using a stepsize $\rho^0 = 1$, the following values for the parameters are:

$$\theta^0 = \begin{pmatrix} 105.0215 \\ 687.0901 \end{pmatrix}.$$

An objective value of $S(\theta^0) = 0.9133969$ results, which is less than $S(\theta_1)$. Even though this is an acceptable value, still a different stepsize may give a better result. Using the approximation given in (14) with the following values of the parameters for the fit:

$$\Psi_i(\rho = 0) = 1.090441,$$
$$\Psi_i(\rho = 1) = 0.9133969,$$
$$\left.\frac{d\Psi_i}{d\rho}\right|_{\rho=0} = -2.081916.$$

The parabola has a minimum, given by (15), at a steplength $\rho^* = 0.5464714$. The resulting values of the parameters using this steplength are:

$$\theta^1 = \begin{pmatrix} 397.5376 \\ 919.7092 \end{pmatrix}.$$

An objective value of $S(\theta^1) = 0.3345645$ results, which is a large improvement over $S(\theta^0)$. This value of the parameter set, θ^1, is accepted as θ_2, and the iterations continue. The results of the iterations can be found in the table below.

i	$S(\theta_i)$	$\theta_{i,1}$	$\theta_{i,2}$
1	1.090411	750	1200
2	0.3345645	397.5376	919.7092
3	0.05765885	646.0847	938.5288
4	0.04038005	810.6260	965.7625
5	0.03980731	818.3628	962.1228
6	0.03980599	813.4583	960.9063

The value of the parameters and the objective function at the sixth iteration are accepted as the solution to the problem. The final values of the gradients and the approximate Hessian are:

$$\mathbf{q} = \begin{pmatrix} -0.218524 \\ 0.631308 \end{pmatrix} \times 10^{-6},$$
$$\mathbf{H}^* = \begin{pmatrix} 0.271890 & -0.957336 \\ -0.957336 & 3.50371 \end{pmatrix} \times 10^{-5}.$$

The above calculation benefited from that fact that the initial guess for the parameter values was relatively close to the solution. Take now the same example, but using the following parameter values as the starting point of the calculation:

$$\theta_1 = \begin{pmatrix} 100 \\ 2000 \end{pmatrix}.$$

This is obviously a 'worse' starting point than the previous calculation. Using these parameter values the following results:

$$S(\theta_1) = 5.299502,$$
$$\mathbf{q}_1 = \begin{pmatrix} -0.0007098080 \\ 0.0002442936 \end{pmatrix},$$
$$\mathbf{H}_1^* = \begin{pmatrix} 0.7036033 & -0.2354773 \\ -0.2354773 & 0.07896382 \end{pmatrix} \times 10^{-7},$$
$$\nu_1 = \begin{pmatrix} -134608.0 \\ -432361.0 \end{pmatrix},$$
$$\theta^0 = \begin{pmatrix} -134508.0 \\ -430361.0 \end{pmatrix}.$$

Using the value of θ^0, calculated with $\rho^0 = 1$, it is not possible to calculate the value of the objective function since the resulting exponentials are very large. The value of ρ was repeatedly halved until a reasonable value of the objective function was obtained. The value $\rho^0 = 2^{-8} = 0.00390625$ resulted in:

$$\theta^0 = \begin{pmatrix} -425.8140 \\ 311.0039 \end{pmatrix}.$$

An objective value of $S(\theta^0) = 0.3366272 \times 10^{20}$ results, which is not acceptable. The stepsize needs to be adjusted such that the objective function decreases. This is accomplished in the same way as outlined previously. The parabolic approximation reaches a minimum at $\rho^* \approx 5 \times 10^{-25}$. This is too small to be practical, so a value of $\rho^1 = \rho^0/4$ will be used. This results in $S(\theta^1) = 5.471375$. Again this is not acceptable since it is larger than $S(\theta_1)$. The value of ρ is iterated on until an acceptable value is determined. Finally after three more iterations, $\rho^4 = 0.0000619701$, which produces:

$$\theta^4 = \begin{pmatrix} 91.65955 \\ 1973.211 \end{pmatrix}.$$

An objective value of $S(\theta^4) = 5.299135$ results, which is just less than the original value of 5.299502, but given

the criterion in (12) is acceptable. θ^4 is accepted as θ_2 and the iterations continue.

The solution, in this case, is obtained after 25 iterations. This illustrates the major downfall of the Gauss–Newton method, that without a 'good' initial guess convergence to the solution is slow at best and not guaranteed. In fact without using a variable stepsize, the algorithm would have blown up after just one iteration.

Modifications and Applications

A very large number of different variations on the basic Gauss–Newton algorithm exist. For the most part, these variations include methods to determine the stepsize, and approaches which actually improve the accuracy of the approximated Hessian matrix. For examples of different variations see [10] or [8]. Others have done comparisons and numerical experiments with popular variations to test their applicability to a wide range of problems [2,15]. The algorithm has also been applied to what is referred to as *weighted least squares* (WLS) in which each term in the objective function receives a different coefficient:

$$\min_{\theta} \overline{S}(\theta) = \sum_{\mu=1}^{n} w_\mu \left[y_\mu - f(\theta, \mathbf{x}_\mu) \right]^2 , \qquad (24)$$

where w_μ is the weighting for the μth data point, see [14] and [9] for examples.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- ABS Algorithms for Optimization
- Gauss, Carl Friedrich
- Generalized Total Least Squares
- Least Squares Orthogonal Polynomials
- Least Squares Problems
- Nonlinear Least Squares: Newton-type Methods
- Nonlinear Least Squares Problems
- Nonlinear Least Squares: Trust Region Methods

References

1. Bard Y (1967) A function minimization method with application to parameter estimation. In: New York Sci Center Report 3220902, IBM
2. Bard Y (1970) Comparison of gradient methods for the solution of nonlinear parameter estimation problems. SIAM J Numer Anal 7(1):157–186
3. Bard Y (1974) Nonlinear parameter estimation. Acad. Press, New York
4. Bard Y, Lapidus L (1968) Kinetics analysis by digital parameter estimation. Catalysis Rev 2(1):67–112
5. Coleman TF, Van Loan C (1988) Handbook of matrix computations. SIAM, Philadelphia
6. Flanagan PD, Vitale PA, Mendelsohn J (1969) A numerical investigation of several one-dimensional search procedures in nonlinear regression problems. Technometrics 11(2):265–284
7. Gauss KF (1809) Theoria motus corporum coelestium. Werke 7:240–254
8. Gill PE, Murray W (1978) Algorithms for the solution of the nonlinear least-squares problem. SIAM J Numer Anal 15(5):977–992
9. Guillaume P, Pintelon R (1996) A Gauss–Newton-like optimization algorithm for "weighted" nonlinear least-squares problems. IEEE Trans Signal Processing 44(9):2222–2228
10. Hartley HO (1960) The modified Gauss–Newton method for the fitting on non-linear regression functions by least squares. Technometrics 3(2):269–280
11. Mckeowen JJ (1975) On algorithms for sums of squares problems. In: Dixon LCW, Szego GP (eds) Toward Global Optimization. North-Holland, Amsterdam, 229–257
12. Meyer RR (1970) Theoretical and computational aspects of nonlinear regression. In: Rosen J, Mangasarian O, Ritter K (eds) Nonlinear Programming. Acad. Press, New York, pp 466–487
13. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1995) Numerical recipes in C, 2nd edn. Cambridge Univ. Press, Cambridge
14. Reedy PV, Niranjan, Sridharan K, Rao PV (1996) WLS method for parameter estimation in water distribution networks. J Water Resources Planning and Management 122(3):157–164
15. Spedicato E, Vespucci MT (1988) Numerical experiments with variations of the Gauss–Newton algorithm for nonlinear least squares. J Optim Th Appl 57(2):323–339

Gene Clustering: A Novel Decomposition-Based Clustering Approach: Global Optimum Search with Enhanced Positioning

MENG PIAO TAN, CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 91C20, 90C11, 90C26

Article Outline

Introduction
Formulations
Notation and Pre-Clustering
Proposed Algorithm
Case Study
Experimental Data
Description of Comparative Study
Results and Discussion
References

Introduction

The aim of cluster analysis is to establish a set of clusters such that the data points in a cluster are more similar to one another than they are to those in other clusters. The clustering problem is old, can be traced back to Aristotle, and has already been studied quite extensively by 18th century naturalists such as Buffon, Cuvier, and Linne [19]. Since then, clustering has been used in many disciplines, such as market research, social network analysis, and geology, thus reflecting its broad appeal and utility as a key step in exploratory data analysis [26]. In market research for instance, cluster analysis is widely used when working with multivariate data from surveys and test panels. Market researchers use cluster analysis methods to segment and determine target markets, and position new products. Cluster analysis is also used in the service of market approaches to the establishment of business enterprise value. Johnson [28] addresses the potential role and utility of cluster analysis in transfer pricing practices. Given the importance of clustering, a substantial number of books, such as [11,20,27,39], as well as review papers, such as [58] have been published on this subject.

In biology, clustering provides insights into transcriptional networks, physiological responses, gene identification, genome organization, and protein structure. Genome-wide measurement of mRNA expression levels is an efficient way of gathering comprehensive information on genetic functions and transcriptional networks. However, extracting useful information from the resulting data sets first involves organizing genes by their pattern and/or intensity of expression in order to define those that are co-regulated. Such information provides a basis for extracting regulatory motifs for transcription factors driving the diverse expression patterns, allowing assembly of predictive transcriptional networks [2]. This information also provides insights into the functions of unknown genes, since functionally related genes are often co-regulated [55]. Furthermore, clustered array data provides identification of distinct categories of otherwise indistinguishable cell types, which can have profound implications in processes such as disease progression [50]. In sequence analysis, clustering is used to group homologous sequences into gene families. Examining characteristic DNA fragments helps in the identification of gene structures and reading frames. In protein structure prediction, clustering the ensemble of low energy conformers is used to identify the top suggested protein structures.

Two common similarity metrics are correlation and Euclidean distance. The latter is often popular, since it is intuitive, can be described by a familiar distance function, and satisfies the triangular inequality. Clustering methods that employ asymmetric distance measures [33,41] are probably more difficult to intuitively comprehend even though they may be highly suited to their intended applications. The earliest work on clustering emphasized visual interpretations for the ease of study, resulting in methods that utilize dendograms and color maps [5]. Other examples of clustering algorithms include: (a) Single-Link and Complete-Link Hierarchical Clustering [27,49], (b) K-Means Algorithm and its family of variants, such as the K-Medians [21,34,37,60,61], (c) Reformulation Linearization-based Clustering [1,46], (d) Fuzzy Clustering [3,9,44,47], (e) Quality Cluster Algorithm (QTClust) [23], (f) Graph-Theoretic Clustering [17,57,59], (g) Mixture-Resolving Clustering Method [7,26], (h) Mode Seeking Algorithms [26], (i) Artificial Neural Networks for Clustering [4,31] such as the Self-Organizing Map (SOM) [32] and a variant that combines the SOM with hierarchical clustering, the Self-Organizing Tree Algorithm (SOTA) [22] (j) Information-Based Clustering [8,48,54], (k) Stochastic Approaches [30,36,38]. Some of these methods, such as the K-Means and Information Clustering, are optimization-based approaches, in which the clustering is represented as an unknown parameter vector of a cost function. The process then seeks to obtain the best clustering by minimizing this cost function. Other classes of clustering methods such as competitive learning may not have a straightforward cost function. For instance,

in the SOM, cluster centers are arbitrarily chosen initially, after which random data points are selected and placed into the nearest cluster, whose center is updated accordingly after each selection. Clustering ceases when the cluster centers become stationary.

Recently, Tan et al. [51,52] presents a novel optimization-based Mixed-Integer Nonlinear Programming (MINLP) clustering algorithm, the Global Optimal Search with Enhanced Positioning (EP_GOS_Clust), which is robust yet intuitive. This algorithm is significant in that it is able to progressively identify and weed out outlier data points. In addition, it involves a pre-clustering process that is rigorous and has a clearly-defined decision criterion. This is notable as the results of many clustering methods based on function optimization schemes often vary depending on the random initialization or starting heuristics. The EP_GOS_Clust also contains a convenient method to predict the optimal cluster number. The algorithm is compared with several approaches commonly used in clustering biological microarray data, namely K-methods, QTClust, SOM, and SOTA. By comparing the intra-cluster and inter-cluster error sums, as well as the strength of biological coherence based on Gene Ontology resources and expression pattern correlation, the EP_GOS_Clust is shown to compare favorably against other methods. The following sections will describe this novel clustering approach in more detail.

Formulations

Notation and Pre-Clustering

The measure of distance for a gene i, for $i = 1, \ldots, n$ having k features (or dimensions), for $k = 1, \ldots, s$ is defined as a_{ik}. Each gene is to be assigned to only one (hard clustering) of c possible clusters, each with center z_{jk}, for $j = 1, \ldots, c$. The binary variables w_{ij} indicates whether gene i falls within cluster j ($w_{ij} = 1$, if yes; $w_{ij} = 0$, if no).

Pre-clustering the data is important to expedite the computational resources required to solve the hard clustering problem by (i) identifying genes with similar experimental responses, and (ii) removing outliers deemed not to be significant to the clustering process. A straightforward pre-clustering approach to provide just the adequate amount of discriminatory characteristics so that the genes can be pre-clustered properly is to reduce the quantities represented in the k-dimensional expression vectors into a set of representative variables $\{+, o, -\}$. The $(+)$ variable represents an increase in expression level compared to the previous time point, the $(-)$ variable represents a decrease in expression level from the previous time point, and the (o) variable represents an expression level that does not vary significantly across the time points. The expression data can also be pre-clustered by creating a rank-ordered list of gene proximities based on Euclidean distance or correlation. Genes that demonstrate an obvious level of proximity, such as a separation of only at most 1% of the maximum inter-gene distances, are then grouped together. The pre-clusters are the proximity genes that form a complete clique, that is, there is a link between every gene within the same pre-cluster. With this choice, a maximal clique search can be performed by using various levels of pre-clustering criteria. Clearly, when the criterion is overly lenient, a large number of pre-clusters are formed, but most of the genes will belong to multiple pre-clusters, and the number of maximal cliques formed is small. On the other hand, an unnecessarily strict cut-off results in a small number of pre-clusters, thus not accurately reflecting the extent of relatedness between the data. In pre-clustering over a range of cut-off values, we can then select the appropriate criterion as the point where the maximum number of complete cliques is formed [53].

Hard Clustering by Global Optimization The global optimization approach seeks to minimize the Euclidean distances between the data points and the centers of their assigned clusters as:

$$\begin{aligned} \underset{w_{ij}, z_{jk}}{\text{Minimize}} \quad & \sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{s} w_{ij} \left(a_{ik} - z_{jk}\right)^2 \\ \text{s.t.} \quad & \sum_{j=1}^{c} w_{ij} = 1\,, \quad \forall i = 1, \ldots, n \\ & w_{ij} \text{ are binary variables, } z_{jk} \text{ are} \\ & \text{continuous variables .} \end{aligned} \tag{Problem 1}$$

There are two sets of variables in the problem, w_{ij} and z_{jk}. While the bounds of w_{ij} are clearly 0 and 1, that of

z_{jk} is obtained by observing the range of a_{ik} values.

$$z_{jk}^{L} = \min\{a_{ik}\}\ , \quad \forall k = 1, \ldots, s$$
$$z_{jk}^{U} = \max\{a_{ik}\}\ , \quad \forall k = 1, \ldots, s\ .$$

The pre-clustering work suggests that some of the genes need only be restricted to some number of known clusters, since it can be determined (for instance by distance and correlation metrics) that certain genes are exceedingly dissimilar from some of the pre-clusters and thus have virtually zero probability of being clustered there. This restriction can be described by introducing an additional binary parameter suit_{ij}. A data point deemed to belong uniquely to just one cluster will only have $\text{suit}_{ij} = 1$ for only one value of j and zero for the others, whereas a data point restricted to a few clusters will have $\text{suit}_{ij} = 1$ for only those clusters. This reduces the computational demands of the problem. The introduction of the suit_{ij} parameters also obviates the need for constraints that prevent the redundant re-indexing of clusters. Together with the necessary first-order optimality condition (i. e., the vector distance sum of all genes within a cluster to the cluster center should be intuitively zero), the formulation becomes:

$$\begin{aligned}
\underset{w_{ij}, z_{jk}}{\text{Minimize}} \quad & \sum_{i=1}^{n}\sum_{k=1}^{s} a_{ik}^2 \\
& - \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} (\text{suit}_{ij})(a_{ik} w_{ij} z_{jk}) \\
\text{s.t.} \quad & (\text{suit}_{ij})\left(z_{jk} \sum_{i=1}^{n} w_{ij} - \sum_{i=1}^{n} a_{ik} w_{ij} \right) \\
& = 0\ , \quad \forall j, \forall k \\
& \sum_{j=1}^{c} (\text{suit}_{ij}) w_{ij} = 1\ , \quad \forall i \\
& 1 \le \sum_{j=1}^{n} (\text{suit}_{ij}) w_{ij} \le n - c + 1 \\
& w_{ij} = 0 - 1\ , \quad \forall i, \forall j \\
& z_{jk}^{L} \le z_{jk} \le z_{jk}^{U}\ , \quad \forall j, \forall k\ .
\end{aligned}$$

(Problem 2)

The first set of constraints are the necessary optimality conditions, the second demand that each gene can belong to only one cluster, and the third state that there is at least one and no more than $(n - c + 1)$ data points in a cluster. Note also that the $\sum_{i=1}^{n}\sum_{k=1}^{s} a_{ik}^2$ term in the objective function of Problem 2 is a constant and can be dropped, though for the sake of completeness we will retain the term throughout the subsequent formulations in the paper. Problems 1 and 2 are Mixed Integer Nonlinear Programming (MINLP) problems with bilinear terms in the objective function and the first set of constraints. To handle the nonlinearities formed by the product of variables w_{ij} and z_{jk}, new variables y_{ijk} along with additional constraints [12] are defined as follows:

$$y_{ijk} = w_{ij} z_{jk} \tag{1}$$

$$z_{jk} - z_{jk}^{U}\left(1 - w_{ij}\right) \le y_{ijk} \le z_{jk} - z_{jk}^{L}\left(1 - w_{ij}\right) \tag{2}$$

$$z_{jk}^{L} w_{ij} \le y_{ijk} \le z_{jk}^{U} w_{ij}, \quad \forall i, \forall j, \forall k\ . \tag{3}$$

The introduction of y_{ijk} and the additional constraints reduces the formulation to an equivalent Mixed-Integer Linear Programming (MILP) problem, but results in an inordinately large number of variables. Thus, there is a need for new approaches to address large datasets.

The GOS Algorithm for Clustering The introduction of the bilinear variable y_{ijk} results in a large number of variables to be considered. In a problem with over 2000 data points, each having 24 features, to be placed into over 380 clusters, the number of variables to be considered numbers over 18 million. Without introducing the y_{ijk} variables will leave the problem in a nonlinear form. Mixed-integer nonlinear programming (MINLP) problems are considered extremely difficult. Theoretical advances and prominent algorithms for solving MINLP problems are addressed in [12,13,15].

The MINLP clustering formulation described in Problem 2 can be solved by a variant of the Generalized Benders Decomposition (GBD) algorithm [14], denoted as the Global Optimum Search (GOS). The primal problem results from fixing the binary variables to a particular 0-1 combination. Here, w_{ij} is fixed and z_{jk} is solved from the resultant linear programming (LP) problem. In addition, the solution also includes the relevant Lagrange multipliers. The master problem is essentially the problem projected onto the y-space (i. e., that of the binary variables). To expedite the solution of this projection, the dual representation of the mas-

ter is used. This dual representation is in terms of the supporting Lagrange functions of the projected problem. It is assumed that the optimal solution of the primal problem as well as its Lagrange multipliers can be used for the determination of the support function. In the master problem, the z_{jk} solution from the accompanying primal is taken and the master is solved for the w_{ij} variables.

The two sequences of upper and lower bounds are then iteratively updated until they converge in a finite number of iterations. With each successive iteration, a new support function is added to the list of constraints for the master problem. Thus in a sense, the support functions for the master problem build up with each iteration, forming a progressively tighter envelope and gradually pushing up the lower bound solution until it converges with the upper bound solution.

With fixed starting values for w_{ij}, the primal problem becomes:

$$\begin{aligned}
\underset{z_{jk}}{\text{Minimize}} \quad & \sum_{i=1}^{n}\sum_{k=1}^{s} a_{ik}^2 - \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} a_{ik} w_{ij}^* z_{jk} \\
\text{s.t.} \quad & z_{jk}\sum_{i=1}^{n} w_{ij}^* - \sum_{i=1}^{n} a_{ik} w_{ij}^* = 0\,, \quad \forall j, \forall k \\
& z_{jk}^L \le z_{jk} \le z_{jk}^U\,, \quad \forall j, \forall k\,.
\end{aligned}$$

(Problem 3.1)

The primal problem is a Linear Programming (LP) problem. All the other constraints drop out since they do not involve z_{jk}, which are the variables to be solved in the primal problem. Besides z_{jk}, the Lagrange multipliers λ_{jk}^m for each of the constraints above is obtained. The objective function is the upper bound solution. These are inputted into the master problem, which becomes:

$$\begin{aligned}
\min_{w_{ij},\mu_B} \ & \mu_B \\
\text{such that} \quad & \mu_B \ge \sum_{i=1}^{n}\sum_{k=1}^{s} a_{ik}^2 - \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} a_{ik} w_{ij} z_{jk}^* \\
& + \sum_{j=1}^{c}\sum_{k=1}^{s} \lambda_{jk}^{m*} \left(z_{jk}^* \sum_{i=1}^{n} w_{ij} \right. \\
& \left. - \sum_{i=1}^{n} a_{ik} w_{ij} \right)\,, \quad m = 1, M
\end{aligned}$$

$$\begin{aligned}
& \sum_{j=1}^{c} w_{ij} = 1\,, \quad \forall i \\
& 1 \le \sum_{j=1}^{n} w_{ij} \le n - c + 1\,, \quad \forall j \\
& w_{ij} = 0 - 1\,, \quad \forall i, \forall j\,.
\end{aligned}$$

(Problem 3.2)

The master problem solves for w_{ij} and μ_B, and results in a lower bound solution (i. e., the objective function). The master problem is a Mixed Integer Linear Programming (MILP) problem. The w_{ij} solutions are cycled back into the primal problem and the process is repeated until the solution converges. Thus, there is no longer a need for the variables y_{ijk}, which substantially reduces the number of variables to be solved. Also, after every solution of the master problem, where a solution set for w_{ij} is generated, an integer cut is added for subsequent iterations to prevent redundantly considering that particular solution set again. The cut is expressed as:

$$\sum_{i \in \{n | w_{ij}=1\}}^{n} w_{ij} - \sum_{i \in \{n | w_{ij}=0\}}^{n} w_{ij} \le n - 1\,. \tag{4}$$

Determining the Optimal Number of Clusters Most clustering algorithms do not contain screening functions to determine the optimal number of clusters. Yet this is important to evaluate the results of cluster analysis in a quantitative and objective fashion. On the other hand, while it is relatively easy to propose indices of cluster validity, it is difficult to incorporate these measures into clustering algorithms and appoint thresholds on which to define key decision values [18,27]. Some of the indices used to compute cluster validity include the Dunn's validity index [10], the Davis–Bouldin validity index [6], the Silhouette validation technique [43], the C index [24], the Goodman–Kruskal index [16], the Isolation index [39], the Jaccard index [25], and the Rand index [42]. We note that the optimal number of clusters occurs when the inter-cluster distance is maximized and the intra-cluster distance is minimized. We adapt the concept of a clustering balance [29], where it has been shown to have a minimum value when intra-cluster similarity is maximized and inter-cluster similarity is minimized. This provides a measure of how optimal is a certain number of clusters used for a partic-

ular clustering algorithm. We introduce the following:

$$\text{Global Center, } z_k^o = \frac{1}{n}\sum_{i=1}^{n} a_{ik}\,, \quad \forall k \tag{5}$$

Intra-cluster error sum,

$$\Lambda = \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} w_{ij}\left\|a_{ik} - z_{jk}\right\|_2^2 \tag{6}$$

Inter-cluster error sum,

$$\Gamma = \sum_{j=1}^{c}\sum_{k=1}^{s}\left\|z_{jk} - z_k^o\right\|_2^2\,. \tag{7}$$

Jung et al. [29] proposed a clustering balance parameter, which is the α-weighted sum of the two error sums.

$$\text{Clustering Balance, } \varepsilon = \alpha\Lambda + (1-\alpha)\,\Gamma\,. \tag{8}$$

We note here that the rightful α-ratio is 0.5. There are two ways to come to this conclusion. We note that the factor α should balance the contributive weights of the two error sums to the clustering balance. At extreme cluster numbers, that is, the largest and smallest number possible, the sum of the intra-cluster and inter-cluster error sums at both cluster numbers should be balanced. In the minimal case, all the data points can be placed into a single cluster, in the case of which the inter-cluster error sum is zero and the intra-cluster error sum can be calculated with ease. In the maximal case, each data point forms its own cluster, in the case of which the intra-cluster error sum is zero and the inter-cluster error sum can be easily found. Obviously the intra-cluster error sum in the minimal case and inter-cluster error sum in the maximal case are equal, suggesting that the most appropriate weighting factor to use is in fact 0.5. The second approach uses a clustering gain parameter proposed by Jung et al. [29], which is given by:

$$\Delta = \sum_{j=1}^{c}\sum_{k=1}^{s}\left(n_j - 1\right)\left\|z_k^o - z_{jk}\right\|_2^2\,. \tag{9}$$

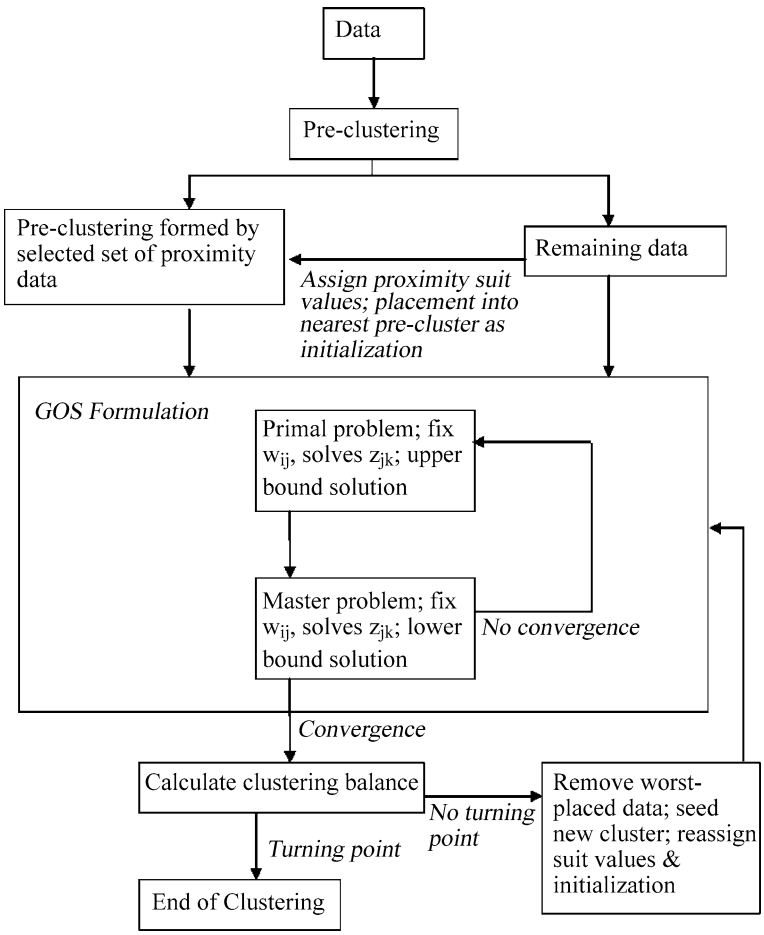
Jung et al. [29] showed the clustering gain to have a maximum value at the optimal number of clusters, and demonstrated that the sum total of the clustering gain and balance parameters is a constant. This is only shown to be only possible if the α-ratio is 0.5 [51]. These derivations suggest that for any clustering algorithm including that using the GOS algorithm, one can deduce the optimal number of clusters by performing multiple repetitions of the clustering process over a suitably large range of cluster numbers and watching for the clustering gain or clustering balance turning points.

Proposed Algorithm

The GOS formulation appears to be a suitable clustering algorithm. But for it to be effective, the formulation must be provided with a good initialization point. Also, we want to expeditiously incorporate the approach to predict the optimal number of clusters into a clustering algorithm. With these considerations in mind, we propose the following GOS clustering algorithm with enhanced data point positioning (EP_GOS_Clust).

Gene Pre-Clustering We pre-cluster the original data by proximity studies to reduce the computational demands by (i) identifying genes with very similar responses, and (ii) removing outliers deemed to be insignificant to the clustering process. To provide just adequate discriminatory characteristics, pre-clustering can be done by reducing the expression vectors into a set of representative variables or by pre-grouping genes that are close to one another by correlation or some other distance function.

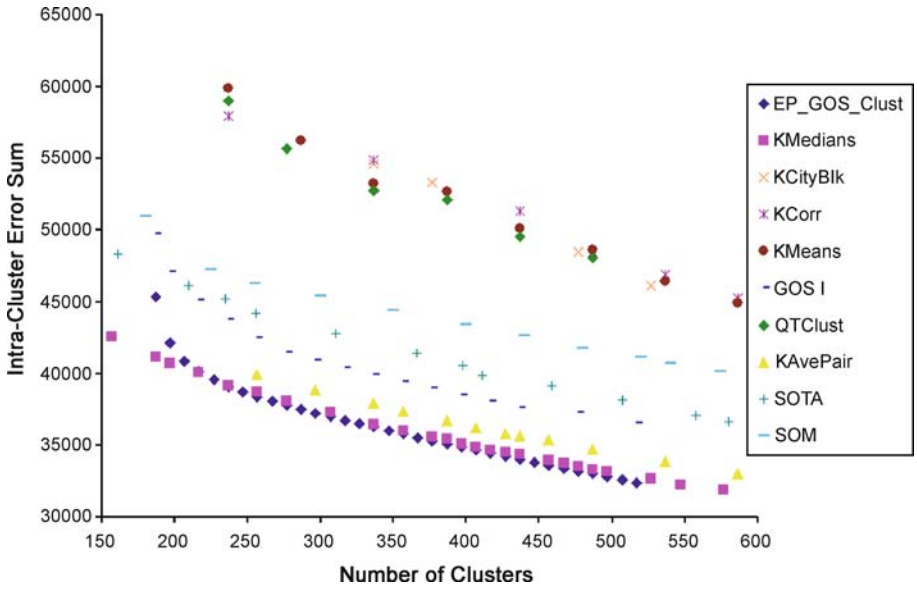
Iterative Clustering We let the initial clusters be defined by the genes pre-clustered previously, and find the distance between each of the remaining genes and these initial clusters and as a good initialization point placed these genes into the nearest cluster. For each gene, we allow its suitability in a limited number of clusters based on the proximity study. In the primal problem of the GOS algorithm, we solve for z_{jk}. These, together with the Lagrange multipliers, are used in the master problem to solve for w_{ij}. The primal gives an upper bound solution and the master a lower bound. The optimal solution is obtained when both bounds converge. Then, the worst-placed gene is removed and used as a seed for a new cluster. This gene has already been subjected to a membership search so there is no reason for it to belong to any one of the older clusters. The iterative steps are repeated and the clusters build up gradually until the optimal number is attained. Figure 1 shows a schematic of EP_GOS_Clust.



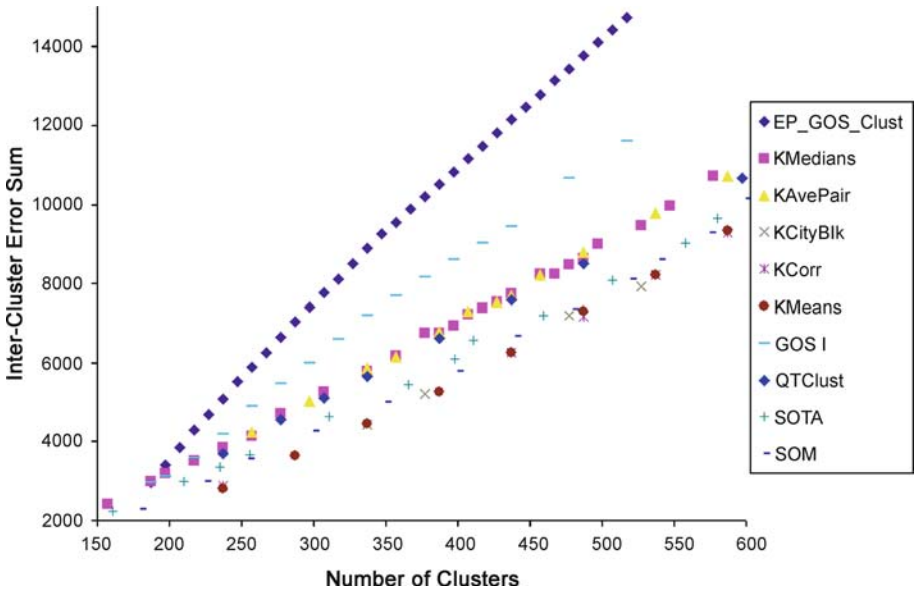
Gene Clustering: A Novel Decomposition-Based Clustering Approach, Figure 1
Schematic of EP_GOS_Clust algorithm

Gene Clustering: A Novel Decomposition-Based Clustering Approach, Table 1
Comparison of cluster correlation. The shaded row contains the results for EP_GOS_Clust and the top three performers in each column are marked with an asterisk

			Correlation coefficient			
		Optimal Cluster Number	Average	Maximum	Minimum	Standard deviation
Clustering Method	EP_GOS_Clust	237	0.617*	0.938*	0.264*	0.128*
	KMedians	445	0.615	0.937	0.197	0.134
	KCityBlk	665	0.398	0.760	-0.159	0.149
	KCorr	665	0.630*	0.931	0.239*	0.119*
	KMeans	775	0.614	0.959*	0.072	0.131
	GOS I	295	0.590	0.933	0.202	0.148
	KAvePair	452	0.567	0.909	0.156	0.141
	SOTA	540	0.604	0.925	0.378*	0.122*
	SOM	485	0.623*	0.968*	0.202	0.156



Gene Clustering: A Novel Decomposition-Based Clustering Approach, Figure 2
Intra-cluster error sum

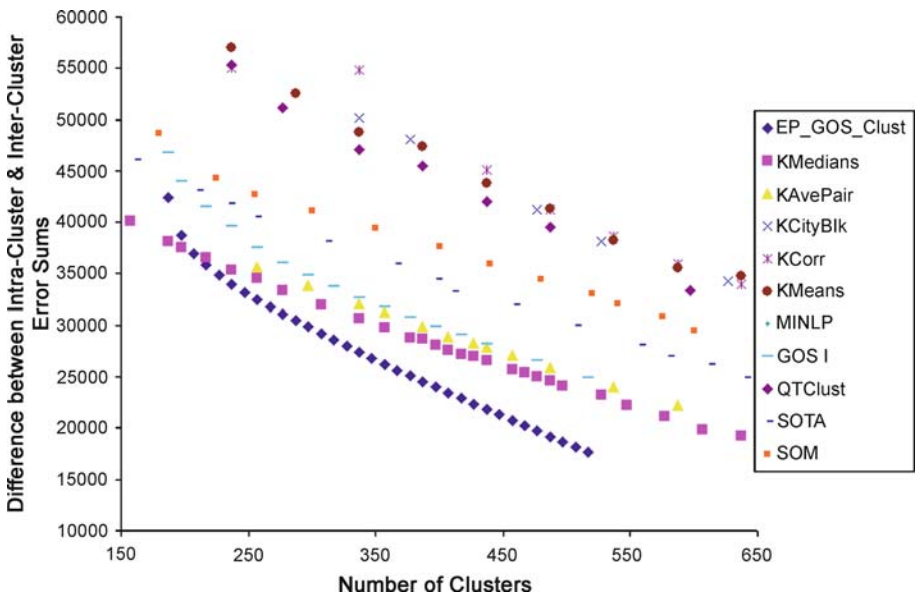


Gene Clustering: A Novel Decomposition-Based Clustering Approach, Figure 3
Inter-cluster error sum

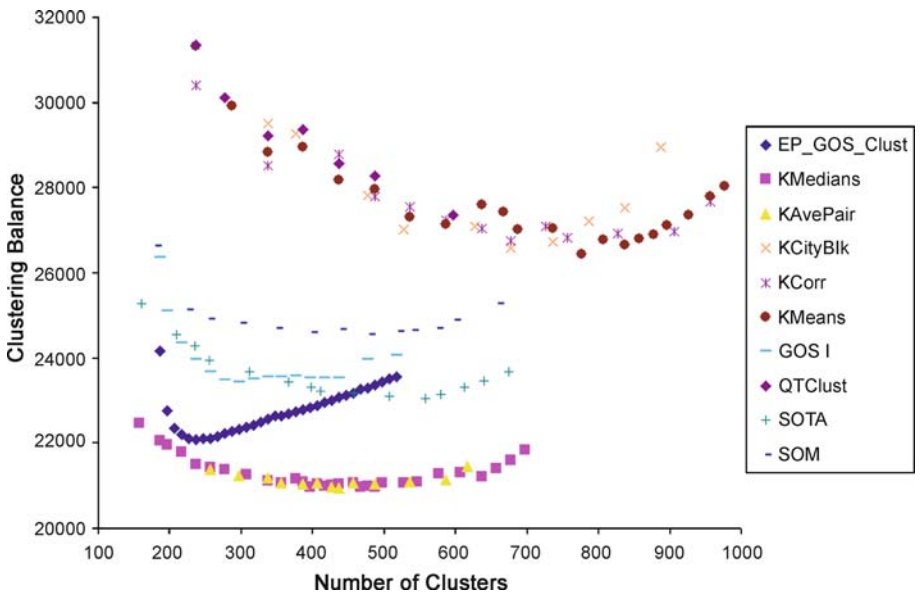
Case Study

Experimental Data

As a study, we use experimental microarray data derived from a study in the role of the Ras/protein kinase A pathway (PKA) on glucose signaling in yeast [56]. These experiments analyzed mRNA levels in samples extracted from cells at various times following stimulation by glucose or following activation of either Ras2 or Gpa2, which are small GTPases involved in the metabolic and transcriptional response of yeast cells to glucose [45]. These experiments were performed in wild type cells and cells defective in PKA activity. Clustering these microarray data has proven to be a critical



Gene Clustering: A Novel Decomposition-Based Clustering Approach, Figure 4
Error sum difference



Gene Clustering: A Novel Decomposition-Based Clustering Approach, Figure 5
Optimal cluster number

step in using the data to develop a predictive model of a topological map of the signaling network surrounding the Ras/PKA pathway [35].

Levels of RNA for each of the 6237 yeast genes in each of the RNA samples from the above experiments were measured using Affymetrix microarray chips and analyzed by the Affymetrix software. We used the Affymetrix MicroArray Suite 5.0, which analyzes the consensus of intensities of hybridization of an RNA to the collection of perfect match probes for a gene on the array, relative to the intensities of hybridization to single mismatch probes, to further determine whether

Gene Clustering: A Novel Decomposition-Based Clustering Approach, Table 2
Gene Ontology comparison. The table compares the $-\log_{10}(P)$ values of the clusters, which reflect the level of annotative richness, as well as the proportion of yeast genes that fall into biologically significant clusters. The latter is important in 'presenting' the maximal amount of relevant genetic information for follow-up work in areas such as motif recognition and regulatory network studies

		$-\log_{10}(P)$ Comparison			
		Average	Standard deviation	In clusters with $-\log_{10}(P)$ values ≥ 4	In clusters with $-\log_{10}(P)$ values ≥ 3
Clustering Method	EP_GOS_Clust	4.40*	0.37	32.82*	64.92*
	KMedians	4.27*	0.34*	30.83*	62.23*
	KCityBlk	3.69	0.49	27.53	56.68
	KCorr	4.15*	0.39	32.59*	60.08*
	KMeans	3.45	0.41	25.11	55.20
	GOS I	3.84	0.42	28.19	57.75
	KAvePair	3.77	0.48	25.18	54.43
	SOTA	3.67	0.31*	30.20	58.86
	SOM	3.94	0.35*	30.47	59.24

a signal for a specific RNA in a sample was reliable (P or present), unreliably low (A or absent), or ambiguous (M). Before clustering the array data, we filtered the data to remove unreliable data. In particular, we retained all genes for which all the time points were present (4105 genes), all the genes for which greater than 50% of the time points were present, and all the genes for which the present/absent calls exhibited a biologically relevant pattern (e. g. PAAA for the four time points in the experiment, suggesting repression of gene expression over the course of the experiment). In all, we retained 5652 genes.

Description of Comparative Study

The clustering algorithms to be compared are (a) K-Means, (b) K-Medians, (c) K-Corr, where the Pearson correlation coefficient is the distance metric, (d) K-CityBlock, where the distance metric is the city block distance, or the 'Manhattan' metric, which is akin to the north-south or east-west walking distance in a place like New York's Manhattan district, (e) K-AvePair, where the cluster metric is the average pair-wise distance between members in each cluster, (f) QTClust, (g) SOM, (h) SOTA, (i) GOS I, where genes with up to 7 different feature points are pre-clustered, initial clusters are defined by uniquely-placed genes, and each gene is placed into its nearest cluster as the initialization point, and (j) EP_GOS_Clust, for which genes are pre-clustered if they have 2 or less different feature points and can be uniquely clustered. Since the K-family approaches are sensitive to the initialization point, we run each 25 times and use only the best result.

Results and Discussion

A good clustering procedure should minimize the intra-cluster error sum and maximize the inter-cluster error sum. We look also at the difference between error sums, which is somewhat indicative of the efficacy of a particular clustering algorithm, since methods using intra-cluster error sum as the cost function would probably outperform methods using inter-cluster error sum as a performance indicator. From Fig. 2, 3 and 4, we can see that EP_GOS_Clust compares very favorably compared to the other clustering algorithms. Also, as seen from Fig. 5, EP_GOS_Clust predicts the lowest number of optimal clusters. Together with the quality of the error sum comparisons, we infer the superior 'economy' of EP_GOS_Clust in producing tighter data groupings by utilizing a lower number of clusters, as it is actually possible to achieve tight groupings by using a large number of clusters, even with an inferior clustering algorithm.

EP_GOS_Clust is also capable of uncovering strongly correlated clusters with high levels of biological coherence. Tables 1 and 2 shows that it performs consistently well when compared against the significance of cluster biological coherence uncovered by the other clustering methods. We find our clusters to ex-

hibit good correlation and a high level of functional coherence strength across all cluster sizes, which indicates that EP_GOS_Clust shows good consistency and lack of size-bias. Also, it can be seen that EP_GOS_Clust compares very well with other clustering methods in producing highly correlated clusters, even against methods such as K-Corr that already explicitly uses correlation as a metric for clustering and the correlation hunting SOM. In addition, EP_GOS_Clust conveniently isolates errant data points and refines the existing groupings as the clustering progresses.

References

1. Adams WP, Sherali HD (1990) Linearization Strategies for a Class of Zero-One Mixed Integer Programming Problems. Oper Res 38(2):217–226
2. Beer M, Tavazoie S (2004) Predicting Gene Expression from Sequence. Cell 117:185–198
3. Bezdek JC (1981) Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York
4. Carpenter G, Grossberg S (1990) ART3: Hierarchical Search using Chemical Transmitters in Self-Organizing Patterns Recognition Architectures. Neural Netw 3:129–152
5. Claverie J (1999) Computational Methods for the Identification of Differential and Coordinated Gene Expression. Hum Mol Genet 8:1821–1832
6. Davis DL, Bouldin DW (1979) A Cluster Separation Measure. IEEE Trans Pattern Anal Mach Intell 1(4):224–227
7. Dempster AP, Laird NM, Rudin DB (1977) Maximum Likelihood from Incomplete Data via the EM Algorithm. J Royal Stat Soc B 39(1):1–38
8. Dhillon IS, Guan Y (2003) Information Theoretic Clustering of Sparse Co-Occurrence Data. In: Proceedings of the Third IEEE International Conference on Data Mining (ICDM), Melbourbe, November 2003
9. Dunn JC (1973) A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. J Cybern 3:32–57
10. Dunn JC (1974) Well Separated Clusters and Optimal Fuzzy Partitions. J Cybern 4:95–104
11. Duran MA, Odell PL (1974) Cluster Analysis: A Survey. Springer, New York
12. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, Oxford
13. Floudas CA (2000) Deterministic Global Optimization: Theory, Algorithms, and Applications. Kluwer, Dordrecht
14. Floudas CA, Aggarwal A, Ciric AR (1989) Global Optimum Search for Non Convex NLP and MINLP Problems. Comp Chem Eng 13(10):1117–1132
15. Floudas CA, Akrotirianakis IG, Caratzoulas S, Meyer CA, Kallrath J (2005) Global Optimization in the 21st Century: Advances and Challenges. Comput Chem Eng 29:1185–2002
16. Goodman L, Kruskal W (1954) Measures of Associations for Cross-Validations. J Am Stat Assoc 49:732–764
17. Gower JC, Ross GJS (1969) Minimum Spanning Trees and Single-Linkage Cluster Analysis. Appl Stat 18:54–64
18. Halkidi M, Batistakis Y, Vazirgiannis M (2002) Cluster Validity Methods: Part 1. SIGMOD Rec 31(2):40–45
19. Hansen P, Jaumard B (1997) Cluster Analysis and Mathematical Programming. Math Program 79:191–215
20. Hartigan JA (1975) Clustering Algorithms. Wiley, New York
21. Hartigan JA, Wong MA (1979) Algorithm AS 136: A K-Means Clustering Algorithm. Appl Stat-J Roy St C 28:100–108
22. Herrero J, Valencia A, Dopazo J (2001) A Hierarchical Unsupervised Growing Neural Network for Clustering Gene Expression Patterns. Bioinformatics 17(2):126–136
23. Heyer LJ, Kruglyak S, Yooseph S (1999) Exploring Expression Data: Identification and Analysis of Co-Expressed Genes. Genome Res 9:1106–1115
24. Hubert L, Schultz J (1976) Quadratic Assignment as a General Data-Analysis Strategy. Br J Math Stat Psychol 29:190–241
25. Jaccard P (1912) The Distribution of Flora in the Alpine Zone. New Phytol 11:37–50
26. Jain AK, Murty MN, Flynn PJ (1999) Data Clustering: A Review. ACM Comput Surv 31(3):264–323
27. Jain AK, Dubes RC (1988) Algorithms for Clustering Data. In: Prentice-Hall Advanced Reference Series. Prentice, New Jersey.
28. Johnson RE (2001) The Role of Cluster Analysis in Assessing Comparability under the US Transfer Pricing Regulations. Bus Econ
29. Jung Y, Park H, Du D, Drake BL (2003) A Decision Criterion for the Optimal Number of Clusters in Hierarchical Clustering. J Glob Optim 25:91–111
30. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by Simulated Annealing. Science 220(4598):671–680
31. Kohonen T (1989) Self Organization and Associative Memory. In: Springer Information Science Series. Springer, New York
32. Kohonen T (1997) Self-Organizing Maps. Springer, Berlin
33. Leisch F, Weingessel A, Dimitriadou E (1998) Competitive Learning for Binary Valued Data. In: Niklasson L, Bod'en M, Ziemke T (eds) Proceedings of the 8th International Conference on Artificial Neural Networks (ICANN 98) vol 2. Springer, Skövde, pp 779–784
34. Likas A, Vlassis N, Vebeek JL (2003) The Global K-Means Clustering Algorithm. Pattern Recognit 36:451–461
35. Lin X, Floudas C, Wang Y, Broach JR (2003) Theoretical and Computational Studies of the Glucose Signaling Pathways in Yeast Using Global Gene Expression Data. Biotechnol Bioeng 84(7):864–886
36. Lukashin AV, Fuchs R (2001) Analysis of Temporal Gene Expression Profiles: Clustering by Simulated Annealing and

Determining the Optimal Number of Clusters. Bioinform 17(5):405–414
37. McQueen J (1967) Some Methods for Classification and Analysis of Multivariate Observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, January 1966. University of California, Berkely, pp 281–297
38. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller EJ (1953) Equations of state calculations by fast computing machines. J Chem Phys 21:1087
39. Pardalos PM, Boginski V, Vazakopoulos A (Co-Ed.) (2007) Data Mining in Biomedicine. Springer, Berlin
40. Pauwels EJ, Fregerix G (1999) Finding Salient Regions in Images: Non-parametric Clustering for Image Segmentation and Grouping. Comput Vis Image Underst 75:73–85
41. Pipenbacher P, Schliep A, Schneckener S, Schonhuth A, Schomburg D, Schrader R (2002) ProClust: Improved Clustering of Protein Sequences with an Extended Graph-Based Approach. Bioinform 18(Supplement 2):S182–191
42. Rand WM (1971) Objective Criteria for the Evaluation of Clustering Methods. J Am Stat Assoc 846–850
43. Rousseeuw PJ (1987) Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. J Comp Appl Math 20:53–65
44. Ruspini EH (1969) A New Approach to Clustering. Inf Control 15:22–32
45. Schneper L, Düvel K, Broach JR (2004) Sense and Sensibility: Nutritional Response and Signal Integration in Yeast. Curr Opin Microbiol 7(6):624–630
46. Sherali HD, Desai J (2005) A Global Optimization RLT-Based Approach for Solving the Hard Clustering Problem. J Glob Optim 32(2):281–306
47. Sherali HD, Desai J (2005) A Global Optimization RLT-Based Approach for Solving the Fuzzy Clustering Approach. J Glob Optim 33(4):597–615
48. Slonim N, Atwal GS, Tkačik G, Bialek W (2005) Information Based Clustering. Proc Natl Acad Sci USA 102(51):18297–18302
49. Sokal RR, Michener CD (1958) A Statistical Method for Evaluating Systematic Relationships. Univ Kans Sci Bull 38:1409–1438
50. Sorlie T, Tibshirani R, Parker J, Hastie T, Marron JS, Nobel A, Deng S, Johnsen H, Pesich R, Geisler S, Demeter J, Perou CM, Lonning PE, Brown PO, Borresen-Dala AL, Botstein D (2003) Repeated Observations of Breast Tumor Subtypes in Independent Gene Expression Data Sets. Proc Natl Acad Sci USA 100:8418–8423
51. Tan MP, Broach JR, Floudas CA (2007) A Novel Clustering Approach and Prediction of Optimal Number of Clusters: Global Optimum Search with Enhanced Positioning. J Glob Optim 39:323–346
52. Tan MP, Broach JR, Floudas CA (2007) Evaluation of Normalization and Pre-Clustering Issues in a Novel Clustering Approach: Global Optimum Search with Enhanced Positioning. J Bioinform Comput Biol 5(4):895–913
53. Tan MP, Broach JR, Floudas CA (2007) Microarray Data Mining: A Novel Optimization-Based Iterative Clustering Approach to Uncover Biologically Coherent Structures. (submitted for publication)
54. Tishby N, Pereira F, Bialek W (1999) The Information Bottleneck Method. In: Proceedings of the 37th Annual Allerton Conference on Communication, Monticello, September 1999. Control and Computing, pp 368–377
55. Troyanskaya OG, Dolinski K, Owen AB, Altman RB, Botstein D (2003) A Bayesian Framework for Combining Heterogeneous Data Sources for Gene Function Prediction (in Saccharomyces Cerevisiae). Proc Natl Acad Sci USA 100:8348–8353
56. Wang Y, Pierce M, Schneper L, Guldal CG, Zhang X, Tavazoie S, Broach JR (2004) Ras and Gpa2 Mediate One Branch of a Redundant Glucose Signaling Pathway in Yeast. PLoS Biol 2(5):610–622
57. Wu Z, Leahy R (1993) An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation. IEEE Trans Pattern Recognit Mach Intell 15(11):1101–1113
58. Xu R, Wunsch IID (2005) Survey of Clustering Algorithms. IEEE Trans Neural Netw 16(3):645–678
59. Zahn CT (1971) Graph Theoretical Methods for Detecting and Describing Gestalt Systems. IEEE Trans Comput C-20:68–86
60. Zhang B, Hsu M, Dayal U (1999) K-Harmonic Means – A Data Clustering Algorithm. Hewlett-Packard Research Laboratory Technical Report HPL-1999-124
61. Zhang B (2000) Generalized K-Harmonic Means: Boosting in Unsupervised Learning. Technical Report, Hewlett-Packard Research Laboratory

Generalizations of Interior Point Methods for the Linear Complementarity Problem

LAURA DI GIACOMO
Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università di Roma "La Sapienza", Rome, Italy

MSC2000: 90C33, 90C51, 65K10

Article Outline

Introduction
Definitions
Formulation
An Interior Point Reduction Algorithm to Solve the LCP
An Interior Point Potential Algorithm to Solve General LCPs

Methods and Applications
Models
An Interior Point Newton Method for the General LCP
Generalization of an Interior Point Reduction Algorithm to Solve General LCPs
Cases
Conclusions
See also
References

Introduction

Some methods are reviewed to solve the resulting complementarity problem and two novel algorithms are described. The use of complementarity problems provides more flexibility to solve optimization problems, as well as a number of other advantages [10].

The existence of a general solution procedure for the linear complementarity problem (LCP) permits the incorporation of this algorithm recursively in an optimization algorithm and so avoids the use of active set strategies to handle inequality constraints and the use of second-order information on the objective function. This is often beneficial in the presence of nonconvex functions [10].

There exist many traditional approaches to solve the LCP. An algorithm was formulated early for the solution of LCPs [5,6]. Later, it was shown that if a LCP has a solution, then there exists a linear program, which, for a suitable objective function, will have an optimal solution that is also a solution to the LCP [2]. This was further generalized [7,8,9] so that for certain classes of LCPs the problem could be specified and solved as a linear program. A characterization of LCP was formulated [11] showing the equivalence of its solution to a solution of an appropriate parametric linear program with one scalar parameter.

A number of interior point algorithms to solve the LCPs have been presented, such as an interior point potential reduction algorithm [4] with P-matrices, positive semidefinite matrices and skew-symmetric matrices, an interior point algorithm which uses the affine scaling algorithm, to solve nonconvex (indefinite or negative definite) quadratic programming problems [14]. A fully polynomial-time approximation algorithm for computing a solution of the LCP with row-sufficient matrices can also be formulated [15]. This algorithm is a fully polynomial-time approximation scheme for finding an ϵ-approximate stationary point of the general LCP.

Here we shall briefly describe some particular methods and indicate two extensions of these algorithms which apply to more general matrices.

Definitions

In this section some definitions will be given and they will be used in the next sections [3].

Definition 1 Given M, an $n \times n$ matrix, and q, an n-dimensional vector. Let N be the index set of the variables, i. e., $N = 1, 2, \ldots, n$; the formulation of the LCP, $LCP(q, M)$, is then as follows:

$$Mx + q \geq 0\,, \tag{1}$$

$$x \geq 0\,, \tag{2}$$

$$x^T(Mx + q) = 0\,. \tag{3}$$

Definition 2 A matrix $M \in R^{n\times n}$ is said to be a P-matrix (P_0-matrix) if all its principal minors are positive (nonnegative). The class of such matrices is denoted P (P_0).

Definition 3 A square matrix is called a Z-matrix if its off-diagonal entries are all nonpositive. A Z-matrix which is also a P-matrix (P_0-matrix) is called a K-matrix (K_0-matrix).

Definition 4 A matrix $M \in R^{n\times n}$ is said to be column-sufficient if it satisfies the implication

$$[z_i(Mz)_i \leq 0 \ \text{ for all } \ i] \rightarrow [z_i(Mz)_i = 0 \ \text{ for all } \ i]\,. \tag{4}$$

The matrix M is called row-sufficient if its transpose is column -sufficient. If M is both column-sufficient and row-sufficient, then it is called sufficient.

Definition 5 A square matrix M is a skew-symmetric matrix if its transpose is also its negative:

$$A^T = -A\,. \tag{5}$$

Definition 6 If M is a positive definite matrix, then there exists a vector z such that

$$Mz > 0\,, \quad z > 0 \tag{6}$$

Definition 7 If M is a positive semidefinite matrix, then there exists a vector z such that

$$Mz \geq 0\,, \quad z > 0\,. \tag{7}$$

Definition 8 A potential function is

$$P(x, \Omega) = n\log(c^T x) - \sum_{j=1}^{n} \log x_j, \quad x \in \text{int}(\chi_\Omega) \tag{8}$$

where $\text{int}(\chi_\Omega)$ indicates the interior of the set χ_Ω which is the set of all feasible solutions of the dual.

Formulation

The aim of this section is to describe two modern implementations with interior point methods. In the first subsection an interior reduction algorithm to solve the LCP is presented, with particular matrix classes, [4], while in the following subsection an interior point potential algorithm to solve the general LCP is presented.

An Interior Point Reduction Algorithm to Solve the LCP

There exist many interior point algorithms to solve LCPs. A particularly interesting approach is an interior point potential reduction algorithm for the LCP [4]. The complementarity problem is viewed as a minimization problem, where the objective function is the product of the solution vector x and the slack vector of the inequalities y.

The objective of the algorithm formulated is to find an ϵ-complementarity solution in time bounded by a polynomial in the input size. This algorithm is formulated to solve LCP(q,M) which will have a solution, such as when the matrix M is a P-matrix. It is then extended to matrices M which are only positive semidefinite and to skew-symmetric matrices.

Consider a LCP, that is, given a rational matrix $M \in R^{n\times n}$ and a rational vector $q \in R^n$, find vectors $x, y \in R^n$ such that

$$y = Mx + q\,, \tag{9}$$

$$x, y \geq 0\,, \tag{10}$$

$$x^T y = 0\,, \tag{11}$$

which can be regarded as a quadratic programming problem

$$\text{Minimize } x^T y \tag{12}$$

$$\text{subject to } y = Mx + q \tag{13}$$

$$x, y \geq 0\,. \tag{14}$$

Given the problem Eqs. (12)–(14) the aim is to find a point with $x^T y < \epsilon$ for a given $\epsilon > 0$.

The algorithm proceeds by iteratively reducing the potential function:

$$f(x, y) = \rho \ln(x^T y) - \sum_j \ln(x_j y_j)\,. \tag{15}$$

Apply a linear scaling transformation to make the coordinates of the current point all equal to 1 and then take a gradient step in the transformed space using the gradient of the transformed potential function. The step size can be determined either by the algorithm or by line search to minimize the value of the potential function. Finally transform the solution point back to the original space.

Consider the potential function Eq. (15) under scaling of x and y, given any feasible interior point (x^0,y^0) if the matrices X and Y are diagonal matrices with the elements on the diagonal given by the values of (x^0,y^0).

Define a linear transformation of the space by

$$\bar{x} = X^{-1}x, \quad \bar{y} = Y^{-1}y\,. \tag{16}$$

and let $W = XY$, $w_j = (x_j^0)^T(y_j^0)$ so that ($w = w_1$, $w_2, \cdots, w_n$) and $\overline{M} = Y^{-1}MX$. Consider the transformed problem as follows:

$$\text{Minimize } \bar{x}^T W \bar{y} \tag{17}$$

$$\text{subject to } \bar{y} = \bar{M}\bar{x} + \bar{q} \tag{18}$$

$$\bar{x}, \bar{y} \geq 0\,. \tag{19}$$

Feasible solutions of the original problem are mapped into feasible solutions of the transformed problem:

$$\bar{y} = Y^{-1}(Mx + q) = \bar{M}\bar{x} + \bar{q}\,. \tag{20}$$

Assume that the current point is indeed (e,e) and the potential function has the form

$$f(\overline{x},\overline{y}) = \rho \ln(\overline{x}^T W \overline{y}) - \sum_{j=1}^{n} \ln(\overline{x}_j w_j \overline{y}_j)\,. \tag{21}$$

The gradient of f is given by

$$\nabla_x f(x,y) = \frac{\rho}{x^T W y} W y - X^{-1} e\,, \tag{22}$$

$$\nabla_y f(x,y) = \frac{\rho}{x^T W y} W x - Y^{-1} e\,, \tag{23}$$

and indicate by g the gradient vector evaluated at the current point (e,e).

Denote by $(\Delta x, \Delta y)$ the projection of $\nabla f(e,e)$ on the linear space Ω defined by $\Delta y = M \Delta x$.

Thus we define the following problem:

$$\text{Minimize } \|\Delta x - g\|^2 + \|\Delta y - g\|^2 \tag{24}$$

$$\text{subject to } \Delta y = M \Delta x\,. \tag{25}$$

It follows that [4]

$$\Delta x = (I + M^T M)^{-1}(I + M^T) g\,, \tag{26}$$

$$\Delta y = M(I + M^T M)^{-1}(I + M^T) g\,. \tag{27}$$

It is possible determine the reduction Δf in the value of f in moving from $x = y = e$ to a point of the form $\tilde{x} = e - t\Delta x$, $\tilde{y} = e - t\Delta y$, where $t > 0$. It is desired to choose t so as to achieve a reduction of at least n^{-k} for some $k > 0$, at every iteration. Since this is shown to be possible, [4], the result follows if the matrix is positive definite, positive semidefinite or skew-symmetric.

An Interior Point Potential Algorithm to Solve General LCPs

In this subsection a "condition-based" iteration complexity will be formulated regarding the solution of various LCPs. This parameter will characterize the degree of difficulty of the problem when a potential reduction algorithm is used. The condition number derived will of course depend on the data of the problem (M,q).

Consider the primal–dual potential function of a LCP as stated in Eqs: (9)–(11), for any interior feasible point, $(x, y) \in F$, and $\rho > 0$, which may be represented so:

$$\Psi(x,y) = \Psi_{n+\rho}(x,y) = (n+\rho)\ln(x^T y) - \sum_{j=1}^{n} \ln(x_j y_j). \tag{28}$$

Suppose the iterations have started from an interior feasible point (x_0,y_0), with $\Psi(x_0,y_0) = \Psi^0$ a sequence of interior feasible points can be generated $\{x^k, y^k\}, (k = 0, 1, \ldots)$ terminating at a point such that $(x^k)^T(y^k) \le \epsilon$. Such a point is found when

$$\Psi(x^k, y^k) \le \rho \ln(\epsilon) + n \ln(n) \tag{29}$$

since by the arithmetic–geometric inequality $n \ln((x^k)^T(y^k)) - \sum_{j=1}^{n} \ln(x_j y_j) \ge n \ln(n) \ge 0$.

The fact that $\Psi(x^T y) \le \Psi^0$ implies that $x^T y \le \Psi^0/\rho$ and therefore the boundedness of $\{(x,y) \in F \mid x^T y \le \Psi^0/\rho\}$ guarantees the boundedness of $\{(x,y) \in \text{int}(F) \mid x^T y \le \Psi^0\}$, where int() indicates the relative interior of its argument.

To obtain a reduction in the potential function the scaled gradient projection method may be used. The gradient vectors of the potential function with respect to x and y are

$$\nabla \Psi_x = (\frac{n+\rho}{x^T y}) y - X^{-1} e\,, \tag{30}$$

$$\nabla \Psi_y = (\frac{n+\rho}{x^T y}) x - Y^{-1} e\,. \tag{31}$$

At the kth iteration the following linear program is solved, subject to an ellipsoid constraint:

$$\text{Minimize } Z = \nabla^T \Psi_{x^k} d_x + \nabla^T \Psi_{y^k} d_y \tag{32}$$

$$\text{subject to } d_y = M d_x \tag{33}$$

$$1 > \alpha^2 \ge \|(X^k)^{-1} d_x\|^2 + \|(X^k)^{-1} d_x\|^2\,. \tag{34}$$

Denote by $(d_x^T, d_y^T)^T$ the minimal solution of Eqs. (32)–(34) and let

$$p^k = \begin{pmatrix} p_x^k \\ p_y^k \end{pmatrix} = \begin{pmatrix} \frac{n+\rho}{(x^k)^T(y^k)} X^k (y^k + M^T \pi) - e \\ \frac{n+\rho}{(x^k)^T(y^k)} Y^k (x^k - \pi) - e \end{pmatrix} \tag{35}$$

$$\pi = \left((Y^k)^2 + M(X^k)^2 M^T\right)^{-1} \left(Y^k - M X^k\right) \cdot \left(X^k y^k - \left(\frac{(x^k)^T(y^k)}{n+\rho}\right) e\right) \tag{36}$$

then there results

$$\begin{pmatrix}(X^k)^{-1}d_x\\(Y^k)^{-1}d_y\end{pmatrix} = \alpha\frac{p^k}{\|p^k\|}\,. \tag{37}$$

By the concavity of the log function and certain elementary results it can be shown [17] that

$$\Psi(x^k+d_x,y^k+d_y)-\Psi(x^k,y^k)\le -\alpha\|p^k\|+\frac{\alpha^2}{2}\left(n+\rho+\frac{1}{(1-\alpha)}\right)\,. \tag{38}$$

Letting

$$\alpha=\min\left\{\frac{||p^k||}{n+\rho+2},\frac{1}{n+\rho+2}\right\}\le\frac{1}{2} \tag{39}$$

results in

$$\Psi(x^k+d_x,y^k+d_y)-\Psi(x^k,y^k)\le -\min\left\{\frac{||p^k||^2}{(2n+\rho+2)},\frac{1}{2(n+\rho+2)}\right\}\,. \tag{40}$$

The expression for $\|p^k\|$ is indicated by (35) and can be considered the potential reduction at the kth iteration of the objective function. For any x,y let

$$g(x,y)=\frac{n+\rho}{x^Ty}Xy-e \tag{41}$$

$$H(x,y)=2I-(XM^T-Y)(Y^2+MX^2M^T)^{-1}(MX-Y) \tag{42}$$

which is a positive semidefinite matrix. Thus

$$\|p^k\|=g^T(x^k,y^k)H(x^k,y^k)g(x^k,y^k) \tag{43}$$

which may also be indicated as $\|g(x,y)\|_H^2=g^T(x,y)H(x,y)g(x,y)$.

Define a condition number for the LCP(q,M) as

$$\gamma(M,q,\epsilon)=\inf\{||g(x,y)||_H^2\mid x^Ty>\epsilon,\Psi(x,y)\le\Psi^0,(x,y)\in\mathrm{int}(F)\}\,. \tag{44}$$

The condition number $\gamma(M,q,\epsilon)$ represents the degree of difficulty for the potential reduction algorithm in solving the LCP(q,M). The larger the condition number that results, the easier can the problem be solved. The condition number for LCPs provides a criterion to subdivide given instances of LCP(q,M) into classes and those that can be solved in polynomial time may be indicated.

Corollary 1 *An instance of a LCP(q,M) is solvable in polynomial time if $\gamma(M,q,\epsilon)>0$ and $1/\gamma(M,q,\epsilon)$ is bounded above by a polynomial in* $\ln(1/\epsilon)$ *and n.*

This corollary is slightly different to corollary 1 in [16]. Further the following definitions are important:

$$\sum^{+}(M,q)=\{\pi\mid x^Ty-q^T\pi<0,x-\pi>0,\ y+M^T\pi>0\quad\text{for some}\quad(x,y)\in\mathrm{int}(F)\} \tag{45}$$

Definition 9 Let G be a set of LCP(q,M) such that the following conditions are satisfied:

$$G=\{(M,q)\mid\mathrm{int}(F)\ne\emptyset,\sum^{+}(M,q)=\emptyset\}\,. \tag{46}$$

Lemma 1 *Let $\sum^{+}(M,q)$ be empty for a LCP(q,M). Then for $\rho\ge n+\sqrt{2n}$, $\gamma(M,q,\epsilon)\ge 1$.*

Lemma 2 *Let $\{\pi\mid x^Ty-q^T\pi>0,x-\pi>0,y+M^T\pi>0$ for some $(x,y)\in int(F)\}$ be empty for a LCP(q,M). Then for $0<\rho\le n-\sqrt{(2n)}$, there results $\gamma(M,q,\epsilon)\ge 1$.*

With these properties it can be shown that for many classes of matrices $\gamma(M,q,\epsilon)>0$ or that the conditions indicated in the lemmas are satisfied, so the LCP is solvable in polynomial time.

Further, the potential reduction algorithm will solve, under general conditions, the LCP(q,M) when M is a $\mathcal{P}$-matrix and when M is a row-sufficient matrix. Thus,

Theorem 1 *Let $\Psi(x^0,y)\le O(n\ln(n))$ and M be a $\mathcal{P}$-matrix. Then the potential reduction algorithm terminates at $x^Ty<\epsilon$ in $O(n^2\max\{|\lambda|/\theta(n),1\}\ln(1/\epsilon))$ iterations and each iteration uses at most $O(n^3)$ arithmetic operations.*

The bound indicates that the algorithm is a polynomial-time algorithm if $|\lambda|/\theta(n)$ is bounded above by a polynomial in $\ln(1/\epsilon)$ and n.

Theorem 2 *Let $\rho>0$ and be fixed. For a row-sufficient matrix M and $\{(x,y)\in F\mid\Psi(x,y)\le\Psi^0\}$ bounded, then $\gamma(M,q,\epsilon)>0$.*

Since for the LCP(q,M) defined by this class of matrices the condition number is bounded away from zero,

the potential reduction algorithm will solve this class of problems.

Methods and Applications

Depending on the algorithm proposed, any penalty function algorithm or any linear programming algorithm will ensure, given the conditions imposed on the problem, a polynomial-time solution is achieved.

Often computationally, the most efficient method is the Newton method with a penalty or a barrier parameter. However, the actual method of solution is left to the interested reader, who can refer to the original contributions, since too many problem -dependent factors are involved.

Models

The aim of this section is to treat the methods described in "Formulation" under some more general conditions.

An Interior Point Newton Method for the General LCP

This algorithm finds a Karush–Kuhn-Tucker point for a nonmonotone LCP with a primal interior point method using Newton's method with a convex barrier function, under some mild assumptions.

Consider a bounded LCP:

$$Mu + q - v = 0 \tag{47}$$

$$u, v \geq 0 \tag{48}$$

$$u^T v = 0 \tag{49}$$

and suppose that the LCP solution set $S = \{u, v | Mu + q - v = 0, u, v \geq 0, u^T v = 0\}$ is bounded above by a vector $(m_1^T, m_2^T)^T \in R^{2n}$. Define two diagonal positive matrices

$$D_1 > Diag(2m_1) \tag{50}$$

$$D_2 > Diag(2m_2) \tag{51}$$

to obtain the following LCP

$$y = D_2^{-1} v = D_2^{-1}(Mu + q) = (D_2^{-1} M D_1)x + D_2^{-1} q \tag{52}$$

$$\frac{1}{2} e \geq x, y \geq 0 \tag{53}$$

$$x^T y = 0 \tag{54}$$

which without loss of generality will be indicated as

$$Mx + q - y = 0 \tag{55}$$

$$x, y \geq 0 \tag{56}$$

$$x^T y = 0\,. \tag{57}$$

Assume that there exists an approximate interior point solution, as is usual with interior point methods, with variables $0 < x_i, y_i \leq \epsilon,\ \epsilon < n^{-2}\ \forall i = 1, 2, \ldots, n$ and consider the following barrier function for the optimization problem for the LCP (55)–(57).

$$\text{Minimize } \psi(x, y, \mu) = x^T y - \mu \sum_{i=1}^{n} \ln(x_i y_i) \tag{58}$$

$$\text{subject to } Mx - y + q = 0 \tag{59}$$

$$x, y < \frac{1}{2} e \tag{60}$$

$$x, y > 0 \tag{61}$$

where $e \in R^n$ is the vector of unit elements and $\beta > 0$ is an arbitrary small parameter.

To convert the optimization problem (58)–(61) into a convex programming problem, consider as a barrier parameter, which is successively reduced, then the gradient of this function is:

$$(\nabla_x \psi(x,\ y))_i = \frac{x_i y_i^2 - (\beta - \mu) y_i}{x_i y_i + \beta}\,, \tag{62}$$

$$(\nabla_x \psi(x,\ y))_i = \frac{x_i^2 y_i - (\beta - \mu) x_i}{x_i y_i + \beta}\,. \tag{63}$$

It is easy to show that if the barrier parameter at any iteration k will satisfy the following inequality

$$\mu > \frac{(x_i y_i + \beta)^2}{y_i^2 + \beta}\,, \tag{64}$$

then the Hessian matrix of the function (58) is positive denite for the conditions imposed. Thus the optimization problem (58)–(61) is a convex programming problem and it may be solved by one of the methods above, which is also suitable to a further generalization [1]. Here it will be solved as a convex quadratic programming [12]. Rewrite the optimization problem (58)–(61) as:

$$\text{Min } \psi(x,\ y,\ \mu) = x^T y - \mu \sum_{i=1}^{n} \ln(x_i y_i + \beta)\,, \tag{65}$$

$$\text{subject to} \begin{pmatrix} M & -I \\ I & 0 \\ 0 & I \\ -I & 0 \\ 0 & -I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + b \geq 0 \,. \qquad (66)$$

Where $b^T = (q^T, 0, 0, \frac{1}{2}e^T, \frac{1}{2}e^T)$.

Indicate the constraint matrix as the matrix A of dimension $5n \times 2n$. Also, idicate with $z^T = (x^T, y^T) \in R^{2n}$.

The algorithm considered is a primal method with a log barrier function. It will follow a central path and will take small steps [12] and it can be shown that from an approximate global minimum, an exact global minimum can be simply derived [12].

Let Π denote the feasible region of Eq. (66) and denote the interior of this feasible region by int(Π), i. e., $Az > b$ by relaxing as is usual in the Interior point algorithms, the equality constraints.

Make the following assumptions:

- rank(A) = $2n$,
- Π is compact,
- int(Π) $\neq \emptyset$.
- $x_i y_i > \varepsilon \quad \forall i = 1, 2, \ldots, n.$

Define the potential function

$$h(z, \mu) = \psi(x, y, \mu) - \mu \sum_{i=1}^{m} \ln(a_i^T z - b_i) \,. \qquad (67)$$

The following lemmas are straight forward adaptations of the original results.

Lemma 3 *For any fixed choice of $\mu > 0$, that meets the condition (64), the function (67) is strictly convex on int(Π).*

Lemma 4 *For any fixed choice of $\mu > 0$, that meets the condition (64), the function (67) has a unique minimum.*

Let $\zeta(\mu)$ be the minimum of $h(z, \mu)$ for a fixed μ. As $\mu \to 0$ there must be an accumulation point by compactness. This point must be an approximate global minimum.

Lemma 5 *Let $\hat{z}$ be an accumulation point of $\zeta(\mu)$. As $\mu \to 0$ then $\hat{z}$ is an approximate global minimum for problem (65)–(66).*

Generalization of an Interior Point Reduction Algorithm to Solve General LCPs

The condition number for LCPs provides a criterion to subdivide given instances of LCP(q,M) into classes. These results will now be extended.

Consider a LCP(q,M) Eqs. (9)–(11) with a nonsingular coefficient matrix M, for which, moreover, (I–M) is nonsingular and the solution set of LCP(q,M) is bounded from above. This LCP can be indicated so:

$$Mu + q - v = 0 \,, \qquad (68)$$

$$u, v \geq 0 \,, \qquad (69)$$

$$u^T v = 0 \,, \qquad (70)$$

where $u, v, q \in R^n$. Suppose that the LCP solution set $S = \{u, v| \quad Mu + q - v = 0, u, v \geq 0, u^T v = 0\}$ is bounded above by a vector $(m_1^T, m_2^T)^T \in R^{2n}$.

Apply the transformation defined by Eqs. (50) and (51), so that there results

$$y = D_2^{-1} v = D_2^{-1}(Mu + q) = (D_2^{-1} M D_1)x + D_2^{-1} q \,, \qquad (71)$$

$$\frac{1}{2}e \geq x, y \geq 0 \,, \qquad (72)$$

$$x^T y = 0 \,, \qquad (73)$$

which will be indicated as

$$Mx + q - y = 0 \,, \qquad (74)$$

$$x, y \geq 0 \,, \qquad (75)$$

$$x^T y = 0 \,. \qquad (76)$$

For the potential reduction algorithm to solve general LCPs, it is required that $x > 0$ and $y > 0$.

Lemma 6 *For a nonsingular M the matrices $\hat{M} = D_1 M D_2$, $(I - \hat{M})$, $(I - XY\hat{M})$ and $(-Y + \hat{M}X)$ are all nonsingular.*

Corollary 2 *Under the conditions of Lemma 3 $(Y + MX)$ is nonsingular.*

The following additional lemma is also required.

Lemma 7 *For all LCP(q,M) with nonsingular matrices M and (I–M) transformed to the form given by Eqs. (71)–(73) so that for any feasible solution* $(x, y) \in int(F)$ *so that* $0 < X < I,\ 0 < Y < I$, *there results* $g(x, y) = \frac{n+\rho}{x^T y} Xy - e \neq 0$.

Theorem 3 *For all LCP(q,M) with nonsingular matrices M and (I–M) transformed to the form given by Eqs. (71)–(73) so that for any feasible solution* $(x, y) \in int(F)$ *there results* $0 < X < I,\ 0 < Y < I$, *the condition number for the LCP* $\gamma(M, q, \epsilon) > 0$ *for some* $\rho > 0$.

For notational simplicity assume that the transformed matrix $\hat{M}$ is indicated by M without loss of generality. $\gamma(M, q, \epsilon) = 0$ if $\|g(x, y)\|_H^2 = 0$. Assume that $\|g(x, y)\|_H^2 = 0$ and expand it in terms of its factors.

$$2g(x, y)^T g(x, y) - g(x, y)^T [(XM^T - Y)(Y^2 + MX^2M^T)^{-1}(MX - Y)] \cdot g(x, y) = 0 \quad (77)$$

It is easy to show that this will never happen under the conditions of the theorem. Hence, for any matrix that satisfies the assumed conditions the condition number is strictly positive and so a solution to the LCP may be obtained straightforwardly by this method. This provides a partial characterization and extension of the matrix class $\mathcal{G}$ defined in [16].

Cases

Algorithms should be tested extensively for their computational efficiency on a wide series of cases, so that suitable comparisons can be made.

One hundred and forty random instances of LCPs were solved for four different sizes (30, 50, 100, 250), with three types of matrices: positive semidefinite, negative semidefinite and indefinite. In Table 1 the number of problems solved for each type of matrix with the parametric LCP algorithm [11] and with an interior point algorithm with the Newton method are indicated.

The instances with positive (semi)definite matrices are easy to solve in fact. The instances with negative (semi)definite and indefinite classes are considered hard to solve, but both algorithms have no trouble with these classes, except that the first seems to be more hap-

Generalizations of Interior Point Methods for the Linear Complementarity Problem, Table 1
Results for 140 linear complementarity problems (LCPs) of different matrix classes and sizes

Type	PSD		NSD		INDF	
Size	PLCP	IPNM	PLCP	IPNM	PLCP	IPNM
30	6	6	12	12	28	28
50	3	3	3	3	26(3)	29
100	6	6	6	6	16	16
250	5	5	7(4)	11	15	15
Total	20	20	28(32)	32	85(88)	88

PSD positive semidefinite matrix, *NSD* negative semidefinite matrix, *INDF* indefinite matrix, *PLCP* parametric LCP algorithm, *IPMN* interior point algorithm with the Newton method.

Generalizations of Interior Point Methods for the Linear Complementarity Problem, Table 2
Timing results for 140 LCPs of different matrix classes and sizes (seconds)

Type	PSD		NSD		INDF	
Size	PLCP	IPNM	PLCP	IPNM	PLCP	IPNM
30	0.06	0.04	0.08	0.06	0.07	0.07
50	0.28	0.18	0.38	0.32	0.33	0.32
100	3.47	1.42	7.00	3.37	5.18	2.78
250	109.37	22.56	121.51	95.12	111.99	87.45

hazard, rather than being subject to numerical difficulties.

Both routines seem to be only slightly affected by the type of matrix, but the interior point algorithm with the Newton method is more efficient, as confirmed in Table 2, where the average time for solving the instances is given in seconds.

Conclusions

Interior point methods to solve the LCP are now well established and allow polynomial solutions to be obtained for such problems with suitable matrix classes. Moreover these routines can be used as a subroutine in general iterative optimization problems.

Evidently research is being actively conducted to generalize the applicable matrix classes for which solutions can be obtained in polynomial time and space.

See also

- ▶ Complementarity Algorithms in Pattern Recognition
- ▶ Mathematical Programming Methods in Supply Chain Management
- ▶ Simultaneous Estimation and Optimization of Nonlinear Problems

References

1. Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge University Press, Cambridge
2. Cottle RW, Dantzig G (1968) Complementarity pivot theory of mathematical programming. Lin Algebra Appl 1:103–125
3. Cottle RW, Pang J-S, Stone RE (1992) The Linear Complementarity Problem. Academic Press, Inc., San Diego
4. Kojima M, Megiddo N, Ye Y (1992) An Interior point potential reduction algorithm for the linear complementarity problem. Math Programm 54:267–279
5. Lemke CE (1965) Bimatrix Equilibrium Points and Mathematical Programming. Manag Sci 11:123–128
6. Lemke CE, Howson JT (1964) Equilibrium points of bimatrix games. SIAM J Appl Math 12:413–423
7. Mangasarian OL (1979) Simplified characterizations of linear complementarity problems solvable as linear programs. Math Programm 10(2):268–273
8. Mangasarian OL (1976) Linear complementarity problems solvable by a single linear program. Math Programm 10:263–270
9. Mangasarian OL (1978) Characterization of linear co complementarity problems as linear program. Math Programm 7:74–87
10. Ferris MC, Sinapiromsaran K (2000) Formulating and Solving Nonlinear Programs as Mixed Complementarity Problems. In: Nguyen VH, Striodot JJ, Tossing P (eds) Optimization. Springer, Berlin, pp 132–148
11. Patrizi G (1991) The Equivalence of an LCP to a Parametric Linear program with a Scalar Parameter. Eur J Oper Res 51:367–386
12. Vavasis S (1991) Nonlinear Optimization: Complexity Issues. Oxford University Press, Oxford
13. Ye Y (1991) An $O(n^3L)$ Potential Reduction Algorithm for linear Programming. Math Programm 50:239–258
14. Ye Y (1992) On affine scaling algorithms for nonconvex quadratic programming. Math Programm 56:285–300
15. Ye Y (1993) A fully polynomial-time approximation algorithm for computing a stationary point of the general linear complementarity problem. Math Oper Res 18:334–345
16. Ye Y, Pardalos PM (1991) A Class of Linear Complementarity Problems Solvable in Polynomial Time. Lin Algebra Appl 152:3–17
17. Ye Y (1997) Interior Point Algorithms: Theory and Analysis. Wiley, New York

Generalized Assignment Problem

O. Erhun Kundakcioglu, Saed Alizamir
Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA

MSC2000: 90-00

Article Outline

Introduction
Extensions
Multiple-Resource Generalized Assignment Problem
Multilevel Generalized Assignment Problem
Dynamic Generalized Assignment Problem
Bottleneck Generalized Assignment Problem
Generalized Assignment Problem with Special Ordered Set
Stochastic Generalized Assignment Problem
Bi-Objective Generalized Assignment Problem
Generalized Multi-Assignment Problem
Methods
Exact Algorithms
Heuristics
Conclusions
References

Introduction

The generalized assignment problem (GAP) seeks the minimum cost assignment of n tasks to m agents such that each task is assigned to precisely one agent subject to capacity restrictions on the agents.

The formulation of the problem is:

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij}x_{ij} \le b_i \quad i = 1,\ldots,m \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad j = 1,\ldots,n \tag{3}$$

$$x_{ij} \in \{0,1\} \quad i = 1,\ldots,m;\ j = 1,\ldots,n \tag{4}$$

where c_{ij} is the cost of assigning task j to agent i, a_{ij} is the capacity used when task j is assigned to agent i, and b_i is the available capacity of agent i. Binary variable x_{ij} equals 1 if task j is assigned to agent i, and 0

otherwise. Constraints 3 are usually referred to as the *semi-assignment constraints*.

The formulation above was first studied by Srinivasan and Thompson [80] to solve a transportation problem. The term *generalized assignment problem* for this setting was introduced by Ross and Soland [74]. This model is a generalization of previously proposed model by DeMaio and Roveda [17] where the capacity absorption is agent independent (i. e., $a_{ij} = a_j,\ \forall i$).

The classical assignment problem, which provides a one to one pairing of agents and tasks, can be solved in polynomial time [47]. However, in GAP, an agent may be assigned to multiple tasks ensuring each task is performed exactly once, and the problem is $\mathcal{NP}$-hard [28]. Even the GAP with agent-independent requirements is an $\mathcal{NP}$-hard problem [23,53].

The GAP has a wide spectrum of application areas ranging from scheduling (see [19,84]) and computer networking (see [5]) to lot sizing (see [31]) and facility location (see [7,30,74,75]). Nowakovski et al. [64] study the ROSAT space telescope scheduling where the problem is formulated as a GAP and heuristic methods are proposed. Multiperiod single-source problem (MPSSP) is reformulated as a GAP by Freling et al. [25]. Janak et al. [38] reformulate the NSF panel-assignment problem as a multiresource preference-constrained GAP. Other applications of GAP include lump sum capital rationing, loading in flexible manufacturing systems (see [45]), p-median location (see [7,75]), maximal covering location (see [42]), cell formation in group technology (see [79]), refueling nuclear reactors (see [31]), R & D planning (see [92]), and routing (see [22]). A summary of applications and assignment model components can be found in [76].

Extensions

Multiple-Resource Generalized Assignment Problem

Proposed by Gavish and Pirkul [29], multi-resource generalized assignment problem (MRGAP) is a special case of *the multi-resource weighted assignment model* that is previously studied by Ross and Zoltners [76]. In MRGAP a set of tasks has to be assigned to a set of agents in a way that permits assignment of multiple tasks to an agent subject to a set of resource constraints. This problem differs from the GAP in that, an agent consumes a variety of resources in performing the tasks assigned to it. Although most of the problems can be modeled as GAP, multiple resource constraints are frequently required in the effective modeling of real life problems. MRGAP may be encountered in large models dealing with processor and database location in distributed computer systems, trucking industry, telecommunication network design, cargo loading on ships, warehouse design and work load planning in job shops.

Gavish and Pirkul [29] introduce and compare various Lagrangian relaxations of the problem and suggest heuristic solution procedures. They design an exact algorithm by incorporating one of these heuristics along with a branch-and-bound procedure.

Mazzola and Wilcox [58] modify Gavish and Pirkul heuristic and develop a hybrid heuristic for MRGAP. Their algorithm defines a three phase heuristic which first constructs a feasible solution and then systematically tries to improve the solution. As an enhanced version of MRGAP, Janak et al. [38] study the NSF panel-assignment problem. In this setting, each task (i. e., proposal) has a specific number of agents (i. e., reviewers) assigned to it and each agent has a lower and upper bound on the number of tasks that can be done. The objective is to optimize the sum of a set of preference criteria for each agent on each task while ensuring that each agent is assigned to approximately the same number of tasks.

Multilevel Generalized Assignment Problem

The Multilevel Generalized Assignment Problem (MGAP) is first introduced by Glover et al. [31] to provide a model for the allocation of tasks in a manufacturing environment. MGAP differs from the classical GAP in that, agents can perform tasks at different efficiency levels, implying both different costs and different resource requirements. Each task must be assigned to one and only one agent at a level and each agent has limited amount of single resource. Important manufacturing problems, such as lot sizing, can be formulated as MGAP.

Laguna et al. [46] use a neighborhood structure for defining moves based on ejection chains and develop a Tabu Search (TS) algorithm for this problem. French and Wilson [26] develop two heuristic solution methods for MGAP from the solution methods

for GAP. Procedures for deriving an upper bound on the solution of the problem are also described. Ceselli and Righini [11] present a branch-and-price algorithm based on decomposition of the MGAP into a master problem and a pricing sub-problem, where the former is a set-partitioning problem and the latter is a multiple-choice knapsack problem. This algorithm is the first exact method proposed in the literature for the MGAP. To provide a flexible assignment tool to the decision maker, Hajri-Gabouj [37] develops a fuzzy genetic multi-objective optimization algorithm to solve a nonlinear MGAP.

Dynamic Generalized Assignment Problem

In The Gap Model, the sequence in which the agent performs the tasks is not considered. This sequence is essential when each task is performed to meet a demand and earliness or tardiness incurs additional cost. Dynamic generalized assignment problem (DGAP) is suggested to track customer demand while assigning tasks to agents. Kogan et al. [44], for the first time, add the impact of time to the GAP model assuming that each task has a due date. They formulate the continuous-time optimal control model of the problem and derive analytical properties of the optimal behavior of such a dynamic system. Based on those properties, an efficient time-decomposition procedure is developed.

Kogan et al. [43] extend the DGAP to cope with stochastic environment and multiple agent-task relationships. They prove that this stochastic, continuous-time generalized assignment problem is strongly $\mathcal{NP}$-hard and reduce the model to a number of classical deterministic assignment problems stated at discrete time points. A pseudo-polynomial time combinatorial algorithm is developed to approximate the solution. The well-known application of such a generalization is found in the stochastic environment of the flow shop scheduling of parallel workstations and flexible manufacturing cells as well as dynamic inventory management.

Bottleneck Generalized Assignment Problem

Bottleneck generalized assignment problem (BGAP), is the min-max version of the well-known (min-sum) generalized assignment problem. In the BGAP, the maximum penalty incurred by assigning each task to an agent is minimized. Min-sum objective functions are commonly used in private sector applications, while min-max objective function can be applied to the public sector. BGAP has several important applications in scheduling and allocation problems. Mazzola and Neebe [57] propose two min-max formulations for the GAP: the Task BGAP and the Agent BGAP. Martello and Toth [56] present an exact branch-and-bound algorithm as well as approximate algorithms for BGAP. They introduce relaxations and produce, as sub-problems, min-max versions of the multiple-choice knapsack problem which can be solved in polynomial time.

Generalized Assignment Problem with Special Ordered Set

GAP is further generalized to include cases where items may be shared by a pair of adjacent knapsacks. This problem is called the generalized assignment problem with special ordered sets of type 2 (GAPS2). In other words, GAPS2 is the problem of allocating tasks to time-periods, where each task must be assigned to a time-period, or shared between two consecutive time-periods. Farias et al. [15] introduce this problem which can also be applied to production scheduling. They study the polyhedral structure of the convex hull of the feasible space, develop three families of facet-defining valid inequalities, and show that these inequalities cut off all infeasible vertices of the LP relaxation. A branch-and-cut procedure is described and facet-defining valid inequalities are used as cuts. Wilson [86] modifies and extends a heuristic algorithm developed previously for the GAP problem to solve GAPS2. He argues that, any feasible solution to GAP is a feasible solution to GAPS2, hence a heuristic algorithm for GAP can also be used as a heuristic algorithm to GAPS2. A solution produced by a GAP heuristic will be close to GAPS2 optimality if it is close to the LP relaxation bound of GAP. The heuristic uses a series of moves starting from an infeasible, but in some senses *optimal* solution and then attempts to restore feasibility with minimal degradation to the objective function value. An existing upper bound for GAP is also generalized to be used for GAPS2.

French and Wilson [27] develop an LP-based heuristic procedure to solve GAPS2. They modify a heuristic for GAP to be used for GAPS2 and show

that, while Wilson [86] heuristic is straightforward for large instances of the problem, and Farias et al. [15] solve smaller instances of the problem by an exact method, their heuristic solves fairly large instances of the problem rapidly and with a consistently high degree of solution quality.

Stochastic Generalized Assignment Problem

In GAP, stochasticity may arise because the actual amount of resource needed to process the tasks by the different agents may not be known in advance or the presence or absence of individual tasks may be uncertain. In such cases, there is a set of potential tasks in which, each task may or may not require to be processed. Dyer and Frieze [20], analyze the generalized assignment problem under the assumption that all coefficients are drawn uniformly and independently from $[0, 1]$ interval. Romeijn and Piersma [72] analyze a probabilistic version of GAP as the number of tasks goes to infinity while the number of machines remains fixed. Their model is different from Dyer and Frieze [20] since it doesn't have the additional assumptions that the cost and resource requirement parameters are independent of each other and among machines. They first derive a tight condition on the probabilistic model of the parameters under which, the corresponding instances of the GAP are feasible with probability one. Next, under an additional sufficient condition, the optimal solution value of the GAP is characterized through a limiting value. It is shown that the optimal solution value, normalized by dividing by the number of tasks, converges with probability one to this limiting value. Toktas et al. [82], consider the uncertain capacities situation and derive two alternative approaches to utilize deterministic solution strategies while addressing capacity uncertainty. Albareda-Sambola et al. [1] assume that a random subset of the tasks would require to be actually processed. Tasks are interpreted as customers that may or may not require a service. They construct a convex approximation of the objective function and present three versions of an exact algorithm to solve this problem based on branch-and-bound techniques, optimality cuts, and a special purpose lower bound. An assignment of tasks can be modified once the actual demands are known. Different penalties are paid for reassigning tasks and for leaving unprocessed tasks with positive demand.

Bi-Objective Generalized Assignment Problem

Zhang and Ong [91] consider the GAP from a multi-objective point of view, and propose an LP-based heuristic to solve the bi-objective generalized assignment problem (BiGAP). In BiGAP, each assignment has two attributes that are to be considered. For example, in production planning, these attributes may be the cost and the time caused by assigning jobs to machines.

Generalized Multi-Assignment Problem

Proposed by Park Et Al. [66], the generalized multi-assignment problem (GMAP) consists of tasks that may be required to be duplicated at several agents. In other words, each task is assigned to r_j agents instead of one. Park et al. [66] develop a Lagrangian dual ascent algorithm for the GMAP that is combined with the subgradient search and used as a lower bounding scheme for the branch-and-bound procedure.

Methods

Determining whether an instance of a GAP has a feasible solution is an $\mathcal{NP}$-complete problem. Hence, unless $\mathcal{P} = \mathcal{NP}$, GAP admits no polynomial-time approximation algorithm with fixed worst-case performance ratio. Nevertheless there are numerous *approximation algorithms for GAP* in the literature which actually address a different setting where the available agent capacities are not fixed and the weighted sum of cost and available agent capacities is minimized. For some of these algorithms, a feasible solution is required as an input. For details, see [14,24,65,78]. Excluding this setting for GAP, the solution approaches proposed in the literature are either exact algorithms or heuristics. For expository surveys on the algorithms, see [10,54,60].

Exact Algorithms

The optimal solution to the GAP is obtained using an implicit enumerative procedure either via branch-and-bound scheme or branch-and-price scheme in the literature. Branch-and-bound method consists of an upper bounding procedure, a lower bounding procedure, a branching strategy, and a searching strategy. It

is known that good bounding procedures are crucial steps in branch-and-bound method. Branch-and-price proceeds similar to branch-and-bound but obtains the bounds by solving the LP-relaxations of the subproblems by column generation. For more details on the valid inequalities and facets for the GAP that are used in the solution procedures, see [16,32,33,40,55,67].

The first branch-and-bound algorithm for the GAP is proposed by Ross and Soland [74]. Considering a minimization problem, they obtain the lower bounds by relaxing the capacity constraints. Martello and Toth [53] propose removing the semi-assignment constraints where the problem decomposes into a series of knapsack problems. Due to the quality of the bounds obtained, this algorithm is frequently used in the literature for benchmarking purposes. Chalmet and Gelders [12] introduce the Lagrangian relaxation of the semi-assignment constraints. Fisher et al. [23] use this technique with multipliers set by a heuristic adjustment method to obtain the lower bounds in the branch-and-bound procedure. Tighter bounds resulted from this method, significantly reduce the solution time. Guignard and Rosenwein [34] design a branch-and-bound algorithm with an enhanced Lagrangian dual ascent procedure that solves a Lagrangian dual at each enumeration node and adds a surrogate constraint to the Lagrangian relaxed model. This algorithm effectively solves generalized assignment problems with up to 500 variables. Drexl [19] presents a hybrid branch-and-bound/dynamic programming algorithm where the upper bounds are obtained via an efficient Monte Carlo type heuristic. Numerous lower bounds are proposed and their benchmark results are presented. Nauss [62] proposes a branch-and-bound algorithm where linear programming cuts, Lagrangian relaxation, and subgradient optimization are used to derive good lower bounds; feasible-solution generators with the heuristic proposed by Ronen [73] are used to derive good upper bounds. Nauss [63] uses similar branch-and-bound techniques to solve the elastic generalized assignment problem (EGAP) as well.

The first branch-and-price algorithm for the generalized assignment problem is proposed by Savelsbergh [77]. A combination of the algorithms proposed by Martello and Toth [53] and Jörnsten and Nasberg [39] is used to calculate the upper bound and the pricing problem is proved to be a knapsack problem. Barnhart et al. [6] reformulate the GAP by applying Dantzig-Wolfe decomposition to obtain a tighter LP relaxation. In order to solve the LP relaxation of the reformulated problem, pricing is done by solving a series of knapsack problems. Pigatti et al. [67] propose a branch-and-cut-and-price algorithm with a stabilization mechanism to speed up the pricing convergence. Ceselli and Righini [11] present a branch-and-price algorithm for *multilevel generalized assignment problem* that is based on decomposition and a pricing subproblem that is a multiple-choice knapsack problem.

Heuristics

Large instances of the GAP are computationally intractable due to the $\mathcal{NP}$-hardness of the problem. This calls for heuristic approaches whose benefits are twofold; they can be used as stand-alone algorithms to obtain good solutions within reasonable time and they can be used to obtain the upper bounds in exact solution methods such as the branch-and-bound procedure. Although the variety among the heuristics is high, they mostly fall into one of the following two categories: greedy heuristics and meta-heuristics.

Klastorin [41] proposes a two phase heuristic algorithm for solving the GAP. In phase one, the algorithm employs a modified subgradient algorithm to search for the optimal dual solution and in phase two, a branch-and-bound approach is used to search the neighborhood of the solution obtained in phase one.

Cattrysse et al. [9] use column generation techniques to obtain upper and lower bounds. In their method, a column represents a feasible assignment of a subset of tasks to a single agent. The master problem is formulated as a set partitioning problem. New columns are added to the master problem by solving a knapsack problem for each agent. LP relaxation of the set partitioning problem is solved by a dual ascent procedure.

Martello and Toth [54] present a greedy heuristic that assigns the jobs to machines based on a desirability factor. This factor is defined as the difference between the largest and second largest weight factors. The algorithm iteratively considers, among the unassigned jobs, the one having the highest desirability factor (or regret factor) and assigns it to its maximum profit agent. This iterative process establishes an initial solution which would be improved in the next step of the algorithm

by simple interchange arguments. This heuristic can be used in a problem size reduction procedure by fixing variables to one or to zero.

Relaxation heuristics are developed by Lorena and Narciso [49] for maximization version of GAP. Feasible solutions are obtained by a subgradient search in a Lagrangian or surrogate relaxation. Six different heuristics are derived particularizing relaxation, the step size in the subgradient search and the method used to obtain the feasible solution. In a Lagrangian heuristic for GAP, Haddadi [35] introduces a substitution variable in the model which is defined as the multiplication of the original variables by their corresponding constraint coefficients. The constraints defining these new variables are then dualized in the Lagrangian relaxation of the problem and the resulted relaxation is decomposed into two subproblems: the knapsack problem and the transportation problem. Narciso and Lorena [61] use relaxation multipliers with efficient constructive heuristics to find good feasible solutions.

A breadth-first branch-and-bound algorithm is described by Haddadi and Ouzia [36] in which a standard subgradient approach is used in each node of the decision tree to solve the Lagrangian dual and to obtain an upper bound. The main contribution in this study is a new heuristic that is applied to exploit the solution of the relaxed problem by solving a GAP of smaller size.

Romeijn and Romero Morales [70] study the optimal value function from a probabilistic point of view and develop a class of greedy algorithms. A family of weight functions is designed to measure desirability of assigning each job to a machine which is used by the greedy algorithms. They derive conditions under which their algorithm is asymptotically optimal in a probabilistic sense.

Meta-heuristics are widely used to solve GAP in the literature. They are either adapted by themselves for GAP or are used in combination with other heuristics and meta-heuristics.

Variable depth search heuristic (VDSH) is a generalization of local search in which the size of the neighborhood adaptively changes to traverse a larger search space. VDSH is a two phase algorithm. In the first phase, an initial solution is developed and a lower bound is obtained. In the second phase, a nested iterative refinement process is applied to improve the quality of the solution. VDSH is introduced by Amini and Racer [2] to solve the GAP. In their method, the improvement phase consists of a two level nested loop. The major iteration creates an action set corresponding to each neighborhood structure alternative. Possible neighborhood structures for GAP are: reassign (shift) a task from one agent to another, swap the assignment of two tasks, and permute the assignment of a subset of the tasks. Then, a subsequence of operations that achieves the highest saving is obtained through performing some minor iterations. A new solution is established based on that and another major operation starts.

Amini and Racer [3] develop a hybrid heuristic (HH) around the two well known heuristics: VDSH (see [2,69]) and Heuristic GAP (HGAP) (see [54]). Previous studies show that HGAP dominates VDSH in terms of solution time, while VDSH obtains solutions of better quality within reasonable time. A computational comparison is conducted with the leading alternative heuristic approaches. Another hybrid approach is by Lourenço and Serra [52] where a MAX-MIN Ant System (MMAS) (see [81]) is applied with GRASP for the GAP.

Yagiura et al. [90] propose a variable depth search (VDS) method for GAP. Their method alternates between shift and swap moves to explore the solution space. The main aspect of their method is that, infeasible solutions are allowed to be considered. However in some of the problem instances, the feasible space is small or contains many small separate regions and the efficiency of the algorithm is affected. In another study, Yagiura et al. [89] improve VDS by incorporating branching search processes to construct the neighborhoods. They show that appropriate choices of branching strategies can improve the performance of VDS. Lin et al. [48] make further observations on the VDSH method through a series of computational experiments. They consider six greedy strategies for generating the initial feasible solution and designed several simplified strategies for the improvement phase of the method.

Osman [68] develops a hybrid heuristic which combines simulated annealing and tabu search. This algorithm takes advantage of the non-monotonic oscillation strategy of tabu search as well as the simulated annealing philosophy.

Yagiura et al. [87] propose a tabu search algorithm for GAP which utilizes an ejection chain approach. An

ejection chain is an embedded neighborhood construction that compounds simple moves to create more complex and powerful moves. The chain considered in their study is a sequence of shift moves in which every two successive moves share a common agent. Searching into the infeasible region is allowed incurring a penalty proportional to the degree of infeasibility. An adaptive adjustment mechanism is incorporated for determining appropriate values of the parameters to control their influence on the problem. Yagiura et al. [88] improve their previous method by adding a path relinking approach which is a mechanism for generating new solutions by combining two or more reference solutions. The main difference of this method with the previous one is the way it generates starting solutions for ejection chains. It is shown that, by this simple change in the algorithm, the improvement in its performance is drastic.

Asahiro et al. [4] develop two parallel heuristic algorithms based on the ejection chain local search (EC) presented by Yagiura et al. [87]. One is a simple parallelization called multi-start parallel EC (MPEC) and the other one is cooperative parallel EC (CPEC). In MPEC, each search process independently explores search space while in CPEC search processes share partial information to cooperate with each other. They show that their proposed algorithms outperform EC by Yagiura [87].

Diaz and Fernandez [18], devise a flexible tabu search algorithm for GAP. Allowing the search to explore infeasible region and adaptively modification of the objective function are the sources of flexibility. The modification of the objective function is caused by the dynamic adjustment of the weight of the penalty incurred for violating feasibility. The main difference of this method with the tabu search method of Yagiura et al. [87,88] in exploring the infeasible region is that, in this method, no solution is qualitatively preferred to others in terms of its structure.

Chu and Beasley [13] develop a genetic algorithm for GAP that incorporates a fitness-unfitness pair evaluation function as a representation scheme. This algorithm uses a heuristic to improve the cost and feasibility. Feltl and Raidl [21] add new features to this algorithm including two alternative initialization heuristics, a modified selection and replacement scheme for handling infeasible solutions more appropriately and a heuristic mutation operator.

Wilson [85] proposes another algorithm for GAP which is operating in a dual sense. Instead of genetically improving a set of feasible solutions as in a regular GA, this algorithm tries to genetically restore feasibility to a set of near optimal ones. The method starts with potentially optimal but infeasible solutions and then improves feasibility while keeping optimality. When the feasible solution is obtained, the algorithm uses local search procedures to improve the solution.

Lorena et al. [50] propose a constructive genetic algorithm (CGA) for GAP. In CGA, unlike classical GA, problems are modeled as bi-objective optimization problems, which consider the evaluation of two fitness functions. The evolution process is conducted to attain the two objectives conserving schemata that survive to an adaptive threshold test. The CGA algorithm has some new features compared to GA including population formation by schemata, recombination among schemata, dynamic population, mutation in structure and the possibility of using heuristics in schemata and/or structure representation.

Lourenço and Serra [51] present two metaheuristic algorithms for GAP. One is a MIN-MAX ant system which is combined with local search and tabu search heuristics. The other one is a greedy randomized adaptive search heuristic (GRASP) studied with several neighborhoods. Both of these algorithms consist of three main steps: *(i)* constructing a solution by either a greedy randomized or an ant system approach, *(ii)* improving these initial solutions by applying local search and a tabu search, *(iii)* updating the parameters. These three steps are repeated until a stopping criterion is verified.

Monfared and Etemadi [59] use a neural network based approach for solving the GAP. They investigate four different methods to structure the energy function of the neural network: exterior penalty function, augmented Lagrangian, dual Lagrangian and interior penalty function. They show that augmented Lagrangian can produce superior results with respect to feasibility and integrality while maintaining feasibility and stability measures.

Problem generators and benchmark instances play an important role in comparing/developing new methods. Romeijn and Romero Morales [71] propose a new stochastic model for the GAP which can be used to analyze the random generators in the literature. They com-

pare the random generators by Ross and Soland [74], Martello and Toth [53], Trick [83], Chalmet and Gelders [12], Racer and Amini [69] and conclude these random generators are not adequate because they tend to generate easier problem instances when the number of machines increases. Cario et al. [8] compare GAP instances generated under two correlation-induction strategies. Using two exact and four heuristic algorithms from the literature, they show how solutions are affected by the correlation between costs and the resource requirements.

Conclusions

This review presents the applications, extensions, and solution methods for the generalized assignment problem. As the GAP receives more attention, it will be more likely to see large sets of classical benchmark instances and comparative results on solution approaches.

References

1. Albareda-Sambola M, van der Vlerk MH, Fernandez E (2006) Exact solutions to a class of stochastic generalized assignment problems. Eur J Oper Res 173:465–487
2. Amini MM, Racer M (1994) A rigorous computational comparison of alternative solution methods for the generalized assignment problem. Manag Sci 40(7):868–890
3. Amini MM, Racer M (1995) A hybrid heuristic for the generalized assignment problem. Eur J Oper Res 87(2):343–348
4. Asahiro Y, Ishibashi M, Yamashita M (2003) Independent and cooperative parallel search methods for the generalized assignment problem. Optim Method Softw 18:129–141
5. Balachandran V (1976) An integer generalized transportation model for optimal job assignment in computer networks. Oper Res 24(4):742–759
6. Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: column generation for solving huge integer programs. Oper Res 46(3):316–329
7. Beasley JE (1993) Lagrangean heuristics for location problems. Eur J Oper Res 65:383–399
8. Cario MC, Clifford JJ, Hill RR, Yang J, Yang K, Reilly CH (2002) An investigation of the relationship between problem characteristics and algorithm performance: a case study of the gap. IIE Trans 34:297–313
9. Cattrysse DG, Salomon M, Van LN Wassenhove (1994) A set partitioning heuristic for the generalized assignment problem. Eur J Oper Res 72:167–174
10. Cattrysse DG, Van LN Wassenhove (1992) A survey of algorithms for the generalized assignment problem. Eur J Oper Res 60:260–272
11. Ceselli A, Righini G (2006) A branch-and-price algorithm for the multilevel generalized assignment problem. Oper Res 54:1172–1184
12. Chalmet L, Gelders L (1976) Lagrangean relaxation for a generalized assignment type problem. In: Advances in OR. EURO, North Holland, Amsterdam, pp 103–109
13. Chu EC, Beasley JE (1997) A genetic algorithm for the generalized assignment problem. Comput Oper Res 24:17–23
14. Cohen R, Katzir L, Raz D (2006) An efficient approximation for the generalized assignment problem. Inf Process Lett 100:162–166
15. de Farias Jr, Johnson EL, Nemhauser GL (2000) A generalized assignment problem with special ordered sets: a polyhedral approach. Math Program, Ser A 89:187–203
16. de Farias Jr, Nemhauser GL (2001) A family of inequalities for the generalized assignment polytope. Oper Res Lett 29:49–55
17. DeMaio A, Roveda C (1971) An all zero-one algorithm for a class of transportation problems. Oper Res 19:1406–1418
18. Diaz JA, Fernandez E (2001) A tabu search heuristic for the generalized assignment problem. Eur J Oper Res 132:22–38
19. Drexl A (1991) Scheduling of project networks by job assignment. Manag Sci 37:1590–1602
20. Dyer M, Frieze A (1992) Probabilistic analysis of the generalised assignment problem. Math Program 55:169–181
21. Feltl H, Raidl GR (2004) An improved hybrid genetic algorithm for the generalized assignment problem. In: SAC '04; Proceedings of the 2004 ACM symposium on Applied computing. ACM Press, New York, pp 990–995
22. Fisher ML, Jaikumar R (1981) A generalized assignment heuristic for vehicle routing. Netw 11:109–124
23. Fisher ML, Jaikumar R, van Wassenhove LN (1986) A multiplier adjustment method for the generalized assignment problem. Manag Sci 32:1095–1103
24. Fleischer L, Goemans MX, Mirrokni VS, Sviridenko M (2006) Tight approximation algorithms for maximum general assignment problems. In SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm. ACM Press, New York, pp 611–620
25. Freling R, Romeijn HE, Morales DR, Wagelmans APM (2003) A branch-and-price algorithm for the multiperiod single-sourcing problem. Oper Res 51(6):922–939
26. French AP, Wilson JM (2002) Heuristic solution methods for the multilevel generalized assignment problem. J Heuristics 8:143–153
27. French AP, Wilson JM (2007) An lp-based heuristic procedure for the generalized assignment problem with special ordered sets. Comput Oper Res 34:2359–2369
28. Garey MR, Johnson DS (1990) Computers and Intractability; A Guide to the Theory of NP-Completeness. Freeman, New York
29. Gavish B, Pirkul H (1991) Algorithms for the multi-resource generalized assignment problem. Manag Sci 37:695–713

30. Geoffrion AM, Graves GW (1974) Multicommodity distribution system design by benders decomposition. Manag Sci 20(5):822–844
31. Glover F, Hultz J, Klingman D (1979) Improved computer based planning techniques, part ii. Interfaces 4:17–24
32. Gottlieb ES, Rao MR (1990) (1, *k*)-configuration facets for the generalized assignment problem. Math Program 46(1):53–60
33. Gottlieb ES, Rao MR (1990) The generalized assignment problem: Valid inequalities and facets. Math Stat 46:31–52
34. Guignard M, Rosenwein MB (1989) An improved dual based algorithm for the generalized assignment problem. Oper Res 37(4):658–663
35. Haddadi S (1999) Lagrangian decomposition based heuristic for the generalized assignment problem. Inf Syst Oper Res 37:392–402
36. Haddadi S, Ouzia H (2004) Effective algorithm and heuristic for the generalized assignment problem. Eur J Oper Res 153:184–190
37. Hajri-Gabouj S (2003) A fuzzy genetic multiobjective optimization algorithm for a multilevel generalized assignment problem. IEEE Trans Syst 33:214–224
38. Janak SL, Taylor MS, Floudas CA, Burka M, Mountziaris TJ (2006) Novel and effective integer optimization approach for the nsf panel-assignment problem: a multiresource and preference-constrained generalized assignment problem. Ind Eng Chem Res 45:258–265
39. Jörnsten K, Nasberg M (1986) A new lagrangian relaxation approach to the generalized assignment problem. Eur J Oper Res 27:313–323
40. Jörnsten KO, Varbrand P (1990) Relaxation techniques and valid inequalities applied to the generalized assignment problem. Asia-P J Oper Res 7(2):172–189
41. Klastorin TD (1979) An effective subgradient algorithm for the generalized assignment problem. Comp Oper Res 6:155–164
42. Klastorin TD (1979) On the maximal covering location problem and the generalized assignment problem. Manag Sci 25(1):107–112
43. Kogan K, Khmelnitsky E, Ibaraki T (2005) Dynamic generalized assignment problems with stochastic demands and multiple agent task relationships. J Glob Optim 31:17–43
44. Kogan K, Shtub A, Levit VE (1997) Dgap – the dynamic generalized assignment problem. Ann Oper Res 69:227–239
45. Kuhn H (1995) A heuristic algorithm for the loading problem in flexible manufacturing systems. Int J Flex Manuf Syst 7:229–254
46. Laguna M, Kelly JP, Gonzfilez-Velarde JL, Glover F (1995) Tabu search for the multilevel generalized assignment problem. Eur J Oper Res 82:176–189
47. Lawler E (1976) Combinatorial Optimization: Networks and Matroids. Holt, Rinehart, Winston, New York
48. Lin BMT, Huang YS, Yu HK (2001) On the variable-depth-search heuristic for the linear-cost generalized assignment problem. Int J Comput Math 77:535–544
49. Lorena LAN, Narciso MG (1996) Relaxation heuristics for a generalized assignment problem. Eur J Oper Res 91:600–610
50. Lorena LAN, Narciso MG, Beasley JE (2003) A constructive genetic algorithm for the generalized assignment problem. J Evol Optim
51. Lourenço HR, Serra D (1998) Adaptive approach heuristics for the generalized assignment problem. Technical Report 288, Department of Economics and Business, Universitat Pompeu Fabra, Barcelona
52. Lourenço HR, Serra D (2002) Adaptive search heuristics for the generalized assignment problem. Mathw Soft Comput 9(2–3):209–234
53. Martello S, Toth P (1981) An algorithm for the generalized assignment problem. In: Brans JP (ed) Operational Research '81, 9th IFORS Conference, North-Holland, Amsterdam, pp 589–603
54. Martello S, Toth P (1990) Knapsack Problems: Algorithms and Computer Implementations. Wiley, New York
55. Martello S, Toth P (1992) Generalized assignment problems. Lect Notes Comput Sci 650:351–369
56. Martello S, Toth P (1995) The bottleneck generalized assignment problem. Eur J Oper Res 83:621–638
57. Mazzola JB, Neebe AW (1988) Bottleneck generalized assignment problems. Eng Costs Prod Econ 14(1):61–65
58. Mazzola JB, Wilcox SP (2001) Heuristics for the multiresource generalized assignment problem. Nav Res Logist 48(6):468–483
59. Monfared MAS, Etemadi M (2006) The impact of energy function structure on solving generalized assignment problem using hopfield neural network. Eur J Oper Res 168:645–654
60. Morales DR, Romeijn HE (2005) Handbook of Combinatorial Optimization, supplement vol B. In: Du D-Z, Pardalos PM (eds) The Generalized Assignment Problem and extensions. Springer, New York, pp 259–311
61. Narciso MG, Lorena LAN (1999) Lagrangean/surrogate relaxation for generalized assignment problems. Eur J Oper Res 114:165–177
62. Nauss RM (2003) Solving the generalized assignment problem: an optimizing and heuristic approach. INFORMS J Comput 15(3):249–266
63. Nauss RM (2005) The elastic generalized assignment problem. J Oper Res Soc 55:1333–1341
64. Nowakovski J, Schwarzler W, Triesch E (1999) Using the generalized assignment problem in scheduling the rosat space telescope. Eur J Oper Res 112:531–541
65. Nutov Z, Beniaminy I, Yuster R (2006) A (1 — 1/*e*)-approximation algorithm for the generalized assignment problem. Oper Res Lett 34:283–288
66. Park JS, Lim BH, Lee Y (1998) A lagrangian dual-based branch-and-bound algorithm for the generalized multiassignment problem. Manag Sci 44(12S):271–275
67. Pigatti A, de Aragao MP, Uchoa E (2005) Stabilized branch-and-cut-and-price for the generalized assignment prob-

lem. In: Electronic Notes in Discrete Mathematics, vol 19 of 2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics, pp 385–395,
68. Osman IH (1995) Heuristics for the generalized assignment problem: simulated annealing and tabu search approaches. OR-Spektrum 17:211–225
69. Racer M, Amini MM (1994) A robust heuristic for the generalized assignment problem. Ann Oper Res 50(1):487–503
70. Romeijn HE, Morales DR (2000) A class of greedy algorithms for the generalized assignment problem. Discret Appl Math 103:209–235
71. Romeijn HE, Morales DR (2001) Generating experimental data for the generalized assignment problem. Oper Res 49(6):866–878
72. Romeijn HE, Piersma N (2000) A probabilistic feasibility and value analysis of the generalized assignment problem. J Comb Optim 4:325–355
73. Ronen D (1992) Allocation of trips to trucks operating from a single terminal. Comput Oper Res 19(5):445–451
74. Ross GT, Soland RM (1975) A branch and bound algorithm for the generalized assignment problem. Math Program 8:91–103
75. Ross GT, Soland RM (1977) Modeling facility location problems as generalized assignment problems. Manag Sci 24:345–357
76. Ross GT, Zoltners AA (1979) Weighted assignment models and their application. Manag Sci 25(7):683–696
77. Savelsbergh M (1997) A branch-and-price algorithm for the generalized assignment problem. Oper Res 45:831–841
78. Shmoys DB, Tardos E (1993) An approximation algorithm for the generalized assignment problem. Math Program 62:461–474
79. Shtub A (1989) Modelling group technology cell formation as a generalized assignment problem. Int J Prod Res 27:775–782
80. Srinivasan V, Thompson GL (1973) An algorithm for assigning uses to sources in a special class of transportation problems. Oper Res 21(1):284–295
81. Stützle T, Hoos H (1999) The Max-Min Ant System and Local Search for Combinatorial Optimization Problems. In: Voss S, Martello S, Osman IH, Roucairol C (eds) Meta-heuristics; Advances and trends in local search paradigms for optimization. Kluwer, Boston, pp 313–329
82. Toktas B, Yen JW, Zabinsky ZB (2006) Addressing capacity uncertainty in resource-constrained assignment problems. Comput Oper Res 33:724–745
83. Trick M (1992) A linear relaxation heuristic for the generalized assignment problem. Nav Res Logist 39:137–151
84. Trick MA (1994) Scheduling multiple variable-speed machines. Oper Res 42(2):234–248
85. Wilson JM (1997) A genetic algorithm for the generalised assignment problem. J Oper Res Soc 48:804–809
86. Wilson JM (2005) An algorithm for the generalized assignment problem with special ordered sets. J Heuristics 11:337–350
87. Yagiura M, Ibaraki T, Glover F (2004) An ejection chain approach for the generalized assignment problem. INFORMS J Comput 16:133–151
88. Yagiura M, Ibaraki T, Glover F (2006) A path relinking approach with ejection chains for the generalized assignment problem. Eur J Oper Res 169:548–569
89. Yagiura M, Yamaguchi T, Ibaraki T (1998) A variable depth search algorithm with branching search for the generalized assignment problem. Optim Method Softw 10:419–441
90. Yagiura M, Yamaguchi T, Ibaraki T (1999) A variable depth search algorithm for the generalized assignment problem. In: Voss S, Martello S, Osman IH, Roucairol C (eds) Meta-heuristics; Advances and Trends in Local Search paradigms for Optimization, Kluwer, Boston, pp 459–471
91. Zhang CW, Ong HL (2007) An efficient solution to biobjective generalized assignment problem. Adv Eng Softw 38:50–58
92. Zimokha VA, Rubinshtein MI (1988) R & d planning and the generalized assignment problem. Autom Remote Control 49:484–492

Generalized Benders Decomposition

GBD

Christodoulos A. Floudas
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 49M29, 90C11

Article Outline

Keywords
Formulation
Theoretical Development
The Primal Problem
The Master Problem
Projection Onto the **y**-Space
Dual of **V**
Dual Representation of $N(\mathbf{y})$
Algorithmic Development
How to Solve the Master Problem
General Algorithmic Statement of GBD
Finite Convergence of GBD
Variants of GBD
Variant 1 of GBD: V1-GBD
V1-GBD Under Separability

Variant 2 of GBD: V2-GBD
Variant 3 of GBD: V3-GBD
V3-GBD Under Separability
V3-GBD Without Separability
GBD in Continuous and Discrete-Continuous Optimization
See also
References

Keywords

Decomposition; Duality; Global optimization

The generalized Benders decomposition, GBD, [7] is a powerful theoretical and algorithmic approach for addressing *mixed integer nonlinear optimization* problems, as well as problems that require exploitation of their inherent mathematical structure via *decomposition* principles. A comprehensive analysis of the Generalized Benders Decomposition approach along with a variety of other approaches for mixed integer nonlinear optimization problems and their applications are presented in [3].

Formulation

[7] generalized the approach proposed by [1], for exploiting the structure of mathematical programming problems stated as:

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \subseteq \mathbf{R}^n \\ & \mathbf{y} \in \{0,1\}, \end{cases}$$

under the following conditions:

C1) $\mathbf{X}$ is a nonempty, convex set and the functions

$$f\colon\ \mathbf{R}^n \times \mathbf{R}^q \to \mathbf{R},$$
$$\mathbf{g}\colon\ \mathbf{R}^n \times \mathbf{R}^q \to \mathbf{R}^p$$

are convex for each fixed $\mathbf{y} \in \mathbf{Y} = \{0,1\}^q$, while the functions $\mathbf{h}\colon \mathbf{R}^n \times \mathbf{R}^l \to \mathbf{R}^m$ are linear for each fixed $\mathbf{y} \in \mathbf{Y} = \{0,1\}^q$.

C2) The set

$$\mathbf{Z}_\mathbf{y} = \left\{ \mathbf{z} \in \mathbf{R}^p : \begin{array}{c} \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}, \\ \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{0} \\ \text{for some } \mathbf{x} \in \mathbf{X} \end{array} \right\}$$

is closed for each fixed $\mathbf{y} \in \mathbf{Y}$.

C3) For each fixed $\mathbf{y} \in \mathbf{Y} \cap \mathbf{V}$, where

$$\mathbf{V} = \left\{ \mathbf{y}: \begin{array}{c} \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}, \\ \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{0}, \\ \text{for some } \mathbf{x} \in \mathbf{X} \end{array} \right\}$$

one of the following two conditions holds:

i) the resulting problem has a finite solution and has an optimal multiplier vector for the equalities and inequalities.

ii) the resulting problem is unbounded, that is, its objective function value goes to $-\infty$.

It should be noted that the above stated formulation is, in fact, a subclass of the problems for which the GBD of [7] can be applied. This is due to the specification of $\mathbf{y} \in \{0,1\}$, while [7] investigated the more general case of $\mathbf{Y} \subseteq \mathbf{R}^q$, and defined the vector of $\mathbf{y}$ variables as 'complicating' variables in the sense that if we fix $\mathbf{y}$, then:

a) the problem may be decomposed into a number of independent problems, each involving a different subvector of $\mathbf{x}$; or

b) the problem takes a well known special structure for which efficient algorithms are available; or

c) the problem becomes convex in $\mathbf{x}$ even though it is nonconvex in the joint $\mathbf{x}$-$\mathbf{y}$ domain, that is, it creates special structure.

Case a) may lead to parallel computations of the independent subproblems. Case b) allows the use of special-purpose algorithms (e. g., generalized network algorithms), while case c) invokes special structure from the convexity point of view that can be useful for the decomposition of nonconvex optimization problems. (e. g., [4]).

In the sequel, we concentrate on $\mathbf{Y} = \{0,1\}^q$ due to our interest in (MINLP; cf. also ► Mixed integer nonlinear programming) models. Note also that the analysis includes the equality constraints $\mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}$ which are not treated explicitly in [7].

Condition C2) is not stringent and it is satisfied if one of the following holds (in addition to C1), C3)):

i) $\mathbf{x}$ is bounded and closed and $\mathbf{h}(\mathbf{x},\mathbf{y})$, $\mathbf{g}(\mathbf{x},\mathbf{y})$ are continuous on $\mathbf{x}$ for each fixed $\mathbf{y} \in \mathbf{Y}$.

ii) there exists a point $\mathbf{z}_\mathbf{y}$ such that the set

$$\{\mathbf{x} \in \mathbf{X}:\ \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0},\ \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{z}_\mathbf{y}\}$$

is bounded and nonempty.

Note though that mere continuity of $\mathbf{h}(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$ on $\mathbf{X}$ for each fixed $\mathbf{y} \in \mathbf{Y}$ does not imply that condition C2) is satisfied. For instance, if $\mathbf{X} = [1, \infty]$ and $h(x, y) = x + y$, $g(x, y) = -1/x$, then $z_y = (-\infty, 0)$ which is not closed since for $x \to \infty$, $g(x, y) \to -\infty$.

Note that the set $\mathbf{V}$ represents the values of $\mathbf{y}$ for which the resulting problem is feasible with respect to $\mathbf{x}$. In others words, $\mathbf{V}$ denotes the values of $\mathbf{y}$ for which there exists a feasible $\mathbf{x} \in \mathbf{X}$ for $\mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0}$, $\mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}$. Then the intersection of $\mathbf{y}$ and $\mathbf{V}$, $\mathbf{Y} \cap \mathbf{V}$, represents the *projection* of the feasible region of the original problem onto the $\mathbf{y}$-space.

Condition C3) is satisfied if a first order constraint qualification holds for the resulting problem after fixing $\mathbf{y} \in \mathbf{Y} \cap \mathbf{V}$.

The basic idea in generalized Benders decomposition, GBD, is the generation, at each iteration, of an upper bound and a lower bound on the sought solution of the MINLP model. The upper bound results from the *primal problem*, while the lower bound results form the *master problem*. The primal problem corresponds to the original problem with fixed $\mathbf{y}$-variables (i. e., it is in the $\mathbf{x}$-space only) and its solution provides information about the upper bound and the Lagrange multipliers associated with the equality and inequality constraints. The master problem is derived via nonlinear *duality theory*, makes use of the Lagrange multipliers obtained in the primal problem, and its solution provides information about the lower bound, as well as the next set of fixed $\mathbf{y}$-variables to be used subsequently in the primal problem. As the iterations proceed, it is shown that the sequence of updated upper bounds is nonincreasing, the sequence of lower bounds is nondecreasing, and that the sequences converge in a finite number of iterations.

Theoretical Development

This Section presents the theoretical development of the generalized Benders decomposition, GBD. The primal problem is analyzed first for the feasible and infeasible cases. Subsequently, the theoretical analysis for the derivation of the master problem is presented.

The Primal Problem

The primal problem results from fixing the $\mathbf{y}$ variables to a particular 0–1 combination, which we denote as $\mathbf{y}^k$ where k stands for the iteration counter. The formulation of the primal problem $\mathrm{P}(\mathbf{y}^k)$, at iteration k is:

$$\mathrm{P}(\mathbf{y}^k) \quad \begin{cases} \min\limits_{\mathbf{x}} & f(\mathbf{x}, \mathbf{y}^k) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}^k) \leq \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \subseteq \mathbf{R}^n. \end{cases}$$

Note that due to conditions C1) and C3i), the solution of the primal problem $\mathrm{P}(\mathbf{y}^k)$ is its global solution.

We will distinguish the two cases 'feasible primal' and 'infeasible primal', and describe the analysis for each case separately.

- Feasible primal.
  If the primal problem at iteration k is feasible, then its solution provides information on $\mathbf{x}^k$, $f(\mathbf{x}^k, \mathbf{y}^k)$ which is the upper bound, and the optimal multiplier vectors λ^k, μ^k for the equality and inequality constraints. Subsequently, using this information we can formulate the Lagrange function as

  $$L(\mathbf{x}, \mathbf{y}, \lambda^k, \mu^k) = f(\mathbf{x}, \mathbf{y}) + \lambda^{k\top}\mathbf{h}(\mathbf{x}, \mathbf{y}) + \mu^{k\top}\mathbf{g}(\mathbf{x}, \mathbf{y}).$$

- Infeasible primal.
  If the primal is detected by the NLP solver to be infeasible, then we consider its constraints

  $$\mathbf{h}(\mathbf{x}, \mathbf{y}^k) = \mathbf{0},$$
  $$\mathbf{g}(\mathbf{x}, \mathbf{y}^k) \leq \mathbf{0},$$
  $$\mathbf{x} \in \mathbf{X} \subseteq \mathbf{R}^n,$$

  where the set $\mathbf{X}$, for instance, consists of lower and upper bounds on the $\mathbf{x}$ variables. To identify a feasible point we can minimize an l_1 or l_∞ sum of constraint violations. An l_1-minimization problem can be formulated as:

  $$\begin{cases} \min\limits_{\mathbf{x}\in\mathbf{X}} & \sum\limits_{i=1}^{p} \alpha_i \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & g_i(\mathbf{x}, \mathbf{y}^k) \leq \alpha_i, \quad i = 1, \ldots, p, \\ & \alpha_i \geq 0, \quad i = 1, \ldots, p, \end{cases}$$

  Note that if $\sum_{i=1}^{p} \alpha_i = 0$, then a feasible point has been determined.

Also note that by defining as

$$\alpha^+ = \max(0, \alpha)$$

and

$$g_i^+(\mathbf{x}, \mathbf{y}^k) = \max\left(0, g_i(\mathbf{x}, \mathbf{y}^k)\right),$$

the l_1-minimization problem is stated as:

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbf{X}} & \sum\limits_{i=1}^{P} g_i^+ \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}^k) = \mathbf{0}. \end{cases}$$

An l_∞-minimization problem can be stated similarly as:

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbf{X}} \max\limits_{1,\ldots,p} & g_i^+(\mathbf{x}, \mathbf{y}^k) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}^k) = \mathbf{0}. \end{cases}$$

Alternative feasibility minimization approaches aim at keeping feasibility in any constraint residual once it has been established. An l_1-minimization in these approaches takes the form:

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbf{X}} & \sum\limits_{i\in\mathbf{I}'} g_i^+(\mathbf{x}, \mathbf{y}^k) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & g_i(\mathbf{x}, \mathbf{y}^k) \le 0, \quad i \in \mathbf{I}, \end{cases}$$

where $\mathbf{I}$ is the set of feasible constraints and $\mathbf{I}'$ is the set of infeasible constraints. Other methods seek feasibility of the constraints one at a time while maintaining feasibility for inequalities indexed by $i \in \mathbf{I}$. This feasibility problem is formulated as:

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbf{X}} & \sum\limits_{i\in\mathbf{I}'} w_i g_i^+(\mathbf{x}, \mathbf{y}^k) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & g_i(\mathbf{x}, \mathbf{y}^k) \le 0, \quad i \in \mathbf{I}, \end{cases}$$

and it is solved at any one time.

To include all mentioned possibilities [2] formulated a general feasibility problem (FP) defined as:

$$(FP)\begin{cases} \min\limits_{\mathbf{x}\in\mathbf{X}} & \sum\limits_{i\in\mathbf{I}'} w_i g_i^+(\mathbf{x}, \mathbf{y}^k) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & g_i(\mathbf{x}, \mathbf{y}^k) \le 0, \quad i \in \mathbf{I}. \end{cases}$$

The weights w_i are nonnegative and not all are zero. Note that with $w_i = 1$, $i \in \mathbf{I}'$, we obtain the l_1-minimization. Also in the l_∞-minimization, there exist nonnegative weights at the solution such that

$$\sum w_i = 1$$

and $w_i = 0$ if $g_i(\mathbf{x}, \mathbf{y}^k)$ does not attain the maximum value.

Note that infeasibility in the primal problem is detected when a solution of (FP) is obtained for which its objective value is greater than zero.

The solution of the feasibility problem (FP) provides information on the Lagrange multipliers for the equality and inequality constraints which are denoted as $\overline{\lambda}^k$, $\overline{\mu}^k$ respectively. Then, the Lagrange function resulting from on infeasible primal problem at iteration k can be defined as:

$$\overline{L}^k(\mathbf{x}, \mathbf{y}, \overline{\lambda}^k, \overline{\mu}^k) = \overline{\lambda}^{k\top}\mathbf{h}(\mathbf{x}, \mathbf{y}) + \overline{\mu}^{k\top}\mathbf{g}(\mathbf{x}, \mathbf{y}).$$

It should be noted that two different types of Lagrange functions are defined depending on whether the primal problem is feasible or infeasible. Also, the upper bound is obtained only from the feasible primal problem.

The Master Problem

The derivation of the master problem in the GBD makes use of nonlinear duality theory, and is characterized by the following three key ideas:

i) projection onto the **y**-space;
ii) dual representation of **V**; and
iii) dual representation of the projection of the original problem on the **y**-space.

In the sequel, the theoretical analysis involved in these three key ideas is presented.

Projection Onto the y-Space

The original problem can be written as:

$$\begin{cases} \min\limits_{\mathbf{y}} \inf\limits_{\mathbf{x}} & f(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \le \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \\ & \mathbf{y} \in \mathbf{Y} = \{0, 1\}^q, \end{cases} \tag{1}$$

where the min operator has been written separately for $\mathbf{y}$ and $\mathbf{x}$. Note that it is infimum with respect to $\mathbf{x}$ since for given $\mathbf{y}$ the inner problem may be unbounded. Let us define $v(\mathbf{y})$ as:

$$v(\mathbf{y}) = \begin{cases} \inf\limits_{\mathbf{x}} & f(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\ & \mathbf{x} \in \mathbf{X}. \end{cases} \tag{2}$$

Note that $v(\mathbf{y})$ is parametric in the $\mathbf{y}$ variables and therefore, from its definition corresponds to the optimal value of the original problem for fixed $\mathbf{y}$ (i. e., the primal problem P($\mathbf{y}^k$) for $\mathbf{y} = \mathbf{y}^k$).

Let us also define the set $\mathbf{V}$ as:

$$\mathbf{V} = \left\{ \mathbf{y}: \begin{array}{c} \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0}, \\ \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\ \text{for some } \mathbf{x} \in \mathbf{X} \end{array} \right\}. \tag{3}$$

Then, problem (1) can be written as:

$$\begin{cases} \min\limits_{\mathbf{y}} & v(\mathbf{y}) \\ \text{s.t.} & \mathbf{y} \in \mathbf{Y} \cap \mathbf{V}, \end{cases} \tag{4}$$

where $v(\mathbf{y})$ and $\mathbf{V}$ are defined by (2) and (3) respectively.

Problem (4) is the projection of the original problem onto the $\mathbf{y}$-space. Note also that in (3) $\mathbf{y} \in \mathbf{Y} \cap \mathbf{V}$ since the projection needs to satisfy the feasibility considerations.

Having defined the projection problem onto the $\mathbf{y}$-space, we can now state the theoretical result of [7].

Theorem 1 (Projection)

i) If ($\mathbf{x}^*$, $\mathbf{y}^*$) is optimal in the original problem, then $\mathbf{y}^*$ is optimal in (4).

ii) If the original problem is infeasible or has unbounded solution, then the same is true for (4) and vice versa.

Note that the difficulty in the original problem is due to the fact that $v(\mathbf{y})$ and $\mathbf{V}$ are known only implicitly via (2) and (3).

To overcome the aforementioned difficulty we have to introduce the dual representation of $\mathbf{V}$ and $v(\mathbf{y})$.

Dual of V

The dual representation of $\mathbf{V}$ will be invoked in terms of the intersection of a collection of regions that contain it, and it is described in the following theorem, due to [7].

Theorem 2 (Dual of V) *Assuming conditions C1) and C2), a point $\mathbf{y} \in \mathbf{Y}$ belongs also to the set $\mathbf{V}$ if and only if it satisfies the (finite) system:*

$$\begin{aligned} & 0 \geq \inf \overline{L}(\mathbf{x}, \mathbf{y}, \overline{\lambda}, \overline{\mu}), \quad \forall \overline{\lambda}, \overline{\mu} \in \Lambda, \\ & \Lambda = \left\{ \overline{\lambda} \in \mathbf{R}^m, \overline{\mu} \in \mathbf{R}^p : \ \overline{\mu} \geq \mathbf{0}, \ \sum_{i=1}^{p} \overline{\mu}_i . = 1 \right\} \end{aligned} \tag{5}$$

Note that (5) is an infinite system because it has to be satisfied for all $\overline{\lambda}, \overline{\mu} \in \Lambda$. The dual representation of the set $\mathbf{V}$ needs to be invoked so as to generate a collection of regions that contain it (i. e., system (5) and system (5) corresponds to the set of constraints that have to be incorporated for the case of infeasible primal problems.

Note that if the primal is infeasible and we make use of the l_1-minimization of the type:

$$\begin{cases} \min\limits_{\mathbf{x}} & \sum\limits_{i \in \mathbf{I}} \alpha_i \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & g_i(\mathbf{x}, \mathbf{y}^k) \leq \alpha_i, \quad i \in \mathbf{I}, \\ & \mathbf{x} \in \mathbf{X}, \end{cases} \tag{6}$$

then the set Λ results from a straightforward application of the KKT gradient conditions to problem (6) with respect to α_i.

Having introduced the dual representation of the set $\mathbf{V}$, which corresponds to infeasible primal problems, we can now invoke the dual representation of $v(\mathbf{y})$.

Dual Representation of $N(\mathbf{y})$

The dual representation of $v(\mathbf{y})$ will be in terms of the pointwise infimum of a collection of functions that support it, and it is described in the following theorem, due to [7].

Theorem 3 (Dual of $v(\mathbf{y})$)

$$
\begin{aligned}
v_{\mathbf{y}} = & \begin{cases} \inf\limits_{\mathbf{x}} & f(\mathbf{x},\mathbf{y}) \\ s.t. & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & g(\mathbf{x},\mathbf{y}) \le \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \end{cases} \\
& = \sup_{\lambda,\mu \ge 0} \inf_{\mathbf{x}\in\mathbf{X}} L(\mathbf{x},\mathbf{y},\lambda,\mu), \\
& \forall \mathbf{y} \in \mathbf{Y} \cap \mathbf{V}, \\
& L(\mathbf{x},\mathbf{y},\lambda,\mu) \\
& = f(\mathbf{x},\mathbf{y}) + \lambda^{\top}\mathbf{h}(\mathbf{x},\mathbf{y}) + \mu^{\top}\mathbf{g}(\mathbf{x},\mathbf{y}).
\end{aligned} \tag{7}
$$

The equality of $v(\mathbf{y})$ and its dual is due to having the strong duality theorem satisfied because of conditions C1), C2) and C3).

Substituting (7) for $v(\mathbf{y})$ and (5) for $\mathbf{y} \in \mathbf{Y} \cap \mathbf{V}$ into problem (4), (which is equivalent to (1)), we obtain:

$$
\begin{cases} \min\limits_{\mathbf{y}\in\mathbf{Y}} \sup\limits_{\lambda,\mu\ge 0} \inf\limits_{\mathbf{x}\in\mathbf{X}} & L(\mathbf{x},\mathbf{y},\lambda,\mu) \\ \text{s.t.} & 0 \ge \inf\limits_{\mathbf{x}\in\mathbf{X}} \overline{L}(\mathbf{x},\mathbf{y},\overline{\lambda},\overline{\mu}). \end{cases}
$$

Using the definition of supremum as the lowest upper bound and introducing a scalar μ_B we obtain:

$$
(M) \begin{cases} \min\limits_{\mathbf{y}\in\mathbf{Y},\mu_B} & \mu_B \\ \text{s.t.} & \mu_B \ge \inf\limits_{\mathbf{x}\in\mathbf{X}} L(\mathbf{x},\mathbf{y},\lambda,\mu), \\ & \quad \forall\lambda, \forall\mu \ge 0, \\ & 0 \ge \inf\limits_{\mathbf{x}\in\mathbf{X}} \overline{L}(\mathbf{x},\mathbf{y},\overline{\lambda},\overline{\mu}), \\ & \quad \forall \left(\overline{\lambda},\overline{\mu}\right) \in \Lambda, \end{cases}
$$

where

$$
\begin{aligned}
& L(\mathbf{x},\mathbf{y},\lambda,\mu) = f(\mathbf{x},\mathbf{y}) \\
& + \lambda^{\top}\mathbf{h}(\mathbf{x},\mathbf{y}) + \mu^{\top}\mathbf{g}(\mathbf{x},\mathbf{y}), \\
& L(\mathbf{x},\mathbf{y},\overline{\lambda},\overline{\mu}) = \overline{\lambda}^{\top}\mathbf{h}(\mathbf{x},\mathbf{y}) + \overline{\mu}^{\top}\mathbf{g}(\mathbf{x},\mathbf{y}),
\end{aligned}
$$

which is called the *master problem*.

If we assume that the optimum solution of $v(\mathbf{y})$ in (2) is bounded for all $\mathbf{y} \in \mathbf{Y} \cap \mathbf{V}$, then we can replace the infimum with a minimum. Subsequently, the master problem will be as follows:

$$
\begin{cases} \min\limits_{\mathbf{y}\in\mathbf{Y},\mu_B} & \mu_B \\ \text{s.t.} & \mu_B \ge \min\limits_{\mathbf{x}\in\mathbf{X}} L(\mathbf{x},\mathbf{y},\lambda,\mu), \\ & \quad \forall\lambda,\mu \ge 0, \\ & 0 \ge \min\limits_{\mathbf{x}\in\mathbf{X}} \overline{L}(\mathbf{x},\mathbf{y},\overline{\lambda},\overline{\mu}), \\ & \quad \forall \left(\overline{\lambda},\overline{\mu}\right) \in \Lambda, \end{cases}
$$

where $L(\mathbf{x},\mathbf{y},\lambda,\mu)$ and $\overline{L}(\mathbf{x},\mathbf{y},\overline{\lambda},\overline{\mu})$ are defined as before.

Note that the master problem involves, an infinite number of constraints and hence we would need to consider a relaxation of the master (e. g., by dropping a number of constraints) which will represent a lower bound on the original problem. Note also that the master problem features an outer optimization problem with respect to $\mathbf{y} \in Y$ and inner optimization problems with respect to $\mathbf{x}$ which are in fact parametric in $\mathbf{y}$. It is this outer-inner nature that makes the solution of even a relaxed master problem difficult.

The inner minimization problems

$$
\begin{aligned}
& \min_{\mathbf{x}\in\mathbf{X}} L(\mathbf{x},\mathbf{y},\lambda,\mu), \quad \forall\lambda,\ \forall\mu \ge 0, \\
& \min_{\mathbf{x}\in\mathbf{X}} \overline{L}(\mathbf{x},\mathbf{y},\overline{\lambda},\overline{\mu}), \quad \forall \left(\overline{\lambda},\overline{\mu}\right) \in \Lambda,
\end{aligned}
$$

are functions of $\mathbf{y}$ and can be interpreted as support functions of $v(\mathbf{y})$. ($\xi(\mathbf{y})$ is a support function of $v(\mathbf{y})$ at point $\mathbf{y_o}$ if and only if $\xi(\mathbf{y}) = v(\mathbf{y})$ and $\xi(\mathbf{y}) \le v(\mathbf{y})$, $\forall \mathbf{y} \ne \mathbf{y_o}$.) If the support functions are linear in $\mathbf{y}$, then the master problem approximates $v(\mathbf{y})$ by tangent hyperplanes and we can conclude that $v(\mathbf{y})$ is convex in $\mathbf{y}$. Note that $v(\mathbf{y})$ can be convex in $\mathbf{y}$ even though the original problem is nonconvex in the joint $\mathbf{x}$-$\mathbf{y}$ space (see [5]).

In the sequel, we will define the aforementioned minimization problems in terms of the notion of support functions, that is:

$$
\begin{aligned}
& \xi(\mathbf{y};\lambda,\mu) = \min_{\mathbf{x}\in\mathbf{X}} L(\mathbf{x},\mathbf{y},\lambda,\mu), \\
& \forall\lambda, \quad \forall\mu \ge 0, \\
& \overline{\xi}(\mathbf{y};\overline{\lambda},\overline{\mu}) = \min_{\mathbf{x}\in\mathbf{X}} \overline{L}(\mathbf{x},\mathbf{y},\overline{\lambda},\overline{\mu}), \\
& \forall \left(\overline{\lambda},\overline{\mu}\right) \in \Lambda.
\end{aligned}
$$

Algorithmic Development

In the previous Section we discussed the primal and master problem for the GBD. We have the primal problem being a (linear or) nonlinear programming, NLP, problem that can be solved via available local NLP solvers (e. g., MINOS 5.3). The master problem, however, consists of outer and inner optimization problems, and approaches towards attaining its solution are discussed in the following.

How to Solve the Master Problem

The master problem has as constraints the two inner optimization problems (i. e., for the case of feasible primal and infeasible primal problems) which however need to be considered for all λ and all $\mu \geq 0$ (i.e feasible primal) and all $(\overline{\lambda}, \overline{\mu}) \in \Lambda$ (i. e., infeasible). This implies that the master problem has a very large number of constraints.

The most natural approach for solving the master problem is *relaxation* [7]. The basic idea in the relaxation approach consists of the following:

i) ignore all but a few of the constraints that correspond to the inner optimization problems (e. g., consider the inner optimization problems for specific or fixed multipliers (λ^1, μ^1) or $(\overline{\lambda}^1, \overline{\mu}^1)$);
ii) solve the relaxed master problem and check whether the resulting solution satisfies all of the ignored constraints. If not, then generate and add to the relaxed master problem one or more of the violated constraints and solve the new relaxed master problem again;
iii) continue until a relaxed master problem satisfies all of the ignored constraints, which implies that an optimal solution at the master problem has been obtained or until a termination criterion indicates that a solution of acceptable accuracy has been found.

General Algorithmic Statement of GBD

Assuming that the problem has a finite optimal value, [7] stated the general algorithm for GBD listed below.

Note that a feasible initial primal is needed in Step 1. However, this does not restrict the GBD since it is possible to start with an infeasible primal problem. In this case, after detecting that the primal is infeasible, Step 3b is applied in which a support function $\overline{\xi}$ is employed.

Note that Step 1 could be altered, that is instead of solving the primal problem we could solve a continuous relaxation of the original problem in which the $\mathbf{y}$ variables are treated as continuous bounded by zero and one:

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \\ & \mathbf{0} \leq \mathbf{y} \leq \mathbf{1}. \end{cases} \tag{8}$$

If the solution of (8) is integral, then we terminate. If there exist fractional values of the $\mathbf{y}$ variables, then these can be rounded to the closest integer values and subsequently these can be used as the starting $\mathbf{y}^1$ vector with the possibility of the resulting primal problem being feasible or infeasible.

Note also that in Step 1, Step 3a and Step 3b a rather important assumption is made, that is we can find the support functions ξ and $\overline{\xi}$ for the given values of the multiplier vectors (λ, μ) and $(\overline{\lambda}, \overline{\mu})$. The determination of these support functions can not be achieved in general since these are parametric functions of $\mathbf{y}$ and result from the solution of the inner optimization problems.

Their determination in the general case requires a global optimization approach as the one proposed by [5,6]. There exist however, a number of special cases for which the support functions can be obtained explicitly as functions of the $\mathbf{y}$ variables. We will discuss these special cases in the next Section. If however, it is not possible to obtain explicitly expressions of the support functions in terms of the $\mathbf{y}$ variables, then assumptions need to be introduced for their calculation. These assumptions, as well as the resulting variants of GBD will be discussed in the next Section. The point to note here is that the validity of lower bounds with these variants of GBD will be limited by the imposed assumptions.

Note that the relaxed master problem (see Step 2) in the first iteration will have as a constraint one support function that corresponds to feasible primal and will be of the form:

$$\begin{cases} \min\limits_{\mathbf{y}\in\mathbf{Y},\mu_B} & \mu_B \\ \text{s.t.} & \mu_B \geq \xi(\mathbf{y};\lambda^1,\mu^1). \end{cases} \tag{9}$$

1 Let an initial point $\mathbf{y}^1 \in \mathbf{Y} \cap \mathbf{V}$ (i.e., by fixing $\mathbf{y} = \mathbf{y}^1$, we have a feasible primal). Solve the resulting primal problem $P(\mathbf{y}^1)$ and obtain an optimal primal solution $\mathbf{x}^1$ and optimal multipliers; vectors λ^1, μ^1. Assume that you can find, somehow, the support function $\xi(\mathbf{y}; \lambda^1, \mu^1)$ for the obtained multipliers λ^1, μ^1. Set the counters $k = 1$ for feasible and $l = 1$ for infeasible and the current upper bound UBD $= v(\mathbf{y}^1)$. Select the convergence tolerance $\epsilon \geq 0$.

2 Solve the relaxed master problem:

$$(\text{RM}) \begin{cases} \min\limits_{\mathbf{y}\in\mathbf{Y},\mu_B} \mu_B \\ \text{s.t.} \quad \mu_B \geq \xi(\mathbf{y}; \lambda^k, \mu^k), \\ \qquad\quad k = 1, \ldots, K, \\ \qquad 0 \geq \overline{\xi}(\mathbf{y} : \overline{\lambda}^l, \overline{\mu}^l), \\ \qquad\quad l = 1, \ldots, \Lambda. \end{cases}$$

Let $(\hat{\mathbf{y}}, \hat{\mu}_B)$ be an optimal solution of the above relaxed master problem. $\hat{\mu}_B$ is a lower bound on the original problem, that is the current lower bound is LBD $= \hat{\mu}_B$. If UBD $-$ LBD $\leq \epsilon$, then terminate.

3 Solve the primal problem for $\mathbf{y} = \hat{\mathbf{y}}$, that is the problem $P(\hat{\mathbf{y}})$. Then we distinguish two cases: feasible and infeasible primal:

3a Feasible Primal $P(\hat{\mathbf{y}})$.
The primal has $v(\hat{\mathbf{y}})$ finite with an optimal solution $\hat{\mathbf{x}}$ and optimal multiplier vectors $\hat{\lambda}, \hat{\mu}$. Update the upper bound UBD $= \min\{\text{UBD}, v(\hat{\mathbf{y}})\}$. If UBD $-$ LBD $\leq \epsilon$, then terminate. Otherwise, set $k = k + 1$, $\lambda^k = \hat{\lambda}$, and $\mu^k = \hat{\mu}$. Return to Step 2, assuming we can somehow determine the support function $\xi(\mathbf{y}; \lambda^{k+1}, \mu^{k+1})$.

3b Infeasible Primal $P(\hat{\mathbf{y}})$.
The primal does not have a feasible solution for $\mathbf{y} = \hat{\mathbf{y}}$. Solve a feasibility problem (e.g., then l_1-minimization) to determine the multiplier vectors $\hat{\overline{\lambda}}, \hat{\overline{\mu}}$ of the feasibility problem. Set $l = l + 1$, $\overline{\lambda}^l = \hat{\overline{\lambda}}$, and $\overline{\mu}^l = \hat{\overline{\mu}}$. Return to Step 2, assuming we can somehow determine the support function $\xi(\mathbf{y}; \overline{\lambda}^{l+1}, \overline{\mu}^{l+1})$.

In the second iteration, if the primal is feasible and (λ^2, μ^2) are its optimal multiplier vectors, then the relaxed master problem will feature two constraints and will be of the form:

$$\begin{cases} \min\limits_{\mathbf{y}\in\mathbf{Y},\mu_B} & \mu_B \\ \text{s.t.} & \mu_B \geq \xi(\mathbf{y}; \lambda^1, \mu^1) \\ & \mu_B \geq \xi(\mathbf{y}; \lambda^2, \mu^2). \end{cases} \tag{10}$$

Note that in this case the relaxed master problem (10), will have a solution that is greater or equal to the solution of (9). This is due to having the additional constraint. Therefore, we can see that the sequence of lower bounds that is created from the solution of the relaxed master problems is nondecreasing. A similar argument holds true in the case of having infeasible primal in the second iteration.

Note that since the upper bounds are produced by fixing the $\mathbf{y}$ variables to different 0–1 combinations, there is no reason for the upper bounds to satisfy any monotonicity property. If we consider however the updated upper bounds (i. e., UBD $= \min_k v(\mathbf{y}^k)$), then the sequence for the updated upper bounds is monotonically nonincreasing since by their definition we always keep the best (least) upper bound.

The termination criterion for GBD is based on the difference between the updated upper bound and the current lower bound. If this difference is less than or equal to a prespecified tolerance $\varepsilon \geq 0$ then we terminate. Note though that if we introduce in the relaxed master integer cuts that exclude the previously found 0–1 combinations then the termination criterion can be met by having found an infeasible master problem (i. e., there is no 0–1 combination that makes it feasible).

Finite Convergence of GBD

[7] proved finite convergence of the GBD algorithm which is as follows:

Theorem 4 (Finite convergence) *If C1), C2) and C3) hold and* $\mathbf{Y}$ *is a discrete set, then the GBD algorithm terminates in a finite number of iterations for any given* $\epsilon > 0$ *and even for* $\epsilon = 0$.

Variants of GBD

In the previous Section we discussed the general algorithmic statement of GBd and pointed out a key assumption made with respect to the calculation of the

support functions $\xi(\mathbf{y};\lambda, \mu)$ and $\overline{\xi}(\mathbf{y};\overline{\lambda}, \overline{\mu})$ from the feasible and infeasible primal problems respectively. In this section, we will discuss a number of *variants of GBD* that result from addressing the calculation of the aforementioned support functions either rigorously for special cases or making assumptions that may not provide valid lower bounds in the general case.

Variant 1 of GBD: V1-GBD

This variant of GBD is based on the following assumption that was denoted by [7] as Property (P):

Theorem 5 (Property (P)) *For every λ and $\mu \geq 0$, the infimum of $L(\boldsymbol{x}, \boldsymbol{y}, \lambda, \mu)$ with respect to $\boldsymbol{x} \in \boldsymbol{X}$ (i. e., the support $\xi(\boldsymbol{y};\lambda, \mu)$) can be taken independently of $\boldsymbol{y}$ so that the support function $\xi(\boldsymbol{y};\lambda, \mu)$ can be obtained explicitly with little or no more effort than is required to evaluate it at a single value of $\boldsymbol{y}$. Similarly, the support function $\overline{\xi}(\mathbf{y};\overline{\lambda}, \overline{\mu})$, $(\overline{\lambda}, \overline{\mu}) \in \Lambda$ can be obtained explicitly.*

[7] identified the following two important classes of problems where Property (P) holds:

- Class 1: f, $\mathbf{h}$, $\mathbf{g}$ are linearly separable in $\mathbf{x}$ and $\mathbf{y}$.
- Class 2: Variable factor programming.

In class-1 problems, we have

$$\begin{aligned} f(\mathbf{x}, \mathbf{y}) &= f_1(\mathbf{x}) + f_2(\mathbf{y}), \\ \mathbf{h}(\mathbf{x}, \mathbf{y}) &= \mathbf{h}_1(\mathbf{x}) + \mathbf{h}_2(\mathbf{y}), \\ \mathbf{g}(\mathbf{x}, \mathbf{y}) &= \mathbf{g}_1(\mathbf{x}) + \mathbf{g}_2(\mathbf{y}). \end{aligned}$$

In class-2 problems, we have

$$\begin{aligned} f(\mathbf{x}, \mathbf{y}) &= -\sum_i f_i(\mathbf{x}^i)\mathbf{y}_i, \\ \mathbf{g}(\mathbf{x}, \mathbf{y})_j &= \sum_i \mathbf{x}^i \mathbf{y}_i - c. \end{aligned}$$

In [8] problems, we have

$$\begin{aligned} f(\mathbf{x}, \mathbf{y}) &= \sum_k \sum_i f_i(\mathbf{x}_i(k))\mathbf{y}_i + \sum_i g_i(\mathbf{y}_i), \\ \mathbf{g}(\mathbf{x}, \mathbf{y})_j &= -\sum_i \mathbf{x}_i(k)\mathbf{y}_i - L(k). \end{aligned}$$

In the sequel, we will discuss the v1-GBD for class-1 problems since this by itself defines an interesting mathematical structure for which other algorithms (e. g., outer approximation) has been developed.

V1-GBD Under Separability

Under the *separability assumption*, the support functions $\xi(\mathbf{y};\lambda^k, \mu^k)$ and $\overline{\xi}(\mathbf{y};\overline{\lambda}^l, \overline{\mu}^l)$ can be obtained as explicit functions of $\mathbf{y}$ since:

$$\begin{aligned} &\xi(\mathbf{y};\lambda^k, \mu^k) = \min_{\mathbf{x}\in\mathbf{X}} L(\mathbf{x}, \mathbf{y}, \lambda^k \mu^k) \\ &= \min_{\mathbf{x}\in\mathbf{X}} \{f(\mathbf{x}, \mathbf{y})) + \lambda^{k\top}\mathbf{h}(\mathbf{x}, \mathbf{y}) + \mu^{k\top}\mathbf{g}(\mathbf{x}, \mathbf{y})\} \\ &= \min_{\mathbf{x}\in\mathbf{X}} \{f_1(\mathbf{x}) + f_2(\mathbf{y}) \\ &\quad + \lambda^{k\top}(h_1(\mathbf{x}) + h_2(\mathbf{y})) + \mu^{k\top}(\mathbf{g}_1(\mathbf{x}) + \mathbf{g}_2(\mathbf{y}))\} \\ &= f_2(\mathbf{y}) + \lambda^{k\top}\mathbf{h}_2(\mathbf{y}) + \mu^{k\top} g_2(\mathbf{x}) \\ &\quad + \min_{\mathbf{x}\in\mathbf{X}} [f_1(\mathbf{x}) + \lambda^{k\top} h_1(\mathbf{x}) + \mu^{k\top}\mathbf{g}_1(\mathbf{x})]. \end{aligned}$$

Note that due to separability we end up with an explicit function of $\mathbf{y}$ and a problem only in $\mathbf{x}$ that can be solved independently.

Similarly, the support function $\overline{\xi}(\mathbf{y};\overline{\lambda}^l, \overline{\mu}^l)$ is

$$\begin{aligned} &\overline{\xi}(\mathbf{y};\overline{\lambda}^l, \overline{\mu}^l) = \min_{\mathbf{x}\in\mathbf{X}} \overline{L}(\mathbf{x}, \mathbf{y}, \overline{\lambda}^l, \overline{\mu}^l) \\ &= \min_{\mathbf{x}\in\mathbf{X}} \{\overline{\lambda}^{l\top}\mathbf{h}(\mathbf{x}, \mathbf{y}) + \overline{\mu}^{l\top}\mathbf{g}(\mathbf{x}, \mathbf{y})\} \\ &= \min_{\mathbf{x}\in\mathbf{X}} \{\overline{\lambda}^{l\top} (h_1(\mathbf{x}, \mathbf{y}) + h_2(\mathbf{x}, \mathbf{y})) \\ &\quad + \overline{\mu}^{l\top} \big(g_1(\mathbf{x}, \mathbf{y}) + g_2(\mathbf{x}, \mathbf{y})\big)\} \\ &= \overline{\lambda}^{l\top} h_2(\mathbf{y}) + \overline{\mu}^{l\top} g_2(\mathbf{y}) \\ &\quad + \min_{\mathbf{x}\in\mathbf{X}} \Big[\overline{\lambda}^{l\top} h_1(\mathbf{x}) + \overline{\mu}^{l\top} g_1(\mathbf{x})\Big]. \end{aligned}$$

Note that to solve the independent problems in $\mathbf{x}$, we need to know the multiplier vectors (λ^k, μ^k) and $(\overline{\lambda}^l, \overline{\mu}^l)$ from feasible and infeasible primal problems respectively.

Under the separability assumption, the primal problem for fixed $\mathbf{y} = \mathbf{y}^k$ takes the form

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbf{X}} & f_1(\mathbf{x}) + f_2(\mathbf{y}^k) \\ \text{s.t.} & h_1(\mathbf{x}) = -h_2(\mathbf{y}^k) \\ & g_1(\mathbf{x}) \leq -g_2(\mathbf{y}^k). \end{cases}$$

Now, we can state the algorithmic procedure for the v1-GBD under the separability assumption.

Note that if in addition to the separability of $\mathbf{x}$ and $\mathbf{y}$, we assume that $\mathbf{y}$ participates linearly (i. e., conditions

1 Let an initial point $\mathbf{y}^1 \in \mathbf{Y} \cap \mathbf{V}$. Solve the primal $P(\mathbf{y}^1)$ and obtain an optimal solution $\mathbf{x}^1$, and multiplier vectors λ^1, μ^1. Set the counters $k = 1$, $l = 1$, and UBD $= v(\mathbf{y}^1)$. Select the convergence tolerance $\epsilon \geq 0$.

2 Solve the relaxed master problem

$$\begin{cases} \min\limits_{\mathbf{y}\in Y,\mu_B} & \mu_B \\ \text{s.t.} & \mu_B \geq f_2(\mathbf{y}) + \lambda^{k\top}\mathbf{h}_2(\mathbf{y}) \\ & \quad +\mu^{k\top}\mathbf{g}_2(\mathbf{y}) + L_1^k, \\ & \quad k = 1,\ldots,K, \\ & 0 \geq \mu_B\overline{\lambda}^{l\top} h_2\mathbf{y} + \overline{\mu}^{l\top} g_2(\mathbf{y}) + L_1^l, \\ & \quad l = 1,\ldots,\Lambda, \end{cases}$$

where

$$L_1^k = \min_{\mathbf{x}\in\mathbf{X}}\{f_1(\mathbf{x}) + \lambda^{k\top}\mathbf{h}_1(\mathbf{x}) + \mu^{k\top}\mathbf{g}_1(\mathbf{x})\},$$

$$\overline{L}_1^k = \min_{\mathbf{x}\in\mathbf{X}}\{f_1(\mathbf{x}) + \overline{\lambda}^{l\top}\mathbf{h}_1(\mathbf{x}) + \overline{\mu}^{l\top}\mathbf{g}_1(\mathbf{x})\}$$

are solutions of the above stated independent problems.
Let $(\hat{\mathbf{y}}, \hat{\mu}_B)$ be an optional solution. $\hat{\mu}_B$ is a lower bound, that is LBD $= \hat{\mu}_B$. If UBD $-$ LBD $\leq \epsilon$, then terminate.

3 As in GBD.

Algorithm for v1-GBD

for outer approximation algorithm), then we have

$$f_2(\mathbf{y}) = c^\top\mathbf{y},$$
$$\mathbf{h}_2(\mathbf{y}) = A\mathbf{y},$$
$$\mathbf{g}_2(\mathbf{y}) = B\mathbf{y},$$

in which case the relaxed master problem of Step 2 of v1-GBD will be a linear 0–1 programming problem with an additional scalar μ_B, which can be solved with available solvers (e. g., CPLEX, ZOOM, SCICONIC).

If the $\mathbf{y}$ variables participate separably but in a nonlinear way, then the relaxed master problem is of 0–1 nonlinear programming type.

Note that due to the strong duality theorem we do not need to solve the problems for L_1^k, $\overline{L}_1^l$ since their optimum solutions are identical to the ones of the corresponding feasible and infeasible primal problems with respect to $\mathbf{x}$ respectively.

Variant 2 of GBD: V2-GBD

This variant of GBD is based on the assumption that we can use the optimal solution x^k of the primal problem $P(y^k)$ along with the multiplier vectors for the determination of the support function $\xi(\mathbf{y};\lambda^k, \mu^k)$.

Similarly, we assume that we can use the optimal solution of the feasibility problem (if the primal is infeasible) for the determination of the support function $\xi(\mathbf{y}; \lambda^k, \mu^k)$.

The aforementioned assumption fixes the $\mathbf{x}$ vector to the optimal value obtained from its corresponding primal problem, and therefore eliminates the inner optimization problems that define the support functions. It should be noted that fixing $\mathbf{x}$ to the solution of the corresponding primal problem may not necessarily produce valid support functions in the sense that there would be no theoretical guarantee for obtaining lower bounds can be claimed in general.

The v2-GBD algorithm can be stated as follows:

1 Let an initial point $\mathbf{y}^1 \in \mathbf{Y} \cap \mathbf{V}$.
Solve the primal problem $P(\mathbf{y}^1)$ and obtain an optimal solution $\mathbf{x}^1$ and multiplier vectors λ^1, μ^1. Set the counters $k = 1$, $l = 1$, and UBD $= v(y^1)$. Select the convergence tolerance $\epsilon \geq 0$.

2 Solve the relaxed master problem:

$$\begin{cases} \min\limits_{\mathbf{y}\in\mathbf{Y}\mu_B} & \mu_B \\ \text{s.t.} & \mu_B \geq L(\mathbf{x}^k, \mathbf{y}, \lambda^k, \mu^k), \\ & \quad k = 1,\ldots,K, \\ & 0 \geq \overline{L}(\mathbf{x}^l, \mathbf{y}, \overline{\lambda}^l, \overline{\mu}^l), \\ & \quad l = 1,\ldots,\Lambda, \end{cases}$$

$$L(\mathbf{x}^k, \mathbf{y}, \lambda^k, \mu^k) = f(\mathbf{x}^k, \mathbf{y}) + \lambda^{k\top}\mathbf{h}(\mathbf{x}^k, \mathbf{y}) + \mu^{k\top}\mathbf{g}(\mathbf{x}^k, \mathbf{y}),$$
$$\overline{L}(\overline{\mathbf{x}}^l, \mathbf{y}, \overline{\lambda}^l, \overline{\mu}^k) = \overline{\lambda}^{k\top}\mathbf{h}(\mathbf{x}^l, \mathbf{y}) + \overline{\mu}^{k\top}\mathbf{g}(\mathbf{x}^l, \mathbf{y})$$

are the Lagrange functions evaluated at the optimal solution x^k of the primal problem.
Let $(\hat{\mathbf{y}}, \hat{\mu}_B)$ be an optimal solution. $\hat{\mu}_B$ is a lower bound, that is LBD $= \hat{\mu}_B$. If UBD $-$ LBD $\leq \epsilon$, then terminate.

3 As in GBD.

Algorithm for v2-GBD

Note that since $\mathbf{y} \in \mathbf{Y} = \{0-1\}$, the master problem is a 0–1 programming problem with one scalar variable μ_B. If the $\mathbf{y}$ variables participate linearly, then it is a 0–1 linear problem which can be solved with standard branch and bound algorithms. In such a case, we can introduce integer cuts of the form:

$$\sum_{i \in B} y_i - \sum_{i \in NB} y_i \leq |B| - 1,$$

where $B = \{i: y_i = 1\}$, $NB = \{i: y_i = 0\}$, $|B|$ is the cardinality of B, which eliminate the already found 0–1 combinations. If we employ such a scheme, then an alternative termination criterion is that of having infeasible relaxed master problems. This of course implies that all 0–1 combinations have been considered.

It is of considerable interest to identify the conditions which if satisfied make the assumption in v2-GBD a valid one. The assumption in a somewhat different restated form is that:

$$\begin{aligned}
&\xi(\mathbf{y}; \lambda^k, \mu^k) = \min_{\mathbf{x} \in \mathbf{X}} L(\mathbf{x}, \mathbf{y}, \lambda^k, \mu^k) \\
&\geq L(\mathbf{x}^k, \mathbf{y}, \lambda^k, \mu^k), \quad k = 1, \ldots, K, \\
&\overline{\xi}(\mathbf{y}; \overline{\lambda}^l, \mu^l) = \min_{\mathbf{x} \in \mathbf{X}} \overline{L}(\mathbf{x}, \mathbf{y}, \overline{\lambda}^l, \overline{\mu}^l) \\
&\geq \overline{L}(\mathbf{x}^l, \mathbf{y}, \overline{\lambda}^l, \overline{\mu}^l), \quad l = 1, \ldots, \Lambda,
\end{aligned}$$

that is, we assume that the Lagrange function evaluated at the solution of the corresponding primal are valid underestimators of the inner optimization problems with respect to $\mathbf{x} \in \mathbf{X}$.

Due to condition C1) the Lagrange functions $L(\mathbf{x}, \mathbf{y}, \lambda^k, \mu^k)$, $\overline{L}(\mathbf{x}, \mathbf{y}, \overline{\lambda}^l, \overline{\mu}^l)$ are convex in $\mathbf{x}$ for each fixed $\mathbf{y}$ since they are linear combinations of convex functions ix $\mathbf{x}$.

$L(\mathbf{x}, \mathbf{y}, \lambda^k, \mu^k)$, $\overline{L}(\overline{\mathbf{x}}^l, \mathbf{y}, \overline{\lambda}^l, \overline{\mu}^l)$ represent local linearizations around the points $\mathbf{x}^k$ and $\overline{\mathbf{x}}^k$ of the support functions $\xi(\mathbf{y}; \lambda^k, \mu^k)$, $\overline{\xi}(\mathbf{y}; \overline{\lambda}^l, \mu^l)$ respectively. Therefore, the aforementioned assumption is valid if the projected problem $v(\mathbf{y})$ is convex in $\mathbf{y}$. If however, the projected problem $v(\mathbf{y})$ is convex in $\mathbf{y}$. If however, the projected problem $v(\mathbf{y})$ is nonconvex, then the assumption does not hold, and the algorithm may terminate at a local (not global) solution or even at a nonstationary point.

Note that in the above analysis we did not assume that $\mathbf{Y} = \{0, 1\}^l$, and hence the argument is applicable even when the $\mathbf{y}$-variables are continuous.

It is also very interesting to examine the validity of the assumption made in v2-GBD under the conditions of separability of $\mathbf{x}$ and $\mathbf{y}$ and linearity in $\mathbf{y}$ (i. e., OA conditions). In this case we have:

$$\begin{aligned}
f(\mathbf{x}, \mathbf{y}) &= \mathbf{c}^\top \mathbf{y} + f_1(\mathbf{x}), \\
\mathbf{h}(\mathbf{x}, \mathbf{y}) &= A\mathbf{y} + \mathbf{h}_1(\mathbf{x}), \\
\mathbf{g}(\mathbf{x}, \mathbf{y}) &= B\mathbf{y} + \mathbf{g}_1(\mathbf{x}).
\end{aligned}$$

Then, the support function for feasible primal becomes

$$\begin{aligned}
&\xi(\mathbf{y}; \lambda^k, \mu^k) = \mathbf{c}^\top \mathbf{y} + \lambda^{k\top}(A\mathbf{y}) \\
&+ \mu^{k\top}(B\mathbf{y}) + \min_{\mathbf{x} \in \mathbf{X}} f_1(\mathbf{x}) + \lambda^{k\top} \mathbf{h}_1(\mathbf{x}) + \mu^{k\top} \mathbf{g}_1(\mathbf{x}),
\end{aligned}$$

which is linear in $\mathbf{y}$ and hence convex in $\mathbf{y}$. Note also that since we fix $\mathbf{x} = \mathbf{x}^k$, the $\min_{\mathbf{x} \in \mathbf{X}}$ is in fact an evaluation at $\mathbf{x}^k$. Similarly the case for $\overline{\xi}(\mathbf{y}; \overline{\lambda}^k, \mu^k)$ can be analyzed.

Therefore, the assumption in v2-GBD holds true if separability and linearity hold which covers also the case of linear 0–1 $\mathbf{y}$ variables. This way under conditions C1), C2), C3) the v2-GBD determined the global solution for separability in $\mathbf{x}$ and $\mathbf{y}$ and linearity in $\mathbf{y}$ problems.

Variant 3 of GBD: V3-GBD

This variant was proposed in [4] and denoted as *global optimum search*, GOS, and was applied to continuous as well as 0–1 set $\mathbf{Y}$. It uses the same assumption as the one in v2-GBD but in addition assumes that:

i) $\mathbf{f}(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$ are convex functions in $\mathbf{y}$ for every fixed $\mathbf{x}$; and
ii) $\mathbf{h}(\mathbf{x}, \mathbf{y})$ are linear functions in $\mathbf{y}$ for every $\mathbf{x}$.

This additional assumption was made so as to create special structure not only in the primal but also in the relaxed master problem. The type of special structure in the relaxed master problem has to do with its convexity characteristics.

The basic idea in GOS is to select the $\mathbf{x}$ and $\mathbf{y}$ variables in a such a way that the primal and the relaxed master problem of the v2-GBD satisfy the appropriate convexity requirements and hence attain their respective global solutions.

We will discuss v3-GBD first under the separability of $\mathbf{x}$ and $\mathbf{y}$ and then for the general case.

V3-GBD Under Separability

Under the separability assumption we have:

$$f(\mathbf{x},\mathbf{y}) = f_1(\mathbf{x}) + f_2(\mathbf{y}),$$
$$\mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{h}_1(\mathbf{x}) + \mathbf{h}_2(\mathbf{y}),$$
$$\mathbf{g}(\mathbf{x},\mathbf{y}) = \mathbf{g}_1(\mathbf{x}) + \mathbf{g}_2(\mathbf{y}).$$

The additional assumption that makes v3-GBD different from v2-GBD implies that

i) $\mathbf{f}_2(\mathbf{y})$, $\mathbf{g}_2(\mathbf{y})$ are convex in $\mathbf{y}$; and
ii) $\mathbf{h}_2(\mathbf{y})$ are linear in $\mathbf{y}$.

Then, the relaxed master problem will be:

$$\begin{cases} \min\limits_{\mathbf{y},\mu_B} & \mu_B \\ \text{s.t.} & \mu_B \geq f_2(\mathbf{y}) + \lambda^{k\top} h_2(\mathbf{y}) + \mu^{k\top} g_2(\mathbf{y}) \\ & \quad + \left[f_1(\mathbf{x}^k) + \lambda^{k\top} h_1(\mathbf{x}^k) + \mu^{k\top} g_1(\mathbf{x}^k) \right], \\ & \quad k = 1,\ldots,K, \\ & 0 \geq \overline{\lambda}^{l\top} h_2(\mathbf{y}) + \overline{\mu}^{l\top} g_2(\mathbf{y}) \\ & \quad + \left[\overline{\lambda}^{l\top} h_1(\overline{\mathbf{x}}^l) + \overline{\mu}^{l\top} g_1(\overline{\mathbf{x}}^l) \right], \\ & \quad l = 1,\ldots,L. \end{cases}$$

Note that the additional assumption makes the problem convex in $\mathbf{y}$ if $\mathbf{y}$ represent continuous variables. If $\mathbf{y} \in \mathbf{Y} = \{0, 1\}$, and the $\mathbf{y}$-variables participate linearly (i. e., $\mathbf{f}_2$, g_2 are linear in $\mathbf{y}$), then the relaxed master is convex. Therefore, this case represents an improvement over v3-GBD, and application of v3-GBD will result in valid support functions, which implies that the global optimum of the original problem will be obtained.

V3-GBD Without Separability

The global optimum search, GOS, aimed at exploiting and invoking special structure for nonconvex nonseparable problems

$$\begin{cases} \min & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = 0 \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq 0 \\ & \mathbf{x} \in X \subseteq \mathbf{R}^n \\ & \mathbf{y} \in Y \subseteq \mathbf{R}^q, \end{cases}$$

under the conditions C1), C2), C3) and the additional condition:

i) $\mathbf{f}(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$ are convex functions in $\mathbf{y}$ for every fixed $\mathbf{x}$;
ii) $\mathbf{h}(\mathbf{x}, \mathbf{y})$ are linear functions in $\mathbf{y}$ for every $\mathbf{x}$.

Hence both the primal and the relaxed problems attain their respective global solutions.

Note that since $\mathbf{x}$ and $\mathbf{y}$ are not separable, then the GOS cannot provide theoretically valid functions in the general case, but only if the $v(\mathbf{y})$ is convex (see the Section v2-GBD).

The global optimization approach (GOP) of [5,6] overcomes this fundamental difficulty and guarantees ϵ-global optimality for several classes of nonconvex problems.

GBD in Continuous and Discrete-Continuous Optimization

We mentioned in the Section Formulation that the original problem represents a sub-class of the problems for which the generalized Benders decomposition, GBD, can be applied. This is because we considered the $\mathbf{y} \in \mathbf{Y}$ set to consist of 0–1 variables, while [7] proposed an analysis for $\mathbf{Y}$ being a continuous, discrete or continuous-discrete set.

The main objective in this section is to present the modifications needed to carry on the analysis for continuous $\mathbf{Y}$ and discrete-continuous $\mathbf{Y}$ set.

The analysis presented for the primal problem remains the same. The analysis though for the Master problem changes only in the dual representation of the projection of the original problem (i. e., $v(\mathbf{y})$) on the $\mathbf{y}$-space. In fact, Theorem 3 is satisfied if in addition to the two conditions mentioned in C3) we have that:

iii) for each fixed $\mathbf{y}$, $v(\mathbf{y})$ is finite, $\mathbf{h}(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$ and $\mathbf{f}(\mathbf{x}, \mathbf{y})$ are continuous on $\mathbf{X}$, $\mathbf{X}$ is closed and the ε-optimal solution of the primal problem $P(\mathbf{y})$ is nonempty and bounded for some $\varepsilon \geq 0$.

Hence, Theorem 3 has as assumptions: C1) and C3), which now has i), ii) and iii). The algorithmic procedure remains the same, while the theorem for the finite convergence becomes finite ε-convergence and requires additional conditions, which are described in the following theorem:

Theorem 6 (Finite ε-convergence) *Let*
i) $\mathbf{Y}$ be a nonempty subset of $\mathbf{V}$;
ii) $\mathbf{X}$ be a nonempty convex set;
iii) f, $\mathbf{g}$ be convex on $\mathbf{X}$ for each fixed $\mathbf{y} \in \mathbf{Y}$;

iv) $\boldsymbol{h}$ *be linear on* $\boldsymbol{X}$ *for each fixed* $\boldsymbol{y} \in \boldsymbol{Y}$;
v) $f, \boldsymbol{g}, \boldsymbol{h}$ *be continuous on* $\boldsymbol{X} \times \boldsymbol{Y}$;
vi) the set of optimal multiplier vectors for the primal problem be nonempty for all $\boldsymbol{y} \in \boldsymbol{Y}$, *and uniformly bounded in some neighborhood of each such point.*

Then, for any given $\epsilon > 0$ *the GBD terminates in a finite number of iterations.*

Assumption i) (i. e., $\mathbf{Y} \subseteq \mathbf{V}$) eliminates the possibility of Step 3b, and there are many applications in which $\mathbf{Y} \subseteq \mathbf{V}$ holds (e. g., *variable factor programming*). If however, $\mathbf{Y} \not\subseteq \mathbf{V}$, then we may need to solve step 3b infinitely many successive times. In such a case, to preserve *finite* ϵ*-convergence*, we can modify the procedure so as to finitely truncate any excessively long sequence of successive executions of Step 3b and return to Step 3a with $\widehat{\mathbf{y}}$ equal to the extrapolated limit point which is assumed to belong to $\mathbf{Y} \cap \mathbf{V}$. If we do not make the assumption $\mathbf{Y} \subseteq \mathbf{V}$, then the key property to seek is that $\mathbf{V}$ has a representation in terms of a finite collection of constraints because if this is the case then Step 3b can occur at most a finite number of times. Note that if in addition to C1), we have that $\mathbf{X}$ represents bounds on the $\mathbf{x}$-variables or $\mathbf{X}$ is given by linear constraints, and $\mathbf{h}$, $\mathbf{g}$ satisfy the separability condition, then $\mathbf{V}$ can be represented in terms of a finite collection of constraints.

Assumption vi) requires that for all $\mathbf{y} \in \mathbf{Y}$ there exist optimal multiplier vectors and that these multiplier vectors do not go to infinity, that is they are uniformly bounded in some neighborhood of each such point. [7] provided the following condition to check the uniform boundedness:

If $\mathbf{X}$ is a nonempty, compact, convex set and there exists a point $\overline{\mathbf{x}} \in \mathbf{X}$ such that

$$\mathbf{h}(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = 0,$$
$$\mathbf{g}(\overline{\mathbf{x}}, \overline{\mathbf{y}}) < 0,$$

then the set of optimal multiplier vectors is uniformly bounded in some open neighborhood of $\overline{\mathbf{y}}$.

See also

- Chemical Process Planning
- Decomposition Principle of Linear Programming
- Extended Cutting Plane Algorithm
- Generalized Outer Approximation
- MINLP: Application in Facility Location-allocation
- MINLP: Applications in Blending and Pooling Problems
- MINLP: Applications in the Interaction of Design and Control
- MINLP: Branch and Bound Global Optimization Algorithm
- MINLP: Branch and Bound Methods
- MINLP: Design and Scheduling of Batch Processes
- MINLP: Generalized Cross Decomposition
- MINLP: Global Optimization with αBB
- MINLP: Heat Exchanger Network Synthesis
- MINLP: Logic-based Methods
- MINLP: Outer Approximation Algorithm
- MINLP: Reactive Distillation Column Synthesis
- Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- Mixed Integer Nonlinear Programming
- Simplicial Decomposition
- Simplicial Decomposition Algorithms
- Stochastic Linear Programming: Decomposition and Cutting Planes
- Successive Quadratic Programming: Decomposition Methods

References

1. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numer Math 4:238
2. Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Math Program 66(3):327–349
3. Floudas CA (1995) Nonlinear and mixed integer optimization: Fundamentals and applications. Oxford Univ. Press, Oxford
4. Floudas CA, Aggarwal A, Ciric AR (1989) Global optimum search for nonconvex NLP and MINLP problems. Comput Chem Eng 13(10):1117–1132
5. Floudas CA, Visweswaran V (1990) A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: I. Theory. Comput Chem Eng 14:1397–1417
6. Floudas CA, Visweswaran V (1993) A primal-relaxed dual global optimization approach. J Optim Th Appl 78(2):187–225
7. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
8. Geromel JC, Belloni MR (1986) Nonlinear programs with complicating variables: theoretical analysis and numerical experience. IEEE Trans Syst, Man Cybern SMC–16:231

Generalized Concavity in Multi-objective Optimization

ALBERTO CAMBINI, LAURA MARTEIN
Department Statistics and Applied Math., University Pisa, Pisa, Italy

MSC2000: 90C29

Article Outline

Keywords
Efficiency
Optimality Conditions
F. John Generalized Conditions
Connectedness of the Efficient Points Sets
See also
References

Keywords

Connectedness; Efficiency; Vector generalized concavity; Optimality conditions

In the context of economics and optimization, a fundamental role is nowadays recognized to generalized concavity which has been widely studied starting from the pioneering work of K. Arrow and A.C. Enthoven [1].

The study of generalized concavity of a vector valued function is not so deep as in the scalar case; nevertheless some classes with related properties have been suggested in order to obtain sufficient optimality conditions and the connectedness of the set of all efficient points.

In this order of ideas, since there are different ways in generalizing to the multi-objective case the definitions of generalized concave functions given in the scalar case, we introduce the following classes of generalized concave vector valued functions, referring to bibliography for further deepenings.

Let X be a convex subset of the n-dimensional space $\mathbf{R}^n$ and let F be a vector function from X to $\mathbf{R}^s$. Assume that $\mathbf{R}^s$ is partially ordered by the convex closed cone U with vertex at the origin $0 \in U$ and with nonempty interior (i. e. $\operatorname{int} U \neq \emptyset$). Set $U^0 = U \setminus \{0\}$.

Definition 1 The function F is said to be *U-concave* if:

$$F(x_1 + \lambda(x_2 - x_1)) \in F(x_1) + \lambda(F(x_2) - F(x_1)) + U,$$

for all $\lambda \in (0, 1)$ and all $x_1, x_2 \in S$.

Definition 2 The function F is said to be *U-quasiconcave* if:

$$x_1, x_2 \in S, \quad F(x_2) \in F(x_1) + U$$

imply

$$F(x_1 + \lambda(x_2 - x_1)) \in F(x_1) + U$$

for all $\lambda \in (0, 1)$.

Definition 3 The function F is said to be U^0-*quasiconcave* if:

$$x_1, x_2 \in S, \quad F(x_2) \in F(x_1) + U^0$$

imply

$$F(x_1 + \lambda(x_2 - x_1)) \in F(x_1) + U^0$$

for all $\lambda \in (0, 1)$.

Definition 4 The function F is said to be $\operatorname{int}U$-*quasiconcave* if:

$$x_1, x_2 \in S, \quad F(x_2) \in F(x_1) + \operatorname{int} U$$

imply

$$F(x_1 + \lambda(x_2 - x_1)) \in F(x_1) + \operatorname{int} U$$

for all $\lambda \in (0, 1)$.

In [12], D.T. Luc suggests another class of quasiconcave functions which results less general than the one given in Definition 2, but which plays an important role in establishing the connectedness of the set of all efficient points.

Definition 5 The function F is said to be *Luc U-quasiconcave* if:

$$y \in \mathbf{R}^s, \quad x_1, x_2 \in S, \quad F(x_1), F(x_2) \in y + U$$

imply

$$F(x_1 + \lambda(x_2 - x_1)) \in y + U$$

for all $\lambda \in (0, 1)$.

In the scalar case, that is, when $s = 1$ and $U = \mathbf{R}_+$, Definitions 1, 2 and 5, 3 and 4 reduce to the ordinary definitions of concavity, quasiconcavity and semistrictly quasiconcavity, respectively.

Inclusion relationships among the previous classes of functions are given in the following Theorem:

Theorem 6

i) if F is U-concave, then F is Luc U-quasiconcave;

ii) if F is Luc U-quasiconcave, then F is U-quasiconcave;

iii) if F is U-concave and U is a pointed cone, then F is intU-quasiconcave;

iv) if F is U-concave and U is a pointed cone, then F is U^0-quasiconcave.

Proof i

i) Assume that $F(x_1), F(x_2) \in y + U$; it follows $(1 - \lambda)F(x_1) \in (1 - \lambda)y + U$ and $\lambda F(x_2) \in \lambda y + U$, so that $(1 - \lambda)F(x_1) + \lambda F(x_2) \in (1 - \lambda)y + \lambda y + U = y + U$.

ii) It is sufficient to choose $y = F(x_1)$.

iii) Assume that $F(x_2) \in F(x_1) + \text{int}U$, that is, $F(x_2) - F(x_1) \in \text{int}U$. Since F is U-concave we have $F(x_1 + \lambda(x_2 - x_1)) \in F(x_1) + \lambda(F(x_2) - F(x_1)) + U$. The thesis follows taking into account that for a pointed cone the property $\text{int}U + U = \text{int}U$ holds.

iv) The proof is similar to the one given in iii). □

Remark 7 When U is the *Paretian cone* $U = \mathbf{R}^s_+$, componentwise generalized concavity implies generalized concavity. For instance:

- if any component of F is quasiconcave then F is U-quasiconcave;
- if any component of F is strongly quasiconcave then F is either int$U$-quasiconcave or U^0-quasiconcave;
- if any component of F is upper semicontinuous and semistrictly quasiconcave then F is either int$U$-quasiconcave or U^0-quasiconcave.

It can be proven that F is $\mathbf{R}^s_+$-concave (Luc $\mathbf{R}^s_+$-quasiconcave) if and only if all its components are concave (quasiconcave); such a property does not hold for the other given classes of generalized concave functions, so that the inclusion relationships stated in i) and ii) of Theorem 6 are strict.

In the particular case of a continuous bicriteria function ($s = 2$, $U = \mathbf{R}^2_+$), the class of Luc U-quasiconcave functions collapses to the class of U-quasiconcave functions [8].

Remark 8 The following examples point out that the classes of int$U$-quasiconcave and U^0-quasiconcave functions are not comparable.

Consider the function $F: \mathbf{R} \to \mathbf{R}^3$, $F(x) = (x, x^2 - x, -x^2 + x)$ and the Paretian cone $U = \mathbf{R}^3_+$. F is int$U$-quasiconcave since there do not exist $x, y \in \mathbf{R}$ such that $F(y) > F(x)$; on the other hand, F is not U^0-quasiconcave since $F(1) = (1, 0, 0) \in F(0) + \mathbf{R}^3_+ \setminus \{0\}$, but $F(1/2) \notin F(0) + \mathbf{R}^3_+ \setminus \{0\}$.

Consider now the function $F: \mathbf{R} \to \mathbf{R}^2$, $F(x) = (x, f(x))$ with $f(x) = 0$ if $x \leq 1$, $f(x) = x - 1$ if $x > 1$ and the Paretian cone $U = \mathbf{R}^2_+$. It is easy to verify that F is U^0-quasiconcave, but F is not U-quasiconcave since $F(2) = (2, 1) \in F(0) + \text{int}\mathbf{R}^2_+$, and $F(1) = (1, 0) \notin F(0) + \text{int}\mathbf{R}^2_+$.

Remark 9 In the scalar case an upper semicontinuous and semistrictly quasiconcave function is also quasiconcave; this property is lost for a vector valued function as is shown in the following example, so that the two classes are not comparable: consider the function $F: \mathbf{R} \to \mathbf{R}^2$ defined as $F(x) = (x \sin 1/x, -x \sin 1/x)$ if $x \neq 0$; $F(x) = 0$ if $x = 0$. F is continuous and U^0-quasiconcave but it is not U-quasiconcave at $x = 0$.

Remark 10 As is known, in the scalar case there exists a characterization of quasiconcave functions in the differentiable case; unfortunately such a characterization cannot be extended in the vector case (for further developing see [7]).

Consider a differentiable vector valued function F. As for the quasiconcave case, there are different ways to extend the concept of pseudoconcavity introduced by O.L. Mangasarian [14]. With the aim to state some sufficient optimality conditions, we introduce the following two classes of functions, where $J_F(x)$ denotes the Jacobian matrix of F evaluated at x.

Definition 11 F is said to be *U-weakly pseudoconcave* if:

$$x_1, x_2 \in S, \quad F(x_2) \in F(x_1) + U^0$$

imply

$$J_F(x_1)d \in U^0, \qquad d = \frac{x_2 - x_1}{\|x_2 - x_1\|}.$$

Definition 12 F is said to be *U-pseudoconcave* if:

$$x_1, x_2 \in S, \quad F(x_2) \in F(x_1) + U^0$$

imply

$$J_F(x_1)d \in \operatorname{int} U, \qquad d = \frac{x_2 - x_1}{\|x_2 - x_1\|}.$$

When $s = 1$ and $U = \mathbf{R}_+$, Definitions 11, 12 reduce to the ordinary definition of a pseudoconcave function.

Obviously, a function which is U-pseudoconcave is U-weakly quasiconcave too; a linear function is U-concave and U-weakly pseudoconcave with respect to every cone U with vertex at the origin $0 \in U$ but it is not U-pseudoconcave. As a consequence the class of U-pseudoconcave functions is properly contained in the class of U-weakly pseudoconcave functions.

Remark 13 When U is the Paretian cone $U = \mathbf{R}^s_+$, we have:

- if any component of F is pseudoconcave then F is $\mathbf{R}^s_+$-weakly pseudoconcave;
- if any component of F is strictly pseudoconcave then F is either $\mathbf{R}^s_+$-weakly pseudoconcave or $\mathbf{R}^s_+$-pseudoconcave.

Efficiency

Consider the following vector optimization problem:

$$(\mathrm{P}) \quad U - \max F(x), \quad x \in S \subseteq X,$$

where X is an open set of $\mathbf{R}^n$, $F\colon X \to \mathbf{R}^s$, and $U \in \mathbf{R}^s$ is a nontrivial cone with vertex at the origin $0 \in U$, $\operatorname{int}U \neq \emptyset$.

A point $x_0 \in S$ is said to be:

- *weakly efficient* if $F(x) \notin F(x_0) + \operatorname{int}U$, for all $x \in S$;
- *efficient* if $F(x) \notin F(x_0) + U^0$, for all $x \in S$;
- *strictly efficient* if $F(x) \notin F(x_0) + U$, for all $x \in S$, $x \neq x_0$.

If the previous conditions are verified in $I \cap S$, where I is a suitable neighborhood of x_0, then x_0 is said to be a *local weakly efficient point* a *local efficient point* and a *local strictly efficient point*, respectively.

In the scalar case ($s = 1$, $U = \mathbf{R}_+$), the definitions of a (local) weakly efficient point and an (local) efficient point reduce to the ordinary definition of a (local) maximum point, while a (local) strictly efficient point reduces to the ordinary definition of a (local) strict maximum point. Obviously (local) strictly efficiency implies (local) efficiency and (local) efficiency implies (local) weakly efficiency.

The concept of efficiency was originally introduced by V. Pareto in the early 1900s when he used the positive orthant $\mathbf{R}^s_+$ to generate the order; therefore when $U = \mathbf{R}^s_+$ efficient points are often called *Pareto points*.

As in the scalar case, vector generalized concavity plays an important role in investigating relationships between local and global optima. Following [14], the assumption of convexity of the feasible region can be weakened requiring that S is star-shaped at the point x_0.

A set $S \subset X$ is said to be *star-shaped* at $x_0 \in S$ if for every $x \in S$ it results:

$$[x, x_0] = \{tx + (1 - t)x_0 \colon\ t \in [0, 1]\} \subset S.$$

Since optimality results involve a feasible point, from now on we will consider generalized concavity at a point x_0; this means that all the given definitions hold with $x_1 = x_0$. The following theorem shows that, under suitable assumption of generalized concavity, local efficiency implies global efficiency.

Theorem 14 *Let us consider problem (P) where S is a star-shaped set at x_0.*

i) if x_0 is a local weakly efficient point and F is $\operatorname{int}U$-quasiconcave at x_0, then x_0 is a weakly efficient point for (P);

ii) if x_0 is a local efficient point and F is U^0-quasiconcave at x_0, then x_0 is an efficient point for (P);

iii) if x_0 is a strict local efficient point and F is U-quasiconcave at x_0, then x_0 is a strictly efficient point for (P);

iv) if x_0 is a local efficient point and F is U-pseudoconcave at x_0, then x_0 is an efficient point for (P).

Proof i) Assume that there exists $x^* \in S$ such that $F(x^*) \in F(x_0) + \operatorname{int}U$. Since F is $\operatorname{int}U$-quasiconcave at x_0, we have $F(x_0 + \lambda(x^* - x_0)) \in F(x_0) + \operatorname{int}U$ for all $\lambda \in (0, 1)$ and such a relation implies, choosing λ small enough, the non local weakly efficiency of x_0.

ii), iii) follow with similar arguments.

iv) Assume that there exists $x^* \in S$ such that $F(x^*) \in F(x_0) + U^0$. Since F is U-pseudoconcave at x_0, we have $J_F(x_0)d \in \operatorname{int}U$, $d = (x^* - x_0)/\|x^* - x_0\|$, that is

$$\lim_{t \to 0^+} \frac{F(x_0 + td) - F(x_0)}{t} \in \operatorname{int} U$$

and this implies the existence of a suitable $\epsilon > 0$ such that $F(x_0 + td) - F(x_0) \in \text{int}U$ for all $t \in (0, \epsilon)$.

Set $t = \lambda \|x^* - x_0\|$; we have $F(x_0+\lambda(x^*-x_0)) \in F(x_0) + \text{int}U$ for all $\lambda \in (0, \epsilon/\|x^* - x_0\|)$ and this contradicts the local efficiency of x_0. □

Corollary 15 *Let us consider problem (P) where S is locally star shaped at x_0.*

i) If U is a pointed cone and F is U-concave at x_0, then a local efficient point x_0 is an efficient point too.

ii) If F is linear, then a local efficient point x_0 is an efficient point too.

Optimality Conditions

Now we point out the role played by vector generalized concavity in stating sufficient optimality conditions. With this aim consider the necessary optimality conditions stated in the following Theorem:

Theorem 16 *Let us consider problem (P) where F is differentiable at x_0.*

i) If x_0 is a local interior efficient point for (P), then

$$\exists \alpha \in U^* \setminus \{0\} : \ \alpha^\top J_{F_{x_0}} = 0, \tag{1}$$

where U^* denotes the positive polar cone of U.

ii) If x_0 is a local efficient point for (P) then

$$J_{F_{x_0}}(v) \notin \text{int}\, U, \qquad \forall v \in T(S, x_0),\ v \neq 0. \tag{2}$$

Here, $T(S, x_0)$ is the *Bouligand tangent cone*, defined as:

$$T(S, x_0) = \left\{ v: \begin{array}{c} \exists \{\alpha_n\} \subset \mathbf{R},\ \{x_n\} \subset S, \\ \alpha_n \to \infty,\ x_n \to x_0, \\ \alpha_n (x_n - x_0) \to v \end{array} \right\}.$$

The following theorem points out the different roles played by weakly pseudoconcavity and pseudoconcavity:

Theorem 17 *Let us consider problem (P) where S is a star shaped set and F is differentiable at x_0.*

i) if (1) holds and F is U-pseudoconcave at x_0, then x_0 is an efficient point for (P);

ii) if (1) holds with $\alpha \in intU^*$ and F is U-weakly pseudoconcave at x_0, then x_0 is an efficient point for (P);

iii) if (2) holds and F is U-pseudoconcave at x_0, then x_0 is an efficient point for (P);

iv) if $J_{F_{x_0}}(v) \notin U^0$, for all $v \in T(S, x_0)$ and F is U-weakly pseudoconcave at x_0, then x_0 is an efficient point for (P).

Proof i) Assume that there exists $x^* \in S$ such that $F(x^*) \in F(x_0) + U^0$. Since F is U-pseudoconcave at x_0, we have $J_{F_{x_0}}(d) \in \text{int}U$, $d = (x^* - x_0)/\|x^* - x_0\|$, so that $\alpha^\intercal(J_{F_{x_0}}(d)) > 0$ and this contradicts (1).

ii) Assume that there exists $x^* \in S$ such that $F(x^*) \in F(x_0) + U^0$. Since F is U-weakly pseudoconcave at x_0, we have $J_{F_{x_0}}(d) \in U^0$, $d = (x^* - x_0)/\|x^* - x_0\|$, so that $\alpha^\intercal(J_{F_{x_0}}(d)) > 0$ and this contradicts (1).

iii), iv) follow immediately. □

When F is a linear vector valued function, Theorem 17ii) can be specified by means of the following theorem:

Theorem 18 *Consider problem (P) where F is linear and U is a pointed cone.*

An interior point x_0 is an efficient point for (P) if and only if there is $\alpha \in intU^*$ such that $\alpha^\intercal J_{F_{x_0}} = 0$.

F. John Generalized Conditions

Now we stress the role of vector generalized concavity in stating the sufficiency of F. John condition.

With this aim consider the vector problem (P) in the following form:

$$(P) \begin{cases} U - \max & F(x), \\ & x \in S = \{x \in X:\ G(x) \in V\}, \end{cases}$$

where $X \subset \mathbf{R}^n$ is an open set, $F: X \to \mathbf{R}^s$, $G: X \to \mathbf{R}^m$ are differentiable functions and $U \subset \mathbf{R}^s$, $V \subset \mathbf{R}^m$ are closed, pointed, convex cones with vertices at the origin and nonempty interiors.

Denote with U^*, V^* the positive polar cones of U and V, respectively, and let x_0 be a feasible point such that $G(x_0) = 0$.

The following *F. John necessary optimality conditions* hold:

Theorem 19 *If x_0 is a local efficient point for (P), then*

$$\begin{aligned} &\exists (\alpha_F, \alpha_G) \neq 0, \alpha_F \in U^*, \alpha_G \in V^* : \\ &\alpha_F^\top J_{F_{x_0}} + \alpha_G^\top J_{G_{x_0}} = 0. \end{aligned} \tag{3}$$

The following theorem points out the role of generalized concavity in stating sufficient optimality conditions:

Theorem 20 *Let us consider the vector optimization problem (P) where S is a star shaped set at x_0 and F, G are differentiable at x_0.*

i) if F is U-weakly pseudoconcave at x_0, G is V-quasiconcave at x_0, and (3) holds with $\alpha_F \in intU^*$, then x_0 is an efficient point for (P).

ii) if F is U-pseudoconcave at x_0, G is V-quasiconcave at x_0, and (3) holds with $\alpha_F \in U^* \setminus \{0\}$, then x_0 is an efficient point for (P).

Proof i) Suppose that there exists $x^* \in S$ such that $F(x^*) \in F(x_0) + U^0$. Since F is U-weakly pseudoconcave at x_0 and G is V-quasiconcave at x_0 we have, respectively, $J_{F_{x_0}}(x^* - x_0) \in U^0$, $J_{G_{x_0}}(x^* - x_0) \in V$ and thus $\alpha_F^\top J_{F_{x_0}}(x^* - x_0) > 0$, $\alpha_G^\top J_{G_{x_0}}(x^* - x_0) \geq 0$ since $\alpha_F \in \text{int}U^*$ and $\alpha_G \in V^*$. Consequently $\alpha_F^\top J_{F_{x_0}}(x^* - x_0) + \alpha_G^\top J_{G_{x_0}}(x^* - x_0) > 0$ and this contradicts (3).

ii) similar to the one given in i). □

Connectedness of the Efficient Points Sets

A vector maximization problem normally has a continuum of optimal alternatives and it may be necessary to select one or several of these which are best with respect to some additional auxiliary criterion, so that a desirable property is connectedness since it provides a possibility of continuous moving from one efficient point to any other along optimal alternatives only. Consider problem (P) where $F = (f_1, \ldots, f_s)$ is a continuous function and U is the Paretian cone; denote with $S(a)$ the upper level set associated to the point $a \in \mathbf{R}^s$, that is $S(a) = \{x \in S: F(x) \in a + U\}$. The following fundamental result was given by A.R. Warburton [16].

Theorem 21

i) if $f_1, \ldots, f_s$ are quasiconcave functions on the closed convex set S and $S(a)$ is compact for each $a \in f_1(S) \times \cdots \times f_s(S)$, then the set of all weakly Pareto points is nonempty and connected;

ii) if $f_1, \ldots, f_s$ are strongly quasiconcave functions on the closed convex set S and $S(a)$ is compact for each $a \in f_1(S) \times \cdots \times f_s(S)$, then the set of all Pareto points is nonempty and connected.

Obviously the compactness of sets $S(a)$ is verified when S is a compact set; in this last case for a bicriteria and three criteria, Theorem 21ii) holds, requiring the weaker assumption of semistrictly quasiconcavity instead of strongly quasiconcavity [9,15].

In [12], Luc extends Theorem 21i) with respect to a pointed closed convex cone requiring that F is U-continuous.

F is said to be *U-continuous* at $x \in S$ if for any neighborhood H of $F(x)$, there exists a neighborhood I of x such that $F(y) \in H - U$ for all $y \in I \cap S$.

Theorem 22 *Assume that F is a U-continuous Luc U-quasiconcave function on S and the set of all weakly efficient points of $S(a)$ is compact for each $a \in \boldsymbol{R}^s$. Then the set of all weakly efficient points is nonempty and connected.*

See also

- ► Invexity and its Applications
- ► Isotonic Regression Problems

References

1. Arrow KJ, Enthoven AC (1961) Quasi-concave programming. Econometrica 29:779–800
2. Cambini A, Martein L (1993) An approach to optimality conditions in vector and scalar optimization. In: Diewert WE et al (eds) Mathematical modelling in Economics. Springer, Berlin, pp 345–358
3. Cambini A, Martein L (1994) Generalized concavity and optimality conditions in vector and scalar optimization. In: Komlosi S, Rapcsak T, Schaible S (eds) Generalized Convexity. vol. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 337–357
4. Cambini A, Martein L (1998) Generalized concavity in multiobjective programming. In: Crouzeix JP, Martinez-Legaz JE, Volle M (eds) Generalized Convexity, Generalized Monotonicity: Recent Results. Kluwer, Dordrecht, pp 453–467
5. Cambini A, Martein L, Cambini R (1997) Some optimality conditions in multiobjective programming. In: Climaco JN (ed) Multicriteria Analysis. Springer, Berlin, pp 168–178
6. Cambini R (1996) Generalized concavity and optimality conditions in vector optimization. In: Du D-Z, Zhang X-S, Cheng K (eds) Oper. Res. Appl. World Publ. Corp., pp 172–180
7. Cambini R (1996) Some new classes of generalized concave vector-valued functions. Optim 36:11–24
8. Cambini R (1998) Generalized concavity for bicriteria functions. In: Crouzeix JP, Martinez-Legaz JE, Volle M (eds) Generalized Convexity, Generalized Monotonicity: Recent Results. Kluwer, Dordrecht, pp 439–451
9. Daniilidis A, Hadjisavvas N, Schaible S (1997) Connectedness of the efficient set for three- objective quasiconcave maximization problems. J Optim Th Appl 93(3):517–524
10. Jahn J (1986) Mathematical vector optimization in partially ordered linear spaces. P. Lang, Frankfurt am Main
11. Jahn J, Sach E (1986) Generalized quasiconvex mappings and vector optimization. SIAM J Control Optim 24(2):306–322

12. Luc DT (1987) Connectedness of the efficient point sets in quasiconcave vector maximization. J Math Anal Appl 122:346–354
13. Luc DT (1988) Theory of vector optimization. Lecture Notes Economics and Math Systems, vol 319. Springer, Berlin
14. Mangasarian OL (1969) Nonlinear programming. McGraw-Hill, New York
15. Schaible S (1983) Bicriteria quasiconcave programs. Cahiers CERO 25:93–101
16. Warburton AR (1983) Quasiconcave vector maximization: connectedness of the sets of Pareto-optimal and weak Pareto-optimal alternatives. J Optim Th Appl 40(4):537–557

Generalized Disjunctive Programming

IGNACIO E. GROSSMANN
Department of Chemical Engineering,
Carnegie Mellon University, Pittsburgh, USA

Article Outline

Keywords and Phrases
Synonyms
Introduction
Mixed-Integer Programming Reformulations
Solution Algorithms for GDP
References

Keywords and Phrases

Generalized disjunctive programming; Disjunctive programming; Convex hull; Mixed-integer programming; Outer-approximation method; Generalized benders decomposition

Synonyms

Boolean variables; Convex functions; Disjunctions; Convex hull disjunctions; Disjunctive programming; Generalized disjunctive programming; Hull relaxation; Mixed-integer programming; MILP; MINLP

Introduction

Generalized Disjunctive Programming (GDP) [13] is an extension of disjunctive programming [1,2] that provides an alternate way of modeling mixed-integer linear programming (MILP) and mixed-integer nonlinear programming (MINLP) problems. The general formulation of a (GDP) is as follows:

$$\begin{aligned} &\min Z = \sum_{k\in K} c_k + f(x) \\ &s.t. \quad r(x) \le 0 \\ &\bigvee_{j\in J_k} \begin{bmatrix} Y_{jk} \\ g_{jk}(x) \le 0 \\ c_k = \gamma_{jk} \end{bmatrix} k \in K \qquad \text{(GDP)} \\ &\Omega(Y) = True \\ &x \in R^n, \quad c \in R^m, \quad Y \in \{true, false\}^m \end{aligned}$$

where Y_{jk} are the Boolean variables that decide whether a given term j in a disjunction $k \in K$ is true or false, and x are the continuous variables. The objective function involves the term $f(x)$ for the continuous variables and the charges c_k that depend on the discrete choices in each disjunction $k \in K$. The constraints $r(x) \le 0$ must hold regardless of the discrete choices, and $g_{jk}(x) \le 0$ are conditional constraints that must hold when Y_{jk} is true in the j-th term of the k-th disjunction. The cost variables c_k correspond to the fixed charges, and their value equals to γ_{jk} if the Boolean variable Y_{jk} is true. $\Omega(Y) = True$ are logical relations for the Boolean variables expressed as propositional logic. An important particular case is the one where the functions $f(x)$, $r(x)$ and $g_{jk}(x)$ are all linear. For the nonlinear case it is assumed for the derivation of basic methods that the functions are convex, although in practical applications these often correspond to nonconvex functions.

Mixed-Integer Programming Reformulations

Problem (GDP) can be reformulated as the following "big-M"MINLP problem,

$$\begin{aligned} &\min Z = \sum_{k\in K}\sum_{j\in J_k} \gamma_{jk} y_{jk} + f(x) \\ &s.t. \quad r(x) \le 0 \\ &g_{jk}(x) \le M_{jk}(1 - y_{jk}), \quad j \in J_k, \quad k \in K \qquad \text{(BM)} \\ &\sum_{j\in J_k} y_{jk} = 1, \quad k \in K \\ &Ay \le a \\ &0 \le x \le x^U, \quad y_{jk} \in \{0,1\}, \quad j \in J_k, \quad k \in K \end{aligned}$$

where the Boolean variables are replaced by binary variables y_{jk}, the disjunctions are replaced by "Big-M"

constraints which involve a parameter M_{jk} and binary variables y_{jk}. The propositional logic statements $\Omega(Y)$ = True are replaced by the linear constraints $Ay \leq a$ as described by Williams [19]. Here we assume that x is a non-negative variable with finite upper bound x^U. An important issue in model (BM) is how to specify a valid value for the Big-M parameter M_{jk}. If the value is too small, then feasible points may be cut off. If M_{jk} is too large, then the continuous relaxation might be too loose yielding weak lower bounds. Therefore, finding the smallest valid value for M_{jk} is the desired selection. For linear constraints one can use the upper and lower bound of the variable x to calculate the maximum value of each constraint, which then can be used to calculate a valid value of M_{jk}. For nonlinear constraints one can in principle maximize each constraint over the feasible region, which is a non-trivial calculation. It is also important to note that if the binary variables y_{jk} are specified as continuous, $0 \leq y_{jk} \leq 1$, and the functions $f(x)$, $r(x)$ and $g_{jk}(x)$ are assumed to be convex, the relaxation of problem (BM) reduces to a convex NLP problem, that provides a valid lower bound to the solution of problem (GDP).

The MINLP hull reformulation of problem (GDP) is based on the following proposition by Lee and Grossmann [11]:

Proposition 1 *The convex hull of each disjunction $k \in K$ in problem (GDP),*

$$\bigvee_{j \in J_k} \begin{bmatrix} Y_{jk} \\ g_{jk}(x) \leq 0 \\ c = \gamma_{jk} \end{bmatrix} \qquad (\mathrm{D}_k)$$
$$0 \leq x \leq x^U, \quad c \geq 0$$

where $g_{jk}(x) \leq 0$ are convex inequalities, is a convex set and is given by,

$$x = \sum_{j \in J_k} v^{jk}, \; c = \sum_{j \in J} y_{jk} \gamma_{jk}$$
$$0 \leq v^{jk} \leq y_{jk} x^U_{jk}, \; j \in J_k$$
$$\sum_{j \in J_k} y_{jk} = 1, 0 \leq y_{jk} \leq 1, \; j \in J_k \qquad (\mathrm{CH}_k)$$
$$y_{jk} g_{jk}(v^{jk}/y_{jk}) \leq 0, \; j \in J_k$$
$$x, c, v^{jk} \geq 0, \; j \in J_k$$

The proof is based on an extension of the work by Stubbs and Mehrotra [16]. In (CH_k), v^{jk} are disaggregated variables that are assigned to each term of the disjunction $\{k \in K\}$, and y_{jk} can be regarded as the weight factors that determine the feasibility of the disjunctive term. Note that when y_{jk} is 1, then the j'th term in the k'th disjunction is enforced and the other terms are ignored. The constraints $y_{jk} g_{jk}(v^{jk}/y_{jk})$ are convex if $g_{jk}(x)$ is convex as discussed in Hiriart-Urruty and Lemaréchal [8]. Formal proofs can be found in [15] and [16]. Note that the convex hull (CH_k) reduces to the result by Balas [2] if the constraints are linear. Based on the convex hull relaxation (CH_k), Lee and Grossmann [11] proposed the following MINLP hull reformulation of (GDP):

$$\min Z = \sum_{k \in K} \sum_{j \in J_k} \gamma_{jk} y_{jk} + f(x)$$
$$s.t. \quad r(x) \leq 0$$
$$x = \sum_{j \in J_k} v^{jk}, \; \sum_{j \in J_k} y_{jk} = 1, \qquad k \in K \qquad (\mathrm{HR})$$
$$0 \leq v^{jk} \leq y_{jk} x^U_{jk}, \; j \in J_k, \quad k \in K$$
$$y_{jk} g_{jk}(v^{jk}/y_{jk}) \leq 0, j \in J_k, \quad k \in K$$
$$Ay \leq a$$
$$0 \leq x, v^{jk} \leq x^U, \; y_{jk} = 0.1, j \in J_k, \quad k \in K.$$

The relaxation of problem (HR) where $0 \leq y_{jk} \leq 1$, reduces to a convex NLP problem that yields a valid lower bound to the optimal solution of problem (GDP). Also, this relaxation, which can also be regarded as a generalization of the disjunctive problem studied by Ceria and Soares [4], can be interepreted as one where the convex hulls of each of the disjunctions are interesected.

The following proposition holds for problems (PR) and (BM) as proved by Grossmann and Lee [7].

Proposition 2 *Let Z^R_{HR} be the optimal value of problem (HR) where the binary variables are relaxed as $0 \leq y_{jk} \leq 1$, and let Z^R_{BM} be the optimal value of problem (BM) where the binary variables are relaxed as $0 \leq y_{jk} \leq 1$. Then, $Z^R_{BM} \leq Z^R_{HR}$.*

Hence, problem (HR) has the useful property that the lower bound of its relaxation is greater than or equal to the lower bound predicted from the relaxation of problem (BM). In some problems this translates into a significantly tighter formulations [13,14]). The trade-off, however, is that in the reformulation (HR) the number

of constraints and variables is larger than the one in the reformulation (BM).

It is also important to point out that for the computer implementation of the constraint $y_{jk} g_{jk}(\nu_{jk}/y_{jk}) \leq 0$ in problem (HR), an approximation is required for the nonlinear functions, $g_{jk}(x)$ in order to avoid the division by zero when $y_{jk} = 0$. Furman et al. [5] have proposed the following approximation, which has the interesting feature that it is exact for $y_{jk} = 0$ and $y_{jk} = 1$,

$$((1-\varepsilon)y_{jk} + \varepsilon)(g_{jk}(\nu_{jk}/((1-\varepsilon)y_{jk} + \varepsilon))) - \varepsilon g_{jk}(0)(1 - y_{jk}) \leq 0\,.$$

Furthermore, it can be shown that this inequality is convex for any value of ε. Note also that this expression reduces to the original one as $\varepsilon \to 0$.

Solution Algorithms for GDP

The most direct way of solving problem (GDP) is by reformulating it as an MINLP (or MILP for the linear case). In both cases the big-M and hull reformulation are the two extreme choices. The latter generally yields tighter relaxations, but involves solving a larger problem. For the linear case LP-based branch and cut methods can be used [10], including special cutting plane techniques [14]. For the nonlinear case, MINLP methods such as branch and bound, outer-approximation, Generalized Benders, extended cutting plane or hybrid methods can be used [6].

Logic-based method for solving linear problems (GDP) include the branch and bound method by Beaumont [3], which branches on the constraints of the disjunctions. Raman and Grossmann [13] developed a branch and bound method which solves GDP problem in hybrid form, by exploiting the tight relaxation of the disjunctions and the tightness of the well-behaved mixed-integer constraints. For the nonlinear case a disjunctive branch and bound method based on the hull relaxation has been proposed by Lee and Grossmann [11] that is coupled with logic inference techniques [9]. Also, for the special case of two-term disjunctions in (GDP), which typically arise in process network problems, Türkay and Grossmann [17] have proposed outer-approximation and Generalized Benders Decomposition algorithms. Some of these algorithms have been implemented in LOGMIP, a computer code based on GAMS [18]. Finally, for the nonconvex case a disjunctive branch and bound method coupled with a spatial branch and bound search has been reported in [12].

References

1. Balas E (1979) Disjunctive programming. Ann Discret Math 5:3–51
2. Balas E (1985) Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM J Alg Disc Meth 6:466–486
3. Beaumont N (1991) An algorithm for disjunctive programs. Eur J Oper Res 48:362–371
4. Ceria S, Soares J (1999) Convex programming for disjunctive optimization. Math Program 86(3):595–614
5. Furman K, Sawaya NW, Grossmann IE (2007) A robust MINLP reformulation for the implementation of nonlinear disjunctive programming. manuscript under preparation
6. Grossmann IE (2002) Review of nonlinear mixed-integer and disjunctive programming techniques. Optim Eng 3:227–252
7. Grossmann IE, Lee S (2003) Generalized disjunctive programming: nonlinear convex hull relaxation and algorithms. Comput Optim Appl 26:83–100
8. Hiriart-Urruty J, Lemaréchal C (1993) Convex analysis and minimization algorithms. Springer, Berlin, New York
9. Hooker JN (1999) Logic-based methods for optimization. Wiley, New York
10. Johnson EL, Nemhauser GL, Savelsbergh MWP (2000) Progress in linear programming based branch-and-bound algorithms: an exposition. INFORMS J Comput 12:2–23
11. Lee S, Grossmann IE (2000) New algorithms for nonlinear generalized disjunctive programming. Comput Chem Eng 24:2125–2141
12. Lee S, Grossmann IE (2001) A global optimization algorithm for nonconvex generalized disjunctive programming and applications to process systems. Comput Chem Eng 25:1675–1697
13. Raman R, Grossmann IE (1994) Modelling and computational techniques for logic based integer programming. Comput Chem Eng 18(7):563–578
14. Sawaya NW, Grossmann IE (2005) A cutting plane method for solving linear generalized disjunctive programming problems. Comput Chem Eng 29:1891–1913
15. Sawaya NW (2006) Reformulations, relaxations and cutting planes for generalized disjunctive programming, Ph.D. thesis. Carnegie Mellon University, Pittsburgh
16. Stubbs R, Mehrotra S (1999) A branch-and-cut method for 0-1 mixed convex programming. Math Program 86(3): 515–532
17. Türkay M, Grossmann IE (1996) Logic-based MINLP algorithms for the optimal synthesis of process networks. Comput Chem Eng 20(8):959–978

18. Vecchietti A, Grossmann IE (1999) LOGMIP: A disjunctive 0-1 nonlinear optimizer for process systems models. Comput Chem Eng 23:555–565
19. Williams HP (1985) Mathematical building in mathematical programming. Wiley, Chichester

Generalized Eigenvalue Proximal Support Vector Machine Problem

MARIO R. GUARRACINO, SALVATORE CUCINIELLO, DAVIDE FEMINIANO
High Performance Computing and Networking Institute, Italian Research Council, Napoli, Italy

MSC2000: 68Q32, 68T10

Article Outline

Problem
Kernel Formulation
Algorithm
Another Algorithm
References

Problem

Consider two matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{k\times m}$, each row being a point in one of two classes in the feature space. The *generalized eigenvalue proximal support vector machine* (GEPSVM) consists in finding two hyperplanes each one being closer to one set of points and farther from another set of points. Let $x'w - \gamma = 0$ be a hyperplane in $\mathbb{R}^m$. In order to satisfy the previous condition for all points in A, the hyperplane can be obtained by solving the following optimization problem:

$$\min_{w,\gamma\neq 0} \frac{\|Aw - e\gamma\|^2}{\|Bw - e\gamma\|^2} . \tag{1}$$

The hyperplane for B can be obtained by minimizing the inverse of the objective function in (1). Now, let

$$\begin{aligned} G &= [A \quad -e]'[A \quad -e], \\ H &= [B \quad -e]'[B \quad -e] \end{aligned} \tag{2}$$

and

$$z = [w' \quad \gamma]' . \tag{3}$$

Then (1) becomes

$$\min_{z\in\mathbb{R}^m} \frac{z'Gz}{z'Hz} . \tag{4}$$

The expression in (4) is the Raleigh quotient of the generalized eigenvalue problem $Gz = \lambda Hz$. When H is positive definite, the stationary points are obtained at and only at the eigenvectors of (4), where the value of the objective function is given by the eigenvalues. The Raleigh quotient is bounded, and it ranges over the interval determined by minimum and maximum eigenvalues [4]. H is positive definite under the assumption that the columns of $[B \quad -e]$ are linearly independent. The reciprocal of the objective function in (4) has the same eigenvectors and reciprocal eigenvalues. Let $z_{\min} = [w_1' \quad \gamma_1]'$ and $z_{\max} = [w_2' \quad \gamma_2]'$ be the eigenvectors related to the eigenvalues of the smallest and largest modulo, respectively. Then $x'w_1 - \gamma_1 = 0$ is the closest hyperplane to the set of points in A and the furthest from those in B and $x'w_2 - \gamma_2 = 0$ is the closest hyperplane to the set of points in B and the furthest from those in A. GEPSVM finds application in many supervised learning problems [3]. For example, a bank prefers to classify customer loan requests as "good" or "bad" depending on their ability to pay back the loan. The Internal Revenue Service tries to discover tax evaders starting from the characteristics of known evaders. As another example, a built-in system in a car could detect if a walking pedestrian is going to cross the street. More applications can be found in biology and medicine. The tissues that are prone to cancer can be detected with high accuracy, or new DNA sequences or proteins can be tracked down to their origins. Given its amino acid sequence, finding out how a protein folds provides important information about its expression level. An unlabeled point x is associated to the class y_i related to the closest hyperplane P_i. Therefore, a point x is classified using its distance for the corresponding hyperplane:

$$y_i = \operatorname{argmin}_{i=1,2}\{\operatorname{dist}(x, P_i)\} , \tag{5}$$

where

$$\operatorname{dist}(x, P_i) = \frac{|x'w_i - \gamma_i|}{\|w_i\|} . \tag{6}$$

Kernel Formulation

To obtain greater separability between classes, nonlinear embedding of data to a higher-dimensional space is required. This nonlinear mapping can be done implicitly by kernel functions, which represent the inner product of the elements in a nonlinear space. Kernel functions can be described as follows:

$$K(x_i, x_j) = \langle \phi(x_i) - \phi(x_j), \phi(x_i) - \phi(x_j) \rangle \,, \tag{7}$$

where $\phi(x)$ is the embedding function.

Using kernels, we can express the problem in terms of inner products between elements, and therefore the computationally expensive calculation of the feature, in the embedded space, is avoided. Some commonly used kernel functions are

$$\begin{aligned} \text{Linear} \quad & K(x_i, x_j) = x_i' \cdot x_j \\ \text{Polynomial} \quad & K(x_i, x_j) = (x_i' \cdot x_j + 1)^d \\ \text{Gaussian} \quad & K(x_i, x_j) = \exp\left(-\frac{\| x_i - x_j \|^2}{\sigma}\right) . \end{aligned}$$

Using the kernel function, each element of the kernel matrix is

$$K(A, B)_{i,j} = K(A^i, B^j) \,. \tag{8}$$

Let

$$C = \begin{bmatrix} A \\ B \end{bmatrix} .$$

Then problem (1) becomes

$$\min_{u,\gamma \neq 0} \frac{\|K(A, C)u - e\gamma\|^2}{\|K(B, C)u - e\gamma\|^2} \,. \tag{9}$$

A point x is classified using its distance for the corresponding hyperplane in the feature space:

$$y_i = \text{argmin}_{i=1,2}\{\text{dist}(x, P_i)\} \,, \tag{10}$$

where

$$\text{dist}(x, P_i) = \frac{|K(x, C)u_i - \gamma_i|}{\|u_i\|} \,. \tag{11}$$

The associated eigenvalue problem has matrices of order $n + k + 1$ and rank at most m. This means a regularization technique is needed since the problem can be singular.

Algorithm

Let G and H be as defined in (2). Note that even if A and B are full rank, matrices G and H are always rank-deficient. The reason is that G and H are matrices of order $m + 1$, and their rank can be at most m. The added complexity due to the singularity of the matrices means that special care must be given to the solution of the generalized eigenvalue problem. Indeed, if the null spaces of G and H have a nontrivial intersection, i. e., $\text{Ker}(A) \bigcap \text{Ker}(B) \neq 0$, then the problem is ill posed and a regularization technique is needed to solve the eigenvalue problem. Mangasarian et al. [2] proposes to use Tikhonov regularization applied to a twofold problem:

$$\min_{w,\gamma \neq 0} \frac{\|Aw - e\gamma\|^2 + \delta\|z\|^2}{\|Bw - e\gamma\|^2} \tag{12}$$

and

$$\min_{w,\gamma \neq 0} \frac{\|Bw - e\gamma\|^2 + \delta\|z\|^2}{\|Aw - e\gamma\|^2} \,, \tag{13}$$

where δ is the regularization parameter and the new problems are still convex. The minimum eigenvalues-eigenvectors of these problems are approximations of the minimum and maximum eigenvalues-eigenvectors of (4). The solutions (w_i, γ_i), $i = 1, 2$ to (12) and (13) represent the two hyperplanes approximating the two classes of training points. The same regularization technique can be applied to the nonlinear formulation.

Another Algorithm

It is possible to solve the problem without regularization. In practice, if $\beta G - \alpha H$ is nonsingular for every α and β, it is possible to transform the problem into another problem that is nonsingular and that has the same eigenvectors of the initial one. We start with the following theorem whose proof can be found in [5], p. 288.

Theorem 1 *Consider the generalized eigenvalue problem $Gx = \lambda Hx$ and the transformed $G^*x = \lambda H^*x$ defined by*

$$G^* = \tau_1 G - \delta_1 H, \quad H^* = \tau_2 H - \delta_2 G \tag{14}$$

for each choice of scalars τ_1, τ_2, δ_1, and δ_2 such that the 2×2 matrix

$$\Omega = \begin{pmatrix} \tau_2 & \delta_1 \\ \delta_2 & \tau_1 \end{pmatrix} \tag{15}$$

is nonsingular. Then the problem $G^*x = \lambda H^*x$ has the same eigenvectors of the problem $Gx = \lambda Hx$. An associated eigenvalue λ^* of the transformed problem is related to an eigenvalue λ of the original problem by

$$\lambda = \frac{\tau_2\lambda^* + \delta_1}{\tau_1 + \delta_2\lambda^*} .$$

In the linear case, Theorem 1 can be applied. By setting $\tau_1 = \tau_2 = 1$ and $\hat{\delta_1} = -\delta_1$, $\hat{\delta_2} = -\delta_2$, the regularized problem becomes

$$\min_{w,\gamma\neq 0} \frac{\|Aw - e\gamma\|^2 + \hat{\delta_1}\|Bw - e\gamma\|^2}{\|Bw - e\gamma\|^2 + \hat{\delta_2}\|Aw - e\gamma\|^2} . \tag{16}$$

If $\hat{\delta_1}$and $\hat{\delta_2}$ are nonnegative, Ω is nondegenerate. The spectrum is now shifted and inverted so that the minimum eigenvalue of the original problem becomes the maximum of the regularized one, and the maximum becomes the minimum eigenvalue. Choosing the eigenvectors related to the new minimum and maximum eigenvalue, we obtain the same solution of the original problem.

This regularization works for the linear case if we suppose that in each class of the training set there is a number of linearly independent rows that is at least equal to the number of the features. This is often the case and, if the number of points in the training set is much greater than the number of features, Ker(G) and Ker(H) have both dimension 1. In this case, the probability of a nontrivial intersection is zero.

In the nonlinear case the situation is different. Guarracino et al. [1] propose to generate the two proximal surfaces

$$K(x, C)u_1 - \gamma_1 = 0, \quad K(x, C)u_2 - \gamma_2 = 0 \tag{17}$$

by solving the following problem

$$\min_{u,\gamma\neq 0} \frac{\|K(A, C)u - e\gamma\|^2 + \delta\|\tilde{K}_B u - e\gamma\|^2}{\|K(B, C)u - e\gamma\|^2 + \delta\|\tilde{K}_A u - e\gamma\|^2} , \tag{18}$$

where $\tilde{K}_A$ and $\tilde{K}_B$ are diagonal matrices with the diagonal entries from the matrices $K(A,C)$ and $K(B,C)$. The perturbation theory of eigenvalue problems [6] provides an estimation of the distance between the original and the regularized eigenvectors. If we call z an eigenvector of the initial problem and $z(\delta)$ the corresponding one in the regularized problem, then $|z - z(\delta)| = \mathcal{O}(\delta)$, which means their closeness is in the order of δ.

References

1. Guarracino MR, Cifarelli C, Seref O, Pardalos PM (2007) A classification algorithm based on generalized eigenvalue problems. Optim Methods Softw 22(1):73–81
2. Mangasarian OL, Wild EW (2004) Multisurface Proximal Support Vector Classification via Generalized Eigenvalues. Data Mining Institute
3. Mitchell T (1997) Machine Learning, McGraw-Hill Education (ISE Editions) http://www.amazon.co.uk/exec/obidos/ASIN/0071154671/citeulike-21
4. Parlett BN (1998) The Symmetric Eigenvalue Problem. SIAM, Philadelphia, pp 343–345
5. Saad Y (1992) Numerical Methods for Large Eigenvalue Problems. Halsted, New York
6. Wilkinson J (1965) The Algebraic Eigenvalue Problem. Clarendon, Oxford

Generalized Geometric Programming: Mixed Continuous and Discrete Free Variables

Han-Lin Li[1], Jung-Fa Tsai[2]

[1] Institute of Information Management, National Chiao Tung University, Hsinchu, Taiwan

[2] Department of Business Management, National Taipei University of Technology, Taipei, Taiwan

MSC2000: 49M37, 90C11, 90C30

Article Outline

Keywords and Phrases
Introduction
Formulation
Methods
  Case Studies
Conclusions
References

Keywords and Phrases

Global optimization; Generalized geometric programming; Free variables; Convexification

Introduction

Generalized geometric programming (GGP) problems with continuous and discrete variables occur quite frequently in various fields such as civil and material

engineering design, chemical engineering, location-allocation, inventory control, production planning, and scheduling etc. These applications are extensively surveyed in Floudas and Pardalos [11] and Floudas [9]. Biegler and Grossmann [3] provided a retrospective on optimization techniques that have been applied in process systems engineering. They indicated that design and synthesis problems have been dominated by nonlinear programming (NLP) and mixed-integer nonlinear programming (MINLP) models. Although MINLP programs appear in many chemical engineering problems, they are often nonconvex and no direct optimization method is available to guarantee global optimality [21]. With the increasing reliance on mathematical programming based approaches in chemical engineering problems, the need for finding global optimum is paramount.

The developed methods for GGP problems with continuous and discrete variables can be divided into two approaches.

(i) Stochastic methods: The stochastic methods involve random elements in their search and rely on a statistical argument to prove their convergence. For instance, Salcedo et al. [23] proposed an improved random search algorithm for solving nonlinear optimization problems. Cardoso et al. [5] solved nonconvex nonlinear integer programming problems with simulated annealing. Yiu et al. [30] developed a hybrid descent approach based on a simulated annealing algorithm and a gradient-based method to solve multidimensional nonconvex continuous optimization problems. Hussain and Al-Sultan [15] proposed a hybrid algorithm for nonconvex function minimization by utilizing the genetic technique to generate search directions. These stochastic methods mentioned above can not guarantee to find the global optimum. Therefore, the quality of the solution is not ensured. Moreover, the probability of finding the global solution decreases when the problem size increases.

(ii) Deterministic methods: Mathematical methods that generate convex underestimators for twice differentiable constrained nonconvex optimization problems are of primary importance in deterministic global optimization [9]. The α BB global optimization algorithm [1,2,9] is a power approach for constructing such convex underestimators for nonconvex functions [10]. In a general survey of optimization techniques ([3,13,14]), many deterministic methods for convex MINLP problems have been reviewed. The methods include Branch and Bound (BB) ([17,24]), Generalized Benders Decomposition (GBD) [12], Outer-Approximation (OA) ([6,7,22]), Extended Cutting Plane Method (ECP) [28], and Generalized Disjunctive Programming (GDP) [16]. One possible approach to circumvent the nonconvex objective function or the nonconvex constraints in MINLP models is transformation. Floudas ([8,9]), Floudas and Pardalos [11] and Maranas and Floudas [20] proposed exponential transformation methods to treat GGP problems with continuous and discrete variables. The core concept of their methods is to convert the problem into a new problem where both the constraints and the objective are decomposed into the difference of two convex functions. By utilizing exponential variable transformations, each signomial term $z = x_1^{\alpha} x_2^{\beta}$, where x_1 and x_2 are positive, can be transferred into an exponential term $z' = e^{\alpha \ln x_1 + \beta \ln x_2}$. However, the exponential transformation technique can only be applied to strictly positive variables and is thus unable to deal with nonconvex GGP problems with continuous and discrete free variables.

Although positive variables are adopted frequently to represent engineering and scientific systems, it is also common to introduce free variables to model the system behavior, such as stresses, temperatures, electrical currents, velocities and accelerations, etc. In general, the values accepted by the machines are under a discrete space. For instance, a controller can only increase temperature from a fixed initial point to a set of fixed points at a fixed interval. Consequently, deriving a global optimum for the GGP problem with continuous and discrete free variables is essential for real applications. Li and Tsai [18] proposed a technique for treating free continuous variables in GGP problems. Pörn et al. [21] introduced different convexification strategies for MINLP problems with both polynomial and negative binomial terms in the constraints. They suggested a simple translation, $x + \tau = e^x$, to treat a free variable x. However, inserting the transformed result into the original signomial term will bring additional signomial terms and therefore increasing the computation bur-

den. This study proposes a method for solving a GGP problem with continuous and discrete free variables to obtain a global optimal solution. The GGP problem is first transformed into another one containing only positive variables. Then the transformed problem is reformulated as a convex mixed-integer program. A global optimum of the GGP problem with continuous and discrete free variables can finally be found within the tolerable error. Furthermore, this study develops several convexification strategies for signomial terms so that the efficiency of the optimization approach can be enhanced. The right choice of transformation for convexification of nonconvex signomial terms might significantly decrease the solution time [4]. By employing the proposed rules, certain classes of signomial terms can be determined as convex terms and do not require any transformation. Moreover, some nonconvex signomial terms with specific features can be transformed into convex terms in accordance with the proposed rules by replacing some variables, thereby making the resulting problem a computationally efficient model.

Formulation

The mathematical formulation of a GGP problem with continuous and discrete free variables is expressed as follows:

GGP:

$$\begin{aligned}
&\text{Minimize} && f(X,Y)\\
&\text{subject to} && g_t(X,Y)\le 0 \quad t=1,\dots,T,\\
&&& X=(x_1,\dots,x_p,x_{p+1}\dots,x_n),\\
&&& \underline{x}_i\le x_i\le \overline{x}_i,\\
&&& Y=(y_1,\dots,y_q,y_{q+1}\dots,y_m),\\
&&& \underline{y}_j\le y_j\le \overline{y}_j,
\end{aligned}$$

where $x_i \in \Re^+$ for $1 \le i \le p$, x_i are bounded free variables for $p+1 \le i \le n$, y_j are positive integer/discrete variables for $1 \le j \le q$, y_j are bounded free variables for $q+1 \le j \le m$, $f(X,Y)$ and $g_t(X,Y)$ are mixed-integer signomial functions, $\underline{x}_i$ and $\overline{x}_i$ are lower and upper bounds of the continuous variable x_i, and $\underline{y}_j$ and $\overline{y}_j$ are lower and upper bounds of the integer/ discrete variable y_j, respectively.

Methods

Treating Free Variables. Li and Tsai [18] proposed a technique for treating free continuous variables in GGP problems. By integrating Li and Tsai method with the approach of dealing with free discrete variables described below, a GGP problem with continuous and discrete free variables can be equivalently transform into a mixed-integer GGP program with positive variables. The following illustrates how to convert free discrete variables into non-positive discrete variables.

$$\text{Let: } y_j = y_j^+ - y_j^-,\ y_j^+, y_j^- \ge 0, \quad \text{for } j = q+1,\cdots,m\,.$$

And a nonlinear term $y_j^{\beta_j}$ is expressed as

$$y_j^{\beta_j} = (y_j^+)^{\beta_j} + (-1)^{\beta_j}(y_j^-)^{\beta_j}, \quad \beta_j \in \text{integer, for } j = q+1,\dots,m.$$

If $y_j^+ > 0$ and $y_j^- = 0$, then y_j is positive. Otherwise, if $y_j^- > 0$ and $y_j^+ = 0$, then y_j is negative. To prohibit from yielding positive values for y_j^+ and y_j^- simultaneously, we have the following remark.

Remark 1 A free discrete variable y_j can be expressed as $y_j = y_j^+ - y_j^-$, $y_j^+, y_j^- \ge 0$, and y_j^+ and y_j^- will not be positive concurrently by the following inequalities.

$$(i)\ -y_j^- \le y_j \le M\theta_j - y_j^-,$$
$$(ii)\ M(\theta_j - 1) + y_j^+ \le y_j \le y_j^+.$$

M is a sufficiently large positive number and $\theta_j \in \{0,1\}$.

By means of changing variables, the GGP problem with free variables can be equivalently solved with another one having non-negative variables. The next is to deal with discrete variables containing zero, consider the following propositions:

Proposition 1 *[21] For positive discrete variables $y_j \in \{d_{j1}, d_{j2}, \cdots, d_{jm_j}\}$ where $d_{j,i+1} > d_{ji} > 0$ for $i = 1,2,\cdots,m_j-1$, a product term $y_1^{\alpha_1} y_2^{\alpha_2} \cdots y_m^{\alpha_m}$ where $\alpha_1,\alpha_2,\cdots,\alpha_m$ are real constants can be transformed into a function $e^{\alpha_1 z_1 + \cdots + \alpha_m z_m}$ where $z_j = \ln d_{j1} + \sum_{i=1}^{m_j-1} u_{ji}(\ln d_{j,i+1} - \ln d_{j1})$, $\sum_{i=1}^{m_j-1} u_{ji} \le 1$ for $u_{ji} \in \{0,1\}$.*

Proof Let $y_j = e^{z_j}$ and $z_j = \ln y_j$, expressing y_j as $y_j = d_{j1} + \sum_{i=1}^{m_j-1} u_{ji}(d_{j,i+1} - d_{j1})$, $\sum_{i=1}^{m_j-1} u_{ji} \le 1$, where $u_{ji} \in \{0,1\}$.

We then have $y_1^{\alpha_1} y_2^{\alpha_2} \cdots y_m^{\alpha_m} = e^{\alpha_1 z_1 + \cdots + \alpha_m z_m}$ and $z_j = \ln d_{j1} + \sum_{i=1}^{m_j - 1} u_{ji}(\ln d_{j,i+1} - \ln d_{j1})$, $\sum_{i=1}^{m_j - 1} u_{ji} \le 1$, for $u_{ji} \in \{0, 1\}$. □

Because some variables y_j in Proposition 1 may have zero value, Proposition 1 needs to be modified as the following proposition:

Proposition 2 *For positive discrete variables* $y_j \in \{d_{j1}, d_{j2}, \cdots, d_{jm_j}\}$ *where* $d_{j,i+1} > d_{ji} > 0$ *for* $i = 1, 2, \cdots, m_j - 1$, $1 \le j \le q$, *and non-negative discrete variables* $y_j \in \{0, d_{j1}, d_{j2}, \cdots, d_{jm_j}\}$ *where* $d_{j,i+1} > d_{ji} > 0$ *for* $i = 1, 2, \cdots, m_j - 1$, $q + 1 \le j \le m$, *a product term* $s = y_1^{\alpha_1} y_2^{\alpha_2} \cdots y_q^{\alpha_q} y_{q+1}^{\alpha_{q+1}} \cdots y_m^{\alpha_m}$ *can be expressed as*

$$(i)\ 0 \le s \le \bar{s} \left(\sum_{i=1}^{m_j} u_{ji} \right), \text{ for } q + 1 \le j \le m,$$

$$(ii)\ \bar{s} \left(\sum_{j=q+1}^{m} \sum_{i=1}^{m_j} u_{ji} - (m - q) \right) + e^{\alpha_1 z_1 + \cdots + \alpha_m z_m} \le s \le \bar{s} \left((m - q) - \sum_{j=q+1}^{m} \sum_{i=1}^{m_j} u_{ji} \right) + L(e^{\alpha_1 z_1 + \cdots + \alpha_m z_m}),$$

where $y_j = d_{j1} + \sum_{i=1}^{m_j - 1} u_{ji}(d_{j,i+1} - d_{j1})$, $z_j = \ln d_{j1} + \sum_{i=1}^{m_j - 1} u_{ji}(\ln d_{j,i+1} - \ln d_{j1})$, $\sum_{i=1}^{m_j - 1} u_{ji} \le 1$, $u_{ji} \in \{0, 1\}$, *for* $1 \le j \le q$, *and* $y_j = \sum_{i=1}^{m_j} u_{ji} d_{ji}$, $z_j = \sum_{i=1}^{m_j} u_{ji}(\ln d_{ji})$, $\sum_{i=1}^{m_j} u_{ji} \le 1$, $u_{ji} \in \{0, 1\}$ *for* $q + 1 \le j \le m$, $L(e^{\alpha_1 z_1 + \cdots + \alpha_m z_m})$ *is a piecewisely linearized expression of* $e^{\alpha_1 z_1 + \cdots + \alpha_m z_m}$, *and* $\bar{s}$ *is the upper bound of* s.

Proof If there is $y_j = 0$ for some j $(q + 1 \le j \le m)$, then $\sum_{i=1}^{m_j} u_{ji} = 0$ and $s = 0$ by (i).

If $y_j > 0$ for all $j = q + 1, \cdots, m$, then $\sum_{i=1}^{m_j} u_{ji} = 1$ for $j = q + 1, \cdots, m$. Therefore we have $\sum_{j=q+1}^{m} \sum_{i=1}^{m_j} u_{ji} - (m - q) = 0$ if all variables in the signomial term are not zero, and this implies $s = e^{\alpha_1 z_1 + \cdots + \alpha_m z_m}$ according to (ii). □

Remark 2 For a non-negative discrete variable y, $y \in \{d_1, d_2, \cdots, d_m\}$, $0 \le d_1 < d_2 < \cdots < d_m$, the exponential term y^α where α is a real constant can be represented as

$$y^\alpha = d_1^\alpha + \sum_{i=1}^{m-1} u_i (d_{i+1}^\alpha - d_1^\alpha) \quad \text{where} \quad \sum_{i=1}^{m-1} u_i \le 1,\ u_i \in \{0, 1\}.$$

According to the above discussions, free discrete variables in GGP can be converted into positive discrete variables. In addition, Li and Tsai method [18] can deal with the free continuous variables. Consequently, the GGP program with continuous and discrete free variables can be transformed into a GGP program with only positive variables. In order to obtain a global optimum of the transformed GGP program, it is required to be converted into a convex mixed-integer problem which is solvable by the conventional convex mixed-integer techniques to derive a globally optimal solution.

Convexification Strategies. Convexification strategies for signomial terms are important techniques for global optimization problems. Sun et al. [25] proposed a convexification method for a class of global optimization problems with monotone functions under some restrictive conditions. Wu et al. [29] developed a more general convexification and concavification transformation for solving a general global optimization problem with certain monotone properties. With different convexification approaches, an MINLP problem can be reformulated into another convex mixed-integer program solvable to obtain an approximately global optimum. Björk et al. [4] proposed a global optimization technique based on convexifying signomial terms. They discussed that the right choice of transformation for convexifying nonconvex signomial terms has a clear impact on the efficiency of the optimization approach. Tsai et al. [26] also suggested convexification techniques for the signomial terms with three variables. This study presents generalized convexification techniques and rules to transform a nonconvex GGP program with continuous and discrete variables into a convex mixed-integer program. Consider the following propositions:

Lemma 1 *For a twice-differentiable function* $f(X) = c \prod_{i=1}^{n} x_i^{\alpha_i}$, $X = (x_1, x_2, \cdots, x_n)$, $c, x_i, \alpha_i \in \Re$, $\forall i$, *let* $H_i(X)$ *be the ith principal minor of a Hessian matrix* $H(X)$ *of* $f(X)$. *The determinant of* $H_i(X)$ *can be expressed as* $\det H_i(x) =$

$$(-c)^i \left(\prod_{j \in J_i}^{i} \alpha_j x_j^{i\alpha_j - 2} \right) \left(\prod_{\substack{j \notin J_i \\ J_i \neq \Phi}}^{n} x_j^{i\alpha_j} \right) \left(1 - \sum_{j \in J_i} \alpha_j \right).$$

Remark 3 If $c \ge 0$, $x_i \ge 0$ and $\alpha_i \le 0$ (for all i), then $\det H_i(x) \ge 0$.

Remark 4 If $c < 0$, $x_i \geq 0$, $\alpha_i \geq 0$ (for all i), and, $1 - \sum_{i=1}^{n} \alpha_j \geq 0$, then $\det H_i(x) \geq 0$.

Proposition 3 *A twice-differentiable function* $f(X) = c \prod_{i=1}^{n} x_i^{\alpha_i}$ *is convex for* $c, x_i \geq 0, \alpha_i \leq 0, i = 1, 2, \cdots, n$.

Proof By Lemma 1 and Remark 3, $\det H_i(x) = (-c)^i \left(\prod_{j \in J_i}^{i} \alpha_j x_j^{i\alpha_j - 2} \right) \left(\prod_{\substack{j \notin J_i \\ J_i \neq \Phi}}^{n} x_j^{i\alpha_j} \right) \left(1 - \sum_{j \in J_i} \alpha_j \right) \geq 0$ for $i = 1, 2, \cdots, n$, when $c, x_i \geq 0$, $\alpha_i \leq 0$, $i = 1, 2, \cdots, n$. Since $\det H_i(x) \geq 0$ for all i, $H_i(X)$ is positive semi-definite and $f(X)$ is convex. □

Proposition 4 *A twice-differentiable function* $f(X) = c \prod_{i=1}^{n} x_i^{\alpha_i}$ *is convex for* $c < 0$, $x_i, \alpha_i \geq 0$ *(for* $i = 1, 2, \cdots, n$*), and* $1 - \sum_{i=1}^{n} \alpha_i \geq 0$.

Proof By Lemma 1 and Remark 4, $\det H_i(x) = (-c)^i \left(\prod_{j \in J_i}^{i} \alpha_j x_j^{i\alpha_j - 2} \right) \left(\prod_{\substack{j \notin J_i \\ J_i \neq \Phi}}^{n} x_j^{i\alpha_j} \right) \left(1 - \sum_{j \in J_i} \alpha_j \right) \geq 0$ for $i = 1, 2, \cdots, n$, when $c < 0$, $x_i, \alpha_i \geq 0$, and $1 - \sum_{i=1}^{n} \alpha_i \geq 0$. Since $\det H_i(x) \geq 0$ for all i, $H_i(X)$ is positive semi-definite and $f(X)$ is convex. □

For a given signomial term s, if s can be converted into a set of convex terms satisfying Proposition 3 and 4, then the whole solution process is more computationally efficient. Under this condition, s does not necessitate the exponential transformation. For instance, $s = x_1^{-1} x_2^{-2} x_3^{-1}$ with $x_1, x_2, x_3 \geq 0$ is a convex term requir-breaking no transformation by Proposition 3, and $s = -x_1^{0.2} x_2^{0.7}$ with $x_1, x_2 \geq 0$ is also a convex term by Proposition 4.

Remark 5 A product term $z = uf(x)$ is equivalent to the following linear inequalities:

$$(i)\ M(u-1) + f(x) \leq z \leq M(1-u) + f(x),$$
$$(ii)\ -Mu \leq z \leq Mu,$$

where $u \in \{0, 1\}$, z is an unrestricted in sign variable, and $M = \max f(x)$ is a large constant.

Remark 6 The product term $u_1 u_2 \cdots u_m$ where $u_i \in \{0, 1\}$ for $i = 1, 2, \cdots, m$ can be replaced by a variable u expressed as

$$(i)\ 0 \leq u \leq u_i, \text{ for } i = 1, 2, \cdots, m,$$
$$(ii)\ u \geq \sum_{i=1}^{m} u_i - m + 1.$$

Following the above discussions, herein we take a signomial term with three variables for instance to describe the strategy of convexification. The strategy can also be extended to convexity a signomial term containing n variables.

Consider a signomial term $cx_1^{\alpha} x_2^{\beta} x_3^{\gamma}$ composed of three positive variables, the term $cx_1^{\alpha} x_2^{\beta} x_3^{\gamma}$ can be convexified by the following rules:

Rule 1 If $c > 0$, $\alpha, \beta, \gamma < 0$, then $cx_1^{\alpha} x_2^{\beta} x_3^{\gamma}$ is already a convex term by Proposition 3.

Rule 2 If $c > 0$, $\alpha, \beta < 0$, and $\gamma > 0$, then let $cx_1^{\alpha} x_2^{\beta} x_3^{\gamma} = cx_1^{\alpha} x_2^{\beta} z_1^{-\gamma}$ where $z_1 = x_3^{-1}$. The term $cx_1^{\alpha} x_2^{\beta} z_1^{-\gamma}$ is convex by Rule 1.

Rule 3 If $c > 0$, $\alpha < 0$, and $\beta, \gamma > 0$, then let $cx_1^{\alpha} x_2^{\beta} x_3^{\gamma} = cx_1^{\alpha} z_1^{-\beta} z_2^{-\gamma}$ where $z_1 = x_2^{-1}, z_2 = x_3^{-1}$. The term $cx_1^{\alpha} z_1^{-\beta} z_2^{-\gamma}$ is convex by Rule 1.

Rule 4 If $c > 0$ and $\alpha, \beta, \gamma > 0$, then let $cx_1^{\alpha} x_2^{\beta} x_3^{\gamma} = ce^{\alpha \ln x_1 + \beta \ln x_2 + \gamma \ln x_3}$.

Rule 5 If $c < 0, \alpha, \beta, \gamma \geq 0$, and $\alpha + \beta + \gamma \leq 1$, then $cx_1^{\alpha} x_2^{\beta} x_3^{\gamma}$ is already a convex term by Proposition 4.

Rule 6 If $c < 0$, $\alpha, \beta > 0$, $\alpha + \beta < 1$, then let $cx_1^{\alpha} x_2^{\beta} x_3^{\gamma} = cx_1^{\alpha} x_2^{\beta} z_1^{1-\alpha-\beta}$ where $z_1 = x_3^{\gamma/(1-\alpha-\beta)}$. The term $cx_1^{\alpha} x_2^{\beta} z_1^{1-\alpha-\beta}$ is convex by Rule 5.

Rule 7 If $c < 0$, $0 < \alpha < 1$, then let $cx_1^{\alpha} x_2^{\beta} x_3^{\gamma} = cx_1^{\alpha} z_1^{(1-\alpha)/2} z_2^{(1-\alpha)/2}$ where $z_1 = x_2^{2\beta/(1-\alpha)}$ and $z_2 = x_3^{2\gamma/(1-\alpha)}$. The term $cx_1^{\alpha} z_1^{(1-\alpha)/2} z_2^{(1-\alpha)/2}$ is convex by Rule 5.

Rule 8 If $c < 0$ and "$\alpha, \beta, \gamma < 0$ or $\alpha, \beta, \gamma \geq 1$", then let $cx_1^{\alpha} x_2^{\beta} x_3^{\gamma} = cz_1^{\frac{1}{3}} z_2^{\frac{1}{3}} z_3^{\frac{1}{3}}$ where $z_1 = x_1^{3\alpha}, z_2 = x_2^{3\beta}$, and $z_3 = x_3^{3\gamma}$. The term $cz_1^{\frac{1}{3}} z_2^{\frac{1}{3}} z_3^{\frac{1}{3}}$ is convex by Rule 5.

Rule 9 If $\alpha, \beta > 0$, $x_1 \in Z$, $x_3 = 1$ and $\alpha + \beta > 1$, then let $cx_1^{\alpha}x_2^{\beta} = c\left[d_{11}^{\alpha} + \sum_{i=1}^{m_1-1} u_{1i}(d_{1,i+1}^{\alpha} - d_{11}^{\alpha})\right]x_2^{\beta}$ for $i \in \{1, 2, \cdots, m_1 - 1\}$. By Remark 5, the product term $u_{1i}x_2^{\beta}$ can be transformed into linear inequalities.

By applying the proposed rules, we can determine certain classes of signomial terms are convex and do not necessitate any transformation. Besides, we can transform a nonconvex signomial term into a convex term accordance with the proposed rules by replacing some variables, thereby decreasing the number of concave functions requiring to be estimated and making the resulting problem a computationally efficient model.

In order to be a valid transformation in the global optimization procedure, the transformation should be selected such that the signomial terms are not only convexified but also underestimated [4,21,27]). If the transformations are appropriately selected, the corresponding approximate signomial term will underestimate the original convexified signomial term by applying piecewise linear approximations to the inverse transformation functions. We examine the proposed rules can satisfy the underestimating condition as follows:

In Rule 2, let $\hat{z}_1$ be the approximate transformation variable obtained from piecewise linear function of $z_1 = x_3^{-1}$. The inverse transformation $z_1 = x_3^{-1}(x_3 > 0)$ is convex and z_1 will be overestimated ($\hat{z}_1 > z_1$). When inserting the approximate variable in the signomial term, we find the underestimating property $cx_1^{\alpha}x_2^{\beta}\hat{z}_1^{-\gamma} \le cx_1^{\alpha}x_2^{\beta}z_1^{-\gamma}$ is fulfilled since $c > 0$ and z_1 has a negative power in the convexified term. Similarly, Rules 3 and 4 meet the underestimating condition.

In Rule 6, let $\hat{z}_1$ be the approximate transformation variable obtained from piecewise linear function of $z_1 = x_3^{\gamma/(1-\alpha-\beta)}$. The inverse transformation $z_1 = x_3^{\gamma/(1-\alpha-\beta)}$ ($x_3 > 0$, $\frac{\gamma}{1-\alpha-\beta} > 1$ or $\frac{\gamma}{1-\alpha-\beta} \le 0$) is convex and z_1 will be overestimated ($\hat{z}_1 > z_1$). When inserting the approximate variable in the signomial term, we find the underestimating property $cx_1^{\alpha}x_2^{\beta}\hat{z}_1^{1-\alpha-\beta} \le cx_1^{\alpha}x_2^{\beta}z_1^{1-\alpha-\beta}$ is fulfilled since $c < 0$ and z_1 has a positive power in the convexified term. Similarly, Rules 7 and 8 satisfy the underestimating property.

From above discussions, we observe the proposed rules not only convexity but underestimate the convexified signomial term. Consequently, utilizing the transformations in the global optimization of a GGP problems, the feasible region of the convexified problem overestimates the feasible region of the original nonconvex problem.

Case Studies

Case1 Minimize $x_1^3x_2^{1.5}x_3^3 + x_2^{5.5}x_3 + x_1^5$

subject to

$$3x_1 + 2x_2 - x_3 \le 7,$$
$$-5 \le x_1 \le 2,\ 0 \le x_2 \le 4,\ -5 \le x_3 \le -1,$$
$$x_1, x_2 \in \mathrm{Z}, \quad x_3 \in \Re.$$

This problem is a nonconvex GGP program with continuous and discrete free variables. Current exponential transformation methods [8,9,11,20]) developed for solving mixed-integer GGP problems can not be adopted to treat this kind of problems. By employing the proposed method, we first utilize a straightforward substitution for the free variables to make the GGP problem with only non-negative variables. By Li and Tsai [18], let the free continuous variable $x_3 = -x_3^-$, $x_3^- \ge 0$.

The free discrete variable can be transformed by Remark 1, $x_1 = x_1^+ - x_1^-$, $x_1^+, x_1^- \ge 0$. The original problem becomes as follows:

$$\begin{aligned}
\text{Minimize} \quad & -(x_1^+)^3x_2^{1.5}(x_3^-)^3 + (x_1^-)^3x_2^{1.5}(x_3^-)^3 - \\
& x_2^{5.5}x_3^- + (x_1^+)^5 - (x_1^-)^5 \\
\text{subject to} \quad & 3x_1^+ - 3x_1^- + 2x_2 + x_3^- \le 7, \\
& 0 \le x_1^+ \le 2,\ 0 \le x_1^- \le 5.0 \le x_2 \le 4, \\
& 1 \le x_3^- \le 5, \quad x_1^+, x_1^-, x_2 \in \mathrm{Z}, \quad x_3^- \in \Re.
\end{aligned}$$

Then, we use the proposed convexification rules to transform all signomial terms into convex terms as follows:

(i) $z_1 = (x_1^+)^3x_2^{1.5}(x_3^-)^3$ and $z_2 = (x_1^-)^3x_2^{1.5}(x_3^-)^3$ are transformed by Rule 4 and Proposition 2.

(ii) $-x_2^{5.5}x_3^-$ is convexified as $-x_2^{5.5}x_3^- = -(u_{21} + 2^{5.5}u_{22} + 3^{5.5}u_{23} + 4^{5.5}u_{24})x_3^- = -z_3 - 2^{5.5}z_4 - 3^{5.5}z_5 - 4^{5.5}z_6$ by Rule 9.

(iii) $(x_1^+)^5$ and $-(x_1^-)^5$ are treated directly as $(x_1^+)^5 = u_{11}^+ + 2^5 u_{12}^+, -(x_1^-)^5 = -u_{11}^- - 2^5 u_{12}^- - 3^5 u_{13}^- - 4^5 u_{14}^- - 5^5 u_{15}^-$ by Remark 2.

Subsequently, the transformed program is presented as a convex mixed-integer program below:

Minimize

$$-z_1 + z_2 - z_3 - 2^{5.5} z_4 - 3^{5.5} z_5 - 4^{5.5} z_6 + z_7 - z_8$$

subject to

$$3x_1^+ - 3x_1^- + 2x_2 + x_3^- \le 7,$$
$$-x_1^- \le x_1 \le 5\theta_1 - x_1^-,$$
$$5(\theta_1 - 1) + x_1^+ \le x_1 \le x_1^+,$$
$$x_1^+ = u_{11}^+ + 2u_{12}^+, \quad y_1^+ = u_{12}^+ \cdot \ln 2,$$
$$x_1^- = u_{11}^- + 2u_{12}^- + 3u_{13}^- + 4u_{14}^- + 5u_{15}^-,$$
$$y_1^- = u_{12}^- \cdot \ln 2 + u_{13}^- \cdot \ln 3 + u_{14}^- \cdot \ln 4 + u_{15}^- \cdot \ln 5,$$
$$u_{11}^+ + u_{12}^+ \le 1, u_{11}^- + u_{12}^- + u_{13}^- + u_{14}^- + u_{15}^- \le 1,$$
$$x_2 = u_{21} + 2u_{22} + 3u_{23} + 4u_{24},$$
$$y_2 = u_{22} \cdot \ln 2 + u_{23} \cdot \ln 3 + u_{24} \cdot \ln 4,$$
$$u_{21} + u_{22} + u_{23} + u_{24} \le 1,$$
$$y_3^- = L(\ln x_3^-),$$
$$0 \le z_1 \le \bar{z}(u_{11}^+ + u_{12}^+),$$
$$0 \le z_1 \le \bar{z}(u_{21} + u_{22} + u_{23} + u_{24}),$$
$$\bar{z}(u_{11}^+ + u_{12}^+ + u_{21} + u_{22} + u_{23} + u_{24} - 2) + e^{3y_1^+ + 1.5y_2 + 3y_3^-} \le z_1,$$
$$z_1 \le \bar{z}\,(2 - (u_{11}^+ + u_{12}^+ + u_{21} + u_{22} + u_{23} + u_{24})) + L(e^{3y_1^+ + 1.5y_2 + 3y_3^-}),$$
$$0 \le z_2 \le \bar{z}(u_{11}^- + u_{12}^- + u_{13}^- + u_{14}^- + u_{15}^-),$$
$$0 \le z_2 \le \bar{z}(u_{21} + u_{22} + u_{23} + u_{24}),$$
$$\bar{z}(u_{11}^- + u_{12}^- + u_{13}^- + u_{14}^- + u_{15}^- + u_{21} + u_{22} + u_{23} + u_{24} - 2) + e^{3y_1^- + 1.5y_2 + 3y_3^-} \le z_2,$$
$$z_2 \le \bar{z}(2 - (u_{11}^- + u_{12}^- + u_{13}^- + u_{14}^- + u_{15}^- + u_{21} + u_{22} + u_{23} + u_{24})) + L(e^{3y_1^- + 1.5y_2 + 3y_3^-}),$$
$$5(u_{21}^- 1) + x_3^- \le z_3 \le x_3^-, \quad 0 \le z_3 \le 5u_{21},$$
$$5(u_{22}^- 1) + x_3^- \le z_4 \le x_3^-, \quad 0 \le z_4 \le 5u_{22},$$
$$5(u_{23}^- 1) + x_3^- \le z_5 \le x_3^-, \quad 0 \le z_5 \le 5u_{23},$$
$$5(u_{24}^- 1) + x_3^- \le z_6 \le x_3^-, \quad 0 \le z_6 \le 5u_{24},$$
$$z_7 = u_{11}^+ + 2^5 u_{12}^+,$$
$$z_8 = u_{11}^- + 2^5 u_{12}^- + 3^5 u_{13}^- + 4^5 u_{14}^- + 5^5 u_{15}^-,$$
$$(0,0,0,1,0,0,0) \le (x_1^+, x_1^-, x_2, x_3^-, y_1^+, y_1^-, y_2, y_3^-) \le (2,5,4,5,\ln 2, \ln 5, \ln 4, \ln 5),$$

where $u_{ij}, u_{ij}^+, u_{ij}^-, \theta_1 \in \{0,1\}$, and $\bar{z} = 125{,}000$.

Solving the original problem without any variable transformation and convexification by LINGO [19], a local optimum obtained is $(x_1, x_2, x_3) = (-5, 0, -5)$ and the objective value is –3125. However, solving the above transformed convex mixed-integer program within the tolerable error 0.001, the globally optimal solution obtained is $(x_1, x_2, x_3) = (-2, 4, -3.266)$ and the objective value found is –4491.16.

Case2 Minimize $x_1^{0.5} x_2 + 3 \ln x_1$ subject to

$$-x_1 + x_2 \le 5$$
$$x_1^{0.5} y - x_2 \le 6,$$
$$x_1 \in \{0.1, 0.5, 0.7, 1.2\}, -6 \le x_2 \le 4, \quad y \in \{0,1\}.$$

This problem contains a discrete variable, a free continuous variable and a binary variable which cannot be treated by the exponential-based methods. The nonlinear terms $x_1^{0.5} x_2$, $3 \ln x_1$ and $x_1^{0.5} y$ are nonconvex functions. By Remarks 2, 5 and 6, the problem can be equivalently transformed into a linear mixed-integer programming problem as follows.

Minimize

$$0.1^{0.5} x_2 + (0.5^{0.5} - 0.1^{0.5}) s_1 + (0.7^{0.5} - 0.1^{0.5}) s_2 + (1.2^{0.5} - 0.1^{0.5}) s_3 + 3(\ln 0.1 + (\ln 0.5 - \ln 0.1) u_1 + (\ln 0.7 - \ln 0.1) u_2 + (\ln 1.2 - \ln 0.1) u_3)$$

subject to

$$-x_1 + x_2 \le 5, \quad x_1 = 0.1 + (0.5 - 0.1)u_1 + (0.7 - 0.1)u_2 + (1.2 - 0.1)u_3,$$
$$u_1 + u_2 + u_3 \le 1, \quad 0.1^{0.5} y + (0.5^{0.5} - 0.1^{0.5}) z_1 + (0.7^{0.5} - 0.1^{0.5}) z_2 + (1.2^{0.5} - 0.1^{0.5}) z_3 - x_2 \le 6,$$
$$0 \le z_i, \quad z_i \le u_i, \quad z_i \le y, \quad z_i \ge u_i + y - 1,$$
$$i = 1,2,3, \quad -6u_i \le s_i \le 6u_i, \quad 6(u_i - 1) + x_2 \le s_i \le 6(1 - u_i) + x_2, \quad i = 1,2,3,$$
$$s_1, s_2, s_3 \text{ are unrestricted in sign variables},$$
$$u_1, u_2, u_3 \in \{0,1\}, -6 \le x_2 \le 4.$$

The transformed program can be solved to locate the globally optimal solution $(x_1, x_2, y) = (0.1, -6, 0)$. The objective value is –8.805.

Conclusions

This paper proposes a generalized method to solve the globally optimal solutions of GGP problems with continuous and discrete free variables. The techniques of dealing with free variables aim to change variables and to convert the logical relationship among the variables in a product term into a set of linear inequalities, which can be merged conveniently into the GGP models. Compared with current GGP methods, the proposed method is capable of dealing with free variables of a GGP problem and is guaranteed to converge to a global optimum. In addition, several computationally efficient convexification rules for signomial terms are presented to enhance the efficiency of the optimization approach.

References

1. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB, for general twice-differentiable NLPs–II. Implementation and computational results. Comput Chem Eng 22:1159–1179
2. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB, for general twice-differentiable NLPs–I. Theoretical advances. Comput Chem Eng 22:1137–1158
3. Biegler LT, Grossmann IE (2004) Retrospective on optimization. Comput Chem Eng 28:1169–1192
4. Björk KM, Lindberg PO, Westerlund T (2003) Some convexifications in global optimization of problems containing signomial terms. Comput Chem Eng 27:669–679
5. Cardoso MF, Salcedo RL, Feyo de Azevedo S (1996) The simplex-simulated annealing approach to continuous nonlinear optimization. Comput Chem Eng 20:1065–1080
6. Duran M, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed integer nonlinear programs. Math Program 36:307–339
7. Fletcher R, Leyffer S (1994) Solving Mixed Integer Nonlinear Programs by outer approximation. Math Program 66:327–349
8. Floudas CA (1995) Nonlinear and mixed integer optimization: Fundamentals and applications. Oxford Univ Press, New York
9. Floudas CA (2000) Deterministic global optimization: Theory, methods and applications. Kluwer, Boston
10. Floudas CA (2007) On the functional form of convex underestimators for twice continuously differentiable functions. Optim Lett 1:187–192
11. Floudas CA, Pardalos PM (1996) State of the art in global optimization: Computational methods and applications. Kluwer, Boston
12. Geoffrion AM (1972) Generalized Benders Decomposition. J Optim Theory Appl 10:237–260
13. Grossmann IE (2002) Review of nonlinear mixed-integer and disjunctive programming techniques. Optim Eng 3:227–252
14. Grossmann IE, Biegler LT (2004) Part II. Future perspective on optimization. Comput Chem Eng 28:1193–1218
15. Hussain MF Al-Sultan KS (1997) A hybrid genetic algorithm for nonconvex function minimization. J Global Optim 11:313–324
16. Lee S, Grossmann IE (2000) New algorithms for nonlinear generalized disjunctive programming. Comput Chem Eng 24:2125–2142
17. Leyffer S (2001) Integrating SQP and branch-and-bound for mixed integer nonlinear programming. Comput Optim Appl 18:295–309
18. Li HL, Tsai JF (2005) Treating free variables in generalized geometric global optimization programs. J Global Optim 33(1):1–13
19. Lingo Release 9.0 (2004) Lindo System Inc, Chicago
20. Maranas CD, Floudas CA (1997) Global optimization in generalized geometric programming. Comput Chem Eng 21:351–370
21. Pörn R, Harjunkoski I, Westerlund T (1999) Convexification of different classes of non-convex MINLP problems. Comput Chem Eng 23:439–448
22. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput Chem Eng 16:937–947
23. Salcedo RL, Goncalves MJ, Feyo de Azevedo S (1990) An improved random-search algorithm for nonlinear optimization. Comput Chem Eng 14:1111–1126
24. Stubbs RA, Mehrotra S (1999) A branch-and-cut method for 0–1 mixed convex programming. Math Program 86:515–532
25. Sun XL, Mckinnon K, Li D (2001) A convexification method for a class of global optimization problems with applications to reliability optimization. J Global Optim 21:185–199
26. Tsai JF, Li HL, Hu NZ (2002) Global optimization for signomial discrete programming problems in engineering design. Eng Optim 34:613–622
27. Westerlund T (2005) Some transformation techniques in global optimization. In: Liberti L, Maculan N (eds) From theory to implementations, pp 45–74. Kluwer, Boston
28. Westerlund T, Pettersson F (1995) An extended cutting plane method for solving convex MINLP problems. Comput Chem Eng 19:131–136
29. Wu ZY, Bai FS, Zhang LS (2005) Convexification and concavification for a general class of global optimization problems. J Global Optim 31:45–60
30. Yiu KFC, Liu Y, Teo KL (2004) A hybrid descent method for global optimization. J Global Optim 28:229–238

Generalized Monotone Multivalued Maps

GMMVM

NICOLAS HADJISAVVAS[1], SIEGFRIED SCHAIBLE[2]
[1] Department Math., University Aegean, Karlovassi, Greece
[2] A.G. Anderson Graduate School of Management, University California, Riverside, USA

MSC2000: 90C26

Article Outline

Keywords
The Subdifferential
The Monotone Case
The Quasimonotone Case
The Pseudomonotone Case
See also
References

Keywords

Generalized convexity; Generalized monotonicity; Quasiconvex function; Quasimonotone operator

The study of multivalued generalized monotone operators is a recent(as of 1999) subject. The first to introduce such a notion seem to have been L.G. Mitjuschin and W.M. Polterovich [16] who defined multivalued quasimonotone operators in demand theory. The same concept was also defined by A. Hassouni [10] and D.T. Luc [14]. Later, Luc [15] and J.P. Penot and P.H. Quang [20] proceeded to define new kinds of generalized monotonicity for multivalued operators. Alarge part of this effort has been devoted to the definition of appropriate concepts so that generalized convex nonsmooth functions are characterized by the generalized monotonicity of their subdifferentials [18].

As it stands today, the theory is not at the stage of development of the corresponding theory for single valued operators (see ► Generalized monotone single valued maps). More concepts have to be introduced and probably some of the already existing ones have to be modified so that a nice correspondence such as the one exhibited in the first theorem of ► Generalized monotone single valued maps can be established, without imposing any additional assumptions. This concerns both generalized monotonicity of multivalued operators and generalized convexity of nonsmooth functions, as some notions of generalized convexity involve subdifferentials.

This article presents various definitions of generalized monotonicity for multivalued operators and generalized convexity for nonsmooth functions. Also, various characterizations of generalized convexity of a function through the corresponding generalized monotonicity of the subdifferential are surveyed. Some characterizations have a 'mixed' form, i. e., they involve both the function and its subdifferential.

The next section contains the definition of the subdifferential for lower semicontinuous functions, along with the necessary notation. Then the less known correspondence between the convexity of a function and the monotonicity of its subdifferential is presented. In the main part of the article, this correspondence is extended to cover the various cases of generalized convexity and generalized monotonicity.

The Subdifferential

There is a host of nonequivalent subdifferentials for nonconvex functions. The interested reader may find a thorough exposition of the various concepts in [19]. The most common, the Clarke–Rockafellar subdifferential, is the one that will be used here, although many of the results hold also for a large number of other subdifferentials; see for instance [1,18,19]. Generalized monotonicity of bifunctions is used in [13] to characterize generalized convex functions through various generalized derivatives.

In this article, X denotes a Banach space, X^* its dual, and $f: X \to \mathbf{R} \cup \{+\infty\}$ a lower semicontinuous (lsc) function with nonempty domain $\mathrm{dom}(f) = \{x \in X: f(x) \neq +\infty\}$. The function f is called *radially continuous* if its restriction to line segments is continuous. The value of a functional $x^* \in X^*$ at a point $x \in X$ will be denoted by $\langle x^*, x\rangle$. Given $x, y \in X$, (x, y) is the open line segment $\{tx + (1-t)y: t \in (0, 1)\}$. The line segments $[x, y]$, $[x, y)$ and $(x, y]$ aredefined analogously.

The *Clarke–Rockafellar generalized derivative* of f at $x_0 \in \mathrm{dom}(f)$ in the direction $d \in X$ is given by

$$f^{\uparrow}(x_0, d) = \sup_{\varepsilon>0} \limsup_{\substack{x \to_f x_0 \\ t \searrow 0}} \inf_{d' \in B_\varepsilon(d)} \frac{f(x+td') - f(x)}{t}.$$

Here, $t \searrow 0$ is used to denote the fact that $t > 0$ and $t \to 0$, and $x \to_f x_o$ means that both $x \to x_o$ and $f(x) \to f(x_o)$.

The (Clarke–Rockafellar) *subdifferential* of f at $x_0 \in \mathrm{dom}(f)$ is defined by

$$\partial f(x_0) = \left\{x^* \in X^* : \ \langle x^*, d\rangle \le f^{\uparrow}(x_0, d), \ \forall d \in X\right\},$$

while for $x_0 \in X \setminus \mathrm{dom}(f)$, $\partial f(x_0) = \emptyset$.

Even for $x_0 \in \mathrm{dom}(f)$, the subdifferential $\partial f(x_0)$ may be empty. Whenever the function f is locally Lipschitz, one has $\partial f(x_0) \neq \emptyset$, for all $x_0 \in \mathrm{dom}(f)$. In this case $f^{\uparrow}$ coincides with the *Clarke generalized derivative*:

$$f^o(x_0; d) = \limsup_{\substack{x \to x_0 \\ t \searrow 0}} \frac{f(x+td) - f(x)}{t}.$$

In case f is convex, ∂f coincides with the classical *Fenchel–Moreau subdifferential*

$$\partial f(x_0) = \{x^* \in X^* : \ \langle x^*, d\rangle \le f(x_0 + d) - f(x_0)\}.$$

The Monotone Case

Let $T: X \to 2^{X^*}$ be a multivalued operator with domain

$$D(T) = \{x \in X : \ T(x) \neq \emptyset\}.$$

The operator T is called:

- *monotone*, if for all $x, y \in X$ and

  $$x^* \in T(x), y^* \in T(y)$$

  one has

  $$\langle y^* - x^*, y - x\rangle \ge 0; \tag{1}$$

- *strictly monotone*, if for all $x \neq y$ the above inequality is strict.

It is well known that the subdifferential of a convex function is amonotone operator. However, the fact that convex functions are characterized by the monotonicity of their Clarke–Rockafellar subdifferentials is a relatively recent result. In addition, there exists a 'mixed' characterization of convexity, involving both the function and its subdifferential:

Theorem 1 *Let f be lsc. The following are equivalent:*

i) The function f is convex.

ii) For all $x, y \in dom(f)$ and all $x^* \in \partial f(x)$ one has:

$$\langle x^*, y - x\rangle \le f(y) - f(x). \tag{2}$$

iii) The subdifferential ∂f is a monotone operator.

The implication i)⇒ii) follows from the equality of the Clarke–Rockafellar and the Fenchel–Moreau subdifferential for convex functions. The implication ii)⇒iii) is shown in every textbook on monotone operators. Finally, the implication iii)⇒i) is shown in [4].

An analogous theorem holds for strictly convex functions (see ▶ Generalized monotone single valued maps for definitions of the various kinds of convexity and generalized convexity):

Theorem 2 *Let f be lsc. Consider the following assertions:*

i) The function f is strictly convex.

ii) For all distinct $x, y \in dom(f)$ and

$$x^* \in \partial f(x),$$

one has

$$\langle x^*, y - x\rangle < f(y) - f(x).$$

iii) The subdifferential ∂f is a strictly monotone operator.

Then i)⇒ii)⇒iii). If, in addition, $\partial f(x) \neq \emptyset$ for all $x \in dom(f)$, then iii)⇒i).

For the proof, see [8].

The Quasimonotone Case

The concepts of quasimonotone, semistrictly quasimonotone and strictly quasimonotone maps are direct generalizations of the corresponding concepts for single valued maps (see ▶ Generalized monotone single valued maps and [9,12]). A multivalued operator $T: X \to 2^{X*}$ is called:

- *quasimonotone* [14], if for all $x, y \in X$ and all $x^* \in T(x), y^* \in T(y)$, the following implication holds:

$$\langle x^*, y - x\rangle > 0 \;\Rightarrow\; \langle y^*, y - x\rangle \geq 0;$$

- *semistrictly quasimonotone* [5], if it is quasimonotone and for any distinct

$$x, y \in D(T)$$

one has the implication:

$$\begin{aligned} &\exists x^* \in T(x):\ \langle x^*, y - x\rangle > 0 \\ \Rightarrow\ &\exists z \in \left(\frac{x+y}{2}, y\right),\ \exists z^* \in T(z):\\ &\langle z^*, y - x\rangle > 0; \end{aligned} \tag{3}$$

- *strictly quasimonotone* [5], if it is quasimonotone and for any distinct $x, y \in D(T)$, there exists $z \in (x, y)$ and $z^* \in T(z)$such that $\langle z^*, y-x\rangle \neq 0$.

It can be shown [5] that relation (3) is equivalentto the following: if $\langle x^*, y- x\rangle > 0$ for some $x^* \in T(x)$, then the set of all $z \in (x, y)$ for which there exists $z^* \in T(z)$ such that $\langle z^*, y- x\rangle > 0$, is dense in $[x, y]$.

In the single valued case, whenever T is a gradient, its quasimonotonicity, semistrict quasimonotonicity and strict quasimonotonicity is equivalent to quasiconvexity, semistrict quasiconvexity and strict quasiconvexity of the underlying function, respectively (see ► Generalized monotone single valued maps for the corresponding definitions, and [3] for properties of such functions). Analogous results hold for multivalued operators which are subdifferentials. The next theorem gives two equivalent characterizations of quasiconvexity: one 'mixed', andone through the quasimonotonicity of the subdifferential.

Theorem 3 *Let f be lsc. The following are equivalent:*

i) The function f is quasiconvex.

ii) For all $x, y \in dom(f)$, the following implication holds:

$$\begin{aligned} &\exists x^* \in \partial f(x):\ \langle x^*, y - x\rangle > 0 \\ &\Rightarrow \forall z \in [x, y]:\ f(z) \leq f(y). \end{aligned} \tag{4}$$

iii) The operator ∂f is quasimonotone.

The equivalence i)⇔iii) is shown in [14, Thm. 3.2], while the equivalence i)⇔ii) is shown in [1, Thm. 2.1]. In [1] it is also shown that, in case f is radially continuous, implication (4) is equivalent to the following implication:

$$\exists x^* \in \partial f(x):\ \langle x^*, y - x\rangle > 0 \;\Rightarrow\; f(x) \leq f(y).$$

A 'mixed' characterization exists also for semistrictly quasiconvex functions [5], but a continuity assumption stronger than lower semicontinuity is needed:

Theorem 4 *Let f be lsc. If f is semistrictly quasiconvex, then for all $x, y \in dom(f)$ one has:*

$$\begin{aligned} &\exists x^* \in \partial f(x):\ \langle x^*, y - x\rangle > 0 \\ &\Rightarrow \forall z \in [x, y):\ f(z) < f(y). \end{aligned} \tag{5}$$

The converse also holds if in addition f is radially continuous.

Radial continuity is an often used, weak continuity assumption. In fact, it is not as weak as it seems. Since X is a Banach space, it can be shown that a lsc quasiconvex function which is radially continuous is also continuous [8].

Characterization of strict or semistrict quasiconvexity via the generalized monotonicity of the subdifferential requires an even stronger continuity assumption:

Theorem 5 *A locally Lipschitz function f is strictly (respectively semistrictly) quasiconvex, if and only if its subdifferential is strictly (respectively semistrictly) quasimonotone.*

For the proof, see [5].

The Pseudomonotone Case

The definition of pseudomonotonicity for multivalued operators was given by J.C. Yao [21] and generalizes the corresponding definition for single valued operators (see ► Generalized monotone single valued maps and [11]). An operator

$$T:\ X \to 2^{X^*}$$

is called *pseudomonotone* if for all $x, y \in X$ one has:

$$\begin{aligned} &\exists x^* \in T(x):\ \langle x^*, y - x\rangle \geq 0 \\ &\Rightarrow\ \forall y^* \in T(y):\ \langle y^*, y - x\rangle \geq 0. \end{aligned}$$

Equivalently, an operator T is pseudomonotone if and only if the following implication holds:

$$\begin{aligned} &\exists x^* \in T(x):\ \langle x^*, y - x\rangle > 0 \\ &\Rightarrow\ \forall y^* \in T(y):\ \langle y^*, y - x\rangle > 0. \end{aligned} \tag{6}$$

Obviously, a pseudomonotone operator T is quasimonotone. If in addition the domain $D(T)$ is convex, then relation (6) implies that T is also semistrictly

quasimonotone. Also, it is clear that a monotone operatoris pseudomonotone.

An operator $T : X \to 2^{X^*}$ is called *strictly pseudomonotone* [21], if for all distinct $x, y \in X$ one has:

$$\exists x^* \in T(x):\ \langle x^*, y - x\rangle \geq 0 \\ \Rightarrow\ \forall y^* \in T(y):\ \langle y^*, y - x\rangle > 0.$$

It is clear that a strictly pseudomonotone operator is pseudomonotone, and that a strictly monotone operator is strictly pseudomonotone. Finally, it can easily be shown [8] that a strictly pseudomonotone operator with convex domain is strictly quasimonotone.

In summary, between the various concepts of generalized monotonicity, the following implications hold (some of which assume convexity of the domain):

$$\begin{array}{ccccc} & & & & qm \\ & & & & \Uparrow \\ m & \Rightarrow & pm & \Rightarrow & sstr.qm \\ \Uparrow & & \Uparrow & & \Uparrow \\ str.m & \Rightarrow & str.pm & \Rightarrow & str.qm \end{array}$$

Here, 'str.' and 'sstr.' stands for 'strictly' and 'semistrictly', respectively, and 'm', 'pm' and 'qm' for 'monotone', 'pseudomonotone' and 'quasimonotone', respectively. These implications are exactly the same as those holding for singlevalued operators (see ► Generalized monotone single valued maps).

In contrast to quasiconvex functions and their variants, pseudoconvex functions have to be redefined in the nonsmooth case. The reason is that the usual definition of pseudoconvexity makes explicit reference to the derivative of the function (however, there exists a definition which does not mention the derivative explicitly [17]; see also [3] for details).

A function f is called *pseudoconvex*, if for all $x, y \in \mathrm{dom}(f)$ the following implication holds:

$$\exists x^* \in \partial f(x):\ \langle x^*, y - x\rangle \geq 0 \\ \Rightarrow\ \forall z \in [x, y):\ f(z) \leq f(y). \tag{7}$$

Note that the above definition, expresses a 'mixed' property in the spirit of relation (4); actually, (7) is stronger than (4), and hence any pseudoconvex function is quasiconvex. In particular, a pseudoconvex function f has a convex domain. If in addition f is radially continuous, then it is semistrictly quasiconvex [8].

The definition of pseudoconvexity given here differs slightly from the definition introduced in [20]. There, a function f is called pseudoconvex if it satisfies the implication

$$\exists x^* \in \partial f(x):\ \langle x^*, y - x\rangle \geq 0 \ \Rightarrow\ f(x) \leq f(y). \tag{8}$$

A pseudoconvex function (as defined by relation (7)) obviously satisfies (8). The converse is not always true; however, if f is radially continuous, or if its domain is convex, then (8) implies that f is quasiconvex (see [20] and [6], respectively). It follows immediately that f satisfies (7), i. e., it is pseudoconvex.

The following theorem connects pseudoconvexity of a function to pseudomonotonicity of its subdifferential (see [8] and [20] for the proof of the first and the second assertion, respectively):

Theorem 6 *If f is pseudoconvex, then ∂f is pseudomonotone.Conversely, if ∂f is pseudomonotone and f is radially continuous, then f is pseudoconvex.*

A function f is called *strictly pseudoconvex* [8] if for all $x, y \in \mathrm{dom}(f)$ one has:

$$\exists x^* \in \partial f(x):\ \langle x^*, y - x\rangle \geq 0 \\ \Rightarrow\ \forall z \in [x, y):\ f(z) < f(y). \tag{9}$$

For radially continuous functions, relation (9) is equivalent to

$$\exists x^* \in \partial f(x):\ \langle x^*, y - x\rangle \geq 0 \ \Rightarrow\ f(x) < f(y). \tag{10}$$

Indeed, if relation (10) holds, then f is pseudoconvex, hence it is semistrictly quasiconvex. Consequently, if $\langle x^*, y - x\rangle \geq 0$ for some

$$x^* \in \partial f(x),$$

then $f(x) < f(y)$ implies that $f(z) < f(y)$ for all $z \in [x, y)$, i. e. (9) holds.

We have the following connection to strict pseudomonotonicity:

Theorem 7 *If f is strictly pseudoconvex, then ∂f is strictly pseudomonotone. Conversely, if ∂f is strictly pseudomonotone and its values are nonempty on dom(f), then f is strictly pseudoconvex.*

For the proof of the first assertion, see [20]; the second assertion is shown in [8].

As a corollary of the last theorem, it can be shown [8] that a locally Lipschitz, strictly pseudoconvex function f is strictly quasiconvex. Hence, between the various kinds of generalized convexity, the following implications hold (some implications need extra continuity assumptions):

$$\begin{array}{ccccc} & & & & qcx \\ & & & & \Uparrow \\ cx & \Rightarrow & pcx & \Rightarrow & sstr.qcx \\ \Uparrow & & \Uparrow & & \Uparrow \\ str.cx & \Rightarrow & str.pcx & \Rightarrow & str.qcx \end{array}$$

Here, 'cx', 'pcx' and 'qcx' stands for 'convex', 'pseudoconvex' and 'quasiconvex', respectively. Thus, the same implications hold as those fordifferentiable functions (see the corresponding diagram in ► Generalized monotone single valued maps). In addition, each type of generalized convex function is characterized by the corresponding generalized monotonicity of the subdifferential, exactly as in the case of differentiable functions (the first theorem in ► Generalized monotone single valued maps).

See also

- ► Fejér Monotonicity in Convex Optimization
- ► Generalized Monotone Single Valued Maps
- ► Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ► Set-valued Optimization

References

1. Aussel D (1998) Subdifferential properties of quasiconvex and pseudoconvex functions. J Optim Th Appl 97:29–45
2. Aussel D, Corvellec JN, Lassonde M (1995) Mean value property and subdifferential criteria for lower semicontinuous functions. Trans Amer Math Soc 347:4147–4161
3. Avriel M, Diewert WE, Schaible S, Zang I (1988) Generalized concavity. Plenum, New York
4. Correa R, Jofre A, Thibault L (1992) Characterization of lower semicontinuous convexfunctions. Proc Amer Math Soc 116:67–72
5. Daniilidis A, Hadjisavvas N (1999) Characterization of nonsmooth semistrictly quasiconvex and strictly quasiconvex functions. J Optim Th Appl 102:525–536
6. Daniilidis A, Hadjisavvas N (1999) On the subdifferentials of quasiconvex and pseudoconvex functions and cyclic monotonicity. J Math Anal Appl 237:30–42
7. Ellaia R, Hassouni A (1991) Characterization of nonsmooth functions through their generalized gradients. Optim 22:401–416
8. Hadjisavvas N (2001) The use of subdifferentials for studying generalized convex functions. J Statist Managem Systems no. March
9. Hadjisavvas N, Schaible S (1993) On strong pseudomonotonicity and (semi)strict quasimonotonicity. J Optim Th Appl 79:139–155
10. Hassouni A (1983) Sous-differentiels des fonctions quasiconvexes. Thése de 3ème cycle: Math Appliq Toulouse
11. Karamardian S (1976) Complementarityover cones with monotone and pseudomonotone maps. J Optim Th Appl 18:445–454
12. Karamardian S, Schaible S (1990) Seven kinds of monotone maps. J Optim Th Appl 66:37–46
13. Komlosi S (1995) Generalized monotonicity and generalized convexity. J Optim Th Appl 84:361–376
14. Luc DT (1993) Characterizations of quasiconvex functions. BullAustral Math Soc 48:393–405
15. Luc DT (1994) On generalized convex nonsmooth functions. Bull Austral Math Soc 49:139–149
16. Mitjuschin LG, Polterovich WM (1978) Criteria for monotonicity of demand functions. Ekonomika i Mat Metody 14:122–128 in Russian.
17. Ortega JM, Rheinboldt WC (1970) Iterative solution of nonlinear equations in several variables. Acad. Press, New York
18. Penot JP (1995) Generalized convexity in the light of nonsmooth analysis. In: Duvier R, Michelot C (eds) Recent Developments In Optimization. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 269–290
19. Penot JP (1998) Are generalized derivatives useful for generalized convex functions? In: Crouzeix JP, Martinez-Legaz JE, Volle M (eds) Generalized Convexity, Generalized Monotonicity: Recent Developments. Proc. 5th Internat. Symp. Generalized Convexity, Kluwer, Dordrecht, pp 3–60
20. Penot JP, Quang PH (1997) Generalized convexity of functions and generalized monotonicity of set-valued maps. J Optim Th Appl 92:343–356
21. Yao JC (1994) Multivalued variational inequalities with K-pseudomonotone operators. J Optim Th Appl 83:391–403

Generalized Monotone Single Valued Maps

GMSVM

Nicolas Hadjisavvas[1], Siegfried Schaible[2]

[1] Department Math., University Aegean, Karlovassi, Greece

[2] A.G. Anderson Graduate School of Management, University California, Riverside, USA

MSC2000: 90C26

Article Outline

Keywords
Seven Kinds of (Generalized) Monotonicity
The Differentiable Case
The Affine Case
The Nondifferentiable Case
Conclusion
See also
References

Keywords

Generalized convexity; Generalized monotonicity; Quasiconvex function; Quasimonotone map

In the analysis and solution of complementarity problems and variational inequalities, it is commonly assumed that the defining map is *monotone*. This is not surprising since in the special case of an underlying optimization problem usually *convexity* is assumed, and convexity of the objective function corresponds to monotonicity of its gradient.

For several decades much effort has been devoted to generalizing convexity in various ways, often with the view of nonconvex optimization inmind [1]. On the other hand, only recently a systematic study of generalizations of monotonicity has emerged. Since the article [14] in 1990 about two hundred publications have appeared. They deal with either concepts and characterizations of generalized monotonicity or with uses in variational inequalities and related models [23].

In this survey characterizations of generalized monotonicity for different subclasses of maps are presented. The need for such criteria is obvious, given that the defining inequalities are often hard to verify.

The article is organized as follows. The next section provides a brief review of some basic generalized monotonicity concepts and their relationships. This is followed by a presentation of criteria for generalized-monotonicity in case of differentiable, affine and nondifferentiable (locally Lipschitz) maps in the subsequent sections.

This article on concepts and characterizations of generalized monotone maps in the single valued case is complemented by one on multi valuedmaps. In a third article in this volume the use of generalized monotonicity in variational inequalities and more general models is surveyed. For amore detailed survey of applications see [11].

Seven Kinds of (Generalized) Monotonicity

Seven basic kinds of convex/generalized convex functions are [1]:

- convex (cx), strictly convex (str.cx);
- pseudoconvex (pcx), strictly pseudoconvex (str.pcx);
- quasiconvex (qcx), semistrictly quasiconvex (sstr.qcx) and strictly quasiconvex (str.qcx).

Strongly convex and strongly pseudoconvex functions [1] are not considered here.

These functions are related to each other as follows:

$$\begin{array}{ccccc} & & & & qcx \\ & & & & \Uparrow \\ cx & \Rightarrow & pcx & \Rightarrow & sstr.qcx \\ \Uparrow & & \Uparrow & & \Uparrow \\ str.cx & \Rightarrow & str.pcx & \Rightarrow & str.qcx \end{array}$$

For the sake of completeness, the related definitions are presentedbelow.

Consider $f: C \to \mathbf{R}$ where $C \subseteq \mathbf{R}^n$ is convex.

- f is *convex* (*cx*) if for all $x, y \in C$ and $t \in (0, 1)$,

$$f\left(tx + (1-t)y\right) \leq tf(x) + (1-t)f(y); \qquad (1)$$

- f is *strictly convex* (str.cx) if (1) is a strict inequality for $x \neq y$.
- f is *quasiconvex* (qcx) if for all $x, y \in C$ such that $f(x) \leq f(y)$, $t \in (0, 1)$,

$$f\left(tx + (1-t)y\right) \leq f(y); \qquad (2)$$

- f is *strictly quasiconvex* (str.qcx) if (2) is a strict inequality for $x \neq y$;
- f is *semistrictly quasiconvex* (sstr.qcx) if for all $x, y \in C$ such that $f(x) < f(y)$ the inequality (2) is strict.

For the remaining two types of generalized convex functions one assumes differentiability of f on the open convex set $C \subseteq R^n$, although more general definitions are available [1]:

- f is *pseudoconvex* (pcx) if for all $x, y \in C$

$$(y-x)^\top \nabla f(x) \geq 0 \;\Rightarrow\; f(y) \geq f(x); \qquad (3)$$

- f is *strictly pseudoconvex* (str.pcx) if for all $x, y \in C$, $x \neq y$ thesecond inequality in (3) is strict.

Different kinds of generalized convexity preserve different properties of convex functions. E.g., the characteristic of a pseudoconvex function is that a stationary point is a global minimum. Furthermore, for a semistrictly quasiconvex function a local is a global minimum and for a quasiconvex function the lower level sets are convex. The qualifier 'strictly' indicates that a global minimum is unique. In contrast to convex functions, inflection points are admissible for all types of generalized convex functions.

Note that in [1] the terminology of quasiconvex and pseudoconvex functions was harmonized, resulting in renaming former 'strongly quasiconvex' functions as strictly quasiconvex and 'strictly quasiconvex' functions as semistrictly quasiconvex.

It is well known that a differentiable convex function is characterized by a monotone gradient. Correspondingly, a strictly convex function is characterized by a strictly monotone gradient. Accordingly, *generalized monotonicity* concepts have been introduced in such a way that incase of a gradient map $F = \nabla f$ generalized monotonicity of F corresponds to some kind of generalized convexity ofthe underlying function f. The definitions of (generalized) monotone maps are listed below.

Consider $F : C \to \mathbf{R}^n$ where $C \subseteq \mathbf{R}^n$.

- F is *monotone* (m) on C if for all $x, y \in C$

$$(y - x)^\top \big(F(y) - F(x)\big) \geq 0; \tag{4}$$

- F is *strictly monotone* (str.m) on C if for all $x, y \in C$, $x \neq y$

$$(y - x)^\top \big(F(y) - F(x)\big) > 0; \tag{5}$$

- F is *pseudomonotone* (pm) on C if for all $x, y \in C$,

$$(y - x)^\top F(x) \geq 0 \;\Rightarrow\; (y - x)^\top F(y) \geq 0, \tag{6}$$

which is equivalent to

$$(y - x)^\top F(x) > 0 \;\Rightarrow\; (y - x)^\top F(y) > 0;$$

- F is *strictly pseudomonotone* (str.pm) on C if for all $x, y \in C, x \neq y$,

$$(y - x)^\top F(x) \geq 0 \;\Rightarrow\; (y - x)^\top F(y) > 0; \tag{7}$$

- F is *quasimonotone* (qm) if for all $x, y \in C$,

$$(y - x)^\top F(x) > 0 \;\Rightarrow\; (y - x)^\top F(y) \geq 0; \tag{8}$$

- F is *strictly quasimonotone* (str.qm) on C if F is quasimonotone on C and for all $x, y \in C, x \neq y$ there exists $z = tx + (1-t)y$, $t \in (0, 1)$, such that

$$(y - x)^\top F(z) \neq 0; \tag{9}$$

- F is *semistrictly quasimonotone* (sstr.qm) on C if F is quasimonotone on C and for $x, y \in C, x \neq y$,

$$(y - x)^\top F(x) > 0 \;\Rightarrow\; (y - x)^\top F(z) > 0 \tag{10}$$

for some $z = tx + (1 - t)y$, $t \in (0, 1/2)$.

If F is continuous, quasimonotonicity does not have to be required explicitly for strictly/semistrictly quasimonotone maps since it is implied by (9), (10), respectively. In terms of references for the concepts above, see [13] for pseudomonotone maps, [14] for quasimonotone and strictly pseudomonotone maps and [9] for strictly quasimonotone and semistrictly quasimonotone maps.

The following diagram was derived in [9,13,14] for general maps which are not necessarily gradient maps:

$$\begin{array}{ccccc} & & & & qm \\ & & & & \Uparrow \\ m & \Rightarrow & pm & \Rightarrow & sstr.qm \\ \Uparrow & & \Uparrow & & \Uparrow \\ str.m & \Rightarrow & str.pm & \Rightarrow & str.qm \end{array}$$

Now consider the special case of a gradient map $F = \nabla f$, where f is differentiable on the open convex set $C \subseteq \mathbf{R}^n$. In analogy to monotone maps it can be shown [9,13,14]:

Theorem 1 *The map $F = \nabla f$ is quasimonotone (respectively, semistrictly quasimonotone, strictly quasimonotone, pseudomonotone, strictly pseudomonotone) if and only if the function f is quasiconvex (respectively, semistrictly quasiconvex, strictly quasiconvex, pseudoconvex, strictly pseudoconvex).*

Note that in the case of semistrictly quasiconvex functions Theorem 1 provides the first successful characterization in terms of the gradient. Before, the existence of such a characterization was doubted [17].

There are several studies where similar results are obtained for nondifferentiable functions in which the gradient is replaced by the subdifferential (see, e. g., ▶ Generalized monotone multivalued maps).

Given the geometric properties of generalized convex functions mentioned above [1], it is not difficult to derive the geometric properties describing generalized monotonicity of gradient maps; e. g. [2,3,15].

New generalized monotone maps can be constructed from existing ones. As an example from [20], consider $z = Ax + b$, where A is an $m \times n$ matrix and $b \in \mathbf{R}^m$. Let $D \subseteq \mathbf{R}^m$ and $C = \{x \in \mathbf{R}^n : Ax + b \in D\}$. Then the map $F(x) = A^\intercal G(Ax + b)$ is quasimonotone (pseudomonotone) on C if G is quasimonotone (pseudomonotone) on D. Moreover, F is strictly pseudomonotone on C if G is strictly pseudomonotone on D and A hasfull rank.

The Differentiable Case

In this section it is assumed that $F: C \to \mathbf{R}^n$ is differentiable and $C \subseteq \mathbf{R}^n$ is an open convex set. Let $J_F(x)$ be the Jacobian of F. First order characterizations of generalized monotone maps have been established in [15]. In case of gradient maps they extend classical second order characterizations of generalized convex functions.

Let $x \in C$, $v \in \mathbf{R}^n$, $v \neq 0$ and consider the following conditions:

A) $v^\intercal F(x) = 0$ implies $v^\intercal J_F(x) v \geq 0$;

A+) $v^\intercal F(x) = 0$ implies $v^\intercal J_F(x) v > 0$;

B) $v^\intercal F(x) = v^\intercal J_F(x) v = 0$ and the condition $v^\top F(x + \widehat{t}v) > 0$ for some $\widehat{t} < 0$ implies that there exists $\widetilde{t} > 0$ such that $x + \widetilde{t}v \in C$, $v^\top F(x + tv) \geq 0$ for all $0 \leq t \leq \widetilde{t}$;

C) $v^\intercal F(x) = v^\intercal J_F(x)\, v = 0$ implies that there exists $\widetilde{t} > 0$ such that $x + \widetilde{t}v \in C v^\top F(x + tv) \geq 0$ for all $0 \leq t \leq \widetilde{t}$.

The following can be shown:

Theorem 2 *Let $F: C \to \mathbf{R}^n$ be differentiable on the open convex set $C \subseteq \mathbf{R}^n$.*

i) F is quasimonotone if and only if A) and B) hold for all $x \in C$ and $v \in \mathbf{R}^n$;

ii) F is pseudomonotone if and only if A) and C) hold for all $x \in C$ and $v \in \mathbf{R}^n$;

iii) F is strictly pseudomonotone if A+) holds for all $x \in C$ and $v \in \mathbf{R}^n$.

More recently, it was shown in [4] that for continuously differentiable maps $v^\intercal F(x) = 0$ in B) and C) can be replaced by the less restrictive assumption $F(x) = 0$, and i) and ii) are still true. An immediate consequence of this stronger characterization is that for a nonvanishing map on an open convex set there is no difference between quasimonotonicity and pseudomonotonicity. Both are characterized by condition A). However, this is no longer true in closed convex sets (see [10, Example 3.1]).

The Affine Case

In this section we focus on the special case of affine maps. Let $F(x) = Mx + q$ where M is an $n \times n$ matrix and $q \in \mathbf{R}^n$. Consider F on an open convex set $C \subseteq \mathbf{R}^n$. For general differentiable maps we have $F = \nabla f$ if and only if $J_F(x)$ is symmetric for all x. Hence for an affine map $F(x) = Mx + q$ we have $F = \nabla f$ if and only if M is symmetric. In this case $f(x) = (x^\intercal Mx)/2 + q^\intercal x$. Therefore first order characterizations of generalized monotone affine maps correspond to second order characterizations of generalized convex quadratic functions.

For affine maps conditions B) and C) are always satisfied. Hence, specializing Theorem 2 we have

Theorem 3 *The map $F(x) = Mx + q$ is quasimonotone on an open convex set $C \subseteq \mathbf{R}^n$ if and only if F is pseudomonotone on C if and only if for all $x \in C$ and $v \in \mathbf{R}^n$*

$$v^\top (Mx + q) = 0 \;\Rightarrow\; v^\top M v \geq 0.$$

As a result, quasimonotonicity in a neighborhood of a point $\overline{x}$ such that $M\overline{x} + q = 0$ implies monotonicity on $\mathbf{R}^n$.

As mentioned earlier, one can construct new generalized monotone maps with the help of a given one as follows. Given the linear map $G(z) = Mz$, if G is quasimonotone (pseudomonotone) on the nonnegative orthant $\mathbf{R}^m_+$, then the map $F(x) = (A^\intercal MA)x$ is quasimonotone (pseudomonotone) on $\mathbf{R}^n_+$, for any nonnegative $m \times n$ matrix A.

Recently a matrix-theoretic characterization of generalized monotone affine maps was obtained [6]. The departure point for its derivation is Theorem 3. The following notation is needed to describe the results.

For the affine map $F(x) = Mx + q$ one considers

$$B = \frac{1}{2}(M + M^\top), \quad P = \frac{1}{2} M^\top B^\dagger M,$$

where $B^\dagger$ is the Moore–Penrose pseudo-inverse of B, n_+, n_- and n_0 is the number of positive, negative and

zero eigenvalues of B, respectively,

$$r = \dim(\ker(M)),$$
$$f(x) = (Mx+q)^{\top} B^{\dagger}(Mx+q),$$
$$S = \{x \in \mathbf{R}^n : \ f(x) \le 0\},$$
$$T = \left\{x \in \mathbf{R}^n : \ x^{\top} P x \le 0\right\},$$

$C \subseteq \mathbf{R}^n$ is convex with $C \neq \emptyset$.

One has [6]:

Theorem 4 *F is quasimonotone on C (and pseudomonotone on (C)) if and only if one of the following conditions holds:*

i) $n_- = 0$, *i. e., B is positive semidefinite and F is monotone on $\boldsymbol{R}^n$;*

ii1) $n_- = 1$, $r = n_0 + 1$, $-q \notin M$ *(int C)*, $q \in B(\boldsymbol{R}^n) \supseteq M(\boldsymbol{R}^n)$, *$P$ is positive semidefinite, S isa closed convex set and $C \subseteq S$;*

ii2) $n_- = 1$, $r = n_0$, $-q \notin M$*(int C)*, $q \in B(\boldsymbol{R}^n) = M(\boldsymbol{R}^n)$, $T = T_+ \cup (-T_+)$ *where T_+ is a closed convex cone, int $T_+ \neq \emptyset$, and for $\overline{x}$ such that $M\overline{x} = q$ either $C \subseteq -\overline{x} + T_+$ or $C \subseteq -\overline{x} - T_+$.*

Hence the maximal domain of quasimonotonicity is:

- $\mathbf{R}^n$ in case i);
- S in case ii1), and
- $-\overline{x} + T_+$ or $-\overline{x} - T_+$ in case ii2).

From Theorem 4 a characterization of quasimonotone (pseudomonotone)affine maps on convex cones can be derived, and further specialized to the nonnegative orthant [6].

It should be noted that in the special case $M^{\intercal} = M$, case ii1) does not occur and Theorem 4 reduces to classical characterizations of generalized convex quadratic functions [7,18,19,21,22]; see also [1]. Case ii1) does not occur either if M is nonsingular. Hence it arises only if M is not symmetric and singular.

Theorem 4 characterizes pseudomonotone affine maps on open convex sets. However in applications, e. g. in complementarity problems and variational inequalities, pseudomonotonicity on closed and convex sets is needed. Such characterizations have very recently been derived in [5] with an approach different from the one in [6]. It involves an extension of Martos' concept of positive subdefinite matrices [18] to the nonsymmetric case. Among others, [5] generalizes previous results on pseudomonotone matrices for linear complementarity problems, e. g. [8].

The Nondifferentiable Case

Finally, characterizations of certain nondifferentiable generalizedmonotone maps [16] are presented in this section.

Let F: $C \to \mathbf{R}^n$ be locally Lipschitz where $C \subseteq \mathbf{R}^n$ is open convex. The criteria below make use of the generalized Jacobian in the senseof Clarke. Given $x \in C$, let $L(x)$ be the set of all limits $DF(x_i)$ where $x_i \to x$, F is differentiable at $x_i \in C$ and $DF(x_i)$ is the Jacobian. Define $\partial F(x)$ to be the convex hull of $L(x)$. Finally, for $x \in C$ and $v \in \mathbf{R}^n$ set

$$D_+F(x;v) = \sup\left\{v^{\top} Av : \ A \in \partial F(x)\right\},$$
$$D_-F(x;v) = \inf\left\{v^{\top} Av : \ A \in \partial F(x)\right\}.$$

In generalization of Theorem 2i) one has:

Theorem 5 *The locally Lipschitz map F is quasimonotone on C if and only if for all $x \in C$, $v \in \boldsymbol{R}^n$*

A′) $v^{\intercal} F(x) = 0$ *implies* $D_+F(x;v) \ge 0$, *and*

B′) $v^{\intercal} F(x) = 0$, $0 \in \{v^{\intercal} Av$: $A \in \partial F(x)\}$ *and* $v^{\top} F(x + \widehat{t}v) > 0$ *for some* $\widehat{t} < 0$ *imply that there exists* $\widetilde{t} > 0$ *such that* $v^{\intercal} F(x + tv) \ge 0$ *for all* $t \in [0, \widetilde{t}]$.

In light of [4], a stronger sufficient condition can be obtained which however is no longer necessary [16], in contrast to the differentiable case.

Theorem 6 *The map F is quasimonotone on C if for all $x \in C$, $v \in \boldsymbol{R}^n$, $\boldsymbol{v \neq 0}$*

A″) $v^{\intercal} F(x) = 0$ *implies* $D_-F(x;v) \ge 0$, *and*

B″) $F(x) = 0$, $D_-(x;v) = 0$ *and* $v^{\top} F(x + \widehat{t}v) > 0$ *for some* $\widehat{t} < 0$ *imply that there exists* $\widetilde{t} > 0$ *such that* $v^{\intercal} F(x + tv) \ge 0$ *for all* $t \in [0, \widetilde{t}]$.

In analogy to the differentiable case (see Theorem 2), corresponding characterizations can be obtained for pseudomonotone maps, replacing B′), B″) by a stronger condition. Furthermore, criteria for strict pseudomonotonicity are derived in [16].

Very recently, generalized monotonicity criteria for locally Lipschitz maps have been extended to the class of general continuous maps [12]. In this study Clarke's generalized Jacobian is replaced by an 'approximate Jacobian'.

Conclusion

In this survey we have presented various characterizations of generalized monotone maps. Details are

shown mainly for quasimonotone and pseudomonotone maps. In retrospect, it becomes clear how the main characterization in the differentiable case (Theorem 2) specializes in the affine case (Theorems 3, 4) and how it can be extended in the nondifferentiable case (Theorem 5).

See also

- ► Fejér Monotonicity Inconvex Optimization
- ► Generalized Monotone Multivalued Maps
- ► Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ► Set-valued Optimization

References

1. Avriel M, Diewert WE, Schaible S, Zang I (1988) Generalized concavity. Plenum, New York
2. Castagnoli E, Mazzoleni P (1991) Order-preserving functionsand generalized convexity. Rivista di Mat per le Sci Economiche e Sociali 14:33–46
3. Crouzeix JP (1998) Characterizations of generalized convexity and generalized monotonicity, a survey. In: Crouzeix JP, Martinez-Legaz JE, Volle M (eds) Generalized Convexity, Generalized Monotonicity: Recent Developments. Kluwer, Dordrecht, 237–256
4. Crouzeix JP, Ferland JA (1996) Criteria for differentiable generalized monotone maps. Math Program 75:399–406
5. Crouzeix JP, Hassouni A, Lahlou A, Schaible S (1999) Positive subdefinite matrices, generalized monotonicity and linear complementarity problems. SIAM J Matrix Anal Appl
6. Crouzeix JP, Schaible S (1996) Generalized monotone affine maps. SIAM J Matrix Anal Appl 17:992–997
7. Ferland JA (1971) Quasi-convex and pseudo-convex functions on solid convex sets. Dept Oper Res Stanford Univ 71–4
8. Gowda MS (1990) Affine pseudomonotone maps and the linear complementarity problem. SIAM J Matrix Anal Appl 11:373–380
9. Hadjisavvas N, Schaible S (1993) On strong pseudomonotonicity and (semi)strict quasimonotonicity. J Optim Th Appl 79:139–155
10. Hadjisavvas N, Schaible S (1996) Quasimonotone variational inequalities in Banach spaces. J Optim Th Appl 90:95–111
11. Hadjisavvas N, Schaible S (1998) Quasimonotonicity and pseudomonotonicity in variational inequalities and equilibrium problems. In: Crouzeix JP, Martinez-Legaz JE, Volle M (eds) Generalized Convexity, Generalized Monotonicity: Recent Developments. Kluwer, Dordrecht, pp 257–275
12. Jeyakumar V, Luc DT, Schaible S (1998) Characterizations of generalized monotone nonsmooth continuous maps using approximate Jacobians. J Convex Anal 5:119–132
13. Karamardian S (1976) Complementarity over cones with monotone and pseudomonotone maps. J Optim Th Appl 18:445–454
14. Karamardian S, Schaible S (1990) Seven kinds of monotone maps. J Optim Th Appl 66:37–46
15. Karamardian S, Schaible S, Crouzeix JP (1993) Characterizations of generalized monotone maps. J Optim Th Appl 76:399–413
16. Luc DT, Schaible S (1996) Generalized monotone nonsmooth maps. J Convex Anal 3:195–205
17. Mangasarian OL (1969) Nonlinear programming. McGraw-Hill, New York
18. Martos B (1969) Subdefinite matrices and quadratic forms. SIAM J Appl Math 17:1215–1223
19. Martos B (1971) Quadratic programming with a quasiconvex objective function. Oper Res 19:82–97
20. Pini R, Schaible S (1994) Invariance properties of generalized monotonicity. Optim 28:211–222
21. Schaible S (1971) Beiträge zur quasikonvexen Programmierung. PhD Thesis Univ. Köln
22. Schaible S (1981) Quasiconvex, pseudoconvex and strictly pseudoconvex quadratic functions. J Optim Th Appl 35:303–338
23. Schaible S (1996) From generalized convexity to generalized monotonicity. In: Du D-Z, Zhang X-S, Cheng K (eds) Operations Research and its Applications, Proc. Second Internat. Symp. Oper. Res.and its Applications (ISORA) in Guilin, P.R. China, Beijing Word Publ. Corp., pp 134–143

Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems

GMVIPEP

NICOLAS HADJISAVVAS[1], SIEGFRIED SCHAIBLE[2]

[1] Department Math., University Aegean, Karlovassi, Greece

[2] A.G. Anderson Graduate School of Management, University California, Riverside, USA

MSC2000: 90C26, 49J40, 46N10

Article Outline

Keywords

Scalar Variational Inequalities

Vector Variational Inequalities
Equilibrium Problems
See also
References

Keywords

Generalized monotonicity; Variational inequalities; Equilibrium problems

This article on generalized monotone maps focuses on some of their uses in variational inequalities and equilibrium problems. Definitions and properties of various types of generalized monotone maps are found in ► Generalized monotone single valued maps and ► Generalized monotone multivalued maps. These articles form the background of the present survey.

Variational inequalities appear in various forms and arise in a wide range of problems in the natural and social sciences, for example [22]. The simplest variational inequality problem (VIP) is the following: Given a nonempty closed convex subset K of $\mathbf{R}^n$ and a map F: $K \to \mathbf{R}^n$, find an element $x_0 \in K$ such that

$$(F(x_0))^\top (x - x_0) \geq 0 \text{ for all } x \in K. \tag{1}$$

The prime example of a variational inequality stems from a minimization problem. Given a differentiable function f: $K \to \mathbf{R}$, if $x_0 \in K$ minimizes f, then x_0 is a solution of the VIP (1) with $F = \nabla f$.

As shown by G.J. Hartman and G. Stampacchia [17], (1) has a solution if K is compact and F is continuous. This result found many applications and holds also, with the same assumptions, in infinite-dimensional Banach spaces (cf. [26, Prop. 77.8]). However, in infinite-dimensional problems this form of the theorem is not useful. The reason for this is that in almost all interesting applications the assumptions of (strong) compactness of the set K and of continuity of the operator F are too strong to be met. A decisive step forward was made by F. Browder who relaxed both assumptions, at the cost of imposing another assumption, namely monotonicity [7]. Specifically, let X be a real Banach space with dual X^*, and K a nonempty, weakly compact and convex subset of X. Given an operator T: $K \to X^*$, consider the following VIP: find $x_0 \in K$ such that

$$\langle Tx_0, x - x_0 \rangle \geq 0 \quad \text{for all } x \in K, \tag{2}$$

where $\langle \cdot, \cdot \rangle$ is the duality pairing between X^* and X. As shown by Browder, the VIP (2) has a solution if T is hemicontinuous and monotone. We recall that an operator T is called *hemicontinuous* if its restriction to line segments is continuous when X^* is equipped with the w^*-topology. The operator is called *monotone* if for all $x, y \in K$ one has

$$\langle Ty - Tx, y - x \rangle \geq 0.$$

It is interesting to note that in the standard example of a variational inequality problem where $X = \mathbf{R}^n$ and T is the gradient of a function f: $K \to \mathbf{R}$ the operator T is monotone if and only if f is convex. This shows that monotonicity is a natural assumption for VIP. But it also shows that it may be too rigid in many applications. This led to the consideration of variational inequality problems and their extensions with *generalized monotone operators*. The first to consider generalized monotonicity in connection with variational inequalities was H. Brezis [5]. Then S. Karamardian [19], coming from convex and generalized convex optimization [1], began a tradition of introducing concepts of generalized monotonicity which, unlike the one of Brezis, preserve the connection between monotonicity and convexity. They ensure that in case of a gradient map, the gradient is generalized monotone (for instance, pseudomonotone, strictly pseudomonotone, quasimonotone, strictly quasimonotone, semistrictly quasimonotone) if the underlying function is generalized convex (i. e., respectively, pseudoconvex, strictly pseudoconvex, quasiconvex, strictly quasiconvex, semistrictly quasiconvex [1]). For definitions and properties of these concepts see ► Generalized monotone single valued maps and ► Generalized monotone multivalued maps for single- and multivalued generalized monotone maps, respectively.

In the next section, results on the existence of solutions for the variational inequality problem with generalized monotone operators are presented. A generalization of these results to vector valued variational inequality problems is given in the third section. Finally, the last section surveys results on the existence of solutions for equilibrium problems, both in the scalar and in the vector case. To begin, consider the following notation and definitions.

Let X be a real Banach space. Given $x, y \in X$, $]x, y[$ and $[x, y]$ denote the open line segment and the closed

line segment joining x and y, respectively; the segments $]x, y]$, and $[x, y[$ are defined analogously. A multivalued operator $T: K \to 2^{X^*} \setminus \{\emptyset\}$ is called *upper hemicontinuous* if for all $x, y \in K$, the restriction of T to $[x, y]$ is upper semicontinuous with respect to the w^*-topology on X^*.

For any nonempty subset D of X, a point $x_0 \in X$ is called an *inner point* of D [14,25] if for all $u \in X^*$ the following implication holds:

$$\langle x, u\rangle \le \langle x_0, u\rangle\,,\ \forall x \in D$$
$$\Rightarrow \quad \langle x, u\rangle = \langle x_0, u\rangle\,, \quad \forall x \in D.$$

The set of inner points of D is denoted by inn D. The concept of an inner point is a generalization of the concept of a relative algebraic interior point. Indeed, in case X is finite dimensional, the two concepts coincide. In the general case, any relative algebraic interior point is an inner point; in case of a closed convex set, inner points have the following properties [14,25]:

Theorem 1 *Let $K \neq \emptyset$ be a closed and convex subset of X. Then one has:*

i) *inn $K \subseteq K$;*

ii) *if K is separable, then inn $K \neq \emptyset$;*

iii) *if $x_1 \in K$, $x_0 \in$ inn K, then*

$$]x_1, x_0] \subseteq \text{inn } K;$$

in particular, inn K is convex.

There are many important examples of closed convex subsets K which contain inner points, without containing any relative algebraic interior points [14].

Scalar Variational Inequalities

Let X be a real Banach space, and K a nonempty, closed, convex subset of X. Let further $T: K \to 2^{X^*} \setminus \{\emptyset\}$ be a multivalued operator with nonempty values. The VIP for such an operator is the following: find $x_0 \in K$ such that

$$\forall x \in K\ \exists x^* \in Tx_0: \quad \langle x^*, x - x_0\rangle \ge 0. \tag{3}$$

This problem is closely related to the so-called *dual variational inequality problem* (DVIP), which is the following: find $x_0 \in K$ such that

$$\forall x \in K\ \forall x^* \in Tx: \ \langle x^*, x - x_0\rangle \ge 0. \tag{4}$$

Indeed, it is well known that, if x_0 is a solution of DVIP, then it is also a solution of VIP, provided that T is upper hemicontinuous [20]. For this reason, most proofs of existence of a solution for VIP establish first the existence of a solution of DVIP.

R.W. Cottle and J.C. Yao [8] were the first to show an existence result for a solution of a VIP with a single valued pseudomonotone operator, hereby extending Karamardian's result [19] for complementarity problems in finite-dimensional spaces. Later, Yao [24] generalized this result to multi valued pseudomonotone operators; I.V. Konnov [20] generalized it further to include semistrictly quasimonotone operators; see ► Generalized monotone multivalued maps. The most general result in this direction with no assumptions (except coercivity) was derived for properly quasimonotone operators [12]. The operator T is called *properly quasimonotone* if for all $x_1, \ldots, x_n \in K$ and all $y \in \text{co}\{x_1, \ldots, x_n\}$ there exists $i \in \{1, \ldots, n\}$ such that $\langle x^*, y - x_i\rangle \le 0$ for all $x^* \in Tx_i$. The name of this property is justified by the fact that a lower semicontinuous function $f: K \to \mathbf{R}$ is quasiconvex if and only if its Clarke-Rockafellar subdifferential is properly quasimonotone [12]. For such operators, the following theorem holds [11]:

Theorem 2 *Let $T: K \to 2^{X^*} \setminus \{\emptyset\}$ be a properly quasimonotone operator. Suppose that K is weakly compact, or alternatively that the following coercivity condition holds: there exists a weakly compact subset W of K and $x_0 \in W$ such that*

$$\forall x \in K \setminus W\ \exists x_0^* \in Tx_0: \ \left\langle x_0^*, x_0 - x\right\rangle < 0. \tag{5}$$

Then the DVIP (4) has a solution. Consequently, if T is upper hemicontinuous, then the VIP (3) also has a solution.

A semistrictly quasimonotone operator (or, a fortiori, a pseudomonotone operator) is properly quasimonotone [11]. Thus, the above result generalizes the corresponding results in [20] and [24].

For the still more general case of a quasimonotone operator, even for single valued operators, one needs a mild assumption on the domain [14]. For multivalued operators one needs still stronger assumptions [9]:

Theorem 3 *Let $T: K \to 2^{X^*} \setminus \{\emptyset\}$ be a quasimonotone operator. Suppose that:*

a) K is weakly compact, or alternatively that coercivity condition (5) holds;
b) inn $K \neq \emptyset$;
c) T has compact values.
d) T is upper hemicontinuous.
Then the VIP (3) has a solution.

Vector Variational Inequalities

The VIP has been generalized in various ways. One of these generalizations proposed by F. Giannessi [13] suggests to consider the variational inequality in a multidimensional space rather than the real number field. This is the so-called *vector variational inequality problem* (VVIP). The VVIP is closely related, just as its scalar counterpart, to the least element problem and the complementarity problem [23].

In the VVIP, apart from the Banach space X and its closed, convex subset K, one considers a Banach space Y and the space $L(X, Y)$ of all continuous linear operators from X to Y. The space Y is ordered by a cone C. In this case, the expression 'the element $x \in Y$ is nonnegative' can have two different meanings: either $x \in C$ or $x \notin -\text{int } C$. It further increases the applicability, especially to economics, without much additional effort if one allows this cone to 'move'; thus, instead of a cone one considers a multivalued mapping $C\colon K \to 2^Y$ such that for each $x \in K$, $C(x)$ is a closed convex cone with nonempty interior. Let further $T\colon K \to 2^{L(X,Y)} \setminus \{\emptyset\}$ be a multivalued operator. The VVIP is the following: find $x_0 \in K$ such that

$$\forall y \in K\ \exists A \in Tx_0 : \quad A(y - x_0) \notin -\text{int } C(x_0). \tag{6}$$

In the scalar case $Y = \mathbf{R}$, $C(x) = \mathbf{R}^+$ one has $L(X, Y) = X^*$, and VVIP becomes VIP. In the general case, monotonicity and generalized monotonicity have to be newly defined. The operator T is called:

- *monotone* if for all $x, y \in K$ one has:

$$\forall A \in Tx\ \forall B \in Ty: \quad (B - A)(y - x) \in C(x);$$

- *pseudomonotone* if for all $x, y \in K$ the following implication holds:

$$\exists A \in Tx : A(y - x) \notin -\text{int } C(x) \Rightarrow \forall B \in Ty: B(y - x) \notin -\text{int } C(x);$$

- *quasimonotone* if for all $x, y \in K$ the following implication holds:

$$\exists A \in Tx : A(y - x) \notin -C(x) \Rightarrow \forall B \in Ty: B(y - x) \notin -\text{int } C(x).$$

We now recall some topological notions. The *strong operator topology* (SOT) on $L(X, Y)$ is the weakest topology such that for each $x \in X$, the function $L(X, Y) \ni A \to Ax \in Y$ is continuous. An operator $A \in L(X, Y)$ is called *completely continuous* if it maps weakly convergent sequences into strongly convergent sequences. Examples of completely continuous operators are compact operators. The following result proved in [10] generalizes many existence results in the literature as well as Theorem 3:

Theorem 4 *Suppose that the following assumptions hold:*
i) the operator T is upper hemicontinuous with respect to the SOT topology;
ii) the graph of the multifunction

$$x \to Y \setminus (-\text{int } C(x))$$

is sequentially closed in $X \times Y$ in the (weak)$\times$(strong) topology;
iii) K is weakly compact;
iv) for each $x \in K$, Tx consists of completely continuous operators;
v) T is pseudomonotone, or
v') T is quasimonotone, its values are norm compact and inn $K \neq \emptyset$.
Then the VVIP (6) has a solution.

As in the scalar case, the assumption 'K is compact' may be replaced by a coercivity condition.

Equilibrium Problems

The remainder of this article deals with problems more general than VIP. Given a nonempty set K and a bifunction $f\colon K \times K \to \mathbf{R}$, the equilibrium problem (EP) [4,6] for f is the following: find $x_0 \in K$ such that

$$f(x_0, y) \geq 0 \quad \text{for all } y \in K. \tag{7}$$

A great variety of problems can be formulated as an EP including problems of optimization, saddle point theory, game theory, fixed point theory and VIP [4]. For

instance, if K is a nonempty closed, convex subset of a Banach space X and $T: K \to 2^{X^*} \setminus \{\emptyset\}$ is a multivalued operator with weakly compact values, let f be defined as

$$f(x, y) = \max\{\langle x^*, y - x\rangle : x^* \in Tx\}. \tag{8}$$

It is easy to see that $x_0 \in K$ is a solution of the EP (7) if and only if it is a solution of the VIP (2). Because of this correspondence, one is led to define concepts of generalized monotonicity for bifunctions. A bifunction f is called:

- *monotone* [4] if for all $x, y \in K$ one has:

$$f(x, y) + f(y, x) \leq 0;$$

- *pseudomonotone* [3] if for all $x, y \in K$ the following implication holds:

$$f(x, y) \geq 0 \Rightarrow f(y, x) \leq 0;$$

- *quasimonotone* [3] if for all $x, y \in K$ the following implication holds:

$$f(x, y) > 0 \Rightarrow f(y, x) \leq 0.$$

It is easy to see that a multi valued operator is monotone (respectively, pseudomonotone, quasimonotone) if and only if the bifunction defined by relation (8) is monotone (respectively, pseudomonotone, quasimonotone). Equilibrium problems with generalized monotone bifunctions in the above sense were considered in [3]. There the following result was proved:

Theorem 5 *Let X be a real topological Hausdorff vector space and $K \subseteq X$ be nonempty, convex and closed. Let further $f: K \times K \to \mathbf{R}$ be a bifunction such that $f(x, x) \geq 0$ for all $x \in K$. Consider the following assumptions:*

i) $f(\cdot, y)$ is hemicontinuous (i. e., continuous on every line segment in K) for all $y \in K$;

ii) $f(x, \cdot)$ is semistrictly quasiconvex [1] and lower semicontinuous for all $x \in K$;

iii) there exists a compact subset $B \subseteq X$ and $y_0 \in B \cap K$ such that $f(x, y_0) < 0$ for all $x \in K \setminus B$ (coercivity);

iv) for all $x \in K$, if $f(x, y) = 0$ and $f(x, y_1) > 0$, then $f(x, z) > 0$ for all $z \in]y, y_1[$;

v) the algebraic interior of K is nonempty.

If f is pseudomonotone and assumptions (i–iii) hold, then the EP (7) has a solution. Likewise, if f is quasimonotone and all assumptions i)–v) hold, then (7) has a solution.

The above theorem generalizes older results by Brezis, L. Nirenberg and Stampacchia [6] and is related to more recent results with monotone bifunctions by E. Blum and W. Oettli [4].

Just like vector variational inequalities, vector equilibrium problems have also been considered where the bifunction takes values in a locally convex vector space ordered by a cone [2]. As shown by Oettli [21] for the pseudomonotone case, vector equilibrium problems can also be treated by considering two real valued bifunctions, rather than one vector valued one. Oettli's approach can even be applied to the quasimonotone case [15]. For this, let X be a real Hausdorff topological vector space, $K \subseteq X$ be nonempty and convex, and f, g: $K \times K \to \mathbf{R}$ be two bifunctions. The bifunction f is said to be *pseudomonotone* with respect to the bifunction g [21] if for all $(x, y) \in K \times K$ the following implication holds:

$$f(x, y) \geq 0 \Rightarrow g(y, x) \leq 0.$$

The bifunctions f, g are said to be a *quasimonotone pair* [15] if for all $(x, y) \in K \times K$ the following implication holds:

$$f(x, y) > 0 \Rightarrow g(y, x) \leq 0.$$

If $f = g$, then the above definitions reduce to those of pseudomonotone and quasimonotone bifunctions, respectively.

The following rather technical, but very useful result was proved in [21] for the pseudomonotone case and in [15] for the quasimonotone case:

Theorem 6 *Consider the following assumptions:*

i) $f(x, x) \geq 0$ for all $x \in K$;

ii) the set $\sigma(y) = \{x \in K: g(y, x) \leq 0\}$ is closed in K for all $y \in K$;

iii) for all $x, y, z \in K$, if $f(x, y) < 0$ and $f(x, z) \leq 0$, then $f(x, u) < 0$ for all $u \in]y, z[$;

iv) there exist a compact subset D of K and $y^* \in D$ such that for all $x \in K \setminus D$ one has $f(x, y^*) < 0$;

v) the set $\{u \in [x, z]: g(u, y) \leq 0\}$ is closed for all $x, z \in K$;

vi) the relative algebraic interior of K is nonempty.

Suppose that f is pseudomonotone with respect to g and assumptions i)–iv) hold, or that the bifunctions f, g are a quasimonotone pair and all assumptions i)–vi) hold.

Then at least one of the following problems has a solution $x_0 \in K$*:*

$$f(x_0, y) \geq 0 \quad \text{for all } y \in K,$$
$$g(y, x_0) \leq 0 \quad \text{for all } y \in K.$$

By choosing the bifunctions f and g appropriately, a variety of results can be produced. For instance, let X and K be as before, and let Z be a real Hausdorff locally convex space. Finally, let $C \subseteq Z$ be a proper, convex, closed cone with nonempty interior int C. Define the relations $\leq, <, \not\leq$ and $\not<$ on Z by

$$x \leq y \quad \Leftrightarrow \quad y - x \in C;$$
$$x < y \quad \Leftrightarrow \quad y - x \in \text{int } C;$$
$$x \not\leq y \quad \Leftrightarrow \quad y - x \notin C;$$
$$x \not< y \quad \Leftrightarrow \quad y - x \notin \text{int } C.$$

Given a bifunction $H: K \times K \to Z$, consider the vector equilibrium problem (VEP): find $x_0 \in K$ such that

$$H(x_0, y) \not< 0 \quad \text{for all } y \in K. \tag{9}$$

Theorem 6 can now be applied to show the existence of a solution for VEP. This is done as follows. Since the cone C has a nonempty interior by assumption, the dual cone C^* has a w^*-compact base B. (Recall that a (closed) base B of a cone W is a convex subset of W such that $0 \notin B$ and $W = \cup_{t \geq 0} tB$.) Define the real valued bifunctions f and g on K as follows:

$$f(x, y) = \max_{\phi \in B} \phi \left(H(x, y)\right),$$
$$g(x, y) = \min_{\phi \in B} \phi \left(H(x, y)\right).$$

Applying Theorem 6 to these bifunctions, one arrives at the following result [15]:

Theorem 7 *Suppose that the bifunction H satisfies the following assumptions, for all x, y, z in K:*

i) $H(x, x) \not< 0$*;*
ii) the set $\{x \in K: H(y, x) \not> 0\}$ *is closed in K;*
iii) if $H(x, y) < 0$ *and* $H(x, z) \leq 0$*, then* $H(x, u) < 0$ *for all* $u \in]y, z[$*;*
iv) the sets $\{u \in]x, z[: H(u, y) \not> 0\}$ *and* $\{u \in]x, z[: H(u, y) \not< 0\}$ *are closed;*
v) there exist a compact subset D of K and $y^* \in D$ *such that for all* $x \in K \backslash D$ *we have* $H(x, y^*) < 0$ *(coercivity);*
vi) $H(x, y) > 0 \Rightarrow H(y, x) \leq 0$ *(quasimonotonicity of H);*
vii) if $H(u, y) < 0$ *for some* $u \in]x, y[$*, then* $H(u, x) > 0$*.*

Then the VEP (9) has a solution.

The above result considerably strengthens a corresponding result in [2].

As another example for using Theorem 6, consider the Banach spaces X, Y, the multivalued operator T and the cone-valued map C as in the previous section on VVIP. For each $x \in K$, choose a w^*-compact base $B(x)$ of the dual cone $C^*(x)$. Now define the bifunctions f and g as follows:

$$f(x, y) = \max_{\substack{\phi \in B(x) \\ A \in Tx}} \phi \left(A(y - x)\right),$$
$$g(x, y) = \min_{\substack{\phi \in B(y) \\ A \in Tx}} \phi \left(A(y - x)\right).$$

Then, applying Theorem 6, one can show Theorem 4 as a corollary, for a set K with nonempty relative algebraic interior. Other variants of Theorem 4 can also be deduced [15].

In conclusion, this article demonstrates that generalized monotonicity rather than monotonicity is sufficient to establish the existence of solutions for VIP, VVIP, EP and VEP. A more extensive survey can be found in [16].

Finally, the reader interested in recent results on the relevance of generalized monotone VIP for the general economic equilibrium is referred to [18].

See also

- ▶ Equilibrium Networks
- ▶ Fejér Monotonicity in Convex Optimization
- ▶ Financial Equilibrium
- ▶ Generalized Monotone Multivalued Maps
- ▶ Generalized Monotone Single Valued Maps
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Oligopolistic Market Equilibrium
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions

- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Spatial Price Equilibrium
- ▶ Traffic Network Equilibrium
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles
- ▶ Walrasian Price Equilibrium

References

1. Avriel M, Diewert WE, Schaible S, Zang I (1988) Generalized concavity. Plenum, New York
2. Bianchi M, Hadjisavvas N, Schaible S (1997) Vector equilibrium problems with generalized monotone bifunctions. J Optim Th Appl 92:527–542
3. Bianchi M, Schaible S (1996) Generalized monotone bifunctions and equilibrium problems. J Optim Th Appl 90:31–43
4. Blum E, Oettli W (1994) From optimization and variational inequalities to equilibrium problems. Math Student 63:123–145
5. Brezis H (1968) Equations et inéquations non linéaires dans les espaces vectoriels en dualité. Ann Inst Fourier 18:115–175
6. Brezis H, Nirenberg L, Stampacchia G (1972) A remark on Ky Fan's minimax principle. Boll Unione Mat Ital 6:293–300
7. Browder F (1966) Existence and approximations of solution of nonlinear variational inequations. Proc Nat Acad Sci USA 56:419–425
8. Cottle RW, Yao JC (1992) Pseudomonotone complementarity problems in Hilbert space. J Optim Th Appl 75:281–295
9. Daniilidis A, Hadjisavvas N (1995) Variational inequalities with quasimonotone multivalued operators. Preprint
10. Daniilidis A, Hadjisavvas N (1996) Existence theorems for vector variational inequalities. Bull Austral Math Soc 54:473–481
11. Daniilidis A, Hadjisavvas N (1999) Characterization of nonsmooth semistrictly quasiconvex and strictly quasiconvex functions. J Optim Th Appl 102:525–536
12. Daniilidis A, Hadjisavvas N (1999) On the subdifferentials of quasiconvex and pseudoconvex functions and cyclic monotonicity. J Math Anal Appl 237:30–42
13. Giannessi F (1980) Theorems of the alternative: quadratic programs and complementarity problems. In: Cottle RW, Giannessi F, Lions JL (eds) Variational inequalities and complementarity problems. Wiley, New York, pp 151–186
14. Hadjisavvas N, Schaible S (1996) Quasimonotone variational inequalities in Banach spaces. J Optim Th Appl 90:95–111
15. Hadjisavvas N, Schaible S (1998) From scalar to vector equilibrium problems in the quasimonotone case. J Optim Th Appl 96:297–309
16. Hadjisavvas N, Schaible S (1998) Quasimonotonicity and pseudomonotonicity in variational inequalities and equilibrium problems. In: Crouzeix JP, Martinez-Legaz JE, Volle M (eds) Generalized Convexity, Generalized Monotonicity: Recent Developments. Kluwer, Dordrecht, pp 257–275
17. Hartman GJ, Stampacchia G (1966) On some nonlinear elliptic differential functional equations. Acta Math 115:271–310
18. John R (1998) Variational inequalities and pseudomonotone functions: Some characterizations. In: Crouzeix JP, Martinez-Legaz JE, Volle M (eds) Generalized Convexity, Generalized Monotonicity: Recent Developments. Kluwer, Dordrecht, pp 291–301
19. Karamardian S (1976) Complementarity over cones with monotone and pseudomonotone maps. J Optim Th Appl 18:445–454
20. Konnov IV (1998) On quasimonotone variational inequalities. J Optim Th Appl 99:165–181
21. Oettli W (1997) A remark on vector-valued equilibria and generalized monotonicity. Acta Math Vietnam 22:213–221
22. Patriksson M (1999) Nonlinear programming and variational inequality problems. Kluwer, Dordrecht

23. Schaible S, Yao JC (1995) On the equivalence of nonlinear complementarity problems and least element problems. Math Program 70:191–200
24. Yao JC (1994) Multi-valued variational inequalities with K-pseudomonotone operators. J Optim Th Appl 83:391–403
25. Zarantonello EH (1971) Projections on convex sets in Hilbert space and spectral theory. In: Zarantonello EH (ed) Contributions to Nonlinear Functional Analysis. Acad. Press, New York, pp 237–424
26. Zeidler E (1988) Nonlinear functional analysis and its applications, vol IV. Springer, Berlin

Generalized Networks

GN

JEFFERY L. KENNINGTON, KAREN R. LEWIS
Southern Methodist University, Dallas, USA

MSC2000: 90C35

Article Outline

Keywords
See also
References

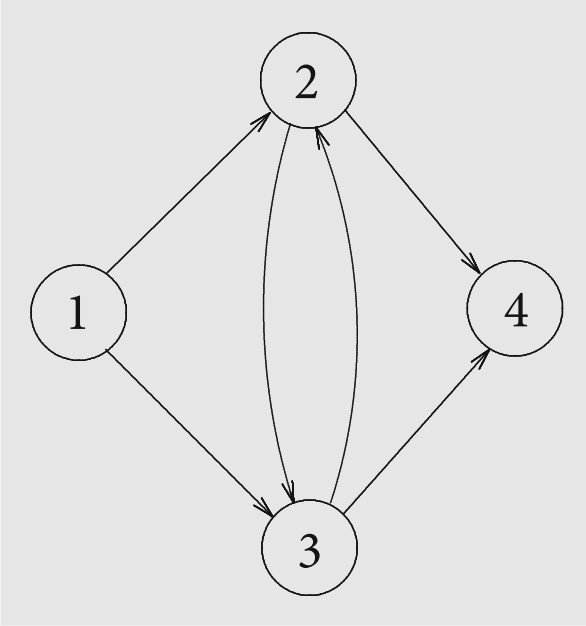
Keywords

Network flows; Generalized networks; Flows with gains

A *network* is composed of two types of entities: *arcs* and *nodes*. The nodes represent locations or terminals, and the arcs represent one-way links connecting pairs of nodes. The arc (i, j) links node i to node j and the flow is from i to j. The structure of a network can be displayed by a drawing, as illustrated in Fig. 1. The structure of a network may also be represented by a *node-arc incidence matrix* A, where A_{ik} is 1 if arc k is directed away from node i, A_{ik} is -1 if arc k is directed toward node i, and A_{ik} is 0 otherwise. Any matrix A in which each column has exactly two nonzero entries, $a + 1$ and $a - 1$, is called a node-arc incidence matrix. The *minimum cost network flow problem* is a linear program, say

$$\min_x \{c'x : \; Ax = b, l \le x \le u\},$$

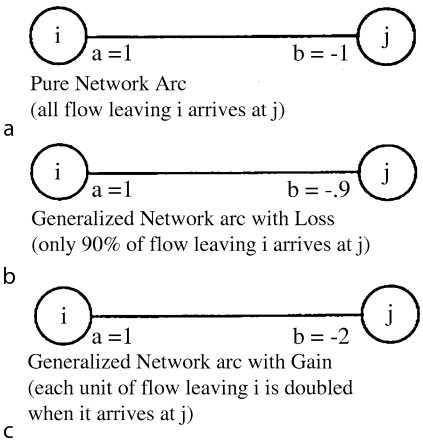
where A is a node-arc incidence matrix. The *generalized network problem*, as its name implies, is a generalization of the minimum cost network flow problem, also referred to as the *pure network problem*.



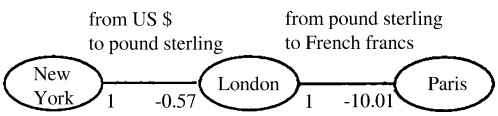
Generalized Networks, Figure 1
Example network with nodes 1, 2, 3, 4 and arcs (1, 2), (1, 3), (2, 3), (2, 4), (3, 2), (3, 4)

Let f denote the flow in arc (i, j) in a pure network. A characteristic of this model is that the f units which depart node i must arrive at node j. Many real applications violate this assumption. In a pipeline distribution network, liquid or gas will be lost due to leakage. In a network carrying a perishable commodity, a certain percentage of the commodity will be lost as it moves along the arcs. For these cases, flow may be lost as it traverses certain arcs. However, if an arc represents holding money in a savings account over a period of time, the value at the end of the period will equal the initial investment plus the interest earned. An arc in a generalized network permits flow to increase, decrease, or remain the same as it traverses the arc. This is illustrated in Fig. 2 for the arc (i, j). Each end of the arc has a constant (*multiplier*) associated with it, which determines the gain or loss during traversal. For the pure network arc, the +1 and -1 correspond to the coefficients in the node-arc incidence matrix.

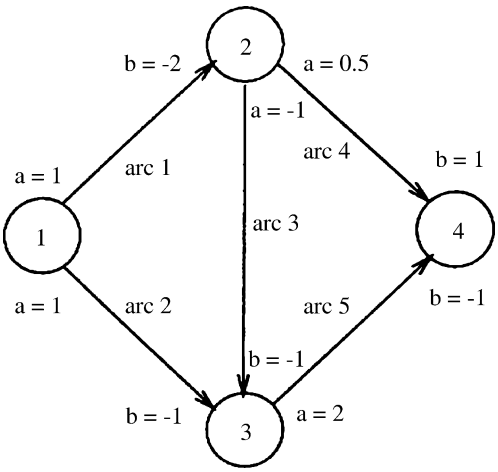
Generalized network models are also used to change units in a flow model. The arcs illustrated in Fig. 3 con-



Generalized Networks, Figure 2
Different types of generalized network arcs



Generalized Networks, Figure 3
Generalized network arcs to convert currency



Generalized Networks, Figure 4
Sample generalized network

vert from US dollars to pound sterling, and from pound sterling to French francs. That is, dollars which depart New York are converted to pounds when they arrive at London. Pounds leaving London are converted to francs when they arrive at Paris. This is also useful to convert from machine-hours to units of finished parts or pallets to truck loads.

In its most general form, the generalized network problem is a linear program with the special feature that each column of the constraint matrix has at most two nonzero entries. Let G be an $m \times n$ matrix with full row rank having this feature. Let c, l, and u be n-component vectors, and r an m-component vector. Let $Y = \{x: Gx = r, l \leq x \leq u\}$, and assume that $Y \neq \emptyset$. The generalized network problem is to find an n-component vector $\tilde{x}$, such that $c\tilde{x} = \min_x \{cx: Gx = r, l \leq x \leq u\}$. For the generalized network model illustrated in Fig. 4, G is

nodes\arcs	1	2	3	4	5
1	1	1	0	0	0
2	−2	0	−1	0.5	0
3	0	−1	−1	0	2
4	0	0	0	1	−1

For each arc, an arbitrary orientation has been assigned so that an arc is defined by the following tuple: (from node, to node, from-node multiplier, to-node multiplier, cost, lower bound, upper bound).

Some authors and computer codes require that the from-node multiplier be 1. The above model can be converted to this form via the variable substitution $\bar{x}_k = a_k x_k$ for $k = 1, \ldots, n$, where a_k is the from-node multiplier for arc k. However, this restriction causes some difficulty if the generalized network solver is ever adapted to solve the integer generalized network model. The code developed by J.L. Kennington and R.A. Mohamed [8] (RAMSES) allows for arbitrary multipliers on both ends of each arc. Other authors assume that the lower bounds are all zero. The above model can be converted to this form via the variable substitution $\bar{x}_k = x_k - l_k$ for $k = 1, \ldots, n$, where l_k is the lower bound for arc k.

Many of the computer codes that have been developed for the generalized network problem are specializations of the *primal simplex algorithm*. These specializations exploit the graphical structure of the basis and solve systems of equations on a graph rather than with standard matrix operations. Let B be a nonsingular $m \times m$ submatrix of G, and N be the submatrix composed of the remaining $n - m$ columns of G. By imposing similar partitions on c, x, and u, the generalized network

problem is represented as

$$\begin{cases} \min & c^B x^B + c^N x^N \\ \text{s.t.} & Bx^B + Nx^N = r, \\ & l^B \le x^B \le u^B, \\ & l^N \le x^N \le u^N. \end{cases}$$

Any solution (x^B, x^N) in which $x_i^N \in \{l_i^N, u_i^N\}$ and $x^B = B^{-1}(r - Nx^N)$ is called a *basic solution* with respect to the basis B. A feasible solution that is also basic is called a *basic feasible solution* BFS. Each iteration of the primal simplex algorithm corresponds to moving from one BFS to another BFS so that the objective function value never increases, proceeding until an optimum is reached.

The *dual variables* associated with a BFS are given by $\pi = c^B B^{-1}$ and the *reduced costs* are given by $\lambda = c - \pi G$. The optimality conditions for a given primal-dual pair are

$$\begin{cases} \lambda_j > 0 & \Rightarrow x_j = l_j, \\ \lambda_j = 0 & \Rightarrow l_j \le x_j \le u_j, \\ \lambda_j < 0 & \Rightarrow x_j = u_j, \end{cases}$$

for each j. Using this notation, an iteration of the primal simplex algorithm is as in the table above.

By re-ordering the rows and columns of B, it can be displayed in block diagonal form as follows:

$$\overline{B} = \begin{pmatrix} \overline{B}^1 & & \\ & \ddots & \\ & & \overline{B}^p \end{pmatrix}.$$

For example, the basis

$$B = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

PROCEDURE primal simplex iteration

1 Let π be a solution to $\pi B = c^B$

2 Set $\lambda^N \leftarrow c^N - \pi N$

3 Select q such that
$\lambda_q^N < 0$ and $x_q^N = l_q^N$, or
$\lambda_q^N > 0$ and $x_q^N = u_q^N$

4a IF no such q exists,
THEN the solution is an optimum.

4b IF $x_q^N = l_q^N$,
THEN $\Delta \leftarrow 1$
ELSE $\Delta \leftarrow -1$

5 Let y be a solution to $By = N_{\cdot q}$, where $N_{\cdot q}$ is the qth column of N

6 Set

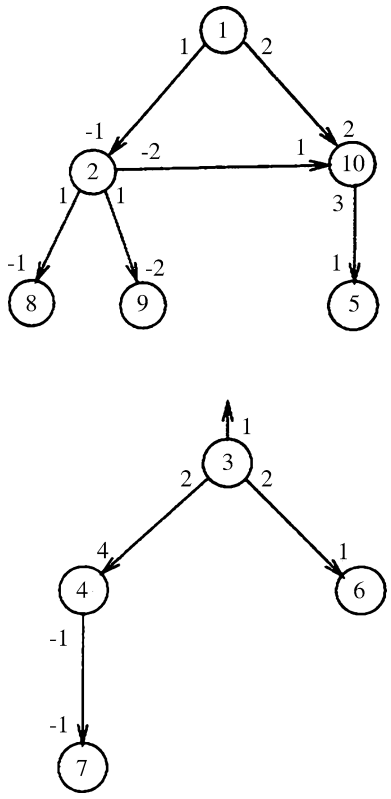
$$d_i \leftarrow \begin{cases} \left|\dfrac{(x_i^B - l_i^B)}{y_i}\right| & \text{for } \Delta y_i > 0, \\ \left|\dfrac{(x_i^B - u_i^B)}{y_i}\right| & \text{for } \Delta y_i < 0, \\ \infty & \text{otherwise} \end{cases}$$

7 Let $s = \operatorname{argmin}\{d_i : i = 1, \ldots, n - m\}$.

8 IF $d_s > u_q^N - l_q^N$
THEN DO Case 1
ELSE DO Case 2.
Case 1.
$x^B \leftarrow x^B - \Delta(u_q^N - l_q^N)y$
IF $\Delta = 1$,
THEN DO $x_q^N \leftarrow u_q^N$
ELSE DO $x_q^N \leftarrow l_q^N$
Case 2.
$x^B \leftarrow x^B - \Delta d_s y$
$x_q^N \leftarrow x_q^N + \Delta d_s$.
Interchange:
the qth column of N and
the sth column of B.

can be displayed as $\overline{B}$ equals

$$\left(\begin{array}{cccccccccc} 1 & 2 & & & & & & & & \\ -1 & & 1 & 1 & -2 & & & & & \\ & 2 & & & & 1 & 3 & & & \\ & & -1 & & & & & & & \\ & & & -2 & & & & & & \\ & & & & & & & 1 & & \\ \hline & & & & & & & & 1 & \\ & & & & & & & & & -1 \\ & & & & & & & & -1 & 4 \\ & & & & & & & & 2 & \end{array}\right)$$



Generalized Networks, Figure 5
A display of the basis *B*

with $p = 2$ and row order 1, 2, 10, 8, 9, 5, 6, 7, 4, and 3. A display of the graph corresponding to B is illustrated in Fig. 5. The direction of the arcs was selected arbitrarily.

A connected network having exactly one cycle (such as the upper component in Fig. 5) is called a *one-tree*. An arc which is incident to a single node (such as the arc corresponding to the last column of $\overline{B}$) is called a *root arc*. A connected network on k nodes having $k - 1$ regular arcs and one root arc is called a *rooted tree* (such as the lower component in Fig. 5). It has been known from at least the 1960s that the connected components of a generalized network are either one-trees or rooted trees ([5,7]). This structure can be exploited in solving the systems $\pi B = c^B$ and $By = N{\cdot}q$ needed in the simplex algorithm, the details of which appear in [6].

In software implementations of the primal simplex algorithm, the basis of a generalized network is maintained using a special data structure. Using the rooted

Generalized Networks, Table 1
Label for the basis illustrated in Fig. 5

Node	Pred	Thrd	Card	Last Node
1	10	2	1	1
2	1	8	3	9
3	3	4	4	6
4	3	7	2	7
5	10	1	1	5
6	3	3	1	6
7	4	6	1	7
8	2	9	1	8
9	2	10	1	9
10	2	5	2	5

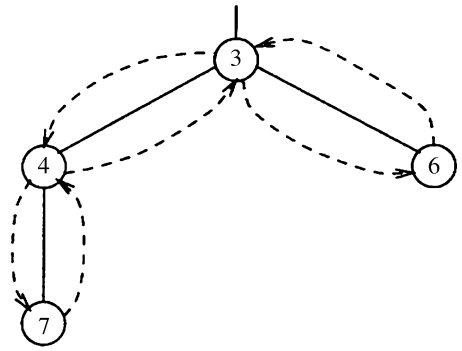
tree illustrated in Fig. 5, one may imagine a line around the contours of the tree as illustrated in Fig. 6a, which is known as a *depth-first search*. For this example, the nodes in this search are ordered 3, 4, 7, 4, 3, 6, 3. An order called *pre-order* is obtained by eliminating all duplicate occurrences (i. e. 3, 4, 7, 6). The label which gives the next node in the pre-order is called the *thread*.

Three additional labels are generally used to maintain the basis. The *predecessor* of node v, denoted $p(v)$ is the first node encountered on the path from v to the root. For root nodes, we adopt the convention $p(v) = v$. If the arc linking v and $p(v)$ were deleted, then there would be two trees, one containing v and the other excluding v. The tree containing v is said to be rooted at v. The *cardinality* of v is defined to be the number of nodes in the tree rooted at v. The *last node* of v is defined to be the last node in the tree rooted at v when the nodes are taken in thread (pre-order) order.

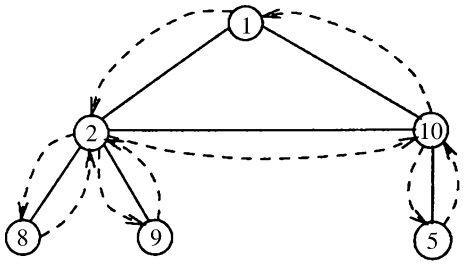
The data structure used to represent a rooted tree is extended for the one-tree in an obvious way. The cycle in the one-tree plays the role of the root node. The predecessor label of the nodes in the cycle point to the next node in the cycle. That is, beginning with any node in the cycle, say v, the sequence v, $p(v)$, $p(p(v))$, ... identifies all nodes in the cycle. The convention adopted for the thread is that traversal around the cycle using the thread is in the opposite direction to that using the predecessor. The pre-order for a one-tree is illustrated in Fig. 6b and the four labels for the basis illustrated in Fig. 5 are given in Table 1. Using these

Generalized Networks, Table 2
Survey of generalized network codes, where A stands for Assembly and F for FORTRAN

Code	Lang	Authors
NETG 1973	F	F. Glover, D. Klingman, J. Stutz
- 1973	F	W. Langley
- 1981	F	D. Adolphson, L. Heum
GENNET 1984	F	G. Brown, R. McBride
GWHIZNET 1984	A	J. Tomlin
GRNET 1985	F	M. Engquist M. Chang
LPNETG 1985	F	J. Mulvey, S. Zenios
- 1986	F	I. Ali, A. Charles, T. Song
GRNET-K (parallel) 1987	F	M. Chang, M. Engquist, M. Finkel, R. Meyer
PGRNET (parallel) 1987	F	R. Clark, R. Meyer, M. Chang
GNO/PC 1988	C	W. Nulty, M. Trick
GRNET-A 1988	A	M. Chang, M. Cheng, C. Chen
GENFLO 1989	F	M. Ramamurti
GRNET2 (serial) 1989	F	R. Clark, R. Meyer, M. Chang
TPGRNET (parallel) 1989	F	R. Clark, R. Meyer
NETPD 1994	F	N. Curet
RAMSES 1997	C	J. Kennington, R. Mohamed



A Depth-First Search for a Rooted Tree
a



A Depth-First Search for a One-Tree
b

Generalized Networks, Figure 6
Depth-first search illustrated

labels and the ideas presented in [2], all operations of the primal simplex algorithm can be performed directly on the *basis forest* composed of rooted trees and one-trees.

Since the generalized network problem is a specially structured linear program, any of the LP algorithms can be used to solve the network problem. By utilizing the structure of the network, however, any of the LP algorithms can be specialized to reduce the solution time. A specialization of the *dual simplex algorithm* may be found in [8] and a primal-dual procedure may be found in [4]. The relaxation method of Bertsekas has also been extended for the generalized network case (see [3]). The interior point algorithm (see [9]) could also be specialized for this problem.

The first specialized software for the generalized network problem was developed in the early 1970's. A partial list of codes which have been developed may be found in Table 2. An extensive list of applications of the generalized network model may be found in [1] and in [10].

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: theory, algorithms and applications. Prentice-Hall, Englewood Cliffs, NJ
2. Barr R, Glover F, Klingman D (1979) Enhancement of spanning tree labeling procedures for network optimization. INFOR 17:16–34
3. Bertsekas D, Tseng P (1988) Relaxation methods for minimum cost ordinary and generalized network flow problems. Oper Res 36(1):93–114
4. Curet ND (1994) An incremental primal-dual method for generalized networks. Comput Oper Res 21(10):1051–1059
5. Dantzig GB (1963) Linear programming and extensions. Princeton Univ. Press, Princeton
6. Helgason RV, Kennington JL (1995) Primal simplex algorithms for minimum cost network flows. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Handbook Oper. Res. and Management Sci. vol 7, Elsevier, Amsterdam
7. Johnson E (1966) Networks and basic solutions. Oper Res 14:619–623
8. Kennington JL, Mohamed RA (1997) An efficient dual simplex optimizer for generalized networks. In: Barr R, Helgason R, Kennington J (eds) Interfaces in Computer Sci. and Oper. Res.: Adv. in Metaheuristics, Optimization, and Stochastic Modeling Techniques. Kluwer, Dordrecht
9. Lustig IJ, Martsten RE, Shanno DF (1994) Interior point methods for linear programming: computational state of the art. ORSA J Comput 6(1):1–14
10. Rardin RL (1998) Optimization in operations research. Prentice-Hall, Englewood Cliffs, NJ

Generalized Nonlinear Complementarity Problem

MICHAEL M. KOSTREVA
Department Math. Sci., Clemson University, Clemson, USA

MSC2000: 90C33

Article Outline

Keywords
See also
References

Keywords

Nonlinear complementarity problem; Generalized complementarity

The *complementarity problem* and its generalizations are now established as an important class of applied mathematical problems. For these problems, there exists a body of theoretical results, algorithms for computing solutions and many applications from engineering to economics and from theoretical physics to computer science. A recent survey, [6], describes some of this progress, including applications in some major industrial research laboratories in the United States. Covered there are models for elasto-hydrodynamic lubrication of bearings (automotive industry) and spatial price equilibrium (telecommunications firm). The application of complementarity allowed engineers and analysts to comprehend and solve a new range of problems which had been out of reach. It is now well documented that other approaches do not adequately model these application problems while complementarity handles them.

Two main *generalizations of the nonlinear complementarity problem* were made:

- Generalization of the ordering to that of a cone (see [3]).
- Generalization to several functions per index (see [1], and [7]).

The first of these generalizations was applied to solve an elasto-hydrodynamic lubrication problem in [5], while the second was applied in [7] to solve a more complex mixed lubrication problem. These particular problems were studied in the past without complementarity models, but it is now recognized that the earlier attempts were incomplete, and failed to comprehend the main features of the physical situation.

In recent years, the second *generalized complementarity problem* above has been reconsidered and a related problem, the generalized order complementarity problem has been studied. It was known for some time that under certain conditions on the functions involved, there exists a solution to the linear generalized complementarity problem. See [1]. Recently, more extensive results have been obtained. For example, B.P. Szanc [8] developed a theory and algorithms for nonlinear functions of the class P, thereby extending the work of G.J. Habetler and M.M. Kostreva [2]. Results for the *infinite-dimensional version* of the generalized order complementarity problem are presented in [4].

The *nonlinear complementarity problem* is as follows: Given $f\colon \mathbf{R}^n \to \mathbf{R}^n$, find $x \in \mathbf{R}^n$ satisfying $x \geq 0$, $f(x) \geq 0$, and $x^{\mathsf{T}} f(x) = 0$. The most general set of conditions known for existence and uniqueness of solutions for this problem (even removing the requirement for continuity of f) are in [2].

Considering the generalized complementarity with cone ordering, let K be a pointed, solid cone in $\mathbf{R}^n$ and let

$$K^* = \left\{ y \in \mathbf{R}^n \colon\ x^{\top} y \geq 0 \text{ for all } x \in K \right\},$$

and let $f\colon \mathbf{R}^n \to \mathbf{R}^n$. The *generalized complementarity problem* (f, K) is to find $x \in \mathbf{R}^n$ satisfying $x \in K$, $f(x) \in K^*$, and $x^{\mathsf{T}} f(x) = 0$.

Finally, the generalization with multiple functions per index is as follows: $f_{ij}\colon \mathbf{R}^n \to \mathbf{R}$, find $x \in \mathbf{R}^n$ satisfying $x_j \geq 0, f_{ij}(x) \geq 0$, $i \in I_j$, $j = 1, \ldots, n$, $x_j \cdot \prod f_j^i(x) = 0$, $i \in I_j$, $j = 1, \ldots, n$. Here the product of the variable x_j with the product of functions ($|I_j|$ of them), plays the role of the complementarity condition.

See also

- ▶ Convex-simplex Algorithm
- ▶ Equivalence Between Nonlinear Complementarity Problem and Fixed Point Problem
- ▶ Integer Linear Complementary Problem
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Lemke Method
- ▶ Linear Complementarity Problem
- ▶ Linear Programming
- ▶ Order Complementarity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Principal Pivoting Methods for Linear Complementarity Problems
- ▶ Sequential Simplex Method
- ▶ Topological Methods in Complementarity Theory

References

1. Cottle RW, Dantzig GB (1970) A generalization of the linear complementarity problem. J Combin Th 8:79–90
2. Habetler GJ, Kostreva MM (1978) On a direct algorithm for nonlinear complementarity problems. SIAM J Control Optim 16:504–511
3. Habetler GJ, Price AL (1971) Existence theory for generalized nonlinear complementarity problems. J Optim Th Appl 7:223–239
4. Isac G, Kostreva M (1991) The generalized order complementarity problem. J Optim Th Appl 71:517–534
5. Kostreva MM (1984) Elasto-hydrodynamic lubrication: A non-linear complementarity problem. Internat J Numer Methods in Fluids 4:377–397
6. Kostreva MM (1990) Recent results on complementarity models for engineering and economics. INFOR 28:324–334
7. Oh KP (1986) The formulation of the mixed lubrication problem as a generalized nonlinear complementarity problem. Trans ASME, J Tribology 108:598–604
8. Szanc BP (1989) The generalized complementarity problem. Rensselaer Polytech. Inst., Troy, NY, PhD Diss.

Generalized Outer Approximation

S. Leyffer

Department Math., University Dundee, Dundee, UK

MSC2000: 90C11, 90C30, 49M20

Article Outline

Keywords

Outer Approximation of (P)

When All $y \in Y$ Are Feasible

Infeasible Subproblems

The General Case

Linear Outer Approximation

Quadratic Outer Approximation
Avoiding Resolving the Master Problems
See also
References

Keywords

Mixed integer nonlinear programming; Outer approximation; Branch and bound

This article deals with the solution of *mixed integer nonlinear programming* (MINLP) problems of the form

$$(P)\quad \begin{cases} \min\limits_{x,y} & f(x,y) \\ \text{s.t.} & g(x,y) \le 0 \\ & x \in X,\ y \in Y \text{ integer.} \end{cases}$$

Throughout the following general assumptions are made:

A1) f and g are twice continuously differentiable and convex functions;

A2) X and Y are nonempty compact convex (polyhedral) sets; and

A3) a constraint qualification holds at the solution of every NLP subproblem obtained by fixing the integer variables y.

MINLP problems arise in a range of engineering applications (see, e. g., [8] and [10] and references therein).

A class of methods for MINLP problems is discussed, which provide an alternative to nonlinear branch and bound (cf. ► MINLP: Branch and bound methods) [3]. These algorithms are based on the concept of defining an *MILP master problem*. Relaxations of such a master problem are then used in constructing algorithms for solving the MINLP problem.

The methods presented here are a generalization of outer approximation proposed by M.A. Duran and I.E. Grossmann [4] (see also [14]) and of LP/NLP based branch and bound of I. Quesada and Grossmann [13].

The next section presents the reformulation of (P) as an MILP master program. Based on this reformulation two algorithms are presented in the following sections which solve a finite sequence of NLP subproblems and MILP or *MIQP master problems*, respectively. The final section shows how the re-solution of these master problems can be avoided by updating their branch and bound trees.

Outer Approximation of (P)

In this section the MINLP model problem (P) is reformulated as an MILP problem using outer approximation. The reformulation employs projection onto the integer variables and linearization of the resulting NLP subproblems by means of supporting hyperplanes. The convexity assumption allows an MILP formulation to be given where all supporting hyperplanes are collected in a single MILP.

In order to improve the readability of the material, the reformulation is first done under the simplifying assumption that all integer assignments $y \in Y$ are feasible. Next a rigorous treatment of infeasible subproblems is outlined, correcting an inaccuracy in [4] and [14], which could cause the algorithm to cycle. Finally, the two parts are combined and the correctly reformulated MILP master program is presented.

The reformulation presented in the next section affords new insight into Outer Approximation. It can be seen, for example, that it suffices to add the linearizations of *strongly active* constraints to the master program. This is very important since it reduces the size of the MILP master program relaxations that are solved in the outer approximation algorithms.

When All $y \in Y$ Are Feasible

In this subsection the simplifying assumption is made that all $y \in Y$ are feasible. The first step in reformulating (P) is to define the *NLP subproblem*

$$NLP(y^j)\quad \begin{cases} \min\limits_{x} & f(x,y^j) \\ \text{s.t.} & g(x,y^j) \le 0 \\ & x \in X \end{cases}$$

in which the integer variables are fixed at the value $y = y^j$. By defining $v(y^j)$ as the optimal value of the subproblem NLP(y^j) it is possible to express (P) in terms of a projection on to the y variables, that is

$$\min_{y^j \in Y} (v(y^j)). \tag{1}$$

The assumption that all $y \in Y$ are feasible implies that *all* subproblems are feasible. Let x^j denote an optimal solution of NLP(y^j) for $y^j \in Y$ (existence of x^j follows by the compactness of X). Because a constraint qualification holds at the solution of every subproblem NLP(y^j)

for every $y^j \in Y$, it follows that (1) has the same optimal value as the problem

$$\min_{y^j \in Y}(u(y^j)), \tag{2}$$

where $u(y^j)$ is the optimal value of the following LP

$$\begin{cases} \min\limits_{x} & f^j + (\nabla f^j)^\top \begin{pmatrix} x - x^j \\ 0 \end{pmatrix} \\ \text{s.t.} & 0 \geq g^j + [\nabla g^j]^\top \begin{pmatrix} x - x^j \\ 0 \end{pmatrix} \\ & x \in X. \end{cases} \tag{3}$$

In fact, it suffices to include those linearizations of constraints about (x^j, y^j) which are strongly active at the solution of the corresponding subproblem. Here $f^j = f(x^j, y^j)$ and $\nabla f^j = \nabla f(x^j, y^j)$, etc.

It is convenient to introduce a dummy variable $\eta \in \mathbf{R}$ into (3), giving rise to the equivalent problem

$$\begin{cases} \min\limits_{x,\eta} & \eta \\ \text{s.t.} & \eta \geq f^j + (\nabla f^j)^\top \begin{pmatrix} x - x^j \\ 0 \end{pmatrix} \\ & 0 \geq g^j + [\nabla g^j]^\top \begin{pmatrix} x - x^j \\ 0 \end{pmatrix} \\ & x \in X. \end{cases}$$

The convexity assumption A1) implies that (x^i, y^i) is feasible in the inner optimization problem above for all $y^i \in Y$, where x^i is an optimal solution to NLP(y^j). Thus an equivalent MILP problem

$$(M_Y) \begin{cases} \min\limits_{x,y,\eta} & \eta \\ \text{s.t.} & \eta \geq f^j + (\nabla f^j)^\top \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & 0 \geq g^j + [\nabla g^j]^\top \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & \forall y^j \in Y \\ & x \in X, \quad y \in Y \text{ integer} \end{cases}$$

is obtained. That is, (M_Y) has one set of linearizations of the objective and constraint functions per integer point $y^j \in Y$.

Infeasible Subproblems

Usually, not all $y \in Y$ give rise to feasible subproblems. Defining the sets

$$T = \left\{ j\colon\ x^j \text{ optimal solution to NLP}(y^j) \right\},$$
$$V = \{ y \in Y\colon\ \exists x \in X \text{ with } g(x, y) \leq 0 \}.$$

Then V is the set of all integer assignments y that give rise to feasible NLP subproblems and T is the set of indices of these integer variables. Then (P) can be expressed as a projection on to the integer variables

$$\min_{y^j \in V}(v(y^j)).$$

In this projection the set V replaces Y in (1). The equivalent MILP problem is now given by

$$(M_V) \begin{cases} \min\limits_{x,y,\eta} & \eta \\ \text{s.t.} & \eta \geq f^j + (\nabla f^j)^\top \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & 0 \geq g^j + [\nabla g^j]^\top \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & \forall j \in T \\ & x \in X, \quad y \in V \text{ integer} \end{cases}$$

obtained from (M_Y) by replacing Y by V.

It remains to find a suitable representation of the constraint $y \in V$ by means of supporting hyperplanes. The master problem given in [4] is obtained from problem (M_V) by replacing $y \in V$ by $y \in Y$. Duran and Grossmann 1986 justify this step by arguing that a representation of the constraints $y \in V$ is included in the linearizations in problem (M_V). However, it is not difficult to derive an MINLP problem where this would lead to an incorrect master problem [5], [11, p. 79].

In order to derive a correct representation of $y \in V$ it is necessary to consider how NLP solvers detect infeasibility. Infeasibility is detected when convergence to an optimal solution of a feasibility problem occurs. At such an optimum, some of the nonlinear constraints will be violated and other will be satisfied and the norm of the infeasible constraints can only be reduced by making some feasible constraints infeasible. A suitable framework for *nonlinear feasibility problems* in the context of

outer approximation is

$$F(y^k)\begin{cases}\min\limits_{x} & \sum\limits_{i\in J^{\perp}} w_i^k g_i^+(x,y^k)\\ \text{s.t.} & g_j(x,y^k)\le 0, \quad j\in J,\\ & x\in X.\end{cases}$$

The constraints in $F(y^k)$ have been divided into two sets: one that can be satisfied and another that cannot be satisfied. Infeasible subproblems now correspond to solutions of $F(y^k)$ with $\sum_{i\in J^{\perp}} w_i^k\, g_i^+\,(x,y^k)>0$. Most common feasibility problems such as l_1 and l_∞ as well as the feasibility filter [6] fit into this framework. The following lemma shows how solutions of $F(y^k)$ can be used to construct a representation of $y\in V$.

Lemma 1 *If NLP(y^k) is infeasible, so that x^k solves F(y^k) with*

$$\sum_{i\in J^{\perp}} w_i^k (g_i^k)^+ > 0, \tag{4}$$

then $y=y^k$ is infeasible in the constraints

$$0\ge g_i^k+(\nabla g_i^k)^{\top}\begin{pmatrix}x-x^k\\ y-y^k\end{pmatrix}, \quad \forall i\in J^{\perp}$$

$$0\ge g_j^k+(\nabla g_j^k)^{\top}\begin{pmatrix}x-x^k\\ y-y^k\end{pmatrix}, \quad \forall j\in J,$$

for all $x\in X$. The proof of this Lemma can be found in [5, Lemma 1].

The General Case

This subsection completes the derivation of the MILP master program by combining the developments of the previous two subsections. Let the integer assignment y^k produce an infeasible subproblem and denote

$$S=\left\{k:\ \text{NLP}(y^k)\ \text{infeasible},\ x^k\ \text{solves}\ \text{F}(y^k)\right\}.$$

Note that S is the complement of the set T defined in the previous subsection. It then follows directly from Lemma 1 that the constraints

$$0\ge g^k+[\nabla g^k]^{\top}\begin{pmatrix}x-x^k\\ y-y^k\end{pmatrix}, \quad \forall k\in S,$$

exclude all integer assignments y^k for which NLP(y^k) is infeasible. Thus a general way to correctly represent the constraints $y\in V$ in (M_V) is to add linearizations from $F(y^k)$ when infeasible subproblems are obtained, giving rise to the following MILP master problem:

$$(M)\begin{cases}\min\limits_{x,y,\eta} & \eta\\ \text{s.t.} & \eta\ge f^j+(\nabla f^j)^{\top}\begin{pmatrix}x-x^j\\ y-y^j\end{pmatrix}\\ & 0\ge g^j+[\nabla g^j]^{\top}\begin{pmatrix}x-x^j\\ y-y^j\end{pmatrix}\\ & \forall j\in T\\ & 0\ge g^k+[\nabla g^k]^{\top}\begin{pmatrix}x-x^k\\ y-y^k\end{pmatrix}\\ & \forall k\in S\\ & x\in X, \quad y\in Y \text{ integer}.\end{cases}$$

The development of the preceding two subsections provides a proof of the following result:

Theorem 2 *If assumptions A1), A2) and A3) hold, then (M) is equivalent to (P) in the sense that (x^*, y^*) solves (P) if and only if it solves (M).*

Problem (M) is an MILP problem, but it is not practical to solve (M) directly, since this would require all subproblems NLP(y^j) to be solved first. This would be a very inefficient way of solving problem (P). Therefore, instead of attempting to solve M directly, relaxations of (M) are used in an iterative process that is the subject of the next section.

Linear Outer Approximation

This section describes, how relaxations of the master program (M), developed in the previous section can be employed to solve the model problem (P). The resulting algorithm is termed *linear outer approximation*. It is shown to iterate finitely between NLP subproblems and MILP master program relaxations. This algorithm is seen to be less efficient if curvature information is present in the problem. A worst-case example, in which linear outer approximation visits all integer assignments has been derived in [5]. This example motivates the introduction of a second order term into the MILP master program relaxations, resulting in a *quadratic outer approximation* algorithm which is considered in the next section.

Each iteration of the linear outer approximation algorithm chooses a new integer assignment y^j and attempts to solve NLP(y^j). Either a feasible solution x^i is obtained or infeasibility is detected and x^i is the solution of a feasibility problem F(y^j) (other pathological cases are eliminated by the assumption that the set X is compact). The algorithm replaces the sets T and S in (M) by the sets

$$T^i = \{j \le i:\ x^j \text{ solution to NLP}(y^j)\},$$
$$S^i = \{k \le i:\ x^k \text{ solution to F}(y^k)\}.$$

It is also necessary to prevent any $y^j, j \in T^i$, from becoming the solution of the relaxed master problem. This can be done by including a constraint

$$\eta < \text{UBD}^i,$$

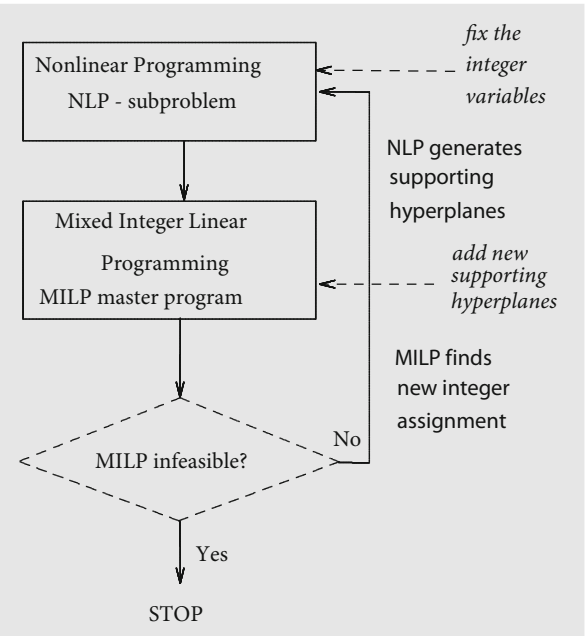
where

$$\text{UBD}^i = \min\{f^j:\ j \in T^i\}$$

is an upper bound on the optimum. Thus the following master problem is defined

$$(M^i)\begin{cases} \min\limits_{x,y,\eta} & \eta \\ \text{s.t.} & \eta < \text{UBD}^i \\ & \eta \ge f^j + \nabla(f^j)^\top \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & 0 \ge g^j + \nabla[g^j]^\top \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & \forall j \in T^i \\ & 0 \ge g^k + \nabla[g^k]^\top \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \\ & \forall k \in S^i \\ & x \in X, \quad y \in Y \text{ integer}. \end{cases}$$

The algorithm solves (M^i) to obtain a new integer assignment y^{i+1}, and the whole process is repeated iteratively. A detailed description of the algorithm is as follows.



Generalized Outer Approximation, Figure 1
Illustration of linear outer approximation

Initialization: y^0 given.
set $i = 0$, $T^{-1} = \emptyset$, $S^{-1} = \emptyset$, $\text{UBD}^{-1} = \infty$
REPEAT
1. Solve NLP(y^i) at F(y^i) as appropiate. Let the solution be x^i.
2. Liniarize objective and constraint functions about (x^i, y^i). Set $T^i = T^{i-1} \cup \{i\}$ or $S^i = S^{i-1} \cup \{i\}$ as appropriate.
3. IF (NLP(y^i) feasible AND $f^i < \text{UBD}^{i-1}$)
   THEN
   update current best point by setting $x^* = x^i$, $y^* = y^i$, $\text{UBD}^i = f^i$
   ELSE Set $\text{UBD}^i = \text{UBD}^{i-1}$.
4. Solve the current relaxation (M^i) of the master program (M), giving a new y^{i+1}. Set $i = i + 1$.

UNTIL ((M^i) is infeasible)

Algorithm 1: Linear outer approximation

The figure below illustrates the different stages of the algorithm.

The algorithm also detects whether or not (P) is infeasible. If UBD = ∞ on exit, then all integer assignments that are visited by the algorithm are infeasible

(i. e. Step 3 is not invoked). The use of upper bounds on η and the definition of the sets T^i and S^i ensure that no y^i is replicated by the algorithm. This enables a proof to be given that the algorithm terminates after a finite number of steps, provided that there is only a finite number of integer assignments.

Theorem 3 *If assumptions A1), A2) and A3) hold, and if $|Y| < \infty$, then Algorithm 1 terminates in a finite number of steps at an optimal solution of (P) or with an indication that (P) is infeasible.*

A proof of this theorem can be found in [5]. Below a brief outline of the proof is given: The optimality of x^i in NLP(y^i) implies that $\eta \geq f^i$ for any feasible point in (3). The upper bound $\eta < f^i$ therefore ensures that the choice $y = y^i$ in (M^k) has no feasible points $x \in X$. Therefore the algorithm is finite. The optimality of the algorithm follows from the convexity of f and g which ensures that the linearizations are supporting hyperplanes.

Quadratic Outer Approximation

Curvature can often play an important role in optimization. If this is the case, then an algorithm based on *linear* representations of the problem functions can be inefficient. In [5], a worst-case example is given for which linear outer approximation visits all integer points. This motivates the introduction of a curvature information into the master programs. In the remainder of this section it is shown how this can be achieved for linear outer approximation by including a second order Lagrangian term into the objective function of the MILP master programs.

These considerations have led to the development of a new algorithm based on the use of second order information. The development of such an algorithm seems contradictory at first sight, since quadratic functions do not provide underestimators of general convex functions. However, the derivation of the previous section allows the inclusion of a curvature term into the objective function of the MILP master problem. This quadratic term influences the choice of the next iterate by the algorithm without surrendering the finite convergence properties which rely on the fact that the feasible region of the master problem is an outer approximation of the feasible region of the MINLP problem P.

The resulting algorithm is referred to as *quadratic outer approximation* and is obtained by replacing the relaxed master problem (M^i) by the MIQP problem (Q^i) in Step 4 of Algorithm 1. Introducing the *quadratic* term

$$q^i(x,y) = \frac{1}{2}\begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix}^\top \nabla^2 \mathcal{L}^i \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix},$$

where

$$\mathcal{L}^i = \mathcal{L}(x^i, y^i, \lambda^i) = f(x^i, y^i) + (\lambda^i)^\top g(x^i, y^i)$$

is the usual Lagrangian function.

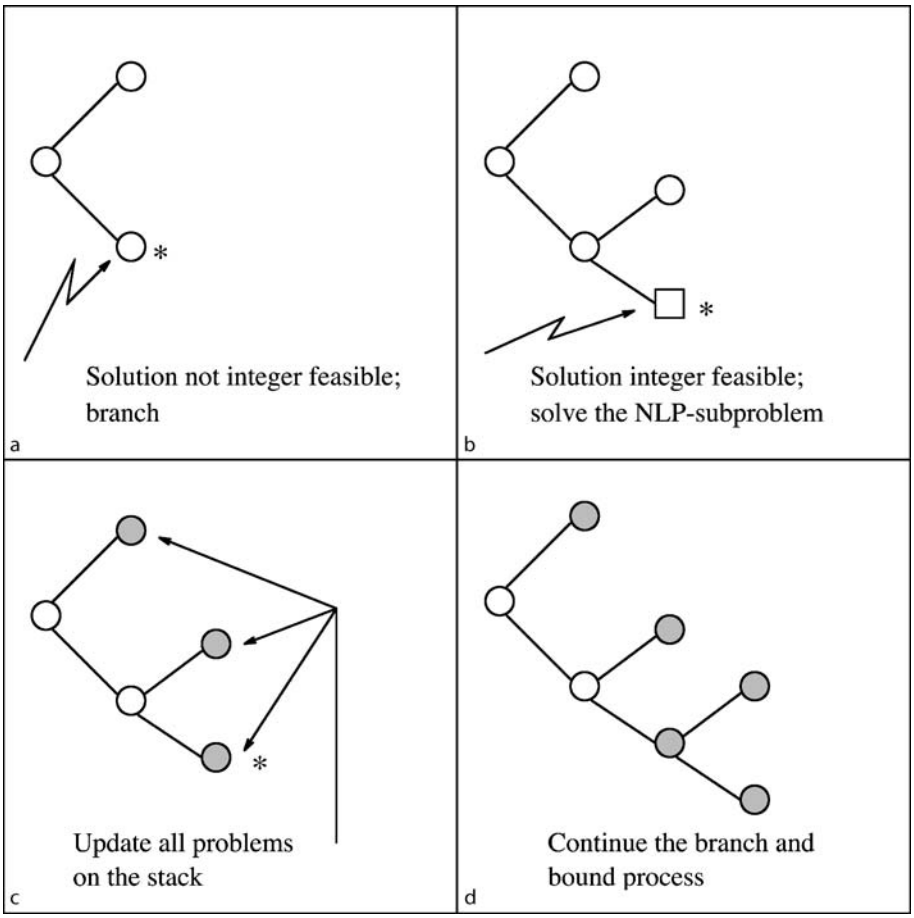
The new master problem (Q^i) can be defined as

$$(Q^i)\begin{cases} \min\limits_{x,y,\eta} & \eta + q^i(x,y) \\ \text{s.t.} & \eta < \text{UBD}^i \\ & \eta \geq f^j + (\nabla f^j)^\top \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & 0 \geq g^j + [\nabla g^j]^\top \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \\ & \forall j \in T^i \\ & 0 \geq g^k + [\nabla g^k]^\top \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \\ & \forall k \in S^i \\ & x \in X, \quad y \in Y \text{ integer.} \end{cases}$$

Numerical experience in [11, Chapter 6] indicates that adding a curvature term reduces the number of iterations of outer approximation if general integer variables are present. However, the iteration count is not reduced for problems involving binary variables only. As a consequence these preliminary results indicate that quadratic outer approximation only improves the computation times for problems with general integer variables, as MIQP problems are usually more expensive to solve than MILP problems.

Avoiding Resolving the Master Problems

This section presents a new approach to the solution of successive master program relaxations. It has been proposed by Quesada and Grossmann [13] for a class of problems whose objective and constraint functions are linear in the integer variables and is called *LP/NLP based branch and bound* algorithm. Their approach is generalized here to cover problems with nonlinearities



Generalized Outer Approximation, Figure 2
Progress of LP/NLP based branch and bound

in the integer variables. In addition a new algorithm *QP/NLP based branch and bound* is proposed based on the quadratic master program (Q^i) which takes curvature information into account.

The motivation for the LP/NLP based branch and bound algorithm is that outer approximation usually spends an increasing amount of computing time in solving successive MILP master program relaxations. Since the MILP relaxations are strongly related to one another this means that a considerable amount of information is re-generated each time a relaxation is solved. The new approach avoids the re-solution of MILP master program relaxations by updating the branch and bound tree. This section makes extensive use of branch and bound terminology; see the extensive literature on branch and bound (e. g., [1,2,8,9,12]) for the relevant definitions.

Instead of solving successive relaxations of (M), the new algorithm solves only one MILP problem which is updated as new integer assignments are encountered *during* the tree search. Initially an NLP-subproblem is solved and the initial master program relaxation (M^0) is set up from the supporting hyperplanes at the solution of the NLP-subproblem. The MILP problem (M^0) is then solved by a branch and bound process with the exception that each time a node (corresponding to an LP problem) gives an integer feasible solution y^i, say, the process is interrupted and the corresponding NLP(y^i) subproblem is solved. New linearizations from NLP(y^i) are then added to every node on the stack, effectively updating the branch and bound tree. The branch and bound process continues in this fashion until no problems remain on the stack. At that moment all nodes are fathomed and the tree search is exhausted.

Initialization: y^0 given;
set $i = 1,\ T^{-1} = \emptyset,\ S^{-1} = \emptyset$
Set up the initial master program:
Solve NLP(y^0). Let the solution be x^0.
Linearize objective and constraint functions about (x^0, y^0).
Set $T^0 = \{0\}$.
Set $x^* = x^0,\ y^* = y^0,\ \mathrm{UBD}^0 = f^0$.
Place (M^0) with its integer restrictions relaxed on the stack.
WHILE (stack is not empty) DO BEGIN
1. Remove a problem (P′) from the stack and solve the LP giving (x', y', η'). η' is a lower bound for all NLP child problems whose root is the current problem.
2. IF (y' integer) THEN
Set $y^i = y'$;
Solve NLP(y^i) or F(y^i).
Let the solution be x^i.
Linearize objective and constraint functions about (x^i, y^i).
Set $T^i = T^{i-1} \cup \{i\}$ or $S^i = S^{i-1} \cup \{i\}$.
Add linearizations to *all* pending problems on the stack.
IF (NLP(y^i)feasible AND $f^i < \mathrm{UBD}^i$) THEN
Update best point $x^* = x^i,\ y^* = y^i,\ \mathrm{UBD}^{i+1} = f^i$.
ELSE Set $\mathrm{UBD}^{i+1} = \mathrm{UBD}^i$.
ENDIF
Place (P′) back on stack; set $i = i + 1$.
Pruning: Remove all problems from stack with $\eta' > \mathrm{UBD}^{i+1}$.
ELSE
Branch on an integer variable and add two new problems to the stack.
ENDIF
END (WHILE-LOOP)

Algorithm 2: LP/NLP based branch and bound

Unlike ordinary branch and bound a node cannot be assumed to have been fathomed, if it produces an integer feasible solution, since the previous solution at this node is cut out by the linearizations added to the master program. Thus only infeasible nodes can be assumed to be fathomed. In the case of MILP master programs there exists an additional opportunity for pruning. Since the LP nodes are outer approximations of the MINLP subproblem corresponding to their respective subtree a node can be regarded as fathomed if its lower bound is greater than or equal to the current upper bound UBD^i.

As in the two outer approximation algorithms the use of an upper bound implies that no integer assignment is generated twice during the tree search. Since both the tree and the set of integer variables are finite the algorithm eventually encounters only infeasible problems and the stack is thus emptied so that the procedure stops. This provides a proof of the following consequence to Theorem 3.

Corollary 4 *If assumptions A1), A2) and A3) hold, and if $|Y| < \infty$, then Algorithm 2 terminates in a finite number of steps at a solution of (P) or with an indication that (P) is infeasible.*

The figure below illustrates the progress of Algorithm 2. In i), the LP relaxation of the initial MILP has been solved and two branches added to the tree. The LP that is solved next (indicated by an $*$) does not give an integer feasible solution and two new branches are introduced. The next LP in ii) produces an integer feasible solution indicated by a box. The corresponding NLP subproblem is solved and in iii) all nodes on the stack

are updated (indicated by the shaded circles) by adding the linearizations from the NLP subproblem including the upper bound UBD^i which cuts out the current assignment y^i. Then, the branch and bound process continues on the updated tree by solving the LP marked by a $*$.

If curvature information plays an important part in the problem (P), then it may be beneficial to add a quadratic term $q^i(x, y)$ to the master problem. This gives rise to *QP/NLP based branch and bound* algorithm. It differs from Algorithm 2 in two important aspects. The first difference is that QP rather than LP problems are solved in the tree search. The second difference is a consequence of the first. Since QP problems do not provide lower bounds on the MINLP problems (P), the pruning step in Algorithm 2 cannot be applied.

In preliminary numerical experiments in [11, Chapter 6] and [7] it has been observed that the LP and QP version of Algorithm 2 are usually faster than their counterparts based on Algorithm 1, often beating the latter by a factor of 2. A detailed numerical comparison of the two approaches is still outstanding.

See also

- ▶ Chemical Process Planning
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming

References

1. Beale EML (1978) Integer programming. In: Jacobs DAH (ed) The State of the Art in Numerical Analysis. Acad. Press, New York
2. Borchers B, Mitchell JE (1994) An improved branch and bound algorithm for mixed integer nonlinear programming. Comput Oper Res 21(4):359–367
3. Dakin RJ (1965) A tree search algorithm for mixed integer programming problems. Comput J 8:250–255
4. Duran M, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
5. Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Math Program 66:327–349
6. Fletcher R, Leyffer S (1997) Nonlinear programming without a penalty function. Numer Anal Report Univ Dundee, Dept Math NA/171
7. Fletcher R, Leyffer S (1998) Computing lower bounds for MIQP, branch-and-bound. SIAM J Optim 8:604–616
8. Floudas CA (1995) Nonlinear and mixed-integer optimization: Topics in chemical engineering. Oxford Univ. Press, Oxford
9. Garfinkel RS, Nemhauser GL (1972) Integer programming. Wiley, New York
10. Grossmann IE, Kravanja Z (1997) Mixed-integer nonlinear programming: A survey of algorithms and applications. In: Conn AR, Biegler LT, Coleman TF, Santosa FN (eds) Large-Scale Optimization with Applications, Part II: Optimal Design and Control. Springer, Berlin
11. Leyffer S (1993) Deterministic methods for mixed integer nonlinear programming. PhD Thesis Univ. Dundee
12. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
13. Quesada I, Grossmann IE (1992) An LP/NLP based branch-and-bound algorithm for convex MINLP: optimization problems. Comput Chem Eng 16:937–947
14. Yuan X, Zhang S, Pibouleau L, Domenech S (1988) Une méthode d'optimization non linéaire en variables mixtes pour la conception de procédés. Oper Res 22:331–346

Generalized Primal-relaxed Dual Approach

GPRD

Wenbin Liu
University Kent, Canterbury, England

MSC2000: 90C26

Article Outline

Keywords
See also
References

Keywords

Global optimization; Primal-relaxed dual approach; GOP algorithm

Generalized primal-relaxed dual approach (GPRD) in the context of global optimization employs the Benders' idea of partitioning (see [2]) in order to exploit the structure of global optimization problems with *complicating variables* (variables which, when temporarily fixed, render the remaining optimization problem much simpler, see [4]). For the class of global optimization problem considered by the GPRD approach, fixing the values of the complicating variables reduces the given problem to a convex program, parameterized by the values of the complicating variables. In order to approximate the solution of this class of problems efficiently, the GPRD approach uses the primal and relaxed dual problems with fixed complicating variables to provide sharper upper and lower bounds of the solution respectively, following the original ideas in [2,4] and [9].

It however adopts a different way of constructing relaxed dual problems by generalizing the original method used in the GOP algorithms (see [3]) so that it can handle a wider range of global optimization problems including nonsmooth ones.

Let k, p, n, m be some positive integers. Let X and Y be two closed sets in $\mathbf{R}^n$ and $\mathbf{R}^m$ respectively. Let f, g_i, h_j ($1 \le i \le k$ and $1 \le j \le p$) be continuous functions on $\mathbf{R}^n \times \mathbf{R}^m$. Let $g = (g_1, \ldots, g_k)^\intercal$ and $h = (h_1, \ldots, h_p)^\intercal$.

Let us consider the following global optimization problem:

$$(OP)\ v = \begin{cases} \min\limits_{x,y} & f(x,y) \\ \text{s.t.} & g(x,y) \le 0, \\ & h(x,y) = 0, \\ & x \in X, \quad y \in Y, \\ & 1 \le i \le k, \quad 1 \le j \le p, \end{cases}$$

where X and Y are nonempty closed convex sets in $\mathbf{R}^n$ and $\mathbf{R}^m$ respectively. It is assumed that for any fixed $y \in Y$, or $x \in X$, $1 \le i \le k$, and $1 \le j \le p$, $f(\cdot, y)$, $g_i(\cdot, y)$, $f(x, \cdot)$, $g_i(x, \cdot)$ are convex functions, and $h_j(\cdot, y)$, $h_j(x, \cdot)$ are affine functions. It is also assumed that for any fixed $y \in V = \{y \in Y$: there is an $x \in X$: $g(x, y) \le 0$ and $h(x, y) = 0\}$, the partial primal problem (OP) is stable in the sense that its perturbation function has a nonempty subgradient at zero point; see [1]. This assumption holds when, for instance, the Slater's constraint qualification holds for (OP) at every fixed $y \in V$, though it is more general than the Slater's.

Although the problem (OP) appears to address only a limited class of global optimization programs, it is shown in [5] that very broad mathematical programming problems can indeed be reformulated in this form by using a simple variable transformation. Furthermore, it is shown in [6] that for any fixed $y \in Y$ the reformulated problems are always stable.

It follows from the stability assumption that for any fixed $y_0 \in V$ there exist Lagrange multipliers $(\lambda_0, \mu_0) \in \mathbf{R}^p \times \mathbf{R}^k_+$ and $x_0 \in X$ such that the triplet (x_0, λ_0, μ_0) is the solution of the following saddle problem:

$$\begin{cases} g(x_0, y_0) \le 0, \\ h(x_0, y_0) = 0, \\ \mu_0^\top g(x_0, y_0) = 0, \end{cases}$$

and for any $(x, \lambda, \mu) \in X \times \mathbf{R}^p \times \mathbf{R}^k_+$

$$(SP)\quad \begin{aligned} L(x_0, y_0, \lambda, \mu) &\le L(x_0, y_0, \lambda_0, \mu_0) \\ &\le L(x, y_0, \lambda_0, \mu_0), \end{aligned}$$

where the Lagrange function of the primal problem (OP) is defined by

$$L(x, y, \lambda, \mu) = f(x, y) + \lambda^\top h(x, y) + \mu^\top g(x, y).$$

The solution (x_0, λ_0, μ_0) of (SP) can be found by solving the following partial primal problem:

$$(PP)\quad \min_{\substack{x \in X, \\ g(x, y_0) \le 0, \\ h(x, y_0) = 0}} f(x, y_0),$$

which is a convex minimization problem.

For a given $y_0 \in V$, the GPRD approach finds an upper bound for the solution of (OP) by solving (PP):

$$\nu^+(y_0) = \min_{\substack{x\in X,\\ g(x,y_0)\le 0,\\ h(x,y_0)=0}} f(x, y_0).$$

The problem (PP) is in general easier to solve as it is convex. The GPRD approach then estimates a lower bound for the solution of (OP) by solving the following relaxed dual problem:

$$(RD) \quad \nu^-(U,H) = \min_{y\in V} \max_{(\lambda_t,\mu_t)\in U} \min_{x\in X} H^{(\lambda_t,\mu_t)}(x,y),$$

where $U = \{(\lambda_t, \mu_t) \in \mathbf{R}^p \times \mathbf{R}^m_+ : 1 \le t \le N\}$, and the mapping $H: U \to C^0(X \times Y)$ is such that the function $H^{(\lambda_t,\mu_t)}(\cdot,\cdot)$ is continuous and satisfies that for any fixed $(\lambda_t, \mu_t) \in U$,

$$\begin{aligned} &L(x, y, \lambda_t, \mu_t) \\ &= f(x, y) + \lambda_t^\top h(x, y) + \mu_t^\top g(x, y) \\ &\ge H^{(\lambda_t,\mu_t)}(x, y), \quad \forall (x, y) \in X \times Y. \end{aligned}$$

In the GPRD approach, the set U is usually constructed to include the multipliers (λ, μ) obtained from solving the problem (PP) above.

The *generalized primal-relaxed dual algorithm* is to construct, for $n = 0, 1, \ldots$, a sequence of elements $y_n \in Y$, sets U_n, and functions $H_n^{(\lambda_t,\mu_t)}$ for each $(\lambda_t, \mu_t) \in U_n$ such that $\nu^+(y_n) - \nu^-(U_n, H_n) \to 0$ as $n \to 0$. The selections of U_n and $H_n^{(\lambda_t,\mu_t)}$ are clearly not unique but they have to be constructed so that the *global* solutions of the relaxed-dual problems (RD) can be solved efficiently. In the literature U_n is set to be the unit of all the Lagrange multipliers (λ, μ) of the partial primal problems (PP) with $y = y_m$ $(m = 1, \ldots, n)$. Assume that the selection $H_n^{(\lambda_t,\mu_t)}$ is given for any $(\lambda_t, \mu_t) \in U_n$. Then the generalized primal-relaxed dual algorithm reads:

1	Given $y_0 \in V$, and $\epsilon > 0$.
2	Given $y_n \in V$, solve (PP) for $y = y_n$ to obtain x_n and Lagrange multiplies (λ_n, μ_n).
3	Solve (RD) to obtain y_{n+1}, where $U_n = \cup_{m=1}^n \{(\lambda_m, \mu_m)\}$.
4	Stop if $\nu^+(y_n) - \nu^-(U_n, H_n) < \epsilon$. Otherwise go to Step 2 with $n = n + 1$.

PRD Algorithm for (OP)

It is shown in [7] that the PRD algorithm converges to the global solutions of (OP) under some mild assumptions.

There are many possible choices for the mapping role H. In the literature the following results have been reported. In Geoffrion's original work in [4], $H_n^{(\lambda_m,\mu_m)}(x, y) = L(x, y, \lambda_m, \mu_m)$ $(1 \le m \le n)$. It is in general difficult to solve (RD) computationally with this choice of H_n. In the pioneer work [3], H_n takes the form of

$$\begin{aligned} &H_n^{(\lambda_m,\mu_m)}(x, y) \\ &\quad = L(x_m, y_m, \lambda_m, \mu_m) \\ &\qquad + \nabla_x L(x_m, y, \lambda_m, \mu_m)(x - x_m) \\ &\qquad + \nabla_y L(x_m, y_m, \lambda_m, \mu_m)(y - y_m) \\ &\qquad\qquad (m = 1, \ldots, n), \end{aligned}$$

where x_m, y_m, λ_m, μ_m are obtained from the previous iterations of the PRD algorithm and $\nabla_x L(x, y, \lambda, \mu)$ (or $\nabla_y L(x, y, \lambda, \mu)$) is the gradient of the Lagrange function L at x (or y) for a fixed (y, λ, μ) (or (x, λ, μ)). The resulting PRD algorithm has been referred to as *GOP algorithm* and has been widely used in various global optimization problems (see, e. g., [8] for a survey). Important progress has been made in developing efficient ways of solving (RD) for the GOP algorithm, see, also [8].

The GOP algorithm is only applicable to smooth optimization problems where the objective functions and the constraints are differentiable. Nonsmooth optimization problems occur in many important real applications. In [7] the GPRD approach is applied to a class of nonsmooth global optimization problems where

$$f = F + \max_{e\in E} F^e$$

and

$$g_i = G_i + \max_{e\in E} G_i^e, \quad i = 1, \ldots, k,$$

where $E = \{1, \ldots, d\}$, and the smooth C^1 functions F, F^e, $G = (G_1, \ldots, G_k)^\top$, $G^e = (G_1^e, \ldots, G_k^e)^\top$ satisfy all the conditions in (OP) for $e = 1, \ldots, d$. The resulting algorithm, referred to as *EGOP*, reads:

1 Given $y_0 \in V$, and $\epsilon > 0$.
2 Given $y_n \in V$, solve (PP) for $y = y_n$ to obtain x_n and Lagrange multipliers (λ_n, μ_n).
3 Solve (RD) to obtain y_{n+1}, where $U_n = \cup_{m=1}^{n}\{(\lambda_m, \mu_n)\}$ and for any $(\lambda_m, \mu_m) \in U_n$,

$$
\begin{aligned}
&H_n^{(\lambda_m,\mu_m)}(x, y)\\
&= LS(x_m, y_m, \lambda_m, \mu_m)\\
&+\nabla_x LS(x_m, y, \lambda_m, \mu_m)(x - x_m)\\
&+\nabla_y LS(x_m, y_m, \lambda_m, \mu_m)(y - y_m)\\
&+ \max_{e\in E}(F^e(x_m, y_m)+\nabla_x F^e(x_m, y)(x-x_m)\\
&+\nabla_y F^e(x_m, y_m)(y - y_m))\\
&+ \sum_{i=1}^{k} \mu_m^i \max_{e\in E}(G_i^e(x_m, y_m))\\
&+\nabla_x G_i^e(x_m, y)(x - x_m)\\
&+\nabla_y G_i^e(x_m, y_m)(y - y_m),
\end{aligned}
$$

where the smooth part of the Lagrange is defined by $LS(x, y, \lambda, \mu) = F(x, y) + \lambda^\top h(x, y) + \mu^\top G(x, y)$.
4 Stop if $v^+(y_n) - v^-(U_n, H_n) < \epsilon$. Otherwise go to Step 2 with $n = n + 1$.

EGOP Algorithm for (OP)

This algorithm is identical with the GOP algorithm in the smooth case where $F^e = 0$, $G^e = 0$ for $e = 1, \ldots, d$. The EGOP covers a wider range of global optimization problems, and it is shown in [7] to be convergent under essentially the same assumptions which ensure the convergence of the GOP algorithm.

Penalty implementation of the PRD algorithm has also been considered in the literature to explore another way of coping with possible infeasible primal or relaxed dual subproblems in the algorithm. In [7], the EGOP algorithm is applied to the following two penalty problems:

$$(NPOP)_\rho \min_{(x,y)\in X\times Y} P(x, y),$$

where

$$P(x, y) = f(x, y) + \rho \sum_{j=1}^{k} \max\left(0, g_j(x, y)\right) + \rho \sum_{j=1}^{p} \left|h_j(x, y)\right|, \quad \rho > 0\,.$$

and

$$(SPOP)_M \min_{(x,y)\in X\times Y} P(x, y),$$

where

$$P(x, y) = f(x, y) + M \sum_{j=1}^{k} \max\left(0, g_j(x, y)\right)^2 + M \sum_{j=1}^{p} \left|h_j(x, y)\right|^2, \quad M > 0.$$

The convergence of the two penalty implementations of EGOP algorithm is established in [7], where it is shown that the sequences $\{(x_n, y_n)\}$ generated by the EGOP algorithm for the penalty problems $(\text{NPOP})_\rho$ and $(\text{SPOP})_M$ converge to the solutions of the (OP) when ρ is large enough or M tends to infinite.

See also

- ▶ αBB Algorithm
- ▶ Global Optimization in Phase and Chemical Reaction Equilibrium

References

1. Avriel M (1976) Nonlinear programming: Analysis and methods. Prentice-Hall, Englewood Cliffs, NJ
2. Benders JF (1962) Partitioning procedures for solving mixed-variables programing problems. Numerische Math 4:238–252
3. Floudas CA, Visweswaran V (1990) A global optimization algorithm (GOP) for certain class of nonconvex NLPs -I. Theory. Comput Chem Eng 14:1397–1417
4. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
5. Liu WB, Floudas CA (1993) A remark on the GOP algorithm for global optimization. J Global Optim 3:519–521
6. Liu WB, Floudas CA (1995) Convergence of the (GOP) algorithm for a large class of smooth optimization problems. J Global Optim 6:607–611
7. Liu WB, Floudas CA (1996) Generalized primal-relaxed dual approach for global optimization. J Optim Th Appl 14:416–434
8. Visweswaran V, Floudas CA (1996) New formulations and branching strategies for the GOP algorithm. In: Grossmann IE (ed) Global optimization in chemical engineering. Kluwer, Dordrecht, pp 75–100
9. Wolsey LA (1981) A resource decomposition algorithm for general mathematical programs. Math Program Stud 90:244–257

Generalized Semi-infinite Programming: Optimality Conditions

Oliver Stein
School of Economics and Business Engineering, University of Karlsruhe, Karlsruhe, Germany

MSC2000: 90C34, 90C46, 90C31

Article Outline

Introduction
Definitions
Topological Properties
The Reduction Ansatz
First-Order Properties of the Feasible Set
Constraint Qualifications
Formulation
Second-Order Optimality Conditions
Conclusions
See also
References

Introduction

In generalized semi-infinite optimization problems, a finite-dimensional decision variable x is subject to infinitely many inequality constraints, that is, in

$$GSIP: \quad \text{minimize} \quad f(x) \quad \text{subject to} \quad x \in M\,,$$

the feasible set is described by

$$M = \{x \in \mathbb{R}^n | g(x,y) \le 0 \text{ for all } y \in Y(x)\}\,,$$

with the index set

$$Y(x) = \{y \in \mathbb{R}^m | v_\ell(x,y) \le 0, \ell \in L\}\,.$$

All defining functions $f, g, v_\ell, \ell \in L = \{1,\dots,s\}$ are assumed to be real valued and at least continuous on their respective domains. Moreover, the set-valued mapping $Y : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$ is assumed to be locally bounded, that is, for each $\bar{x} \in \mathbb{R}^n$ there exists a neighborhood U of $\bar{x}$ such that $\bigcup_{x\in U} Y(x)$ is bounded in $\mathbb{R}^m$.

In applications such as design centering, robust optimization, and (reverse) Chebyshev approximation ([13,32]), often finitely many semi-infinite constraints $g_i(x,y) \le 0,\ y \in Y_i(x),\ i \in I$, describe the feasible set M of *GSIP*, along with finitely many equality constraints in the definitions of M and $Y(x)$. In order to avoid technicalities this article focuses on the basic case of a single semi-infinite constraint (see [13,32] for more general formulations).

As opposed to a standard semi-infinite optimization problem, the possibly infinite index set $Y(x)$ of the semi-infinite inequality constraint is allowed to vary with x in a *GSIP*. For surveys and detailed studies about *standard* semi-infinite optimization see [10,15,25]. In contrast to standard semi-infinite programs, the feasible set of *GSIP* is not necessarily a closed set, and it might possess a stable disjunctive structure ([32]).

Powerful optimality conditions are based on a thorough analysis of these topological structures. This article mainly deals with first-order optimality conditions and will, thus, begin with a discussion of different first-order properties of the feasible set.

Definitions

The key to understanding the topological features in the feasible set of *GSIP* lies in the bilevel structure of semi-infinite programming ([27,32]). In fact, it is not hard to see that the semi-infinite constraint in *GSIP* is equivalent to

$$\varphi(x) := \max_{y\in Y(x)} g(x,y) \le 0\,,$$

which means that the feasible set M of *GSIP* is the lower-level set of some optimal value function:

$$M = \{x \in \mathbb{R}^n | \varphi(x) \le 0\}\,. \tag{1}$$

The usual convention "$\max\emptyset = -\infty$" is consistent here, as an empty index set $Y(x)$ corresponds, loosely speaking, to "the absence of restrictions" at x and, hence, to the feasibility of x.

The function ϕ is the optimal value function of the so-called *lower-level problem*

$$Q(x): \quad \max_{y\in\mathbb{R}^m} g(x,y) \quad \text{subject to} \quad v_\ell(x,y) \le 0, \quad \ell \in L.$$

In contrast to the upper-level problem, which consists in minimizing f over M, in the lower-level problem x plays the role of an n-dimensional parameter, and y is the decision variable. The main computational problem in semi-infinite programming is that the lower-level

problem has to be solved to global optimality, even if, for example, only a stationary point of the upper-level problem is sought.

Topological Properties

The alternative description of the feasible set in (1) shows that the topological properties of M are determined by the continuity properties of ϕ, whereas first- and second-order optimality conditions will rely on the first- and second-order properties of ϕ. The properties of optimal value functions have been studied extensively in parametric optimization ([2]; for a brief introduction see [32]).

The optimal value function ϕ can be shown to be at least upper semicontinuous, so that points $x \in \mathbb{R}^n$ with $\varphi(x) < 0$ belong to the topological interior of M. On the other hand, for investigations of the local structure of M or of local optimality conditions one is only interested in feasible points from the boundary ∂M of M. Hence it suffices to consider the zeros of ϕ, that is, points $x \in \mathbb{R}^n$ for which $Q(x)$ has maximal value $\varphi(x) = 0$. We denote the globally maximal points of $Q(x)$ for arbitrary $x \in \mathbb{R}^n$ by

$$Y_\star(x) = \{y \in Y(x) | g(x, y) = \varphi(x)\}$$

and for the special case of $x \in M \cap \partial M$ by

$$Y_0(x) = \{y \in Y(x) | g(x, y) = 0\} .$$

The set $Y_0(x)$ is also called the upper-level *active index set* of *GSIP*.

Note that M is closed if for all $x \in \mathbb{R}^n$ the index set $Y(x)$ is nonempty and the *Mangasarian–Fromovitz constraint qualification (MFCQ)* holds at some element of $Y_0(x)$ ([13,32]). If M is not closed, there may exist infeasible boundary points $x \in \partial M$, that is, boundary points with $\varphi(x) > 0$.

The Reduction Ansatz

For theoretical as well as numerical purposes it is of crucial importance to keep track of the elements of $Y_\star(x)$ for varying x. These points solve the lower-level problem so that for functions g and $v_\ell, \ell \in L$, which are continuously differentiable with respect to y, they satisfy the first-order necessary optimality condition of Karush–Kuhn–Tucker: let

$$\mathcal{L}(x, y, \gamma) = g(x, y) - \gamma^\top v(x, y) ,$$

denote the Lagrangian of $Q(x)$ with multiplier vector $\gamma \in \mathbb{R}^s$. Then for $\bar{x} \in M$ and each $\bar{y} \in Y_\star(\bar{x})$ such that MFCQ holds at $\bar{y}$ in $Q(\bar{x})$, there exist multipliers $\bar{\gamma} \geq 0$ with $D_y \mathcal{L}(\bar{x}, \bar{y}, \bar{\gamma}) = 0$ and $\bar{\gamma}_\ell \cdot v_\ell(\bar{x}, \bar{y}) = 0$, $\ell \in L$. Here $D_y \mathcal{L}$ denotes the gradient of $\mathcal{L}$ with respect to y as a row vector. Note that the multiplier vector $\bar{\gamma}$ is uniquely determined if instead of MFCQ the stronger *linear independence constraint qualification (LICQ)* holds at $\bar{y}$.

Keeping track of the elements of $Y_\star(x)$ can now be achieved, for example, by means of the implicit function theorem, if the functions g and $v_\ell, \ell \in L$, are C^2 with respect to y. For $\bar{x} \in M$ a local maximizer $\bar{y}$ of $Q(\bar{x})$ is called *nondegenerate* in the sense of Jongen/Jonker/Twilt ([19]), if LICQ, strict complementary slackness and a second-order sufficiency condition are satisfied. The *Reduction Ansatz* ([14,16,35]) is said to hold at $\bar{x} \in M$ if all global maximizers of $Q(\bar{x})$ are nondegenerate. The set $Y_\star(\bar{x})$ can then only contain finitely many points, say, $Y_\star(\bar{x}) = \{\bar{y}^1, \ldots, \bar{y}^p\}$ with $p \in \mathbb{N}$. By a result from [8] the local variation of these points with x can be described by the implicit function theorem.

In fact, for x locally around $\bar{x}$ there exist continuously differentiable functions $y^i(x)$, $1 \leq i \leq p$, with $y^i(\bar{x}) = \bar{y}^i$ such that $y^i(x)$ is the locally unique local maximizer of $Q(x)$ around $\bar{y}^i$. It turns out that the functions $\varphi_i(x) := g(x, y^i(x))$ are even C^2 in a neighborhood of $\bar{x}$. Their gradients are

$$D\varphi_i(\bar{x}) = D_x \mathcal{L}(\bar{x}, \bar{y}^i, \bar{\gamma}^i) , \tag{2}$$

where $\bar{\gamma}^i$ is the uniquely determined multiplier vector corresponding to $\bar{y}^i$.

A major consequence of the Reduction Ansatz is the so-called Reduction Lemma ([16]): if the Reduction Ansatz holds at $\bar{x}$, then for all x from a neighborhood U of $\bar{x}$ one has

$$\varphi(x) = \max_{1 \leq i \leq p} \varphi_i(x) .$$

In view of (1) this means that locally around a feasible boundary point $\bar{x} \in M \cap \partial M$, the feasible set M can be

described by finitely many C^2—constraints, that is, *GSIP* locally looks like a smooth finite optimization problem:

$$M \cap U = \{x \in U \mid \varphi_i(x) \leq 0, i = 1, \ldots, p\}\,. \quad (3)$$

In particular, locally around $\bar{x}$ set M is closed, and it does not possess a stable disjunctive structure at $\bar{x}$.

First-Order Properties of the Feasible Set

Since the Reduction Ansatz cannot be expected to hold generically *everywhere* in $M \cap \partial M$, the first-order structure of M is also studied under considerably weaker assumptions. For the first-order approximation of M one defines the *contingent cone* $\Gamma^\star(\bar{x}, M)$ to M at $\bar{x}$ as follows: $\bar{d} \in \Gamma^\star(\bar{x}, M)$ if and only if there exist sequences of scalars $(t^\nu)_{\nu\in\mathbb{N}}$ and of vectors $(d^\nu)_{\nu\in\mathbb{N}}$ such that

$$t^\nu \searrow 0, d^\nu \to \bar{d} (\nu \to \infty) \quad \text{and} \quad \bar{x} + t^\nu d^\nu \in M \quad \text{for all} \quad \nu \in \mathbb{N}.$$

The contingent cone is a closed cone, not necessarily convex, containing first-order information about M. In view of (1) the contingent cone to M at $\bar{x}$ should be related to a level set of a first order approximation of ϕ at $\bar{x}$. Unfortunately, the differentiability properties of ϕ can be very weak, so that *lower and upper directional derivatives* of ϕ at $\bar{x}$ in direction $\bar{d}$ in the Hadamard sense ([4]) come into play:

$$\varphi'_-(\bar{x}, \bar{d}) = \liminf_{t\searrow 0, d\to\bar{d}} \frac{\varphi(\bar{x} + td) - \varphi(\bar{x})}{t}$$

and

$$\varphi'_+(\bar{x}, \bar{d}) = \limsup_{t\searrow 0, d\to\bar{d}} \frac{\varphi(\bar{x} + td) - \varphi(\bar{x})}{t}\,.$$

ϕ is called *directionally differentiable* at $\bar{x}$ (in the Hadamard sense) if for each direction $d \neq 0$ one has $\varphi'_-(\bar{x}, d) = \varphi'_+(\bar{x}, d) =: \varphi'(\bar{x}, d)$. The *outer linearization cone* of M at $\bar{x}$ can now be defined as

$$L^\star(\bar{x}, M) = \{d \in \mathbb{R}^n \mid \varphi'_-(\bar{x}, d) \leq 0\}$$

and the *inner linearization cone* by

$$L(\bar{x}, M) = \{d \in \mathbb{R}^n \mid \varphi'_+(\bar{x}, d) < 0\}\,.$$

For $\bar{x} \in \partial M \cap M$ the chain of inclusions

$$L(\bar{x}, M) \subset \Gamma^\star(\bar{x}, M) \subset L^\star(\bar{x}, M) \quad (4)$$

holds ([22,33]). A good first-order description of M around $\bar{x}$ can thus be obtained if the linearization cones $L(\bar{x}, M)$ and $L^\star(\bar{x}, M)$ do not differ too much from each other.

For example, in standard semi-infinite programming the index set mapping $Y(x) \equiv Y$ is constant, and the theorem of Danskin ([6]) then says that ϕ is directionally differentiable with

$$\varphi'(\bar{x}, d) = \max_{y\in Y_0(\bar{x})} D_x g(\bar{x}, y)d$$

for all $d \in \mathbb{R}^n$. The linearization cones

$$L(\bar{x}, M) = \bigcap_{y\in Y_0(\bar{x})} \{d \in \mathbb{R}^n \mid D_x g(\bar{x}, y)d < 0\}$$

and

$$L^\star(\bar{x}, M) = \bigcap_{y\in Y_0(\bar{x})} \{d \in \mathbb{R}^n \mid D_x g(\bar{x}, y)d \leq 0\}$$

thus differ only by the strictness of inequalities, and they do not possess a disjunctive structure.

If in *GSIP* the Reduction Ansatz holds at $\bar{x}$, using (2) it is not hard to see that ϕ is directionally differentiable with

$$\varphi'(\bar{x}, d) = \max_{1\leq i\leq p} D_x \mathcal{L}(\bar{x}, \bar{y}^i, \bar{\gamma}^i)d$$

for all $d \in \mathbb{R}^n$. The linearization cones

$$L(\bar{x}, M) = \bigcap_{i=1}^{p} \{d \in \mathbb{R}^n \mid D_x \mathcal{L}(\bar{x}, \bar{y}^i, \bar{\gamma}^i)d < 0\}$$

and

$$L^\star(\bar{x}, M) = \bigcap_{i=1}^{p} \{d \in \mathbb{R}^n \mid D_x \mathcal{L}(\bar{x}, \bar{y}^i, \bar{\gamma}^i)d \leq 0\}$$

again differ only by the strictness of inequalities.

Under weaker assumptions than the Reduction Ansatz the situation in *GSIP* becomes more involved since ϕ does not even have to be directionally differentiable. The following estimates for the upper and lower directional derivatives from [9,23] are known to

be tight: for $\bar{x} \in \partial M \cap M$ such that MFCQ is satisfied at each $y \in Y_0(\bar{x})$ one has for each $d \in \mathbb{R}^n$

$$\sup_{y \in Y_0(\bar{x})} \min_{\gamma \in KKT(\bar{x},y)} D_x \mathcal{L}(\bar{x}, y, \gamma)\, d \le \varphi'_-(\bar{x}, d)$$
$$\le \varphi'_+(\bar{x}, d) \le \max_{y \in Y_0(\bar{x})} \max_{\gamma \in KKT(\bar{x},y)} D_x \mathcal{L}(\bar{x}, y, \gamma) d\,.$$

Here

$$KKT(x, y) = \{\gamma \in \mathbb{R}^s | \gamma \ge 0, D_y \mathcal{L}(x, y, \gamma) = 0, \gamma_\ell \cdot v_\ell(x, y) = 0, \ell \in L\}$$

denotes the set of Karush–Kuhn–Tucker multipliers at y in $Q(x)$.

At least this yields estimates for the linearization cones:

$$\bigcap_{y \in Y_0(\bar{x})} \bigcap_{\gamma \in KKT(\bar{x},y)} \{d \in \mathbb{R}^n | D_x \mathcal{L}(\bar{x}, y, \gamma)\, d < 0\}$$
$$\subset L(\bar{x}, M) \subset \Gamma^\star(\bar{x}, M) \subset L^\star(\bar{x}, M)$$
$$\subset \bigcap_{y \in Y_0(\bar{x})} \bigcup_{\gamma \in KKT(\bar{x},y)} \{d \in \mathbb{R}^n | D_x \mathcal{L}(\bar{x}, y, \gamma) d \le 0\}\,.$$

However, the estimate for the inner linearization cone is rather poor in many situations in which the problem data are endowed with a special structure. In [31] analogous estimates are given without the assumption of MFCQ in $Y_0(\bar{x})$.

A disjunctive structure of $\Gamma^\star(\bar{x}, M)$ is intimately related to the nonuniqueness of the lower-level Karush–Kuhn–Tucker multipliers. This becomes clearer under the assumption that the lower-level problems $Q(x)$, $x \in U$, are convex for some neighborhood U of $\bar{x}$, and that $Y(\bar{x})$ possesses a Slater point. Due to results from [11,18,26] the multiplier set $KKT(\bar{x})$ then does not depend on $y \in Y_0(\bar{x})$, and ϕ is directionally differentiable at $\bar{x}$ with

$$\varphi'(\bar{x}, d) = \min_{\gamma \in KKT(\bar{x})} \max_{y \in Y_0(\bar{x})} D_x \mathcal{L}(\bar{x}, y, \gamma)\, d$$

for all $d \in \mathbb{R}^n$. This yields

$$L(\bar{x}, M) = \bigcup_{\gamma \in KKT(\bar{x})} \bigcap_{y \in Y_0(\bar{x})} \{d \in \mathbb{R}^n |\, D_x \mathcal{L}(\bar{x}, y, \gamma)\, d < 0\}$$

and

$$L^\star(\bar{x}, M) = \bigcup_{\gamma \in KKT(\bar{x})} \bigcap_{y \in Y_0(\bar{x})} \{d \in \mathbb{R}^n | D_x \mathcal{L}(\bar{x}, y, \gamma)\, d \le 0\}\,.$$

Now both the inner and outer linearization cones possess a disjunctive structure, and they only differ by the strictness of inequalities. Moreover it becomes obvious that the occurrence of a stable disjunctive structure in *GSIP* is caused by nonunique lower-level Karush–Kuhn–Tucker multipliers. For more details on lower-level problems with a special structure see [27,29,32].

Constraint Qualifications

In what follows let the functions f, g, and v_ℓ , $\ell \in L$, be continuously differentiable. It is well known ([3]) that at a local minimizer $\bar{x}$ of f on M the following primal first-order necessary optimality condition holds:

$$\{d \in \mathbb{R}^n | Df(\bar{x}) d < 0\} \cap \Gamma^\star(\bar{x}, M) = \emptyset\,. \tag{5}$$

To obtain a more explicit condition from (5) one needs an explicit description of $\Gamma^\star(\bar{x}, M)$. A good candidate would be the outer linearization cone $L^\star(\bar{x}, M)$, which contains the contingent cone by (4). Even in finite optimization simple examples show, however, that $\Gamma^\star(\bar{x}, M)$ can be a proper subset of $L^\star(\bar{x}, M)$. In this case one cannot replace the contingent cone in (5) by the outer linearization cone.

On the other hand, in view of (4) it is always possible to replace the contingent cone in (5) by the inner linearization cone. However, the resulting optimality condition may be trivially satisfied since $L(\bar{x}, M)$ can be void itself.

These observations give rise to the following definitions.

Definition 1 The *extended Mangasarian–Fromovitz constraint qualification* (EMFCQ) holds at $\bar{x} \in M$ if $L(\bar{x}, M) \ne \emptyset$, and the *extended Abadie constraint qualification* (EACQ) holds at $\bar{x} \in M$ if $\Gamma^\star(\bar{x}, M) = L^\star(\bar{x}, M)$.

Note that EMFCQ coincides with MFCQ for finite differentiable optimization problems ([24]). Furthermore, it is obvious that EACQ coincides with the Abadie constraint qualification (ACQ, [1]) for finite differentiable optimization problems. Whereas in finite optimization MFCQ is stronger than ACQ, for *GSIP* this is not necessarily the case as an example in [33] shows (see, however, [31]). For extensions of the Karush–Kuhn–Tucker constraint qualification to *GSIP* see [12]. Explicit formulations of EMFCQ under different structural as-

sumptions on the lower-level problem $Q(\bar{x})$ can easily be obtained from the descriptions of $L(\bar{x}, M)$ given above.

Formulation

An important difference to finite or standard semi-infinite programming is that, for *GSIP*, there does not exist a single first-order necessary optimality condition, but the explicit formulation of the condition heavily depends on the structure of the lower-level problem. In fact, from the abstract primal first-order optimality condition (5) one can derive explicit dual conditions by replacing the contingent cone by an appropriate linearization cone and then cast the resulting conditions on certain infinite inequality systems in a dual formulation by means of theorems of the alternative, like, for example, the lemma of Gordan ([5,17]).

First-Order Optimality Conditions. In what follows, such optimality conditions are given for the structures discussed above. Recall that optimality conditions are trivial at interior points of M.

Theorem 1 ([16]) *Let $\bar{x} \in \partial M \cap M$ be a local minimizer of GSIP, at which the Reduction Ansatz holds. Moreover, let there exist a $d_0 \in \mathbb{R}^n$ such that*

$$D_x \mathcal{L}(\bar{x}, \bar{y}^i, \bar{\gamma}^i)\, d_0 < 0 \quad \text{for all} \quad 1 \le i \le p\,,$$

(i. e. EMFCQ holds at $\bar{x}$). Then there exist multipliers $\lambda_i \ge 0$, $i = 1, \ldots, p$, with $|\{1 \le i \le p | \lambda_i > 0\}| \le n$ such that

$$Df(\bar{x}) + \sum_{i=1}^{p} \lambda_i\, D_x \mathcal{L}(\bar{x}, \bar{y}^i, \bar{\gamma}^i) = 0\,.$$

Theorem 2 ([29,32]) *Let $\bar{x} \in \partial M \cap M$ be a local minimizer of GSIP, let the lower-level problems $Q(x)$, $x \in U$, be convex for some neighborhood U of $\bar{x}$, and let $Y(\bar{x})$ possess a Slater point. Then for each choice $\gamma \in KKT(\bar{x})$ such that there exists a d_0 with*

$$D_x \mathcal{L}(\bar{x}, y, \gamma)\, d_0 < 0 \quad \text{for all} \quad y \in Y_0(\bar{x})\,, \tag{6}$$

there exist $y^i \in Y_0(\bar{x})$ and multipliers $\lambda_i \ge 0$, $i = 1, \ldots, p$, with $|\{1 \le i \le p | \lambda_i > 0\}| \le n$, such that

$$Df(\bar{x}) + \sum_{i=1}^{p} \lambda_i\, D_x \mathcal{L}(\bar{x}, y^i, \gamma) = 0\,.$$

If EMFCQ holds at $\bar{x}$, then at least one such choice γ exists.

Theorem 3 ([27,32]) *Let $\bar{x} \in \partial M \cap M$ be a local minimizer of GSIP, and let MFCQ hold at all $y \in Y_0(\bar{x})$. Moreover, let there exist a $d_0 \in \mathbb{R}^n$ such that*

$$D_x \mathcal{L}(\bar{x}, y, \gamma)\, d_0 < 0 \qquad \text{for all} \quad \gamma \in KKT(\bar{x}, y)\,,\ \ y \in Y_0(\bar{x})\,,$$

(which is sufficient for EMFCQ at $\bar{x}$). Then there exist $y^i \in Y_0(\bar{x})$, $\gamma^i \in KKT(\bar{x}, y^i)$, and multipliers $\lambda_i \ge 0$, $i = 1, \ldots, p$, with $|\{1 \le i \le p | \lambda_i > 0\}| \le n$, such that

$$Df(\bar{x}) + \sum_{i=1}^{p} \lambda_i\, D_x \mathcal{L}(\bar{x}, y^i, \gamma^i) = 0\,.$$

Note that, under the convexity assumption on the lower-level problem, Theorem 2 provides a whole family of optimality conditions (parametrized by $\gamma \in KKT(\bar{x})$) and, thus, takes a possible disjunctive structure of M at $\bar{x}$ into account. On the other hand, in the absence of a nice lower-level structure, Theorem 3 yields a much weaker condition (which cannot be strengthened without further assumptions, as examples show).

First-order necessary optimality conditions for *GSIP* have been derived under several other structural assumptions and other theoretical approaches as well. In fact, without the assumption of EMFCQ, Fritz John-type optimality conditions can be derived ([32]), and there also exist optimality conditions without the assumption of *any* regularity condition, either in the upper- *or in the lower-level problem* ([20,31,32]). Conditions under other constraint qualifications are investigated in [12]. Furthermore, other theoretical approaches to optimality conditions are the linearization approach from [27] and conditions based on quasidifferentiable calculus ([7,27,30]). First-order *sufficient* optimality conditions for *GSIP* are examined in [32,34].

Second-Order Optimality Conditions

Second-order necessary and sufficient optimality conditions can be obtained in a straightforward manner under the Reduction Ansatz. One must simply write down the corresponding condition for the reduced finite optimization problem with the feasible set from

(3). Unfortunately, the Hessians of the optimal value functions $\varphi_i(x) = g(x, y^i(x)), 1 \le i \le p$, have a more complicated structure than the gradients from (2), due to the appearance of so-called *shift terms*. In fact, one has

$$D_x^2\varphi_i(\bar{x}) = D_x^2\mathcal{L}(\bar{x}, \bar{y}^i, \bar{\gamma}^i) - \begin{pmatrix} D_{yx}^2\mathcal{L}_i(\bar{x}, \bar{y}^i, \bar{\gamma}^i) \\ -D_x v_{L_0^i}(\bar{x}, \bar{y}^i) \end{pmatrix}^\top \cdot \begin{pmatrix} D_y^2\mathcal{L}_i(\bar{x}, \bar{y}^i, \bar{\gamma}^i) & -D_y^\top v_{L_0^i}(\bar{x}, \bar{y}^i) \\ -D_y v_{L_0^i}(\bar{x}, \bar{y}^i) & 0 \end{pmatrix}^{-1} \cdot \begin{pmatrix} D_{yx}^2\mathcal{L}_i(\bar{x}, \bar{y}^i, \bar{\gamma}^i) \\ -D_x v_{L_0^i}(\bar{x}, \bar{y}^i) \end{pmatrix},$$

where $D_x v_{L_0^i}$ stands for the matrix with rows $D_x v_\ell$, $\ell \in L_0^i := \{\ell \in L \mid v_\ell(\bar{x}, \bar{y}^i) = 0\}$.

Second-order conditions are also known under weaker assumptions, for example without the strict complementary slackness assumption of the Reduction Ansatz ([16]), and in connection with second-order epiregularity ([13,28], see also [4]). A second-order stability analysis for *GSIP* is given in [21].

Conclusions

First- and second-order optimality conditions are not only of theoretical importance, but also of high significance for the design of efficient numerical methods for *GSIP*. A review of such methods, including methods of feasible directions, KKT methods, and discretization methods, is given in [13].

See also

- ► Bilevel Optimization: Feasibility Test and Flexibility Index
- ► Parametric Optimization: Embeddings, Path Following and Singularities
- ► Second Order Constraint Qualifications

References

1. Abadie JM (1967) On the Kuhn–Tucker theorem. In: Abadie J (ed) Nonlinear Programming. Wiley, New York, pp 21–36
2. Bank B, Guddat J, Klatte D, Kummer B, Tammer K (1983) Non-linear Parametric Optimization. Birkhäuser, Basel
3. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear Programming. Theory and Algorithms. Wiley, New York
4. Bonnans JF, Shapiro A (2000) Perturbation Analysis of Optimization Problems. Springer, New York
5. Cheney EW (1966) Introduction to Approximation Theory. McGraw-Hill, New York
6. Danskin JM (1967) The Theory of Max-Min and Its Applications to Weapons Allocation Problems. Springer, New York
7. Demyanov VF, Vasilev LV (1985) Nondifferentiable Optimization, Optimization Software Inc. Publications Division, New York
8. Fiacco AV, McCormick GP (1968) Nonlinear Programming: Sequential Unconstrained Minimization Techniques. Wiley, New York
9. Gauvin J, Dubeau F (1982) Differential properties of the marginal function in mathematical programming. Math Programm Study 19:101–119
10. Goberna MA, López MA (1998) Linear Semi-infinite Optimization. Wiley, Chichester
11. Gol'stein EG (1972) Theory of Convex Programming, Translations of Mathematical Monographs, vol 36. American Mathematical Society, Providence, RI
12. Guerra Vázquez F, Rückmann J-J (2005) Extensions of the Kuhn–Tucker constraint qualification to generalized semi-infinite programming. SIAM J Optim 15(3):926–937
13. Guerra Vázquez F, Rückmann J-J, Stein O (2007) Still G Generalized semi-infinite programming: a tutorial. J Comput Appl Math, DOI: 10.1016/j.cam.2007.02.012
14. Hettich R, Jongen HT (1978) Semi-infinite programming: conditions of optimality and applications. In: Stoer J (ed) Optimization Techniques, Part 2, Lecture Notes in Control and Information Sciences, vol 7. Springer, Berlin, pp 1–11
15. Hettich R, Kortanek KO (1993) Semi-infinite programming: theory, methods, and applications. SIAM Rev 35:380–429
16. Hettich R, Still G (1995) Second order optimality conditions for generalized semi-infinite programming problems. Optim 34:195–211
17. Hettich R, Zencke P (1982) Numerische Methoden der Approximation und semi-infiniten Optimierung. Teubner, Stuttgart
18. Hogan WW (1973) Directional derivatives for extremal value functions with applications to the completely convex case. Oper Res 21:188–209
19. Jongen HT, Jonker P, Twilt F (1986) Critical sets in parametric optimization. Math Programm 34:333–353
20. Jongen HT, Rückmann J-J, Stein O (1998) Generalized semi-infinite optimization: a first order optimality condition and examples. Math Programm 83:145–158
21. Klatte D (1994) Stable local minimizers in semi-infinite optimization: regularity and second-order conditions. J Comp Appl Math 56:137–157
22. Laurent P-J (1972) Approximation et Optimisation. Hermann, Paris
23. Lempio F, Maurer H (1980) Differential stability in infinite-dimensional nonlinear programming. Appl Math Optim 6:139–152
24. Mangasarian OL, Fromovitz S (1967) The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. J Math Anal Appl 17:37–47

25. Polak E (1997) Optimization. Algorithms and Consistent Approximations. Springer, Berlin
26. Rockafellar RT (1984) Directional differentiability of the optimal value function in a nonlinear programming problem. Math Program Stud 21:213–226
27. Rückmann J-J, Shapiro A (1999) First-order optimality conditions in generalized semi-infinite programming. J Optim Theory Appl 101:677–691
28. Rückmann J-J, Shapiro A (2001) Second-order optimality conditions in generalized semi-infinite programming. Set-Valued Anal 9:169–186
29. Rückmann J-J, Stein O (2001) On convex lower level problems in generalized semi-infinite optimization. In: Goberna MA, López MA (eds) Semi-infinite Programming – Recent Advances. Kluwer, Dordrecht, pp 121–134
30. Shapiro A (1984) On optimality conditions in quasidifferentiable optimization. SIAM J Control Optim 22:610–617
31. Stein O (2001) First order optimality conditions for degenerate index sets in generalized semi-infinite programming. Math Oper Res 26:565–582
32. Stein O (2003) Bi-level Strategies in Semi-infinite Programming. Kluwer, Boston
33. Stein O (2004) On constraint qualifications in non-smooth optimization. J Optim Theory Appl 121:647–671
34. Stein O, Still G (2000) On optimality conditions for generalized semi-infinite programming problems. J Optim Theory Appl 104:443–458
35. Wetterling W (1970) Definitheitsbedingungen für relative Extrema bei Optimierungs- und Approximationsaufgaben. Numerische Mathematik 15:122–136

Generalized Total Least Squares

CHENGXIAN XU
Xian Jiaotong University, Xian, China

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Generalized nonlinear least squares; Gauss–Newton method; Separable optimization; separated Newton method

One important application of *nonlinear least squares* concerns with *data fitting* or *parameter estimations*. In ordinary least squares for data fitting, it is assumed that the errors in independent variables are either zero or negligible. Although there are situations in which errors in independent variables are zero or negligible, there exist many cases such as experiments and observations where this isnot so and use of the ordinary least squares may lead to bias in the estimated values of parameter vector and variance values [8]. *Generalized total least squares* problems are formulated from data fitting if errors in all variables are taken into account. Suppose that we have chosen a model function $y = \phi(x, t)$ to fit a set of data $y_1, \ldots, y_m$ sampled at m points $t_1, \ldots, t_m$, where $x \in \mathbf{R}^n$ is an adjustable parameter vector. The generalized total least squares problem concerning with this data fitting determines an optimal value of x and τ such that the function

$$f(x,\tau) = \frac{1}{2}\sum_{j=1}^{m}[w_j(\phi(x,\tau_j) - y_j)^2 + v_j(\tau_j - t_j)^2] = \frac{1}{2}[r^\top W r + e^\top V e]$$

is minimized, where $(\phi(x, \tau_j), \tau_j)$, $j = 1, \ldots, m$, are true but unknown values of pair (y, t), $W = \mathrm{diag}(w_1, \ldots, w_m)$, $V = \mathrm{diag}(v_1, \ldots, v_m)$, $w_j \geq 0$, $v_j \geq 0$, $j = 1, \ldots, m$, are weighting factors, r and e are two m-vectors with components $r_j = \phi(x, \tau_j) - y_j$, $e_j = \tau_j - t_j$, $j = 1, \ldots, m$, respectively.

Generalized total least squares problems can be solved by directly applying any method for ordinary nonlinear least squares or general minimization problems. Since these methods minimize the objective function $f(x, \tau)$ with respect to $(n+m)$ variables x and τ, and do not allow for the use of the special structureof the function, direct use of these methods will not be efficient. Assuming that the functions $r_j(x, \tau)$, $j = 1, \ldots, m$, hence the function $f(x, \tau)$ is twice continuously differentiable, the first and the second order derivatives of $f(x, \tau)$ are defined by

$$\nabla f = \begin{bmatrix} \nabla_x f \\ \nabla_\tau f \end{bmatrix} = \begin{bmatrix} AWr \\ Ve + DWr \end{bmatrix},$$

$$\nabla^2 f = \begin{pmatrix} \nabla^2_{xx} f & \nabla^2_{x\tau} f \\ \nabla^2_{\tau x} f & \nabla^2_{\tau\tau} f \end{pmatrix},$$

where

$$\nabla^2_{xx} f = AWA^\top + \sum_{j=1}^{m} w_j r_j \nabla^2_{xx} r_j,$$

$$\nabla^2_{x\tau} f = AWD + \sum_{j=1}^{m} w_j r_j \nabla^2_{x\tau} r_j,$$

$$\nabla^2_{\tau\tau} f = V + DWD + \sum_{j=1}^{m} w_j r_j \nabla^2_{\tau\tau} r_j,$$

$$A = \left[\nabla_x r_1 \cdots \nabla_x r_m\right],$$

$$D = \operatorname{diag}\left[\frac{\partial r_1}{\partial \tau_1} \cdots \frac{\partial r_m}{\partial \tau_m}\right].$$

The $(m\times m)$-matrix $\nabla^2_{\tau\tau} f$ is a diagonal matrix with diagonal elements

$$v_j + w_j \left(\frac{\partial r_j}{\partial \tau_j}\right)^2 + w_j r_j \frac{\partial^2 r_j}{\partial \tau_j^2}.$$

In developing algorithms for generalized total least squares, it is important to exploit the special structures of the function $f(x, \tau)$ and its derivatives, and in particular, the fact that variables x and τ can be treated separately. W.E. Demming [2], M. O'Neill, I.G. Sinclair and J. Smith [5], D.R. Powell and J.R. Macdonald [6] proposed *approximate Newton methods* for polynomial data fitting. These methods evaluate the second order derivatives $\nabla^2_{xx} f$ and $\nabla^2_{\tau\tau} f$ analytically or numerically, but ignore the mixed partial derivatives $\nabla^2_{x\tau} f$ and $\nabla^2_{\tau x} f$. When analytical derivatives are used, approximate Newton methods are not very efficient because of the unreasonable approximations. When derivatives are evaluated from difference quotient and compensations for ignoring mixed parts are made, the behavior of these methods is improved, because in this case the methods are equivalent to using one Newton step to separate problem variables and then the separated problem is solved using Newton method.

An optimization problem is *separable* if the optimization with respect to some of the variables is easier than with respect to others. Generalized total least squares problems are a kind of separable optimization problems. W.H. Southwell [7] uses the first order necessary condition to separate the vector x and the vector τ and then the separated problem is solvedusing Newton method. Gauss-Newton and quasi-Newton methods can also be used to solve the separated problems.

When Newton method is applied to solve a generalized total least squares problem, the solution of the Newton equation

$$\begin{bmatrix} \nabla^2_{xx} f & \nabla^2_{x\tau} f \\ \nabla^2_{\tau x} f & \nabla^2_{\tau\tau} f \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \tau \end{bmatrix} = - \begin{bmatrix} \nabla_x f \\ \nabla_\tau f \end{bmatrix}$$

gives a correction $(\delta x, \delta\tau)$ to (x, τ), that is,

$$x_+ = x + \delta x, \qquad \tau_+ = \tau + \delta\tau,$$

where x_+, τ_+ denote the new iterate. When the fitting function $\phi(x, t)$ is a polynomial in the form

$$\phi(x, t) = \sum_{i=1}^{n} x_i p_i(t),$$

where $p_i(t)$, $i = 1, \ldots, n$, are a set of *orthogonal polynomials*, then off-diagonal elements of the $(n \times n)$-matrix $\nabla^2_{xx} f$ are all zeros. Thus both the matrices $\nabla^2_{xx} f$ and $\nabla^2_{\tau\tau} f$ are diagonal. By assuming the elements of matrices $\nabla^2_{x\tau} f$ and $\nabla^2_{\tau x} f$ are negligible, approximate Newton methodsapproximate the Hessian matrix $\nabla^2 f$ by the simple diagonal matrix

$$\begin{bmatrix} \nabla^2_{xx} f & \\ & \nabla^2_{\tau\tau} f \end{bmatrix}.$$

Since $\nabla^2_{xx} f$ and $\nabla^2_{\tau\tau} f$ are diagonal, the solution δx and $\delta\tau$ can be easily given by

$$\delta x_i = -\frac{\sum_{j=1}^{m} w_j r_j p_i(\tau_j)}{\sum_{j=1}^{m} w_j p_i(\tau_j)^2}, \quad i = 1, \ldots, n,$$

$$\delta\tau_j = -\frac{v_j e_j + w_j \frac{\partial \phi(x,\tau_j)}{\partial \tau_j} r_j}{v_j + w_j \left(\frac{\partial \phi(x,\tau_j)}{\partial \tau_j}\right)^2 + w_j r_j \frac{\partial^2 \phi(x,\tau_j)}{\partial \tau_j^2}},$$

$$j = 1, \ldots, m.$$

Polynomials $p_i(t)$, $i = 1, \ldots, n$, orthogonal over a set of points τ_j, $j = 1, \ldots, m$, can be generated using the *recurrence relation*

$$p_1(t) = 1, \quad p_2(t) = t - \alpha_1,$$

$$p_i(t) = (t - \alpha_{i-1}) p_{i-1}(t) - \beta_{i-1} p_{i-2}(t),$$

$$i = 3, \ldots, n,$$

where

$$\alpha_{i-1} = \frac{\sum_{j=1}^{m} w_j \tau_j p_{i-1}(\tau_j)^2}{\sum_{j=1}^{m} w_j p_{i-1}(\tau_j)^2},$$

$$\beta_{i-1} = \frac{\sum_{j=1}^{m} w_j \tau_j p_{i-1}(\tau_j) p_{i-2}(\tau_j)}{\sum_{j=1}^{m} w_j p_{i-2}(\tau_j)^2}.$$

Approximate Newton methods begin iteration from the initial point $x^{(1)} = 0$ and $\tau_j^{(1)} = t_j, j = 1, \ldots, m$. At each iteration, the polynomials $p_i(t)$, $i = 1, \ldots, n$, orthogonal over the set of points $\tau_j^{(k)}$, $j = 1, \ldots, m$, are first calculated from the recurrence relation, then iteration

$$x^{(k+1)} = x^{(k)} + \delta x^{(k)}, \quad \tau^{(k+1)} = \tau^{(k)} + \delta\tau^{(k)}$$

is implemented to generate a new iterate. The process is repeated until convergence is reached. If the resulting fitting polynomial is required to express in the form of power series

$$\phi(x, t) = \sum_{i=1}^{n} c_i t^{i-1} = \sum_{i=1}^{n} x_i p_i(t),$$

the coefficients c_i can be calculated from

$$c_i = \sum_{k=i}^{n} x_k \sigma_{i+1k+1}, \quad 1 \le i \le n,$$

where

$$\sigma_{ik} = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{if } i > k \text{ or } i, k < 2, \end{cases}$$

$$\sigma_{i+1k+1} = \sigma_{ik} - \alpha_{k-1}\sigma_{i+1,k} - \beta_{k-1}\sigma_{i+1k-1}, \quad i < k.$$

Powell and Macdonald extended the method to more general case where $\phi(x, t)$ is a general nonlinear function of both the variables x and t. In this case, the $(n \times n)$-matrix $\nabla^2_{xx} f$ is no longer diagonal, and the correction δx needs the solution of the equations

$$\nabla^2_{xx} f \delta x = -\nabla_x f.$$

By taking account of the omitted parts of the mixed partial derivatives $\nabla^2_{x\tau} f$ and $\nabla^2_{\tau x} f$, they use 'unconventional formulas', rather than analytical derivatives or usual difference approximations, to approximate derivatives in $\nabla_x f$ and $\nabla^{\,2}_{xx} f$ so that the omission parts can be compensated to some degree. In fact, their approximate Newton method is equivalent, in some sense, to the separated Newton method.

Approximate Newton methods require evaluations of second order derivatives for problem functions. Ignoring all the second order terms in $\nabla^2_{xx} f$, $\nabla^2_{x\tau} f$, $\nabla^2_{\tau x} f$ and $\nabla^2_{\tau\tau} f$, an approximation to $\nabla^2 f$ is directly obtained from the first order derivatives of functions r_j and e_j, $j = 1, \ldots, m$. The iteration scheme $x^{(k+1)} = x^{(k)} + \delta x^{(k)}$, $\tau^{(k+1)} = \tau^{(k)} + \delta\tau^{(k)}$ with $\delta x^{(k)}$ and $\delta\tau^{(k)}$ given by

$$\begin{bmatrix} A_k W_k A_k^\top & A_k W D_k \\ D_k W A_k^\top & V + D_k W D_k \end{bmatrix} \begin{bmatrix} \delta x \\ \delta\tau \end{bmatrix} = -\begin{bmatrix} A_k W r^{(k)} \\ V e^{(k)} + D_k W r^{(k)} \end{bmatrix}$$

gives the *Gauss–Newton method* for generalized total least squares. Special structure of the system can be exploited to get savings in finding its solution. Define $P_k = V + D_k \mathrm{W} D_k$. From the bottom part of the system we have

$$\delta\tau = -P_k^{-1}\left[V e^{(k)} + D_k W r^{(k)} + D_k W A_k^\top \delta x\right].$$

Since P_k is a diagonal matrix, once δx is obtained, $\delta\tau$ can be directly obtained by substitutions. Substituting $\delta\tau$ into the top part of the system we obtain

$$\left[A_k W A_k^\top - A_k W D_k P_k^{-1} D_k W A_k^\top\right]\delta x = A_k W^{\frac{1}{2}} b^{(k)}$$

with

$$b^{(k)} = W^{\frac{1}{2}} \times \left[-r^{(k)} + D_k P_k^{-1}(V e^{(k)} + D_k W r^{(k)})\right].$$

This equation can be expressed as

$$A_k W^{\frac{1}{2}} U_k W^{\frac{1}{2}} A_k^\top \delta x = A_k W^{\frac{1}{2}} b^{(k)}$$

where $U_k = I - W^{1/2} D_k P_k^{-1} D_k W^{1/2}$ is a diagonal matrix with diagonal elements $v_j/[v_j + w_j(\partial r_j^{(k)}/\partial\tau_j)^2] > 0, j = 1, \ldots, m$. The solution $\delta x^{(k)}$ can be generated by first performing a *QR factorization* to the matrix $U_k^{1/2} W^{1/2} A_k^\top$

$$U_k^{\frac{1}{2}} W^{\frac{1}{2}} A_k^\top = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$$

and then back substitutions in

$$R\delta x = Q U_k^{-\frac{1}{2}} b^{(k)}.$$

The Gauss–Newton method is locally convergent and convergence behavior depends upon the closeness of the Gauss–Newton matrix to the true Hessian matrix

$\nabla^2 f$ at the solution. In order to introduce global convergence for Gauss–Newton method, line search technique or trust region strategy can be used. Let

$$J_k = \begin{bmatrix} A_k W^{\frac{1}{2}} & 0 \\ D_k W^{\frac{1}{2}} & V^{\frac{1}{2}} \end{bmatrix}$$

Then

$$\begin{bmatrix} A_k W A_k^\top & A_k W D_k \\ D_k W A_k^\top & V + D_k W D_k \end{bmatrix} = J_k J_k^\top$$

and the Gauss–Newton matrix is at least positive semidefinite, often positive definite, $(\delta x^{(k)}, \delta \tau^{(k)})$ is a descent direction of $f(x, \tau)$ at $(x^{(k)}), \tau^{(k)})$. A line searchalong the direction determines a steplength α_k satisfying some descent conditions and the new iteration point is

$$x^{(k+1)} = x^{(k)} + \alpha_k \delta x^{(k)},$$
$$\tau^{(k+1)} = \tau^{(k)} + \alpha_k \delta \tau^{(k+1)}.$$

P.T. Boggs, R.H. Byrd and R.B. Schnabel [1] use trust region technique in their modification of Gauss–Newton method for generalized total least squares problems. The modification is a generalization of the Levenberg–Marquardt method, in which the trust region subproblem

$$\begin{cases} \min & q_k(\delta z) = \left\| J_k^\top \delta z + h^{(k)} \right\|^2 \\ \text{s.t.} & \|\delta z\| \le \Delta_k \end{cases}$$

is solved, where Δ_k is the trust region radius,

$$z = \begin{pmatrix} x \\ \tau \end{pmatrix}, \qquad h^{(k)} = \begin{pmatrix} W^{\frac{1}{2}} r^{(k)} \\ V^{\frac{1}{2}} e^{(k)} \end{pmatrix}.$$

The solution, denoted by $\delta z(\mu)$, of the subproblem satisfies the system of equations

$$B_k \begin{bmatrix} \delta x \\ \delta \tau \end{bmatrix} = -\begin{bmatrix} A_k W r^{(k)} \\ V e^{(k)} + D_k W r^{(k)} \end{bmatrix},$$
$$\|\delta z(\mu)\| = \Delta_k,$$

$\mu > 0$, unless $\| \delta z(0) \| \le \Delta_k$, where B_k denotes the matrix

$$\begin{bmatrix} A_k W A_k^\top + \mu I & A_k W D_k \\ D_k W A_k^\top & V + D_k W D_k + \mu I \end{bmatrix}.$$

Let $\overline{P}_k = V + D_k W D_k + \mu I$. From the buttom part of the system, we get

$$\delta \tau = -\overline{P}_k^{-1} [V e^{(k)} + D_k W r^{(k)} + D_k W A_k^\top \delta x].$$

Substituting it into the top part we have

$$(A_k W^{\frac{1}{2}} \overline{U}_k W^{\frac{1}{2}} A_k^\top + \mu I) \delta x = A_k W^{\frac{1}{2}} \overline{b}^{(k)},$$
$$\overline{U}_k = I - W^{\frac{1}{2}} D_k \overline{P}_k^{-1} D_k W^{\frac{1}{2}},$$
$$\overline{b}^{(k)} = W^{\frac{1}{2}} \times [-r^{(k)} + D_k \overline{P}_k^{-1} (V e^{(k)} + D_k W r^{(k)})].$$

Since this system is the *normal equation* of the linear least squares problem

$$\min \left\| \begin{bmatrix} \overline{U}_k^{\frac{1}{2}} W^{\frac{1}{2}} A_k^\top \\ \mu^{\frac{1}{2}} I \end{bmatrix} \delta x + \begin{bmatrix} \overline{U}_k^{\frac{1}{2}} \overline{b}^{(k)} \\ 0 \end{bmatrix} \right\|,$$

the solution $\delta x^{(k)}$ can be obtained by performing a QR factorization to the matrix $\overline{U}_k^{\frac{1}{2}} W^{\frac{1}{2}} A_k^\top$, a sequence of plane rotations to eliminate $\mu^{1/2} I$ and back substitutions.

For a given value $\mu^{(\ell)}$, $\delta x(\mu^{(\ell)})$ is obtained from the solution of the system and then $\delta\, \tau(\mu^{(\ell)})$ from substitution. If

$$\left| \phi(\mu^{(\ell)}) \right| = \left| \left\| \delta z(\mu^{(\ell)}) \right\| - \Delta_k \right| \le \rho \Delta_k$$

is satisfied, $\delta z(\mu^{(\ell)})$ is accepted as an approximate solution of the trust region subproblem where $\rho \in (0, 1)$is a preset tolerance. Otherwise, $\mu^{(\ell)}$ is updated to give a new value $\mu^{(\ell+1)}$ and a solution $\delta z(\mu^{(\ell+1)})$ is recomputed from the system. *Moré's updating formula* [4]

$$\mu^{(\ell+1)} = \mu^{(\ell)} - \frac{\phi(\mu^{(\ell)})}{\nabla \phi(\mu^{(\ell)})} \frac{\left\| \delta z(\mu^{(\ell)}) \right\|}{\Delta_k}$$

can be used to generate $\mu^{(\ell+1)}$, where $\nabla \phi(\mu^{(\ell)})$ is evaluated from difference approximation

$$\nabla \phi(\mu^{(\ell)}) = \frac{\phi(\mu^{(\ell)}) - \phi(\mu^{(\ell-1)})}{\mu^{(\ell)} - \mu^{(\ell-1)}}.$$

For generalized total least squares problems, the parameter vector x and the variable vector τ can be treated separately. The first order necessary condition for a point to be a solution of the problem can be used to eliminate the τ dependence in the function $f(x, \tau)$. Consider the system of equations

$$\nabla_\tau f = Ve + DWr = 0.$$

These contain m nonlinear equations with m unknowns, each of which only contains one unknown τ_j

for fixed value of x

$$v_j(\tau_j - t_j) + w_j(\phi(x, \tau_j) - y_j)\frac{\partial \phi(x, \tau_j)}{\partial \tau_j} = 0,$$
$$j = 1, \ldots, m.$$

When these equations can be algebraically solved to give an explicit solution expression $\tau(x)$, substitution it into the function $f(x, \tau)$ allows the parameter vector x to be determined by directly using any conventional method to minimize the function $f(x, \tau(x))$ which now is a function of the parameter vector x. However, in most cases, it is impossible or difficult to get an explicit form of the solution $\tau(x)$ and each equation mustbe solved numerically for each given value of x by minimizing the functions

$$\psi(x, \tau_j) = \frac{1}{2}[w_j(\phi(x, \tau_j) - y_j)^2 + v_j(\tau_j - t_j)^2],$$
$$j = 1, \ldots, m,$$

to get an approximate solution, $\overline{\tau}(x)$ say, to the solution $\tau(x)$ so that the values of function $f(x, \tau(x))$ and its derivatives with respect to x can be evaluated from the values x and $\overline{\tau}(x)$.

Assume that $\nabla^2_{\tau\tau} f(x^*, \tau^*)$ is positive definite, then it follows from the *implicit function theorem* [3] that there exist open neighborhoods $N(x^*)$, $N(\tau^*)$ of x^*, τ^* such that for any $x \in N(x^*)$, a unique τ satisfying the system exists in $N(\tau^*)$, this being the vector $\tau(x)$. Furthermore, $\tau(x)$ is continuously differentiable and $\nabla^2_{\tau\tau} f(x, \tau(x))$ is positive definite for all $x \in N(x^*)$. Substituting $\tau(x)$ into the function $f(x, \tau)$ we get a separable minimization problem

$$\min f(x, \tau(x)),$$

which is defined only in terms of x and reduces the problem dimension from $m + n$ to n. The separation is particularly efficient since in most cases, m is very large. Using the *chain rule*, the differentiability of $\tau(x)$ and the fact that $\nabla_\tau f = 0$ we get derivatives of the function $f(x, \tau(x))$

$$\begin{aligned} g(x) &= \nabla_x f + \nabla_x \tau \nabla_\tau f = \nabla_x f, \\ G(x) &= \nabla^2_{xx} f + \nabla^2_{x\tau} f \nabla_x \tau \\ &= \nabla^2_{xx} f - \nabla^2_{x\tau} f [\nabla^2_{\tau\tau} f]^{-1} \nabla^2_{\tau x} f. \end{aligned}$$

Since the positive definiteness of the matrix $G(x)$ is implied by that of the matrix $\nabla^2 f$, if $\nabla^2 f$ is positive definite at the solution (x^*, τ^*), the matrix $G(x^*)$ is positive definite, too.

The *separated Newton method* minimizes the function $f(x, \tau(x))$ using Newton iteration

$$G_k \delta x^{(k)} = -g^{(k)}, \quad x^{(k+1)} = x^{(k)} + \delta x^{(k)}$$

to generate a sequence $\{x^{(k)}\}$, where G_k and $g^{(k)}$ are evaluated at $x^{(k)}$ and $\tau(x^{(k)})$. $\tau(x^{(k)})$ is an approximate solution of the system $\nabla_\tau f = 0$ obtained using Newton iteration

$$\tau_j^{(s+1)} = \tau_j^{(s)} - \frac{\psi'(x^{(k)}, \tau_j^{(s)})}{\psi''(x^{(k)}, \tau_j^{(s)})},$$
$$s = 1, 2, \ldots, \quad j = 1, \ldots, m.$$

When

$$\left| \tau_j^{(s+1)} - \tau_j^{(s)} \right| \le \epsilon,$$

$\tau_j^{(s+1)}$ is accepted as $\tau_j(x^{(k)})$ where $\epsilon > 0$ isa preset small constant. The values t_j and $\tau_j(x^{(k-1)})$, $j = 1, \ldots, m$, can be used as starting values of these iterations for $k = 1$ and $k \ge 2$, respectively.

A careful observation shows that the difference between the Powell–Macdonald method and the separated Newton method is that for given value $x^{(k)}$, the former carries out only one Newton iteration for the system $\nabla_\tau f = 0$ while the later one solves the system quite exactly by repeated doing the iteration.

The separated Newton method still requires the evaluation of secondorder derivatives. Ignoring second order terms in all derivatives $\nabla^2_{xx} f$, $\nabla^2_{x\tau} f$, $\nabla^2_{\tau x} f$ and $\nabla^2_{\tau\tau} f$, we get an approximation to G

$$M_k = A_k W^{\frac{1}{2}} U_k W^{\frac{1}{2}} A_k^\top,$$
$$U_k = (I + V^{-1} D_k W D_k)^{-1}.$$

Then the iteration

$$M_k \delta x^{(k)} = -g^{(k)}, \quad x^{(k+1)} = x^{(k)} + \delta x^{(k)}$$

is the separated Gauss–Newton method [8]. The property that the convergence of Gauss–Newton method for ordinary least squares depends on the closeness of the Gauss–Newton matrix to true Hessian matrix is applicable to the separated Gauss–Newton method. If $M(x^*) = G(x^*)$, the method is locally convergent and

rate of convergence is quadratic. If $M(x^*) \neq G(x^*)$, the method may not converge and if it converges, the rate is at best linear. In order to force global convergence, line search or trust region techniques can be incorporated.

For *large residual problems*, the Gauss–Newton matrix M is not a good approximation to G and quasi-Newton updates can be used to generate better approximations. When quasi-Newton updates, for example BFGS update, are used, the separated problem is regarded as a general minimization problem, the special structure of the problem function is not exploited and approximations are not directly obtained from the first order derivatives. The vectors $\delta^{(k)}$ and $\gamma^{(k)}$ used in updating formulas can be defined by

$$\delta^{(k)} = \delta x^{(k)} = x^{(k+1)} - x^{(k)},$$
$$\gamma^{(k)} = g(x^{(k+1)}, \tau(x^{(k+1)})) - g(x^{(k)}, \tau(x^{(k)}))$$

Alternative definitions for $\gamma(k)$ can be derived by using thespecial structure of the derivatives. Two common used definitions for $\gamma^{(k)}$ are

$$\gamma^{(k)} = A_{k+1} W A_{k+1}^{\top} \delta x^{(k)} + A_{k+1} W D_{k+1} \delta \tau^{(k)} + (A_{k+1} - A_k) r^{(k+1)},$$
$$\gamma^{(k)} = A_{k+1} W (r^{(k+1)} - r^{(k)}) + (A_{k+1} - A_k) W r^{(k+1)},$$

where $\delta\tau^{(k)} = \tau(x(k+1)) - \tau(x^{(k)})$. Numerical experiments favors the last definition of $\gamma^{(k)}$ [9].

Based on the separated Gauss–Newton method and the separated BFGS method, separated hybrid method is a simple generalization of the hybrid method for ordinary nonlinear least squares problems, where a test [9] is derived to determine what step should be chosen at each iteration. When the test chooses the Gauss–Newton step, the approximation B_k to G_k is set to the Gauss– Newton matrix M_k and when the test chooses the BFGS step, the matrix B_k is obtained from B_{k-1} using BFGS updating formula.

When separated methods are used to solve generalized total least squares problems, computational savings can be obtained if we initially ignore errors in t_j, $j = 1, \ldots, m$, and just solve an ordinary nonlinear least squares problem. Whenreasonable reduction in the objective function has been achieved, errors in all variables are then considered and separated methods are applied. This modification of any separated method is effective in solving generalized total least squares problems.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- ABS Algorithms for Optimization
- Gauss–Newton Method: Least Squares, Relation to Newton's Method
- Least Squares Orthogonal Polynomials
- Least Squares Problems
- Nonlinear Least Squares: Newton-type Methods
- Nonlinear Least Squares Problems
- Nonlinear Least Squares: Trust Region Methods

References

1. Boggs PT, Byrd RH, Schnabel RB (1987) A stable and efficient algorithm for nonlinear orthogonal distance regression. SIAM J Sci Statist Comput 8:1052–1078
2. Demming WE (1943) Statistics adjustment of data. Wiley, New York
3. Hestens MR (1966) Calculus of variations and optimal control problems. New York Wiley, New York
4. Moré JJ (1977) The Levenberg-Marquardt algorithm: Implementation and theory. In: Watson GA (ed) Numerical Analysis, Dundee. Lecture Notes Math. Springer, Berlin, pp 105–116
5. O'Neill M, Sinclair IG, Smith J (1969) Polynomial curve fitting when abscisses and ordinates are both subject to error. Comput J 12:52–56
6. Powell DR, Macdonld JR (1972) A rapidly convergent iterative method for the solution of the generalizednonlinear least squares problem. Comput J 15:148–155
7. Southwell WH (1975) Fitting data to nonlinear functions with uncertainties in all measurement variables. Comput J 19:67–73
8. Watson GA (1985) The solution of generalized least squares problems. Internat Ser Numer Math 75:388–400
9. Xu CX (1987) Hybrid methods for nonlinear least squares and related problems. PhD Thesis Univ. Dundee

Generalized Variational Inequalities: A Brief Review

Barbara Panicucci
Department of Applied Mathematics, University of Pisa, Pisa, Italy

MSC2000: 49J53, 90C30

Article Outline

Keywords and Phrases
Introduction
Problem Formulation and Framework
Existence and Uniqueness
Existence of Solutions: Bounded Domain
Existence of Solutions: Unbounded Domain
GVI and Related Problems
GVI and Fixed-Point Problems
GVI and Optimization Problems
GVI and Complementarity Problems
References

Keywords and Phrases

Generalized variational inequality; Optimization problem; Gap function

Introduction

The theory as well the applications of variational inequalities (VIs) and the nonlinear complementarity problem (NCP) have proved to be a very powerful tool for studying a wide range of problems arising in mechanics, physics, optimization, and applied sciences. A survey on the developments of VI and NCP is in [7]. In recent years, considerable interest has been shown in developing various extensions and generalizations of the VI problem. An important class of such generalizations, introduced in [2], is the so-called generalized variational inequality (GVI). This class has many important and significant applications in various fields such as mathematical physics and control theory, economics, and transportation equilibrium (see, e. g., [1,11]). For example, it is known that the traffic equilibrium problem can be formulated as a VI when the travel cost between any two given nodes for a given flow is fixed [4]. However, the traffic conditions may vary and the travel cost between two given nodes may not be fixed, but within a cost interval. In this case the corresponding problem can be formulated as a GVI. Moreover, GVI provides a unifying framework for many general problems such us fixed-point, optimization, and complementarity problems. In what follows we give an overview of recent developments concerning the issue of existence of a solution and equivalent reformulations.

Problem Formulation and Framework

In its general form, the GVI problem can be stated as follows:

find $x^* \in X$ and $u^* \in F(x^*)$ such that

$$\langle u^*, y - x^* \rangle \geq 0 \quad \forall\, y \in X\,,$$

where

- $\langle \cdot, \cdot \rangle$ denotes the usual inner product in $\mathbb{R}^n$,
- $X \subseteq \mathbb{R}^n$ is a nonempty closed and convex set,
- $\mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is a set-valued map, i. e., an operator that associates with each $x \in \mathbb{R}^n$ a set $F(x) \subseteq \mathbb{R}^n$.

If F is a single valued function, then the GVI problem reduces to the classical VI, which is to find $x^* \in X$ such that

$$\langle F(x^*), y - x^* \rangle \geq 0 \quad \forall\, y \in X\,.$$

In connection with the set-valued map $F\colon \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ a few definitions need to be recalled. First, F is characterized by its graph:

$$\text{graph}\,(F) = \{(x, u) \in \mathbb{R}^n \times \mathbb{R}^n\colon\ u \in F(x)\}\,.$$

The image of X under F is

$$F(X) = \bigcup_{x \in X} F(x)\,,$$

the inverse of F is defined by

$$F^{-1}(u) = \{x\colon\ u \in F(x)\},$$

and the domain of F is the set

$$\text{dom}\,(F) = \{x \in \mathbb{R}^n\colon F(x) \neq \emptyset\}\,.$$

Throughout we assume that $\text{dom}\,(F) \supseteq X$. Over the past two decades, most effort has been concentrated on the question of the existence of solutions to GVI problems. The study of the existence of solutions of GVI involves several continuity properties of set-valued maps. We recall these conditions in the sequel.

- A set-valued map $F\colon \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is said to be upper semicontinuous (u.s.c.) at $x \in \mathbb{R}^n$ if for each open set $V \supseteq F(x)$ there exists a neighborhood U of x such that $F(U) \subseteq V$; F is u.s.c. on a set $X \subseteq \mathbb{R}^n$ if it is u.s.c. at every point in X.
- A set-valued map $F\colon \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is upper hemicontinuous on $X \subseteq \mathbb{R}^n$, if its restriction to line segments of X is upper semicontinuous.

The study of the existence of solutions of GVI involves also some monotonicity-type properties for set-valued maps. In what follows we recall the definitions.

(M1) F is quasimonotone on X if, for every pair of distinct points $x, y \in X$ and every $u \in F(x)$, $v \in F(y)$, we have:

$$\langle v, x - y\rangle > 0 \Longrightarrow \langle u, x - y\rangle \geq 0 \,.$$

(M2) F is properly quasimonotone on X if, for any $x^1, \ldots, x^n \in X$ and any $\lambda_1, \ldots, \lambda_n > 0$ with $\sum_{i=1}^n \lambda_i = 1$, there exists $j \in \{1, \ldots, n\}$ such that for all $u^j \in F(x^j)$ and $x = \sum_{i=1}^n \lambda_i x^i$, we have:

$$\langle u^j, x - x^j\rangle \leq 0 \,.$$

(M3) F is pseudomonotone on X if, for every pair of distinct points $x, y \in X$ and every $u \in F(x), v \in F(y)$, we have:

$$\langle v, x - y\rangle \geq 0 \Longrightarrow \langle u, x - y\rangle \geq 0 \,.$$

(M4) F is monotone on X if, for every pair of distinct points $x, y \in X$ and every $u \in F(x), v \in F(y)$, we have:

$$\langle u - v, x - y\rangle \geq 0 \,.$$

(M5) F is strictly monotone on X if, for every pair of distinct points $x, y \in X$ and every $u \in F(x), v \in F(y)$, we have:

$$\langle u - v, x - y\rangle > 0 \,.$$

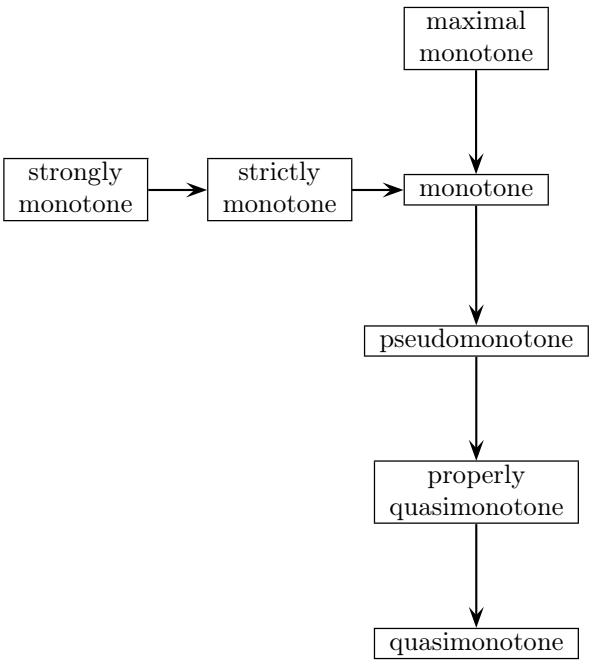
(M6) F is strongly monotone on X with constant $\beta > 0$ if, for every pair of distinct points $x, y \in X$ and every $u \in F(x), v \in F(y)$, we have:

$$\langle u - v, x - y\rangle \geq \beta \|x - y\|^2 \,,$$

where $\|\cdot\|$ denotes the classical euclidean norm.

(M7) F is maximal monotone on X if it is monotone on X and its graph is not properly contained in the graph of any other monotone operator on X.

The relationships among these kinds of monotonicity are represented in Fig. 1.



Generalized Variational Inequalities: A Brief Review, Figure 1
Relationships among generalized monotonicity conditions

Existence and Uniqueness

In recent years the existence of solutions to GVIs has been investigated extensively. In what follows we provide some of the most fundamental results. The basic result on the existence of a solution to the GVI problem requires the set X to be compact and convex and the map F to be u.s.c. From this basic result many others can be derived by replacing the compactness of X with additional coercivity conditions on F.

Existence of Solutions: Bounded Domain

This section presents some existence results for solutions of GVI in the case of a compact domain. The following existence theorem exploits the formulation of GVI as a fixed-point problem.

Theorem 1 ([8]) *If X is compact and F is u.s.c. on X with compact and convex values, then GVI has a solution.*

Theorem 2 ([12]) *If X is compact and F is upper hemicontinuous and properly quasimonotone on X with compact and convex values, then GVI has a solution.*

Existence of Solutions: Unbounded Domain

The existence of solutions of GVI on unbounded domains is guaranteed by the same conditions as for bounded domains, together with a coercivity condition. In the literature various coercivity conditions have been considered. In particular (see [5]):

(C1)

$$\begin{aligned} &\exists\, R > 0, \quad \forall x \in X \setminus X_R, \quad \forall u \in F(x)\,, \\ &\exists\, y \in X_R\colon \quad \langle u, y - x\rangle < 0\,; \end{aligned}$$

(C2)

$$\begin{aligned} &\exists\, R > 0, \quad \forall\, x \in X \setminus X_R, \quad \exists\, y \in X_R\,, \\ &\forall\, u \in F(x)\colon \quad \langle u, y - x\rangle < 0\,; \end{aligned}$$

(C3)

$$\begin{aligned} &\exists\, R > 0, \quad \forall\, x \in X \setminus X_R, \quad \exists\, y \in X_R\,, \\ &\exists\, v \in F(y)\colon \quad \langle v, y - x\rangle < 0\,; \end{aligned}$$

(C4)

$$X_\infty \cap (F(X))^- = \{0\}\,,$$

where

$$X_R = \{x \in X\colon \|x\| \le R\}$$

and

$$(F(X))^- = \{d \in \mathbb{R}^n\colon \langle u, d\rangle \le 0, \forall u \in F(X)\}$$

is the polar cone of $F(X)$. Further, the recession cone X_∞, for X closed and convex, is defined by

$$X_\infty = \{d \in \mathbb{R}^n\colon x + t\,d \in X,\ \forall\, t \ge 0,\ x \in X\}\,.$$

Some basic relationships among these coercivity conditions are summarized in the following result.

Theorem 3 ([5])

- *(C2) $\Longrightarrow$ (C1).*
- *If F has convex values, then (C2) and (C1) are equivalent.*
- *If F is pseudomonotone on X, then (C3) $\Longrightarrow$ (C2).*
- *(C4) $\Longrightarrow$ (C3).*
- *If F is upper hemicontinuous and pseudomonotone on X, then (C2), (C3) and (C4) are equivalent.*
- *If F has convex values and it is upper hemicontinuous and pseudomonotone on X, then (C1), (C2), (C3), and (C4) are equivalent.*

The coercivity conditions allow us to exhibit a sufficiently large ball intersecting with X such that no point outside this ball is a solution of the GVI; then one can establish the existence of a solution stated below.

Theorem 4 ([5]) *If F is upper hemicontinuous and pseudomonotone on X with compact and convex values, then the following statements are equivalent:*

- *GVI has a nonempty and compact solution set.*
- *(C1) holds;*
- *(C2) holds.*
- *(C3) holds.*
- *(C4) holds.*

In what follows we state an existence theorem for which we require neither the upper semicontinuity of F, nor the compactness, nor the convexity of $F(x)$, but we need the maximal monotonicity of F.

Theorem 5 ([15]) *Assume that F is maximal monotone on $\mathbb{R}^n$. Then the solution set of GVI is nonempty and compact if and only if (C4) holds.*

In general, GVI can have more than one solution. The following theorem gives conditions under which GVI can have at most one solution.

Theorem 6

- *If F is strictly monotone on X, then GVI has at most one solution.*
- *If F is u.s.c., strongly monotone on X, and has nonempty convex and compact values, then GVI has a unique solution.*

GVI and Related Problems

As stated, the theory of GVI is a powerful unifying methodology that contains as special cases several well-known problems such as fixed-point, optimization, and complementarity problems. In what follows we describe these equivalent formulations of the GVI problem. Such formulations can be very beneficial for both analytical and computational purposes. Indeed we can apply classic results of these problems to treat the GVI.

GVI and Fixed-Point Problems

In what follows we exploit the formulation of GVI as a fixed-point problem. We recall that x^* is a fixed point of the set-valued map $F\colon X \rightrightarrows \mathbb{R}^n$ if

$$x^* \in X \quad \text{and} \quad x^* \in F(x^*)\,.$$

The fixed-point reformulation is very relevant for the GVI problem. Indeed we can apply Kakutani's fixed-point theorem, which is instrumental for proving the existence result on a bounded domain. We define the following set-valued map:

$$\begin{aligned}\Theta\colon X \times \operatorname{conv}(F(X)) &\rightrightarrows X \times \operatorname{conv}(F(X))\\ (x,u) &\mapsto \Psi(u) \times F(x)\,,\end{aligned}$$

where $\Psi(u) = \arg\min_{x\in X}\langle u, x\rangle$ is the set of constrained minimizers of the map $\langle u, x\rangle$ on X and $\operatorname{conv}(F(X))$ denotes the convex hull of $F(X)$. Assuming that X is compact, $\Psi(u)$ results in being nonempty. It easy to see that the problem of finding a fixed point (x^*, u^*) of Θ, i. e.,

$$x^* \in K, \quad u^* \in F(x^*), \quad x^* \in \arg\min_{x\in K}\langle u^*, x\rangle\,,$$

is equivalent to GVI.

It is worth noting that the GVI problem can also be formulated as an inclusion as follows:

$$\text{find } x^* \in K \text{ such that } 0 \in F(x^*) + N_K(x^*)\,,$$

i. e., finding a zero of the set-valued map $F + N_K$ in the domain X, where the normal cone $N_X(x)$ to the set X at point $x \in X$ is given by:

$$N_X(x) = \{d \in \mathbb{R}^n\colon \langle d, y - x\rangle \le 0 \quad \forall\, y \in X\}\,.$$

GVI and Optimization Problems

Let us consider the constrained optimization problem:

$$\begin{cases}\min f(x)\\ x \in X\,,\end{cases} \tag{1}$$

where

- X is a closed and convex subset of $\mathbb{R}^n$,
- The objective function f is defined on an open neighborhood of X, denoted Ω.

It is well known that if f is continuously differentiable, then the classical VI with $F = \nabla f$ is a necessary optimality condition for (1). The VI gives also a sufficient condition if f is pseudoconvex on X, i. e.,

$$f(x) > f(y) \Longrightarrow \langle \nabla f(x), y - x\rangle < 0\,,$$

for all $x, y \in X$.

Therefore, if f is continuously differentiable and pseudoconvex on X, the VI with $F = \nabla f$ is equivalent to the optimization problem (1). In what follows we extend these results in terms of GVI when $f\colon \Omega \to \mathbb{R}$ is a locally Lipschitz continuous function, that is, for each point $x \in \Omega$ there exists a neighborhood U of x such that f is Lipschitz continuous on U. To this end we recall some basic facts about Clarke calculus for a locally Lipschitz continuous function, see [3]. The Clarke's generalized derivative of f at x in the direction v, denoted by $f^0(x;v)$, is given by

$$f^0(x;v) = \limsup_{\substack{y\to x\\ t\downarrow 0}} \frac{f(y + t\,v) - f(y)}{t}\,.$$

The generalized gradient of f at x, denoted by $\partial f(x)$, is defined as follows:

$$\partial f(x) = \{\xi \in \mathbb{R}^n\colon\ \langle \xi, v\rangle \le f^0(x;v) \quad \forall\, v \in \mathbb{R}^n\}\,.$$

A generalized derivative can be obtained from the generalized gradient:

$$f^0(x;v) = \max\{\langle \xi, v\rangle : \xi \in \partial f(x)\}\,.$$

We can extend the definition of pseudoconvexity for a locally Lipschitz continuous function $f\colon \Omega \to \mathbb{R}$, [16]: f is pseudoconvex on Ω if, for all $x, y \in \Omega$, there exists $\xi \in \partial f(x)$ such that

$$\langle \xi, y - x\rangle \ge 0 \quad \Longrightarrow \quad f(x) \le f(y)\,.$$

Let us now consider the GVI with Clarke gradient operator $F = \partial f$. We can state the following result.

Theorem 7 ([3]) *A GVI with $F = \partial f$ provides necessary optimality conditions for problem (1).*

In general, a GVI does not give sufficient optimality conditions. However, as shown in [16], when f is pseudoconvex on Ω, the GVI gives sufficient optimality conditions too. Consequently, as for the single-valued

case, if f is pseudoconvex on Ω, a GVI with $F = \partial f$ is equivalent to the optimization problem (1). The above discussion focused on the GVI with gradient operator; however, an arbitrary set-valued map, in general, is not a gradient map. A powerful tool in dealing with the GVI problem by way of its equivalent optimization reformulation is given by the so-called gap functions. Specifically, we say that a function $\varphi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a gap function for GVI if

- $\varphi(x, u) \geq 0$ for all $(x, u) \in \text{graph}\,(F)$,
- x^* is a solution of GVI if and only if $x^* \in X$ and there exists $u^* \in F(x^*)$ such that $\varphi(x^*, u^*) = 0$.

Hence, the GVI problem can be rewritten as the following constrained optimization problem:

$$\begin{cases} \min\ \varphi(x, u) \\ (x, u) \in \text{graph}\,(F)\,. \end{cases}$$

An example of a gap function, proposed in [6], is:

$$\varphi(x, u) = \sup_{y \in X} \langle u, x - y \rangle, \quad (x, u) \in \mathbb{R}^n \times \mathbb{R}^n\,. \quad (2)$$

The function $\varphi(x, \cdot)$ is convex and closed for every fixed $x \in \mathbb{R}^n$ and $\varphi(\cdot, u)$ is affine for every fixed $u \in \mathbb{R}^n$ (see [6]). It is worth noting that ϕ represents a duality gap in the Mosco duality scheme [14] for GVI. Let us consider this more general GVI problem: find $x^* \in \mathbb{R}^n$ and $u^* \in F(x^*)$ such that

$$\langle u^*, x - x^* \rangle \geq \phi(x^*) - \phi(x) \quad \forall\, x \in \mathbb{R}^n\,, \qquad (3)$$

where $\phi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a proper, lower semicontinuous convex function. The dual problem of (3) is defined as: find $v^* \in \mathbb{R}^n$ and $y^* \in -F^{-1}(-v^*)$ such that

$$\langle y^*, v - v^* \rangle \geq \phi^*(v^*) - \phi^*(y) \quad \forall\, v \in \mathbb{R}^n\,,$$

where $\phi^*(v) = \sup_{x \in \mathbb{R}^n} \{\langle v, x \rangle - \phi(x)\}$ is the Fenchel conjugate of φ.

Theorem 8 ([15]) *The gap function (2) measures the duality gap of Mosco's duality scheme:*

$$\phi(x) + \phi^*(-u) + \langle u, x \rangle = \begin{cases} \varphi(x, u) & \text{if } x \in X \\ +\infty & \text{otherwise.} \end{cases}$$

The gap function ϕ is not differentiable in general. Moreover, when graph (F) is unbounded, it is in general not finite valued. These drawbacks can be avoided by using a regularized gap function. Let us consider

$$\varphi_G(x, u) = \max_{y \in X} \left[\langle u, x - y \rangle - \frac{1}{2} \|x - y\|_G^2 \right],$$

where $(x, u) \in \mathbb{R}^n \times \mathbb{R}^n$, G is a symmetric positive definite matrix, and $\|\cdot\|_G$ is the norm in $\mathbb{R}^n$ defined by $\|x\|_G = \sqrt{\langle x, Gx \rangle}$. This function, introduced in [6] for generalized quasivariational inequalities, i. e., GVIs where set X depends on solution x, is a gap function for GVI and is called a regularized gap function. Since

$$\psi_G(x, u, y) = \langle u, x - y \rangle - \frac{1}{2} \|x - y\|_G^2$$

is strongly concave with respect to y, there is a unique maximizer over X denoted by $y(x, u)$. If we denote the projection operator onto set X with respect to the norm $\|\cdot\|_G$ by $\Pi_{X,G}(\cdot)$, it is easy to check that this maximizer is

$$y(x, u) = \Pi_{X,G}(x - G^{-1} u)\,.$$

Therefore, the regularized gap function

$$\varphi_G(x, u) = \langle u, x - y(x, u) \rangle - \frac{1}{2} \|x - y(x, u)\|_G^2$$

is finite valued everywhere. Moreover, the regularized gap function is continuously differentiable, and its gradient is given by

$$\nabla_x \varphi_G(x, u) = u + G\,[y(x, u) - x]\,,$$
$$\nabla_u \varphi_G(x, u) = x - y(x, u)\,.$$

Therefore, using the regularized gap function we obtain an equivalent differentiable optimization reformulation of the GVI problem. Gap functions can be used in the design of numerical algorithms for solving the GVI.

GVI and Complementarity Problems

It is well known that, when X is a closed convex cone and $F : X \to \mathbb{R}^n$, the VI problem is equivalent to the NCP problem, which consists in finding $x^* \in X$ such that

$$F(x^*) \in X^* \quad \text{and} \quad \langle F(x^*), x^* \rangle = 0\,,$$

where

$$X^* = \{d \in \mathbb{R}^n : \langle u, d \rangle \geq 0, \forall u \in X\}$$

is the negative polar cone of X. Such a relationship is preserved in the GVI problems. First, let us consider an extension of the NCP problem, see [17], that can be defined as follows.

Let X be a closed convex cone of $\mathbb{R}^n$ and F a set-valued map. The generalized complementarity problem (GCP) is to find $x^* \in X$ such that there exists $u^* \in F(x^*)$ satisfying the following properties:

$$u^* \in X^* \quad \text{and} \quad \langle u^*, x^* \rangle = 0 .$$

As in the single-valued case, both problems GVI and GCP have the same solution set if the underlying set X is a closed convex cone.

References

1. Aubin JP (1984) L'Analyse Non Linéaire et Ses Motivations Economiques. Masson, Paris
2. Browder FE (1965) Multivalued Monotone Nonlinear Mappings and Duality Mappings in Banach Spaces. Trans Am Math Soc 71:780–785
3. Clarke FH (1990) Optimization and nonsmooth analysis, vol 5 of Classics in Applied Mathematics. SIAM, Philadelphia
4. Dafermos S (1980) Traffic Equilibrium and Variational Inequalities. Transp Sci 14:42–54
5. Daniilidis A, Hadjisavvas N (1999) Coercivity conditions and variational inequalities. Math Programm 86:433–438
6. Dietrich H (1999) A smooth dual gap function solution to a class of quasivariational inequalities. J Math Anal Appl 235:380–393
7. Facchinei F, Pang JS (2003) Finite-dimensional variational inequalities and complementarity problems, vol I, II. Springer, New York
8. Fang SC, Peterson EL (1982) Generalized variational inequalities. J Optim Theory Appl 38:363–383
9. Giannessi F, Maugeri A, Pardalos P (2001) Equilibrium Problems: Nonsmooth Optimization and Variational Inequality Models. Kluwer, Dordrecht
10. Giannessi F, Pardalos P, Rapcsák T (2001) Optimization theory. Recent developments. Kluwer, Dordrecht
11. Harker PT, Pang S (1990) Finite-Dimensional Variational Inequality and Nonlinear Complementarity Problems: a survey of theory, algorithms and applications. Math Programm 48:161–220
12. John R (2001) A note on Minty variational inequalities and generalized monotonicity. In: Generalized convexity and generalized monotonicity. Springer, Berlin, pp 240–246
13. Konnov I (2001) Combined relaxation methods for variational inequalities. Springer, Berlin
14. Mosco U (1972) Dual Variational Inequalities. J Math Anal Appl 202–206
15. Panicucci B, Pappalardo M, Passacantando M (2006) On finite-dimensional generalized variational inequalities. J Indust Manag Optim 2:43–53
16. Penot J-P, Quang PH (1997) Generalized convexity of functions and generalized monotonicity of set-valued maps. J Optim Theory Appl 92:343–356
17. Saigal R (1976) Extension of the generalized complementarity problem. Math Oper Res 1:260–266

General Moment Optimization Problems

GEORGE A. ANASTASSIOU
Department Math. Sci., The University Memphis, Memphis, USA

MSC2000: 28-XX, 49-XX, 60-XX

Article Outline

Keywords
The Standard Moment Problem
The Method of Optimal Distance
The Method of Optimal Ratio
The Convex Moment Problem
Description of the Problem
Solving the Convex Moment Problem
Infinite Many Conditions Moment Problem
Applications and Discussion
Final Conclusion
See also
References

Keywords

Geometric moment theory; Probability; Integral constraint; Optimal integral bounds subject to moment conditions; Finite moment problem; Convex moment problem; Convexity; Infinite moment problem

In this article we describe the main moment problems and their solution methods from theoretical to applied. In particular we present the standard moment problem, the convex moment problem, and the infinite many conditions moment problem. Optimization moment theory has a lot of important applications in many sciences and subjects, for a detailed list please see the final section.

The Standard Moment Problem

Let $g_1, \ldots, g_n$ and h be given real-valued Borel measurable functions on a fixed measurable space $X := (X, A)$. We would like to find the best upper and lower bound on

$$\mu(h) := \int_X h(t)\mu(dt),$$

given that μ is a probability measure on X with prescribed moments

$$\int g_i(t)\,\mu(dt) = y_i, \qquad i = 1, \ldots, n.$$

Here we assume μ such that

$$\int_X |g_i|\ \mu(dt) < +\infty, \qquad i = 1, \ldots, n,$$

and

$$\int_X |h|\ \mu(dt) < +\infty.$$

For each $y := (y_1, \ldots, y_n) \in \mathbf{R}^n$, consider the optimal quantities

$$L(y) := L(y|h) := \inf_{\mu} \mu(h),$$
$$U(y) := U(y|h) := \sup_{\mu} \mu(h),$$

where μ is a probability measure as above with

$$\mu(g_i) = y_i, \qquad i = 1, \ldots, n.$$

If there is no such probability measure μ we set $L(y) := +\infty$, $U(y) := -\infty$.

If $h := \chi_S$ the characteristic function of a given measurable set S of X, then we agree to write

$$L(y|\chi_S) := L_S(y), \quad U(y|\chi_S) := U_S(y).$$

Hence, $L_S(y) \le \mu(S) \le U_S(y)$. Consider $g: X \to \mathbf{R}^n$ such that $g(t) := (g_1(t), \ldots, g_n(t))$. Set also $g_0(t) := 1$, all $t \in X$. Here we basically present J.H.B. Kemperman's (1968) geometric methods for solving the above main moment problems [13] which were related to and motivated by [18,20,24]. The advantage of the geometric method is that many times is simple and immediate giving us the optimal quantities L, U in a closed-numerical form, on the top of this is very elegant. Here the σ-field A contains all subsets of X.

The next result comes from [22,23,25].

Theorem 1 *Let $f_1, \ldots, f_N$ be given real-valued Borel measurable functions on a measurable space Ω (such as $g_1, \ldots, g_n$ and h on X). Let μ be a probability measure on Ω such that each f_i is integrable with respect to μ. Then there exists a probability measure μ' of finite support on Ω (i. e., having nonzero mass only at a finite number of points) satisfying*

$$\int_\Omega f_i(t)\,\mu(dt) = \int_\Omega f_i(t)\,\mu'(dt),$$

all $i = 1, \ldots, N$.

One can even achieve that the support of μ' has at most $N+1$ points. So from now on we can talk only about finitely supported probability measures.

Call

$$V := \operatorname{conv} g(X)$$

(conv stands for convex hull), where $g(X) := \{z \in \mathbf{R}^n: z = g(t)$ for some $t \in X\}$ is a curve in $\mathbf{R}^n$ (if $X = [a, b] \subset \mathbf{R}$ or if $X = [a, b] \times [c, d] \subset \mathbf{R}^2$).

Let $S \subset X$, and let $M^+(S)$ denote the set of all probability measures on X whose support is finite and contained in S.

The next results come from [13].

Lemma 2 *Given $y \in \boldsymbol{R}^n$, then $y \in V$ if and only if $\exists \mu \in M^+(X)$ such that*

$$\mu(g) = y$$

(i. e. $\mu(g_i) := \int_X g_i(t)\,\mu(dt) = y_i$, $i = 1, \ldots, n$).

Hence $L(y|h) < +\infty$ if and only if $y \in V$ (note that by Theorem 1,

$$L(y|h) = \inf\{\mu(h):\ \mu \in M^+(X), \mu(g) = y\}$$

and

$$U(y|h) = \sup\{\mu(h):\ \mu \in M^+(X), \mu(g) = y\}).$$

Easily one can see that

$$L(y) := L(y|h)$$

is a convex function on V, i. e.

$$L(\lambda y' + (1-\lambda)y'') \le \lambda L(y') + (1-\lambda)L(y''),$$

whenever $0 \le \lambda \le 1$ and $y', y'' \in V$. Also $U(y) := U(y|h) = -L(y|-h)$ is a concave function on V.

One can also prove that the following three properties are equivalent:

i) $\text{int}(V) :=$ interior of $V \neq \phi$;
ii) $g(X)$ is not a subset of any hyperplane in $\mathbf{R}^n$;
iii) $1, g_1, \ldots, g_n$ are linearly independent on X.

From now on we assume that $1, g_1, \ldots, g_n$ are linearly independent, i. e. $\text{int}(V) \neq \phi$.

Let D^* denote the set of all $(n+1)$-tuples of real numbers $d^* := (d_0, \ldots, d_n)$ satisfying

$$h(t) \ge d_0 + \sum_{i=1}^{n} d_i g_i(t), \qquad \text{all} \quad t \in X. \tag{1}$$

Theorem 3 *For each $y \in int\,(V)$ we have that*

$$L(y|h) \tag{2}$$

$$= \sup \left\{ d_0 + \sum_{i=1}^{n} d_i y_i : \; d^* = (d_0, \ldots, d_n) \in D^* \right\}.$$

Given that $L(y|h) > -\infty$, the supremum in (2) is even assumed by some $d^* \in D^*$.

If $L(y|h)$ is finite in $\text{int}(V)$, then for almost all $y \in \text{int}(V)$ the supremum in (2) is assumed by a unique $d^* \in D^*$. Thus $L(y|h) < +\infty$ in $\text{int}(V)$ if and only if $D^* \neq \emptyset$. Note that $y := (y_1, \ldots, y_n) \in \text{int}(V) \subset \mathbf{R}^n$ if and only if $d_0 + \sum_{i=1}^{n} d_i y_i > 0$ for each choice of the real constants d_i not all zero such that $d_0 + \sum_{i=1}^{n} d_i g_i(t) \ge 0$, all $t \in X$. (The last statement comes from [8 p. 5] and [12 p. 573].)

If h is bounded then $D^* \neq \emptyset$, trivially.

Theorem 4 *Let $d^* \in D^*$ be fixed and set*

$$B(d^*) := \left\{ z = g(t) : \; d_0 + \sum_{i=1}^{n} d_i g_i(t) = h(t), \, t \in X \right\} \tag{3}$$

Then for each point

$$y \in \text{conv}\, B(d^*) \tag{4}$$

the quantity $L(y|h)$ is found as follows. Set

$$y = \sum_{j=1}^{m} p_j g(t_j)$$

with

$$g(t_j) \in B(d^*),$$

and

$$p_j \ge 0, \quad \sum_{j=1}^{m} p_j = 1. \tag{5}$$

Then

$$L(y|h) = \sum_{j=1}^{m} p_j h(t_j) = d_0 + \sum_{i=1}^{n} d_i y_i. \tag{6}$$

Theorem 5 *Let $y \in int(V)$ be fixed. Then the following are equivalent:*

i) $\exists \mu \in M^+(X)$ such that $\mu(g) = y$ and $\mu(h) = L(y|h)$, i. e. infimum is attained.
ii) $\exists d^* \in D^*$ satisfying (4).

Furthermore for almost all $y \in int(V)$ there exists at most one $d^* \in D^*$ satisfying (4).

In many situations the above infimum is not attained so that Theorem 4 is not applicable. The next theorem has more applications. For that, set

$$\eta(z) := \liminf_{\delta \to 0} \inf_{t} \{ h(t) : \; t \in X, |g(t) - z| < \delta \}. \tag{7}$$

If $\varepsilon \ge 0$ and $d^* \in D^*$, define

$$C_\varepsilon(d^*) := \left\{ z \in \overline{g(T)} : \; 0 \le \eta(z) - \sum_{i=0}^{n} d_i z_i \le \varepsilon \right\}, \tag{8}$$

and

$$G(d^*) := \bigcap_{N=1}^{\infty} \overline{\text{conv}} C_{\frac{1}{N}}(d^*). \tag{9}$$

It is easily proved that $C_\varepsilon(d^*)$ and $G(d^*)$ are closed; furthermore $B(d^*) \subset C_0(d^*) \subset C_\varepsilon(d^*)$, where $B(d^*)$ is defined by (3).

Theorem 6 *Let $y \in int(V)$ be fixed.*
i) Let $d^* \in D^*$ be such that $y \in G(d^*)$. Then

$$L(y|h) = d_0 + d_1 y_1 + \cdots + d_n y_n. \tag{10}$$

ii) Assume that g is bounded. Then there exists $d^* \in D^*$ *satisfying*

$$y \in \operatorname{conv} C_0(d^*) \subset G(d^*)$$

and

$$L(y|h) = d_0 + d_1 y_1 + \cdots + d_n y_n. \tag{11}$$

iii) We further obtain, whether or not g is bounded, that for almost all $y \in int(V)$ *there exists at most one* $d^* \in D^*$ *satisfying* $y \in G(d^*)$.

The above results suggest the following practical simple geometric methods for finding $L(y|h)$ and $U(y|h)$, see [13].

The Method of Optimal Distance

Call

$$M := \operatorname{conv}_{t\in X}(g_1(t), \ldots, g_n(t), h(t)).$$

Then $L(y|h)$ is equal to the *smallest* distance between $(y_1, \ldots, y_n, 0)$ and $(y_1, \ldots, y_n, z) \in \overline{M}$. Also $U(y|h)$ is equal to the *largest* distance between $(y_1, \ldots, y_n, 0)$ and $(y_1, \ldots, y_n, z) \in \overline{M}$. Here, $\overline{M}$ stands for the closure of M. In particular we see that $L(y|h) = \inf\{y_{n+1} : (y_1, \ldots, y_n, y_{n+1}) \in M\}$ and

$$\begin{aligned} &U(y|h) \\ &= \sup\{y_{n+1} : (y_1, \ldots, y_n, y_{n+1}) \in M\}. \end{aligned} \tag{12}$$

Example 7 Let μ denote probability measures on $[0, a]$, $a > 0$. Fix $0 < d < a$. Find

$$L := \inf_{\mu} \int_{[0,a]} t^2 \, \mu(dt)$$

and

$$U := \sup_{\mu} \int_{[0,a]} t^2 \, \mu(dt)$$

subject to

$$\int_{[0,a]} t \, \mu(dt) = d.$$

So consider the graph $G := \{(t, t^2) : 0 \le t \le a\}$. Call $M := \overline{\operatorname{conv} G} = \operatorname{conv} G$.

A direct application of the optimal distance method here gives us $L = d^2$ (an optimal measure μ is supported at d with mass 1), and $U = da$ (an optimal measure μ here is supported at 0 and a with masses $(1 - d/a$ and d/a, respectively).

The Method of Optimal Ratio

We would like to find

$$L_S(y) := \inf \mu(S)$$

and

$$U_S(y) := \sup \mu(S),$$

over all probability measures μ such that

$$\mu(g_i) = y_i, \qquad i = 1, \ldots, n.$$

Set $S' := X - S$. Call $W_S := \overline{\operatorname{convg}(S)}$, $W_{S'} := \overline{\operatorname{convg}(S')}$ and $W := \overline{\operatorname{convg}(X)}$, where $g := (g_1, \ldots, g_n)$.

Finding $L_S(y)$.

1) Pick a boundary point z of W and 'draw' through z a hyperplane H of support to W.
2) Determine the hyperplane H' parallel to H which supports $W_{S'}$ as well as possible, and on the same side as H supports W.
3) Denote

$$A_d := W \cap H = W_S \cap H$$

and

$$B_d := W_{S'} \cap H'.$$

Given that $H' \neq H$, set $G_d := \overline{\operatorname{conv}}(A_d \cup B_d)$. Then we have that

$$L_S(y) = \frac{\Delta(y)}{\Delta}, \tag{13}$$

for each $y \in \operatorname{int}(V)$ such that $y \in G_d$. Here, $\Delta(y)$ is the distance from y to H' and Δ is the distance between the distinct parallel hyperplanes H, H'.

Finding $U_S(y)$. (Note that $U_S(y) = 1 - L_{S'}(y)$.)

1) Pick a boundary point z of W_S and 'draw' through z a hyperplane H of support to W_S. Set $A_d := W_S \cap H$.
2) Determine the hyperplane H' parallel to H which supports $g(X)$ and hence W as well as possible, and on the same side as H supports W_S. We are interested only in $H' \neq H$ in which case H is between H' and W_S.
3) Set $B_d := W \cap H' = W_{S'} \cap H'$. Let G_d as above. Then

$$U_S(y) = \frac{\Delta(y)}{\Delta}, \tag{14}$$

for each $y \in \operatorname{int}(V)$, where $y \in G_d$, assuming that H and H' are distinct. Here, $\Delta(y)$ and Δ are defined as above.

Examples here of calculating $L_S(y)$ and $U_S(y)$ tend to be more involved and complicated, however the applications are many.

The Convex Moment Problem

Definition 8 Let $s \geq 1$ be a fixed natural number and let $x_0 \in \mathbf{R}$ be fixed. By $m_s(x_0)$ we denote the set of probability measures μ on $\mathbf{R}$ such that the associated cumulative distribution function F possesses an $(s-1)$th derivative $F^{(s-1)}(x)$ over $(x_0, +\infty)$ and furthermore $(-1)^s F^{(s-1)}(x)$ is convex in $(x_0, +\infty)$.

Description of the Problem

Let g_i, $i = 1, \ldots, n$; h are Borel measurable functions from $\mathbf{R}$ into itself. These are assumed to be locally integrable on $[x_0, +\infty)$ relative to Lebesgue measure. Consider $\mu \in m_s(x_0)$, $s \geq 1$ such that

$$\mu(|g_i|) := \int_{\mathbb{R}} |g_i(t)| \ \mu(dt) < +\infty, \quad i = 1, \ldots, n \tag{15}$$

and

$$\mu(|h|) := \int_{\mathbb{R}} |h(t)| \ \mu(dt) < +\infty. \tag{16}$$

Let $c := (c_1, \ldots, c_n) \in \mathbf{R}^n$ be such that

$$\mu(g_i) = c_i, \quad i = 1, \ldots, n, \quad \mu \in m_s(x_0). \tag{17}$$

We would like to find $L(c) := \inf_\mu \mu(h)$ and

$$U(c) := \sup_\mu \mu(h), \tag{18}$$

where μ is as above described.

Here, the method will be to transform the above convex moment problem into an ordinary one handled by the first section, see [14].

Definition 9 Consider here another copy of $(\mathbf{R}, \mathbf{B})$; $\mathbf{B}$ is the Borel σ-field, and further a given function $P(y, A)$ on $\mathbf{R} \times \mathbf{B}$.

Assume that for each fixed $y \in \mathbf{R}$, $P(y, \cdot)$ is a probability measure on $\mathbf{R}$, and for each fixed $A \in \mathbf{B}$, $P(\cdot, A)$ is a Borel-measurable real-valued function on $\mathbf{R}$. We call P a *Markov kernel*. For each probability measure ν on $\mathbf{R}$, let $\mu := T\nu$ denote the probability measure on $\mathbf{R}$ given by

$$\mu(A) := (T\nu)(A) := \int_{\mathbb{R}} P(y, A) \ \nu(dy).$$

T is called a *Markov transformation*.

In particular: Define the kernel

$$K_s(u, x) := \begin{cases} \frac{s(u-x)^{s-1}}{(u-x_0)^s} & \text{if } x_0 < x < u, \\ 0 & \text{elsewhere.} \end{cases} \tag{19}$$

Notice $K_s(u, x) \geq 0$ and $\int_{\mathbf{R}} K_s(u, x) = dx = 1$, all $u > x_0$. Let δ_u be the unit (Dirac) measure at u. Define

$$P_s(u, A) := \begin{cases} \delta_u(A) & \text{if } u \leq x_0; \\ \int_A K_s(u, x) \, dx & \text{if } u > x_0. \end{cases} \tag{20}$$

Then

$$(T\nu)(A) := \int_{\mathbb{R}} P_s(u, A)\nu(\, du) \tag{21}$$

is a Markov transformation.

Theorem 10 *Let $x_0 \in \mathbf{R}$ and natural number $s \geq 1$ be fixed. Then the Markov transformation (21) $\mu = T\nu$ defines a 1-1 correspondence between the set m_* of all probability measures ν on $\mathbf{R}$ and the set $m_s(x_0)$ of all probability measures μ on $\mathbf{R}$ as in Definition 8. In fact T is a homeomorphism given that m^* and $m_s(x_0)$ are endowed with the weak*-topology.*

Let ϕ: $\mathbf{R} \to \mathbf{R}$ be a bounded and continuous function. Introducing

$$\phi^*(u) := (T\phi)(u) := \int_{\mathbb{R}} \phi(x) \cdot P_s(u, dx), \tag{22}$$

then

$$\int \phi d\mu = \int \phi^* \, d\nu. \tag{23}$$

Here ϕ^* is a bounded and continuous function from $\mathbf{R}$ into itself.

We obtain that

$$\phi^*(u) = \begin{cases} \phi(u) & \text{if } u \leq x_0; \\ \int_0^1 \phi((1-t)u + tx_0)st^{s-1} \, dt & \\ \text{if } u > x_0. & \end{cases} \tag{24}$$

In particular

$$\frac{1}{s!}(u-x_0)^s\phi^*(u) = \frac{1}{(s-1)!}\int_{x_0}^{u}(u-x)^{s-1}\phi(x)\,dx. \tag{25}$$

Especially, if $r > -1$ we get for $\phi(u) := (u-x_0)^r$ that $\phi^*(u) = \binom{r+s}{s}^{-1}(u-x_0)^r$, for all $u > x_0$. Here $r! := 1\cdot 2\cdots r$ and

$$\binom{r+s}{s} := \frac{(r+1)\cdots(r+s)}{s!}.$$

Solving the Convex Moment Problem

Let T be the Markov transformation (21) as described above. For each $\mu \in m_s(x_0)$ corresponds exactly one $\nu \in m^*$ such that $\mu = T\nu$. Call $g_i^* := Tg_i$, $i = 1, \ldots, n$ and $h^* := Th$. We have

$$\int_{\mathbb{R}} g_i^*\,d\mu = \int_{\mathbb{R}} g_i\,d\mu$$

and

$$\int_{\mathbb{R}} h^*\,d\nu = \int_{\mathbb{R}} h\,d\mu.$$

Notice that we get

$$\nu(g_i^*) := \int_{\mathbb{R}} g_i^*\,d\nu = c_i; \quad i = 1, \ldots, n. \tag{26}$$

From (15), (16) we get that

$$\int_{\mathbb{R}} T|g_i|\,d\nu < +\infty, \quad i = 1, \ldots, n,$$

and

$$\int_{\mathbb{R}} T|h|\,d\nu < +\infty. \tag{27}$$

Since T is a positive linear operator we obtain $|Tg_i| \le T|g_i|$, $i = 1, \ldots, n$, and $|Th| \le T|h|$, i. e.

$$\int_{\mathbb{R}} |g_i^*|\,d\nu < +\infty, \quad i = 1, \ldots, n,$$

and

$$\int_{\mathbb{R}} |h^*|\,d\nu < +\infty.$$

That is, g_i^*, h^* are ν-integrable.

Finally

$$L(c) = \inf_{\nu}\nu(h^*) \tag{28}$$

and

$$U(c) = \sup_{\nu}\nu(h^*), \tag{29}$$

where $\nu \in m^*$ (probability measure on $\mathbf{R}$) such that (26) and (27) are true.

Thus the convex moment problem is solved as a standard moment problem (see the first section).

Remark 11 Here we restrict our probability measures on $[0, +\infty)$ and we consider the case $x_0 = 0$. That is $\mu \in m_s(0)$, $s \ge 1$, i. e. $(-1)^s F^{(s-1)}(x)$ is convex for all $x > 0$ but $\mu(\{0\}) = \nu(\{0\})$ can be positive, $\nu \in m^*$. We have

$$\phi^*(u) = su^{-s}\cdot\int_0^u (u-x)^{s-1}\cdot\phi(x)\cdot dx, \quad u > 0. \tag{30}$$

Further $\phi^*(0) = \phi(0)$, $(\phi^* = T\phi)$. Especially,

$$\text{if} \quad \phi(x) = x^r \quad \text{then} \quad \phi^*(u) = \binom{r+s}{s}^{-1}\cdot u^r, \quad (r \ge 0). \tag{31}$$

Hence the moment

$$\alpha_r := \int_0^{+\infty} x^r\mu(\,dx) \tag{32}$$

is also expressed as

$$\alpha_r = \binom{r+s}{s}^{-1}\cdot\beta_r, \tag{33}$$

where

$$\beta_r := \int_0^{+\infty} u^r\,\nu(du). \tag{34}$$

Recall that $T\nu = \mu$, where ν can be any probability measure on $[0, +\infty)$.

Here we restrict our probability measures on $[0, b]$, $b > 0$ and again we consider the case $x_0 = 0$. Let $\mu \in$

$m_s(0)$ and

$$\int_{[0,b]} x^r \, \mu(dx) := \alpha_r, \tag{35}$$

where $s \geq 1$, $r > 0$ are fixed.

Also let ν be a probability measure on $[0, b]$ unrestricted, i. e. $\nu \in m^*$. Then $\beta_r = \binom{r+s}{s} \alpha_r$, where

$$\beta_r := \int_{[0,b]} u^r \, \nu(du). \tag{36}$$

Let h: $[0, b] \to \mathbf{R}_+$ be an integrable function with respect to Lebesgue measure. Consider $\mu \in m_s\,(0)$ such that

$$\int_{[0,b]} h \, d\mu < +\infty. \tag{37}$$

i. e.

$$\int_{[0,b]} h^* \, d\nu < +\infty, \quad \nu \in m^*. \tag{38}$$

Here $h^* = Th$, $\mu = T\nu$ and

$$\int_{[0,b]} h \, d\mu = \int_{[0,b]} h^* \, d\nu.$$

Letting α_r be free, we have that the set of all possible $(\alpha_r, \mu(h)) = (\mu(x^r), \mu(h))$ coincides with the set of all

$$\left(\binom{r+s}{s}^{-1} \cdot \beta_r, \nu(h^*) \right)$$
$$= \left(\binom{r+s}{s}^{-1} \cdot \nu(u^r), \nu(h^*) \right),$$

where μ as in (37) and ν as in (38), both probability measures on $[0, b]$. Hence, the set of all possible pairs $(\beta_r, \mu(h)) = (\beta_r, \nu(h^*))$ is precisely the convex hull of the curve

$$\Gamma := \{(u^r, h^*(u)) \colon \ 0 \leq u \leq b\}. \tag{39}$$

In order one to determine $L(\alpha_r)$ the infimum of all $\mu(h)$, where μ is as in (35) and (37), one must determine the lowest point in this convex hull which is on the vertical through $(\beta_r, 0)$. For U(α_r) the supremum of all $\mu(h)$, μ as above, one must determine the highest point of above convex hull which is on the vertical through $(\beta_r, 0)$.

For more on the above see again §1.

Infinite Many Conditions Moment Problem

See also [16].

Definition 13 A finite nonnegative measure μ on a compact and Hausdorff space S is said to be *inner regular* when

$$\mu(B) = \sup\{\mu(K) \colon \ K \subseteq B;\ K \text{ compact}\} \tag{40}$$

holds for each Borel subset B of S.

Theorem 14 *See [16]. Let S be a compact Hausdorff topological space and a_i: $S \to \mathbf{R} (i \in I)$ continuous functions (I is an index set of arbitrary cardinality), also let α_i $(i \in I)$ be an associated set of real constants. Call $M_0(S)$ the set of finite nonnegative inner regular measures μ on S which satisfy the moment conditions*

$$\mu(a_i) = \int_S a_i(s) \, \mu(ds) \leq \alpha_i, \quad \text{all} \quad i \in I. \tag{41}$$

Also consider the function b: $S \to \mathbf{R}$ which is continuous and assume that there exist numbers $d_i \geq 0$ $(i \in I)$, all but finitely many equal to zero, and further a number $q \geq 0$ such that

$$1 \leq \sum_{i \in I} d_i a_i(s) - qb(s), \quad \text{all} \quad s \in S. \tag{42}$$

Finally assume that $M_0(S) \neq \emptyset$ and call

$$U_0(b) = \sup\{\mu(b) \colon \ \mu \in M_0(S)\}. \tag{43}$$

$(\mu(b) := \int_S b(s) \, \mu(ds))$. Then

$$U_0(b) = \inf\left\{ \sum_{i \in I} c_i \alpha_i \colon \ \begin{matrix} c_i \geq 0; \\ b(s) \leq \sum_{i \in I} c_i a_i(s) \text{ all } s \in S \end{matrix} \right\}, \tag{44}$$

here all but finitely many c_i, $i \in I$, are equal to zero. Moreover, $U_0(b)$ is finite and the above supremum is assumed.

Remark 15 In general we have: let S be a fixed measurable space such that each 1-point set $\{s\}$ is measurable. Further let $M_0(S)$ denote a fixed nonempty set of finite nonnegative measures on S.

For f: $S \to \mathbf{R}$ a measurable function we denote

$$L_0(f) := L(f, M_0(S)) := \inf\left\{ \int_S f(s) \, \mu(ds) \colon \ \mu \in M_0(S) \right\}. \tag{45}$$

Then we have

$$L_0(f) = -U_0(-f). \tag{46}$$

Now one can apply Theorem 14 in its setting to find $L_0(f)$.

Applications and Discussion

The above described moment theory optimization methods have a lot of applications in many sciences. To mention a few of them: physics, chemistry, statistics, stochastic processes and probability, functional analysis in mathematics, medicine, material science, etc. Optimization moment theory could be also considered the theoretical part of linear finite or semi-infinite programming (here we consider discretized finite nonnegative measures).

The above described methods have in particular important applications: in the marginal moment problems and the related transportation problems, also in the quadratic moment problem, see [17].

Other important applications are in tomography, crystallography, queueing theory, rounding problem in political science, and martingale inequalities in probability. At last, but not least, optimization moment theory has important applications in estimating the speeds: of the convergence of a sequence of positive linear operators to the unit operator, and of the weak convergence of nonnegative finite measures to the unit-Dirac measure at a real number, for that and the solutions of many other important optimal moment problems please see [2].

Final Conclusion

Optimization moment theory is a very active area of mathematical probability theory with a lot of applications in other subjects, and with a lot of researchers from around the world in it contributing new useful results, continuously during all of the 20th century.

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Akhiezer NI (1965) The classical moment problem. Hafner, New York

2. Anastassiou GA (1993) Moments in probability and approximation theory. Res Notes Math, vol 287. Pitman, Boston, MA
3. Anastassiou GA, Rachev ST (1992) How precise is the approximation of a random queue by means of deterministic queueing models. Comput Math Appl 24(8-9):229–246
4. Anastassiou GA, Rachev ST (1992) Moment problems and their applications to characterization of stochastic processes, queueing theory, and rounding problems. In: Anastassiou G (ed) Proc. 6th S.E.A. Meeting, Approximation Theory. 1–77 M. Dekker, New York
5. Benes V, Stepan J (eds) (1997) Distributions with given marginals and moment problems. Kluwer, Dordrecht
6. Isii K (1960) The extreme of probability determined by generalized moments (I): bounded random variables. Ann Inst Math Statist 12:119–133
7. Johnson NL, Rogers CA (1951) The moment problems for unimodal distributions. Ann Math Stat 22:433–439
8. Karlin S, Shapley LS (1953) Geometry of moment spaces. Memoirs, vol 12. Amer Math. Soc., Providence, RI
9. Karlin S, Studden WJ (1966) Tchebycheff systems: with applications in analysis and statistics. Interscience, New York
10. Kellerer HG (1964) Verteilungsfunktionen mit gegebenen Marginalverteilungen. Z Wahrscheinlichkeitsth Verw Gebiete 3:247–270
11. Kellerer HG (1984) Duality theorems for marginal problems. Z Wahrscheinlichkeitsth Verw Gebiete 67:399–432
12. Kemperman JHB (1965) On the sharpness of Tchebycheff type inequalities. Indagationes Mathematicae 27:554–601
13. Kemperman JHB (1968) The general moment problem, a geometric approach. Ann MathStat 39:93–122
14. Kemperman JHB (1971) Moment problems with convexity conditions. In: Rustagi JS (ed) Optimizing Methods in Statistics. Acad. Press, New York, pp 115–178
15. Kemperman JHB (1972) On a class of moment problems. Proc. Sixth Berkeley Symp. Math. Stat. Prob. 2, pp 101–126
16. Kemperman JHB (1983) On the role of duality in the theory of moments. In: Fiacco AV, Kortanek KO (eds) Semi-Infinite Programming and Applications. of Lecture Notes Economics and Math Systems. Springer, Berlin, pp 63–92
17. Kemperman JHB (1987) Geometry of the moment problem. Moments in Math., of In: Short Course Ser, San Antonio, Texas, 1986, vol 34. Amer. Math. Soc., Providence, RI), pp 20–22
18. Krein MG (1959) The ideas of P.L. Cebysev and A.A. Markov in the theory of limiting values of integrals and their further development. AmerMathSoc Transl 2(12):1–121. ((1951) Uspekhi Mat Nauk 6:3–130)
19. Krein MG, Nudel'man AA (1977) The Markov moment problem and extremal problems. Amer. Math. Soc., Providence, RI
20. Markov A (1884) On certain applications of algebraic continued fractions. Thesis Univ St Petersburg
21. Mises R von (1939) The limits of a distribution function if two expected values are given. Ann Math Stat 10:99–104
22. Mulholland HP, Rogers CA (1958) Representation theorems for distribution functions. Proc London Math Soc 8:177–223
23. Richter H (1957) Parameterfreie Abschätzung und Realisierung von Erwartungswerten. Blätter Deutschen Gesellschaft Versicherungsmath 3:147–161
24. Riesz F (1911) Sur certaines systèmes singuliers d'équations intégrales. Ann Sci Ecole Norm Sup 28:33–62
25. Rogosinsky WW (1958) Moments of non-negative mass. Proc Royal Soc London Ser A 245:1–27
26. Rogosinsky WW (1962) Non-negative linear functionals, moment problems, and extremum problems in polynomial spaces. Stud. Math. Anal. and Related Topics. Stanford Univ Press, Palo Alto, CA, pp 316–324
27. Selberg HL (1940) Zwei Ungleichungen zur Ergänzung des Tchebycheffschen Lemmas. Skand Aktuarietidskrift 23:121–125
28. Shohat JA, Tamarkin JD (1983) The problem of moments. Math Surveys, vol 1. Amer. Math. Soc., Providence, RI
29. Shortt RM (1983) Strassen's marginal problem in two or more dimensions. Z Wahrscheinlichkeitsth Verw Gebiete 64:313–325

General Routing Problem

GRP

RICHARD EGLESE, ADAM LETCHFORD
Lancaster University, Lancaster, UK

MSC2000: 90B20

Article Outline

Keywords
See also
References

Keywords

Routing

The *general routing problem* (GRP) is a routing problem defined on a graph or network where a minimum cost tour is to be found and where the route must include visiting certain required vertices and traversing certain required edges. More formally, given a connected, undirected graph G with vertex set V and (undirected) edge set E, a cost c_e for traversing each edge e

$\in E$, a set $V_R \subseteq V$ of *required vertices* and a set $E_R \subseteq E$ of *required edges*, the GRP is the problem of finding a minimum cost vehicle route, starting and finishing at the same vertex, passing through each $v \in V_R$ and each $e \in E_R$ at least once ([13]).

The GRP contains a number of other routing problems as special cases. When $E_R = \emptyset$, the GRP reduces to the *Steiner graphical traveling salesman problem* (SGTSP) ([4]), also called the *road traveling salesman problem* in [7]. On the other hand, when $V_R = \emptyset$, the GRP reduces to the *rural postman problem* (RPP) ([13]). When $V_R = V$, the SGTSP in turn reduces to the *graphical traveling salesman problem* or GTSP ([4]). Similarly, when $E_R = E$, the RPP reduces to the *Chinese postman problem* or CPP ([5,8]).

The CPP can be solved optimally in polynomial time by reduction to a matching problem ([6]), but the RPP, GTSP, SGTSP and GRP are all NP-hard. This means that the computational effort to solve such a problem increases exponentially with the size of the problem. Therefore exact algorithms are only practical for a GRP if it is not too large, otherwise a heuristic algorithm is appropriate. The GRP was proved to be NP-hard in [10].

In [3], an integer programming formulation of the GRP is given, along with several classes of valid inequalities which induce facets of the associated polyhedra under mild conditions. Another class of valid inequalities for the GRP is introduced in [11] and in [12] it is shown how to convert facets of the GTSP polyhedron into valid inequalities for the GRP polyhedron. These valid inequalities form the basis for a promising branch and cut style of algorithm described in [2] which can solve GRPs of moderate size to optimality.

In [9], a heuristic algorithm for the GRP is described. The author adapts Christofides' heuristic for the TSP to show that when the triangle inequality holds in the graph, the heuristic has a worst-case ratio of heuristic solution value to optimum value of 1.5.

There are many vehicle routing applications of the GRP. In these cases, the edges of the graph are used to represent streets or roads and the vertices represent road junctions or particular locations on a map. In any practical application there are likely to be many additional constraints which must also be taken into account such as the capacity of the vehicles, time-window constraints for when the service may be carried out, the existence of one-way streets and prohibited turns etc.

Many applications are for the special cases when either $E_R = \emptyset$ or $V_R = \emptyset$. However, there are some types of vehicle routing applications where the problem is most naturally modeled as a GRP with both required edges and required vertices. For example, in designing routes for solid waste collection services, collecting waste from all houses along a street could be modeled as a required edge and collecting waste from the foot of a multistory apartment block could be modeled as a required vertex. Other examples include postal delivery services where some customers with heavy demand might be modeled as required vertices, while other customers with homes in the same street might be modeled together as a required edge. School bus services are other examples of GRPs where a pick-up in a remote village could be modeled as a required vertex, but if the school bus must pick-up at some point along a street (and is not allowed to perform a U-turn in the street) then that may best be modeled as a required edge.

Further details about solution methods and applications for various network routing problems can be found in [1].

See also

- ▶ Stochastic Vehicle Routing Problems
- ▶ Vehicle Routing
- ▶ Vehicle Scheduling

References

1. Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) (1995) Network routing. vol 8, Handbook Oper. Res. and Management Sci. North-Holland, Amsterdam
2. Corberáan A, Letchford AN, Sanchis JM (1998) A cutting-plane algorithm for the general routing problem. Working Paper
3. Corberáan A, Sanchis JM (1998) The general routing problem polyhedron: Facets from the RPP and GTSP polyhedra. Europ J Oper Res 108:538–550
4. Cornué;jols G, Fonlupt J, Naddef D (1985) The travelling salesman problem on a graph and some related integer polyhedra. Math Program 33:1–27
5. Edmonds J (1963) The Chinese postman problem. Oper Res 13:B73–B77
6. Edmonds J, Johnson EL (1973) Matchings, Euler tours and the Chinese postman. Math Program 5:88–124

7. Fleischmann B (1985) A cutting-plane procedure for the travelling salesman problem on a road network. Europ J Oper Res 21:307–317
8. Guan M (1962) Graphic programming using odd or even points. Chinese Math 1:237–277
9. Jansen K (1992) An approximation algorithm for the general routing problem. Inform Process Lett 41:333–339
10. Lenstra JK, Rinnooy Kan AHG (1976) On general routing problems. Networks 6:273–280
11. Letchford AN (1997) New inequalities for the general routing problem. Europ J Oper Res 96:317–322
12. Letchford AN (1999) The general routing polyhedron: A unifying framework. Europ J Oper Res 112:122–133
13. Orloff CS (1974) A fundamental problem in vehicle routing. Networks 4:35–64

Genetic Algorithms

GA

Richard S. Judson
Genaissance Pharmaceuticals, New Haven, USA

MSC2000: 92B05

Article Outline

Keywords
See also
References

Keywords

Optimization; Genetic algorithms; Evolution; Stochastic global optimization; Population; Fitness; Crossover; Mutation; Binary encoding; Individual; Chromosome; Generation; Elitism; Premature convergence; Gray code; Random walk search; Roulette wheel procedure; Population size; Schema theorem; Schema; Local minimum; Selection; Evolution strategy

Genetic algorithms (GAs) comprise a class of *stochasticglobal optimization* methods based on several strategies from biological evolution. The basic genetic algorithm was developed by J.H. Holland and his students ([5,6,7,8]), and was based on the observation that selection (either natural or artificial) can produce highly optimized individuals in a relatively short number of generations. This is true despite the fact that the space of all gene mutations through which a population must sort is astronomical. For instancethe genome of the yeast *Saccharomyces cerevisiae*, which is the simplest eukaryote, contains just over 6000 genes, each of which can occur in several mutant forms. Despite this, *S. cerevisiae* can reoptimize itself to survive and flourish in many new environments in a relatively short number of generations. This is equivalent to having a computer search for a near-optimal solution to a 6000-dimensional problem where each of the 6000 variables can take on any one of a large number of values.

The most important notion from natural systems that the GA employs is the use of a *population* of individuals which go through a *selection* step to produce offspring and pass on their genetic material.Optimality or *fitness* is measured by how many offspring an individual produces. A second notion is the use of *crossover* in which individuals share genetic information and pass the shared information onto their offspring. A third borrowing from nature is the idea of *mutation*, the consequence of which is that the transfer of genetic informationis prone to random errors. This helps maintain the level of genetic diversity in a population.

The implementation of a simple GA (SGA) which uses these ideas is straightforward. The description that follows uses a *binary encoding*, but all of the ideas follow identically for integer or even real number encodings. The most important idea is that one works with a population of *individuals* which will interact through genetic operators to carry out an optimization process. An individual is specified by a *chromosome* C which is a bit string of length N_c that can be decoded to give a set of N parameters x_i which are the natural parameters for the optimization application. Each parameter x_i is encoded by n_i bits so that $\sum_i^N n_i = N_c$. In what follows, chromosome and bit string are synonymous. A fitness function $f(x_1, \ldots, x_N)$, which is the function to be optimized, is used to rank the individual chromosomes. An initial population of N_{pop} individuals is formed by choosing N_{pop} bit strings at random, and evaluating each individual's fitness. (Decode $C \to (x_1, \ldots, x_N)$, calculate $f(x_1, \ldots, x_N)$.)Subsequent *generations* are formed as follows. All parents (members of the current generation) are ranked by fitness and the highest fitness individual is placed directly into the next generation with

no change. (This step of keeping the most-fit individual intact is termed *elitism* and is a purely heuristic addition. It insures that good solutions to the problem at hand are not lost until better ones are found.) Next, pairs of parents are selected and their chromosomes are crossed over to form chromosomes of the remaining individuals in the next generation. A parent's probability of being selected increases with its fitness. So for a minimization application, the parent with the current lowest value of $f(x_1, \ldots, x_N)$ has the highest chance of being selected for mating. Crossover consists of taking some subset of the bits from parent 1 and the complementary set of bits from parent 2 and combining them to form the chromosome of child 1. A childis simply a member of the next generation. The remaining bits from the two parents are combined to form the chromosome of child 2. Additionally, during replication there is a small probability of a bit flip or mutation in a chromosome. This serves primarily to maintain diversity and prevent *premature convergence*. Convergence occurs when the population becomes largely homogeneous – most individuals have almost the same values for all of their parameters. Premature convergence occurs when the population converges early in a run, before significant amount of searching has been performed. The most common cause is a poor choice of the scaling of the fitness function. It should be noted that 'premature' and 'early' are loosely defined. To bound the magnitude of the effect of mutations, the binary chromosomes are usually *Gray coded*. An integer that is represented as a Gray coded binary number has the property that most single bit flips change the value of the decimal integer represented by the chromosome by ± 1. In sum, the algorithm consists of successively transforming one generation of individuals into the next using the operations of selection, crossover and mutation. Since the selection process is biased towards individuals with higher fitness, individuals are produced that come ever closer to being optimal solutions to the function of interest.

It is important to emphasize that crossover is the key feature that distinguishes the GA from other stochastic global search methods. If crossover is ineffective, GA degenerates into a *random walk search* being executed separately by each individual in the population. The random walk is generated by the mutation operator.

The GA is presented below as pseudocode:

```
PROCEDURE genetic algorithm()
    Initialize population;
    FOR (g = 1 to N_gen generations) DO
        FOR (i = 1 to N_pop individuals) DO
            Evaluate fitness of individual i: f_i(g):
        END FOR;
        Save best individual to population g + 1;
        FOR (i = 2 to N_pop) DO
            Select 2 individuals;
            Crossover: create 2 new individuals;
            Mutate the new individuals;
            Move new individuals to population g+1;
        END FOR;
    END FOR;
END genetic algorithm;
```

Pseudocode for the Simple Genetic Algorithm

Selection commonly uses a *roulette wheel procedure*. Each individual is assigned a slice of the unit circle proportional to its fitness ($f(x_1, \ldots, x_N)$).One then chooses pairs of random numbers to select the next two individuals to be mated. A typical crossover operator takes the chromosomes from apair of individuals and chooses a common cut point along them. One child gets the portion of the first parent's chromosome to the left of thecut point, and the portion of the second parent's chromosome to the right of the cut point. The chromosome of the second child is comprised of the remaining fragments of the two parent chromosomes. In the most common mutation operator each bit in the binary chromosome has an equal and low probability being flipped from 1 to 0 or vice versa.Many variants on these operators have been used.

The important variables in the GA method are the *population size*, N_{pop}, the total number of generations allowed, N_{gen}, the number of bits used to represent a real variable, and the mutation rate. The total CPU time used in an optimization run is proportional to $N_{\text{pop}} \times N_{\text{gen}} \times T(f)$, where $T(f)$ is the time required to evaluate the fitness function $f(x_1, \ldots, x_N)$. This leads to a trade-off between having large, diverse populations that explore parameter space widely, and having smaller populations that explore longer. In practice, the choice is problem dependent.

The simple GA and a large number of variants have been successfullyused to find near-optimal solutions to many engineering and scientific applications. ([2,3,4,6,9,10,11]) Although much effort has gone into formally analyzing the GA to understand why it is so robust, the most important formal result is the *Schema theorem* ([6,7,8]). *Schemata* are strings made up of the characters 1, 0 and $*$ which is the 'don't care' character. These schemata are building blocks out of which the strings representing individuals' chromosomes can be constructed. For instance the string 11100 contains schema such as 111, 1100 and $1*10$. The schema theorem provides a powerful statement about the behavior of schemata in a chromosome. Mathematically, it states

$$m(H, g+1) \geq m(H, g)\frac{f(H)}{\overline{f}}\left(1 - p_c\frac{\delta(H)}{l-1} - p_m\frac{o(H)}{p_m}\right), \quad (1)$$

where $m(H, g)$ is the number of examples of a schema H that exist in the population at generation g; $f(H)$ is the average fitness of chromosomes containing H; $\overline{f}$ is the average fitness of all chromosomes; p_c is the probability that crossover will occur at a particular mating; p_m is the probability that a particular bit will be mutated; l is the length of the chromosome; $\delta(H)$ is the length of the schema in bits; and o(H) is the order of the schema, defined to be the number of fixed (as opposed to don't care) positions in the schema.

The factors outside the brackets in (1) indicate that a particular schema will increase its representation in the population at a rate proportional to its fitness relative to the average fitness. Good schemata will increase their representation exponentially and bad schemata will decrease their representation likewise. The terms inside the bracket serve to decrease this exponential convergence by disrupting the selection-based pressure. Both crossover and mutation can disrupt good schemata. The longer a schema is, the more likely it is to be disrupted by crossover, and disappear from the population. In the same fashion, schemata with many fixed positions are more likely to be disrupted by mutations.

The competition between selection which drives the population towards convergence on a good solution and crossover and mutation which drive the population towards more diverse states are the keys to the GA. Crossover is especially important for keeping the method from being trapped in *local minima*. One consequence of the parameter shuffling brought about by the crossover operator is that the GA is most efficient at optimizing functions that are at least partially separable. One individual can find a state where half of the parameters of the fitness function are optimized and a second individual can find a state where the other half are optimized. If these individuals crossover at the correct point, one of theirchildren will have the parameter values that globally optimize the function.

As with most other heuristic global optimization methods, no definitive statements can be made about the global optimality of GA-generated solutions.

A family of algorithms that are very similar to the GA, called *evolution strategies* were developed independently and virtually simultaneously in Germany by I. Rechenberg ([1,12]).

See also

- ▶ Adaptive Simulated Annealing and its Application to Protein Folding
- ▶ Genetic Algorithms for Protein Structure Prediction
- ▶ Global Optimization in Lennard–Jones and Morse Clusters
- ▶ Global Optimization in Protein Folding
- ▶ Molecular Structure Determination: Convex Global Underestimation
- ▶ Monte-Carlo Simulated Annealing in Protein Folding
- ▶ Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach
- ▶ Packet Annealing
- ▶ Phase Problem in X-ray Crystallography: Shake and Bake Approach
- ▶ Protein Folding: Generalized-ensemble Algorithms
- ▶ Simulated Annealing
- ▶ Simulated Annealing Methods in Protein Folding

References

1. Bäck T, Schwefel H-P (1993) An overview of evolutionalgorithms for parameter optimization. Evolutionary Computation 1(1)
2. Belew RK, Booker LB (eds) (1991) Proc. fourth Internat. Conf. Genetic Algorithms. Morgan Kaufmann, San Mateo
3. Davis L (ed) (1987) Genetic algorithms and simulated annealing. Pitman, Boston

4. Davis L (1991) Handbook of genetic algorithms. v. Nostrand Reinhold, Princeton, NJ
5. DeJong K (1976) An analysis of the behavior of a class of genetic adaptive systems. PhD Thesis Univ. Michigan
6. Goldberg D (1989) Genetic algorithms in search, optimization and learning. Addison-Wesley, Reading
7. Holland JH (1992) Adaptation in natural and artificial systems. MIT, Cambridge
8. Holland JH (1992) Genetic algorithms. Scientif Amer 267:66
9. Judson RS (1997) Genetic algorithms and their use in chemistry. In: Lipkowitz KB, Boyd DB (eds) Rev. Computational Chemistry, vol 10. Wiley-VCH, Weinheim, pp 1–73
10. Koza J (1992) Genetic programming. MIT, Cambridge
11. Rawlins GJE (1991) Foundations of genetic algoritms. Morgan Kaufmann, San Mateo
12. Rechenberg I (1973) Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart-Bad Cannstatt

Genetic Algorithms for Protein Structure Prediction

RICHARD S. JUDSON
Genaissance Pharmaceuticals, New Haven, USA

MSC2000: 92B05

Article Outline

Keywords
See also
References

Keywords

Optimization; Evolution; Protein structure; Amino acid; Active site; Free energy; Conformation; Configuration; Primary structure; Tertiary structure; Cartesian coordinates; Internal coordinates; Bond distance; Bond angle; Dihedral angle; Rotamer library; Rotamer; Empirical potential; Nonbonded distance; Secondary structure

Genetic algorithms (GAs; cf. also ► Genetic algorithms) have been used for a large number of modeling applications in chemical and biological fields [5,9]. At least three factors contribute to this. First, GAs provide an easy-to-use global search and optimization approach. Second, they can easily handle noncontinuous functions. Finally, they are relatively robust even for moderately high-dimensional problems. All of these have contributed to the use of the GA for the important but computationally demanding field of protein structure prediction.

Proteins carry out a wide variety of functions in living cells, almost all of which require that the protein molecules assume precise 3-dimensional shapes [2,3]. Enzymes are typical examples. They generally consist of a large structure of 100–300 *amino acids* stabilizing a small *active site* which is designed to carry out a specific chemical reaction such as cleaving a bond in a target molecule. Even slight changes in the structure of the active site can destroy the protein's ability to function. Many drugs act by fitting snugly into enzymes' active sites, causing them to shut down. Therefore, a detailed understanding of the 3-dimensional structure of a protein can enhance our understanding of its function. This can in turn help understand related disease processes and can finally lead to disease cures. Unfortunately the experimental determination of protein structures, using *x*-ray crystallography or solution NMR is very difficult. Currently the structures of only a few thousand of the estimated 100,000 proteins that are used by the human body have been determined this way. The alternative is to predict the structures computationally.

The basic computational approach is simple to state, although many details have yet to be worked out. It relies on the experimental fact that a protein in solution (as well as any other molecule) will tend to find a state of low *free energy*. Free energy accounts for the internal energy (potential plus kinetic) of single molecules as well as the entropy of the ensemble of molecules of the same type. At absolute zero, the entropy contribution to the energy, as well as the kinetic energy, go to zero, leaving only the potential energy. Therefore, the most likely shape or state of a protein at absolute zero is the one of lowest potential energy. The simplest computational model then needs a method to search the space of conformations and an energy function (approximating the physical potential energy) which is minimized during the search. (A protein's *conformation* is the description of the 3-dimensional positions of all of the atoms for a fixed set of atoms and atom-atom connections. The *configuration* describes the atom-atom connectivity and only changes through chemical bond forming or breaking.) The conformation which yields the lowest

value of the energy function is a best estimate of conformation of the natural protein. It is possible to extend this simple model to include the effects of finite temperature, but these extensions are beyond the scope of this article. In-depth discussions of molecular modeling, including energy functions for proteins and other molecules can be found in [6,8,10], and [1].

Because proteins possess many degrees of freedom, and the energy functions have many local minima, global optimization methods that search efficiently and are not prone to being caught in local minima are required. The GA is often used because it fits both of these criteria.

Proteins [2] are long linear polymers composed of well-conserved sequences of the 20 amino acids. Each amino acid is in turn made up of a backbone

$$\begin{array}{ccccccc} & & & & R & & \\ & & & & | & & \\ - & (\mathrm{NH} & - & \mathrm{C}_\alpha & - & \mathrm{CO}) & - \end{array}$$

where R stands for one of the 20 side groups that make the amino acids unique. These range from a single hydrogen atom to chains having many degrees of freedom. The *primary structure* of the protein is simply the sequence of amino acids. For many naturally occurring proteins, this sequence carries sufficient information to determine the final 3-dimensional or *tertiary structure* of the protein. Experimentally, proteins that have been denatured (caused to unfold by heating the solution or changing its chemical composition) will spontaneously refold to their active, or native conformation, when the solution is returned to its original state.

There are two sets of coordinates often used for specifying the conformation of a protein. The first are the standard *Cartesian coordinates* for each atom. For N atoms, this requires $3N - 6$ numbers. The alternative is to use *internal coordinates* which are the *bond distances* (distances between atoms bound together), the *bond angles* (angles formed by a given atom and two atoms bound to it), and the *dihedral angles* (the angle of rotation about a center bond for a set of 4 atoms bound as A – B – C – D). To a good first approximation, the bond distances and bond angles are fixed at values that are independent of the particular amino acid or protein. Therefore, the conformation of a protein is determined largely by the values of its dihedral angles. There are on average about 15 atoms and about 3 dihedrals per amino acid, requiring about $N/5$ degrees of freedom to describe the conformation of an N-atom protein. The dimension of conformation space for a moderate-size protein of 100 amino acids ($\approx$ 1500 atoms) is $\approx$ 4500 when using Cartesian coordinates vs. $\approx$ 300 when using internal coordinates with fixed bond distances and angles.

In many protein structure prediction applications, the simple GA approach is used. For each generation, one calculates the fitness (energy) of each individual in the population, selects pairs of individuals based on their energy, performs crossover and mutation. The GA chromosome directly codes for the values of the dihedral angles. Both binary encoded and real number encoded chromosomes have been used with equal success. For binary encoded dihedrals, one must decide on the resolution of the GA search. The maximum one would use is 10 bits per angle which gives a resolution of about 1/3 degree. Often as few as 5 or 6 bits will be sufficient, especially if the GA-generated conformations will be subjected to local gradient minimization.

For each GA individual, the chromosome is decoded to give the values of the dihedrals which are passed to the energy function. This in turn returns an energy which is used as the fitness for the subsequent selection process.

Another encoding scheme that is often used is based on the idea of a *rotamer library*. It is known from studying the set of experimentally known structures that the dihedral angles in many amino acid side chains take on restricted sets of values. Also, the values of several neighboring dihedrals are often correlated. It has then been possible to develop libraries of preferred sidechain conformations (called *rotamers*) for each amino acid. This can be incorporated into the GA by having each word in the chromosome simply determine which of a set of rotamers to use for each amino acid in the sequence. The use of rotamer libraries in the GA framework is illustrated in references [7,12,13,14], and [11].

The other major ingredient needed for a protein structure prediction method is an energy function to be minimized. This is a huge area of research which is beyond the scope of this article, but two major approaches will be summarized. The first scheme uses physics-based *empirical potentials*. These are functions of the bond distances, bond angles, dihedral angles, and *nonbonded distances* (distances between atoms not directly

bound together). The functional forms are derived from the results of accurate but computationally expensive quantum mechanical calculations that are performed on small molecular fragments such as individual amino acids. The results are fitted to simple functions with several free parameters. The parameter values are either taken from the original quantum calculations or from independent spectroscopic experiments. Various methods are used to approximate the effect of the water and salt environment around the protein. The advantage of these potentials is that they are continuous and very general. They can be constructed for any protein and give reasonable energies for any conformation requested. The disadvantage is that they are not yet sufficiently accurate to give reliable structure predictions. For many if not all of the proteins whose structure is known, there are conformations that have much lower calculated energy than that of the experimental conformation.

The second approach is to use potentials based on observations of known protein structures. Basically, more probable conformations (ones that look more like real proteins) will have lower energy values. For instance certain sequences of amino acids almost always assume a particular *secondary structure*. The secondary structure of a protein describes the presence of multi-amino acid helices, sheets and turns but not the exact placement of the atoms in the secondary structure elements or the spatial orientation of these elements. These potentials have the advantage that they build on our observations of proteins as entire molecules and incorporate long-range order. As with the empirical potentials, though, they suffer from accuracy problems. However, except for very small proteins (less than 20 amino acids) the structure-based potentials show the most promise.

A common feature of GA-based protein structure prediction methods is the use of hybrid approaches combining standard GA with a local search method. The GA is then used primarily to perform an efficient global search which is biased towards regions of conformation space with low energy. This is a pragmatic approach driven by the large number of degrees of freedom even when internal coordinates are used. A simple and often used approach [5] is to subject GA-generated conformations to gradient minimization. Another approach is to use a population of individuals which carry out independent Monte-Carlo or simulated annealing walks (cf. also ► Simulated annealing methods in protein folding; ► Monte-Carlo simulated annealing in protein folding) for a number of steps and then undergo selection, crossover and mutation [4,15,16].

See also

- ► Adaptive Simulated Annealing and its Application to Protein Folding
- ► Bayesian Global Optimization
- ► Genetic Algorithms
- ► Global Optimization Based on Statistical Models
- ► Monte-Carlo Simulated Annealing in Protein Folding
- ► Packet Annealing
- ► Random Search Methods
- ► Simulated Annealing Methods in Protein Folding
- ► Stochastic Global Optimization: Stopping Rules
- ► Stochastic Global Optimization: Two-phase Methods

References

1. Allen MP, Tildesley DJ (1996) Computer simulation of liquids. Oxford Sci. Publ., Oxford
2. Branden C, Tooze J (1991) Introduction to protein structure. Garland Publ., Oxford
3. Creighton TE (1993) Proteins: structure and molecular properties. Freeman, New York
4. Friesner JR, Gunn A, Monge RA, Marshall GH (1994) Hierarchical algorithms for computer modeling of protein tertiary structure: folding of myoglobin to 6.2Å resolution. J Phys Chem 98:702
5. Judson RS (1997) Genetic algorithms and their use in chemistry. In: Lipkowitz KB, Boyd DB (eds) Rev. Computational Chemistry, vol 10. Wiley-VCH, Weinheim, pp 1–73
6. Karplus CL, Brooks M, Pettitt BM (1988) Proteins: A theoretical perspective of dynamics, structure and thermodynamics. Wiley/Interscience, New York
7. LeGrand S, Merz K (1993) The application of the genetic algorithm to the minimization of potential energy functions. J Global Optim 3:49
8. McCammon JA, Harvey S (1987) Dynamics of proteins and nucleic acids. Cambridge Univ. Press, Cambridge
9. Pedersen J, Moult J (1996) Genetic algorithms for protein structure prediction. Curr Opin Struct Biol 227–231
10. Rapaport DC (1995) The art of molecular dynamics simulation. Cambridge Univ. Press, Cambridge
11. Ring CS, Cohen FE (1994) Conformational sampling of loop structures using genetic algorithms. Israel J Chem 34:245

12. Sun S (1993) Reduced representation model of, protein structure prediction: statistical potential and genetic algorithms. Protein Sci 2:762
13. Tuffery P, Etchebest C, Hazout S, Lavery R (1991) A new approach to the rapid determiniation of protein side chain conformations. J Biomol Struct Dynam 8:1267
14. Tuffery P, Etchebest C, Hazout S, Lavery R (1993) A critical comparison of search algorithms applied to the optimization of protein side chain conformations. J Comput Chem 14:790
15. Unger R, Moult J (1993) Effects of mutations on the performance of genetic algorithms suitable for protein folding simulations. In: Tanaka M, Doyoma M, Kihara J, Yamamoto R (eds) Computer-Aided Innovation In New Materials. Elsevier, Amsterdam, pp 1283–1286
16. Unger R, Moult J (1993) Genetic algorithms for protein folding simulations. J Mol Biol 231:638

Geometric Programming

YANJUN WANG
Department of Applied Mathematics, Shanghai University of Finance and Economics, Shanghai, China

MSC2000: 90C28, 90C30

Article Outline

Keywords and Phrases
Introduction
Formulation
Methods and Applications
  Transformation
  Linear Relaxation Programming
  Branch-and-Bound Algorithm
  Algorithm Statement
Applications
References

Keywords and Phrases

Generalized geometric programming; Global optimization; Linear relaxation programming; Branch and bound

Introduction

Geometric programming is an important class of nonlinear optimization problems. Their source dates back to the 1960s when Zener began to study a special type of minimization cost problem for design in engineering, now known as geometric programming. The term geometric programming is adopted because of the crucial role that the arithmetic-geometric mean inequality plays in its initial development.

Actually, the early work in geometric programming was, for the most part, concerned with minimizing posynomial functions subject to inequality constraints on such functions, which was called posynomial geometric programming. In the past decade, because a number of models abstracted from application fields were not posynomial geometric programming, the theory had to be generalized to a much broader class of optimization problems called generalized geometric programming, which has spawned a wide variety of applications since its initial development. Its great impact has been in the areas of (1) engineering design [1,4,10,11]; (2) economics and statistics [2,3,6,9]; (3) manufacturing [8,17]; (4) chemical equilibrium [13,16]. Reference [19] focuses on solutions for generalized geometric programming.

Formulation

[19] provides a global optimization algorithm for the generalized geometric programming (GGP) problem stated as:

$$GGP \begin{cases} \min & G_0(x) \\ \text{s.t.} & G_m(x) \le \delta_m,\ m = 1, \ldots, M \\ & x \in X = \{x : 0 < x_i^l \le x_i \le x_i^u \\ & i = 1, \ldots, N\} \end{cases}$$

where $G_m(x) = \sum_{t=1}^{T_m} \delta_{mt} c_{mt} \prod_{i=1}^{N} x_i^{\gamma_{mti}}$, $m = 0, 1, \ldots, M$, and c_{mt} are positive coefficients, T_m are the given number of the terms in the function $G_m(x)$, $\delta_{mt} = +1$ and -1; $\delta_m = +1$ or -1, γ_{mti} are arbitrary real constant exponents. In general, formulation GGP corresponds to a nonlinear optimization problem with a nonconvex objective function and constraint set. In $G_m(x)$, if $\delta_{mt} = +1$ for all t, $t = 1, \ldots, T_m$, and $x_i > 0$, $i = 1, \ldots, N$, then the function $G_m(x)$ is called a posynomial. Note that if we set $\delta_{mt} = +1$ for all $m = 0, 1, \ldots, M$, $t = 1, \ldots, T_m$ and $\delta_m = +1$ for all $m = 1, \ldots, M$, then the GGP formulation reduces to the classical posynomial geometric programming (PGP) formulation that laid the foundation for the theory of the GGP problem.

Local optimization approaches for solving the GGP problem include three kinds of methods in general. First, successive approximation by posynomials, called "condensation," is the most popular [14]. Second, Passy and Wilde [15] developed a weaker type of duality, called "pseudo-duality," to accommodate this class of nonlinear optimization. Third, some nonlinear programming methods are adopted to solve the GGP problem based on exploiting the characteristics of the GGP problem [12].

Though local optimization methods for solving the GGP problem are ubiquitous, global optimization algorithms based on the characteristics of the GGP problem are scarce. Maranas and Floudas [13] proposed such a global optimization algorithm based on the exponential variable transformation of GGP, the convex relaxation, and branch and bound on some hyperrectangle region. Reference [19] proposes a branch-and-bound optimization algorithm that solves a sequence of linear relaxations over partitioned subsets in order to find a global solution, and to generate the linear relaxation of each subproblem and to ensure convergence to a global solution, special strategies have been applied. (1) The equivalent reverse convex programming (RCP) formulation is considered. (2) A linear relaxation method for the RCP problem is proposed based on the arithmetic-geometric mean inequality and the linear upper bound of the reverse convex constraints; this method is more convenient with respect to computation than the convex relaxation method [13]. (3) A bound tightening method is developed that will enhance the solution procedure, and, based on this method, a branch-and-bound algorithm is proposed.

Methods and Applications

Transformation

In [5], Duffin and Peterson show that any GGP problem can be transformed into the following reverse posynomial geometric programming (RPGP):

$$\begin{cases} \min & x_0 \\ \text{s.t.} & g_m(x) \le 1, \quad m = 1, \ldots, p \\ & g_m(x) \ge 1, \quad m = p+1, \ldots, q \\ & x \in \Omega_0 = \{x : 0 < x_i^l \le x_i \le x_i^u < \infty \\ & i = 0, \ldots, n\} \end{cases}$$

where $g_m(x)$ are posynomials for $m = 1, \ldots, q$, and $n \ge N$.

To see how such a reformulation is possible, first consider the objective function in GGP. If the optimal value of GGP is positive, the GGP problem is equivalent to the following form:

$$(GGP1): \begin{cases} \min & x_0 \\ \text{s.t.} & x_0^{-1} G_0(x) \le 1, \\ & G_m(x) \le \delta_m, m = 1, \ldots, M \\ & x \in X . \end{cases}$$

And if the optimal value of GGP is negative, then GGP can be transformed into the following form:

$$(GGP2): \begin{cases} \min & x_0 \\ \text{s.t.} & x_0 G_0(x) \le -1 , \\ & G_m(x) \le \delta_m, m = 1, \ldots, M \\ & x \in X . \end{cases}$$

We can add a large constant to the objective function of GGP in order to ensure that the optimal value of (GGP) is positive, then derive the form GGP1. In this method a probably lower bound estimation for the optimal value of GGP is needed.

Secondly we turn to consider the constraints. If the primal constrained function $G_m(x)$ is either a posynomial or the negative of a posynomial, then it is obvious. So we only consider the following constrained function:

$$G_m(x) = h_1(x) - h_2(x) \le 1 ,$$

where each $h_i(x)(i = 1, 2)$ is a posynomial. Notice that x satisfies the above inequality if and only if there exists a single variable $s > 0$ such that (x, s) satisfies

$$h_1(x) \le s \le h_2(x) + 1 .$$

Now note that the above formulation is equivalent to the following two constraints

$$s^{-1} h_1(x) \le 1 \quad \text{and} \quad s^{-1} h_2(x) + s^{-1} \ge 1 ,$$

which are in a form consistent with the formulation RPGP.

By applying the following exponent transformation

$$x_i = \exp z_i, \quad i = 0, \ldots, n$$

to the formulation *RPGP*, we can obtain the following reverse convex programming (RCP) problem:

$$\begin{cases} \min & \exp(z_0) \\ \text{s.t.} & g_m(z) \le 1,\ m = 1, \dots, p \\ & g_m(z) \ge 1,\ m = p+1, \dots, q \\ & z \in \Omega = \{z: z_i^L \le z_i \le z_i^U, \\ & i = 0, 1, \dots, n\} \end{cases}$$

where

$$g_m(z) = \sum_{t=1}^{T_m} c_{mt} \exp\left\{\sum_{i=0}^{n} \gamma_{mti} z_i\right\}, \quad m = 1, \dots, q$$

Because each $\exp\{\sum_{i=0}^{n} \gamma_{mti} z_i\}$ is convex, both the objective and constrained functions are convex.

The main difficulty for solving the RCP problem is connected with the presence of the reverse convex constraints $g_m(z) \ge 1,\ m = p+1, \dots, q$, which destroy the convexity and possibly even the connectivity of the feasible set and give rise to a nonconvex feasible region.

Linear Relaxation Programming

The principal construct in the development of a solution procedure for solving the RCP problem is the construction of a linear relaxation programming of RCP for obtaining the lower bound for this problem, as well as for its partitioned subproblems [19] derives such a linear relaxation by applying the arithmetic-geometric mean inequality for the convex constraints and overestimating every reverse convex constraint in either the initial bounds on the variables of the problem or modified bounds as defined for some partitioned subproblem in a branch-and-bound scheme.

(1) Linear Relaxation for Convex Constraints The arithmetic-geometric mean inequality that played such a crucial role in developing the duality theory for posynomial programming is also used to obtain linear relaxation programming. Recall that this inequality states that for any vector $\omega > 0$ and any nonnegative weight vector ε whose components sum to one, we have

$$\sum_t \omega_t \ge \prod_t \left(\frac{\omega_t}{\varepsilon_t}\right)^{\varepsilon_t}$$

provided $(\omega_t/\varepsilon_t)^{\varepsilon_t}$ is defined to be 1 when $\varepsilon_t = 0$. Give a posynomial

$$g_m(x) = \sum_t u_{mt}(x) = \sum_t c_{mt} \prod_i x_i^{mti}$$

and $\varepsilon_m \ge 0$ with $\sum_t \varepsilon_{mt} = 1$. Then a condensed posynomial $\bar{g}_m(x)$ is defined by

$$\bar{g}_m(x) = \bar{c}_m \prod_i x_i^{\bar{\gamma}_{mi}}$$

where $\bar{c}_m = \prod_t (c_{mt}/\varepsilon_{mt})^{\varepsilon_{mt}}$ and $\bar{\gamma}_{mi} = \sum_t \gamma_{mti}\varepsilon_{mt}$.

Thus the condensed posynomial $\bar{g}_m(x)$ is also a posynomial, and it has a single posynomial term. According to this method, the condensed single term for the convex constraints $g_m(z) \le 1$ of RCP, where $z_i = \ln x_i$, is of the following form:

$$\bar{g}_m(z) = \bar{c}_m \exp\left(\sum_i \bar{\gamma}_{mi} z_i\right) \tag{1}$$

where the definitions of $\bar{c}_m$ and $\bar{\gamma}_{mi}$ have been given in the former.

To illustrate how the condensed term can be used to obtain the linear relaxation, we consider the following convex constraints $g_m(z) \le 1, m = 1, \dots, p$ and select an arbitrary weight vector $\varepsilon_m \ge 0$ whose components sum to one. We use the condensed constrained functions to replace the above convex constraints:

$$\bar{g}_m(z) \le 1, \quad m = 1, \dots, p. \tag{2}$$

It follows that

$$\bar{g}_m(z) \le g_m(z)$$

for each $m = 1, \dots, p$. Thus if in RCP the convex constraints are replaced by the condensed constraints, the feasible region for RCP will be contained in the new feasible region. Notice that the condensed constraints (2) can be easily transformed into equivalent formulations as linear constraints:

$$L_m(z) = \sum_i \bar{\gamma}_{mi} z_i + \ln \bar{c}_m \le 0,\ m = 1, \dots, p.$$

(2) Linear Relaxation for Reverse Convex Constraints For reverse convex constraints such a linear relaxation can be obtained by overestimating every convex func-

tion $g_m(z)$ of the reverse convex constraint with a linear function $L_m(z)$ for every $m = p+1, \ldots, q$. The method in [13] of underestimating a concave function with a linear function is adopted, and we describe the linear function as follows:

$$L_m(z) = \sum_{t=1}^{T_m} c_{mt} \left\{ A_{mt} + B_{mt} \left(\sum_{i=0}^{n} \gamma_{mti} z_i \right) \right\}$$

and

$$A_{mt} = \frac{Y_{mt}^U \exp(Y_{mt}^L) - Y_{mt}^L \exp(Y_{mt}^U)}{Y_{mt}^U - Y_{mt}^L} ,$$

$$B_{mt} = \frac{\exp(Y_{mt}^U) - \exp(Y_{mt}^L)}{Y_{mt}^U - Y_{mt}^L} ,$$

$$Y_{mt}^L = \sum_{i=0}^{n} \min(\gamma_{mti} z_i^L, \gamma_{mti} z_i^U) ,$$

$$Y_{mt}^U = \sum_{i=0}^{n} \max(\gamma_{mti} z_i^L, \gamma_{mti} z_i^U) ,$$

and it follows that

$$L_m(z) \geq g_m(z), \ m = p+1, \ldots, q .$$

Thus if in (RCP) the reverse convex constraints are replaced by the overestimation linear constraints, the feasible region for RCP will be contained in the new feasible region.

(3) Linear Relaxation Programming For the objective function of RCP, it is obvious that $\min \exp(z_0)$ is equivalent to $\min z_0$. From the above discussion for the two kinds of constraints respectively, [19] constructs the corresponding linear relaxation programming on the region Ω $LRP(\Omega)$ as follows:

$$\begin{cases} \min & z_0 \\ \text{s.t.} & L_m(z) \leq 0, \ m = 1, \ldots, p \\ & L_m(z) \geq 1, \ m = p+1, \ldots, q \\ & z \in \Omega = \{z : z_i^L \leq z_i \leq z_i^U , \\ & i = 0, 1, \ldots, n\} . \end{cases}$$

The following results establish some salient properties of the linear relaxation programming $LRP(\Omega)$ that are essential in designing the proposed algorithm.

Lemma 1 *Assume the minimum of $LRP(\Omega)$ is LB^*; then $\exp(LB^*)$ provides a lower bound of the optimal value of the RCP problem.*

Proof We denote the feasible region of RCP and $LRP(\Omega)$ D and P; then it is immediate that $P \supseteq D$ by the construction method. So based on the above assumption, $\exp(LB^*)$ is a lower bound of the minimum of the RCP problem. □

Branch-and-Bound Algorithm

Reference [19] develops a branch-and-bound algorithm to solve the RCP based on the former linear relaxation method. This algorithm needs to solve a sequence of linear relaxation programming problems over Ω or the subsets of Ω in order to find a global solution. Furthermore, to ensure convergence to a global solution, a new bound tightening method (BTM) is proposed and will be applied to enhance the solution procedure.

The critical element in guaranteeing convergence to a global minimum is the choice of a suitable branching rule. In [18] three kinds of branching methods are provided. Reference [19] chooses the first method, a simple and standard bisection rule. This method is sufficient to ensure convergence since it drives all the intervals to zero for the variables that are associated with the term that yields the greatest discrepancy in the employed approximation along any infinite branch of the branch-and-bound tree.

Branching rule:

Assume that the hyperrectangle Ω^q is going to be divided. Then the selection of the branching variable z_e, which possesses a maximum length in Ω^q and the partitioning of Ω^q are done using the following rules, where $\Omega^q = \{z : z_j^L(\Omega^q) \leq z_j \leq z_j^U(\Omega^q), \ j = 0, \ldots, n\}$. Let

$$e = \arg\max \left\{ z_j^U(\Omega^q) - z_j^L(\Omega^q) \right\} ,$$

and partition Ω^q by bisecting the interval $[z_e^L(\Omega^q), z_e^U(\Omega^q)]$ into the subintervals $[z_e^L(\Omega^q), (z_e^L(\Omega^q) + z_e^U(\Omega^q))/2]$ and $[(z_e^L(\Omega^q) + z_e^U(\Omega^q))/2, z_e^U(\Omega^q)]$.

In what follows we describe the BTM strategy proposed by [19].

Assume that the subhyperectangle $\Omega^{q(s)}$ (s is the iteration counter) is selected for further consideration. If in the node $q(s)$ the corresponding solution $\hat{z}(\Omega^{q(s)})$ is

not feasible in some convex constraint, let

$$l = \arg\max\{g_m(\hat{z}(\Omega^{q(s)}))|$$
$$g_m(\hat{z}(\Omega^{q(s)})) = \sum_{t=1}^{T_m} u_{mt}(\hat{z}(\Omega^{q(s)})) > 1\}\,.$$

Compute the weight vector $\bar{\varepsilon}_l$ by $\bar{\varepsilon}_{li} = u_{li}(\hat{z})/g_l(\hat{z})$, $i = 1, \ldots, T_l$, and then condense the function $g_l(z)$ using this weight vector as described in Sect. "Linear Relaxation Programming." Then a new single term is obtained, and therefore a new linear constraint is added to the linear relaxation programming $LRP(\Omega^{q(s)})$. Denote this new linear relaxation programming and new added condensed single term $\overline{LRP}(\Omega^{q(s)})$ and $\bar{g}_l(z)$. And from the discussion in Sect. "Linear Relaxation Programming" we know $\bar{g}_l(z) = \bar{c}_l \exp(\sum_i \bar{\gamma}_{li} z_i)$, where $\bar{c}_l = \prod_t (c_{lt}/\bar{\varepsilon}_{lt})^{\bar{\varepsilon}_{lt}}$ and $\bar{\gamma}_{li} = \sum_t \gamma_{lti}\bar{\varepsilon}_{lt}$.

It is obvious that

$$\bar{g}_l(\hat{z}(\Omega^{q(s)})) = g_l(\hat{z}(\Omega^{q(s)}))\,,$$

and since $g_l(\hat{z}(\Omega^{q(s)})) > 1$, it follows that $\hat{z}(\Omega^{q(s)})$ does not satisfy the new added constraint $\bar{g}_l(z) \le 1$. From the arithmetic-geometric mean inequality, we have $\bar{g}_l(z) \le g_l(z)$. Of course, the new single-term constraint $\bar{g}_l(z) \le 1$ is equivalent to a linear constraint. Hence, if z is feasible for RCP, it is certainly feasible for $\overline{LRP}(\Omega^{q(s)})$, whose feasible region obviously does not contain the point $\hat{z}(\Omega^{q(s)})$. Clearly, this BTM technique will enhance the solution procedure.

Based on the previous BTM technique, [19] constructs the global optimization algorithm. The basic steps of the algorithm are summarized in the following statement.

Algorithm Statement

step 0: (Initialization)

0.1: Assume a convergence tolerance δ>0, and the initial weights $\varepsilon_m, m = 1, \ldots, p$. Set the iteration counter $s = 0$, then $Q_s = Q_0 = \{1\}, q(s) = q(0) = 1$, $\Omega^{q(s)} = \Omega^1 = \Omega$. Set an initial upper bound $U^* = \infty$.

0.2: Solve the problem $LRP(\Omega^{q(s)})$, and denote the solution and the minimum $(\hat{z}(\Omega^{q(s)}), LB_{q(s)})$.

0.3: If $\hat{z}(\Omega^{q(s)})$ is feasible for RCP, then stop with $\hat{z}(\Omega^{q(s)})$ as the prescribed solution to the RCP problem, else let $LB(s) = LB_{q(s)}$;

0.4: If $\hat{z}(\Omega^{q(s)})$ is not feasible on some convex constraints, the BTM technique will be adopted.

step 1: (Partitioning step) Choose a branching variable z_e, then partition $\Omega^{q(s)}$ to get $\Omega^{q(s).1}$ and $\Omega^{q(s).2}$. Replace $q(s)$ by node indices $q(s).1, q(s).2$ in Q_s.

step 2: (Feasibility check for (RCP)) For each $q(s).w$, where $w = 1, 2$, compute:

$$g_m(w) = \bar{c}_m \exp\left(\sum_{i=0}^{n} \min(\bar{\gamma}_{mi} z_i^L, \bar{\gamma}_{mi} z_i^U)\right)\,,$$
$$\text{for } m = 1, \ldots, p$$
$$g_m(w) = \sum_{t=1}^{T_m} c_{mt} \exp\left(Y_{mt}^U\right)\,,$$
$$\text{for } m = p+1, \ldots, q$$

where $\bar{c}_m, \gamma_{mi}, Y_{mt}^U$ have been defined in Sect. "Linear Relaxation Programming." If for some $m \in \{1, \ldots, p\}$, $g_m(z)$>1, or for some $m \in \{p+1, \ldots, q\}$, $g_m(z) < 1$, then the node indices $q(s).w$ will be eliminated. If $\Omega^{q(s).w}(w = 1, 2)$ are all eliminated, then go to step 5.

step 3: (Updating upper bound) For undeleted sub-hyperrectangle update

$$A_{mt}, B_{mt}, Y_{mt}^L, Y_{mt}^U\,.$$

Solve $LRP(\Omega^{q(s).w})$, where $w = 1$ or $w = 2$ or $w = 1, 2$, and denote the solutions and optimal values $(\hat{z}(\Omega^{q(s).w}), LB_{q(s).w})$. Then if $\hat{z}(\Omega^{q(s).w})$ is feasible for RCP, $U^* = \min\{U^*, LB_{q(s).w}\}$.

step 4: (Deleting step) If $LB_{q(s).w} > U^* + \delta$, then delete the corresponding node;

step 5: (Fathoming step) Fathom any nonimproving nodes by setting $Q_{s+1} = Q_s - \{q \in Q_s : LB_q \ge U^* - \delta\}$. If $Q_{s+1} = \emptyset$, then stop, and $\exp(U^*)$ is the optimal value, $z^*(\kappa)$ (where $\kappa \in \kappa_0$) are the global solutions, where $\kappa_0 = \{\kappa : z_0^*(\kappa) = U^*\}$. Otherwise, $s = s + 1$;

step 6: (Node-selection step) Set $LB(s) = \min\{LB_q : q \in Q_s\}$, then select an active node $q(s) \in \arg\min\{LB(s)\}$ for further considering;

step 7: (Bound tightening step) If in this node $q(s)$, $\hat{z}(\Omega^{q(s)})$ is feasible in all convex constraints of RCP, then return to step 1, else the BTM technique will be adopted, and then return to step 1.

Theorem 1 (convergence result) *The above algorithm either terminates finitely with the incumbent solution being optimal to RCP or it generates an infinite sequence of iterations such that along any infinite branch of the branch-and-bound tree, any accumulation point of the sequence LB(s) will be the global minimum of the RCP problem.*

Proof A sufficient condition for a global optimization to be convergent to the global minimum, stated in Horst and Tuy [7], requires that the bounding operation be consistent and the selection operation bound improving.

A bounding operation is called consistent if at every step any unfathomed partition can be further refined and if any infinitely decreasing sequence of successively refined partition elements satisfies:

$$\lim_{s\to+\infty}(U^* - LB(s)) = 0\ , \tag{3}$$

where $LB(s)$ is a lower bound inside some subhyperrectangle in stage s and U^* is the best upper bound at iteration s, not necessarily occurring inside the above same subhyperrectangle. In the following we will demonstrate that (3) holds.

Since the employed subdivision process is the bisection, the process is exhaustive. Consequently, from the discussion in [13] (3) holds, and this means that the employed bounding operation is consistent.

A selection operation is called bound improving if at least one partition element where the actual lower bound is attained is selected for further partition after a finite number of refinements. Clearly, the employed selection operation is bound improving because the partition element where the actual lower bound is attained is selected for further partition in the immediately following iteration.

In summary, it is shown that the bounding operation is consistent and that the selection operation is bound improving; therefore, according to Theorem IV.3. in Horst and Tuy [7], the employed global optimization algorithm is convergent to the global minimum. □

Applications

Reference [19] reports the numerical experiment for the deterministic global optimization algorithm described above to demonstrate its potential and feasibility. The experiment is carried out with the C programming language. The simplex method is applied to solve the linear relaxation programming problems.

To illustrate how the proposed algorithm works, first [19] gives a simple example to show the solving procedure of the proposed algorithm.

Example 1:

$$\begin{cases} \min & x_1^2 + x_2^2 \\ \text{s.t.} & 0.3x_1x_2 \geq 1 \\ & x \in X = \{2 \leq x_1 \leq 5; \\ & 1 \leq x_2 \leq 3\}\ . \end{cases}$$

First, transform the above problem into the RPGP form as follows:

$$\begin{cases} \min & x_0 \\ \text{s.t.} & g_1(x) = x_0^{-1}x_1^2 + x_0^{-1}x_2^2 \leq 1 \\ & g_2(x) = 0.3x_1x_2 \geq 1 \\ & x \in \Omega_0 = \{x \mid 5 \leq x_0 \leq 10; \\ & 2 \leq x_1 \leq 5;\ 1 \leq x_2 \leq 3\}\ . \end{cases}$$

Let $x_i = \exp z_i\ (i = 0, 1, 2)$, then we can obtain the following reverse convex programming problem (P) of Example 1 :

$$\begin{cases} \min & \exp(z_0) \\ \text{s.t.} & f_1(z) = \exp(-z_0 + 2z_1) \\ & + \exp(-z_0 + 2z_2) \leq 1 \\ & f_2(z) = 0.3\exp(z_1 + z_2) \geq 1 \\ & z \in \Omega = \{z \mid \\ & 1.6094 \leq z_0 \leq 2.3026; \\ & 0.6931 \leq z_1 \leq 1.6094; \\ & 0 \leq z_2 \leq 1.0986\}\ . \end{cases}$$

In step 0, set $\delta = 10^{-3}$, s=0, $U^* = \infty$. For the convex constraint function $f_1(z)$, choose the initial weight as $\varepsilon_1 = (1/2, 1/2)$ since it has two terms. Then $q(s) = 1, Q_s = Q_0 = \{1\}$, $\Omega^{q(s)} = \Omega^1 = \Omega$. According to the discussion in Sect. "Methods and Applica-

tions", the $LRP(\Omega^1)$ of problem P is formulated below:

$$\begin{cases} \min & z_0 \\ \text{s.t.} & L_1(z) = -z_0 + z_1 + z_2 \le -0.6931 \\ & L_2(z) = 1.9356z_1 + 1.9356z_2 \ge 1.7416 \\ & z \in \Omega^1 . \end{cases}$$

The solution and optimal value of $LRP(\Omega^1)$ are:

$$\hat{z}(\Omega^1) = (1.6094, 0.6931, 0.2231) ,$$
$$LB_1 = 1.6094 .$$

Since $\hat{z}(\Omega^1)$ is not feasible for problem P, then $LB(s) = LB(0) = 1.6094$. Since $\hat{z}(\Omega^1)$ is not feasible for $f_1(z) \le 1$, then the BTM technique will be adopted. First, update the weight ε_1 according to the solution $\hat{z}(\Omega^1)$, and derive $\varepsilon_1 = (0.7191, 0.2809)$, then from formula (1) in Sect. "Methods and Applications", we obtain a new linear constraint:

$$L_3(z) = -z_0 + 1.4382z_1 + 0.5618z_2 \le -0.5938 .$$

The current linear relaxation programming denoted as $\overline{LRP}(\Omega^1)$ is:

$$\begin{cases} \min & z_0 \\ \text{s.t.} & L_1(z) = -z_0 + z_1 + z_2 \le -0.6931 \\ & L_2(z) = 1.9356z_1 + 1.9356z_2 \ge 1.7416 \\ & L_3(z) = -z_0 + 1.4382z_1 + 0.5618z_2 \\ & \le -0.5938 \\ & z \in \Omega^1 . \end{cases}$$

In step 1, divide the region Ω^1 into the following two regions:

$$\Omega^2 = \{z \,|\, 1.6094 \le z_0 \le 2.3026 ;$$
$$0.6931 \le z_1 \le 1.6094;\ 0 \le z_2 \le 0.5493\} ,$$
$$\Omega^3 = \{z \,|\, 1.6094 \le z_0 \le 2.3026 ;$$
$$0.6931 \le z_1 \le 1.6094;\ 0.5493 \le z_2 \le 1.0986\} ,$$

then the node set $Q_0 = \{2, 3\}$.

In step 2, the two nodes in Q_0 have not been deleted; then go to step 3. After updating the parameters according to the formula in Sect. "Linear Relaxation Programming" respectively, we can obtain the new function $L_2(z)$ in each node. Then we have $LRP(\Omega^2)$:

$$\begin{cases} \min & z_0 \\ \text{s.t.} & L_1(z) = -z_0 + z_1 + z_2 \le -0.6931 \\ & L_2(z) = 1.3633z_1 + 1.3633z_2 \ge 1.3450 \\ & L_3(z) = -z_0 + 1.4382z_1 + 0.5618z_2 \\ & \le -0.5938 \\ & z \in \Omega^2 \end{cases}$$

and we have $LRP(\Omega^3)$:

$$\begin{cases} \min & z_0 \\ \text{s.t.} & L_1(z) = -z_0 + z_1 + z_2 \le -0.6931 \\ & L_2(z) = 2.3613z_1 + 2.3613z_2 \ge 2.8946 \\ & L_3(z) = -z_0 + 1.4382z_1 + 0.5618z_2 \\ & \le -0.5938 \\ & z \in \Omega^3 . \end{cases}$$

The solutions and optimal values are respectively

$$\hat{z}(\Omega^2) = (1.7555, 0.6931, 0.2934) ,$$
$$LB_2 = 1.7555$$
$$\hat{z}(\Omega^3) = (1.9356, 0.6931, 0.8427) ,$$
$$LB_3 = 1.9356 .$$

In step 4 the two nodes have not been deleted; then go to step 5. Compute

$$Q_1 = Q_0 - \{q \in Q_0 \colon LB_q \ge U^* - \delta\} = \{2, 3\} ,$$

and set $s = 1$. In step 6, the current lower bound is

$$LB(s) = LB(1) = \min\{LB_q, q \in Q_s\}$$
$$= \min\{LB_2, LB_3\} = 1.7555 .$$

So we will choose the active node as $q(1) = 2$ for further consideration.

In step 7 in the node $q(1)$, the BTM technique is adopted. From formula (1) in Sect. "Methods and Applications" we compute the new weight $\varepsilon_1 = (0.6899, 0.3101)$ according to the solution $\hat{z}(\Omega^2)$,

and we obtain the following new linear constraint:

$$L_4(z) = -z_0 + 1.3797z_1 + 0.6203z_2 \le 1.2919 .$$

The current linear relaxation programming denoted as $\overline{LRP}(\Omega^2)$ is:

$$\begin{cases} \min & z_0 \\ \text{s.t.} & L_1(z) = -z_0 + z_1 + z_2 \le -0.6931 \\ & L_2(z) = 1.3633z_1 + 1.3633z_2 \ge 1.3450 \\ & L_3(z) = -z_0 + 1.4382z_1 + 0.5618z_2 \\ & \le -0.5938 \\ & L_4(z) = -z_0 + 1.3797z_1 + 0.6203z_2 \\ & \le 1.2919 \\ & z \in \Omega^2 . \end{cases}$$

Then return to step 1, divide the region Ω^2, and go into a new circle. After 22 iterations, the procedure stops. The global minimum of problem P is 1.9140, and the global solution is

$$z^* = (1.9140, 0.6933, 0.5107) .$$

Then the global minimum of example 1 is 6.7804, and the global solution is $x^* = (2.0003, 1.6664)$.

Additionally, to test the algorithm, [19] chooses five examples, all of which are taken from engineering, concerning the detailed application context, please refer to the releveant references.

Example 2 ([1]):

$$\begin{cases} \min & x_0 \\ \text{s.t.} & x_0^{-1}x_2^{-1}x_3^{-1}x_5 + 5x_0^{-1}x_1^{\frac{1}{2}}x_4x_5 \le 1 \\ & x_2^{\frac{1}{3}}x_3 - x_4^{\frac{1}{2}} \le -1 \\ & -x_5 - 2x_0x_1x_2x_3^4x_4^{-1}x_5 \le -1 \\ & x \in X = \{x \mid 30 \le x_0 \le 40; \\ & 0.01 \le x_1 \le 1; \\ & 0.0001 \le x_2 \le 1; \\ & 15 \le x_3 \le 20; \\ & 15 \le x_4 \le 20; \\ & 0.1 \le x_5 \le 1\} . \end{cases}$$

Example 3 ([11]):

$$\begin{cases} \min & x_0 \\ \text{s.t.} & 0.274x_3x_4^4 + 2520.66x_1x_4^5 + x_0x_3^2 \\ & -x_0x_1x_2x_3 + 1 \le 1 \\ & x_1x_2^{-1}x_3 \le 1 \\ & x_1x_4^4 \le 1 \\ & x_3x_4^3 \le 1 \\ & x \in X = \{x \mid 10^{-12} \le x_0 \le 2; \\ & 20 \le x_1 \le 35; \\ & 120 \le x_2 \le 160; \\ & 1 \le x_3 \le 10; \\ & 10^{-6} \le x_4 \le 1\} . \end{cases}$$

Example 4 ([20]):

$$\begin{cases} \min & x_0 \\ \text{s.t.} & 3.7x_0^{-1}x_1^{0.85} + 1.985x_0^{-1}x_1 \\ & +700.3x_0^{-1}x_2^{-0.75} \le 1 \\ & 0.7673x_2^{0.05} - 0.05x_1 \le 1 \\ & x \in X = \{x \mid 5 \le x_0 \le 15; \\ & 0.1 \le x_1 \le 5; \\ & 380 \le x_2 \le 450\} . \end{cases}$$

Example 5 ([20]):

$$\begin{cases} \min & x_0 \\ \text{s.t.} & 4x_1 - 4x_0^2 \le 1 \\ & -x_0 - x_1 \le -1 \\ & x \in X = \{x \mid 0.01 \le x_0 \le 15; \\ & 0.01 \le x_1 \le 15\} . \end{cases}$$

Example 6 ([5]):

$$\begin{cases} \min & x_3^{0.8}x_4^{1.2} \\ \text{s.t.} & x_1x_4^{-1} + x_2^{-1}x_4^{-1} \le 1 \\ & -x_1^{-2}x_3^{-1} - x_2x_3^{-1} \le -1 \\ & x \in X = \{x \mid 0.1 \le x_1 \le 1; \\ & 5 \le x_2 \le 10; \\ & 8 \le x_3 \le 15; \\ & 0.01 \le x_4 \le 1\} . \end{cases}$$

The following table summarizes the computational results on the above five examples. In the table s denotes

the number of the iteration, L denotes the longest node number in Q_s described in the algorithm statement, and δ denotes the convergence tolerance. The results show that the algorithm of [19] can globally solve the GGP problem effectively.

No.	Solution
2	(37.0070,0.4489,0.0048,18.0348,16.0449,0.5667)
3	(0.0000, 32.7781,155.0000, 4.7288, 0.0027)
4	(11.9637, 0.8098, 442.0915)
5	(0.5, 0.5)
6	(0.1020, 7.0711, 8.3284, 0.2434)

no	s	L	δ	CPU time
2	131	28	10^{-3}	4s
3	191	74	10^{-6}	6s
4	138	39	10^{-6}	5s
5	96	10	10^{-9}	1s
6	146	42	10^{-6}	6s

References

1. Avriel M, Williams AC (1971) An extension of geometric programming with applications in engineering optimization. J Eng Math 5(3):187–199
2. Bricker DL, Kortanek KO, Xu L (1995) Maximum likelihood estimates with order restrictions on probabilities and odds ratios: a geometric programming approach. J Appl Math Decis Sci 1(1):53–65, The University of IA, Iowa City
3. Choi JC, Bricker DL (1996) Effectiveness of a geometric programming algorithm for optimization of machining economics models. Comput Oper Res 23(10):957–961
4. Das K, Roy TK, Maiti M (2000) Multi-item inventory model with under imprecise objective and restrictions: a geometric programming approach. Product Plann Control 11(8):781–788
5. Duffin RJ, Peterson EL (1973) Geometric programming with signomial. J Optim Theory Appl 11(1):3–35
6. El Barmi H, Dykstra RL (1994) Restricted multinomial maximum likelihood estimation based upon Fenchel duality. Statist Probab Lett 21:121–130
7. Horst R, Tuy H (1990) Global optimization, Deterministic Approaches. Springer, Berlin
8. Sonmez AI, Baykasoglu A, Dereli T, Flz IH (1999) Dynamic optimization of multipass milling operations via geometric programming. Int J Mach Tools Manufact 39:297–320
9. Jagannathan R (1990) A stochastic geometric programming problem with multiplicative recourse. Oper Res Lett 9:99–104
10. Jefferson TR, Scott CH (1978) Generalized geometric programming applied to problems of optimal control: I. Theory. JOTA 26:117–129
11. Jha NK (1995) Geometric programming based robot control design. Comput Ind Eng 29(1–4):631–635
12. Kortanek KO, Xu X, Ye Y (1996) An infeasible interior-point algorithm for solving primal and dual geometric programs. Math Program 76:155–181
13. Maranas CD, Floudas CA (1997) Global optimization in generalized geometric programming. Comput Chem Eng 21(4):351–369
14. Passy U (1971) Generalized weighted mean programming. SIAM J Appl Math 20:763–778
15. Passy U, Wilde DJ (1967) Generalized polynomial optimization. J Appl Math 15(5):1344–1356
16. Rijckaert MJ, Martens XM (1974) Analysis and optimization of the Williams–Otto process by geometric programming. AIChE J 20(4):742–750
17. Scott CH, Jefferson TR (1995) Allocation of resources in project management. Int J Syst Sci 26:413–420
18. Sherali HD (1998) Global optimization of nonconvex polynomial programming problems having ratilnal exponents. J Global Optim 12:267–283
19. Wang Y, Zhang K, Gao Y (2004) Global optimization of generalized geometric programming. Comput Math Appl 48:1505–1516
20. Zhang K (1990) Geometric Programming and Optimal Design. Xi'an Jiaotong University publishing house, China

Global Equilibrium Search

ERIKA J. SHORT, OLEG V. SHYLO
Center for Applied Optimization,
Department of Industrial and Systems Engineering,
University of Florida, Gainesville, USA

Article Outline

Abstract
References

Abstract

Global equilibrium search is a method that can be applied to a variety of hard optimization problems. The algorithm utilizes ideas similar to those of the simulated annealing method. The algorithm accumulates information about the search space in order to generate new solutions for the subsequent stages. This method has been successfully applied to well-known problems such as the multidimensional knapsack problem, the job-shop scheduling problem, the unconstrained

quadratic programming problem, the maximum satisfiability problem, etc.

The numerous discrete optimization problems that arise in practice have such different characteristics that development of a general purpose solution method is clearly impracticable. One way of tackling this issue is to develop a library of suitable solution methods, allowing the practitioner to choose the most suitable for his problem under his time constraints and quality requirements. In recent decades, heuristic approaches, such as tabu search [1], simulated annealing (SA) [3], etc., have gained a considerable amount of attention from the scientific community for being the only practical tool that can be applied to a wide range of difficult problems. Global equilibrium search (GES) offers another highly effective tool for solving large-scale optimization problems.

The method was introduced by Shylo [7] in 1999. It shares ideas similar to those that inspired the SA technique, while providing, in practice, faster asymptotic convergence to the optimal solution on a wide class of optimization problems. Moreover, the GES method can be used in an ensemble with other techniques, which makes it more versatile than most of its predecessors.

Consider a discrete optimization problem of the following form:

$$\min\{f(x) : x \in S : S \subseteq \{0,1\}^n\} \tag{1}$$

where f is some quality function. Let us introduce a random binary vector ξ that takes a value from a feasible set S according to the Boltzmann distribution, with $\mu \geq 0$ being the temperature parameter:

$$P\{\xi(\mu) = x\} = \frac{\exp(-\mu f(x))}{\sum_{x \in S} \exp(-\mu f(x))} . \tag{2}$$

Consider the SA method applied to problem (1). It can be shown easily that under certain conditions (i. e., symmetric neighborhood structure) the stationary probabilities of the Markov chain associated with the SA method are given by (2).

Set S can be split into two subsets in such a way that one of them contains the feasible solutions for which the jth component is 1, and another set will contain the solution with the jth component equal to 0. Let us name these two sets S_j^1 and S_j^0. Obviously, $S_j^1 \cup S_j^0 = S$. Then the probability of the jth component of ξ being 1 can be expressed as

$$p_j(\mu) \equiv P\{\xi_j(\mu) = 1\} = \frac{\sum_{x \in S_j^1} \exp(-\mu f(x))}{\sum_{x \in S} \exp(-\mu f(x))} . \tag{3}$$

The idea of the GES method is to use some subset of known solutions $\widehat{S}$ to generate new solutions in the successive stages of the algorithm. The distribution (3) or any other equivalent formula [4] can be used for such a generation (substituting S with $\widehat{S}$ in the formula):

$$\widehat{p}_j(\mu) \equiv P\{\widehat{\xi}_j(\mu) = 1\} = \frac{\sum_{x \in \widehat{S}_j^1} \exp(-\mu f(x))}{\sum_{x \in \widehat{S}} \exp(-\mu f(x))} . \tag{4}$$

If $\arg\min\{f(x) : x \in \widehat{S}\}$ is unique, then the average Hamming distance between newly generated solutions and the best solution in the set $\widehat{S}$ converges to zero as μ goes to infinity. However, the speed of such convergence is not the same for different components of the solutions generated, i. e., the speed of convergence of the jth component depends on the quality of the solutions S_j^1 compared with the quality of solutions in S_j^0. Simply put, the temperature parameter in (3) controls the level of similarity of the newly generated solutions with high-quality solutions in $\widehat{S}$. The uniqueness of the best solution x^* in $\widehat{S}$ mentioned above should be maintained at all stages of the algorithm.

One of the limitations of the strategy described above is that in order to implement it, there should exist an easy way of generating random solutions from S with the distribution given by (4). Unfortunately, for some problems, the structure of set S would make this hard to achieve. For such cases, the local search based techniques (i. e., SA method, tabu method, GES method) are not easily applicable.

Another issue with generating the random solution x from S using (4) is that the components of the random solution x are not independent random variables. However, for the simplicity of an algorithm, this is usually ignored because the convergence property is more important for the performance of the algorithm.

Whenever the new solution is added to set $\widehat{S}$, it is easy to recalculate the probabilities $\widehat{p}_j$ if the denominator and numerator in (4) are stored separately. Therefore, there is no need to store the whole set $\widehat{S}$ in the

Input: μ – vector of temperature values, K – number of temperature stages, $maxnfail$ – restart parameter, $ngen$ – # of solutions generated during each stage
Output:

$x_{\text{best}} \leftarrow$ construct random solution; $\widehat{S}$=E=$\{x_{\text{best}}\}$
while stopping criterion = FALSE **do**
  if $\widehat{S} = \emptyset$ **then**
    $x \leftarrow$ construct random solution
    $x_{\max} = x$
    $\widehat{S} = \{x_{\max}\}$ (set of known solutions)
    $E = \{x_{\max}\}$ (set of elite solutions)
  end if
  for $nfail = 0$ to $nfail^*$ **do**
    $x_{\text{old}} = x_{\max}$
    for $k = 0$ to K **do**
      calculate generation probabilities($p^k, \widehat{S}, \mu_k$)
      for $g = 0$ to $ngen$ **do**
        $x \leftarrow$ generate solution(x_{max}, p^k)
        $R \leftarrow$ search method(x) (R is some subset of encountered solutions)
        $\widehat{S} = \widehat{S} \cup R$
        $x_{\max} = \arg\min \{f(x) : x \in \widehat{S}\}$
        if $f(x_{\max}) < f(x_{\text{best}})$ **then**
          $x_{\text{best}} = x_{\max}$
        end if
        update_elite_set(E,R)
      end for
    end for
    if $f(x_{\text{old}}) > f(x_{\max})$ **then**
      $nfail = 0$
    end if
    $\widehat{S} = E$
  end for
  $P = P \cup N(x_{\text{best}}, d_p)$
  $E = E - P$
  if RESTART-criterion= TRUE **then**
    $E = \emptyset$
  end if
  $\widehat{S} = E$;
  $x_{\max} = \arg\min \{f(x) : x \in \widehat{S}\}$
end while
return x_{best}

Global Equilibrium Search, Figure 1
Global equilibrium search method (general scheme)

memory! The notion of $\widehat{S}$ is used below mainly for the simplicity of discussion.

The performance of any GES-based algorithm is dependent on the choice of the temperature schedule. As with the SA method, there is no basic recipe to provide an optimal schedule for the GES. The general advice here is to choose the sequence of increasing values $\mu_0 = 0, \mu_1, \mu_2 = \mu_1\alpha, \ldots, \mu_K = \mu_{K-1}\alpha$ (K

is a number of temperature stages and $\alpha > 0$), in such a manner that the algorithm will find the best solution from set $\widehat{S}$ almost for sure when generating solutions with temperature parameter μ_K. However, there is no need to provide a separate cooling schedule for each problem solved. Simple scaling of the cost function ($f'(x) = C \cdot f(x)$, $C > 0$) can make one temperature schedule suitable for a wide range of problems from the same class. The choice of scaling factor can be made, for example, in the initial stage of the algorithm, when $\mu = 0$. Additionally, if we multiply the denominator and numerator of (4) by $\exp(\mu f(x^*))$, where x^* is the best solution from $\widehat{S}$, then the convergence to the best solution from $\widehat{S}$ is less dependent on the absolute values of solution costs.

The general scheme of the GES method is presented in Fig. 1. There are some elements that are included in the scheme, but that were not discussed above: elite solutions set, prohibition of certain solutions and restarting the search. These elements are not necessary for success of the GES method and can be easily excluded. However, for some classes of problems they can provide a significant performance improvement.

The main cycle (lines 2–36) is repeated until some stopping criterion is satisfied. The algorithm execution can be terminated when the best known record for the given problem is improved, or when the running time exceeds some limiting value. If the set of known solutions $\tilde{S}$ is empty, then the initialization of the data set is performed in lines 3–7. The cycle in lines 9–28 is executed until there is no improvement in *nfail** consecutive cycles. The main element of the GES method is the temperature cycle (lines 11–23). The probabilities that guide the search are estimated using expression (4) at the beginning of each temperature stage (line 12). For each probability vector, *ngen* solutions are generated (lines 13–22). These solutions are used as initial solutions for the local search procedure (line 15). The subset of encountered solutions R is used to update set $\widehat{S}$ (line 16).

Some set of the solutions can be stored in memory, in order to provide a fast initialization of the algorithm's memory structures (lines 27 and 34). Such a set is referred to as an elite set in the algorithm pseudocode. Certain solutions can be excluded from this set to avoid searching the same areas multiple times. In lines 29 and 30, the solutions for which the Hamming distance to x_{best} is less than parameter d_p are excluded from the elite set.

A number of successful applications of the GES method have been reported in recent years [6]. The application of the GES method for the multidimensional knapsack problem is described in [8].

The GES based method was presented in [5] for solving job-shop scheduling problems. To date, suitable exact solution methods are not able to find high-quality solutions with reasonable computational effort for the problems involving more than ten jobs and ten machines. The computational testing of the GES algorithm provided a set of new upper bounds for a wide set of challenging benchmark problems [2]. The comparison with existing techniques for job-shop scheduling asserts that the GES method has a great potential for solving scheduling problems.

The application of GES for the unconstrained quadratic programming problem was discussed in [4], where GES was used in a combination with a tabu algorithm. Such an ensemble proved to be an extremely efficient tool for large-scale problems, outperforming some of the best available solution techniques.

In conclusion, the universality of the GES method together with its flexibility make it an optimization tool worth considering.

References

1. Glover F, Laguna M (1993) Tabu search in Modern Heuristic Techniques for Combinatorial Problems. In: Reeves C (ed). Blackwell, Oxford, pp 70–141
2. Job Shop Scheduling webpage, http://plaza.ufl.edu/shylo/jobshopinfo.html. Accessed 14 Oct 2007
3. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by Simulated Annealing. Science 220(4598):671–680
4. Pardalos PM, Prokopyev OA, Shylo OV, Shylo VP (2007) Global equilibrium search applied to the unconstrained binary quadratic optimization problem. Optim Meth Softw. doi:10.1080/10556780701550083
5. Pardalos PM, Shylo OV (2006) An algortihm for the Job Shop Scheduling based on Global Equilibrium Search Techniques. Comput Manag Sci 3(4):331–348
6. Sergienko IV, Shylo VP (2006) Problems of discrete optimization: Challenges and main approaches to solve them. Cybernet Syst Anal 42:465–482
7. Shylo VP (1999) A global equilibrium search method. Kybernetika i Systemnuiy Analys 1:74–80 (in Russian)
8. Shylo VP (2000) Solution of multidimensional knapsack problems by global equilibrium search, Theory of Optimal Solutions. Glushkov VM Inst Cybern, NAS Ukraine Kiev, p 10

Globally Convergent Homotopy Methods

LAYNE T. WATSON
Virginia Polytechnic Institute and State University, Virginia, USA

MSC2000: 65F10, 65F50, 65H10, 65K10

Article Outline

Keywords
Probability-One Globally Convergent Homotopies
Optimization Homotopies
Software
See also
References

Keywords

Continuation; Globally convergent; Homotopy; Nonlinear equations; Probability-one homotopy

Probability-one homotopy methods are a class of algorithms for solving nonlinear systems of equations that are accurate, robust, and converge from an arbitrary starting point almost surely. These new globally convergent homotopy techniques have been successfully applied to solve Brouwer fixed point problems, polynomial systems of equations, constrained and unconstrained optimization problems, discretizations of nonlinear two-point boundary value problems based on shooting, finite differences, collocation, and finite elements, and finite difference, collocation, and Galerkin approximations to nonlinear partial differential equations.

Probability-One Globally Convergent Homotopies

A *homotopy* is a continuous map from the interval [0, 1] into a function space, where the continuity is with respect to the topology of the function space. Intuitively, a homotopy $\rho(\lambda)$ continuously deforms the function $\rho(0) = g$ into the function $\rho(1) = f$ as λ goes from 0 to 1. In this case, f and g are said to be *homotopic*. Homotopy maps are fundamental tools in topology, and provide a powerful mechanism for defining equivalence classes of functions.

Homotopies provide a mathematical formalism for describing an old procedure in numerical analysis, variously known as continuation, incremental loading, and embedding. The continuation procedure for solving a nonlinear system of equations $f(x) = 0$ starts with a (generally simpler) problem $g(x) = 0$ whose solution x_0 is known. The *continuation* procedure is to track the set of zeros of

$$\rho(\lambda, x) = \lambda f(x) + (1 - \lambda)g(x) \tag{1}$$

as λ is increased monotonically from 0 to 1, starting at the known initial point $(0, x_0)$ satisfying $\rho(0, x_0) = 0$. Each step of this tracking process is done by starting at a point $(\widetilde{\lambda}, \widetilde{x})$ on the zero set of ρ, fixing some $\Delta\lambda > 0$, and then solving $\rho(\widetilde{\lambda} + \Delta\lambda, x) = 0$ for x using a locally convergent iterative procedure, which requires an invertible Jacobian matrix $D_x\rho(\widetilde{\lambda} + \Delta\lambda, x)$. The process stops at $\lambda = 1$, since $f(\overline{x}) = \rho(1, \overline{x}) = 0$ gives a zero $\overline{x}$ of $f(x)$. Note that continuation assumes that the zeros of ρ connect the zero x_0 of g to a zero $\overline{x}$ of f, and that the Jacobian matrix $D_x\rho(\lambda, x)$ is invertible along the zero set of ρ; these are strong assumptions, which are frequently not satisfied in practice.

Continuation can fail because the curve γ of zeros of $\rho(\lambda, x)$ emanating from $(0, x_0)$ may:
1) have turning points,
2) bifurcate,
3) fail to exist at some λ values, or
4) wander off to infinity without reaching $\lambda = 1$.

Turning points and bifurcation correspond to singular $D_x\rho(\lambda, x)$. Generalizations of continuation known as *homotopy methods* attempt to deal with cases 1) and 2) and allow tracking of γ to continue through singularities. In particular, continuation monotonically increases λ, whereas homotopy methods permit λ to both increase and decrease along γ. Homotopy methods can also fail via cases 3) or 4).

The map $\rho(\lambda, x)$ connects the functions $g(x)$ and $f(x)$, hence the use of the word 'homotopy'. In general the homotopy map $\rho(\lambda, x)$ need not be a simple convex combination of g and f as in (1), and can involve λ nonlinearly. Sometimes λ is a physical parameter in the original problem $f(x; \lambda) = 0$, where $\lambda = 1$ is the (nondimensionalized) value of interest, although 'artificial parameter' homotopies are generally more computation-

ally efficient than 'natural parameter' homotopies $\rho(\lambda, x) = f(x; \lambda)$. An example of an artificial parameter homotopy map is

$$\rho(\lambda, x) = \lambda f(x;\lambda) + (1-\lambda)(x-a), \tag{2}$$

which satisfies $\rho(0, a) = 0$. The name 'artificial' reflects the fact that solutions to $\rho(\lambda, x) = 0$ have no physical interpretation for $\lambda < 1$. Note that $\rho(\lambda, x)$ in (2) has a unique zero $x = a$ at $\lambda = 0$, regardless of the structure of $f(x; \lambda)$.

All four shortcomings of continuation and homotopy methods have been overcome by probability-one homotopies, proposed in 1976 by S.N. Chow, J. Mallet-Paret, and J.A. Yorke [2]. The supporting theory, based on differential geometry, will be reformulated in less technical jargon here.

Definition 1 Let $U \subset \mathbf{R}^m$ and $V \subset \mathbf{R}^p$ be open sets, and let ρ: $U\times[0, 1)\times V \to \mathbf{R}^p$ be a C^2 map. ρ is said to be *transversal to zero* if the $p\times(m+1+p)$ Jacobian matrix $D\rho$ has full rank on $\rho^{-1}(0)$.

The C^2 requirement is technical, and part of the definition of transversality. The basis for the probability-one homotopy theory is the *parametrized Sard's theorem*, [2]:

Theorem 2 *Let ρ: $U \times [0, 1) \times V \to \mathbf{R}^p$ be a C^2 map. If ρ is transversal to zero, then for almost all $a \in U$ the map*

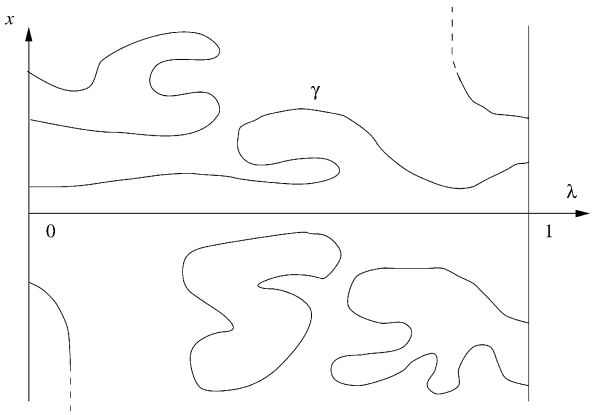
$$\rho_a(\lambda, x) = \rho(a, \lambda, x)$$

is also transversal to zero.

To discuss the importance of this theorem, take $U = \mathbf{R}^m$, $V = \mathbf{R}^p$, and suppose that the C^2 map ρ: $\mathbf{R}^m \times [0, 1) \times \mathbf{R}^p \to \mathbf{R}^p$ is transversal to zero. A straightforward application of the implicit function theorem yields that for almost all $a \in \mathbf{R}^m$, the zero set of ρ_a consists of smooth, nonintersecting curves which either:

1) are closed loops lying entirely in $(0, 1) \times \mathbf{R}^p$,
2) have both endpoints in $\{0\} \times \mathbf{R}^p$,
3) have both endpoints in $\{1\} \times \mathbf{R}^p$,
4) are unbounded with one endpoint in either $\{0\} \times \mathbf{R}^p$ or in $\{1\} \times \mathbf{R}^p$, or
5) have one endpoint in $\{0\} \times \mathbf{R}^p$ and the other in $\{1\} \times \mathbf{R}^p$.

Furthermore, for almost all $a \in \mathbf{R}^m$, the Jacobian matrix $D\rho_a$ has full rank at *every* point in $\rho_a^{-1}(0)$. The goal is to construct a map ρ_a whose zero set has an endpoint in $\{0\} \times \mathbf{R}^p$, and which rules out 2) and 4). Then 5) obtains, and a zero curve starting at $(0, x_0)$ is *guaranteed* to reach a point $(1, \overline{x})$. All of this holds for almost all $a \in \mathbf{R}^m$, and hence with probability one [2]. Furthermore, since $a \in \mathbf{R}^m$ can be almost any point (and, indirectly, so can the starting point x_0), an algorithm based on tracking the zero curve in 5) is legitimately called *globally convergent*. This discussion is summarized in the following theorem (and illustrated in Fig. 1).



Globally Convergent Homotopy Methods, Figure 1
Zero set for $\rho_a(\lambda, x)$ satisfying properties 1)–4)

Theorem 3 *Let f: $\mathbf{R}^p \to \mathbf{R}^p$ be a C^2 map, ρ: $\mathbf{R}^m\times[0, 1)\times \mathbf{R}^p \to \mathbf{R}^p$ a C^2 map, and $\rho_a(\lambda, x) = \rho(a, \lambda, x)$. Suppose that*

1) ρ is transversal to zero.

Suppose also that for each fixed $a \in \mathbf{R}^m$,

2) $\rho_a(0, x) = 0$ has a unique nonsingular solution x_0,

3) $\rho_a(1, x) = f(x)$ $(x \in \mathbf{R}^p)$.

Then, for almost all $a \in \mathbf{R}^m$, there exists a zero curve γ of ρ_a emanating from $(0, x_0)$, along which the Jacobian matrix $D\rho_a$ has full rank.

If, in addition,

4) $\rho_a^{-1}(0)$ is bounded,

then γ reaches a point $(1, \overline{x})$ such that $f(\overline{x}) = 0$. Furthermore, if $Df(\overline{x})$ is invertible, then γ has finite arc length.

Any algorithm for tracking γ from $(0, x_0)$ to $(1, \overline{x})$, based on a homotopy map satisfying the hypotheses of this theorem, is called a *globally convergent probability-one homotopy algorithm*. Of course, the practical numerical details of tracking γ are nontriv-

ial, and have been the subject of twenty years of research in numerical analysis. Production quality software called HOMPACK90 [6] exists for tracking γ. The distinctions between continuation, homotopy methods, and probability-one homotopy methods are subtle but worth noting. Only the latter are provably globally convergent and (by construction) expressly avoid dealing with singularities numerically, unlike continuation and homotopy methods which must explicitly handle singularities numerically.

Assumptions 2) and 3) in Theorem 3 are usually achieved by the construction of ρ (such as (2)), and are straightforward to verify. Although assumption 1) is trivial to verify for some maps, if λ and a are involved nonlinearly in ρ the verification is nontrivial. Assumption 4) is typically very hard to verify, and often is a deep result, since 1)–4) holding implies the *existence* of a solution to $f(x) = 0$.

Note that 1)–4) are sufficient, but not necessary, for the existence of a solution to $f(x) = 0$, which is why homotopy maps not satisfying the hypotheses of the theorem can still be very successful on practical problems. If 1)–3) hold and a solution does *not* exist, then 4) must fail, and nonexistence is manifested by γ going off to infinity. Properties 1)–3) are important because they guarantee good numerical properties along the zero curve γ, which, if bounded, results in a *globally convergent* algorithm. If γ is unbounded, then either the homotopy approach (with this particular ρ) has failed or $f(x) = 0$ has no solution.

A few remarks about the applicability and limitations of probability-one homotopy methods are in order. They are designed to solve a *single* nonlinear system of equations, *not* to track the solutions of a parameterized family of nonlinear systems as that parameter is varied. Thus drastic changes in the solution behavior with respect to that (natural problem) parameter have no effect on the efficacy of the homotopy algorithm, which is solving the problem for a *fixed* value of the natural parameter. In fact, it is precisely for this case of rapidly varying solutions that the probability-one homotopy approach is superior to classical continuation (which would be trying to track the rapidly varying solutions with respect to the problem parameter). Since the homotopy methods described here are not for general solution curve tracking, they are not (directly) applicable to bifurcation problems.

Homotopy methods also require the nonlinear system to be C^2 (twice continuously differentiable), and this limitation cannot be relaxed. However, requiring a finite-dimensional discretization to be smooth does not mean the solution to the infinite-dimensional problem must also be smooth. For example, a Galerkin formulation may produce a smooth nonlinear system in the basis function coefficients even though the basis functions themselves are discontinuous. Homotopy methods for optimization problems may converge to a local minimum or stationary point, and in this regard are no better or worse than other optimization algorithms. In special cases homotopy methods can find all the solutions if there is more than one, but in general the homotopy algorithms are only guaranteed to find one solution.

Optimization Homotopies

A few typical convergence theorems for optimization are given next (see the survey in [5] for more examples and references). Consider first the *unconstrained optimization* problem

$$\min_x f(x). \tag{3}$$

Theorem 4 *Let f: $\boldsymbol{R}^n \to \boldsymbol{R}$ be a C^3 convex map with a minimum at $\widetilde{x}$, $\|\widetilde{x}\|_2 \leq M$. Then for almost all a, $\|a\|_2 < M$, there exists a zero curve γ of the homotopy map*

$$\rho_a(\lambda, x) = \lambda \nabla f(x) + (1-\lambda)(x-a),$$

along which the Jacobian matrix $D\rho_a(\lambda, x)$ has full rank, emanating from (0, a) and reaching a point $(1, \widetilde{x})$, where $\widetilde{x}$ solves (3).

A function is called *uniformly convex* if it is convex and its Hessian's smallest eigenvalue is bounded away from zero. Consider next the constrained optimization problem

$$\min_{x \geq 0} f(x). \tag{4}$$

This is more general than it might appear because the general convex *quadratic program* reduces to a problem of the form (4).

Theorem 5 *Let f : $\boldsymbol{R}^n \to \boldsymbol{R}$ be a C^3 uniformly convex map. Then there exists $\delta > 0$ such that for almost all $a \geq 0$*

with $\|a\|_2 < \delta$ there exists a zero curve γ of the homotopy map

$$\rho_a(\lambda, x) = \lambda K(x) + (1-\lambda)(x-a),$$

where

$$K_i(x) = -\left|\frac{\partial f(x)}{\partial x_i} - x_i\right|^3 + \left(\frac{\partial f(x)}{\partial x_i}\right)^3 + x_i^3,$$

along which the Jacobian matrix $D\rho_a(\lambda, x)$ has full rank, connecting (0, a) to a point $(1, \overline{x})$, where $\overline{x}$ solves the constrained optimization problem (4).

Given $F : \mathbf{R}^n \to \mathbf{R}^n$, the *nonlinear complementarity problem* is to find a vector $x \in \mathbf{R}^n$ such that

$$x \geq 0, \quad F(x) \geq 0, \quad x^\top F(x) = 0. \tag{5}$$

It is interesting that homotopy methods can be adapted to deal with nonlinear inequality constraints and combinatorial conditions as in (5). Define $G : \mathbf{R}^n \to \mathbf{R}^n$ by

$$G_i(z) = -|F_i(z) - z_i|^3 + (F_i(z))^3 + z_i^3,$$
$$i = 1, \ldots, n,$$

and let

$$\rho_a(\lambda, z) = \lambda G(z) + (1-\lambda)(z-a).$$

Theorem 6 *Let $F : \boldsymbol{R}^n \to \boldsymbol{R}^n$ be a C^2 map, and let the Jacobian matrix $DG(z)$ be nonsingular at every zero of $G(z)$. Suppose there exists $r > 0$ such that $z > 0$ and $z_k = \|z\|_\infty \geq r$ imply $F_k(z) > 0$. Then for almost all $a > 0$ there exists a zero curve γ of $\rho_a(\lambda, z)$, along which the Jacobian matrix $D\rho_a(\lambda, z)$ has full rank, having finite arc length and connecting (0, a) to $(1, \overline{z})$, where $\overline{z}$ solves (5).*

Theorem 7 *Let $F : \boldsymbol{R}^n \to \boldsymbol{R}^n$ be a C^2 map, and let the Jacobian matrix $DG(z)$ be nonsingular at every zero of $G(z)$. Suppose there exists $r > 0$ such that $z \geq 0$ and $\|z\|_\infty \geq r$ imply $z_k F_k(z) > 0$ for some index k. Then there exists $\delta > 0$ such that for almost all $a \geq 0$ with $\|a\|_\infty < \delta$ there exists a zero curve γ of $\rho_a(\lambda, z)$, along which the Jacobian matrix $D\rho_a(\lambda, z)$ has full rank, having finite arc length and connecting (0, a) to $(1, \overline{z})$, where $\overline{z}$ solves (5).*

Homotopy algorithms for convex unconstrained optimization are generally not computationally competitive with other approaches. For constrained optimization the homotopy approach offers some advantages, and, especially for the nonlinear complementarity problem, is competitive with and often superior to other algorithms. Consider next the general nonlinear programming problem

$$\begin{cases} \min & \theta(x) \\ \text{s.t.} & g(x) \leq 0, \\ & h(x) = 0, \end{cases} \tag{6}$$

where $x \in \mathbf{R}^n$, θ is real valued, g is an m-dimensional vector, and h is a p-dimensional vector. Assume that θ, g, and h are C^2. The *Kuhn–Tucker necessary optimality conditions* for (6) are (cf. also ▶ Equality-constrained nonlinear programming: KKT necessary optimality conditions):

$$\begin{cases} \nabla\theta(x) + \beta^\top \nabla h(x) + \mu^\top \nabla g(x) = 0, \\ h(x) = 0, \\ g(x) \leq 0, \\ \mu \geq 0, \\ \mu^\top g(x) = 0, \end{cases} \tag{7}$$

where $\beta \in \mathbf{R}^p$ and $\mu \in \mathbf{R}^m$. The complementarity conditions $\mu \geq 0$, $g(x) \leq 0$, $\mu^\intercal g(x) = 0$ are replaced by the equivalent nonlinear system of equations

$$W(x, \mu) = 0, \tag{8}$$

where

$$W_i(x, \mu) = -|\mu_i + g_i(x)|^3 + \mu_i^3 - \left(g_i(x)\right)^3, \quad i = 1, \ldots, m. \tag{9}$$

Thus the optimality conditions (7) take the form

$$F(x, \beta, \mu) = \begin{pmatrix} [\nabla\theta(x) + \beta^\top \nabla h(x) + \mu^\top \nabla g(x)]^\top \\ h(x) \\ W(x, \mu) \end{pmatrix} = 0. \tag{10}$$

With $z = (x, \beta, \mu)$, the proposed homotopy map is

$$\rho_a(\lambda, z) = \lambda F(z) + (1-\lambda)(z-a), \tag{11}$$

where $a \in \mathbf{R}^{n+p+m}$. Simple conditions on θ, g, and h guaranteeing that the above homotopy map $\rho_a(\lambda, z)$ will work are unknown, although this map has worked very well on some difficult realistic engineering problems.

Globally Convergent Homotopy Methods, Table 1
Taxonomy of homotopy subroutines

$x = f(x)$		$F(x) = 0$		$\rho(a, \lambda, x) = 0$		algorithm
dense	sparse	dense	sparse	dense	sparse	
FIXPDF	FIXPDS	FIXPDF	FIXPDS	FIXPDF	FIXPDS	ordinary differential equation
FIXPNF	FIXPNS	FIXPNF	FIXPNS	FIXPNF	FIXPNS	normal flow
FIXPQF	FIXPQS	FIXPQF	FIXPQS	FIXPQF	FIXPQS	augmented Jacobian matrix

Frequently in practice the functions θ, g, and h involve a parameter vector c, and a solution to (6) is known for some $c = c^{(0)}$. Suppose that the problem under consideration has parameter vector $c = c^{(1)}$. Then

$$c = (1 - \lambda)c^{(0)} + \lambda c^{(1)} \tag{12}$$

parametrizes c by λ and $\theta = \theta(x;c) = \theta(x;c(\lambda))$, $g = g(x;c(\lambda))$, $h = h(x;c(\lambda))$. The optimality conditions in (10) become functions of λ as well, $F(\lambda, x, \beta, \mu) = 0$, and

$$\rho_a(\lambda, z) = \lambda F(\lambda, z) + (1 - \lambda)(z - a) \tag{13}$$

is a highly implicit nonlinear function of λ. If $F(0, z^{(0)}) = 0$, a good choice for a in practice has been found to be $a = z^{(0)}$. A natural choice for a homotopy would be simply

$$F(\lambda, z) = 0, \tag{14}$$

since the solution $z^{(0)}$ to $F(0, z) = 0$ (the problem corresponding to $c = c^{(0)}$) is known. However, for various technical reasons, (13) is much better than (14).

Software

There are several software packages implementing both continuous and simplicial homotopy methods; see [1] and [6] for a discussion of some of these packages. A production quality software package written in Fortran 90 is described here. *HOMPACK90* [6] is a Fortran 90 collection of codes for finding zeros or fixed points of nonlinear systems using globally convergent probability-one homotopy algorithms. Three qualitatively different algorithms (ordinary differential equation based, normal flow, quasi-Newton augmented Jacobian matrix) are provided for tracking homotopy zero curves, as well as separate routines for dense and sparse Jacobian matrices. A high level driver for the special case of polynomial systems is also provided. HOMPACK90 features elegant interfaces, use of modules, support for several sparse matrix data structures, and modern iterative algorithms for large sparse Jacobian matrices.

HOMPACK90 is logically organized in two different ways: by algorithm/problem type and by subroutine level. There are three levels of subroutines. The top level consists of drivers, one for each problem type and algorithm type. The second subroutine level implements the major components of the algorithms such as stepping along the homotopy zero curve, computing tangents, and the end game for the solution at $\lambda = 1$. The third subroutine level handles high level numerical linear algebra such as QR factorization, and includes some LAPACK and BLAS routines. The organization of HOMPACK90 by algorithm/problem type is shown in Table 1, which lists the driver name for each algorithm and problem type.

The naming convention is

$$FIXP \begin{Bmatrix} D \\ N \\ Q \end{Bmatrix} \begin{Bmatrix} F \\ S \end{Bmatrix},$$

where $D \approx$ ordinary differential equation algorithm, $N \approx$ normal flow algorithm, $Q \approx$ quasi-Newton augmented Jacobian matrix algorithm, $F \approx$ dense Jacobian matrix, and $S \approx$ sparse Jacobian matrix. Depending on the problem type and the driver chosen, the user must write exactly two subroutines, whose interfaces are specified in the module HOMOTOPY, defining the problem (f or ρ). The module REAL_PRECISION specifies the real numeric model with

SELECTED_REAL_KIND(13),

which will result in 64-bit real arithmetic on a Cray, DEC VAX, and IEEE 754 Standard compliant hardware.

The special purpose polynomial system solver POLSYS1H can find all solutions in complex projective space of a *polynomial system of equations*. Since a polynomial programming problem (where the objective function, inequality constraints, and equality constraints are all in terms of polynomials) can be formulated as a polynomial system of equations, POLSYS1H can effectively find the *global optimum* of a polynomial program. However, polynomial systems can have a huge number of solutions, so this approach is only practical for small polynomial programs (e. g., surface intersection problems that arise in CAD/CAM modeling).

The organization of the Fortran 90 code into modules gives an object oriented flavor to the package. For instance, all of the drivers are encapsulated in a single MODULE HOMPACK90. The user's calling program would then simply contain a statement like

```
USE HOMPACK90, ONLY : FIXPNF
```

Many scientific programmers prefer the reverse call paradigm, whereby a subroutine returns to the calling program whenever the subroutine needs certain information (e. g., a function value) or a certain operation performed (e. g., a matrix-vector multiply). Two reverse call subroutines (STEPNX, ROOTNX) are provided for 'expert' users. STEPNX is an expert reverse call stepping routine for tracking a homotopy zero curve γ that returns to the caller for all linear algebra, all function and derivative values, and can deal gracefully with situations such as the function being undefined at the requested steplength.

ROOTNX provides an expert reverse call end game routine that finds a point on the zero curve where $g(\lambda, x) = 0$, as opposed to just the point where $\lambda = 1$. Thus ROOTNX can find turning points, bifurcation points, and other 'special' points along the zero curve. The combination of STEPNX and ROOTNX provide considerable flexibility for an expert user.

See also

- ► Parametric Optimization: Embeddings, Path Following and Singularities
- ► Topology of Global Optimization

References

1. Allgower EL, Georg K (1990) Numerical continuation methods. Springer, Berlin
2. Chow SN, Mallet-Paret J, Yorke JA (1978) Finding zeros of maps: homotopy methods that are constructive with probability one. Math Comput 32:887–899
3. Forster W (1980) Numerical solution of highly nonlinear problems. North-Holland, Amsterdam
4. Watson LT (1986) Numerical linear algebra aspects of globally convergent homotopy methods. SIAM Rev 28:529–545
5. Watson LT (1990) Globally convergent homotopy algorithms for nonlinear systems of equations. Nonlinear Dynamics 1:143–191
6. Watson LT, Haftka RT (1989) Modern homotopy methods in optimization. Comput Methods Appl Mechanics Engrg 74:289–305
7. Watson LT, Sosonkina M, Melville RC, Morgan AP, Walker HF (1997) Algorithm 777: HOMPACK90: A suite of Fortran 90 codes for globally convergent homotopy algorithms. ACM Trans Math Softw 23:514–549

Global Optimization Algorithms for Financial Planning Problems

Panos Parpas, Berç Rustem
Department of Computing, Imperial College, London, UK

MSC2000: 90B50, 78M50, 91B28

Article Outline

Abstract
Background
Models
  Scenario Generation
  Portfolio Selection
Methods
  A Stochastic Optimization Algorithm
  Other Methods
References

Abstract

It is becoming apparent that convex financial planning models are at times a poor approximation of the real world. More realistic, and more relevant, models need to dispense with normality assumptions and concavity of the utility functions to be optimized. Moreover, the problems are large scale but structured; consequently specialized algorithms have been proposed for their solution. The aim of this article is to discuss a non-

convex portfolio-selection problem and describe algorithms that can be used for its solution.

Background

Modern portfolio theory started in the 1950s with H. Markowitz's work [16,17]. Since then a lot of research has been done in improving the basic models and dispensing with the limiting assumptions of the field. The aim of this article is to introduce the problem of optimization of higher-order moments of a portfolio. This model is an extension of the celebrated mean-variance model of Markowitz [16,17]. The inclusion of higher-order moments has been proposed as one possible augmentation to the model in order to make it more applicable. The applicability of the model can be broadened by relaxing one of its major assumptions, i. e. that the rate of returns are normal. In order to solve the portfolio-selection problem, we first need to address the problem of scenario generation, i. e. the description of the uncertainties used in the portfolio-selection problem. Both problems are non-convex, large-scale, and highly relevant in financial optimization.

We focus on a single-period model where the decision maker (DM) provides as input preferences with respect to mean, variance, skewness and possibly kurtosis of the portfolio. Using these four parameters we then formulate the multicriterion optimization problem as a standard non-linear programming problem. This version of the decision model is a non-convex linearly constrained problem.

Before we can solve the portfolio-selection problem we need to describe the uncertainties regarding the returns of the risky assets. In particular we need to specify: (1) the possible states of the world and (2) the probability of each state. A common approach to this modelling problem is the method of matching moments (see e. g. [5,9,20]). The first step in this approach is to use the historical data to estimate the moments (in this paper we consider the first four central moments, i. e. mean, variance, skewness and kurtosis). The second step is to compute a discrete distribution with the same statistical properties as those calculated in the previous step. Given that our interest is on real-world applications, we recognize that there may not always be a distribution that matches the calculated statistical properties. For this reason we formulate the problem as a least-squares problem [5,9]. The rationale behind this formulation is that we try to calculate a description of the uncertainty that matches our beliefs as well as possible. The scenario-generation problem also has a non-convex objective function and is linearly constrained.

For the two problems described above we apply a new stochastic global optimization algorithm that has been developed specifically for this class of problems. The algorithm is described in [19]. It is an extension of the constrained case of the so-called diffusion algorithm [1,4,6,7]. The method follows the trajectory of an appropriately defined stochastic differential equation (SDE). Feasibility of the trajectory is achieved by projecting its dynamics onto the set defined by the linear equality constraints. A barrier term is used for the purpose of forcing the trajectory to stay within any bound constraints (e. g. positivity of the probabilities, or bounds on how much of each asset to own).

A review of applications of global optimization to portfolio selection problems appeared in [13]. A deterministic global optimization algorithm for a multiperiod model appeared in [15]. This article complements the work mentioned above in the sense that we describe a complete framework for the solution of a realistic financial model. The type of models we consider, due to the large number of variables, cannot be solved by deterministic algorithms. Consequently, practitioners are left with two options: solve a simpler, but less relevant, model or use a heuristic algorithm (e. g. tabu-search or evolutionary algorithms). The approach proposed in this paper lies somewhere in the middle. The proposed algorithm belongs to the simulated-annealing family of algorithms, and it has been shown in [19] that it converges to the global optimum (in a probabilistic sense). Moreover, the computational experience reported in [19] seems to indicate that the method is robust (in terms of finding the global optimum) and reliable. We believe that such an approach will be useful in many practical applications.

Models

Scenario Generation

From its inception stochastic programming (SP) has found several diverse applications as an effective paradigm for modelling decisions under uncertainty. The focus of initial research was on developing effec-

tive algorithms for models of realistic size. An area that has only recently received attention is on methods to represent the uncertainties of the decision problem.

A review of available methods to generate meaningful descriptions of the uncertainties from data can be found in [5]. We will use a least-squares formulation (see e. g. [5,9]). It is motivated by the practical concern that the moments, given as input, may be inconsistent. Consequently, the best one can do is to find a distribution that fits the available data as well as possible. It is further assumed that the distribution is discrete. Under these assumptions the problem can be written as

$$\min_{\omega,p} \sum_{i=1}^{n} \Big(\sum_{j=1}^{k} p_j m_i(\omega_j) - \mu_i \Big)^2$$
$$\text{s.t} \sum_{j=1}^{k} p_j = 1 p_j \geq 0 j = 1, \ldots, k,$$

where μ_i represents the statistical properties of interest and $m_i(\cdot)$ is the associated 'moment' function. For example, if μ_i is the target mean for the ith asset, then $m_i(\omega_j) = \omega_j^i$ i. e. the jth realization of the ith asset. Numerical experiments using this approach for a multistage model were reported in [9] (without arbitrage considerations). Other methods such as maximum entropy [18] and semidefinite programming [2] enjoy strong theoretical properties but cannot be used when the data of the problem are inconsistent. A disadvantage of the least-squares model is that it is highly nonconvex, which makes it very difficult to handle numerically. These considerations lead to the development of the algorithm described in Sect. "A Stochastic Optimization Algorithm" (see also [19]) that can efficiently compute global optima for problems in this class.

When using scenario trees for financial planning problems it becomes necessary to address the issue of arbitrage opportunities [9,12]. An arbitrage opportunity is a self-financing trading strategy that generates a strictly positive cash flow in at least one state and whose payoffs are non-negative in all other states. In other words, it is possible to get something for nothing. In our implementation we eliminate arbitrage opportunities by computing a sufficient set of states so that the resulting scenario tree has the arbitrage-free property. This is achieved by a simple two-step process. In the first step we generate random rates of returns; these are sampled by a uniform distribution. We then test for arbitrage by solving the system

$$x_0^i = e^{-r} \sum_{j=1}^{m} x_j^i \pi_j, \quad \sum_{j=1}^{m} \pi_j = 1, \pi_j \geq 0, \qquad (1)$$
$$j = 1, \ldots, m \quad i = 1, \ldots, n,$$

where x_0^i represents the current (known) state of the world for the ith asset and x_j^i represents the jth realization of the ith asset in the next time period (these are generated by the simulations mentioned above). r is the riskless rate of return. The π_j are called the risk-neutral probabilities. According to a fundamental result of Harisson and Kerps [10], the existence of the risk-neutral probabilities is enough to guarantee that the scenario tree has the desired property. In the second step, we solve the least-squares problem with some of the states fixed to the states calculated in the first step. In other words, we solve the following problem:

$$\min_{\omega,p} \sum_{i=1}^{n} \Big(\sum_{j=1}^{k} p_j m_i(\omega_j) + \sum_{l=1}^{m} p_l m_i(\hat{\omega}_l) - \mu_i \Big)^2 \qquad (2)$$
$$\text{s.t} \sum_{j=1}^{k+m} \quad p_j = 1 p_j \geq 0 \quad j = 1, \ldots, k+m.$$

In the problem above, $\hat{\omega}$ are fixed. Solving the preceding problem guarantees a scenario tree that is arbitrage free.

Portfolio Selection

In this section we describe the portfolio-selection problem when higher-order terms are taken into account. The classical mean–variance approach to portfolio analysis seeks to balance risk (measured by variance) and reward (measured by expected value). There are many ways to specify the single-period problem. We will be using the following basic model:

$$\min_{w} -\alpha \mathbb{E}[w] + \beta \mathbb{V}[w]$$
$$\text{s.t} \sum_{i=1}^{n} w_i = 1 \quad l_i \leq w_i \leq u_i \quad i = 1, \ldots, n, \qquad (3)$$

where $\mathbb{E}[\cdot]$ and $\mathbb{V}[\cdot]$ represent the mean rate of return and its variance respectively. The single constraint is known as the *budget constraint* and it specifies the initial wealth (without loss of generality we have assumed

that this is one). The α and β are positive scalars and are chosen so that $\alpha + \beta = 1$. They specify the DM's preferences, i. e. $\alpha = 1$ means that the DM is risk seeking, while $\beta = 1$ implies that the DM is risk averse. Any other selection of the parameters will produce a point on the efficient frontier. The decision variable (w) represents the commitment of the DM to a particular asset. Note that this problem is a convex quadratic programming problem for which very efficient algorithms exist. The interested reader is referred to the review in [23] for more information regarding the Markowitz model.

We propose an extension of the mean–variance model using higher-order moments. The vector-optimization problem can be formulated as a standard non-convex optimization problem using two additional scalars to act as weights. These weights are used to enforce the DM's preferences. The problem is then formulated as follows:

$$\begin{aligned} &\min_{w} -\alpha\mathbb{E}[w] + \beta\mathbb{V}[w] - \gamma\mathbb{S}[w] + \delta\mathbb{K}[w] \\ &\text{s.t} \sum_{i=1}^{n} w_i = 1 \quad l_i \le w_i \le u_i \quad i = 1, \ldots, n\,, \end{aligned} \tag{4}$$

where $\mathbb{S}[\cdot]$ and $\mathbb{K}[\cdot]$ represent the skewness and kurtosis of the rate of return respectively. γ and δ are positive scalars. The four scalar parameters are chosen so that they sum to one. Positive skewness is desirable (since it corresponds to higher returns, albeit with low probability), while kurtosis is undesirable since it implies that the DM is exposed to more risk. The model in (4) can be extended to multiple periods while maintaining the same structure (non-convex objective and linear constraints). The numerical solution of (2) and (4) will be discussed in the next section.

Methods

A Stochastic Optimization Algorithm

The models described in the previous section can be written as:

$$\begin{aligned} &\min_{x} f(x) \\ &\text{s.t } Ax = b \\ &\quad x \ge 0\,. \end{aligned}$$

A well-known method for obtaining a solution to an unconstrained optimization problem is to consider the following ordinary differential equation (ODE):

$$\mathrm{d}X(t) = -\nabla f(X(t))\,\mathrm{d}t\,. \tag{5}$$

By studying the behaviour of $X(t)$ for large t, it can be shown that $X(t)$ will eventually converge to a stationary point of the unconstrained problem. A review of so-called continuous-path methods can be found in [25]. A deficiency of using (5) to solve optimization problems is that it will get trapped in local minima. To allow the trajectory to escape from local minima, it has been proposed by various authors (e. g. [1,4,6,7]) to add a stochastic term that would allow the trajectory to 'climb' hills. One possible augmentation to (5) that would enable us to escape from local minima is to add noise. One then considers the *diffusion process*:

$$\mathrm{d}X(t) = -\nabla f(X(t))\,\mathrm{d}t + \sqrt{2T(t)}\,\mathrm{d}B(t)\,, \tag{6}$$

where $B(t)$ is the standard Brownian motion in $\mathbb{R}^n$. It has been shown in [4,6,7], under appropriate conditions on f and $T(t)$, that as $t \to \infty$, the transition probability of $X(t)$ converges to a probability measure Π. The latter has its support on the set of global minimizers.

For the sake of argument, suppose we did not have any linear constraints but only positivity constraints. We could then consider enforcing the feasibility of the iterates by using a barrier function. According to the algorithmic framework sketched out above, we could obtain a solution to our (simplified) problem by following the trajectory of the following SDE:

$$\mathrm{d}X(t) = -\nabla f(X(t))\,\mathrm{d}t + \mu X(t)^{-1}\,\mathrm{d}t + \sqrt{2T(t)}\,\mathrm{d}B(t)\,, \tag{7}$$

where $\mu > 0$ is the barrier parameter. By X^{-1} we will denote an n-dimensional vector whose ith component is given by $1/X_i$. Having used a barrier function to deal with the positivity constraints, we can now introduce the linear constraints into our SDE@. This process has been carried out in [19] using the projected SDE:

$$\mathrm{d}X(t) = P[-\nabla f(X(t)) + \mu X(t)^{-1}]\,\mathrm{d}t + \sqrt{2T(t)}P\,\mathrm{d}B(t)\,, \tag{8}$$

where $P = I - A^T(AA^T)^{-1}A$. The proposed algorithm works in a similar manner to gradient-projection algorithms. The key difference is the addition of a barrier parameter for the positivity of the iterates and

a stochastic term that helps the algorithm escape from local minima.

The global optimization problem can be solved by fixing μ and following the trajectory of (8) for a suitably defined function $T(t)$. After sufficient time passes, we reduce μ and repeat the process. The proof that following the trajectory of (8) will eventually lead us to the global minimum appears in [19]. Note that the projection matrix for the type of constraints we need to impose for our models is particularly simple. For a constraint of the type $\sum_{i=1}^{n} x_i = 1$ the projection matrix is given by

$$P_{ij} = \begin{cases} -\frac{1}{n} & \text{if } i \neq j, \\ \frac{n-1}{n} & \text{otherwise.} \end{cases}$$

Other Methods

In this article we have focused on the numerical solution of a financial planning problem using a stochastic algorithm. We end this article by briefly discussing other possible approaches. Only stochastic methods will be discussed; for deterministic methods we refer the interested reader to [13].

Two-phase methods: Methods belonging to this class, as the name suggests, have two phases: a local and global phase. In the global phase, the feasible region is uniformly sampled. From each feasible point a local optimization algorithm is started. The later process is the local phase. This basic algorithmic framework has been modified to improve its performance by various authors. Improving this type of method requires careful selection of the sample points from which to start the local optimizations. Inevitably there is some compromise between computational efficiency and theoretical convergence. For a review of two-phase methods we refer the reader to [21] and references therein.

Simulated annealing (SA): This family of algorithms was inspired by the physical behaviour of atoms in a liquid. The method was independently proposed by Cerny[3] and Kirkpatrick et al. [11]. The method is inspired by a fundamental question of statistical mechanics concerning the behaviour of the system in low temperatures. For example, will the atoms remain fluid or will they solidify? If they solidify, do they form a crystalline solid or a glass? It turns out [11] that if the temperature is decreased slowly, then they form a pure crystal; this state corresponds to the minimum energy of the system. If the temperature is decreased too quickly, then they form a crystal with many defects. SA algorithms generate a point from some distribution. Whether to accept the new point or not is decided by an acceptance function. The latter function is 'temperature' dependent. At high temperatures the function is likely to accept the new point, while at low temperatures only points close to the global optimum value are supposed to be accepted. As can be anticipated, the performance of the algorithm depends on the annealing schedule, i. e. how fast the temperature is reduced. Performance also depends on how points are sampled, the acceptance function and, of course, the stopping conditions. An excellent review article for SA is [14].

Stochastic adaptive search methods: These types of algorithms have strong theoretical properties but present challenging implementation issues. A typical algorithm from this class is the pure adaptive search method. This method works like a pure random search method but with the additional assumption of the ability to sample from a distribution that gives realizations that are strictly better than the incumbent. There exist many variants and combinations of this type of method, and an excellent review of them is given in [24].

Genetic algorithms: This class of algorithms has been inspired by concepts from evolutionary biology and from aspects of natural selection. There are two phases in these algorithms: generation of the population and updating. During the generation phase, candidate points (offsprings) are generated by sampling a p.d.f. This p.d.f. is usually specified from the original or the previous generation (the parents). In the second phase the population is updated. This update is performed by applying a selection mechanism and performing mutation operations on the population. There are very few theoretical results concerning the convergence properties of genetic algorithms. However, if their success in applications is anything to go by, then more attention needs to devoted to convergence aspects of the method. An excellent review of genetic algorithms is given in [22].

Tabu search: This is another heuristic algorithm that has been successfully used for global optimization (especially combinatorial problems) but lacks theoretical backing. This class of algorithms was proposed by Glover, and a review of the method appeared in [8]. The

algorithm has three phases: preliminary search, intensification, and diversification. In the first phase, the algorithm takes the current configuration, examines neighbouring solutions, and selects the one with the best objective function value. This process is continued until no improving state can be identified. At this stage the possibility of returning to this point is ruled out by placing it into a list. This list is called the tabu list. In the second phase (intensification), the tabu list is cleared and the algorithm returns to the first phase. In the final stage (diversification), the most frequent moves that were placed into the tabu list during the first phase are placed from the start into the list. The algorithm then starts from a random initial point. In this phase the algorithm is not allowed to make any moves that are in the tabu list.

References

1. Aluffi-Pentini F, Parisi V, Zirilli F (1985) Global optimization and stochastic differential equations. J Optim Theory Appl 47(1):1–16
2. Bertsimas D, Sethuraman J (2000) Moment problems and semidefinite optimization. In: Handbook of semidefinite programming, vol 27. Int Ser Oper Res Manage Sci. Kluwer, Boston, pp 469–509
3. Černý V (1985) Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. J Optim Theory Appl 45(1):41–51
4. Chiang TS, Hwang CR, Sheu SJ (1987) Diffusion for global optimization in R^n. SIAM J Control Optim 25(3):737–753
5. Dupacova J, Consigli G, Wallace SW (2000) Scenarios for multistage stochastic programs. Ann Oper Res 100:25–53 (2001)
6. Geman S, Hwang CR (1986) Diffusions for global optimization. SIAM J Control Optim 24(5):1031–1043
7. Gidas B (1986) The Langevin equation as a global minimization algorithm. In: Disordered systems and biological organization (Les Houches, 1985), vol 20. NATO Adv Sci Inst Ser F Comput Syst Sci. Springer, Berlin, pp 321–326
8. Glover F, Laguna M (1998) Tabu search. In: Handbook of combinatorial optimization, vol. 3. Kluwer, Boston, pp 621–757
9. Gülpınar N, Rustem B, Settergren R (2004) Simulation and optimization approaches to scenario tree generation. J Econ Dyn Control 28(7):1291–1315
10. Harrison JM, Kreps DM (1979) Martingales and arbitrage in multiperiod securities markets. J Econom Theory 20:381–408
11. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
12. Klaassen P (1997) Discretized reality and spurious profits in stochastic programming models for asset/liability management. Eur J Oper Res 101(2):374–392
13. Konno H (2005) Applications of global optimization to portfolio analysis. In: Audet C, Hansen P, Savard G (eds) Essays and Surveys in Global Optimization. Springer, Berlin, pp 195–210
14. Locatelli M (2002) Simulated annealing algorithms for continuous global optimization. In: Handbook of global optimization, vol 2, vol 62. Nonconvex Optim Appl. Kluwer, Dordrecht, pp 179–229
15. Maranas CD, Androulakis IP, Floudas CA, Berger AJ, Mulvey JM (1997) Solving long-term financial planning problems via global optimization. J Econ. Dynam Control 21(8–9):1405–1425 Computational financial modeling.
16. Markowitz HM (1952) Portfolio selection. J Finance 7:77–91
17. Markowitz HM (1952) The utility of wealth. J Polit Econ (60):151–158
18. Parpas P (2006) Algorithms in Stochastic Optimization. PhD Thesis, Imperial College London, May 2006
19. Parpas P, Rustem B, Pistikopoulos EN (2006) Linearly constrained global optimization and stochastic differential equations. J Global Optim 36(2):191–217
20. Prékopa A (1995) Stochastic programming, vol 324. Math Appl. Kluwer, Dordrecht
21. Schoen F (2002) Two-phase methods for global optimization. In: Handbook of global optimization, vol 2, vol 62. Nonconvex Optim Appl. Kluwer, Dordrecht, pp 151–177
22. Smith JE (2002) Genetic algorithms. In: Handbook of global optimization, vol 2, vol 62. Nonconvex Optim Appl. Kluwer, Dordrecht, pp 275–362
23. Steinbach MC (2001) Markowitz revisited: mean-variance models in financial portfolio analysis. SIAM Rev 43(1):31–85
24. Wood GR, Zabinsky ZB (2002) Stochastic adaptive search. In: Handbook of global optimization, vol 2, vol 62. Nonconvex Optim Appl. Kluwer, Dordrecht, pp 231–249
25. Zirilli F (1982) The use of ordinary differential equations in the solution of nonlinear systems of equations. In: Nonlinear optimization, 1981 (Cambridge, 1981), NATO Conf Ser II: Syst Sci. Academic, London, pp 39–46

Global Optimization in the Analysis and Management of Environmental Systems

JÁNOS D. PINTÉR
Pintér Consulting Services, Inc.,
and Dalhousie University, Halifax, Canada

MSC2000: 90C05

Article Outline

Keywords
Environmental Systems Analysis and Optimization
Model Calibration
'Black Box' Optimization (in Environmental Systems)
See also
References

Keywords

Nonlinear decision models; Multi-extremality; Continuous global optimization; Applications in environmental systems modeling and management

Environmental Systems Analysis and Optimization

The harmonized consideration of technical, economic and environmental objectives in strategic planning and operational decision making is of paramount importance, on a worldwide scale. Environmental quality issues are of serious concern even in the most developed countries, although direct pollution control expenditures are typically in the 2–3 percent range of their gross domestic product. The 'optimized' or at least 'acceptable' solution of environmental quality problems requires the combination of knowledge from a multitude of areas, and requires an interdisciplinary effort.

In the past decades, mathematical programming (MP) models have been applied also to the analysis and management of environmental systems. The annotated bibliography [9] reviews over 350 works, including some thirty books. Note further that the engineering, economic and environmental science literature contains a very large amount of work that can serve as a basis and therefore is closely related to such modeling efforts. For instance, the classic textbook [28] reviews the basic quantitative models applied in describing physical, chemical and biological phenomena of relevance. A more recent exposition (with a somewhat broader scope) is presented in, for instance, [11]. The chapters in the latter edited volume discuss the following issues:

- environmental crisis, as a multidisciplinary challenge;
- soil pollution;
- air pollution;
- water pollution;
- water resources management;
- pesticides;
- gene technology;
- landscape planning;
- environmental economics;
- ecological aspects;
- environmental impact assessment;
- environmental management models.

Environmental management models are discussed – in the broader context of governmental planning and operations – already in [8]. In addition to items listed above, the (relevant) topics covered include also

- solid waste management;
- urban development;
- policy analysis.

Numerous further books can be mentioned; with varying emphasis on environmental science, engineering, economics or systems analysis. Consult, e. g., [1,2,3,4,6,10,13,15,16,17,18,19,23,24,25,29,31,32,33]. Most of these works also provide extensive lists of additional references.

In the framework of this short article there is no room to go into any detailed discussion of environmental models. Therefore we shall only emphasize one important methodological aspect reflected by the title: namely, the relevance of global optimization in this context.

The predominant majority of MP models presented, e. g., in the books listed or in [9] belong to (continuous or possibly mixed integer) linear programming, or to convex nonlinear programming, with additional – usually rather simplified – considerations regarding system stochasticity. At the same time, more detailed or more realistic models of natural systems and their governing processes often possess high (explicit or hidden) high nonlinearity. For instance, one may think of power laws, periodic or chaotic processes, and (semi)random fluctuations, reflected by many natural objects on various scales: mountains, waters, plants, animals, and so on. For related far-reaching discussions, consult, for example, [5,7,20,21], or [30]. Since many natural objects and processes are inherently nonlinear, management models that optimize the behavior of environmental systems frequently lead to multi-extremal deci-

sion problems. Continuous global optimization (GO) is aimed at finding the 'absolutely best' solution of such models, in the possible presence of many other (locally optimal) solutions of various quality. See ► Continuous global optimization: Models, algorithms and software and ► Continuous global optimization: Applications for a number of textbooks and WWW sites related to the subject of GO. Therefore, here we mention only the handbook [14] and the WWW site [22].

We shall illustrate the relevance of GO by two very general examples, adapted from [26]. The latter book presents also a number of other case studies related to environmental modeling and management, with numerous additional references pertinent to this subject.

Model Calibration

The incomplete or poor understanding of environmental – as well as many other complex – systems calls for descriptive model development as an essential tool of the related research. The following main phases of quantitative systems modeling can be distinguished:

- identification: formulation of principal modeling objectives, determination (selection) of suitable model structure;
- calibration: (inverse) model fitting to available data and background information;
- validation and application in analysis, forecasting, control, management.

Consequently, the 'adequate' or 'best' parameterization of descriptive models is an important stage in the process of understanding environmental systems. Interesting, practically motivated discussions of the model calibration problem are presented also in [1,3,12,32].

A fairly simple and commonly applied instance of the model calibration problem can be stated as follows. Given

- a descriptive system model (e. g. of a lake, river, groundwater or atmospheric system) that depends on certain unknown (physical, chemical) parameters; their vector is denoted by x;
- the set of a priori feasible parameterizations D;
- the model output values $y_t^{(m)} = y_t^{(m)}(x)$ at time moments $t = 1, \ldots, T$;
- a set of corresponding observations y_t at $t = 1, \ldots, T$;
- a discrepancy measure denoted by f which expresses the distance between $y_t^{(m)}$ and y_t.

Then the optimized model calibration problem can be formulated as

$$\begin{cases} \min & f(x) := f\{y_t^{(m)}(x), y_t\} \\ \text{s.t.} & x \in D. \end{cases} \tag{1}$$

Frequently, D is a finite n-interval (a 'box'); furthermore, f is a continuous or somewhat more special (smooth, Lipschitz, etc.) function. Additional structural assumptions regarding f may be difficult to postulate, due to the following reason. For each fixed parameter vector x, the model output sequence $\{y_t^{(m)}(x)\}$ may be produced by some implicit formulas, or by a computationally demanding numerical procedure (such as e. g., the solution of a system of partial differential equations). Consequently, although model (1) most typically belongs to the general class of continuous GO problems, a more specific classification may be difficult to provide. Therefore one needs to apply a GO procedure that enables the solution of the calibration problem under the very general conditions outlined above.

To conclude the brief discussion of this example, note that in [26] several variants of the calibration problem statement are studied in detail. Namely, the model development and solver system LGO is applied to solve model calibration problems related to water quality analysis in rivers and lakes, river flow hydraulics, and aquifer modeling. (More recent implementations of LGO are described elsewhere: consult, e. g., [27].)

'Black Box' Optimization (in Environmental Systems)

As outlined above, the more realistic – as opposed to strongly simplified – analysis of environmental processes frequently requires the development of sophisticated systems of (sub)models: these are then connected to a suitable optimization modeling framework. For examples of various complexity, consult [1,2,10,19,32]. We shall illustrate this point by briefly discussing a modeling framework for river water quality management: for additional details, see [26] and references therein.

Assume that the ambient water quality in a river at time t is characterized by a certain vector $s(t)$. The components in $s(t)$ can include, for instance the following: suspended solids concentration, dissolved oxygen concentration, biological oxygen demand, chemical oxy-

gen demand, concentrations of micro-pollutants and heavy metals, and so on. Naturally, the resulting water quality is influenced by a number of factors. These include the often stochastically fluctuating (discharge or nonpoint source) pollution load, as well as the regional hydro-meteorological conditions (streamflow rate, water temperature, etc). Some of these factors can be directly observed, while some others may not be completely known. In a typical model development process, submodels are constructed to describe all physical, chemical, biological, and ecological processes of relevance. (As for an example, one can refer to the classical Streeter–Phelps differential equations that approximate the longitudinal evolution of biological oxygen demand in a river; consult [25,28].)

In order to combine such system description with management models, one has to be able to evaluate all decision considered. Each given decision x can be related, inter alia, to the location and sizing of industrial and municipal wastewater treatment plants, the control of nonpoint source (agricultural) pollution, the design of a wastewater sewage collection network, the daily operation of these facilities, and so on. The analysis frequently involves the computationally intensive evaluation of environmental quality – e. g., by solving a system of (partial) differential equations – for each decision option considered. The quite (possibly) more realistic stochastic extensions of such models may also require the execution of Monte-Carlo simulation cycles. Under such or similar circumstances, environmental management models can be (very) complex consisting of a number of 'black box' submodels. Consequently, the following general conceptual modeling framework may, and often will, lead to multi-extremal model instances requiring the application of suitable GO techniques:

$$\begin{aligned} &\min\{\mathrm{TCEM}(x)\}, \\ &\mathrm{EQ}_{\min} \le \mathrm{EQ}(x) \le \mathrm{EQ}_{\max}, \\ &\mathrm{TF}_{\min} \le \mathrm{TF}(x) \le \mathrm{TF}_{\max}, \end{aligned} \tag{2}$$

in which

- $\mathrm{TCEM}(x)$ is total (discounted, expected) costs of environmental management;
- $\mathrm{EQ}(x)$ is resulting environmental quality (vector);
- $\mathrm{EQ}_{\min}$ and $\mathrm{EQ}_{\max}$ are vector bounds on 'acceptable' environmental quality indicators;
- $\mathrm{TF}(x)$ are resulting technical system characteristics (vector);
- $\mathrm{TF}_{\min}$ and $\mathrm{TF}_{\max}$ are vector bounds on 'acceptable' technical characteristics.

Numerous other examples could be cited: similarly to the case considered above, they may involve the solution of systems of (algebraic, ordinary or partial differential) equations, and/or the statistical analysis of the environmental (model) system studied. For further examples – including data analysis, combination of expert opinions, environmental model calibration, industrial wastewater management, regional pollution management in rivers and lakes, risk assessment and control of accidental pollution – in the context of global optimization consult, e. g., [26], and references therein.

See also

- ▶ Continuous Global Optimization: Applications
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Interval Global Optimization
- ▶ Mixed Integer Nonlinear Programming
- ▶ Optimization in Water Resources

References

1. Beck MB (1985) Water quality management: A review of the development and application of mathematical models. Springer, Berlin
2. Beck MB (ed) (1987) Systems analysis in water quality management. Pergamon, Oxford
3. Beck MB, van Straten G (eds) (1983) Uncertainty and forecasting in water quality. Springer, Berlin
4. Bower BT (ed) (1977) Regional residuals environmental quality management. Johns Hopkins Univ. Press, Baltimore, MD
5. Casti JL (1990) Searching for certainty. Morrow, New York
6. Dorfman R, Jacoby HD, Thomas HA (eds) (1974) Models for managing regional water quality. Harvard Univ. Press, Cambridge, MA
7. Eigen M, Winkler R (1975) Das Spiel. Piper, Munich
8. Gass SI, Sisson RI (eds) (1974) A guide to models in governmental planning and operations. Environmental Protection Agency, Washington, DC
9. Greenberg HJ (1995) Mathematical programming models for environmental quality control. Oper Res 43:578–622
10. Haith DA (1982) Environmental systems optimization. Wiley, New York

11. Hansen PE, Jørgensen SE (eds) (1991) Introduction to environmental management. Elsevier, Amsterdam
12. Hendrix EMT (1998) Global optimization at work. PhD Thesis, LU Wageningen
13. Holling CS (ed) (1978) Adaptive environmental assessment and management. IIASA & Wiley, New York
14. Horst R, Pardalos PM (eds) (1995) Handbook of global optimization. Kluwer, Dordrecht
15. Jørgensen SE (ed) (1983) Applications of ecological modelling in environmental management. Elsevier, Amsterdam
16. Kleindorfer PR, Kunreuther HC (eds) (1987) Insuring and managing hazardous risks: From Seveso to Bhopal. Springer, Berlin
17. Kneese AV, Ayres RU, d'Arge RC (1970) Economics and the environment: A materials balance approach. Johns Hopkins Univ. Press, Baltimore, MD
18. Kneese AV, Bower BT (1968) Managing water quality: Economics, technology, institutions. Johns Hopkins Univ. Press, Baltimore, MD
19. Loucks DP, Stedinger JR, Haith DA (1981) Water resources systems planning and analysis. Prentice-Hall, Englewood Cliffs, NJ
20. Mandelbrot BB (1983) The fractal geometry of nature. Freeman, New York
21. Murray JD (1983) Mathematical biology. Springer, Berlin
22. Neumaier A (1999) Global optimization. http://solon.cma.univie.ac.at/~neum/glopt.html
23. Nijkamp P (1980) Environmental policy analysis: Operational methods and models. Wiley, New York
24. Novotny W, Chesters G (1982) Handbook of nonpoint pollution. v. Nostrand, Princeton, NJ
25. Orlob GT (1983) Mathematical modeling of water quality: Streams, lakes and reservoirs. Wiley, New York
26. Pintér JD (1996) Global optimization in action. Kluwer, Dordrecht
27. Pintér JD (1998) A model development system for global optimization. In: De Leone R, Murli A, Pardalos PM, Toraldo G (eds) High Performance Software for Nonlinear Optimization: Status and Perspectives. Kluwer, Dordrecht, pp 301–314
28. Rich LG (1972) Environmental systems engineering. McGraw-Hill, New York
29. Richardson ML (ed) (1988) Risk assessment of chemicals in the environment. The Royal Soc. Chemistry London, London
30. Schroeder M (1991) Fractals, chaos, power laws. Freeman, New York
31. Seneca JJ, Taussig MK (1974) Environmental economics. Prentice-Hall, Englewood Cliffs, NJ
32. Somlyódy L, van Straten G (eds) (1983) Modeling and managing shallow lake eutrophication. Springer, Berlin
33. United States Environmental Protection Agency (1988) Waste minimization opportunity assessment manual. Techn. Report EPA Cincinnati

Global Optimization: Application to Phase Equilibrium Problems

MARK A. STADTHERR
Department Chemical Engineering,
University Notre Dame, Notre Dame, USA

MSC2000: 80A10, 80A22, 90C90, 65H20

Article Outline

Keywords
Background
Phase Stability Analysis
Interval Analysis
Conclusion
See also
References

Keywords

Interval analysis; Global optimization; Phase equilibrium; Phase stability; Interval Newton

The reliable calculation of phase equilibrium for multicomponent mixtures is a critical aspect in the simulation, optimization and design of a wide variety of industrial processes, especially those involving separation operations such as distillation and extraction. It is also important in the simulation of enhanced oil recovery processes such as miscible or immiscible gas flooding. Unfortunately, however, even when accurate models of the necessary thermodynamic properties are available, it is often very difficult to actually solve the phase equilibrium problem reliably.

Background

The computation of phase equilibrium is often considered in two stages, as outlined by M.L. Michelsen [12, 13]. The first involves the *phase stability* problem, that is, to determine whether or not a given mixture will split into multiple phases. The second involves the *phase split* problem, that is to determine the amounts and compositions of the phases assumed to be present. After a phase split problem is solved it may be necessary to do phase stability analysis on the results to determine whether the postulated number of phases was cor-

rect, and if not repeat the phase split problem. Both the phase stability and phase split problems can be formulated as minimization problems, or as equivalent nonlinear equation solving problems.

For determining phase equilibrium at constant temperature and pressure, the most commonly considered case, a model of the Gibbs free energy of the system is required. This is usually based on an excess Gibbs energy model (activity coefficient model) or an equation of state model. At equilibrium the total Gibbs energy of the system is minimized. Phase stability analysis may be interpreted as a global optimality test that determines whether the phase being tested corresponds to a global optimum in the total Gibbs energy of the system. If it is determined that a phase will split, then a phase split problem is solved, which can be interpreted as finding a *local* minimum in the total Gibbs energy of the system. This local minimum can then be tested for global optimality using phase stability analysis. If necessary the phase split calculation must then be repeated, perhaps changing the number of phases assumed to be present, until a solution is found that meets the global optimality test. Clearly the correct solution of the phase stability problem, itself a global optimization problem, is the key in this two-stage global optimization procedure for phase equilibrium. As emphasized in [10], while it is possible to apply rigorous global optimization techniques directly to the phase equilibrium problem, it is computationally more efficient to use a two-stage approach such as outlined above, since the dimensionality of the global optimization problem that must be solved (phase stability problem) is less than that of the full phase equilibrium problem.

In solving the phase stability problem, the conventional solution methods are initialization dependent, and may fail by converging to trivial or nonphysical solutions or to a point that is a local but not a global minimum. Thus there is no guarantee that the phase equilibrium problem has been correctly solved. Because of the difficulties that may arise in solving phase equilibrium problems by standard methods (e. g., [12,13]), there has been significant interest in the development of more reliable methods. For example, the methods of A.C. Sun and W.D. Seider [16], who use a homotopy continuation approach, and of S.K. Wasylkiewicz, L.N. Sridhar, M.F. Malone and M.F. Doherty [18], who use an approach based on topological considerations, can offer significant improvements in reliability. C.M. McDonald and C.A. Floudas [7,8,9,10] show that, for certain activity coefficient models, the phase stability and equilibrium problems can be made amenable to solution by powerful global optimization techniques, which provide a mathematical guarantee of reliability.

An alternative approach for solving the phase stability problem, based on interval analysis, that provides both mathematical and computational guarantees of global optimality, was originally suggested by M.A. Stadtherr, C.A. Schnepper and J.F. Brennecke [15], who applied it in connection with activity coefficient models, as later done also in [11]. This technique, in particular the use of an *interval Newton* and *generalized bisection* algorithm, is initialization independent and can solve the phase stability problem with mathematical certainty, and, since it deals automatically with rounding error, with computational certainty as well. J.Z. Hua, Brennecke and Stadtherr [3,4,5,6] extended this method to problems modeled with cubic *equation of state* models, in particular the Van der Waals, Peng–Robinson, and Soave–Redlich–Kwong models. Though interval analysis provides a *general purpose* and *model independent* approach for guaranteed solution of the phase stability problem, the discussion below will focus on the use of cubic equation of state models.

Phase Stability Analysis

The determination of phase stability is often done using tangent plane analysis [1,12]. A phase at specified temperature T, pressure P, and feed mole fraction vector $\mathbf{z}$ is unstable and can split (in this context, 'unstable' refers to both the thermodynamically metastable and classically unstable cases), if the molar Gibbs energy of mixing surface $m(\mathbf{x}, v)$ ever falls below a plane tangent to the surface at $\mathbf{z}$. That is, if the tangent plane distance

$$D(\mathbf{x}, v) = m(\mathbf{x}, v) - m_0 - \sum_{i=1}^{n} \left(\frac{\partial m}{\partial x_i} \right)_0 (x_i - z_i)$$

is negative for any composition (mole fraction) vector $\mathbf{x}$, the phase is unstable. The subscript zero indicates evaluation at $\mathbf{x} = \mathbf{z}$, n is the number of components, and v is the molar volume of the mixture. A common approach for determining if D is ever negative is to min-

imize D subject to the mole fractions summing to one

$$1 - \sum_{i=1}^{n} x_i = 0 \tag{1}$$

and subject to the equation of state relating $\mathbf{x}$ and v:

$$P - \frac{RT}{v-b} + \frac{a}{v^2 + ubv + wb^2} = 0. \tag{2}$$

Here a and b are functions of $\mathbf{x}$ determined by specified mixing rules. The 'standard' mixing rules are $b = \sum_{i=1}^{n} x_i b_i$ and $a = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j a_{ij}$, with $a_{ij} = (1 - k_{ij})\sqrt{a_i a_j}$. The $a_i(T)$ and b_i are pure component properties determined from the system temperature T, the critical temperatures T_{ci}, the critical pressures P_{ci} and acentric factors ω_i. The binary interaction parameter k_{ij} is generally determined experimentally by fitting binary vapor-liquid equilibrium data. Equation (2) is a generalized cubic equation of state model. With the appropriate choice of u and w, common models such as Peng–Robinson ($u = 2$, $w = -1$), Soave–Redlich–Kwong ($u = 1$, $w = 0$), and Van der Waals ($u = 0$, $w = 0$) may be obtained. It is readily shown that the stationary points in this optimization problem must satisfy

$$s_i(\mathbf{x}, v) - s_i(\mathbf{z}, v_0) = 0, \quad i = 1, \ldots, n-1, \tag{3}$$

where

$$s_i = \left(\frac{\partial m}{\partial x_i}\right) - \left(\frac{\partial m}{\partial x_n}\right).$$

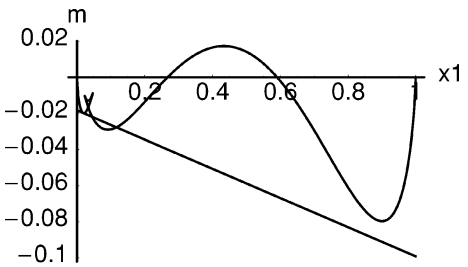
The $(n + 1) \times (n + 1)$ system given by equations (1), (2) and (3) above can be used to solve for the stationary points in the optimization problem.

The equation system for the stationary points has a trivial root at $(\mathbf{x}, v) = (\mathbf{z}, v_0)$ and frequently has multiple nontrivial roots as well. Thus conventional equation solving techniques may fail by converging to the trivial root or give an incorrect answer to the phase stability problem by converging to a stationary point that is not the global minimum of D. This is aptly demonstrated by the experiments of K.A. Green, S. Zhou and K.D. Luks [2], who show that the pattern of convergence from different initial guesses demonstrates a complex fractal-like behavior for even very simple models like Van der Waals. The problem is further complicated by the fact that the cubic equation of state (2) may have multiple real volume roots v.

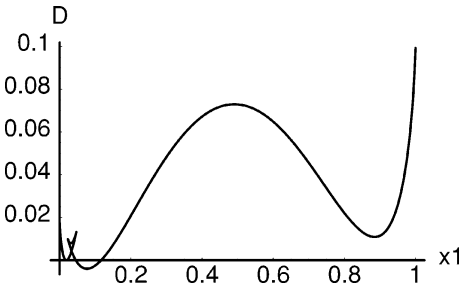
As an example of a system that causes numerical difficulties, consider the binary mixture of hydrogen sulfide (component 1) and methane (component 2) at a temperature of 190 K and pressure of 40.53 bar (40 atm) modeled using the Soave–Redlich–Kwong equation of state, and with an overall feed composition of $z_1 = 0.0187$. Figure 1 shows a plot of the reduced Gibbs energy of mixing m vs. x_1 for this system (in the reduced composition space where $x_2 = 1 - x_1$), and also shows the tangent at the feed composition.

The corresponding tangent plane distance function is shown in Fig. 2 and Fig. 3.

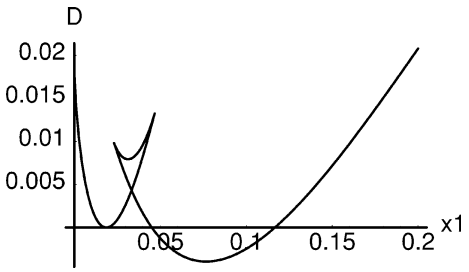
Note that this system has a region, around x_1 of 0.03 to 0.05, where multiple real volume roots occur and thus multiple values of m and D exist; only the lowest values are physically significant. This system has five stationary points, four minima and one maximum. Conventional locally convergent methods are typically used with multiple initial guesses, generally at or near



Global Optimization: Application to Phase Equilibrium Problems, Figure 1
Reduced Gibbs energy of mixing m versus x_1 for the system hydrogen sulfide and methane, showing tangent at a feed composition of 0.0187



Global Optimization: Application to Phase Equilibrium Problems, Figure 2
Tangent plane distance D versus x_1 for the example system of Fig. 1. See Fig. 3 for enlargement of area near the origin



Global Optimization: Application to Phase Equilibrium Problems, Figure 3
Enlargement of part of Fig. 2, showing area near the origin

the pure components ($x_1 = 0$ and $x_1 = 1$). When this is done convergence will likely be to the local minimum at the feed composition (0.0187) and to the local minimum around 0.88. The global minimum with $D < 0$ is missed, leading to the incorrect conclusion that the mixture is stable.

Interval Analysis

Interval analysis makes possible the mathematically and computationally guaranteed solution of the phase stability problem. Since the mole fraction variables x_i are known to lie between zero and one, and it is easy to put physical upper and lower bounds on the molar volume v as well, a feasible interval for all variables is readily identified. By applying an interval Newton/generalized bisection approach to the entire feasible interval, enclosures of *all* the stationary points of the tangent plane distance D can be found by solving the nonlinear equation system (1)–(3), and the *global* minimum of D thus identified. This approach requires no initial guess, and is applicable to any model for the Gibbs energy, not just those derived from equations of state. For the binary system used as an example above, all five stationary points are easily found, and the global minimum at $x_1 = 0.0767$, $v = 64.06$ cm^3/mol, and $D = -0.004$ thus identified [3,6].

The efficiency of the interval approach can depend significantly on how tightly one can compute *interval extensions* for the functions involved. The interval extension of a function over a given interval is an enclosure for the range of the function over that interval. When the *natural* interval extension, that is the function range computed using interval arithmetic, is used, it may tightly bound the actual function range. However, it is not uncommon for the natural interval extension to provide a significant overestimation of the true function range, especially for functions of the complexity encountered in the phase stability and equilibrium problems.

Some tightening of bounds can be achieved by taking advantage of information about function monotonicity. Another simple and effective way to alleviate this difficulty in this context is to focus on tightening the enclosure when computing interval extensions of mole fraction weighted averages, such as $\bar{r} = \sum_{i=1}^{n} x_i r_i$, where the r_i are constants. Due to the mixing rules for determining a and b, such expressions occur frequently, both in the equation of state (2) itself, as well in the derived model $m(\mathbf{x}, v)$ for the Gibbs energy of mixing and thus in equation (3). The natural interval extension of $\bar{r}$ will yield the true range (within roundout) of the expression in the space in which all the mole fraction variables x_i are independent. However, the range can be tightened by considering the constraint that the mole fractions must sum to one. One approach for doing this is simply to eliminate one of the mole fraction variables, say x_n. Then an enclosure for the range of $\bar{r}$ in the constrained space can be determined by computing the natural interval extension of $r_n + \sum_{i=1}^{n-1}(r_i - r_n)x_i$. However, this may not yield the sharpest possible bounds on $\bar{r}$ in the constrained space.

For constructing the *exact* (within roundout) bounds on $\bar{r}$ in the constrained space, S.R. Tessier [17] and Hua, Brennecke and Stadtherr [5] have presented a very simple method, based on the observation that at the extrema of $\bar{r}$ in the constrained space, at least $n - 1$ of the mole fraction variables must be at their upper or lower bound. This observation can be derived by viewing the problem of bounding the range of $\bar{r}$ in the constrained space as a linear programming problem. As shown in [5], when the constrained space interval extensions for mole fraction weighted averages are used, together with information about function monotonicity, significant improvements in computational efficiency, nearly an order of magnitude even for small (binary and ternary) problems, can be achieved in using the interval approach for solving the phase stability problem.

For small problems, it is usually efficient to globally minimize D by finding all of its stationary points, since this does not require repeated evaluation of the range

of D. However, in general, for making a determination of phase stability or instability, finding *all* the stationary points is not really necessary, nor for larger problems, desirable. For example, if an interval is encountered over which the interval extension of D has a negative upper bound, this guarantees that there is a point at which $D < 0$, and so one can immediately conclude that the mixture is unstable without determining all the stationary points. It is also possible to easily make use of the underlying global minimization problem. Since the objective function D has a known value of zero at the mixture feed composition (tangent point), any interval over which the interval extension of D has a lower bound greater than zero cannot contain the global minimum and can be discarded, even though it may contain a stationary point (at which D will be positive and thus not of interest). Thus, one can essentially combine the interval-Newton technique with an interval branch and bound procedure in which lower bounds are generated using interval techniques.

Also, it should be noted that the global interval approach described here can easily be combined with existing local methods for determining phase stability and equilibrium. First, some (fast) local method is used. If it indicates instability then this is the correct answer as it means a point at which $D < 0$ has been found. If the local method indicates stability, however, this may not be the correct answer since the local method may have missed the global minimum in D. Applying interval analysis as described here can then be used to confirm that the mixture is stable if that is the case, or to correctly determine that it is really unstable if that is the case.

Conclusion

As demonstrated in [3,4,5,6,11,15], interval analysis can be used to solve phase stability and equilibrium problems efficiently and with complete reliability, providing a method that can guarantee with mathematical and computational certainty that the correct result is found, and thus eliminating computational problems that are encountered with conventional techniques. The method is initialization independent; it is also model independent, straightforward to use, and can be applied in connection with any equation of state or activity coefficient model for the Gibbs free energy of a mixture. There are many other problems in the analysis of phase behavior, and in chemical process analysis in general [14], that likewise are amenable to solution using this powerful approach.

See also

- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Point and Interval Taylor Operators
- ▶ Bounding Derivative Ranges
- ▶ Global Optimization in Phase and Chemical Reaction Equilibrium
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Interval Analysis: Differential Equations
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Interval Analysis: Intermediate Terms
- ▶ Interval Analysis: Nondifferentiable Problems
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ▶ Interval Analysis: Systems of Nonlinear Equations
- ▶ Interval Analysis: Unconstrained and Constrained Optimization
- ▶ Interval Analysis: Verifying Feasibility
- ▶ Interval Constraints
- ▶ Interval Fixed Point Theory
- ▶ Interval Global Optimization
- ▶ Interval Linear Systems
- ▶ Interval Newton Methods
- ▶ Optimality Criteria for Multiphase Chemical Equilibrium

References

1. Baker LE, Pierce AC, Luks KD (1982) Gibbs energy analysis of phase equilibria. Soc Petrol Eng J 22:731–742
2. Green KA, Zhou S, Luks KD (1993) The fractal response of robust solution techniques to the stationary point problem. Fluid Phase Equilib 84:49–78
3. Hua JZ, Brennecke JF, Stadtherr MA (1996) Reliable phase stability analysis for cubic equation of state models. Comput Chem Eng 20:S395–S400
4. Hua JZ, Brennecke JF, Stadtherr MA (1996) Reliable prediction of phase stability using an interval-Newton method. Fluid Phase Equilib 116:52–59
5. Hua JZ, Brennecke JF, Stadtherr MA (1998) Enhanced interval analysis for phase stability: Cubic equation of state models. Industr Eng Chem Res 37:1519–1527

6. Hua JZ, Brennecke JF, Stadtherr MA (1998) Reliable computation of phase stability using interval analysis: Cubic equation of state models. Comput Chem Eng 22:1207–1214
7. McDonald CM, Floudas CA (1995) Global optimization and analysis for the Gibbs free energy function using the UNIFAC, Wilson, and ASOG equations. Industr Eng Chem Res 34:1674–1687
8. McDonald CM, Floudas CA (1995) Global optimization for the phase and chemical equilibrium problem: Application to the NRTL equation. Comput Chem Eng 19:1111–1139
9. McDonald CM, Floudas CA (1995) Global optimization for the phase stability problem. AIChE J 41:1798–1814
10. McDonald CM, Floudas CA (1997) GLOPEQ: A new computational tool for the phase and chemical equilibrium problem. Comput Chem Eng 21:1–23
11. McKinnon KIM, Millar CG, Mongeau M (1996) Global optimization for the chemical and phase equilibrium problem using interval analysis. In: Floudas CA, Pardalos PM (eds) State of the Art in Global Optimization: Computational Methods and Applications. Kluwer, Dordrecht, pp 365–382
12. Michelsen ML (1982) The isothermal flash problem. Part I: Stability. Fluid Phase Equilib 9:1–19
13. Michelsen ML (1982) The isothermal flash problem. Part II: Phase-split calculation. Fluid Phase Equilib 9:21–40
14. Schnepper CA, Stadtherr MA (1996) Robust process simulation using interval methods. Comput Chem Eng 20:187–199
15. Stadtherr MA, Schnepper CA, Brennecke JF (1995) Robust phase stability analysis using interval methods. AIChE Symp Ser 91(304):356–359
16. Sun AC, Seider WD (1995) Homotopy-continuation method for stability analysis in the global minimization of the Gibbs free energy. Fluid Phase Equilib 103:213–249
17. Tessier SR (1997) Enhanced interval analysis for phase stability: Excess Gibbs energy models. MSc Thesis Dept Chemical Engin Univ Notre Dame
18. Wasylkiewicz SK, Sridhar LN, Malone MF, Doherty MF (1996) Global stability analysis and calculation of liquid-liquid equilibrium in multicomponent mixtures. Industr Eng Chem Res 35:1395–1408

Global Optimization Based on Statistical Models

Antanas Žilinskas
Institute Math. and Informatics,
Vytautas Magnus University, Vilnius, Lithuania

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Global optimization; Statistical models; Multimodal functions; Rational choice

Many practically significant problems require to optimize in a 'black box' situation, when the objective function is given by a code, but its structure is not known. In some algorithms, developed for such a case, different heuristic ideas are implemented. A disadvantage of the heuristic algorithms is dependence of the results on many parameters which choice is difficult because of rather vague meaning of these parameters. To develop a theory of global optimization the 'black box' should be replaced by a 'grey box' corresponding to some model of predictability/uncertainty of values of an objective function.

A model of an objective function is an important counterpart of any optimization theory (e. g., quadratic models are widely used to construct algorithms for local nonlinear optimization). The uncertainty on values of multimodal functions at the arbitrary points of the feasible region is more essential than uncertainty on the value of the objective function which will be calculated at the current iteration of the local descent. Therefore, the global optimization models that describe the objective function with respect to information obtained during the previous iterations are different from polynomial models used in local optimization. Different models may be used; e. g., a deterministic model, defining the guaranteed intervals for unknown function values, or a statistical model, modeling the uncertainty on function value by means of a random variable. The choice of a model is crucial because it defines the methodology of constructing the corresponding algorithms. A Lipschitzian type model enables the construction of global optimization algorithms with guaranteed (worst case) accuracy. However, the number of function evaluations in the worst case grows drastically with the dimensionality of the problem and the prescribed accuracy. In spite of this pessimistic theoretical result many practical rather complicated problems have been

solved heuristically. Because a *heuristics* is a human experience based methodology, oriented towards average (typical, normal) conditions, it seems reasonable to develop a theory formalizing the principle of rational behavior with respect to average conditions in global optimization. The average rationality is well justified for playing a 'game against nature' which models optimization conditions better than an antagonistic game where the principle of minimax (guaranteed result) is well justified. The method ology of average rationality was applied to develop the general theory of rational choice under statistically interpreted uncertainty [4]. This general theory was further specified to develop the theory of global optimization based on statistical models of multimodal functions [11].

To construct a statistical model of multimodal function $f(x)$, $x \in A \subset \mathbf{R}^n$, the *axiomatic approach* is applied: the rationality of comparisons of likelihood of different values of $f(\cdot)$ is postulated by simple, intuitively acceptable axioms, and it is proved that the interpretation of an unknown value $f(x)$ as a Gaussian random variable ξ_x is compatible with the axioms. The parameters of ξ_x (mean value $m(x|(x_i, y_i))$ and variance $\sigma^2(x|(x_i, y_i))$, where $y_i = f(x_i)$ are known function values obtained during the search) are introduced by axiomatic theory of extrapolation under uncertainty. In the one-dimensional case both functions are very simple: $m(x|(x_i, y_i))$ is piecewise linear (connecting the neighboring trial points) and σ^2 $(x|(x_i, y_i))$ is piecewise quadratic.

By means of further (more restrictive) assumptions, the statistical models, corresponding to the stochastic functions, may be specified. The one-dimensional model corresponding to the Wiener process was introduced in [3]. However, the specification of a model as a stochastic function is not very reasonable: this normally involves additional very serious implementation difficulties and does not help to choose the model according to the a priori information on the problem. Using a statistical model the algorithm is constructed maximizing the probability to find better points than those found during the previous search. Such a strategy is justified also by the natural axioms of rationality of search. In the one-dimensional case the algorithm is easy to implement. In the multidimensional case, an auxiliary optimization problem must be solved [8].

Although the algorithm is based on the statistical model it is described without of use of randomization. Therefore it may be investigated by usual deterministic methods, e. g. the convergence of the algorithm in the is proved under weak assumptions on the underlying statistical model (continuity of $m(x|\cdot)$, $\sigma^2(x|\cdot)$ and weak dependence of both characteristics at point x on (x_i, y_i) for relatively remote points x_i [8]).

The models and algorithms of this approach are well grounded theoretically because they are derived from natural assumptions on rational behavior of an optimizer. As a topic for further research, the theory of average complexity seems very prospective. It would be important to evaluate the complexity of practically efficient algorithms constructed by the approach as well as to obtain general bounds and compare them with those obtained for Lipschitzian algorithms. The first results in this direction are interesting even for the one-dimensional case: the limit distribution of error of passive random search in case of the Wiener model exists or does not exist depending on a subtle interpretation of the model [2]. Other important theoretical topics are: developing dual (global-local) models for the multidimensional case, and justification of multidimensional statistical models oriented towards algorithms of the branch and bound type (cf. also ► Integer programming: Branch and bound methods), whose auxiliary computations would be essentially less time consuming than maximization of the probability over the whole feasible region at each iteration.

Many algorithms were constructed using different statistical models and more or less theoretically justified ideas. For example, a Bayesian algorithm (cf. also ► Bayesian global optimization) is defined by minimizing the average error with respect to the stochastic function chosen for a model [5]. By interpolation, the next calculation of a value of the objective function is performed at minimum point of $m(\cdot|(x_i, y_i))$ [1,6]. For the information-statistical method, an *ad hoc* one-dimensional model is constructed [1,7]. The algorithms may be generalized for the case with 'noisy' functions, see for example the algorithm in [8,10].

The known results from the theory of stochastic functions as well as axiomatic construction of statistical models do not give numerically tractable models which are completely adequate to describe local and global properties of a typical global optimization prob-

lem [1]. But in the framework of statistical models the adequacy, e. g., to local prop erties of the objective function, might be tested as a statistical hypothesis. If the statistical model is locally inadequate in a subset of the feasible region, then the objective function is assumed unimodal in this subset and a local minimum of $f(x)$ may be found by a local technique. An example of the combination of global and local search with a stopping rule corresponding to a high probability of finding the global minimum is presented in [9].

In the case of one-dimensional global optimization there are many competing algorithms including algorithms based on statistical models [8]. The algorithms representing different approaches may be compared with sufficient reliability by means of experimental testing. Since the codes in one-dimensional case are very precise realizations of theoretical algorithms then influence of implementation specifics is insignificant (at least with respect to multidimensional cases) and the comparison results may be generalized from codes to corresponding approaches. The results in [8] show that the algorithm from [9] and its modification [8] outperforms algorithms based on Lipschitzian type models even if a good estimate of the Lipschitz constant is available. The comparison of multidimensional algorithms is methodologically more difficult, partly because of very different stopping conditions. But generally speaking, the algorithms based on statistical models are efficient with respect to the number of evaluations of the objective function for the multimodal functions up to 10–15 variables [8]. The auxiliary computations require much computing time and computer memory. Therefore, such algorithms are rational to use for the problems, whose objective unction is expensive to evaluate. If an objective function is cheap to evaluate, the gain obtained from a low number of function evaluations may be less than the loss caused by the auxiliary computations.

A detailed review of the subject is presented in [8]; further references may be found in [1].

See also

- ▶ Adaptive Global Search
- ▶ Adaptive Simulated Annealing and its Application to Protein Folding
- ▶ αBB Algorithm
- ▶ Bayesian Global Optimization
- ▶ Continuous Global Optimization: Applications
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Differential Equations and Global Optimization
- ▶ DIRECT Global Optimization Algorithm
- ▶ Genetic Algorithms for Protein Structure Prediction
- ▶ Global Optimization in Binary Star Astronomy
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization Using Space Filling
- ▶ Monte-Carlo Simulated Annealing in Protein Folding
- ▶ Packet Annealing
- ▶ Random Search Methods
- ▶ Simulated Annealing
- ▶ Simulated Annealing Methods in Protein Folding
- ▶ Stochastic Global Optimization: Stopping Rules
- ▶ Stochastic Global Optimization: Two-phase Methods
- ▶ Topology of Global Optimization

References

1. Boender G, Romeijn E (1995) Stochastic methods. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 829–869
2. Calvin J, Glynn P (1997) Average case behavior of random search for the maximum. J Appl Probab 34:631–642
3. Kushner H (1962) A versatile stochastic model of a function of unknown and time-varying form. J Math Anal Appl 5:150–167
4. Luce D, Suppes P (1965) Preference, utility and subjective probability. In: Luce D, Bush R, Galanter E (eds) Handbook Math. Psychology. Wiley, New York, pp 249–410
5. Mockus J (1989) Bayesian approach to global optimization. Kluwer, Dordrecht
6. Shagen I (1980) Stochastic interpolation applied to the optimization of expensive objective functions. In: COMPSTAT 1980. Physica Verlag, Heidelberg, pp 302–307
7. Strongin R (1978) Numerical methods in multiextremal optimization. Nauka, Moscow
8. Törn A, Žilinskas A (1989) Global optimization. Springer, Berlin
9. Žilinskas A (1978) Optimization of one-dimensional multimodal functions, Algorithm AS 133. Applied Statist, 23:367–385
10. Žilinskas A (1980) MIMUN-optimization of one-dimensional multimodal functions in the presence of noise, Algoritmus 44. Aplikace Mat 25:392–402
11. Žilinskas A (1985) Axiomatic characterisation of a global optimization algorithm and investigation of it's search strategy. Oper Res Lett 4:35–39

Global Optimization in Batch Design Under Uncertainty

S. T. Harding, Christodoulos A. Floudas
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C26

Article Outline

Keywords
Conceptual Framework
Constraints on Batch Size
Minimum Cycle Time
Constraints on Production Time
Demand Constraints
Economic Objective Function
Sources of Uncertainty
Uncertainty in Process Parameters
Uncertainty in Product Demand
Global Optimization Approaches
The GOP Approach
αBB Approach
Other Types of Batch Plants
Mixed-Product Campaign
Multipurpose Batch Plant-Single Equipment Sequence
Multipurpose Batch Plant-Multiple Equipment Sequence
See also
References

Keywords

Batch plant design; Multiproduct; Multipurpose; Uncertainty

Batch processes are a popular method for manufacturing products in low volume or that require several complicated steps in the synthesis procedure. The growth in the market for specialty chemicals has contributed to the demand for efficient batch plants. Batch processes are especially attractive due to their inherent *flexibility*. They can accommodate a wide range of production requirements. Batch equipment can be reconfigured to produce more than one product. Finally, certain pieces of equipment in batch processes can be used for more than one task.

An important area of concern in the design of batch processes is their ability to accommodate changes in production requirements and processing parameters. The key issue is: given some degree of uncertainty in a) the future demand for the products and b) the parameters that describe the chemical and physical steps involved in the process, what is the appropriate amount of flexibility the process should possess so as to maintain feasible operation while maximizing profits?

Many methods have been proposed for the design of batch plants under known market conditions and nominal operating conditions. Two major classes of batch plant designs are *multiproduct* plants and *multipurpose* plants. In the multiproduct plant, all products follow the same sequence of processing steps. Typically, one product is produced at a time in what is termed a *single-product campaign* (SPC). Multipurpose batch plants allow products to be processed using different sequences of equipment, and in some cases products can be produced simultaneously.

While significant progress has been made in the design and scheduling of batch plants, until recently the issues of flexibility and design under uncertainty have received little attention. Among the first to address the problem of batch plant design under uncertainty in a novel way were [10], and [8]. They divided the variables in the design problem into five categories: structural, design, state, operating, and uncertain. Structural variables describe the interconnections of the equipment in the plant. Design variables describe the size of the process equipment and are fixed once the plant is constructed. State variables are dependent variables and are determined once the design and operating variables are specified. Operating variables are those whose values can be changed in response to variations in the uncertain variables. Finally, the uncertain parameters are the quantities that can have random values which can be described by a *probability distribution*. Usually the uncertain parameters have normal distributions and are considered to be independent of each other. [8] also introduced the distinction between variations which have short-term effects and those with long-term effects. [18] extended this idea, suggesting a distinction between 'hard' and 'soft' constraints in which the former must be satisfied for feasible plant operation, but the latter may be violated, subject to a penalty in the objective function. They considered the time required to produce a product as uncertain and developed a problem formulation.

In [12], and [13] the authors addressed the problem of multiproduct batch plant design with uncertainties in both demand for the products and in technical parameters such as processing times and size factors. They restricted their designs to one piece of equipment per stage. [3] presented several variations on the problem of design with uncertain demands. They used interval methods to develop different solution procedures, including a two-stage approach and a penalty function approach. Another type of batch plant is the multipurpose plant. [14] proposed a *scenario*-based approach for the design of multipurpose batch plants with uncertain production requirements. The multipurpose approach resulted in a large scale MILP model for which efficient techniques for obtaining good upper and lower bounds were proposed. [15] developed a model for the multiproduct batch design problem which takes into account uncertainties in the product demands and in equipment availability. They considered the problem of design feasibility separately from the maximization of profits and presented an approach for achieving both criteria. [16] addressed the problem of uncertain demands, and used a scenario-based approach with discrete probability distributions for the demands. In addition, they considered the *scheduling* problem as a second stage, following the design problem. [6], and [7] considered the multiproduct batch plant design problem based on a stochastic programming formulation. They developed a relaxation of the production feasibility requirement and added a penalty term to the objective function to account for partial feasibility. Through this analysis, the problem can be reformulated as a single large scale *nonconvex optimization problem*. [2] extended this work to the design of multipurpose batch plants and implemented an efficient Gaussian quadrature technique to improve the estimation of the expected profit. [5] identified special structures in the nonconvex constraints for multiproduct and multipurpose batch design formulations. These properties can be exploited to obtain tight bounds on the global solution. This allows very large scale design problems to be solved in reasonable CPU time using the αBB method of [1].

Conceptual Framework

Most batch design problems are variations on the same basic model of a batch plant. The plant consists of M processing stages where each stage j contains N_j identical pieces of equipment. The volume of each unit, V_j, is a design variable, and the number of units per stage, N_j, may be a variable or a fixed parameter.

In the batch plant, NP products are to be made, and the amount of each produced is Q_i. Each product is produced in a number of batches of identical size, B_i. Using these definitions, a number of constraints on the design of the plant can be imposed. These constraints are:

1) an upper limit on the batch size,
2) a lower limit on the amount of time between batches,
3) an upper limit on the total processing time allowed, and
4) a constraint on the production related to the demand for each product. The basic form of these constraints is shown below, for a multiproduct batch plant with single-product campaigns.

Constraints on Batch Size

The batch size for each product i cannot be larger than the size of the pieces of equipment in each stage j. This can be written

$$B_i \le \frac{V_j}{S_{ij}},$$
$$i = 1, \ldots, NP, \quad j = 1, \ldots, M.$$

The size factor, S_{ij}, is the capacity required in stage j to process one unit of product i.

Minimum Cycle Time

In order to make sure that each batch is processed separately in a given stage, one batch cannot begin processing until the previous batch has been processed for a certain amount of time. This is called the *cycle time*

$$T_{Li} \ge \frac{t_{ij}}{N_j},$$
$$i = 1, \ldots, NP, \quad j = 1, \ldots, M.$$

The time factor, t_{ij} is the amount of time to process one batch of product i in stage j.

Constraints on Production Time

The amount of time needed to produce all of the batches must be less than the total time available, H,

$$\sum_{i=1}^{NP} \frac{Q_i}{B_i} T_{Li} \leq H.$$

Demand Constraints

The production for each product must meet the demand.

$$Q_i = D_i.$$

Economic Objective Function

The objective is to maximize profits. The *profit* is calculated by subtracting the annualized capital costs from the revenues:

$$\text{Profit} = \sum_{i=1}^{NP} Q_i \cdot p_i - \sum_{j=1}^{M} \alpha_j N_j V_j^{\beta_j},$$

where p_i is the price of product i. The annualization factor for the cost of the units in stage j is α_j.

In the case where the number of units per stage, N_j is variable and/or the unit sizes, V_j, take only discrete values, this problem is a *mixed integer nonlinear optimization* problem (MINLP). If N_j is fixed and the unit sizes are continuous, the problem is a *nonlinear program* (NLP). In either case, the problem is nonconvex, therefore conventional mixed integer and nonlinear solvers cannot be used robustly. Instead, global optimization techniques must be employed to guarantee that the optimal solution is located.

Sources of Uncertainty

Within the mathematical framework for a multiproduct batch plant there are a number of possible sources of uncertainty. The most commonly studied are uncertainty in the process parameters, like the size factors, S_{ij}, and the time factors, t_{ij}, and uncertainty in the product demand, D_i. In addition to these, [3] considered uncertainty in the time horizon, H, and in the product prices, p_i.

Uncertainty in the process parameters is model inherent uncertainty, as classified by [11]. That is, uncertainty in the process parameters affects the feasible operation of the batch plant. Conversely, uncertainty in the product demand is an external source of uncertainty, therefore it only affects the objective function, and not the feasibility of the plant design.

Uncertainty in Process Parameters

The size factors and processing times affect the feasible design and operation of the batch plant. The goal is to design a plant that can operate feasibly, even if there is some uncertainty in the values of these parameters. The approach that is commonly followed is to consider a number of different scenarios, where each scenario corresponds to a set of parameter realizations. For example, if the size factors, S_{ij}, have some nominal value, $\overline{S}_{ij}$, then one scenario is that all of the size factors are at their nominal value. Similarly, if we have some knowledge about the amount of uncertainty in the size factors, we can construct a lower extreme scenario, where each size factor is at its lower bound, S_{ij}^L, and an upper extreme scenario, S_{ij}^U. The new set of size factors, reflecting the different scenarios is represented by the parameter S_{ij}^p. The scenarios can be weighted using the factor, w^p.

The set of constraints for the batch design problem must be modified so that the design is feasible over the whole set of scenarios, P:

$$B_i \leq \frac{V_j}{S_{ij}^p}, \quad T_{Li}^p \geq \frac{t_{ij}^p}{N_j},$$

$$\sum_{i=1}^{NP} \frac{Q_i^p}{B_i} T_{Li}^p \leq H.$$

Uncertainty in Product Demand

Uncertainty in the demand for the products affects the profitability of the plant. In this case, the product demand is given by a probability distribution function $J(\theta_i)$ where θ_i represents the uncertain demand for product i. The calculation of the expected revenues requires the integration over an optimization problem:

$$\mathsf{E}_\theta \left[\max_{Q_i} \sum_{i=1}^{NP} p_i Q_i \right] = \int_{\theta \in R(V_j, N_j)} \max_{Q_i} \left\{ \sum_{i=1}^{NP} p_i Q_i \right\} J(\theta)\, d\theta \,. \quad (1)$$

The integration should be performed over the feasible region of the plant, which is unknown at the design stage. See [6] for a *Gaussian quadrature* approach to discretize the integration. The range of uncertain demands is covered by a grid, where each point on the grid represents a set of demand realizations, and is assigned a weight corresponding to its probability, $\omega^q J^q$. The set of quadrature points is represented by Q. The expected revenues are now calculated as a multiple summation:

$$\mathsf{E}_\theta \left[\max_{Q_i} \sum_{i=1}^{NP} p_i Q_i \right] = \sum_{p=1}^{P} \frac{1}{w^p} \sum_{q=1}^{Q} \omega^q J^q \sum_{i=1}^{NP} p_i Q_i^{qp}.$$

In addition, the time horizon constraint must be modified:

$$\sum_{i=1}^{NP} \frac{Q_i^{qp}}{B_i} T_{Li}^p \le H, \quad \forall p \in P, \ \forall q \in Q.$$

Global Optimization Approaches

The set of constraints for the design of a multiproduct batch plant under uncertainty form a nonconvex optimization problem. Global optimization techniques must be used in order to ensure that the true optimal design is located.

Following the analysis of [9], an exponential transformation can be applied, reducing the number of nonlinear terms in the model.

$$\begin{aligned} V_j &= \exp(v_j), \quad \forall j \in M, \\ B_i &= \exp(b_i), \quad \forall i \in NP, \\ T_{Li}^p &= \exp(t_{Li}^p), \quad \forall i \in NP. \end{aligned}$$

In [5] and [6] global optimization methods were developed to solve this problem, where the number of units in each stage, N_j, is fixed. In this case, the cycle time becomes a parameter, determined by,

$$t_{Li}^p = \max_j \left\{ \ln \left(\frac{t_{ij}^p}{N_j} \right) \right\}, \quad \forall i \in NP, \ \forall p \in P.$$

The nonlinear optimization problem to be solved is written as a minimization:

$$\begin{cases} \min\limits_{b_i, v_j, Q_i^{qp}} & \delta \sum\limits_{j=1}^{M} \alpha_j N_j \exp\left(\beta_j v_j\right) \\ & \quad - \sum\limits_{p=1}^{P} \frac{1}{w^p} \sum\limits_{q=1}^{Q} \omega^q J^q \sum\limits_{i=1}^{NP} p_i Q_i^{qp} \\ & \quad + \gamma \sum\limits_{p=1}^{P} \frac{1}{w^p} \sum\limits_{q=1}^{Q} \omega^q J^q \sum\limits_{i=1}^{NP} p_i \left(\theta_i^q - Q_i^{qp} \right) \\ \text{s.t.} & v_j \ge \ln(S_{ij}^p) + b_i \\ & \sum\limits_{i=1}^{NP} Q_i^{qp} \cdot \exp(t_{Li}^p - b_i) \le H \\ & \theta_i^L \le Q_i^{qp} \le \theta_i^q \\ & \ln(V_j^L) \le v_j \le \ln(V_j^U) \\ & \min\limits_{j,p} \ln \left(\frac{V_j^L}{S_{ij}^p} \right) \le b_i \le \min\limits_{j,p} \ln \left(\frac{V_j^U}{S_{ij}^p} \right). \end{cases} \tag{2}$$

Note that the time horizon constraint is the only nonconvex constraint remaining in the problem formulation. A *penalty* term is added to the objective function to account for unsatisfied demand, the penalty parameter is γ.

The GOP Approach

In [7] and [2] the GOP algorithm of [4,17] has been applied to solve design formulations for both multipurpose and multiproduct batch plants. GOP converges to the global optimum solution by solving a primal problem and a number of relaxed dual problems in each iteration. In [7] it is observed that if the variables in the batch design problem are partitioned so that $y = \{v_j, b_i\}$ and $x = \{Q_i^{qp}\}$, then the problem is convex in y for every fixed x, and linear in x for every fixed y. This satisfies Condition A) of the GOP algorithm.

A property was developed in [7] that allows the number of relaxed duals per iteration to be reduced from $2^{NP \cdot Q}$ to 2^{NP}, making the problem computationally tractable.

αBB Approach

The αBB approach of [1] was applied in [5] to solve both multiproduct and multipurpose design formulations. αBB is a *branch and bound* approach that

converges to the global solution by solving a sequence of upper and lower bounding problems. The lower bounding problem is formulated by subtracting a quadratic term, multiplied by the constant α, from each of the nonconvex terms, thus convexifying the problem. Often, the size of the α term must be estimated, resulting in poor lower bounds in the first few levels of the branch and bound tree. However, the nonconvex terms in the batch plant design formulation allow the *exact* value of α to be calculated, resulting in a tight lower bound on the global solution. This technique has been used to find the optimal design for a multiproduct batch plant with 5 products in 6 stages. This corresponds to a nonconvex NLP with 15,636 variables, 3155 constraints, and 15,625 nonconvex terms.

Other Types of Batch Plants

In addition to the multiproduct batch plant with single-product campaign illustrated in the preceding sections, there are many other batch plant design formulations that can be adapted to consider the issue of uncertainty in design.

Mixed-Product Campaign

This is another example of a multiproduct batch plant. In this case, storage of the intermediate products is allowed between processing steps. In addition, batches of different products can be alternated. This allows a reduction in the total production time. Rather than being limited by the largest cycle time for all stages, this method calculates the total production time for each stage:

$$T_j^{qp,\mathrm{tot}} \geq \sum_{i=1}^{NP} \left(\frac{Q_i^{qp}}{B_i} \right) t_{ij}^p.$$

The total time for each stage must be less than the total time allowed:

$$H \geq T_j^{qp,\mathrm{tot}} \geq \sum_{i=1}^{NP} \left(\frac{Q_i^{qp}}{B_i} \right) t_{ij}^p.$$

This can be written

$$\sum_{i=1}^{NP} \left(\frac{Q_i^{qp}}{B_i} \right) t_{ij}^p \leq H.$$

Note that this constraint has the same form as the time horizon constraint for the single-product campaign formulation.

Multipurpose Batch Plant-Single Equipment Sequence

In a multipurpose batch plant, the equipment can be used for more than one function, therefore each product may have a different route through the plant. In the single equipment sequence case, there is one distinct route for each product. Production is carried out in a sequence of campaigns L, and there may be more than one product produced simultaneously in a campaign, h. The time needed for each campaign, C_h, is based on the maximum cycle time for all products in the campaign,

$$\sum_{h=1}^{L} \alpha_{hi} C_h^{qp} \geq \left(\frac{Q_i^{qp}}{B_i} \right) T_{Li}^p,$$

where

$$\alpha_{hi} = \begin{cases} 1 & \text{if product } i \text{ is allowed} \\ & \quad \text{in campaign } h, \\ 0 & \text{else.} \end{cases}$$

Finally, the sum of all campaign times must be less than the total time available:

$$\sum_{h=1}^{L} C_h^{qp} \leq H.$$

Multipurpose Batch Plant-Multiple Equipment Sequence

In this case, there are multiple routes through the plant for each product i, PR_i. The total amount of product i produced is the sum over the production of i in each route:

$$Q_i^{qp} = \sum_{r \in PR_i} q_r^{qp}.$$

The time for campaign C_h is based on the maximum cycle time for each route in the campaign,

$$\sum_{h=1}^{L} \alpha_{hr} C_h^{qp} \geq \left(\frac{q_r^{qp}}{B_r} \right) t_{Lr}^p.$$

The sum of all campaign times must be less than the total time available,

$$\sum_{h=1}^{L} C_h^{qp} \leq H.$$

Note that in both of the multipurpose batch design formulations shown above, the constraints that are added are either linear, or have the exact same form of nonconvexities as shown for the multiproduct batch design formulation. Therefore, the global optimization techniques discussed in Section 'Global Optimization Approaches' are applicable to these problems.

See also

- ▶ αBB Algorithm
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Global Optimization in Generalized Geometric Programming
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization in Phase and Chemical Reaction Equilibrium
- ▶ Interval Global Optimization
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Global Optimization with αBB
- ▶ Smooth Nonlinear Nonconvex Optimization

References

1. Androulakis IP, Maranas CD, Floudas CA (1995) αBB: A global optimization method for general constrained nonconvex problems. J Global Optim 7:337–363
2. Epperly TGW, Ierapetritou MG, Pistikopoulos EN (1997) On the global and efficient solution of stochastic batch plant design problems. Comput Chem Eng 21:1411–1431
3. Fichtner G, Reinhart H-J, Rippin DWT (1990) The design of flexible chemical plants by the application of interval mathematics. Comput Chem Eng 14:1311–1316
4. Floudas CA, Visweswaran V (1990) A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: I. Theory. Comput Chem Eng 14:1397–1417
5. Harding ST, Floudas CA (1997) Global optimization in multiproduct and multipurpose batch design under uncertainty. Industr Eng Chem Res 36:1644–1664
6. Ierapetritou MG, Pistikopoulos EN (1995) Design of multiproduct batch plants with uncertain demands. Comput Chem Engin 19:S627–S632
7. Ierapetritou MG, Pistikopoulos EN (1996) Batch plant design and operations under uncertainty. Industr Eng Chem Res 35:772–787
8. Johns WR, Marketos G, Rippin DWT (1978) The optimal design of chemical plant to meet time-varying demands in the presence of technological and commercial uncertainty. Trans Inst Chem Eng 56:249–257
9. Kocis GR, Grossmann IE (1988) Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis. Industr Eng Chem Res 27:1407
10. Marketos G (1975) The optimal design of chemical plant considering uncertainty and changing circumstances. PhD Thesis, ETH Zurich
11. Pistikopoulos EN (1995) Uncertainty in process design and operations. Comput Chem Eng 19:S553–S563
12. Reinhart HJ, Rippin DWT (1986) The design of flexible batch chemical plants. In: 1986 AIChE Annual Meeting
13. Reinhart HJ, Rippin DWT (1987) Design of flexible multiproduct plants: A new procedure for optimal equipment sizing under uncertainty. In: 1987 AIChE Annual Meeting
14. Shah N, Pantelides CC (1992) Design of multipurpose batch plants with uncertain production requirements. Industr Eng Chem Res 31:1325–1337
15. Straub DA, Grossmann IE (1992) Evaluation and optimization of stochastic flexibility in multiproduct batch plants. Comput Chem Eng 16:69–87
16. Subrahmanyam S, Pekny JF, Reklaitis GV (1994) Design of batch chemical plants under market uncertainty. Industr Eng Chem Res 33:2688–2701
17. Visweswaran V, Floudas CA (1993) New properties and computational improvement of the GOP algorithm for problems with quadratic objective function and constraints. J Global Optim 3(3):439–462
18. Wellons HS, Reklaitis GV (1989) The design of multiproduct batch plants under uncertainty with staged expansion. Comput Chem Eng 13:115–126

Global Optimization in Binary Star Astronomy

GO4BSA

DIMITRI POURBAIX
Royal Observatory of Belgium, Brussels, Belgium

MSC2000: 90C26, 90C90

Article Outline

Keywords
Astronomical Problem
Objective Function

Global Search
Conclusion
See also
References

Keywords

Astronomy; Binary; Star; Orbit; Mass

The global optimization techniques are still quite unpopular in the astronomical community, in particular, among the double stars astronomers. Among the reasons of their reticence one finds a long practice of manual and graphical methods, 'least squares' adjustments of a linearized objective function, differential correction, etc.

This article does not present, unfortunately, the state of the art in *orbits determination*, even if a few astronomers, mostly young ones, tries to convince the others that a global minimization step is useful. This article presents a possible way to obtain the orbital parameters of double-lined spectroscopic visual *binaries*.

Astronomical Problem

The generic terms 'binary star' designate two stars that are gravitationally linked together. Since J. Kepler, one knows that such an interaction leads to an elliptic orbital motion of one star around each the other (Kepler's first law). The Kepler third law tells us that there is a simple relation between the orbital period (P), the semimajor axis of the relative orbit (a) and the *mass* sum of the 2 stars (M_A (the mass of the brighter star) and M_B (the mass of the fainter component)):

$$\frac{a^3}{P^2} = M_A + M_B,$$

where a is expressed in astronomical unit (1 A.U. is equal to the average distance of the Earth from the Sun), P is expressed in years and the masses in solar masses ($M_\odot$). This relation is still, almost 400 years after Kepler, the only direct and hypothesis-free method to estimate stellar masses.

A visual binary corresponds to a situation where the 2 stars are visually resolved and the orbital motion, projected on the plane orthogonal to the sight direction, can be perceived. From the relative positions of B with respect to A along time (t, x and y), one can extract the 7 parameters characterizing the visual orbit. Among these parameters, there are P and the angular value of a (expressed in seconds of arc). The latter cannot be converted into its linear value in A.U. unless the distance to the binary system is known (or, equivalently, the parallax of the system, ϖ, is known).

A binary star is spectroscopic if the motion of its spectral lines is observable. This motion is due to the *Doppler effect*: all lines issued from one star are shifted toward the blue (red) side of the spectrum when that star is moving toward (away from) the observer. The wavelength shift between the laboratory wavelength, λ_L, and the observed one, λ_O, is connected to the radial velocity V through:

$$\frac{\lambda_O - \lambda_L}{\lambda_L} = \frac{V}{c}$$

where c stands for the speed of the light in the vacuum. In a double-lined spectroscopic binary, lines from the two components are seen in the spectrum.

The radial velocity curve $((t, V_A), (t, V_B))$ of each component along time shows a periodic variation. Lets K_A designates the amplitude of the radial velocity curve of component A and K_B the amplitude of component B. There is a relation between the mass ratio and the K. values:

$$\frac{K_A}{K_B} = \frac{M_B}{M_A}$$

The amplitudes are usually expressed in km/s.

Hence, if a binary star is simultaneously visual and double-lined spectroscopic, one can extract the individual masses and the distance to the system with no extra hypothesis.

Objective Function

To describe the observations of a double-lined *spectroscopic visual binary* requires at least 10 parameters. By observations, one means the relative positions of the fainter component with respect to the brighter star and the radial velocities of both components. Why more than 10 parameters could be necessary is beyond the scope of this paper. Among the different possible sets of 10 parameters, we select:

- $a^{(")}$: the angular semimajor axis of the relative orbit of the fainter component around the brighter star;
- i: the inclination of the orbital plane with respect to the plane orthogonal to the direction sight;

- ω: the argument of the periastron;
- Ω: the longitude of the ascending node;
- e: the eccentricity;
- P: the period;
- T: the periastron epoch (one of them);
- V_0: the radial velocity of the system's center of mass;
- ϖ: the *parallax* of the system;
- κ: the ratio of the semimajor axis (relative to the brighter component) to the sum of the two semimajor axes.

The most natural way to combine visual and spectroscopic observations is to use a least squares approach and to seek the minimum of an expression like:

$$\begin{aligned} &D(a, i, \ldots, \varpi, \kappa) \\ &= \sum_{j=1}^{N_v} \left[\left(\frac{\overset{o}{x}_j - \widehat{x}_j}{\sigma_{x_j}} \right)^2 + \left(\frac{\overset{o}{y}_j - \widehat{y}_j}{\sigma_{y_j}} \right)^2 \right] \\ &+ \sum_{k=1}^{N_{sA}} \left(\frac{\overset{o}{V}_{A_k} - \widehat{V}_{A_k}}{\sigma_{V_{A_k}}} \right)^2 + \sum_{l=1}^{N_{sB}} \left(\frac{\overset{o}{V}_{B_l} - \widehat{V}_{B_l}}{\sigma_{V_{B_l}}} \right)^2 \end{aligned} \quad (1)$$

where the hat (super) stands for the adjusted (observed) quantity and $\sigma_\cdot$ are the a priori known (or estimated) standard deviations of the observations.

In fact, yet this idea of combining the two aspects of the orbit is unusual. Most of the time, astronomers keep the separation when computing the orbital parameters. Visual observers compute their own orbit and spectroscopists theirs: one group simply fixes some parameters (w, e, P and T) to the values obtained by the other group (e. g., [5]). A few papers only presents a *simultaneous adjustment* of the ten parameters (e. g., [12,18]).

The reader could be puzzled by the fact that the expression of D seems to be too kind to have numerous local minima and to require a global optimization method to be minimized. A description of how x, y, V_A and V_B are computed is going to justify our approach.

The visual orbit requires

$$\begin{aligned} x &= AX + FY, \\ y &= BX + GY, \\ X &= \cos E - e, \\ Y &= \sqrt{1 - e^2} \sin E, \end{aligned}$$

where X and Y (x and y) are the angular rectangular coordinates, in the orbital (tangential) plane, of the fainter component with respect to the brighter one; A, B, F and G are the Thiele–Innes constants, expressed in terms of $a^{(\prime\prime)}$, i, ω and Ω as

$$\begin{aligned} A &= a^{(\prime\prime)}(\cos\omega\cos\Omega - \sin\omega\sin\Omega\cos i), \\ B &= a^{(\prime\prime)}(\cos\omega\sin\Omega + \sin\omega\cos\Omega\cos i), \\ F &= a^{(\prime\prime)}(-\sin\omega\cos\Omega - \cos\omega\sin\Omega\cos i), \\ G &= a^{(\prime\prime)}(-\sin\omega\sin\Omega + \cos\omega\cos\Omega\cos i). \end{aligned}$$

E is the eccentric anomaly at time t, determined unambiguously by Kepler's equation

$$E - e\sin E = \frac{2\pi}{P}(t - T).$$

For a spectroscopic orbit j ($j = A$ or $j = B$), one needs

$$\begin{aligned} V_A &= V_0 - K_A(\cos(\omega + \nu) + e\cos\omega), \\ V_B &= V_0 + K_B(\cos(\omega + \nu) + e\cos\omega), \\ K_j &= \frac{2\pi\, a_j^{(\mathrm{km})} \sin i}{86400 \cdot 365.242198781 P\sqrt{1 - e^2}}, \\ \tan\frac{\nu}{2} &= \sqrt{\frac{1+e}{1-e}} \tan\frac{E}{2}. \end{aligned}$$

The angular separation in arcseconds is converted into its linear value using

$$\begin{aligned} a^{(\mathrm{km})} &= \frac{a^{(\prime\prime)}}{\varpi} \cdot 1.49598 \cdot 10^8, \\ a_A^{(\mathrm{km})} &= \kappa a^{(\mathrm{km})}, \\ a_B^{(\mathrm{km})} &= (1 - \kappa) a^{(\mathrm{km})}. \end{aligned}$$

Global Search

In front of a low-dimension but highly nonlinear problem, what can be used to find the minimum of an expression such as D (equation (1))? *Simulated annealing* ([8,11]) has already been successfully applied to the determination of the orbital parameters of *visual binaries* [14]. In that case, only 7 parameters are required, but the nature of the problem seems close enough to the current one to be tempted to use the same approach.

The implementation of SA used for the visual problem gives satisfaction ([1,15]). Nevertheless, the increase of the working space dimension is, by itself, enough to justify the search for an improved algorithm for the combined spectroscopic-visual problem.

Among the few SA implementations for continuous functions, the one in 'Numerical Recipes' [16] was selected. Although the published code behaves very well,

some improvements (at least for our purpose) are possible. We are going to focus on modifications of the basic algorithm, mainly some improvements of the guess generator. A rough pseudocode of the algorithm in [16] is given below:

```
DO
        use a simplex to get a new solution;
        decrease the temperature;
WHILE   (temperature > T_min);
```

Suggested pseudocode after [16]

Let's first remind that the guess generator proposed in [16] is based on a thermally disturbed *simplex* [13]. When the temperature approaches 0, the generator reduces to the *Nelder–Mead algorithm* and a local convergence can be expected. W.H. Press et al. announce a local convergence whereas V. Torczon [17] showed such a convergence cannot be guaranteed with the Nelder–Mead algorithm.

The major drawback of this algorithm is that the simplex can degenerate (a vertex becomes a linear combination of strictly less than the other n ones). If that happens, only a subspace of the complete working space can be visited and the risk of missing the minimum raises.

To decide whether or not to reinitialize the simplex can be based on the mean of the values at the n+1 vertices. The mean is compared with the mean at the previous temperature. If the relative change is not important enough or the generator stops at a local minimum, a new simplex is generated. The best point ever met is chosen as one of the vertices.

A natural way to initialize a simplex is to choose the n remaining vertices such that each edge issued from the (n+1)th point is parallel to a different axis of coordinates. A refined version of that approach is adopted. Instead of randomly choosing the value of the component in the interval of accepted values for that component, some *'taboo' restrictions* are added.

The overall working space is divided in regions. When a new simplex is generated, each cells containing a vertex are marked as taboo. The random selection of the value of a component is repeated until the resulting cell (C) does not lie in a taboo region (TL). Even if the best point ever met does not change between two successive re-initializations, this procedure guarantees that the two simplices are different. That raises the probability of visiting the overall space. Practically, the taboo cells are kept in a circular linked list and discarded when space for a new cell is required. The resulting pseudocode is given below:

```
DO
        use a simplex to get a new solution;
        IF      initialization required
        THEN    adopt the best solution as the (n+1)th
                vertex;
                for the first n vertices (V_i)
                DO
                    V_i = V_{n+1};
                    DO
                    change the ith component of V_i;
                    identify C;
                    WHILE (C in TL);
                    add C to TL;
                OD;
        FI;
        decrease the temperature;
WHILE (temperature > T_min);
```

Adopted pseudocode

Ingber's algorithm ([6,7]) is used for the annealing schedule. The initial temperature is set to $10^{\langle log_{10}(D)\rangle}$ where $\langle \log_{10}(D)\rangle$ stands for the mean of the logarithm of the objective function over the first generated simplex.

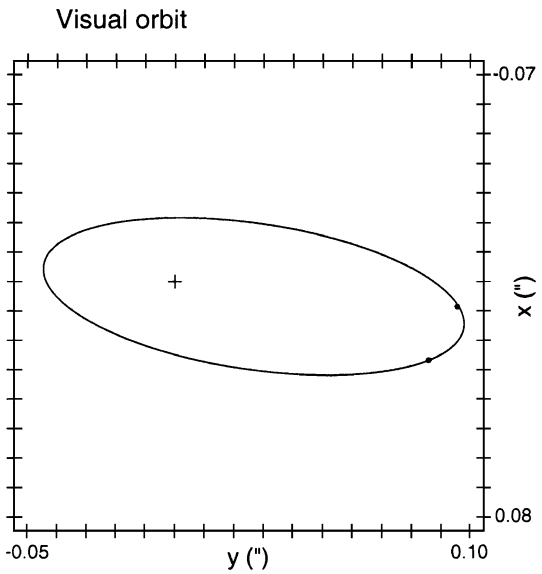
Element	Value	Std. dev.
$a(")$	0.072	0.0010
$i(°)$	68	1.3
$\omega(°)$	352	2.2
$\Omega(°)$	262.0	0.53
e	0.38	0.016
P(yr)	1.7255	0.00098
T (Besselian yr)	1979.332	0.0099
V_0(km/s)	−9.78	0.13
$\overline{\omega}(")$	0.038	0.0012
κ	0.349	0.0096
mass $A(M_\odot)$	1.5	0.18
mass $B(M_\odot)$	0.8	0.12

Orbital parameters of HIP111170 and their standard derivations

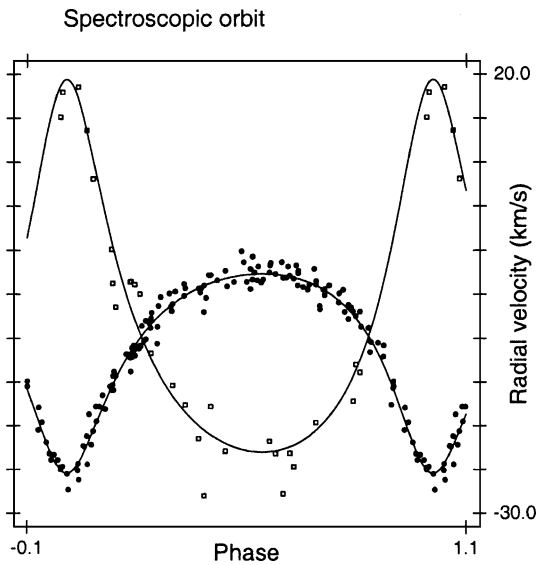
Example 1 (HIP111170)

The *double star* HIP111170 (= HR8851 = HD213429) is a good example to illustrate how appropriate a *simultaneous adjustment* is whereas a disjoint one would failed. The visual observations ([9,10]) are too few to allow a visual orbit determination: 3.5 observations (2 quantities) are necessary to adjust 7 parameters. Fortunately, the spectroscopic data are more numerous and the two radial velocity curves are well covered. From a mathematical point of view, two visual observations is the minimum if the spectroscopic observations [3] are well spread over the two curves.

The table above gives the orbital parameters used for the figures. The obtained *parallax* is in quite good agreement with the $0.03918 \pm 0.00183''$ after the *Hipparcos* mission [4].



Global Optimization in Binary Star Astronomy, Figure 1
Adjusted visual orbit of HIP 111170. The cross represents component *A*



Global Optimization in Binary Star Astronomy, Figure 2
Adjusted spectroscopic orbits of HIP 111170

Conclusion

Even when the observations seem very precise, the objective function describing the residual between the observed and computed data has many local minima. Astronomers should be aware of that fact as they should be aware of techniques to efficiently tackle such situations.

See also

- ▶ αBB Algorithm
- ▶ Continuous Global Optimization: Applications
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Differential Equations and Global Optimization
- ▶ DIRECT Global Optimization Algorithm
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization Using Space Filling
- ▶ Topology of Global Optimization

References

1. Carette E, de Greve JP, van Rensebergen W, Lampens P (1995) γ Circini: A young visual binary with pre-main-sequence component(s)? Astronomy and Astrophysics 296:139
2. Docobo JA, Elipe A, McAlister H (eds) (1997) Visual Double Stars: Formation, Dynamics and Evolutionary Tracks. Kluwer, Dordrecht
3. Duquennoy A, Mayor M, Griffin RF, Beavers WI, Eitter JJ (1988) Duplicity in the solar neighbourhood; V. Spectroscopic orbit of the nearby double-lined star HR 8581. Astronomy and Astrophysics Suppl Ser 75:167
4. ESA (1997) The Hipparcos and Tycho catalogues. ESA SP-1200
5. Hummel CA, Armstrong JT, Buscher DF, Mozurkewich D, Quirrenbach A, Vivekanand M (1995) Orbits of small angu-

lar scale binaries resolved with the Mark III interferometer. Astronomical J 110:376
6. Ingber L (1993) Adaptative simulated annealing (asa). Techn Report Caltech 1
7. Ingber L (1993) Simulated annealing: Practice versus theory. Res Note Caltech 1
8. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671
9. McAlister HA, Hartkopf WI, Franz OG (1990) ICCD speckle observations of binary stars. V Measurements during 1988-1989 from the Kitt Peak and the Cerro Tololo 4 m telescope. Astronomical J 99:965
10. McAlister HA, Hartkopf WI, Hutter DJ, Shara MM, Franz OG (1987) ICCD speckle observations of binary stars. I: A survey of duplicity among the bright stars. Astronomical J 93:183
11. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21(6):1087
12. Morbey CL (1975) A synthesis of the solutions of spectroscopic and visual binary orbits. Publ Astronomical Soc Pacific 87:689
13. Nelder JA, Mead R (1965) A simplex method for function minimization. Comput J 7:308
14. Pourbaix D (1994) A trial-and-error approach to the determination of the orbital parameters of visual binaries. Astronomy and Astrophysics 290:682
15. Pourbaix D, Lampens P (1997) A new method used to revisit the visual orbit of the spectroscopic triple system eta Orionis A. In: Docobo JA, Elipe A and McAlister H (eds) Visual Double Stars: Formation, Dynamics and Evolutionary Tracks. Kluwer, Dordrecht, p 383
16. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) Numerical recipes in C, 2nd edn. Cambridge Univ. Press, Cambridge
17. Torczon V (1991) On the convergence of the multidirectional search algorithm. SIAM J Optim 1(1):123
18. Torres G (1995) A visual-spectroscopic orbit for the binary Σ248. Publ Astronomical Soc Pacific 107:524

Global Optimization: Cutting Angle Method

ADIL BAGIROV[1], GLEB BELIAKOV[2]
[1] Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Victoria, Australia
[2] School of Engineering and Information Technology, Deakin University, Victoria, Australia

MSC2000: 90C26, 65K05, 90C56, 65K10

Article Outline

Introduction
Definitions
Notation
Abstract Convex Functions
IPH Functions
Lipschitz Functions
Methods
Generalized Cutting Plane Method
Global Minimization of IPH Functions over Unit Simplex
Global Minimization of Lipschitz Functions
The Auxiliary Problem
Solution of the Auxiliary Problem
Conclusions
References

Introduction

The cutting angle method (CAM) is a deterministic method for solving different classes of global optimization problems. It is a version of the generalized cutting plane method, and it works by building a sequence of tight underestimates of the objective function. The sequence of global minima of the underestimates converges to the global minimum of the objective function. It can also be seen from the perspective of branch-and-bound type methods, which iterate the steps of branching (partitioning the domain), bounding the objective function on the elements of the partition, and also fathoming (eliminating those elements of the partition which cannot contain the global minimum).

The key element of CAM is the construction of tight underestimates of the objective function and their efficient minimization in a structured optimization problem. CAM is based on the theory of abstract convexity [23], which provides the necessary tools for building accurate underestimates of various classes of functions. Such underestimates arise from a generalization of the following classical result: each convex function is the upper envelop of its affine minorants [21]. In abstract convex analysis, the requirement of linearity of the minorants is dropped, and abstract convex functions are represented as the upper envelops of some simple minorants, or support functions, which are not necessarily affine. Depending on the choice of the support functions, one obtains different flavours of abstract convex analysis.

By using a subset of support functions, one obtains an approximation of an abstract convex function from below. Such one-sided approximation, or underestimate, is very useful in optimization, as the global minimum of the underestimate provides a lower bound on the global minimum of the objective function. One can find the global minimum of the objective function as the limiting point of the sequence of global minima of the underestimates. This is the principle of the cutting angle method of global optimization [1,2,23].

The cutting angle method was first introduced for global minimization of increasing positive homogeneous (IPH) functions over the unit simplex [1,2,23]. Then it was extended to a broader class of Lipschitz programming problems [9,25]. In this Chapter, after providing the necessary theoretical background, we will describe versions of CAM for global minimization of IPH and Lipschitz functions over a polytope (in particular the unit simplex), and provide details of its algorithmic implementation.

Definitions

Notation

- n is the dimension of the optimization problem;
- $I = \{1, \ldots, n\}$;
- x_i is the ith coordinate of a vector $x \in \mathbb{R}^n$;
- $x^k \in \mathbb{R}^n$ denotes the k-th vector of some sequence $\{x^k\}_{k=1}^K$;
- $[l, x] = \sum_{i \in I} l_i x_i$ is the inner product of vectors l and x;
- if $x, y \in \mathbb{R}^n$ then $x \geq y \Leftrightarrow x_i \geq y_i$ for all $i \in I$;
- if $x, y \in \mathbb{R}^n$ then $x \gg y \Leftrightarrow x_i > y_i$ for all $i \in I$;
- $\mathbb{R}^n_+ := \{x = (x_1, \ldots, x_n) \in \mathbb{R}^n : x_i \geq 0$ for all $i \in I\}$ (nonnegative orthant);
- $\mathbb{R}_{+\infty}$ denotes $(-\infty, +\infty]$;
- $e^m = (0, \ldots, 0, 1, 0, \ldots, 0)$ denotes the m-th unit orth of the space $\mathbb{R}^n$.
- $S = \{x \in \mathbb{R}^n_+ : \sum_{i \in I} x_i = 1\}$ (unit simplex).

Abstract Convex Functions

Let $X \subset \mathbb{R}^n$ be some set, and let H be a nonempty set of functions $h\colon X \to V \subset [-\infty, +\infty]$. We have the following definitions [23].

Definition 1 A function f is abstract convex with respect to the set of functions H (or H-convex) if there exists $U \subset H$ such that:

$$f(x) = \sup\{h(x)\colon h \in U\}, \quad \forall x \in X\,.$$

Definition 2 The set U of H-minorants of f is called the support set of f with respect to the set of functions H:

$$\operatorname{supp}(f, H) = \{h \in H, h(x) \leq f(x) \quad \forall x \in X\}\,.$$

Definition 3 H-subgradient of f at x is a function $h \in H$ such that:

$$f(y) \geq h(y) - (h(x) - f(x)), \quad \forall y \in X\,.$$

The set of all H-subgradients of f at x is called H-subdifferential

$$\partial_H f(x) = \{h \in H\colon f(y) \geq h(y) - (h(x) - f(x)), \quad \forall y \in X\}\,.$$

Definition 4 The set $\partial^*_H f(x)$ at x is defined as

$$\partial^*_H f(x) = \{h \in \operatorname{supp}(f, H)\colon h(x) = f(x)\}\,.$$

Proposition 1 [23], *p.10. If the set H is closed under vertical shifts, i. e., $(h \in H, c \in R)$ implies $h - c \in H$, then $\partial^*_H f(x) = \partial_H f(x)$.*

When the set of support functions H consists of all affine functions, then we obtain the classical convexity. Next we examine two other examples of sets of support functions H.

IPH Functions

Recall that a function f defined on $\mathbb{R}^n_+$ is increasing if $x \geq y$ implies $f(x) \geq f(y)$.

Definition 5 A function $f\colon \mathbb{R}^n_+ \to \mathbb{R}$ is called IPH (Increasing Positively Homogeneous functions of degree one) if

$$\forall x, y \in \mathbb{R}^n_+, \quad x \geq y \Rightarrow f(x) \geq f(y);$$
$$\forall x \in \mathbb{R}^n_+, \forall \lambda > 0\colon f(\lambda x) = \lambda f(x)\,.$$

Let the set H_1 be the set of min-type functions

$$H_1 = \{h\colon h(x) = \min_{i \in I} a_i x_i, a \in \mathbb{R}^n_+, x \in \mathbb{R}^n_+\}\,.$$

Proposition 2 [23] *A function $f\colon \mathbb{R}^n_+ \to \mathbb{R}_{+\infty}$ is abstract convex with respect to H_1 if and only if f is IPH.*

Example 1 The following functions are IPH:

1) $f(x) = \sum_{i\in I} a_i x_i$ with $a_i \geq 0$;
2) $p_k(x) = \left(\sum_{i\in I} x_i^k\right)^{\frac{1}{k}} (k > 0)$;
3) $f(x) = \sqrt{[Ax, x]}$ where A is a matrix with non-negative entries;
4) $f(x) = \prod_{j\in J} x_j^{t_j}$ where $J \subset I,\ t_j > 0, \sum_{j\in J} t_j = 1$.

It is easy to check that

- the sum of two IPH functions is also an IPH function;
- if f is IPH, then the function γf is IPH for all $\gamma > 0$;
- let T be an arbitrary index set and $(f_t)_{t\in T}$ be a family of IPH functions. Then the function $f_{\inf}(x) = \inf_{t\in T} f_t(x)$ is IPH;
- let $(f_t)_{t\in T}$ be the same family and there exists a point $y \gg 0$ such that $\sup_{t\in T} f_t(y) < +\infty$ then the function $f_{\sup}(x) = \sup_{t\in T} f_t(x)$ is finite and IPH.

These properties allow us to give two more examples of IPH functions.

Example 2 The following maxmin functions are IPH:

1)
$$f(x) = \max_{k\in K} \min_{j\in J} \sum_{i\in I} a_i^{jk} x_i$$
where $a_i^{jk} \geq 0,\ k \in K, j \in J, i \in I$. Here J and K are finite sets of indices;

2)
$$f(x) = \max_{k\in K} \min_{j\in J_k} \sum_{i\in I} a_i^j x_i \tag{1}$$
where $a_i^j \geq 0, \quad j \in J_k, k \in K$. Here J_k and K are finite sets of indices.

Note that an arbitrary piecewise linear function f generated by a collection of linear functions $f^1, \ldots, f^m$ can be represented in the form (1) (see [5]); hence an arbitrary piecewise linear function generated by non-negative vectors is IPH.

Let $l \in \mathbb{R}^n_+, l \neq 0$ and $I(l) = \{i \in I : l_i > 0\}$. We consider the function $x \mapsto \langle l, x\rangle$ defined by the formula $l(x) = \langle l, x\rangle$ where the coupling function $\langle\cdot,\cdot\rangle$ is defined as

$$\langle l, x\rangle = \min_{i\in I(l)} l_i x_i\,. \tag{2}$$

Here $I(l) = \{i \in \{1, \ldots, n\} \mid l_i > 0\}$. This function is called a min-type function generated by the vector l. We shall denote this function by the same symbol $l(x)$. Clearly a min-type function is IPH. It follows from Proposition 2 that:

- A finite function f defined on $\mathbb{R}^n_+$ is IPH if and only if
$$f(x) = \max\{\langle l, x\rangle : l \in H_1, l \leq f\}; \tag{3}$$
- Let $x^0 \in \mathbb{R}^n_+$ be a vector such that $f(x^0) > 0$ and $l = f(x^0)/x^0$. Then
$$\langle l, x\rangle \leq f(x)$$
for all $x \in \mathbb{R}^n_+$ and $\langle l, x^0\rangle = f(x^0)$.

The vector $f(x^0)/x^0$ is called the support vector of a function f at a point x^0.

Lipschitz Functions

Definition 6 A function $f : X \to \mathbb{R}$ is called Lipschitz-continuous in X, if there exists a number $M > 0$ such that

$$\forall x, y \in X : |f(x) - f(y)| \leq M||x - y||\,.$$

The smallest such number is called the Lipschitz constant of f in the norm $||\cdot||$[1].

Let the set H_2 be the set of functions of the form

$$H_2 = \{h : h(x) = a - C||x - b||, \quad x, b \in \mathbb{R}^n, a \in \mathbb{R}, C \in \mathbb{R}_+\}\,.$$

Proposition 3 [23] *A function $f : \mathbb{R}^n \to \mathbb{R}_{+\infty}$ is H_2-convex if and only if f is a lower semicontinuous function. The H_2-subdifferential of f is not empty if f is Lipschitz.*

There is an interesting relation between IPH functions and Lipschitz functions, which allows one to formulate the problem of minimization of Lipschitz function over the unit simplex as the problem of minimization of IPH functions restricted to the unit simplex.

Theorem 1 *(see [23,25]). Let $f : S \to \mathbb{R}$ be a Lipschitz function and let*

$$M = \sup_{x,y\in S, x\neq y} \frac{|f(x) - f(y)|}{\|x - y\|_1} \tag{4}$$

[1] The norm $||\cdot||$ can be replaced by any metric, or, more generally, any distance function based on Minkowski gauge. For example, a polyhedral distance $d_P(x, y) = \max\{[(x - y), h_i] \mid 1 \leq i \leq m\}$, where $h_i \in \mathbb{R}^n, i = 1, \ldots, m$ is the set of vectors that define a finite polyhedron $P = \bigcap_{i=1}^m \{x \mid [x, h_i] \leq 1\}$.

be the least Lipschitz constant of f in $\|\cdot\|_1$-norm, where $\|x\|_1 = \sum_{i\in I}|x_i|$. Assume that

$$\min_{x\in S} f(x) \geq 2M\,.$$

Then there exists an IPH function $g\colon \mathbb{R}^n_+ \to \mathbb{R}$ such that $g(x) = f(x)$ for all $x \in S$.

Methods

We consider the problem of global minimization of an H-convex function f on a compact convex set $D \subset X$,

$$\text{minimize } f(x) \quad \text{subject to } x \in D\,. \tag{5}$$

We will deal with the two mentioned cases of f being H_1-convex (IPH) and H_2-convex (Lipschitz).

Generalized Cutting Plane Method

A consequence of Propositions 2 and 3 is that we can approximate H-convex functions from below using a finite subset of functions from $\mathrm{supp}(f, H)$. Suppose we know a number of values of the function f at the points x^k, $k = 1, \ldots, K$. Then the pointwise maximum of the support functions $h^k \in \partial^*_H f(x^K)$,

$$H^K(x) = \max_{k=1,\ldots,K} h^k(x) \tag{6}$$

is a lower approximation, or underestimate of f. We have the following generalization of the classical cutting plane method by Kelley [16].

$K_{\max}$ is the limit on the number of iterations of the algorithm. The problem at Step 2.1 is called the auxiliary, or relaxed, problem. Its efficient solution is the key to numerical performance of the algorithm. For convex objective functions, H^K is piecewise affine, and the solution to the relaxed problem is done by linear programming. However, when we consider other abstract convex functions, like IPH or Lipschitz, the relaxed problem is not linear, but it also has a special structure that leads to its efficient solution.

Step 0. (Initialisation)

0.1 Set $K = 1$.

0.2 Choose an arbitrary initial point $x^1 \in D$.

Step 1. (Calculate H-subdifferential)

1.1 Calculate $h^K \in \partial^*_H f(x^K)$.

1.2 Define $H^K(x) := \max_{k=1,\ldots,K} h^k(x)$, for all $x \in D$.

Step 2. (Minimize H^K)

2.1 Solve the Problem

$$\text{Minimize} \quad H^K(x) \quad \text{subject to} \quad x \in D.$$

Let x^* be its solution.

2.2 Set $K := K + 1$, $x^K := x^*$.

Step 3. (Stopping criterion)

3.1 If $K < K_{max}$ and $f_{best} - H^K(x^*) > \epsilon$ go to Step 1.

Global Optimization: Cutting Angle Method, Algorithm 1
Generalized Cutting Plane Algorithm

Global Minimization of IPH Functions over Unit Simplex

In this section we present an algorithm for the search for a global minimizer of an IPH function f over the unit simplex S, that is we shall study the following optimization problem:

$$\text{minimize } f(x) \quad \text{subject to } x \in S \tag{7}$$

where f is an IPH function defined on $\mathbb{R}^n_+$. Note that an IPH function is nonnegative on $\mathbb{R}^n_+$, since $f(x) \geq f(0) = 0$. We assume that $f(x) > 0$ for all $x \in S$. It follows from positiveness of f that $I(l) = I(x)$ for all $x \in S$ and $l(x) = f(x)/x$.

Since $I(e^m) = \{m\}$, then the vector $l = f(e^m)/e^m$ can be represented in the form $l = f(e^m)e^m$ and

$$\langle f(e^m)e^m, x\rangle = f(e^m)x_m\,.$$

Remark 1 Note that $H^K(x) := \max_{k=1,\ldots,K} \min_{i\in I(l^k)} l^k_i x_i \equiv \max\left\{H^{K-1}(x), \min_{i\in I(l^K)} l^K_i x_i\right\}$, which simplifies solution to the auxiliary problem at Step 2.1.

This Algorithm reduces the problem of global minimization (7) to the sequence of auxiliary problems. It provides lower and upper estimates of the global minimum $f_\star$ for the problem (7). Indeed, let $\lambda_K = \min_{x\in S} H^K(x)$ be the value of the auxiliary problem. It

follows from (3) that

$$\langle l^k, x\rangle \equiv \min_{i\in I(l^k)} l_i^k x_i \le f(x) \text{ for all } x \in S,$$
$$k = 1, \ldots, K\,.$$

Hence $H^K(x) \le f(x)$ for all $x \in S$ and $\lambda_K \equiv \min_{x\in S} H^K(x) \le \min_{x\in S} f(x)$. Thus λ_K is a lower estimate of the global minimum $f_\star$. Consider the number $\mu_K = \min_{k=1,\ldots,K} f(x^k) =: f_{best}$. Clearly $\mu_K \ge f_*$, so μ_K is an upper estimate of $f_\star$. It is shown in [23] that λ_K is an increasing sequence and $\mu_K - \lambda_K \to 0$ as $K \to +\infty$. Thus we have a stopping criterion, which enables us to obtain an approximate solution with an arbitrary given tolerance.

Global Minimization of Lipschitz Functions

Method Based on IPH Functions By using Theorem 1, global minimization of Lipschitz function over the simplex S can be reduced to the global minimization of a certain IPH function over S.

Let $f\colon S \to \mathbb{R}$ be a Lipschitz function and let

$$c \ge 2M - \min_{x\in S} f(x)\,, \tag{8}$$

where M is defined by (4). Let $f_1(x) = f(x) + c$. It follows from Theorem 1 that the function f_1 can be extended to an IPH function g. The problem

$$\text{minimize } g(x) \quad \text{subject to} \quad x \in S \tag{9}$$

is clearly equivalent to the problem

$$\text{minimize } f_1(x) \quad \text{subject to} \quad x \in S\,. \tag{10}$$

Thus we apply the cutting angle method to solve problem (10). Clearly functions f and f_1 have the same minimizers on the simplex S. If the constant c in (8) is known, CAM is applied for the minimization of a Lipschitz function f over S with no modification. If c is unknown, we can assume that c is a sufficiently large number, however numerical experiments show that CAM is rather sensitive to the choice of c, in particular, when c is very large, the method converges very slowly. In order to estimate c we need to know an upper bound on the least Lipschitz constant M and a lower estimate of the global minimum of f.

If the feasible domain is not the unit simplex S but a polytope, it can be embedded into S with a simple change of variables. Solution to the constrained auxiliary problem in Step 2.1 of the algorithm was investigated in [8].

Direct Method Consider H_2-convex functions, which, by Proposition 3 include all Lipschitz functions. Let d_P be a polyhedral distance function. As a consequence of H_2-convexity, we can approximate Lipschitz functions from below using underestimates of the form

$$\begin{aligned} H^K(x) &= \max_{k=1,\ldots,K} h^k(x) \\ &= \max_{k=1,\ldots,K} \left(f(x^k) - C d_P(x, x^k)\right), \end{aligned} \tag{11}$$

where $C \ge M$, and M is the Lipschitz constant of f with respect to the distance d_P. Then we apply the Algorithm 1 to function f in the feasible domain D. The auxiliary problem as Step 2.1 becomes

$$\text{minimize} \max_{k=1,\ldots,K} \left(f(x^k) - C d_P(x, x^k)\right)$$
$$\text{subject to } x \in D\,.$$

The same considerations about the convergence of the algorithm as those for Algorithm 2 are applied. Note

Step 0. (Initialisation)

0.1 Take points $x^m = e^m$, $m = 1, \ldots, n$. Set $K = n$.

0.2 Calculate $l^k = f(x^k)/x^k$, $k = 1, \ldots, K$.

Step 1. (Calculate H-subdifferential)

1.1 Define $H^K(x) := \max_{k=1,\ldots,K} \min_{i\in I(l^k)} l_i^k x_i$, for all $x \in S$.

Step 2. (Minimize H^K)

2.1 Solve the Problem

Minimize $H^K(x)$ subject to $x \in S$.

Let x^* be its solution.

2.2 Set $K := K + 1$, $x^K := x^*$.

2.3 Compute $l^K = f(x^K)/x^K$

Step 3. (Stopping criterion)

3.1 If $K < K_{max}$ and $f_{best} - H^K(x^*) > \epsilon$ go to Step 1.

Global Optimization: Cutting Angle Method, Algorithm 2
Cutting Angle Algorithm for IPH functions

that in the univariate case the underestimate H^K in (11) is exactly the same as the saw-tooth underestimate in Piyavski-Shubert method [20,26] if d_P is symmetric.

For minimization of Lipschitz functions, an estimate of the Lipschitz constant is required in both cases, when transforming f to an IPH function, or using Algorithm 1 directly. The crucial part in both methods is the efficient solution to the auxiliary problem in Step 2.1. The next section presents a very fast combinatorial algorithm for enumeration of all local minimizers of functions H^K.

The Auxiliary Problem

The *Step 2.1* (find the global minimum of $H^K(x)$) is the most difficult part of the cutting angle method. This problem is stated in the following form:

$$\text{minimize } H^K(x) \quad \text{subject to } x \in S \tag{12}$$

where

$$H^K(x) = \max_{k \le K} \min_{i \in I(l^k)} l_i^k x_i = \max_{k \le K} h^k(x)\,, \tag{13}$$

$K \ge n$, $l^k = f(x^k)/x^k$ are given vectors, $k = 1, \ldots, K$. Note that $x^k = e^k,\ k = 1, \ldots, n$.

Proposition 4 [2,3] *Let $K > n$, $l^k = l_k^k e^k,\ k = 1, \ldots, n, l^k > 0, |I(l^k)| \ge 2,\ k = n+1, \ldots, K$. Then each local minimizer of the function $H^K(x)$ defined by (13) over the simplex S is a strictly positive vector.*

Corollary 1 *Let $\{x^k\}$ be a sequence generated by Algorithm 2. Then $x^k \gg 0$ for all $k > n$.*

Let $\mathrm{ri}(S) = \{x \in S\colon x_i > 0 \text{ for all } i \in I\}$ be the relative interior of the simplex S. It follows from Proposition 4 and Corollary 1 that we can solve the problem (12) by sorting the local minima of the function H^K over the set $\mathrm{ri}\,(S)$. We now describe some properties of local minima of H^K on $\mathrm{ri}\,(S)$, which will allow us to identify these minima explicitly.

It is well known that functions h^k and H^K are directionally differentiable. Let $f'(x,u)$ denote directional derivative of the function f at the point x in the direction u. Also let

$$\begin{aligned} R(x) &= \{k\colon h^k(x) = H^K(x)\}\,,\\ Q_k(x) &= \{i \in I(l^k)\colon l_i^k x_i = h^k(x)\}\,. \end{aligned} \tag{14}$$

Proposition 5 *(see, for example, [13]). Let $x \gg 0$. Then*

$$(h^k)'(x,u) = \min_{i \in Q_k(x)} l_i^k u_i\,;$$

$$(H^K)'(x,u) = \max_{k \in R(x)} (h^k)'(x,u) = \max_{k \in R(x)} \min_{i \in Q_k(x)} l_i^k u_i\,.$$

Let $x \in S$. The cone

$$\begin{aligned} K(x,S) = \{u \in \mathbb{R}^n\colon\ &\exists \alpha_0 > 0\\ &\text{such that } x + \alpha u \in S\ \ \forall \alpha \in (0, \alpha_0)\} \end{aligned}$$

is called the tangent cone at the point x with respect to the simplex S. The following necessary conditions for a local minimum hold (see, for example, [13]). Suppose $x \in \mathrm{ri}(S)$. Then $K(x,S) = \{u\colon \sum_{i \in I} u_i = 0\}$.

Proposition 6 *Let $x \in S$ be a local minimizer of the function H^K over the set S. Then $(H^K)'(x,u) \ge 0$ for all $u \in K(x,S)$.*

Applying Propositions 5 and 6 we obtain the following result.

Proposition 7 [2,3] *Let $x \gg 0$ be a local minimizer of the function H^K over the set ri (S), such that $H^K(x) > 0$. Then there exists an ordered subset $\{l^{k_1}, l^{k_2}, \ldots, l^{k_n}\}$ of the set $\{l^1, \ldots, l^K\}$ such that*

1)

$$x = \left(\frac{d}{l_1^{k_1}}, \ldots, \frac{d}{l_n^{k_n}}\right) \text{ where } d = \frac{1}{\sum_{i \in I} \frac{1}{l_i^{k_i}}}\,; \tag{15}$$

2)

$$\max_{k \le K} \min_{i \in I(l^k)} \frac{l_i^k}{l_i^{k_i}} = 1; \tag{16}$$

3) Either $k_i = \{i\}$ for all $i \in I$ or there exists $m \in I$ such that $k_m \ge n+1$; if $k_m \le n$ then $k_m = m$;

4) if $k_m \ge n+1$ and $l_i^{k_m} \ne 0$ then $l_i^{k_m} > l_i^{k_i}$ for all $i \in I, i \ne m$.

Solution of the Auxiliary Problem

It follows from Propositions 4 and 7 that we can find a global minimizer of the function H^K defined by (13) over the unit simplex using the following procedure:

- sort all subsets $\{l^{k_1},\dots,l^{k_n}\}$ of the given set $l^1,\dots,l^K$ vectors, such that (16) holds and $l_i^{k_m} > l_i^{k_i},\ i \neq m$ if $k_m \geq n+1, i \in I(l^{k_m})$ and $k_m = m$ if $k_m \leq n$;
- for each such subset, find the vector x defined by (15);
- choose the vector with the least value of the function H^K among all the vectors described above.

Thus, the search for a global minimizer is reduced to sorting some subsets, containing n elements of the given set $\{l^1,\dots l^K\}$ with $K > n$. Fortunately, Proposition 7 allows one to substantially diminish the number of sorted subsets.

The subsets $L = \{l^{k_1},\dots,l^{k_n}\}$ can be visualized with the help of an $n \times n$ matrix whose rows are given by the participating support vectors

$$L = \begin{pmatrix} l_1^{k_1} & l_2^{k_1} & \dots & l_n^{k_1} \\ l_1^{k_2} & l_2^{k_2} & \dots & l_n^{k_2} \\ \vdots & \vdots & \ddots & \vdots \\ l_1^{k_n} & l_2^{k_n} & \dots & l_n^{k_n} \end{pmatrix}. \tag{17}$$

The conditions 2) and 4) of Proposition 7 are then easily interpreted as follows. Condition 4) implies that the diagonal elements of matrix L are smaller than elements in their respective columns, and condition 2) implies that the diagonal of L is not dominated by any other support vector $l^k \notin L$ (zero entries of matrix L are excluded from compaisons). Thus we obtain a combinatorial problem of enumerating all combinations L that satisfy conditions 2) and 4).

However it is impractical to enumerate all such combinations directly for large K. Fortunately there is no need to do so. It was shown in [6,7,8] that the required combinations can be put into a tree structure. The leaves of the tree correspond to the local minimizers of H^K, whereas the intermediate nodes correspond to the minimizers of $H^n, H^{n+1},\dots,H^{K-1}$.The incremental algorithm based on the tree structure makes computations very efficient numerically (as processing of queries using trees requires logarithmic time of the number of nodes). It is possible to enumerate several billions of local minimizers of H^K (e. g., when $n = 5$ and $K = 100,000$) in a matter of seconds on a standard Pentium IV based workstation.

The direct method of minimization of Lipschitz functions involves solution to a different auxiliary problem, that of minimizing H^K given in (11), with d_P being a simplicial distance function. It turns out that a very similar method of enumeration of local minimizers of H^K, by putting them in a tree structure, also works [9]. There is a counterpart of Proposition 7, with the difference that the support vectors are defined by

$$l_i^k = \frac{f(x^k)}{C} - x_i^k, \tag{18}$$

and the local minima and minimizers of H^K are identified through

$$\begin{aligned} d &= H^K(x^*) = \frac{C(Trace(L)+1)}{n}, \\ x_i^* &= \frac{d}{C} - l_i^{k_i}, i = 1, \dots, n, \end{aligned} \tag{19}$$

where constant C is chosen greater or equal to the Lipschitz constant M of f in the simplicial distance d_P. Thus both versions of CAM, for IPH and for Lipschitz functions, share the same algorithm, but with different definitions of support vectors.

The actual algorithms for enumeration of local minima of H^K and maintaining the tree structure, as well as treatment of linear constraints, are presented in [7,8,9]. The algorithms involve a crucial fathoming step, and can be seen as branch-and-bound type algorithms [9,12,23].

Conclusions

Cutting angle methods are versions of the generalized cutting plane method for IPH, Lipschitz and other classes of abstract convex functions. The main idea of this deterministic method is to replace the original problem of minimizing f with a sequence of relaxed problems with special structure. The objective functions in the relaxed problems provides tight lower estimates of f, and the sequence of their solutions converge to the global minimum of f. Efficient solution to the relaxed problem makes CAM very fast on a class of global optimization problems.

Optimization is not the only field such underestimates are applied. Versions of CAM are also used for non-uniform random variate generation [10] and multivariate data interpolation [11].

Both versions of CAM described here have been successfully applied to a number of real life problems,

including very difficult molecular geometry prediction and protein folding problems [12,17]. A software library GANSO for global and non-smooth optimization, which includes the cutting angle method, is available from http://www.ganso.com.au.

References

1. Andramonov MY, Rubinov AM, Glover BM (1999) Cutting angle method in global optimization. Appl Math Lett 12:95–100
2. Bagirov AM, Rubinov AM (2000) Global minimization of increasing positively homogeneous functions over the unit simplex. Ann Oper Res 98:171–187
3. Bagirov AM, Rubinov AM (2001) Modified versions of the cutting angle method. In: Hadjisavvas N, Pardalos PM (eds) Advances in Convex Analysis and Global Optimization. Kluwer, Dordrecht, pp 245–268
4. Bagirov AM, Rubinov AM (2003) The cutting angle method and a local search. J Global Optim 27:193–213
5. Bartels SG, Kuntz L, Sholtes S (1995) Continuous selections of linear functions and nonsmooth critical point theory. Nonlinear Anal TMA 24:385–407
6. Batten LM, Beliakov G (2002) Fast algorithm for the cutting angle method of global optimization. J Global Optim 24:149–161
7. Beliakov G (2003) Geometry and combinatorics of the cutting angle method. Optimization 52:379–394
8. Beliakov G (2004) The Cutting Angle Method – a tool for constrained global optimization. Optim Methods Softw 19:137–151
9. Beliakov G (2005) A review of applications of the Cutting Angle methods. In: Rubinov A, Jeyakumar V (eds) Continuous Optimization. Springer, New York, pp 209–248
10. Beliakov G (2005) Universal nonuniform random vector generator based on acceptance-rejection. ACM Trans Modelling Comp Simulation 15:205–232
11. Beliakov G (2006) Interpolation of Lipschitz functions. J Comp Appl Math 196:20–44
12. Beliakov G, Lim KF (2007) Challenges of continuous global optimization in molecular structure prediciton. Eur J Oper Res 181(3):1198–1213
13. Demyanov VF, Rubinov AM (1995) Constructive Nonsmooth Analysis. Peter Lang, Frankfurt am Main
14. Horst R, Pardalos PM, Thoai NV (1995) Introduction to Global Optimization. Kluwer, Dordrecht
15. Horst R, Tuy H (1996) Global Optimization: Deterministic Approaches, 3rd edn. Springer, Berlin
16. Kelley JE (1960) The cutting-plane method for solving convex programs. J SIAM 8:703–712
17. Lim KF, Beliakov G, Batten LM (2003) Predicting molecular structures: Application of the cutting angle method. Phys Chem Chem Phys 5:3884–3890
18. Pallaschke D, Rolewicz S (1997) Foundations of Mathematical Optimization (Convex Analysis without Linearity). Kluwer, Dordrecht
19. Pinter JD (1996) Global Optimization in Action. Continuous and Lipschitz Optimization: Algorithms, Implementation and Applications. Kluwer, Dordrecht
20. Piyavskii SA (1972) An algorithm for finding the absolute extremum of a function. USSR Comp Math Math Phys 12:57–67
21. Rockafellar RT (1970) Convex Analysis. Princeton University Press, Princeton
22. Rolewicz S (1999) Convex analysis without linearity. Control Cybernetics 23:247–256
23. Rubinov AM (2000) Abstract Convexity and Global Optimization. Kluwer, Dordrecht
24. Rubinov AM, Andramonov MY (1999) Minimizing increasing star-shaped functions based on abstract convexity. J Global Optim 15:19–39
25. Rubinov AM, Andramonov MY (1999) Lipschitz programming via increasing convex-along-rays fuinctions. Optim Methods Softw 10:763–781
26. Shubert BO (1972) A sequential method seeking the global maximum of a function. SIAM J Numerical Anal 9:379–388
27. Singer I (1997) Abstract Convex Analysis. Wiley-Interscience Publication, New York

Global Optimization: Envelope Representation

A. M. Rubinov
School Inform. Techn. and Math. Sci.,
University Ballarat, Ballarat, Australia

MSC2000: 90C26

Article Outline

Keywords
See also
References

Keywords

Abstract convexity; Envelope; -subdifferential; Support set; Supremal generator; Min-type function; Cutting angle method; Global optimization; Lipschitz programming

Some classical methods of finite-dimensional convex minimization can be extended for quite broad classes of multi-extremal optimization problems. One successful

generalization is based on the so-called *envelope representation* of the objective function.

We begin with the simplest case of a convex differentiable function f in order to introduce this approach. For such a function the *tangent hyperplane* $T = \{x\nabla f(y)(x-y)+f(y)=0\}$ is simultaneously a *support hyperplane*. That is, the inequality $f(x) \geq f(y)+\nabla f(y)(x-y)$ holds for each x. This inequality can be expressed also in the following form: the affine function

$$h_y(x) = \nabla f(y)(x-y) + f(y) \tag{1}$$

is a support function for the function f. Thus the function f can be represented as the pointwise maximum of the functions of the form h_y:

$$f(x) = \max_y h_y(x).$$

One of the main results of convex analysis asserts that an arbitrary lower semicontinuous convex function f (perhaps admitting the value $+\infty$) is the *upper envelope* (UE) of the set of all its affine minorants:

$$f(x) = \sup\left\{h(x): \begin{array}{c} h \text{ is an affine function,} \\ h \leq f \end{array}\right\}.$$

(The inequality $h \leq f$ stands for $h(x) \leq f(x)$ for all x.) The supremum above is attained if and only if the *subdifferential* of f at the point x is nonempty. Since affine functions are defined by means of linear functions, one can say that convexity is'linearity + envelope representation'.

As it turns out the contribution of'envelope representation' to the convexity is fairly large. This observation stimulated the development of the rich theory of'convexity without linearity'. (See [12,14,19] and references therein.) In particular, functions which can be represented as UE of subsets of a set of sufficiently simple functions are studied in this theory.

We need the following definition. Let H be a set of functions. A function f is called *abstract convex* (AC) with respect to H (or H-*convex*) if f is the UE of a subset from H, that is

$$f(x) = \sup\{h(x): \ h \in H, h \leq f\}. \tag{2}$$

The set H is called the *set of elementary functions*. For applications we need sufficiently simple elementary functions.

Many results from convex analysis related to various kinds of convex duality can be extended to *abstract convex analysis* Abstract convexity sheds some new lights to the classical *Fenchel–Moreau duality* and the so-called *level sets conjugation* (see [19]). The set $s(f, H) = \{h \in H: h \leq f\}$, presented in (2), is called the *support set* of f. The mapping $f \longmapsto s(f, H)$ is called the *Minkowski duality* ([9]). The support set accumulates a global information of a function f in terms of the set of elementary functions H and it can be useful in the study of global optimization problems involving the function f.

One of the main notions of convex analysis, which plays the key role for applications to optimization, is the subdifferential. There are two equivalent definitions of the subdifferential of a convex function. The first of them is based on the global behavior of the function. A linear function l is called a *subgradient* (i. e. a member of the subdifferential) of the function f at a point y if the affine function $h(x) = l(x) - (l(y) - f(y))$ is a *support function* with respect to f, that is $h(x) \leq f(x)$ for all x. The second definition has a local nature and is connected with local approximation of the function: the subdifferential is a closed convex set of linear functions such that the *directional derivative* $u \longmapsto f'_x(u)$ at the point x is presented as the UE of this set. For a differentiable convex function these two definitions reflect respectively support and tangent sides of the gradient.

The various generalizations of the second definition have led to development of the rich theory of *nonsmooth analysis*. The natural field for generalizations of the first definition is AC.

A function $h \in H$ is called the *subgradient* (or H-*subgradient*) of an H-convex function f at a point y if $f(x) \geq h(x) - (h(y) - f(y))$ for all x. The set $\partial_H f(y)$ of all subgradients of f at y is referred to as the *subdifferential* of the function f at the point y.

Let H' be the closure of the set H under vertical shifts, that is

$$H' = \left\{h': \begin{array}{c} h'(x) = h(x) - c, \\ h \in H, \ c \in \mathbf{R} \end{array}\right\}.$$

Clearly $h \in \partial_{H'} f(y)$ if and only if $f(y) = \max\{h'(y): h' \leq f, h' \in H'\}$. Thus if H is already closed under shifts then

$$\partial_H f(y) = \{h \in s(f, H): \ h(y) = f(y)\}. \tag{3}$$

Thus the subdifferential is not empty if and only if the supremum in (2) is attained.

Sometimes (3) is used for the definition of the subdifferential for an arbitrary set of elementary functions H (not necessary closed under shifts).

Many methods of convex minimization are based on the local properties of the convex subdifferential (more precisely, on the directional derivative). However there are some methods which exploit only the support property of the subdifferential. The conceptual schemes of these methods can be easily extended for AC functions. One of these methods is presented below.

Consider the following problem

$$f(x) \to \min, \quad x \in X, \tag{4}$$

where X is a compact set. Assume that f is AC with respect to a set of elementary functions H. We consider the following algorithm based on the *generalized cutting plane* idea, which is a nonlinear generalization of the classical cutting plane method.

0 Let $k := 0$. Choose an arbitrary initial point $x_0 \in X$;

1 Calculate a subgradient in the form (3) that is an element $h_k \in s(f, H)$ such that $h_k(x_k) = f(x_k)$;

2 Find a global optimum y^* of the problem

$$\max_{0 \le i \le k} h_i(x) \to \min, \quad x \in X. \tag{5}$$

3 Let $x_{k+1} = y^*$, $k := k + 1$. Go to step 1.

Conceptual scheme (generalized cutting plane method)

Convergence of the sequence constructed by this procedure to a global minimizer has been proved under very mild assumptions by D. Pallaschke and S. Rolewicz [12]. Upper and lower estimates of the optimal value of the problem (4) can be computed, which lead to an efficient stopping criterion (compare with [2]).

There are two major difficulties in the numerical implementation of the Algorithm. The first is the calculation of a subgradient. In general it is very difficult to find it numerically, however it is possible in several important particular cases. The second difficulty is the solution of the auxiliary problem (5). This is a linear programming problem in the case of the set H of affine functions, but for sets of more complicated functions the problem (5) is essentially of a combinatorial nature or a problem of convex maximization.

The simplest example of this approach is *Lipschitz programming*. If f is a Lipschitz function we can, for example, take as H the set of functions h of the form $h(x) = -a\|x - x_o\| - c$, where a is a positive and c is a real number, $x_o \in X$. In order to find an H-subgradient we should take $a > L$ where L is the Lipschitz constant of the function f; thus we need to know an upper estimate of this constant; this is a special piece of global information about this function. With such H the problem (4) can be reduced to a sequence of special problems of concave minimization. Some known algorithms of Lipschitz programming fall within the described approach [11,21].

For fairly large classes of functions defined on the cone $\mathbf{R}^n_+$ of all n-vectors with nonnegative coordinates it is possible to take as H a set of functions which includes as its main part a *min-type function* of the form

$$l(x) = \min_{i \in \mathcal{T}(l)} l_i x_i, \quad x \in \mathbf{R}^n_+, \quad \text{with } \mathcal{T}(l) = \{i:\ l_i > 0\}. \tag{6}$$

We define the infimum over empty set to be zero. If l is a strictly positive vector and c a positive number then the set $\{x: \min_i l_i x_i \le c\}$ is a complement to a 'right angle'. Exploiting min-type functions instead of linear functions allows us to separate a point from the (not necessary convex) set by the complements of 'right angles'.

Various classes of elementary functions arise, based on the set L of all functions of the form (6) with $l \in \mathbf{R}^n_+$. In particular, L itself and sets

$$H_1 = \{h:\ h(x) = l(x) - c,\ l \in L,\ c \in \mathbf{R}\},$$
$$H_2 = \{h:\ h(x) = \min(l(x),\ c),\ l \in L,\ c \in \mathbf{R}\}$$

are convenient for applications. The classes of AC with respect to H_1 and H_2 functions are quite large [14]. The first of them consists of all increasing (with respect to the usual order relation) functions f such that the function of a real variable $t \to f(tx)$, $t \in [0, +\infty)$, is convex for all $x \in \mathbf{R}^n_+$. This class contains all homogeneous functions of degree $\delta \ge 1$, their sums and UE of sets of

such functions. In particular it contains all polynomials with nonnegative coefficients. The second class consists of all increasing functions f such that $f(tx) \geq tf(x)$ for all $x \in \mathbf{R}^n_+$ and $t \in [0, 1]$. Concave increasing functions f with $f(0) \geq 0$ and UE of sets of such functions belong to this class. Also, positively homogeneous functions of degree $\delta \leq 1$, their sums and UE of sets of such functions belong to it.

For minimizing AC functions with respect to H_i ($i = 1, 2$) we need again to calculate the H_i-subgradients in the form (5) and then to reduce the problem (4) to a sequence of auxiliary problems. A version of the generalized cutting plane method in such a case is called *'cutting angle method'* ([2,14]).

A.M. Rubinov et al. ([1,14,16,17]) have demonstrated that for AC functions generated by various classes of min-type functions it is possible to find subgradients very easily. In particular, only the number $f(x)$ (resp. $f'(x, x)$) is required for the calculation of an element of $\partial_{H_2} f(x)$ (resp. $\partial_{H_1} f(x)$), without any additional information about a global behavior of the function f. Thus the main problem with implementation of the cutting angle method is to solve the auxiliary subproblem, which is a problem of the mixed integer programming of a special kind in this case.

Let L be the set of all functions (6) with $l \in \mathbf{R}^n_+$. It can be shown ([14,16]) that a function f defined on $\mathbf{R}^n_+$ is L-convex if and only if f is IPH (increasing and positively homogeneous of degree one).*IPH functions* can serve for the miminization of a Lipschitz function over the unit simplex $S_n = \{x \in \mathbf{R}^n_+ : \sum_i x_i = 1\}$. First ([14,15]), for each Lipschitz function g defined on S_n there exists a constant $c>0$ such that the function $\widetilde{g}(x) = g(x) + c$ can be extended to an IPH function defined on $\mathbf{R}^n_+$. Second, the auxiliary problem (5) for problem (4) with an IPH function f and $X = S_n$, has a special structure and can be efficiently solved for fairly large n (see [14, Chap. 9] and references therein). Thus, the minimization of a Lipschitz function over the unit simplex can be efficiently accomplished by the cutting angle method.

Numerical experiments demonstrate that a combination of the cutting angle method with a local search is very efficient, since the cutting angle method allows one to leave a local minimizer fairly quickly.

Envelope representation is useful also in the study of some theoretical problems arising in optimization. Many interesting examples of such applications can be found in the books [12,14,19]. In particular, a general scheme of penalty and augmented Lagrangian based on the notion of the subdifferential is presented in [12]. I. Singer [19] demonstrated that Fenchel–Moreau duality leads to a unified theory of duality results for very general optimization problems. It can be shown [18] that AC forms the natural framework for the study of *solvability theorems* (generalizations of Farkas' lemma; cf. ▶ Farkas lemma; ▶ Farkas lemma: Generalizations). In contrast with numerical methods based on applications of subdifferentials, the study of solvability theorems is based on application of support sets. AC serves also for the study of some problems of quasiconvex minimization (see for example [10,13,20]).

A subset H of a set X of functions is called the *supremal generator* ([9]) of X if each function from X is AC with respect to H. There exist very small supremal generators of very large classes of functions. The following two examples of such supremal generators are useful for nonsmooth optimization.

1) Recall that a function f is called *positively homogeneous* (PH) of degree k if $p(\lambda x) = \lambda^k p(x)$ for $\lambda > 0$. It can be shown ([14]) that the set of all functions of the form

$$h(x) = -a\left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}} + \sum_{i=1}^{n} l_i x_i, \tag{7}$$

where $a \leq 0$, $l_1, \ldots, l_n$ are real numbers is a supremal generator of the set PH_1 of all lower semicontinuous PH functions of degree one defined on n-dimensional space $\mathbf{R}^n$. Since each function (7) is concave it follows that the set of all concave PH functions of degree one is a supremal generator of PH_1.

2) It can be shown ([3,4,9,14]) that the set $\mathcal{H}$ of all quadratic functions h of the form

$$h(x) = -a\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} l_i x_i + c, \tag{8}$$

where $a \geq 0$, $l_1, \ldots, l_n$, c are real numbers is a supremal generator of the set of all lower semicontinuous functions $f : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ minored by $\mathcal{H}$ in the following sense: there exists $h \in \mathcal{H}$ such that $f \geq h$.

Supremal generators are a convenient tool in the study of nonsmooth optimization problems. A local approximation of the first (resp. second) order of a nonsmooth

function is fulfilled very often by various kinds of generalized derivatives of the first (resp. second) order, which are PH functions of the first (resp. second) degree. Practical applications of these derivatives to optimization are based on their representation in terms of linear (resp. quadratic) functions.

Linearization of lower semicontinuous PH functions of the first degree can be accomplished by supremal generators of the space PH_1, consisting of concave functions. Each finite concave function $g \in \mathrm{PH}_1$ can be presented as $\min\left\{l(x):\ l \in \overline{\partial} g(0)\right\}$ where $\overline{\partial} g(0)$ is the superdifferential (in the sense of convex analysis) of this function g at the origin. Hence each function $g \in \mathrm{PH}_1$ can be linearized by the operation sup min.

The second order approximation of a nonsmooth function f at a point x can be accomplished by the *subjet*, that is the set

$$\partial^{2,-} f(x) = \left\{ (\nabla g(x), \nabla^2 g(x)): \begin{array}{c} f - g \text{ has a} \\ \text{local minimum } x \\ \text{with } g \in C^2(\mathbf{R}^n) \end{array} \right\}.$$

(Here $\nabla\ g(x)$ (resp. $\nabla^2\ g(x)$) stands for the gradient (resp. Hessian) of a function g at a point x.) Let $\mathcal{H}$ be the set of all functions of the form (8). It can be shown (see [5,6]) that the subjet $\partial^{2,-} f(x)$ is nonempty if and only if the $\mathcal{H}$-subdifferential $\partial_{\mathcal{H}} f(x)$ is not empty. AC with respect to $\mathcal{H}$ can also serve for supremal representation of the second order generalized derivatives of nonsmooth functions in terms of quadratic functions (see[5]).

See also

- ▶ Dini and Hadamard Derivatives in Optimization
- ▶ Nondifferentiable Optimization
- ▶ Nondifferentiable Optimization: Cutting Plane Methods
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Nondifferentiable Optimization: Newton Method
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Nondifferentiable Optimization: Relaxation Methods
- ▶ Nondifferentiable Optimization: Subgradient Optimization Methods

References

1. Abasov TM, Rubinov AM (1994) On the class of H-convex functions. Russian Acad Sci Dokl Math 48:95–97
2. Andramonov MYu, Rubinov AM, Glover BM (1999) Cutting angle methods in global optimization. Applied Math Lett 12:95–100
3. Balder EJ (1977) An extension of duality-stability relations to nonconvex optimization problems. SIAM J Control Optim 15:329–343
4. Dolecki S, Kurcyusz S (1978) On Φ-convexity in extremal problems. SIAM J Control Optim 16:277–300
5. Eberhard A, Nyblom M (1998) Jets, generalized convexity, proximal normality and differences of functions. Nonlinear Anal (TMA) 34:319–360
6. Eberhard A, Nyblom N, Ralph D (1998) Applying generalized convexity notions to jets. In: Croizeix J-P, Martinez-Legaz J-E, Volle M (eds) Generalized Convexity, Generalized Monotonicity. Kluwer, Dordrecht, pp 111–158
7. Horst R, Pardalos PM (eds) (1996) Handbook of global optimization. Kluwer, Dordrecht
8. Kelley J (1960) The cutting plane method for solving convex programs. SIAM J 8:703–712
9. Kutateladze SS, Rubinov AM (1972) Minkowski duality and its applications. Russian Math Surveys 27:137–191
10. Martinez-Legaz J-E (1988) Quasiconvex duality theory by generalized conjugation methods. Optim 19:603–652
11. Mladineo RH (1986) An algorithm for finding the global maximum of a multimodal, multivariate function. Math Program 34:188–200
12. Pallaschke D, Rolewicz S (1997) Foundations of mathematical optimization (convex analysis without linearity). Kluwer, Dordrecht
13. Penot JP, Volle M (1990) On quasiconvex duality. Math Oper Res 15:597–625
14. Rubinov AM (2000) Abstract convexity and global optimization. Kluwer, Dordrecht
15. Rubinov AM, Andramonov MYu (1999) Lipschitz programming via increasing convex-along-rays functions. Optim Methods Softw 10:763–781
16. Rubinov AM, Andramonov MYu (1999) Minimizing increasing star-shaped functions based on abstract convexity. J Global Optim 15:19–39
17. Rubinov AM, Glover BM (1999) Increasing convex-along-rays functions with applications to global optimization. J Optim Th Appl 102(3)
18. Rubinov AM, Glover BM, Jeyakumar V (1995) A general approach to dual characterization of solvability of inequality systems with applications. J Convex Anal 2:309–344
19. Singer I (1997) Abstract convex analysis. Wiley/Interscience, New York
20. Volle M (1985) Conjugasion par tranches. Ann Mat Pura Appl 139:279–312
21. Wood GR (1992) The bisection method in higher dimensions. Math Program 55:319–337

Global Optimization: Filled Function Methods

HONG-XUAN HUANG
Department of Industrial Engineering, Tsinghua University, Beijing, People's Republic of China

MSC2000: 90C26, 90C30, 90C59, 65K05

Article Outline

Keywords and Phrases
Introduction
Definitions
Methods
Two-Parameter Filled Functions
Single-Parameter Filled Functions
Nonsmooth Filled Functions
Discrete Filled Functions
Summary
References

Keywords and Phrases

Basin; Filled function; Filled function method

Introduction

The *filled function methods* describe a class of global optimization methods for attacking the problem of finding a global minimizer of a function $f: X \to \Re$ over a certain subset $X \subset \Re^n$. Each variant of such methods replaces the objective function $f(x)$ by a specific auxiliary function that is associated with a local minimum and some parameters in every iteration, and is minimized through some local search strategies. The term "filled function" means that every auxiliary function can fill the region of attraction at a certain neighborhood of a local minimum of the objective function.

The definition of a filled function involves some basic concepts. The term "basin" was introduced first in [1]. A *basin* of a function $f(x)$ at an isolated minimizer x_1^* denotes a connected domain B_1^* which contains x_1^* and in which starting from any point the steepest descent trajectory of $f(x)$ converges to x_1^*, but outside of which the steepest descent trajectory of $f(x)$ does not converge to x_1^*. Accordingly, a *hill* of a function $f(x)$ at a maximizer x_1^* is a basin of $-f(x)$ at the point x_1^*.

In addition, the basin B_2^* at a minimizer x_2^* is *lower* (or *higher*) than the basin B_1^* at another minimizer x_1^* if the following inequality holds:

$$f(x_2^*) < f(x_1^*) \text{ (or } f(x_2^*) \geq f(x_1^*)) .$$

Definitions

The first kind of filled function method was proposed in [5] for the unconstrained optimization problem

$$\min_{x \in \Re^n} f(x) .$$

The corresponding filled function involved two parameters, and was defined by

$$P(x, x_1^*, r, \rho) = \frac{1}{r + f(x)} \exp\left(-\frac{\|x - x_1^*\|^2}{\rho^2}\right) , \quad (1)$$

where x_1^* is a minimizer of the objective function $f(x)$, and r and ρ are parameters such that $r + f(x_1^*) > 0$, $\rho > 0$. In order to demonstrate the principle of the filled function method, people usually assume that the function $f(x)$ is twice continuously differentiable and coercive, i. e., its Hessian is continuous and the following condition holds:

$$\lim_{\|x\| \to +\infty} f(x) = +\infty . \quad (2)$$

It is also assumed that the function $f(x)$ has only a finite number of minimizers in a closed domain $\Omega \subset \Re^n$ that contains all global minimizers of $f(x)$.

Under certain other conditions concerning the parameters r and ρ, the function $P(x, x_1^*, r, \rho)$ defined in (1) has three properties as follows:

(a) x_1^* is a maximizer of $P(x, x_1^*, r, \rho)$ and the whole basin B_1^* at x_1^* becomes a part of a hill of $P(x, x_1^*, r, \rho)$ at x_1^*.
(b) $P(x, x_1^*, r, \rho)$ has no minimizers or saddle points in any higher basin of $f(x)$ than B_1^* at x_1^*.
(c) $f(x)$ has a lower basin B than B_1^* at x_1^*, then there is a point x' in such a basin B that minimizes $P(x, x_1^*, r, \rho)$ on the line through x' and x_1^*.

A function satisfying the above three properties is said to be a *filled function* of $f(x)$ at the local minimizer x_1^*. Note that the above definition just lists the main properties required for a filled function, in which the number of parameters is not an important factor (see the discussion about categories of filled functions below).

Usually, when people develop a variant of the filled function method, property c in the above definition may be replaced by a similar one. For example, it was replaced in [21] by

(c_1) If $f(x)$ has a basin B_2^* at x_2^* that is lower than B_1^*, then there is a point $x' \in B_2^*$ that minimizes $P(x, x_1^*, r, \rho)$ on the line through x_1^* and x'', for every x'' in some neighborhoods of x_2^*.

Note that property c_1 is much stronger than that required in [5] since a minimizer is required for lines connecting the current minimizer with every point in some neighborhoods of a next better minimizer.

In addition, for the unconstrained global optimization problem, in [16] two classes of continuously differentiable filled functions with multiplicative and additive structures, respectively, were proposed which assumed the existence of a local minimizer in a lower basin but not just on lines.

Under such assumptions as the objective function $f: \Re^n \to \Re$ is coercive, continuously differentiable and has finite local minimizers, another stronger variant of the filled functions can be found in [18], where the concept of a basin at a local minimizer was extended to that of a *G-basin*. A subset $B^* \subset \Re^n$ is said to be a *G-basin* of $f(x)$ corresponding to a local minimizer x^* if it is a connected domain with the following properties:

(i) $f(x) \geq f(x^*)$ for any $x \in B^*$;
(ii) $\bar{x} \in B^*$ is a local minimizer of $f(x)$ if and only if $f(\bar{x}) = f(x^*)$.

The definition in [18] requires that a filled function $p(x)$ is differentiable and satisfies some modifications of conditions a and b as follows:

(a′) *x_1^* is a strictly local maximizer of $p(x)$.*
(b′) *For any $x \neq x^*$ satisfying $f(x) \geq f(x^*)$, x is not a stationary point of $p(x)$.*

Furthermore, any lower local minimizer $\bar{x}$ of $f(x)$ than a nonglobal minimizer x^* is also a local minimizer of the filled function and is lower than every point on the boundary of the box set Ω which contains all global minimizers of $f(x)$. For points higher than x^* in Ω, the farther they are from x^* implies a lower value of the filled function.

Recently, in order to take advantages of filled functions and reduce the difficulty in adjusting the value of parameters, the concept a locally filled function was introduced in [9,22], which was based on the concept of a local basin.

Given a bounded and closed convex set $\omega \subset \Omega$ and a basin B_1 of the objective function $f(x)$ at a local minimizer x_1^*, if the set

$$B_1(\omega) := \omega \cap B_1 \neq \emptyset \,,$$

then $B_1(\omega)$ is called a *local basin* associated with x_1^* and ω. Furthermore, a continuously differential function $P(x)$ is said to be a *locally filled function* associated with ω at a local minimizer $x_1{}^*$ of $f(x)$ if the following conditions hold:

(a_2) x_1^* is an interior point of ω and a strict local maximizer of $P(x)$.
(b_2) If $B_1(\omega)$ is a local basin containing the point x_1^*, then $P(x)$ does not have any local minimizer or saddle point in $B_1(\omega)$.
(c_2) If there exist local basins lower than $B_1(\omega)$, then at least one of such local basins, e. g., $B_2(\omega)$, satisfies the following condition: There is a point $x_2 \in B_2(\omega)$ such that $P(x)$ decreases strictly along the segment connecting x_1^* and x_2, that is, $P((1-\alpha)x_1^* + \alpha x_2)$ is decreasing strictly with respect to $\alpha \in [0, 1]$.

In [9,22], the difference between a filled function and a locally filled function was illustrated by such a function $y = f(x)$ defined on the interval [–0.5,0.5] as

$$f(x) = z_1(\sin(12\pi x) + 1.5) \,,$$

where the variable z_1 was defined by

$$z_1 = \log(z_2 + 10^{-5}) + 10 \,,$$
$$z_2 = \left(\left(x - \frac{1}{4}\right)^2 \left(x + \frac{1}{4}\right)^2 + 10^{-4}\right) x^2 \,.$$

Note that $x^* = 0.2366$ is one of its local minimizers. An auxiliary function

$$Q(x, x^*, A) = -[f(x) - f(x^*)] \exp\left(A\|x - x^*\|^2\right)$$

does not satisfy the definition of the filled function on [–0.5,0.5] for the parameter $A = 16$, but it satisfies all conditions associated with a locally filled function for the parameter $A = 16$ and the choice of the interval $\omega = [-0.1, 0.3]$.

Methods

If the objective function $f\colon \Re^n \to \Re$ is coercive, then its global minimizer can be found in a suitable large bounded closed set $\Omega \subset \Re^n$ which should be explored completely. In general, let us denote the feasible region for minimizing $f\colon X \subset \Re^n \to \Re$ by Ω, and assume that for any point $x \in \partial\Omega$, $f(x) > \min_{y\in\Omega} f(y)$.

The *basic outline of filled function methods* can be described as follows:

Step 1 Choose an initial point $x_1 \in \Omega$. Denote the maximum of the iteration number and the index of the iterative process by Iter_No and k, respectively. Set $k = 0$.

Step 2 Minimize the function $f(x)$ in Ω starting from the point $x_1 \in \Omega$ and obtain a local minimizer x_1^* of $f(x)$. Denote the basin of the objective function f at x_1^* by B_1^*.

Step 3 Choose two suitable parameters r and ρ, and construct a filled function $P(x, x_1^*, r, \rho)$ associated with x_1^* and f, for example, which is defined by (1).

Step 4 Minimize the filled function $P(x, x_1^*, r, \rho)$ and find another point x_2 in a lower basin B_2^* of f than B_1^* if such a point x_2 exists for a suitable choice of parameters r and ρ.

Step 4.1 If a lower basin B_2^* of f than B_1^* at x_1^* is found, then a new local minimizer x_2^* can be obtained by any local search strategy. Furthermore, perform the replacement of variables such as

$$x_2^* \to x_1^*, \quad B_2^* \to B_1^*, \quad k+1 \to k\,,$$

and go to step 3 (The method continues searching for a global minimum by minimizing another filled function corresponding to the local minimizer x_2^*).

Step 4.2 Otherwise, either the parameters should be adjusted again by an internal updating strategy, or no better local minimizer than x_1^* can be found in Ω.

Step 5 If the iterative index $k >$ Iter_No, or no better local minimizer of f can be found in Ω, the current best local minimizer will be regarded as a global minimizer of f in Ω.

In the above outline of filled function methods, how to choose parameters in a filled function is an important issue, and it may be implemented through an internal iterative process for minimizing $P(x, x_1^*, r, \rho)$ approximately in order to find a lower basin of f or an increasing direction $\bar{x} - x_1^*$ for $P(x, x_1^*, r, \rho)$ at a point $\bar{x}$. An algorithmic implementation and some practical considerations can be found in [5].

Until now people have proposed many kinds of filled functions, for which some are general, while many others are specific [3,5,6,7,8,10,11,12,13,14,15,17,20,21,23]. These filled functions can be classified into four categories.

Two-Parameter Filled Functions

A two-parameter filled function was presented in (1). Although the first filled function method was proposed to deal with unconstrained optimization problems, the two-parameter filled function method had been extended to find a constrained global minimizer [3].

The constrained optimization problem can be formulated as follows:

$$\begin{aligned} &\text{Minimize } f(x),\\ &\text{subject to } g_i(x) \geq 0, \quad i \in \mathcal{I}\,,\\ &\qquad\qquad h_j(x) = 0, \quad j \in \mathcal{E}\,, \end{aligned} \tag{3}$$

where $\mathcal{I}$ and $\mathcal{E}$ are indices sets corresponding to inequalities and equalities, respectively. The two-parameter filled function for problem (3) is defined by

$$P_F(x, x_1^*, r, \rho) = \frac{1}{r + F(x)} \exp\left(-\frac{\|x - x_1^*\|^2}{\rho^2}\right), \tag{4}$$

where

$$F(x) = f(x) + \sum_{i\in\mathcal{I}} \lambda_i \max\{0, -g_i(x)\} + \sum_{j\in\mathcal{E}} \mu_j |h_j(x)| \tag{5}$$

is an exact penalty function for the constrained minimization problem (3), and $\lambda \in \Re_{++}^{|\mathcal{I}|}$, $\mu \in \Re_{++}^{|\mathcal{E}|}$. Since the function defined by (4) is a nonsmooth filled function, the definitions such as basin and filled function should be modified accordingly, see [3].

Two-parameter filled functions have two disadvantages. One is that the changes of both the filled function and its gradient (if available) are affected by the term $\exp(-\|x - x_1^*\|^2/\rho^2)$. When $\|x - x_1^*\|^2$ is large, it is difficult to distinguish these changes, so some pseudo-minimizers, or saddle points or higher minimizers of

the filled functions may be located. The other is that the coordination between r and ρ is very difficult; even a global minimizer x^* may be lost for an improper setting of parameters.

Several modified two-parameter filled functions were proposed in [7] as follows:

$$\tilde{P}(x, x_1^*, r, \rho) = \frac{1}{r + f(x)} \exp\left(-\frac{\|x - x_1^*\|}{\rho^2}\right),$$
$$G(x, x_1^*, r, \rho) = -\rho^2 \log[r + f(x)] - \|x - x_1^*\|^2,$$
$$\tilde{G}(x, x_1^*, r, \rho) = -\rho^2 \log[r + f(x)] - \|x - x_1^*\|.$$

A more general form of filled functions with two parameters can be found in [20]:

$$P(x, r, A) = \psi(r + f(x))\exp(-Aw(\|x - x_k^*\|^\beta)), \quad (6)$$

where $\beta \geq 1$, $A > 0$, the parameter r is chosen such that $r + f(x) > 0$ for all $x \in \Omega$, and the functions $\psi(t)$, $w(t)$ have the following properties:

(i) $\psi(t)$ and $w(t)$ are continuously differentiable for $t \in (0, +\infty)$.
(ii) For $t \in (0, +\infty)$, $\psi(t) > 0$, $\psi'(t) < 0$ and $\psi'(t)/\psi(t)$ is monotonically increasing.
(iii) $w(0) = 0$ and for any $t \in (0, +\infty)$, $w(t) > 0$, $w'(t) \geq c > 0$.

Note that choices for the functions $\psi(t)$ and $w(t)$ can be $1/t^a (a > 0)$, $\mathrm{csch}(t)$, $\exp(1/t) - 1, \ldots$ and $t, \sinh(t), e^t - 1, \ldots$, respectively. The general form of filled functions in (6) includes the class of generalized filled functions considered in [24], which are special two-parameter filled functions.

Since the above filled functions may tend to zero or $-\infty$ as $\|x\| \to +\infty$ for some objective functions $f(x)$ or $F(x)$, they do not approximate a coercive objective function properly. In such a case, a coercive filled function may be preferred. In [8] the concept of a globally convexized filled function for a twice continuously differentiable function $f: \Omega \to \Re$ was introduced.

A continuous function $U(x)$ is a *globally convexized filled function* if it has three properties:

(a) $U(x)$ has no stationary point in the region

$$S_1 = \{x \mid f(x) \geq f(x_1^*), x \in \Omega\},$$

except a prefixed point $x_0 \in S_1$ that is a minimizer of $U(x)$.

(b) $U(x)$ has a minimizer in the region (if it exists)

$$S_2 = \{x \mid f(x) < f(x_1^*), x \in \Omega\}.$$

(c) $\lim_{\|x\| \to +\infty} U(x) = +\infty$.

Two successful globally convexized filled functions can be found in [8] as follows:

$$U_1(x, x_1^*, x_0, A, h) = \|x - x_0\| \times \arctan\{A[f(x) - f(x_1^*) + h]\},$$
$$U_2(x, x_1^*, x_0, A, h) = \|x - x_0\| \times \tanh\{A[f(x) - f(x_1^*) + h]\}.$$

In general, such globally convexized filled functions may be expressed by

$$U(x, x_1^*, x_0, A, h) = \eta(\|x - x_0\|)\phi(A[f(x) - f(x_1^*) + h])$$

for a large enough $A > 0$ and a suitable parameter h such that

$$0 < h < f(x_1^*) - f(x^*),$$

where x^* is a global minimizer of $f(x)$, x_1^* is not a global but is a local minimizer of $f(x)$, and $\eta(t)$ and $\varphi(t)$ are continuously differentiable univariate functions satisfying the following conditions [8]:

(i) $\eta(0) = 0, \eta'(t) \geq \alpha > 0, \forall t \geq 0$.
(ii) $\phi(0) = 0$, $\phi(t)$ is monotonically increasing for all $t \in \Re$ (or for $t \in (-t_1, +\infty)$, where $t_1 > 0$).
(iii) $\phi'(t) > 0, \forall t \in \Re$ (or $\phi'(t) > 0, \forall t \in (-t_1, +\infty)$, where $t_1 > 0$).
(iv) When $t \to +\infty$, $\phi'(t)$ is monotonically decreasing to 0 at least as fast as $1/t$.

Note that choices for these two functions can be t, $\tan(t)$, $e^t - 1, \ldots$ for $\eta(t)$ and $\arctan t, \tanh(t), 1 - e^{-t}, \ldots$ for $\phi(t)$.

Single-Parameter Filled Functions

In order to reduce the difficulty in coordination between r and ρ in a two-parameter filled function, several single-parameter filled functions were proposed in [7]:

$$Q(x, x_1^*, A) = -[f(x) - f(x_1^*)] \exp\left(A\|x - x_1^*\|^2\right),$$
$$\tilde{Q}(x, x_1^*, A) = -[f(x) - f(x_1^*)] \exp\left(A\|x - x_1^*\|\right),$$
$$\nabla E(x, x_1^*, A) = -\nabla f(x) - 2A[f(x) - f(x_1^*)](x - x_1^*),$$
$$\nabla \tilde{E}(x, x_1^*, A) = -\nabla f(x) - A[f(x) - f(x_1^*)]\frac{x - x_1^*}{\|x - x_1^*\|}.$$

More and more single-parameter filled functions appeared afterwards. For example,

$$H(x, x_1^*, a) = \frac{1}{\ln[1 + f(x) - f(x_1^*)]} - a\|x - x_1^*\|^2$$

was proposed in [11], which is defined only for the region where $f(x) \geq f(x_1^*) - 1$. The L function

$$L(x, x_1^*, a) = -\rho\|x - x_1^*\|^2 - [f(x) - f(x_1^*)]^{1/m}$$

and the mitigated L_2 function

$$\mathrm{ML}_2(x, x_1^*, a) = \rho\phi\left(\frac{1}{\|x - x_1^*\|^p}\right) - [f(x) - f(x_1^*)]^{1/m}$$

were proposed in [12] and [13], respectively, where $m > 1$ is a prefixed natural number, ρ is a positive parameter, and φ is a mitigator. A function $y\colon \Re \to \Re$ is said to be a *mitigator* if it is a twice continuously differentiable function in its domain and has the following properties:

(i) $y(0) = 0$, $y'(t) > 0$, and $y''(t) < 0$ for all $t > 0$.
(ii) $\lim_{t \to +\infty} y(t)$ exists.

Note that the ML_2 function can reduce the negative definite effect in the Hessian of a single-parameter filled function such as the L function significantly. The numerical results and generalizations can be found in [12,13,14,15].

A more general form for the single-parameter filled functions can be expressed by

$$Q(x, A) = -\phi(f(x) - f(x_k^*))\exp(Aw(\|x - x_k^*\|^\beta)),$$

where $\beta \geq 1$, $A > 0$, and the functions $\varphi(t)$ and $w(t)$ have the following properties [20]:

(i) $\varphi(t)$ is continuously differentiable for $t \geq 0$.
(ii) $\phi(0) = 0, \phi'(t) > 0, \forall t \geq 0$.
(iii) $\phi'(t)/\phi(t)$ is monotonically decreasing for $t \in (0, +\infty)$.
(iv) $w(0) = 0$ and for any $t \in (0, +\infty)$, $w(t) > 0$, $w'(t) \geq c > 0$.

Note that the choices for these functions can be t, $a^t - 1 (a > 1)$, $\sinh(t)$, ... for $\varphi(t)$ and $t, \sinh(t), e^t - 1$, ... for $w(t)$.

In order to avoid the influence of the exponential term, a general single-parameter filled function can be set by

$$U(x, A) = -\eta(f(x) - f(x_k^*)) - Aw(\|x - x_k^*\|^\beta),$$

where the function $\eta(t)$ is continuous on $[0, +\infty)$ and is differentiable in $(0, +\infty)$. Furthermore, the functions $\eta(t)$ and $w(t)$ have the following properties [20]:

(i) $\eta(0) = 0$;
(ii) $\eta'(t) > 0$ is monotonically decreasing for $t \in (0, +\infty)$ and $\lim_{t \to 0^+} \eta'(t) = +\infty$;
(iii) $w(0) = 0$ and for any $t \in (0, +\infty)$, $w(t) > 0$, $w'(t) \geq c > 0$.

Nonsmooth Filled Functions

It is well known that the constrained optimization problem can be formulated as a nonsmooth optimization problem by using the exact penalty function; see [3] or (3)–(5). With use of the methods of nonsmooth analysis, a nonsmooth unconstrained optimization problem was studied in [10], which involved a modified filled function as follows

$$P_F(x, x_1^*, r, \rho) = \ln\left(1 + \frac{1}{r + F(x)}\right)\exp\left(-\frac{\|x - x_1^*\|^2}{\rho^2}\right), \quad (7)$$

where $F(x)$ is a weak semismooth objective function and x_1^* is a local minimizer of $F(x)$.

For a composite function $F(x)$ in the form

$$F(x) = f(x) + h(c(x)),$$

where $f(x)$ and $c(x) = (c_1(x), \ldots, c_m(x))^T$ are smooth functions and $h\colon R^m \to R$ is convex but nonsmooth [2], a kind of two-parameter filled function

$$P(x, r, A) = \psi(r + f(x))\exp(-A\|x - x_k^*\|^2)$$

was considered in [20], where the function $\psi(t)$ has properties such as:

(i) $\psi(t) > 0$ for $t \geq 0$.
(ii) $\psi(t)$ is monotonically decreasing for $t \geq 0$.
(iii) $\psi(t_1) - \psi(t_2) \leq c_2(t_2 - t_1)$ for $t_2 > t_1 \geq 0$, where $c_2 > 0$ is a constant.

In addition, for the single-parameter filled functions, we can also consider some general forms as follows:

$$U(x, A) = -\phi(f(x) - f(x_k^*))\exp(A\|x - x_k^*\|^2),$$

or

$$\tilde{U}(x, A) = -\phi(f(x) - f(x_k^*)) - A\|x - x_k^*\|^2,$$

where $A > 0$ is a parameter, and the function $\varphi(t)$ is required to satisfy certain conditions [20]:

(i) $\phi(0) = 0$, $\phi(t)$ is monotonically increasing for $t \geq 0$;
(ii) $c_1(t_2 - t_1) \leq \phi(t_2) - \phi(t_1) \leq c_2(t_2 - t_1)$ for $t_2 > t_1 \geq 0$, where $0 < c_1 \leq c_2$ are constants.

Note that even for a continuously differential unconstrained optimization problem, there may exist a nonsmooth filled function. For example, a two-parameter nonsmooth filled function

$$P(x, x_1^*, \rho, \mu) = f(x_1^*) - \min[f(x_1^*), f(x)] - \rho\|x - x_1^*\|^2 + \mu\{\max[0, f(x) - f(x_1^*)]\}^2 \quad (8)$$

was introduced in [21], where $f(x)$ is coercive and Lipschitz continuous with a constant L in $\Re^n$.

Discrete Filled Functions

After the concept of the filled functions was introduced for continuous global optimization by Ge [5], some people tried to transform discrete global optimization problems into continuous ones and then to solve them by the continuous filled function methods [6,17,23].

For the discrete case, since the third property of a continuous filled function usually does not hold, such an extension is not trivial. Difficulties may also occur when continuous optimization methods are applied to deal with discrete optimization problems where the gradient vectors are unavailable or expensive to compute.

Discrete filled functions are related to the concept of the discrete neighborhood. The *discrete neighborhood* for a point $x \in \mathbb{Z}^n$ is usually defined by

$$\mathcal{N}(x) = \{x, x \pm e_i \mid i = 1, 2, \ldots, n\},$$

where e_i is the ith unit vector (i. e., the n-dimensional vector with the ith component equal to 1 and all other components equal to 0). On the basis of the local search approach and the two-parameter filled function defined by (1), Zhu [23] proposed an approximate algorithm for a class of nonlinear integer programming problems

$$\min_{x \in \Omega \cap \mathbb{Z}^n} f(x),$$

where Ω is a bounded closed box with all vertices integral. The algorithm is a direct method, which tries to improve a current discrete local minimal solution by minimizing an associated filled function. In [23], the author used two examples to illustrate the numerical performance of the algorithm proposed there.

In addition, based on the concept of 1/5-*neighborhood* of an integer point x such as

$$\mathcal{N}(x) = \left\{ y \in \Re^n \mid \|y - x\|_\infty \leq \frac{1}{5} \right\},$$

Ge and Huang [6] investigated unconstrained nonlinear integer programming, constrained nonlinear integer programming, and mixed nonlinear integer programming problems. For such cases, the authors tried to use a penalty function to transform a nonlinear integer programming problem into a global optimization problem, which can be solved by the filled function method if the objective function is twice continuously differentiable in $\Re^n$, and its gradient and Hessian matrix are bounded. In particular, when the constraints are equalities, all constrained functions are assumed to be twice continuously differentiable.

The unconstrained nonlinear integer programming model in [6] has the form:

$$\begin{aligned} &\text{Minimize } f(x), \\ &\text{subject to } |x_i| \leq b_i, i = 1, 2, \ldots, n \\ &\qquad x \in \mathbb{Z}^n, \end{aligned} \quad (9)$$

where each b_i is an integer. Under certain conditions, if x^* is a global minimizer of a penalty function

$$\phi_1(x, k) = f(x) - k \sum_{i=1}^{n} \cos 2\pi x_i$$

in the box $\{x \mid |x_i| \leq b_i, i = 1, 2, \ldots, n\}$ and x^* is in a 1/5-neighborhood of an integer point $\bar{x}$, then $\bar{x}$ is a solution of problem (9).

For some integer $m < n$, if the second constraint in (9), $x \in \mathbb{Z}^n$, is replaced by $x_i \in \mathbb{Z} (i = m, m + 1, \ldots, n)$, then the corresponding problem is called the *mixed nonlinear integer programming problem*, for which a similar function

$$\phi_2(x, k) = f(x) - k \sum_{i=m}^{n} \cos 2\pi x_i$$

can be used as a penalty function.

Similarly, for a constrained nonlinear integer programming problem

$$\begin{aligned} &\text{Minimize } f(x)\,, \\ &\text{subject to } c_i(x) = 0, i = 1, 2, \ldots, p\,, \\ &\qquad \min\{0, c_i(x)\} = 0, i = p+1, \ldots, q\,, \\ &\qquad |x_i| \le b_i, i = 1, 2, \ldots, n \\ &\qquad x \in \mathbb{Z}^n\,, \end{aligned} \tag{10}$$

some results can be derived by using the following penalty function:

$$\begin{aligned} \phi_3(x, r, k) = f(x) &+ r \sum_{i=1}^{p} c_i^2(x) \\ &+ r \sum_{i=p+1}^{q} \left[\min\{0, c_i(x)\}\right]^2 \\ &- k \sum_{i=1}^{n} \cos 2\pi x_i\,. \end{aligned}$$

The minimization of $\phi_3(x, r, k)$ can be dealt with by the filled function method proposed for constrained optimization problems [3].

For the discrete optimization problem

$$\min_{x \in X \subset \mathbb{Z}^n} f(x)\,,$$

where f is a Lipschitz function, X is a bounded and (strictly) pathwise connected domain, Ng et al. [17] modified the definition of continuous filled functions in order to allow them to be applied to discrete cases. Now we give a definition of a discrete filled function as follows:

> Given a discrete local minimizer x^* of a function $f: X \subset \mathbb{Z}^n \to R$, let B^* be the discrete basin of f at x^* over X. A function $F: X \to R$ is said to be a *discrete filled function* of f at x^* if it satisfies the following conditions:
>
> (a) x^* is a strict local maximizer of F over X;
> (b) F has no discrete local minimizers in B^* or in any discrete basin of f higher than B^*;
> (c) If f has a discrete basin B^{**} at x^{**} that is lower than B^*, then there is a discrete point $x' \in B^{**}$ that minimizes F on a discrete path $\{x^*, \ldots, x', \ldots, x^{**}\}$ in X.

On the basis of the two-parameter nonsmooth filled function defined by (8) at a local minimizer x_1^*, a two-phase algorithm was proposed to solve a discrete global optimization problem in [17]. In phase 1, a discrete steepest descent method was applied to find a local minimizer x_1^* of f over X, which was called the *local search*. Phase 2 searched for a minimum of the discrete filled function defined by (8) on a discrete path in X via some special search directions, which was called *global search*. The global search would identify a point x' in a discrete basin lower than the discrete basin B_1^* of f at x_1^*. The algorithm stopped when minimizing a discrete filled function did not yield a better solution than the current best local minimizer.

Summary

Many existing filled function methods require the assumption that the objective function has only a finite number of local minimizers. In addition, they also require that these local minima have different objective values. The assignment of single/two parameters in a filled function is a very important issue for ensuring the existence of a specific point for the filled function, by which a better local minimum of the original objective function can be found in a lower basin if it exists. Note that even for a local minimizer existing in a lower basin, how to find it is still a reduced optimization problem.

Furthermore, it is hard to find a general stopping criterion for the filled function methods, i. e., to check whether a feasible point obtained by any of the filled function methods is a global minimizer or not. All these drawbacks indicate that research on the filled function methods will be fascinating in the future. People may consider extensive approaches to solve global optimization problems, for example, by using modified functions which include some nonfilled functions [19], or by using locally filled functions which are integrated with techniques in cluster analysis [9,22].

References

1. Dixon LCW, Gomulka J, Hersom SE (1976) Reflections on global optimization. In: Dixon LCW Optimization in Action. Academic Press, New York, pp 398–435
2. Fletcher R (1981) Practical Methods of Optimization, vol 2. Wiley, New York

3. Ge RP (1987) The theory of filled function methods for finding global minimizers of nonlinearly constrained minimization problems. Presented at SIAM Conference on Numerical Optimization, Boulder, Colorado, 1984. See also J Comput Math 5(1):1–9
4. Ge RP (1989) A parallel global optimization algorithm for rational separable-fractorable functions. Appl Math Comput 32(1):61–72
5. Ge RP (1990) A filled function method for finding a global minimizer of a function of several variables. Presented at the Dundee Conference on Numerical Analysis, Dunded, Scotland, 1983. See also Math Programm 46:191–204
6. Ge RP, Huang CB (1989) A continuous approach to nonlinear integer programming. Appl Math Comput 34(1):39–60
7. Ge RP, Qin YF (1987) A class of filled functions for finding global minimizers of a function of several variables. J Optim Theory Appl 54(2):241–252
8. Ge RP, Qin YF (1990) The globally convexized filled functions for globally optimization. Appl Math Comput 35:131–158
9. Huang HX, Zhao Y (2007) A hybrid global optimization algorithm based on locally filled functions and cluster analysis. Int J Comput Sci Eng 3:194–202
10. Kong M, Zhuang JN (1996) A modified filled function method for finding a global minimizer of a non-smooth function of several variables (In Chinese). Num Math J Chinese Univ 18(2):165–174
11. Liu X (2001) Finding global minima with a computable filled function. J Global Optim 19:151–161
12. Liu X (2002) A computable filled function used for global minimization. Appl Math Comput 126(2–3):271–278
13. Liu X (2002) Several filled functions with mitigators. Appl Math Comput 133(2–3):375–387
14. Liu X (2004) Two new classes of filled functions. Appl Math Comput 149(2):577–588
15. Liu X (2004) The impelling function method applied to global optimization. Appl Math Comput 151(3):745–754
16. Lucidi S, Piccialli V (2002) New classes of global convexized filled functions for global optimization. J Global Optim 24:219–236
17. Ng CK, Zhang LS, Li D, Tian WW (2005) Discrete filled function method for discrete global optimization. Comput Optim Appl 31:87–115
18. Wu ZY, Li HWJ, Zhang LS, Yang XM (2006) A novel filled function method and quasi-filled function method for global optimization. Comput Optim Appl 34(2):249–272
19. Wu ZY, Zhang LS, Teo KL, Bai FS (2005) New modified function method for global optimization. J Optim Theory Appl 125(1):181–203
20. Xu Z, Huang HX, Pardalos PM, Xu CX (2001) Filled functions for unconstrained global optimization. J Global Optim 20:49–65
21. Zhang LS, Ng CK, Li D, Tian WW (2004) A new filled function method for global optimization. J Global Optim 28:17–43
22. Zhao Y (2006) Study of hybrid optimization methods based on locally filled functions. Master Thesis (In Chinese), Tsinghua University
23. Zhu WX (1998) An approximate algorithm for nonlinear integer programming. Appl Math Comput 93(2/3):183–193
24. Zhuang JN (1994) A generalized filled function method for finding the global minimizer of a function of several variables (In Chinese). Num Math J Chinese Univ 16(3):279–287

Global Optimization: Functional Forms

Chrysanthos E. Gounaris,
Christodoulos A. Floudas
Department of Chemical Engineering,
Princeton University, Princeton, USA

Article Outline

Keywords and Phrases
Introduction
Selecting Convex Underestimators:
The αBB Method
Shift Invariance
Sign Invariance
Scale Invariance
Generalization of Shift Invariance
Final Remarks
References

Keywords and Phrases

Global optimization; Convex underestimators; αBB; Functional forms

Introduction

Given the wide variety of different global optimization techniques, every time we have a new optimization problem we must select the best technique for solving this problem. This selection problem is made more complex by the fact that most techniques for solving global optimization problems have parameters that need to be adjusted to the problem or to the class of problems. For example, in gradient methods, one can select different step sizes.

When we have a single or few parameters to choose, it is possible to empirically try many values and come

up with an (almost) optimal value. Thus, in such situations, we can identify an optimal version of the corresponding technique. In other approaches, such as methods like convex underestimators (described in detail in the next section), instead of selecting the value of single *number*-valued parameter, we have to select the auxiliary *function*. It is not practically possible to test all possible functions, so it is not easy to identify an optimal version of the corresponding technique [9].

This entry presents the work of Floudas and Kreinovich [9,10] on the functional forms of convex underestimators for twice continuously differentiable functions. They consider the problem of selecting the best auxiliary function within a given global optimization technique. Specifically, they showed that in many such selection situations, natural symmetry requirements enables one to either analytically solve the problem of finding the optimal auxiliary function, or at least reduce this problem to the easier-to-solve problem of finding a few parameters.

In particular, they showed that we can thus explain both the αBB method [1,2,6,16] and the generalized αBB recently proposed in [4,5]. A recent review article on these deterministic global optimization approaches can be found in [8].

Selecting Convex Underestimators: The αBB Method

It is well known that convex functions are computationally easier to minimize than non-convex ones (see [7]). This relative easiness is not only an empirical fact, it also has a theoretical justification (see [13,19]).

Because of this relative easiness, one of the approaches for minimization of a non-convex function $f(x) = f(x_1, \ldots, x_n)$ (under certain constraints) over a box $[x^L, , x^U] = [x_1^L, x_1^U] \times \ldots \times [x_n^L, x_n^U]$ is to first minimize its convex "underestimator", i. e., a convex function $L(x) \le f(x)$. Since $L(x)$ is an underestimator, the minimum of $L(x)$ is a lower bound for the minimum of $f(x)$. By selecting $L(x)$ as close to $f(x)$ as possible, we can get estimates for $\min f(x)$ which are as close to the actual minimum as possible.

The quality of approximation improves when the boxes become smaller. To get more accurate bounds on $\min f(x)$, we can bisect the box $[x^L, x^U]$ into sub-boxes whithin a regular branch-and-bound framework, and use the above technique to estimate $\min f(x)$ after considering the result of each node and utilizing fathoming of branches where appropriate.

A known efficient approach to designing a convex underestimator is the αBB global optimization algorithm [1,2,6,16], in which we select an underestimator $L(x) = f(x) + \Phi(x)$, where

$$\Phi(x) = -\sum_{i=1}^{n} \alpha_i \cdot (x_i - x_i^L) \cdot (x_i^U - x_i)\,. \tag{1}$$

Here, the parameters α_i are selected in such a way that the resulting function $L(x)$ is convex and still not too far away from the original objective function $f(x)$. For a thorough presentation of ways to select these parameters, see [1,2,3,11].

In many optimization problems, the αBB techniques are very efficient, but in some non-convex optimization problems, it is desirable to improve their performance. One way to do that is to provide a more general class of methods, with more parameters to tune. In the αBB techniques, for each coordinate x_i, we have a single parameter α_i affecting this coordinate. Changing α_i is equivalent to a linear re-scaling of x_i. Indeed, if we change the unit for measuring x_i to a new unit which is λ_i times smaller, then all the numerical values become λ_i times larger: $x_i \to y_i = g_i(x_i)$, where $g_i(x_i) = \lambda_i \cdot x_i$. In principle, we can have two different re-scalings:

- $x_i \to y_i = g_i(x_i) = \lambda_i \cdot x_i$ on the interval $[x^L{}_i, x_i]$, and
- $x_i \to z_i = h_i(x_i) = \mu_i \cdot x_i$ on the interval $[x_i, x^U{}_i]$.

If we substitute the new values $y_i = g_i(x_i)$ and $z_i = h_i(x_i)$ into the formula (1), then we get the following expression

$$\Phi(x) = -\sum_{i=1}^{n} \alpha_i \cdot \big(g_i(x_i) - g_i(x_i^L)\big) \cdot \big(h_i(x_i^U) - h_i(x_i)\big)\,. \tag{2}$$

For the above linear re-scalings, we get

$$\widetilde{\Phi}(x) = -\sum_{i=1}^{n} \widetilde{\alpha}_i \cdot (x_i - x_i^L) \cdot (x_i^U - x_i)\,,$$

where $\widetilde{\alpha}_i = \alpha_i \cdot \lambda_i \cdot \mu_i$.

From this viewpoint, a natural generalization is to replace *linear* re-scalings $g_i(x_i)$ and $h_i(x_i)$ with *non-linear* ones, that is, to consider convex underestimators

of the type $L(x) = f(x) + \Phi(x)$, where $\Phi(x)$ is described by the formula (2) with non-linear functions $g_i(x_i)$ and $h_i(x_i)$. Now, instead of selecting a number α_i for each coordinate i, we have an additional freedom of choosing arbitrary non-linear functions $g_i(x_i)$ and $h_i(x_i)$. The question of which are the best choices is naturally posed. In [4,5], several different non-linear functions were tried, and it turned out that among the tested functions, the best results were achieved for the exponential functions $g_i(x_i) = \exp(\gamma_i \cdot x_i)$ and $h_i(x_i) = -\exp(-\gamma_i \cdot x_i)$. For these functions, the expression (2) can be somewhat simplified: indeed,

$$\begin{aligned}
&\alpha_i \cdot \big(g_i(x_i) - g_i(x_i^L)\big) \cdot \big(h_i(x_i^U) - h_i(x_i)\big) \\
&\quad = \alpha_i \cdot (e^{\gamma_i \cdot x_i} - e^{\gamma_i \cdot x_i^L}) \cdot (-e^{-\gamma_i \cdot x_i^U} + e^{-\gamma_i \cdot x_i}) \\
&\quad = \widetilde{\alpha}_i \cdot (1 - e^{\gamma_i \cdot (x_i - x_i^L)}) \cdot (1 - e^{\gamma_i \cdot (x_i^U - x_i)}) \,,
\end{aligned}$$

where $\widetilde{\alpha}_i \stackrel{\text{def}}{=} \alpha_i \cdot e^{\gamma_i \cdot (x_i^U - x_i^L)}$.

Two related questions naturally arise and are addressed in the work of Floudas and Kreinovich [9,10]:

- first, a *practical* question: an empirical choice is made by using only finitely many functions; is this choice indeed the best – or there are other, even better functions $g_i(x_i)$ and $h_i(x_i)$, which we did not discover because we did not try them?
- second, a *theoretical* question: how can we explain the above empirical fact?

Shift Invariance

The starting point for measuring each coordinate x_i is often a matter of arbitrary choice. If a selection of the functions $g_i(x_i)$ and $h_i(x_i)$ is "optimal" (in some intuitive sense), then the results of using these optimal functions should not change if we simply change the starting point for measuring x_i, that is, replace each value x_i with a new value $x_i + s$, where s is the shift in the starting point. Indeed, otherwise, if the "quality" of the resulting convex underestimators changes with shift, we could apply a shift and get better functions $g_i(x_i)$ and $h_i(x_i)$ – which contradicts the assumption that the selection of $g_i(x_i)$ and $h_i(x_i)$ is already optimal.

The "optimal" choices $g_i(x_i)$ and $g_i(x_i)$ can be determined from the requirement that each component $\alpha_i \cdot (g_i(x_i) - g_i(x_i^L)) \cdot (h_i(x_i^U) - h_i(x_i))$ in the sum (2) be invariant under the corresponding shift, that is, that they satisfy the following definition.

Definition 1 A pair of smooth functions $(g(x), h(x))$ from real numbers to real numbers is *shift-invariant* if for every s and α, there exists $\widetilde{\alpha}(\alpha, s)$ such that for every x^L, x, and x^U, we have

$$\begin{aligned}
&\alpha \cdot \big(g(x) - g(x^L)\big) \cdot \big(h(x^U) - h(x)\big) \\
&\quad = \widetilde{\alpha}(\alpha, s) \cdot \big(g(x+s) - g(x^L + s)\big) \\
&\qquad \cdot \big(h(x^U + s) - h(x+s)\big) \,.
\end{aligned} \tag{3}$$

At first glance, shift invariance is a reasonable but weak property. It turns out, however, that this seemingly weak property actually almost uniquely determines the optimal selection of exponential functions. Proposition 1 applies.

Proposition 1 *If a pair of functions $(g(x), h(x))$ is shift-invariant, then this pair is either exponential or linear, i. e., each of the functions $g(x)$ and $h(x)$ has the form $g(x) = A + C \cdot \exp(\gamma \cdot x)$ or $g(x) = A + k \cdot x$.*

For a proof, see [9] or [10].

Sign Invariance

In addition to shift, another natural symmetry is changing the sign. If we require that the expression (2) remain invariant under a replacement of x by $-x$, then we get the relation between $g(x)$ and $h(x)$: $h(x) = -g(-x)$. So, if a pair $(g(x), h(x))$ is shift-invariant and sign-invariant, then:

- either $g(x) = \exp(\gamma \cdot x)$ and $h(x) = -\exp(-\gamma \cdot x)$,
- or $g(x) = h(x) = x$.

In other words, the optimal generalized αBB scheme is either the original αBB [1,2,6,16], or the scheme with exponential functions described in [4,5].

Scale Invariance

Sign-invariance can be perceived as a special case of scale-invariance. Scale-invariance addresses a change in the unit for measuring x, that is, transformations $x \to \lambda \cdot x$.

We have already shown that there are only two shift-invariant solutions: exponential and linear functions. Out of these two solutions, only the linear solution – corresponding to the original αBB – is scale-invariant. Thus, if we also require scale-invariance, we restrict ourselves only to original techniques and miss the (often better) exponential generalizations.

Although imposing both shift- and scale-invariance leads to restrictions, one could still choose to employ only the latter, formally expressed as follows:

Definition 2 A pair of smooth functions $(g(x), h(x))$ from real numbers to real numbers is *scale-invariant* if for every λ and α, there exists $\widetilde{\alpha}(\alpha,\ \lambda)$ such that for every x^L, x, and x^U, we have

$$\alpha \cdot \big(g(x) - g(x^L)\big) \cdot \big(h(x^U) - h(x)\big) = \widetilde{\alpha}(\alpha, \lambda) \cdot \big(g(\lambda \cdot x) - g(\lambda \cdot x^L)\big) \cdot \big(h(\lambda \cdot x^U) - h(\lambda \cdot x)\big) \quad (4)$$

The following proposition applies. For a proof, see [9].

Proposition 2 *If a pair of functions $(g(x), h(x))$ is scale-invariant, then this pair is either exponential or linear, i. e., each of the functions $g(x)$ and $h(x)$ has the form $g(x) = A \cdot x^\gamma$ or $g(x) = A + k \cdot \ln(x)$.*

From the theoretical viewpoint, these functions may look as good as the exponential functions coming from shift invariance, but in practice, they do not work so well. The problem with these solutions is that they do not preserve smoothness. Both linear and exponential functions which come from shift-invariance are infinitely differentiable for all x and hence, adding the corresponding term $\Phi(x)$ will not decrease the smoothness level of the objective function. In contrast, the functions $g(x) = x^\gamma$ which come from scale invariance are not infinitely differentiable at $x = 0$ or when γ is not integer. So, if we use scale invariance to select a convex underestimator, we end up with a new parameter γ which only attains integer-valued values and is, thus, less flexible than the continuous-valued parameters coming from scale-invariance.

Generalization of Shift Invariance

Instead of the expression (2), we can consider an even more general expression

$$\Phi(x) = -\sum_{i=1}^{n} \alpha_i \cdot a_i(a,\ x^L) \cdot b_i(x_i,\ x_i^U)\,. \quad (5)$$

What can be concluded from shift-invariance in this more general case?

Definition 3 A pair of smooth functions $(a(x, x^L), b(x, x^U))$ from real numbers to real numbers is *shift-invariant* if for every s and α, there exists $\widetilde{\alpha}(\alpha, s)$ such that for every x^L, x, and x^U, we have

$$\alpha \cdot a(x,\ x^L) \cdot b(x,\ x^U) = \widetilde{\alpha}(\alpha,\ s) \cdot a(x + s,\ x^L + s) \cdot b(x + s,\ x^U + s)\,. \quad (6)$$

Regarding such functions, Floudas and Kreinovich [9] proved the following proposition:

Proposition 3 *If a pair of functions $(a(x, x^L), b(x, x^U))$ is shift-invariant, then*

$$a(x,\ x^L) \cdot b(x,\ x^U) = A(x - x^L) \cdot B(x^U - x) \cdot e^{\gamma \cdot x^L}$$

for some functions $A(x)$ and $B(x)$ and for some real number γ.

Comment. If we additionally require that the expression $a(x,\ x^L) \cdot b(x,\ x^U)$ be invariant under $x \to -x$, then we conclude that $B(x) = A(x)$.

Another shift-invariance result comes from the following observation. Both the αBB expression

$$-(x - x^L) \cdot (x^U - x)$$

and the generalized expression

$$-(1 - e^{\gamma \cdot (x - x^L)}) \cdot (1 - e^{\gamma \cdot (x^U - x)})$$

have the form $a(x - x^L) \cdot a(x^U - x)$ with $a(0) = 0$. The differences $x - x^L$ and $x^U - x$ come from the fact that we want these expressions to be shift-invariant. The product form makes sense, since we want the product to be 0 on each border $x = x^L$ and $x = x^U$ of the corresponding interval $[x^L, x^U]$.

On the other hand, it is well known that optimizing a product is more difficult than optimizing a sum; since we will be minimizing the expression $f(x) + \Phi(x)$, it is therefore desirable to be able to reformulate it in terms of the easier-to-minimize sum, e. g., as $b(x - x^L) + b(x^U - x) + c(x^U - x^L)$ for some functions b and c (for minimization purposes, c does not depend on x and is thus a constant). It is worth mentioning that both the αBB expression and its exponential generalization al-

low such representation. Note that:

$$-(x-x^L)\cdot(x^U-x) = \frac{1}{2}\cdot(x-x^L)^2+\frac{1}{2}\cdot(x^U-x)^2-\frac{1}{2}\cdot(x^U-x^L)^2;$$

and

$$-(1-e^{\gamma\cdot(x-x^L)})\cdot(1-e^{\gamma\cdot(x^U-x)}) = -1+e^{\gamma\cdot(x-x^L)}+e^{\gamma\cdot(x^U-x)}-e^{\gamma\cdot(x^U-x^L)}.$$

Interestingly, the above two expressions are the only ones which have this easiness-to-compute property:

Definition 4 We say that a smooth function $a(x)$ from real numbers to real numbers describes an *easy-to-compute* underestimator if $a(0) = 0$, $a'(0) \neq 0$, and there exist smooth functions $b(x)$ and $c(x)$ such that for every x, x^L, and x^U, we have

$$a(x-x^L)\cdot a(x^U-x) = b(x-x^L)+b(x^U-x)+c(x^U-x^L). \tag{7}$$

The condition $a'(0) \neq 0$ comes from the fact that otherwise, for small $\Delta x \stackrel{\text{def}}{=} x-x^L$ and x^U-x, each value $a(x-x^L)$ will be quadratic in $x-x^L$, the resulting product will be fourth order, and we will not be able to compensate for quadratic non-convex terms in the original objective function $f(x)$ – which defeats the purpose of using $f(x)+\Phi(x)$ as a *convex* underestimator.

Proposition 4 *The only functions which describe easy-to-compute underestimators are* $a(x) = k\cdot x$ *and* $a(x) = k\cdot(1-e^{\gamma\cdot x})$.

This is another shift-invariance related result that is also proven in [9]. It selects linear and exponential functions as "the best" in some reasonable sense. Floudas and Kreinovich [9] proved that any "natural" shift-invariant optimality criterion on the set of all possible underestimator methods selects either a linear or an exponential function.

Final Remarks

The work of Floudas and Kreinovich [9,10] has a much further-reaching effect than on the case of αBB-based convex underestimation mainly discussed here. A symmetry-based approach leads to optimal techniques also in the cases of optimal bisection (for selecting box-splitting strategies) and optimal selection of penalty and barrier functions. Other empirically optimal techniques can also be explained by symmetry-based arguments. These include the "epsilon-inflation" technique [15,18], results in simulated annealing and genetic algorithms [17], as well as optimal selection of probabilities in swarm optimization [12,14].

References

1. Adjiman CS, Androulakis IP, Floudas CA (1998) A Global Optimization Method, αBB, for General Twice-Differentiable Constrained NLPs II. Implementation and Computational Results. Comput Chem Eng 22:1159–1179
2. Adjiman CS, Dallwig S, Floudas CA, and Neumaier A (1998) A Global Optimization Method, αBB, for General Twice-Differentiable Constrained NLPs I. Theoretical Advances. Comput Chem Eng 22:1137–1158
3. Adjiman CS, Floudas CA (1996) Rigorous Convex Underestimators for General Twice-Differentiable Problems. J Global Optim 9:23–40
4. Akrotirianakis IG, Floudas CA (2004) A New Class of Improved Convex Underestimators for Twice Continuously Differentiable Constrained NLPs. J Global Optim 30:367–390
5. Akrotirianakis IG, Floudas CA (2004) Computational Experience with a New Class of Convex Underestimators : Box-Constrained NLP Problems. J Global Optim 29:249–264
6. Androulakis IP, Maranas CD, Floudas CA (1995) αBB: A Global Optimization Method for General Constrained Nonconvex Problems. J Global Optim 7:337–363
7. Floudas CA (2000) Deterministic Global Optimization: Theory, Algorithms and Applications. Kluwer, Dordrecht
8. Floudas CA, Akrotirianakis IG, Caratzoulas S, Meyer CA, Kallrath J (2005) Global Optimization in the 21st Century: Advances and Challenges. Comput Chem Eng 29:1185–1202
9. Floudas CA, Kreinovich V (2006) Towards Optimal Techniques for Solving Global Optimization Problems: Symmetry-Based Approach. In: Torn A, Zilinskas J (eds) Models and Algorithms for Global Optimization. Springer, Berlin, pp 21–42
10. Floudas CA, Kreinovich V (2007) On the Functional Form of Convex Understimators for Twice Continuously Differentiable Functions. Optim Lett 1:187–192
11. Hertz D, Adjiman CS, Floudas CA (1999) Two results on bounding the roots of interval polynomials. Comput Chem Eng 23:1333–1339
12. Iourinski D, Starks SA, Kreinovich V, Smith SF (2002) Swarm Intelligence: Theoretical Proof that Empirical Techniques are Optimal. In: Proceedings of the 2002 World Automation Congress WAC 2002, Orlando, FL, pp 107–112
13. Kearfott RB, Kreinovich V (2005) Beyond Convex? Global Optimization is Feasible only for Convex Objective Functions: A Theorem. J Global Optim 33:617–624

14. Kennedy J, Eberhart R, Shi Y (2001) Swarm Intelligence. Morgan Kaufmann, New York
15. Kreinovich V, Starks SA, Meyer G (1997) On a Theoretical Justification of the Choice of Epsilon-Inflation in PASCAL-XSC. Reliable Comput 3:437–452
16. Maranas CD, Floudas CA (1994) Global Minimum Potential Energy Conformations of Small Molecules. J Global Optim 4:135–170
17. Nguyen HT, Kreinovich V (1997) Applications of Continuous Mathematics to Computer Science. Kluwer, Dordrecht
18. Rump SM (1998) A Note on Epsilon-Inflation. Reliable Comput 4:371–375
19. Vavasis SA (1991) Nonlinear Optimization: Complexity Issues. Oxford University Press, Oxford

Global Optimization: g-αBB Approach

CHRYSANTHOS E. GOUNARIS, CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering, Princeton University, Princeton, USA

MSC2000: 49M37, 65K10, 90C26, 90C30, 46N10, 47N10

Article Outline

Keywords and Phrases
Introduction
The New Relaxation Term
The New Underestimating Function
Selection of Appropriate Parameter Values
Computational Results
References

Keywords and Phrases

Convex underestimators; *αBB*; Global optimization

Introduction

Various deterministic global optimization algorithms that utilize a branch and bound framework make use of convex underestimators of the functions under consideration. For a recent review of such approaches, see [7]. For arbitrarily nonconvex C^2-continuous functions $f(x)$, defined in domain $X = [x^L, x^U]$, the αBB underestimator [1,2,3,6,10] is typically used. This is constructed by adding to the original function the following separable relaxation term, $\phi(x;\alpha)$:

$$\phi(x;\alpha) = -\sum_{i=1}^{n} \alpha_i (x_i - x_i^L)(x_i^U - x_i)\,, \tag{1}$$

where $\alpha_i \geq 0$, $i = 1, 2, \ldots, n$. The resulting underestimator of $f(x)$ would thus be

$$L_{\alpha BB}(x;\alpha) = f(x) + \phi(x;\alpha)\,. \tag{2}$$

Since the relaxation term is separable, the following relationship exists among the Hessian matrices of $L_{\alpha BB}(x;\phi)$, $f(x)$ and $\phi(x;\alpha)$:

$$\nabla^2 L_{\alpha BB}(x;\alpha) = \nabla^2 f(x) + 2A\,, \tag{3}$$

where $A = \nabla^2 \phi(x;\alpha) = \operatorname{diag}\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. The addition of the relaxation term corresponds to a diagonal shift of the Hessian matrix. Therefore, if we select large enough values for the α_i parameters, the nonconvexities in the original function can be overpowered and the resulting underestimator $L_{\alpha BB}(x;\alpha)$ becomes convex.

A number of rigorous methods have been devised in order to select appropriate values for these parameters [1,2,3,8]. Extensive computational testing of the algorithm [3] showed that the most efficient of those methods is the one based on the *scaled Gerschgorin* theorem. According to this method, it suffices to select

$$\alpha_i = \max\left[0, -\frac{1}{2}\left(\underline{h_{ii}} - \sum_{\substack{j=1\\ j\neq i}}^{n} \max\left\{|\underline{h_{ij}}|, |\overline{h_{ij}}|\right\} \frac{(x_j^U - x_j^L)}{(x_i^U - x_i^L)}\right)\right]\,, \tag{4}$$

where $\underline{h_{vu}}$ and $\overline{h_{vu}}$ are lower and upper bounds of $\partial^2 f/\partial x_v x_u$ that can be calculated by interval analysis.

The g-αBB approach was developed in [4,5] and offers an alternative convex underestimation functional form than the one originally proposed in the αBB theory. The new relaxation scheme suggests subtraction of a similar separable term that is of exponential, rather than quadratic, nature.

The New Relaxation Term

In this section, we present the new relaxation function. It shares most of the characteristics of the relaxation

function, $\phi(x;\alpha)$, used in the original αBB underestimator described above. However, it possesses novel additional properties that enable it to derive convex underestimators that are tighter to the original function. Thus, the new underestimators can help expedite the branch and bound process of the overall global optimization framework.

The new relaxation function is defined as follows:

$$\Phi(x;\gamma) = -\sum_{i=1}^{n}(1-e^{\gamma_i(x_i-x_i^L)})(1-e^{\gamma_i(x_i^U-x_i)}) , \quad (5)$$

where $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)^{\mathrm{T}}$ is a vector of nonnegative parameters. As will be explained later, these parameters play a similar role as the α_i's in the original αBB method.

The gradient of $\Phi(x;\gamma)$ is

$$\nabla\Phi(x;\gamma) = -\begin{pmatrix} -\gamma_1 e^{\gamma_1(x_1-x_1^L)} + \gamma_1 e^{\gamma_1(x_1^U-x_1)} \\ -\gamma_2 e^{\gamma_2(x_2-x_2^L)} + \gamma_2 e^{\gamma_2(x_2^U-x_2)} \\ \vdots \\ -\gamma_n e^{\gamma_n(x_n-x_n^L)} + \gamma_n e^{\gamma_n(x_n^U-x_n)} \end{pmatrix}$$

and its Hessian is defined by the diagonal matrix

$$\nabla^2\Phi(x;\gamma) = \mathrm{diag}\left\{\gamma_i^2 e^{\gamma_i(x_i-x_i^L)} + \gamma_i^2 e^{\gamma_i(x_i^U-x_i)} : i = 1,2,\ldots,n\right\} .$$

Note that $\nabla^2\Phi(x;\gamma)$ is a function of x as opposed to the Hessian matrix of $\phi(x;\alpha)$, used in αBB, which is constant throughout the domain X.

The new relaxation function $\Phi(x;\gamma)$ has the following important properties:

P1: $\Phi(x;\gamma) \le 0$, for all $x \in [x^L, x^U]$.

P2: $\Phi(x;\gamma) = 0$ at the corner points of the interval $[x^L, x^U]$.

P3: $\Phi(x;\gamma)$ is a convex function.

P4: $\Phi(x;\gamma)$ achieves its minimum at the middle point, x^{mid}, of X and its maximum at the corner points.

P5: The diagonal element of $\nabla^2\Phi(x;\gamma)$ is a convex function and achieves its minimum at the middle point and its maximum at the endpoints of $[x_i^L, x_i^U]$.

The New Underestimating Function

The new underestimating function, $L_1(x;\gamma)$, is formed by adding $\Phi(x;\gamma)$ to the nonconvex function $f(x)$, that is,

$$L_1(x;\gamma) = f(x) + \Phi(x;\gamma) . \quad (6)$$

The Hessian of L_1 is

$$\nabla^2 L_1(x;\gamma) = \nabla^2 f(x) + \nabla^2\Phi(x;\gamma) .$$

The underestimator $L_1(x;\gamma)$ has the following important properties:

U1: $L_1(x;\gamma)$ is an underestimator of $f(x)$.

U2: $L_1(x;\gamma)$ matches $f(x)$ at all corner points of X.

U3: The maximum separation distance between the nonconvex function $f(x)$ and its underestimator $L_1(x;\gamma)$ is bounded.

U4: The underestimators constructed over supersets of the current set are always less tight than the underestimator constructed over the current box constraints.

Since the function $\Phi(x;\gamma)$ is convex for every $x \in X$ and $\gamma \ge 0$, all nonconvexities in the original function $f(x)$ can be eliminated, provided that the parameters γ_i have the appropriate values. The selection of these values is presented in the next section.

Selection of Appropriate Parameter Values

The initial values for the γ_i parameters are selected by solving the following system of nonlinear equations:

$$\ell_i + \gamma_i^2 + \gamma_i^2 e^{\gamma(x_i^U-x_i^L)} = 0, \quad i = 1,2\ldots,n , \quad (7)$$

where $\ell_i \le 0, i = 1,2,\ldots,n$. The parameters ℓ_i convey second-order characteristics of the original nonconvex function into the construction process of the underestimator. Candidate values for these parameters can be selected as follows:

$$\ell_i = -2\alpha_i, \quad i = 1,2\ldots,n , \quad (8)$$

where $\alpha_i \ge 0, i = 1,2,\ldots,n$ are the parameters that correspond to the original αBB method, as given by (4). Akrotirianakis and Floudas [4] proved that such a selection for the γ_i parameters always results in an underestimator that is tighter than the one resulting from the original method, i. e., (2). However, this new underestimator is not necessarily convex. Furthermore, they proved that there always exists some selection of γ_i parameters that results in a convex underestimator.

Therefore, they developed a systematic procedure that determines values for all parameters γ_i that not only guarantee the convexity of the underestimating function $L_1(x;\gamma)$ but also ensure that $L_1(x;\gamma)$ is at least as tight as the underestimating function $L_{\alpha BB}(x;\alpha)$. This procedure is an iterative scheme that is based on interval analysis and consecutive partitions of the domain X. Before we present the scheme, let us present two additional results from [4] that are relevant:

Theorem 1 *Let $\underline{\gamma} = (\underline{\gamma}_1, \underline{\gamma}_2, \ldots, \underline{\gamma}_n)^T$ be the solution of system (7), with ℓ_i defined by (8). Then, the two underestimators $L_1(x;\underline{\gamma})$ and $L_{\alpha BB}(x;\underline{\alpha})$, where*

$$\underline{\alpha} = \left(\frac{4(1 - e^{0.5\underline{\gamma}_1(x_1^U - x_1^L)})^2}{(x_1^U - x_1^L)^2}, \ldots, \frac{4(1 - e^{0.5\underline{\gamma}_n(x_n^U - x_n^L)})^2}{(x_n^U - x_n^L)^2} \right)^T, \quad (9)$$

have the same maximum separation distance from $f(x)$.

Theorem 2 *Let $\overline{\alpha} = (\overline{\alpha}_1, \overline{\alpha}_2, \ldots, \overline{\alpha}_n)^T$ be the values of the α parameters as computed by (4). Then, the two underestimators $L_1(x;\overline{\gamma})$ and $L_{\alpha BB}(x;\overline{\alpha})$, where*

$$\overline{\gamma} = \left(\frac{2\log(1 + \sqrt{\overline{\alpha}_1}(x_1^U - x_1^L)/2)}{x_1^U - x_1^L}, \ldots, \frac{2\log(1 + \sqrt{\overline{\alpha}_n}(x_n^U - x_n^L)/2)}{x_n^U - x_n^L} \right)^T, \quad (10)$$

have the same maximum separation distance from $f(x)$.

The main result of the above two theorems is that for any $\gamma \in [\underline{\gamma}, \overline{\gamma}]$ there exists an $\alpha \in [\underline{\alpha}, \overline{\alpha}]$, such that the underestimators $L_1(x;\gamma)$ and $L_{\alpha BB}(x;\alpha)$ have the same maximum separation distance from the nonconvex function $f(x)$. From all these pairs of underestimators, the only one that is known to be convex a priori is $L_{\alpha BB}(x;\overline{\alpha})$, since this is the one resulting from the classical αBB method. However, as will be apparent from the examples presented later, the underestimators $L_{\alpha BB}(x;\alpha)$ and $L_1(x;\gamma)$ are convex within a large portion of the intervals $[\underline{\alpha}, \overline{\alpha}]$ and $[\underline{\gamma}, \overline{\gamma}]$, respectively. On the basis of the above observations, it is natural to search for a vector γ in the interval $[\underline{\gamma}, \overline{\gamma}]$ or for a vector α in the interval $[\underline{\alpha}, \overline{\alpha}]$, so that at least one of the underestimators $L_1(x;\gamma)$ and $L_{\alpha BB}(x;\alpha)$ is convex.

The algorithm described below was developed in [4] for the appropriate selection of values for the γ parameters, so that the corresponding underestimator is both a convex function and at least as tight as the underestimator used by the classical αBB method. It searches for a vector $\gamma \in [\underline{\gamma}, \overline{\gamma}]$ so that the corresponding $\alpha \in [\underline{\alpha}, \overline{\alpha}]$ produces an underestimating function $L_{\alpha BB}(x;\alpha)$ that is convex. The search starts by setting $\gamma = \underline{\gamma}$ and $\alpha = \underline{\alpha}$ and then checking whether $L_{\alpha BB}(x;\underline{\alpha})$ is convex. This is done by using the scaled Gerschgorin method to determine lower bounds on the eigenvalues of the Hessian matrix $\nabla^2 L_{\alpha BB}(x;\underline{\alpha})$. For those lower bounds that are negative, the intervals of the corresponding variables are bisected, thereby generating a number of subdomains that are stored in a list, denoted by Λ_1. Then, the algorithm checks whether $\nabla^2 L_{\alpha BB}(x;\underline{\alpha})$ is positive semidefinite in each of those subdomains using again the scaled Gerschgorin method. If the size of the list, Λ_1, exceeds a certain number of nodes, then $\nabla^2 L_{\alpha BB}(x;\underline{\alpha})$ is most likely *not* positive semidefinite. The values of all γ_i's have to then be increased by a prespecified positive quantity, $\eta > 0$, and the corresponding values of the new α_i's are calculated. The algorithm now tries to verify whether $\nabla^2 L_{\alpha BB}(x;\alpha)$, with the new increased α parameters, is positive semidefinite. It continues in this manner until the list Λ_1 becomes empty. In that case, the corresponding α values make the Hessian matrix, $\nabla^2 L_{\alpha BB}(x;\alpha)$, positive semidefinite for all $x \in X$ and consequently $L_{\alpha BB}(x;\alpha)$ is a convex underestimator. The main reason for using the underestimator $L_{\alpha BB}(x;\alpha)$ instead of the underestimator $L_1(x;\gamma)$ is that it is easier to verify the positive definiteness of the matrix $\nabla^2 L_{\alpha BB}(x;\alpha)$ than that of the matrix $\nabla^2 L_1(x;\gamma)$. For more details see Alg. 1

Termination of the above algorithm is guaranteed by the fact that $L_{\alpha BB}(x;\overline{\alpha})$ is known, a priori, to be convex underestimator.

Computational Results

Because an iterative procedure is needed to determine appropriate values for the γ_i parameters, the construction of the new underestimators requires more computational effort than that required for the classical αBB method. However, within a global optimization framework, actual computational savings may be realized since the tighter underestimators produced by the

Algorithm:

Step 1 (*initialization*): Set $K = 1$, $J = 1$, $J_{\max} = 2^n + 1$ $\eta = 1.1$ $X_J = X\Lambda_1 = \{X_J\}$ and $\gamma_{i,K} = \underline{\gamma}_i$.

Step 2: For all $i = 1, 2\cdots, n$, use (9) to calculate the $\alpha_{i,K}$ that correspond to $\gamma_{i,K}$, and form the underestimator $L_{\alpha BB}(x;\alpha_K)$.

Step 3: If the maximum separation distance of $L_{\alpha BB}(x;\alpha_K)$ from $f(x)$ is less than the maximum separation distance of $L_{\alpha BB}(x;\overline{\alpha})$ from $f(x)$ then go to step 4.
Otherwise, adopt as an underestimator the classical αBB underestimator, $L_{\alpha BB}(x;\overline{\alpha})$, and stop.

Step 4: Check whether $L_{\alpha BB}(x;\alpha_K)$ is convex:
Repeat

Step 4.1: Remove the last element from the list Λ_1 of unexplored subdomains. Let us name that subdomain X_{last}.

Step 4.2: Form the interval Hessian $[\nabla^2 L_{\alpha BB}(x;\alpha_K)]$ with $x \in X_{last}$.

Step 4.3: Use (4) and (8) to find lower bounds on each eigenvalue of the interval Hessian $[\nabla^2 L_{\alpha BB}(x;\alpha_K)]$ in X_{last}.

Step 4.4: Form the set $I_- = \{i : \ell_i < 0\}$.

Step 4.5: If $I_- \neq \emptyset$, bisect all intervals $[x^L_{i,last}, x^U_{i,last}]$ with $i \in I_-$, and add them at the end of the list Λ_1.

Step 4.6: Set $J = J + 2^{|I_-|} - 1$, where $|I_-|$ represents the cardinality of the set I_- (i. e., a total of $2^{|I_-|}$ new subdomains have been generated and added to the list and one node has been removed).

Until ($\Lambda_1 = \emptyset$ or $J = J_{\max}$).

Step 5: If $\Lambda_1 = \emptyset$ then stop. The Hessian $\nabla^2 L_{\alpha BB}(x;\alpha_K)$ is positive semidefinite for all $x \in X$ and $L_{\alpha BB}(x;\alpha_K)$ is a convex underestimator. Also the underestimator $L_{\alpha BB}(x;\alpha_K)$ is tighter than the underestimator $L_{\alpha BB}(x;\overline{\alpha})$ obtained by the classical αBB method.
Otherwise, increase the values of all $\gamma_{i,K}$, $i = 1, 2, \ldots, n$ by setting $\gamma_{i,K+1} = \eta\gamma_{i,K}$. Set $K = K + 1$ and go to step 2.

Global Optimization: g-αBB Approach, Algorithm 1

new method could expedite the branch and bound process through faster fathoming and visits to fewer tree nodes.

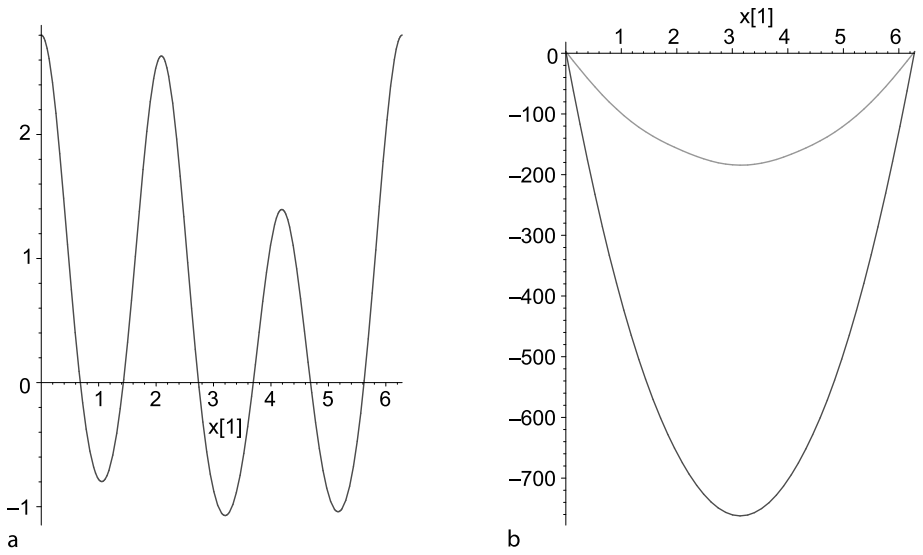
A detailed computational comparison between the new underestimators and the ones used by the classical αBB method was performed by Akrotirianakis and Floudas [5]. They concluded that the new underestimators usually perform better than the classical αBB method, in terms of both the overall CPU time and the number of nodes generated by the enumeration tree. It was also observed that the new underestimators perform better when the problem involves many arbitrarily nonconvex terms in the objective or constraints.

In the same study, Akrotirianakis and Floudas [5] also presented a hybrid optimization framework where underestimators $L_1(x;\underline{\gamma})$ were used to construct the relaxation in every node of the branch and bound tree. A stochastic random-linkage algorithm [9] was then employed to solve these relaxations and the method exhibited improved computational efficiency. Interestingly enough, the method located the actual global optimum in all case studies, despite the lack of theoretical guarantees owing to the fact that the underestimators $L_1(x;\underline{\gamma})$ are not necessarily convex.

As an illustration, we present here two examples from [5]:

Example 1

This example involves a nonconvex function that describes the molecular conformation of pseudoethane. It is taken from [11], where the global minimum potential energy conformation of small molecules is studied. The Lennard-Jones potential is expressed in terms of a sim-



Global Optimization: g-αBB Approach, Figure 1
Function $f_1(x)$ and comparison of underestimators $L_{\alpha BB}(x;\overline{\alpha})$ and $L_{\alpha BB}(x;\alpha)$

ple dihedral angle. The potential energy of the molecule is given by the following function:

$$
\begin{aligned}
f_1(x) = & \frac{588600}{(3r_0^2-4cos(\theta)r_0^2-2(sin^2(\theta)cos(x-\frac{2\pi}{3})-cos^2(\theta))r_0^2)^6} \\
& - \frac{1079.1}{3r_0^2-4cos(\theta)r_0^2-2(sin^2(\theta)cos(x-\frac{2\pi}{3})-cos^2(\theta))r_0^2)^3} \\
& + \frac{600800}{(3r_0^2-4cos(\theta)r_0^2-2(sin^2(\theta)cos(x)-cos^2(\theta))r_0^2)^6} \\
& - \frac{1071.5}{(3r_0^2-4cos(\theta)r_0^2-2(sin^2(\theta)cos(x)-cos^2(\theta))r_0^2)^3} \\
& + \frac{481300}{(3r_0^2-4cos(\theta)r_0^2-2(sin^2(\theta+\frac{2\pi}{3})cos(x)-cos^2(\theta))r_0^2)^6} \\
& - \frac{1064.6}{(3r_0^2-4cos(\theta)r_0^2-2(sin^2(\theta+\frac{2\pi}{3})cos(x)-cos^2(\theta))r_0^2)^3} ,
\end{aligned}
$$

where r_0 is the covalent bond length ($r_0 = 1.54A$), θ is the covalent bond angle ($\theta = 109.5^o$) and x is the dihedral angle ($x \in X = [0, 2\pi]$). Figure 1 depicts the graph of $f_1(x)$.

The value of the α parameter computed by the classical αBB method using (4) is $\overline{\alpha} = 77.124$ and the corresponding value for the γ parameter, obtained by (10), is $\overline{\gamma} = 1.0673$. Also, by solving (7) for γ we obtain $\underline{\gamma} = 0.8521$ and the corresponding value for the α parameter, obtained by (9), is $\underline{\alpha} = 18.579$. The iterative algorithm checks whether there exist values of $\gamma \in [\underline{\gamma}, \overline{\gamma}]$ and $\alpha \in [\underline{\alpha}, \overline{\alpha}]$ such that the underestimator $L_{\alpha BB}(x;\alpha)$ is convex. After 16 iterations it concludes that if $\alpha = \underline{\alpha}$, then $L_{\alpha BB}(x;\alpha)$ is a convex underestimator of $f_1(x)$. Furthermore, if $\gamma = \underline{\gamma}$, then $L_1(x;\gamma)$ is also a convex underestimator of $f_1(x)$. Note that the values of γ and α did not have to increase at all.

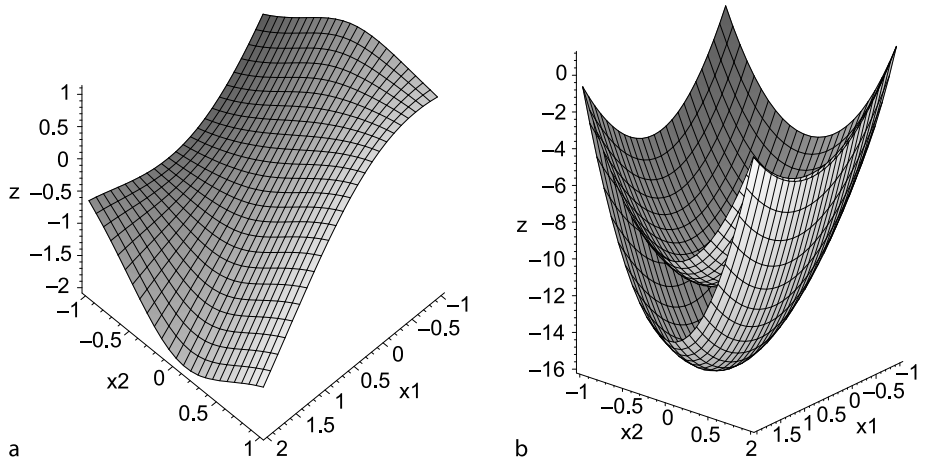
The resulting minima of the two underestimators $L_{\alpha BB}(x;\overline{\alpha})$ and $L_{\alpha BB}(x;\alpha)$ are -762.2377 and -184.4244, respectively. Figure 1 depicts these two underestimators and reveals the improvement in tightness.

Example 2 This example is taken from [2] and examines the following two-dimensional nonconvex function:

$$f_2(x) = \cos(x_1)\sin(x_2) - \frac{x_1}{x_2^2 + 1} ,$$

where $x_1 \in [-1, 2]$ and $x_2 \in [1, 1]$. The above function possesses three minima and its graph is depicted in Fig. 2. The values of the $\overline{\alpha}$ parameters computed by the classical αBB method using (4) are $\overline{\alpha}_1 = 1.921$ and $\overline{\alpha}_2 = 10.921$. Using (10), we can determine the corresponding value for the $\overline{\gamma}$ parameters; these are $\overline{\gamma}_1 = 0.75$ and $\overline{\gamma}_2 = 1.46$. Also, by solving (7) for $\gamma_i, i = 1, 2$, we obtain $\underline{\gamma}_1 = 0.672$ and $\underline{\gamma}_2 = 1.267$. Using (9), we can determine the corresponding values for the $\underline{\alpha}$ parameters; these are $\underline{\alpha}_1 = 1.3456$ and $\underline{\alpha}_2 = 6.5$.

The iterative algorithm checks whether there exist values of $\gamma_i \in [\underline{\gamma}_i, \overline{\gamma}_i], i = 1, 2$ and $\alpha_i \in [\underline{\alpha}_i, \overline{\alpha}_i], i = 1, 2$, such that the underestimator $L_{\alpha BB}(x;\alpha)$ is convex. After eight iterations it concludes



Global Optimization: g-αBB Approach, Figure 2
Function $f_2(x)$ and comparison of underestimators $L_{\alpha BB}(x;\overline{\alpha})$ and $L_{\alpha BB}(x;\alpha)$

that if $\alpha = (1.8325, \underline{\alpha}_2)$, then $L_{\alpha BB}(x;\alpha)$ is a convex underestimator of $f_2(x)$. Also, if $\gamma = (0.74, \underline{\gamma}_2)$, then $L_1(x;\gamma)$ is also a convex underestimator of $f_2(x)$. Note that only the value of γ_1 had to be increased from its original value, $\underline{\gamma}_1$, and the increase was only by 10%.

The resulting minima of the two underestimators $L_{\alpha BB}(x;\overline{\alpha})$ and $L_{\alpha BB}(x;\alpha)$ are -15.88469 and -10.22767, respectively. Figure 2 depicts these two underestimators and reveals the improvement in tightness.

References

1. Adjiman CS, Floudas CA (1996) Rigorous convex underestimators for general twice-differentiable problems. J Glob Optim 9:23–40
2. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, *αBB*, for general twice-differentiable constrained NLPS I. theoretical advances. Comput Chem Eng 22:1137–1158
3. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB, for general twice-differentiable constrained NLPs II. Implementation and Computational Results. Comput Chem Eng 22:1159–1179
4. Akrotirianakis IG, Floudas CA (2004) A new class of improved convex underestimators for twice continuously differentiable constrained NLPs. J Glob Optim 30:367–390
5. Akrotirianakis IG, Floudas CA (2004) Computational experience with a new class of convex underestimators: box-constrained NLP problems. J Glob Optim 29:249–264
6. Androulakis IP, Maranas CD, Floudas CA (1995) *αBB*: A global optimization method for general constrained nonconvex problems. J Glob Optim 7:337–363
7. Floudas CA, Akrotirianakis IG, Caratzoulas S, Meyer CA, Kallrath J (2005) Global optimization in the 21st century: advances and challenges. Comput Chem Eng 29: 1185–1202
8. Hertz D, Adjiman CS, Floudas CA (1999) Two results on bounding the roots of interval polynomials. Comput Chem Eng 23:1333–1339
9. Locatelli M, Schoen F (1999) Random linkage: a family of acceptance/rejection algorithms for global optimization. Math Program 85:379–396
10. Maranas CD, Floudas CA (1994) Global minimum potential energy conformations of small molecules. J Glob Optim 4:135–170
11. Maranas CD, Floudas CA (1994) A deterministic global optimization approach for molecular structure determination. J Chem Phys 100:1247–1261

Global Optimization in Generalized Geometric Programming

COSTAS D. MARANAS
Pennsylvania State University, University Park, USA

MSC2000: 90C26, 90C90

Article Outline

Keywords
Robust Stability Analysis
See also
References

Keywords

Signomials; Generalized geometric programming; Global optimization; Robust stability analysis

Generalized geometric GGP or *signomial programming* (GGP) problems are characterized by an objective function and constraints which are the difference of two *posynomials*. A posynomial $G(\mathbf{x})$ is simply the sum of a number of *posynomial terms* or *monomials* $g_k(\mathbf{x})$, $k = 1, \ldots, K$, multiplied by some positive real constants c_k, $k = 1, \ldots, K$. Each monomial $g_k(\mathbf{x})$ is in turn the product of a number of positive variables each of them raised to some real power,

$$g_k(\mathbf{x}) = x_1^{d_{1,k}} \cdots x_n^{d_{N,k}}, \quad k = 1, \ldots, K,$$

where $d_{1,k}, \ldots, d_{N,k} \in \mathbf{R}$ and are not necessarily integers. The term 'geometric programming' was adopted because of the key role that the well-known arithmetic-geometric inequality played in the initial developments. Generalized geometric problems were first introduced and studied by U. Passy and D.J. Wilde [28] and G.J. Blau and Wilde [8] when existing (posynomial) geometric programming (GP) formulations failed to account for the presence of negatively signed monomials in models for important engineering applications. These applications are extensively reviewed in [31] and [16]. Chemical engineering applications include heat exchanger network design [14], chemical reactor design [8,9], optimal condenser design [4], oxygen production [21], membrane separation process design [12], optimal design of cooling towers [16], chemical equilibrium problems [29], optimal control [23], batch plant modeling [20,33], optimal location of hydrogen supply centers [3] and many more.

By grouping together monomials with identical sign, the generalized geometric problem can be formulated as the following nonlinear optimization problem:

$$GGP \begin{cases} \min\limits_{\mathbf{t}} & G_0(\mathbf{t}) = G_0^+(\mathbf{t}) - G_0^-(\mathbf{t}) \\ \text{s.t.} & G_j(\mathbf{t}) = G_j^+(\mathbf{t}) - G_j^-(\mathbf{t}) \le 0, \\ & j = 1, \ldots, M, \\ & t_i \ge 0, \quad i = 1, \ldots, N, \end{cases}$$

where

$$G_j^+(\mathbf{t}) = \sum_{k \in K_j^+} c_{jk} \prod_{i=1}^{N} t_i^{\alpha_{ijk}},$$

$$j = 0, \ldots, M,$$

$$G_j^-(\mathbf{t}) = \sum_{k \in K_j^-} c_{jk} \prod_{i=1}^{N} t_i^{\alpha_{ijk}},$$

$$j = 0, \ldots, M,$$

where $\mathbf{t} = (t_1, \ldots, t_N)$ is the positive variable vector; G_j^+, G_j^-, $j = 0, \ldots, M$, are positive posynomial functions in $\mathbf{t}$; α_{ijk} are arbitrary real constant exponents; and c_{jk} are positive coefficients. Also, the sets K_j^+, K_j^- count how many positively/negatively signed monomials form posynomials G_j^+, G_j^- respectively. In general, formulation GGP corresponds to a nonlinear optimization problem with nonconvex objective function and/or constraint set. Note that if we set $K_j^- = 0$ for all $j = 0, \ldots, M$ then the mathematical model for GGP reduces to the (posynomial) geometric programming (GP) formulation which laid the foundation for the theory of generalized geometric problems.

Unlike (posynomial) problems (GP), the problems GGP remain nonconvex in both their primal and dual representation and no known transformation can convexify them. They may involve multiple local minima and/or nonconvex feasible regions and therefore are much more difficult problems to solve. Local optimization approaches for solving GGP problems include bounding procedures based on *posynomial condensation* [2,5,13,15,23]; iterative solution of KKT conditions [9,25,32]; and adaptations of general purpose nonlinear programming methods [1,7,10,19,24,26,31]. A computational comparison of available codes for signomial programming is given in [12,32]. While local optimization methods for solving GGP problems are ubiquitous, application of specialized global optimization algorithms on GGP problems is scarce. J.E. Falk [17] proposed such a global optimization algorithm based on the exponential variable transformation of GGP and the convex relaxation and branch and bounding on the space of exponents of negative monomials ($j = 1, \ldots, M$ and $k \in K_j^-$). Based on these ideas, C.D. Maranas and C.A. Floudas [27] proposed an alternative partitioning in the typically smaller space of variables $i =$

1, …, N. The proposed branch and bound type algorithm attains finite ϵ-convergence to the global minimum through the successive refinement of a convex relaxation of the feasible region and/or of the objective function and the subsequent solution of a series of nonlinear convex optimization problems. The efficiency of the proposed approach is enhanced by eliminating variables through monotonicity analysis, by maintaining tightly bound variables through rescaling, by further improving the supplied variable bounds through convex minimization. The proposed approach was applied to a large number of test examples, in particular robust stability analysis problems.

Robust Stability Analysis

Robust stability analysis of nonlinear systems involves the identification of the largest possible region in the un- certain model parameter space for which the controller manages to attenuate any disturbances in the system. The stability of a feedback structure is determined by the roots of the closed loop characteristic equation:

$$\det\left(I + P(s,\mathbf{q})C(s,\mathbf{q})\right) = 0,$$

where $\mathbf{q}$ is the vector of the uncertain model parameters, and $P(s)$, $C(s)$ the transfer functions of the plant and controller, respectively. After expanding the determinant we have:

$$P(s,\mathbf{q}) = a_n(\mathbf{q})s^n + a_{n-1}(\mathbf{q})s^{n-1} + \cdots + a_0(\mathbf{q}) = 0,$$

where the coefficients $a_i(\mathbf{q})$, $i = 0, \ldots, n$, are typically multivariable polynomial functions. The 'zero exclusion condition' (ZEC) implies that a system with characteristic equation $P(\mathbf{q}, s) = 0$ is stable only if it does not have any roots on the imaginary axis for any realization of the $\mathbf{q}$s in the uncertain model parameter space $\mathcal{Q}$:

$$0 \notin P(j\omega,\mathbf{q}), \quad \forall \mathbf{q} \in \mathcal{Q}, \text{ and } \forall \omega \in [0,\infty].$$

A stability margin k_m can then be defined as follows:

$$k_m(j\omega) = \inf\{k\colon\ P(j\omega,\mathbf{q}(k)) = 0,\ \forall \mathbf{q} \in \mathcal{Q}\}.$$

Robust stability for this model is then guaranteed if and only if

$$k_m \geq 1.$$

Geometrically, k_m expands the initial uncertain parameter region $\mathcal{Q}$ as much as possible without loosing stability. Note that, typically real parameter uncertainty is expressed as bounds on the real parameters of the model.

Checking the stability of a particular system with characteristic equation $P(j\omega, \mathbf{q})$ involves the solution of the following nonconvex optimization problem.

$$(S)\begin{cases} \min\limits_{q_i, k\geq 0, \omega \geq 0} & k \\ \text{s.t.} & \mathrm{Re}[P(j\omega,\mathbf{q})] = 0 \\ & \mathrm{Im}[P(j\omega,\mathbf{q})] = 0 \\ & q_i^N - \Delta q_i^- k \leq q_i \\ & \quad \leq q_i^N + \Delta q_i^+ k, \\ & i = 1, \ldots, n, \end{cases}$$

where $\mathbf{q}^N$ is a stable nominal point for the uncertain parameters and $\Delta\mathbf{q}^+$, $\Delta\mathbf{q}^-$ are estimated bounds. Note that it is important to be able to always locate the global minimum of (S), otherwise the stability margin might be overestimated. This overestimation can sometimes lead to the erroneous conclusion that a system is stable when it is not. Because for most problems without time delays $a_i(\mathbf{q})$, $i = 0, \ldots, n$, are multivariable polynomial functions, formulation (S) corresponds to a generalized geometric problem. Next, an illustrative robust stability example is highlighted.

This example was studied in [18] and [30]. The plant has three uncertain parameters and the characteristic equation is:

$$P(s,q_1,q_2,q_3) = s^4 + (10 + q_2 + q_3)s^3 + (q_2q_3 + 10q_2 + 10q_3)s^2 + (1 - q_2q_3 + q_1)s + 2q_1\,.$$

The nominal values of the parameters of the system are

$$q_1^N = 800, \quad q_2^N = 4, \quad q_3^N = 6,$$

and the bounded perturbations are:

$$\Delta q_1^+ = \Delta q_1^- = 800,$$
$$\Delta q_2^+ = \Delta q_2^- = 2,$$
$$\Delta q_3^+ = \Delta q_3^- = 3.$$

After eliminating ω the zero exclusion formulation becomes:

$$\begin{cases} \min & k \\ \text{s.t.} & 10q2^2q_3^3 + 10q_2^3q_3^2 + 200q_2^2q_3^2 \\ & \quad +100q_2^3q_3 + 100q_2q_3^3 + q_1q_2q_3^2 \\ & \quad +q_1q_2^2q_3 + 1000q_2q_3^2 + 8q_1q_3^2 \\ & \quad +1000q_2^2q_3 + 8q_1q_2^2 + 6q_1q_2q_3 \\ & \quad +60q_1q_3 + 60q_1q_2 \\ & \quad -q_1^2 - 200q_1 \le 0 \\ & 800 - 800k \le q_1 \le 800 + 800k \\ & 4 - 2k \le q_2 \le 4 + 2k \\ & 6 - 3k \le q_3 \le 6 + 3k. \end{cases}$$

The stability margin is found to be $k_m = 0.3417$, which implies that the system is unstable. Furthermore, the first instability occurs at:

$$q_1^* = 1073.4,$$
$$q_2^* = 3.318,$$
$$q_3^* = 4.975.$$

See also

- ▶ αBB Algorithm
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Convex Envelopes in Optimization Problems
- ▶ Global Optimization in Batch Design Under Uncertainty
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization in Phase and Chemical Reaction Equilibrium
- ▶ Interval Global Optimization
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Global Optimization with αBB
- ▶ Smooth Nonlinear Nonconvex Optimization

References

1. Abrams RA, Wu CT (1978) Projection and restriction methods in geometric and related problems. JOTA 26:59
2. Avriel M, Dembo R, Passy U (1975) Solution of generalized geometric programs. SIAM Internat J Numer Methods Eng 26:291
3. Avriel M, Gurovich V (1967) Optimal condenser design by geometric programming. I–EC Proc Des Developm 6:256
4. Avriel M, Wilde DJ (1967) Optimal condenser design by geometric programming. I–EC Proc Des Developm 6:256
5. Avriel M, Williams AC (1970) Complementary geometric programming. SIAM J Appl Math 19:125
6. Avriel M, Williams AC (1971) An extension of geometric programming with applications in engineering optimization. J Eng Math 5(3):187
7. Beck PA, Ecker JG (1975) A modified concave simplex algorithm for geometric programming. JOTA 15:189
8. Blau GE, Wilde DJ (1969) Generalized polynomial programming. Canad J Chem Eng 47:317
9. Blau GE, Wilde DJ (1971) A Lagrangian algorithm for equality constrained generalized polynomial optimization. AIChE 17:235
10. Bradley J (1973) An algorithm for the numerical solution of prototype geometric programs. Inst Indust Res and Standards (Dublin, Ireland)
11. Dembo RS (1976) A set of geometric programming test problems and their solutions. Math Program 10:192
12. Dembo RS (1978) Optimal design of a membrane separation process using signomial programming. Math Program 15:12
13. Duffin RJ (1970) Linearizing geometric problems. SIAM Rev 12:211
14. Duffin RJ, Peterson EL (1966) Duality theory for geometric programming. SIAM J Appl Math 14:1307
15. Duffin RJ, Peterson EL (1972) Reversed geometric programming treated by harmonic means. Indiana Univ Math J 22:531
16. Ecker JG, Wiebking RD (1978) Optimal design of a dry–type natural–draft cooling tower by geometric programming. JOTA 26:305
17. Falk JE (1973) Global solutions of signomial programs. Report The George Washington Univ Program in Logistics
18. Gaston RRE, Safonov MG (1988) Exact calculation of the multi-loop stability margin. IEEE Trans Autom Control 33:68
19. Haarhoff PC, Buys JD (1970) A new method for the optimization of a nonlinear function subject to nonlinear constraints. Comput J 13:178
20. Hellinckx LJ, Rijckaert MJ (1971) Minimization of capital investment for batch processes. I–EC Proc Des Developm 10:422
21. Hellinckx LJ, Rijckaert MJ (1972) Optimal capacities of production facilities: An application of geometric programming. Canad J Chem Eng 50:148
22. Horst R, Tuy H (1990) Global optimization, deterministic approaches. Springer, Berlin
23. Jefferson TR, Scott CH (1978) Generalized geometric programming applied to problems of optimal control: I. Theory. JOTA 26:117

24. Kochenberger A, Woolsey RED, McCarl BA (1973) On the solution of geometric programming via separable programming. Oper Res Quart 24:285
25. Kuester JL, Mize JH (1973) Optimization techniques with Fortran. McGraw-Hill, New York
26. Ratner M, Lasdon LS, Jain A (1978) Solving geometric problems using GRG: Results and comparisons. JOTA 26:253
27. Maranas CD, Floudas CA (1997) Global optimization in generalized geometric programming. Comput Chem Eng 21:251
28. Passy U, Wilde DJ (1967) Generalized polynomial optimizations. SIAM J Appl Math 15:1344
29. Passy U, Wilde DJ (1967) A geometric programming algorithm for solving chemical equilibrium problems. SIAM Rev 11:89
30. Psarris P, Floudas CA (1995) Robust stability analysis of linear and nonlinear systems with real parameter uncertainty. Comput Chem Eng 5:699
31. Rijckaert MJ, Martens XM (1978) Bibliographical note on geometric programming. JOTA 2:325
32. Rijckaert MJ, Martens XM (1978) Comparison of generalized geometric programming algorithms. JOTA 26:205
33. Salomone HE, Iribarren OA (1992) Posynomial modeling of batch plants: A procedure to include process decision variables. Comput Chem Eng 16:173

Global Optimization of Heat Exchanger Networks

Juan M. Zamora[1], Ignacio E. Grossmann[2]

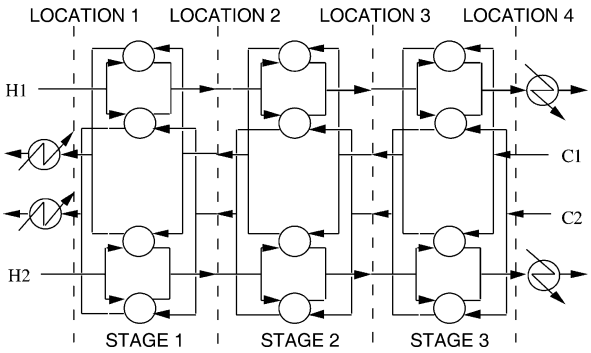
[1] University Autónoma Metropolitana-Iztapalapa, Mexico City, Mexico

[2] Carnegie Mellon University, Pittsburgh, USA

MSC2000: 90C26, 90C90

Article Outline

Keywords
Nonconvex Model (P)
  Indices
  Parameters
  Positive Variables
  Objective Function
  Model Constraints
Convex Model (R)
  Objective Function
  Model Constraints
See also
References



Global Optimization of Heat Exchanger Networks, Figure 1
Head exchanger network superstructure

Keywords

Global optimization; Heat exchanger networks; Convex relaxations; Global optimal design; Bilinear terms; Linear fractional terms

The cost of energy represents an important part of the total operating cost of many processing plants. Therefore, the recovery of energy through heat exchanger networks (HENs) has played an important role in industry, and has been a major concern of design engineers for the last two decades (for reviews, see [5,10,11]). Design approaches based on mathematical programming techniques and models have been developed and applied in the synthesis and the optimization of HENs (see for instance [3,12,18]). The synthesis of HENs with a mathematical modeling framework involves the optimization of a superstructure like the one in Fig. 1 [18], and represents a difficult global optimization problem from a deterministic point of view [20].

Nonconvexities are introduced into mathematical models for HENs by the fractional powers of linear fractional terms that appear in heat transfer area cost terms,

$$\text{Area Cost} = C\left(\frac{q}{U\Delta T}\right)^{\beta}.$$

Here the variables are the heat transfer rate, q, and the logarithmic mean temperature difference driving force, ΔT or LMTD, U is the heat transfer coefficient, and C and β are cost coefficient and exponent, respectively. Other sources of nonconvexities in mathematical programming models for heat exchanger networks arise due to the logarithmic mean temperature difference

driving force, which can be given rigorously

$$\text{LMTD} = \frac{\left[dt_\text{h} - dt_\text{c}\right]}{\log\left[\frac{dt_\text{h}}{dt_\text{c}}\right]},$$

or by an approximation like the ones due to W.R. Paterson [13]

$$\text{LMTD} = \frac{1}{3}\left[\frac{1}{2}(dt_\text{h} + dt_\text{c})\right] + \frac{2}{3}\sqrt{dt_\text{h} dt_\text{c}}$$

or J.J.J. Chen [2]

$$\text{LMTD} = \left[\frac{(dt_\text{h})(dt_\text{c})(dt_\text{h} + dt_\text{c})}{2}\right]^{\frac{1}{3}}.$$

Here, dt_h and dt_c are the temperature differences at the hot and cold extremes in the heat exchanger. Nonconvexities in mathematical models of HENs also may appear in the form of bilinear terms that are used to model the nonisothermal mixing of process streams. For instance, the energy balance for modeling the nonisothermal mixing of process streams 1 and 2 to produce stream 3 would require the inclusion of the following bilinear equation in the mathematical model:

$$f_1 t_1 + f_2 t_2 = f_3 t_3,$$

in which f stands for heat capacity flowrate, and t for stream temperature.

The issue of determining a global optimum solution for problems involving heat exchanger networks was first considered in [17]. Since then, representative global optimization problems in heat exchanger networks have been posed, see for instance [4]. Nevertheless, deterministic global optimization algorithms, and their application to the optimization of certain classes of NLP and MINLP models in heat exchanger networks appeared only until the 1990s in [1,6,9,14,15,16,20,21,22].

Most of the applications of deterministic global optimization algorithms for the solution of nonconvex problems involving HENs are based on a *branch and bound* framework [7,8]. Within the branch and bound approach for global optimization, lower bounds of the global minimum value of the objective function are computed by solving a convex relaxation of the original nonconvex problem over subsections of the search region. For the development of the convex relaxations for nonconvex problems in HENs, the following properties are exploited.

Property 1 ([19,20,21,22]) Let θ and ΔT be continuous positive variables with $\Delta T > 0$. Also, let U, C, α and β be positive constants, with $\beta > 0$, and $\alpha = (\beta + 1)/\beta$. Then, the function

$$C\left(\frac{\theta^\alpha}{U\Delta T}\right)^\beta$$

is convex. Furthermore, if q is a positive variable, and S is a convex subset in $\mathbf{R}^2_+$, the convex optimization problem in (2) can be used to compute a rigorous lower bound for the solution of the problem in (1), i. e., the problem in (2) is a valid convex relaxation of the problem in (1):

$$\begin{cases} \text{GloMin} & C\left(\frac{q}{U\Delta T}\right)^\beta \\ \text{s.t.} & (q, \Delta T) \subseteq S \\ & 0 \le q^L \le q \le q^U, \end{cases} \tag{1}$$

$$\begin{cases} \min & C\left(\frac{\theta^\alpha}{U\Delta T}\right)^\beta \\ \text{s.t.} & \theta \ge (q^L)^{\frac{1}{\alpha}} \\ & \quad + \frac{(q^U)^{\frac{1}{\alpha}} - (q^L)^{\frac{1}{\alpha}}}{q^U - q^L}(q - q^L) \\ & (q, \Delta T) \subseteq S, \\ & 0 \le q^L \le q \le q^U, \quad \theta \ge 0. \end{cases} \tag{2}$$

Property 2 ([19]) Let dt_h, dt_c and ΔT, be continuous positive variables. Also, let T_1 and T_2, be positive constants such that $T_1 - T_2 > 0$. Then the following inequalities are convex:

$$\Delta T \le \frac{\left[dt_\text{h} - dt_\text{c}\right]}{\log\left[\frac{dt_\text{h}}{dt_\text{c}}\right]},$$

$$\Delta T \le \frac{\left[dt_\text{h} - (T_1 - T_2)\right]}{\log\left[\frac{dt_\text{h}}{(T_1 - T_2)}\right]},$$

$$\Delta T \le \frac{\left[(T_1 - T_2) - dt_\text{c}\right]}{\log\left[\frac{(T_1 - T_2)}{dt_\text{c}}\right]}$$

Property 3 ([19]) Let dt_h, dt_c and ΔT, be continuous positive variables. Also, let T_1 and T_2, be positive constants such that $T_1 - T_2 > 0$. Then the following in-

equalities, which are based on the Paterson approximation [13] for the LMTD, are convex:

$$\Delta T \le \frac{1}{3}\left[\frac{(dt_h + dt_c)}{2}\right] + \frac{2}{3}\sqrt{dt_h dt_c},$$
$$\Delta T \le \frac{1}{3}\left[\frac{(dt_h + T_1 - T_2)}{2}\right] + \frac{2}{3}\sqrt{dt_h(T_1 - T_2)},$$
$$\Delta T \le \frac{1}{3}\left[\frac{(T_1 - T_2 + dt_c)}{2}\right] + \frac{2}{3}\sqrt{(T_1 - T_2)dt_c}.$$

Property 4 ([19]) Let dt_h, dt_c and ΔT, be continuous positive variables. Also, let T_1 and T_2, be positive constants such that $T_1 - T_2 > 0$. Then the following inequalities, which are based on the Chen approximation [2] for the LMTD, are convex:

$$\Delta T \le \left[\frac{(dt_h)(dt_c)(dt_h + dt_c)}{2}\right]^{\frac{1}{3}},$$
$$\Delta T \le \left[\frac{(dt_h)(T_1 - T_2)(dt_h + T_1 - T_2)}{2}\right]^{\frac{1}{3}},$$
$$\Delta T \le \left[\frac{(T_1 - T_2)(dt_c)(T_1 - T_2 + dt_c)}{2}\right]^{\frac{1}{3}}.$$

Property 5 ([19]) Let dt_h, dt_c be continuous positive variables, and let ΔT be the logarithmic mean temperature difference, $\Delta T = [dt_h - dt_c/\log[dt_h/dt_c]$. Also, assume that r is a constant determined by the ratio of two particular values of dt_h and dt_c. Then, the following bounding inequality is valid, and holds as an equality along the line determined by the ratio $r = dt_h/dt_c$:

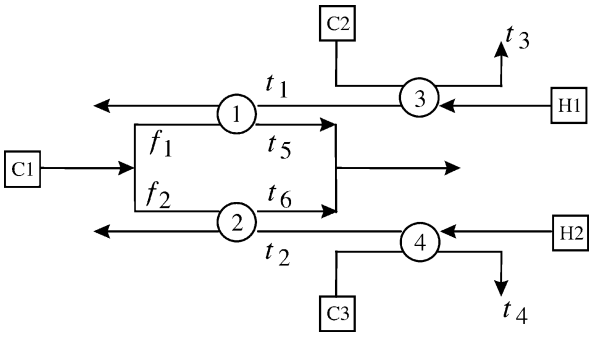
$$\Delta T \le P(r)dt_h + Q(r)dt_c,$$

where

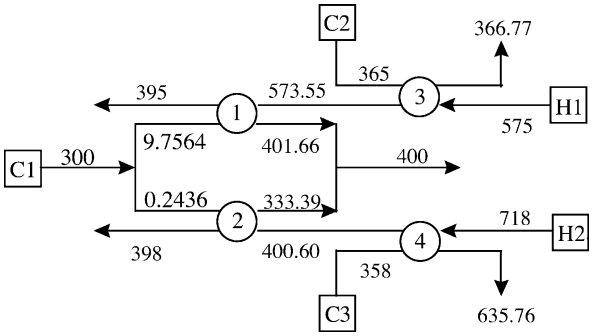
$$P(r) = \begin{cases} 0.5 & \text{if } r = 1, \\ \frac{1/r - 1 + \log(r)}{[\log(r)]^2} & \text{if } r \ne 1, \end{cases}$$
$$Q(r) = \begin{cases} 0.5 & \text{if } r = 1, \\ \frac{r - 1 - \log(r)}{[\log(r)]^2} & \text{if } r \ne 1. \end{cases}$$

Several other useful properties and their application in the development of convex relaxations for HENs problems can be found in [1,6,14,19], and [20,21,22]

As an illustrative example of the use of the above properties, and the application of global optimization techniques in heat exchanger networks, consider the



Global Optimization of Heat Exchanger Networks, Figure 2
Heat exchanger network for the illustrative problem



Global Optimization of Heat Exchanger Networks, Figure 3
Global optimum HEN design of the illustrative problem

determination of the global optimal design of the HEN shown in Fig. 2 [14]; stream data and cost information are included in Table 1. This problem was originally solved in [14] and [21] using the arithmetic mean temperature difference driving force (AMTD), and assuming isothermal mixing of process streams ($t_5 = t_6$).

Figure 3 shows the global optimum solution of the nonconvex model (P) associated with the illustrative problem. A design with a total network cost of \$36,199 is determined. Note that model (P) does not assume isothermal mixing, utilizes the approximation by Chen [2], and enforces a minimum approach temperature of 5 degrees. The global optimization of model (P) was performed with the *branch and contract algorithm* proposed in [21,23]; the convex model (R) was used in the computation of rigorous lower bounds of the total network cost. The solution process required 7 branch and bound nodes, and approximately 37 cpu seconds of a Pentium I processor running at 133Mhz. Alternative suboptimal solutions for the illustrative problem based

Global Optimization of Heat Exchanger Networks, Table 1
Problem data for illustrative example

Stream	Tin (K)	Tout (K)	F (kW K^{-1})
H1	575	395	5.555
H2	718	398	3.125
C1	300	400	10
C2	365	—	4.545
C3	358	—	3.571

Cost of Heat Exchanger 1 ($\$yr^{-1}$) = $270[A_1(m^2)]$
Cost of Heat Exchanger 2 ($\$yr^{-1}$) = $720[A_2(m^2)]$
Cost of Heat Exchanger 3 ($\$yr^{-1}$) = $240[A_3(m^2)]$
Cost of Heat Exchanger 4 ($\$yr^{-1}$) = $900[A_4(m^2)]$

$U_1 = U_1 = 0.1$ kW m^{-2} K^{-1}
$U_3 = U_4 = 1.0$ kW $m^{-2}K^{-1}$

on the rigorous LMTD include network designs with total costs of \$38,513, \$39,809, \$41,836, and \$47,681.

Nonconvex Model (P)

Indices

1, 2, 3, 4	=	index for heat exchangers
1*h*, 2*h*, 3*h*, 4*h*	=	hot side of heat exchangers
1*c*, 2*c*, 3*c*, 4*c*	=	cold side of heat exchangers

Parameters

U_1, U_2, U_3, U_4	=	overall heat transfer coefficients

Positive Variables

t	=	stream temperature
dt	=	temperature difference at end of heat exchanger
ΔT	=	approximation of the logarithmic mean temperature difference
q	=	heat transfer rate
f	=	heat capacity flowrate

Objective Function

$$\min 270\frac{q_1}{U_1\Delta T_1} + 720\frac{q_2}{U_2\Delta T_2} + 240\frac{q_3}{U_3\Delta T_3} + 900\frac{q_4}{U_4\Delta T_4}.$$

Model Constraints

$$q_1 = 5.555(t_1 - 395),$$
$$q_1 = f_1(t_5 - 300),$$
$$q_2 = 3.125(t_2 - 398),$$
$$q_2 = f_2(t_6 - 300),$$
$$q_3 = 4.545(t_3 - 365),$$
$$q_3 = 5.555(575 - t_1),$$
$$q_4 = 3.571(t_4 - 358),$$
$$q_4 = 3.125(718 - t_2),$$
$$q_1 + q_2 = 1000,$$
$$q_1 + q_3 = 999.9,$$
$$q_2 + q_4 = 1000,$$
$$f_1 + f_2 = 10,$$
$$dt_{1h} = t_1 - t_5,$$
$$dt_{1c} = 95,$$
$$dt_{2h} = t_2 - t_6,$$
$$dt_{2c} = 98,$$
$$dt_{3h} = 575 - t_3,$$
$$dt_{3c} = t_1 - 365,$$
$$dt_{4h} = 718 - t_4,$$
$$dt_{4c} = t_2 - 358,$$
$$\Delta T_1 = \left[\frac{(dt_{1h})(dt_{1c})(dt_{1h} + dt_{1c})}{2}\right]^{\frac{1}{3}},$$
$$\Delta T_2 = \left[\frac{(dt_{2h})(dt_{2c})(dt_{2h} + dt_{2c})}{2}\right]^{\frac{1}{3}},$$
$$\Delta T_3 = \left[\frac{(dt_{3h})(dt_{3c})(dt_{3h} + dt_{3c})}{2}\right]^{\frac{1}{3}},$$
$$\Delta T_4 = \left[\frac{(dt_{4h})(dt_{4c})(dt_{4h} + dt_{4c})}{2}\right]^{\frac{1}{3}},$$
$$f_1 t_5 + f_2 t_6 = 4000,$$
$$0 \le q_i^L \le q_i \le q_i^U, \quad i = 1, 2, 3, 4,$$
$$0 \le t_j^L \le t_j \le t_j^U, \quad j = 1, 2, 3, 4, 5, 6,$$
$$dt_k \ge 5, \quad k = 1h, 1c, 2h, 2c, 3h, 3c, 4h, 4c$$
$$0 \le f_1^L \le f_1 \le f_1^U, \quad 0 \le f_2^L \le f_2 \le f_2^U.$$

Convex Model (R)

Objective Function

$$\min 270\frac{[\theta_1]^2}{U_1\Delta T_1} + 720\frac{[\theta_2]^2}{U_2\Delta T_2} + 240\frac{[\theta_3]^2}{U_3\Delta T_3} + 900\frac{[\theta_4]^2}{U_4\Delta T_4}.$$

Model Constraints

$$\theta_i \geq (q_i^L)^{\frac{1}{2}} + \frac{(q_i^U)^{\frac{1}{2}} - (q_i^L)^{\frac{1}{2}}}{q_i^U - q_i^L}(q_i - q_i^L),$$

$$i = 1, 2, 3, 4,$$

$$q_1 = 5.555(t_1 - 395),$$

$$q_1 = y_{15} - 300f_1,$$

$$q_2 = 3.125(t_2 - 398),$$

$$q_2 = y_{26} - 300f_2,$$

$$q_3 = 4.545(t_3 - 365),$$

$$q_3 = 5.555(575 - t_1),$$

$$q_4 = 3.571(t_4 - 358),$$

$$q_4 = 3.125(718 - t_2),$$

$$q_1 + q_2 = 1000,$$

$$q_1 + q_3 = 999.9,$$

$$q_2 + q_4 = 1000,$$

$$f_1 + f_2 = 10,$$

$$dt_{1\mathrm{h}} = t_1 - t_5,$$

$$dt_{1\mathrm{c}} = 95,$$

$$dt_{2\mathrm{h}} = t_2 - t_6,$$

$$dt_{2\mathrm{c}} = 98,$$

$$dt_{3\mathrm{h}} = 575 - t_3,$$

$$dt_{3\mathrm{c}} = t_1 - 365,$$

$$dt_{4\mathrm{h}} = 718 - t_4,$$

$$dt_{4\mathrm{c}} = t_2 - 358,$$

$$\Delta T_1 \leq \left[\frac{(dt_{1\mathrm{h}})(dt_{1\mathrm{c}})(dt_{1\mathrm{h}} + dt_{1\mathrm{c}})}{2}\right]^{\frac{1}{3}},$$

$$\Delta T_2 \leq \left[\frac{(dt_{2\mathrm{h}})(dt_{2\mathrm{c}})(dt_{2\mathrm{h}} + dt_{2\mathrm{c}})}{2}\right]^{\frac{1}{3}},$$

$$\Delta T_3 \leq \left[\frac{(dt_{3\mathrm{h}})(dt_{3\mathrm{c}})(dt_{3\mathrm{h}} + dt_{3\mathrm{c}})}{2}\right]^{\frac{1}{3}},$$

$$\Delta T_4 \leq \left[\frac{(dt_{4\mathrm{h}})(dt_{4\mathrm{c}})(dt_{4\mathrm{h}} + dt_{4\mathrm{c}})}{2}\right]^{\frac{1}{3}},$$

$$z_{11} = t_5 - 300,$$

$$z_{22} = t_6 - 300, \quad y_{15} + y_{26} = 4000,$$

$$y_{15} \geq t_5^L f_1 + f_1^L t_5 - f_1^L t_5^L,$$

$$y_{15} \geq t_5^U f_1 + f_1^U t_5 - f_1^U t_5^U,$$

$$y_{15} \leq t_5^L f_1 + f_1^U t_5 - f_1^U t_5^L,$$

$$y_{15} \leq t_5^U f_1 + f_1^L t_5 - f_1^L t_5^U,$$

$$y_{26} \geq t_6^L f_2 + f_2^L t_6 - f_2^L t_6^L,$$

$$y_{26} \geq t_6^U f_2 + f_2^U t_6 - f_2^U t_6^U,$$

$$y_{26} \leq t_6^L f_2 + f_2^U t_6 - f_2^U t_6^L,$$

$$y_{26} \leq t_6^U f_2 + f_2^L t_6 - f_2^L t_6^U,$$

$$z_{11} \geq \frac{1}{f_1}\left(\frac{q_1 + \sqrt{q_1^L q_1^U}}{\sqrt{q_1^L} + \sqrt{q_1^U}}\right)^2,$$

$$z_{22} \geq \frac{1}{f_2}\left(\frac{q_2 + \sqrt{q_2^L q_2^U}}{\sqrt{q_2^L} + \sqrt{q_2^U}}\right)^2,$$

$$z_{11} \geq \frac{q_1}{f_1^L} + q_1^U\left(\frac{1}{f_1} - \frac{1}{f_1^L}\right),$$

$$z_{11} \geq \frac{q_1}{f_1^U} + q_1^L\left(\frac{1}{f_1} - \frac{1}{f_1^U}\right),$$

$$z_{22} \geq \frac{q_2}{f_2^L} + q_2^U\left(\frac{1}{f_2} - \frac{1}{f_2^L}\right),$$

$$z_{22} \geq \frac{q_2}{f_2^U} + q_2^L\left(\frac{1}{f_2} - \frac{1}{f_2^U}\right),$$

$$z_{11} \leq \frac{1}{f_1^L f_1^U}\left(f_1^U q_1 - q_1^L f_1 + q_1^L f_1^L\right),$$

$$z_{11} \leq \frac{1}{f_1^L f_1^U}\left(f_1^L q_1 - q_1^U f_1 + q_1^U f_1^U\right),$$

$$z_{22} \leq \frac{1}{f_2^L f_2^U}(f_2^U q_2 - q_2^L f_2 + q_2^L f_2^L),$$

$$z_{22} \leq \frac{1}{f_2^L f_2^U}\left(f_2^L q_2 - q_2^U f_2 + q_2^U f_2^U\right),$$

$$0 \leq q_i^L \leq q_i \leq q_i^U, \quad i = 1, 2, 3, 4,$$

$$0 \leq t_j^L \leq t_j \leq t_j^U, \quad j = 1, 2, 3, 4, 5, 6,$$

$$dt_k \geq 5, \quad k = 1\mathrm{h}, 1\mathrm{c}, 2\mathrm{h}, 2\mathrm{c}, 3\mathrm{h}, 3\mathrm{c}, 4\mathrm{h}, 4\mathrm{c},$$

$$0 \leq f_1^L \leq f_1 \leq f_1^U, \quad 0 \leq f_2^L \leq f_2 \leq f_2^U,$$

$$y_{15}, y_{26}, z_{11}, z_{22} \geq 0.$$

See also

- MINLP: Global Optimization with αBB
- MINLP: Heat Exchanger Network Synthesis
- MINLP: Mass and Heat Exchanger Networks
- Mixed Integer Linear Programming: Heat Exchanger Network Synthesis
- Mixed Integer Linear Programming: Mass and Heat Exchanger Networks

References

1. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP problems in process synthesis and design. Comput Chem Eng 21:S445–S450
2. Chen JJJ (1987) Letter to the Editors: Comments on improvement on a replacement for the logarithmic mean. Chem Eng Sci 42:2488–2489
3. Ciric AR, Floudas CA (1991) Heat exchanger network synthesis without decomposition. Comput Chem Eng 15:385–396
4. Floudas CA, Pardalos PM (1990) A collection of test problems for constrained global optimization algorithms. no. 455 of Lecture Notes Computer Sci Springer, Berlin
5. Gundersen T, Naess L (1988) The synthesis of cost optimal heat exchanger network synthesis, An industrial review of the state of the art. Comput Chem Eng 12:503–530
6. Hashemi-Ahmady A, Zamora JM, Gundersen T (1999) A sequential framework for optimal synthesis of industrial size heat exchanger networks. In: Proc. 2nd Conf. Process Integration, Modeling and Optimization for Energy Saving and Pollution Reduction (PRESS'99), Hungarian Chemical Soc.
7. Horst R, Pardalos PM (eds) (1995) Handbook of global optimization. Kluwer, Dordrecht
8. Horst R, Tuy H (1993) Global optimization: Deterministic approaches, 2nd edn. Springer, Berlin
9. Iyer RR, Grossmann IE (1996) Global optimization of heat exchanger networks with fixed configuration for multiperiod design. In: Grossmann IE (ed) Global Optimization in Engineering Design. Kluwer, Dordrecht
10. Jezowski J (1994) Heat exchanger network grassroot and retrofit design: The review of the state of the art - Part I: Heat exchanger network targeting and insight based methods of synthesis. Hungarian J Industr Chem 22:279–294
11. Jezowski J (1994) Heat exchanger network grassroot and retrofit design: The review of the state of the art - Part II: Heat exchanger network synthesis by mathematical methods and approaches for retrofit design. Hungarian J Industr Chem 22:295–308
12. Papoulias SA, Grossmann IE (1983) A structural optimization approach in process synthesis II. Heat recovery networks. Comput Chem Eng 7:707–721
13. Paterson WR (1984) A replacement for the logarithmic mean. Chem Eng Sci 39:1635–1636
14. Quesada I, Grossmann IE (1993) Global optimization algorithm for heat exchanger networks. Industr Eng Chem Res 32:487–499
15. Ryoo HS, Sahinidis NV (1995) Global optimization of nonconvex NLPs and MINLPs with applications in process design. Comput Chem Eng 19:551–566
16. Visweswaran V, Floudas CA (1996) Computational results for an efficient implementation of the GOP algorithm and its variants. In: Grossmann IE (ed) Global Optimization in Engineering Design. Kluwer, Dordrecht
17. Westerberg AW, Shah JV (1978) Assuring a global optimum by the use of an upper bound on the lower (dual) bound. Comput Chem Eng 2:83–92
18. Yee TF, Grossmann IE (1990) Simultaneous optimization models for heat integration-II. Heat exchanger network synthesis. Comput Chem Eng 14:1165–1184
19. Zamora JM (1997) Global optimization of nonconvex NLP and MINLP models. PhD Thesis, Dept. Chemical Engin. Carnegie-Mellon Univ.
20. Zamora JM, Grossmann IE (1997) A comprehensive global optimization approach for the synthesis of heat exchanger networks with no stream splits. Comput Chem Eng 21:S65–S70
21. Zamora JM, Grossmann IE (1998) Continuous global optimization of structured process systems models. Comput Chem Eng 22:1749–1770
22. Zamora JM, Grossmann IE (1998) A global MINLP optimization algorithm for the synthesis of heat exchanger networks with no stream splits. Comput Chem Eng 22:367–384
23. Zamora JM, Grossmann IE (1999) A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms. J Global Optim 14:217–249

Global Optimization: Hit and Run Methods

ZELDA B. ZABINSKY
Industrial Engineering University Washington, Seattle, USA

MSC2000: 90C26, 90C90

Article Outline

Keywords
Hit and Run Based Algorithms
See also
References

Keywords

Global optimization; Stochastic methods; Random search algorithms; Adaptive search; Simulated annealing; Improving hit and run; Hit and run methods; Mixed discrete-continuous global optimization; Pure random search; Pure adaptive search

The *hit and run algorithms* fall into the category of sequential random search methods (cf. also ► Random search methods), or stochastic methods. These methods can be applied to a broad class of global optimization problems. They seem especially useful for problems with black-box functions which have no known structure. These problems often involve a very large number of variables, and may include both continuous and discrete variables.

The concept of hit and run is to iteratively generate a sequence of points by taking steps of random length in randomly generated directions. R.L. Smith, in 1984 [12], showed that this method can be used to generate points within a set S that are asymptotically uniformly distributed. The hit and run method was originally applied to identifying nonredundant constraints in linear programs [1,3], and in stochastic programming [2].

Hit and run was first applied to optimization in [16], and the name *improving hit and run* (IHR) was adopted. The term 'improving' was intended to indicate that the sequence of points were improving with regard to their objective function values. The IHR algorithm couples the idea of *pure adaptive search* [8,15] with the hit and run generator to produce an easily implemented sequential random search algorithm. Pure adaptive search (PAS, see also ► Random search methods) predicts that points uniformly generated in improving level sets has, on the average, a linear number of iterations in terms of dimension. One way to approximate PAS, would be to use hit and run to generate approximately uniform points, and then select those that land in improving level sets. This is the idea behind improving hit and run.

In addition to IHR, a family of methods have been developed that are based on hit and run. Other variations include: adding an acceptance probability with a cooling schedule, varying the choice of direction, varying the length of step, and modifying the sampling method to include a mixture of continuous and discrete variables.

Hit and Run Based Algorithms

The underlying concept of hit and run based algorithms is that, if hit and run could generate a uniformly distributed point in an improving level set, then PAS predicts that we need only a linear number of such points. The point generated by just one iteration of hit and run is far from uniform and may not be in the improving set, so the number of function evaluations is not expected to be linear in dimension, but in [16] it was shown that the expected number of function evaluations for IHR on the class of elliptical programs (e. g. positive definite quadratic programs) is polynomial in dimension, $O(n^{5/2})$. The number of function evaluations includes those points that are rejected because they do not fall into the improving level set. This theoretical performance result motivates the use of hit and run for optimization. Numerical experience indicates that IHR has been especially useful in high-dimensional global optimization problems when there are many local minima embedded within a broad convex structure.

The general framework for a hit and run based optimization algorithm for solving a global optimization problem,

$$\begin{cases} \min & f(x) \\ \text{s.t.} & x \in S, \end{cases}$$

where f is a real-valued function on S, is stated below.

PROCEDURE hit and run optimization method()
    InputInstance();
    Generate an initial solution X_0;
    Set $Y_0 = f(X_0)$;
    Set $k = 0$;
    DO until stopping criterion is met;
        Generate a random direction D_k;
        Generate a random steplength λ_k;
        Evaluate candidate point $W_k = X_k + \lambda_k D_k$;
        Update the new point,
        $$X_{k+1} = \begin{cases} W_k & \text{if candidate point accepted} \\ X_k & \text{if rejected} \end{cases}$$
        Set $Y_{k+1} = \min(Y_k, f(X_{k+1}))$;
    OD;
    RETURN(Best solution found, Y_{k+1});
END hit and run optimization method;

Pseudocode for a hit and run based optimisation algorithm

Improving hit and run uses the most basic hit and run generator, which is to generate a direction vector D_k that is uniformly distributed on a hypersphere, and then generate a steplength λ_k which is generated uniformly on the intersection of D_k with the feasible set S. In many applications, S may be an n-dimensional polytope described by linear constraints, in which case the intersection of a direction with S is easily computed using a slight modification of a minimum ratio test (see [16] for details). This is the most basic hit and run generator, but several variations have been developed.

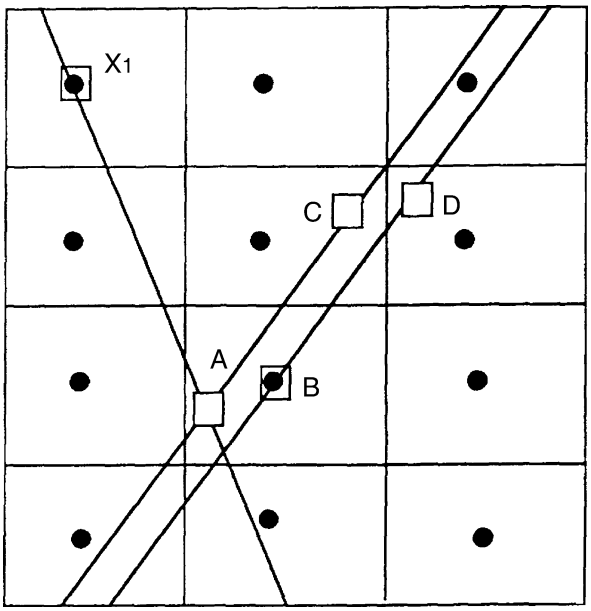
One variation is to add an acceptance probability with a cooling schedule to the *hit and run generator*, as in simulated annealing (cf. ► Simulated annealing). This was developed in [10] and called the *hide-and-seek algorithm*. Just as IHR was motivated by pure adaptive search, hide-and-seek was motivated by adaptive search [9] (see also ► Random search methods). Adaptive search generates a series of points according to a sequence of Boltzman distributions, with parameter T changing on each iteration. The theory predicts that *adaptive search* with decreasing temperature parameter T will converge with probability one to the global optimum, and the number of improving points have the same linear bound as PAS. Hide-and-seek uses the basic hit and run generator, but accepts the candidate point with the Metropolis criterion and parameter T. It is interesting to consider the two extremes of the acceptance probability: if the temperature is fixed at infinity, then all candidate points are accepted, and the hit and run generator approximates *pure random search* with a uniform distribution; at the other extreme if the temperature is fixed to zero, then only improving points are accepted, and we have *improving hit and run*. H.E. Romeijn and Smith derived a cooling schedule which essentially starts with hit and run, and approaches IHR. They proved that hide-and-seek will eventually converge to the global optimum, even though it may experience deteriorations in objective function values. They also present computational results on several test functions, which compare favorably with other algorithms in the literature.

A second variation to the basic hit and run generator is to modify the direction distribution. Thus far, we have only described choosing a direction according to a uniform distribution on an n-dimensional hypersphere, which has also been termed *hyperspherical direction* (HD) choice. In [16] and [10], the direction distribution is defined more generally; the direction may be generated from a multivariate normal distribution with mean 0 and covariance matrix H. If the H matrix is the identity matrix, then the direction distribution is essentially the uniform distribution on a hypersphere. In [4] a nonuniform direction distribution is derived that optimizes the rate of convergence of the algorithm. Although exact implementation of the optimal direction distribution may be very difficult, it motivates an adaptive direction choice rule called *artificial centering hit and run*.

Another choice for direction distribution is the *coordinate direction* (CD) method, in which the direction is chosen uniformly from the n coordinate vectors (spanning $\mathbf{R}^n$). Both HD and CD versions of direction choice were presented and applied to identifying nonredundant linear constraints in [1]. They were also tested in the context of global optimization in [14]. Computationally, CD can outperform HD on specific problems where the optimum is properly aligned, however HD is guaranteed to converge with probability one, while it is easy to construct problems where CD will never converge to the global optimum. A simple example is given in [5] where local minima are lined up on the coordinate directions, and it is impossible for the CD algorithm to leave the local minimum unless it accepts a nonimproving point. For such an example, in [5] it is shown that the CD algorithm coupled with a nonzero acceptance probability for nonimproving points will converge with probability one. Experimental results were also reported.

A third variation to the basic hit and run generator modifies it to be applicable to discrete domains [7,11]. Hit and run as described so far has been defined on a continuous domain. An extension to a discrete domain was accomplished by superimposing the discrete domain onto a continuous real number system. It was motivated by design variables such as fiber angles in a composite laminate, or diameters in a 10-bar truss, where the discrete variables have a natural continuous analog. Two slightly different modifications have been introduced.

In [11] the candidate points were generated using Hit and run on the expanded continuous domain, where the objective function of a nondiscrete point is equal to the objective function evaluated at its nearest



Global Optimization: Hit and Run Methods, 1
Two schemes to modify hit and run to disrete domains

discrete value. In this way, the modified algorithm operates on a continuous domain where the objective function is a multidimensional step function, with plateaus surrounding the discrete points. This modification still converges with probability 1 to the global optimum, as proven in [11].

The diagram in Fig. 1 illustrates this method. Starting from point X_1, hit and run on the continuous domain generates a candidate point such as A. The objective function at A is set equal to that of its nearest discrete point B, forcing $f(A) = f(B)$. If the candidate point is accepted, then $X_2 = A$, and another candidate point (shown as C) is generated.

A second scheme to modify hit and run to operate on discrete domains is to similarly generate a point on a continuous domain, and then round the generated point to its nearest discrete point in the domain on each iteration [6,7,13]. Again starting from point X_1 in Fig. 1, suppose A is generated. In this version, the candidate point is taken as the nearest discrete neighbor, in this example B. The objective function is evaluated at B, $f(B)$, and if the point is accepted, then $X_2 = B$. The difference in this variation is illustrated by noting that the next candidate point is generated from B instead of from A, see point D in Fig. 1. Also note that only discrete points are maintained. In [6,7] it is shown that this second scheme dominates the first scheme in terms of average performance for the special class of spherical programs, and numerical results have been promising.

Another modification to the basic hit and run generator is in the way the steplength is generated. Instead of generating the point uniformly on the whole line segment, the line segment can be restricted to a fixed length, or adaptively modified. S. Neogi [6] refers to this as full-line length, restricted line length, or adaptive stepsize. In [6] the adaptive stepsize is coupled with an acceptance probability to maintain a fixed probability of generating an improving point. See [6] for a more detailed discussion of this variation of a *simulated annealing* algorithm based on the *hit and run generator*.

The many variations of hit and run have been numerically tested on many test functions and applied to real applications. All of the papers referenced in this article include numerical results, but the details are left to the individual papers. Overall, the theoretical motivations and numerical experience leads us to believe that hit and run is a promising approach to global optimization.

See also

- ► Random Search Methods
- ► Stochastic Global Optimization: Stopping Rules
- ► Stochastic Global Optimization: Two-phase Methods

References

1. Berbee HCP, Boender CGE, Rinnooy Kan AHG, Scheffer CL, Smith RL, Telgen J (1987) Hit-and-run algorithms for the identification of nonredundant linear inequalities. Math Program 37:184–207
2. Birge JR, Smith RL (1984) Random procedures for nonredundant constraint identification in stochastic linear programs. Amer J Math Management Sci 4:41–70
3. Boneh A, Golan A (1979) Constraints' redundancy and feasible region boundedness by random feasible point generator. Third European Congress Oper. Res., EURO III, Amsterdam, 9-11 April 1979
4. Kaufman DE, Smith RL (1998) Direction choice for accelerated convergence in hit-and-run sampling. Oper Res 46(1):84–95
5. Kristinsdottir BP (1997) Analysis and development of random search algorithms. PhD Thesis Univ. Washington
6. Neogi S (1997) Design of large composite structures using global optimization and finite element analysis. PhD Thesis Univ. Washington

7. Neogi S, Zabinsky ZB, Tuttle ME (1994) Optimal design of composites using mixed discrete and continuous variables. Proc. ASME Winter Annual Meeting, Symp. Processing, Design and Performance of Composite Materials, vol 52. Dekker, New York, pp 91–107
8. Patel NR, Smith RL, Zabinsky ZB (1988) Pure adaptive search in Monte Carlo optimization. Math Program 4:317–328
9. Romeijn HE, Smith RL (1994) Simulated annealing and adaptive search in global optimization. Probab Eng Inform Sci 8:571–590
10. Romeijn HE, Smith RL (1994) Simulated annealing for constrained global optimization. J Global Optim 5:101–126
11. Romeijn HE, Zabinsky ZB, Graesser DL, Neogi S (1999) Simulated annealing for mixed integer/continuous global optimization. J Optim Th Appl 101(1)
12. Smith RL (1984) Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. Oper Res 32:1296–1308
13. Zabinsky ZB (1998) Stochastic methods for practical global optimization. J Global Optim 13:433–444
14. Zabinsky ZB, Graesser DL, Tuttle ME, Kim GI (1992) Global optimization of composite laminate using improving hit and run. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton Univ. Press, Princeton, 343–365
15. Zabinsky ZB, Smith RL (1992) Pure adaptive search in global optimization. Math Program 53:323–338
16. Zabinsky ZB, Smith RL, McDonald JF, Romeijn HE, Kaufman DE (1993) Improving hit and run for global optimization. J Global Optim 3:171–192

Global Optimization: Interval Analysis and Balanced Interval Arithmetic

JULIUS ŽILINSKAS[1], IAN DAVID LOCKHART BOGLE[2]
[1] Institute of Mathematics and Informatics, Vilnius, Lithuania
[2] Centre for Process Systems Engineering, Department of Chemical Engineering, University College London, London, UK

MSC2000: 65K05, 90C30, 90C57, 65G30, 65G40

Article Outline

Keywords and Phrases
Introduction
Methods / Applications
Interval Analysis in Global Optimization
Underestimating Interval Arithmetic
Random Interval Arithmetic
Balanced Interval Arithmetic
See also
References

Keywords and Phrases

Global optimization; Interval arithmetic; Interval computations

Introduction

Mathematically the global optimization problem is formulated as

$$f^* = \min_{X \in D} f(X)\,,$$

where a nonlinear function $f(X)$, $f : \mathbb{R}^n \to \mathbb{R}$, of continuous *variables* X, is an *objective function*; $D \in \mathbb{R}^n$ is a *feasible region*; n is a *number of variables*. A *global minimum* f^* and one or all *global minimizers* X^*:

$$f(X^*) = f^*$$

should be found. No assumptions on unimodality are included in the formulation of the problem. Most often an objective function is defined by an analytical formula or an algorithm, which evaluates the value of the objective function using the values of variables and arithmetic operations. ► Continuous global optimization: models, algorithms and software.

One of the classes of methods for global optimization are methods based on *interval arithmetic*. Interval arithmetic [10] provides bounds for the function values over hyper-rectangular regions defined by intervals of variables. The bounds may be used in global optimization to detect the sub-regions of the feasible region which cannot contain a global minimizer. Such subregions may be discarded from the subsequent search for a minimum.

Interval arithmetic provides guaranteed bounds but sometimes they are too pessimistic. Interval arithmetic is used in global optimization to provide guaranteed solutions, but there are problems for which the time for optimization is too long. A disadvantage of interval arithmetic is the dependency problem [5]: when a given variable occurs more than once in interval computation, it is treated as a different variable in each occur-

rence. This causes widening of computed intervals and overestimation of the range of function values.

Analysis of both overestimating and underestimating intervals is useful to estimate how much interval bounds overestimate the range of function values. Moreover inner interval arithmetic operations may be used instead of standard interval arithmetic operations in some cases when dependency of operands is known or operands are known to be monotonic. Although monotonicity cannot easily be determined in advance, inner and standard interval arithmetic operations may be chosen randomly building random interval arithmetic, estimating the range of real function values from a sample of random intervals.

Methods / Applications

Interval Analysis in Global Optimization

Interval arithmetic is proposed in [10]. Interval arithmetic operates with real intervals $\overline{\underline{x}} = \left[\underline{x}, \overline{x}\right] = \{x \in \mathbb{R} | \underline{x} \le x \le \overline{x}\}$, defined by two real numbers $\underline{x} \in \mathbb{R}$ and $\overline{x} \in \mathbb{R}$, $\underline{x} \le \overline{x}$. For any real arithmetic operation $x \circ y$ the corresponding interval arithmetic operation $\overline{\underline{x}} \circ \overline{\underline{y}}$ is defined as an operation whose result is an interval containing every possible number produced by the real operation with the real numbers from each interval. The interval arithmetic operations are defined as:

$$\overline{\underline{x}} + \overline{\underline{y}} = \left[\underline{x} + \underline{y}, \overline{x} + \overline{y}\right],$$

$$\overline{\underline{x}} - \overline{\underline{y}} = \left[\underline{x} - \overline{y}, \overline{x} - \underline{y}\right],$$

$$\overline{\underline{x}} \times \overline{\underline{y}} = \begin{cases} \left[\underline{x}\,\underline{y}, \overline{x}\,\overline{y}\right], & \underline{x} > 0\,, \underline{y} > 0\,, \\ \left[\overline{x}\,\underline{y}, \overline{x}\,\overline{y}\right], & \underline{x} > 0\,, 0 \in \overline{\underline{y}}\,, \\ \left[\overline{x}\underline{y}, \underline{x}\,\overline{y}\right], & \underline{x} > 0\,, \overline{y} < 0\,, \\ \left[\underline{x}\,\overline{y}, \overline{x}\,\overline{y}\right], & 0 \in \overline{\underline{x}}\,, \underline{y} > 0\,, \\ \left[\begin{array}{l}\min(\underline{x}\overline{y}\,, \overline{x}\underline{y})\,, \\ \quad \max(\underline{x}\underline{y}, \overline{x}\overline{y})\end{array}\right], & 0 \in \overline{\underline{x}}\,, 0 \in \overline{\underline{y}}\,, \\ \left[\overline{x}\underline{y}, \underline{x}\underline{y}\right], & 0 \in \overline{\underline{x}}\,, \overline{y} < 0\,, \\ \left[\underline{x}\overline{y}, \overline{x}\underline{y}\right], & \underline{x} < 0\,, \underline{y} > 0\,, \\ \left[\underline{x}\overline{y}, \underline{x}\underline{y}\right], & \underline{x} < 0\,, 0 \in \overline{\underline{y}}\,, \\ \left[\overline{x}\overline{y}, \underline{x}\underline{y}\right], & \underline{x} < 0\,, \overline{y} < 0\,, \end{cases}$$

$$\overline{\underline{x}} / \overline{\underline{y}} = \begin{cases} \left[\underline{x}/\overline{y}, \overline{x}/\underline{y}\right], & \underline{x} > 0\,, \underline{y} > 0\,, \\ \left[\overline{x}/\overline{y}, \underline{x}/\underline{y}\right], & \underline{x} > 0\,, \overline{y} < 0\,, \\ \left[\underline{x}/\underline{y}, \overline{x}/\underline{y}\right], & 0 \in \overline{\underline{x}}\,, \underline{y} > 0\,, \\ \left[\overline{x}/\overline{y}, \underline{x}/\overline{y}\right], & 0 \in \overline{\underline{x}}\,, \overline{y} < 0\,, \\ \left[\underline{x}/\underline{y}, \overline{x}/\overline{y}\right], & \underline{x} < 0\,, \underline{y} > 0\,, \\ \left[\overline{x}/\underline{y}, \underline{x}/\overline{y}\right], & \underline{x} < 0\,, \overline{y} < 0\,. \end{cases}$$

An interval function can be constructed replacing the usual arithmetic operations by interval arithmetic operations in the formula or the algorithm for calculating values of the function. An interval value of the function can be evaluated using the interval function with interval arguments. The resulting interval always encloses the range of real function values in the hyperrectangular region defined by the vector of interval arguments:

$$\left\{f(X) \,|\, X \in \overline{\underline{X}}, \underline{X} \in \mathbb{R}^n\,, \overline{X} \in \mathbb{R}^n\right\} \subseteq \underline{\overline{f}}\left(\overline{\underline{X}}\right)\,,$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $\underline{\overline{f}} : [\mathbb{R}, \mathbb{R}]^n \to [\mathbb{R}, \mathbb{R}]$. Because of this property the interval value of the function can be used as the lower and upper bounds for the function in the region which may be used in global optimization.

The first version of interval global optimization algorithm was oriented to minimization of a rational function by bisection of sub-domains [12]. Interval methods for global optimization were further developed in [3,4,11], where the interval Newton method and the test of strict monotonicity were introduced. A thorough description including theoretical as well as practical aspects can be found in [5] where a very efficient interval global optimization method involving monotonicity and non-convexity tests and the special interval Newton method is described. ► Interval global optimization.

A *branch and bound* technique is usually used to construct interval global optimization algorithms. An iteration of a classical branch and bound algorithm processes a yet unexplored sub-region of the feasible region. Iterations have three main components: selection of the sub-region from a candidate list to process, bound calculation, and branching. In interval global optimization algorithms bounds are calculated using interval arithmetic. All interval global opti-

mization branch and bound algorithms use the hyper-rectangular partitions and branching is usually performed bisecting the hyper-rectangle into two. Variants of interval branch-and-bound algorithms for global optimization where the bisection was substituted by the subdivision of subregions into many subregions in a single iteration step have been investigated in [2]. The convergence properties have been investigated in detail. An extensive numerical study is presented in [8]. ► Bisection global optimization methods; ► Interval analysis: Subdivision directions in interval branch and bound techniques.

The tightness of bounds is a very important factor for efficiency of branch and bound based global optimization algorithms. An experimental model of interval arithmetic with controllable tightness of bounds to investigate the impact of bound tightening in interval global optimization was proposed in [14]. The experimental results on efficiency of tightening bounds were presented for several test and practical problems. Experiments have shown that the relative tightness of bounds strongly influences efficiency of global optimization algorithms based on the branch and bound approach combined with interval arithmetic.

Underestimating Interval Arithmetic

Kaucher arithmetic [6,7] defining underestimates is useful to estimate how much interval bounds overestimate the range of function values. Kaucher arithmetic operations ($\circ_{\mathrm{u}}$) are defined as:

$$\underline{x} +_{\mathrm{u}} \underline{y} = \left[\underline{x} + \overline{y} \vee \overline{x} + \underline{y}\right],$$

$$\underline{x} -_{\mathrm{u}} \underline{y} = \left[\underline{x} - \underline{y} \vee \overline{x} - \overline{y}\right],$$

$$\underline{x} \times_{\mathrm{u}} \underline{y} = \begin{cases} \left[\underline{x}\,\overline{y} \vee \overline{x}\underline{y}\right], & \overline{x} > 0\,, \overline{y} > 0 \\ & \text{or } \overline{x} < 0\,, \overline{y} < 0\,, \\ \left[\underline{x}\,\underline{y}, \underline{x}\,\overline{y}\right], & \overline{x} > 0\,, 0 \in \overline{y}\,, \\ \left[\overline{x}\underline{y} \vee \underline{x}\,\underline{y}\right], & \overline{x} > 0\,, \overline{y} < 0 \\ & \text{or } \overline{x} < 0\,, \overline{y} > 0\,, \\ \left[\underline{x}\,\underline{y}, \overline{x}\underline{y}\right], & 0 \in \overline{x}\,, \overline{y} > 0\,, \\ [0,0]\,, & 0 \in \overline{x}\,, 0 \in \overline{y}\,, \\ \left[\overline{x}\underline{y}, \underline{x}\,\overline{y}\right], & 0 \in \overline{x}\,, \overline{y} < 0\,, \\ \left[\overline{x}\underline{y}, \overline{x}\underline{y}\right], & \overline{x} < 0\,, 0 \in \overline{y}\,, \end{cases}$$

$$\underline{x} /_{\mathrm{u}} \underline{y} = \begin{cases} \left[\underline{x}/\underline{y} \vee \overline{x}/\overline{y}\right], & \overline{x} > 0\,, \overline{y} > 0 \\ & \text{or } \overline{x} < 0\,, \overline{y} < 0\,, \\ \left[\overline{x}/\underline{y} \vee \underline{x}/\overline{y}\right], & \overline{x} > 0\,, \overline{y} < 0 \\ & \text{or } \overline{x} < 0\,, \overline{y} > 0\,, \\ \left[\underline{x}/\overline{y}, \overline{x}/\overline{y}\right], & 0 \in \overline{x}\,, \overline{y} > 0\,, \\ \left[\overline{x}/\underline{y}, \underline{x}/\underline{y}\right], & 0 \in \overline{x}\,, \overline{y} < 0\,, \end{cases}$$

where $[a \vee b] = [\min(a,b), \max(a,b)]$. Underestimating interval arithmetic guarantees to underestimate:

$$\underline{f}_{\mathrm{u}}\left(\underline{X}\right) \subseteq \left\{f(X) \,|X \in \underline{X},\right\} \subseteq \underline{f}\left(\underline{X}\right).$$

An interval defined by Kaucher arithmetic is a worst case estimate and can be the degenerate interval $[0,0]$. A regularized version of Kaucher arithmetic proposed in [13] assumes regularity of the dependency between variables. In the underestimation assuming regularity of the dependency between variables, multiplication operation ($\times_{\mathrm{ur}}$) is defined differently from Kaucher arithmetic:

$$\underline{x} +_{\mathrm{ur}} \underline{y} = \underline{x} +_{\mathrm{u}} \underline{y}\,,$$

$$\underline{x} -_{\mathrm{ur}} \underline{y} = \underline{x} -_{\mathrm{u}} \underline{y}\,,$$

$$\underline{x} \times_{\mathrm{ur}} \underline{y} = \begin{cases} \left[\min(\underline{x}\,\overline{y}, \overline{x}\underline{y}), \mu(\underline{x}, \overline{x}, \underline{y}, \overline{y})\right], \\ \qquad \overline{x} > 0\,, \overline{y} > 0 \;\text{ or }\; \overline{x} < 0\,, \overline{y} < 0\,, \\ \left[\mu(\underline{x}, \overline{x}, \overline{y}, \underline{y}), \max(\underline{x}\,\underline{y}, \overline{x}\overline{y})\right], \\ \qquad \overline{x} > 0\,, \overline{y} < 0 \;\text{ or }\; \overline{x} < 0\,, \overline{y} > 0\,, \\ \left[\mu(\underline{x}, \overline{x}, \overline{y}, \underline{y}), \mu(\underline{x}, \overline{x}, \underline{y}, \overline{y})\right], \\ \qquad \text{otherwise}\,, \end{cases}$$

$$\underline{x} /_{\mathrm{ur}} \underline{y} = \underline{x} /_{\mathrm{u}} \underline{y}\,,$$

where

$$\mu(x_1, x_2, y_1, y_2) = \begin{cases} x_2 y_1\,, & \frac{(x_2 - x_1)y_2 - x_1(y_2 - y_1)}{2(x_2 - x_1)(y_2 - y_1)} > 1\,, \\ x_1 y_2\,, & \frac{(x_2 - x_1)y_2 - x_1(y_2 - y_1)}{2(x_2 - x_1)(y_2 - y_1)} < 0\,, \\ \frac{(x_2 y_2 - x_1 y_1)^2}{4(x_2 - x_1)(y_2 - y_1)}\,, & \text{otherwise}\,. \end{cases}$$

In [1,9] *inner interval arithmetic* is defined. If the operands in the interval operations to calculate the function values are known to be monotonic then standard interval arithmetic operations may be combined with inner interval operations to tighten resulting intervals without losing the guarantee of enclosure [1]. If it is known that operands in subtraction or division are dependent or are monotonic and have the same monotonicity (either both are monotonically increasing or

both monotonically decreasing) then inner interval operations may be used instead of standard interval operations. If it is known that operands in summation or multiplication are monotonic and do not have the same monotonicity (one is monotonically increasing and another is monotonically decreasing) then inner interval operations may be used instead of standard interval operations.

The difference between inner interval operations ($\circ_{\text{in}}$) and underestimating interval operations ($\circ_{\text{u}}$) concerns the result of multiplication:

$$\underline{x} +_{\text{in}} \underline{y} = \underline{x} +_{\text{u}} \underline{y} \,,$$
$$\underline{x} -_{\text{in}} \underline{y} = \underline{x} -_{\text{u}} \underline{y} \,,$$
$$\underline{x} \times_{\text{in}} \underline{y} = \begin{cases} \left[\max(\underline{x}\,\underline{y}, \overline{x}\,\underline{y}), \min(\underline{x}\,\underline{y}, \overline{x}\,\overline{y})\right] \,, & \\ & 0 \in \underline{x}, 0 \in \underline{y}, \\ \underline{x} \times_{\text{u}} \underline{y}, & \text{otherwise} \,, \end{cases}$$
$$\underline{x} \,/_{\text{in}}\, \underline{y} = \underline{x} \,/_{\text{u}}\, \underline{y} \,.$$

Random Interval Arithmetic

It is difficult computationally to find which operands are dependent, to be certain they are monotonic, and to determine their monotonicity (intervals of the derivatives of all operands need to be found). *Random interval arithmetic* proposed in [1] is obtained by choosing standard or inner interval operations *randomly with the same probability* at each step of the computation. The range of function values is estimated using a number of sample intervals evaluated using random interval arithmetic. The estimation is based on the assumptions that the distribution of the centres of the evaluated intervals is normal with a very small relative standard deviation and the distribution of the radii is normal but taking only positive values. The mean value of the centres μ_{centres}, the mean value of the radii μ_{radii} and the standard deviations of the radii σ_{radii} of the random intervals are used to evaluate an approximate range of the function

$$[\mu_{\text{centres}} \pm (\mu_{\text{radii}} + \alpha\sigma_{\text{radii}})] \,, \qquad (1)$$

where α is between 1 and 3 depending on the number of samples and the desired probability that the exact range is included in the estimated range. It is suggested in [1] that a compromise between efficiency and robustness can be obtained using $\alpha = 1.5$ and 30 samples. Experimental results presented in [1] for some functions over small intervals show that random interval arithmetic provides tight estimates of the ranges of the considered function values with probability close to 1. However, in the experiments, the intervals of variables of the function considered were small. In the case of large intervals of variables, and particularly for multi-variable functions, the obtained estimates for a range of function values frequently do not fully enclose the range of function values.

For the application of random interval arithmetic to global optimization it is important to extend these ideas to the case of functions defined over large multidimensional regions. *Balanced random interval arithmetic* proposed in [16] extending the ideas of [1], is obtained by choosing standard and inner interval operations at each step of the computation *randomly with predefined probabilities* for the standard and inner operations. A number of sample intervals are evaluated. It is assumed that the distribution of centres of the evaluated balanced random intervals is normal and that the distribution of radii is folded normal, also known as absolute normal, because the radii cannot be negative. The range of values of the function in the defined region is estimated using the mean values (μ) and the standard deviations (σ) of centres and radii of the evaluated balanced random intervals:

$$[\mu_{\text{centres}} \pm (3.0\sigma_{\text{centres}} + \mu_{\text{radii}} + 3.0\sigma_{\text{radii}})] \,. \qquad (2)$$

The ranges of values of the objective function estimated using balanced random interval arithmetic can be used in the general branch and bound framework building a stochastic global optimization algorithm. The performance of such an algorithm has been evaluated experimentally on market model estimation [17] and on chemical engineering problems. When speed of optimization is more important than guaranteed reliability, such an algorithm is a good alternative to the algorithm with standard interval arithmetic because it is several times faster.

Balanced Interval Arithmetic

The exact range of function values lies between the results of overestimating and underestimating interval arithmetic. Estimates of the ranges of function values estimated from the results of standard interval arith-

metic and inner interval arithmetic were investigated in [15]. There, *balanced interval arithmetic* is defined as the weighted mean of the overestimating and underestimating intervals of the function:

$$pc \times \overline{\underline{f}}\left(\overline{\underline{X}}\right) + (1 - pc) \times \overline{\underline{f}}_{\text{u}}\left(\overline{\underline{X}}\right) , \qquad (3)$$

where the predefined coefficient $0 \leq pc \leq 1$ defines the balance between overestimating and underestimating intervals.

The ranges of the values of several functions estimated using balanced interval arithmetic and using balanced random interval arithmetic have been experimentally compared [15]. The results of the experiments have shown that ranges estimated using balanced interval arithmetic compete with ranges estimated using balanced random interval arithmetic. However balanced interval arithmetic is not based on the assumptions of normal distributions and does not require several samples.

The ranges of values of the objective function estimated using balanced interval arithmetic can be used in the general branch and bound framework building a deterministic global optimization algorithm. When the predefined coefficient pc is less than 1, the algorithm may be faster than the algorithm with standard interval arithmetic.

For each interval function, there exists α, $0 \leq \alpha \leq 1$, for which

$$\left\{ f(X) \,|X \in \overline{\underline{X}}, \right\} \subseteq \alpha \times \overline{\underline{f}}\left(\overline{\underline{X}}\right) + (1 - \alpha) \times \overline{\underline{f}}_{\text{u}}\left(\overline{\underline{X}}\right)$$

for all possible sub-regions of the feasible region, $\overline{\underline{X}} \subseteq D$. The algorithm guarantees the exact solution if $pc \geq \alpha$.

See also

- Bisection Global Optimization Methods
- Continuous Global Optimization: Models, Algorithms and Software
- Interval Analysis: Parallel Methods for Global Optimization
- Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- Interval Analysis: Unconstrained and Constrained Optimization
- Interval Global Optimization
- Interval Linear Systems

References

1. Alt R, Lamotte JL (2001) Experiments on the evaluation of functional ranges using random interval arithmetic. Math Comput Simul 56:17–34
2. Csallner AE, Csendes T, Markót MC (2000) Multisection in interval branch-and-bound methods for global optimization I. Theoretical results. J Glob Optim 16:371–392
3. Hansen E (1978) A globally convergent interval method for computing and bounding real roots. BIT 18:415–424
4. Hansen E (1978) Global optimization using interval analysis – the multidimensional case. Numer Math 34:247–270
5. Hansen E, Walster G (2003) Global Optimization Using Interval Analysis, 2nd edn. Marcel Dekker, New York
6. Kaucher E (1977) Über Eigenschaften und Anwendungsmöglichkeiten der erweiterten Intervallrechnung und des hyperbolischen Fastkörpers über **R**′. Comput Suppl 1:81–94
7. Kreinovich V, Nesterov VM, Zheludeva NA (1996) Interval methods that are guaranteed to underestimate (and the resulting new justification of Kaucher arithmetic). Reliab Comput 2:119–124
8. Markót MC, Csendes T, Csallner AE (2000) Multisection in interval branch-and-bound methods for global optimization II. Numerical tests. J Global Optim 16:219–228
9. Markov S (1995) On directed interval arithmetic and its applications. J Univers Comput Sci 1:514–526
10. Moore RE (1966) Interval Analysis. Prentice-Hall, Englewood Cliffs
11. Moore RE (1977) A test for existence of solutions to nonlinear systems. SIAM J Numer Anal 14:611–615
12. Skelboe S (1974) Computation of rational interval functions. BIT 14:87–95
13. Žilinskas A, Žilinskas J (2005) On underestimating in interval computations. BIT Numer Math 45:415–427
14. Žilinskas A, Žilinskas J (2006) On efficiency of tightening bounds in interval global optimization. Lect Note Comput Sci 3732:197–205
15. Žilinskas J (2006) Estimation of functional ranges using standard and inner interval arithmetic. Inform 17:125–136
16. Žilinskas J, Bogle IDL (2004) Balanced random interval arithmetic. Comput Chem Eng 28:839–851
17. Žilinskas J, Bogle IDL (2006) Balanced random interval arithmetic in market model estimation. Eur J Oper Res 175:1367–1378

Global Optimization in Lennard–Jones and Morse Clusters

COSTAS D. MARANAS
Pennsylvania State University, University Park, USA

MSC2000: 90C26, 90C90

Article Outline

Keywords
See also
References

Keywords

Lennard–Jones microcluster; Morse microcluster; Minimum potential energy; Global optimization

Microclusters are [11] aggregates of atoms, ions, or molecules, sufficiently small that a significant proportion of these units is present on their surfaces. They correspond to systems that are neither single entities nor continua composed by an infinite number of units, but lie somewhere in between bridging the gap between single atoms or molecules and bulk matter. Typically, microclusters consist of two to several hundred atoms. A key word pertaining to the novel features of microclusters is *size effects* [26]. The microscopic size of microclusters gives rise to unique properties in two ways. First, a large percentage of a cluster's atoms are on or close to the surface, and surface atoms do not arrange themselves in the same way as do atoms in bulk matter, but instead they tend to avoid being exposed on the surface. Assuming a spherical shape, the fraction of the number of surface atoms is $4/n^{1/3}$. For $n = 10^2$ this number is 86%, for $n = 10^3$ is 40% and for $n = 10^4$ is still 20%. For example, in a cluster of 55 argon atoms at least 42 atoms are on the surface in some sense. This effect completely overwhelmsthe tendency of atoms to arrange themselves in a regular crystalline array as they normally do in bulk matter. For instance, the ordering of silicon atoms in the Si_{10} cluster is completely different from the ordering in the silicon crystalline structure. It appears that clusters consisting of specific numbers of atoms are extremely stable, as they show up more prominently in the mass spectrum than neighboring cluster sizes. These numbers of particles that enhance stability are called *magic numbers* and they are substance specific [2]. For instance [3], xenium clusters consisting of $N = 13, 19, 23, 25, \ldots$ are particularly stable, although for sodium clusters the magic numbers are $N = 8, 20, 40, 58, 92, \ldots$.

The study of the topography of the potential energy function of a microcluster in the internal configurational space was and still remains a central problem in this area of research [11,13]. This problemcan be succinctly stated as follows: Given N particles interacting with two-body central forces, find their configuration(s) in the three-dimensional Euclidean space involving the global minimum total potential energy.

This can be expressed mathematically as follows:

$$V = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} v(r_{ij}),$$

where

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2},$$

$$x_1 = y_1 = z_1 = y_2 = z_2 = z_3 = 0.$$

Here, V is the total potential energy of the microcluster as the summation of all two-body interaction terms, $v(r_{ij})$ is the potential energy term corresponding to the interaction of particle i with particle j, and r_{ij} is the Euclidean distance between i and j. Note that in the double summation, j spans from $i + 1$ to N so that we avoid double counting pair interactions and the interaction of a particle with itself. Furthermore, by specifying $x_1 = y_1 = z_1 = 0$, we fix the first particle at (0, 0, 0) eliminating all three translational degrees of freedom of the microcluster. By further imposing $y_2 = z_2 = z_3 = 0$ we eliminate the rotational degrees of freedom as well. Pair potentials that have been used in cluster studies include the following [11]:

1) $v(r) = (n - m)^{-1} [nr^{-m} - mr^{-n}]$ (Mie);
2) $v(r) = 4\epsilon\{(\sigma/r)^{12} - (\sigma/r)^6\}$ (Lennard–Jones);
3) $v(r) = [1 - e^{a(1-r)}]^2 - 1$ (Morse);
4) $v(r) = Ae^{-ar^2} - Be^{-br^2}$ (Gaussian);
5) $v(r) = z^{\alpha} z^{\beta}/r + Ae^{-r/\rho}$ (Born–Meyer);

Lennard–Jones and Morse potential models are the most popular selections to describe the force field.

Even under simplifying assumptions about the interaction energy, the minimization of the total potential energy is very difficult to solve because it corresponds to a nonconvex optimization problem involving numerous local minima. Hoare [11] claimed that the number of local minima of an n—atom microcluster grows as $\exp(n^2)$. In fact, L.T. Wille [34] has shown that the complexity of determining the global minimum energy of a cluster of particles interacting via two-body forces belongs to the class *NP*. In other words, there is no known algorithm that can solve this problem in nonexponential time [22]. A geometrical, possibly topological

proof that a local minimum is both unique and global is not likely to be found because there still exist unsolved problems in the theory of sphere packings where difficulties are without any doubt less acute [4,5,6,10], than those in the minimization problem at hand.

Existing methods use physical intuition, approximation procedures, mimicking of physical phenomena, random searches, lattice optimization/relaxation, or local/global optimization approaches. M.R. Hoare, in a series of papers [12,13,14,15,16], proposed a method of finding minima of the total potential function of an $5 \leq N \leq 66$ particle Lennard–Jones cluster based on a *growth scheme* involving the following steps: First, a particular compact *seed structure* involving a small number of atoms is selected which is likely to appear in the N-particle structure. At each iteration an extra particle is placed at all packing vertices and the resulting structures are tested for geometrical uniqueness. The distinct structures are then relaxed and a local optimization procedure locates and records the local minima involved. Each of the minima then serve as a new seed structure in repetition of the procedure. Finally, all of the generated distinct local minima are tabulated in decreasing order of binding energies. A number of 'growth rules' are incorporated in the procedure that alleviates the computational effort. Using this method, Hoare generated a large number of local minima for structures from 5 to 66 particles. However, no claim for complete enumeration of all local minima, and thus detection of the global minimum, can be made. In fact, it has been reported [32] that solutions of low-symmetry are not likely to be found with this method.

Piela's method [25] is based on the simple idea of smoothly deforming the potential energy hypersurface [29], in such a way as to make shallow potential wells disappear gradually, while the deeper ones grow at their expense. As the potential wells evolve they change their position and size. One then eventually ends up with a single potential well that has absorbed all the others which hopefully corresponds to the global minimum. A local optimization procedure then can easily find the single local minimum corresponding to the global one as well. The hypersurface is deformed using the diffusion equation, with the original shape of the hypersurface representing the initial concentration distribution. The main advantage of this method is that you do not have to explore the myriads of local optima, nor do you have to know their position beforehand. However, the approach depends on the conjecture that shallow potential wells disappear faster than deeper ones. In fact, it has been observed that when the global minimum lies on a narrow potential well of large depth, it might disappear faster than a wider, originally shallower, potential well.

Simulated annealing [18] variations has been widely used either alone, or in conjunction with some other method(s). A large number of researchers have been using this method for finding the global minimum of the potential energy function. Wille [32,33] solved the potential minimization problem for up to 25 particles, interacting under two-body Lennard–Jones forces and he found two new minima for $N = 24$ that were better than the one reported in [11]. P. Ballone and P. Milani [1] using a semi-empirical many-body potential, solved for the ground-geometries of carbon clusters in the range $50 \leq M \leq 72$ and found that all the structures of low energies are hollow spheres with nearly graphitic atomic arrangement. D. Hohl and R.O. Jones [17] applied the same methodology also to phosphorus clusters P_2 to P_8, arriving to arather counterintuitive most stable structure for P_8. In [23] a combined simulated annealing and a quasi-Newton-like conjugate-gradient method is used for determining the structure of mixed argon-xenon clusters interactingwith two-body Lennard–Jones forces. In [30,31] the binding energy of Nickel Lennard–Jones clusters is studied using the simulated annealing method in a canonical ensemble Monte-Carlo technique. The simulated annealing method can be viewed as a method for stochastically tracing the annealing process by Monte-Carlo simulation. D. Shalloway [27,28] presented a deterministic method for annealing the objective function by tracing the evolution of a multiple-Gaussian-packet approximation and using notions from renormalization group theory. This method has been applied to microcluster conformation problems and it appears that in most of the test problems was able to identify the global minimum.

Lattice optimization techniques have been very efficient in generating structures involving the lowest known potential energy. In [7] it is proposed that the most energetically favored microclusters in the range $20 \leq N \leq 50$ are the onesthat involve interpenetrating icosahedra (polyicosahedra) or (PIC). For $N \leq 55$

a double icosahedral (DIC) growth scheme was introduced [8] and for $55 \leq N \leq 147$ [9] a third layer icosahedral structure using two different surface arrangements was presented. Using these notions, J.A. Northby [24] derived optimal configurations for Lennard–Jones microclusters in the range $13 \leq N \leq 147$ based on a lattice optimization/relaxation algorithm. First a heuristic procedure is employed for finding a set of lattice local minimizers assuming icosahedral- (IC) or face-centered (FC) arrangements. Then, the currently best lattice minimizers are relaxed by using a local optimization algorithm. G.L. Xue [35] improved on Northby's method [24] by reducing the time complexity of the algorithm. Furthermore, by relaxing every lattice local minimizer a number of better optimal configurations were found in the range $13 \leq N \leq 147$. However, it appeared that the best local lattice does not always relax to the structure involving thelowest total Lennard–Jones potential energy. A parallel implementation [19] allowed results on minimum energies for clusters of up to $N = 1{,}000$ atoms. Also by employing a parallel version of a two-level simulated annealing algorithm [36,37,38] solutions for clustersizes as large as $N = 100{,}000$ have been reported.

C.D. Maranas and C.A. Floudas [20,21] introduced deterministic global optimization to the microcluster minimum potential energy problem. It was shown that the problem is convex only if both the first and second derivatives of the pairwise potential energy model with respect to the Euclidean distance are positive. This left only a narrow convex envelope for both Lennard–Jones and Morse potential energy models. To widen this envelope, the sum of squares of all Cartesian coordinates multiplied by a positive parameter α were added to the original objective function. It was shown that there exists a value for α such that the augmented objective function is convex. An upper bound for this value was identified. Based on these developments a branch and bound algorithm was devised based on the convex lower bounding of the objective function through the use of the α parameter. The algorithm was implemented for finding the global minimum configuration of small Lennard–Jones and Morse microclusters. For larger ones lower and upper bounds were derived by using a relaxation procedure. Later, these ideas sparked the development of the αBB algorithm for general nonconvex optimization problems.

See also

- ▶ Adaptive Simulated Annealing and its Application to Protein Folding
- ▶ Genetic Algorithms
- ▶ Global Optimization in Protein Folding
- ▶ Molecular Structure Determination: Convex Global Underestimation
- ▶ Monte-Carlo Simulated Annealing in Protein Folding
- ▶ Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach
- ▶ Packet Annealing
- ▶ Phase Problem in X-ray Crystallography: Shake and Bake Approach
- ▶ Protein Folding: Generalized-ensemble Algorithms
- ▶ Simulated Annealing
- ▶ Simulated Annealing Methods in Protein Folding

References

1. Ballone P, Milani P (1990) Phys Rev 42:3905
2. Beck TL, Jellinek J, Berry RS (1987) J Chem Phys 87:545
3. Berry RS (1990) Scientif Amer
4. Boerdijk AH (1953) Philips Res Report 7:303
5. Conway JH, Sloane NJA (1988) Sphere packings, lattices and groups. Springer, Berlin
6. Coxeter HSM (1961) Introduction to geometry. Wiley, New York
7. Farges J, de Feraudy MF, Raoult B, Torchet G (1983) J Chem Phys 78:5067
8. Farges J, de Feraudy MF, Raoult B, Torchet G (1985) Surf Sci 156:370
9. Farges J, de Feraudy MF, Raoult B, Torchet G (1986) J Chem Phys 84:3491
10. Fejes-Toth L (1954) Regular figures. MacMillan, New York
11. Hoare MR (1979) Adv Chem Phys 40:49
12. Hoare MR, McInnes J (1972) Faraday Discuss Chem Soc 61:12
13. Hoare MR, McInnes J (1983) Adv Phys 32:791
14. Hoare MR, Pal P (1971) Nat Phys Sci 230:5
15. Hoare MR, Pal P (1971) Adv Phys 20:161
16. Hoare MR, Pal P (1972) J Crystallogr Growth 17:77
17. Hohl D, Jones RO, Car R, Parrinello M (1988) J Chem Phys 89:6823
18. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Science 220:671
19. Maier RS, Rosen JB, Xue GL (1992) Army High Performance Computing Res Center Preprint Univ Minnesota, 92–031
20. Maranas CD, Floudas CA (1992) J Chem Phys 97:10
21. Maranas CD, Floudas CA (1993) Ann Oper Res 42:85
22. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, New York

23. Navon IM, Brown FB, Robertson H (1990) Comput Chem 14:305
24. Northby JA (1987) J Chem Phys 87:6166
25. Piela L, Kostrowicki J, Scheraga HA (1989) J Phys Chem 93:3339
26. Pool R (1990) Science 24:1186
27. Shalloway D (1991) Recent advances in globaloptimization. Princeton Univ. Press, Princeton
28. Shalloway D (1992) J Global Optim 3:281
29. Stillinger FH, Weber TA 1429
30. Vlachos DG, Schmidt LD, Aris R (1992) J Chem Phys 96:6880
31. Vlachos DG, Schmidt LD, Aris R (1992) J Chem Phys 96:6891
32. Wille LT (1986) Nature 34:46
33. Wille LT (1987) Chem Phys Lett 133:405
34. Wille LT, Vennik J (1985) J Phys A 18:L419
35. Xue GL (1992) Army High Performance Computing Res Center Preprint Univ Minnesota
36. Xue GL (1992) Army High Performance Computing Res Center Preprint Univ Minnesota, 92-047
37. Xue GL (1997) J Global Optim 11:83
38. Xue GL, Maier RS, Rosen JB (1992) Internat. Conf. Supercomputing 6th ACM Internat. Conf. Supercomputing, Washington, DC, 19–23 July, p 409

Global Optimization in Location Problems

HOANG TUY
Institute of Mathematics, VAST, Hanoi, Vietnam

MSC2000: 90C26, 65D18, 90B85

Article Outline

Single Facility Location
  Minisum and Maxisum
  Maximin and Minimax
Constrained Location
  Location on Union of Convex Sets
  Location on Area with Forbidden Regions
  General Constrained Location Problem
Multiple Source
  Clustering
Multiple Facility
  Molecular Conformation
  Distance Geometry
References

A general mathematical problem encountered in various applications is to find the configuration of r unknown points in $\mathbb{R}^n$ (quite often $n \leq 3$) satisfying a number of constraints on their mutual distances and their distances to m fixed points, while minimizing a given function of these distances. Often the unknown points represent the locations of facilities to be constructed to serve the users located at the fixed points, so as to minimize a cost function (travel time, transport cost for customers, etc.) or to maximize the global attraction (utility, number of customers, etc.). Also the unknown points may represent the cluster centers while the fixed points are the objects to be classified into groups (clusters). The biggest challenge occurs when the unknown points represent the objects (atoms, particles) whose interactions depend upon their mutual distances: the objective function in these problems is then interpreted as a "potential energy function" that should attain a global minimum at the unknown configuration.

For many years, combinatorial geometric reasoning and nonlinear programming methods have been the basic tools in the study of these problems. However, since most nonconvex problems are characterized by the existence of many local nonglobal minimizers, other more suitable methods have to be used to efficiently cope with this difficulty.

Global optimization methods began to be introduced in these fields more than two decades ago [9,15]. Subsequently, dc optimization techniques were used to tackle facility location with nonconvex objective functions and nonconvex constraints [5,6,12,13,19,20,21].

Single Facility Location

The first location problem, introduced by Weber (1909), was to find the location of a facility so as to minimize the sum of its weighted distances to a given set of users located in a plane. Over the years this unconstrained convex minimization problem has been further and further generalized, leading to more and more complex models of location.

Minisum and Maxisum

Suppose a new facility is designed to serve m users located at $a^1, \ldots, > a^m \in \mathbb{R}^2_+$. Certain users, henceforth called the "attraction points," are interested in having the facility located as close to them as possible. Others, called the "repulsion points," would like the facility to be located as far away from them as possible. Let J_1, J_2 denote the index sets of attraction and repulsion

points, respectively. For each user $j = 1, \ldots, m$ a function $q_j(t)$ is known that measures the cost of traveling a distance t away from a^j; also, $h_j(x)$ is a function of the distance from user j to point $x \in \mathbb{R}^2$. It is assumed that the function $q_j(t)$ is *concave increasing* with $q_j(t) \to +\infty$ as $t \to \infty$, while $h_j(x)$ is a convex function such that $h_j(x) \to +\infty$ as $\|x\| \to +\infty$. So if x is the unknown location of the facility, then to take account of the interest of attraction points, one should try to minimize the sum $\sum_{j\in J_1} q_j(h_j(x))$, whereas from the point of view of repulsion points one should try to maximize the sum $\sum_{j\in J_2} q_j(h_j(x))$. Under these conditions, a reasonable objective of the decision maker may be to locate the facility so as to minimize the quantity

$$\sum_{j\in J_1} q_j(h_j(x)) - \sum_{j\in J_2} q_j(h_j(x))$$

over $\mathbb{R}^n_+$. Denoting the right derivative of $q_j(t)$ at 0 by $q_j{}^+(0)$ and assuming $q_j^+(0) < +\infty \forall j$, it can easily be seen that each function $g_j(x) := K_j h_j(x) + q_j[h_j(x)]$ is convex for $K_j \geq q_j^+(0)$, and so we come up with the dc optimization problem

$$\min\{G(x) - H(x) | x \in \mathbb{R}^n_+\}\,, \tag{1}$$

where $G(x)$, $H(x)$ are convex functions defined by

$$G(x) = \sum_{j\in J_2} g_j(x) + \sum_{j\in J_1} K_j h_j(x)\,,$$
$$H(x) = \sum_{j\in J_1} g_j(x) + \sum_{j\in J_2} K_j h_j(x)\,.$$

Problems with the above objective function are called *minisum* problems.

In other circumstances, instead of minimizing the cost, one may seek to maximize the total attraction

$$\sum_{j\in J_1} q_j[h_j(x)] - \sum_{j\in J_2} q_j[h_j(x)]\,,$$

where each q_j is a *convex decreasing* function. Assuming $q_j^+(0) > -\infty$, the problem is then

$$\max\{\tilde{G}(x) - \tilde{H}(x) | x \in \mathbb{R}^n_+\}\,, \tag{2}$$

where $\tilde{G}(x)$, $\tilde{H}(x)$ are now the convex functions

$$\tilde{G}(x) = \sum_{j\in J_1} g_j(x) + \sum_{j\in J_2} K_j h_j(x)\,,$$
$$\tilde{H}(x) = \sum_{j\in J_2} g_j(x) + \sum_{j\in J_1} K_j h_j(x)\,.$$

Obviously, any maxisum problem can be converted into a minisum one and vice versa. Most problems studied in the literature are minisum, under much more restricted assumptions than in the above setting (see [16] and references therein). Weber's classical formulation corresponds to the case $J_2 = \varnothing$ (no repulsion points) and $h_j(x) = \|x - a^j\|, q_j(t) = w_j t, w_j \geq 0, \forall j$. The cases $J_2 \neq \emptyset$ with $q_j(t)$ nonlinear have begun to be investigated only recently, motivated by growing concerns about the environment.

Maximin and Minimax

When siting emergency services, like a fire station, one does not want to maximize the overall attraction but rather to guarantee for every user a minimal attraction as large as possible. The problem, often referred to as the *p-center problem*, can be formulated as

$$\max\left\{\min_{j=1,\ldots,m} q_j[h_j(x)] | x \in \mathbb{R}^n_+\right\}, \tag{3}$$

where $q_j(t)$ are *convex decreasing* functions (minimax problem). Assuming $|q_j^+(0)| < \infty \forall j$ as previously, we have the dc representation $q_j[h_j(x)] = g_j(x) - K_j h_j(x)$, hence

$$\min_{j=1,\ldots,m} q_j[h_j(x)] = \sum_{j=1}^n g_j(x) - \max_{j=1,\ldots,n}\left[K_j h_j(x) + \sum_{i\neq j} g_i(x)\right],$$

and so (3) is again a dc optimization problem.

By contrast, when siting an obnoxious facility, one wants to minimize the maximal attraction to an user, so the optimization problem to be solved is

$$\max\left\{\min_{j=1,\ldots,k} q_j[h_j(x)] | x \in \mathbb{R}^n_+\right\}, \tag{4}$$

where $q_j(t)$ are *concave increasing* functions (minimax problem). Again, assuming $|q_j^+(0)| < \infty \forall j$, we have the dc representation $q_j[h_j(x)] = K_j(x) - g_j(x)$, and so

$$\max_{j=1,\ldots,m} q_j[h_j(x)] = \max_{j=1,\ldots,n}\left[K_j h_j(x) + \sum_{i\neq j} g_j(x)\right] - \sum_{j=1}^m g_j(x)\,,$$

i. e., the minmax location problem (4) is again a dc optimization problem.

A special maximin location problem worth mentioning is the *design centering* problem encountered in engineering design. Given a compact convex set $B \subset \mathbb{R}^n$ containing 0 in its interior and m compact convex sets D_j, $j = 1, \ldots, m$ contained in a compact convex set $C \subset \mathbb{R}^n$, find $x \in C$ so as to maximize

$$r(x) = \min_{j=0,1,\ldots,m} r_j(x) ,$$

where $r_j(x) = \min\{p(y-x)\colon y \in D_j\}$, $p\colon \mathbb{R}^n \to \mathbb{R}_+$ is the gauge of B and $D_0 = \mathbb{R}^n \setminus C$. It can be shown [17] that the function $r_0(x)$ is concave while $r_1(x), \ldots, r_m(x)$ are convex, so this can be viewed as a maximin problem in which each D_j is a user and $r_j(x)$ is the distance from point x to user j.

Constrained Location

In the real world many factors may set restrictions on the facility sites. Therefore, practical location problems are often constrained.

Location on Union of Convex Sets

The most simple type of restriction is that the facility can be located only in one of several given convex regions $C_1, \ldots, C_k$ [8]. If $C_i = \{x\colon c_i(x) \le 0\}$, with $c_i(x)$ being convex functions, then the constraint $x \in \cup_{i=1}^k C_i$ can be expressed as

$$\min_{i=1,\ldots,k} c_i(x) \le 0 ,$$

which is a dc constraint.

Location on Area with Forbidden Regions

In other circumstances, the facility can be located only outside some forbidden regions that are, for instance, open convex sets $C_i^o = \{x\colon c_i(x) < 0\}$, with $c_i(x)$ being convex functions (see, e. g., [2]). Since the constraint $x \notin \cup_{i=1}^k C_i^o$ is equivalent to $\min_{i=1,\ldots,k} c_i(x) \ge 0$, this is again a dc constraint.

General Constrained Location Problem

The most general situation occurs when the constraint set is a compact, not necessarily convex, set. However, a striking result of dc analysis shows that even in this general case the constraint can be expressed as a dc inequality [12,22].

Of course the corresponding dc optimization problem is very hard. Although a method (the relief indicator method [18]) exists for dealing with general nonconvex constraints, so far it only works in low dimension.

Multiple Source

When more than one facility is to be located, the objective function depends upon whether these facilities provide the same service or different services to the users.

If there are $r \ge 2$ facilities providing the same service, these facilities are called *sources*. Each user is then served by the closest source. So if x^i is the unknown location of the ith facility and $X = (x^1, \ldots, x^r) \in (\mathbb{R}^2)^r$, then the overall attraction is

$$\sum_{j in J_1} q_j[\tilde{h}_j(X)] - \sum_{j \in J_2} q_j[\tilde{h}_j(X)] , \tag{5}$$

where $\tilde{h}_j(X) = \min\{h_j(x^i)\colon i = 1, \ldots, r\}$ and q_j, h_j are as previously. Since $\tilde{h}_j(X) = \sum_{i=1}^r h_j(x^i) - \max_{i=1,\ldots,r} \sum_{i \ne l} h_j(x^i)$, the first term in (5) is the dc function

$$\sum_{j \in J_1} g_j(X) - \sum_{j \in J_1} K_j \left[\sum_{i=1}^r h_j(x^i) + \max_{l=1,\ldots,r} \sum_{i \ne l} h_j(x^i) \right],$$

where $K_j \ge |q_j^+(0)|$ and

$$g_j(X) = q_j[\tilde{h}_j(X)] + K_j \left[\sum_{i=1}^r h_j(x^i) + \max_{l=1,\ldots,r} \sum_{i \ne l} h_j(x^i) \right]$$

is a convex function. Similarly for the second term in (5). Hence the objective function in the r source problem is a dc function on $(\mathbb{R}^2)^r$.

The multisource problem is usually referred to as the *generalized Weber problem*, or also the *r-median problem* when $J_2 = \emptyset$. Traditionally it is often viewed as a location-allocation problem and formulated as a mixed 0-1 integer programming problem (see, e. g., [16]).

Clustering

In many practical situations we have a set of objects of a certain kind that we want to classify into $r \ge 2$ groups

(clusters), each including elements close to each other in some well-defined sense. In the simplest case, this gives rise to the following problem: for a given finite set of points $a^1, \ldots, a^m \in \mathbb{R}^n$, find r cluster centers $x^i \in \mathbb{R}^n$, $i = 1, \ldots, r$, such that the sum of the minima over $i \in \{1, \ldots, r\}$ of the "distance" between each point a^j and the cluster centers x^i, $i = 1, \ldots, r$, is minimized. If $d(a, x)$ denotes the distance from a to x, then the problem is

$$\min \left\{ \sum_{j=1}^{m} \min_{i=1,\ldots,r} d(a^j, x^i) \colon x^i \in [0, b] \right\} . \tag{6}$$

Formally, this is nothing but the r-median problem, i. e., the generalized Weber problem with $J_2 = \emptyset$.

If $d(a, x) = \sum_{i=1}^{n} |a_i - x_i|$, then, using the equality $|a_i - x_i| = \min\{y_i \colon -y_i \le a_i - x_i \le y_i\}$, problem (6) can be written as

$$\begin{aligned} &\sum_{j=1}^{m} \min_{l=1,\ldots,r} \left(\sum_{i=1}^{n} y_i^{jl} \right) \\ &-y^{jl} \le a^j - x^l \le y^{jl} \\ &j = 1, \ldots, m, l = 1, \ldots, r , \end{aligned}$$

which is a concave minimization problem under linear constraints. One way to cope with the large dimension of this problem is to replace it with the equivalent bilinear program

$$\begin{aligned} &\min \sum_{j=1}^{m} \sum_{l=1}^{r} t_{jl} y^{jl} \\ &\text{s.t.} -y^{jl} \le a^j - x^l \le y^{jl} \quad j = 1, \ldots, m, l = 1, \ldots, r \\ &\sum_{l=1}^{r} t_{jl} y^{jl}, \; t_{jl} \ge 0, \sum_{l=1}^{r} t_{jl} = 1 . \end{aligned}$$

and to solve the latter approximately to a local optimum by alternately fixing t and y.

When $d(a, x) = \sqrt{\sum_{i=1}^{n} (a_i - x_i)^2}$, the problem is no longer a concave minimization but can be reduced to a dc program by easy manipulations. In [1] results of solving the generalized Weber problem with $m = 10{,}000$, $p = 2$, and $m = 1{,}000$, $p = 3$, by dc methods are reported. Alternatively, (6) can also be transformed into a monotonic optimization and solved by recently developed monotonic optimization methods [23,24]. For this observe that $d(a, x) = (d(a, x) + \sum_{i=1}^{n} x_i) - \sum_{i=1}^{n} x_i$, and since $u(a, x) = d(a, x) + \sum_{i=1}^{n} x_i$ and $\sum_{i=1}^{n} x_i$ are both increasing functions, it follows that $d(a,x)$ is a dm (*d*ifference of *m*onotonic) function, and, hence, (6) is a monotonic optimization problem.

Multiple Facility

When the $r \ge 2$ facilities to be located provide different services, aside from the costs due to interactions between facilities and users, one should also consider the costs due to pairwise interactions between facilities. The latter costs can be expressed by functions of the form $\rho_{il}[h_{il}(x^i, x^l)]$, where again $h_{il}(x^i, x^l)$ are convex nonnegative valued functions and $\rho_{il}(t)$ are concave increasing functions on $[0, +\infty)$ with finite right derivatives at 0. The total cost one would like to minimize is then

$$\sum_{i=1}^{r} F_i(x^i) + \sum_{i<l} \rho_{il}[h_{il}(x^i, x^l)] , \tag{7}$$

where $F_i(x^i) = \sum_{j \in J_1} q_{ji}[h_j(x^i)] - \sum_{j \in J_2} q_{ji}[h_j(x^i)]$ and q_{ji}, h_j are as in minisum single facility problems.

As we saw above, each function $F_i(x^i)$ is dc, hence each function $\rho_{il}[h_{il}(x^i, x^l)]$ is dc, too, and (7) is again a dc function on $(\mathbb{R}^2)^r$. In the special case when there are no repulsion points (every $F_i(x^i)$ is convex) and the pairwise interactions between facilities $\rho_{il}(t)$ are convex, this is simply a convex function. Also, in the absence of interactions between facilities ($\rho_{ij}(.) = 0 \; \forall ij$), the minimization of function (7) splits into r independent single facility minisum problems.

Molecular Conformation

A variant of the multifacility problem that has risen to attract much research in recent years is the so-called *molecular conformation* problem encountered in computational biology, computational chemistry, and protein folding. This is the problem of determining ground states or stable states of certain classes of molecular clusters and proteins and can be stated as follows [14]. Given a cluster of N atoms (in three-dimensional space), we wish to locate their centers $x^1, \ldots, x^N$ so as to minimize the potential energy function

$$V_N(x^1, \ldots, x^N) = \sum_{1 \le i < j \le N} v(\|x^i - x^j\|) ,$$

where $\|.\|$ is the euclidean norm and $v(r)$ the interatomic pair potential. This can be viewed as a multifacility problem in which there is no user but many facilities (the number N may be rather large; see, e. g., [14]). In models used for computation, the pair potentials of interest include the following:

$$v(r) = r^{-12} - 2r^{-6} \text{ (Lennard-Jones)},$$
$$v(r) = \left[1 - e^{\alpha(1-r)}\right]^2 - 1 \text{ (Morse)},$$
$$v(r) = \frac{z^\alpha z^\beta}{r} + Ae^{\frac{-r}{\rho}} \text{ (Born-Meyer)}.$$

Using representation theorems in dc optimization, it can be seen that these functions are dc (at least for $r \geq \varepsilon$, where ε is an arbitrary small positive number).

Distance Geometry

A related problem that also has applications in molecular conformation, and other questions such as surveying and satellite ranging, data visualization, and pattern recognition, etc., is the *multidimensional scaling problem* or *distance geometry problem*. It consists in finding r objects $x^1, \ldots, x^r$ in $\mathbb{R}^n$ such that the quantity

$$V_r(x^1, \ldots, x^r) = \sum_{i<j} w_{ij} \left(\delta_{ij}^2 - \|x^i - x^j\|^2\right)^2 \quad (8)$$

is smallest, where $\Delta = (\delta_{ij})$, $W = (w_{ij})$ are symmetric matrices of order r such that

$$\delta_{ij} = \delta_{ji} \geq 0, \quad w_{ij} = w_{ji} \geq 0 \quad (i < j);$$
$$\delta_{ii} = w_{ii} = 0 \quad (i = 1, \ldots, r).$$

By writing this problem as

$$\min \sum_{i<j} w_{ij}\|x^i - x^j\|^2 - 2\sum_{i<j} w_{ij}\delta_{ij}\|x^i - x^j\| \quad (9)$$
$$\text{s.t. } x^i \in \mathbb{R}^n (i = 1, \ldots, r)$$

or, alternatively, as

$$\min \sum_{i,j} w_{ij} t_{ij}^2 \left| \begin{array}{l} -t_{ij} \leq \delta_{ij}^2 - \|x^i - x^j\|^2 \leq t_{ij} \ (\forall i < j) \\ x^i \in \mathbb{R}^n, \quad i = 1, \ldots, r \end{array} \right. \quad (10)$$

we again obtain a dc optimization problem that is also a monotonic optimization problem.

References

1. Al-Khayyal FA, Tuy H, Zhou F (2002) Large-Scale Single Facility Continuous Location by D.C. Optimization. Optimization 51:271–292
2. Aneja YP, Parlar M (1994) Algorithms for Weber facility location in the presence of forbidden regions and/or barriers to travel. Transp Sci 28:70–216
3. Chen R (1983) Solution of minisum and minimax location-allocation problems with euclidean distances. Nav Res Logist Q 30:449–459
4. Chen R (1988) Conditional minisum and minimax location-allocation problems in Euclidean space. Transp Sci 22:157–160
5. Chen P, Hansen P, Jaumard B, Tuy H (1992) Weber's problem with attraction and repulsion. J Reg Sci 32:467–409
6. Chen P, Hansen P, Jaumard B, Tuy H (1998) Solution of the multifacility Weber and conditional Weber problems by D.C. Programming. Oper Res 46:548–562
7. Dresner Z (ed) (1995) Facility Location: A Survey of Applications and Methods. Springer, Berlin
8. Hansen P et al (1982) An Algorithm for a Constrained Weber Problem. Manage Sci 28:1285–1295
9. Hansen P et al (1985) The Minisum and Minimax Location-Probems Revisited. Oper Res 33:1251–1265
10. Horst R, Tuy H (1996) Global Optimization, 3rd edn. Springer, Berlin
11. Idrissi H, Loridan P, Michelot C (1988) Approximation of Solutions for Location Problems. J Optim Theory Appl 56:127–143
12. Konno H, Thach PT, Tuy H (1997) Optimization on Low Rank Nonconvex Structures. Kluwer, Dordrecht
13. Maranas CD, Floudas CA (1993) A global Optimization method for Weber's problem with attraction and repulsion. In: Hager WW, Heran DW, Pardalos PM (eds) Large Scale Optimization: State of the Art. Kluwer, Dordrecht, pp 1–12
14. Maranas CD, Floudas CA (1994) Global minimum potential energy conformations of small molecules. J Global Optim 4:135–171
15. Plastria F (1992) The generalized big square small square method for planar single facility location. Eur J Oper Res 62:163–174
16. Plastria F (1995) Continuous location problems. In: Dresner Z (ed) Facility Location: A Survey of Applications and Methods. Springer, Berlin, pp 225–262
17. Thach PT (1988) The design centering problem as a dc programming problem. Math Programm 41:229–248
18. Thach PT, Tuy H (1990) The relief indicator method for constrained global optimization. Naval Res Logist 37:473–497
19. Tuy H, Al-Khayyal FA (1992) Global Optimization of a Nonconvex Single Facility Problem by Sequential Unconstrained Convex Minimization. J Global Optim 2:61–71

20. Tuy H (1996) A General D.C. Approach to Location Problems. In: Floudas CA, Pardalos PM (eds) State of the Art in Global Optimization. Kluwer, Dordrecht, pp 413–432
21. Tuy H, Al-Khayyal FA, Zhou F (1995) D.C. optimization method for single facility location problem. J Global Optim 7:209–227
22. Tuy H (1998) Convex Analysis and Global Optimization. Kluwer, Dordrecht
23. Tuy H (2000) Monotonic Optimization: Problems and Solution Approaches. SIAM J Optim 11:464–494
24. Tuy H, Minoux M, Hoai Phuong NT (2006) Discrete Monotonic Optimization with Application to A Discrete Location Problem. SIAM J Optim 17:78–97

Global Optimization Methods for Harmonic Retrieval

WILLIAM EDMONSON[1], WEN LEE[2]
[1] Hampton University, Virginia, USA
[2] University Florida, Gainesville, USA

MSC2000: 90C26, 65T40, 90C30, 90C90

Article Outline

Keywords
Interval Arithmetic
Interval Method for Solving HR
  Simulations
EMIM
  Simulations
Conclusion
See also
References

Keywords

Global optimization; Harmonic retrieval; Expectation maximization; Interval methods

The *harmonic retrieval* (HR) problem is an ubiquitous problem that arises in various applications, such as signal modeling and direction-of-arrival. It consists of estimating the parameters of multiple sinusoids from noisy data. The data is modeled as

$$y(t) = \sum_{k=1}^{K} a_k^* \sin(2\pi f_k^* t) + n(t), \quad t = 1, \ldots, N, \tag{1}$$

where a_k^*, f_k^*, and ϕ_k^* are the amplitude, frequency, and phase of the kth sinusoid, respectively. It is assumed that the number of sinusoids, K, is known and all frequencies satisfy $0 < f_k^* < 0.5$, $k = 1, \ldots, K$, and $f_k^* \neq f_j^*$ for $k \neq j$. In addition, the noise, $n(t)$, is assumed to be zero-mean, white Gaussian noise (WGN) with variance σ_n^2. Given the data, $y(t)$ for $t = 1, \ldots, N$, the goal is to estimate the sinusoid parameters, $\theta^* = [a_1^*, \ldots, a_K^*, f_1^*, \ldots, f_K^*]$.

The conventional FFT or periodogram-based methods [4, Chapt. 1]are only able to solve the HR problem when frequencies are spaced more than $1/N$ cycles/sample apart, where N is the number of available data points. To tackle the problem where the difference between any two frequencies is smaller than the threshold $1/N$, high resolution techniques must be used [4, Chapt. 5]. The *sinusoidal parameter estimation problem* is based on solving the *least squares* (LS) problem (P):

$$(P) \quad \widehat{\theta}_{\mathrm{LS}} \triangleq \arg\min_{\theta} J(\theta), \tag{2}$$

where

$$J(\theta) = \sum_{t=1}^{N} \left\{ y(t) - \sum_{k=1}^{K} a_k \sin(2\pi f_k t + \phi_k) \right\}^2, \tag{3}$$

and $\theta = [a_1, \ldots, a_K, f_1, \ldots, f_K, \phi_1, \ldots, \phi_K]$. We can see from (3) that the objective function is nonconvex, which suggests that a global optimization method represents the most appropriate procedure for determining $\widehat{\theta}_{\mathrm{LS}}$.

Two methods that have been proposed for solving (P) are the one proposed in [8], for which we will refer to as *Stoica's method* and the *Iterative Quadratic Maximum Likelihood method* (briefly: IQML method) [1]. Both methods can not guarantee convergence unless the initial conditions are sufficiently close to the global minimum. Stoica's method first generates initial estimates using the *overdetermined Yule–Walker method*. Then, it improves on these estimates by using a periodogram-based procedure and a simplified Gauss–Newton algorithm to iteratively maximize the likelihood function. In [8], it was shown experimentally that Stoica's method requires extremely large data records. The well known IQML method is an iterative quadratic maximization algorithm that attempts

to determine the *maximum likelihood* (ML) estimates in terms of a prediction polynomial. This algorithm, as our experiments show, produces poor estimates for short data records and/or low signal-to-noise ratio (SNR). The IQML algorithm is also noted to sometimes fail to converge and the estimated frequencies are almost always inconsistent [9].

Taking a different approach, we will apply the *global optimization* algorithm of *interval methods* (IM) to the HR problem (2). Interval method type algorithms [3,6,7] have proven to be an excellent and reliable procedure for solving global optimization problems involving nonconvex objective functions. One of the reasons one chooses interval methods is because they are applicable to most optimization problems regardless of convexity and differentiability of the objective function, or knowledge of its Lipschitz constant. Additionally, for *continuous* objective functions, its convergence to a global optimum interval has been proven [3]. In using the IM method for solving the LS estimates of (2), convergence is very slow.

One way to overcome the problem of slow convergence is to decompose the problem whereby optimization occurs over smaller dimensions and in parallel. This can be accomplished through combining the *expectation-maximization algorithm* (briefly: *EM algorithm*) [2] with the interval method. This proposed combination of the EM algorithm with the interval method is defined as the *expectation-maximization interval* method (EMIM) algorithm. The EM algorithm represents a computationally efficient method for solving estimation problems. For the HR problem, the EM algorithm decomposes the HR problem into K subproblems, where K is the number of sinusoids. The K subproblems, which are nonconvex optimization problems, are then solved using an IM global optimization method. This results in an algorithm that is able to converge to the global minimum interval with significantly reduced computational complexity, in comparison with using the IM algorithm alone for solving (P).

Interval Arithmetic

Interval methods are a class of global optimization algorithms that utilize *interval arithmetic*. An interval which contains the global minimum is found by partitioning the search space into regions, where at each iteration, regions are selected for further search by additional partitioning. Those partitions that cannot contain the global minimum are discarded. A major advantage of interval methods is their ability to find the global minimum of nonconvex differentiable or nondifferentiable objective functions.

Interval arithmetic [6] was developed to automatically estimate and control numerical errors caused by finite precision of computer arithmetic. The *INTLIB* library [5] is used to implement interval arithmetic as used in the IM algorithm. A real interval number $X = [a, b]$ consists of the set set$\{x: a \le x \le b\}$ of real numbers. Additional notations used here are: the upper bound (ub) of $X = b$, the lower bound (lb) of $X = a$, the mid-point of X is $m(X) = (a + b)/2$, and the width of X is $w(X) = b - a$. Furthermore, $w(\mathbf{X}) = \max\{w(X_i)\}_{i=1}^{i=n}$ where $\mathbf{X} = [X_1, \ldots, X_n]^\mathsf{T}$. The general interval arithmetic operational rules is defined as $X \square Y = \{x \square y : x \in X, y \in Y\}$, where X and Y are real interval numbers and $\square$ represents the arithmetic operations of plus, minus, multiplication, and division. For additional information on interval arithmetic see [6,7].

The *unconstrained global optimization* problem can be described as

$$\min_{x \in D} g(x), \tag{4}$$

where $g(x)$: $\mathbf{R}^n \leftarrow \mathbf{R}$, $x \in \mathbf{R}^n$ and $D \in \mathbf{R}^n$ represents the feasible region. The main tool for solving the problem in (4) is the concept of *inclusion function*. A function $G(X)$: $\mathbb{J}^n \rightarrow \mathbb{J}$ is an inclusion function of the objective function $g(x)$, if $x \in Y$ implies that $g(x) \in G(Y)$ and that the *isotonicity property* is met (i. e. $X \subseteq Y$ implies that $F(X) \subseteq G(Y)$). The inclusion function with isotonicity property provides the theory for the use of interval methods as a global optimization procedure. In short, inclusion functions represent the range of function values of f over the interval X.

The optimization procedure for the interval method involves continually bisecting a box X_i from an initial box, X_0, until $G(X^i)$, the inclusion function, contains the global minimum given that $w(G(X^i)) \le \epsilon$. What differentiates this method from the method of exhaustive search is that regions of the objective are discarded from evaluation if the lb $G(X^i)$ in the list, $\mathcal{L}$, is greater than the minimum between the past or present value of ub $G(X^j)$ given that $i \neq j$. The algorithm of E.R.

Global Optimization Methods for Harmonic Retrieval, Table 1
A pseudocode for interval methods

```
PROCEDURE interval method
  Set Y := X;
  Calculate G(Y), y :=lbG(Y), g̃ :=ub G(m)
    where m =mid Y
    Initialize list ℒ := {(Y, y)}.
  REPEAT until convergence
    Choose a coordinate direction k, parallel to Y_i,
    and of max length.
    Bisect Y to obtain boxes V1, V2, where
    Y = V1 ∪ V2.
    Calculate G(V1) and G(V2) and v_i :=lb G(V_i)
    for i = 1,2.
    Place (G(V_i), v_i) at end of list.
    Choose pair (Ỹ, ỹ) from ℒ such that ỹ ≤ z,
    ∀(Z, z).
    Discard pairs from list, (Z, z), if z > g̃.
    Terminate if ω(Z) < ϵ, ∀Z, in ℒ.
    Denote first pair of list by (Y, y).
    Compute m := mid Y and
    g̃ = min(g̃, ub G(m)).
  RETURN
END interval method
```

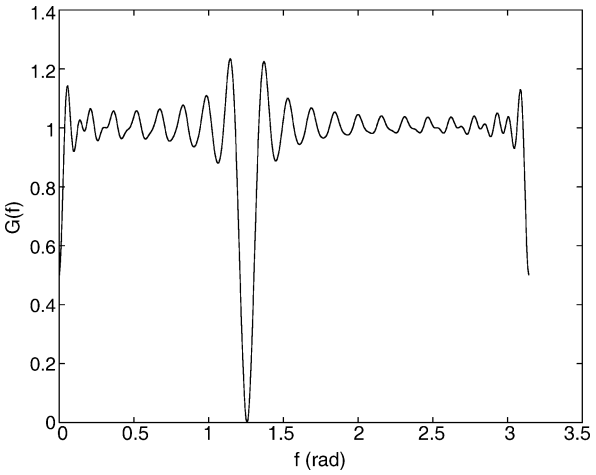
Hansen [3,7] is the particular interval method that will be used for locating the LS estimates of the HR problem and is outlined in Table 1. In [7], it was proven that convergence to the global minimum was achieved if $w(G(X)) \to 0$ as $w(X) \to 0$.

Interval Method for Solving HR

To apply the IM to solving the HR problem, the objective function (3) must be placed in its inclusion form:

$$\mathbf{J}(\Theta) = \sum_{t=1}^{N} \left[y(t) - \sum_{k=1}^{K} A_k \sin(2\pi F_k t + \Phi_k) \right]^2, \quad (5)$$

where $\Theta = [A_1, \ldots, A_K, F_1, \ldots, F_K, \Phi_1, \ldots, \Phi_K]$ and A_k, F_k, and Φ_k are the interval counterparts of a_k, f_k, and ϕ_k, respectively. Throughout this paper capital letters represent interval variables that correspond to its real variable equivalent. The initial interval, Θ_0, is chosen such that it encompasses the global minimum. This is accomplished by choosing an interval that is determined from a priori information or from other high resolution HR methods [4]. The IM of Hansen's, described in previous section, is used to determine the global minimum, Θ^*, of (5). The objection function (5) for a single frequency, phase and amplitude held constant, is plotted in Fig. 1). It can easily see that this represents a very difficult but practical problem for global optimization.



Global Optimization Methods for Harmonic Retrieval, Figure 1
Objective function of a single sinusoid

Simulations

In this section, a numerical experiment will be demonstrated to show the performance of the IM for solving the HR problem (P). The experiments consist of estimating the sinusoid parameters for the following data,

$$y(t) = 1.0 \sin(2\pi(0.2)t + 0.0) + n(t), \quad t = 1, \ldots, 35, \quad (6)$$

where $n(t)$ is white Gaussian noise. We choose the initial box for the IM algorithm to be $\Theta = [A, F, \Phi]^\intercal = [[0.71.2], [0.10.3], [00.4]]^\intercal$. The signal-to-noise-ratio (SNR) is defined as

$$10 \log \left[\sum_{k=1}^{K} 0.5 \frac{(a_k^*)^2}{\sigma_n^2} \right],$$

where σ_n^2 is the variance of the noise. The results of this simulation, shown in Table 2, is described in terms of sample mean and standard deviation based on 50

Global Optimization Methods for Harmonic Retrieval, Table 2
IM estimates

IM: $N = 35$ (50 MC runs)			
SNR	10	5	0
$a^* = 1.0$	1.0155 ± 0.0518	1.0447 ± 0.0913	1.0633 ± 0.1365
$f^* = .20$	0.1995 $\pm 6.591 \cdot 10^{-4}$	0.1993 ± 0.0012	0.1989 ± 0.0017
$\phi^* = 0$	0.0654 ± 0.0564	0.0839 ± 0.0975	0.1193 ± 0.1319

Global Optimization Methods for Harmonic Retrieval, Table 3
IQML Estimates

IQML: $N = 35$ (50 MC runs)			
SNR	10	5	0
$a^* = 1.0$	1.0080 ± 0.0533	0.9862 ± 0.1479	0.8919 ± 0.3230
$f^* = .20$	0.1998 $\pm 7.623 \cdot 10^{-4}$	0.1970 ± 0.0202	0.1728 ± 0.0839
$\phi^* = 0$	0.0141 ± 0.0949	-0.0126 ± 0.2852	0.5288 ± 1.1429

Monte-Carlo (MC) runs. This results are based on the midpoints of Θ. Note that the final estimates, $\widehat{\theta}$, are very close to the true value of θ^* with a small standard deviation. In comparison with IQML, see Table 3, the IM fares considerably better in both mean and standard deviation. This is particularly notable when comparing the frequency component, which represents the most important feature of harmonic retrieval.

The convergence rate of the IM is sensitive to the order, K, of the HR problem. In fact, the dimensionality of the parameter space, $\mathfrak{I}^n$, increases at a rate of $3K$. Thus, as n increases, the convergence rate becomes prohibitively slow. The *curse of dimensionality* can be mitigated through decomposition and parallelizing the problem by utilizing the EM algorithm as described in the next section.

EMIM

The detailed development of the EM algorithm [2] is well-known, and will be outlined here as part of the development of the EMIM algorithm for solving the HR problem. To determine the LS estimates of the sinusoidal parameters, the EM algorithm first decomposes the observed data $y(t)$ into its signal components (E step) and then estimates the parameters of each signal component separately (M step). The algorithm iterates back and forth between the E step and M step, using the current estimate to decompose the observed data better and thus improve the next parameter estimate.

For the HR problem, the incomplete data is the observed data, $y(1), \ldots, y(N)$. The complete data is modeled as the following K data records:

$$y_k(t) = a_k^* \sin(2\pi f_k^* t + \phi_k^*) + n_k(t), \quad k = 1, \ldots, K,$$

where $n_k(t) = \beta_k[y(t) - \sum_{k=1}^K a_k^* \sin(2\pi f_k^* t + \phi_k^*)]$. The β_k's are arbitrary real-valued scalars satisfying $\sum_{k=1}^K \beta_k = 1$ and $\beta_k \geq 0$. Thus $\sum_{k=1}^K n_k(t) = n(t)$, for $t = 1, \ldots, N$. The EM algorithm, beginning with $n = 0$, is represented by the following two steps:

E) For $k = 1, \ldots, K$, compute

$$\widehat{\gamma}_k^{(n)}(t) = \widehat{a}_k^{(n)} \sin(2\pi \widehat{f}_k^{(n)} t + \widehat{\phi}_k^{(n)}) + \beta_k \left[y(t) - \sum_{l=1}^K \widehat{a}_l^{(n)} \sin(2\pi \widehat{f}_l^{(n)} t + \widehat{\phi}_l^{(n)}) \right]. \quad (7)$$

M) For $k = 1, \ldots, K$,

$$\widehat{\theta}_k^{(n+1)} = \arg \min_{a_k, f_k, \phi_k} J_k^{(n)}, \quad (8)$$

where

$$J_k^{(n)} = \sum_{t=1}^N (\widehat{\gamma}_k^{(n)}(t) - a_k \sin(2\pi f_k t + \phi_k))^2. \quad (9)$$

The parameter vector $\widehat{\theta}_k^{(n)} \triangleq [\widehat{a}_k^{(n)}, \widehat{f}_k^{(n)}, \widehat{\phi}_k^{(n)}]^\top$ is the estimate for $\theta_k^* \triangleq [a_k^*, f_k^*, \phi_k^*]^\top$ after n iterations. In the original HR problem, we have to search the $(3 \times K)$-dimensional parameter space to find the minimum value of the least squares objective function. But after the EM algorithm decomposes the HR problem into K smaller subproblems, we only have to solve K subproblems each of which requires the search of a 3-dimensional parameter space to find the global optimal point(s). This results in a significant reduction in computational complexity.

To solve the minimization problem in M step, we resort to using the IM for finding the final interval that contains the point minimizing the objective function. Since IM has been proven to converge to the global optimum for *continuous* objective functions [3], this algorithm will not be trapped in the local extremum. Needed in the IM algorithm is the inclusion function of the objective function (6), which is constructed by forming the natural interval extension [3,7] of J_k:

$$\mathcal{J}_k^{(n)} = \sum_{t=1}^{N} \left(\widehat{\gamma}_k^{(n)}(t) - A_k \sin(2\pi F_k t + \Phi_k)\right)^2, \quad (10)$$

where A_k, F_k, and Φ_k are the interval counterparts of a_k, f_k, and ϕ_k, respectively. The initial value $\widehat{\theta}_k^{(0)} = [\widehat{a}_k^{(0)}, \widehat{f}_k^{(0)}, \widehat{\phi}_k^{(0)}]^\top$ are arbitrarily guessed or can come from other high-resolution estimation methods. The initial interval $\Theta_{k,0} = [A_{k,0}, F_{k,0}, \Phi_{k,0}]^\intercal$ for the M) step is the region over which the minimization is carried out. This initial interval $\Theta_{k,0}$ is used at the beginning of each M) step of the EMIM algorithm. At the $(n+1)$st iteration of EMIM, the IM partitions $\Theta_{k,0}$ iteratively to find the final interval estimate $\widehat{\Theta}_k^{(n+1)}$. The $m(\widehat{\Theta}_k^{(n+1)}) = \widehat{\theta}_k^{(n+1)}$ will be used as the parameter estimate to compute $\widehat{\gamma}_k^{(n+1)}(t)$ for the next iteration of the EMIM algorithm. The process is repeated until $\sum_{k=1}^{K} \left\|\widehat{\theta}_k^{(n+1)} - \widehat{\theta}_k^{(n)}\right\| \leq \rho$, where ρ is chosen by the user.

Consider the case where $\widehat{\theta}_i^{(0)} = \widehat{\theta}_j^{(0)}$ and $\beta_i = \beta_j$. It is straightforward to see that $\widehat{\gamma}_i^{(n)}(t) = \widehat{\gamma}_j^{(n)}(t)$ and $\mathcal{J}_i = \mathcal{J}_j$ in the E)-step and M)-step, respectively. Thus, $\widehat{\Theta}_i^{(n+1)} = \widehat{\Theta}_j^{(n+1)}$ for all n which means that the final estimates for θ_i and θ_j will be the same. In order to avoid this problem, β_i must not equal β_j or $\widehat{\theta}_i^{(0)}$ must not equal $\widehat{\theta}_j^{(0)}$ in order to fully exploit the capability of the EMIM algorithm.

Simulations

Our experiments consist of estimating the sinusoidal parameters for the following data,

$$y(t) = 1.0\sin(2\pi(0.2)t + 0.0) + 1.0\sin(2\pi(0.22)t + 0.0) + n(t), \quad t = 1, \ldots, 35,$$

where $n(t)$ is white Gaussian noise. Since $|0.2 - 0.22| < 1/35 = 0.02857$, the periodogram cannot be used to determine the frequencies. We choose the initial box for the EMIM algorithm to be:

$$[\Theta_{1,0}, \Theta_{2,0}]^\top = [A_{1,0}, F_{1,0}, \Phi_{1,0}, A_{2,0}, F_{2,0}, \Phi_{2,0}]^\top = [[0.7\ 1.2], [0.1\ 0.3], [0\ 0.4], [0.7\ 1.2], [0.1\ 0.3], [0\ 0.4]]^\top$$

and $\beta_1 = 0.09$, $\beta_2 = 0.91$. The signal-to-noise-ratio (SNR) is defined as

$$10\log\left[\sum_{k=1}^{K} 0.5\frac{(a_k^*)^2}{\sigma_n^2}\right],$$

where σ_n^2 is the variance of the noise. If no a priori information about the possible values of the sinusoid parameters is available, the full range of possible values for the frequency, the phase, and the amplitude must be used as the initial intervals. Utilizing the full range will impose no difficulty when very fast computing engines are used. However, other high resolution techniques can be used to yield a smaller and more cogent initial interval.

Using 50 MC runs, we computed the sample means and standard deviations for the EMIM and the IQML algorithms. (See Table 4 and Table 5, respectively). As for the EMIM, the mid-points of the final interval estimates are considered as the final estimates, thus the

Global Optimization Methods for Harmonic Retrieval, Table 4
EMIM estimates

EMIM: $N = 35$, $\rho = 10^{-6}$ (50MC runs)			
SNR	10	5	0
$a_1^* = 1.0$	1.0305 ± 0.0992	1.0235 ± 0.1389	1.0263 ± 0.1622
$f_1^* = .20$	0.1993 $\pm 2.209 \cdot 10^{-4}$	0.1993 $\pm 4.119 \cdot 10^{-4}$	0.1969 ± 0.0110
$\phi_1^* = 0$	0.0631 ± 0.0764	0.0851 ± 0.1152	0.1369 ± 0.1609
$a_2^* = 1.0$	1.0284 ± 0.0744	1.0501 ± 0.1036	1.0995 ± 0.1054
$f_2^* = .22$	0.2192 ± 0.0012	0.2194 ± 0.0016	0.2182 ± 0.0051
$\phi_2^* = 0$	0.0746 ± 0.1177	0.0757 ± 0.1224	0.1314 ± 0.1662

Global Optimization Methods for Harmonic Retrieval, Table 5
IQML estimates

IQML: $N = 35$ (50 MC runs)			
SNR	10	5	0
$a_1^* = 1.0$	0.9549 $\pm$0.3283	0.6615 $\pm$0.2908	0.7404 $\pm$0.2778
$f_1^* = .20$	0.1963 $\pm$0.0137	0.1707 $\pm$0.0476	0.1404 $\pm$0.0836
$\phi_1^* = 0$	0.3332 $\pm$0.6117	0.9323 $\pm$1.0843	0.5472 $\pm$1.0659
$a_2^* = 1.0$	0.9013 $\pm$0.3732	0.7567 $\pm$0.2683	0.8582 $\pm$0.2788
$f_2^* = .22$	0.2428 $\pm$0.0685	0.2559 $\pm$0.0867	0.2721 $\pm$0.0985
$\phi_2^* = 0$	$-$0.0886 $\pm$0.4852	$-$0.0079 $\pm$0.7185	0.3123 $\pm$0.8358

sample mean and variance can be calculated accordingly. Note that the EMIM generates estimates which have mean values very close to the true parameter values and relatively very small variances. As for the IQML, its variance for each value of SNR is significantly larger than the corresponding EMIM. Clearly, EMIM outperforms IQML by providing estimates that are less biased with smaller variances.

Conclusion

In comparison between the two types of IM algorithms with the IQML method, it was shown that both the IM and EMIM algorithms represent a powerful tool for solving the HR problem. Furthermore, it has been noted that by decomposing the problem by the EMIM algorithm does not degrade the performance of using the IM.

We have shown *experimentally* that the IM and EMIM algorithms are robust for very short data records and low SNR. Nevertheless, if the dimensionality is low or convergence to the ML estimates is desired, then the IM algorithm can be used. For either EMIM or IM, convergence time can be improved by generating initial interval of smaller widths by using other high resolution HR methods. Furthermore, using a multi-processor computer to implement the decomposed sub-problems in parallel can also reduce the execution time.

See also

► Signal Processing with Higher Order Statistics

References

1. Bresler Y, Macovski A (1986) Exact maximum likelihood parameter estimation of superimposed exponential signals in noise. IEEE Trans Acoustics, Speech and Signal Processing 34:1081–89
2. Feder M, Weinstein E (1988) Parameter estimation of superimposed signals using the EM algorithm. IEEE Trans Acoustics, Speech and Signal Processing 36:477–489
3. Hansen E (1992) Global optimization using interval analysis. M. Dekker, New York
4. Kay SM (1988) Modern spectral estimation: Theory and application. Prentice-Hall, Englewood Cliffs, NJ
5. Kearfott R, Dawande M, Du K, Hu C (1992) INTLIB: A portable FORTRAN 77 elementary function library. Interval Comput 3:96–105
6. Moore RE (1979) Methods and applications of interval analysis. SIAM, Philadelphia
7. Ratschek H, Rokne J (1988) New computer methods for global optimization. Horwood, Westergate
8. Stoica P et al (1989) Maximum likelihood estimation of the parameters of multiple sinusoids from noisy measurements. IEEE Trans Acoustics, Speech and Signal Processing 37:378–392
9. Stoica P, Li J, Söderström T (1998) On the inconsistency of IQML. IEEE Trans Signal Processing 37:378–392

Global Optimization Methods for Systems of Nonlinear Equations

GO for SNE

NGUYEN V. THOAI
University Trier, Trier, Germany

MSC2000: 65H10, 90C26, 90C30

Article Outline

Keywords
See also
References

Keywords

Systems of nonlinear equations; Global optimization

The problem of finding a solution of a *system of equations* and/or *system of inequalities* is one of the main re-

search subjects in numerical analysis and optimization. The source of systems of equations and/or inequalities contains many 'real-world' problems ([2,7]), the nonlinear complementarity problem (cf. also ► Generalized nonlinear complementarity problem), the variational inequality problem (cf. also ► Variational inequalities) over a convex set, Karush-Kuhn-Tucker systems, the feasibility problem, the problem of computing a Brouwer's fixed point ([10,15]).

In general, a system of nonlinear equations and/or inequalities is given by

$$(SNE)\begin{cases} h_i(x) = 0, & i \in I, \\ g_j(x) \le 0, & j \in J, \\ & x \in X, \end{cases}$$

where I, J are finite index sets, $X \subseteq \mathbf{R}^n$ is a convex set, and h_i $(i \in I)$, g_j $(j \in J)$ are nonlinear functions defined on a suitable set containing X.

Solution methods for (SNE), which are based on convex and nonsmooth optimization techniques, and fixed point algorithms can be found in [2,3,4,5,14,15], and references given therein.

In order to apply global optimization methods for solving (SNE), one defines a vector function h: $\mathbf{R}^n \to \mathbf{R}^{|I|}$ having components $h_i(x)(i \in I)$, a function

$$f(x) = \max\{\|h(x)\|, \{g_j(x): j \in J)\}\},$$

where $\|\cdot\|$ is any vector norm on $\mathbf{R}^{|I|}$, and considers the following *global optimization problem*

$$(GOP)\ f^* = \min\{f(x): x \in X\}.$$

In particular, the function f in (GOP) can be defined by

$$f(x) = \max\{\{|h_i(x)|: i \in I\}, \{g_j(x): j \in J)\}\}.$$

In general, a vector $x^* \in \mathbf{R}^n$ is a solution of (SNE) if and only if it is a *global optimal solution* of (GOP) and $f^* = f(x^*) = 0$. Thus, finding a solution of (SNE) can be replaced by computing a global optimal solution of (GOP). In the case that $I = \emptyset$, i. e., (SNE) is a system of inequalities, global optimization algorithms to (GOP) will terminate whenever a feasible point $x \in X$ is found satisfying $f(x) \le 0$. While applying a global optimization algorithm to (GOP), if it is pointed out that $f^* > 0$ (e. g., a lower bound μ of f^* can be computed such that $\mu > 0$), then obviously (SNE) has no solution.

There are three main classes of (SNE), which can be solved by implementable methods in global optimization:

i) The functions h_i $(i \in I)$ and g_j $(j \in J)$ are all d.c. (a function is called *d.c.* if it can be expressed as the difference of two convex functions, see ► D.C. programming).
ii) The functions h_i $(i \in I)$ and g_j $(j \in J)$ are all Lipschitzian with Lipschitz constants L_i $(i \in I)$ and M_j $(j \in J)$, respectively.
iii) The corresponding problem (GOP) can be replaced by a *convex relaxation problem*.

For class i), the function f in (GOP) is d.c., and one can find an explicit form of f as the difference of two convex functions, so that d.c. programming techniques can be applied ([9,11,12,18,19]).

For class ii), if in the definition of f, ℓ_p-norms are used, i. e.

$$\|h(x)\|_p = \begin{cases} \left(\sum_{i \in I} |h_i(x)|^p\right)^{\frac{1}{p}}, & 1 \le p < \infty \\ \max_{i \in I} |h_i(x)|, & p = \infty, \end{cases}$$

then f is Lipschitzian with Lipschitz constant $L = \max\{\sum_{i \in I} L_i, \{M_j: j \in J\}\}$. Algorithms for solving Lipschitz optimization problems can be found in [6,7,8,9,10,12,16,17].

Techniques for the construction of convex relaxation problems for some special cases of class iii) are given in [13].

See also

- ► αBB Algorithm
- ► Continuous Global Optimization: Applications
- ► Continuous Global Optimization: Models, Algorithms and Software
- ► Contraction-mapping
- ► Convex Envelopes in Optimization Problems
- ► Differential Equations and Global Optimization
- ► DIRECT Global Optimization Algorithm
- ► Global Optimization Based on Statistical Models
- ► Global Optimization in Batch Design Under Uncertainty

- ▶ Global Optimization in Binary Star Astronomy
- ▶ Global Optimization in Generalized Geometric Programming
- ▶ Global Optimization of Heat Exchanger Networks
- ▶ Global Optimization in Phase and Chemical Reaction Equilibrium
- ▶ Global Optimization Using Space Filling
- ▶ Interval Analysis: Systems of Nonlinear Equations
- ▶ Interval Global Optimization
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Linear Programming: Heat Exchanger Network Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Nonlinear Least Squares: Newton-type Methods
- ▶ Nonlinear Systems of Equations: Application to the Enclosure of All Azeotropes
- ▶ Smooth Nonlinear Nonconvex Optimization
- ▶ Topology of Global Optimization

References

1. Allgower EL, Georg K (1980) Simplicial and continuation methods for approximating fixed points and solutions to system of equations. SIAM Rev 22:28–85
2. Dennis JE, Schnabel RB (1983) Numerical methods for nonlinear equations and unconstrained optimization. Prentice-Hall, Englewood Cliffs, NJ
3. Forster W (1992) Some computational methods for systems of nonlinear equations and systems of polynomial equations. J Global Optim 2:317–356
4. Gabriel A, Pang J-S (1994) A trust region method for constrained nonsmooth equations. In: Hage WW, Hearn DW, Pardalos PM (eds) Large Scale Optimization: State of the Art. Kluwer, Dordrecht
5. Goffin J-L, Luo Z-Q, Ye Y (1994) On the complexity of a column generation algorithm for convex or quasiconvex feasibility problems. In: Hage WW, Hearn DW, Pardalos PM (eds) Large Scale Optimization: State of the Art. Kluwer, Dordrecht
6. Hansen P, Jaumard B (1995) Lipschitz optimization. In: Horst R, Pardalos PM (eds) Handbook of Global Optimization. Kluwer, Dordrecht
7. Hendrix EMT, Pinter J (1991) An application of Lipschitzian global optimization to product design. J Global Optim 1:389–401
8. Horst R, Nast M, Thoai NV (1997) New LP-bound in multivariate Lipschitz optimization: Theory and applications. J Optim Th Appl 86:369–388
9. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
10. Horst R, Thoai NV (1989) Branch and bound methods for solving systems of Lipschitzian equations and inequalities. J Optim Th Appl 58:139–145
11. Horst R, Thoai NV (1999) DC programming: Overview. J Optim Th Appl 103:1–43
12. Horst R, Tuy H (1993) Global optimization: Deterministic approaches, 2nd edn. Springer, Berlin
13. Maranas CD, Floudas CA (1995) Finding all solutions of nonlinearly constrained systems of equations. J Global Optim 7:143–182
14. Nesterov Y, Nemirovskii A (1994) Interior-point polynomial algorithms in convex programming. SIAM, Philadelphia
15. Pang J-S, Qi L (1993) Nonsmooth equations: Motivation and algorithms. SIAM J Optim 3:443–465
16. Pinter J (1991) Solving nonlinear equation systems via global partition and search. Computing 43:309–323
17. Strongin RG (1992) Algorithms for multi-extremal mathematical programming problems employing the set of joint space-filling curves. J Global Optim 2:357–378
18. Thoai NV (1994) Employment of conical algorithm and outer approximation method in D.C. programming. J Math 22:71–85
19. Tuy H (1995) D.C. optimization: Theory, methods and algorithms. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 149–216

Global Optimization in Multiplicative Programming

TAKAHITO KUNO
University Tsukuba, Ibaraki, Japan

MSC2000: 90C26

Article Outline

Keywords
Linear Multiplicative Program
Convex Multiplicative Program
Other Multiplicative Programs
See also
References

Keywords

Global optimization; Nonconvex minimization; Low-rank nonconvexity; Parametric methods

Multiplicative functions, products of real-valued functions f_i, $i = 1, \ldots, p$, are generally nonconvex functions even though each f_i is convex. As a result, most multiplicative programming problems containing $\prod_{i=1}^{p} f_i(x)$ in the objective and/or constraints are nonconvex minimization; and hence we need global optimization to look for a global minimum in stacks of local minima. Fortunately, however, the number p of f_is in multiplicative functions encountered in practical applications is rather small in comparison with the number n of variables; e. g. two or three in geometrical optimization [10] and at most five in multiple objective optimization [1]. As will be seen later, this enables us to embed the troublesome nonconvexity into a small subspace of dimension p. Exploiting such a property, called *low-rank nonconvexity* [6], a number of researchers have developed efficient algorithms since the late 1980s years to solve various subclasses of multiplicative programming problems, including the *linear multiplicative program*

$$\begin{cases} \min & (\mathbf{c}_1^\top \mathbf{x} + c_{10})(\mathbf{c}_2^\top \mathbf{x} + c_{20}) \\ \text{s.t.} & \mathbf{x} \in D, \end{cases} \tag{1}$$

where $D \subset \mathbf{R}^n$ is a polytope and $\mathbf{c}_i^\top \mathbf{x} + c_i 0 > 0$ for any $\mathbf{x} \in D$; the *convex multiplicative program*

$$\begin{cases} \min & \displaystyle\prod_{i=1}^{p} f_i(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D, \end{cases} \tag{2}$$

where D is a compact convex set and the f_is are convex functions positive-valued on D; the *generalized convex multiplicative program*

$$\begin{cases} \min & \displaystyle\sum_{i=1}^{p} f_{2i-1}(\mathbf{x}) f_{2i}(\mathbf{x}) + g(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D, \end{cases} \tag{3}$$

where D and the f_is are the same as in (2) and g is a convex function; and the convex program with an additional *convex multiplicative constraint*

$$\begin{cases} \min & g(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D \\ & \displaystyle\prod_{i=1}^{p} f_i(\mathbf{x}) \le 1, \end{cases} \tag{4}$$

where D, the f_is and g are the same as in (3). As long as p is a small number, all of these nonconvex programs can be solved in a practical amount of time even if n exceeds a few hundreds.

Linear Multiplicative Program

Problem (1), though simple looking, is *NP*-hard (cf. also ► Complexity theory; ► Complexity classes in optimization) as shown in [11]. There are two major methods, each of which is based on a variant of parametric simplex algorithms for linear programming [12].

The first method introduces a parameter $\xi \ge 0$ and transforms (1) into an equivalent problem:

$$\begin{cases} \min & \xi f_1(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D \\ & f_2(\mathbf{x}) \le \xi, \quad \xi \ge 0, \end{cases} \tag{5}$$

where $f_i(\mathbf{x}) = \mathbf{c}_i^\top \mathbf{x} + c_i 0$, $i = 1, 2$. To solve (5), we need only to solve

$$\min \{ f_1(\mathbf{x}) \colon \ \mathbf{x} \in D, \ f_2(\mathbf{x}) \le \xi \} \tag{6}$$

for all $\xi \ge \xi_{\min} = \min\{f_2(\mathbf{x}) \colon \mathbf{x} \in D\}$, using the *parametric right-hand side simplex algorithm* (cf. also ► Parametric linear programming: Cost simplex algorithm). We then have a set of optimal solutions $\mathbf{x}(\xi)$ to (6) and the analytical expression of

$$\phi(\xi) = \xi f_1(\mathbf{x}(\xi)),$$

which is a piecewise quadratic function over $\xi \ge \xi_{\min}$. Let

$$\xi^* \in \arg\min \{ \phi(\xi) \colon \ \xi \ge \xi_{\min} \}.$$

Then $\mathbf{x}(\xi^*)$ is an optimal solution to (1).

This parametric method, proposed by K. Swarup [13] in the middle 1960s, was originally used for finding a locally optimal solution to (1). Strangely, it had

not been appreciated as an efficient global optimization tool until the second parametric method was developed by H. Konno and T. Kuno [4] more than twenty years later.

The second method also introduces a parameter $\xi \geq 0$, but in a deferent way:

$$\begin{cases} \min & F(\mathbf{x}, \xi) \equiv \xi f_1(\mathbf{x}) + \frac{f_2(\mathbf{x})}{\xi} \\ \text{s.t.} & \mathbf{x} \in D, \quad \xi \geq 0. \end{cases} \tag{7}$$

For any $\mathbf{x}$ we have

$$\min\{F(\mathbf{x}, \xi) : \ \xi \geq 0\} = 2\sqrt{f_1(\mathbf{x}) f_2(\mathbf{x})}.$$

Therefore, (7) is equivalent to (1); moreover, (7) is equivalent to finding a minimum point ξ^* of a function

$$\psi(\xi) = \min\{F(\mathbf{x}, \xi) : \ \mathbf{x} \in D\} \tag{8}$$

over $\xi > 0$. Since the right-hand side of (8) is a linear program, we can locate ξ^* using the *parametric objective simplex algorithm*. In fact, noting that $\lambda = \xi/(\xi + 1/\xi)$ maps $\Xi = \{\xi : \xi > 0\}$ to a unit interval $\{\lambda : 0 < \lambda < 1\}$, we solve

$$\min\left\{\lambda \mathbf{c}_1^\top \mathbf{x} + (1-\lambda)\mathbf{c}_2^\top \mathbf{x} : \ \mathbf{x} \in D\right\} \tag{9}$$

parametrically over $\lambda \in (0, 1)$. Let $\mathbf{x}(\lambda)$ denote an optimal solution to (9). Then

$$\lambda^* \in \arg\min\{f_1(\mathbf{x}(\lambda)) f_2(\mathbf{x}(\lambda)) : \ \lambda \in (0, 1)\}$$

gives $\xi^* = \sqrt{\frac{\lambda^*}{(1-\lambda^*)}}$; and $\mathbf{x}(\lambda^*)$ is an optimal solution to (1).

Under some probabilistic assumptions, the average number of simplex pivots needed to solve a linear program with a single parameter is known to be polynomial in the problem input length [12]. Hence, (1) can also be solved in polynomial time on the average, which contrasts sharply with the result of the worst-case analysis.

Convex Multiplicative Program

The above parametric methods for (1) can be extended to more general classes of multiplicative programming problems. For example, (7) is directly applicable to the special case of (2) where $p = 2$; but it is difficult to design an algorithm for solving (7) parametrically when the f_is are nonlinear functions. One effective approach in this case is branch and bound on the set of parameter values $\Xi = \{\xi : \xi > 0\}$ [7] (cf. also ▶ Integer programming: Branch and bound methods).

Let $\mathcal{F}$ denote the family of functions of the form:

$$\alpha\xi + \frac{\beta}{\xi},$$

where $\alpha, \beta \in \mathbf{R}$. The function ψ defined by (8) is a pointwise minimum of some functions in $\mathcal{F}$ such that $\alpha = f_1(\mathbf{x})$ and $\beta = f_2(\mathbf{x})$ for $\mathbf{x} \in D$. The family $\mathcal{F}$ possesses the following properties:

i) Any two points $(\xi_s, \psi_s), (\xi_t, \psi_t) \in \mathbf{R}^2$, with $0 < \xi_s < \xi_t$, uniquely determine

$$\frac{\psi_s \xi_s - \psi_t \xi_t}{\xi_s^2 - \xi_t^2}\xi + \frac{\psi_s/\xi_s - \psi_t/\xi_t}{1/\xi_s^2 - 1/\xi_t^2}/\xi \in \mathcal{F};$$

ii) Any function in $\mathcal{F}$ is Lipschitz continuous over $\xi \geq \xi'$ for any $\xi' > 0$;

iii) Two distinct functions in $\mathcal{F}$ have at most one intersection point over $\xi > 0$.

Suppose $[\xi_s, \xi_t] \subset \Xi$ is an interval containing ξ^*. Since f_1 and f_2 are convex, $F(\cdot, \xi)$ is also a convex function for any $\xi > 0$; and hence $\psi(\xi_s)$ and $\psi(\xi_t)$ can be computed by convex programming. For $(\xi_s, \psi(\xi_s))$ and $(\xi_t, \psi(\xi_t))$, let us construct a function in $\mathcal{F}$ according to i):

$$u(\xi; \xi_s, \xi_t) = \frac{\psi(\xi_s)\xi_s - \psi(\xi_t)\xi_t}{\xi_s^2 - \xi_t^2}\xi + \frac{\psi(\xi_s)/\xi_s - \psi(\xi_t)/\xi_t}{1/\xi_s^2 - 1/\xi_t^2}/\xi.$$

From iii) we have

$$u(\xi; \xi_s, \xi_t) \leq \psi(\xi), \quad \forall \xi \in [\xi_s, \xi_t].$$

Let $\xi_m \in \arg\min\{u(\xi; \xi_s, \xi_t) : \xi \in [\xi_s, \xi_t]\}$ and

$$u_2(\xi) = \begin{cases} u(\xi; \xi_s, \xi_m) & \text{if } 0 < \xi \leq \xi_m, \\ u(\xi; \xi_m, \xi_t) & \text{if } \xi \geq \xi_m. \end{cases}$$

Then u_2 underestimates ψ over $[\xi_s, \xi_t]$ and is better than $u_1 = u(\cdot; \xi_s, \xi_t)$ in the sense:

$$u_1(\xi) \leq u_2(\xi) \leq \psi(\xi), \quad \forall \xi \in [\xi_s, \xi_t].$$

In this way, as improving the *underestimator* of ψ successively, we can generate the sequence of minimum points of u_ks convergent to ξ^*.

The parametrization (7) can further be extended to (2) with $p \geq 2$ [8] as follows:

$$\begin{cases} \min & F(\mathbf{x}, \boldsymbol{\xi}) \equiv \sum_{i=1}^{p} \xi_i f_i(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D, \\ & \prod_{i=1}^{p} \xi_i \geq 1, \quad \boldsymbol{\xi} \geq 0. \end{cases} \tag{10}$$

Karush–Kuhn–Tucker conditions with respect to $\boldsymbol{\xi}$ imply the equivalence between (2) and (10). Let

$$\psi(\boldsymbol{\xi}) = \min \{F(\mathbf{x}, \boldsymbol{\xi}) : \ \mathbf{x} \in D\}.$$

Then (10) reduces to a problem with p variables:

$$\begin{cases} \min & \psi(\boldsymbol{\xi}) \\ \text{s.t.} & \prod_{i=1}^{p} \xi_i \geq 1, \quad \boldsymbol{\xi} \geq 0. \end{cases} \tag{11}$$

The objective function ψ is concave and coordinatewise nondecreasing; and its value at any $\boldsymbol{\xi} \geq 0$ can be computed by convex programming.

An alternative approach to (2) with $p \geq 2$ [14] is a generalization of (5):

$$\begin{cases} \min & \prod_{i=1}^{p} \xi_i \\ \text{s.t.} & \mathbf{x} \in D, \\ & f_i(\mathbf{x}) \leq \xi_i, \ i = 1, \ldots, p, \\ & \boldsymbol{\xi} \geq 0. \end{cases} \tag{12}$$

Let $W \in \mathbf{R}^n \times \mathbf{R}^p$ denote the feasible region of (12) and

$$\Omega = \left\{\boldsymbol{\xi} \in \mathbf{R}^p : \ \exists \mathbf{x}, \ (\mathbf{x}, \boldsymbol{\xi}) \in W\right\}.$$

Then (12) also reduces to a problem with p variables:

$$\begin{cases} \min & \sum_{i=1}^{p} \log \xi_i \\ \text{s.t.} & \boldsymbol{\xi} \in \Omega. \end{cases} \tag{13}$$

The objective function is concave; the feasible region Ω is a projection of the convex set W and hence a convex set.

Both (11) and (13) are concave minimization problems (cf. also ▶ Concave programming); however, even general-purpose algorithms such as branch and bound and outer approximation (cf. also ▶ Generalized outer approximation) [3] can handle them very efficiently when p is less than five.

Other Multiplicative Programs

In a way similar to (11), problem (3) can reduce to a concave minimization problem with $2p$ variables [5] through a parametrization:

$$\begin{cases} \min & \sum_{i=1}^{p} \dfrac{\xi_{2i-1}(f_{2i-1}(\mathbf{x}))^2 + \xi_{2i}(f_{2i}(\mathbf{x}))^2}{2} \\ & \quad + g(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D \\ & \xi_{2i-1}\xi_{2i} \geq 1, \ i = 1, \ldots, p, \\ & \boldsymbol{\xi} \geq 0. \end{cases} \tag{14}$$

Let $\psi(\boldsymbol{\xi})$ denote the optimal value of (14) with fixed $\boldsymbol{\xi}$. Then (14) reduces to

$$\begin{cases} \min & \psi(\boldsymbol{\xi}) \\ \text{s.t.} & \xi_{2i-1}\xi_{2i} \geq 1, \ i = 1, \ldots, p, \\ & \boldsymbol{\xi} \geq 0. \end{cases} \tag{15}$$

The objective function ψ is concave and coordinatewise nondecreasing. For problem (4), we can use the following parametrization [9]:

$$\begin{cases} \min & g(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D \\ & f_i(\mathbf{x}) \leq \xi_i, \ i = 1, \ldots, p, \\ & \prod_{i=1}^{p} \xi_i \leq 1, \quad \boldsymbol{\xi} \geq 0. \end{cases} \tag{16}$$

Let $\psi(\boldsymbol{\xi})$ denote the optimal value of (16) with fixed $\boldsymbol{\xi}$. Then (16) is equivalent to

$$\begin{cases} \min & \psi(\boldsymbol{\xi}) \\ \text{s.t.} & \prod_{i=1}^{p} \xi_i \leq 1, \quad \boldsymbol{\xi} \geq 0. \end{cases} \tag{17}$$

The objective function ψ is convex; but the feasible region is a *d.c. set* (difference of two convex sets). Thus, we can solve (3) and (4) by solving smaller-size problems (15) and (17), respectively. For a more complete survey of the algorithms, see the article by Konno and Kuno in [2].

We have seen that the parametric approach offers an efficient tool to handle multiplicative programming problems. This approach is not specific to the mul-

tiplicative structure but can be extended to a much wider class of nonconvex minimization problems, including minimum concave-cost flow problems, facility location, multilevel programming and so forth. The textbook [6] shows how the parametric approach can be generalized to a broad class of problems.

See also

- ► Linear Programming
- ► Multiparametric Linear Programming
- ► Multiplicative Programming
- ► Parametric Linear Programming: Cost Simplex Algorithm

References

1. Geoffrion M (1967) Solving bicriterion mathematical programs. Oper Res 15:39–54
2. Horst R, Pardalos PM (1995) Handbook of global optimization. Kluwer, Dordrecht
3. Horst R, Tuy H (1993) Global optimization: deterministic approaches, 2nd edn. Springer, Berlin
4. Konno H, Kuno T (1992) Linear multiplicative programming. Math Program 56:51–64
5. Konno H, Kuno T, Yajima Y (1994) Global minimization of a generalized convex multiplicative function. J Global Optim 4:47–62
6. Konno H, Thach PT, Tuy H (1997) Optimization on low rank nonconvex structures. Kluwer, Dordrecht
7. Kuno T, Konno H (1991) A parametric successive underestimation method for convex multiplicative programming problems. J Global Optim 1:267–285
8. Kuno T, Yajima Y, Konno H (1993) An outer approximation method for minimizing the product of several convex functions on a convex set. J Global Optim 3:325–335
9. Kuno T, Yajima Y, Yamamoto Y, Konno H (1994) Convex program with an additional constraint on the product of several convex functions. Europ J Oper Res 77:314–324
10. Maling K, Mueller SH, Heller WR (1982) On finding most optimal rectangular package plans. In: Proc. 19th Design Automation Conf, pp 663–670
11. Matsui T (1996) NP-hardness of linear multiplicative programming and related problems. J Global Optim 9:113–119
12. Schrijver A (1986) Theory of linear and integer programming. Wiley, New York
13. Swarup K (1966) Programming with indefinite quadratic function with linear constraints. Cahiers CERO 8:132–136
14. Thoai NV (1991) A global optimization approach for solving the convex multiplicative programming problem. J Global Optim 1:341–357

Global Optimization: p-αBB Approach

CHRYSANTHOS E. GOUNARIS, CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering, Princeton University, Princeton, USA

MSC2000: 49M37, 65K10, 90C26, 90C30, 46N10, 47N10

Article Outline

Keywords and Phrases
Introduction
An α Spline Underestimator
Nonconcave Perturbation
Illustrative Example
References

Keywords and Phrases

Convex underestimators; αBB; Global optimization

Introduction

Various deterministic global optimization algorithms that utilize a branch and bound framework make use of convex underestimators of the functions under consideration. This entry presents the work of Meyer and Floudas [11] on the convex underestimation of C^2-continuous functions. The work extends and refines the convex underestimation approach used in the αBB global optimization algorithm [1,2,3,4,10]. A recent review of deterministic global optimization approaches can be found in [6].

Let $f : \mathcal{R}^n \to \mathcal{R}$ be a smooth nonconvex C^2-continuous function. Its convex underestimator $\phi : \mathcal{R}^n \in x \to \mathcal{R}$ is defined as:

$$\phi(x) := f(x) - q(x) \tag{1}$$

where $q : \mathcal{R}^n \to \mathcal{R}$ is some perturbation function.

In the classical αBB approach, a series of simplifications are made to yield an efficient convexification procedure. The first of these simpifications is the imposition of a quadratic structure on the perturbation func-

tion:

$$q(x) := \sum_{i=1}^{n} \alpha_i \left(\overline{x}_i - x_i\right)\left(x_i - \underline{x}_i\right) . \tag{2}$$

To ensure that $q(x)$ is nonnegative, α is assumed to be nonnegative. Observe that $q(x)$, a quadratic function with a diagonal Hessian matrix $\nabla^2 q(x) := 2\,\mathrm{diag}(\alpha)$ has an eigenvalue–eigenvector structure that is uniform over the entire domain $\mathbf{x}$ with eigenvectors that are aligned with the coordinate axes. In the work of Adjiman et al. [2], a second simplification is introduced in which the interval extension, $\mathbf{H}^{\mathbf{x}}$, is used instead of $\nabla^2 f(x)$ itself. The interval extension of the matrix $\nabla^2 f(x) \in \mathcal{R}^{n\times n}$ is a matrix of intervals of $\mathcal{R}$. Each element $\mathbf{H}^{\mathbf{x}}{}_{ij}$ of the matrix $\mathbf{H}^{\mathbf{x}}$ is defined in such a way that

$$\left.\frac{\partial^2 f}{\partial x_i \partial x_j}\right|_x \in \mathbf{H}^{\mathbf{x}}_{ij} \qquad \text{for all } x \in \mathbf{x}.$$

Computing the tightest possible interval extension is in itself a global optimization problem. In practice, an interval extension can be calculated using interval arithmetic [12,14,16]. The overestimation made in the interval calculations may result in a significant loss of accuracy. Adjiman et al. [2] applied the work of [5,7,8, 9,13,15,17,18], and devised various methods to compute α vectors that guarantee the convexity of the underestimators. The tightness of the underestimators is dependent on the particular α calculation method used. Extensive computational testing [1] showed that the method based on the scaled Gerschgorin theorem performs better in practice.

In the work of Meyer and Floudas [11], the form of the αBB perturbation function and the way in which it is calculated are reexamined, a novel spline based method for convex underestimation is proposed and an efficient means of computing these tighter underestimators is elucidated.

An α Spline Underestimator

The size of the domain $\mathbf{x}$ affects the result of every step in the α calculation and strongly influences the tightness of the resulting convex underestimator. In particular, reducing $\mathbf{x}$ reduces the mismatch between the assumed quadratic functional form and the ideal form; it reduces the overestimation in the interval extension of the Hessian matrix; and the maximum separation distance has been shown to be a quadratic function of interval length [10]. It is therefore useful to construct a convex underestimator using a number of different α vectors, each applying to a subregion of the full domain $\mathbf{x}$.

Let $f(x) : \mathcal{R}^n \to \mathcal{R}$ be a C^2-continuous function. For each variable $x_i \in \mathcal{R}$, let the interval $[\underline{x}_i, \overline{x}_i]$ be partitioned into N_i subintervals. The endpoints of these subintervals are denoted with $x_i^0, x_i^1, \cdots, x_i^{N_i}$, where $\underline{x}_i = x_i^0 < x_i^1 < \cdots < x_i^k < \cdots < x_i^{N_i} = \overline{x}_i$. In this notation, the kth interval is $[x_i^{k-1}, x_i^k]$. A smooth convex underestimator of $f(x)$ over $\mathbf{x}$ is defined by (1). The new perturbation function, $q(x)$, would be:

$$q(x) := \sum_{i=1}^{n} q_i^k\left(x_i\right) \qquad \text{for } x_i \in \left[x_i^{k-1}, x_i^k\right] , \tag{3}$$

$$q_i^k\left(x_i\right) := \alpha_i^k \left(x_i - x_i^{k-1}\right) \cdot \left(x_i^k - x_i\right) + \beta_i^k x_i + \gamma_i^k . \tag{4}$$

In each interval $[x_i^{k-1}, x_i^k]$, $\alpha_i^k \geq 0$ is chosen such that $\nabla^2\phi(x)$, the Hessian matrix of $\phi(x)$, is positive semi-definite for all members of the set $\{x \in \mathbf{x} : x_i \in [x_i^{k-1}, x_i^k]\}$. $q_i^k(x_i)$ is the quadratic function associated with variable i in interval k. The function $q(x)$ is a piecewise quadratic function contructed from the functions $q_i^k(x_i)$.

The continuity and smoothness properties of $q(x)$ are produced in a spline-like manner. For $q(x)$ to be smooth the q_i^k functions and their gradients must match at the endpoints x_i^k. In addition, we require that $q(x) = 0$ at the vertices of the hyperrectangle $\mathbf{x}$. To satisfy these requirements, the following conditions are imposed for all $i = 1, \ldots, n$:

$$\begin{aligned}
q_i^1\left(x_i^0\right) &= 0 \\
q_i^k\left(x_i^k\right) &= q_i^{k+1}\left(x_i^k\right) \quad \forall k = 1, \ldots, N_i - 1 \\
q_i^{N_i}\left(x_i^{N_i}\right) &= 0 \\
\left.\frac{\mathrm{d}q_i^k}{\mathrm{d}x_i}\right|_{x_i^k} &= \left.\frac{\mathrm{d}q_i^{k+1}}{\mathrm{d}x_i}\right|_{x_i^k} \quad \forall k = 1, \ldots, N_i - 1
\end{aligned} \tag{5}$$

Expanding these equations for each $i = 1, \ldots, n$, one obtains the following system of equations:

$$\begin{aligned}
\beta_i^1 x_i^0 + \gamma_i^1 &= 0 \\
\beta_i^k x_i^k + \gamma_i^k &= \beta_i^{k+1} x_i^k + \gamma_i^{k+1} \\
&\qquad \forall k = 1, \ldots, N_i - 1 \\
\beta_i^{N_i} x_i^{N_i} + \gamma_i^{N_i} &= 0 \\
-\alpha_i^k \left(x_i^k - x_i^{k-1} \right) + \beta_i^k &= \alpha_i^{k+1} \left(x_i^{k+1} - x_i^k \right) + \beta_i^{k+1} \\
&\qquad \forall k = 1, \ldots, N_i - 1
\end{aligned} \tag{6}$$

which can be represented as:

$$\begin{bmatrix}
-x_i^0 & & & & & -1 & & & & \\
x_i^1 & -x_i^1 & & & & 1 & -1 & & & \\
& \ddots & \ddots & & & & \ddots & \ddots & & \\
& & x_i^k & -x_i^k & & & & 1 & -1 & \\
& & & \ddots & \ddots & & & & \ddots & \ddots \\
& & & & x_i^{N_i} & & & & & 1 \\
-1 & 1 & & & & & & & & \\
& -1 & 1 & & & & & & & \\
& & \ddots & \ddots & & & & & & \\
& & & -1 & 1 & & & & &
\end{bmatrix}
\begin{bmatrix}
\beta_i^1 \\ \beta_i^2 \\ \vdots \\ \beta_i^k \\ \vdots \\ \beta_i^{N_i} \\ \gamma_i^1 \\ \gamma_i^2 \\ \vdots \\ \gamma_i^{N_i}
\end{bmatrix}$$

$$= \begin{bmatrix}
0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ s_1 \\ s_2 \\ \vdots \\ s_i^{N_i - 1}
\end{bmatrix} \tag{7}$$

where $s_i^k = -\alpha_i^k (x_i^k - x_i^{k-1}) - \alpha_i^{k+1} (x_i^{k+1} - x_i^k)$.

The solution of the above linear system of equations is:

$$\beta_i^1 = \frac{\sum_{k=1}^{N_i - 1} s_i^k \left(x_i^k - x_i^{N_i} \right)}{x_i^{N_i} - x_i^0}$$

$$\beta_i^k = \beta_i^1 + \sum_{j=1}^{k-1} s_i^j \qquad \forall k = 2, \ldots, N_i \tag{8}$$

$$\gamma_i^k = -\beta_i^1 x_i^0 - \sum_{j=1}^{k-1} s_i^j x_i^j \qquad \forall k = 1, \ldots, N_i$$

For a rigorous proof of the continuity, smoothness, convexity and underestimation properties of underestimator $\phi(x)$, see [11].

Nonconcave Perturbation

Consider a function $f(x)$ which is convex in one subdomain and concave in another. In the α spline approach, $\phi(x)$ can be convex even if the α values are negative in the regions in which $f(x)$ is strictly convex. In the classical αBB underestimator, the underestimation property is guaranteed by the concavity of $q(x)$, as given in (2). The concavity of $q(x)$ is, in turn, a result of the non-negativity of the α values. In this section, we discuss how the underestimation property of $\phi(x)$ can be maintained when some α values are negative.

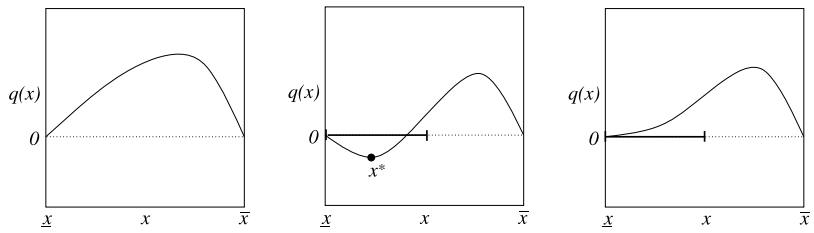
The underestimation property, $\phi(x) \leq f(x)$ for all $x \in \mathbf{x}$, is ensured by the following condition:

$$\min_{x \in \mathbf{x}} q(x) \geq 0 \tag{9}$$

Instead of solving minimization problems, the key idea is to adjust the α's to prevent the creation of local minima at any nonvertex point in $\mathbf{x}$ by prohibiting the occurrence of stationary points on convex regions of the perturbation function. This is illustrated in Fig. 1. In Fig. 1a, a concave perturbation function is depicted. The non-negativity of this function follows from its concavity. In Fig. 1b, a perturbation function is shown which is convex over the domain marked with a bold line.

The point x^* is a stationary point of q in this convex region and we note that $q(x^*)$ is negative. In Fig. 1c, the perturbation function is again convex over the marked region but there is no stationary point in this region. This function is non-negative over the entire domain $[\underline{x}, \overline{x}]$.

Using this idea, Meyer and Floudas [11] derived a tight convex underestimator by starting with $q(x)$, with non-negative α values as defined in Sect. "An α Spline Underestimator", and making the zero α's negative, one at a time, while maintaining the convexity



Global Optimization: p-αBB Approach, Figure 1
(a) Concave, (b) nonconcave, and (c) nonnegative nonconcave perturbation functions

of $\phi(x)$ and avoiding the generation of stationary points on the convex portions of $q(x)$. For the rest of this section we will assume that $f : \mathbf{x} \to \mathcal{R}$ is a univariate function, $\mathbf{x} = [\underline{x}, \overline{x}] \subset \mathcal{R}$. The separable structure of the α spline function allows the techniques developed here to be applied to the multivariate case.

Note that the β and γ parameters defining $q(x)$ are functions of the α's and the endpoints, $x^0, \ldots, x^N$. The following formula, derived from (8), is an expression for β^k in terms of $\alpha^1, \ldots, \alpha^N$.

$$\begin{aligned}\beta^k = \quad & \frac{1}{x^N - x^0} \sum_{j=1}^{k-1} \Big(-\alpha^j \left(x^j - x^{j-1}\right)\left(x^j - x^0\right) \\ & \qquad - \alpha^{j+1}\left(x^{j+1} - x^j\right)\left(x^j - x^0\right)\Big) \\ & + \frac{1}{x^N - x^0} \sum_{j=k}^{N-1} \Big(-\alpha^j \left(x^j - x^{j-1}\right)\left(x^j - x^N\right) \\ & \qquad - \alpha^{j+1}\left(x^{j+1} - x^j\right)\left(x^j - x^N\right)\Big)\end{aligned} \tag{10}$$

Suppose that having calculated $\beta \in \mathcal{R}^N$ for some given $\alpha \in \mathcal{R}^N$, we wish to modify some element α^j. Meyer and Floudas [11] derived formulae that may be used to update the β's following such an α update. Under the substitution $\alpha^j \to \tilde{\alpha}^j$, the elements $\tilde{\beta}^1, \ldots, \tilde{\beta}^N$ that satisfy (8) may be expressed in terms of $\beta^1, \ldots, \beta^N$, α^j and $\tilde{\alpha}^j$ using the following update formulae:

$$\begin{aligned}\tilde{\beta}^k - \beta^k &= \frac{1}{x^N - x^0}\left(\alpha^j - \tilde{\alpha}^j\right)\left(x^j - x^{j-1}\right)\left(x^{j-1} - x^0\right) \\ &\quad + \frac{1}{x^N - x^0}\left(\alpha^j - \tilde{\alpha}^j\right)\left(x^j - x^{j-1}\right)\left(x^j - x^0\right) \\ &= \frac{1}{x^N - x^0}\left(\alpha^j - \tilde{\alpha}^j\right)\left(x^j - x^{j-1}\right) \\ &\qquad \times \left(x^{j-1} + x^j - 2x^0\right) \qquad \text{if } j < k\end{aligned} \tag{11}$$

$$\begin{aligned}\tilde{\beta}^k - \beta^k &= \frac{1}{x^N - x^0}\left(\alpha^j - \tilde{\alpha}^j\right)\left(x^j - x^{j-1}\right)\left(x^{j-1} - x^0\right) \\ &\quad + \frac{1}{x^N - x^0}\left(\alpha^j - \tilde{\alpha}^j\right)\left(x^j - x^{j-1}\right)\left(x^j - x^N\right) \\ &= \frac{1}{x^N - x^0}\left(\alpha^j - \tilde{\alpha}^j\right)\left(x^j - x^{j-1}\right) \\ &\qquad \times \left(x^{j-1} + x^j - x^0 - x^N\right) \qquad \text{if } j = k\end{aligned} \tag{12}$$

$$\begin{aligned}\tilde{\beta}^k - \beta^k &= \frac{1}{x^N - x^0}\left(\alpha^j - \tilde{\alpha}^j\right)\left(x^j - x^{j-1}\right) \\ &\qquad \cdot \left(x^{j-1} - x^N\right) \\ &\quad + \frac{1}{x^N - x^0}\left(\alpha^j - \tilde{\alpha}^j\right)\left(x^j - x^{j-1}\right) \\ &\qquad \cdot \left(x^j - x^N\right) \\ &= \frac{1}{x^N - x^0}\left(\alpha^j - \tilde{\alpha}^j\right)\left(x^j - x^{j-1}\right) \\ &\qquad \times \left(x^{j-1} + x^j - 2x^N\right) \qquad \text{if } j > k\end{aligned} \tag{13}$$

A stationary point x^* of the function $q : \mathcal{R} \to \mathcal{R}$ is one that satisfies:

$$\left.\frac{\mathrm{d}q}{\mathrm{d}x}\right|_{x^*} = 0 \Leftrightarrow \alpha^k\left(x^k + x^{k-1} - 2x^*\right) + \beta^k = 0$$

in some interval $x^* \in [x^{k-1}, x^k]$. It follows that an interval k contains no stationary point if either $\frac{1}{2}(x^k + x^{k-1} + \beta^k/\alpha^k) > x^k$ or $\frac{1}{2}(x^k + x^{k-1} + \beta^k/\alpha^k) < x^{k-1}$.

Meyer and Floudas [11] derived conditions on α^j that guarantee the absence of such stationary points. Their results are summarized in the following three Lemmas, which correspond to cases $j < k$, $j = k$ and $j > k$, respectively.

Lemma 1 *Consider two intervals* $[x^{j-1}, x^j]$ *and* $[x^{k-1}, x^k]$ *where* $j < k$. *Let the sequence of* α *values*

defining $q^k(x)$ be

$$\{\alpha^1, \ldots, \alpha^j, \ldots, \alpha^k, \ldots, \alpha^{N-1}\},$$

where $\alpha^k < 0$. Let $\tilde{q}^k(x)$ be the function defined by the sequence of α values

$$\{\alpha^1, \ldots, \tilde{\alpha}^j, \ldots, \alpha^k, \ldots, \alpha^{N-1}\},$$

where $\tilde{\alpha}^j < 0$. There exists no stationary point of $\tilde{q}^k(x)$ on the interval $[x^{k-1}, x^k]$ if either of the following bounds on $\tilde{\alpha}^j$ hold:

$$\tilde{\alpha}^j > \frac{\left(x^N - x^0\right)\left(-\alpha^k\left(x^k - x^{k-1}\right) + \beta^k\right)}{\left(x^j - x^{j-1}\right)\left(x^j + x^{j-1} - 2x^0\right)} + \frac{\alpha^j\left(x^j - x^{j-1}\right)\left(x^{j-1} + x^j - 2x^0\right)}{\left(x^j - x^{j-1}\right)\left(x^j + x^{j-1} - 2x^0\right)},$$

or

$$\tilde{\alpha}^j < \frac{\left(x^N - x^0\right)\left(\alpha^k\left(x^k - x^{k-1}\right) + \beta^k\right)}{\left(x^j - x^{j-1}\right)\left(x^j + x^{j-1} - 2x^0\right)} + \frac{\alpha^j\left(x^j - x^{j-1}\right)\left(x^{j-1} + x^j - 2x^0\right)}{\left(x^j - x^{j-1}\right)\left(x^j + x^{j-1} - 2x^0\right)}.$$

Lemma 2 *Consider an interval $[x^{k-1}, x^k]$. Let $\{\alpha^1, \alpha^2, \ldots, \alpha^{N-1}\}$ be the sequence of α values determining $q^k(x)$. Let $\tilde{q}^k(x)$ be the function defined by the sequence of α values*

$$\{\alpha^1, \ldots, \alpha^{k-1}, \tilde{\alpha}^k, \alpha^{k+1}, \ldots, \alpha^{N-1}\}$$

where $\tilde{\alpha}^k < 0$. A stationary point of $\tilde{q}(x)$ does not exist on the interval $[x^{k-1}, x^k]$ if either of the following conditions hold:

$$\begin{aligned} \tilde{\alpha}^k &> \frac{\zeta}{\left(x^k - x^{k-1}\right)\left(x^k + x^{k-1} - 2x^0\right)} \quad \text{if } \zeta \le 0 \\ \tilde{\alpha}^k &> \frac{\zeta}{\left(x^k - x^{k-1}\right)\left(x^k + x^{k-1} - 2x^N\right)} \quad \text{if } \zeta > 0 \end{aligned} \tag{14}$$

where

$$\zeta = \beta^k\left(x^N - x^0\right) + \alpha^k\left(x^k - x^{k-1}\right)\left(x^{k-1} + x^k - x^0 - x^N\right).$$

Lemma 3 *Consider two intervals $[x^{j-1}, x^j]$ and $[x^k, x^{k-1}]$ where $j > k$. Let $\alpha^k < 0$, and $\{\alpha^1, \ldots, \alpha^k, \ldots, \alpha^j, \ldots, \alpha^{N-1}\}$ be the sequence of α values determining $q^k(x)$. Let $\tilde{q}^k(x)$ be the function defined by the sequence of α values $\{\alpha^1, \ldots, \alpha^k, \ldots, \tilde{\alpha}^j, \ldots, \alpha^{N-1}\}$ where $\tilde{\alpha}^j < 0$. A stationary point of $\tilde{q}^k(x)$ does not exist on the interval $[x^{k-1}, x^k]$ if either of the following bounds on $\tilde{\alpha}^j$ hold:*

$$\tilde{\alpha}^j > \frac{\left(x^N - x^0\right)\left(\alpha^k\left(x^k - x^{k-1}\right) + \beta^k\right)}{\left(x^j - x^{j-1}\right)\left(x^j + x^{j-1} - 2x^N\right)} + \frac{\alpha^j\left(x^j - x^{j-1}\right)\left(x^{j-1} + x^j - 2x^N\right)}{\left(x^j - x^{j-1}\right)\left(x^j + x^{j-1} - 2x^N\right)},$$

$$\tilde{\alpha}^j < -\frac{\left(x^N - x^0\right)\left(\alpha^k\left(x^k - x^{k-1}\right) - \beta^k\right)}{\left(x^j - x^{j-1}\right)\left(x^j + x^{j-1} - 2x^N\right)} + \frac{\alpha^j\left(x^j - x^{j-1}\right)\left(x^{j-1} + x^j - 2x^N\right)}{\left(x^j - x^{j-1}\right)\left(x^j + x^{j-1} - 2x^N\right)}.$$

When $q(x)$ is concave on a set of intervals and is guaranteed to have no stationary point on the remainder of the intervals, $q(x)$ is monotonically nondecreasing between x^0 and a global maximum x^* and monotonically nonincreasing between x^* and x^N. Under the aforementioned conditions, the perturbation function $q(x)$ is always non-negative and, thus, $\phi(x)$ is a valid underestimator of $f(x)$ [11].

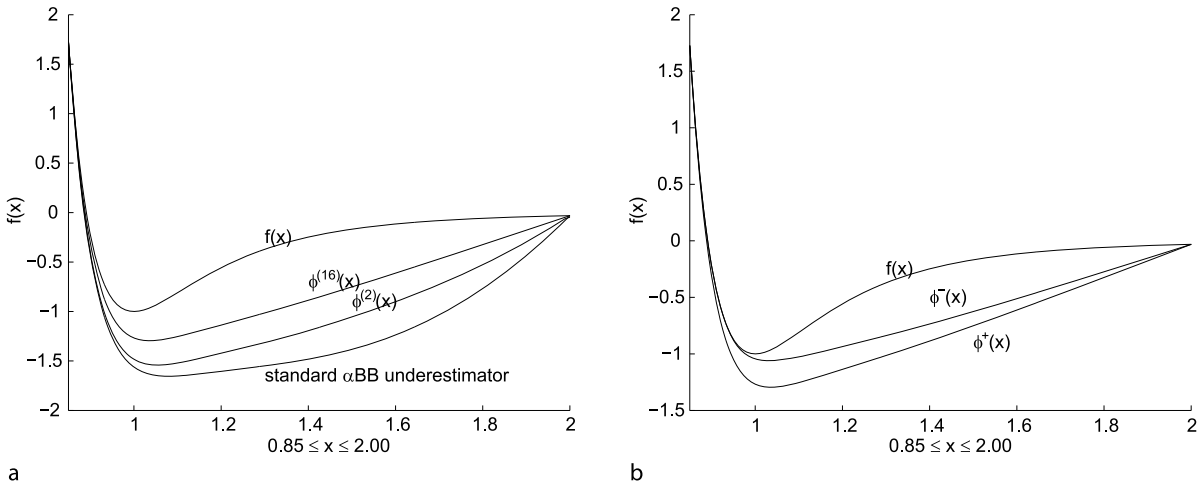
Illustrative Example

As an illustration, we present here an example from Meyer and Floudas [11]. It involves the well-known Lennard–Jones potential energy function:

$$f(x) = \frac{1}{x^{12}} - \frac{2}{x^6}$$

in the interval $[\underline{x}, \overline{x}] = [0.85, 2.00]$. The first term of this function is a convex function and dominates when x is small, while the second term is a concave function which dominates when x is large. The minimum eigenvalue of this function in an interval $[\underline{x}, \overline{x}]$ can be calculated explicitly as follows:

$$\min f'' = \begin{cases} \dfrac{156}{\overline{x}^{14}} - \dfrac{84}{\overline{x}^8} & \text{if } \overline{x} \le 1.21707 \\ -7.47810 & \text{if } [\underline{x}, \overline{x}] \ni 1.21707 \\ \dfrac{156}{\underline{x}^{14}} - \dfrac{84}{\underline{x}^8} & \text{if } \underline{x} \ge 1.21707. \end{cases}$$



Global Optimization: p-αBB Approach, Figure 2
Lennard–Jones convex underestimators with (a) concave and (b) nonconcave perturbations

Global Optimization: p-αBB Approach, Table 1
Parameters for 2 subinterval perturbation for Lennard–Jones function

k	x^k	min f''	α^k	β^k	γ^k
0	0.850				
1	1.425	−7.47810	3.73905	1.62764	−1.38349
2	2.000	−3.84462	1.92231	−1.62764	3.25528

Global Optimization: p-αBB Approach, Table 2
Parameters for 16 subinterval perturbation of Lennard–Jones function

k	x^k	min f''	α^k	β^k	γ^k
0	0.850000				
1	0.921875	326.18127	0.00000	1.78326	−1.51577
2	0.993750	81.99112	0.00000	1.78326	−1.51577
3	1.065625	13.55346	0.00000	1.78326	−1.51577
4	1.137500	−4.27629	2.13815	1.62958	−1.35200
5	1.209375	−7.46047	3.73024	1.20779	−0.87222
6	1.281250	−7.47810	3.73905	0.67093	−0.22296
7	1.353125	−6.71098	3.35549	0.16101	0.43038
8	1.425000	−5.21291	2.60645	−0.26750	1.01021
9	1.496875	−3.84462	1.92231	−0.59301	1.47405
10	1.568750	−2.78248	1.39124	−0.83117	1.83055
11	1.640625	−2.00473	1.00236	−1.00321	2.10044
12	1.712500	−1.44791	0.72395	−1.12729	2.30401
13	1.784375	−1.05201	0.52600	−1.21713	2.45786
14	1.856250	−0.77029	0.38515	−1.28262	2.57472
15	1.928125	−0.56887	0.28443	−1.33074	2.66405
16	2.000000	−0.42385	0.21192	−1.36642	2.73284

Global Optimization: p-αBB Approach, Table 3
Parameters defining nonconcave perturbations for the Lennard–Jones potential

k	x^k	min f''	α^k	β^k	γ^k
0	0.850000				
1	0.921875	326.18127	0.00000	0.00000	0.00000
2	0.993750	81.99112	−7.37920	0.53038	−0.48894
3	1.065625	13.55346	−6.77673	1.54784	−1.50004
4	1.137500	−4.27629	2.13815	1.88124	−1.85532
5	1.209375	−7.46047	3.73024	1.45945	−1.37553
6	1.281250	−7.47810	3.73905	0.92259	−0.72627
7	1.353125	−6.71098	3.35549	0.41267	−0.07294
8	1.425000	−5.21291	2.60645	−0.01584	0.50689
9	1.496875	−3.84462	1.92230	−0.34135	0.97074
10	1.568750	−2.78248	1.39124	−0.57951	1.32724
11	1.640625	−2.00473	1.00236	−0.75155	1.59713
12	1.712500	−1.44791	0.72395	−0.87563	1.80069
13	1.784375	−1.05201	0.52600	−0.96547	1.95454
14	1.856250	−0.77029	0.38515	−1.03096	2.07140
15	1.928125	−0.56887	0.28443	−1.07909	2.16074
16	2.000000	−0.42385	0.21192	−1.11476	2.22952

The classical αBB underestimator for this function and interval is $f(x) - \frac{7.47810}{2}(\overline{x} - x)\left(x - \underline{x}\right)$. Bisecting the domain and applying (8), we obtain a convex underestimator defined by the parameters in Table 1.

Partitioning the domain into 16 equal sized subintervals and applying (8), we obtain the convex under-

estimator $\phi(x)$ with the parameters defining $q(x)$ of Table 2.

The potential energy function, the classical αBB underestimator, and the $\phi(x)$ underestimators are shown in Fig. 2a. In this figure, the α spline underestimator based on 2 subregions is denoted as $\phi^{(2)}$, while that based on 16 subregions is denoted as $\phi^{(16)}$.

Figure 2b depicts the strengthening of an underestimation function through the use of nonconcave perturbations. A negative α value has been assigned to two of the three regions in which the second derivative is strictly positive, as shown in Table 3. The resulting underestimator is depicted as $\phi^{-}(x)$, while the notation $\phi^{+}(x)$ is used to depict the underestimator with no negative α's (same as $\phi^{(16)}$ in Fig. 2a).

References

1. Adjiman CS, Androulakis IP, Floudas CA (1998) A Global Optimization Method, αBB, for General Twice-Differentiable Constrained NLPs II. Implementation and Computational Results. Comput Chem Eng 22:1159–1179
2. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A Global Optimization Method, αBB, for General Twice-Differentiable Constrained NLPs I. Theoretical Advances. Comput Chem Eng 22:1137–1158
3. Adjiman CS, Floudas CA (1996) Rigorous Convex Underestimators for General Twice-Differentiable Problems. J Glob Optim 9:23–40
4. Androulakis IP, Maranas CD, Floudas CA (1995) αBB: A Global Optimization Method for General Constrained Nonconvex Problems. J Glob Optim 7:337–363
5. Deif AS (1991) The Interval Eigenvalue Problem. Z Angew Math Mech 71:61–64
6. Floudas CA, Akrotirianakis IG, Caratzoulas S, Meyer CA, Kallrath J (2005) Global Optimization in the 21st Century: Advances and Challenges. Comput Chem Eng 29:1185–1202
7. Gerschgorin S (1931) Über die Abgrenzung der Eigenwerte einer Matrix. Izv Akad Nauk SSSR Ser Mat 6:749–754
8. Hertz D (1992) The Extreme Eigenvalues and Stability of Real Symmetric Interval Matrices. IEEE Trans Autom Control 37:532–535
9. Kharitonov VL (1979) Asymptotic Stability of an Equilibrium Position of a Family of Systems of Linear Differential Equations. Differ Equations 78:1483–1485
10. Maranas CD, Floudas CA (1994) Global Minimum Potential Energy Conformations of Small Molecules. J Glob Optim 4:135–170
11. Meyer CA, Floudas CA (2005) Convex Underestimation of Twice Continuously Differentiable Functions by Piecewise Quadratic Perturbation : Spline αBB Underestimators. J Glob Optim 32:221–258
12. Moore RE (1966) Interval Analysis. Prentice Hall, Englewood Cliffs
13. Mori T, Kokame H (1994) Eigenvalue Bounds for a Certain Class of Interval Matrices. IEICE Trans Fundam 10: 1707–1709
14. Neumaier A (1990) Interval Methods for Systems of Equations. Cambridge University Press, Cambridge
15. Neumaier A (1992) An Optimality Criterion for Global Quadratic Optimization. J Glob Optim 2:201–208
16. Ratschek H, Rokne J (1984) Computer Methods for the Range of Functions. Ellis Horwood Limited, Chichester
17. Rohn J (1994) Bounds on Eigenvalues of Interval Matrices. In: Institute of Computer Science. Technical Report no.688. Academy of Sciences, Prague
18. Stephens C, Bomze IM, Csendes T, Horst R, Pardalos PM (1997) Interval and Bounding Hessians. In: Bomze IM (ed) Developments in Global Optimization. Kluwer, Dordrecht, pp 109–199

Global Optimization in Phase and Chemical Reaction Equilibrium

Conor M. McDonald
E.I. DuPont de Nemours & Co., Wilmington, USA

MSC2000: 90C26, 90C90

Article Outline

Keywords
Problem Statement
Thermodynamic Models
- Liquid Phases
- The Wilson Equation
- Regular Solutions
- The NRTL Equation
- The UNIQUAC Equation
- The UNIFAC Equation
- The ASOG Equation
- Vapor Phases

Obtaining Equilibrium Solutions
- Equation Based Approaches
- Local Optimization
- Global Optimization

Verifying Equilibrium Solutions
- Equation Based Approaches
- Global Optimization

Combining Approaches
- Global Optimization

Reaction Equilibria
General Comments on Efficiency

See also
References

Keywords

Global optimization; Phase equilibrium; Phase stability problem

The prediction of the behavior of fluid mixtures is a fundamental aspect of chemical process engineering. The physico-chemical problem of computing solutions to the phase and chemical equilibrium problem is central to the design, control and operation of many important processes. These include distillation (standard and azeotropic), extraction trains, petroleum reservoirs and applications involving gases at high pressure. The ubiquity of the flash calculation in chemical engineering is just one example of its prevalence. Because the properties of many fluids vary in a complex fashion, the thermodynamic models that have arisen to describe their behavior create some difficulties in their application. These challenges will be explored in this article.

Problem Statement

The equilibrium condition is characterized by an extremum of some thermodynamic condition. Most commonly, the focus is on systems that attain equilibrium states under conditions of constant pressure (P) and temperature (T) where the global minimum value of the Gibbs free energy describes the true equilibrium state. The problem may be stated as follows:

> Given C components participating in up to P potential phases under isothermal and isobaric conditions find the number of phases and the distribution of components in those phases that yield the global minimum of the Gibbs free energy.

The requisite material balance constraints must also be satisfied. In what follows, all quantities associated with the Gibbs free energy are treated as dimensionless by dividing by RT, where R is the universal gas constant. The total Gibbs free energy is given by the summation of the molar Gibbs free energies for each phase:

$$G = \sum_{k \in P} n^k g^k = \sum_{k \in P} G^k,$$

where n^k is the total number of mols present in phase k; g^k and G^k are respectively the *molar* and *total* Gibbs free energy of phase k. The composition variables can be defined intensively in terms of mol fractions ($\mathbf{x} \equiv \{x_i^k\}$), or extensively, as the number of mols of component i in phase k ($\mathbf{n} \equiv \{n_i^k\}$). It is easy to move from one form to the other via the relation $n_i^k = n^k x_i^k$. g^k is naturally expressed with intensive variables while extensive variables are appropriate for G^k. The equilibrium solution must also satisfy the linear material balance constraints.

Thermodynamic Models

Turning to the available thermodynamic models available to predict fluid phase behavior, these typically lead to expressions for the molar Gibbs functions that are mathematically complex, nonlinear and nonconvex. In this section, the analysis is presented for the molar Gibbs function.

Liquid Phases

Many liquid phases are only partially miscible (referred to as phase splitting). Nonideality is often expressed through the employment of *excess functions* which attempt to correlate the deviation of the system from ideality. The excess Gibbs free energy is simply the amount by which the Gibbs free energy is above that of an ideal solution:

$$g^{\mathrm{E}}(\mathbf{x}) = g(\mathbf{x}) - g^{\mathrm{I}}(\mathbf{x})$$

with

$$g^{\mathrm{I}}(\mathbf{x}) = \sum_{i \in C} x_i \mu_i^{\circ} + \sum_{i \in C} x_i \ln x_i,$$

where μ_i° is the chemical potential of pure component i referred to the standard state. $g^{\mathrm{I}}(\mathbf{x})$ is convex. A number of different expressions of increasing complexity are now summarized for the excess Gibbs functions The only variables are the mol fractions x_i and all other quantities are parameters particular to the thermodynamic model. References to these equations and their parameters can be found in [21].

The Wilson Equation

Because the molar Gibbs free energy is convex in this case, this equation is the only model described here that cannot be used to predict phase splitting.

$$g^{\mathrm{E}}(\mathbf{x}) = -\sum_{i \in C} x_i \ln \sum_{j \in C} \Lambda_{ij} x_j.$$

Regular Solutions

This equation is bilinear:

$$g^{\mathrm{E}}(\mathbf{x}) = \sum_{i\in C}\sum_{j\in C} A_{ij}x_i x_j.$$

The NRTL Equation

This widely used model consists of a summation of bilinear fractional terms:

$$g^{\mathrm{E}}(\mathbf{x}) = \sum_{i\in C} x_i \frac{\sum_j \tau_{ji}G_{ji}x_j}{\sum_j G_{ji}x_j}.$$

The next three models are nonconvex in form. They are grouped together because it has been shown in [16] how they can be transformed into the difference of two convex functions (d.c. form), allowing the application of standard branch and bound global optimization algorithms.

The UNIQUAC Equation

The excess Gibbs function is composed of a residual part and a combinatorial part, denoted $g^{\mathrm{E}}_{\mathrm{C}}(\mathbf{x})$, defined as:

$$g^{\mathrm{E}}_{\mathrm{C}}(\mathbf{x}) = \sum_{i\in C} x_i\left[1-\frac{z}{2}q_i\right]\ln\frac{r_i x_i}{\sum_j r_j x_j} + \sum_{i\in C}\frac{z}{2}q_i x_i \ln\frac{q_i x_i}{\sum_j q_j x_j}.$$

The excess Gibbs function is then given as:

$$g^{\mathrm{E}}(\mathbf{x}) = g^{\mathrm{E}}_{\mathrm{C}}(\mathbf{x}) + \sum_{i\in C} q'_i x_i \ln\frac{\sum_j q'_j x_j}{\sum_j q'_j \tau_{ji}x_j}.$$

The next two models represent the behavior of molecules in mixtures by aggregating the properties of constituent functional groups (represented by the index set $G = \{g\} = \{l\}$).

The UNIFAC Equation

The combinatorial part is the same as for the UNIQUAC equation:

$$g^{\mathrm{E}}(\mathbf{x}) = g^{\mathrm{E}}_{\mathrm{C}}(\mathbf{x}) + \sum_{i\in C} x_i \sum_{g\in G}\nu_{gi} \times\left\{Q_g \ln\frac{\sum_j x_j q_j}{\sum_j x_j \sum_l Q_l \nu_{lj}\Psi_{lg}} - \ln\Gamma_g^{(i)}\right\}.$$

The ASOG Equation

$$g^{\mathrm{E}}(\mathbf{x}) = \sum_{i\in C} x_i \ln\frac{s_i}{\sum_j x_j s_j} + \sum_{i\in C} x_i \sum_{g\in G}\nu_{gi} \times\left\{\ln\frac{\sum_j x_j\sum_l \nu_{lj}}{\sum_j x_j\sum_l \nu_{lj}a_{gl}} - \ln\Gamma_g^{(i)}\right\}.$$

Of all the above methods, the NRTL, UNIQUAC and UNIFAC are currently the most commonly used. Notice that some of the correlations are of high mathematical complexity. While this is necessary in order to predict multiple liquid phases, it can lead to problems where extraneous and erroneous additional phases are predicted. An example is given in [19] where the NRTL equation mathematically predicts three liquid phases when the physical mixture has only two phases.

Vapor Phases

Deviation from ideality in vapor phases is often expressed through the use of fugacity coefficients:

$$g(\mathbf{x}, z) - g^I(\mathbf{x}) = \ln\phi(\mathbf{x}, z),$$

where $\phi(\mathbf{x}, z)$ is the fugacity coefficient of the mixture. The standard state is usually assumed to be an ideal gas at T and unit fugacity. The compressibility $z = pv/RT$ measures deviation from the ideal gas law, and an expression for it is required to calculate $\phi(\mathbf{x}, z)$. For an ideal gas, $z = 1$; otherwise, z is often obtained from an equation of state (EOS) which correlates the temperature, pressure, volume and the composition of nonideal mixtures. This equation of state then becomes an additional constraint (typically nonlinear and nonconvex) that must be obeyed over all compositions. One possible generalized equation of state can be written in its standard form as:

$$z - \alpha B - \frac{z-\alpha B}{z-B} + \frac{A}{z+\beta B} = 0, \tag{1}$$

$$A = \sum_{i\in C}\sum_{j\in C} A_{ij}x_i x_j, \quad B = \sum_{i\in C} B_i x_i, \tag{2}$$

where α and β are constants that depend on the equation of state employed. The more important equations of state include the van der Waals ($\alpha = \beta = 0$), Soave–Redlich–Kwong ($\alpha = 0$, $\beta = 1$), and Peng–Robinson ($\alpha = \sqrt{2}-1$, $\beta = \sqrt{2}+1$). See [26] for a thorough review. Note that (1) is composed of the sum of a linear

fractional and a bilinear fractional function, and that (2) defines A as a bilinear function. This means that when an equation of state is used, an additional level of complexity is added to the problem in the form of nonconvex and nonlinear constraints.

As is demonstrated in several standard texts [26], the overall mixture fugacity coefficient can be obtained using (1) as:

$$\ln \phi(\mathbf{x}, z) = (z-1) - \ln(z-B) + \frac{1}{(\alpha+\beta)} \frac{A}{B} \ln \frac{z-\alpha B}{z+\beta B}.$$

This function is highly nonlinear and nonconvex, consisting of a bilinear fractional function (A/B) multiplying the logarithm of a linear fractional function.

Obtaining Equilibrium Solutions

Here the global minimum of the total Gibbs function is sought subject to the material balance constraints. Because the total Gibbs function is used, extensive variables are appropriate. Following [23], assume there are π phase classes characterized by a separate thermodynamic model. π_{EOS} represents the phase class where an EOS is used. Before solving the problem, P_π, the number of phases consistent with phase class π, must be selected. $P = \cup_\pi P_\pi$. The solution will then yield $P_\pi^{eq} \le P_\pi$ where P_π^{eq} is the number of phases of class π present in nonzero amounts at equilibrium. Consider a potential LLV mixture: if the NRTL is used to model two liquid phases, and the Peng–Robinson equation for a single vapor phase, then π_1 = NRTL, $P_{\pi_1} = 2$; π_2 = PR, $P_{\pi_2} = 1$. If the actual physical mixture at equilibrium is calculated as LV, then $P_{\pi_1}^{eq} = P_{\pi_2}^{eq} = 1$. The phase rule [26] gives an upper bound on the number of possible phases. The optimization formulation can now be written as:

$$(G) \begin{cases} \min\limits_{n \in N} & G = \sum\limits_{p \in \pi} \sum\limits_{k \in P_\pi} G^k \\ \text{s.t.} & \text{EOS}^k = 0, \quad \forall k \in P_{\pi_{\text{EOS}}}, \end{cases}$$

where

$$N = \left\{ \mathbf{n}: \ \sum_k n_i^k = n_i^T, \ \forall i, \quad n_i^k \ge 0, \ \forall i, k \right\}.$$

Here, n_i^T is the total number of moles of component i in the mixture. Note that the equation of state in (G) comprising (1) and (2) is assumed to be written in extensive form.

Equation Based Approaches

Even though (G) is naturally expressed as an optimization problem, equation based approaches are by far the most prevalent due to their use in commercial chemical process simulators. The first order necessary optimality conditions of (G) reduce to a set of nonlinear equations, corresponding to the condition of equality of chemical potentials (μ_i^k):

$$\mu_i^k = \mu_i^{k'}, \quad \forall i \in C, \ \forall k, k' \in P. \tag{3}$$

All chemical engineering undergraduates encounter the direct iteration K-value method for solving (3), known as the single stage flash calculation. A general description is supplied by [12]. The inside-out algorithm of [2] is of especial prominence due to its superior performance to other methods. Because these equations are nonconvex, there may be several solutions which satisfy them, and these methods are prone to failure, especially at conditions close to the critical point (which is called the plait point for liquid phases).

Local Optimization

Given the problems associated with the equation based approaches, various attempts to solve (G) using local optimization have been attempted. A steepest descent method was used in [27] and is known as the RAND method. Various methods were compared to an implementation of Wolfe's quadratic programming algorithm in [5]. A variable projection method was used in [3]. Several other variants of Newton based methods have been employed (see [17] for a brief summary). None of these methods—typically Newton or quasi-Newton algorithms—removes the possibility of converging to a local optimum, or a trivial solution (saddle point where the mol fractions in two phases of the same class π are the same), and are highly dependent on starting point. A major problem is that P_π is unknown and must be guessed, and therefore, the incorrect number of phases P_π^{eq} is easily obtained with these methods. Another key problem in these approaches is the development of numerical singularities when phases coalesce or split as the algorithm progresses [22].

Global Optimization

The above facts motivate the employment of global optimization techniques because if an approach can be guaranteed to obtain the global optimum solution of (G), then a sufficiency condition for phase equilibrium is automatically supplied. The first use of global optimization to solve (G) was undertaken in [17], where it was shown how the GOP algorithm [4] could be used in cases where the NRTL equation was used to model liquid phase behavior. New variables were introduced so that the formulation of (G) would consist of a biconvex objective function and a bilinear constraint set, satisfying the requirements of the GOP to guarantee global optimality. For the UNIQUAC equation, a branch and bound global optimization algorithm described in [10] was implemented to determine the global minimum of (G) [15]. A key aspect of the work in [17] was the mathematical transformation of the nonconvex expressions for the Gibbs free energies into forms with special structure, namely the difference of two convex function (d.c. form). Similar transformations and this same algorithm can be also applied to the UNIFAC, ASOG and modified Wilson equations, as shown in [16]. These were the first approaches to guarantee convergence to the global solution of (G), regardless of the supplied initial point.

Verifying Equilibrium Solutions

The tangent plane criterion provides an alternative sufficiency condition for a candidate equilibrium solution to correspond to a global minimum of the Gibbs free energy [7]. A candidate solution must satisfy the necessary condition for equilibrium—that is, satisfy (3). Stability requires that the tangent hyperplane constructed using the chemical potential values of the candidate solution (denoted $\mu_i^\dagger$) at no point lies above the molar Gibbs surface for all phase classes used to model the mixture. Stated in optimization terms, if the global minimum of the tangent plane distance function, D_π, for each phase class π used to represent the behavior of the mixture, is nonnegative $\forall \pi$, then the candidate solution corresponds to a global minimum of the Gibbs free energy [1]. The phase stability problem is defined for a phase class π as:

$$(S)\begin{cases}\min\limits_{x\in X} & D^\pi = g^\pi - \sum\limits_{i\in C} x_i\mu_i^\dagger \\ \text{s.t.} & \text{EOS}(\mathbf{x}, z) = 0 \quad \text{if } \pi_{\text{EOS}} \subset \pi,\end{cases}$$

where $X = \{\mathbf{x}: \sum_i x_i = 1,\ x_i \geq 0,\ \forall i\}$. Clearly, (1) and (2) are required for (S) when $\pi_{\text{EOS}} \subset \pi . g^\pi$ is obtained from the appropriate thermodynamic models described earlier. Therefore, it is seen that the approach involves verifying that a candidate solution is the equilibrium one.

Equation Based Approaches

As with (G), the first order necessary optimality conditions of (S) reduce to a set of nonlinear equations:

$$\mu_i^\pi - \mu_i^\dagger = K, \quad \text{s.t. } \mathbf{x} \in X, \tag{4}$$

where K is a constant. The EOS must be satisfied if $\pi_{\text{EOS}} \subset \pi$. If a nonnegative solution to this set of equations is obtained, then the postulated solution is assumed to be stable. Standard direct iteration methods have been used [20] as well as homotopy continuation methods [24] to solve (4). However, no guarantee of obtaining all stationary points can be provided with the typical equation based approach. However, an interval Newton method has been used in [11] to ϵ-enclose all stationary points. This work can be considered a 'global' method for equation solving. It should be noted that a branch and bound global optimization algorithm [13] has been used to obtain all homogeneous azeotropes in mixtures [9]; because the condition of azeotropy adds a single linear constraint (equality of mol fractions in all phases) to (3), this approach can in principle be used to guarantee obtaining all ϵ-global solutions to both (3) and (4).

Global Optimization

The advantage of a global optimization approach is that if a nonnegative solution is found, then it can be definitively asserted that the candidate solution is the globally stable equilibrium one, unlike available local algorithms. It is shown in [18] how global optimization can be used to solve (S), using the GOP algorithm for the NRTL equation, and a branch and bound algorithm for the UNIQUAC equation. For the modified Wilson, ASOG and UNIFAC equations, it was shown in [16] how this same branch and bound algorithm could be used after transforming the expressions for $g(\mathbf{x})$ into d.c. form. It has been shown how the formulations for (G) and (S) involving equations of state can

be transformed into biconvex form allowing the application of a number of global optimization algorithms [14], although no implementation was undertaken. An important recent extension of global optimization to the case of equations of state is supplied in [8] where the nonlinear terms are validly underestimated within the framework of a branch and bound algorithm.

Combining Approaches

From the above development, it is apparent that:

1) To obtain a candidate equilibrium solution, either solve (G) or (3); and
2) To verify a candidate as the equilibrium solution, either solve (S) or (4).

Approach 1) is problematic because the a priori selection of P_π represents a formidable challenge. If too few phases are allowed, then convergence to constrained minima can occur; if too many are assumed, then numerical problems may arise, or convergence to trivial or local extrema may occur. Therefore, the concept of using the tangent plane criterion to provide initial guesses for (G) or (3) has been shown to greatly increase reliability with a tolerable increase in computational effort. In addition, when solving (G) or (3), the number of composition variables is $N_V = |C||P|$, while for (S) or (4), $N_V = |C|$. The performance of the RAND method was found to considerably improve when combined with a phase-splitting algorithm [6]. The seminal work of M.L. Michelsen [20] proposed an iterative approach whereby the solution from the tangent plane criterion is used to initialize the search for the equilibrium solution. This is implemented using a direct substitution method (K-value approach) as well as an optimization method. The calculations are computationally efficient and reported to be quite reliable, although there is the danger of predicting a stable phase distribution, when, in fact, this is not the case. In a comparative study for liquid-liquid phase splitting [25], Michelsen's method was found to be the most reliable. A similar iterative approach using homotopy continuation methods to solve (4) have also been used in [24]. However, there are a number of difficulties associated with these approaches. First, no guarantee of obtaining all stationary points can be provided. Second, since the solutions obtained from the stability problem are then used to initiate the search for a solution with a lower Gibbs free energy, these guesses may lead to local optima, or even infeasible equilibrium solutions. Therefore, no guarantees can be made of having obtained the equilibrium solution, even though overall reliability is significantly increased.

Global Optimization

When solving (G) using global optimization, the maximum allowable number of phases P_π, $\forall \pi$, must be considered for rigorous determination of phase and chemical equilibrium. This leads to high computational effort when often the global solution is generated early in the global optimization search [19]. For these reasons, an algorithm known as GLOPEQ (global optimization for the phase and chemical equilibrium problem) was implemented in [19]. An iterative approach was proposed based on the fact that solving (S) to global optimality to verify a candidate solution is vastly preferable to solving (G). GLOPEQ therefore leads to significant computational savings over other global optimization approaches. It should be noted that the approach described in [8] can be incorporated into GLOPEQ, extending its applicability and giving the first global optimization method for both nonideal liquid and vapor phases. The key difference between GLOPEQ and the other local iterative approaches is that global optimization is used at each step of the algorithm, allowing a guarantee to be made of obtaining the true equilibrium solution no matter the starting point.

Reaction Equilibria

If reaction occurs in the mixture, then the permissible regions N and X must be adjusted. See [23] for an elegant analysis of the tangent plane criterion for reacting mixtures. Note that N and X remain linear and they do not affect the global optimization approach for solving (G) or (S).

General Comments on Efficiency

Clearly local approaches, while less reliable, are more efficient than global optimization approaches. Because of the relatively heavy computational burden of global optimization, these approaches are more justified for off-line analysis as they could not be practically used in

a chemical process simulator. Having said that, computational times of seconds for highly nonideal mixtures of up to eight components [8] provide a great deal of promise for improving the robustness of the equilibrium calculation without resulting in excessive solution times.

See also

- ▶ αBB Algorithm
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Generalized Primal-relaxed Dual Approach
- ▶ Global Optimization: Application to Phase Equilibrium Problems
- ▶ Global Optimization in Batch Design Under Uncertainty
- ▶ Global Optimization in Generalized Geometric Programming
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Interval Global Optimization
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Global Optimization with αBB
- ▶ Optimality Criteria for Multiphase Chemical Equilibrium
- ▶ Smooth Nonlinear Nonconvex Optimization

References

1. Baker LE, Pierce AC, Luks KD (1982) Gibbs energy analysis of phase equilibria. Soc Petrol Eng J 731
2. Boston JF, Britt HI (1978) A radically different formulation and solution of the single stage flash problem. Comput Chem Eng 2:109–122
3. Castillo J, Grossmann IE (1981) Computation of phase and chemical equilibria. Comput Chem Eng 5:99
4. Floudas CA, Visweswaran V (1993) A primal-relaxed dual global optimization approach. JOTA 78(2):187
5. Gautam R, Seider WD (1979) Computation of phase and chemical equilibrium, Part I: Local and constrained minima in Gibbs free energy. AIChE J 25(6):991
6. Gautam R, Seider WD (1979) Computation of phase and chemical equilibrium, Part II: Phase-splitting. AIChE J 25(6):999
7. Gibbs JW (1873) A method of geometrical representation of the thermodynamic properties of substances by means of surfaces. Trans Connecticut Acad 2:382–404
8. Harding ST, Floudas CA (2000) Phase stability with cubic equations of state: A global optimization approach. AIChE J
9. Harding ST, Maranas CD, McDonald CM, Floudas CA (1997) Locating all homogeneous azeotropes in multicomponent mixtures. I-EC Res 36:160–178
10. Horst R, Tuy H (1992) Global optimization, 2nd edn. Springer, Berlin
11. Hua JZ, Brennecke JF, Stadtherr MA (1998) Reliable computation of phase stability using interval analysis: Cubic equation of state models. Comput Chem Eng 22(9):1207–1214
12. King CJ (1980) Separation processes, 2nd edn. McGraw-Hill, New York
13. Maranas CD, Floudas CA (1995) Finding all solutions of nonlinearly constrained systems of equations. JOGO 7:143–182
14. McDonald CM (1999) A novel reformulation of the phase equilibrium problem when using equations of state and subsequent application of global optimization. Presented at AIChE Annual Meeting
15. McDonald CM, Floudas CA (1994) Decomposition based and branch and bound global optimization approaches for the phase equilibrium problem. J Global Optim 5:205–251
16. McDonald CM, Floudas CA (1995) Global optimization and analysis for the Gibbs free energy function using the UNIFAC, Wilson and ASOG equations. I-EC Res 34:1674
17. McDonald CM, Floudas CA (1995) Global optimization for the phase and chemical equilibrium problem: Application to the NRTL equation. Comput Chem Eng 19(11):1111
18. McDonald CM, Floudas CA (1995) Global optimization for the phase stability problem. AIChE J 41(7):1798
19. McDonald CM, Floudas CA (1997) GLOPEQ: A new computational tool for the phase and chemical equilibrium problem. Comput Chem Eng 21(1):1–23
20. Michelsen ML (1982) The isothermal flash problem. Part I. Stability. Part II. Phase–split calculation. Fluid Phase Equilib 9:1–40
21. Reid RC, Prausnitz JM, Poling BE (1987) The properties of gases and liquids, 4th edn. McGraw-Hill, New York
22. Seider WD, Brengel DD, Widagdo S (1991) Nonlinear analysis in process design. AIChE J 37(1):1
23. Smith JV, Missen RW, Smith WR (1993) General optimality criteria for multiphase multireaction chemical equilibrium. AIChE J 39(4):707
24. Sun AC, Seider WD (1995) Homotopy-continuation method for stability analysis in the global minimization of the Gibbs free energy. Fluid Phase Equilib 103:213
25. Swank DJ, Mullins JC (1986) Evaluation of methods for calculating liquid-liquid phase-splitting. Fluid Phase Equilib 30:101
26. Walas SM (1985) Phase equilibria in chemical engineering. Butterworths, London
27. White WB, Johnson SM, Dantzig GB (1958) Chemical equilibrium in complex mixtures. J Chem Phys 28(5):751

Global Optimization of Planar Multilayered Dielectric Structures

CLAIRE S. ADJIMAN[1], R.F. OULTON[2]
[1] Centre for Process Systems Engineering, Dept. of Chemical Engineering, Imperial College London, London, UK
[2] NSF Center for Scalable and Integrated Nanomanufacturing, University of California at Berkeley, Berkeley, USA

MSC2000: 65K99

Article Outline

Introduction
Formulation
Statement of Physical Problem
Analytical Gradients for Effective Optimization
Methods and Applications
Multi-Level Approaches
Simulated Annealing
Genetic and Memetic Algorithms
Needle Optimization
Deterministic Methods
A Comparison of Methods for an Infrared Filter Design
Conclusions
References

Introduction

Multi-layered dielectric structures are relevant in many applications that seek to influence electromagnetic radiation across the infrared, optical and X-ray spectra. Anti-reflection coatings, components for integrated optics and semiconductor lasers are based on multilayered dielectric designs; they are generally modeled using the transfer matrix method that has been in widespread use for the past thirty years [5,26]. In many cases optical designs can be devised by deductive reasoning, but, as design objectives have become more elaborate, robust numerical optimization techniques have become increasingly relevant. Baumeister reported the first refinement technique for multilayer dielectric in 1958 [3].

The synthesis of multi-layered dielectric structure designs requires a robust global optimization approach. The mathematical model that describes the optical properties of these structures is highly non-linear and presents any solver the task of sifting through countless local minima. Early approaches relied on stochastic global methods. The lack of deterministic methods in the literature highlights the challenging mathematical task of identifying minimizing convex approximations. As far as the authors know, the only deterministic approach proposed to date is limited in scope due to approximations that are made to derive model equations such that the problem has a unique solution.

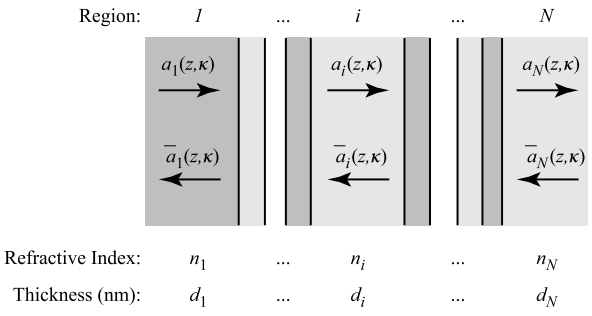
This encyclopedia entry examines the problem of multilayer dielectric design, which has been treated with a range of algorithms over the past 20 years. Stochastic approaches are reviewed including Simulated Annealing, Genetic Algorithms and a Multi-Level approach. A deterministic minimization approach is also discussed. The study may be considered a review and critical comparison of techniques for electromagnetic filter design.

Formulation

Statement of Physical Problem

Multilayered dielectric structures have two modes of operation: in passive mode, a structure reflects or transmits light from an external source as a function of the input wavelength and direction; in active mode, a structure creates light internally and distributes the emission both spectrally and spatially. Figure 1 illustrates these two modes of operation. Here, $a_i^{(\mu)}(\kappa, z_i)$ and $\bar{a}_i^{(\mu)}(\kappa, z_i)$ are the forward and backward propagating amplitudes in Region i respectively. The superscript in brackets ($\mu = \{s, p\}$) indicates the polarization, which is described as either Transverse Electric (s) or Transverse Magnetic (p).

In the passive geometry, amplitudes are equated with real measurable quantities: $|a_1(\kappa, z_1)|^2 = 1$, $|\bar{a}_1(\kappa, z_1)|^2 = R(\kappa)$, $|a_N(\kappa, z_N)|^2 = T(\kappa)$ and $|\bar{a}_N(\kappa, z_N)|^2 = 0$, where $R(\kappa)$ and $T(\kappa)$ are the reflectivity and transmissivity of the structure respectively. In active mode $|a_1(\kappa, z)|^2 = 0$, $|\bar{a}_1(\kappa, z_1)|^2 = P_b(\kappa)$, $|a_N(\kappa, z_N)|^2 = P_f(\kappa)$, $|\bar{a}_N(\kappa, z_N)|^2 = 0$ where P_f and P_b are the forward and backwards emission powers. The source amplitudes, $a_i(\kappa, z_i) = A(\kappa)$ and $\bar{a}_2(\kappa, z_i) = \bar{A}(\kappa)$ are dependent on the type of emission source and will not be elaborated upon here. For more information on these issues, the reader should refer to [10] and [4]. In both operation modes, the structure interacts with



Global Optimization of Planar Multilayered Dielectric Structures, Figure 1
Schematic of a multilayer dielectric structure highlighting the nomenclature …

light as a function of its wavelength, λ, and the optical angle parameter, $\kappa = n_i \sin\theta_i$, which is related to the propagation angle, θ_i, in region i. Note that κ is invariant throughout the structure unlike θ_i, which varies with the refractive index, n_i, due to Snell's law ($n_i \sin\theta_i = \text{constant}$).

The following analysis considers only the passive mode of operation, although the approach is readily adaptable to describe problems involving the active mode. Therefore, $R(\kappa)$ and $T(\kappa)$ are usually part of some expression to be minimized. The design variables to be optimized are the refractive indices, n_i, and layer thicknesses, d_i, throughout the structure. The optimization problem is posed as follows: a single valued objective function $\mathcal{F}\{R(\kappa;\mathbf{n},\mathbf{d}), T(\kappa;\mathbf{n},\mathbf{d})\}$ involving the reflectivity and transmissivity must be minimized subject to unknown variables $\mathbf{n} = \{n_i\}$ and $\mathbf{d} = \{d_i\}$ where $i \in \{1, \ldots, N\}$. The problem is typically bounded, defining a variable space of finite extent: for example, the unknown variables here are constrained to upper, $\mathbf{n}^U$, $\mathbf{d}^U$ and lower, $\mathbf{n}^L$, $\mathbf{d}^L$ bounds. This is summarized as,

$$\begin{aligned} &\min_{\mathbf{n},\mathbf{d}} \mathcal{F}\{R(\kappa;\mathbf{n},\mathbf{d}),\ T(\kappa;\mathbf{n},\mathbf{d})\} \\ s.t.\quad &\mathbf{n}-\mathbf{n}^U \le 0 \\ &\mathbf{n}^L-\mathbf{n} \le 0 \\ &\mathbf{d}-\mathbf{d}^U \le 0 \\ &\mathbf{d}^L-\mathbf{d} \le 0\,. \end{aligned} \tag{1}$$

The use of the transfer matrix method to describe the propagation of light through multi-layer planar dielectric materials is well-established [5,26]. Oulton and Adjiman [28] present an alternative and more compact representation highlighting the mathematical details of the model and symmetries that are useful for writing efficient code and deriving compact analytical gradients for local optimization.

Consider the schematic for a general multilayered structure in Fig. 1. The transfer matrix, $\mathbf{T}_{ji}^{(\mu)}(\kappa)$, relates the electromagnetic field amplitudes in regions i and j at z_i as follows,

$$\begin{aligned} &\begin{pmatrix} a_j^{(\mu)}(\kappa,\, z_i) \\ \bar{a}_j^{(\mu)}(\kappa,\, z_i) \end{pmatrix} \\ &= \begin{pmatrix} X_{j,i}^{(\mu)+}(\kappa) & X_{j,i}^{(\mu)-}(\kappa) \\ X_{j,i}^{(\mu)-}(\kappa) & X_{j,i}^{(\mu)+}(\kappa) \end{pmatrix} \begin{pmatrix} a_i^{(\mu)}(\kappa,\, z_i) \\ \bar{a}_i^{(\mu)}(\kappa,\, z_i) \end{pmatrix} \\ &= \mathbf{T}_{ji}^{(\mu)}(\kappa) \begin{pmatrix} a_i^{(\mu)}(\kappa,\, z_i) \\ \bar{a}_i^{(\mu)}(\kappa,\, z_i) \end{pmatrix} \end{aligned} \tag{2}$$

$$X_{j,i}^{(\mu)\pm} = \frac{1}{2}\left(C_{i,j}^{(\mu)} \pm \frac{1}{C_{i,j}^{(\mu)}}\right). \tag{3}$$

The coupling coefficients, $C_{i,j}^{(\mu)}$ are

$$\begin{aligned} C_{i,j}^{(s)} &= \sqrt{\frac{\beta_j}{\beta_i}} \\ C_{i,j}^{(p)} &= \frac{n_j}{n_i}\sqrt{\frac{\beta_i}{\beta_j}}\,. \end{aligned} \tag{4}$$

Here, $\beta_i = k_0\sqrt{n_i^2-\kappa^2}$ is related to κ and λ through $k_0 = 2\pi/\lambda$, the wavenumber of the incident light. β_i is the component of the wavenumber normal to the planar layers and is sometimes referred to as the propagation constant. Note that $\mathbf{T}_{ji}^{(\mu)}(\kappa)$ is symmetric with only two independent elements.

To describe propagation across region j, of thickness $d_j = z_j - z_i$, the amplitudes, $a_j(\kappa, z_j)$ and $\bar{a}_j(\kappa, z_j)$ at z_j are related to the amplitudes, $a_j(\kappa, z_i)$ and $\bar{a}_j(\kappa, z_i)$ at z_i by the transfer matrix, $\mathbf{P}_j(\kappa)$, which is independent of polarization for isotropic materials.

$$\begin{aligned} &\begin{pmatrix} a_j^{(\mu)}(\kappa,\, z_j) \\ \bar{a}_j^{(\mu)}(\kappa,\, z_j) \end{pmatrix} \\ &= \begin{pmatrix} e^{\mathrm{i}\beta_j d_j} & 0 \\ 0 & e^{-\mathrm{i}\beta_j d_j} \end{pmatrix} \begin{pmatrix} a_j^{(\mu)}(\kappa,\, z_i) \\ \bar{a}_j^{(\mu)}(\kappa,\, z_i) \end{pmatrix} \\ &= \mathbf{P}_j(\kappa) \begin{pmatrix} a_j^{(\mu)}(\kappa,\, z_i) \\ \bar{a}_j^{(\mu)}(\kappa,\, z_i) \end{pmatrix}. \end{aligned} \tag{5}$$

In order to relate the fields at the interfaces between regions $i+2$ and $i+1$ at z_{i+1} and regions $i+1$ and i at z_i, interface and propagation matrices are multiplied together such that,

$$\begin{pmatrix} a_{i+1}(\kappa, z_{i+1}) \\ \bar{a}_{i+1}(\kappa, z_{i+1}) \end{pmatrix} = \mathbf{M}^{(\mu)}_{i+1,i}(\kappa) \begin{pmatrix} a_i(\kappa, z_i) \\ \bar{a}_i(\kappa, z_i) \end{pmatrix} \quad (6)$$

where $\mathbf{M}^{(\mu)}_{i+1,i}(\kappa) = \mathbf{P}_{i+1}(\kappa)\mathbf{T}^{(\mu)}_{i+1,i}(\kappa)$. In the general formulation, the amplitudes can be expressed as a vector of plane wave modes corresponding to the various angles of propagation within the planar dielectric medium. When considering N values of κ (angles of incidence), the transfer matrix, $\mathbf{M}^{(\mu)}_{i+1,i}$ will be a $2N \times 2N$ matrix. Notice, however, that due to Snell's law and the law of reflection $\mathbf{M}^{(\mu)}_{i+1,i}$ is sparse with only $4N$ components along the diagonals of each quadrant of $\mathbf{M}^{(\mu)}_{i+1,i}$. Therefore there are only $2N$ independent components. From here on, the parameter κ will be dropped from the mathematical expressions for brevity.

Analytical Gradients for Effective Optimization

The efficiency and accuracy of local optimization can be enhanced by using analytically determined gradients. Methods for determining the gradients of transfer matrices have been presented in the literature [29], but here the new formalism allows a compact analytical evaluation which leads to simplified coding and efficient strategies for calculating large numbers of gradients for one structure.

Consider the derivative of the standard mode matching matrix equation with respect to the variable, ζ_i for an optical structure with N layers:

$$\frac{\partial}{\partial \zeta_i}\begin{pmatrix} a^{(\mu)}_N(z_N) \\ \bar{a}^{(\mu)}_N(z_N) \end{pmatrix} = \frac{\partial \mathbf{M}^{(\mu)}_{N,1}}{\partial \zeta_i}\begin{pmatrix} a^{(\mu)}_1(z_1) \\ \bar{a}^{(\mu)}_1(z_1) \end{pmatrix} + \mathbf{M}^{(\mu)}_{N,1}\frac{\partial}{\partial \zeta_i}\begin{pmatrix} a^{(\mu)}_1(z_1) \\ \bar{a}^{(\mu)}_1(z_1) \end{pmatrix}. \quad (7)$$

Given any two constant boundary condition amplitudes the matrix equation can be solved for the derivatives of the unknown amplitudes. Consider now the derivative of the matrix with respect to the variables of a given layer.

The derivative with respect to d_i is the easiest to evaluate as only one phase matrix, $\mathbf{P}_i$, needs to be differentiated. The matrix derivative is:

$$\frac{\partial \mathbf{M}^{(\mu)}_{N,1}}{\partial d_i} = \mathbf{M}^{(\mu)}_{N,i}\mathbf{dM}^{(\mu)}_{di}\mathbf{M}^{(\mu)}_{i,1} = \mathrm{i}k_{z\,i}\mathbf{M}^{(\mu)}_{N,i}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\mathbf{M}^{(\mu)}_{i,1}. \quad (8)$$

Differentiation with respect to the refractive index, n_i, is much more complicated as it involves the product of three matrices and must be evaluated using the Leibniz rule. In the current representation, transfer matrix symmetries can be exploited to give a concise form as in Eq. (8), which can be written for the two polarizations as follows:

$$\frac{\partial \mathbf{M}^{(\mu)}_{N,1}}{\partial n_i} = \mathbf{M}^{(\mu)}_{N,i}\mathbf{dM}^{(\mu)}_{n_i}\mathbf{M}^{(\mu)}_{i,1} \quad (9)$$

where,

$$\mathbf{dM}^{(s)}_{n_i} = \frac{n_i k_0^2}{\beta_i^2}\begin{pmatrix} \mathrm{i}\beta_i d_i & \frac{1}{2}\left\{e^{2\mathrm{i}\beta_i d_i} - 1\right\} \\ \frac{1}{2}\left\{e^{-2\mathrm{i}\beta_i d_i} - 1\right\} & -\mathrm{i}\beta_i d_i \end{pmatrix}$$

$$\mathbf{dM}^{(p)}_{n_i} = \frac{n_i k_0^2}{\beta_i^2}\begin{pmatrix} \mathrm{i}\beta_i d_i & \frac{1}{2}\frac{\beta_i^2 - k_x^2}{k_0^2 n_i^2}\left\{e^{2\mathrm{i}\beta_i d_i} - 1\right\} \\ \frac{1}{2}\frac{\beta_i^2 - k_x^2}{k_0^2 n_i^2}\left\{e^{-2\mathrm{i}\beta_i d_i} - 1\right\} & -\mathrm{i}\beta_i d_i \end{pmatrix}. \quad (10)$$

These are extremely concise forms for the matrix derivatives of fairly complicated expressions where each gradient only requires the evaluation of a supplementary transfer matrix, $\mathbf{dM}^{(\mu)}_{\zeta_i}$ and one additional matrix evaluation.

The reflectivity, R and transmissivity, T, involve the absolute square of the field amplitudes. Given the derivative of the amplitude, $a_i(z_i)$, the derivative of its absolute square is given by,

$$\frac{dR}{d\zeta_i} = \frac{d(a_i(z_i)a_i(z_i)^*)}{d\zeta_i} = 2\Re\{a_i(z_i)\}\Re\left\{\frac{da_i(z_i)}{d\zeta_i}\right\} + 2\Im\{a_i(z_i)\}\Im\left\{\frac{da_i(z_i)}{d\zeta_i}\right\}. \quad (11)$$

Methods and Applications

By the early 1990s, a range of optimization methods were being used to generate optical multi-layer filter de-

signs. Dobrowolski and co-workers reviewed ten methods for computational speed and effectiveness at reaching an optimum solution to determine which would be best suited to these highly non-linear problems [11]. Amongst these were both global and local methods but, at this time of limited computer power, no particular approach was deemed superior over the fixed calculation time of 2 hours. The authors rated a local gradient-based modified Gauss-Newton method highly for its consistency over the problems investigated. The observation that the global optimization methods performed on a par with local methods, despite the clear limitation in fixed calculation time, was also noted.

Computing power today is not as great an issue and the use of global optimization techniques for multi-layered optical design has attracted a great deal of attention (see references throughout this Section). Global algorithms can be broadly divided into two categories: deterministic and stochastic. Deterministic methods guarantee a global solution, usually at the expense of calculation time, whereas stochastic methods converge rapidly to solutions with only a probabilistic guarantee of global optimality in finite time. Liberti and Kucherenko investigated these contrasting philosophies by comparing the deterministic spatial Branch and Bound (sBB) and Stochastic Multi Level Single Linkage (MLSL) methods for a range of test functions [24]. The authors concluded that the stochastic method, in the cases studied, converged faster to a global optimum with a high degree of probability, but the deterministic method could perform better in cases where specific theoretical assumptions about a problem's analytical structure could be taken into account. In general, deterministic methods require preparation for a particular problem, whereas stochastic methods can be more readily adapted for *black-box* scenarios. Nevertheless, stochastic approaches cannot guarantee global optimality in finite time.

In this section, a range of global optimization approaches are reviewed. It is most useful that in the study of these methods, some authors have examined the same numerical synthesis problem: the design of an anti-reflection coating to operate in the far infrared [1,6,11,25,28,36,44]. The objective is to minimize a Germanium (Ge: refractive index $n_{Ge} = 4.2$) and Zinc Oxide (ZnS: refractive index $n_{\mathrm{ZnS}} = 2.2$) multi-layered structure to achieve a normal incidence reflectivity, $R(\kappa = 0) \mapsto 0$ for $N_\lambda = 47$ equidistant wavelengths in the spectral band $7.7 \le \lambda \le 12.3\,\mu\mathrm{m}$. The incident medium is air and the substrate which the structure is built on has refractive index $n_{\mathrm{Sub}} = 4$.

The objective function, $\mathcal{F}(\mathbf{d},\ \lambda_i,\ R_i)$, was chosen to be the same as that used by authors in the literature to allow comparative studies.

$$\mathcal{F}(\mathbf{d},\ \lambda_i,\ R_i) = \left[\frac{1}{N_\lambda}\sum_{i=1}^{N_\lambda} R_i(\lambda_i)^2\right]^{-1/2}. \qquad (12)$$

In the following studies, the optimum layer thicknesses for reproducing the best designs are omitted for brevity, so the reader should consider the relevant references for this information.

Multi-Level Approaches

In Multi-Level (ML) approaches (e. g., [20,21,23,24,28]), different starting points are generated by a higher-level algorithm, and the problem is solved from each starting point by a lower-level local optimization algorithm. This approach is very general because it requires no tuning. It has been applied by Oulton and Adjiman [28] to the design of multi-layered dielectric device design by using a deterministic sampling of the parameter space and local nonlinear programming (NLP) solver. The approach is able to rank many local solutions for post-optimization analysis. It is also non-adaptive at the global level in that the algorithm depends only on the current state and not on previously calculated states. This brings two advantages: firstly, non-adaptive methods are deemed superior to adaptive ones in multi-processor applications, which is certainly a benefit for computationally intensive global optimization problems. Secondly, non-adaptive algorithms have freedom over the specification of convergence criteria. Since the optimization algorithm in [28] essentially operates by batch local optimization, it can be halted according to criteria such as the number of global iterations or after a set time limit. Rigorous criteria are also applicable to the ML strategy [20,21,23]: as the algorithm progresses the probability that the current best local solution is the global one increases, primarily due to the global search coverage guaranteed through the appropriate choice of the sampling ap-

proach; for instance, Oulton and Adjiman used a Sobol' sequence [33], a deterministic Low Discrepancy Sequence (LDS) which provides a *good* coverage of the variable space. The Sobol' LDS was selected because its construction is based on i) homogeneity as the number of sample points, $n \mapsto \infty$, ii) good distribution for small n and iii) fast computational algorithm. All these features, but particularly ii), make this LDS most applicable to the current problem. There are a variety of LDSs that are constructed on differing requirements such as Holton, Faure, Niederreiter and Sobol' amongst others [7,18,27,33].

Simulated Annealing

Simulated annealing (SA) has been applied to a variety of electromagnetic multilayer design problems in the infra-red, ultra-violet and X-ray spectra [6,8,9,15, 22,41,42]. SA [22] operates by randomly changing an initial design in small steps and accepting the changes based on an evaluation of the new design performance according to criteria that become increasingly stringent as the algorithm progresses. Changes are always accepted if they result in a better design. On the other hand, a worse design is accepted with a probability based on a Boltzmann distribution. The probability of acceptance is tuned by changing the Boltzmann temperature according to a user-specified schedule. Wider exploration of the variable space at the start of the optimization is achieved by setting a high temperature, which essentially allows the algorithm to accept worse designs and thereby move between local regions of attraction. A cooling schedule restricts the algorithm's ability to investigate adjacent local regions and forces convergence to a local optimum. In this case, it is clear that convergence to the global optimum will be dependent on the initial design and especially on the cooling schedule.

The first reports on SA applied to multilayer design highlighted mainly the technique's ability to avoid local minima [41], although adaptations to avoid deep local minima were also reported [8]. These reports were for structure in the visible to near infrared spectra. The method has recently seen use in the design of reflectors for UV [15] and X-ray [9] spectra. These have applications that include neutron optics, X-ray astrophysics and synchrotron radiation. In this region of the spectrum, matter interacts with electromagnetic radiation differently requiring a modified transfer matrix description that accounts for surface roughness and interdiffusion (See [9] and references therein). Wu et al. have applied simulated annealing to a different optics problem involving diffraction gratings [42]. This important design problem concerns the efficient coupling of light into and out of waveguides and optical interconnects.

Boudet et al. [6] use SA to synthesize multilayer designs for the problem given in the introduction to this section. Results for $N_L = 17$ (triangle) and $N_L = 20$ (filled triangle) are shown in Fig. 2b along with results generated using other approaches. The merits of the method are discussed in Sect. "A Comparison of Methods for an Infrared Filter Design".

Genetic and Memetic Algorithms

Evolutionary or Genetic Algorithms (GA) are the preferred method in the optics community [2,14,16,17,19, 25,39,43,44,45,46,47]. GAs operate on the principle that the evolution of a random population of parameterizations, subject to iterative rules of reproduction and mutation, will converge to a region of attraction containing the global optimum [17]. Members of the population with high performance are given a greater likelihood of reproducing thereby generating a better population than the one before. Mutation prevents the algorithm converging too quickly and provides the mechanism by which the variable space can be explored more fully. Usually, local optimization is required to refine the final solution. In the case of the Memetic algorithm, local optimization is performed on each new member of the population. This approach could therefore be considered as a multi level approach (see earlier Section).

Eisenhammer et al. [14] optimized the performance of heat mirrors for solar cells: these are high pass filters that transmit optical solar radiation but reflect thermal radiation, which would otherwise be lost by the solar cell. Their designs differ slightly from typical dielectric multilayers since they incorporate metals, which help to reflect thermal frequencies. Bagnoud and Salin [2] and Yakovelev and Tempea [43] have applied GAs to the design of chirpped mirrors, which are used to make ultra-fast lasers with fempto-second pulses. These sophisticated mirrors are designed to have a reflectance

over a broad range of frequencies and, in addition, must also be compensated to ensure stability of the reflection phases. Yakovelev and Tempea [43] used a memetic algorithm and report fast convergence compared with the standard GA. Hoorfar et al. [19] developed the GA approach by considering the choice of dielectric materials from a list of candidates for the multilayer design. The authors thereby treat the mixed parameter approach (i. e. discrete and continuous optimization parameters) to which the GA approach appears amenable. Other authors have developed the technique still further by considering multiple objective and constraint functions [39]. The standard infrared filter design problem introduced by Aguilera et al. [1] has been treated by Martin et al. [25,44,46]: the results of these studies are shown in Fig. 2b along with those of other global optimization approaches.

Needle Optimization

Most optimization strategies for multilayer design problems consider the variation of layer thickness d_i and layer refractive index n_i. Variations of the standard techniques including multiple objective functions and mixed parameter optimization have also been discussed in this article. However, few methods consider the variation in the number of layers of a multilayer design problem. The needle optimization approach tackles the problem exactly from this perspective [32,34,35,37,38,40]. Firstly, the optimum position for introducing a needle like layer perturbation to a structure is determined: this usually corresponds to an insertion point that gives optimum convergence of the objective function. Tikhonov Jr et al. [35] provide an algorithm for locating this optimum position before needle insertion. For some objective functions, this is analytically determined, but, for flexibility, numerical approaches are available also [34,40]. Following insertion of a needle, the new design is used as the starting point for a local optimization. The needle insertion and local optimization procedure is repeated until no more refinement is possible within the constraints of the problem at hand.

An alternative approach to this problem, which has not yet been explored, would be to formulate the problem as Mixed-Integer Program, in which the existence or otherwise of each putative layer would be represented by a binary variable. This problem could be solved locally using standard techniques, and many of the global methods discussed here could be applied.

Deterministic Methods

Deterministic algorithms generally require the nonlinear set of model equations to be analyzed to obtain a convex problem which underestimates the minimum of the original design problem. Using one of various search approaches, such as Branch and Bound, it is possible to converge to a feasible global minimum by successively solving such problems, which produce tighter and tighter bounds on the solution along an infeasible design path. These methods are reviewed extensively elsewhere in the Encyclopedia.

Due to the complexity of the highly coupled transfer matrix equations, it is difficult to find appropriate convex estimators. However, Tikhonravov and Dobrowolsky [36] treat the above problem using an approximate infeasible path approach, reducing the problem to a quadratic programming problem with linear inequality constraints with one global optimum solution. Feasible solutions are obtained by local optimization of the resulting design. In their method, the reflection calculation is approximated for $\kappa = 0$ by i) assuming continuous variation of the refractive index profile, and ii) assuming only a small reflectivity, $R(0)$. In the scope of general multilayer optimization problems, these conditions are fairly restrictive, but they are applicable to the filter design problem posed by Aguilera et al. [1]. Strictly, this is not a deterministic global optimization method because it is based on solving an approximate problem to global optimality, and there is no guarantee that this corresponds to the global solution of the original problem. Tikhonravov and Dobrowolsky [36] perform the local optimization of a discretized structure to find a feasible solution. One interesting aspect of their approach is the proof of an optimal relationship between the minimum objective function and the optical thickness of a filter for a given set of material parameters. Although solutions along this line may not exist, the condition marks the limit of global optimality. The limiting condition of optimality is plotted in Fig. 2 and marks a theoretical boundary above which all solutions must lie. This is useful as a benchmark for the development of deterministic global optimization algo-

Global Optimization of Planar Multilayered Dielectric Structures, Table 1
Details of solvers and their implementations in this study. (a) [25]; (b) [6]; (c) [44]; (d) [28]. * Value estimated based on 1600 generations and 100 members per population. ** Value estimated based on computation time and CPU type, taking into account details from [25]. † Number of function evaluations depends on number of layers design and fixed computation time of 5 hrs

	Function Evals.	CPU Time	Language	CPU
(a)	160,000*	6–10 Hrs	C++	HP Apollo Series
(b)	∼100,000**	5 Hrs	C++	HP Apollo 715/75
(c)	150,000	*Unknown*	C++	*Unknown*
(d)	150,000–250,000†	5 Hrs	MatLab/C++	Intel PIV 2 GHz

Global Optimization of Planar Multilayered Dielectric Structures, Table 2
Comparison of optimum solutions found using ML [28] and GA [44]. The ML algorithm performed between approximately 150, 000 and 250, 000 function evaluations, depending on the number of layers, while the GA algorithm used between 150, 000 and 650, 000 function evaluations (specific number is unknown)

	Number of Layers	15	17	23	26	27	36
GA	Merit Function (%)	0.855	0.697	0.577	0.523	0.553	0.494
	Optical Thickness (μm)	20.34	27.04	40.17	50.99	44.98	71.15
ML	Merit Function (%)	0.675	0.638	0.556	0.531	0.535	0.507
	Optical Thickness (μm)	31.26	38.19	45.19	56.28	52.10	64.47

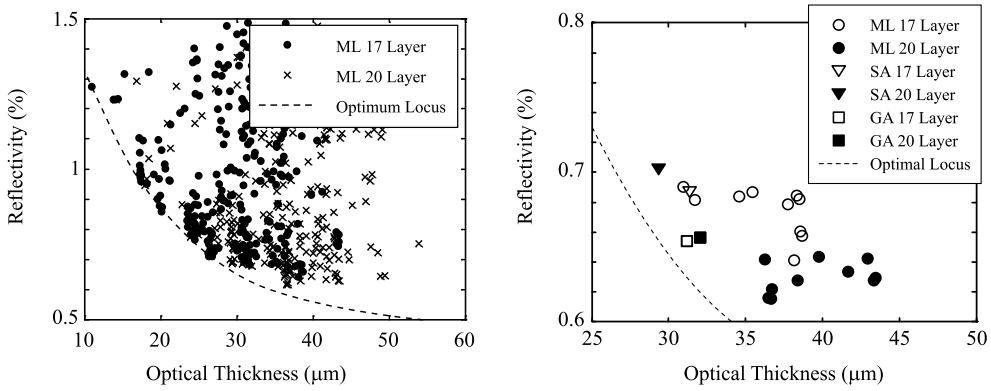
rithms and for assessing the performance of stochastic methods.

A Comparison of Methods for an Infrared Filter Design

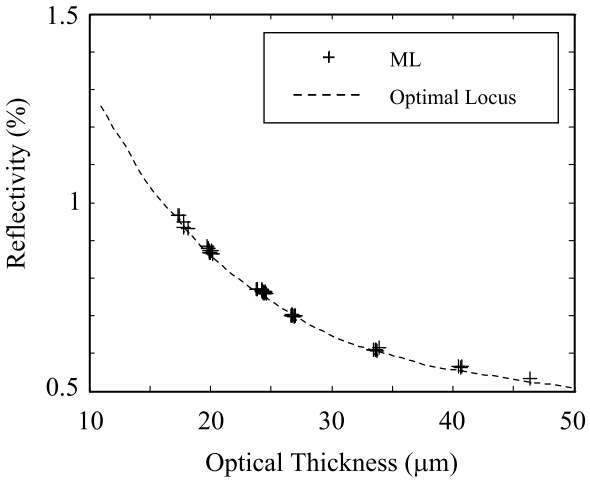
It is very difficult to compare the general performance of optimization approaches. In the following study, methods are compared through their performance in solving the infrared filter design problem that was described in the introduction to this Section. In each case, the same problem with precisely the same objective function is considered. In addition, past authors have terminated their solvers after a set number of iterations to allow a fair comparison with other methods. However, this can be a confusing measure of convergence as, in the case of GAs, a global optimum may not have been reached and in the case of SA, the cooling schedule may limit the effectiveness of the method. Consequently, the reader will note that there is no consensus over the global optimum between any of the optimization methods. Nevertheless, it is important to place each solver on an equal footing, and the number of function evaluations will be used as a measure of this. Table 1 shows information specific to each solver used in the study. It should be noted here, that only past studies of this problem using the methods discussed are compared in this study. Other studies that treat this problem can be found in [1,11,12,30,31] amongst others.

Yang and Kao [44] have provided an extensive study on this problem analyzing designs with varying layer number. The same approach was taken to generate data for the ML approach following the strategy in [28]. A direct comparison of optima found by GA [44] and ML methods is shown in Table 2 as a function of the number of design layers. Here, GA was allowed 150,000 function evaluations before stopping, whereas ML was allowed 5 hours, which, depending on the number layers, allows between 150,000 and 250,000 function evaluations. Both methods operate equally well, but, ML tends to locate slightly better solutions at the expense of optical thickness (this is equivalent the sum of the layer thicknesses multiplied by the respective refractive indices). This is to be expected due to the slightly larger number of function evaluaitons allowed for structures with lower numbers of layers.

The trade-off between the optical thickness of a filter and its reflectivity has been examined by Dombrowolsky et al. [13]. Based on a quasi-deterministic quadratic approach [36] to the anti-reflection coating design, they



Global Optimization of Planar Multilayered Dielectric Structures, Figure 2
(a) Positions of all local solutions found using ML approach for 17 and 20 layers in the 5 hour calculation time. (b) Comparison of optimum solutions of GA [25], SA [6], ML [28] and optimum locus for this problem [13]. Note that for the GA and SA algorithms, the maximum optical thickness of the filter is 32 μm, whereas, the ML algorithm has no upper limit. The top 10 solutions for the ML approach are shown



Global Optimization of Planar Multilayered Dielectric Structures, Figure 3
New results of the ML approach generated by varying the number of layers from 15 to 30. Through post optimization analysis, the top 50 results nearest the optical locus [13] were identified

specify a locus merit function against optical thickness. This represents a theoretical limit on optimality for a given anti-reflection bandwidth. This can be tested in this instance: Fig. 2a shows the locus of solutions for merit function, $\mathcal{F}$, against optical thickness using the ML approach [28]. Here, dots represent the solutions for the 17 layer structure and crosses, solutions of the 20 layer structure for the ML approach. It is clear that all solutions appear on or above the optimal locus represented by the broken line. Note however, that the optimal locus does not guarantee that solutions should be found on or near it. Figure 2b shows these results alongside optical designs using GA [25] and SA [6]. Here, it is important to note that the GA and SA approaches limit the total optical thickness to 32 μm, whereas the ML algorithm is free to locate solutions over a larger range. Despite this, all methods appear comparable, with perhaps the GA appearing superior over SA. The effectiveness of the ML approach in identifying solutions near the optimal locus can actually be assessed after optimization.

An advantage of the ML approach is the ability to perform post optimization analysis on local minima making the optimization problem highly adaptive. This is appealing because supplementary design criteria can be taken into account without having to alter the objective function directly; the optimization is usually extremely sensitive to the form of the objective function. This can be quite effective since ML generates between 100 and 200 local solutions in the 5 hour calculation time, depending on the number of layers in a design. For example, further analysis of the local optima in the current example allows solutions near the optimal locus to be identified. New results were generated using the

same approach as in [28] for designs ranging in layer number from 15–30. The top 50 solutions nearest the optimal locus were then filtered out from the complete set and are plotted in Fig. 3. Clearly, solutions very close to the optimal locus highlight the effectiveness of ML. However, the reduced number of function evaluations for structures with larger numbers of periods limits the effectiveness of the search. It is also interesting to note, that this analysis identifies gaps along the optimal locus. This suggests that, in some cases, extra layers are redundant when seeking to optimize both layer thickness and merit function.

Conclusions

The design of multi-layered dielectric optical structures can be formulated as a highly nonlinear optimization problem in which the thickness and refractive index of each layer is to be optimized, based on an appropriate objective function. This problem is known to have a large number of local minima, and several global optimization algorithm have been proposed to tackle it. These algorithms are mostly stochastic search algorithms (Simulated Annealing, Genetic Algorithms and Memetic Algorithms) or deterministic algorithms with a probabilistic guarantee of convergence (Multi-Level Algorithm). A deterministic approach with guaranteed global optimality has also been proposed based on an approximation of the design problem. The performance of several of these algorithms has been compared for a specific problem.

Future work on this design problem must continue to address the challenges posed by the large number of local optima which exist. The design formulation can also be extended to include the number of layers as one of the design variables. An early and encouraging effort in this direction is the needle optimization algorithm.

References

1. Aguilera JA, Aguilera J, Baumeister P, Bloom A, Coursen D, Dobrowolski JA, Goldstein FT, Gustafson DE, Kemp RA (1988) Antireflection coatings for germanium IR optics: a comparison of numerical design methods. Appl Opt 27:2832–2840
2. Bagnoud V, Salin F (1998) Global optimization of pulse compression in chirped pulse amplification. IEEE J Sel Top Quant Electron 4:445–448
3. Baumeister P (1958) Design of multilayer filters by successive approximations. J Opt Soc Am 48:955–958
4. Benisty H, Stanley R, Mayer M (1998) Method of source terms for dipole emission modification in modes of arbitrary planar structures. J Opt Soc Am A 15:1192–1201
5. Born M, Wolf E (1999) Principles of Optics, 6th edn. Cambridge University Press, Cambridge, UK
6. Boudet T, Chaton P, Herault L, Gonon G, Jouanet L, Keller P (1996) Thin-film designs by simulated annealing. Appl Opt 35:6219–6222
7. Bratley P, Fox BL, and Neiderreiter H (1992) Implementation and tests of low discrepancy sequences. ACM Trans Mode Comput Simul 2:195–213
8. Chang CP, Lee YH (1990) Optimization of a thin-film multilayer design by use of the generalized simulated-annealing method. Opt Lett 15:595–597
9. Cheng X, Wang Z, Zhang Z, Wang F, Chen L (2006) Design of X-ray super-mirrors using simulated annealing algorithm. Opt Commun 265:197–206
10. Crawford OH (1988) Radiation from dipoles embedded in layered system. J Chem Phys 89:6017–6027
11. Dobrowolski JA, Kemp RA (1990) Refinement of optical multilayer systems with different optimization procedures. Appl Opt 29:2876–2892
12. Dobrowolski JA, Kemp RA (1992) Interface design method for two-material optical multilayer coatings. Appl Opt 31:6747–6756
13. Dobrowolski JA, Tikhonaravov AV, Trubetskov MK, Sullivan BT, Verly PG (1996) Optimal single-band normal-incidence antireflection coatings. Appl Opt 35:644–658
14. Eisenhammer T, Lazarov M, Leutbecher M, Schiffel U, Sizmann R (1993) Optimization of interference filters with genetic algorithms applied to silver-based heat mirrors. Appl Opt 32:6310–6315
15. Graf T, Michette A, Powell A (2003) Design of multilayer mirrors for XUV applications using simulated annealing. J Phys IV France 104:243
16. Hagemana J, Wehrens R, van Sprang H, Buydens L (2003) Hybrid genetic algorithm-tabu search approach for optimizing multilayer optical coatings. Anal Chim Acta 490:211–222
17. Holland JH (1975) Adaptation in Natural and Artificial Systtems. University of Michigan Press, Ann Harbor
18. Holton JH (1960) On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numer Math 2:84–90
19. Hoorfar A, Zhu J, Nelatury S (2003) Electromagnetic optimization using a mixed-parameter self adaptve evolutionary algorithm. Microw Opt Technol Lett 39:267–271
20. Kan AHGR, Timmer GT (1987) Stochastic global optimization methods, part I, Clustering methods. Math Prog 39:27–56
21. Kan AHGR, Timmer GT (1987) Stochastic global optimization methods, part II, Multilevel methods. Math Prog 39:57–78

22. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
23. Kucherenko S, Sytsko Y (2005) Application of deterministic low-discrepancy sequences in global optimization. Comput Optim Appl 30:297–318
24. Liberti L, Kucherenko S (2005) Comparison of deterministic and stochastic approaches to global optimization. Int Trans Oper Res 12:263–281
25. Martin S, Rivory J, Schoenauer M (1995) Synthesis of optical multilayer systems using genetic algorithms. Appl Opt 34:2247–2254
26. McLeod HA (2001) Thin Film Optical Filters, 3rd edn. Adam Hilger L&D, Bristol
27. Niederreiter H (1987) Point sets and sequences with small discrepancy. Monatsh Math 104:273–337
28. Oulton RF, Adjiman CS (2006) Global optimization and modelling techniques for planar multilayered dielectric structures. Appl Optim 45:5910–5922
29. Peng K-O, de la Fonteijne M (1985) Derivatives of transmittance and reflectance for an absorbing multilayer stack. Appl Optim 24:501–503
30. Premoli A, Rastello ML (1992) Minimax refining of optical multilayer systems. Appl Opt 31:1597–1605
31. Rastello ML, Premoli A (1992) Continuation method for synthesizing antireflection coatings. Appl Opt 31:6741–6746
32. Schulz U, Schallenberg UB, Kaiser N (2002) Antireflection coating design for plastic optics. Appl Opt 41:3107–3110
33. Sobol IM (1967) On the distribution of points in a cube and the approximate evaluation of integrals. Comput Math Phys 7:86–112
34. Sullivan BT, Dobrowolski JA (1996) Implementation of a numerical needle method for thin-film design. Appl Opt 35:5484–5492
35. Tikhonov Jr AN, Tikhonravov AV, Trubetskov MK (1993) Second order optimization methods in the synthesis of multilayer coatings. J Comput Math Math Phys 33:1339–1352
36. Tikhonravov AV, Dobrowolski JA (1993) Quasi-optimal synthesis for antireflection coatings: a new method. Appl Opt 32:4265–4275
37. Tikhonravov AV, Trubetskov MK, DeBell GW (1996) Application of the needle optimization technique to the design of optical coatings. Appl Opt 35:5493–5508
38. Tikhonravov AV, Trubetskov MK, DeBell GW (2007) Optical coating design approaches based on the needle optimization technique. Appl Opt 46:704–710
39. Venkatarayalu NV, Ray T, Gan Y-B (2005) Multilayer dielectric filter design using a multi-objective evolutionary algorithm. IEEE Trans Antennas Propag 53:3625–3632
40. Verly PG (2001) Modified needle method with simultaneous thickness and refractive-index refinement for the synthesis of inhomogeneous and multilayer optical thin films. Appl Opt 40:5718–5725
41. Wild WJ, Buhay H (1986) Thin-film multilayer design optimization using a monte carlo approach. Opt Lett 11:745–747
42. Wu S-D, Gaylord TK, Maikisch JS, Glytsis EN (2006) Optimization of anisotropically etched silicon surface-relief gratings for substrate-mode optical interconnects. Appl Opt 45:15–21
43. Yakovelev V, Tempea G (2002) Optimization of chirped mirrors. Appl Opt 41:6514–6520
44. Yang J-M, Kao C-Y (1998) An Evolutionary Algorithm for Synthesizing Optical Thin-Film Dessigns. Lecture Notes in Computer Science. Springer, Heidelberg, p 1498
45. Yang J-M, Kao C-Y (2001) An evolutionary algorithm for the synthesis of multilayer coatings at oblique light incidence. IEEE J Lightwave Technol 19:559–570
46. Yang J-M, Horng J-T, Lin C-J, Kao C-Y (2001) Optical coating designs using the family competition evolutionary algorithm. Evol Comput 9:421–443
47. Zhou G, Chen Y, Wang Z, Song H (1999) Genetic local search algorithm for optimization design of diffractive optical elements. Appl Opt 38:4281–4290

Global Optimization in Protein Folding

Daniel R. Ripoll[1], Adam Liwo[2], Harold A. Scheraga[2]
[1] Computational Biology Service Unit, Cornell University, Ithaca, USA
[2] Baker Laboratory of Chemistry, Cornell University, Ithaca, USA

MSC2000: 60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10

Article Outline

Keywords and Phrases
Introduction
The Build-up Procedure
  Outline of the Procedure
  Drawbacks of the Procedure
  Applications
The Self Consistent Electrostatic Field Method
  Computation of the Electric Field and Dipole Moments
  Degree of Alignment of a Dipole Moment with the Electric Field

Best-possible Alignment of a Dipole Moment with the Electric Field
Applications
The Monte Carlo-Minimization Method
Applications
The Electrostatically Driven Monte Carlo Method
The Electrostatically Driven Monte Carlo Method
Backtrack
Applications
The Diffusion Equation Method and Other Methods Based on the Deformation of the Potential-Energy Surface
The Conformational Space Annealing Method
Hierarchical Approach
References

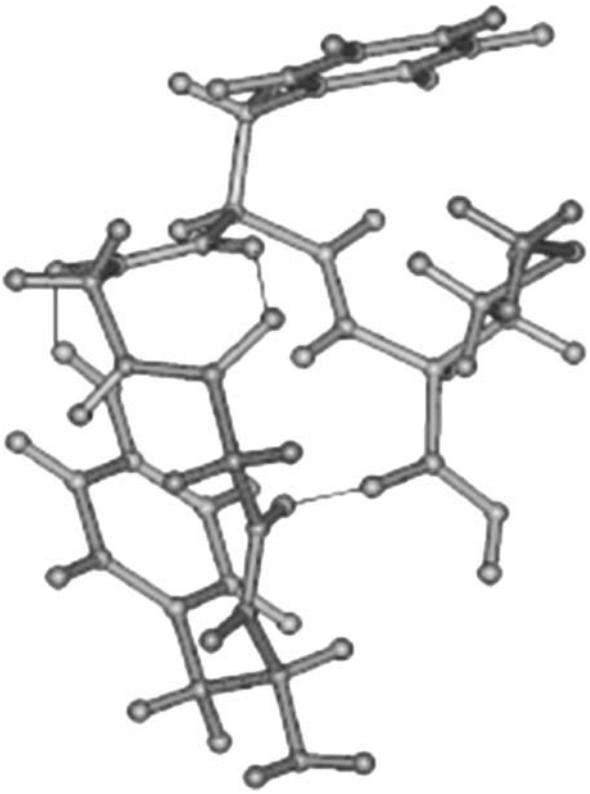
Keywords and Phrases

Global optimization; Protein folding; Multiple-minima problem; Markov process; Molecular dynamics

Introduction

The *multiple-minima problem*, i. e., the large number of minima associated with the potential functions used to represent the conformational energy of a polypeptide chain, is one of the greatest obstacles to overcome in order to compute the three-dimensional structure of a protein. Despite much effort and a large number of interesting ideas and approaches, progress toward the solution of this problem has been very slow. An exhaustive search of the conformational hypersurface of a large polypeptide is not computationally feasible even with today's supercomputers. Originally, the challenge was to locate the *global energy minimum* of small oligopeptides such as the pentapeptide Metenkephalin [1,17,20,26,45,46,48,55,57,58,59,72,73,79,91].

Since the global minimum of a potential function for a specific sequence is not known a priori, the only possibility of locating the global minimum of the potential energy is to carry out a large number of independent tests and determine if there is convergence to a unique conformation. This approach has been used in the test studies of Met-enkephalin in which hundreds of independent runs using different techniques have led to a *unique* lowest energy conformation, shown in Fig. 1, for the Empirical Conformational Energy Program for Peptides (ECEPP/2 [44,50,89], and ECEPP/3 [49]) potential energy functions. Similar results have been achieved for other test cases corresponding to larger sequences [1,23,39,56,62,77,78,80,81,94]. More recently, we have focused our efforts on the development of searching techniques that combine molecular dynamics with a coarse-grained representation of the protein structure. This approach to the protein folding problem is more rigorous since it accounts for entropic contributions and, on the other hand, is computationally more advantageous due to the simplified treatment of the complexities of the amino acid geometry. Our laboratory has made considerable progress in this area of research during the past few years, and we present a description of some of the successful methods that we have developed.



Global Optimization in Protein Folding, Figure 1
Lowest-energy conformation of Metenkephalin using the ECEPP/2 force field [27]

The Build-up Procedure

While systematic and exhaustive enumeration of all possible conformations is not practically feasible for

polypeptides and proteins, attempts have been made to develop algorithms that lead to a truncated *systematic search* of the conformational space of these molecules. One of these methods, developed in our laboratory, the build-up procedure, [9,71,85,86,88,91] assumes that short-range interactions play a dominant role in determining the conformation of a polypeptide chain. Thus, the method starts by locating the low-energy conformations of small fragments of the chain by an exhaustive energy minimization procedure. Then, a selection of the minima is carried out, keeping those that lie within an appropriate chosen upper bound (the cutoff energy) of the lowest-energy fragment. Subsequently, the limited set of minima for one fragment is combined with the set of another fragment to form larger peptides which are also subjected to energy-minimization. This process is repeated until the whole chain is eventually built up from its constituent parts. At successive stages of the algorithm, more and more long-range interactions come into play.

Outline of the Procedure

1. The smallest fragment that the build-up procedure uses to construct a polypeptide conformation is the single amino acid. The ECEPP/2 minimum-energy conformations of terminally blocked single residues were reported by M. Vásquez, G. Némethy and H.A. Scheraga [90]. The conformations were ordered by increasing energy using a cutoff energy of 5 kcal/mol and were classified according to the code defined by S.S. Zimmerman, M.S. Pottle, G. Némethy and H.A. Scheraga [98]. The ECEPP/3 force field produces the same energy minima for all blocked amino acids with the exception of the proline and hydroxyproline residues.
2. All possible dipeptides for a given molecule are generated from single-residue data (for a peptide with *n* residues there are n−1 dipeptides). After energy-minimization, the dipeptides are sorted and are used to construct tripeptides.
3. Subsequent steps to form larger fragments of the polypeptide chain involve joining two fragments with one or more residues in common, e. g. after generating conformations for the tripeptides, these can be used to construct tetrapeptides from two tripeptides having two residues in common. This process is continued until the whole polypeptide chain is built.

Drawbacks of the Procedure

One of the major difficulties of the build-up procedure is that the number of conformations of fragments that must be energy-minimized and stored at each step increases exponentially. A partial solution, aside from using an energy cut-off, is to retain only those minima whose backbone conformations differ significantly: e. g. when several local minima have almost identical backbone but different side-chain conformations, only the lowest-energy minimum is kept while the degenerate ones are discarded. This approach drastically reduces the number of conformations to be stored at each stage of the procedure; however, it may lead to problems at later stages because the side-chain rotamers that are most favorable energetically in smaller fragments are not necessarily favored in the whole polypeptide chain. Another difficulty associated with the procedure is that atomic overlaps can occur when two fragments are joined in an arbitrary manner. These overlaps lead to conformations with extremely high energy for which minimization is usually not computationally feasible. A set of algorithms designed to surmount these problems was presented by K.D. Gibson and H.A. Scheraga [9].

Applications

The build-up procedure has been used extensively in a number of studies of different molecules, among them Metenkephalin [91], Gramicidin S [6,51], Melittin [71], bovine pancreatic trypsin inhibitor [92,93] and collagen [41,42,43]. The method appears to work well for small oligopeptides and fibrous proteins but, except in a few cases, its application to larger molecules becomes unmanageable for polypeptide chains containing 10 or more amino acid residues.

The Self Consistent Electrostatic Field Method

Among all the interactions that lead to protein folding, *electrostatic interactions* are the only ones characterized as long-range. Therefore, they undoubtedly must play an important role in folding. The dominant effects

of electrostatic interactions in proteins are well recognized [60]. Among these effects, it is worth mentioning:

- The orientation of the CO and NH dipoles in α-helices are very favorable electrostatically [95] leading to a large dipole moment associated with this type of secondary structure.
- The electric field produced by an α-helix constitutes a very important stabilizing factor of the native conformations of proteins containing this type of secondary structure [11].
- The relative orientations of α-helices and β-sheets in proteins are favorable electrostatically [4,12,25].

Based on this evidence, L. Piela and H.A. Scheraga [62], hypothesized that that the native conformation of a protein arises when the electrostatic interactions are near optimal, or equivalently, that the native conformation must have approximately optimal orientations of its group dipoles in the electric field generated by the whole molecule and its surrounding solvent. Based on this assumption (which was later confirmed through rigorous calculations on an extensive set of proteins [82]), a conformational search method, named the Self-Consistent Electric Field (SCEF) method, was developed. The SCEF procedure was implemented as follows:

1. Given an arbitrary starting conformation of the molecule, minimize the total (e. g. ECEPP/3) conformational energy to reach the nearest local minimum.
2. For this minimized conformation, the *electric field* due to the whole molecule is calculated at each CO and NH group of the peptide units, and also in the middle of the C′-N peptide bond.
3. The direction of the electric field with respect to the CO and NH bond dipole moments provides information as to which peptide units are badly oriented. This electrostatic analysis of the alignment between the permanent dipoles and the electric field, is used to generate a *diagnostic rotation*. The diagnostic rotation is the variation that must be applied to a given torsional angle to obtain the best alignment of the worst oriented peptide-unit dipoles with respect to the electric field, e. g., if the electrostatic analysis indicates that the *dipole moment* of the peptide bond between residues *i* and *i+1* is the worst oriented, the diagnostic rotation will describe a change of the corresponding backbone dihedral angles ψ_i and ϕ_{i+1} required to align the dipole moment of the unit.
4. Carry out the diagnostic rotation.
5. Use the new conformation of the molecule as the starting point in step 1:
   - if a *new* local minimum is reached, then *repeat* the procedure from step 2 for the new local minimum;
   - if the *same* local minimum is found, then *step 3 must be repeated*, but using the diagnostic rotation for the next worst-oriented dipole.
6. Steps 1–5 are repeated until self-consistency is achieved, i. e., until further application of the procedure does not change the conformation of the molecule.

Computation of the Electric Field and Dipole Moments

If $\mathbf{r}$ represents the position vector assigned to the dipole moment i of a group of atoms, then the electric field, $\mathcal{E}(\boldsymbol{r})$, is computed as:

$$\mathcal{E}(\boldsymbol{r}) = (1/\epsilon)\sum_{k}{}' q_k(\boldsymbol{r}-\boldsymbol{r}_k)\big/|\boldsymbol{r}-\boldsymbol{r}_k|^3 \qquad (1)$$

where ϵ is the dielectric constant, q_k indicates the charge on atom k with position vector $\boldsymbol{r}_k$ and the prime in the summation sign indicates that the atoms which contribute to the ith dipole moment as well as those other atoms covalently bonded to them should be excluded from the computation.

The electric field is computed at three points, $\boldsymbol{r}_{i,\mathrm{CO}}$, $\boldsymbol{r}_{i,\mathrm{NH}}$, and $\boldsymbol{r}_i$. These are reference points with respect to which the dipole moments of the CO bond, $\boldsymbol{\mu}_i^{\mathrm{CO}}$, the NH bond, $\boldsymbol{\mu}_i^{\mathrm{NH}}$, and the whole ith peptide unit, $\boldsymbol{\mu}_i$, respectively, are calculated. These dipole moments are computed according to the following relations:

$$\boldsymbol{\mu}_i^{\mathrm{CO}} = q_{\mathrm{C}}(\boldsymbol{r}_{i,\mathrm{C}}-\boldsymbol{r}_{i,\mathrm{CO}}) + q_{\mathrm{O}}(\boldsymbol{r}_{i,\mathrm{O}}-\boldsymbol{r}_{i,\mathrm{CO}}) \qquad (2)$$

$$\boldsymbol{\mu}_i^{\mathrm{NH}} = q_{\mathrm{N}}(\boldsymbol{r}_{i,\mathrm{N}}-\boldsymbol{r}_{i,\mathrm{NH}}) + q_{\mathrm{H}}(\boldsymbol{r}_{i,\mathrm{H}}-\boldsymbol{r}_{i,\mathrm{NH}}) \qquad (3)$$

$$\begin{aligned}\boldsymbol{\mu}_i = {} & q_{\mathrm{C}}(\boldsymbol{r}_{i,\mathrm{C}}-\boldsymbol{r}_i) + q_{\mathrm{O}}(\boldsymbol{r}_{i,\mathrm{O}}-\boldsymbol{r}_i) \\ & + q_{\mathrm{N}}(\boldsymbol{r}_{i,\mathrm{N}}-\boldsymbol{r}_i) + q_{\mathrm{H}}(\boldsymbol{r}_{i,\mathrm{H}}-\boldsymbol{r}_i)\end{aligned} \qquad (4)$$

$\boldsymbol{r}_{i,\mathrm{CO}}$, $\boldsymbol{r}_{i,\mathrm{NH}}$, and $\boldsymbol{r}_i$ are chosen so that the bond quadrupole moments of the CO and NH bonds, $\boldsymbol{Q}_{\mathrm{CO}}$, $\boldsymbol{Q}_{\mathrm{NH}}$, respectively, vanish, i. e., the three points satisfy the following relations:

$$\boldsymbol{Q}_{\mathrm{CO}} = q_C|\boldsymbol{r}_{i,C}-\boldsymbol{r}_{i,\mathrm{CO}}|^2 + q_O|\boldsymbol{r}_{i,O}-\boldsymbol{r}_{i,\mathrm{CO}}|^2 = 0 \qquad (5)$$

$$\mathbf{Q}_{\mathrm{NH}} = q_N|\boldsymbol{r}_{i,N} - \boldsymbol{r}_{i,\mathrm{NH}}|^2 + q_H|\boldsymbol{r}_{i,H} - \boldsymbol{r}_{i,\mathrm{NH}}|^2 = 0 \quad (6)$$

and,

$$\boldsymbol{r}_i = (\boldsymbol{r}_{i,C} - \boldsymbol{r}_{i,N})/2\,. \quad (7)$$

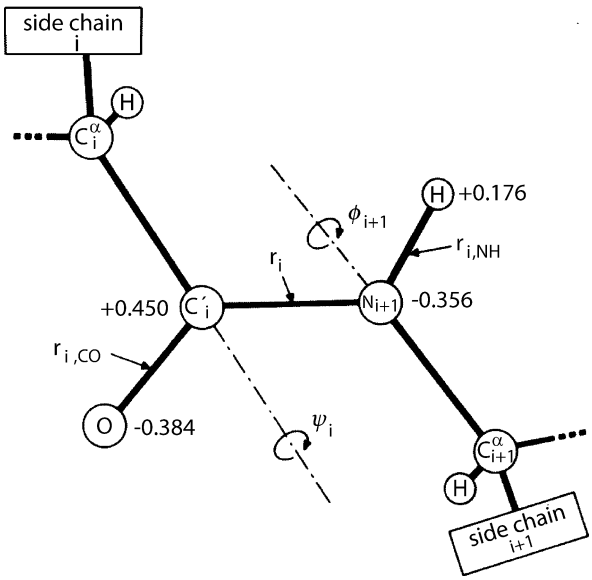
Degree of Alignment of a Dipole Moment with the Electric Field

The process of aligning a particular dipole moment, $\boldsymbol{\mu}_i^X$ (with X being CO or NH), with the electric field can be accomplished by rotations of the backbone dihedral angles ψ_i, and ϕ_{i+1} (see Fig. 2). When such a rotation is carried out, only the electric field components perpendicular to the rotation axis will change:

$$\mathcal{E}_{\perp k}(\boldsymbol{r}_{i,\mathrm{CO}}) = \mathcal{E}(\boldsymbol{r}_{i,\mathrm{CO}}) - [\mathcal{E}(\boldsymbol{r}_{i,\mathrm{CO}}) \cdot \boldsymbol{e}_{i,k}]\boldsymbol{e}_{i,k} \quad (8)$$

$$\mathcal{E}_{\perp k}(\boldsymbol{r}_{i,\mathrm{NH}}) = \mathcal{E}(\boldsymbol{r}_{i,\mathrm{NH}}) - [\mathcal{E}(\boldsymbol{r}_{i,\mathrm{NH}}) \cdot \boldsymbol{e}_{i,k}]\boldsymbol{e}_{i,k} \quad (9)$$

where $\boldsymbol{e}_{i,k}$ for $k = 1, 2$ denotes the unit vector along the axes of rotation, ψ_i, and ϕ_{i+1}, respectively. Furthermore, in writing these equations it was assumed that the points $\boldsymbol{r}_{i,\mathrm{CO}}$ and $\boldsymbol{r}_{i,\mathrm{NH}}$ are sufficiently close to the rotation axis.



Global Optimization in Protein Folding, Figure 2
SCEF peptide unit *i* with the atomic charges used in the ECEPP force field

The energy, E, of a dipole in an electric field is a given by:

$$E = -\boldsymbol{\mu} \cdot \mathcal{E}\,. \quad (10)$$

Assuming that the electric field in the neighborhood of the ith peptide group is relatively uniform, a lower bound for the energy gain due to a rotation is represented by:

$$\Delta E_i = \Delta E_i^{\mathrm{CO}} + \Delta E_i^{\mathrm{NH}} \quad (11)$$

where the individual energy gains, $\Delta E_i^{\mathrm{CO}}(< 0)$ and $\Delta E_i^{\mathrm{NH}}(< 0)$, to align the dipole and the field vectors are,

$$\Delta E_i^X = -|\boldsymbol{\mu}_{i,\perp k}^X|\,|\mathcal{E}_{\perp k}(\boldsymbol{r}_{i,X})| + \left[\boldsymbol{\mu}_{i,\perp k}^X \cdot \mathcal{E}_{\perp k}(\boldsymbol{r}_{i,X})\right] \quad (12)$$

with

$$\boldsymbol{\mu}_{i,\perp k}^X = \boldsymbol{\mu}_i^X - (\boldsymbol{\mu}_i^X \cdot \boldsymbol{e}_{i,k})\boldsymbol{e}_{i,k}\,. \quad (13)$$

The value of ΔE_i given by Eq. (11) is used as a measure of the deviation from perfect alignment in the electric field of the ith peptide unit.

Best-possible Alignment of a Dipole Moment with the Electric Field

From an analysis of the ΔE_is, it is possible to detect which peptide unit is the most unfavorably oriented in the electric field. The SCEF method provides a mechanism to compute the rotation that should lead to an improved orientation of this peptide unit with respect to the electric field. To accomplish this, the electric field $\mathcal{E}(\boldsymbol{r}_i)$ at the ith peptide unit can be viewed as the sum of two contributions generated by the portions of the polypeptide chain on both sides of the ith unit: (a) $\mathcal{E}_N(\boldsymbol{r}_i)$ generated by the part of the molecule containing the N-terminus; and (b) $\mathcal{E}_C(\boldsymbol{r}_i)$ generated by the part of the molecule containing the C-terminus,

$$\mathcal{E}(\boldsymbol{r}_i) = \mathcal{E}_N(\boldsymbol{r}_i) + \mathcal{E}_C(\boldsymbol{r}_i)\,. \quad (14)$$

The components of $\boldsymbol{\mu}_i$, $\mathcal{E}_N(\boldsymbol{r}_i)$ and $\mathcal{E}_C(\boldsymbol{r}_i)$ parallel to an axis of rotation do not change with rotations about this axis. On the other hand, the perpendicular components of these vectors with respect to a given

axis, say $\boldsymbol{e}_{i,k}$, do change with rotations about the axis and they are given by:

$$\boldsymbol{\mu}_{i,\perp k} = \boldsymbol{\mu}_i - \boldsymbol{e}_{i,k}(\boldsymbol{\mu}_i \cdot \boldsymbol{e}_{i,k}) \tag{15}$$

$$\mathcal{E}_{N,\perp k}(\boldsymbol{r}_i) = \mathcal{E}_N(\boldsymbol{r}_i) - \boldsymbol{e}_{i,k}\left[\mathcal{E}_N(\boldsymbol{r}_i) \cdot \boldsymbol{e}_{i,k}\right] \tag{16}$$

$$\mathcal{E}_{C,\perp k}(\boldsymbol{r}_i) = \mathcal{E}_C(\boldsymbol{r}_i) - \boldsymbol{e}_{i,k}\left[\mathcal{E}_C(\boldsymbol{r}_i) \cdot \boldsymbol{e}_{i,k}\right] . \tag{17}$$

If $\boldsymbol{\mu}_{i,\perp k}$ does not lie along $\mathcal{E}_{\perp k} = \mathcal{E}_{N,\perp k} + \mathcal{E}_{C,\perp k}$, perfect alignment between the vectors can be obtained by a *single* rotation about $\boldsymbol{e}_{i,k}$. For $k = 1$, a rotation about the ψ_i axis produces a change of $\mathcal{E}_{N,\perp 1}$ to $\mathcal{E}'_{N,\perp 1}$. Similarly for $k = 2$, a rotation about the ϕ_{i+1} axis leads to a change of $\mathcal{E}_{C,\perp 2}$ to $\mathcal{E}'_{C,\perp 2}$. Therefore, alignment is achieved if either *one* of the following equations is satisfied: for $k = 1$ (ψ_i axis),

$$\boldsymbol{\mu}_{i,\perp 1} \cdot (\mathcal{E}'_{N,\perp 1} + \mathcal{E}_{C,\perp 1}) = |\boldsymbol{\mu}_{i,\perp 1}|\,|\mathcal{E}'_{N,\perp 1} + \mathcal{E}_{C,\perp 1}| \tag{18}$$

for $k = 2$ (ϕ_{i+1} axis),

$$\boldsymbol{\mu}_{i,\perp 2} \cdot (\mathcal{E}_{N,\perp 2} + \mathcal{E}'_{C,\perp 2}) = |\boldsymbol{\mu}_{i,\perp 2}|\,|\mathcal{E}_{N,\perp 2} + \mathcal{E}'_{C,\perp 2}|. \tag{19}$$

From geometrical considerations (see Fig. 3), the solution of Eq. (18) (similarly for Eq. (19)) is found to satisfy the relation:

$$|\alpha| = \arccos(c/b) \tag{20}$$

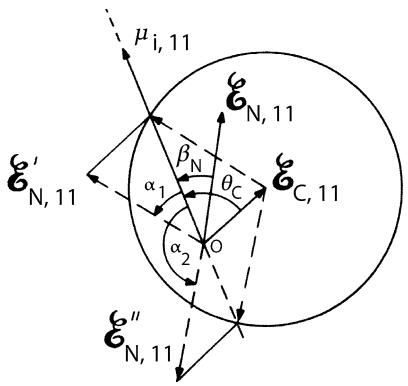
where $b = |\mathcal{E}_{N,\perp 1}|$, $c = d^{1/2}$ with $d = b^2 - a^2 \sin^2\theta_C$, $a = |\mathcal{E}_{C,\perp 1}|$, and θ_C is the angle between $\mathcal{E}_{C,\perp 1}$ and $\boldsymbol{\mu}_{i,\perp 1}$. Equation (20) has various numbers of solutions. If they exist, these solutions correspond to rotations of the dipole moment $\boldsymbol{\mu}_{i,\perp 1}$ with different energies. The value leading to the lowest energy represents the solution to the alignment Eq. (18). This rotation of ψ_i leads to an energy gain given by,

$$\Delta E_{i,N} = -\boldsymbol{\mu}_{i,\perp 1} \cdot (\mathcal{E}'_{N,\perp 1} - \mathcal{E}_{N,\perp 1}) . \tag{21}$$

Expressions similar to Eq. (20) and (21) have been derived for the rotation around the ϕ_{i+1} axis ($k = 2$) and for the corresponding energy gain, $\Delta E_{i,C}$.

It should be mentioned that, in reality, the solution given by Eq. (20) produces an approximate alignment of $\boldsymbol{\mu}_{i,\perp 1}$ with the corresponding electric field component. The reason is that the derivation of these equations was based on the assumptions that (a) the center of the peptide unit is on the ψ_i axis of rotation, and (b) the electric field is homogeneous. While, in reality, these conditions are not satisfied, the results obtained from these expressions are reasonably accurate [62].

Finally, after both rotations about the ψ_i and ϕ_{i+1} axis have been computed, the SCEF method has to decide which rotation should be implemented. The method selects the rotation associated with the more negative energy gain ($\Delta E_{i,N}$ or $\Delta E_{i,C}$). In those cases where no solution is found for ψ_i and ϕ_{i+1}, another unfavorable peptide unit is chosen.

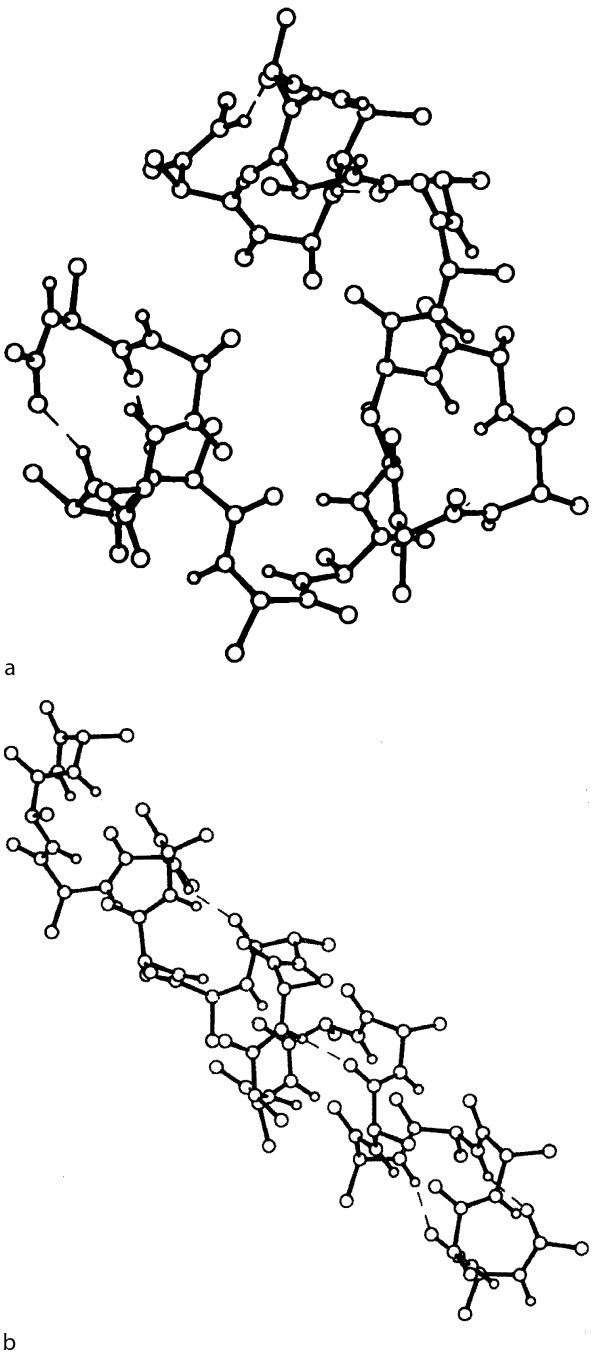


Global Optimization in Protein Folding, Figure 3
SCEF: solution of alignment Eq. (18)

Applications

The procedure was tested on a 19-residue poly(L-alanine) chain [62] with acetyl-and N-methyl amide terminal blocking groups. The starting conformations were a series of partially α-helical conformations representing different degrees of distortion from the canonical right-handed α-*helix*. The right-handed α-helical conformation corresponds to the global energy minimum of the ECEPP/2 (and ECEPP/3) potential function. In the four cases reported, the procedure was able to achieve the conformation corresponding to the global energy minimum in a very short computation time.

Figure 4a shows the starting conformation of one of the tests. The conformation contains only 1.5 α-helical turns at each terminus and 70.6% of the native hydrogen bonds are broken. In subsequent iterations of the SCEF procedure, the right-handed α-helix shown at the bottom of Fig. 4b, was completely recovered.



Global Optimization in Protein Folding, Figure 4
SCEF method: application to poly(L-alanine)

The SCEF procedure was also used [76] in a restrictive search of the conformational space of the 58-residue protein bovine pancreatic trypsin inhibitor (BPTI). In this application, the algorithm led to a series of conformations with up to 50 kcal/mol lower than the starting conformation.

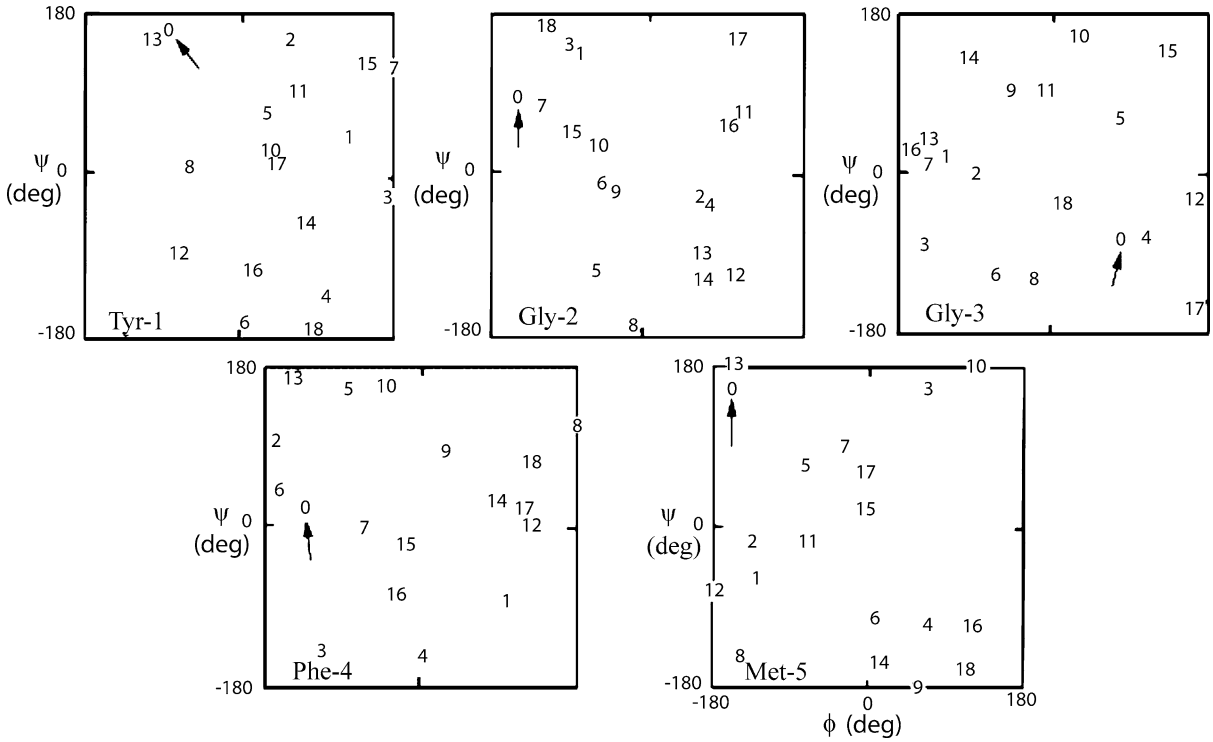
The Monte Carlo-Minimization Method

The Monte Carlo-Minimization (MCM) [26], [27] method developed by Z. Li and H.A. Scheraga was motivated by experimental studies indicating that proteins are not static structures but instead undergo fluctuations. For a protein to be stable, its *native conformation* must be stable not only to small perturbations but also against larger-scale *thermal fluctuations*. Based on these considerations, Li and Scheraga developed a *stochastic approach* for *global optimization* of polypeptides and proteins that combines the power of the Metropolis Monte Carlo method [40] in global *combinatorial optimization* and that of conventional *energy minimization* to find local minima. The underlying working hypothesis of the method is that protein folding can be considered as a *Markov process*, with (a) Boltzmann transition probabilities, and (b) this Markov process should lead to a unique absorbing state [3] that corresponds to the native state for a natural biologically active protein. For this absorbing state, equilibrium is reached after a sufficiently long time and the stationary probability of occurrence approaches unity.

The *Metropolis Monte Carlo method* can simulate the thermal processes, by taking into account both random fluctuations and energetic considerations. However, straightforward applications of the Metropolis Monte Carlo method to polypeptides has proven to be quite inefficient [10,57,74] mainly because (a) a high-dimensional conformational space has to be sampled by making small increments of the variables in each step, and (b) The large energy barriers in the conformational space tend to confine the sampling to a very restrictive region of the space. To overcome these difficulties, the MCM method includes conventional energy minimization as a second important feature. Thus, the MCM method generates a Markov walk on the hyperlattice of all discrete energy minima, with Boltzmann transition probabilities.

The procedure implemented in the MCM algorithm is as follows:

- Given an energy-minimized conformation, $C^{\min}_{\text{curr}}$, with total energy $E^{\min}_{\text{curr}}$, a Monte Carlo sampling strategy is used to generate a perturbed conforma-



Global Optimization in Protein Folding, Figure 5
$\phi - \psi$ maps for the five residues of Metenkephalin showing the backbone dihedral angles of 18 random starting conformations (indicated by the numbers 1 to 18) for the MCM method. The backbone dihedral angles of the global minimum achieved by the MCM method in all the runs (see Fig. 1) are indicated by 0

tion C_{pert}. The sampling strategy consists of random changes, involving k dihedral angles of the total number N_{dieh} used to described the molecule. The number of changes are generated with probabilities 2^{-k} ($k = 1, 2, \ldots, N_{\text{dieh}}$). This probability selection implies that fluctuations involving more degrees of freedom are sampled with successively lower probabilities. This sampling strategy satisfies the ergodicity requirements, i. e., any local minimum is accessible from any other one after a finite number of random sampling steps. Furthermore, in order to improve the average acceptance ratio, random changes involving backbone dihedral angles are sampled more frequently than those of side chains. This type of sampling strategy led to an average acceptance ratio of approximately 20% at 0°C for Metenkephalin.

- The randomly generated conformation, C_{pert}, is then subjected to conventional energy minimization until it reaches the nearest local minimum of the *potential energy function* (ECEPP/2 or ECEPP/3). Minimization of the energy is carried out with the Secant Unconstrained Minimization Solver (SUMSL) algorithm [8]. The resulting conformation, $C^{\min}_{\text{pert}}$, has a total energy $E^{\min}_{\text{pert}}$ and is usually free of atomic overlaps.
- The energies of the conformations $C^{\min}_{\text{pert}}$ and $C^{\min}_{\text{curr}}$ are compared, and the Metropolis criterion is used to decide which conformation is to be kept, i. e., if the energy difference $\Delta E = E^{\min}_{\text{pert}} - E^{\min}_{\text{curr}} < 0$, or (when $\Delta E > 0$) if $e^{-\Delta E/RT}$ is greater than a randomly generated number between 0 and 1, the new conformation, $C^{\min}_{\text{pert}}$ replaces the current $C^{\min}_{\text{curr}}$; otherwise, $C^{\min}_{\text{pert}}$ is discarded.

Applications

The MCM procedure was successfully applied to study the conformational preference of the pentapeptide Metenkephalin [26,27]. In its initial application [26], 13

of 18 random starting conformations of this oligopeptide converged to the global minimum, shown in Fig. 1, within the time of the simulations. Using a different sampling strategy [27], the 5 remaining runs also converged to the same lowest energy structure. Figure 5 shows the values of the backbone dihedral angles, ϕ and ψ for the 18 starting conformations.

As a further development, we extended the concept of MCM to include biasing the perturbations to electrostatic interaction, giving the Electrostatically Driven Monte Carlo (EDMC) method, which is described in the next section. More recently, we took advantage of grouping the conformations obtained in the search into families which are updated on the fly and, using the properties of the families in the subsequent steps of the search, this resulted in the conformation-family Monte Carlo (CFMC) method [65]. The CFMC method was used to search the conformational space of the B-domain of staphylococcal protein A in the united-residue representation [65] and for crystal structure prediction of small molecules [63].

The Electrostatically Driven Monte Carlo Method

The Electrostatically Driven Monte Carlo (EDMC) method, introduced by D.R. Ripoll and H.A. Scheraga, is a procedure for iteratively searching the conformational hypersurface of relatively small polypeptide molecules. The EDMC method incorporates the best features of the SCEF and MCM methods and combines them with a set of new techniques to produce a more efficient search of the conformational space.

The search for the the global energy minimum of a molecule proceeds as a "quasi-random walk" along a conformational pathway. As with the MCM method, this pathway is defined, in principle, by an infinite sequence of energy-minimized conformations encountered over an unbounded number of iterative steps of the algorithm. In practice, however, a finite number of iterations is specified for a given run. The underlying assumption behind the EDMC method is that (a) the electrostatic interactions should lead to conformations representing an improvement of the charge distribution, i. e. the new conformations are expected to have lower electrostatic and total energies; and (b) thermal fluctuations, on the other hand, are expected to introduce disorder within the molecule. These thermal effects could force the molecule to adopt conformations that are higher in energy, but may allow it to escape from stable local minima of relatively high energy.

The implementation of these ideas is accomplished as follows: Thermal effects are associated with random changes in the molecular conformation, i. e. a small set of randomly-chosen variables was altered randomly. On the other hand, the reordering effect of the electrostatic interactions was viewed as a tendency of all permanent dipole moments associated with the peptide units of the polypeptide, to attain their best possible alignment in the local electric field produced by the rest of the molecule. Additionally, a series of new features [77], included in the latest implementation of the EDMC method, has helped to accelerate the search and to optimize the process of generation of new conformations.

The Procedure

The first accepted conformation on the conformational pathway followed by the EDMC method is usually an unfolded state of the polypeptide chain (i. e. the initial values of the variables describing the molecular conformation are assigned randomly); its energy is minimized to relieve possible atomic overlaps. The subsequent accepted conformations are obtained by a variety of techniques described below. An *iteration* of the procedure is defined as a set of manipulations of the currently accepted conformation that leads to its *replacement* by a newly generated conformation.

The strategy used to produce new conformations within an iteration of the method is based upon a combination of movements associated with the electrostatic interactions and thermal motion.

(a) An important technique that the EDMC method uses to generate new conformations is based on an electrostatic analysis similar to that produced by the SCEF method [62], but extended to consider the permanent dipole moments of polar side-chains. As a first step of an iteration, this electrostatic analysis of the currently accepted conformation (the initial energy–minimized conformation or the accepted conformation from the previous iteration) is carried out to determine the alignment of the permanent dipoles with the local electric

field produced by the whole molecule. As a result, *diagnostic rotations* that could improve the local dipole alignments with the electric field are produced for all permanent dipole moments. These diagnostic rotations are incorporated into a *prediction list* of possible conformational changes. The information contained in this list is used to generate new conformations in a subsequent search for states of lower energy.

(b) Since it may happen that none of these predictions leads to an acceptable conformation, a random and/or biased sampling technique is also used to generate additional conformations. The following procedure is followed:
   1. Specification of the mode in which the variable dihedral angles of the *selected residues* are to be altered:
      i) Select all variables at random;
      ii) Select the backbone variables randomly within specific regions of the $\phi - \psi$ map;
      iii) Select all variables from pre-computed low-energy conformations of the tri-peptides included in the sequence;
      iv) Select backbone variables compatible with regular structures β-sheets or α-helices).
   2. Random selection of i) the number of residues to be affected by the changes, and ii) their positions in the sequence.

The latest implementation of the algorithm [77] includes a technique to produce a *cluster analysis* of the accepted minima. The conformations are grouped into clusters using rms distance criteria and ranked on the basis of their total energies. Furthermore, every generated conformation, even if rejected, is associated with an existing cluster or family, but added to it only if its energy is lower than the one corresponding to the best member of that family.During an iteration, randomly generated conformations can also be produced by perturbing low-energy conformations included in any of the clusters (except the one containing the current accepted minima) using the protocol described in item (b) above.

A conformation generated by any of these two procedures (a or b) is subjected to minimization of the total energy where the backbone and side-chain dihedral angles of the molecule are considered as variables. The energy-minimization procedure is carried out with the SUMSL algorithm [8]. The value of the potential energy constitutes the basis for either the acceptance or rejection of the new minimum-energy conformation. A newly generated conformation must fulfill two criteria to be accepted:

1. If a generated conformation is found to correspond to an accepted minimum that has already been encountered more than a pre-defined number of times (usually 5–10), then it is automatically excluded from further consideration. This analysis of the long–term behavior of the search provides one of the criteria to ensure that the search does not become trapped in a set of local minima of the conformational space.
2. If a conformation satisfies the previous condition, its energy E_{new} is compared with the energy, E_{curr}, of the current accepted conformation, and the Metropolis criterion [40], as described for the MCM method, is applied.

When the energy of the new conformation passes both tests successfully, the conformation is accepted, replacing the current one, and a new iteration begins.

Backtrack

The number of conformations generated within a given iteration is limited (usually 100 to 200 conformations). It may happen that neither the set of electrostatic predictions, nor the set of randomly generated conformations produces an acceptable conformation. Under these circumstances, the algorithm then assumes that the current local minimum is quite stable and a new procedure named *backtrack* is triggered. The backtrack procedure attempts to displace the search to a different region of the conformational hypersurface by substantially altering the processes of generation and acceptance of conformations.

The backtrack procedure involves the following:

a) A new set of conformations is generated by changing a large number of variables simultaneously. In particular, the procedure tends to select the variables associated mainly with the backbone of the polypeptide chain; and,
b) the temperature parameter, T, used in the Metropolis acceptance criterion is *(i)* raised abruptly to a very high value, or *(ii)* steadily increased by means of a pre-defined heating scheme.

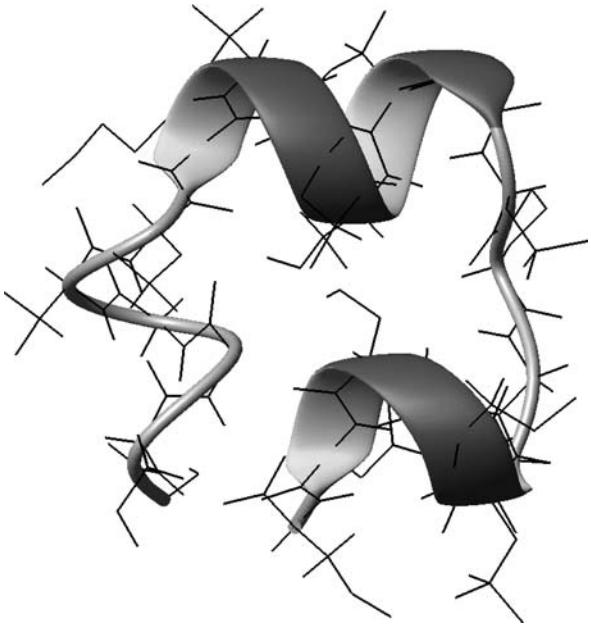
The backtrack procedure is applied until the acceptance test is satisfied, or until the number of generated conformations reaches a predetermined maximum value. In the rare event that the latter situation occurs, the run is terminated since it is assumed that it is practically impossible to escape from the current region of the conformational space. On the other hand, when a conformation from the backtrack procedure is accepted, the temperature parameter is reset to its original user-specified value, and the generation mechanism is switched back to the standard protocol described above.

The objective of the modified generation procedure during backtrack is to produce conformations substantially different from the current minima, while raising the temperature has the effect of increasing the probability of acceptance of conformations with energies much higher than the current local minimum. The backtrack mechanism has been shown to be an effective technique to help the search avoid being trapped in stable, high-energy regions of the conformational space.

The EDMC method has some similarities with *simulating annealing*, proposed by S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi [15], since both make use of high temperatures to surmount large energy barriers. The difference is that the EDMC procedure concentrates the search in the low-energy regions of the conformational space using energy minimization and a low temperature value. High temperatures are used rarely during backtrack to escape from stable or already visited regions. Once this is accomplished, the temperature parameter is reset to its initial (low) value. A search using simulated annealing, on the other hand, starts with a high temperature value and this parameter is gradually reduced during the simulation. The expectation is that, given a *sufficiently high* initial temperature and a *good* annealing schedule, the search will overcome large energy barriers and will become localized in the low-energy region containing the global minimum.

Applications

The multiple-minima problem has been found to be computationally tractable by the EDMC method on existing computers for polypeptides sequences consisting of up to 20 amino acid residues.



Global Optimization in Protein Folding, Figure 6
Lowest-energy conformation of the membrane-bound portion of melittin for the ECEPP/3 force field determined by the Conformational Space Annealing [23] and the EDMC [77] methods

In applications to Metenkephalin [79], oxytocin [39], arginine-vasopressin [39], decaglycine [80], a 19-residue chain of poly(L-alanine) [78], and the 20-residue membrane-bound portion of melittin [77] (see Fig. 6), the EDMC algorithm converged to unique conformations presumed to be the global energy minima for those particular sequences.

In other applications, to a seven-residue peptide epitope [75], and a twelve-residue analogue of mastoparan and mastoparan X [7], the method identified very low-energy conformations, but it is not certain that the global energy minima were attained in these cases.

Lately, the EDMC method has been applied to the 36-residue villin headpiece subdomain [81], and the 45-residue fragment B-domain of staphylococcal protein A [94]. In both applications, unrestricted global searches that started from randomly generated conformations encountered in their paths low-energy basins that included native-like conformations. To our knowledge, the application to the B-domain of staphylococcal protein A was, at the time, the first *all-atom* sim-

ulation in which such a large protein was ever folded from random initial conformations without resort to knowledge-based information.

The EDMC method has also been used in restrictive searches of the conformational space of larger molecules. In an application to the 58-residue protein BPTI [76], the algorithm produced the lowest energy conformation known for BPTI using the ECEPP/2 or ECEPP/3 potential. In addition, the EDMC method has also been used to search the conformational properties of a non-oncogenic p21 protein [30] and a molecular switch designed as a biological logic gate [2].

The Diffusion Equation Method and Other Methods Based on the Deformation of the Potential-Energy Surface

The *diffusion equation method* (DEM) is a deterministic approach that attempts to solve the multiple-minima problem by deforming the potential energy hypersurface. The basic idea of the method, introduced by Piela et al.[61], is to deform the multivariable function that represents the potential energy in such a manner as to make the shallow wells disappear gradually, while other potential wells grow at their expense. Under the assumption that the shallower wells will disappear more easily than the deep wells, it is possible to envision an iterative procedure that, applied to the potential function, will change its shape, making most of the minima become shallower until they disappear, while leaving a single absorbing minimum related to the lowest minimum of the original function. At this point of the *deformation process*, a simple local minimization algorithm should be able to retrieve the position of the unique minimum from any starting point. However, since the deformation of the potential should likely have altered the location of all minima, the global minimum of the original function is not the same as the minimum of the deformed surface. Its location can, in principle, be attained by slowly reversing the deformation and using standard local minimization procedures. Piela et al. showed that the deformation of the hypersurface can be carried out with the aid of the diffusion equation. In this context, the original shape of the potential function has the meaning of an initial concentration (or temperature) distribution.

The diffusion equation method which must be solved to obtain a deformed potential-energy surface is given by Eq. (22).

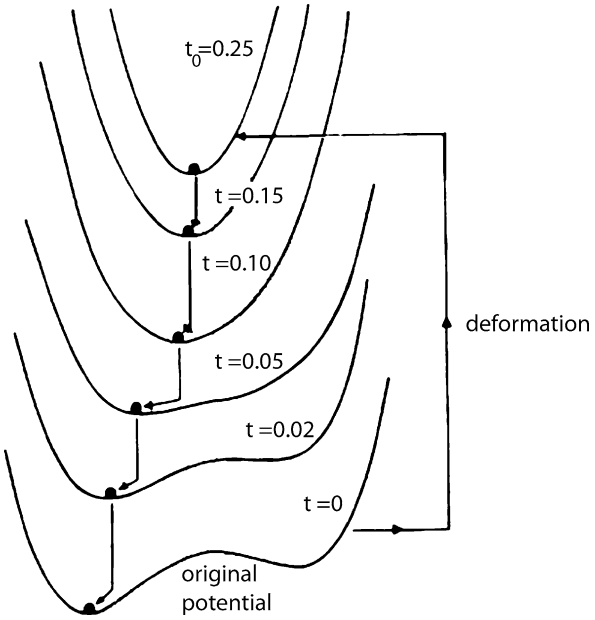
$$\nabla^2 F(x_1, x_2, \ldots, x_n; t) = \frac{\partial F(x_1, x_2, \ldots, x_n; t)}{\partial t} \quad (22)$$

where $x_1, x_2, \ldots, x_n$ are variables describing the conformation of a molecule, $\nabla^2 = \left(\partial^2/\partial x_1^2, \partial^2/\partial x_2^2, \ldots, \partial^2/\partial x_n^2\right)$ is the Laplacian operator, the variable t represents time and can be identified with the extent of deformation, and F is the deformed potential-energy function. Additionally, Eq. (22) is solved with the initial condition $F(x_1, x_2, \ldots, x_n; 0) = f(x_1, x_2, \ldots, x_n)$, where $f(x_1, x_2, \ldots, x_n)$ is the original (undeformed) potential-energy function. The function F usually represents a concentration or a temperature distribution. If the function $f(x_1, x_2, \ldots, x_n)$ is bounded, a solution of Eq. (22) exists for any positive value of t.

The procedure described above represents a spontaneous mass transport (or flow of heat) in a medium for an initial distribution of concentration (or temperature) given by the function $f(x_1, x_2, \ldots, x_n)$ (which in our case represents the conformational energy). Governed by the diffusion equation and independent of the initial conditions, the concentration (or temperature), will evolve with time in such a manner that it will become constant for $t = \infty$. However, it is expected that the concentration (or temperature) will exhibit a single minimum for certain (very large) values of t. This single minimum should represent the last trace of the potential well corresponding to the global minimum of the original hypersurface $f(x_1, x_2, \ldots, x_n)$. The deformation and its subsequent reversal to retrieve the position of the original minimum is illustrated in Fig. 7.

Application of the DEM consists of the following steps:

- Solve Eq. (22) using $F(x, 0) = f(x)$ as the initial condition or apply the operator $T(t)$ for a sufficiently large value of t (t_0); then, use a local minimization to locate the position $x^*_{t_0}$ of the unique minimum on the deformed surface. This is the starting point to be used in the reversing procedure.
- Apply the reversing procedure described above.
- For a reversing procedure involving m steps, the position x^*_0 obtained by minimizing $F(x^*_{t_0-(m-1)\Delta t}, t_0 = 0)$ should correspond, hopefully, to the position of the global minimum of the function f.



Global Optimization in Protein Folding, Figure 7
The DEM method: Illustration of the deformation of the original potential $f(x) = x^4 + 2x^3 + 0.9x^2$ by the operator $T(t) = exp(td^2/dx^2)$, and of the reversing procedure. The deformation applied by the operator $T(t_0 = 0.25)$ leads to a curve with a unique minimum that is achievable from any point of the space with a simple minimization. The reversing procedure is shown by the arrows directed downward. Each step of the reversing procedure is followed by minimization symbolized in the figure by a ball moving down hill from the minimum position of the upper curve and always reaching the position of the minimum in the lower curve. In the final step, the global minimum of the original function is found

Among other applications, the DEM has been applied to:

- A cluster of 55 *Lennard-Jones* atoms for which the global minimum was found [16].
- A single terminally blocked alanine [17].
- The pentapeptide Met-enkephalin [17] for which the method led to practically the same global-minimum backbone structure obtained by other methods. The test, however, was carried out under more restrictive conditions since only the backbone dihedral angles ϕ and ψ were considered as variables.
- Prediction of the crystal structures of hexasulfur and benzene molecules [96,97].

Although the DEM method is, in theory, a deterministic approach, we found [96,97] that it must be combined with a Monte Carlo search to work for more complex systems. When the potential-energy surface is deformed to contain just a single minimum, it is so flat that, to the numerical accuracy, it is effectively constant. Thus, deformation cannot be carried out to leave only one minimum. Moreover, the position of a minimum on a highly deformed surface is too far from that on the original energy surface. During the process of reversal, the single minimum splits into multiple minima and it is not clear which one of those should be chosen to continue the reversal. In our successful application to crystal-structure prediction [96,97] we, therefore, introduced the MCM search both on the deformed potential-energy surface and during reversal.

Taking advantage of the concept of the deformation of potential-energy surfaces, we developed several other methods for the search of the global minimum of the energy of polypeptide and proteins. The *distance scaling method* (DSM) [70] developed by J. Pillardy and L. Piela, (as well as its predecessor, the shift method (SM) [68]) attempts to solve the multiple-minima problem using transformations of the atom–atom distances that lead to *smoothing of the potential energy* hypersurface. These methods have subsequently evolved into the Self-Consistent Basin-to-Deformed-Basin Mapping (SCBDBM) method, in which the coupling between the basin containing the global energy minimum to the corresponding basin in the deformed potential-energy surface is established. The SCBDBM involves some Monte Carlo search on the deformed potential-energy surface and during the process of reversal. All three methods have been applied successfully to clusters of *argon atoms* and water molecules [67,68,69,70] and to the *prediction of crystal structures* [97]. The SCBDBM method was also applied [66] in searches for low-energy minima of poly-L-ananine chains of up to 100 amino-acid residues in length and the 10–55 fragment of the B-domain of staphylococcal protein A using a united-residue representation of the polypeptide chain. As opposed to DEM, the SM, DSM, and SCBDBM approaches, although not so elegant from the theoretical point of view, involve simple transformations of the potential-energy surface and are, therefore, much better for practical use than DEM, which requires solving a parabolic differential equation in multiple dimensions and with complicated boundary conditions, which is a highly non-trivial task.

Another approach related to deformation of the potential-energy surface has been developed by K.A. Olszewski, L. Piela and H.A. Scheraga [55] and termed Self-Consistent Mean Torsional Field (SCMTF) method. It is based on the idea that the ground-state solution of the *Schroedinger equation* contains information about the location of the global minimum. Their implementation uses a *mean field approximation* to solve a set of coupled Schroedinger equations in a dihedral-angle space. Each equation describes the changes of a single dihedral angle in the averaged field of the others. This approach was successful in finding the lowest-energy conformations of Met-enkephalin [55], and decaglycine and eikosaalanine chains [56].

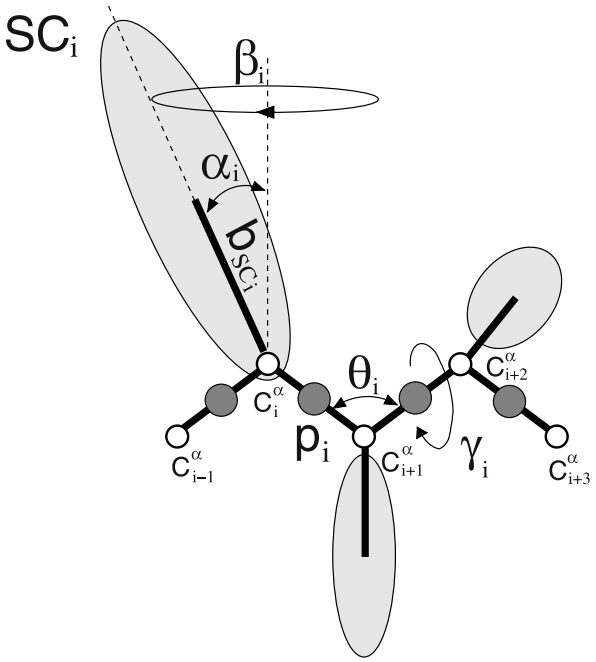
The Conformational Space Annealing Method

One of the most efficient methods to search the conformational space of polypeptide chains developed in our laboratory is the Conformational Space Annealing (CSA) method [19,21,22,24], which combines the ideas of genetic algorithms, the build-up procedure, random search, and local minimization. The CSA method begins with a randomly-generated population of conformations which are energy minimized to generate the *first bank* of conformations. The first bank is meant to represent a sparse sampling of the conformational space that captures short-range interactions. From the initial population, a number of conformations (called seeds) are selected as parents for the trial population. These "seed" conformations are altered in a non-random fashion to create new trial conformations. As in any genetic algorithm, the trial population is generated by the use of genetic operators: mutations and crossovers. Unlike traditional genetic algorithms, the mutation operator applied in CSA does not change the value of the selected variable randomly; instead it uses values of the corresponding variables in the initial population (the first bank) or in the current population of conformations as a pool of random numbers. A copy of the first bank is used as a source of "random" variables, which are not uniformly distributed but their distribution is determined by intramolecular interactions at this stage, mainly by steric overlap. The crossover operators copy a set of variables representing a continuous segment of the polypeptide chain of various size taken from a randomly selected conformation in the current population to a selected parent conformation (seed). This is described in detail in the next section. Attention is paid to assure that all trial conformations are significantly different from each other and from the parent conformations. After generation, all trial conformations are energy minimized. The next step of the CSA algorithm is the update of the current population (the bank) without increasing its size. Each trial conformation is compared to each existing conformation of the bank. If the trial conformation is similar to an existing conformation of the bank, only the lower-energy conformation out of these two is preserved. If the trial conformation is not similar to any existing conformation in the bank it represents a new distinct region of conformational space. Then it replaces the highest-energy conformation in the bank, if its energy is lower than the highest energy in the bank, otherwise it is discarded. The distance between conformations i and j is defined as the difference of their dihedral angles. If the distance, D_{ij}, is less than or equal to some predefined cutoff value, D_{cut}, conformations i and j are considered similar, otherwise they are considered different. CSA achieves its efficacy by beginning with a large D_{cut} value to essentially search all possible structures, and then gradually reduces ("anneals") D_{cut} by reducing the minimum distance between the conformations of the bank and focusing the search in low-energy regions of conformational space. After updating the current population, the seed conformations are selected from the set of conformations not selected as seeds previously; additionally attention is paid to cover the conformational space as broadly as possible by selecting conformations not similar to each other as seed conformations.

The CSA method was shown to be very efficient in finding the global minimum of the ECEPP/3 potential energy function for Metenkephalin [22] and melittin [24]; it was also implemented as a standard search technique with the coarse-grained UNRES force field developed in our laboratory (see next section).

Hierarchical Approach

Another approach developed in our laboratory [38,87] starts with a coarse-grained representation of a protein and provides atomistic details at the end. It can be summarized in the following three stages:



Global Optimization in Protein Folding, Figure 8
The UNRES model of polypeptide chains. The interaction sites are side-chain centroids of different sizes (SC) and the peptide-bond centers (*p*) indicated by shaded circles, whereas the α-carbon atoms (small empty circles) are introduced only to assist in defining the geometry. The virtual C^{α}–C^{α} bonds have a length of 3.8 Å, corresponding to a trans peptide group; θ and γ, denoting the virtual-bond angle and virtual-bond dihedral angle, respectively, are variable. Each side chain is attached to the corresponding α-carbon with a "bond length", b_{SC_i}, variable "bond angle", α_{SC_i}, formed by SC_i and the bisector of the angle defined by C^{α}_{i-1}, C^{α}_{i}, and C^{α}_{i+1}, and with a variable "dihedral angle" β_{SC_i} of counterclockwise rotation about the bisector, starting from the right side of the C^{α}_{i-1}, C^{α}_{i}, C^{α}_{i+1} frame

1 Extensive simulations with using the coarse-grained UNRES model [28,29,35,36,37,53,54] developed in our laboratory and subsequent selection of structures with the lowest free energy.
2 Conversion of selected coarse-grained structures to all-atom structures.
3 Exploration of the conformational space of all-atom structures in the neighborhood of geometries obtained in Stage 2.

In the UNRES model, a polypeptide chain is represented as a sequence of α-carbon atoms (C^{α}) with attached united side chains (SC) and united peptide groups (p), each of which is positioned in the middle between two consecutive C^{α} atoms, as shown in Fig. 8.

All three stages are executed using physics-based potentials; therefore, energy is the determinant of each of them. Stage 1 is the key point of the approach, because it provides the widest range of exploration of the conformational space. Consequently, we have put most of our effort in the development of the coarse-grained UNRES force field. To execute stage 2, we developed an approach in which the peptide groups are positioned first within an α-carbon trace to minimize their energy of local and electrostatic interactions [13] and, subsequently, the side-chain atoms are added to minimize the energy of the chain given a coarse-grained geometry [14]. The all-atom ECEPP/3 [49] force field is used in stage 3.

The effective energy function is a sum of different terms corresponding to interactions between the SC ($U_{SC_iSC_j}$), SC and p ($U_{SC_ip_j}$), and p ($U_{p_ip_j}$) sites, as well as local terms corresponding to bending of virtual-bond angles θ (U_b), side-chain rotamers ($U_{\rm rot}$), virtual-bond torsional ($U_{\rm tor}$) and double-torsional ($U_{\rm tord}$) terms, virtual-bond-stretching ($U_{\rm bond}$) terms, correlation terms ($U^{(m)}_{\rm corr}$) pertaining to coupling between backbone-local and backbone-electrostatic interactions [29] (where m denotes the order of correlation), and a term accounting for the energetics of disulfide bonds (U_{SS}). Each of these terms is multiplied by an appropriate weight, w. The energy function is given by Eq. (23).

$$\begin{aligned} U = {} & w_{\rm SC}\sum_{i<j} U_{SC_iSC_j} + w_{SCp}\sum_{i\neq j} U_{SC_ip_j} \\ & + w_{pp}\sum_{i<j-1} U^{el}_{p_ip_j} + w_{\rm tor}\sum_i U_{\rm tor}(\gamma_i) \\ & + w_{\rm tord}\sum_i U_{\rm tord}(\gamma_i,\gamma_{i+1}) + w_b\sum_i U_b(\theta_i) \\ & + w_{\rm rot}\sum_i U_{\rm rot}(\alpha_{SC_i},\beta_{SC_i}) \\ & + \sum_{m=3}^{6} w^{(m)}_{\rm corr}U^{(m)}_{\rm corr} \\ & + w_{\rm bond}\sum_{i=1}^{n\rm bond} U_{\rm bond}(d_i) + w_{SS}\sum_i U_{SS;i}\,. \end{aligned} \tag{23}$$

The expression for the effective energy in the UNRES model was derived based on the physics of interactions, as a cluster-cumulant [18] expansion of the effective free energy of a protein plus the surround-

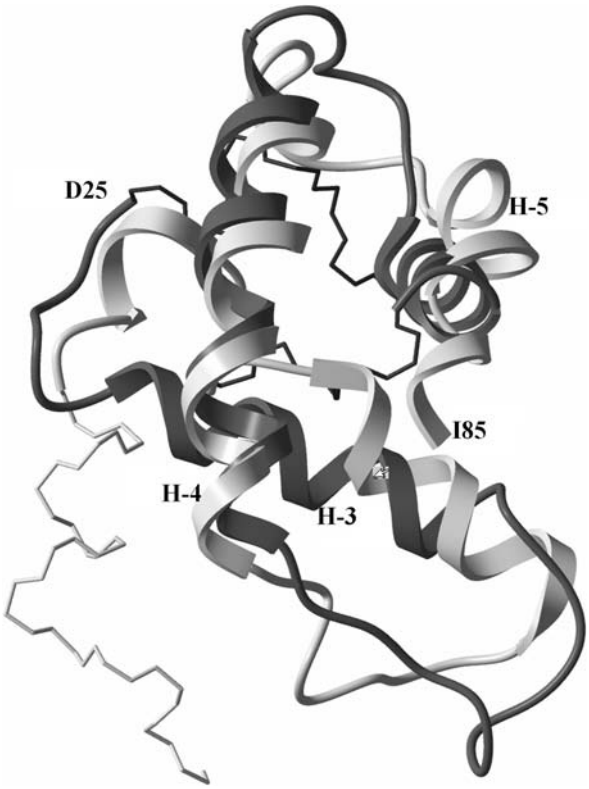
ing solvent, in which the secondary degrees of freedom had been averaged out [29,31,35]. Most of the expressions were parameterized based on energy surfaces of models systems computed by ab initio molecular quantum mechanics [35,53]; some of them were parameterized based on the statistics from the PDB [36,37]. The energy-term weights (the w's in Eq. (23)) were determined [54] by using the method of hierarchical optimization of the potential-energy landscape developed in our laboratory [28], in which the energy of selected training proteins decreases with increasing native-likeness.

Using the Conformational Space Annealing (CSA) method [19,21,22] to search for the global energy minimum of the UNRES energy function, we achieved considerable success in the Community Wide Experiments of Techniques for Protein Structure Prediction (CASP). In CASP3, we made the best prediction for target T0061 (protein HDEA), predicting its 60-residue segment within 4.2 Å C^{α} RMSD from the experimental structure (PDB code: 1BG8) [34]. The experimental and predicted structures are superposed in Fig. 9.

At that time, our force field did not contain sufficient correlation terms and was unable to account for β-sheet formation. After introducing correlation terms [29], in the CASP4 – CASP6 experiments we were able to predict significant portions of the structures of $\alpha + \beta$ and β proteins [52,64]. In the CASP6 experiment [52], we predicted complete structures of five proteins and large portions of structure of other protein *without* ancillary information from protein structural databases. The largest α-helical protein, the whole of which except for a short C-terminal fragment was predicted in CASP6 was target T0198 (235 residues; we predicted the topology of its 208-residue α-helical part) and the largest $\alpha + \beta$ protein was T0230 (97 residues).

We extended our hierarchical approach to treat oligomeric proteins [83,84] and to proteins containing disulfide bonds [5]; the second extension includes the energy-based prediction of disulfide-bond topology.

Recently [33] we extended the implementation of the UNRES force field to mesoscopic dynamics. The corresponding simulations led us to the conclusion that conformational entropy makes a major contribution to the probability of occurrence of a family of conformations. A particular single conformation can have a very low *potential* energy but no chance to appear at room temperature if it belongs to a very narrow basin in the potential-energy surface. On the contrary, higher-energy conformations could form a very broad basin and, consequently, make an overwhelming contribution to the statistical ensemble at room temperature. Consequently, in our latest work [32] we have reformulated energy-based protein-structure prediction as a search of the basin with the lowest free energy at physiological temperatures, by using techniques based on molecular dynamics, such as replica-exchange molecular dynamics [47] to search conformational space.



Global Optimization in Protein Folding, Figure 9
Superposition of the crystal (dark grey) and predicted (light gray) structures of HDEA. The C^{α} atoms of the fragment included between residues D25 to I85 were superposed. The RMSD is 4.2 Å. Helices 3, 4 and 5 are indicated as H-3, H-4 and H-5, respectively

References

1. Androulakis IP, Maranas CD, Floudas CA (1997) Prediction of oligopeptide conformations via deterministic global optimization. J Glob Optim 11:1–34

2. Ashkenazi G, Ripoll DR, Lotan N, Scheraga HA (1997) A molecular switch for biological logic gates: conformational studies. Biosens Bioelectron 12:85–95
3. Bharucha-Reid AT (1960) Elements of the theory of Markov processes and their applications. McGraw-Hill, New York
4. Chou K-C, Némethy G, Scheraga HA (1983) Energetic approach to the packing of α-helices. 1. Equivalent helices. J Phys Chem 87:2869–2881
5. Czaplewski C, Ołdziej S, Liwo A, Scheraga HA (2004) Prediction of the structures of proteins with the UNRES force field, including dynamic formation and breaking of disulfide bonds. PEDS 17:29–36
6. Dygert M, Gō N, Scheraga HA (1975) Use of a symmetry condition to compute the conformation of gramicidin S. Macromolecules 8:750–761
7. Faerman CH, Ripoll DR (1992) Conformational analysis of a twelve-residue analogue of mastoparan and mastoparan X. Proteins Struc Func Gen 12:111–116
8. Gay DM (1983) Algorithm 611. Subroutines for unconstrained minimization using a model/trust-region approach. ACM Trans Math Softw 9:503–524
9. Gibson KD, Scheraga HA (1987) Revised algorithms for the build-up procedure for predicting protein conformations by energy minimization. J Comput Chem 8:826–834
10. Hagler AT, Stern PS, Sharon R, Becker JM, Naider F (1979) Computer simulation of the conformational properties of oligopeptides. Comparison of theoretical methods and analysis of experimental results. J Am Chem Soc 101:6842–6852
11. Hol WGJ (1985) The role of the α-helix dipole in protein function and structure. Prog Biophys Molec Biol 45: 149–195
12. Hol WGJ, Halie LM, Sander C (1981) Dipoles of the α-helix and β-sheet: their role in protein folding. Nature 294: 532–536
13. Kaźmierkiewicz R, Liwo A, Scheraga HA (2002) Energy-based reconstruction of a protein backbone from its α-carbon-trace by a Monte Carlo method. J Comput Chem 23:715–723
14. Kaźmierkiewicz R, Liwo A, Scheraga HA (2003) Addition of side chains to a known backbone with defined side-chain centroids. Biophys Chem 100:261–280, Erratum: Biophys Chem 106:91
15. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
16. Kostrowicki J, Piela L, Cherayil BJ, Scheraga HA (1991) Performance of the diffusion equation method in searches for optimum structures of clusters of Lennard-Jones atoms. J Phys Chem 95:4113–4119
17. Kostrowicki J, Scheraga HA (1992) Application of the diffusion equation method for global optimization to oligopeptides. J Phys Chem 96:7442–7449
18. Kubo R (1962) Generalized cumulant expansion method. J Phys Soc Japan 17:1100–1120
19. Lee J, Liwo A, Scheraga HA (1999) Energy-based *de-novo* protein folding by conformational space annealing and an off-lattice united-residue force field: application to the 10-55 fragment of staphylococcal protein A and to apo calbindin D9K. Proc Natl Acad Sci USA 96:2025–2030
20. Lee J, Scheraga HA, Rackovsky S (1997) New optimization method for conformational energy calculations on polypeptides: Conformational space annealing. J Comput Chem 18:1222–1232
21. Lee J, Scheraga HA (1999) Conformational space annealing by parallel computations: extensive conformational search of Met-enkephalin and of the 20-residue membrane-bound portion of melittin. Int J Quant Chem 75:255–265
22. Lee J, Scheraga HA, Rackovsky S (1997) New optimization method for conformational energy calculations on polypeptides: conformational space annealing. J Comput Chem 18:1222–1232
23. Lee J, Scheraga HA, Rackovsky S (1998) Conformational analysis of the 20-residue membrane-bound portion of melittin by conformational space annealing. Biopolymers 46:103–115
24. Lee J, Scheraga HA, Rackovsky S (1998) Conformational analysis of the 20-residue membrane-bound portion of melittin by conformational space annealing. Biopolymers 46:103–115
25. Levitt M, Chothia C (1976) Structural patterns in globular proteins. Nature 261:552–558
26. Li Z, Scheraga HA (1987) Monte Carlo-minimization approach to the multiple-minima problem in protein folding. Proc Natl Acad Sci USA 84:6611–6615
27. Li Z, Scheraga HA (1988) Structure and free energy of complex thermodynamic systems. J Molec Str (Theochem) 179:333–352
28. Liwo A, Arłukowicz P, Czaplewski C, Ołdziej S, Pillardy J, Scheraga HA (2002) A method for optimizing potential-energy functions by a hierarchical design of the potential-energy landscape: application to the UNRES force field. Proc Natl Acad Sci USA 99:1937–1942
29. Liwo A, Czaplewski C, Pillardy J, Scheraga HA (2001) Cumulant-based expressions for the multibody terms for the correlation between local and electrostatic interactions in the united-residue force field. J Chem Phys 115:2323–2347
30. Liwo A, Gibson KD, Scheraga HA, Brandt-Rauf PW, Monaco R, Pincus MR (1994) Comparison of the low energy conformations of an oncogenic and a non-oncogenic p21 protein, neither of which binds GTP or GDP. J Protein Chem 13:237–251
31. Liwo A, Kaźmierkiewicz R, Czaplewski C, Groth M, Ołdziej S, Wawak RJ, Rackovsky S, Pincus MR, Scheraga HA (1998) United-residue force field for off-lattice protein-structure simulations; III. Origin of backbone hydrogen-bonding cooperativity in united-residue potentials. J Comput Chem 19:259–276

32. Liwo A, Khalili M, Czaplewski C, Kalinowski S, Ołdziej S, Wachucik K, Scheraga HA (2007) Modification and optimization of the united-residue (UNRES) potential energy function for canonical simulations. I. Temperature dependence of the effective energy function and tests of the optimization method with single training proteins. J Phys Chem B 111:260–285
33. Liwo A, Khalili M, Scheraga HA (2005) Molecular dynamics with the united-residue (UNRES) model of polypeptide chains; test of the approach on model proteins. Proc Natl Acad Sci USA 102:2362–2367
34. Liwo A, Lee J, Ripoll DR, Pillardy J, Scheraga HA (1999) Protein structure prediction by global optimization of a potential energy function. Proc Natl Acad Sci USA 96:5482–5485
35. Liwo A, Ołdziej S, Czaplewski C, Kozłowska U, Scheraga HA (2004) Parameterization of backbone-electrostatic and multibody contributions to the UNRES force field for protein-structure prediction from ab initio energy surfaces of model systems. J Phys Chem B 108:9421–9438
36. Liwo A, Ołdziej S, Pincus MR, Wawak RJ, Rackovsky S, Scheraga HA (1997) A united-residue force field for off-lattice protein-structure simulations. I. Functional forms and parameters of long-range side-chain interaction potentials from protein crystal data. J Comput Chem 18:849–873
37. Liwo A, Pincus MR, Wawak RJ, Rackovsky S, Ołdziej S, Scheraga HA (1997) A united-residue force field for off-lattice protein-structure simulations. II: Parameterization of local interactions and determination of the weights of energy terms by Z-score optimization. J Comput Chem 18: 874–887
38. Liwo A, Pincus MR, Wawak RJ, Rackovsky S, Scheraga HA (1993) Prediction of protein conformation on the basis of a search for compact structures; test on avian pancreatic polypeptide. Protein Sci 2:1715–1731
39. Liwo A, Tempczyk A, Ołdziej S, Shenderovich MD, Hruby VJ, Talluri S, Ciarkowski J, Kasprzykowski F, Łankiewicz L, Grzonka Z (1996) Exploration of the conformational space of oxytocin and arginine-vasopressin using the electrostatically-driven Monte Carlo and molecular dynamics methods. Biopolymers 38:157–175
40. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:1087–1092
41. Miller MH, Némethy G, Scheraga HA (1980) Calculation of the structures of collagen models. Role of interchain interactions in determining the triple-helical coiled-coil conformation. 2. Poly(glycyl-prolyl-hydroxyprolyl). Macromolecules 13:470–478
42. Miller MH, Némethy G, Scheraga HA (1980) Calculation of the structures of collagen models. Role of interchain interactions in determining the triple-helical coiled- coil conformation. 3. Poly(glycyl-prolyl-alanyl). Macromolecules 13:910–913
43. Miller MH, Scheraga HA (1976) Calculation of the structures of collagen models. Role of interchain interactions in determining the triple-helical coiled-coil conformation. I. Poly(glycyl-prolyl-prolyl). J Polym Sci Polym Symposia 54:171–200
44. Momany FA, McGuire RF, Burgess AW, Scheraga HA (1975) Energy parameters in polypeptides. VII. Geometric parameters, partial atomic charges, non-bonded interactions, hydrogen bond interactions and intrinsic torsional potential for the naturally occurring amino-acids. J Phys Chem 79:2361–2381
45. Morales LB, Garduño Juárez RG, Romero D (1991) Applications of simulated annealing to the multiple-minima problem in small peptides. J Biomol Struct Dyn 8:721–735
46. Morales LB, Garduño Juárez RG, Romero D (1992) The multiple-minima problem in small peptides revisited. The Threshold Accepting approach. J Biomol Struct Dyn 9: 951–957
47. Nanias M, Czaplewski C, Scheraga HA (2006) Replica exchange and multicanonical algorithms with the coarse-grained united-residue (UNRES) force field. J Chem Theor Comput 2:513–528
48. Nayeem A, Vila J, Scheraga HA (1991) A comparative study of simulated-annealing and Monte Carlo-with-minimization approaches to the minimum-energy structures of polypeptides: Metenkephalin. J Comp Chem 12:595–605
49. Némethy G, Gibson KD, Palmer KA, Yoon CN, Paterlini G, Zagari A, Rumsey S, Scheraga H (1992) Energy parameters in polypeptides. 10. Improved geometrical parameters and nonbonded interactions for use in the ECEPP/3 algorithm, with application to proline-containing peptides. J Phys Chem 96:6472–6484
50. Némethy G, Pottle MS, Scheraga HA (1983) Energy parameters in polypeptides. 9. Updating of geometrical parameters, nonbonded interactions, and hydrogen bond interactions for the naturally occurring amino acids. J Phys Chem 87:1883–1887
51. Némethy G, Scheraga HA (1984) Hydrogen bonding involving the ornithine side chain of gramicidin S. Biochem Biophys Res Commun 118:643–647
52. Ołdziej S, Czaplewski C, Liwo A, Chinchio M, Nanias M, Vila JA, Khalili M, Arnautova YA, Jagielska A, Makowski M, Schafroth HD, Kaźmierkiewicz R, Ripoll DR, Pillardy J, Saunders JA, Kang Y-K, Gibson KD, Scheraga HA (2005) Physics-based protein-structure prediction using a hierarchical protocol based on the UNRES force field – test with CASP5 and CASP6 targets. Proc Natl Acad Sci USA 102:7547–7552
53. Ołdziej S, Kozłowska U, Liwo A, Scheraga HA (2003) Determination of the potentials of mean force for rotation about $C^{\alpha}\cdots C^{\alpha}$ virtual bonds in polypeptides from the ab initio energy surfaces of terminally-blocked glycine, alanine, and proline. J Phys Chem A 107:8035–8046
54. Ołdziej S, Łagiewka J, Liwo A, Czaplewski C, Chinchio M, Nanias M, Scheraga HA (2004) Optimization of the UNRES

force field by hierarchical design of the potential-energy landscape: III. Use of many proteins in optimization. J Phys Chem B 108:16950–16959
55. Olszewski KA, Piela L, Scheraga HA (1992) Mean-field theory as a tool for intramolecular conformational optimization. 1. Tests on terminally-blocked alanine and Metenkephalin. J Phys Chem 96:4672–4676
56. Olszewski KA, Piela L, Scheraga HA (1993) Mean field theory as a tool for intramolecular conformational optimization. 2. Tests on the homopolypeptides decaglycine and icosalanine. J Phys Chem 97:260–266
57. Paine GH, Scheraga HA (1985) Prediction of the native conformation of a polypeptide by a statistical-mechanical procedure. I. Backbone structure of enkephalin. Biopolymers 24:1391–1436
58. Paine GH, Scheraga HA (1986) Prediction of the native conformation of a polypeptide by a statistical-mechanical procedure. II. Average backbone structure of enkephalin. Biopolymers 25:1547–1563
59. Paine GH, Scheraga HA (1987) Prediction of the native conformation of a polypeptide by a statistical-mechanical procedure. III. Probable and average conformations of enkephalin. Biopolymers 26:1125–1162
60. Perutz MF (1978) Electrostatic effects in proteins. Science 201:1187–1191
61. Piela L, Kostrowicki J, Scheraga HA (1989) The multiple-minima problem in the conformational analysis of molecules. Deformation of the potential energy hypersurface by the diffusion equation method. J Phys Chem 93:3339–3346
62. Piela L, Scheraga HA (1987) On the multiple-minima problem in the conformational analysis of polypeptides. I. Backbone degrees of freedom for a perturbed α-helix. Biopolymers 26:S33–S58
63. Pillardy J, Arnautova YA, Czaplewski C, Gibson KD, Scheraga HA (2001) Conformation-family Monte Carlo: a new method for crystal structure prediction. Proc Natl Acad Sci USA 98:12351–12356
64. Pillardy J, Czaplewski C, Liwo A, Lee J, Ripoll DR, Kaźmierkiewicz R, Ołdziej S, Wedemeyer WJ, Gibson KD, Arnautova YA, Saunders J, Ye Y-J, Scheraga HA (2001) Recent improvements in prediction of protein structure by global ptimization of a potential energy function. Proc Natl Acad Sci USA 98:2329–2333
65. Pillardy J, Czaplewski C, Wedemeyer WJ, Scheraga HA (2000) Conformation-family Monte Carlo (CFMC): an efficient computational tool for identifying the low-energy states of a macromolecule. Helv Chim Acta 83:2214–2230
66. Pillardy J, Liwo A, Groth M, Scheraga HA (1999) An efficient deformation-based global optimization method for off-lattice polymer chains; self-consistent basin-to-deformed-basin mapping (SCBDBM). Application to united-residue polypeptide chains. J Phys Chem B 103:7353–7366
67. Pillardy J, Liwo A, Scheraga HA (1999) An efficient deformation-based global optimization method (Self-Consistent Basin-to-Deformed-Basin Mapping (SCBDBM)). Application to Lennard-Jones atomic clusters. J Phys Chem A 103:9370–9377
68. Pillardy J, Olszewski KA, Piela L (1992) Performance of the shift method of global minimization in searches for optimum structures of clusters of Lennard-Jones atoms. J Phys Chem 96:4337–4341
69. Pillardy J, Olszewski KA, Piela L (1992) Theoretically predicted lowest-energy structures of water clusters. J Mol Struct (Theochem) 270:277–285
70. Pillardy J, Piela L (1997) Smoothing techniques of global optimization. The distance scaling method in searches for the most stable Lennard-Jones atomic clusters. J Comp Chem 18:2040–2049
71. Pincus MR, Klausner RD, Scheraga HA (1982) Calculation of the three-dimensional structure of the membrane-bound portion of melittin from its amino acid sequence. Proc Natl Acad Sci USA 79:5107–5110
72. Pincus MR, Murphy RB, Carty RP, Chen J, Shah D, Scheraga HA (1988) Conformational analysis of possible biologically active (receptor-bound) conformations of peptides derived from cholecystokinin, cerulein and little gastrin and the opiate peptide, Metenkephalin. Peptides 9(1):145–152
73. Purisima EO, Scheraga HA (1987) An approach to the multiple-minima problem in protein folding by relaxing dimensionality. Tests on enkephalin. J Mol Biol 196:697–709
74. Rapaport DC, Scheraga HA (1981) Evolution and stability of polypeptide chain conformation: a simulation study. Macromolecules 14:1238–1246
75. Ripoll DR (1992) Conformational study of a peptide epitope shows large preferences for β-turn conformations. Int J Pept Protein Res 40:575–581
76. Ripoll DR, Piela L, Vásquez M, Scheraga HA (1991) On the multiple-minima problem in the conformational analysis of polypeptides. V. Application of the self-consistent electrostatic field and the electrostatically driven Monte Carlo methods to bovine pancreatic trypsin inhibitor. Proteins Struc Func Gen 10:188–198
77. Ripoll DR, Liwo A, Scheraga HA (1998) New developments of the electrostatically driven Monte Carlo method – Test on the membrane bound portion of melittin. Biopolymers 46:117–126
78. Ripoll DR, Scheraga HA (1988) On the multiple-minima problem in the conformational analysis of polypeptides. II. An electrostatically driven Monte Carlo method-tests on poly(L-alanine). Biopolymers 27:1283–1303
79. Ripoll DR, Scheraga HA (1989) The multiple-minima problem in the conformational analysis of polypeptides. III. An electrostatically driven Monte Carlo method; tests on enkephalin. J Protein Chem 8:263–287
80. Ripoll DR, Vásquez MJ, Scheraga HA (1991) The electrostatically driven Monte Carlo method: Application to conformational analysis of decaglycine. Biopolymers 31:319–330

81. Ripoll DR, Vila JA, Scheraga HA (2004) Folding of the villin headpiece subdomain from random structures. Analysis of the charge distribution as a function of the pH. J Mol Biol 339:915–925
82. Ripoll DR, Vila JA, Scheraga HA (2005) On the orientation of the backbone dipoles in native folds. Proc Natl Acad Sci USA 102:7559–7564
83. Saunders JA, Scheraga HA (2003) Ab initio structure prediction of two α-helical oligomers with a multiple-chain united-residue force field and global search. Biopolymers 68:300–317
84. Saunders JA, Scheraga HA (2003) Challenges in structure prediction of oligomeric proteins at the united-residue level: searching the multiple-chain energy landscape with CSA and CFMC procedures. Biopolymers 68:318–332
85. Scheraga HA (1974) Prediction of protein conformation. In: Anfinsen CB, Schechter AN (eds) Current Topics in Biochemistry, 1973. Academic Press, New York, pp 1–42
86. Scheraga HA (1983) Recent progress in the theoretical treatment of protein folding. Biopolymers 22:1–14
87. Scheraga HA, Liwo A, Ołdziej S, Czaplewski C, Pillardy J, Ripoll DR, Vila JA, Kaźmierkiewicz R, Saunders JA, Arnautova YA, Jagielska A, Chinchio M, Nanias M (2004) The protein folding problem: global optimization of force fields. Front Biosci 9:3296–3323
88. Simon I, Némethy G, Scheraga HA (1978) Conformational energy calculations of the effects of sequence variations on the conformations of two tetrapeptides. Macromolecules 11:797–804
89. Sippl MJ, Némethy G, Scheraga HA (1984) Intermolecular potentials from crystal data. 6. Determination of empirical potentials for O-H$\cdots$O-C hydrogen bonds from packing configurations. J Phys Chem 88:6231–6233
90. Vásquez M, Némethy G, Scheraga HA (1983) Computed conformational states of the 20 naturally occurring amino acid residues and of the prototype residue a-aminobutyric acid. Macromolecules 16:1043–1049
91. Vásquez M, Scheraga HA (1985) Use of buildup and energy-minimization procedures to compute low-energy structures of the backbone of enkephalin. Biopolymers 24:1437–1447
92. Vásquez M, Scheraga HA (1988) Calculation of protein conformation by the build-up procedure. Application to bovine pancreatic trypsin inhibitor using limited simulated nuclear magnetic resonance data. J Biomol Struct Dyn 5:705–755
93. Vásquez M, Scheraga HA (1988) Variable-target-function and build-up procedures for the calculation of protein conformation. Application to bovine pancreatic trypsin inhibitor using limited simulated nuclear magnetic resonance data. J Biomol Struct Dyn 5:757–784
94. Vila JA, Ripoll DR, Scheraga HA (2003) Atomically detailed folding simulation of the B domain of staphylococcal protein A from random structures. Proc Natl Acad Sci USA 100:14812–14816
95. Wada A (1976) The α-helix as an electric macro-dipole. Adv Biophys 9:1–63
96. Wawak RJ, Gibson KD, Liwo A, Scheraga HA (1996) Theoretical prediction of a crystal structures. Proc Natl Acad Sci USA 93:1743–1746
97. Wawak RJ, Pillardy J, Liwo A, Gibson KD, Scheraga HA (1998) The diffusion equation and distance scaling methods of global optimization; applications to crystal structure prediction. J Phys Chem 102:2904–2918
98. Zimmerman SS, Pottle MS, Némethy G, Scheraga HA (1977) Conformational analysis of the twenty naturally occurring amino acid residues using ECEPP. Macromolecules 10:1–9

Global Optimization: Tight Convex Underestimators

Chrysanthos E. Gounaris,
Christodoulos A. Floudas
Department of Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 49M37, 65K10, 90C26, 90C30, 46N10, 47N10

Article Outline

Keywords and Phrases
Introduction
Theoretical Results for Univariate Functions
Tightness of Univariate Underestimator
Extension to Multivariate Functions
Domain Rotation
Examples
References

Keywords and Phrases

Convex underestimators; αBB; Global optimization

Introduction

In their effort to locate the global solution, deterministic global optimization algorithms, like the αBB [1,2,6,14], employ a branch and bound framework. During this process, convex underestimation techniques are used to formulate relaxed convex problems that can be solved to optimality with the use of local solvers, thus providing valid lower bounds for the original problem. The tightness of the underestimators used is of fundamental

importance for the computational performance of these algorithms, since a tighter relaxation can lead to faster fathoming and less nodes of the branch and bound tree to be visited [7]. A recent review article on deterministic global optimization approaches can be found in [8].

In the case of arbitrary nonconvex functions that do not exhibit an exploitable mathematical structure, the αBB general underestimator [3,6] can be used:

$$L(x) = f(x) - \sum_{v=1}^{V} \alpha_v (x_v - x_v^L)(x_v^U - x_v) \; . \tag{1}$$

Originally introduced in [14], this underestimator derives from the function by subtracting a positive quadratic ($\alpha_v \geq 0 \forall v$). Given sufficiently large values of the α_v parameters, all nonconvexities in the original function $f(x)$ can be overpowered, resulting into a convex underestimator $L(x)$ that is valid for the entire domain $[x^L, x^U]$. A number of rigorous methods have been devised in order to select appropriate values for these parameters [2,3,13]. Extensive computational testing of the algorithm [1] showed that the most efficient of those methods is the one based on the scaled Gherschgorin theorem. According to this method, it suffices to select:

$$\alpha_v = \max \left\{ 0, -\frac{1}{2} \left(\underline{h_{vv}} - \sum_{\substack{u=1 \\ u \neq v}}^{V} \max \left\{ |\underline{h_{vu}}|, |\overline{h_{vu}}| \right\} \frac{(x_u^U - x_u^L)}{(x_v^U - x_v^L)} \right) \right\} \tag{2}$$

where $\underline{h_{vu}}$ and $\overline{h_{vu}}$ are lower and upper bounds of $\partial^2 f / \partial x_v x_u$ that can be calculated by interval analysis.

One could use alternatively a new class of general purpose convex underestimators that has been developed by Akrotirianakis and Floudas [4,5]. These underestimators are derived in a similar fashion, by subtracting an exponential term from the original function, that is:

$$L_1(x) = f(x) - \sum_{v=1}^{V} \left(1 - e^{\gamma_v (x_v - x_v^L)} \right) \left(1 - e^{\gamma_v (x_v^U - x_v)} \right) \; . \tag{3}$$

An iterative systematic procedure is used to determine the values of the γ_v parameters so as the underestimating function to be convex. The procedure ensures also that the resulting underestimator $L_1(x)$ is tighter than $L(x)$, the one that results from the original method. Floudas and Kreinovich [9,10] have in fact shown that these two functional forms (original quadratic and exponential) are the only optimal ones, since they are the only ones to be shift-, sign- and scale-invariant.

Maranas and Floudas [14] showed that the maximum separation distance between the original function $f(x)$ and the underestimator $L(x)$ of (1) is a quadratic function of interval length. Because of this, as well as because of potentially less overestimation in the interval extension of the Hessian matrix elements h_{vu}, the underestimator would become tighter with shrinkage of the domain under consideration. This was firstly exploited in Meyer and Floudas [15], where a piecewise approach was utilized. The method proposed partitioning of the domain into many subdomains and construction of the corresponding αBB underestimator for each one of them. These underestimators, although not valid for the entire domain, are much tighter in their respective subdomains. A hyperplane is subsequently added to each one of these underestimators and is selected in such a way, so that the combination of all these convex *pieces* results into an overall convex underestimator that is continuous and smooth (C^1-continuity).

This entry describes the work of [11,12] on the development of tight convex underestimators. The construction of these underestimators is based on a piecewise application of the αBB underestimator, in a similar fashion with the p-αBB approach [15], but, instead of adding hyperplanes, we identify those supporting line segments that have to be combined with convex parts of the original underestimators so as to form a C^1-continuous convex underestimator that is valid for the overall domain under consideration. One can also consider only the lines defined by these linear segments, thus coming up with a piecewise linear underestimator that can easily be incorporated in the NLP relaxation as a set of linear constraints.

In their work, Gounaris and Floudas [12] also demonstrated how one can make use of the high quality results of the approach in the univariate case so as to extend its applicability to functions with a higher number of variables. This is achieved by proper projections of the multivariate αBB underestimators into select two-dimensional planes. Furthermore, since the method utilizes projections into lower-dimensional spaces, they

explored ways to recover some of the information lost in this process. In particular, they apply the method after having transformed the original problem in an orthonormal fashion. This leads to the construction of even tighter underestimators, through the accumulation of additional valid linear cuts in the relaxation.

Theoretical Results for Univariate Functions

Let $f(x)$ be a univariate function that needs to be underestimated in $D = [x^L, x^U]$. We select an integer $N > 1$ and partition the complete domain in N segments of equal length. Thus, the *i-th* subdomain would be defined as $D_i = [x^{i-1}, x^i]$, where: $x^i = x^L + \frac{i}{N}(x^U - x^L)$, $i = 0, 1, \ldots, N$.

For every subdomain D_i,$i = 1, 2, \ldots, N$, we construct the corresponding αBB underestimator:

$$P_i(x) = f(x) - \alpha^i (x - x^{i-1})(x^i - x)$$
$$\alpha^i = \max\left\{0, -\frac{1}{2}\underline{f''}_{(D_i)}\right\} \tag{4}$$

where $\underline{f''}_{(D_i)}$ is a lower bound of the second derivative that is valid for the entire subdomain D_i.

Note that although an underestimator $P_i\}(x)$ can be defined outside its respective subdomain, its convexity is only guaranteed for $x \in [x^{i-1}, x^i]$.

We define $P(x), x \in [x^L, x^U]$ to be the following branched function:

$$P(x) = P_i(x), \text{ if } \quad x^{i-1} \le x \le x^i \ . \tag{5}$$

Function $P(x)$ is a *piecewise* convex valid underestimator of $f(x)$. Since it is not convex, a convexification technique has to be employed. The proposed technique involves the identification of those supporting line segments that are required for an overall underestimator $U(x)$. The technique is based on two algorithms, called "*inner*" and "*outer*", which are described in detail in [11].

The underestimator $U(x)$ consists of the identified linear parts, as well as convex parts of the underestimators $P_i(x)$, therefore it is a C^1-continuous branched function. This might pose some computational complications if the lower bounding (relaxation) problem is to be solved by local optimization solvers that require C^2-continuity. In order to avoid this problem, one can take into account only the lines defined by the line segments. According to this alternative, we first identify the linear segments needed for the construction of underestimator $U(x)$, but we consider those as lines defined in $[x^L, x^U]$. Let there be K such lines denoted as $T_k(x), k = 1, 2, \ldots, K$ and arranged in order of ascending slope. If applicable, this set can be augmented with lines that are tangential to P_1 and P_N at the respective domain edges x^L and x^U.

Each of these lines T_k is a valid underestimator of function $f(x)$ across the whole domain. We define the function $V(x)$ to be the pointwise maximum of all these lines. $V(x)$ is convex, since it is the pointwise maximum of linear functions and it is obviously an underestimator, since it consists of pieces of other underestimators. At the expense of some tightness (in the regions where underestimator $U(x)$ consisted of convex parts), we now have a piecewise linear underestimator $V(x)$ that can be incorporated in the relaxation as a set of linear constraints. The whole lower bounding problem can now be formulated as a linear programming problem (LP).

Tightness of Univariate Underestimator

It is apparent that as the level of partitioning increases, the underestimator $P(x)$ comes closer to the function, and therefore convex underestimators $U(x)$ and $V(x)$ approach the convex envelope of $f(x)$. Gounaris and Floudas [11] proved the following two theorems that are relevant with the tightness of the resulting underestimators in the univariate case:

Theorem 1. *There is some finite partitioning level N, for which the convex underestimator U(x) is the convex envelope of function f(x).*

Theorem 2. *There is some finite partitioning level N, for which underestimator V(x) is ϵ-close to underestimator U(x), that is:*

$$\max_{x \in D}\{U(x) - V(x)\} < \epsilon \tag{6}$$

where: $\epsilon > 0$ is an arbitrarily small constant.

Since these univariate underestimators are very tight, the remaining question is whether we can exploit them so as to construct underestimators of functions in higher dimensions. Gounaris and Floudas [12] presented some extensions of the method for application on multivariate functions that involve dimension reduction of the problem through proper projections into

lower-dimensional spaces. These extensions are described in Sect. "Extension to Multivariate Functions".

Extension to Multivariate Functions

Let $f(x)$ be a function of V variables that needs to be underestimated in a box domain $D = [x_1^L, x_1^U] \times \cdots \times [x_V^L, x_V^U]$. We choose integers $N_v > 1, v = 1, 2, \ldots, V$ and partition each range $[x_v^L, x_v^U]$ in N_v segments of equal length. Thus, the *j-th* segment of the v^{th} set would be defined as $[x_v^{j-1}, x_v^j]$, where: $x_v^j = x_v^L + \frac{j}{N_v}(x_v^U - x_v^L)$, $j = 0, 1, \ldots, N_v$. The complete V-dimensional domain D has now been partitioned into $N = \prod_{v=1}^{V} N_v$ box subdomains of equal measures. Let D_i be such a V-dimensional subdomain. It is uniquely defined by a set of indices $i_v, 1 \le i_v \le N_v, \forall v = 1, 2, \ldots, V$. Thus, the i^{th} subdomain would be defined as $D_i = [x_1^{i_1-1}, x_1^{i_1}] \times \cdots \times [x_V^{i_V-1}, x_V^{i_V}]$.

For every subdomain D_i, $i = 1, 2, \ldots, N$, we construct the corresponding αBB underestimator [1,2,3,6,14]:

$$P_i(x) = f(x) - \sum_{v=1}^{V} \alpha_v^i (x_v - x_v^{i_v-1})(x_v^{i_v} - x_v)$$

$$\alpha_v^i = \max\left\{0, -\frac{1}{2}\left(\underline{h_{vv}^{(i)}} - \sum_{\substack{u=1 \\ u \neq v}}^{V} \max\left\{|\underline{h_{vu}^{(i)}}|, |\overline{h_{vu}^{(i)}}|\right\} \frac{(x_u^{i_u} - x_u^{i_u-1})}{(x_v^{i_v} - x_v^{i_v-1})}\right)\right\} \quad (7)$$

where $\underline{h_{vu}^{(i)}}$ and $\overline{h_{vu}^{(i)}}$ are respectively lower and upper bounds of $\partial^2 f / \partial x_v x_u$ that are valid for the entire subdomain D_i.

Note that although an underestimator $P_i(x)$ can be defined outside its respective subdomain, its convexity is only guaranteed for $x \in D_i$.

We select variable $w, 1 \le w \le V$, which we designate to be the *active* variable, and enumerate all $M_w = N/N_w$ permutations of indices i_v, $v \neq w$. Every such permutation m, $1 \le m \le M_w$, corresponds to a subdomain $D_{wm} = [x_w^L, x_w^U] \times \prod_{\substack{v=1 \\ v \neq w}}^{V} [x_v^{i_v-1}, x_v^{i_v}]$, which can be further divided into N_w subdomains $D_{wmj} = [x_w^{j-1}, x_w^j] \times \prod_{\substack{v=1 \\ v \neq w}}^{V} [x_v^{i_v-1}, x_v^{i_v}]$, $j = 1, 2, \ldots, N_w$. These subdomains, belong to the set of the original subdomains D_i (for $i_w = j$) and therefore each one has an underestimator $P_{wmj}(x)$ associated with it, that is:

$$P_{wmj}(x) = f(x) - \alpha_w^i (x_w - x_w^{j-1})(x_w^j - x_w) - \sum_{\substack{v=1 \\ v \neq w}}^{V} \alpha_v^i (x_v - x_v^{i_v-1})(x_v^{i_v} - x_v) \quad (8)$$

where index i satisfies $D_i = D_{wmj}$ and parameters $\alpha_v^i, v = 1, 2, \ldots, V$ are calculated according to (7).

For every such subdomain $D_{wmj}, j = 1, 2, \ldots, N_w$, we define the following univariate function:

$$G_{wmj}(x_w) = \min_{\substack{x_v \\ \forall v \neq w}} P_{wmj}(x), \quad x_w^{j-1} \le x_w \le x_w^j \ . \quad (9)$$

Since they correspond to the minimum of a convex function over a subset of its variables, these functions are convex. Furthermore, each one is defined over a different segment of $[x_w^L, x_w^U]$. Therefore, each one can be considered as a convex *piece* of an overall *piecewise* convex underestimator. The latter is fully suitable for application of the convex underestimation method for univariate functions which was described in the previous sections.

Let $V_{wm}(x_w)$ be the piecewise linear underestimator obtained by the univariate method, and let it be the pointwise maximum of K_{wm} associated lines, that is:

$$V_{wm}(x_w) = \max\{T_{wmk}(x_w), \forall k = 1, 2, \ldots, K_{wm}\}, \quad x_w^L \le x_w \le x_w^U \quad (10)$$

Without loss of generality, let us assume that the lines T_{wmk} are arranged in order of ascending slope, that is, $slope(T_{wm(k-1)}) < slope(T_{wmk}), k = 2, 3, \ldots, K_{wm}$, and that the set already includes the potential augmented tangents at the domain edges, designated earlier as T_0 and T_{K+1}.

Univariate underestimator $V_{wm}(x_w)$ could, in principle, be considered as a multivariate function that is dependent to only one variable, x_w, and defined over the whole multidimensional (dimension V) subdomain D_{wm}. That is:

$$V_{wm}(x_w) \to V_{wm}(x), \quad x \in D_{wm} \quad (11)$$

Function $V_{wm}(x)$ is piecewise affine and consists of segments of V-dimensional hyperplanes. Since these hyperplanes depend only on the w^{th} variable, they are

parallel to all standard basis vectors e_v with the exception of e_w (to which they are parallel only if the slope of the corresponding line T_{wmk} is zero). This function is a valid underestimator for the original function $f(x)$ across the whole subdomain D_{wm}.

Applying the aforementioned procedure for every permutation $m = 1, 2, \ldots, M_w$, we come up with a collection of such underestimating segments, each of which is a valid underestimator for the function $f(x)$ across a subset of its original domain D. In order to develop a convex underestimator that would be valid for the whole domain, we have to *combine* all these segments. Let $m = 0$ denote the combination of all permutations $m = 1, 2, \ldots, M_w$. This combination can be achieved back in the projection space, by computing the lower hull of the set of all underestimators $V_{wm}(x_w)$. In fact, one needs to consider only the vertex points of each underestimator $V_{wm}(x_k)$ (that is the points of intersection between two lines $T_{wm(k-1)}$ and T_{wmk}), as well as their end points $\left(x_w^L, T_{wm1}(x_w^L)\right)$ and $\left(x_w^U, T_{wm(K_{wm})}(x_w^U)\right)$. Any standard 2d convex hull algorithm (e. g., *Graham-Scan*) can be used for this purpose. The lower hull is a convex piecewise linear function $V_{w0}(x_w)$, and it is the pointwise maximum of K_{w0} lines, that is:

$$V_{w0}(x_w) = \max\left\{T_{w0k}(x_w), \forall k = 1, 2, \ldots, K_{w0}\right\}, \quad x_w^L \le x_w \le x_w^U . \tag{12}$$

By construction, this function is a convex underestimator of all pieces $G_{wmj}(x_w)$ for all permutations, that is:

$$V_{w0}(x_w) \le G_{wmj}(x_w), x_w \in [x_w^{j-1}, x_w^j], \quad \forall j = 1, 2, \ldots, N_w, \forall m = 1, 2, \ldots, M_w . \tag{13}$$

Therefore, function $V_{w0}(x_w)$, if considered as $V_{w0}(x)$, is a valid underestimator for function $f(x)$ across its whole original domain D.

For any selection of the active variable w, the method will yield a convex (piecewise affine) underestimator which would be valid for the whole domain of interest, D. However, the method can be independently applied for every variable being active (one at a time), leading to a collection of valid underestimators. The pointwise maximum of all these is itself a valid convex underestimator, and is tighter (or equally tight) to the original function than any of its predecessors. Thus, the resulting underestimator is:

$$V(x) = \max\left\{V_{w0}(x), \forall w = 1, 2, \ldots, V\right\}, x \in D . \tag{14}$$

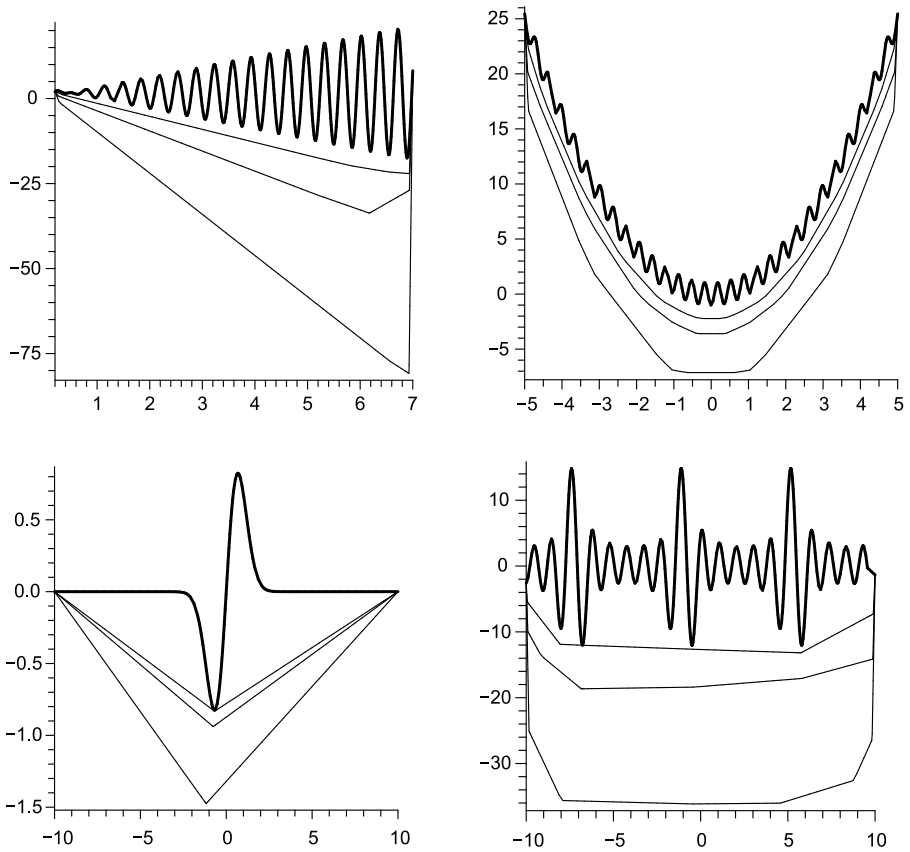
Note that the underestimator $V(x)$ is also piecewise hyperplanar, and can be represented in the problem relaxation as a set of linear constraints. Since we do not know explicitly which hyperplanes $T_{w0k}(x_w) \to T_{w0k}(x), k = 1, 2, \ldots, K_{w0}, w = 1, 2, \ldots, V$ contribute some part of theirs to the overall underestimator $V(x)$, all of them should be included in this relaxation, despite the fact that some may end up being redundant.

Since our method produces piecewise affine underestimators $L \equiv V$, the resulting convex relaxation is just a linear programming problem (LP), which takes the form of (15).

$$\begin{aligned} &\min_{\mu, x} \mu \\ &s.t. \quad \mu \ge T_{w0k}^{(0)}(x_w) \left\{ \begin{array}{l} \forall k = 1, 2, \ldots, K_{w0}^{(0)} \\ \forall w = 1, 2, \ldots, V \end{array} \right\} \\ &T_{w0k}^{(q)}(x_w) \le 0 \left\{ \begin{array}{l} \forall k = 1, 2, \ldots, K_{w0}^{(q)} \\ \forall w = 1, 2, \ldots, V \\ \forall q = 1, 2, \ldots, Q \end{array} \right\} \end{aligned} \tag{15}$$

Domain Rotation

The methodology presented in Section 4 involves the minimization of underestimators $P_{wmj}(x)$, over all their variables with the exception of one, variable x_w, which is designated as "active". Whenever such a projection into spaces of lower dimensionality is involved, there is the possibility that some useful information is lost. Some of this lost information will be recovered if we opt to apply the methodology for every variable being "active", one at a time, which basically calls for projecting into V different two-dimensional planes, each one being parallel to a different basis vector $e_v, v = 1, 2, \ldots, V$. However, since there is a finite number of variables in our problem, there is a limited number of planes to which we can project. If we want to enhance further the col-



Global Optimization: Tight Convex Underestimators, Figure 1
Univariate functions f_{1-4} with underestimators $V(x)$ for three different partitioning levels ($N = 24$, 36 and 48)

lection of underestimators that we will eventually accumulate in the relaxation (thus improve our chances for better tightness/lower bound), we will have to project into additional planes, that do not correspond to some variable that is "natural" to the problem, rather than to some linear combination of theirs.

This can be achieved by applying an orthonormal transformation to the problem's variable space, that is:

$$x \to x' = R \cdot x \ . \tag{16}$$

This transformation has to be orthogonal, which means that it should preserve the lengths of vectors and the angles between vectors. Furthermore, it should be an orientation-preserving transformation. A $V \times V$ matrix R that could provide such a transformation is called a *rotation matrix* and has to be a member of the special orthogonal group, that is:

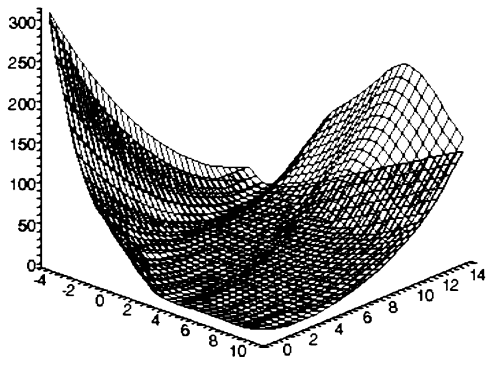
$$R \in SO(V) \Leftrightarrow \begin{cases} R^{-1} = R^T \\ |R| = +1 \ . \end{cases} \tag{17}$$

In their work, Gounaris and Floudas [12] discuss the selection of a suitable such matrix. They rigorously address the issue of selecting a suitable "rotated" domain and some suitable level of partitioning, and they also present a method to calculate appropriate values for the α parameters in the transformed counterpart of the problem.

Examples

Figure 1 depicts the plots for four nonlinear univariate functions. In particular, for functions: $f_1(x) = (3x - 1.4)sin(18x) + 1.7$, $f_2(x) = x^2 - cos(18x)$, $f_3(x) = (x + sinx)e^{-x}$ and $f_4(x) = -\sum_{k=1}^{5} ksin[(k+1)x + k]$.

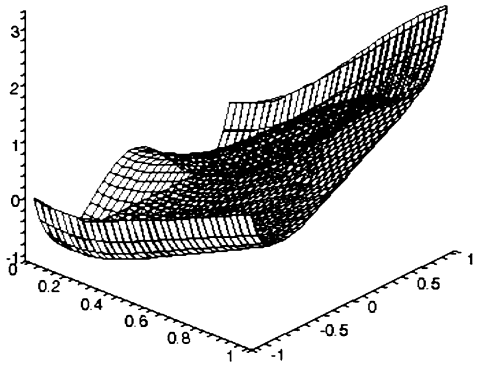
$$f(x_1,x_2)=\left(x_2-\frac{5.1x_1^2}{4\pi^2}+\frac{5x_1}{\pi}-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$$



$N = (32 \times 32) \quad \Delta\varphi = \pi / 8$

Total Linear Cuts = 162
Global minimum = 0.398
Lower Bound = 0.316
αBB Lower Bound = - 884

$$f(x_1,x_2)=4x_1^2-2.1x_1^4+\frac{x_1^6}{3}+x_1x_2-4x_2^2+4x_2^4$$

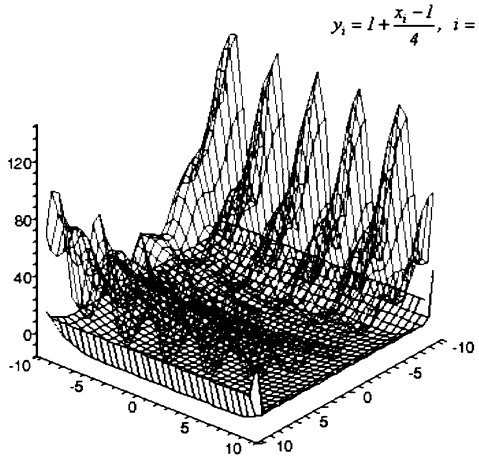


$N = (32 \times 32) \quad \Delta\varphi = \pi / 16$

Total Linear Cuts = 309
Global minimum = -1.03163
Lower Bound = -1.03164
αBB Lower Bound = - 6.04

$$f(x_1,x_2)=\frac{\pi}{2}\left\{10\sin^2(\pi y_1)+(y_1-1)^2\left[1+10\sin^2(\pi y_2)\right]+(y_2-1)^2\right\}$$

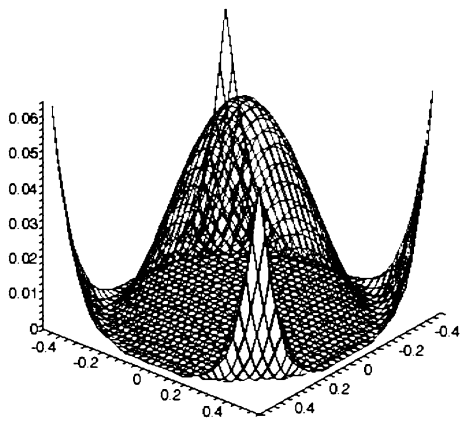
$$y_i=1+\frac{x_i-1}{4},\quad i=1,2$$



$N = (8 \times 24) \quad \Delta\varphi = \pi / 8$

Total Linear Cuts = 49
Global minimum = 0
Lower Bound = - 14
αBB Lower Bound = - 8441

$$f(x_1,x_2)=10^5(x_1-1)^2+10^5(x_2-1)^2+\left(x_1^2-x_2^2-\frac{1}{4}\right)^2$$



$N = (32 \times 32) \quad \Delta\varphi = \pi / 8$

Total Linear Cuts = 191
Global minimum = 8 x 10⁻⁶
Lower Bound = -7 x 10⁻⁶
αBB Lower Bound = - 0.69

Global Optimization: Tight Convex Underestimators, Figure 2
Piecewise planar underestimators of bivariate functions

The underestimators presented correspond to partitioning in $N = 24, 36$ and 48 subdomains (increasing tightness).

Figure 2 depicts plots for four nonlinear bivariate functions. For each case, $N_1 \times N_2$ is the level of partitioning used and $\Delta\varphi$ is the resolution of domain

rotation. Some additional information regarding the improvement of lower bound, as well as the number of linear cuts that have to be accumulated in the relaxation, is also included.

References

1. Adjiman CS, Androulakis IP, Floudas CA (1998) A Global Optimization Method, αBB, for General Twice-Differentiable Constrained NLPs II Implementation and Computational Results. Comput Chem Eng 22:1159–1179
2. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A Global Optimization Method, αBB, for General Twice-Differentiable Constrained NLPs I Theoretical Advances. Comput Chem Eng 22:1137–1158
3. Adjiman CS, Floudas CA (1996) Rigorous Convex Underestimators for General Twice-Differentiable Problems. J Global Optim 9:23–40
4. Akrotirianakis IG, Floudas CA (2004) A New Class of Improved Convex Underestimators for Twice Continuously Differentiable Constrained NLPs. J Global Optim 30:367–390
5. Akrotirianakis IG, Floudas CA (2004) Computational Experience with a New Class of Convex Underestimators : Box-Constrained NLP Problems. J Global Optim 29:249–264
6. Androulakis IP, Maranas CD, Floudas CA (1995) αBB: A Global Optimization Method for General Constrained Nonconvex Problems. J Global Optim 7:337–363
7. Floudas CA (2000) Deterministic Global Optimization: Theory, Algorithms and Applications. Kluwer
8. Floudas CA, Akrotirianakis IG, Caratzoulas S, Meyer CA, Kallrath J (2005) Global Optimization in the 21st Century: Advances and Challenges. Comput Chem Eng 29:1185–1202
9. Floudas CA, Kreinovich V (2007) Towards Optimal Techniques for Solving Global Optimization Problems: Symmetry-Based Approach. In: Torn A, Zilinskas J (eds) Models and Algorithms for Global Optimization. Springer, pp 21–42, ISBN 978-0-387-36720-0
10. Floudas CA, Kreinovich V (2007) On the Functional Form of Convex Underestimators for Twice Continuously Differentiable Functions. Optim Lett 1:187–192
11. Gounaris CE, Floudas CA (2008) Tight Convex Underestimators for C^2-Continuous Functions: I. Univariate Functions. J Global Optim, in press
12. Gounaris CE, Floudas CA (2008) Tight Convex Underestimators for C^2-Continuous Functions: II. Multivariate Functions. J Global Optim, in press
13. Hertz D, Adjiman CS, Floudas CA (1999) Two results on bounding the roots of interval polynomials. Comput Chemical Eng 23:1333–1339
14. Maranas CD, Floudas CA (1994) Global Minimum Potential Energy Conformations of Small Molecules. J Global Optim 4:135–170
15. Meyer CA, Floudas CA (2005) Convex Underestimation of Twice Continuously Differentiable Functions by Piecewise Quadratic Perturbation : Spline αBB Underestimators. J Global Optim 32:221–258

Global Optimization Using Space Filling

GOSF

ROMAN G. STRONGIN
Nizhni Novgorod State University,
Nizhni Novgorod, Russia

MSC2000: 90C26

Article Outline

Keywords
Approximation of SFC
Algorithm
Convergence Conditions
See also
References

Keywords

Constrained global optimization; Lipschitz optimization; Space filling curve; Peano curve; Index approach; Partial computability; E-reserved solution

A large number of decision problems in the world of applications may be formulated as searching for a *constrained global optimum* (minimum, for certainty)

$$\varphi^* = \varphi(y^*) = \min\{\varphi(y):\ y \in D,\ g_i(y) \le 0,\ 1 \le i \le m\},$$

where the *domain of search* (DS)

$$D = \left\{y \in \mathbf{R}^N:\ -2^{-1} \le y_j \le 2^{-1},\ 1 \le j \le N\right\},$$

$\mathbf{R}^N$ is the N-dimensional Euclidian space and the *objective function* $\varphi(y)$ (henceforth denoted $g_{m+1}(y)$) and the left-hand sides $g_i(y)$, $1 \le i \le m$, of the *constraints* are *Lipschitzian* (with respective constants L_i, $1 \le i \le m+1$) and may be multi-extremal.

If DS is set defined by the hyperparallelepiped

$$S = \left\{w \in \mathbf{R}^N:\ a_j \le w_j \le b_j,\ 1 \le j \le N\right\},$$

then, by introducing the transformation

$$y_j = \frac{w_j - (a_j + b_j)/2}{\rho},$$
$$\rho = \max\{b_j - a_j : \ 1 \le j \le N\},$$

and the extra constraint

$$g_0(y) = \max\left\{|y_j| - \frac{b_j - a_j}{2\rho} : \ 1 \le j \le N\right\} \le 0,$$

it is possible to keep up the initial presentation D for DS (which is assumed to be the standard one) not altering the relations of Lipschitzian properties in dimensions.

The assumption of the divided functions g_i, $0 \le i \le m+1$, differences being bounded by the respective constants L_i (Lipschitzian property), which may be interpreted as a mathematical description of a limited power of change in real systems, provides a basis for estimating φ^* and y^*; by exploring DS with finite number of trials depending on the desired accuracy of search. This Lipschitzian approach ([2,5,9,20]) requires, in general, substantially less trials than the plain *uniform grid technique* owing to the thorough selection of each subsequent trial with the account of all the previously computed functions' values.

Such a selection turns into solving some auxiliary multidimensional optimization problem (MOP) of increasing multi-extremality (along with the accumulation of trial outcomes) at each step of the search process. But the case $N = 1$ is effectively solvable and, therefore, it is of interest to present MOP by its one-dimensional equivalent.

A possible way to do so ([1,7,11,12,14,15,18]) is to employ single-valued *Peano curves* $y(x)$ continuously mapping the unit interval [0, 1] on the x-axis onto the hypercube D and, thus, yielding the equality

$$\varphi^* = \varphi(x^*) = \min\left\{\varphi(y(x)) : \begin{array}{c} x \in [0,1], \\ g_i(y(x)) \le 0, 0 \le i \le m \end{array}\right\}. \quad (1)$$

These curves, first introduced in [4,8], are 'filling' the cube, i. e. they pass through every point of D, and this gave rise to the term *space filling curves* (SFC); see survey [10].

The construction of SFC can be explained by following the scheme from [4]. Divide D into 2^N equal hypercubes of 'first-partition' by cutting D with the set of N mutually orthogonal hyperplanes (each plain is parallel to one of the coordinate ones and passes through the middle points of D edges orthogonal to this hyperplane). Then divide (in the above manner) each of the obtained first-partition cubes into 2^N second-partition cubes. Continuing this process, i. e. consequently cutting each cube of a current partition into 2^N cubes of the subsequent partition, yields hypercubes of any Mth partition with the edge-length equal 2^{-M}. The total number of cubes in the Mth partition is equal 2^{MN}.

Next, cut the interval [0, 1] into 2^N equal parts. Then, once again, cut each of these parts into 2^N smaller (equal) parts, etc. Designate $d(M, v)$ the subinterval of Mth partition, where v is the coordinate of the left endpoint of this interval. The length of $d(M, v)$ is equal 2^{-MN}. Assume that $v \in d(M, v)$, but the right endpoint of this subinterval (if it is not equal 1) does not belong to it.

Establish a mutually single-valued correspondence between all subintervals of any particular Mth partition and all subcubes of Mth partition. Henceforth, the notation $D(M, v)$ will stay for the subcube corresponding to the subinterval $d(M, v)$ and vice versa. Assume this correspondence to satisfy the following conditions:

- $D(M+1, v') \subset D(M, v'')$ if and only if $d(M+1, v') \subset d(M, v'')$.
- $d(M, v')$ and $d(M, v'')$ have a common endpoint (which is either v' or v'') if and only if $D(M, v')$ and $D(M, v'')$ have a common face (i. e. these subcubes are contiguous).

Now, a single-valued continuous map $y(x)$ is set by introducing the third requirement

- If $x \in d(M, v)$, then $y(x) \in D(M, v)$, for $M \ge 1$.

Note that for any integer $M \ge 1$ and any given $x \in [0, 1]$ there is just one subinterval meeting the condition $x \in d(M, v)$; the continuity is the consequence of the first two conditions.

Approximation of SFC

The center $y^c(x)$ of the subcube $D(M, v)$ containing $y(x)$ may be interpreted as an approximation to $y(x)$; the inequalities

$$\max\left\{\left|y_j^c(x) - y_j(x)\right| : \ 1 \le j \le N\right\} \le 2^{-(M+1)},$$
$$x \in [0, 1],$$

reflect the accuracy attainable for any particular preset value of M.

A constructive way to establishing the above correspondence is described and substantiated in [3,12,15,19] and, in short, can be presented as follows. Introduce the auxiliary hypercube

$$\Delta = \{y \in \mathbf{R}^N: \ -0.5 \le y_i \le 1.5, \ 1 \le i \le N\}$$

and designate $\Delta(s)$, $0 \le s \le 2^N - 1$, the subcubes of the first partition of Δ. The centers of $\Delta(s)$ (to be referred as $u(s)$) are N-dimensional binary vectors defined by the relations

$$\begin{aligned} &u_i(s) = (\beta_i + \beta_{i-1}) \mod 2, \\ &1 \le i < N, \quad u_N = \beta_{N-1}, \end{aligned} \tag{2}$$

where β_i, $0 \le i < N$, are digits in binary presentation of s:

$$s = \beta_{N-1}2^{N-1} + \cdots + \beta_0 2^0. \tag{3}$$

Owing to this numeration, any two centers $u(s)$ and $u(s + 1)$, $0 \le s < 2^N - 1$, have just one different coordinate, which means that the corresponding subcubes $\Delta(s)$ and $\Delta(s + 1)$ are contiguous.

Next, let the binary form of v in $d(M, v)$ be

$$0 \le v = \sum_{i=1}^{MN} \alpha_i 2^{-i} < 1.$$

Then the identity $d(M, v) = d(z_1, \ldots, z_M)$, where

$$z_j = \sum_{i=1}^{N} \alpha_{(j-1)N+i} 2^i, \quad 1 \le j \le M, \tag{4}$$

provides the possibility to interpret $d(z_1, \ldots, z_M)$, as the z_Mth subinterval of the interval $d(z_1, \ldots, z_{M-1})$ divided into 2^N equal parts (the numeration streams from left to right along the x-axis). Note that the above identity implies $D(M, v) = D(z_1, \ldots, z_M)$.

Now, mapping Δ onto D by the linear transformation and assuming that $D(z_1) = D(s)$ if $D(s)$ is the image of $\Delta(s)$, we obtain the numeration (in the first partition of D) satisfying the above conditions. Then by mapping Δ onto each subcube $D(z_1)$ of the first partition, we get the desired numeration in the second partition of D, where $D(z_1, z_2) = D(z_1, s)$ if $D(z_1, s)$ is the image of $\Delta(s)$, and so on. To ensure that $D(z_1, 2^N - 1)$ and $D(z_1 + 1, 0)$ would also have a common face (and, in general, the last subcube in the first partition of $D(z_1, \ldots, z_M)$ and the first subcube in the first partition of $D(z_1, \ldots, z_M + 1)$ would also be contiguous) we add some mechanism in the above numeration procedure to provide the necessary juxtapositioning.

Introduce the integer $l = l(z_1, \ldots, z_M)$ indicating the number of the only coordinate which has to be different for the center of the initial subcube $D(z_1, \ldots, z_M, 0)$ and the last subcube $D(z_1, \ldots, z_M, 2^N - 1)$ of the next partition of $D(z_1, \ldots, z_M)$ and the binary vector $w = w(z_1, \ldots, z_M)$ indicating the position of the center of the subcube $D(z_1, \ldots, z_M, 0)$. To do so we employ the integer function

$$l(s) = \begin{cases} 1 & \text{if } s = 0 \text{ or } s = 2^N - 1, \\ \min \{j: \ 2 \le j \le N, \ \beta_{j-1} = 1\}, & \\ \quad \text{otherwise}, & \end{cases} \tag{5}$$

where β_{j-1} is from (3), and the binary vector-function

$$w_i(s+1) = w_i(s) = \begin{cases} \overline{u}_i(s), & i = 1, \\ u_i(s), & 2 \le i \le N, \end{cases} \tag{6}$$

where s is supposed to be the odd number, $\overline{u}_i$ stays for logical negation of u_i, and $w(0) = u(0)$. The amended procedure for successive numeration in subsequent partitions includes the operations:

- permutation of u_N and u_t in $u(z_j)$ from (2) and of w_N and w_t in $w(z_j)$ from (6) with $t = l(z_{j-1})$, where z_{j-1} is from (4), $1 < j \le M$, and $l(z_{j-1})$ is from (5); $t = N$ if $j = 1$. New vectors are to be referred as $u^t(z_j)$ and $w^t(z_j)$;
- addition

$$\begin{aligned} u_i^{tq}(z_j) &= (u_i^t(z_j) + q_i) \mod 2, \ 1 \le i \le N, \\ w_i^{tq}(z_j) &= (w_i^t(z_j) + q_i) \mod 2, \ 1 \le i \le N, \end{aligned}$$

  where $q = w(z_{j-1})$, $1 < j \le M$, and $q = (0, \ldots, 0) \in \mathbf{R}^N$ if $j = 1$;
- transformation

$$l^t(z_j) = \begin{cases} N, & l(z_j) = t, \\ t, & l(z_j) = N, \\ l(z_j), & l(z_j) \ne N \text{ and } l(z_j) \ne t, \end{cases}$$

  where t is from the above permutation.

The successively computed values $u^{tq}(z_j)$, $w^{tq}(z_j)$, $l^t(z_j)$ are used instead of the initial values $u(z_j)$, $w(z_j)$, $l(z_j)$, $1 \le j \le M$, to obtain the approximation

$$y^c(x) = \sum_{j=1}^{M} (u^{tq}(z_j) - p)2^{-j}, \quad x \in d(M, v),$$

with $p = (2^{-1}, \ldots, 2^{-1}) \in \mathbf{R}^N$.

The important property of reducing dimensionality through SFC is that functions $g_i(y(x))$, $0 \le i \le m+1$, from (1) corresponding to Lipschitzian functions from the initial MOP satisfy the *uniform Hölder conditions* ([7,11,15,19])

$$\left|g_i(y(x')) - g_i(y(x''))\right| \le K_i(\left|x' - x''\right|)^{\frac{1}{N}},$$
$$x', x'' \in [0, 1],$$

with respective coefficients $K_i = 4L_i\sqrt{N}$, $0 \le i \le m+1$.

Problem (1) can further be reduced to an unconstrained case by employing the *index approach* (IA) ([7,13,17,18]) which makes no use of *penalties* and, thus, does not require any adjustments of penalty coefficients. Within IA functions $g_i(y(x))$ from (1) may not be defined throughout [0, 1]; they have to be computable only at the points $x \in [0, 1]$ meeting the conditions $g_k(y(x)) \le 0$, $1 \le k < i$ (this property is to be referred as *partial computability* of problem functionals). Therefore, within IA the *outcome* of each trial is given by a dyad

$$f(x) = g_\nu(y(x)), \quad \nu = \nu(x) = \nu(y(x)), \tag{7}$$

where ν is the number of the first constraint violated at the point x; this number is to be referred as the *index* of the corresponding point.

The unconstrained equivalent of (1) is

$$\psi(x^*) = \min\{\psi(x) : \ x \in [0, 1]\},$$

where

$$\psi(x) = \frac{g_\nu(y(x))}{K_\nu} - \begin{cases} 0, & \nu = \nu(x) \le m, \\ \frac{\varphi^*}{K_\nu}, & \nu = \nu(x) = m+1, \end{cases}$$

and x^* is a solution to (1). The algorithm presented below solves (1) by minimizing $\psi(x)$. It substitutes the unknown values φ^* and K_i, $0 \le i \le m+1$, by their running estimates; it also surmounts the discontinuity inherent to $\psi(x)$.

Algorithm

The first trial is to be executed at an arbitrary interior point $x^1 \in (0, 1)$. The choice of any subsequent point x^{k+1}, $k \ge 1$, is due to the rules:

1) Renumber the points $x^1, \ldots, x^k$ of the previous trials by subscripts in the increasing order of the coordinate, i. e.

$$0 = x_0 < \cdots < x_k < x_{k+1} = 1,$$

and associate them with the computed values $z_i = f(x_i)$, $1 \le i \le k$, from (7); values z_0 and z_{k+1} are undefined.

2) Collect in the sets

$$I_\nu = \{i : \ 1 \le i \le k, \ \nu = \nu(x_i)\},$$
$$0 \le \nu \le m+1,$$

all subscripts corresponding to the points with equal indices; it is assumed that $\nu(x_0) = \nu(x_{k+1}) = -1$ and $I_{-1} = \{0, k+1\}$.

3) Construct the unions

$$S_\nu = I_{-1} \cup \cdots \cup I_{\nu-1}, \quad 0 \le \nu \le m+1,$$

and

$$T_\nu = I_{\nu+1} \cup \cdots \cup I_{m+1} \cup I_{m+2},$$
$$0 \le \nu \le m+1,$$

of subscripts corresponding to the trial points with the indices less than ν and exceeding ν respectively; $I_{m+2} = \emptyset$ by the definition.

4) Compute the running lower bounds

$$\mu_\nu = \max \left\{ \frac{|z_j - z_i|}{(x_j - x_i)^{\frac{1}{N}}} : \ \begin{matrix} i, j \in I_\nu, \\ i < j \end{matrix} \right\} \tag{8}$$

for respective Hölder coefficients of the functions $g_\nu(y(x))$, $0 \le \nu \le m+1$. If I_ν contains less than two elements or if μ_ν from (8) is equal zero, assume that $\mu_\nu = 1$.

5) Find the values

$$z_\nu^* = \begin{cases} -\varepsilon_\nu, & T_\nu \ne \emptyset, \\ \min\{z_i : \ i \in I_\nu\}, & T_\nu = \emptyset, \end{cases}$$

for all nonempty sets I_ν, $0 \le \nu \le m+1$; vector $\varepsilon = (\varepsilon_0, \ldots, \varepsilon_m)$ is the input of the algorithm.

6) Compute characteristics $R(i)$, $1 \le i \le k+1$, where

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{r^2\mu_\nu^2\Delta_i} - \frac{2(z_i + z_{i-1} - 2z_\nu^*)}{r\mu_\nu},$$
$$i-1, \quad i \in I_\nu,$$
$$R(i) = 2\Delta_i - \frac{4(z_i - z_\nu^*)}{r\mu_\nu},$$
$$i \in I_\nu, \quad i-1 \in S_\nu,$$
$$R(i) = 2\Delta_i - \frac{4(z_{i-1} - z_\nu^*)}{r\mu_\nu},$$
$$i-1 \in I_\nu, \quad i \in S_\nu,$$
$$\Delta_i = (x_i - x_{i-1})^{\frac{1}{N}}.$$

Proper choice of the parameter $r > 1$ allows to use the product $r\mu_\nu$ as an upper bound for K_ν.

7) Select integer t from

$$R(t) = \max\{R(i):\ 1 \le i \le k+1\}$$

and execute the subsequent trial at the point

$$x^{k+1} = \frac{x_t + x_{t-1}}{2} - \operatorname{sign}(z_t - z_{t-1})\left[\frac{|z_t - z_{t-1}|}{\mu_\nu}\right]^N \cdot \frac{1}{2r}$$

if $\nu(x_t) = \nu(x_{t-1})$; otherwise, i. e. if $\nu(x_{t-1}) \ne \nu(x_t)$, the second term is omitted.

The concept of *ε-reserved solution* $y\varepsilon$, where

$$\varphi(y_\varepsilon) = \min\left\{\varphi(y):\ \begin{matrix} y \in D, \\ g_i(y) \le -\varepsilon_i, \\ 0 \le i \le m \end{matrix}\right\}$$

and $\varepsilon_i > 0$, $0 \le i \le m$, provides interpretation for ε from Step 5). The sequence of points $\{x^k\}$ selected by the Steps 1)–7) in the interval $[0, 1]$ generates the corresponding sequence $\{y^k\} = \{y(x^k)\}$ in D.

Convergence Conditions

([15,16,17] [18]). Assume that the following is true:

- the problem (1) has an ε-reserved solution;
- functions $g_i(y)$, $0 \le i \le m+1$, admit Lipschitzian continuations throughout D;
- from some Step onwards, the values μ_ν, $0 \le \nu \le m+1$, from (8) satisfy the inequalities

$$r\mu_\nu > 16L_\nu\sqrt{N}, \quad 0 \le \nu \le m+1.$$

Then any limit point $\overline{y}$ of the sequence $\{y^k\}$ generated by the above algorithm satisfies the conditions:

$$\varphi(\overline{y}) = \inf\left\{\varphi(y^k):\ \begin{matrix} k \in N_1, \\ g_i(y^k) \le 0, \\ 0 \le i \le m \end{matrix}\right\} \le \varphi(y_\varepsilon),$$

where N_1 is the set of positive integers.

As long as in applications SFC $y(x)$ is to be approximated by $y^c(x)$ corresponding to some Mth partition, it is important to notice that the substantiation of the above convergence conditions implies the relation

$$2^{-M} \ll \frac{1}{\sqrt{N}} \min_{0 \le \nu \le m}\left(\frac{\varepsilon_\nu}{L_\nu}\right),$$

which means that the existence of an ε-reserved solution may be interpreted as some kind of the *regularity conditions* (cf. [6]).

Dimensionality reduction through SFC causes some loss of the information on the closeness of trial points is the initial multidimensional space. Two close points in D may have substantially nonclose pre-images in $[0, 1]$. To overcome this obstacle, it is possible either to store all pre-images of each trial point (close points in D always have some close pre-images; see [12]) or to use some sets of shifted SFC to provide the better transfer of metric information (see [17]).

GOSF based on the reduction to one dimension by using SFC and on the reduction to unconstrained problems by employing IA admits effective parallelization (see [16,19]).

See also

- ▶ αBB Algorithm
- ▶ Continuous Global Optimization: Applications
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Differential Equations and Global Optimization
- ▶ DIRECT Global Optimization Algorithm
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Binary Star Astronomy
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Topology of Global Optimization

References

1. Butz AR (1968) Space filling curves and mathematical programming. Inform Control 12(4):313–330
2. Evtushenko YuG (1985) Numerical optimization techniques. Transl Ser Math and Engin Optim. Software, New York
3. Gergel VP, Strongin LG, Strongin RG (1988) Neighbourhood method in recognition problems. Soviet J Comput Syst Sci 26(2):46–54
4. Hilbert D (1891) Über die steitige Abbildung einer Linie auf ein Flächenstück. Math Ann 38:459–460
5. Horst R, Pardalos PM (1995) Handbook of global optimization. Kluwer, Dordrecht
6. Kuhn HW, Tucker AW (1951) Nonlinear programming. Proc. Second Berkeley Symp. Math. Statistics and Probability, Univ. Calif. Press, Berkeley, pp 481–492
7. Markin DL, Strongin RG (1988) A method for solving multiextremal problems with non-convex constraints, that uses a priori information about estimates of the optimum. USSR J Comput Math Math Phys 27(1):33–39
8. Peano G (1890) Sur une courbe, qui remplit toute une aire plane. Math Ann 36:157–160
9. Pinter J (1996) Global optimization in action (continuous and Lipschitz optimization: Algorithms, implementations and applications). Kluwer, Dordrecht
10. Sagan H (1994) Space-filling curves. Springer, Berlin
11. Strongin RG (1973) On the convergence of an algorithm for finding a global extremum. Eng Cybernetics 11(4):549–555
12. Strongin RG Numerical methods in multi-extremal problems (Information-statistical algorithms), Nauka, Moscow. (In Russian)
13. Strongin RG (1985) Numerical methods for multiextremal nonlinear programming problems with nonconvex constraints. Lecture Notes Economics and Math Systems, 255:278–282
14. Strongin RG (1989) The information approach to multiextremal optimization problems. Stochastics and Stochastic Reports 27:65–82
15. Strongin RG (1990) Search for global optimum. Znanie, Moscow
16. Strongin RG (1991) Parallel multi-extremal optimization using a set of evolvents. USSR J Comput Math Math Phys 31(8):1173–1185. (In Russian)
17. Strongin RG (1992) Algorithms for multi-extremal mathematical programming problems employing the set of joint space-filling curves. J Global Optim 2:357–378
18. Strongin RG, Markin DL (1986) Minimization of multiextremal functions with nonconvex constraints. Cybernetics 22(4):486–493
19. Strongin RG, Sergeyev YAD (1992) Global multidimensional optimization on parallel computer. Parallel Comput 18:1259–1273
20. Sukharev AG (1989) Minimax algorithms in problems of numerical analysis. Nauka, Moscow

Global Optimization in Weber's Problem with Attraction and Repulsion

COSTAS D. MARANAS
Pennsylvania State University, University Park, USA

MSC2000: 90C26, 90C90

Article Outline

Keywords
See also
References

Keywords

Weber problem; Facility location; Global optimization

Weber's problem and all its variations with positive weights is clearly one of the most extensively studied problems in the area of continuous location theory. It frequently arises in planning situations where a single central facility must be located so as to minimize the total cost associated with serving a number of demand centers. In all these cases, the underlying assumption, that the associated service costs are directly proportional to the Euclidean distance of the demand center from the central facility, has been adopted.

Weber's problem with attraction and repulsion can be stated as follows: Given a number of 'attractive' or 'repulsive' points located on a $2D$-plane, find the position of a single facility inside an arbitrary region P such that the sum of the weighted distances of all points from the single facility is at its global minimum.

This problem can be formulated as the following nonlinear optimization problem:

$$\min_{(x,y)\in P} \sum_{i\in I^+} w_i \sqrt{(x-x_i)^2+(y-y_i)^2} - \sum_{i\in I^-} w_i \sqrt{(x-x_i)^2+(y-y_i)^2},$$

where I^+, I^- are the sets of attractive (users) and repulsive (residents) points, respectively; w_i, $i \in I^+$ the positive weight of the ith attractive point and $-w_i$, $i \in I^-$ the negative weight of the ith repulsive point; (x_i, y_i) are the coordinates of the ith attractive or repulsive point;

and P is the region where where the single facility must be situated.

The unconstrained version of this problem has been shown to involve a number of important properties. The first property provides a sufficient condition for having finite solutions or solutions at infinity. Z. Drezner and G.O. Wesolowsky [6] by using the well-known triangle inequality relation proved the following:

Property 1 For the unconstrained problem, if $W > 0$ then the global optimum location is finite; if $W < 0$ then the global optimum location is at infinity, where

$$W = \sum_{i \in I^+} w_i - \sum_{i \in I^-} w_i.$$

The second property deals with the localization of all local minimum solutions. Let R be the radius of the smallest circle enclosing all points. The square of this radius R can be obtained through the solution of the following nonlinear optimization problem:

$$\begin{cases} \min\limits_{x^c, y^c, R^2} & R^2 \\ \text{s.t.} & (x^c - x_i)^2 + (y^c - y_i)^2 \le R^2, \\ & \forall i \in I^+ \cup I^-, \end{cases}$$

which is convex in the combined space of the coordinates of the center of the circle (x^c, y^c) and the square of the radius of the circle R^2 enclosing all points. Drezner and Wesolowsky [6] proved the following localization property, which generalizes the *majority theorem* [24] for Weber's problem.

Property 2 For the unconstrained problem, all local minima and therefore the global minimum are inside a disc with a radius equal to

$$\rho = \frac{R}{\sqrt{1 - \alpha^2}}$$

where

$$\alpha = \frac{W^-}{W^+},$$
$$W^+ = \sum_{i \in I^+} w_i, \quad W^- = \sum_{i \in I^-} w_i.$$

Note that the boundary of this disc is attainable.

The case $\alpha = 1$ or, equivalently, $W = 0$ is accounted for by finding the optimal solution at infinity and comparing it with the best finite solution. Drezner and Wesolowsky [6] by using asymptotic analysis showed the following:

Property 3 For the unconstrained problem, if $W = 0$ the best solution at infinity is $-(A^2 + B^2)^{1/2}$ where:

$$A = \sum_{i \in I^+} w_i x_i - \sum_{i \in I^-} w_i x_i,$$
$$B = \sum_{i \in I^+} w_i y_i - \sum_{i \in I^-} w_i y_i.$$

The following property examines whether a demand point corresponds to a local minimum [6].

Property 4 For the unconstrained problem, if there is a point i such that

$$w_i - (W_x + W_y)^{1/2} \begin{cases} > 0, & \text{then point } i \text{ is a local minimum,} \\ < 0, & \text{then point } i \text{ is not a local minimum,} \\ = 0, & \text{then both possibilities are open,} \end{cases}$$

where

$$W_x = \sum_{i,j \in I^+, i \ne j} \frac{w_i (x_i - x_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} - \sum_{i,j \in I^-, i \ne j} \frac{w_i (x_i - x_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}},$$
$$W_y = \sum_{i,j \in I^+, i \ne j} \frac{w_i (y_i - y_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} - \sum_{i,j \in I^-, i \ne j} \frac{w_i (y_i - y_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}.$$

P.-C. Chen and others [5] and F. Plastria [14] derived independently the following sufficient condition for a demand point to be the global minimum solution.

Property 5 For the unconstrained problem, if there is a point $i^* \in I^+$ such that

$$w_{i^*} \ge \sum_{i \in I^+ \cup I^-, i \ne i^*} w_i,$$

then (x_{i^*}, y_{i^*}) is the global optimum location.

It is quite straightforward to show that if all weights are positive then the expression for the weighted sum of the Euclidean distances is convex [13] and therefore the single local minimum corresponds to the global minimum. This means that the total expression for the sum of weighted Euclidean distances is a difference of two convex functions. As it has been noted earlier the presence of negative weights greatly complicates the location of the global minimum solution by introducing concave contributions in the objective function. This special class of difference of two convex functions (DC) optimization problems has recently (1990) received considerable attention [7]. The next theorem introduces a set of conditions for convexity of $F(x, y)$ at some point (x, y).

Property 6 $F(x, y)$ is convex at (x, y) if

$$\sum_{i \in I^+ \cup I^-} \frac{W_i}{r_i} \geq 0,$$

and

$$\sum_{i \in I^+ \cup I^-} \sum_{\substack{j \in I^+ \cup I^- \\ j > i}} \frac{W_i W_j}{r_i^3 r_j^3} \times \left[(x - x_i)(y - y_j) + (x - x_j)(y - x_i)\right]^2 \geq 0,$$

where $r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}$, $i \in I^+ \cup I^-$ and

$$W_i = \begin{cases} w_i, & i \in I^+, \\ -w_i, & i \in I^-. \end{cases}$$

A proof of this property can be found in [11].

A special case of this problem, involving three points with weights equal to one, was first posed by P. Fermat in the seventeenth century and it was solved geometrically by E. Toricelli. E. Weiszfeld [23] first proposed a simple iterative algorithm but with no convergence proof. Later, H.W. Kuhn [8,9,10] proved that Weiszfeld's algorithm was convergent assuming no iterate coincided with any of the demand points. L.M. Ostresh [12] and E. Balas and others [1] proposed modifications of the Weiszfeld algorithm where by perturbing the current point, if it coincided with a demand point, was global convergence guaranteed. C.Y. Wang [22] proved that Weiszfeld's algorithm has linear rate of convergence under certain conditions and sublinear otherwise. More recently (1980s), P.H. Calamai and A.R. Conn [2,3,4] and M.L. Overton [13] introduced second order methods which involved local quadratic convergence and global convergence under conditions. G.L. Xue [25,26], and Xue and J.B. Rosen [15] proved unconditional global convergence and conditional local quadratic convergence for a second order algorithm and computational comparisons were carried out between Weiszfeld's algorithm and Newton's algorithm on a parallel machine.

Most papers address only positive weights reflecting the inherent assumption that all points 'attract' the central facility. However, in real world there exists an abundance of example problems where certain points 'repel' the central facility. For example nuclear plants, sewage treatment plants, or polluting industrial units may be desired to be as close as possible to their customers so that transportation costs are minimized but at the same time environmental considerations require that these facilities be as far as possible from residential areas and fragile ecological systems. This need to locate a facility away from certain points can be quantified through the use of negative weights as shown in [16,19]. A negative weight means that the value of the objective function is increased as the facility approaches the corresponding point. Therefore, the global optimum location of a facility is now the one that balances the repulsion and the attraction acting on the central facility. It is interesting to note that the introduction of negative weights greatly increases the complexity of the problem.

Weber's problem with some negative weights was first considered by L.-N. Tellier [17], who studied the case of two attractive and one repulsive point. Later, Tellier and D. Pollanski [18] analyzed exhaustively all different cases involving three demand points and derived statistical conclusions regarding the types of possible solutions. Drezner and Wesolowsky [6] proved a number of theoretical results and proposed a heuristic algorithm for locating the global minimum solution. However, it was Chen and others [5] who first presented an exact outer approximation algorithm for Weber's problem with attraction and repulsion by exploiting the d.c. structure of the problem. In addition, they [5] extend their procedure to exponentially decaying repulsion and facility location within a set of disjoint convex polygons. Later, Maranas and Floudas [11] proposed a branch and bound type global optimization algorithm for solving Weber's problems with attraction

and repulsion. The approach was based on the iterative solution of a set of convex and concave lower bounding problems. Convergence to an ϵ-global minimum was proven and examples were solved with as many as 10,000 points. By analyzing the computational results they observed that for any given number of points N the difficulty of the problem increases as we introduce more repulsive points. This trend continues until about equal numbers of attractive and repulsive points are reached. Then, a sharp decrease in computational requirements is observed as more repulsive points are added. In fact, it is easier to solve problems involving more repulsive points than attractive ones. The standard deviation of the total number of required iterations and function evaluations is fairly small for all ratios of attractive to repulsive points with the sole exception of the $N^+ = N^- = N/2$ case where the standard deviation is substantially increased. For a given ratio of attractive to repulsive points the CPU requirements increase almost linearly with N reflecting the fact that most of CPU time is spent on function evaluations.

A generalization of Weber's problem is the maximization of the sum of decreasing convex functions of arbitrary metrics. H. Tuy and F.A. Al-Khayyal [20] proposed the first algorithm for finding global solutions to the problem by reducing it to a sequence of unconstrained nondifferentiable convex minimization problems. Later, they [21] extended this work to account for repulsion as well and proposed a d.c. reformulation of the problem which enabled them to develop a global optimization procedure.

See also

- ► Combinatorial Optimization Algorithms in Resource Allocation Problems
- ► Competitive Facility Location
- ► Facility Location with Externalities
- ► Facility Location Problems with Spatial Interaction
- ► Facility Location with Staircase Costs
- ► MINLP: Application in Facility Location-allocation
- ► Multifacility and Restricted Location Problems
- ► Network Location: Covering Problems
- ► Optimizing Facility Location with Rectilinear Distances
- ► Production-distribution System Design Problem
- ► Resource Allocation for Epidemic Control
- ► Single Facility Location: Circle Covering Problem
- ► Single Facility Location: Multi-objective Euclidean Distance Location
- ► Single Facility Location: Multi-objective Rectilinear Distance Location
- ► Stochastic Transportation and Location Problems
- ► Voronoi Diagrams in Facility Location
- ► Warehouse Location Problem

References

1. Balas E, Yu CS (1982) A note on the Weiszfeld–Kuhn algorithm for the general Fermat problem. Managem Sci Res Report 484:1–6
2. Calamai PH, Conn AR (1980) A stable algorithm for solving the multifacility location problem involving Euclidean distances. SIAM J Sci Statist Comput 1:512–526
3. Calamai PH, Conn AR (1982) A second-order method for solving the continuous multifacility location problem. In: Numerical Analysis. Proc. 9th Biennial Conf. Dundee, Scotland, pp 1–25
4. Calamai PH, Conn AR (1987) A projected Newton method for lp norm location problems. Math Program 38:75–109
5. Chen P-C, Hansen P, Jaumard B, Tuy H (1992) Weber's problem with attraction and repulsion. J Reg Sci
6. Drezner Z, Wesolowsky GO (1991) The Weber problem on the plane with some negative weights. INFOR 29:87–99
7. Horst R, Tuy H (1990) Global optimization, deterministic approaches. Springer, Berlin
8. Kuhn HW (1967) On a pair of dual nonlinear programs. Nonlinear Programming. North-Holland, Amsterdam, pp 38–54
9. Kuhn HW (1973) A note on Fermat's problem. Math Program 4:94–107
10. Kuhn HW (1974) Steiner's problem revisited. In: Studies in Optimization. Math. Assoc. America, Washington, DC, pp 52–70
11. Maranas CD, Floudas CA (1993) A global optimization method for Weber's problem with attraction and repulsion. In: Proc. Large Scale Optimization: State of the Art Conf., Florida Univ., 15-17 Feb. 1993. Kluwer, Dordrecht, pp 259–293
12. Ostresh LM (1978) On the convergence of a class of iterative methods for solving the Weber location problem. Oper Res 26:597–609
13. Overton ML (1983) A quadratically convergent method for minimizing a sum of Euclidean norms. Math Program 27:34–63
14. Plastria F (1992) The effects of majority in Fermat–Weber problems with attraction and repulsion. YUGOR 1
15. Rosen JB, Xue G-L (1991) Computational comparison of two algorithms for the Euclidean single facility location problem. ORSA J Comput 3:207–212

16. Tellier L-N (1972) The Weber problem: Solution and interpretation. Geographical Anal 4:215–233
17. Tellier L-N (1985) Économie patiale: rationalitée économique de l'espace habité. Gaétan Morin, Chicoutimi, Québec
18. Tellier L-N (1989) The Weber problem: frequency of different solution types and extension to repulsive forces and dynamic processes. J Reg Sci 29:387–405
19. Tellier L-N, Ceccaldi X (1983) Phenomenes de polarization et de repulsion dans le context du probleme de Weber. Canad Regional Sci Assoc
20. Tuy H, Al-Khayyal FA (1992) Global optimization of a nonconvex single facility problem by sequential unconstrained convex minimization. J Global Optim 2:61–71
21. Tuy H, Al-Khayyal FA, Zhou F (1995) A D.C. optimization method for single facility location problems. J Global Optim 2:61–71
22. Wang CY (1975) On the convergence and rate of convergence of an iterative algorithm for the plant location problem. Qufu Shiyun Xuebao 2:14–25
23. Weiszfeld E (1937) Sur le point pour lequel la somme des distances de n points donnés est minimum. Tôhoku Math J 43:355–386
24. Witzgall C (1984) Optimal location of a single facility: Mathematical models and concepts. Report Nat Bureau Standards 8388
25. Xue G-L (1987) A fast convergent algorithm for min $\sum_{i=1}^{m} \| x - a_i \|$ on a closed convex set. J Qufu Normal Univ 13(3):15–20
26. Xue G-L (1989) A globally and quadratically convergent algorithm for min $\sum_{i=1}^{m} \| x - a_i \|$ type plant location problem. Acta Math Applic Sinica 12:65–72

Global Pairwise Protein Sequence Alignment via Mixed-Integer Linear Optimization

S. R. McAllister, R. Rajgaria, Christodoulos A. Floudas
Department of Chemical Engineering, Princeton University, Princeton, USA

MSC2000: 90C10, 92-08

Article Outline

Abstract
Keywords and Phrases
Introduction
Models
  Template-Based Model
  Template-Free Model
  Path Selection Model
  Functionally-Specific Constraints
  Integer Cut Constraints
  Pairwise Interaction Scores
Results and Discussion
  G-protein Coupled Receptors
  Serine Protease Inhibitors
References

Abstract

A well-studied problem in the area of computational biology is the sequence alignment problem. Three mixed-integer linear optimization models have been developed to address the global pairwise sequence alignment problem in a mathematically rigorous fashion. These formulations, in addition to their rigor, allow for (a) the natural introduction of functionally important conservation constraints, (b) the creation of a rank-ordered list of the highest scoring alignments and (c) the refinement of alignments by using pairwise interaction scores from simplified force fields. The third model, a path selection approach, employs some of the algorithmic advantages of dynamic programming methods, to outperform other optimization models.

Keywords and Phrases

Sequence alignment; Integer linear optimization; Global pairwise alignment; Rank-order list of alignments

Introduction

Sequence alignment methods aim to both identify related protein sequences and determine the best alignment between them. This approach provides a rough measure of evolutionary distance and may indicate possible relationships between the protein structure and function of similar sequences. Multiple scoring matrices have been developed based on the techniques of the percent of accepted mutations (PAM) [3] and protein blocks (BLOSUM) [5] to quantify this evolutionary distance between aligned residues.

The pairwise sequence alignment problem is most commonly addressed through either (i) global alignment or (ii) local alignment techniques. The goal of global alignment algorithms is to determine the highest

scoring overall alignment spanning the length of both sequences. One widely used approach for this problem is a dynamic programming approach proposed by Needleman and Wunsch [10].

Proteins may share sequence similarity in some regions, but not in others. Local alignment algorithms are more suited to these problems and strive to align only the highest scoring subsequence match. Smith and Waterman extended the dynamic programming approach for global pairwise sequence alignment problems to address the local alignment problem [14]. Dynamic programming approaches are computationally inadequate for large scale database searches, so a number of heuristic algorithms for local pairwise sequence alignment have been proposed [1,2,11,12].

Several researchers have studied the effect of including information about near-optimal alignments. The investigation of the suboptimal paths and scores allows for an evaluation of the reliability of portions of a sequence alignment. A review of several approaches to this problem and their impact can be found elsewhere [17].

In some cases, an alignment between two sequences can be improved by constraining the problem to include biologically important information in the overall alignment. One example of this is the required conservation of certain residues that form a motif necessary for function. This problem has been addressed recently by dynamic programming algorithms [4,15].

Models

Several integer linear optimization (ILP) models have been developed to rigorously and completely address the problem of global pairwise sequence alignment in a general fashion. A comparison of the three approaches, a template-based model, a template-free model, and a path selection model are presented in the following sections. The formulation of the problem as an integer linear optimization problem provides a deterministic guarantee of identifying the global maximum alignment [6], allows for the introduction of integer cut constraints, provides a framework for the introduction of functionally-specific constraints, and shows promise for the optimal identification of pairwise interactions.

Template-Based Model

Consider two protein sequences $S1$, $S2$ of lengths M and N respectively, where $M > N$. Let the index i represent each position in Sequence S1 and the index j represent each position in S2, as shown in Eqs. 1–2.

$$i \in 1, 2 \ldots M \tag{1}$$

$$j \in 1, 2 \ldots N \tag{2}$$

The template-based optimization model assigns each amino acid of both sequences to a template to generate the optimal alignment. Equation 3 defines a template length K as the sum of the length of the larger sequence and the parameter N_GAPS_m, representing the maximum number of allowed gaps. This model requires the introduction of an index k, representing the position in the template, as defined by Eq. 4.

$$K = M + N_GAPS_m \tag{3}$$

$$k \in 1, 2 \ldots K \tag{4}$$

The assignment of an amino acid to a template position requires the definition of the binary variables, y_{ik} and z_{jk}, as shown in Eqs. 5–6.

$$y_{ik} = \begin{cases} 1 & \text{if amino acid } i \text{ of S1 is assigned to template position} k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$z_{jk} = \begin{cases} 1 & \text{if amino acid } j \text{ of S2 is assigned to template position } k \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

A position in the template may not have an amino acid assigned to it in the overall alignment. Therefore, Eqs. 7–8 introduce additional binary variables to represent these alignment gaps.

$$yg_k = \begin{cases} 1 & \text{if template position } k \text{ is a gap for Sequence S1} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$zg_k = \begin{cases} 1 & \text{if template position } k \text{ is a gap for Sequence S2} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The objective function of this optimization model maximizes the alignment score, which is the sum of a scoring matrix value for each matching amino acid pair minus any associated penalties for gaps inserted in the

sequence. The scoring matrix will assign a weight, w_{ij}, to any template position k that contains the amino acid in position i of Sequence S1 and also the amino acid in position j of S2.

For an affine gap penalty model with no penalties for gaps that begin or end a sequence, the objective function can then be posed as shown in Eq. 9. The contribution of the scoring matrix at positions i, j,w_{ij}, is considered only if position i of Sequence 1 is assigned to position k of the template, y_{ik}, and if position j of Sequence 2 is assigned to position k of the template, z_{jk}. The gap opening existence terms of Sequence 1, go_k^{S1}, and Sequence 2, go_k^{S2}, are weighted by the gap opening penalty value of wo to assess the penalty for the first residue of any gap in a sequence. The existence of a gap extension variable of either sequence, gl_k^{S1} and gl_k^{S2}, produces a penalty of wl for each occurrence. The active gl_k^{S1} and gl_k^{S2} variables that are contained within a beginning or an ending gap are counteracted by the product of gb_k^{S1}, gb_k^{S2}, ge_k^{S1}, or ge_k^{S2} with wl. Gap opening penalties at the beginning or end of a sequence are explicitly omitted through only summing over the reduced index, such that $2 \leq k \leq K-1$.

$$\begin{aligned}\max \sum_i \sum_j \sum_k w_{ij} \cdot y_{ik} \cdot z_{jk} - \sum_{k=2}^{K-1} (go_k^{S1} + go_k^{S2}) \cdot wo \\ - \sum_{k=2}^{K-1} \left[(gl_k^{S1} - gb_k^{S1} - ge_k^{S1}) \right. \\ \left. + (gl_k^{S2} - gb_k^{S2} - ge_k^{S2}) \right] \cdot wl\end{aligned} \tag{9}$$

The objective function of Eq. 9 requires the linearization of the product of two binary variables and is subject to numerous constraints. The details of the model are available elsewhere [8].

Template-Free Model

Unlike the previously described mixed-integer linear programming formulation of the global pairwise sequence alignment problem in Sect. "Template-Based Model", the optimization model presented here does not assign the amino acids of each sequence to a template. However, information about the maximum number of allowable gaps is still included in this model, through the variable K in Eq. 3.

In the template-free model, a binary variable, z_{ij}, is defined in Eq. 10 to represent the alignment of position i in $S1$ to position j in $S2$. A method to handle gaps in the sequence still must be introduced into the model to account for the evolutionary changes that lead to residue insertions and deletions. Aligning a gap residue to another gap residue is not allowed. This observation leads to two possibilities of gap occurrences. A gap can either be in Sequence 1, across from a residue j in Sequence 2 or in Sequence 2, across from a residue i in Sequence 1. These possibilities are modeled with the binary variables zg_i and yg_j, defined by Eq. 11–12.

$$z_{ij} = \begin{cases} 1 & \text{if position } j \text{ in } S2 \text{ aligns with position } i \text{ in } S1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$zg_i = \begin{cases} 1 & \text{if no position } j \text{ in } S2 \text{ aligns to the residue in position } i \text{ of } S1 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$yg_j = \begin{cases} 1 & \text{if no position } i \text{ in } S1 \text{ aligns to the residue in position } j \text{ of } S2 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

The objective function in Eq. 13 maximizes the sum of the weights of the residue-residue alignments minus the sum of the gap penalties, plus the appropriate terms that remove the penalties from the gaps at the beginning and ends of the sequences. The scoring matrix values at any given pair of positions, w_{ij} are included when the binary variables indicating a sequence alignment that matches positions i and j, z_{ij}, are activated. For an affine gap penalty model, the variables representing the existence of a gap opening, go_j^{S1} and go_i^{S2}, and the existence of a gap extension, gl_j^{S1} and gl_i^{S2}, are multiplied by their respective weights, wo and wl. If a gap residue is present at the beginning or ending of a sequence, it will be accounted for in an active value for one of gb_j^{S1}, ge_j^{S1}, gb_i^{S2}, ge_i^{S2} to remove the penalty assigned by the previous terms.

$$\begin{aligned}\max \sum_{ij} w_{ij} z_{ij} \\ - \sum_j (wo \cdot go_j^{S1} + wl \cdot gl_j^{S1})\end{aligned}$$

$$
\begin{aligned}
&- \sum_{i} (wo \cdot go_i^{S2} + wl \cdot gl_i^{S2}) \\
&+ \sum_{j>1}^{N-1} wl \cdot (gb_j^{S1} + ge_j^{S1}) \\
&+ \sum_{i>1}^{M-1} wl \cdot (gb_i^{S2} + ge_i^{S2}) \\
&+ wo \cdot (ge_1^{S1} + ge_1^{S2}) \\
&+ wo \cdot (ge_M^{S1} + ge_N^{S2})
\end{aligned} \tag{13}
$$

The objective function of Eq. 13 is subject to numerous constraints. The details of these constraints are available elsewhere [9].

Path Selection Model

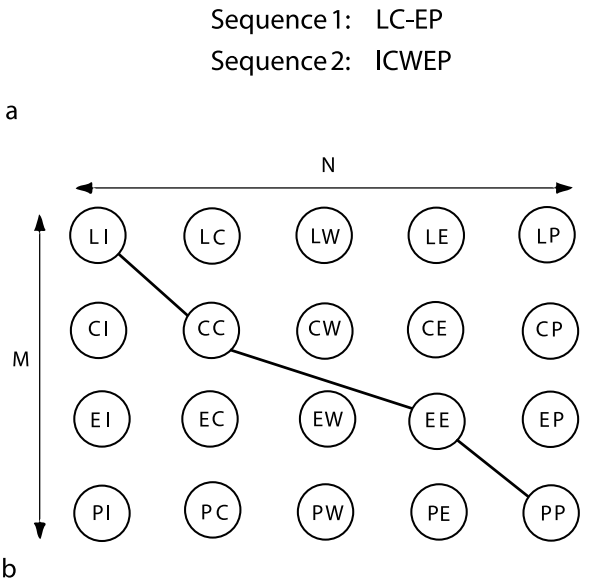
Let us introduce a binary variable N_{ij} that represents the alignment of the residue at position i in Sequence 1 to the residue at position j in Sequence 2. This binary variables performs a similar role as z_{ij} in Sect. "Template-Free Model". The typical assignment of this match assesses a weight, w_{ij}, based on a scoring matrix developed through evolutionary analysis of protein sequences.

A successful sequence alignment will have many active N_{ij} variables, which we will designate as nodes. Let the binary variable $y_{ii'jj'}$ represent the existence of a connecting path between node N_{ij} and a neighboring node $N_{i'j'}$. Associated with this connecting path, is a weight parameter, $C_{ii'jj'}$, which can be calculated in advance from the scoring matrix w and any position dependent gap penalty form that is specified a priori. An example of the representation of the node and path variables is illustrated in Fig. 1.

Once these variables have been defined, the objective function of the optimal sequence alignment is merely the sum of the product of the variable for the existence of the path, $y_{ii'jj'}$, and the path weight, $C_{ii'jj'}$ as shown in Eq. 14.

$$
\max \sum_{i} \sum_{i'>i} \sum_{j} \sum_{j'>j} y_{ii'jj'} \cdot C_{ii'jj'} \tag{14}
$$

The variable $y_{ii'jj'}$ is defined only as the existence of a contact between two neighboring nodes, where each node $N_{i',j'}$ that has an incoming connecting path activated must also have an outgoing path. In effect, this constraint can be thought of as a "mass" balance around the node. This constraint is specified for all nodes except those that are allowed to begin or end an alignment by Eq. 15.



Global Pairwise Protein Sequence Alignment via Mixed-Integer Linear Optimization, Figure 1
(a) Alignment of two hypothetical sequence fragments. (b) A node and path representation of the alignment problem as formulated by the mathematical model. Note the three active paths connecting the four selected node variables

$$
\begin{aligned}
&\sum_{i<i'} \sum_{j<j'} y_{ii'jj'} - \sum_{i''>i'} \sum_{j''>j'} y_{i'i''j'j''} = 0 \\
&\forall\, 1 < i' < M,\ 1 < j' < N
\end{aligned} \tag{15}
$$

Equation 16 requires an alignment that matches the first residue in one of the two sequences to a residue in the other sequence. This constraint invalidates any alignment that aligns the first residue in both sequences to a gap, a physically meaningless alignment and allows for the path weights, $C_{ii'jj'}$, to be precalculated.

$$
\begin{aligned}
&\sum_{i'>1} \sum_{j} \sum_{j'>j} y_{i=1,i'jj'} + \sum_{i} \sum_{i'>i} \sum_{j'>j} y_{ii',j=1,j'} \\
&\qquad - \sum_{i'>1} \sum_{j'>1} y_{i=1,i',j=1,j'} = 1
\end{aligned} \tag{16}
$$

If one sequence ends in a gap, the terminal residues of the other sequence must be prevented from aligning to earlier residues in a physically unrealistic way. Equa-

tion 17 allows exactly one active node $N_{i,j}$ involving a terminal residue in either Sequence 1 or Sequence 2.

$$\sum_{i<M}\sum_{j}\sum_{j'>j} y_{i,i'=M,jj'} + \sum_{i}\sum_{i'>i}\sum_{j<N} y_{ii'j,j'=N} - \sum_{i<M}\sum_{j<N} y_{i,i'=M,j,j'=N} = 1 \tag{17}$$

It is more efficient and more meaningful to restrict the search to within a maximum alignment length, K. Equations 18–19 require the sum of the sequence length and the number of gaps created by the alignment to be less than the maximum alignment length for Sequences 1 and 2 respectively.

$$\begin{aligned}&\sum_{i}\sum_{i'>i}\sum_{j}\sum_{j'>j+1}(j'-j)\cdot y_{ii'jj'} + \\ &\sum_{i}\sum_{i'>i}\sum_{j'>1}(j-1)\cdot y_{ii',j=1,j'} + \\ &\sum_{i}\sum_{i'>i}\sum_{j<N}(j'-N)\cdot y_{ii'jj'=N} + M \le K\end{aligned} \tag{18}$$

$$\begin{aligned}&\sum_{i}\sum_{i'>i+1}\sum_{j}\sum_{j'>j}(i'-i)\cdot y_{ii'jj'} + \\ &\sum_{i'>1}\sum_{j}\sum_{j'>j}(j-1)\cdot y_{i=1,i'jj'} + \\ &\sum_{i<M}\sum_{j}\sum_{j'>j}(j'-N)\cdot y_{i,i'=M,jj'} + N \le K\end{aligned} \tag{19}$$

Equations 14–19 form the general mathematical model for the path selection approach to the global pairwise sequence alignment problem. Any of the three models presented can be expanded to include functionally-specific constraints, integer cut constraints, and pairwise interactions. Only the constraints necessary to include these features in the path selection model will be presented here.

Functionally-Specific Constraints

For some sequence alignment problems, specific residues are related to the function of a protein and should be maintained in a meaningful sequence alignment. This idea can be enforced in a mathematically rigorous way. These constraints can only be defined if the node existence variables, N_{ij}, are connected to the path existence variables, $y_{ii'jj'}$. One way to accomplish this is by summing over a pair of indices within the path variables, as shown in Eqs. 20–21.

$$\sum_{i<i',j<j'} y_{ii'jj'} = N_{i'j'} \quad \forall i' > 1, j' > 1 \tag{20}$$

$$\sum_{i'>i,j'>j} y_{ii'jj'} = N_{ij} \quad \forall i = 1 \text{ or } j = 1 \tag{21}$$

Constraints enforcing residue identity can then be written in terms of the N_{ij} variables. If position i^* in Sequence 1 must be conserved to maintain function, then Eq. 22 enforces this requirement.

$$\sum_{j;AA_{i^*}=AA_j} N_{i^*j} = 1 \tag{22}$$

Integer Cut Constraints

This alignment model can be further extended by introducing integer cut constraints. After each solve of the above model, the previous solution is excluded from the feasible solution space by Eq. 23. A is the set of active variables in the solution to be excluded, I is the set of inactive variables and card(A) is the cardinality of set A, or the number of members of set A.

$$\sum_{(ii'jj')\in A} y_{ii'jj'} - \sum_{(ii'jj')\in I} y_{ii'jj'} \le \text{card}(A) - 1 \tag{23}$$

Pairwise Interaction Scores

A score can also be assigned for the alignment of a pair of amino acids i, i' in one sequence to a specific pair of amino acids j, j' in the second sequence. One promising application of these pairwise interactions scores is the ability to better evaluate the fitness of an alignment between a protein of known structure and an unknown protein with remote sequence homology. A number of recently developed C^{α}-based distance dependent force fields [7,13,16] are a good source for these scores because they allow some flexibility between the backbones of these two structures.

A pairwise interaction score requires the definition of the variable $z_{ii'jj'}$, representing the successful alignment of both i, j (N_{ij}) and i', j' ($N_{i'j'}$). This variable is initially introduced in Eq. 24 as the product of two node existence binary variables.

$$z_{ii'jj'} = N_{ij} \cdot N_{i'j'} \quad \forall i, i', j, j' \tag{24}$$

Equation 24 is nonlinear and must be linearized using standard optimization techniques by Eqs. 25–27, which replace Eq. 24.

$$\sum_{ij} z_{ii'jj'} \leq N_{i'j'} \quad \forall i', j' \tag{25}$$

$$\sum_{i'j'} z_{ii'jj'} \leq N_{ij} \quad \forall i, j \tag{26}$$

$$N_{ij} + N_{i'j'} - 1 \leq z_{ii'jj'} \quad \forall i, i', j, j' \tag{27}$$

Let the score of a pairwise interaction be denoted as $P_{ii'jj'}$. The objective function of Eq. 14 is expanded to include an additional contribution as shown in Eq. 28.

$$\max \sum_{i} \sum_{i'>i} \sum_{j} \sum_{j'>j} y_{ii'jj'} \cdot C_{ii'jj'} + z_{ii'jj'} \cdot P_{ii'jj'} \tag{28}$$

The ability of the sequence alignment models to easily allow for pairwise interaction scores illustrates their true power and flexibility. The model is guaranteed to converge to the optimal solution even for problems of this type. This guarantee suggests the effectiveness that could be achieved by incorporating such a model into a fold recognition framework.

Results and Discussion

The mixed-integer linear programming models of Sect. "Models" can address generic sequence alignment problems of a reasonable size. This method will be illustrated on an alignment of G-protein coupled receptors with the use of integer cut constraints and an alignment of pancreatic trypsin inhibitors demonstrating the use of functionally-relevant conservation constraints. All the alignments are calculated using the BLOSUM62 scoring matrix and an affine gap model with a gap opening penalty of 11 and a gap extension penalty of 1.

G-protein Coupled Receptors

G-protein coupled receptors are a type of membrane protein that regulate material and ion transport across a cell membrane, a reason they are a popular target for drug development. The alignment of the seventh transmembrane helix of bovine rhodopsin (34 amino acids) to the seventh transmembrane helix of H1R (35 amino acids), the first human histamine receptor, will be considered to illustrate alignment uncertainty [9]. Figure 2 shows the the regions of uncertainty in the sequence alignment using integer cut constraints. There is a strong conservation of alignment at the ends of the selected sequence, including the preservation of the highly conserved NPxxY motif. The central regions of the aligned sequences shows more variability. This observation could be a result of less structural conservation in the region, or less sequence similarity required for structural (and functional) conservation.

A comparison of the computational resources required for this problem is presented in Table 1. A larger template length results in a more complex optimization problem to be solved. The path selection model significantly outperforms the other formulations, especially for the larger template lengths.

```
Sequence 1:
KNCCNEHLHM FTIWLGYINS TLNPLIYPLC NENFK
Sequence 2:
SDFGPIFMTI PAFFAKTSAV YNPVIYIMMN KQFR

ITERATION: 1        OBJECTIVE:   26 (9 matches)
    1234567890 1234567890 1234567890 12345678
S1:  KNCCNEHLHM F-TI--WLGY INSTLNPLIY PLCNENFK
                | ||           || ||    |  |
S2:  ----SDFGPI FMTIPAFFAK TSAVYNPVIY IMMNKQFR
--------------------------------------------------
ITERATION: 2        OBJECTIVE:   25 (8 matches)
    1234567890 1234567890 1234567890 12345678
S1:  KNCCNEHLHM FTIWLGYINS T---LNPLIY PLCNENFK
                |          |    || ||    |  |
S2:  ----SDFGPI FMTIPAFFAK TSAVYNPVIY IMMNKQFR
--------------------------------------------------
ITERATION: 3        OBJECTIVE:   25 (7 matches)
    1234567890 1234567890 1234567890 12345678
S1:  KNCCNEHLHM FTI---WLGY INSTLNPLIY PLCNENFK
                |             || ||    |  |
S2:  ----SDFGPI FMTIPAFFAK TSAVYNPVIY IMMNKQFR
--------------------------------------------------
ITERATION: 4        OBJECTIVE:   25 (7 matches)
    1234567890 1234567890 1234567890 12345678
S1:  KNCCNEHLHM FT---IWLGY INSTLNPLIY PLCNENFK
                |           || ||    |  |
S2:  ----SDFGPI FMTIPAFFAK TSAVYNPVIY IMMNKQFR
```

Global Pairwise Protein Sequence Alignment via Mixed-Integer Linear Optimization, Figure 2
A rank-ordered list of the top four optimal alignments of the helix 7 region of the human histamine receptor (Sequence 1) to the helix 7 region of the bovine rhodopsin (Sequence 2) for a template length of 50 residues

```
Sequence 1:
MLKYTSISFL LIILLFSFTN ANPDCLLPIK TGPCKGSFPR YAYDSSEDKC
VEFIYGGCQA NANNFETIEE CEAACL

Sequence 2:
RPDFCLEPPY TGPCKARIIR YFYNAKAGLC QTFVYGGCRA KRNNFKSAED
CMRTCGGA

OBJECTIVE:     141 (26 exact matches)
    1234567890 1234567890 1234567890 1234567890 1234567890
S1:  MLKYTSISFL LIILLFSFTN ANPD-CLLPI KTGPCKGSFP RYAYDSSEDK
                             || || |   |||||     || |
S2:  ---------- ---------- -RPDFCLEPP YTGPCKARII RYFYNAKAGL

S1:  CVEFIYGGCQ ANANNFETIE ECEAACL--
     |  | ||||  |  |||   |  |   |
S2:  CQTFVYGGCR AKRNNFKSAE DCMRTCGGA
```

Global Pairwise Protein Sequence Alignment via Mixed-Integer Linear Optimization, Figure 3
Optimal alignment of bombyx mori kazal-type serine proteinase inhibitor 1 (Sequence 1) to bovine pancreatic trypsin inhibitor (Sequence 2), given the requirement of cysteine conservation and a template length of 100

Global Pairwise Protein Sequence Alignment via Mixed-Integer Linear Optimization, Table 1
Computational performance of the template-based (TB), template-free (TF) and path selection (PS) models for helix 7 of the G-protein coupled receptor proteins (run times in seconds on an Intel Pentium 3.2 GHz processor, using CPLEX 9.0)

K	TB	TF	PS	Objective
40	1000+	35.21	1.30	26,25,25,25
45	1000+	177.8	5.27	26,25,25,25
50	1000+	1000+	14.71	26,25,25,25

Global Pairwise Protein Sequence Alignment via Mixed-Integer Linear Optimization, Table 2
Computational performance of the template-based (TB), template-free (TF) and path selection (PS) models for the alignment of bombyx mori kazal-type serine proteinase inhibitor 1 to bovine pancreatic trypsin inhibitor (run times in seconds on an Intel Pentium 3.2 GHz processor, using CPLEX 9.0)

K	TB	TF	PS	Objective
80	886.4	0.36	0.64	141
90	1000+	80.3	1.74	141
100	1000+	912.4	2.77	141

Serine Protease Inhibitors

Serine protease inhibitors are responsible for regulating serine proteases, proteins necessary for hydrolyzing peptides. One well-studied protein within this class is the bovine pancreatic trypsin inhibitor (BPTI). Its native three-dimensional structure is stabilized by 3 disulfide bonds that are conserved across the class of serine protease inhibitors. An alignment of BPTI (58 amino acids) to the bombyx mori (domestic silkworm) kazal-type serine protease inhibitor (76 amino acids) has previously been investigated in the context of introducing constraints for the functionally important conservation of the disulfide bonds [8]. The results of such an alignment are presented in Fig. 3. The six conserved cysteine residues necessary for the formation of the three disulfide bridges that stabilize the functional protein are apparent from this alignment.

A comparison of the computational resources required for this problem is presented in Table 2. Even with the inclusion of the conservation constraints, the path selection model still solves this alignment example quite rapidly for large template lengths. Although the template-free approach slightly outperforms the path selection approach for short template length restrictions, it does not scale very well with increases in template length. Similar to the first example, the template-free approach solves the problem significantly faster than the template-based approach, but the path selection approach is superior to both of the mixed-integer linear programming techniques.

References

1. Altschul S, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. J Mol Biol 215:403–410
2. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucl Acids Res 25:3389–3402
3. Dayhoff M, Schwartz R, Orcutt B (1978) A model of evolutionary change in proteins. Atlas of Protein Sequences and Structures, vol 5. National Biomedical Research Foundation, Washington DC
4. He D, Arslan AN (2005) A space-efficient algorithm for the constrained pairwise sequence alignment problem. Genome Inform 16:237–246
5. Henikoff S, Henikoff JG (1992) Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci USA 89:10915–10919
6. Floudas CA (1995) Nonlinear and Mixed-integer Optimization: Fundamentals and Applications. Oxford University Press, New York
7. Loose C, Klepeis JL, Floudas CA (2004) A new pairwise folding potential based on improved decoy generation and side-chain packing. Prot Struct Funct Bioinf 54:303–314
8. McAllister SR, Rajgaria R, Floudas CA (2007) A template-based mixed integer linear programming sequence alignment model. In: Torn A, Zilinskas J (eds) Models and Algorithms for Global Optimization, Springer Optimization and Its Applications. Springer, New York, pp 343–360
9. McAllister SR, Rajgaria R, Floudas CA (2007) Global pairwise sequence alignment through mixed integer linear programming: A template free approach. Optim Method Softw 22:127–144
10. Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48:443–453
11. Pearson WR (1990) Rapid and sensitive sequence comparison with fastp and fasta. Methods Enzymol 183:63–98
12. Pearson WR, Lipman DJ (1988) Improved tools for biological sequence comparison. Proc Natl Acad Sci USA 85:2444–2448
13. Rajgaria R, McAllister SR, Floudas CA (2006) A novel high resolution C-alpha C-alpha distance dependent force field based on a high quality decoy set. Prot Struct Funct Bioinf 65:726–741
14. Smith TF, Waterman MS (1981) Identification of common molecular subsequences. J Mol Biol 147:195–197
15. Tang CY, Lu CL, Chang MD, Tsai YT, Sun YJ, Chao KM, Chang JM, Chiou YH, Wu CM, Chang HT, Chou WI (2003) Constrained multiple sequence alignment tool development and its application to RNase family alignment. J Bioinform Comput Biol 1:267–287
16. Tobi D, Elber R (2000) Distance-dependent, pair potential for protein folding: Results from linear optimization. Prot Struct Funct Bioinf 41:40–46
17. Vingron M (1996) Near-optimal sequence alignment. Curr Opin Struct Biol 6:346–352

Global Supply Chain Models

BURAK EKSIOGLU
Industrial and Systems Engineering Department, University Florida, Gainesville, USA

MSC2000: 90B05, 90B06

Article Outline

Keywords
See also
References

Keywords

Stochastic dynamic programming; Multiperiod stochastic program

A *supply chain* (SC) may be defined as an integrated process where several business entities such as suppliers, manufacturers, distributors, and retailers work together to plan, coordinate and control the flow of materials, parts, and finished goods from suppliers to customers. This chain is concerned with two distinct flows: a forward flow of materials and a backward flow of information. Similarly, a *global supply chain* (GSC) may be defined as a SC where one or more of these business entities operate in different countries. For many years, researchers and practitioners have concentrated on the individual processes and entities within the SC. Within the past few years, however, there has been an increasing effort in optimizing the entire SC. This article intends to highlight some of the early results from the 1960s to 1995 that have led to today's SC research and most of the recent results that address the design and management of GSC networks (as of 2000).

Within manufacturing and logistics research, the current stream of SC research is largely built on prior work in the area of multi-echelon inventory models. The early works [4,5] and [14] form the basis for most of the research done in this area. See [13] and [3] for extensive reviews of multi-echelon inventory models. For detailed and more recent discussions of multi-echelon models, see [12,19,20].

As companies began to realize the benefits of optimizing the SC as a single entity, researchers began utilizing operations research (OR) techniques to better model supply chains. See [2] for an extensive review of the literature in SC modeling. Typically, a SC model tries to determine:

- the transportation modes to be used;
- the suppliers to be selected;
- the amount of inventory to be held at various locations in the chain;
- the number of warehouses and plants to be used; and
- the location and capacities of these warehouses and plants.

However, as a result of the globalization of the economy, the models have become more complex. GSC models now often try to include factors such as exchange rates, international interest rates, trade barriers, taxes and duties, market prices, and duty drawbacks. All of these factors are generally difficult to include in mathematical models because of the uncertainty and nonlinearity they introduce.

See [21] and [7] for extensive reviews on GSC models. [21] concentrates on strategic production-distribution models whereas [7] focus on the integration of SC network optimization with real options pricing methods. This article complements these reviews by giving a chronological listing of the models in both areas.

In [15] an international *facility location* model is presented. This is one of the first mathematical programs that includes financial aspects in GSC modeling. The authors develop a *large scale nonlinear mixed integer programming problem* (MIP). The objective function takes into account the expected profit and the variance of the profit, where the variance of the profit is multiplied by a risk aversion factor. Plant capacities, market demands and financial constraints are included in the model. The formulation considers production and transportation costs, exchange rate fluctuations, international interest rates, market prices, import tariffs, and export taxes.

In [9] a deterministic model is proposed for maximizing the after tax profit of a large scale international distribution network. Transportation costs, fixed setup costs, variable production and purchasing costs, and fixed vendor costs are included in the model. The model enforces production capacity constraints, demand limits, material requirements at each plant, supplier capacity constraints, balance constraints at plants and distribution centers, feasible flow constraints, and offset trade requirements. The model is run sequentially over a fixed time horizon and computational results are presented for various problem sizes.

In [6] the differences are analyzed between an international SC model and a single-country model, and a dynamic, nonlinear MIP model is developed. The inclusion of features such as duties, tariffs, tax rates, and exchange rates produce models that are very difficult to solve optimally even for small size problems.

In [8] a normative model is presented for the operations of a global company. Plant location, capacity and product mix, and material and cash flow determination are the decisions included in the model. The model consists of a master problem and a set of subproblems. The master problem is a *multiperiod stochastic program* and the subproblems are single period stochastic programs. These problems are linked through a set of submodels such as a stochastic SC model, a financial flow model, a stochastic exchange rate model, and a price-demand model.

In [17] a *stochastic dynamic programming* (DP) model is developed that treats the SC as equivalent to owning a financial option instrument. The value of the option depends on the real exchange rate. The authors consider production switching between two manufacturing plants located in different countries depending on the real exchange rate. The model does not consider characteristics such as multiple products or different SC stages. The model becomes intractible for more than one exchange rate process.

In [1] a comprehensive, multiperiod, multicommodity MIP model is proposed which is used to optimize the SC of Digital Equipment Corporation (DEC). The objective of the model is to minimize a function of total production and distribution cost, savings from credit, and an additional term which contains production and transportation times. The total cost includes fixed and variable costs of production, transportation cost, material handling, inventory, and overhead costs. The savings from credit are due to reexporting products. The model enforces constraints on demand satisfaction, production and throughput capacities at each facility, and bounds on decision variables. In addition,

international constraints such as duty drawback, duty relief, and offset trade are included. The authors describe how DEC used this model to manage their GSC.

In [18] a multiperiod stochastic DP is introduced that allows the firm to switch among several production modes to maximize profit. The production modes they consider are exporting from the home country, a joint venture with local partners, and establishing a wholly owned subsidiary for a foreign firm. It concludes by identifying cases in which one of these modes would be preferred to the others.

In [16] a stochastic DP formulation is developed for the valuation of global manufacturing strategy options. A hierarchical approach is proposed. First, the exchange rates are modeled by multinomial approximations. Then, options for alternative product and SC network designs are determined based on the firm's global manufacturing strategy. Finally, an MIP model for each exchange rate within every period is solved and the value of several manufacturing options is determined. The expected profit for each policy option is found by solving a stochastic DP using the values of the manufacturing policies.

In [11] the problem of operating a network of plants that are partially-owned subsidiaries of a multinational corporation is analyzed. Using real data, a model of three subsidiaries and four countries is developed for one industry and the effects of coordination under various macroeconomic conditions are discussed.

In [10] optimal policies for operating a network of plants located in different countries is studied. It is assumed that production costs are stochastic and are influenced by factors such as exchange rates, inflation, taxes, and tariffs. There is a one-time charge for switching (production volume changes between countries) and variable production costs are either concave or piecewise linear convex at each plant. It is also assumed that demand is deterministic and stationary. Under these assumptions a two-country, single market stochastic DP model is developed. The authors show that the optimal policy is always a barrier policy when switching costs are linear or step functions. (A *barrier policy* is a policy in which each plant operates either at a minimum or a maximum output level.)

The literature on GSC management is quite recent and the models developed usually do not consider most of the uncertainties that international corporations face. Each model addresses a limited number of the aspects of managing a GSC. There is an ongoing effort to develop more comprehensive and practical GSC design models that will accommodate the needs of the rapidly changing global economy.

See also

- ► Inventory Management in Supply Chains
- ► Nonconvex Network Flow Problems
- ► Operations Research Models for Supply Chain Management and Design
- ► Piecewise Linear Network Flow Problems

References

1. Arntzen BC, Brown GG, Harrison TP, Trafton LL (1995) Global supply chain management at Digital Equipment Corporation. Interfaces 25(1):69–93
2. Beamon BM (1998) Supply chain design and analysis: Models and methods. Internat J Production Economics 55:281–294
3. Bhatnagar R, Chandra P, Goyal SK (1993) Models for multi-plant coordination. Europ J Oper Res 67:141–160
4. Clark AJ, Scarf H (1960) Optimal policies for a multi-echelon inventory problem. Managem Sci 6(4):475–490
5. Clark AJ, Scarf H (1962) Approximate solutions to a simple multi-echelon inventory problem. In: Arros KJ, Karlin S, Scarf H (eds) Stud. Appl. Probab. and Management Sci., Stanford Univ. Press, Palo Alto, CA, pp 88–110
6. Cohen MA, Fischer M, Jaikumar R (1989) Internat. manufacturing and distribution networks: A normative model framework. In: Ferdows K (ed) Managing Internat. Manufacturing. North-Holland, Amsterdam, pp 67–93
7. Cohen MA, Huchzermeier A (1998) Global supply chain management: A survey of research and applications. In: Tayur S, Ganeshan R, Magazine M (eds) Quantitative Models for Supply Chain Management. Kluwer, Dordrecht
8. Cohen MA, Kleindorfer PR (1993) Creating value through operations: The legacy of Elwood S. Buffa. In: Sarin RK (ed) Perspectives in Oper. Management (Essays in Honor of Elwood S. Buffa), Kluwer, Dordrecht, pp 3–21
9. Cohen MA, Lee HL (1989) Resource deployment analysis of global manufacturing and distribution networks. J Manufacturing Oper Management 2:81–104
10. Dasu S, Li L (1997) Optimal operating policies in the presence of exchange rate variability. Managem Sci 43(5):705–722
11. Dasu S, de la Torre J (1997) Optimizing an international network of partially owned plants under contiditions of trade liberalization. Managem Sci 43(3):313–333

12. Diks EB, de Kok AG, Lagodimos AG (1996) Multi-echelon systems: A service measure perspective. Europ J Oper Res 95:241–263
13. Federgruen A (1993) Centralized planning models for multi-echelon inventory systems under uncertainty. In: Graves S, Rinnooy Kan AHG, Zipkin P (eds) Logistics of Production and Inventory, North-Holland, Amsterdam, pp 133–173
14. Geffrion AM, Graves GW (1974) Multicommodity distribution system design by Benders Decomposition. Managem Sci 20(5):822–844
15. Hodder JE, Dincer MC (1986) A multifactor model for international plant location and financing under uncertainty. Comput Oper Res 13(5):601–609
16. Huchzermeier A, Cohen MA (1996) Valuing operational flexibility under exchange rate risk. Oper Res 44(1):100–113
17. Kogut B, Kulatilaka N (1994) Operating flexibility, global manufacturing, and the option value of a multinational network. Managem Sci 40(1):123–139
18. Kouvelis P, Sinha V (1995) Exchange rates and the choice of production strategies: Supplying Foreign Markets. Duke Univ., Durham, NC
19. Tayur S, Ganeshan R, Magazine M (1998) Quantitative models for supply chain management. Kluwer, Dordrecht
20. van Houtum GJ, Inderfurth K, Zijm WHM (1996) Materials coordination in stochastic multi-echelon systems. Europ J Oper Res 95:1–23
21. Vidal CJ, Goetschalckx M (1997) Strategic production-distribution models: A critical review with emphasis on global supply chain models. Europ J Oper Res 98:1–18

Global Terrain Methods

ANGELO LUCIA
Department of Chemical Engineering,
University of Rhode Island, Kingston, USA

Article Outline

Introduction
Formulation
  Problem Statement
  Geometrical Foundation
Methods
  1) Moving Downhill
  2) Acceleration to Singular Points
  3) Moving Uphill
  4) Eigenvalue-Eigenvector Computations
  5) Effective Bookkeeping
  6) Termination
  7) Advanced Techniques
  Parametric Disconnectedness
  Non-differentiable Points and Manifolds
  Integral Path Bifurcations
  Gauss Curvature
  Finding Integral Path Tangent Bifurcations
  Finding Integral Path Pitchfork Bifurcations
  Finding All Branches Associated with a Bifurcation Point
Cases
  1) Nonlinear Objective Functions with Simple Bounds
  2) Systems of Nonlinear Algebraic Equations
  3) Nonlinear Objective Functions with Simple Bounds and Linear Constraints
References

Introduction

Global terrain methods [5,6,7] are a class of methods for solving nonlinear programming problems that are based on the simple concept of intelligently following valleys up and down on the terrain or landscape of three times continuously differentiable or C^3 objective function surfaces. They belong to the class of integral path or path following methods [1,2,3,4,8] and can also be used to solve systems of nonlinear equations formulated as nonlinear least-squares problems. The overall approach is based on the reliable and efficient computation of minima, saddle points, and singular points and a terrain-following algorithm to efficiently move from one stationary point to another or to a boundary of the feasible region. What makes global terrain methods superior to other path following methods is the Newton-based *predictor-corrector* method used to move uphill on the objective function landscape.

Formulation

The problem under consideration is that of finding a number of minima, saddle points, and singular points of a C^3 objective function, $\phi = \phi(z)$, defined on R^n subject to bounds on variables, $c(z)$, where z are the optimization variables. Let $F = F(z)$ denote the gradient of φ and $J(z)$ denote the $n \times n$ symmetric Jacobian matrix of F (or Hessian matrix of φ).

Problem Statement

The problem can be stated in the form

$$\text{Find } \{z_k^*\} : z_k^* \leq c(z^*) \text{ such that } \nabla(F^T F) = 0\,, \quad (1)$$

where $\{z_k^*\}$ denotes a set of minima, saddle points, and/or singular points, and the constraints are given by

$$-z_i^* \le z_i^L \text{ and } z_i^* \le z_i^U \,, \tag{2}$$

where z_i^L and z_i^U are the lower and upper bounds on the variable z_i.

Note that $\nabla(F^T F) = J^T F = 0$ implies that either

$$F(z_k^*) = 0 \,, \tag{3}$$

$$\det J = 0 \text{ with null space vector } F \neq 0 \,. \tag{4}$$

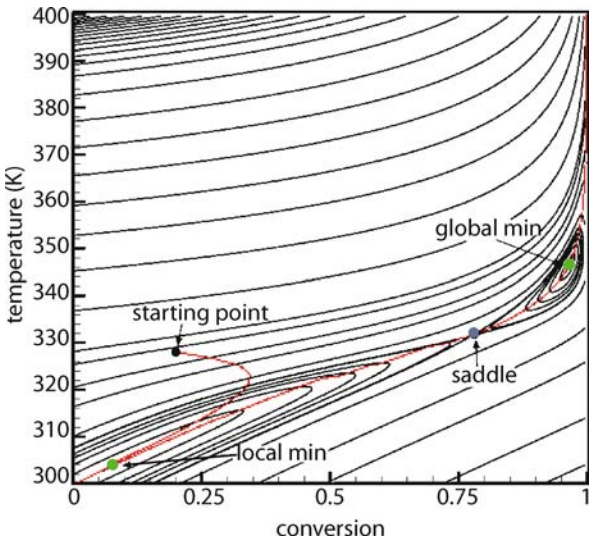
If $z_k^* *$ satisfies Eq. (3), it is either a minimum or a saddle point of φ whereas if z_k^* satisfies Eq. (4), it is a singular point of J. To distinguish between minima and saddle points, the Hessian matrix of $F^T F$ is required, which is

$$H = J^T J + \Sigma F_i G_i \,, \tag{5}$$

where F_i is the ith element function of F and where G_i is the corresponding element Hessian matrix of F_i. If all eigenvalues of H are positive, z_k^* is a minimum of φ. If at least one eigenvalue of H is negative, z_k^* is a saddle point of φ.

Geometrical Foundation

Figure 1 shows the contours of $F^T F$ along with the terrain path for a simple two dimensional reactor example. To understand the underlying geometric foundation on which global terrain methods are built, consider two neighboring contours or level curves along the curved valley shown in Fig. 1. Note that the distance, Δ, between any two neighboring level curves in the normalized gradient direction is largest exactly in the valley and that this distance decreases in magnitude as points move out of the valley along the same neighboring level curves (i. e., the contours become more tightly packed together). Therefore the norm of $J^T F$ must be smaller at any point in the valley than at any neighboring point on any given level curve since the same change in the least-squares function results from the largest change in distance. Thus the valley connecting the stationary points shown in Fig. 1 can be characterized as the collection of local minima in the norm of $J^T F$ over a set of level curves. This same constrained extremum in the gradient norm also characterizes ridges, ledges and other distinct features of the objective function landscape in any n-dimensional space.



Global Terrain Methods, Figure 1
Contours of a least squares surface

Valleys, ridges, ledge, etc. can be defined mathematically by a set of solutions, V, to a sequence of general nonlinearly, constrained optimization problems

$$V = \{\min g^T g \text{ such that } F^T F = L \,, \text{ for all } L \in \Lambda\} \,, \tag{6}$$

where F and J are defined as before and where $g = 2J^T F$, L is any given value (or level) of the least-squares objective function, and Λ is some collection of contours. That is, for any given level curve, we find the point on L that corresponds to a local minimum in $g^T g$. The collection of minima for all levels gives all (or part) of a valley, ridge, or ledge. Equation (6) forms the geometrical backbone for global terrain methods and plays an important role in the development of predictor-corrector algorithms used to implement those ideas. Moreover, Λ is actually a computational by-product of the terrain-following approach.

It is useful to simultaneously monitor behavior on the landscape of $F^T F$ and the objective function landscape, noting that minima and saddle points on φ are minima on $F^T F$ while singular points on φ are saddle points on $F^T F$. Valleys on both surfaces closely align.

Methods

Terrain-following methods are comprised of a sequence of sub-problems that unfold dynamically dur-

ing the course of solving a nonlinear programming problem. Since global terrain methods move up and down the landscape of $F^T F$, these sub-problems include

1) Reliable downhill equation solving.
2) Reliable and efficient computation of singular points.
3) Efficient uphill movement comprised of predictor-corrector calculations.
4) Reliable and efficient eigenvalue-eigenvector computations.
5) Effective bookkeeping.
6) A termination criterion to decide when the computations have finished.
7) Advanced techniques to deal with bifurcations and non-differentiable points.

1) Moving Downhill

Downhill computations use a trust region method and are capable of finding minima, saddle points, and sometimes singular points on an objective function surface. In finding the first point, say z_1^*, initiation of downhill computations is arbitrary. On subsequent downhill sub-problems, calculations always begin in the direction of the smallest negative eigenvalue of H.

The basic downhill iteration is defined as follows

$$\Delta = -\beta \Delta_N + (\beta - 1) g \,, \tag{7}$$

where $\Delta_N = J^{-1} F$ is the Newton direction and $\beta \in [0, 1]$ is determined by the following simple rules. If $||\Delta_N|| \leq R$, then $\beta = 1$, where R is the trust region radius. If $||\Delta_N|| > R$ and $||F|| \geq R$, then $\beta = 0$. Otherwise, β is the unique value in Eq. (7) on [0,1] that satisfies $||\Delta|| = R$. The new iterate is accepted if it reduces $||F||$. Otherwise, the new iterate is rejected, the trust region radius is reduced and the calculations are repeated until a reduction in $||F||$ occurs. Downhill movement is terminated when either $||F|| \leq \varepsilon$, where ε is a convergence tolerance, or $||F|| \,/\, ||\Delta_N|| \leq \zeta$, where ζ is some small number (typically 10^{-6}). This latter condition implies that the Newton step is very large in comparison to the gradient and the computations are converging to a singular point. The algorithm then switches to quadratic acceleration.

2) Acceleration to Singular Points

During downhill movement, quadratic acceleration is used if $||F||/||\Delta_N|| \leq \zeta$. Quadratic acceleration is also used during uphill calculations to converge to singular points and is defined by

$$\Delta = -H^{-1} J^T F \,. \tag{8}$$

During acceleration, norm reduction in F is not enforced because H can have eigenvalues of mixed sign.

3) Moving Uphill

Uphill movement is initiated in the eigen-direction associated with the smallest positive eigenvalue of the Hessian matrix H and consists of two basic parts – Newton predictor steps and successive quadratic programming (SQP) corrector steps.

Uphill Predictor Steps Predictor steps follow a valley uphill but will 'drift' from the valley – as shown in the slight zigzag in the terrain path in Fig. 1, which shows this 'drift' (followed by corrector steps). Uphill Newton steps are defined by

$$\Delta_p = \alpha \Delta_N \,, \tag{9}$$

where $\Delta_N = J^{-1} F$ and the step size $\alpha \in (0, 1]$.

Uphill Corrector Steps Corrector steps (again see Fig. 1) are used intermittently to force iterates back to a valley and are invoked when the condition

$$\theta = 57.295 \arccos \left[\left(\Delta_N^T c \right) / (||\Delta_N|| \; ||v||) \right] \geq \Theta \,, \tag{10}$$

is satisfied, where v is the current estimate of the eigenvector associated with the smallest positive eigenvalue of H and Θ is 5 degrees. Corrector steps are formulated as

$$\min g^T g \text{ such that } F^T F = L \,, \tag{11}$$

where L is the current value of $F^T F$. Corrector steps are iterative and are considered converged when the necessary conditions

$$F^T F - L = 0 \,, \tag{12}$$

$$Hg - \lambda g = 0 \,, \tag{13}$$

are satisfied. Corrector steps are computed using a successive quadratic programming (SQP) method; however, other methods can be used for this purpose. The SQP formulation for the problem defined by Eq. (11) is given by

$$\min g^T H\Delta_c + \frac{1}{2}\Delta_c^T M\Delta_c \text{ such that } g^T\Delta_c = -(F^T F - L)\,, \tag{14}$$

where M is the Hessian matrix of the Lagrangian function. The Lagrangian function is defined by $L = g^T g - \lambda(F^T F - L)$, where λ is a Lagrange multiplier and where M is approximated by the rule $M = H^T H - \lambda H$.

4) Eigenvalue-Eigenvector Computations

It is not always necessary to find all eigenvalues and eigenvectors of H to decide whether to begin the next phase of the computations uphill or downhill – particularly for problems with large n. Often it is sufficient to compute a subset of eigenvalues and eigenvectors, which can be conveniently performed using the inverse power method. The inverse power method solves the inverse form of

$$Hv - \lambda v = 0\,, \tag{15}$$

by constructing the iteration

$$v_{k+1} = \lambda_k H^{-1} v_k\,, \tag{16}$$

$$\lambda_{k+1} = \frac{v_{k+1}^T v_{k+1}}{v_{k+1}^T H^{-1} v_{k+1}}\,, \tag{17}$$

where the calculations alternate between Eqs. (16) and (17) until $||v_{k+1} - \lambda_k H^{-1} v_k|| < \varepsilon$, where ε is some pre-specified tolerance. Note that an estimate of v is necessary to begin the inverse power method. Once the first eigenvalue, say λ_1, and its corresponding eigenvector, v_1, have been determined, the Hessian matrix is deflated using symmetric orthonormal projection to give an $(n-1)\times(n-1)$ symmetric matrix whose basis spans the space orthogonal to v_1. The inverse power method is used to find the next eigenvalue, λ_2, and its associated eigenvector, v_2, and then v_2 is lifted to R^n. This procedure of deflation by orthonormal projection to form an $(n-j)\times(n-j)$ symmetric matrix whose basis spans the space orthogonal to $\{v_1, v_2, \ldots, v_j\}$ followed by the inverse power method and the lifting of v_{j+1} to R^n is continued until as many eigenvalues and eigenvectors as desired are determined.

5) Effective Bookkeeping

Another important aspect of global terrain methods is that it is possible to avoid calculating the same z_k^* more than once by effective bookkeeping. This is accomplished by storing solution information that includes the set of solutions, the solution types (i. e., minimum, saddle point, or singular point), corresponding values of φ and $F^T F$, and the current set of eigenconnections (i. e., the smallest positive eigenvalue and associated eigenvector for minima and saddles, and the largest negative eigenvalue and associated eigenvector for singular points). Following the determination of the first stationary or singular point, z_1^*, uphill movement proceeds in the +/− eigen-direction associated with the smallest positive eigenvalue of H. Assume that two new stationary or singular points, z_2^* and z_3^*, have been determined by these uphill calculations. The next move will be downhill from z_2^* in the eigen-directions, v_2, associated with the largest negative eigenvalue, λ_2, of H at z_2^*. However since z_2^* and z_3^* are connected by path to z_1^*, care must be exercised so as not follow the path back to z_1^*. To do this, nearest neighbors are determined by finding k such that

$$||z_2^* - z_k^*|| \text{ is minimum for all } k \neq 2\,. \tag{18}$$

Let j be the index for which Eq. (18) is satisfied. Following this, the direction $d_2 = z_2^* - z_j^*$ is defined. Correct downhill movement away from z_2^* is defined by whichever inequality

$$v_2^T d_2 < 0 \quad \text{or} \quad -v_2^T d_2 < 0\,, \tag{19}$$

is satisfied. Note that the selection of the proper condition in Eq. (19) guarantees that initial movement from z_2^* will be in the direction away from the nearest solution z_j^*. Equations (18) and (19) can be easily generalized to give

$$||z_i^* - z_k^*|| \text{ is minimum for all } k \neq K\,, \tag{20}$$

$$v_i^T d_i < 0 \quad \text{or} \quad -v_i^T d_i < 0\,, \tag{21}$$

where $d_2 = z_i^* - z_j^*$, j is the index that satisfies Eq. (20), and K is the current number of solutions.

6) Termination

Termination occurs when either the desired number of points $\{z_k^*\}$ have been calculated or a certain number of bounds are encountered. The first termination criterion is straightforward. In the normal case, termination occurs when two bounds have been encountered. When bifurcations have been detected, then the number of bounds that must be encountered for termination to occur is $n_b + 2$, where n_b is the number of distinct bifurcations.

7) Advanced Techniques

For any global terrain method to be effective it must also to address issues such as parametric disconnectedness, integral path bifurcations, and non-differentiable points or manifolds.

Parametric Disconnectedness

Following solutions parametrically is the basis for many homotopy-continuation methods. However, when parametric solutions exist on disconnected branches of solution curves, continuation methods can have difficulties. Global terrain methods are completely unaffected by parametric disconnectedness since they operate in variable and not parameter space.

Non-differentiable Points and Manifolds

There are many engineering applications that exhibit non-differentiable points and/or manifolds as a consequence of inherent switching contained in the objective function. At the 'switch' points, non-differentiability can occur and there can be families of 'switch' points that form manifolds. Non-differentiable points or manifolds are easily detected because they often exhibit retrograde curvature as well as other qualitative changes in model behavior that can be readily monitored.

Figure 2 illustrates a case in which there is a non-differentiable manifold. In this figure, $z_1 = C_{10}, z_2 = C_{18}, z_3 = C_{21}, z_1+z_2+z_3 = 1$, which is why the feasible region is triangular shaped, and $0 \le z_i \le 1, i = 1, 2, 3$. This curved manifold of non-differentiable points denotes the boundary between qualitatively different types of behavior for the case where $\phi = \min[\phi_1, \phi_2]$ at each z and is usually not mentioned in discussions of optimization of physical models. However, it is important in computations. The global terrain methodology has no difficulties finding stationary and singular points on $F^T F$ in this case because it monitors all aspects of the φ thereby allowing switching take place on the fly and the correct stationary and singular points to be easily found.

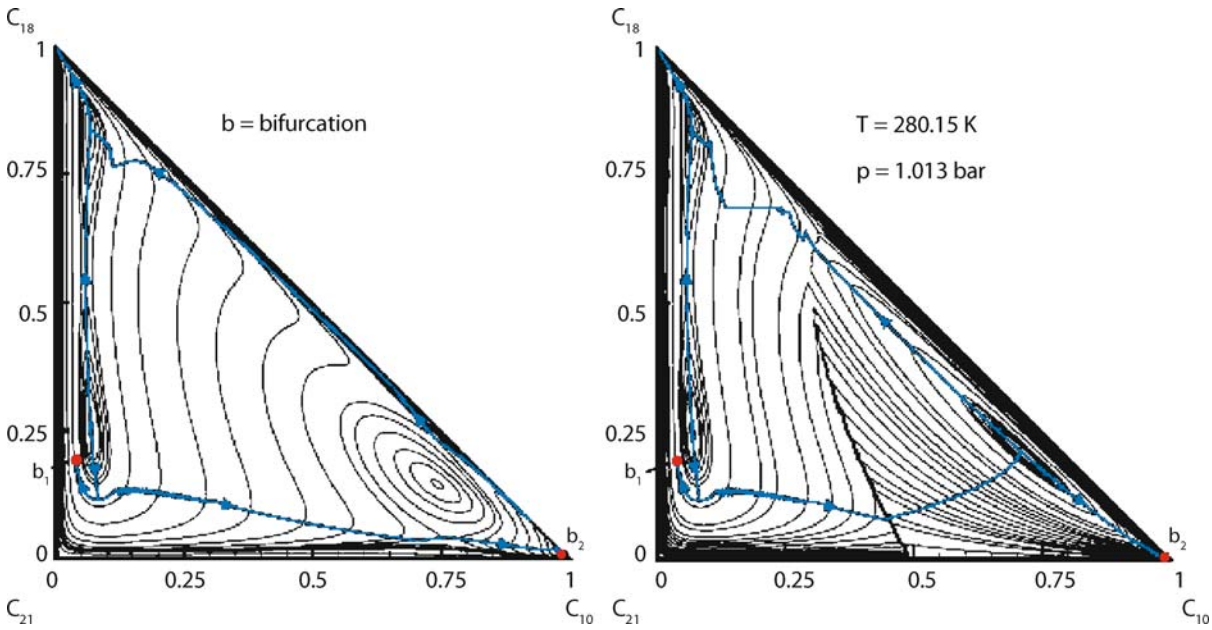
Integral Path Bifurcations

There are many applications in which integral paths either split into two or become tangent to a contour. These occurrences are called integral path bifurcations and can significantly impact the reliability of global terrain methods. Fortunately, Gauss curvature can provide a deterministic measure of the presence of bifurcation points.

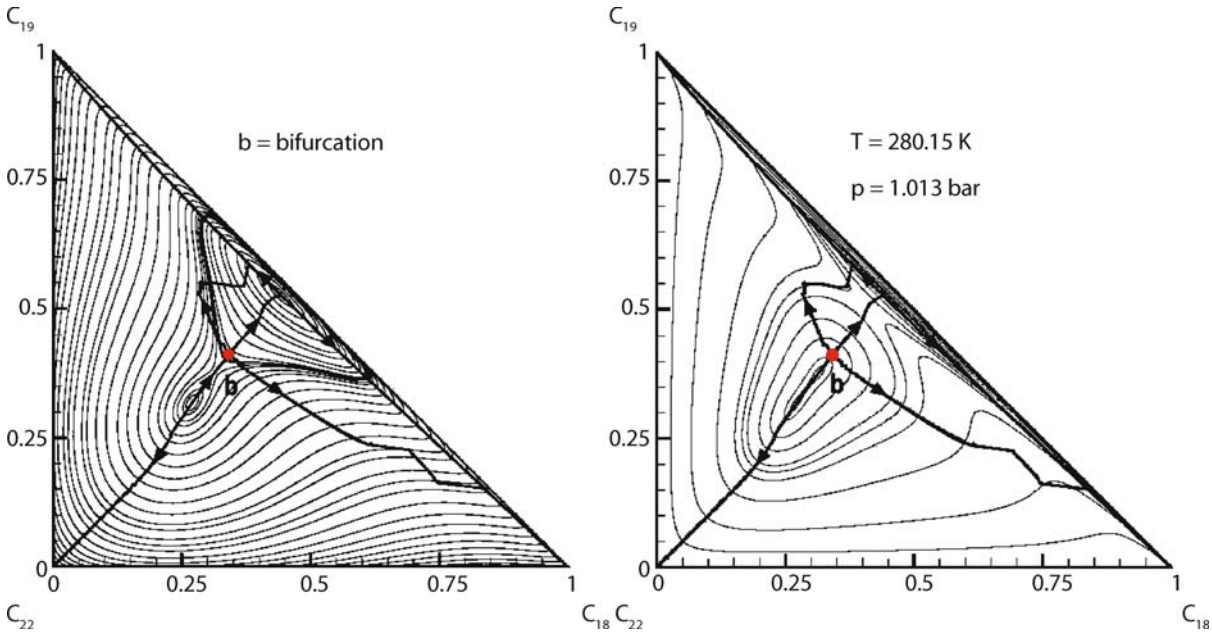
It is often easier to understand integral bifurcations from a geometrical perspective. Consider Fig. 3 where $z_1 = C_{18}, z_2 = C_{19}, z_3 = C_{22}, z_1 + z_2 + z_3 = 1$, and $0 \le z_i \le 1, i = 1, 2, 3$. Note that there is a pitchfork bifurcation at the point denoted by the point b on the integral path that runs from the two minima and the saddle point of $F^T F$ in the center of the triangle toward the saddle point and minimum very close to the hypotenuse of the triangular region. If the integral path bifurcation at b goes undetected, then the saddle point and minimum closest to the hypotenuse will not be found because corrector iterations will force iterates to turn toward the left or right hand branches of the pitchfork that end at the corners of the hypotenuse. Note, however, that the level curves begin to flatten in the neighborhood of the bifurcation point as the path moves toward the hypotenuse. This flattening, together with an eigenvector exchange from $J^T F$ to a vector in the tangent subspace of the level constraint, is a necessary condition for integral path *pitchfork* bifurcations, like the one that occurs at b. Moreover, flattening is relatively easy to measure by calculating (Gauss) curvature along a contour.

Gauss Curvature

To measure Gauss or Gauss–Kronecker curvature, it is necessary to calculate eigenvalues of the Hessian matrix, H, projected onto the tangent subspace of the level constraint, which is orthogonal to the gradient at any



Global Terrain Methods, Figure 2
An Objective Function & Gradient Surface with a Non-Differentiable Manifold (*left*) b ($F_1^T F_1$); (*right*) a Composite $F^T F$



Global Terrain Methods, Figure 3
Integral Path Bifurcation on Objective Function & Gradient Surfaces (*left*) Landscape of φ; (*right*) Landscape of $F^T F$

given point along the integral path. Gauss–Kronecker curvature corresponds to the determinant of this projected Hessian matrix. When the number of unknowns is two, this curvature is called Gauss curvature. Decreasing Gauss or Gauss–Kronecker curvature in a particular part of the feasible region indicates that the level curves are flattening and provides a strong reason to check for an exchange in the 'minimum' eigenvector of

H and, if warranted, to search for an integral path bifurcation.

Current implementation of these ideas measures flattening by calculating a few of the smallest eigenvalues (and eigenvectors) of the projection of H onto the tangent subspace *at each iteration* of the calculations. Without a theoretical basis that defines how often Gauss curvature should be measured, intermittent measurement seems ad hoc at best since very small regions of decreasing Gauss curvature could go undetected.

Finding Integral Path Tangent Bifurcations

This type of bifurcation point can be detected by measuring Gauss curvature and by comparing vectors along the flow of an integral path and vectors in the tangent subspace of the level sets for points on the path. When Gauss curvature decreases and the flow of the integral path becomes collinear to the tangent subspace, a tangent bifurcation point has occurred. Generally, this shows up as a 'jump' in the path to a point a considerable distance away on a neighboring level curve. Between these two points the value of the constrained minimum defining the path is degenerate and H has repeated eigenvalues.

Finding Integral Path Pitchfork Bifurcations

When flattening occurs but the flow of the integral path is not collinear to the tangent subspace of the level constraint, an eigenvalue exchange is sought. This exchange in the minimum eigenvalue of H from one associated with $J^T F$ to one associated with the tangent subspace of a level curve is easily determined by monitoring the eigenvalue associated with the terrain path and the smallest eigenvalue of the matrix H projected onto the tangent subspace. Once an eigenvalue exchange is detected, the algorithm searches for a possible bifurcation point by locating a maximum in the norm of $J^T F$ on the level curve, say L^*, where the eigenvalue exchange has been detected. This is because as contours flatten, the distance between these level curves becomes smaller and smaller, which is an indication that the nature of $||J^T F||$ on L^* has changed from a constrained minimum to a constrained maximum. See the discussion in [7]. Therefore, an approximate bifurcation point is calculated by solving the NLP problem

$$\max g^T g \text{ such that } F^T F = L^* . \tag{22}$$

Note that Eq. (22) is very similar to Eq. (11). Thus the numerical methodology needed to solve Eq. (22) already exists in the form of the corrector algorithm. However, it is important to note that predictor iterates rarely land exactly on the contour corresponding to a pitchfork bifurcation because finite step sizes are used in the predictor-corrector calculations. They generally land close and thus the solution to Eq. (22) is usually a very good approximation of the bifurcation point – since all that is really needed to follow all branches of a pitchfork bifurcation is knowledge at a point following the eigenvector exchange. Moreover, because contours in the neighborhood of a pitchfork bifurcation point can be very flat, solving Eq. (22) can be challenging in some cases. Extreme flatness creates numerical problems because it implies that the Kuhn–Tucker conditions for Eq. (22) have a near singular coefficient matrix. Therefore, good step size control should be used when solving Eq. (22).

Finding All Branches Associated with a Bifurcation Point

Once a bifurcation point is located, all branches from the bifurcation must be followed in order to increase the probability of finding all relevant solution information. Locating these branches is reasonably straightforward. Tangent bifurcation points are characterized by collinearity and provide only a single branch for further exploration that, as noted, manifests itself by a 'jump' to a widely different point on a neighboring level curve. Pitchfork bifurcation points, on the other hand, provide three branches of further exploration defined by the gradient to the level curve L^*, and $+/-$ the 'minimum' eigenvector of H projected onto the tangent subspace at the bifurcation on L^*. Each of these vectors is easily computed. The gradient vector at a bifurcation, which corresponds to the middle part of the pitchfork, is a readily available byproduct of the calculations. The 'minimum' eigenvector of H on the tangent subspace at L^* is also easily determined. What is difficult is locating the valleys that correspond to the pair of minima of $g^T g$ on L^*. For this a careful initialization of our corrector algorithm is required to solve Eq. (11) with $L = L^*$.

Cases

There are several problem cases that are encompassed by the global terrain-following formulations and methods presented in earlier sections. These cases include

1) Nonlinear objective functions with simple bounds on variables.
2) Systems of nonlinear algebraic equations.
3) Nonlinear objective functions with simple bounds and linear constraints.

1) Nonlinear Objective Functions with Simple Bounds

This is the case on which the developments in the formulation and methods sections are based and no further discussion is necessary.

2) Systems of Nonlinear Algebraic Equations

For a system of algebraic equations, $F = 0$ is usually given and φ is irrelevant. The function of interest becomes the traditional nonlinear least squares function, $F^T F$, and the terrain methodology follows the strategies outlined in previous sections.

3) Nonlinear Objective Functions with Simple Bounds and Linear Constraints

Nonlinear programming problems that involve linear equality constraints are easily handled by global terrain methods by using the linear constraints to eliminate optimization variables. For m linear constraints, m optimization variables can be eliminated. However, it is important to understand that the gradient and Hessian matrix of φ must be adjusted to accommodate this variable elimination. This can be done by either using projection methods or by explicitly doing the elimination before formulating the optimization problem to be solved by the terrain methodology.

If projection is used then F is replaced by $P^T F$, where P is the $n \times m$ orthonormal projection matrix whose columns are orthogonal to all rows of the Jacobian matrix of the linear constraints. That is, if J_{LEQ} is the $m \times n$ Jacobian of the m linear equality constraints, then the projection matrix P satisfies $J_{LEQ} P = 0$. Additionally, the Hessian matrix of ϕ, J, must reflect implicit elimination and is easily computed to be $P^T J P$. These projections of F and J permit the use of the terrain methodology in R^{n-m} while still allowing any bounds on all variables to be enforced.

References

1. Baker J (1986) An algorithm for the location of transition states. J Comput Chem 7:385–395
2. Cerjan CJ, Miller WH (1981) On finding transition states. J Chem Phys 75:2800–2806
3. Diener I (1987) On the global convergence of path-following methods to determine all solutions to a system of nonlinear equations. Math Prog 39:181–188
4. Jongen HT, Stein O (2004) Constrained global optimization: adaptive gradient flows. In: Floudas CA, Pardalos P (eds) Frontiers in Global Optimization. Kluwer Acad, Boston
5. Lucia A, DiMaggio PA, Depa P (2004) A geometric methodology for global optimization. J Global Optim 29:297–314
6. Lucia A, Yang F (2002) Global terrain methods. Comput Chem Eng 26:529–546
7. Lucia A, Yang F (2003) Multivariable terrain methods. AIChE J 49:2553–2563
8. Page M, McIver JW (1988) On evaluating the reaction path Hamiltonian. J Chem Phys 88:922–935

Graph Coloring

GC

Jue Xue
Department Management Sci.,
City University Hong Kong, Kowloon, Hong Kong

MSC2000: 90C35

Article Outline

Keywords
See also
References

Keywords

Graph; Coloring; Optimization; Approximation; Algorithms

A graph $G = (V, E)$ consists of a *vertex set* V and an *edge set* $E \subseteq V \times V$. If $e = (i, j)$ ($\in E$) is an edge of G, then e is *incident* to i and j, and i and j are *adjacent*. Similarly, if two edges are incident to the same vertex, they are adjacent.

A vertex coloring of $G = (V, E)$ is an assignment of k colors to members of V (a *coloring*) so that adjacent vertices have different colors (G is *k-colorable*). The *graph coloring problem* (GC) is to find the minimum number k such that G is k-colorable.

When a positive integer weight w_i is associated with every $i \in V$ and a color assignment satisfies:

- every vertex i gets w_i different colors,
- $\forall (i, j) \in E$, i and j get $w_i + w_j$ different colors,

then this color assignment is a *weighted coloring*. The *weighted graph coloring problem* asks for the minimum number of colors needed for a weighted coloring of G.

An *edge coloring* and a *total coloring* of a given graph can be defined in a similar way:

- An edge coloring assigns colors to edges so that adjacent edges have different colors.
- A total coloring assigns colors to vertices and edges so that any pair of adjacent vertices, adjacent edges, and a vertex and any incident edge will have different colors.

The *edge coloring problem* or the *total coloring problem* asks for the minimum number of colors needed for an edge coloring or a total coloring, respectively [12,28,30]. Although the weighted graph coloring, edge coloring, and total coloring problems seem different from GC, they can be transformed into a GC [33,42]. Further generalizations of GC tend to change the structure of a coloring solution, and they move closer to other well-known combinatorial optimization problems [16,37].

GC is well-known in graph theory and combinatorial optimization. It starts with the famous four-coloring conjecture [24,38] which says four colors are enough to color any geographic map so that every country gets a color different from those used by its neighbors. Although the four-coloring conjecture is now considered a theorem [1,2], the process to prove or disprove it has inspired many interesting questions [32], and has helped the development of several branches of science, for example, the GC and the graph theory [27]. The interest in GC also comes from its vast number of applications in solving real world problems. For example, GC can be used to model problems in timetabling, scheduling, computer science, information systems, telecommunications, and other industrial applications [9,11,39]. Typically, a graph is constructed with its vertices representing items of interest and edges representing some undesirable binary relationship.

GC has several mathematical programming formulations. For example, one can use an integer variable $x_{ik} = 1$ to indicate when a vertex i is colored by k, and $x_{ik} = 0$ otherwise. One can also use an integer variable $y_k = 1$ to indicate color k is assigned to at least one vertex of G, and $y_k = 0$ otherwise. Then, the solution to the following mathematical programming problem provides an optimal (minimum) coloring of G:

$$\begin{cases} \min & \displaystyle\sum_{k=1}^{|V|} y_k \\ \text{s.t.} & \displaystyle\sum_{k=1}^{|V|} x_{ik} = 1, \quad \forall i \in V, \\ & x_{ik} + x_{jk} \le 1, \quad \forall (i, j) \in E, \\ & y_k \ge x_{ik}, \quad y_k, x_{ik} \in \{0, 1\}, \\ & \quad \forall i \in V,\ k = 1, \ldots, |V|, \end{cases}$$

where $|V|$ is the cardinality of the set V. In this problem, the objective function equals the number of colors used. The constraints ensure that every vertex is colored, that no adjacent vertices get the same color, and that the counting of used colors is correct.

For a feasible coloring, one can group the vertices into subsets based on their colors. Thus, vertices of each subset will be mutually nonadjacent. Such a subset of vertices is called a *stable set*, a *color class*, or an *independent set* [5,8,35,41]. Using the concept of a stable set, one can formulate GC as a set partitioning problem.

Let $S_1, \ldots, S_t$ be all the stable sets of G. Let A_S be a $0 - 1$ matrix whose rows are the characteristic vectors of the S_js. One can use a variable $s_j = 1$ to indicate that all members of S_j have the same color, and $s_j = 0$ otherwise. Then the solution to the following problem also provides an optimal (minimum) coloring of G:

$$\begin{cases} \min & \displaystyle\sum_{j=1}^{t} s_j \\ \text{s.t.} & sA_S = \vec{1}, \\ & s_j \in \{0, 1\}, \quad j = 1, \ldots, t, \end{cases}$$

where $s = (s_1, \ldots, s_t)$, and $\vec{1} = (1, \ldots, 1)$ is of dimension $|V|$.

Other mathematical formulations of GC based on quadratic programming, semidefinite programming etc. are also available. Different formulations have their own distinctive advantages in understanding the problem structure and in designing solution methods to solve the problem [22,33,34].

Checking whether G is k-colorable for an arbitrary integer k is an *NP*-complete problem [14,23]. It remains *NP*-complete even for fixed $k \geq 3$ [14,40]. Therefore, it is unlikely that the solution time of GC can be bounded by any polynomial function (*polynomial time*) [13]. However, GC can be solved in polynomial time for graphs of some special structures. For example, polynomial algorithms exist for perfect graphs, Meyniel graphs, and triangulated graphs [3,4,15,17,19,41].

Let us define the *performance guarantee* of an approximation method to be the worst ratio between the approximation solution value and the corresponding optimal solution value over all graphs of size $|V|$. Then, $O(|V| \log |V|)$ seems to be the first performance guarantee provided by a polynomial time GC heuristic [20]. This performance guarantee has being improved over the years. Let k be the optimal (minimum) number of colors needed to color a graph, and let Δ be the *maximum degree* (number of edges incident to a vertex) among all vertices. The two recent performance guarantees achieved by polynomial approximation algorithms for GC are $O(|V|(\log \log |V|)^2/(\log |V|)^3)$ and $\min\{O(\Delta^{1-2/k}), O(|V|^{1-3/(k+1)})\}$ [18,22]. On the other hand, it is known that unless $P = NP$, it is *NP*-hard to approximate an optimal graph coloring within a performance guarantee of $O(|V|^{\epsilon})$, $\epsilon > 0$ [14,15].

Available solution methods for GC can be divided into approximation algorithms and exact algorithms. These methods find a feasible graph coloring and an optimal graph coloring, respectively [29].

A popular way to find an approximation solution to GC is the *sequential greedy coloring heuristic* (SGCH). In a SGCH, the vertices are ordered in a sequence and are colored one at a time according to the sequence. Every vertex is colored by the smallest (first) feasible color. It is not hard to see that the initial vertex sequence decides the resulting graph coloring of a SGCH.

It is also known that there exists at least one sequence under which a SGCH will find an optimal coloring. However, finding an optimal vertex sequence is *NP*-hard. Extensive work aimed at finding 'good' vertex sequences can be found in the literature [10,32]. Once a feasible coloring is available, further improvement can be made using various methods, including: interchange, iterative improvement, and other searching techniques (such as simulated annealing and tabu search) [36].

To date, the most popular and efficient way to find an optimal solution to GC is through a *branch and bound* (BB), or *implicit enumeration*, algorithm. A BB algorithm typically consists of two parts: the *forward phases* and the *backtrack phases*. A forward phase starts from a partial coloring (e. g. $\emptyset$) and colors the remaining vertices to find a feasible graph coloring. For example, a SGCH can be used in place of a forward phase. A backtrack phase will decide the starting point of the next forward phase so that an alternative feasible graph coloring can be found.

Now let us consider how a simple BB algorithm [7] finds an optimal coloring of $G = (V, E)$. Let UB be the value of a current best coloring (initially set $UB = \infty$). Suppose the first forward phase applies a SGCH to vertex sequence $(v_1, \ldots, v_{|V|})$ and finds a feasible coloring of G. The number of colors used by the feasible coloring will be the new UB. Apparently, UB is an upper bound on the value of any feasible coloring that one needs to search for.

Since SGCH assigns the smallest feasible color to every vertex, a backtrack phase can be carried out by scanning the vertices in the reverse order of $(v_1, \ldots, v_{|V|})$. That is, finding the first vertex v_j that can be recolored by an alternative feasible color $< UB$, not used for v_j before. The new forward phase will start from the partial coloring of $\{v_1, \ldots, v_{j-1}\}$ and applies a SGCH to $(v_j, \ldots, v_{|V|})$, up to a v_i whose smallest feasible color is UB, or to $v_{|V|}$ that has a feasible color $< UB$. In the latter case, a better coloring is found. Then the BB algorithm will backtrack and repeat the above until it backtracks to vertex v_1 (the algorithm terminates).

Various improvement measurements are designed and tested for the above basic BB method. They include 'look ahead', 'dynamic reordering', choosing an appropriate feasible color (instead of the 'smallest') to color a vertex, using tighter lower and upper bounds, and a column generation approach [6,21,26,31,33]. These improvements have greatly reduced the search tree size and enhanced our ability to solve GC optimally. The

state-of-the-art method for solving GC on randomly generated graphs seems to be limited to graphs of 100 vertices [21,31,42,43].

See also

- Adaptive Simulated Annealing and its Application to Protein Folding
- Branch and Price: Integer Programming with Column Generation
- Decomposition Techniques for MILP: Lagrangian Relaxation
- Feedback Set Problems
- Frequency Assignment Problem
- Generalized Assignment Problem
- Genetic Algorithms
- Global Optimization in Lennard–Jones and Morse Clusters
- Global Optimization in Protein Folding
- Graph Planarization
- Greedy Randomized Adaptive Search Procedures
- Heuristics for Maximum Clique and Independent Set
- Integer Linear Complementary Problem
- Integer Programming
- Integer Programming: Algebraic Methods
- Integer Programming: Branch and Bound Methods
- Integer Programming: Branch and Cut Algorithms
- Integer Programming: Cutting Plane Algorithms
- Integer Programming Duality
- Integer Programming: Lagrangian Relaxation
- LCP: Pardalos–Rosen Mixed Integer Formulation
- Maximum Constraint Satisfaction: Relaxations and Upper Bounds
- Mixed Integer Classification Problems
- Molecular Structure Determination: Convex Global Underestimation
- Monte-Carlo Simulated Annealing in Protein Folding
- Multi-objective Integer Linear Programming
- Multi-objective Mixed Integer Programming
- Multiparametric Mixed Integer Linear Programming
- Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach
- Packet Annealing
- Parametric Mixed Integer Nonlinear Optimization
- Phase Problem in X-ray Crystallography: Shake and Bake Approach
- Protein Folding: Generalized-ensemble Algorithms
- Quadratic Assignment Problem
- Quadratic Semi-assignment Problem
- Set Covering, Packing and Partitioning Problems
- Simplicial Pivoting Algorithms for Integer Programming
- Simulated Annealing Methods in Protein Folding
- Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- Stochastic Integer Programs
- Time-dependent Traveling Salesman Problem

References

1. Appel K, Haken W (1977) Every planar map is four colorable. Part 1: Discharging. Illinois J Math 21:429–490
2. Appel K, Haken W (1977) Every planar map is four colorable. Part 2: Reducibility. Illinois J Math 21:491–567
3. Balas E (1986) A fast algorithm for finding an edge-maximal subgraph with a TR-formative coloring. Discrete Appl Math 15:123–134
4. Balas E, Xue J (1991) Minimum weighted coloring of triangulated graphs with application to maximum weight vertex packing and clique finding in arbitrary graphs. SIAM J Comput 20:209–221
5. Balas E, Xue J (1996) Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. Algorithmica 15:397–412
6. Brelaz D (1979) New methods to color vertices of a graph. Comm ACM 22:251–256
7. Brown JR (1972) Chromatic scheduling and the chromatic number problem. Managem Sci 19:456–463
8. Carraghan R, Pardalos PM (1990) An exact algorithm for the maximum clique problem. Oper Res Lett 9:375–382
9. de Werra D (1985) An introduction to timetabling. Europ J Oper Res 19:151–162
10. de Werra D (1990) Heuristics for graph coloring. Computing 7:191–208
11. de Werra D, Gay Y (1994) Chromatic scheduling and frequency assignment. Discrete Appl Math 49:165–174
12. Fiorini S, Wilson RJ (1977) Edge-coloring of graphs. Res Notes Math, vol 16. Pitman, Boston, MA
13. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, New York
14. Garey MR, Johnson DS, Stockmeyer I (1976) Some simplified NP-complete graph problems. Theoret Comput Sci 1:237–267
15. Gavril F (1972) Algorithms for coloring maximum clique: minimum covering by cliques, and maximum independent set of a chordal graph. SIAM J Appl Math 1:181–187

16. Gionfriddo M (1979) A short survey of some generalized colorings of graphs. Ars Combin 21:295–322
17. Grötschel M, Lovász L, Schrijver A (1989) Polynomial algorithms for perfect graphs. Ann Discret Math 21:325–356
18. Halldórsson MM (1993) A still better performance guarantee for approximate graph coloring. Inform Process Lett 45:19–23
19. Hertz A (1990) A fast algorithm for coloring Meyniel graphs. J Combin Th B 50:231–240
20. Johnson DS (1974) Worst-case behavior of graph-coloring algorithms. In: Proc. 5th Southeastern Conf. Combinatorics: Graph Theory and Computing, Winnipeg, pp 513–528
21. Johnson DS, Trick M (eds) (1996) Cliques, coloring, and satisfiability: Second DIMACS implementation challenge. DIMACS. Amer. Math. Soc., Providence, RI
22. Karger D, Motwani R, Sudan M (1994) Approximate graph coloring by semidefinite programming. In: 35th Annual Symp. Foundations of Computer Sci., IEEE, New York, pp 2–13
23. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations, Plenum, New York, pp 85–104
24. Kempe AB (1879) On the geographical problem of four colours. Amer J Math 2:193–200
25. Khanna S, Linial N, Safra S (1993) On the hardness of approximating the chromatic number. Proc. 2nd Israel Symp. Theory of Computing and Systems, IEEE, New York, pp 250–260
26. Korman SM (1979) The graph-colouring problem. In: Christofides N, Mingozzi A, Toth P, Sandi C (eds) Combinatorial Optimization. Wiley, New York, pp 211–235
27. Kubale M (1991) Graph coloring. In: Kent A, Williams JG (eds) Encycl. Microcomputers, vol 8. M. Dekker, New York, pp 47–69
28. Lee J, Leung J (1993) A comparison of two edge-coloring formulations. Oper Res Lett 13:215–223
29. Matula DW, Marble G, Isaacson D (1972) Graph coloring algorithms. In: Reed R (ed) Graph theory and computing. Academic Press, New York, pp 109–122
30. McDiarmid C, Sanchezarroyo A (1993) On total colorings of graphs. J Combin Th B 57:122–130
31. Mehrotra A, Trick MA (1995) A column generation approach for graph coloring. Techn Report GSIA Carnegie-Mellon Univ
32. Nelson R, Wilson RJ (eds) (1990) Graph colorings. Longman, Harlow
33. Pardalos PM, Mavridou T, Xue J (1998) The graph coloring problem: A bibliographic survey. In: Du D-Z and Pardalos PM (eds) Handbook Combinatorial Optim, vol 2. Kluwer, Dordrecht, pp 331–395
34. Pardalos PM, Wolkowicz H (1994) Quadratic assignment and related problems. DIMACS. Amer. Math. Soc., Providence, RI
35. Pardalos PM, Xue J (1994) The maximum clique problem. J Global Optim 3:463–482
36. Rayward-Smith VJ, Osman IH, Reeves CR, Smith GD (eds) (1996) Modern heuristic search methods. Wiley, New York
37. Roberts FS (1995) From garbage to rainbows: generalizations of graph coloring and their applications. In: Alavi Y, Schwenk AJ (eds) Graph Theory, Combinatorics, and Algorithms, vol 2. Wiley, New York, pp 1031–1052
38. Saaty TL, Keinen PC (1977) The four-color problem. McGraw-Hill, New York
39. Schmidt G, Ströhleim T (1980) Timetable construction - an annotated bibliography. Comput J 23:307–316
40. Stockmeyer L (1973) Planar 3-colorability is polynomial complete. ACM SIBACT News 5:19–25
41. Xue J (1994) Edge-maximal triangulated subgraphs and heuristics for the maximum clique problem. Networks 24:109–120
42. Xue J (1998) Solving the minimum weighted integer coloring problem. J Comput Optim Appl 11:53–64
43. Xue J, Liu J A network flow based lower bound for the minimum weighted integer coloring problem. Inform Process Lett (to appear).

Graph Planarization

MAURICIO G.C. RESENDE[1], CELSO C. RIBEIRO[2]
[1] Information Sci. Res., AT&T Labs Res., Florham Park, USA
[2] Department Computer Sci., Catholic University Rio de Janeiro, Rio de Janeiro, Brazil

MSC2000: 94C15, 90C10, 90C27

Article Outline

Keywords
Variants and Applications
An Exact Algorithm
Heuristics Based on Planarity Testing
Two-Phase Heuristics
Computational Results
See also
References

Keywords

Planarization; Graph; Planar graph; Heuristics; Exact algorithms

A graph is said to be *planar* if it can be drawn on the plane in such a way that no two of its edges cross. Given

a graph $G = (V, E)$ with vertex set V and edge set E, the objective of *graph planarization* is to find a minimum cardinality subset of edges $F \subseteq E$ such that the graph $G' = (V, E \setminus F)$, resulting from the removal of the edges in F from G, is planar. This problem is also known as the *maximum planar subgraph* problem. A related and simpler problem is that of finding a *maximal planar subgraph*, which is a planar subgraph $G' = (V, E')$ of G such that the addition of any edge $e \in E \setminus E'$ to G' destroys its planarity.

Graph planarization is known to be *NP*-hard [21]. The proof of *NP*-completeness of its decision version is based on a transformation from the Hamiltonian path problem restricted to bipartite graphs. Although exact methods for solving the maximum planar subgraph problem have been recently proposed, most algorithms to date attempt to find good approximate solutions.

In this article, we survey graph planarization and related problems. In the next section, we describe variants and applications of the basic problem formulated above. Next, we describe the branch and cut algorithm of M. Jünger and P. Mutzel [16]. We then review work on heuristics based on planarity testing and those based on two- phase procedures. Finally, computational results are considered.

Variants and Applications

An application of graph planarization arises in the design of integrated circuits, in which a graph describing the circuit has to be decomposed into a minimum number of layers, each of which is a planar graph [19]. Other applications arise from variants of the basic graph planarization problem.

One such variant is the *maximum weighted planar graph* problem, in which positive weights are associated with the edges of the graph and one seeks a planar subgraph of maximum weight. Note that the basic graph planarization problem is a special case of the maximum weighted planar graph problem, in which all edge weights are equal to one. An application of this problem to *facility layout* is described in [13]. A graph is built in which the vertices represent the facilities and the edges define the relationships between them. The weight of each edge is the desirability that the two facilities that define the edge be adjacent in the design. A maximum weighted planar subgraph corresponds to a feasible layout with maximum benefit. In this paper, the authors also propose simulated annealing and tabu search heuristics for the approximate solution of the maximum weighted planar graph problem. Constructive heuristics based on maintaining a triangulated subgraph while making node and edge insertions are given in [8,11], and [20].

Another related variant is that of drawing a given graph such that the number of *edge crossings* is minimized. The *crossing number* problem has practical applications in *circuit design* and graph drawing, such as in CASE tools [27] and automated graphical display systems. One particular case is that of minimizing straight-line crossings in layered graphs. A GRASP and path relinking approach for the two-layer case is given in [17], where one can also find a survey of the literature. Algorithms for graph drawing are reviewed in [6].

In the *planar augmentation* problem, one wants to determine the minimum number of edges that need to be added to a planar graph such that the resulting graph is still planar and at least k-connected, where k is usually fixed to two or three. This variant has applications in automatic *graph drawing*, as well as in the design of *survivable networks* [24].

An Exact Algorithm

An exact branch and bound algorithm for the weighted graph planarization problem was introduced in [10], but was limited to small dense graphs. Only recently (1999) has there been a leap in the performance of exact methods for graph planarization with the *Jünger–Mutzel branch and cut algorithm* [16], which we describe next.

Given a graph $G = (V, E)$, their approach uses facet-defining inequalities for the planar subgraph polytope $\mathcal{PLS}(G)$. Let x_e be a 0–1 variable associated with each edge $e \in E$, such that $x_e = 1$ if and only if edge e appears in the maximum planar subgraph of G. Furthermore, let $x(F) = \sum_{e \in F} x_e$, for $F \subseteq E$.

Trivial inequalities $0 \leq x_e \leq 1$ are implicitly handled by the linear programming (LP) solver. The inequality $x(E) \leq 3|V| - 6$ is added to the initial linear program. Let x be the optimal solution of the LP relaxation associated with some node of the enumeration tree. For $0 \leq \epsilon \leq 1$, let $E_\epsilon = \{e \in E\}\ x_e \geq 1 - \epsilon\}$ and consider the graph $G_\epsilon = (V, E_\epsilon)$, to which the

Hopcroft–Tarjan planarity-testing algorithm [14] is applied. The algorithm stops if it finds an edge set F which induces a nonplanar graph in G. If the inequality $x(F) \leq |F| - 1$ is violated, it is added to the set of constraints of the current LP. The back edge of the path which proved the nonplanarity of the graph induced in G by F is removed and the planarity-testing algorithm proceeds, eventually identifying other forbidden subgraphs of the graph G_ϵ. Although these forbidden subgraphs do not necessarily define facets of $\mathcal{PLS}(G)$, they must contain facet-defining subgraphs. Facet-defining inequalities are identified as follows. Once a forbidden set F is found, where the inequality $x(F) \leq |F| - 1$ is violated, one successively deletes each edge $f \in F$ and applies the planarity-testing algorithm. If the graph induced by $F \setminus \{f\}$ is planar, then edge f is returned to F. In at most $|F|$ steps, F is reduced to a smaller edge set which induces a minimal planar subgraph, leading to the facet-defining inequality $x(F) \leq |F| - 1$ still violated by the current LP solution. Another simple heuristic searches for violated Euler facet-defining inequalities $x(F) \leq 3|V'| - 6$ or $x(F) \leq 2|V'| - 4$, where (V', F) is, respectively, a clique or a complete bipartite subgraph of G.

After an LP has been solved, its solution is exploited by the planarity-testing algorithm, to produce a feasible solution for the graph planarization problem. Such feasible solutions are used as lower bounds that are used not only for fathoming nodes in the branch and cut tree, but also for fixing variables using their reduced costs during a cutting plane phase. Other heuristics are implemented to enhance the practical performance of the algorithm.

Branching is done if no cutting plane has been found for the current infeasible solution. The variable chosen for branching is one with fractional value closest to 1/2, among those with maximum cost coefficient in the objective function.

Heuristics Based on Planarity Testing

The first linear time algorithm for planarity testing was proposed by J. Hopcroft and R.E. Tarjan [14]. T. Chiba, I. Nishioka and I. Shirakawa [4] used the basic ideas of this approach to devise an algorithm for finding a maximal planar subgraph of $G = (V, E)$ with time complexity $O(|V||E|)$. Later, J. Cai, X. Han and Tarjan [3] proposed another version of the above planarity testing algorithm. This new algorithm is based on processing edges instead of paths. It leads to another algorithm to find a maximal planar subgraph, with improved $O(|E| \log |V|)$ time complexity.

A. Lempel, S. Even and I. Cederbaum [18] have proposed another approach to planarity testing. Although its original complexity was $O(|V|^2)$, K. Booth and G. Lueker [2] have shown that it can be implemented in linear time using PQ-trees. A few algorithms for finding a maximal planar subgraph based on this planarity testing approach have been proposed in the literature. However, Jünger, S. Leipert and Mutzel [15] show that attempts following this strategy are forced to fail.

Another approach for finding a maximal planar subgraph of a given graph works as follows. Start with an empty subgraph and successively add the edges of the original graph, whenever such addition maintains the planarity of the subgraph under construction. Using any of the planarity testing algorithms above described, such approach can be implemented in $O(|V||E)$ time complexity. An incremental planarity testing algorithm, based on an $O(\log|V)$ time-per-operation strategy for the problem of maintaining a planar graph under edge additions, was proposed by G. Di Battista and R. Tamassia [7]. Hence, their algorithm leads to a more efficient implementation of the incremental approach for finding a maximal planar subgraph with $O(|E| \log |V|)$ time complexity.

Two-Phase Heuristics

The heuristics described in this section are based on the separation of the computation into two phases. The first phase consists in devising a linear permutation of the nodes of the input graph, followed by placing them along a line. The second phase determines two sets of edges that may be represented without crossings above and below that line, respectively. Y. Takefuji and K.C. Lee [25] were the first to propose a heuristic using this idea. They use an arbitrary sequence of nodes in the first phase and apply a parallel heuristic using a neural network for the second phase. Takefuji, Lee, and Y.B. Cho [26] claimed superior performance of the two-phase approach of Takefuji and Lee [25] with respect to the heuristics described in the previous section.

Their approach was later extended and improved by O. Goldschmidt and A. Takvorian [12]. In the first phase, these authors attempt to use a linear permutation of the nodes associated with an Hamiltonian cycle of G. Two strategies are used:

i) a randomized algorithm [1] that almost certainly finds a Hamiltonian cycle if one exists; and
ii) a greedy deterministic algorithm that seeks a Hamiltonian cycle.

In the latter, the first node in the linear permutation is a minimum degree node in G. After the first k nodes of the permutation have been determined, say $v_1, \ldots, v_k$, the next node v_{k+1} is selected from the nodes adjacent to v_k in G having the least adjacencies in the subgraph G_k of G induced by $V\backslash\{v_1, \ldots, v_k\}$. If there is no node of G_k adjacent to v_k in G, then v_{k+1} is selected as a minimum degree node in G_k.

Let $H = (E, I)$ be a graph where each of its nodes corresponds to an edge of the input graph G. Nodes e_1 and e_2 of H are connected by an edge if the corresponding edges of G cross with respect to the linear permutation of the nodes established during the first phase. A graph is called an *overlap graph* if its nodes can be placed in one-to-one correspondence with a family of intervals on a line. Two intervals are said to *overlap* if they cross and none is contained in the other. Two nodes of the overlap graph are connected by an edge if and only if their corresponding intervals overlap. Hence, the graph H as constructed above is the overlap graph associated with the representation of G defined by the linear permutation of its nodes.

The second phase of the heuristic of Goldschmidt and Takvorian consists in two-coloring a maximum number of the nodes of the overlap graph H, such that each of the two color classes $\mathcal{B}$ (blue) and $\mathcal{R}$ (red) forms an independent set. Equivalently, the second phase seeks a *maximum bipartite subgraph* of the overlap graph H, i. e. a bipartite subgraph having the largest number of nodes. This problem is equivalent to drawing the edges of the input graph G above or below the line where its nodes have been placed, according to their linear permutation. A greedy algorithm is used to construct a maximal bipartite subgraph of the overlap graph. This algorithm finds a maximum independent set $\mathcal{B} \subseteq E$ of the overlap graph $H = (E, I)$, reduces the overlap graph by removing from it the nodes in $\mathcal{B}$ and all edges incident to nodes in $\mathcal{B}$, and then finds a maximum independent set $\mathcal{R} \subseteq E\backslash\mathcal{B}$ in the remaining overlap graph $H' = (E\backslash\mathcal{B}, I')$. The two independent sets so obtained induce a bipartite subgraph of the original overlap graph, not necessarily with a maximum number of nodes.

The linear permutation obtained in the first phase affects the size of the planar subgraph found in the second phase of the above heuristic. Moreover, it is not clear that the permutation produced by the greedy algorithm is the best. To produce possibly better permutations, *randomization* and *local search* have been introduced in the *greedy algorithm* by M.G.C. Resende and C.C. Ribeiro [22] in the form of a *greedy randomized adaptive search procedure* (GRASP).

A GRASP [9] is an iterative process, in which each iteration consists of two phases: construction and local search. The construction phase builds a feasible solution, whose neighborhood is explored by local search. The best solution over all GRASP iterations is returned as the result.

In the construction phase, a feasible solution is built, one element at a time. At each construction iteration, the next element to be added is determined by ordering all elements in a candidate list with respect to a greedy function that estimates the benefit of selecting each element. The adaptive component of the heuristic arises from the fact that the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous elements. The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidates in the list, but usually not the top candidate. This way of making the choice allows for different solutions to be obtained at each iteration, but does not necessarily jeopardize the power of GRASP's adaptive greedy component.

The solutions generated by a GRASP construction are not guaranteed to be locally optimal, even with respect to simple neighborhood definitions. Hence, it is almost always beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution from its neighborhood.

Resende and Ribeiro [22] proposed an extension of the above described heuristic of Goldschmidt and Takvorian, in which a GRASP is used for finding

a linear permutation of the nodes. In the construction phase of this GRASP, the greedy algorithm used in the first phase by Goldschmidt and Takvorian is randomized: instead of selecting the node of minimum degree among those yet unselected, the selection is made from a set of low degree nodes. The local search phase of this GRASP explores the neighborhood of the current permutation by swapping the positions of two nodes at a time, attempting to reduce the number of possible edge crossings.

Incorporating the second phase of the Goldschmidt–Takvorian heuristic to the above GRASP for finding a linear permutation of the nodes results in a GRASP for graph planarization.

Each iteration of this GRASP produces three edge sets: $\mathcal{B}$ (blue edges), $\mathcal{R}$ (red edges), and $\mathcal{P}$ (the remaining edges, which are referred to as the *pale edges*). By construction, $\mathcal{B}$, $\mathcal{R}$, and $\mathcal{P}$ are such that no red or pale edge can be colored blue. Likewise, pale edges cannot be colored red. However, if there exists a pale edge $p \in \mathcal{P}$ such that all blue edges that cross with p (let $\widehat{\mathcal{B}}_p \subseteq \mathcal{B}$ be the set of those blue edges) do not cross with any red edge $r \in \mathcal{R}$, then all blue edges $b \in \widehat{\mathcal{B}}_p$ can be colored red and p can be colored blue. In case this reassignment of colors is possible, then the size of the planar subgraph is increased by one edge. This post-optimization procedure is incorporated at the end of each GRASP iteration.

Computational Results

Detailed results on a set of 75 test problems described in the literature [5,12] are reported in [22]. The description of the code used can be found in [23]. Here, we summarize computational results illustrating the effectiveness of the two-phase heuristics described in the previous section, as well as that of the exact branch and cut algorithm. These results are based on a Fortran implementation of the GRASP heuristic of Resende and Ribeiro [22], on the original code of the branch and cut algorithm of Jünger and Mutzel [16], and on published results for the heuristics of Takefuji and Lee [25] and Goldschmidt and Takvorian [22] (using the greedy algorithm for building the linear permutation of the nodes).

We give, in the table below, results comparing the four approaches on a subset of the test problems described in [12]. For each instance, the table lists the number of nodes, the number of edges, and the size of the planar subgraphs produced by each algorithm. A time limit of 1000 seconds (on a SUN SPARCstation 10/41) was imposed on the runs of the branch and cut algorithm and the best solution found was returned as a heuristic solution when optimality was not attained in that time limit. This time limit was reached on instances G12–G19.

The results in this table show that the Goldschmidt–Takvorian algorithm is a substantial improvement over the neural network approach of Takefuji and Lee. The GRASP consistently outperforms both other two-phase heuristics, not only for the problems reported in this table, but also for all of the remaining instances considered in [22].

Problem	Nodes	Edges	T-L	G-T	R-R	J-M
G1	10	22	20	20	20	20
G2	45	85	80	80	82	82
G3	10	24	21	21	24	24
G4	10	25	22	21	24	24
G5	10	26	22	21	24	24
G6	10	27	22	21	24	24
G7	10	34	23	22	24	24
G8	25	69	58	60	69	69
G9	25	70	59	60	69	69
G10	25	71	58	59	69	69
G11	25	72	60	59	69	69
G12	25	90	61	62	67	68
G13	50	367	70	131	135	125
G14	50	491	100	136	143	133
G15	50	582	101	142	144	138
G16	100	451	92	180	196	187
G17	100	742	116	219	236	213
G18	100	922	115	237	246	223
G19	150	1064	127	297	311	290

A comparison of GRASP with the branch and cut algorithm depends heavily on the instances. The results reported in [22] can be separated into two groups. On 49 of the 55 instances in the first group, the GRASP either matched or produced better solutions than the branch and cut algorithm. On 30 of those 55 instances, the GRASP solution was strictly better than the branch and cut solution. Note that, on these instances, the branch and cut algorithm was forced to stop because

of the 1000 second time limit. However, on all the remaining 20 instances, the branch and cut algorithm performs remarkably well and outperforms all other algorithms.

See also

- ► Feedback Set Problems
- ► Generalized Assignment Problem
- ► Graph Coloring
- ► Greedy Randomized Adaptive Search Procedures
- ► Optimization in Leveled Graphs
- ► Quadratic Assignment Problem
- ► Quadratic Semi-assignment Problem

References

1. Angluin D, Valiant LG (1979) Probabilistic algorithms for Hamiltonian circuits and matchings. J Comput Syst Sci 18:155–190
2. Booth K, Lueker G (1976) Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J Comput Syst Sci 13:335–379
3. Cai J, Han X, Tarjan R (1993) An O(m log n)-time algorithm for the maximal planar subgraph problem. SIAM J Comput 22:1142–1162
4. Chiba T, Nishioka I, Shirakawa I (1979) An algorithm of maximal planarization of graphs. In: Proc. 1979 IEEE Symp. Circuits and Sys., pp 649–652
5. Cimikowski RJ (1995) An analysis of heuristics for the maximum planar subgraph problem. In: Proc. 6th ACM-SIAM Symp. Discrete Algorithms, pp 322–331
6. Di Battista G, Eades P, Tamassia R, Tollis IG (1994) Algorithms for drawing graphs: An annotated bibliography. Comput Geom Th Appl 1:235–282
7. Di Battista G, Tamassia R (1989) Incremental planarity testing. Proc. 30th IEEE Symp. FOCS, pp 436–441
8. Eades P, Foulds LR, Giffin JW (1982) An efficient heuristic for identifying a maximum weight planar subgraph. In: Lecture Notes Math, vol 952. Springer, Berlin, pp 239–251
9. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Global Optim 6:109–133
10. Foulds LR, Robinson RW (1976) A strategy for solving the plant layout problem. Oper Res Quart 27:845–855
11. Foulds LR, Robinson RW (1978) Graph theoretic heuristics for the plant layout problem. Internat J Production Res 16:27–37
12. Goldschmidt O, Takvorian A (1994) An efficient graph planarization two-phase heuristic. Networks 24:69–73
13. Hasan M, Osman IH (1995) Local search algorithms for the maximal planar layout problem. Internat Trans Oper Res 2:89–106
14. Hopcroft J, Tarjan RE (1974) Efficient planarity testing. J ACM 21:549–568
15. Jünger M, Leipert S, Mutzel P (1998) A note on computing a maximal planar subgraph using PQ-trees. Techn Report Inst Informatik Univ Köln 98.320
16. Jünger M, Mutzel P (1996) Maximum planar subgraphs and nice embeddings: Practical layout tools. Algorithmica 16:33–59
17. Laguna M, Marti R (1999) GRASP and path relinking for 2-layer straight line crossing minimization. INFORMS J Comput 11:44–52
18. Lempel A, Even S, Cedarbaum I (1966) An algorithm for planarity testing of graphs. Proc. Theory of Graphs Internat. Symp. Gordon and Breach, New York, pp 215–232
19. Lengauer T (1990) Combinatorial algorithms for integrated circuit layout. Wiley, New York
20. Leung J (1992) A new graph-theoretic heuristic for facility layout. Managem Sci 38:594–605
21. Liu PC, Geldmacher RC (1977) On the deletion of nonplanar edges of a graph. In: Proc. 10th SE Conf. Comb., Graph Theory, and Comput., pp 727–738
22. Resende MGC, Ribeiro CC (1997) A GRASP for graph planarization. Networks 29:173–189
23. Ribeiro CC, Resende MGC (1999) Algorithm 797: FORTAN subroutines for approximate solution of graph planarization problems using GRASP. ACM Trans Math Softw 25:341–352
24. Stoer M (1992) Design of survivable networks. Lecture Notes Math, vol 1531. Springer, Berlin
25. Takefuji Y, Lee KC (1989) A near-optimum parallel planarization algorithm. Science 245:1221–1223
26. Takefuji Y, Lee K-C, Cho YB (1991) Comments on an O(n2)algorithm for graph planarization. IEEE Trans Computer-Aided Design 10:1582–1583
27. Tamassia R, DiBattista G (1988) Automatic graph drawing and readability of diagrams. IEEE Trans Syst, Man Cybern 18:61–79

Graph Realization via Semidefinite Programming

Anthony Man-Cho So[1], Yinyu Ye[2]

[1] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong, China

[2] Department of Management Science and Engineering and, by courtesy, Electrical Engineering, Stanford University, Stanford, USA

MSC2000: 51K05, 52C25, 68Q25, 68U05, 90C22, 90C35

Article Outline

Introduction
Formulation
Applications
Relation to the Maximum Variance Unfolding Method
References

Introduction

Due to its fundamental nature and versatile modelling power, the *Graph Realization Problem* is one of the most well-studied problems in distance geometry and has received attention in many communities. In that problem, one is given a graph $G = (V, E)$ and a set of non-negative edge weights $\{d_{ij} : (i, j) \in E\}$, and the goal is to compute a *realization* of G in the Euclidean space $\mathbb{R}^k$ for a given dimension $k \geq 1$, i. e. to place the vertices of G in $\mathbb{R}^k$ such that the Euclidean distance between every pair of adjacent vertices v_i, v_j is equal to the prescribed weight d_{ij}. The Graph Realization Problem and its variants arise from applications in very diverse areas, the two most prominent of which being molecular conformation (see, e. g., [13,15,16,19,32]) and wireless sensor network localization (see, e. g., [2,8,14,22,24]). In molecular conformation, one is interested in determining the spatial structure of a molecule from a set of geometric constraints; in wireless sensor network localization, one is interested in inferring the locations of sensor nodes in a sensor network from connectivity-imposed proximity constraints. Thus, in these contexts, an algorithm that finds a realization of the vertices in the required dimension will have interesting biochemical and engineering consequences. Unfortunately, unless P = NP, there is no efficient algorithm for solving the Graph Realization Problem for any fixed $k \geq 1$ ([23]; see also [3,4]). Nevertheless, many heuristics have been developed for the problem over the years, and various approaches have been taken to improve their efficiency (see, e. g., [1,2,13,14,15,18,20]). However, these approaches have their limitations. Specifically, either they solve the original problem only for a very restricted family of instances, or it is not clear when the algorithm would solve the original problem. Thus, an interesting question arises: given a relaxation of the Graph Realization Problem, can one derive reasonably general conditions under which the relaxation is *exact*?

We begin by examining a semidefinite programming (SDP) relaxation proposed by [10] in Section Formulation. We introduce the notion of unique k-realizability and show that the SDP relaxation is exact if and only if the input instance is uniquely k-realizable, where k is the given dimension. The notion of unique k-realizability is attractive, as it has a straightforward geometric interpretation and is also very suitable for the algorithmic treatment of the Graph Realization Problem.

Although we have formulated the Graph Realization Problem as a feasibility problem, it is clear that one can also formulate various optimization versions of it. One particularly useful objective is to maximize the sum of the distances between certain pairs of non-adjacent vertices. Such an objective essentially stretches apart pairs of non-adjacent vertices, and is more likely to flatten a high-dimensional realization into a lower dimensional one. Indeed, such a device has been proven to be very useful for finding low-dimensional realizations both in theory (see, e. g., [6,7]) and in practice (see, e. g., [9,29,30]). In Section Applications, we show how these ideas can be incorporated into the SDP model and demonstrate a connection between SDP theory and tensegrity theory in discrete geometry.

Formulation

We begin by introducing the semidefinite programming (SDP) relaxation proposed by [10]. Let $G = (V, E)$ be a graph, and let $k \geq 1$ be an integer. Let $V_1 = \{1, \ldots, n\}$ and $V_2 = \{n+1, \ldots, n+m\}$ be a partition of V. The vertices in V_1 (resp. V_2) are said to be *unpinned* (resp. *pinned*). Specifically, let $\mathbf{a} = (a_i)_{i \in V_2}$ be given, where $a_i \in \mathbb{R}^k$ for all $i \in V_2$. Then, the vertex $i \in V_2$ is constrained to be at a_i, while there are no such restrictions on the vertices in V_1. For our purposes, we may assume that $V_2 \neq \emptyset$, since we can always pin one vertex at the origin. We may also assume that $E' = \{(i, j) : i, j \in V_2\} \subset E$, since the distance between any two pinned vertices is trivially known. Now, let $E_1 = \{(i, j) \in E : i, j \in V_1\}$ be the set of edges between two unpinned vertices, and let $E_2 = \{(i, j) \in E : i \in V_2, j \in V_1\}$ be the set of edges between a pinned and an unpinned vertex. Let $\mathbf{d} = (d_{ij}^2)_{(i,j)\in E_1}$ (resp. $\bar{\mathbf{d}} = (\bar{d}_{ij}^2)_{(i,j)\in E_2}$) be a set of weights on the edges in E_1 (resp. E_2). We are then in-

terested in finding vectors $x_1, \ldots, x_n \in \mathbb{R}^k$ such that:

$$\begin{aligned} \|x_i - x_j\|^2 &= d_{ij}^2 \quad \text{for } (i,j) \in E_1 \\ \|a_i - x_j\|^2 &= \bar{d}_{ij}^2 \quad \text{for } (i,j) \in E_2 \end{aligned} \tag{1}$$

Here, $\|\cdot\|$ is the Euclidean norm, i. e. $\|x\| = \left(\sum_{i=1}^k x_i^2\right)^{1/2}$ for $x \in \mathbb{R}^k$. We say that $\mathbf{p} = (p_1, \ldots, p_n) \in \mathbb{R}^{kn}$ is a *realization* of $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a})$ in $\mathbb{R}^k$ if it satisfies (1). One may obtain a semidefinite relaxation of (1) as follows. Let $X = [x_1\ x_2\ \ldots\ x_n]$ be the $k \times n$ matrix that needs to be determined. Then, for all $(i,j) \in E_1$, we have:

$$\begin{aligned} \|x_i - x_j\|^2 &= (e_i - e_j)^T X^T X (e_i - e_j) \\ &= (e_i - e_j)(e_i - e_j)^T \bullet (X^T X) \end{aligned}$$

and for all $(i,j) \in E_2$, we have:

$$\|a_i - x_j\|^2 = \begin{pmatrix} a_i \\ -e_j \end{pmatrix}^T [I_k\ X]^T [I_k\ X] \begin{pmatrix} a_i \\ -e_j \end{pmatrix} = \begin{pmatrix} a_i \\ -e_j \end{pmatrix} \begin{pmatrix} a_i \\ -e_j \end{pmatrix}^T \bullet \begin{bmatrix} I_k & X \\ X^T & X^T X \end{bmatrix}$$

Here, e_i is the ith standard basis vector of $\mathbb{R}^n$, I_k is the k-dimensional identity matrix, and $\bullet$ is the Frobenius inner product on the space of symmetric matrices, i. e. $A \bullet B = \mathrm{tr}(A^T B) = \sum_{i,j=1}^n a_{ij} b_{ij}$ for symmetric $n \times n$ matrices A and B. Thus, problem (1) becomes that of finding a symmetric matrix $Y \in \mathbb{R}^{n \times n}$ and a matrix $X \in \mathbb{R}^{k \times n}$ that satisfy the following system:

$$\begin{gathered} (e_i - e_j)(e_i - e_j)^T \bullet Y = d_{ij}^2 \\ \text{for } (i,j) \in E_1 \\ \begin{pmatrix} a_i \\ -e_j \end{pmatrix} \begin{pmatrix} a_i \\ -e_j \end{pmatrix}^T \bullet \begin{bmatrix} I_k & X \\ X^T & Y \end{bmatrix} = \bar{d}_{ij}^2 \\ \text{for } (i,j) \in E_2 \\ Y = X^T X \end{gathered} \tag{2}$$

By relaxing $Y = X^T X$ to $Y \succeq X^T X$ and using Schur's complement (see, e. g., [11]), we obtain the following relaxed problem:

$$\begin{aligned} \text{sup} \quad & 0 \\ \text{subject to} \quad & E_{ij} \bullet Z = d_{ij}^2 \quad \text{for } (i,j) \in E_1 \\ & \bar{E}_{ij} \bullet Z = \bar{d}_{ij}^2 \quad \text{for } (i,j) \in E_2 \\ & Z \succeq \mathbf{0},\ Z_{1:k,1:k} = I_k \end{aligned} \tag{3}$$

where $Z_{1:k,1:k}$ is the $k \times k$ principal submatrix of Z indexed by the first k rows (columns),

$$E_{ij} = \begin{pmatrix} \mathbf{0} \\ e_i - e_j \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ e_i - e_j \end{pmatrix}^T$$

$$\text{and} \quad \bar{E}_{ij} = \begin{pmatrix} a_i \\ -e_j \end{pmatrix} \begin{pmatrix} a_i \\ -e_j \end{pmatrix}^T$$

Note that this formulation forces any feasible solution matrix to have rank at least k. To derive the dual of (3), let $(\theta_{ij})_{(i,j)\in E_1}$ and $(w_{ij})_{(i,j)\in E_2}$ be the dual multipliers of the constraints on E_1 and E_2, respectively. Then, the dual of (3) is given by:

$$\begin{aligned} \text{inf} \quad & I_k \bullet V + \sum_{(i,j)\in E_1} \theta_{ij} d_{ij}^2 \\ & + \sum_{(i,j)\in E_2} w_{ij} \bar{d}_{ij}^2 \\ \text{subject to} \quad & U \equiv \begin{bmatrix} V & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \sum_{(i,j)\in E_1} \theta_{ij} E_{ij} \\ & + \sum_{(i,j)\in E_2} w_{ij} \bar{E}_{ij} \succeq \mathbf{0} \\ & \theta_{ij} \in \mathbb{R} \ \text{ for all } (i,j) \in E_1; \\ & w_{ij} \in \mathbb{R} \ \text{ for all } (i,j) \in E_2 \end{aligned} \tag{4}$$

Note that the dual is always feasible, as $V = \mathbf{0}$, $\theta_{ij} = 0$ for all $(i,j) \in E_1$ and $w_{ij} = 0$ for all $(i,j) \in E_2$ is a feasible solution. Moreover, this solution has a dual objective value of 0. Thus, by the SDP strong duality theorem, if the primal is also feasible, then there is no duality gap between (3) and (4). Moreover, if Z is feasible for (3) and U is optimal for (4), then by complementarity, we have $\mathrm{rank}(Z) + \mathrm{rank}(U) \leq k + n$. In particular, since $\mathrm{rank}(Z) \geq k$, we must have $\mathrm{rank}(U) \leq n$.

We are interested in deriving the conditions under which the relaxation (3) is exact for (2). Towards that end, let us first introduce a definition:

Definition 1 We say that an instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a})$ is uniquely k-realizable if (i) there is a unique realization $\mathbf{p} = (p_1, \ldots, p_n)$ of $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a})$ in $\mathbb{R}^k$, and (ii) there does not exist $p'_1, \ldots, p'_n \in \mathbb{R}^l$, where $l > k$, such that:

$$\begin{aligned} \|p'_i - p'_j\|^2 &= d_{ij}^2 \quad \text{for } (i,j) \in E_1 \\ \left\| \begin{pmatrix} a_i \\ \mathbf{0} \end{pmatrix} - p'_j \right\|^2 &= \bar{d}_{ij}^2 \quad \text{for } (i,j) \in E_2 \\ p'_i &\neq \begin{pmatrix} p_i \\ \mathbf{0} \end{pmatrix} \text{ for some } 1 \leq i \leq n \end{aligned}$$

For the motivation of this definition, see [25]. We remark that Definition 1 can be viewed as a new notion of rigidity which takes into account both the combinatorial and the geometric aspects of the Graph Realization Problem.

At this point it is fair to ask whether Definition 1 is vacuous, i. e. whether uniquely k-realizable instances exist at all. It is not hard to see that they do exist for all $k \geq 1$. In fact, there exists a family of uniquely k-realizable instances in which the number of edges scales linearly with the number of vertices ([25]). This refutes a common belief in the literature (see, e. g., [2,5]) that the graph of any uniquely k-realizable instance must have $\Omega(n^2)$ edges.

Having established the existence of uniquely k-realizable instances, we are now ready to state the main theorem of this section. For a proof, see [25,27].

Theorem 1 *Let $G = (V, E)$ be connected, and let $\mathbf{d}$, $\bar{\mathbf{d}}$ and $\mathbf{a}$ be given. Then, the following are equivalent:*
(1) The instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a})$ is uniquely k-realizable.
(2) The max-rank solution matrix of (3) has rank k.
(3) The solution matrix of (3) satisfies $Y = X^T X$.

Although unique k-realizability is a useful notion in determining the solvability of the Graph Realization Problem, it is not stable under perturbation. Indeed, there exist instances that are uniquely k-realizable, but may no longer be so after small perturbation of the unpinned vertices; see [27]. This motivates us to define another notion called strong k-realizability:

Definition 2 We say that an instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a})$ is strongly k-realizable if (4) has a rank–n optimal dual slack matrix.

Note that if an instance is strongly k-realizable, then it is uniquely k-realizable by complementarity and Theorem 1, since the rank of any solution to (3) is equal to k.

Given an instance $\mathcal{I} = (G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a})$, we say that the instance $(G', (\mathbf{d}', \bar{\mathbf{d}}'), \mathbf{a})$ is a *sub–instance* of $\mathcal{I}$ if G' is a subgraph of G that includes all the pinned vertices, and $(\mathbf{d}', \bar{\mathbf{d}}')$ is the restriction of $(\mathbf{d}, \bar{\mathbf{d}})$ on G'. As indicated by the following theorem, the notion of strong k-realizability is very useful in identifying the uniquely k-realizable sub–instances of a given instance. Its proof can be found in [25,27].

Theorem 2 *Suppose that a given instance $\mathcal{I}$ contains a sub–instance $\mathcal{I}'$ that is strongly k-realizable. Then, in any solution to (3), the submatrix that corresponds to $\mathcal{I}'$ has rank k.*

Applications

It is often observed in practice that by "stretching apart" pairs of non-adjacent vertices, one is more likely to flatten a high-dimensional realization into a lower dimensional one. We now formalize this observation using elements of tensegrity theory (see, e. g., [12,21]). We begin with some definitions:

Definition 3 A *tensegrity* $G(\mathbf{p})$ is a graph $G = (V, E)$ together with a configuration $\mathbf{p} = (p_1, \ldots, p_n) \in \mathbb{R}^{kn}$ such that each edge is labelled as a cable, strut, or bar; each vertex is labelled as pinned or unpinned; and vertex $i \in V$ is assigned the coordinates $p_i \in \mathbb{R}^k$ for $1 \leq i \leq n$.

The label on each edge is intended to indicate its functionality. Cables (resp. struts) are allowed to decrease (resp. increase) in length (or stay the same length), but not to increase (resp. decrease) in length. Bars are forced to remain the same length. As before, a pinned vertex is forced to remain where it is. Given a graph $G = (V, E)$ and a set $\mathbf{d}$ of weights on the edges, if (i, j) is a cable (resp. strut), then d_{ij} will be the upper (resp. lower) bound on its length. If (i, j) is a bar, then d_{ij} will simply be its length.

An important concept in the study of tensegrities is that of an *equilibrium stress*:

Definition 4 An *equilibrium stress* for $G(\mathbf{p})$ is an assignment of real numbers $\omega_{ij} = \omega_{ji}$ to each edge $(i, j) \in E$ such that for each unpinned vertex i of G, we have:

$$\sum_{j:(i,j)\in E} \omega_{ij}(p_i - p_j) = \mathbf{0} \tag{5}$$

Furthermore, we say that the equilibrium stress $\omega = \{\omega_{ij}\}$ is *proper* if $\omega_{ij} = \omega_{ji} \geq 0$ (resp. ≤ 0) if (i, j) is a cable (resp. strut).

Clearly, the zero stress $\omega = \mathbf{0}$ is a proper equilibrium stress, but it is not too interesting. On the other hand, suppose that $G(\mathbf{p})$ has a non-zero equilibrium stress, and that at least one of the incident edges of vertex i has a non-zero stress. Then, Eq. (5) implies that the set of vectors $\{p_j - p_i : (i, j) \in E\}$ is linearly dependent, and

hence those vectors span a lower dimensional space. Thus, it would be nice to have conditions that guarantee the existence of a non-zero proper equilibrium stress. It turns out that the concept of an *unyielding tensegrity* is useful for that purpose.

Definition 5 Let $G = (V, E)$ be a graph, and let $\mathbf{p}$ and $\mathbf{q}$ be two configurations of G. We say that $G(\mathbf{p})$ *dominates* $G(\mathbf{q})$ (denoted by $G(\mathbf{p}) \trianglerighteq G(\mathbf{q})$) if for every pinned vertex i, we have $p_i = q_i$, and for every edge $(i, j) \in E$, we have:

$$\|p_i - p_j\| \left\{ \begin{array}{c} \geq \\ = \\ \leq \end{array} \right\} \|q_i - q_j\| \quad \text{if } (i,j) \text{ is a } \left\{ \begin{array}{c} \text{cable} \\ \text{bar} \\ \text{strut} \end{array} \right\}$$

We call $G(\mathbf{p})$ an *unyielding tensegrity* and $\mathbf{p}$ an *unyielding configuration* if any other configuration $\mathbf{q}$ with $G(\mathbf{p}) \trianglerighteq G(\mathbf{q})$ satisfies $\|p_i - p_j\| = \|q_i - q_j\|$ for all $(i, j) \in E$.

We are now ready to state the following theorem due to [6], which plays a crucial role in the characterization of the so-called 3-realizable graphs (informally, a graph G is 3-realizable if, given *any* set $\mathbf{d}$ of edge weights, whenever $(G, \mathbf{d})$ is realizable at all, then it can also be realized in $\mathbb{R}^3$; for further details, see [7]):

Theorem 3 *If $G(\mathbf{p})$ is an unyielding tensegrity with exactly one strut or cable, then $G(\mathbf{p})$ has an equilibrium stress that is non-zero on at least one edge.*

Belk's proof of Theorem 3 uses the Inverse Function Theorem and hence is not constructive. It turns out that the problem of computing an unyielding configuration $\mathbf{p}$ of a graph G can be formulated as an SDP. What is even more interesting is that the optimal dual multipliers of the SDP will give rise to a non-zero proper equilibrium stress for $G(\mathbf{p})$. Consequently, we obtain a constructive proof of Theorem 3. In fact, the SDP-based proof yields more information than that offered by Belk's proof.

Specifically, let V_1, V_2, E_1, E_2 be as before, and set $E_1^c = \{(i, j) \notin E : i, j \in V_1\}$ and $E_2^c = \{(i, j) \notin E : i \in V_2, j \in V_1\}$. Let C_1, S_1 be disjoint subsets of E_1^c, and let C_2, S_2 be disjoint subsets of E_2^c. The pairs in C_i are intended to be *cables*, and those in S_i are intended to be *struts*. We remark that we do not assume the sets C_1, C_2, S_1, S_2 to be non-empty.

Now, consider the following SDP, where we augment the formulation (3) with an objective function:

$$\begin{array}{lll} \sup & \displaystyle\sum_{(i,j)\in S_1} E_{ij} \bullet Z + \sum_{(i,j)\in S_2} \bar{E}_{ij} \bullet Z & \\ & \displaystyle - \sum_{(i,j)\in C_1} E_{ij} \bullet Z - \sum_{(i,j)\in C_2} \bar{E}_{ij} \bullet Z & \\ \text{subject to} & E_{ij} \bullet Z = d_{ij}^2 & \text{for } (i,j) \in E_1 \\ & \bar{E}_{ij} \bullet Z = \bar{d}_{ij}^2 & \text{for } (i,j) \in E_2 \\ & Z \succeq \mathbf{0}, Z_{1:k,1:k} = I_k & \end{array} \tag{6}$$

The dual of (6) is given by:

$$\begin{array}{lll} \inf & I_k \bullet V + \displaystyle\sum_{(i,j)\in E_1} \theta_{ij} d_{ij}^2 & \\ & \displaystyle + \sum_{(i,j)\in E_2} w_{ij} \bar{d}_{ij}^2 & \\ \text{subject to} & U \equiv \displaystyle - \sum_{(i,j)\in S_1} E_{ij} - \sum_{(i,j)\in S_2} \bar{E}_{ij} & \\ & \displaystyle + \sum_{(i,j)\in C_1} E_{ij} + \sum_{(i,j)\in C_2} \bar{E}_{ij} & \\ & \displaystyle + \begin{bmatrix} V & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \sum_{(i,j)\in E_1} \theta_{ij} E_{ij} & \\ & \displaystyle + \sum_{(i,j)\in E_2} w_{ij} \bar{E}_{ij} \succeq \mathbf{0} & \end{array} \tag{7}$$

We then have the following theorem due to [26]:

Theorem 4 *Let $G = (V, E)$, $\mathbf{d}$, $\bar{\mathbf{d}}$ and $\mathbf{a}$ be given such that:*

(1) there is at least one pinned vertex, and

(2) the graph $G \backslash \{n+2, \ldots, n+m\}$ is connected.

Consider the SDP (6), where we assume that:

(3) it is strictly feasible, and

(4) the objective function is not vacuous, i. e. at least one of the sets C_1, C_2, S_1, S_2 is non-empty.

Let $\bar{\mathbf{x}} = (\bar{x}_1, \ldots, \bar{x}_n) \in \mathbb{R}^{ln}$ be the positions of the unpinned vertices in $\mathbb{R}^l$ (for some $l \geq k$), obtained from the optimal primal matrix $\bar{Z}$, and let $\{\bar{\theta}_{ij}, \bar{w}_{ij}\}$ be the optimal dual multipliers. Suppose that we assign the stress $\bar{\theta}_{ij}$ (resp. $\bar{w}_{ij}$) to the bar $(i, j) \in E_1$ (resp. $(i, j) \in E_2$), a stress of 1 to all the cables in $C_1 \cup C_2$, and a stress of -1 to all the struts in $S_1 \cup S_2$. Then, the resulting assignment yields a non-zero proper equilibrium stress for the

tensegrity $G'(\bar{\mathbf{x}}, \bar{\mathbf{a}})$, *where* $G' = (V, E \cup C_1 \cup C_2 \cup S_1 \cup S_2)$ *and* $\bar{\mathbf{a}} = (\bar{a}_{n+1}, \ldots, \bar{a}_{n+m})$, *where:*

$$\bar{a}_i = \begin{pmatrix} a_i \\ \mathbf{0} \end{pmatrix} \in \mathbb{R}^l$$

The intuition behind the proof of Theorem 4 is simple. Suppose that (6) and (7) achieve the same optimal value, and that the common optimal value is attained by the primal matrix $\bar{Z}$ and the dual matrix $\bar{U}$. Then, the desired result should follow from one of the conditions for strong duality, namely the identity $\bar{Z}\bar{U} = \mathbf{0}$. Of course, strong duality for SDP does not necessarily hold, and even when it does, there is no guarantee that the optimal value is attained by any matrix (see, e. g., [17] for some examples). Thus, some additional technical assumptions are needed, and items (2) and (3) in the statement of Theorem 4 turn out to be sufficient. In fact, the conclusion of Theorem 4 remains valid if we replace (3) by the following:

(3′) the optimal value of (7) *is attained by some dual feasible matrix*

We remark that in most applications of Theorem 4, there will only be one pinned vertex, namely $a_{n+1} = \mathbf{0}$. Thus, primal strict feasibility can be ensured if the given weights $\mathbf{d}$ admit a realization whose vertices are in general position, and the connectivity condition is simply the statement that G is connected. However, the strict feasibility assumption (or the dual attainment assumption) does weaken the applicability of Theorem 4. In particular, Theorem 4 is not as general as Theorem 3, although this can be fixed (see [25] for details).

Besides strict feasibility, it is also assumed that the given instance has at least one pinned vertex. Such an assumption is necessary in order to ensure that the entries of $\bar{Z}$ are bounded, but one can no longer argue that the net stress exerted on a pinned vertex is zero. However, if there is only one pinned vertex in the given instance, then the net stress exerted on it will be zero. Thus, one may assume without loss of generality that the given instance has one pinned vertex.

Finally, observe that the assumptions in the statement of Theorem 4 buy us some additional information that is not offered by Theorem 3. Specifically, the equilibrium stress obtained in Theorem 4 is non-zero on all the cables and struts, and the magnitudes of the stress on all the cables and struts can be prescribed (by assigning appropriate weights to each summand in the primal objective function).

Relation to the Maximum Variance Unfolding Method

The idea of stretching apart pairs of non-adjacent vertices has also been used in the artifical intelligence community to detect and discover low-dimensional structure in high-dimensional data. For instance, in [29] (see also [30]), the authors proposed the so-called *Maximum Variance Unfolding* (MVU) method for the problem of manifold learning. The idea is to map a given set of high-dimensional vectors $p_1, \ldots, p_n \in \mathbb{R}^l$ to a set of low-dimensional vectors $q_1, \ldots, q_n \in \mathbb{R}^k$ (where $1 \leq k \ll l$ are given) with maximum total variance, while at the same time preserves the local distances. More precisely, consider an n-vertex connected graph $G = (V, E)$, where the set E of edges represents the set of distances that need to be preserved. The desired set of low-dimensional vectors can then be obtained by solving the following quadratic program:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} \|x_i\|^2 \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = \mathbf{0} \\ & \|x_i - x_j\|^2 = \|p_i - p_j\|^2 \\ & \text{for } (i,j) \in E \\ & x_i \in \mathbb{R}^k \qquad \text{for } 1 \leq i \leq n \end{aligned} \tag{8}$$

To explain the rationale behind the above formulation, we observe that the first constraint centers the solution vectors at the origin and eliminates the translational degree of freedom. Moreover, it implies that the objective function of (8) can be written as:

$$\sum_{i=1}^{n} \|x_i\|^2 = \frac{1}{2n} \sum_{i,j=1}^{n} \|x_i - x_j\|^2$$

Thus, we see that the MVU method attempts to "unfold" the manifold by pulling the data points as far apart as possible while preserving the local distances. We remark that such a technique has also been used for the problem of sensor network localization (see, e. g., [9,31]). Now, using the ideas in Section Formulation, we can formulate a semidefinite relaxation of (8)

as follows:

$$\begin{aligned} \text{sup} \quad & I \bullet X \\ \text{subject to} \quad & ee^T \bullet X = 0 \\ & E_{ij} \bullet X = \|v_i - v_j\|^2 \quad \text{for } (i,j) \in E \\ & X \succeq \mathbf{0} \end{aligned} \tag{9}$$

Here, $e = (1, 1, \ldots, 1)$, $E_{ij} = (e_i - e_j)(e_i - e_j)^T$, and e_i is the ith standard basis vector of $\mathbb{R}^n$. It turns out that problem (9) and its dual are closely related to the problem of finding the fastest mixing Markov process on a graph, as well as to various spectral methods for dimensionality reduction. We shall not elaborate on these results here and refer the interested reader to [28,33] for further details. Instead, we will show that the MVU problem (9) can be viewed as a problem of finding an unyielding configuration of a certain tensegrity. To begin, suppose that we are given an n-vertex connected graph $G = (\{1, \ldots, n\}, E)$ and a configuration $\mathbf{p} = (p_1, \ldots, p_n) \in \mathbb{R}^{ln}$ of the vertices. Consider the tensegrity $G'(\mathbf{p}')$, where G' is obtained from G by adding a new vertex $n + 1$ and connecting it to all the vertices of G, and $\mathbf{p}' = (\mathbf{p}, \mathbf{0}) \in \mathbb{R}^{l(n+1)}$, i. e. vertex $n+1$ is located at the origin. Furthermore, we label the edges in E as bars and the edges in $S \equiv \{(n+1, i) : 1 \le i \le n\}$ as struts. Suppose that we pin vertex $n + 1$ at the origin, i. e. $a_{n+1} = \mathbf{0}$. Now, consider the following SDP:

$$\begin{aligned} \text{sup} \quad & \sum_{i:(n+1,i)\in S} \bar{E}_{n+1,i} \bullet Z \\ \text{subject to} \quad & E_{ij} \bullet Z = \|p_i - p_j\|^2 \quad \text{for } (i,j) \in E \\ & Z \succeq \mathbf{0}, Z_{1:k,1:k} = I_k \end{aligned} \tag{10}$$

where:

$$E_{ij} = \begin{pmatrix} \mathbf{0} \\ e_i - e_j \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ e_i - e_j \end{pmatrix}^T$$

and $$\bar{E}_{n+1,i} = \begin{pmatrix} \mathbf{0} \\ -e_i \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ -e_i \end{pmatrix}^T$$

It is clear that (10) is an instance of (6). Moreover, it can be shown ([25]) that the positions $\bar{\mathbf{x}} \in \mathbb{R}^{ln}$ of the unpinned vertices obtained from the optimal primal matrix $\bar{Z}$ are automatically centered at the origin, even though such a constraint is not explicitly enforced. Thus, we see that problem (10) is equivalent to the MVU problem (9).

From the above discussion, we see that the formulation (6) is more general than the MVU formulation (9). Moreover, the flexibility in the formulation (6) often allows one to achieve the desired dimensionality reduction which the MVU formulation cannot achieve. For instance, consider the case where the input graph G is a tree. It is not hard to show that there is a placement of struts such that *all* the optimal solutions to (6) have rank 1 and hence they all give rise to one-dimensional realizations. On the other hand, the MVU formulation may yield a two-dimensional realization; see [25] for an example.

References

1. Alfakih AY, Khandani A, Wolkowicz H (1999) Solving Euclidean Distance Matrix Completion Problems via Semidefinite Programming. Comput Optim Appl 12:13–30
2. Aspnes J, Eren T, Goldenberg DK, Morse AS, Whiteley W, Yang YR, Anderson BDO, Belhumeur PN (2006) A Theory of Network Localization. IEEE Trans Mobile Comput 5(12):1663–1678
3. Aspnes J, Goldenberg D, Yang YR (2004) On the Computational Complexity of Sensor Network Localization. In: Nikoletseas S, Rolim JDP (eds) In: Proc. 1st Int Workshop Algorithmic Aspects Wirel Sens Netw (ALGOSENSORS 2004) Lecture Notes in Computer Science, vol 3121. Springer, Berlin, pp 32–44
4. Bădoiu M, Demaine ED, Hajiaghayi M, Indyk P (2006) Low–Dimensional Embedding with Extra Information. Discret Comput Geom 36(4):609–632
5. Basu A, Gao J, Mitchell JSB, Sabhnani G (2006) Distributed Localization Using Noisy Distance and Angle Information. In: Proc. 7th ACM Int Symp Mobile Ad Hoc Netw Comput (MobiHoc 2006), pp 262–273
6. Belk M (2007) Realizability of Graphs in Three Dimensions. Discret Comput Geom 37(2):139–162
7. Belk M, Connelly R (2007) Realizability of Graphs. Discret Comput Geom 37(2):125–137
8. Biswas P, Lian T-C, Wang T-C, Ye Y (2006) Semidefinite Programming Based Algorithms for Sensor Network Localization. ACM Trans Sensor Netw 2(2):188–220
9. Biswas P, Liang T-C, Toh K-C, Wang T-C, Ye Y (2006) Semidefinite Programming Approaches for Sensor Network Localization with Noisy Distance Measurements. IEEE Trans Autom Sci Eng 3(4):360–371
10. Biswas P, Ye Y (2004) Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization. In: Proc. 3rd Int Symposium on Information Processing in Sensor Networks (IPSN 2004), pp 46–54

11. Boyd S, El Ghaoui L, Feron E, Balakrishnan V (1994) Linear Matrix Inequalities in System and Control Theory, volume 15 of SIAM Stud Appl Numer Math. Soc Ind Appl Math, Philadelphia, Pennsylvania
12. Connelly R (1982) Rigidity and Energy. Invent Math 66:11–33
13. Crippen GM and Havel TF (1988) Distance Geometry and Molecular Conformation. Chemometrics Series, vol 15, Res Stud Press Ltd., Taunton, Somerset, England
14. Doherty L, Pister KSJ, El Ghaoui L (2001) Convex Position Estimation in Wireless Sensor Networks. In: Proc. 20th Annu IEEE Conference Comput Commun (INFOCOM 2001), vol 3, pp 1655–1663
15. Havel TF (2003) Metric Matrix Embedding in Protein Structure Calculations, NMR Spectra Analysis, and Relaxation Theory. Magn Reson Chem 41(S1):37–50
16. Havel TF, Wüthrich K (1985) An Evaluation of the Combined Use of Nuclear Magnetic Resonance and Distance Geometry for the Determination of Protein Conformations in Solution. J Mol Biol 182(2):281–294
17. Helmberg C (2000) Semidefinite Programming for Combinatorial Optimization. Technical Report ZR–00–34, Konrad–Zuse–Zentrum für Informationstechnik Berlin, Berlin, Germany
18. Hendrickson B (1995) The Molecule Problem: Exploiting Structure in Global Optimization. SIAM J Optim 5(4):835–857
19. Kaptein R, Boelens R, Scheek RM, van Gunsteren WF (1988) Protein Structures from NMR. Biochemistry 27(15):5389–5395
20. Laurent M (2000) Polynomial Instances of the Positive Semidefinite and Euclidean Distance Matrix Completion Problems. SIAM J Matrix Analysis Appl 22(3):874–894
21. Roth B, Whiteley W (1981) Tensegrity Frameworks. Trans Am Math Soc 265(2):419–446
22. Savvides A, Han C-C, Strivastava MB (2001) Dynamic Fine–Grained Localization in Ad–Hoc Networks of Sensors. In: Proc. 7th Annu Int Conference Mobile Comput Netw (MobiCom 2001), pp 166–179
23. Saxe JB (1979) Embeddability of Weighted Graphs in *k*–Space is Strongly NP–Hard. In: Proc. 17th Allerton Conference Commun, Control, and Comput, pp 480–489
24. Shang Y, Ruml W, Zhang Y, Fromherz M (2004) Localization from Connectivity in Sensor Networks. IEEE Trans Parallel Distrib Syst 15(11):961–974
25. So AM-C (2007) A Semidefinite Programming Approach to the Graph Realization Problem: Theory, Applications and Extensions. PhD thesis, Department of Computer Science, Stanford University, Stanford
26. So AM-C, Ye Y (2006) A Semidefinite Programming Approach to Tensegrity Theory and Realizability of Graphs. In: Proc. 17th Annu ACM–SIAM Symposium Discrete Algorithm (SODA 2006), pp 766–775
27. So AM-C, Ye Y (2007) Theory of Semidefinite Programming for Sensor Network Localization. Math Program Ser B 109(2):367–384
28. Sun J, Boyd S, Xiao L, Diaconis P (2006) The Fastest Mixing Markov Process on a Graph and a Connection to a Maximum Variance Unfolding Problem. SIAM Rev 48(4):681–699
29. Weinberger KQ, Saul LK (2006) Unsupervised Learning of Image Manifolds by Semidefinite Programming. Int J Comput Vision 70(1):77–90
30. Weinberger KQ, Sha F, Saul LK (2004) Learning a Kernel Matrix for Nonlinear Dimensionality Reduction. In: Proc. 21st Int Conference Mach Learn (ICML 2004), pp 839–846
31. Weinberger KQ, Sha F, Zhu Q, Saul LK (2007) Graph Laplacian Regularization for Large–Scale Semidefinite Programming. In: Schölkopf B, Platt J, Hofmann T (eds) Advances in Neural Information Processing Systems 19. MIT Press, Cambridge, pp 1489–1496
32. Wüthrich K (1989) The Development of Nuclear Magnetic Resonance Spectroscopy as a Technique for Protein Structure Determination. Acc Chem Res 22(1):36–44
33. Xiao L, Sun J, Boyd S (2006) A Duality View of Spectral Methods for Dimensionality Reduction. In: Proc. 23rd Int Conference Mach Learn (ICML 2006), pp 1041–1048

Greedy Randomized Adaptive Search Procedures

GPASP

MAURICIO G.C. RESENDE
Information Sci. Res., AT&T Labs Res.,
Florham Park, USA

MSC2000: 90-01, 90B40, 90C10, 90C35, 90C27, 94C15, 65H20, 65K05

Article Outline

Keywords
A Basic GRASP
Enhancements to the Basic GRASP
  Path Relinking
  Long-Term Memory
  Reactive GRASP
  Proximate Optimality Principle
  Global Convergence
  Parallel GRASP
  GRASP in Hybrid Metaheuristics

Applications of GRASP
Operations Research Problems
Industrial Applications
Conclusion
See also
References

Keywords

Combinatorial optimization; GRASP; Metaheuristics; Search heuristic

Optimization problems that involve a large finite number of alternatives often arise in industry, government and science. In these problems, one is given a finite solution set X and a real-valued function f: $X \to \mathbf{R}$, and one seeks a solution $x^* \in X$ with $f(x^*) \leq f(x)$, $\forall x \in X$. Common examples include designing efficient telecommunication networks and constructing cost effective airline crew schedules. To find the optimal solution in a com binatorial optimization problem it is theoretically possible to enumerate the solutions and evaluate each with respect to the stated objective. However, from a practical perspective, it is infeasible to follow such a strategy of complete enumeration because the number of combinations often grows exponentially with the size of problem.

Much work has been done over the last five decades to develop optimal seeking methods that do not explicitly require an examination of each alternative. This research has given rise to the field of *combinatorial optimization* (see [55]), and an increasing capability to solve ever larger real-world problems. Nevertheless, most problems found in industry and government are either computationally intractable by their nature, or sufficiently large so as to preclude the use of exact algorithms. In such cases, *heuristic methods* are usually employed to find good, but not necessarily guaranteed optimal solutions. The effectiveness of these methods depends upon their ability to adapt to a particular realization, avoid entrapment at local optima, and exploit the basic structure of the problem, such as a network or a natural ordering among its components. Furthermore, restart procedures, controlled randomization, efficient data structures, and preprocessing are also beneficial. Building on these notions, various heuristic search techniques have been developed that have demonstrably improved our ability to obtain good solutions to difficult combinatorial optimization problems. The most promising of such techniques include simulated annealing [35], tabu search [27,28,29], genetic algorithms [30] and GRASP (greedy randomized adaptive search procedures) [21,22].

In this article, we review GRASP. The components of a basic GRASP heuristic are addressed and enhancements proposed to the basic heuristic are discussed. The paper concludes with a brief literature review of applications of GRASP.

A Basic GRASP

A GRASP is a *multistart* or iterative process, in which each GRASP iteration consists of two phases, a *construction phase*, in which a feasible solution is produced, and a *local search phase*, in which a local optimum in the neighborhood of the constructed solution is sought. The best overall solution is kept as the result. The pseudocode below illustrates a GRASP procedure for minimization in which maxitr GRASP iterations are done.

```
x* = ∞;
FOR k = 1, . . . , maxitr DO
    construct (g(·), α, x);
    local (f(·), x);
    IF f(x) < f(x*) DO
        x* = x;
    END IF;
END FOR
```

Procedure grasp($f(\cdot)$, $g(\cdot)$, maxitr, x^*)

In the construction phase, a feasible solution is iteratively constructed, one element at a time. The basic GRASP construction phase is similar to the *semigreedy heuristic* proposed independently by J.P. Hart and A.W. Shogan [31]. At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i. e. those that can be added to the solution) in a candidate list C with respect to a *greedy function* g: $C \to \mathbf{R}$. This function measures the (myopic) benefit of selecting each element. The heuristic is adaptive because the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. The probabilistic

component of a GRASP is characterized by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. The list of best candidates is called the *restricted candidate list* (RCL). This choice technique allows for different solutions to be obtained at each GRASP iteration, but does not necessarily compromise the power of the adaptive greedy component of the method. Let $\alpha \in [0, 1]$ be a given parameter. The pseudocode below describes a basic GRASP construction phase.

$x = \emptyset$;
Initialize candidate set C;
WHILE $C \neq \emptyset$ DO
  $\underline{s} = \min\{g(t) : t \in C\}$;
  $\bar{s} = \max\{g(t) : t \in C\}$;
  RCL= $\{s \in C : g(s) \leq \underline{s} + \alpha(\bar{s} - \underline{s})\}$;
  Select s, at random, from the set RCL;
  $x = x \cup \{s\}$;
  Update candidate set C;
END WHILE

Procedure construct($g(\cdot)$, α, x)

The pseudocode shows that the parameter α controls the amounts of greediness and randomness in the algorithm. A value $\alpha = 0$ corresponds a *greedy construction* procedure, while $\alpha = 1$ produces *random construction*.

As is the case for many deterministic methods, the solutions generated by a GRASP construction are not guaranteed to be locally optimal with respect to simple neighborhood definitions. Hence, it is almost always beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood. The *neighborhood structure* N for a problem P relates a solution s of the problem to a subset of solutions $N(s)$. A solution s is said to be *locally optimal* if there is no better solution in $N(s)$. The key to success for a local search algorithm consists of the suitable choice of a neighborhood structure, efficient neighborhood search techniques, and the starting solution.

While such local optimization procedures can require exponential time from an arbitrary starting point, empirically their efficiency significantly improves as the initial solution improves. Through the use of customized data structures and careful implementation, an efficient construction phase can be created which produces good initial solutions for efficient local search. The result is that often many GRASP solutions are generated in the same amount of time required for the local optimization procedure to converge from a single random start. Furthermore, the best of these GRASP solutions is generally significantly better than the single solution obtained from a random starting point. The pseudocode below describes a basic local search procedure.

$H = \{y \in N(x) : f(y) < f(x)\}$;
WHILE $|H| > 0$ DO
  Select $x \in H$;
  $H = \{y \in N(x) : f(y) < f(x)\}$;
END WHILE

Procedure local($f(\cdot)$, $N(\cdot)$, x)

It is difficult to formally analyze the quality of solution values found by using the GRASP methodology. However, there is an intuitive justification that views GRASP as a repetitive sampling technique. Each GRASP iteration produces a sample solution from an unknown distribution of all obtainable results. The mean and variance of the distribution are functions of the restrictive nature of the candidate list. For example, if the cardinality of the restricted candidate list is limited to one, then only one solution will be produced and the variance of the distribution will be zero. Given an effective greedy function, the mean solution value in this case should be good, but probably suboptimal. If a less restrictive cardinality limit is imposed, many different solutions will be produced implying a larger variance. Since the greedy function is more compromised in this case, the mean solution value should degrade. Intuitively, however, by order statistics and the fact that the samples are randomly produced, the best value found should outperform the mean value. Indeed, often the best solutions sampled are optimal.

An especially appealing characteristic of GRASP is the ease with which it can be implemented. Few parameters need to be set and tuned, and therefore development can focus on implementing efficient data structures to assure quick GRASP iterations. Finally, GRASP can be trivially implemented in parallel. Each processor can be initialized with its own copy of the procedure, the instance data, and an independent random number sequence. The GRASP iterations are then performed in parallel with only a single global variable required to store the best solution found over all processors.

Enhancements to the Basic GRASP

A number of enhancements to the basic GRASP, presented in the previous section, have been proposed in the literature. In this section we review the use path relinking, long-term memory, the proximate optimality principle, and bias functions in a GRASP. We discuss a parallelization scheme and the use of GRASP in hybrid metaheuristics.

Path Relinking

M. Laguna and R. Martí [43] adapted the concept of *path relinking* for use within a GRASP. To test their concept, they im plemented a GRASP with path relinking for the 2-layer straight line crossing minimization problem. A small set of high-quality, or elite, solutions is stored to serve as guiding solutions for path relinking. Each GRASP iteration produces a locally optimal solution x^*. A solution y^* is chosen at random from the elite set and a path of solutions linking x^* to y^* is constructed by applying a series of changes to the original solution. For example, let $x^* = (1, 0, 0, 0)$ and $y^* = (0, 1, 0, 1)$. A path relinking of x^* and y^* is $x^* = (1, 0, 0, 0) \rightarrow (0, 0, 0, 0) \rightarrow (0, 1, 0, 0) \rightarrow (0, 1, 0, 1) = y^*$. Each of these path solutions is evaluated for solution quality. Laguna and Martí report that often improvements to the incumbent are found in this path relinking.

Long-Term Memory

Long-term memory is the basis for tabu search. Besides path relinking, which can thought of as a form of long-term memory, other uses of long term memory have been proposed for use in a GRASP. C. Fleurent and F. Glover [26] observe the fact that the basic GRASP does not make use of information gathered in previous iterations and propose a long term memory scheme to address this issue. M. Prais and C.C. Ribeiro [64] propose a scheme to learn an appropriate value for the RCL parameter α.

Fleurent and Glover introduced a way to use long-term memory in multistart heuristics such as GRASP. Their scheme maintains a set S of elite solutions to be used in the construction phase. To become an elite solution a solution s must be either better than the best member of S, or better than the worst member of S and sufficiently different from the other elite solutions. For example, one can count identical solution vector components and set a threshold for rejection. A *strongly determined variable* is one that cannot be changed without eroding the objective or changing significantly other variables. A *consistent variable* is one that receives a particular value in a large portion of the elite solution set. Let $I(e)$ be a measure of the strongly determined and consistent features of choice e, i. e. $I(e)$ becomes larger as e resembles solutions in elite set S. The intensity function $I(e)$ is used in the construction phase as follows. Recall that $g(e)$ is the greedy function. Let $E(e) = F(g(e), I(e))$ be a function of the greedy and the intensification functions. For example, $E(e) = \lambda\, g(e) + I(e)$. The intensification scheme biases selection from the RCL to those elements e with a high value of $E(e)$ by setting the probability of selecting e to be $p(e) = E(e)/\sum_{s \in \mathrm{RCL}} E(s)$. The function $E(e)$ can vary with time by changing the value of λ, e. g. initially λ is set to a large value and when diversification is called for, λ is decreased. A procedure for changing the value of λ is given by Fleurent and Glover. See also [11] for an application of this long-term memory strategy.

Reactive GRASP

The term 'reactive GRASP' was introduced by Prais and Ribeiro [64] for a GRASP that reacts to solutions produced by different settings of the RCL parameter α and seeks to adjust α to give the GRASP an appropriate level of greediness and randomness. At each GRASP iteration, the value of α is chosen from a discrete set of values $\{\alpha_1, \ldots, \alpha_m\}$. The probability of selecting the value α_k is $p(\alpha_k)$, for $k = 1, \ldots, m$. Reactive GRASP adaptively changes the probabilities $\{p(\alpha_1), \ldots, p(\alpha_m)\}$ to favor

values that produce good solutions. Consider applying Reactive GRASP to a minimization problem. Initially the probabilities are set as $p(\alpha_k) = 1/m$, for $i = 1, \ldots, m$, so that the values are selected uniformly. To adaptively redefine the probabilities, define $F(S^*)$ to be the value of the best solution found so far and let A_i be the average value of the solutions obtained with α_i. Prais and Ribeiro propose a period of warm-up iterations to initialize the A_i values. Periodically (say every N_α iterations) the quantities $q_i = (F(S^*)/A_i)^\delta$ are computed for $i = 1, \ldots, m$ and the probabilities are updated to $p(\alpha_i) = q_i / \sum_{j=1}^{m} q_j$, for $i = 1, \ldots, m$. Observe that the more suitable a value α_i is, the larger the value of q_i is and, consequently, the higher the value of $p(\alpha_i)$, making α_i more likely to be selected. The parameter δ can be used as an attenuation parameter. See also [16] for an application of reactive GRASP.

Proximate Optimality Principle

The *proximate optimality principal* is based on the idea that 'good solutions at one level are likely to be found close to good solutions at an adjacent level' [29]. Fleurent and Glover [26] provide a GRASP interpretation of this principle. They suggest that imperfections introduced during steps of GRASP construction can be 'ironed-out' by applying local search during (and not only at the end of) GRASP construction. Because of efficiency considerations, a practical implementation of POP to GRASP is to apply local search during a few points in the construction phase and not during each construction iteration. See also [11] for an application of the proximate optimality principle.

Global Convergence

In [52] it was pointed out that GRASP with a fixed nonzero RCL parameter α is not asymptotically convergent to a global optimum. During construction, a fixed RCL parameter may rule out a candidate that is present in all optimal solutions. Several remedies have been proposed to get around this problem. The most straightforward is the use of a randomly selected α [72]. In this approach, the parameter is selected at random from the continuous interval [0, 1] at the start of each GRASP iteration. That value is used during the entire iteration. Since a subset of the iterations are random, the algorithm becomes asymptotically globally convergent. Reactive GRASP, as described above, can also be made asymptotically globally convergent by making $\alpha_m = 1$, i. e. allowing the choice of a value that produces a random GRASP iteration. J.L. Bresina [13] introduced the concept of a *bias function* to select a candidate element to be included in the solution. Bresina's method, which is directly applicable to GRASP construction, also allows for purely random construction and is therefore asymptotically globally convergent. At each construction step, the elements in the candidate set C are ranked by their greedy function values. A bias value bias(r) is assigned to the rth ranked element. Bresina proposes several bias functions. In logarithmic bias, bias(r) = $1/\log(r + 1)$. In linear bias, bias(r) = $1/r$. In polynomial bias of order n, bias(r) = $1/r^n$. In exponential bias, bias(r) = $1/e^r$. Finally, in random bias, bias(r) = 1. During construction, the probability of selecting the rth ranked candidate is bias(r) / $\sum_{i=1}^{|C|}$ bias(i). See also [11] for an application of this bias function strategy.

Parallel GRASP

Parallel implementation of GRASP is straightforward. Two general strategies have been proposed. In search space decomposition, the search space is partitioned into several regions and GRASP is applied to each in parallel. An example of this is the GRASP for maximum independent set [23,69] where the search space is decomposed by fixing two vertices to be in the independent set. In iteration parallelization, the GRASP iterations are partitioned and each partition is assigned to a processor. See [54,56,57,58,67] for examples of parallel implementations of GRASP. Some care is needed so that different random number generator seeds are assigned to the different iterations. This can be done by running the random number generator through an entire cycle, recording all N_g seeds in a seed array. Iteration i is started with seed(i). GRASP has been implemented on distributed architectures. In [58] a *PVM-based implementation* is described. Two *MPI-based implementations* are given in [4,50]. A.C.F. Alvim [4] proposes a general scheme for MPI implementations. A master process manages seeds for slave processors. It passes blocks of seeds to each slave processor and awaits

the slaves to indicate that they have finished processing the block and need another block. Slaves also pass back to the master the best solution found for each block of iterations.

GRASP in Hybrid Metaheuristics

GRASP has been used in *hybrid metaheuristic* schemes. Laguna and J.L. González-Velarde [41] proposed a GRASP in which local search is done by tabu search. See also [16,46] for implementations of GRASP using tabu search as the local search procedure. Simulated annealing can also be used as a GRASP local search procedure if the initial temperature is low so that it remains near the neighborhood of the constructed solution. R.K. Ahuja, J.B. Orlin and A. Tiwari [3] use GRASP construction as a mechanism for generating the initial population in a genetic algorithm. GRASP is used in [45] in a genetic algorithm to implement a type of crossover called *perfect offspring*.

Applications of GRASP

We now turn our attention to a number of GRASP implementations that have appeared in the literature, covering a wide range of applications. An early tutorial on GRASP appears in [22]. We group the work into two categories, applications to operations research problems and to industrial applications.

Operations Research Problems

Applications of GRASP to operations research problems can be classified into eight categories: scheduling problems, routing problems, logic, partitioning problems, location problems, graph theoretic problems, assignment problems, and nonconvex network flow problems.

GRASP has been applied to several scheduling problems, including operations sequencing in discrete parts manufacturing [7], flight scheduling [18], just-in-time scheduling in parallel machines [41], printed wire assembly scheduling [9,19], single machine scheduling with sequence dependent setup costs and delay penalties [24], field technician scheduling [79], flow-shop with setup costs [76,77], and bus-driver scheduling [45].

Applications of GRASP to routing problems include vehicle routing with time windows [38], vehicle routing [32], aircraft routing [5], inventory routing problem with satellite facilities [10], and permanent virtual circuit (PVC) routing [66].

Problems in logic have been approached with GRASP. These include the satisfiability problem [68], maximum satisfiability [58,71,72], and inference of logical clauses from examples [15].

GRASP has been applied to partitioning problems, including graph two partition [40] and number partitioning [6].

Applications of GRASP to location problems include p-hub location [36], pure integer capacitated plant location [14], location with economies of scale [33], single source capacitated plant location [16], location of concentrators in network access design [74], and maximum covering [67].

GRASP has been used for finding approximate solutions to a number of graph theoretic problems, including set covering [21], maximum independent set [23,69], maximum clique with weighted edges [48], graph planarization [73,75], 2-layer straight line crossing minimization [43], sparse graph coloring [42], maximum weighted edge subgraph [47], the Steiner tree problem in graphs [49,50], feedback vertex set in directed graphs [60], maximum clique [1,61], and the capacitated minimum spanning tree problem [2].

Several assignment problems have been approached with GRASP. A GRASP was introduced for the quadratic assignment problem in [44]. A parallel version of this GRASP is described in [57]. Fortran subroutines for dense and sparse quadratic assignment problems can be found respectively in [70] and [59]. A modified local search for the GRASP for quadratic assignment problems is proposed in [65]. GRASP has been used to generate the initial population of a genetic algorithm for the quadratic assignment problem [3]. Long term memory schemes have been adapted to a GRASP for the quadratic assignment problem in [26]. A GRASP for the biquadratic assignment problem is described in [51]. GRASP has been applied to two multidimensional assignment problems [53,78] and to the radio link frequency assignment problem [62]. A GRASP for the generalized assignment problem was proposed in [46].

GRASP has been used for finding approximate solutions to a concave-cost network flow problem [34].

Industrial Applications

Industrial applications of GRASP can be classified into seven categories: manufacturing, transportation, telecommunications, automatic drawing, electrical power systems, military, and biology.

GRASP has been applied to several manufacturing problems, including operations sequencing in discrete parts manufacturing [7], cutting path and tool selection in computer-aided process planning [17], manufacturing equipment selection [8], component grouping [37], and printed wire assembly scheduling [9,19].

Applications of GRASP in transportation include flight scheduling and maintenance base planning [18], intermodal trailer assignment [20], and aircraft routing in response to groundings and delay [5].

In telecommunications, GRASP has been applied to the design of SDH mesh-restorable networks [63], the Steiner tree problem in graphs [49,50], permanent virtual circuit (PVC) routing [66], location of concentrators in network access design [74], traffic scheduling in satellite switched time division multi-access (SS/TDMA) systems [64], location of points of presence (PoPs) [67], and to the multicriteria radio link frequency assignment problem [62].

GRASP has been applied to automatic drawing problems, including seam drawing in mosaicing of aerial photographic maps [25], graph planarization [73,75], and 2-layer straight line crossing minimization [43].

GRASP has been applied to other industrial problems. An application to *electrical power systems* is transmission expansion planning [12]. A military application of GRASP is in multitarget multisensor tracking [53]. GRASP has been applied in biology for protein structure prediction [39].

Conclusion

We have surveyed the literature on greedy randomized adaptive search procedures (GRASP) in the 1990s. In these years many enhancements to the basic GRASP introduced in 1988 have been proposed. The number and variety of applications has grown and continues to grow.

See also

- ▶ Feedback Set Problems
- ▶ Generalized Assignment Problem
- ▶ Graph Coloring
- ▶ Graph Planarization
- ▶ Heuristics for Maximum Clique and Independent Set
- ▶ Maximum Satisfiability Problem
- ▶ Quadratic Assignment Problem
- ▶ Quadratic Semi-assignment Problem

References

1. Abello J, Pardalos PM, Resende MGC (1999) On maximum clique problems in very large graphs. In: Abello J, Vitter J (eds) External memory algorithms and visualization. 50DIMACS, Amer Math Soc, pp 119–130
2. Ahuja RK, Orlin JB, Sharma D (1998) New neighborhood search structures for the capacitated minimum spanning tree problem. Techn Report Dept ISE Univ Florida
3. Ahuja RK, Orlin JB, Tiwari A (2000) A greedy genetic algorithm for the quadratic assignment problem. Comput Oper Res 27:917–934
4. Alvim ACF (Apr. 1998) Parallelization strategies for the metaheuristic GRASP. MSc Thesis Dept Computer Sci Catholic Univ Rio de Janeiro
5. Argüello MF, Bard JF, Yu G (1997) A GRASP for aircraft routing in response to groundings and delays. J Combin Optim 1:211–228
6. Argüello MF, Feo TA, Goldschmidt O (1996) Randomized methods for the number partitioning problem. Comput Oper Res 23(2):103–111
7. Bard JF, Feo TA (1989) Operations sequencing in discrete parts manufacturing. Managem Sci 35:249–255
8. Bard JF, Feo TA (1991) An algorithm for the manufacturing equipment selection problem. IIE Trans 23:83–92
9. Bard JF, Feo TA, Holland S (1996) A GRASP for scheduling printed wiring board assembly. IIE Trans 28:155–165
10. Bard JF, Huang L, Jaillet P, Dror M (1998) A decomposition approach to the inventory routing problem with satellite facilities. Transport Sci 32:189–203
11. Binato S, Hery WJ, Loewenstern D, Resende MGC (1999) Approximate solution of the job shop scheduling problem using GRASP. Techn Report AT&T Lab Res
12. Binato S, Oliveira GC, Araújo JL (1998) A greedy randomized adaptive search procedure for transmission expansion planning. IEEE Trans Power Systems
13. Bresina JL (1996) Heuristic-biased stochastic sampling. In: Proc. AAAI-96, pp 271–278

14. Delmaire H, Díaz JA, Fernández E, Ortega M (1997) Comparing new heuristics for the pure integer capacitated plant location problem. Techn Report Dept Statist and Oper Res Univ Politecn Catalunya, Barcelona, no. DR97/10
15. Deshpande AS, Triantaphyllou E (1998) A greedy randomized adaptive search procedure (GRASP) for inferring logical clauses from examples in polynomial time and some extensions. Math Comput Modelling 27:75–99
16. Díaz JA, Fernández E (1998) A hybrid GRASP-tabu search algorithm for the single source capacitated plant location problem. Techn Report Dept Statist and Oper Res Univ Politecn Catalunya, Barcelona
17. Feo TA, Bard JF (1989) The cutting path and tool selection problem in computer-aided process planning. J Manufacturing Systems 8:17–26
18. Feo TA, Bard JF (1989) Flight scheduling and maintenance base planning. Managem Sci 35:1415–1432
19. Feo TA, Bard J, Holland S (1995) Facility-wide planning and scheduling of printed wiring board assembly. Oper Res 43:219–230
20. Feo TA, González-Velarde JL (1995) The intermodal trailer assignment problem: Models, algorithms, and heuristics. Transport Sci 29:330–341
21. Feo TA, Resende MGC (1989) A probabilistic heuristic for a computationally difficult set covering problem. Oper Res Lett 8:67–71
22. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Global Optim 6:109–133
23. Feo TA, Resende MGC, Smith SH (1994) A greedy randomized adaptive search procedure for maximum independent set. Oper Res 42:860–878
24. Feo TA, Sarathy K, McGahan J (1996) A GRASP for single machine scheduling with sequence dependent setup costs and linear delay penalties. Comput Oper Res 23:881–895
25. Fernández E, Martí R (1999) GRASP and tabu search for seam drawing in mosaicking of aerial photographic maps. J Heuristics 5:181–197
26. Fleurent C, Glover F (1999) Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. INFORMS J Comput 11:198–204
27. Glover F (1989) Tabu search – Part I. ORSA J Comput 1:190–206
28. Glover F (1990) Tabu search – Part II. ORSA J Comput 2:4–32
29. Glover F, Laguna M (1997) Tabu search. Kluwer, Dordrecht
30. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading
31. Hart JP, Shogan AW (1987) Semi-greedy heuristics: An empirical study. Oper Res Lett 6:107–114
32. Hjorring CA (1995) The vehicle routing problem and local search metaheuristics. PhD Thesis, Univ. Auckland
33. Holmqvist K, Migdalas A, Pardalos PM (1997) Greedy randomized adaptive search for a location problem with economies of scale. In: Bomze IM et al (eds) Developments in Global Optimization. Kluwer, Dordrecht, pp 301–313
34. Holmqvist K, Migdalas A, Pardalos PM (1998) A GRASP algorithm for the single source uncapacitated minimum concave-cost network flow problem. In: Pardalos PM, Du D-Z (eds) Network design: Connectivity and facilities location. DIMACS 40. Amer. Math. Soc., Providence, pp 131–142
35. Kirkpatrick S (1984) Optimization by simulated annealing: Quantitative studies. J Statist Phys 34:975–986
36. Klincewicz JG (1992) Avoiding local optima in the p-hub location problem using tabu search and GRASP. Ann Oper Res 40:283–302
37. Klincewicz JG, Rajan A (1994) Using GRASP to solve the component grouping problem. Naval Res Logist 41:893–912
38. Kontoravdis G, Bard JF (1995) A GRASP for the vehicle routing problem with time windows. ORSA J Comput 7:10–23
39. Krasnogor N, Pelta DA, Russo W, Terrazas G (1998) A GRASP approach to the protein structure prediction problem. Techn Report LIFIA Lab Univ La Plata
40. Laguna M, Feo TA, Elrod HC (1994) A greedy randomized adaptive search procedure for the two-partition problem. Oper Res 42:677–687
41. Laguna M, González-Velarde JL (1991) A search heuristic for just-in-time scheduling in parallel machines. J Intelligent Manufacturing 2:253–260
42. Laguna M, Martí R (1998) A GRASP for coloring sparse graphs. Techn Report Graduate School Business, Univ Colorado
43. Laguna M, Martí R (1999) GRASP and path relinking for 2-layer straight line crossing minimization. INFORMS J Comput 11:44–52
44. Li Y, Pardalos PM, Resende MGC (1994) A greedy randomized adaptive search procedure for the quadratic assignment problem. In: Pardalos PM, Wolkowicz H (eds) Quadratic Assignment and Related Problems. DIMACS 16. Amer. Math. Soc., Providence, pp 237–261
45. Lourenço H Ramalhinho, Paixao JP, Portugal R (1998) Metaheuristics for the bus-driver scheduling problem. Techn Report Dept Economics and Management Univ Pompeu Fabra, Barcelona
46. Lourenço H Ramalhinho, Serra D (May 1998) Adaptive approach heuristics for the generalized assignment problem. Techn Report Dept Economics and Management Univ Pompeu Fabra, Barcelona
47. Macambira EM, Meneses CN (1998) A GRASP algorithm for the maximum weighted edge subgraph problem. Techn Report Dept Statist and Computation Univ Ceará, Fortaleza, CE 60740-000
48. Macambira EM, Souza CC de (Oct. 1997) A GRASP for the maximum clique problem with weighted edges. In: Proc. XXIX Brazilian Symp. Operations Research, p 70 In Portuguese

49. Martins SL, Pardalos PM, Resende MGC, Ribeiro CC (1999) Greedy randomized adaptive search procedures for the Steiner problem in graphs. In: Pardalos PM, Rajasejaran S, Rolim J (eds) Randomization methods in algorithmic design. DIMACS 43. Amer. Math. Soc., Providence, pp 133–145
50. Martins SL, Ribeiro CC (1998) A parallel GRASP for the Steiner problem in graphs. Proc. Irregular'98, In: Lecture Notes Computer Sci, vol 1457. Springer, Berlin, pp 285–297
51. Mavridou T, Pardalos PM, Pitsoulis LS, Resende MGC (1997) A GRASP for the biquadratic assignment problem. Europ J Oper Res 105:613–621
52. Mockus J, Eddy E, Mockus A, Mockus L, Reklaitis GV (1997) Bayesian discrete and global optimization. Kluwer, Dordrecht
53. Murphey RA, Pardalos PM, Pitsoulis LS (1998) A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. In: Pardalos PM, Du D-Z (eds) Network design: Connectivity and facilities location. DIMACS 40. Amer. Math. Soc., Providence, pp 277–301
54. Murphey RA, Pardalos PM, Pitsoulis LS (1998) A parallel GRASP for the data association multidimensional assignment problem. In: Pardalos PM (ed) Parallel processing of discrete problems. IMA vol Math Appl, vol 106. Springer, Berlin, pp 159–180
55. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: Algorithms and complexity. Prentice-Hall, Englewood Cliffs
56. Pardalos PM, Pitsoulis L, Mavridou T, Resende MGC (1995) Parallel search for combinatorial optimization: Genetic algorithms, simulated annealing and GRASP. In: Ferreira A, Rolim J (eds) Parallel Algorithms for Irregularly Structured Problems, Proc. 2nd Internat. Workshop –Irregular'95, Lecture Notes Computer Sci. Springer, Berlin, pp 317–331
57. Pardalos PM, Pitsoulis LS, Resende MGC (1995) A parallel GRASP implementation for the quadratic assignment problem. In: Ferreira A, Rolim J (eds) Parallel Algorithms for Irregularly Structured Problems – Irregular'94. Kluwer, Dordrecht, pp 111–130
58. Pardalos PM, Pitsoulis LS, Resende MGC (1996) A parallel GRASP for MAX-SAT problems. In: Lecture Notes Computer Sci, vol 1180. Springer, Berlin, pp 575–585
59. Pardalos PM, Pitsoulis LS, Resende MGC (1997) Algorithm 769: Fortran subroutines for approximate solution of sparse quadratic assignment problems using GRASP. ACM Trans Math Softw 23:196–208
60. Pardalos PM, Qian T, Resende MGC (1998) A greedy randomized adaptive search procedure for the feedback vertex set problem. J Combin Optim 2(3)
61. Pardalos PM, Resende MGC, Rappe J (1998) An exact parallel algorithm for the maximum clique problem. In: De Leone R et al (eds) High performance algorithms and software in nonlinear optimization. Kluwer, Dordrecht, pp 279–300
62. Pasiliao EL (1998) A greedy randomized adaptive search procedure for the multi-criteria radio link frequency assignment problem. Techn Report Dept ISE Univ Florida
63. Poppe F, Pickavet M, Arijs P, Demeester P (1997) Design techniques for SDH mesh-restorable networks. Proc. European Conf. Networks and Optical Communications (NOC'97), Volume 2: Core and ATM Networks, pp 94–101
64. Prais M, Ribeiro CC (2000) Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. INFORMS J Comput 12(3):164–176
65. Rangel MC, Abreu NMM de, Boaventura-Netto PO, Boeres MCS (1998) A modified local search for GRASP in the quadratic assignment problem. Techn Report Production Engin Program, COPPE, Federal Univ Rio de Janeiro
66. Resende LIP, Resende MGC (1997) A GRASP for frame relay PVC routing. Techn Report AT&T Lab Res
67. Resende MGC (1998) Computing approximate solutions of the maximum covering problem using GRASP. J Heuristics 4:161–171
68. Resende MGC, Feo TA (1996) A GRASP for satisfiability. In: Johnson DS, Trick MA (eds) Cliques, Coloring and Satisfiability: The Second DIMACS Implementation Challenge. DIMACS 26. Amer. Math. Soc., Providence, pp 499–520
69. Resende MGC, Feo TA, Smith SH (1998) Fortran subroutines for approximate solution of maximum independent set problems using GRASP. ACM Trans Math Softw 24:386–394
70. Resende MGC, Pardalos PM, Li Y (1996) Algorithm 754: Fortran subroutines for approximate solution of dense quadratic assignment problems using GRASP. ACM Trans Math Softw 22:104–118
71. Resende MGC, Pitsoulis LS, Pardalos PM (1997) Approximate solution of weighted MAX-SAT problems using GRASP. In: Gu J, Pardalos PM (eds) Satisfiability problems. DIMACS 35. Amer. Math. Soc., Providence, pp 393–405
72. Resende MGC, Pitsoulis LS, Pardalos PM (2000) Fortran subroutines for computing approximate solutions of MAX-SAT problems using GRASP. Discrete Appl Math 100:95–113
73. Resende MGC, Ribeiro CC (1997) A GRASP for graph planarization. Networks 29:173–189
74. Resende MGC, Ulular O (1997) SMART: A tool for AT&T Worldnet access design – Location of Cascade 9000 concentrators. Techn Report AT&T Lab Res
75. Ribeiro CC, Resende MGC (1999) Algorithm 797: Fortran subroutines for approximate solution of graph planarization problems using GRASP. ACM Trans Math Softw 25:341–352
76. Ríos-Mercado RZ, Bard JF (1998) Heuristics for the flow line problem with setup costs. Europ J Oper Res 110:76–98
77. Ríos-Mercado RZ, Bard JF (1999) An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup costs. J Heuristics 5:57–74

78. Robertson AJ (1998) A set of greedy randomized adaptive local search procedure (GRASP) implementations for the multidimensional assignment problem
79. Xu J, Chiu S (1996) Solving a real-world field technician scheduling problem. Proc. Internat. Conf. Management Sci. and the Economic Development of China, Hong-Kong, July 1996. pp 240–248

Gröbner Bases for Polynomial Equations

P. O. Lindberg[1], Lars Svensson[2]
[1] Linköping University, Linköping, Sweden
[2] KTH, Stockholm, Sweden

MSC2000: 12D10, 12Y05, 13P10

Article Outline

Keywords
What is a Gröbner Basis
What are Gröbner Bases good for
Using Gröbner Bases
and Learning more About them
See also
References

Keywords

Polynomial equations; Zeros; Gröbner basis

Polynomial equations (in several variables) arise in many areas connected to management science. They could describe the feasible set of an optimization problem, the Karush–Kuhn–Tucker conditions for the same problem, or maybe constraints on the positions of the links of a robot arm in a flexible manufacturing system.

There are many analogies between polynomial equations and their special case, linear equations.

- One might want to solve the equations, i. e. find one or all solutions, determine whether a solution is unique or determine whether the system in inconsistent.
- One might want to answer more abstract questions, such as whether a given equation is a consequence of a given set of equations (cf. ► Farkas lemma; ► Farkas lemma: Generalizations).

For linear equations a fundamental concept is that of a (linear) basis and the fundamental tool is that of Gaussian elimination, by which one can construct a basis from a given set of vectors. Similarly, for polynomials there is the corresponding concepts of a *Gröbner basis* and the *Buchberger algorithm*, which for a given set of polynomials constructs a Gröbner basis. In particular one can convert a system of polynomial equations to triangular form, which allows for a solution by back substitution. In Gaussian elimination, the variables/columns have an ordering that influences the end result. Similarly, for Gröbner bases we need an order, not only for the variables, but for *monomials*, i. e. the simplest possible polynomials, such as $x_1^3 x_4$, that are products of variables. In this short note we will review Gröbner basis for polynomial equations.

Before defining a Gröbner base we will give an example.

Example 1 Suppose we want to find the local optima of the following optimization problem ([4, Problem 337]; also used in [3]), by solving the KKT-conditions:

$$(P)\begin{cases}\min & f(x) = 9x_1^2 + x_2^2 + 9x_3^2\\ \text{s.t.} & g_1(x) = 1 - x_1x_2 \le 0\\ & g_2(x) = 1 - x_2 \le 0\\ & g_3(x) = x_3 - 1 \le 0.\end{cases}$$

The KKT conditions for (P) are:

$$(KKT)\begin{cases}18x_1 - \lambda_1 x_2 = 0\\ 2x_2 - \lambda_1 x_1 - \lambda_2 = 0\\ 18x_3 + \lambda_3 = 0\\ \lambda_1(1 - x_1x_2) = 0\\ \lambda_2(1 - x_2) = 0\\ \lambda_3(x_3 - 1) = 0.\end{cases}$$

Further suppose we use a *lexicographical order* of the monomials such that $x_1 > x_2 > x_3 > \lambda_1 > \lambda_2 > \lambda_3$. Then, computing the Gröbner basis for the set of polynomials in the above system and forming the corresponding

equation system, we get

$$\begin{cases} 18x_1 - x_2\lambda_1 = 0 \\ x_2\lambda_1 - 36x_3 + 18\lambda_2 = 0 \\ x_2\lambda_2 - \lambda_2 = 0 \\ 2x_2 - \lambda_1 - \lambda_2 = 0 \\ 18x_3 - \lambda_3 = 0 \\ \lambda_1^3 - 36\lambda_1 - 18\lambda_2^2 + 36\lambda_2 = 0 \\ \lambda_1\lambda_2 + \lambda_2^2 - 2\lambda_2 = 0 \\ \lambda_2^3 + 14\lambda_2^2 - 32\lambda_2 = 0 \\ \lambda_3^2 + 18\lambda_3 = 0. \end{cases}$$

This system has an obvious triangular structure, that we have tried to display graphically. The last equation contains only λ_3. Then comes equations in λ_2 (and possibly λ_3) and so on. In a similar way as in Gaussian elimination, the system can thus be solved by back substitution. In each step, one then has to solve a single variable polynomial equation, giving possibly several solutions, each of which is substituted into the preceding equations. Thus the solution process evolves in a tree-like structure. It might happen, that one has to solve for a variable that is already computed. Then of course the solutions have to agree, else they are discarded.

The above type of structure will always occur if there are finitely many solutions. It might happen, though, that the system allows a manifold of solutions. In this case it might e. g. happen that the last equation contains two variables or that you in the back substitution process comes to an equation with two (or more) undetermined variables. These equations then give a parametrization of the manifold.

What is a Gröbner Basis

In Gaussian elimination the variables are ordered and the basic reduction rule is to replace the equations $f = 0$, $g = 0$ by $f = 0$, $g - cf = 0$ where the constant c is chosen so that the leading terms in g and cf coincide.

In systems of polynomial equations we do something quite similar. First we extend the ordering of the variables to a total ordering of all monomials in a way such that $m' < m'' \Rightarrow mm' < mm''$ for all monomials m, m' and m'' and so that 1 is the least one.

The basic reduction rule is now to replace the equations $f = 0$, $g = 0$ by $f = 0$, $g - cmf = 0$ where the constant c and the monomial m are chosen so that the leading terms of g and cmf coincide. This implies that $h = g - cmf$ is 'smaller' than g in the ordering. If such a reduction of g with f is possible and $h = g - cmf$ we will write $g \to_f h$.

Definition 2 A finite set G of polynomials is a *Gröbner basis* if for every polynomial q there exist a *unique* r and a finite reduction chain $q \to_{g_1} q_1 \to_{g_2} \cdots \to_{g_k} q_k = r$ for some $g_1, \ldots, g_k$ in G and such that r cannot be reduced further. The unique polynomial r is called the *normal form* of q modulo G.

Given a finite set of vectors we can use Gaussian elimination to compute a basis of vectors spanning the same linear space. Given a finite set P of polynomials (and an admissible monomial ordering), one can use the Buchberger algorithm to compute a Gröbner basis G, spanning the same 'space' of polynomials as P. (By the space of polynomial spanned by P is meant the *ideal* generated by P, i. e. the set of finite linear combinations $q_1p_1 + \cdots + q_sp_s$ where the p_i-s are in P and the q_i-s are arbitrary polynomials.) We say that G is a Gröbner basis for P. Moreover, the common zeros of P are the same as those of G.

What are Gröbner Bases good for

Roughly speaking, all questions concerning a system of polynomial equations $f_1 = \cdots = f_s = 0$ can be answered if we have a corresponding Gröbner basis. Here we list just a few of them.

- Is the system solvable?
- If the system is solvable, how many solutions are there, and which are they?
- How many real solutions are there? (in case the coefficients are real). Here we can also allow for inequalities.
- Is it possible to eliminate some of the variables?
- Given some polynomial f, does f vanish whenever $f_1 \cdots f_s$ does? This can be used for automated proofs in geometry.
- Given some polynomial f, does there exist polynomials $q_1, \ldots, q_s$ such that $f = q_1f_1 + \cdots + q_sf_s$?
- Is it possible to describe the algebraic relations between the f_i-s, i. e. the set of polynomials q in s variables such that $q(f_1, \ldots, f_s)$ is the zero polynomial.

- Can a given polynomial f be written as $f = q(f_1, \ldots, f_s)$ for some polynomial q in s variables, and in case it can, is it possible to compute q?
- Can we compute a vector space basis for the vector space of polynomials modulo $f_1 \cdots f_s$?

Using Gröbner Bases and Learning more About them

Essentially all major mathematical computer packages with symbolic capabilities contain modules for Gröbner bases. The main examples are Maple and Mathematica. For a short but more detailed introduction to Gröbner bases, see [3]. The book [2] gives a rather short introduction to the field. One standard textbook is [1]

See also

- ► Contraction-mapping
- ► Fundamental Theorem of Algebra
- ► Global Optimization Methods for Systems of Nonlinear Equations
- ► Interval Analysis: Systems of Nonlinear Equations
- ► Nonlinear Least Squares: Newton-type Methods
- ► Nonlinear Systems of Equations: Application to the Enclosure of All Azeotropes

References

1. Cox D, O'Shea D (1992) Ideals, varieties and algorithms. An Introduction to computational algebraic geometry and commutative algebra. Springer, Berlin
2. Fröberg R (1997) An introduction to Gröbner bases. Wiley, New York
3. Hägglöf K, Lindberg PO, Svensson L (1995) Computing global minima to polynomial optimization problems using Gröbner bases. J Global Optim 7:115–125
4. Schittkowski K (1987) More test examples for nonlinear programming codes. Springer, Berlin

H

Hamilton–Jacobi–Bellman Equation

WILLIAM R. ESPOSITO
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 49L20, 34H05, 90C39

Article Outline

Keywords
Problem Formulation
Derivation
Sufficiency Theorem
Example
Linear-Quadratic Problem
Solution Methods and Applications
See also
References

Keywords

Dynamic programming; Continuous-time optimal control; Hamilton–Jacobi–Bellman equation

Even though *dynamic programming* [2] was originally developed for the solution of problems which exhibit discrete types of decisions, it has also been applied to continuous formulations. In this article, the application of dynamic programming to the solution of continuous-time optimal control problems is discussed. By discretizing the problem, applying the dynamic programming equations, then returning to the continuous domain, a partial differential equation results, the *Hamilton–Jacobi–Bellman equation* (HJB equation). This equation is often referred to as the *continuous-time equivalent of the dynamic programming algorithm*. In this article, the HJB equation will first be derived. A simple application will be presented, in addition to its use in solving the linear quadratic control problem. Finally, a brief overview of some solution methods and applications presented in the literature will be given.

Problem Formulation

The dynamic programming approach will be applied to a system of the following form:

$$\begin{cases} \dot{z}(t) = f(z(t), u(t)), \\ z(0) = z_0, \qquad\qquad 0 \le t \le T, \end{cases} \tag{1}$$

where $z(\mathrm{t}) \in \mathbf{R}^n$ is the state vector at time t with time derivative given by $\dot{z}(t)$, $u(t) \in U \subset \mathbf{R}^m$ is the control vector at time t, U is the set of control constraints, and T is the terminal time. The function $f(z(t), u(t))$ is continuously differentiable with respect to z and continuous with respect to u. The set of admissible control trajectories are given by the piecewise constant functions, $\{u(t): u(t) \in U, \forall t \in [0, T]\}$. It is assumed that for any admissible control trajectory, that a state trajectory $z^u(t)$ exists and is unique.

The objective is to determine a control trajectory and the corresponding state trajectory which minimizes a cost function of the form:

$$h(z^u(T)) + \int_0^T g(z^u(t), u(t))\, dt, \tag{2}$$

where the functions g, and h are continuously differentiable with respect to both z and u.

Derivation

The derivation of the Hamilton–Jacobi–Bellman equation is taken from [3]. The time horizon is first dis-

cretized into N equally spaced intervals with:

$$\delta = \frac{T}{N}.$$

Also, the state and control are represented by:

$$z_k = z(k\delta), \quad k = 0, \ldots, N,$$
$$u_k = u(k\delta), \quad k = 0, \ldots, N.$$

The continuous-time system is approximated by:

$$z_{k+1} = z_k + f(z_k, u_k)\delta.$$

The cost function is rewritten as:

$$h(z_N) + \sum_{k=0}^{N-1} g(z_k, u_k)\delta.$$

The dynamic programming algorithm is now applied with the following definitions:

- $J^*(t, z)$ is the optimal cost-to-go for the continuous problem;
- $\widehat{J}^*(t, z)$ is the optimal cost-to-go for the discrete approximation.

The dynamic programming equations then take the form:

$$\widehat{J}^*(N\delta, z) = h(z), \tag{3}$$

$$\widehat{J}^*(k\delta, z) = \min_{u\in U}\left[g(z, u)\delta + \widehat{J}^*((k+1)\delta, z + f(z, u)\delta)\right], \quad k = 0, \ldots, N-1. \tag{4}$$

It is assumed that $\widehat{J}^*(t, z)$ has the necessary differentiability requirements to write the following Taylor series expansion:

$$\widehat{J}^*((k+1)\delta, z + f(z, u)\delta) = \widehat{J}^*(k\delta, z) + \nabla_t \widehat{J}^*(k\delta, z)\delta + \nabla_z \widehat{J}^{*\top}(k\delta, z) f(z, u)\delta + o(\delta), \tag{5}$$

where $o(\delta)$ represents second order terms which satisfy $o(\delta)/\delta \to 0$ as $\delta \to 0$. Substituting (5) into (4) results in:

$$\widehat{J}^*(k\delta, z) = \min_{u\in U}\left[g(z, u)\delta + \widehat{J}^*(k\delta, z) + \nabla_t \widehat{J}^*(k\delta, z)\delta + \nabla_z \widehat{J}^{*\top}(k\delta, z) f(z, u)\delta + o(\delta)\right]. \tag{6}$$

Dividing (6) by δ and $\widehat{J}^*(k\delta, z)$, and taking the limit as $\delta \to 0$ with the assumption that

$$\lim_{\substack{k\to\infty \\ \delta\to 0 \\ k\delta = t}} \widehat{J}^*(k\delta, z) = J^*(t, z)$$

results in

$$0 = \min_{u\in U}\left[g(z, u) + \nabla_t J^*(t, z) + \nabla_x J^{*\top}(t, z) f(z, u)\right], \quad \forall t, z, \tag{7}$$

with the boundary condition

$$J^*(T, z) = h(z).$$

This partial differential equation is known as the Hamilton–Jacobi–Bellman equation (HJB equation).

Sufficiency Theorem

This theorem is presented in [3]. Suppose $V(t, z)$ is a solution to the HJB equation, that is, V is continuously differentiable with respect to z and t and satisfies:

$$0 = \min_{u\in U}\left[g(z, u) + \nabla_t V(t, z) + \nabla_x V^{\top}(t, z) f(z, u)\right], \quad \forall z, t, \tag{8}$$

$$V(T, z) = h(z), \quad \forall z. \tag{9}$$

Suppose also that $\mu^*(t, z)$ attains the minimum in (8) for all t and z. Let $z^*(t)$ be the state trajectory obtained from the given initial condition $z(0)$ when the control trajectory $u^*(t) = \mu^*(t, z^*(t))$ is used. (That is, $z^*(0) = z(0), \dot{z}^* = f(z^*(t), \mu^*(t, z^*(t)))$; one also assumes that this differential equation has a unique solution starting at any pair (t, z) and that the control trajectory is piecewise continuous in time.) Then V is the unique solution of the HJB equation and is equal to the optimal cost-to-go function

$$V(t, z) = J^*(t, z), \quad \forall z, t.$$

Furthermore, the control trajectory, $u^*(t)$ is optimal for all $t \in [0, T]$.

Example

Consider the simple dynamic system:

$$\dot{z}(t) = u(t)$$

with the control bounded by $u(t) \in [-1, 1]$ and time over the range $t \in [0, T]$. The cost function is given as:

$$\frac{1}{2} z(T)^2.$$

Writing the HJB equation for this system gives

$$0 = \min_{u \in [-1,1]} \left[\nabla_t V(t,z) + \nabla_z V(t,z) u \right], \quad \forall t, z,$$

with the boundary condition,

$$V(T,z) = \frac{1}{2} z^2.$$

The obvious choice of a control policy is to drive the state to zero as fast as possible and keep it there. This corresponds to the policy:

$$\mu^*(t,z) = -\operatorname{sgn}(z) = \begin{cases} 1 & \text{if } z < 0, \\ 0 & \text{if } z = 0, \\ -1 & \text{if } z > 0. \end{cases}$$

The cost associated with this policy for a given initial time and state is:

$$J^*(t,z) = \frac{1}{2} \left(\max\{0, |z| - (T-t)\} \right)^2.$$

This function satisfies the terminal condition $J^*(T,z) = z^2/2$. Also,

$$\nabla_t J^*(t,z) = \max\{0, |z| - (T-t)\},$$
$$\nabla_z J^*(t,z) = \operatorname{sgn}(z) \max\{0, |z| - (T-t)\}.$$

Substituting these expressions into the HJB equation results in

$$0 = \min_{u \in [-1,1]} \left[1 + \operatorname{sgn}(z) u \right] \max\{0, |z| - (T-t)\},$$

which can be shown to hold for all (t,z). The minimum is attained for $u = -\operatorname{sgn}(z)$, and one therefore concludes from the sufficiency theorem presented above that $J^*(t,z)$ is indeed the optimal cost-to-go function.

Linear-Quadratic Problem

Consider a general n-dimensional time-invariant linear system

$$\dot{z}(t) = Az(t) + Bu(t)$$

with a cost function defined by

$$z^\top(T) Q_T z(T) + \int_0^T z^\top(t) Q z(t) + u^\top(t) R u(t)\, dt,$$

where the matrices Q and Q_T are symmetric positive semidefinite, and the matrix R is symmetric positive definite. The HJB equation is written as

$$0 = \min_{u \in \mathbf{R}^m} \left[z^\top Q z + u^\top R u + \nabla_t V(t,z) + \nabla_z V^\top(t,z)(Az + Bu) \right], \quad V(T,z) = z^\top Q_T z. \tag{10}$$

Try a solution of the form:

$$V(t,z) = z^\top K(t) z,$$

where $K(t)$ is a symmetric $n \times n$ matrix. One then has

$$\nabla_z V(t,z) = 2K(t)z,$$
$$\nabla_t V(t,z) = z^\top \dot{K}(t) z.$$

Substituting the above expressions into (10) results in

$$0 = \min_{u \in \mathbf{R}^m} \left[z^\top Q z + u^\top R u + z^\top \dot{K}(t) z + 2 z^\top K(t) A z + 2 z^\top K(t) B u \right]. \tag{11}$$

The minimum is obtained when the gradient with respect to u is zero. This results in

$$2B^\top K(T) z + 2Ru = 0$$

or

$$u = -R^{-1} B^\top K(t) z.$$

Substituting this expression into (11), the following results:

$$0 = z^\top \left(\dot{K}(t) + K(t)A + A^\top K(t) - K(t) B R^{-1} B^\top K(t) + Q \right) z.$$

Therefore, $K(t)$ must satisfy the following matrix differential equation:

$$\dot{K}(t) = -K(t)A - A^\top K(t) + K(t) B R^{-1} B^\top K(t) - Q,$$

with the terminal condition

$$K(T) = Q_T.$$

This equation is known as the *continuous-time Riccati equation*.

Solution Methods and Applications

In the general case of a nonlinear system, the solution can not be determined analytically and numerical methods need to be relied on. The numerical solution of the Hamilton–Jacobi–Bellman equation is not trivial due to its partial differential nature. Additionally the HJB equation and accompanying numerical methods have been used to solve a wide variety of problems.

See [4] for many applications in the area of optimal control, and for an advocate solution by the *method of characteristics*. This classical technique for the solution of partial differential equations can be found in many textbooks. See [6] for remarks about the application of the HJB equation to minimum time optimal control problems. See [1] for an approximate method for the solution of the time-invariant HJB equation. The method consists of a reduction to a set on linear partial differential equations and an approximation via the *Galerkin spectral method*. It also presents an extensive review of various approximation approaches and an application for the voltage regulation of a power generator. See [7] for an alternating direction algorithm for the solution of HJB equations. See [5] for an application for the optimal path timing of robot manipulators and for the approximate solution of the resulting HJB equation using *finite difference methods*.

The aforementioned references are a subset of the various solution methods for and applications of the Hamilton–Jacobi–Bellman equation.

See also

- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ High-order Maximum Principle for Abnormal Extremals
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming
- ▶ Optimal Control of a Flexible Arm
- ▶ Optimization Strategies for Dynamic Systems
- ▶ Pontryagin Maximum Principle
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Beard RW, Saridis GN, Wen JT (1998) Approximate solutions to the time-invariant Hamilton–Jacobi–Bellman equation. J Optim Th Appl 96(3):589–626
2. Bellman R (1957) Dynamic programming. Princeton Univ. Press, Princeton
3. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont
4. Bryson AE, Ho Y (1975) Applied optimal control. Hemisphere, Washington, DC
5. Cahill AJ, James MR, Kieffer JC, Williamson D (1998) Remarks on the application of dynamic programming to the optimal path timing of robot manipulators. Internat J Robust and Nonlinear Control 8:463–482
6. Evans LC, James MR (1989) The Hamilton–Jacobi–Bellman equation for time-optimal control. SIAM J Control Optim 27(6):1477–1489
7. Sun M (1996) Alternating direction algorithms for solving Hamilton–Jacobi–Bellman equations. Applied Math Optim 34:267–277

Hemivariational Inequalities: Applications in Mechanics

EURIPIDIS MISTAKIDIS[1],
GEORGIOS E. STAVROULAKIS[2]
[1] University Thessaly, Volos, Greece
[2] Carolo Wilhelmina Techn. University, Braunschweig, Germany

MSC2000: 49S05, 74G99, 74H99, 74Pxx, 49J52, 90C33

Article Outline

Keywords
Abstract Hemivariational Inequality
Elastostatics with Nonlinear Boundary Conditions
  Single-Valued Boundary Laws and Variational Equalities
  Multivalued, Monotone Laws and Variational Inequalities
  Multivalued, Nonmonotone Laws and Hemivariational Inequalities
  Inequality or Nonsmooth Mechanics
Discretized Hemivariational Inequalities for Nonlinear Material Laws
Other Applications in Mechanics
Numerical Algorithms
See also
References

Keywords

Hemivariational inequalities; Nonsmooth mechanics; Nonconvex energy function; Generalized subdifferential of F.H. Clarke

Variational expressions, also called for historical reasons *variational principles*, play a significant role in mechanics. They have their origin in the study of problems of analytical mechanics, which have extensively been studied in previous centuries, a time where scientists used to work multidisciplinary. Today, variational principles provide the basis for a correct and efficient modeling of a variety of physical phenomena, for instance, they provide the theoretical basis of the *finite element method* [19].

Variational equalities are the commonly met form of variational expressions. Having in mind problems which can be obtained from the minimization of a smooth (i. e., sufficiently differentiable) potential energy function, one may consider the variation of this function at a given point. A necessary condition for this function to attain a critical point is that every variation of the function in the neighborhood of this point is equal to zero. Thus, one formulates a variational equality problem. In mechanics, the differential of a potential energy function has the physical meaning of (stored or consumed) work. Let us consider a problem in *elastostatics*. In a formulation based on displacements, all variations of the system 's variables around a sought point are called *virtual displacements*. For obvious reasons the variational equality is called in this case *principle of virtual work*: for small virtual displacements around the equilibrium the virtual work of the system is equal to zero. Analogously, one arrives at the principles of complementary virtual work, or at mixed variational principles (the latter being derived from saddle point theorems). At this point it should be mentioned that a variational formulation may also be written for certain classes of problems which does not possess a potential.

The introduction of inequality constraints in the studied problem, or the assumption of nondifferentiable (nonsmooth) potential energy functions, lead to variational inequalities or more complicated variational problems. Intuitively speaking, either not all virtual variations of the problem variables around a given point are permitted (the case of inequality constraints, for instance, unilateral contact constraints), or, a linear approximation of the potential energy function is no more sufficient (the case of nondifferentiable or nonsmooth energy). Convex problems have certain theoretical and numerical advantages. They are connected with monotone operators. This is the case, e. g., of small displacement and deformation elastostatics with monotone material laws or interface and boundary conditions. These problems lead to *variational inequalities* and, in some cases, to convex (possibly nonsmooth) energy minimization problems (convex superpotentials in the sense of J.-J. Moreau [10]). The techniques of *convex analysis* and minimization can be used for their effective solution. Unilateral contact problems [10,15,17] and problems of elasto-

plasticity [7,17] have been studied within this framework.

Hemivariational inequalities are connected with nonconvex and possibly nonsmooth energy functions. In elastostatics, convexity is usually lost if the effects of large displacements or deformations are considered. Moreover, falling branches in material, interface or boundary laws lead to nonconvex potentials. The latter laws may be of a phenomenological nature and may be used for modeling of delamination and strength degradation effects, fracture, etc. Several methods have been developed for the study of nonconvex problems. The notion of the generalized gradient in the sense of F.H. Clarke has been used by P.D. Panagiotopoulos for the construction of hemivariational inequalities [16,17,18]. Following the example of nonsmooth analysis, he called this new field *nonsmooth mechanics*. A short introduction to this theory and its applications in mechanics is outlined in this article. The interested reader may also consult ▶ Nonconvex energy functions: Hemivariational inequalities and the monographs [14,18].

One should mention that the study of hemivariational inequalities provides an interesting field for mathematicians and engineers alike. For engineers several types of hemivariational inequalities have been used for the study and the efficient numerical treatment of yet unsolved or partially solved problems, e. g., in nonmonotone semipermeability problems, in modeling of delamination of simple and multilayered plates, in the theory of composite structures and adhesive joints, etc. Several of these concrete practical applications can not be treated by more naive, without mathematical justification engineering methods. Furthermore, the potential of this research field can be estimated if one thinks that nonconvex energy functions are connected with instabilities, complex dynamics, fractals and chaos. Certainly, a lot of work remains to be done in this area.

Abstract Hemivariational Inequality

The derivation of hemivariational inequalities is based on the mathematical notion of the generalized gradient of Clarke (denoted here by $\overline{\partial}$). In contrast to the variational inequalities, the hemivariational inequalities are not equivalent to minimum problems, but they give rise to substationarity problems. A hemivariational inequality problem reads: find $u \in V$ such as to satisfy the inequality

$$a(u, v-u) + \int_{\Omega} j^0(u, v-u)\, d\Omega \geq (l, v-u), \qquad \forall v \in V. \quad (1)$$

In the abstract form used here, let V be a real Hilbert space, V' be its dual space and such that $V \subset L^2(\Omega) \subset V'$, with continuous and dense injections. The problem is defined in Ω, which is an open bounded subset of $\mathbf{R}^n$. Furthermore let $(\cdot, \cdot)$ be the $L^2(\Omega)$ product and the duality pairing, $\| \cdot \|$ the norm of V and $| \cdot |_2$ the $L^2(\Omega)$-norm. Note that $(\cdot, \cdot)$ extends uniquely from $V \times L^2(\Omega)$ to $V \times V'$. Further, let $V \subset L^2(\Omega)$ be compact and $V \cap L^\infty(\Omega)$ be dense in V for the V-norm, and have a Galerkin base. The bilinear form $a(\cdot, \cdot)$: $V \times V \to \mathbf{R}$ is symmetric continuous and coercive, i. e. there exists $c > 0$ constant such that

$$a(v, v) \geq c \|v\|^2, \quad \forall v \in V. \quad (2)$$

Moreover j: $\mathbf{R} \to \mathbf{R}$ denotes a locally Lipschitz function which is defined by the following procedure: let $\beta \in L^\infty_{\text{loc}}(\mathbf{R})$ and consider

$$\overline{\beta}_\mu(\xi) = \text{esssup}_{|\xi_1 - \xi| \leq \mu} \beta(\xi_1) \quad (3)$$

and

$$\overline{\overline{\beta}}_\mu(\xi) = \text{essinf}_{|\xi_1 - \xi| \leq \mu} \beta(\xi_1). \quad (4)$$

They are increasing and decreasing functions of μ, respectively and thus the limits for $\mu \to 0_+$ exist. We denote them by $\overline{\beta}(\xi)$ and $\overline{\overline{\beta}}(\xi)$ respectively and we define the multivalued function

$$\widehat{\beta}(\xi) = [\overline{\beta}(\xi), \overline{\overline{\beta}}(\xi)]. \quad (5)$$

If $\beta(\xi_{\pm 0})$ exists for every $\xi \in \mathbf{R}$, then a locally Lipschitz function j: $\mathbf{R} \to \mathbf{R}$ can be determined (up to an additive constant) such that $\widehat{\beta}(\xi) = \overline{\partial} j(\xi)$. Finally, in relation (1) $j^0(u, v-u)$ denotes the generalized gradient of the nonconvex and nonsmooth locally Lipschitz potential j. By definition one has the following connection with the generalized gradient, in the sense of Clarke:

$$j^0(u, v) = \{\max \langle w, v \rangle : \ w \in \partial_{\text{CL}} j(u)\}. \quad (6)$$

Speaking in terms of mechanics one identifies relation (1) to be a virtual work expression in inequality form.

The first term is the internal work, the second term is the energy contribution of the nonlinear elements modeled by the nonconvex superpotential j and the right-hand side term represents the loading contribution. Detailed formulations of variational problems, up to the hemivariational inequality (1) for concrete applications follow in the next section.

Elastostatics with Nonlinear Boundary Conditions

A variational formulation is a statement that a solution of an operator equation subjected to certain boundary and/or initial conditions makes an expression involving variations of the quantities of the problems equal to zero or nonnegative. Thus one may distinguish between the bilateral or equality problems and the unilateral or inequality problems. Certain variational principles for a deformable body with nonlinear boundary interaction effects are derived in this section in order to demonstrate the hemivariational inequalities and their relation to classical equations and convex variational inequalities. Let $\Omega \in \mathbf{R}^3$ be an open bounded subset occupied by a deformable body in its undeformed state. On the assumption of small deformations we can write the relation:

$$\int_\Omega \sigma_{ij}(u)\varepsilon_{ij}(v-u)\,d\Omega = \int_\Omega f_i(v_i-u_i)\,d\Omega + \int_\Gamma \sigma_{ij}n_j(v_i-u_i)\,d\Gamma, \quad \forall v \in V, \quad (7)$$

for $u \in V$. Here V denotes the function space of the displacements which will be defined further. Relation (7) is the expression of the principle of virtual work for the body when it is considered free, without constraints on its boundary Γ. For the derivation of (7) the following steps are followed. The elastostatic equilibrium equation is first considered:

$$\sigma_{ij,j} + f_i = 0, \quad (8)$$

where the f_i is the volume force vector. Relation (8) is multiplied by the virtual variation $v_i - u_i$ and then an integration over Ω is performed. On the assumption of appropriately smooth functions, the Green – Gauss theorem is applied. One recalls here the strain-displacement relation (small deformation theory):

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}). \quad (9)$$

Let a linearly elastic body be assumed, i. e., the constitutive material relation reads:

$$\sigma_{ij} = C_{ijhk}\varepsilon_{hk}, \quad (10)$$

where $C = \{ C_{ijhk} \}$, $i, j, h, k = 1, 2, 3$, is the elasticity tensor which satisfies the well-known symmetry and ellipticity properties

$$C_{ijhk} = C_{jihk} = C_{khij}, \quad (11)$$

$$C_{ijhk}\varepsilon_{ij}\varepsilon_{hk} \geq c\varepsilon_{ij}\varepsilon_{hk}, \quad \forall \varepsilon = \{\varepsilon_{ij}\}. \quad (12)$$

The bilinear form of linear elasticity $\alpha(\cdot,\cdot)$ reads in this case:

$$\alpha(u,v) = \int_\Omega C_{ijhk}\varepsilon_{ij}(u)\varepsilon_{hk}(v)\,d\Omega. \quad (13)$$

For further reference one splits the last term in (7) into the work of the normal and of the tangential tractions to the boundary. Then (7) may also be written in the form:

$$\int_\Omega \sigma_{ij}\varepsilon_{ij}(v-u)\,d\Omega = \int_\Omega f_i(v_i-u_i)\,d\Omega + \int_\Gamma S_N(v_N-u_N)\,d\Gamma + \int_\Gamma S_{T_i}(v_{T_i}-u_{T_i})\,d\Gamma, \quad \forall v \in V. \quad (14)$$

Single-Valued Boundary Laws and Variational Equalities

Let us assume first that on Γ the classical boundary conditions $S_N = 0$ and $u_{T_i} = 0$, $i = 1, 2, 3$, hold. Then (14) with (13) leads to the following variational equality:

$$\begin{cases} \text{Find} & u \in V_0 = \{v:\ v \in V,\ v_{T_i} = 0 \text{ on } \Gamma\} \\ \text{s.t.} & \alpha(u,v) = \int_\Omega f_i v_i\,d\Omega, \quad \forall v \in V_0. \end{cases} \quad (15)$$

Analogously, one treats all linear or nonlinear boundary conditions which can be expressed in an equality form. Relation (15), under appropriate smoothness assumptions, imply that the governing equations of the mechanical problem (8) and the assumed boundary conditions hold in a weak (integral or energetic) form.

Multivalued, Monotone Laws and Variational Inequalities

Let us assume now that on Γ the general monotone multivalued boundary condition

$$-S \in \partial j(u) \tag{16}$$

holds. Here $j(u)$ is assumed to be a convex superpotential and ∂denotes the subdifferential of convex analysis. Moreover, all (normal and tangential) contributions of boundary displacements u and tractions S are included in (16), which holds as a multidimensional boundary condition at each point of the boundary Γ. Relation (16) is, by definition of the subdifferential, equivalent to:

$$j(v) - j(u) \geq -S_i(v_i - u_i), \quad \forall v = \{v_i\} \in \mathbf{R}^3. \tag{17}$$

By using (17) and (7) one gets the variational inequality:

$$\begin{cases} \text{Find} & u \in V \text{ with } j(u) < \infty, \\ \text{s.t.} & \alpha(u, v-u) \\ & \quad + \displaystyle\int_\Gamma (j(v) - j(u))\, d\Gamma \\ & \quad \geq \displaystyle\int_\Omega f_i(v_i - u_i)\, d\Omega, \\ & \forall v \in V \text{ with } j(v) < \infty. \end{cases} \tag{18}$$

It is trivial to formulate analogous variational inequalities for more simple one-dimensional laws. This is the case where independent contact laws and tangential (e. g., due to friction) mechanisms are assumed on the boundary Γ. One should mention in passing that unilateral contact relations are included in this formulation by means of the indicator function in the place of $j(u)$. The *indicator function* is defined by $I_{U_{ad}}(u) = 0$ if $u \in U_{\text{ad}}$ and $+\infty$ otherwise, and includes the inequality constraints that describe the no-penetration requirements.

Multivalued, Nonmonotone Laws and Hemivariational Inequalities

In this case the basic building element is the definition of boundary conditions and material laws based on Clarke subdifferential (6). For instance, let on Γ the nonmonotone, possibly multivalued boundary condition

$$-S \in \partial_{\text{CL}} j(u) \tag{19}$$

hold, where j is a locally Lipschitz superpotential functional. Combining (7) with the inequality

$$j^0(u, v-u) \geq -S_i(v_i - u_i), \quad \forall v = \{v_i\} \in \mathbf{R}^3, \tag{20}$$

which defines on Γ the condition (19), one gets the following hemivariational inequality:

$$\begin{cases} \text{Find} & u \in V \\ \text{s.t.} & \alpha(u, v-u) + \\ & \quad + \displaystyle\int_\Gamma j^0(u, v-u)\, d\Gamma \\ & \quad \geq \displaystyle\int_\Omega f_i(v_i - u_i)\, d\Omega, \\ & \forall v \in V. \end{cases} \tag{21}$$

If instead of (19) one assumes on Γ that:

$$-S_N \in \partial_{\text{CL}} j_N(u_N), -S_T \in \partial_{\text{CL}} j_T(u_T), \tag{22}$$

then one gets analogously the hemivariational inequality:

$$\begin{cases} \text{Find} & u \in V \\ \text{s.t.} & \alpha(u, v-u) \\ & \quad + \displaystyle\int_\Gamma j_N^0(u_N, v_N - u_N)\, d\Gamma \\ & \quad + \displaystyle\int_\Gamma j_T^0(u_T, v_T - u_T)\, d\Gamma \\ & \quad \geq \displaystyle\int_\Omega f_i(v_i - u_i)\, d\Omega, \\ & \forall v \in V. \end{cases} \tag{23}$$

The last type of variational expressions involving $j^0(\cdot,\cdot)$ or $j_N^0(\cdot,\cdot)$ and $j_T^0(\cdot,\cdot)$ have been called *hemivariational inequalities* by Panagiotopoulos, who introduced and studied them in mechanics [14,16,17,18]. Note that in the more general case in which j or j_N and j_T are not locally Lipschitz $j^0(\cdot,\cdot)$ in (21) and $j_N^0(\cdot,\cdot)$, $j_T^0(\cdot,\cdot)$ in (23) are replaced by $j^\uparrow(\cdot,\cdot)$ and $j_N^\uparrow(\cdot,\cdot)$, $j_T^\uparrow(\cdot,\cdot)$. Moreover a combination of monotone subdifferential laws (cf. (6)) and nonmonotone laws (cf. (19)) for different (nonoverlapping) parts of the boundary Γ is possible. One then gets variational-hemivariational inequality problems.

The solution of variational problems, like the variational equalities, or the hemivariational inequalities derived previously, satisfies the operator equations of the problem, e. g. the equation of equilibrium, and the boundary conditions of the problem in a weak sense. This means, roughly speaking, that these relations are satisfied in an integral form, on the body or the boundary of the structure respectively. Analogous considerations are familiar within the weak formulations used in the finite element method.

Inequality or Nonsmooth Mechanics

A boundary value problem is called *bilateral* (resp. *unilateral*) if it leads to variational equality (resp. variational, or hemivariational inequality) formulations. The unilateral problems are called *inequality problems* too. Inequality problems in mechanics usually characterize structures with variable mechanical behavior, i. e. where the material or boundary law depends on the direction of the stress or boundary traction variation. Due to their connection with nonsmooth energy functions, all inequality problems belong to the area called by Panagiotopoulos *nonsmooth mechanics* [11,12].

Discretized Hemivariational Inequalities for Nonlinear Material Laws

In order to make the subject more accessible to engineers a discretized hemivariational inequality is formulated in this section. A finite element discretization is assumed. All relations are written in an elementary matrix analysis form. An elastic structure with both classical, linearly elastic and degrading elements is considered.

The *stress equilibrium equations* read:

$$\overline{G}\overline{s} = \begin{pmatrix} G & G_n \end{pmatrix} \begin{pmatrix} s \\ s_n \end{pmatrix} = p \tag{24}$$

where $\overline{G}$ is the equilibrium matrix of the discretized structure which takes into account the stress contribution of the linear s and nonlinear s_n elements and p is the loading vector.

The *strain-displacement compatibility equations* take the form:

$$\overline{e} = \begin{pmatrix} e \\ e_n \end{pmatrix} = \overline{G}^\top u = \begin{pmatrix} G^\top \\ G_n^\top \end{pmatrix} u, \tag{25}$$

where e, u are the deformation and displacement vectors respectively.

The linear material constitutive law for the structure reads:

$$s = K_0(e - e_0), \tag{26}$$

where K_0 is the natural and stiffness flexibility matrix and e_0 is the initial deformation vector.

The nonlinear material law is considered in the form:

$$s_n \in \partial_{\mathrm{CL}}\phi_n(e_n). \tag{27}$$

Here $\phi_n(\cdot)$, is a general nonconvex superpotential and summation over all nonlinear elements gives the total strain energy contribution of them as:

$$\Phi_n(e_n) = \sum_{i=1}^{q} \phi_n^{(i)}(e_n). \tag{28}$$

Finally classical support boundary conditions complete the description of the problem.

The discretized form of the virtual work equation reads:

$$s^\top(e^* - e) + s_n^\top(e_n^* - e_n) = p^\top(u^* - u), \quad \forall\, e^*,\ u^*,\ e_n^*. \tag{29}$$

Entering the elasticity law (26) into the virtual work equation (29), and using (25) we get:

$$u^\top G K_0^\top G^\top(u^* - u) - (p + GK_0e_0)^\top(u^* - u) + s_n^\top(e_n^* - e_n) = 0, \forall u^* \in V_{\mathrm{ad}}, \tag{30}$$

where $K = G^\top K_0^\top G$ denotes the stiffness matrix of the structure, $\overline{p} = p + GK_0 e_0$ denotes the nodal equivalent loading vector and V_{ad} includes all support boundary conditions of the structure.

Further one considers the nonlinear elements (27) in the inequality form:

$$s_n^\top (e_n^* - e_n) \le \Phi_n^o(e_n^* - e_n), \quad \forall e_n^*, \tag{31}$$

where $\Phi_n^o(e_n^* - e_n)$ is the directional derivative of the potential Φ_n. Thus the following discretized hemivariational inequality is obtained:

$$\begin{cases} \text{Find} & \text{kinematically admissible} \\ & \quad \text{displacements } u \in V_{\text{ad}} \\ \text{s.t.} & u^\top K(u^* - u) - \overline{p}^\top (u^* - u) \\ & \quad + \Phi_n^o(u_n^* - u_n) \ge 0, \\ & \forall u^* \in V_{\text{ad}}. \end{cases} \tag{32}$$

Equivalently a substationarity problem for the total potential energy can be written:

$$\begin{cases} \text{Find} & u \in V_{\text{ad}} \\ \text{s.t.} & \Pi(u) = \text{stat}_{v \in V_{\text{ad}}} \{\Pi(v)\}. \end{cases} \tag{33}$$

Here the potential energy reads $\Pi(v) = \frac{1}{2} v^\top K v - \overline{p}^\top v + \Phi_n(v)$, where the first two terms (quadratic potential) are well-known in the structural analysis community.

Other Applications in Mechanics

Hemivariational inequalities have been used for the modeling and solution of delamination effects in composite and multilayered plates, in composite structures, for nonmonotone friction and skin effects and for nonlinear mechanics applications (for instance, in the analysis of semi-rigid joints in steel structures). Details can be found in [9,11,12,17,18] and in the citations given there. Another area of applications are nonconvex problems arising in elastoplasticity (cf. [4,5,6]). Some nonconvex problems in elastoplasticity have been treated by hemivariational inequality techniques in [17,18]. Mathematical results which are useful for the study of hemivariational inequalities can also be found in [2,3,13,14].

Numerical Algorithms

A number of algorithms based on nonsmooth and nonconvex optimization concepts, on engineering methods or heuristics and on combination of these two approaches have been tested till now for the numerical solution of hemivariational inequality problems. Both finite elements and boundary elements have been used, the latter for boundary only nonlinear problems; see ▶ Nonconvex energy functions: Hemivariational inequalities and [1,8,9,17].

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods

- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
2. Goeleven D (1996) Noncoercive variational problems and related results. Addison-Wesley and Longman
3. Haslinger J, Miettinen M, Panagiotopoulos PD (1999) Finite element method for hemivariational inequalities. Kluwer, Dordrecht
4. Kim SJ, Oden JT (1984) Generalized potentials in finite elastoplasticity. Part I. Internat J Eng Sci 22:1235–1257
5. Kim SJ, Oden JT (1985) Generalized potentials in finite elastoplasticity. Part II. Internat J Eng Sci 23:515–530
6. Kuczma MS, Stein E (1994) On nonconvex problems in the theory of plasticity. Arch Mechanicky 46(4):603–627
7. Maier G, Novati G (1990) Extremum theorems for finite-step backward-difference analysis of elastic-plastic nonlinearly hardening solids. Internat J Plasticity 6:1–10
8. Miettinen M, Mäkelä MM, Haslinger J (1995) On numerical solution of hemivariational inequalities by nonsmooth optimization methods. J Global Optim 8(4):401–425
9. Mistakidis ES, Stavroulakis GE (1998) Nonconvex optimization in mechanics. Algorithms, heuristics and engineering applications by the F.E.M. Kluwer, Dordrecht
10. Moreau JJ (1968) La notion de sur-potentiel et les liaisons unilatérales enélastostatique. CR 267A:954–957
11. Moreau JJ, Panagiotopoulos PD (eds) (1988) Nonsmooth mechanics and applications. CISM, vol 302. Springer, Berlin
12. Moreau JJ, Panagiotopoulos PD, Strang G (eds) (1988) Topics in nonsmooth mechanics. Birkhäuser, Basel
13. Motreanu D, Panagiotopoulos PD (1999) Minimax theorems and qualitative properties of the solutions of hemivariational inequalities. Kluwer, Dordrecht
14. Naniewicz Z, Panagiotopoulos PD (1995) Mathematical theory of hemivariational inequalities and applications. M. Dekker, New York
15. Oden JT, Kikuchi N (1988) Contact problems in elasticity: A study of variational inequalities and finite element methods. SIAM, Philadelphia
16. Panagiotopoulos PD (1983) Nonconvex energy functions. Hemivariational inequalities and substationary principles. Acta Mechanics 42:160–183
17. Panagiotopoulos PD (1985) Inequality problems in mechanics and applications. Convex and nonconvex energy functions. Birkhäuser, Basel
18. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Berlin
19. Washizu K (1968) Variational methods in elasticity and plasticity. Pergamon, Oxford

Hemivariational Inequalities: Eigenvalue Problems

Daniel Goeleven[1], Dumitru Motreanu[2]

[1] I.R.E.M.I.A., University de la Réunion, Saint-Denis, France

[2] Department Mat., University Al.I.Cuza, Iasi, Romania

MSC2000: 49J52

Article Outline

Keywords
See also
References

Keywords

Eigenvalue problem; Hemivariational inequalities; Critical point theory; Unilateral mechanics

The theory of *hemivariational inequalities* has been created by P.D. Panagiotopoulos et al. (see [3,5,6,7]) for studying nonconvex and nonsmooth energy functions under nonmonotone multivalued laws. In this setting many relevant models lead to *nonsmooth eigenvalue problems*. A typical example is provided by the analysis of hysteresis phenomena. To illustrate it we present here the loading and unloading problems with *hysteresis* modes.

Consider a plane linear elastic body Ω with the boundary Γ whose mechanical behavior is described by the virtual displacement variable u and the scalar parameter λ which determines the magnitude of the external loading on the system. The variable u must satisfy certain boundary or support conditions. For the sake of simplicity we assume that $u = 0$ on Γ, so the space of kinematically admissible displacements u is the *Sobolev*

space $H_0^1(\Omega)$, that is the closure of $C_0^\infty(\Omega)$ with respect to the L^2-norm of the gradient. Let us suppose that there exist a fundamental (pre-bifurcation) solution $\lambda \longmapsto u_0(\lambda)$ and another solution $\lambda \longmapsto u(\lambda) = u_0(\lambda) + z(\lambda)$ that coincide for $\lambda < \lambda_0$. Then one has $\lim_{\lambda\to\lambda_0} z(\lambda) = 0$ and the hysteresis bifurcation mode has the expression

$$u_1(\lambda_0) := \lim_{\lambda\to\lambda_0} \|z(\lambda)\|^{-1} z(\lambda). \tag{1}$$

Using the principle of virtual works together with physically realistic assumptions on the data θ and S (see e. g. [7]), we obtain the relation

$$a(u_1(\lambda_0), v) + \langle S(u_1(\lambda_0)), v\rangle - \lambda_0 \int_\Omega u_1(\lambda_0) v dx = 0, \quad \forall v \in H_0^1(\Omega). \tag{2}$$

It is justified to accept that a generalized nonmonotone reaction-displacement $(-S, u)$ holds in Ω expressed by the next law

$$\int_\Omega j^o(u_1(\lambda_0); v)\, dx \geq \langle S(u_1(\lambda_0)), v\rangle\,, \quad \forall v \in H_0^1(\Omega), \tag{3}$$

where j: $\mathbf{R} \to \mathbf{R}$ stands for a *locally Lipschitz function* with the *generalized gradient* ∂j and the *generalized directional derivative*

$$j^o(x; y) = \max\{\langle z, y\rangle : z \in \partial j(x)\}$$

(see [2]). Relations (2) and (3) yield the following eigenvalue problem in hemivariational inequality form: Find $(u = u(\lambda), \lambda) \in H_0^1(\Omega) \times \mathbf{R}$ such that

$$a(u, v) + \int_\Omega j^o(u; v)\, dx \geq \lambda \int_\Omega uv\, dx, \quad \forall v \in H_0^1(\Omega). \tag{4}$$

Additional information concerning problems of type (4) can be found in [3,5,6,7].

Relation (4), as well as other models, motivates the study of abstract eigenvalue problems for hemivariational inequalities. The specific case of Problem (4) can be reformulated as follows: given a Banach space V embedded in $L^2(\Omega)$, i. e. the space of square-integrable functions on $\Omega \subset \mathbf{R}^N$, a continuous symmetric bilinear form a: $V \times V \to \mathbf{R}$ and a locally Lipschitz function j: $\mathbf{R} \to \mathbf{R}$ with an appropriate growth condition for its generalized gradient, find $u \in V$ and $\lambda \in \mathbf{R}$ such that

$$a(u, v) + \int_\Omega j^o(u; v)\, dx \geq \lambda \int_\Omega uv dx, \quad \forall v \in V. \tag{5}$$

Note that this last mathematical model can also be used to formulate various other problems in Mechanics like unilateral bending problems in elasticity.

A general approach for studying the abstract eigenvalue problem (5) is the nonsmooth critical point theory as developed by K.-C. Chang [1]. In that paper the *minimax principles* in the critical point theory are extended from the smooth functionals (see [8]) to the case of locally Lipschitz functionals. In this respect we associate to Problem (5), for each λ, the locally Lipschitz functional I_λ: $V \to \mathbf{R}$,

$$I_\lambda(u) = \frac{1}{2} a(u, u) + \int_\Omega j(u)\, dx - \frac{\lambda}{2} \int_\Omega u^2\, dx, \quad \forall u \in V. \tag{6}$$

Note that a *critical point* u of I_λ, i. e. $0 \in \partial I_\lambda\,(u)$, is a solution of (5) because

$$\partial I_\lambda(u) \subset a(u, \cdot) - \lambda(u, \cdot)_{L^2} + \partial \int_\Omega j(u)\, dx \subset a(u, \cdot) - \lambda(u, \cdot)_{L^2} + \int_\Omega \partial j(u)\, dx$$

(see [2]). Thus, to solve (5), it suffices to establish the existence of nontrivial critical points of the functional I_λ introduced in (6). To this end we proceed along the lines in [4] by arguing in an abstract framework.

Given a Banach space V and a bounded domain Ω in $\mathbf{R}^m$, $m \geq 1$, let T: $V \to L^s(\Omega;\mathbf{R}^N)$ be a *compact linear operator*, where $L^s(\Omega;\mathbf{R}^N)$ stands for the Banach space of all Lebesgue measurable functions f: $\Omega \to \mathbf{R}^N$ for which $|f|^s$ is integrable with $1 < s < \infty$. Let F: $V \to \mathbf{R}$ be a locally Lipschitz function and let G: $\Omega \times \mathbf{R}^N \to \mathbf{R}$ be a (Carathéodory) function such that $G(x, y)$ is measurable in $x \in \Omega$, locally Lipschitz in $y \in \mathbf{R}^N$ and $G(x, 0) = F(0) = 0$, $x \in \Omega$. The hypotheses below are imposed

H1) $|w| \le c(1 + |y|^{s-1})$, $\forall w \in \partial_y G(x,y)$, $x \in \Omega$, $y \in \mathbf{R}^N$, with a constant $c > 0$;

H2) i) $F(v) - r\langle z, v_V\rangle \ge \alpha \|v\|_V^{\sigma} - \alpha_0$, $\forall v \in V$, $z \in \partial F(v)$;

ii) $G(x,y) - r\langle w, y\rangle \ge -b|y|^{\sigma_0} - b_0$, for a.e. $x \in \Omega$, $y \in \mathbf{R}^N$, $w \in \partial_y G(x,y)$, with positive constants $r, \alpha, \alpha_0, b, b_0, \sigma, \sigma_0$, where $1 \le \sigma_0 < \min\{\sigma, r^{-1}, s\}$;

H3) any bounded sequence $\{v_n\} \subset V$ for which there is $z_n \in \partial F(v_n)$ converging in V^* contains a convergent subsequence in V;

H4) i) $\liminf_{v\to 0} F(v)\|v\|_V^{-p} > 0$;

ii)

$$\liminf_{v\to 0} F(v)\|v\|_V^{-p} + |\Omega|^{(s-p)/p}\|T\|^p \liminf_{y\to 0} G(x,y)|y|^{-p} > 0$$

uniformly with respect to x, $1 \le p < s$;

H5)

$$\liminf_{t\to+\infty} F(tv_0)t^{-1/r} < -\liminf_{t\to+\infty} t^{-1/r}\int_\Omega G(x, tTv_0)\,dx$$

for some $v_0 \in V$.

The following statement is our main result in studying the abstract eigenvalue problem (5).

Theorem 1 *Assume that the hypotheses H1)–H5) hold. Then there exists a nontrivial critical point $u \in V$ of $I: V \to \mathbf{R}$ defined by*

$$I(v) = F(v) + \int_\Omega G(x,(Tv)(x))\,dx\,, \quad v \in V.$$

Moreover, there exists $z \in \partial F(u)$ and $w \in L^{s(s-1)}(\Omega;\mathbf{R}^N)$ such that

$$w(x) \in \partial_y G(x,(Tu)(x)) \quad \text{a.e. } x \in \Omega,$$
$$\langle z, v\rangle_V + \int_\Omega \langle w(x),(Tv)(x)\rangle\,dx = 0\,, \quad v \in V.$$

Conversely, if $u \in V$ verifies the relations above, corresponding to some z and w, and the function $G(x,\cdot)$ is regular at $(Tu)(x)$ (in the sense of F.H. Clarke [2]) for each $x \in \Omega$, then u is a critical point of I.

The foregoing locally Lipschitz functional I satisfies the Palais–Smale condition in the sense of Chang [1]. Indeed, let (v_n) be a sequence in V with $I(v_n) \le M$ and for which there exists a sequence $J_n \in \partial I(v_n)$ with $J_n \to 0$ in V^*. Then from H2) and taking into account that

$$J_n = z_n + T^* w_n,$$
$$z_n \in \partial F(v_n),$$
$$w_n(x) \in \partial_y G(x,(Tv_n)(x)) \quad \text{a.e. } x \in \Omega,$$

we infer that

$$M + r\|v_n\|_V \ge F(v_n) - r\langle z_n, v_n\rangle_V + \int_\Omega (G(x,(Tv_n)(x)) - r\langle w_n(x),(Tv_n)(x)\rangle)\,dx \ge \alpha\|v_n\|_V^{\sigma} + C_1\|v_n\|_V^{\sigma_0} + C_2,$$

with real constants C_1, C_2, provided that n is large enough. It is clear that the estimate above implies that the sequence (v_n) is bounded in V. Then a standard argument based on the assumption H3) allows to conclude that (v_n) possesses a strongly convergent subsequence. Namely, the boundedness of (v_n) implies that (Tv_n) is bounded in $L^s(\Omega;\mathbf{R}^N)$. Thus (w_n) is bounded in $L^{s/s-1)}(\Omega;\mathbf{R}^N)$ due essentially to the assumption H1). Since T^* is a compact operator and $J_n \to 0$ we derive that (z_n) has a convergent subsequence in V^*. This fact combined with the boundedness of (v_n) allows to use the hypothesis H3). The claim that the locally Lipschitz functional I verifies the Palais –Smale condition is proved.

Assumption H4) insures the existence of some constants $\delta > 0$, $A > 0$ and $B > 0$, with

$$A - B|\Omega|^{(s-p)/p}\|T\|^p > 0,$$

such that

$$F(v) \ge A\|v\|_V^p\,, \qquad \|v\|_V \le \delta, \tag{7}$$

and

$$G(x,y) \ge -B|y|^p\,, \quad \forall x \in \Omega, \quad |y| \le \delta.$$

Combining the inequality above with H1) one obtains that

$$\int_\Omega G(x,(Tv)(x))\,dx \ge -(A-\eta)\|v\|_V^p\,, \qquad \|v\|_V \le \rho, \tag{8}$$

for some $\eta > 0$ and $0 < \rho \le \delta$. Indeed, assumption H1) and Lebourg 's mean value theorem imply that G fulfills the following growth condition

$$|G(x,y)| \le a_1 + a_2 |y|^s, \forall x \in \Omega, \quad y \in \mathbf{R}^N,$$

with constants $a_1, a_2 \ge 0$. The two estimates above for $G(x,y)$ show that

$$G(x,y) \ge -B|y|^p - (a_1\delta^{-s} + a_2)|y|^s, \quad \forall x \in \Omega, \quad y \in \mathbf{R}^N.$$

Then one deduces from the continuity of T that one has

$$\int_\Omega G(x,(Tv)(x))dx \ge \left(-B|\Omega|^{(s-p)/p}\|T\|^p - (a_1\delta^{-s} + a_2)\|T\|^s \|v\|_V^{s-p}\right)\|v\|_V^p, \quad \forall v \in V.$$

Since $s > p$ we see that the numbers $\eta > 0$ and $\rho > 0$ can be chosen so small that relation (8) be verified.

By (7) and (8) we arrive at the conclusion that there exist positive numbers ρ, η such that

$$I(v) \ge \eta, \quad \|v\|_V = \rho. \tag{9}$$

The formula

$$\partial_t(t^{-1/r}G(x,ty)) = \frac{1}{r}t^{-1-1/r}[r\langle \partial_y G(x,ty), ty\rangle - G(x,ty)],$$

the absolute continuity property and H2ii) show that

$$t^{-1/r}G(x,ty) - G(x,y) = \int_1^t \partial_\tau(\tau^{-1/r}(G(x,\tau y)d\tau \le C|y|^{\sigma_0} + C_0$$

for a.e. $x \in \Omega$, $y \in \mathbf{R}^N$, $t > 1$, where C, C_0 are positive constants. Then one obtains

$$I(t\theta v_0) \le (t\theta)^{1/r} \times \left[F(t\theta v_0)(t\theta)^{-1/r} + \overline{C}\|v_0\|_V^{\sigma_0}\theta^{\sigma_0 - 1/r} + \overline{C}_0\theta^{-1/r} + \theta^{-1/r}\int_\Omega G(x,\theta(Tv_0)(x))\,dx\right]$$

for all $t > 1$, $\theta > 1$, with new positive constants $\overline{C}$, $\overline{C}_0$. In view of H5) and since $\sigma_0 < 1/r$, we can find θ sufficiently large such that

$$\overline{C}\|v_0\|_V^{\sigma_0}\theta^{\sigma_0 - 1/r} + \overline{C}_0\theta^{-1/r} + \theta^{-1/r}\int_\Omega G(x,\theta(Tv_0)(x))\,dx < -\liminf_{\tau\to+\infty} F(\tau v_0)\tau^{-1/r}.$$

With such fixed number θ, we see that there exists arbitrarily large t satisfying

$$F(t\theta v_0)(t\theta)^{-1/r} + \overline{C}\|v_0\|_V^{\sigma_0}\theta^{\sigma_0 - 1/r} + \overline{C}_0\theta^{-1/r} + \theta^{-1/r}\int_\Omega G(x,\theta(Tv_0)(x))\,dx < 0.$$

We deduce that

$$I(t_n v_0) \le 0 \tag{10}$$

for a subsequence $t_n \to \infty$. The properties (9) and (10) permit to apply the *mountain pass theorem* in the nonsmooth version of Chang [1]. This yields the desired critical point u of I. The other assertions of the first part of Theorem are direct consequences of the last statement.

The converse part of Theorem follows from the next formula

$$\partial\int_\Omega G(x,u(x))\,dx = \int_\Omega \partial_y G(x,u(x))\,dx, \quad \forall u \in L^s(\Omega;\mathbf{R}^N),$$

which is valid under the growth condition in H1) and the regularity assumption for G (see [2]). The proof of Theorem is thus complete.

In the case of problem (4) we choose $V = H_0^1(\Omega)$, the compact linear operator T: $H_0^1(\Omega) \to L^s(\Omega)$ equal to the embedding $H_0^1(\Omega) \subset L^s(\Omega)$ with $2 < s < 2m(m-2)^{-1}$ if $m \ge 3$,

$$F(v) = \frac{1}{2}\int_\Omega (|\nabla v|^2 - \lambda v^2)\,dx, \quad \forall v \in H_0^1(\Omega),$$

where for simplicity we take $a(u,v) = \int_\Omega \nabla u \cdot \nabla v\,dx$, and $G(x,t) = j(t)$. A significant possible choice for j is the following one

$$j(t) = -\frac{|t|^s}{s} + \int_0^t \beta(\tau)\,d\tau, \quad t \in \mathbf{R}, \tag{11}$$

where $\beta \in L^\infty_{loc}(\mathbf{R})$ verifies $t\,\beta(t) \ge 0$ for t near 0, $|\beta(t)| \le c(1 + |t|^\gamma)$, $t \in \mathbf{R}$, with constants $c > 0$, $0 \le \gamma < 1$.

Corollary 2 *Let $j: \boldsymbol{R} \to \boldsymbol{R}$ be given by (11). If λ_1 denotes the first eigenvalue of $-\Delta$ on $H_0^1(\Omega)$, then for every $\lambda < \lambda_1$ the problem (5) with a as above, has a nontrivial eigenfunction $u \in H_0^1(\Omega)$ which solves in addition the nonsmooth Dirichlet problem containing both superlinear and sublinear terms*

$$\Delta u + \lambda u + |u|^{s-2} u \in [\underline{\beta}(u(x)), \overline{\beta}(u(x))]$$
$$\text{a.e. } x \in \Omega, u = 0 \quad \text{on } \partial\Omega,$$

where the notations in [1] are used.

The argument consists in verifying the assumptions H1)–H5) for the functional $I = I_\lambda$, for $\lambda < \lambda_1$, with I_λ described in (6). To this end it is sufficient to take $r \in (1/s, 1/2)$, $p = \sigma = 2$, $\sigma_0 = \gamma + 1$ and $v_0 \in H_0^1(\Omega) \{ 0 \}$. Applying Theorem one finds the stated result.

Other related results and applications for eigenvalue problems in the form of hemivariational inequalities are given in [3,4,5,6,7] and the references therein.

See also

- ▶ αBB Algorithm
- ▶ Eigenvalue Enclosures for Ordinary Differential Equations
- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Semidefinite Programming and Determinant Maximization
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Chang K-C (1981) Variational methods for non-differentiable functionals and their applications to partial differential equations. J Math Anal Appl 80:102–129
2. Clarke FH (1984) Nonsmooth analysis and optimization. Wiley, New York
3. Goeleven D, Motreanu D, Panagiotopoulos PD (1997) Multiple solutions for a class of eigenvalue problems in hemivariational inequalities. Nonlinear Anal Th Methods Appl 29:9–26
4. Motreanu D (1995) Existence of critical points in a general setting. Set-Valued Anal, 3:295–305
5. Motreanu D, Panagiotopoulos PD (1999) Minimax theorems and qualitative properties of the solutions of hemivariational inequalities. Kluwer, Dordrecht
6. Naniewicz Z, Panagiotopoulos PD (1995) The mathematical theory of hemivariational inequalities and applications. M. Dekker, New York
7. Panagiotopoulos PD (1993) Hemivariational inequalities. applications in mechanics and engineering. Springer, Berlin
8. Rabinowitz PH (1986) Minimax methods in critical point theory with applications to differential equations, vol 65. CBMS Reg. Conf. Ser. Math., Amer. Math. Soc., Providence

Hemivariational Inequalities: Static Problems

HVI

Zdzisław Naniewicz[1,2]
[1] Institute Appl. Math. Mech., Warsaw University, Warsaw, Poland
[2] Institute Math. Comp. Science, Techn. University Czestochowa, Czestochowa, Poland

MSC2000: 49J40, 47J20, 49J40, 35A15

Article Outline

Keywords
References

Keywords

Semicoercive hemivariational inequality; Unilateral growth condition; Pseudomonotone mapping; Recession functional

Let $V = H^1(\Omega;\mathbf{R}^N)$, $N \geq 1$, be a vector valued Sobolev space of functions square integrable together with their first partial distributional derivatives in Ω, Ω being a bounded domain in $\mathbf{R}^m$, $m > 2$, with sufficiently smooth boundary Γ. Assume that V is compactly imbedded into $L^p(\Omega;\mathbf{R}^N)$ $(1 < p < 2m/m - 2)$, [12]). We write $\| \cdot \|_V$ and $\| \cdot \|_{L^p(\Omega;R^N)}$ for the norms in V and $L^p(\Omega;\mathbf{R}^N)$, respectively. For the pairing over $V^* \times V$ the symbol $\langle \cdot, \cdot \rangle_V$ will be used, V^* being the dual of V.

Let A: $V \to V^*$ be a bounded, *pseudomonotone operator*. This means that A maps bounded sets into bounded sets and that the following conditions hold [3,5]:

i) The effective domain of A coincides with the whole V;

ii) If $u_n \to u$ weakly in V and $\limsup_{n\to\infty} \langle Au_n, u_n - u_V \rangle \leq 0$, then $\liminf_{n\to\infty} \langle Au_n, u_n - v_V \rangle \geq \langle Au, u - v \rangle_V$ for any $v \in V$.
Note that i) and ii) imply that A is demicontinuous, i. e.

iii) If $u_n \to u$ strongly in V, then $Au_n \to Au$ weakly in V^*.

Moreover, we assume that V is endowed with a direct sum decomposition $V = \widehat{V} + V_0$, where V_0 is a finite-dimensional linear subspace, with respect to which A is semicoercive, i. e. $\forall u \in V$ there exist $\widehat{u} \in \widehat{V}$ and $\theta \in V_0$ such that $u = \widehat{u} + \theta$ and

$$\langle Au, u \rangle_V \geq c(\|\widehat{u}\|_V)\, \|\widehat{u}\|_V\,, \tag{1}$$

where c: $\mathbf{R}^+ \to \mathbf{R}$ stands for a coercivity function with $c(r) \to \infty$ as $r \to \infty$. Further, let j: $\mathbf{R}^N \to \mathbf{R}$ be a locally Lipschitz function fulfilling the *unilateral growth conditions* ([16,21]):

$$j^0(\xi;\eta - \xi) \leq \alpha(r)(1 + |\xi|^\sigma), \qquad \forall \xi, \eta \in \mathbf{R}^N,\ |\eta| \leq r,\ r \geq 0, \tag{2}$$

and

$$j^0(\xi;-\xi) \leq k\,|\xi|\,, \quad \forall \xi \in \mathbf{R}^N, \tag{3}$$

where $1 \leq \sigma < p$, k is a nonnegative constant and α :$\mathbf{R}^+ \to \mathbf{R}^+$ is assumed to be a nondecreasing function from $\mathbf{R}^+$ into $\mathbf{R}^+$. Here, $j^0(\cdot;\cdot)$ stands for the *directional Clarke derivative*

$$j^0(\xi;\eta) = \limsup_{\substack{h\to 0\\ \lambda \to 0_+}} \frac{j(\xi + h + \lambda\eta) - j(\xi + h)}{\lambda}, \tag{4}$$

by means of which the *Clarke generalized gradient* of j is defined by [6]

$$\partial j(\xi) := \{\mu \in \mathbf{R}^N :\ j^0(\xi;\eta) \geq \mu \cdot \eta,\ \forall \eta \in \mathbf{R}^N\}\,, \qquad \xi, \eta \in \mathbf{R}^N.$$

Remark 1 The unilateral growth condition (2) is the generalization of the well known sign condition used for the study of nonlinear partial differential equations in the case of scalar-valued function spaces (cf. [27,28]).

Consider the problem of finding $u \in V$ such as to satisfy the hemivariational inequality

$$\langle Au - g, v - u \rangle_V + \int_\Omega j^0(u; v - u)\, d\Omega \geq 0, \qquad \forall v \in V. \tag{5}$$

It will be assumed that $g \in V^*$ fulfills the *compatibility condition*

$$\langle g, \theta \rangle_V < \int_\Omega j^\infty(\theta)\, d\Omega, \quad \forall \theta \in V_0 \setminus \{0\}, \tag{6}$$

where j^∞: $\mathbf{R}^N \to \mathbf{R} \cup \{+\infty\}$ stands for the *recession functional* given by (cf. [2,4,10])

$$j^\infty(\xi) = \liminf_{\substack{\eta\to\xi \\ t\to+\infty}} [-j^0(t\eta;-\eta)], \quad \xi \in \mathbf{R}^N. \tag{7}$$

Because of (1), the problem to be considered here will be referred to as a *semicoercive hemivariational inequality*.

The notion of hemivariational inequality has been first introduced by P.D. Panagiotopoulos in [22,23] for the description of important problems in physics and engineering, where nonmonotone, multivalued boundary or interface conditions occur, or where some nonmonotone, multivalued relations between stress and strain, or reaction and displacement have to be taken into account. The theory of hemivariational inequalities (as the generalization of variational inequalities, cf. [7]) has been proved to be very useful in understanding of many problems of mechanics involving nonconvex, nonsmooth energy functionals. For the general study of hemivariational inequalities and their applications, see [13,14,15,17,18,19,20,21,24,26] and the references quoted there. Some results in the area of static, semicoercive inequality problems can be found in [9,10,25].

To prove the existence of solutions to (5), the Galerkin method combined with the pseudomonotone regularization of the nonlinearities will be applied.

Let us start with the following preliminary results.

The regularization $\widetilde{j}^0_R(\cdot;\cdot)$, $R>0$, of the Clarke directional derivative $j^0(\cdot;\cdot)$ will be defined as follows: for any $\xi, \eta \in \mathbf{R}^N$, set

$$\widetilde{j}^0_R(\xi,\eta) = \begin{cases} j^0(\xi;\eta) & \text{if } |\xi| \le R, \\ j^0\left(R\frac{\xi}{|\xi|};\eta\right) & \text{if } |\xi| > R. \end{cases} \tag{8}$$

Lemma 2 *Suppose that (2) and (3) are fulfilled. Then for $R>0$,*

$$\widetilde{j}^0_R(\xi;\eta-\xi) \le \widetilde{\alpha}(r)(1+|\xi|^\sigma), \quad \forall \xi \in \mathbf{R}^N, \quad \forall \eta \in \mathbf{R}^N,\ |\eta| \le r,\ r \ge 0. \tag{9}$$

$$\widetilde{j}^0_R(\xi;-\xi) \le k|\xi|, \quad \forall \xi \in \mathbf{R}^N, \tag{10}$$

where $\widetilde{\alpha}$: $\mathbf{R}^+ \to \mathbf{R}^+$ is a nondecreasing function independent of R.

Proof To establish (9) and (10) it suffices to consider the case $|\xi| \ge R$ and to invoke the estimates

$$\begin{aligned} \widetilde{j}^0_R(\xi;\eta-\xi) &= j^0\left(R\frac{\xi}{|\xi|};\eta-\xi\right) \\ &\le j^0\left(R\frac{\xi}{|\xi|};\eta-R\frac{\xi}{|\xi|}\right) \\ &\quad + \frac{|\xi|-R}{R} j^0\left(R\frac{\xi}{|\xi|};-R\frac{\xi}{|\xi|}\right) \\ &\le \alpha(|\eta|)(1+R^\sigma) + \frac{|\xi|-R}{R}kR \\ &\le \alpha(r)(1+|\xi|^\sigma) + k|\xi|, \\ &\forall \xi,\eta \in \mathbf{R}^N,\ |\eta| \le r,\ r \ge 0, \end{aligned}$$

and

$$\begin{aligned} \widetilde{j}^0_R(\xi;-\xi) &= j^0\left(R\frac{\xi}{|\xi|};-\xi\right) \\ &\le \frac{|\xi|}{R} j^0\left(R\frac{\xi}{|\xi|};-R\frac{\xi}{|\xi|}\right) \le \frac{|\xi|}{R}kR = k|\xi|, \end{aligned}$$

respectively. The proof is complete.

For any $R>0$, the following regularization of the primal problem can be formulated:

(P_R) Find $(u_R, \chi_R) \in V \times L^q(\Omega;\mathbf{R}^N)$,

1/p +1/q = 1, such that

$$\langle Au_R - g, v-u_R\rangle_V + \int_\Omega \chi_R \cdot (v-u_R)\,d\Omega = 0, \quad \forall v \in V, \tag{11}$$

$$\chi_R \in \Gamma_R(u_R), \tag{12}$$

where

$$\Gamma_R(u_R) := \left\{ \psi \in L^q(\Omega;\mathbf{R}^N) : \int_\Omega \psi \cdot v\,d\Omega \le \int_\Omega \widetilde{j}^0_R(u_R;v)d\Omega,\ \forall\, v \in L^p(\Omega;\mathbf{R}^N) \right\}.$$

In order to show that (P_R) has solutions, the following auxiliary result is to be applied.

Lemma 3 *Suppose that (1)-(3) and (6) hold. Then there exists $R_0 > 0$ such that for any $R > R_0$ the set of all $u \in V$ with the property that*

$$\langle Au - g, u\rangle_V - \int_\Omega \widetilde{j}_R^0(u; -u)\, d\Omega \le 0 \tag{13}$$

is bounded in V, i. e. there exists $\mathcal{M} > 0$ (possibly depending on $R > R_0$), such that (13) implies

$$\|u\|_V \le \mathcal{M}. \tag{14}$$

Proof Suppose on the contrary that this claim is not true, i. e. there exists a sequence $\{u_n\}_{n=1}^\infty \subset V$ with the property that

$$\langle Au_n - g, u_n\rangle_V - \int_\Omega \widetilde{j}_R^0(u_n; -u_n)\, d\Omega \le 0, \tag{15}$$

where $\|u_n\|_V \to \infty$ as $n \to \infty$. By the hypothesis, each element u_n can be represented as

$$u_n = \widehat{u}_n + e_n\theta_n, \tag{16}$$

where $\widehat{u}_n \in \widehat{V}$, $e_n \ge 0$, $\theta_n \in V_0$, $\|\theta_n\|_V = 1$, and $\langle Au_n, u_n\rangle_V \ge c(\|\widehat{u}_n\|_V)\|\widehat{u}_n\|_V$). Taking into account (3) it follows that

$$\begin{aligned}
0 &\ge \langle Au_n - g, u_n\rangle_V - \int_\Omega \widetilde{j}_R^0(u_n; -u_n)\, d\Omega \\
&\ge c(\|\widehat{u}_n\|_V)\|\widehat{u}_n\|_V - \|g\|_{V^*}\|\widehat{u}_n\|_V \\
&\quad - e_n\langle g, \theta_n\rangle_V - k\int_\Omega |u_n|\, d\Omega \\
&\ge c(\|\widehat{u}_n\|_V)\|\widehat{u}_n\|_V - \|g\|_{V^*}(\|\widehat{u}_n\|_V + e_n) \\
&\quad - k_1\|\widehat{u}_n\|_V - e_n k_1\|\theta_n\|_V,
\end{aligned} \tag{17}$$

where $k_1 = \text{const}$. The obtained estimates imply that $\{e_n\}$ is unbounded. Indeed, if it would not be so, then due to the behavior of $c(\cdot)$ at infinity, $\{\widehat{u}_n\}$ had to be bounded. In such a case the contradiction with $\|u_n\|_V \to \infty$ as $n \to \infty$ results. Therefore one can suppose without loss of generality that $e_n \to +\infty$ as $n \to \infty$. The next claim is that

$$\frac{1}{e_n}\widehat{u}_n \to 0 \quad \text{strongly in } V. \tag{18}$$

Indeed, if $\{\|\widehat{u}_n\|_V\}$ is bounded, then (18) follows immediately. If $\|\widehat{u}_n\|_V \to \infty$ then $c(\|\widehat{u}_n\|_V) \to +\infty$. From (17) one has

$$k_1 + \|g\|_{V^*} \ge \left(c(\|\widehat{u}_n\|_V) - \|g\|_{V^*} - k_1\right)\frac{\|\widehat{u}_n\|_V}{e_n}.$$

Thus, the boundedness of the sequence

$$\left\{\left(c(\|\widehat{u}_n\|_V) - \|g\|_{V^*} - k_1\right)\frac{\|\widehat{u}_n\|_V}{e_n}\right\}_{n=1}^\infty$$

results, which in view of

$$c(\|\widehat{u}_n\|_V) - \|g\|_{V^*} - k_1 \to +\infty \quad \text{as } n \to \infty$$

implies the assertion (18). The obtained results give rise to the following representation of u_n:

$$u_n = e_n\left(\frac{1}{e_n}\widehat{u}_n + \theta_n\right),$$

where $\widehat{u}_n/e_n \to 0$ strongly in V and $\theta_n \to \theta$ in V_0 as $n \to \infty$ for some $\theta \in V_0$ with $\|\theta\|_V = 1$ (recall that V_0 has been assumed to be finite dimensional). Moreover, the compact imbedding $V \subset L^p(\Omega;\mathbf{R}^N)$ permits one to suppose that $\widehat{u}_n/e_n \to 0$ and $\theta_n \to \theta$ a.e. in Ω.

Further, (15), together with the fact that A is semicoercive, leads to

$$\begin{aligned}
0 &\ge \langle Au_n - g, u_n\rangle_V - \int_\Omega \widetilde{j}_R^0(u_n; -u_n)d\Omega \\
&\ge \left(c(\|\widehat{u}_n\|_V) - \|g\|_{V^*}\right)\|\widehat{u}_n\|_V - e_n\langle g, \theta_n\rangle_V \\
&\quad + e_n \\
&\quad \cdot \int_\Omega -\widetilde{j}_R^0\left(e_n\left(\frac{1}{e_n}\widehat{u}_n + \theta_n\right); -\frac{1}{e_n}\widehat{u}_n - \theta_n\right) d\Omega.
\end{aligned}$$

Hence

$$\begin{aligned}
\langle g, \theta_n\rangle_V &\ge \left(c(\|\widehat{u}_n\|_V) - \|g\|_{V^*}\right)\frac{1}{e_n}\|\widehat{u}_n\|_V \\
&\quad + \int_\Omega -\widetilde{j}_R^0\left(e_n\left(\frac{\widehat{u}_n}{e_n} + \theta_n\right); -\frac{\widehat{u}_n}{e_n} - \theta_n\right) d\Omega.
\end{aligned} \tag{19}$$

Now observe that either

$$\left(c(\|\widehat{u}_n\|_V) - \|g\|_{V^*}\right)\frac{1}{e_n}\|\widehat{u}_n\|_V \to 0 \quad \text{as } n \to \infty,$$

if $\{\|\widehat{u}_n\|_V\}$ is bounded, or

$$\left(c(\|\widehat{u}_n\|_V) - \|g\|_{V^*}\right)\frac{1}{e_n}\|\widehat{u}_n\|_V \geq 0$$

for sufficiently large n, if $\|\widehat{u}_n\|_V \to \infty$ as $n \to \infty$. Therefore, for any case

$$\liminf_{n\to\infty}\left(c(\|\widehat{u}_n\|_V) - \|g\|_{V^*}\right)\frac{1}{e_n}\|\widehat{u}_n\|_V \geq 0.$$

Moreover, by (10) the estimate follows:

$$-\widetilde{j}_R^0\left(e_n\left(\frac{\widehat{u}_n}{e_n}+\theta_n\right);-\frac{\widehat{u}_n}{e_n}-\theta_n\right) \geq -k\left|\frac{\widehat{u}_n}{e_n}+\theta_n\right|. \quad (20)$$

This allows the application of Fatou 's lemma in (19), from which one is led to

$$\begin{aligned}\langle g,\theta\rangle_V &\geq \liminf_{n\to\infty}\int_\Omega\left[-\widetilde{j}_R^0\left(e_n\left(\frac{\widehat{u}_n}{e_n}+\theta_n\right);-\frac{\widehat{u}_n}{e_n}-\theta_n\right)\right]d\Omega\\ &\geq \int_\Omega \liminf_{n\to\infty}\left[-\widetilde{j}_R^0\left(e_n\left(\frac{\widehat{u}_n}{e_n}+\theta_n\right);-\frac{\widehat{u}_n}{e_n}-\theta_n\right)\right]d\Omega.\end{aligned} \quad (21)$$

Taking into account (8) and upper semicontinuity of $j^0(\cdot,\cdot)$, one can easily verify that

$$\begin{aligned}&\liminf_{n\to\infty}\left[-\widetilde{j}_R^0\left(e_n(\frac{\widehat{u}_n}{e_n}+\theta_n);-\frac{\widehat{u}_n}{e_n}-\theta_n\right)\right]\\ &\geq -j^0\left(R\frac{\theta}{|\theta|};-\theta\right),\end{aligned}$$

which leads to

$$\langle g,\theta\rangle_V \geq \int_\Omega -j^0\left(R\frac{\theta}{|\theta|};-\theta\right)d\Omega. \quad (22)$$

Since $j^\infty(\cdot)$ is lower semicontinuous and V_0 is finite dimensional, from (6) it follows that a $\delta > 0$ can be found such that for any $\theta \in V_0$ with $\|\theta\|_V\| = 1$,

$$\langle g,\theta\rangle_V + \delta < \int_\Omega j^\infty(\theta)\,d\Omega. \quad (23)$$

With the help of Fatou 's lemma (permitted by (20)) we arrive at

$$\liminf_{R\to\infty}\int_\Omega -j^0\left(\frac{R}{|\theta|}\theta;-\theta\right)d\Omega \geq \int_\Omega j^\infty(\theta)\,d\Omega.$$

The upper semicontinuity of $j^0(\cdot;\cdot)$ allows us to conclude the existence of $R_\theta > 0$ and $\varepsilon_\theta > 0$ such that

$$\int_\Omega -j^0\left(\frac{R}{|\theta'|}\theta';-\theta'\right)d\Omega \geq \int_\Omega j^\infty(\theta)\,d\Omega - \frac{\delta}{2}$$

for each $R > R_\theta$ and $\theta' \in V_0$ with $\|\theta - \theta'\|_V\| < \varepsilon_\theta$. As the sphere $\{v \in V_0 : \|v\|_V\| = 1\}$ is compact in V_0, there exists $R_0 > 0$ such that

$$\int_\Omega -j^0\left(\frac{R}{|\theta|}\theta;-\theta\right)d\Omega \geq \int_\Omega j^\infty(\theta)\,d\Omega - \frac{\delta}{2},$$

for any $\theta \in V_0$ with $\|\theta\|_V\| = 1$, $R > R_0$. This combined with (23) contradicts (22). Accordingly, the existence of a constant $\mathcal{M} > 0$ has been established such that (13) implies (14), whenever $R > R_0$. The proof of Lemma 3 is complete.

Proposition 4 *Let us assume all the hypotheses stated above. Then for any $R > R_0$ the problem (P_R) possesses at least one solution. Moreover, if (u_R, χ_R) is a solution of (P_R), then*

$$\|u_R\|_V\| \leq M \quad (24)$$

for some constant M not depending on $R > R_0$.

Proof Let Λ be the family of all finite-dimensional subspaces F of V, ordered by inclusion. Denote by i_F: $F \to V$ the inclusion mapping of F into V and by i_F^*: $V^* \to F^*$ the dual projection mapping of V^* into F^*, F^* being the dual of F. The pairing over $F^* \times F$ will be denoted by $\langle\cdot,\cdot\rangle_F$. Set $A_F := i_F^* \circ A \circ i_F$ and $g_F := i_F^* g$.

Fix $R > R_0$. For any $F \in \Lambda$ consider a finite-dimensional regularization of (P_R):

(P_F) Find $(u_F, \chi_F) \in F \times L^q(\Omega;\mathbf{R}^N) \in F$

such that

$$\langle Au_F - g, v\rangle_V + \int_\Omega \chi_F \cdot v\,d\Omega = 0, \forall v \in F, \quad (25)$$

$$\chi_F \in \Gamma_R(u_F). \quad (26)$$

The first task is to show that for each $F \in \Lambda$, (P_F) has solutions. Notice that $\Gamma_R(\cdot)$ has nonempty, convex and closed values and if $\psi \in \Gamma_R(v)$, $v \in L^p(\Omega;\mathbf{R}^N)$, then

$$\|\psi\|_{L^q(\Omega;\mathbf{R}^N)} \leq K_R, \tag{27}$$

for some $K_R > 0$ depending on the Lipschitz constant of j in the ball $\{\eta \in \mathbf{R}^N : |\eta| \leq \mathbf{R}\}$. Moreover, from the upper semicontinuity of $j_R^0(\cdot;\cdot)$ and Fatou 's lemma it follows immediately that Γ_R is upper semicontinuous from $L^p(\Omega;\mathbf{R}^N)$ to $L^q(\Omega;\mathbf{R}^N)$, $L^q(\Omega;\mathbf{R}^N)$ being endowed with the weak topology.

Further, let $\tau_F: L^q(\Omega;\mathbf{R}^N) \to F^*$ be the operator that to any $\psi \in L^q(\Omega;\mathbf{R}^N)$ assigns $\tau_F \psi \in F^*$ defined by

$$\langle \tau_F \psi, v\rangle_F := \int_\Omega \psi \cdot v \, d\Omega \quad \text{for any } v \in F. \tag{28}$$

Note that τ_F is a linear and continuous operator from the weak topology of $L^q(\Omega;\mathbf{R}^N)$ to the (unique) topology on F^*. Therefore $G_F: F \to 2^{F^*}$, given by the formula

$$G_F(v_F) := \tau_F \Gamma_R(v_F) \quad \text{for } v_F \in F, \tag{29}$$

is upper semicontinuous.

By the pseudomonotonicity of A it follows that $A_F: F \to F^*$ is continuous. Thus, $A_F + G_F - g_F: F \to 2^{F^*}$ is an upper semicontinuous multivalued mapping with nonempty, bounded, closed and convex values. Moreover, for any $v_F \in F$ and $\psi_F \in G_F(v_F)$ one has

$$\begin{aligned}\langle A_F v_F + \psi_F - g_F, v_F\rangle_F \\ \geq \langle Av_F - g, v_F\rangle_V - \int_\Omega \widetilde{j_R^0}(v_F; -v_F)\, d\Omega.\end{aligned} \tag{30}$$

Hence, in view of Lemma 3, for $R > R_0$ there exists $\mathcal{M} > 0$ not depending on $F \in \Lambda$ such that the condition $\|v_F\|_V = \mathcal{M} + 1$ implies

$$\langle A_F v_F + \psi_F - g_F, v_F\rangle_F \geq 0. \tag{31}$$

Accordingly, one can invoke [1, Corol. 3, p. 337] to deduce the existence of $u_F \in F$ with

$$\|u_F\|_V \leq \mathcal{M} + 1 \tag{32}$$

such that $0 \in A_F u_F + G_F(u_F) - g_F$. This implies that for some $\chi_F \in \Gamma_R(u_F)$ it follows that $\psi_F = \tau_F(\chi_F)$ and (u_F, χ_F) is a solution of (P_F).

In the next step it will be shown that (P_R), $R > R_0$, has solutions.

For $F \in \Lambda$, let

$$\mathcal{W}_F := \bigcup_{\substack{F' \in \Lambda,\\ F' \supset F}} \left\{ u_{F'} \in V : \begin{array}{c}(u_{F'}, \chi_{F'})\\ \text{satisfies } (P_{F'})\\ \text{for some}\\ \chi_{F'} \in L^q(\Omega;\mathbf{R}^N)\end{array} \right\}.$$

The symbol weakcl $(\mathcal{W}_F)$ will be used to denote the closure of $\mathcal{W}_F$ in the weak topology of V. From (32) one gets

$$\text{weakcl}(\mathcal{W}_F) \subset B_V(O, \mathcal{M} + 1), \quad \forall F \in \Lambda,$$

where $B_V(O, \mathcal{M} + 1) := \{v \in V : \|v\|_V \leq \mathcal{M} + 1\}$. Thus, the family $\{\text{weakcl}(\mathcal{W}_F): F \in \Lambda\}$ is contained in the weakly compact set $B_V(O, \mathcal{M} + 1)$ of V. Further, for any $F_1, \ldots, F_k \in \Lambda$, $k = 1, 2, \ldots$, the inclusion $\mathcal{W}_{F_1} \cap \cdots \cap \mathcal{W}_{F_k} \supset \mathcal{W}_F$ results, with $F = F_1 + \cdots + F_k$. Therefore, the family $\{\text{weakcl}(\mathcal{W}_F): F \in \Lambda\}$ has the finite intersection property. This implies that $\cap_{F \in \Lambda}$ weakcl $(\mathcal{W}_F)$ is not empty. From now on, let $u_R \in B_V(0, \mathcal{M} + 1)$ belong to this intersection.

Fix $v \in V$ arbitrarily and choose $F \in \Lambda$ such that u_R, $v \in F$. Thus, there exists a sequence $\{u_{F_n}\} \subset \mathcal{W}_F$ with $u_{F_n} \to u_R$ weakly in V. Let $\chi_{F_n} \in \Gamma_R(u_{F_n})$ denote the corresponding sequence for which (u_{F_n}, χ_{F_n}) is a solution of (P_{F_n}) (for simplicity of notation, the symbols $\{u_n\}$ and $\{\chi_n\}$ will be used instead of u_{F_n} and χ_{F_n}, respectively). Therefore

$$\begin{aligned}\langle Au_n - g, w - u_n\rangle_V + \int_\Omega \chi_n \cdot (w - u_n)\, d\Omega = 0,\\ \forall\, w \in F_n.\end{aligned} \tag{33}$$

Since $\|\chi_n\|_{L^q(\Omega;\mathbf{R}^N)} \leq K_R$ and $L^q(\Omega;\mathbf{R}^N)$ is reflexive, it can also be supposed that for some $\chi_R \in L^q(\Omega;\mathbf{R}^N)$, $\chi_n \to \chi_R$ weakly in $L^q(\Omega;\mathbf{R}^N)$. By the hypothesis, the imbedding $V \subset L^p(\Omega;\mathbf{R}^N)$ is compact, so $u_n \to u_R$ strongly in $L^p(\Omega;\mathbf{R}^N)$. Consequently, by the upper semicontinuity of Γ_R from $L^p(\Omega;\mathbf{R}^N)$ to $L^q(\Omega;\mathbf{R}^N)$ ($L;^q(\Omega;\mathbf{R}^N)$ being endowed with the weak topology) it follows immediately that $\chi_R \in \Gamma_R(u_R)$, i. e. (12) holds. Moreover, $\int_\Omega \chi_n \cdot (u_R - u_n) d\Omega \to 0$ as $n \to \infty$ and (33) with $w = u_R$ lead to $\lim \langle Au_n, u_n - u_R\rangle_V = 0$. Accordingly, the pseudomonotonicity of A allows the conclusion that $\langle Au_n, u_n\rangle_V \to \langle Au_R, u_R\rangle_V$ and $Au_n \to Au_R$

weakly in V^*. Finally, substituting $w = v$ in (33) and letting $n \to \infty$ give in conclusion (11) with $v \in V$ chosen arbitrarily. Thus the existence of solutions of (P_R) has been established.

Let us proceed to the boundedness of solutions $\{u_R\}_{R>R_0}$ of (P_R). Suppose on the contrary that this claim is not true. Then according to (11) and (12) there would exist a sequence $R_n \to \infty$ such that $\| u_{R_n} \|_V \to \infty$ as $n \to \infty$, and

$$\langle Au_{R_n} - g, u_{R_n} \rangle_V - \int_\Omega \widetilde{j^0_{R_n}}(u_{R_n}; -u_{R_n})\, d\Omega \le 0. \quad (34)$$

From now on, for simplicity of notations, instead of the subscript 'R_n' we write 'n'. Eq. (34) allows us to follow the lines of the proof of Lemma 3. First, analogously one arrives at the representation

$$u_n = e_n \left(\frac{1}{e_n} \widehat{u}_n + \theta_n \right),$$

with $\widehat{u}_n / e_n \to 0$ strongly in V and $\theta_n \to \theta_0$ in V_0 as $n \to \infty$ for some $\theta_0 \in V_0$ with $\| \theta_0 \|_V = 1$. Secondly, the counterpart of (21) can be obtained in the form

$$\langle g, \theta \rangle_V \ge \limsup_{n\to\infty} \int_\Omega \left[-\widetilde{j^0_{R_n}} \left(e_n \left(\frac{1}{e_n} \widehat{u}_n + \theta_n \right); -\frac{1}{e_n} \widehat{u}_n - \theta_n \right) \right] d\Omega. \quad (35)$$

But

$$\widetilde{j^0_{R_n}} \left(e_n \left(\frac{1}{e_n} \widehat{u}_n + \theta_n \right); -\frac{1}{e_n} \widehat{u}_n - \theta_n \right) = j^0 \left(e_n \left(\frac{1}{e_n} \widehat{u}_n + \theta_n \right); -\frac{1}{e_n} \widehat{u}_n - \theta_n \right),$$

if $\left| \widehat{u}_n + e_n \theta_n \right| \le R_n$ and

$$\widetilde{j^0_{R_n}} \left(e_n \left(\frac{1}{e_n} \widehat{u}_n + \theta_n \right); -\frac{1}{e_n} \widehat{u}_n - \theta_n \right) = j^0 \left(\frac{R_n}{\left| \frac{1}{e_n} \widehat{u}_n + \theta_n \right|} \left(\frac{1}{e_n} \widehat{u}_n + \theta_n \right); -\frac{1}{e_n} \widehat{u}_n - \theta_n \right),$$

if $\left| \widehat{u}_n + e_n \theta_n \right| > R_n$.. Therefore we easily conclude, using (7), that

$$\limsup_{n\to\infty} -\widetilde{j^0_{R_n}} \left(e_n \left(\frac{1}{e_n} \widehat{u}_n + \theta_n \right); -\frac{1}{e_n} \widehat{u}_n - \theta_n \right) \ge j^\infty(\theta_0).$$

Consequently, by Fatou 's lemma,

$$\langle g, \theta_0 \rangle_V \ge \int_\Omega j^\infty(\theta_0)\, d\Omega,$$

contrary to (6). Thus, the boundedness of $\{u_R\}_{R>R_0}$ follows and the proof of Proposition 4 is complete.

The next result is related to the compactness property of $\{ \chi_R : R > R_0 \}$ in $L^1(\Omega; \mathbf{R}^N)$.

Proposition 5 *Let a pair $(u_R, \chi_R) \in V \times L^q(\Omega; \mathbf{R}^N)$ be a solution of (P_R). Then the set*

$$\left\{ \chi_R \in L^q(\Omega; \mathbf{R}^N) : \begin{array}{c} (u_R, \chi_R) \\ \text{is a solution of} \\ (P_R) \\ \text{for some } u_R \in V, \\ R > R_0 \end{array} \right\}$$

is weakly precompact in $L^1(\Omega; \mathbf{R}^N)$.

Proof According to the Dunford–Pettis theorem [8] it is sufficient to show that for each $\varepsilon > 0$ a $\delta > 0$ can be determined such that for any $\omega \subset \Omega$ with meas $\omega < \delta$,

$$\int_\omega |\chi_R|\, d\Omega < \varepsilon, \quad R > R_0. \quad (36)$$

Fix $r > 0$ and let $\eta \in \mathbf{R}^N$ be such that $| \eta | \le r$. Then, by (9), from $\chi_R \cdot (\eta - u_R) \le \widetilde{j^0_R}(u_R; \eta - u_R)$ it results that

$$\chi_R \cdot \eta \le \chi_R \cdot u_R + \widetilde{\alpha}(r)(1 + |u_R|^\sigma) \quad (37)$$

a.e. in Ω. Let us set

$$\eta \equiv \frac{r}{\sqrt{N}} (\operatorname{sgn} \chi_{R_1}, \ldots, \operatorname{sgn} \chi_{R_N}),$$

where χ_{R_i}, $i = 1, \ldots, N$, are the components of χ_R and where $\operatorname{sgn} y = 1$ if $y > 0$, $\operatorname{sgn} y = 0$ if $y = 0$, and $\operatorname{sgn} y = -1$ if $y < 0$. It is not difficult to verify that $| \eta | \le r$ for almost all $x \in \Omega$ and that

$$\chi_R \cdot \eta \ge \frac{r}{\sqrt{N}} |\chi_R|.$$

Therefore, by (37) the estimate follows

$$\frac{r}{\sqrt{N}} |\chi_R| \le \chi_R \cdot u_R + \widetilde{\alpha}(r)(1 + |u_R|^\sigma).$$

Integrating this inequality over $\omega \subset \Omega$ yields

$$\int_\omega |\chi_R|\, d\Omega \le \frac{\sqrt{N}}{r}\int_\omega \chi_R \cdot u_R\, d\Omega + \frac{\sqrt{N}}{r}\widetilde{\alpha}(r)\,\mathrm{meas}\,\omega + \frac{\sqrt{N}}{r}\widetilde{\alpha}(r)(\mathrm{meas}\,\omega)^{(p-\sigma)/p}\,\|u_R\|^{\sigma}_{L^p(\Omega)}\, d\Omega. \quad (38)$$

Thus, from (24) one obtains

$$\begin{aligned}\int_\omega |\chi_R|\, d\Omega &\le \frac{\sqrt{N}}{r}\int_\omega \chi_R \cdot u_R\, d\Omega + \frac{\sqrt{N}}{r}\widetilde{\alpha}(r)\,\mathrm{meas}\,\omega \\ &\quad + \frac{\sqrt{N}}{r}\widetilde{\alpha}(r)(\mathrm{meas}\,\omega)^{(p-\sigma)/p}\gamma^{\sigma}\,\|u_R\|^{\sigma}_{V}\, d\Omega \\ &\le \frac{\sqrt{N}}{r}\int_\omega \chi_R \cdot u_R\, d\Omega + \frac{\sqrt{N}}{r}\widetilde{\alpha}(r)\,\mathrm{meas}\,\omega \\ &\quad + \frac{\sqrt{N}}{r}\widetilde{\alpha}(r)(\mathrm{meas}\,\omega)^{(p-\sigma)/p}\gamma^{\sigma} M^{\sigma}\, d\Omega \qquad (39)\end{aligned}$$

($\|\cdot\|_{L^p(\Omega;R^N)} \le \gamma\,\|\cdot\|_V$).

Further, it will be shown that

$$\int_\omega \chi_R \cdot u_R\, d\Omega \le C \qquad (40)$$

for some positive constant C not depending on $\omega \subset \Omega$ and $R > R_0$. Indeed, from (10) one can easily deduce that

$$\chi_R \cdot u_R + k\,|u_R| \ge 0 \quad \text{a.e. in } \Omega.$$

Thus it follows that

$$\int_\omega (\chi_R \cdot u_R + k\,|u_R|)\, d\Omega \le \int_\Omega (\chi_R \cdot u_R + k\,|u_R|)\, d\Omega,$$

and consequently

$$\int_\omega \chi_R \cdot u_R\, d\Omega \le \int_\Omega \chi_R \cdot u_R\, d\Omega + 2k_1\,\|u_R\|_V.$$

But A maps bounded sets into bounded sets. Therefore, by means of (11) and (24),

$$\begin{aligned}\int_\Omega \chi_R \cdot u_R\, d\Omega &= -\langle Au_R - g, u_R\rangle_V \\ &\le \|Au_R - g\|_{V^*}\,\|u_R\|_V \le C_0, \quad C_0 = \text{const},\end{aligned}$$

and consequently, (40) easily follows. Further, from (39) and (40), for $r > 0$,

$$\int_\omega |\chi_R|\, d\Omega \le \frac{\sqrt{N}}{r}C + \frac{\sqrt{N}}{r}\widetilde{\alpha}(r)\,\mathrm{meas}\,\omega + \frac{\sqrt{N}}{r}\widetilde{\alpha}(r)(\mathrm{meas}\,\omega)^{(p-\sigma)/p}\gamma^{\sigma} M^{\sigma}\, d\Omega. \quad (41)$$

This estimate is crucial for obtaining (36). Namely, let $\varepsilon > 0$. Fix $r > 0$ with

$$\frac{\sqrt{N}}{r}C < \frac{\varepsilon}{2} \qquad (42)$$

and determine $\delta > 0$ small enough to fulfill

$$\frac{\sqrt{N}}{r}\widetilde{\alpha}(r)\,\mathrm{meas}\,\omega + \frac{\sqrt{N}}{r}\widetilde{\alpha}(r)(\mathrm{meas}\,\omega)^{(p-\sigma)/p}\gamma^{\sigma} M^{\sigma} \le \frac{\varepsilon}{2},$$

provided that meas $\omega < \delta$. Thus, from (41) it follows that for any $\omega \subset \Omega$ with meas $\omega < \delta$,

$$\int_\omega |\chi_R|\, d\Omega \le \varepsilon, \quad R > R_0. \qquad (43)$$

Finally, $\{\chi_R\}_{R>R_0}$ is equi-integrable and its precompactness in $L^1(\Omega;\mathbf{R}^N)$ has been proved [8].

Now the main result will be formulated.

Theorem 6 *Let $A: V \to V^*$ be a pseudomonotone, bounded operator, $j: \mathbf{R}^N \to \mathbf{R}$ a locally Lipschitz function. Suppose that (1)-(3) and (6) hold. Then there exist $u \in V$ and $\chi \in L^1(\Omega;\mathbf{R}^N)$ such that*

$$\langle Au - g, v - u\rangle_V + \int_\Omega \chi \cdot (v - u)\, d\Omega = 0, \quad \forall v \in V \cap L^\infty(\Omega;\mathbf{R}^N), \quad (44)$$

$$\begin{cases}\chi \in \partial j(u) & \text{a.e. in } \Omega,\\ \chi \cdot u \in L^1(\Omega).\end{cases} \qquad (45)$$

Moreover, the hemivariational inequality holds:

$$\langle Au - g, v - u\rangle_V + \int_\Omega j^0(u; v - u)\, d\Omega \ge 0, \quad \forall v \in V, \quad (46)$$

where the integral above is assumed to take $+\infty$ as its value if $j^0(u;v-u) \notin L^1(\Omega)$.

Proof The proof is divided into a sequence of steps.

Step 1. From Propositions 4 and 5 it follows that from the set $\{ u_R, \chi_R \}_{R>R_0}$ of solutions of (P_R) a sequence $\{ u_{R_n}, \chi_{R_n} \}$ can be extracted with $R_n \to \infty$ as $n \to \infty$ (for simplicity of notations it will be denoted by (u_n, χ_n)), such that

$$\langle Au_n - g, v - u_n \rangle_V + \int_\Omega \chi_n \cdot (v - u_n)\, d\Omega = 0, \forall v \in V, \quad (47)$$

and

$$\begin{cases} \chi_n \in \Gamma_{R_n}(u_n), & \\ u_n \to u & \text{weakly in } V, \\ \chi_n \to \chi & \text{weakly in } L^1(\Omega;\mathbf{R}^N) \end{cases} \quad (48)$$

for some $u \in V$ and $\chi \in L^1(\Omega;\mathbf{R}^N)$.

The boundedness of $\{Au_n\}$ in V^* (recall that A has been assumed to be bounded and that $\| u_n \|_V \le M$) allows the conclusion that for some $B \in V^*$,

$$Au_n \to B \quad \text{weakly in } V^* \quad (49)$$

(by passing to a subsequence, if necessary). Thus, (47) implies that the equality

$$\langle B - g, v \rangle_V + \int_\Omega \chi \cdot v\, d\Omega = 0 \quad (50)$$

is valid for any $v \in V \cap L^\infty(\Omega;\mathbf{R}^N)$.

Step 2. Now it will be proved that $\chi \in \partial j(u)$ a.e in Ω, i. e. the first condition in (45) is fulfilled. Since V is compactly imbedded into $L^p(\Omega;\mathbf{R}^N)$, due to (48) one may suppose that

$$u_n \to u \quad \text{strongly in } L^p(\Omega;\mathbf{R}^N). \quad (51)$$

This implies that for a subsequence of $\{u_n\}$ (again denoted by the same symbol) one gets $u_n \to u$ a.e. in Ω. Thus, from Egoroff 's theorem it follows that for any $\varepsilon > 0$ a subset $\omega \subset \Omega$ with meas $\omega < \varepsilon$ can be determined such that $u_n \to u$ uniformly in $\Omega \setminus \omega$ with $u \in L^\infty(\Omega \setminus \omega;\mathbf{R}^N)$. Let $v \in L^\infty(\Omega \setminus \omega;\mathbf{R}^N)$ be an arbitrary function. From the estimate

$$\int_{\Omega\setminus\omega} \chi_n \cdot v\, d\Omega \le \int_{\Omega\setminus\omega} \widetilde{j^0_{R^n}}(u_n; v)\, d\Omega = \int_{\Omega\setminus\omega} j^0(u_n; v)\, d\Omega, \quad \text{(for large } n)$$

(u_n remains pointwise uniformly bounded in $\Omega \setminus \omega$ and $R_n \to \infty$ as $n \to \infty$) combined with the weak convergence in $L^1(\Omega;\mathbf{R}^N)$ of χ_n to χ, (51) and with the upper semicontinuity of

$$L^\infty(\Omega \setminus \omega;\mathbf{R}^N) \ni u_n \longmapsto \int_{\Omega\setminus\omega} j^0(u_n; v)\, d\Omega,$$

it follows that

$$\int_{\Omega\setminus\omega} \chi \cdot v\, d\Omega \le \int_{\Omega\setminus\omega} j^0(u; v)\, d\Omega, \quad \forall v \in L^\infty(\Omega \setminus \omega;\mathbf{R}^N).$$

But the last inequality allows us to state that $\chi \in \partial j(u)$ a.e. in $\Omega \setminus \omega$. Since meas $\omega < \varepsilon$ and ε was chosen arbitrarily,

$$\chi \in \partial j(u) \quad \text{a.e. in } \Omega, \quad (52)$$

as claimed.

Step 3. Now it will be shown that $\chi \cdot u \in L^1(\Omega)$, i. e. the second condition in (45) holds. For this purpose we shall need the following truncation result for vector-valued Sobolev spaces.

Theorem 7 ([20]) *For each $v \in H^1(\Omega;\mathbf{R}^N)$ there exists a sequence of functions $\{ \varepsilon_n \} \subset L^\infty(\Omega)$ with $0 \le \varepsilon_n \le 1$ such that*

$$\begin{aligned} &\{(1-\varepsilon_n)v\} \subset H^1(\Omega;\mathbf{R}^N) \cap L^\infty(\Omega;\mathbf{R}^N) \\ &(1-\varepsilon_n)v \to v \quad \textit{strongly in } H^1(\Omega;\mathbf{R}^N). \end{aligned} \quad (53)$$

Remark 8 For the truncation procedure of the form (53) in the case of a scalar-valued Sobolev space $W^{p,m}(\Omega)$ the reader is referred to [11].

According to the aforementioned theorem, for $u \in V$ one can find a sequence $\{ \varepsilon_k \} \in L^\infty(\Omega)$ with $0 \le \varepsilon_k \le 1$ such that $\widetilde{u}_k := (1-\varepsilon_k)u \in V \cap L^\infty(\Omega;\mathbf{R}^N)$ and $\widetilde{u}_k \to u$ in V as $k \to \infty$. Without loss of generality it can be assumed that $\widetilde{u}_k \to u$ a.e. in Ω. Since it is already

known that $\chi \in \partial j(u)$, one can apply (3) to obtain $\chi \cdot (-u) \le j^0(u;-u) \le k\,|u|$. Hence

$$\chi \cdot \widetilde{u}_k = (1-\varepsilon_k)\chi \cdot u \ge -k\,|u|\,. \tag{54}$$

This implies that the sequence $\{\chi \cdot \widetilde{u}_k\}$ is bounded from below and $\chi \cdot \widetilde{u}_k \to \chi \cdot u$ a.e. in Ω. On the other hand, due to (50) one gets

$$C \ge \langle -B+g, \widetilde{u}_k\rangle_V = \int_\Omega \chi \cdot \widetilde{u}_k \, d\Omega$$

for a positive constant C. Thus, by Fatou 's lemma $\chi \cdot u \in L^1(\Omega)$, as required.

Step 4. Now the inequality

$$\liminf_{n\to\infty} \int_\Omega \chi_n \cdot u_n \, d\Omega \ge \int_\Omega \chi \cdot u \, d\Omega \tag{55}$$

will be established. It can be supposed that $u_n \to u$ a.e. in Ω, because $u_n \to u$ strongly in $L^p(\Omega;\mathbf{R}^N)$. Fix $v \in L^\infty(\Omega;\mathbf{R}^N)$ arbitrarily. Since $\chi_n \in \Gamma_{R_n}(u_n)$, Eq. (9) implies

$$\begin{aligned}\chi_n \cdot (v-u_n) &\le \widetilde{j^0_{R^n}}(u_n; v-u_n)\\ &\le \widetilde{\alpha}(\|v\|_{L^\infty(\Omega;\mathbf{R}^N)})(1+|u_n|^\sigma).\end{aligned} \tag{56}$$

From Egoroff 's theorem it follows that for any $\varepsilon > 0$ a subset $\omega \subset \Omega$ with meas $\omega < \varepsilon$ can be determined such that $u_n \to u$ uniformly in $\Omega \setminus \omega$. One can also suppose that ω is small enough to fulfill $\int_\omega \widetilde{\alpha}(\|v\|_{L^\infty(\Omega;\mathbf{R}^N)})(1+|u_n|^\sigma)\,d\Omega \le \varepsilon$, $n = 1, 2, \ldots$, and $\int_\omega \alpha(\|v\|_{L^\infty(\Omega;\mathbf{R}^N)})(1+\|u^\sigma)\,d\Omega \le \varepsilon$. Hence

$$\begin{aligned}&\int_\Omega \widetilde{j^0_{R^n}}(u_n; v-u_n)\,d\Omega\\ &\le \int_{\Omega\setminus\omega} \widetilde{j^0_{R^n}}(u_n; v-u_n)\,d\Omega + \varepsilon\\ &= \int_{\Omega\setminus\omega} j^0(u_n; v-u_n)\,d\Omega + \varepsilon \quad \text{(for large } n),\end{aligned}$$

which by Fatou 's lemma and upper semicontinuity of $j^0(\cdot;\cdot)$ yields

$$\begin{aligned}&\liminf_{n\to\infty} \int_\Omega -\widetilde{j^0_{R^n}}(u_n; v-u_n)\,d\Omega\\ &\ge \int_\Omega -j^0(u; v-u)\,d\Omega - 2\varepsilon.\end{aligned}$$

By arbitrariness of $\varepsilon > 0$ and (56) one obtains

$$\begin{aligned}&\liminf_{n\to\infty} \int_\Omega \chi_n \cdot u_n \, d\Omega\\ &\ge \int_\Omega \chi \cdot v \, d\Omega - \int_\Omega j^0(u; v-u)\,d\Omega,\\ &\qquad\qquad \forall v \in V \cap L^\infty(\Omega;\mathbf{R}^N).\end{aligned} \tag{57}$$

By substituting $v = \widetilde{u}_k := (1-\varepsilon_k)u$ (with $\widetilde{u}_k$ as described in the truncation argument of Theorem 7) into the right-hand side of (57) one gets

$$\begin{aligned}&\liminf_{n\to\infty} \int_\Omega \chi_n \cdot u_n \, d\Omega\\ &\ge \liminf_{k\to\infty} \int_\Omega \chi \cdot \widetilde{u}_k \, d\Omega\\ &\quad - \limsup_{k\to\infty} \int_\Omega j^0(u; \widetilde{u}_k - u)\,d\Omega.\end{aligned} \tag{58}$$

Taking into account that $\widetilde{u}_k \to u$ a.e. in Ω,

$$j^0(u; \widetilde{u}_k - u) = \varepsilon_k j^0(u;-u) \le \varepsilon_k k\,|u| \le k\,|u|$$

and $|\chi \cdot u| \ge \chi \cdot \widetilde{u}_k = (1-\varepsilon_k)\chi \cdot u \ge -k\,|u|$, Fatou 's lemma and the dominated convergence can be used to deduce

$$\limsup_{k\to\infty} \int_\Omega j^0(u; \widetilde{u}_k - u)\,d\Omega \le 0,$$

and

$$\lim_{k\to\infty} \int_\Omega \chi \cdot \widetilde{u}_k \, d\Omega = \int_\Omega \chi \cdot u \, d\Omega.$$

Finally, combining the last two inequalities with (58) yields (55), as required.

Step 5. The next claim is that

$$\langle B-g, u\rangle_V + \int_\Omega \chi \cdot u \, d\Omega = 0. \tag{59}$$

Indeed, (50) implies

$$\langle B-g, \widetilde{u}_k\rangle_V + \int_\Omega \chi \cdot \widetilde{u}_k \, d\Omega = 0, \tag{60}$$

with $\{\widetilde{u}_k\}$ as in Step 3. Since $\chi \cdot u \in L^1(\Omega)$ and $-k|u| \leq \chi \cdot \widetilde{u}_k = (1-\varepsilon_k)\chi \cdot u \leq |\chi \cdot u|$, by the dominated convergence,

$$\int_\Omega \chi \cdot \widetilde{u}_k \, d\Omega \to \int_\Omega \chi \cdot u \, d\Omega.$$

It means that (59) has to hold by passing to the limit as $k \to \infty$ in (60).

Step 6. In this step it will be shown that the pseudomonotonicity of A and (47) imply (44). Indeed, (47) with $v \in V \cap L^\infty(\Omega;\mathbf{R}^N)$ and (49) allows to state that

$$\limsup_{n\to\infty} \langle Au_n, u_n - u\rangle_V \leq \langle B - g, v - u\rangle_V + \int_\Omega \chi \cdot v \, d\Omega - \liminf_{n\to\infty} \int_\Omega \chi_n \cdot u_n \, d\Omega.$$

Substituting $v = \widetilde{u}_k$ with $\widetilde{u}_k$ as in Step 3 and taking into account (55) one arrives at $\limsup_{n\to\infty} \langle Au_n, u_n - u\rangle_V \leq 0$ (by the application of the limit procedure as $k \to \infty$). Therefore the use of pseudomonotonicity of A is allowed and yields $\langle Au_n, u_n\rangle_V \to \langle Au, u\rangle_V$ and $Au_n \to B = Au$ weakly in V^* as $n \to \infty$. Finally, (47) implies (44), as claimed.

Step 7. In the final step of the proof it will be shown that (44) and (45) imply (46). For this purpose, choose $v \in V \cap L^\infty(\Omega;\mathbf{R}^N)$ arbitrarily. From (2) one has $\chi \cdot (v - u) \leq j^0(u;v-u) \leq \alpha(\|v\|_{L^\infty(\Omega;\mathbf{R}^N)})(1 + |u|^\sigma)$ with $\chi \cdot (v-u) \in L^1(\Omega)$ and $\alpha(\|v\|_{L^\infty(\Omega;\mathbf{R}^N)})(1 + |u|^\sigma) \in L^1(\Omega)$. Hence $j^0(u;v-u)$ is finite integrable and consequently, (46) follows immediately from (44).

Now consider the case $j^0(u;v-u) \in L^1(\Omega)$ with $v \notin V \cap L^\infty(\Omega;\mathbf{R}^N)$. According to Theorem 7 there exists a sequence $\widetilde{v}_k = (1-\varepsilon_k)v$ such that $\{\widetilde{v}_k\} \subset V \cap L^\infty(\Omega;\mathbf{R}^N)$ and $\widetilde{v}_k \to v$ strongly in V. Since

$$\langle Au - g, \widetilde{v}_k - u\rangle_V + \int_\Omega j^0(u;\widetilde{v}_k - u)\, d\Omega \geq 0,$$

so in order to establish (46) it remains to show that

$$\limsup_{k\to\infty} \int_\Omega j^0(u;\widetilde{v}_k - u)\, d\Omega \leq \int_\Omega j^0(u;v-u)\, d\Omega.$$

For this purpose let us observe that $\widetilde{v}_k - u = (1-\varepsilon_k)(v-u) + \varepsilon_k(-u)$ which combined with the convexity of $j^0(u;\cdot)$ yields the estimate

$$j^0(u;\widetilde{v}_k - u) \leq (1-\varepsilon_k)j^0(u;v-u) + \varepsilon_k j^0(u;-u) \leq |j^0(u;v-u)| + k|u|.$$

Thus the application of Fatou 's lemma gives the assertion. Finally, the proof of Theorem 6 is complete.

Remark 9 The analogous result to that of Theorem 6 can be formulated for the hemivariational inequality (46) in which $\int_\Omega(\cdot)\, d\Omega$ is replaced by the boundary integral $\int_\Gamma(\cdot)\, d\Gamma$, provided the imbedding $H^1(\Omega) \subset L^p(\Gamma)$ is compact $(1 < p < (2m-2)/(m-2)$, [12]).

Example 10 Let us consider a linear elastic body which in its undeformed state occupies an open, bounded, connected subset Ω of $\mathbf{R}^3$. Ω is referred to a fixed Cartesian coordinate system $0x_1x_2x_3$ and its boundary Γ is assumed to be Lipschitz regular; $n = (n_i)$ denotes the outward unit normal vector to Γ. We decompose Γ into two disjointed parts Γ_F and Γ_S such that $\Gamma = \overline{\Gamma}_F \cup \overline{\Gamma}_S$. As usual, the symbols u: $\Omega \to \mathbf{R}^3$ and $\sigma : \Omega \to \mathbf{S}^3$ are used to denote the displacement field and the stress tensor field, respectively. Here $\mathbf{S}^3$ stands for the space of all real-valued 3×3 symmetric matrices.

Consider the boundary value problem:

i) The equilibrium equations:

$$\sigma_{ij,j} + b_i = 0 \quad \text{in } \Omega. \tag{61}$$

ii) The displacement—strain relation:

$$\varepsilon_{ij}(u) = \frac{1}{2}(u_{i,j} + u_{j,i}) \quad \text{in } \Omega. \tag{62}$$

iii) *Hook's law*:

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl}(u) \quad \text{in } \Omega. \tag{63}$$

iv) The *surface traction* conditions

$$\sigma_{ij}n_j = F_i \quad \text{on } \Gamma_F. \tag{64}$$

v) The nonmonotone subdifferential boundary conditions

$$-S \in \partial j(u) \quad \text{on } \Gamma_S. \tag{65}$$

Here, $S = (S_i) = (\sigma_{ij}n_j)$ is the stress vector, and $\partial j(\cdot)$ is the generalized gradient of Clarke of a locally Lipschitz function j: $\mathbf{R}^3 \to \mathbf{R}$; the summation convention over repeated indices holds and the elasticity tensor $C = (C_{ijkl})$ is assumed to satisfy the classical conditions of ellipticity and symmetry [24].

Let $V = H^1(\Omega;\mathbf{R}^3)$. By making use of the standard technique (cf. [24]), Eqs. (61)-(65) lead to the problem

of finding $u \in V$ such as to satisfy the hemivariational inequality

$$\int_{\Omega} C_{ijkl}\varepsilon_{ij}(u)\varepsilon_{kl}(v-u)\,d\Omega - \int_{\Omega} b_i(v_i-u_i)\,d\Omega - \int_{\Gamma_F} F_i(v_i-u_i)\,d\Gamma + \int_{\Gamma_S} j^0(u;v-u)\,d\Gamma \geq 0, \quad \forall v \in V. \quad (66)$$

Define $A\colon V \to V^*$ by

$$\langle Au, v\rangle_V = \int_{\Omega} C_{ijkl}\varepsilon_{ij}(u)\varepsilon_{kl}(v)\,d\Omega, \quad u, v \in V,$$

and let $V_0 := \mathcal{R} = \{\, \rho \in V : \varepsilon_{ij}(\rho) = 0,\ i, j = 1, 2, 3 \,\}$ denote the space of all rigid-body displacements. Then (1) holds (for details see [24, p. 121]). Accordingly, if (2) (with $\sigma < 4$) and (3) are fulfilled and, moreover, the compatibility condition

$$\int_{\Omega} b_i\rho_i\,d\Omega + \int_{\Gamma_F} F_i\rho_i\,d\Gamma < \int_{\Gamma_S} j^{\infty}(\rho)\,d\Gamma$$

is valid for any $\rho \in \mathcal{R} \setminus \{\, 0 \,\}$, then the hypotheses of the theorem mentioned in Remark 9 are satisfied. Consequently, the existence of solutions to the hemivariational inequality (66) is ensured.

References

1. Aubin JP, Ekeland I (1984) Applied nonlinear analysis. Wiley, New York
2. Baiocchi C, Buttazzo G, Gastaldi F, Tomarelli F (1988) General existence theorems for unilateral problems in continuum mechanics. Arch Rational Mechanics Anal 100:149–180
3. Brèzis H (1968) Èquations et inèquations non-lin èaires dans les espaces vèctoriels en dualitè. Ann Inst Fourier Grenoble 18:115–176
4. Brèzis H, Nirenberg L (1978) Characterizations of the ranges of some nonlinear operators and applications to boundary value problems. Ann Scuola Norm Sup Pisa Cl Sci IV V(2):225–326
5. Browder FE, Hess P (1972) Nonlinear mappings of monotone type in Banach spaces. J Funct Anal 11:251–294
6. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
7. Duvaut G, Lions JL (1972) Les inèquations en mècanique et en physique. Dunod, Paris
8. Ekeland I, Temam R (1976) Convex analysis and variational problems. North-Holland, Amsterdam
9. Fichera G (1972) Boundary value problems in elasticity with unilateral constraints. Handbuch der Physik, vol VIa/2. Springer, Berlin, pp 347–389
10. Goeleven D (1996) Noncoercive variational problems and related topics. Res Notes Math, vol 357. Longman
11. Hedberg LI (1978) Two approximation problems in function space. Ark Mat 16:51–81
12. Kufner A, John O, Fučik S (1977) Function spaces. Academia, Prague
13. Motreanu D, Naniewicz Z (1996) Discontinuous semilinear problems in vector-valued function spaces. Differential Integral Eq 9:581–598
14. Motreanu D, Panagiotopoulos PD (1995) Nonconvex energy functions, related eigenvalue hemivariational inequalities on the sphere and applications. J Global Optim 6:163–177
15. Motreanu D, Panagiotopoulos PD (1996) On the eigenvalue problem for hemivariational inequalities: Existence and multiplicity of solutions. J Math Anal Appl 197:75–89
16. Naniewicz Z (1994) Hemivariational inequalities with functions fulfilling directional growth condition. Appl Anal 55:259–285
17. Naniewicz Z (1994) Hemivariational inequality approach to constrained problems for star-shaped admissible sets. J Optim Th Appl 83:97–112
18. Naniewicz Z (1995) Hemivariational inequalities with functionals which are not locally Lipschitz. Nonlinear Anal 25:1307–1320
19. Naniewicz Z (1995) On variational aspects of some nonconvex nonsmooth global optimization problem. J Global Optim 6:383–400
20. Naniewicz Z (1997) Hemivariational inequalities as necessary conditions for optimality for a class of nonsmooth nonconvex functionals. Nonlinear World 4:117–133
21. Naniewicz Z, Panagiotopoulos PD (1995) Mathematical theory of hemivariational inequalities and applications. M. Dekker, New York
22. Panagiotopoulos PD (1981) Nonconvex superpotentials in the sense of F.H. Clarke and applications. Mechanics Res Comm 8:335–340
23. Panagiotopoulos PD (1983) Noncoercive energy function, hemivariational inequalities and substationarity principles. Acta Mechanics 48:160–183
24. Panagiotopoulos PD (1985) Inequality problems in mechanics and applications. Convex and nonconvex energy functions. Birkhäuser, Basel
25. Panagiotopoulos PD (1991) Coercive and semicoercive hemivariational inequalities. Nonlinear Anal 16:209–231
26. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Berlin
27. Rauch J (1977) Discontinuous semilinear differential equations and multiple valued maps. Proc Amer Math Soc 64:277–282
28. Webb JRL (1980) Boundary value problems for strongly nonlinear elliptic equations. J London Math Soc 21:123–132

Heuristic and Metaheuristic Algorithms for the Traveling Salesman Problem

YANNIS MARINAKIS
Department of Production Engineering and Management, Decision Support Systems Laboratory, Technical University of Crete, Chania, Greece

MSC2000: 90C59

Article Outline

Introduction
Heuristics for the Traveling Salesman Problem
Metaheuristics for the Traveling Salesman Problem
References

Introduction

The **Traveling Salesman Problem** (**TSP**) is one of the most representative problems in combinatorial optimization. If we consider a salesman who has to visit n cities [46], the Traveling Salesman Problem asks for the shortest tour through all the cities such that no city is visited twice and the salesman returns at the end of the tour back to the starting city. Mathematically, the problem may be stated as follows: Let $G = (V, E)$ be a graph, where V is a set of n nodes and E is a set of arcs, let $C = [c_{ij}]$ be a cost matrix associated with E, where c_{ij} represents the cost of going from city i to city j, $(i, j = 1, \ldots, n)$, the problem is to find a permutation $(i_1, i_2, i_3, \ldots, i_n)$ of the integers from 1 through n that minimizes the quantity $c_{i_1 i_2} + c_{i_2 i_3} + \ldots + c_{i_n i_1}$.

We speak of a *Symmetric TSP*, if for all pairs (i, j), the distance c_{ij} is equal to the distance c_{ji}. Otherwise, we speak of the *Asymmetric TSP* [7]. If the triangle inequality holds ($c_{ij} \leq c_{ii_1} + c_{i_1 j}$, $\forall i, j, i_1$), the problem is said to be metric. If the cities can be represented as points in the plain such that c_{ij} is the Euclidean distance between point i and point j, then the corresponding TSP is called the Euclidean TSP. Euclidean TSP obeys in particular the triangle inequality $c_{ij} \leq c_{ii_1} + c_{i_1 j}$ for all i, j, i_1.

An integer programming formulation of the Traveling Salesman Problem is defined in a complete graph $G = (V, E)$ of n nodes, with node set $V = \{1, \ldots, n\}$, arc set $E = \{(i, j) | i, j = 1, \ldots, n\}$, and nonnegative costs c_{ij} associated with the arcs [8]:

$$c^* = \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{1}$$

s.t.

$$\sum_{j \in V} x_{ij} = 1, \quad i \in V \tag{2}$$

$$\sum_{i \in V} x_{ij} = 1, \quad j \in V \tag{3}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad \forall S \subset V, S \neq \emptyset \tag{4}$$

$$x_{ij} \in \{0, 1\}, \quad \text{for all } i, j \in V, \tag{5}$$

where $x_{ij} = 1$ if arc (i,j) is in the solution and 0 otherwise. In this formulation, the objective function clearly describes the cost of the optimal tour. Constraints (2) and (3) are degree constraints: they specify that every node is entered exactly once and left exactly once. Constraints (4) are subtour elimination constraints. These constraints prohibit the formation of subtours, i. e. tours on subsets of less than V nodes. If there was such a subtour on a subset S of nodes, this subtour would contain $|S|$ edges and as many nodes. Constraints (4) would then be violated for this subset since the left-hand side of (4) would be equal to $|S|$ while the right-hand side would be equal to $|S| - 1$. Because of degree constraints, subtours over one node (and hence, over $n - 1$ nodes) cannot occur. For more formulations of the problem see [34,60].

The Traveling Salesman Problem (TSP) is one of the most famous hard combinatorial optimization problems. It has been proven that TSP is a member of the set of NP-complete problems. This is a class of difficult problems whose time complexity is probably exponential. The members of the class are related so that if a polynomial time algorithm was found for one problem, polynomial time algorithms would exist for all of them [41]. However, it is commonly believed that no such polynomial algorithm exists. Therefore, any attempt to construct a general algorithm for finding optimal solutions for the TSP in polynomial time must (probably) fail. That is, for any such algorithm it is possible to construct problem instances for which the execution time grows at least exponentially with the size of the input. Note, however, that time complexity here

refers to the worst case behavior of the algorithm. It can not be excluded that there exist algorithms whose average running time is polynomial. The existence of such algorithms is still an open question. Since 1950s many algorithms have been proposed, developed and tested for the solution of the problem. Algorithms for solving the TSP may be divided into two categories, *exact algorithms* and *heuristic–metaheuristic algorithms*.

Heuristics for the Traveling Salesman Problem

There is a great need for powerful heuristics that find good suboptimal solutions in reasonable amounts of computing time. These algorithms are usually very simple and have short running times. There is a huge number of papers dealing with finding near optimal solutions for the TSP. Our aim is to present the most interesting and efficient algorithms and the most important ones for facing practical problems. In the 1960s, 1970s and 1980s the attempts to solve the Traveling Salesman Problem focused on **tour construction methods** and **tour improvement methods**. In the last fifteen years, metaheuristics, such as **simulated annealing, tabu search, genetic algorithms** and **neural networks**, were introduced. These algorithms have the ability to find their way out of local optima. Heuristics and metaheuristics constitute an increasingly essential component of solution approaches intended to tackle difficult problems, in general, and global and combinatorial problems in particular.

When a heuristic is designed, the question which arises is about the quality of the produced solution. There are three different ways that one may try to answer this question.

1. **Empirical.** The heuristic is applied to a number of test problem instances and the solutions are compared to the optimal values, if there are known, or to lower bounds on these values [33,35].
2. **Worst Case Analysis.** The idea is to derive bounds on the worst possible deviation from the optimum that the heuristic could produce and to devise bad problem instances for which the heuristic actually achieves this deviation [42].
3. **Probabilistic Analysis.** In the probabilistic analysis it is assumed that problem instances are drawn from certain simple probability distributions, and it is, then, proven mathematically that particular solution methods are highly likely to yield near-optimal solutions when the number of cities is large [43].

Tour Construction methods build up a tour step by step. Such heuristics build a solution (tour) from scratch by a growth process (usually a greedy one) that terminates as soon as a feasible solution has been constructed. The problem with construction heuristics is that although they are usually fast, they do not, in general, produce very good solutions. One of the simplest tour construction methods is the **nearest neighborhood** in which, a salesman starts from an arbitrary city and goes to its nearest neighbor. Then, he proceeds from there in the same manner. He visits the nearest unvisited city, until all cities are visited, and then returns to the starting city [65,68].

An extension of the nearest neighborhood method is the **double-side nearest neighborhood method** where the current path can be extended from both of its endnodes. Some authors use the name **Greedy** for Nearest Neighborhood, but it is more appropriately reserved for the special case of the greedy algorithm of matroid theory [39]. Bentley [11] proposed two very fast and efficient algorithms, the **K-d Trees** and the **Lazily Update Priority Queues**. In his paper, it was the first time that somebody suggested the use of data structures for the solution of the TSP. A priority queue contains items with associated values (the priorities) and support operations that [40]:

- remove the highest priority item from the queue and deliver it to the user,
- insert an item,
- delete an item, and
- modify the priority of an item.

The **insertion procedures** [68] take a subtour of V nodes and attempt to determine which node (not already in the subtour) should join the subtour next (the *selection step*) and then determine where in the subtour it should be inserted (the *insertion step*). The most known of these algorithms is the **nearest insertion algorithm**. Similar to the nearest insertion procedure are the **cheapest insertion** [65], the **arbitrary insertion** [12], the **farthest insertion** [65], the **quick insertion** [12], and the **convex hull insertion** [12] algorithms.

There is a number of heuristic algorithms that are designed for speed rather for quality of the tour they construct [40]. The three most known heuristic algo-

rithms of this category are the **Strip algorithm**, proposed by Beardwood et al. [10], the **Spacefilling Curve** proposed by Platzmann and Bartholdi [58] and the **Fast Recursive Partitioning** heuristic proposed by Bentley [11]. The **saving algorithms** are exchange procedures. The most known of them is the **Clarke-Wright algorithm** [17]. Christofides [12,65] suggested a procedure for solving the TSP based on **spanning trees**. He proposed a method of transforming spanning trees to Eulerian graphs.

The **improvement methods** or **local search methods** start with a tour and try to find all tours that are "neighboring" to it and are shorter than the initial tour and, then, to replace it. The tour improvements methods can be divided into three categories according to the type of the neighborhood that they use [64]. Initially, the **constructive neighborhood methods**, which successively add new components to create a new solution, while keeping some components of the current solution fixed. Some of these methods will be presented in the next section where the most known metaheuristics are presented. Secondly, the **transition neighborhood methods**, which are the classic local search algorithms (classic tour improvement methods) and which iteratively move from one solution to another based on the definition of a neighborhood structure. Finally, the **population based neighborhood methods**, which generalize the two previous categories by considering neighborhoods of more than one solution.

The most known of the local search algorithms is the **2-opt heuristic**, in which two edges are deleted and the open ends are connected in a different way in order to obtain a new tour [48]. Note that there is only one way to reconnect the paths. The **3-opt heuristic** is quite similar with the 2-opt but it introduces more flexibility in modifying the current tour, because it uses a larger neighborhood. The tour breaks into three parts instead of only two [48]. In the general case, δ **edges** in a feasible tour are exchanged for δ edges not in that solution as long as the result remains a tour and the length of that tour is less than the length of the previous tour. **Lin-Kernighan method (LK)** was developed by Lin and Kernighan [37,49,54] and for many years was considered to be the best heuristic for the TSP. The **Or-opt procedure**, well known as **node exchange heuristic**, was first introduced by Or [56]. It removes a sequence of up-to-three adjacent nodes and inserts it at another location within the same route. Or-opt can be considered as a special case of 3-opt (three arcs exchanges) where three arcs are removed and substituted by three other arcs. The **GENI algorithm** was presented by Gendreau, Hertz and Laporte [22]. GENI is a hybrid of tour construction and local optimization.

Metaheuristics for the Traveling Salesman Problem

The last fifteen years an incremental amount of metaheuristic algorithms have been proposed. Simulated annealing, genetic algorithms, neural networks, tabu search, ant algorithms, together with a number of hybrid techniques are the main categories of the metaheuristic procedures. These algorithms have the ability to find their way out of local optima. A number of metaheuristic algorithms have been proposed for the solution of the Traveling Salesman Problem. The most important algorithms published for each metaheuristic algorithm are given in the following:

- **Simulated Annealing** (**SA**) belongs [1,2,45,64] to a class of local search algorithms that are known as *threshold accepting algorithms*. These algorithms play a special role within local search for two reasons. First, they appear to be quite successful when applied to a broad range of practical problems. Second, some threshold accepting algorithms such as SA have a stochastic component, which facilitates a theoretical analysis of their asymptotic convergence. Simulated Annealing [3] is a stochastic algorithm that allows random uphill jumps in a controlled fashion in order to provide possible escapes from poor local optima. Gradually the probability allowing the objective function value to increase is lowered until no more transformations are possible. Simulated Annealing owes its name to an analogy with the annealing process in condensed matter physics, where a solid is heated to a maximum temperature at which all particles of the solid randomly arrange themselves in the liquid phase, followed by cooling through careful and slow reduction of the temperature until the liquid is frozen with the particles arranged in a highly structured lattice and minimal system energy. This ground state is reachable only if the maximum temperature is sufficiently high and the cooling sufficiently slow.

Otherwise a meta-stable state is reached. The meta-stable state is also reached with a process known as quenching, in which the temperature is instantaneously lowered. Its predecessor is the so-called Metropolis filter. Simulated Annealing algorithms for the TSP are presented in [15,55,65].

- **Tabu search (TS)** was introduced by Glover [24,25] as a general iterative metaheuristic for solving combinatorial optimization problems. Computational experience has shown that TS is a well established approximation technique, which can compete with almost all known techniques and which, by its flexibility, can beat many classic procedures. It is a form of local neighbor search. Each solution S has an associated set of neighbors $N(S)$. A solution $S' \in N(S)$ can be reached from S by an operation called a *move*. TS can be viewed as an iterative technique which explores a set of problem solutions, by repeatedly making moves from one solution S to another solution S' located in the neighborhood $N(S)$ of S [31]. TS moves from a solution to its best admissible neighbor, even if this causes the objective function to deteriorate. To avoid cycling, solutions that have been recently explored are declared *forbidden or tabu* for a number of iterations. The tabu status of a solution is overridden when certain criteria (*aspiration criteria*) are satisfied. Sometimes, *intensification* and *diversification* strategies are used to improve the search. In the first case, the search is accentuated in the promising regions of the feasible domain. In the second case, an attempt is made to consider solutions in a broad area of the search space. The first Tabu Search algorithm implemented for the TSP appears to be the one described by Glover [23,29]. Limited results for this implementation and variants on it were reported by Glover [26]. Other Tabu Search algorithms for the TSP are presented in [74].
- **Genetic Algorithms (GAs)** are search procedures based on the mechanics of natural selection and natural genetics. The first GA was developed by John H. Holland in the 1960s to allow computers to evolve solutions to difficult search and combinatorial problems, such as function optimization and machine learning [38]. Genetic algorithms offer a particularly attractive approach for problems like traveling salesman problem since they are generally quite effective for rapid global search of large, non-linear and poorly understood spaces. Moreover, genetic algorithms are very effective in solving large-scale problems. Genetic algorithms mimic the evolution process in nature. GAs are based on an imitation of the biological process in which new and better populations among different species are developed during evolution. Thus, unlike most standard heuristics, GAs use information about a population of solutions, called individuals, when they search for better solutions. A GA is a stochastic iterative procedure that maintains the population size constant in each iteration, called a generation. Their basic operation is the mating of two solutions in order to form a new solution. To form a new population, a binary operator called crossover, and a unary operator, called mutation, are applied [61,62]. Crossover takes two individuals, called parents, and produces two new individuals, called offsprings, by swapping parts of the parents. Genetic algorithms for the TSP are presented in [9,51,59,64,67].
- **Greedy Randomized Adaptive Search Procedure - GRASP** [66] is an iterative two phase search method which has gained considerable popularity in combinatorial optimization. Each iteration consists of two phases, a construction phase and a local search procedure. In the construction phase, a randomized greedy function is used to build up an initial solution. This randomized technique provides a feasible solution within each iteration. This solution is then exposed for improvement attempts in the local search phase. The final result is simply the best solution found over all iterations. Greedy Randomized Adaptive Search Procedure algorithms for the TSP are presented in [50,51].
- The use of **Artificial Neural Networks** to find good solutions to combinatorial optimization problems has recently caught some attention. A neural network consists of a network [57] of elementary nodes (neurons) that are linked through weighted connections. The nodes represent computational units, which are capable of performing a simple computation, consisting of a summation of the weighted inputs, followed by the addition of a constant called the threshold or bias, and the application of a non-linear response (activation) function. The result of the computation of a unit constitutes its output. This output is used as an input for the nodes to which

it is linked through an outgoing connection. The overall task of the network is to achieve a certain network configuration, for instance a required input–output relation, by means of the collective computation of the nodes. This process is often called *self–organization*. Neural Networks algorithms for the TSP are presented in [4,6,53,69].

- The **Ant Colony Optimization (ACO)** metaheuristic is a relatively new technique for solving combinatorial optimization problems (COPs). Based strongly on the Ant System (AS) metaheuristic developed by Dorigo, Maniezzo and Colorni [19], ant colony optimization is derived from the foraging behaviour of real ants in nature. The main idea of ACO is to model the problem as the search for a minimum cost path in a graph. Artificial ants walk through this graph, looking for good paths. Each ant has a rather simple behavior so that it will typically only find rather poor-quality paths on its own. Better paths are found as the emergent result of the global cooperation among ants in the colony. An ACO algorithm consists of a number of cycles (iterations) of solution construction. During each iteration a number of ants (which is a parameter) construct complete solutions using heuristic information and the collected experiences of previous groups of ants. These collected experiences are represented by a digital analogue of trail pheromone which is deposited on the constituent elements of a solution. Small quantities are deposited during the construction phase while larger amounts are deposited at the end of each iteration in proportion to solution quality. Pheromone can be deposited on the components and/or the connections used in a solution depending on the problem. Ant Colony Optimization algorithms for the TSP are presented in [16,18,19,70].
- One way to invest extra computation time is to exploit the fact that many local improvement heuristics have random components, even if in their initial tour construction phase. Thus, if one runs the heuristic multiple times he will get different results and can take the best. The **Iterated Lin Kernighan algorithm** (**ILK**) [54] has been proposed by Johnson [39] and it is considered to be one of the best algorithms for obtaining a first local minimum. To improve this local minimum, the algorithm examines other local minimum tours 'near' the current local minimum. To generate these tours, ILK first applies a random and unbiased nonsequential 4-opt exchange to the current local minimum and then optimizes this 4-opt neighbor using the LK algorithm. If the tour obtained by this process is better than the current local minimum then ILK makes this tour the current local minimum and continues from there using the same neighbor generation process. Otherwise, the current local minimum remains as it is and further random 4-opt moves are tried. The algorithm stops when a stopping criterion based either on the number of iterations or the computational time is satisfied. Two other approaches are the **Iterated 3-opt** and the **Chained Lin-Kernighan** [5], where random kicks are generated from the solution and from these new points the exploration for a better solution is continued [40].
- **Ejection Chain Method** provides a wide variety of reference structures, which have the ability to generate moves not available to neighborhood search approaches traditionally applied to TSP [63,64]. Ejection Chains are variable depth methods that generate a sequence of interrelated simple moves to create a more complex compound move. An ejection consists of a succession of operations performed on a given set of elements, where the m_t operation changes the state of one or more elements which are said to be ejected in the m_t+1 operations. Of course, there is a possibility to appear changes in the state of other elements, which will lead to other ejections, until no more operations can be made [27]. Other Ejection Chain Algorithms are presented in [20,21].
- **Scatter Search** is an evolutionary strategy originally proposed by Glover [28,30]. Scatter Search operates on a set of reference solutions to generate a new set of solutions by weighted linear combinations of structured subset of solutions. The reference set is required to be made up of high quality and diverse solutions and the goal is to produce weighted centers of selected subregions that project these centers into regions of the solution space that are to be explored by auxiliary heuristic procedures.
- **Path Relinking** [28,30], combines solutions by generating paths between them using local search neighborhoods, and selecting new solutions encountered along these paths.

- **Guided Local Search** (**GLS**), originally proposed by Voudouris and Chang [71,72], is a general optimization technique suitable for a wide range of combinatorial optimization problems. The main focus is on the exploitation of problem and search–related information to effectively guide local search heuristics in the vast search spaces of NP-hard optimization problems. This is achieved by augmenting the objective function of the problem to be minimized with a set of penalty terms which are dynamically manipulated during the search process to steer the heuristic to be guided. GLS augments the cost function of the problem to include a set of penalty terms and passes this, instead of the original one, for minimization by the local search procedure. Local search is confined by the penalty terms and focuses attention on promising regions of the search space. Iterative calls are made to local search. Each time local search gets caught in a local minimum, the penalties are modified and local search is called again to minimize the modification cost function. Guided Local Search algorithms for the TSP are presented in [71,72].
- **Noising Method** was proposed by Charon and Hudry [13] and is a metaheuristic where if it is wanted to minimize the function f^1, this method do not take the true values of f^1 into account but it considers that they are perturbed in some way by noises in order to get a noised function ${f^1}_{\text{noised}}$. During the run of the algorithm, the range of the perturbing noises decreases (typically to zero), so that, at the end, there is no significant noise and the optimization of ${f^1}_{\text{noised}}$ leads to the same solution as the one provided by a descent algorithm applied to f^1 with the same initial solution. This algorithm was applied to the Traveling Salesman Problem by Charon and Hudry [14].
- **Particle Swarm Optimization (PSO)** is a population-based swarm intelligence algorithm. It was originally proposed by Kennedy and Eberhart as a simulation of the social behavior of social organisms such as bird flocking and fish schooling [44]. PSO uses the physical movements of the individuals in the swarm and has a flexible and well-balanced mechanism to enhance and adapt to the global and local exploration abilities. PSO algorithms for the solution of the Traveling Salesman Problem are presented in [32,47,73].
- **Variable Neighborhood Search (VNS)** is a metaheuristic for solving combinatorial optimization problems whose basic idea is systematic change of neighborhood within a local search [36]. Variable Neighborhood Search algorithms for the TSP are presented in [52].

References

1. Aarts E, Korst J (1989) Simulated Annealing and Boltzmann Machines - A stochastic Approach to Combinatorial Optimization and Neural Computing. John Wiley and Sons, Chichester
2. Aarts E, Ten Eikelder HMM (2002) Simulated Annealing. In: Pardalos PM, Resende MGC (eds) Handbook of Applied Optimization. Oxford University Press, Oxford, pp 209–221
3. Aarts E, Korst J, Van Laarhoven P (1997) Simulated Annealing. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. John Wiley and Sons, Chichester, pp 91–120
4. Ansari N, Hou E (1997) Computational Intelligence for Optimization, 1st edn. Kluwer, Boston
5. Applegate D, Cook W, Rohe A (2003) Chained Lin-Kernighan for Large Traveling Salesman Problems. Informs J Comput 15:82–92
6. Bai Y, Zhang W, Jin Z (2006) An New Self-Organizing Maps Strategy for Solving the Traveling Salesman Problem. Chaos Solitons Fractals 28(4):1082–1089
7. Balas E, Fischetti M (2002) Polyhedral Theory for the Assymetric Traveling Salesman Problem. In: Gutin G, Punnen A (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 117–168
8. Balas E, Toth P (1985) Branch and Bound Methods. In: Lawer EL, Lenstra JK, Rinnoy Kan AHG, Shmoys DB (eds) The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley and Sons, Chichester, pp 361–401
9. Baralia R, Hildago JI, Perego R (2001) A Hybrid Heuristic for the Traveling Salesman Problem. IEEE Trans Evol Comput 5(6):1–41
10. Beardwood J, Halton JH, Hammersley JM (1959) The Shortest Path Through Many Points. Proc Cambridge Philos Soc 55:299–327
11. Bentley JL (1992) Fast Algorithms for Geometric Traveling Salesman Problems. ORSA J Comput 4:387–411
12. Bodin L, Golden B, Assad A, Ball M (1983) The State of the Art in the Routing and Scheduling of Vehicles and Crews. Comput Oper Res 10:63–212
13. Charon I, Hudry O (1993) The Noising Method: A New Combinatorial Optimization Method. Oper Res Lett 14:133–137
14. Charon I, Hudry O (2000) Applications of the Noising Method to the Traveling Salesman Problem. Eur J Oper Res 125:266–277

15. Chen Y, and Zhang P (2006) Optimized Annealing of Traveling Salesman Problem from the nth-Nearest-Neighbor Distribution. Physica A: Stat Theor Phys 371(2):627–632
16. Chu SC, Roddick JF, Pan JS (2004) Ant Colony System with Communication Strategies. Inf Sci 167(1–4):63–76
17. Clarke G, and Wright J (1964) Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Oper Res 12:568–581
18. Dorigo M, Gambardella LM (1997) Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Trans Evol Comput 1(1):53–66
19. Dorigo M, Stutzle T (2004) Ant Colony Optimization, A Bradford Book. The MIT Press Cambridge, Massachusetts, London
20. Gamboa D, Rego C, Glover F (2005) Data Structures and Ejection Chains for Solving Large-Scale Traveling Salesman Problems. Eur J Oper Res 160(1):154–171
21. Gamboa D, Rego C, Glover F (2006) Implementation Analysis of Efficient Heuristic Algorithms for the Traveling Salesman Problem. Comput Oper Res 33(4):1154–1172
22. Gendreau M, Hertz A, Laporte G (1992) New Insertion and Postoptimization Procedures for the Traveling Salesman Problem. Oper Res 40:1086–1094
23. Glover F (1986) Future Paths for Integer Programming and Links to Artificial Intelligence. Comput Oper Res 13:533–549
24. Glover F (1989) Tabu Search I. ORSA J Comput 1(3):190–206
25. Glover F (1990) Tabu Search II. ORSA J Comput 2(1):4–32
26. Glover F (1990) Tabu search: A tutorial. Center for Applied Artificial Intelligence, University of Colorado, pp 1–47
27. Glover F (1992) Ejection Chains, Reference Structures and Alternating Path Algorithms for Traveling Salesman Problem. Discrete Appl Math 65:223–253
28. Glover F (1997) A Template for Scatter Search and Path Relinking. Lecture Notes in Computer Science, vol 1363. pp 13–54
29. Glover F, and Laguna M (2002) Tabu Search. In: Pardalos PM, Resende MGC (eds) Handbook of Applied Optimization. Oxford University Press, Oxford, pp 194–209
30. Glover F, Laguna M, Marti R (2003) Scatter Search and Path Relinking: Advances and Applications. In: Glover F, Kochenberger GA (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 1–36
31. Glover F, Laguna M, Taillard E, de Werra D (eds) (1993) Tabu Search. J.C. Baltzer AG, Science Publishers, Basel, Switzerland
32. Goldbarg EFG, Souza GR, Goldbarg MC (2006) Particle Swarm Optimization for the Traveling Salesman Problem. EVOCOP 2006 LNCS 3906:99–110
33. Golden BL, Stewart WR (1985) Empirical Analysis of Heuristics. In: Lawer EL, Lenstra JK, Rinnoy Kan AHG, Shmoys DB (eds) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley and Sons, Chichester, pp 207–249
34. Gutin G, Punnen A (eds) (2002) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht
35. Haimovich M, Rinnoy Kan AHG, Stougie L (1988) Analysis of Heuristics for Vehicle Routing Problems. In: Golden BL, Assad AA (eds) Vehicle Routing: Methods and Studies. Elsevier Science Publishers, North Holland, pp 47–61
36. Hansen P, Mladenovic N (2001) Variable Neighborhood Search: Principles and Applications. Eur J Oper Res 130:449–467
37. Helsgaun K (2000) An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. Eur J Oper Res 126:106–130
38. Holland J H. (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor
39. Johnson DS, McGeoch LA (1997) The Traveling Salesman Problem: A Case Study. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. John Wiley and Sons, Chichester, pp 215–310
40. Johnson DS, McGeoch LA (2002) Experimental Analysis of the STSP. In: Gutin G, Punnen A (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 369–444
41. Johnson DS, Papadimitriou CH (1985) Computational Complexity. In: Lawer EL, Lenstra JK, Rinnoy Kan AHD, Shmoys DB (eds) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley and Sons, Chichester, pp 37–85
42. Johnson DS, Papadimitriou CH (1985) Performance Guarantees for Heuristics. In: Lawer EL, Lenstra JK, Rinnoy Kan AHD, Shmoys DB (eds) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley and Sons, Chichester, pp 145–181
43. Karp RM, Steele JM (1985) Probabilistic Analysis of Heuristics. In: Lawer EL, Lenstra JK, Rinnoy Kan AHD, Shmoys DB (eds) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley and Sons, Chichester, pp 181–206
44. Kennedy J, Eberhart R (1995) Particle Swarm Optimization. Proc. 1995 IEEE Int Conf Neural Netw 4:1942–1948
45. Kirkpatrick S, Gelatt CD, Vecchi MP (1982) Optimization by Simulated Annealing. Science 220:671–680
46. Lawer EL, Lenstra JK, Rinnoy Kan AHG, Shmoys DB (1985) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley and Sons, New York
47. Li X, Tian P, Hua J, Zhong N (2006) A Hybrid Discrete Particle Swarm Optimization for the Traveling Salesman Problem. SEAL 2006, LNCS 4247:181–188
48. Lin S (1965) Computer Solutions of the Traveling Salesman Problem. Bell Syst Tech J 44:2245–2269
49. Lin S, Kernighan BW (1973) An Effective Heuristic Algorithm for the Traveling Salesman Problem. Oper Res 21:498–516
50. Marinakis Y, Migdalas A, Pardalos PM (2005) Expanding Neighborhood GRASP for the Traveling Salesman Problem. Comput Optim Appl 32:231–257

51. Marinakis Y, Migdalas A, Pardalos PM (2005) A Hybrid Genetic-GRASP algortihm Using Langrangean Relaxation for the Traveling Salesman Problem. J Combinat Optim 10:311–326
52. Mladenovic N, Hansen P (1997) Variable Neighborhood Search. Comput Oper Res 24:1097–1100
53. Modares A, Somhom S, Enkawa T (1999) A Self-Organizing Neural Network Approach for Multiple Traveling Salesman and Vehicle Routing Problems. Int Trans Oper Res 6(6):591–606
54. Neto DM (1999) Efficient Cluster Compensation for Lin - Kernighan Heuristics. Ph.D. Thesis, Computer Science University of Toronto, Canada
55. Ninio M, Schneider JJ (2005) Weight Annealing. Physica A: Stat Theor Phys 349(3–4):649–666
56. Or I (1976) Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking. Ph.D. Thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston IL
57. Sodererg B, Peterson C (1997) Artificial Neural Networks. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. John Wiley and Sons, Chichester, pp 173–214
58. Platzmann LK, Bartholdi JJ (1989) Spacefilling Curves and the Planar Traveling Salesman Problem. J Assoc Comput Mach 36:719–735
59. Potvin J Y. (1996) Genetic Algorithms for the Traveling Salesman Problem. Metaheuristics Combinatorial Optim Ann Oper Res 63:339–370
60. Punnen AP (2002) The Traveling Salesman Problem: Applications, Formulations and Variations. In: Gutin G, Punnen A (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 1–28
61. Reeves CR (1995) Genetic Algorithms. In: Reeves CR (ed) Modern Heuristic Techniques for Combinatorial Problems. McGraw - Hill, London, pp 151–196
62. Reeves CR (2003) Genetic Algorithms. In: Glover F, Kochenberger GA (eds) Handbooks of Metaheuristics. Kluwer, Dordrecht, pp 55–82
63. Rego C (1998) Relaxed Tours and Path Ejections for the Traveling Salesman Problem. Eur J Oper Res 106:522–538
64. Rego C, Glover F (2002) Local Search and Metaheuristics. In: Gutin G, Punnen A (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 309–367
65. Reinelt G (1994) The Traveling Salesman, Computational Solutions for TSP Applications. Springer, Berlin
66. Resende MGC, Ribeiro CC (2003) Greedy Randomized Adaptive Search Procedures. In: Glover F, Kochenberger GA (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 219–249
67. Ronald S (1995) Routing and Scheduling Problems. In: Chambers L (ed) Practical Handbook of Genetic Algorithms. CRC Press, New York, pp 367–430
68. Rosenkratz DJ, Stearns RE, Lewis PM (1977) An Analysis of Several Heuristics for the Travelling Salesman Problem. SIAM J Comput 6:563–581
69. Siqueira PH, Teresinha M, Steiner A, Scheer S (2007) A New Approach to Solve the Traveling Salesman Problem. Neurocomputing 70(4–6):1013–1021
70. Taillard ED (2002) Ant Systems. In: Pardalos PM, Resende MGC (eds) Handbook of Applied Optimization. Oxford University Press, Oxford, pp 130–138
71. Voudouris C, Tsang E (1999) Guided Local Search and its Application to the Travelling Salesman Problem. Eur J Oper Res 113:469–499
72. Voudouris C, Tsang E (2003) Guided Local Search. In: Glover F, Kochenberger GA (eds) Handbooks of Metaheuristics. Kluwer, Dordrecht, pp 185–218
73. Wang Y, Feng XY, Huang YX, Pu DB, Zhou WG, Liang YC, Zhou CG (2007) A Novel Quantum Swarm Evolutionary Algorithm and its Applications. Neurocomputing 70 (4–6):633–640
74. Zachariasen M, Dam M (1996) Tabu Search on the Geometric Traveling Salesman Problem. In: Osman IH, Kelly JP (eds) Meta-heuristics: Theory and Applications. Kluwer, Boston, pp 571–587

Heuristic Search

ALEXANDER REINEFELD
ZIB Berlin, Berlin, Germany

MSC2000: 68T20, 90B40, 90C47

Article Outline

Keywords and Phrases
Introduction
Depth-First Search
Best-First Search
Applications
See also
References

Keywords and Phrases

Optimization; Heuristic search

Introduction

Heuristic search [7,9] is a common technique for finding a solution in a decision tree or graph containing one

or more solutions. Many applications in operations research and artificial intelligence rely on heuristic search as their primary solution method.

Heuristic search techniques can be classified into two broad categories: depth-first search (DFS) and best-first search (BFS). As a consequence of its better information base, BFS usually examines fewer nodes but occupies more storage space for maintaining the already explored nodes.

Depth-First Search

DFS expands an initial state by generating its immediate successors. At each subsequent step, one of the most recently generated successors is selected and expanded. At terminal states, or when it can be determined that the current state does not lead to a solution, the search backtracks, that is, the node expansion proceeds with the next most recently generated state. Practical implementations use a stack data structure for maintaining the states (nodes) on the path to the currently explored state. The space complexity of the stack, $O(d)$, increases only linearly with the search depth d.

Backtracking is the most rudimentary variant of DFS. It terminates as soon as any solution has been found; hence, there is no guarantee for finding an optimal (least-cost) solution. Moreover, backtracking might not terminate in graphs containing cycles or when the search depth is unbounded.

Depth-first branch and bound (DFBB) [6] employs a heuristic function to eliminate parts of the search space that cannot contain an optimal solution. It continues after finding a first solution until the search space is completely exhausted. Whenever a better solution is found, the current solution path and its value are updated. Inferior subtrees, i. e., subtrees that are known to be worse than the current solution, are eliminated.

The alpha-beta algorithm [2] used in game tree searching is a variant of DFBB that operates on trees with alternating levels of AND and OR nodes [5]. Because the strength of play correlates to the depth of the search, much effort has been spent on devising efficient parallel implementations (► parallel heuristic search).

Best-First Search

BFS sorts the sequence of node expansions according to a heuristic function. The A* search algorithm [7] uses a heuristic evaluation function $f(n) = g(n) + h(n)$ to decide which successor node n to expand next. Here, $g(n)$ is the cost of the path from the initial state to the current node n and $h(n)$ is the estimated completion cost to a nearest goal state. If h does not overestimate the remaining cost, A* is guaranteed to find an optimal (least-cost) solution: it is said to be admissible. It does so with a minimal number of node expansions [9]—no other search algorithm (with the same heuristic h) can do better. This is possible, because A* keeps the search graph in memory, occupying $O(w^d)$ memory cells for trees of width w and depth d.

Best-first frontier search [4] also finds an optimal solution, but with a much lower space complexity than A*. It only keeps the frontier nodes in memory and discards the interior (closed) nodes. Care must be taken to ensure that the search frontier does not contain gaps that would allow the search to leak back into interior regions. The memory savings are most pronounced in directed acyclic graphs. In the worst case, that is, in trees of width w, it still saves a fraction of 1/w of the nodes that BFS would need to store.

Iterative-deepening A* (IDA*) [3] simulates A*'s best-first node expansion by a series of DFSs, each with the cost-bound $f(n)$ increased by the minimal amount. The cost-bound is initially set to the heuristic estimate of the root node, h (root). Then, for each iteration, the bound is increased to the minimum value that exceeded the previous bound. Like A* , IDA* is guaranteed to find an optimal solution [3], provided the heuristic estimate function h is admissible and never overestimates the path to the goal. IDA* obeys the same asymptotic branching factor as A* [7], if the number of newly expanded nodes grows exponentially with the search depth [3]. This growth rate, the heuristic branching factor, depends on the average number of applicable operators per node and the discrimination power of the heuristic function h.

Applications

Typical applications of heuristic search techniques may be found in many areas—not only in the fields of artificial intelligence and operations research, but also in other parts of computer science.

In the two-dimensional rectangular cutting-stock problem [1], we are given a set $R_s = \{(l_i, w_i), i = 1,\ldots, m\}$

of m rectangles of width w_i and length l_i that are to be cut out of a single rectangular stock sheet S. Assuming that S is of width W and that the theoretically unbounded length is L, the problem is to find an optimal cut with minimal length expansion. Since the elements R_i are cut after the cutting pattern has been determined, we can look at the problem as a bin-packing or vehicle-routing problem, which are also known to be nondeterministic polynomial-time (NP) complete [8].

Very large scale integration (VLSI) floorplan optimization is a stage in the design of VLSI chips, where the dimensions of the basic building blocks (cells) must be determined, subject to the minimization of the total chip layout area. This can be done with a BFS or a DFBB approach. Again, only small problem cases can be solved optimally, because VLSI floorplan optimization is also NP-complete.

In the satisfiability problem, it must be determined whether a Boolean formula containing binary variables in conjunctive normal form is satisfiable, that is, whether an assignment of truth values to the variables exists for which the formula is true.

The 15-puzzle benchmark in single-agent game-tree search consists of 15 square tiles located in a square tray of size 4×4. One square, the "blank square," is kept empty so that an orthogonally adjacent tile can slide into its position, thus leaving an empty position at its origin. The problem is to re-arrange a given initial configuration with the fewest number of moves into a goal configuration without lifting one tile over another. While it would seem easy to obtain any solution, finding optimal (shortest) solutions is NP-complete. The 15-puzzle spawns a search space of $16! \approx 2{\cdot}10^{13}$ states.

See also

- ► Asynchronous Distributed Optimization Algorithms
- ► Automatic Differentiation: Parallel Computation
- ► Load Balancing for Parallel Optimization Techniques
- ► Parallel Computing: Complexity Classes
- ► Parallel Computing: Models
- ► Parallel Heuristic Search
- ► Stochastic Network Problems: Massively Parallel Solution

References

1. Christofides N, Whitlock C (1977) An algorithm for two-dimensional cutting problems. Oper Res 25(1):30–44
2. Knuth DE, Moore RW (1975) An analysis of alpha-beta pruning. Artif Intell 6(4):293–326
3. Korf RE (1985) Depth-first iterative-deepening: An optimal admissible tree search. Artif Intell 27:97–109
4. Korf RE, Zhang W, Thayer I, Hohwald H (2005) Frontier Search J ACM 52:715–748
5. Kumar V, Nau DS, Kanal L (1988) A general branch-and-bound formulation for AND/OR graph and game-tree search. In: Kanal L, Kumar V (eds) Search in Artificial Intelligence. Springer, New York, pp 91–130
6. Lawler EL, Wood DE (1966) Branch and bound methods: A survey. Oper Res 14:600–719
7. Nilsson NJ (1980) Principles of artificial intelligence. Tioga Publ., Palo Alto
8. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: Algorithms and complexity. Prentice-Hall, Englewood Cliffs, NJ
9. Pearl J (1984) Heuristics. Intelligent search strategies for computer problem solving. Addison-Wesley, Reading

Heuristics for Maximum Clique and Independent Set

MARCELLO PELILLO
University Ca' Foscari di Venezia,
Venezia Mestre, Italy

MSC2000: 90C59, 05C69, 05C85, 68W01

Article Outline

Keywords
Sequential Greedy Heuristics
Local Search Heuristics
Advanced Search Heuristics
Simulated Annealing
  Neural Networks
  Genetic Algorithms
  Tabu Search
Continuous Based Heuristics
Miscellaneous
Conclusions
See also
References

Keywords

Heuristics; Algorithms; Clique; Independent set

Throughout this article, $G = (V, E)$ is an arbitrary undirected and weighted graph unless otherwise specified, where $V = (1, \ldots, n\}$ is the vertex set of G and $E \subseteq V \times V$ is its edge set. For each vertex $i \in V$, a positive weight w_i is associated with i, collected in the *weight vector* $w \in \mathbf{R}^n$. For a subset $S \subseteq V$, the weight of S is defined as $W(S) = \sum_{i \in S} w_i$, and $G(S) = (S, E \cap S \times S)$ is the *subgraph induced by S*. The *cardinality* of S, i. e., the number of its vertices, will be denoted by $|S|$.

A graph $G = (V, E)$ is *complete* if all its vertices are pairwise adjacent, i. e. $\forall i, j \in V$ with $i \neq j$, we have $(i, j) \in E$. A *clique* C is a subset of V such that $G(C)$ is complete. The *clique number* of G, denoted by $\omega(G)$ is the cardinality of the maximum clique. The maximum clique problem asks for cliques of maximum cardinality. The maximum weight clique problem asks for cliques of maximum weight. Given the weight vector $w \in \mathbf{R}^n$, the *weighted clique number* is the total weight of the maximum weight clique, and will be denoted by $\omega(G, w)$.

We should distinguish a *maximum clique* from a *maximal clique*. A maximal clique is one that is not a proper subset of any other clique. A maximum (weight) clique is a maximal clique that has the maximum cardinality (weight).

An *independent set* (also called *stable set* or *vertex packing*) is a subset of V whose elements are pairwise nonadjacent. The maximum independent set problem asks for an independent set of maximum cardinality. The size of a maximum independent set is the *stability number* of G, (denoted by $\alpha(G)$). The maximum weight independent set problem asks for an independent set of maximum weight. Given the weight vector $w \in \mathbf{R}^n$, the *weighted stability number*, denoted $\alpha(G, w)$, is the weight of the maximum weight independent set.

The *complement graph* of $G = (V, E)$ is the graph $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{(i, j) : i, j \in V, i \neq j \text{ and } (i, j) \notin E\}$. It is easy to see that S is a clique of G if and only if S is an independent set of $\overline{G}$. Any result or algorithm obtained for one of the two problems has its equivalent forms for the other one. Hence $\alpha(G) = \omega(\overline{G})$, more generally, $\alpha(G, w) = \omega(\overline{G}, w)$.

The maximum clique and independent set problems are well-known examples of intractable combinatorial optimization problems [18]. Apart from the theoretical interest around these problems, they also find practical applications in such diverse domains as computer vision, experimental design, information retrieval, fault tolerance, etc. Moreover, many important problems turn out to be easily reducible to them, and these include, for example, the Boolean satisfiability problem, the subgraph isomorphism problem, and the vertex covering problem. The maximum clique problem has also a certain historical value, as it was one of the first problems shown to be *NP*-complete in the now classical paper of R.M. Karp on computational complexity [64].

Due to their inherent computational complexity, exact algorithms are guaranteed to return a solution only in a time which increases exponentially with the number of vertices in the graph, and this makes them inapplicable even to moderately large problem instances. Moreover, a series of recent theoretical results show that the problems are in fact difficult to solve even in terms of approximation. Strong evidence of this fact came in 1991, when it was proved in [32] that if there is a polynomial time algorithm that approximates the maximum clique within a factor of $2^{log^{1-\epsilon} n}$, then any *NP*-hard problem can be solved in 'quasipolynomial' time (i. e., in $2^{log^{O(1)} n}$ time). The result was further refined in [6,7] one year later. Specifically, it was proved that there exists an $\epsilon > 0$ such that no polynomial time algorithm can approximate the size of the maximum clique within a factor of n^{ϵ}, unless $P = NP$. Developments along these lines can be found in [14,15,49].

In light of these negative results, much effort has recently been directed towards devising efficient heuristics for maximum clique and independent set, for which no formal guarantee of performance may be provided, but are anyway of interest in practical application. Lacking (almost by definition) a general theory of how these algorithms work, their evaluation is essentially based on massive experimentation. In order to facilitate comparisons among different heuristics, a set of benchmark graphs arising from different applications and problems has recently been constructed in conjunction with the 1993 DIMACS challenge on cliques, coloring and satisfiability [63].

In this article we provide an informal survey of recent heuristics for maximum clique and related problems, and up-to-date bibliographic pointers to the relevant literature. A more comprehensive review and bibliography can be found in [18].

Sequential Greedy Heuristics

Many approximation algorithms in the literature for the maximum clique problem are called *sequential greedy heuristics*. These heuristics generate a maximal clique through the repeated addition of a vertex into a partial clique, or the repeated deletion of a vertex from a set that is not a clique. Decisions on which vertex to be added in or moved out next are based on certain indicators associated with candidate vertices as, for example, the vertex degree. There is also a distinction between heuristics that update the indicators every time a vertex is added in or moved out, and those that do not. Examples of such heuristics can be found in [62,89]. The differences among these heuristics are their choice of indicators and how indicators are updated. A heuristic of this type can run very fast.

Local Search Heuristics

Let us define C_G to be the set of all the maximal cliques of G. Basically, a sequential greedy heuristic finds one set in C_G, hoping it is (close to) the optimal set, and stops. A possible way to improve our approximation solutions is to expand the search in C_G. For example, once we find a set $S \in C_G$, we can search its 'neighbors' to improve S. This leads to the class of the *local search heuristics* [2]. Depending on the neighborhood structure and how the search is performed, different local search heuristics result.

A well-known class of local search heuristics in the literature is the *k-interchange heuristics*. They are based on the *k-neighbor* of a feasible solution. In the case of the maximum clique problem, a set $C \in C_G$ is a k-neighbor of S if $|C \Delta S| \leq k$, where $k \leq |S|$. A k-interchange heuristic first finds a maximal clique $S \in C_G$, then it searches all the k-neighbors of S and returns the best clique found. Clearly, the main factors for the complexity of this class of heuristics are the size of the neighborhood and the searches involved. For example, in the k-interchange heuristic, the complexity grows roughly with $O(n^k)$.

A class of heuristics designed to search various sets of C_G is called the *randomized heuristics*. The main ingredient of this class of heuristics is the part that finds a random set in C_G. A possible way to do that is to include some random factors in the generation of a set of C_G. A randomized heuristic runs a heuristic (with random factors included) a number of times to find different sets over C_G. For example, we can randomize a sequential greedy heuristic and let it run N times. The complexity of a randomized heuristic depends on the complexity of the heuristic and the number N.

An elaborated implementation of the randomized heuristic for the maximum independent set problem can be found in [33], where local search is combined with randomized heuristic. The computational results in it indicated that the approach was effective in finding large cliques of randomly generated graphs. A different implementation of a randomized algorithm for the maximum independent set problem can be found in [5].

Advanced Search Heuristics

Local search algorithms are only capable of finding *local solutions* of an optimization problem. Powerful variations of the basic local search procedure have been developed which try to avoid this problem, many of which are inspired from various phenomena occurring in nature.

Simulated Annealing

In condensed-matter physics, the term 'annealing' refers to a physical process to obtain a pure lattice structure, where a solid is first heated up in a heat bath until it melts, and next cooled down slowly until it solidifies into a low-energy state. During the process, the free energy of the system is minimized. Simulated annealing, introduced in 1983 by S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi [65], is a randomized neighborhood search algorithm based on the physical annealing process. Here, the solutions of a combinatorial optimization problem correspond to the states of the physical system, and the cost of a solution is equivalent to the energy of the state.

In its original formulation, simulated annealing works essentially as follows. Initially, a tentative solution in the state space is somehow generated. A new neighboring state is then produced from the previous one and, if the value of the cost function f improves, the new state is accepted, otherwise it is accepted with probability $\exp\{\Delta f/\tau\}$, where Δf is the difference of the cost function between the new and the current state, and τ is a parameter usually called the *temperature* in

analogy with physical annealing, which is varied carefully during the optimization process. The algorithm proceeds iteratively this way until a stopping condition is met. One of the critical aspects of the algorithm relates to the choice of the proper 'cooling schedule,' i. e., how to decrease the temperature as the process evolves. While a logarithmic slow cooling schedule (yielding an exponential time algorithm) provably guarantees the exact solution, faster cooling schedules, producing acceptably good results, are in widespread use. Introductory textbooks describing both theoretical and practical issues of the algorithm are [1,66].

E. Aarts and J. Korst [1], without presenting any experimental result, suggested the use of simulated annealing for solving the independent set problem, using a *penalty function* approach. Here, the solution space is the set of all possible subsets of vertices of the graph G, and the problem is formulated as one of maximizing the cost function $f(V') = |V'| - \lambda\ |E'|$, where $|E'|$ is the number of edges in $G(V')$, and λ is a weighting factor exceeding 1.

M. Jerrum [61] conducted a theoretical analysis of the performance of a clique-finding *Metropolis process*, i. e., simulated annealing at fixed temperature, on random graphs. He proved that the expected time for the algorithm to find a clique that is only slightly bigger than that produced by a naive greedy heuristic grows faster than any polynomial in the number of vertices. This suggests that 'true' simulated annealing would be ineffective for the maximum clique problem.

Jerrum's conclusion seems to be contradicted by practical experience. In [56], S. Homer and M. Peinado compare the performance of three heuristics, namely the greedy heuristic developed in [62], a randomized version of the Boppana–Halldórsson subgraph-exclusion algorithm [24], and simulated annealing, over very large graphs. The simulated annealing algorithm was essentially that proposed by Aarts and Korst, with a simple cooling schedule. This penalty function approach was found to work better than the method in which only cliques are considered, as proposed in [61]. The algorithms were tested on various random graphs as well as on DIMACS benchmark graphs. The authors ran the algorithms over an SGI workstation for graphs with up to 10,000 vertices, and on a Connection Machine for graphs with up to 70,000 vertices. The overall conclusion was that simulated annealing outperforms the other competing algorithms; it also ranked among the best heuristics for maximum clique presented at the 1993 DIMACS challenge [63].

Neural Networks

Artificial neural networks (often simply referred to as 'neural networks') are massively parallel, distributed systems inspired by the anatomy and physiology of the cerebral cortex, which exhibit a number of useful properties such as learning and adaptation, universal approximation, and pattern recognition (see [50,52] for an introduction).

In the mid 1980s, J.J. Hopfield and D.W. Tank [57] showed that certain feedback continuous neural models are capable of finding approximate solutions to difficult optimization problems such as the traveling salesman problem [57]. This application was motivated by the property that the temporal evolution of these models is governed by a quadratic Liapunov function (typically called 'energy function' because of its analogy with physical systems) which is iteratively minimized as the process evolves. Since then, a variety of combinatorial optimization problems have been tackled within this framework. The customary approach is to formulate the original problem as one of energy minimization, and then to use a proper relaxation network to find minimizers of this function. Almost invariably, the algorithms developed so far incorporate techniques borrowed from statistical mechanics, in particular mean field theory, which allow one to escape from poor local solutions. We mention the articles [69,82] and the textbook [88] for surveys of this field. In [1], an excellent introduction to a particular class of neural networks (the Boltzmann machine) for combinatorial optimization is provided.

Early attempts at encoding the maximum clique and related problems in terms of a neural network were already done in the late 1980s in [1,12,44,83], and [84] (see also [85]). However, little or no experimental results were presented, thereby making it difficult to evaluate the merits of these algorithms. In [68], F. Lin and K. Lee used the quadratic zero-one formulation from [78] as the basis for their neural network heuristic. On random graphs with up to 300 vertices, they found their algorithm to be faster than the implicit enumerative algorithm in [26], while obtaining slightly worse results in terms of clique size.

T. Grossman [45] proposed a discrete, deterministic version of the Hopfield model for maximum clique, originally designed for an all-optical implementation. The model has a threshold parameter which determines the character of the stable states of the network. The author suggests an annealing strategy on this parameter, and an adaptive procedure to choose the network's initial state and threshold. On DIMACS graphs the algorithm performs satisfactorily but it does not compare well with more powerful heuristics such as simulated annealing.

A. Jagota [58] developed several variations of the Hopfield model, both discrete and continuous, to approximate maximum clique. He evaluated the performance of his algorithms over randomly generated graphs as well as on harder graphs obtained by generating cliques of varying size at random and taking their union. Experiments on graphs coming from the Solomonoff–Levin, or 'universal' distribution are also presented in [59]. The best results were obtained using a stochastic steepest descent dynamics and a mean-field annealing algorithm, an efficient deterministic approximation of simulated annealing. These algorithms, however, were also the slowest, and this motivated Jagota et al. [60] to improve their running time. The mean-field annealing heuristic was implemented on a 32-processor Connection Machine, and a two-temperature annealing strategy was used. Additionally, a 'reinforcement learning' strategy was developed for the stochastic steepest descent heuristic, to automatically adjust its internal parameters as the process evolves. On various benchmark graphs, all their algorithms obtained significantly larger cliques than other simpler heuristics but ran slightly slower. Compared to more sophisticated heuristics, they obtained significantly smaller cliques on average but were considerably faster.

M. Pelillo [80] takes a completely different approach to the problem, by exploiting a continuous formulation of maximum clique and the dynamical properties of the so-called relaxation labeling networks. His algorithm is described in the next section.

Genetic Algorithms

Genetic algorithms are parallel search procedures inspired from the mechanisms of evolution in natural systems [45,55]. In contrast to more traditional optimization techniques, they work on a population of points, which in the genetic algorithm terminology, are called chromosomes or individuals. In the simplest and most popular implementation, chromosomes are simply long strings of bits. Each individual has an associated 'fitness' value which determines its probability of survival in the next 'generation': the higher the fitness, the higher the probability of survival. The genetic algorithm starts out with an initial population of members generally chosen at random and, in its simplest version, makes use of three basic operators: reproduction, crossover and mutation. Reproduction usually consists of choosing the chromosomes to be copied in the next generation according to a probability proportional to their fitness. After reproduction, the crossover operator is applied between pairs of selected individuals to produce new offsprings. The operator consists of swapping two or more subsegments of the the strings corresponding to the two chosen individuals. Finally, the mutation operator is applied, which randomly reverses the value of every bit within a chromosome with a fixed probability. The procedure just described is sometimes referred to as the 'simple' genetic algorithm [45].

One of the earliest attempts to solve the maximum clique problem using genetic algorithms was done in 1993 by B. Carter and K. Park [27]. After showing the weakness of the simple genetic algorithm in finding large cliques, even on small random graphs, they introduced several modifications in an attempt to improve performance. However, despite their efforts they did not get satisfactory results, and their general conclusion was that genetic algorithms need to be heavily customized in order to be competitive with traditional approaches, and that they are computationally very expensive. In a later study [79], genetic algorithms were proven to be less effective than simulated annealing. At almost the same time, T. Bäck and S. Khuri [8], working on the maximum independent set problem, arrived at the opposite conclusion. By using a straightforward, general-purpose genetic algorithm called GENEsYs and a suitable fitness function which included a graded penalty term to penalize infeasible solutions, they got interesting results over random and regular graphs with up to 200 vertices. These results indicate that the choice of the fitness function is crucial for genetic algorithms to provide satisfactory results.

A.S. Murthy et al. [74] also experimented with a genetic algorithm using a novel 'partial copy crossover', and a modified mutation operator. However, they presented results over very small (i. e., up to 50 vertices) graphs, thereby making it difficult to properly evaluate the algorithm.

T.N. Bui and P.H. Eppley [25] obtained encouraging results by using a hybrid strategy which incorporates a local optimization step at each generation of the genetic algorithm, and a vertex-ordering preprocessing phase. They tested the algorithm over some DIMACS graphs getting results comparable to that in[39]

Instead of using the standard binary representation for chromosomes, J.A. Foster and T. Soule [36] employed an integer-based encoding scheme. Moreover, they used a time weighting fitness function similar in spirit to those in [27]. The results obtained are interesting, but still not comparable to those obtained using more traditional search heuristics.

C. Fleurent and J.A. Ferland [35] developed a general-purpose system for solving graph coloring, maximum clique, and satisfiability problems. As far as the maximum clique problem is concerned, they conducted several experiments using a hybrid genetic search scheme which incorporates tabu search and other local search techniques as alternative mutation operators. The results presented are encouraging, but running time is quite high.

In [53], M. Hifi modifies the basic genetic algorithm in several aspects:

a) a particular crossover operator creates two new different children;
b) the mutation operator is replaced by a specific heuristic feasibility transition adapted to the weighted maximum stable set problem.

This approach is also easily parallelizable. Experimental results on randomly generated graphs and also some (unweighted) instances from the DIMACS testbed [63] are reported to validate this approach.

Finally, E. Marchiori [71] has developed a simple heuristic-based genetic algorithm which consists of a combination of the simple genetic algorithm and a naive greedy heuristic procedure. Unlike previous approaches, here there is a neat division of labor, the search for a large subgraph and the search for a clique being incorporated into the fitness function and the heuristic procedure, respectively. The algorithm outperforms previous genetic-based clique finding procedures over various DIMACS graphs, both in terms of quality of solutions and speed.

Tabu Search

Tabu search, introduced independently by F. Glover [41,42] and P. Hansen and B. Jaumard [48], is a modified local search algorithm, in which a prohibition-based strategy is employed to avoid cycles in the search trajectories and to explore new regions in the search space. At each step of the algorithm, the next solution visited is always chosen to be the best *legal neighbor* of the current state, even if its cost is worse than the current solution. The set of legal neighbors is restricted by one or more *tabu lists* which prevent the algorithm to go back to recently visited solutions. These lists are used to store historical information on the path followed by the search procedure. Sometimes the tabu restriction is relaxed, and tabu solutions are accepted if they satisfy some *aspiration level* condition. The standard example of a tabu list is one which contains the last k solutions examined, where k may be fixed or variable. Additional lists containing the last modifications performed, i. e., changes occurred when moving from one solution to the next, are also common. These types of lists are referred to as *short-term memories*; other forms of memories are also used to intensify the search in a promising region or to diversify the search to unexplored areas. Details on the algorithm and its variants can be found in [43] and [51].

In 1989, C. Friden et al. [37] proposed a heuristic for the maximum independent set problem based on tabu search. The size of the independent set to search for is fixed, and the algorithm tries to minimize the number of edges in the current subset of vertices. They used three tabu lists: one for storing the last visited solutions and the other two to contain the last introduced/deleted vertices. They showed that by using hashing for implementing the first list and choosing a small value for the dimensions of the other two lists, a best neighbor may be found in almost constant time.

In [38,86], three variants of tabu search for maximum clique are presented. Here the search space consists of complete subgraphs whose size has to be maximized. The first two versions are deterministic algorithms in which no sampling of the neighborhood is

performed. The main difference between the two algorithms is that the first one uses just one tabu list (of the last solutions visited), while the second one uses an additional list (with an associated aspiration mechanism) containing the last vertices deleted. Also, two diversification strategies were implemented. The third algorithm is probabilistic in nature, and uses the same two tabu lists and aspiration mechanism as the second one. It differs from it because it performs a random sampling of the neighborhood, and also because it allows for multiple deletion of vertices in the current solution. Here no diversification strategy was used. In [38,86] results on randomly generated graphs were presented and the algorithms were shown to be very effective. P. Soriano and M. Gendreau [87] tested their algorithms over the DIMACS benchmark graphs and the results confirmed the early conclusions.

R. Battiti and M. Protasi [13] extended the tabu search framework by introducing a reactive local search method. They modified a previously introduced reactive scheme by exploiting the particular neighborhood structure of the maximum clique problem. In general reactive schemes aim at avoiding the manual selection of control parameters by means of an internal feedback loop. Battiti and Protasi's algorithm adopts such a strategy to automatically determine the so-called prohibition parameter k, i. e., the size of the tabu list. Also an explicit memory-influenced restart procedure is activated periodically to introduce diversification. The search space consists of all possible cliques, as in the approach by Friden et al., and the function to be maximized is the clique size. The worst-case computational complexity of this algorithm is $O(\max\{n, m\})$, where n and m are the number of vertices and edges of the graph respectively. They noticed, however, that in practice, the number of operations tends to be proportional to the average degree of the vertices of the complement graph. They tested their algorithm over many DIMACS benchmark graphs obtaining better results then those presented at the DIMACS workshop in competitive time.

Continuous Based Heuristics

In 1965, T.S. Motzkin and E.G. Straus [73] established a remarkable connection between the maximum clique problem and a certain quadratic programming problem. Let $G = (V, E)$ be an undirected (unweighted) graph and let Δ denote the standard simplex in the n-dimensional Euclidean space $\mathbf{R}^n$:

$$\Delta = \left\{x \in \mathbf{R}^n : \; x_i \geq 0 \text{ for all } i \in V, \; e^\top x = 1\right\},$$

where the letter e is reserved for a vector of appropriate length, consisting of unit entries (hence $e^\top x = \sum_{i \in V} x_i$).

Now, consider the following quadratic function, sometimes called the *Lagrangian* of G:

$$g(x) = x^\top A_G x, \tag{1}$$

where $A_G = (a_{ij})$ is the adjacency matrix of G, i. e. the symmetric $n \times n$ matrix where $a_{ij} = 1$ if $(i, j) \in E$ and $a_{ij} = 0$ if $(i, j) \notin E$, and let x^* be a global maximizer of g on Δ. In [73] it is proved that the clique number of G is related to $g(x^*)$ by the following formula:

$$\omega(G) = \frac{1}{1 - g(x^*)}.$$

Additionally, it is shown that a subset of vertices S is a maximum clique of G if and only if its *characteristic vector* x^S, which is the vector of Δ defined as $x_i^S = 1/|S|$ if $i \in S$ and $x_i^S = 0$ otherwise, is a global maximizer of g on Δ. In [40,81], the Motzkin–Straus theorem has been extended by providing a characterization of maximal cliques in terms of local maximizers of g on Δ.

One drawback associated with the original Motzkin–Straus formulation relates to the existence of spurious solutions, i. e., maximizers of g which are not in the form of characteristic vectors [77,81]. In principle, spurious solutions represent a problem since, while providing information about the cardinality of the maximum clique, they do not allow us to easily extract its vertices.

During the 1990s, there has been much interest around the Motzkin–Straus and related continuous formulations of the maximum clique problem. They suggest in fact a fundamentally new way of solving the maximum clique problem, by allowing us to shift from the discrete to the continuous domain in an elegant manner. As pointed out in [76], continuous formulations of discrete optimization problems turn out to be particularly attractive. They not only allow us to exploit the full arsenal of continuous optimization techniques, thereby leading to the development of new algorithms, but may also reveal unexpected theoretical properties.

In [77], P.M. Pardalos and A.T. Phillips developed a global optimization approach based on the Motzkin–Straus formulation and implemented an iterative clique retrieval process to find the vertices of the maximum clique. However, due to its high computational cost they were not able to run the algorithm over graphs with more than 75 vertices.

Pelillo [80] used *relaxation labeling algorithms* to approximately determining the size of the maximum clique using the original Motzkin–Straus formulation. These are parallel, distributed algorithms developed and studied in computer vision and pattern recognition, which are also surprisingly related to *replicator equations*, a class of dynamical systems widely studied in evolutionary game theory and related fields [54], Heuristics for maximum clique and independent set. The model operates in the simplex Δ and possesses a quadratic Liapunov function which drives its dynamical behavior. It is these properties that naturally suggest using them as a local optimization algorithm for the Motzkin–Straus program. The algorithm is especially suited for parallel implementation, and is attractive for its operational simplicity, since no parameters need to be determined. Extensive simulations over random graphs with up to 2000 vertices have demonstrated the effectiveness of the approach and showed that the algorithm outperforms previous neural network heuristics.

In order to avoid time-consuming iterative procedures to extract the vertices of the clique, L.E. Gibbons, D.W. Hearn and Pardalos [39] have proposed a heuristic which is based on a parameterized formulation of the Motzkin–Straus program. They consider the problem of minimizing the function:

$$h(x) = \frac{1}{2}x^\top A_{\overline{G}}x + \left(\sum_{i=1}^{n} x_i - 1\right)^2$$

on the domain:

$$S(k) = \left\{ x \in \mathbf{R}^n : \sum_{i=1}^{n} x_i^2 \le \frac{1}{k},\ x_i \ge 0,\ \forall i \right\},$$

where k is a fixed parameter. Let x^* be a global minimizer of h on $S(k)$, and let $V(k) = h(x^*)$. In [39] it is proved that $V(k) = 0$ if and only if there exists an independent set S of $\overline{G}$ with size $|S| \ge k$. Moreover, the vertices of $\overline{G}$ associated with the indices of the positive components of x^* form an independent set of size greater than or equal k.

These properties motivated the following procedure to find a maximum independent set of $\overline{G}$ or, equivalently, a maximum clique of G. Minimize the function h over $S(k)$, for different values of k between predetermined upper and lower bounds. If $V(k) = 0$ and $V(k+1) \neq 0$ for some k, then the maximum clique of G has size k, and its vertices are determined by the positive components of the solution. Since minimizing h on $S(k)$ is a difficult problem, they developed a heuristic based on the observation that by removing the nonnegativity constraints, the problem is that of minimizing a quadratic form over a sphere, a problem which is solvable in polynomial time. However, in so doing a heuristic procedure is needed to round the approximate solutions of this new problem to approximate solutions of the original one. Moreover, since the problem is solved approximately, we have to find the value of the spherical constraint $1k$ which yields the largest independent set. A careful choice of k is therefore needed. The resulting algorithm was tested over various DIMACS benchmark graphs [63] and the results obtained confirmed the effectiveness of the approach.

The spurious solution problem has been solved in [16]. Consider the following regularized version of function g:

$$\widehat{g}(x) = x^\top A_G x + \frac{1}{2}x^\top x \tag{2}$$

which is obtained from (1) by substituting the adjacency matrix A_G of G with

$$\widehat{A}_G = A_G + \frac{1}{2}I,$$

where I is the identity matrix. Unlike the Motzkin–Straus formulation, it can be proved that all maximizers of $\widehat{g}$ on Δ are strict, and are characteristic vectors of maximal/maximum cliques in the graph. In an exact sense, therefore, a one-to-one correspondence exists between maximal cliques and local maximizers of $\widehat{g}$ in Δ on the one hand and maximum cliques and global maximizers on the other hand. In [16,20], replicator equations are used in conjunction to this spurious-free formulation to find maximal cliques of G. Note that here the vertices comprising the clique are directly given by the positive components of the converged vectors, and no iterative procedure is needed to determine

them, as in [77]. The results obtained over a set of random as well as DIMACS benchmark graphs were encouraging, especially considering that replicator equations do not incorporate any mechanism to escape from local optimal solutions. This suggests that the basins of attraction of the global solution with respect to the quadratic functions g and $\widehat{g}$ are quite large; for a thorough empirical analysis see also [23]. One may wonder whether a subtle choice of initial conditions and/or a variant of the dynamics may significantly improve the results, but experiments in [22] indicate this is not the case.

In [19] the properties of the following function are studied:

$$\widehat{g}_\alpha(x) = x^\top A_G x + \alpha x^\top x.$$

It is shown that when α is positive all the properties enjoyed by the standard regularization approach [16] hold true. Specifically, in this case a one-to-one correspondence between local/global maximizers in the continuous space and local/global solutions in the discrete space exists. For negative α's an interesting picture emerges: as the absolute value of α grows larger, local maximizers corresponding to maximal cliques disappear. In [19], bounds on the parameter α are derived which affect the stability of these solutions. These results have suggested an *annealed replication heuristic*, which consists of starting from a large negative α and then properly reducing it during the optimization process. For each value of α standard replicator equations are run in order to obtain local solutions of the corresponding objective function. The rationale behind this idea is that for values of α with a proper large absolute value only local solutions corresponding to large maximal cliques will survive, together with various spurious maximizers. As the value of α is reduced, spurious solutions disappear and smaller maximal cliques become stable. An annealing schedule is proposed which is based on the assumption that the graphs being considered are random. In this respect, the proposed procedure differs from usual simulated annealing approaches, which mostly use a 'black-box' cooling schedule. Experiments conducted over several DIMACS benchmark graphs confirm the effectiveness of the proposed approach and the robustness of the annealing strategy. The overall conclusion is that the annealing procedure does help to avoid inefficient local solutions, by initially driving the dynamics towards promising regions in state space, and then refining the search as the annealing parameter is increased.

The Motzkin–Straus theorem has been generalized to the weighted case in [40]. Note that the Motzkin–Straus program can be reformulated as a minimization problem by considering the function

$$f(x) = x^\top (I + A_{\overline{G}})x,$$

where $A_{\overline{G}}$ is the adjacency matrix of the complement graph $\overline{G}$. It is straightforward to see that if x^* is a global minimizer of f in Δ, then we have: $\omega(G) = 1/\mathrm{f}(x^*)$. This is simply a different formulation of the Motzkin–Straus theorem. Given a weighted graph $G = (V, E)$ with weight vector w, let $\mathcal{M}(G, w)$ be the class of symmetric $n \times n$ matrices $B = (b_{ij})_{i,j \in V}$ defined as $2b_{ij} \geq b_{ii} + b_{jj}$ if $(i, j) \notin E$ and $b_{ij} = 0$ otherwise, and $b_{ii} = 1/w_i$ for all $i \in V$.

Given the following quadratic program, which is in general indefinite,

$$\begin{cases} \min & f(x) = x^\top Bx \\ \text{s.t.} & x \in \Delta, \end{cases} \tag{3}$$

in [40] it is shown that for any $B \in \mathcal{M}(G, w)$ we have:

$$\omega(G, w) = \frac{1}{f(x^*)},$$

where x^* is a global minimizer of program (3). Furthermore, denote by x^S the *weighted characteristic vector* of S, which is a vector with coordinates $x_i^S = w_i/W(S)$ if $i \in S$ and $x_i^S = 0$ otherwise. It can be seen that a subset S of vertices of a weighted graph G is a maximum weight clique if and only if its characteristic vector x^S is a global minimizer of (3). Notice that the matrix $I + A_{\overline{G}}$ belongs to $\mathcal{M}(G, e)$. In other words, the Motzkin–Straus theorem turns out to be a special case of the preceding result.

As in the unweighted case, the existence of spurious solutions entails the lack of one-to-one correspondence between the solutions of the continuous problem and those of the original, discrete one. In [21] these spurious solutions are characterized and a regularized version which avoids this kind of problems is introduced, exactly as in the unweighted case (see also [17]). Replicator equations are then used to find maximal weight

cliques in weighted graphs, using this formulation. Experiments with this approach on both random graphs and DIMACS graphs are reported. The results obtained are compared with those produced by a very efficient maximum weight clique algorithm of the branch and bound variety. The algorithm performed remarkably well especially on large and dense graphs, and it was typically an order of magnitude more efficient than its competitor.

Finally, we mention the work by Massaro and Pelillo [72], who transformed the Motzkin–Straus program into a linear complementarity problem [31], and then solved it using a variation of Lemke's well-known algorithm [67]. The preliminary results obtained over many weighted and unweighted DIMACS graphs show that this approach substantially outperforms all other continuous based heuristics.

Miscellaneous

Another type of heuristics that finds a maximal clique of G is called the *subgraph approach* (see [11]). It is based on the fact that a maximum clique C of a subgraph $G' \subseteq G$ is also a clique of G. The subgraph approach first finds a subgraph $G' \subseteq G$ such that the maximum clique of G' can be found in polynomial time. Then it finds a maximum clique of G' and use it as the approximation solution. The advantage of this approach is that in finding the maximum clique $C \subseteq G'$, one has (implicitly) searched many other cliques of G' ($C_{G'} \subseteq C_G$). Because of the special structure of G', this implicit search can be done efficiently. In [11], G' is a maximal induced triangulated subgraph of G. Since many classes of graphs have polynomial algorithms for the maximum clique problem, the same idea also applies there. For example, the class of edge-maximal triangulated subgraphs was chosen in [9,90], and [91]. Some of the greedy heuristics, randomized heuristics and subgraph approach heuristics are compared in [90] and [91] on randomly generated weighted and unweighted graphs.

Various new heuristics were presented at the 1993 DIMACS challenge devoted to clique, coloring and satisfiability [63]. In particular, in [10] an algorithm is proposed which is based on the observation that finding the maximum clique in the union of two cliques can be done using bipartite matching techniques. In [46] restricted backtracking is used to provide a trade-off between the size of the clique and the completeness of the search. In [70] an edge projection technique is proposed to obtain a new upper bound heuristic for the maximum independent set problem. This procedure was used, in conjunction with the Balas–Yu branching rule [11], to develop an exact branch and bound algorithm which works well especially on sparse graphs.

See [3] for a new population-based optimization heuristic inspired by the natural behavior of human or animal scouts in exploring unknown regions, and applied it to maximum clique. The results obtained over a few DIMACS graphs are comparable with those obtained using continuous-based heuristics but are inferior to those obtained by reactive local search.

Recently, DNA computing [4] has also emerged as a potential technique for the maximum clique problem [75,92]. The major advantage of DNA computing is its high parallelism, but at present the size of graphs this algorithm can handle is limited to a few tens.

Additional heuristics for the maximum clique/independent set and related problems on arbitrary or special class of graphs can be found in [28,29,30,34].

Conclusions

During the 1990s, research on the maximum clique and related problems has yielded many interesting heuristics. This article has provided an expository survey on these algorithms and an up-to-date bibliography (as of 2000). However, the activity in this field is so extensive that a survey of this nature is outdated before it is written.

See also

- ► Graph Coloring
- ► Greedy Randomized Adaptive Search Procedures
- ► Replicator Dynamics in Combinatorial Optimization

References

1. Aarts E, Korst J (1989) Simulated annealing and Boltzmann machines. Wiley, New York
2. Aarts E, Lenstra JK (eds) (1997) Local search in combinatorial optimization. Wiley, New York
3. Abbattista F, Bellifemmine F, Dalbis D (1998) The Scout algorithm applied to the maximum clique problem. Ad-

vances in Soft Computing—Engineering and Design. Springer, Berlin
4. Adleman LM (1994) Molecular computation of solutions to combinatorial optimization. Science 266:1021–1024
5. Alon N, Babai L, Itai A (1986) A fast and simple randomized parallel algorithm for the maximal independent set problem. J Algorithms 7:567–583
6. Arora S, Lund C, Motwani R, Sudan M, Szegedy M (1992) Proof verification and the hardness of approximation problems. Proc. 33rd Ann. Symp. Found. Comput. Sci., Pittsburgh, pp 14–23
7. Arora S, Safra S (1992) Probabilistic checking of proofs: A new characterization of NP. Proc. 33rd Ann. Symp. Found. Comput. Sci., Pittsburgh, pp 2–13
8. Bäck T, Khuri S (1994) An evolutionary heuristic for the maximum independent set problem. Proc. 1st IEEE Conf. Evolutionary Comput., 531–535
9. Balas E (1986) A fast algorithm for finding an edge-maximal subgraph with a TR-formative coloring. Discrete Appl Math 15:123–134
10. Balas E, Niehaus W (1996) Finding large cliques in arbitrary graphs by bipartite matching. In: Johnson DS, Trick MA (eds) Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, DIMACS 26. AMS, Providence, RI, pp 29–51
11. Balas E, Yu CS (1986) Finding a maximum clique in an arbitrary graph. SIAM J Comput 14:1054–1068
12. Ballard DH, Gardner PC, Srinivas MA (1987) Graph problems and connectionist architectures. Techn Report Dept Comput Sci Univ Rochester 167
13. Battiti R, Protasi M (1995) Reactive local search for the maximum clique problem. TR-95-052 Techn Report Internat Comput Sci Inst Berkeley, to appear in Algorithmica
14. Bellare M, Goldwasser S, Lund C, Russell A (1993) Efficient probabilistically checkable proofs and application to approximation. Proc. 25th Ann. ACM Symp. Theory Comput., pp 294–304
15. Bellare M, Goldwasser S, Sudan M (1995) Free bits, PCPs and non-approximability – Towards tight results. Proc. 36th Ann. Symp. Found. Comput. Sci., pp 422–431
16. Bomze IM (1997) Evolution towards the maximum clique. J Global Optim 10:143–164
17. Bomze IM (1998) On standard quadratic optimization problems. J Global Optim 13:369–387
18. Bomze IM, Budinich M, Pardalos PM, Pelillo M (1999) The maximum clique problem. In: Du D-Z, Pardalos PM (eds) Handbook Combinatorial Optim., Suppl. A. Kluwer, Dordrecht, pp 1–74
19. Bomze IM, Budinich M, Pelillo M, Rossi C (2000) An new "annealed" heuristic for the maximum clique problem. In: Pardalos PM (eds) Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems. Kluwer, Dordrecht, pp 78–95
20. Bomze IM, Pelillo M, Giacomini R (1997) Evolutionary approach to the maximum clique problem: Empirical evidence on a larger scale. In: Bomze IM, Csendes T, Horst R, Pardalos PM (eds) Developments in Global Optimization. Kluwer, Dordrecht, pp 95–108
21. Bomze IM, Pelillo M, Stix V (2000) Approximating the maximum weight clique using replicator dynamics. IEEE Trans Neural Networks 11(6)
22. Bomze IM, Rendl F (1998) Replicator dynamics for evolution towards the maximum clique: Variations and experiments. In: De Leone R, Murli A, Pardalos PM, Toraldo G (eds) High Performance Algorithms and Software in Nonlinear Optimization. Kluwer, Dordrecht, pp 53–67
23. Bomze IM, Stix V (1999) Genetic engineering via negative fitness: Evolutionary dynamics for global optimization. Ann Oper Res 90
24. Boppana R, Halldóorsson MM (1992) Approximating maximum independent sets by excluding subgraphs. BIT 32:180–196
25. Bui TN, Eppley PH (1995) A hybrid genetic algorithm for the maximum clique problem. Proc. 6th Internat. Conf. Genetic Algorithms, pp 478–484
26. Carraghan R, Pardalos PM (1990) An exact algorithm for the maximum clique problem. Oper Res Lett 9:375–382
27. Carter B, Park K (1993) How good are genetic algorithms at finding large cliques: An experimental study. Techn Report Comput Sci Dept Boston Univ no. BU-CS-93-015
28. Chiba N, Nishizeki T, Saito N (1983) An algorithm for finding a large independent set in planar graphs. Networks 13:247–252
29. Chrobak M, Naor J (1991) An efficient parallel algorithm for computing a large independent set in a planar graph. Algorithmica 6:801–815
30. Chvátal V (1979) A greedy heuristic for the set-covering problem. Math Oper Res 4:233–23
31. Cottle RW, Pang J, Stone RE (1992) The linear complementarity problem. AP
32. Feige U, Goldwasser S, Lováasz L, Safra S, Szegedy M (1991) Approximating clique is almost NP-complete. Proc. 32nd Ann. Symp. Found. Comput. Sci., San Juan, Puerto Rico, pp 2–12
33. Feo TA, Resende MGC, Smith SH (1994) A greedy randomized adaptive search procedure for maximum independent set. Oper Res 42:860–878
34. Fisher ML, Wolsey LA (1982) On the greedy heuristic for continuous covering and packing problems. SIAM J Alg Discrete Meth 3:584–591
35. Fleurent C, Ferland JA (1996) Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In: Johnson DS, Trick MA (eds) Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, DIMACS 26. AMS, Providence, RI, pp 619–652
36. Foster JA, Soule T (1995) Using genetic algorithms to find maximum cliques. Techn Report Dept Comput Sci Univ Idaho no. LAL 95-12
37. Friden C, Hertz A, de Werra M (1989) STABULUS: A tech-

nique for finding stable sets in large graphs with tabu search. Computing 42:35–44
38. Gendreau A, Salvail L, Soriano P (1993) Solving the maximum clique problem using a tabu search approach. Ann Oper Res 41:385–403
39. Gibbons LE, Hearn DW, Pardalos PM (1996) A continuous based heuristic for the maximum clique problem. In: Johnson DS, Trick MA (eds) Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, DIMACS 26. AMS, Providence, RI, pp 103–124
40. Gibbons LE, Hearn DW, Pardalos PM, Ramana MV (1997) Continuous characterizations of the maximum clique problem. Math Oper Res 22:754–768
41. Glover F (1989) Tabu search–Part I. ORSA J Comput 1:190–260
42. Glover F (1990) Tabusearch–Part II. ORSA J Comput 2:4–32
43. Glover F, Laguna M (1993) Tabu search. In: Reeves C (ed) Modern Heuristic Techniques for Combinatorial Problems. Blackwell, Oxford pp 70–141
44. Godbeer GH, Lipscomb J, Luby M (1988) On the computational complexity of finding stable state vectors in connectionist models (Hopfield nets). Techn Report Dept Comput Sci Univ Toronto 208,
45. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA
46. Goldberg MK, Rivenburgh RD (1996) Constructing cliques using restricted backtracking. In: Johnson DS, Trick MA (eds) Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, DIMACS 26. AMS, Providence, RI, pp 89–101
47. Grossman T (1996) Applying the INN model to the max clique problem. In: Johnson DS, Trick MA (eds) Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, DIMACS 26. AMS, Providence, RI, pp 125–146
48. Hansen P, Jaumard B (1990) Algorithms for the maximum satisfiability problem. Computing 44:279–303
49. Håstad J (1996) Clique is hard to approximate within $n^{1-\epsilon'}$:. Proc. 37th Ann. Symp. Found. Comput. Sci., pp 627–636
50. Haykin S (1994) Neural networks: A comprehensive foundation. MacMillan, New York
51. Hertz A, Taillard E, de Werra D (1997) Tabu search. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, New York pp 121–136
52. Hertz J, Krogh A, Palmer RG (1991) Introduction to the theory of neural computation. Addison-Wesley, Reading, MA
53. Hifi M (1997) A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems. J Oper Res Soc 48:612–622
54. Hofbauer J, Sigmund K (1998) Evolutionary games and population dYnamics. Cambridge Univ. Press, Cambridge
55. Holland JH (1975) Adaptation in natural and artificial systems. Univ. Michigan Press, Ann Arbor, MI
56. Homer S, Peinado M (1996) Experiments with polynomial-time CLIQUE approximation algorithms on very large graphs. In: Johnson DS, Trick MA (eds) Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, DIMACS 26. AMS, Providence, RI, pp 147–167
57. Hopfield JJ, Tank DW (1985) Neural computation of decisions in optimization problems. Biol Cybern 52:141–152
58. Jagota A (1995) Approximating maximum clique with a Hopfield neural network. IEEE Trans Neural Networks 6:724–735
59. Jagota A, Regan KW (1997) Performance of neural net heuristics for maximum clique on diverse highly compressible graphs. J Global Optim 10:439–465
60. Jagota A, Sanchis L, Ganesan R (1996) Approximately solving maximum clique using neural networks and related heuristics. In: Johnson DS, Trick MA (eds) Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, DIMACS 26. AMS, Providence, RI, pp 169–204
61. Jerrum M (1992) Large cliques elude the Metropolis process. Random Struct Algorithms 3:347–359
62. Johnson DS (1974) Approximation algorithms for combinatorial problems. J Comput Syst Sci 9:256–278
63. Johnson DS, Trick MA (eds) (1996) Cliques, coloring, and satisfiability: 2nd DIMACS implementation challenge, DIMACS 26. AMS, Providence, RI
64. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations. Plenum, New York, pp 85–103
65. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
66. van Laarhoven PJM, Aarts EHL (1987) Simulated annealing: Theory and applications. Reidel, London
67. Lemke CE (1965) Bimatrix equilibrium points and mathematical programming. Managem Sci 11:681–689
68. Lin F, Lee K (1992) A parallel computation network for the maximum clique problem. Proc. 1st Internat. Conf. Fuzzy Theory Tech.,
69. Looi C-K (1992) Neural network methods in combinatorial optimization. Comput Oper Res 19:191–208
70. Mannino C, Sassano A (1996) Edge projection and the maximum cardinality stable set problem. In: Johnson DS, Trick MA (eds) Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, DIMACS 26. AMS, Providence, RI, pp 205–219
71. Marchiori E (1998) A simple heuristic based genetic algorithm for the maximum clique problem. Proc. ACM Symp. Appl. Comput., pp 366–373
72. Massaro A, Pelillo M (2000) A pivoting-based heuristic for the maximum clique problem. Presented at the Internat. Conf. Advances in Convex Analysis and Global Optimization, Samos, June 2000
73. Motzkin TS, Straus EG (1965) Maxima for graphs and a new proof of a theorem of Turán. Canad J Math 17:533–540
74. Murthy AS, Parthasarathy G, Sastry VUK (1994) Clique finding–A genetic approach. Proc. 1st IEEE Conf. Evolutionary Comput., pp 18–21

75. Ouyang Q, Kaplan PD, Liu S, Libchaber A (1997) DNA solutions of the maximal clique problem. Science 278:1446–449
76. Pardalos PM (1996) Continuous approaches to discrete optimization problems. In: Di Pillo G, Giannessi F (eds) Nonlinear Optimization and Applications. Plenum, New York pp 313–328
77. Pardalos PM, Phillips AT (1990) A global optimization approach for solving the maximum clique problem. Internat J Comput Math 33:209–216
78. Pardalos PM, Rodgers GP (1990) Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing 45:131–144
79. Park K, Carter B (1994) On the effectiveness of genetic search in combinatorial optimization. no.BU-CS-9-010 Techn Report Computer Sci Dept Boston Univ
80. Pelillo M (1995) Relaxation labeling networks for the maximum clique problem. J Artif Neural Networks 2:313–328
81. Pelillo M, Jagota A (1995) Feasible and infeasible maxima in a quadratic program for maximum clique. J Artif Neural Networks 2:411–420
82. Peterson C, Söderberg B (1997) Artificial neural networks. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, New York pp 173–213
83. Ramanujam J, Sadayappan P (1988) Optimization by neural networks. Proc. IEEE Internat. Conf. Neural Networks, pp 325–332
84. Shrivastava Y, Dasgupta S, Reddy SM (1990) Neural network solutions to a graph theoretic problem. Proc. IEEE Internat. Symp. Circuits Syst., pp 2528–2531
85. Shrivastava Y, Dasgupta S, Reddy SM (1992) Guaranteed convergence in a class of Hopfield networks. IEEE Trans Neural Networks 3:951–961
86. Soriano P, Gendreau M (1996) Diversification strategies in tabu search algorithms for the maximum clique problem. Ann Oper Res 63:189–207
87. Soriano P, Gendreau M (1996) Tabu search algorithms for the maximum clique problem. In: Johnson DS, Trick MA (eds) Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, DIMACS 26. AMS, Providence, RI, pp 221–242
88. Takefuji Y (1992) Neural network parallel computing. Kluwer, Dordrecht
89. Tomita E, Mitsuma S, Takahashi H (1988) Two algorithms for finding a near-maximum clique. Techn Report UEC-TR-C1
90. Xue J (1991) Fast algorithms for vertex packing and related problems. PhD Thesis GSIA Carnegie-Mellon Univ.
91. Xue J (1994) Edge-maximal triangulated subgraphs and heuristics for the maximum clique problem. Networks 24:109–120
92. Zhang B-T, Shin S-Y (1998) Code optimization for DNA computing of maximal clique's. Advances in Soft Computing–Engineering and Design. Springer, Berlin

High-order Maximum Principle for Abnormal Extremals

Urszula Ledzewicz[1], Heinz Schättler[2]
[1] Department Math. and Statist., Southern Illinois University at Edwardsville, Edwardsville, USA
[2] Department Systems Sci. and Math., Washington University, St. Louis, USA

MSC2000: 49K15, 49K27, 41A10, 47N10

Article Outline

Keywords
Regularity in the Equality Constraint
Critical Directions
p-Order Local Maximum Principle
Conclusion
See also
References

Keywords

Local maximum principle; High-order tangent sets; High-order necessary conditions for optimality; Abnormal processes

We formulate a generalized local maximum principle which gives necessary conditions for optimality of abnormal trajectories in optimal control problems. The results are based on a hierarchy of primal constructions of high-order approximating cones (consisting of tangent directions for equality constraints, feasible directions for inequality constraints, and directions of decrease for the objective) and dual characterizations of empty intersection properties of these cones (see ► High-order necessary conditions for optimality for abnormal points). Characteristic for the theorem is that the multiplier associated with the objective is nonzero.

We consider an optimal control problem in Bolza form with fixed terminal time:

(OC) Minimize the functional

$$I(x,u) = \int_0^T L(x(t),u(t),t)\,dt + \ell(x(T)) \qquad (1)$$

subject to the constraints

$$\dot{x}(t) = f(x(t), u(t), t),$$
$$x(0) = 0, \qquad q(x(T)) = 0,$$
$$u(\cdot) \in \mathcal{U} = \{u \in L^r_\infty(0, T):\ u(t) \in U\}.$$

The terminal time T is fixed and we make the following *regularity assumptions* on the data: L: $\mathbf{R}^n \times \mathbf{R}^m \times [0, T] \to \mathbf{R}$ and f: $\mathbf{R}^n \times \mathbf{R}^m \times \to [0, T]$ $\mathbf{R}^n$ are C_∞ in (x, u) for every $t \in [0, T]$; both functions and their derivatives are measurable in t for every (x, u) and the functions and all partial derivatives are bounded on compact subsets of $\mathbf{R}^n \times \mathbf{R}^m \times [0, T]$; ℓ: $\mathbf{R}^n \to \mathbf{R}$ and q: $\mathbf{R}^n \to \mathbf{R}^k$ are C^∞ and the rows of the Jacobian matrix q_x (i. e. the gradients of the equations defining the terminal constraint) are linearly independent; $U \subset \mathbf{R}^m$ is a closed and convex set with nonempty interior. Let

$$H(\lambda_0, \lambda, x, u, t) = \lambda_0 L(x, u, t) + \lambda f(x, u, t) \qquad (2)$$

be the Hamiltonian for the control problem. If the input-trajectory pair (x_*, u_*) is optimal for problem (OC), then the *local maximum principle* [7] states that there exist a constant $\lambda_0 \geq 0$, an absolutely continuous function λ:$[0, T] \to (\mathbf{R}^n)^*$ (which we write as a row vector), which is a solution to the adjoint equation

$$\dot{\lambda} = -H_x(\lambda_0, \lambda(t), x_*(t), u_*(t), t),$$

with terminal condition

$$\lambda(T) = \lambda_0 \ell_x(x_*(T)) + \nu q_x(x_*(T)), \qquad (3)$$

(for some row vector $\nu \in (\mathbf{R}^k)^*$) such that $(\lambda_0, \lambda(t)) \neq 0$ for all $t \in [0, T]$ and the following local minimum condition holds for all $u \in U$:

$$\langle H_u(\lambda_0, \lambda(t), x_*(t), u_*(t), t), u - u_*(t)\rangle \geq 0. \qquad (4)$$

Input-trajectory pairs (x_*, u_*) for which multipliers λ_0 and λ exist such that these conditions are satisfied are called (*weak*) *extremals*. If $\lambda_0 > 0$, then it is possible to normalize $\lambda_0 = 1$ and the extremal is called *normal* while extremals with $\lambda_0 = 0$ are called *abnormal*. Although the terminology abnormal, which has its origins in the calculus of variations [4], seems to suggest that these type of extremals are an aberration, for optimal control problems this is not the case. The phenomenon is quite general and abnormal extremals cannot be excluded from optimality a priori. For instance, there exist optimal abnormal trajectories for the standard problem of stabilizing the harmonic oscillator time-optimally in minimum time, a simple time-invariant linear system.

In the abnormal case conventional necessary conditions for optimality provide conditions which only describe the structure of the constraints. For example, if there are no control constraints, then these conditions only involve the equality constraint defined by the dynamics and terminal conditions as zero set of an operator F: $Z \to Y$ between Banach spaces. If $F'(z_*)$ is not onto, but $\mathrm{Im}F'(z_*)$ is closed (and this is always the case for the optimal control problem) then the standard Lagrange multiplier type necessary conditions for optimality (which imply the local maximum principle [7]) can be satisfied trivially by choosing a multiplier which annihilates the image of $F'(z_*)$ and setting all other multipliers to zero.) The corresponding necessary conditions are independent of the objective and describe only the structure of the constraint yielding little information about the optimality of the abnormal trajectory.

Much of the difficulty in analyzing abnormal points in extremum problems can be traced back to the fact that the equality constraint is typically no longer a manifold near abnormal points, but intersections of manifolds are common. Hence, in order to develop necessary and/or sufficient conditions for optimality of abnormal extremals, it is imperative to analyze different branches of the zero-set of F. Finding these branches is at the heart of the matter. Generalizing a result of E.R. Avakov [2,3] in [10] a high-order generalization of the *classical Lyusternik theorem* is given which for general $p \in \mathbf{N}$ describes the structure of p-order tangent directions to an operator equality constraint in a Banach space for nonregular operators under a more general surjectivity assumption involving the first p derivatives of the operator. Based on these results p-order tangent cones to the equality constraint can explicitly be calculated along critical directions which comprise the low order terms. Combining these cones with standard constructions of high-order cones of decrease for the functional and high-order feasible cones to inequality constraints, all taken along critical directions, generalized necessary conditions for optimality for extremum problems in Banach spaces can be derived which allow to incorporate the objective with a nonzero mul-

tiplier. Characteristic of these results is that they are parametrized by critical directions as it is 'natural' near abnormal points.

In [12] (see ▶ High-order necessary conditions for optimality for abnormal points) an abstract formulation of these results is presented for minimization problems in Banach spaces. The main result gives a dual characterization for the empty intersection property of the various approximating cones along critical directions, but primal arguments using the cones themselves are often equally effective. In this article we formulate these abstract results for the optimal control problem, but we only consider the so-called weak or local version of the maximum principle. This result is weaker than the *Pontryagin maximum principle* [15] in the sense that the Pontryagin maximum principle asserts that the Hamiltonian of the control problem is indeed minimized over the control set at every time along the reference trajectory by the reference control. The local version only gives the necessary conditions for optimality for this property. However, it is well-known how to use an argument of A. Ya. Dubovitskii to derive the Pontryagin maximum principle from the local version [7, Lecture 13] and a preliminary strong version of the results of this article is given in [9].

Other theories of necessary conditions which are tailored to abnormal processes include a method known as 'weakening equality constraints' introduced in [14] and developed further in [5]. References [2,3] are along the lines of the results described here and give necessary conditions for optimality of abnormal extremals based on quadratic approximations. Similarly, both weak and strong versions of a second order generalized maximum principle are given by the authors in [8]. While mostly optimization related techniques are used in these papers, on a different level [1] uses differential geometric techniques to develop a theory of the second variation for abnormal extremals. They give both necessary and sufficient conditions for so-called corank-1 abnormal extremals (extremals for which there exists a unique multiplier) in terms of the Jacobi equation and related Morse indices and nullity theorems. Second order necessary conditions for optimality in the type of accessory problem results without normality assumptions have first been given in [6]. Also, the results in [16] are derived without making normality assumptions.

Regularity in the Equality Constraint

We model the optimal control problem (OC) in the framework of optimization theory as a minimization problem in a Banach space under equality and inequality constraints. Let $W_{11}^n(0,\,T)$ denote the Banach space of all absolutely continuous functions $x\colon [0,\,T]\to \mathbf{R}^n$ with norm $|x| = \|x(0)\| + \int_0^T \|\dot{x}(s)\|\ ds$ and let $\overline{W}_{11}^n(0,\,T) = W_{11}^n(0,\,T)\cap\{x\in W_{11}^n(0,\,T)\colon\ x(0)=0\}$. Then the problem is to minimize the functional I over a class $\mathcal{A}$ of input-trajectory pairs $(x,u)\in \overline{W}_{11}^n(0,\,T)\times L_\infty^m(0,\,T)$ which is defined by equality constraints and the convex inequality constraint $u\in\mathcal{U}$. The equality constraints can be modeled as $\mathcal{F} = \big\{(x,u)\in \overline{W}_{11}^n(0,\,T)\times L_\infty^m(0,\,T)\colon\ F(x,u)=0\big\}$ where F is the operator

$$F\colon\ \overline{W}_{11}^n(0,\,T)\times L_\infty^m(0,\,T)\to \overline{W}_{11}^n(0,\,T)\times \mathbf{R}^k$$

with $F(x,u)$ given by

$$\left(x(\cdot) - \int_0^{(\cdot)} f(x(s),u(s),s)\ ds,\quad q(x(T))\right).$$

It is easy to see that the operator F has continuous Fréchet derivatives of arbitrary order. For instance, $F'(x,u)$ acting on $(\eta,\xi)\in \overline{W}_{11}^n(0,\,T)\times L_\infty^m(0,\,T)$ is given by

$$\left(\eta(t) - \int_0^t f_x\eta + f_u\xi\ ds,\quad q_x(x(T))\eta(T)\right).$$

All partial derivatives of f are evaluated along a reference input-trajectory pair $(x,\,u)\in\mathcal{A}$. The formulas for higher order derivatives are given by equally straightforward multilinear forms.

We first describe the image of the operator $F'(x_*,\,u_*)$ for a reference input-trajectory pair $(x_*,\,u_*)$. Denote the fundamental matrix of the variational equation by $\Phi(t,s)$, i. e.

$$\frac{\partial}{\partial t}\Phi(t,s) = f_x(x(t),u(t),t)\Phi(t,s),$$
$$\Phi(s,s) = \mathrm{Id}.$$

Furthermore, let $\mathrm{R}\subset\mathbf{R}^n$ denote the reachable subspace of the linearized system

$$\dot{h}(t) = f_x h + f_u v,\quad h(0)=0, \tag{5}$$

at time T. It is well-known that R is a linear subspace of $\mathbf{R}^n$ and that $R = \mathbf{R}^n$ if and only if equation (5) is completely controllable. In general we have that

Lemma 1 *$ImF'(x_*, u_*)$ consists of all pairs $(a, b) \in \overline{W}^n_{11}(0, T) \times \mathbf{R}^k$ such that*

$$b \in q_x(x_*(T)) \left(\int_0^T \Phi(T, s)\dot{a}(s)\, ds + R \right). \tag{6}$$

In particular, $ImF'(x_*, u_*)$ is closed and of finite codimension.

The following characterizations of the nonregularity of the operator F and its codimension are well-known.

Proposition 2 *The codimension of $F'(x_*, u_*)$ is equal to the number of linearly independent solutions to $\dot{\lambda}(t) = -\lambda(t) f_x(x_*(t), u_*(t), t)$ which satisfy $\lambda(t) f_u(x_*(t), u_*(t), t) \equiv$ on [0, T] and for which $\lambda(T)$ is orthogonal to ker $q_x(x_*(T))$.*

Proposition 3 *The operator F is nonregular at $\Gamma = (x_*, u_*)$ if and only if Γ is an abnormal weak extremal which satisfies $H_u(0, \lambda(t), x_*(t), u_*(t), t) \equiv 0$ on [0, T].*

Critical Directions

We describe the set of critical directions along which high-order tangent approximations to the equality constraint $\mathcal{F}$ can be set up. Let $Z = \overline{W}^n_{11}(0, T) \times L^m_\infty(0, T)$ and suppose an admissible process $z_* = (x_*, u_*) \in \mathcal{A}$ and a finite sequence $H_{p-1} = (h_1, \ldots, h_p - 1) \in Z^p - 1$ are given. The following operators allow to formalize high-order approximations to an equality constraint at nonregular points (see, ▶ High-order necessary conditions for optimality for abnormal points). For $k = 1, \ldots, p - 1$, the *directional derivatives* $\nabla^k F(z_*)(H_k)$ of F at z_* along the sequence $H_k = (h_1, \ldots, h_k)$ are given by

$$\sum_{r=1}^{k} \frac{1}{r!} \left(\sum_{j_1 + \cdots + j_r = k} F^{(r)}(z_*)(h_{j_1}, \ldots, h_{j_r}) \right) \tag{7}$$

and we let $G_k[F](z_*;H_{k-1})$ denote the Fréchet-derivatives of the $(k-1)$th directional derivative of F at z_* along H_{k-1}. Thus formally $G_1[F](z_*) = F'(z_*)$ and in general for $k \geq 2$, $G_k = G_k[F](z_*;H_{k-1}) \colon Z \to Y$, $v \to G_k(v)$, is given by

$$G_k(v) = \sum_{r=1}^{k-1} \frac{1}{r!} \times \left(\sum_{j_1 + \cdots + j_r = k-1} F^{(r+1)}(z_*)(h_{j_1}, \ldots, h_{j_r}, v) \right). \tag{8}$$

We also denote by $R_q[F](z_*;H_\ell)$ those terms in the Taylor expansion of $F(z_* + \sum_{i=1}^p \varepsilon^i h_i)$ which are homogeneous of degree $q \geq 2$, but only involve vectors from H_ℓ. The general structure of these remainders is given by

$$\sum_{r=2}^{q} \frac{1}{r!} \left(\sum_{\substack{j_1 + \cdots + j_r = q, \\ 1 \leq j_k \leq \ell, \\ 1 \leq k \leq r}} F^{(r)}(z_*)(h_{j_1}, \ldots, h_{j_r}) \right). \tag{9}$$

Let

$$Y_i = \sum_{k=1}^{i} \operatorname{Im} G_k[F](z_*; H_{k-1}), \quad i = 1, \ldots, p. \tag{10}$$

The following conditions are necessary for the existence of a p-order tangent vector along H_{p-1} [10]:

i) the first $p - 1$ directional derivatives of F along H_{p-1} vanish,

$$\nabla^i F(z_*)(H_i) = 0, \forall i = 1, \ldots, p - 1;$$

ii) the *compatibility conditions*

$$R_{p-1+i}[F](z_*; H_{p-1}) \in Y_i, \quad i = 1, \ldots, p - 1,$$

are satisfied.

In these equations all partial derivatives of f are evaluated along the reference trajectory. These conditions are also sufficient if the operator F is *p-regular* at z_* in direction of the sequence H_{p-1} in the sense of the following definition.

Definition 4 Let $F \colon Z \to Y$ be an operator between Banach spaces. We say the operator F is *p-regular* at z_* in direction of the sequence $H_{p-1} \in Z^{p-1}$ if the following conditions are satisfied:

A1) $F\colon Z \to Y$ is $(2p-1)$-times continuously Fréchet differentiable in a neighborhood of z_*.
A2) The subspaces Y_i, $i = 1, \ldots, p$, are closed.
A3) The map $\mathcal{G}_p = \mathcal{G}_p[F](z_*; H_{p-1})$,

$$\mathcal{G}_p\colon\ Z \to Y_1 \times \frac{Y_2}{Y_1} \times \cdots \times \frac{Y}{Y_{p-1}}$$
$$v \mapsto \mathcal{G}_p(v) = \left(G_1(v), \pi_1 G_2(v), \ldots, \pi_{p-1} G_p(v)\right),$$

where $\pi_i\colon Y_{i+1} \to Y_{i+1}/Y_i$ denotes the canonical projection onto the quotient space, is onto.

In the sense of this definition 1-regularity corresponds to the classical Lyusternik condition while 2-regularity is similar to Avakov's definition [3]. Under these assumptions vectors h_p exist which extend H_{p-1} to p-order tangent vectors to $\mathcal{F}$ at z_* [10,12].

For the critical directions for the objective I we focus on the least degenerate critical case and therefore make the following assumption:

iii) $I'(z_*)$ is not identically zero and $\nabla^i I(z_*)(H_i) = 0$ for $i = 1, \ldots, p-1$.

The assumption that the first $p-1$ directional derivatives vanish is directly tied in with optimality. If there exists a first nonzero directional derivative $\nabla^j I(z_*)(H_j)$ with $j < i$ which is positive, then z_* indeed is a local minimum for any curve $z(\varepsilon) = z_* + \sum_{i=1}^{p} \varepsilon^i h_i + o(\varepsilon^p)$, $\varepsilon > 0$, and none of the directions H_{p-1} is of any use in improving the value. We restrict to $\varepsilon \geq 0$ since we also want to include inequality constraints. On the other hand, if $\nabla^j I(z_*)(H_j) < 0$, then H_j is indeed a direction of decrease and arbitrary high-order extensions of this sequence will give better values. Thus the reference trajectory is not optimal.

We also need to define the critical directions for the inequality constraint $\mathcal{U}$ in the optimal control problem. More generally, we define a p-order feasible set to an inequality constraint in a Banach space.

Definition 5 Let $S \subset Z$ be a subset with nonempty interior. We call v a p-order feasible vector for S at z_* in direction of $H_{p-1} = (h_1, \ldots, h_{p-1}) \in Z^{p-1}$ if there exist an $\varepsilon_0 > 0$ and a neighborhood V of v so that for all $0 < \varepsilon \leq \varepsilon_0$,

$$z_* + \sum_{i=1}^{p-1} \varepsilon^i h_i + \varepsilon^p V \subset S.$$

The collection of all p-order feasible vectors v for S at z_* in direction of the sequence H_{p-1} will be called the *p-order feasible set* to S at z_* in direction of the sequence H_{p-1} and will be denoted by $\mathrm{FS}^{(p)}(S; z_*, H_{p-1})$.

It follows from this definition that $\mathrm{FS}^{(p)}(S; z_*, H_{p-1})$ is open. It is also clear that $\mathrm{FS}^{(p)}(S; z_*, H_{p-1})$ is convex, if S is. Furthermore, if $h_j \in \mathrm{FS}^{(j)}(S; z_*, H_{j-1})$ for some integer $j < p$, then any vector v is allowed as a p-order feasible direction and thus trivially $\mathrm{FS}^{(p)}(S; z_*, H_p - 1) = X$.

For the optimal control problem and $H_{p-1} = ((\eta_1, \xi_1), \ldots, (\eta_{p-1}, \xi_{p-1}))$ let $V_{p-1} = (\xi_1, \ldots, \xi_{p-1}) \in L_\infty^m(0, T)^p$ denote the sequence of controls. Then the critical feasible directions for the convex inequality constraint $\mathcal{U}$ in $L_\infty^m(0, T)$ consist of all H_{p-1} for which

iv) $\mathrm{FS}^{(p)}(\mathcal{U}; u_*, V_{p-1})$ is nonempty.

Definition 6 We call a direction H_{p-1} a *p-regular critical direction* for the extremum problem at z_* if the operator F is p-regular at z_* along H_{p-1} and if conditions (i–iv) are satisfied.

p-Order Local Maximum Principle

Theorem 7 below gives a generalized p-order version of the maximum principle obtained from a dual characterization of the fact that if (x_*, u_*) is optimal, then the p-order tangent cones to the set $\{F = 0\}$, the p-order feasible cone to $\mathcal{U}$ and the p-order cone of decrease for the functional I cannot intersect. Notice that we write covectors like ψ as row vectors. This is consistent with a multiplier interpretation of the adjoint variable. Also we denote partial derivatives by subscripts. For instance, if $\nabla^i f(H_i)$ denotes the ith directional derivative of $f = f(x, u, t)$ with respect to the sequence H_i, then $(\nabla^i f(H_i))_x$ denotes its partial derivative in x. For example, suppose $H_1 = (\eta_1, \xi_1)$. Then

$$\nabla^1 f(H_1) = f_x(x, u, t)\eta_1 + f_u(x, u, t)\xi_1$$

and thus

$$\left(\nabla^1 f(H_1)\right)_x = f_{xx}(x, u, t)\eta_1 + f_{ux}(x, u, t)\xi_1$$

and

$$\left(\nabla^1 f(H_1)\right)_u = f_{xu}(x, u, t)\eta_1 + f_{uu}(x, u, t)\xi_1.$$

Theorem 7 (*p*-order local maximum principle) *Suppose the admissible process (x_*, u_*) is optimal for the optimal control problem (OC). Then for every p-regular critical direction H_{p-1} there exist a number $\nu_0 = \nu_0(H_{p-1}) \geq 0$, vectors $a_i = a(H_{p-1}) \in (\mathbf{R}^k)^*$, $i = 0, \ldots, p-1$, and absolutely continuous functions $\psi(\cdot) = \psi(H_{p-1})(\cdot)$ and $\rho_i(\cdot) = \rho_i(H_{p-1})(\cdot)$, $i = 1, \ldots, p-1$, from $[0, T]$ into $(\mathbf{R}^n)^*$, which satisfy the following conditions along the optimal trajectory $(x_*(t), u_*(t), t)$:*

a) nontriviality condition: ν_0 and the functional λ: $L^m_\infty(0, T) \to \mathbf{R}$, $\xi \longmapsto \lambda(\xi)$, given by

$$\int_0^T \left\langle \nu_0 L_u + \psi f_u + \sum_{i=1}^{p-1} \rho_i \left(\nabla^i f(H_i)\right)_u, \xi \right\rangle dt \quad (11)$$

do not both vanish identically.

b) extended adjoint equation

$$\dot\psi(t) = -\nu_0 L_x - \psi(t) f_x - \sum_{i=1}^{p-1} \rho_i(t) \left(\nabla^i f(H_i)\right)_x \quad (12)$$

with terminal condition

$$\psi(T) = \nu_0 \ell_x(x_*(T)) + a_0 q_x(x_*(T)) + \sum_{i=1}^{p-1} a_i \left(\nabla^i q(x_*(T); H_i)\right)_x. \quad (13)$$

c) orthogonality conditions on the additional multipliers: The functions $\rho_i(\cdot)$, $i = 1, \ldots, p-1$, satisfy

$$\begin{aligned} &\dot\rho_i(t) = -\rho_i(t) f_x, \quad \rho_i(t) f_u \equiv 0, \\ &\rho_i(T) = a_i q_x(x_*(T)) \end{aligned} \quad (14)$$

and for $j = 1, \ldots, i-1$, the following conditions are satisfied for a.e. $t \in [0, T]$:

$$\rho_i(t) \left(\nabla^j f(H_j)\right)_x = 0, \quad (15)$$

$$\rho_i(t) \left(\nabla^j f(H_j)\right)_u = 0, \quad (16)$$

$$a_i \left(\nabla^j q(x_*(1); H_j)\right)_x = 0; \quad (17)$$

d) separation condition: for all vectors $\xi \in FS^{(p)}(\mathcal{U}; u_*, V_{p-1})$ we have that

$$\begin{aligned} 0 \leq\ & \nu_0 R_p[\ell](H_{p-1}) + a_0 R_p[q](H_{p-1}) \\ &+ \sum_{i=1}^{p-1} a_i R_{p+i}[q](H_{p-1}) \\ &+ \int_0^T \left\langle \nu_0 L_u + \psi f_u + \sum_{i=1}^{p-1} \rho_i(t) \left(\nabla^i f(H_i)\right)_u, \xi \right\rangle dt \\ &+ \int_0^T \nu_0 R_p[L](H_{p-1}) + \psi(t) R_p[f](H_{p-1}) \\ &+ \sum_{i=1}^{p-1} \rho_i(t) R_{p+i}[f](H_{p-1})\, dt. \end{aligned} \quad (18)$$

Corollary 8 *The separation condition d) implies the following p-order local minimum condition: along $(x_*(t), u_*(t), t)$ we have for every $u \in U$ and a.e. $t \in [0, T]$:*

$$0 \leq \left\langle \nu_0 L_u + \psi(t) f_u + \sum_{i=1}^{p-1} \rho_i(t) \left(\nabla^i f(H_i)\right)_u,\ u - u_*(t) \right\rangle. \quad (19)$$

In the case of a Lagrangian minimization problem which has no control constraints, or more generally if the control takes values in the interior of the control set, the functional λ vanishes identically. In this case we can normalize $\nu_0 = 1$ and we obtain the following Corollary:

Corollary 9 (*p-order local maximum principle for Lagrangian problems*) *Consider the optimal control problem (OC) without control constraints ($U = \mathbf{R}^m$) and suppose the admissible process (x_*, u_*) is optimal. Then for every p-regular critical direction H_{p-1} there exist vectors $a_i = a(H_{p-1}) \in (\mathbf{R}^k)^*$, $i = 0, \ldots, p-1$, and absolutely continuous functions $\psi(\cdot) = \psi(H_{p-1})(\cdot)$ and $\rho_i(\cdot) = \rho_i(H_{p-1})(\cdot)$, $i = 1, \ldots, p-1$, from $[0, T]$ into $(\mathbf{R}^n)^*$, which satisfy the conditions b)–d) of Theorem 7 along the optimal trajectory $(x_*(t), u_*(t), t)$ for $\nu_0 = 1$. In particular, we thus have*

$$L_u + \psi(t) f_u + \sum_{i=1}^{p-1} \rho_i(t) \left(\nabla^i f(H_i)\right)_u \equiv 0.$$

Example 10 We illustrate Theorem 7 with an example. Consider the problem to minimize the functional $I(x, u)$ given by

$$\int_0^T \left[(x_1 - 1)^2 + x_2^p + (x_3 + 1)^2 - 2\right] dt \tag{20}$$

over all $(x, u) \in \overline{W}^3_{11}(0, T) \times L^2_\infty(0, T)$ subject to the dynamics

$$\dot{x}(t) = \begin{pmatrix} 0 \\ x_1^p \\ \alpha x_2^{p-1} x_3 \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \tag{21}$$

initial condition $x(0) = 0$ and terminal constraints $x_1(T) = 0$ and $x_3(T) = 0$. Here p is an integer, $p \geq 2$, and α is an arbitrary real number. For simplicity we have not imposed any control constraints.

It can easily be seen that the reference trajectory $\Gamma = (x_*, u_*) \equiv (0, 0)$ is an abnormal extremal for each problem. In fact, setting $\lambda(t) = (\nu, 0, \nu)$ with $\nu \neq 0$ and $\lambda_0 = 0$ defines an adjoint vector for Γ such that $H_u \equiv 0$. Hence $F'(0, 0)$ is nonregular.

Theorem 7 can be used to eliminate Γ from optimality for any $p \geq 2$. We choose H_{p-1} of the form

$$H_{p-1} = ((\eta_1, \xi_1); (0, 0); \cdots; (0, 0)) \tag{22}$$

with $(\eta_1, \xi_1) \in F'(0, 0)$. With this choice of directions the compatibility conditions ii) simplify considerably and reduce to the first condition only which becomes

$$\int_0^T \left(\eta_1^{[2]}\right)^{p-1} \left(\eta_1^{[3]}\right) ds = 0.$$

Here the superscripts denote the components of the vector η_1. We satisfy this by choosing $\eta_1^{[3]} = -\eta_1^{[1]} \equiv 0$ (i. e., $\xi_1^{[2]} \equiv 0$). Then choosing a nonzero $\eta_1^{[2]}$ with zero boundary conditions defines a nontrivial vector H_{p-1} of the form (22) for which conditions i) and ii) in the definition of p-regular critical directions are satisfied. Furthermore, it is easily seen that the operator F is p-regular in direction of H_{p-1} at Γ. Finally, these directions are also critical for the objective: we have $I'(0, 0)(\eta_1, \xi_1) = 0$ and furthermore

$$\begin{aligned} \nabla^2 I(0, 0)(H_2) &= \frac{1}{2} I''(0, 0)((\eta_1, \xi_1); (\eta_1, \xi_1)) \\ &= \int_0^T \left(\eta_1^{[1]}\right)^2 + \left(\eta_1^{[3]}\right)^2 ds = 0 \end{aligned}$$

provided $p > 2$. Since no other I-derivatives arise in the directional derivatives $\nabla^i I(0, 0)(H_i)$ for $i = 3, \ldots, p - 1$, the direction $H_p - 1 = ((\eta_1, \xi_1); (0, 0); \cdots; (0, 0))$ with $\eta_1^{[1]} = \eta_1^{[3]} \equiv 0$ and a nonzero $\eta_1^{[2]}$ is a nonzero p-regular critical direction for the problem to minimize I subject to $F = 0$ for any $p \geq 2$.

We thus can apply Theorem 7. Since there are no control constraints we can normalize the multipliers so that $\nu_0 = 1$. The additional multipliers ρ_i, $i = 1, \ldots, p - 1$, are associated with elements in the dual spaces of the quotients Y_{i+1}/Y_i (see ▶ High-order necessary conditions for optimality for abnormal points). But here $Y_i = \mathrm{Im}\, F'(0, 0)$ for $i = 1, \ldots, p - 1$, and Y_p is the full space. Thus we have $\rho_i \equiv 0$ for $i = 2, \ldots, p - 1$ and the only nonzero multipliers are ψ and ρ_{p-1} which for simplicity of notation we just call ρ. Now (14) states that ρ is an adjoint multiplier for which the conditions of the local Maximum Principle for an abnormal extremal are satisfied. This multiplier is unique and of the form $\rho(t) = (\nu, 0, \nu)$, but $\nu \in \mathbf{R}$ could be zero. For the extended adjoint equation and minimum condition (19) we need to evaluate the directional derivatives $\nabla^{p-1} f(x, u)(H_i)$. Straightforward, but a bit tedious calculations show that

$$\left(\nabla^{p-1} f(0, 0)(H_i)\right)_x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \left(\eta_1^{[2]}\right)^{p-1} \end{pmatrix}$$

and

$$\left(\nabla^{p-1} f(0, 0)(H_i)\right)_u \equiv 0.$$

Thus the extended minimum condition reduces to $\psi B \equiv 0$, the minimum condition of the weak maximum principle. Hence also $\psi_2(t) \equiv 0$ and $\psi_1(t) = \psi_3(t)$. But now the extended adjoint equation is given by

$$\dot{\psi}(t) = (2, 0, -2) - \rho \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \left(\eta_1^{[2]}\right)^{p-1} \end{pmatrix}$$

and thus

$$4 = \dot{\psi}_1(t) - \dot{\psi}_3(t) - \nu\left(\eta_1^{[2]}(t)\right)^{p-1} = -\nu\left(\eta_1^{[2]}(t)\right)^{p-1}.$$

But we can certainly choose $\eta_1^{[2]}$ nonconstant to violate this condition. This contradiction proves that Γ cannot be optimal for the problem to minimize I for any $p \geq 2$.

Conclusion

Theorem 7 is based on p-order approximations. If these remain inconclusive, higher order approximations can easily be set up. If the operator F is p-regular in direction of H_{p-1}, then given a p-regular tangent direction, it is possible to set up higher order approximations of arbitrary order. In fact, only a system of p linear equations needs to be solved in every step. These results provide a complete hierarchy of primal constructions of higher-order approximating directions and dual characterizations of empty intersection properties of approximating cones which can be used to give necessary conditions for optimality for increasingly more degenerate structures. For these results see [13].

See also

- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Pontryagin Maximum Principle

References

1. Agrachev AA, Sarychev AV (1995) On abnormal extremals for Lagrange variational problems. J Math Syst, Estimation and Control 5:127–130
2. Avakov ER (1988) Necessary conditions for a minimum for nonregular problems in Banach spaces. Maximum principle for abnormal problems of optimal control. Trudy Mat Inst Akad Nauk SSSR 185:3–29 (In Russian.)
3. Avakov ER (1989) Necessary extremum conditions for smooth anormal problems with equality-and inequality constraints. J Soviet Math 45. Matematicheskie Zametki 45:3–11
4. Caratheodory C (1935) Variationsrechnung und partielle Differentialgleichungen erster Ordnung. Teubner, Leipzig
5. Dmitruk AV (1998) Quadratic order conditions of a local minimum for abnormal extremals. In: Proc. 2nd World Congress of Nonlinear Analysts, Part 4, Athens 1996. Nonlinear Anal 30:2439–2448
6. Gilbert EG, Bernstein DS (1983) Second-order necessary conditions in optimal control: accessory-problem results without normality conditions. J Optim Th Appl 41:75–106
7. Girsanov IV (1972) Lectures on mathematical theory of extremum problems. Springer, Berlin
8. Ledzewicz U, Schättler H (1997) An extended maximum principle. Nonlinear Anal 29:59–183
9. Ledzewicz U, Schättler H (1998) High order extended maximum principles for optimal control problems with nonregular constraints. In: Hager WW, Pardalos PM (eds) Optimal Control: Theory, Algorithms and Applications. Kluwer, Dordrecht, pp 298–325
10. Ledzewicz U, Schättler H (1998) A high-order generalization of the Lyusternik theorem. Nonlinear Anal 34:793–815
11. Ledzewicz U, Schättler H (1998) A high-order generalization of the Lyusternik theorem and its application to optimal control problems. In: Chen W, Hu S (eds) Dynamical Systems and Differential Equations II. pp 45–59
12. Ledzewicz U, Schättler H (1999) High-order approximations and generalized necessary conditions for optimality. SIAM J Control Optim 37:33–53
13. Ledzewicz U, Schättler H (2000) A high-order generalized local maximum principle. SIAM J Control Optim 38:823–854
14. Milyutin AA (1981) Quadratic conditions of an extremum in smooth problems with a finite-dimensional image. Methods of the Theory of Extremal Problems in Economics. Nauka Moscow, Moscow, pp 138–177 (In Russian.)
15. Pontryagin LS, Boltyanskii VG, Gamkrelidze RV, Mishchenko EF (1962) The mathematical theory of optimal processes. Wiley, New York
16. Stefani G, Zezza PL (1996) Optimality conditions for a constrained control problem. SIAM J Control Optim 34:635–659

High-order Necessary Conditions for Optimality for Abnormal Points

Urszula Ledzewicz[1], Heinz Schättler[2]

[1] Department Math. and Statist., Southern Illinois University at Edwardsville, Edwardsville, USA

[2] Department Systems Sci. and Math., Washington University, St. Louis, USA

MSC2000: 49K27, 46N10, 41A10, 47N10

Article Outline

Keywords
A High-Order Formulation
of the Dubovitskii–Milyutin Theorem
High-Order Directional Derivatives
High-Order Tangent Cones
High-Order Cones of Decrease
High-Order Feasible Cones to Inequality Constraints
Given by Smooth Functionals
High-Order Feasible Cones
to Closed Convex Inequality Constraints
Generalized Necessary Conditions for Optimality
See also
References

Keywords

Lyusternik theorem; High-order tangent sets; High-order necessary conditions for optimality; Abnormal processes

We consider the problem of minimizing a functional I: $X \to \mathbf{R}$ in a Banach space X under both *equality* and *inequality constraints*. The inequality constraints are of two types, either described by smooth functionals f: $X \to \mathbf{R}$ as $P = \{x \in X : f(x) \leq 0\}$ or described by closed convex sets C with nonempty interior. The equality constraints are given in operator form as $Q = \{x \in X : F(x) = 0\}$ where F: $X \to Y$ is an operator between Banach spaces. Models of this type are common in optimal control problems.

The standard first order Lagrange multiplier type necessary conditions for optimality at the point x_* state that there exist multipliers $\lambda_0, \ldots, \lambda_m, y^*$ which do not all vanish identically such that the *Euler–Lagrange equation*

$$\lambda_0 I'(x_*) + \sum_{j=1}^{m} \lambda_j f_j'(x_*) + F'^*(x_*)y^* = 0, \qquad (1)$$

is satisfied (see for instance [7,9]). This article addresses the case when the Fréchet-derivative $F'(x_*)$ of the operator defining the equality constraint is not onto, i. e. the regular case. In this case the *classical Lyusternik theorem* [14] does not apply to describe the tangent space to Q and (1) can be satisfied trivially by choosing a nonzero multiplier y^* from the annihilator of Im $F'(x_*)$ while setting all other multipliers zero. This generates so-called *abnormal points* for which the standard necessary conditions for optimality only describe the degeneration of the equality constraint without any relation to optimality. Here we describe an approach to high-order necessary conditions for optimality in these cases which is based a *high-order generalization of the Lyusternik theorem* [12]. By using this theorem one can determine the precise structure of polynomial approximations to Q at x^* when the surjectivity condition on $F'(x_*)$ is not satisfied, but when instead a certain operator G_p which takes into account all derivatives up to and including order p is onto. The order p is chosen as the minimum number for which the operator G_p becomes onto. If G_p is onto, then the precise structure of q-order polynomial approximations to Q at x_* for any $q \geq p$ can be determined. This leads to the notion of high-order tangent cones to the equality constraint Q at points x_* in a nonregular case. Combining these with high-order feasible cones for the inequality constraints and high-order cones of decrease, a generalization of the *Dubovitskii–Milyutin theorem* is formulated. From this theorem generalized necessary conditions for optimality can be deduced which reduce to classical conditions for normal cases, but give new and nontrivial conditions for abnormal cases.

First results of this type have been obtained for quadratic approximations ($p = 2$) in [3,4,5] and [11]. Some of these conditions have been analyzed further also in connection with sufficient conditions for optimality, [1,2]. In [10] also quadratic approximations for problems with inequality constraints are considered. For the regular case when $F'(x_*)$ is onto second order approximating sets were introduced in [6] to derive second order necessary conditions for optimality, while higher order necessary conditions for optimality in this case are given, for instance, in [8] or [15]. These, however, are not the topic of this article.

A High-Order Formulation of the Dubovitskii–Milyutin Theorem

Let X and Y be Banach spaces. Let I: $X \to \mathbf{R}$ be a functional, F: $X \to Y$ an operator, f_j: $X \to \mathbf{R}$, $j = 1, \ldots, m$, functionals and let $C \subset X$ be a closed convex set with nonempty interior. We assume that I, the functionals f_j

and the operator F are sufficiently often continuously Fréchet-differentiable and consider the problem

$$(P)\begin{cases}\min\limits_{x} & I\\ \text{s.t.} & x\in A=\left(\cap_{j=1}^{m}P_j\right)\cap Q\cap C,\\ & P_j=\left\{x\in X\colon\ f_j(x)\le 0\right\}\\ & Q=\left\{x\in X\colon\ F(x)=0\right\}.\end{cases}$$

We define high-order polynomial approximations to the admissible domain A. We denote sequences $(h_1, \ldots, h_k)\in X^k$ by H_k with the subscript giving the length of the sequence.

Definition 1 Let $H_{p-1}=(h_1,\ldots,h_{p-1})\in X^{p-1}$ and set $x(\varepsilon)\doteq x_*+\sum_{i=1}^{p-1}\varepsilon^i h_i$. We call H_{p-1} a $(p-1)$-*order approximating sequence* to a set $S\subseteq X$ at $x_*\in \operatorname{Clos} S$, respectively we call $x{:}\varepsilon\to x(\varepsilon)$, a $(p-1)$-*order approximating curve*, if there exist an $\varepsilon_0>0$ and a function r defined on $[0,\varepsilon_0]$ with values in X, $r\colon [0,\varepsilon_0]\to X$, with the property that

$$x(\varepsilon)+r(\varepsilon)=x_*+\sum_{i=1}^{p-1}\varepsilon^i h_i+r(\varepsilon)\in S \tag{2}$$

and

$$\lim_{\varepsilon\to 0}\frac{\|r(\varepsilon)\|}{\varepsilon^{p-1}}=0. \tag{3}$$

We call a $(p-1)$-order approximating sequence/curve $(p-1)$-*order feasible* if S is an inequality constraint, respectively $(p-1)$-*order tangent* if S is an equality constraint.

Let $x_*\in F$ and assume as given a $(p-1)$-order approximating sequence $H_{p-1}=(h_1,\ldots,h_{p-1})\in X^{p-1}$ with corresponding $(p-1)$-order approximation $x(\varepsilon)\doteq x_*+\sum_{i=1}^{p-1}\varepsilon^i h_i$. It is implicitly assumed that x_* has not been ruled out for optimality. Then we extend the existing $(p-1)$-order approximations to p-order approximations and derive the corresponding necessary conditions for optimality. The following definitions are direct generalizations of standard existing definitions [7].

Definition 2 We call v_0 a *p-order vector of decrease* for a functional $I\colon X\to \mathbf{R}$ at $x_*\in X$ in direction of the sequence $H_{p-1}=(h_1,\ldots,h_{p-1})\in X^{p-1}$ if there exist a neighborhood V of v_0 and a number $\alpha<0$ so that for all $v\in V$ we have

$$I\left(x_*+\sum_{i=1}^{p-1}\varepsilon^i h_i+\varepsilon^p v\right)=I(x(\varepsilon)+\varepsilon^p v)\le I(x_*)+\alpha\varepsilon^p. \tag{4}$$

The collection of all p-order vectors of decrease for I at x_* in direction of the sequence H_{p-1} will be called the *p-order set of decrease* to I at x_* in direction of the sequence H_{p-1} and will be denoted by $\mathrm{DS}^{(p)}(I;x_*,H_{p-1})$.

Definition 3 We call v_0 a *p-order feasible vector* for an inequality constraint P at $x_*\in X$ in direction of H_{p-1} if there exist an $\varepsilon_0>0$ and a neighborhood V of v_0 so that for all $0<\varepsilon\le\varepsilon_0$

$$x_*+\sum_{i=1}^{p-1}\varepsilon^i h_i+\varepsilon^p V=x(\varepsilon)+\varepsilon^p V\subset P. \tag{5}$$

The collection of all p-order feasible vectors v_0 for P at x_* in direction of the sequence H_{p-1} will be called the *p-order feasible set* to P at x_* in direction of the sequence H_{p-1} and will be denoted by $\mathrm{FS}^{(p)}(P;x_*,H_{p-1})$.

Note that by definition the p-order set of decrease to I and the p-order feasible set to P, both at x_* in direction of the sequence H_{p-1}, are open.

Definition 4 We call h_p a *p-order tangent vector* to an equality constraint Q at x_* in direction of the sequence H_{p-1} if $H_p=(h_1,\ldots,h_p)\in X^p$ is a p-order approximating sequence to the set Q at $x_*\in Q$. The collection of all p-order tangent vectors to Q at x_* in direction of the sequence H_{p-1} will be called the *p-order tangent set* to Q at x_* in direction of the sequence H_{p-1} and will be denoted by $\mathrm{TS}^{(p)}(Q;x_*,H_{p-1})$.

These approximating sets can be embedded into cones in the extended state-space $X\times\mathbf{R}$. This has the advantage that many classical results like the Minkowski–Farkas lemma or the annihilator lemma can be directly applied in calculating dual cones (see also [11]). Let us generally refer to p-order sets of decrease, feasible sets and tangent sets as p-order approximating sets and denote them by $\mathrm{AS}^{(p)}(Z;x_*,H_{p-1})$. Then we define the corresponding approximating cones as follows:

Definition 5 Given a p-order approximating set $\mathrm{AS}^{(p)}(Z;x_*,H_{p-1})$ to a set $Z\subset X$ at x_* in direction

of the sequence H_{p-1}, the *p-order approximating cone* to Z at x_* in direction of H_{p-1}, $\mathrm{AC}^{(p)}(Z;x_*, H_{p-1})$, is the cone in $X \times \mathbf{R}$ generated by the vectors $(v, 1) \in \mathrm{AS}^{(p)}(Z;x_*, H_{p-1}) \times \mathbf{R}$.

Thus we talk of the *p-order cone of decrease* for the functional I, *p-order feasible cones* for inequality constraints and *p-order tangent cones* for equality constraints, all at x_* in direction of the sequence H_{p-1}.

Definition 6 Let $C \subseteq Z$ be a cone in a Banach space Z with apex at 0. The *dual* (or *polar*) cone to C consists of all continuous linear functionals $\lambda \in Z^*$ which are nonnegative on C, i. e.

$$C^* = \{\lambda \in Z^* : \ \langle \lambda, v\rangle \geq 0, \ \forall v \in C\}. \tag{6}$$

Then we have

Theorem 7 [11,13] *(p-order Dubovitskii–Milyutin theorem). Suppose the functional I attains a local minimum for problem (P) at $x_* \in A$. Let $H_{p-1} = (h_1, \ldots, h_{p-1}) \in X^{p-1}$ be a $(p-1)$-order approximating sequence such that the p-order cone of decrease for the functional I, the p-order feasible cones for the inequality constraints P_j, $j = 1, \ldots, m$, and C, and the p-order tangent cone to the equality constraint Q, all at x_* in direction of the sequence H_{p-1}, are nonempty and convex. Then there exist continuous linear functionals*

$$\Psi_0 = (\lambda_0, \mu_0) \in \left(\mathrm{DC}^{(p)}(I; x_*, H_{p-1})\right)^*,$$
$$\Psi_j = (\lambda_j, \mu_j) \in \left(\mathrm{FC}^{(p)}(f_j; x_*, H_{p-1})\right)^*,$$

for $j = 1, \ldots, m$,

$$\Omega = (\lambda_{m+1}, \mu_{m+1}) \in \left(\mathrm{FC}^{(p)}(C; x_*, H_{p-1})\right)^*$$

and

$$\Phi = (\lambda_{m+2}, \mu_{m+2}) \in \left(\mathrm{TC}^{(p)}(Q; x_*, H_{p-1})\right)^*,$$

all depending on H_{p-1}, such that

$$\sum_{j=0}^{m+2} \lambda_j \equiv 0, \qquad \sum_{j=0}^{m+2} \mu_j \equiv 0 \tag{7}$$

hold. Furthermore, not all the λ_j, $j = 0, \ldots, m+2$, vanish identically.

High-Order Directional Derivatives

We describe a formalism to calculate higher derivatives [12,13] which will be needed to describe high-order approximating cones. Let $F: X \to Y$ be an operator between Banach spaces which is sufficiently often continuously Fréchet differentiable in a neighborhood of $x_* \in X$ and consider the Taylor expansion of F along a curve

$$\gamma(\varepsilon) = x_* + \sum_{i=1}^{m} \varepsilon^i h_i.$$

We have

$$F(\gamma(\varepsilon)) = F(x_*) + \sum_{i=1}^{m} \varepsilon^i \nabla^i F(x_*)(h_1, \ldots, h_i) + \widetilde{r}(\varepsilon),$$

where $\nabla^i F(x_*)(h_1, \ldots, h_i)$ is given by

$$\sum_{r=1}^{i} \frac{1}{r!} \left(\sum_{j_1 + \cdots + j_r = i} F^{(r)}(x_*)(h_{j_1}, \ldots, h_{j_r}) \right) \tag{8}$$

and $\widetilde{r}(\varepsilon)$ is a function of order $o(\varepsilon^m)$ as $\varepsilon \to 0$. Note that $\nabla^i F(x_*)(h_1, \ldots, h_i)$ simply collects the ε^i-terms in this expansion. These terms, which we call the *ith-order directional derivatives of F along the sequence* $H_i = (h_1, \ldots, h_i)$, $1 \leq i \leq m$, are easily calculated by straightforward Taylor expansions. For example,

$$\nabla^1 F(x_*)(H_1) = F'(x_*)h_1,$$
$$\nabla^2 F(x_*)(H_2) = F'(x_*)h_2 + \frac{1}{2}F''(x_*)(h_1, h_1).$$

The higher-order directional derivative $\nabla^i F(x_*)$ is homogeneous of degree i in the directions in the sense that

$$\nabla^i F(x_*)(\varepsilon h_1, \ldots, \varepsilon^i h_i) = \varepsilon^i \nabla^i F(x_*)(h_1, \ldots, h_i).$$

In particular, no indices j_1 and j_2 with $j_1 + j_2 > i$ can occur together as arguments in any of the terms in $\nabla^i F(x_*)$. Thus all vectors h_j whose index satisfies $2j > i$ appear linearly in $\nabla^i F(x_*)$ and are multiplied by terms which are homogeneous of degree $i - j$. In fact, there exist linear operators $G_k = G_k[F](x_*;H_{k-1})$, $k \in \mathbf{N}$, depending on the derivatives up to order k of F in the point x_* and on the vectors $H_{k-1} = (h_1, \ldots, h_{k-1})$, which describe the contributions of these components. We have $G_1[F](x_*) = F'(x_*)$ and in general

$G_k = G_k[F](x_*;H_{k-1})\colon Z \to Y$, $v \to G_k(v)$, is given by

$$G_k(v) = \sum_{r=1}^{k-1} \frac{1}{r!} \times \left(\sum_{j_1+\cdots+j_r=k-1} F^{(r+1)}(x_*)(h_{j_1},\ldots,h_{j_r},v) \right). \tag{9}$$

These operators $G_k[F](x_*;H_{k-1})$ are the Fréchet-derivatives of the $(k-1)$th directional derivative of F at x_* along H_{k-1}. Note that these terms are homogeneous of degree $k-1$. For simplicity of notation we often suppress the arguments. For example, we write

$$G_1(v) = F'(x_*)v, \quad G_2(v) = F''(x_*)(h_1,v),$$
$$G_3(v) = F''(x_*)(h_2,v) + \frac{1}{2}F'''(x_*)(h_1,h_1,v).$$

Given an order $p \in \mathbf{N}$, it follows that we can separate the linear contributions of the vectors $h_p,\ldots,h_{2p-1}$ in derivatives of orders p through $2p-1$ and for $i=1,\ldots,p$, we have an expression of the form

$$\nabla^{p-1+i}F(x_*)(H_{p-1+i}) = \sum_{k=1}^{i} G_k[F](x_*;H_{k-1})h_{p+i-k} + R_{p-1+i}[F](x_*;H_{p-1}).$$

Here among the terms which are homogeneous of degree $p-1+i$ the sum gives the terms which contain one of the vectors $h_p,\ldots,h_{p-1+i}$, and the remainder R combines all other terms which only include vectors of index $\le p-1$. The general structure of the remainder $R_q[F](z_*;H_\ell)$ for arbitrary $q \ge 2$ and ℓ is given by

$$\sum_{r=2}^{q} \frac{1}{r!} \left(\sum_{\substack{j_1+\cdots+j_r=q,\\ 1\le j_k\le \ell,\\ 1\le k\le r}} F^{(r)}(x_*)(h_{j_1},\ldots,h_{j_r}) \right). \tag{10}$$

Thus $R_q(H_\ell)$ consists of the terms which are homogeneous of degree q, but only involve vectors from H_ℓ. For example, $R_3[F](z_*;H_2)$ is given by

$$F''(z_*)(h_1,h_2) + \frac{1}{6}F^{(3)}(z_*)(h_1,h_1,h_1).$$

Note that the remainders only have contributions from derivatives of at least order two. These operators allow to formalize high-order approximations to an equality constraint at nonregular points [13].

High-Order Tangent Cones

We first describe the set of critical directions along which high-order tangent approximations to the equality constraint Q can be set up. For a given admissible process $z_* \in A$ and a finite sequence $H_{p-1} = (h_1,\ldots,h_{p-1}) \in X^{p-1}$, let

$$Y_i = \sum_{k=1}^{i} \operatorname{Im} G_k[F](x_*;H_{k-1}), \quad i = 1,\ldots,p.$$

It is clear that the first $p-1$ directional derivatives of F along H_{p-1} must vanish,

$$\nabla^i F(z_*)(H_i) = 0, \quad \forall\, i = 1,\ldots,p-1, \tag{11}$$

if H_{p-1} is a $(p-1)$-order tangent direction. But additional compatibility conditions of the form

$$R_{p-1+i}[F](x_*;H_{p-1}) \in Y_i, \quad i = 1,\ldots,p-1, \tag{12}$$

are necessary as well if we want to extend H_{p-1} to a p-order tangent direction $H_p = (H_{p-1};h_p)$. Conditions (11) and (12) are indeed sufficient for the existence of p-order approximations along H_{p-1} under the following regularity condition:

Definition 8 Let $F\colon X \to Y$ be an operator between Banach spaces. We say the operator F is *p-regular* at x_* in direction of the sequence $H_{p-1} \in X^{p-1}$ if the following conditions are satisfied:

A1) $F\colon X \to Y$ is $(2p-1)$-times continuously Fréchet differentiable in a neighborhood of x_*;

A2) the subspaces Y_i, $i=1,\ldots,p$, are closed;

A3) the map $\mathcal{G}_p = \mathcal{G}_p[F](x_*;H_{p-1})$

$$\mathcal{G}_p\colon X \to Y_1 \times \frac{Y_2}{Y_1} \times \cdots \times \frac{Y}{Y_{p-1}},$$
$$v \mapsto \mathcal{G}_p(v) = (G_1(v),\ldots,\pi_{p-1}G_p(v)),$$

where $\pi_i\colon Y_{i+1} \to Y_{i+1}/Y_i$ denotes the canonical projection onto the quotient space, is onto.

In the sense of this Definition, 1-regularity corresponds to the classical Lyusternik condition while 2-regularity is similar to Avakov's definition [5].

Theorem 9 [12] *Let H_{p-1} be a sequence so that $\nabla^i F(x_*)(H_i) = 0$ for $i = 1,\ldots,p-1$, and suppose the operator F is p-regular at x_* in direction of H_{p-1}. Then*

$TS^{(p)}(Q;x_*, H_{p-1})$ is nonempty if and only if for $i = 1, \ldots, p-1$, the compatibility conditions

$$R_{p-1+i}[F](x_*; H_{p-1}) \in Y_i$$

are satisfied. In this case $TS^{(p)}(Q;x_*, H_{p-1})$ is the closed affine subspace of X given by the solutions to the linear equation

$$\mathcal{G}_p[F](x_*; H_{p-1})(v) + \mathcal{R}_{p-1}[F](x_*, H_{p-1}) = 0, \quad (13)$$

where $\mathcal{R}_{p-1}[F](x_*, H_{p-1}) \in Z$ is the point with components

$$\big(R_p[F](x_*; H_{p-1}), \pi_1 R_{p+1}[F](x_*; H_{p-1}), \ldots, \pi_{p-1} R_{2p-1}[F](x_*; H_{p-1})\big).$$

This formulation of the result clearly brings out the geometric structure of the p-order tangent sets as closed affine linear subspaces of X generated by the kernel of $\mathcal{G}_p$, $\ker\mathcal{G}_p$.

Corollary 10 [12] *Let H_{p-1} be a sequence such that the operator F is p-regular at x_* in direction of H_{p-1}. Suppose the first $(p-1)$ directional derivatives $\nabla^i F(x_*)(H_i)$ vanish for $i = 1, \ldots, p-1$, and the compatibility conditions $R_{p-1+i}[F](x_*;H_{p-1}) \in Y_i$ are satisfied for $i = 1, \ldots, p$. Then the p-order tangent cone to $Q = \{x \in X: F(x) = F(x_*)\}$ at x_* in direction of H_{p-1}, $TC^{(p)}(Q;x_*, H_{p-1})$, consists of all solutions $(w, \gamma) \in X \times \mathbf{R}_+$ (i. e. $\gamma > 0$) of the linear equation*

$$\mathcal{G}_p[F](w) + \gamma \mathcal{R}_{p-1}[F](x_*, H_{p-1}) = 0.$$

For applications to optimization problems we need the subspace of continuous linear functionals which annihilate $\mathcal{G}_p$. Since the operator $\mathcal{G}_p$ is onto, it follows by the annihilator lemma or the closed-range theorem [9] that

$$(\ker \mathcal{G}_p)^\perp = \mathrm{Im}(\mathcal{G}_p^*),$$

where $\mathcal{G}_p^*$:

$$Z^* = Y_1^* \times (\frac{Y_2}{Y_1})^* \times \cdots \times (\frac{Y}{Y_{p-1}})^* \to X^*,$$

denotes the adjoint map. Let

$$\tau_i : (\frac{Y_{i+1}}{Y_i})^* \to Y_i^{\perp_{i+1}}$$

denote the canonical isomorphism. Here $\perp_{i+1}$ denotes the annihilator in Y_{i+1}, i. e.

$$Y_i^{\perp_{i+1}} = \{y^* \in Y_{i+1}^* : \ \langle y^*, v\rangle = 0, \ \forall v \in Y_i\}$$

and we formally set $Y_0 = \{0\}$, so that $Y_0^{\perp_1} \cong Y_1^*$. Then we have:

Proposition 11 [11,13] *A functional $\lambda \in X^*$ lies in $(\ker\mathcal{G}_p)^\perp$ if and only if it can be represented in the form*

$$\lambda = \sum_{i=1}^{p} G_i^*[F](x_*; H_{i-1}) y_i^* \quad (14)$$

for some functionals $y_i^* \in Y_{i-1}^{\perp_i}$, $i = 1, \ldots, p$.

Proposition 12 [11,13] *The dual or polar p-order tangent cone consists of all linear functionals $(\lambda, \mu) \in X^* \times \mathbf{R}$ which can be represented in the following form: There exist functionals $y_i^* \in Y_{i-1}^{\perp_i}$, $i = 1, \ldots, p$, and a number $r \geq 0$ such that*

$$\lambda = \sum_{i=1}^{p} G_i^*[F](x_*; H_{i-1}) y_i^*,$$

$$\mu = \sum_{i=1}^{p} \langle y_i^*, R_{p-1+i}[F](x_*; H_{p-1})\rangle + r.$$

High-Order Cones of Decrease

We now consider critical directions for the objective I and determine the p-order sets of decrease of a functional $I: X \to \mathbf{R}$. These results also apply to p-order feasible sets to inequality constraints defined by smooth functionals. We assume as given a $(p-1)$-order sequence H_{p-1} and we calculate the p-order set of decrease of I at x_* along H_{p-1}. Trivial cases arise if there exists a first nonzero directional derivative $\nabla^i I(x_*)(H_i)$ of I with $i \leq p-1$. In this case we have either $DS^{(p)}(I;x_*, H_{p-1}) = \emptyset$ if $\nabla^i I(x_*)(H_i) > 0$ or $DS^{(p)}(I;x_*, H_{p-1}) = X$ if $\nabla^i I(x_*)(H_i) < 0$. In the first case the sequence H_{p-1} cannot be used to exclude optimality of x_* since indeed x_* is a local minimum along the approximating curve generated by H_{p-1}. In the second case h_i is an ith-order direction of decrease along H_{i-1} and thus every vector $v \in X$ is admissible as a pth order component. The only nontrivial case arises

if $\nabla^i I(x_*)(H_i) = 0$ for all i with $i \le p-1$ and if $I'(x_*) \neq 0$.

Proposition 13 [13] *Suppose $I'(x_*) \neq 0$ and for all i with $i \le p-1$ we have $\nabla^i I(x_*)(H_i) = 0$. Then the p-order cone of decrease for the functional I at x_* in direction of H_{p-1}, $DC^{(p)}(I;x_*, H_{p-1})$, consists of all vectors $(w, \gamma) \in X \times \mathbf{R}$ which satisfy*

$$I'(x_*)w + \gamma R_p[I](x_*; H_{p-1}) < 0.$$

Thus $DC^{(p)}(I;x_*, H_{p-1})$ is nonempty, open and convex. The dual or polar cone to $DC^{(p)}(I;x_*, H_{p-1})$ can easily be calculated using the Minkowski–Farkas lemma [7].

High-Order Feasible Cones to Inequality Constraints Given by Smooth Functionals

In this section we give the form of the p-order feasible cones, $FC^{(p)}(P;x_*, H_{p-1})$, for inequality constraints P described by smooth functionals,

$$P = \{x \in X:\ f(x) \le 0\}.$$

Similar like for sets of decrease, if there exists a first index $i \le p-1$ such that $\nabla^i f(x_*)(H_i) \neq 0$, then the constraint will either be satisfied for any p-order vector $v \in X$ if $\nabla^i f(x_*)(H_i) < 0$ or it will be violated if $\nabla^i f(x_*)(H_i) > 0$. This leads to the definition of p-order active constraints.

Definition 14 The inequality constraint P is said to be *p-order active* along the sequence H_{p-1} if for all i, $i = 1, \ldots, p-1$, we have $\nabla^i f(x_*)(H_i) = 0$.

Only p-order active constraints enter the necessary conditions for optimality derived via p-order approximations along an admissible sequence H_{p-1}; p-order inactive constraints generate zero multipliers since $DS^{(p)}(P;x_*, H_{p-1}) = X$ (p-order complementary slackness conditions) and can be ignored for high-order approximations.

Proposition 15 *If the constraint $P = \{x \in X: f(x) \le 0\}$ is p-order active along the sequence H_{p-1}, then the p-order feasible cone, $FC^{(p)}(P;x_*, H_{p-1})$, consists of all vectors $(w, \gamma) \in X \times \mathbf{R}_+$ which satisfy*

$$f'(x_*)w + \gamma R_p[f](x_*; H_{p-1}) < 0.$$

Hence, if $f'(x_*) \neq 0$, then $FC^{(p)}(P;x_*, H_{p-1})$ is nonempty, open and convex.

High-Order Feasible Cones to Closed Convex Inequality Constraints

Let $C \subset X$ be a closed convex set with nonempty interior. Again we assume that H_{p-1} is a $(p-1)$-order feasible sequence. Note that it follows from Definition 3 that $FS^{(p)}(C;x_*, H_{p-1})$ is open (since any vector in the neighborhood V of v also lies in $FS^{(p)}(C;x_*, H_{p-1})$). It is also clear that $FS^{(p)}(C;x_*, H_{p-1})$ is convex, since C is. Thus $FC^{(p)}(C;x_*, H_{p-1})$ is an open, convex cone. Furthermore, if there exists an integer $j <$ p so that $h_j \in FS^{(j)}(C;x_*, H_{j-1})$, then any vector v is allowed as a p-order feasible direction and thus trivially $FS^{(p)}(C;x_*, H_{p-1}) = X$, i. e. the convex constraint $x \in C$ is not p-order active. In this case the necessary conditions for optimality along H_{p-1} are exactly the same as without C.

The dual or polar cone $FC^{(p)}(C;x_*, H_{p-1})^*$ can be identified with all supporting hyperplanes to $FS^{(p)}(C;x_*, H_{p-1})$ at x_*. More precisely, it consists of all linear functionals $(\lambda, \mu) \in X^* \times \mathbf{R}$ which satisfy

$$\langle \lambda, v \rangle + \mu \ge 0, \quad \forall v \in FS^{(p)}(C; x_*, H_{p-1}).$$

Corollary 16 [13] *Let $C \subset X$ be a closed convex set with nonempty interior and suppose the p-order feasible set $FS^{(p)}(C;x_*, H_{p-1})$ is nonempty. If $(\lambda, \mu) \in FC^{(p)}(C;x_*, H_{p-1})^*$, then λ is a supporting hyperplane to C at x_*.*

Generalized Necessary Conditions for Optimality

We now give generalized necessary conditions for optimality for problem (P) based on general p-order approximations. We assume as given a sequence $H_{p-1} = (h_1, \ldots, h_{p-1}) \in X^{p-1}$ with the following properties:

P1) The first $p-1$ directional derivatives of F along H_{p-1} vanish,

$$\nabla^i F(x_*)(H_i) = 0, \quad \forall i = 1, \ldots, p-1,$$

the compatibility conditions

$$R_{p-1+i}[F](x_*; H_{p-1}) \in Y_i$$

are satisfied for $i = 1, \ldots, p-1$, and the operator F is p-regular at x_* in direction of the sequence H_{p-1}.

P2) Either the first nonvanishing derivative $\nabla^i I(x_*)(H_i)$ is negative or $\nabla^i I(x_*)(H_i) = 0$ for $i = 1, \ldots, p-1$.

P3) If the jth constraint is not p-order active, then the first nonzero derivative $\nabla^i f(x_*)(H_i)$ is negative.

P4) $FS^{(p)}(C; x_*, H_{p-1})$ is nonempty.

These conditions guarantee respectively that the corresponding p-order approximating cones to the constraints or the functional I are nonempty and convex. The next theorem generalizes the classical first order necessary conditions for optimality for a mathematical programming problem with convex inequality constraints [7, Thm. 11.4].

Theorem 17 *If x_* is optimal for problem (P), then given any sequence $H_{p-1} = (h_1, \ldots, h_{p-1}) \in X^{p-1}$ for which conditions P1)–P4) are satisfied, there exist Lagrange multipliers $\nu_i \geq 0$, $i = 0, \ldots, m$, functionals $y_i^* \in Y_{i-1}^{\perp_i}$, $i = 1, \ldots, p$, and a supporting hyperplane $\langle \lambda, \nu \rangle + \mu \geq 0$ for all $\nu \in FS^{(p)}(C; x_*, H_{p-1})$, all depending on the sequence H_{p-1}, such that the multipliers ν_i, $i = 0, \ldots, m$, and λ do not all vanish, and*

$$\lambda \equiv \nu_0 I'(x_*) + \sum_{j=1}^{m} \nu_j f_j'(x_*) + \sum_{i=1}^{p} G_i^* y_i^*, \tag{15}$$

$$\mu \leq \nu_0 R_p[I](x_*; H_{p-1}) + \sum_{j=1}^{m} \nu_j R_p[f_j](x_*; H_{p-1}) + \sum_{i=1}^{p} \left\langle y_i^*, R_{p-1+i}[F](H_{p-1}) \right\rangle. \tag{16}$$

Furthermore, the following p-order complementary slackness conditions hold:

- *$\nu_0 = 0$ if $DS^{(p)}(I; x_*, H_{p-1}) = X$;*
- *$\nu_j = 0$ if $FS^{(p)}(P_j; x_*, H_{p-1}) = X$;*
- *$\lambda = 0$ if $FS^{(p)}(C; x_*, H_{p-1}) = X$.*

Remark 18 This theorem gives the formulation for the case which is *nondegenerate* in the sense that the operator $\mathcal{G}_p$ is onto and it is this condition which implies the nontriviality of the multipliers ν_j, $j = 0, \ldots, m$, and λ. If $\mathcal{G}_p$ is not onto, but $\mathrm{Im}\mathcal{G}_p$ is closed, while all the other conditions remain in effect, then a degenerate version of this theorem can easily be obtained by choosing a nontrivial multiplier $\widetilde{y}^* \in (\mathrm{Im}\,\mathcal{G}_p)^{\perp}$. This then gives rise to nontrivial multipliers $y_i^* \in Y_{i-1}^{\perp_i}$ which have the property that $\sum_{i=1}^{p} G_i^* y_i^* \equiv 0$. Thus (15) still holds if we set $\nu_j = 0$, for $j = 0, \ldots, m$, and $\lambda = 0$. Thus the difference is that it can only be asserted that not all of the multipliers $\nu_j, j = 0, \ldots, m$, $y_i^* \in Y_{i-1}^{\perp_i}$, $i = 1, \ldots, p$, and λ do vanish.

See also

► Kuhn–Tucker Optimality Conditions

References

1. Arutyunov AV (1991) Higher-order conditions in anormal extremal problems with constraints of equality type. Soviet Math Dokl 42(3):799–804
2. Arutyunov AV (1996) Optimality conditions in abnormal extremal problems. System Control Lett 27:279–284
3. Avakov ER (1985) Extremum conditions for smooth problems with equality-type constraints. USSR Comput Math Math Phys 25(3):24–32. (Zh Vychisl Mat Fiz 25(5))
4. Avakov ER (1988) Necessary conditions for a minimum for nonregular problems in Banach spaces. Maximum principle for abnormal problems in optimal control. Trudy Mat Inst Akad Nauk SSSR 185:3–29; 680–693 (In Russian.)
5. Avakov ER (1989) Necessary extremum conditions for smooth anormal problems with equality-and inequality constraints. J Soviet Math 45:3–11. (Matematicheskie Zametki 45)
6. Ben-Tal A, Zowe J (1982) A unified theory of first and second order conditions for extremum problems in topological vector spaces. Math Program Stud 19:39–76
7. Girsanov IV (1972) Lectures on mathematical theory of extremum problems. Springer, Berlin
8. Hoffmann KH, Kornstaedt HJ (1978) Higher-order necessary conditions in abstract mathematical programming. J Optim Th Appl (JOTA) 26:533–568
9. Ioffe AD, Tikhomirov VM (1979) Theory of extremal problems. North-Holland, Amsterdam
10. Izmailov AF (1994) Optimality conditions for degenerate extremum problems with inequality-type constraints. Comput Math Math Phys 34:723–736
11. Ledzewicz U, Schättler H (1995) Second-order conditions for extremum problems with nonregular equality constraints. J Optim Th Appl (JOTA) 86:113–144
12. Ledzewicz U, Schättler H (1998) A high-order generalization of the Lyusternik theorem. Nonlinear Anal 34:793–815
13. Ledzewicz U, Schättler H (1999) High-order approximations and generalized necessary conditions for optimality. SIAM J Control Optim 37:33–53
14. Lyusternik LA (1934) Conditional extrema of functionals. Math USSR Sb 31:390–401
15. Tretyakov AA (1984) Necessary and sufficient conditions for optimality of p-th order. USSR Comput Math Math Phys 24(1):123–127

Hilbert's Thirteenth Problem

VICTOR KOROTKICH
Central Queensland University, Mackay, Australia

MSC2000: 01A60, 03B30, 54C70, 68Q17

Article Outline

Keywords
See also
References

Keywords

Superpositions of functions; Algebraic equations; E-entropy; Information

The formulation of Hilbert's thirteenth problem [8] reads: 'impossibility of solving the general equation of degree 7 by means of any continuous functions depending only on two variables' [21].

On this basis, D. Hilbert proposed that the complexity of functions is specified essentially by the number of variables. However, as turned out later, this proposal being valid for analytic functions is not true in the general case. In particular, complexity of r times continuously differentiable functions of n variables depends not on the number of variables n but on the ratio n/r.

It is known that the equation of third degree can be reduced by translation to

$$X^3 + pX + q = 0,$$

which has the solution (S. del Ferro, 16th century)

$$X = \left[-\frac{q}{2} + \sqrt{\frac{4p^3 + 27q^2}{4(27)}}\right]^{1/3} + \left[-\frac{q}{2} - \sqrt{\frac{4p^3 + 27q^2}{4(27)}}\right]^{1/3}.$$

The equation of fourth degree can be solved by superposition of addition, multiplication, square roots, cube roots and fourth roots.

To try to solve *algebraic equations* of higher degree (a vain hope according to N.H. Abel and E. Galois), the idea of W. Tschirnhausen in 1683 [24] was to adjoin a new equation, i. e., to

$$P(X) = 0$$

one adjoins

$$Y = Q(X),$$

where Q is a polynomial of degree strictly less than that of P, chosen expediently. In this way one can show that the roots of an equation of degree 5 can be expressed via the usual arithmetic operations in terms of radicals and of the solution $\phi(x)$ of the quintic equation

$$X^5 + xX + 1 = 0$$

depending on the parameter x. Similarly for the equation of degree 6, the roots are expressible in the same way if we include also a function $\theta(x, y)$, a solution of a 6th-degree equation depending on two parameters x and y.

For degree 7 we would have to include also a function $\sigma(x, y, z)$, solution of the equation

$$X^7 + xX^3 + yX^2 + zX + 1 = 0.$$

Hence the natural question: Can $\sigma(x, y, z)$ be expressed by superposition of algebraic functions of two variables [10]?

A great number of papers are devoted to the representability of functions as *superpositions of functions* depending on a smaller number of variables and satisfying certain additional conditions such as algebraicity, analyticity and smoothness. Hilbert was aware of the fact that superpositions of discontinuous functions represent all functions of a larger number of variables. He also knew about the existence of analytic functions of three variables that cannot be represented by any finite superpositions of analytic functions of two variables [8].

In the statement of his 13th problem, Hilbert proceeded from a result of Tschirnhausen [24], according to which a root of an algebraic equation of degree $n > 5$, i. e., a function $f(x_1, \ldots, x_n)$ determined by an equation

$$f^n + x_1 f^{n-1} + \cdots + x_n = 0, \tag{1}$$

can be expressed as a superposition of algebraic functions of $n-4$ variables [21]. Hilbert assumed that the

number $n-4$ cannot be reduced for $n = 6, 7, 8$ and also proved that in order to solve an equation of degree $n = 9$ it suffices to have functions of $n-5$ variables [9]. A. Wiman [26] extended the latter result to $n > 9$, while N. Chebotarev [6] reduced the number of variables involved in the representation of functions to $n-6$ for $n \geq 21$ and to $n-7$ for $n \geq 121$.

Chebotarev was the first to attempt to find topological obstructions to the representability of algebraic functions as superpositions of algebraic functions, but his proofs were not convincing [5,17]. Using topological notions related to the behavior of a many-valued algebraic function on and near a branching manifold, it is proved that algebraic functions cannot be represented by complete superpositions of integral algebraic functions. Completeness means that the represented function must involve all the branches of the many-valued functions and not only one of them as, for example, in the formulas expressing solutions to equations of the 3rd and the 4th degree [21].

Certain topological obstructions to the representation by a complete superpositions of algebraic functions were constructed in this way [2]. V. Lin [15] established the following, most complete, result: In any neighborhood of the origin for $n \geq 3$ the root $f(x_1, \ldots, x_n)$ of equation (1) is not a complete superposition of entire algebroid functions of fewer than $n-1$ variables and single-valued holomorphic functions of an arbitrary number of variables. Thus, from the standpoint of complete superpositions of entire algebraic functions, even fourth-degree equations cannot be solved without using functions of three variables [21].

Hilbert had had another motivation for his thirteenth problem: *nomography*, the method of solving equations by drawing a one-parameter family of curves. This problem, arising in the methods of computation of Hilbert's time, inspired the development of Kolmogorov's notion of *ε-entropy* [20]. Applications of ε-entropy have its crucial role in theories of approximation now used in computer science [22].

In *Kolmogorov ε-entropy*, a natural characteristic of a function class F is

$$H_\varepsilon(F) = \log_2 N_\varepsilon(F),$$

where $N_\varepsilon(F)$ is the minimum number of points in an ε-net in F. Broadly speaking, the ε-entropy of a function class F is the amount of *information* needed to specify with accuracy ε a function of the class F. A main problem in ε-entropy is estimates for the rate of growth of $H_\varepsilon(F)$ as $\varepsilon \to 0$ for Lipschitz functions, classes of analytic functions and functions possessing a given number of derivatives. A.N. Kolmogorov showed that the ε-entropy of r times continuously differentiable functions of n variables grows as $\varepsilon^{-n/r}$ [20].

Since a digital computer can store only a finite set of numbers, functions must be replaced by such finite sets. Therefore, studies in ε-entropy are important for the correct estimation of the possibilities of computational methods for approximately representing functions, their implementation on computers and their storage in the computer memory.

Also ε-entropy has many other applications [23]. An ε-net of Lipschitz functions of n variables is constructed to design global optimization algorithms. This ε-net is based on the Kolmogorov's minimal ε-net of one-dimensional Lipschitz functions and is encoded in terms of monotone functions of k-valued logic. This construction gives a representation of an n-dimensional global optimization problem by a minimal number of one-dimensional ones without loss of information [13].

Let us briefly recall the history of the solution of the Hilbert's thirteenth problem by Kolmogorov and V. Arnol'd. Hilbert's problem was first solved on the basis of ideas by using technique developed by A. Kronrod [14]. In this way Kolmogorov proved that any continuous function of $n \geq 4$ variables can be represented as a superposition of continuous functions of three variables [11]. For an arbitrary function of four variables the representation has the form

$$f(x_1, x_2, x_3, x_4) = \sum_{r=1}^{4} h^r[x_4, g_1^r(x_1, x_2, x_3), g_2^r(x_1, x_2, x_3)].$$

The question whether an arbitrary continuous function of three variables can be represented as a superposition of continuous functions of two variables remained open. The method reduced the representability of functions of three variables as superpositions of functions of two variables to a representability problem for functions defined on universal trees of three-dimensional space [21].

Contrary to the expectations of Hilbert and of his contemporary mathematicians, in 1957 Arnol'd [1], who was a student of Kolmogorov, solved the latter problem and gave the final solution to Hilbert's thirteenth problem in the form of a theorem asserting that any continuous function of $n \geq 3$ variables can be represented as a superposition of functions of two variables [21].

A few weeks later Kolmogorov showed that any continuous function f of n variables can be represented as a superposition

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \chi_q \left[\sum_{p=1}^{n} \phi^{pq}(x_p) \right] \qquad (2)$$

of continuous functions of one variable and the operation of addition [12]. In Kolmogorov's representation (2) the inner functions ϕ^{pq} are fixed and only the outer functions χ_q depend on the represented function f.

The results of [11] do not follow from the theorem presented in [12] in their exact statements, but their essence (in the sense of the possibility of representing functions of several variables by means of superpositions of functions of a smaller number of variables and their approximation by superpositions of a fixed form involving polynomials in one variable and addition) is obviously contained in it [12]. The method for proving the theorem is more elementary than that in [1,11] and reduces to direct constructions and calculations. In Kolmogorov's opinion, the proof of the theorem was his most technically difficult achievement [21].

Thorough proofs of Kolmogorov's theorem and the lemmas of his paper [12] were published in [16,18,20] and others. G. Lorenz [16] noted that the outer functions χ_q can be replaced by a single function χ. D. Sprecher [18] reduced all the inner functions to translations and extensions of a single function ψ with the property that there exits $\varepsilon > 0$ and $\lambda > 0$ such that any continuous function of n variables can be represented as

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \chi[\lambda^p \psi(x_p + \varepsilon q) + q)]. \qquad (3)$$

B. Fridman [7] proved that the inner functions ϕ^{pq} in (2) can be chosen so that they satisfy a Lipschitz condition. Sprecher [19] extended this result to the representation (3) (the function ψ can be chosen to satisfy a Lipschitz condition).

It follows from Kolmogorov's representation (2) and Bari's representation [3] of any continuous function of one variable as a sum of three superpositions of absolutely continuous functions $\sum f_k \circ gk$ that all continuous functions of any number of variables can be represented by means of superpositions of absolutely continuous functions of one variable and the operation of addition [21].

In the opposite direction are the results of A. Vitushkin [25] and L. Bassalygo [4]. When we deal with superpositions of formal series or analytic functions it can be shown that, for example, almost every entire function has at an arbitrary point of C^3 a germ which is not expressible by superposition of series in two variables. So there are many more entire functions of three variables than of two [10]. The result of Vitushkin is that there exist r times continuously differentiable functions of n variables that cannot be expressed in terms of finite superpositions of $s \geq 1$ times continuously differentiable functions of $k < n$ variables if $n/r > ks$ [25], representability depends on n/r. Bassalygo proved that for any three functions ψ_k continuous on a square there exists a continuous function f which cannot be represented as $\sum \chi_k \circ \psi_k$ for any continuous χ_k [4].

See also

► History of Optimization

References

1. Arnold V (1957) On the representation of continuous functions of three variables as superpositions of continuous functions of two variables. Dokl Akad Nauk SSSR 114(4):679–681(in Russian.)
2. Arnold V (1970) On cohomology classes of algebraic functions invariant under a Tschirnhausen transformation. Funkts Anal i Prilozhen 4(1):84–85(in Russian.)
3. Bari N (1930) Memoire sur la representation finie des fonctions continues. Math Anal 103:185–248
4. Bassalygo L (1966) On the representation of continuous functions of two variables by continuous functions of one variable. Vestn MGU Ser Mat-Mekh 21:58–63(in Russian).
5. Chebotarev N (1943) The resolvent problem and critical manifolds. Izv Akad Nauk SSSR Ser Mat 7:123–146(in Russian.)
6. Chebotarev N (1954) On the resolvent problem. Uchen Zap Kazan Univ 114(2):189–193(in Russian.)

7. Fridman B (1967) Increasing smoothness of functions in Kolmogorov's theorem on superpositions. Dokl Akad Nauk SSSR 177:1019–1022(in Russian.)
8. Hilbert D (1902) Sur les problemes futurs des mathematiques. Proc. Second Internat. Congress of Mathematicians, Gauthier-Villars, 58–114
9. Hilbert D (1927) Ueber die Gleichung neunten Grades. Math Ann 97:243–250
10. Kantor J (1996) Hilbert's problems and their sequels. Math Intelligencer 18(1):21–30
11. Kolmogorov A (1956) On the representation of continuous functions of several variables as superpositions of continuous functions of a smaller number of variables. Dokl Akad Nauk SSSR 108(2):179–182 (in Russian.)
12. Kolmogorov A (1957) On the representation of continuous functions of several variables as superpositions of continuous functions of one variable and addition. Dokl Akad Nauk SSSR 114(5):953–956 (in Russian.)
13. Korotkich V (1990) Multilevel dichotomy algorithm in global optimization. In: Sebastian H, Tammer K (eds) System Modelling and Optimization. Springer, Berlin, pp 161–169
14. Kronrod A (1950) Uspekhi Mat Nauk 5(1):24–134(in Russian.)
15. Lin V (1976) Superpositions of algebraic functions. Funkts Anal i Prilozhen 10(1):37–45(in Russian.)
16. Lorenz G (1962) Metric entropy, width and superpositions of functions. Amer Math Monthly 69:469–485
17. Morozov V (1954) On some questions in the resolvent problem. Uchen Zap Kazan Univ 114(2):173–187(in Russian.)
18. Sprecher D (1965) On the structure of continuous functions of several variables. Trans Amer Math Soc 115:340–355
19. Sprecher D (1972) An improvement in the superposition theorem of Kolmogorov. J Math Anal Appl 38:208–213
20. Tikhomirov V (1963) A.N. Kolmogorov's work on ε-entropy of function classes and superpositions of functions. Uspekhi Mat Nauk 18(5):55–92(in Russian.)
21. Tikhomirov V (ed) (1991) Selected works of A.N. Kolmogorov: Mathematics and mechanics, vol 1. Kluwer, Dordrecht
22. Traub J, Wasilkowski G, Wozniakowski H (1988) Information-based complexity. Acad. Press, New York
23. Traub J, Wozniakowski H (1980) Theory of optimal algorithms. Acad. Press, New York
24. Tschirnhausen W (1683) Methodus auferendi omnes terminos intermedios ex data equatione. Acta Eruditorum
25. Vitushkin A (1955) On multidimensional variations. Gostekhteorizdar
26. Wiman A (1927) Ueber die Anwendung der Tschirnhausen Transformation auf die Reduktion algebraischer Gleichungen. Nova Acta R Soc Sci Uppsaliensis extraordin. edit.: 3–8

History of Optimization

Ding-Zhu Du[1], Panos M. Pardalos[2], Weili Wu[3]

[1] Department Computer Sci. and Engineering, University Minnesota, Minneapolis, USA

[2] Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

[3] Department Computer Sci. and Engineering, University Minnesota, Minneapolis, USA

MSC2000: 01A99

Article Outline

Keywords
See also
References

Keywords

History; Optimization

Did you ever watch how a spider catches a fly or a mosquito? Usually, a spider hides at the edge of its net. When a fly or a mosquito hits the net, the spider will pick up each line in the net to choose the tense one and then goes rapidly along the line to its prey. Why does the spider chooses the tense line? Some biologists explain that the line gives the shortest path from the spider to its prey.

Did you heard the following story about a wise general? He had a duty to capture a town behind a mountain. When he and his soldiers reached the top of the mountain, he found that his enemy had already approached the town very closely from another way. His dilemma was how to get in the town before the enemy arrive. It was a challenging problem for the general. The general solved the problem by asking each soldier to roll down the mountain in a blanket. Why is this faster? Physicists tell us that a free ball rolling down a mountain always chooses the most rapid way.

Do you know the tale of a horse match of Tian Gi? It is a story set in BC time. Tian Gi was a general in one of several small counties of China, called Qi. The King of Qi knew that Tian Gi had several good horses and ordered Tian Gi to have a horse match with him. The match consisted of three rounds. In each round, each

side chose a horse to compete with the other side. Tian Gi knew that his best horse could not compete with the best one of the King, his second best horse could not compete with the second best one of King, and his third best horse could not compete with the third best one of the King. Therefore, he did not use his best horse against the best horse of the King. Instead, he put his third best horse in the first round against the best one of the King, his best horse in the second round against the second best one of the King, and his second best horse in the third round against the third best one of the King. The final result was that although he lost the first round of the match, he won the last two rounds. Tian Gi's strategy was the best to win this match. Today, economists tell us that many economic systems and social systems can be modeled into games. Each contestant in the game tries to maximize certain benefits.

Optimality is a fundamental principle, establishing natural lows, ruling biologic behaviors, and conducting social activities. Therefore, optimization started from the earliest stages of human civilization. Of course, before mathematics was well established, optimization could be done only by simulation. One may find many wise men's stories in the human history about it. For example, to find the best way to get out of a mountain, someone followed a stream, and to find the best way to get out from a desert, someone set an old horse free and followed the horse's trace.

In the 19th century or even today, simulation is still used for optimizing something. For example, to find a shortest path on a network, one may make a net with rope in a proportional size and pull the net tightly between two destinations. The tense rope shows the shortest path. To find an optimal location of a school for three villages, one may drill three holes on a table and put a piece of rope in each hole. Then tie three rope-ends above the table together and hang a one-kg-weight on each rope-end under the table. When this mechanical system is balanced, the knot of the three rope-pieces points out the location of the school.

The history of optimization in mathematics can be divided into three periods.

In the first period, one did not know any general method to find a maximum/minimum point of a function. Only special techniques were found to maximize/minimize some special functions. A typical function is the quadratic function of one variable

$$y = ax^2 + bx + c.$$

The study of quadratic functions was closely related to the study of constantly-accelerating movement. What is the highest point that a stone is thrown out with certain initial speed and certain angle? What is the farthest point where a stone thrown with certain initial speed can reach when throwing angle varies? These were questions considered by some physicists and generals. In fact, the stone-throwing machine was an important weapon in military.

Today (as of 2000), computing maximum/minimum points of a quadratic function is still an important technique of optimization, existing in elementary mathematics books. The technique had been also extended to other functions such as

$$y = \frac{x^2 + x + 1}{x^2 + 2x + 3}.$$

Actually, multiplying both sides by $x^2+ 2x+3$ and simplifying, we obtain

$$(y-1)x^2 + (2y-1)x + (3y-1) = 0.$$

Since x is a real number, we must have

$$(2y-1)^2 - 4(y-1)(3y-1) \geq 0.$$

Therefore,

$$-8y^2 + 12y - 3 \geq 0,$$

that is,

$$2(3-\sqrt{3}) \leq y \leq 2(3+\sqrt{3}).$$

It is interesting to note that with this technique we obtained the global maximum and minimum of y.

A new period started in 1646 by P. de Fermat. He proposed, in his paper [5], a general approach to compute local maxima/minima points of a differentiable function, that is, setting the derivative of the function to be zero. Today, this approach is still included in almost all textbooks of calculus as an application of differentiation. In this period, optimization existed scattered and disorderly in mathematics. Because optimization had not become an important branch of applied mathematics, some mathematicians did not pay so much attention to results on optimization and some contributions

were even not put in any publication. This left many mysteries in the history of optimization.

For example, who is the first person who proposed the Steiner tree? It was one such mystery. To obtain a clear view, let us explain it in a little detail.

In the same paper mentioned above, Fermat also studied a problem of finding a point to minimize the total distance from it to three given points in the Euclidean plane. Suppose three given points are (x_1, y_1), (x_2, y_2), and (x_3, y_3). Then the total distance from a point (x, y) to these three points is

$$f(x, y) = \sum_{i=1}^{3} \sqrt{(x - x_i)^2 + (y - y_i)^2}.$$

By Fermat's general method, the minimum point of $f(x, y)$ must satisfy the following equations

$$\frac{\partial f}{\partial x} = \sum_{i=1}^{3} \frac{x - x_i}{\sqrt{(x - x_i)^2 + (y - y_i)^2}} = 0,$$

$$\frac{\partial f}{\partial y} = \sum_{i=1}^{3} \frac{y - y_i}{\sqrt{(x - x_i)^2 + (y - y_i)^2}} = 0.$$

However, obtaining x and y from this system of equations seems hopeless. Therefore, Fermat mentioned this problem again in a letter to A. Mersenne that it would be nice if a clear solution could be obtained for this problem.

E. Torricelli, a student of G. Galilei, obtained a clever solution with a geometric method. He showed that if three given points form a triangle without an angle of at least 120°, then the solution is a point at which three segments from it to three given points produce three angles of 120°. Otherwise, the solution is the given point at which the triangle formed by the three given points has an angle of at least 120°.This result can also be proved by the mechanic system described at the beginning of this article. In the first case, the knot of the three rope-pieces stays not at any given point and hence the balance condition of the three forces of equal magnitude yields the condition on the angles. In the second case, the knot falls in one of the three holes, and the condition on the angle guarantees that the knot would not move away from the hole.

Fermat's problem was extensively studied later and was generalized to four points by J.Fr. Fagnano in 1775 and to n points by P. Tedenat and S. L'Huiller in 1810. Fagnano pointed out that it is very easy to find the solution of Fermat's problem for four points. When four given points form a convex quadrilateral, the solution of Fermat's problem is the intersection of two diagonals, i. e., the intersection of two diagonals minimizes the total distance from one point to four given points. Otherwise, there must be one of the given points lying inside the triangle formed by the other three given points; this given point is the solution.

On March 19, 1836, H.C. Schumacher wrote a letter to C.F. Gauss. In his letter, he mentioned a paradox about Fermat's problem: Consider a convex quadrilateral $ABCD$. It has been known that the solution of Fermat's problem for four points A, B, C, and D is the intersection E of diagonals AC and BD. Suppose extending DA and CB can obtain an intersection F. Now, moving A and B to F. Then E will also be moved to F. However, when the angle at F is less than 120°, the point F cannot be the solution of Fermat's problem for three given points F, D, and C.What happens?

On March 21, 1836, Gauss wrote a letter to Schumacher in which he explained that the mistake of Schumacher's paradox occurs at the place where Fermat's problem for four points A, B, C, and D is changed to Fermat's problem for three points F, C, and D. When A and B are identical to F, the total distance from E to four points A, B, C, and D equals $2EF + EC + ED$, not $EF + EC + ED$. Thus, the point E may not be the solution of Fermat's problem for F, C, and D. More importantly, Gauss proposed a new problem. He said that it is more interesting to find a shortest network rather than a point. Gauss also presented several possible connections of the shortest network for four given points.

Unfortunately, Gauss' letter was discovered only in 1986. From 1941 to 1986, many publications have followed R. Courant and H. Robbins who in their popular book [2] called Gauss' problem as the Steiner tree problem. The Steiner tree has become a popular and important name. If you search 'Steiner tree' with 'yahoo.com' on the internet, then you will receive a list of 4675 webpages on Steiner trees. We have no way to change back the name from Steiner trees to Gauss trees. It may be worth mentioning that J. Steiner, a geometrician in 19th century whose name is used for the shortest networks, has not been found so far to have any significant contribution to Steiner trees.

G.B. Dantzig, who first proposed the simplex method to solve linear programming in 1947, stated in [4]: 'What seems to characterize the pre- 1947 era was lack of any interests in trying to optimize'. Due to the lack of interests in optimization, many important works appeared before 1947 were ignored. This happened not only for Steiner trees, but also to other areas of optimization, including some important contributions in linear and nonlinear programming.

The discovery of linear programming started a new age of optimization. However, in [4], Dantzig made the following comment: 'Linear programming was unknown prior to 1947'. This is not quite correct; there were some late exceptions. J.B.J. Fourier (of Fourier series fame) in 1823 and the well-known Belgian mathematician Ch. de la Vallée Poussin in 1911 each wrote a paper about it. Their work had as much influence on post- 1947 developments as would finding in an Egyptian tomb an electronic computer built in 3000 BC. L.V. Kantorovich's remarkable 1939 monograph on the subject was also neglected for ideological reasons in the USSR. It was resurrected two decades later after the major developments had already taken place in the West. An excellent paper by F.L. Hitchcock in 1941 on the transportation problem was also overlooked until after others in the late 1940s and early 1950s have independently rediscovered its properties.

He also recalled how he made his discovery: 'My own contribution grew out of my World War II experience in the Pentagon. During the war period (1941–1945), I had become an expert on programming-planning methods using desk calculators. In 1946 I was mathematical advisor to the US Air Force Comptroller in the Pentagon. I had just received my PhD (for research I had done mostly before the war) and was looking for an academic position that would pay better than a low offer I had received from Berkeley. In order to entice me to not take another job, my Pentagon colleagues, D. Hitchcock and M. Wood, challenged me to see what I could do to mechanize the planning process. I was asked to find a way to more rapidly compute a time-staged development, training and logistical supply program. In those days *mechanizing* planning meant using analog devices or punch-card equipment. There were no electronic computers'.

This challenge problem made Dantzig discover his great work in linear programming without electronic computer. But, we have to point out that it is due to the rapid development of computer technology that applications of linear programming can be made so wide and so great, and areas of optimization can have so fast growing.

In 1951, A.W. Tucker and his student H.W. Kuhn published the Kuhn–Tucker conditions. This is considered as an initial point of nonlinear programming. However, A. Takayama has an interesting comment on these condition: 'Linear programming aroused interest in constraints in the form of inequalities and in the theory of linear inequalities and convex sets. The Kuhn–Tucker study appeared in the middle of this interest with a full recognition of such developments. However, the theory of nonlinear programming when constraints are all in the form of equalities has been known for a long time – in fact, since Euler and Lagrange. The inequality constraints were treated in a fairly satisfactory manner already in 1939 by Karush. Karush's work is apparently under the influence of a similar work in the calculus of variations by Valentine. Unfortunately, Karush's work has been largely ignored'. Yet, this is another work that appeared before 1947 and it was ignored. In the 1960s, G. Zoutendijk, J.B. Rosen, P. Wolfe, M.J.D. Powell, and others published a number of algorithms for solving nonlinear optimization problems. These algorithms form the basis of contemporary nonlinear programming.

In 1954, L.R. Ford and D.R. Fulkerson initiated the study on network flows. This is considered as a starting point on *combinatorial optimization* although Fermat is the first one who studied a major combinatorial optimization problem. In fact, it was because of the influence of the results of Ford and Fulkerson, that interests on combinatorial optimization were growing, and so many problems, including Steiner trees, were proposed or re-discovered in history. In 1958, R.E. Gomory published the cutting plane method. This is considered as an initiation of *integer programming*, an important direction of combinatorial optimization.

In 1955, Dantzig published his paper [3] and E.M.L. Beale proposed an algorithm to solve similar problems. They started the study on *stochastic programming*. R.J-B. Wets in the 1960s, and J.R. Birge and A. Prékopa in the 1980s made important contributions in this branch of optimization.

Now, optimization has merged into almost every corner of economics. New branches of optimization appeared in almost every decade, *global optimization*, *nondifferential optimization*, *geometric programming*, *large scale optimization*, etc. No one in his/her whole life is able to study all branches in optimization. Each researcher can only be an expert in a few branches of optimization.

Of course, the rapid development of optimization is accomplished with recognition of its achievements. One important fact is that several researchers in optimization have received the Nobel Prize in economics, including Kantorovich and T.C. Koopmans. They received the Nobel Prize on economics in 1975 for their contributions to the theory of optimum allocation of resources. H.M. Markowitz received the Nobel Prize on economics in 1990 for his contribution on the quadratic programming model of financial analysis.

Today, optimization has become a very large and important interdisciplinary area between mathematics, computer science, industrial engineering, and management science. The 'International Symposium on Mathematical Programming' is one of major conferences on optimization. From the growing number of papers presented in this conference we may see the projection of growing optimization area:

1949) Chicago, USA, 34 papers;
1951) Washington DC, USA, 19 papers;
1955) Washington DC, USA, 33 papers;
1959) Santa Monica, USA, 57 papers;
1962) Chicago, USA, 43 papers;
1964) London, UK, 83 papers;
1967) Princeton, USA, 91 papers;
1970) The Hague, The Netherlands, 137 papers;
1973) Stanford, USA, about 250 papers;
1976) Budapest, Hungary, 327 papers;
1979) Montreal, Canada, 458 papers;
1982) Bonn, FRG, 554 papers;
1985) Cambridge, USA, 589 papers;
1988) Tokyo, Japan, 624 papers.
(This data is quoted from [1].)

With the current fast growth of computer technology optimization it is expected to continue its great speed of developments. These developments may contain include a deep understanding of the successful heuristics for combinatorial optimization problems with nonlinear programming approaches. It may also include digital simulations to some natural optimization process. As many mysteries and open problems still exist in optimization, it will still be an area receiving a great attention.

See also

- ► Carathéodory, Constantine
- ► Carathéodory Theorem
- ► Inequality-constrained Nonlinear Optimization
- ► Kantorovich, Leonid Vitalyevich
- ► Leibniz, Gottfried Wilhelm
- ► Linear Programming
- ► Operations Research
- ► Von Neumann, John

References

1. Balinski ML (1991) Mathematical programming: Journal, society, recollections. In: Lenstra JK, Rinnooy Kan AHG, Schrijver A (eds) History of Mathematical Programming. North-Holland, Amsterdam, pp 5–18
2. Courant R, Robbins H (1941) What is mathematics? Oxford Univ. Press, Oxford
3. Dantzig GB (1955) Linear programming under uncertainty. Managem Sci 1:197–206
4. Dantzig GB (1991) Linear programming: The story about how it began. In: Lenstra JK, Rinnooy Kan AHG, Schrijver A (eds) History of Mathematical Programming. North-Holland, Amsterdam, pp 19–31
5. de Fermat P (1934) Abhandlungen über Maxima und Minima. In: Oswalds Klassiker der exakten Wissenschaft, vol 238. H. Miller, reprint from original

Homogeneous Selfdual Methods for Linear Programming

ERLING D. ANDERSEN
Odense University, Odense M, Denmark

MSC2000: 90C05

Article Outline

Keywords
See also
References

Keywords

Optimization; Linear programming; Interior point methods; Homogeneous; Selfdual

The linear program

$$\begin{cases} \min & c^\top x \\ \text{s.t.} & Ax = b, \\ & x \geq 0 \end{cases} \tag{1}$$

may have an optimal solution, be primal infeasible or be dual infeasible for a particular set of data $c \in \mathbf{R}^n$, $b \in \mathbf{R}^m$, and $A \in \mathbf{R}^{m \times n}$. In fact the problem can be both primal and dual infeasible for some data where (1) is denoted dual infeasible if the dual problem

$$\begin{cases} \max & b^\top y \\ \text{s.t.} & A^\top y + s = c, \\ & s \geq 0 \end{cases} \tag{2}$$

corresponding to (1) is infeasible. The vector s is the so-called *dual slacks*.

However, most methods for solving (1) assume that the problem has an optimal solution. This is in particular true for interior point methods. To overcome this problem it has been suggested to solve the *homogeneous and selfdual model*

$$\begin{cases} \min & 0 \\ \text{s.t.} & Ax - b\tau = 0, \\ & -A^\top y + c\tau \geq 0, \\ & b^\top y - c^\top x \geq 0, \\ & x \geq 0, \quad \tau \geq 0, \end{cases} \tag{3}$$

instead of (1). Clearly, (3) is a homogeneous LP and is selfdual which essentially follows from the constraints form a skew-symmetric system. The interpretation of (3) is τ is a homogenizing variable and the constraints represent primal feasibility, dual feasibility, and reversed weak duality.

The homogeneous model (3) was first studied by A.J. Goldman and A.W. Tucker [2] in 1956 and they proved that (3) always has a nontrivial solution (x^*, y^*, τ^*) satisfying

$$\begin{cases} x_j^* s_j^* = 0, & \forall j \\ x_j^* + s_j^* > 0, & \forall j, \\ \tau^* \kappa^* = 0, \\ \tau^* + \kappa^* > 0, \end{cases} \tag{4}$$

where $s^* := c\,\tau^* - A^\top y^* \geq 0$ and $\kappa^* := b^\top y^* - c^\top x^* \geq 0$. A solution to (3) satisfying the condition (4) is said to be a *strictly complementary solution*. Moreover, Goldman and Tucker showed that if $(x^*, \tau^*, y^*, s^*, \kappa^*)$ is any strictly complementary solution, then exactly one of the two following situations occurs:

- $\tau^* > 0$ if and only if (1) has an optimal solution. In this case$(x^*, y^*, s^*)/\tau^*$ is an optimal primal-dual solution to (1).
- $\kappa^* > 0$ if and only if (1) is primal or dual infeasible. In the case $b^\top y^* > 0$ ($c^\top x^* < 0$) then (1) is primal (dual) infeasible.

The conclusion is that a strictly complementary solution to (3) provides all the information required, because in the case $\tau^* > 0$ then an optimal primal-dual solution to (1) is trivially given by $(x, y, s) = (x^*, y^*, s^*)/\tau^*$. Otherwise, the problem is primal or dual infeasible. Therefore, the main algorithmic idea is to compute a strictly complementary solution to (3) instead of solving (1) directly.

Y. Ye, M.J. Todd, and S. Mizuno [6] suggested to solve (3) by solving the problem

$$\begin{cases} \min & n^0 z \\ \text{s.t.} & Ax - b\tau - \bar{b}z = 0, \\ & -A^\top y + c\tau + \bar{c}z \geq 0, \\ & b^\top y - c^\top x + \bar{d}z \geq 0, \\ & \bar{b}^\top y - \bar{c}^\top x - \bar{d}\tau = -n^0, \\ & x \geq 0, \quad \tau \geq 0, \end{cases} \tag{5}$$

where

$$\bar{b} := Ax^0 - b\tau^0,$$
$$\bar{c} := -c\tau^0 + A^\top y^0 + s^0,$$
$$\bar{d} := c^\top x^0 - b^\top y^0 + \kappa^0,$$
$$n^0 := (x^0)^\top s^0 + \tau^0 \kappa^0,$$

and

$$(x^0, \tau^0, y^0, s^0, \kappa^0) = (e, 1, 0, 1)$$

(e is an n vector of all ones). It can be proved that the problem (5) always has an optimal solution. Moreover, the optimal value is identical to zero and it is easy to verify that if (x, τ, y, z) is an optimal strictly complementary solution to (5), then (x, τ, y) is a strictly complementary solution to (3). Hence, the problem (5) can solved using any method that generates an optimal strictly complementary solution because the problem always has a solution. Note by construction then $(x, \tau, y, z) = (x^0, \tau^0, y^0, 1)$ is an interior feasible solution to (5). This implies that the problem (1) can be solved by most feasible-interior point algorithms.

X. Xu, P.-F. Hung, and Ye [4] suggest an alternative solution method which is also an interior point algorithm, but specially adapted to the problem (3).The so-called *homogeneous algorithm* can be stated as follows:

1) Choose $(x^0, \tau^0, y^0, s^0, \kappa^0)$ such that $(x^0, \tau^0, s^0, \kappa^0) > 0$. Choose $\varepsilon_f, \varepsilon_g > 0$ and $\gamma \in (0, 1)$ and let $\eta := 1 - \gamma$.
2) $k := 0$.
3) Compute:

$$
\begin{aligned}
r_p^k &:= b\tau^k - Ax^k,\\
r_d^k &:= c\tau^k - A^\top y^k - s^k,\\
r_g^k &:= \kappa^k + c^\top x^k - b^\top y^k,\\
\mu^k &:= \frac{(x^k)^\top s^k + \tau^k\kappa^k}{n+1}.
\end{aligned}
$$

4) If $\| (r_p^k; r_d^k; r_g^k) \| \le \varepsilon_f$ and $\mu^k \le \varepsilon_g$, then terminate.
5) Solve the linear equations

$$
\begin{aligned}
Ad_x - bd_\tau &= \eta r_p^k,\\
A^\top d_y + d_s - cd_\tau &= \eta r_d^k,\\
-c^\top d_x + b^\top d_y - d_\kappa &= \eta r_g^k,\\
S^k d_x + X^k d_s &= -X^k s^k + \gamma\mu^k e,\\
\kappa^k d_\tau + \tau^k d_\kappa &= -\tau^k\kappa^k + \gamma\mu^k,
\end{aligned}
$$

for $(d_x, d_\tau, d_y, d_s, d_\kappa)$ where $X^k := \mathrm{diag}(x^k)$ and $S^k := \mathrm{diag}(s^k)$.
6) For some $\theta \in (0, 1)$, let α^k be the optimal objective value to

$$
\begin{cases}
\max & \theta\alpha\\
\text{s.t.} & \begin{pmatrix} x^k\\ \tau^k\\ s^k\\ \kappa^k \end{pmatrix} + \alpha \begin{pmatrix} d_x\\ d_\tau\\ d_s\\ d_\kappa \end{pmatrix} \ge 0,\\
& \alpha \le \theta^{-1}.
\end{cases}
$$

7)

$$
\begin{pmatrix} x^{k+1}\\ \tau^{k+1}\\ y^{k+1}\\ s^{k+1}\\ \kappa^{k+1} \end{pmatrix} := \begin{pmatrix} x^k\\ \tau^k\\ y^k\\ s^k\\ \kappa^k \end{pmatrix} + \alpha^k \begin{pmatrix} d_x\\ d_\tau\\ d_y\\ d_s\\ d_\kappa \end{pmatrix}
$$

8) $k = k + 1$.
9) goto 3)

The following facts can be proved about the algorithm

$$
\begin{cases}
r_p^{k+1} = (1 - (1-\gamma)\alpha^k) r_p^k,\\
r_d^{k+1} = (1 - (1-\gamma)\alpha^k) r_d^k,\\
r_g^{k+1} = (1 - (1-\gamma)\alpha^k) r_g^k,
\end{cases}
$$

and

$$
\begin{aligned}
&((x^{k+1})^\top s^{k+1} + \tau^{k+1}\kappa^{k+1})\\
&= (1 - (1-\gamma)\alpha^k)((x^k)^\top s^k + \tau^k\kappa^k),
\end{aligned}
$$

which shows that the primal residuals (r_p), the dual residuals (r_d), the gap residual (r_g), and the complementary gap $(x^\top s + \tau\kappa)$ all are reduced strictly if $\alpha^k > 0$ and at the same rate. This shows that $(x^k, \tau^k, y^k, s^k, \kappa^k)$ generated by the algorithm converges towards an optimal solution to (3) (and the termination criteria in step 4) is ultimately reached). In principle the initial point and the stepsize α^k should be chosen such that

$$
\min_j (x_j^k s_j^k, \tau^k\kappa^k) \ge \beta\mu^k, \quad \text{for } k = 0, 1, \ldots,
$$

is satisfied for some $\beta \in (0, 1)$ because this guarantees $(x^k, \tau^k, y^k, s^k, \kappa^k)$ converges towards a strictly complementary solution. Finally, it is possible to prove that the algorithm has the complexity $O(n^{3.5}L)$ given an appropriate choice of the starting point and the algorithmic parameters.

Further details about the homogeneous algorithm can be seen in [3,5]. Issues related to implementing the homogeneous algorithm are discussed in [1,4].

See also

- ▶ Entropy Optimization: Interior Point Methods
- ▶ Interior Point Methods for Semidefinite Programming
- ▶ Linear Programming: Interior Point Methods

- ▶ Linear Programming: Karmarkar Projective Algorithm
- ▶ Potential Reduction Methods for Linear Programming
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Andersen ED, Andersen KD (2000) The MOSEK interior point optimizer for linear programming: An implementation of the homogeneous algorithm. In: Frenk H, Roos K, Terlaky T, Zhang S (eds) High Performance Optimization. Kluwer, Dordrecht, pp 197–232
2. Goldman AJ, Tucker AW (1956) Theory of linear programming. In: Kuhn HW, Tucker AW (eds) Linear Inequalities and related Systems. Princeton Univ. Press, Princeton, pp 53–97
3. Roos C, Terlaky T, Vial J-P (1997) Theory and algorithms for linear optimization: An interior point approach. Wiley, New York
4. Xu X, Hung P-F, Ye Y (1996) A simplified homogeneous and self-dual linear programming algorithm and its implementation. Ann Oper Res 62:151–171
5. Ye Y (1997) Interior point algorithms: theory and analysis. Wiley, New York
6. Ye Y, Todd MJ, Mizuno S (1994) An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm. Math Oper Res 19:53–67

Hyperplane Arrangements

PETER ORLIK
Department Math., University Wisconsin, Madison, USA

MSC2000: 52C35, 05B35, 57N65, 20F36, 20F55

Article Outline

Keywords
Some Examples
Combinatorics
Divisor
Complement
Ball Quotients
Logarithmic Forms
Hypergeometric Integrals
See also
References

Keywords

Hyperplane arrangement; Geometric semilattice; Orlik–Solomon algebra; Divisor; Singularity; Complement; Homotopy type; Poincaré polynomial; Ball quotient; Logarithmic form; Hypergeometric integral

Let V be an ℓ-dimensional affine space over the field $\mathbf{K}$. An *arrangement of hyperplanes*, $\mathcal{A}$, is a finite collection of codimension one affine subspaces in V, [5].

Some Examples

1) A subset of the coordinate hyperplanes is called a *Boolean arrangement*.
2) An arrangement is in *general position* if at each point it is locally Boolean.
3) The *braid arrangement* consists of the hyperplanes $\{x_i = x_j : 1 \le i < j \le \ell\}$. It is the set of reflecting hyperplanes of the symmetric group on ℓ letters.
4) The reflecting hyperplanes of a finite reflection group is a *reflection arrangement*.

Combinatorics

An edge X of $\mathcal{A}$ is a nonempty intersection of elements of $\mathcal{A}$. Let $L(\mathcal{A})$ be the set of edges partially ordered by reverse inclusion. Then L is a *geometric semilattice* with minimal element V, rank given by codimension, and maximal elements of the same rank, $r(\mathcal{A})$. The *Moebius function* on L is defined by $\mu(V) = 1$ and for $X > V$, $\sum_{V \le Y \le X} \mu(Y) = 0$. The *characteristic polynomial* of $\mathcal{A}$ is $\chi(\mathcal{A}, t) = \sum_{X \in L} \mu(X) t^{\dim X}$. The β-invariant of $\mathcal{A}$ is $\beta(\mathcal{A}) = (-1)^{r(\mathcal{A})} \chi(\mathcal{A}, 1)$. For a generic arrangement of n hyperplanes $\chi(\mathcal{A}, t) = \sum_{k=0}^{r(\mathcal{A})} (-1)^k \binom{n}{k} t^{\ell - k}$. For the braid arrangement $\chi(\mathcal{A}, t) = t(t-1)(t-2)\cdots(t-(\ell-1))$. Similar factorizations hold for all reflection arrangements involving the (co)exponents of the reflection group. Given a p-tuple of hyperplanes, $S = (H_1, \ldots, H_p)$, let $\cap S = H_1 \cap \cdots \cap H_p$ and note that $\cap S$ may be empty. We say that S is *dependent* if $\cap S \neq \emptyset$ and $\operatorname{codim}(\cap S) < |S|$. Let $E(\mathcal{A})$ be the exterior algebra on symbols (H) for $H \in \mathcal{A}$ where product is juxtaposition. Define $\partial : E \to E$ by $\partial 1 = 0$, $\partial(H) = 1$ and for $p \ge 2$, $\partial(H_1 \cdots H_p) = \sum_{k=1}^{p} (-1)^{k-1} (H_1 \cdots \widehat{H_k} \cdots H_p)$. Let $I(\mathcal{A})$ be the ideal of $E(\mathcal{A})$ generated by $\{S : \cap S = \emptyset\} \cup \{\partial S : S \text{ is dependent}\}$. The *Orlik–Solomon algebra* of $\mathcal{A}$ is

$A(\mathcal{A}) = E(\mathcal{A})/I(\mathcal{A})$. See also connections with *matroid theory* [3].

Divisor

The *divisor* of $\mathcal{A}$ is the union of the hyperplanes, $N(\mathcal{A})$. If $\mathbf{K} = \mathbf{R}$ or $\mathbf{K} = \mathbf{C}$, then N has the *homotopy type* of a wedge of $\beta(\mathcal{A})$ spheres of dimension $r(\mathcal{A}) - 1$, [4]. The *singularities* of N are not isolated. The divisor of a general position arrangement has normal crossings, but this is not true for arbitrary $\mathcal{A}$. Blowing up N along all edges where it is not locally a product of arrangements yields a normal crossing divisor.

Complement

The *complement* of $\mathcal{A}$ is $M(\mathcal{A}) = V - N(\mathcal{A})$.

1) If $\mathbf{K} = \mathbf{F}_q$, then M is a finite set of cardinality $|M| = \chi(\mathcal{A}, q)$.
2) If $\mathbf{K} = \mathbf{R}$, then M is a disjoint union of open convex sets (*chambers*) of cardinality $(-1)^\ell \chi(\mathcal{A}, -1)$. If $r(\mathcal{A}) = \ell$, M contains $\beta(\mathcal{A})$ chambers with compact closure, [7].
3) If $\mathbf{K} = \mathbf{C}$, then M is an open complex (Stein) *manifold* of the homotopy type of a finite CW complex.

Its *cohomology* is torsion-free and its *Poincaré polynomial* is $\mathrm{Poin}(M, t) = (-t)^\ell \chi(\mathcal{A}, -t^{-1})$. The product structure is determined by the isomorphism of graded algebras $H^*(M) \simeq A(\mathcal{A})$. The *fundamental group* of M has an effective presentation but the *higher homotopy groups* of M are not known in general. The complement of a Boolean arrangement is a complex torus. In a general position arrangement of $n > \ell$ hyperplanes M has nontrivial higher homotopy groups. For the braid arrangement, M is called the pure braid space and its higher homotopy groups are trivial. The symmetric group acts freely on M with orbit space the braid space whose fundamental group is the *braid group*. The quotient of the divisor by the symmetric group is called the *discriminant*, which has connections with singularity theory.

Ball Quotients

Examples of algebraic surfaces whose *universal cover* is the complex ball were constructed as 'Kummer' covers of the projective plane branched along certain arrangements of projective lines, [2].

Logarithmic Forms

For $H \in \mathcal{A}$ choose a linear polynomial α_H with $H = \ker \alpha_H$ and let $Q(\mathcal{A}) = \prod_{H \in \mathcal{A}} \alpha_H$. Let $\Omega^p[V]$ denote all global regular (i. e., polynomial) p-forms on V. Let $\Omega^p(V)$ denote the space of all global rational p-forms on V. The space $\Omega^p(\mathcal{A})$ of logarithmic p-forms with poles along $\mathcal{A}$ is

$$\Omega^p(\mathcal{A}) = \{\omega \in \Omega^p(V) \colon\ Q\omega \in \Omega^p[V], \\ Q(d\omega) \in \Omega^{p+1}[V]\}\,.$$

The arrangement is free if $\Omega^1(\mathcal{A})$ is a free module over the polynomial ring. A *free arrangement* $\mathcal{A}$ has integer exponents $\{b_1, \ldots, b^\ell\}$ so that $\chi(\mathcal{A}, t) = \prod_{k=1}^{\ell}(t - b_k)$. Reflection arrangements are free. This explains the factorization of their characteristic polynomials.

Hypergeometric Integrals

Certain rank one *local system cohomology* groups of M may be identified with spaces of *hypergeometric integrals*, [1]. If the local system is suitably generic, these cohomology groups may be computed using the algebra $A(\mathcal{A})$. Only the top cohomology group is nonzero and it has dimension $\beta(\mathcal{A})$. See [6] for connections with the representation theory of *Lie algebras* and *quantum groups*, and with the *Knizhnik–Zamolodchikov differential equations* of physics.

See also

► Hyperplane Arrangements in Optimization

References

1. Aomoto K, Kita M (1994) Hypergemetric functions. Springer, Berlin
2. Barthel G, Hirzebruch F, Höfer T (1987) Geradenkonfigurationen und Algebraische Flächen. Vieweg, Braunschweig/Wiesbaden
3. Björner A, Las Vergnas M, Sturmfels B, White N, Ziegler GM (1993) Oriented matroids. Cambridge Univ. Press, Cambridge
4. Goresky M, MacPherson R (1988) Stratified Morse theory. Springer, Berlin
5. Orlik P, Terao H (1992) Arrangements of hyperplanes. Springer, Berlin
6. Varchenko A (1995) Multidimensional hypergeometric functions and representation theory of Lie algebras and quantum groups. World Sci., Singapore

7. Zaslavsky T (1975) Facing up to arrangements: Face-count formulas for partitions of space by hyperplanes. Memoirs Amer Math Soc 154

Hyperplane Arrangements in Optimization

PANOS M. PARDALOS
Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 05B35, 20F36, 20F55, 52C35, 57N65

Article Outline

Keywords
See also
References

Keywords

Hyperplane arrangement; Polynomial time algorithm

A finite set S of hyperplanes in $\mathbf{R}^d$ defines a dissection of $\mathbf{R}^d$ into connected sets of various dimensions. We call this dissection the *arrangement* $A(S)$ of S.

Given a vector $\eta = (\eta_1, \ldots, \eta_d) \in \mathbf{R}^d - \{0\}$ and a number $\eta_0 \in \mathbf{R}$, we may define a hyperplane H and associated halfspaces H^-, H^+ by

$$H = \left\{x \in \mathbf{R}^d : \ \eta \cdot x = \eta_0\right\},$$
$$H^- = \left\{x \in \mathbf{R}^d : \ \eta \cdot x < \eta_0\right\},$$
$$H^+ = \left\{x \in \mathbf{R}^d : \ \eta \cdot x > \eta_0\right\}.$$

Clearly, H, H^-, H^+ are disjoint and $H \cup H^- \cup H^+ = \mathbf{R}^d$.

We may now specify the location of a point relative to the set of hyperplanes $S = \{H_1, \ldots, H_n\}$. For a point p and $1 \le j \le n$, define

$$s_j(p) = \begin{cases} -1 & \text{if } p \in H_j^-, \\ 0 & \text{if } p \in H_j, \\ +1 & \text{if } p \in H_j^+. \end{cases}$$

The vector $s(p) = (s_1(p), \ldots, s_n(p))$ is called the *position vector* of p.

Clearly there are at most 3^n possible position vectors, however, in general most of these will not occur. We say that points p and *qlie on the same face* if $s(p) = s(q)$. The nonempty set of points with position vector r is called the *face* $f(r)$:

$$f(r) = \left\{p \in \mathbf{R}^d : \ s(p) = r\right\}$$

The nonempty sets of this form are called the *faces of the arrangement* $A(S)$. The position vector of a face $f(r) = g$ is defined to be r,

$$s(f(r)) = r.$$

A face f is called a k-face if its dimension is k. Special names are used to denote k-faces for special values of k: a 0-face is called a *vertex*, a 1-face is called an *edge*, a $(d-1)$-face is called a *facet*, and a d-face is called a *cell*. A face is said to be a *subface* of another face g if the dimension of f is one less than the dimension of g and f is contained in the boundary of g; it follows that $s_i(f) = 0$ unless $s_i(f) = s_i(g)$ for $1 \le i \le n$. If f is a subface of g, then we also say that f and g are *incident* (upon each other) or that they define an *incidence*.

An arrangement $A(S)$ of $n \ge d$ hyperplanes is called *simple* if any d hyperplanes of S have a unique point in common and if any $d + 1$ hyperplanes have no point in common. If $n < d$, we say that $A(S)$ is simple if the common intersection of the n hyperplanes is a $(d-n)$-flat. For more details see [3,4] and [5].

As an application of hyperplane arrangements in algorithm design for optimization problems, see [1].In it the problem of minimizing the Euclidean distance function on $\mathbf{R}^n$ subject to m equality constraints and upper and lower bounds (box constraints) is considered. A parametric characterization in $\mathbf{R}^m$ of the family of solutions to this problem is provided, thereby showing equivalence with a problem of search in an arrangement of hyperplanes in $\mathbf{R}^m$. This characterization and the technique for constructing arrangements due to H. Edelsbrunner, J. O'Rourke and R. Seidel are used to develop an exact algorithm for the problem. The algo-

rithm is strongly polynomial running in time $\Theta(n^m)$ for each fixed m.

See also

► Hyperplane Arrangements

References

1. Berman P, Kovoor N, Pardalos PM (1993) Algorithms for the least distance problem. Complexity in Numerical Optimization. World Sci., Singapore, pp 33–56
2. Chazelle B, Guibas J, Lee DT (1985) The power of geometric duality. Proc. 15th Annual ACM Symp. Theory of Computing. ACM, New York, pp 217–225
3. Edelsbrunner H (1987) Algorithms in combinatorial geometry. Springer, Berlin
4. Edelsbrunner H, O'Rourke J, Seidel R (1986) Constructing arrangements of lines and hyperplanes with applications. SIAM J Comput 15:341–363
5. Orlik P, Terao H (1992) Arrangements of hyperplanes. Springer, Berlin
6. Pardalos PM, Kovoor N (1990) An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. Math Program 46:321–328

Identification Methods for Reaction Kinetics and Transport

André Bardow, Wolfgang Marquardt
AVT – Process Systems Engineering,
RWTH Aachen University, Aachen, Germany

MSC2000: 34A55, 35R30, 62G05, 62G08, 62P30, 62P10, 62J02, 62K05, 76R50, 80A23, 80A30, 80A20

Article Outline

Introduction
Methods and Applications
Model-Based Experimental Analysis
Incremental vs. Simultaneous Model Identification
Case Studies
References

Introduction

Kinetic phenomena drive the macroscopic behavior of biological, chemical, and physical systems. The lack of mechanistic understanding of these kinetic phenomena is still the major bottleneck for a more widespread application of model-based techniques in process design, optimization, and control. In recent years, kinetic phenomena have become of increasing importance given the rapidly developing capabilities for the numerical treatment of more complex models on the one hand and the need for predictive models on the other.

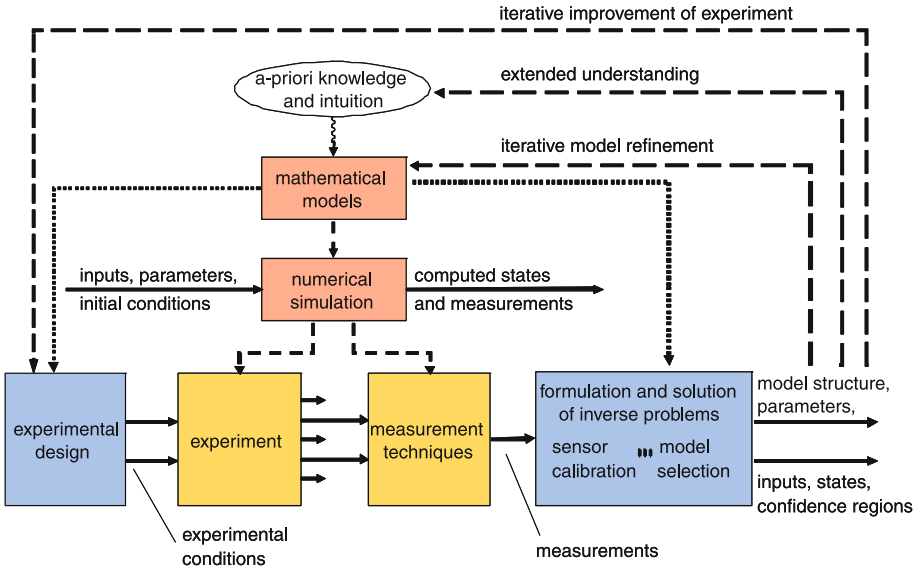
Despite this demand, kinetic modeling of process systems is still a challenge. This contribution presents systematic work processes to derive and validate models that capture the underlying physicochemical mechanisms of an observed behavior. The work process of *model-based experimental analysis* (or MEXA for short) is introduced in the next section. The key factor in the procedure is an *incremental strategy for model structure refinement* tailored for the identification of reaction kinetics and transport phenomena [30]. While identification of kinetic models from experimental data can, in principle, be performed by application of standard statistical tools of nonlinear regression [2] and model discrimination [39], this direct approach in general leads to a large number of NLP or even MINLP problems being solved [16,21,34,37] that may be computationally prohibitive and in particular does not reflect the underlying physics. In contrast, the incremental identification approach discussed here presents a physically motivated and adapted divide-and-conquer strategy to the complex optimization problem of kinetic model identification. Applications of this approach in the areas of (bio)chemical reactions [6,12,13,15,32], multicomponent diffusion [3,5], and heat transfer in fluid flow [22,25] are discussed.

Methods and Applications

Model-Based Experimental Analysis

The typical work flow of the MEXA procedure is as follows (Fig. 1):

1. An initial *experiment* with a suitable measurement system is designed on the basis of a priori knowledge and intuition.
2. A first *mathematical model* of experiment and measurement system is proposed.
3. *Numerical simulation* studies are performed to explore the expected behavior of the experiment.
4. The model is then employed for rigorous *experimental design* [41] to gain maximum information with respect to the goal of the investigation.



Identification Methods for Reaction Kinetics and Transport, Figure 1
Model-based experimental analysis [30]

5. The designed *experiment* is performed and at least some of the variables of interest are observed using appropriate *measurement techniques*.
6. *Formulation and solution of inverse problems* refers to combinations of state, parameter, and unknown input estimation as well as model structure identification and selection.
7. Typically, the first model does not reflect the studied phenomena with sufficient detail and accuracy. Therefore, *iterative model refinement*, intertwined with iterative improvement of the experimental and measurement techniques, must be carried out to improve the predictive capabilities of the model based on the extended understanding gained.

Work processes consisting of the steps design of experiments, data analysis, and modeling date back to at least the 1970s [26]. However, the development and benchmarking of such work processes has only recently been formulated as an important research objective, e. g., by the Collaborative Research Center CRC 540 "Model-based Experimental Analysis in Fluid Multi-Phase Reactive Systems" (http://www.sfb540.rwth-aachen.de/) at RWTH Aachen University as well as by Asprey and Macchietto [1]. The power of these work processes depends on the specific strategies employed for systematically improving both the model structure and the experimental setup in every refinement step during model identification. While experimental design is the focus of the work of Asprey and Macchietto [1], the research in CRC 540 is complementary and emphasizes the strategy for model structure refinement as discussed in what follows.

Incremental vs. Simultaneous Model Identification

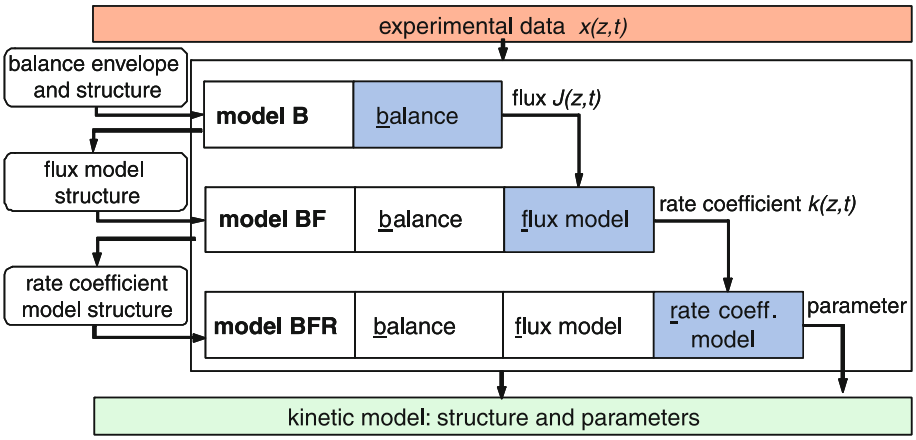
Incremental Modeling and Identification The key idea of the incremental approach for model structure refinement is to follow the incremental steps of systematic *model development* [29] also in model *identification* (Fig. 2).

Therefore, the main steps of model development and their connection to incremental identification are outlined next.

Model B In model development, balance envelopes and their interactions are determined first, the spatiotemporal resolution of the model is decided, and the extensive quantities x to be balanced are selected. The balance equation is formulated as a sum of generalized fluxes, e. g.,

$$\frac{\partial x}{\partial t} = -\nabla \cdot J_\mathrm{f} + J_\mathrm{s} \,, \tag{1}$$

$$\frac{\mathrm{d}x}{\mathrm{d}t} = A(x) + B(x)w \,. \tag{2}$$



Identification Methods for Reaction Kinetics and Transport, Figure 2
Incremental modeling and identification [30]

Equation (1) exemplifies a balance for a distributed quantity x flowing with the flux J_f and being generated/consumed according to the source/sink term J_s. Note that further generalized fluxes may arise through initial or boundary conditions. A lumped quantity is balanced in Eq. (2), where A,B are matrix functions of appropriate dimensions describing, e. g., inter- and intraphase transport and source/sink terms. Note that no constitutive equations are considered yet to specify the generalized fluxes $J(\cdot)$[1] (here: J_f, J_s, w) as a function of the intensive thermodynamic state variables.

In incremental model identification, the unknown generalized fluxes $J(\cdot)$ are estimated directly from the balance equation. For this purpose, measurements of the states $x(\cdot)$ with sufficient resolution in time t and/or space z are assumed. The unknown flux $J(\cdot)$ in the balance equation is then determined as a function of time and space coordinates – without the need for specifying a constitutive equation.

Model BF In model development, constitutive equations are specified for each flux term in the balances on the next decision level:

$$J(\cdot) = J(x(\cdot), \nabla x(\cdot), \ldots, k(\cdot)) \,. \tag{3}$$

This could be, e. g., correlations for interfacial fluxes or reaction rates.

[1]The (·)-argument summarizes the spatial and/or respective temporal dependency of the quantity.

Similarly, in incremental model identification on *level BF*, flux model candidates (3) are selected or generated to relate the flux to rate coefficients, to measured states, and to their derivatives. The flux estimates obtained on *level B* are now interpreted as inferential measurements. These can then be used, together with the real measurements, to determine a rate coefficient $k(\cdot)$ as a function of time and space. Often, the flux model can directly be solved for the rate coefficient function $k(\cdot)$.

Model BFR In model development, the rate coefficients introduced in the correlations on the *level BF* – such as a reaction rate or heat and mass transfer coefficients – often themselves depend on the states. Consequently, a model relating rate coefficients and states has to be chosen on yet another *level BFR*

$$k(\cdot) = k(x(\cdot), \nabla x(\cdot), \ldots, \theta) \,. \tag{4}$$

This cascaded decision process can continue as long as the submodels considered involve not only constant parameters θ but also functions of the states.

Mirroring this step in incremental model identification, a model for the rate coefficients is identified. This model (4) is assumed to only depend on the measured states and constant parameters. These parameters θ can be computed from the estimated rate coefficients $k(\cdot)$ and the measured states $x(\cdot)$ by solving an algebraic regression problem.

Such a structured approach during model identification renders the individual decisions in the modeling and identification process completely transparent: the modeler is in full control of the model refinement process.

Simultaneous Model Identification In the previous section, it was shown that there exists a natural hierarchy in models of kinetic phenomena. Classic approaches to model identification, however, neglect this inherent structure. These *simultaneous* approaches assume that the model structure is correct and consider only the fully specified model (Fig. 2). Models for the flux expression (*Model BF*) and the phenomenological coefficients (*Model BFR*) have to be specified a priori.

In practical situations, these models are initially uncertain. Now, all assumptions on the process will *simultaneously* influence the results of the model identification procedure. The estimates may be biased if the parameter estimation is based on a model containing structural errors [42]. The theoretically optimal properties of a maximum likelihood approach [2] are therefore lost in the presence of structural model mismatch. Initialization and convergence may be difficult since the whole problem is solved in one step [18]. More importantly, it may be difficult in a simultaneous approach to identify which part of the model introduced the error.

Furthermore, several candidate model structures may exist for each kinetic phenomenon. The aggregation of such submodels with the balance equations will inevitably lead to a multitude of candidate models. Alternatively, general approximation schemes like neural nets can be used, often leading to several hundred unknown parameters. Both approaches may be prohibitive due to computational cost, especially when more complex or even distributed parameter systems are considered.

Discussion of Identification Approaches The incremental approach splits the identification procedure into a sequence of inverse problems, thereby reducing uncertainty and computational complexity. It thus has the potential to overcome a number of the disadvantages of the simultaneous approach:

- *Avoid combinatorial complexity:* Rather than postulating large numbers of nested model structures, a structured, fully transparent process is used in the incremental model refinement strategy. An uncontrolled combinatorial growth of the number of model candidates is avoided.
- *Reduce uncertainty:* In the incremental approach, any decision on the model structure relates to a single physicochemical phenomenon. Submodel selection is guided by the previous estimation step, which provides input–output data inferred from the measurements. Identifiability can also be assessed more easily on the level of the submodel.
- *Computational advantages:* The decomposition inherent in incremental model refinement avoids the solution of many difficult output least-squares problems with (partial-)differential-algebraic constraints and potentially large data sets. Rather, an often *linear* inverse problem must be solved first. All the following problems are nonlinear regression problems with algebraic constraints – regardless of the complexity of the overall model. This decomposition not only facilitates initialization and convergence, but it also allows for incremental testing of model validity at every decision level for the submodels. Largely intractable estimation problems may become computationally feasible.

Still, it should be kept in mind that the incremental and the simultaneous methods were derived for different purposes: the incremental approach is aimed at gross elimination of candidate models and/or systematic derivation of suitable candidate *model structures*, whereas the simultaneous approach gives the best *parameter estimates* once the correct model structure is known [6].

Multistep approaches to model identification have been applied rather intuitively in the past. The sequence of flux estimation and parameter regression is, e. g., commonly employed in reaction kinetics as the so-called differential method [19]. More recently, a two-step approach has been applied for the hybrid modeling of fermentation processes [36,38]. First reaction fluxes are estimated from measured data, then neural networks and fuzzy models are employed to correlate the fluxes with the measurements. Mahoney et al. [28] estimate the crystal growth rate directly from the population balance equations using a method of characteristics approach and indicate the possibility of correlating it with solute concentration next.

Though the incremental refinement approach is rather intuitive, a successful implementation requires tailored ingredients such as

- high-resolution field measurement techniques for state variables,
- algorithms for model-free flux estimation by inversion of the balance equations,
- methodologies for the generation, assessment, and selection of the most suitable model structures, and
- model-based experimental design methods.

A detailed discussion of these areas in relation to incremental model identification can be found in [30]. Various aspects are highlighted in the following case studies. Here, the progress in the development of the incremental model identification approach is reported for challenging kinetic modeling problems of gradually increasing complexity from (bio)chemical reactions to diffusion in liquids and to heat transfer at falling liquid films. In addition, the incremental approach has been successful in the identification of hybrid process models [24].

Case Studies

(Bio)chemical Reaction Kinetics The identification of the mechanism and kinetics of chemical reactions is one of the most relevant and still not yet fully satisfactorily solved tasks in process systems modeling [8]. In biological systems, the situation is often even more severe due to the complexity of living systems. The incremental identification approach has been applied for a variety of reaction systems [6,12,13,15,32]. Here, selected features are discussed to elucidate the general properties of this problem class.

Model B: *Reaction flux estimation in lumped systems* For illustration, we assume a well-mixed and isothermal homogenous reaction system. The balance equation for the mole number n_i of species i is then

$$\frac{\mathrm{d}n_i}{\mathrm{d}t} = f_i^{\mathrm{in}} - f_i^{\mathrm{out}} + f_i^{\mathrm{r}}\,, \quad i = 1, \ldots, n_c\,, \tag{5}$$

where f_i^{in}, f_i^{out} are, respectively, the molar flow rates into and out of the reactor and f_i^{r} is the unknown reaction flux of species i. It is worth noting that the fluxes enter the balance equations linearly and the equations are decoupled for each species.

All reaction fluxes f_i^{r} can thus be estimated individually by numerical differentiation of concentration data for each measured species on *level B* from material balances only. Tikhonov-Arsenin filtering [31] or smoothing splines [6] with regularization parameter choice based on the L-curve or generalized cross-validation have been shown to give reliable estimates.

Model BF: *Estimation of reaction rates and stoichiometry* If the reaction stoichiometry is unknown, target factor analysis (TFA) [11] is used to test possible stoichiometries and to determine the number of relevant reactions. The reaction rates $r(t)$ can then be calculated from the typically nonsquare linear equation system relating reaction fluxes $f_i^{\mathrm{r}}(t)$ and rates by the stoichiometric matrix N:

$$f^{\mathrm{r}} = v(t) N^T r(t)\,, \tag{6}$$

with $v(t)$ denoting the reactor volume.

Model BFR: *Estimation of kinetic coefficients* On the next level, concentrations are determined either from smoothed measurements using nonparametric methods [40] or unmeasured concentrations are reconstructed from stoichiometry and mass balances [13]. Since a complete set of concentration and rate data is now available, candidate reaction rate laws of the general form

$$r(t) = m(c(t), \theta) \tag{7}$$

can now be discriminated by nonlinear algebraic regression [42].

Model identification may not immediately result in reliable model structures and parameters because of a lack of information content in the data. Iterative improvement with optimally chosen experimental conditions as suggested in the MEXA work process can then be employed [13].

The incremental identification method has been worked out for arbitrary reaction schemes with reversible or irreversible as well as dependent or independent reactions. The minimum type of concentration measurements required to guarantee identifiability has been assessed theoretically.

The incremental identification strategy has been used in a benchmark study considering a homogeneous reaction system [13]. Computational effort for model

identification could be reduced by almost two orders of magnitude using the structured search in the incremental method. The inclusion of data-driven model substructures in hybrid models is straightforward, as already exemplified for neural networks [15] and sparse grids [14]. The basic framework can easily be extended to nonisothermal systems, and even multiphase transport has been considered [12,23]. An application study to a biochemical reaction system is presented in [32].

Multicomponent Diffusion While phase equilibrium models are available even for complex multicomponent mixtures [17], there is a lack of experimentally validated diffusion models in particular for multicomponent liquid mixtures [9]. The incremental identification of diffusive mass transport models is therefore outlined in this section. The application is based on the recently introduced Raman diffusion experiment [4,7]. Here, one-dimensional interdiffusion of two initially layered liquid mixtures is observed by 1D-Raman spectroscopy. Concentration profiles c_i of all species are obtained with high resolution in time and space [20].

Model B: *Estimation of 1D-diffusion fluxes* For the 1D-diffusion process, the mass balance equation for each species i can be given as

$$\frac{\partial c_i}{\partial t} = -\frac{\partial J_i}{\partial z}\,, \quad i = 1, \ldots, n_c - 1\,. \tag{8}$$

The determination of the diffusive flux J_i falls into the class of interior flux estimation in distributed parameter systems [30]. While interior fluxes cannot be determined in 2D or 3D situations without specification of a constitutive model, the model-free flux estimation is possible in the one-dimensional situation considered here. Only one nonzero mass flux component has to be determined from differentiated concentrations measured along a line in the direction of the diffusive flux. Such a strategy has been followed in [3,5].

The Raman concentration measurements were first differentiated with respect to time by means of spline smoothing [33] and subsequently integrated over the spatial coordinate to render a diffusive flux estimate without specifying a diffusion model:

$$J_i(z,t) = -\int_0^z \frac{\partial c_i(\xi,t)}{\partial t}\, d\xi\,. \tag{9}$$

This technique directly carries over to multicomponent diffusion [5] provided concentration measurements are available for every species. In particular, there is only a linear increase in complexity due to the natural decoupling of the multicomponent material balances (8).

Model BF: *Estimation of diffusion flux models* A flux model has to be introduced on the next level. For example, generalized Fick or Maxwell–Stefan models could be selected as candidates. In case of binary mixtures, the Fick diffusion coefficient can, e. g., be determined at any point in time and space:

$$D(z,t) = -\frac{J(z,t)}{\partial c(z,t)/\partial z}\,. \tag{10}$$

Positivity requirements may now be used, e. g., to assess model assumptions on this level.

Model BFR: *Estimation of model parameters* The estimated diffusion coefficient data can now be correlated with the measured concentrations to obtain a diffusion model:

$$D(z,t) = m(c(z,t),\theta)\,. \tag{11}$$

Error-in-variables methods [10] and statistical model discrimination techniques [35] are employed to decide on the most appropriate model for the concentration dependence of the diffusion coefficient. This concentration dependency has been shown to be even identifiable from a single Raman diffusion experiment [3]. An application in food science has recently been presented in [27].

In case of multicomponent diffusion, the last two levels have to be merged because all species concentration gradients are determined by the diffusive flux of any species due to the cross effects of multicomponent diffusion [3,5]. The merged steps BF and BFR then allow for efficient initialization of these complex estimation problems.

Heat Transfer at Falling Films Liquid falling films are a challenging benchmark problem for general fluid multiphase reaction systems as they show all the relevant features of this problem class. Here, the first steps in the application of the incremental approach to heat transfer in falling films are considered [22,25].

Model B: *Boundary flux estimation in distributed systems* In order to study its heat transfer character-

istics, a laminar-wavy falling film is heated by resistance heating using a supporting wall as heater. Infrared thermography is employed to measure a transient 2D temperature field on the backside of the wall. An inverse heat conduction problem for the three-dimensional wall has to be solved to determine the boundary heat flux between the wall and the falling film as the first step of the incremental approach:

$$\frac{\partial T}{\partial t} = a\Delta T\,, \tag{12}$$

$$-\lambda\nabla T|_{\Gamma} = w(z_{\Gamma}, t)\,, \tag{13}$$

$$-\lambda\nabla T|_{\Lambda} = q(z_{\Lambda}, t)\,, \tag{14}$$

with Γ and Λ being the parts of the surface with unknown and with known boundary heat fluxes $w(z_{\Gamma}, t)$ and $q(z_{\Lambda}, t)$, respectively. The boundary flux estimation problem is solved by means of a multigrid finite-element discretization of the heat conduction Eq. (12) in conjunction with the conjugate gradient method. Gradient computation is performed using the adjoint method. This framework allows for the solution of the discretized problem involving about three million variables on a desktop computer [22].

These results show that the identification of kinetic phenomena may become feasible even in complex flow problems using the structured search strategy of the incremental approach. A generalization of the presented problem to work out the full incremental identification concept for heat transfer problems in falling films is currently in progress [25].

References

1. Asprey SP, Macchietto S (2000) Statistical tools for optimal dynamic model building. Comput Chem Eng 24(2–7):1261–1267
2. Bard Y (1974) Nonlinear Parameter Estimation. Academic, New York
3. Bardow A, Göke V, Koß HJ, Lucas K, Marquardt W (2005) Concentration-dependent diffusion coefficients from a single experiment using model-based Raman spectroscopy. Fluid Phase Equilib 228:357–366
4. Bardow A, Göke V, Koß HJ, Marquardt W (2006) Ternary diffusivities by model-based analysis of Raman spectroscopy measurements. AIChE J 52(12):4004–4015
5. Bardow A, Marquardt W (2004) Identification of diffusive transport by means of an incremental approach. Comput Chem Eng 28(5):585–595
6. Bardow A, Marquardt W (2004) Incremental and simultaneous identification of reaction kinetics: methods and comparison. Chem Eng Sci 59(13):2673–2684
7. Bardow A, Marquardt W, Göke V, Koß HJ, Lucas K (2003) Model-based measurement of diffusion using Raman spectroscopy. AIChE J 49(2):323–334
8. Berger R, Stitt E, Marin G, Kapteijn F, Moulijn J (2001) Eurokin – chemical reaction kinetics in practice. CATTECH 5(1):30–60
9. Bird RB (2004) Five decades of transport phenomena. AIChE J 50(2):273–287
10. Boggs PT, Byrd RH, Rogers J, Schnabel RB (1992) User's reference guide for ODRPACK version 2.01 – software for weighted orthogonal distance regression. Technical Report NISTIR 92-4834. US Department of Commerce, National Institute of Standards and Technology
11. Bonvin D, Rippin DWT (1990) Target factor-analysis for the identification of stoichiometric models. Chem Eng Sci 45(12):3417–3426
12. Brendel M (2006) Incremental identification of complex reaction systems. PhD thesis, RWTH Aachen University
13. Brendel M, Bonvin D, Marquardt W (2006) Incremental identification of kinetic models for homogeneous reaction systems. Chem Eng Sci 61(16):5404–5420
14. Brendel M, Marquardt W (2008) An algorithm for multivariate function estimation based on hierarchically refined sparse grids. Computing and Visualization in Science, doi:10.1007/s00791-008-0085-1
15. Brendel M, Mhamdi A, Bonvin D, Marquardt W (2004) An incremental approach for the identification of reactor kinetics. In: Allgöwer F, Gao F ADCHEM – 7th International Symposium on Advanced Control of Chemical Processes. Preprints vol I, pp 177–182, Hong Kong
16. Brink A, Westerlund T (1995) The joint problem of model structure determination and parameter-estimation in quantitative IR spectroscopy. Chemom Intell Lab Syst 29(1):29–36
17. Chen C-C, Mathias PM (2002) Applied thermodynamics for process modelling. AIChE J 48(2):194–200
18. Cheng ZM, Yuan WK (1997) Initial estimation of heat transfer and kinetic parameters of a wall-cooled fixed-bed reactor. Comput Chem Eng 21(5):511–519
19. Froment GF, Bischoff KB (1990) Chemical Reactor Analysis and Design. Wiley, New York
20. Göke V (2005) Messung von Diffusionskoeffizienten mittels eindimensionaler Raman-Spektroskopie. PhD thesis, RWTH Aachen University
21. Goodwin GC, Payne RL (1977) Dynamic system identification: experiment design and data analysis. Academic, New York
22. Gross S, Somers M, Mhamdi A, Al Sibai F , Reusken A, Marquardt W, Renz U (2005) Identification of boundary heat

fluxes in a falling film experiment using high resolution temperature measurements. Int J Heat Mass Tran 48(25–26):5549–5562
23. Kahrs O, Brendel M, Marquardt W (2005) Incremental identification of NARX models by sparse grid approximation. In: Proceedings of the 16th IFAC World Congress, 3–8 July 2005 Prague
24. Kahrs O, Marquardt W (2008) Incremental identification of hybrid process models. Comput Chem Eng 32(4–5):694–705
25. Karalashvili M, Groß S, Mhamdi A, Reusken A, Marquardt W (2007) Incremental identification of transport phenomena in wavy films. In: Plesu V, Agachi P (eds) 17th European Symposium on Computer Aided Process Engineering – ESCAPE17, Elsevier, Amsterdam (CD-ROM)
26. Kittrell J (1970) Mathematical modeling of chemical reactions. Adv Chem Eng 8:97–183
27. Lucas T, Bohuon P (2005) Model-free estimation of mass fluxes based on concentration profiles. I. Presentation of the method and of a sensitivity analysis. J Food Eng 70(2):129–137
28. Mahoney AW, Doyle FJ, Ramkrishna D (2002) Inverse problems in population balances: growth and nucleation from dynamic data. AIChE J 48(5):981–990
29. Marquardt W (1995) Towards a process modeling methodology. In: Berber R (eds) Methods of Model-Based Control. Kluwer, Dordrecht, pp 3–41
30. Marquardt W (2005) Model-based experimental analysis of kinetic phenomena in multi-phase reactive systems. Chem Eng Res Des 83(A6):561–573
31. Mhamdi A, Marquardt W (1999) An inversion approach to the estimation of reaction rates in chemical reactors. In: Proceedings of the European Control Conference (ECC'99), Karlsruhe, Germany, pp F1004–1
32. Michalik C, Schmidt T, Zavrel M, Ansorge-Schumacher M, Spiess A, Marquardt W (2007) Application of the incremental identification method to the formate dehydrogenase. Chem Eng Sci 62(18–20):5592–5597
33. Reinsch CH (1967) Smoothing by spline functions. Numer Math 10(3):177–183
34. Skrifvars H, Leyffer S, Westerlund T (1998) Comparison of certain MINLP algorithms when applied to a model structure determination and parameter estimation problem. Comput Chem Eng 22(12):1829–1835
35. Stewart WE, Shon Y, Box GEP (1998) Discrimination and goodness of fit of multiresponse mechanistic models. AIChE J 44(6):1404–1412
36. Tholudur A, Ramirez WF (1999) Neural-network modeling and optimization of induced foreign protein production. AIChE J 45(8):1660–1670
37. Vaia A, Sahinidis NV (2003) Simultaneous parameter estimation and model structure determination in FTIR spectroscopy by global MINLP optimization. Comput Chem Eng 27(6):763–779
38. van Lith PF, Betlem BHL, Roffel B (2002) A structured modeling approach for dynamic hybrid fuzzy-first principles models. J Process Control 12(5):605–615
39. Verheijen PJT (2003) Model selection: an overview of practices in chemical engineering. In: Asprey SP, Macchietto S Dynamic Model Development: Methods, Theory and Applications. Elsevier Science, Amsterdam, pp 85–104
40. Wahba G (1990) Spline Models for Observational Data. SIAM, Philadelphia
41. Walter E, Pronzato L (1990) Qualitative and quantitative experiment design for phenomenological models – a survey. Automatica 26(2):195–213
42. Walter E, Pronzato L (1997) Identification of Parametric Models from Experimental Data. Springer, Berlin

Ill-posed Variational Problems

IVP

ALEXANDER KAPLAN, RAINER TICHATSCHKE
Universität Trier, Trier, Germany

MSC2000: 65K05, 65M30, 65M32, 49M30, 49J40

Article Outline

Keywords
See also
References

Keywords

Ill-posed variational problem; Well-posedness; Tikhonov regularization; Proximal point methods

It is generally accepted that the notion'ill-posed problem' originates from a considered concept of *well-posedness*: A problem is called ill-posed if it is not well-posed. There are a lot of different notions of well-posedness (cf. [15,23,27,35,38] and [40]), which correspond to certain classes of variational problems and numerical methods and take into account the'quality' of the input data, in particular their exactness. For a comparison of different concepts of well-posedness see [12,15] and [35].

For instance, Tikhonov well-posedness [35,38] is convenient if we deal with methods generating feasible minimizing sequences, and it is not appropriate to analyse stability of exterior penalty methods.

We shall proceed from two concepts of well-posedness which are suitable for wide classes of problems and methods.

The first concept is destined to the problem

$$\min\{J(u):\ u \in K\}, \tag{1}$$

where K is a nonempty closed subset of a Banach space V with the norm $\|\cdot\|$ and $J: V \to \mathbf{R} \cup \{+\infty\}$ is a proper lower-semicontinuous functional.

Definition 1 (cf. [27]) The sequence $\{u^n\} \subset V$ is said to be a *generalized minimizing sequence* (*Levitin–Polyak minimizing sequence*) for (1) if

$$\lim_{n\to\infty} d(u^n, K) = 0$$

and

$$\lim_{n\to\infty} J(u^n) = \inf_{u\in K} J(u),$$

with

$$d(u, K) = \inf_{v\in K} \|u - v\|$$

distance function.

Definition 2 (1) is called *well-posed* (*Levitin–Polyak well-posed*) if

i) it is uniquely solvable, and

ii) any generalized minimizing sequence converges to $u^* = \arg\min\{J(u): u \in K\}$.

The second concept (cf. [20,23]) concerns (1) with

$$K = \{u \in U_0:\ B(u) \le 0\}, \tag{2}$$

$U_0 \subset V$ a convex closed set, $B: U_0 \to Y$ a convex continuous mapping into a Banach space Y, and $J: U_0 \to \mathbf{R}$ a convex continuous functional. The relation '$\le$' in (2) and the convexity of B are defined according to a positive cone in Y.

In this case, the study of the dependence of a solution on data perturbations is often more natural and simpler than the analysis of the convergence of a generalized minimizing sequence.

We suppose that U_0 is exactly given and a violation of the condition $u \in U_0$ does not arise. For a fixed $\delta > 0$, the set of variations is defined by

$$\Phi_\delta = \Big\{\varphi_\delta \equiv (J_\delta, B_\delta):\ \|J - J_\delta\|_{C(U_0)} \le \delta, \sup_{u\in U_0} \|B(u) - B_\delta(u)\|_Y \le \delta\Big\}, \tag{3}$$

where $J_\delta: U_0 \to \mathbf{R}$, $B_\delta: U_0 \to Y$ are assumed to be continuous. Then, the problem

$$\min\{J_\delta(u):\ u \in U_0,\quad B_\delta(u) \le 0\} \tag{4}$$

corresponds to an arbitrary but fixed variation $\varphi_\delta \in \Phi_\delta$. The set of optimal solutions of (4) will be denoted by $U^*(\varphi_\delta)$.

Definition 3 Problem (1), (2) is called *well-posed* if

i) it is uniquely solvable,

ii) there exists a constant $\delta_0 > 0$ such that for any $\delta \in (0, \delta_0)$ and any $\varphi_\delta \in \Phi_\delta$ the set $U^*(\varphi_\delta)$is nonempty,

iii) $\lim_{\delta\to 0} d(u^*, U^*(\varphi_\delta)) = 0$ for arbitrary $\varphi_\delta \in \Phi_\delta$.

Depending on the pecularities of the problem considered, the 'quality' of data as well as the requirements to an approximation, other norms in (3) and additional assumptions w.r.t. J_δ, B_δ can be considered (for instance, convexity of J_δ, B_δ). For a relaxation of the inequalities in (3) see [39,40].

Of course, the Definitions 2 and 3 are not equivalent, and in the framework of the chosen concept of well-posedness the problem is called *ill-posed* if any condition is violated in the corresponding definition used.

Example 4 Problem (1), (2) with

$$V = \mathbf{R}^3, \quad Y = \mathbf{R}^3,$$
$$J(u) = u_2,$$
$$U_0 = \{u \in \mathbf{R}^3:\ 0 \le u_k \le 2\},$$
$$B(u) = \left(u_1 + u_2 - \frac{1}{2}, u_3 - \frac{1}{2}, -u_1 - u_2 - u_3 + 1\right)^\top$$

is well-posed according to Definition 2, but it is ill-posed according to Definition 3. This example reflects, in particular, the situation that an arbitrary small data perturbation may lead to an unsolvable problem.

Example 5 The unconstrained problem

$$\text{minimize} \quad J(u) = \sum_{k=1}^{\infty} k^{-1} u_k^2 \quad \text{over} \quad V = l_2$$

is ill-posed according to both Definitions 2 and 3. To verify that take

$$u^n = (-1, 0, \dots, 0, \overset{n}{1}, 0 \dots), \quad \delta_n = \frac{1}{n},$$
$$J_{\delta_n} = \sum_{\substack{k=1,\\ k\neq n}}^{\infty} k^{-1} u_k^2 + \max\{n^{-1} u_n^2, n^{-1}\}.$$

If it is supposed that V is a reflexive Banach space, $Y = C(T)$ (with T a compact set), that Problem (1), (2) is uniquely solvable and that Slater's condition is valid, then the condition

$$\lim_{\delta \to +0} \sup_{u \in W_\delta} \|u - u^*\| = 0,$$

with

$$W_\delta = \{u \in U_0 : \ J(u) \le J(u^*) + \delta, \ \max_{t \in T} B(u)(t) \le \delta\}$$

is necessary and sufficient for this problem to be well-posed according to Definition 3 (with convex J_δ, B_δ) (cf. [20]).

Let us mention also concepts of well-posedness using different notions of hyper- or epiconvergence. As an example, identifying the functions with their epigraphs, in [9] for the class of Problem (1) the closeness of data is measured in the Attouch–Wets metric defined on the data space. Here, Problem (1) is said to be well-posed if it is uniquely solvable and its solution depends continuously (in V) on the data perturbation (for details see [35]). These concepts are closely related to the classical idea of Hadamard of the continuous dependence of the solution on the data.

Some notions of well-posedness do not suppose uniqueness of a solution of the problem considered (cf. [35,40]). A correspondinggeneralization of Definition 2 leads to the following conditions:

i) the optimal set U^* is nonempty,
ii) each generalized minimizing sequence has a subsequence converging to an element of U^*,
or (the weaker condition)
ii') $d(u^n, U^*) \to 0$ for each generalized minimizing sequence $\{u^n\}$.

If the problem is ill-posed, the following difficulties occur:

1) using approximate data one cannot be sure that a solution of the 'perturbed' problem is close to the solution (or to the solution set) of the original problem;
2) in the majority of the numerical methods it is possible that the calculated minimizing sequence does not converge (in a suitable sense) to a solution of the problem.

It may also happen that standard solution methods break down for such problems.

Example 6 Problem (1), (2) with

$$V = \mathbf{R}^2, \quad Y = C[0,1], \quad J(u) = -u_1,$$
$$U_0 = \left\{u \in \mathbf{R}^2 : \ u_2 \ge 0\right\},$$

and

$$Bu(t) = u_1 - \left(t - \frac{1}{\sqrt{2}}\right)^2 u_2.$$

Obviously, solutions of this linear semi-infinite problem are points $u^* \in U^* \equiv \{(0, a): a \ge 0\}$. Choosing a finite grid T' on $[\,0, 1]$ with

$$t_0 = \arg\min \left\{ \left| t - \frac{1}{\sqrt{2}} \right| : \ t \in T' \right\}$$

and $t_0 \neq 1/\sqrt{2}$, then for the approximate problem (with T' instead of $[0, 1]$), the ray

$$\left\{ u \in \mathbf{R}^2 : \ u_1 = \left(t_0 - \frac{1}{\sqrt{2}}\right)^2 u_2, \quad u_2 \ge 0 \right\}$$

is feasible and $J(u) \to -\infty$ on this ray if $\|u\| \to \infty$.

This example shows the typical behavior of ill-posed semi-infinite problems: Although the original problem is solvable, the discretized ones may be not solvable, even if dense grids are used. Due to unsolvability of the discretized problems, the direct application of discretization and exchange methods for solving semi-infinite programs is impossible. Moreover, the assumptions required for the application of reduction methods are violated in this example, too. (For the conceptual description of the methods mentioned see [16]).

Nevertheless, it is well-known that some classical methods, applied to ill-posed problems, possess stabilizing qualities: They generate minimizing sequences with better convergence properties than those properties which are guaranteed for an arbitrary minimizing sequence.

For instance, for the ill-posed problem (1), where J is a convex functional of the class $C^{1,1}$ and K is a convex closed subset of a Hilbert space V, the gradient projection method (with a constant steplength parameter and an inexact calculation of the gradient $p^k \approx \nabla J(u^k)$ at each step k) converges weakly to some element of U^* if $U^* \neq \emptyset$ and $\| p^k - \nabla J(u^k) \| \leq \epsilon_k$, $\sum_{k=1}^{\infty} \epsilon_k < \infty$ (cf. [33]).

In [20] it is shown that penalty methods applied to a finite-dimensional convex programming problem, for which the conditions ii), iii) in Definition 3 may be violated, converge to the unique solution of this problem if the exactness of the data is improved within the solution process by a special rule, depending on the change of the penalty parameter.

Stable methods for solving convex ill-posed variational problems are mainly based on *Tikhonov's regularization approach* (cf. [29,39,40]) and the *proximal point approach* (cf. [30,37]). Nowadays the direct application of these approaches (when multiple regularization of the original problem is performed and the regularized problems are solved with high accuracy) loses its importance in comparison with techniques using regularization inwards of the basic numerical algorithm which is suitably chosen for solving well-posed problems of the corresponding class of problems.

Let us briefly describe these techniques under the assumption that V is a Hilbert space. Suppose a certain basic method (for instance, discretization or penalization method) generates the sequence of auxiliary problems

$$J_i(u) \to \min, \quad u \in K_i \subset V,$$

then in the Tikhonov approach successively the auxiliary problems

$$\begin{aligned} &J_i(u) + \alpha_i \|u - \overline{u}\|^2 \to \min, \quad u \in K_i, \\ &(\alpha_i > 0, \quad \lim \alpha_i = 0, \quad \overline{u} \in V \text{ a fixed element}) \end{aligned} \tag{5}$$

are solved, whereas the proximal point approach leads to the following sequence of auxiliary problems

$$J_i(u) + \chi_i \left\|u - u^{i-1}\right\|^2 \to \min, \quad u \in K_i, \tag{6}$$

with $0 < \chi_i < \overline{\chi}$, u^{i-1} an approximate solution of (6) at the stage $i := i-1$ and $u^0 \in V$ an arbitrary starting point.

We refer to (5) and (6) as *Tikhonov's iterative regularization method* and *proximal-like method*, respectively.

Usually, dealing with a convex variational problem, the functions J_i are convex and the sets K_i are convex and closed. Therefore, the objective functions in the Problems (5) and (6) are strongly convex, and hence, these problems are uniquely solvable (if $K_i \neq \emptyset$). It should be emphasized that, inasmuch $\chi_i \to 0$ is not necessary for the convergence of the proximal-like methods (in particular, $\chi_i \equiv \chi > 0$ can be chosen), they possess a better stability and provide a better efficiency of fast convergent methods solving the regularized auxiliary problems.

Theoretical foundations for the construction and the convergence analysis of Tikhonov's iterative regularization methods have been developed in [32,40]. We refer to some methods coupling Tikhonov's regularization with gradient projection methods [8], Newton methods [7], augmented Lagrangian [2] and penalty methods [40]. In the latter paper the stability of regularized penalty methods for Problem (1), (2) with $Y = \mathbf{R}^n$ is proved without assuming convexity of J and B. For applications of Tikhonov's regularization in the framework of successive discretization of ill-posed variational problems see [28,40].

Proximal-like methods have been intensively developed during the last two decades. Starting with the papers [3] and [36], where the proximal method of multipliers has been investigated, regularized variants of different penalty methods (cf. [1,5,6,19]), steepest descent method [18], Newton methods [34,41] and quasi-Newton methods [10] have been suggested. In [21] proximal regularization is coupled with penalization and successive discretization for solving ill-posed convex semi-infinite problems, and in [22] a proximal method with successive discretization has been studied for solving elliptic variational inequalities. There is a couple of papers in which proximal regularization is used to obtain new decomposition (splitting) algorithms ([11,14,17]) and new bundle algorithms for nondifferentiable optimization problems ([4,10,24,25,31]). In some papers mentioned a nonquadratic proximal regularization is carried out by means of the Bregman function [13].

General schemes for the investigation of proximal-like methods have been developed in [20,26] and [37].

The scheme in [20] includes a generalization of (6), where the proximal iterations are repeated for fixed J_i, K_i until they providean'appropriate' decrease of the functional J_i.

See also

► Sensitivity and Stability in NLP

References

1. Alart P, Lemaire B (1991) Penalization in non-classicalconvex programming via variational convergence. Math Program 51:307–331
2. Antipin AS (1975) Regularization methods for convex programming problems. Ekonomika i Mat Metody 11:336–342 (in Russian).
3. Antipin AS (1976) On a method for convex programs using a symmetrical modification of the Lagrange function. Ekonomika i Mat Metody 12:1164–1173 (in Russian).
4. Auslender A (1987) Numerical methods for nondifferentiable convex optimization. Math Program Stud 30:102–126
5. Auslender A, Crouzeix JP, Fedit P (1987) Penalty proximal methods in convex programming. J Optim Th Appl 55:1–21
6. Auslender A, Haddou M (1995) An interior-proximal method for convex linearly constrained problems and its extension to variational inequalities. Math Program 71:77–100
7. Bakushinski AB, Goncharski AV (1989) Ill-posed problems – Numerical methods and applications. Moscow Univ. Press, Moscow
8. Bakushinski AB, Polyak BT (1974) About the solution of variational inequalities. Soviet Math Dokl 15:1705–1710
9. Beer G (1993) Topologies on closed and convex closed sets. Math and its Appl, vol 268. Kluwer, Dordrecht
10. Bonnans J, Gilbert JCh, Lemaréchal C, Sagastizabal C (1995) A family of variable metric proximal methods. Math Program 68:15–47
11. Chen G, Teboulle M (1992) A proximal-based decomposition method for convex minimization problems. Math Program 66:293–318
12. Dontchev AL, Zolezzi T (1993) Well-posed optimization problems. Lecture Notes Math, vol 1543. Springer, Berlin
13. Eckstein J (1993) Nonlinear proximal point algorithms using Bregman functions, with application to convex programming. Math Oper Res 18:202–226
14. Eckstein J, Bertsekas D (1992) On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math Program 55:293–318
15. Furi M, Vignoli A (1970) About well-posed minimization problems for functionals in metric spaces. J Optim Th Appl 5:225–290
16. Hettich R,Kortanek KO (1993) Semi-infinite programming: Theory, methods and applications. SIAM Rev 35:380–429
17. Ibaraki S, Fukushima M, Ibaraki T (1992) Primal-dual proximal point algorithm for linearly constrained convex programming problems. Comput Optim Appl 1:207–226
18. Iusem AN, Svaiter BF (1995) A proximal regularization of the steepest descent method. RAIRO Rech Opérat 29:123–130
19. Kaplan A (1982) Algorithm for convex programming using a smoothing for exact penalty functions. Sibirsk Mat Zh 23:53–64
20. Kaplan A, Tichatschke R (1994) Stable methods for ill-posed variational problems: Prox-regularization of elliptical variational inequalities and semi-infinite optimization problems. Akad. Verlag, Berlin
21. Kaplan A, Tichatschke R (1996) Path-following proximal approach for solving ill-posed convex semi-infinite programming problems. J Optim Th Appl 90:113–137
22. Kaplan A, Tichatschke R (1997) Prox-regularization and solution of ill-posed elliptic variational inequalities. Appl Math 42:111–145
23. Karmanov VG (1980) Mathematical programming. Fizmatgiz, Moscow
24. Kiwiel K (1995) Proximal level bundle methods for convex nondifferentiable optimization, saddle point problems and variational inequalities. Math Program 69B:89–109
25. Kiwiel K (1996) Restricted step and Levenberg–Marquardt techniques in proximal bundle methods for non-convex nondifferentiable optimization. SIAM J Optim 6:227–249
26. Lemaire B (1988) Coupling optimization methods and variational convergence. ISNM 84:163–179
27. Levitin ES, Polyak BT (1966) Minimization methods under constraints. J Vycisl Mat i Mat Fiz 6:787–823
28. Liskovets OA (1987) Regularization of problems with monotone operators when the spaces and operators are discretly approximated. USSR J Comput Math Math Phys 27:1–8
29. Louis AK (1989) Inverse and schlecht gestellte Probleme. Teubner, Leipzig Studienbücher Math
30. Martinet B (1970) Régularisation d'inéquations variationelles par approximations successives. RIRO 4:154–159
31. Mifflin R (1996) A quasi-second-order proximal bundle algorithm. Math Program 73:51–72
32. Mosco U (1969) Convergence of convex sets and of solutions of variational inequalities. Adv Math 3:510–585
33. Polyak BT (1987) Introduction to optimization. Optim. Software, New York
34. Qi L, Chen X (1997) A preconditioning proximal Newton method for non-differentiable convex optimization. Math Program 76B:411–429
35. Revalski JP (1995) Various aspects of well-posedness of optimization problems. In: Lucchetti R, Revalski J (eds) Recent Developments in Well-Posed Variational Problems. Kluwer, Dordrecht

36. Rockafellar RT (1976) Augmented Lagrange multiplier functions and applications of the proximal point algorithm in convex programming. Math Oper Res 1:97–116
37. Rockafellar RT (1976) Monotone operators and the proximal point algorithm. SIAM J Control Optim 14:877–898
38. Tikhonov AN (1966) On the stability of the functional optimization problems. USSR J Comput Math Math Phys 6:631–634
39. Tikhonov AN, Arsenin VJ (1977) Methods for solving ill-posed problems. Wiley, New York
40. Vasil'ev FP (1981) Methods for solving extremal problems. Nauka, Moscow
41. Wei Z, Qi L (1996) Convergence analysis of a proximal Newton method. Numer Funct Anal Optim 17:463–472

Image Space Approach to Optimization

Franco Giannessi
Department Math., Universita di Pisa, Pisa, Italy

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Image space; Separation; Separation functions; Necessary conditions; Sufficient condition; Lagrange multipliers; Lagrange function; Euler equation

The study of the properties of the image of a real-valued functionis an old one; recently, it has been extended to multifunctions and to vector-valued functions. However, in most cases the properties of the image have not been the purpose of study and their investigation has occurred as an auxiliary step toward other achievements (see, e. g., [4,16,17]).

Traces of the idea of studying the images of functions involved ina constrained extremum problem go back to the work of C. Carathéodory (in 1935, [3, Chap.5]). In the 1950s R. Bellman [1], with his celebrated maximum principle, proposed – for the first time in the field of optimization – to replace the given unknown by a new one which runs in the image; however, alsohere the image is not the main purpose. Only in the late 1960s and 1970ssome Authors, independently from each other, have brought explicitly such a study into the field of optimization [2,6,7,10,11].

The approach consists in introducing the space, call it image space (IS), where the images of the functions of the given optimization problem run. Then, a new problem is defined in the IS, which is equivalent to the given one. In a certain sense, such an approach has some analogieswith what happens in the measure theory when one goes from Mengoli–Cauchy–Riemann measure to the Lebesgue one.

The approach will now briefly be described. Assume we are given the integers m and p with $m \geq 0$ and $0 \leq p \leq m$, the subset X of a Hilbert space H whose scalar product is denoted by $\langle \cdot, \cdot \rangle$, and the functions $f{:}X \to \mathbf{R}$, $g_i{:}X \to \mathbf{R}$, $i = 1, \ldots, m$. Consider the minimization problem:

$$(P)\begin{cases} \min & f(x), \\ \text{s.t.} & g_i(x) = 0, \quad i \in I^0, \\ & g_i(x) \geq 0, \quad i \in I^+, \quad x \in X, \end{cases}$$

where $I^0 := \{1, \ldots, p\}$, $I^+ := \{p+1, \ldots, m\}$, and $p = 0 \Rightarrow I^0 = \emptyset$, $p = m \Rightarrow I^+ = \emptyset$; $m = 0 \Rightarrow I := I^0 \cup I^+ = \emptyset$. Here and in the sequel, all the considered extrema and integrals are supposed to exist; the discussion of their existence goes beyond the scope of this paper. Let us set $g(x) := (g_1(x), \ldots, g_m(x))$, $O_p := (0, \ldots, 0) \in \mathbf{R}^p$, $C := O_p \times \mathbf{R}_+^{m-p}$ and $R := \{x \in X{:}g(x) \in C\}$, with the stipulation that $C = \mathbf{R}_+^m$ when $p = 0$ and $C = O_m := (0, \ldots, 0) \in \mathbf{R}^m$ when $p = m$; $m = 0$ does not require to define C.

A particular case of (P), call it $(P)_{iso}$, is a classic isoperimetric type problem defined in the following way. Let $\mathcal{A}C_n(T)$ denote the class of absolutely continuous n-vector functions $x(t) := (x_1(t), \ldots, x_n(t))$ on $T := [a, b] \subset \mathbf{R}$ with square integrable derivatives. By suitably defining the scalar product – and, consequently, the norm – such a class is a Hilbert space; set $H = \mathcal{A}C_n(T)$ and

$$f(x) = \int_T \psi_0(t, x(t), \dot{x}(t))\, dt,$$

$$g_i(x) = \int_T \psi_i(t, x(t), \dot{x}(t))\, dt, \quad i \in I,$$

where $\psi_i \mathbf{R}^{1+2n} \to \mathbf{R}$, $i \in \{0\} \cup I$, are given integrands. Fixed endpoints conditions can be included in the definition of X. We point out that the problems which can be reduced to the format of (P) share the characteristic of having a finite-dimensional image. Hence, certain problems, for instance of geodesic type, are not covered by (P); the image analysis of them is outside the scope of the present writing.

The IS approach arises naturally in as much as an optimality condition for (P) is achieved through the impossibility of a system. More precisely, by paraphrasing the very definition of global minimum we can say that a feasible $\overline{x} \in X$ is a *global minimum point* for (P) if and only if the system (in the unknown x):

$$(S)\begin{cases}\varphi(\overline{x};x) := f(\overline{x}) - f(x) > 0,\\ g(x) \in C, \qquad\qquad x \in X,\end{cases}$$

is impossible, or

$$(S') \quad \mathcal{H} \cap \mathcal{K}(\overline{x}) = \emptyset,$$

where $\mathcal{H} := \{(u,v) \in \mathbf{R} \times \mathbf{R}^m : u > 0, v \in C\}$ and $\mathcal{K}(\overline{x}) := \{(u,v) \in \mathbf{R} \times \mathbf{R}^m : \ u = \varphi(\overline{x};x),\ v = g(x),\ x \in X\} \ = F(X)$, where $F := (\varphi, g)$. It is easy to see that (S′) holds if and only if

$$(S'') \quad \mathcal{H} \cap [\mathcal{K}(\overline{x}) - \operatorname{clos} \mathcal{H}] = \emptyset,$$

where the difference is in vector sense and clos denotes closure. $\mathcal{K}(\overline{x})$ is called the *image* of (P) and $\mathcal{K}(\overline{x}) - \operatorname{clos} \mathcal{H}$ its *conic extension*. The replacement of the image with its conic extension – which corresponds to modifying f and g – does not affect the optimality conditions and has several advantages. For instance, if f and $-g$ are convex (or, more generally, $(f, -g)$ is convex like [6,20]), then $\mathcal{K}(\overline{x}) - \operatorname{clos} \mathcal{H}$ is convex even if $\mathcal{K}(\overline{x})$ is not. Note that a change of $\overline{x}$ produces merely a translation of $\mathcal{K}(\overline{x})$ with respect to the u-axis. Hence, the properties of the image can be studied independently of the choice of $\overline{x}$.

The analysis in the IS must be viewed as a preliminary and auxiliary step – and not as a concurrent analysis – for studying (P). If this aspect is understood, then the IS analysis may be highly fruitful. In fact, in the IS we may have a sort of 'regularization': The conic extension of the image of (P) may be convex or continuous or smooth when (P) (and its image) do not enjoy the same property, so that convex or continuous or smooth analysis can be developed in the IS but not in H. (P) at $H = \mathbf{R}^n$ and $(\mathrm{P})_{\mathrm{iso}}$ have their unknowns in a finite and in an infinite-dimensional spaces, respectively; while the images of both problems run in a finite-dimensional space; hence, in the given spaces, namely $\mathbf{R}^n$ and $\mathcal{AC}_n(T)$, they require substantially different mathematical tools, while in the IS they can be treated in the same way.

It is easy to show that (P) is equivalent to the following *image problem*:

$$IP\begin{cases}\max & u\\ \text{s.t.} & (u,v) \in \mathcal{K}(\overline{x}), \quad v \in C.\end{cases}$$

A maximum point, say $(\widehat{u}, \widehat{v})$, of IP is the image – through the pair (φ, g)—of a minimum point, say $\widehat{x}$, of (P), and we have $f(\overline{x}) - \widehat{u} = f(\widehat{x})$, whatever $\overline{x} \in X$ may be. If IP is replaced by the other one:

$$IP(\xi)\begin{cases}\max & u\\ \text{s.t.} & (u,v) \in \mathcal{K}(\overline{x}),\\ & v_i = \xi_i, \quad i \in I^0,\\ & v_i \geq \xi_i, \quad i \in I^+,\end{cases}$$

then the maximum, which is now a function of $\xi := (\xi_1, \ldots, \xi_m)$, gives the so-called *perturbation function* (called also *optimal value function*) of (P), and, with obvious notation, we have $f(\overline{x}) - \widehat{u}(\xi) = f(\widehat{x}(\xi))$.

If in (P) minimization is replaced by maximization, then the entire image space analysis remains unchanged provided φ receives the new definition as $\varphi(\overline{x};x) := f(x) - f(\overline{x})$.

To prove directly whether or not (S′) (or (S″) and hence (S)) holds is generally impracticable. An indirect way of showing it consists in proving that $\mathcal{H}$ and $\mathcal{K}(\overline{x})$ lie in two disjoint level sets, respectively. This separation approach is exactly equivalent to finding a theorem of the alternative (cf. ► Theorems of the alternative and optimization and [6]) related to (S); only the mathematical languages are different. The separation scheme enjoys mainly a geometrical appeal, while in studying alternative the algebraic characters are dominant. In general terms, a separation scheme begins by introducing a family W of functionals such that the intersection of the positive level sets equals $\mathcal{H}$. If a $w \in W$ is found whose nonpositive level set contains $\mathcal{K}(\overline{x})$, then (S) holds and $\overline{x}$ is a solution of (P).

Let us consider the column vector $v = (v_1, \ldots, v_m)^\top$ and the row vectors $\lambda = (\lambda_1, \ldots, \lambda_m)$, $\mu = (\mu_1, \ldots, \mu_m)$. A particular, but wide, family of type W is offered by the class of *parabolic-exponential functions* $w: \mathbf{R}^{1+m} \to \mathbf{R}$, defined by

$$W_{\text{pe}}: \quad w = w(u, v; \lambda, \mu) := u + \gamma(v; \lambda, \mu),$$

where $\lambda \in C^* = \mathbf{R}^p \times \mathbf{R}_+^{m-p}$, $\mu \in \mathbf{R}_+^m$, and

$$\gamma(v; \lambda, \mu) := -\sum_{i \in I^0} (\mu_i v_i^2 - \lambda_i v_i) + \sum_{i \in I^+} \lambda_i v_i \exp(-\mu_i v_i),$$

where $*$ marks positive polar, and λ, μ are parameters which describe the family. At $\mu = 0$, this family collapses to that of linear functions:

$$W_\ell: \quad \ell = \ell(u, v; \lambda) := u + \langle \lambda, v \rangle, \quad \lambda \in C^*,$$

where $\langle \cdot, \cdot \rangle$ marks the usual scalar product. It is easy to check that the family W_{pe} (and hence W_ℓ) fulfills the above mentioned intersection property (with respect to λ, μ). Starting from such a separation scheme, it is possible to develop most of the theory of constrained extrema as will be briefly shown.

The intersection of the positive level sets of the elements of W_{pe} is not open, since $\mathcal{H}$ is not open. Therefore, an element of $\mathcal{H}$ may be a limit point of elements of $\mathcal{K}(\overline{x})$ or the Bouligand tangent cone to $\mathcal{K}(\overline{x})$ at 0_{1+m} may intersect $\mathcal{H}$ even when (S′) holds, so that no element of W_{pe} may exist which shows separation between $\mathcal{H}$ and $\mathcal{K}(\overline{x})$. This drawback can be overcome by enlarging the family W_{pe}. For instance, in the linear case, W_ℓ is replaced by the family $\ell(u, v; \theta, \lambda) := \theta\, u + \lambda\, v$, $\theta \geq 0$, $\lambda \in C^*$, $(\theta, \lambda) \neq 0$; at $p = 0$, this is the family considered in ► Theorems of the alternative and optimization. Of course, such a relaxation does not guarantee any longer (S′), even if $\mathcal{K}(\overline{x})$ is included in the nonpositive level set of ℓ, since at $\theta = 0$ the nonpositive level set of ℓ intersects $\mathcal{H}$, the intersection containing the positive u axis. As a consequence, the separation scheme would be useless. A way of remedying this consists in cutting off the set of problems whose images can be included in the nonpositive level set of ℓ at $\theta = 0$ only. Such an exclusion, which is extended to the (linear)approximations of (P), is done by imposing suitable conditions on f and g, which are called *constraint qualifications* if they implicate only g and are called *regularity conditions* if they implicate both f and g (cf. ► Theorems of the alternative and optimization; [12]).

Now, let us show some consequences of the separation scheme. A first result is a sufficient condition. From the very definition of w, we see that the existence of vectors $\lambda \in C^*$ and $\mu \in \mathbf{R}_+^m$, such that $\mathcal{K}(\overline{x}) \subseteq \text{lev}_{\leq 0}\, w(u, v; \lambda, \mu)$ (where $\text{lev}_{\leq 0}$ denotes nonpositive level set) is sufficient for $\overline{x}$ to be a solution of (P). Hence, to achieve a sufficient condition in terms of X, f and g it is enough to replace, in the above inclusion, u and v with their expressions:

Theorem 1 *Let $\overline{x} \in \mathbf{R}$. If there exist $\overline{\lambda} \in C^*$ and $\overline{\mu} \in \mathbf{R}_+^m$, such that*

$$f(\overline{x}) - f(x) + \gamma(g(x); \overline{\lambda}, \overline{\mu}) \leq 0, \quad \forall x \in X, \tag{1}$$

then $\overline{x}$ is a global minimum point of (P).

Let us introduce the function $\mathcal{L}(x; \lambda, \mu) := f(x) - \gamma(g(x); \lambda, \mu)$. It is a generalization of the classic *Lagrangian function*, which is found at $\mu = 0$. At $p = m$, $\mathcal{L}$ becomes the so-called *augmented Lagrangian function*. At $p = m$ and $\lambda = 0$, $\mathcal{L}$ becomes the classic *Courant penalty function*. (See [5, p.12], where such a function has been introduced under the name 'sensitized'; see also [15].) Obviously, (1) holds if and only if $f(\overline{x}) \leq \mathcal{L}(x; \overline{\lambda}, \overline{\mu})$, $\forall x \in X$. Under the equality

$$\gamma(g(\overline{x}); \overline{\lambda}, \overline{\mu}) = 0, \tag{2}$$

we have $f(\overline{x}) = \mathcal{L}(\overline{x}; \overline{\lambda}, \overline{\mu})$; then (1), which is the algebraic form of a separation condition, can be rewritten in a different (but equivalent) form, which corresponds to a saddle-point condition. At $\mu = 0$, (2) collapses to the *orthogonal condition*

$$\left\langle \overline{\lambda}, g(\overline{x}) \right\rangle = 0, \tag{3}$$

which is classically known as *complementarity condition* [6,12], due to the role it has played in some algorithms. The fundamental condition (2), which subtends the entire theory of constrained extrema, will be proved within the next theorem.

Theorem 2 *Let $\overline{x} \in X$. If there exist $\overline{\lambda} \in C^*$ and $\overline{\mu} \in \mathbf{R}_+^m$, such that*

$$\mathcal{L}(\overline{x}; \lambda, \mu) \leq \mathcal{L}(\bar{x}; \overline{\lambda}, \overline{\mu}) \leq \mathcal{L}(x; \overline{\lambda}, \overline{\mu}), \quad \forall x \in X, \quad \forall \lambda \in C^*, \quad \forall \mu \in \mathbf{R}_+^m, \tag{4}$$

then $\overline{x}$ is a global minimum point of (P).

Proof It is enough to prove that (1) $\Leftrightarrow$ (4).

($\Rightarrow$) $\overline{x} \in \mathbf{R}, \overline{\lambda} \in C^* \Rightarrow \gamma(g(\bar{x}); \overline{\lambda}, \overline{\mu}) \geq 0$; the reverse of this inequality is implied by (1) at $x = \overline{x}$. Thus (2) follows. Because of (2), (1) is equivalent to the second of inequalities (4). It is easy to show that

$$\begin{aligned} &g(\overline{x}) \in C \quad \Leftrightarrow \quad \gamma(g(\overline{x}); \lambda, \mu) \geq 0, \\ &\forall \lambda \in C^*, \quad \forall \mu \in \mathbf{R}^m_+. \end{aligned} \tag{5}$$

In fact, the implication $\Rightarrow$ is trivial; the reverse implication follows quickly by reasoning ab absurdo. (2) and (5) imply the inequality

$$\begin{aligned} &\gamma(g(\overline{x}); \overline{\lambda}, \overline{\mu}) \leq \gamma(g(\overline{x}); \lambda, \mu), \\ &\forall \lambda \in C^*, \quad \forall \mu \in \mathbf{R}^m_+, \end{aligned} \tag{6}$$

which is equivalent to the first of (4).

($\Leftarrow$) The first of (4) is equivalent to (6) which implies $\overline{x} \in \mathbf{R}$, so that $\gamma(g(\overline{x}); \overline{\lambda}, \overline{\mu}) \geq 0$. The strict inequality contradicts (6) since, $\forall i \in I^+$, $\lambda_i g_i(\overline{x}) \exp(-\mu_i g_i(\overline{x}))$ can be made arbitrarily small so that the same happens to $\gamma(g(\overline{x}); \lambda, \mu)$. Hence (2) holds here too. Taking into account (2), it is easy to see that the second of inequalities (4) implies (1).

At $\mu = 0$, (4) is the classic *saddle-point sufficient condition* and (2) becomes (3). Note that Theorem 2 does not contain any assumption on X, f and g.

Example 3 Let us set $H = \mathbf{R}$, $X =]-1, +\infty[$, $p = 0$, $m = 1$, $f(x) = \log(x + 1)$, $g(x) = x$. At $\mu = 0$, (4) is equivalent to the system of $(\lambda - \overline{\lambda})\overline{x} \geq 0$ and

$$\log\left(\frac{x+1}{\overline{x}+1}\right) \geq \overline{\lambda}(x - \overline{x}), \qquad \forall x \in]-1, +\infty[, \quad \forall \lambda \geq 0,$$

which is impossible. Hence, the classic saddle-point condition is not satisfied. At $\overline{\mu} > 0$, (4) can be satisfied. In fact, at $\overline{x} = 0$, (4) is equivalent to $\log(x + 1) \geq \overline{\lambda} x \exp(-\overline{\mu} x)$, $\forall x \in]-1, +\infty[$, which is true if $\overline{\lambda} = 1$ and $\overline{\mu}$ is large enough. Hence, Theorem 2 can now be applied to state that $\overline{x} = 0$ is a global minimum point of (P).

Example 4 Let us set $X \subset H = \mathcal{A}C_2(T)$, where $T := [0, \tau]$ is the domain of the elements $x = (x_1, x_2) \in H$; $x_1 = x_1(t)$ and $x_2 = x_2(t)$, $t \in T$, are the parametric equations of a curve γ in $\mathbf{R}^2$; given a positive real ℓ, τ must be such that the length of γ be ℓ. X is now the set of pairs $x = (x_1, x_2) \in H$, such that $x_1(0) = x_2(0) = 0$, $x_2(t) \geq 0$, $\forall t \in T$, each x_i is regular in the sense of Jordan and closed. Moreover, we set $p = m = 1$, $T = [0, 2\pi]$, and

$$f(x) = \int x_1 \, dx_2, \quad g(x) = \int \sqrt{dx_1^2 + dx_2^2} - \ell.$$

Consider the problem

$$P(\ell) \begin{cases} \max & f(x), \\ \text{s.t.} & g(x) = 0, \quad x \in X. \end{cases}$$

The solution of this classic isoperimetric problem is well known:

$$\begin{aligned} &\overline{x}_1(t; \ell) = \frac{\ell}{2\pi} \cos\left(t - \frac{\pi}{2}\right), \\ &\overline{x}_2(t; \ell) = \frac{\ell}{2\pi}\left[1 + \sin\left(t - \frac{\pi}{2}\right)\right], \\ &t \in T = [0, 2\pi], \end{aligned}$$

or, in nonparametric form, $x_1^2 + x_2^2 - x_2\, \ell/\pi = 0$, and the maximum is $\ell^2/4\pi$. If in P(l) we replace $g(x) = 0$ with $g(x) = \xi$, so that we consider P($\ell + \xi$), then $\forall \overline{x} \in X$ we have

$$\max_{\substack{(u,v) \in \mathcal{K}(\overline{x}) \\ v = \xi}} (u) = f(\overline{x}) - \frac{(\ell + \xi)^2}{4\pi}.$$

It follows that $\mathcal{K}(\overline{x})$ is included in a convex (with respect to u-axis) parabola; hence $\mathcal{H}$ and $\mathcal{K}(\overline{x})$ can be separated by a line, so that (1) and (4) can be verified at $\mu = 0$. Any $\overline{x} \in X$ (and not necessarily an optimal one) allows to carry on the analysis in the image space. Of course, in general, it is impossible to have an explicit form of $\mathcal{K}(\overline{x})$. In the present example, to show explicitly a part of $\mathcal{K}(\overline{x})$, namely the perturbation function, we have exploited the knowledge of the maximum point.

Let us stress that the sufficient condition (1), as well as (4), is an important result; however, in general, it is not necessary and it is difficult to be verified since the inequality must be fulfilled $\forall x \in X$. Therefore, it is useful to weaken the analysis, by replacing $\mathcal{K}(\overline{x})$ withan 'approximation' which be 'easier' to handle. A natural way of doing this consists in approximating $\mathcal{K}(\overline{x})$ at $F(\overline{x}) := (\varphi(\overline{x}; \overline{x}), g(\overline{x}))$ by means of its tangent cone (e. g., in the sense of Bouligand [4,6]); in general a cone is obviously easier than any set. For the sake of simplicity, now we will consider a particular case of (P)

and adopt a separation scheme less general than above, which however embeds the classic theory [12]; for more general results see [6,7,9,10,11,13,14,18,19,20].

Consider the particular case where $p = 0$ (so that $C = \mathbf{R}^m_+$; the presence of bilateral constraints makes the analysis extremely difficult, unless f and g are assumed to fulfill conditions which make applicable Dini or Lyusternik implicit function theorems) and X is open. Denote by C the set of *sublinear real-valued functions* (i. e., positively homogeneous of degree one and convex) defined on H; f is *superlinear* if and only if $-f$ is sublinear. f is said to be *C-differentiable* at $\overline{x}$ if and only if $\exists \mathcal{D}_C f\colon\ X \times X \to \mathbf{R}$, such that $\mathcal{D}_C(\overline{x};\cdot) \in C$, and

$$\lim_{z\to 0} \frac{1}{\|z\|}[\epsilon(\overline{x};z) := f(\overline{x}+z) - f(\overline{x}) - \mathcal{D}_C f(\overline{x};z)] = 0, \quad (7)$$

where $\|\cdot\|$ is the norm in H generated by the scalar product and z belongs to a neighborhood, say Z, of $\overline{x}$. $\mathcal{D}_C f$ is said to be the C-derivative of f at $\overline{x}$. If $\mathcal{D}_C f$ is linear (the linear functions are obviously elements of C), then f is *differentiable* [7]. It is easy to see that a C-differentiable function is *directionally differentiable* in any direction z (in the sense that there exists the limit of $[f(\overline{x}+\alpha z) - f(\overline{x})]/\alpha$ as $\alpha \downarrow 0$); the vice versa is not true, as shown by the following example(which generalizes the so-called *Peano function* showed by G. Peano to detect a famous mistake by Lagrange):

Example 5 $H = X = R = \mathbf{R}^2$, $x = (x_1, x_2)$, $\bar{x} = 0$, $\|z\| = \|z\|_2$, $f(x) = (x_1^2 + x_2^2)^{1/2}$ if $x \not> 0$ and $f(x) = \alpha(x_2/x_1^2)(x_2^2 + x_2^2)^{1/2}$ if $x > 0$, where α:$\mathbf{R}_+ \setminus \{0\} \to \mathbf{R}$ is defined by $\alpha(t) = 1$ if $0 < t \leq 1$ or $t \geq 3$, $\alpha(t) = 3 - 2t$ if $1 < t \leq 2$ and $\alpha(t) = 2t - 5$ if $2 < t < 3$. In this Example, at $x = \overline{x}$ the directional derivative exists and is $f'(\bar{x};z) = (z_1^2 + z_2^2)^{\frac{1}{2}}$, while it is not possible to verify (7). Note that f is continuous, but not locally Lipschitz, and $f'(0;z) > 0$, $\forall z \in \mathbf{R}^2$.

Example 6 $H = X = R = \mathbf{R}$, $f\colon \mathbf{R} \to \mathbf{R}_+$ with $f(x) = |x| + x^2$ if $x \in \mathbf{Q}$ and $f(x) = |x| + 2x^2$ if $x \notin \mathbf{Q}$, $\mathbf{Q}$ being the set of rational numbers. f is C-differentiable at $\overline{x} = 0$ with C-derivative $\mathcal{D}_C f(0;z) = |z|$. Note that f is continuous at $\overline{x} = 0$ only.

The *C-subdifferential* of a C-differentiable function f at $\overline{x} \in X$ is defined by

$$\partial_C f(\overline{x}) := \left\{ z^* \in H'\colon \begin{array}{c} \mathcal{D}_C f(\overline{x};z) \geq \langle z^*, z\rangle, \\ \forall z \in Z \end{array} \right\},$$

where H' is the continuous dual of H; z^* is called the *C-subgradient* of f at $\overline{x}$. When f is convex, then $\partial_C f(\overline{x})$ collapses to the classic subdifferential which is denoted simply by $\partial f(\overline{x})$; hence, $\partial_C f(\overline{x})$ is nothing more than the subdifferential of $\mathcal{D}_C f(\overline{x};z)$ or $\partial_C f(\overline{x};z) = \partial \mathcal{D}_C f(\overline{x};z)$. When $\mathcal{D}_C f(\overline{x};z)$ is linear, then $\partial_C f(\overline{x})$ is a singleton and collapses to the classic differential. In the latest example $\partial_C f(0) = [-1, 1]$. Consider the further example: $H = X = R = \mathbf{R}$, $f(x) = x^2 \sin 1/x$ if $x \neq 0$ and $f(0) = 0$. We find $\mathcal{D}_C f(0;z) = 0$, $\forall z \in \mathbf{Z}$ (indeed f is differentiable), so that $\partial_C f(0) = \{0\}$, while the Clarke subdifferential [4] is $[-1, 1]$. Now consider the following regularity condition:

$$(RC) \quad T(\mathcal{K}(\overline{x})) \cap \{(u,v) \in \mathcal{H}\colon\ v = 0\} = \emptyset,$$

where $T(\mathcal{K}(\overline{x}))$ is the Bouligand tangent cone of $\mathcal{K}(\overline{x})$ at $\overline{x}$. Several conditions on f and g are well known (mainly when $H = \mathbf{R}^n$) which guarantee (RC). Consider for instance the case where $H = \mathbf{R}^n$ and f, g are derivable. (RC) holds if the gradients $\nabla g_i(\overline{x})$, $i \in \{i \in I\colon\ g_i(\overline{x}) = 0\}$ are linearly independent. (RC) holds if g is affine. (RC) holds if g is concave and $\exists \widehat{x} \in X$ such that $g(\widehat{x}) > 0$. For additional conditions see ► Theorems of the alternative and optimization; [6,12].

The approximation of (P), we want to discuss in the present particular case $p = 0$, consists in replacing f and $-g_i$, $i \in I$, with their C-derivatives. More precisely, instead of the map $F = (\varphi, g)$, we consider now the superlinear map

$$F_C(\overline{x};z) := (-\mathcal{D}_C f(\overline{x};z), g_i(\overline{x}) + \mathcal{D}_C g_i(\overline{x};z) \quad i \in I),$$

which is the first order expansion of the $(-C)$-differentiable map F. $\mathcal{K}(\overline{x})$ is now replaced by the cone $\mathcal{K}_C(\overline{x}) := F_C(\overline{x}; X - \overline{x})$. (S′) is now replaced by

$$(S''') \quad \mathcal{H} \cap \mathcal{K}_C(\overline{x}) = \emptyset,$$

which holds if $\overline{x}$ is a minimum point of (P) (but not necessarily vice versa due to the above approximation of $\mathcal{K}(\overline{x})$; hence, from the necessity and sufficiency of

(S′) we jump to the sole necessity of (S‴), since $\mathcal{H}$ and $\mathcal{K}_C(\overline{x})$ are linearly separable; hence (S‴) can be proved by means of the subclass of W_{pe} at $\mu = 0$. As a consequence the Lagrangian function $\mathcal{L}$ (which, as we have seen above, is, up to a formal transformation, the separation function w) will be used at $\mu = 0$; thus we set $L(x;\lambda) := \mathcal{L}(x;\lambda, 0)$. This leads to the following necessary condition, whose proof can be found in [7].

Theorem 7 *Let the functions f, $-g_i$, $i \in I$, be C-differentiable, and assume that (RC) be fulfilled. If $\overline{x}$ is a minimum point of (P), then $\exists\overline{\lambda} \in \mathbf{R}^m$, such that*

$$\inf_{z\in B} \mathcal{D}_C L(\overline{x}; z; \overline{\lambda}) \geq 0, \tag{8}$$

$$g(\overline{x}) \geq 0, \quad \overline{\lambda} \geq 0, \tag{9}$$

$$\left\langle \overline{\lambda}, g(\overline{x}) \right\rangle = 0, \tag{10}$$

where $\mathcal{D}_C L(\overline{x}; z; \overline{\lambda})$ is the C-derivative of L at $x = \overline{x}, \lambda = \overline{\lambda}$, and $B := \{z: \| z \| = 1\}$.

(8) is equivalent to

$$0 \in \partial_C f(\overline{x}) + \sum_{i\in I} \overline{\lambda}_i \partial_C(-g_i(\overline{x})), \tag{11}$$

which becomes

$$0 \in \partial f(\overline{x}) + \sum_{i\in I} \overline{\lambda}_i \partial(-g_i(\overline{x})), \tag{12}$$

if, in particular, X, f and $-g$ are convex. When f and g are differentiable on X, then (8) collapses to $\mathcal{V}_L = 0$ along $x = \overline{x}$, where $\mathcal{V}_L$ is the first variation of L, and in case $(P)_{iso}$ becomes

$$\Psi'_x(t, \overline{x}, \overline{x}'; \overline{\lambda}) - \frac{d}{dt}\Psi'_{x'}(t, \overline{x}, \overline{x}'; \overline{\lambda}) = 0, \tag{13}$$

where $\Psi := \psi_0 - \sum_{i\in I} \lambda_i \psi_i$ is the integrand of L. If $X = H = \mathbf{R}^n$, then (8) collapses to

$$L'_x(\overline{x}; \overline{\lambda}) = 0, \tag{14}$$

where L'_x is the gradient of L with respect to x.

Note that (13) is the classic *Euler equation* and (14) is the classic *Lagrange equation*; $\overline{\lambda}$ is the vector of *Lagrange multipliers* which turns out to be the gradient of the hyperplane ($w = 0$ at $\mu = 0$) which separates the two sets of (S″).

Now, let us go back to the separation scheme which led to the sufficient condition (1). The choice of proving (S′) indirectly through separation has a lot of interesting consequences which go beyond the initial purpose. One of them is the introduction of a (nonlinear) dual space: that of functionals w. When we restrict ourselves to W_{pe}, then the dual space is isomorphic to $\mathbf{R}^{2m}$ (to $\mathbf{R}^m$ at $\mu = 0$; this is the classic duality scheme in finite-dimensional optimization). Such an isomorphism is the characteristic of constrained extremum problems having finite-dimensional image (independently of the dimension of the space where the unknown runs).

Having recognized that we have introduced a dual space, to define a dual problem is immediate. Indeed, looking at (1), since the inequality must be fulfilled $\forall x \in X$, it is straightforward, for each λ and μ, to search for $\max_{x\in X} w(\varphi(\overline{x}; x), g(x); \lambda, \mu)$ and then to find λ, μ which make such a maximum as small as possible and, hopefully, not greater than zero. Hence, we are led to study the problem:

$$(P^*) \qquad \max_{\lambda\in C^*, \mu\in\mathbf{R}^m_+} \min_{x\in X} \mathcal{L}(x; \lambda, \mu),$$

which we call *generalized dual problem* of (P); any pair (λ^*, μ^*) which solves (P*) is a *dual variable* [6,19]. At $\mu = 0$, (P*) is the classic *dual problem* of (P) [12]; indeed, the classic duality theory starts by defining (P*) as a dual problem, independently of the separation scheme and hence of the other theories like the saddle-point one. It is easy to show that the maximum in (P*) is $\leq$ of the minimum in (P); the difference between the latter and the former is called *duality gap*; it is now clear that a positive duality gap corresponds to a lack of separation between $\mathcal{H}$ and $\mathcal{K}(\overline{x})$ at the minimum point $\overline{x}$.

Another important topic which can be derived by the separation scheme is the penalization theory. Seemingly independent of the other topics of Optimization, it is indeed strictly related to them, since it can be drawn from the separation scheme, as will be now briefly outlined (recall the remark after Theorem 1). Consider again the family W_{pe} within which select a sequence, say $\{w_r := w(u, v; \lambda^r, \mu^r)\}_{r=1}^{\infty}$, of separation functions, such that the positive level set (with respect to (u, v)) of w_{r+1} be strictly included in that of w_r. Then, we can try to 'fulfill (1) asymptotically' or to set up the sequence of problems:

$$(P_r) \qquad \min_{x\in X} \mathcal{L}(x; \lambda^r, \mu^r); \quad r = 1, 2, \ldots.$$

Under suitable conditions, a limit point of $\{x^r\}_1^\infty$ (x^r being a solution of (Pr) is a solution of (P). See [6,15] for details.

Let us stress the fact that the separation scheme and its consequences come down from (S), and do not 'see' (P); they are unacquainted with the fact that the impossibility of (S) expresses optimality for (P). Therefore, it is obvious that the separation approach can be applied to every kind of problem which leads to the impossibility of a system like (S). In fact, such an approach can be applied to vector optimization and to variational inequalities [8], and to generalized systems [14,19].

See also

► Vector Optimization

References

1. Bellmann R (1957) Dynamic programming. Princeton Univ. Press, Princeton
2. Camerini PM, Galbiati G, Maffioli F (1991) The image of weighted combinatorial problems. Ann Oper Res 33:181–197
3. Carathéodory C (1982) Calculus of variations and partial differential equations of the first order. Chelsea, New York
4. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
5. Courant R (1943) Variational methods for the solution of problems of equilibrium and vibrations. Bull Amer Math Soc 49:1–23
6. Giannessi F (1984) Theorems of the alternative and optimality conditions. J Optim Th Appl 42(3):331–365, Preliminary version published in: Proc. Math. Program. Symp., Budapest, 1976
7. Giannessi F (1994) General optimality conditions via a separation scheme. In: Spedicato E (ed) Algorithms for Continuous Optimization. Kluwer, Dordrecht, 1–23
8. Giannessi F, Mastroeni G, Pellegrini L (1999) On the theory of vector optimization and variational inequalities. Image space analysis and separation. Vector Variational Inequalities and Vector Equilibria. Mathematical Theories. Kluwer, Dordrecht, 153–215
9. Giannessi F, Rapcsák T (1995) Images, separation of sets and extremum problems. In: Agarwal RP (ed) Recent Trends in Optimization Theory and Applications. Ser Appl Anal. World Sci., Singapore, 79–106
10. Hestenes MR (1966) Calculus of variations and optimal control theory. Wiley, New York
11. Hestenes MR (1975) Optimization theory: The finite dimensional case. Wiley, New York
12. Mangasarian OL (1994) Nonlinear programming. SIAM Ser, vol 10. SIAM, Philadelphia
13. Mastroeni G, Pappalardo M, Yen ND (1994) Image of a parametric optimization problem and continuity of the perturbation function. J Optim Th Appl 81(1):193–202
14. Mastroeni G, Rapcsák T (1999) On convex generalized systems. J Optim Th Appl
15. Pappalardo M (1994) Image space approach to penalty methods. J Optim Th Appl 64(1):141–152
16. Porciau BH (1980) Multipliers rules. Amer Math Monthly 87:443–452
17. Porciau BH (1983) Multipliers rules and separation of convex sets. J Optim Th Appl 40:321–331
18. Quang PH (1992) Lagrangian multiplier rules via image space analysis. Nonsmooth Optimization: Methods and Applications. Gordon and Breach, New York, 354–365
19. Rapcsák T (1997) Smooth nonlinear optimization in $\mathbf{R}^n$. Nonconvex, no. 19. Optim Appl. Kluwer, Dordrecht
20. Tardella F (1989) On the image of a constrained extremum problem and some applications to the existence of a minimum. J Optim Th Appl 60(1):93–104

Implicit Lagrangian

IL

MICHAEL V. SOLODOV
Institute Mat. Pura e Apl., Rio de Janeiro, Brazil

MSC2000: 90C33, 90C30

Article Outline

Keywords
Unconstrained Implicit Lagrangian
Restricted Implicit Lagrangian
Regularity Conditions
Derivative-Free Descent Methods
Error Bounds
Extensions
See also
References

Keywords

Complementarity problem; Merit function; Optimization

The *nonlinear complementarity problem* (see [3,15]) is to find a point $x \in \mathbf{R}^n$ such that

$$x \geq 0, \qquad F(x) \geq 0, \qquad \langle x, F(x)\rangle = 0, \qquad (1)$$

where $F : \mathbf{R}^n \to \mathbf{R}^n$ and $\langle \cdot, \cdot \rangle$ denotes the usual inner product in $\mathbf{R}^n$. A popular approach for solving the nonlinear complementarity problem (NCP) is to construct a *merit function* f such that solutions of NCP are related in a certain way to the optimal set of the problem

$$\begin{cases} \min & f(x) \\ \text{s.t.} & x \in C. \end{cases}$$

Of practical interest is the case when the set C has simple structure and smoothness properties of F and dimensionality n of the variables space are preserved. There is a number of ways to reformulate the NCP as an equivalent optimization problem (for a survey, see [7]).

Unconstrained Implicit Lagrangian

The first smooth unconstrained merit function was proposed by O.L. Mangasarian and M.V. Solodov [12]. This function is commonly referred to as the implicit Lagrangian; it has the following form:

$$\begin{aligned} M_\alpha(x) = \langle x, F(x)\rangle & \\ + \frac{1}{2\alpha}\left(\|(x - \alpha F(x))_+\|^2 - \|x\|^2\right) & \\ + \frac{1}{2\alpha}\left(\|(F(x) - \alpha x)_+\|^2 - \|F(x)\|^2\right) & \end{aligned}$$

where $\alpha > 1$ is a parameter and $(\cdot)_+$ denotes the orthogonal projection map onto the nonnegative orthant $\mathbf{R}^n_+$, i. e. the ith component of the vector $(z)_+$ is $\max\{0, z_i\}$. It turns out that $M_\alpha(x)$ is nonnegative on $\mathbf{R}^n$ provided $\alpha > 1$, and is zero if and only if x is a solution of the NCP. If F is differentiable on $\mathbf{R}^n$, then so is $M_\alpha(\cdot)$ and its gradient vanishes at all solutions of NCP for $\alpha > 1$. Hence, one can attempt to solve the NCP by means of solving the smooth unconstrained optimization problem

$$\begin{cases} \min & M_\alpha(x) \\ \text{s.t.} & x \in \mathbb{R}^n. \end{cases} \tag{2}$$

The implicit Lagrangian owes its name to the way the function was first derived in [12]. Consider the constrained minimization problem (MP)

$$\begin{cases} \min & \langle x, F(x)\rangle \\ \text{s.t.} & x \geq 0,\ F(x) \geq 0 \end{cases}$$

which is related to the NCP (1) in the sense that its global minima of zero coincide with the solutions of NCP. Because of the special structure of the MP (the objective function is the inner product of the functions defining constraints), for every feasible $\overline{x}$ such that $\langle \overline{x}, F(\overline{x})\rangle = 0$ it can be observed that $\overline{x}$ plays the role of the *Lagrange multiplier* [2] for the constraint $F(x) \geq 0$, while $F(\overline{x})$ plays a similar role for the constraint $x \geq 0$. Keeping in mind this observation, consider the *augmented Lagrangian* [1] for the above MP:

$$\begin{aligned} L_\alpha(x, u, v) = \langle x, F(x)\rangle & \\ + \frac{1}{2\alpha}\left(\|(-\alpha F(x) + u)_+\|^2 - \|u\|^2\right) & \\ + \frac{1}{2\alpha}\left(\|(-\alpha x + v)_+\|^2 - \|v\|^2\right), & \end{aligned}$$

where $u \in \mathbf{R}^n$ and $v \in \mathbf{R}^n$ are Lagrange multipliers corresponding to the constraints $F(x) \geq 0$ and $x \geq 0$ respectively. Since it is known a priori that at any solution $\overline{x}$ of MP (and NCP) one could take $\overline{u} = \overline{x}$ and $\overline{v} = F(\overline{x})$, it is intuitively reasonable to 'solve' for multipliers u and v in terms of the original variables. Replacing u by x and v by $F(x)$ in the augmented Lagrangian, one obtains the implicit Lagrangian function $M_\alpha(x)$.

The parameter α must be strictly bigger than one, because it can be checked that $M_1(x) = 0$ for all $x \in \mathbf{R}^n$. Another interesting property is that the partial derivative of the implicit Lagrangian with respect to α is also nonnegative for all x, and is zero if and only if x is a solution of the NCP [11]. However, a merit function based on this derivative is nonsmooth.

Restricted Implicit Lagrangian

When the implicit Lagrangian is restricted to the nonnegative orthant $\mathbf{R}^n_+$, where nonnegativity of x is explicitly enforced, the last two terms in the expression for $M_\alpha(x)$ can be dropped. Thus the restricted implicit Lagrangian is obtained:

$$N_\alpha(x)6 = \langle x, F(x)\rangle + \frac{1}{2\alpha}\left(\|(x - \alpha F(x))_+\|^2 - \|x\|^2\right),$$

where $\alpha > 0$. In this form, the function was introduced in [12]. It is also equivalent to the *regularized gap function* proposed by M. Fukushima [6] in the more general context of variational inequality problems (cf. also ► Variational inequalities).

The restricted implicit Lagrangian is nonnegative for all $x \in \mathbf{R}^n_+$ provided the parameter α is positive, and its zeroes coincide with solutions of the NCP. It also

inherits the differentiability of F. Thus the NCP can be solved via the bound constrained optimization problem

$$\begin{cases} \min & N_\alpha(x) \\ \text{s.t.} & x \geq 0. \end{cases} \tag{3}$$

Note that since it is known a priori that every solution of the NCP is nonnegative, one may also consider a bound constrained problem with the function $M_\alpha(x)$. However, for the constrained reformulations the function $N_\alpha(x)$ is probably preferable because it is somewhat simpler.

Regularity Conditions

It should be emphasized that only global solutions of optimization problems (2) and (3) are solutions of the underlying NCP (1). On the other hand, standard iterative methods are guaranteed to find stationary points rather than global minima of optimization problems. It is therefore important to derive conditions which ensure that these stationary points are also solutions of NCP. One such condition is *convexity*. However, the implicit Lagrangian is known to be convex only in the case of strongly monotone affine F, and provided parameter α is large enough [17]. Clearly, this is very restrictive. Thus other regularity conditions were investigated.

For the unconstrained problem (2), the first sufficient condition was given by N. Yamashita and Fukushima [24]. They established that if the Jacobian $\nabla F(\overline{x})$ is positive definite at a stationary point $\overline{x}$ of (2), then $\overline{x}$ solves the NCP. This result was later extended in [8] to the case when $\nabla F(\overline{x})$ is a *P-matrix*. Finally, F. Facchinei and C. Kanzow [5] obtained a certain regularity condition which is both necessary and sufficient for a stationary point point of the unconstrained implicit Lagrangian to be a solution of NCP (it is similar to the condition stated below for the restricted case).

For constrained problem (3), Fukushima [6] first showed the equivalence of stationary points to NCP solutions under the positive definiteness assumption on the Jacobian of F. A regularity condition which is both necessary and sufficient, was given by Solodov [20]: a point $x \in \mathbf{R}^n_+$ is said to be regular if $\nabla F(x)^\top$ reverses the sign of no nonzero vector $z \in \mathbf{R}^n$ satisfying

$$z_P > 0, \qquad z_C = 0, \qquad z_N < 0, \tag{4}$$

where

$$\begin{aligned} C &:= \{i\colon\ x_i \geq 0,\ F_i(x) \geq 0,\ x_i F_i(x) = 0\}, \\ P &:= \{i\colon\ x_i > 0,\ F_i(x) > 0\}, \\ N &:= \{i\colon\ x_i \geq 0,\ F_i(x) < 0\}. \end{aligned}$$

Recall [4] that the matrix $\nabla F(x)^\top$ is said to reverse the sign of a vector $z \in \mathbf{R}^n$ if

$$z_i [\nabla F(x)^\top z]_i \leq 0, \qquad \forall i \in \{1, \ldots, n\}. \tag{5}$$

Therefore a point $x \in \mathbf{R}^n_+$ is regular if the only vector $z \in \mathbf{R}^n$ satisfying both (4) and (5) is the zero vector.

A stationary point of (3) solves the NCP if and only if it is regular in the sense of the given definition.

Derivative-Free Descent Methods

When F is differentiable, so are the functions $M_\alpha(x)$ and $N_\alpha(x)$. Therefore, any standard optimization algorithm which makes use of first order derivatives can be applied to problems (2) and (3). However, taking advantage of the underlying structure one can also devise special descent algorithms which do not use derivatives of F. This can be especially useful in cases when derivatives are not readily available or are expensive to compute.

In [24], it was shown that when F is *strongly monotone* and continuously differentiable, then the direction

$$\begin{aligned} d(x) = (\beta - \alpha)(x - (x - \alpha F(x))_+) \\ + (1 - \alpha\beta)(F(x) - (F(x) - \alpha x)_+) \end{aligned}$$

is a descent direction for $M_\alpha(\cdot)$ at $x \in \mathbf{R}^n$, provided $\beta > 0$ is chosen appropriately. A descent method based on this direction with appropriate *line search*, converges globally to the unique solution of the NCP [24]. In [13], it was established that the *rate of convergence* is actually at least linear.

For the restricted implicit Lagrangian, a descent method with the direction

$$d(x) = (x - \alpha F(x))_+ - x$$

was proposed in [6]. The algorithm was proven to be convergent for the strongly monotone NCP (no rate of convergence has been established however). In [26], by using adaptive parameter α, this method was further extended to monotone (not necessarily strongly monotone) and Lipschitz continuous (not necessarily differentiable) functions.

Error Bounds

The implicit Lagrangian also appears useful for providing bounds on the distance from a given point to the solution set of the NCP. In particular, if F is affine then there exists a constant $c > 0$ such that (see [11])

$$\operatorname{dist}(x, S) \leq c M_\alpha(x)^{1/2}$$

for all x close to S, where S denotes the solution set. This inequality is called a local *error bound*. X.-D. Luo and P. Tseng [10] proved that, in the affine case, this bound is global (i. e. it holds for all $x \in \mathbf{R}^n$) if and only if the associated matrix is of the class R_0.

For the nonlinear case, Kanzow and Fukushima [9] showed that $M_\alpha(x)^{1/2}$ provides a global error bound if F is a *uniform P-function* which is Lipschitz continuous.

In the context of error bounds, the following relation established in [11] is useful: for all $x \in \mathbf{R}^n$,

$$\alpha^{-1}(\alpha - 1)\,\|r(x)\|^2 \leq M_\alpha(x) \leq (\alpha - 1)\,\|r(x)\|^2$$

where

$$r(x) = x - (x - F(x))_+$$

is the *natural residual* [14]. Therefore the implicit Lagrangian $M_\alpha(x)$ provides a local/global error bound if and only if so does the natural residual $r(x)$.

For the restricted implicit Lagrangian, one only has the following relation:

$$2\alpha N_\alpha(x) \geq \|r(x)\|^2 .$$

Thus, in principle, $N_\alpha(x)$ may provide a bound in cases when the natural residual does not.

For a general discussion of error bounds see [16].

Extensions

The implicit Lagrangian can be extended to the context of generalized complementarity problems and variational inequality problems via its relation with the regularized gap function. As observed by J.-M. Peng and Y.X. Yuan [19], the function $M_\alpha(x)$ can be represented as a difference of two regularized gap functions with parameters $1/\alpha$ and α. Since the regularized gap function can also be defined for variational inequalities, one might consider a similar expression in this more general context. Peng [18] established the equivalence of the variational inequality problem to unconstrained minimization of a difference of regularized gap functions. This result was further extended by Yamashita, K. Taji and Fukushima [25] who obtained similar results for differences of regularized gap functions whose parameters are not necessarily the inverse of each other. For algorithms based on this approach, see [22].

For a unified treatment of extensions of the implicit Lagrangian and the regularized gap function for the generalized complementarity problems see [23].

Yet another context where the implicit Lagrangian can be used is optimization reformulation of the extended linear complementarity problem[21].

See also

- ► Kuhn–Tucker Optimality Conditions
- ► Lagrangian Duality: Basics

References

1. Bertsekas DP (1982) Constrained optimization and Lagrange multiplier methods. Acad. Press, New York
2. Bertsekas DP (1995) Nonlinear programming. Athena Sci., Belmont, MA
3. Cottle RW, Giannessi F, Lions J-L (1980) Variational inequalities and complementarity problems: Theory and applications. Wiley, New York
4. Cottle RW, Pang J-S, Stone RE (1992) The linear complementarity problem. Acad. Press, New York
5. Facchinei F, Kanzow C (1997) On unconstrained and constrained stationary points of the implicit Lagrangian. J Optim Th Appl 92:99–115
6. Fukushima M (1992) Equivalent differentiable optimization problems and descent methods for asymmetric variational inequality problems. Math Program 53:99–110
7. Fukushima M (1996) Merit functions for variational inequality and complementarity problems. In: Di Pillo G, Giannessi F (eds) Nonlinear Optimization and Applications. Plenum, New York pp 155–170
8. Jiang H (1996) Unconstrained minimization approaches to nonlinear complementarity problems. J Global Optim 9:169–181
9. Kanzow C, Fukushima M (1996) Equivalence of the generalized complementarity problem to differentiable unconstrained optimization. J Optim Th Appl 90:581–603
10. Luo X-D, Tseng P (1997) On a global projection-type error bound for the linear complementarity problem. Linear Alg Appl 253:251–278
11. Luo Z-Q, Mangasarian OL, Ren J, Solodov MV (1994) New error bounds for the linear complementarity problem. Math Oper Res 19:880–892

12. Mangasarian OL, Solodov MV (1993) Nonlinear complementarity as unconstrained and constrained minimization. Math Program 62:277–297
13. Mangasarian OL, Solodov MV (1999) A linearly convergent descent method for strongly monotone complementarity problems. Comput Optim Appl 14:5–16
14. Pang J-S (1986) Inexact Newton methods for the nonlinear complementarity problem. Math Program 36(1):54–71
15. Pang J-S (1995) Complementarity problems. In: Horst R, Pardalos PM (eds) Handbook Global Optimization. Kluwer, Dordrecht, pp 271–338
16. Pang J-S (1997) Error bounds in mathematical programming. Math Program 79:299–332
17. Peng J-M (1996) Convexity of the implicit Lagrangian. J Optim Th Appl 92:331–341
18. Peng J-M (1997) Equivalence of variational inequality problems to unconstrained optimization. Math Program 78:347–356
19. Peng J-M, Yuan YX Unconstrained methods for generalized complementarity problems. J Comput Math (to appear)
20. Solodov MV (1997) Stationary points of bound constrained reformulations of complementarity problems. J Optim Th Appl 94:449–467
21. Solodov MV (1999) Some optimization reformulations for the extended linear complementarity problem. Comput Optim Appl 13:187–200
22. Solodov MV, Tseng P Some methods based on D-gap function for solving monotone variational inequalities. Comput Optim Appl (to appear)
23. Tseng P, Yamashita N, Fukushima M (1996) Equivalence of complementarity problems to differentiable minimization: A unified approach. SIAM J Optim 6:446–460
24. Yamashita N, Fukushima M (1995) On stationary points of the implicit Lagrangian for nonlinear complementarity problems. J Optim Th Appl 84:653–663
25. Yamashita N, Taji K, Fukushima M (1997) Unconstrained optimization reformulations of variational inequality problems. J Optim Th Appl 92:439–456
26. Zhu DL, Marcotte P (1993) Modified descent methods for solving the monotone variational inequality problem. Oper Res Lett 14:111–120

Increasing and Convex-Along-Rays Functions on Topological Vector Spaces

HOSSEIN MOHEBI
Mahani Mathematical Research Center
and Department of Mathematics,
University of Kerman, Kerman, Iran

MSC2000: 26A48, 52A07, 26A51

Article Outline

Keywords and Phrases
Introduction
ICAR Functions
Subdifferentiability of ICAR Functions
DCAR Functions
References

Keywords and Phrases

Monotonic analysis; ICAR functions; Abstract convexity

Introduction

The role of convex analysis in optimization is well known. One of the major properties of a convex function is its representation as the upper envelope of a family of affine functions. More specifically, every lower semicontinuous proper convex function can be expressed as the supremum of the family of affine functions, majorized by it [4]. The subject of abstract convexity arose precisely by generalizing this idea (see [5,6]). A function is said to be abstract convex if and only if it can be represented as the upper envelope of a class of functions, usually called *elementary functions*. One of the first studies in abstract convexity concerned the analysis of *increasing and positively homogeneous* (IPH) functions. It was initially carried out for functions defined over $\mathbb{R}^n_{++}$ and $\mathbb{R}^n_{+}$, where $\mathbb{R}^n_{++} := \operatorname{int} \mathbb{R}^n_{+}$, and later on extended to an arbitrary closed convex cone in [1] and an arbitrary topological vector space in [3]. This study was further extended to include increasing and convex-along-rays (ICAR) functions over $\mathbb{R}^n_{+}$. The study of IPH and ICAR functions has given rise to the subject of *monotonic analysis*, the study of increasing functions enjoying some additional properties, which has important applications in global optimization (see [5] for more details). The systematic study of this subject was started in [1] and [2] by J. Dutta, J. E. Martinez-Legaz, and A. M. Rubinov, where they analyzed IPH and ICAR functions defined on a cone. In the present article, we extend this analysis to the study of ICAR functions defined over an arbitrary topological vector space. We want to emphasize that the role of IPH functions in monotonic analysis is the same as the role of sublinear functions in

convex analysis, whereas ICAR functions play the role of convex functions. We define elementary functions, which can be considered as generalizations of min-type functions, and demonstrate that ICAR functions are abstract convex with respect to the class of such elementary functions. This leads us to develop suitable notions of subdifferential for ICAR functions. Finally, we study the class of decreasing and convex-along-rays (DCAR) functions.

Let H be a set of functions $h\colon X \longrightarrow \mathbb{R}_{+\infty}$ defined on a set X. Recall (see [6]) that a function $f\colon X \longrightarrow \mathbb{R}_{+\infty}$ is called abstract convex with respect to H (H-convex) if there exists a set $U \subseteq H$ such that $f(x) = \sup\{h(x)\colon h \in U\}$. Let $supp(f, H) := \{h \in H\colon h \leq f\}$ be the support set of a function $f\colon X \longrightarrow \mathbb{R}_{+\infty}$ with respect to H. The function $co_H f\colon X \longrightarrow \mathbb{R}_{+\infty}$ defined by $co_H f(x) := \sup\{h(x)\colon h \in supp(f, H)\}$ is called the H-convex hull of f. Clearly, f is H-convex if and only if $f = co_H f$. Let $f\colon X \longrightarrow \mathbb{R}_{+\infty}$ be a proper function and $x_0 \in domf = \{x \in X\colon f(x) < +\infty\}$. The set

$$\partial_H f(x_0) := \{h \in H\colon f(x) \geq f(x_0) + h(x) - h(x_0) \quad \text{for all } x \in X\}$$

is called the H-subdifferential of f at the point x_0. Obviously, $\partial_H f(x_0)$ is nonempty if $f(x_0) = \max\{h(x_0)\colon h \in supp(f, H)\}$.

Let (X,Y) be a pair of sets with a coupling function $\varphi\colon X \times Y \longrightarrow \mathbb{R}_{+\infty}$. Denote by F_X the union of the set of all functions $f\colon X \longrightarrow \mathbb{R}_{+\infty}$ and the function $-\infty$, where $-\infty(x) = -\infty$ for all $x \in X$. The Fenchel–Moreau conjugation corresponding to φ is the mapping $f \longrightarrow f^{\varphi}$ defined on F_X by

$$f^{\varphi}(y) = \sup_{x \in X}\{\varphi(x, y) - f(x)\}, \quad y \in Y\,.$$

Let φ' be the function defined on $Y \times X$ by $\varphi'(y, x) = \varphi(x, y)$. Then the Fenchel–Moreau conjugation corresponding to φ' is the mapping $g \longrightarrow g^{\varphi'}$ defined on F_Y by

$$g^{\varphi'}(x) := \sup_{y \in Y}\{\varphi\prime(y, x) - g(y)\} = \sup_{y \in Y}\{\varphi(x, y) - g(y)\}\,.$$

In the case where Y is a set of functions defined on set X, for each $y \in Y$ and $\gamma \in \mathbb{R}$, consider the function $h_{y,\gamma}(x) := y(x) - \gamma,\ x \in X$. Denote by H_Y the set $\{h_{y,\gamma}\colon y \in Y,\ \gamma \in \mathbb{R}\}$. Let $\varphi\colon X \times Y \longrightarrow \mathbb{R}_{+\infty}$ defined by $\varphi(x, y) = y(x)$. The following result is well known (see, e. g., [5], Theorem 8.8):

Theorem 1 *Let $f \in F_X$. Then $f^{\varphi\varphi'} = co_{H_Y} f$. In particular, $f^{\varphi\varphi'} = f$ if and only if f is H_Y-convex.*

ICAR Functions

Let X be a topological vector space. A set $K \subseteq X$ is called conic if $\lambda K \subseteq K$ for all $\lambda > 0$. We assume that X is equipped with a closed convex pointed cone K (the letter means that $K \cap (-K) = \{0\}$). The increasing property of our functions will be understood to be with respect to the ordering $\leq$ induced on X by K:

$$x \leq y \iff y - x \in K\,.$$

A function $f\colon X \longrightarrow \mathbb{R}_{+\infty}$ is called convex along rays (shortly CAR) if the function $f_x(\alpha) = f(\alpha x)$ is convex on the ray $[0, +\infty)$ for each $x \in X$. Similarly, f is called increasing along rays (shortly IAR) if the function $f_x(\alpha) = f(\alpha x)$ is increasing on the ray $[0, +\infty)$ for each $x \in X$. Also the function $f\colon X \longrightarrow \mathbb{R}_{+\infty}$ is called increasing if $x \geq y \implies f(x) \geq f(y)$ and it is called decreasing if $x \geq y \implies f(x) \leq f(y)$. In the sequel, we shall study the increasing convex-along-rays (ICAR) and decreasing convex-along-rays (DCAR) functions. Consider the coupling function $l\colon X \times X \longrightarrow \bar{\mathbb{R}}_+$ defined by

$$l(x, y) = \max\{\lambda \geq 0\colon \lambda y \leq x\}\,,$$

with the conventions $\max \emptyset := 0$ and $\max \mathbb{R} := +\infty$. This function is introduced and examined in [3]. We shall include some properties of l for the sake of completeness.

Proposition 1 *For every $x,\ x',\ y \in X$ and $\gamma > 0$, one has*

$$l(\gamma x, y) = \gamma\, l(x, y)\,, \tag{1}$$

$$l(x, \gamma y) = \frac{1}{\gamma} l(x, y)\,, \tag{2}$$

$$l(x, x) = 1 \iff x \notin -K\,, \tag{3}$$

$$x \leq x' \implies l(x, y) \leq l(x', y)\,, \tag{4}$$

$$x \in K,\ y \in -K \implies l(x, y) = +\infty\,. \tag{5}$$

Proof 1 We only prove (3). Note that, since $0 \in K$, we have $l(x,x) \geq 1$ for all $x \in X$. If $x \notin -K$, then $\lambda x \leq x$ for some $\lambda > 0$. This implies that $\lambda \leq 1$. Thus, $l(x,x) = 1$. Conversely, let $l(x,x) = 1$. Assume, toward a contradiction, that $x \in -K$. Then, $\lambda x \leq x$ for all $\lambda > 1$, and so $l(x,x) = +\infty > 1$, which contradicts the assumption that $l(x,x) = 1$. Hence, $x \notin -K$. □

In view of the above proposition, for example, when $x = y \in K$, the maximum in the definition of $l(x,y)$ is actually attained. The following proposition gives us a necessary condition for which $l(x,y)$ is finite.

Proposition 2 *If $y \notin -K$, then $l(x,y) < +\infty$ for all $x \in X$.*

Proof 2 If $l(x,y) = +\infty$, then there exists a sequence $\lambda_n \longrightarrow +\infty$ such that $y \leq (1/\lambda_n)x$. Hence $y \leq 0$. □

By ([3], Remark 2.1), $l_y \colon X \longrightarrow \bar{\mathbb{R}}_+$ is an IPH function for each $y \in X$. We also have the following proposition:

Proposition 3 *Let $y \notin -K$. The function $l_y : X \longrightarrow \bar{\mathbb{R}}_+$ is upper semicontinuous.*

Proof 3 Fix $x \in X$. Let $\{x_n\} \subset X$ be such that $x_n \longrightarrow x$. Set $\lambda = \overline{lim}\, l_y(x_n)$. If $\lambda = 0$, then, by the nonnegativity of l_y, we have $l_y(x) \geq \lambda$. Let $\lambda > 0$. It follows from $y \notin -K$ that $\lambda < +\infty$. Consider the subsequence $\{n_s\}_{s\geq 1}$ such that $l_y(x_{n_s}) > 0$ and $l_y(x_{n_s}) \longrightarrow \lambda$. We have $x_{n_s} - l_y(x_{n_s})y \in K$ for all $s \geq 1$. Since K is closed, we get $x - \lambda y \in K$ and so, by the definition of l, $l(x,y) \geq \lambda$. Hence l_y is upper semicontinuous. □

Set $X' = X \setminus (-K)$ and $L' = \{l_y \colon y \notin -K\}$. Fix $y \in X'$. Let $l \colon X \longrightarrow \bar{\mathbb{R}}_+$ defined by

$$l(x) := l_y(x)\,.$$

We have $\lambda l(x) = l(\lambda x)$ for all $\lambda \in [0, +\infty)$. (Note that $l_y(x) < +\infty$ for all $x \in X$). For each $x \in X$, consider the function $l^x \colon \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ defined by

$$l^x(t) := l(tx) = l_y(tx)\,.$$

It is not difficult to check that l^x is increasing and continuous. Hence the function l is ICAR, IAR, and continuous along rays.

Proposition 4 *Let $f \colon X \longrightarrow \mathbb{R}_{+\infty}$ be increasing and IAR. Then $f(x) = f(0)$ for all $x \leq 0$.*

Proof 4 Fix $x \in X$ such that $x \leq 0$. It follows from $x \leq 0$ and the monotonicity of f that $f(x) \leq f(0)$. On the other hand, since f is IAR, we have

$$f(x) = f_x(1) \geq f_x(0) = f(0)\,.$$

Hence $f(x) = f(0)$. □

We give an example of an increasing function that is not IAR.

Example 1 Consider the function $f \colon \mathbb{R}^3 \longrightarrow \mathbb{R}$ defined by

$$f(x) = \min_{1\leq i\leq 3} x_i, \quad \forall\, x \in \mathbb{R}^3\,.$$

Recall that $x \leq y$ if and only if $x_i \leq y_i$ for all $1 \leq i \leq 3$. It is easy to see that f is increasing but, if we set $x = (-2, 3, 4)$, then $f_x \colon [0, +\infty) \longrightarrow \mathbb{R}$ is not increasing. Therefore, f is not IAR.

The following functions are samples of ICAR and IAR functions.

Example 2 Consider the functions $f \colon \mathbb{R}^n \longrightarrow \mathbb{R}$ and $g \colon \mathbb{R}^n \longrightarrow \mathbb{R}$ defined by

$$f(x) = \begin{cases} \max_{1\leq i\leq n} x_i & x \notin \mathbb{R}^n_-, \\ 0 & \text{otherwise.} \end{cases}$$

and

$$g(x) = \exp(f)\,.$$

It is easy to check that f and g are ICAR and IAR with respect to coordinatewise ordering on $\mathbb{R}^n$.

Let $W = L' \cup \{\mathbf{0}\}$, where $\mathbf{0}(x) = 0$ for all $x \in X$. Consider the set $\mathrm{H} = \{l - \gamma \colon\ l \in W,\ \gamma \in \mathbb{R}\}$. We have the following result:

Theorem 2 *Let $f \colon X \longrightarrow \mathbb{R}_{+\infty}$ be a function. Then f is ICAR, IAR, and lscAR if and only if it is H-convex.*

Proof 5 It is clear that each function $h \in \mathrm{H}$ is ICAR, IAR, and continuous along rays. Therefore each H-convex function is ICAR, IAR, and *lsc*AR. Conversely, let $f \colon X \longrightarrow \mathbb{R}_{+\infty}$ be an ICAR, IAR, and *lsc*AR function. Consider $y \in X$. Since f_y is increasing, convex, and *lsc*, it follows from ([5], Lemma 3.1) that there exists a set $V_y \subseteq \mathbb{R}_+ \times \mathbb{R}$ such that $f_y(t) = \sup_{v\in V_y}\{v_1 t - v_2\}$, for each $t \geq 0$, where

$v = (v_1, v_2) \in V_y$. First, we suppose that $y \notin -K$. For $v = (v_1, v_2) \in V_y$, we set

$$h^v(x) = v_1 l_y(x) - v_2, \quad x \in X.$$

Clearly $h^v \in \mathrm{H}$. Let $v_1 = 0$ or $l_y(x) = 0$. Since f is IAR, we have

$$f(x) = f_x(1) \geq f_x(0) = f(0) = f_y(0) \geq -v_2 = h^v(x), \quad \text{for all } x \in X.$$

Suppose now that $v_1 > 0$ and $l_y(x) > 0$. (Note that $l_y(x) < +\infty$). Since $x - l_y(x)y \in K$ and f is increasing, it follows that

$$h^v(x) \leq f_y(l_y(x)) = f(l_y(x)y) \leq f(x).$$

Thus, $h^v(x) \leq f(x)$ for all $x \in X$, that is, $h^v \in supp(f, \mathrm{H})$, where

$$supp(f, \mathrm{H}) = \{h \in \mathrm{H} \colon h(x) \leq f(x), \quad \forall x \in X\}.$$

Therefore

$$\sup\{h^v(y), v \in V_y\} \leq \sup\{h(y) \colon h \in supp(f, \mathrm{H})\} \leq f(y). \tag{6}$$

On the other hand, in view of (3), we have

$$f(y) = f_y(1) = \sup_{v \in V_y}(v_1 - v_2) = \sup_{v \in V_y}(v_1 l_y(y) - v_2) = \sup_{v \in V_y} h^v(y).$$

Thus, $f(y) = \sup\{h(y) \colon h \in supp(f, \mathrm{H})\}$. We now assume that $y \in -K$. By Proposition 4, we have $f(y) = f(0)$. It follows from the proof of ([5], Lemma 3.1) that $v_1 = 0$ for all $v \in V_y$. Consider $v \in V_y$. We set

$$h^v(x) = -v_2, \quad x \in X.$$

Then

$$f(x) = f_x(1) \geq f(0) = \sup_{v \in V_y} -v_2 \geq h^v(x), \quad \text{for all } x \in X$$

and

$$f(y) = f(0) = \sup_{v \in V_y} -v_2 = \sup_{v \in V_y} h^v(y). \tag{7}$$

Hence $\{h^v \colon v \in V_y\} \subseteq supp(f, \mathrm{H})$, and in view of (6) and (7), we get $f(y) = \sup\{h(y) \colon h \in supp(f, \mathrm{H})\}$. This completes the proof. □

It follows from ([3], Proposition 3.3) that there exists a bijection ψ from $X' \cup \{0\}$ onto W. Therefore we can identify W with $Y = X' \cup \{0\}$ by means of the mapping ψ.

We define the coupling function $\varphi \colon X \times Y \to \mathbb{R}_{+\infty}$ by

$$\varphi(x, y) = \begin{cases} l_y(x) & y \in X', \\ \mathbf{0} & y = 0. \end{cases} \tag{8}$$

Combining Theorem 1 with Theorem 2, one gets:

Theorem 3 *A function $f \colon X \longrightarrow \mathbb{R}_{+\infty}$ is ICAR, IAR, and lscAR if and only if $f = f^{\varphi\varphi'}$.*

Subdifferentiability of ICAR Functions

Consider the subdifferential ∂_W of $f \colon X \longrightarrow \mathbb{R}_{+\infty}$ at point $x_0 \in domf$:

$$\partial_W f(x_0) = \{h \in W \colon f(x) \geq f(x_0) + h(x) - h(x_0) \quad \forall x \in X\}.$$

We have the following result:

Proposition 5 *Let $f \colon X \longrightarrow \mathbb{R}_{+\infty}$ be an ICAR and IAR function and $x_0 \in X'$. If $\lambda x_0 \in domf$ for some $\lambda > 1$, then $\partial_W f(x_0)$ is nonempty and we have*

$$\{\tau l_{x_0} \colon \tau \in \partial f_{x_0}(1)\} \subseteq \partial_W f(x_0), \tag{9}$$

where $f_{x_0}(\alpha) = f(\alpha x_0)$. Moreover, iff is strictly increasing at point x_0 (i.e., $x \in X$, $x \leq x_0$ and $x \neq x_0$ imply $f(x) < f_{x0}$) and f_{x0} is strictly increasing at point $\alpha = 1$, by replacing ∂_W with $\partial_{L'}$, then equality holds in (9).

Proof 6 Since the increasing convex function f_{x0} is continuous at 1, it has a subgradient $\tau \geq 0$ at this point. Thus

$$\tau t - \tau 1 \leq f_{x_0}(t) - f_{x_0}(1) \quad \text{for all } t \geq 0. \tag{10}$$

Let $x \in X$ be arbitrary. If $l_{x_0}(x) = 0$, then by setting $t = l_{x_0}(x) = 0$ in (10), we have

$$\tau l_{x_0}(x) - \tau 1 \leq f_{x_0}(0) - f_{x_0}(1).$$

Since $l_{x_0}(x_0) = 1$ and f is IAR, we get

$$\tau l_{x_0}(x) - \tau l_{x_0}(x_0) \leq f(x) - f(x_0)\,.$$

Assume now that $l_{x_0}(x) > 0$. We have $l_{x_0}(x)x_0 \leq x$. In view of the monotonicity of f, one has

$$\begin{aligned} f(x) &\geq f(l_{x_0}(x)x_0) = f_{x_0}(l_{x_0}(x)) \\ &\geq f_{x_0}(1) + \tau l_{x_0}(x) - \tau\,. \end{aligned}$$

Thus

$$f(x) - f(x_0) \geq \tau l_{x_0}(x) - \tau l_{x_0}(x_0)\,,$$

which shows that $\tau l_{x_0} \in \partial_W f(x_0)$. This implies that $\partial_W f(x_0) \neq \emptyset$ and we have (9). Now, let f_{x0} be strictly increasing at $\alpha = 1$ and f be strictly increasing at x_0. In this case τ is different from zero in the left-hand side of (9). Consider $l \in \partial_{L'} f(x_0)$. Then there exists $y \in X'$ such that $l = l_y$. We have

$$f(x) - f(x_0) \geq l_y(x) - l_y(x_0), \quad \text{for all } x \in X\,. \tag{11}$$

We will show that $l_y(x_0) > 0$. Reasoning by contradiction, let us assume that $l_y(x_0) = 0$. It follows from (11) that $f_{x_0}(t) - f_{x_0}(1) \geq 0$ for all $t \geq 0$. Since f_{x0} is strictly increasing at point 1, we get a contradiction. Thus $l_y(x_0) > 0$. Since $l_y(x_0)y \leq x_0$, one has

$$\begin{aligned} f(x_0) \geq f(l_y(x_0)y) &\geq f(x_0) + l_y(l_y(x_0)y) - l_y(x_0) \\ &= f(x_0) + l_y(x_0)l_y(y) - l_y(x_0)\,, \\ &= f(x_0)\,. \end{aligned}$$

Hence $f(x_0) = f(l_y(x_0)y)$. Since f is strictly increasing at x_0, we get $l_y(x_0)y = x_0$ or $y = x_0/(l_y(x_0))$. Moreover, for all $t \geq 0$, by (11), we have

$$\begin{aligned} f_{x_0}(t) = f(tx_0) &\geq f(x_0) + l_y(tx_0) - l_y(x_0)\,, \\ &= f_{x_0}(1) + l_y(x_0)(t-1)\,, \end{aligned}$$

which shows that $l_y(x_0) \in \partial f_{x_0}(1)$. We set $\tau = l_y(x_0)$. Then $l = l_{x_0/\tau} = \tau l_{x_0}$. This completes the proof. □

Proposition 6 *Let $f\colon X \longrightarrow \mathbb{R}_{+\infty}$ be an ICAR and IAR function. If $x \in X'$ is a point such that the one-sided derivative $f'(x, x) = 0$, then x is a global minimizer of f over X.*

Proof 7 Since the function f_x is convex and $f_x'(1) = f'(x, x) = 0$, we have $f_x(1) = \min_{t\in[0,+\infty)} f_x(t)$. Thus $f_x(1) \leq f_x(0)$. On the other hand, since f is IAR, $f(x) = f_x(1) \geq f_x(0) = f(0)$. Hence $f(x) = f(0) = \min_{x\in X} f(x)$. □

Recall that, by ([3], Proposition 3.3) , we can identify L' with X' by means of the mapping ψ'. Let us denote by $\partial_{X'} f(x_0)$ the set $(\psi')^{-1}(\partial_{L'} f(x_0))$. Then

$$\begin{aligned} &\partial_{X'} f(x_0) \\ &= \{y \in X'\colon l_y(x) - l_y(x_0) \leq f(x) - f(x_0),\, \forall x \in X\}\,. \end{aligned}$$

Proposition 7 *Let $f\colon X \longrightarrow \mathbb{R}$ be an ICAR and IAR function. Then*

$$\partial_{X'} f(0) = \{y \in X'\colon l_y(x) \leq (f_x)'_+(0), \quad \forall x \in X'\}\,,$$

where $(f_x)'_+$ is the right derivative of the function f_x given by $f_x(\alpha) = f(\alpha x)$.

Proof 8 For each $y \in X'$, one has

$$\begin{aligned} y \in \partial_{X'} f(0) &\iff l_y(x) - l_y(0) \leq f(x) - f(0), \\ &\qquad \forall x \in X \\ &\iff l_y(\alpha x) \leq f_x(\alpha) - f_x(0), \\ &\qquad \forall x \in X',\ \forall \alpha > 0 \\ &\iff l_y(x) \leq \frac{f_x(\alpha) - f_x(0)}{\alpha}, \\ &\qquad \forall x \in X',\ \forall \alpha > 0 \\ &\iff l_y(x) \leq (f_x)'_+(0), \quad \forall x \in X'\,. \end{aligned}$$

The second of equivalence is a consequence of Proposition 1 and ([3], Remark 2.1). □

DCAR Functions

In this section, we shall study decreasing convex-along-rays (DCAR) functions defined on X. To this end, we introduce the coupling function $u\colon X \times X \longrightarrow \bar{\mathbb{R}}$ defined by

$$u(x, y) = \max\{\lambda \in \mathbb{R}\colon x \leq \lambda y\}\,.$$

Let x, x', $y \in X$ be arbitrary and $\gamma > 0$. It is easy to check that the function u has the following properties:

(1) $x \leq x' \implies u(x', y) \leq u(x, y)$,
(2) $u(\gamma x, y) = \gamma u(x, y)$,
(3) $u(x, y) = +\infty \implies y \in K$.

We have also $u(-x, -y) = l(x, y)$. For each $y \in X$, consider the cone

$$Q_y = \{x \in X\colon 0 \leq u(x, y) \leq +\infty\}\,.$$

Lemma 1 *Let $y \in X$ and $x,\ x' \in Q_y$. The following inequality holds:*

$$u(x + x', y) \geq u(x, y) + u(x', y)\,. \tag{12}$$

Proof 9 Let $A = \{\alpha \in \mathbb{R}\colon \alpha x \leq y\}$, $B = \{\beta \in \mathbb{R}\colon \beta x' \leq y\}$ and $C = \{\gamma \in \mathbb{R}\colon \gamma(x + x') \leq y\}$. In view of the transitive property of the relation $\leq$, we get $A + B \subseteq C$ and this yields (12). □

It follows from the properties of u and Lemma 1 that Q_y is a downward convex cone. Fix $y \in X$. We define the function $r_y\colon X \longrightarrow \bar{\mathbb{R}}_+$ by

$$r_y(x) = \begin{cases} u(x, y) & x \in Q_y\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{13}$$

By the properties of u, we get r_y is a decreasing and positively homogenous function of degree one. Let $y \notin K$ and set $r = r_y$. It is not difficult to see that the function $r^x\colon \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ defined by $r^x(t) = r(tx)$ is increasing, convex, and continuous. Thus, for each $y \notin K$, the function r_y is DCAR, IAR, and continuous along rays.

Example 3 Let $X = \mathbb{R}^n$ and K be the cone $\mathbb{R}^n_+$ of all vectors in $\mathbb{R}^n$ with nonnegative coordinates. Let $I = \{1, 2, \ldots, n\}$. Each vector $x \in \mathbb{R}^n$ generates the following sets of indices:

$$I_+(x) = \{i \in I\colon x_i > 0\}\,,$$
$$I_0(x) = \{i \in I\colon x_i = 0\}\,,$$
$$I_-(x) = \{i \in I\colon x_i < 0\}\,.$$

Let $x \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Denote by $\frac{c}{x}$ the vector with coordinates

$$\left(\frac{c}{x}\right)_i = \begin{cases} \frac{c}{x_i}, & i \notin I_0(x), \\ 0, & i \in I_0(x). \end{cases}$$

Then, for each $x,\ y \in \mathbb{R}^n_+$, we have

$$r_y(x) = \begin{cases} \min_{i \in I_-(y)} \dfrac{x_i}{y_i}, & x \in Q_y\,, \\ 0, & x \notin Q_y, \end{cases}$$

where

$$Q_y = \left\{ x \in \mathbb{R}^n\colon I_0(y) \cup I_-(y) \subseteq I_0(x) \cup I_-(x); \right.$$
$$\left. \max_{i \in I_+(y)} \frac{x_i}{y_i} \leq \min_{i \in I_-(y)} \frac{x_i}{y_i} \right\}\,.$$

Example 4 Consider the function $g\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ defined by

$$g(x) = \begin{cases} -\min_{1 \leq i \leq n} x_i & x \notin \mathbb{R}^n_+\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

It is easy to check that g is DCAR and IAR.

Now, let

$$U = \{r_y\colon y \in (X \setminus K)\} \cup \{\mathbf{0}\}$$

and

$$H_U = \{h - \gamma\colon h \in U,\ \gamma \in \mathbb{R}\}\,.$$

It is clear that each H_U-convex function is DCAR, IAR, and *lsc*AR. The proof of the following theorem is similar to that of Theorem 2, and therefore we omit its proof.

Theorem 4 *Let $f\colon X \longrightarrow \mathbb{R}_{+\infty}$ be a DCAR, IAR, and lscAR function. Then f is H_U-convex.*

Corollary 1 *The function $f\colon X \longrightarrow \mathbb{R}_{+\infty}$ is H_U-convex if and only if it is DCAR, IAR, and lscAR.*

References

1. Dutta J, Martinez-Legaz JE, Rubinov AM (2004) Monotonic analysis over cones: I. Optimization 53:129–146
2. Dutta J, Martinez-Legaz JE, Rubinov AM (2004) Monotonic analysis over cones: II. Optimization 53:529–547
3. Mohebi H, Sadeghi H (2007) Monotonic analysis over ordered topological vector spaces: I. Optimization 56(3): 305–321
4. Rockafellar RT (1970) Convex Analysis. Princeton University Press, Princeton, NJ
5. Rubinov AM (2000) Abstract Convex Analysis and Global Optimization. Kluwer, Dordrecht
6. Singer I (1997) Abstract Convex Analysis. Wiley, New York

Increasing and Positively Homogeneous Functions on Topological Vector Spaces

Hossein Mohebi
Mahani Mathematical Research Center,
and Department of Mathematics,
University of Kerman, Kerman, Iran

MSC2000: 26A48, 52A07, 26A51

Article Outline

Keywords and Phrases
Introduction
Characterizations of Nonnegative IPH Functions
Abstract Convexity of Nonnegative IPH Functions
Abstract Concavity of DPH Functions
References

Keywords and Phrases

Monotonic analysis; IPH functions; Downward sets; Upward sets; Abstract convexity

Introduction

We study IPH (increasing and positively homogeneous of degree one) functions defined on a topological vector space X that is equipped with a closed convex pointed cone K (see [7]). The theory of IPH functions defined on a closed convex cone K in a topological vector space X is well developed [2]. There are two main results of this theory, which have a central role. First, each IPH function p defined on K can be represented as the Minkowski gauge of a normal closed along rays (Definitions 1 and 2) subset U of K, namely, $U = \{x \in K\colon p(x) \le 1\}$. Conversely, the Minkowski gauge of a normal closed along rays set is an IPH function. The second result is based on ideas of abstract convexity: each IPH function defined on K can be represented as the upper envelope of a set of so-called min-type functions. This result can be considered as a certain form of a dual representation of IPH functions. IPH functions can be useful for the description of radiant and coradiant downward sets and, through them, can have applications to the study of some NTU games arising in mathematical economics [1,13] and to the analysis of topical functions, which are used in the analysis of discrete event systems [3,4,5]. Some of the results related to monotonic analysis on the space $\mathbb{R}^n$ with respect to the coordinatewise order relation can be found in [5,6,10,11]. IPH functions defined on $\mathbb{R}^n$ are examined in [5]. Nevertheless, as it turned out, the main results from [5] can be extended for arbitrary topological vector spaces. Such extension is one of the main topics of this article. The study of some problems in a more general framework clarifies and simplifies the main ideas and approaches. The results obtained can be used, for example, in the study of vector optimization problems.

Let X be a topological vector space. We assume that X is equipped with a closed convex pointed cone K (the latter means that $K \cap (-K) = \{0\}$). The increasing property of our functions will be understood to be with respect to the ordering $\le$ induced on X by K:

$$x \le y \iff y - x \in K\,.$$

We use the following notations:

$$\begin{aligned}
&\mathbb{R} = (-\infty,\ +\infty),\\
&\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty\} \cup \{+\infty\},\\
&\mathbb{R}_{+\infty} = (-\infty,\ +\infty],\\
&\mathbb{R}_{+} = [0,\ +\infty),\\
&\bar{\mathbb{R}}_{+} = [0,\ +\infty],\\
&\mathbb{R}_{-} = (-\infty, 0],\\
&\bar{\mathbb{R}}_{-} = [-\infty, 0]\,.
\end{aligned}$$

We accept the following conventions:

$$\frac{1}{0} = \infty,\quad \frac{1}{\infty} = 0\,. \tag{1}$$

A function $p\colon X \longrightarrow \bar{\mathbb{R}}$ is called positively homogeneous if $p(\lambda x) = \lambda p(x)$ for all $x \in X$ and $\lambda \ge 0$. Function p is called increasing if $x \ge y \implies p(x) \ge p(y)$. We shall study IPH (increasing and positively homogeneous of degree one) functions p such that $0 \in dom\ p := \{x \in X\colon -\infty < p(x) < +\infty\}$, and hence we have $p(0) = 0$.

The following definitions can be found in [9].

Definition 1 A nonempty subset W of X is called closed along rays if $(x \in W,\ \lambda_n > 0,\ \lambda_n x \in W,\ n = 1, 2, \ldots,\ \lambda_n \longrightarrow \lambda,\ \lambda > 0) \implies \lambda x \in W$.

Definition 2 A nonempty subset A of K is called normal if $x \in A$, $x' \in K$ and $x' \le x$ imply $x' \in A$.

Definition 3 A nonempty subset B of K is called conormal if $x \in B$, $x' \in K$ and $x \le x'$ imply $x' \in B$.

A normal subset A of K is radiant, that is, $x \in A$ and $0 < \lambda < 1$ imply $\lambda x \in A$. A conormal subset B of K is coradiant, that is, $x \in B$ and $\lambda > 1$ imply $\lambda x \in B$. A set $W \subseteq X$ is called downward if $(x \in W,\ x' \le x) \implies x' \in W$. (In particular, the empty set is downward). Similarly, a set $V \subseteq X$ is called

upward if $(x' \in V,\ x' \le x) \Longrightarrow x \in V$. Let $W \subseteq X$ be a radiant set. The Minkowski gauge $\mu_W : X \longrightarrow \bar{\mathbb{R}}_+$ of this set is defined by

$$\mu_W(x) = \inf\{\lambda > 0 : \frac{x}{\lambda} \in W\}. \tag{2}$$

The Minkowski cogauge $\nu_V : X \longrightarrow \bar{\mathbb{R}}_+$ of a coradiant set $V \subseteq X$ is defined by

$$\nu_V(x) = \sup\{\lambda > 0 : \frac{x}{\lambda} \in V\}. \tag{3}$$

It is easy to check that the Minkowski gauge of a downward set and the Minkowski cogauge of an upward set are IPH.

Consider the function $l : K \times K \longrightarrow \bar{\mathbb{R}}_+$ defined by

$$l(x, y) = \max\{\lambda \in \mathbb{R}_+ : \lambda y \le x\}.$$

This function is introduced and examined in [2]. To motivate our study, we characterize the IPH functions defined on K.

Theorem 1 ([2], Theorem 16) *Let $p : K \longrightarrow \mathbb{R}_+$ be a function. Then p is IPH if and only if $p(x) \ge l(x, y)p(y)$ for all $x,\ y \in K$, with the convention $(+\infty) \times 0 = 0$.*

Characterizations of Nonnegative IPH Functions

Carrying forward the motivation from Theorem 1, we shall now proceed to develop a similar type of property for IPH functions $p : X \longrightarrow \bar{\mathbb{R}}_+$. To achieve this, we need to introduce the coupling function $l : X \times X \longrightarrow \bar{\mathbb{R}}_+$ defined by

$$l(x, y) := \max\{\lambda \ge 0 :\ \lambda y \le x\} \tag{4}$$

(we use the conventions $\max \emptyset := 0$ and $\max \mathbb{R}_+ := +\infty$). The next proposition gives some properties of the coupling function l.

Proposition 1 *For every $x,\ x',\ y \in X$ and $\gamma > 0$, one has*

$$l(\gamma x, y) = \gamma l(x, y), \tag{5}$$

$$l(x, \gamma y) = \frac{1}{\gamma} l(x, y), \tag{6}$$

$$l(x, y) = +\infty \iff y \in -K, \tag{7}$$

$$l(x, x) = 1 \iff x \notin -K, \tag{8}$$

$$x \in K,\ y \in -K \Longrightarrow l(x, y) = +\infty, \tag{9}$$

$$x \le x' \Longrightarrow l(x, y) \le l(x', y), \tag{10}$$

$$y \le y' \Longrightarrow l(x, y) \ge l(x, y'). \tag{11}$$

Proof We only prove parts (7) and (10). Let $l(x, y) = +\infty$ for some $x,\ y \in X$. By (4) there exists a sequence $\{\lambda_n\}_{n \ge 1}$ such that $\lambda_n \longrightarrow +\infty$ and $y \le 1/\lambda_n x$ for all $n \ge 1$. Since K is a closed cone, we get $y \le 0$. This proves (7). To prove (10), let $x \le x'$, $\Lambda_{x,y} = \{\lambda \ge 0 : \lambda y \le x\}$ and $\Lambda_{x',y} = \{\gamma \ge 0 : \gamma y \le x'\}$. It is clear that $\Lambda_{x,y} \subseteq \Lambda_{x',y}$ (notice that if $\Lambda_{x',y} = \emptyset$, then $\Lambda_{x,y} = \emptyset$). Hence $l(x, y) \le l(x', y)$. □

Example 1 Let $X = \mathbb{R}^n$ and K be the cone $\mathbb{R}^n_+$ of all vectors in $\mathbb{R}^n$ with nonnegative coordinates. Let $I = \{1, 2, \ldots, n\}$. Each vector $x \in \mathbb{R}^n$ generates the following sets of indices:

$$I_+(x) = \{i \in I : x_i > 0\},$$
$$I_0(x) = \{i \in Ix_i = 0\},\ \ I_-(x) = \{i \in I : x_i < 0\}.$$

Let $x \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Denote by c/x the vector with coordinates

$$\left(\frac{c}{x}\right)_i = \begin{cases} \frac{c}{x_i}, & i \notin I_0(x), \\ 0, & i \in I_0(x). \end{cases}$$

Then, for each $x,\ y \in \mathbb{R}^n$, we have

$$l(x, y) = \begin{cases} \min_{i \in I_+(y)} \frac{x_i}{y_i}, & x \in K^+_y, \\ 0, & x \notin K^+_y, \end{cases}$$

where

$$K^+_y = \{x \in \mathbb{R}^n :\ \forall\, i \in I_+(y) \cup I_0(y),\ \ x_i \ge 0;\ \max_{i \in I_+(y)} \frac{x_i}{y_i} \le \min_{i \in I_-(y)} \frac{x_i}{y_i}\}.$$

We also need to introduce the coupling function $u : X \times X \longrightarrow \bar{\mathbb{R}}_+$ defined by

$$u(x, y) := \min\{\mu \ge 0 :\ x \le \mu y\} \tag{12}$$

(with the convention $\min \emptyset := +\infty$).

The following proposition gives some properties of the coupling function u.

Proposition 2 *For every $x,\ x', y \in X$ and $\gamma > 0$, one has*

$$u(\gamma x, y) = \gamma u(x, y)\,, \tag{13}$$

$$u(x, \gamma y) = \frac{1}{\gamma} u(x, y)\,, \tag{14}$$

$$u(x, y) = 0 \iff x \in -K\,, \tag{15}$$

$$u(x, x) = 1 \iff x \notin -K\,, \tag{16}$$

$$x \le x' \implies u(x, y) \le u(x', y), \tag{17}$$

$$y \le y' \implies u(x, y) \ge u(x, y')\,. \tag{18}$$

Example 2 Let $X = \mathbb{R}^n$ and K be the cone $\mathbb{R}^n_+$ of all vectors in $\mathbb{R}^n$ with nonnegative coordinates. Then, for each $x,\ y \in \mathbb{R}^n$, we have

$$u(x, y) = \begin{cases} \max_{i \in I_+(y)} \frac{x_i}{y_i}, & x \in c_y^+, \\ 0, & x \notin c_y^+, \end{cases}$$

where

$$c_y^+ = \{x \in \mathbb{R}^n\colon\ \exists\ i \in I_+(y) \text{ s.t. } x_i \ge 0 \quad \text{and} \quad \forall\ i \in I_-(y) \cup I_0(y),\ \ x_i \le 0\}.$$

Theorem 2 *Let $p\colon X \longrightarrow \bar{\mathbb{R}}_+$ be a function. Then the following assertions are equivalent:*

(i) *p is IPH.*

(ii) *$p(x) \ge \lambda p(y)$ for all $x,\ y \in X$, and $\lambda > 0$ such that $\lambda y \le x$.*

(iii) *$p(x) \ge l(x, y)p(y)$ for all $x,\ y \in X$, with the convention $(+\infty) \times 0 = 0$.*

(iv) *$p(x) \le u(x, y)p(y)$ for all $x, y \in X$, with the convention $(+\infty) \times 0 = +\infty$.*

Proof It is clear that (i) implies (ii). To prove the implication (ii) $\longrightarrow$ (iii), notice first that, due to (7), $l(x, y) = +\infty$ implies that $y \in -K$ and so $p(y) = 0$. Then, by the convention $(+\infty) \times 0 = 0$, we have $p(x) \ge l(x, y)p(y)$. If $l(x, y) = 0$, then, by the nonnegativity of p, we get $p(x) \ge l(x, y)p(y)$. Finally, let $0 < l(x, y) < +\infty$. Then in view of (4) and the closedness of K, we have $x \ge l(x, y)y$, and so (ii) implies (iii). We shall now prove the implication (iii) $\longrightarrow$ (i). Consider $x,\ y \in X$ such that $y \le x$. By (4) we get $l(x, y) \ge 1$. Then (iii) yields that $p(x) \ge p(y)$. Hence p is increasing. Let $x \in X,\ \lambda > 0$ and $l(x, \lambda x) = +\infty$. It follows from (6) and (7) that $x,\ \lambda x \in -K$. Since p is increasing, we get $\lambda p(x) = p(\lambda x) = 0$. Let $x \notin -K$ and set $y = \lambda x$. Then, by (6) and (8), we have $l(x, \lambda x) = 1/\lambda$. Thus $p(x) \ge 1/\lambda\, p(\lambda x)$, and so $\lambda p(x) \ge p(\lambda x)$. By replacing λ with $1/\lambda$ and x with λx, we obtain $p(\lambda x) \ge \lambda p(x)$. This proves that p is positively homogeneous. We next prove the implication (i) $\longrightarrow$ (iv). Let $u(x, y) = 0$. By (15) we get $x \le 0$. Then $p(x) = 0$, and so $p(x) \le u(x, y)p(y)$. If $u(x, y) = +\infty$, then, in view of the convention $(+\infty) \times 0 = +\infty$, we have $p(x) \le u(x, y)p(y)$. We now assume that $0 < u(x, y) < +\infty$. Then, in view of (12) and the closedness of K, we get $x \le u(x, y)y$. Hence $p(x) \le u(x, y)p(y)$. Finally, the proof of the implication (iv) $\longrightarrow$ (i) can be done in a manner analogous to that of the implication (iii) $\longrightarrow$ (i). □

We shall now describe a class of elementary functions with respect to which the IPH functions are supremally generated. Given $y \in X$, let us set $l_y(x) := l(x, y)$ for all $x \in X$. Thus, by (4),

$$l_y(x) = \max\{\lambda \ge 0\colon \lambda y \le x\}, \quad \forall\ x \in X. \tag{19}$$

Remark 1 The function $l_y\colon X \longrightarrow \bar{\mathbb{R}}_+$ is an IPH function for each $y \in X$. It obviously follows from (5) and (10).

Let L be the set of all supremally generating elementary functions, defined by (19), that is,

$$L := \{l_y : y \in X\}\,. \tag{20}$$

Consider the mapping $\psi : X \longrightarrow L$ defined by

$$\psi(y) = l_y, \qquad y \in X.$$

We have the following proposition:

Proposition 3 *The mapping $\psi : X \longrightarrow L$ is onto. Moreover, it is antitone:*

$$y_1 \le y_2 \implies l_{y_2} \le l_{y_1} \tag{21}$$

and antihomogeneous (positively homogeneous of degree -1):

$$l_{\lambda y} = \lambda^{-1} l_y \quad \forall\ y \in X, \quad \forall\ \lambda > 0\,. \tag{22}$$

Proof By the definition of L, ψ is obviously onto. Implications (21) and (22) follow from (11) and (6), respectively. □

The following example shows that the mapping ψ is not one-to-one.

Example 3 Let $X = \mathbb{R}^2$ and $K = \mathbb{R}^2_+$. Consider the distinct points $y_1 = (a, b)$ and $y_2 = (c, d)$, where a, b, c and d are negative numbers. By the results obtained in Example 1, we have $K^+_{y_1} = K^+_{y_2} = X$. Since $I_+(y_1) = I_+(y_2) = \emptyset$, it follows that $l_{y_1}(x) = l_{y_2}(x) = +\infty$ for every $x \in X$. Thus, $l_{y_1} = l_{y_2}$.

Let $X' = X \setminus (-K)$ and $L' = \{l_y : y \in X'\}$. We can get:

Proposition 4 *The mapping $\psi' = \psi|_{X'}$ is a bijection from X' onto L', where $\psi|_{X'}$ is the restriction of ψ to X'.*

Proof Since, by the definition of L', ψ' is obviously onto, we only have to prove that ψ' is one-to-one. To this aim, assume that $y_1, y_2 \in X'$ are such that $l_{y_1} = l_{y_2}$. By (8), we have $1 = l(y_1, y_1) = l(y_1, y_2)$. Hence, by (4), we get $y_2 \preceq y_1$. By symmetry it follows that $y_1 \preceq y_2$. Since K is pointed, we conclude that $y_1 = y_2$. □

Recall (see [8]) that a function $p : X \longrightarrow \bar{\mathbb{R}}_+$ is called abstract convex with respect to the set L or L-convex if and only if there exists a set $W \subseteq L$ such that $p(x) = \sup_{l \in W} l(x)$. If $W \subseteq L'$, then using ψ' we can identify W with some subset of X. In terms of X, p is L'-convex if there exists a subset $Y \subseteq X'$ such that $p(x) = \sup_{y \in Y} l_y(x)$. It follows from Remark 1 that L consists of nonnegative IPH functions, hence each L-convex function is IPH.

Theorem 3 *Let $p : X \longrightarrow \bar{\mathbb{R}}_+$ be a function and L be the set described by Eq. (20). Then p is IPH if and only if there exists a set $Y \subseteq X$ such that*

$$p(x) = \max_{y \in Y} l_y(x) \quad \forall\, x \in X$$

(with the convention $\max \emptyset := 0$). In this case, one can take $Y = \{y \in X : p(y) \geq 1\}$. Hence, $p : X \longrightarrow \bar{\mathbb{R}}_+$ is IPH if and only if it is L-convex.

Proof We shall only show that every IPH function $p : X \longrightarrow \bar{\mathbb{R}}_+$ satisfies $p(x) = \max_{y \in Y} l_y(x)$, for all $x \in X$, with

$$Y = \{y \in X : p(y) \geq 1\}.$$

It is clear that $Y \cap (-K) = \emptyset$. For any x, $y \in X$ with $p(y) \geq 1$, it follows from Theorem 2 that $p(x) \geq l_y(x)$. This means that $p \geq l_y$ for all $y \in Y$, and so $p \geq \max_{y \in Y} l_y$. If $p(x) = 0$, then, by nonnegativity of the function l_y, we have $\max_{y \in Y} l_y(x) = 0 = p(x)$. Assume now that $0 < p(x) < +\infty$. Since $p(x/p(x)) = 1$, we get $x/p(x) \in Y$. Moreover, it follows from (6) that $p(x) = l(x, x/p(x))$. Therefore, $p(x) = \max_{y \in Y} l_y(x)$. Finally, suppose that $p(x) = +\infty$. It follows from the positive homogeneity of p that $(1/\lambda)x \in Y$ for all $\lambda > 0$. Then, $\max_{y \in Y} l_y(x) \geq l_{(1/\lambda)x}(x) = \lambda$ for all $\lambda > 0$. This means that $\max_{y \in Y} l_y(x) = +\infty = p(x)$. This completes the proof. □

The IPH functions are also infimally generated by the elementary functions $u_y : X \longrightarrow \bar{\mathbb{R}}_+$, $y \in X$, defined by

$$u_y(x) := u(x, y) = \min\{\mu \geq 0 : x \preceq \mu y\}, \quad \forall\, x \in X.$$

In view of (13) and (17), it is clear that the function u_y is IPH. Set

$$U := \{u_y : y \in X\}. \tag{23}$$

We define the mapping $\varphi : X \longrightarrow \bar{\mathbb{R}}_+$ by

$$\varphi(y) := u_y, \qquad y \in X.$$

We omit the proof of the following results, which are similar to those of Propositions 3 and 4.

Proposition 5 *The mapping $\varphi : X \longrightarrow U$ is onto. Moreover, it is antitone and antihomogeneous (positively homogeneous of degree -1).*

Let $U' = \{u_y : y \in X'\}$. We can get:

Proposition 6 *The mapping $\varphi' = \varphi|_{X'}$ is a bijection from X' onto U', where $\varphi|_{X'}$ is the restriction of φ to X'.*

A function $p : X \longrightarrow \bar{\mathbb{R}}_+$ is called abstract concave with respect to the set U, or U-concave, if there exists a set $W \subseteq U$ such that $p(x) = \inf_{u \in W} u(x)$. Since U consists of nonnegative IPH functions, we get each U-concave function is IPH.

The proof of the following theorem can be done in a manner analogous to the one of Theorem 3.

Theorem 4 *Let $p : X \longrightarrow \bar{\mathbb{R}}_+$ be a function and U be the set described by (23). Then p is IPH if and only if*

there exists a set $W \subseteq U$ such that

$$p(x) = \min_{u_y \in W} u_y(x) \qquad \forall\, x \in X\,.$$

In this case, one can take $W = \{u_y\colon\ y \in X,\ p(y) \le 1\}$. Hence $p : X \longrightarrow \bar{\mathbb{R}}_+$ is IPH if and only if it is U-concave.

Abstract Convexity of Nonnegative IPH Functions

We are now going to develop an abstract convexity (resp. abstract concavity) approach to IPH functions. The set L (resp. U) will play the role of the conjugate space in the usual linear model, while IPH functions will be regarded as analogous to sublinear functions. The well-known dual object related to a sublinear function is the so-called polar function (see, for example, [12,14]). We now give an analog of this concept for IPH functions and define also a related notion of polar set of a set $W \subset X$.

Definition 4 The lower polar function of $p\colon X \longrightarrow \bar{\mathbb{R}}_+$ is the function $p^0\colon L \longrightarrow \bar{\mathbb{R}}_+$ defined by

$$p^0(l_y) = \sup_{x \in X} \frac{l_y(x)}{p(x)}, \quad l_y \in L \tag{24}$$

(with the conventions $0/0 = 0$ and $\infty/\infty = 0$).

Theorem 5 *Let $p : X \longrightarrow \bar{\mathbb{R}}_+$ be a function. Then*

$$p^0(l_y) \ge \frac{1}{p(y)} \quad \forall\, l_y \in L\,, \tag{25}$$

and p is IPH if and only if

$$p^0(l_y) = \frac{1}{p(y)} \quad \forall\, l_y \in L\,. \tag{26}$$

Proof By (8), (9), and (24) we have $p^0(l_y) \ge l_y(y)/p(y) \ge 1/p(y)$ for every $y \in X$. Let p be an IPH function and $x,\ y \in X$ be arbitrary. Suppose that $0 < p(x) < +\infty$ and $0 < p(y) < +\infty$. It follows from Theorem 2 that

$$\frac{l_y(x)}{p(x)} \le \frac{1}{p(y)}\,. \tag{27}$$

If $p(x) = 0$, then, by part (iii) of Theorem 2, we have $l_y(x) = 0$ or $p(y) = 0$, which in both cases (27) holds. In view of (1), (27) holds in the other cases. Therefore, $p^0(l_y) = \sup_{x \in X} l_y(x)/p(x) \le 1/p(y)$. This, together with (25), yields that $p^0(l_y) = 1/p(y)$. To prove the converse, let $x,\ y \in X$ be arbitrary. It follows from (26) that $l_y(x)/p(x) \le 1/p(y)$. Thus, $l_y(x)p(y) \le p(x)$. Since x and y are arbitrary, by Theorem 2 (the implication $(iii) \Longrightarrow (i)$), we conclude that p is IPH. □

The set $supp(p, L) = \{l_y \in L : l_y(x) \le p(x)\ \ \forall\, x \in X\}$ is called the support set of the function $p : X \longrightarrow \bar{\mathbb{R}}_+$ with respect to set L. If p is finite-valued or IPH, then, in view of (9), we get $supp(p, L) \subseteq L'$, and using ψ' we can identify $supp(p, L)$ with some subset of X. Let us denote by $supp(p, X)$ the set $(\psi')^{-1}(supp(p, L))$. Then

$$supp(p, X) = \{y \in X\colon l_y \le p\}\,.$$

We shall call $supp(p, X)$ the $X - support$ of p.

Proposition 7 *Let $p : X \longrightarrow \bar{\mathbb{R}}_+$ be a function. Then, p is IPH if and only if*

$$supp(p, X) = \{y \in X\colon p(y) \ge 1\}\,. \tag{28}$$

Proof Let p be an IPH function. We have

$$\begin{aligned} supp(p, X) &= \{y \in X : l_y(x) \le p(x)\ \ \forall\, x \in X\} \\ &= \{y \in X : p^0(l_y) \le 1\}\,. \end{aligned}$$

Then (26) immediately yields (28). To prove the converse, let $x,\ y \in X$ be arbitrary. If $p(y) = 0$, then it is clear that $p(y)l_y(x) \le p(x)$. Let $0 < p(y) < +\infty$. Then, by hypothesis, we have $r = y/p(y) \in supp(p, X)$. Thus $l_r(x) \le p(x)$, and by (6) we get $l_y(x)p(y) \le p(x)$. Finally, let $p(y) = +\infty$. By (28), $y \in supp(p, X)$. Thus, $l_y(x) \le p(x)$. If $p(x) = 0$, then the nonnegativity of l_y yields that $l_y(x) = 0$, and so $l_y(x)p(y) \le p(x)$. Clearly the latter inequality holds for $p(x) = +\infty$. Let $0 < p(x) < +\infty$. Then $r = x/p(x) \in supp(p, X)$. Therefore, $l_r(y) \le p(y)$ and by (6), $p(x)l_x(y) \le p(y)$. Hence, by Theorem 2, p is IPH, which completes the proof. □

Proposition 8 *For any set $W \subseteq X$, the following assertions are equivalent:*

(i) W is upward, coradiant and closed along rays.

(ii) There exists an IPH function $p : X \longrightarrow \bar{\mathbb{R}}_+$ such that $supp(p, X) = W$. Furthermore, function p of (ii) is unique, namely, p is the Minkowski cogauge ν_W of W.

Proof (i) $\Longrightarrow$ (ii). Let $p = \nu_W$. It is clear that p is positively homogeneous. Moreover, since W is upward, p is increasing. By [9, Proposition 5.6], since W is closed along rays and coradiant, one has

$$W = \{y \in X\colon p(y) \geq 1\}. \tag{29}$$

Whence in view of (28), $W = supp(p, X)$.

(ii) $\Longrightarrow$ (i). Let $W = supp(p, X)$ for an IPH function $p : X \longrightarrow \bar{\mathbb{R}}_+$. By (28) W is coradiant, upward, and closed along rays. Finally, the uniqueness of p in (ii) follows from the following equalities, the last one of which uses the convention $\sup \emptyset = 0$ and is a consequence of (29):

$$\begin{aligned} p(y) &= \sup\{\lambda > 0 : \lambda \leq p(y)\} \\ &= \sup\left\{\lambda > 0 : 1 \leq p\left(\frac{y}{\lambda}\right)\right\} \\ &= \sup\left\{\lambda > 0 : \frac{y}{\lambda} \in W\right\} \\ &= \nu_W(y). \end{aligned}$$

This completes the proof. □

For a function $p : X \longrightarrow \bar{\mathbb{R}}_+$, the L-subdifferential at a point $x_0 \in X$ is defined as follows:

$$\partial_L p(x_0) = \{l_y \in L\colon p(x) - p(x_0) \geq l_y(x) - l_y(x_0)\}.$$

If $\partial_L p(x_0) \subseteq L'$, then the set $\partial_X p(x_0) = (\psi')^{-1}(\partial_L p(x_0))$ will be called X-subdifferential of p at x_0 (note that $\partial_X p(x_0) \subseteq X'$). Thus

$$\partial_X p(x_0) = \{y \in X\colon p(x) - p(x_0) \geq l_y(x) - l_y(x_0)\}. \tag{30}$$

The following simple statement will be useful in the sequel.

Proposition 9 *Let $p : X \longrightarrow \bar{\mathbb{R}}$ be an IPH function and $x \in dom\, p$ be a point such that $p(x) \neq 0$. Then, $r = x/p(x) \notin -K$.*

Proof Let $p(x) > 0$. Then $p(r) = 1 > 0$. Since p is an IPH function, we get $r \notin -K$. If $p(x) < 0$, then $p(-r) = -1$. Then, in view of the monotonicity of p, we get $-r \notin K$ or $r \notin -K$. This completes the proof. □

Theorem 6 *Let $p : X \longrightarrow \bar{\mathbb{R}}_+$ be an IPH function and $x \in dom\, p$ be a point such that $p(x) \neq 0$. Let $r = x/p(x)$. Then $l_r \in \partial_L p(x)$, and hence $\partial_L p(x)$ is nonempty.*

Proof It follows from Proposition 9 and (7) that $r \notin -K$ and $l(y, r) < +\infty$ for all $y \in X$. Clearly $p(x) \in \{\lambda \geq 0 : \lambda r \leq x\}$. Then, by the definition of l, we have $l(x, r) \geq p(x) > 0$. We shall now show that $l_r(y) \leq p(y)$ for any $y \in X$. To this end, let $y \in X$ be arbitrary. If $l(y, r) = 0$, then $l_r(y) \leq p(y)$. Let $0 < l(y, r) < +\infty$. We have $l(y, r)r \leq y$. Since p is IPH, we get $l(y, r)p(r) \leq p(y)$. Because of $p(r) = 1$, we get $l_r(y) \leq p(y)$. Since $y \in X$ was arbitrary, we conclude that $l_r(x) = p(x)$ and $l_r(y) \leq p(y)$ for all $y \in X$. This yields that $l_r \in \partial_L p(x)$. □

Remark 2 Let $\operatorname{int} K \neq \emptyset$. Consider nonzero IPH function $p : X \longrightarrow \bar{\mathbb{R}}_+$ and $x \in X$ such that $p(x) = 0$. We can show $\partial_L p(x) \neq \emptyset$. Indeed, since $p \not\equiv 0$, there exists $r \in \operatorname{int} K$ such that $p(r) > 0$ (see [2], Proposition 6). Set $r' = r/p(r)$. It is clear that $p(r') = 1$, and so, by (28), $r' \in supp(p, X)$, that is, $l_{r'}(t) \leq p(t)$ for all $t \in X$. It follows from the nonnegativity of $l_{r'}$ that $l_{r'}(x) = p(x) = 0$. Hence, $l_{r'} \in \partial_L p(x)$.

We next define the upper polar function $p_0\colon U \longrightarrow \bar{\mathbb{R}}_+$ of the function $p : X \longrightarrow \bar{\mathbb{R}}_+$ by

$$p_0(u_y) = \inf_{x \in X} \frac{u_y(x)}{p(x)}, \qquad u_y \in U \tag{31}$$

(with the conventions $0/0 = +\infty$ and $+\infty/+\infty = +\infty$).

The proof of the following result can be done in a manner analogous to that of Theorem 5.

Theorem 7 *Let $p : X \longrightarrow \bar{\mathbb{R}}_+$ be a function. Then*

$$p_0(u_y) \leq \frac{1}{p(y)}, \qquad \forall\, u_y \in U, \tag{32}$$

and p is IPH if and only if

$$p_0(u_y) = \frac{1}{p(y)}, \qquad \forall\, u_y \in U. \tag{33}$$

We shall now study the structure of support sets from above, which are characterized by the elementary functions u_y rather than by the functions l_y (which characterize support sets from below). We shall denote the support set from above, or upper support set, of the function $p : X \longrightarrow \bar{\mathbb{R}}_+$ with respect to set U as

$$Supp^+(p, U) = \{u_y \in U\colon u_y \geq p\}.$$

In what follows, we state the counterpart of Proposition 7 for the support set from above.

Proposition 10 *Let $p : X \longrightarrow \bar{\mathbb{R}}_+$ be a function. Then*

$$Supp^+(p, U) = \{u_y \in U\colon\ p_0(u_y) \geq 1\}. \tag{34}$$

Furthermore, p is IPH if and only if

$$Supp^+(p, U) = \{u_y \in U\colon p(y) \leq 1\}. \tag{35}$$

Proof Equality (34) follows easily from the definitions of $Supp^+(p, U)$ and p_0. Furthermore, if p is IPH, then (35) follows from (33) and (34). To prove the converse, let $x,\ y \in X$ be arbitrary. If $0 < p(y) < +\infty$, then $u_{y/p(y)} \in Supp^+(p, U)$. Thus, $u_{y/p(y)}(x) \geq p(x)$. By (14) we get $p(y)u(x, y) \geq p(x)$. If $p(y) = +\infty$, we have $p(x) \leq u(x, y)p(y)$ (here we use the convention $(+\infty) \times 0 = +\infty$). Finally, let $p(y) = 0$. It is clear that $u_y \in Supp^+(p, U)$. Thus, $u_y(x) \geq p(x)$. If $u_y(x) = 0$, then, in view of the nonnegativity of p, we get $p(x) = 0$, and so $p(x) \leq u(x, y)p(y)$. Now, suppose that $0 < u_y(x) < +\infty$. It follows from $p(\lambda y) = 0$ for all $\lambda > 0$ that $u_{\lambda y} \in Supp^+(p, U)$ for all $\lambda > 0$, and in view of (14), we get $(1/\lambda)u_y(x) = u_{\lambda y}(x) \geq p(x)$ for all $\lambda > 0$. This means that $p(x) = 0$. Therefore, $p(x) \leq u(x, y)p(y)$. Hence, by Theorem 2 (implication (iv) $\Longrightarrow$ (i)), p is IPH. □

For the function $p : X \longrightarrow \bar{\mathbb{R}}_+$, in a manner analogous to the case of L-subdifferential, we now define the U-superdifferential of p at x_0 as follows:

$$\partial_U^+ p(x_0) := \{u_y \in U\colon\ u_y(x) - u_y(x_0) \geq p(x) - p(x_0)\}.$$

One can prove the following result for U-superdifferential in a manner analogous to the proof of Theorem 6, and therefore we omit its proof.

Theorem 8 *Let $p : X \longrightarrow \bar{\mathbb{R}}_+$ be an IPH function and $x \in dom\, p$ be a point such that $p(x) \neq 0$. Let $r = x/p(x)$. Then $u_r \in \partial_{U'}^+ p(x)$.*

Definition 5 Let $U \subseteq X$. Then the left polar set of W is defined by

$$W^{ol} = \{x \in X\colon\ l(x, y) \leq 1 \quad \forall\, y \in W\}.$$

Analogously, we define the right polar set of $V \subseteq X$.

Definition 6 Let $V \subseteq X$. Then the right polar set of V is defined by

$$V^{or} = \{y \in X\colon\ l(x, y) \leq 1 \quad \forall\, x \in V\}.$$

In the following theorem, we assume that $int\, K \neq \emptyset$.

Theorem 9 *Let $W,\ V \subseteq X$ and $V \cap int\, K \neq \emptyset$. Then the following assertions are true:*

(i) One has $W = W^{olor}$ if and only if W is upward, coradiant and closed along rays.

(ii) One has $V = V^{orol}$ if and only if V is downward, radiant, and closed along rays.

Proof Since $X^{ol} = X^{or} = \emptyset,\ \emptyset^{ol} = \emptyset^{or} = X$, and X is upward, downward, radiant, coradiant and closed in itself, both statements are true when $W = V = X$. For the rest of the proof we shall assume that $W \neq X$ and $V \neq X$.

(i) Let $W \subseteq X$ and $W^{or} \neq \emptyset$. By the definition of W^{or}, Remark 1, and Proposition 3, W^{or} is coradiant, upward, and closed along rays. Therefore, $W = W^{olor}$ implies that W is coradiant, upward, and closed along rays. To prove the converse, we shall first show that $W \subseteq W^{olor}$. Let $y \in W$. Since for any $x \in W^{ol}$ we have $l_y(x) \leq 1$, it is clear that $y \in W^{olor}$. We shall now show that $W^{olor} \subseteq W$. Let $y \in W^{olor}$. By Proposition 8 we have $W = supp(p, X)$ for some IPH function $p\colon X \longrightarrow \bar{\mathbb{R}}_+$. Let $x \in X$ and $\lambda \in (p(x), +\infty)$ be arbitrary. For every $y' \in W = supp(p, X)$, since $l_{y'}(x) \leq p(x) < \lambda$, using (5), one gets $l_{y'}(x/\lambda) = \lambda^{-1} l_{y'}(x) < 1$, whence $x/\lambda \in W^{ol}$. Therefore, $l_y(x) = \lambda\, l_y(x/\lambda) < \lambda$. Hence, $l_y(x) \leq p(x)$. This proves that $l_y \leq p$, that is, $y \in supp(p, X) = W$.

(ii) Suppose that V^{ol} is a nonempty set. Then, by the definition of V^{ol}, Proposition 3, and Remark 1, V^{ol} is downward, radiant, and closed along rays. Therefore, $V = V^{orol}$ implies that V is downward, radiant, and closed along rays. To prove the converse, we shall first show that $V \subseteq V^{orol}$. Let $x \in V$. Since for any $y \in V^{or}$ we have $l_y(x) \leq 1$, it follows that $x \in V^{orol}$. We shall now show that $V^{orol} \subseteq V$. Let $x \in V^{orol}$. Consider the Minkowski gauge $\mu_V\colon X \longrightarrow \bar{\mathbb{R}}_+$. It follows from [9], Proposition 5.1 that

$$V = \{t \in X : \mu_V(t) \leq 1\}. \tag{36}$$

Thus, if $\mu_V(x) = 0$, then $x \in V$. Assume that $\mu_V(x) > 0$. Since $V \cap int\, K \neq \emptyset$, we get $\mu_V(x) < +\infty$. Set $r = x/\mu_V(x)$. By (28), $r \in supp(\mu_V, X)$. Then $l_r(t) \leq \mu_V(t)$ for each $t \in X$. In view of (36), we obtain $l_r(t) \leq 1$ for all $t \in V$, that is, $r \in V^{or}$. Thus, $l_r(x) \leq 1$, and so by (6) and (8), $\mu_V(x) \leq 1$ (note that $\mu_V(x) > 0$ implies that $x \notin -K$). This proves that $x \in V$, which completes the proof. □

Abstract Concavity of DPH Functions

Recall that a function $q : X \longrightarrow \bar{\mathbb{R}}$ is called decreasing if $x \geq y \implies q(x) \leq q(y)$. If p is an IPH function, then the functions $q(x) = p(-x)$ and $q_*(x) = -p(x)$ are DPH (decreasing and positively homogeneous of degree one). Hence, DPH functions can be investigated by using the properties of IPH functions. In this section, we shall study DPH functions separately. To this end, we need to introduce the function $g\colon X \times X \longrightarrow \bar{\mathbb{R}}$ defined by

$$g(x, y) := \min\{\lambda \in \mathbb{R}\colon \lambda y \leq x\} \tag{37}$$

(with the conventions $\min \emptyset := +\infty$ and $\min \mathbb{R} := -\infty$).

The following proposition can be easily proved:

Proposition 11 *For every $x, x', y \in X$ and $\lambda > 0$, one has*

$$g(\lambda x, y) = \lambda g(x, y)\,, \tag{38}$$

$$g(x, \lambda y) = \frac{1}{\lambda} g(x, y)\,, \tag{39}$$

$$x \leq x' \implies g(x, y) \geq g(x', y)\,, \tag{40}$$

$$g(x, y) = -\infty \implies y \in K\,, \tag{41}$$

$$g(x, x) = 1 \iff x \notin K\,. \tag{42}$$

It is worth noting that in (37) we cannot restrict the definition of g to $\lambda \leq 0$ because we shall lose property (42).

For each $y \in X$, we consider the cones C_y, C_y^+ and C_y^- defined by

$$C_y = \{x \in X : g(x, y) \in \mathbb{R}_{-\infty}\}\,, \tag{43}$$

$$C_y^+ = \{x \in C_y : g(x, y) > 0\}\,, \tag{44}$$

$$C_y^- = \{x \in C_y : g(x, y) \in \bar{\mathbb{R}}_-\}\,. \tag{45}$$

It is easy to check that C_y^- is an upward convex cone and C_y^+ is a downward cone. Each element $y \in X$ generates the following functions:

$$f_y^+(x) = \begin{cases} g(x, y), & x \in C_y^+ \\ +\infty, & \text{otherwise}, \end{cases} \tag{46}$$

and

$$f_y^-(x) = \begin{cases} g(x, y), & x \in C_y^- \\ +\infty, & \text{otherwise}. \end{cases} \tag{47}$$

Let F be the set of all functions defined by (46) and (47).

Remark 3 The function f_y^- is DPH for each $y \in X$.

The proof of the following proposition is similar to that of Proposition 9, and therefore we omit its proof.

Proposition 12 *Let $q : X \longrightarrow \bar{\mathbb{R}}$ be a DPH function and $x \in dom\, q$ be a point such that $q(x) \neq 0$. Then $r = x/q(x) \notin K$.*

Proposition 13 *Let $q : X \longrightarrow \bar{\mathbb{R}}$ be a DPH function and $x \in dom\, q$ be a point such that $q(x) \neq 0$. Let $r = x/q(x)$. Then the superdifferential $\partial_F^+ q(x)$ is nonempty and the following assertions are true:*

1. *If $q(x) > 0$, then $f_r^+ \in \partial_F^+ q(x)$.*
2. *If $q(x) < 0$, then $f_r^- \in \partial_F^+ q(x)$.*

Proof We only prove part (i). Since $q(x) \in \{\lambda \in \mathbb{R}\colon \lambda r \leq x\}$, by (37), we get $g(x, r) \leq q(x)$. In view of (39) and (42), we have

$$g(x, r) = q(x) g(x, x) = q(x) > 0$$

(note that since $q(x) \neq 0$, it follows from Proposition 12 that $x \notin K$). By (44) and (46) we have $x \in C_r^+$ and $f_r^+(x) = g(x, r) \leq q(x)$. We shall now show that $f_r^+(y) \geq q(y)$ for every $y \in X$. Let $y \in X$ be arbitrary. If $y \notin C_r^+$, then $f_r^+(y) = +\infty \geq q(y)$. Assume that $y \in C_r^+$. Then $g(y, r)r \leq y$. Since q is DPH, we get $g(y, r)q(r) \geq q(y)$. It follows from $q(r) = 1$ and (46) that $f_r^+(y) \geq q(y)$. This yields that $f_r^+(x) = q(x)$ and $f_r^+(y) \geq q(y)$ for each $y \in X$. Hence $f_r^+ \in \partial_F^+ q(x)$. □

It follows from the preceding proposition that we do not need functions of the form (47) in the study of nonpositive DPH functions. For each $r \in X$, we can consider the function $s_r : X \longrightarrow \bar{\mathbb{R}}_-$ defined by

$$s_r(x) = \begin{cases} g(x, r), & x \in C_r^- \\ 0, & x \notin C_r^-, \end{cases} \tag{48}$$

instead of the function f_r^- defined by (47). Let S be the set of all functions defined by (48). Since C_r^- is an upward set, we get that set S consists of nonpositive DPH functions; hence each S-concave function is DPH. We shall now give an infimal representation of DPH functions.

Proposition 14 *Let $q : X \longrightarrow \mathbb{R}_-$ be a nonzero function. Then q is DPH if and only if it is S-concave.*

Proof We only prove the part if. Let $W' = supp^+(q, S)$. We shall show that $W' \neq \emptyset$. Consider $x \in X$ such that $q(x) < 0$. Set $r = x/q(x)$. It follows from Proposition 12 and (41) that $r \notin K$ and $g(y, r) > -\infty$ for all $y \in X$. Since $q(x) \in \{\lambda \in \mathbb{R} : \lambda r \leq x\}$, by (37) we get $g(x, r) \leq q(x) < 0$. Then $x \in C_r^-$, and by (48) we obtain $s_r(x) \leq q(x)$. We shall now show that $s_r(y) \geq q(y)$ for each $y \in X$. Let $y \in X$ be arbitrary. If $y \notin C_r^-$, then $s_r(y) = 0 \geq q(y)$. Assume that $y \in C_r^-$. Then $(-g(y, r))(-r) = g(y, r)r \leq y$. Since q is DPH, we get $-g(y, r)q(-r) \geq q(y)$. It follows from $q(-r) = -1$ and $y \in C_r^-$ that $s_r(y) \geq q(y)$. Thus $s_r \in W' = supp^+(q, S)$ and $s_r(x) = q(x)$. Finally, if $q(x) = 0$, then $s(x) = 0$ for each $s \in W'$. Hence $q(x) = \min_{s \in W'} s(x)$, that is, q is S-concave. □

In the sequel, we introduce the function $h : X \times X \longrightarrow \mathbb{R}$ defined by

$$h(x, y) := \max\{\lambda \in \mathbb{R} : \lambda y \leq x\} \tag{49}$$

(we use the conventions $max\, \emptyset := -\infty$ and $\max \mathbb{R} := +\infty$). The next proposition gives some properties of the coupling function h. We omit its easy proof.

Proposition 15 *For every x, x', $y \in X$ and $\gamma > 0$, one has*

$$h(\gamma x, y) = \gamma h(x, y)\,, \tag{50}$$

$$h(x, \gamma y) = \frac{1}{\gamma} h(x, y)\,, \tag{51}$$

$$h(x, y) = +\infty \implies y \in -K\,, \tag{52}$$

$$h(x, x) = 1 \iff x \notin -K\,, \tag{53}$$

$$x \leq x' \implies h(x, y) \leq h(x', y)\,, \tag{54}$$

$$x \in K,\ y \in -K \implies h(x, y) = +\infty\,. \tag{55}$$

For each $y \in X$, consider the cones

$$K_y = \{x \in X : h(x, y) \in \mathbb{R}_{+\infty}\} \tag{56}$$

and

$$K_y^- = \{x \in K_y : h(x, y) < 0\}\,. \tag{57}$$

Clearly, K_y^- is a downward cone. Each element $y \in X$ generates the function $g_y^- : X \longrightarrow \bar{\mathbb{R}}_-$ defined by

$$g_y^-(x) = \begin{cases} h(x, y), & x \in K_y^- \\ -\infty, & x \notin K_y^-. \end{cases} \tag{58}$$

Let G_- be the set of all functions defined by (52). We conclude this section by a result on negative IPH functions.

Theorem 10 *Let $p : X \longrightarrow \bar{\mathbb{R}}_-$ be an IPH function and $x \in dom\, p$ be a point such that $p(x) \neq 0$. Let $r = x/p(x)$. Then $g_r^- \in \partial_{G_-} p(x)$, and hence $\partial_{G_-} p(x)$ is nonempty.*

Proof It is clear that $p(x) < 0$. Since $p(x) \in \{\lambda \in \mathbb{R} : \lambda r \leq x\}$, by (49) we get $h(x, r) \geq p(x)$. In view of Proposition 15, we have

$$h(x, r) = p(x)h(x, x) = p(x) < 0$$

(note that since $p(x) \neq 0$, it follows from Proposition 9 that $x \notin -K$). By (51) and (52) we have $x \in K_r^-$ and $g_r^-(x) = h(x, r) \geq p(x)$. We shall now show that $g_r^-(y) \leq p(y)$ for every $y \in X$. Let $y \in X$ be arbitrary. If $y \notin K_r^-$, then $g_r^-(y) = -\infty \leq p(y)$. Assume that $y \in K_r^-$. Then $(-h(y, r))(-r) = h(y, r)r \leq y$. Since p is IPH, we have $-h(y, r)p(-r) \leq p(y)$. It follows from $p(-r) = -1$ and (52) that $g_r^-(y) \leq p(y)$. This yields that $g_r^-(x) = p(x)$ and $g_r^-(y) \leq p(y)$ for each $y \in X$. Hence $g_r^- \in \partial_{G_-} p(x)$. □

References

1. Billera LJ (1974) On games without side payments arising from a general class of markets. J Math Econom 1(2):129–139
2. Dutta J, Martinez-Legaz JE, Rubinov AM (2004) Monotonic analysis over cones: I. Optimization 53:129–146
3. Gunawardena J (1998) An introduction to idempotency. Cambrige University Press, Cambridge
4. Gunawardena J (1999) From max-plus algebra to nonexpansive mappings: a nonlinear theory for discrete event systems. Theor Comp Sci 293(1):141–167

5. Martinez-Legaz JE, Rubinov AM (2001) Increasing positively homogeneous functions on $\mathbb{R}^n$. Acta Math Vietnam 26(3):313–331
6. Martinez-Legaz JE, Rubinov AM, Singer I (2002) Downward sets and their separation and approximation properties. J Glob Optim 23(2):111–137
7. Mohebi H, Sadeghi H (2007) Monotonic analysis over ordered topological vector spaces: I. Optimization 56(3):1–17
8. Rockafellar RT (1970) Convex analysis. In: Princeton Mathematical Series, vol 28. Princeton University Press, Princeton
9. Rubinov AM (2003) Monotonic analysis: convergence of sequences of monotone functions. Optimization 52:673–692
10. Rubinov AM (2000) Abstract convex analysis and global optimization. Kluwer, Boston Dordrecht London
11. Rubinov AM, Singer I (2000) Best approximation by normal and co-normalsets. J Approx Theory 107:212–243
12. Rubinov AM, Singer I (2001) Topical and sub-topical functions, downward sets and abstract convexity. Optimization 50:307–351
13. Sharkey WW (1981) Convex games without sidepayments. Int J Game Theory 10(2):101–106
14. Singer I (1997) Abstract convex analysis. Wiley-Interscience, New York

Inequality-constrained Nonlinear Optimization

WALTER MURRAY
Stanford University, Stanford, USA

MSC2000: 49M37, 65K05, 90C30

Article Outline

Keywords
Synonyms
The Problem
First Order Optimality Conditions
Second Order Optimality Conditions
Algorithms
See also
References

Keywords

Constrained optimization; Optimality conditions; Algorithms

Synonyms

IEQNO

The Problem

An inequality-constrained nonlinear programming problem may be posed in the form

$$\begin{cases} \min\limits_{x\in\mathbf{R}^n} & f(x) \\ \text{s.t.} & c(x) \geq 0, \end{cases} \tag{1}$$

where $f(x)$ is a nonlinear function and $c(x)$ is an m-vector of nonlinear functions with ith component $c_i(x)$, $i = 1, \ldots, m$. We shall assume that f and c are sufficiently smooth. Let x^* denote a solution to (1). We are mainly concerned about smoothness in the neighborhood of x^*. In such a neighborhood we assume that both the gradient of $f(x)$ denoted by $g(x)$ and the $m \times n$ Jacobian of $c(x)$ denoted by $J(x)$ exist and are Lipschitz continuous. As is the case with the unconstrained problem a solution to this problem may not exist. Typically additional assumptions are made to ensure a solution does exist. A common assumption is to assume that the objective $f(x)$ is bounded below on the feasible set. However, even this is not sufficient to assure a minimizer exists but it is obviously a necessary condition for an algorithm to be assured of converging. If the feasible region is compact then a solution does exist. We shall only be concerned with local solutions.

First Order Optimality Conditions

The problem is closely related to the equality-constrained problem. If it was known which constraints were active (exactly satisfied) at a solution and which were slack (strictly positive) then the optimality conditions for (1) could be replaced by the optimality conditions for the equality case. Note that this does not imply the inequality problem could be replaced by an equality problem when it comes to determining a solution by an algorithm. The inequality problem may have solutions corresponding to different sets of constraints being active. Also an equality problem may have solutions that are not solutions of the inequality problem. Nonetheless this equivalence in a local neighborhood enables us to determine the optimality conditions for this problem from those of an equality-constrained problem. In order to study the optimality conditions it is necessary to introduce some notation.

Let $\widehat{c}(x)$ and $\overline{c}(x)$ denote the constraints active and slack at x respectively. Likewise, let $\widehat{J}(x)$ and $\overline{J}$ denote

their respective Jacobians. Assume that $\widehat{J}(x^*)$ is full rank. Points at which the Jacobian of the active constraints is full rank are said to be *regular*. It follows from the necessary conditions for the equality case that

$$\begin{aligned} g(x^*) - \widehat{J}(x^*)^\top \widehat{\lambda} &= 0, \\ \widehat{c}(x^*) &= 0, \\ \overline{c}(x^*) &> 0, \end{aligned}$$

where $\widehat{\lambda}$ is vector of Lagrange multipliers. These equations may be written in the form:

$$\begin{aligned} &g(x^*) - J(x^*)^\top \lambda^* = 0, \\ &c(x^*) \geq 0, \\ &\lambda^{*T} c(x^*) = 0, \end{aligned}$$

where λ^* is the *extended set of Lagrange multipliers*. The set is extended by defining a multiplier to be zero for the slack constraints at x^* $\overline{c}(x^*)$.

The above first order optimality conditions are not the only necessary conditions. Unlike the equality case there may be a feasible arc that moves off one or more of the active constraints along which the objective is reduced. In other words we need some characterization that is necessary for the active set to be binding. The key to identifying the binding set is to examine the sign of $\widehat{\lambda}$.

It follows from the definition of $\widehat{\lambda}$ that

$$\widehat{\lambda} = (\widehat{J}\widehat{J}^\top)^{-1}\widehat{J}g, \tag{2}$$

where the argument x^* has been dropped for simplicity. Note that (2) implies that $\left\|\widehat{\lambda}\right\|$ is bounded.

Define p as

$$\widehat{J}p = \delta e + e_j,$$

where $\delta > 0$, e denotes the vector of ones and e_j is the unit column with one in the jth position. It follows from the assumption on the continuity of the Jacobian that $x^* + \alpha\, p$ is feasible for $0 \leq \alpha \leq \overline{\alpha}$ is sufficiently small. From the mean value theorem we have

$$f(x^* + \alpha p) = f(x^*) + \alpha p^\top g(x^* + \xi\alpha p),$$

where $0 \leq \xi \leq 1$. The Lipschitz continuity of g implies M exists such that

$$p^\top g(x^* + \xi\alpha p) \leq p^\top g(x^*) + \alpha M.$$

It follows that

$$f(x^* + \alpha p) \leq f(x^*) + \alpha(p^\top g(x^*) + \alpha M).$$

From the necessary conditions on x^* we get

$$p^\top g(x^*) = p^\top \widehat{J}^\top \widehat{\lambda},$$

which implies

$$f(x^* + \alpha p) \leq f(x^*) + \alpha(p^\top \widehat{J}^\top \widehat{\lambda} + \alpha M).$$

Using the definition of p gives

$$f(x^* + \alpha p) \leq f(x^*) + \alpha(\delta e^\top \widehat{\lambda} + \widehat{\lambda}_j + \alpha M).$$

It follows from the boundedness of $\widehat{\lambda}$ that if $\widehat{\lambda}_j < 0$ then for δ sufficiently small there exists $\overline{\alpha}$ such that for $0 < \alpha \leq \overline{\alpha}$,

$$f(x^* + \alpha p) < f(x^*).$$

Consequently, a necessary condition for x^* to be a minimizer under the assumptions made is that $\widehat{\lambda} \geq 0$. Equivalently, $\lambda^* \geq 0$.

For different assumptions such as $\widehat{J}$ not being full rank the condition need not hold as the following simple case illustrates. Suppose we have an equality-constrained problem with $c(x) = 0$ then an equivalent inequality-constrained problem is

$$\begin{cases} \min\limits_{x \in \mathbf{R}^n} & f(x) \\ \text{s.t.} & c(x) \geq 0, \\ & -c(x) \geq 0. \end{cases}$$

It follows that all constraints are active at a solution. We know in this case there are no necessary conditions on the Lagrange multipliers. Clearly the Jacobian of the active constraints is not full rank. Geometrically what breaks down is that there is no perturbation from x^* that moves feasible with respect to one constraint without violating at least one other constraint.

The condition $c(x^*)^\top\ \lambda^* = 0$ is a *complementarity condition*. At least one of $(c_i(x^*), \lambda_i^*)$ must be zero. It is possible for both to be zero. If there is no index for which both are zero then $c(x^*)$ and λ^* are is said to satisfy *strict complementarity*.

If $\widehat{J}(x^*)$ is full rank then it follows from (2) that λ^* is an isolated point.

The function $L(x, \lambda)$,

$$L(x, \lambda) = F(x) - \lambda^\top c(x),$$

is known as the *Lagrangian*. The optimality condition

$$g(x^*) - J(x^*)^\top \lambda^* = 0$$

is equivalent to $\nabla_x L(x^*, \lambda^*) = 0$. It is also equivalent to $Z(x^*)^\top g(x^*) = 0$, where the columns of $Z(x)$ are a basis for the null space of the rows of $\widehat{J}(x)$. The vector $Z(x)^\top g(x)$ is called the *reduced gradient*.

Clearly Lagrange multipliers play a significant role in defining the solution of an inequality-constrained problem. There is a significant difference in that role between linear and nonlinear constraints. In the case of linear constraints the numerical value of the multiplier plays no role in defining x^* only the sign of the multiplier is significant. For nonlinear constraints the numerical value as well as the sign is of significance. To appreciate why it first necessary to appreciate that for problems that are nonlinear in either the constraints or the objective, curvature of the functions are relevant in defining x^*. More precisely the curvature of the Lagrangian. It easily seen that curvature of the objective is relevant since for unconstrained problems no solution would exist otherwise. To appreciate that curvature in $c(x)$ is relevant note that any problem can be transformed into a problem with just a linear objective by adding an extra variable. For example, add the constraint $x_{n+1} - f(x) \geq 0$ and minimize x_{n+1} instead of $f(x)$. Since we have established the curvature of $f(x)$ is relevant that relevance must still be there even though $f(x)$ now appears only within a constraint. It is harder to appreciate that it is the relative curvature of the various constraints and objective that is of significance.

Second Order Optimality Conditions

We shall now assume that the problem functions are twice continuous differentiable. From the unconstrained case it is known that a necessary condition is that $\nabla^2 f(x^*)$ is positive semidefinite. Obviously a generalization of this condition needs to hold for (1). Again the Lagrangian will be shown to play a key role. We start by examining the behavior of $f(x)$ along a feasible arc emanating from x^*. Although the first order optimality conditions make the first order change in the objective along a feasible arc nonnegative, it could be zero. Consequently, the second order change needs to be nonnegative for arcs where this is true.

We restrict our interest to feasible arcs that remain on the set of constraints active at x^*. If $x(\alpha)$ represents a twice differentiable arc, with $x(0) = x^*$, that lies on the active set then $\widehat{c}(x(\alpha)) = 0$. Define $p \equiv d(x(0))/d\alpha$ and $h \equiv d^2(x(0))/d\alpha^2$. We have

$$\frac{d}{d\alpha}\widehat{c}_i\,(x(\alpha)) = \nabla(\widehat{c}_i(x(\alpha))^\top \frac{d}{d\alpha}x(\alpha),$$
$$\frac{d^2}{d\alpha^2}\widehat{c}_i\,(x(\alpha)) = \frac{d}{d\alpha}x(\alpha)^\top \nabla^2\widehat{c}_i(x(\alpha))\frac{d}{d\alpha}x(\alpha) + \nabla\widehat{c}_i(x(\alpha))^\top \frac{d^2}{d\alpha^2}x(\alpha).$$

Since $\widehat{c}(x(\alpha)) = 0$ it follows that

$$\frac{d^2}{d\alpha^2}\widehat{c}_i\,(x(0)) = \nabla\widehat{c}_i(x^*)^\top h + p^\top \nabla^2\widehat{c}_i(x^*)p = 0. \quad (3)$$

Similarly we get

$$\frac{d^2}{d\alpha^2}f(x(0)) = g(x^*)^\top h + p^\top \nabla^2 f(x^*)p.$$

Since

$$\frac{d}{d\alpha}f(x(0)) = g(x^*)^\top p = 0$$

(otherwise there would be a descent direction from x^*) we require that

$$g(x^*)^\top h + p^\top \nabla^2 f(x^*)p \geq 0.$$

Substituting for $g(x^*)$ using the first order optimality conditions gives

$$h^\top J(x^*)^\top \lambda^* + p^\top \nabla^2 f(x^*)p \geq 0.$$

It follows from (3) and the definition of the extended multipliers that we require

$$-\sum_{i=1}^{m} \lambda_i^* p^\top \nabla^2 c_i(x^*)p + p^\top \nabla^2 f(x^*)p \geq 0.$$

From the definition of $L(x^*, \lambda^*)$ and $\widehat{J}(x^*)p = 0$ this condition is equivalent to requiring that $Z(x^*)^\top \nabla^2 L(x^*, \lambda^*)\, Z(x^*)$ be positive semidefinite. This matrix is called the *reduced Hessian of the Lagrangian*. Since the

condition is on the second derivatives it is termed a *second order optimality condition*. It can now be appreciated that the numerical value of the Lagrange multipliers play a role in defining the solution of a nonlinearly-constrained problem. Note that when there are n active constraints then there is no feasible arc that remains on the active set and the second order optimality condition is empty. When $\widehat{J}$ has n rows then the reduced Hessian has zero dimension. For convenience we can define symmetric matrices of zero dimension to be positive definite.

Necessary and sufficient conditions for x^* to be a minimizer are complex. However, sufficient conditions are easy to appreciate. We have established no feasible descent direction exists that moves off any of the active constraints. Consequently, if $\widehat{\lambda} > 0$ then $f(x)$ increases along any feasible arc emanating from x^* that moves off a constraint. We now only need to be sure the same is true for all arcs emanating from x^* that remaining on the active set. This is assured if

$$\frac{d^2}{d\alpha^2} f(x(0)) = g(x^*)^\top h + p^\top \nabla^2 f(x^*) p > 0,$$

which implies $Z(x^*)^\top \nabla^2 L(x^*, \lambda^*)\, Z(x^*)$ is positive definite. Assuming that x^* is a regular point, strict complementarity hold, the first order necessary conditions hold, and the reduced Hessian at x^* is positive definite then x^* is a minimizer and an isolated point.

Algorithms

Algorithms for inequality problems have a combinatorial element not present in algorithms for equality-constrained problems. The simplest case of linear programming (LP) illustrates the point. Under mild assumptions the solution of an LP is given by the solution of a set of linear equations, i. e. a vertex of the feasible region. The difficult issue is determining which of the constraints define those equations. If there are m inequality constraints and n variables there are $m!/n!(n-m)!$ choices of active constraints. Even for modest values of m and n the possible choices are astronomical. This clearly rules out methods based on exhaustive search.

One class of methods to solve inequality problems are so-called *active set methods*, an example being the simplex method for LP. First a guess is made of the active set (called the working set) and then an estimate to the solution of the resulting equality-constrained problem is computed (in the case of LP or quadratic programming (QP) this would be precise) and at the new point a new guess is made of the active set. The estimate of the solution of the equality-constrained problem is usually made by finding a point that satisfies an approximation to the first order necessary conditions. Unless an intelligent guess is made of the active set such algorithms are doomed to fail. Typically after the initial active set such algorithms generate subsequent working sets automatically. For linearly-constrained problems this is usually a very simple procedure. Assuming the current iterate is feasible an attempt is made to move to the new estimate of the solution. If this is infeasible the best (or a point better than the current iterate) is found along the direction to the new estimate. The constraints active at the new feasible point are then used to define the working set. Usually the active set will be the working set but occasionally we need to move off a constraint that is currently active. How to identify such a constraint is usually straightforward and can be done by examining an estimate to the Lagrange multipliers (obtained from the solution to the approximation of the first order necessary conditions). More complex strategies are possible that move off several constraints simultaneously.

An initial feasible point is found by solving an LP. One consequence of this strategy is that it is only necessary to consider working sets for which the objective function has a lower value than at the current iterate. Once we are in a neighborhood of the solution the working set does not change if strict complementarity holds at the solution and x^* is a regular point. Typically the change in the working set at each iteration of active set methods for linearly-constrained problems is small (usually one), which results in efficiencies when computing the estimate to the new equality-constrained problem. In practice active set methods work well and usually identify the active set at the solution with very little difficulty. For an LP the number of iterations required to identify the active set usually grows linearly with the size of the problem. However, pathological cases exist in which the number of iterations is astronomical and real LP problems do arise where the number of iterates required is much greater than the typical case. Nonetheless algorithms for linearly-constrained

problems based on active set methods are highly successful.

For nonlinear problem the issue of identifying the active set at the solution is usually less significant since even when the active set is known the number of iterations required to solve a problem may be large. A more relevant issue is that not knowing the active set causes some problems such as making the linear algebra routines much more complicated. For small problems this is of little consequence but in the large scale case it complicates the data structures required.

Nonlinearly-constrained problems are usually an order of magnitude more complicated to solve than linearly-constrained problems. One reason is that algorithms for problems with nonlinear constraints usually do not maintain feasible iterates. If a problem has just one nonlinear equality constraint then generating each member of a sequence that lies on that constraint is itself an infinite process. Methods that generate infeasible iterates need to have some means of assessing whether a point is better than another point. For feasible-point algorithms this is a simple issue since the objective provides a measure of merit. A typical approach is to define a merit function, which balances a change in the objective against the change in the degree of infeasibility. A commonly used *merit function* is

$$M(x,\rho) = f(x) + \rho \sum_{i=1}^{m} \max\{0, -c_i(x)\},$$

where ρ is a parameter that needs to be sufficiently large. Usually it will not be known what 'sufficiently' large is so this parameter is adjusted as the sequence of iterates is generated. Note that $M(x, \rho)$ is not a smooth function and has a discontinuity in its derivative when any element of $c(x)$ is zero. In particular it is not continuous at x^* when a constraint is active at x^*. Were this not the case then constrained problems could be transformed to unconstrained problems and solved as such. While transforming a constrained problem into a simple single smooth unconstrained problem is not possible the transformation approach is the basis of a variety of methods. A popular alternative to direct methods is to transform the problem into that of solving a sequence of smooth linearly-constrained problems. This is the method at the heart of *MINOS* (see [8,9]) one of the most widely used methods for solving problems with nonlinear constraints. Other transformation methods transform the problem to that of solving a sequence of unconstrained or bounds-constrained problem. Transformation methods have an advantage of over direct methods when developing software. For example, if you have a method for solving large scale linearly-constrained problems then it can be used as a kernel in an algorithm to solve large scale nonlinearly-constrained problems.

See also

- Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- First Order Constraint Qualifications
- History of Optimization
- Kuhn–Tucker Optimality Conditions
- Lagrangian Duality: Basics
- Redundancy in Nonlinear Programs
- Relaxation in Projection Methods
- Rosen's Method, Global Convergence, and Powell's Conjecture
- Saddle Point Theory and Optimality Conditions
- Second Order Constraint Qualifications
- Second Order Optimality Conditions for Nonlinear Optimization
- SSC Minimization Algorithms
- SSC Minimization Algorithms for Nonsmooth and Stochastic Optimization

References

1. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: Theory and algorithms. second Wiley, New York
2. Bertsekas DP (1999) Nonlinear programming, 2nd edn. Athena Sci., Belmont, MA
3. Fletcher R (1988) Practical methods of optimization. Wiley, New York
4. Gill PE, Murray W, Wright MH (1981) Practical optimization. Acad. Press, New York
5. Karush W (1939) Minima of functions of several variables with inequalities as side constraints. Dept Math Univ Chicago
6. Kuhn HW (1991) Nonlinear programming: A historical note. In: Lenstra JK, Rinnooy Kan AHG, Schrijver A (eds) Elsevier, AmsterdamHistory of Mathematical Programming: A Collection of Personal Reminiscences. , pp 82–96
7. Kuhn HW, Tucker AW (1951) Nonlinear programming. In: Neyman J (ed) Proc. 2nd Berkeley Symposium on Mathe-

matical Statistics and Probability. Univ Calif Press, Berkeley, CA, pp 481–492
8. Murtagh BA, Saunders MA (1982) A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints. Math Program Stud 16:84–117
9. Murtagh BA, Saunders MA (1993) MINOS 5.4 user's guide. SOL Report Dept Oper Res Stanford Univ 83-20R,
10. Nash SG, Sofer A (1996) Linear and nonlinear programming. McGraw-Hill, New York

Inference of Monotone Boolean Functions

VETLE I. TORVIK, EVANGELOS TRIANTAPHYLLOU
Department Industrial and Manufacturing Systems Engineering, Louisiana State University,
Baton Rouge, USA

MSC2000: 90C09

Article Outline

Keywords
Inference of Monotone Boolean Functions
The Shannon Function and the Hansel Theorem
Hansel Chains
Devising a Smart Question-Asking Strategy
Conclusions
See also
References

Keywords

Boolean function; Monotone Boolean function; Isotone Boolean function; Antitone Boolean function; Classification problem; Boolean function inference problem; Free distributive lattice; Conjunctive normal form; CNF; Disjunctive normal form; DNF; Interactive learning of Boolean functions; Shannon function; Hansel theorem; Hansel chain; Sequential Hansel chains question-asking strategy; Binary search-Hansel chains question-asking strategy; Binary search

The goal in a classification problem is to uncover a system that places examples into two or more mutually exclusive groups. Identifying a classification system is beneficial in several ways. First of all, examples can be organized in a meaningful way, which will make the exploration and retrieval of examples belonging to specific group(s) more efficient. The tree-like directory structure, used by personal computers in organizing files, is an example of a classification system which enables users to locate files quickly by traversing the directory paths. A classification system can make the relations between the examples easy to understand and interpret. A poor classification strategy, on the other hand, may propose arbitrary, confusing or meaningless relations. An extracted classification system can be used to classify new examples. For an incomplete or stochastic system, its structure may pose questions whose answers may generalize the system or make it more accurate.

A special type of *classification problem*, called the *Boolean function inference problem*, is when all the examples are represented by binary (0 or 1) attributes and each example belongs to one of two categories. Many other types of classification problems may be converted into a Boolean function inference problem. For example, a multicategory classification problem may be converted into several two-category problems. In a similar fashion, example attributes can be converted into a set of binary variables.

In solving the Boolean function inference problem many properties of Boolean logic are directly applicable. A *Boolean function* will assign a binary value to each Boolean vector (example). See [22] for an overview of Boolean functions. Usually, a Boolean function is expressed as a conjunction of disjunctions, called the *conjunctive normal form* (CNF), or a disjunction of conjunctions, called the *disjunctive normal form* (DNF). CNF can be written as:

$$\bigwedge_{j=1}^{k} \left(\bigvee_{i \in \rho_j} x_i \right),$$

where x_i is either the attribute or its negation, k is the number of attribute disjunctions and ρ_j is the jth index set for the jth attribute disjunction. Similarly, DNF can be written as:

$$\bigvee_{j=1}^{k} \left(\bigwedge_{i \in \rho_j} x_i \right).$$

It is well known that any Boolean function can be written in CNF or DNF form. See [20] for an algorithm con-

verting any Boolean expression into CNF. Two functions in different forms are regarded as equivalent as long as they assign the same function values to all the Boolean vectors. However, placing every example into the correct category is only one part of the task. The other part is to make the classification criteria meaningful and understandable. That is, an inferred Boolean function should be as simple as possible. One part of the Boolean function inference problem that has received substantial research efforts is that of simplifying the representation of Boolean functions, while maintaining a general representation power.

Inference of Monotone Boolean Functions

When the target function can be any Boolean function with n attributes, all of the 2^n examples have to be examined to reconstruct the entire function. When we have a priori knowledge about the subclass of Boolean functions the target function belongs to, on the other hand, it may be possible to reconstruct it using a subset of the examples. Often one can obtain the function values on examples one by one. That is, at each inference step, an example is posed as a question to an oracle, which, in return, provides the correct function value. A function, f, can be defined by its oracle A_f which, when fed with a vector $x = \langle x_1, \ldots, x_n \rangle$, returns its value $f(x)$. The inference of a Boolean function from questions and answers is known as *interactive learning of Boolean functions*. In many cases, especially when it is either difficult or costly to query the oracle, it is desirable to pose as few questions as possible. Therefore, the choice of examples should be based on the previously classified examples.

The *monotone Boolean functions* form a subset of the Boolean functions that have been extensively studied not only because of their wide range of applications (see [2,7,8] and [24]) but also their intuitive interpretation. Each attribute's contribution to a monotone function is either nonnegative or nonpositive (not both). Furthermore, if all of the attributes have nonnegative (or nonpositive) effects on the function value then the underlying monotone Boolean function is referred to as *isotone* (respectively *antitone*). Any isotone function can be expressed in DNF without using negated attributes. In combinatorial mathematics, the set of isotone Boolean functions is often represented by the *free distributive lattice* (FDL). To formally define monotone Boolean function, consider ordering the binary vectors as follows [21]:

Definition 1 Let E^n denote the set of all binary vectors of length n; let x and y be two such vectors. Then, the vector $x = \langle x_1, \ldots, x_n \rangle$ *precedes* vector $y = \langle y_1, \ldots, y_n \rangle$ (denoted as $x \preceq y$) if and only if $x_i \leq y_i$ for $1 \leq i \leq n$. If, at the same time $x \neq y$, then x *strictly precedes* y (denoted as $x \prec y$).

According to this definition, the order of vectors in E^2 can be listed as follows:

$$\langle 11 \rangle \prec \langle 01 \rangle \prec \langle 00 \rangle$$

and

$$\langle 11 \rangle \prec \langle 10 \rangle \prec \langle 00 \rangle .$$

Note that the vectors $\langle 01 \rangle$ and $\langle 10 \rangle$ are in a sense incomparable.

Based on the order of the Boolean vectors, a nondecreasing monotone (isotone) Boolean function can be defined as follows [21]:

Definition 2 A Boolean function f is said to be an *nondecreasing monotone Boolean function* if and only if for any vectors $x, y \in E^n$, such that $x \prec y$, then $f(x) \prec f(y)$.

A *nonincreasing monotone* (antitone) Boolean function can be defined in a similar fashion. As the method used to infer an antitone Boolean function is the same as that of a isotone Boolean function, we will restrict our attention to the isotone Boolean functions.

When analyzing a subclass of Boolean functions, it is always informative to determine its size. This may give some indications of how general the functions are and how hard it is to infer them. The number of isotone Boolean functions, $\Psi(n)$, defined on E^n is sometimes referred to as the nth *Dedekind number* after R. Dedekind, [6] who computed it for $n = 4$. Since then it has been computed for up to E^8.

- $\Psi(1) = 3$;
- $\Psi(2) = 6$;
- $\Psi(3) = 20$;
- $\Psi(4) = 168$ [6];
- $\Psi(5) = 7,581$ [4];
- $\Psi(6) = 7,828,354$ [28];
- $\Psi(7) = 2,414,682,040,998$ [5];
- $\Psi(8) = 56,130,437,228,687,557,907,788$ [29].

Wiedeman's algorithm [29] employed a Cray-2 processor for 200 hours to compute the value for $n = 8$. This gives a flavor of the complexity of computing the exact number of isotone Boolean functions. The computational infeasibility for larger values of n provides the motivation for approximations and bounds. The best known bound on $\Psi(n)$ is due to D. Kleitman, [12] and Kleitman and G. Markowsky, [13]:

$$\Psi(n) \le 2^{\binom{n}{\lfloor n/2 \rfloor}\left(1+c\frac{\log n}{n}\right)},$$

where c is a constant and $\lfloor n/2 \rfloor$ is the integer part of n/2.

This bound, which is an improvement over the first bound obtained by G. Hansel, [11], are also based on the Hansel chains described below. Even though these bounds can lead to good approximations for $\Psi(n)$, when n is large, the best known asymptotic is due to A.D. Korshunov, [15]:

$$\Psi(n) \sim \begin{cases} 2^{\binom{n}{n/2}} e^{f(n)} & \text{for even } n, \\ 2^{\binom{n}{n/2-1/2}+1} e^{g(n)} & \text{for odd } n, \end{cases}$$

where

$$f(n) = \binom{n}{n/2-1}\left(\frac{1}{2^{n/2}} + \frac{n^2}{2^{n+5}} - \frac{n}{2^{n+4}}\right),$$

$$g(n) = \binom{n}{n/2-3/2}\left(\frac{1}{2^{(n+3)/2}} - \frac{n^2}{2^{n+6}} - \frac{n}{2^n+3}\right) + \binom{n}{n/2-1/2}\left(\frac{1}{2^{(n+1)/2}} + \frac{n^2}{2^{n+4}}\right).$$

I. Shmulevich [24] achieved a similar but slightly inferior asymptotic for even n in a simpler and more elegant manner, which led to some interesting distributional conjectures regarding isotone Boolean functions.

Even though the number of isotone Boolean functions is large, it is a small fraction of the number of general Boolean functions, 2^{2^n}. This is the first hint towards the feasibility of efficiently inferring monotone Boolean functions. Intuitively, one would conjecture that the generality of this class was sacrificed. That is true, however, a general Boolean function consists of a set of areas where it is monotone. In fact, any Boolean function $q(x_1, \ldots, x_n)$ can be represented by several nondecreasing g_i (x) and nonincreasing $h_j(x)$ monotone Boolean functions in the following manner [17]:

$$q(x) = \bigvee_i \left(g_i(x) \bigwedge_j h_j(x) \right).$$

As a result, one may be able to solve the general Boolean function inference problem by considering several *monotone Boolean function inference* problems. Intuitively, the closer the target function is to a monotone Boolean function, the fewer monotone Boolean functions are needed to represent it and more successful this approach might be. In [17] the problem of joint restoration of two nested monotone Boolean functions f_1 and f_2 is stated. The approach in [17] allows one to further decrease the dialogue with an expert (*oracle*) and restore a complex function of the form $f_1 \,\&\, \neg f_2$, which is not necessarily monotone.

The Shannon Function and the Hansel Theorem

The complexity of inferring isotone Boolean functions was mentioned in the previous section, when realizing that the number of isotone Boolean functions is a small fraction of the total number of general Boolean functions. In defining the most common complexity measure for the Boolean function inference problem, consider the following notation. Let M_n denote the set of all monotone Boolean functions, and $A = \{F\}$ be the set of all algorithms which infer $f \in M_n$, and $\varphi(F, f)$ be the number of questions to the oracle A_f required to infer f. The *Shannon function* $\varphi(n)$ can be introduced as follows [14]:

$$\varphi(n) = \min_{F \in A}\left(\max_{f \in M_n} \varphi(F, f)\right).$$

An upper bound on the number of questions needed to restore a monotone Boolean function is given by the following equation (known as the *Hansel theorem*) [11]:

$$\varphi(n) = \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}.$$

That is, if a proper *question-asking strategy* is applied, the total number of questions needed to infer any monotone Boolean function should not exceed $\varphi(n)$. The Hansel theorem can be viewed as the worst-case scenario analysis. Recall, from the previous section, that

all of the 2^n questions are necessary to restore a general Boolean function. D.N. Gainanov [9] proposed three other criteria for evaluating the efficiency of algorithms used to infer isotone Boolean functions. One of them is the average case scenario and the two others consider two different ways of normalizing the Shannon function by the size of the target function.

Hansel Chains

The vectors in E^n can be placed in chains (sequences of vectors) according to *monotonicity*. The Hansel chains is a particular set of chains that can be formed using a dimensionally recursive algorithm [11]. It starts with the single Hansel chain in E^1:

$$H^1 = \{\langle 0\rangle, \langle 1\rangle\}.$$

To form the Hansel chains in E^2, three steps are required, as follows:

1. Attach the element '0' to the font of each vector in H^1 and get chain $C^{2\ \min} = \{\langle 00\rangle, \langle 01\rangle\}$.
2. Attach the element '1' to the front of each vector in H^1 and get chain $C^{2\ \max} = \{\langle 10\rangle, \langle 11\rangle\}$.
3. Move the last vector in $C^{2\ \max}$, i.e. vector $\langle 11\rangle$, to the end of $C^{2\ \min}$: $H^{2,1} = \{\langle 00\rangle, \langle 01\rangle, \langle 11\rangle\}$; $H^{2,2} = \{\langle 10\rangle\}$.

To form the Hansel chains in E^3, these steps are repeated:

1. $C^{3,1\ \min} = \{\langle 000\rangle, \langle 001\rangle, \langle 011\rangle\}$; $C^{3,2\ \min} = \{\langle 010\rangle\}$.
2. $C^{3,1\ \max} = \{\langle 100\rangle, \langle 101\rangle, \langle 111\rangle\}$; $C^{3,2\ \max} = \{\langle 110\rangle\}$.
3. $H^{3,1} = \{\langle 000\rangle, \langle 001\rangle, \langle 011\rangle, \langle 111\rangle\}$; $H^{3,2} = \{\langle 100\rangle, \langle 101\rangle\}$; $H^{3,3} = \{\langle 010\rangle, \langle 110\rangle\}$.

Note that since there is only one vector in the $C^{3,2\ \max}$ chain, it can be deleted after the vector $\langle 110\rangle$ is moved to $C^{3,2\ \min}$. This leaves the three chains listed in Table 1. In general, the Hansel chains in E^n can be generated recursively from the Hansel chains in E^{n-1} by following the three steps described above.

A nice property of the Hansel chains is that all the vectors in a particular chain are arranged in increasing order. That is, if the vectors V_j and V_k are in the same chain then $V_j < V_k$ (i. e., V_j strictly precedes V_k when $j < k$). Therefore, if the underlying Boolean function is isotone, then one can classify vectors within a chain easily. For example, if a vector V_j is negative (i. e., $f(V_j) = 0$), then all the vectors preceding V_j in the same chain are also negative (i. e., $f(V_k) = 0$ for any $k < j$). Similarly, if a vector V_j is positive, then all the vectors succeeding V_j in the same chain are also positive. The monotone ordering of the vectors in Hansel chains motivates the composition of an efficient question-asking strategy discussed in the next section.

Inference of Monotone Boolean Functions, Table 1
Hansel chains for E^3

chain #	vector in-chain index	vector
1	1	000
	2	001
	3	011
	4	111
2	1	100
	2	101
3	1	010
	2	110

Devising a Smart Question-Asking Strategy

The most straightforward question-asking strategy, which uses Hansel chains, sequentially moves from chain to chain. Within each chain one may also sequentially select vectors to pose as questions. After an answer is given, the vectors (in other chains also) that are classified as a result of monotonicity are eliminated from further questioning. Once all the vectors have been eliminated, the underlying function is revealed. The maximum number of questions for this method, called the *sequential Hansel chains question-asking strategy*, will not exceed the upper limit $\varphi(n)$, given in the Hansel theorem, as long as the chains are searched in increasing size.

Although the sequential question-asking strategy is easy to implement and effective in reducing the total number of questions, there is still room for improvements. N.A. Sokolov [25] introduced an algorithm that sequentially moves between the Hansel chains in de-

Inference of Monotone Boolean Functions, Table 2
Iteration 1

chain #	index of vectors in the chain	vector	vector classified	middle vector in the chain	reward P if the vector is positive	reward N if the vector is negative	selected middle vector with the largest $\min(P, N)$	answer	other vectors determined
1	1	000							
	2	001		←	4	2	←	1	
	3	011							1
	4	111							1
2	1	100		←	4	2			
	2	101							1
3	1	010		←	4	2			
	2	110							

creasing size and performs a middle vector search of each chain. His algorithm does not require storing all the Hansel chains since at each iteration it only requires a single chain. This advantage is obtained at the cost of asking more questions than needed.

In an entirely different approach, Gainanov [9] presented a heuristic that has been used in numerous algorithms for inferring a monotone Boolean function, such as in [3] and in [18]. This heuristic takes as input an unclassified vector and finds a border vector (maximal false or minimal true) by sequentially questioning neighboring vectors. The problem with most of the inference algorithms based on this heuristic is that they do not keep track of the vectors classified, only the resulting border vectors. Note that for an execution of this heuristic, all of the vectors questioned are not necessarily covered by the resulting border vector, implying that valuable information may be lost. In fact, several border vectors may be unveiled during a single execution of this heuristic, but only one is stored. Many of these methods are designed to solve large problems where it might be inefficient or even infeasible to store all of the information gained within the execution of the heuristic. However, these methods are not efficient (not even for small size problems), in terms of the number of queries they require.

One may look at each vector as carrying a 'reward' value in terms of the number of other vectors that will be classified concurrently. This reward value is a random variable that takes on one of two (one if the two values are the same) values depending on whether the vector is a positive or a negative example of the target function. The expected reward is somewhere between these two possible values. If one wishes to maximize the expected number of classified vectors at each step, the probabilities associated with each of these two values need to be computed in addition to the actual values. Finding the exact probabilities is hard, while finding the reward values is relatively simple for a small set of examples.

This is one of the underlying ideas for the new inference algorithm termed the *binary search-Hansel chains question-asking strategy*. This method draws its motivation, for calculating and comparing the 'reward' values for the middle vectors in each Hansel chain, from the widely used *binary search algorithm* (see, for instance, [19]). Within a given chain, a binary search will dramatically reduce the number of questions (to the order of $\log_2$ while the sequential search is linear). Once the 'reward' values of all the middle vectors have been found, the most promising one will be posed as a question to the oracle. Because each vector has two values, selecting the most promising vector is subjective and several different evaluative criteria can be used.

The binary search-Hansel chains question-asking strategy can be divided into the following steps:

1) Select the middle vector of the unclassified vectors in each Hansel chain.
2) Calculate the reward values for each middle vector. That is, calculate the number of vectors that can be classified as positive (denoted as P) if it is positive and negative (denoted as N) if it is negative.

Inference of Monotone Boolean Functions, Table 3
Iteration 2. The vector ⟨100⟩ is chosen and based on the answer, the class membership of the vectors ⟨100⟩ and ⟨000⟩ is determined

chain #	index of vectors in the chain	vector	vector classified	middle vector in the chain	reward P if the vector is positive	reward N if the vector is negative	selected middle vector with the largest min(P, N)	answer	other vectors determined
1	1	000		←	4	1			0
	2	001	1						
	3	011	1						
	4	111	1						
2	1	100		←	2	2	←	0	
	2	101	1						
3	1	010		←	2	2			
	2	110							

3) Select the most promising middle vector, based on the (P, N) pairs of the middle vectors, and ask the oracle for its membership value.
4) Based on the answer in Step 3, eliminate all the vectors that can be classified as a result of the previous answer and the property of monotonicity.
5) Redefine the middle vectors in each chain as necessary.
6) Unless all the vectors have been classified, go back to Step 2.

The inference of a monotone Boolean function on E^3 by using the binary search-Hansel chains question-asking strategy is illustrated below. The specifics of Iteration 1, described below, are also shown in Table 2. At the beginning of first iteration, the middle vectors in each Hansel chain (as described in Step 1) are selected and marked with the '←' symbol in Table 2. Then, according to Step 2, the reward value for each one of these middle vectors is calculated. For instance, if ⟨001⟩ (the second vector in chain 1) has a function value of 1, then the three vectors ⟨000⟩, ⟨001⟩ and ⟨010⟩ are also classified as positive. That is, the value of P for vector ⟨001⟩ equals 4. Similarly, ⟨000⟩ will be classified as 0 if ⟨001⟩ is classified as 0 and thus its reward value N equals 2.

Once the 'reward' values of all the middle vectors have been evaluated, the most promising middle vector will be selected based on their (P, N) pairs. Here we choose the vector whose min (P, N) value is the largest among the middle vectors. If there is a tie, it will be broken randomly. Based on this evaluative criterion, vector 2 is chosen in chain 1 and is marked with '←' in the column 'selected middle vector with the largest min (P, N)'. After receiving the function value of 1 for vector ⟨001⟩, its value is placed in the 'answer' column. This answer is used to eliminate all of the vectors succeeding ⟨001⟩. The middle vector in the remaining chains are updated as needed. At least one more iteration is required, as there still are unclassified vectors.

After the second iteration, no unclassified vectors are left in chains 1 and 2, and the middle of these chains need not be considered anymore. Therefore, an 'X' is placed in the column called 'middle vector in the chain' in Table 4. At the beginning of the third iteration, the vector ⟨010⟩ is chosen and the function value of the remaining two vectors ⟨010⟩ and ⟨110⟩ are determined. At this point all the vectors have been classified and the question-asking process stops.

The algorithm posed a total of three questions in order to classify all the examples. The final classifications listed in Table 5. corresponds to the monotone Boolean function $x_2 \vee x_3$.

Conclusions

This paper described some approaches and some of the latest developments in the problem of inferring monotone Boolean functions. As it has been described here, by using Hansel chains in the sequential question-asking strategy, the number of questions will not exceed the upper bound stated in the Hansel theorem. How-

Inference of Monotone Boolean Functions, Table 4
Iteration 3

chain #	index of vectors in the chain	vector	vector classified	middle vector in the chain	reward P if the vector is positive	reward N if the vector is negative	selected middle vector with the largest $\min(P, N)$	answer	other vectors determined
1	1	000	0						
	2	001	1	X					
	3	011	1						
	4	111	1						
2	1	100	0	X					
	2	101	1						
3	1	010		←	2	1	←	1	
	2	110							1

Inference of Monotone Boolean Functions, Table 5
The resulting class memberships

chain #	vector in-chain index	vector	function value
1	1	100	0
	2	101	1
2	1	010	1
	2	110	1
3	1	000	0
	2	001	1
	3	011	1
	4	111	1

ever, by combining the binary search of Hansel chains with the notion of an evaluative criterion, the number of questions asked can be further reduced. At present, the binary search-Hansel chains question-asking strategy is only applied to Hansel chains with a dimension of less than 10. However, it is expected that this method can be applied to infer monotone Boolean functions of larger dimensions with slight modifications.

See also

- ► Alternative Set Theory
- ► Boolean and Fuzzy Relations
- ► Checklist Paradigm Semantics for Fuzzy Logics
- ► Finite Complete Systems of Many-valued Logic Algebras
- ► Optimization in Boolean Classification Problems
- ► Optimization in Classifying Text Documents

References

1. Alekseev DVB (1988) Monotone Boolean functions. Encycl. Math., vol 6. Kluwer, Dordrecht, 306–307
2. Bioch JC, Ibaraki T (1995) Complexity of identifixation and dualization of positive Boolean functions. Inform and Comput 123:50–63
3. Boros E, Hammer PL, Ibaraki T, Kawakami K (1997) Polynomial-time recognition of 2-monotonic positive Boolean functions given by an oracle. SIAM J Comput 26(1):93–109
4. Church R (1940) Numerical analysis of certain free distributive structures. J Duke Math 9:732–734
5. Church R (1965) Enumeration by rank of the elements of the free distributive lattice with 7 generators. Notices Amer Math Soc 12:724
6. Dedekind R (1897) Ueber Zerlegungen von Zahlen durch ihre grössten gemeinsamen Teiler. Festchrift Hoch Braunschweig u Ges Werke, II:103–148
7. Eiter T, Gottlob G (1995) Identifying the minimal transversals of a hybergraph and related problems. SIAM J Comput 24(6):1278–1304
8. Fredman ML, Khachiyan L (1996) On the complexity of dualization of monotone disjunctive normal forms. J Algorithms 21:618–628
9. Gainanov DN (1984) On one criterion of the optimality of an algorithm for evaluating monotonic Boolean functions. USSR Comput Math Math Phys 24(4):176– 181
10. Gorbunov Y, Kovalerchuk B (1982) An interactive method of monotone Boolean function restoration. J Acad Sci USSR Eng 2:3–6 (in Russian.)
11. Hansel G (1966) Sur le nombre des fonctions Boolenes

monotones den variables. CR Acad Sci Paris 262(20):1088–1090
12. Kleitman D (1969) On Dedekind's problem: The number of monotone Boolean functions. Proc Amer Math Soc 21(3):677–682
13. Kleitman D, Markowsky G (1975) On Dedekind's problem: The number of isotone Boolean functions. II. Trans Amer Math Soc 213:373–390
14. Korobkov VK (1965) On monotone functions of the algebra of logic. Probl Kibernet 13:5–28 (in Russian.)
15. Korshunov AD (1981) On the number of monotone Boolean functions. Probl Kibernet 38:5–108 (in Russian.)
16. Kovalerchuk B, Triantaphyllou E, Deshpande AS, Vityaev E (1996) Interactive learning of monotone Boolean functions. Inform Sci 94(1–4):87–118
17. Kovalerchuk B, Triantaphyllou E, Vityaev E (1995) Monotone Boolean functions learning techniques integrated with user interaction. Proc Workshop Learning from Examples vs Programming by Demonstrations: Tahoe City, pp 41–48
18. Makino K, Ibaraki T (1997) The maximum latency and identification of positive Boolean functions. SIAM J Comput 26(5):1363–1383
19. Neapolitan R, Naimipour K (1996) Foundations of algorithms. D.C. Heath, Lexington, MA
20. Peysakh J (1987) A fast algorithm to convert Boolean expressions into CNF. Techn Report IBM Comput Sci RC 12913(57971)
21. Rudeanu S (1974) Boolean functions and equations. North-Holland, Amsterdam
22. Schneeweiss WG (1989) Boolean functions: With engineering applications and computer applications. Springer, Berlin
23. Schneeweiss WG (1996) A necessary and sufficient criterion for the monotonicity of Boolean functions with deterministic and stochastic applications. IEEE Trans Comput 45(11):1300–1302
24. Shmulevich I (1997) Properties and applications of monotone Boolean functions and stack filters. PhD Thesis, Dept. Electrical Engin. Purdue Univ. http://shay.ecn.purdue.edu/~shmulevi/
25. Sokolov NA (1982) On the optimal evaluation of monotonic Boolean functions. USSR Comput Math Math Phys 22(2):207–220
26. Torvik VI, Triantaphyllou E (2000) Minimizing the average query complexity of learning monotone Boolean functions. Working Paper Dept Industrial and Manuf Syst Engin, Louisiana State Univ
27. Triantaphyllou E, Lu J (1998) The knowledge acquisition problem in monotone Boolean systems. Encycl. Computer Sci. and Techn. In: Kent A, Williams JG (eds) MD
28. Ward M (1946) Note on the order of free distributive lattices. Bull Amer Math Soc no. Abstract 135 52:423
29. Wiedemann D (1991) A computation of the eight Dedekind number. Order 8:5–6

Infinite Horizon Control and Dynamic Games

IHDG

Dean A. Carlson[1], Alain B. Haurie[2]
[1] University Toledo, Toledo, USA
[2] University Geneva, Geneva, Switzerland

MSC2000: 91Axx, 49Jxx

Article Outline

Keywords
Unbounded Cost
Nonzero-Sum Infinite Horizon Games
Cooperative Solution
Noncooperative Solutions
See also
References

Keywords

Dynamic optimization; Noncooperative equilibrium; Cooperative equilibrium; Overtaking equilibrium

In economics or biology there is no natural end time for a process. Nations as well as species have a very long future to consider. A mathematical abstraction for this phenomenon is the concept of *infinite time horizon* simply defined as an unbounded time interval of the form $[0, +\infty)$. The study of competing agents in a dynamic deterministic setting over a long time period can be cast in the framework of an infinite horizon dynamic game. This game is defined by the following 'objects':

- A system evolving over an infinite horizon is characterized by a *state* $x \in X \subseteq \mathbf{R}^{m_0}$. Some agents also called the *players* $i = 1, \ldots, p$ can influence the state's evolution through the choice of an appropriate *control* in an admissible class. The control value at a given time n for player i is denoted $u_i(n) \in U_i \subseteq \mathbf{R}_{mi}$.
- The state evolution of such a dynamical system may be described either as a difference equation, if discrete time is used, or a differential equation in a continuous time framework. For definiteness we fix our attention here on a stationary difference equation

and merely remark that similar comments apply for the case when other types of dynamical systems are considered.

$$x(n+1) = f(x(n), u_1(n), \ldots, u_p(n))$$

for $n = 0, 1, \ldots$, where $\mathbf{f}$: $\mathbf{R}^{m_0} \times \ldots \times \mathbf{R}^{mp} \to \mathbf{R}^{m_0}$ is a given state transition function.

- We assume that the agents can observe the state of the system and remember the *history* of the system evolution up to the current time n, that is, the sequence

$$h_n = \{x(0), \mathbf{u}(0), \ldots, \mathbf{u}(n-1), x(n)\},$$

where $\mathbf{u}(n)$ denotes the controls chosen by all players at period n (i. e., $\mathbf{u}(n) = (u_1(n), \ldots, u_p(n))$). A *policy* or a *strategy* is a way for each agent, to adapt his/her current control choice to the history of the system, that is a mapping γ_i: $(n, h_n) \to U_i$ which tells player i which control $u_i(n) \in U_i$ to select given that the time period is n and the state history is h_n.
- Once such a model is formulated the question arises as to what strategy or policy should each agent adopt so that his/her decision provides him/her with the most benefit. The decision to adopt a good strategy is based on a performance criterion defined over the life of the agent (in this case $[0, +\infty)$), that is, for each time horizon N the payoff to player i is determined by

$$J_N^i(x, \mathbf{u}) = \sum_{n=1}^{N} \beta_i^n g_i(x(n), \mathbf{u}(n)),$$

where x and $\mathbf{u}$ denote the state and control evolutions over time, g_i: $\mathbf{R}^{m_0} \times \cdots \times \mathbf{R}^{m_p} \to \mathbf{R}$ is a given reward function and $\beta_i \in [0, 1]$ is a discount factor for each player $i = 1, \ldots, p$.

Two categories of difficulties have to be dealt with when one studies infinite horizon dynamic games:

- the consideration of an unbounded time horizon gives rise to the possibility of having diverging values for the performance criterion (i. e., tending to $+\infty$ on all possible evolutions). This happens typically when there is no discounting ($\beta_i = 1$). A related issue is the stability vs. instability of the optimally controlled system.
- A second category of difficulties are associated with the consideration of all possible actions and reactions of the different agents over time, since an infinite time horizon will always give any agent enough time to correct his/her strategy choice, if necessary.

The first difficulty is already present in a single agent system where the problem reduces to a dynamic optimization problem and is typically cast in the framework of the calculus of variations or optimal control in either discrete or continuous time. The second type of difficulty arises typically in nonzero-sum games.

Unbounded Cost

To introduce the difficulties involved in studying infinite horizon problems we first consider the single player case. The single player case is the most studied of these problems with a relatively rich history beginning with the seminal paper of F. Ramsey [8]. Therefore we shall introduce the subject with the *Ramsey model*, using simpler notations than the one introduced above. In Ramsey's work a continuous time model for the economic growth of a nation is developed and analyzed. In discrete time, the dynamics for Ramsey's model is described by the difference equation

$$x_{n+1} = x_n + f(x_n) - c_n$$

with a fixed initial condition x_0. Here, $x_n \geq 0$ denotes the amount of capital stock at the end of the time period n; $f(x_n)$ is a nonnegative valued function, known as the production function, which is defined for all positive x_n and represents the rate at which capital stock is produced given a stock level x_n; and $c_n > 0$ represents the rate at which the nation consumes the capital stock. Since a nation usually does not consume at a rate faster than it produces we also have the inequality constraint

$$0 \leq c_n \leq f(x_n) \quad \text{for all } n = 1, 2, \ldots.$$

The performance of the system is measured as an accumulation of social welfare over the time scale. Thus, up to a fixed time N, this is represented by the sum

$$J_N(\{c_n\}) = \sum_{n=1}^{N} U(c_n),$$

in which $U(c_n)$ is called a *social utility function* and represents the 'rate of enjoyment' of society at a consump-

tion rate c_n. The goal of a decision maker in this model is to determine c_n, $n = 1, 2, \ldots$, so that

$$\lim_{N\to+\infty} J_N(\{c_n\}) = \lim_{N\to+\infty} \sum_{n=1}^{N} U(c_n)$$

is maximized. An immediate concern in attempting to solve such a problem is that the performance criterion is well defined. That is, for a given feasible element $\{x_n, c_n\}$, $n = 1, 2, \ldots$, is the above infinite series convergent? Additionally, if there exists feasible elements for which the convergence is assured how does one know if the supremum is finite. Ramsey was aware of these two difficulties and these issues were addressed in his work. In dealing with this lack of precision two ideas have arisen. The first of these is to introduce the notion of discounting to 'level the playing field' by scaling units to present value terms. This manifests itself through a positive weighting scheme. Specifically the performance criterion is modified through the introduction of a constant 'discount rate', β, between 0 and 1. That is, the above infinite series is replaced by

$$\lim_{N\to+\infty} J_N(\{c_n)\} = \lim_{N\to+\infty} \sum_{n=1}^{N} \beta^n U(c_n).$$

It is now an easy matter to see that if the sequence $\{U(c_n)\}$ is bounded then the infinite series converges. Moreover, if all feasible sequences $\{c_n\}$ are bounded and $U(\cdot)$ is a continuous function it is easy to see that the supremum (as well as the infimum) over all such sequences is bounded above and the optimization problem is well defined. A criticism of discounting voiced by Ramsey is that it weights a decision makers preference toward the present at the expense of the past. Consequently Ramsey seeks another approach. This alternate idea was that the rate at which a nation consumes is bounded and that ideally the best system would be one in which the rate is as large as possible. Thus Ramsey introduced the notion of a 'maximal sustainable rate of enjoyment' which he referred to as *bliss*. The notion of bliss, denoted by B, is defined now as an *optimal steady state* problem. That is,

$$\begin{aligned} B &= \max\{U(c):\ c = f(x),\ x \geq 0\} \\ &= \max\{U(c):\ c \geq 0\}. \end{aligned}$$

With this idea, the performance index is replaced by a new performance given as

$$\lim_{N\to+\infty} J_N(\{c_n\}) = \lim_{N\to+\infty} \sum_{n=1}^{N} B - U(c_n),$$

and the goal is to choose $\{c_n\}$ as a minimizer instead of a maximizer. Observe that $B - U(c_n) \geq 0$ for all n so that the above limit is bounded below by zero. Thus, if bliss is attained by some feasible sequence (that is, $c_n = \overline{c}$ for all n sufficiently large with $B = U(\overline{c})$, then the performance criterion is finite for at least one feasible element $\{x_n, c_n\}$ and the minimization problem is well defined. Using the notion of bliss, Ramsey solved this problem using classical variational analysis (i. e., the Euler–Lagrange equation from the calculus of variations) to arrive at what is now referred to as *Ramsey's rule of economic growth*. Finally we remark that the solution, say $\{x_n^*, c_n^*\}$, obtained by Ramsey asymptotically approaches $\{\overline{x}, \overline{c}\}$, where $\overline{x}$ the unique solution to the equation $\overline{c} = f(x)$.

The approach adopted by Ramsey in his model has become a prototype for studying more complex problems. In particular, the notion of bliss and the optimal steady state problem combined with the idea that bliss is obtained in finite time is now referred to as a *reduction to finite costs*. Finally the asymptotic convergence to the optimal steady state is referred to as an asymptotic turnpike property.

Since Ramsey 'solved' his problem through an application of necessary conditions he did not directly address the question of existence of an optimal solution. He assumed that the solution to the necessary condition was in fact a solution. However, in 1962, S. Chakravarty [4] gave a simple example in which the solution of Ramsey's rule was not a minimizer but a maximizer! This led to the quest for the existence of optimal solutions for these problems. As the performance objective is unbounded, the traditional notion of a minimizer is no longer valid. Thus, new types of optimality were introduced in the 1960s by C.C. von Weizäcker [10] to deal with this problem. These notions are now known as *overtaking optimality*, *weakly overtaking optimality*, and *finite optimality*. The most useful and strongest of these three types of optimality is overtaking optimality. In words, a sequence $\{x_n^*, c_n^*\}$ is overtaking optimal if when compared with any other fea-

sible sequence $\{x_n, c_n\}$ the finite horizon performance criterion, $J_N(\{c_n^*\})$ is larger than $J_N(\{c_n\})$ to within an arbitrarily small margin of error for all N sufficiently large.

The introduction of new types of optimality led to new important results concerning these problems. The first necessary conditions for these types optimality were given in 1974 by H. Halkin [5] in which the classical Pontryagin maximum principle was extended. Of particular notice in this result was the fact that the classical transversality condition found in corresponding finite horizon problems does not necessarily hold. This fact led to many results which insure some sort of boundary condition holds at infinity. The first general existence theorem for these optimization problems was given by W.A. Brock and H. Haurie [1] in 1976. During the 1980s these major results were extended in a variety of directions and many of these results are discussed in [3].

Nonzero-Sum Infinite Horizon Games

We now turn our attention to p-player games. We use from now on the general notations introduced in the introduction. To simplify a little the exposition we shall use a simplified paradigm where each player is controlling his/her own dynamical system. Hence each player enjoys his/her own state and control, say $\{x_i(n), u_i(n)\}$ for $i = 1, \ldots, p$ and $n = 1, 2, \ldots$, and has a performance criterion, say $J_N^i(\mathbf{x}, \mathbf{u})$, which is described in discrete time up to the end of period N as

$$J_N^i(\mathbf{x}, \mathbf{u}) = \sum_{n=1}^{N} g_i(\mathbf{x}(n), \mathbf{u}(n)).$$

Here we use the notation

$$\mathbf{x}(n) = \{(x_1(n), \ldots, x_p(n))\}$$

and

$$\mathbf{u}(n) = \{(u_1(n), \ldots, u_p(n))\}.$$

From the notation we see that each players performance measure depends not only on their own decision but also those of the other players. This coupling may also occur in the dynamical system as well. In discrete time these systems may be represented by a system of p difference equations

$$x_i(n+1) = f_i(\mathbf{x}(n), \mathbf{u}(n))$$

for $n = 0, 1, \ldots$ and $i = 1, \ldots, p$.

The goal of each of the players is to 'play' the game so that their decisions provide them with the best performance possible. This action is in conflict with the other players and therefore generally it is not possible for the players to minimize or maximize their performance. The way one defines *optimality* in a game depends on the *mood of play*, i. e. if the players behave in a *cooperative* or in a *noncooperative* fashion.

Cooperative Solution

If players cooperate they will want to reach an undominated solution, also called a *Pareto solution* after its originator, V. Pareto [7], who introduced the concept in 1896. A pair $\{\mathbf{x}, \mathbf{u}\}$ is called a cooperative solution if there does not exist a feasible point $\{\mathbf{y}, \mathbf{v}\}$ satisfying $J^i(\mathbf{x}, \mathbf{u}) \geq J^i(\mathbf{y}, \mathbf{v})$ for all players $i = 1, \ldots, p$ with at least one strict inequality for one of the players. It is well known that such an equilibrium can be obtained by solving an appropriate single player game in which the payoff is a weighted sum of the payoffs of all of the players

$$\mathbf{J_r}(\mathbf{x}, \mathbf{u}) = \sum_{j=1,\ldots,p} r_j J^i(\mathbf{x}, \mathbf{u}), \qquad r_j \geq 0, \quad j = 1, \ldots, p.$$

In this way the problem is reduced to the case of infinite horizon optimization and the remarks made earlier apply.

Noncooperative Solutions

If players do not cooperate one may consider that they will be satisfied of the outcome if, for each player, his/her strategy is the best response he/she can make to the strategies adopted by the other players. This is the concept of *equilibrium*, introduced in 1951 by J.F. Nash [6] in the context of matrix games. In general, the search for a *Nash equilibrium* can not be reduced to an optimization problem. Since each players decision is his/her best decision under the assumption that the decisions of the other players are fixed, the search for an equilibrium is equivalent to the search for a fixed-point

of a reaction mapping that associates with each strategy choice by the p players the set of optimal responses by each of them. To better understand this concept it is preferable to consider first a game defined in its *normal form*. Let $\gamma_j \in \Gamma_j$ design the strategies of player j. Let $V_j(\gamma_1, \ldots, \gamma_p) \in \mathbf{R}$ be the payoff to player j associated with the strategy choices $\underline{\gamma} = (\gamma_1, \ldots, \gamma_p)$ of the p players. $\underline{\gamma}^*$ is a Nash equilibrium if

$$V_j(\gamma_1^*, \ldots, \gamma_j, \ldots, \gamma_p^*) \leq V_j(\underline{\gamma}^*), \quad \forall \gamma_j \in \Gamma_j, \quad j = 1, \ldots, p.$$

Now we introduce the product strategy set $\underline{\Gamma} = \prod_{j=1}^{p} \Gamma_j$ and the mapping

$$\sigma : \underline{\Gamma} \times \underline{\Gamma} \to \mathbf{R},$$

$$\sigma(\underline{\gamma}^1, \underline{\gamma}^2) = \sum_{j=1}^{p} V_j(\gamma_1^1, \ldots, \gamma_j^2, \ldots, \gamma_p^1).$$

Finally let us define the point to set mapping $\Psi : \underline{\Gamma} \to 2^{\underline{\Gamma}}$ defined by

$$\Psi(\underline{\gamma}) = \left\{ \underline{\tilde{\gamma}}^+ : \sigma(\underline{\gamma}, \underline{\gamma}^+) = \sup_{\underline{\gamma}^0 \in \underline{\Gamma}} \sigma(\underline{\gamma}, \underline{\gamma}^0) \right\}.$$

Ψ is the *best response mapping* for the game. A fixed-point of Ψ is a strategy vector $\underline{\gamma}^*$ such that

$$\underline{\gamma}^* \in \Psi(\underline{\gamma}^*).$$

$\underline{\gamma}^*$ is a fixed-point of Ψ if and only if it is a Nash equilibrium.

In a dynamic setting the concept of strategy is closely related to the *information structure* of the game. We have assumed, in the beginning that the players can remember the whole (state and control) history of the dynamical system they contribute to control. This is the most precise information that can be available to the players at each instant of time. On the other end we can assume that the only information available to a player is the initial state of the system $x^0 = x(0)$ and the current time t. A strategy γ_j for player j will thus be an *open-loop control* $\{u_j(n)\}_n = 0, \ldots, \infty$. An equilibrium in this class of strategies is called an *open-loop Nash equilibrium*. An intermediate case is the one where each player can observe the state of the system at each time period but does not recall the previous history of the system, neither the state nor the control values. A (stationary) strategy γ_j for player j will thus be a *closed-loop control* or a feedback control $\gamma_j : \mathbf{x} \longmapsto u_j = \gamma_j(\mathbf{x})$. An equilibrium in this class of strategies is called a *feedback Nash equilibrium*. In the economics literature, feedback strategies are also called *Markov strategies* to emphasize the lack of memory in the information structure.

For a single agent deterministic system, i. e. an optimal control problem, the information structure does not really matter. The agent will not be able to do better than the optimal open-loop control, even if he/she has a perfect memory. In a two-player zero-sum dynamic game this will also be the case. In a nonzero-sum game the different information structures lead to different types of equilibria.

A criticism of the open-loop Nash equilibrium is that it is not necessarily *subgame perfect* in the sense of R. Selten [9]. This means that if a player deviates from the equilibrium control for a while and then decides to play again 'correctly', then the previously defined equilibrium is not an equilibrium any more. A *feedback Nash equilibrium* can be made subgame perfect if one uses *dynamic programming* to characterize it. A *memory strategy Nash equilibrium* can also be made subgame perfect. Furthermore, the possibility to remember past actions or state values permit the player to define a so-called *communication equilibrium* where, before the play the agents communicate with each other and decide to use a specific *memory strategy equilibrium*. The memory permits the inclusion of threats that would support a cooperative outcome. The cooperative outcome becomes also a Nash equilibrium outcome. This type of results have been known as the *'folks theorem'* in economics. The infinite horizon is essential to obtain this type of result.

Nevertheless, the open-loop concept still has wide interest for a variety of reasons. In infinite horizon games the notion of overtaking Nash equilibrium is defined analogously to the concept in the single-player. These ideas have just recently begun to be studied extensively with the first existence theory for open-loop Nash equilibria and a corresponding turnpike theory being given in 1996 in [2]. Finally, from a practical setting, the numerical computation of a feedback Nash equilibrium is much less understood than the computation of an open-loop Nash equilibrium. The analo-

gous theory for feedback (or closed-loop) equilibria is still waiting to be developed.

In closing, the theory of infinite horizon dynamic games is for the most part still in its infancy and much remains to be studied and researched. One important open question concerns the existence of overtaking feedback Nash equilibria and another is that once such an equilibrium is known to exist can a robust numerical procedure for computation of equilibrium be developed.

See also

- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Optimal Control of a Flexible Arm
- ▶ Optimization Strategies for Dynamic Systems
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Brock WA, Haurie A (1976) On existence of overtaking optimal trajectories over an infinite time horizon. Math Oper Res 1:337–346
2. Carlson DA, Haurie A (1996) A turnpike theory for infinite horizon open-loop competitive processes. SIAM J Control Optim 34(4):1405–1419
3. Carlson DA, Haurie A, Leizarowitz A (1991) Infinite horizon optimal control: Deterministic and stochastic systems, 2nd edn. Springer, Berlin
4. Chakravarty S (1962) The existence of an optimum savings program. Econometrica 30:178–187
5. Halkin H (1974) Necessary conditions for optimal control problems with infinite horizon. Econometrica 42:267–273
6. Nash JF (1951) Non-cooperative games. Ann of Math 54:286–295
7. Pareto V (1896) Cours d'economie politique. Rouge, Lausanne
8. Ramsey F (1928) A mathematical theory of saving. Economic J 38:543–549
9. Selten R (1975) Reexamination of the perfectness concept for equilibrium points in extensive games. Internat J Game Theory 4:25–55
10. Von Weizäcker CC (1965) Existence of optimal programs of accumulation for an infinite time horizon. Review of Economic Studies 32:85–104

Information-based Complexity and Information-based Optimization

J. F. Traub[1], A. G. Werschulz[2,1]

[1] Department Computer Sci., Columbia University, New York, USA

[2] Department Computer and Information Sci., Fordham University, New York, USA

MSC2000: 65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26

Article Outline

Keywords
Information-Based Complexity
  Computational Complexity of High-Dimensional Integration
  Mathematical Finance
  General Theory
Information-Based Optimization
See also
References

Keywords

Information-based complexity; Information-based optimization; Real number model; Linear programming; Nonlinear optimization; High-dimensional integration; Mathematical finance; Curse of dimensionality

This article concerns optimization in two senses. The first is that *information-based complexity* (IBC) is the

study of the minimal computational resources to solve continuous mathematical problems. (Other types of mathematical problems are also studied; the problems studied by IBC will be characterized later.) J.F. Traub and A.G. Werschulz [14] provide an expository introduction to the theory and applications of IBC, with over 400 recent papers and books. A general formulation with proofs can be found in [13].

The second is that the *computational complexity of optimization problems* is one of the areas studied in IBC. S.A. Vavasis [16 pag. 135] calls this *information-based optimization*. We will discuss information-based complexity and information-based optimization in turn.

Information-Based Complexity

To introduce computational complexity, we first define the *model of computation*. The model of computation states which operations are permitted and how much they cost. The model of computation is based on two assumptions:

1) We can perform arithmetic operations and comparisons on real numbers at unit cost.
2) We can perform an information operation at cost c. Usually, $\gg 1$.

We comment on these assumptions. The *real number model* (Assumption 1) is used as an abstraction of the floating-point model typically used in scientific computation. Except for the possible effect of roundoff errors and numerical stability, complexity results will be the same in these two models.

The real number model should be contrasted with the *Turing machine model*, typically used for discrete problems. The cost of an operation in a Turing machine model depends on the size of the operands, which is not a good assumption for floating point numbers. For a full discussion of the pros and cons of the Turing machine and real number models see [14 Chapt. 8]. Whether the real number or Turing machine model is used can make an enormous difference. For example, L.G. Khachiyan [3] shows that *linear programming* is polynomial in the Turing machine model. In 1982, Traub and H. Woźniakowski [15] showed that Khachiyan's algorithm is not polynomial in the real number model and conjectured that linear programming is not polynomial in this model. This conjecture is still open.

The purpose of information operations (Assumption 2) is to replace the input by a finite set of numbers. For integration, the information operations are typically function evaluations.

Computational Complexity of High-Dimensional Integration

We illustrate some of the important ideas of IBC with the example of high-dimensional integration.

We wish to compute the integral of a real-valued function f of d variables over the unit cube in d dimensions. Typically, we have to settle for computing a numerical approximation with an error ε. To guarantee an ε-approximation we have to know some global information about the integrand. We assume that the class F of integrands has smoothness r. One such class is F_r, which consists of those functions having continuous derivatives of order through r, these derivatives satisfying a uniform bound.

A real function of a real variable cannot be entered into a digital computer. We evaluate f at a finite number of points and we call the set of values of f the local information, for brevity *information*, about f. An algorithm combines the function values into a number that approximates the integral.

In the worst-case setting we want to guarantee an error at most ε for every $f \in F$. The computational complexity, for brevity complexity, is the least cost of computing the integral to within ε for every f. We want to stress that the complexity depends on the problem and on ε, but not on the algorithm. Every possible algorithm, whether or not it is known, and all possible points at which the integrand is evaluated are permitted to compete when we consider least possible cost.

It can be shown that if $F = F_r$, then the complexity of our integration problem is of order ε^{-dr}. If $r = 0$, e. g., if our set of integrands consists of uniformly bounded continuous functions, the complexity is infinite. That is, it is impossible to solve the problem to within ε. Let r be positive and in particular let $r = 1$. Then the complexity is of order $\varepsilon^{-\mathrm{d}}$. Because of the exponential dependence on d, we say the problem is computationally intractable. This is sometimes called the *curse of dimensionality*.

We will compare this d-dimensional integration problem with the well-known traveling salesman prob-

lem (TSP), an example of a discrete combinatorial problem. The input is the location of the n cities and the desired output is the minimal route; the city locations are usually represented by a finite number of bits. Therefore the input can be exactly entered into a digital computer. The complexity of this problem is unknown but conjectured to be exponential in the number of cities. That is, the problem is conjectured to be computationally intractable and many other combinatorial problems are conjectured to be intractable.

Most problems in scientific computation which involve multivariate functions belonging to F_r have been proven computationally intractable in the number of variables in the worst-case setting. These include nonlinear equations [10], partial differential equations [19], function approximation [7], integral equations [19], and optimization [6]. Material on the computational complexity of optimization will be presented in the second half of this article.

Very high-dimensional integrals occur in many disciplines. For example, problems with dimension ranging from the hundreds to the thousands occur in mathematical finance. Path integrals, which are of great importance in physics, are infinite-dimensional, and therefore invite high-dimensional approximations. This motivates our interest in breaking the curse of dimensionality. Since this is a complexity result, we cannot get around it by a clever algorithm. We can try to break the curse by settling for a stochastic assurance rather than a worst-case deterministic assurance. Examples of stochastic assurance are provided by the randomized and average case settings which we will consider below. We can also try to break the curse by changing the class of inputs. A good example of this occurs in mathematical finance.

Mathematical Finance

The valuation of financial instruments often requires the calculation of very high-dimensional integrals. Dimensions of 360 and higher are not unusual. Furthermore, since the integrals can be very complicated requiring between 10^5 and 10^6 floating point operations per integrand evaluation, it is important to minimize the number of evaluations. Extensive numerical testing shows that these problems do not suffer from the curse of dimensionality. A possible explanation is given by I. Sloan and Woźniakowski [11], who show that the curse can be broken by changing the class of integrands to capture the essence of the mathematical finance problem. See [14 Chapt. 4] for a survey of high-dimensional integration and mathematical finance.

General Theory

In general, IBC is defined by the assumptions that the *information* concerning the mathematical model is

- partial,
- contaminated, and
- priced.

Referring to the integration example, the mathematical input is the integrand and the information is a finite set of function values. It is *partial* because the integral cannot be recovered from function values. For a partial differential equation the mathematical input consists of the functions specifying the initial value and/or boundary conditions. Generally, the mathematical input is replaced using a finite number of information operations. These operations may be functionals on the mathematical input or physical measurements that are fed into a mathematical model.

In addition to being partial the information is often *contaminated* by, for example, round-off or measurement error ([8]). If the information is partial or contaminated it is impossible to solve the problem exactly. Finally, the information is *priced*. As examples, functions can be costly to evaluate or information needed for oil exploration models can be obtained by setting off shocks. With the exception of certain finite-dimensional problems, such as roots of systems of polynomial equations and problems in numerical linear algebra, the problems typically encountered in scientific computation have information that is partial and/or contaminated and priced.

As part of our study of complexity we investigate *optimal algorithms*, that is, algorithms whose cost is equal or close to the complexity of the problem. This has sometimes led to new solution methods. The reason that we can often obtain the complexity and an optimal algorithm for IBC problems is that partial and/or contaminated information permits arguments at the information level. This level does not exist for combinatorial problems where we usually have to settle for trying

to establish a complexity hierarchy and trying to prove conjectures such as P $\neq$ NP.

A powerful tool at the information level is the notion of the *radius of information*, R. The radius of information measures the intrinsic uncertainty of solving a problem using given information. We can compute an ε-approximation if and only if $R \leq \varepsilon$. The radius depends only on the problem being solved and the available information; it is independent of the algorithm. The radius of information is defined in all IBC settings.

Information-Based Optimization

We turn to the application of IBC concepts to information-based optimization.

In their seminal book, A.S. Nemirovsky and D.B. Yudin [6] study a *constrained optimization* problem. They wish to minimize a nonlinear function subject to nonlinear constraints. Let $f = [f_0, \ldots, f_m]$, where f_0 denotes the objective function and $f_1, \ldots, f_m$ denote constraints. Let F be the product of $m+1$ copies of F_r. Then

$$\mathrm{comp}(\varepsilon) = \Theta\left(\left(\frac{1}{\varepsilon}\right)^{d/r}\right).$$

Thus this problem suffers from the curse of dimensionality.

Vavasis [16 Chapt. 6] reports on the worst-case complexity of minimizing an objective function with box constraints. He assumes objective functions defined on the unit cube in d dimensions and takes F as the class of continuous functions with uniform Lipschitz constant L. For *global minimization*,

$$\mathrm{comp}(\varepsilon) = \Theta\left(\left(\frac{L}{2\varepsilon}\right)^{d}\right).$$

Thus global minimization is intractable.

In contrast to global minimization, the problem of computing a *local minimum* is tractable with suitable conditions on F. Let F consist of continuously differentiable real functions on $[0, 1]^d$ whose gradients satisfy a uniform Lipschitz condition with constant M. Then $4d(M/\varepsilon)^2$ function and gradient evaluations are sufficient.

As discussed above, there are two ways one can attempt to break the curse of dimensionality: by settling for a stochastic assurance, or by changing the class of inputs. For the constrained optimization problem, we first describe changing the class of functions, and then turn to weakening the assurance.

Nemirovsky and Yudin [6] take $F = F_{\mathrm{conv}}$ to be the class of convex functions that satisfy a Lipschitz condition with a uniform constant on a bounded convex set D. Then

$$\mathrm{comp}(\varepsilon) = \Theta\left(\log \frac{1}{\varepsilon}\right),$$

where the constant in the Θ-notation depends polynomially on the dimension d of D and m, the number of constraints. Thus, convexity breaks the curse of dimensionality.

The worst-case deterministic assurance may be weakened to a stochastic assurance; we report on the randomized and average case settings.

Nemirovsky and Yudin [6] show that *randomization* does not break the curse of dimensionality for computing the minimum value of the nonlinear constrained problem. G.W. Wasilkowski [17] establishes an even more negative result if an ε-approximation to the value of x that minimizes f_0 is sought. He permits randomization and shows that for all $\varepsilon < 1/2$, this problem is unsolvable even if $d = 1$.

The results considered so far use a sequential model of computation. One could also ask about the complexity under a *parallel model of computation*. If we have k processors running in parallel, how much can the computation of the minimum be sped up? Clearly, the best possible speedup is k. Nemirovsky [5] considers this problem for the case $F = F_{\mathrm{conv}}$, showing that

$$\mathrm{comp}^{\mathrm{par}}(\varepsilon, k) = \Omega\left(\left(\frac{d}{\ln(2kd)}\right)^{1/3} \ln\left(\frac{1}{\varepsilon}\right)\right),$$

where the Ω-constant is independent of k and ε. Hence we find that

$$\frac{\mathrm{comp}(\varepsilon)}{\mathrm{comp}^{\mathrm{par}}(\varepsilon, k)} = O\left(\left(\frac{\ln(2kd)}{d}\right)^{1/3}\right),$$

which is much less than k. Thus parallel computation is not very attractive for this problem.

The average case setting looks more promising than the randomized setting, but since it is technically very

difficult, the results to date are quite limited. In the average case setting we want to guarantee that the expected error is at most ε and we minimize the expected cost.

In the *average case setting*, an a priori measure must be placed on F. Typically, this measure is *Gaussian*; in particular, *Wiener measures* are used. Since the distribution of the random variable $\min_x f(x)$ is difficult to obtain, the average case analysis of the global optimization problem is very difficult. Only partial results have been obtained.

Let $d = 1$ and $F \subset C^r[0, 1]$. Assume that F is endowed with the r-fold Wiener measure. Wasilkowski [18] shows that approximately $(\varepsilon^{-1}\sqrt{\ln \varepsilon^{-1}})^{1/(r+1/2)}$ function evaluations suffice. This is better than the worst case, where some $\varepsilon^{-1/r}$ function values are needed.

Stronger results have been obtained for the case of $d = 1$ and $r = 0$, i. e., optimization for continuous scalar functions, equipped with the Wiener measure. K. Ritter [9] considers the case of *nonadaptive methods*, showing that

$$\mathrm{comp}^{\mathrm{non}}(\varepsilon) = \Theta\left(\left(\frac{1}{\varepsilon}\right)^2\right).$$

Moreover, the optimal evaluation points are equidistant knots. More recently (1997), J.M. Calvin [1] investigates *adaptive methods* for this problem, showing that for any $\delta \in (0, 1)$,

$$\mathrm{comp}^{\mathrm{ad}}(\varepsilon) = O\left(\left(\frac{1}{\varepsilon}\right)^{1/(1-\delta)}\right).$$

The study of optimization in the average case setting is a very promising area for future research. Important open problems include:

- obtaining multivariate results,
- obtaining lower bounds,
- obtaining better upper bounds.

We now restrict our attention to the special optimization problem of linear programming (LP), which we discuss in the worst-case setting.

In 1979, Khachiyan [3] studied an *ellipsoid algorithm* and proved that LP is polynomial in the Turing machine model. In 1982, Traub and Woźniakowski [15] showed that the cost of this ellipsoid algorithm is not polynomial in the real-number model, and conjectured that the LP problem is not polynomial in the real-number model. This nicely illustrates the difference between the cost of an algorithm and the complexity of a problem, since the result concerning the cost of the ellipsoid algorithm leaves open the question of problem complexity. The Traub–Woźniakowski conjecture remains open.

A related open question is whether LP can be solved in *strongly polynomial time*. (Note that the underlying models of computation are different: the real-number model versus the Turing machine model.) This question is also still open, with results known only for special cases. In 1984, N. Megiddo [4] showed that LP can be solved in linear time if the number of variables is fixed, while in 1986, É. Tardos [12] showed that many LP problems that arise from combinatorial applications can be solved in strongly polynomial time.

We now discuss the computation of *fixed points*, which we include here because the result involves ellipsoid methods. The problem is to compute the fixed point of $f(x)$; that is, to solve the nonlinear equation $x = f(x)$ for any $f \in F$, where F is the class of functions on $[0, 1]^d$ having a Lipschitz constant of q, with $q \in (0, 1)$.

The simple iteration algorithm $x_{i+1} = f(x_i)$, with $x_0 = 0$, can compute an ε-approximation with at most

$$n_{\mathrm{si}}(\varepsilon, q) = \left\lceil \frac{\ln 1/\varepsilon}{\ln 1/q} \right\rceil$$

evaluations of f. Thus the simple iteration algorithm behaves poorly if q is close to one.

Z. Huang, Khachiyan, and K. Sikorski [2] show that an inscribed ellipsoid algorithm computes an ε-approximation with

$$n_{\mathrm{e}}(\varepsilon, q) = O\left(d\left(\ln\frac{1}{\varepsilon} + \ln\frac{1}{1-q}\right)\right)$$

function evaluations. Thus their algorithm is excellent for computing fixed points of functions with q close to unity; that is, almost noncontracting functions.

See also

- ▶ Complexity Classes in Optimization
- ▶ Complexity of Degeneracy
- ▶ Complexity of Gradients, Jacobians, and Hessians
- ▶ Complexity Theory
- ▶ Complexity Theory: Quadratic Programming

- ▶ Computational Complexity Theory
- ▶ Fractional Combinatorial Optimization
- ▶ Kolmogorov Complexity
- ▶ Mixed Integer Nonlinear Programming
- ▶ NP-complete Problems and Proof Methodology
- ▶ Parallel Computing: Complexity Classes

References

1. Calvin JM (1997) Average performance of adaptive algorithms for global optimization. Ann Appl Probab 34:711–730
2. Huang Z, Khachiyan L, Sikorski K (1999) Approximating fixed points of weakly contracting mappings. J Complexity 15:200–213
3. Khachiyan LG (1979) A polynomial algorithm in linear programming. Soviet Math Dokl 20:191–194
4. Megiddo N (1984) Linear programming in linear time when the dimension is fixed. J ACM 31:114–127
5. Nemirovsky AS (1994) On parallel complexity of nonsmooth convex optimization. J Complexity 10:451–463
6. Nemirovsky AS, Yudin DB (1983) Problem complexity and method efficiency in optimization. Wiley/Interscience, New York
7. Novak E (1988) Deterministic and stochastic error bounds in numerical analysis, vol 1349. Springer, Berlin
8. Plaskota L (1996) Noisy information and computational complexity. Cambridge Univ. Press, Cambridge
9. Ritter K (1990) Approximation and optimization on the Wiener space. J Complexity 337–364
10. Sikorski K (2000) Optimal solution of nonlinear equations. Oxford Univ. Press, Oxford
11. Sloan IH, Woźniakowski H (1998) When are quasi-Monte Carlo algorithms efficient for high dimensional integrals? J Complexity 14:1–33
12. Tardos E (1986) A strongly polynomial algorithm to solve combinatorial linear programs. Oper Res 34(2):250–256
13. Traub JF, Wasilkowski GW, Woźniakowski H (1988) Information-based complexity. Acad. Press, New York
14. Traub JF, Werschulz AG (1998) Complexity and information. Cambridge Univ. Press, Cambridge
15. Traub JF, Woźniakowski H (1982) Complexity of linear programming. Oper Res Lett 1:59–62
16. Vavasis SA (1991) Nonlinear optimization: Complexity issues. Oxford Univ. Press, Oxford
17. Wasilkowski GW (1989) Randomization for continuous problems. J Complexity 5:195–218
18. Wasilkowski GW (1992) On average complexity of global optimization problems. Math Program 57(2)(Ser. B):313–324
19. Werschulz AG (1991) The computational complexity of differential and integral equations: An information-based approach. Oxford Univ. Press, Oxford

Integer Linear Complementary Problem

Integer LCP

PANOS M. PARDALOS[1], YASUTOSHI YAJIMA[2]

[1] Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

[2] Tokyo Institute Technol., Tokyo, Japan

MSC2000: 90C25, 90C33

Article Outline

Keywords
See also
References

Keywords

LCP; Zero-one integer programming

Let M be a given $n \times n$ matrix, and let q be a given n vector. The *linear complementary problem* (LCP; cf. also ▶ Linear complementarity problem) is to find a vector x which satisfies the following system:

$$\begin{cases} x \geq 0, \\ Mx + q \geq 0, \\ x^{\top}(Mx + q) = 0. \end{cases} \tag{1}$$

When some or all the variables are required to be integers, the problem is called *integer linear complementary problem* (ILCP).

Suppose that for each i ($i = 1, \ldots, k$), the variable x_i is required to be integer among

$$x_i \in \{0, \ldots, n_i\},$$

while for each i ($i = k+1, \ldots, n$), the variable x_i is continuous and

$$0 \leq x_i \leq \beta_i.$$

The problem can be formulated as the feasibility problem which finds a solution x and z such that

$$\begin{cases} 0 \le M_i x + q_i \le B_i(1 - z_i), & \\ 0 \le n_i z_i - x_i, & i = 1, \ldots, k, \\ x_i \in \{0, \ldots, n_i\}, & i = 1, \ldots, k, \\ 0 \le \beta_i z_i - x_i, & i = k+1, \ldots, n, \\ 0 \le x_i \le \beta_i, & i = k+1, \ldots, n, \\ z \in \{0,1\}^n, & \end{cases} \tag{2}$$

where M_i is the ith row of M, q_i is the ith component of q and B_i is the optimal value of the following problem:

$$\begin{cases} \max & M_i x + q_i \\ \text{s.t.} & x_i \in \{0, \ldots, n_i\}, \quad i = 1, \ldots, k, \\ & 0 \le x_i \le \beta_i, \quad i = k+1, \ldots, n, \end{cases}$$

which can be solved analytically.

It has been shown [2] that if the region (2) is empty, the associated (ILCP) has no solution. Otherwise, if $(\overline{x}, \overline{z})$ satisfies (2), then $\overline{x}$ solves the (ILCP).

Obviously, when all the variables of LCP are zero-one integers, the (ILCP) is formulated as a *zero-one integer feasibility problem* of the form:

$$\begin{cases} \text{Find} & x, z \\ \text{s.t.} & 0 \le M_i x + q_i \le B_i(1 - z_i), \\ & 0 \le z_i - x_i, \quad i = 1, \ldots, n, \\ & x, z \in \{0,1\}^n, \end{cases} \tag{3}$$

where

$$B_i = \max\{M_i x + q_i : \ x \in \{0,1\}^n\}.$$

It is worth noting that the *minimum norm solution* of the zero-one (ILCP) can be obtained by solving the following *linear zero-one integer problem*:

$$\begin{cases} \min & \sum_{i=1}^{n} x_i \\ \text{s.t.} & (x, z) \text{ satisfies (3).} \end{cases} \tag{4}$$

We note that there are many algorithms for solving the problem with practical size.

Integer variables without known upper bounds make the problem much harder. Let us consider the (ILCP) defined below:

$$\begin{cases} \text{Find} & x \\ \text{s.t.} & x \ge 0, \\ & Mx + q \ge 0, \\ & x^\top (Mx + q) = 0, \\ & x_i \text{ is integer for } i = 1, \ldots, k. \end{cases}$$

This problem can be rewritten in the form:

$$\begin{cases} \text{Find} & \alpha, y, z \\ \text{s.t.} & 0 \le My + \alpha q \le e - z, \\ & 0 \le \alpha, \\ & 0 \le y \le z, \\ & z \in \{0,1\}^n, \\ & y_i/\alpha \text{ is integer for } i = 1, \ldots, k, \end{cases} \tag{5}$$

where e is a vector of all ones. If $(\overline{\alpha}, \overline{y}, \overline{z})$ solves (5), i. e., for each $i = 1, \ldots, k$, $\overline{y_i}/\overline{\alpha}$ is integer, $x = \overline{y}/\overline{\alpha}$ solves its associated (ILCP). See [3] and [2] for a proof of the equivalence.

The (ILCP) arises in several contexts such as polymatrix games in pure strategies, economic equilibrium with discrete activity levels and spatial price equilibrium in discrete commodities. See [2], for details.

Let us consider the *polymatrix game*. Suppose that each player i ($i = 1, \ldots, n$) has a finite number m_i of strategies, and the partial payoffs to player i, resulting from choices by him/her and player j, are given by $m_i \times m_j$ matrices A^{ij} ($i, j = 1, \ldots, n$). The elements of A^{ij} are assumed to be positive without loss of generalities.

Let

$$X^{i\top} = (X^i_1, \ldots, X^i_{m_i}), \quad i = 1, \ldots, n,$$

be a vector where each component X^i_s expresses the probability of i playing his sth strategy. It has been shown [1] that finding equilibria of polymatrix games is equivalent to find solutions of the linear complementarity problem defined below.

Let

$$v^i = (X^i)^\top \sum_{j \ne i} A^{ij} X^j, \quad i = 1, \ldots, n.$$

Also, let us define

$$x^\top = \left(X^{1\top}, \ldots, X^{n\top}, v^1, \ldots, v^n\right),$$

$$q^\top = (\overbrace{0, \ldots, 0}^{\sum_{i=1}^n m_i}, \overbrace{-1, \ldots, -1}^{n}),$$

and

$$M = \begin{pmatrix} 0 & A^{12} & \cdots & A^{1n} & -e & 0 & \cdots & 0 \\ A^{21} & 0 & \cdots & A^{2n} & 0 & -e & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ A^{n1} & A^{n2} & \cdots & 0 & 0 & 0 & \cdots & -e \\ e^\top & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & e^\top & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & e^\top & 0 & 0 & \cdots & 0 \end{pmatrix},$$

where e is a vector of all ones whose dimension is given by context. Then, the above polymatrix game can be equivalently written as (1). Moreover, suppose that some players, i ($i = 1, \ldots, k$), can select only one pure strategies, while the other players, i ($i = k+1, \ldots, n$), can select mixed strategies. For each player i ($i = 1, \ldots, k$), the vector X^i is required to be zero-one integer, which results (ILCP).

See also

- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Convex-simplex Algorithm
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Equivalence Between Nonlinear Complementarity Problem and Fixed Point Problem
- ▶ Generalized Nonlinear Complementarity Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Lemke Method
- ▶ Linear Complementarity Problem
- ▶ Linear Programming
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Order Complementarity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Principal Pivoting Methods for Linear Complementarity Problems
- ▶ Sequential Simplex Method
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem
- ▶ Topological Methods in Complementarity Theory

References

1. Howson JT Jr (1972) Equilibria of polymatrix games. Managem Sci 18(5):312–318
2. Pardalos PM, Nagurney A (1990) The integer linear complementarity problem. Internat J Comput Math 31:205–214
3. Pardalos PM, Rosen JB (1987) Global optimization approach to the linear complementarity problem. SIAM J Sci Statist Comput 9(2):341–353

Integer Linear Programs for Routing and Protection Problems in Optical Networks

Meeyoung Cha[1], W. Art Chaovalitwongse[2], Zhe Liang[2], Jennifer Yates[3], Aman Shaikh[3], Sue B. Moon[4]

[1] Division of Computer Science, Korean Advanced Institute of Science and Technology, Daejeon, Korea
[2] Department of Industrial and Systems Engineering, Rutgers University, Piscataway, USA
[3] AT&T Labs – Research, Florham Park, USA
[4] Division of Computer Science, Korean Advanced Institute of Science and Technology, Daejeon, Korea

MSC2000: 68M10, 90B18, 90B25, 46N10

Article Outline

Keywords and Phrases
Introduction
Motivations and Challenges in Optimization Models
Path-Protection (Diverse) Routing Problem
Minimum Color Problem
Finding the Dual Link Problem
SRRG-Diverse Routing Problem
Shared Path-Protection Problem
Path-Restoration Problem
Optical Path Restoration Problem
IP Path–Restoration Problem
Concluding Remarks
References

Keywords and Phrases

Integer linear programming; Routing; Optical networks; Diverse path

Introduction

Owing to the exponential growth of traffic demand, there is an emerging challenge as well as an opportunity for service providers to employ Internet Protocol (IP) backbone networks carried on top of optical transport networks, forming IP over optical infrastructure. This technology is poised to take over most of broadband operational services as an integrated transport platform for the following reasons. Optical networks offer the capability to carry numerous wavelength signals or channels simultaneously without interaction between each wavelength, known as wavelength division multiplexing (WDM) [14,16,18].

Also WDM optical switches are known to be reliable, support high speed, and are economical, which makes them an attractive selection for the default modern transport network. Moreover, new emerging services that require high bandwidth and reliability (such as Internet Protocol TV – IPTV [2]) are considering optical networks as the underlying network to directly carry the traffic. However, to best efficiently utilize WDM networks, network operators face a number of management and operation challenges, which often require complex mathematical models and advanced optimization techniques. This article focuses on these challenges and briefly review how integer linear programming (ILP) formulation and algorithms have been developed and applied to the domain of optical networks.

Motivations and Challenges in Optimization Models

The management and operation of WDM networks involve a number of challenges, which should address the physical topology formation, logical topology formation, survivability and fault management. Designing a new transport network is very complex, as it requires one to make decisions on where to place optical nodes so as to provide survivability, connectivity, and cost-effectiveness. Once the physical topology has been fixed, the logical topology of the backbone is decided by setting up lightpaths from one optical node to another. In transport networks, providing survivability and fault management is the most important task. Especially, routing should rapidly recover from any failure in the logical topology, because even a short outage reflects a massive amount of traffic loss in high-speed transport networks. The management and operation of these complex challenges benefit greatly from using mathematical modeling and optimization techniques.

Today's backbone mostly takes a form of a layered IP over optical network. For survivability, it is extremely important to address the practical issue of *how IP routing and protection schemes can effectively exploit the lower layer path diversity*. IP layer failures are known to occur most frequently, while fiber span failures are catastrophic in that they lead to multiple simultaneous upper layer failures. Moreover, some of the IP layer failures can be only addressed in the IP layer, as lower layer (optical) survivability mechanisms cannot detect failures occurring at higher (IP or applications) layers. In order to rapidly recover from network-wide failures and provide persistent end-to-end path quality, service providers may set up two diverse paths: the service (primary) path and the restoration (backup) path. Any failure in the service path can be hidden, as traffic can be instantly rerouted to the restoration path. Obviously, the efficacy of the restoration path depends heavily on how disjoint these two paths are (under the most frequent single failures). Therefore, it is important to understand how the layering employed in the network affects the correlation of failures among paths.

This demonstrates the importance of protection against failures in layered networks arising out of shared risk resource groups (SRRG). An example of SRRG is multiple IP links sharing a common optical component, including ducts or conduits through which multiple optical links are routed under the ground. To effectively deliver high-quality services to customers, network providers are required to incorporate this SRRG information into their routing and protection schemes, which has become one of the most challenging problems in networking practice. Note that the SRRG-diverse constraint involves multiple link failure models, in the form of shared risk link groups (SRLGs) and shared risk node groups (SRNGs). Most of previous studies in IP-over-WDM networks considered only two possible alternatives of routing and protection schemes: protection at the optical layer or restoration at the IP layer [7,11,15,16,17].

Path-Protection (Diverse) Routing Problem

Finding a backup path that is disjoint for each working or "primary" path, in general, has been recognized as *path-protection* schemes and has been widely studied in optical networks. Medard et al. [12] focused on the problem of identifying two redundant trees from a single source to a set of destinations that can survive any single link failure (i. e., the elimination of any vertex in the graph leaves each destination vertex connected to the source via at least one of the directed trees). Ellinas et al. [5] focused on the problem of identifying two diverse paths that are SRRG failure resilient. They were the first to theoretically prove that if an arbitrary set of links can belong to a common SRLG, then the problem of finding SRLG-diverse paths between a given source and destination is NP-complete for unicast traffic. Subsequently, Zang et al. [22] proposed heuristic algorithms for the combined problem of finding SRLG-diverse paths and wavelength assignment for one-to-one (unicast) traffic. Most recently, Cha et al. [2] studied the SRLG-diverse routing for one-to-many (multicast) traffic, where they focused on the combined problem of minimizing the network cost of multicasting traffic from dual sources to multiple destinations while providing path protection against a single SRLG failure.

Minimum Color Problem

Coudert et al. [4] proposed new techniques for the minimum color path problem for multiple failure tolerance from a SRRG failure. The consequence minimum color *st*-cut problem was also shown to be NP-complete and hard to approximate. Each SRRG is associated with a so-called color in a colored graph $G_c = (V, E, C)$, where C is a family of subsets of E. The minimum color path problem is to find a path from a node s to a node t that minimizes the number of different colors of its links. This problem was proven to bee NP-complete in [21] and polynomial in the special case where all the edges of each color have a common extremity. Many insightful theoretical results of this problem were reported in [4].

Definition 1 [4] Let $G = (V, E, C)$ be a colored graph, where C is a partition of E. The minimum color cut consists in finding a minimal set of colors disconnecting G. Let $s, t \in V$ be two distinct vertices in G. The minimum color *st*-cut problem is to find a minimal set of colors disconnecting s from t.

Theorem 1 [4] *The minimum color st-cut is NP-hard.*

Coudert et al. [4] proved this theorem by proposing the reduction of each set of the set cover instance to a color of the minimum *st*-cut instance.

Theorem 2 [4] *The minimum color st-cut problem is not approximable within a factor* $o(\log n)$ *unless* $NP \subseteq TIME(n^{O(\log \log n)})$.

Theorem 3 [4] *When the edges of each color induce a connected subgraph the minimum color st-cut and the minimum color cut problems are polynomial.*

Theorem 4 [4] *The minimum color st-cut is k-approximable when the number of connected components of the subgraph induced by edges of each color is bounded by k.*

Define a nonnegative variable for each node, where some edge e between nodes i and j institutes a cut if $x_i \neq x_j$. If any edge of color c institutes a cut, then c is a color cut. Let a binary variable y_c be associated with each color c, where $y_c = 1$ when the color c is selected to be in a set of color cut, and $y_c = 0$ otherwise. The minimum color *st*-cut problem can be formulated as a mixed integer linear program as follows:

$$\min \sum_{c \in C} y_c \tag{1}$$

$$\text{subject to} \quad y_c \geq |x_i - x_j| \forall c \in C\,, \forall i, j \in c\,, \tag{2}$$

$$x_i \geq 0 \quad \forall i \in V\,, \tag{3}$$

$$x_s = 0\,, x_t = 1\,. \tag{4}$$

Finding the Dual Link Problem

The problems related to finding a pair of link or node disjoint paths in single cost networks have been studied since the mid 1980s [19]. The min–sum problem of dual link is to minimize the sum of the costs of the two disjoint paths and can be solved using a polynomial time algorithm called the shortest pair of paths [19]. In a recent study, the min–sum problem was shown to be a special case of the min–cost flow problem [2]. In contrast to the min–sum problem, the min–max problem, whose objective is to minimize the length of the longer one of the two paths was proven to be NP-complete [10]. The min–min problem, whose objective is to minimize the length of the shorter one of the two paths, can also be proven to be NP-complete [20] by using the reduction of a the well-known 3-satisfiability (3SAT) problem. The proof can be described as follows [20]. An instance of 3SAT is a boolean formula that is the *AND* of m clauses $C_j (j = 1, \ldots, m)$. A clause is the *OR* of three literals, each of which is an occurrence of variable $x_i (i = 1, \ldots, n)$ or its negation. A truth assignment is a function $\tau : \{x_i\} \to \{\text{true}, \text{false}\}$. C_j is satisfied by τ if it contains a literal with truth value. The question of 3SAT is to determine whether there is a truth assignment that satisfies all m clauses simultaneously. With the 3SAT approach, Xu et al. [20] proposed the following theorem.

Theorem 5 [20] *The problem of finding two node/link-disjoint paths between a pair of source and destination nodes in a directed/undirected network with minimum cost for the shorter one is NP-complete.*

SRRG-Diverse Routing Problem

The SRRG-diverse routing problem can be considered to be a generalization of the *link-diverse/disjoint* routing problem. In a multicast context, the link-disjoint path-protection problem can be viewed as a diverse routing problem of identifying two redundant trees from a single source to a set of destinations that can survive any single link failure [6,12]. The diverse routing problem has been previously shown to be NP-complete [1,7,8]. During the past few years, there has been increasing interest in the diverse routing problem with SRRG-diverse constraints as SRRG-diversity requirements play a very crucial role in real life network provisioning problems. An example of real life problems is finding a pair of diverse paths at the optical layer, which involves the search of two SRLG-diverse paths as each link at the OXC (optical cross connect) layer may be related to several SRLGs. Many recent studies have shown that the generalized SRLG (or SRRG) diverse routing is a special case of the diverse routing problem, which is also NP-complete [5,8,11,13,22]. Among those studies, the diverse routing problem of unicast (one-to-one) traffic with SRLG-diverse constraints has been proven to be NP-complete [5]. In later studies, the diverse routing problem was extended to many special cases (e. g., diverse routing under both wavelength capacity and path length constraints [22], multicast routing under SRLG-diverse constraint [3]).

Cha et al. [3] proposed a generalized case of the SRRG-diverse routing problem where there are two source nodes. The problem can be formally defined as follows. Let $G = (V, E)$ be an undirected graph representing the backbone network. We denote the set of network nodes by V, while E is the set of duplex communications links (edges). Let the number of nodes and edges be n and m, respectively. There is a set of two source nodes, denoted by $S \subset V$, and there is a set of destination nodes, denoted by $D \subset V$. Denote B as a set of SRLGs. Each link $(i, j) \in E$ in the graph has a cost function (c_{ij}) associated with it and belongs to a subset of SRLGs in B. Note that the cost c_{ij} is the sum of the port cost at nodes i and j and the transport cost relative to the distance of link (i, j). This problem was proven to be NP-hard in [3] by using a reduction from the SRLG-diverse path problem [5].

Theorem 6 *The two-source SRRG-diverse routing problem is strongly NP-hard.*

This theorem was proven in [3] where this problem was claimed to be a generalization of the problem of finding SRLG-diverse paths between a source and a destination in a given graph (SRG (shared risk group)-diverse routing) proposed in the paper by Ellinas et al. [5]. We add

two nodes ($|S|$) in the graph and two links connecting each of the two new nodes with the source with the costs of 0. Assume that the two new links do not share any SRLGs. We then add $|D|$ nodes and $2 * |D|$ edges. Each of these nodes is connected to the destination node by two edges that do not share any SRLGs. Then the transformation is complete and clearly polynomial.

The ILP of the two-source SRRG-diverse routing problem can be formulated as follows. Define the following decision variables. $Y^s_{i,j} = 1$ *if* link (i, j) is used by the multicast tree rooted at source node s; 0 *otherwise*. $X^s_{i,j,d} = 1$ *if* link (i, j) is used by the multicast tree rooted at source node s to destination d; 0 *otherwise*. $Z^s_{b,d} = 1$ *if* the path from source s to destination d uses an SRLG b; 0 *otherwise*. The ILP formulation is given by

$$\min \sum_{s\in S} \sum_{(i,j)\in E} Y^s_{i,j} c_{i,j} \tag{5}$$

$$\text{subject to} \quad Y^s_{i,j} \geq X^s_{i,j,d} \quad \forall (i, j) \in E, \quad \forall s \in S, \forall d \in D, \tag{6}$$

$$\sum_{\{j|(i,j)\in E\}} X^s_{i,j,d} - \sum_{\{j|(j,i)\in E\}} X^s_{j,i,d} = \sigma^s_{i,d} \quad \forall i \in V, \forall s \in S, \forall d \in D, \tag{7}$$

$$Z^s_{b,d} \geq X^s_{i,j,d} \quad \forall (i, j) \in b, \forall s \in S, \forall d \in D, \quad \forall b \in B, \tag{8}$$

$$\sum_{s\in S} Z^s_{b,d} \leq 1 \quad \forall d \in D, \forall b \in B, \tag{9}$$

$$X^s_{i,j,d}, Y^s_{i,j}, Z^s_{b,d} \in \{0, 1\} \quad \forall s \in S, \forall d \in D, \quad \forall (i, j) \in E, \forall b \in B, \tag{10}$$

$$\text{where} \quad \sigma^s_{i,d} = \begin{cases} 1 & \text{if} \quad i = s, \\ -1 & \text{if} \quad i = d, \\ 0 & \text{otherwise}. \end{cases} \tag{11}$$

The constraints in (6) ensure that an edge must be selected by the multicast tree when it is used by the multicast tree to carry any traffic. The flow constraints in (7) ensure the flow conservation at each node, allowing each destination to have a flow path from the source. More precisely, $\sigma^s_{i,d}$ is the net flow capacity generated, carried, or destined at node i for destination d by the multicast tree rooted at node s, which should have the value of 1 if node i is the source, -1 if node i is the destination (acting as a sink), and 0 otherwise (whether node i belongs to the multicast tree or not). The constraints in (8) ensure that a SRLG must be selected by the path from a source to a destination when a link that belongs to the SRLG is used by the path. The constraints in (9) state that W^b_d is greater than or equal to the number of number of distinct sources that uses bundle b to reach d; that is, if b is used in only one or none of the sources to reach b, the value is greater than or equal to zero.

Shared Path-Protection Problem

Sahasrabuddhe et al. [17] proposed fault management in IP over WDM networks using techniques which are protection at the WDM layer and restoration at the IP layer. "Protection" refers to preprovisioned failure recovery (i. e., set up a backup lightpath for every primary lightpath), whereas "restoration" refers to more dynamic recovery (i. e., overprovision the network so that after a fiber failure, the network should still be able to carry all the traffic it was carrying before the fiber failure). Typically, their protection scheme focuses on shared path protection against single fiber span failures, where multiple independent primary paths share the backup path capacity to minimize the total capacity required in the network.

Given E as a set of unidirectional fiber links in the network, F as a set of bidirectional fibers in the network, R_{ij} as a set of alternate routes for node pair ij, and W as the maximum number of wavelengths on a link, the ILP of the shared path-protection routing problem can be formulated as follows. Define the following decision variables. w_k is the number of wavelengths used by primary lightpaths on link k, s_k is the number of spare wavelengths used on link k, and V_{ij} is the number of primary lightpaths between node pair ij. $m^w_k = 1$ *if* one or more backup lightpaths are using wavelength w on link k; 0 *otherwise*. $\gamma^w_{ij,r} = 1$ *if* the rth route between node pair ij utilizes wavelength w before any fiber failures; 0 *otherwise*. $\delta^{b,w}_{ij,p} = 1$ *if* a primary on route between node pair ij is protected by route between the node pair by employing wavelength; 0 *otherwise*. The ILP of the shared path-protection routing problem is rather complicated as it considers end-to-end lightpath assignment on the physical links, physical diversity of the primary and backup paths, and sharing backup path

capacity for failure-independent primary paths. Therefore, the key ILP is formulated as follows (please refer to [17] for the complete set of equations):

$$\min \sum_{k=1}^{E} (w_k + s_k) \tag{12}$$

$$\text{subject to} \sum_{ij} \sum_{p \in R_{ij}: f \in p} \sum_{b \in R_{ij}: k \in b} \delta_{ij,p}^{b,w} \leq 1 \tag{13}$$

$$1 \leq f \leq F, 1 \leq k \leq E, 1 \leq w \leq W\,,$$

$$w_k + s_k \leq W \quad 1 \leq k \leq E\,, \tag{14}$$

$$\sum_{r \in R_{ij}} \sum_{w=1}^{W} \gamma_{ij,r}^{w} = V_{ij} \quad \forall ij\,, \tag{15}$$

$$\sum_{ij} \sum_{r \in R_{ij}, k \in r} \sum_{w=1}^{W} \gamma_{ij,r}^{w} = w_k \quad 1 \leq k \leq E\,, \tag{16}$$

$$\sum_{w=1}^{W} m_k^w = s_k \quad 1 \leq k \leq E\,, \tag{17}$$

$$\left(\sum_{ij} \sum_{r \in R_{ij}, k \in r} \gamma_{ij,r}^{w} \right) + m_k^w \leq 1 \quad 1 \leq k \leq E,$$

$$1 \leq w \leq W\,, \tag{18}$$

$$\sum_{w=1}^{W} \gamma_{ij,r}^{w} = \sum_{b \in R_{ij}, b \neq p} \sum_{w=1}^{W} \delta_{ij,p}^{b,w} \quad \forall ij, \forall p \in R_{ij},$$

$$1 \leq w \leq W\,. \tag{19}$$

The objective of ILP is to minimize the total capacity used given in (12). The crux of the formulation is at the set of constraints in (13), which ensure that that two backup lightpaths share wavelength w on link k *only if* the corresponding primary paths are fiber-disjoint. The above ILP formulation includes constraints for setting up lightpaths with shared path protection, where the number of channels on each link is bounded by (14), the number of primary lightpaths between a node pair is defined by (15), the number of primary lightpaths traversing a link is defined by (16), the spare capacity of each link is defined by (17), the usage of primary or backup lightpaths on a wavelength is defined by (18), and every primary lightpath is ensured to be protected by a backup lightpath by (19). In addition, multicommodity flow constraints (omitted here) are added to ensure the amount of traffic sourced from node to destination node is covered by the capacity of the wavelength. Note that additional constraints on the number of receivers and transmitters used can be incorporated in the model [17].

Path-Restoration Problem

Path-restoration techniques have been frequently employed to provide highly capacity efficient (close to) real-time restoration of a network failure. In contrast to the path-protection routing, the operation mode of the path restoration only uses the full bandwidth from the primary while finding an alternative path. The path-restoration routing problem can involve different network layers: physical (optical) and IP. The optical path restoration directly replaces the prefailure capacity at the transmission carrier signal level, which has no performance effects in the upper layers. On the other hand, the IP path restoration dynamically reroutes the signals around failures using routing table updates or dynamic call-routing. An interior gateway protocol (IGP) is widely used to dynamically find an alternative path and perform load sharing in traffic distribution.

Optical Path Restoration Problem

Iraschko and Grover [9] studied the path-restoration routing problem, where the task is to deploy a set of replacement signal paths between two end nodes of the failed span, capable of yielding the maximum total amount of replacement capacity, while respecting the finite number of spare links on each span. They assume that, given failure scenarios, a predefined set of distinct eligible routes are precomputed for end node pairs (i. e., primary paths). Then, the goal of the path-restoration routing problem is to maximize the total of all restoration flow assignments for those primary paths, using only the commodities selected by the failure scenario and only the surviving spans in the reserve network. It also requires that all flow assignments made over all routes, for all simultaneously restored node pairs, should not exceed the spare capacity of any span in the reserve network. The outcome of such optimized design will require minimal capacity for the reserve network.

The ILP of the path-restoration routing problem can be formulated as follows. Define the following decision variables and parameters. Let i represent a failure scenario, such as a single span cut or a node loss. D_i

is the number of end node pairs that have lost one or more units of demand owing to the failure i and r is the index to indicate these node pairs, $r \in (1 \cdots D_i)$. X_i^r is the number of demand units lost by an end pair r under failure i. Finally, the network is $G(N, E, s)$, where N is the set of nodes, E is the set of spans, and s is the vector of spare capacities on each span s_j. The ILP formulation is given by

$$\max \sum_{r=1}^{D_j} \sum_{p=1}^{P_i^r} f_i^{r,p} \tag{20}$$

$$\text{subject to} \quad \sum_{p=1}^{P_i^r} f_i^{r,p} = X_i^r \quad \forall r \in (1 \cdots D_i)\,, \tag{21}$$

$$\sum_{r=1}^{D_j} \sum_{p=1}^{P_i^r} \delta_{i,j}^{r,p} f_i^{r,p} \le s_j \quad \forall j \in E\,, \tag{22}$$

$$f_i^{r,p} \ge 0, \quad \text{integer} \quad \forall (r, p)\,, \tag{23}$$

where $f_i^{r,p}$ is a whole number assignment of flow to the pth route available for restoration of node pair r under failure scenario i. P_i^r is the total number of eligible restoration routes available to node pair r for the restoration of failure i. $\delta_{i,j}^{r,p} = 1$ *if* span j is in the pth eligible route for restoration of node pair r in the event of failure scenario i; 0 *otherwise*.

IP Path–Restoration Problem

In the IP-layer communication, as apposed to the abovementioned optical path restoration problem, the path-restoration problem is defined differently on the basis of the WDM protection model from Sect. "Shared Path-Protection Problem" (for the complete model, see [17]). The key ILP model for this IP path restoration problem is given by

$$\max \sum_{sd} \sum_{ij} \lambda_{sd}^{ij} \tag{24}$$

$$\text{subject to} \quad \sum_{j} \sum_{r \in R_{ij}} \gamma_{i,j,r}^{w} \le Trans_i^w \quad \forall i, 1 \le w \le W\,, \tag{25}$$

$$\sum_{i} \sum_{r \in R_{ij}} \gamma_{i,j,r}^{w} \le Rec_j^w \quad \forall j, 1 \le w \le W\,, \tag{26}$$

$$\sum_{ij} \sum_{r \in R_{ij}, k \in r} \gamma_{ij,r}^{w} \le 1 \quad 1 \le k \le E, 1 \le w \le W\,, \tag{27}$$

where the objective function in (24) is to minimize the average hop distance before a fault, the constraints in (25) and (26) ensure that node i uses at most $Trans_i^w$ transmitters and node j uses at most Rec_j^w receivers on wavelength w, and the constraints in (27) ensure that the wavelength w on link j is used either by a primary lightpath or by backup lightpaths.

Concluding Remarks

In this article, we reviewed how ILP formulations are used in WDM optical networking. In optical networks, preprovisioning the networks to support fast restoration of failures is critical, which means to set up physically disjoint backup paths (links) for the primary paths (links). Here, ILP formulations are valuable in finding the global optimal backup paths among all the possible alternative paths for traffic demands of interest. A set of constraints in ILP can be set up to represent the restoration flow balance constraint, the link capacity flow constraint, and physical diversity of the primary and backup paths.

References

1. Bhandari R (1999) Survivable Networks: Algorithms for Diverse Routing, Klumer, Norwell
2. Cha M, Chaovalitwongse W, Art, Ge Z, Yates J, Moon S (2006) Path Protection Routing with Constraints SRLG to Support IPTV in Mesh WDM Networks. IEEE Global Internet Symposium
3. Cha M, Choudhury G, Yates J, Shaikh A, Moon S (2006) Case Study: Resilient Backbone Network Design for IPTV Services: WWW IPTV Workshop
4. Coudert D, Datta P, Rivano H, Voge M-E (2005) Minimum Color Problems and Shared Risk Resource Group in Multilayer Networks, Research Report RR-2005–37-FR, I3S
5. Ellinas G, Bouillet E, Ramamurthy R, Labourdette J-F, Chaudhuri S, Bala K (2003) Routing and Restoration Architectures in Mesh Optical Networks. Opt Netw Mag 4(1):91–106
6. Fei A, Cui J, Gerla M, Cavendish D (2001) A "Dual-Tree" Scheme for Fault-tolerant Multicast. IEEE ICC, June 2001
7. Grover, Wayne D (2003) Mesh-Based Survivable Networks. Prentice Hall, Upper Saddle River
8. Hu JQ (2003) Diverse Routing in Optical Mesh Networks. IEEE/ACM ToN 51(3):489–494
9. Iraschko R, Grover W (2002) A Highly Efficient Path-Restoration Protocol for Management of Optical Network Transport Integrity. IEEE JSAC 18(5):779–793
10. Li C, McCormick ST, Simchi-Levi D (1992) Finding Disjoint Paths with Different Path Costs: Complexity and Algorithms. Network 22:653–667

11. Li G, Kalmanek C, Doverspike R (2002) Fiber Span Failure Protection in Mesh Optical Networks. Opt Netw Mag 3(3):21–31
12. Medard M, Finn SG, Barry RA (1999) Redundant Trees for Preplanned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graphs. IEEE/ACM ToN 7(5):641–652
13. Modiano E, Narula-Tam A (2002) Survivable Lightpath Routing: A New Approach to the Design of WDM-Based Networks. IEEE JSAC 20(4):800–809
14. Murthy C, Siva R, Gurusamy M (2001) WDM Optical Networks. Prentice Hall, Englewood Cliffs
15. Ramamurthy S, Sahasrabuddhe L, Mukherjee B (2003) Survivable WDM Mesh Networks. J Lightwave Tech 21(11): 870–883
16. Ramaswami R, Sivarajan K (1998) Optical Networks: A practical Perspective. Morgan Kaufmann, Los Altos
17. Sahasrabuddhe L, Ramamurthy S, Mukherjee B (2002) Fault Management in IP-Over-WDM Networks: WDM Protection Versus IP Restoration. IEEE JSAC 20(1):21–33
18. Stern TE, Bala K (1999) Multiwavelength Optical Networks: A Layered Approach. Addison-Wesley, Boston
19. Suurballe J, Tarjan R (1984) A Quick Method for Finding Shortest Pairs of Disjoint Paths. Network 14:325–336
20. Xu D, Chen Y, Xiong Y, Qiao C, He X (2006) On the Complexity of and Algorithms for Finding the Shortest Path with a Disjoint Counterpart. IEEE/ACM ToN 14(1):147–158
21. Xu D, Xiong Y, Qiao C (2003) Protection with Multi-Segments (PROMISE) in Networks with Shared Risk Link Group (SRLG). IEEE/ACM ToN 11(2):248–258
22. Zang H, Ou C, Mukherjee B (2003) Path-Protection Routing and Wavelength Assignment (RWA) in WDM Mesh Networks Under Duct-Layer Constraints. IEEE/ACM ToN 11(2):248–258

Integer Programming

EGON BALAS
Graduate School Industrial Admin.,
Carnegie Mellon University, Pittsburgh, USA

MSC2000: 90C10, 90C11, 90C27, 90C57

Article Outline

Keywords
Scope and Applicability
Combinatorial Optimization
Solution Methods
The State of the Art
See also
References

Keywords

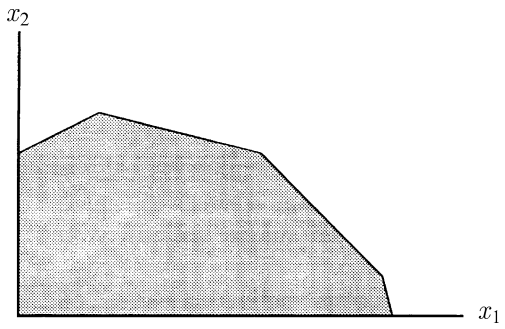
Integer programming; Combinatorial optimization; Enumeration; Polyhedral methods; Disjunctive programming; Lift-and-project

One of the byproducts of World War II was the discovery that the routing of ship convoys, and other human activities like transportation, production, allocation, etc. could be modeled mathematically, i. e. the often intricate choices that they involve could be captured into a system of equations and inequalities and could be optimized according to some agreed upon criterion. The simplest such model, involving only linear functions, became known as linear programming. Parallel developments have led to the discovery of the computer, which made it practical to solve linear programs of a realistic size. A few years later the theory was extended to systems involving nonlinear convex functions. Convexity was needed to ensure that any local optimum is a global optimum.

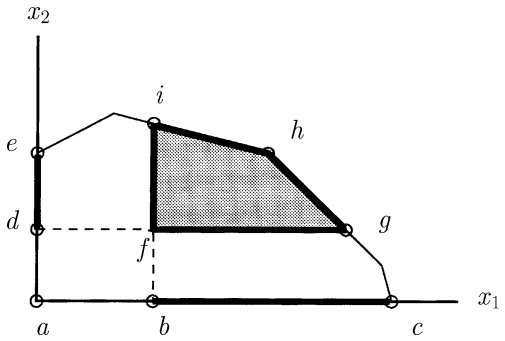
An amazing variety of activities and situations can be adequately modeled as linear programs (LPs) or convex nonlinear programs (NLPs). Adequately means that the degree of approximation to reality that such a representation involves is acceptable. However, as the world that we inhabit is neither linear nor convex, the most common obstacle to the acceptability of these models is their inability to represent nonconvexities or discontinuities of different sorts. This is where integer programming comes into play: it is a universal tool for modeling nonconvexities of various kinds.

To illustrate, imagine a factory that produces two items and whose capacity is determined by the four linear constraints represented in Fig. 1. These inequalities, along with $x_1 \geq 0$, $x_2 \geq 0$ (only nonnegative amounts can be produced), define the feasible set, shown as the shaded area. Since the latter is a convex polyhedron, if profit is a linear function of the amounts produced, then an optimal (i. e. profit-maximizing) production plan will correspond to one of the vertices of the polyhedron.

Imagine now that the following reasonable condition is imposed: for each item there is a threshold quantity below which it is not worth producing it. The threshold is b units for item 1 and d units for item 2. Furthermore, at least one of the two items must be produced. As shown in Fig. 2., the feasible set now consists



Integer Programming, Figure 1
Feasible set without tresholds



Integer Programming, Figure 2
Feasible set with tresholds

of three separate pieces: the two lines $[b, c]$ and $[d, e]$, and the shaded area (polyhedron) whose vertices are f, g, h, i. Depending on the slope of the objective function, the optimum may now occur at any of the points b, c, d, e, g, h, i. The problem has become qualitatively different. The 'threshold' conditions, of the form 'either $x_1 = 0$ or else $x_1 \geq b$', and 'either $x_2 = 0$ or $x_2 \geq d$', as well as the condition 'at most one of $x_1 = 0$ and $x_2 = 0$ can hold', cannot be modeled by linear or convex programming techniques. The same is true of a host of other 'logical' conditions: disjunctions ('either this or that'; 'at least one of several constraints must hold', 'at most one of several variables can be positive'), implications ('if this action is taken, then that action must be taken'), precedence relations ('this event must precede that', 'this action cannot start until some others are completed'), etc. Yet, it is quite obvious that conditions of this type are in no way exceptional; on the contrary, their presence is pervasive in many real world situations.

A linear (nonlinear) programming problem whose variables are restricted to integer values is called a linear (nonlinear) integer programming problem, or simply an *integer program* (IP, linear unless otherwise stated). If only some of the variables are restricted to integer values, we have a *mixed integer program* (MIP). Such a problem can be stated as

$$\begin{cases} \min & cx \\ \text{s.t.} & Ax \geq b \\ & x \geq 0 \\ & x_j \text{ integer} \\ & j \in N_1 \subseteq N, \end{cases}$$

where A is a given $m \times n$ matrix, c and b are given vectors of conformable dimensions, $N := \{1, \ldots, n\}$ and x is a variable n-vector. The 'pure' integer program (IP) is the special case of MIP when $N_1 = N$. If, in addition, all entries of A, b, c are integer, then the slack or surplus variables can also be restricted to integers.

Integer programming as a field started in the mid-1950s. A number of excellent textbooks are available for its study [15,16,17,20].

Scope and Applicability

Applications of integer programming abound in all spheres of decision making. Some typical real-world problem areas where integer programming is particularly useful as a modeling tool, include facility (plant, warehouse, hospital, fire station) location; scheduling (of personnel, production, other activities); routing (of trucks, tankers, aircraft); design of communication (road, pipeline, telephone, computer) networks; capital budgeting; project selection; analysis of capital development alternatives. Various problems in science (physics: the Ising spin glass problem; genetics: the sequencing of DNA segments) and medicine (optimizing tumor radiation patterns) have been successfully modeled as integer programs. In engineering (electrical, chemical, civil and mechanical) the sphere of applications is growing steadily.

By far the most important special case of integer programming is the (pure or mixed) 0–1 *programming problem*, in which the integer-constrained variables are restricted to 0 or 1. This is so because a host of frequently occurring nonconvexities, such as the ones listed above, can be formulated via 0–1 variables. If the constraint set of the production planning problem

shown in Fig. 1. is

$$a_{i1}x_1 + a_{i2}x_2 \le b_i, \quad i = 1, \ldots, 4,$$
$$x_1 \ge 0, \quad x_2 \ge 0,$$

then the conditions

$$x_1 \le 0 \quad \text{or} \quad x_1 \ge b,$$
$$x_2 \le 0 \quad \text{or} \quad x_2 \ge d,$$
$$x_1 > 0 \quad \text{or} \quad x_2 > 0,$$

imposed in the variant shown in Fig. 2., can be formulated by introducing two 0–1 variables, δ_1 and δ_2, and the constraints

$$b\delta_1 \le x_1 \le c\delta_1,$$
$$d\delta_2 \le x_2 \le e\delta_2,$$
$$\delta_1 + \delta_2 \ge 1, \quad \delta_1, \delta_2 \in \{0, 1\}.$$

Next we present a few well-known pure and mixed integer models.

The *fixed charge problem* asks for the minimization, subject to linear constraints, of a function of the form $\sum_i c_i(x_i)$, with

$$c_i(x_i) := \begin{cases} f_i + c_i x_i & \text{if } x_i > 0, \\ 0 & \text{if } x_i = 0. \end{cases}$$

Whenever x_i is bounded by U_i and $f_i > 0$ for all i, such a problem can be restated as a (linear) MIP by setting

$$c_i(x_i) = c_i x_i + f_i y_i,$$
$$x_i \le U_i y_i,$$
$$y_i \in \{0, 1\} \text{ for all } i.$$

Clearly, when $x_i > 0$ then y_i is forced to 1, and when $x_i = 0$ the minimization of the objective function drives y_i to 0.

The *facility location problem* consists of choosing among m potential sites (and associated capacities) of facilities to serve n clients at a minimum total cost:

$$\begin{cases} \min & \displaystyle\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} f_i y_i \\ & \displaystyle\sum_{i=1}^{m} x_{ij} = d_j, \quad j = 1, \ldots, n, \\ & \displaystyle\sum_{j=1}^{n} x_{ij} \le a_i y_i, \quad i = 1, \ldots, m, \\ & x_{ij} \ge 0, \quad i = 1, \ldots, m; j = 1, \ldots, n, \\ & y_i \in \{0, 1\}, \quad i = 1, \ldots, m \end{cases}$$

Here d_j is the demand of client j, a_i is the capacity of a potential facility to be located at site i, c_{ij} is the per-unit cost of shipments from facility i to client j, and f_i is the fixed cost of opening a facility of capacity a_i at location i. In any feasible solution, the indices i such that $y_i = 1$ designate the chosen locations for the facilities to be opened.

Variants of this problem include the *uncapacitated facility location problem* (where $d_j = 1$ for all j and the constraints involving the capacities can be replaced by $x_{ij} \le y_i$, $i = 1, \ldots, m$, $j = 1, \ldots, n$), the *warehouse location problem* (which considers cheap bulk shipments from plants to warehouses and expensive packaged shipments to retailers), and various *emergency facility location* problems (where one chooses locations to minimize the maximum distance traveled by any user of a facility, rather than the sum of travel costs).

The *knapsack problem* is an integer program with a single constraint:

$$\max\{cx : \ ax \le b, \ x \ge 0 \text{ integer}\},$$

where c and a are positive n-vectors, while b is a positive scalar. When the variables are restricted to 0 or 1, we have the *0–1 knapsack problem*.

A variety of situations can be fruitfully modeled as *set covering problems*: Given a set M and a family of weighted subsets $S_1, \ldots, S_n$ of M, find a minimum-weight collection C of subsets whose union is M. If A is a 0– 1 matrix whose rows correspond to the elements of M and whose columns are the incidence vectors of the subsets $S_1, \ldots, S_n$, and c is the n-vector of subset-weights, the problem can be stated as

$$\begin{cases} \min & cx \\ & Ax \ge 1 \\ & x \in \{0, 1\}^n, \end{cases}$$

where the right-hand side of the inequality is the m-vector of 1s. This model and its close relative, the *set partitioning problem* (in which $\ge$ is replaced by $=$) has been (and is being) widely used in airline, bus, and train crew scheduling (each row represents a leg of a trip that has to be covered; each column stands for a potential duty period of a crew). Another application is in medical diagnostics (each column represents a diagnostic test, each row stands for a pair of diseases, with a 1 in column j if the pair's reactions to the tests are different, and a 0 if they are the same; the goal being to select

a minimum-cost battery of tests guaranteed not to yield identical outcomes for any two diseases).

Combinatorial Optimization

A host of interesting combinatorial problems can be formulated as 0–1 programming problems defined on graphs, undirected or directed, vertex-weighted or edge-weighted. The joint study of these problems by mathematical programmers and computer scientists, starting from around 1960, has led to the development of the burgeoning field called *combinatorial optimization* [7]. Some typical problems of this field are: edge matching (finding a maximum-weight collection of pairwise non-adjacent edges) and edge covering (finding a minimum-weight collection of edges that together cover every vertex); vertex packing (finding a maximum-weight independent set, i. e. collection of pairwise non-adjacent vertices) and vertex covering (finding a minimum-weight collection of vertices that together cover every edge); maximum clique (finding a maximum cardinality complete subgraph) and minimum vertex coloring (partitioning the vertices into a minimum number of independent sets, i. e. coloring the vertices with a minimum number of colors such that all adjacent pairs differ in color); the traveling salesman problem (finding a cycle of minimum total edge-weight that meets every vertex).

We will briefly discuss two of the above problems, which in a sense span the universe of combinatorial optimization. At one end of the spectrum, the *matching problem* on a graph $G = (V, E)$ can be stated as

$$\begin{cases} \max & x(E) \\ \text{s.t.} & x(\delta(v)) \le 1, \quad v \in V, \\ & x \ge 0, \quad x_e \text{ integer}, \quad e \in E, \end{cases}$$

where for $F \subset E$, $x(F) = \sum_{e \in F} x_e$, and $\delta(v)$ is the set of edges incident with v. Its weighted version asks for maximizing $\sum_{e \in E} w_e x_e$, where w_e is the weight of edge e. This problem has the nice property that the integrality condition can be omitted if the above nonnegativity and degree constraints are supplemented with the inequalities

$$x(\gamma(S)) \le \frac{|S| - 1}{2} \quad \text{for all } S \subseteq V,\ |S| \text{ odd},$$

where $\gamma(S)$ is the set of edges with both ends in S.

In other words, the 'odd set inequalities', along with the nonnegativity and degree constraints, fully describe the convex hull of incidence vectors of matchings. The discovery of this remarkable phenomenon in the mid-1960s ([8]) has started a massive pursuit of facets of the convex hull of other combinatorial polyhedra, and can be viewed as the inaugural step in the development of the field called *polyhedral combinatorics*. Close relatives of the matching problem are the *perfect matching problem* (in which a maximum or minimum-weight matching is sought, when it exists, that leaves no vertex unmatched), the *2-matching problem* (in which the degree constraints have right-hand side 2) and more generally, the *b-matching*, or *degree-constrained subgraph problem* (in which the degree constraint for vertex v has the positive integer b_v as right-hand side). In each of these cases, a complete description of the convex hull of feasible integer points is available in the form of a class of inequalities similar to the above ones, which makes these problems polynomially solvable.

At the other end of the spectrum, one of the hardest and most thoroughly investigated combinatorial optimization problems is the *traveling salesman problem* (TSP) already mentioned, in which a salesman is looking for a cheapest tour of n cities, given the cost of travel between all pairs of cities. This is the prototype model for situations dealing with the optimal sequencing of objects (e. g., items to be processed on a machine in the presence of sequence-dependent setup costs). The standard formulation on a complete directed graph with node set N and arc costs c_{ij} is

$$\begin{cases} \min & \displaystyle\sum_{i \in N} \sum_{j \in N \setminus \{i\}} c_{ij} x_{ij} \\ \text{s.t.} & \displaystyle\sum_{j \in N \setminus \{i\}} x_{ij} = 1, \quad i \in N, \\ & \displaystyle\sum_{i \in N \setminus \{j\}} x_{ij} = 1, \quad j \in N, \\ & \displaystyle\sum_{i \in S} \sum_{j \in S \setminus \{i\}} x_{ij} \le |S| - 1 \\ \text{for} & S \subset N,\ 2 \le |S| \le n - 1, \\ & x_{ij} \in \{0, 1\}, \quad i, j \in N, \quad i \ne j. \end{cases}$$

The first two sets of equations define an assignment problem whose solutions are spanning unions of directed cycles. The third set, consisting of inequalities called subtour elimination constraints, exclude all cy-

cles with fewer than $n = |N|$ arcs. The number of solutions – tours – is factorial in the number of nodes. Problems with random costs can be solved even for thousands of nodes, but some cost structures that occur in practice tend to give rise to very hard problems even at small sizes: a class of machine scheduling problems at chemical plants turn out to be hard to solve as TSPs on 30–50 node directed graphs. The TSP has become a test bed for the development of, and experimentation with, various approaches to combinatorial optimization. A summary of results until the mid- 1980s is to be found in [12].

A generalization of the TSP, in which the salesman does not have to visit all the cities, but gets a prize for every city that he does visit, is called the *prize collecting traveling salesman problem* (PCTSP). It asks for a cheapest tour of just enough cities to collect a required amount of prize money. This is the model used for scheduling the daily 'rounds' of a steel rolling mill, an operation that combines the tasks of selecting the items for the next round with that of putting them in the proper sequence. On a directed graph with loops, it can be formulated as the problem of finding a directed cycle of minimum arc cost subject to an upper bound on the sum of loop penalties on nodes not included in the cycle:

$$
\begin{cases}
\min & \displaystyle\sum_{i\in N}\sum_{j\in N\setminus\{i\}} c_{ij}x_{ij} \\
\text{s.t} & \displaystyle\sum_{j\in N\setminus\{i\}} x_{ij} + y_i = 1, \quad i \in N \\
& \displaystyle\sum_{i\in N\setminus\{j\}} x_{ij} + y_j = 1, \quad j \in N \\
& \displaystyle\sum_{i\in S}\sum_{j\in S\setminus\{i\}} x_{ij} + \sum_{i\in S\setminus\{k\}} y_i - y_\ell \le |S| - 1 \\
\text{for} & S \subset N,\ 2 \le |S| \le n-1, \\
& k \in S,\ \ell \in N \setminus S, \\
& \displaystyle\sum_{i\in N} w_i y_i \le U, \\
& x_{ij} \in \{0,1\}, \quad y_i \in \{0,1\}, \quad \forall i,j.
\end{cases}
$$

The first two sets of equations are satisfied by any spanning union of cycles and loops, while the third set of constraints excludes multiple cycles. Finally, the last inequality bounds at U the weighted sum of loop variables.

Solution Methods

Unlike linear programs, which are polynomially solvable, integer programming problems, including 0–1 programming and most combinatorial optimization problems, are notoriously difficult: in the language of computational complexity theory, they are *NP*-complete. Polynomial time integer programming algorithms do not exist. However, sometimes an integer program can be solved as a linear program, in the sense that solving the linear programming relaxation (L) of the integer program (i. e. the problem obtained by removing the integrality conditions), one obtains an integer solution. In particular, this is the case when all the basic solutions of (L) are integer. For an arbitrary integer vector b, the constraint set $Ax \ge b$, $x \ge 0$, if feasible, is known [10] to have only integer basic solutions if and only if the matrix A is totally unimodular (i. e. all square submatrices of A have a determinant equal to 0, 1 or -1).

The best-known instances of total unimodularity are the vertex-edge incidence matrices of directed graphs, and of undirected bipartite graphs. As a consequence, shortest path and network flow problems on arbitrary directed graphs, as well as edge matching (or covering) and vertex packing (or covering) problems on undirected bipartite graphs, are in fact linear programs, as are all those integer programs whose LP relaxation has as its coefficient matrix the incidence matrix of a directed graph, or that of an undirected bipartite graph, with arbitrary integer right-hand sides.

Apart from this important but very special class of problems, the difficulty in solving integer programs, as already mentioned, lies in the nonconvexity of the feasible set, which makes it impossible to establish global optimality from local conditions. The two principal approaches to solving integer programs try to circumvent this difficulty in two different ways.

The first approach, which until the late 1980s was the standard way of solving integer programs, is *enumerative* (branch and bound, implicit enumeration). It partitions the feasible set into successively smaller subsets tied together as nodes of a branch and bound tree, calculates bounds on the objective function value over each subset, and uses these bounds to discard certain subsets (nodes) from further consideration. The lower bounds (in a minimization problem) typically come

from solving the linear programming relaxation corresponding to the given node, the upper bounds come from integer solutions found at some of the nodes. The procedure ends when each subset has either produced a feasible solution, or was shown to contain no better solution than the one in hand. The efficiency of the procedure depends crucially on the strength of the bounds. Two early prototypes of this approach are due to A.H. Land and A.G. Doig [11] and E. Balas [1].

The second approach, known as the *cutting plane method*, is a convexification procedure: it approximates the convex hull of the set of feasible integer points by a sequence of inequalities that cut off (hence the term 'cutting planes') part of the linear programming polyhedron, without removing any feasible integer point. The first finitely convergent procedure of this type, which uses modular arithmetic to derive valid cutting planes for pure integer programs, is due to R.E. Gomory [9]. V. Chvátal [6] has shown that the procedure can be viewed as one of *integer rounding*, in which positive multiples of $Ax \geq b$, $x \geq 0$ are added up and the coefficients of the resulting inequality are rounded down to the nearest integer. The resulting inequalities form the elementary closure of $Ax \geq b$, $x \geq 0$. The procedure can then be applied to the elementary closure, and so on. The number of times the procedure needs to be iterated in order to obtain the convex hull of feasible integer points is called the Chvátal rank of the given polyhedron. No bound is known on the Chvátal rank of an arbitrary integer polyhedron (convex hull of feasible integer points). By contrast, the matching polyhedron has Chvátal rank one, since the odd set inequalities can be obtained from the degree inequalities by integer rounding.

The Gomory–Chvátal procedure has been extended to mixed integer programming and has been enhanced by the use of subadditive functions and group theory.

A different approach comes from *disjunctive programming* [2,3], or linear programming with logical conditions (conjunctions, disjunctions and implications involving inequalities). In this approach, which uses the tools of convex analysis, like polarity and projection, 0–1 programming (pure or mixed) is viewed as optimization over the (nonconvex) union of (convex) polyhedra, i. e. a set of the form $\cup_{i \in Q} P_i$, where $P_i = \{x: A^i x \geq b^i\}$, $i \in Q$. There is a compact characterization of the convex hull $P_D := \text{conv} \cup_{i \in Q} P_i$ in a higher-dimensional space, whose projection onto the original space yields all the valid cutting planes. Thus $\alpha x \geq \beta$ is a valid inequality for $\cup_{i \in Q} P_i$ if and only if $\alpha \geq u^i A^i$ and $\beta \leq u^i b^i$ for some $u^i \geq 0$, $i \in Q$. A central result of this approach is that an important class of disjunctive programs, called *facial*, which includes pure and mixed 0–1 programs, are *sequentially convexifiable*. For a 0–1 program (pure or mixed) with n 0–1 variables and a linear programming relaxation P_0, this means that one can impose the 0–1 condition on x_1 and generate the convex hull P_1 of $P_{00} \cup P_{01}$, where $P_{00} := \{x \in P_0: x_1 = 0\}$, $P_{01} := \{x \in P_0: x_1 = 1\}$; then impose the 0–1 condition on x_2 and generate the convex hull P_2 of $P_{10} \cup P_{11}$, where $P_{10} := \{x \in P_1: x_2 = 0\}$, $P_{11} := \{x \in P_1: x_2 = 1\}$; etc., and at the end of n steps, the convex hull P_n of $P_{n-1,0} \cup P_{n-1,1}$ turns out to be the convex hull of $\{x \in P_0: x_j \in \{0, 1\}, j = 1, \ldots, \}$. This property does not hold for arbitrary integer programs, and is thus a main distinguishing feature of 0–1 programs. If one defines the disjunctive rank of a polyhedron as the number of times the above procedure has to be iterated in order to generate all of its facets, it follows that an arbitrary 0–1 programming polyhedron has disjunctive rank n.

Although these results date back to 1974, it was not until the early 1990s that they were implemented into an efficient computational tool called *lift-and-project* ([4]). The name conveys the idea of a higher-dimensional representation of the convex hull (lifting), which is then projected back to generate cutting planes. In the meantime, L. Lovász and A. Schrijver [13] (see also [18]) developed a closely related procedure which derives higher-dimensional representations of a 0–1 programming polyhedron by multiplying the constraint set of P_0 with the inequalities $x_j \geq 0$ and $1 - x_j \geq 0$, $j \in N$, then linearizing the resulting quadratic forms, and projecting them back into the original space. As in the disjunctive programming approach, n iterations of this procedure yield the convex hull of the 0– 1 programming polyhedron. However, the quadratic forms obtained during the procedure can also be used to derive *positive semidefiniteness constraints* that are stronger than the inequalities obtained by linearization.

Semidefiniteness constraints aside, a streamlined version of the Lovász–Schrijver procedure, in which P_0 is multiplied at every iteration by just one pair of inequalities $x_j \geq 0$, $1 - x_j \geq 0$, rather than by all pairs, was shown in [4] to be equivalent to the disjunctive

programming procedure for 0–1 polyhedra, in that the linearized version of the quadratic constraints obtained by multiplication is exactly the same as the higher-dimensional representation of the convex hull used in disjunctive programming. The paper [4] also showed how to use a cut generating linear program (CGLP) to obtain lift-and-project (or disjunctive) cuts that are deepest in a well defined sense. Most importantly, these cuts can be generated in a subspace, i. e. using only a subset of the variables, and then lifted to the full space. It was this aspect which has led to a computational breakthrough. Earlier attempts to implement cutting plane procedures of whatever type for general integer or 0–1 programs foundered on the phenomenon known as 'stalling': in the process of generating a sequence of cutting planes and reoptimizing the linear program, after a while the new cuts tended to become shallower and the process tended to run into numerical difficulties. Now the possibility of lifting cuts generated in a subspace has opened the door to combining the enumerative and convexifying approaches into a *branch and cut procedure*, which generates cutting planes as long as they 'work', but branches whenever the cut generating 'stalls'. This was made possible by the fact that cuts generated at a node of the search tree can be lifted to be valid at any node. The outcome was a robust procedure, considerably more efficient than either a branch and bound or a cutting plane algorithm by itself [5].

Besides these two basic approaches (enumerative and convexifying), two further procedures need to be mentioned that do not belong to either category, but can be combined with either of them. Both procedures essentially *decompose* the problem, one of them by partitioning the variables, the other by partitioning the constraints. The first one, known as *column generation*, starts with a subset of the columns and generates the missing columns as needed, by pricing them out. It is an extension to integer programming of well-known linear programming decomposition techniques. The second one, known as *Lagrangian relaxation*, works with a subset of the constraints, while assigning Lagrange multipliers to the remaining constraints and taking them into the objective function.

Each of the approaches outlined above aims at solving the integer program exactly. However, due to the *NP*-completeness of the problem, approximation methods and *heuristics* play an increasingly important role in this field. Some highly efficient heuristics are known for several special structures. As to the general problem, heuristics have so far been less uniformly successful.

The State of the Art

Until about 1985, most of the integer programming problems encountered in practice were too large to be solved in useful time by existing algorithms and codes. This situation has drastically changed during the decade of the 1990s. MIPLIB is a collection of integer programming problems that various people have tried to solve over the last two decades or so, often unsuccessfully. It contains scores of instances varying in structure, size and computational toughness. A few years ago even those instances that could be solved took a very long time. Today a large majority of these problems have been solved and can be solved in useful time. A number of tools are available for this purpose. Among academic codes, we mention MIPO [5], MINTO [14] and ABACUS [19]. The best-known commercial codes are CPLEX, XPRESS and OSL.

See also

- ▶ Airline Optimization
- ▶ Alignment Problem
- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Graph Coloring
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Maximum Satisfiability Problem
- ▶ Mixed Integer Classification Problems
- ▶ Multidimensional Knapsack Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Optimization in Leveled Graphs

- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Quadratic Knapsack
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem
- ▶ Vehicle Scheduling

References

1. Balas E (1965) An additive algorithm for solving linear programs with 0-1 variables. Oper Res 13:517–546
2. Balas E (1979) Disjunctive programming. Ann Discret Math 5:3–51
3. Balas E (1998) Disjunctive programming: Properties of the convex hull of feasible points. Discrete Appl Math 89:1–44, Invited paper with a foreword by G. Cornuéjols and W.R. Pulleyblank. Originally MSRR 348 Carnegie-Mellon Univ., July 1974.
4. Balas E, Ceria. S, Cornuéjols G (1993) A lift-and-project cutting plane algorithm for mixed 0-1 programs. Math Program 58:295–324
5. Balas E, Ceria. S, Cornuéjols G (1996) Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. Managem Sci 42:1229–1246
6. Chvátal V (1973) Edmonds polytopes and a hierarchy of combinatorial problems. Discret Math 4:305–337
7. Cook WJ, Cunningham WH, Pulleyblank WR, Schrijver A (1997) Combinatorial optimization. Wiley, New York
8. Edmonds J (1965) Maximum matching and a polyhedron with 0-1 vertices. J Res Nat Bureau Standards 69B:125–130
9. Gomory R (1963) An algorithm for integer solutions to linear programs. In: Graves R, Wolfe P (eds) Recent Advances in Mathematical Programming. McGraw-Hill, New York, pp 269–302
10. Hoffman AJ, Kruskal JB (1956) Integral boundary points of convex polyhedra. In: Kuhn HW, Tucker AW (eds) Linear Inequalities and Related Systems. Princeton Univ. Press, Princeton, pp 223–246
11. Land AH, Doig AG (1960) An automatic method for solving discrete programming problems. Econometrica 28:497–520
12. Lawler EL Lenstra JK, Rinrooy Kan AHG, Shmoys DB (eds) (1985) The traveling salesman problem: A guided tour of combinatorial optimization. Wiley, New York
13. Lov´sz L, Schrijver A (1991) Cones of matrices and set functions and 0-1 optimization. SIAM J Optim 1:166–190
14. Nemhauser GL, Savelsbergh MWP, Sigismondi G (1994) MINTO, a mixed integer optimizer. Oper Res Lett 15:47–58
15. Nemhauser GL, Wolsey L (1988) Integer and combinatorial optimization. Wiley, New York
16. Parker G, Rardin R (1988) Discrete optimization. Acad. Press, New York
17. Schrijver A (1986) Theory of linear and integer programming. Wiley, New York
18. Sherali H, Adams W (1990) A hierarchy of relaxations between the continuous and convex hull presentations for 0-1 programming problems. SIAM J Discret Math 3:411–430
19. Thienel S (1995) ABACUS: a branch-and-cut system. PhD Thesis Dept. Computer Sci. Univ. Cologne
20. Wolsey L (1998) Integer programming. Wiley, New York

Integer Programming: Algebraic Methods

REKHA R. THOMAS
Department Math., University Washington, Seattle, USA

MSC2000: 13Cxx, 13Pxx, 14Qxx, 90Cxx

Article Outline

Keywords
Toric Ideals and Integer Programming
The Conti–Traverso Algorithm
Computational Issues
Test Sets in Integer Programming
Universal Gröbner Bases
Variation of Cost Functions in Integer Programming
Group Relaxations in Integer Programming
See also
References

Keywords

Toric ideal; Gröbner basis; Integer programming; Computer algebra; Test sets; universal Gröbner basis; Sensitivity analysis; Group relaxation; Lattice program; Scarf formulation

This article highlights some of the recent results in theoretical integer programming that have been obtained by studying integer programs using tools from commutative algebra and algebraic geometry. The main computational tool involved in the discussion here is the Gröbner basis of a special polynomial ideal called a 'toric ideal' [29]. For connections between Gröbner

bases of toric ideals and polytopes see [29] and for Gröbner basis theory for general polynomial ideals see [1] and [11]. Toric ideals and more generally, 'lattice ideals' [32], have been the subject of much research in the past decade. The discussion in this article follows a specific route through the work done in this area. All effort will be made to include references needed for details and further reading.

Toric Ideals and Integer Programming

We will be concerned with integer programs of the form $\mathrm{IP}_{A,c}(b) := \min\{c \cdot x\colon Ax = b,\ x \in \mathbf{N}^n\}$ where A is a fixed $d \times n$ integer matrix of rank d. Here $\mathbf{N}$ denotes the nonnegative integers. The right-hand side vector b will be assumed to lie in the monoid $\mathrm{pos}_{\mathbf{Z}}(A) := \{Ax\colon x \in \mathbf{N}^n\}$ which guarantees that $\mathrm{IP}_{A,c}(b)$ is always feasible. Let $\ker_{\mathbf{Z}}(A)$ denote the $(n-d)$-dimensional saturated *lattice* $\{u \in \mathbf{Z}^n\colon Au = 0\}$. For simplicity we assume that $\ker_{\mathbf{Z}}(A) \cap \mathbf{N}^n = \{0\}$ which implies that $P_b := \mathrm{conv}\{x \in \mathbf{N}^n\colon Ax = b\}$ is a polytope for all $b \in \mathrm{pos}_{\mathbf{Z}}(A)$. For $b \in \mathrm{pos}_{\mathbf{Z}}(A)$ and a $v \in P_b \cap \mathbf{N}^n$ the set of lattice points in P_b is precisely the congruence class in $\mathbf{N}^n$ of v modulo $\ker_{\mathbf{Z}}(A)$.

The *toric ideal* of A is the d-dimensional binomial prime ideal

$$I_A := \left\langle x^{u^+} - x^{u^-}\colon \quad \begin{array}{c} u := u^+ - u^- \in \ker_{\mathbf{Z}}(A), \\ u^+, u^- \in \mathbf{N}^n \end{array} \right\rangle$$

in $k[\mathbf{x}] := k[x_1, \ldots, x_n]$ where k is a field. The cost vector c lies in $\mathbf{R}^n$ and for each polynomial $f = \sum_{i=1}^{l} k_i x^{\alpha_i} \in I_A$ the *initial term* of f with respect to c, denoted as $\mathrm{in}_c(f)$, is the sum of those terms in f for which $c \cdot \alpha_i$ is maximal. The *initial ideal* of I_A with respect to c is then the ideal $\mathrm{in}_c(I_A) := \langle \mathrm{in}_c(f)\colon f \in I_A\rangle \subset k[\mathbf{x}]$. We will assume unless stated otherwise that the cost vector c is such that $\mathrm{in}_c(I_A)$ is a *monomial ideal*, i.e, $\mathrm{in}_c(I_A)$ can be generated by monomials. Such a c is said to be *generic* with respect to IP_A, the family of all integer programs $\mathrm{IP}_{A,c}(b)$ as b and c vary. Equivalently, c is generic with respect to IP_A if and only if each integer program in the family IP_A, $c := \{\mathrm{IP}_{A,c}(b)\colon b \in \mathrm{pos}_{\mathbf{Z}}(A)\}$ has a unique optimal solution. Note that each lattice point $\alpha \in \mathbf{N}^n$ is a solution to a unique integer program in $\mathrm{IP}_{A,c}$ since α lies in $P_{A\alpha}$ and in no other polytope of the form P_b. The following theorem relates $\mathrm{in}_c(I_A)$ to $\mathrm{IP}_{A,c}$.

Lemma 1 *The lattice point $\alpha \in \mathbf{N}^n$ is a nonoptimal solution to $\mathrm{IP}_{A,c}(A\alpha)$ if and only if the monomial x^α lies in the initial ideal $\mathrm{in}_c(I_A)$.*

Proof The lattice point $\alpha \in \mathbf{N}^n$ is a nonoptimal solution to $\mathrm{IP}_{A,c}(A\alpha)$ if and only if there exists β in $P_{A\alpha} \cap \mathbf{N}^n$ such that $c \cdot \alpha > c \cdot \beta$. This is equivalent to the statement that $x^\alpha - x^\beta$ is a nonzero element of I_A with $\mathrm{in}_c(x^\alpha - x^\beta) = x^\alpha$.

The *standard monomials* of $\mathrm{in}_c(I_A)$ are precisely all the monomials in $k[\mathbf{x}]$ that do not lie in $\mathrm{in}_c(I_A)$.

Corollary 2 *A monomial $x^\gamma \in k[\mathbf{x}]$ is a standard monomial of $\mathrm{in}_c(I_A)$ if and only if γ is the unique optimal solution to the integer program $\mathrm{IP}_{A,c}(A\gamma)$.*

By Corollary 2, there is a bijection between the standard monomials of $\mathrm{in}_c(I_A)$ and the elements of the monoid $\mathrm{pos}_{\mathbf{Z}}(A)$.

The Conti–Traverso Algorithm

In [9], P. Conti and C. Traverso gave an algorithm to solve integer programs using Gröbner bases of toric ideals. A *Gröbner basis* with respect to c, of the toric ideal I_A, is any finite subset $\mathcal{H}$ of I_A such that $\mathrm{in}_c(I_A) = \langle \mathrm{in}_c(f)\colon f \in \mathcal{H}\rangle$. A Gröbner basis $\mathcal{H}$ is *reduced* if it has the additional property that for each $f \in \mathcal{H}$, the coefficient of $\mathrm{in}_c(f)$ is the identity in k and $\mathrm{in}_c(f)$ does not divide any term in another element g of $\mathcal{H}$. Reduced Gröbner bases are unique.

Let $\mathcal{G}_c$ denote the reduced Gröbner basis of I_A with respect to c. Then $\mathcal{G}_c$ has the form $\{x^{\alpha_i} - x^{\beta_i}\colon i = 1, \ldots, t\}$ where $\alpha_i - \beta_i \in \ker_{\mathbf{Z}}(A)$, $\alpha_i, \beta_i \in \mathbf{N}^n$ and $\mathrm{supp}(\alpha_i) \cap \mathrm{supp}(\beta_i) = \emptyset$ for all $i = 1, \ldots, t$. For $p \in \mathbf{Z}^n$, $\mathrm{supp}(p) := \{i \in [n] := \{1, \ldots, n\}\colon p_i \neq 0\}$ denotes the *support* of p. If $x^{\alpha_i} - x^{\beta_i} \in \mathcal{G}_c$ then we always assume that $c \cdot \alpha_i > c \cdot \beta_i$.

Lemma 3 *If $\mathcal{G}_c = \{x^{\alpha_i} - x^{\beta_i}\colon i = 1, \ldots, t\}$ is the reduced Gröbner basis of I_A with respect to c then*

i) $\{x^{\alpha_i}\colon i = 1, \ldots, t\}$ is the minimal generating set of the initial ideal $\mathrm{in}_c(I_A)$; and

ii) for each binomial $x^{\alpha_i} - x^{\beta_i} \in \mathcal{G}_c$, β_i is the unique optimal solution to the integer program $\mathrm{IP}_{A,c}(A\alpha_i)$.

Proof Part i) follows from the definition of reduced Gröbner bases. For each binomial $x^{\alpha_i} - x^{\beta_i} \in \mathcal{G}_c$ we have $A\alpha_i = A\beta_i$, $\alpha_i, \beta_i \in \mathbf{N}^n$ and $c \cdot \alpha_i > c \cdot \beta_i$. If β_i is a nonoptimal solution to $\mathrm{IP}_{A,c}(A\alpha_i)$ then x^{β_i} lies in $\mathrm{in}_c(I_A)$ by Lemma 1 and hence some x^{α_j} for $j = 1, \ldots, t$

will divide x^{β_i} contradicting the definition of a reduced Gröbner basis.

The conditions in Lemma 3 are in fact also sufficient for a finite subset of binomials in I_A to be the reduced Gröbner basis of I_A with respect to c. Given $f \in I_A$, the *normal form* of f with respect to $\mathcal{G}_c$ is the unique remainder obtained upon dividing f by $\mathcal{G}_c$. See [11] for details on the *division algorithm* in $k[\mathbf{x}]$. The structure of $\mathcal{G}_c$ implies that the normal form of a monomial x^v with respect to $\mathcal{G}_c$ is a monomial $x^{v'}$ such that both v and v' are solutions to $\mathrm{IP}_{A,c}(Av)$. The Conti–Traverso algorithm for $\mathrm{IP}_{A,c}$ can be summarized as follows.

	Input: The matrix A and cost vector c.
	Pre-processing:
1	Find a generating set for the toric ideal I_A.
2	Compute the reduced Gröbner basis, $\mathcal{G}_c$, of I_A with respect to the cost vector c.
	To solve $\mathrm{IP}_{A,c}(b)$:
3	Find a solution v to $\mathrm{IP}_{A,c}(b)$.
4	Compute the normal form x^{v^*} of the monomial x^v with respect to the reduced Gröbner basis $\mathcal{G}_c$. Then v^* is the optimal solution to $\mathrm{IP}_{A,c}(b)$.

Conti-Traverso algorithm: How to solve programs in $\mathrm{IP}_{A,c}$

Proof In order to prove the correctness of this algorithm, it suffices to show that for each solution v of $\mathrm{IP}_{A,c}(b)$, the normal form of x^v modulo $\mathcal{G}_c$ is the monomial x^{v^*} where v^* is the unique optimal solution to $\mathrm{IP}_{A,c}(b)$. Suppose x^w is the normal form of the monomial x^v. Then w is also a solution to $\mathrm{IP}_{A,c}(b)$ since the exponent vectors of all monomials $x^{w'}$ obtained during division of x^v by $\mathcal{G}_c$ satisfy $b = Av = Aw'$, $w' \in \mathbf{N}^n$. If $w \neq v^*$, then $x^w - x^{v^*} \in I_A$ and $\mathrm{in}_c(x^w - x^{v^*}) = x^w$ since $c \cdot w > c \cdot v^*$. This implies that $x^w \in \mathrm{in}_c(I_A)$ and hence can be further reduced by $\mathcal{G}_c$ contradicting the definition of the normal form.

Computational Issues

The Conti–Traverso algorithm above raises several computational issues. In Step 1, we require a generating set of the toric ideal I_A which can be a computationally challenging task as the size of A increases. The original Conti–Traverso algorithm starts with the ideal $J_A := \langle x_j t^{a_j -} - t^{a_j +} : j = 1, \ldots, n, t_0 t_1 \cdots t_d - 1 \rangle$ in the larger polynomial ring $k[t_0, \ldots, t_d, x_1, \ldots, x_n]$ where $a_j = a_j^+ - a_j^-$ is the jth column of the matrix A. The toric ideal $I_A = J_A \cap k[\mathbf{x}]$ and hence the reduced Gröbner basis of I_A with respect to c can be obtained by *elimination* (see [11 Chapt. 3]). Although conceptually simple, this method has its limitations as the size of A increases since it requires $d + 1$ extra variables over those present in I_A and the *Buchberger algorithm* for computing Gröbner bases [8] is sensitive to the number of variables involved. Two different algorithms for computing a generating set for I_A without introducing extra variables can be found in [5] and [18] respectively.

Once the generating set of I_A has been found, one needs to compute the reduced Gröbner basis $\mathcal{G}_c$ of I_A. This can be done by any *computer algebra package* that does Gröbner basis computations like Macaulay2, Maple, Reduce, Singular or Cocoa to name a few. Cocoa has a dedicated implementation for toric ideals [6]. As the size of the problem increases, a straightforward computation of reduced Gröbner bases of I_A can become expensive and even impossible. Several tricks can be applied to help the computation, many of which are problem specific.

In Step 3 of the Conti–Traverso algorithm above one requires an initial solution to $\mathrm{IP}_{A,c}(b)$. The original Conti–Traverso algorithm achieves this indirectly during the elimination procedure. Theoretically this task can be as hard as solving $\mathrm{IP}_{A,c}(b)$, although in practice this depends on the specific problem at hand. The last step – to compute the normal form of a monomial with respect to the current reduced Gröbner basis – is (relatively speaking) a computationally easy task.

In practice, one is often only interested in solving $\mathrm{IP}_{A,c}(b)$ for a fixed b. In this situation, the Buchberger algorithm can be *truncated* to produce a sufficient set of binomials that will solve this integer program [35]. This idea was originally introduced in [36] in the context of 0 – 1 integer programs in which all the data is nonnegative. See also [10]. A 'nontoric algorithm' for solving integer programs with fixed right-hand sides was recently proposed in [4].

Test Sets in Integer Programming

A geometric interpretation of the Conti–Traverso algorithm above and more generally of the Buchberger al-

gorithm for toric ideals can be found in [34]. A *test set* for $IP_{A,c}$ is a finite subset of vectors in $\ker_{\mathbf{Z}}(A)$ such that for an integer program $IP_{A,c}(b)$ and a nonoptimal solution v to this program, there is some u in the test set such that $c \cdot v > c \cdot (v - u)$. By interpreting a binomial $x^{\alpha_i} - x^{\beta_i} \in \mathcal{G}_c$ as the vector $\alpha_i - \beta_i \in \ker_Z(A)$, it can be seen that $\mathcal{G}_c$ is the unique minimal test set for the family $IP_{A,c}$. A closely related test set for integer programming is the set of *neighbors of the origin* introduced by H.E. Scarf [26].

The binomial $x^{\alpha_i} - x^{\beta_i} \in \mathcal{G}_c$ can also be viewed as the directed line segment $[\alpha_i, \beta_i]$ directed from α_i to β_i. For each $b \in \text{pos}_{\mathbf{Z}}(A)$ we now construct a directed graph $\mathcal{F}_{b,c}$ as follows: the vertices of this graph are the solutions to $IP_{A,c}(b)$ and the edges of this graph are all possible directed line segments from $\mathcal{G}_c$ that connect two vertices of this graph. Then $\mathcal{G}_c$ is a necessary and sufficient set of directed line segments such that $\mathcal{F}_{b,c}$ is a connected graph with a unique sink (at the optimal solution) for each $b \in \text{pos}_{\mathbf{Z}}(A)$. This geometric interpretation of $\mathcal{G}_c$ can be used to solve several problems. By reversing the directions on all edges in $\mathcal{F}_{b,c}$, one obtains a directed graph with a unique root. One can enumerate all lattice points in P_b by searching this graph starting at its root. This idea was used in [33] to solve a class of manufacturing problems. The graphs $\mathcal{F}_{b,c}$ provide a way to connect all the feasible solutions to an integer program by lattice moves. This idea was applied to statistical sampling in [13].

Universal Gröbner Bases

A subset $\mathcal{U}_A$ of I_A is a *universal Gröbner basis* for I_A if $\mathcal{U}_A$ contains a Gröbner basis of I_A with respect to all (generic) cost vectors $c \in \mathbf{R}^n$. The *Graver basis* of A [16] is a finite universal Gröbner basis of I_A that can be described as follows. For each $\sigma \in \{+, -\}^n$, let $\mathcal{H}_\sigma$ be the unique minimal generating set (over $\mathbf{N}$) of the semigroup $\ker_{\mathbf{Z}}(A) \cap \mathbf{R}^n_\sigma$. Then the Graver basis, $\text{Gr}_A := \cup_\sigma \mathcal{H}_\sigma \backslash \{0\}$. An algorithm to compute Gr_A can be found in [30]. It was shown in [34] that all reduced Gröbner bases of I_A are contained in Gr_A which implies that there are only finitely many distinct reduced Gröbner bases for I_A as c varies over generic cost vectors. Let UGB_A denote the union of all the distinct reduced Gröbner bases of I_A. Then UGB_A is a universal Gröbner basis of I_A that is contained in the Graver basis Gr_A. The following theorem from [30] characterizes the elements of UGB_A and thus allows one to test whether a binomial $x^{\alpha_i} - x^{\beta_i} \in \text{Gr}_A$ belongs UGB_A. A second test can also be found in [30]. A vector $u \in \mathbf{Z}^n$ is said to be *primitive* if the g.c.d. of its components is one.

Theorem 4 *For a primitive vector $u \in \ker_{\mathbf{Z}}(A)$, the binomial $x^{u^+} - x^{u^-}$ belongs to UGB_A if and only if the line segment $[u^+, u^-]$ is a primitive edge in the polytope P_{Au^+}.*

The *degree of a binomial* $x^{\alpha_i} - x^{\beta_i} \in I_A$, is defined to be $\sum \alpha_{ij} + \sum \beta_{ij}$. The degree of the universal Gröbner basis UGB_A is then simply the maximum degree of any binomial in UGB_A. This number is an important complexity measure for the family of integer programs that have A as coefficient matrix. The current best bound for the degree of UGB_A is as follows. See [29, Chapt. 4] for a full discussion.

Theorem 5 *The degree of a binomial $x^{\alpha_i} - x^{\beta_i} \in UGB_A$, is at most $(n - d)(d + 1)D(A)$ where $D(A)$ is the maximum absolute value of the determinant of a $d \times d$ submatrix of A.*

It has been conjectured that this bound can be improved to $(d + 1)D(A)$ and some partial results in this direction can be found in [17].

The universal Gröbner bases of several special instances of A have been investigated in the literature, a few of which we mention here. For the family of $1 \times n$ matrices $A(n) := [1, \ldots, n]$ it was shown in [12] that the Graver basis of $A(n)$ is in bijection with the *primitive partition identities* with largest part n. A matrix $A \in \mathbf{Z}^{d \times n}$ is *unimodular* if the absolute values of the determinants of all its nonsingular maximal minors are the same positive constant. For $u \in \ker_{\mathbf{Z}}(A)$, the binomial $x^{u^+} - x^{u^-} \in \text{I}_A$ is a *circuit* of A if u is primitive and has minimal support with respect to inclusion. Let C_A denote the set of circuits of A. Then in general, $C_A \subseteq \text{UGB}_A \subseteq \text{Gr}_A$. If A is unimodular, then all of the above containments hold at equality although the converse is false: there are nonunimodular matrices for which $C_A = \text{Gr}_A$. If A_n is the node-edge incidence matrix of the complete graph K_n then the elements in UGB_{A_n} can be identified with certain subgraphs of K_n. Gröbner bases of these matrices were investigated in [23]. The integer programs associated with A_n are the *b-matching prob-*

lems in the literature [24]. See [29, Chapt. 14] for some other specific examples of Gröbner bases.

Variation of Cost Functions in Integer Programming

We now consider all cost vectors in $\mathbf{R}^n$ (not just the generic ones) and study the effect of varying them. As seen earlier I_A has only finitely many distinct reduced Gröbner bases as c is varied over the generic cost vectors. We say that two cost vectors c_1 and c_2 are *equivalent* with respect to IP_A if for each $b \in \mathrm{pos}_{\mathbf{Z}}(A)$, the integer programs $\mathrm{IP}_{A,c_1}(b)$ and $\mathrm{IP}_{A,c_2}(b)$ have the same set of optimal solutions. The Gröbner basis approach to integer programming allows a complete characterization of the structure of these equivalence classes of cost vectors.

Theorem 6 [30]

i) There exists only finitely many equivalence classes of cost vectors with respect to IP_A.

ii) Each equivalence class is the relative interior of a convex polyhedral cone in $\mathbf{R}^n$.

iii) The collection of all these cones defines a complete polyhedral fan in $\mathbf{R}^n$ called the Gröbner fan of A.

iv) Let db denote any probability measure with support $pos_{\mathbf{Z}}(A)$ such that $\int_b b\, db < \infty$.

Then the Minkowski integral $St(A) = \int_b P_b\, db$ is an $(n-d)$-dimensional convex polytope, called the state polytope of A. The normal fan of St(A) equals the Gröbner fan of A.

Gröbner fans and state polytopes of graded polynomial ideals were introduced in [25] and [2] respectively. For a toric ideal both these entities have self contained construction methods that are rooted in the combinatorics of these ideals [30]. For a software system for computing Gröbner fans of toric ideals see [21].

We call P_b for $b \in \mathrm{pos}_{\mathbf{Z}}(A)$ a *Gröbner fiber* of A if there is some $x^{u^+} - x^{u^-} \in \mathrm{UGB}_A$ such that $b = Au^+ = Au^-$. Since there are only finitely many elements in UGB_A the matrix A has only finitely many Gröbner fibers. Then the *Minkowski sum* of all Gröbner fibers of A is a state polytope of A. For a survey of algorithms to construct state polytopes and Gröbner fans of graded polynomial ideals see [29, Chapt. 2; 3]. The Gröbner fan of A provides a model for global *sensitivity analysis* for the family of integer programs $\mathrm{IP}_{A,c}$.

We now briefly discuss a theory analogous to the above for linear programming based on results in [7] and [14]. For a comparison of integer and linear programming from this point of view see [30]. Let $\mathrm{LP}_{A,c}(b) := \min\{c \cdot x\colon Ax = b,\ x \geq 0\}$ where A and c are as before and b is any vector in the rational polyhedral cone $\mathrm{pos}(A) := \{Ax\colon x \geq 0\}$. We define two cost vectors c_1 and c_2 to be *equivalent* with respect to LP_A if the linear programs $\mathrm{LP}_{A,c_1}(b)$ and $\mathrm{LP}_{A,c_2}(b)$ have the same set of optimal solutions for all $b \in \mathrm{pos}(A)$. Let $\mathcal{A} := \{a_1, \ldots, a_n\}$ be the vector configuration in $\mathbf{Z}^d$ consisting of the columns of A. For a subset $\sigma \subseteq \mathcal{A}$, we let $\mathrm{pos}(\sigma)$ denote the cone generated by σ. A *polyhedral subdivision* Δ of $\mathcal{A}$ is a collection of subsets of $\mathcal{A}$ such that $\{\mathrm{pos}(\sigma)\colon \sigma \in \Delta\}$ is a set of cones in a polyhedral fan whose support is $\mathrm{pos}(\mathcal{A})$. The elements of Δ are called the *faces* or *cells* of Δ. For convenience we identify $\mathcal{A}$ with the set of indices $[n]$ and any subset of $\mathcal{A}$ by the corresponding subset $\sigma \subseteq [n]$. A cost vector $c \in \mathbf{R}^n$ induces the *regular subdivision* Δ_c of $\mathcal{A}$ [7,14] as follows: σ is a face of Δ_c if there exists a vector $y \in \mathbf{R}^d$ such that $a_j \cdot y = c_j$ whenever $j \in \sigma$ and $a_j \cdot y < c_j$ otherwise. A cost vector $c \in \mathbf{R}^n$ is said to be *generic* with respect to LP_A if every linear program in the family $\mathrm{LP}_{A,c}$ has a unique optimal solution. When c is generic for LP_A, the regular subdivision Δ_c is in fact a *triangulation* called the *regular triangulation* of $\mathcal{A}$ with respect to c.

Two cost vectors c_1 and c_2 are equivalent with respect to LP_A if and only if $\Delta_{c_1} = \Delta_{c_2}$. The equivalence class of c with respect to LP_A is hence $\{c' \in \mathbf{R}^n\colon \Delta_{c'} = \Delta_c\}$ which is the relative interior of a polyhedral cone in $\mathbf{R}^n$ called the *secondary cone* of c, denoted as S_c. The cone S_c is n-dimensional if and only if c is generic with respect to LP_A. The set of all equivalence classes of cost vectors fit together to form a complete polyhedral fan in $\mathbf{R}^n$ called the *secondary fan* of A. This fan is the normal fan of a polytope called the *secondary polytope* of A. See [7] for construction methods for both the secondary fan and polytope of A.

We conclude this section by showing that the Gröbner and secondary fans of A are related. The *Stanley–Reisner ideal* of Δ_c is the square-free monomial ideal $\langle x_{i_1} \cdots x_{i_r}\colon \{i_1, \ldots, i_r\}$ is a nonface of $\Delta_c\rangle \subset k[\mathbf{x}]$.

Theorem 7 [28] *The radical of the initial ideal $in_c(I_A)$ is the Stanley–Reisner ideal of the regular triangulation Δ_c.*

Corollary 8 [28]

i) The Gröbner fan of A is a refinement of the secondary fan of A.

ii) A secondary polytope of A is a summand of a state polytope of A.

Corollary 8 reaffirms the view that integer programming is an arithmetic refinement of linear programming.

Group Relaxations in Integer Programming

We now investigate group relaxations of integer programs in the family $\mathrm{IP}_{A,c}$ from an algebraic point of view. The results in this section are taken from [19,20] and [32], sometimes after an appropriate translation into polyhedral language. See these papers for the algebraic motivations that led to these results.

The group relaxation of $\mathrm{IP}_{A,c}(b)$ [15] is the program $\mathrm{Group}^{\sigma}(b) := \min\{\widetilde{c}_{\overline{\sigma}} \cdot x_{\overline{\sigma}} : A_{\sigma}x_{\sigma} + A_{\overline{\sigma}}x_{\overline{\sigma}} = b, x_{\overline{\sigma}} \geq 0, x = (x_{\sigma}, x_{\overline{\sigma}}) \mathbf{Z}^n\}$, where A_{σ}, the submatrix of A whose columns are indexed by $\sigma \subseteq [n]$, is the optimal basis of the linear program $\mathrm{LP}_{A,c}(b)$ and $\widetilde{c}_{\overline{\sigma}} = c_{\overline{\sigma}} - c_{\sigma}A_{\sigma}^{-1}A_{\overline{\sigma}}$. Here the cost vector c has also been partitioned as $c = (c_{\sigma}, c_{\overline{\sigma}})$ using the set $\sigma \subseteq [n]$.

Definition 9 Suppose $\mathcal{L}$ is any sublattice of $\mathbf{Z}^n$, $w \in \mathbf{R}^n$ and $v \in \mathbf{N}^n$. The *lattice program* $\mathrm{Lat}_{\mathcal{L},w}(v)$ defined by this data is

$$\begin{cases} \min & w \cdot u \\ \text{s.t.} & u \equiv v \mod \mathcal{L}, \\ & u \in \mathbf{N}^n. \end{cases}$$

Lattice programs are a generalization of integer programs: $\mathrm{IP}_{A,c}(b) = \mathrm{Lat}_{\mathcal{L},c}(v)$ where $\mathcal{L} = \ker_{\mathbf{Z}}(A)$ and v is any feasible solution to $\mathrm{IP}_{A,c}(b)$. Gröbner basis methods for integer programs can be extended to solve lattice programs (see [20,32]). Given the lattice $\mathcal{L}$ and a cost vector w, we first construct the *lattice ideal* $I_{\mathcal{L}} = \langle \mathbf{x}^{\alpha} - \mathbf{x}^{\beta} : \alpha - \beta \in \mathcal{L}, \alpha, \beta \in \mathbf{N}^n \rangle \subset k[\mathbf{x}]$. We then compute the reduced Gröbner basis of $I_{\mathcal{L}}$ with respect to w denoted as $\mathcal{G}_w(I_{\mathcal{L}})$. (If w does not induce a total order on $\mathbf{N}^n$ via the inner product $w \cdot x$, $x \in \mathbf{N}^n$, then we use a tie breaking term order to refine the order induced by w.) For a particular lattice program $\mathrm{Lat}_{\mathcal{L},w}(v)$, the optimal solution is the exponent vector of the normal form of x^v with respect to $\mathcal{G}_w(I_{\mathcal{L}})$.

Let $\tau \subseteq [n]$ and $\pi_{\tau} : \mathbf{Z}^n \to \mathbf{Z}^{|\overline{\tau}|}$ be the coordinate projection map where the coordinates indexed by τ are eliminated. Consider the lattice $\mathcal{L}_{\tau} := \pi_{\tau}(\mathcal{L})$ where $\mathcal{L} = \ker_{\mathbf{Z}}(A)$. Given a basis $\{b_1, \ldots, b_{n-d}\}$ of $\mathcal{L}$, the set $\{\pi_{\tau}(b_1), \ldots, \pi_{\tau}(b_{n-d})\}$ forms a basis for $\mathcal{L}_{\tau}$. Further, $\pi_{\tau} : \mathcal{L} \to \mathcal{L}_{\tau}$ is an isomorphism whenever $\mathrm{rank}(A_{\tau}) = |\tau|$.

Proposition 10 [32] *Let v be a feasible solution to $\mathrm{IP}_{A,c}(b)$ and A_{σ} be the optimal basis of $\mathrm{LP}_{A,c}(b)$. Then the group relaxation $\mathrm{Group}^{\sigma}(b)$ of $\mathrm{IP}_{A,c}(b)$ is the lattice program $\mathrm{Lat}_{\mathcal{L}_{\sigma},\widetilde{c}_{\sigma}}(\pi_{\sigma}(v))$ where $\widetilde{c}_{\sigma} = \pi_{\sigma}(c - c_{\sigma}(A_{\sigma})^{-1}A) = c_{\overline{\sigma}} - c_{\sigma}A_{\sigma}^{-1}A_{\overline{\sigma}}$.*

The program $\mathrm{Group}^{\sigma}(b)$ can be solved by Gröbner basis methods as explained earlier or by dynamic programming [15]. The optimal solution $x^*_{\overline{\sigma}}$ to $\mathrm{Group}^{\sigma}(b)$ is then lifted to the unique vector $x^* = (x^*_{\sigma}, x^*_{\overline{\sigma}}) \in \mathbf{Z}^n$ by solving the equation $A_{\sigma}x_{\sigma} + A_{\overline{\sigma}}x^*_{\overline{\sigma}} = b$. If all components of x^*_{σ} are nonnegative then x^* is the optimal solution to $\mathrm{IP}_{A,c}(b)$. Otherwise $c \cdot x^*$ is a lower bound to the optimal value of $\mathrm{IP}_{A,c}(b)$.

When $\mathrm{Group}^{\sigma}(b)$ fails to solve $\mathrm{IP}_{A,c}(b)$, L. Wolsey [37] suggested using *extended group relaxations* of $\mathrm{IP}_{A,c}(b)$. We introduce a more general set of extended group relaxations of $\mathrm{IP}_{A,c}(b)$ inspired by the following close relationship between the linear programs in $\mathrm{LP}_{A,c}$ and the regular triangulation Δ_c.

Proposition 11 [30] *The optimal solutions x to $\mathrm{LP}_{A,c}(b)$ are the solutions to the problem: Find $x \in \boldsymbol{R}^n$ such that $Ax = b$, $x \geq 0$, and $\mathrm{supp}(x)$ is a subset of a face of Δ_c.*

Proposition 11 says that the set σ in $\mathrm{Group}^{\sigma}(b)$ is a maximal face of Δ_c.

Definition 12 Consider the integer program $\mathrm{IP}_{A,c}(b)$ and a feasible solution v to this program. Let τ be a face of Δ_c and σ be any maximal face of Δ_c containing τ. Then the *group relaxation* of $\mathrm{IP}_{A,c}(b)$ with respect to τ denoted as $\mathrm{Group}^{\tau}(b)$ is the lattice program $\mathrm{Lat}_{\mathcal{L}_{\tau},\widetilde{c}_{\tau}}(\pi_{\tau}(v))$ where $\widetilde{c}_{\tau} := \pi_{\tau}(c - c_{\sigma}A_{\sigma}^{-1}A)$.

The extended group relaxations in [37] are precisely those $\mathrm{Group}^{\tau}(b)$s where τ is a subset of the maximal face σ of Δ_c that gives the optimal basis of $\mathrm{LP}_{A,c}(b)$. Clearly, one such relaxation will solve $\mathrm{IP}_{A,c}(b)$. However, we consider all relaxations of $\mathrm{IP}_{A,c}(b)$ of the form $\mathrm{Group}^{\tau}(b)$ as τ varies over all faces of Δ_c in order to

avoid keeping track of which b is being considered and what the optimal basis of $\mathrm{LP}_{A,c}(b)$ is.

It was shown in [32] that the lattice program $\mathrm{Group}^{\tau}(b)$ is related to the *localization* of the initial ideal $\mathrm{in}_c(I_A)$ at the prime ideal $p_\tau := \langle x_j : j \notin \tau \rangle$ in $k[\mathbf{x}]$. Since group relaxations are always defined with respect to a face τ of Δ_c, we are guaranteed that $\mathrm{rank}(A_\tau) = |\tau|$ which allows a unique lifting of the optimal solution of $\mathrm{Group}^{\tau}(b)$ to a vector in the same congruence class modulo $\mathcal{L}$ as the solutions to $\mathrm{IP}_{A,c}(b)$.

Theorem 13 [20] *Suppose $u' \in \mathbf{N}^{|\overline{\tau}|}$ is the optimal solution to the group relaxation $\mathrm{Group}^{\tau}(b)$. Then there exists a unique $u \in \mathbf{Z}^n$ such that $A(u - v) = 0$ for any feasible solution v to $IP_{A,c}(b)$ and $\pi_\tau(u) = u'$. If $u \geq 0$ then it is the optimal solution to $IP_{A,c}(b)$.*

A group relaxation $\mathrm{Group}^{\tau}(b)$ is easiest to solve when τ is a maximal face of Δ_c. In this situation, the lattice ideal $I_{\mathcal{L}_\tau}$ is zero dimensional and hence their Gröbner bases are easier to compute than otherwise. We call such group relaxations the *Gomory relaxations* of $\mathrm{IP}_{A,c}(b)$. In general one is most interested in those group relaxations $\mathrm{Group}^{\tau}(b)$ that solve $\mathrm{IP}_{A,c}(b)$ with $|\tau|$ as large as possible. In the rest of this section we study several structural properties of these 'least tight' extended group relaxations that solve programs in $\mathrm{IP}_{A,c}$. We first need a diversion into combinatorics.

For $m \in \mathbf{N}^n$, we define *support* of $x^m \in k[\mathbf{x}]$ to be $\mathrm{supp}(m)$.

Definition 14 For a monomial $x^m \in k[\mathbf{x}]$ and $\sigma \subseteq [n]$, we say that (x^m, σ) is an *admissible pair* of a monomial ideal M if

i) $\mathrm{supp}(m) \cap \sigma = \emptyset$; and
ii) every monomial in $x^m \cdot k[x_j : j \in \sigma]$ is a standard monomial of M.

There is a natural partial order on the set of all admissible pairs of M given by $(x^m, \sigma) \leq (x^{m'}, \sigma')$ if and only if x^m divides $x^{m'}$ and $\mathrm{supp}(x^{m'}/x^m) \cup \sigma' \subseteq \sigma$.

Definition 15 An admissible pair (x^m, σ) of M is called a *standard pair* of M if it is a minimal element in the poset of all admissible pairs with respect to the above partial order.

The standard pairs of M induce a unique covering of the set of standard monomials of M which we refer to as the *standard pair decomposition* of M. This decomposition was introduced in [31] to study the associated primes of M and their multiplicities and thus the arithmetic degree [3] of M. When M is the initial ideal of a toric ideal stronger conclusions can be drawn. In our exposition below we bypass much of the algebraic results associated with the standard pair decomposition of M, but instead use these results to motivate appropriate definitions to continue our discussion of group relaxations.

Definition 16

i) For $\tau \subseteq [n]$, we define the *multiplicity* of τ, denoted as $\mathrm{mult}(\tau)$, to be the number of standard pairs of the form (x^m, τ) in the standard pair decomposition of M.
ii) The sum of the multiplicities of τ as τ varies over the subsets of $[n]$ is called the *arithmetic degree* of M, denoted as $\mathrm{arithdeg}(M)$.

In the rest of this section we let $M = \mathrm{in}_c(I_A)$.

Proposition 17 [29, Sect. 12.D]

i) If (x^m, τ) is a standard pair of $in_c(I_A)$ then τ is a face of Δ_c.
ii) The standard pair $(1, \sigma)$ occurs in the standard pair decomposition of $in_c(I_A)$ if and only if σ is a maximal face of Δ_c. In this case, $mult(\sigma)$ is the normalized volume of σ in Δ_c.

The *normalized volume* of a maximal face $\sigma \in \Delta_c$ is the quotient $|\det(A_\sigma)|/T$ where T is the g.c.d. of all $|\det(A_{\sigma'})|$ as σ' varies over the maximal faces of Δ_c. We note that the converse to Proposition 17i) is false. If τ is a nonmaximal face of Δ_c then there may not be a standard pair of the form (x^m, τ) in the standard pair decomposition of $\mathrm{in}_c(I_A)$.

The standard pair decomposition of $\mathrm{in}_c(I_A)$ reduces the problem of solving integer programs in $\mathrm{IP}_{A,c}$ to solving systems of linear equations: if β is the optimal solution to the program $\mathrm{IP}_{A,c}(b)$, then the monomial x^β is covered by some standard pair (x^u, τ). Thinking of u as a vector in $\mathbf{N}^{|\overline{\tau}|}$ (by adding zero components if necessary), we get $\beta_{\overline{\tau}} = u$ and β_τ is the unique solution to the linear system $A_\tau x_\tau = b - A_{\overline{\tau}} u$. Therefore, if the standard pairs of $\mathrm{in}_c(I_A)$ are known a priori, then one can set up $\mathrm{arithdeg}(\mathrm{in}_c(I_A))$-many systems of linear equations – one for each standard pair. For each $b \in \mathrm{pos}_{\mathbf{Z}}(A)$, one then solves for β_τ as above. Whenever the β_τ obtained this way is both integral and nonnegative, we have found the optimal solution to $\mathrm{IP}_{A,c}(b)$. Hence

arithdeg($\text{in}_c(I_A)$) can be seen as a complexity measure of $\text{IP}_{A,c}$. See [22] for another preprocessing of $\text{IP}_{A,c}$ that reduces solving $\text{IP}_{A,c}(b)$ to solving a sequence of subproblems involving super additive functions.

Theorem 18 [20]

i) The integer program $IP_{A,c}(b)$ is solved by the group relaxation $\text{Group}^\tau(b)$ if and only if the monomial x^β, where β is the optimal solution to $IP_{A,c}(b)$, is covered by a standard pair (x^α, τ') of $in_c(I_A)$ for some $\tau' \supseteq \tau$.

In order to state the main results, we need yet another interpretation of group relaxations of programs in $\text{IP}_{A,c}$.

Let $\phi_A: \mathbf{N}^n \to \mathbf{Z}^d$ be the linear map $x \longmapsto Ax$. Then P_b is the convex hull of $\phi_A^{-1}(b)$ for each $b \in \text{pos}_{\mathbf{Z}}(A)$. Consider a matrix $B \in \mathbf{Z}^{n \times (n-d)}$ such that the columns of B form a basis for $\ker_{\mathbf{Z}}(A)$ (as an Abelian group). For $v \in \phi_A^{-1}(b)$ we can identify $\phi_A^{-1}(b)$ with the lattice points in the polytope

$$Q_v := \left\{u \in \mathbf{R}^{n-d}:\ Bu \le v\right\}, \tag{1}$$

via the bijection $Q_v \cap \mathbf{Z}^{n-d} \to \phi_A^{-1}(b)$ such that $u \to v - Bu$. Under this bijection, $v \in \phi_A^{-1}(b)$ corresponds to $0 \in Q_v$. We refer to Q_v as a *Scarf formulation* of $P_b = \text{conv}(\phi_A^{-1}(b))$. If $v, v' \in \phi_A^{-1}(b)$, then Q_v and $Q_{v'}$ are simply lattice translates of each other.

Proposition 19 *If v is a feasible solution to $IP_{A,c}(b)$, then $IP_{A,c}(b)$ is equivalent to*

$$\min\left\{-(cB)\cdot u:\ u \in Q_v \cap \mathbf{Z}^{n-d}\right\} \tag{2}$$

Proof A lattice point v^* is the optimal solution to $\text{IP}_{A,c}(b)$:

$\Leftrightarrow$ there exists $u^* \in \mathbf{Z}^{n-d}$ such that $v^* = v - Bu^* \ge 0$ and $c(v - Bu^*) < c(v - Bu)$ for all $u \ne u^* \in \mathbf{Z}^{n-d}$ with $v - Bu \ge 0$

$\Leftrightarrow$ there exist $u^* \in Q_v \cap \mathbf{Z}^{n-d}$ such that $-(cB)\cdot u^* < -(cB)\cdot u$ for all $u \in Q_v \cap \mathbf{Z}^{n-d}$

$\Leftrightarrow$ u^* is the optimal solution of the integer program (2).

We will refer to the integer program (2) as a *Scarf formulation* of $\text{IP}_{A,c}(b)$. Using the optimal solution u^* of the Scarf formulation (2), we define the following subpolytope of Q_v:

$$Q_v(u^*) := \left\{u \in \mathbf{R}^{n-d}:\ Bu \le v,\ -(cB)\cdot u \le -(cB)\cdot u^*\right\}. \tag{3}$$

Theorem 20 *Let $v \in \mathbf{N}^n$ be a feasible solution to $IP_{A,c}(b)$. Then u^* is the optimal solution to (2) if and only if u^* is the unique lattice point in $Q_v(u^*)$. In particular, v is the optimal solution to $IP_{A,c}(b)$ if and only if 0 is the unique lattice point in $Q_v(0) = \{u \in \mathbf{R}^{n-d}: Bu \le v, -(cB)\cdot u \le 0\}$.*

Proof A vector $u^* \in \mathbf{Z}^{n-d}$ is the optimal solution to (2) if and only if u^* is in Q_v and there is no $u \in Q_v \cap \mathbf{Z}^{n-d}$ such that $-(cB)\cdot u < -(cB)\cdot u^*$. Since c is a generic cost vector, this is equivalent to u^* being the unique lattice point in $Q_v(u^*)$. The second statement follows immediately.

Corollary 21 *A monomial x^v is a standard monomial of $in_c(I_A)$ if and only if 0 is the unique lattice point in $Q_v(0)$.*

Let B^τ denote the submatrix of B whose rows are indexed by the set $\tau \subseteq [n]$.

Lemma 22 *Suppose σ is a maximal face of Δ_c and τ a subface of σ. Then $\widetilde{c}_\tau B^{\overline{\tau}} = cB$ where $\widetilde{c}_\tau = \pi_\tau(c - c_\sigma(A_\sigma)^{-1}A)$.*

Proof Since the support of $c - c_\sigma\,(A_\sigma)^{-1}A$ is contained in $\overline{\tau}$, $\widetilde{c}_\tau B^{\overline{\tau}} = (c - c_\sigma(A_\sigma)^{-1}A)B = cB$.

Theorem 23 *Let v be a feasible solution to $IP_{A,c}(b)$, and suppose that σ is a maximal face of Δ_c and τ a subface of σ. Then the group relaxation $\text{Group}^\tau(b)$ is the integer program $\min\left\{-(cB)\cdot u:\ B^{\overline{\tau}}u \le \pi_\tau(v),\ u \in \mathbf{Z}^{n-d}\right\}$.*

Proof Since $\mathcal{L}_\tau = \left\{B^{\overline{\tau}}u:\ u \in \mathbf{Z}^{n-d}\right\}$, we have:

$$\begin{aligned}
&\text{Lat}_{\mathcal{L}_\tau,\widetilde{c}_\tau}(\pi_\tau(v))\\
&:= \min\left\{\widetilde{c}_\tau\cdot w:\ \begin{array}{c} w \equiv \pi_\tau(v) \pmod{\mathcal{L}_\tau},\\ w \in \mathbf{N}^{|\overline{\tau}|}\end{array}\right\}\\
&= \min\left\{\widetilde{c}_\tau\cdot w:\ \begin{array}{c} w \in \mathbf{N}^{|\overline{\tau}|},\\ w = \pi_\tau(v) - B^{\overline{\tau}}u,\\ u \in \mathbf{Z}^{n-d}\end{array}\right\}\\
&= \min\left\{\widetilde{c}_\tau\cdot w:\ \begin{array}{c} \pi_\tau(v) - B^{\overline{\tau}}u \ge 0,\\ u \in \mathbf{Z}^{n-d}\end{array}\right\}\\
&= \min\left\{\widetilde{c}_\tau\cdot(\pi_\tau(v) - B^{\overline{\tau}}u):\ \begin{array}{c} B^{\overline{\tau}}u \le \pi_\tau(v),\\ u \in \mathbf{Z}^{n-d}\end{array}\right\}\\
&= \min\left\{(-\widetilde{c}_\tau B^{\overline{\tau}})\cdot u:\ \begin{array}{c} B^{\overline{\tau}}u \le \pi_\tau(v),\\ u \in \mathbf{Z}^{n-d}\end{array}\right\}
\end{aligned}$$

$$= \min\left\{-(cB)\cdot u:\ \begin{array}{c} B^{\overline{\tau}}u \le \pi_\tau(v),\\ u \in \mathbf{Z}^{n-d}\end{array}\right\}$$

by Lemma 22.

We will denote the polyhedron obtained from Q_v by removing the inequalities corresponding to τ by $Q_v^{\overline{\tau}}$. By the above theorem, solving the group relaxation of $\mathrm{IP}_{A,c}(b)$ with respect to $\tau \in \Delta_c$ is equivalent to minimizing the linear functional $-(cB)\cdot u$ over the lattice points in $Q_v^{\overline{\tau}}$. Now we can characterize which group relaxations will solve $\mathrm{IP}_{A,c}(b)$.

Corollary 24

i) Let v be a feasible solution to $IP_{A,c}(b)$. Then Group$^\tau(b)$ solves $IP_{A,c}(b)$ if and only if the programs $\min\{-(cB)\cdot u:\ u \in Q_v \cap \mathbf{Z}^{n-d}\}$ and $\min\left\{-(cB)\cdot u:\ u \in Q_v^{\overline{\tau}} \cap \mathbf{Z}^{n-d}\right\}$ have the same optimal solutions.

ii) If v is optimal for $IP_{A,c}(b)$, then Group$^\tau(b)$ solves $IP_{A,c}(b)$ if and only if 0 is the unique lattice point in $Q_v^{\overline{\tau}}(0) := \{u \in \mathbf{R}^{n-d}:\ B^{\overline{\tau}}u \le \pi_\tau(v), -(cB)\cdot u \le 0\}$.

For a polyhedron $P = \{x \in \mathbf{R}^p: Tx \le t\}$ we say that an inequality $T_i x \le t_i$ is *essential* if the relaxation of the polyhedron obtained by removing $T_i x \le t_i$ contains a new lattice point.

Theorem 25 [19] *An admissible pair (x^v, τ) is a standard pair of $in_c(I_A)$ if and only if 0 is the unique lattice point in $Q_v^{\overline{\tau}}(0)$ and all of the inequalities in the system $B^{\overline{\tau}}u \le \pi_\tau(v)$ are essential.*

Using the above characterization of the standard pairs of $\mathrm{in}_c(I_A)$ we obtain a combinatorial interpretation formult(τ) and arithdeg($\mathrm{in}_c(I_A)$).

Corollary 26

i) The multiplicity of τ is the number of polytopes of the form $Q_v^{\overline{\tau}}(0) := \{u \in \mathbf{R}^{n-d}:\ B^{\overline{\tau}}u \le v,\ -(cB)\cdot u \le 0\}$ where $v \in \mathbf{N}^{|\overline{\tau}|}$, 0 is the unique lattice point in $Q_v^{\overline{\tau}}(0)$ and all inequalities in $B^{\overline{\tau}}u \le v$ are essential.

ii) The arithmetic degree of $in_c(I_A)$ is the total number of such polytopes $Q_v^{\overline{\tau}}(0)$ as τ ranges over the subsets of $[n]$.

The result that mult(σ) is the normalized volume of σ when σ is a maximal face of Δ_c is a special case of the above more general interpretation of multiplicity. See [19].

Corollary 27 *For the initial ideal $in_c(I_A)$, the following are equivalent:*

i) The initial ideal $in_c(I_A)$ has no standard pairs of the form (x^m, τ) where τ is a nonmaximal face of Δ_c.

ii) For a face $\tau \in \Delta_c$, if there exists a $v \in \mathbf{N}^{|\overline{\tau}|}$ such that $Q_v^{\overline{\tau}}(0)$ contains the origin as its unique lattice point and all inequalities are essential then τ is a maximal face of Δ_c and $Q_v^{\overline{\tau}}(0)$ is a simplex.

iii) All programs in $IP_{A,c}$ can be solved by group relaxations with respect to maximal faces of Δ_c.

iv) The arithmetic degree of $in_c(I_A)$ is vol(conv($\mathcal{A}$)).

Proposition 17 shows that the set of all τ in Δ_c that index standard pairs of $\mathrm{in}_c(I_A)$ is a sub poset (with respect to inclusion) of the face lattice of Δ_c. We denote this subposet by Std(Δ_c). Note that both (face lattice of) Δ_c and Std(Δ_c) have the same maximal elements. We now show that the elements of Std(Δ_c) come in chains.

Theorem 28 [19] *Let τ, $|\tau| < d$ be a nonmaximal face of Δ_c such that $\tau \in Std(\Delta_c)$. Then there exists some $\tau' \in \Delta_c$ such that $\tau' \in Std(\Delta_c)$ with the property that*

i) $\tau' \supset \tau$ and

ii) $|\tau'| = |\tau| + 1$.

See [19] for a proof of this theorem. The tools needed in the proof are polyhedral and depend heavily on the polyhedral interpretation of a standard pair as given in Theorem 25. In terms of group relaxations, Theorem 28 is saying that whenever there is a $b \in \mathrm{pos}_{\mathbf{Z}}(A)$ that is solved by a 'least tight' Group$^\tau(b)$, then there exists a $b' \in \mathrm{pos}_{\mathbf{Z}}(A)$ that is solved by a 'least tight' Group$^{\tau'}(b)$ where

i) $\tau' \supset \tau$ and

ii) $|\tau'| = |\tau| + 1$.

Hence the 'least tight' extended group relaxations that solve the programs in $\mathrm{IP}_{A,c}$ form saturated chains in the poset Std(Δ_c).

Since a maximal face of Δ_c has dimension d, the length of a maximal chain in Std(Δ_c), which we denote as length(Std(Δ_c)), is at most d. However, when $n - d$ which is the the corank of A is small compared to d, length(Std(Δ_c)) has a stronger upper bound as shown below. We need the following result [27, Corol. 16.5a]:

Theorem 29 *Let $Ax \le b$ be a system of linear inequalities in n variables, and let $c \in \mathbf{R}^n$. If $\max\{c\cdot x: Ax \le b, x \in \mathbf{Z}^n\}$ is finite, then $\max\{c\cdot x: Ax \le b, x \in \mathbf{Z}^n\} = \max\{c\cdot x: A'x \le b', x \in \mathbf{Z}^n\}$ for some subsystem $A'x \le b'$ of $Ax \le b$ with at most $2^n - 1$ inequalities.*

Theorem 30 *The length of a maximal chain in Std(Δ_c) is at most min(d, $2^{n-d} - (n - d + 1)$).*

Proof Suppose v is the optimal solution to $\mathrm{IP}_{A,c}(b)$ which is equivalent to $\min\{-(cB)\cdot u: Bu \le v, u \in \mathbf{Z}^{n-d}\}$. By Theorem 29, we need at most $2^{n-\mathbf{d}} - 1$ inequalities to describe the same integer program. This means we can remove at least $n - (2^{n-d} - 1)$ inequalities from $Bu \le v$ without changing the optimal solution. Therefore by Theorem 23, $\mathrm{IP}_{A,c}(b)$ can be solved by a group relaxation with respect to a $\tau \in \Delta_c$ of size at least $n - (2^{n-d} - 1)$. This implies that the maximal length of a chain in Std(Δ_c) is at most $d - (n - (2^{n-d} - 1)) = 2^{n-d} - (n - d + 1)$.

Corollary 31 *If $A \in \mathbf{Z}^{d\times n}$ has corank two, then length(Std(Δ_c)) ≤ 1.*

Proof In this situation, $2^{n-d} - (n - d + 1) = 4 - (4 - 2 + 1) = 4 - 3 = 1$.

Corollary 32 *All programs in the family $IP_{A,c}$ can be solved by group relaxations with respect to a $\tau \in \Delta_c$ of size at least max(0, $n - (2^{n-d} - 1)$).*

We conclude by remarking that the bound in Theorem 30 is sharp. See [19] for details.

See also

- Branch and Price: Integer Programming with Column Generation
- Decomposition Techniques for MILP: Lagrangian Relaxation
- Integer Linear Complementary Problem
- Integer Programming
- Integer Programming: Branch and Bound Methods
- Integer Programming: Branch and Cut Algorithms
- Integer Programming: Cutting Plane Algorithms
- Integer Programming Duality
- Integer Programming: Lagrangian Relaxation
- LCP: Pardalos–Rosen Mixed Integer Formulation
- Mixed Integer Classification Problems
- Multi-objective Integer Linear Programming
- Multi-objective Mixed Integer Programming
- Multiparametric Mixed Integer Linear Programming
- Parametric Mixed Integer Nonlinear Optimization
- Set Covering, Packing and Partitioning Problems
- Simplicial Pivoting Algorithms for Integer Programming
- Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- Stochastic Integer Programs
- Time-dependent Traveling Salesman Problem

References

1. Adams WW, Loustaunau P (1994) An introduction to Gröbner bases. Grad Stud Math, vol III. Amer. Math. Soc., Providence, RI
2. Bayer D, Morrison I (1998) Standard bases and geometric invariant theory I. J Symbolic Computation 6:209–217
3. Bayer D, Mumford D (1993) What can be computed in algebraic geometry? In: Eisenbud D, Robbiano L (eds) Computational Algebraic Geometry and Commutative Algebra. Symp Math. Cambridge Univ. Press, Cambridge, pp 1–48
4. Bertsimas D, Perakis G, Tayur S (1999) A new algebraic geometry algorithm for integer programming. Manuscript
5. Biase F Di, Urbanke R (1995) An algorithm to compute the kernel of certain polynomial ring homomorphisms. Experimental Math 4:227–234
6. Bigatti A, LaScala R, Robbiano L (1999) Computing toric ideals. J Symbolic Computation 27:351–365
7. Billera LJ, Filliman P, Sturmfels B (1990) Constructions and complexity of secondary polytopes. Adv Math 83:155–179
8. Buchberger B (1965) On finding a vector space basis of the residue class ring modulo a zero dimensional polynomial ideal. PhD Thesis, Univ. Innsbruck (In German.)
9. Conti P, Traverso C (1991) Buchberger algorithm and integer programming. In: Mattson HF Mora T, Rao TRN (eds) Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. Lecture Notes Computer Sci. Springer, Berlin, pp 130–139
10. Cornuejols G, Urbaniak R, Weismantel R, Wolsey L (1997) Decomposition of integer programs and of generating sets. In: Burkard R, Woeginger G (eds) Lecture Notes Computer Sci. Springer, Berlin, pp 92–103
11. Cox D, Little J, O'Shea D (1996) Ideals, varieties, and algorithms, 2nd edn. Springer, Berlin
12. Diaconis P, Graham R, Sturmfels B (1996) Primitive partition identities. In: Miklós D Sós VT, Szönyi T (eds) Combinatorics, Paul Erdös is Eighty. J. Bolyai Math. Soc., pp 173–192
13. Diaconis P, Sturmfels B (1998) Algebraic algorithms for sampling from conditional distributions. Ann Statist 26:363–397
14. Gel'fand IM, Kapranov M, Zelevinsky A (1994) Multidimensional determinants, discriminants and resultants. Birkhäuser, Basel
15. Gomory RE (1969) Some polyhedra related to combinatorial problems. Linear Alg Appl 2:451–558

16. Graver JE (1975) On the foundations of linear and integer programming I. Math Program 8:207–226
17. Hoşten S (1997) Degrees of Gröbner bases of integer programs. PhD Thesis, Cornell Univ.
18. Hoşten S, Sturmfels B (1995) GRIN: An implementation of Gröbner bases for integer programming. In: Balas E, Clausen J (eds) Integer Programming and Combinatorial Optimization. Lecture Notes Computer Sci. Springer, Berlin, pp 267–276
19. Hoşten S, Thomas RR (1999) The associated primes of initial ideals of lattice ideals. Math Res Lett 6:83–97
20. Hoşten S, Thomas RR (1999) Standard pairs and group relaxations in integer programming. J Pure Appl Algebra 139:133–157
21. Huber B, Thomas RR (1999) Computing Gröbner fans of toric ideals. Experimental Math (to appear)
22. Kannan R (1993) Optimal solution and value of parametric integer programs. In: Rinaldi G, Wolsey L (eds) Proc. 3rd IPCO Conf., pp 11–21
23. Loera J De, Sturmfels B, Thomas RR (1995) Gröbner bases and triangulations of the second hypersimplex. Combinatorica 15:409–424
24. Lovász L, Plummer MD (1986) Matching theory. North-Holland, Amsterdam
25. Mora T, Robbiano L (1988) The Gröbner fan of an ideal. J Symbolic Computation 6:183–208
26. Scarf HE (1986) Neighborhood systems for production sets with indivisibilities. Econometrica 54:507–532
27. Schrijver A (1986) Theory of linear and integer programming, Discrete Math and Optim. Wiley/Interscience, New York
28. Sturmfels B (1991) Gröbner bases of toric varieties. Tôhoku Math J 43:249–261
29. Sturmfels B (1995) Gröbner Bases and convex polytopes. Amer. Math. Soc., Providence, RI
30. Sturmfels B, Thomas RR (1997) Variation of cost functions in integer programming. Math Program 77:357–387
31. Sturmfels B, Trung N, Vogel W (1995) Bounds on projective schemes. Math Ann 302:417–432
32. Sturmfels B, Weismantel R, Ziegler G (1995) Gröbner bases of lattices, corner polyhedra and integer programming. Beiträge zur Algebra und Geometrie 36:281–298
33. Tayur SR, Thomas RR, Natraj NR (1995) An algebraic geometry algorithm for scheduling in the presence of setups and correlated demands. Math Program 69:369–401
34. Thomas RR (1995) A geometric Buchberger algorithm for integer programming. Math Oper Res 20:864–884
35. Thomas RR, Weismantel R (1997) Truncated Gröbner bases for integer programming. Applic Algebra in Engin, Communication and Computing 8:241–257
36. Urbaniak R, Weismantel R, Ziegler G (1997) A variant of Buchberger's algorithm for integer programming. SIAM J Discret Math 10:96–108
37. Wolsey L (1971) Extensions of the group theoretic approach in integer programming. Managem Sci 18:74–83

Integer Programming: Branch and Bound Methods

EVA K. LEE[1], JOHN E. MITCHELL[2]
[1] Industrial and Systems Engineering Georgia Institute Technol., Atlanta, USA
[2] Math. Sci. Rensselaer Polytechnic Institute, Troy, USA

MSC2000: 90C10, 90C05, 90C08, 90C11, 90C06

Article Outline

Keywords
Synonyms
Overview
Partitioning Strategies
Branching Variable Selection
Node Selection
Preprocessing and Reformulation
Heuristics
Continuous Reduced Cost Implications
Subproblem Solver
See also
References

Keywords

Branch and bound; Integer program; Exact algorithms

Synonyms

Branch and bound

Overview

An *integer programming problem* (IP) is an optimization problem in which some or all of the variables are restricted to take on only integer values. The exposition presented here will focus on the case in which the objective and constraints of the optimization problem are defined via linear functions. In addition, for simplicity, it will be assumed that all of the variables are restricted to be nonnegative integer valued. Thus, the mathematical formulation of the problem under consideration can be stated as:

$$(IP)\quad \begin{cases} \max & c^\top x \\ \text{s.t.} & Ax \le b \\ & x \in \mathbf{Z}^n_+, \end{cases}$$

where $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$ and $c \in \mathbf{R}^n$. For notational convenience, let S denote the *constraint set* of problem (IP); i. e.,

$$S := \left\{x \in \mathbf{Z}_+^n : \; Ax \leq b\right\}.$$

The classical approach to solving integer programs is branch and bound [39]. The branch and bound method is based on the idea of iteratively partitioning the set S (branching) to form subproblems of the original integer program. Each subproblem is solved – either exactly or approximately – to obtain an upper bound on the subproblem objective value. The driving force behind the branch and bound approach lies in the fact that if an upper bound for the objective value of a given subproblem is less than the objective value of a known integer feasible solution (e. g., obtained by solving some other subproblem) then the optimal solution of the original integer program cannot lie in the subset of S associated with the given subproblem. Hence, the upper bounds on subproblem objective values are, in essence, used to construct a proof of optimality without exhaustive search.

One concept that is fundamental to obtaining upper bounds on subproblem objective values is that of problem relaxation. A *relaxation* of the optimization problem

$$\max\left\{c^\top x : \; x \in S\right\}$$

is an optimization problem

$$\max\left\{c_R^\top x : \; x \in S_R\right\},$$

where $S \subseteq S_R$ and $c^\top x \leq c_R^\top x$ for all $x \in S$. Clearly, solving a problem relaxation provides an upper bound on the objective value of the underlying problem. Perhaps the most common relaxation of problem (IP) is the *linear programming relaxation* formed by relaxing the integer restrictions and enforcing appropriate bound conditions on the variables; i. e., $c_R = c$ and $S_R = \{\, x \in \mathbf{R}^n : Ax \leq b, l \leq x \leq u \,\}$.

A formal statement of a general branch and bound algorithm [48] is presented in Table 1. The notation L is used to denote the list of active subproblems $\{\mathrm{IP}^i\}$, where $\mathrm{IP}^0 = (\mathrm{IP})$ denotes the original integer program. The notation $\bar{z}_i$ denotes an upper bound on the optimal objective value of IP^i, and $\underline{z}_{ip}$ denotes the *incumbent objective value* (i. e., the objective value corresponding to the current best integral feasible solution to (IP)).

Integer Programming: Branch and Bound Methods, Table 1
General branch and bound algorithm

1	(*Initialization*): Set $L = \{\mathrm{IP}^0\}$, $\bar{z}_0 = +\infty$, and $\underline{z}_{ip} = \infty$.
2	(*Termination*): If $L = \emptyset$, then the solution x^* which yielded the incumbent objective value $\underline{z}_{ip}$ is optimal. If no such x^* exists (i.e., $\underline{z}_{ip} = -\infty$), then (IP) is infeasible.
3	(*Problem selection and relaxation*): Select and delete a problem IP^i from L. Solve a relaxation of IP^i. Let z_i^R denote the optimal objective value of the relaxation, and let x^{iR} be an optimal solution if one exists. (Thus, $z_i^R = c^\top x^{iR}$, or $z_i^R = -\infty$.)
4	(*Fathoming and Pruning*):
i)	If $z_i^R \leq \underline{z}_{ip}$ go to Step 2.
ii)	If $z_i^R > \underline{z}_{ip}$ and x^{iR} is integral feasible, update $\underline{z}_{ip} = z_i^R$. Delete from L all problems with $\bar{z}_i \leq \underline{z}_{ip}$. Go to Step 2.
5	(*Partitioning*): Let $\{S^{ij}\}_{j=1}^{j=k}$ be a partition of the constraint set S^i of the problem IP^i. Add problems $\{\mathrm{IP}^{ij}\}_{j=1}^{j=k}$ to L, where IP^{ij} is IP^i with feasible region restricted to S^{ij} and $\bar{z}_{ij} = z_i^R$ for $j = 1, \ldots, k$. Go to Step 2.

The actual implementation of a branch and bound algorithm is typically viewed as a *tree search*, where the problem at the root node of the tree is the original (IP). The tree is constructed in an iterative fashion with new nodes formed by *branching* on an existing node for which the optimal solution of the relaxation is fractional (i. e., some of the integer restricted variables have fractional values). Typically, two child nodes are formed by selecting a fractional valued variable and adding appropriate constraints in each child subproblem to ensure that the associated constraint sets do not include solutions for which this chosen branching variable assumes the same fractional value.

The phrase *fathoming a node* is used in reference to criteria that imply that a node need not be explored further. As indicated in Step 4, these criteria include:

a) the objective value of the subproblem relaxation at the node is less than or equal to the incumbent objective value; and

b) the solution for the subproblem relaxation is integer valued.

Note that a) includes the case when the relaxation is infeasible, since in that case its objective value is $-\infty$. Condition b) provides an opportunity to *prune* the tree; effectively fathoming nodes for which the objective value of the relaxation is less than or equal to the updated incumbent objective value. The tree search ends when all nodes are fathomed.

A variety of strategies have been proposed for intelligently selecting branching variables, for problem partitioning, and for selecting nodes to process. However, no single collection of strategies stands out as being best in all cases. In the remainder of this article, some of the strategies that have been implemented or proposed are summarized. An illustrative example is presented. Some of the related computational strategies – *preprocessing and reformulation*, *heuristic procedures*, and the concept of *reduced cost fixing* – which have proved to be highly effective in branch and bound implementations are considered. Finally, there is a discussion of recent linear programming based branch and bound algorithms that have employed *interior point methods* for the subproblem relaxation solver, which is in contrast to using the more traditional simplex-based solvers.

Though branch and bound is a classic approach for solving integer programs, there are practical limitations to its success in applications. Often integer feasible solutions are not readily available, and node pruning becomes impossible. In this case, branch and bound fails to find an optimal solution due to memory explosion as a result of excessive accumulation of active nodes. In fact, general integer programs are *NP-hard*; and consequently, as of this writing (1998), there exists no known *polynomial time algorithm* for solving general integer programs [30].

In 1983, a breakthrough in the computational possibilities of branch and bound came as a result of the research by H. Crowder, E.L. Johnson, and M.W. Padberg. In their paper [22], cutting planes were added at the root node to strengthen the LP formulation before branch and bound was called. In addition, features such as reduced cost fixing, heuristics and preprocessing were added within the tree search algorithm to facilitate the solution process. See ► Integer programming: Cutting plane algorithms for details on cutting plane applications to integer programming.

Since branch and bound itself is an inherently parallel technique, there has been active research activity among the computer science and operations research communities in developing parallel algorithms to improve its solution capability.

Most commercial integer programming solvers use a branch and bound algorithm with linear programming relaxations. Unless otherwise mentioned, the descriptions of the strategies discussed herein are based on using the linear programming relaxation.

See [48] for references not listed here; [51] also includes useful material about branch and bound.

Partitioning Strategies

When linear programming relaxation is employed, partitioning is done via addition of linear constraints. Typically, two new nodes are formed on each division. Suppose x^R is an optimal solution to the relaxation of a branch and bound node. Common partitioning strategies include:

- *Variable dichotomy* [23]. If x_j^R is fractional, then two new nodes are created, one with the simple bound $x_j \leq \lfloor x_j^R \rfloor$ and the other with $x_j \geq \lceil x_j^R \rceil$; where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the floor and the ceiling of a real number. In particular, if x_j is restricted to be binary, then the branching reduces to fixing $x_j = 0$ and $x_j = 1$, respectively. One advantage of simple bounds is that they maintain the size of the basis among branch and bound nodes, since the simplex method can be implemented to handle both upper and lower bounds on variables without explicitly increasing the dimensions of the basis.
- *Generalized-upper-bound dichotomy* (*GUB dichotomy*) [8]. If the constraint $\sum_{j \in Q} x_j = 1$ is present in the original integer program, and x_i^R, $i \in Q$, are fractional, one can partition $Q = Q_1 \cup Q_2$ such that $\sum_{j \in Q_1} x_j^R$ and $\sum_{j \in Q_2} x_j^R$ are approximately of equal value. Then two branches can be formed by setting $\sum_{j \in Q_1} x_j = 0$ and $\sum_{j \in Q_2} x_j = 0$, respectively.
- *Multiple branches for bounded integer variable*. If x_j^R is fractional, and $x_j \in \{0, \ldots, l\}$, then one can create $l + 1$ new nodes, with $x_j = k$ for node k, $k = 0, \ldots, l$. This idea was proposed in the first branch and

bound algorithm by A.H. Land and A.G. Doig [39], but currently (1998) is not commonly used.

Branching Variable Selection

During the partitioning process, branching variables must be selected to help create the children nodes. Clearly the choice of a branching variable affects the running time of the algorithm. Many different approaches have been developed and tested on different types of integer programs. Some common approaches are listed below:

- *Most/least infeasible integer variable*. In this approach, the integer variable whose fractional value is farthest from (closest to) an integral value is chosen as the branching variable.
- *Driebeck–Tomlin penalties* [25,57]. Penalties give a lower bound on the degradation of the objective value for branching each direction from a given variable. The penalties are the cost of the dual pivot needed to remove the fractional variable from the basis. If many pivots are required to restore primal feasibility, these penalties are not very informative. The *up penalty*, when forcing the value of the kth basic variable up, is

$$u_k = \min_{j:a_{kj}<0} \frac{(1-f_k)\bar{c}_j}{-a_{kj}},$$

where f_k is the fractional part of x_k, $\bar{c}_j$ is the reduced cost of variable x_j, and the a_{kj} are the transformed matrix coefficients from the kth row of the optimal dictionary for the LP relaxation. The *down penalty* d_k is calculated as

$$d_k = \min_{j:a_{kj}>0} \frac{f_k \bar{c}_j}{a_{kj}}.$$

Once the penalties have been computed, a variety of rules can be used to select the branching variable (e. g., $\max_k \max(u_k, d_k)$, or $\max_k \min(u_k, d_k)$). A penalty can be used to eliminate a branch if the LP objective value for the parent node minus the penalty is worse than the incumbent integer solution. Penalties are out of favor because their cost is considered too high for their benefit.

- *Pseudocost estimate*. Pseudocosts provide a way to estimate the degradation to the objective value by forcing a fractional variable to an integral value. The technique was introduced in 1970 by M. Benichou et al. [10]. Pseudocosts attempt to reflect the total cost, not just the cost of the first pivot, as with penalties. Once a variable x_k is labeled as a candidate branching variable, the pseudocosts are computed as:

$$U_k = \frac{\bar{z}_k - z_k^u}{1-f_k} \quad \text{and} \quad D_k = \frac{\bar{z}_k - z_k^d}{f_k},$$

where $\bar{z}_k$ is the objective value of the parent, z_k^u is the objective value resulting from forcing up, and z_k^d is the objective value from forcing down. (If the subproblem is infeasible, the associated pseudocost is not calculated.) If a variable has been branched upon repeatedly, an average may be used.
The branching variable is chosen as that with the maximum degradation, where the degradation is computed as: $D_k f_k + U_k (1 - f_k)$. Pseudocosts are not considered to be beneficial on problems where there is a large percentage of integer variables.
- *Pseudoshadow prices*. Similar to pseudocosts, pseudoshadow prices estimate the total cost to force a variable to an integral value. Up and down pseudoshadow prices for each constraint and pseudoshadow prices for each integer variable are specified by the user or given an initial value. The degradation in the objective function for forcing an integer variable x_k up or down to an integral value can be estimated. The branching variable is chosen using criteria similar to penalties and pseudocosts. See [27,40] for precise mathematical formulations on this approach.
- *Strong branching*. This branching strategy arose in connection with research on solving difficult instances of the traveling salesman problem and general mixed zero-one integer programming problems [2,12,13]. Applied to zero-one integer programs within a simplex-based branch and cut setting, strong branching works as follows. Let N and K be positive integers. Given the solution of some linear programming relaxation, make a list of N binary variables that are fractional and closest to 0.5 (if there are fewer than N fractional variables, take all fractional variables). Suppose that I is the index set of this list. Then, for each $i \in I$, fix x_i first to 0 and then to 1 and perform K iterations (starting with the *optimal basis* for the LP relaxation of the current node) of the dual simplex method with steepest edge pricing. Let L_i, U_i, $i \in I$, be the objective values

that result from these simplex runs, where L_i corresponds to fixing x_i to 0 and U_i to fixing it to 1. A branching variable can be selected based on the best weighted-sum of these two values.

- *Priorities selection*. Variables are selected based on their priorities. Priorities can be user-assigned, or based on objective function coefficients, or on pseudocosts.

Node Selection

Given a list of active problems, one has to decide which subproblem should be selected to be examined next. This in turn will affect the possibilities of improving the incumbent, the chance of node fathoming, and the total number of problems needed to be solved before optimality is achieved. Below, various strategies given in [7,10,11,20,27,29,31,47] are presented.

- *Depth-first search with backtracking*. Choose a child of the previous node as the next node; if it is pruned, choose the other child. If this node is also pruned, choose the most recently created unexplored node, which will be the other child node of the last successful node.
- *Best bound*. Among all unexplored nodes, choose the one which has the best LP objective value. In the case of maximization, the node with the largest LP objective value will be chosen. The rationale is that since nodes can only be pruned when the relaxation objective value is less than the current incumbent objective value, the node with largest LP objective value cannot be pruned, since the best objective value corresponding to an integer feasible solution cannot exceed this largest value.
- *Sum of integer infeasibilities*. The sum of infeasibilities at a node is calculated as

$$s = \sum_j \min(f_j, 1 - f_j).$$

Choose the node with either maximum or minimum sum of integer infeasibilities.

- *Best estimate using pseudocosts*. This technique was introduced [10] along with the idea of using pseudocosts to select a branching variable. The individual pseudocosts can be used to estimate the resulting integer objective value attainable from node k:

$$\epsilon_k = \overline{z}_k - \sum_i \min(D_i f_i, U_i(1 - f_i)),$$

where $\overline{z}_k$ is the value of the LP relaxation at node k. The node with the best estimate is chosen.

- *Best estimate using pseudoshadow prices*. Pseudoshadow prices can also be used to provide an estimate of the resulting integer objective value attainable from the node, and the node with the best estimate can then be chosen.
- *Best projection* [29,47]. Choose the node among all unexplored nodes which has the best projection. The projection is an estimate of the objective function value associated with an integer solution obtained by following the subtree starting at this node. It takes into account both the current objective function value and a measure of the integer infeasibility. In particular, the projection p_k associated with node k is defined as

$$p_k = \overline{z}_k - \frac{s_k(\overline{z}_0 - z_{ip})}{s_0},$$

where $\overline{z}_0$ denotes the objective value of the LP at the root node, z_{ip} denotes an estimate of the optimal integer solution, and s_k denotes the sum of the integer infeasibilities at node k.
The projection is a weighting between the objective function and the sum of infeasibilities. The weight $(\overline{z}_0 - z_{ip})/s_0$ corresponds to the slope of the line between node 0 and the node producing the optimal integer solution. It can be thought of as the cost to remove one unit of infeasibility. Let n_k be the number of integer infeasibilities at node k. A more general projection formula is to let $w_k = \mu\, n_k + (1 - \mu)\, s_k$, where $\mu \in [0, 1]$, and define

$$p_k = \overline{z}_k - \frac{w_k(\overline{z}_0 - z_{ip})}{w_0}.$$

Example 1 Here, a two-variable integer program is solved using branch and bound. The most infeasible integer variable is used as the branching variable, and best bound is used for node selection. Consider the problem

$$\mathrm{IP}^0 \quad \begin{cases} \max & 13x_1 + 8x_2 \\ \text{s.t.} & x_1 + 2x_2 \le 10 \\ & 5x_1 + 2x_2 \le 20 \\ & x_1 \ge 0, \quad x_2 \ge 0, \\ & x_1, x_2 \text{ integer}. \end{cases}$$

Initially, L consists of just this problem IP^0. The solution to the LP relaxation is $x_1^0 = 2.5, x_2^0 = 3.75$, with value $z_0^R = 59.5$. The most infeasible integer variable is x_1, so two new subproblems are created, IP^1 where $x_1 \geq 3$ and IP^2 where $x_1 \leq 2$, and $L = \{IP^1, IP^2\}$.

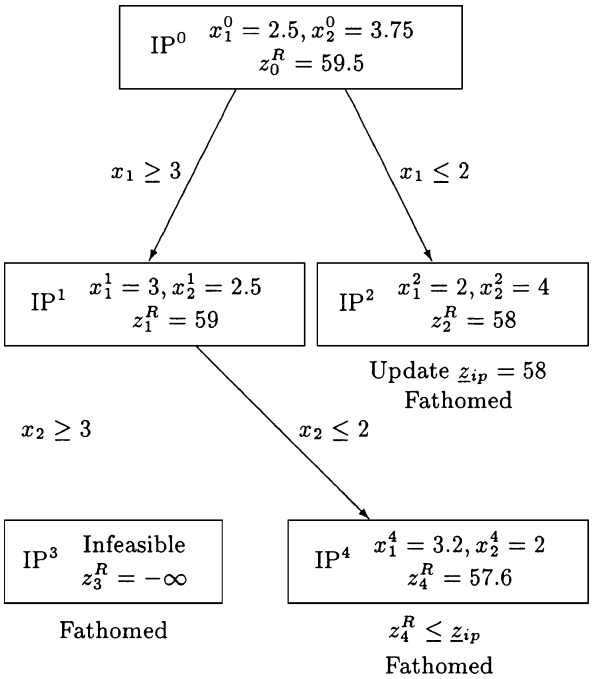
Both problems in L have the same bound 59.5, so assume the algorithm arbitrarily selects IP^1. The optimal solution to the LP relaxation of IP^1 is $x_1^1 = 3$, $x_2^1 = 2.5$, with value $z_1^R = 59$. The most infeasible integer variable is x_2, so two new subproblems of IP^1 are created, IP^3 where $x_2 \geq 3$ and IP^4 where $x_2 \leq 2$, and now $L = \{IP^2, IP^3, IP^4\}$.

The algorithm next examines IP^2, since this is the problem with the best bound. The optimal solution to the LP relaxation is $x_1^2 = 2$, $x_2^2 = 4$, with value $z_2^R = 58$. Since x^2 is integral feasible, $\underline{z}_{ip}$ can be updated to 58 and IP^2 is fathomed.

Both of the two problems remaining in L have best bound greater than 58, so neither can yet be fathomed. Since these two subproblems have the same bound 59, assume the algorithm arbitrarily selects IP^3 to examine next. The LP relaxation to this problem is infeasible, since it requires that x satisfy $x_1 \geq 3$, $x_2 \geq 3$ and $5x_1 + 2x_2 \leq 20$ simultaneously. Therefore, $z_3^R = -\infty$, and this node can be fathomed by bounds since $z_3^R \leq \underline{z}_{ip}$.

That leaves the single problem IP^4 in L. The solution to the LP relaxation of this problem is $x_1^4 = 3.2$, $x_2^4 = 2$, with value $z_4^R = 57.6$. Since $z_4^R \leq \underline{z}_{ip}$, this subproblem can also be fathomed by bounds. The set L is now empty, so x^2 is optimal for the integer programming problem IP^0.

The progress of the algorithm is indicated in Fig. 1. Each box contains the name of the subproblem, the solution to the LP relaxation, and the value of the solution.



Integer Programming: Branch and Bound Methods, Figure 1
Branch and bound example

Preprocessing and Reformulation

Problem preprocessing and reformulation has been shown to be a very effective way of improving integer programming formulations prior to and during branch and bound [14,15,18,19,22,24,33,34,35,54]. Below, some commonly employed preprocessing techniques are listed. For more details on these procedures, see the references.

1) Removal of empty (all zeros) rows and columns. Detection of implicit bounds and implicit slack variables.
2) Removal of rows dominated by multiples of other rows, including pairs of rows for which the support of one is a subset of the support of the other.
3) Strengthen the bounds within rows by comparing individual variables and coefficients to the right-hand side. Additional strengthening may be possible for integral variables using rounding.
4) Use variable bounds to determine upper and lower bounds for the left-hand side of a constraint, and compare these bounds to the right-hand side. Where possible, conclude that a constraint is inconsistent, redundant, or forces the fixing of some or all variables in its support. Several of these row-driven operations can be dualized to columns.
5) *Aggregation*: Given an equality constraint where the bound on some variable is implied by the satisfaction of the bounds on the other variables, this variable can be substituted out, and the constraint deleted. Note that free variables always satisfy this condition. Note also that in order to control fill-in (and coefficient growth), not all such substitutions may be desirable. For integral variables, there is the added restriction that they can be eliminated only if their integrality is implied by the integrality of

the remaining variables. For integer programming problems, an added advantage of aggregation relative to LP 's, is that the reduction in the number of equality constraints increases the relative dimension of the underlying polytope.

6) *Coefficient reduction*: Consider a constraint $\sum_{j \in K} a_j x_j \geq b$ in which all $a_j \geq 0$ and all $x_j \geq 0$. If x_j is a 0–1 variable and $a_j > b$, for some $j \in K$, replace a_j by b. A stronger version of this procedure is possible when the problem formulation involves other constraints of appropriate structure.
7) Logical implications and probing:
   a) *Logical implications*: Choose a binary variable x_k and fix it to 0 or 1. Perform 4. This analysis may yield logical implications such as $x_k = 1$ implies $x_j = 0$, or $x_k = 1$ implies $x_j = 1$, for some other variable x_j. The implied equality is then added as an explicit constraint.
   b) *Probing*: Perform logical implications recursively. An efficient implementation of probing appears to be very difficult. Details of computational issues regarding probing are discussed in [33], and [54].

Heuristics

Heuristic procedures provide a means for obtaining integer feasible solutions quickly, and can be used repeatedly within the branch and bound search tree. A good heuristic – one that produces good integer feasible solutions – is a crucial component in the branch and bound algorithm since it provides an upper bound for reduced cost fixing (discussed later) at the root, and thus allows reduction in the size of the linear program that must be solved. This in turn may reduce the time required to solve subsequent linear programs at nodes within the search tree. In addition, a good upper bound increases the likelihood of being able to fathom active nodes, which is extremely important when solving large scale integer programs as they tend to create many active nodes leading to memory explosion.

Broadly speaking, five ideas are commonly used in developing heuristics. The first idea is that of greediness. Greedy algorithms work by successively choosing variables based on best improvement in the objective value. Kruskal's algorithm [37], which is an exact algorithm for finding the minimum-weight spanning tree in a graph, is one of the most well-known greedy algorithms. Greedy algorithms have been applied to a variety of problems, including 0–1 knapsack problems [36,41,53], uncapacitated facility location problems [38,56], set covering problems [3,4], and the traveling salesman problem [52].

A second idea is that of local search, which involves searching in a local neighborhood of a given integer feasible solution for a feasible solution with a better objective value. The k-interchange heuristic is a classic example of a local search heuristic [38,44,46]. Simulated annealing is another example, but with a bit of a twist. It allows, with a certain probability, updated solutions with less favorable objective values in order to increase the likelihood of escaping from a local optimum [16].

Randomized enumeration is a third idea that is used to obtain integer feasible solutions. One such method is that of genetic algorithms (cf. ► Genetic algorithms), where the randomness is modeled on the biological mechanisms of evolution and natural selection [32]. Recent work on applying a genetic algorithm to the set covering problem can be found in [9].

The term *primal heuristics* refers to certain LP-based procedures for constructing integral feasible solutions from points that are in some sense good, but fail to satisfy integrality. Typically, these nonintegral points are obtained as optimal solutions of LP relaxations. Primal heuristic procedures involve successive variable fixing and rounding (according to rules usually governed by problem structure) and subsequent re-solves of the modified primal LP [6,12,14,34,35].

The fifth general principle is that of *exploiting the interplay between primal and dual solutions*. For example, an optimal or heuristic solution to the dual of an LP relaxation may be used to construct a heuristic solution for the primal (IP). Problem dependent criteria based on the generated primal-dual pair may suggest seeking an alternative heuristic solution to the dual, which would then be used to construct a new heuristic solution to the primal. Iterating back-and-forth between primal and dual heuristic solutions would continue until an appropriate termination condition is satisfied [21,26,28].

It is not uncommon that a heuristic involves more than one of these ideas. For example, pivot-and-complement is a simplex-based heuristic in which binary variables in the basis are pivoted out and replaced by slack variables. When a feasible integer solution is

obtained, the algorithm performs a local search in an attempt to obtain a better integer feasible solution [5]. Obviously, within a branch and bound implementation, the structure of the problems that the implementation is targeted at influences the design of an effective heuristic [2,12,13,14,22,26,34,35,43].

Continuous Reduced Cost Implications

Reduced cost fixing is a well-known and important idea in the literature of integer programming [22]. Given an optimal solution to an LP relaxation, the reduced costs $\bar{c}_j$ are nonpositive for all nonbasic variables x_j at lower bound, and nonnegative for all nonbasic variables at their upper bounds. Let x_j be a nonbasic variable in a continuous optimal solution having objective value z_{LP}, and let $\underline{z}_{ip}$ be the objective value associated with an integer feasible solution to (IP). The following are true:

a) If x_j is at its lower bound in the continuous solution and $z_{LP} - \underline{z}_{ip} \leq -\bar{c}_j$, then there exists an optimal solution to the integer program with x_j at its lower bound.
b) If x_j is at its upper bound in the continuous solution and $z_{LP} - \underline{z}_{ip} \leq \bar{c}_j$, then there exists an optimal solution to the integer program with x_j at its upper bound.

When reduced cost fixing is applied to the root node of a branch and bound tree, variables which are fixed can be removed from the problem, resulting in a reduction in the size of the integer program. A variety of studies have examined the effectiveness of reduced cost fixing within the branch and bound tree search [12,14,22,34,35,49,50].

Subproblem Solver

When linear programs are employed as the relaxations within a branch and bound algorithm, it is common to use a simplex-based algorithm to solve each subproblem, using *dual simplex* to reoptimize from the optimal basis of the parent node. This technique of *advanced basis* has been shown to reduce the number of simplex iterations to solve the child node to optimality, and thus speedup the overall computational effort. Recently with the advancement in computational technology, the increase in the size of integer programs, and the success of interior point methods (cf. also ► Linear programming: Interior point methods) to solve large scale linear programs [1,45] there are some branch and bound algorithms employing interior point algorithms as the linear programming solver [17,42,43,55]. In this case, advanced basis is no longer available and care has to be taken to take advantage of warmstart vectors for the interior point solver so as to facilitate effective computational results. In [42,43], a description of the ideas of *'advanced warmstart'* and computational results are presented.

See also

- ► Branch and Price: Integer Programming with Column Generation
- ► Decomposition Techniques for MILP: Lagrangian Relaxation
- ► Integer Linear Complementary Problem
- ► Integer Programming
- ► Integer Programming: Algebraic Methods
- ► Integer Programming: Branch and Cut Algorithms
- ► Integer Programming: Cutting Plane Algorithms
- ► Integer Programming Duality
- ► Integer Programming: Lagrangian Relaxation
- ► LCP: Pardalos-Rosen Mixed Integer Formulation
- ► Mixed Integer Classification Problems
- ► Multi-objective Integer Linear Programming
- ► Multi-objective Mixed Integer Programming
- ► Multiparametric Mixed Integer Linear Programming
- ► Parametric Mixed Integer Nonlinear Optimization
- ► Set Covering, Packing and Partitioning Problems
- ► Simplicial Pivoting Algorithms for Integer Programming
- ► Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ► Stochastic Integer Programs
- ► Time-dependent Traveling Salesman Problem

References

1. Andersen ED, Gondzio J, Mészáros C, Xu X (1996) Implementation of interior point methods for large scale linear programming. In: Terlaky T (ed) Interior Point Methods in Mathematical Programming. Kluwer, Dordrecht, ftp://ftp.sztaki.hu/pub/oplab/PAPERS/kluwer.ps.Z
2. Applegate D, Bixby RE, Chvátal V, Cook W (1998) On the solution of travelling salesman problems. Documenta Math no. Extra Vol. Proc. ICM III:645–656
3. Baker EK (1981) Efficient heuristic algorithms for the weighted set covering problem. Comput Oper Res 8:303–310

4. Baker EK, Fisher ML (1981) Computational results for very large air crew scheduling problems. OMEGA Internat J Management Sci 19:613–618
5. Balas E, Martin CH (1980) Pivot and complement: A heuristic for 0/1 programming. Managem Sci 26:86–96
6. Baldick R (1992) A randomized heuristic for inequality-constrained mixed-integer programming. Techn Report Dept Electrical and Computer Engin Worcester Polytechnic Inst
7. Beale EML (1979) Branch and bound methods for mathematical programming systems. Ann Discret Math 5:201–219
8. Beale EML, Tomlin JA (1970) Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables. In: Lawerence J (ed) Proc. Fifth Internat. Conf. Oper. Res., Tavistock Publ., pp 447–454
9. Beasley JE, Chu PC (1996) A genetic algorithm for the set covering problem. Europ J Oper Res 194:392–404
10. Benichou M, Gauthier JM, Girodet P, Hehntges G, Ribiere G, Vincent O (1971) Experiments in mixed integer linear programming. Math Program 1:76–94
11. Benichou M, Gauthier JM, Hehntges G, Ribiere G (1977) The efficient solution of large-scale linear programming problems - some algorithmic techniques and computational results. Math Program 13:280–322
12. Bixby RE, Cook W, Cox A, Lee EK (1995) Parallel mixed integer programming. Techn Report Center Res Parallel Computation, Rice Univ CRPC-TR95554
13. Bixby RE, Cook W, Cox A, Lee EK (1999) Computational experience with parallel mixed integer programming in a distributed environment. Ann Oper Res 90:19–43
14. Bixby RE, Lee EK (1998) Solving a truck dispatching scheduling problem using branch-and-cut. Oper Res 46:355–367
15. Bixby RE, Wagner DK (1987) A note on detecting simple redundancies in linear systems. Oper Res Lett 6:15–18
16. Bonomi E, Lutton JL (1984) The N-city traveling salesman problem: Statistical mechanics and the metropolis algorithm. SIAM Rev 26:551–568
17. Borchers B, Mitchell JE (March 1991) Using an interior point method in a branch and bound algorithm for integer programming. Techn Report Math Sci Rensselaer Polytech Inst 195
18. Bradley GH, Hammer PL, Wolsey L (1975) Coefficient reduction in 0–1 variables. Math Program 7:263–282
19. Brearley AL, Mitra G, Williams HP (1975) Analysis of mathematical programming problems prior to applying the simplex method. Math Program 5:54–83
20. Breu R, Burdet CA (1974) Branch and bound experiments in zero-one programming. Math Program 2:1–50
21. Conn AR, Cornuejols G (1987) A projection method for the uncapacitated facility location problem. Techn Report Graduate School Industr Admin Carnegie-Mellon Univ 26-86-87
22. Crowder H, Johnson EL, Padberg M (1983) Solving large-scale zero-one linear programming problem. Oper Res 31:803–834
23. Dakin RJ (1965) A tree search algorithm for mixed integer programming problems. Comput J 8:250–255
24. Dietrich B, Escudero L (1990) Coefficient reduction for knapsack-like constraints in 0/1 programs with variable upper bounds. Oper Res Lett 9:9–14
25. Driebeek NJ (1966) An algorithm for the solution of mixed integer programming problems. Managem Sci 21:576–587
26. Erlenkotter D (1978) A dual-based procedure for uncapacitated facility location. Oper Res 26:992–1009
27. Fenelon M (1991) Branching strategies for MIP. CPLEX
28. Fisher ML, Jaikumer R (1981) A generalized assignment heuristic for vehicle routing. Networks 11:109–124
29. Forrest JJ, Hirst JPH, Tomlin JA (1974) Practical solution of large mixed integer programming problems with UMPIRE. Managem Sci 20:736–773
30. Garey MR, Johnson DS (1979) Computers and intractability – A guide to the theory of NP-completeness. Freeman, New York
31. Gauthier JM, Ribiere G (1977) Experiments in mixed integer programming using pseudo-costs. Math Program 12:26–47
32. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA
33. Guignard M, Spielberg K (1981) Logical reduction methods in zero-one programming. Oper Res 29:49–74
34. Hoffman KL, Padberg M (1991) Improving LP-representations of zero-one linear programs for branch-and-cut. ORSA J Comput 3:121–134
35. Hoffman KL, Padberg M (1992) Solving airline crew-scheduling problems by branch-and-cut. Managem Sci 39:657–682
36. Ibarra OH, Kim CE (1975) Fast approximation algorithms for the knapsack and sum of subset problems. J ACM 22:463–468
37. Kruskal JB (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. Proc Amer Math Soc 7:48–50
38. Kuehn AA, Hamburger MJ (1963) A heuristic program for locating warehouses. Managem Sci 19:643–666
39. Land AH, Doig AG (1960) An automatic method for solving discrete programming problems. Econometrica 28:497–520
40. Land AH, Powell S (1979) Computer codes for problems of integer programming. Ann Discret Math 5:221–269
41. Lawler EL (1979) Fast approximation algorithms for the knapsack problems. Math Oper Res 4:339–356
42. Lee EK, Mitchell JE (1996) Computational experience in nonlinear mixed integer programming. In: The Oper. Res. Proc. 1996. Springer, Berlin, pp 95–100
43. Lee EK, Mitchell JE (2000) Computational experience of an interior-point SQP algorithm in a parallel branch-and-

bound framework. In: Frenk H et al (eds.) High Performance Optimization. Kluwer, Dordrecht, pp 329–347 (Chap. 13).
44. Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling salesman problem. Oper Res 21:498–516
45. Lustig IJ, Marsten RE, Shanno DF (1994) Interior point methods for linear programming: Computational state of the art. ORSA J Comput 6(1):1–14. see also the following commentaries and rejoinder
46. Manne AS (1964) Plant location under economies of scale-decentralization and computation. Managem Sci 11:213–235
47. Mitra G (1973) Investigations of some branch and bound strategies for the solution of mixed integer linear programs. Math Program 4:155–170
48. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
49. Padberg M, Rinaldi G (1989) A branch-and-cut approach to a traveling salesman problem with side constraints. Managem Sci 35:1393–1412
50. Padberg M, Rinaldi G (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Rev 33:60–100
51. Parker RG, Rardin RL (1988) Discrete optimization. Acad. Press, New York
52. Rosenkrantz DJ, Stearns RE, Lewis PM (1977) An analysis of several heuristics for the traveling salesman problem. SIAM J Comput 6:563–581
53. Sahni S (1975) Approximate algorithms for the 0–1 knapsack problem. J ACM 22:115–124
54. Savelsbergh MWP (1994) Preprocessing and probing for mixed integer programming problems. ORSA J Comput 6:445–454
55. de Silva A, Abramson D (1998) A parallel interior point method and its application to facility location problems. Comput Optim Appl 9:249–273
56. Spielberg K (1969) Algorithms for the simple plant location problem with some side-conditions. Oper Res 17:85–111
57. Tomlin JA (1971) An improved branch and bound method for integer programming. Oper Res 19:1070–1075

Integer Programming: Branch and Cut Algorithms

Branch and Cut

JOHN E. MITCHELL
Math. Sci. Rensselaer Polytechnic Institute, Troy, USA

MSC2000: 90C10, 90C11, 90C05, 90C08, 90C06

Article Outline

Keywords
A Standard Form
Primal Heuristics
Preprocessing
Families of Cutting Planes
When to Add Cutting Planes
Lifting Cuts
Implementation Details
Solving Large Problems
Conclusions
See also
References

Keywords

Cutting planes; Branch and bound; Integer program; Exact algorithms

Branch and cut methods are exact algorithms for *integer programming problems*. They consist of a combination of a cutting plane method (cf. ► Integer programming: Cutting plane algorithms) with a branch and bound algorithm (cf. ► Integer programming: Branch and bound methods). These methods work by solving a sequence of linear programming relaxations of the integer programming problem. Cutting plane methods improve the relaxation of the problem to more closely approximate the integer programming problem, and branch and bound algorithms proceed by a sophisticated divide-and-conquer approach to solve problems. The material in this entry builds on the material contained in the entries on cutting plane and branch and bound methods.

Perhaps the best known branch and cut algorithms are those that have been used to solve *traveling salesman problems*. This approach is able to solve and prove optimality of far larger instances than other methods. Two papers that describe some of this research and also serve as good introductions to the area of branch and cut algorithms are [21,32]. A more recent work on the branch and cut approach to the traveling salesman problem is [1]. Branch and cut methods have also been used to solve other combinatorial optimization problems; recent references include [8,10,13,23,24,26]. For these problems, the cutting planes are typically derived

from studies of the polyhedral combinatorics of the corresponding integer program. This enables the addition of strong cutting planes (usually facet defining inequalities), which make it possible to considerably reduce the size of the branch and bound tree. Far more detail about these strong cutting planes can be found in ▶ Integer programming: Cutting plane algorithms.

Branch and cut methods for general integer programming problems are also of great interest (see, for example, the papers [4,7,11,16,17,22,28,30]). It is usually not possible to efficiently solve a general integer programming problem using just a cutting plane approach, and it is therefore necessary to also to branch, resulting in a branch and cut approach. A pure branch and bound approach can be sped up considerably by the employment of a cutting plane scheme, either just at the top of the tree, or at every node of the tree.

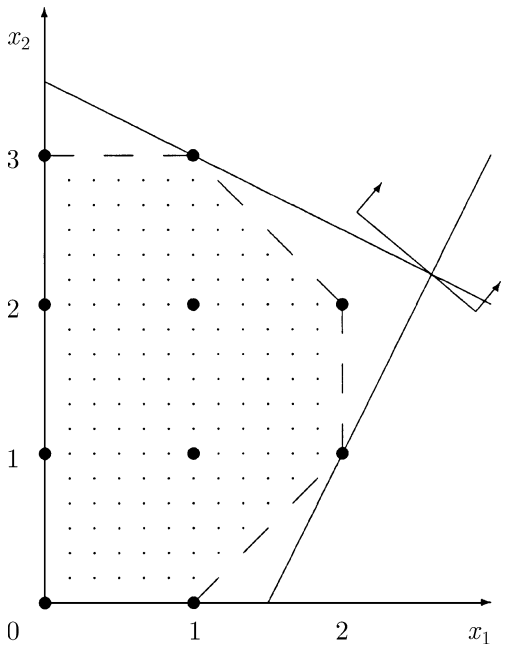
For general problems, the specialized facets used when solving a specific combinatorial optimization problem are not available. Some useful families of general inequalities have been developed; these include cuts based on knapsack problems [17,22,23], Gomory cutting planes [5,12,19,20], lift and project cutting planes [3,4,29,33], and Fenchel cutting planes [9]. All of these families of cutting planes are discussed in more detail later in this entry.

The software packages MINTO [30] and ABACUS [28] implement branch and cut algorithms to solve integer programming problems. The packages use standard linear programming solvers to solve the relaxations and they have a default implementation available. They also offer the user many options, including how to add cutting planes and how to branch.

Example 1 Consider the integer programming problem

$$\begin{cases} \min & -5x_1 - 6x_2 \\ \text{s.t.} & x_1 + 2x_2 \le 7 \\ & 2x_1 - x_2 \le 3 \\ & x_1, x_2 \ge 0 \quad \text{and integer.} \end{cases}$$

This problem is illustrated in Fig. 1. The feasible integer points are indicated. The *linear programming relaxation* (or *LP relaxation*) is obtained by ignoring the integrality restrictions; this is given by the polyhedron contained in the solid lines.



Integer Programming: Branch and Cut Algorithms, Figure 1
A branch-and-cut example

The first step in a branch and cut approach is to solve the linear programming relaxation, which gives the point (2.6, 2.2), with value − 26.2. There is now a choice: should the LP relaxation be improved by adding a cutting plane, for example, $x_1 + x_2 \le 4$, or should the problem be divided into two by splitting on a variable?

Assume the algorithm makes the second choice, and further assume that the decision is to split on x_2, giving two new problems:

$$\begin{cases} \min & -5x_1 - 6x_2 \\ \text{s.t.} & x_1 + 2x_2 \le 7 \\ & 2x_1 - x_2 \le 3 \\ & \mathbf{x}_2 \ge 3 \\ & x_1, x_2 \ge 0 \quad \text{and integer,} \end{cases}$$

and

$$\begin{cases} \min & -5x_1 - 6x_2 \\ \text{s.t.} & x_1 + 2x_2 \le 7 \\ & 2x_1 - x_2 \le 3 \\ & \mathbf{x}_2 \le 2 \\ & x_1, x_2 \ge 0 \quad \text{and integer.} \end{cases}$$

The optimal solution to the original problem will be the better of the solutions to these two subproblems. The solution to the linear programming relaxation of the first subproblem is (1, 3), with value − 23. Since this solution is integral, it solves the first subproblem. This solution becomes the incumbent best known feasible solution. The optimal solution for the linear programming relaxation of the second subproblem is (2.5, 2), with value − 24.5. Since this point is nonintegral, it does not solve the subproblem. Therefore, the second subproblem must be attacked further.

It is possible to branch using x_1 in the second subproblem; instead, assume the algorithm uses a cutting plane approach and adds the inequality $x_1 + 2x_2 \leq 6$. This is a valid inequality, in that it is satisfied by every integral point that is feasible in the second subproblem. Further, this inequality is a cutting plane: it is violated by (2.5, 2). Adding this inequality to the relaxation and resolving gives the optimal solution (2.4, 1.8), with value − 22.6. The subproblem still does not have an integral solution. However, notice that the optimal value for this modified relaxation is larger than the value of the incumbent solution. The value of the optimal integral solution to the second subproblem must be at least as large. Therefore, the incumbent solution is better than any feasible integral solution for the second subproblem, so it actually solves the original problem.

Of course, there are several issues to be resolved with this algorithm, including the major questions of deciding whether to branch or to cut and deciding how to branch and how to generate cutting planes. Notice that the cutting plane introduced in the second subproblem is not valid for the first subproblem. This inequality can be modified to make it valid for the first subproblem by using a *lifting* technique, which is discussed later in this entry.

A Standard Form

To fix notation, the following problem is regarded as the standard form mixed integer linear programming problem in this entry:

$$\begin{cases} \min & c^\top x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \\ & x_i \text{ integer}, \quad i = 1, \ldots, p. \end{cases}$$

Here, x and c are n-vectors, b is an m-vector, and A is an $m \times n$ matrix. The first p variables are restricted to be integer, and the remainder may be fractional. If $p = n$ then this is an integer programming problem. If a variable is restricted to take the values 0 or 1 then it is a binary variable. If all variables are binary then the problem is a binary program.

Primal Heuristics

In the example problem, it was possible to prune the second subproblem by bounds, once an appropriate cutting plane had been found. The existence of a good incumbent solution made it possible to prune in this way. In this case, the solution to the linear programming relaxation of the first subproblem was integral, providing the good incumbent solution. In many cases, it takes many stages until the solution to a relaxation is integral. Therefore, it is often useful to have good heuristics for converting the fractional solution to a relaxation into a good integral solution that can be used to prune other subproblems. Primal heuristics are discussed further in ► Integer programming: Branch and bound methods.

Preprocessing

A very important component of a practical branch and cut algorithm is preprocessing to eliminate unnecessary constraints, determine any fixed variables, and simplify the problem in other ways. Preprocessing techniques are discussed in ► Integer programming: Branch and bound methods.

Families of Cutting Planes

Perhaps the first family of cutting planes for general mixed integer programming problems were *Chvátal-Gomory cutting planes* [15,19,20]. These inequalities can be derived from the final tableau of the linear programming relaxation, and they are discussed in more detail in ► Integer programming: Cutting plane algorithms. These cuts can be useful if they are applied in a computationally efficient manner [5,12]. Gomory cuts can contain a large number of nonzeros, so care is required to ensure that the LP relaxation does not become very hard with large memory requirements. The cuts are generated directly from the basis inverse, so

care must also be taken to avoid numerical difficulties.

One of the breakthroughs in the development of branch and cut algorithms was the paper by H.P. Crowder, E.L. Johnson, and M.W. Padberg [17]. This paper showed that it was possible to solve far larger general problems than had previously been thought possible. The algorithm used extensive preprocessing, good primal heuristics, and cutting planes derived from *knapsack problems* with binary variables. Any inequality in binary variables can be represented as a knapsack inequality $\sum_{i \in N} a_i x_i \leq b$ with all $a_i > 0$ for some subset N of the variables, eliminating variables or replacing a variable x_j by $1 - x_j$ as necessary. The facial structure of the knapsack polytope can then be used to derive valid inequalities for the problem. For example, if $R \subseteq N$ with $\sum_{i \in R} a_i > b$ then $\sum_{i \in R} x_i \leq |R| - 1$ is a valid inequality. Further, if R is a minimal such set, so deleting any member of R leaves the sum of coefficients smaller than b, then the inequality defines a facet of the corresponding knapsack polytope. Other inequalities can be derived from the knapsack polytope. These inequalities have been extended to knapsacks with general integer variables and one continuous variable [11] and to binary problems with generalized upper bounds [34].

Another family of useful inequalities are *lift-and-project* or *disjunctive inequalities*. These were originally introduced by E. Balas [2], and it is only in the last few years that the value of these cuts has become apparent for general integer programming problems [3,4]. Given the feasible region for a binary programming problem $S := \{ x: Ax \leq b, x_i = 0, 1, \forall i \}$, each variable can be used to generate a set of disjunctive inequalities. Let $S_j^0 := \{ x: Ax \leq b, 0 \leq x_i \leq 1, \forall i, x_j = 0 \}$ and $S_j^1 := \{x: Ax \leq b, 0 \leq x_i \leq 1, \forall i, x_j = 1 \}$. Then $S \subseteq S_j^0 \cup S_j^1$, so valid inequalities for S can be generated by finding valid inequalities for the convex hull of $S_j^0 \cup S_j^1$. These inequalities are generated by solving linear programming problems. Because of the expense, the cuts are usually only generated at the root node. Nonetheless, they can be very effective computationally.

Other general cutting planes have been developed. The paper[16] describes several families and discusses routines for identifying violated inequalities. Fenchel cutting planes, which are generated using ideas from Lagrangian duality and convex duality, are introduced in [9].

When to Add Cutting Planes

The computational overhead of searching for cutting planes can be prohibitive. Therefore, it is common to not search at every node of the tree. Alternatives include searching at every eighth node, say, or at every node at a depth of a multiple of eight in the tree.

Generally, at each node of the branch and bound tree, the linear programming relaxation is solved, cutting planes are found, these are added to the relaxation, and the process is repeated. Usually, there comes a point at which the process tails off, that is, the solution to one relaxation is not much better than the solutions to the recent relaxations. It is then advisable to stop work on this node and branch. Tailing off is a function of lack of knowledge about the polyhedral structure of the relaxation, rather than a fundamental weakness of the cutting plane approach [32]. In some implementations, a fixed number of rounds of cutting plane searching are performed at a node, with perhaps several rounds performed at the root node, and fewer rounds performed lower in the tree.

The *cut-and-branch* variant adds cutting planes only at the root node of the tree. Usually, an implementation of such a method will expend a great deal of effort on generating cutting planes, requiring time far greater than just solving the relaxation at the root. The benefits of cut-and-branch include

- all generated cuts are valid throughout the tree, since they are valid at the root.
- bookkeeping is reduced, since the relaxations are identical at each node.
- no time is spent generating cutting planes at other nodes.

Cut-and-branch is an excellent technique for many general integer programs, but it lacks the power of branch and cut for some hard problems. See [16] for more discussion of the relative computational performance of cut-and-branch and branch and cut.

Lifting Cuts

A cut added at one node of the branch and cut tree may well not be valid for another subproblem. Of course, it is not necessary to add the cut at any other node, in which case the cut is called a *local cut*. This cut will then only affect the current subproblem and its descendants. The drawback to such an approach is in the potential mem-

ory requirement of needing to store a different version of the problem for each node of the tree. In order to make a cut valid throughout the tree (or *global*), it is necessary to *lift* it.

Lifting can be regarded as a method of rotating a constraint. Returning to the example problem once again, the constraint $x_1 + 2x_2 \leq 6$ is valid if $x_2 \leq 2$. To extend this constraint to be valid when $x_2 \geq 3$, consider the inequality

$$x_1 + 2x_2 + \alpha(x_2 - 2) \leq 6.$$

It is desired to take α as large as possible while ensuring that this is a valid inequality. If $x_2 = 3$ then $x_1 \leq 1$, so the inequality is valid for $x_2 = 3$ provided $\alpha \leq -1$. If $x_2 = 1$, the inequality is valid provided $\alpha \geq -2$. Finally, the inequality is valid when $x_2 = 0$ provided $\alpha \geq -2.5$. Combining these conditions gives that the valid range is $-2 \leq \alpha \leq -1$. The two extreme choices $\alpha = -1$ and $\alpha = -2$ give the valid inequalities $x_1 + x_2 \leq 4$ and $x_1 \leq 2$, respectively. Other valid choices for α give inequalities that are convex combinations of these two. In this way, valid inequalities for one node of the tree can be extended to valid inequalities at any node.

See [11] for more discussion of lifting in the case of general mixed integer linear programming problems. It is often not possible to lift inequalities for such problems because the upper and lower bounds on the coefficients conflict. Of course, if an inequality is valid at the root node then it is valid throughout the tree so there is no need to lift. This is one of the reasons why general inequalities such as Chv átal-Gomory cuts or lift-and-project cuts are often more successfully employed in a cut-and-branch approach.

The method of calculating coefficients in the case of binary problems is now outlined – see [31] for more details. The inequality generated at a node in the tree will generally only use the variables that are not fixed at that node. Lifting can be used to make the inequality valid at any node of the tree. It is necessary to apply the lifting process for each variable that has been fixed at the node, examining the opposite value for that variable. For example, if the inequality

$$\sum_{j \in J} a_j x_j \leq h \quad \text{for some subset } J \subseteq \{1, \ldots, n\}$$

is valid at a node where x_i has been fixed to zero, the lifted inequality takes the form

$$\sum_{j \in J} a_j x_j + \alpha_i x_i \leq h$$

for some scalar α_i. This scalar should be maximized in order to make the inequality as strong as possible. Now, maximizing α_i requires solving another integer program, so it may be necessary to make an approximation. This process has to be applied successively to each variable that has been fixed at the node. The order in which the variables are examined may well affect the final inequality, and other valid inequalities can be obtained by lifting more than one variable at a time.

Implementation Details

Many details of tree management can be found in ► Integer programming: Branch and bound methods. This includes node selection, branching variable selection, and storage requirements, among other issues. Typically, a branch and bound algorithm stores the solution to a node as a list of the indices of the basic variables. Branch and cut algorithms may require more storage if cuts are added locally, because it is then necessary to be able to recreate the current relaxation at any active node with just the appropriate constraints. If cuts are added globally, then it suffices to store a single representation of the problem.

It is possible to fix variables using information about reduced costs and the value of the best known feasible integral solution, as described in ► Integer programming: Cutting plane algorithms. Once variables have been fixed in this way, it is often possible to fix additional variables using logical implications. In order to fully exploit the fixing of variables, *parent node reconstruction* [32] is performed as follows. Once a parent node has been selected, it is not immediately divided into two children, but is solved again using the cutting plane algorithm. When the cutting plane procedure terminates, the optimal reduced cost vector has been reconstructed and this is used to perform variable fixing.

Many branch and cut implementations use a *pool of cuts* [32]. This is typically a set of constraints that have been generated earlier and either not included in the relaxation or subsequently dropped because they no longer appeared to be active. It is easy to check these

cuts for violation and this is usually done before more involved separation routines are invoked. The pool of cuts also makes it possible to reconstruct the parent node more efficiently, partly because difficulties with tailing off are reduced.

Solving Large Problems

The difficulty of a particular integer programming problem is not purely a function of the size of the problem. There are problems in the MIPLIB test set [6] with just a few hundred variables that prove resistant to standard solution approaches. The difficulty is caused by an explosion in the size of the tree.

For some problems, difficulties are caused by the size of the LP relaxation, and *interior point methods* may be useful in such cases. Interior point methods are superior to simplex methods for many linear programming problems with thousands of variables. However, restarting is harder with an interior point method than with a simplex method when the relaxation is only slightly changed. Therefore, for very large problems, the first relaxation at the top node of the tree can be solved using an interior point method, and subsequent relaxations can be solved using the (dual) simplex method. For some problems, the relaxations are just too large to be handled with a simplex method. For example, the relaxations of the *quadratic assignment problem* given in [25] were solved using interior point methods. Interior point methods also handle degeneracy better than the simplex method. Therefore, the branch and cut solver described in [1] occasionally uses an interior point method to handle some subproblems.

One way to enable the solution of far larger problems is to use a *parallel computer*. The nature of branch and cut and branch and bound algorithms makes it possible for them to exploit coarse grain parallel computers efficiently: typically, a linear programming relaxation is solved on a node of the computer. It is possible to use one node to manage the distribution of linear programs to nodes. Alternatively, methods have been developed where a common data structure is maintained and all nodes access this data structure to obtain a relaxation that requires solution, for example [18]. For a discussion of parallel branch and cut algorithms, see [7,27]. It is also possible to generate cutting planes in parallel; see, for example, [14].

Conclusions

Branch and cut methods have been successfully used to solve both specialized integer programming problems such as the traveling salesman problem and vehicle scheduling, and also general integer programming problems. In both cases, these methods are the most promising techniques available for proving optimality. For specialized problems, cutting planes are derived using the polyhedral theory of the underlying problem. For general mixed integer linear programming problems, important components of an efficient implementation include preprocessing, primal heuristics, routines for generating cutting planes such as lift-and-project or Gomory 's rounding procedure or cuts derived from knapsack problems, and also routines for lifting constraints to strengthen them. This is an active research area, with refinements and developments being continuously discovered.

See also

- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos-Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem

References

1. Applegate D, Bixby RE, Chvátal V, Cook W (1998) On the solution of travelling salesman problems. Documenta Math no. Extra Vol. Proc. ICM III:645–656
2. Balas E (1971) Intersection cuts - a new type of cutting planes for integer programming. Oper Res 19:19–39
3. Balas E, Ceria S, Cornuéjols G (1993) A lift-and-project cutting plane algorithm for mixed 0–1 programs. Math Program 58:295–324
4. Balas E, Ceria S, Cornuéjols G (1996) Mixed 0–1 programming by lift-and-project in a branch-and-cut framework. Managem Sci 42:1229–1246ftp: cumparsita.gsb.columbia.edu
5. Balas E, Ceria S, Cornuéjols G, Natraj N (1996) Gomory cuts revisited. Oper Res Lett 19:1–9 ftp: cumparsita.gsb.columbia.edu
6. Bixby RE, Ceria S, McZeal CM, Savelsbergh MWP (1998) An updated mixed integer programming library: MIPLIB 3.0. Optima 58:12–15, Problems available at www.caam.rice.edu/bixby/miplib/miplib.html
7. Bixby RE, Cook W, Cox A, Lee EK (June 1995) Parallel mixed integer programming. Techn Report Dept Comput Appl Math Rice Univ
8. Bixby RE, Lee EK (1998) Solving a truck dispatching scheduling problem using branch-and-cut. Oper Res 46:355–367
9. Boyd EA (1994) Fenchel cutting planes for integer programs. Oper Res 42:53–64
10. Brunetta L, Conforti M, Rinaldi G (1997) A branch-and-cut algorithm for the equicut problem. Math Program 78:243–263
11. Ceria S, Cordier C, Marchand H, Wolsey LA (1998) Cutting plane algorithms for integer programs with general integer variables. Math Program 81:201–214, ftp: cumparsita.gsb.columbia.edu
12. Ceria S, Cornuéjols G, Dawande M (1995) Combining and strengthening Gomory cuts. In: Balas E, Clausen J (eds) Lecture Notes Computer Sci., vol 920. Springer, Berlin, ftp: cumparsita.gsb.columbia.edu
13. Christof T, Jünger M, Kececioglu J, Mutzel P, Reinelt G (1997) A branch-and-cut approach to physical mapping with end-probes. In: Proc. 1st Internat. Conf. Computational Molecular Biology (RECOMB-1), Santa Fe, New Mexico, ftp://ftp.zpr.uni-koeln.de/pub/paper/zpr96-250.ps.gz or www.informatik.uni-koeln.de/ls_juenger/publications/list.html or www.iwr.uni-heidelberg.de/iwr/comopt/publications/recomb.ps.gz
14. Christof T, Reinelt G (1995) Parallel cutting plane generation for the TSP. In: Fritzson P, Finmo L (eds) Parallel Programming and Applications. IOS, 163–169
15. Chvátal V (1973) Edmonds polytopes and a hierarchy of combinatorial problems. Discret Math 4:305–337
16. Cordier C, Marchand H, Laundy R, Wolsey LA (1999) bc-opt: A branch-and-cut code for mixed integer programs. Math Program 86(2):335–353
17. Crowder HP, Johnson EL, Padberg M (1983) Solving large-scale zero-one linear programming problems. Oper Res 31:803–834
18. Eckstein J (1997) How much communication does parallel branch and bound need? INFORMS J Comput 9:15–29
19. Gomory RE (1958) Outline of an algorithm for integer solutions to linear programs. Bull Amer Math Soc 64:275–278
20. Gomory RE (1963) An algorithm for integer solutions to linear programs. In: Graves RL, Wolfe P (eds) Recent Advances in Mathematical Programming. McGraw-Hill, New York, pp 269–302
21. Grötschel M, Holland O (1991) Solution of large-scale travelling salesman problems. Math Program 51(2):141–202
22. Hoffman KL, Padberg M (1991) Improving LP-representation of zero-one linear programs for branch-and-cut. ORSA J Comput 3(2):121–134
23. Hoffman KL, Padberg M (1993) Solving airline crew scheduling problems by branch-and-cut. Managem Sci 39(6):657–682
24. Joy S, Borchers B, Mitchell JE (1997) A branch-and-cut algorithm for MAX-SAT and weighted MAX-SAT. In: Du D-Z, Gu J, Pardalos PM (eds) Satisfiability Problem: Theory and Appl. DIMACS 35. Amer. Math. Soc. and DIMACS, Providence, RI, pp 519–536
25. Jünger M, Kaibel V (1996) A basic study of the QAP polytope. Techn Report Inst Informatik Univ Köln, 96.215
26. Jünger M, Mutzel P (1996) Maximum planar subgraphs and nice embeddings – practical layout tools. Algorithmica 16:33–59
27. Jünger M, Störmer P (1995) Solving large-scale traveling salesman problems with parallel branch-and-cut. Techn Report Inst Informatik Univ Köln 95.191ftp://ftp.zpr.uni-koeln.de/pub/paper/zpr95-191.ps.gz or www.informatik.uni-koeln.de/ls_juenger/publications/list.html
28. Jünger M, Thienel S (1998) Introduction to ABACUS - A branch-and-cut system. Oper Res Lett 22:83–95
29. Lovász L, Schrijver A (1991) Cones of matrices and set-functions and 0-1 optimization. SIAM J Optim 1:166–190
30. Nemhauser GL, Savelsbergh MWP, Sigismondi GC (1994) MINTO, a Mixed INTeger Optimizer. Oper Res Lett 15:47–58
31. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
32. Padberg M, Rinaldi G (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Rev 33(1):60–100
33. Sherali HD, Adams WP (1990) A hierarchy of relaxation between the continuous and convex hull representations for zero-one programming problems. SIAM J Discret Math 3:411–430
34. Wolsey LA (1990) Valid inequalities for mixed integer programs with generalized and variable upper bound constraints. Discrete Appl Math 25:251–261

Integer Programming: Cutting Plane Algorithms

JOHN E. MITCHELL
Math. Sci. Rensselaer Polytechnic Institute, Troy, USA

MSC2000: 90C10, 90C05, 90C08, 90C11, 90C06, 90C08

Article Outline

Keywords
Totally Unimodular Matrices
Chvátal–Gomory Cutting Planes
Strong Cutting Planes From Polyhedral Theory
Alternative General Cutting Planes
Fixing Variables
Solving Large Problems
Provably Good Solutions
Equivalence of Separation and Optimization
Conclusions
See also
References

Keywords

Cutting planes; Integer program; Exact algorithms

Cutting plane methods are exact algorithms for *integer programming problems*. They have proven to be very useful computationally in the last few years, especially when combined with a branch and bound algorithm (cf. ► Integer programming: Branch and bound methods) in a branch and cut framework (cf. ► Integer programming: Branch and cut algorithms). These methods work by solving a sequence of linear programming relaxations of the integer programming problem. The relaxations are gradually improved to give better approximations to the integer programming problem, at least in the neighborhood of the optimal solution. For hard instances that cannot be solved to optimality, cutting plane algorithms can produce approximations to the optimal solution in moderate computation times, with guarantees on the distance to optimality.

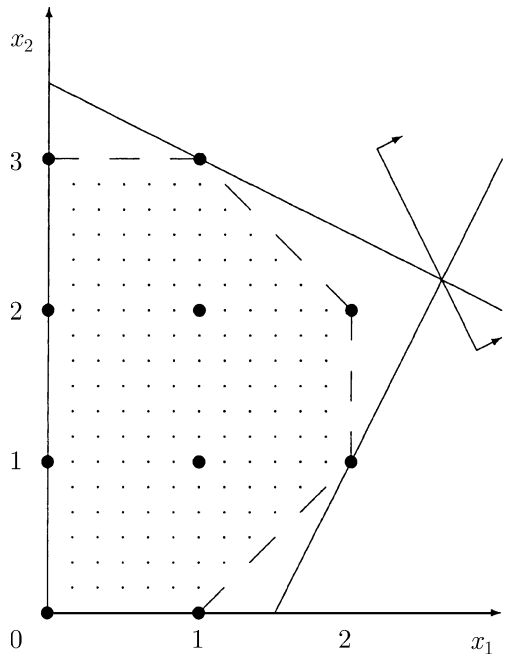
Cutting plane algorithms have been used to solve many different integer programming problems, including the *traveling salesman problem* [1,15,33], the *linear ordering* problem [16,29,30], *maximum cut* problems [4,28,36], and *packing* problems [18,31]. See [22] for a survey of applications of cutting plane methods, as well as a guide to the successful implementation of a cutting plane algorithm. G.L. Nemhauser and L. Wolsey [32] provide an excellent and detailed description of cutting plane algorithms and the other material in this entry, as well as other aspects of integer programming. The book [34] and also the more recent article [35] are excellent sources of additional material.

Cutting plane algorithms for general integer programming problems were first proposed by R.E. Gomory in [12,13]. Unfortunately, the cutting planes proposed by Gomory did not appear to be very strong, leading to slow convergence of these algorithms, so the algorithms were neglected for many years. The development of polyhedral theory and the consequent introduction of strong, problem specific cutting planes led to a resurgence of cutting plane methods in the 1980s, and cutting plane methods are now the method of choice for a variety of problems, including the traveling salesman problem. Recently, there has also been some research showing that the original cutting planes proposed by Gomory can actually be useful. There has also been research on other types of cutting planes for general integer programming problems. Current research is focused on developing cutting plane algorithms for a variety of hard combinatorial optimization problems, and on solving large instances of integer programming problems using these methods. All of these issues are discussed below.

Example 1 Consider, for example, the integer programming problem

$$\begin{cases} \min & -2x_1 - x_2 \\ \text{s.t.} & x_1 + 2x_2 \leq 7 \\ & 2x_1 - x_2 \leq 3 \\ & x_1, x_2 \geq 0 \quad \text{and integer.} \end{cases}$$

This problem is illustrated in Fig. 1. The feasible integer points are indicated. The *linear programming relaxation* (or *LP relaxation*) is obtained by ignoring the integrality restrictions; this is given by the polyhedron contained in the solid lines. The boundary of the *convex hull* of the feasible integer points is indicated by dashed lines.



Integer Programming: Cutting Plane Algorithms, Figure 1
A cutting plane example

If a cutting plane algorithm were used to solve this problem, the linear programming relaxation would first be solved, giving the point $x_1 = 2.6$, $x_2 = 2.2$, which has value -7.4. The inequalities $x_1 + x_2 \leq 4$ and $x_1 \leq 2$ are satisfied by all the feasible integer points but they are violated by the point (2.6, 2.2). Thus, these two inequalities are valid *cutting planes*. These two constraints can then be added to the relaxation, and when the relaxation is solved again, the point $x_1 = 2$, $x_2 = 2$ results, with value -6. Notice that this point is feasible in the original integer program, so it must actually be optimal for that problem, since it is optimal for a relaxation of the integer program.

If, instead of adding both inequalities, just the inequality $x_1 \leq 2$ had been added, the optimal solution to the new relaxation would have been $x_1 = 2$, $x_2 = 2.5$, with value -6.5. The relaxation could then have been modified by adding a cutting plane that *separates* this point from the convex hull, for example $x_1 + x_2 \leq 4$. Solving this new relaxation will again result in the optimal solution to the integer program. This illustrates the basic structure of a cutting plane algorithm:

- Solve the linear programming relaxation.
- If the solution to the relaxation is feasible in the integer programming problem, STOP with optimality.
- Otherwise, find one or more cutting planes that separate the optimal solution to the relaxation from the convex hull of feasible integral points, and add a subset of these constraints to the relaxation.
- Return to the first step.

Typically, the first relaxation is solved using the primal simplex algorithm. After the addition of cutting planes, the current primal iterate is no longer feasible. However, the dual problem is only modified by the addition of some variables. If these extra dual variables are given the value 0 then the current dual solution is still dual feasible. Therefore, subsequent relaxations are solved using the dual simplex method.

Notice that the values of the relaxations provide lower bounds on the optimal value of the integer program. These lower bounds can be used to measure progress towards optimality, and to give performance guarantees on integral solutions.

Totally Unimodular Matrices

Consider the integer program

$$\min \left\{ c^\top x \colon Ax = b, 0 \leq x \leq u, x \text{integer} \right\},$$

where A is an $m \times n$ matrix, c, x, and u are n-vectors, and b is an m-vector. A cutting plane method attempts to refine a linear programming relaxation until it gives a good approximation of the convex hull of feasible integer points, at least in the region of the optimal solution. In some settings, the solution to the initial linear programming relaxation $\min \{c^\top x \colon Ax = b, 0 \leq x \leq u\}$ may give the optimal solution to the integer program. This is guaranteed to happen if the constraint matrix A is *totally unimodular*, that is, the determinant of every square submatrix of A is either 0 or ± 1. Examples of totally unimodular matrices include the node-arc incidence matrix of a directed graph, the node-edge incidence matrix of a bipartite undirected graph, and interval matrices (where each row of A consists of a possibly empty set of zeroes followed by a set of ones followed by another possibly empty set of zeros). It therefore suffices to solve the linear programming relaxation of *maximum flow problems* and *shortest path* problems on directed graphs, the *assignment problem*, and some problems that involve assigning workers to shifts, among others.

Chvátal–Gomory Cutting Planes

One method of generating cutting planes involves combining together inequalities from the current description of the linear programming relaxation. This process is known as *integer rounding*, and the cutting planes generated are known as *Chv átal-Gomory cutting planes*. Integer rounding was implicitly described by Gomory in [12,13], and described explicitly by V. Chv átal in [7].

Consider again the example problem given earlier. The first step is to take a weighted combination of the inequalities. For example,

$$0.2(x_1 + 2x_2 \leq 7) + 0.4(2x_1 - x_2 \leq 3)$$

gives the valid inequality for the relaxation:

$$x_1 \leq 2.6.$$

In any feasible solution to the integer programming problem, the left hand side of this inequality must take an integer value. Therefore, the right hand side can be rounded down to give the following valid inequality for the integer programming problem:

$$x_1 \leq 2.$$

This process can be modified to generate additional inequalities. For example, taking the combination 0.5 ($x_1 + 2x_2 \leq 7$) + 0 ($2x_1 - x_2 \leq 3$) gives 0.5 $x_1 + x_2 \leq 3.5$, which is valid for the relaxation. Since all the variables are constrained to be nonnegative, rounding down the left hand side of this inequality will only weaken it, giving $x_2 \leq 3.5$, also valid for the LP relaxation. Now rounding down the right hand side gives $x_2 \leq 3$, which is valid for the integer programming problem, even though it is not valid for the LP relaxation.

Gomory originally derived constraints using the optimal simplex tableau. The LP relaxation of the simple example above can be expressed in equality form as:

$$\begin{cases} \min & -2x_1 - x_2 \\ \text{s.t.} & x_1 + 2x_2 + x_3 = 7 \\ & 2x_1 - x_2 + x_4 = 3 \\ & x_i \geq 0, \quad i = 1, \ldots, 4. \end{cases}$$

Notice that if x_1 and x_2 take integral values, then so must x_3 and x_4. Solving this LP using the simplex algorithm gives the optimal tableau

7.4	0	0	0.8	0.6
2.2	0	1	0.4	−0.2
2.6	1	0	0.2	0.6

The rows of the tableau are linear combinations of the original objective function and constraints, and cutting planes can be generated using them. The objective function row implies that $0.8x_3 + 0.6x_4 \geq 0.4$ in any integral feasible solution. It can be seen that this is equivalent to requiring that $2x_1 + x_2 \leq 7$, by substituting for x_3 and x_4 from the equality form given above. It is also possible to generate constraints from the other rows of the tableau. For example, the first constraint row of the tableau is equivalent to the equality $2.2 = x_2 + 0.4x_3 - 0.2x_4$. The fractional part of the right-handside of this equation is $0.4x_3 + 0.8x_4$. This must be at least as large as the fractional part of the left hand side in any feasible integral solution, giving the valid cutting plane $0.4x_3 + 0.8x_4 \geq 0.2$, which is equivalent to $x_1 \leq 2.5$. Similarly, the final row of the tableau can be used to generate the constraint $0.2x_3 + 0.6x_4 \geq 0.6$, or equivalently $7x_1 - x_2 \leq 13$. In practice, the cut added to the tableau should be expressed in the nonbasic variables, here x_3 and x_4, since the tableau will then be in standard form for the dual simplex algorithm.

Gomory 's cutting plane algorithm solves an integer program by solving the LP relaxation to optimality, generating a cutting plane from a row of the tableau if necessary, adding this additional constraint to the relaxation, solving the new relaxation, and repeating until the solution to the relaxation is integral. It was shown in [13] that if a cutting plane is always generated from the first possible row, then Gomory 's cutting plane algorithm will solve an integer program in a finite number of iterations.

Unfortunately, this finite convergence appears to be slow. However, it was shown in [3,6] that Gomory 's cutting plane algorithm can be made competitive with other methods if certain techniques are used, such as adding many Chv átal-Gomory cuts at once.

It follows from the finite convergence of Gomory 's cutting plane algorithm that every valid inequality for the convex hull of feasible integral points is either generated by repeated application of integer rounding or is dominated by an inequality generated in such a way. There are many different ways to generate a given in-

equality using integer rounding. The *Chv átal rank* of a valid inequality is the minimum number of successive applications of the integer rounding procedure that are needed in order to generate the inequality; it should be noted that a rank 2 inequality can be generated by applying the integer rounding procedure to a large number of rank 1 and rank 0 inequalities, for example.

It was shown in [26] that Gomory cutting planes can be generated even when an interior point method is used to solve the LP relaxations, because much of the information in the simplex tableau can still be obtained easily.

Strong Cutting Planes From Polyhedral Theory

The resurgence of interest in cutting plane algorithms in the 1980s was due to the development of *polyhedral combinatorics* and the consequent implementation of cutting plane algorithms that used *facets* of the convex hull of integral feasible points as cuts. A facet is a face of a polytope that has dimension one less than the dimension of the polytope. Equivalently, to have a complete linear inequality description of the polytope, it is necessary to have an inequality that represents each facet.

In the example above, the convex hull of the set of feasible integer points has dimension 2, and all of the dashed lines represent facets. The valid inequality $x_1 + 2x_2 \leq 7$ represents a face of the convex hull of dimension 0, namely the point (1, 3).

If a complete description of the convex hull of the set of integer feasible points is known, then the integer problem can be solved as a linear programming problem by minimizing the objective function over this convex hull. Unfortunately, it is not easy to get such a description. In fact, for an *NP*-complete problem [11] (cf. also ► *NP*-complete problems and proof methodology), such a description must contain an exponential number of facets, unless P = NP.

The paper [22] contains a survey of problems that have been solved using *strong cutting plane* algorithms. Typically in these algorithms, first a partial polyhedral description of the convex hull of the set of integer feasible points is determined. This description will usually contain families of facets of certain types. *Separation routines* for these families can often be developed; such a routine will take as input a point (for example, the optimal solution to the LP relaxation), and return as output violated constraints from the family, if any exist.

The prototypical combinatorial optimization problem that has been successfully attacked using cutting plane methods is the *traveling salesman problem*. In this problem, a set of cities is provided along with distances between the cities. A route that visits each city exactly once and returns to the original city is called a *tour*. It is desired to choose the shortest tour. This problem has many applications, including printed circuit board (PCB) production: a PCB needs holes drilled in certain places to hold electronic components such as resistors, diodes, and integrated circuits. These holes can be regarded as the cities, and the objective is to minimize the total distance traveled by the drill.

The traveling salesman problem can be represented on a graph, $G = (V, E)$, where V is the set of vertices (or cities) and E is the set of edges (or links between the cities). Each edge $e \in E$ has an associated cost (or length) c_e. If the incidence vector x is defined by

$$x_e = \begin{cases} 1 & \text{if edge } e \text{ is used,} \\ 0 & \text{otherwise,} \end{cases}$$

then the traveling salesman problem can be formulated as

$$\min \left\{ \sum c_e x_e : \ x \text{ the incidence vector of a tour} \right\}.$$

Notice that for a tour, at each vertex the sum of the edge variables must be two; this is called a degree constraint. This leads to the relaxation of the traveling salesman problem:

$$\begin{cases} \min & \sum c_e x_e \\ \text{s.t.} & \sum_{e \in \delta(v)} x_e = 2 \quad \text{for all vertices } v \\ & x_e = 0 \text{ or } 1 \quad \text{for all edges } e. \end{cases}$$

Here, $\delta(v)$ denotes the set of all edges incident to vertex v. All tours are feasible in this formulation, but it also allows infeasible solutions corresponding to *subtours*, consisting of several distinct unconnected loops. To force the solution to be a tour, it is necessary to include *subtour elimination constraints* of the form

$$\sum_{e \in \delta(U)} x_e \geq 2$$

for every subset $U \subseteq V$ with cardinality $2 \leq |U| \leq |V|/2$, where $\delta(U)$ denotes the set of edges with exactly one endpoint in U. Any feasible solution to the relaxation given above which also satisfies the subtour elimination constraints must be the incidence vector of a tour. Unfortunately, the number of subtour elimination constraints is exponential in the number of cities. This led G.B. Dantzig et al. to propose a cutting plane algorithm in [9], where the subtour elimination constraints are added as cutting planes as necessary.

The degree constraints and the subtour elimination constraints, together with the simple bounds $0 \leq x_e \leq 1$, are still not sufficient to describe the convex hull of the incidence vectors of tours. This approach of [9] has been extended in recent years by the incorporation of additional families of cutting planes – see, for example, [1,15,33].

Thus, cutting plane algorithms can be used even when the integer programming formulation of the problem has an exponential number of constraints. Similar ideas are used in papers on the matching problem [10,14], maximum cut problems [4,28,36], and the linear ordering problem [16,30], among others. The pioneering work of J. Edmonds on the matching problem gave a complete description of the matching polytope, and this work was used in subsequent algorithms; it was also an inspiration to future work on many other problems and even to the formulation of complexity theory and the concept of a 'good' algorithm.

Alternative General Cutting Planes

A knapsack problem is an integer programming problem with just one linear inequality constraint. A general integer programming problem can be regarded as the intersection of several knapsack problems, one for each constraint. This observation was used in [8,19,20] to solve general integer programming problems. The approach consists of finding facets and strong cutting planes for the knapsack problem and adding these constraints to the LP relaxation of the integer program as cutting planes.

There has been interest recently in other families of cutting planes for general integer programming problems. Two such families of cuts are *lift-and-project cuts* [2] and *Fenchel cuts* [5]. To find a cut of this type, it is generally necessary to solve a linear programming problem.

These alternative general cutting planes are not usually strong enough on their own to solve an integer programming problem, and they are most successfully employed in branch and cut algorithms for integer programming; they are discussed in more detail in ► Integer programming: Branch and cut algorithms.

Fixing Variables

If the *reduced cost* of a nonbasic variable is sufficiently large at the optimal solution to an LP relaxation, then that variable must take its current value in any optimal solution to the integer programming problem. To make this more precise, suppose the binary variable x_j takes value zero in the optimal solution to an LP relaxation and that the reduced cost of this variable is r_j. The optimal value of the relaxation gives a lower bound $\underline{z}$ on the optimal value of the integer programming problem. The value z_{UB} of the best known feasible integral solution provides an upper bound on the optimal value. Any feasible point in the relaxation with $x_j = 1$ must have value at least $\underline{z} + r_j$, so such a point cannot be optimal if $r_j > z_{UB} - \underline{z}$. Similar tests can be derived for nonbasic variables at their upper bounds. It is also possible to fix variables when an interior point method is used to solve the relaxations [26].

Once some variables have been fixed in this manner, it is often possible to fix further variables using logical implications. For example, in a traveling salesman problem, if x_e has been set equal to one for two edges incident to a particular vertex, then all other edges incident to that vertex can have their values fixed to zero.

Solving Large Problems

It is generally accepted that interior point methods are superior to the simplex algorithm for solving sufficiently large linear programming problems. The situation for cutting plane algorithms for large integer programming problems is not so clear, because the dual simplex method is very good at reoptimizing if only a handful of cutting planes are added. Nonetheless, it does appear that interior point cutting plane algorithms may well have a role to play, especially for problems with very large relaxations (thousands of variables and constraints) and where a large number of cutting planes are added simultaneously (hundreds or thousands). LP relaxations of integer programming problems can experience severe degeneracy, which can cause the simplex

method to stall. Interior point methods suffer far less from the effects of degeneracy.

In [27], an interior point cutting plane algorithm is used for a maximum cut problem on a sparse graph, and the use of the interior point solver enables the solution of far larger instances than with a simplex solver, because of both the size of the problems and their degeneracy.

A combined interior point and simplex cutting plane algorithm for the linear ordering problem is described in [30]. In the early stages, an interior point method is used, because the linear programs are large and many constraints are added at once. In the later stages, the dual simplex algorithm is used, because just a few constraints are added at a time and the dual simplex method can then reoptimize very quickly. The combined algorithm is up to ten times faster than either a pure interior point cutting plane algorithm or a pure simplex cutting plane algorithm on the larger instances considered.

The polyhedral combinatorics of the quadratic assignment problem are investigated in [21]. It was found necessary to use an interior point method to solve the relaxations, because of the size of the relaxations.

Provably Good Solutions

Even if a cutting plane algorithm is unable to solve a problem to optimality, it can still be used to generate good feasible solutions with a *guaranteed bound to optimality*. This approach for the traveling salesman problem is described in [23]. The value of the current LP relaxation provides a lower bound on the optimal value of the integer programming problem. The optimal solution to the current LP relaxation (or a good feasible solution) can often be used to generate a good integral feasible solution using a heuristic procedure. The value of an integral solution obtained in this manner provides an upper bound on the optimal value of the integer programming problem.

For example, for the traveling salesman problem, edges that have x_e close to one can be set equal to one, edges with x_e close to zero can be set to zero, and the remaining edges can be set so that the solution is the incidence vector of a tour. Further refinements are possible, such as using 2-change or 3-change procedures to improve the tour, as described in [25].

This has great practical importance. In many situations, it is not necessary to obtain an optimal solution, and a good solution will suffice. If it is only necessary to have a solution within 0.5% of optimality, say, then the cutting plane algorithm can be terminated when the gap between the lower bound and upper bound is smaller than this tolerance. If the objective function value must be integral, then the algorithm can be stopped with an optimal solution once this gap is less than one.

Equivalence of Separation and Optimization

The *separation problem* for an integer programming problem can be stated as follows:

> Given an instance of an integer programming problem and a point x, determine whether x is in the convex hull of feasible integral points. Further, if it is not in the convex hull, find a separating hyperplane that cuts off x from the convex hull.

An algorithm for solving a separation problem is called a separation routine, and it can be used to solve an integer programming problem.

The *ellipsoid algorithm* [17,24] is a method for solving linear programming problems in polynomial time. It can be used to solve an integer programming problem with a cutting plane method, and it will work in a polynomial number of stages, or calls to the separation routine. If the separation routine requires only polynomial time then the ellipsoid algorithm can be used to solve the problem in polynomial time. It can also be shown that if an optimization problem can be solved in polynomial time then the corresponding separation problem can also be solved in polynomial time.

There are instances of any *NP*-hard problem that cannot be solved in polynomial time unless P = NP.Therefore, a cutting plane algorithm cannot always generate good cutting planes quickly for *NP*-hard problems. In practice, fast heuristics are used, and these heuristics may occasionally be unable to find a cutting plane even when one exists.

Conclusions

Cutting plane methods have been known for almost as long as the simplex algorithm. They have come back into favor since the early 1980s because of the development of strong cutting planes from polyhedral theory.

In practice, cutting plane methods have proven very successful for a wide variety of problems, giving provably optimal solutions. Because they solve relaxations of the problem of interest, they make it possible to obtain bounds on the optimal value, even for large instances that cannot currently be solved to optimality.

See also

- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos-Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem

References

1. Applegate D, Bixby RE, Chvátal V, Cook W (1998) On the solution of travelling salesman problems. Documenta Math, no. Extra Vol. Proc. ICM III:645–656
2. Balas E, Ceria S, Cornuéjols G (1996) Mixed 0–1 programming by lift-and-project in a branch-and-cut framework. Managem Sci 42:1229–1246, ftp: cumparsita.gsb.columbia.edu
3. Balas E, Ceria S, Cornuéjols G, Natraj N (1996) Gomory cuts revisited. Oper Res Lett 19:1–9, ftp: cumparsita.gsb.columbia.edu
4. Barahona F, Grötschel M, Jünger M, Reinelt G (1988) An application of combinatorial optimization to statistical physics and circuit layout design. Oper Res 36(3):493–513
5. Boyd EA (1994) Fenchel cutting planes for integer programs. Oper Res 42:53–64
6. Ceria S, Cornuéjols G, Dawande M (1995) Combining and strengthening Gomory cuts. In: Balas E, Clausen J (eds) Lecture Notes Computer Sci., vol 920. Springer, Berlin, ftp: cumparsita.gsb.columbia.edu
7. Chvátal V (1973) Edmonds polytopes and a hierarchy of combinatorial problems. Discret Math 4:305–337
8. Crowder HP, Johnson EL, Padberg M (1983) Solving large-scale zero-one linear programming problems. Oper Res 31:803–834
9. Dantzig GB, Fulkerson DR, Johnson SM (1954) Solutions of a large-scale travelling salesman problem. Oper Res 2:393–410
10. Edmonds J (1965) Maximum matching and a polyhedron with 0, 1 vertices. J Res Nat Bureau Standards 69B:125–130
11. Garey MR, Johnson DS (1979) Computers and intractibility: A guide to the theory of NP-completeness. Freeman, New York
12. Gomory RE (1958) Outline of an algorithm for integer solutions to linear programs. Bull Amer Math Soc 64:275–278
13. Gomory RE (1963) An algorithm for integer solutions to linear programs. In: Graves RL, Wolfe P (eds) Recent Advances in Mathematical Programming. McGraw-Hill, New York, pp 269–302
14. Grötschel M, Holland O (1985) Solving matching problems with linear programming. Math Program 33:243–259
15. Grötschel M, Holland O (1991) Solution of large-scale travelling salesman problems. Math Program 51(2):141–202
16. Grötschel M, Jünger M, Reinelt G (1984) A cutting plane algorithm for the linear ordering problem. Oper Res 32:1195–1220
17. Grötschel M, Lovasz L, Schrijver A (1988) Geometric algorithms and combinatorial optimization. Springer, Berlin
18. Grötschel M, Martin A, Weismantel R (1996) Packing Steiner trees: A cutting plane algorithm and computational results. Math Program 72:125–145
19. Hoffman KL, Padberg M (1985) LP-based combinatorial problem solving. Ann Oper Res 4:145–194
20. Hoffman KL, Padberg M (1991) Improving LP-representation of zero-one linear programs for branch-and-cut. ORSA J Comput 3(2):121–134
21. Jünger M, Kaibel V (1996) A basic study of the QAP polytope. Techn Report Inst Informatik Univ Köln 96.215
22. Jünger M, Reinelt G, Thienel S (1995) Practical problem solving with cutting plane algorithms in combinatorial optimization. In: Combinatorial Optimization. In: DIMACS. Amer. Math. Soc., Providence, RI, pp 111–152
23. Jünger M, Thienel S, Reinelt G (1994) Provably good solutions for the traveling salesman problem. ZOR - Math Meth Oper Res 40:183–217
24. Khachiyan LG (1979) A polynomial algorithm in linear programming. Soviet Math Dokl 20:1093–1096 (Dokl Akad Nauk SSSR 224:1093–1096)

25. Lin S, Kernighan BW (1973) An effective heuristic for the traveling salesman problem. Oper Res 21:498–516
26. Mitchell JE (1997) Fixing variables and generating classical cutting planes when using an interior point branch and cut method to solve integer programming problems. Europ J Oper Res 97:139–148
27. Mitchell JE (1998) An interior point cutting plane algorithm for Ising spin glass problems. In: Kischka P, Lorenz H-W (eds) Oper. Res. Proc., SOR 1997, Jena, Germany. Springer, Berlin, pp 114–119. www.math.rpi.edu/mitchj/papers/isingint.ps
28. Mitchell JE (2000) Computational experience with an interior point cutting plane algorithm. SIAM J Optim 10(4):1212–1227
29. Mitchell JE, Borchers B (1996) Solving real-world linear ordering problems using a primal-dual interior point cutting plane method. Ann Oper Res 62:253–276
30. Mitchell JE, Borchers B (2000) Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm. In: Frenk H et al (eds) High Performance Optimization. Kluwer, Dordrecht, pp 349–366 (Chap. 14)
31. Nemhauser GL, Sigismondi G (1992) A strong cutting plane/branch-and-bound algorithm for node packing. J Oper Res Soc 43:443–457
32. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
33. Padberg M, Rinaldi G (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Rev 33(1):60–100
34. Schrijver A (1986) Theory of linear and integer programming. Wiley, New York
35. Schrijver A (1995) Polyhedral combinatorics. In: Graham RL, Grötschel M, Lovász L (eds) Handbook Combinatorics, vol 2. Elsevier, Amsterdam, pp 1649–1704
36. De Simone C, Diehl M, Jünger M, Mutzel P, Reinelt G, Rinaldi G (1995) Exact ground states of Ising spin glasses: New experimental results with a branch and cut algorithm. J Statist Phys 80:487–496

Integer Programming Duality

J. N. Hooker
Graduate School of Industrial Admin.,
Carnegie Mellon University, Pittsburgh, USA

MSC2000: 90C10, 90C46

Article Outline

Keywords
Linear Programming Duality
Integer Programming
Surrogate Duality
Lagrangian Duality
Superadditive Duality
Solving the Superadditive Dual
Another Functional Dual
Inference Duality
Conclusions
See also
References

Keywords

Integer programming; Duality

One of the more elegant and satisfying ideas in the theory of optimization is linear programming duality. The dual of a linear programming problem is not only interesting theoretically but has great practical value, because it provides *sensitivity analysis*, bounds on the optimal value, and marginal values for resources.

It is natural to want to extend duality to integer programming in order to obtain these same benefits. The matter is not so simple, however. Linear programming duality actually represents several concepts of duality that happen to coincide in the case of linear programming but diverge as one moves to other types of optimization problems. The benefits also decouple, because each duality concept provides some of them but not others.

Five types of integer programming duality are surveyed here. None is clearly superior to the others, and their strengths and weaknesses are summarized in at the end of the article.

Linear Programming Duality

A brief summary of *linear programming duality* will provide a foundation for the rest of the discussion. Consider the linear programming (primal) problem,

$$\begin{cases} \max & cx \\ \text{s.t.} & Ax \le b \\ & x \ge 0, \end{cases} \tag{1}$$

where A is an $m \times n$ matrix. The dual problem may be stated

$$\begin{cases} \min & ub \\ \text{s.t.} & uA \ge c \\ & u \ge 0, \end{cases} \tag{2}$$

where u is a vector of *dual variables*. This is a *strong dual* because its optimal value is the same as that of the primal problem, unless both primal and dual are infeasible. (An unbounded or infeasible maximization problem is regarded as having optimal value ∞ or $-\infty$, respectively, and analogously for minimization problems.)

The linear programming dual brings at least three important benefits.

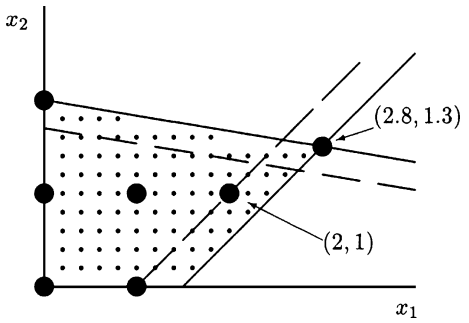
a) (Bounds) The value of any feasible dual solution provides an upper bound on the optimal value of the primal problem. For any x and y that are primal and dual feasible, respectively, $ub \geq uAx \geq cx$. The first inequality is due to the fact that $Ax \leq b$ and $u \geq 0$, and the second is due to the fact that $uA \geq c$ and $x \geq 0$. By finding a dual feasible solution, one can estimate how much a primal feasible solution falls short of optimality. Although this property is less important for linear programming, where robust solution algorithms are available, it is essential for integer programming.

b) (Sensitivity analysis) Due to a), the dual solution provides a partial sensitivity analysis. Let $\overline{u}$ be an optimal solution of the dual problem (2), so that $\overline{u}b$ is the optimal value of both primal and dual. If the right-hand side of the constraint in (1) is perturbed by Δb, so that it becomes $Ax \leq b + \Delta b$, then the dual (2) becomes

$$\begin{cases} \min & u(b + \Delta b) \\ \text{s.t.} & uA \geq c \\ & u \geq 0. \end{cases} \tag{3}$$

Because only the objective function changes, $\overline{u}$ is feasible in (3) as well as (2). So $\overline{u}(b+\Delta b)$ is an upper bound on the optimal value of the perturbed primal problem. The (possibly negative) change in the optimal value $\overline{u}b$ of the original problem is bounded above by $\overline{u}\Delta b$. The change is in fact equal to $\overline{u}\Delta b$ if the perturbation Δb lies within easily computable ranges.

c) (Complementary slackness) Due to b), the marginal values of resources are readily available. If the right-hand side b_i of a particular constraint of (1) represents a resource constraint, then a change Δb_i in the amount of resource available raises the optimal profit by at most $\overline{u}_i \Delta b_i$. In particular there is a *complementary slackness* property, which says that a surplus resource has no marginal value. If $\overline{x}$ is optimal in (1), then $\overline{u}(b - A\overline{x}) = 0$.



Integer Programming Duality, Figure 1
The shaded polyhedron is the feasible set of a linear programming problem with optimal solution $(x_1, x_2) = (2.8, 1.3)$ and optimal value 69. The dashed lines represent a perturbation of the right-hand sides. The black dots represent feasible solutions of the corresponding integer programming problem, which has optimal solution $(x_1, x_2) = (2, 1)$ and optimal value 50

Consider for example the linear programming problem

$$\begin{cases} \max & 20x_1 + 10x_2 \\ \text{s.t.} & x_1 + 4x_2 \leq 8 \\ & 2x_1 - 2x_2 \leq 3 \\ & x_1, x_2 \geq 0, \end{cases} \tag{4}$$

which is graphed in Fig. 1. The optimal dual solution is $(u_1, u_2) = (6, 7)$. If the two constraints represent two resource limitations, then the resources have marginal values of at most 6 and 7, respectively. If one less unit of each resource is available (represented by dashed lines in Fig. 1.), then the change in the objective function value is bounded above by $-6-7 = -13$. In fact, the profit decreases by exactly 13.

Integer Programming

Integer programming modifies the linear programming problem (1) by requiring the variables to take integral values:

$$\begin{cases} \max & cx \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \text{ and integer.} \end{cases} \tag{5}$$

In *mixed integer/linear programming* (MILP) some variables are continuous and some are integral. For ease of exposition, the discussion here is restricted to pure (unmixed) integer programming.

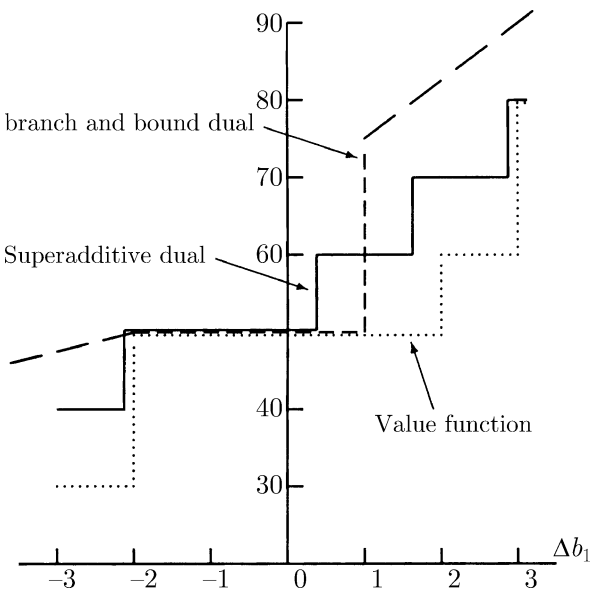
The example problem (4) may be modified to obtain the integer programming problem,

$$\begin{cases} \max & 20x_1 + 10x_2 \\ \text{s.t.} & x_1 + 4x_2 \leq 8 \\ & 2x_1 - 2x_2 \leq 3 \\ & x_1, x_2 \geq 0 \text{ and integers.} \end{cases} \tag{6}$$

Figure 1 illustrates this problem as well.

In linear programming, a constraint with slack at the optimal solution is redundant. It may be omitted without changing the optimal solution. The example shows that this is untrue for integer programming. Both constraints in (6) contain slack at the solution $(x_1, x_2) = (2, 1)$. Yet removing either would result in a different solution.

The concept of a marginal value is problematic in integer programming. Let the *value function* $v^*(b)$ indicate the optimal value of (1) or (5) for a given right-hand side b. In linear programming the marginal value of resource i is essentially the partial derivative of $v^*(b)$ with respect to b_i. Yet in integer programming $v^*(b)$ is a step function with respect to any b_i, as illustrated by the dotted line in Fig. 2. So it is unclear what would be meant by a marginal value. However, there may be a complementary slackness property of some kind, depending on the duality in question.



Integer Programming Duality, Figure 2
Upper bounds on the optimal value of an integer programming problem provided by the superadditive and branch and bound duals, as a function of the right-hand side perturbation Δb_1. The value function indicates the exact optimal value for each Δb_1

Surrogate Duality

One general scheme for formulating integer programming duals is to define a family of relaxations of the original problem that are parameterized by dual variables. The dual problem is then the problem of finding the tightest relaxation. It will be seen that the linear programming dual does exactly this.

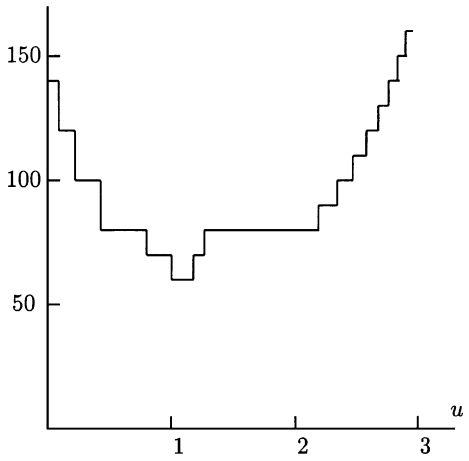
One instance of this scheme is *surrogate duality* [9,10,11]. The integer programming problem (5) can be relaxed by replacing the constraints $Ax \leq b$ with a *surrogate constraint*, i. e., a nonnegative linear combination of the inequalities in $Ax \leq b$. This yields a *surrogate relaxation* of (1):

$$\begin{cases} \max & cx \\ \text{s.t.} & uAx \leq ub \\ & x \geq 0 \text{ and integer.} \end{cases} \tag{7}$$

This is a relaxation in the sense that its feasible set contains that of (5). Its optimal value $\sigma(u)$ is therefore an upper bound on that of (5) for any $u \geq 0$. The surrogate relaxation may be much easier to solve than the original problem because it has only one constraint (other than nonnegativity). The surrogate dual problem is to find a u that gives the best bound:

$$\begin{cases} \min & \sigma(u) \\ \text{s.t.} & u \geq 0. \end{cases} \tag{8}$$

The surrogate relaxation of a linear programming problem (1) is (7) without the integrality constraint. From strong linear programming duality, its optimal value is $\sigma(u) = \min_\alpha \{\alpha ub\colon \alpha uA \geq \alpha c,\ \alpha \geq 0\}$. So the surrogate dual (8) becomes precisely the linear programming dual (2).



Integer Programming Duality, Figure 3
Plot of $\sigma(1, u)$ for a surrogate dual problem

Surrogate duality can be illustrated with the integer programming problem (6). The surrogate relaxation is

$$\begin{cases} \min & 20x_1 + 10x_2 \\ \text{s.t.} & (1+2u)x_1 + (4-2u)x_2 \le 8 + 3u \\ & x_1, x_2 \ge 0 \text{ and integers.} \end{cases}$$

Because there are only two constraints and only the ratio u_2/u_1 matters, u_1 is set to 1 and u_2 replaced with u. A plot of $\sigma(1, u)$ appears in Fig. 3.

The primary utility of the surrogate dual is to provide an upper bound on the optimal value of the original problem. In the example, the dual attains its optimal value of 60 when $1 < u \le 20/17$. This is better than the bound of 69 provided by the linear programming relaxation. But there is a duality gap of $60 - 50 = 10$.

One might speculate that the surrogate multipliers indicate the relative importance of the two constraints, but it is unclear what this means. One can say, however, that omitting a constraint with a vanishing multiplier does not raise the optimal value above that of the surrogate dual. Vanishing multipliers therefore identify redundant constraints when there is no duality gap.

The surrogate dual (8) must be solved by a search method that does not require gradient or subgradient information. Possible algorithms are discussed in [16]. The dual problem need not be solved to optimality, because only an upper bound is sought in any case.

Lagrangian Duality

Another form of relaxation duality, *Lagrangian duality* [5,6,7], removes some of the more troublesome constraints from (5) but inserts into the objective function a penalty for violating them. Thus the constraints are partitioned into 'hard' constraints $A^1x \le b^1$ and 'easy' constraints $A^2x \le b^2$:

$$\begin{cases} \max & cx \\ \text{s.t.} & A^1x \le b^1 \\ & A^2x \le b^2 \\ & x \ge 0 \text{ and integer.} \end{cases} \tag{9}$$

The hard constraints are *dualized* to obtained the *Lagrangian relaxation*:

$$\begin{cases} \max & cx + u(b^1 - A^1x) \\ \text{s.t.} & A^2x \le b^2 \\ & x \ge 0 \text{ and integer.} \end{cases} \tag{10}$$

This is a relaxation in the sense that its optimal value $\theta(u)$ is an upper bound on the optimal value of (5). For any x that is feasible in (9), $cx \le cx + u(b^1 - A^1x)$ because $u \ge 0$ and $b^1 - A^1x \ge 0$. The *Lagrangian dual problem* is

$$\begin{cases} \min & \theta(u) \\ \text{s.t.} & u \ge 0. \end{cases} \tag{11}$$

If all the constraints of (9) are dualized, then the Lagrangian dual is no improvement over the linear programming dual. In this case $\theta(u) = \max\{(c - uA)\, x + ub: x \ge 0, \text{integer}\}$. So $\theta(u)$ is ub when $c - uA \le 0$ and is ∞ otherwise. The Lagrangian dual problem (2) is now the problem of minimizing ub subject to $c - uA \le 0$ and $u \ge 0$, which is precisely the linear programming dual (2). (It follows that linear programming duality is a special case of Lagrangian duality.)

The Lagrangian dual is therefore useful only when some constraints are not dualized. These constraints must be carefully chosen so that the integer programming problem (5) is easy to solve. It may, for example, decouple into smaller problems or have other special structure.

As an example of Lagrangian duality, suppose that the first constraint of (6) is dualized. The Lagrangian

relaxation is

$$\begin{cases} \max & 20x_1 + 10x_2 + u(8 - x_1 - 4x_2) \\ \text{s.t.} & 2x_1 - 2x_2 \leq 3 \\ & x_1, x_2 \geq 0 \text{ and integers.} \end{cases}$$

The optimal solution of the Lagrangian dual is $u = 6$, with value $\theta(u) = 62$, slightly worse than the surrogate bound of 60 but still better than the linear programming bound of 69.

The Lagrangian and surrogate duals can be compared in general if the surrogate relaxation dualizes the same constraints as the Lagrangian relaxation. In this case it can be shown that the surrogate duality gap is never larger than the Lagrangian duality gap [7], and it tends to be smaller. The Lagrangian relaxation has the advantage, however, that it is often easier to solve than the surrogate relaxation that dualizes the same constraints. Moreover, $\theta(u)$ is convex and piecewise linear. A subgradient optimization method can be used to find a global minimum of $\theta(u)$ by finding a local minimum. In fact, if $\theta(u) = cx_u + u(b - Ax_u)$, then $b - Ax_u$ is a subgradient of $\theta(u)$ at u.

When there is no duality gap, the Lagrangian multipliers u_i can be viewed as sensitivities to right-hand side perturbations, with respect to at least one optimal solution. It can be shown that there is no duality gap if and only there exists a feasible solution $\overline{x}$ of (5) and $\overline{u} \geq 0$ that satisfy $\theta(\overline{u}) = c\overline{x} + \overline{u}(b^1 - A^1\overline{x})$ and complementary slackness: $\overline{u}(b^1 - A^1\overline{x}) = 0$. However, solution of the Lagrangian dual does not necessarily yield a solution $\overline{x}$ with these properties. Further search may be required.

Superadditive Duality

So far the linear programming dual has been viewed as a relaxation dual, of either the surrogate or Lagrangian type. It can also be viewed as representing the classical duality of vectors and linear functionals. For this purpose the dual of (1) is written:

$$\begin{cases} \min & f(b) \\ \text{s.t.} & f(A) \geq c \\ & f \in F. \end{cases} \tag{12}$$

Here f is a linear functional defined by a nonnegative row vector u, so that $f(b) = ub$ and $f(A) = uA$. The minimization is over the set F of all such functionals.

A similar dual of the integer programming problem (5) can be written as (12), but with minimization over a broader class F of functions. It can be shown that if F is the class of superadditive nondecreasing functions f with $f(0) = 0$, then (12) is a strong integer programming dual. This *superadditive dual* [2,15,20,24] provides sensitivities to right-hand side perturbations. It is also possible, at least in principle, to construct a function f that solves the dual, by means of a cutting plane algorithm.

A *superadditive function* f is one that satisfies $f(a + b) \geq f(a) + f(b)$ for all vectors a, b. The superadditive dual satisfies weak duality because if x is feasible in (5) and f is feasible in (12), then

$$cx \leq \sum_j f(a^j)x_j \leq \sum_j f(a^j x_j) \leq f(Ax) \leq f(b),$$

where a^j is row j of A. The first inequality follows from $f(A) \geq c$. The second is due to superadditivity of f and the fact that multiplication by a nonnegative integer x_j creates a sum of zero or more terms (also $f(0) = 0$). The third is due to superadditivity. The fourth follows from the fact that f is nondecreasing and $Ax \leq b$.

Strong duality can be established by exhibiting a dual feasible solution f for which there is no duality gap. Let a *rounding function* be a function of the form

$$R(d) = \lfloor M_k \lfloor M_{k-1} \cdots \lfloor M_1 d \rfloor \cdots \rfloor \rfloor, \tag{13}$$

where each M_i is a nonnegative matrix and $\lfloor\ \alpha\ \rfloor$ is α rounded down. A *Chvátal function* has the form uR, where $u \geq 0$ and R is a rounding function. Because Chvátal functions clearly belong to F, it suffices to exhibit a Chvátal function uR for which $uR(b)$ is the optimal value of (5).

This is done by generating Chvátal-Gomory cuts. A rank 1 *Chvátal-Gomory cut* for $Ax \leq b$, $x \geq 0$ is an inequality of the form $\lfloor\ mA\ \rfloor\, x \leq \lfloor\ mb\ \rfloor$, where $m \geq 0$ defines a linear combination of the rows of $Ax \leq b$. Rank 2 cuts are obtained by applying the same operation to rank 1 cuts, and so forth. Chvátal showed that the integer hull of any polyhedron (i. e., the convex hull of its integral points) is described by finitely many cuts of finite rank.

This implies that for some rounding function $\overline{R}$, $\overline{R}(A)x \leq \overline{R}(b)$ and $x \geq 0$ describe the integer hull of $P = \{x \geq 0 \colon Ax \leq b\}$. So the optimal value of the integer

programming problem (5) is the optimal value of

$$\begin{cases} \max & cx \\ \text{s.t.} & \overline{R}(A)x \le \overline{R}(b) \\ & x \ge 0. \end{cases} \tag{14}$$

The linear programming dual of (14) is

$$\begin{cases} \min & u\overline{R}b \\ \text{s.t.} & u\overline{R}(A) \ge c \\ & u \ge 0. \end{cases} \tag{15}$$

If $\overline{u}$ solves problem (15), then its optimal value $\overline{u}\overline{R}(b)$ is the optimal value of (14) and therefore of the original integer programming problem (5). Thus $f = \overline{u}\overline{R}$ is a Chvátal function that solves the dual problem (12).

The dual solution provides sensitivity analysis with respect to right-hand sides. Due to weak duality, $\overline{u}\overline{R}(b + \Delta b)$ is an upper bound on the optimal value when the right-hand side in (5) is perturbed to $b + \Delta b$. There is a form of complementary slackness, because for any optimal solution $\overline{x}$ of (5), $(\overline{u}\overline{R}(A) - c)\overline{x} = 0$.

Consider again the example problem (6). It will be seen below that $\overline{R}$ is

$$\overline{R}(d) = \left\lfloor \begin{pmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 1 \end{pmatrix} \left\lfloor \begin{pmatrix} \frac{4}{5} & \frac{1}{10} \\ \frac{1}{5} & \frac{2}{5} \\ 0 & \frac{1}{2} \end{pmatrix} d \right\rfloor \right\rfloor. \tag{16}$$

Problem (14) is

$$\begin{cases} \max & 20x_1 + 10x_2 \\ \text{s.t.} & x_1 + 2x_2 \le 4 \\ & x_1 - x_2 \le 1 \\ & x_1, x_2 \ge 0. \end{cases}$$

The solution of (15) is $\overline{u} = (10 \quad 10)$. So if the right-hand side of (6) is perturbed by $\Delta b = (\Delta b_1, \Delta b_2)$, the new optimal value of (6) is bounded above by

$$(10\ 10) \left\lfloor \begin{pmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 1 \end{pmatrix} \left\lfloor \begin{pmatrix} \frac{4}{5} & \frac{1}{10} \\ \frac{1}{5} & \frac{2}{5} \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 8 + \Delta b_1 \\ 3 + \Delta b_2 \end{pmatrix} \right\rfloor \right\rfloor.$$

For instance, if each resource is reduced by one ($\Delta b = (-1, -1)$), then the new optimal value is at most 50. In fact it is exactly 50. Figure 2 plots $\overline{u}\overline{R}(b + \Delta b)$ against Δb_1 for comparison with the value function $v^*(b + \Delta b)$. Note that there is complementary slackness, because $(\overline{u}\overline{R}(A) - c)\overline{x} = [(20 \quad 10) - (20 \quad 10)](2, 1) = 0$.

Solving the Superadditive Dual

A solution of the dual problem (12) can be constructed in stages that correspond to Chvátal ranks [3,14,23]. It is assumed without practical loss of generality that the components of A, b and c are rational numbers.

The first stage proceeds as follows. Let $x^1, \ldots, x^p$ be the vertices of $P_0 = P$. For each x^k consider the cone C_k of directions d for which x^k maximizes dx subject to $x \in P_0$. To describe C_k, let $\overline{A}x \le \overline{b}$ be the constraints of $Ax \le b$ that are *active* at x^k (i. e., the constraints $a^i x \le b_i$ for which $a^i x^k = b_i$), and let $-\overline{I}x \le 0$ be the active constraints of $-x \le 0$ (the constraints $-x_j \le 0$ for which $x_j^k = 0$). Then C_k is the cone spanned by the rows of $\overline{A}$ and $-\overline{I}$.

It suffices to identify a *Hilbert basis* [8] for C_k; i. e., a set of directions $d^1, \ldots, d^q$ such that every integer vector in C_k is a nonnegative integer combination of $d^1, \ldots, d^q$. Assume without loss of generality that the components of $\overline{A}$ are integers (the inequalities $\overline{A}x \le \overline{b}$ can be multiplied by appropriate integers to achieve this). Then the integer vectors $d^1, \ldots, d^q$ in the set

$$\{\lambda\overline{A} - \mu\overline{I}:\ 0 \le \lambda \le e,\ 0 \le \mu \le e\}$$

form a Hilbert basis for C_k, where e is a row vector of ones.

The next step is to generate rank 1 Chvátal-Gomory cuts associated with x^k. First note that each inequality of the form $d^j x \le d^j x^k$ supports P_0 at x^k and is therefore a nonnegative linear combination of the rows of $\overline{A}x \le \overline{b}$, $-\overline{I}x \le 0$. Thus one can write

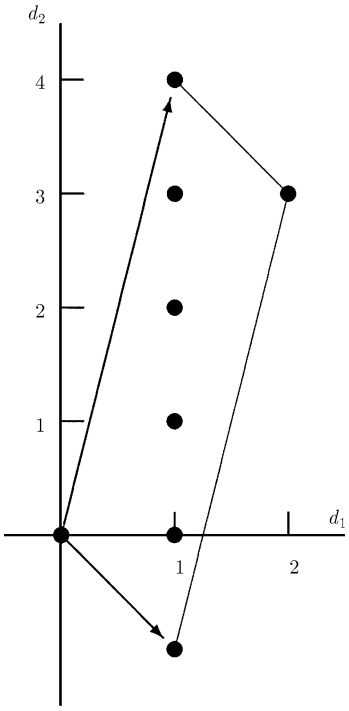
$$d^j = m^j\overline{A} - p^j\overline{I}. \tag{17}$$

The multipliers m^j and p^j can be obtained by solving (17). The valid inequalities $d^j x \le \lfloor d^j x^k \rfloor$ are clearly rank 1 cuts for $Ax \le b$, $-x \le 0$. Rank 1 cuts are generated in this fashion for all the vertices x^k.

Now let P_1 be defined by all of the rank 1 cuts generated, plus $x \ge 0$. Let the rows of M_1 be the vectors m^j corresponding to rank 1 cuts that define facets of P_1. Then $P_1 = \{x \ge 0: \lfloor M_1 A \rfloor x \le \lfloor M_1 b \rfloor\}$.

This same procedure is now applied to the vertices of P_1 to obtain M_2 and P_2, and so forth until all the vertices of P_k are integral. At this point (13) is the desired rounding function $\overline{R}$, and $f = \overline{u}\overline{R}$ solves the dual (12).

The Hilbert bases and inequalities $d^j x \le \lfloor d^j x^k \rfloor$ for problem (6) appear in Table 1. (The origin need not be



Integer Programming Duality, Figure 4
The black dots indicate a Hilbert basis for the cone spanned by (1, 4) and (1, — 1)

considered as a vertex.) At vertex $x^1 = (2.8, 1.3)$, for example, the two constraints $x_1 + 4x_2 \leq 8$ and $2x_1 - 2x_2 \leq 3$ are active, and so the cone C_1 is spanned by (1, 4) and (1, − 1). The Hilbert basis consists of the integer vectors of the region depicted in Fig. 4.

The polyhedra P_1 and P_2 are shown by dashed and dotted lines, respectively, in Fig. 5. Their facets (other than $x \geq 0$) and the corresponding vectors m^j appear in Table 2. The vectors M_1 and M_2 that appear in the rounding function (16) can be read from Table 2.

Another Functional Dual

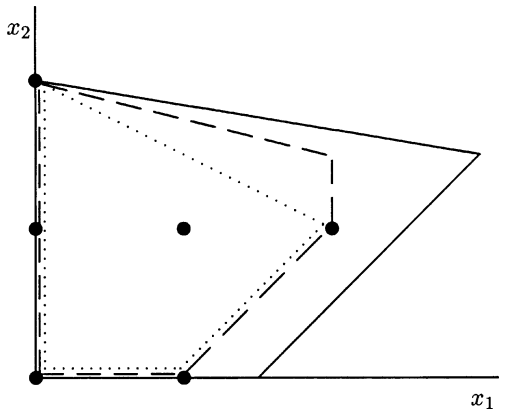
It is practical to solve the superadditive dual only when the problem is small or has special structure. An alternative is to derive a dual solution for (12) from the branch and bound tree that solves the primal problem, as proposed in [18] on the basis of work in [24]. This maneuver sacrifices an independently computed upper bound on the optimal value, but it provides useful sensitivity analysis in a more practical fashion than the superadditive dual. It might be called a 'branch and bound dual'.

Integer Programming Duality, Table 1
Hilbert basis vectors d^j and rank 1 cuts $d^j x \leq \lfloor d^j x^k \rfloor$ corresponding to vertices x^k of an integer programming problem

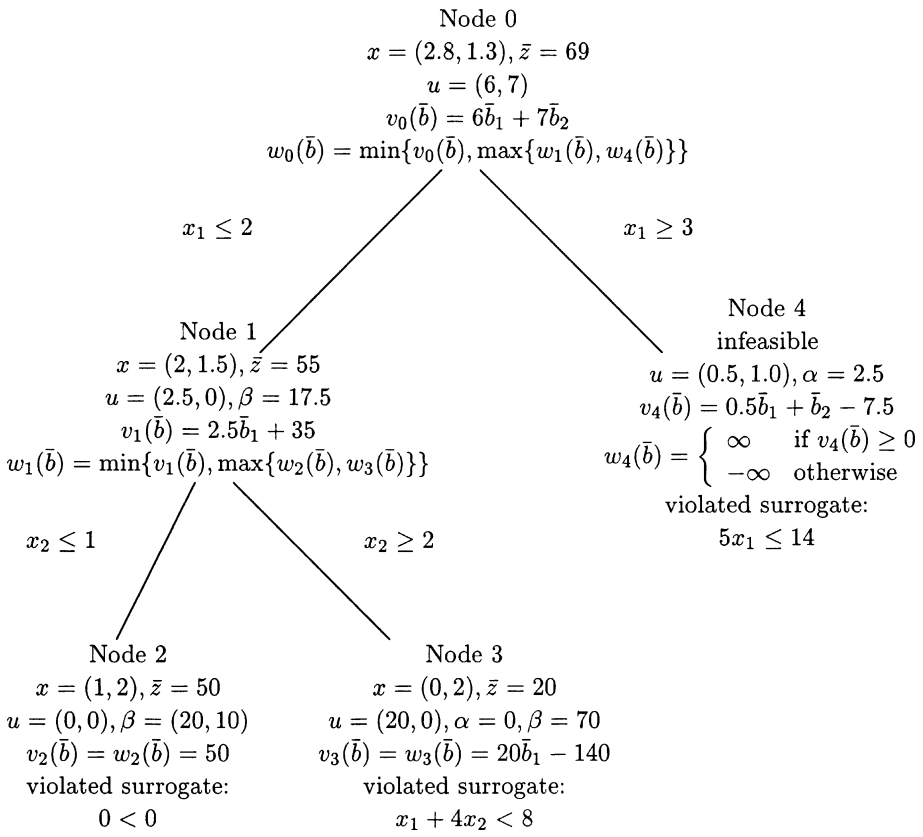
x^k	d^j	$d^j x \leq [d^j x^k]$
(2.8, 1.3)	(1, −1)	$x_1 - x_2 \leq 1$
	(1, 0)	$x_1 \leq 2$
	(1, 1)	$x_1 + x_2 \leq 4$
	(1, 2)	$x_1 + 2x_2 \leq 5$
	(1, 3)	$x_1 + 3x_2 \leq 6$
	(1, 4)	$x_1 + 4x_2 \leq 8$
	(2, 3)	$2x_1 + 3x_2 \leq 9$
(1.5, 0)	(0, −1)	$-x_2 \leq 0$
	(1, −1)	$x_1 - x_2 \leq 1$
	(1, −2)	$x_1 - 2x_2 \leq 1$
(0, 2)	(−1, 0)	$-x_1 \leq 0$
	(0, 1)	$x_2 \leq 2$
	(0, 2)	$2x_2 \leq 4$
	(0, 3)	$3x_2 \leq 6$
	(1, 3)	$x_1 + 3x_2 \leq 6$

Integer Programming Duality, Table 2
Polyhedra P_1, P_2 and vectors m^j corresponding to their facets

P_i	Facet	m^j
P_1	$x_1 + 3x_2 \leq 6$	$(\frac{4}{5}\ \ \frac{1}{10})$
	$x_1 \leq 2$	$(\frac{1}{5}\ \ \frac{2}{5})$
	$x_1 - x_2 \leq 1$	$(0\ \ \frac{1}{2})$
P_2	$x_1 + 2x_2 \leq 4$	$(\frac{2}{3}\ \ \frac{1}{3}\ \ 0)$
	$x_1 - x_2 \leq 1$	$(0\ \ 0\ \ 1)$



Integer Programming Duality, Figure 5
The polyhedra P_0 (solid line), P_1 (dashed line), and P_2 (dotted line) for an integer programming problem



Integer Programming Duality, Figure 6
A branch and bound tree with information relevant to the branch and bound dual and the inference dual

Rather than superadditive functions, the feasible set F in (12) will contain functions of the form

$$f(d) = \min\{yd + y_0,\ \max\{f_1(d), f_2(d)\}\}, \tag{18}$$

where $y \ge 0$ and f_1 and f_2 are either identically zero or of the form (18). Weak duality is easily shown. Strong duality is shown by constructing a solution as follows.

At each node t of the branch and bound tree for (5), one solves the linear relaxation

$$\begin{cases} \max & cx & \\ \text{s.t.} & Ax \le b & (u) \\ & -x \le -L^t & (\alpha) \\ & x \le U^t & (\beta), \end{cases} \tag{19}$$

where the lower and upper bounds L^t, U^t are defined by branching, and associated dual variables are shown on the right. By weak linear programming duality, $v_t(\bar{b}) = u\bar{b} - \alpha L^t + \beta U^t$ is an upper bound on the optimal value of (19) with perturbed right-hand side $d = b + \Delta b$. If (19) is infeasible, let u, α, β be the dual solution of the phase I problem in which the objective function is the sum of negative constraint violations. In this case $v_t(\bar{b})$ is $-\infty$ if $u\bar{b} - \alpha + \beta < 0$ and is ∞ otherwise.

Now if t_1, t_2 are the successor nodes of node t in the search tree,

$$w_t(\bar{b}) = \min\{v_t(\bar{b}), \max\{w_{t_1}(\bar{b}), w_{t_2}(\bar{b})\}\}$$

is an upper bound on the optimal value of (19) with right-hand side $\bar{b} = b + \Delta b$ and integral x. (At leaf nodes, the max expression is omitted.) The recursively computed function w_0 associated with the root node solves the dual problem (12) because $w_0(b)$ is the optimal value of (5).

The dual solution w_0 for the example problem (6) is indicated in the branch and bound tree for this problem depicted in Fig. 6. A plot of $w_0(b + \Delta b)$ as a function of Δb_1 appears in Fig. 2.

Inference Duality

Still another interpretation of the linear programming dual views it as an inference problem. It wishes to find the smallest upper bound z^* on the objective function that can be inferred from the constraints. This dual of (1) can be written

$$\begin{cases} \min & z \\ \text{s.t.} & \binom{Ax \le b}{x \ge 0} \text{ imply } cx \le z. \end{cases} \tag{20}$$

A corollary of the classical Farkas lemma states that the constraints $Ax \le b$, $x \ge 0$ imply $cx \le z$ if and only if they are infeasible or some surrogate $uAx \le ub$ dominates $cx \le z$; i. e., $uA \ge c$ and $ub \le z$. So the inference dual (20) seeks the smallest ub for which $uA \ge c$ and $u \ge 0$ (assuming the constraints are feasible), which is precisely the linear programming dual.

The inference dual can be generalized to integer programming if the implication in (20) is interpreted differently [4,13]. Constraints $Ax \le b$, $x \ge 0$ imply $cx \le z$ if and only if all *integer* (rather than all real) vectors x that satisfy the former also satisfy the latter. There is obviously no duality gap, because the maximum value z^* of cx is the smallest upper bound on cx implied by the constraints. As will be seen, inference duality allows calculation of sensitivity ranges for all problem data (not just right-hand sides) by solving linear programming problems.

To solve the dual (20) is in effect to exhibit a proof that the value of cx is at most z^*. In linear programming, a proof is a nonnegative linear combination of constraints, and the optimal dual multipliers u encode the desired proof. A method of proof suitable for integer programming is developed in [12], but for present purposes it suffices to reconstruct a proof from the branch and bound tree that solves the primal problem. Actually it will be proved that cx is at most $z^* + \Delta z$ (for any $\Delta z \ge 0$), to provide a more flexible analysis.

The proof is by contradiction. Assume, contrary to the claim, that the optimal value of (5) is strictly more than $z^* + \Delta z$. Then each branch of the tree can be seen as leading to a contradiction. At any given leaf node t let z_{LB} be the value of the best integral solution found so far ($z_{LB} = -\infty$ if none has been found). One of the following cases obtains.

a) The linear relaxation (19) is infeasible. Then the dual solution (u, α, β) proves infeasibility; i. e., $u^t A - \alpha + \beta \ge 0$ and $u^t b - \alpha^t L^t + \beta^t U^t < 0$. So the constraints $u^t Ax \le u^t b$, $L^t \le x \le U^t$, $x \ge 0$ are also infeasible. In other words, the bounds $L^t \le x \le U^t$ are inconsistent with the surrogate $u^t Ax \le u^t b$.

b) The solution of (19) is integral with value $\bar{z}_t$, where $\bar{z}_t > z^{LB}$. So the constraints

$$\begin{cases} -cx < -\bar{z}_t - \Delta z \\ Ax \le b \\ -x \le -L^t \\ x \le U^t \end{cases} \tag{21}$$

are infeasible. If (u^t, α^t, β^t) is the dual solution of (19), the multipliers $(1, u^t, \alpha^t, \beta^t)$ prove infeasibility of (21). This means that the bounds $L^t \le x \le U^t$ are inconsistent with the surrogate $(u^t A - c)x < u^t b - \bar{z}_t - \Delta z$.

c) The optimal value $\bar{z}$ of (19) satisfies $\bar{z} \le z_t^{\mathrm{LB}}$, where z_t^{LB} is the current lower bound (the tree is pruned at this node). Here the bounds $L^t \le x \le U^t$ are inconsistent with the surrogate $(u^t A - c)\, x < u^t b - z_t^{LB} - \Delta z$.

Thus there is a contradiction at every leaf node, because the bounds $L^t \le x \le U^t$ are inconsistent with some surrogate at every leaf node.

The key to sensitivity analysis is that a contradiction remains at every leaf node, and the proof remains valid, so long as the bounds remain inconsistent with the surrogates after perturbation of the data. To analyze how much perturbation is possible, the following observation is helpful. The bounds $L \le x \le U$ are inconsistent with inequality $dx \le \delta$ if and only if there exists a vector $\bar{d} \ge 0$ such that

$$\begin{cases} dL - \bar{d}(U - L) > \delta \\ \bar{d} \ge d, \quad \bar{d} \ge 0. \end{cases} \tag{22}$$

Now let (5) be perturbed as follows:

$$\begin{cases} \max & (c + \Delta c)x \\ \text{s.t.} & (A + \Delta A)x \le b + \Delta b \\ & x \ge 0 \text{ and integer.} \end{cases} \tag{23}$$

Thus the violated surrogate in case a) becomes $u^t(A + \Delta A)\, x \le u^t(b + \Delta b)$, and similarly in cases b) and c). Using (22), the optimal value of (23) rises no more than Δz ($\Delta z \ge 0$) if the perturbation satisfies the following

Integer Programming Duality, Table 3
Properties of five integer programming duals

Type of dual	Strong duality?	Computational bounds?	Sensitivity analysis?	Complementary slackness?
Surrogate	No	Yes	Very limited, even if no duality gap	No
Lagrangian	No	Yes	For RHS, if no duality gap	Yes, if no duality gap
Superadditive	Yes	Not practical	For RHS only	Yes
Branch and bound	Yes	No	For RHS only	No
Interface	Yes	No	Bounds for all problem date	No

for some $\overline{q}^t \geq 0$ at every leaf node t,

$$\begin{cases} (q^t + \Delta q^t)L^t - \overline{q}^t(U^t - L^t) \\ \quad \geq u^t(A + \Delta A) + z_t \\ \overline{q}^t \geq q^t + \Delta q^t, \quad \overline{q}^t \geq 0. \end{cases} \tag{24}$$

Here

$$\begin{aligned} q^t &= u^t A - u_0^t c, \\ \Delta q^t &= u^t \Delta A - u_0^t \Delta c, \\ (u_0^t, z_t) &= \begin{cases} (0, \epsilon) & \text{in case a)}, \\ (1, \overline{z}_t + \Delta z) & \text{in case b)}, \\ (1, z_t^{LB} + \Delta z) & \text{in case c)}. \end{cases} \end{aligned}$$

This can be checked by linear programming. Note that the perturbations ΔA, Δb, Δc are not restricted to be nonnegative. Ranges for any perturbation can be computed by minimizing and maximizing it subject to (24) with all other perturbations set to zero.

The dual solutions in Fig. 6 suffice to generate the inequalities (24) for the example problem (6). Leaf nodes 2, 3 and 4 respectively illustrate cases b), c) and a). For instance, the inequalities for leaf node $t = 2$ are

$$\begin{aligned} &-2\Delta c_1 - \Delta c_2 - 2\overline{q}_1^2 - \overline{q}_2^2 \geq 0, \\ &\overline{q}_1^2 \geq -20 - \Delta c_1, \quad \overline{q}_1^2 \geq 0, \\ &\overline{q}_2^2 \geq -10 - \Delta c_2, \quad \overline{q}_2^2 \geq 0. \end{aligned}$$

At leaf nodes 3 and 4 one must assume some large but finite upper bound on variables x_j for which U_j is otherwise infinite. The resulting sensitivity range for b_1 is given below, along with the ranges yielded by the superadditive dual, the branch and bound dual, and the true value function (the last three from Fig. 2).

- inference dual: $-\infty < \Delta b_1 < 1$;
- superadditive dual: $-\infty < \Delta b_1 < 0.375$;
- branch and bound dual: $-\infty < \Delta b_1 < 1$;
- maximum range: $-\infty < \Delta b_1 < 2$.

No perturbation within the maximum range causes the optimal value to rise above 50. The various forms of sensitivity analysis generally provide more conservative ranges (the same is true of classical linear programming). This example shows that the superadditive dual, although the hardest to compute, does not necessarily provide the sharpest analysis. The inference dual, unlike the others, provides ranges for all problem data:

$$\begin{aligned} \begin{pmatrix} -\infty \\ -\infty \end{pmatrix} < \Delta b < \begin{pmatrix} 1 \\ \frac{1}{2} \end{pmatrix}, \\ \begin{pmatrix} -\frac{1}{3} + \epsilon & -\frac{1}{3} + \epsilon \\ 0 & 0 \end{pmatrix} \leq \Delta A < \begin{pmatrix} \infty & \infty \\ \infty & \infty \end{pmatrix}, \\ (0 \quad 0) \leq \Delta c < (\infty \quad \infty). \end{aligned}$$

By setting Δz to 10 rather than zero in (24), one obtains ranges within which perturbations do not increase the optimal value more than 10, and so forth.

Like branch and bound duality, inference duality is computationally impractical if the branch and bound tree is too large, although it requires fewer data from the tree. It does not provide an explicit approximation of the value function as superadditive and branch and bound duality do. However, only inference duality provides easily computable sensitivity ranges, not only for right-hand sides but for all problem data.

Conclusions

Table 3 summarizes the properties of the various duals. The surrogate and Lagrangian duals are used primarily

for computational purposes, because they provide independent bounds on the optimal value. The remaining duals are useful for sensitivity analysis. The superadditive and branch and bound duals provide a more complete analysis of right-hand side sensitivity. The latter requires a branch and bound solution of the problem but considerably less computation. Inference duality requires a branch and bound solution and provides only sensitivity ranges, but ranges can be obtained for all problem data by solving linear programming problems.

One can also formulate a dual based on congruence relations [21] that is not discussed here. H.P. Williams provides an interesting discussion of this and some other duals (surrogate, Lagrangian, superadditive) in [22]. General treatments of Lagrangian and superadditive duality may be found in [17,19], with a brief discussion of surrogate duality in the former. Excellent presentations of Lagrangian duality appear in [6] and [1, Chap. 6].

See also

- ► Decomposition Techniques for MILP: Lagrangian Relaxation
- ► Integer Programming
- ► Integer Programming: Algebraic Methods
- ► Integer Programming: Branch and Bound Methods
- ► Integer Programming: Branch and Cut Algorithms
- ► Integer Programming: Cutting Plane Algorithms
- ► Integer Programming: Lagrangian Relaxation
- ► Simplicial Pivoting Algorithms for Integer Programming
- ► Time-dependent Traveling Salesman Problem

References

1. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: Theory and algorithms, 2nd edn. Wiley, New York
2. Blair CE, Jeroslow R (1977) The value function of a mixed integer program 1. Discret Math 19:121–138
3. Blair CE, Jeroslow R (1982) The value function of a mixed integer program. Math Program 23:237–273
4. Dawande M, Hooker JN Inference-based sensitivity analysis for mixed intetger linear programming. Oper Res (to appear).
5. Everett III H (1963) Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Oper Res 11:399–417
6. Fisher ML (1981) The Lagrangean relaxation method for solving integer programs. Managem Sci 27:1–18
7. Geoffrion AM (1974) Lagrangean relaxation for integer programming. Math Program Stud 2:82–114
8. Giles FR, Pulleyblank WR (1979) Total dual integrality and integer polyhedra. Linear Alg Appl 25:191–196
9. Glover F (1968) Surrogate constraints. Oper Res 16:741–749
10. Glover F (1975) Surrogate constraint duality in mathematical programming. Oper Res 23:434–451
11. Greenberg H, Pierskalla WP (1970) Surrogate mathematical programming. Oper Res 18:924–939
12. Hooker JN (1992) Generalized resolution for 0–1 linear inequalities. Ann Math Artificial Intelligence 6:271–286
13. Hooker JN (1996) Inference duality as a basis for sensitivity analysis. In: Freuder EC (ed) Principles and Practice of Constraint Programming-CP96. no. 1118 of Lecture Notes Computer Sci. Springer, Berlin, pp 224–236
14. Jeroslow RG (1978) Cutting plane theory: Algebraic methods. Discret Math 23:121–150
15. Johnson EL (1973) Cyclic groups, cutting planes and shortest paths. In: Hu TC, Robinson S (eds) Mathematical Programming. Acad. Press, New York, pp 185–211
16. Karwan MH, Rardin RL (1979) Some relationships between Lagrangean & surrogate duality in integer programming. Math Program 17:320–334
17. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
18. Schrage L, Wolsey LA (1985) Sensitivity analysis for branch and bound integer programming. Oper Res 33:1008–1023
19. Schrijver A (1986) Theory of linear and integer programming. Wiley, New York
20. Tind J, Wolsey LA (1981) An elementary survey of general duality theory in mathematical programming. Math Program 21
21. Williams HP (1984) A duality theorem for linear congruences. Discrete Appl Math 7:93–103
22. Williams HP (1996) Duality in mathematics and linear and integer programming. J Optim Th Appl 90:257–278
23. Wolsey LA (1981) The b-hull of an integer program. Discrete Appl Math 3:193–201
24. Wolsey LA (1981) Integer programming duality: Price functions and sensitivity analysis. Math Program 20:173–195

Integer Programming: Lagrangian Relaxation

J. N. HOOKER
Graduate School of Industrial Admin.,
Carnegie Mellon University, Pittsburgh, USA

MSC2000: 90C10, 90C30

Article Outline

Keywords
Purpose of Relaxation
Lagrangian Relaxation
The Lagrangian Dual
Integer Programming
Solving the Dual
Further Reading and Extensions
See also
References

Keywords

Lagrangian relaxation; Integer programming; Lagrangian dual; Lagrange multipliers; Branch and bound

Relaxation is important in optimization because it provides bounds on the optimal value of a problem. One of the more popular forms of relaxation is Lagrangian relaxation, which is used in integer programming and elsewhere.

A problem is relaxed by making its constraints weaker, so that the feasible set is larger, or by approximating the objective function. In the case of a minimization problem, the optimal value of the relaxation is a lower bound on the optimal value of the original problem. For a maximization problem it is an upper bound. The art of relaxation is to design a relaxed problem that is easy to solve and yet provides a good bound.

Purpose of Relaxation

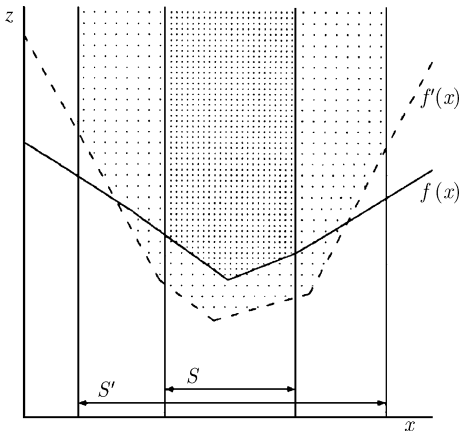
Relaxation bounds are useful for two reasons. First, they can indicate whether a suboptimal solution is close to the optimum. If a minimization problem, for example, is hard to solve, one might settle for a suboptimal solution whose value is close to a known lower bound. An optimal solution would not be much better.

Second, relaxation bounds are useful in accelerating a search for an optimal solution. In a solution of an integer programming problem, for example, one normally solves a relaxation of the problem at each node of the *branch and bound* tree. Suppose again that the objective is to minimize. If the value of the relaxation at some node is greater than or equal to the value of a feasible solution found earlier in the search, then there is no point in branching further at that node. Any optimal solution found by branching further will have a value no better than that of the relaxation and therefore no better than that of the solution already found. Lagrangian relaxation is often used in this context, because it may provide better bounds than the standard linear programming (LP) relaxation.

Lagrangian Relaxation

Lagrangian relaxation is named for the French mathematician J.L. Lagrange, presumably due to the occurrence of what we now call Lagrange multipliers in his calculus of variations [2]. Because this form of relaxation changes the objective function as well as enlarging the feasible set, it is necessary to broaden the concept of relaxation somewhat.

Consider the problem of minimizing a function $f(x)$ subject to $x \in S$, where x is a vector of variables and S the set of feasible solutions. The *epigraph* of the problem is the set of all points (z, x) for which $x \in S$ and $z \geq f(x)$. This is illustrated in Fig. 1. The problem of minimizing $f'(x)$ subject to $x \in S'$ is a *relaxation* of the original problem if its epigraph contains the epigraph of the original problem. That is, a) $S \subset S'$ and $b)$ $f(x) \leq f'(x)$ for all $x \in S$. Relaxation is therefore conceived as enlarging the epigraph; enlarging the feasible set is a special case. It is clear that the optimal value of a relaxation still provides a lower bound on the optimal value of the original problem.



Integer Programming: Lagrangian Relaxation, Figure 1
Epigraph of an optimization problem min {$f(x)$: $x \in S$} (darker shaded area) and of a relaxation min {$f'(x)$: $x \in S'$} (darker and lighter shaded areas)

Lagrangian relaxation is available for problems in which some of the constraints are inequalities or equations. Such problems may be written as

$$\text{minimize } f(x) \tag{1}$$

$$\text{subject to } g(x) \leq 0 \tag{2}$$

$$x \in S. \tag{3}$$

Here, $g(x)$ is a vector of functions $(g_1(x), \ldots, g_m(x))$, and (2) is a family of m constraints $g_i(x) \leq 0$. There is no loss of generality in omitting equality constraints $h_i(x) = 0$ from this formulation, because they can be written as two inequality constraints, $h_i(x) \leq 0$ and $-h_i(x) \leq 0$. The constraints (3) may take any form, inequality or otherwise.

The *Lagrangian relaxation* is formed by 'dualizing' the constraints (2):

$$\begin{cases} \min & f(x) + \lambda g(x) \\ & x \in S. \end{cases} \tag{4}$$

Here, $\lambda = (\lambda_1, \ldots, \lambda_m)$ is a vector of nonnegative *Lagrange multipliers* that correspond to the inequality constraints. The aim of dualization is to remove the hardest constraints from the constraint set, so that the relaxed problem is relatively easy to solve.

The Lagrangian relaxation is in fact a relaxation because its epigraph contains the epigraph of the original problem (1)-(3). This can be verified by checking conditions a) and b):

a) The feasible set of the original problem is a subset of the feasible set of the relaxation, because the relaxation omits some of the original constraints.
b) If x is feasible in the original problem, then $f(x) \geq f(x) + \lambda\, g(x)$. This is because $\lambda \geq 0$ and, due to the feasibility of x, $g(x) \leq 0$.

The Lagrangian Dual

A relaxation can be constructed simply by eliminating the constraints (2) rather than dualizing them. One might ask what is the advantage of dualization. One rationale is that when the Lagrange multipliers are properly chosen, the penalties $\lambda_i g_i(x)$ in the objective function hedge against infeasibility. To the extent that constraints $g_i(x) \leq 0$ are violated and the bound thereby weakened, the objective function will be penalized, restoring the quality of the bound.

Fortunately one can search for a proper choice of multipliers. The Lagrangian relaxation is actually a 'family' of relaxations, parameterized by the vector λ of multipliers. This provides the possibility of searching over values of λ to find a relaxation that gives a good lower bound on the optimal value.

The problem of finding the best possible relaxation bound is the *Lagrangian dual* problem. If $\theta(\lambda)$ is the optimal value of the relaxation (4), the Lagrangian dual of (1)-(3) is the problem of maximizing $\theta(\lambda)$ subject to $\lambda \geq 0$.

Under certain conditions the best relaxation bound is equal to the optimal value of the original problem (1)-(3) [1]. Generally, however, it falls short. The amount by which it falls short is the *duality gap*.

The Lagrangian dual problem has three attractive features:

- It need not be solved to optimality. Any feasible solution provides a valid lower bound.
- Its objective function $\theta(\lambda)$ is always a concave function of λ. One need only find a local maximum, which is necessarily a global maximum as well.
- Its solutions have a *complementary slackness* property. If certain λ_i's are positive in an optimal solution of the dual problem, then the corresponding constraints $g_i(x) \leq 0$ are satisfied as equations in some optimal solution of the primal problem (1)-(3).

A serious drawback of the Lagrangian dual is that simply evaluating the objective function $\theta(\lambda)$ for a given λ normally requires solution of an optimization problem. The relaxation must be carefully chosen so that this is practical. Moreover the function θ is typically nondifferentiable.

Why is the Lagrangian dual a 'dual'? Perhaps because it generalizes the LP dual, which is the Lagrangian dual of an LP problem. To see this, consider the LP problem $\min \{cx: Ax \geq a, x \geq 0\}$. Its Lagrangian dual is to maximize

$$\theta(\lambda) = \min_{x \geq 0} \{cx + \lambda(a - Ax)\}$$
$$= \min_{x \geq 0} \{(c - \lambda A)x + \lambda a\}$$

over $\lambda \geq 0$. So $\theta(\lambda)$ is $-\infty$ if some component of $c - \lambda A$ is negative and is λa otherwise. This means that

maximizing $\theta(\lambda)$ over $\lambda \geq 0$ is equivalent to maximizing λa subject to $\lambda A \leq c$ and $\lambda \geq 0$, which is precisely the LP dual.

The duality relationship holds more convincingly, however, between two problem-solving strategies: solution of strengthenings and solution of relaxations [12]. Methods that solve strengthenings include branching methods, local search heuristics, and other techniques that enumerate solutions or partial solutions by fixing some or all of the variables. Solution of each strengthening provides an upper bound on the optimal value, and the goal is to find the smallest upper bound. If the search is exhaustive, the smallest upper bound is equal to the optimal value.

The dual strategy is to solve relaxations of the problem in order to find the largest possible lower bound on the optimal value. There is no obvious way to enumerate relaxations, however, unless they are somehow parametrized, in which case one can enumerate values of the parameters. The Lagrangian dual is one way of doing this but by no means the only. Another is the *surrogate dual* [7,8,9], in which the relaxed constraint set is a nonnegative linear combination of inequality constraints, and relaxations are parametrized by the vector of multipliers in the linear combination. The dual approach also differs from the primal in that an exhaustive enumeration normally does not guarantee that the best bound obtained is equal to the optimal value. There is usually a duality gap.

Integer Programming

The application of Lagrangian ideas to integer programming dates back at least to H. Everett [4]. In this arena the optimization problem (1)-(3) becomes,

$$\begin{cases} \min & cx \\ \text{s.t.} & Ax \leq a \\ & Bx \leq b \\ & x_j \text{ integer for all } j. \end{cases} \tag{5}$$

The 'hard' constraints $Ax \leq a$ are dualized in the Lagrangian relaxation,

$$\begin{cases} \min & cx + \lambda(Ax - a) \\ \text{s.t.} & Bx \leq b \\ & x_j \text{ integer for all } j, \end{cases} \tag{6}$$

and $\theta(\lambda)$ is the minimum value of this problem for a given λ. The optimal value z_{LD} of the Lagrangian dual is a lower bound on the optimal value z_{IP} of (5). It will be seen shortly that the bound z_{LD} is at least as good as the bound z_{LP} obtained by solving the LP relaxation of (5). (The LP relaxation is the result of dropping the integrality constraints.)

In the context of integer programming, the Lagrangian function $\theta(\lambda)$ is not only concave but piecewise linear. This is because $\theta(\lambda)$ is the maximum of a set of linear functions $cx + \lambda(Ax - a)$ over all integral values of x that satisfy $Bx \leq b$.

A fundamental property of the Lagrangian dual is that z_{LD} is equal to the optimal value z_C of

$$\begin{cases} \min & cx \\ \text{s.t.} & Ax \leq a \\ & x \in \text{conv}(S), \end{cases} \tag{7}$$

where S is the set of integer points satisfying $Bx \leq b$, and conv(S) is the convex hull of S [6]. The Lagrangian dual can therefore be written as an LP problem, if a linear description of conv(S) is available.

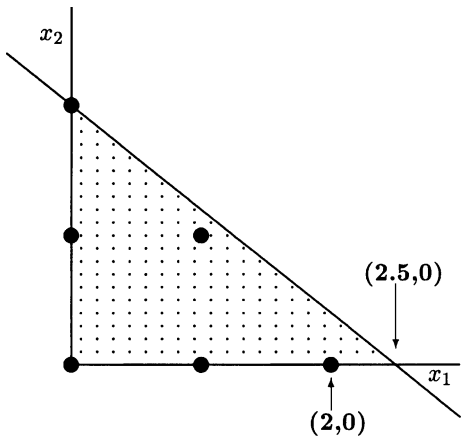
The reasoning behind this fact goes as follows. Because the feasible set of (7) is that of an LP problem, the optimal value of its Lagrangian dual is equal to z_C. To see that it is also equal to z_{LD}, thereby proving $z_C = z_{LD}$, it suffices to show that the Lagrangian relaxation of (7) always has the same optimal value as the Lagrangian relaxation of (5). But this is true because the former is the same problem as the latter, except that the constraints in former are $x \in \text{conv}(S)$ and in the latter are $x \in S$. This substitution has no effect on the optimal value because the objective function is linear.

It can now be seen that the bound z_{LD} is always at least as good as z_{LP}. Let C_{IP} be the problem (7) corresponding to (5), and let C_{LP} be the problem (7) corresponding to the LP relaxation of (5). C_{LP} 's feasible set contains that of C_{IP}, and its optimal value is therefore less than or equal to z_{LD}. But because C_{LP} is identical to (5) 's LP relaxation, $z_{LP} \leq z_{LD}$.

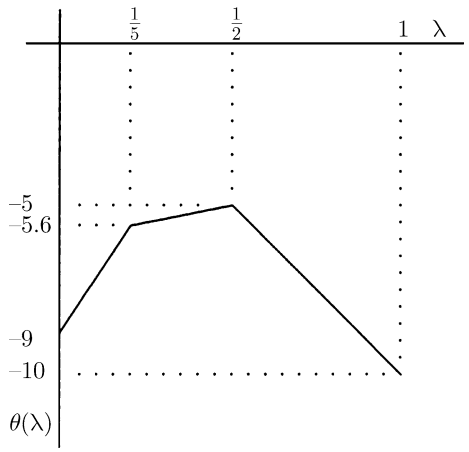
When $Bx \leq b$ happens to describe a polyhedron whose vertices have integral coordinates, C_{IP} and C_{LP} are the same problem. In this case $z_{LD} = z_{LP}$.

To sum up,

$$z_{LP} \leq z_{LD} = z_C \leq z_{IP},$$



Integer Programming: Lagrangian Relaxation, Figure 2
Feasible set of an integer programming problem (large dots) and its linear programming relaxation (area shaded by small dots). The point (2, 0) is the optimal solution, and (2.5, 0) is the solution of the LP relaxation



Integer Programming: Lagrangian Relaxation, Figure 3
The Lagrangian function $\theta(\lambda)$ for an integer programming problem. The optimal value of the Lagrangian dual problem is $\theta(1/2) = -5$

where the first inequality is an equation when $Bx \leq b$ describes an integral polyhedron.

As an example consider the integer programming problem (Fig. 2):

$$\begin{cases} \min & -2x_1 - x_2 \\ \text{s.t.} & 4x_1 + 5x_2 \leq 10 \\ & 0 \leq x_j \leq 3 \\ & x_j \text{ integer}, \ j = 1, 2. \end{cases} \tag{8}$$

The optimal solution is $x = (2, 0)$, with value $z_{IP} = -4$. Dualizing the first constraint decouples the variables:

$$\begin{aligned} \theta(\lambda) &= \min_{\substack{0 \leq x_j \leq 3 \\ x_j \text{ integer}}} \{-2x_1 - x_2 + \lambda(4x_1 + 5x_2 - 10)\} \\ &= \min_{\substack{0 \leq x_j \leq 3 \\ x_j \text{ integer}}} \{(4\lambda - 1)x_1 + (5\lambda + 1)x_2 - 10\lambda\}. \end{aligned}$$

Because of the decoupling, $\theta(\lambda)$ is easily computed:

$$\theta(\lambda) = \begin{cases} 17\lambda - 9 & \text{if } 0 \leq \lambda \leq \frac{1}{5}, \\ 2\lambda - 6 & \text{if } \frac{1}{5} \leq \lambda \leq \frac{1}{2}, \\ -10\lambda & \text{if } \lambda \geq \frac{1}{2}. \end{cases}$$

It is evident in Fig. 3 that θ is a concave, piecewise linear function. The optimal value of the Lagrangian dual is $z_{LD} = \theta(1/2) = -5$, resulting in a duality gap of $z_{IP} - z_{LD} = 1$. The optimal value of the LP relaxation is likewise -5, so that in the present case $z_{LP} = z_{LD}$. This is predictable because $Bx \leq b$ consists of the bounds $0 \leq x_j \leq 3$, which define an integral polyhedron.

In practical applications, the Lagrangian relaxation is generally constructed so that it can be solved in polynomial time. It might be a problem in which the variables can be decoupled, as in the above example, or whose feasible set is an integral polyhedron. Popular relaxations include assignment or transportation problems, which can be solved quickly.

A notable example is the *traveling salesman problem* on n cities:

$$\text{minimize} \quad \sum_{ij} c_{ij} x_{ij} \tag{9}$$

subject to

$$\sum_j x_{ij} = 1, \quad \text{all } i, \tag{10}$$

$$\sum_i x_{ij} = 1, \quad \text{all } j, \tag{11}$$

$$\sum_{i \notin V} \sum_{j \in V} x_{ij} \geq 1, \text{all nonempty } V \subset \{2, \ldots, n\}, \tag{12}$$

$$x_{ij} \geq 0, \ x_{ij} \text{ integral, all } i, j. \tag{13}$$

If the assignment constraints (10) are dualized, the Lagrangian relaxation minimizes

$$\sum_{ij} c_{ij}x_{ij} + \sum_i \lambda_i(\sum_j x_{ij} - 1) = \sum_{ij}(c_{ij} + \lambda_i)x_{ij} - \sum_i \lambda_i$$

subject to (11)-(13). This is equivalent to finding a minimum-cost spanning arborescence that is rooted at node 1, which can be done in polynomial time [3]. Alternatively, the Lagrangian relaxation can be solved as an LP problem without the integrality constraints, because the same optimal value results [3]. See [10,14] for a survey of efforts along this line.

Solving the Dual

Subgradient optimization is a popular method for solving the Lagrangian dual, because subgradients of θ (and gradients when they exist) can be readily calculated.

Let $X(\overline{\lambda})$ be the set of optimal solutions of the Lagrangian relaxation (4) when $\lambda = \overline{\lambda}$. If $X(\overline{\lambda})$ is a singleton $\{\overline{x}\}$, then the gradient of θ at $\overline{\lambda}$ is simply the vector $g(\overline{x})$. This is because for values of λ in a neighborhood of $\overline{\lambda}$, $\theta(\lambda)$ is the linear function $f(\overline{x}) + \lambda g(\overline{x})$.

More generally, for any $\overline{x} \in X(\overline{\lambda})$, $g(\overline{x})$ is a subgradient of θ at $\overline{\lambda}$. In fact, every subgradient of θ at $\overline{\lambda}$ is a convex combination of subgradients that correspond to the solutions in $X(\overline{\lambda})$.

In the integer programming case, the subgradients of θ at $\overline{\lambda}$ are $A\overline{x} - a$ for each $\overline{x} \in X(\overline{\lambda})$, and convex combinations thereof. Consider the example (8), where

$$X(\lambda) = \begin{cases} \{(3,3)\} & \text{if } 0 \le \lambda < 1/5, \\ \{(3,3),(3,0)\} & \text{if } \lambda = 1/5, \\ \{(3,0)\} & \text{if } 1/4 < \lambda < 1/2, \\ \{(3,0),(0,0)\} & \text{if } \lambda = 1/2, \\ \{(0,0)\} & \text{if } \lambda > 1/2. \end{cases}$$

Thus at $\lambda = 0$, θ has the gradient (slope) of $4(3) + 5(3) - 10 = 17$. At $\lambda = 1/5$, the subgradients of θ are 17, 2, and their convex combinations; i. e., all slopes in the interval [2, 17]. This can be seen in Fig. 3.

Further Reading and Extensions

A lucid geometrical exposition of Lagrangian duality may be found in [1, Chap. 6]. A widely read treatment of its application to integer programming is [5]. A recent tutorial is [11], which also surveys methods for solving the dual. [13, Sect. III.2.6] describes some methods for strengthening the Lagrangian relaxation. There is a vast literature on applications and enhancements.

The idea of the Lagrangian dual need not be limited to the use of Lagrange multipliers. A dual problem can be solved over any parametrized family of relaxations. The dual problem might be solved by a local search heuristic over the parameter space.

See also

- Branch and Price: Integer Programming with Column Generation
- Decomposition Techniques for MILP: Lagrangian Relaxation
- Integer Linear Complementary Problem
- Integer Programming
- Integer Programming: Algebraic Methods
- Integer Programming: Branch and Bound Methods
- Integer Programming: Branch and Cut Algorithms
- Integer Programming: Cutting Plane Algorithms
- Integer Programming Duality
- Lagrange, Joseph-Louis
- Lagrangian Multipliers Methods for Convex Programming
- LCP: Pardalos-Rosen Mixed Integer Formulation
- Mixed Integer Classification Problems
- Multi-objective Integer Linear Programming
- Multi-objective Mixed Integer Programming
- Multi-objective Optimization: Lagrange Duality
- Multiparametric Mixed Integer Linear Programming
- Parametric Mixed Integer Nonlinear Optimization
- Set Covering, Packing and Partitioning Problems
- Simplicial Pivoting Algorithms for Integer Programming
- Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- Stochastic Integer Programs
- Time-dependent Traveling Salesman Problem

References

1. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: Theory and algorithms, 2nd edn. Wiley, New York

2. Boyer CB (1985) A history of mathematics. Princeton Univ. Press, Princeton
3. Edmonds J (1967) Optimum branchings. J Res Nat Bureau Standards (B) 71:233–240
4. Everett III H (1963) Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Oper Res 11:399–417
5. Fisher ML (1981) The Lagrangean relaxation method for solving integer programs. Managem Sci 27:1–18
6. Geoffrion AM (1974) Lagrangean relaxation for integer programming. Math Program Stud 2:82–114
7. Glover F (1968) Surrogate constraints. Oper Res 16:741–749
8. Glover F (1975) Surrogate constraint duality in mathematical programming. Oper Res 23:434–451
9. Greenberg H, Pierskalla WP (1970) Surrogate mathematical programming. Oper Res 18:924–939
10. Grötschel M, Padberg MW (1985) Polyhedral theory. In: Lawler EL et al (eds) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York, pp 251–305
11. Guinard M (1995) Lagrangean relaxation: A short course. JORBEL 35:5–21
12. Hooker JN (2000) Logic-based methods for optimization: Combining optimization and constraint satisfaction. Wiley, New York
13. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
14. Padberg MW, Grötschel M (1985) Polyhedral computations. In: Lawler EL et al (eds) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York, pp 307–360

Integrated Planning and Scheduling

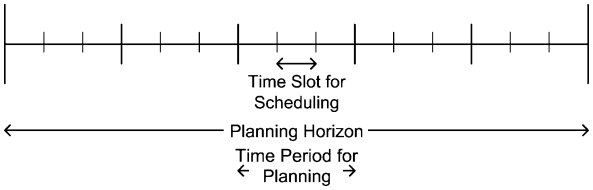
Metin Türkay, Uğur Kaplan
Department of Industrial Engineering,
Koç University, Istanbul, Turkey

Article Outline

Introduction
  Representation of Time Slots
Problem Definition
Solution Strategy
Conclusions
References

Introduction

Integrated planning and scheduling of events in production systems are among the most important factors that affect the efficient operation of these systems. The main objective of integrated planning and scheduling is the allocation of resources that change dynamically over the time domain, and the coordination of the activities that are required to satisfy customer demand. One difference between planning and scheduling is the time scale over which the actions for these factors are triggered. During the planning phase, allocation of available resources and satisfaction of the given demand are made for a medium-term horizon that is usually expressed in months. During the scheduling phase, however, short-term allocation of available resources and the timing of the production of specific orders are the main decisions and these involve time scales of days to weeks.



Integrated Planning and Scheduling, Figure 1
Time relation between planning and scheduling

During both planning and scheduling phases, the time horizon to be planned is divided into time slots. The main difference between planning and scheduling problems in terms of execution times is the length of time scales. Whereas the time scale for the planning phase is longer, the time scales for the scheduling phase are shorter as shown in Fig. 1.

For a realistic case, the time period for the planning phase is measured in weeks or months, therefore, total production and the inventory of each product at the end of each time period are the only performance issues for the system. For scheduling, however, the length of the time slots is measured in hours. Therefore in addition to the production quantity, the production sequence of each product becomes important during scheduling .

Traditionally planning and scheduling have been performed separately on the shop floor, but this decomposition of two activities leads to decreased efficiency of the operations performed in the production centers. Moreover, medium-term plans may result in infeasible projections if they are made without consideration of

short-term performance requirements, and conversely, short-term plans may cause myopic decisions without consideration of long-term performance issues.

Ideally, an integrated model is required in which planning and scheduling can be considered simultaneously. This model should include both medium-term capacity utilization and production level values and short-term production sequence and machine assignment decisions. Due to simultaneous consideration of medium- and short-term decisions, the representation of time slots is one of the primary concerns in integrated planning and scheduling problems.

Representation of Time Slots

The first decision that should be taken during the modeling of the planning and scheduling of a system is the representation of time. Time can be represented in integrated planning and scheduling problems using discrete or continuous formulations [2]. The choice of time representation is related to whether an event in the system can take place only at predefined times or at any instant. If the events can take place only at predefined times, then a discrete time representation is required.

Discrete Time Model: In a discrete time model, the whole planning horizon is divided into predefined time intervals. This representation assumes that an event can occur only at the boundaries of each time interval [4]. Therefore, during the solution of the model, no planning horizons, other than the boundaries of a finite number of time slots, are considered, and this simplification makes the model more tractable. To express the exact behavior of the system during the planning horizon, however, the length of the time slots should be kept as short as possible, which may cause an explosive increase in the number of variables. On the other hand, increasing the length of time slots may give infeasible results.

Continuous Time Model: In the continuous time model an event can occur at any instant within the whole planning horizon. This makes the model more dependable and flexible and the total number of variables decreases. The representation of some constraints becomes more complex, however, and this decreases the tractability of the model [2].

Mixed Time Representation: A mixed time representation that includes both discrete and continuous time has also been studied [3]. In this situation, the time slots are fixed and the durations of the processes are kept constant in discrete time. The durations of the process task are expressed as variables in the mixed time representation. This is accomplished by setting the durations of the process tasks to be multiples of a fixed time grid.

Problem Definition

In this section, the model given by Dogan and Grossmann [1] is examined in detail. Given that several products are to be produced in a single production unit, the planning horizon is divided into planning periods of one week and each week-long planning period is divided into N time slots, where there are N products to be produced. At the end of each week, the demand is determined for each product. The production level in the system is constant and it is also a cost issue for the model. Transition time of production with respect to production sequence is given within this system.

The problem is the determination of products that should be produced each week and the sequencing of the production of these items. During sequencing, the production time, the amount and production duration are determined for each product. In addition, the inventory levels should be determined for each time period.

The MILP Model:

The objective of the model is the maximization of profit.

$$\sum_i \sum_t p_{it} S_{it} - c_{\text{inv}} \sum_i \sum_t Area_{it} - \sum_i \sum_t c_{it}^{\text{oper}} X_{it} - \sum_i \sum_k \sum_l \sum_t c_{ik}^{\text{trans}} Z_{iklt} - \sum_t \sum_i \sum_k c_{ik}^{\text{trans}} TRT_{ikt} \quad (1)$$

In the objective function, p_{it} is the price of product i in time period t, S_{it} is the amount of sales of product i in time period t, c_{inv} is the inventory cost, $Area_{it}$ is the area below the inventory versus time graph for product i at time period t, c_{it}^{oper} is the operating cost for product i in time period t, X_{it} is the amount of the product i produced in time period t, c_{ik}^{trans} is the transition cost from product i to product k, Z_{iklt} is a binary variable that indicates that production of product i is followed by product k in time slot l of the time period t, and TRT_{ikt} is

a binary variable that denotes production of product i is followed by product k at the end of time period t.

The first term of the objective function gives the revenue generated, the second term is the inventory cost, the third term is the total operation cost, the fourth term is the transition cost within each week, and the last term is the transition cost between each week.

The assignment of production orders and corresponding processing times for each order are given by the following:

$$\sum_i W_{ilt} = 1 \quad \forall l, t \tag{2}$$

$$0 \le \tilde{\theta}_{ilt} \le H_t W_{ilt} \quad \forall i, l, t \tag{3}$$

$$\theta_{it} = \sum_l \tilde{\theta}_{ilt} \quad \forall i, t \tag{4}$$

$$\tilde{X}_{ilt} = r_i \tilde{\theta}_{ilt} \quad \forall i, l, t \tag{5}$$

$$X_{it} = \sum_l \tilde{X}_{ilt} \quad \forall i, t \tag{6}$$

In Eqs. (2)–(6), W_{ilt} is a binary variable that denotes the production of product i in the time slot l of the time period t, $\tilde{\theta}_{ilt}$is the production time of product i, in time slot l of the time period t, H_t is the length of time period t, X_{it} is the total production time of the product i in the time period t, and $\tilde{X}_{ilt}$is the production amount of product i in time slot l of the time period t.

Equation (2) states that only one product can be produced in the each time slot. According to Eq. (3), the production time of product i, at time slot l of the time period t will be zero, if this product is not assigned to time slot l of time period t. Equation (4) states that the total production time of the product i, at time period t is equal to the sum of the production time of this product during time slots of the corresponding time period. Equation (5) represents the total production amount of product i, during time slot l of the time period t, and Eq. (6) calculates the total production of product i, during the time period t.

The transition from one product to another product is expressed by the following constraint:

$$Z_{iklt} \ge W_{ilt} + W_{k,l+1;t} - 1 \quad \forall i, k, l, t \tag{7}$$

Equation (7) ensures that if product k is produced after product i, in the time period l, then Z_{iklt} will be equal to 1, otherwise it will be 0.

An important consideration is the starting and ending times for each task in the production schedule. The following constraints are used to calculate them:

$$Te_{lt} = Ts_{lt} + \sum_i \tilde{\theta}_{ilt} + \sum_i \sum_k \tau_{ik} Z_{iklt} \quad \forall l, t \tag{8}$$

$$TRT_{ikt} \ge W_{ilt} + W_{k,ll,t+1} - 1 \quad \forall i, k, t, l = N, ll = 1 \tag{9}$$

$$Te_{lt} + \sum\sum \tau_{ik} TRT_{ikt} = Ts_{llt+1} \quad \forall t, l = N, ll = 1 \tag{10}$$

$$Te_{lt} = Ts_{l+1t} \quad \forall l \ne N, t \tag{11}$$

$$Te_{Nt} \le HT_t \quad \forall t \tag{12}$$

In Eq. (8), Te_{lt} is the end of the time slot l of the time period t, Ts_{lt} is the start time of the time slot l of the time period t and τ_{ik} is the transition time between product i and product k. Equation (8) states that the end of time slot l of time period l is equal to the start time plus the total processing time for the products produced in that slot and the total transition time. In Eq. (9), TRT_{ikt} is equal to 1, if at the end of period t production of product i takes place, and at the beginning of the time period t+1 production of product k takes place.

Equations (10) and (11) ensure the connectivity between consecutive time periods. In Eq. (12), HT_t is the length of the time period t and Eq. (12) also ensures that the end time of the last time slots in each time period cannot be greater than the length of the corresponding time period.

The inventory levels for each product are updated with the following constraints:

$$INV_{it} = INV_{i0} + \sum_l r_i \tilde{\theta}_{ilt} \quad \forall i, t = 1 \tag{13}$$

$$INV_{it} = INVO_{it-1} + \sum_l r_i \tilde{\theta}_{ilt} \quad \forall i, t \ne 1 \tag{14}$$

$$INVO_{it} = INV_{it} - S_{it} \quad \forall i, t \tag{15}$$

$$Area_{it} \ge INVO_{it-1} H_t + r_i \theta_{it} H_t \quad \forall i, t \tag{16}$$

In Eq. (13), INV_{it} is the inventory level of the product i at time period t and INV_{i0} is the initial in-

ventory of product i. Equation (13) updates the inventory of each product in the first time period. In Eq. (14), $INVO_{it-1}$ is the inventory of product i at time period t–1 after demand of that product is satisfied. Equation (14) establishes the inventory and production quantity relationship for all periods other than initial time period. According to Eq. (15), the amount of inventory of product i after demand of it is satisfied is equal to total inventory of product i minus sales of this product for each time period. In Eq. (16) $Area_{it}$ is the area below the inventory versus time plot for product i at time period t. As the exact area is nonlinear the equation overestimates this area.

The demand for products is incorporated into the integrated planning and scheduling model with the following constraints:

$$S_{it} \geq d_{it} \quad \forall i, t \tag{17}$$

$$NY_{it} = \sum_{l} W_{ilt} \quad \forall i, l, t \tag{18}$$

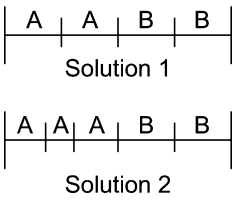
$$YOP_{it} \geq W_{ilt} \quad \forall i, l, t \tag{19}$$

$$YOP_{it} \leq NY_{it} \leq NYOP_{it} \quad \forall i, t \tag{20}$$

$$NY_{it} \geq N - \left[\left(\sum_{i} YOP_{it}\right) - 1\right] - M\,(1 - W_{i1t}) \quad \forall i, t \tag{21}$$

$$NY_{it} \leq N - \left[\left(\sum_{i} YOP_{it}\right) - 1\right] - M\,(1 - W_{i1t}) \quad \forall i, t \tag{22}$$

In Eqs. (17)–(22), the d_{it} is the demand for the product i at time period t, NY_{it} is the number of time slots during which product i is produced in the time period t, YOP_{it} is a binary variable that shows whether product i is produced during time period t and M is a sufficiently large number. In Eq. (17), the lower bound of demand satisfied is ensured. In Eq. (18), the number of time slots during which product i is produced within the time period t is found. Equation (19) ensures that if production of product i at time slot l of time period t takes place, then YOP_{it} will be equal to 1. In Eqs. (21) and (22) the occurrence of a degenerate solution is prevented. According to the model formulation, if production of a product takes place in more than one time slot in the given time period, these time slots should be consecutive. If the production takes place in non-consecutive time slots, then the solution obtained will be suboptimal because of the existence of transition costs. The reason for degeneracy in this consecutive production is illustrated in Fig. 2.



Integrated Planning and Scheduling, Figure 2
An example of degenerate solution

As seen in Fig. 2 both solutions give the same objective value since the total production time of all products is the same and therefore, without Eqs. (21) and (22) the model formulation will be highly degenerate.

Solution Strategy

The integrated planning and scheduling model presented in Eqs. (1)–(22) gives rise to a complex mixed-integer programming problem. In the model, inventory and demand satisfaction trends are observed weekly because these two performance issues are planned in the medium-term time horizon. Timing and assignment constraints, however, are applied for each time slot within a week. An important assumption in this model is the production center. It is assumed that there is only one production center that carries out all of the planning and scheduling activities but the model is intractable even for this specific case of the problem. This intractability is not specific to this formulation as most planning and scheduling models that include realistic details are intractable. Two approaches are considered for addressing the intractability of the problem. In the first approach, an integrative model is first formulated and then, with respect to some defined criterion, a decomposition scheme is applied to the model. In the second approach, a simple modeling technique such as single-item capacitated or incapacitated lot sizing is formulated and then a superposition of all the en-

tire simple models is derived to apply to the planning and scheduling scheme.

In the first approach, both planning and scheduling problems are considered in an integrative manner and the quality of the result depends on the decomposition scheme. The aim of the decomposition is to decrease of the intractability of the model by removing from the formulation those details that make the model computationally complex. These details are the scheduling constraints that ensure feasibility on the shop floor. In this approach, an iterative solution is applied to make the problem computationally tractable while generating feasible schedules.

According to this method, an integrative planning and scheduling model is first constructed and then the details that make the model difficult to solve are removed or aggregated on the time domain or with respect to common parts. This high-level model is solved to obtain a temporary solution that is then applied with respect to the scheduling criteria. If the obtained solution is feasible on the shop floor, it is accepted. Otherwise, the system returns to the high-level model and changes it to make it to produce a feasible plan. The success of the results of the high-level plan depends on the accuracy of this model. If the model is not accurate enough to produce a feasible solution, much iteration can be done, although this is undesirable for the dependability and tractability of the model.

To avoid the direct solution of the integrated MILP planning and scheduling model, a bilevel decomposition algorithm that applies a hierarchical decomposition scheme has been proposed by Dogan and Grossmann [1]. In this scheme, the original model is decomposed into two separate models. The first model is the upper-level planning model that determines the products to be produced and the level of production and inventory in each time period. The second model is the lower-level planning and scheduling model that is modeled initially. In the lower-level problem, the original model formulation is solved by only applying it to the products that the upper-level planning model has decided to produce.

The upper-level planning problem is an MILP model and it is used to predict an upper bound for the original model formulation. This is obtained by ignoring the detailed sequencing constraint that is important for scheduling model. The result of the lower-level model, which is obtained by using the result of the upper-level model, creates a lower bound for the global optimum. As the lower-level model which is solved with respect to the result of high-level planning problem is a sub-problem of the original model, the result produced is feasible.

The proposed algorithm is applied in an iterative manner. The upper bound found by the upper-level planning problem and the lower-bound found by the detailed planning and scheduling model are compared. If the difference between these bounds is less than some predefined tolerance value, the algorithm terminates. Otherwise, some integer and logic cuts are added to the upper-level planning model to obtain a more refined solution for the original model.

Zhu and Majozi [6] proposed another decomposition algorithm to address the intractability of a planning and scheduling model for multipurpose batch plants. They classified the economic concerns of the model as part of the planning problem and sequencing concerns of the model as a part of the scheduling problem. The proposed decomposition scheme of the detailed planning and scheduling problem is based on the assumption of a block angular structure for the model. With respect to the block angular structure of the model, there should be two types of blocks in the model. The first type is the constraints that are common for all plants in the system. In the multi-plant model, the first blocks are concerned with the allocation of resources, such as raw materials and labor. The second type of blocks concern the set of constraints that are specific to each plant. These second-type blocks intersect with the sequence of resources allocated by the constraints of the first block. In the context of the planning and scheduling problem, the first type of blocks represent the planning problem and the second type of blocks represent the scheduling problem for each plant, separately. The proposed solution lies in the extraction of the block angular structure of the integrated model. After this step, the model is decomposed into two separate parts: The first part consists of only planning blocks. Within these constraints, the allocations of common resources are planned so that they can be classified as part of the model for the planning problem. The second part consists of only scheduling blocks, which include separate sets of constraints for each plant. In this second part the aim is to obtain the

optimal schedule for each plant and therefore it is classified as a scheduling problem.

To apply this decomposition scheme successfully, the integrated model must satisfy two main conditions. The first condition is the convexity of the constraints. If that is the case, the feasible region of the integrated model will form a convex set. Therefore, the schedules obtained from the separate scheduling block problems for a given optimal resource allocation will provide the global optimal solution of the integrated model. The second condition to be satisfied is the block angular decomposability of the integrated model formulation. It is necessary to show that all scheduling constraints can be separated into a non-intersecting set of constraints in order to satisfy this condition. This makes the scheduling part of the decomposition scheme, plant specific. On the shop floor this structure can be achieved if manufacturing of the products begin and end in the same plant.

After the model has been decomposed, an iterative solution approach is applied to get the feasible and near optimal schedule. First the planning problem that includes A blocks is solved and the allocation of the common resources to the each plant is obtained. Then this result is incorporated into separate scheduling models to produce detailed schedules for each plant. If the results of the scheduling problems do not match the targets of the planning problem, the planning problem is resolved with the results obtained from scheduling problem. This procedure continues until the differences between the results of the planning problem and scheduling problems converge to a small threshold value.

In the second approach, superposition of the simple and frequently studied models such as lot-sizing models is used. Pocket and Wolsey [5] proposed a single item decomposition method to deal with the intractability problem. In this method, a capacitated lot-sizing problem is solved for each finished product individually. This model is solved to satisfy the demand for the end product and the inventory and production quantity of the end product are also monitored during the solution process. The proposed model is solved only for single product and multi-product cases, however, and as some products may share common resources, there may be infeasibilities when the schedules are combined. The superpositioning of individual models is required in order to address this problem. The bill of materials (BOM) should be used during the superposition since each end product is produced by using many intermediate products. The single-item decomposition technique begins by solving a capacitated lot-sizing model for each product, which is the master production scheduling (MPS) model [5]. After the solution of the MPS, the batch size of each product during the planning horizon is obtained. The batch sizes are decomposed into batches of intermediate products by using the BOM. A rough cut capacity planning (RCCP) is executed in parallel to roughly check the feasibility of the MPS model with respect to the capacity available. In cases of infeasibility, the MPS model is revisited or the capacity of the rare resource is increased.

Conclusions

The execution of planning and scheduling tasks separately on the shop floor causes infeasible and suboptimal decisions. To prevent these problems, it is necessary to optimize the planning and scheduling tasks simultaneously but models in which planning and scheduling are integrated can become computationally intractable because of the highly complex structure of the model. To deal with this problem, two approaches can be used. In the first approach the detailed planning and scheduling model is decomposed into simpler ones, and these models are solved iteratively with the detailed original model until a feasible solution that has satisfactory objective value is obtained. In contrast, in the second approach, a single model is solved for each product individually and then superposition of these models is used to determine the feasibility of the results with respect to some shared resources.

References

1. Erdirik MD, Grossmann IE A (2006) Decomposition Method for the Simultaneous Planning and Scheduling of Single Stage Continuous Multiproduct Plants. Ind Eng Chem Res 45:299–315
2. Floudas CA, Lin X (2004) Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. Comput Chem Eng 28:2109–2129
3. Maravelias CT (2006) Mixed-Time Representation for the State-Task Network Models. Ind Eng Chem Res 44:9129–9145
4. Mendez CA, Cerda J, Grossmann IE, Harjunkoski I, Fahl M (2006) State-of-the-art review of optimization methods for

short-term scheduling of batch processes. Comput Chem Eng 30:913–946
5. Pochet Y, Wolsey LA (2006) Production Planning by Mixed Integer Programming. Springer, New York
6. Zhu XX, Majozi T (2001) Novel Continuous Time MILP Formulation for Multipurpose Batch Plants. 2. Integrated Planning and Scheduling. Ind Eng Chem Res 40:5621–5634

Interior Point Methods for Semidefinite Programming

Levent Tunçel
Department of Combinatorics and Optimization
Faculty of Mathematics, University of Waterloo,
Waterloo, Canada

MSC2000: 90C51, 90C22, 90C25, 90C05, 90C30

Article Outline

Keywords and Phrases
Preliminaries
Path-Following Algorithms
Potential Reduction Algorithms
Step Size
Search Directions
References

Keywords and Phrases

Interior-point methods; Positive semidefinite matrices; Barrier function; Potential function; Central path; Path-following; Newton's method

The problem of optimizing a linear function of an unknown symmetric positive semidefinite matrix, subject to finitely many linear equations and inequalities, is called the *semidefinite programming* (SDP) problem. As is clear from the above definition, the SDP problem is a generalization of the very well known linear programming (LP) problem.

Parallel to the great success of LP in applications, at the time of this writing, there is a tremendous amount of activity and excitement regarding the applications of SDP. Even at these early stages of the development, the applications are far-reaching.

The SDP problem brings together many fields of engineering, computer science and mathematics. The theory and practice of SDP both draw from and contribute to a very large number of fields. For applications in combinatorial optimization see [4], for eigenvalue optimization see [8], and for applications in engineering, system and control theory see [22]. Also see the special issue of *Mathematical Programming* dedicated to SDP [14], as well as the *Handbook on Semidefinite Programming* [18] and the proceedings of a Fields Institute workshop [15].

Preliminaries

Let a symmetric $n \times n$ matrix X with real entries be given. Then X is *positive semidefinite*, denoted $X \succeq 0$, if $u^T X u \geq 0$ for all $u \in \mathbb{R}^n$. X is *positive definite*, denoted $X \succ 0$, if $u^T X u > 0$ for all $u \in \mathbb{R}^n \backslash \{0\}$. To define linear functions of this variable X, one equips the space of $n \times n$ matrices with an inner-product: $\langle C, X \rangle$ denotes the trace of $(C^T X)$. Using this notation, one can define a specific form of the SDP problem. Let a symmetric $n \times n$ matrix C, a column vector b, and m symmetric $n \times n$ matrices $A_1, A_2, \ldots, A_m$ be given. Then the following is the primal form of SDP:

$$(P) \quad \min \langle C, X \rangle ,$$
$$\langle A_i, X \rangle = b_i, \ i \in \{1, \ldots, m\} ,$$
$$X \succeq 0 .$$

Any SDP problem can be put into the above form. The dual of (P) can be defined (similarly to the LP dual) as follows:

$$(D) \quad \max b^T y ,$$
$$\sum_{i=1}^{m} y_i A_i + S = C,$$
$$S \succeq 0 .$$

Without loss of generality, it can be assumed that the matrices $A_1, A_2, \ldots, A_m$ are linearly independent. If they are linearly dependent, then either the system $\langle A_i, X \rangle = b_i, i \in \{1, \ldots, m\}$ has no solution, or there are some redundant equations which can be eliminated. In the first case, (P) is infeasible. In the second case, all redundant equations and corresponding A_i, b_i can be eliminated, to arrive at an equivalent problem satisfying the assumption. Under this assumption, for any solution, (y, S), of the equation $\sum_{i=1}^{m} y_i A_i + S = C$, the S part of the solution uniquely identifies y. Sometimes, in interior-point algorithms, it is convenient to refer only to S when one mentions a feasible solution of (D).

Even though one writes "min" and "max" in the definitions of (P) and (D), the optimum values of these problems may not be attained, and even if they are attained, primal and the dual objective values may not equal each other. Thus, the duality theory for SDP is quite a bit more complicated than that for LP. (See [16] and the references therein, for a discussion of various duals and duality theorems.)

However, if we assume that there exists

$$X^{(0)} \succ 0$$

such that

$$\langle A_i, X^{(0)} \rangle = b_i, \ i \in \{1, \ldots, m\}$$

and that there exists $(y^{(0)}, S^{(0)})$ such that

$$\sum_{i=1}^{m} y_i^{(0)} A_i + S^{(0)} = C, \ S^{(0)} \succ 0 \,,$$

then the duality theory guarantees that the optimum values of both problems (P) and (D) are attained and that they are equal.

Now, we introduce the very general notion of *interior-point methods*. These are the methods of solving convex optimization problems by generating a sequence which lies in the relative interior of a convex set defined by the "difficult" constraints. In this definition, one envisions an abstract formulation of convex optimization problems in which one has a maximal set of linear equality constraints and some convex set constraint. The convex set constraint is the "difficult" constraint.

The basic idea of interior-point methods goes back at least to Frisch [3] (1950s) and to Fiacco and McCormick [2] (1960s). However, the current modern interior-point algorithms have their origins in Karmarkar's groundbreaking work [6].

The general definition above does not refer to the fact that much of the interest is in the algorithms that can be proven to generate approximately optimal solutions in polynomially many iterations in the dimension of the problem and a desired accuracy, prescribed as a part of the input. The amount and the type of work required per iteration will be described shortly. Certain practical variants of such interior-point algorithms turn out to be very fast and robust for a wide range of applications. Indeed, as in just like any other optimization problem, the algorithms which perform very well in extensive computational tests spark great interest for theoretical investigations in the area to further our comprehension of the efficiency of the interior-point methods as well as our understanding of the degree of difficulty of certain SDP problems for interior-point approaches.

At the time of this writing, the most popular algorithms are the primal–dual ones. These algorithms work almost equally hard in improving both the primal and the dual solutions. However, for certain applications, primal-only (or dual-only) algorithms, which work almost exclusively on the primal problem and use the dual to only generate bounds on the optimum value, are indispensable. This is usually due to the special structure of the problem at hand. The main ingredients of interior-point methods for SDP will be illustrated for primal-dual algorithms.

Interior-point algorithms can be classified with respect to many criteria. A rather obvious criterion is the initial iterate $(X^{(0)}, y^{(0)}, S^{(0)})$. All interior-point algorithms start with $X^{(0)} \succ 0$, $S^{(0)} \succ 0$, and keep all iterates positive definite. If the algorithm allows $X^{(0)}$ or $(y^{(0)}, S^{(0)})$ not to satisfy the corresponding equality constraints, then the algorithm is called an *infeasible-start interior-point algorithm*. In this article, the illustration of the details of the methods will be done mostly for feasible starting points.

If X is feasible in (P) and (y, S) is feasible in (D), then

$$\langle C, X \rangle - b^T y = \langle X, S \rangle \geq 0 \,.$$

In the above, $\langle X, S \rangle = 0$ if and only if both X and (y, S) are optimal in their respective problems. For the above reasons, $\langle X, S \rangle$ is called the *duality gap*.

Next, some important concepts used in the making of an interior-point algorithm are mentioned. Once the initial iterate, $(X^{(0)}, y^{(0)}, S^{(0)})$, such that $X^{(0)} \succ 0$, $S^{(0)} \succ 0$, is given, one needs to generate a *search direction* (d_x, d_y, d_s) along which to move. Then a *step size* $\alpha > 0$ describing how much to move in the given search direction must be determined. Practical methods allow two different step sizes: one for the primal iterate $X^{(k)}$, and the other for the dual iterate $(y^{(k)}, S^{(k)})$. Below, the main steps of a very basic interior-point algorithm for SDP are given. For this description and the rest of the article a common step size α, for both iterates, is used.

The way in which the search directions and the step sizes are determined has a very significant impact on

```
InputInstance(C, A_1, ..., A_m, b, X^(0), y^(0), S^(0), ε);
iteration counter k := 0;
WHILE ⟨X^(k), S^(k)⟩ > ε DO
    find a search direction (d_x, d_y, d_s);
    find a step size α > 0 such that
        X^(k) + α d_x ≻ 0, S^(k) + α d_s ≻ 0;
    set (X^(k+1), y^(k+1), S^(k+1)) :=
        (X^(k), y^(k), S^(k)) + α(d_x, d_y, d_s);
    set k := k + 1;
  END{WHILE};
OUTPUT(X^(k), y^(k), S^(k))
```

Interior Point Methods for Semidefinite Programming, Algorithm 1
Main steps of an interior-point algorithm

the performance of interior-point methods. Two of the main theoretical foundations for choosing the search direction are mentioned below.

Path-Following Algorithms

Many practical algorithms are influenced by this foundation. One first picks a *barrier function*, $F(X)$. This is a function which is defined on the set of symmetric, positive-definite matrices such that for every sequence of matrices from this set, converging to a point on the boundary of the set, the value of F tends to infinity. For improved theoretical and/or practical performance, one must enforce further conditions on the barrier function. For the purposes of this article, the barrier function (which does possess many of the such desired properties) is

$$F(X) := -\ln \det X \,,$$

the negation of the logarithm of the determinant of X. Consider the family of optimization problems parameterized by $\mu > 0$:

$$\begin{aligned} (P_\mu) \quad \min \; & \langle C, X\rangle + \mu F(X) \,, \\ & \langle A_i, X\rangle = b_i, \forall i \,, \\ & X \succeq 0 \,. \end{aligned}$$

The unique minimizer of this optimization problem defines a point $(X(\mu), y(\mu), S(\mu))$ on the *primal–dual central path*. Here, $y(\mu)$ and $S(\mu)$ represent the dual variables for the equality and the semidefiniteness constraints of (P_μ), respectively. As $\mu \to 0$, $(X(\mu), y(\mu), S(\mu))$ converges to the optimal solutions of (P) and (D).

The primal–dual central path can be expressed more explicitly. $(X(\mu), y(\mu), S(\mu))$ is the unique solution of the following system

$$\begin{aligned} & \langle A_i, X\rangle = b_i, \quad \forall i, \; X \succ 0 \,, \\ & \sum_{i=1}^{m} y_i A_i + S = C \,, \\ & S = \mu X^{-1} \,. \end{aligned}$$

Path-following algorithms choose the search direction to approximately follow this path. They are usually based on *Newton's method* or are related to it. For a review of such search directions, see [7] and [19].

Potential Reduction Algorithms

For these algorithms, one defines a *potential function*, based on a barrier function, to measure how good a given point is with respect to the duality gap and the proximity to the central path:

$$\phi(X, S) := (n + q)\ln\langle X, S\rangle + F(X) + F(S) \,.$$

One chooses $q := \Theta(\sqrt{n})$ for (current) the best theoretical complexity results and $q := \Theta(n)$ or larger for better practical performance. In this setting, the search directions are usually obtained by computing a *steepest-descent direction* for the potential function $\varphi(X, S)$ and projecting it onto the appropriate linear subspace to satisfy the equality constraints in (P) and (D).

Step Size

Once the search direction (d_x, d_y, d_s) is computed, then a practical interior-point algorithm usually calculates α as a constant fraction of the maximum step size that keeps the next iterate positive definite. In a path-following algorithm for robustness of the performance, or good theoretical results, one might want to confine all (or some) of the iterates into a neighborhood of the central path. In a potential reduction algorithm, one usually chooses the value of α as the minimizer of φ along the given search direction. There are many other possibilities, see the potential reduction survey [20] and the references therein (such as [11] and [12]).

Currently, interior-point methods provide the fastest algorithms in theory with respect to the worst-case complexity bounds proven so far. In the current

practice, interior-point methods also provide the fastest and most robust solution techniques for general SDP problems. One such theoretical result can be summarized as follows.

Theorem: *Let $X^{(0)}, y^{(0)}, S^{(0)}$, a feasible solution of (P) and (D), and $\epsilon > 0$ such that*

$$n \ln\left(\frac{\langle X^{(0)}, S^{(0)}\rangle}{n}\right) + F(X^{(0)}) + F(S^{(0)}) \leq \sqrt{n}\ln(1/\epsilon)$$

be given. Then (certain variants of) the potential-reduction interior-point algorithm described above will generate in $O\left(\sqrt{n}\ln(1/\epsilon)\right)$ iterations feasible interior points X, S such that

$$\langle X, S\rangle \leq \epsilon \langle X^{(0)}, S^{(0)}\rangle .$$

In the above theorem, the function

$$n \ln\left(\frac{\langle X, S\rangle}{n}\right) + F(X) + F(S)$$

is a proximity measure. It is nonnegative for every pair of interior points (X, S). It is equal to zero if and only if (X, S) lies on the central path (for $\mu := \langle X, S\rangle/n$).

Alizadeh [1], and Nesterov and Nemirovskii [10] independently generalized interior-point methods to SDP problems. The above theorem is a specialization of a more general result for *convex programming* problems from [10]. A similar result was independently obtained for SDP [1].

Search Directions

In obtaining search directions, both path-following and potential reduction approaches end up with some linear system of equations, defined by the input of the problem, the current iterate $(X^{(k)}, S^{(k)})$ and some other parameters of the algorithm. The resulting system has the following structure:

$$\langle A_i, d_x\rangle = r_i^{(P)}, \quad \forall i ,$$
$$\sum_{i=1}^{m}(d_y)_i A_i + d_s = r^{(D)} , \quad \mathcal{E}d_x + \mathcal{F}d_s = r^{(XS)} ,$$

where $r^{(P)}$ is an m-vector representing the primal residual (infeasibility of $X^{(k)}$), $r^{(D)}$ is an $n \times n$ symmetric matrix representing the dual residual (infeasibility of $(y^{(k)}, S^{(k)})$), and $\mathcal{E}$ and $\mathcal{F}$ are linear operators on $n \times n$ symmetric matrices. The operators $\mathcal{E}$ and $\mathcal{F}$ vary from one algorithm to the next and depend on the current iterate, as well as some other parameters of the underlying algorithm. The choice of $\mathcal{E}$ and $\mathcal{F}$ can have profound theoretical and/or practical effects on the performance of the algorithm. Finally, $r^{(XS)}$ denotes an $n \times n$ symmetric matrix, a residual related to the desired value of $\langle X, S\rangle$ for the next iterate as well as the desired value of the proximity measure for the next iterate.

In solving such systems to determine the search directions, many tools from numerical analysis become relevant. Moreover, one must exploit the existing structure of the problem at hand to be able to efficiently solve the large-scale instances.

As shown in [10], many of the fundamental ideas of interior-point methods can be applied to general convex programming problems. For primal–dual interior-point algorithms for general convex programming problems, see also [9] and [21]. The most general theoretical results in this direction rely on the existence of very special barrier functions for every convex set (see [10], also see [5] for connections of interior-point methods to many other branches of mathematics via the barrier functions for SDP and more general convex optimization problems).

Another important issue is that of the initial point and the detection of infeasibility in interior-point methods. For such problems, the quality of the initial point and the value of the *condition measures* (measuring in the input space how far a given instance is from the boundary separating feasible and infeasible instances, etc. [17]) for the given instance are good attributes for evaluating the performance of the methods.

References

1. Alizadeh F (1995) Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM J Optim 5:13–51
2. Fiacco AV, McCormick GP (1990) Nonlinear programming sequential unconstrained minimization techniques. SIAM, Philadelphia
3. Frisch KR (1955) The logarithmic potential method of convex programming. Memorandum, University Institute of Economics, Oslo
4. Goemans MX (1997) Semidefinite programming in combinatorial optimization. Math Programm 79:143–161

5. Güler O (1996) Barrier functions in interior point methods. Math Oper Res 21:860–885
6. Karmarkar N (1984) A new polynomial time algorithm for linear programming. Combinatorica 4:373–395
7. Kojima M, Shida M (1999) Shindoh S: Search directions in the SDP and the monotone SDLCP: Generalization and inexact computation. Math Programm 85:51–80
8. Lewis AS, Overton ML (1996) Eigenvalue optimization. Acta Numerica 7:149–190
9. Nemirovskii AS, Tunçel L (2005) "Cone-free" primal-dual path-following and potential reduction polynomial time interior-point methods. Math Programm 102:261–294
10. Nesterov YE, Nemirovskii AS (1994) Interior point polynomial methods in convex programming: Theory and algorithms. SIAM, Philadelphia
11. Nesterov YE, Todd MJ (1998) Primal-dual interior-point methods for self-scaled cones. SIAM J Optim 8:324–364
12. Nesterov YE, Todd MJ (1997) Self-scaled barriers and interior-point methods for convex programming. Math Oper Res 22:1–42
13. Nesterov YE, Todd MJ, Ye Y (1999) Infeasible-start primal-dual methods and infeasibility detectors for nonlinear programming problems. Math Programm 84:227–267
14. Overton ML, Wolkowicz H (eds) (1997) Semidefinite programming. Math Programm 77
15. Pardalos PM, Wolkowicz H (1998) Topics in semidefinite and interior-point methods. Fields Institute Commun 18
16. Ramana MV, Tunçel L, Wolkowicz H (1997) Strong duality for semidefinite programming. SIAM J Optim 7:641–662
17. Renegar J (1995) Linear programming, complexity theory and elementary functional analysis. Math Programm 70:279–351
18. Saigal R, Vandenberghe L, Wolkowicz H (2000) Handbook on semidefinite programming. Kluwer, Dordrecht
19. Todd MJ (1999) A study of search directions in primal-dual interior-point methods for semidefinite programming. Optim Methods Softw 11/12:1–46
20. Todd MJ (1997) Potential-reduction methods in mathematical programming. Math Programm 76:3–45
21. Tunçel L (2001) Generalization of primal-dual interior-point methods to convex optimization problems in conic form. Found Comput Math 1:229–254
22. Vandenberghe L, Boyd S (1996) Semidefinite programming. SIAM Rev 38:49–95

Interval Analysis: Application to Chemical Engineering Design Problems

MARK A. STADTHERR
Department Chemical Engineering,
University Notre Dame, Notre Dame, USA

MSC2000: 65C20, 65G20, 65G30, 65G40, 90C90, 65H20

Article Outline

Keywords
Process Simulation
Process Optimization
Process Synthesis
Conclusion
See also
References

Keywords

Interval analysis; Process design; Process simulation; Process synthesis; Process optimization

Interval analysis can provide valuable tools in several aspects of chemical process design, including steady-state process simulation and optimization, and the initial synthesis and screening of process alternatives. The discussion below highlights the use of interval analysis in these areas. For a general description of problems and issues in chemical engineering design, see [8] and [7].

Process Simulation

Process simulators are used to compute the performance of a chemical process given its design (the *process simulation* problem), or to compute a design that meets given performance specifications (the *process design* problem). In either case, the central problem in steady-state process simulation is the solution of an $n \times n$ system of nonlinear algebraic equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, where n may be very large (hundreds of thousands or more) and the equation system represents a mathematical model of the process, including material and energy balances, thermodynamic equilibrium relationships, and other equations needed to describe the process.

For solving the process model, Newton and quasi-Newton methods are widely used, but may not reliably converge, especially since a good initial guess is often hard to obtain. To improve convergence in these circumstances, various approaches have been used. These include trust region techniques, such the *dogleg method* [4], and *homotopy continuation* methods (e. g., [10]).

An additional difficulty is that in process simulation there are invariably upper and lower bounds on the variables, $\mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U$, violation of which may cause some functions to become undefined. Bounds are often dealt with in an ad hoc manner involving truncation or reflection of the correction step. A more natural way of dealing with bounds is to use a mathematical programming approach (e. g., [1]), in which the bounds become an integral part of the problem. While a number of the techniques noted above demonstrate excellent global convergence properties in practice, none offer a rigorous mathematical guarantee of convergence.

A further difficulty in solving the nonlinear equation systems arising in process simulation is that they may have multiple solutions. With the exception of homotopy based methods, none of the techniques mentioned above are designed for finding multiple solutions when they exist. While in practice homotopy based methods are frequently able to locate all solutions to a problem, they offer no guarantee that all solutions have been found, except in special cases.

All of the difficulties noted above, namely the lack of good initial guesses, the presence of variable bounds, and the possibility of multiple solutions, can be dealt with using interval analysis. For example, R.E. Swaney and C.E. Wilhelm [11] use a technique, based on repeated solution of linear programs, which, through the use of bounds generated using interval analysis within a branch and bound framework, provides rigorous global convergence to a solution of the process model.

C.A. Schnepper and M.A. Stadtherr [5] use an *interval Newton* approach. This can rigorously enclose any and all solutions to the process model, and is essentially initialization independent, since it requires only initial intervals for the variables, and some of these bounds may be specified as part of the problem. Both serial and parallel implementations are described in [5], and provision is made for efficient handling of sparse matrices. Several example problems were successfully solved, ranging in size from 3 to 177 variables, including problems with multiple solutions. Performance on the larger problems was unpredictable, with two problems of over one hundred variables being solved very efficiently, even with very large initial bounds on the variables, but one problem of 50 variables being unsolvable due to excessive computation time. However, for this 50-variable problem, once smaller, more intelligently chosen (using knowledge of boiling points and critical temperatures) initial intervals were used, the problem was easily and efficiently solved.

Process Optimization

Perhaps the most natural formulation for a process design problem is as an optimization problem. The process simulation problem is then viewed as an optimization problem with zero degrees of freedom. A typical process optimization problem features a nonlinear objective function, nonlinear equality constraints (the process model), nonlinear inequality constraints, and upper and lower bounds on variables. Frequently these nonlinear programming problems are nonconvex as well, prompting interest in global optimization techniques to deal with the potential for multiple extrema.

Several approaches to global optimization in process engineering have been proposed, including both deterministic and nondeterministic methods. Among the deterministic techniques used are branch and bound, cutting plane, primal-dual decomposition, and interval analysis. The work of R. Vaidyanathan and M.M. El-Halwagi [9] provides a good example of the use of interval analysis in this context. This is an interval branch and bound approach that is guaranteed to yield the global solution. The procedure is accelerated by using a 'distrust region' method for eliminating infeasible portions of the search space and by use of local methods for some purposes.

R.P. Byrne and I.D.L. Bogle [3] and Byrne [2] also use an interval branch and bound approach, but treat the interval lower bounding process as a convex programming problem. In [2] it is also shown how this interval-based approach can be applied in the context of modular process optimization software. Since modular software predominates commercially, this work is of particular interest.

Process Synthesis

Before the process simulation and optimization problems discussed above can be formulated, it is necessary to synthesize and screen process alternatives. These provide the base case problems for later process simulation and optimization studies. When there is uncertainty in design specifications, or when the design spec-

ification covers a range of values, then interval analysis can be a particularly useful tool in *process synthesis*.

For example, see [6] for a problem involving the processing of high level nuclear waste. In this problem, the waste to be processed is characterized by intervals of composition, as are the requirements for a stable glass product. In [6], an interval propagation scheme that exploits the structure of the problem and a simple process model is developed, and it is demonstrated how to use this to screen process alternatives and to infer other knowledge about the process design.

Conclusion

Interval analysis provides tools that can be used to solve process simulation problems with complete reliability, providing a method that can guarantee with mathematical and computational certainty that the correct result is found, and thus eliminating computational problems that are encountered with conventional techniques. The method is essentially initialization independent, deals with variable bounds naturally, and also guarantees the enclosure of multiple solutions if present. In process optimization, similar guarantees can also be provided that the global extremum has been found. There are many other problems in chemical process design, for instance in many aspects of process synthesis, that likewise are amenable to solution using this powerful approach.

See also

- ► Automatic Differentiation: Point and Interval
- ► Automatic Differentiation: Point and Interval Taylor Operators
- ► Bilevel Programming: Applications in Engineering
- ► Bounding Derivative Ranges
- ► Design Optimization in Computational Fluid Dynamics
- ► Global Optimization: Application to Phase Equilibrium Problems
- ► Interval Analysis: Differential Equations
- ► Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ► Interval Analysis: Intermediate Terms
- ► Interval Analysis: Nondifferentiable Problems
- ► Interval Analysis: Parallel Methods for Global Optimization
- ► Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ► Interval Analysis: Systems of Nonlinear Equations
- ► Interval Analysis: Unconstrained and Constrained Optimization
- ► Interval Analysis: Verifying Feasibility
- ► Interval Constraints
- ► Interval Fixed Point Theory
- ► Interval Global Optimization
- ► Interval Linear Systems
- ► Interval Newton Methods
- ► Multidisciplinary Design Optimization
- ► Multilevel Methods for Optimal Design
- ► Optimal Design of Composite Structures
- ► Optimal Design in Nonlinear Optics
- ► Structural Optimization: History

References

1. Bullard LG, Biegler LT (1991) Iterative linear programming strategies for constrained simulation. Comput Chem Eng 15:239–254
2. Byrne RP (1997) Global optimisation in process design. PhD Thesis Dept. Chemical and Biochemical Engin. Univ. College London
3. Byrne RP, Bogle IDL (1996) An accelerated interval method for global optimization. Comput Chem Eng 20:S49–S54
4. Chen HS, Stadtherr MA (1981) A modification of Powell's dogleg method for solving systems of nonlinear equations. Comput Chem Eng 5:143–150
5. Schnepper CA, Stadtherr MA (1996) Robust process simulation using interval methods. Comput Chem Eng 20:187–199
6. Schug BW, Realff MJ (1998) Analysis of waste vitrification product-process systems. Comput Chem Eng 22:789–800
7. Seider WD, Seader JD, Lewin DR (1999) Process design principles: Synthesis, analysis and evaluation. Wiley, New York
8. Turton R, Bailie RC, Whiting WB, Shaeiwitz JA (1998) Analysis, synthesis, and design of chemical processes. Prentice-Hall, Englewood Cliffs, NJ
9. Vaidyanathan R, El-Halwagi M (1994) Global optimization of nonconvex nonlinear programs via interval analysis. Comput Chem Eng 18:889–897
10. Wayburn TL, Seader JD (1987) Homotopy continuation methods for computer-aided process design. Comput Chem Eng 11:7–25
11. Wilhelm CE, Swaney RE (1994) Robust solution of algebraic process modeling equations. Comput Chem Eng 18:511–531

Interval Analysis: Differential Equations

RAMON E. MOORE
Worthington, USA

MSC2000: 65G20, 65G30, 65G40, 65L99

Article Outline

Keywords
See also
References

Keywords

Interval analysis; Differential equation

Optimization can involve *differential equations*, for instance in the formulation of constraints. Interval analysis provides methods for computing interval-valued functions, for example polynomials with interval coefficients, guaranteed to contain solutions to differential equations. Methods have been developed for initial and boundary value problems for both ordinary (ODE) and partial (PDE) differential equations [1]–[32].

For the initial value problem in ODEs, the Cauchy–Peano approach of classical analysis can be made into a constructive method using interval analysis. With interval arithmetic and interval extensions to standard functions, we can computationally verify sufficient conditions for existence of solutions, as well as construct upper and lower bounds on solutions. The techniques of automatic differentiation provide for efficient use of and (using interval arithmetic) bounding of remainder terms in Taylor series expansions, making interval *Taylor series* an effective method for initial value problems in ODEs [5,13,21]. See especially [33].

Many problems in differential equations, both initial and boundary value problems for ODEs and PDEs, can be reformulated as integral equations. Interval analysis provides means for using fixed-point theory and Picard–Lindelöf-type iteration constructively on such reformulations [3,4,9,15], [17]–[29].

For initial value problems, it was noticed early on [12] that local coordinate transformations are often needed to prevent excessive growth ('the wrapping effect') of the widths of interval enclosures. It has been a continuing project to improve on such transformations [6,8,12,13,14,16,17,21,28,30,32,33].

Variable-precision interval computation provides a means of controlling computational error in ill-posed problems [1,2,7,8]. Using interval methods, we can, in principle (with enough computing), find solutions to prescribed accuracy for differential equations [28].

Some examples will illustrate the kinds of results obtainable by interval methods.

Example 1 A problem that occurs in chemical reactor theory involves the differential equation

$$y'' + \frac{1}{x}y' + be^{\left(-\frac{1}{|y|}\right)} = 0, \quad 0 \le x \le 1, \quad b > 0,$$

with boundary values $y'(0) = 0$ and $y(1) = t > 0$.

It turns out there may be one or more solutions depending on the values of t and b. Using interval methods, it can be proved easily [20] that, for every t, $b > 0$, we have

$$y(t) \in t + b\left[e^{-\frac{1}{t}}\frac{(1-x^2)}{4}, \frac{(1-x^2)}{4}\right]$$

for all solutions y and all $0 \le x \le 1$.

Example 2 Consider the nonlinear hyperbolic PDE

$$u_{xy} = 1 + (u_x + u_y)u$$

with initial conditions $u(x, 0) = 0$ and $u(0, y) = 0$.

Using interval methods, we can prove [23] that for all $0 \le x \le 0.5$ and all $0 \le y \le 0.5$, the solution $u(x, y)$ is contained in the interval-valued function

$$xy + [0.1666, 0.2144](x^2y^3 + x^3y^2).$$

This means that we have guaranteed lower and upper bounds on the solution, namely

$$xy + 0.1666(x^2y^3 + x^3y^2) \le u(x, y) \le xy + 0.2144(x^2y^3 + x^3y^2)$$

for all x and y in $[0, 0.5]$.

Example 3 Consider the initial value problem [21]

$$x' = ct^2 + x^b \quad \text{with} \quad x(0) = a.$$

Suppose there is uncertainty about the values of a, b and c, and all we know is that $0 \le a \le 0.1$, $0.2 \le b \le 0.38$, $3.3 \le c \le 3.6$.

In a single iteration of the integral equation representation of the problem, we find that every solution satisfies

$$1.1t^3 \le x(t) \le 0.1 + t + 1.2t^3$$

for all $t \in [0, 0.6]$ and all a, b, c in the given ranges.

Moving coordinate systems for initial value problems help reduce the growth of interval bounds from one expansion step to the next. The *wrapping effect* arises from a rotation of the vector field associated with the differential equations. Such a rotation cannot be followed by interval vector bounds, which are boxes with faces parallel to the coordinate planes. This wrapping effect can be partially controlled in a number of ways with varying degrees of success. R.E. Moore [16] suggested the use of the 'connection matrix'. F. Krückeberg [12] devised an algorithm he called the *3PM process*. R.J. Lohner [13] suggests the use of parallepipeds and also QR-decomposition for matrix transformations of enclosing hyper-rectangles.

Among the sample programs given in [10 Appendix D] there is 'AWA', (AnfangsWert Aufgabe) for initial value problems ([10, pp. 248–251]). AWA implements the ideas of Lohner concerning control of the wrapping effect. Five options are provided:

0) interval vector;
1) parallelepiped;
2) QR-decomposition;
3) intersections of 0) and 1);
4) intersections of 0) and 2).

For an example using AWA, we considered a *Volterra model of conflicting populations*

$$x_1' = 2x_1(1 - x_2),$$
$$x_2' = -x_2(1 - x_1),$$

with initial conditions $x_1(0) = 1$ and $x_2(0) = 3$.

Automatic differentiation software is incorporated in AWA which automatically introduces auxiliary variables $T_1, T_2, \ldots$ as needed, and derives recursion relations for generating Taylor coefficients $(x_1)_k$ and $(x_2)_k$, $k \ge 1$, line-by-line from a compiler code list, see [31]. For the example above, the automatically derived recursion relations for derivatives of any order would look like the following:

$$\begin{cases} T_1 = 1 - x_2, & \text{so} \quad (T_1)_k = -(x_2)_k, \\ T_2 = x_1 T_1, & \text{so} \quad (T_2)_k = \sum_{j=0}^{k} (x_1)_j (T_1)_{k-j}, \\ (x_1)_1 = 2T_2, & \text{so} \quad (x_1)_{k+1} = \frac{2}{k+1}(T_2)_k, \\ T_3 = 1 - x_1, & \text{so} \quad (T_3)_k = -(x_1)_k, \\ T_4 = x_2 T_3, & \text{so} \quad (T_4)_k = \sum_{j=0}^{k} (x_2)_j (T_3)_{k-j}, \\ (x_2)_1 = -T_4, & \text{so} \quad (x_2)_{k+1} = -\frac{1}{k+1}(T_4)_k. \end{cases}$$

The interval Taylor expansion about some t_0 is then, for $i = 1, 2$:

$$(x_i)(t_0 + h) = \left\{ \sum_{j=0}^{K-1} (x_i)_j (t_0) h^j \right\} + R_i h^K,$$

where the remainder coefficients R_i, $i = 1, 2$, are computed from the above recursion relations with *interval inputs* (also found automatically) for $x_1 = (x_1)_0$ and $x_2 = (x_2)_0$.

Using the program AWA given on the diskette for C-XSC, we obtained the following results using the options described.

Using nine terms in the Taylor expansions, $K = 9$, and continuing the solution to $t = 10$, using a relatively large stepsize $h = 0.1$, we obtained different results for two options, both containing the exact solution.

Option 0) produced

$$x_2(10) \in [0.347636\ldots, 0.350002\ldots],$$

whereas option 1) produced the narrower enclosure

$$x_2(10) \in [0.34875\ldots, 0.34888\ldots].$$

The optimal choices of all the various program parameters and the method of coordinate transformation and how they depend on a particular initial value problem are matters that are still being studied. See [5,13,14,33].

J.S. Ely [7] developed a *variable precision interval package* (VPI), using the programming language C++, and applied it to the study of the following ill-posed partial differential equation from the theory of vortex

dynamics ([8]):

$$\frac{\partial z^*(p,t)}{\partial t} = \frac{1}{4\pi i}\text{P.V.}\int B(q)\cot\left[\frac{z(p,t)-z(q,t)}{2}\right]dq$$

with initial condition $z(p, 0) = p$, and the integral taken over $[0, 2\pi]$.

Here, $z(p, t)$ is a 2π periodic, complex function of two real variables p and t, z^* denotes complex conjugation, $B(q) = 1 + A\cos(q)$, and P.V. stands for the Cauchy principal value of the singular integral.

In order to obtain satisfactory results on this difficult problem, in particular to determine the time of onset of turbulence, and to rule out rounding errors as the cause of the observed behavior of the computer simulation, as many as 896 bits were used in the interval arithmetic. To improve efficiency, trigonometric transformations were used for a reformulation of the differential equation. Further speed-ups were obtained using *parallel programming* for a distributed network of computers, and in another version a Cray supercomputer. The results [7,8] settled a long-standing controversy concerning the reliability of the mathematical model in the face of previously unknown effects of rounding error. It is certainly not obvious in advance how many bits are needed for such a difficult problem. The point is that with interval computation we can see, from the widths of interval results, how accurately the answers have been determined. If we have not yet obtained desired accuracy, we can repeat the computations carrying more bits. This can be automated so that, in the words of O. Aberth [1]: 'We expect the computer to do whatever is necessary to obtain such answers'.

See [3] for further discussion of interval methods for differential equations, and some nontrivial applications (e. g.: existence proofs for bifurcations, computer assisted proofs in dynamics, globally convergent domain decomposition methods).

See also

- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Point and Interval Taylor Operators
- ▶ Bounding Derivative Ranges
- ▶ Global Optimization: Application to Phase Equilibrium Problems
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Interval Analysis: Intermediate Terms
- ▶ Interval Analysis: Nondifferentiable Problems
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ▶ Interval Analysis: Systems of Nonlinear Equations
- ▶ Interval Analysis: Unconstrained and Constrained Optimization
- ▶ Interval Analysis: Verifying Feasibility
- ▶ Interval Constraints
- ▶ Interval Fixed Point Theory
- ▶ Interval Global Optimization
- ▶ Interval Linear Systems
- ▶ Interval Newton Methods

References

1. Aberth O (1988) Precise numerical analysis. W.C. Brown, Dubuque, IA
2. Aberth O, Schaefer MJ (1992) Precise computation using range arithmetic, via C++. ACM Trans Math Softw 18(4):481–491
3. Alefeld G, Frommer A, Lang B (eds) (1996) Scientific computing and validated numerics. Akademie, Berlin
4. Caprani O, Madsen K, Rall LB (1981) Integration of interval functions. SIAM J Math Anal 12:321–341
5. Corliss G, Chang YF (1982) Solving ordinary differential equations using Taylor series. ACM Trans Math Softw 8(2):114–144
6. Daniel JW, Moore RE (1970) Computation and theory in ordinary differential equations. Freeman, New York
7. Ely JS (1990) Prospects for using variable precision interval software in C++. PhD Thesis Ohio State Univ.
8. Ely JS, Baker GR (1994) High precision calculations of vortex sheet motion. J Comput Phys 111(2):275–281
9. Fefferman CL, Seco LA (1996) Interval arithmetic in quantum mechanics. In: Kearfott RB, Kreinovich V (eds) Applications of Interval Computations. Kluwer, Dordrecht
10. Klatte R, Kulisch U, Wiethoff A, Lawo C, Rauch M (1993) C-XSC. Springer, Berlin
11. Krückeberg F (1968) Defekterfassung bei gewöhnlichen und partiellen Differentialgleichungen. ISNM, vol 9. Birkhäuser, Basel, pp 69–82

12. Krückeberg F (1969) Ordinary differential equations. In: Hansen E (ed) Topics in Interval Analysis. Clarendon Press, pp 91–97
13. Lohner RJ (1987) Enclosing the solutions of ordinary initial and boundary value problems. In: Kaucher E et al (eds) Computer Arithmetic, Scientific Computation and Programming Languages. Teubner, Leipzig, pp 255–286
14. Lohner RJ (1989) Einschließungen bei Anfangs- und Randwertaufgaben gewöhnlicher Differentialgleichungen. In: Kulisch U (ed) Wissenschaftliches Rechen mit Ergebnisverifikation: Eine Einführung. Viewg, Wlesbaden, pp 183–207
15. Moore RE (1962) Interval arithmetic and automatic error analysis in digital computing. PhD Thesis Stanford Univ.
16. Moore RE (1965) Automatic local coordinate transformations to reduce the growth of error bounds in interval computation of solutions of ordinary differential equations. In: Rall LB (ed) Error in Digital Computation, vol 2. Wiley, New York, pp 103–140
17. Moore RE (1966) Interval analysis. Prentice-Hall, Englewood Cliffs, NJ
18. Moore RE (1969) Functional analysis for computers. In: Funktionalanalytische Methoden der Numerischen Mathematik, ISNM, vol 12. Birkhäuser, Basel, pp 113–126
19. Moore RE (1975) Two-sided approximation to solutions of nonlinear operator equations-a comparison of methods from classical analysis, functional analysis and interval analysis. In: Nickel K (ed) Interval Mathematics, Lecture Notes Computer Sci, vol 29. Springer, Berlin, pp 31–47
20. Moore RE (1978) Bounding sets in function spaces with applications to nonlinear operator equations. SIAM Rev 20:492–512
21. Moore RE (1979) Methods and applications of interval analysis. SIAM, Philadelphia
22. Moore RE (1982) A generalization of the method of upper and lower solutions for integral equations. Nonlinear Anal Th Methods Appl 6(8):829–831
23. Moore RE (1984) A survey of interval methods for differential equations. Proc. 23rd Conf. Decision and Control, IEEE, pp 1529–1535
24. Moore RE (1984) Upper and lower bounds on solutions of differential and integral equations without using monotonicity, convexity, or maximum principles. In: Vichnevetsky R, Stepelman RS (eds) Advances in Computer Methods for Partial Differential Equations V. IMACS, pp 458–462
25. Moore RE (1985) Computational functional analysis. Horwood and Wiley, London
26. Moore RE (1985) Simple simultaneous super- and subfunctions. J Integral Eq 8:165–174
27. Moore RE (1991) Interval tools for computer aided proofs in analysis. In: Meyer KR, Schmidt DS (eds) Computer Aided Proofs in Analysis. Springer, Berlin, pp 211–216
28. Moore RE (1994) Numerical solution of differential equations to prescribed accuracy. Comput Math Appl 28(10-12):253–261
29. Moore RE, Shen Z (1983) An interval version of Chebyshev's method for nonlinear operator equations. Nonlinear Anal Th Methods Appl 7(1):21–34
30. Nickel K (1985) How to fight the wrapping effect. In: Nickel K (ed) Interval Mathematics. Lecture Notes Computer Sci, vol 212. Springer, Berlin, pp 121–132
31. Rall LB (1981) Automatic differentiation: Techniques and applications. Lecture Notes Computer Sci, vol 120. Springer, Berlin
32. Stewart NF (1971) A heuristic to reduce the wrapping effect in the numerical solution of $X' = F(T, X)'$. BIT 11:328–337
33. Website: http://bt.nscl.msu.edu/pub/, work of Berz M and others

Interval Analysis: Eigenvalue Bounds of Interval Matrices

David Hertz
RAFAEL Department 82, Haifa, Israel

MSC2000: 65G20, 65G30, 65G40, 65L99

Article Outline

Keywords
Real Symmetric Interval Matrices
Hermitian Interval Matrices
Complex Interval Matrices
See also
References

Keywords

Complex interval matrix; Real interval matrix; Vertex matrix of an interval matrix; Real symmetric interval matrix; Hermitian interval matrix; Interval of variation of an eigenvalue of an interval matrix; Extreme eigenvalue of an interval matrix

Most of the square matrices appearing in practice undergo quantization that can be coarse and are represented by finite precision numbers. Hence, the underlying unquantized matrices belong to real (complex) interval matrices whose entries are closed intervals (rectangles). Interval matrices can also be used to model unstructured matrix perturbations. This self contained article focuses on eigenvalue bounds of interval matrices and provides proofs to all theorems and lemmas.

The organization of this article is as follows. In Section 1 we study the eigenvalues of $(n \times n)$-dimensional real symmetric interval matrices and show that the exact real interval of variation of their first and last eigenvalues can be found by considering two sets of vertex matrices each of cardinality 2^{n-1}. In addition, we give a counterexample that falsify the conjecture that for $n \geq 3$ the interval(s) of variation of the other eigenvalues is attained at vertex matrices. We also remark that this result can be applied to real skew-symmetric interval matrices, as well as to finding the interval of variation of the real part of the eigenvalues of a class of interval matrices whose endpoints are real symmetric matrices. In Section 2 we study the eigenvalues of $(n \times n)$-dimensional Hermitian interval matrices and show that the exact real interval of variation of their first and last eigenvalues can be found by considering two sets of vertex matrices each of cardinality $2^{(n^2+n-2)/2}$. The above mentioned counterexample also falsifies the conjecture that for $n \geq 3$ the interval(s) of variation of the other eigenvalues is attained at vertex matrices. We also remark that this result can be applied to skew-Hermitian interval matrices, as well as to finding the interval of variation of of the real part of the eigenvalues of a class of interval matrices whose endpoints are Hermitian matrices with their imaginary part fixed. Finally, in Section 3 we present rectangular bounds for the eigenvalues of complex interval matrices.

In signal processing, control, and statistics real symmetric and Hermitian interval matrices represent, e. g., the quantized sampled covariance matrices of vector stochastic processes and their eigenvalues represent the variances of their decorrelated elements.

In a recent global optimization algorithm, see [1], real symmetric interval matrices were used to tightly bound the sets of Hessian matrices resulting from the objective function's nonconvex addends and then their minimal eigenvalues were used to tightly convexify them.

Eigenvalues of general interval matrices are useful to study robust stability margins of analog and discrete systems, and convergence rates in numerical analysis. The reader interested in additional work in this area is referred to [7] and the references therein, [5]; and, [4,6,10], where Toeplitz and Hankel interval matrices were studied. Genetic algorithms are promising for solving the above problems for large n.

Real Symmetric Interval Matrices

A *real* $(n \times n)$*-dimensional symmetric interval matrix* $\mathbf{S} = \mathbf{S}[\underline{S}, \overline{S}]$, where $\underline{S}$ and $\overline{S}$ are both real symmetric matrices is defined by

$$\mathbf{S} \equiv \left\{ S = [s_{k\ell}]: \quad \begin{array}{c} S = S^{\top}, \\ [\underline{s}_{k\ell} \leq s_{k\ell} \leq \overline{s}_{k\ell}], \\ k, \ell = 1, \ldots, n \end{array} \right\}, \tag{1}$$

where the superscript $\top$ denotes transposition. Further, let $\mathcal{S} \subset \mathbf{S}$ denote the set of all vertex matrices such that if $S = [s_{k\ell}] \in \mathcal{S}$, then $s_{k\ell} = \underline{s}_{k\ell}$ or $s_{k\ell} = \overline{s}_{k\ell}$. Note that $|\mathcal{S}| = 2^{(n^2+n)/2}$, where $|\mathcal{S}|$ denotes the cardinality of $\mathcal{S}$.

It is well known that all the eigenvalues of a real symmetric matrix S are real, see e. g. [11]. So let $\lambda_1(S) \leq \cdots \leq \lambda_n(S)$ be the ordered eigenvalues of S and $\lambda(S) \equiv \{\lambda_k(S): k = 1, \ldots, n\}$. Further, let $\lambda_k(\mathbf{S}) \equiv \{\lambda: \lambda = \lambda_k(S), S \in \mathbf{S}\}$, $\lambda(\mathbf{S}) \equiv \{\lambda: \lambda \in \lambda(S), S \in \mathbf{S}\}$,

$$\begin{cases} \underline{\lambda}_k(\mathbf{S}) \equiv \min(\lambda_k(\mathbf{S})), \\ \overline{\lambda}_k(\mathbf{S}) \equiv \max(\lambda_k(\mathbf{S})). \end{cases} \tag{2}$$

Because $\mathbf{S}$ is a compact set (i. e., closed and bounded in $\mathbf{R}^m$, where $m = (n^2 + n)/2$ is the number of free parameters of $\mathbf{S}$) and the eigenvalues of a matrix depend continuously upon its entries [9 Appendix D], it follows from [14 Thm. 4.16] that $\underline{\lambda}_k(\mathbf{S})$ and $\overline{\lambda}_k(\mathbf{S})$ are attained. That is, there exist matrices $\underline{S}_k, \overline{S}_k \in \mathbf{S}$ that satisfy $\underline{\lambda}_k(\mathbf{S}) = \lambda_k(\underline{S}_k), \overline{\lambda}_k(\mathbf{S}) = \lambda_k(\overline{S}_k)$, where $k = 1, \ldots, n$.

The purpose of this Section is to study the problem of computing the possibly overlapping eigenvalue intervals $\Lambda_k(\mathbf{S}) = [\underline{\lambda}_k(\mathbf{S}), \overline{\lambda}_k(\mathbf{S})]$, where $k = 1, \ldots, n$.

We have shown in [3] and [7] that the four endpoints of $\Lambda_1(\mathbf{S})$ and $\Lambda_n(\mathbf{S})$ are attained by considering two subsets of $\mathcal{S}$ each of size at most 2^{n-1}, see Theorem below.

Regarding $\Lambda_k(\mathbf{S})$ for $k = 2, \ldots, n-1$ and $n > 2$ we present below a (3×3)-dimensional real symmetric interval matrix $\mathbf{S}$ and a matrix $S_o \in \mathbf{S}$ for which $\overline{\lambda}_2(\mathbf{S}) \geq \overline{\lambda}_2(S_o) > \overline{\lambda}_2(S)$. That is, for $k \neq 1, n$ the endpoints of $\Lambda_k(\mathbf{S})$ are not necessarily attained at vertex matrices of $\mathcal{S}$.

It is well known that if $S \in \mathbf{S}$, $\lambda_1(S)$ and $\lambda_n(S)$ are attained at the minimal and maximal values, respectively, of the set $\{\mathbf{x}^{\mathsf{T}} S\mathbf{x}: \| \mathbf{x} \| = 1, \mathbf{x} \in \mathbf{R}^n\}$ [11 Thm. 5.2], where $\| \mathbf{x} \|$ denotes the Euclidean norm of the vector $\mathbf{x}$. Note

that because $\mathbf{x}^{\mathsf{T}}S\mathbf{x} = (-\mathbf{x})^{\mathsf{T}}S(-\mathbf{x})$ we can let $x_1 \geq 0$. Hence, the endpoints of $\Lambda_1(\mathbf{S})$ are given by

$$\underline{\lambda}_1(\mathbf{S}) = \min_{S\in\mathbf{S}} \min_{\substack{\mathbf{x}\in\mathbb{R}^n,\\ \|\mathbf{x}\|=1,\\ x_1\geq 0}} \mathbf{x}^{\top}S\mathbf{x} \tag{3}$$

and

$$\overline{\lambda}_1(\mathbf{S}) = \max_{S\in\mathbf{S}} \min_{\substack{\mathbf{x}\in\mathbb{R}^n,\\ \|\mathbf{x}\|=1,\\ x_1\geq 0}} \mathbf{x}^{\top}S\mathbf{x} \tag{4}$$

Similarly, one can give expressions for the endpoints of $\Lambda_n(\mathbf{S})$, or, alternatively use the relation $\Lambda_n(\mathbf{S}) = -\Lambda_1(-\mathbf{S})$.

Expanding $\mathbf{x}^{\mathsf{T}}S\mathbf{x}$, where $\mathbf{x} \in \mathbf{R}^n$ and $\mathrm{S} \in \mathbf{S}$ we obtain

$$\mathbf{x}^{\top}S\mathbf{x} = \sum_{k=1}^{n} s_{kk}x_k^2 + 2\sum_{k=1}^{n}\sum_{\ell=k+1}^{n} s_{k\ell}x_kx_\ell. \tag{5}$$

Since $x_1 \geq 0$, $\mathbf{x}$ may vary in 2^{n-1} closed orthants wherein the sign pattern of its elements is preserved. Let $\mathbf{O}_p \subset \mathbf{R}^n$, $p = 1, \ldots, 2^{n-1}$, denote the set of unit-length real vectors $\mathbf{x}$ with $x_1 \geq 0$ that belong to the pth orthant, where the orthants are ordered according to the binary order of the signs of the last $n-1$ elements of $\mathbf{x}$ and a negative (nonnegative) element corresponds to '0' ('1').

Hence we obtain the following subset of $\mathbb{S}$:

$$\underline{S} \equiv \left\{ S: \begin{array}{c} S = \arg\min_{S\in\mathbf{S}} \mathbf{x}^{\top}S\mathbf{x}, \\ \text{some} \quad \mathbf{x} \in \mathbf{O}_p^o, \\ p = 1, \ldots, 2^{n-1} \end{array} \right\}$$
$$= \left\{ \underline{S}^p : \; p = 1, \ldots, 2^{n-1} \right\}, \tag{6}$$

where $\underline{S}^p = [\underline{s}_{k\ell}^p]$, $\mathbf{O}_p^o$ denotes the interior of $\mathbf{O}_p$, and

$$\underline{s}_{k\ell}^p = \begin{cases} \underline{s}_{kk} & \text{if } k = \ell, \\ \underline{s}_{k\ell} & \text{if } (x_kx_\ell > 0) \wedge (k \neq \ell), \\ \overline{s}_{k\ell} & \text{if } (x_kx_\ell < 0) \wedge (k \neq \ell). \end{cases} \tag{7}$$

Similarly as above, by maximizing $\mathbf{x}^{\mathsf{T}}S\mathbf{x}$ over $S \in \mathbf{S}$ and some $\mathbf{x} \in \mathbf{O}_p^o$ one arrives at $\overline{S}^p$ and $\overline{S}$, where $\overline{S}^p \in \overline{S} \subset S$ and $|\overline{S}| = |\underline{S}|$.

Theorem 1

$$\underline{\lambda}_1(\mathbf{S}) = \lambda_1(\underline{S}), \quad \overline{\lambda}_1(\mathbf{S}) = \lambda_1(\overline{S}),$$

and

$$\underline{\lambda}_n(\mathbf{S}) = \lambda_n(\underline{S}), \quad \overline{\lambda}_n(\mathbf{S}) = \lambda_n(\overline{S}).$$

Proof We will prove that $\underline{\lambda}_1(\mathbf{S}) = \underline{\lambda}_1(\underline{S})$. The rest of the proof is similar and will therefore be omitted. Because the minimization in (3) is over a compact set (i. e., $\{\mathbf{x}, S\colon \mathbf{x} \in \mathbf{R}^n, \| \mathbf{x} \| = 1, S \in \mathbf{S}\}$) and $\mathbf{x}^{\mathsf{T}}S\mathbf{x}$ is a real continuous function of $\mathbf{x}$ and S, it follows that $\mathbf{x}^{\mathsf{T}}S\mathbf{x}$ attains its minimal value for some $\mathbf{x}^o \in \mathbf{O}^p$ and $S^o \in \mathbf{S}$. By expanding the quadratic form $\mathbf{x}^{o\mathsf{T}}S^o\mathbf{x}^o$ as in (5) it can be seen that $\mathbf{x}^{o\top}S^o\mathbf{x}^o \geq \mathbf{x}^{o\top}\underline{S}^p\mathbf{x}^o \geq \mathbf{x}^{p\top}\underline{S}^p\mathbf{x}^p$, where $\mathbf{x}^p$ denotes the unit length eigenvector of $\underline{S}^p \in \underline{S}$ corresponding to $\lambda_1(\underline{S}^p)$. Moreover, because $\mathbf{x}^o$ and S^o solve the optimization problem (3) it follows that $\mathbf{x}^{o\top}S^o\mathbf{x}^o \leq \mathbf{x}^{p\top}\underline{S}^p\mathbf{x}^p$. Hence $\mathbf{x}^{o\top}S^o\mathbf{x}^o = \mathbf{x}^{p\top}\underline{S}^p\mathbf{x}^p$ and therefore $\underline{\lambda}_1(\mathbf{S}) = \underline{\lambda}_1(\underline{S})$.

Note that similar results as in Theorem 1 hold for real skew-symmetric interval matrices, see [13].

Remark 2 Let $\mathbf{S}[\underline{S}, \overline{S}]$ be defined as before with $\underline{S} = \underline{S}^{\top}$ and $\overline{S} = \overline{S}^{\top}$. Define the real interval matrix $\mathbf{B}[\underline{S}, \overline{S}] \supset \mathbf{S}[\underline{S}, \overline{S}]$ by $\mathbf{B} \equiv \{B = [b_{k\ell}]\colon [\underline{s}_{k\ell} \leq b_{k\ell} \leq \overline{s}_{k\ell}],\ k, \ell = 1, \ldots, n\}$. Using Bendixon's theorem [11 Thm. 5.3] (i. e., for $B \in \mathbf{B}$, $\min \Re\lambda(B) \geq \underline{\lambda}(B')$ and $\max \Re\lambda(B) \leq \overline{\lambda}(B')$, where $B' \equiv \frac{(B+B^{\top})}{2}$, and $\Re, \Im$ denote the real and imaginary parts, respectively), it follows that $\min \Re\lambda(\mathbf{B}[\underline{S}, \overline{S}]) = \underline{\lambda}(\mathbf{S})$ and $\max \Re\lambda(\mathbf{B}[\underline{S}, \overline{S}]) = \overline{\lambda}(\mathbf{S})$. Hence, using Theorem 1 we obtain that $\min \Re\lambda(\mathbf{B}) = \underline{\lambda}(\underline{S})$ and $\max \Re\lambda(\mathbf{B}) = \overline{\lambda}(\overline{S})$, see also [12].

Example 3 Let the 3-dimensional real symmetric matrix $\mathbf{S} = [\underline{S}, \overline{S}]$ be given by

$$\underline{S} = \begin{pmatrix} 2 & 1 & -7 \\ 1 & 3 & -1 \\ -7 & -1 & -1 \end{pmatrix},$$

$$\overline{S} = \begin{pmatrix} 6 & 2 & -2 \\ 2 & 11 & 7 \\ -2 & 7 & 5 \end{pmatrix}.$$

Here $\mathbf{S}$ contains $2^6 = 64$ vertex matrices, i. e., $|S| = 64$. Using MatLab we obtain that $\overline{\lambda}(S) = \overline{\lambda}(S_v) = 10.1549$, where

$$S_v = \begin{pmatrix} 6 & 1 & -7 \\ 1 & 11 & -1 \\ -7 & -1 & 5 \end{pmatrix}.$$

Next, results of 10^5 computer runs produced the following matrix $S_o \in \mathbf{S}$,

$$S_o = \begin{pmatrix} 5.2054 & 1.6556 & -6.8321 \\ 1.6556 & 10.9244 & 1.9721 \\ -6.8321 & 1.9721 & 4.6703 \end{pmatrix}$$

with $\overline{\lambda}_2(S_o) = 11.3514$. Hence $\overline{\lambda}_2(\mathbf{S}) \geq \overline{\lambda}_2(S_o) > \overline{\lambda}_2(S)$.

Next, considering $-\mathbf{S} = \{S: -S \in \mathbf{S}\}$ we obtain that $S_1 \equiv -S_o \in -\mathbf{S}$ satisfies $\underline{\lambda}_2(S_1) = -\overline{\lambda}_2(-S_o)$ and consequently $\underline{\lambda}_2(-\mathbf{S}) \leq \underline{\lambda}_2(S_1) < \underline{\lambda}_2(-S)$, where $-\mathcal{S}$ has a similar meaning as $-\mathbf{S}$.

Hence, the endpoints of $\Lambda_2(\mathbf{S})$ need not be attained at vertex matrices of $\mathcal{S}$.

Hermitian Interval Matrices

An $(n \times n)$-dimensional Hermitian interval matrix $\mathbf{H} = \mathbf{H}[\underline{H}, \overline{H}]$, where $\underline{H}$ and $\overline{H}$ are both Hermitian matrices is defined by

$$\begin{aligned} \mathbf{H} \equiv \{H:\ & H = H^*, \\ & [\Re \underline{h}_{k\ell} \leq \Re h_{k\ell} \leq \Re \overline{h}_{k\ell}, \quad \Im \underline{h}_{k\ell} \leq \Im h_{k\ell} \leq \Im \overline{h}_{k\ell}], \\ & k, \ell = 1, \ldots, n,\ k \geq \ell\}, \end{aligned} \tag{8}$$

where $\Im\ h_{kk} = 0$, $k = 1, \ldots, n$, * denotes the Hermitian operator (i. e. conjugation followed by transposition). Further, let $\mathcal{H} \subset \mathbf{H}$ denote the set of all vertex matrices such that if $H = [h_{k\ell}] \in \mathcal{H}$, then $h_{k\ell} = \underline{h}_{k\ell}$ or $h_{k\ell} = \overline{h}_{k\ell}$. Note that the cardinality of $\mathcal{H}$ is $|\mathcal{H}| = 2^{n^2}$. It is well known that the eigenvalues of an Hermitian matrix H are real, see e. g. [11]. So let $\lambda_1(H) \leq \lambda_2(H) \leq \cdots \leq \lambda_n(H)$ be the ordered eigenvalues of H and $\lambda(H) = \{\lambda_k(H)$: k = 1, …, n$\}$. Further, let as before $\lambda_k(\mathbf{H}) \equiv \{\lambda:\ \lambda = \lambda_k(H),\ H \in \mathbf{H}\}$, $\lambda(\mathbf{H}) \equiv \{\lambda:\ \lambda \in \lambda(H),\ H \in \mathbf{H}\}$, $\underline{\lambda}_k(\mathbf{H}) \equiv \min(\lambda_k(\mathbf{H}))$, and $\overline{\lambda}_k(\mathbf{H}) \equiv \max(\lambda_k(\mathbf{H}))$.

Because $\mathbf{H}$ is a compact set (i. e., closed and bounded in $\mathbb{R}^{n^2}$, where n^2 is the number of free real parameters of $\mathbf{H}$) and the eigenvalues of a matrix depend continuously upon its entries [9, Appendix D], it follows from [14, Thm. 4.16] that $\underline{\lambda}_k(\mathbf{H})$ and $\overline{\lambda}_k(\mathbf{H})$ are attained. That is, there exist matrices $\underline{H}_k, \overline{H}_k \in \mathbf{H}$ that satisfy $\underline{\lambda}_k(\mathbf{H}) = \lambda_k(\underline{H}_k)$, $\overline{\lambda}_k(\mathbf{H}) = \lambda_k(\overline{H}_k)$, where $k = 1, \ldots, n$.

The purpose of this Section is to study the problem of computing the possibly overlapping eigenvalue intervals $\Lambda_k(\mathbf{H}) = [\underline{\lambda}_k(\mathbf{H}), \overline{\lambda}_k(\mathbf{H})]$, where $k = 1, \ldots, n$. We have shown in [2] and [7] that the four endpoints of $\Lambda_1(\mathbf{H})$ and $\Lambda_n(\mathbf{H})$ are attained by considering two subsets of $\mathcal{H}$ each of size at most $2^{(n^2+n-2)/2}$, see Theorem 4 below.

Regarding $\Lambda_k(\mathbf{H})$ for $k = 2, \ldots,$ n − 1 and $n \geq 3$, since real-symmetric interval matrices are a special case of Hermitian interval matrices, it follows from Example 3 that for $k \neq 1$, n the endpoints of $\Lambda_k(\mathbf{H})$ are not necessarily attained at vertex matrices of $\mathcal{H}$.

It is well known that if $H \in \mathbf{H}$, $\lambda_1(H)$ and $\lambda_n(H)$ are attained at the minimal and maximal values, respectively, of the set $\{x^*\ Hx$: $\|x\| = 1, x \in C^n\}$, see [11 Thm. 5.2]. Note that because $\mathbf{x}^* H \mathbf{x} = (e^{i\phi}\mathbf{x})^* H (e^{i\phi}\mathbf{x})$ for all ϕ we can choose $x_1 \in \mathbf{R}^+$, where $\mathbf{R}^+ = \{x \in \mathbf{R}$: $x \geq 0\}$. Hence, the endpoints of $\Lambda_1(\mathbf{H})$ are given by

$$\underline{\lambda}_1(\mathbf{H}) = \min_{H \in \mathbf{H}} \min_{\substack{\mathbf{x} \in \mathbb{C}^n, \\ \|\mathbf{x}\| = 1, \\ x_1 \in \mathbb{R}^+}} \mathbf{x}^* H \mathbf{x} \tag{9}$$

and

$$\overline{\lambda}_1(\mathbf{H}) = \max_{H \in \mathbf{H}} \min_{\substack{\mathbf{x} \in \mathbb{C}^n, \\ \|\mathbf{x}\| = 1, \\ x_1 \in \mathbb{R}^+}} \mathbf{x}^* H \mathbf{x}. \tag{10}$$

Similarly, one can give expressions for the endpoints of $\Lambda_n(\mathbf{H})$, or, alternatively use the relation $\Lambda_n(\mathbf{H}) = -\Lambda_1(-\mathbf{H})$. Since $x_1 \in \mathbf{R}^+$, $\mathbf{x} \in \mathbf{C}^n$ may vary in 2^{2n-2} closed orthants contained in $\mathbf{R}^{2n-1}$ wherein the sign pattern of its elements is preserved. Let $\mathbf{O}_p$, $p = 1, \ldots, 2^{2n-2}$, denote the set of unit-length real vectors $\mathbf{x}$ with $x_1 \in \mathbf{R}^+$ that belong to the pth orthant, where the orthants are ordered according to the binary order of the signs of the vector $(\Re x_2, \Im x_2, \ldots, \Re x_n, \Im x_n) \in \mathbb{R}^{2n-2}$ and a negative (nonnegative) element corresponds to '0' ('1').

Expanding $\mathbf{x}^* H \mathbf{x}$, where $\mathbf{x} = \mathbf{u} + i\mathbf{v}$, $\mathbf{u}, \mathbf{v} \in \mathbf{R}^n$ and $H = B + jC \in \mathbf{H}$, $B = B^{\mathsf{T}} \in \mathbf{R}^{n \times n}$, $C = -C^{\mathsf{T}} \in \mathbf{R}^{n \times n}$, and

noting that diag(C) = 0 we obtain

$$\begin{aligned}\mathbf{x}^* H\mathbf{x} &= b_{11}u_1^2 \\ &+ \sum_{k=2}^{n} b_{kk}(u_k^2 + v_k^2) \\ &+ 2\sum_{\ell=2}^{n}[b_{1\ell}u_1u_\ell + c_{1\ell}(-u_1v_\ell)] \\ &+ 2\sum_{k=2}^{n}\sum_{\ell=k+1}^{n}[b_{k\ell}(u_ku_\ell \\ &+ v_kv_\ell) + c_{k\ell}(-u_kv_\ell + v_ku_\ell)]. \end{aligned} \tag{11}$$

Hence, minimizing $\mathbf{x}^*H\mathbf{x}$ over $H = [h_{k\ell}] = [b_{k\ell} + ic_{k\ell}] \in \mathbf{H}$ and some $\mathbf{x} \in \mathbf{O}_p^o$ we obtain that only part of the entries $b_{k\ell}$ and $c_{k\ell}$ can be chosen at vertex points, i. e.,

$$\underline{b}_{k\ell}^p = \begin{cases} \underline{h}_{kk} & \text{if } k = \ell, \\ \Re\underline{h}_{1\ell} & \text{if } (u_\ell > 0) \wedge (\ell > k = 1), \\ \Re\overline{h}_{1\ell} & \text{if } (u_\ell < 0) \wedge (\ell > k = 1), \\ \Re\underline{h}_{k\ell} & \text{if } (u_ku_\ell > 0, v_kv_\ell > 0) \\ & \quad \wedge(\ell > k > 1), \\ \Re\overline{h}_{k\ell} & \text{if } (u_ku_\ell < 0, v_kv_\ell < 0) \\ & \quad \wedge(\ell > k > 1). \end{cases} \tag{12}$$

and

$$\underline{c}_{k\ell}^p = \begin{cases} 0 & \text{if } k = \ell, \\ \Im\underline{h}_{1\ell} & \text{if } (v_\ell < 0) \wedge (\ell > k = 1), \\ \Im\overline{h}_{1\ell} & \text{if } (v_\ell > 0) \wedge (\ell > k = 1), \\ \Im\underline{h}_{k\ell} & \text{if } (u_kv_\ell < 0, v_ku_\ell > 0) \\ & \quad \wedge(\ell > k > 1), \\ \Im\overline{h}_{k\ell} & \text{if } (u_kv_\ell > 0, v_ku_\ell < 0) \\ & \quad \wedge(\ell > k > 1). \end{cases} \tag{13}$$

Let $\underline{\mathcal{H}}^p \subset \mathcal{H}$ denote the set of all vertex matrices that satisfy (12) and (13). Using (12) and (13) one can set in B and C their diagonal, first column, first row, and if $\ell > k > 1$ either an entry of B or of C, see [2] for more details. Hence, since the number of free parameters in $\mathbf{H}$ is n^2, it follows that $\left|\underline{\mathcal{H}}^p\right| = 2^{\frac{(n^2-3n+2)}{2}}$. Further, let $\underline{\mathcal{H}}^p = \left\{\underline{H}^{p\ell}\colon\ \ell = 1, \ldots, 2^{\frac{(n^2-3n+2)}{2}}\right\}$ and $\underline{\mathcal{H}} = \left\{\underline{\mathcal{H}}^p\colon\ p = 1, \ldots, 2^{2n-2}\right\}$ hence $|\underline{\mathcal{H}}| = 2^{\frac{(n^2+n-2)}{2}}$. Similarly as above, by maximizing $\mathbf{x}^*H\mathbf{x}$ over $H \in \mathbf{H}$ and some $\mathbf{x} \in \mathbf{O}_p^o$ we arrive at $\overline{\mathcal{H}}^p$ and $\overline{\mathcal{H}}$, where $\overline{\mathcal{H}}^p \subset \overline{\mathcal{H}} \subset \mathcal{H}$, $\left|\overline{\mathcal{H}}^p\right| = \left|\underline{\mathcal{H}}^p\right|$, and $\left|\overline{\mathcal{H}}\right| = |\underline{\mathcal{H}}|$.

Theorem 4

$$\underline{\lambda}_1(\mathbf{H}) = \lambda_1(\underline{\mathcal{H}}), \quad \overline{\lambda}_1(\mathbf{H}) = \lambda_1(\overline{\mathcal{H}})$$

and

$$\underline{\lambda}_n(\mathbf{H}) = \lambda_n(\underline{\mathcal{H}}), \quad \overline{\lambda}_n(\mathbf{H}) = \lambda_n(\overline{\mathcal{H}}).$$

Proof We will prove that $\underline{\lambda}_1(\mathbf{H}) = \underline{\lambda}_1(\underline{\mathcal{H}})$. The rest of the proof is similar and will therefore be omitted. Because the minimization in (3) is over a compact set (i. e., $\{\mathbf{x}, \mathrm{H}\colon \mathbf{x} \in \mathbf{C}^n, \|\mathbf{x}\| = 1, H \in \mathbf{H}\}$) and $\mathbf{x}^*H\mathbf{x}$ is a real continuous function of $\mathbf{x}$ and H, it follows that $\mathbf{x}^*H\mathbf{x}$ attains its minimal value for some $\mathbf{x}^o \in \mathbf{O}^p$ and $H^o \in \mathbf{H}$. By expanding $\mathbf{x}^{o\,*}H\mathbf{x}^o$ as in (11) and noting that $\mathbf{x}^o$ is constant, it can be seen that there is an $\underline{H}^{p\ell} \in \underline{\mathcal{H}}^p$ for which $\mathbf{x}^{o*}H^o\mathbf{x}^o \geq \mathbf{x}^{o*}\underline{H}^{p\ell}\mathbf{x}^o \geq \mathbf{x}^{p\ell*}\underline{H}^{p\ell}\mathbf{x}^{p\ell}$, where $\mathbf{x}^{p\ell}$ denotes the unit-length eigenvector of $\underline{H}^{p\ell}$ corresponding to $\lambda_1(\underline{H}^{p\ell})$. Moreover, because $\mathbf{x}^o$ and H^o solve the optimization problem (9), it follows that $\mathbf{x}^{o*}H^o\mathbf{x}^o \leq \mathbf{x}^{p\ell*}\underline{H}^{p\ell}\mathbf{x}^{p\ell}$. Hence $\mathbf{x}^{o*}H^o\mathbf{x}^o = \mathbf{x}^{p\ell*}\underline{H}^{p\ell}\mathbf{x}^{p\ell}$ and therefore $\underline{\lambda}_1(\mathbf{H}) = \underline{\lambda}_1(\underline{\mathcal{H}})$.

Note that similar results as in Theorem 4 hold for skew-Hermitian interval matrices.

Remark 5 This remark is similar to Remark 2. Let $\mathbf{H}[\underline{H}, \overline{H}]$ be defined as before with $\underline{H} = \underline{H}^*$, $\overline{H} = \overline{H}^*$, and $\Im\underline{H} = \Im\overline{H}$. Define the complex interval matrix $\mathbf{A}[\underline{H}, \overline{H}] \supset \mathbf{H}[\underline{H}, \overline{H}]$ by

$$\mathbf{A} \equiv \left\{ A = [a_{k\ell}]\colon \begin{array}{c} [\Re\underline{h}_{k\ell} \leq \Re a_{k\ell} \leq \Re\overline{h}_{k\ell}, \\ \Im a_{k\ell} = \Im\underline{h}_{k\ell}], \\ k, \ell = 1, \ldots, n \end{array} \right\}.$$

Using Bendixon's theorem and Theorem 4, it follows that $\min \Re\lambda(\mathbf{A}[\underline{H}, \overline{H}]) = \underline{\lambda}(\mathbf{H}) = \underline{\lambda}(\underline{\mathcal{H}})$ and $\max \Re\lambda(\mathbf{A}[\underline{H}, \overline{H}]) = \overline{\lambda}(\mathbf{H}) = \overline{\lambda}(\overline{\mathcal{H}})$.

Complex Interval Matrices

In this Section we present rectangular bounds on the eigenvalues of a complex interval matrix by extending similar results for a real interval matrix [13]. A complex interval matrix, denoted by $\mathbf{A}$, is defined by $\mathbf{A} \equiv \mathbf{B} + i\mathbf{C}$, where $\mathbf{B} \equiv \{B\colon \underline{B} \leq B \leq \overline{B},\ B \in \mathbb{R}^{n\times n}\}$, $\mathbf{C} \equiv \{C\colon \underline{C} \leq C \leq \overline{C},\ C \in \mathbb{R}^{n\times n}\}$, $i^2 = -1$, and

$\underline{B}, \overline{B}, \underline{C}, \overline{C} \in \mathbb{R}^{n\times n}$ are fixed matrices. Further, let $\underline{A} = \underline{B} + i\underline{C}$, $\overline{A} = \overline{B} + i\overline{C}$, and $A_c = \frac{(\underline{A}+\overline{A})}{2}$.

Let $\lambda = \lambda_r + i\lambda_i$ ($\lambda_r, \lambda_i \in \mathbf{R}^1$) and $\mathbf{x} = \mathbf{u} + i\mathbf{v}$ ($\mathbf{u}, \mathbf{v} \in \mathbf{R}^{n\times 1}$, $\| \mathbf{x} \| = 1$) be an eigenvalue and the corresponding unit length eigenvector of some matrix $A \in \mathbf{A}$. First, note that $\| \mathbf{x} \| = 1$ implies

$$\|\mathbf{x}\|^2 = \mathbf{x}^*\mathbf{x} = \mathbf{u}^\top\mathbf{u} + \mathbf{v}^\top\mathbf{v} = 1. \tag{14}$$

Let $A = B + iC$, $B \in \mathbf{B}$, $C \in \mathbf{C}$. Since $A\mathbf{x} = \lambda\mathbf{x}$ we obtain

$$\begin{aligned}(\lambda_r + i\lambda_i)\mathbf{x} &= (B + iC)(\mathbf{u} + i\mathbf{v})\\ &= B\mathbf{u} - C\mathbf{v} + i(B\mathbf{v} + C\mathbf{u}).\end{aligned}$$

Premultiplying the above equation by $\mathbf{x}^* = \mathbf{u}^\top - i\mathbf{v}^\top$, equating the real and imaginary parts, and noting that $\mathbf{x}^*\mathbf{x} = 1$ we obtain

$$\lambda_r = \mathbf{u}^\top B\mathbf{u} - \mathbf{u}^\top C\mathbf{v} + \mathbf{v}^\top B\mathbf{v} + \mathbf{v}^\top C\mathbf{u}$$

and

$$\lambda_i = -\mathbf{v}^\top B\mathbf{u} + \mathbf{v}^\top C\mathbf{v} + \mathbf{u}^\top B\mathbf{v} + \mathbf{u}^\top C\mathbf{u}.$$

We have that $\mathbf{u}^\top B\mathbf{u} = \mathbf{u}^\top B'\mathbf{u} \le \overline{\lambda}(B')\mathbf{u}^\top\mathbf{u}$ and $\mathbf{v}^\top B\mathbf{v} = \mathbf{v}^\top B'\mathbf{v} \le \overline{\lambda}(B')\mathbf{v}^\top\mathbf{v}$, see [11], where

$$B' \equiv \frac{B + B^\top}{2}. \tag{15}$$

Note that similar results pertain to the real matrix C.

Hence, using (14) we obtain

$$\lambda_r \le \overline{\lambda}(B') - \mathbf{u}^\top C\mathbf{v} + \mathbf{v}^\top C\mathbf{u}. \tag{16}$$

Choose $B_c = \frac{(\underline{B}+\overline{B})}{2}$ and $C_c = \frac{(\underline{C}+\overline{C})}{2}$, then

$$\begin{aligned}\overline{\lambda}(B') &= \max_{\|\mathbf{x}\|=1, \mathbf{x}\in\mathbb{R}^n} \mathbf{x}^\top B\mathbf{x}\\ &= \max_{\|\mathbf{x}\|=1, \mathbf{x}\in\mathbb{R}^n} \left(\mathbf{x}^\top B_c\mathbf{x} + \mathbf{x}^\top(B - B_c)\mathbf{x}\right)\\ &\le \overline{\lambda}(B'_c) + \max_{\|\mathbf{x}\|=1, \mathbf{x}\in\mathbb{R}^n} |\mathbf{x}|^\top \Delta_{\mathbf{B}} |\mathbf{x}|\\ &= \overline{\lambda}(B'_c) + \overline{\lambda}(\Delta'_{\mathbf{B}}),\end{aligned} \tag{17}$$

where $|\mathbf{x}| \equiv \mathrm{abs}(\mathbf{x})$ taken elementwise,

$$\Delta_{\mathbf{B}} \equiv \overline{B} - B_c, \tag{18}$$

and both Δ'_B and B'_c have similar meaning as B' defined in (15).

To obtain the final form of the upper bound on λ_r, $\overline{\lambda}_r$, it remains to carry out the following derivation:

$$\begin{aligned}&-\mathbf{u}^\top C\mathbf{v} + \mathbf{v}^\top C\mathbf{u}\\ &= -\mathbf{u}^\top C_c\mathbf{v} + \mathbf{v}^\top C_c\mathbf{u} - \mathbf{u}^\top(C - C_c)\mathbf{v} + \mathbf{v}^\top(C - C_c)\mathbf{u}\\ &\le \max_{\|(\mathbf{u}^\top,\mathbf{v}^\top)\|=1} (-\mathbf{u}^\top C_c\mathbf{v} + \mathbf{v}^\top C_c\mathbf{u})\\ &\quad + \max_{\|(\mathbf{u}^\top,\mathbf{v}^\top)\|=1} (|\mathbf{u}|^\top \Delta_{\mathbf{C}} |\mathbf{v}| + |\mathbf{v}|^\top \Delta_{\mathbf{C}} |\mathbf{u}|)\\ &\le \max_{\|(\mathbf{u}^\top,\mathbf{v}^\top)\|=1} \begin{pmatrix}\mathbf{u}\\ \mathbf{v}\end{pmatrix}^\top \begin{pmatrix}0 & -C_c\\ C_c & 0\end{pmatrix}\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\end{pmatrix}\\ &\quad + \max_{\|(\mathbf{u}^\top,\mathbf{v}^\top)\|=1} \begin{pmatrix}|\mathbf{u}|\\ |\mathbf{v}|\end{pmatrix}^\top \begin{pmatrix}0 & \Delta_{\mathbf{C}}\\ \Delta_{\mathbf{C}} & 0\end{pmatrix}\begin{pmatrix}|\mathbf{u}|\\ |\mathbf{v}|\end{pmatrix}\\ &\le \overline{\lambda}(C''_c) + \overline{\lambda}(\Delta''_{\mathbf{C}}),\end{aligned} \tag{19}$$

where $\Delta_{\mathbf{C}}$ has similar meaning as $\Delta_{\mathbf{B}}$ defined in (18),

$$\begin{aligned}C''_c &\equiv \begin{pmatrix}0 & \frac{-C_c + C_c^\top}{2}\\ \frac{C_c - C_c^\top}{2} & 0\end{pmatrix},\\ \Delta''_{\mathbf{C}} &\equiv \begin{pmatrix}0 & \Delta'_{\mathbf{C}}\\ \Delta'_{\mathbf{C}} & 0\end{pmatrix},\end{aligned} \tag{20}$$

and Δ'_C has similar meaning as B' defined in (15). Hence, using (16), (17), and (19) we finally obtain

$$\lambda_r \le \overline{\lambda}_r = \overline{\lambda}(B'_c) + \overline{\lambda}(\Delta'_{\mathbf{B}}) + \overline{\lambda}(C''_c) + \overline{\lambda}(\Delta''_{\mathbf{C}}). \tag{21}$$

The lower bound on λ_r, $\underline{\lambda}_r$, can be obtained by noting that $-\lambda = -\lambda_r - i\lambda_i$ is an eigenvalue of $-A \in -\mathbf{A} = -\mathbf{B} + i(-\mathbf{C})$, where $-\mathbf{B} = \{B\colon -\overline{B} \le B \le -\underline{B},\ B \in \mathbb{R}^{n\times n}\}$ and, $-\mathbf{C} = \{C\colon -\overline{C} \le C \le -\underline{C}, C \in \mathbb{R}^{n\times n}\}$ using (21) and then replacing the roles of $\mathbf{B}$ and $\mathbf{C}$ by $-\mathbf{B}$ and $-\mathbf{C}$ we obtain

$$\lambda_r \ge \underline{\lambda}_r = \underline{\lambda}(B'_c) - \overline{\lambda}(\Delta'_{\mathbf{B}}) + \underline{\lambda}(C''_c) - \overline{\lambda}(\Delta''_{\mathbf{C}}). \tag{22}$$

The upper and lower bounds on λ_i can be similarly obtained by noting that $-i\lambda = \lambda_i - i\lambda_r$ is an eigenvalue of $-iA \in -i\mathbf{A} = \mathbf{C} + i(-\mathbf{B})$, using (21) and (22), respectively, and then replacing the roles of $\mathbf{B}$ and $\mathbf{C}$ by $\mathbf{C}$ and $-\mathbf{B}$. We thus obtain

Theorem 6 *Let $\mathbf{A} = \mathbf{B} + i\mathbf{C}$ be as defined above, $A_c = \frac{(\underline{A}+\overline{A})}{2} = B_c + iC_c$ be the central matrix of $\mathbf{A}$, and $\lambda = \lambda_r + i\lambda_i$ be any eigenvalue of the matrix $A \in \mathbf{A}$, then*

$$\underline{\lambda}_r \le \lambda_r \le \overline{\lambda}_r \quad \textit{and} \quad \underline{\lambda}_i \le \lambda_i \le \overline{\lambda}_i,$$

where

$$\underline{\lambda}_r = \underline{\lambda}(B_c') - \overline{\lambda}(\Delta_{\mathbf{B}}') + \underline{\lambda}(C_c'') - \overline{\lambda}(\Delta_{\mathbf{C}}''),$$
$$\overline{\lambda}_r = \overline{\lambda}(B_c') + \overline{\lambda}(\Delta_{\mathbf{B}}') + \overline{\lambda}(C_c'') + \overline{\lambda}(\Delta_{\mathbf{C}}''),$$
$$\underline{\lambda}_i = \underline{\lambda}(C_c') - \overline{\lambda}(\Delta_{\mathbf{C}}') - \overline{\lambda}(B_c'') - \overline{\lambda}(\Delta_{\mathbf{B}}''),$$
$$\overline{\lambda}_i = \overline{\lambda}(C_c') + \overline{\lambda}(\Delta_{\mathbf{C}}') - \underline{\lambda}(B_c'') + \overline{\lambda}(\Delta_{\mathbf{B}}'');$$

all the primed matrices (i. e., B_c', C_c', Δ_B', and Δ_C') have similar meaning as B' defined in (15); $\Delta_{\mathbf{B}}$ is as in (18) and $\Delta_{\mathbf{C}}$ has similar meaning; and, C_c' and $\Delta_{\mathbf{C}}''$ are as in (20) with B_c'' and $\Delta_{\mathbf{B}}''$ having similar meaning, respectively.

Corollary 7 *Note the following consequences:*

i) if $\overline{\lambda}_r < 0$, then the interval matrix $\boldsymbol{A}$ is Hurwitz stable.

ii) if the rectangle

$$\left\{(x, y):\ \underline{\lambda}_r \le x \le \overline{\lambda}_r,\ \underline{\lambda}_i \le y \le \overline{\lambda}_i\right\}$$

is contained in the open unit disk, then $\boldsymbol{A}$ is Schur stable.

Some computational simplifications for Theorem 6 can be obtained by using the following lemmas.

Lemma 8 *Let C_c'' be as defined in (20), then $\overline{\lambda}(C_c'') = -\underline{\lambda}(C_c'') = \rho(\frac{(C_c - C_c^{\mathsf{T}})}{2})$, where $\rho(A) = \{|\lambda|: \lambda \in \lambda(A)\}$ denotes the spectral radius of the matrix A.*

Proof Let $G = (C_c - C_c^{\mathsf{T}})/2$ and $Gv = \lambda v$ (note that since G is skew symmetric, λ is purely imaginary, see [11]); the eigenvalues of C_c' are $\pm i\lambda$ with corresponding eigenvectors $(v^{\mathsf{T}}, \mp iv^{\mathsf{T}})^{\mathsf{T}}$, which gives the desired result.

Note that this Lemma can also be applied to B_c''.

Lemma 9 *Let*

$$D \equiv \begin{pmatrix} 0 & S \\ S & 0 \end{pmatrix},$$

where $S \in \boldsymbol{R}^{n\times n}$ and $S = S^{\mathsf{T}}$, then $\overline{\lambda}(D) = -\underline{\lambda}(D) = \rho(S)$.

Proof 4 The eigenvectors of D are $(w^{\mathsf{T}}, \pm w^{\mathsf{T}})^{\mathsf{T}}$, where w is an eigenvector of S. Hence the eigenvalues of D are $\pm\lambda$, with λ an eigenvalue of S, which gives the desired result.

Note that this Lemma can also be applied to $\Delta_{\mathbf{B}}''$ and $\Delta_{\mathbf{C}}''$.

See also

- ▶ αBB algorithm
- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Point and Interval Taylor Operators
- ▶ Bounding Derivative Ranges
- ▶ Eigenvalue Enclosures for Ordinary Differential Equations
- ▶ Global Optimization: Application to Phase Equilibrium Problems
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Interval Analysis: Differential Equations
- ▶ Interval Analysis: Intermediate Terms
- ▶ Interval Analysis: Nondifferentiable Problems
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ▶ Interval Analysis: Systems of Nonlinear Equations
- ▶ Interval Analysis: Unconstrained and Constrained Optimization
- ▶ Interval Analysis: Verifying Feasibility
- ▶ Interval Constraints
- ▶ Interval Fixed Point Theory
- ▶ Interval Global Optimization
- ▶ Interval Linear Systems
- ▶ Interval Newton Methods
- ▶ Semidefinite Programming and Determinant Maximization

References

1. Adjiman CS, Floudas CA (1996) Rigorous convex underestimators for general twice-differentiable problems. J Global Optim 9:23–40
2. Hertz D (1992) The extreme eigenvalues and stability of Hermitian interval matrices. IEEE Trans Circuits and Systems I 39(6):463–466
3. Hertz D (1992) The extreme eigenvalues and stability of real symmetric interval matrices. IEEE Trans Autom Control 37(4):532–535
4. Hertz D (1992) Simple bounds on the extreme eigenvalues of Toeplitz matrices. IEEE Trans Inform Theory 38(1):175–176
5. Hertz D (1993) The maximal eigenvalue and stability of a class of real symmetric interval matrices. IEEE Trans Cir-

cuits and Systems I: Fundamental Theory and Applications 40(1):56–57
6. Hertz D (1993) On the extreme eigenvalues of Toeplitz and Hankel interval matrices. Multidimensional Signals and Systems 4:83–90
7. Hertz D (1993) Root clustering of interval matrices. In: Jamshidi M, Mansour M, Anderson BDO (eds) Fundamentals of Discrete-Time Systems: A Tribute to Professor Eliahu I. Jury. IITSI Press, Albuquerque, NM, pp 271–278
8. Horn RA, Johnson CR (1985) Matrix analysis. Cambridge Univ. Press, Cambridge
9. Horn RA, Johnson CR (1991) Topics in matrix analysis. Cambridge Univ. Press, Cambridge
10. Jorge P, Ferreira SG (1994) Localization of the eigenvalues of Toeplitz matrices using additive decomposition, embedding in circulants, and the Fourier transform. SYSID, Copenhagen, Denmark 3:271–275; 175–176
11. Marcus M, Minc H (1966) Introduction to linear algebra. MacMillan, New York
12. Rohn J (1992) Positive definiteness and stability of interval matrices. Report NI-92-05, Numerik Inst Denmarks Tekniske Hojskole, Lyngby 2800, March
13. Rohn J (1996) Bounds on eigenvalues of interval matrices. Techn Report Inst Computer Sci Acad Sci Prague, Czech Republic 688, October
14. Rudin W (1978) Principles of mathematical analysis. McGraw-Hill, New York

Interval Analysis: Intermediate Terms

R. Baker Kearfott
Department Math., University Louisiana at Lafayette, Lafayette, USA

MSC2000: 65G20, 65G30, 65G40, 65H20

Article Outline

Keywords
Use In Automatic Differentiation
Use In Constraint Satisfaction Techniques.
Use In Symbolic Preprocessing.
See also
References

Keywords

Expression parsing; Constraint satisfaction techniques; Interval dependency; Verification; Interval computations; Global optimization

Interval Analysis: Intermediate Terms, Table 1

	$v_1 = x_1,$
	$v_2 = x_2,$
(i)	$v_3 = v_1^2,$
(ii)	$v_4 = v_2^2,$
(iii)	$v_5 = v_4 v_3,$
(iv)	$v_6 = v_3 - v_5,$
(v)	$v_7 = v_6 + v_4.$

Interval Analysis: Intermediate Terms, Table 2

OP	p	q	r
5	3	1	—
5	4	2	—
4	5	4	3
21	6	3	5
20	7	6	4

In global optimization algorithms, the computer must repeatedly evaluate an objective function, as well as, possibly, inequality and equality constraints. Such functions are given as algebraic expressions or as subroutines or sections of computer code. When such computer code is executed, operations are applied to the *independent variables*, producing *intermediate terms*. These intermediate terms are, in turn, combined to produce other intermediate terms, or, eventually, the objective function value. For example, consider the problem

$$\begin{cases} \min & \phi(x) = x_1^2 - x_1^2 x_2^2 + x_2^2 \\ \text{over the box} & \mathbf{x} = ([-1,1],[-1,1])^\top. \end{cases} \tag{1}$$

To evaluate ϕ, the computer may start with the independent variable values $v_1 = x_1$ and $v_2 = x_2$ internally produce quantities v_3, v_4, v_5, and v_6, to finally produce the dependent variable value $\phi(x) = v_7$. Table 1 indicates how this may be done.

A list such as in Table 1 may be represented as a table of addresses of variables and operations. For example, if the operation $x_p \leftarrow x_q^2$ corresponds to operation code 5, $x_p \leftarrow x_q x_r$ corresponds to operation code 4, $x_p \leftarrow x_q + x_r$ corresponds to operation code 20, and $x_p \leftarrow x_q - x_r$ corresponds to operation code 21, then the set of relations in Table 1 is represented by Table 2.

Such a sequence of operations is called a *code list*, but is sometimes called other things, such as a *tape*. As-

Interval Analysis: Intermediate Terms, Table 3

	$v_1 = x_1$
	$v_2 = x_2$
(i)	$v_3 = v_1^2$,
(ii)	$v_4 = v_2^2$,
(iii)	$v_5 = v_4 v_3$,
(iv)	$v_6 = v_3 - v_5$,
(v)	$v_7 = v_6 + v_4$,
(vi)	$v_8 = 2v_1$,
(vii)	$v_9 = v_8 v_4$,
(viii)	$v_{10} = v_8 - v_9$,
(ix)	$v_{11} = 2v_2$,
(x)	$v_{12} = v_3 v_{11}$,
(xi)	$v_{13} = -v_{12}$,
(xii)	$v_{14} = v_{13} + v_{11}$,
	$\phi = v_7$,
	$\frac{\partial \phi}{\partial x_1} = v_{10}$,
	$\frac{\partial \phi}{\partial x_2} = v_{14}$.

suming the axioms of real arithmetic hold for evaluation, code lists for a given algebraic expression or portion of a computer program are not unique.

The concept of a code list is familiar to computer science students who have worked with compilers, since a compiler produces such lists while translating algebraic expressions into machine language. However, code lists and access to the intermediate expressions are of particular importance in interval global optimization, for the following reasons.

- Code lists provide a convenient internal representation for the objective and constraints, to be used for *automatic differentiation*, for both point and interval evaluation of objectives, gradients, and *Hessian matrices*.
- The values of the intermediate quantities can be used within the optimization algorithm in processes that reduce the size of the search region.
- Symbolic manipulation can reduce the overestimation, or *interval dependency* that would otherwise occur with interval evaluations.

Details are given below.

Use In Automatic Differentiation

A code list can be used either as a pattern to specify the computations in the *forward mode* of automatic differentiation or as a symbolic representation of the system of equations to be solved in the *backward mode*. See [7] for an in-depth look at the forward mode of automatic differentiation, and see [3] for somewhat more recent research on the subject. See [6, pp. 37–39] for some examples and additional references. Also see ► Automatic differentiation: Introduction, history and rounding error estimation.

Use In Constraint Satisfaction Techniques.

Since each intermediate variable in the code list is connected to one or two others via an elementary, invertible operation, narrow bounds on one such intermediate variable can be used to obtain narrow bounds on others. For example, suppose that the code list in Table 1 has been symbolically differentiated, to get the code list in Table 3. Then, if the subbox $\mathbf{x} = ([0.5, 1], [-1, -0.5])^\mathsf{T}$ is to be considered for possible inclusion of optima, the derivative code list in Table 3 can be evaluated by *forward substitution* to obtain the interval set of intermediate values in Table 4. Furthermore, since (1) is an unconstrained problem, an optimum must occur where $\partial\phi/\partial x_1 = 0$ and $\partial\phi/\partial x_2 = 0$. In particular, any global optimizer x^* must have

$$v_{10}(x^*) = 0. \tag{2}$$

Using (2) in line (viii) of the derivative code list in Table 5,

$$v_9 = v_8 - v_{10},$$

whence

$$\tilde{\mathbf{v}}_9 \leftarrow [1, 2] - 0,$$
$$\mathbf{v}_9 \leftarrow \tilde{\mathbf{v}}_9 \cap \mathbf{v}_9 = [1, 2]\ .$$

Now, using (vii) of Table 5,

$$\tilde{\mathbf{v}}_4 \leftarrow \frac{\mathbf{v}_9}{\mathbf{v}_8} = \frac{[1, 2]}{[1, 2]} = [0.5, 2],$$
$$\mathbf{v}_4 \leftarrow \tilde{\mathbf{v}}_4 \cap \mathbf{v}_4 = [0.5, 1].$$

Now using (ii) of Table 5 gives

$$\tilde{\mathbf{v}}_2 \leftarrow \sqrt{\mathbf{v}_4} \cup -\sqrt{\mathbf{v}_4} \tag{3}$$

$$\subseteq [0.70, 1] \cup [-1, -0.70],$$
$$\mathbf{v}_2 \leftarrow \tilde{\mathbf{v}}_2 \cap \mathbf{v}_2 = [-1, -0.70]. \tag{4}$$

The last computation represents a narrowing of the range of one of the independent variables.

Interval Analysis: Intermediate Terms, Table 4

$\mathbf{v}_1 = [.5, 1]$,
$\mathbf{v}_2 = [-1, -.5]$,
$\mathbf{v}_3 = [.25, 1]$,
$\mathbf{v}_4 = [.25, 1]$,
$\mathbf{v}_5 = [.0625, 1]$,
$\mathbf{v}_6 = [-.75.9375]$,
$\mathbf{v}_7 = [-.5, 1.9375]$,
$\mathbf{v}_8 = [1, 2]$,
$\mathbf{v}_9 = [.25, 2]$,
$\mathbf{v}_{10} = [-1, 1.75]$,
$\mathbf{v}_{11} = [-2, -1]$,
$\mathbf{v}_{12} = [-2, -.25]$,
$\mathbf{v}_{13} = [.25, 2]$,
$\mathbf{v}_{14} = [-1.75, 1]$,
$\phi \in [-.5, 1.9375]$,
$\frac{\partial \phi}{\partial x_1} \in [-1, 1.75]$,
$\frac{\partial \phi}{\partial x_2} \in [-1.75, 1]$.

Interval Analysis: Intermediate Terms, Table 5

	$v_1 = x_1$,
	$v_2 = x_2$,
(i)	$v_3 = v_1^2$,
(ii)	$v_4 = v_2^2$,
(iii)	$v_5 = v_3 v_2$,
(iv)	$v_6 = v_1^3$,
(v)	$v_7 = v_6 + v_4$,
(vi)	$v_8 = v_7 + 1$,
(vii)	$v_8 + v_5 = 0$,
(viii)	$v_8 - 3v_5 = 0$

(A similar computation could also have been carried out to obtain narrower bounds on v_1.)

If, in addition, an upper bound $\overline{\phi} = 0$ for the global optimum of ϕ is known, then

$$v_7 \in [-\infty, 0] \cap [-0.5, 1.9375] = [-0.5, 0].$$

This can now be used in Table 3, (v), along with new intermediate variable bounds, wherever possible, to obtain

$$\widetilde{\mathbf{v}}_4 \leftarrow \mathbf{v}_7 - \mathbf{v}_6 = [-0.5, 0] - [-0.75, 0.9375] = [-1.4375, 0.75],$$
$$\mathbf{v}_4 \leftarrow \mathbf{v}_4 \cap \widetilde{\mathbf{v}}_4 = [0.5, 0.75].$$

Now using Table 3, (vii),

$$\widetilde{\mathbf{v}}_9 \leftarrow [1, 2][0.5, 0.75] = [0.5, 1.5],$$
$$\mathbf{v}_9 \leftarrow \widetilde{\mathbf{v}}_9 \cap \mathbf{v}_9 = [1, 1.5],$$

then using (viii) and $v_{10} = 0$ gives $\mathbf{v}_8 = [1, 1.5]$. Finally, using Table 3, (vi), gives

$$\mathbf{v}_1 \leftarrow [0.5, 0.75] \cap [0.5, 1] = [0.5, 0.75]. \tag{5}$$

Now, evaluating ϕ in (1) (or redoing the forward substitution represented in Table 4) at $(\mathbf{x}_1, \mathbf{x}_2) = ([0.5, 0.75], [-1, -0.70])$ gives

$$\begin{aligned}\phi &\in [.5.75]^2 - [.5.75]^2[-1, -.7]^2 + [-1, -.7]^2 \\ &= [.25.5625] - [.25.5625][.49, 1] + [.49, 1] \\ &= [.25.5625] + [-.5625, -.1225] + [.49, 1] \\ &= [.1775, 1.44],\end{aligned}$$

contradicting the known upper bound $\overline{\phi} = 0$. This proves that there can be no global optimizer of (1) within $([0.5, 1], [-1, -0.5])^\mathsf{T}$. (Note that, in fact, there are no global optimizers in $([-1, 1], [-1, 1])^\mathsf{T}$ if the problem is considered to be unconstrained.)

The above procedure is easily automated, as is done in, say, *GlobSol* [2,6], *UniCalc* [1], or other interval constraint propagation software.

This example illustrates a more general technique, associated with *constraint propagation* and *logic programming*. See [4] for an introduction to this view of the subject, and see [5] for alternate techniques of interval constraint satisfaction.

Use In Symbolic Preprocessing.

To understand how symbolic analysis based on the code list may help, consider the following example:

$$\begin{cases} \text{Find} & \text{all solutions to} \\ & f(x) = 0,\ f = (f_1, f_2)^\top \\ \text{within} & \text{the box } \mathbf{x} = ([-2, 0], [-1, 1])^\top \\ \text{where} & f_1(x_1, x_2) = x_1^3 + x_1^2 x_2 + x_2^2 + 1 \\ & f_2(x_1, x_2) = x_1^3 - 3x_1^2 x_2 + x_2^2 + 1. \end{cases} \tag{6}$$

A possible code list is
There is much interval dependency in this system, both in the individual equations (since each variable occurs in various terms), and between the equations (since the

equations share common terms). However, examination of the code list in Table 5 reveals that a change of variables can make the system more amenable to interval computation. Seeing that (vii) and (viii) are linear in v_5 and $v_8 = v_4 + v_6 + 1$, define

$$\begin{cases} y_1 = v_5 = x_1^2 x_2, \\ y_2 = v_4 + v_6 = x_1^3 + x_2^2. \end{cases} \tag{7}$$

Then the system becomes

$$\begin{cases} y_2 + y_1 + 1 = 0, \\ y_2 - 3y_1 + 1 = 0. \end{cases} \tag{8}$$

Thus, the linear system (8) may be solved easily for y_1 and y_2. The interval bounds may then be plugged into (7) to obtain x_1 and x_2. There is no overestimation in any of the expressions for function components or partial derivatives in either (8) or (7).

Additional research should reveal how to automate this change of variables process.

See also

- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Point and Interval Taylor Operators
- ▶ Bounding Derivative Ranges
- ▶ Global Optimization: Application to Phase Equilibrium Problems
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Interval Analysis: Differential Equations
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Interval Analysis: Nondifferentiable Problems
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ▶ Interval Analysis: Systems of Nonlinear Equations
- ▶ Interval Analysis: Unconstrained and Constrained Optimization
- ▶ Interval Analysis: Verifying Feasibility
- ▶ Interval Constraints
- ▶ Interval Fixed Point Theory
- ▶ Interval Global Optimization
- ▶ Interval Linear Systems
- ▶ Interval Newton Methods

References

1. Babichev AB, Kadyrova OB, Kashevarova TP, Leshchenko AS, Semenov AL (1993) UniCalc, a novel approach to solving systems of algebraic equations. Interval Comput 2:29–47
2. Corliss GF, Kearfott RB (1998) Rigorous global search: Industrial applications. In: Csendes T (ed) (Special issues of the journal 'Reliable Computing'). Kluwer, Dordrecht
3. Griewank A and Corliss GF (eds) (1991) Automatic differentiation of algorithms: Theory, implementation, and application. SIAM, Philadelphia
4. Van Hentenryck P (1989) Constraint satisfaction in logic programming. MIT, Cambridge, MA
5. Van Hentenryck P, Michel L, Deville Y (1997) Numerica: A modeling language for global optimization. MIT, Cambridge, MA
6. Kearfott RB (1996) Rigorous global search: Continuous problems. Kluwer, Dordrecht
7. Rall LB (1981) Automatic differentiation: Techniques and applications. Lecture Notes Computer Sci, vol 120. Springer, Berlin

Interval Analysis: Nondifferentiable Problems

R. Baker Kearfott
Department Math., University Louisiana at Lafayette, Lafayette, USA

MSC2000: 65G20, 65G30, 65G40, 65H20

Article Outline

Keywords
Posing As Continuous Problems
A Special Method for Minimax Problems
  Treating As Continuous Problems
See also
References

Keywords

Nondifferentiability; Interval slopes; Verification; Interval computations; Global optimization

Nondifferentiable problems arise in various places in global optimization. One example is in l_1 and l_∞ optimization. That is,

$$\min_x \phi(x) = \min \|F\|_1 = \min_x \sum_{i=1}^{m} |f_i(x)| \tag{1}$$

and

$$\min_x \phi(x) = \min \|F\|_\infty = \min_x \left\{ \max_{1\le i\le m} |f_i(x)| \right\}, \quad (2)$$

where x is an n-vector, arise in data fitting, etc., and ϕ has a discontinuous gradient. In other problems, piecewise linear or piecewise quadratic approximations are used, and the gradient or the Hessian matrix are discontinuous. In fact, in some problems, even the objective function can be discontinuous.

Much thought has been given to nondifferentiability in algorithms to find local optima, and various techniques have been developed for local optimization. Some of these techniques can be used directly in interval global optimization algorithms. However, the power of interval arithmetic to bound the range of a point-valued function, even if that function is discontinuous, can be used to design effective algorithms for nondifferentiable or discontinuous problems whose structure is virtually identical to that of algorithms for differentiable or continuous problems.

Posing As Continuous Problems

Several techniques are available for re-posing problems as differentiable problems, in particular for Problem (1) and Problem (2). One such technique, suggested in [2, p. 74] and elsewhere, involves rewriting the forms $|e|$, $\max\{e_1, e_2\}$, and $\min\{e_1, e_2\}$ occurring in variable expressions in the objective and constraints in terms of additional constraints, as follows:

- Replace an expression $|e|$ by a new variable x_{n+1} and the two constraints $x_{n+1} \ge 0$ and $x_{n+1}^2 = e^2$.
- Replace $\max\{e_1, e_2\}$ by
$$\frac{e_1 + e_2 + |e_1 - e_2|}{2}.$$
- Replace $\min\{e_1, e_2\}$ by
$$\frac{e_1 + e_2 - |e_1 - e_2|}{2}.$$

Alternately, as explained in [1] and elsewhere, the entire Problems (1) and (2) can be replaced by constrained problems. In particular, (1) can be replaced by

$$\begin{cases} \min & \sum_{i=1}^m v_i \\ \text{s.t.} & v_i \ge f_i(x), \quad i = 1, \ldots, m, \\ & v_i \ge -f_i(x), \quad i = 1, \ldots, m, \end{cases} \quad (3)$$

where the v_i are new variables.

Likewise, (2) can be replaced by

$$\begin{cases} \min & v \\ \text{s.t.} & v \ge f_i(x), \quad i = 1, \ldots, m, \\ & v \ge -f_i(x), \quad i = 1, \ldots, m, \end{cases} \quad (4)$$

where v is a new variable.

A Special Method for Minimax Problems

In [4], a special interval algorithm for (2) is presented.

Treating As Continuous Problems

Due to inclusion properties of interval arithmetic, interval algorithms based on a particular degree of smoothness can be effective, essentially unchanged when less smoothness is present. In particular,

- If the objective function is discontinuous, algorithms designed for continuous objective functions can be used effectively.
- If the function is nonsmooth (that is, if the gradient has discontinuities), then algorithms based on second order information can often be used effectively.

For a brief discussion and further references for these general algorithms, see ► Interval analysis: Unconstrained and constrained optimization. For a more in-depth discussion of how continuous algorithms can be used for discontinuous problems, see [3, Chap. 6]. The main ideas are highlighted below.

Minima of ϕ: $\mathbf{R}^n \to \mathbf{R}^1$ can still be located when the objective ϕ is discontinuous because bounds on the range of ϕ are all that is necessary to do a branch and bound search. For a simple example, suppose

$$\phi(x) = \begin{cases} x^2 & \text{if } x \le 1, \\ 1 + x & \text{if } x > 1, \end{cases} \quad (5)$$

and suppose the interval $[-2, 2]$ is to be searched for global minima. For illustration purposes, suppose $\phi(0.25) = 0.125$ has been evaluated, so that 0.125 is an upper bound on the global optimum, and suppose the subinterval $\mathbf{x} = [0.5, 1.5]$ is to be analyzed. To obtain an interval enclosure for the range of ϕ over $\mathbf{x}$, we take

$$\begin{aligned} \phi(x) &\in [0.5, 1.0]^2 \underline{\cup} (1 + [1.0, 1.5]) \\ &= [0.25, 1.0] \underline{\cup} [2.0, 2.5] = [0.25, 2.5], \end{aligned}$$

where $\mathbf{a} \cup \mathbf{b}$ is the smallest interval that contains both $\mathbf{a}$ and $\mathbf{b}$. Thus, since $0.125 < [0.25, 2.5]$, a minimum of ϕ cannot possibly occur within the interval $[0.5, 1.5]$.

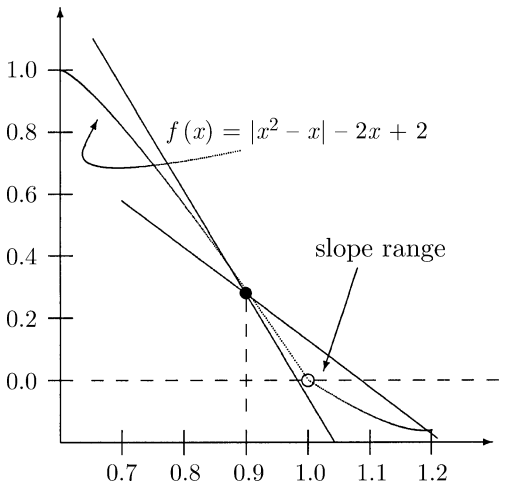
Similar considerations apply if the gradient $\nabla\phi$ is discontinuous. In such cases, the gradient test (see ► Interval analysis: Unconstrained and constrained optimization) will keep boxes that either contain zeros of the gradient or critical points corresponding to gradient discontinuities where the gradient changes sign.

When the gradient is discontinuous, interval Newton methods can still be used for iteration, as well as to verify existence. (See [3, (6.4) and (6.5), p. 217] for a formula; see ► Interval Newton methods for an introduction to interval Newton methods; see ► Interval fixed point theory for an explanation of interval fixed point theory.) Application to problems with discontinuous gradients is based on extended interval arithmetic (with infinities) and astute computation of slope bounds; see [3, pp. 214–215] for details.

Example 1 Consider

$$f(x) = |x^2 - x| - 2x + 2 = 0. \tag{6}$$

This function has both a root and a cusp at $x = 1$, with a left derivative of -3 and a right derivative of -1 at $x = 1$. If $1 \in \mathbf{x}$, then a slope enclosure is given by $\mathbf{S}(f, \mathbf{x}, x) = [-1, 1](\mathbf{x} + x - 1) - 2$.



Interval Analysis: Nondifferentiable Problems, Figure 1
The concept of a slope range for a nondifferentiable function

Consider using the interval Newton method

$$\widetilde{\mathbf{x}} \leftarrow \check{x}^{(k)} - \frac{f(\check{x}^{(k)})}{\mathbf{S}(f, \mathbf{x}^{(k)}, \check{x}^{(k)})},$$
$$\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} \cap \widetilde{\mathbf{x}},$$

with $\check{x}^{(k)}$ equal to the midpoint $\check{x} = 0.9$ of $\mathbf{x}^{(k)}$, and $\mathbf{x}^{(0)} = [0.7, 1.1]$, where $\mathbf{S}(f, \mathbf{x}^{(k)}, \check{x}^{(k)})$ is a bound on the slope enclosure of f at $\check{x}$. (See Fig. 1 for the concept of slope range.)

An initial slope enclosure is then $\mathbf{S}(f, [0.7, 1.1], 0.9) = [-3, -1]$,

$$\widetilde{\mathbf{x}} = .9 - \frac{.29}{[-3, -1]} = [.99\overline{6}, 1.19],$$

and $\mathbf{S}(f, [0.7, 1.1], 0.9) = [-3, -1]$. If this interval Newton method is iterated, then on iteration 3, existence of a root within $\mathbf{x}^{(3)}$ was proven, since $\mathbf{x}^{(3)} \subset \text{int}\mathbf{x}^{(2)}$, where $\text{int}\mathbf{x}^{(2)}$ is the interior of $\mathbf{x}^{(2)}$. For details, see [3, pp. 224–225].

See also

- ► Automatic Differentiation: Point and Interval
- ► Automatic Differentiation: Point and Interval Taylor Operators
- ► Bounding Derivative Ranges
- ► Global Optimization: Application to Phase Equilibrium Problems
- ► Interval Analysis: Application to Chemical Engineering Design Problems
- ► Interval Analysis: Differential Equations
- ► Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ► Interval Analysis: Intermediate Terms
- ► Interval Analysis: Parallel Methods for Global Optimization
- ► Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ► Interval Analysis: Systems of Nonlinear Equations
- ► Interval Analysis: Unconstrained and Constrained Optimization
- ► Interval Analysis: Verifying Feasibility
- ► Interval Constraints
- ► Interval Fixed Point Theory
- ► Interval Global Optimization
- ► Interval Linear Systems
- ► Interval Newton Methods

References

1. Gill PE, Murray W, Wright M (1981) Practical optimization. Acad. Press, New York
2. Van Hentenryck P, Michel L, Deville Y (1997) Numerica: A modeling language for global optimization. MIT, Cambridge, MA
3. Kearfott RB (1996) Rigorous global search: Continuous problems. Kluwer, Dordrecht
4. Shen Z, Neumaier A, Eiermann MC (1990) Solving minimax problems by interval methods. BIT 30:742–751

Interval Analysis for Optimization of Dynamical Systems

YOUDONG LIN, MARK A. STADTHERR
Department of Chemical and Biomolecular Engineering, University of Notre Dame, Notre Dame, USA

Article Outline

Introduction
Formulation
Methods
  Taylor Models
  Verifying Solver for Parametric ODEs
  Deterministic Global Optimization Method
Cases
  Catalytic Cracking of Gas Oil
  Singular Control Problem
Conclusions
References

Introduction

There are many applications of optimization for dynamical systems, including parameter estimation from time series data, determination of optimal operating profiles for batch and semibatch processes, optimal start-up, shutdown, and switching of continuous system, etc. To address such problems, one approach is to discretize any control profiles that appear as decision variables. There are then basically two types of methods available: (1) the complete discretization or simultaneous approach [20,28], in which both state variables and control profiles are discretized, and (2) the control parameterization or sequential approach [3,26], in which only the control profiles are discretized. In this article, only the sequential approach is considered. Since these problems are often nonconvex and thus may exhibit multiple local solutions, the classical techniques based on solving the necessary conditions for a local minimum may fail to determine the global optimum. This is true even for a rather simple temperature-control problem with a batch reactor [12]. Therefore, there is an interest in *global* optimization algorithms which can rigorously *guarantee* optimal performance.

There has been significant recent work on this problem. For example, Esposito and Floudas [6,7] used the αBB approach [1,2] to address the global optimization of dynamic systems. In this method, convex underestimating functions are used in connection with a branch-and-bound framework. A theoretical guarantee of attaining an ϵ-global solution is offered as long as rigorous underestimators are used, and this requires that sufficiently large values of α be used. However, this is difficult in this context because determining proper values of α depends on the Hessian of the function being underestimated, and this matrix is not available in explicit functional form when the sequential approach is used. Thus, as discussed in more detail by Papamichail and Adjiman [21], this approach does not provide a theoretical guarantee of global optimality. Alternative approaches have been given by Chachuat and Latifi [4] and by Papamichail and Adjiman [21,22] that do provide a theoretical guarantee of ϵ-global optimality; however, this is achieved at a high computational cost. Singer and Barton [25] have described a branch-and-bound approach for determining a theoretically guaranteed ϵ-global optimum with significantly less computational effort. In this method, convex underestimators and concave overestimators are used to construct two bounding initial value problems (IVPs), which are then solved to obtain lower and upper bounds on the trajectories of the state variables [24]. However, the bounding IVPs are solved using standard numerical methods that do not provide guaranteed error estimates, and so this approach does not provide fully guaranteed results from a computational standpoint.

In this article we discuss an approach [8,9] for the deterministic global optimization of dynamical systems based on interval analysis. A key feature of the method is the use of a verifying solver [10] for parametric ordinary differential equations (ODEs), which is used to produce guaranteed bounds on the solutions of dynamic systems with interval-valued parameters. This is

combined with a technique for domain reduction based on using Taylor models [19] in an efficient constraint propagation scheme. The result is that problems can be solved to global optimality with both mathematical and computational certainty.

Formulation

In this section we give the mathematical formulation of the nonlinear dynamic optimization problem to be addressed. Assume the system is described by the nonlinear ODE model $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$. Here $\boldsymbol{x}$ is the vector of state variables (length n) and $\boldsymbol{\theta}$ is a vector of adjustable parameters (length p), which may be a parameterization of a control profile $\theta(t)$. The model is given as an autonomous system; a nonautonomous system can easily be converted into autonomous form by treating the independent variable (t) as an additional state variable with derivative equal to 1. The objective function φ is expressed in terms of the adjustable parameters and the values of the states at discrete points $t_\mu, \mu = 0, 1, \ldots, r$. That is, $\phi = \phi\left[\boldsymbol{x}_\mu(\boldsymbol{\theta}), \boldsymbol{\theta};\ \mu = 0, 1, \ldots, r\right]$, where $\boldsymbol{x}_\mu(\boldsymbol{\theta}) = \boldsymbol{x}(t_\mu, \boldsymbol{\theta})$. If an integral appears in the objective function, it can be eliminated by introducing an appropriate quadrature variable.

The optimization problem is then stated as

$$\min_{\boldsymbol{\theta}, \boldsymbol{x}_\mu} \phi\left[\boldsymbol{x}_\mu(\boldsymbol{\theta}), \boldsymbol{\theta};\ \mu = 0, 1, \ldots, r\right] \tag{1}$$

$$\begin{aligned} \text{subject to} \quad \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}), \\ \boldsymbol{x}_0 &= \boldsymbol{x}_0(\boldsymbol{\theta}), \\ t &\in [t_0, t_r], \\ \boldsymbol{\theta} &\in \boldsymbol{\Theta}. \end{aligned}$$

Here $\boldsymbol{\Theta}$ is an interval vector that provides upper and lower parameter bounds (uppercase will be used to denote interval-valued quantities, unless noted otherwise). We assume that $\boldsymbol{f}$ is $(k-1)$ times continuously differentiable with respect to the state variables $\boldsymbol{x}$, and $(q+1)$ times continuously differentiable with respect to the parameters $\boldsymbol{\theta}$. We also assume that φ is $(q+1)$ times continuously differentiable with respect to the parameters $\boldsymbol{\theta}$. Here k is the order of the truncation error in the interval Taylor series (ITS) method to be used in the integration procedure, and q is the order of the Taylor model to be used to represent parameter dependence. When a typical sequential approach is used, an ODE solver is applied to the constraints with a given set of parameter values, as determined by the optimization routine. This effectively eliminates $\boldsymbol{x}_\mu, \mu = 0, 1, \ldots, r$, and leaves a bound-constrained minimization in the adjustable parameters $\boldsymbol{\theta}$ only. The method discussed here can also be extended to optimization problems with general state path constraints, and more general equality or inequality constraints on parameters. This is done by adapting the constraint propagation procedure (CPP) discussed below to handle the additional constraints.

Methods

Taylor Models

Makino and Berz [13] have described a remainder differential algebra (RDA) approach that uses Taylor models for bounding function ranges. This represents an approach for controlling the "dependency problem" of interval arithmetic, which leads to overestimation of function ranges. In the RDA approach, a function is represented using a model consisting of a Taylor polynomial and an interval remainder bound.

One way of forming a Taylor model of a function is by using a truncated Taylor series. Consider a function $f : \boldsymbol{x} \in \boldsymbol{X} \subset \mathbb{R}^m \to \mathbb{R}$ that is $(q+1)$ times partially differentiable on $\boldsymbol{X}$ and let $\boldsymbol{x}_0 \in \boldsymbol{X}$. The Taylor theorem states that for each $\boldsymbol{x} \in \boldsymbol{X}$, there exists a $\zeta \in \mathbb{R}$ with $0 < \zeta < 1$ such that

$$\begin{aligned} f(\boldsymbol{x}) = &\sum_{i=0}^{q} \frac{1}{i!}\left[(\boldsymbol{x} - \boldsymbol{x}_0)\cdot\nabla\right]^i f(\boldsymbol{x}_0) \\ &+ \frac{1}{(q+1)!}\left[(\boldsymbol{x} - \boldsymbol{x}_0)\cdot\nabla\right]^{q+1} f\left[\boldsymbol{x}_0 + (\boldsymbol{x} - \boldsymbol{x}_0)\zeta\right], \end{aligned} \tag{2}$$

where the partial differential operator $[\boldsymbol{g}\cdot\nabla]^k$ is

$$\begin{aligned} &[\boldsymbol{g}\cdot\nabla]^k \\ &= \sum_{\substack{j_1+\cdots+j_m=k \\ 0\le j_1,\cdots,j_m\le k}} \frac{k!}{j_1!\cdots j_m!} g_1^{j_1}\cdots g_m^{j_m} \frac{\partial^k}{\partial x_1^{j_1}\cdots\partial x_m^{j_m}}. \end{aligned} \tag{3}$$

The last (remainder) term in (2) can be quantitatively bounded over $0 < \zeta < 1$ and $\boldsymbol{x} \in \boldsymbol{X}$ using interval arithmetic or other methods to obtain an interval remainder bound R_f. The summation in (2) is a qth

order polynomial (truncated Taylor series) in $(\boldsymbol{x} - \boldsymbol{x_0})$ which we denote by $p_f(\boldsymbol{x} - \boldsymbol{x}_0)$. A qth order Taylor model T_f for $f(\boldsymbol{x})$ then consists of the polynomial p_f and the interval remainder bound R_f and is denoted by $T_f = (p_f, R_f)$. Note that $f \in T_f$ for $\boldsymbol{x} \in \boldsymbol{X}$ and thus T_f encloses the range of f over $\boldsymbol{X}$.

In practice, it is more useful to compute Taylor models of functions by performing Taylor model operations. Arithmetic operations with Taylor models can be done using the RDA operations described by Makino and Berz [13,14], which include addition, multiplication, reciprocal, and intrinsic functions. Therefore, it is possible to compute a Taylor model for any function representable in a computer environment by simple operator overloading through RDA operations. When RDA operations are performed, only the coefficients of p_f are stored and operated on; however, rounding errors are bounded and added to R_f. It has been shown that, compared with other rigorous bounding methods, the Taylor model can be used to obtain sharper bounds for modest to complicated functional dependencies [13,19].

An interval bound on a Taylor model $T = (p, R)$ over $\boldsymbol{X}$ is denoted by $B(T)$, and is found by determining an interval bound $B(p)$ on the polynomial part p and then adding the remainder bound; that is, $B(T) = B(p) + R$. The range bounding of the polynomials $B(p) = P(\boldsymbol{X} - \boldsymbol{x}_0)$ is an important issue, which directly affects the performance of Taylor model methods. Unfortunately, the exact range bounding of an interval polynomial is nondeterministic polynomial-time hard, and direct evaluation using interval arithmetic is very inefficient, often yielding only loose bounds. Thus, various bounding schemes [15,19] have been used, mostly focused on exact bounding of the dominant parts of P, i. e., the first- and second-order terms. However, exact bounding of a general interval quadratic is also computationally expensive (in the worst case, exponential in the number of variables m). Lin and Stadtherr [8] have adopted a very simple compromise approach, in which only the first-order and the *diagonal* second-order terms are considered for exact bounding, and other terms are evaluated directly. That is,

$$B(p) = \sum_{i=1}^{m} \left[a_i \, (X_i - x_{i0})^2 + b_i (X_i - x_{i0}) \right] + Q \,, \quad (4)$$

where Q is the interval bound of all other terms, and is obtained by direct evaluation with interval arithmetic. In (4), since X_i occurs twice, there exists a dependency problem. For $|a_i| \geq \omega$, where ω is a small positive number, (4) can be rearranged so that each X_i occurs only once; that is,

$$B(p) = \sum_{i=1}^{m} \left[a_i \left(X_i - x_{i0} + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} \right] + Q \,. \quad (5)$$

In this way, the dependence problem in bounding the interval polynomial is alleviated so that a sharper bound can be obtained. If $|a_i| < \omega$, direct evaluation can be used instead.

Verifying Solver for Parametric ODEs

When a traditional sequential approach is applied to the optimization of nonlinear dynamical systems, the objective function φ is evaluated, for a given value of $\boldsymbol{\theta}$, by applying an ODE solver to the constraints to eliminate the state variables $\boldsymbol{x}$. In the global optimization approach discussed here, a sequential approach based on interval analysis is used. This approach requires the evaluation of bounds on φ, given some parameter interval $\boldsymbol{\Theta}$. Thus, an ODE solver is needed that can compute bounds on $\boldsymbol{x}_\mu$, $\mu = 0, 1, \ldots, r$, for the case in which the parameters are interval-valued. Interval methods (also called validated methods or verified methods) for ODEs [16] provide a natural approach for computing the desired enclosure of the state variables at $t_\mu, \mu = 0, 1, \ldots, r$. An excellent review of interval methods for IVPs has been given by Nedialkov et al. [17]. Much work has been done for the case in which the initial values are given by intervals, and there are several software packages available that deal with this case. However, less work has been done on the case in which parameters are also given by intervals. In the global optimization method discussed here, a verifying solver for parametric ODEs [10], called VSPODE, is used to produce guaranteed bounds on the solutions of dynamic systems with interval-valued initial states and parameters. In this section, we review the key ideas behind the method used in VSPODE, and outline the procedures used. Additional details are given by Lin and Stadtherr [10].

Consider the parametric ODE system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}), \quad \boldsymbol{x}_0 \in \boldsymbol{X}_0, \quad \boldsymbol{\theta} \in \boldsymbol{\Theta}, \quad (6)$$

where $t \in [t_0, t_r]$ for some $t_r > t_0$. The interval vectors $\boldsymbol{X}_0$ and $\boldsymbol{\Theta}$ represent enclosures of initial values and parameters, respectively. It is desired to determine a verified enclosure of all possible solutions to this initial value problem. We denote by $\boldsymbol{x}(t; t_j, \boldsymbol{X}_j, \boldsymbol{\Theta})$ the set of solutions $\boldsymbol{x}(t; t_j, \boldsymbol{X}_j, \boldsymbol{\Theta}) = \{\boldsymbol{x}(t; t_j, \boldsymbol{x}_j, \boldsymbol{\theta}) \mid \boldsymbol{x}_j \in \boldsymbol{X}_j, \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$, where $\boldsymbol{x}(t; t_j, \boldsymbol{x}_j, \boldsymbol{\theta})$ denotes a solution of $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$ for the initial condition $\boldsymbol{x} = \boldsymbol{x}_j$ at $t = t_j$. We will outline a method for determining enclosures $\boldsymbol{X}_j$ of the state variables at each time step $j = 1, \ldots, r$, such that $\boldsymbol{x}(t_j; t_0, \boldsymbol{X}_0, \boldsymbol{\Theta}) \subseteq \boldsymbol{X}_j$.

Assume that at t_j we have an enclosure $\boldsymbol{X}_j$ of $\boldsymbol{x}(t_j; t_0, \boldsymbol{X}_0, \boldsymbol{\Theta})$, and that we want to carry out an integration step to compute the next enclosure $\boldsymbol{X}_{j+1}$. Then, in the first phase of the method, the goal is to find a step size $h_j = t_{j+1} - t_j > 0$ and an a priori enclosure (coarse enclosure) $\widetilde{\boldsymbol{X}}_j$ of the solution such that a unique solution $\boldsymbol{x}(t; t_j, \boldsymbol{x}_j, \boldsymbol{\theta}) \in \widetilde{\boldsymbol{X}}_j$ is guaranteed to exist for all $t \in [t_j, t_{j+1}]$, all $\boldsymbol{x}_j \in \boldsymbol{X}_j$, and all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. One can apply a traditional interval method, with high-order enclosure, to the parametric ODEs by using an ITS with respect to time. That is, h_j and $\boldsymbol{X}_j$ are determined such that for $\boldsymbol{X}_j \subseteq \widetilde{\boldsymbol{X}}_j^0$,

$$\widetilde{\boldsymbol{X}}_j = \sum_{i=0}^{k-1} [0, h_j]^i \boldsymbol{F}^{[i]}(\boldsymbol{X}_j, \boldsymbol{\Theta}) + [0, h_j]^k \boldsymbol{F}^{[k]}(\widetilde{\boldsymbol{X}}_j^0, \boldsymbol{\Theta}) \subseteq \widetilde{\boldsymbol{X}}_j^0. \tag{7}$$

Here $\widetilde{\boldsymbol{X}}_j^0$ is an initial estimate of $\widetilde{\boldsymbol{X}}_j$, k denotes the order of the Taylor expansion, and the coefficients $\boldsymbol{F}^{[i]}$ are interval extensions of the Taylor coefficients $\boldsymbol{f}^{[i]}$ of $\boldsymbol{x}(t)$ with respect to time. Satisfaction of (7) demonstrates [5] that there exists a unique solution $\boldsymbol{x}(t; t_j, \boldsymbol{x}_j, \boldsymbol{\theta}) \in \widetilde{\boldsymbol{X}}_j$ for all $t \in [t_j, t_{j+1}]$, all $\boldsymbol{x}_j \in \boldsymbol{X}_j$, and all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

In the second phase of the method, a tighter enclosure $\boldsymbol{X}_{j+1} \subseteq \widetilde{\boldsymbol{X}}_j$ is computed such that $\boldsymbol{x}(t_{j+1}; t_0, \boldsymbol{X}_0, \boldsymbol{\Theta}) \subseteq \boldsymbol{X}_{j+1}$. This is done by using an ITS approach to compute a Taylor model $\boldsymbol{T}_{\boldsymbol{x}_{j+1}}$ of $\boldsymbol{x}_{j+1}$ in terms of the parameter vector $\boldsymbol{\theta}$ and initial state vector $\boldsymbol{x}_0$, and then obtaining the enclosure $\boldsymbol{X}_{j+1} = B(\boldsymbol{T}_{\boldsymbol{x}_{j+1}})$ by bounding $\boldsymbol{T}_{\boldsymbol{x}_{j+1}}$ over $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and $\boldsymbol{x}_0 \in \boldsymbol{X}_0$. To determine enclosures of the ITS coefficients $\boldsymbol{f}^{[i]}(\boldsymbol{x}_j, \boldsymbol{\theta})$ an approach combining RDA operations with the mean value theorem is used to obtain the Taylor models $\boldsymbol{T}_{\boldsymbol{f}^{[i]}}$. Now using an ITS for $\boldsymbol{x}_{j+1}$ with coefficients given by $\boldsymbol{T}_{\boldsymbol{f}^{[i]}}$, one can obtain a result for $\boldsymbol{T}_{\boldsymbol{x}_{j+1}}$ in terms of the parameters and initial states. In order to address the wrapping effect [16], results are propagated from one time step to the next using a type of Taylor model in which the remainder bound is not an interval but a parallelepiped. That is, the remainder bound is a set of the form $\mathcal{P} = \{Av \mid v \in V\}$, where $A \in \mathbb{R}^{n \times n}$ is a real and regular matrix. If A is orthogonal, as from a QR-factorization, then $\mathcal{P}$ can be interpreted as a rotated n-dimensional rectangle. Complete details of the computation of $\boldsymbol{T}_{\boldsymbol{x}_{j+1}}$ were given by Lin and Stadtherr [10].

The approach outlined above, as implemented in VSPODE, has been tested by Lin and Stadtherr [10], who compared its performance with results obtained using the popular VNODE package [18]. For the test problems used, VSPODE provided tighter enclosures on the state variables than VNODE, and required significantly less computation time.

Deterministic Global Optimization Method

In this section, we summarize a method for the deterministic global optimization of dynamical systems, based on the use of the tools described above. As noted previously, when a sequential approach is used, the state variables are effectively eliminated using the ODE constraints, in this case by employing VSPODE, leaving a bound-constrained minimization of $\phi(\boldsymbol{\theta})$ with respect to the adjustable parameters (decision variables) $\boldsymbol{\theta}$. The optimization method discussed here can be thought of as a type of branch-and-bound method, with a CPP used for domain reduction. Therefore, it can also be viewed as a branch-and-reduce algorithm. The basic idea is that only those parts of the decision variable space $\boldsymbol{\Theta}$ that satisfy the constraint $c(\boldsymbol{\theta}) = \phi(\boldsymbol{\theta}) - \widehat{\phi} \leq 0$, where $\widehat{\phi}$ is a known upper bound on the global minimum found using local minimization, need to be retained. To perform this domain reduction, a CPP can be used.

Partial information expressed by a constraint can be used to eliminate incompatible values from the domain of its variables. This domain reduction can then be propagated to all constraints on that variable, where it may be used to further reduce the domains of other variables. This process is known as constraint propaga-

tion. It is applied to a sequence of subintervals of $\boldsymbol{\Theta}$, which arises in a bisection process. For a subinterval $\boldsymbol{\Theta}^{(k)}$, the Taylor model T_{ϕ_k} of the objective function φ over $\boldsymbol{\Theta}^{(k)}$ is computed. To do this, Taylor models of $\boldsymbol{x}_\mu$, the state variables at times t_μ, $\mu = 1, \ldots, r$, in terms of $\boldsymbol{\theta}$ are determined using VSPODE. Note that T_{ϕ_k} then consists of a qth order polynomial in the decision variables $\boldsymbol{\theta}$, plus a remainder bound. The part of $\boldsymbol{\Theta}^{(k)}$ that can contain the global minimum must satisfy the constraint $c(\boldsymbol{\theta}) = \phi(\boldsymbol{\theta}) - \widehat{\phi} \leq 0$. In the CPP outlined here, $B(T_c)$ is determined and then there are three possible outcomes (in the following, an underline is used to indicate the lower bound of an interval, and an overline is used to indicate the upper bound):

1. If $\underline{B(T_c)} > 0$, then no $\boldsymbol{\theta} \in \boldsymbol{\Theta}^{(k)}$ will ever satisfy the constraint; thus, the CPP can be stopped and $\boldsymbol{\Theta}^{(k)}$ discarded. Testing for this outcome amounts to checking if the lower bound of T_{ϕ_k}, $\underline{B(T_{\phi_k})}$, is greater than $\widehat{\phi}$. If so, then $\boldsymbol{\Theta}^{(k)}$ can be discarded because it cannot contain the global minimum and need not be tested further.
2. If $\overline{B(T_c)} \leq 0$, then every $\boldsymbol{\theta} \in \boldsymbol{\Theta}^{(k)}$ will always satisfy the constraint; thus, $\boldsymbol{\Theta}^{(k)}$ cannot be reduced and the CPP can be stopped. This amounts to checking if the upper bound of T_{ϕ_k}, $\overline{B(T_{\phi_k})}$, is less than $\widehat{\phi}$. This also indicates, with certainty, that there is a point in $\boldsymbol{\Theta}^{(k)}$ that can be used to update $\widehat{\phi}$, which can then be done using a local optimization routine.
3. If neither of the previous two cases occur, then part of the interval $\boldsymbol{\Theta}^{(k)}$ may be eliminated. To do this, an approach [8,9] based on the range bounding strategy for Taylor models is used, as given by (5). If insufficient reduction of $\boldsymbol{\Theta}^{(k)}$ occurs, then it is bisected and the resulting subintervals are added to the sequence of subintervals to be processed.

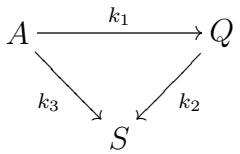
Complete details of the optimization method based on these ideas were given by Lin and Stadtherr [8,9]. It can be implemented either as an ϵ-global algorithm, or, by incorporating interval-Newton steps in the method, as an exact ($\epsilon = 0$) algorithm. The latter requires the application of VSPODE to the first- and second-order sensitivity equations. An exact algorithm using interval-Newton steps was implemented by Lin and Stadtherr [8] for the special case of parameter estimation problems. However, this has not been fully implemented for more general cases.

Cases

Lin and Stadtherr [8,9] have tested the performance of the algorithm discussed above on a variety of test problems. In this section we summarize the results for two of these problems. Both example problems were solved using an Intel Pentium 4 3.2 GHz machine running Red Hat Linux. The VSPODE package [9], with a $k = 17$ order ITS, $q = 3$ order Taylor model, and QR approach for wrapping, was used to integrate the dynamical systems in each problem. Using a smaller value of k will result in the need for smaller step sizes in the integration and so will tend to increase computation time. Using a larger value of q will result in somewhat tighter bounds on the states, though at the expense of additional complexity in the Taylor model computations.

Catalytic Cracking of Gas Oil

This problem involves parameter estimation in a model representing the catalytic cracking of gas oil (A) to gasoline (Q) and other side products (S), as described by Tjoa and Biegler [27] and also studied by several others [4,7,22,25]. The reaction is



Only the concentrations of A and Q were measured. This reaction scheme involves nonlinear reaction kinetics. A least-squares objective was used for parameter estimation, resulting in the optimization problem

$$\min_{\boldsymbol{\theta}} \phi = \sum_{\mu=1}^{20} \sum_{i=1}^{2} \left(\widehat{x}_{\mu,i} - x_{\mu,i}\right)^2$$

$$\begin{aligned}
\text{subject to} \quad & \dot{x}_1 = -(\theta_1 + \theta_3) x_1^2 \,, \\
& \dot{x}_2 = \theta_1 x_1^2 - \theta_2 x_2 \,, \\
& t \in [0, 0.95] \,, \\
& \boldsymbol{x}_\mu = \boldsymbol{x}(t_\mu) \,, \\
& \boldsymbol{x}_0 = (1, 0)^{\mathrm{T}} \,, \\
& \boldsymbol{\theta} \in [0, 20] \times [0, 20] \times [0, 20] \,,
\end{aligned}$$

where $\widehat{\boldsymbol{x}}_\mu$ is given experimental data. Here the state vector, $\boldsymbol{x}$, is defined as the concentration vector $(A, Q)^{\mathrm{T}}$ and the parameter vector, $\boldsymbol{\theta}$, is defined as $(k_1, k_2, k_3)^{\mathrm{T}}$.

For the ϵ-global algorithm, with a relative convergence tolerance of $\epsilon^{\text{rel}} = 10^{-3}$, 14.3 s was required to solve this problem. For the exact ($\epsilon = 0$) global algorithm using interval-Newton, 11.5 s was required. For this problem, the exact algorithm required less computation than the ϵ-global algorithm. However, this may or may not be the case for other problems [8]. Papamichail and Adjiman [22] solved this problem to ϵ-global optimality in 35,478 s (Sun UltraSPARC-II 360 MHz; Matlab), and Chachuat and Latifi [4] obtained an ϵ-global solution in 10,400 s (unspecified machine; prototype implementation). Singer and Barton [25] solved this problem to ϵ-global optimality for a series of absolute tolerances, so their results are not directly comparable. However, the computational cost of their method on this problem appears to be quite low. These other methods all provide for ϵ-convergence only.

Singular Control Problem

This example is a nonlinear singular optimal control problem originally formulated by Luus [11] and also considered by Esposito and Floudas [7], Chachuat and Latifi [4], and Singer and Barton [25]. This problem is known to have multiple local solutions. In autonomous form and using a quadrature variable, this problem is given by

$$
\begin{aligned}
&\min_{\theta(t)} \phi = x_5(t_f) \\
&\text{subject to} \quad \dot{x}_1 = x_2 \,, \\
&\qquad \dot{x}_2 = -x_3\theta + 16x_4 - 8 \,, \\
&\qquad \dot{x}_3 = \theta \,, \\
&\qquad \dot{x}_4 = 1 \,, \\
&\qquad \dot{x}_5 = x_1^2 + x_2^2 \\
&\qquad\quad + 0.0005(x_2 + 16x_4 - 8 - 0.1x_3\theta^2)^2 \,, \\
&\qquad \boldsymbol{x}_0 = (0, -1, -\sqrt{5}, 0, 0)^{\mathrm{T}} \,, \\
&\qquad t \in [t_0, t_f] = [0, 1] \,, \\
&\qquad \theta \in [-4, 10] \,.
\end{aligned}
\tag{8}
$$

The control $\theta(t)$ is parameterized as a piecewise constant profile with a specified number of equal time intervals. Five problems are considered, corresponding to one, two, three, four, and five time intervals in the parameterization. Each problem was solved to an absolute tolerance of $\epsilon^{\text{abs}} = 10^{-3}$. Computational results [9] are presented in Table 1. This shows, for each problem, the globally optimal objective value ϕ^* and the corresponding optimal controls $\boldsymbol{\theta}^*$, as well as the CPU time (in seconds) and number of iterations required. Chachuat and Latifi [4] solved the two-interval problem to ϵ-global optimality using four different strategies, with the most efficient requiring 502 CPU seconds, using an unspecified machine and a "prototype" implementation. Singer and Barton [25] solved the one-, two-, and three-interval cases with $\epsilon^{\text{abs}} = 10^{-3}$ using two different problem formulations (with and without a quadrature variable) and two different implementations (with and without branch-and-bound heuristics). The best results in terms of efficiency were achieved with heuristics and without a quadrature variable, with CPU times of 1.8, 22.5, and 540.3 s (1.667 GHz AMD Athlon XP2000+) for the one-, two-, and three-interval problems, respectively. This compares with CPU times of 0.02, 0.32, and 10.88 s (3.2 GHz Intel Pentium 4) for the method discussed here. Even accounting for the roughly factor of 2 difference in the speeds of the machines used, the method described here appears to be well over an order of magnitude faster. The four- and five-interval problems were solved [9] in 369 and 8,580.6 CPU seconds, respectively, and apparently had not been solved previously using a method rigorously guaranteed to find an ϵ-global minimum. It should be noted that the solution to the three-interval problem, as given in Table 1, differs from the result reported by Singer and Barton [25], which is known to be a misprint [23].

Conclusions

In this article, we have described an approach for the deterministic global optimization of dynamical systems, including parameter estimation and optimal control problems. This method [8,9] is based on interval analysis and Taylor models and employs a type of sequential approach. A key feature of the method is the use of a new verifying solver [10] for parametric ODEs, which is used to produce guaranteed bounds on the solutions of dynamic systems with interval-valued parameters. This is combined with techniques for domain reduction based on using Taylor models in an efficient constraint propagation scheme. The result is that problems can be solved to global op-

Interval Analysis for Optimization of Dynamical Systems, Table 1
Results [9] for the singular control problem

Time intervals	φ^*	θ^*	CPU time (s)	No. of iterations
1	0.4965	(4.071)	0.02	9
2	0.2771	(5.575, −4.000)	0.32	71
3	0.1475	(8.001, −1.944, 6.042)	10.88	1, 414
4	0.1237	(9.789, −1.200, 1.257, 6.256)	369.0	31, 073
5	0.1236	(10.00, 1.494, −0.814, 3.354, 6.151)	8, 580.6	493, 912

timality with both mathematical and computational certainty. On parameter estimation problems, an exact ($\epsilon = 0$) algorithm, using interval-Newton steps, can be applied at a cost comparable to, and perhaps less than, that of the ϵ-global algorithm. The new approach can provide significant improvements in computational efficiency, potentially well over an order of magnitude, relative to other recently described methods.

References

1. Adjiman CS, Androulakis IP, Floudas CA, Neumaier A (1998) A global optimization method, αBB, for general twice-differentiable NLPs–I. Theoretical advances. Comput Chem Eng 22:1137–1158
2. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB, for general twice-differentiable NLPs–II. Implementation and computational results. Comput Chem Eng 22:1159–1179
3. Brusch R, Schappelle R (1973) Solution of highly constrained optimal control problems using nonlinear programming. AIAA J 11:135–136
4. Chachuat B, Latifi MA (2004) A new approach in detetermin-istic global optimisation of problems with ordinary differential equations. In: Floudas CA, Pardalos PM (eds) Frontiers in Global Optimization. Kluwer, Dordrecht
5. Corliss GF, Rihm R (1996) Validating an a priori enclosure using high-order Taylor series. In: Alefeld G, Frommer A (eds) Scientific Computing : Computer Arithmetic, and Validated Numerics. Akademie Verlag, Berlin
6. Esposito WR, Floudas CA (2000) Deterministic global optimization in nonlinear optimal control problems. J Global Optim 17:97–126
7. Esposito WR, Floudas CA (2000) Global optimization for the parameter estimation of differential-algebraic systems. Ind Eng Chem Res 39:1291–1310
8. Lin Y, Stadtherr MA (2006) Determinstic global optimization for parameter estimation of dynamic systems. Ind Eng Chem Res 45:8438–8448
9. Lin Y, Stadtherr MA (2007) Deterministic global optimization of nonlinear dynamic systems. AIChE J 53:866–875
10. Lin Y, Stadtherr MA (2007) Validated solutions of initial value problems for parametric ODEs. Appl Num Math 58:1145–1162
11. Luus R (1990) Optimal control by dynamic programming using systematic reduction in grid size. Int J Control 51:995–1013
12. Luus R, Cormack DE (1972) Multiplicity of solutions resulting from the use of variational methods in optimal control problems. Can J Chem Eng 50:309–311
13. Makino K, Berz M (1999) Efficient control of the dependency problem based on Taylor model methods. Reliab Comput 5:3–12
14. Makino K, Berz M (2003) Taylor models and other validated functional inclusion methods. Int J Pure Appl Math 4:379–456
15. Makino K, Berz M (2005) Verified global optimization with Taylor model-based range bounders. Trans Comput 11:1611–1618
16. Moore RE (1966) Interval Analysis. Prentice-Hall, Englewood Cliffs
17. Nedialkov NS, Jackson KR, Corliss GF (1999) Validated solutions of initial value problems for ordinary differential equations. Appl Math Comput 105:21–68
18. Nedialkov NS, Jackson KR, Pryce JD (2001) An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE. Reliab Comput 7:449–465
19. Neumaier A (2003) Taylor forms – Use and limits. Reliab Comput 9:43–79
20. Neuman C, Sen A (1973) A suboptimal control algorithm for constraint problems using cubic splines. Automatica 9:601–613
21. Papamichail I, Adjiman CS (2002) A rigorous global optimization algorithm for problems with ordinary differential equations. J Global Optim 24:1–33
22. Papamichail I, Adjiman CS (2004) Global optimization of dynamic systems. Comput Chem Eng 28:403–415
23. Singer AB (2006) Personal communication
24. Singer AB, Barton PI (2006) Bounding the solutions of parameter dependent nonlinear ordinary differential equations. SIAM J Sci Comput 27:2167–2182
25. Singer AB, Barton PI (2006) Global optimization with nonlinear ordinary differential equations. J Global Optim 34:159–190

26. Teo K, Goh G, Wong K (1991) A unified computational approach to optimal control problems. Pitman Monographs and Surveys in Pure and Applied Mathematics, vol 55. Wiley, New York
27. Tjoa TB, Biegler LT (1991) Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. Ind Eng Chem Res 30:376
28. Tsang TH, Himmerlblau DM, Edgar TF (1975) Optimal control via collocation and nonlinear programming. Int J Control 21:763–768

Interval Analysis: Parallel Methods for Global Optimization

ANTHONY P. LECLERC
College of Charleston, Charleston, USA

MSC2000: 65K05, 65Y05, 65Y10, 65Y20, 68W10

Article Outline

Keywords and Phrases
Introduction
Definitions
  Interval Arithmetic
  Sequential IA Branch and Bound
Formulation
  Parallel Computer Models
  PIAGO
  Workload Management
  Load Balancing
  Superlinear Speedup
Methods
  Distributed Approaches
  Centralized Approaches
  Hybrid Approaches
Conclusions
See also
References

Keywords and Phrases

Optimization; Branch and bound; Covering methods; Parallel algorithms; Mathematical programming; Interval analysis

Introduction

The ability of interval arithmetic (IA) [21,22,23,24] to automatically compute reliable solution bounds in numerical computations makes it an ideal mechanism for solving continuous nonlinear global optimization problems. To date, most efforts at developing parallel IA methods for global optimization have used the branch and bound (B&B) global search strategy [1,4,8]. The sequential B&B-based IA global optimization algorithm [10,17] executes a tree-like search process which is naturally parallelized and amenable to massive coarse-grained data parallelism (i. e. workload scalable [14]).

Several noteworthy advances in parallel algorithms for global optimization using interval arithmetic have occurred over the past few years [7,15,26]. In addition, new software packages have been developed as a result of recent implementations of new or existing parallel IA global optimization algorithms [13,29]. A parallel programming language expressing a *message-driven* model is utilized in one implementation, resulting in a significantly different computational flow than is typical with the more classic and popular message-passing (e. g. MPI, PVM) and shared-memory (e. g. pthreads) parallel implementations [20]. Recently, the ubiquity of *multi-core* processor architectures has opened up new possibilities for exploiting thread-level parallelism.

In the sections that follow, a sequential (B&B) IA global optimization algorithm is presented along with relevant IA and parallel computing definitions. Next, a general formulation of a parallel IA global optimization algorithm (PIAGO) based on the B&B global search strategy is presented. In the methods section, a survey of recent algorithmic advances, novel implementations, and pertinent language and programming environments is discussed. Finally, some concluding remarks are made along with thoughts on fertile future research avenues.

Definitions

Interval Arithmetic

A "box" is an n-dimensional interval:

$$\begin{aligned} \boldsymbol{X} &= \left\{\vec{x} : \underline{\boldsymbol{x}_i} \le x_i \le \overline{\boldsymbol{x}_i},\ i = 0, 1, \ldots, n-1\right\} \\ &= \left([\underline{\boldsymbol{x}_0}, \overline{\boldsymbol{x}_0}], [\underline{\boldsymbol{x}_1}, \overline{\boldsymbol{x}_1}], \ldots, [\underline{\boldsymbol{x}_{n-1}}, \overline{\boldsymbol{x}_{n-1}}]\right)^{\mathrm{T}} \\ &= \left(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}\right)^{\mathrm{T}}. \end{aligned}$$

Boldface letters and capital letters are used to denote interval quantities and vectors, respectively (as

proposed in [17]). The midpoint of $\boldsymbol{X}$ is denoted, $\mathrm{m}(\boldsymbol{X})$. The width of $\boldsymbol{X}$ is denoted, $\mathrm{w}(\boldsymbol{X})$. The greatest lower bound and the least upper bound for the interval $\boldsymbol{x}$ is denoted $\underline{\boldsymbol{x}}$ and $\overline{\boldsymbol{x}}$, respectively.

Sequential IA Branch and Bound

A canonical sequential (B&B) IA global optimization algorithm, SIAGO, iterates over a prioritized list of boxes, representing current candidate subregions (of the initial search space) for containing global minimizer(s). The prioritized list, Q, is typically implemented as a heap data structure (see Algorithm 1). In each iteration, a box, $\boldsymbol{X}$, is removed from Q. If $\mathrm{w}(\boldsymbol{X})$ and $\mathrm{w}(f(\boldsymbol{X}))$ are less than the prescribed tolerances, ϵ_x and ϵ_f, respectively, then $\boldsymbol{X}$ is placed on the solution list, S; Otherwise, $\boldsymbol{X}$ is subdivided into smaller boxes, $\boldsymbol{X_0}, \boldsymbol{X_1}, \ldots, \boldsymbol{X_{k-1}}$. Each of the k boxes, $\boldsymbol{X}_i$, is subjected to a set of deletion/reduction tests. Surviving $\boldsymbol{X}_i$ boxes are placed onto Q.

The boolean operator, *Delete()*, takes as input a box, $\boldsymbol{X}$, and a floating point number, U^* (the upper bound on the smallest function value known thus far), and returns TRUE if and only if one of the following tests returns TRUE:

- $\underline{f(\boldsymbol{X})} > U^*$
- if $\boldsymbol{X}$ is strictly feasible (i. e. does not lie on the boundary of the feasible space) and $0 \notin \nabla_i f(\boldsymbol{X})$ (the gradient) for some $i = 0, \ldots, n-1$,
- if $\boldsymbol{X}$ is strictly feasible and the Hessian, $\nabla^2 f(\boldsymbol{X})$, is not positive semi-definite anywhere in $\boldsymbol{X}$
- interval Newton's method can eliminate all of $\boldsymbol{X}$.

These tests are known as the *midpoint test*, *monotonicity test*, *Hessian test*, and *Newton test*, respectively. More elaborate versions of SIAGO exist today (e. g. Newton's method box reduction, unique critical point existence tests) [10,16] but have little effect on the survey in Sect. "Methods" of parallel IA global optimization algorithms[1].

Formulation

The following two facts about SIAGO (see Algorithm 1) reveal a potential for scalable parallelism.

First, *Delete()* can be performed independently on different feasible subregions and therefore can be done in parallel. This allows for "massive" data parallelism as sub-boxes can be distributed across all processors.

Actually, some dependence exists for the midpoint test (see U^* in Algorithm 1). However, this dependence only affects the "sharpness" of this test and not the correctness. In practice, newly discovered lower U^* values are shared among all participating processors via broadcasts or shared memory (see Sect. "Parallel Computer Models").

Second, if a feasible region is not deleted (and not reduced via interval Newton's method), the procedure, *Subdivide()*, will divide it into k subregions which together entirely cover the whole feasible region. The workload has just grown by k.

Such workload growth makes possible workload scalability [14]. This means that the workload can scale to match the parallel computing power (i. e. CPU utilization is optimized). In fact, the workload growth of SIAGO is potentially exponential and for high dimensional problems can overtake the parallel computing power and memory resources.

The exponential workload growth of SIAGO is no surprise in that the global optimization problem in general is *NP*-hard (i. e. no algorithm has yet been found which is better than simply performing a complete space search for the solution, requiring exponen-

```
U* = ∞, Q.insert(X);   // initial box while true do
  repeat
    if Q.empty then S.print, Halt   Q.remove(X)
  until f(X) ≤ U*;          // cut-off test
  if WithinTol(X, ε_x, ε_f, U*) then
    S.insert(X)
  else
    Subdivide(X, X_0, X_1, ..., X_{k-1})   for i=0 to k-1
    do
      if not Delete(X_i, U*) then
        U* = min(f(m(X_i)), U*)   Q.insert(X_i)
      end
    end
  end
end
```

Interval Analysis: Parallel Methods for Global Optimization, Algorithm 1
SIAGO

[1] SIAGO efficiency can affect experimental parallel speedup measurements as noted in Sect. "Superlinear Speedup"

tial time in the worst case). IA allows one to remove (or reduce using interval Newton's method) potentially large "chunks" of the search space with the hope of pruning/squeezing one's way to a solution.

Parallel Computer Models

A parallel version of SIAGO is implementable on two basic categories of parallel computers: shared memory multiprocessors (including multi-core processors common today) and distributed memory multicomputers. Although multiprocessors are easier to program than multicomputers (one doesn't have to worry about communication primitives), multicomputers have been the most popular choice for PIAGO for several reasons. First, multicomputers are more scalable. Second, there are freely-available, robust, easy-to-use parallel programming language extensions such as Parallel Virtual Machine (PVM) and the Message Passing Interface (MPI). Third, multicomputers are more cost effective (e. g. a simple cluster of workstations (COW) with inexpensive gigabit Ethernet). Fourth, the massive workload generated by PIAGO implementations on hard global optimization problems (for which PIAGO algorithms were designed to solve) keeps each processor busy working on a local subregion of the search space. If an effective workload management scheme is adopted (see Sect. "Workload Management"), CPU utilization will be maximized and communication will not be a limiting factor.

PIAGO

A generalized distributed memory parallel IA global optimization algorithm (PIAGO) has the following form:

```
Initialize/Startup all processors
Perform SIAGO in parallel
    manage workload
    broadcast improved U* values
Detect global termination state
Terminate all processors
```

Interval Analysis: Parallel Methods for Global Optimization, Algorithm 2
PIAGO

Workload Management

Workload in PIAGO algorithms is characterized at any given time by the set of boxes remaining to be processes or searched. PIAGO methods distinguish themselves primarily in the manner they manage workload (see Algorithm 2). In SIAGO, workload resides on a single priority queue of boxes, Q. In PIAGO, workload can be centrally managed on a single "master" node (processor), or it can be distributed among all nodes with each processor managing its own local Q. Hybrid schemes can also be employed consisting of a centrally managed global priority search queue on the master node working in concert with local search queues on each slave node.

Distributed Management In this scheme, workload is distributed either statically to all processors at the beginning of the computation (*static load balancing*) or dynamically during computation (*dynamic load balancing*). With dynamic load balancing, processors coordinate and redistribute workload during computation in order to maximize CPU utilization and minimize total execution time. Workload state information (e. g. local search queue size) is continually (but not necessarily frequently) broadcast among all processors.

Dynamic load balancing is generally scalable. However, each processor must communicate (by request, event, or at programmed time intervals) workload state information in order to make effective workload balancing decisions. Too much state information being broadcast frequently detracts from box processing and may saturate the machine's bandwidth. Stale information concerning a processor's state risks poor load balancing decisions being made on an inaccurate depiction of the current global state.

Centralized Management In this scheme (sometimes called master/slave), one master node is responsible for managing (scheduling) the workload. Slave (worker) nodes request work (or are "pushed" work) from the master. The master node is responsible for scheduling the workload in a way that maximizes CPU utilization and (hopefully) minimizes total execution time.

One advantage of centralized control is that workload can be prioritized globally (e. g. boxes, $\boldsymbol{X}$, ordered on a priority queue based on minimum $\underline{f(\boldsymbol{X})}$). Global

termination detection is easy: The computation is done when the master node has no more workload and all slave processors are idle. Load balancing is achieved through effective scheduling.

Centralized workload management is not scalable, in the theoretical sense. In practice, a centralized scheme is successful provided the master node does not become a "bottleneck". Communication between one master node and a large set of worker nodes can become intensive and exhaust the communication bandwidth of the parallel machine. Moreover, memory and CPU resources on one processor are limited (relative to the total CPU power and memory of the parallel machine) and can easily become saturated if stressed with too much workload or communication.

Hybrid Workload Management Hybrid schemes allow each worker processor to manage its own local Q while still maintaining a master process responsible for handling work requests from idle processors. The benefits of the hybrid approach over the pure centralized approach are two-fold:

- fewer requests for work (to the master) are required since each worker must first complete its local workload (including self-generated workload resulting from box splitting) before it becomes idle
- the potential memory bottleneck at the master is mitigated since the local memory resources on each worker are utilized.

The main disadvantage of the hybrid approach versus the centralized approach is the sacrifice of total (global) workload ordering. The master node "running out of boxes" or a worker process generating too much work to be held in local memory are two other issues that need to be addressed.

The main advantages of the hybrid approach compared to the distributed approach are two-fold:

- a better approximation to a total workload ordering
- fewer possible retransmissions for work as the master node is (usually) guaranteed to have boxes.

Because the hybrid approach still uses a master node for scheduling workload, this method inherits the scalability weakness of its centralized parent.

Load Balancing

One necessary condition for load balancing is ensuring no worker processor sits idle. A second goal of load balancing in PIAGO algorithms is the distribution of "quality" boxes among the worker processors. A quality box is defined as a box more likely to contain a minimizer (or near minimizer). It is natural to expect that global minima will be discovered more quickly if participating processors focus their efforts on subregions of the workspace that more likely to contain minimizers. Early improvements to the SIAGO algorithm recognized this fact and (efficiently) sorted boxes, $\boldsymbol{X}$, on increasing $\underline{f(\boldsymbol{X})}$ using a priority search queue, Q.

Superlinear Speedup

Speedup is defined as $S_m = T_1/T_m$, where T_1 is the sequential execution time (e. g. SIAGO on one processor) and T_m is the parallel execution time (e. g. PIAGO on m processors). Theoretically, superlinear speedup (i. e. $S_m > p$) of an efficient algorithm is not possible [6]. In practice, however, superlinear speedup has been reported often for B&B algorithms in general and PIAGO algorithms in particular [2,5,7,13,15,18,20,25,26].

One reason why superlinear speedup may be achieved in practice is that the sequential algorithm may be inefficient. Some of the earliest PIAGO implementations reported large superlinear speedups. For example, a superlinear speedup of 170 is reported on 32 nodes in [25]. Using a priority search queue ordered on lowest $\underline{f(\boldsymbol{X})}$ [9], Leclerc [18] reports only sublinear speedup of approximately 1/2 for the same problem.

In [2] a theorem is presented that "clearly indicates that no substantial superlinear speedup is possible, assuming that the *best-first* strategy is used". Here, best-first strategy refers to the same lowest $\underline{f(\boldsymbol{X})}$ ordering of boxes on the search queue used by Leclerc. Note, the theorem does not claim that the best-first strategy is best strategy to use. It only claims that if the best-first strategy is used for both the the sequential version and the parallel version, then superlinear speedup is not expected.

In fact, most of the superlinear speedups that have been reported recently are just slightly above linear. This can be explained by the combination of one or more of the following factors:

- high memory utilization in the sequential case may result in poor caching and possible paging thus extending execution time

- non-deterministic timing anomalies (race conditions) that occur in parallel executions may not have been "smoothed out" by averaging the results of many execution runs
- the partial breadth-first search that parallelization introduces into the computation may indeed accelerate finding global solutions for some problems.

Methods

Following is a survey of PIAGO methods that have evolved over the past 15 years. Performance comparisons of the the various methods based on execution times are difficult to make. For example, differences in implementation hardware, the problems being solved, and IA software used, will affect execution times.

Instead, most articles report speedup as a measure of the efficiency of the parallel algorithm. However, speedup is also dependent on the several factors including box ordering on the search queue, memory utilization, non-deterministic parallel "race condition" effects, implementation hardware, and the specific problem being solved (see Sect. "Superlinear Speedup"). For these reasons, no effort is made to compare the various algorithms with regard to reported efficiency.

Various acceleration techniques or general improvements to SIAGO are not considered. It is assumed that such improvements would benefit most if not all of the methods surveyed.

Finally, no discussion of global termination detection is made. Although this is an interesting topic [28], the methods (both centralized and distributed) are few, well analyzed, and not affected by the particular nature of B&B IA global optimization algorithms. Moreover, the contribution of global termination detection to the total execution time for hard problems is negligible.

The key component differentiating the various PIAGO algorithms is workload management (see Sect. "Workload Management"). Each considered PIAGO algorithm is categorized into one of distributed, centralized, or hybrid categories. A discussion of the workload management scheme is given along with relevant comments concerning scalability, code complexity, and communication costs.

Distributed Approaches

As mentioned in "Distributed Management", distributed workload approaches are generally scalable. Asynchronous non-blocking communication is more efficient, but also more difficult to program. By either interleaving messaging probing (e. g. MPI_Iprobe) within the main computation loop (see Algorithm 1) or dedicating a separate thread to the task of receiving messages, one can use efficient non-blocking communication in the approaches that follow. No further discussion of synchronous versus asynchronous communication is made.

Let $P_0, P_1, \ldots, P_{m-1}$ represent m processors on a parallel machine. Let $\widetilde{W}_0, \widetilde{W}_1, \ldots, \widetilde{W}_{m-1}$ represent recorded workload state information for each processor. A given processor can query the (approximate) current workload queue size or minimum $\underline{f(\boldsymbol{X})}$ on processor j using $\widetilde{W}_j$.Qsize or $\widetilde{W}_j$.Qlbf (the lower bound on the function over all boxes in the queue), respectively.

The Leclerc Approach This approach [18,19] is fully distributed and utilizes the best-first queuing strategy. It uses the load balancing procedure listed in Procedure *loadbalance* with the function *WorkloadBalanced* returning TRUE when the processor's Q is empty (i. e. no work). This is a simple demand-driven load balancing scheme. The lowest $\underline{f(\boldsymbol{X})}$ value for boxes on each processor's local Q are broadcast at regular intervals to all processors and recorded in $\widetilde{W}$.

```
// Load balance on processor, P_i
E = {i} if not WorkloadBalanced(Q, W̃) then
    repeat
        P_b = min(W̃_i.Qlbf), i ∉ E  Request fraction
        of boxes from P_b  if no boxes received then
            E = E ∪ {b}
        end
    until boxes received
end
```

Interval Analysis: Parallel Methods for Global Optimization, Procedure `loadbalance(i)`

The Hu, Kearfott, Xu, and Yang Approach This approach [13] is similar to the one used by Leclerc, but

with an initial static assignment of one box to each processor on startup. One box is requested, instead of a fraction of boxes, when a processor becomes idle. From the paper it is unclear to which processor(s) a request for workload is made. It is also unclear whether workload state information, $\widetilde{W}$, is maintained.

The Caprani and Madsen Approach This simple, yet promising approach [3] uses static load balancing rather than dynamic load balancing. First, a "good" U^* is computed on one processor. Next, a "sufficient number" (e. g. $10m$) of sub-boxes are generated using SIAGO and placed into m sets of "approximately equal difficulty". The m sets along with U^* are statically distributed onto m processors. SIAGO is performed on each processor with no communication.

The Eriksson and Lindström Approach Here [5], load balancing is considered on a specialized parallel computer—an Intel iPSC/2 hypercube. No workload state information, $\widetilde{W}$, is maintained. In order to load balance qualitatively as well as quantitatively, a hybrid of two load balancing strategies is used: *receiver-initiated* and *sender-initiated*.

The receiver-initiated load balancer is conceptually similar to Procedure *loadbalance*. But, rather than a selection based on min($\widetilde{W}_i$.Qlbf), an un-prioritized linear search (for a non-idle node) along a ring is performed. This ensures no processor stays idle for very long.

The sender-initiated load balancer seeks to balance qualitatively. Here, the "best" box on the Q (i. e. the one with the lowest $\underline{f(\boldsymbol{X})}$) is "pushed" to a random processor each time G boxes have been split. The frequency of a push operation, G, on a particular processor is decremented by one when the pushed box gets placed at the front of the Q of the randomly selected processor; otherwise, G is incremented by one. The net effect is that if truly "good" boxes are being pushed, then they will continue to be pushed at a high frequency; otherwise, pushes will occur less often.

The Gau and Stadther Approach Here [7] two fundamental algorithms are proposed. First is the synchronous work stealing (SWS) approach. This approach is very similar to the approaches by Hu, Kearfott, Xu, and Yang, and Leclerc. The difference is that largest Q length is used instead of lowest $\underline{f(\boldsymbol{X})}$.

Next, an asynchronous diffusive load balancing (ADLB) scheme is proposed. A group of "nearest neighbors" is defined. Neighbors exchange workload information. Then, boxes are either "pushed" or "pulled" to/from neighbors depending on workload distribution inequities as determined by each processor. The mechanism is analogous to heat or mass diffusion.

In theory this approach should be able to handle qualitative issues regarding workload. However, this is not considered in the paper.

The Martínez, Casado, Alvarez, and García Approach This recent approach [20] is most novel for it's implementation language—Charm++. The execution model of Charm++ is message-driven (i. e. the arrival of messages "triggers" associated computations). This model is similar to a data flow machine.

Essentially a process (chare) runs on each processor. This process responds to (is triggered by) messages to either process a box, *Process-Box*, or update U^*, *update-U**. A *Process-Box* message can either:

- reject the box with no messages generated
- subdivide the box generating two *Process-Box* messages sent to two random processors
- send a message to the main chare to enqueue a new solution.

Messages can be prioritized so that *update-U** messages take precedence over *Process-Box* messages. This should help improve the efficiency of the parallel algorithm. Also, *Process-Box* messages can be prioritized on lowest $\underline{f(\boldsymbol{X})}$ in order to load balance qualitatively.

Data flow solutions are truly elegant. Load balancing quantitively and qualitatively is achieved via randomness and built-in message prioritization.

Centralized Approaches

The Henriksen and Madsen Approach An early implementation of a PIAGO algorithm using a centralized workload manager is that of Henriksen and Madsen [11]. A master node maintains the priority workload queue, Q, and schedules work to each slave processor. When a slave node splits a box, it keeps only one box and sends the remainder back to the master, to be inserted into Q. U^* is also maintained at the master.

The algorithm is load balanced (both quantitively and qualitatively) and has the advantage of total order-

ing of boxes. However, its weakness is poor scalability. The master quickly becomes a memory bottleneck and communication "hotspot" on parallel machines with 32 or more nodes [2].

To be fair, however, such an algorithm is better suited to shared memory multiprocessors, and in particular, multi-core processors (e. g. AMD Opteron, Intel Core 2 Duo). Though multi-core processors don't offer as great an opportunity for massive parallelism (usually 16 cores or less on a processor), they are ubiquitous today and inexpensive. Therefore, one can envision Henriksen and Madsen's approach being used on a distributed memory multicomputer in which the individual processors are multi-core. A more scalable algorithm would be used on the multicomputer architecture as a whole, but the centralized approach could be used as a multithreaded PIAGO application running on each multi-core processor.

The advantage of this hierarchical workload management approach is a better approximation to the best-first strategy. In addition, more efficiency would be obtained with the centralized implementation on each multi-core processor, since shared memory is faster than message passing. The main disadvantage would be code complexity.

Hybrid Approaches

As was mentioned in Sect. "Centralized Approaches", pure centralized approaches, though offering total ordering of the workload Q, are not scalable. Hybrid approaches are theoretically not scalable either. However, some of the scalability issues are mitigated by leveraging local memory on worker processors. Three hybrid approaches are considered.

The Berner Approach Here [2], a master node handles requests from idle processors. A dynamically adjusted variable, *max*, is used to "throttle" the workload on the worker processors as well as help ensure the master does not run out of work. Processors with more than *max* boxes on the local Q will send "some of them" to the master.

The Ibraev Approach This approach [15] is a variation on the Berner approach, with the master (leader) node continually "floating" to the processor that discovers a better $\underline{f(\boldsymbol{X})}$. Workers discovering a possibly lower $\underline{f(\boldsymbol{X})}$ "challenge" the current leader. The current leader makes a determination as to the next leader and broadcasts the index of the new leader along with the improved $\underline{f(\boldsymbol{X})}$ to all processors.

In this approach, no effort is made to approximate a totally ordered global Q. Rather, the approach seeks only to ensure that work requests are made to the processor with the best quality boxes.

The Tapamo and Frommer Approach Tapamo and Frommer [26] propose a variation of the Berner approach which allows non-idle processors to serve requests. The master node keeps track of the lengths of each processor's local Q. When one or more processors become idle, the master then instructs non-idle processors (in decreasing order of Q length) to concurrently satisfy requests from idle processors.

Workload state information (i. e. local Q sizes) must be sent to the master at some frequency. The same issues regarding this frequency are present in the various distributed approaches (See Sect. "Distributed Management"). Delay is introduced due to the indirection of requests having to "pass" through the master node.

Conclusions

The pure centralized workload management scheme is clearly impractical to implement on large distributed memory multicomputers due to issues of scalability. Fully distributed algorithms are scalable but some would question their efficiency based on concerns that the following phenomena may significantly impact performance:

- frequent broadcasting of workload state
- repeated retransmissions for workload due to idle P_b in Procedure *loadbalance*
- a global best-fit exploration of boxes is not being performed (i. e. perhaps the best quality boxes are not being evenly distributed).

Hybrid methods were apparently developed to resolve one or more of the perceived deficiencies of distributed methods and the scalability problem of the pure centralized method. Though hybrid methods have reduced bottleneck potential, they still suffer from poor scalability.

A closer examination of the apparent deficiencies of the distributed methods is worth making. Efficient (up

to practically constant-time) broadcast primitives have been implemented [27,12]. Thus, it would seem, that frequent broadcasting of workload state may not significantly effect performance. Moreover, the frequency of broadcast can easily be throttled if required.

A good estimate of the workload state on each processor for large problems is reasonable to expect. Thus, a high probability exists that the first or possibly second request will fall on a non-idle processor with "good" work. Retransmissions may in fact be few.

Finally, a global best-fit exploration of boxes is not being performed using distributed schemes. However, such a totally ordered exploration is not being done using any of the hybrid methods either. An argument claiming hybrid methods yield better approximations to a global ordering is difficult to make.

A complete and fair assessment of the various PIAGO algorithms (in particular distributed methods versus hybrid methods) should cover a wide range of difficult global optimization test problems. The same efficient SIAGO algorithm (e. g. using best-first ordering) should be used in each and a common hardware platform should be utilized. Furthermore, multiple runs of each test case should be run and averaged in order to "smooth out" non-deterministic parallel computation effects. To date no such comprehensive analysis has been performed.

See also

- ► Interval Analysis: Intermediate Terms
- ► Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ► Interval Analysis: Systems of Nonlinear Equations
- ► Interval Analysis: Unconstrained and Constrained Optimization
- ► Interval Analysis: Verifying Feasibility
- ► Interval Global Optimization
- ► Interval Newton Methods

References

1. Bader DA, Hart WE, Phillips CA (2004) Parallel Algorithm Design for Branch and Bound. In: Greenberg H (ed) Tutorials on Emerging Methodologies and Applications in Operations Research, pp 1–44
2. Berner S (1996) Parallel methods for verified global optimization practice and theory. J Global Optim 9(1):1–22
3. Caprani O, Madsen K (1998) An Almost Embarrasingly Parallel Interval Global Optimization Method
4. Crainic T, Cun BL, Roucairol C (2006) Parallel Branch and Bound Algorithms. In: Talbi E-G (ed) Parallel Combinatorial Optimization, Chap 1. Willey, New York, pp 1–28
5. Eriksson J, Lindstrom P (1995) A parallel interval method implementation for global optimization using dynamic load balancing. Reliab Comput 1(1):77–92
6. Faber V, Lubeck O, White A (1986) Superlinear Speedup of an Efficient Sequential Algorithm is Not Possible. Parallel Comput 3:259–260
7. Gau C, Stadtherr M (2001) Parallel Interval-Newton Using Message Passing: Dynamic Load Balancing Strategies. In: Proceedings of the 2001 ACM/IEEE conference on Supercomputing, pp 23–23
8. Gendron B, Crainic TG (1994) Parallel Branch-And-Bound Algorithms: Survey and Synthesis. Operat Res 42(6):1042–1066
9. Hansen ER (1992) Global Optimization Using Interval Analysis. Marcel Dekker, New York
10. Hansen ER, Walster GW (2004) Global Optimization Using Interval Analysis, 2nd edn. CRC Press, Boca Raton
11. Henriksen T, Madsen K (1992) Use of a depth-first strategy in parallel global optimization. Technical Report 92-10, Institute for Numerical Analysis, Technical University of Denmark, Lyngby
12. Hoefler T, Siebert C, Rehm W (2007) A practically constant-time MPI Broadcast Algorithm for large-scale InfiniBand Clusters with Multicast. In: Parallel and Distributed Processing Symposium, 2007, pp 1–8
13. Hu C, Kearfott B, Xu S, Yang X (2000) A parallel software package for nonlinear global optimization
14. Hwang K (1993) Advanced Computer Architecture. McGraw-Hill, New York
15. Ibraev S (2002) A new parallel method for verified global optimization. Proc Appl Math Mech PAMM 1(1):470–471
16. Kearfott RB (1996) A Review of Techniques in the Verified Solution of Constrained Global Optimization Problems. In: Kearfott RB, Kreinovich V (eds) Applications of Interval Computations. Kluwer, Dordrecht, pp 23–59
17. Kearfott RB (1996) Rigorous Global Search: Continuous Problems. Kluwer, Dordrecht
18. Leclerc A (1993) Parallel interval global optimization in C++. Interval Comput 3:148–163
19. Leclerc A (2001) Interval Analysis: Parallel Methods for Global Optimization. Encycl Optim 3:23–30
20. Martinez J, Casado L, Alvarez J, Garcia I (2006) Interval Parallel Global Optimization with Charm++. In: Dongarra J, Madsen K, Wasniewski J (eds) PARA'04 State-of-the-Art in Scientific Computing, pp 161–168
21. Moore RE (1962) Interval Arithmetic and Automatic Error Analysis in Digital Computing. Applied mathematics and statistics laboratories technical report, Stanford University
22. Moore RE (1966) Interval Analysis. Prentice-Hall, Englewood Cliffs

23. Moore RE (1979) Methods and Applications of Interval Analysis, SIAM Studies in Applied Mathematics. SIAM, Philadelphia
24. Moore RE (1988) Reliability in Computing. Academic Press. See especially the papers by Hansen E 289–308, Walster GW 309–324, Ratschek H 325–340, and Lodwick WA 341–354
25. Moore RE, Hansen ER, Leclerc AP (1992) Recent Advances in Global Optimization. Princeton University Press, Princeton, pp 321–342
26. Tapamo H, Frommer A (2007) Two acceleration mechanisms in verified global optimization. J Comput Appl Math 199(2):390–396
27. Tinetti F, Barbieri A (2003) An efficient implementation for broadcasting data in parallel applications over Ethernet clusters. Advanced Information Networking and Applications, pp 593–596
28. Topor RW (1984) Termination Detection for Distributed Computations. Inform Process Lett 18(1):33–36
29. Zilinskas J (2005) A Package for Development of Algorithms for Global Optimization. In: Proceedings of the 10th International Conference MMA2005&CMAM2, pp 185–190

Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods

TIBOR CSENDES
University of Szeged, Szeged, Hungary

MSC2000: 65K05, 90C30

Article Outline

Keywords and Phrases
Subdivision Directions
Properties of Direction Selection Rules
  Theoretical Properties
  Numerical Properties
References

Keywords and Phrases

Branch-and-bound; Interval arithmetic; Optimization; Subdivision direction

The selection of subdivision direction is one of the points where the efficiency of the basic ► branch-and-bound algorithm for unconstrained global optimization can be improved (see ► Interval analysis: unconstrained and constrained optimization). The traditional approach is to choose that direction for subdivision in which the actual box has the largest width. If the inclusion function $\phi(x)$ is the only available information about the problem

$$\min_{x \in \boldsymbol{x}_0} \phi(x) \ ,$$

then it is usually the best possible choice. If, however, other information like the inclusion of the gradient ($\nabla\phi$), or even the inclusion of the Hessian ($\boldsymbol{H}$) is calculated, then a better decision can be made.

Subdivision Directions

All the rules select a direction with a merit function:

$$k := \arg\max_{i=1}^{n} D(i), \tag{1}$$

where $D(i)$ is determined by the given rule. If many such optimal k indices exist then the algorithm can chose the smallest one, or it can select an optimal direction randomly.

Rule A. *The first rule was the interval-width oriented rule. This rule chooses the coordinate direction with*

$$D(i) := w(\boldsymbol{x}_i). \tag{2}$$

This rule is justified by the idea that, if the original interval is subdivided in a uniform way, then the width of the actual subintervals goes to zero most rapidly.

The algorithm with Rule A is convergent both with and without the monotonicity test [8]. *This rule allows a relatively simple analysis of the convergence speed (as in* [8], *Chapter 3, Theorem 6).*

Rule B. *E. Hansen described another rule (initiated by G. W. Walster* [5]*). The direct aim of this heuristic direction selection rule is to find the component for which* $W_i = \max_{t \in \boldsymbol{x}_i} \phi(m_1, \ldots, m_{i-1}, t, m_{i+1}, \ldots, m_n) - \min_{t \in \boldsymbol{x}_i} \phi(m_1, \ldots, m_{i-1}, t, m_{i+1}, \ldots, m_n)$ *is the largest (where* $m_i = (\underline{\boldsymbol{x}}_i + \overline{\boldsymbol{x}}_i)/2$ *is the midpoint of the interval* $\boldsymbol{x}_i$*). The factor* W_i*, that should reflect how much* φ *varies as* x_i *varies over* $\boldsymbol{x}_i$*, is then approximated by* $w(\nabla\phi_i(\boldsymbol{x}))w(\boldsymbol{x}_i)$ *(where* $\nabla\phi_i(\boldsymbol{x})$ *denotes the ith component of* $\nabla\phi(\boldsymbol{x})$*). The latter is not an upper bound for*

W_i (cf. [5] page 131 and Example 2 in Section 3 of [4]), yet it can be useful as a merit function.

Rule B selects the coordinate direction, for which (1) holds with

$$D(i) := w(\nabla\phi_i(\boldsymbol{x}))w(\boldsymbol{x}_i). \tag{3}$$

It should be noted that the basic bisection algorithm represents only one way in which Rule B was applied in [5]. There the subdivision was, e. g., also carried out for many directions in a single iteration step.

Rule C. *The next rule was defined by Ratz [9]. The underlying idea was to minimize the width of the inclusion: $w(\boldsymbol{\phi}(\boldsymbol{x})) = w(\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\phi}(m(\boldsymbol{x}))) \approx w(\nabla\boldsymbol{\phi}(\boldsymbol{x})(\boldsymbol{x} - m(\boldsymbol{x}))) = \sum_{i=1}^{n} w(\nabla\boldsymbol{\phi}_i(\boldsymbol{x})(\boldsymbol{x}_i - m(\boldsymbol{x}_i)))$. Obviously, that component is to be chosen for which the term $w(\nabla\phi_i(\boldsymbol{x})(\boldsymbol{x}_i - m(\boldsymbol{x}_i)))$ is the largest. Thus, Rule C can also be formulated with (1) and*

$$D(i) := w(\nabla\boldsymbol{\phi}_i(\boldsymbol{x})(\boldsymbol{x}_i - m(\boldsymbol{x}_i))). \tag{4}$$

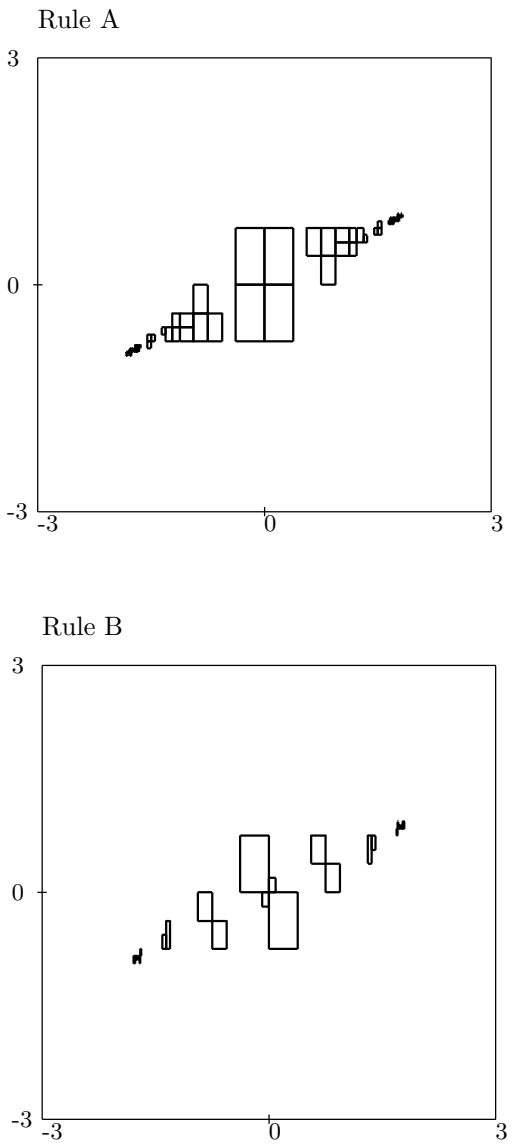
The important difference between (3) and (4) is that in Rule C the width of the multiplied intervals is maximized, not the multiplied widths of the respective intervals (and these are in general not equal). After a short calculation, the right-hand side of (4) can be written as $\max\{|\min \nabla\boldsymbol{\phi}_i(\boldsymbol{x})|, |\max \nabla\boldsymbol{\phi}_i(\boldsymbol{x})|\}w(\boldsymbol{x}_i)$. This corresponds to the maximum smear defined by R.B. Kearfott (used as a direction selection merit function solving systems of nonlinear equations [6,7]) for the case $\phi : \mathbb{R}^n \to \mathbb{R}$. It is easy to see that the Rules B and C give the same merit function value if and only if either $\underline{\nabla\boldsymbol{\phi}}_i(\boldsymbol{x}) = 0$ or $\overline{\nabla\boldsymbol{\phi}}_i(\boldsymbol{x}) = 0$.

Rule D. *The fourth rule, Rule D is derivative-free like Rule A, and reflects the machine representation of the inclusion function $\boldsymbol{\phi}(\boldsymbol{x})$ (see [5]). It is again defined by (1) and by*

$$D(i) := \begin{cases} w(\boldsymbol{x}_i) & \text{if } 0 \in \boldsymbol{x}_i, \\ w(\boldsymbol{x}_i)/ < \boldsymbol{x}_i > & \text{otherwise}, \end{cases} \tag{5}$$

where $< \boldsymbol{x} >$ is the magnitude of the interval $\boldsymbol{x}$: $< \boldsymbol{x} > := \min_{x\in\boldsymbol{x}} |x|$.

This rule may decrease the excess width $w(\boldsymbol{\phi}(\boldsymbol{x})) - w(\phi^u(\boldsymbol{x}))$ of the inclusion function (where $\phi^u(\boldsymbol{x})$ is the range of φ on $\boldsymbol{x}$) that is caused in part by the floating point computer representation of real numbers. Consider the case when the component widths are of similar order, and the absolute value of one component is dominant. The subdivision of the latter component may result in a worse inclusion, since the representable numbers are sparser in this direction.



Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods, Figure 1
Remaining subintervals after 250 iteration steps of the model algorithm with the direction selection Rules A, and B for the Three-Hump-Camel-Back problem [3]

Rule E. *Similar to Rule C, the underlying idea of Rule E is to minimize the width of the inclusion, but this time based on second order information (suggested by Ratz [10]):*

$$D(i) := w((\boldsymbol{x}_i - m(\boldsymbol{x}_i))(\nabla\phi_i(m(\boldsymbol{x})) + \frac{1}{2}\sum_{j=1}^{n}(\boldsymbol{H}_{ij}(\boldsymbol{x})(\boldsymbol{x}_i - m(\boldsymbol{x}_i))) \ . \quad (6)$$

Many interval optimization codes use ► automatic differentiation to produce the gradient and Hessian values. For such an implementation the subdivision selection Rule E requires not much overhead.

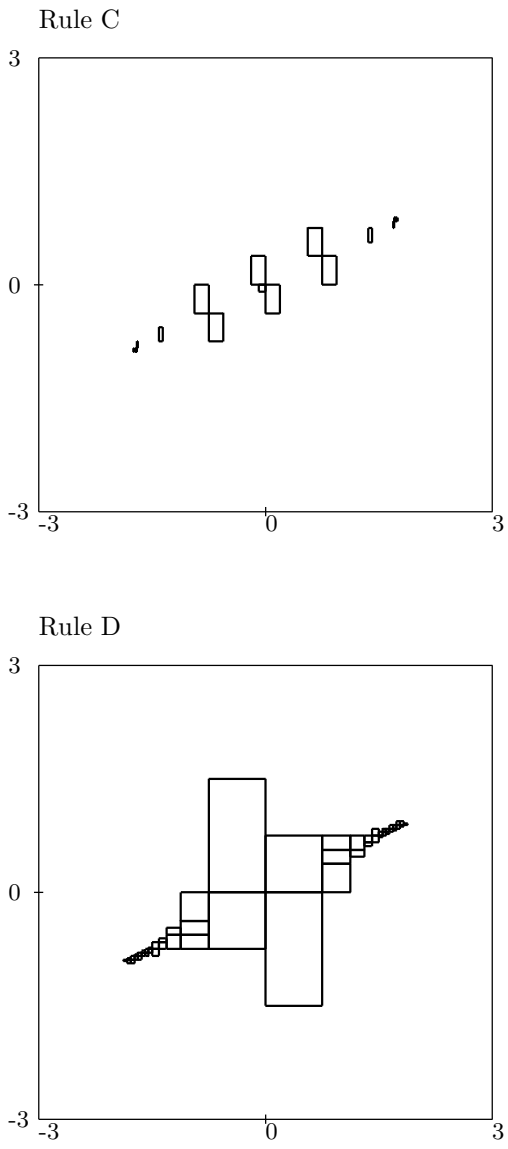
Properties of Direction Selection Rules

Both the theoretical and numerical properties of subdivision direction selection rules have been studied extensively [1,3,4,10,11]. The exact definitions, theorems and details of numerical comparison tests can be found in these papers. Denote the global minimum value by ϕ^*.

Theoretical Properties

In [4] the property of balanced direction selection has been defined. A subdivision direction selection rule is balanced basically if the B&B algorithm with this direction selection rule will not be unfair with any coordinate direction: each direction will be selected an infinite number of times in each infinite subdivision sequence of the leading boxes generated by the optimization algorithm. A global minimizer point $x' \in \boldsymbol{x}^0$ is called hidden global minimizer point, if there exists a subbox $\boldsymbol{x}' \subseteq \boldsymbol{x}^0$ with positive volume for which $x' \in \boldsymbol{x}'$ and $\underline{\boldsymbol{\phi}}(\boldsymbol{x}') = \phi^*$ while there exists an other global minimizer point x'' of the same problem such that $\underline{\boldsymbol{\phi}}(\boldsymbol{x}'') < \phi^*$ holds for each subbox $\boldsymbol{x}'' \subseteq \boldsymbol{x}^0$ with positive volume that contains x'' [11]. Now the following statements can be made:

1. The basic branch-and-bound algorithm converges in the sense that $\lim_{s\to\infty} w(\boldsymbol{x}^s) = 0$ if and only if the interval subdivision selection rule is balanced [4] (where $\boldsymbol{x}^s$ is the leading box of the algorithm in the iteration step number s).
2. Assume that the subdivision direction selection rule is balanced. Then the basic B&B algorithm converges to global minimizer points in the sense that $\lim_{s\to\infty} \underline{\boldsymbol{\phi}}(\boldsymbol{x}^s) = \phi^*$, the set of accumulation points A of the leading box sequence is not empty, and A contains only global minimizer points.
3. Assume that the optimization algorithm converges for a given problem in the sense that $\lim_{s\to\infty} \underline{\boldsymbol{\phi}}(\boldsymbol{x}^s) = \phi^*$. Then either the algorithm proceeds on the problem as one with a balanced



Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods, Figure 2
Remaining subintervals after 250 iteration steps of the model algorithm with the direction selection Rules C, and D for the Three-Hump-Camel-Back problem [3]

direction-selection rule, or there exists a box $\boldsymbol{y}$ such that $\phi(x) = \phi^*$ for all $x \in \boldsymbol{y}$, and $w(\boldsymbol{y}_i) > 0$ $(i = 1, 2, \ldots, n)$ for all coordinate directions that are selected only a finite number of times.

4. The subdivision selection Rules A and D are balanced, and thus the related algorithms converge to global minimizer points.
5. Either the subdivision selection Rules B and C choose each direction an infinite number of times (they behave as balanced), or the related algorithms converge to a positive width subinterval of the search region $\boldsymbol{x_0}$ that contains only global minimizer points.
6. Sonja Berner proved that the basic algorithm is convergent with Rule E in the sense of $\lim_{s\to\infty} \boldsymbol{\phi}(\boldsymbol{x}^s) = \phi^*$, if an additional condition holds for the inclusion function [1].
7. If the branch-and-bound algorithm with any of the direction selection Rules A–E converges to a global minimizer point, then it converges to all non-hidden global minimizer points [11].

Numerical Properties

The numerical comparison tests were carried out on a wide set of test problems and in several computational environments. The set of numerical test problems contained the standard global optimization test problems [3,4], the set of problems studied in [5], and also some additional ones [10,11]. The computing environments include IBM RISC 6000-580 and HP 9000/730 workstations and Pentium PC-s. The programs were coded in FORTRAN-90, PASCAL-XSC, and also in C++. The tests were carried out both with simple natural interval extension and with more sophisticated inclusion functions involving centered forms. The derivatives were handcoded in some test [4], while they were generated by automatic differentiation in the others [3,10,11]. The range of the investigated algorithms included simple B&B procedures and also optimization codes with many acceleration devices (like the ► interval Newton method).

The conclusions were essentially the same: the Rules B, C, and E had similar, substantial efficiency improvements against Rules A and D, and these improvements were the greater the more difficult the solved problem was. The average performance of Rule D was the worst. Rule C was usually the best, closely followed by Rule B and E. It seems that the use of Rule E is justified only if the second derivatives are calculated also for other purposes. The numerical results were diverse, thus if the user has a characteristic problem set, then it is worth to test all the subdivision direction selection rules to find the most fitting one.

A computationally intensive numerical study [2] has proven that the most efficient subdivision direction selection rules are not those that minimize the width of the objective function inclusions for the result subintervals (which was the common belief), but those that maximize the lower bound of the worse subinterval obtained or minimize the width of the intersection of the result subintervals. The decisions of these a posteriori rules coincide the most with the a priori Rules B, C, and E. These findings confirm the earlier mentioned numerical efficiency results.

References

1. Berner S (1996) New results on verified global optimization. Computing 57:323–343
2. Csendes T, Klatte R, Ratz D (2000) A Posteriori Direction Selection Rules for Interval Optimization Methods. CEJOR 8:225–236
3. Csendes T, Ratz D (1996) A review of subdivision direction selection in interval methods for global optimization. ZAMM 76:319–322
4. Csendes T, Ratz D (1997) Subdivision direction selection in interval methods for global optimization. SIAM J Numer Anal 34:922–938
5. Hansen E (1992) Global optimization using interval analysis. Marcel Decker, New York
6. Kearfott RB (1996) Rigorous global search: continuous problems. Kluwer, Dordrecht
7. Kearfott RB, Novoa M (1990) INTBIS, a Portable Interval Newton/Bisection Package. ACM T Mathemat Softw 16:152–157
8. Ratschek H, Rokne J (1988) New Computer Methods for Global Optimization. Ellis Horwood, Chichester
9. Ratz D (1992) Automatische Ergebnisverifikation bei globalen Optimierungsproblemen. Dissertation, Universität Karlsruhe
10. Ratz D (1996) On Branching Rules in Second-Order Branch-and-Bound Methods for Global Optimization. In: Alefeld G, Frommer A, Lang B (eds) Scientific Computing and Validated Numerics. Akademie-Verlag, Berlin, pp 221–227
11. Ratz D, Csendes T (1995) On the selection of Subdivision Directions in Interval Branch-and-Bound Methods for Global Optimization. J Glob Optim 7:183–207

Interval Analysis: Systems of Nonlinear Equations

RAMON E. MOORE
Worthington, USA

MSC2000: 65G20, 65G30, 65G40

Article Outline

Keywords
Numerical Example
See also
References

Keywords

Interval Newton operator; Nonlinear equations

A system of *nonlinear equations* can be represented in vector form as $f(x) = 0$, where the components are $f_i(x) = f_i(x_1, \ldots, x_n) = 0$, $i = 1, \ldots, n$.

Sometimes we seek one solution; sometimes we are interested in locating all solutions.

A naive interval approach can be used to search a *box* (an interval vector) V for solutions. Using repeated bisections in various coordinate directions, we can chisel off parts of V that cannot contain a solution. That is, if $f(W)$ does not contain the zero vector for some W in V, then we can delete W as containing no solutions to $f(x) = 0$. The remaining parts of V contain all the solutions, if any, that were in the initial V.

For differentiable systems, there are much more efficient methods for finding a solution or the set of all solutions. Even so, the naive approach does have its uses. In practice it often pays to combine a number of techniques.

One approach to solving $f(x) = 0$ is to formulate an equivalent fixed-point problem, and use iterative methods to solve it. We can define

$$g(x) = x + Yf(x)$$

for any linear mapping Y. If Y is nonsingular, then $f(x) = 0$ is equivalent to $g(x) = x$.

If g is continuous and S is a compact, convex subset of $\mathbf{R}^n$, and g maps S into itself, then g has a fixed point in S and so $f(x) = 0$ has a solution in S.

An interval vector V is a compact, convex set, so $g(V) \subset V$ implies $f(x) = 0$ for at least one point x in V.

Classical iterative methods consider sequences of points generated by

$$x^{(k+1)} = g(x^{(k)})$$

starting from some initial point $x^{(0)}$.

If we denote the Jacobian matrix for the system by $f'(x)$, then choosing $Y = -f'(x)^{-1}$, we will have Newton's method. If we take Y as an approximation to $-f'(x)^{-1}$, then we obtain a Newton-like method.

Interval versions of Newton's method, however, also involve *intersections*, as we will see.

An *interval Newton method* for finite systems of nonlinear equations was introduced by R.E. Moore [11,12]. Subsequently, many improvements have been made, e. g., [4,6,7,8,10,13,16,17,18].

In order to explain as clearly as possible, consider the one-dimensional case. We have the mean value theorem for continuously differentiable f:

$$f(x) = f(x^{(0)}) + f'(\xi)(x - x^{(0)})$$

for some ξ between $x^{(0)}$ and x.

We have $f(x) = 0$ if x satisfies

$$x = x^{(0)} - [f'(\xi)]^{-1} f(x^{(0)}).$$

Now the ordinary Newton method replaces the unknown ξ by $x^{(0)}$.

The initial idea was to use an interval for ξ and use interval computation throughout the iterations. If we start with an interval, say $X^{(0)}$, that contains $x^{(0)}$ and happens to also contain a solution, say x, of $f(x) = 0$, then $X^{(0)}$ also contains ξ and therefore x is contained in

$$N(X^{(0)}) = x^{(0)} - [f'(X^{(0)})]^{-1} f(x^{(0)})$$

(N for Newton), where $f'(X^{(0)}) \subset \{f'(x) : x \in X^{(0)}\}$. The first idea was to iterate $X^{(k+1)} = N(X^{(k)})$, but this turns out not to converge in all cases.

Then the following idea was proposed, [12]. Since a solution x in $X^{(0)}$ is also in $N(X^{(0)})$, it follows that x is also contained in the intersection: $X^{(0)} \cap N(X^{(0)})$.

Therefore, we iterate

$$X^{(k+1)} = X^{(k)} \cap N(X^{(k)})$$

with

$$N(x^{(k)}) = y^{(k)} - [f'(X^{(k)})]^{-1} f(y^{(k)}),$$

choosing $y^{(k)}$ in $X^{(k)}$, say the midpoint of $X^{(k)}$.

With this modification, the interval Newton method does what we want, as will be explained. From the above arguments we have proved that for an interval X,

1) if $N(X) \cap X$ is empty, then there is no solution in X.

If we divide by an interval containing zero, we may obtain one, or the union of two, semi-infinite intervals for $N(X)$. The intersection with the finite interval X in the interval Newton method, reduces the result to a finite interval, or the union of two finite intervals, or the empty set.

During the iterations, if $X^{(k)}$ turns out to be the union of two intervals, we put one on a list and proceed to further iterate with the other one. This idea was first presented in E.R. Hansen [6].

We can also prove that [5,6]:

2) if $N(X) \subset X$, then there exists a unique solution in $N(X)$.

The existence follows for the compact, convex interval X, and from the continuity of f'. The uniqueness follows from the boundedness of $N(X) \subset X$. If there were two solutions in $N(X)$, then $f'(y)$ would be zero for some y in X and $N(X)$ would be unbounded.

If f is twice continuously differentiable, then we can also prove the following [16]:

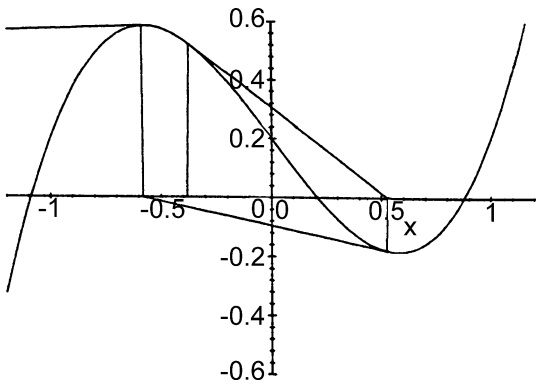
3) if $N(X) \subset X$, then the interval Newton method converges quadratically to the unique solution in X, as does the *ordinary* Newton method from any starting point in X.

'Quadratically' here means there is a constant C such that $w(X^{(k+1)}) < Cw(X^{(k)})^2$, $k = 1, 2, \ldots$, where $w(X)$ denotes the width of an interval X; thus, $w([a, b]) = b - a$.

We illustrate the different behaviors of the ordinary Newton method and the interval Newton method in the following figures. Fig 1 shows that the ordinary Newton method cannot find the middle solution unless we start very close to it.

The first three iterations of the ordinary Newton method

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$



Interval Analysis: Systems of Nonlinear Equations, Figure 1
The ordinary Newton method

for $f(x) = x^3 - x + 0.2$, starting with $x^{(0)} = -0.375$, are shown in Fig 1. The algorithm produces $x^{(1)} = 0.528\ldots$, $x^{(2)} = -0.584\ldots$, $x^{(3)} = -22.569\ldots$. In order to converge to the middle root, we need an initial guess $x^{(0)}$ very close to that root.

The *interval Newton method* finds all three solutions on the starting interval $X^{(0)} = [-1.2, 1.2]$ without difficulty. We choose that starting interval because the roots of a polynomial

$$p(z) = a_n z^n + \cdots + a_1 z + a_0$$

with $a_n \neq 0$ are well-known to lie in the complex disk

$$|z| \leq \max\left\{1, |a_n|^{-1} \sum_{k<n} |a_k|\right\};$$

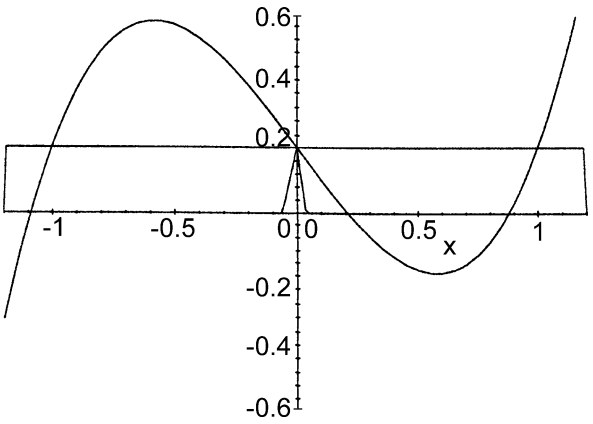
so, for the example $p(x) = x^3 - x + 0.2$, the real roots are known to satisfy

$$-1.2 \leq x \leq 1.2.$$

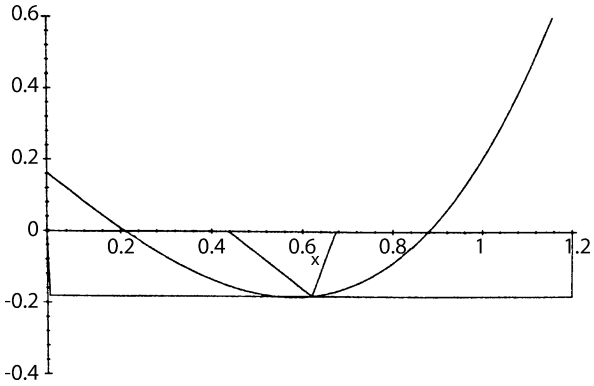
If the intersection $N(X^{(k)}) \cap X^{(k)}$ splits into two intervals, we list one and analyze the other. See Fig 2 and Fig 3

We used the usual recursive algorithm for evaluating the polynomial and its derivative, namely

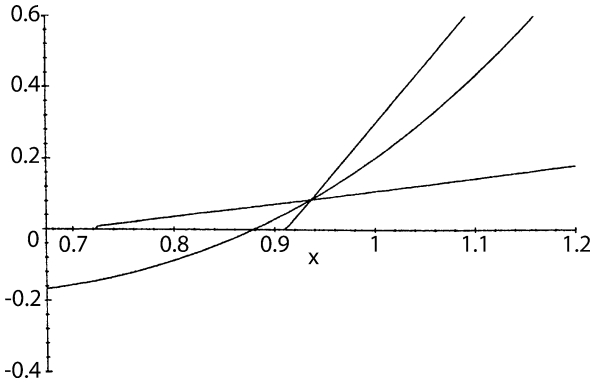
```
p ← a_n
p' ← 0
FOR i = n − 1 TO 0 STEP −1 DO
    p' ← xp' + p
    p ← a_i + xp
END DO
```



Interval Analysis: Systems of Nonlinear Equations, Figure 2
The interval $N([-1.2, 1.2]) \cap [-1.2, 1.2]$ splits into $[-1.2, -0.0602\ldots] \cup [0.0375\ldots, 1.2]$. We list the first and analyze the second. See Fig 3



Interval Analysis: Systems of Nonlinear Equations, Figure 3
The interval being analyzed, $X = [0.0375\ldots, 1.2]$, is shown enlarged for clarity. Again $N(X) \cap X$ splits into two intervals $[0.0375\ldots, 0.436\ldots] \cup [0.6738\ldots, 1.2]$. We list the first and analyze the second. See Fig 4



Interval Analysis: Systems of Nonlinear Equations, Figure 4
We analyze the interval $X = [0.6738\ldots, 1.2]$. This time we have $N(X) \subset X$, because $N(X) = [0.724\ldots, 0.911\ldots]$, so we will have convergence to the unique solution in X

Of course, for the interval Newton method, the evaluations are carried out in interval arithmetic with outward rounding. The real coefficients are entered as degenerate intervals, for example

$$0 \equiv [0, 0], \quad 1 \equiv [1, 1], \quad 0.2 \equiv [0.2, 0.2].$$

Recalling that we began with the initial interval $X^{(0)} = [-1.2, 1.2]$, we find that the midpoint of $X^{(0)}$ is $y^{(0)} = [0, 0]$ and so:

$$\begin{aligned} N(X^{(0)}) &= y^{(0)} - [p'(X^{(0)})]^{-1} p(y^{(0)}) \\ &= [0, 0] - \frac{1}{[-5.32, 3.32]}[0.2, 0.2] \\ &= (-\infty, -0.06024\ldots] \cup [0.03759\ldots, \infty). \end{aligned}$$

When we intersect this union with $X^{(0)}$, we obtain

$$[-1.2, -0.06024\ldots] \cup [0.030759\ldots, 1.2].$$

The calculation of $p'([-1.2, 1.2])$ was as follows:

$$\begin{aligned} &p'([-1.2, 1.2]) \\ &= [3, 3]([-1.2, 1.2][-1.2, 1.2]) - [1, 1] \\ &= [3, 3][-1.44, 1.44] - [1, 1] \\ &= [-4.32, 4.32] - [1, 1] \\ &= [-5.32, 3.32]. \end{aligned}$$

Referring to Fig 2 and Fig 3, we see that the interval Newton method first splits the starting interval $X^{(0)}$ into two subintervals $[-1.2, -0.0602\ldots]$ and $[0.0375\ldots, 1.2]$, then it splits the second one again into two subintervals $[0.0375\ldots, 0.436\ldots]$ and $[0.6738\ldots, 1.2]$.

The intervals

$$[-1.2, 0.0602\ldots] \quad \text{and} \quad [0.0375\ldots, 0.0436\ldots]$$

were listed for later analysis, and the interval $[0.6738\ldots, 1.2]$ was analyzed. With $X^{(0)} = [0.6738\ldots, 1.2]$, the method produced

$$N(X^{(0)}) = [0.724\ldots, 0.911\ldots] \subset X^{(0)},$$

so there is a unique solution in that $X^{(0)}$.

We can prove the following:

4) If $N(X) \subset X$ and we carry out the interval computations with outward rounding after a fixed number of digits (or bits), the iterations

$$X^{(k+1)} = N(X^{(k)}) \cap X^{(k)}, \quad k = 0, 1, \ldots,$$

with $X^{(0)} = X$ form a nested sequence of intervals containing the solution until, after a finite number of iterations we can stop with $X^{(k+1)} = X^{(k)}$.

This follows from the fact that there are only finitely many different machine numbers of a given 'precision', that is with a given finite number of digits (bits). In the example at hand, the final results were as follows. For $X^{(0)} = [0.6738\ldots, 1.2]$, after four iterations:

THERE IS A ROOT IN

$$[0.8788850662, 0.8788850663].$$

The program then removed a new $X^{(0)} = [0.037\ldots, 0.436]$ from the list. It turned out that $N(X^{(0)}) = [0.2007\ldots, 0.213\ldots] \subset X^{(0)}$, so there is a unique solution in the new $X^{(0)}$. After three iterations, we obtained:

THERE IS A ROOT IN

$$[0.2091488484, 0.2091488485].$$

Finally, the program removed the last remaining interval on the list, namely $[-1.2, -0.06\ldots]$, which is then taken as a new $X^{(0)}$. This time the intersection came into play because

$$N(X^{(0)}) = (-\infty, -0.809\ldots] \cup [-0.04\ldots, \infty).$$

The intersection of this union with $X^{(0)} = [-1.2, -0.06\ldots]$ gave the single interval

$$X^{(1)} = N(X^{(0)}) \cap X^{(0)} = [-1.2, -0.809\ldots].$$

After four iterations, we obtained the result:

THERE IS A ROOT IN

$$[-1.0880339147, -1.0880339146].$$

A final message was printed out (which happens when the list becomes empty after the last one taken out is analyzed):

THERE ARE NO MORE ROOTS IN

$$[-1.2, 1.2].$$

The following additional examples were carried out, using a program implementing the interval Newton method just described, in C-XSC [9], and run on an Intel 486 processor.

1) Find the real roots in $[-3, 3]$ of

$$p(x) = x^3 - 1.5201x^2 + 0.770201x - 0.1300755.$$

This polynomial is the expanded form of

$$p(x) = (x - 0.5)(x - 0.51)(x - 0.5101),$$

thus the roots are fairly closely packed.

When we entered $p(x)$ as

$$x*x*x - 1.5201*x*x + 0.770201*x - 0.1300755,$$

the program SES (Single Equation Solver, in C-XSC [9]) produced three roots in the intervals (shown outwardly rounded to six places):

$$[0.510099, 0.510101],$$
$$[0.509999, 0.510001],$$
$$[0.499999, 0.500001].$$

Sadly, when we entered $p(x)$ as

$$x\hat{}3 - 1.5201*x\hat{}2 + 0.770201*x - 0.1300755,$$

the program SES produced a false result, a 'root' in the interval $[0.168885, 0.168886]$.

Unfortunately, one must be cautious when using a *programmed implementation* ('software') even for a method that is guaranteed. It can be very difficult to prove the correctness of a computer program, particularly when it still has bugs, such as the implementation of $x\hat{}3$ in an early version of SES. Hopefully, it has been fixed in later versions. Usually extensive testing of a program before its release will uncover most such bugs. The subject of proving correctness of computer programs is a difficult and active area of research in computer science.

Here are some additional examples run in C-XSC. They were all independently checked with another program written from scratch and run on another computer. This is another way to test a computer implementation of a computational method.

2) Find all the roots in [.01, 1.0] of

$$f(x) = x^2 + \sin\left(\frac{1}{x}\right).$$

The program found 30 roots from right to left, becoming more and more closely packed. The first four digits of the last three roots found were .0109, .0106, .0102.

The last root lies in the interval (shown to 11 digits)

[.01026804972, .01026804973].

The CPU time was 0.16 seconds using C-XSC on a 486 processor.

3) An example suggested by E.A. Galperin, see [14], is

$$f(x) = x^2 + \sin\left(\frac{1}{x^p}\right), \qquad p = 1, 2, 3.$$

We ran the program to find all roots in [0.1, 1.0] for the cases $p = 1, 2, 3$.

Three roots were found right to left for $p = 1$; thirty-one roots for $p = 2$; and 318 roots for $p = 3$. The last root found for $p = 3$ lies in the interval

[0.10003280626, 0.10003280628].

To find all the 318 closely packed roots in the case $p = 3$, the CPU time was 1.5 seconds using C-XSC on a 486 processor.

The method generalizes to n dimensions. Interval Newton methods for n-dimensional nonlinear systems have some remarkable properties; among them are the following:

1) If $N(X) \cap X$ is empty, then there is no solution in X;
2) If $N(X) \subset X$, then there is a unique solution in $N(X)$;
3) If $N(X) \subset X$, then the interval Newton method, with $X^{(0)} = X$, converges quadratically to the unique solution in X, as does the ordinary Newton method from any starting point in X.
4) If $N(X) \subset X$, with outward rounding in the interval computation of $N(X)$, then the interval Newton method converges in a finite number of iterations, because of the intersection, to an interval vector containing the unique solution in X, using the stopping criterion: STOP when $X^{(k+1)} = X^{(k)}$.

The general form of such algorithms is

$$X^{(k+1)} = N(X^{(k)}) \cap X^{(k)},$$

where the *interval Newton operator* N is defined in various ways.

The original way, [12], was

$$N(X^{(k)}) = m(X^{(k)}) - f'(X^{(k)})^{-1} f(m(X^{(k)})),$$

where $m(X^{(k)})$ is the midpoint of $X^{(k)}$. Newer versions (see e. g. [4,7,10,18]) avoid having to find the inverse of the Jacobian matrix $f'(X)$ for an interval vector X.

Krawczyk's variation [10] was to define $N(X)$ as

$$N(X) = y - Yf(y) + \{I - YF'(X)\}Z,$$

where y is the midpoint of the interval vector X, I is the identity matrix, Y is a nonsingular real matrix, such as an approximation to the inverse of $f'(m(X))$, and $Z = X - y$.

In cases 2) and 3) above, the subsequent iterations converge to the solution (quadratically, if Y converges to $[f'(m(X))]^{-1}$ as the width of X goes to zero); and with outward rounding in a computer implementation, the iterations will stop at a finite value of k with the stopping criterion: STOP if $X^{(k+1)} = X^{(k)}$.

For a technique for searching for a *safe starting region* $X^{(0)}$ satisfying 2) from which convergence to a solution is guaranteed, see [13,16]. The technique involves starting with a large initial box and using bisections in a depth-first search in suitable directions to find a sub-box that satisfies property 2) above.

To find enclosures of all solutions in a given initial box, we form a list of 'sub-boxes' (interval vectors) in a way analogous to that explained in the one-dimensional example. When the intersection $N(X) \cap X$ produces two sub-boxes, we can list one and analyze (continue iterations with) the other, or we can list them both and choose some other box on the list to analyze next. Such an interval method lends itself to *parallelization*. We can distribute sub-boxes remaining on the list to processors in a network, and gain a speed-up factor. This is particularly important for applications to *global optimization*; see e. g. [8,15].

Numerical Example

A solution was sought for the following nine-dimensional system obtained from P. Rabinowitz (private communication). The system concerns finding weights and argument spacings for a certain type of multidi-

mensional integration formula

$$
\begin{aligned}
&x_1 + x_3 + x_5 + 2x_7 = 1,\\
&x_1x_2 + x_3x_4 + 2x_5x_6 + 2x_7x_8 + 2x_7x_9 = \frac{2}{3},\\
&x_1(x_2)^2 + x_3(x_4)^2 + 2x_5(x_6)^2\\
&\quad + 2x_7(x_8)^2 + 2x_7(x_9)^2 = \frac{2}{5},\\
&x_1(x_2)^3 + x_3(x_4)^3 + 2x_5(x_6)^3\\
&\quad + 2x_7(x_8)^3 + 2x_7(x_9)^3 = \frac{2}{7},\\
&x_1(x_2)^4 + x_3(x_4)^4 + 2x_5(x_6)^4\\
&\quad + 2x_7(x_8)^4 + 2x_7(x_9)^4 = \frac{2}{9},\\
&x_5(x_6)^2 + 2x_7x_8x_9 = \frac{1}{9},\\
&x_5(x_6)^4 + 2x_7(x_8)^2(x_9)^2 = \frac{1}{25},\\
&x_5(x_6)^3 + x_7x_8(x_9)^2 + x_7(x_8)^2x_9 = \frac{1}{15},\\
&x_5(x_6)^4 + x_7x_8(x_9)^3 + x_7(x_8)^3x_9 = \frac{1}{21}.
\end{aligned}
$$

A solution was sought in the unit 9-cube. We started with an initial box slightly larger than the unit 9-cube in case there was a solution on the boundary. A depth-first search for a safe starting region was carried out and was successful after 168 bisections in a certain sequence of coordinate directions determined by the program as the process proceeded, see [16]. Finally, a solution was quickly bounded in a small box (9-dimensional interval vector here). The reader is invited to try a favorite non-interval nonlinear systems solver to find a solution of this system. Better still, find all the solutions and prove there are no more, as the interval method did.

An alternative approach using interval analysis to solve nonlinear systems is computing the *topological degree* of the mapping f over a box (n-dimensional interval vector). See [1,2].

For access to voluminous literature, available software, current research efforts, conferences, etc. in the area of interval computation, see [21].

See also

- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Point and Interval Taylor Operators
- ▶ Bounding Derivative Ranges
- ▶ Contraction-mapping
- ▶ Global Optimization: Application to Phase Equilibrium Problems
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Interval Analysis: Differential Equations
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Interval Analysis: Intermediate Terms
- ▶ Interval Analysis: Nondifferentiable Problems
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ▶ Interval Analysis: Unconstrained and Constrained Optimization
- ▶ Interval Analysis: Verifying Feasibility
- ▶ Interval Constraints
- ▶ Interval Fixed Point Theory
- ▶ Interval Global Optimization
- ▶ Interval Linear Systems
- ▶ Interval Newton Methods
- ▶ Nonlinear Least Squares: Newton-type Methods
- ▶ Nonlinear Systems of Equations: Application to the Enclosure of All Azeotropes

References

1. Aberth O (1994) Computation of topological degree using interval arithmetic, and applications. Math Comput 62(205):171–178
2. Aberth O (1998) Precise numerical methods using C++. Acad. Press, New York
3. Alefeld G, Frommer A, Lang B (eds) (1996) Scientific computing and validated numerics. Akademie, Berlin
4. Alefeld G, Herzberger J (1983) Introduction to interval computations. Acad. Press, New York
5. Hansen E (ed) (1969) Topics in interval analysis. Oxford Univ. Press, Oxford
6. Hansen E (1978) A globally convergent interval Newton method for computing and bounding real roots. BIT 18:415–424
7. Hansen E (1992) Global optimization using interval analysis. M. Dekker, New York
8. Kearfott RB (1996) Rigorous global search. Kluwer, Dordrecht

9. Klatte R, Kulisch U, Wiethoff A, Lawo C, Rauch M (1993) C-XSC. Springer, Berlin
10. Krawczyk R (1969) Newton-Algorithmen zur bestimmung von Nullstellen mit Fehlerschranken. Computing 4:187–201
11. Moore RE (1962) Interval arithmetic and automatic error analysis in digital computing. PhD Thesis Stanford Univ.
12. Moore RE (1966) Interval analysis. Prentice-Hall, Englewood Cliffs, NJ
13. Moore RE (1979) Methods and applications of interval analysis. SIAM, Philadelphia
14. Moore RE (1993) The resolution of close minima. Comput Math Appl 25(10–11):57–58
15. Moore RE, Hansen E, Leclerc AP (1992) Rigorous methods for global optimization. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton Univ. Press, Princeton, 321–342
16. Moore RE, Jones ST (1977) Safe starting regions for iterative methods. SIAM J Numer Anal 14(6):1051–1065
17. Moore RE, Kioustelidis JB (1980) A simple test for accuracy of approximate solutions to nonlinear (or linear) systems. SIAM J Numer Anal 17:521–529
18. Neumaier A (1990) Interval methods for systems of equations. Cambridge Univ. Press, Cambridge
19. Ratschek H, Rokne J (1984) Computer methods for the range of functions. Horwood and Wiley, London
20. Ratschek H, Rokne J (1988) New computer methods for global optimization. Horwood and Wiley, London
21. Website: http://cs.utep.edu/interval-comp/main.html.

Interval Analysis: Unconstrained and Constrained Optimization

R. Baker Kearfott
Department Math., University Louisiana at Lafayette, Lafayette, USA

MSC2000: 65G20, 65G30, 65G40, 65H20

Article Outline

Keywords
The Basic Branch and Bound Algorithm for Unconstrained Optimization
Acceleration Tools
Differences Between Unconstrained and Constrained Optimization
See also
References

Keywords

Constrained optimization; Automatic result verification; Interval computations; Global optimization

Interval algorithms for constrained and unconstrained optimization are based on adaptive, exhaustive search of the domain. Their overall structure is virtually identical to *Lipschitz optimization* as in [4], since interval evaluations of an objective function ϕ over an interval vector $\mathbf{x}$ correspond to estimation of the range of ϕ over $\mathbf{x}$ with Lipschitz constants. However, there are additional opportunities for acceleration of the process with interval algorithms, and use of outwardly rounded interval arithmetic gives the computations the rigor of a mathematical proof.

The interval algorithms are both complicated and accelerated by the presence of constraints, as is explained below.

See [5,2] or [3] for further details of concepts in this article.

The basic problem is

$$\begin{cases} \min & \phi(x) \\ \text{s.t.} & c(x) = 0 \\ & g(x) \le 0, \end{cases} \tag{1}$$

where ϕ: $\mathbf{x} \subset \mathbf{R}^n \to \mathbf{R}$, c: $\mathbf{x} \to \mathbf{R}^{m_1}$, and g: $\mathbf{x} \to R^{m_2}$, where $\mathbf{x}$ is an interval vector

$$\mathbf{x} = ([\underline{x}_1, \overline{x}_1], \ldots, [\underline{x}_n, \overline{x}_n])^\top.$$

The values $m_1 = 0$ and $m_2 = 0$ will be allowed, in which case the problem is considered to be unconstrained. It is emphasized here that, in problem (1), a *global optimum*, that is, the lowest possible value of ϕ over the feasible set, is sought.

The Basic Branch and Bound Algorithm for Unconstrained Optimization

The overall outline of an interval branch and bound algorithm for unconstrained global optimization is given in Table 1.

One way that a box is rejected in step 2b) of this algorithm is by using a bound on the range of the function ϕ over the interval vector (box) $\mathbf{x}$. In particular, suppose the value $\phi(x)$ at a point x is known. Then $\phi(x)$

Interval Analysis: Unconstrained and Constrained Optimization, Table 1

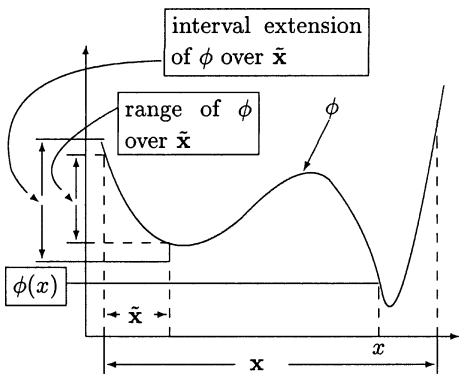
INPUT:	an initial box $\mathbf{x}_0$.
OUTPUT:	a list C of boxes that have been proven to contain critical points and a list U of boxes with small objective function values, but which could not otherwise be resolved.
1.	Initialize a list of boxes L by placing the initial search region $\mathbf{x}_0$ in L.
2.	DO WHILE $L \neq \emptyset$.
	a) Remove the first box x from L. (The boxes in L are in general in a particular order, depending on the actual algorithm.)
	b) (Process $\mathbf{x}$) Do one of the following:
	– reject $\mathbf{x}$;
	– reduce the size of $\mathbf{x}$;
	– determine that $\mathbf{x}$ contains a unique critical point, then find the critical point to high accuracy;
	– subdivide $\mathbf{x}$ to make it more likely to succeed at rejecting, reducing, or verifying uniqueness.
	c) Do the following to the box(es) resulting from Step 2b):
	– If $\mathbf{x}$ was rejected, do nothing.
	– If more then one box was derived from $\mathbf{x}$, insert all but one of them into L. Call the remaining box derived from $\mathbf{x}$ $\tilde{\mathbf{x}}$.
	– If there is a $\tilde{\mathbf{x}}$ that has been proven to contain critical point, insert it into C.
	– If there is a $\tilde{\mathbf{x}}$ that is small, but has not been proven to contain feasible point, insert it into U.
	END DO

is an upper bound for the global optimum. (In fact, if ϕ has been evaluated at various points, then the minimum of the resulting values is a usable upper bound on the global optimum.) Now suppose a lower bound ϕ on the range of ϕ over a box (or more generally, a region) $\mathbf{x} \subset \mathbf{R}^n$ can be computed, and that $\phi > \phi(x)$. Then there cannot be any global optimizers of ϕ within $\mathbf{x}$. The value ϕ can be obtained through an interval function value. This process is illustrated in the following figure.

The lower bound ϕ for the objective over the box $\mathbf{x}$ need not be obtained via interval computations. Indeed, if a Lipschitz constant $L_{\mathbf{x}}$ for ϕ is known over $\mathbf{x}$, and $\phi(\check{x})$ is known for $\check{x}$, the center of $\mathbf{x}$, then, for any $\widetilde{x} \in \mathbf{x}$,

$$\phi(\widetilde{x}) \geq \phi(\check{x}) - \tfrac{1}{2} L_{\mathbf{x}} \, \|\mathrm{w}(\mathbf{x})\| \,,$$

where $\mathrm{w}(\mathbf{x})$ is the vector of widths of the components of the interval vector $\mathbf{x}$. However, getting rigorous bounds on Lipschitz constants can require more human effort than the interval computation, and often results in bounds that are not as sharp as those from interval computation. (However, heuristically obtained approximate Lipschitz constants, as employed in the calculations in [4], have been highly successful at solving practical problems, albeit not rigorously.) Similarly, automated computations for Lipschitz constants as presently formulated result in bounds that are prov-



Interval Analysis: Unconstrained and Constrained Optimization, Figure 1
The midpoint test: Rejecting $\widetilde{\mathbf{x}}$ because of a high objective value

ably not as sharp as interval computations. Furthermore, use of properly rounded interval arithmetic, if used both in computing ϕ and $\phi(x)$, allows one to conclude *with mathematical rigor* that there are no global optima of ϕ within $\mathbf{x}$.

Use of this lower bound for ϕ is sometimes called the *midpoint test*, since the points x at which $\phi(x)$ is evaluated are often taken to be the vectors of midpoints of the boxes $\mathbf{x}$ produced during the subdivision process. (Actually, some implementations use the output of an approximate or local optimizer as x, to get an upper bound on the global optimum that is as low as possible.)

The simplest possible branch and bound algorithms need to contain both a box rejection mechanism and a subdivision mechanism. A common subdivision mechanism is to form two sub-boxes by bisecting the widest coordinate interval of $\mathbf{x}$ (with possible scaling factors). Heuristics and scaling factors, as well as several references to the literature, appear in [3, §4.3.2, p. 157 ff]. Alternatives to bisection, such as trisection, forming two boxes by cutting other than at a midpoint, etc. have also been discussed at conferences and studied empirically [1].

Acceleration Tools

Early and simple algorithms contain *only* the midpoint test mechanism and bisection mechanism described above. Such algorithms produce as output a large list U of small boxes (with diameters smaller than a stopping tolerance) and no list C of boxes that contain verified critical points. The list U in such algorithms contains clusters of boxes around actual global optimizers. Some Lipschitz constant-based algorithms are of this form. Note, however, that such algorithms are of limited use in high dimensions, since the number of boxes produced increases exponentially in the dimension n.

Interval computations provide more powerful tools for accelerating the algorithm. For a start, if an interval extension of the gradient $\nabla\phi(\mathbf{x})$ is computable then $0 \notin \nabla\phi(\mathbf{x})$ implies that $\mathbf{x}$ cannot contain a critical point, and $\mathbf{x}$ can be rejected. This tool for rejecting a box $\mathbf{x}$ is sometimes called the *monotonicity test*, since $0 \notin (\nabla\phi(\mathbf{x}))_i$ implies ϕ is monotonic over $\mathbf{x}$ in the ith component x_i, where $(\nabla\phi(\mathbf{x}))_i$ represents the ith component of the interval evaluation of the gradient $\nabla\phi$.

Perhaps the most powerful interval acceleration tool is *interval Newton methods*, applied to the system $\nabla\phi = 0$. Interval Newton methods can result in quadratic convergence to a critical point in the sense that the widths of the coordinates of the image of $\mathbf{x}$ are proportional to the square of the widths of the coordinates of $\mathbf{x}$. Interval Newton methods also can prove existence and uniqueness of a critical point or nonexistence of a critical point in $\mathbf{x}$. Thus, the need to subdivide a relatively large $\mathbf{x}$ is often eliminated, making a previously impractical algorithm practical. See ► Interval Newton methods and ► Interval fixed point theory.

For a more detailed algorithm, and for a discussion of parallelization of the branch and bound process, see ► Interval analysis: Parallel methods for global optimization.

Differences Between Unconstrained and Constrained Optimization

If $m_1 > 0$ or $m_2 > 0$ in problem (1), then the problem is one of *constrained optimization*. The midpoint test cannot be applied directly to constrained problems, since $\phi(x)$ is guaranteed to be an upper bound on the global optimum only if the constraints $c(x) = 0$ and $g(x) \leq 0$ are also satisfied at x. If there are only inequality constraints and none of the inequality constraints are active at x, then an interval evaluation of $g(x)$ will rigorously verify $g(x) < 0$, and x can be used in the midpoint test. However, if there are equality constraints (or if one or more of the inequality constraints is active), then an interval evaluation will yield $0 \in \mathbf{c}(x)$ (or $0 \in \mathbf{g}_i(x)$ for some i), and it cannot be concluded that x is feasible. In such cases, a small box $\check{\mathbf{x}}$ can be constructed about x, and it can be verified with interval Newton methods that $\check{\mathbf{x}}$ contains a feasible point. The upper bound of the interval evaluation $\phi(\check{\mathbf{x}})$ then serves as an upper bound on the global optimum, for use in the midpoint test. For details and references, see ► Interval analysis: Verifying feasibility.

On the other hand, constraints can be beneficial in eliminating infeasible boxes $\mathbf{x}$. In particular, $0 \notin c(\mathbf{x})$ or $g(\mathbf{x}) > 0$ implies that $\mathbf{x}$ can be rejected.

It is sometimes useful to consider *bound constraints* of the form $x_i \geq x_i$ and $x_j \leq x_j$ separately from the general inequality constraints $g(x) \leq 0$. Such bound constraints can generally coincide with the limits on

the search region $\mathbf{x}_0$, but are distinguished from simple search bounds. (It is possible for an unconstrained problem to have no optima within a search region, but it is not possible if all of the search region limits represent bound constraints.) See [3, §5.2.3, p. 180 ff] for details.

Example 1 Consider

$$\begin{cases} \min & \phi(x) = -(x_1 + x_2)^2 \\ \text{s.t.} & c(x) = x_2 - 2x_1 = 0. \end{cases} \tag{2}$$

Example (2) represents a constrained optimization problem with a single equality constraint and no bound constraints or inequality constraints. To apply the midpoint test in a rigorously verified algorithm, a box must first be found in which a feasible point is verified to exist. Suppose that a point algorithm, such as a generalized Newton method, has been used to find an approximate feasible point, say $\check{x} = (\frac{1}{4}, \frac{1}{2})^\top$. Now observe that $\nabla c \equiv (-2, 1)^\top$. Therefore, as suggested in ► Interval analysis: Verifying feasibility, x_2 can be held fixed at $x_2 = 1/2$. Thus, to prove existence of a feasible point in a neighborhood of $\check{x}$, an interval Newton method can be applied to $f(x_1) = c(x_1, 0.5) = 0.5 - 2x_1$. We may choose initial interval $\mathbf{x}_1 = [\,0.25 - \epsilon, 0.25 + \epsilon\,]$ with $\epsilon = 0.1$, to obtain

$$\mathbf{x}_1 = [-.15, -.35],$$

$$\widetilde{\mathbf{x}}_1 = 0.25 - \frac{0}{-2} = [0.25, 0.25] \subset \mathbf{x}_1,$$

This computation proves that, for $x_2 = 0.5$, there is a feasible point for $x_1 \in [0.25, 0.25]$. (See ► Interval Newton methods and ► Interval fixed point theory.) We may now evaluate ϕ over the box $([-0.25, -0.25], [0.5, 0.5])^\top$ (that is degenerate in the second coordinate, and also happens to be degenerate in the first coordinate for this example). We thus obtain

$$\phi([0.25, 0.25], [0.5, 0.5]) = \left[-\frac{9}{16}, -\frac{9}{16}\right],$$

and $-9/16$ has been proven to be an upper bound on the global optimum for example problem (2).

See also

- ► Automatic Differentiation: Point and Interval
- ► Automatic Differentiation: Point and Interval Taylor Operators
- ► Bounding Derivative Ranges
- ► Direct Search Luus–Jaakola Optimization Procedure
- ► Global Optimization: Application to Phase Equilibrium Problems
- ► Interval Analysis: Application to Chemical Engineering Design Problems
- ► Interval Analysis: Differential Equations
- ► Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ► Interval Analysis: Intermediate Terms
- ► Interval Analysis: Nondifferentiable Problems
- ► Interval Analysis: Parallel Methods for Global Optimization
- ► Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ► Interval Analysis: Systems of Nonlinear Equations
- ► Interval Analysis: Verifying Feasibility
- ► Interval Constraints
- ► Interval Fixed Point Theory
- ► Interval Global Optimization
- ► Interval Linear Systems
- ► Interval Newton Methods

References

1. Csallner AE, Csendes T (1995) Convergence speed of interval methods for global optimization and the joint effects of algorithmic modifications. Talk given at SCAN'95, Wuppertal, Germany, Sept. 26–29
2. Hansen ER (1992) Global optimization using interval analysis. M. Dekker, New York
3. Kearfott RB (1996) Rigorous global search: continuous problems. Kluwer, Dordrecht
4. Pinter JD (1995) Global optimization in action: continuous and Lipschitz optimization. Kluwer, Dordrecht
5. Ratschek H, Rokne J (1988) New computer methods for global optimization. Wiley, New York

Interval Analysis: Verifying Feasibility

R. Baker Kearfott
Department Math., University Louisiana at Lafayette, Lafayette, USA

MSC2000: 65G20, 65G30, 65G40, 65H20

Article Outline

Keywords
Introduction
General Feasibility: the Fritz John Conditions
Feasibility of Inequality Constraints
Infeasibility
Feasibility of Equality Constraints
See also
References

Keywords

Constrained optimization; Automatic result verification; Interval computations; Global optimization

Introduction

Constrained optimization problems are of the form

$$\begin{cases} \min & \phi(x) \\ \text{s.t.} & c_i(x) = 0, \quad i = 1, \ldots, m, \\ & g_j(x) \le 0, \quad j = 1, \ldots, q_1, \\ & \underline{x}_{i_k} \le x_{i_k}, \quad k = 1, \ldots, q_2 - \mu, \\ & x_{i_k} \le \overline{x}_{i_k}, \quad k = \mu + 1, \ldots, q_2, \end{cases} \tag{1}$$

where ϕ: $\mathbf{R}^n \to \mathbf{R}$, c_i: $\mathbf{R}^n \to \mathbf{R}$, and g_j: $\mathbf{R}^n \to \mathbf{R}$.

In interval branch and bound algorithms for finding global optima for problem (1), a search box of the form

$$\mathbf{x} = \left\{ (x_1, \ldots, x_n)^\top \in \mathbb{R}^n : \begin{array}{c} \underline{x}_i \le x_i \le \overline{x}_i, \\ 1 \le i \le n \end{array} \right\} \tag{2}$$

is generally given, where some of the sides in (2) correspond to bound constraints of problem (1), and the other sides merely define the extent of the search region. If there are no constraints c_i and g_j, then the box $\mathbf{x}$ is systematically tessellated into sub-boxes. The branch and bound algorithm, in its most basic form, proceeds as follows: Over each sub-box $\widetilde{\mathbf{x}}$, $\phi(\check{x})$ is computed for some $\check{x} \in \widetilde{\mathbf{x}}$, and the range of ϕ over $\widetilde{\mathbf{x}}$ is bounded (e. g. with a straightforward interval evaluation). If there are no constraints c_i and g_j, then the value $\phi(\check{x})$ represents an upper bound on the minimum of ϕ. The minimum such value $\overline{\phi}$ is kept as the tessellation and search proceed; if any box $\widetilde{\mathbf{x}}$ has a lower range bound greater than $\overline{\phi}$, it is rejected as not containing a global optimum. See [1,2], or [3] for details of such algorithms.

The situation is more complicated in the constrained case. In particular, the values $\phi(\check{x})$ cannot be taken as upper bounds on the global optimum unless it is known that $\check{x}$ is feasible. More generally, an upper bound on the range of ϕ over a small box $\check{\mathbf{x}}$ can be taken as an upper bound for the global optimum provided it is proven that there exists a feasible point of problem (1) within $\check{\mathbf{x}}$. This article outlines how this can be done.

General Feasibility: the Fritz John Conditions

An interval Newton method (see ▶ Interval Newton methods) can sometimes be used to prove existence of a feasible point of problem (1) that is a critical point of ϕ. In particular, the interval Newton method can sometimes prove existence of a solution to the Lagrange multiplier or *Fritz John system* within $\check{\mathbf{x}}$. For the Fritz John system, it is convenient to consider the q_2 bound constraints in the same form as the q_1 general inequality constraints, so that there are $q = q_1 + q_2$ general inequality constraints of the from $g_j(x) \le 0$. With that, the Fritz John system can be written as

$$F(x, u, v) = \begin{pmatrix} u_0 \nabla\phi(x) + \sum_{j=1}^{q} u_j \nabla g_j + \sum_{i=1}^{m} v_i \nabla c_i(x) \\ u_1 g_1 \\ \vdots \\ u_q g_q \\ c_1(x) \\ \vdots \\ c_m(x) \\ \left(u_0 + \sum_{j=1}^{q} u_j + \sum_{i=1}^{m} v_i^2\right) - 1 \end{pmatrix} = 0, \tag{3}$$

where $u_j \ge 0$, $j = 1, \ldots, q$, the v_i are unconstrained, and the last equation is one of several possible normalization conditions. For details, see [1, §10.5] or [2, §5.2.5].

However, computational problems occur in practice with the system (3). It is more difficult to find a good approximate critical point (for an appropriate small box $\check{\mathbf{x}}$) of the entire system (3) than it is to find a point where the inequality and equality constraints are satisfied. Furthermore, if an interval Newton method is applied to (3) over a large box, the corresponding interval Jacobi matrix or slope matrix typically contains singular matrices and hence is useless for

existence verification. This is especially true if it is difficult to get good estimates for the *Lagrange multipliers* u_j and v_i. For this reason, the techniques outlined below are useful.

Feasibility of Inequality Constraints

Proving feasibility of the inequality constraints is sometimes possible by evaluating the g_j with interval arithmetic: If $\mathbf{g}_j(\check{\mathbf{x}}) \leq 0$), then every point in $\check{\mathbf{x}}$ is feasible with respect to the constraint $g_j(x) \leq 0$; see [3]. However, if $\check{\mathbf{x}}$ corresponds to a point at which g_j is active, then $0 \in \mathbf{g}_j(\check{\mathbf{x}})$, and no conclusion can be reached from an interval evaluation. In such cases, feasibility can sometimes be proven by treating $g_j(x) = 0$ as one of the equality constraints, then using the techniques below.

Infeasibility

An inequality constraint g_j is proven infeasible over $\check{\mathbf{x}}$ if $\mathbf{g}_j(\check{\mathbf{x}}) > 0$, and an equality constraint c_i is infeasible over $\check{\mathbf{x}}$ if either $\mathbf{c}_i(\check{\mathbf{x}}) > 0$ or $\mathbf{c}_i(\check{\mathbf{x}}) < 0$. See [3].

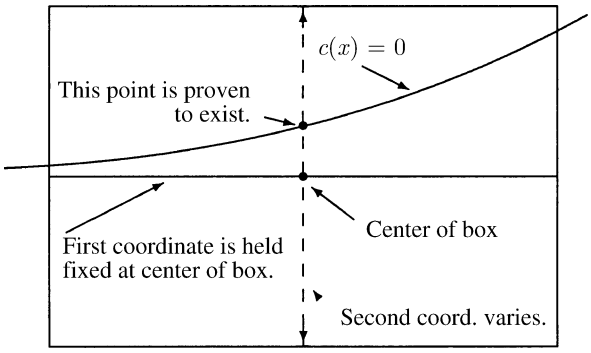
Feasibility of Equality Constraints

The equality constraints

$$c(x) = (c_1(x), \ldots, c_m(x))^\top = 0,$$

c: $\mathbf{R}^n \to \mathbf{R}^m$, $n \geq m$, can be considered an underdetermined system of equations, whereas interval Newton methods generally prove existence and/or uniqueness for square systems. However, fixing $n - m$ coordinates $\check{x}_i \in \check{\mathbf{x}}_i$ allows interval Newton methods to work with $\widetilde{c}$: $\mathbb{R}^m \to \mathbb{R}^m$, to prove existence of a feasible point within $\check{\mathbf{x}}$. In principle, indices of the coordinates to be held fixed are chosen to correspond to coordinates in which c is varying least rapidly. For a set of test problems, the most successful way appears to be choosing those coordinates corresponding to the rightmost columns after Gaussian elimination with complete pivoting has been applied to the rectangular matrix $c'(\check{x})$ for some $\check{x} \in \check{\mathbf{x}}$. Figure 1 illustrates the process in two dimensions.

Certain complications arise. For example, if bound constraints or inequality constraints are active, then either the point $\check{x}$ must be perturbed or else the bound or inequality constraints must be treated as equality constraints. Handling this case by perturbation is discussed in [2, p. 191ff].



Interval Analysis: Verifying Feasibility, Figure 1
Proving that there exists a feasible point of an underdetermined constraint system

For the original explanation of the Gaussian elimination-based process, see [1, §12.4]. In [2, §5.2.4], additional background, discussion, and references appear.

See also

- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Point and Interval Taylor Operators
- ▶ Bounding Derivative Ranges
- ▶ Global Optimization: Application to Phase Equilibrium Problems
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Interval Analysis: Differential Equations
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Interval Analysis: Intermediate Terms
- ▶ Interval Analysis: Nondifferentiable Problems
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ▶ Interval Analysis: Systems of Nonlinear Equations
- ▶ Interval Analysis: Unconstrained and Constrained Optimization
- ▶ Interval Constraints
- ▶ Interval Fixed Point Theory
- ▶ Interval Global Optimization
- ▶ Interval Linear Systems
- ▶ Interval Newton Methods

References

1. Hansen ER (1992) Global optimization using interval analysis. M. Dekker, New York
2. Kearfott RB (1996) Rigorous global search: continuous problems. Kluwer, Dordrecht
3. Ratschek H, Rokne J (1988) New computer methods for global optimization. Wiley, New York

Interval Constraints

Interval Propagation

FRÉDÉRIC BENHAMOU
IRIN, Université de Nantes, Nantes, France

MSC2000: 68T20, 65G20, 65G30, 65G40

Article Outline

Keywords
See also
References

Keywords

Constraint programming; Continuous constraint satisfaction problems; Local consistency; Propagation

Interval constraint processing is an alternative technology designed to process sets of (generally nonlinear) continuous or mixed *constraints* over the real numbers. It associates propagation and *search* techniques developed in *artificial intelligence* and methods from interval analysis.

Interval constraints are used in the design of the constraint solving and optimization engines of most modern *constraint programming* languages and have been used to solve industrial applications in areas like mechanical design, chemistry, aeronautics, medical diagnosis or image synthesis.

The term interval constraint is a generic term denoting a constraint (that is a first order atomic formula such as an equation, inequation or more generally a relation) in which variables are associated with intervals. These intervals denote domains of possible values for these variables. In general, intervals are defined over the real numbers but the concept is general enough to address other constraint domains (e. g. non negative integers, Booleans, lists, sets, etc.). When defined over the real numbers, interval constraint sets are often called *continuous* or *numerical constraint satisfaction problems*.

The main idea underlying interval constraint processing—also called *interval propagation*—is, given a set of constraints S involving variables $\{v_1, \ldots, v_n\}$ and a set of *floating point intervals* $\{I_1, \ldots, \mathrm{I}_n\}$ representing the domains of possible values of variables, to isolate a set of $\{n\}$-ary *canonical boxes* (Cartesian products of I_i*s* subintervals whose bounds are either equal or consecutive floating point numbers) approximating the constraint system solution space. To compute such a set, a search procedure navigates through the Cartesian product $I_1 \times \cdots \times I_n$ alternating pruning and branching steps.

The pruning step uses a relational form of *interval arithmetic* [1,11]. Given a set of constraints over the reals, interval arithmetic is used to compute local approximations of the solution space for a given constraint. This approximation results in the elimination of values from the domains of the variables and these domain modifications are propagated through the whole constraint set until reaching a stable state. This stable state is closely related to the notion of *arc consistency* [9,10], a well-known concept in artificial intelligence. The branching step consists in a bisection-based divide-and-conquer procedure on which a number of strategies and heuristics can be applied.

Interval constraints were first introduced by J.G. Cleary in [5]. The initial goal was to address the incorrectness of floating point numerical computations in the *Prolog* language while introducing a relational form of arithmetic more adapted to the language formal model. These ideas, clearly connected to the concepts developed in *constraint logic programming* [6,7], were then implemented in *BNR-Prolog* [12]. Since then, many other constraint languages and systems have used interval constraints as their basic constraint solving engine, for example *CLP(BNR)* [4], *Prolog IV* [13] or *Numerica* [16].

In the interval framework, the basic data structure is a set of ordered pairs of numbers taken in a finite set of particular rational numbers augmented with the two infinities (this set generally coincides with a set of IEEE floating point numbers). Such a pair, called a floating point interval or, more concisely, an interval, denotes,

as expected, the set of real numbers in between the lower and upper bounds. Operations and relations over the reals can be lifted to intervals (using floating point operations and outward rounding) so as to keep numerical errors under control. In particular, correctness of computations is guaranteed by a fundamental theorem due to R.E. Moore [11].

Assuming a finite set of intervals closed under intersection, every relation ρ over **R** can be approximated with its *interval enclosure* (i. e. the intersection of all intervals containing it). The approximation of any $\{n\}$-ary relation is then defined as the Cartesian product of its projection approximations. These Cartesian products of intervals are called *boxes*. The set of boxes, partially ordered by inclusion, is the complete lattice made of the fixed points of the closure operator that maps $\{n\}$-ary relations over **R** to their approximations. The intersection of all boxes containing an n-ary relation defines an outer approximation notion. A dual notion of inner approximation can be defined as the union of all boxes contained in the relation.

Given a finite set of constraints S and an initial n-ary box X representing the domains (intervals) for all variables occurring in S, every constraint in S represents an n-ary relation ρ (modulo an appropriate cylindrification). The main idea is then to compute a box approximating the solution set defined by S and X. In the interval constraint framework, this approximation is generally computed by applying the following algorithm, called here *NC3* to reflect its similarity to the arc consistency algorithm *AC3* [10].

The call to the function narrow in NC3 is an algorithmic narrowing process. Every constraint c in S and its corresponding relation ρ is associated with an operator N_c, called *constraint narrowing operator*, mapping boxes to boxes and verifying the properties of correctness, contractance, monotonicity and idempotence, that is for every boxes X, X'

1) $X \cap \rho \subseteq N_c(X)$;
2) $N_c(X) \subseteq X$;
3) $X \subseteq X'$ implies $N_c(X) \subseteq N_c(X')$
4) $N_c(N_c(X)) = N_c(X)$.

When such operators are associated with the constraints of a set S, the function narrow(X, c) simply returns $N_c(X)$. The algorithm stops when a stable state is reached, i. e. no (strict) narrowing is possible with respect to any constraint. The result of the main step

```
function NC3()
  input: S, a (nonempty) constrain system,
         X, a (nonempty) box
  output: X' ⊆ X
  Queue all constraints from S in Q
  REPEAT
    select a constraint c from Q
    % Narrow down X with respect to c
    X' ← narrow(X, c)
    % if X' is empty, S is inconsistent
    IF X' = ∅ THEN return ∅
    % Queue the constraints whose variables'
    % domains have changed. Delete c from Q
    Let S' = {c ∈ S : ∃v ∈ var(c), X'_v ⊂ X_v}
    Q ← Q ∪ S'\{c}
    X ← X'
  UNTIL Q is empty
  return X
END % NC3
```

NC3: A generic narrowing algorithm

is to remove (some) incompatible values from the domains of the variables occurring in c. Furthermore, it can be shown that NC3 terminates, is correct (the final box contains all solutions of the initial system included in $\{X\}$, confluent (selection of constraints in the main loop is strategy independent) and computes the greatest common fixed point of the constraint narrowing operators that is included in the initial box [2].

Over the real numbers, different constraint narrowing operators can be defined, resulting in different local consistency notions. A system is said to be *locally consistent* (with respect to a family of constraint narrowing operators), if the Cartesian product of the domains of its variables is a common fixed point of the constraint narrowing operators associated with its constraints. The main local consistency notions used in continuous constraint satisfaction problems are: first order local consistencies deriving from arc consistency (*hull* (or *2B*) consistency [4,8], *box consistency* [3], and higher order local consistencies deriving from k-consistency (*3B*, *kB-consistency* [8], *box*(2)*-consistency* [14]).

More precisely, let apx(c) denote the smallest box enclosing the relation associated with a constraint c. The family of constraint narrowing operators N defined as: For all box X and all constraint c, $N_c(X)$ = apx($X \cap c$) is the support of hull consistency. These opera-

tors can be computed for very simple constraints, often named primitive constraints (e. g. $x + y = z$, $\sin(x) = y$, ...) and complex constraints are decomposed into primitive constraints, eventually adding fresh variables.

The definition of *box consistency* involves the introduction of *projection constraints*. Given a multivariate constraint c over the reals, the projection constraint of c with respect to a variable v is obtained by computing an interval extension of the constraint and by replacing every variable but v with the interval corresponding to its domain. The constraint narrowing operator associated with this projection constraint computes the greatest interval $[a, b]$ such that $[a, a^+$ and $(b^-, b]$, where a^+ (resp. b^-) denotes the successor (resp. the predecessor) of a (resp. b), verify the projection constraint. Besides the fact that this technique does not require the addition of any additional variable, these operators can be computed with an algorithm mixing interval Newton methods (cf. ► Interval Newton methods), propagation and bisection-based search.

Higher-order local consistencies are based on their first order counterparts. Operationally, the idea is to improve the enclosures accuracy by eliminating subintervals of the locally consistent domains that can be detected locally inconsistent. The general procedure is as follows: Consider a hull consistent interval constraint system (S, X). An equivalent 3B-consistent system is a system (S, X') such that, for every variable v, if $X_v' = [a, b]$, the system $(S, X'_{v \leftarrow [a, a^+]})$(resp. $(S, X'_{v \leftarrow [b^-, b]})$) is hull consistent, where $X'_{v \leftarrow I}$ denotes the Cartesian product X' where X_v' is replaced with I. Box(2)-consistency is defined in the same manner with respect to box consistency. The computational cost of higher-order local consistencies is generally high, but the local gain in accuracy was shown to outperform most existing techniques in several challenging problems (see, for example the circuit design problem in [14]).

Finally, the above mentioned interval constraint techniques are also used for unconstrained and constrained optimization problems (see for example [15,16]).

See also

- ► Automatic Differentiation: Point and Interval
- ► Automatic Differentiation: Point and Interval Taylor Operators
- ► Bounding Derivative Ranges
- ► Global Optimization: Application to Phase Equilibrium Problems
- ► Interval Analysis: Application to Chemical Engineering Design Problems
- ► Interval Analysis: Differential Equations
- ► Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ► Interval Analysis: Intermediate Terms
- ► Interval Analysis: Nondifferentiable Problems
- ► Interval Analysis: Parallel Methods for Global Optimization
- ► Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ► Interval Analysis: Systems of Nonlinear Equations
- ► Interval Analysis: Unconstrained and Constrained Optimization
- ► Interval Analysis: Verifying Feasibility
- ► Interval Fixed Point Theory
- ► Interval Global Optimization
- ► Interval Linear Systems
- ► Interval Newton Methods

References

1. Alefeld G, Herzberger J (1983) Introduction to interval computations. Acad. Press, New York
2. Benhamou F, Granvilliers L (1997) Automatic generation of numerical redundancies for non-linear constraint solving. Reliable Computing 3(3):335–344
3. Benhamou F, McAllester D, Van Hentenryck P (1994) CLP(intervals) revisited. Proc. of ILPS'94, MIT, Cambridge, MA, pp 1–21
4. Benhamou F, Older WJ (1997) Applying interval arithmetic to real, integer and Boolean constraints. J Logic Programming 32(1):1–24
5. Cleary JG (1987) Logical arithmetic. Future Computing Systems 2(2):125–149
6. Colmerauer A (1990) An introduction to Prolog III. Comm ACM 33(7):69–90
7. Jaffar J, Lassez JL (1987) Constraint logic programming. Proc. 14th ACM Symp. Principles of Programming Languages (POPL'87), ACM, New York, pp 111–119
8. Lhomme O (1993) Consistency techniques for numeric CSPs. In: Bajcsy R (ed) Proc. 13th IJCAI. IEEE Computer Soc. Press, New York, pp 232–238
9. Mackworth A (1977) Consistency in networks of relations. Artif Intell 8(1):99–118
10. Montanari U (1974) Networks of constraints: Fundamental properties and applications to picture processing. Inform Sci 7(2):95–132

11. Moore RE (1966) Interval analysis. Prentice-Hall, Englewood Cliffs, NJ
12. Older W, Vellino A (1993) Constraint arithmetic on real intervals. In: Benhamou F, Colmerauer A (eds) Constraint Logic Programming: Selected Research. MIT, Cambridge, MA
13. PrologIA (1994) Prolog IV: Reference manual and user's guide
14. PugetJ-F, Van Hentenryck P (1998) A constraint satisfaction approach to a circuit design problem. J Global Optim 13:75–93
15. Van Hentenryck P, McAllester D, Kapur D (1997) Solving polynomial systems using a branch and prune approach. SIAM J Numer Anal 34(2):797–827
16. Van Hentenryck P, Michel L, Deville Y (1997) Numerica: A modeling language for global optimization. MIT, Cambridge, MA

Interval Fixed Point Theory

R. Baker Kearfott
Department Math., University Louisiana at Lafayette, Lafayette, USA

MSC2000: 65G20, 65G30, 65G40, 65H20

Article Outline

Keywords
Classical Fixed Point Theory and Interval Arithmetic
The Krawczyk Method and Fixed Point Theory
Interval Newton Methods and Fixed Point Theory
Uniqueness
Infinite-Dimensional Problems
See also
References

Keywords

Fixed point iteration; Automatic result verification; Interval computations; Global optimization

Interval methods (interval Newton methods and the Krawczyk method) can be used to prove existence and uniqueness of solutions to linear and nonlinear finite-dimensional and infinite-dimensional systems, given floating-point approximations to such solutions. (See ► Interval Newton methods; and [6,8].) In turn, these existence-proving interval operators have a close relationship with the classical theory of fixed-point iteration. This relationship is sketched here.

Classical Fixed Point Theory and Interval Arithmetic

Various fixed point theorems, applicable in finite- or infinite-dimensional spaces, state roughly that, if a mapping maps a set into itself, then that mapping has a fixed point within that set. For example, the *Brouwer fixed point theorem* states that, if D is homeomorphic to the closed unit ball in $\mathbf{R}^n$ and P is a continuous mapping such that P maps D into D, then P has a fixed point in D, that is, there is an $x \in D$ with $x = P(x)$.

Interval arithmetic can be naturally used to test the hypotheses of the Brouwer fixed point theorem. An interval extension $\mathbf{P}$ of P has the property that, if $\mathbf{x}$ is an interval vector with $\mathbf{x} \subseteq D$, then $\mathbf{P}(\mathbf{x})$ contains the range $\{P(x)\colon x \in \mathbf{x}\}$, and an interval extension $\mathbf{P}$ can be obtained simply by evaluating P with interval arithmetic. Furthermore, with *outward roundings*, this evaluation can be carried out so that the floating point intervals (whose end points are machine numbers) *rigorously* contain the actual range of P. Thus, if $\mathbf{P}(\mathbf{x}) \subset \mathbf{x}$, one can conclude that P has a fixed point within $\mathbf{x}$.

Another fixed point theorem, *Miranda's theorem*, follows from the Brouwer fixed point theorem, and is directly useful in theoretical studies of several interval methods. Miranda's theorem is most easily stated with the notation of interval computations: Suppose $\mathbf{x} \subset \mathbf{R}^n$ is an interval vector, and for each i, look at the lower ith face $\mathbf{x}_i$ of $\mathbf{x}$, defined to be the interval vector all of whose components except the ith component are those of $\mathbf{x}$, and whose ith component is the lower bound x_i of the ith component $\mathbf{x}_i$ of $\mathbf{x}$. Define the upper ith face $\mathbf{x}_{\overline{i}}$ of $\mathbf{x}$ similarly. Let $P\colon \mathbf{x} \to \mathbf{R}^n$, $P(x) = (P_1(x), \ldots, P_n(x))$ be continuous, and let $\mathbf{P} = (\mathbf{P}_1, \ldots, \mathbf{P}_n)$ be any interval extension of P. Miranda's theorem states that, if

$$\mathbf{P}_i(\mathbf{x}_{\underline{i}})\mathbf{P}_i(\mathbf{x}_{\overline{i}}) \leq 0, \tag{1}$$

then P has a fixed point within $\mathbf{x}$.

The Krawczyk Method and Fixed Point Theory

R.E. Moore provided one of the earlier careful analyses of interval Newton methods in [5]. There, the *Krawczyk*

method was analyzed as follows: The chord method is defined as

$$P(x) = x - Yf(x), \tag{2}$$

where the iteration matrix is normally taken to be $Y = \left(f'(\widetilde{x})\right)^{-1}$ for some Jacobi matrix $f'(\widetilde{x})$ with $\widetilde{x} \in \mathbf{x}$, where solutions of $f(x) = 0, f\colon D \subseteq \mathbf{R}^n \to \mathbf{R}^n$ are sought. A *mean value extension* is then used:

$$P(x) \in P(\check{x}) + \mathbf{P}'(\mathbf{x})(x - \check{x}),$$

whence

$$\begin{aligned} &\mathbf{K}(\mathbf{x}, \check{x}) = \mathbf{P}(\mathbf{x}) \\ &= P(\check{x}) + \mathbf{P}'(\mathbf{x})(\mathbf{x} - \check{x}) \\ &= \check{x} - Yf(\check{x}) + \left(I - Yf'(\mathbf{x})\right)(\mathbf{x} - \check{x}) \end{aligned} \tag{3}$$

is an interval extension of P. Thus, the fact that the range of P obeys

$$\{P(x)\colon\ x \in \mathbf{x}\} \subseteq \mathbf{P}(\mathbf{x}) = K(\mathbf{x}, \check{x})$$

coupled with the Brouwer fixed point theorem implies that, if

$$K(\mathbf{x}, \check{x}) \subseteq \mathbf{x},$$

then there exists a fixed point of P, and hence solution $x^* \in K(\mathbf{x}, \check{x}), f(x^*) = 0$.

By analyzing the norm norm $\| I - Yf'(\mathbf{x}) \|$, Moore further concludes, basically, that if

$$\left\| I - Yf'(\mathbf{x}) \right\| < 1,$$

then any solution $x^* \in \mathbf{x}$ must be unique; for an exact statement and details, see [5].

Interval Newton Methods and Fixed Point Theory

Traditional interval Newton methods are of the form

$$\mathbf{N}(f, \mathbf{x}, \check{x}) = \check{x} + \mathbf{v}, \tag{4}$$

where $\mathbf{v}$ is an interval vector that contains all solutions v to point systems $Av = -f(\check{x})$, for $A \in \mathbf{f}'(\mathbf{x})$, where $\mathbf{f}'(\mathbf{x})$ is either an interval extension to the Jacobi matrix of f over $\mathbf{x}$ or an interval slope matrix; see ► Interval Newton methods. [7, Thm. 5.1.7] asserts that, if $\mathbf{N}(f, \mathbf{x}, \check{x}) \subset \operatorname{int} \mathbf{x}$, where $\mathbf{f}'(\mathbf{x})$ is a 'Lipschitz set' for $\mathbf{f}$, int$\mathbf{x}$ denotes the interior of $\mathbf{x}$, and $\check{x} \in \operatorname{int}(\mathbf{x})$, then there is a solution of $f(x) = 0$ within $\mathbf{N}(f, \mathbf{x}, \check{x})$, and this solution is unique within $\mathbf{x}$. Classical fixed point theory is used in the succinct proof of this general theorem.

When the interval Gauss–Seidel method is used to find the solution set bounds $\mathbf{v}$, a very clear correspondence to Miranda's theorem can be set up. This is done in [3].

Uniqueness

In classical fixed point theory, the contractive mapping theorem (a nongeneric property) is often used to prove uniqueness. For example, suppose P is Lipschitz with Lipschitz constant $L < 1$, that is,

$$\|P(x) - P(y)\| \leq L \|x - y\| \quad \text{for some } L < 1. \tag{5}$$

Then $x = P(x)$ and $y = P(y)$ implies $\| x - y \| = \| P(x) - P(y) \| \leq L \| x - y \|$, which can only happen if $x = y$. (This argument appears in many elementary numerical analysis texts, such as [4].)

An alternate proof of uniqueness involves nonsingularity (i. e., *regularity*) of the mapping f for which we seek x with $f(x) = 0$. In particular, if $f(x) = Ax$ is linear, corresponding to a nonsingular matrix A, then $f(x) = 0$ and $f(y) = 0$ implies

$$0 = f(x) - f(y) = Ax - Ay = A(x - y), \tag{6}$$

whence nonsingularity of A implies $x - y = 0$, i. e. $x = y$.

Without interval arithmetic, the argument in (6) cannot be generalized easily to nonlinear systems. Basically, invertibility implies uniqueness, and one must somehow prove invertibility. However, with interval arithmetic, uniqueness follows directly from an equation similar to (6), and regularity can be proven directly with an interval Newton method. In particular, if the image under the interval Newton method (4) is bounded, then *every* point matrix $A \in \mathbf{f}'(\mathbf{x})$ must be nonsingular. (This is because the bounds on the solution set to the linear system $\mathbf{f}'(\mathbf{x})v = -f(\check{x})$ must contain the set of solutions to all systems of the form $Av = -f(\check{x}), A \in \mathbf{f}'(\mathbf{x})$.) Then, the mean value theorem implies that, for every $x \in \mathbf{x}, y \in \mathbf{x}$,

$$f(x) - f(y) = A(x - y) \quad \text{for some } A \in \mathbf{f}'(\mathbf{x}). \tag{7}$$

This is in spite of the fact that, in (7), A is in general not equal to any $f'(x)$ for some $x \in \mathbf{x}$. In fact, (7) follows

from considering f componentwise:

$$f_i(y) = f_i(x) + \left(\nabla f_i(c_i)\right)^{\top}(y - x),$$

for some c_i, *different for each i*, on the line connecting x and y; the matrix $A \in \mathbf{f}'(\mathbf{x})$ can be taken to have its ith row equal to $(\nabla f_i(c_i))^{\mathsf{T}}$. Thus, because of the nonsingularity of A in (7), $f(x) = 0, f(y) = 0$ implies $0 = A(x - y$ and $x = y$.

Summarizing the actual results,

$$\mathbf{N}(f, \mathbf{x}, \check{x}) \subset \operatorname{int} \mathbf{x}, \tag{8}$$

where $\mathbf{N}(f, \mathbf{x}, \check{x})$ is as in (4), then classical fixed-point theory combined with properties of interval arithmetic implies that there is a unique solution to $f(x) = 0$ in $\mathbf{N}(f, \mathbf{x}, \check{x})$, and hence in $\mathbf{x}$.

If slope matrices are used in place of an interval Jacobi matrix $\mathbf{f}'(\mathbf{x})$, then (7) no longer holds, and (8) no longer implies uniqueness. However, a two-stage process, involving evaluation of an interval derivative over a small box containing the solution and evaluation of a slope matrix over a large box containing the small box, leads to an even more powerful existence and uniqueness test than using interval Jacobi matrices alone. This technique perhaps originally appeared in [9]. A statement and proof of the main theorem can also be found in [3, Thm. 1.23, p. 64].

Infinite-Dimensional Problems

Many problems in infinite-dimensional spaces (e. g. certain variational optimization problems) can be written in the form of a compact operator fixed point equation, $x = P(x)$, where P: $S \to S$ is some compact operator operating on some normed linear space S. In many such cases, P is approximated numerically from a finite-dimensional space of basis functions $\{\phi_i: i = 1, \ldots, n\}$ (e. g. splines or finite element basis functions ϕ_i), and the approximation error can be computed. That is, $P(x) = P_n(y) + R_n(y)$, where $y \in \mathbf{R}^n$ is an approximation to $x \in S$, and $R_n(y)$ is the error that is computable as a function of y. Thus, a fixed point iteration can be set up of the form

$$y \leftarrow \widetilde{P}_n(y) \equiv P_n(y) + R_n(y), \tag{9}$$

where $y \in \mathbf{R}^n$. (The dimension n can be increased as iteration proceeds.)

For (9), the *Schauder fixed point theorem* is an analogue of the Brouwer fixed point theorem; see [1, p. 154]. Furthermore, interval extensions can be provided to both P_n and R_n, so that an analogue to finite-dimensional computational fixed point theory exists. In particular, if

$$\widetilde{P}_n(\mathbf{y}) \subset \operatorname{int} \mathbf{y}, \tag{10}$$

then there exists a fixed point of P within the ball in S centered at the midpoint of $\mathbf{y}$ and with radius equal to the radius of $\mathbf{y}$. (For these purposes,

$$\mathbf{y} = \sum_{i=1}^{n} \mathbf{a}_i \phi_i$$

can be identified with the interval vector $(\mathbf{a}_1, \ldots, \mathbf{a}_n)^{\mathsf{T}}$ corresponding to the coefficients in the expansion.) For details, see [6, Chap. 15]. Also see [2] for a theoretical development and various examples worked out in detail.

See also

- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Point and Interval Taylor Operators
- ▶ Bounding Derivative Ranges
- ▶ Global Optimization: Application to Phase Equilibrium Problems
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Interval Analysis: Differential Equations
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Interval Analysis: Intermediate Terms
- ▶ Interval Analysis: Nondifferentiable Problems
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ▶ Interval Analysis: Systems of Nonlinear Equations
- ▶ Interval Analysis: Unconstrained and Constrained Optimization
- ▶ Interval Analysis: Verifying Feasibility
- ▶ Interval Constraints
- ▶ Interval Global Optimization
- ▶ Interval Linear Systems
- ▶ Interval Newton Methods

References

1. Istrăţescu VI (1981) Fixed point theory: An introduction. Reidel, London
2. Kaucher EW, Miranker WL (1984) Self-validating numerics for function space problems. Acad. Press, New York
3. Kearfott RB (1996) Rigorous global search: Continuous problems. Kluwer, Dordrecht
4. Kincaid D, Cheney W (1991) Numerical analysis. Brooks/Cole, Pacific Grove, CA
5. Moore RE (Sept. 1977) A test for existence of solutions to nonlinear systems. SIAM J Numer Anal 14(4):611–615
6. Moore RE (1985) Computational functional analysis. Horwood, Westergate
7. Neumaier A (1990) Interval methods for systems of equations. Cambridge Univ. Press, Cambridge
8. Plum M (1994) Inclusion methods for elliptic boundary value problems. In: Herzberger J (ed) Topics in Validated Computations. Stud Computational Math. North-Holland, Amsterdam, pp 323–380
9. Rump SM (1994) Verification methods for dense and sparse systems of equations. In: Herzberger J (ed) Topics in Validated Computations. Elsevier, Amsterdam, pp 63–135

Interval Global Optimization

H. Ratschek[1], J. Rokne[2]

[1] Math. Institut, Universität Düsseldorf, Düsseldorf, Germany

[2] Department Computer Sci., University Calgary, Calgary, Canada

MSC2000: 65K05, 90C30, 65G20, 65G30, 65G40

Article Outline

Keywords
Introduction
Interval Arithmetic
Interval Newton Methods
  The Preconditioning Step
  The Relaxation Steps (Gauss–Seidel)
Three Prototype Algorithms for the Unconstrained Problem
Convergence Properties of the Prototype Algorithms
Accelerating and Related Devices
  The Monotonicity Test
  The Interval Newton Method
  Finding a Function Value as Small as Possible
  Bisections
  Use of Good Inclusion Functions, Slope Arithmetic
  The Nonconvexity Test
  Thin Evaluation of the Hessian Matrix
  Constraint Logic Programming
  Automatic Differentiation
  Parallel Computations
Global Optimization Over Unbounded Domains and Nonsmooth Optimization
  Global Optimization over Unbounded Domains
  Nonsmooth Optimization
Constrained Optimization
  The Penalty Approach
  The Direct Approach
Applications
  Chemistry and Chemical Engineering
  Physics, Electronics and Mechanical Engineering
  Economics
See also
References

Keywords

Global optimization; Nonsmooth optimization; Unbounded optimization; Interval methods

We give an overview of the general ideas involved in solving constrained and unconstrained global optimization problems using interval arithmetic. We include a discussion of a few prototype optimization algorithms and enumerate some applications in engineering, chemistry, manufacturing, economics and physics.

Introduction

Let I be the set of real compact intervals, $\mathbf{R}$ the set of reals, m a positive integer, $X \in I^m$, and $f: X \to \mathbf{R}$ the objective function. We assume that a *global minimum* f^* of f exists over X. Let X^* be the set of *global minimizers* of f over X. Then the *global unconstrained optimization problem* is written down concisely as

$$\min_{x \in X} f(x) \tag{1}$$

which means that f^* or X^* is to be determined. The *global constrained optimization problem* arises if a more general set $M \subseteq \mathbf{R}^m$, the so-called *feasible domain*, is considered. Solution methods for global constrained problems use the tools for global unconstrained problems, but additionally, further concepts are needed such

as numerical proofs of the guaranteed existence of feasible points in subareas of the working domain. Therefore we separate the treatment of the constrained case from the treatment of the unconstrained case.

The first interval techniques for treating global optimization problems were established by [13,20,30,31,46,64,65,66,67,71,72,74,81,98,103,104,105], etc. Although some of these references were focusing on special problems like convex or signomial programming, they provided concepts which would give insight into more general problems where they were later applied.

The overview that we provide in this article, can only cast a quick glance at the various topics that will be considered. Their thorough investigation may be found in [33,39,47,48,49,56,70,90,95,100].

Solving an optimization problem such as (1) requires, in general, the repetitive comparison of continua of values and the choosing of an optimum value. Since interval computation is a tool for handling continua, it provides competitive methods for solving global optimization problems. Simple prototype algorithms for unconstrained problems are discussed in order not to get too sophisticated. We choose three variants, on the one hand in order to keep track of the historical origins, on the other hand in order to show how small changes in the prototypes influence their convergence behavior. These prototypes are based on ideas of S. Skelboe [103], R.E. Moore [74], N.S. Asaithambi, Z. Shen and Moore [2], E.R. Hansen [30,31], and K. Ichida and Y. Fujii [46]. We do not have the space to provide prototype algorithms for constrained problems as well in this article. Thus we only discuss parts which we have to add to the unconstrained prototypes in order to get a procedure for constrained problems.

In general, interval algorithms for solving global optimization problems consist of

i) the main algorithm,
ii) accelerating devices.

The main algorithm is a *sequential deterministic algorithm* where *branch and bound techniques* are used. (An algorithm is called sequential if the nth step of the computation depends on the former steps. A method is deterministic if stochastic methods are avoided. By branch and bound principles is meant that the whole area X or M is not searched uniformly for the global minimizers; instead some parts (branches) are preferred. The branching depends on the bounding. It is required that for any box Y of the working area a lower bound for f over Y is known or computable.)

Interval arithmetic is used for point i) to achieve the bounds needed for the branch and bound techniques (f need not be Lipschitz, convex, etc.) and for point ii) to remove superfluous parts of the domains X or M.

The contents of this article is as follows: In the next two sections we introduce the interval tools which are required in the article. In section 4, three algorithms for solving (1) are presented. They are seemingly very similar, but their convergence properties, which are discussed in section 5, are different. The three algorithms are also of interest for historical reasons. A survey of acceleration devices, which aim to speed up the computation, is given in section 6. It is shown in section 7 that interval analysis is an excellent means for dealing with problems which have an unbounded domain or a nonsmooth objective function. In section 8, the constrained case is touched upon. Applications of these methods are collected in the final section 9.

Interval Arithmetic

The interval tools which are needed for the explanation of the basic features of interval methods in global optimization are described in this section. A thorough introduction to the whole area of interval arithmetic can be found, for example, in [1,4,52,74,102], etc. More advanced readers will enjoy [79]. The development of interval tools appropriate for dealing with optimization problems is presented in [88,90]; cf. also the Appendix of [86].

The *interval arithmetic operations* are defined by

$$A * B = \{a * b\colon\ a \in A, b \in B\} \quad \text{for } A, B \in I, \quad (2)$$

where the symbol $*$ may denote $+$, $-$, $\cdot$, or $/$. In general, A/B is not defined if $0 \in B$. (But see the sections on 'interval Newton methods' and 'global optimization over unbounded domains' below.) The meaning of (2) is the following: If some unknown reals α, β are included in known intervals, say $\alpha \in A$, $\beta \in B$, then it is guaranteed that the desired result $\alpha * \beta$, which is in general unknown, is contained in the known interval $A * B$. Definition (2) is equivalent to the following rules,

$$[a, b] + [c, d] = [a + c, b + d],$$
$$[a, b] - [c, d] = [a - d, b - c],$$

$$[a,b]\cdot[c,d]$$
$$= [\min(ac,ad,bc,bd), \max(ac,ad,bc,bd)],$$
$$\frac{[a,b]}{[c,d]} = [a,b]\cdot\left[\frac{1}{d},\frac{1}{c}\right] \quad \text{if } 0 \notin [c,d].$$

Therefore, the interval arithmetic operations can easily be realized on a computer. The algebraic properties of (2) are different from those of real arithmetic operations. The distributive law, for instance, does not hold for (2). A summary of the algebraic behavior of interval arithmetic is given in [85].

The main interval arithmetic tool applied to optimization problems is the concept of an inclusion function. Let again $X \in I^m$ and $f: X \to \mathbf{R}$. The set of compact intervals contained in X is denoted by $I(X)$. Let $\overline{f}(Y) = \{f(x): x \in Y\}$ for $Y \in I(X)$ be the range of f over Y. A function F is called an *inclusion function* for f if

$$\overline{f}(Y) \subseteq F(Y) \quad \text{for any } Y \in I(X).$$

The left and the right endpoint of $F(Y)$ will be denoted by $\min F(Y)$ and $\max F(Y)$, respectively.

Inclusion functions can be constructed in any programming language in which interval arithmetic is simulated or implemented via natural interval extensions: Firstly, let g be any function pre-declared in some programming language (like sin, cos, exp, etc.). Then the corresponding *pre-declared interval function* IG is defined by

$$\mathrm{IG}(Y) = \overline{g}(Y)$$
for any $Y \in I$ contained in the domain of g.

Since the monotonicity intervals of pre-declared functions g are well known it is easy to realize the interval functions IG on a computer. Nevertheless, the influence of rounding errors may be considered, see [30], for instance.

Secondly, let $f(x)$ be any function expression in the variable $x \in \mathbf{R}^m$. So, $f(x)$ may be an explicit formula or described by an algorithm not containing logical connectives at the moment. For simplicity, we assume that $f(x)$ is representable in a programming language. Let $Y \in I^m$ or let Y be an interval variable over I^m. Then the expression which arises if each occurrence of x in $f(x)$ is replaced by Y, if each occurrence of a pre-declared function g in $f(x)$ is replaced by IG, and if the arithmetic operations in $f(x)$ are replaced by the corresponding interval arithmetic operations, is called the *natural interval extension* of $f(x)$ to Y, and it is denoted by $f(Y)$, see [71]. Due to (2) and the definition of the IG's we get the *inclusion principle* for (programmable) functions

$$a \in Y \quad \text{implies} \quad f(a) \in f(Y). \tag{3}$$

Therefore, $f(Y)$, seen as a function in Y, is an inclusion function for the function $f(x)$.

For example, if $f(x) = x_1 \sin x_2 - x_3$ for $x \in \mathbf{R}^3$, then $f(Y) = Y_1 \in Y_2 - Y_3$ is the natural interval extension of $f(x)$ to $Y \in I^3$.

If logical connectives occur in an expression, the extensions are similar, cf. [55,87].

Due to the algebraic properties of interval arithmetic, different expressions for a real function f can lead to interval expressions which are different as functions. For example, if $f_1(x) = x - x^2$ and $f_2(x) = x(1 - x)$ for $x \in \mathbf{R}$, then $f_1(Y) = Y - Y^2 = [-1, 1]$ and $f_2(Y) = Y(1 - Y) = [0, 1]$ for $Y = [0, 1]$. For comparison, $\overline{f}(Y) = [0, \frac{1}{4}]$. In general, the problem arises as to how to find expressions of a given function that lead to natural interval extensions that are as good as possible. A partial solution to this problem can be found in [88].

A measure of the quality of an inclusion function F for $f: X \to \mathbf{R}$ is the so-called *excess width* ([71]), defined as $w(F(Y)) - w(\overline{f}(Y))$ for all $Y \in I(X)$, where $w([a, b]) = b - a$ is the width of an interval. F is called of *order* $\alpha > 0$ if

$$w(F(Y)) - w(\overline{f}(Y)) = O(w(Y^\alpha)) \quad \text{for } Y \in I(X),$$

where the width of a box $Y = Y_1 \times \cdots \times Y_m$ is defined by $w(Y) = \max_{i=1,\ldots,m} w(Y_i)$. In order to obtain good computational results it is necessary to choose inclusion functions having as high an order α as possible, when $w(Y)$ is small, see for example [88].

The endpoints of the intervals must be machine numbers, if interval arithmetic is implemented on a machine. This leads to a special topic called *machine interval arithmetic*. It can be considered as an approximation to interval arithmetic on computer systems.

Machine interval arithmetic is based on the inclusion isotonicity of the interval operations in the following manner: Let us again assume that α, β are the unknown exact values at any stage of the calculation, and that only including intervals are known, $\alpha \in A$, $\beta \in B$.

Then A, B might not be representable on the machine. Therefore A and B are replaced by the *smallest machine intervals* that contain A and B,

$$A \subseteq A_M, \qquad B \subseteq B_M.$$

A machine interval is an interval which has left and right endpoints that are machine numbers. From (2) it follows that

$$A * B \subseteq A_M * B_M.$$

The interval $A_M * B_M$ need not be a machine interval and it is therefore approximated by $(A_M * B_M)_M$ which is the smallest interval representable on the machine and which contains $A_M * B_M$. This leads to the *inclusion principle of machine interval arithmetic*:

$$\alpha \in A, \beta \in B \quad \text{implies} \quad \alpha * \beta \in (A_M * B_M)_M. \quad (4)$$

Thus, the basic principle of interval arithmetic is retained in machine interval arithmetic, that is, the exact unknown result is contained in the corresponding known interval, and *rounding errors are under control.*

We sum up: When a concrete problem has to be solved then our procedure is as follows: Firstly, the theory is done in interval arithmetic, secondly, the calculation is done in machine interval arithmetic, and finally, the inclusion principle provides the transition from interval arithmetic to machine interval arithmetic.

Many software packages for interval arithmetic are meanwhile available, which work under Fortran 77, Fortran 90, Pascal, C, C++, Prolog, etc. A good survey can be found, for instance, in [57].

Interval Newton Methods

Interval Newton methods are excellent methods for determining *all zeros* of a continuously differentiable vector-valued function $\phi: X \to \mathbf{R}^m$ where $X \in I^m$. These methods are important tools for nonlinear optimization problems since they can be used for computing all critical points of the objective function, f, by applying the methods to $J_\phi(x)$, where ϕ is the gradient function of f and J_ϕ the Jacobian of ϕ, or for solving the Karush–Kuhn–Tucker or John conditions in constrained optimization.

The interval Newton method was introduced by Moore [71] and it has been further extensively developed by many researchers. The latest state of art for interval Newton methods may be found in [79]. See also

► Interval Newton methods. The extensive treatment of the interval Newton method is not part of this introductory article so that we sketch it in an extremely simplified manner just in order to make the aim of the method understandable. For a detailed treatment see, for instance, [1,33,79,90,93], etc.

Interval Newton methods are closely connected to solving systems of linear interval equations. An unfortunate notation is widely used to describe this situation since it uses the notation of interval arithmetic in a doubtful manner which can lead to misunderstandings. I.e., let $A \in I^{m \times m}$, $B \in I^m$ then the solution of the *linear interval equation* (with respect to x or X)

$$Ax = B \quad \text{or} \quad AX = B$$

is not an interval vector X_0 that satisfies the equation, $AX_0 = B$, as one would expect. The solution is defined as the set

$$X = \{x \in \mathbb{R}^m : \ ax = b \text{ for some } a \in A, b \in B\}.$$

Thus, for example, the solution of the linear interval equation

$$[1, 2]x = [1, 2]$$

is $X = [1/2, 2]$. In general, the solution set is not a box if $m \geq 2$.

> It is therefore the aim of interval arithmetic solution methods to find at least a box which contains the solution set.

Accordingly, if $c \in \mathbf{R}^m$, then the solution of the linear interval equation

$$A(x - c) = B \quad \text{or} \quad A(X - c) = B$$

with respect to x or X is defined to be the set

$$X := c + Y := \{c + y : \ y \in Y\}$$

where Y is the solution of the interval equation $Ay = B$.

The following prototype algorithm aims to determine the zeros of $\phi: X \to \mathbf{R}^m$ in $X \in I^m$. Let $J(Y)$ be an inclusion function for the Jacobian matrix $J_\phi(x)$ for $Y \in I(X)$.

1. Set $X_0 := X$.
2. For $n = 0, 1, \ldots$
   a) choose $x_n \in X_n$,
   b) determine a superbox Z_{n+1} of the solution Y_{n+1} of the linear interval equation with respect to Y, $J(X_n)(x_n - Y) = \phi(x_n)$,
   c) set $X_{n+1} := Z_{n+1} \cap X_n$.

The interval Newton algorithm

Since we use it later we emphasize that *one iteration of the interval Newton algorithm* is just the execution of a), b) and c) for a particular value of n.

Interval Newton methods are distinguished by the particular choice of the superbox Z_{n+1}. For example, if Z_{n+1} is the box hull of Y_{n+1}, that is, the smallest box containing Y_{n+1}, then the method is called the interval Newton method (in the proper sense). If Z_{n+1} is obtained by using interval Gauss–Seidel steps combined with preconditioning as will be explained in the sequel, the method is named after Hansen and S. Sengupta [35]. Krawczyk's method [60] and Hansen–Greenberg's methods [34] are also widely used. Convergence properties exist under certain assumptions. The following general properties are useful for understanding the principle of application of the algorithm, see [1,71,73,79]:

1) If a zero, ξ, of ϕ exists in X then $\xi \in X_n$ for all n. This means that no zero is ever lost! This implies that:
2) If X_n is empty for some n then ϕ has no zeros in X.
3) If Z_{n+1} is obtained by Gauss–Seidel or Gauss elimination, possibly combined with preconditioning as mentioned below then
   i) if $Z_{n+1} \subseteq X_n$ for some n then ϕ has a zero in X,
   ii) $Z_{n+1} \subseteq \operatorname{int} X_n$ for some n then ϕ has a unique zero in X (where int means topological interior).
4) Under certain conditions one obtains

$$w(X_{n+1}) \leq \alpha(w(X_n))^2$$

for some constant $\alpha \geq 0$.

A very promising realization of the interval Newton algorithm is the Hansen–Sengupta version [35] where the linear system occurring in the Newton iteration step is solved by a *preconditioning step* and by *relaxation steps* (Gauss–Seidel).

Now we discuss just one iteration of the Hansen–Sengupta variant and suppress the index n when writing down the formulas that occur in the nth iteration. That is, we write

$$J(X)(x - Y) = \phi(x) \tag{5}$$

instead of

$$J(X_n)(x_n - Y) = \phi(x_n)$$

and, accordingly, we search for a superset Z of the solution set of (5), where X, $J(X)$, x and $\phi(x)$ are given. The solution set of (5) is also denoted by Y.

The Preconditioning Step

It was already argued by Hansen and R.R. Smith [37] that (5) was best solved by pre-multiplying by an approximate inverse of the midpoint of $J(X)$. If the approximate inverse is B, we obtain

$$BJ(X)(x - Y) = B\phi(x)$$

or

$$M(x - Y) = b \tag{6}$$

where $M = BJ(X)$ and $b = B\phi(x)$. In this manner the system has been modified to a system that is almost diagonally dominant provided the widths of the Jacobian entries are not too large and it is then amenable to Gauss–Seidel type iterations. It is obvious that the solution set of (6) contains the solution set of (5) such that no solution is lost in the above transformation. During the last years much research has been focusing on the preconditioning step, cf. for example, [59].

The Relaxation Steps (Gauss–Seidel)

The relaxation procedure for linear interval equations was developed in [36]. It consists mainly in the interpretation of the well-known noninterval Gauss–Seidel iteration procedure in an interval context. But much care is taken in the interval realization if division through intervals that contain zero occurs. We do not have the space for a complete discussion and refer, for example, to [33,90,93].

Instead of a relaxation iteration, interval Gauss elimination can be used. This is nothing more than the well-known Gauss elimination performed in an interval setting. Interval Gauss elimination is not as robust

as the interval Gauss–Seidel steps. It is, however, more effective under certain conditions (for instance, if the Jacobian or the preconditioned Jacobian matrix is diagonally dominant, see [79]). Practical experiences show that it is best to combine Gauss–Seidel steps with Gaussian elimination, cf. [33,79,90].

There is no urgent need for discussing convergence properties of the interval Newton algorithm since only single iterations are incorporated into the optimization algorithms, cf. the sections on 'accellerating and related devices' and 'applications' below, and hence the convergence theory of the latter is applicable, cf. the sections on 'convergence properties of the prototype algorithms' and 'applications' below. Only if it is already certain or very likely that the computation is approaching a global minimizer does it make sense to switch to the complete interval Newton algorithm and enjoy finally the quadratic convergence property (cf. property 4). Such a situation occurs, for example, if the objective function, in the unconstrained case, or the Lagrangian in the constrained case, is convex.

Three Prototype Algorithms for the Unconstrained Problem

The algorithms are designed to determine f^* or X^* or both as will be described later. They have the box X, the inclusion function F for $f: X \to \mathbf{R}$ and some accuracy parameters which may occur in the termination criteria, as input parameters. The termination criteria will depend on the actual case and will not be specified here, but see, for example, item c) in the section on 'convergence properties of the prototype algorithms'. For historical reasons, we go back to the roots of interval arithmetic optimization theory. We start with Moore's algorithm [71], which used uniform subdivision, but we already incorporate the first branch and bound steps as proposed by Skelboe [103], and finally we land at Hansen's algorithm [30,31], which was the first algorithm which featured convergence to both, to f^* and to X^*.

Algorithm 1 initializes a list $\mathcal{L} = \mathcal{L}_1$ consisting of one pair (X, y), see Step 3. Then the list is modified and enlarged at each iteration, see Steps 8 and 9. At the nth iteration a list $\mathcal{L} = \mathcal{L}_n$ consisting of n pairs is present,

$$\mathcal{L}_n = ((Z_{ni}, z_{ni}))_{i=1}^{n} \quad \text{where } z_{ni} = \min F(Z_{ni}).$$

1. Calculate $F(X)$.
2. Set $y := \min F(X)$.
3. Initialize list $\mathcal{L} = ((X, y))$.
4. Choose a coordinate direction k parallel to an edge of maximum length of $X = X_1 \times \ldots \times X_m$, i.e. $k \in \{i : w(X) = w(X_i)\}$.
5. Bisect X normal to direction k obtaining boxes V_1, V_2 such that $X = V_1 \cup V_2$.
6. Calculate $F(V_1)$, $F(V_2)$.
7. Set $v_i := \min F(V_i)$ for $i = 1, 2$.
8. Remove (X, y) from the list $\mathcal{L}$.
9. Enter the pairs (V_1, v_1) and (V_2, v_2) into the list such that the second members of all pairs of the list do not decrease.
10. Denote the first pair of the list by (X, y).
11. If the termination criteria hold, go to 13.
12. Go to 4.
13. End.

Algorithm 1: Moore–Skelboe

The leading pair of the list $\mathcal{L}_n$ will be denoted by

$$(X_n, y_n) = (Z_{n1}, z_{n1}).$$

The boxes X_n are called the *leading boxes* of the algorithm. It is assumed that the termination criteria of Step 11 are not satisfied during the whole computation such that the algorithm will not stop. In this case an infinite sequence of lists is produced.

Algorithm 1 was mainly established to determine f^*. Now, Ichida and Fujii [46] and Hansen [30,31] focused on the boxes Z_{ni} in order to get reasonable inclusions for X^*. While midpoint tests (cf. [2,30,31,46]) have no impact on the convergence properties of Algorithm 1, they are now important when getting inclusions of X^*. *Midpoint tests* are incorporated as follows: Let f_n be the lowest function value which has been calculated up to the completion of the list $\mathcal{L}_n$. (If no function values are available then $\min_{i=1,\ldots,n} \max F(X_i)$ can be taken as f_n.) Then all pairs (Z_{ni}, z_{ni}) of $\mathcal{L}$ are discarded that satisfy

$$f_n < z_{ni}.$$

This gives a reduced list $\overline{\mathcal{L}}_n$. Let $U_n = \cup Z_{ni}$ for all Z_{ni} of the reduced list. Then two different procedures are known:

- Algorithm 2 [46] emerges from Algorithm 1 by keeping track of $\overline{\mathcal{L}}_n$ instead of $\mathcal{L}_n$ (and thus having U_n available at each iteration).
- Algorithm 3 [30,31] is like Algorithm 2, but the reduced lists $\overline{\mathcal{L}}_n$ are ordered with respect to the age or the widths of the boxes.

Variants of the three prototypes occur if the ordering of the lists and the bisection directions are changed, cf. the next two sections.

Convergence Properties of the Prototype Algorithms

The results presented in this section are proven in [77,86,89].

Let us first consider Algorithm 1. As in the previous section, we denote the leading pairs of Algorithm 1 by (X_n, y_n). One can show that

a) $w(X_n) \to 0$ as $n \to \infty$.

This fact seems to be self-evident but it is not. For example, small modifications of the basic algorithm do not satisfy a) as is the case with the cyclic bisection method [74]. From the assumption

$$w(F(Y)) - w(\overline{f}(Y)) \to 0 \quad \text{as } w(Y) \to 0 \quad (Y \in I(X)) \tag{7}$$

it follows that

b)

$$\begin{aligned} &y_n \le f^* \quad \text{for any } n,\\ &y_n \to f^* \quad \text{as } n \to \infty,\\ &f^* - y_n \le w(F(X_n)) \text{ (error estimate).} \end{aligned}$$

Assumption (7) is not very restrictive. It is almost always satisfied if natural interval extensions are used. However, (7) does not imply continuity, Lipschitz condition on f, etc. Let F now satisfy

$$w(F(Y)) \to 0 \quad \text{as } w(Y) \to 0. \tag{8}$$

Clearly, (8) implies (7) and the continuity of f. Then

c) $w(F(X_n)) \to 0$ as $n \to \infty$

(that is, the error estimate tends to 0 and can thus be used for termination criteria),

d) each accumulation point of the sequence (X_n) is a global minimizer.

The convergence order of the approach $y_n \to f^*$ is described by the following two results:

e) Let any $\alpha > 0$ and any converging sequence of reals be given. Then, to any f, there exists an inclusion function of order α for which (y_n) converges slower than the given sequence.

This result indicates that the convergence can be arbitrarily slow and that no worst case exists, which is usually taken in order to establish formulas for the convergence speed or convergence order. If, however, only isotone inclusion functions (F is called *isotone* if $Y \subseteq Z$ implies $F(Y) \subseteq F(Z)$) are considered then the following estimate of the convergence speed is valid. Practically this estimate characterizes the complete convergence theory since it is always possible to find isotone inclusion functions with small effort.

f) If F is isotone and of order α, then

$$f^* - y_n = O(n^{\frac{-\alpha}{m}}).$$

In [16] some variants of this assertion are proven.

Algorithms 2 and 3 have nearly the same behavior as Algorithm 1 if the convergence to f^* is considered. Their properties with respect to a determination of X^* are as follows:

Let (U_n) be the sequence of unions produced by Algorithm 2. If (8) is assumed, then

g) the sequence (U_n) is nested and converges (with respect to the Hausdorff metrics for compact sets) to a superset $D \supseteq X^*$. The probability, that D is not equal to X^* is zero, however.

Let now (U_n) be the sequence of unions produced by Algorithm 3. If (8) is assumed, then

h) the sequence (U_n) is nested and converges to X^*.

Therefore, Hansen's Algorithm 3 is the only one of the three which features a satisfactory and guaranteed convergence to f^* and X^*. This algorithm will therefore play the main role in our further considerations.

Accelerating and Related Devices

Algorithm 3 and its predecessors which we have treated so far are based on the *exhaustion principle*, that is, the principle of removing areas (subboxes of X) which cannot contain a global minimizer. In the same manner we realize that the *branch and bound principle* forms the

overlying structure, that is, areas are processed which have the largest chance to contain a global minimizer. This is a time-consuming process and it is therefore important to combine the principle mentioned with techniques for speeding up the computations. In this section we deal with only a few of these techniques in order to demonstrate how they may be combined with the basic algorithm. Much research is done in order to find an optimal combination of the basic algorithms and the acceleration devices, cf. for instance, [26,31,33,48,49,93,95,96].

In the following we give an overview of several acceleration devices and other tools that are used to improve the computational efficiency of unconstrained interval based global optimization. Most of them are also developed for constrained optimization.

The Monotonicity Test

It can be applied if f is differentiable and if an inclusion function for ∇f is available ([30,31,72]). It allows one to automatically recognize that f is strictly monotone in one of the variables in the subbox $Y \subseteq X$ on which the algorithm is focusing. Then Y can be discarded from the list if Y lies in the interior of X or otherwise Y can be replaced by an edge piece of Y. This can be done since the parts removed do not contain a global minimizer. I. e., let G_i be an inclusion function of $\partial f/\partial x_i$ for $i = 1, \ldots, m$. If now $0 \notin G_i(Y)$ just for one index i, then f is strictly monotone in the variable x_i over Y such that Y can be discarded or replaced by an edge piece as mentioned before. (For the application of the test it is already sufficient that f is *locally Lipschitz*, cf. the next section).

The Interval Newton Method

If f is twice continuously differentiable and if an inclusion function for the Hessian matrix function, f, exists then the interval Newton algorithm can be applied to f' in order to get boxes that contain all zeros of f'. Together with the monotonicity test, the interval Newton algorithm counts as one of the most effective tools for solving optimization problems. The main advantage is not only the localization of the zeros of f', but also a computationally very successful performance. This is based on the properties mentioned in the section on 'interval Newton methods' which result in reducing or splitting the search area. Finally, the contraction shows quadratic convergence under reasonable conditions.

Interval Newton methods can be applied in two different manners:

i) The method is applied to f' in X (necessarily combined with some splittings of the search area) until all critical points of f are included in sufficiently small boxes Z, for example where $w(Z) < \epsilon$. Then the search for the global minimizers is restricted to these remaining boxes Z and to the facet of X. This approach is, however, not too effective since these zeros can be saddle points, local maximizers, or even local but not global minimizers. Hence the following procedure is used generally:

ii) Each iteration of the optimization algorithm is combined with the monotonicity test and one or two interval Newton iterations. I. e., after having X bisected into the subboxes V_1 and V_2, cf. Step 5 of Algorithm 3, the midpoint test, the monotonicity test and one interval Newton iteration is applied to V_1 and V_2 in order to diminish the size of V_1, V_2 or to discard them. This procedure avoids superfluous and costly interval Newton iterations in boxes in which f is strictly monotone or which have too large function values.

The interval Newton can be improved by using slopes whenever possible, cf. [79]. See also 'use of good inclusion functions' below.

Finding a Function Value as Small as Possible

The smaller the smallest known or computed function value is at the nth iteration the more effective is the midpoint test, that is, boxes are removed earlier than without these values.

There are many possible techniques for getting lower function values such as statistical and line search methods, bundle methods (line search in the nonsmooth case), descent methods, Newton-like methods, where the application of the methods depends on the differentiability of the objective function. Many of these variations lead to so-called globally convergent methods. This does not mean that a global solution is found, however, it does mean that a local solution is always found. Good results in finding small function values have been attained with generating a not very dense set

of points and to use them as starting points for the globally convergent methods mentioned above.

Bisections

The computations can be accelerated by a good choice of

A) the next box of the list to be bisected; and
B) the bisection direction of that box.

These two topics did not draw too much attention in the first years of interval optimization. They were considered as a tiresome task for completing the algorithms rather than topics important for the success of the algorithms. Meanwhile it has been recognized how important the right choice of box and bisection direction is for keeping computation time and costs low. The right choice of bisection direction is equally important for the global zero search of systems of functions.

Strategies for *choosing the next box* include uniform subdivision [71], bisecting a box which has a minimal lower bound, cf. Algorithms 1 and 2, bisecting that box which has been longest on the list [31], bisecting a box which has maximum width [31], last in-first out [79], that is, the youngest boxes are always processed first which keeps the list length short under certain circumstances.

When a box has been selected for getting bisected one has to *choose the bisection direction*. Historically, the first three criteria were uniform subdivision [71] (that is, bisections were done in all m directions), cyclic bisection [74] (that is, the bisection directions change cyclically, i. e., the first box gets bisected normal to the first coordinate direction, the second normal to the second coordinate direction, etc.), and bisection normal to one of the longest box edges [31].

It turned out that using the box width as the only criterion for deciding the bisection direction could be very ineffective. For a typical example, see [91]. The conclusion from such examples is that the choice of the bisection direction should consider the behavior of the function f over the box as well. Hence, formulas for deciding a bisection direction are built up using bounds for the box width of the objective function and bounds for the first and second partial derivatives. Natural interval extensions of noninterval scaling formulas are also used. Our own tests and experiments show that an optimum bisection strategy does not exist and that it is reasonable to use several bisection strategies each pursuing another heuristic aim. This led to systematic investigations of bisections and also trisections by several authors, mainly [15,18,53,54,96,97,106]. For further strategies see [93], where also a survey of convergence properties of some of the strategies can be found, and [92].

Use of Good Inclusion Functions, Slope Arithmetic

The better the inclusion functions are, the more effective are the tests like midpoint test, monotonicity test, etc., cf. for instance, [88]. The derivatives can frequently be replaced by slopes which leads to inclusions with smaller width, There is also an automatic slope arithmetic available which is comparable to automatic differentiation, cf. 'autoimatic differentiation' below. The interested reader is referred to [1,40,61,62,79,88,101].

The Nonconvexity Test

The aim of this test is to verify that the objective function is nowhere convex in some subbox $Y \in I(X)$ by computationally checking whether the Hessian of the objective function does not satisfy some standard conditions of convexity. Then the interior of Y cannot contain a minimizer. $f \in C^2$ is assumed. The first such test seems to date back to [64].

Thin Evaluation of the Hessian Matrix

If interval Newton steps are incorporated they will be applied to $\phi = f'$ where the matrix $J_\phi(Y) = H_f(Y)$ is required and $H_f(Y)$ is the natural interval extension of the Hessian matrix. By certain rearrangements of $H_f(Y)$ and a special method of getting an interval extension, where not all real entries are replaced by intervals, it is possible to obtain an interval matrix which is thinner, hence better (cf. 'use of good inclusion functions' above) than $H_f(Y)$. A detailed discussion and formulas can be found in [33,90].

Constraint Logic Programming

(also known as *constraint solving*) involves techniques where, among others, equations (for example, the Karush–Kuhn–Tucker or F. John conditions) are primarily not evaluated numerically but seen as constraints for or as relations between the variables (a con-

cept overtaken from artificial intelligence). The relation is then used to shrink the search domain. For example, the equation (constraint) $y - x^2 = 0$ immediately enables the halfspace defined by $y < 0$ to be removed from the working area. There are several methods based on that idea which are best embedded in appropriate languages, where symbolic manipulation such as PROLOG is available. For example, a method called *relational interval algebra* is embedded in the computer language CLP having PROLOG as metalanguage (cf. [7] or [83]). In this connection it is also opportune to automatically add redundant constraints in order to accelerate the computations (see, for example, [5] or [82]).

Another approach is called *branch and prune* ([41]). The *pruning* concept aims to shrink the search area by several tests. The crucial property which is searched for is the so-called *box consistency* which has been introduced in [6] and is also known in connection with discrete combinatorial search problems. The box consistency is primarily used to indicate the existence of solutions in the considered subarea and is some kind of a substitution of interval Newton techniques. An interesting means for proving box consistency is the *bound consistency* which requires the checking of the facets of the box instead of the box itself. The branch and prune algorithm is embedded in NUMERICA, which is designed as a modeling language for global optimization and related problems, cf. [41].

There are several other approaches that are based on constrained logical programming such as the use of relational manipulations or of set-valued operations, see for example, [3] or [45] and the references listed there.

Automatic Differentiation

This technique seems to go back to [108]. It helps to reduce costs when computing derivatives or their inclusion functions, or expressions like $(x - c)^{\mathsf{T}} f'(c)$, $(Y - c) f'(Y)$, $(x - c)\, f'(Y)$, $(Y - c)^{\mathsf{T}} f'(Y)(Y - c)$, etc., where $x, c \in X$, $Y \in I(X)$. There are two modes of automatic differentiation, a forward and a reverse. Both modes use recursive techniques for evaluating function values and chain rules of differentiation. In the forward mode all intermediate values of the function are simultaneously determined with the corresponding intermediate values of derivative, Hessian, etc, and all these intermediate values are computed from values calculated in former steps. The reverse mode requires some structural planing of the formulas similar to the construction of Kantorovich graphs of functional expressions, where a new variable is assigned to each node. The differentiation finally starts backwards from the function in dependency of the variables introduced. Both modes have advantages.

Our own experiences, however, show that in case of interval expressions like $(Y - c)\, f'(Y)$ or in case of computing generalized gradients, it is not always wise to use automatic differentiation. The reason is that in such cases information about dependencies between intervals can be lost so that the widths of the resulting interval values increase unnecessarily.

For a detailed description of automatic differentiation cf. for instance, [23,24,29,84].

Parallel Computations

for global optimization were investigated and implemented primarily by [8,12,21,22].

Global Optimization Over Unbounded Domains and Nonsmooth Optimization

Global Optimization over Unbounded Domains

Almost all methods for solving global optimization problems need the assumption that a bounded domain which contains the solution points is known. The boundedness is necessary for the numerical computation as well as for guaranteeing the convergence properties. If an a-priori box X as search area for the global solutions is not known, it is possible to extend the previous algorithms, especially Algorithm 3 in such a manner, that they can operate over unbounded boxes as well cf. [90,94]. It is not even necessary, to change the algorithms formally, one only has to define midpoint and width of infinite intervals (both values have to be finite) and an arithmetic for infinite intervals. This arithmetic should provide intervals with minimal widths in order to get reasonable inclusion functions. It would go to far to present this arithmetic here, but a short example could be illustrative: This arithmetic assigns to the quotient $[0, \infty]/[0, \infty]$ the value $[0, \infty]$, whereas by an arithmetic which is called Kahan–Novea–Ratz arithmetic in [55] the value $[-\infty, \infty]$ results. Most of the convergence properties of the section on ‘convergence

properties of the prototype algorithms' remain valid under slight modifications of the assumptions since one can interpret the algorithms as algorithms in $(\overline{\mathbb{R}})^m$ where $\overline{\mathbb{R}}$ is the two-point compactification of the real axis, $\mathbf{R}$. Thus compact intervals are generated by the algorithms and that is all one needs for convergence proofs. For details see [90,94] or the survey in [93].

Nonsmooth Optimization

A broad spectrum of mathematical programming problems can be reduced to nondifferentiable problems without constraints or with simple constraints. The use of exact nonsmooth penalty functions in problems of nonlinear programming, maximum functions to estimate discrepancies in constraints, piecewise smooth approximation of technical-economic characteristics in practical problems of optimal planning and design, minimax compromise function in problems of multicriterion optimization, all generate problems of nonsmooth optimization. Thus, the objective function, f, of the optimization problem may look like $f(x) = \max\{f_1(x), \ldots, f_n(x)\}$ where $f_i \in C^1$, or like $f(x) = \mu f_0(x) + \sum_{i=1}^{k} \max(0, f_i(x))$ which is a typical objective function arising from penalty methods where $f_0, f_i \in C^1$ and $\mu > 0$ is a (reciprocal) penalty factor.

Interval methods have no difficulties at all to handle nonsmooth problems, a fact which was discovered in [87] and rediscovered in [55] with great emphasis. The construction of inclusion functions does not depend at all on the smoothness of a function. The application of monotonicity tests and other devices where gradients are used (for instance, local noninterval methods, cf. 'finding a function value as small as possible' in the section on 'accellerating and related devices') is still possible as long as the function is *locally Lipschitz*, which means that, at any argument, x of the function, f, an open neighborhood of x, say U_x, exists in which f satisfies a Lipschitz condition. It follows by a theorem of H. Rademacher that f is differentiable almost everywhere in U_x. Let Ω be the set of points in U_x at which f is not differentiable, and let S be any other set of Lebesque measure 0. Then the *generalized gradient* (also called *subdifferential*) of f at x is defined as

$$\partial f(x) = \operatorname{conv}\left\{ \lim_{n\to\infty} \nabla f(x_n) \colon\ x_n \to x, x_n \notin S \cup \Omega \right\}$$

where conv denotes the convex hull, cf. [14]. Let $(x, y) \subseteq \mathbf{R}^m$ denote the open line segment between x and y. A theorem of G. Lebourg says that, if $y \in U_x$ with $(x, y) \subseteq U_x$ is given then some $u \in (x, y)$ exists such that

$$f(y) - f(x) \in (y - x)^{\top} \partial f(u). \tag{9}$$

Locally, (9) can be approximated by means of the Lipschitz constant. Globally, (9) can be used to find inclusion functions of f of a mean value type explicitly: If $G(Y)$ is a (not necessarily bounded) box that contains $\partial f(u)$ for any $u \in Y$, then

$$F(Y) = f(c) + (Y - c)^{\top} G(Y) \quad \text{for } Y \in I(X),$$

where c denotes the midpoint of Y (any other point of Y may also be chosen), is an inclusion function of f and appropriate for its use in the Algorithms 1 to 3. Furthermore, $G(Y)$ can be used for the *monotonicity test*: If only one component of $G(Y)$ does not contain zero, then f is strictly monotone with respect to the corresponding direction.

Algorithms 1 to 3 as well as the monotonicity test therefore can be applied to problem (1) without modifications, if the objective function of f is locally Lipschitz. It is, however, only possible to apply the interval Newton algorithm for a very restricted class of functions since second 'derivatives' of locally Lipschitz functions are are not yet explored satisfactory. With the aid of the infinite interval arithmetic mentioned in the subsection above one can admit also unbounded subdifferentials and handle them. For the construction of inclusion functions of the objective function and the subdifferential and for numerical tests (with bounded and unbounded search areas) see, for instance, [27,28,55,87,90,94]. For further results in connection with estimates of the penalty factor see [111].

Constrained Optimization

The principles which were developed in the previous sections are also useful for constrained problems, that is,

$$\min_{x \in M} f(x) \tag{10}$$

where $M \subseteq \mathbf{R}^m$ means the feasible set defined by constraints

$$g_i(x) \le 0, \quad i = 1, \ldots, k,$$
$$h_j(x) = 0, \quad j = 1, \ldots, s.$$

For simplicity, we assume that $M \subseteq X$ for some $X \in I^m$ and that the functions f, g_i and h_j are defined on X. For a successful treatment of problem (10) we need inclusion functions F, G_i and H_j of f, g_i and h_j, respectively which satisfy (8) and which have the property that

$$\begin{cases} w(G_i(Y)) \to 0 & \text{as } w(Y) \to 0, \\ w(H_j(Y)) \to 0 & \text{as } w(Y) \to 0 \end{cases} \tag{11}$$

for $i = 1, \ldots, k$, $j = 1, \ldots, s$, and $Y \in I(X)$.

Then a very effective means of interval arithmetic is the *infeasibility test* which is applicable to any $Y \in I(X)$: If either

$$G_i(Y) > 0 \quad \text{for some } i \in \{1, \ldots, k\}$$

or if

$$0 \notin H_j(Y) \quad \text{for some } j \in \{1, \ldots, s\}$$

then all points of Y are infeasible. (The notation $[a, b] > 0$ or $[a, b] \leq 0$ is used to indicate that $a > 0$ or $b \leq 0$ holds, respectively.) Hence the box Y can never contain a solution of (10) such that Y can be discarded from any procedure for solving (10). Conversely, if

$$G_i(Y) \leq 0 \quad \text{for } i = 1, \ldots, k,$$

and

$$H_j(Y) = 0 \quad \text{for } j = 1, \ldots, s,$$

then all points of Y are feasible (*feasibility test*). This is due to the inclusion principle, (3), by which $a \in Y$ implies $g_i(a) \in G_i(Y)$ as well as $h_j(a) \in H_j(Y)$ for all indices i and j, that is, $g_i(a) \leq 0$ and $h_j(a) = 0$ for all i and j. This gives, in fact, the guarantee that every point $a \in Y$ is feasible. However, if equality constraints are present in (10) it is extremely unlikely that conditions like $H_j(Y) = 0$ are satisfied such that the feasibility test is rather an academic tool if $s > 0$.

There are principally two main possibilities for solving the constrained problem. The first possibility is to transform the problem to an unconstrained problem within a penalty setup and apply the methods of the former sections together with the feasibility, respectively infeasibility, test in order to have the guarantee to be in M or to discard infeasible areas. The second possibility is a direct approach where Algorithm 3 is enriched by feasibility and infeasibility test and adapted to handle the constrained case. We will now give a brief discussion of these possibilities.

The Penalty Approach

There are two kinds of penalty functions which are usually preferred. The first one is the so-called L_1-exact penalty function, $\phi(x) = \mu f(x) + \sum_{i=1}^{k} \max(0, g_i(x)) + \sum_{j=1}^{s} |h_j(x)|$, cf. also the subsection 'nonsmooth optimization' in the previous section. The second one, already introduced by R. Courant, is defined as $\psi(x) = \mu f(x) + \sum_{i=1}^{k} \max(0, g_i(x))^2 + \sum_{j=1}^{s} (h_j(x))^2$. In both cases, μ is a penalty factor. For details, and how penalty methods are applied to solve constrained optimization problems, cf. [25]. (Augmented Lagrangian functions could also be taken for the penalty approach.) When locally solving (10) with standard noninterval methods, ϕ has the advantage that there exists a μ so that the local minimizers of ϕ are also local minimizers of (10), but has the disadvantage of being nonsmooth. The use of ψ has the advantage of dealing with a smooth function (provided f and the constraints are smooth), but the disadvantage, that the minimizers of ψ might attain the solutions of (10) only asymptotically as μ tends to zero. If f and the constraint functions are smooth there exists a value μ in both cases of penalty functions so that the global minimizers of ϕ and ψ are also global solutions of (10) when solving (10) with interval methods. The explicit determination of this number μ is still under investigation, cf. [111]. On the other hand, the knowledge of the value is not necessary if only convergence is expected because infeasible areas are removed by the infeasible test which has to be incorporated in the prototype algorithm such as Algorithm 3. The knowledge of the value of μ accelerates the computation. A further discussion would be too extensive for this article.

The Direct Approach

Algorithm 3 is also appropriate as a base algorithm for dealing with the constrained case. In order to consider the constraints, one just has to add the feasibility and infeasibility test and to apply the latter test as a box deleting device to the boxes V_1 and V_2 after Step 5 of the algorithm. If it turns out that the box is feasible, it should be marked as *feasible* by a flag or a Boolean value. The remaining boxes of the list are *indeterminate*, that is, the tests executed up to the current state of the computation have not yet been able to decide whether the box is feasible or not. It can happen that boxes V_i which are feasible (respectively, infeasible) are not rec-

ognized as feasible (respectively, infeasible) by the feasibility (respectively, infeasibility) test. This is due to the excess width (see the section 'interval arithmetic' above) which, for instance, can cause that $0 \in G_i(V)$ for some box V occurs even if $0 < \overline{g_i}(V)$ holds. The continued processing of the indeterminate boxes of the list by the steps of Algorithm 3, however, reduces the box widths to zero so that their excess widths also tend to zero as long as (11) is assumed. This implies also that the *union of the boxes of* $\mathcal{L}$ *tends, as the computation proceeds, to* M with respect to the Hausdorff metrics (cf. [90]) if one dropped the midpoint test.

The *midpoint test* itself helps to discard feasible as well as indeterminate boxes which contain no global minimizer. The execution is as in the unconstrained case: Let f_n be the lowest function value which has been calculated up to the completion of the list $\mathcal{L}_n$. Then all pairs (Z_{ni}, z_{ni}) of $\mathcal{L}$ are discarded that satisfy

$$f_n < z_{ni}.$$

It is important for the correctness of the algorithm that only function values of points $x \in M$ are admitted. Hence, if x is taken from a feasible box of the list, x is certainly feasible. If the list contains only indeterminate boxes no direct access to feasible points of M is at hand. This is regularly the case if equality constraints are present. But without the knowledge of points $x \in M$ the midpoint test cannot be executed. Two possibilities are known for overcoming this hurdle. The first possibility is the so-called ϵ-inflation. It accepts that the constraints are satisfied within a tolerance of ϵ. If ϵ-inflation, which is widely used in noninterval computations, is applied then the reliability of the computation is lost. Thus this possibility is avoided as far as possible in interval computations.

The second possibility to overcome the difficulties arising by equality constraints is based on the application of Moore's *test for the existence of solutions of equations* [73]. Hansen and G.W. Walster [38] were the first who suggested to apply this test to constrained optimization. It is used in the following manner: If Y is an indetermined box under processing and one looks for a feasible point in Y, the equality constraints and the inequality constraints which are active with respect to Y are combined to a system of equations. Then interval Newton iterations are applied to this system in Y, not to solve the system but only to prove the existence of a solution within Y by a contraction of the Newton operator. Then all boxes (Z_{ni}, z_{ni}) of the list (feasible or indetermined) can be discarded that satisfy $\max f(Y) < z_{ni}$ since $\max f(Y)$ is an upper bound for a function value of a feasible point. If the system of equations shows more variables than equations, some variables are replaced by constants.

The existence test in Y is best done in the following manner: Apply a local simple noninterval optimization algorithm to the objective function $\psi(x) = \sum_{i=1}^{k} \max(0, g_i(x))^2 + \sum_{j=1}^{s} (h_j(x))^2$ (this is the Courant penalty function for $f(x) = 0$, cf. the first subsection in this section) in order to come near a feasible point, say c. Put a small box which has to lie in Y around c and apply the existence test to the system in the box (eventually cleaned up by meanwhile inactive inequality constraints). If the test is positive, the existence of a feasible point in the box and hence in Y is guaranteed. However, it is not at all a proof that Y is infeasible if the test fails.

An improvement is due to [58] where techniques to search for points $c \in Y$ are designed so that the chances of finding a nearby feasible point is optimal. Also the number of variables can be larger than the number of equations in the underlying system.

The convergence of the union of the list boxes to the set of global minimizers can be shown if the test for the existence of feasible points is applied systematically and successfully to the boxes of the list (as far as they are indeterminate). Other convergence proofs can be found in [8,90].

In order to not only get a convergent but also a fast convergent algorithm, *acceleration devices and related techniques* are again extremely important for practical computations. Well-known techniques are the following:

i) *Interval Newton iterations*. They are applied to the F. John conditions to enclose the stationary points, similar as to the unconstrained case. Since the number of equations exceeds the number of variables by 1 in the F. John conditions, an additional equation is added which does not influence obtaining the stationary points, cf. [39]. As in the unconstrained case, the interval Newton iterations are not executed until termination, but they merge with the steps of the optimization algorithm. Again, if an iteration shows the existence of a F. John point, it is

a feasible point and can be used for the midpoint test.

In contrast to several authors we do not count the interval Newton iterations as basic steps of an optimization algorithm, since they do not influence the convergence, only the convergence speed of the algorithm.

ii) *Monotonicity and nonconvexity test*, cf. the section on 'accellerating and related devices'. These tests are best applied to feasible boxes, but there are also exemptions of this suggestion, cf. [70,93]. Also *linearization of the constraints* supplementing the infeasibility test is used [33].

iii) *Good inclusion functions*, *slope arithmetic*, *automatic differentiation*, *bisections*, *parallel algorithms*, *constrained logic programming*, are already mentioned in the section on 'accellerating and related devices'.

iv) *Local search devices*. In order to get soon function values of feasible points, local noninterval optimization procedures are applied to the function ψ, as defined above, related to the current box until one reaches a feasible point or until one is near a feasible point. In the latter case the existence test has to be applied at this approximation w. r. to a small surrounding box in order to guarantee the existence. In case of full-dimensional feasible domains, the local search can be continued with ϕ instead ψ, but one has to take care not to leave the domain M.

It turned out that the performance of the algorithm was greatly influenced by how the steps of the optimization algorithm and the acceleration devices were combined. Several investigations dealing with this matter have been done, cf. for example, [8,18,19,26,38,39,49,56,95,100,109,110].

Applications

Global optimization using interval arithmetic has been applied to optimization problems in a variety of science, engineering and social science areas. Below we briefly describe representative examples from several areas.

Chemistry and Chemical Engineering

Many optimization problems in the fields of chemistry and chemical engineering can be investigated effectively using the tools described in the previous sections.

As a first example we consider the diagram of a chemical process showing the processing units and the connections between them. This depicts the flow of chemical components through the system and it is often referred to as process flowsheeting and the associated optimization problems are called process flowsheeting problems. They require the solution of large sparse differential-algebraic systems. In [99] a parallel interval Newton algorithm combined with bisection techniques is applied to solve a number of simple problems of this type where the parallelization is required in order to complete the computations within a reasonable timeframe.

The reliable prediction of phase stability in a chemical process simulation has been considered by [42,43]. It is pointed out that conventional methods that are initialization dependent may converge to trivial or nonphysical solutions or to a nonglobal local minimum. It is furthermore shown that these difficulties can be avoided using a cubic equation of the state model combined with interval tools. Their technique is initialization independent and it solves the phase stability problem with complete reliability. In [44] the approach is further developed with respect to computational efficiency. An enhanced method is presented based on sharpening the range of the interval functions that occur in the algorithm. It is shown that the computation time can be reduced by nearly an order of magnitude in some cases.

The paper [69] addresses the problem of minimizing the Gibbs free energy in the m-component multiphase chemical and phase equilibrium problem involving different thermodynamic models. The solution method is based on the tangent-plane criterion of Gibbs and it is reduced to a finite sequence of local optimization steps in $K(m-1)$-dimensional space where $K \leq m$ is the number of phases, and global optimization steps in $(m-1)$-dimensional space. The algorithm developed in the lower-dimensional space uses techniques from interval analysis. Some promising results are reported for the algorithm. A parallel interval algorithm for the problem was developed in [9]. Chemists performing photoelectron spectroscopy collide photons with atoms or molecules. These collisions result in the ejections of photoelectrons. The chemist is

left with a photoelectron spectrum which is a plot of the number of photoelectrons ejected as a function of the kinetic energy of the photoelectron. A typical spectrum consists of a number of peaks. The chemist would like to resolve the individual peaks in the spectrum. In the paper [76] a test problem is constructed as a sum of two Gaussian functions involving a number of parameters. These parameters are found using interval techniques of global optimization.

Physics, Electronics and Mechanical Engineering

A wide variety of problems in physics, electronics and mechanical engineering can be formulated as optimization problems amenable to the techniques described in the previous sections. We provide some representative examples below.

An early application is found in [68] who applies interval global optimization to electronic switching systems for efficiency reasons.

In [10] interval global optimization is used to determine rigorous bounds on Taylor maps of general optical systems. It is also pointed out that stability for storage rings and other weakly nonlinear systems can be guaranteed using their developments.

In [78] Hansen's method is applied to a demagnifying system for electron beam lithography device for finding all real minimizers of a real valued objective function of several variables.

Computer-aided simulation tools for liquid crystal displays have been developed in recent years. These tools calculate the molecule orientation of the liquid crystal material by minimizing an energy function. The results of such simulations are used to optimize notebook computer displays. In the paper [80] interval global optimization is used to calculate all minimizing molecule configurations.

Interval global optimization is applied to the optimal design of a flat composite plate and a composite stiffened panel structure in [63]. The methodology is to generate a feasible suboptimal interval which is used to examine the manufacturing tolerance in the design optimization.

Economics

Global optimization using interval analysis has also found applications in economics. Two examples are presented below.

A model of copyable products such as software is considered by [107] who based their model on the model developed by I.E. Besanko and W.L. Winston [11]. In the paper [107] this model is solved for a globally optimal result using an interval branch and bound method.

In [50] another problem in economics is considered. The problem is to minimize an econometric function

$$\sum (y_t - \widehat{\beta}_1 - \widehat{\beta}_2 X_{t2} - \widehat{\beta}_2^2 X_{t3})^2$$

where the data are artificially generated for the variables. Several tests are performed and it is shown that interval methods are competitive with other methods such as simulated annealing.

See also

- ▶ αBB Algorithm
- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Point and Interval Taylor Operators
- ▶ Bounding Derivative Ranges
- ▶ Continuous Global Optimization: Applications
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Global Optimization in the Analysis and Management of Environmental Systems
- ▶ Global Optimization: Application to Phase Equilibrium Problems
- ▶ Global Optimization in Batch Design Under Uncertainty
- ▶ Global Optimization in Generalized Geometric Programming
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization in Phase and Chemical Reaction Equilibrium
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Interval Analysis: Differential Equations
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Interval Analysis: Intermediate Terms
- ▶ Interval Analysis: Nondifferentiable Problems
- ▶ Interval Analysis: Parallel Methods for Global Optimization

- ▶ Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- ▶ Interval Analysis: Systems of Nonlinear Equations
- ▶ Interval Analysis: Unconstrained and Constrained Optimization
- ▶ Interval Analysis: Verifying Feasibility
- ▶ Interval Constraints
- ▶ Interval Fixed Point Theory
- ▶ Interval Linear Systems
- ▶ Interval Newton Methods
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Global Optimization with αBB
- ▶ Mixed Integer Nonlinear Programming
- ▶ Smooth Nonlinear Nonconvex Optimization

References

1. Alefeld G, Herzberger J (1983) Introduction to interval computations. Acad. Press, New York
2. Asaithambi NS, Shen Z, Moore RE (1982) On computing the range of values. Computing 28:225–237
3. Babichev AB, Kadyrova AB, Kashevarova TP, Leshchenko AS, Semenov AL (1993) UniCalc, a novel approach to solving systems of algebraic equations. Interval Comput 2:29–47, (Special Issue).
4. Bauch H, Jahn K-U, Oelschlägel D, Süsse H, Wiebigke V (1987) Intervallmathematik: Theorie und Anwendungen. Teubner, Leipzig
5. Benhamou F, Granvilliers L (1997) Automatic generation of numerical redundancies for non-linear constraint solving. Reliable Computing 3:335–344
6. Benhamou F, McAllester D, Van Hentenryck P (1994) CLP(intervals) revisited. In: Logic Programming, Proc. 1994 Internat. Symp., Ithaca, NY, MIT, Cambridge, MA, pp 124–138
7. Benhamou F, Older WJ (1997) Applying interval arithmetic to real, integer and boolean constraints. J Logic Programming 32:1–24
8. Berner S (1995) Ein paralleles Verfahren zur verifizierten globalen Optimierung. Diss. Univ. Wuppertal,).
9. Berner S, McKinnon KIM, Millar C (1997) A parallel algorithm for the global minimization of Gibbs free energy. Ann Oper Res 90:271–292
10. Berz M (1997) Differential algebras with remainder and rigorous proofs of long-term stability. AIP Conf. Proc. 391, 221–228
11. Besanko IE, Winston WL (1989) Copyright protection and intertemporal pricing: does unauthorized copying lead to higher prices? Working Paper School Business Indiana Univ.
12. Caprani O, Godthaab B, Madsen K (1993) Use of a real-valued local minimum in parallel interval global optimization. Interval Comput 2:71–82
13. Caprani O, Madsen K (1979) Interval methods for global optimization. In: Vogt WG, Mickle MH (eds) Modeling and Simulation, 10:3, Instrument Soc. Amer., Pittsburgh, PA, pp 589–593
14. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
15. Csallner AE, Csendes T (1995) Convergence speed of interval methods for global optimization and the joint effects of algorithmic modifications. Talk given at SCAN'95, Wuppertal,
16. Csallner AE, Csendes T (1996) The convergence speed of interval methods for global optimization. Comput Math Appl 31:173–178
17. Csendes T, Pinter J (1993) A new interval method for locating the boundary of level sets. Internat J Comput Math 49:53–59
18. Csendes T, Ratz D (1997) Subdivision direction selection in interval methods for global optimization. SIAM J Numer Anal 34:922–938
19. Csendes T, Zabinsky ZB, Kristinsdottir BP (1995) Constructing large feasible suboptimal intervals for constrained linear optimization. Ann Oper Res 58:279–293
20. Dussel R (1972) Einschliessung des Minimalpunktes einer streng konvexen Funktion auf einem n-dimensionalen Quader. Diss. Univ. Karlsruhe
21. Eriksson J (1991) Parallel global optimization using interval analysis. PhD Thesis Univ. Umeåa,).
22. Eriksson J, Lindstroem P (1995) A parallel interval method implementation for global optimization using dynamic load balancing. Reliable Computing 1:77–92
23. Fischer H (1993) Automatic differentiation and applications. In: Adams E, Kulisch U (eds) Scientific Computing with Automatic Result Verification. Acad. Press, New York, pp 105–142
24. Fischer H (1995) Automatisches Differenzieren. In: Herzberger J (ed) Wissenschaftliches Rechnen: Eine Einführung in das Scientific Computing. Akad. Verlag, Berlin, pp 53–104
25. Fletcher R (1987) Practical methods of optimization. second, Wiley, New York
26. Goos A, Ratz D (1997) Praktische Realisierung und Test eines Verifikationsverfahrens zur Lösung globaler Optimierungsprobleme mit Ungleichungsnebenbedingungen, Inst. Angew. Math. Univ. Karlsruhe, Karlsruhe
27. Görges C, Ratschek H (1997) ϵ-Interval methods in nonsmooth optimization. In: Gritzmann P Horst R Sachs E, Tichatschke R (eds) Recent Advances in Optimization. Springer, Berlin, pp 75–89
28. Görges C, Ratschek H (1999) Global interval methods for local nonsmooth optimization. J Global Optim 14:157–179

29. Griewank A and Corliss G (eds) (1991) Automatic differentiation of algorithms: Theory, implementation, and application. SIAM, Philadelphia
30. Hansen ER (1979) Global optimization using interval analysis: the one-dimensional case. J Optim Th Appl 29:331–344
31. Hansen ER (1980) Global optimization using interval analysis: the multidimensional case. Numer Math 34:247–270
32. Hansen ER (1988) An overview of global optimization using interval analysis. In: Moore RE (ed) Reliability in Computing: The Role of Interval Methods in Scientific Computing. Acad. Press, New York, pp 289–307
33. Hansen ER (1992) Global optimization using interval analysis. M. Dekker, New York
34. Hansen ER, Greenberg RI (1983) An interval Newton method. Appl Math Comput 12:89–98
35. Hansen ER, Sengupta S (1980) Global constrained optimization using interval analysis. In: Nickel K (ed) Interval Mathematics. Acad. Press, New York, 25–47
36. Hansen ER, Sengupta S (1981) Bounding solutions of systems of equations using interval analysis. BIT 21:203–211
37. Hansen ER, Smith RR (1967) Interval arithmetic in matrix computations, Part II. SIAM J Numer Anal 4:1–9
38. Hansen ER, Walster GW (1987) Nonlinear equations and optimization. Preprint,).
39. Hansen ER, Walster GW (1993) Bounds for Lagrange multipliers and optimal points. Comput Math Appl 25:59–69
40. Hansen P, Jaumard B, Xiong J (1992) The cord-slope form of Taylor's expansion in univariate global optimization. JOTA 80:441–464
41. Van Hentenryck P, Michel L, Deville Y (1997) Numerica: A modeling language for global optimization. MIT, Cambridge, MA
42. Hua JZ (1997) Interval methods for reliable computations of phase equilibrium from equation of state models. PhD Thesis Univ. Illinois at Urbana-Champaign
43. Hua JZ, Brennecke JF, Stadtherr MA (1996) Reliable phase stability analysis for cubic equation of state models. Comput Chem Eng 20:S395–S400
44. Hua JZ, Brennecke JF, Stadtherr MA (1998) Enhanced interval analysis for phase stability: Cubic equation of state models. Industr Eng Chem Res 37:1519–1527
45. Hyvönen E, De Pascale S (1996) Interval computations on the spreadsheet. In: Kearfott RB, Kreinovich V (eds) Applications of Interval Computations. Kluwer, Dordrecht
46. Ichida K, Fujii Y (1979) An interval arithmetic method of global optimization. Computing 23:85–97
47. Van Iwaarden RJ (1996) An improved unconstrained global optimization algorithm. PhD Thesis Univ. Colorado at Denver
48. Jansson C (1994) On self-validating methods for optimization problems. In: Herzberger J (ed) Topics in Validated Computations. Elsevier, Amsterdam, pp 381–438
49. Jansson C, Knüppel O (1995) A branch and bound algorithm for bound constrained optimization problems without derivatives. J Global Optim 7:297–331
50. Jerrell M (1994) Global optimization using interval arithmetic. J Comput Economics 7:55–62
51. Kahan WM (1968) A more complete interval arithmetic. Lecture Notes Univ. Michigan, Ann Arbor, MI
52. Kalmykov SA, Shokin YuI, Yuldashev ZKh (1986) Methods of interval analysis. Nauka, Moscow (In Russian)
53. Kearfott RB (1987) Abstract generalized bisection and a cost bound. Math Comput 49:187–202
54. Kearfott RB (1990) Preconditioners for the interval Gauss–Seidel method. SIAM J Numer Anal 27:804–822
55. Kearfott RB (1996) Interval extensions of non-smooth functions for global optimization and nonlinear systems solver. Computing 57:149–162
56. Kearfott RB (1996) A review of techniques in the verified solution of constrained global optimization problems. In: Kearfott RB, Kreinovich V (eds) Applications of Interval Computations. Kluwer, Dordrecht, pp 23–59
57. Kearfott RB (1996) Rigorous global search: continuous problems. Kluwer, Dordrecht
58. Kearfott RB (1997) On proving existence of feasible points in equality constrained optimization. Preprint
59. Kearfott RB, Shi X (1996) Optimal preconditioners for the interval Gauss-Seidel method. In: Alefeld G Frommer A and Lang B (eds) Scientific Computing and Validated Numerics. Akad. Verlag, Berlin, pp 173–178
60. Krawczyk R (1969) Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. Computing 4:187–201
61. Krawczyk R (1983) Intervallsteigungen für rationale Funktionen und zugeordnete zentrische Formen. Freiburger Intervall-Ber, Inst Angew Math Univ Freiburg 83(2):1–30
62. Krawczyk R, Neumaier A (1985) Interval slopes for rational functions and associated centered forms. SIAM J Numer Anal 22:604–616
63. Kristinsdottir BP, Zabinsky ZB, Csendes T, Tuttle ME (1993) Methodologies for tolerance intervals. Interval Comput 3:133–147
64. Mancini LJ (1975) Applications of interval arithmetic in signomial programming. SOL Techn Report Stanford Univ 75-23
65. Mancini LJ, McCormick GP (1979) Bounding global minima with interval arithmetic. Oper Res 27:743–754
66. Mancini LJ, Wilde DJ (1978) Interval arithmetic in unidimensional signomial programming. J Optim Th Appl 26:277–289
67. Mancini LJ, Wilde DJ (1979) Signomial dual Kuhn–Tucker intervals. J Optim Th Appl 28:11–27
68. Maruyama K (1986) Global optimization with interval analysis (electronic switching systems). Trans Inform Process Soc Japan 27:837–844
69. McKinnon KIM, Millar C, Mongeau M (1995) Global optimization for the chemical and phase equilibrium prob-

lem using interval analysis. In: Floudas CA, Pardalos PM (eds) State of the Art in Global Optimization. Kluwer, Dordrecht, pp 365–382
70. Mohd IB (1962) Global optimization using interval arithmetic. PhD Thesis Univ. St. Andrews, Scotland,).
71. Moore RE (1966) Interval analysis. Prentice-Hall, Englewood Cliffs, NJ
72. Moore RE (1976) On computing the range of values of a rational function of n variables over a bounded region. Computing 16:1–15
73. Moore RE (1977) A test for existence of solutions to nonlinear systems. SIAM J Numer Anal 14:611–615
74. Moore RE (1979) Methods and applications of interval analysis. SIAM, Philadelphia
75. Moore RE (1988) Reliability in computing: the role of interval methods in scientific computing. Acad. Press, New York
76. Moore RE, Hansen ER, Leclerc A (1992) Rigorous methods for global optimization. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton Univ. Press, Princeton, pp 336–342
77. Moore RE, Ratschek H (1988) Inclusion functions and global optimization II. Math Program 41:341–356
78. Munack H (1992) Global optimization of an electron beam lithography system using interval arithmetic. Optik 90:175–183
79. Neumaier A (1990) Interval methods for systems of equations. Cambridge Univ. Press, Cambridge
80. Nonnenmacher A, Mlynski DM (1995) Liquid crystal simulation using automatic differentiation and interval arithmetic. In: Alefeld G, Frommer A and Lang B (eds) Scientific Computing and Validated Numerics. Akad. Verlag, Berlin, pp 334–340
81. Oelschlägel D, Süsse H (1978) Fehlerabschätzung beim Verfahren von Wolfe zur Lösung Quadratischer Optimierungsprobleme mit Hilfe der Intervallarithmetik. Math Operationsforsch Statist Ser Optim 9:389–396
82. Older WD (1993) Using interval arithmetic for non-linear constrained optimization. Manuscript WCLP Invited talk
83. Older WD, Vellino A (1993) Constraint arithmetic on real intervals. In: Benhamou F, Colmerauer A (eds) Constraint Logic Programming. Selected Res. MIT, Cambridge, MA, pp 175–195
84. Rall LB (1981) Automatic differentiation: techniques and applications. Springer, Berlin
85. Ratschek H (1975) Nichtnumerische Aspekte der Intervallarithmetik. In: Nickel K (ed) Interval Mathematics. Springer, Berlin, pp 48–74
86. Ratschek H (1985) Inclusion functions and global optimization. Math Program 33:300–317
87. Ratschek H (1988) Some recent aspects of interval algorithms for global optimization. In: Moore RE (ed) Reliability in Computing: The Role of Interval Methods in Scientific Computing. Acad. Press, New York, pp 325–339
88. Ratschek H, Rokne J (eds) (1984) Computer methods for the range of functions. Horwood, Westergate
89. Ratschek H, Rokne J (1987) Efficiency of a global optimization algorithm. SIAM J Numer Anal 24:1191–1201
90. Ratschek H, Rokne J (1988) New computer methods for global optimization. Horwood, Westergate
91. Ratschek H, Rokne J (1992) Nonuniform variable precision bisecting. In: Brezinski C, Kulisch U (eds) Comput. Appl. Math., vol I. Elsevier, Amsterdam, pp 419–428
92. Ratschek H, Rokne J (1992) The transistor modeling problem again. Microelectronics and Reliability 32:1725–1740
93. Ratschek H, Rokne J (1995) Interval methods. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 751–828
94. Ratschek H, Voller RL (1990) Unconstrained optimization over unbounded domains. SIAM J Control Optim 28:528–539
95. Ratz D (1992) Automatische Ergebnisverifikation bei globalen Optimierungsproblemen. Diss. Univ. Karlsruhe
96. Ratz D (1994) Box-splitting strategies for the interval Gauss–Seidel step in a global optimization method. Computing 53:337–354
97. Ratz D (1996) On branching rules in second-order branch-and-bound methods in a global optimization method. In: Alefeld G, Frommer A and Lang B (eds) Scientific Computing and Validated Numerics. Akad. Verlag, Berlin, pp 221–227
98. Robinson SM (1973) Computable error bounds for nonlinear programming. Math Program 5:235–242
99. Schnepper CA, Stadtherr MA (1993) Application of a parallel interval Newton/generalized bisection algorithm to equation-based chemical process flowsheeting. Interval Comput 3:40–64
100. Sengupta S (1981) Global nonlinear constrained optimization. Diss. Dept. Pure Appl. Math., Washington State Univ.
101. Shen Z, Wolfe MA (1990) On interval enclosures using slope arithmetic. Appl Math Comput 39:89–105
102. Shokin Yul (1981) Interval analysis. Nauka, Moscow (In Russian)
103. Skelboe S (1974) Computation of rational interval functions. BIT 14:87–95
104. Stroem T (1971) Strict estimation of the maximum of a function of one variable. BIT 11:199–211
105. Süsse H (1977) Intervallarithmetische Behandlung von Optimierungsproblemen und Damit Verbundener Numerischer Aufgabenstellungen. Diss., Techn. Hochsch. 'Carl Schorlemmer' Leuna-Merseburg
106. Vaidyanathan R, El-Halwagi M (1994) Global optimization of nonconvex nonlinear programs via interval analysis. Comput Chem Eng 18:889–897
107. Venkataramanan MA, Cabot AV, Winston WL (1995) An interval branch and bound algorithm for global optimization of a multiperiod pricing model. Comput Oper Res 22:681–687

108. Wengert RE (1964) A simple automatic derivative evaluation program. Comm ACM 7:463–464
109. Wolfe MA (1994) An interval algorithm for constrained global optimization. J Comput Appl Math 50:605–612
110. Wolfe MA (1996) Interval methods for global optimization. Appl Math Comput 75:179–206
111. Wolfe MA, Zhang LS (1994) An interval algorithm for non-differentiable global optimization. Appl Math Comput 63:101–122

Interval Linear Systems

C. Jansson
Informatik III, Techn. Universität Hamburg-Harburg, Hamburg, Germany

MSC2000: 15A99, 65G20, 65G30, 65G40, 90C26

Article Outline

Keywords
An Iterative Interval Method
Optimal Bounds
See also
References

Keywords

Interval arithmetic; Optimization; Linear systems of equations

In many applications the coefficients of real linear systems are, due to measurement or approximation errors, not known exactly. Therefore, the family of real linear systems

$$A \cdot x = b, \tag{1}$$

where A, b satisfy the inequalities

$$|A^c - A| \leq \Delta, \quad |b - b^c| \leq \delta \tag{2}$$

is considered. The absolute value and comparisons are used entrywise. The matrices A^c, A, $\Delta \in \mathbf{R}^{n \times n}$ are real $n \times n$ matrices, b^c, b, $\delta \in \mathbf{R}^n$, and Δ, δ, which describe the perturbation bounds, are assumed to be nonnegative. This family of real linear systems is called an *interval linear system*, because each matrix A, right-hand side b is contained in the interval matrix $\mathbf{A} := [A^c - \Delta, A^c + \Delta]$, interval vector $\mathbf{b} := [b^c - \delta, b^c + \delta]$, respectively. A^c and b^c are called the *centers* of the interval linear system.

The corresponding *solution set* X is defined as the union of all solutions of this family, that is

$$X := \{x \in \mathbb{R}^n : \ x, A, b \text{ satisfy (1), (2)}\}. \tag{3}$$

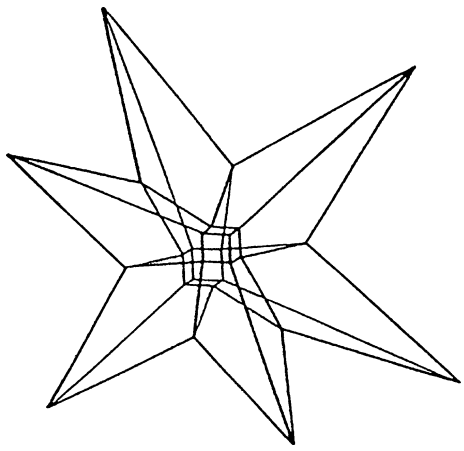
Naturally, the main interest is to determine the exact range of each component of the solution set, that is to calculate the exact or *optimal componentwise bounds*

$$\min\{x_i : \ x \in X\}, \quad \max\{x_i : \ x \in X\} \tag{4}$$

for $i = 1, \ldots, n$. The minima and maxima exist provided $\mathbf{A}$ is *regular*, that is all matrices $A \in \mathbf{A}$ are nonsingular. Otherwise, $\mathbf{A}$ is called *singular*, and X is unbounded or empty.

In general, the solution set X is not convex and has a complicated shape: see Fig. 1 which is taken from a book of A. Neumaier [18, p. 97]. Hence, calculating bounds for the solution set X is a global optimization problem. Moreover, X needs not to be connected or bounded. This is shown by the simple one-dimensional equation $A \cdot x = 1$, $A \in [-1, 1]$ with solution set $X = (-\infty, -1] \cup [1, \infty)$.

From the point of view of complexity theory, J. Rohn [25] has proved that the problem of calculating bounds for the solution set is *NP*-hard. Roughly speaking, he has shown that there is no polynomial time algorithm which calculates bounds of the solution set with overestimation less than any given positive constant.



Interval Linear Systems, Figure 1
A projection of a three-dimensional solution set

This is true, even if the interval matrix $\mathbf{A}$ is *strongly regular*, i. e. if the spectral radius $\rho(|(A^c)^{-1}| \cdot \Delta)$ is less than one. If $\mathbf{A}$ is strongly regular, then the regularity of $\mathbf{A}$ follows immediately by observing that for $A \in \mathbf{A}$ there holds

$$A = A^c - \widetilde{\Delta} = A^c \cdot (I - (A^c)^{-1} \cdot \widetilde{\Delta}), \tag{5}$$

where $|\widetilde{\Delta}| \leq \Delta$. Hence, singularity of A is equivalent to the fact that $(A^c)^{-1} \cdot \widetilde{\Delta}$ has the eigenvalue 1. Since $\rho(|(A^c)^{-1}| \cdot \Delta) < 1$ it follows that A is regular. By Perron–Frobenius theory, strong regularity implies that the radius matrix Δ is not too large. For further *NP*-hardness results related to other interval problems see [27].

During the last three decades the problem of calculating componentwise bounds for X, not necessarily optimal bounds, has received much attention, and many methods were developed. No attempt can be made in this short survey to review all different approaches. But the literature given in this section shall serve as a guide for further reading.

The first algorithm for calculating optimal componentwise bounds was given by W. Oettli and W. Prager [19,20]. There the solution set X is described as the set of feasible solutions of a special system of nonlinear inequalities:

$$X = \{x \in \mathbb{R}^n : \ |A^c x - b^c| \leq \Delta \cdot |x| + \delta\} \tag{6}$$

But in each orthant this system is a convex polyhedron. Hence, in each orthant optimal bounds can be calculated by using linear programming techniques. Unfortunately, there are 2^n orthants, and therefore this method needs for each instance a priori exponential time, and can work only for problems of very small size.

Recently, based on the result of Oettli and Prager, in [9] a more efficient method for calculating optimal bounds is presented. This method uses linear programming techniques in only those orthants which are intersected by the solution set X.

Starting with the pioneering book of R.E. Moore [15], a large number of methods were proposed using the tools of interval arithmetic. Many algorithms can be found for example in the monographs [2,16], and [18]. These methods are polynomial time algorithms, calculate only componentwise (not optimal) bounds, and work under special assumptions: in almost all cases strong regularity of $\mathbf{A}$ is required.

In interval arithmetic the elementary operations for intervals $\mathbf{x} = [\underline{x}, \overline{x}], \mathbf{y} = [\underline{y}, \overline{y}] \in \mathbb{IR}$ are defined by

$$\mathbf{x} * \mathbf{y} = \{x * y : \ x \in \mathbf{x}, \quad y \in \mathbf{y}\} \tag{7}$$

where $* \in \{+, -, \cdot, /\}$, and in case of division $0 \notin \mathbf{y}$ is assumed. By a simple monotonicity argument it follows that

$$\mathbf{x} * \mathbf{y} = [\min S, \max S], \tag{8}$$

where the set S is defined by

$$S := \{\underline{x} * \underline{y}, \underline{x} * \overline{y}, \overline{x} * \underline{y}, \overline{x} * \overline{y}\}.$$

Interval operations between real matrices, interval matrices, real vectors and interval vectors are defined as in the real case, only the real operations are replaced by the corresponding interval operations (7).

For example, if $R = (r_{ij}) \in \mathbf{R}^{n \times n}$ is a real $n \times n$ matrix, and $\mathbf{b} \in \mathbf{IR}^n$, then $R \cdot \mathbf{b}$ is defined as follows: the real coefficients r_{ij} are replaced by the point intervals $\mathbf{r}_{ij} = r_{ij} = [r_{ij}, r_{ij}]$ and

$$(R \cdot \mathbf{b})_i := \sum_{j=1}^{n} \mathbf{r}_{ij} \cdot \mathbf{b}_j.$$

By definition (7), for all i the equation

$$(R \cdot \mathbf{b})_i = \left\{ \sum_{j=1}^{n} r_{ij} \cdot b_j : \ b \in \mathbf{b} \right\}$$

holds. Therefore $R \cdot \mathbf{b}$ is the smallest interval vector containing the set $\{R \cdot b : b \in \mathbf{b}\}$. But in general, $R \cdot \mathbf{b}$ overestimates the latter set.

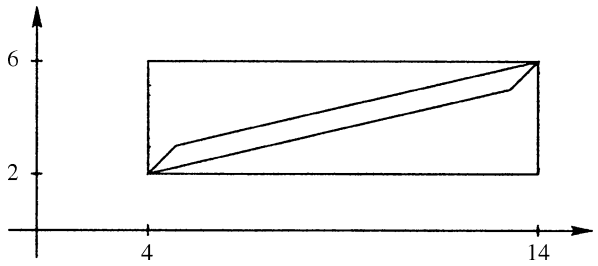
Example 1 Let

$$R = \begin{pmatrix} 1 & 3 \\ 1 & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} [1, 2] \\ [1, 4] \end{pmatrix},$$

then $R \cdot \mathbf{b} = ([4, 14], [2, 6])^\mathsf{T}$, but $\{R \cdot b : b \in \mathbf{b}\}$ is the convex hull of the set $\{(4, 2)^\mathsf{T}, (13, 5)^\mathsf{T}, (5, 3)^\mathsf{T}, (14, 6)^\mathsf{T}\}$, see Fig. 2.

In many interval methods for calculating bounds of X the interval linear system is first preconditioned by an appropriate matrix R, which in most cases is an approximate inverse of the center A^c. This yields the preconditioned interval linear system

$$(R \cdot \mathbf{A}) \cdot x = R \cdot \mathbf{b}. \tag{9}$$



Interval Linear Systems, Figure 2

with interval matrix $R \cdot \mathbf{A}$ and interval right hand side $R \cdot \mathbf{b}$.

From the discussion above it follows that $R \cdot \mathbf{A}$, $R \cdot \mathbf{b}$ overestimate the sets $\{R \cdot A: A \in \mathbf{A}\}$, $\{R \cdot b: b \in \mathbf{b}\}$, respectively. Therefore, the solution set X' of the preconditioned interval linear system (9) contains but overestimates the solution set X. However, for small Δ the interval matrix $R \cdot \mathbf{A}$ is close to the identity matrix and diagonal dominant. Hence, only small overestimations will occur.

Bounds for the solution set X' of the preconditioned interval linear system (9) are then calculated by using interval Gaussian elimination (cf. for example [1,7]), interval Gauss–Seidel iteration (cf. for example [6,17,22]), or fixed point iteration (cf. for example [14,28]). In general, these bounds are not optimal for X', and the overestimation depends on the method.

But recently E.R. Hansen [5], and Rohn [26] have presented a polynomial time algorithm for calculating optimal bounds for the solution set X' of the preconditioned interval system (9). Only two matrix inversions are required.

Preconditioning of interval linear systems was first suggested by Hansen and R. Smith [7]. Later, R.B. Kearfott [11,12] introduced the so-called width optimal preconditioners by using linear programming techniques.

Preconditioning requires the computation of an approximate inverse. For sparse linear systems the inverse in general is full. Therefore, the approaches described above are not applicable for large dimensions. But recently, S.M. Rump [29,30] generalized his iteration method (cf. [28]) to sparse nonlinear systems without preconditioning with a full inverse. His idea, roughly spoken, was to replace the inverse by a lower bound of the smallest singular value of the center matrix.

Last, an interval method not using preconditioning should be mentioned. This method is a branch and bound scheme proposed by S.P. Shary [31].

In the following, two methods are described in more detail. First, in the next section 2 Rump 's method [28] is presented. This method is implemented in several programming packages like ACRITH [8], ARITHMOS [3], PASCAL-XSC [4], and PROFIL [13]. Moreover, as mentioned above, the method can be modified for solving sparse interval linear systems and nonlinear systems. Then, in the last section the method presented in [9] for calculating optimal bounds of X is described.

An Iterative Interval Method

It is assumed that A, b satisfy the inequalities (2), R is an approximate inverse of A^c, and $\check{x}$ is an approximate solution of $A^c x = b^c$. No assumptions about the quality of these approximations are made. It is well known from numerical linear algebra that defect iteration with the iteration function

$$f(x) := x + R \cdot (b - A(\check{x} + x)) = R \cdot (b - A\check{x}) + (I - RA)x \quad (10)$$

can be used to improve the quality of the approximation $\check{x}$. This function is continuous, and if for a given interval vector $\mathbf{x}$ the condition $f(\mathbf{x}) \subseteq \mathbf{x}$ holds, then by Brouwer 's fixed point theorem there exists $\widehat{x} \in \mathbf{x}$ with $f(\widehat{x}) = \widehat{x}$. Using (10) yields $R \cdot (b - A(\check{x} + \widehat{x})) = 0$. If R is nonsingular, then $A(\check{x} + \widehat{x}) = b$ implying that $\check{x} + \widehat{x} \in \check{x} + \mathbf{x}$ is the exact solution of $Ax = b$. Moreover, by using a contradiction argument, it can be shown that the solution $\check{x} + \widehat{x}$ is unique and R, A are nonsingular provided that $f(\mathbf{x})$ is contained in $\mathrm{int}(\mathbf{x})$, the interior of $\mathbf{x}$.

An immediate consequence is that if the condition

$$R \cdot (\mathbf{b} - \mathbf{A} \cdot \check{x}) + (I - R \cdot \mathbf{A}) \cdot \mathbf{x} \subseteq \mathrm{int}(\mathbf{x}) \quad (11)$$

is satisfied, then $f(\mathbf{x}) \subseteq \mathrm{int}(\mathbf{x})$ holds for all $A \in \mathbf{A}$, $b \in \mathbf{b}$. Hence $X \subseteq \check{x} + \mathbf{x}$. Notice that (11) can be easily checked by using interval arithmetic.

The remaining problem is to find an appropriate box $\mathbf{x}$ satisfying (11). The following iteration starting with $\mathbf{x}^0 := R \cdot (\mathbf{b} - \mathbf{A} \cdot \check{x})$ can be used:

$$\begin{aligned} \mathbf{y}^k &:= \mathbf{x}^k \cdot [1 - \epsilon, 1 + \epsilon] + [-\mu, \mu] \cdot e, \\ \mathbf{x}^{k+1} &:= \mathbf{x}^0 + (I - R \cdot \mathbf{A}) \cdot \mathbf{y}^k. \end{aligned} \quad (12)$$

The values $\epsilon, \mu > 0$ are called inflation parameters, and e is the vector with 1 in each component. The main property of this iteration is that (cf. [30]) for every starting box $\mathbf{x}^0$ holds

$$\exists k \in \mathbb{N} : \ \mathbf{x}^{k+1} \subseteq \operatorname{int}(\mathbf{y}^k)$$
$$\Updownarrow$$
$$\rho(|I - R \cdot \mathbf{A}|) < 1,$$

where ρ denotes the spectral radius of the absolute value of $I - R \cdot \mathbf{A}$. This means that after a finite number of steps bounds $\check{x} + \mathbf{x}^{k+1}$ of X are calculated, provided the spectral radius $\rho(|I - R\mathbf{A}|) < 1$; this means that $\mathbf{A}$ is strongly regular. For practical applications it is recommended to execute at most $k = 10$ iteration steps, and μ should be greater than the smallest positive floating point number. Obviously, by using this parameters we get an $0(n^3)$ polynomial time algorithm.

Example 2 To demonstrate how this algorithm works, the following interval linear system with centers

$$A^c = \begin{pmatrix} 1.2 & 1.2 \\ -1.2 & 1.2 \end{pmatrix}, \quad b^c = \begin{pmatrix} 1.5 \\ 3.5 \end{pmatrix},$$

and perturbation bounds

$$\Delta = \begin{pmatrix} 0.2 & 0.2 \\ 0.2 & 0.2 \end{pmatrix}, \quad \delta = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

is considered.

This system is a slight modification of an example of Rohn [23]. We have chosen $\epsilon = 0.05$ and μ equal to the smallest positive machine number. In the following, five (appropriately rounded) decimal digits are displayed. The two-dimensional interval vector with components equal to $[-1, 1]$ is denoted by $[-\mathbf{1}, \mathbf{1}]$.

The spectral radius $\rho(|I - R \cdot \mathbf{A}|) \approx 0.3333 < 1$ where $R \approx (A^c)^{-1}$, and therefore the iteration (12) will compute a box containing the solution set X in finitely steps.

The approximate solution of the center system is $\check{x} = (-0.83333, 2.0833)^\top$ yielding the starting box $\mathbf{x}^0 := R \cdot (\mathbf{b} - \mathbf{A} \cdot \check{x}) = 0.9028 \cdot [-\mathbf{1}, \mathbf{1}]$.

Iteration (12) yields

$$\mathbf{y}^0 = 0.9480 \cdot [-\mathbf{1}, \mathbf{1}], \quad \mathbf{x}^1 = 1.2188 \cdot [-\mathbf{1}, \mathbf{1}],$$
$$\mathbf{y}^1 = 1.2797 \cdot [-\mathbf{1}, \mathbf{1}], \quad \mathbf{x}^2 = 1.3294 \cdot [-\mathbf{1}, \mathbf{1}],$$
$$\mathbf{y}^2 = 1.3959 \cdot [-\mathbf{1}, \mathbf{1}], \quad \mathbf{x}^3 = 1.3681 \cdot [-\mathbf{1}, \mathbf{1}].$$

Hence, for $k = 2$ it follows $\mathbf{x}^3 \subseteq \operatorname{int}(\mathbf{y}^2)$, and the solution set X is contained in

$$\check{x} + \mathbf{y}^2 = \begin{pmatrix} [-2.2014, 0.5348] \\ [0.7152, 3.4514] \end{pmatrix} \tag{13}$$

For numerical results of this method and its generalization to sparse systems, see [29,30]. There, examples up to 1000000 variables including the 'Harwell test cases' are presented.

Optimal Bounds

As pointed out in the introduction, a polynomial time algorithm may overestimate the solution set X drastically or may fail. Therefore, in this section a method (cf. [9]) which produces optimal bounds of X if and only if $\mathbf{A}$ is regular is described.

An immediate consequence of (6) is that the solution set X is the finite union of convex polyhedrons. To see this, let $\{-1, 1\}^n$ denote the set of all sign vectors with components equal to 1 or -1. For a sign vector $s \in \{-1, 1\}^n$ let $D(s)$ denote the diagonal matrix with diagonal s and $\mathbf{R}^n(s) := \{x \in \mathbf{R}^n : D(s) \cdot x \geq 0\}$. Then the intersection $X(s) := X \cap \mathbf{R}^n(s)$ of the solution set with the orthant corresponding to s is given by the following system of linear inequalities

$$\begin{aligned} &(A^c - \Delta \cdot D(s)) \cdot x \leq b^c + \delta \\ &(A^c + \Delta \cdot D(s)) \cdot x \geq b^c - \delta \\ &D(s) \cdot x \geq 0. \end{aligned} \tag{14}$$

Therefore, for a fixed orthant $\mathbf{R}^n(s)$ optimal bounds of $X(s)$ can be calculated by minimizing and maximizing each coordinate x_i subject to the constraints (14). These are linear programming problems which can be solved in polynomial time, implying that optimal bounds of $X(s)$ can also be calculated in polynomial time.

Now, one can get optimal bounds of X by calculating optimal bounds of $X(s)$ for each orthant $\mathbf{R}^n(s)$. Unfortunately, there are 2^n orthants, and this approach can work only for very small dimension n.

For interval linear systems with $\Delta = 0, \delta = 0$ the solution set X is by definition equal to the exact solution of the corresponding real linear system, and therefore X will be in one orthant (with exception of degenerated

cases). With growing radii Δ, δ the solution set may intersect more orthants. But in many cases only few orthants will intersect the solution set.

Then most of the computing time of the above approach will be spent for checking that $X \cap \mathbf{R}^n(s)$ is empty for almost all orthants. Therefore, the question arises if it is possible to construct an algorithm which picks up exactly those orthants where $X \cap \mathbf{R}^n(s)$ is nonempty. In the following such an algorithm is presented. This approach heavily relies on the following topological alternative statement, which says that for nonempty X exactly one of the following two statements is true:

i) X is compact and connected, and $\mathbf{A}$ is regular;
ii) X is unbounded, each topologically connected component of X is unbounded, and $\mathbf{A}$ is singular.

An immediate consequence is that the solution set X cannot be the union of bounded and unbounded topologically connected components. Therefore, each method which only calculates optimal bounds of a topologically connected component of X suffices to solve the problem. To do this, the representation graph $G = (V, E)$ of the solution set X with the set of nodes

$$V = \{s \in \{-1, 1\}^n : \ X(s) \neq \emptyset\}, \tag{15}$$

and the set of edges

$$E = \left\{\{s, t\}: \quad \begin{array}{l} s, t \in V, \ s \text{ and } t \text{ differ in} \\ \text{exactly one component} \end{array}\right\} \tag{16}$$

is defined.

Now the following basic relationship between the solution set and its representation graph can be proved:

a) Each nonempty topologically connected component $\widehat{X}$ of X can be represented in the form

$$\widehat{X} = \cup\{X(s): \ s \in U\}, \tag{17}$$

where U is the node set of a connected component of G.

b) If X is nonempty and bounded, then $G = (V, E)$ is a connected graph, and

$$X = \cup\{X(s): \ s \in V\}. \tag{18}$$

This property gives the possibility to apply to the implicitly defined representation graph G the well-known graph search method (see for example [21]) for calculating a connected component:

1) Compute a starting node $s \in V$ by solving the midpoint system $A^c x = b^c$. The vector s is defined as the sign vector of this solution, and stored in a list L.
2) Put a sign vector $s \in L$, and solve the linear programming problems

$$\begin{cases} \min\{x_i : \ x \in X(s)\}, \\ \max\{x_i : \ x \in X(s)\} \end{cases} \tag{19}$$

for $i = 1, \ldots, n$.

If a problem is unbounded, then an unbounded topologically connected component of X is found. Hence, each other topologically connected component of X is unbounded, $\mathbf{A}$ is singular and the method is stopped. Otherwise, the linear programming problems calculate optimal bounds of $X(s)$, which are also stored. By definition of the edge set E, it follows immediately that

$$t := (s_1, \ldots, s_{i-1}, -s_i, s_{i+1}, \ldots, s_n) \tag{20}$$

is adjacent to s, if and only if one of the lp 's in (19) has the exact bound equal to zero. All neighbored nodes t of s are stored in list L, except those which have been already treated. Then we proceed by going to 2), and repeat this process until L is empty.

It follows that this algorithm terminates in a finite number of steps, and either calculates optimal bounds of the solution set and proves regularity of $\mathbf{A}$, or shows that X is unbounded and $\mathbf{A}$ is singular. The algorithm searches only in those orthants which have a nonempty intersection with the solution set, and avoids all other ones. Therefore, $|V|$ calls of a polynomial time algorithm are needed, where $|V|$ is the number of nonempty intersections of the solution set with the orthants.

In many cases in practice, due to physical or economical requirements, only few variables will change the sign implying that only few orthants will be intersected by the solution set. In those cases the method works efficiently. Nevertheless, due to the mentioned *NP*-hardness results of Rohn, there are also cases where an exponential computing time occurs.

Example 3 In order to see how this algorithm behaves in detail, the example of the previous section is discussed. The solution $\check{x} = (-0.8333, 2.0833)^\top$ gives the sign vector $s = (-1, 1)$ which is stored in L.

Now we take this sign vector from list L (then L is empty) and solve the lp 's (19) which gives the optimal

bounds of $X(s)$

$$\begin{pmatrix} [-1.9167, -0.0595] \\ [1.3095, 3.1667] \end{pmatrix}. \tag{21}$$

No optimal bound has a value equal to zero, which implies that $s = (-1, 1)$ has no neighbor with respect to the edge set E. It follows that $X(s)$ is a topologically connected component and $X = X(s)$. Therefore, (21) gives the optimal bounds of X.

Following, the original example of Rohn [24] is discussed, which differs from the previous one by changing

$$A_c := \begin{pmatrix} 500.5 & 500.5 \\ -500.5 & 500.5 \end{pmatrix},$$

$$\Delta := \begin{pmatrix} 499.5 & 499.5 \\ 499,5 & 499.5 \end{pmatrix}.$$

Thus very large perturbations Δ are allowed, and the spectral radius $\rho(|(A^c)^{-1}| \cdot \Delta) = 1.9960$. Hence the iteration method of the previous section cannot work, because $\mathbf{A}$ is not strongly regular. The solution $\check{x} = (-0.001998, 0.004995)^T$ gives $s = (-1, 1)^T$ and $L = \{s\}$.

Now s is removed from list L (then L is empty) and the lp 's (19) yield the following optimal bounds of $X(s)$:

$$\begin{pmatrix} [-3.9950, 0] \\ [0.001002, 3.9980] \end{pmatrix}. \tag{22}$$

One optimal bound of the first component has a value equal to zero. Therefore, by (20) $t = (-s_1, s_2)^T = (1, 1)^T$ is adjacent to s and list $L := \{t\}$.

Now we take t from list L (then L is empty), and the lp 's (19) yield the optimal bounds of $X(t)$:

$$\begin{pmatrix} [0, 1.9950] \\ [0.0030, 2.0000] \end{pmatrix}. \tag{23}$$

Only the lower optimal bound of the first component is equal to zero. This gives the adjacent sign vector $s = (-t_1, t_2) = (-1, 1)^T$. But this is the sign vector already treated, and therefore not stored in list L. Since list L is empty, the algorithm is finished, and the optimal bounds (22) and (23) together deliver the optimal bounds

$$\begin{pmatrix} [-3.9950, 1.99950] \\ [0.001002, 3.9980] \end{pmatrix} \tag{24}$$

for the solution set X.

By comparing the bounds (21) and (13), we see that the optimal bounds (21) clearly improve the bounds (13) calculated by the iteration method of the previous section. This overestimation is mainly due to the preconditioning with the midpoint inverse. However, the bounds (13) give additionally the information that the solution set X intersects at most 2 orthants.Thus, an a priori estimation on the computing time for the exact method in this section is given: the above method has only to search in two orthants. Hence, first using in the strongly regular case a polynomial time method, provides rough bounds for X as well as a bound for the computing time which is needed for calculating exact bounds.

Several other examples up to dimension $n = 50$ can be found in [9] and [10].

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- Automatic Differentiation: Point and Interval
- Automatic Differentiation: Point and Interval Taylor Operators
- Bounding Derivative Ranges
- Cholesky Factorization
- Global Optimization: Application to Phase Equilibrium Problems
- Interval Analysis: Application to Chemical Engineering Design Problems
- Interval Analysis: Differential Equations
- Interval Analysis: Eigenvalue Bounds of Interval Matrices
- Interval Analysis: Intermediate Terms
- Interval Analysis: Nondifferentiable Problems
- Interval Analysis: Parallel Methods for Global Optimization
- Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- Interval Analysis: Systems of Nonlinear Equations
- Interval Analysis: Unconstrained and Constrained Optimization
- Interval Analysis: Verifying Feasibility
- Interval Constraints
- Interval Fixed Point Theory
- Interval Global Optimization
- Interval Newton Methods

- ▶ Large Scale Trust Region Problems
- ▶ Large Scale Unconstrained Optimization
- ▶ Linear Programming
- ▶ Nonlinear Least Squares: Trust Region Methods
- ▶ Orthogonal Triangularization
- ▶ Overdetermined Systems of Linear Equations
- ▶ QR Factorization
- ▶ Solving Large Scale and Sparse Semidefinite Programs
- ▶ Symmetric Systems of Linear Equations

References

1. Alefeld G (1994) Inclusion methods for systems of nonlinear equations. In: Herzberger J (ed) Topics in Validated Computations: Studies in Computational Mathematics. North-Holland, Amsterdam, pp 7–26
2. Alefeld G, Herzberger J (1983) Introduction to interval computations Acad. Press, New York
3. Arithmos (1986) Benutzerhandbuch, Siemens, Bibl.-Nr. U 2900-I-Z87-1.
4. Hammer R, Hocks M, Kulisch U, Ratz D (1993) PASCAL-XSC: Basic numerical problems Springer, Berlin
5. Hansen ER (1992) Bounding the solution set of interval linear systems. SIAM J Numer Anal 29:1493–1503
6. Hansen E, Sengupta S (1981) Bounding solutions of systems of equations using interval analysis. BIT 21:203–211
7. Hansen E, Smith R (1967) Interval arithmetic in matrix computations. SIAM Numer Anal 2(4):1–9
8. IBM (1986) High-accuracy arithmetic subroutine library (ACRITH). Program Description and User's Guide SC 33-6164-02
9. Jansson C (1997) Calculation of exact bounds for the solution set of linear interval equations. Linear Alg & Its Appl 251:321–340
10. Jansson C, Rohn J (1999) An algorithm for checking regularity of interval matrices. SIAM J Matrix Anal Appl 20(3):756–776
11. Kearfott RB (1990) Preconditioners for the interval-Gauss-Seidel method. SIAM J Numer Anal 27(3):804–822
12. Kearfott RB (1996) Rigorous global search: continuous problems. Kluwer, Dordrecht
13. Knüppel O (1994) PROFIL/BIAS: A fast interval library. Computing 53:277–287
14. Krawczyk R (1969) Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. Computing 4:187–201
15. Moore RE (1966) Interval analysis. Prentice-Hall, Englewood Cliffs, NJ
16. Moore RE (1979) Methods and applications of interval analysis. SIAM, Philadelphia
17. Neumaier A (1984) New techniques for the analysis of linear interval equations. Linear Alg & Its Appl 58:273–325
18. Neumaier A (1990) Interval methods for systems of equations. Encycl Math Appl. Cambridge Univ. Press, Cambridge
19. Oettli W (1965) On the solution set of a linear system with inaccurate coefficients. SIAM J Numer Anal 2:115–118
20. Oettli W, Prager W (1964) Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. Numer Math 6:405–409
21. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: Algorithms and complexity. Prentice-Hall, Englewood Cliffs, NJ
22. Ris FN (1972) Interval analysis and applications to linear algebra. PhD Thesis Oxford Univ.
23. Rohn J (1986) A note on the sign-accord algorithm. Freiburger Intervall-Ber 86(4):39–43
24. Rohn J (1989) Systems of linear interval equations. Linear Alg & Its Appl 126:39–78
25. Rohn J (1991) Linear interval equations: computing enclosures with bounded relative or absolute overestimation is NP-hard. In: Kearfott RB, Kreinovich V (eds) Applications of Interval Computations. Kluwer, Dordrecht, pp 81–89
26. Rohn J (1992) Cheap and tight bounds: the result of E. Hansen can be made more efficient. Interval Comput 4:13–21
27. Rohn J (1994) NP-hardness results for linear algebraic problems with interval data. In: Herzberger J (ed) Topics in Validated Computations: Studies in Computational Mathematics. Elsevier, Amsterdam, 463–472
28. Rump SM (1983) Solving algebraic problems with high accuracy. Habilitationsschrift. In: Kulisch UW, Miranker WL (eds) A New Approach to Scientific Computation. Acad. Press, New York, pp 51–120
29. Rump SM (1993) Validated solution of large linear systems. In: Albrecht R, Alefeld G, Stetter HJ (eds) Computing Supplementum 9, Validation Numerics. Springer, Berlin, pp 191–212
30. Rump SM (1994) Verification methods for dense and sparse systems of equations. In: Herzberger J (ed) Topics in Validated Computations: Studies in Computational Mathematics. Elsevier, Amsterdam, pp 63 –136
31. Shary SP (1991) Optimal solutions of interval linear algebraic systems. Interval Comput 2:7–30

Interval Newton Methods

R. Baker Kearfott
Department Math.,
University Louisiana at Lafayette,
Lafayette, USA

MSC2000: 65G20, 65G30, 65G40, 65H20, 65K99

Article Outline

Keywords
Introduction
Univariate Interval Newton Methods
Multivariate Interval Newton Methods
Existence-Proving Properties
See also
References

Keywords

Nonlinear system of equations; Automatic result verification; Interval computations; Global optimization

Introduction

Interval Newton methods combine the classical Newton method, the *mean value theorem*, and *interval analysis*. The result is an iterative method that can be used both to refine enclosures to solutions of *nonlinear systems of equations*, to prove *existence* and *uniqueness* of such solutions, and to provide *rigorous bounds* on such solutions, including tight and rigorous bounds on critical points of constrained optimization problems. Interval Newton methods can also prove nonexistence of solutions within regions. Such capabilities can be used in isolation, for example, to provide rigorous *error bounds* for an approximate solution obtained with floating point computations, or as an integral part of global *branch and bound algorithms*.

Univariate Interval Newton Methods

Suppose $f\colon\ \mathbf{x} = [\underline{x}, \overline{x}] \to \mathbb{R}$ has a continuous first derivative on $\mathbf{x}$, suppose that there exists $x^* \in \mathbf{x}$ such that $f(x^*) = 0$, and suppose that $\check{x} \in \mathbf{x}$. Then, since the mean value theorem implies

$$0 = f(x^*) = f(\check{x}) + f'(\xi)(x^* - \check{x}),$$

we have $x^* = \check{x} - \frac{f(\check{x})}{f'(\xi)}$ for some $\xi \in \mathbf{x}$. If $\mathbf{f}'(\mathbf{x})$ is any interval extension of the derivative of f over $\mathbf{x}$, then

$$x^* \in \check{x} - \frac{f(\check{x})}{\mathbf{f}'(\mathbf{x})} \quad \text{for any } \check{x} \in \mathbf{x}. \tag{1}$$

(Note that, in certain contexts, a *slope set* for f centered at $\check{x}$ may be substituted for $\mathbf{f}'(\mathbf{x})$; see [1] for further references.) Equation (1) forms the basis of the *univariate interval Newton operator*:

$$\mathbf{N}(\mathbf{f}, \mathbf{x}, \check{x}) = \check{x} - \frac{f(\check{x})}{\mathbf{f}'(\mathbf{x})}. \tag{2}$$

Because of (1), any solutions of $f(x) = 0$ that are in $\mathbf{x}$ must also be in $\mathbf{N}(\mathbf{f}, \mathbf{x}, \check{x})$. Furthermore, local convergence of iteration of the interval Newton method (2) is quadratic in the sense that the width of $\mathbf{N}(\mathbf{f}, \mathbf{x}, \check{x})$ is roughly proportional to the square of the width of $\mathbf{x}$. Furthermore, if an interval derivative extension (in contrast to an interval slope) is used for $\mathbf{f}'(\mathbf{x})$, then

$$\mathbf{N}(\mathbf{f}, \mathbf{x}, \check{x}) \subset \operatorname{int}(\mathbf{x}),$$

where $\operatorname{int}(\mathbf{x})$ represents the interior of $\mathbf{x}$, implies that there is a unique solution of $f(x) = 0$ within $\mathbf{N}(\mathbf{f}, \mathbf{x}, \check{x})$, and hence within $\mathbf{x}$.

Multivariate Interval Newton Methods

Multivariate interval Newton methods are analogous to univariate ones in the sense that they obey an iteration equation similar to equation (2), and in the sense that they have quadratic convergence properties and can be used to prove existence and uniqueness. However, multivariate interval Newton methods are complicated by the necessity to bound the solution set of a linear system of equations with interval coefficients.

Suppose now that $f\colon \mathbf{R}^n \to \mathbf{R}^n$, suppose $\mathbf{x}$ is an interval vector (i. e. a *box*), and suppose that $\check{x} \in \mathbb{R}^n$. (If interval derivatives, rather than slope sets, are to be used, then further suppose that $\check{x} \in \mathbf{x}$.) Then a general form for multivariate interval Newton methods is

$$\mathbf{N}(f, \mathbf{x}, \check{x}) = \check{x} + \mathbf{v}, \tag{3}$$

where $\mathbf{v}$ is an interval vector that contains all solutions v to point systems $Av = -f(\check{x})$, for $A \in \mathbf{f}'(\mathbf{x})$, where $\mathbf{f}'(\mathbf{x})$ is an interval extension to the Jacobi matrix of f over $\mathbf{x}$. (Under certain conditions, $\mathbf{f}'$ may be replaced by an interval slope matrix.) As with the univariate interval Newton method, under certain natural smoothness conditions,

- $\mathbf{N}(f, \mathbf{x}, \check{x})$ must contain all solutions $x^* \in \mathbf{x}$ with $f(x^*) = 0$. (Consequently, if $\mathbf{N}(f, \mathbf{x}, \check{x}) \cap \mathbf{x} = \emptyset$, then there are no solutions of $f(x) = 0$ in $\mathbf{x}$.)
- For $\mathbf{x}$ containing a solution of $f(x) = 0$ and the widths of the components of $\mathbf{x}$ sufficiently small, the width of $\mathbf{N}(f, \mathbf{x}, \check{x})$ is roughly proportional to the square of the widths of the components of $\mathbf{x}$.

- If $\mathbf{N}(f, \mathbf{x}, \check{x}) \subset \text{int}(\mathbf{x})$, where int($\mathbf{x}$) represents the interior of $\mathbf{x}$, then there is a unique solution of $f(x) = 0$ within $\mathbf{N}(f, \mathbf{x}, \check{x})$, and hence within $\mathbf{x}$.

For details and further references, see [1, §1.5].

Finding the interval vector $\mathbf{v}$ in the iteration formula (3), that is, bounding the solution set to the interval linear system

$$\mathbf{f}'(\mathbf{x})v = -f(\check{x}),$$

is a major aspect of the multivariate interval Newton method. Finding the narrowest possible intervals for the components of $\mathbf{v}$ is, in general, an *NP*-hard problem. (See ► Complexity classes in optimization.) However, procedures that are asymptotically good in the sense that the overestimation in $\mathbf{v}$ decreases as the square of the widths of the elements of $\mathbf{f}'$ can be based on first *preconditioning* the interval matrix $\mathbf{f}'(\mathbf{x})$ by the inverse of its matrix of midpoints or by other special preconditioners (see [1, Chapt. 3]), then applying the interval Gauss–Seidel method or interval Gaussian elimination.

Existence-Proving Properties

The existence-proving properties of interval Newton methods can be analyzed in the framework of classical fixed-point theory. See ► Interval fixed point theory, or [1, §1.5.2]. Of particular interest in this context is a variant interval Newton method, not fitting directly into the framework of formula (3), that is derived directly by considering the classical chord method (Newton method with fixed iteration matrix) as a fixed point iteration. Called the *Krawczyk method*, this method has various nice theoretical properties, but its image is usually not as narrow as other interval Newton methods. See [1, p. 56].

Uniqueness-proving properties of interval Newton methods are based on proving that each point matrix formed elementwise from the interval matrix $\mathbf{f}'(\mathbf{x})$ is nonsingular.

Example 1 For an example of a multivariate interval Newton method, take

$$f_1(x) = x_1^2 - x_2^2 - 1,$$
$$f_2(x) = 2x_1x_2,$$

with

$$\mathbf{x} = \begin{pmatrix} [0.9, 1.2] \\ [-0.1, 0.1] \end{pmatrix}, \qquad \check{x} = \begin{pmatrix} 1.05 \\ 0 \end{pmatrix}.$$

An interval extension of the Jacobi matrix for f is

$$\mathbf{f}'(\mathbf{x}) = \begin{pmatrix} 2\mathbf{x}_1 & -2\mathbf{x}_2 \\ 2\mathbf{x}_2 & 2\mathbf{x}_1 \end{pmatrix},$$

and its value at $\mathbf{x}$ is

$$\begin{pmatrix} [1.8, 2.4] & [-0.2, 0.2] \\ [-0.2, 0.2] & [1.8, 2.4] \end{pmatrix}.$$

The usual procedure (although not required in this special case) is to precondition the system

$$\mathbf{f}'(\mathbf{x})\mathbf{v} = -f(\check{x}),$$

say, by the inverse of the midpoint matrix

$$Y = \begin{pmatrix} 2.1 & 0 \\ 0 & 2.1 \end{pmatrix}^{-1} = \begin{pmatrix} 0.476 & 0 \\ 0 & 0.476 \end{pmatrix}$$

to obtain

$$Y\mathbf{f}'(\mathbf{x})\mathbf{v} = -Yf(\check{x}),$$

i. e., rounded out,

$$\begin{pmatrix} [0.85, 1.15] & [-.096, .096] \\ [-.096, .096] & [0.85, 1.15] \end{pmatrix}\mathbf{v} = \begin{pmatrix} [-.0488, 0.487] \\ 0 \end{pmatrix}.$$

(Rigor is not lost by taking floating point approximations for the preconditioner, but the interval arithmetic should be outwardly rounded.) The interval Gauss–Seidel method can then be used to compute sharper bounds on $\mathbf{v} = \mathbf{x} - \check{x}$, beginning with $\mathbf{v} = \begin{pmatrix} [-0.15, 0.15] \\ [-0.1, 0.1] \end{pmatrix}$. That is,

$$\widetilde{\mathbf{v}}_1 \subset \frac{[-0.0488, -0.0488] - [-0.096, 0.096]\mathbf{v}_2}{[0.85, 1.15]} \subset [-0.0688, -0.034].$$

Thus, the first component of $\mathbf{N}(f, \mathbf{x}, \check{x})$ is

$$\check{x} + \mathbf{v} \subset [0.9833, 1.016].$$

In the second step of the interval Gauss–Seidel method,

$$\widetilde{\mathbf{v}}_2 = \frac{0 - [-0.096, 0.096]\widetilde{\mathbf{v}}_1}{[0.085, 1.15]} \subset [-0.00778, 0.00778],$$

so, rounded out, $\mathbf{N}(f, \mathbf{x}, \check{x})$ is computed to be

$$\begin{pmatrix} [0.981, 1.016] \\ [-0.00778, 0.00778] \end{pmatrix} \subset \begin{pmatrix} [0.9, 1.2] \\ [-0.1, 0.1] \end{pmatrix}.$$

This last inclusion proves that there exists a unique solution to $f(x) = 0$ within $\mathbf{x}$, and hence, within $\mathbf{N}(f, \mathbf{x}, \check{x})$. Furthermore, iteration of the procedure will result in bounds on the exact solution that become narrow quadratically.

See also

- Automatic Differentiation: Calculation of Newton Steps
- Automatic Differentiation: Point and Interval
- Automatic Differentiation: Point and Interval Taylor Operators
- Bounding Derivative Ranges
- Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- Global Optimization: Application to Phase Equilibrium Problems
- Interval Analysis: Application to Chemical Engineering Design Problems
- Interval Analysis: Differential Equations
- Interval Analysis: Eigenvalue Bounds of Interval Matrices
- Interval Analysis: Intermediate Terms
- Interval Analysis: Nondifferentiable Problems
- Interval Analysis: Parallel Methods for Global Optimization
- Interval Analysis: Subdivision Directions in Interval Branch and Bound Methods
- Interval Analysis: Systems of Nonlinear Equations
- Interval Analysis: Unconstrained and Constrained Optimization
- Interval Analysis: Verifying Feasibility
- Interval Constraints
- Interval Fixed Point Theory
- Interval Global Optimization
- Interval Linear Systems
- Nondifferentiable Optimization: Newton Method
- Unconstrained Nonlinear Optimization: Newton–Cauchy Framework

References

1. Kearfott RB (1996) Rigorous global search: continuous problems. Kluwer, Dordrecht

Inventory Management in Supply Chains

IM in SC

Sandra Duni Eksioglu
Industrial and Systems Engineering Department, University Florida, Gainesville, USA

MSC2000: 90B50

Article Outline

Keywords
Single Stage Inventory Management Models
Multistage Inventory Management Models
Conclusions
See also
References

Keywords

Inventory management; Multistage inventory management; Supply chain; EOQ; Newsboy problem; (s, S)policy; Periodic review model; Continuous review model; Metric

A supply chain (SC) can be defined as an integrated system, where various firms work together, including suppliers of raw materials, manufacturers, distributors and retailers. Their efforts are concentrated on transforming the raw materials into final products that satisfy customer requirements, and delivering these products to the right place, at the right time. A SC contains two basic, integrated processes:

a) production planning and inventory management (IM); and
b) distribution and logistics processes [6].

This article gives a brief review of literature on *single-stage IM* and *multistage IM* models. The objective is to provide an overview of this research and emphasize current achievements in this field. Inventories exist

throughout the SC in the form of raw materials, work-in-process, and finished goods. Typical relevant inventory costs are: inventory carrying costs, order costs, and shortage costs. These costs often tend to conflict, in other words, decreasing one generally requires increasing another. The main motivation for keeping inventories is to cope with the uncertainty of external demand, supply and lead-time [18]. Keeping inventories is important to increase customer service level and reduce distribution costs, but it is estimated [5] that inventories cost approximately 20% to 40% of their value per year. Thus, managing inventories in a scientific manner to maintain minimal levels required for meeting service objectives makes economic sense. K. Arrow [2] presents an interesting discussion of the motives of a firm for holding inventories. There are several opportunities for streamlining SC inventories. It is important to understand that for a given service level the lowest inventory investment results when the entire SC is considered as a single system. Such coordinated decisions at Xerox and Hewlett Packard reduced their inventory levels by over 25% [9].

Single Stage Inventory Management Models

The simplest inventory model is the deterministic economic order quantity (EOQ) model presented by F. Harris [12]. He recognized this problem in 1913 in his work at Westinghouse. The model determines the constant order quantity that minimizes the average annual cost of purchasing and carrying inventory, assuming deterministic and constant demand rate, no shortages, and zero order lead-times. A number of important scholars turned their attention to mathematical inventory models during the 1950s. A collection of mathematical models by Arrow, S. Karlin and H.E. Scarf [3] influenced later work in this area. At about the same time, H.M. Wagner and T.M. Whitin [24] developed a solution algorithm to the dynamic lot-sizing problem subject to time varying demand. Their model assumes periodic, deterministic demand over a finite planning horizon, no capacity restrictions on production, and zero inventory at the beginning and the end of the planning horizon. This problem is formulated as a mixed integer linear program (MILP) and can be represented as a fixed-charge network flow problem. The Wagner–Whitin algorithm is best illustrated using a shortest-path graph representation. Although the Wagner–Whitin model gives an optimal solution, in practice other heuristic lot-sizing algorithms are adopted. See [18] for a survey on the EOQ lot-sizing, silver-meal, least unit cost heuristics, etc. These models trade-off productivity losses from making small batches and the opportunity costs of tying up capital in inventory due to large batches. U.S. Karmarkar [14] extends the lot-sizing model to include lead-time related costs. Inventory control models subject to uncertain demand are basically of two types: *periodic review models* and *continuous review models*. Periodic review models exist for one planning period or for multiple planning periods. The single-period, stochastic inventory model is known as the *newsboy model*. The case of single period models with fixed order cost and initial inventories, leads to the optimality of (s, S) *optimal policies*. These policies state that if inventory position is less than s, then order up to S, otherwise do not order. The periodic review models with an infinite horizon are formulated in a dynamic programming framework [23]. Continuous review systems under uncertain demand track demands as they occur and the inventory position is always known. These models lead to the (Q, R) *policy*, under which a fixed amount of Q units is ordered each time the inventory position reaches a certain level R. The model typically assumes either backordering or lost sales when shortages occur.

Multistage Inventory Management Models

Coordinating decisions at different levels of an organization comes as a need to reduce operating costs. This coordination can be seen in terms of integrating different decision types e. g., facility location, inventory planning, distribution, etc., or linking decisions within the same function at different stages in the SC. Multistage inventory management models (MSIM models) concentrate on integrating IM policies in different stages of the SC. The typical MSIM problem analyzed in the literature is a two-level system composed of a number of retailers being served by a central warehouse. The demand at each retailer is satisfied using on-hand inventory. When insufficient inventory is available, a backorder typically occurs, and demand must be satisfied later using inventory from the warehouse. The model decides on the inventory level at each retailer and the

warehouse, such that a set of prespecified criteria is satisfied at minimum inventory-related costs. The first MSIM model was developed by A. Clark and Scarf [7]. They consider a system with a single product and N facilities, where facility i supplies facility $i + 1$, for $i = 1, \ldots, N - 1$. The model considers a periodic review of the inventory level and assumes fixed lead-times, a finite planning horizon, backordering of demand shortages and variable order cost. The aim is to find IM policies to be applied in each of the echelons, such that system cost is minimized. They show that under the above assumptions an optimal policy for the system can be found by decomposing the problem into N separate single-location problems and solving the problem recursively. The above model was extended to incorporate an infinite horizon and lead time uncertainty. A generalization of the system described above is the *multi-echelon arborescence system*, where each location has a unique supplier. A.F. Veinott [23] provides an excellent summary of these early modeling efforts. One of the earliest continuous review MSIM models was presented by C.C. Sherbrooke [21]. He considers a two-stage system with several retailers and a single warehouse that supplies to these retailers. He introduces the well-known METRIC approximation to determine the optimal level of inventory in the system. The METRIC approximation assumes a Poisson distribution of demand and constant replenishment lead-times. S.C. Graves [11] extends the METRIC approximation by estimating the mean and the variance of the outstanding retailer orders. He fits the negative binomial distribution to these parameters to determine the optimal inventory policy. S. Axsäter} [4] provides an exact solution to the problem and shows that the METRIC approximation provides an underestimate, whereas Graves ' two-parameter approximation [11] overestimates the retailer backorders. The above studies use the *one-for-one ordering policy* $(S - 1, S)$, i. e., an order is placed as soon as a demand occurs. This policy is appropriate for items with high value and a low demand rate. Axsäter [4] shows that the models used for the one-for-one ordering policy can be extended in the case of batch ordering with only one retailer. Analysis of batch ordering policies in arborescent systems (when the number of retailers is greater than one) is similar to Sherbrooke 's model. B.L. Deuermeyer and L.B. Schwarz [8] were the first to analyze such a system. They estimate the mean and the variance of lead-time demand to obtain average inventory levels and backorders at the warehouse, assuming that lead-time demand is normally distributed. The retailer lead-time demand is also approximated using a normal distribution. In addition to reviewing the literature in the area, [15,17] and [22] also provide several extensions to the Deuermeyer and Schwarz model.In [10] the concept of stability in a capacitated, multi-echelon production-inventory system under a base-stock policy is introduced. W.L. Maxwell and others [16] extend the analysis to multiproduct, continuous review and deterministic demand, MSIM problems. Their model tends to schedule the orders for each of the products over an infinite horizon so as to minimize the long-run average cost. The authors define a new class of policies in which each product uses a stationary interval of time between successive orders. Their model finds a lot-sizing rule that is within 6% of the average cost of the optimal policy. R. Roundy [19] develops a similar multistage, multiproduct lot-sizing model. Under the assumption that the ratio of the order intervals of any two products is an integer power of two, it is shown that the solution is within 2% of optimality.

D. Sculli and others [20] extend the analysis of MSIM systems for the case when two suppliers are used to replenish stock of a single item. They calculate the mean and the standard deviation of the effective lead time demand and interarrival time when replenishment orders are placed at the same time with the two suppliers, in a continuous review system. The lead-time distribution of each supplier is assumed to be normal. R. Ganeshan [9] presents a near-optimal (s, Q) inventory policy for a production/distribution network with multiple suppliers replenishing a central warehouse, which distributes to a large number of retailers. The model concentrates on inventory analysis at the retailers and the warehouse, and demand process at the warehouse. The model finds a near-optimal order quantity and a reorder point at both the retailer and the distribution center (DC) under stochastic demand and lead-time, subject to customer service constraints. The main contribution of this model is the integration of the above components for analyzing simple supply chains. P. Afentakis and others [1] develop a procedure for optimally solving medium size lot-scheduling problems in multistage structures with periodic review of the inventory and dynamic deterministic demand. They formu-

late the problem in terms of *echelon stock*, which simplifies its decomposition by Lagrangian relaxation. An efficient *branch and bound* algorithm is used to solve the problem.

M.C. van der Heijden and others [13] consider the periodic review, order-up-to (R, S) inventory system under stochastic demand. They propose a new approach to calculate the mean physical stock. The standard approximation appears to yield inaccurate results in the case of low service levels. Low service levels usually occur at intermediate nodes in optimal solutions for multi-echelon systems.

Conclusions

With the trend toward just-in-time deliveries and reduction of inventories, many firms are reexamining their inventory and logistics policies. Some firms are altering their inventory, production and shipping policies, and others are working on coordinating inventory decisions throughout their SC, with the goal of reducing costs and improving service. Single stage IM models give some insights on how to manage inventories under certain demand and lead-time considerations, while MSIM models take the analysis further, coordinating inventory decisions throughout the SC. This article reviews the literature on single-stage and multi-stage inventory management models, with an emphasis on achievements in this field.

See also

- ► Global Supply Chain Models
- ► Nonconvex Network Flow Problems
- ► Operations Research Models for Supply Chain Management and Design
- ► Piecewise Linear Network Flow Problems

References

1. Afentakis P, Gavish B, Karmarkar U (1984) Computationally efficient optimal solutions to the lot-sizing problem in multistage assembly systems. Managem Sci 30(2):222–239
2. Arrow KA (1958) Historical background. In: Arrow KA, Karlin S, Scarf HE (eds) Studies in the Math. Theory of Inventory and Production. Stanford Univ. Press, Palo Alto, CA
3. Arrow KA, Karlin S, Scarf HE (eds) (1958) Studies in the mathematical theory of inventory and production. Stanford Univ. Press, Palo Alto, CA
4. Axsäter S (1993) Continuous review policies for multi-level inventory system with stochastic demand. In: Graves SC, Rinnooy Kan AHG, Zipkin P (eds) Handbook Oper. Res. and Management Sci.: Logistics of Production and Inventory, vol 4. North-Holland, Amsterdam, pp 175–197
5. Ballou RH (1992) Business logistics management. Third edn. Prentice-Hall, Englewood Cliffs, NJ
6. Beamon BM (1998) Supply chain design and analysis: Models and methods. Internat J Production Economics 55:281–294
7. Clark A, Scarf H (1960) Optimal policies for a multi-echelon inventory problem. Managem Sci 6:475–490
8. Deuermeyer BL, Schwarz LB (1981) A model for the analysis of system service level in warehouse retailer distribution systems: The identical retailer case. In: Schwarz LB (ed) Studies in Management Sci.: Multi-Level Production/Inventory Control Systems: Theory and Practice, vol 16. North-Holland, Amsterdam, pp 163–193
9. Ganeshan R (1999) Managing supply chain inventories: A multiple retailer, one warehouse, multiple supplier model. Internat J Production Economics 59:341–354
10. Glasserman P, Tayur S (1994) The stability of a capacitated, multi-echelon production-inventory system under a base-stock policy. Oper Res 42(5):913–926
11. Graves SC (1985) A multi-echelon inventory model for a repairable item with one-for-one replenishment. Managem Sci 31(10):1247–1256
12. Harris FW (1913) How many parts to make at once. Factory, The Magazine of Management 10(2):135–136; 152
13. Heijden MC van der, Kok Tde (1998) Estimating stock levels in periodic review inventory systems. Oper Res Lett 22:179–182
14. Karmarkar US (1987) Lot sizes, lead times and in-process inventories. Managem Sci 33(3):409–418
15. Lee H, Moinzadeh K (1987) Two-parameter approximations for multi-echelon repairable inventory models with batch ordering policy. IIE Trans 19:140–147
16. Maxwell WL, Muckstadt JA (1985) Establishing consistent and realistic reorder intervals in production-distribution systems. Oper Res 33:1316–1341
17. Moinzadeh K, Lee H (1986) Batch size and stocking levels in multi-echelon repairable systems. Managem Sci 32(12):1567–1581
18. Nahmias S (1997) Production and operations analysis, 3rd edn. IRWIN Publ., Homewood, IL
19. Roundy R (1986) A 98%-effective lot-sizing rule for a multi-product, multi-stage production/inventory system. Math Oper Res 11(4):699–727
20. Sculli D, Wu SY (1981) Stock control with two suppliers and normal lead times. J Oper Res Soc 32:1003–1009
21. Sherbrooke CC (1968) METRIC: A multi-echelon technique for recoverable item control. Oper Res 16:122–141
22. Svoronos A, Zipkin P (1988) Estimating the performance of multi-level inventory systems. Oper Res 36(1):57–72

23. Veinott AF (1966) The status of mathematical inventory. Managem Sci 12:745–775
24. Wagner HM, Whitin TM (1958) Dynamic version of the economic lot size model. Managem Sci 5(1):89–96

Invexity and its Applications

B. D. Craven
University Melbourne, Melbourne, Australia

MSC2000: 90C26

Article Outline

Keywords
See also
References

Keywords

Optimization; Invex; Lagrangian conditions; Constrained minimization; Scale function; E-convex; Wolfe dual; Mond–Weir dual; Generalized invex; V-invex; Convexifiable; Protoconvex; Quasi-invex; Pseudo-invex; Pseudoconvex; Quasiconvex; Basic alternative theorem; Convex-like; Optimal control; Continuous programming

Lagrangian conditions are often necessary, but not sufficient, for a minimum of an optimization problem in continuous variables. Sufficiency holds under convex assumptions, which are however often not satisfied in applications. Invexity is a less restrictive assumption than convexity, under which Lagrangian conditions are sufficient for a minimum, and also related duality results hold. It also provides a structure, showing relations with various other kinds of generalized convexity.

Consider a *constrained minimization problem*

$$\text{(P1)} \quad \begin{cases} \min & f(x), \\ \text{s.t.} & g_j(x) \le 0, \\ & j = 1, \dots, m, \end{cases}$$

where the functions f and g_j are differentiable. Define the Lagrangian $L(\cdot;\lambda) := f(\cdot) + \Sigma\ \lambda_j\ g_j(\cdot)$, where $\lambda = (\lambda_1, \dots, \lambda_m)$ is a vector of nonnegative Lagrange multipliers. When p is a feasible point, define also a reduced Lagrangian $L_{(p)}(\cdot;\lambda)$ obtained by omitting constraints inactive at p (thus, when $g_j(p) < 0$.) A minimum point p for (P1), assuming some regularity for the constraints, is then a *Karush–Kuhn–Tucker* (KKT) point, namely one where the gradient of the Lagrangian with respect to x satisfies $L_{(p)}{}'(p;\lambda) = 0$ for some $\lambda \ge 0$. However, a KKT point is not generally a minimum point. It is, in particular, if the functions f and each g_k are convex. However, convexity often does not hold in applications, and less restrictive conditions are sought when a KKT point is a minimum.

A differentiable vector function $F := (f, g_1, \dots, g_m)$, with gradient F', is called *invex* if, for some *scale function* η, and all x and p,

$$\text{(INV)} \quad F(x) - F(p) \ge F'(p)\eta(x,p).$$

This property was first called *η-convex* by M.A. Hanson [11].

Usually, at a given p, $\eta(x,p) = (x-p) + o(\| x-p \|)$. In particular, F is convex if $\eta\ (x,p) = x - p$. If F is invex and $\lambda \ge 0$, then it follows that $L_{(p)}(\cdot;\lambda)$ is invex, so if x is a feasible point, then

$$\begin{aligned} f(x) - f(p) &\ge L_{(p)}(x,\lambda) - L_{(p)}(p,\lambda) \\ &\ge L'_{(p)}(p;\lambda)\eta(x,p) = 0, \end{aligned}$$

from KKT, so that p is a minimum point of (P1). This minimum is global if (INV) holds globally, otherwise local. (But how can (INV) be verified, as a global property?)

By a similar argument, since $L(\cdot;v)$ is invex when $v \ge 0$, duality holds for (P1) and the *Wolfe dual* problem:

$$\text{(D1)} \quad \begin{cases} \max & f(u) + \sum v_j g_j(u) \\ \text{s.t.} & f'(u) + \sum v_j g_j'(u) = 0, \\ & v_1 \ge 0, \dots, v_m \ge 0, \end{cases}$$

if KKT holds for (P1) and the invex property holds (see [17]). Duality means that $f(x) \ge f(u) + \sum v_j g_j(u)$ whenever x and u, v are feasible for their respective problems, and also that the optimum objectives are equal. Consider also the *Mond–Weir dual* [18]:

$$\text{(D2)} \quad \begin{cases} \max & f(u) \\ \text{s.t.} & f'(u) + \sum v_j g_j'(u) = 0, \\ & v_1 \ge 0, \dots, v_m \ge 0, \\ & \sum v_j g_j(u) \ge 0. \end{cases}$$

Assume that F is invex. If x is feasible for (P1) and (x, u) for (D2), then

$$\begin{aligned} f(x) - f(u) &\geq f'(u)\eta(x,u) \\ &= -\sum \lambda_j g_j'(u)\eta(x,u) \\ &\geq \sum(-\nu_j g_j(x) + \nu_j g_j(u) \geq 0, \end{aligned}$$

and this, with KKT, proves duality.

If the vector function F is locally Lipschitz, rather than differentiable, then (INV) is replaced by *generalized invex* (see [6]):

$$\text{(INV2)} \quad F(x) - F(p) \geq F^{\circ}(p, \eta(x,p)).$$

where $F^{\circ}(p, d)$ denotes Clarke 's generalized directional derivative (see [2]) of F at p in direction d. Most properties of (INV) extend to (INV2). In particular, if F_1, $F_2, \cdots$ are (generalized) invex functions with the same scale function η, then so also are $\max_j F_j(\cdot)$, any element of the convex hull co $F_j(\cdot)$, and (if it exists) $\lim_j F_j(\cdot)$. But the assumption of the same η is necessary here.

Suppose now that the following *V-invex* property (see [14]) holds for $g_0(\cdot) := f(\cdot) - f(p)$ and for the active constraints $g_j(\cdot)$ (those for which $g_j(p) = 0$):

$$\forall x: \quad g_j(x) - g_j(p) \geq \beta_j(x,p) g_j'(p)\eta(x,p).$$

Note that $\eta(\cdot,\cdot)$ is a vector function, the same for each j, and the *weight* $\beta_j(\cdot,\cdot)$ is a positive scalar function. If KKT holds, and if x is feasible for (P1), then setting $\mu_j := \lambda_j/\beta_j$, the minimum follows from

$$\begin{aligned} f(x) - f(p) &\geq L_{(p)}(x;\mu) - L_{(p)}(p;\mu) \\ &\geq \sum \beta_j \mu_j g_j'(p)\eta(x,p) = 0. \end{aligned}$$

In the problem (P1), set $G_j(x) := r_j(x)\, g_j(x)$, where $r_j(\cdot) > 0$; then $g_(\ x) \leq 0$ if and only if $G_j(x) \leq 0$ ($j = 1, \ldots, m$), and $f(x) \geq f(p)$ if and only if $G_0(x) \geq 0$. So (P1) is equivalently formulated in terms of the weighted constraint functions G_j. For each active constraint $g_j(x) \leq 0$ with g_j invex,

$$G_j(x) - G_j(p) = G_j(x) \geq \beta_j(x,p) G_j'(p)\eta(x,p),$$

where the weight $\beta_j(x,p) = r_j(x)/r_j(p)$. Thus the G_j have the V-invex property. Note also that if n and d are real functions with $n(\cdot) \geq 0$ and $d(\cdot) > 0$, and n and $-d$ are invex with the same same scale function, then [14] the ratio $n(\cdot)/d(\cdot)$ is V-invex with the same scale function, and weight $\beta(x,p) = d(p)/d(x)$.

Invexity for (P1) can also be relaxed to the requirement, called *Type I* in [12]:

$$\begin{aligned} &f(x) - f(p) \geq f'(p)\eta(x,p); \\ &-g_j(p) \geq g_j'(p)\eta(x,p), \quad \forall j \in J, \end{aligned}$$

where J is the set of indices of constraints active at p. If this property holds at a KKT point p, and $g_j(x) \leq 0$ ($\forall j$), then

$$f(x) - f(p) \geq -\sum_{j\in J} \lambda_j g_j'(p)\eta(x,p) \geq 0,$$

thus p is a minimum for (P1).

Invexity is related as follows to some other properties. The vector function F: $\mathbf{R}^n \to \mathbf{R}^r$ is *convexifiable* if $H := F \circ \phi^{-1}$ is convex, for some invertible transformation ϕ: $\mathbf{R}^n \to \mathbf{R}^n$. For $0 < \alpha < 1$,

$$\begin{aligned} \text{(CL)} \quad &(1-\alpha)F(p) + \alpha F(x) \\ &= (1-\alpha)H(\phi(p)) + \alpha H(\phi(x)) \\ &\geq H((1-\alpha)\phi(p) + \alpha\phi(x)) \\ &= F(\xi(\alpha,x,p)) \end{aligned}$$

if H is convex, where

$$\text{(K)} \quad \xi(\alpha,x,p) := \phi^{-1}((1-\alpha)\phi(p) + \alpha\phi(x)).$$

This reduces, for a convex function F, to $\xi(\alpha, x, p) = (1 - \alpha)\, p + \alpha\, x$. If ϕ is differentiable, then also

$$\begin{aligned} \text{(K2)} \quad &\frac{\partial}{\partial\alpha}\xi(\alpha,x,p)|_{\alpha=0} \\ &= \phi^{-1'}(\phi(p))[\phi(x) - \phi(p)] \equiv \eta(x,p). \end{aligned}$$

Hence, letting $\alpha \downarrow 0$, invexity follows. Thus,

$$\text{Convexifiable} \quad \Rightarrow \quad \text{(CL)} + \text{(K)} \quad \Rightarrow \quad \text{Invex},$$

with the second implication assuming differentiable functions. The name 'invex' was given [4], from *invariant convex*, since invex preserves that part of the convex property that is invariant to the transformation ϕ. (See also [1].) The property (CL), together with the existence of $(\partial/\partial\alpha)\, \xi(\alpha, x, p)|_{\alpha=0}$, was called *protoconvex* in [10]. It holds, in particular, if $\xi(\alpha, x, p) = p + \alpha\, \phi\, (x - p)$ holds in (CL). If F is locally Lipschitz, then (CL) and

(K2) for all x and p in a domain imply generalized invex. Note that invex does not imply convexifiable. A counterexample (see [8] is $F(x) := (-x_1x_2 + 1, -x_1x_2 + 1)$.

Suppose that f is invex, with scale function η, and that ϕ is a differentiable invertible transformation of the domain space. An application of the chain rule shows that $f \circ \phi$ is also invex, with a new scale function

$$\hat{\eta}(\alpha, \beta) = \phi'(\phi(\alpha))^{-1}\eta(\phi(\beta), \phi(\alpha)).$$

This invariance of invexity to domain transformations extends to two relaxed properties (see [15,18]), defined by:

- *quasi-invex*:

$$f(x) \le f(p) \quad \Rightarrow \quad f'(p)\eta(x, p) \le 0;$$

- *pseudo-invex*

$$f(x) < f(p) \quad \Rightarrow \quad f'(p)\eta(x, p) < 0.$$

These reduce to *quasiconvex* (respectively *pseudoconvex*; see [16,19]) when $\eta(x, p) = x - p$. If f is quasi-invex (respectively, pseudo-invex) with scale function η, then $f \circ \phi$ is quasi-invex (respectively pseudo-invex) with scale function $\widehat{\eta}$. Note that each pseudoconvex real function is invex, but not conversely.

In (P1), if f is pseudo-invex, and each g_j is quasi-invex, all with the same scale function η, then KKT is sufficient for a minimum. For each active constraint,

$$g_j(x) \le 0 = g_j(p) \quad \Rightarrow \quad g_j'(p)\eta(x, p) \le 0.$$

Then $\sum \lambda_j g_j'(p)\, \eta(x, p) \le 0$, hence KKT gives $f'(p)\, \eta(x, p) \ge 0$. From pseudoconvexity, $f(x) \ge f(p)$, proving the minimum. There are various results (see [18,21]) showing sufficiency of KKT for a minimum, when various combinations of the functions f and g_j have specified pseudo-invex and quasi-invex properties, all with the same scale function η. Some further examples of pseudo-invex functions are given in [15] (they called η-*pseudoconvex*.)

The property (CL) is called *convex-like* (see [13]). If $\Gamma \subset \mathbf{R}^n$ is a convex set, F is (CL), and Q is the orthant $\mathbf{R}_+^{m+1}$, then $F(\Gamma) + Q$ is a convex set. For, taking $x, p \in \Gamma$ and $q, r \in Q$, with $0 < \alpha < 1$,

$$(1-\alpha)[F(p) + q] + \alpha[F(x) + r] - [F(\xi(\alpha, x, p) + (1-\alpha)q + \alpha r] \in Q.$$

From this follows (see [3,8,13]) the *basic alternative theorem* that

$$\text{(BAT)} \qquad \begin{array}{c} \neg(\exists x \in \Gamma):\ F(x) < 0 \\ \Leftrightarrow \quad (\exists 0 \ne \rho \ge 0):\ \rho F(\cdot) \ge 0. \end{array}$$

Consider (P1) with inactive constraints omitted. If (CL) holds, then (from (BAT)) (P1) reaches a minimum at p if and only if there are nonnegative multipliers τ and λ, not both zero, for which

$$\tau[f(x) - f(p)] + \lambda g(x) \ge 0, \quad \forall x.$$

If *Slater's constraint* qualification holds, that $g(c) < 0$ for some c, then $\tau = 1$ can be assumed. If f and g are directionally differentiable, then the directional derivatives satisfy

$$f'(p; d) + \lambda g'(p; d) \ge 0$$

for each direction d. If f and g are Lipschitz functions and (CL) holds, then

$$0 \in \partial(f + \lambda g)(p), \quad \lambda g(p) = 0,$$

is necessary and sufficient for a minimum, where ∂denotes here Clarke 's subdifferential [2].

When is a vector function invex at a point p? Assume now twice differentiable functions, and expand

$$F(x) = F(p) + F'(p)v + \frac{1}{2}v^{\top}F''(p).v + \cdots,$$
$$\eta(x, p) = v + \frac{1}{2}v^{\top}Q.v + \cdots,$$

where $v = x - p$, and $v^{\intercal}F''(p).v$ means that component j of F has second order term $v^{\intercal}F_j''(p)\ v$, and similarly for $v^{\intercal}Q.v$; denote the matrix component k of $Q.$ by Q_k. Then ([5]), by substituting in (INV), local invexity implies that

$$F''(p)_s - \sum_k F'(p)_{sk}Q_k$$

is positive semidefinite, for each s. Conversely, if each of these matrices is positive definite, then F is locally invex at p.

Some further classes of invex functions are described as follows (see [9]). Let X_0 be an open domain in $\mathbf{R}^n$, let $A: X_0 \to \mathbf{R}^m$ be convex, and let $B: X_0 \to \mathbf{R}$ be differentiable and satisfy $B(X_0) \subset (0, \infty)$. Then $A(\cdot)/B(\cdot)$

is invex if also B is convex with $A(X) \subset -\mathbf{R}^m_+$, or if B is concave with $A(X_0) \subset \mathbf{R}^m_+$. If g: $X_0 \to \mathbf{R}^m$ is differentiable, and $g'(a)\, d < 0$ for some direction d, then g is invex at the point a.

Let ϕ: $\mathbf{R}^n \to \mathbf{R}^n$ be an invertible C^2 mapping; let r: $\mathbf{R}_+ \to \mathbf{R}_+$ be strictly increasing, with $r(0) = 0$, $r'(0) = 0$, and $r''(0) < 0$ on some interval. Then r and $h(\cdot) := r \circ \|\cdot\|$ are pseudoconvex, and $h \circ \phi$ is pseudo-invex (hence also quasi-invex).

The invex property, and also pseudo-invex, quasi-invex and V-invex, are also applicable when the spaces $\mathbf{R}^n$ and $\mathbf{R}^m$ are replaced by infinite-dimensional normed spaces of functions, such as occur in *optimal control* (see [3,8]) and *continuous programming* (see [7,20]). The definitions, and proofs of basic properties (see [3,7,8]), are unchanged, interpreting $a \geq b$ as $a - b \in Q$, where the *order cone* Q is a closed convex cone. Examples of spaces of control functions are the spaces $C(\mathbf{R}^r)$ (respectively, $PC(I, \mathbf{R}^r)$) of continuous (respectively, piecewise continuous) functions from an interval I into $\mathbf{R}^r$, with the uniform norm, and the space $L^2(I, \mathbf{R}^r)$ of square-integrable functions from I into $\mathbf{R}^r$). Consider, for example, an integral objective function $f(x) := \int_0^T \phi(x(t), \dot{x}(t), t)\, dt$, where $f \in C^1(0, T)$, ϕ is differentiable, and $\dot{x}(t) = (\frac{d}{dt})x(t)$. Assume boundary conditions $x(0) = x_0$, $x(T) = x_T$. Denote $\phi_x(x(t), \dot{x}(t), t) := (\frac{\partial}{\partial x})\phi(x(t), t)$, and similarly $\phi_{\dot{x}}$. Then the gradient $f'(p)$ of f at $p \in C^1[0, T)$ is given by

$$f'(p)z = \int_0^T \phi_x(p(t), \dot{p}(t), t)z(t)\, dt$$
$$= \int_0^T \left[\phi_x(p(t), \dot{p}(t), t) - \frac{d}{dt}\phi_{\dot{x}}(p(t), \dot{p}(t), t)\right] \times z(t)\, dt$$

after integrating by parts. Then f is *imvex* if, for some scale function η,

$$f(x) - f(p)$$
$$\geq \int_0^T \left[\phi_x(p(t), \dot{p}(t), t) - \frac{d}{dt}\phi_{\dot{x}}(p(t), \dot{p}(t), t)\right] \times \eta(x(t), p(t), t)\, dt.$$

For a constraint $\psi\ (x(t), t) \leq 0$ $(\forall t \in [0, T])$, the analog of the term $\sum \lambda_j\, g_j(x)$ in the Lagrangian is

$$\int_0^T \lambda(t)\psi(x(t), t)\, dt,$$

and invexity requires that

$$\int_0^T \lambda(t)\left[\psi(x(t), t) - \psi(p(t), t)\right] dt$$
$$\geq \int_0^T \lambda(t)\psi_x(p(t), t)\eta(x(t), p(t), t)\, dt.$$

There are converse KKT and duality properties for such infinite-dimensional problems (see e. g. [8,20]), using invexity quite similarly to finite-dimensional problems.

See also

- ▶ Generalized Concavity in Multi-objective Optimization
- ▶ Isotonic Regression Problems
- ▶ L-convex Functions and M-convex Functions

References

1. Ben-Israel A, Mond B (1986) What is invexity? J Austral Math Soc (Ser B) 22:1–9
2. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley/Interscience, New York
3. Craven BD (1978) Mathematical programming and control theory. Chapman and Hall, London
4. Craven BD (1981) Duality for generalized convex fractional programs. In: Generalized Concavity in Optimization and Economics. Acad. Press, New York, pp 473–489
5. Craven BD (1981) Invex functions and constrained local minima. Bull Austral Math Soc 24:357–366
6. Craven BD (1986) Nondifferentiable optimization by smooth approximations. Optim 17:3–17
7. Craven BD (1993) On continuous programming with generalized convexity. Asia–Pacific J Oper Res 10:219–231
8. Craven BD (1995) Control and optimization. Chapman and Hall, London
9. Craven BD, Glover BM (1985) Invex functions and duality. J Austral Math Soc (Ser A) 39:1–20
10. Craven BD, Luu DV (1997) Optimization with set-functions described by functions. Optim 42:39–50
11. Hanson MA (1980) On the sufficiency of the Kuhn–Tucker conditions. J Math Anal Appl 80:545–550
12. Hanson MA, Mond B (1987) Necessary and sufficient conditions in constrained optimization. Math Program 37:51–56
13. Jeyakumar V (1985) Convexlike alternative theorems and mathematical programming. Optim 16:643–652
14. Jeyakumar V, Mond B (1992) On generalized convex mathematical programming. J Austral Math Soc (Ser B) 34:43–53
15. Kaul RN, Kaur S (1985) Optimality criteria in nonlinear programming involving nonconvex functions. J Math Anal Appl 105:104–112

16. Mangasarian OL (1969) Nonlinear programming. McGraw-Hill, New York
17. Mond B, Hanson MA (1984) On duality with generalized convexity. Math Operationsforsch Statist Ser Optim 15:313–317
18. Mond B, Weir T (1981) Generalized concavity and duality. In: Generalized Concavity in Optimization and Economics. Acad. Press, New York, pp 263–275
19. Mond B, Weir T (1982) Duality for fractional programming with generalized convex conditions. J Inform Optim Sci 3:105–124
20. Weir T, Hanson MA, Mond B (1984) Generalized concavity and duality in continuous programming. J Math Anal Appl 104:212–218
21. Weir T, Mond B (1988) Pre-invex functions in multiple objective optimization. J Math Anal Appl 137:29–38

Isotonic Regression Problems

VALENTINA DE SIMONE, MARINA MARINO, GERARDO TORALDO
University Naples 'Federico II' and CPS, Naples, Italy

MSC2000: 62G07, 62G30, 65K05

Article Outline

Keywords
Synonyms
Problem Statement
The Pool Adjacent Violators Algorithm
Minimum Lower Set Algorithm
Other Algorithms
See also
References

Keywords

Order restriction; Algorithms; Optimization

Synonyms

IR

Problem Statement

Given a finite set X with an ordering $\preccurlyeq$, a real function g on X and a positive weight function w on X, the *isotonic regression problem* is to find a function g^* which minimizes

$$\sum_{x\in X}[g(x)-f(x)]^2 w(x),$$

among the class F of isotonic functions f defined on X, i. e.

$$F=\{f:\ \forall x,y\in X \text{ and } x\preceq y \Rightarrow f(x)\le f(y)\}.$$

The function g^* is called *isotonic regression*, and it exists and is unique [5].

Isotonic regression can be viewed as a *least squares problem* under order restrictions; here, order restrictions on parameters can be regarded as requiring that the parameter, as a function of an index, will be isotonic (the adjective 'isotonic' is used as a synonym for 'order preserving') with respect to an order on the index set.

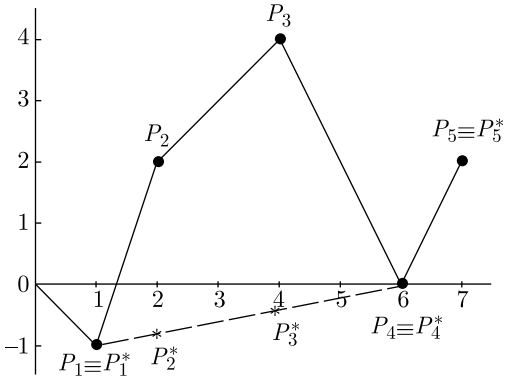
If $\preccurlyeq$ is reflexive, transitive, antisymmetric and every pair of elements are comparable, the problem is called *simple order isotonic regression*.

A very important result in the theory of isotonic regression, is that the increasing function f closest to a given function g on X in the (weighted) least squares sense, can be constructed graphically. A geometrical interpretation of isotonic regression over a simple order finite set $X=\{x_1,\ldots,x_n\}$ is the following. Let $W_j=\sum_{i=1}^{j} w(x_i)$ and $G_j=\sum_{i=1}^{j} g(x_i)\,w(x_i)$; the points $P_j=(W_j,G_j)$ obtained plotting the cumulative sums G_j against the cumulative sums W_j, $j=0,\ldots,n$ ($W_0=0$, $G_0=0$), constitute the *cumulative sum diagram* (CSD) of a given function g with weights w. The isotonic regression of g is given by the *slope* of the *greatest convex minorant* (GCM) (i. e., the graph of the supremum of all convex functions whose graphs lie below the CSD) of the CSD; the slope of the segment joining P_{j-1} to P_j is just $g(x_j)$, $j=1,\ldots,n$, while the slope of the segment joining P_{i-1} to P_j, $i\le j$ is the weighted average

$$Av\{x_i,\ldots,x_j\}=\frac{\sum_{r=i}^{j} g(x_r)w(x_r)}{\sum_{r=i}^{j} w(x_r)}.$$

Therefore, the value of the isotonic regression g^* at the point x_j is just the slope of the GCM at the point $P_j^*=(W_j,G_j^*)$, where $G_j^*=\sum_{i=1}^{j} g^*(x_i)\,w(x_i)$. Note that, if P_j^* is a 'corner' of the graph, g^* is the slope of the segment extending to the left. An illustrative example is shown in Fig. 1.

j	$w(x_j)$	W_j	$g(x_j)$	G_j	$g^*(x_j)$	G_j^*
1	1	1	−1	−1	−1	−1
2	1	2	3	2	1/5	−4/5
3	2	4	1	4	1/5	−2/5
4	2	6	−2	0	1/5	0
5	1	7	2	2	2	2



Isotonic Regression Problems, Figure 1
Example of CSD and GCM

Other isotonic regression problems are based on a less restrictive kind of order: *partial order* and *quasi-order*. In the partial order isotonic regression problem the binary relation $\preccurlyeq$ on X is reflexive, transitive and antisymmetric, but there may be noncomparable elements. In the quasi-order isotonic regression problem, the ordering relation satisfies only the first two conditions.

The isotonic regression problem arises in both statistics and operations research. Applications in statistics are discussed in [2,10] and [14]. Applications in operations research can be found in [11] and [15].

The problem under consideration is also of theoretical interest being one of the very few quadratic problems known for which *strongly polynomial solution* algorithms exist (an algorithm is said to be *strongly polynomial* if the number of elementary arithmetic operations it requires is a polynomial in the problem parameter and not just the size of the input data). That is why many researcher in the area of order restricted statistical inference have paid a great deal of attention to the problem of developing algorithms for isotonic regression. Most of the algorithms proposed involve averaging g over suitably selected subsets S of X on each of which $g^*(x)$ is constant.

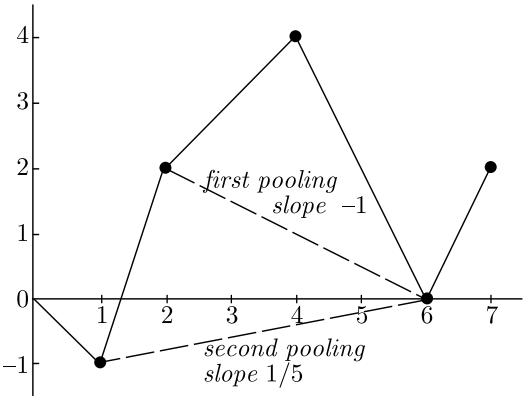
The Pool Adjacent Violators Algorithm

The first and the most widely used algorithm for the simply ordered isotonic regression is the pool adjacent violators algorithm (PAV) proposed by M. Ayer et al. [1] in 1955.

This algorithm follows directly from the geometrical interpretation of isotonic regression. As it is said before, the solution of the problem under consideration is given by the left derivative of the greatest convex function lying below the CSD. If, for some index i, $g(x_{i-1}) > g(x_i)$, then the graph of the part of the GCM between the points P_{i-2}^* and P_i^* is a straight line segment. Thus the CSD could be altered by connecting P_{i-2} with P_i by a straight line segment, without changing the GCM. The PAV algorithm is based on this idea of successive approximation to the GCM. (See Fig. 2 for a geometrical interpretation of 'pooling' adjacent violators.)

In describing the algorithm, an arbitrary set of consecutive elements of X will be referred to as a *block*. The

j	1	2	3	4	5
$w(x_j)$	1	1	2	2	1
$g(x_j)$	−1	3	1	−2	2
First pool					
$w(x_j)$	1	2	4	4	1
$g(x_j)$	−1	3	−1	−1	2
Second pool					
$w(x_j)$	1	5	5	5	1
$g(x_j) \equiv g^*(x_j)$	−1	1/5	1/5	1/5	2



Isotonic Regression Problems, Figure 2
Geometrical interpretation of pooling adjacent violators

aim is to find the *solution blocks*, that is a partitioning of X into sets on each of which the isotonic regression function g^* is constant.

The PAV algorithm starts from the initial block class Δ consisting of the singleton sets $\{x_i\}$, $1 \le i \le n$. At each stage of the algorithm, a new block class is obtained from the previous block class by joining the blocks together until the final partition is reached. If $g(x_1) \le \cdots \le g(x_n)$, then the initial partition is also the final partition, and $g^*(x_i) = g(x_i)$, $i = 1, \ldots, n$.

Otherwise, select any of the pairs of violators of the ordering; that is, select an index i such that $g(x_i) > g(x_{i+1})$.'Pool' these two values of g: i. e., join the two points x_i and x_{i+1} in a block $\{x_i, x_{i+1}\}$ ordered between $\{x_{i-1}\}$ and $\{x_{i+2}\}$, and associate to this block the average value $Av\ \{x_i, x_{i+1}\}$ and the weight $w(x_i) + w(x_{i+1})$. After each step in the algorithm, the average values associated with the blocks are examined to see whether or not they are in the required order. If so, the final partition has been reached, and the value of g^* at each point of a block is the 'pooled' value associated with the block. If not, a pair of adjacent violating blocks is selected, and pooled to form a single block, with associated weight the sum of their weights and associated average value the weighted average of their average values, completing another step of the algorithm.

A pseudocode for PAV algorithm is presented below, where B is the first block in Δ and B_+ is the block that follows B in the blocks partition.

$\Delta = \{\{x_1\}, \ldots, \{x_n\}\}$
REPEAT
  set B and B_+
  WHILE $B_+ \neq 0$
    IF $AvB \ge AvB_+$ THEN
      $\Delta = \Delta \setminus \{B, B_+\} \cup \{B \cup B_+\}$
      $B = B \cup B_+$
      $g^*(x) = AvB, \forall x \in B$
    ELSE
      $B = B_+$
    ENDIF
    $B_+ = \mathrm{succ}(B)$
  ENDWHILE
UNTIL there are no violating blocks

A pseudocode for PAV algorithm

S.J. Grotzinger and C. Witzgall [9] have shown that the computational complexity of the PAV algorithm is $O(n)$.

Minimum Lower Set Algorithm

The first algorithm proposed for partially order isotonic regression is the *minimum lower sets* (MLS) due to H.D. Brunk [4].

For describing this algorithm, as for most of the algorithms for general partial order, it is convenient to introduce the concept of 'level set' that generalizes the concept of 'block'. In order to define this set, important concepts are lower and upper set. A set $L \subseteq X$ is called *lower set* if $\forall x \in X$ and $\forall y \in L$ with $x \preccurlyeq y \Rightarrow x \in L$. A set $U \subseteq X$ is called *upper set* if $\forall x \in X$ and $\forall y \in U$ with $x \preccurlyeq y \Rightarrow x \in U$. Finally, $S \subseteq X$ is called *level set* if there are a lower set $L \subseteq X$ and an upper set $U \subseteq X$ such that $S = L \cap U$.

The isotonic regression with respect to any partial order is constant on level sets. The MLS algorithm computes the isotonic regression function by partitioning X into l level sets $S_1, \ldots, S_l$ such that $AvS_1 < \cdots < AvS_l$. In doing that, the algorithm performs l steps in each of which searches for the largest level set of minimum average S_i among the level sets $L_{i+1} \cap L_i^C$, where L_i^C is the complement of L_i. The isotonic regression values are given by the weighted average of the observations in each of the level set that belong to the solution partition.

In the following a pseudocode for MLS algorithm is given, where $\mathbf{L}$ is the lower set family of X.

M.J. Best and N. Chakravarti [3] have proved that the MLS algorithm is of computational complexity $O(n^2)$.

select $L_1 \subseteq X : AvB_1 = AvL_1$
  $= \min\{AvL : L \in \mathbf{L}\}$
$i = 1$
REPEAT
  $i = i + 1$
  select $L_2 \subseteq X : AvB_i = Av(L_2 \cap L_1^C)$
    $= \min\{Av(L \cap L_1^C) : L \in \mathbf{L}\}$
  $L_1 = L_2$
UNTIL X is exhausted
$g^*(x) = AvB_j,\ \forall x \in B_j,\ j = 1, \ldots, i$

A pseudocode for MLS algorithm

Other Algorithms

Several other algorithms are available for solving the isotonic regression problem as well as its various special cases. Their description are provided in [2,3,6,7,8,10,11,12,13,15], among others.

Best and Chakravarti, in their paper [3], have pointed out that several of the proposed algorithms are *active set quadratic programming methods* and that this methodology provides a unifying framework for studying algorithms for isotonic regression.

See also

► Regression by Special Functions

References

1. Ayer M, Brunk HD, Ewing GM, Reid WT, Silverman E (1955) An empirical distribution function for sampling with incomplete information. Ann Math Statist 26:641–647
2. Barlow RE, Barthlomew DJ, Bremner JM, Brunk HD (1972) Statistical inference under order restrictions. Wiley, New York
3. Best MJ, Chakravarti N (1990) Active set algorithms for isotonic regression: A unifying framework. Math Program 47:425–439
4. Brunk HD (1955) Maximum likelihood estimates of monotone parameters. Ann Math Statist 26:607–616
5. Brunk HD (1965) Conditional expectation given a σ-lattice and applications. Ann Math Statist 36:1339–1350
6. Dykstra RL (1981) An isotonic regression algorithm. J Statist Planning Inference 5:355–363
7. Eeden Cvan (1957) Maximal likelihood estimation of partially or completely ordered parameters. I. Indag Math 19:128–136
8. Gebhardt F (1970) An algorithm for monotone regression with one or more indipendent variables. Biometrika 57:263–271
9. Grotzinger SJ, Witzgall C (1984) Projection onto order simplexes. Appl Math Optim 12:247–270
10. Lee CIC (1983) The min-max algorithm and isotonic regression. Ann Statist 11:467–477
11. Maxwell WL, Muchstadt JA (1985) Establishing consistent and realistic reorder intervals in production-distributioni systems. Oper Res 33:1316–1341
12. Pardalos PM, Xue G (1998) Algorithms for a class of isotonic regression problems. Algorithmica 23:211–222
13. Pardalos PM, Xue G, Young L (1995) Efficient computation of the isotonic median regression. Applied Math Lett 8:67–70
14. Robertson T, Wright FT, Dykstra RL (1988) Ordered restricted statistical inference. Wiley, New York
15. Roundy R (1986) A 98% effective lot-sizing rule for a multi-product multistage production/inventory system. Math Oper Res 11:699–727

J

Jaynes' Maximum Entropy Principle

MaxEnt

H. K. KESAVAN
University Waterloo, Waterloo, Canada

MSC2000: 94A17

Article Outline

Keywords
Entropy and Uncertainty
Why Choose Maximum Uncertainty?
Shannon Entropy
Jaynes' Maximum Entropy Formalism
Applications of MaxEnt and Conclusions
See also
References

Keywords

Entropy; Uncertainty; Optimization; Jaynes; Shannon

C.E. Shannon's seminal discovery [7] (1948) of his entropy measure in connection with communication theory has found useful applications in several other probabilistic systems. E.T. Jaynes has further extended its scope by discovering the *maximum entropy principle* (MaxEnt) [1] (1957) which is inherent in the process of optimization of the entropy measure when some incomplete information is given about a system in the form of moment constraints. MaxEnt has, over the past four decades, given rise to an interdisciplinary methodology for the formulation and solution of a large class of probabilistic systems. Furthermore, MaxEnt's natural kinship with the Bayesian methods of analyses has further bolstered its importance as a viable tool for statistical inference.

Entropy and Uncertainty

The word *entropy* first originated in the discipline of thermodynamics, but Shannon entropy has a much broader meaning since it deals with the more pervasive concept of *information*. The word entropy itself has now crept into common usage to mean transformation of a quantity, or phenomenon, from order to disorder. This implies an irreversible rise in *uncertainty*. In fact, the word uncertainty would have been more unambiguous as to its intended meaning in the context of information theory, but for historic reasons, the usage of the word entropy has come to stay in the literature.

Uncertainty arises both in probabilistic phenomena such as in the tossing of a coin and, equally well, in deterministic phenomena where we know that the outcome is not a chance event, but we are merely *fuzzy* about the possibility of the specific outcome. What is germane to our study of MaxEnt is only *probabilistic uncertainty*. The concept of probability that is used in this context is what is generally known as the *subjective interpretation* as distinct from the *objective interpretation* based on frequency of outcome of an event. The subjective notion of probability considers a probability distribution as representing a *state of knowledge* and hence it is observer dependant.

The underlying basis for an initial probability assignment is given by the *Laplace's principle of insufficient reason*. According to this, if in an experiment with n possible outcomes, we have no information except that each probability $p_i \geq 0$ and $\sum_{i=1}^{n} p_i = 1$, then the most unbiased choice is the uniform distribution: $(1/n, \ldots, 1/n)$. Laplace's principle underscores the

choice of maximum uncertainty based on logical reasoning only.

Why Choose Maximum Uncertainty?

We shall now consider the example of a die in order to highlight the importance of maximum uncertainty as a preamble to our later discussion of MaxEnt. When the only information available is that the die has six faces, the uniform probability distribution (1/6, …, 1/6), satisfying the natural constraint

$$\sum_{i=1}^{n} p_i = 1, \quad p_1 \geq 0, \ldots, p_6 \geq 0, \tag{1}$$

represents the maximum uncertainty. If, in addition, we are also given the mean number of points on the die, that is, if we are given that

$$p_1 + 2p_2 + 3p_3 + 4p_4 + 5p_5 + 6p_6 = 4.5, \tag{2}$$

the choice of distributions is restricted to the incomplete information given by both (1) and (2), and, consequently, the maximum uncertainty encountered at the first stage is reduced. Since there are only two independent equations in six variables, there is an infinity of probability distributions satisfying the constraints. Out of all such distributions, one can anticipate the existence of a distribution giving rise to the maximum uncertainty $S_{\max}$. The importance of $S_{\max}$ can be deduced from a careful consideration of the process by which uncertainty is reduced (or never increased) by providing more and more information in terms of moment constraints. If we use any distribution from amongst the infinity of distributions satisfying the constraints that is different from the one corresponding to $S_{\max}$, it would imply that we would be using some information in addition to those given by (1) and (2). But scientific objectivity would behoove that we should use only the information that is given to us, and scrupulously avoid using any extraneous information. The principle of maximum uncertainty can, accordingly, be stated as:

> Out of all probability distributions consistent with a given set of constraints, the distribution with maximum uncertainty should be chosen.

At first glance, it may seem paradoxical that while the goal is reduction of uncertainty, we are actually trying to maximize it. However, what we are ensuring by the principle of maximum uncertainty is that we are *maximally uncertain about what we do not know*.

Shannon Entropy

The conclusion from the example of the die is that in making inferences based on incomplete information, the probability distribution that has the maximum uncertainty permitted by the available information should be used. It is therefore necessary to have a quantitative measure of uncertainty in a probability distribution. A unique function was defined by Shannon [7] to measure uncertainty. Let $\mathbf{p} = (p_1, \ldots, p_n)$ be a probability distribution satisfying the natural constraint

$$\sum_{i=1}^{n} p_i = 1. \tag{3}$$

Shannon's measure of entropy (uncertainty) for this distribution is given by

$$S = -\sum_{i=1}^{n} p_i \ln p_i \tag{4}$$

Shannon arrived at this unique measure by first stating the desirable properties that such a measure should have. Since not all this long list of properties are independent, he considered a smaller independent set of properties and deduced the uniqueness of (4). Similarly, A.I. Kinchin [5] assumed a different independent set and arrived at the same measure.

The Shannon entropy measure is the basis for Jaynes' maximum entropy principle. Of particular importance is the *property of concavity* of the measure which guarantees the existence of a maximum entropy distribution with all $p_i \geq 0$. Shannon's work in information theory did not involve optimization and as such he did not make use of the concavity property whereas here, it is central to the development.

Jaynes' Maximum Entropy Formalism

We will now present Jaynes' maximum entropy formalism based on discrete multivariate distributions of the random variable X and state some important results arising from it.

The ensemble,

$$(X, \mathbf{p}) = ((x_1, p_1), \ldots, (x_n, p_n)),$$

where n is finite, represents all the possible realizations of X and their probabilities of occurrence. **p** is estimated by maximizing the Shannon measure (4) subject to the natural constraint (3) and the moment constraints

$$\sum_{i=1}^{n} p_i g_{ri} = a_r, \quad r = 1, \ldots, m; \quad p_i \geq 0. \tag{5}$$

The Lagrangian is given by

$$L = -\sum_{i=1}^{n} p_i \ln p_i - (\lambda_0 - 1)\left(\sum_{i=1}^{n} p_i - 1\right) - \sum_{r=1}^{m} \lambda_r \left(\sum_{i=1}^{n} p_i g_{ri} - a_r\right) \tag{6}$$

where $\lambda_0, \ldots, \lambda_m$ are the $(m+1)$ Lagrange multipliers corresponding to the $(m+1)$ constraints of (3) and (5).

$$\frac{\partial L}{\partial p_i} = 0 \Rightarrow -\ln p_i - \lambda_0 - \sum_{r=1}^{m} \lambda_r g_{ri} = 0 \tag{7}$$

or,

$$p_i = \exp(-\lambda_0 - \lambda_1 g_{1i} - \cdots - \lambda_m g_{mi}), \tag{8}$$

for $i = 1, \ldots, n$. The m multipliers are determined by substituting for p_i from (8), in (3) and (5) so that

$$\sum_{i=1}^{n} \exp\left(-\lambda_0 - \sum_{j=1}^{m} \lambda_j g_{ji}\right) = 1 \tag{9}$$

and

$$\sum_{i=1}^{n} g_{ri} \exp\left(-\lambda_0 - \sum_{j=1}^{m} \lambda_j g_{ji}\right) = a_r, \tag{10}$$

for $r = 1, \ldots, m$, or

$$\exp(\lambda_0) = \sum_{i=1}^{n} \exp\left(-\sum_{j=1}^{m} \lambda_j g_{ji}\right) \tag{11}$$

and

$$a_r \exp(\lambda_0) = \sum_{i=1}^{n} g_{ri} \exp\left(-\sum_{j=1}^{m} \lambda_j g_{ji}\right), \tag{12}$$

for $r = 1, \ldots, m$ so that

$$a_r = \frac{\sum_{i=1}^{n} g_{ri} \exp\left(-\sum_{j=1}^{m} \lambda_j g_{ji}\right)}{\sum_{i=1}^{n} \exp\left(-\sum_{j=1}^{n} \lambda_j g_{ji}\right)}, \tag{13}$$

$r = 1, \ldots, m$. Equation (11) gives λ_0 as a function of $\lambda_1, \ldots, \lambda_m$. Equations (13) give $a_1, \ldots, a_m$ as functions of $\lambda_1, \ldots, \lambda_m$.

Based on the above formalism, we can derive the following results which are useful in applications.

- The Lagrange multiplier λ_0 is a convex function of $\lambda_1, \ldots, \lambda_m$.
- The value of the maximum entropy $S_{\max} = \lambda_0 + \lambda_1 a_1 + \cdots + \lambda_m a_m$.
- $S_{\max}$ is a concave function of $a_1, \ldots, a_m$.
- The Lagrange multipliers $\lambda_1, \ldots, \lambda_m$ are the partial derivatives of $S_{\max}$ with respect to $a_1, \ldots, a_m$, respectively.
- An alternative proof that MaxEnt gives globally maximum values of entropy, that is, $S_{\max} - S \geq 0$, can be given on the basis of Shannon's inequality

$$\sum_{i=1}^{n} q_i \ln \frac{q_i}{p_i} \geq 0 \tag{14}$$

where **q** is the probability distribution with entropy S.

- Jaynes' formalism also leads to *Jaynes' entropy concentration theorem* that asserts that the constrained maximum probability distribution is the one that best represents our state of knowledge about the behavior of the system and that MaxEnt is the preferred method of inferring that distribution. The conclusion is based on (14) and the chi-square test.
- Jaynes' formalism is applicable to continuous-variate probability distributions also.
- In our earlier discussion, we had stated that the statement of the *Laplace's principle of insufficient reason* was based purely on logic. We can now show that uniform distribution results from MaxEnt when only the natural constraint (3) is specified.

Applications of MaxEnt and Conclusions

As the very first application, Jaynes demonstrated the power of MaxEnt by deriving all the principal distributions of statistical mechanics without reference to the classical methods of derivation [1]. An important application that closely followed the application of MaxEnt to statistical mechanics, was the correspondence that M. Tribus [9,10] established with the laws of thermodynamics. This application, incidentally, clarified the connection between the Shannon entropy and ther-

modynamic entropy. He also demonstrated that most of the statistical distributions that are commonly encountered can be derived from MaxEnt by making use of appropriate moment constraints, which, later, came to be known as *characterizing moments*. Thus, he established the integral link between information theory and statistics. For example, the normal distribution is a maximum-entropy distribution resulting from maximizing the Shannon entropy with respect to the characterizing moments of mean and variance.

These remarkable successes set in motion the applications of MaxEnt in several other disciplines. To name only a few, MaxEnt has been applied to problems in urban and regional planning, transportation, business, economics, finance, statistical inference, operations research, queueing theory, nonlinear spectral analysis, pattern recognition and image processing, computerized tomography, risk analysis, population growth models, chemical reactions and many other areas. These are all problems that are inherently probabilistic in nature or, alternatively, where the MaxEnt model is made to fit by artificially interpreting probabilities as proportions. References to these problems can be found in [2,3,4].

For the past ten years, the direction of research into MaxEnt has gone in the direction of using the principle in conjunction with Bayes theorem. There has been a series of workshops conducted under this heading which appears in the series [6].

Also of great interest is the concept of minimum entropy which is found useful in recognizing patterns contained in an information structure. However, research in this direction has been hampered by the computational complexity in determining the quantity because it results from the global minimization of a concave function which is a *NP*-hard problem.

Closely associated with MaxEnt are the methods of optimization based on Kullback–Leibler measure [8] to measure distance between two probability distributions. However, the school dedicated to the use of MaxEnt steers clear of this approach since it does not involve the concept of entropy.

See also

- ► Entropy Optimization: Interior Point Methods
- ► Entropy Optimization: Parameter Estimation
- ► Entropy Optimization: Shannon Measure of Entropy and its Properties
- ► Maximum Entropy Principle: Image Reconstruction

References

1. Jaynes ET (1957) Information theory and statistical mechanics. Phys Rev 106:620–630
2. Kapur JN (1989) Maximum entropy models in science and engineering. Wiley Eastern, New Delhi
3. Kapur JN, Kesavan HK (1987) Generalized maximum entropy principle (with applications). Sandford Educational Press Univ. Waterloo, Waterloo, Canada
4. Kapur JN, Kesavan HK (1992) Entropy optimization principles with applications. Acad. Press, New York
5. Kinchin AI (1957) Mathematical foundations of information theory. Dover, Mineola, NY
6. Series: Maximum entropy and Bayesian methods. Kluwer, Dordrecht
7. Shannon CE (1948) A mathematical theory of communication. Bell System Techn J 27:379–423, 623–659
8. Shore JE, Johnson RW (1980) Properties of cross-entropy minimization. IEEE Trans Inform Theory IT-27:472–482
9. Tribus M (1961) Thermostatics and thermodynamics. v. Nostrand, Princeton, NJ
10. Tribus M (1966) Rational descriptions, decisions and designs. Pergamon, Oxford

Job-shop Scheduling Problem

PETER BRUCKER
Fachber. Math./Informatik, Universität Osnabrück, Osnabrück, Germany

MSC2000: 90B35

Article Outline

Keywords
Complexity Results
Branch and Bound Algorithms
Upper Bounds
Lower Bounds
Branching
Immediate Selection
Heuristic Procedures
Priority Rule Based Heuristics
Shifting Bottleneck Heuristic
Local Search
See also
References

Keywords

Job-shop; Scheduling; Complexity; Heuristics

The *job-shop problem* may be formulated as follows. Given are n jobs $j = 1, \ldots, n$ and m machines $M_1, \ldots, M_m$. Job j consists of a sequence

$$O_{1j}, \ldots, O_{n_j,j}$$

of n_j operations which must be processed in the given order, i. e. $O_{i+1,j}$ cannot start before O_{ij} is completed for $i = 1, \ldots, n_j-1$. Associated with each operation O_{ij} there is a processing time p_{ij} and a machine $\mu_{ij} \in \{M_1, \ldots, M_m\}$. O_{ij} must be processed for p_{ij} time units on machine μ_{ij}. Each job can be processed by at most one machine at a time and each machine can process only one operation at a time. If not stated differently preemptions of operations are not allowed. One has to find a feasible schedule which minimizes the makespan.

It is assumed that all processing times are nonnegative integers and that all jobs and machines are available at starting time zero. Furthermore, if not stated differently, *machine repetition* is allowed, i. e. $\mu_{i+1,j} = \mu_{ij}$ is possible.

For a precise formulation of the job-shop problem, let $\mathcal{O}$ be the set of all operations, let $p(k)$ be the processing time of operation $k \in \mathcal{O}$, and define $(k, l) \in C$ if and only if $k = O_{ij}$ and $l = O_{i+1,j}$ for some job j and some $i = 1, \ldots, n_j - 1$. Finally, let $M(k)$ be the machine on which operation k must be processed.

Then the job-shop problem may be formulated as disjunctive linear program (cf. ► Linear programming):

$$\min \max_{k \in \mathcal{O}} \{s(k) + p(k)\} \tag{1}$$

such that

$$s(k) + p(k) \leq s(l) \quad \text{for all } k, l \in \mathcal{O}; (k, l) \in C, \tag{2}$$

$$s(k) + p(k) \leq s(l) \quad \text{or} \quad s(l) + p(l) \leq s(k) \quad \text{for all} \quad k, l \in \mathcal{O}; \quad k \neq l; \quad M(k) = M(l), \tag{3}$$

$$s(k) \geq 0 \quad \text{for all } k \in \mathcal{O}. \tag{4}$$

$s(k)$ represents the starting time of operation k. Due to (2) all operations of the same job are processed in the right order. The constraints (3) make sure that a machine cannot process two operations at the same time.

The job-shop problem may be represented by a *mixed graph* $G = (\mathcal{O}, C, D)$ with vertex set $\mathcal{O}$, the set C of (directed) arcs, and a set $D = \{\{k, l\}: k, l \in \mathcal{O}; k \neq l; M(k) = M(l)\}$ of (undirected) edges. Furthermore, the processing time $p(k)$ is associated with each vertex $k \in \mathcal{O}$. The arcs are called *conjunctions* and the edges are called *disjunctions*.

The basic scheduling decision is to define a processing order of the operations on each machine, i. e. to fix precedence relations between these operations.

In the mixed graph model this is done by orienting edges, i. e. by turning disjunctions into conjunctions. A set S of these oriented edges is called an *orientation*. An orientation S is called a *complete orientation* if

- every edges becomes oriented; and
- the resulting directed graph $G(S) = (\mathcal{O}, C \cup S)$ has no cycles.

A complete orientation S defines a feasible schedule which is calculated in the following way. For each operation k let $l(k)$ be the length of a longest path to vertex k in $G(S)$. A path to k is a sequence of vertices $v_1, \ldots, v_r = k$ with (v_i, v_{i+1}) is an arc for $i = 1, \ldots, r-1$. The length of a path P to k is the sum of all processing times of operations in P excluding operation k. Choose $l(k)$ as the starting time of operation k. It is not difficult to see that this schedule is feasible. Furthermore, the length of the longest path in $G(S)$ defines the makespan of this schedule. A corresponding path is called *critical path*.

On the other hand a feasible schedule $s = (s(k))_{k \in \mathcal{O}}$ defines a complete orientation S and the critical path length in $G(S)$ is not greater than the makespan of the schedule s. Thus, one may restrict attention to schedules defined by complete orientations.

There are only a few special cases of the job-shop problem which are polynomially solvable (cf. ► Complexity classes in optimization). They will be discussed next.

Complexity Results

The two-machine job-shop problem in which each job has at most two operations can be solved by a simple extension of Johnson's algorithm for the two machine flow-shop problem [16]. Let I_i be the set of jobs with operations on M_i only ($i = 1, 2$), and let $I_{1,2}$ ($I_{2,1}$) be the

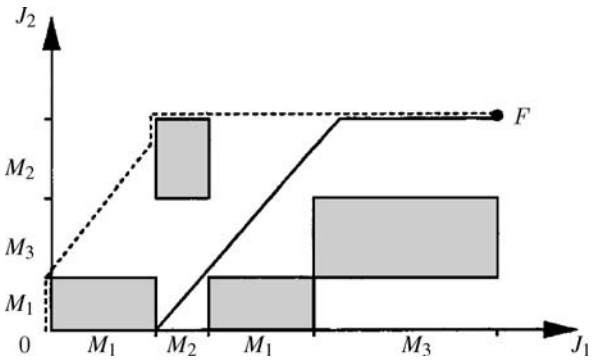
set of jobs which are processed first on M_1 (M_2) and then on M_2 (M_1). Order the latter two sets by means of Johnson's algorithm and the former two sets arbitrarily. Then one obtains an optimal schedule by executing the jobs on M_1 in order (I_{12}, I_1, I_{21}) and on M_2 in order (I_{21}, I_2, I_{12}).

In [15] the two-machine job-shop problem with unit-time operations ($p_{ij} = 1$) and no machine repetition is solved in time linear in the total number of operations, through a rule that gives priority to the longest remaining job. Despite the fact that this algorithm is fast it is not polynomial if we represent each job j by the machine which processes the first operation O_{1j} and the number n_j of operation of j. However, there exists a clever implementation of this algorithm which is polynomial ([17,26]). Surprisingly, the problem is *NP*-hard if we allow repetition of machines [12].

This, however, is probably as far as one can get if the number of jobs is not fixed but part of the input. Two-machine job-shop problems with $n_j \leq 3$ or $p_{ij} \in \{1, 2\}$ are *NP*-hard as well as three machine problems with $n_j \leq 2$ or $p_{ij} = 1$ ([18,19]).

The job-shop problem with two jobs may be formulated as a shortest path problem in the plane with regular objects as obstacles [2]. Figure 1 shows a shortest path problem with obstacles which corresponds to a job-shop problem with two jobs with $n_1 = 4$ and $n_2 = 3$. The processing times of the operations of job 1 (job 2) are represented by intervals on the x-axis (y-axis) which are arranged in the order in which the corresponding operations are to be processed. Furthermore, the intervals are labeled by the machines on which the corresponding operations must be processed.

A feasible schedule corresponds to a path from 0 to F consisting of segments which are either parallel to one of the axes or diagonal, and avoids the interior of any rectangular obstacle. If one defines the length of the diagonal parts of the path to be equal to the projections of these parts on one of the axes then the length of the path corresponds to the length of the schedule. It can be shown that this geometric problem can be formulated as a shortest path problem in an acyclic network with $O(r^2)$ arcs where $r = \max\{n_1, n_2\}$ and that this network can be calculated in time $O(r^2 \log r)$ which is also the complexity for solving the problem [7]. The corresponding preemptive problem can be solved in $O(r^3)$ time by allowing



Job-shop Scheduling Problem, Figure 1
Path problem with obstacles

the paths to go horizontally or vertically through the obstacles [24].

The two-machine job-shop problem with a fixed number k of jobs has been solved with time complexity $O(n^{2k})$ [9]. However, the three machine job-shop problem with $k = 3$ is *NP*-hard [25] (cf. also ▶ Complexity theory). If one allows preemption even the two-machine problem with $k = 3$ is *NP*-hard [12]. This is very surprising because the corresponding problem without preemption is polynomially solvable.

Branch and Bound Algorithms

Effective branch and bound algorithms (cf. ▶ Integer programming: Branch and bound methods) have been developed for the job-shop scheduling problem from the 1990s onwards ([3,11,13,20]). Rather than a description of each of these algorithms in detail some of the main concepts, like lower and upper bounds, branching rules, and immediate selection are presented.

Most of the branch and bound algorithms use the mixed graph model. The nodes of the enumeration tree correspond to orientations of edges representing sets of feasible schedules. Branching is done by adding further orientations in different ways. The leaves of the enumeration tree correspond to complete orientations while the root is defined by the empty orientation. Given an orientation S one may define heads and tails. A *head* $r(k)$ of operation k is a lower bound for an earliest possible starting time of k. A *tail* $q(k)$ of operation k is a lower bound for the time period between the finishing time of operation k and the optimal makespan. A simple way to derive a head $r(k)$ would be to calcu-

late the length of a longest path to k in $G(S)$. Similarly, a tail $q(k)$ could be calculated as the length of a longest path starting in k excluding $p(k)$.

Let P be a critical path in $G(S)$ and $L(S)$ be the length of P. A maximal sequence $u_1, \ldots, u_l$ of successive operations in P to be processed on the same machine is called a *block* if it contains at least two operations.

The following block theorem is used in connection with branch and bound algorithms. It also plays an important role when defining neighborhoods for local search methods.

Theorem 1 (*block theorem*) *Let S be a complete orientation. If there exists another complete orientation S' such that $L(S') < L(S)$, then in S' at least one operation of some block B of $G(S)$ has to be processed before the first or after the last operation of B.*

Upper Bounds

Each feasible solution provides an *upper bound*. At the root of the enumeration tree some time is invested for calculating a good upper bound to start with. This is accomplished by applying tabu search using an appropriate neighborhood. Some branch and bound algorithms also calculate heuristically a feasible solution satisfying the given orientation in each vertex of the enumeration tree. If this solution provides a better upper bound than the current one then the current bound is updated. Furthermore, informations provided by the heuristic solution are used for the branching process.

Lower Bounds

Lower bounds are calculated for each node of the enumeration tree, i. e. for the set of solutions feasible with respect to the current orientation S. Lower bounds may be calculated constructively or destructively. *Constructive lower bounds* are calculated by solving relaxations of the problem. The *destructive methods* work as follows. For a given integer U one tries to prove that there exists no feasible solution with value $C_{\max} \leq U$. In this case $U + 1$ is a valid lower bound. In case of a failure one repeats the test for a smaller U-value. Binary search can be applied to find a large lower bound.

The length of a critical path in $G(S)$ provides a constructive lower bound which can be calculated fast. Good bounds are provided by certain one-machine relaxations denoted as *head-body-tail problem*: Given a set of jobs $j = 1, \ldots, n$ with release times (heads) $r(j)$, processing times $p(j)$, and tails $q(j)$ to be processed on a single machine. Find a schedule with starting times $s(j)$ satisfying the release times such that $\max_{j=1}^{n} \{s(j) + p(j) + q(j)\}$ is minimized.

Unfortunately the one-machine head-body-tail problem is *NP*-hard. However, the preemptive version of this problem can be solved in time $O(n \log n)$ by applying the following scheduling rules:

- Take the release times and completion times as decision points.
- Schedule jobs in increasing order of decision points preferring an available job with longest tail.

By applying this algorithm to all operations to be processed on M_k one gets a lower bound L_k ($k = 1, \ldots, m$). The best of all these L_k-values is chosen.

Other constructive lower bounds are based on two job relaxations [10] and cutting plane approaches (cf. also ► Integer programming: Cutting plane algorithms) [3].

For destructive methods one assumes that U is a fictive upper bound and wants to prove that no feasible schedule satisfying $C_{\max} \leq U$ exists. From U one derives the time window $[r(k), d(k)]$ with $d(k) = U - q(k)$ in which each operation k must be processed if U is valid. For each job $j = 1, \ldots, n$ let $\mathcal{S}_j$ the set of schedules for j which are feasible with respect to its time window. The feasibility problem can be reduced to a zero-one linear program as follows. For each schedule $\sigma \in \mathcal{S}_j$ ($j = 1, \ldots, n$) one defines $a(\sigma, i, t) = 1$ if σ requires machine M_i in time-period $[t-1, t]$ and $a(\sigma, i, t) = 0$ otherwise. Let $x_{j\sigma}$ a 0–1 decision variable that indicates whether job j is scheduled according to schedule σ. Then there exists no feasible schedule if and only if the following linear program has an optimal solution value $\lambda^* > 1$ (see [20]).

$$\min \lambda \tag{5}$$

such that

$$\sum_{\sigma \in S_j} x_{j\sigma} = 1, \quad j = 1, \ldots, n, \tag{6}$$

$$\sum_{j=1}^{n} \sum_{\sigma \in S_j} a(\sigma, i, t) x_{j\sigma} \leq \lambda, \quad i = 1, \ldots, m; \quad t = 1, \ldots, U, \tag{7}$$

$$x_{j\sigma} \in \{0, 1\}, \quad j = 1, \ldots, n; \quad \sigma \in S_j. \tag{8}$$

Due to (6) exactly one schedule is chosen from each set $\mathcal{S}_j$. The left-hand side of (7) counts the number of jobs to be processed on machine M_i in time-period $[t-1, t]$. Thus, one has a feasible schedule if and only if $\lambda^* = 1$. To check infeasibility one uses the continuous version of (5) to (8) where (8) is replaced by

$$x_{j\sigma} \geq 0, \quad j = 1, \ldots, n; \quad \sigma \in S_j.$$

A second destructive lower bound based on immediate selection will be discussed later.

Branching

The simplest branching scheme is to choose a not yet oriented edge and orient it in the two possible ways. Another more sophisticated branching scheme is based on the block theorem. There is a branch to several children of the same father node in the enumeration tree. The idea behind such a branching is to orient many edges when branching (see [11]). In [20] a time oriented branching schemes has been used.

Immediate Selection

By branching disjunctions are oriented. There is another method to orient disjunctions which is due to [13]. This method is called *immediate selection*. It uses an upper bound UB for the optimal makespan and simple lower bounds based on heads $r(k)$ and tails $q(k)$ of operations k.

Let I be a set of n operations to be processed on the same machine and consider a strict subset $J \subset I$ and an operation $c \in I \setminus J$. If condition

$$\min_{j \in J \cup \{c\}} r(j) + \sum_{j \in J \cup \{c\}} p(j) + \min_{j \in J} q(j) \geq UB \qquad (9)$$

holds, then all operations $j \in J$ must be processed before operation c if we want to improve the current upper bound UB. This follows from the fact that the left-hand side of (9) is a lower bound for all schedules in which c does not succeed all jobs in J. Due to integrality, (9) is equivalent to

$$\min_{j \in J \cup \{c\}} r(j) + \sum_{j \in J \cup \{c\}} p(j) + \min_{j \in J} q(j) > UB - 1$$

or

$$\min_{j \in J \cup \{c\}} r(j) + \sum_{j \in J \cup \{c\}} p(j) > \max_{j \in J} d(j), \qquad (10)$$

where $d(j) := UB - q(j) - 1$.

(J, c) is called a *primal pair* if (9) or, equivalently, (10) holds. The corresponding arcs $j \to c$ with $j \in J$ are called *primal arcs*. Similarly, (c, J) is called a *dual pair* and arcs $c \to j$ are called *dual arcs* if

$$\min_{j \in J} r(j) + \sum_{j \in J \cup \{c\}} p(j) > \max_{j \in J \cup \{c\}} d(j) \qquad (11)$$

holds. In this case all operations $j \in J$ must be processed after operation c if we want to improve the current solution value UB.

It can be shown [9] that all primal and dual arcs for the set I can be calculated in $O(n^2)$ time.

Immediate selection is applied to speed up a branch and bound algorithm. For each machine the set I of all operations to be processed on this machine is considered and all corresponding primal and dual arcs are calculated. Then heads and tails are recalculated and the whole process is repeated until there are no new primal or dual arcs to be added.

By this method the orientation S is increased step by step. A possible outcome of this process is that one deduces a graph $G(S)$ which contains cycles. This implies that there exists no feasible solution with makespan $\leq UB$.

Immediate selection and a corresponding cycle check is applied to each node of the enumeration tree. If the cycle check is positive, one can backtrack.

Immediate selection is also used to calculate a lower bound by the destructive method.

Heuristic Procedures

Using a branch and bound algorithm and immediate selection J. Carlier and E. Pinson [13] were able to solve the notorious 10×10 benchmark problem in [21] for the first time. Recently (2000), 15×15 benchmark problems have been solved [6]. Problems of dimension $n \times n$ for $n > 15$ are still out of the reach if one is interested in optimal solutions. Thus, the only way to find solutions for larger job-shop problems is to apply heuristics, which provide solutions which are not too far away from the optimum. Some of these heuristics will be discussed next.

Priority Rule Based Heuristics

These are probably most frequently applied due to their ease of implementation and low computation times.

The idea of a priority heuristic is to schedule operations step by step each as early as possible. In each step among all unscheduled operations with the property that their conjunctive predecessors are scheduled a candidate with the highest priority is chosen to be scheduled next. For extended summaries and discussions of priority rules see [5,14,23].

Shifting Bottleneck Heuristic

(See [1,4].) This is one of the most powerful heuristics for the job-shop scheduling problem. In each iteration a machine M_i is chosen and all disjunctions between operations to be processed on M_i are oriented. This is done according to the exact solution of the head-body-tail problem for M_i. Thus, after k steps the disjunctions for k machines are oriented. Let $\mathcal{M}_k$ the set of these k machines. Before choosing a new machine $M_i \notin \mathcal{M}_k$ in the next step the orientations for the machines in $\mathcal{M}_k$ are updated by applying the head-body-tail algorithm to each of the machines in $\mathcal{M}_k$ in a given machine order. As a machine $M_i \notin \mathcal{M}_k$ added to $\mathcal{M}_k$ in the next step a *bottleneck machine* is chosen. A bottleneck machine is a machine $M_i \notin \mathcal{M}_k$ with a largest head-body-tail problem solution value. It is important to note that before each application of the solution procedure for a head-body-tail problem heads and tails are updated according to the current orientation.

Local Search

An important class of heuristics are *local search* methods like iterative improvement, simulated annealing (cf. ► Simulated annealing), tabu search and genetic algorithms (cf. ► Genetic algorithms). All these methods have been applied to the job-shop problem (see [27] for an excellent survey).

The local search techniques are based on the concept of local improvement. Given an existing solution or representation of such a solution, a modification is made in order to obtain a different (usually better) solution.

For the job-shop problem solutions are represented by complete orientations. To modify a solution one usually restricts to critical paths (which must be destroyed for improving the current makespan). One possibility for modifications is to choose an arc (v, w) on a critical path such that v and w are processed on the same machine and replace (v, w) by the reverse arc (w, v). It can be shown that the corresponding new orientation is complete again, i. e. no cycles are created by such a reversal. Other modifications are based on the block theorem. One modifies an orientation by shifting an operation of some block at the beginning or the end of the block. Such modifications are not defined if the resulting selections are not complete, i. e. contain cycles.

One of the best local search methods in terms of solution quality and computation time is a tabu search procedure described in [22].

See also

- ► MINLP: Design and Scheduling of Batch Processes
- ► Stochastic Scheduling
- ► Vehicle Scheduling

References

1. Adams J, Balas E, Zawack D (1988) The shifting bottleneck procedure for job shop scheduling. Managem Sci 34:391–401
2. Akers SB, Friedman J (1955) A non-numerical approach to production scheduling problems. Oper Res 3:429–442
3. Applegate D, Cook W (1991) A computational study of the job-shop scheduling problem. ORSA J Comput 3:149–156
4. Balas E, Lenstra JK, Vazacopoulos A (1995) One machine scheduling with delayed precedence constraints. Managem Sci 41:94–1096
5. Blackstone JH, Phillips DT, Hogg GL (1982) A state of art survey of dispatching rules for manufacturing. Internat J Production Res 20:27–45
6. Brinkkötter W, Brucker P (2001) Solving open benchmark instances for the jobshop problem by parallel head-tail adjustments. J Scheduling 4:53–64
7. Brucker P (1988) An efficient algorithm for the job-shop problem with two jobs. Computing 40:353–359
8. Brucker P (1994) A polynomial algorithm for the two machine job-shop scheduling problem with fixed number of jobs. Oper Res Spektrum 16:5–7
9. Brucker P (1998) Scheduling algorithms, 2nd edn. Springer, Berlin
10. Brucker P, Jurisch B (1993) A new lower bound for the job-shop scheduling problem. Europ J Oper Res 64:156–167
11. Brucker P, Jurisch B, Sievers B (1994) A branch and bound algorithm for the job-shop problem. Discrete Appl Math 49:107–127

12. Brucker P, Kravchenko SA, Sotskov YN (1999) Preemptive job-shop scheduling problems with a fixed number of jobs. Math Meth Oper Res 49:41–76
13. Carlier J, Pinson E (1989) An algorithm for solving the job-shop problem. Managem Sci 35:164–176
14. Haupt R (1989) A survey of priority-rule based scheduling. OR Spektrum 11:3–16
15. Hefetz N, Adiri I (1982) An efficient algorithm for the two-machine unit-time jobshop schedule-length problem. Math Oper Res 7:354–360
16. Jackson JR (1956) An extension of Johnson's results on job lot scheduling. Naval Res Logist Quart 3:201–203
17. Kubiak W, Sethi S, Sriskandarajah C (1995) An efficient algorithm for a job shop problem. Ann Oper Res 57:203–216
18. Lenstra JK, Rinnooy Kan AHG (1979) Computational complexity of discrete optimization problems. Ann Discret Math 4:121–140
19. Lenstra JK, Rinnooy Kan AHG, Brucker P (1977) Complexity of machine scheduling problems. Ann Discret Math 1:343–362
20. Martin PD, Shmoys DB (1996) A new approach to computing optimal schedules for the job shop scheduling problem. Proc. 5th Internat. IPCO Conf.
21. Muth JF, Thompson GL (1963) Industrial scheduling. Prentice-Hall, Englewood Cliffs, NJ
22. Nowicki E, Smutnicki C (1996) A fast tabu search algorithm for the job shop problem. Managem Sci 42:797–813
23. Panwalker SS, Iskander W (1977) A survey of scheduling rules. Oper Res 25:45–61
24. Sotskov YN (1991) On the complexity of shop scheduling problems with two or three jobs. Europ J Oper Res 53:323–336
25. Sotskov YN, Shakhlevich NV (1995) NP-hardness of shop-scheduling problems with three jobs. Discrete Appl Math 59:237–266
26. Timkovsky VG (1985) On the complexity of scheduling an arbitrary system. Soviet J Comput Syst Sci 23(5):46–52
27. Vaessens RJM, Aarts EHL, Lenstra JK (1996) Job shop scheduling by local search. INFORMS J Comput 8:302–317

K

Kantorovich, Leonid Vitalyevich

PANOS M. PARDALOS
Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 01A99

Article Outline

Keywords
See also
References

Keywords

Kantorovich; Linear programming; Economics; Functional analysis

L.V. Kantorovich was born in St. Petersburg, Russia, on January 6, 1912 and died on April 7, 1986. Kantorovich shared the 1975 *Nobel Prize* for Economics with T. Koopmans for their work on the optimal allocation of scarce resources [4,5].

Kantorovich was educated at Leningrad State Univ., receiving his doctorate in mathematics (1930) there at the age of 18. He became a professor at Leningrad in 1934, a position he held until 1960. He headed the department of mathematics and economics in the Siberian branch of the U.S.S.R. Academy of Sciences from 1961 to 1971 and then served as head of the research laboratory at Moscow's Institute of National Economic Planning (1971–1976). Kantorovich was elected to the prestigious Academy of Sciences of the Soviet Union (1964) and was awarded the Lenin Prize in 1965. For detailed interesting information on the life and scientific views of Kantorovich, see his paper [2]

Kantorovich was one of the first to use *linear programming* as a tool in economics. His most famous work is [1]. The characteristic of Kantorovich's work is a combination of theoretical and applied research. His first works concerned delicate problems of set theory. Later he became one of the first Soviet specialists on *functional analysis*. In the 1930s he laid down the foundations of the theory of *semi-ordered spaces* which constitutes now a vast chapter of functional analysis bordering algebra and measure theory. At the same time he anticipated the ideas of the future *theory of generalized functions* which became current only in the 1950s. Kantorovich obtained beautiful results on approximation theory. The approach to Sobolev's embedding theorem suggested by Kantorovich (based on his estimations of integral operators) is well known.

See also

► History of Optimization
► Linear Programming

References

1. Kantorovich LV (1959) The best use of economic resources
2. Kantorovich LV (1987) My way in mathematics. Uspekhi Mat Nauk 42(2)
3. Leifman LJ (ed) (1990) Functional analysis, optimization and mathematical economics (dedicated to the memory of L.V. Kantorovich). Oxford Univ. Press, Oxford
4. Linbeck A (ed) (1992) Nobel Lectures Economic Sciences 1969–1980. World Sci., Singapore
5. Mäler KG (ed) (1992) Nobel Lectures Economic Sciences 1981–1990. World Sci., Singapore

Kolmogorov Complexity

VICTOR KOROTKICH
Central Queensland University, Mackay, Australia

MSC2000: 90C60

Article Outline

Keywords and Phrases
See also
References

Keywords and Phrases

Complexity; Computation; Information; Randomness; Algorithmic complexity; Algorithmic information; Algorithmic entropy; Solomonoff–Kolmogorov–Chaitin complexity; Descriptional complexity; Shortest program length; Algorithmic randomness

In the mid 1960s R. Solomonoff [16], A.N. Kolmogorov [11] and G. Chaitin [4] independently invented the field now generally known as Kolmogorov complexity. It is also known variously as *algorithmic complexity*, *algorithmic information*, *algorithmic entropy*, *Solomonoff-Kolmogorov-Chaitin complexity*, *descriptional complexity*, *shortest program length*, *algorithmic randomness* and others. An extensive history of the field can be found in [14].

The Kolmogorov complexity formalizes the notion of amount of information necessary to uniquely describe a digital object. A digital object means one that can be represented as a finite binary string, for example, a genome, an Ising microstate, or an appropriately coarse-grained representation of a point in some continuum state space. In particular, the Kolmogorov complexity of a string of bits is the length of the shortest computer program that prints that string and stops running. The Kolmogorov complexity of an object is a form of absolute information of the individual object. This is not possible to do by Shannon's information theory. Unlike Kolmogorov complexity, information theory is only concerned with the average information of a random source [14].

Solomonoff was addressing the problem: How do we assign a priori probabilities to hypotheses when we begin an experiment? He represented a scientist's observations as a series of binary digits and weighted together all the programs for a given result into a probability measure. The scientist seeks to explain these observations through a theory, which can be regarded as an algorithm capable of generating the series and extending it, that is, predicting future observations. For any given series of observations there are always several competing theories and the scientist must choose among them. The model demands that the smallest algorithm, the one consisting of the fewest bits, be selected. Stated another way, this rule is the familiar formulation of *Occam's razor*: Given differing theories of apparently equal merit, the simplest is to be preferred [6].

Thus in the Solomonoff model a theory that enables one to understand a series of observations is seen as a small computer program that reproduces the observations and makes predictions about possible future observations. The smaller the program, the more comprehensive the theory and the greater the degree of understanding. Observations that are random cannot be reproduced by a small program and therefore cannot be explained by a theory. In addition the future behavior of a random system cannot be predicted. For random data the most compact way for the scientist to communicate his or her observations is to publish them in their entirety [6].

Kolmogorov and Chaitin independently suggested that computers be applied to the problem of defining what is meant by a random finite binary string of 0s and 1s [5]. In the traditional foundations of the mathematical theory of probability, as expounded by Kolmogorov in his classic [10], there is no place for the concept of an individual random string of 0s and 1s. Yet it is not altogether meaningless to say that the string

0011101000011100110100001111110

is more random than the string

0000000000000000000000000000000

for we may describe the second string as thirty 0s, but there is no shorter way to specify the first string than by just writing it all out [5].

We believe that the random strings of a given length are those that require the longest programs. Those

strings of length n that can be obtained by putting into a computer program much shorter than n are the non-random strings. The more possible it is to compress a binary string into a short program calculation, the less random is the string.

Solomonoff, Kolmogorov and Chaitin saw that the notion of 'computable' lay at the heart of their questions. They arrived at equivalent notions, showing that these two questions are fundamentally related.

The Kolmogorov complexity of a string is low if it can easily be obtained by a computation, whereas it will be high if it is difficult to obtain it [1]. The difficulty is measured by the length of the shortest program that computes the string on a universal Turing machine. The use of Turing machines to determine the length of the shortest program that computes a particular bit-string is intuitive: since a universal Turing machine can simulate any other Turing machine, the length of the program computing string s on Turing machine T, can only differ from the program computing the same string on Turing machine T' by a finite length $l(T, T')$, the length of the prefix code necessary to simulate T on T'. As this difference is constant (for each string s), the length of the shortest program to compute string s on a universal Turing machine is constant in the limit of infinitely long strings s and the *Kolmogorov complexity* of string s is defined as

$$K(s) = \min\{|p| : s = A_T(p)\},$$

where $|p|$ stands for the length of program p and $A_T(p)$ represents the result of running program p on Turing machine T.

This measure can be illustrated by a few examples. A blank tape (the string with all zeros) is clearly a highly regular string and correspondingly its Kolmogorov complexity will be low. Indeed, the program needed to produce this string can be very short: print zero, advance, repeat. The same is true, of course, for every string with a repetitive pattern.

Another way of viewing algorithmic regularity is by saying that an algorithmically regular string can be compressed to a much smaller size: the size of the smallest program that computes it. More interesting is the regularity of a string that can be obtained by the application of a finite but nontrivial algorithm, such as the calculation of the transcendental number π. The string representing the binary equivalent of π certainly appears completely random, yet the minimal program necessary to compute it is finite. Thus, such a string is also classified as algorithmically regular (though not quite as regular as the blank tape) [1].

Kolmogorov complexity also provides a means to define randomness in this context. According to the Kolmogorov measure, a string r is declared random if the size of the smallest program to compute r is as long as r itself, i. e.,

$$K(r) \approx |r| .$$

Why should this definition of randomness be preferable to any other we might come up with? The answer to that was provided by P. Martin-Loef, who was a postdoc of Kolmogorov. Roughly, he demonstrated that the definition 'an n-bit string s is random if and only if $K(s) \geq$ n' ensures that every such individual random string possesses with certainty all effectively testable properties of randomness that hold for strings produced by random sources on the average [9].

The algorithmic definition of randomness provides a new foundation for the theory of probability. By no means does it supersede classical probability theory, which is based on an ensemble of possibilities, each of which is assigned a probability. Rather, the algorithmic approach complements the ensemble method by giving precise meaning to concepts that had been intuitively appealing but that could not be formally adopted [6].

The Kolmogorov complexity of a string s is also defined as the negative base-2 logarithm of the string's algorithmic probability $\mathsf{P}(s)$ [2,18]. This in turn is defined as the probability that a standard universal computer T, randomly programmed, would embark on a computation yielding s as its sole output, afterward halting. The algorithmic probability $\mathsf{P}(s)$ may be thought of a weighted sum of contributions from all programs that produce s, each weighted according to the negative exponential of its binary length.

Turning to the sum of $\mathsf{P}(s)$ over outputs, the sum $\sum_s \mathsf{P}(s)$ is not equal to unity, because, as is well known, an undecidable subset of all universal computations fail to halt and so produce no output. Therefore $\sum_s \mathsf{P}(s)$ is an uncomputable irrational number less than 1. This number, called *Chaitin's Omega* [7], has many remarkable properties [8], such as the fact that its uncomputable digit string is a maximally compressed form of the information required to solve the halting problem [2].

Though very differently defined, Kolmogorov complexity is typically very close to ordinary statistical entropy $-\sum p \log p$ in value. To take a simple example, it is known that almost all N-bit strings drawn from a uniform distribution (of statistical entropy N bits) have Kolmogorov complexity nearly N bits. More generally, in any concisely describable ensemble of digital objects, i. e., a canonical ensemble of Ising microstates at a given temperature, the ensemble average of the objects' Kolmogorov complexity closely approximates the whole ensemble's statistical entropy [2,18]. In the case of continuous ensembles, the relation between Kolmogorov complexity and statistical entropy is less direct because it depends on the choice of coarse-graining. Some of the conceptual issues involved are discussed in [17].

The basic flaw in the Kolmogorov construction (as far as physical complexity is concerned) is the absence of a context [1]. This is easily rectified by providing the Turing machine with a tape u, which represents the physical 'universe', while the Turing machine with u as input computes various strings from u.

The *conditional Kolmogorov complexity* of a string s is defined as the length of the shortest program that computes s given string u [12]

$$K(s|u) = \min\{|p| : \ s = A_T(p, u)\},$$

where the notation $A_T(p, u)$ is introduced as the result of the computation running p on Turing machine T with u as input tape. The conditional complexity measures the remaining randomness in string s, i. e., it counts those bits that are not correlated with bits in u. In other words, the program p is the maximally compressed string containing those bits that cannot be computed from u, as well as the instructions necessary to compute those bits of s that can be obtained from u.

The latter part of the program is of vanishing length in the limit of infinitely long strings, which implies that the program p mainly contains the remaining randomness of s. The mutual complexity is defined by

$$K(s : u) = K(s) - K(s|u),$$

which clearly just measures the number of bits that mean something in the universe u.

Let us consider $K(s : u)$ in more detail. Its meaning becomes clearer if instead of considering a string s obtained by Turing machine T with universe u, the ensemble of strings S that can be obtained from a universe u with T is considered. This ensemble can be thought of as a probabilistic mixture subject to random bit-flips.

In other words, the output tapes to be connected to a heat bath can be imagined. In that case, an entropy $H(S)$ can be associated with the ensemble of strings S. Consider a Turing machine operating on u, a specific universe. Obtaining s from u then constitutes a measurement on the universe U and consequently not only reduces the conditional entropy of S given u, but also the conditional entropy of U given s.

Note that the universe is assumed here to be fully known, i. e., there is only one tape u in the ensemble U. While this must not strictly be so, sometimes it is convenient to assume that there is no randomness in the universe. Also, the length of the smallest program that computes s from u, averaged over the possible realizations of s, then just equals the conditional entropy of S given u [1].

It is known that the average Kolmogorov complexity over an ensemble of strings just equals the entropy of the ensemble. Then

$$H(S|u) = \langle K(s|u)\rangle_S = -\sum_s p(s|u)\log p(s|u) \quad (1)$$

and

$$\langle K(s) - K(s|u)\rangle_S = H(S) - H(S|u).$$

Note that (1) is not strictly a conditional entropy, as no average over different realizations of the universe takes place. Indeed, it looks just like a conventional Shannon entropy only with all probabilities being probabilities conditional on u.

Determining the Kolmogorov complexity of a string is a difficult problem. For this reason, Kolmogorov complexity remained more of a curiosity than a practical mathematical tool. In the last few years, mainly due to P. Vitanyi and M. Li, a significant progress in using Kolmogorov complexity has been made [14].

In particular, several successful applications of Kolmogorov complexity in the theory of computation are made and the general pattern of the incompressibility method emerged [14]. The incompressibility method and Kolmogorov complexity is truly a versatile mathematical tool. The incompressibility method is a basic general technique such as the 'pigeon hole argument', the 'counting method' or the 'probabilistic method'. It is

a sharper relative of classical information theory (absolute information of individual object rather than average information over a random ensemble) and yet satisfies many of the laws of classical information theory, although with a slight error term. Applications of Kolmogorov complexity have been given in a number of areas, including [14]:

- randomness of individual finite objects or infinite strings, Martin–Loef tests for randomness, Gödel's incompleteness result, information theory of individual objects;
- universal probability, general inductive reasoning, inductive inference, prediction, mistake bounds, computational learning theory, inference in statistics;
- the incompressibility method, combinatorics, graph theory, Kolmogorov 0–1 laws, probability theory;
- theory of parallel and distributed computation, time and space complexity of computations, average case analysis of algorithms, language recognition, string matching, routing in computer networks, circuit theory, formal language and automata theory, parallel computation, Turing machine complexity, complexity of tapes, stacks, queues, average complexity, lower bound proof techniques;
- structural complexity theory, oracles;
- logical depth, universal optimal search, physics and computation, dissipationless reversible computing, information distance and picture similarity, thermodynamics of computing, statistical thermodynamics and Boltzmann entropy.

Based on the Turing model of computation, the field of Kolmogorov complexity probably will need to be modified to account for the new quantum modes of computation. From recent studies there appear more facts that this modification is likely to be based on notions that go beyond the framework of space-time (for example [15]) and sought within the world view, which considers natural systems not as separate entities but as integrated parts of a undivided whole [3].

An attempt to contribute to such a modification is made in [13]. The results are based on a mathematical structure, called a web of relations, that is a collection of hierarchical formations of integer relationships. The web of relations allows to introduce a concept of structural complexity to measure the complexity of a binary string in terms of corresponding hierarchical formations of integer relationships. Importantly, the concept of structural complexity is based on the integers only and does not rely on notions that derive from space-time.

See also

- ▶ Complexity Classes in Optimization
- ▶ Complexity of Degeneracy
- ▶ Complexity of Gradients, Jacobians, and Hessians
- ▶ Complexity Theory
- ▶ Complexity Theory: Quadratic Programming
- ▶ Computational Complexity Theory
- ▶ Fractional Combinatorial Optimization
- ▶ Information-based Complexity and Information-based Optimization
- ▶ Mixed Integer Nonlinear Programming
- ▶ Parallel Computing: Complexity Classes

References

1. Adami C (1998) Introduction to artificial life. Springer, Berlin
2. Bennett C (1982) The thermodynamics of computation - a review. Internat J Theoret Physics 21:905–940
3. Bohm D (1980) Wholeness and the implicate order. Routledge and Kegan Paul, London
4. Chaitin G (1966) On the length of programs for computing binary sequences. J ACM 13:547–569
5. Chaitin G (1970) On the difficulty of computations. IEEE Trans Inform Theory 16:5–9
6. Chaitin G (1975) Randomness and mathematical proof. Scientif Amer 232:47–52
7. Chaitin G (1975) A theory of program size formally identical to information theory. J ACM 22:329–340
8. Gardner M (1979) Mathematical games. Scientif Amer 10:20–34
9. Kirchherr W, Li M, Vitanyi P (1997) The miraculous universal distribution. Math Intelligencer 19(4):7–15
10. Kolmogorov A (1950) Foundations of the theory of probability. Chelsea, New York
11. Kolmogorov A (1965) Three approaches to the definition of the concept "quantity of information". Probl Inform Transmission 1:1–7
12. Kolmogorov A (1983) Combinatorial foundations of information theory and the calculus of probabilities. Russian Math Surveys 38:29
13. Korotkich V (1999) A mathematical structure for emergent computation. Kluwer, Dordrecht
14. Li M, Vitanyi P (1997) An introduction to Kolmogorov complexity and its applications. second, revised and expanded edn. Springer, Berlin

15. Penrose R (1995) Shadows of the mind. Vintage, London
16. Solomonoff R (1964) A formal theory of inductive inference. Inform and Control 7:1–22
17. Zurek W (1989) Algorithmic randomness and physical entropy. Phys Rev A 40:4731–4751
18. Zvonkin A, Levin L (1970) The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. Russian Math Surv 256:83–124

Krein–Milman Theorem

GABRIELE E. DANNINGER-UCHIDA
University Vienna, Vienna, Austria

MSC2000: 90C05

Article Outline

Keywords
See also
References

Keywords

Convex; Convex hull; Extreme point

A theorem stating that a compact closed set can be represented as the convex hull of its extreme points. First shown by H. Minkowski [4] and studied by some others ([1,2,5]), it was named after the paper by M. Krein and D. Milman [3]. See also, for example, [6,7,8].

Theorem 1 *Let $C \subset \boldsymbol{R}^n$ be convex and compact, let $S = ext(C)$ be the set of extreme points of C.*

Then $conv(S) = C$, i. e. the convex hull of the extreme points of C coincides with the set C.

Proof Since $S \subset C$, $\text{conv}(S) \subset \text{conv}(C) = C$. So we are left to show that $C \subset \text{conv}(S)$. We prove this by induction.

Let $d = \dim C$. For $d = -1 (C = \emptyset)$, $d = 0$ and $d = 1$ the proof is trivial.

Let us assume that the theorem is true for all convex compact sets of dimension $d - 1 \geq 0$. If $x \in C$, but not in $\text{conv}(S)$, there exists a line segment in C such that x is in the interior of it (since x is not an extreme point). This line segment intersects the (relative) boundary of C in two points u and v. At least one of them is not extremal, else $x \in \text{conv}(S)$. Assume $u \notin S$. Since u is on the (relative) boundary of C there exists a supporting hyperplane H of C, that contains u. So $u \in C \cap H = \text{conv}(\text{ext}(C \cap H))$ (by induction, since $\dim(C \cap H) \leq d-1$). But $\text{ext}(C \cap H) = \text{ext}(C) \cap H$ and so $u \in \text{ext}(C) \cap H \subset \text{ext}(C)$.

An analogous result holds for v. Since $x \in [u, v]$, $x = \lambda u + (1 - \lambda)v$ with $\lambda \in]0, 1[$ and so $x \in \text{conv}(\text{ext}(C))$.

See also

- ▶ Carathéodory Theorem
- ▶ Linear Programming

References

1. Klee VL (1957) Extremal structure of convex sets. Arch Math 8:234–240
2. Klee VL (1958) Extremal structure of convex sets II. Math Z 69:90–104
3. Krein M, Milman D (1940) On extreme points of regular convex sets. Studia Math 9:133–138
4. Minkowski H (1911) Gesammelte Abhandlungen. Teubner, Leipzig-Berlin
5. Price GB (1937) On the extreme points of convex sets. Duke Math J 3:56–67
6. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton
7. Valentine FA (1964) Convex sets. McGraw-Hill, New York
8. Wets RJ-B (1976) Grundlagen Konvexer Optimierung. Lecture Notes Economics and Math Systems, vol 137. Springer, Berlin

Kuhn–Tucker Optimality Conditions

Kuhn–Tucker Conditions, KT Conditions

PANOS M. PARDALOS
Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Optimality conditions; KT point; Constraint qualification; Complexity

In this article we discuss necessary conditions for local optimality for an optimization problem in terms of a system of equations and inequalities which form the well-known Kuhn–Tucker KT conditions. Under suitable convexity assumptions about the objective function and the feasible domain, the Kuhn–Tucker conditions are sufficient even for global optimality. However, in the nonconvex case sufficiency is no longer guaranteed. The material of this article has been adapted from [2] and [4].

First, we consider the nonlinear optimization problem with inequality constraints,

$$\min_{x \in S} f(x), \tag{1}$$

where $S = \{x: g_i(x) \leq 0, i = 1, \ldots, p\} \subseteq \mathbf{R}^n$. We may assume that all the functions in the optimization problem are continuously differentiable on an open set containing S.

A vector $d \in \mathbf{R}^n$ is called a *feasible direction* at x^* if there exists $\lambda^* > 0$ such that $x^* + \lambda d \in S$ for every $0 < \lambda \leq \lambda^*$. Let $Z(x^*)$ denote the set of all feasible directions at x^*. A local minimum point $x^* \in S$ satisfies $d^\intercal \nabla f(x^*) \geq 0$ for every feasible direction $d \in Z(x^*)$.

Let

$$I(x^*) := \{i \in \{1, \ldots, p\}: \ g_i(x^*) = 0\}$$

be the index set of the *active constraints* at x^*. Recall that $d^\intercal \nabla g_i(x^*) < 0$ implies $g_i(x^* + \lambda d) < g_i(x^*)$ for $0 < \lambda \leq \overline{\lambda}_i$, for some $\overline{\lambda}_i > 0$, and $d^\intercal \nabla g_i(x^*) > 0$ implies $g_i(x^* + \lambda d) > g_i(x^*)$, $0 < \lambda \leq \widetilde{\lambda}_i$, for some $\widetilde{\lambda}_i > 0$. Moreover, each constraint which is not active in x^*, i. e., for which we have $g_i(x^*) < 0$, does not influence $Z(x^*)$, because $g_i(x) \leq 0$ holds in a neighborhood of x^*. It follows that $\{d: d^\intercal \nabla g_i(x^*) < 0, i \in I(x^*)\} \subseteq Z(x^*) \subseteq \{d: d^\intercal \nabla g_i(x^*) \leq 0, i \in I(x^*)\} =: L(x^*)$.

It is easy to see that for linearly constrained problems, where $g_i(x) = a_i^\intercal x - b_i$, $a_i \in \mathbf{R}^n \setminus \{0\}$, $b_i \in \mathbf{R}$, we have $Z(x^*) = L(x^*)$. On the other hand, one can readily construct examples of nonlinear constraints such that

$$Z(x^*) = \left\{d: \ d^\top \nabla g_i(x^*) < 0, \ i \in I(x^*)\right\}.$$

Now $\{d: d^\intercal \nabla g_i(x^*) < 0\}$ is an open set whereas $L(x^*)$ is closed. Recall that the closure cl M of a set $M \subset \mathbf{R}^n$ is the smallest closed set containing M. Because of the continuity of the inner product $d^\intercal \nabla g_i(x^*)$, we see that $d^\intercal \nabla f(x^*) \leq 0$ for every $d \in Z(x^*)$ implies that $d^\intercal \nabla f(x^*) \leq 0$ for every $d \in$ cl $Z(x^*)$. Hence, the condition $d^\intercal \nabla f(x^*) \leq 0$ for every $d \in$ cl $Z(x^*)$ is as well necessary for x^* to be a local minimum point. Clearly, by the discussion so far, we have cl $Z(x^*) \subseteq L(x^*)$. One would expect that also cl $\{d: d^\intercal \nabla g_i(x^* < 0, i \in I(x^*)\} = L(x^*)$, and hence cl $Z(x^*) = L(x^*)$. Indeed, this is true, apart from a few rather pathological cases. An example of such a pathological case is $S = \{x \in \mathbf{R}: g_1(x) := x^2 \leq 0\}$ and $x^* = 0$. Here we have $S = Z(x^*) =$ cl $Z(x^*) = \{0\}$, but $L(x^*) = \{\mathrm{d} \in \mathbf{R}: d \cdot 2 \cdot 0 \leq 0\} = \mathbf{R}$.

The constraints of the optimization problem $\min_{x \in S} f(x)$ are said to be *regular* in $x^* \in S$ when $L(x^*) =$ cl $Z(x^*)$. Every condition which ensures regularity in this sense is called a *constraint qualification*. Three of the most well-known constraint qualifications are given in the next theorem.

Theorem 1 *Each of the following conditions is a constraint qualification:*

- $g_i(x) = a_i^\intercal x - b_i$, $a_i \in \mathbf{R}^n \setminus \{0\}$, $b_i \in \mathbf{R}$ $(i = 1, \ldots, p;$ *linear constraints).*
- $g_i(x)$ *is convex* $(i = 1, \ldots, p)$, *and there exists* $\overline{x}$ *satisfying* $g_i(\overline{x}) < 0, \ldots, p;$ *Slater condition).*
- *The vectors* $\nabla g_i(x^*)$, $i \in I(x^*)$ *are linearly independent.*

The first two conditions ensure regularity in every $x^* \in S$ whereas the third requires knowledge of x^*.

Finally, one applies the well-known *Farkas lemma* (cf. ▶ Farkas lemma). This states that, whenever $d^\intercal \nabla f(x^*) \geq 0$ for every d satisfying $d^\intercal \nabla g_i(x^*) \leq 0$, $i \in I(x^*)$, there exists $\lambda_i \geq 0$, $i \in I(x^*)$ such that

$$\nabla f(x^*) + \sum_{i \in I(x^*)} \lambda_i \nabla g_i(x^*) = 0.$$

Since $I(x^*)$ is not known in advance, one formulates this equation in the following equivalent form:

Theorem 2 *Let* f, g_i *be continuously differentiable on an open set containing* S, *and let* x^* *be a local minimum point such that the constraints are regular in* x^*. *Then the following KT conditions hold:*

- $g_i(x^*) \le 0$, $i = 1, \ldots, p$;
- *there exist* $\lambda_i \ge 0$ *such that* $\lambda_i g_i(x^*) = 0$, $i = 1, \ldots, p$;
- $\nabla f(x^*) + \sum_{i=1}^{p} \lambda_i \nabla g_i(x^*) = 0$.

Theorem 3 *The KT conditions are sufficient for a constrained global minimum at* x^* *provided that the functions* $f(x)$ *and* $g_i(x)$, $i = 1, \ldots, p$, *are convex.*

Proof By convexity of $f(x)$ and $g_i(x)$ we have

$$f(x) \ge f(x^*) + (x - x^*)^\top \nabla f(x^*),$$
$$g(x) \ge g(x^*) + (x - x^*)^\top \nabla g(x^*).$$

Multiplying the last inequalities by λ_i and adding to the first one we obtain:

$$f(x) + \sum_{i=1}^{p} \lambda_i g_i(x) \ge f(x^*) + \sum_{i=1}^{p} \lambda_i g_i(x^*) + (x - x^*)^\top \left[\nabla f(x^*) + \sum_{i=1}^{p} \lambda_i \nabla g_i(x^*) \right].$$

Since the last two terms vanish because of the KT conditions, this implies $f(x) \ge f(x^*) - \sum_{i\in 1}^{p} \lambda_i g_i(x) \ge f(x^*)$ for all $x \in S$, that is x^* is a global minimum.

Note that no constraint qualification is required in the above Theorem.

Consider now problems with inequality and equality constraints

$$\min_{x \in S} f(x),$$

where

$$S = \left\{ x: \quad \begin{matrix} g_i(x) \le 0 \ (i = 1, \ldots, p), \\ h_i(x) = 0 \ (i = 1, \ldots, t) \end{matrix} \right\}.$$

Theorem 4 *Let* f, g_i $(i = 1, \ldots, p)$, h_i $(i = 1, \ldots, t)$ *be continuously differentiable in an open set containing* S, *and let* x^* *be a local minimum point. Further, assume that the vectors* $\nabla g_i(x^*)$ $(i \in I(x^*))$, $\nabla h_i(x^*)$ $(i = 1, \ldots, t)$ *are linearly independent. Then the following KT conditions hold:*

- $g_i(x^*) \le 0$ $(i = 1, \ldots, p)$, $h_i(x^*) = 0$ $(i = 1, \ldots, t)$.
- *There exist* $\lambda_i \ge 0$ $(i = 1, \ldots, p)$ *and* $\mu_i \in \boldsymbol{R}$ $(i = 1, \ldots, t)$ *such that*

$$\nabla f(x^*) + \sum_{i=1}^{p} \lambda_i \nabla g_i(x^*) + \sum_{i=1}^{t} \mu_i \nabla h_i(x^*) = 0,$$
$$\lambda_i g_i(x^*) = 0 \quad (i = 1, \ldots, p).$$

When the functions f, g_i $(i = 1, \ldots, p)$ *are convex, and the functions* $h_i(x)$ *are affine then the above two conditions are again sufficient for* x^* *to be a global minimum.*

Next, we consider the situation when Kuhn–Tucker theory is applied to nonconvex programming. We illustrate some difficulties arisen from nonconvexity by the following simple examples of concave minimization problems.

Example 5 Consider the problem

$$\begin{cases} \min & -(x_1^2 + x_2^2) \\ \text{s.t.} & x_1 \le 1. \end{cases}$$

The KT conditions for this problem are

$$x_1^* - 1 \le = 0, \quad \lambda_1(x_1^* - 1) = 0, \quad \lambda_1 \ge 0,$$
$$\lambda_1 - 2x_1^* = 0, \quad -2x_2^* = 0.$$

It is easy to see that the KT conditions are satisfied at $x^* = (0, 0)$ with $\lambda_1 = 0$ and at $x^* = (1, 0)$ with $\lambda_1 = 2$. The first is a global maximum. The second is neither a local minimum nor a local maximum. The problem has no local minima. Moreover, $\inf\{f(x): x \in S\} = -\infty$ since f is unbounded from below over S.

Example 6

$$\begin{cases} \min & 2x - x^2 \\ \text{s.t.} & 0 \le x \le 3. \end{cases}$$

The KT conditions for this problem are

$$\lambda_1(x^* - 3) = 0, \quad \lambda_2 x^* = 0,$$
$$2(1 - x^*) + \lambda_1 - \lambda_2 = 0,$$
$$\lambda_1 \ge 0, \lambda_2 \ge 0.$$

Since the objective function is strictly concave local minima occur at the endpoints of the interval $[0, 3]$. The point $x^* = 3$ is the global minimizer. The endpoints satisfy the KT conditions ($x^* = 0$ with $\lambda_1 = 0$, $\lambda_2 = 2$ and $x^* = 3$ with $\lambda_1 = 4$, $\lambda_2 = 0$). However, we can easily see that the KT conditions are also satisfied at $x^* = 1$ (with $\lambda_1 = \lambda_2 = 0$) and that this is a global maximum point.

These examples show that for minimization problems with nonconvex functions KT points may not be local minima.

Next, we consider the complexity of the problem of deciding existence of a Kuhn–Tucker point for

quadratic programming problems. When the feasible domain is unbounded, we prove that the decision problem is *NP*-hard.

Most classical optimization algorithms compute points that satisfy the Kuhn–Tucker KT conditions. When the feasible domain is not bounded it is not easy to check existence of a KT point. More precisely, consider the following quadratic problem

$$\begin{cases} \min & f(x) = c^\top x + \frac{1}{2} x^\top Q x \\ \text{s.t.} & x \geq 0 \end{cases} \tag{2}$$

where Q is an $n \times n$ symmetric matrix, and $c \in \mathbf{R}^n$. The Kuhn–Tucker optimality conditions for this problem become the following so-called *linear complementarity problem* (denoted by LCP(Q, c)): Find $x \in \mathbf{R}^n$ (or prove that no such an x exists) such that

$$Qx + c \geq 0, \quad x \geq 0, \qquad x^\top (Qx + c) = 0.$$

Hence, the complexity of finding (or proving existence) of Kuhn–Tucker points for the above quadratic problem is reduced to the complexity of solving the corresponding LCP.

Theorem 7 *The problem LCP(Q, c), where Q is symmetric, is NP-hard.*

Proof Consider the LCP(Q, c) problem in $\mathbf{R}^{n+3}$ defined by

$$Q_{(n+3)\times(n+3)} = \begin{pmatrix} -I_n & e_n & -e_n & 0_n \\ e_n^\top & -1 & -1 & -1 \\ -e_n^\top & -1 & -1 & -1 \\ 0_n^\top & -1 & -1 & -1 \end{pmatrix}$$

and $c_{n+3}^\top = (a_1, \ldots, a_n, -b, b, 0)$, where a_i, $i = 1, \ldots, n$, and b are positive integers, I_n is the $(n \times n)$-unit matrix and the vectors $e_n \in \mathbf{R}^n$, $0_n \in \mathbf{R}^n$ are defined by

$$e_n^\top = (1, \ldots, 1), \quad 0_n^\top = (0, \ldots, 0).$$

Define now the following *knapsack problem*: Find a feasible solution to the system

$$\sum_{i=1}^{n} a_i x_i = b, \quad x_i \in \{0, 1\} \quad (i = 1, \ldots, n).$$

This problem is known to be *NP*-complete [1]. Next we will show that LCP(Q, c) is solvable if and only if the associated knapsack problem is solvable.

Obviously, if x solves the knapsack problem, then $y = (a_1 x_1, \ldots, a_n x_n, 0, 0, 0)^\top$ solves LCP(Q, c).

Conversely, assume the point y solves LCP(Q, c) given above. Since $Qy + c \geq 0$, $y \geq 0$ we obtain $y_{n+1} = y_{n+2} = y_{n+3} = 0$. This in turn implies that $\sum_{i=1}^{n} y_i = b$ and $0 \leq y_i \leq a_i$. Finally, if $y_i < a_i$, then $y^\top (Qy + c) = 0$ enforces $y_i = 0$. Hence, $x = (y_1/a_1, \ldots, y_n/a_n)$ solves the knapsack problem.

Therefore, even in quadratic programming, the problem of 'deciding whether a Kuhn–Tucker point exists' is *NP*-hard.

See also

- Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- First Order Constraint Qualifications
- High-order Necessary Conditions for Optimality for Abnormal Points
- Implicit Lagrangian
- Inequality-constrained Nonlinear Optimization
- Lagrangian Duality: Basics
- Rosen's Method, Global Convergence, and Powell's Conjecture
- Saddle Point Theory and Optimality Conditions
- Second Order Constraint Qualifications
- Second Order Optimality Conditions for Nonlinear Optimization

References

1. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, New York
2. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
3. Kuhn HW, Tucker AW (1951) Nonlinear programming. In: Neyman J (ed) Proc. Second Berkeley Symp. Math. Statist. and Probab. Univ. Calif. Press, Berkeley, CA, pp 481–492
4. Pardalos PM, Rosen JB (1987) Constrained global optimization: Algorithms and applications. Lecture Notes Computer Sci, vol 268. Springer, Berlin

L

Lagrange, Joseph-Louis

TANIA QUERIDO, DUKWON KIM
University Florida, Gainesville, USA

MSC2000: 01A99

Article Outline

Keywords
See also
References

Keywords

Calculus of variations; Lagrange multipliers

J.L. Lagrange (1736–1813) made significant contributions to many branches of mathematics and physics, among them the theory of numbers, the theory of equations, ordinary and partial differential equations, the calculus of variations, analytic geometry and mechanics. By his outstanding discoveries he threw the first seeds of thought that later nourished C.F. Gauss and N.H. Abel.

During the first thirty years of his life he lived in Turin (France, now Italy) and, as a boy, his tastes were more classical than scientific. His interest in mathematics began while he was still in school when he read a paper by E. Halley on the uses of algebra in optics. He then began a course of independent study, and excelled so rapidly in the field of mathematical analysis that by the age of nineteen he was appointed professor at the Royal Artillery School and helped to found the Royal Academy of Science in 1757. His ideas had greatly impressed L. Euler, one of the giants of European mathematics. Euler and Lagrange, together, would join the first rank of the eighteenth century mathematicians, and their careers and research where often related [5].

In 1759 Lagrange focused his research in analysis and mechanics and wrote 'Sur la Propagation du son dans les fluides', a very difficult issue for that time [4]. From 1759 to 1761 he had his first publications in the 'Miscellanea of the Turin Academy'. His reputation was established.

Lagrange developed a new calculus which would enrich the sciences, called *calculus of variations*. In its simplest form the subject seeks to determine a functional relationship $y = f(x)$ such that an $\int_a^b g(x, y)\, dx$ could produce a maximum or a minimum. It was viewed as a mathematical study of economy or the 'best income' [4]. That was Lagrange's earliest contribution to the optimization area.

In 1766, Lagrange was appointed the Head of the Berlin Academy of Science, succeeding Euler. In offering this appointment, Frederick the Great wanted to turn his Academy into one of the best institutes of its day, proclaiming that the 'greatest mathematician in Europe' should live near the 'greatest king in Europe' [1]. During this period, he had a prosperous time, developing important works in the field of calculus, introducing the strictness in the calculus differential and integral. Later (1767) he published a memoir on the approximation of roots of polynomial equations by means of continued fractions; in 1770 he wrote a paper considering the *solvability of equations* in terms of permutations on their roots.

After Frederick's death, Lagrange left Berlin and became a member of the Paris Academy of Science by the invitation of Louis XVI (1787). He remained in Paris for the rest of his career, making a lengthy treatise on

the numerical solution of equations, representing a significant portion of his mathematical research. His papers on solution of third - and fourth-degree polynomial equations by radicals received considerable attention. His methods, laid in the early development of group theory to solving polynomials, were later taken by E. Galois. Lagrange's name was attached to one of the most important theorems of group theory [3]:

Theorem 1 *If o is the order of a subgroup g of a group G of order O, then o is a factor of O.*

In 1788 he published his masterpiece, the treatise 'Méchanique Analytique', which summarized and unified all the work done in the field of general mechanics since the time of I. Newton. This work, notable for its use of theory of differential equations, transformed mechanics into a branch of mathematical analysis. As W. Hamilton later said, 'he made a kind of scientific poem' [6].

In 1793, Lagrange headed a commission, which included P.S. Laplace and A. Lavoisier, to devise a new system of weights and measures. Out of this came the metric system.

Lagrange developed the method of variation of parameters in the solution of nonhomogeneous linear differential equations. In the determination of maxima and minima of a function, say $f(x, y, z, w)$, subject to constraints such as $g(x, y, z, w) = 0$ and $h(x, y, z, w) = 0$, he suggested the use of *Lagrange multipliers* to provide an elegant algorithm. By this method two undetermined constants λ and μ are introduced, forming the function $F \equiv f + \lambda g + \mu h$, from the related equations $F_x = 0$, $F_y = 0$, $F_z = 0$, $F_w = 0$, $g = 0$, and $h = 0$, the multipliers λ and μ are then eliminated, and the problem is solved. This procedure and its variations have emerged as a very important class of optimization method [1,2].

One can characterize Lagrange's contribution to optimization as his formalist foundation. Most of his results were retained and developed further by the following generations, who gave to his theory a different and practical course.

By the end of his life, Lagrange could not think futuristically for the mathematics. He felt that other sciences such as chemistry, physics and biology would attract the ablest minds of the future. His pessimism was unfounded. Much more was to be forthcoming with Gauss and his successors, making the nineteenth century the richest in the history of mathematics.

See also

- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ Lagrangian Multipliers Methods for Convex Programming
- ▶ Multi-objective Optimization: Lagrange Duality

References

1. Bertsekas DP (1986) Constrained optimization and Lagrange multiplier methods. Athena Sci., Belmont, MA
2. Boyer CB (1968) A history of mathematics. Wiley, New York
3. Fraser CG (1990) Lagrange's analytical mathematics, its Cartesian origins and reception in Comte's positive philosophy. Studies History Philos Sci 21(2):243–256
4. Julia G (1942-1950) La vie et l'oeuvre de J.-L. Lagrange. Enseign Math 39:9–21
5. Koetsier T (1986) Joseph Louis Lagrange (1736-1813): His life, his work and his personality. Nieuw Arch Wisk (4) 4(3):191–205
6. Simmons GF (1972) Differential equations, with applications and historical notes. McGraw-Hill, New York

Lagrange-Type Functions

A. M. Rubinov[1], X. Q. Yang[2]

[1] School of Information Technology and Mathematical Sciences, The University of Ballarat, Ballarat, Australia

[2] Department of Applied Mathematics, The Hong Kong Polytechnic University, Kowloon, Hong Kong

MSC2000: 90C30, 90C26, 90C46

Article Outline

Keywords and Phrases
References

Keywords and Phrases

Lagrange-type function; IPH functions; Multiplicative inf-convolution; Zero duality gap; Exact penalty function

Lagrange and penalty function methods provide a powerful approach, both as a theoretical tool and a computational vehicle, for the study of constrained optimization problems. However, for a nonconvex constrained optimization problem, the classical Lagrange primal-dual method may fail to find a minimum as a zero duality gap is not always guaranteed. A large penalty parameter is, in general, required for classical quadratic penalty functions in order that minima of penalty problems are a good approximation to those of the original constrained optimization problems. It is well-known that penalty functions with too large parameters cause an obstacle for numerical implementation. Thus the question arises how to generalize classical Lagrange and penalty functions, in order to obtain an appropriate scheme for reducing constrained optimization problems to unconstrained ones that will be suitable for sufficiently broad classes of optimization problems from both the theoretical and computational viewpoints.

One of the approaches for such a scheme is as follows: an unconstrained problem is constructed, where the objective function is a convolution of the objective and constraint functions of the original problem. While a linear convolution leads to a classical Lagrange function, different kinds of nonlinear convolutions lead to interesting generalizations. We shall call functions that appear as a convolution of the objective function and the constraint functions, *Lagrange-type functions*. It can be shown that these functions naturally arise as a result of a nonlinear separation of the image set of the problem and a cone in the image-space of the problem under consideration (see [4]). The class of Lagrange-type functions includes also augmented Lagrangians, corresponding to the so-called canonical dualizing parameterization. However, augmented Lagrangians constructed by means of some general dualizing parameterizations cannot be included in this scheme.

Consider the following problem $P(f,g)$:

$$\min f(x) \quad \text{subject to} \quad x \in X, g(x) \leq 0\,,$$

where X is a metric space, f is a real-valued function defined on X, and g maps X into $\mathbb{R}^m$, that is, $g(x) = (g_1(x), \ldots, g_m(x))$, where g_i are real-valued functions defined on X. We assume that the set of feasible solutions $X_0 = \{x \in X\colon g(x) \leq 0\}$ is nonempty and that the objective function f is bounded from below on X.

Let Ω be a set of parameters and $h\colon \mathbb{R}^{1+m} \times \Omega \to \mathbb{R}$ be a function. Let $\eta \in \mathbb{R}$. Then the function

$$L(x,\omega) = h(f(x) - \eta, g(x); \omega) + \eta, \quad x \in X, \omega \in \Omega\,, \tag{1}$$

is called a Lagrange-type function for problem $P(f,g)$ corresponding to h and η, and h is called a convolution function.

If h is linear with respect to the first variable, more specifically:

$$h(u, v; \omega) = u + \chi(v; \omega)\,,$$

where $\chi\colon \mathbb{R}^m \times \Omega \to \mathbb{R}$ is a real-valued function, then the parameter η can be omitted. Indeed, for each $\eta \in \mathbb{R}$, we have

$$L(x,\omega) = f(x) + \chi(g(x); \omega)\,.$$

However in general nonlinear situation the presence of η is important and different η lead to Lagrange-type functions with different properties.

One of the possible choices of the number η is $\eta = f(x_*)$ where x_* is a reference point, in particular x_* is a solution of $P(f,g)$ (see [4]). Then the Lagrange-type function has the form

$$L(x,\omega) = h(f(x) - f(x_*), g(x); \omega) + f(x_*)\,, \quad x \in X, \omega \in \Omega\,.$$

The Lagrange-type function (1) is a very general scheme and includes linear Lagrange functions, classical penalty functions, and augmented Lagrange functions as special cases.

Let $\Omega = \mathbb{R}^m_+$ and p be a real-valued function defined on $\mathbb{R}^{1+m}$. Define

$$h(u, v; \omega) = p(u, \omega_1 v_1, \ldots, \omega_m v_m)\,. \tag{2}$$

The Lagrange-type function has the form

$$L_p(x,\omega) = p(f(x) - \eta, \omega_1 g_1(x), \ldots, \omega_m g_m(x)) + \eta\,.$$

We can obtain fairly good results if the function p enjoys some properties. In particular, we assume that

(i) p is increasing;

(ii) $p(u, 0_m) \leq u$, for all $u \in \mathbb{R}$. (Here 0_m is the origin of $\mathbb{R}^m$.) One more assumption is useful for applications.

(iii) p is positively homogeneous ($p(\lambda x) = \lambda p(x)$ for $\lambda > 0$).

If both (i) and (iii) hold, then p is called an IPH function. Let p be a real-valued function defined on $\mathbb{R}^{1+m}$ and h be a convolution function defined by (2). Then

(a) If p enjoys properties (i) and (ii), then

$$\sup_{w\in\Omega} h(u,v;w) \le u, \quad \forall u \in \mathbb{R}, v \in \mathbb{R}^m_- .$$

(b) If p is an IPH function and $p(u, e_i) > 0$, where e_i is the i-th unit vector, $i = 1, \dots, m$, then

$$\sup_{w\in\Omega} h(u,v;w) = +\infty, \quad \forall v \notin \mathbb{R}^m_- .$$

We now give some examples of Lagrange-type functions. First two examples correspond to functions of the form (2).

1) Let $p(u,v) = u + \sum_{i=1}^m v_i$. Then $L_p(x,\omega) = f(x) + \sum_{i=1}^m \omega_i g_i(x)$ coincides with the classical Lagrange function.
2) Let $p(u,v) = u + \sum_{i=1}^m v_i^+$ where $v^+ = \max(v,0)$. Then $L_p(x,\omega) = f(x) + \sum_{i=1}^m \omega_i g_i(x)^+$ coincides with the classical (linear) penalty function. If $p(u,v) = u + \sum_{i=1}^m (v_i^+)^2$, then $L_p(x,\omega) = f(x) + \sum_{i=1}^m \omega_i (g_i(x)^+)^2$ is a quadratic penalty function.

We now give the definition of a penalty-type function. Let Ω be a set of parameters and $h: \mathbb{R}^{1+m} \times \Omega \to \mathbb{R}$ be a convolution function with the property:

$$h(u,v;\omega) = u, \quad u \in \mathbb{R}, v \in \mathbb{R}^m_-, \omega \in \Omega .$$

Then the Lagrange-type function $L(x,\omega)$, corresponding to h, is called a penalty-type function.

Next two examples cannot be presented in the form (2).

3) Augmented Lagrangians

Let $\sigma: \mathbb{R}^m \to \mathbb{R}$ be an augmenting function, i. e., $\sigma(0) = 0$ and $\sigma(z) > 0$, for $z \neq 0$, and $\Omega \subset \{(y,r): y \in \mathbb{R}^m, r \ge 0\}$ be a set of parameters satisfying $(0,0) \in \Omega$ and $(y,r) \in \Omega$ implying $(y,r') \in \Omega$, for all $r' \ge r$. Let $h: \mathbb{R}^m \times \Omega \to \mathbb{R}$ be the convolution function defined by

$$\begin{aligned} h(u,v;(y,r)) &= \inf_{z+v\le 0} (u - [y,z] + r\sigma(z)) \\ &= u + \inf_{z+v\le 0} (-[y,z] + r\sigma(z)) . \end{aligned}$$

Then the Lagrange-type function, corresponding to $\eta = 0$, coincides with the augmented Lagrangian [5], that is,

$$\begin{aligned} L(x,(y,r)) &= h(f(x), g(x); (y,r)) \\ &= \inf_{z+g(x)\le 0} (f(x) - [y,z] + r\sigma(z)) . \end{aligned}$$

4) Morrison-type functions. Let $\Omega = \mathbb{R}_+$ and

$$h(u,v,\omega) = ((u-\omega)^+)^2 + \sigma(v_1^+, \dots, v_m^+) ,$$

where σ is an augmenting function. Then the Lagrange-type function corresponding to $\eta = 0$ has the form

$$L(x,\omega) = ((f(x)-\omega)^+)^2 + \sigma(g_1(x)^+, \dots, g_m(x)^+) .$$

Functions of this kind have been introduced by Morrison [6].

Consider problem $P(f,g)$, a convolution function $h: \mathbb{R}^{1+m} \times \Omega \to \mathbb{R}$ and the corresponding Lagrange-type function

$$L(x,\omega) = h(f(x) - \eta, g(x); \omega) + \eta .$$

The dual function $q: \Omega \to \bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ of $P(f,g)$ with respect to h and η is defined by

$$q(\omega) = \inf_{x\in X} h(f(x) - \eta, g(x); \omega) + \eta, \quad \omega \in \Omega .$$

Consider the dual problem to $P(f,g)$ with respect to h and η:

$$\max q(\omega), \quad \text{subject to} \quad \omega \in \Omega .$$

We are interested in the following questions: Find conditions under which

1) the weak duality holds, i. e.,

$$M(f,g) := \inf_{x\in X_0} f(x) \ge \sup_{\omega\in\Omega} q(\omega) := M^*(f,g) ;$$

2) the zero duality gap property holds, i. e.,

$$\inf_{x\in X_0} f(x) = \sup_{\omega\in\Omega} q(\omega) ;$$

3) an exact Lagrange parameter exists, i. e., the weak duality holds and there exists $\bar{\omega} \in \Omega$ such that

$$\inf_{x\in X_0} f(x) = \inf_{x\in X} L(x,\bar{\omega}) ;$$

4) a strong exact parameter exists: there exists an exact parameter $\bar{\omega} \in \Omega$ such that

$$\begin{aligned}\operatorname{argmin} P(f,g) &:= \operatorname{argmin}_{x\in X_0} f(x)\\ &= \operatorname{argmin}_{x\in X} L(x,\bar{\omega})\,;\end{aligned}$$

5) a saddle point exists and generates a solution of $P(f,g)$. The first part of this question means that there exists $(x_*,\omega_*) \in X\times\Omega$ such that

$$\begin{aligned}L(x,\omega_*) \le L(x_*,\omega_*) \le L(x_*,\omega),\\ x\in X\,,\ \omega\in\Omega\,.\end{aligned} \tag{3}$$

The second part means that (3) implies $x_* \in \operatorname{argmin} P(f,g)$.

The weak duality allows one to estimate from below the optimal value $M(f,g)$ by solving the unconstrained problem $\inf_{x\in X} L(x,\omega)$. The zero duality gap property allows one to find $M(f,g)$ by solving a sequence of unconstrained problems $\inf_{x\in X} L(x,\omega_t)$ where $\{\omega_t\}\subset\Omega$. The existence of an exact Lagrange parameter $\bar{\omega}$ means that $M(f,g)$ can be found by solving one unconstrained problem $\inf_{x\in X} L(x,\bar{\omega})$. The existence of a strong exact parameter $\bar{\omega}$ means that the solution set of $P(f,g)$ is the same as that of $\min_{x\in X} L(x,\bar{\omega})$.

Let $h : \mathbb{R}^{1+m}\times\Omega\to\mathbb{R}$ be a convolution function such that

$$\sup_{\omega\in\Omega} h(u,v;\omega) \le u, \quad \text{for all } (u,v)\in\mathbb{R}\times\mathbb{R}^m_-\,. \tag{4}$$

Then the weak duality holds.

Condition (4) can be guaranteed if

$$\begin{aligned}h(u,v;w) = p(u,w_1v_1,\dots,w_mv_m)\,,\\ (u,v)\in\mathbb{R}^{1+m}, w\in\mathbb{R}^m_+\,,\end{aligned}$$

and $p\colon \mathbb{R}^{1+m}\to\mathbb{R}$ is an IPH function satisfying

$$p(1,0_m)\le 1, \quad p(-1,0_m)\le -1\,.$$

Assume that η is a lower estimate of the function f over the set X, i. e., $f(x)-\eta \ge b>0$, for all $x\in X$. Then, in order to establish the weak duality, we need only to consider convolution functions defined on $[b,+\infty)\times\mathbb{R}^m\times\Omega$ such that

$$\sup_{\omega\in\Omega} h(u,v;\omega)\le u, \quad \forall (u,v)\in[b,+\infty)\times\mathbb{R}^m_-\,. \tag{5}$$

To investigate the zero duality gap property, we further assume that, for any $\epsilon\in(0,b)$, there exists $\delta>0$ such that

$$\inf_{\omega\in\Omega} h(u,v;\omega)\ge u-\epsilon, \quad \forall u\ge b, r(v)\le\delta\,; \tag{6}$$

and that, for each $c>0$, there exists $\bar{\omega}\in\Omega$ such that

$$h(u,v;\bar{\omega})\ge cr(v), \quad \forall u\ge b, v\in\mathbb{R}^m\,, \tag{7}$$

where $r\colon\mathbb{R}^m\to\mathbb{R}$ is such that $r(v)\le 0 \iff v\in\mathbb{R}^m_-$.

Assume further that

(f_1) The function f is uniformly positive on X_0, i. e.,

$$\inf_{x\in X_0} f(x) = M(f,g)>0\,;$$

(f_2) The function f is uniformly continuous on an open set containing the set X_0;

(g) The mapping g is continuous and the set-valued mapping

$$D(\delta) = \{x\in X\colon r(g(x))\le\delta\}$$

is upper semi-continuous at the point $\delta=0$.

Theorem 1 *Under the assumptions (5)–(7) and (f_1), (f_2) and (g), the zero duality gap property holds for P(f,g) with respect to the Lagrange-type function $L(x,\omega)$, corresponding to h and $\eta=0$.*

Let $b\ge 0$. Define a convolution function $h\colon[b,+\infty)\times\mathbb{R}^m\to\mathbb{R}$ by

$$h(u,v;\omega) = p(u,\omega_1v_1,\dots,\omega_mv_m)\,,$$

where $p\colon\mathbb{R}_+\times\mathbb{R}^m\to\mathbb{R}$ is an increasing function satisfying

$$p(u,0_m)\le u, \quad \text{for all } u\ge 0\,. \tag{8}$$

Consider the $P(f,g)$ with uniformly positive objective function f on X. Let L be the Lagrange-type function defined by

$$L(x,\omega) = p(f(x),\omega_1g_1(x),\dots,\omega_mg_m(x))\,,$$

where p is defined on $\mathbb{R}_+\times\mathbb{R}^m$. Define the perturbation function $\beta(y)$ of $P(f,g)$ by

$$\beta(y) = \inf\{f(x)\colon x\in X, g(x)\le y\}, \quad y\in\mathbb{R}^m\,.$$

Theorem 2 *Let p be a continuous increasing function satisfying (7). Let the zero duality gap property with respect to p holds. Then the perturbation function β is lower semi-continuous at the origin.*

Further assume that p satisfies the following property: there exist positive numbers $a_1, \ldots, a_m$ such that, for all $u > 0$, $(v_1, \ldots, v_m) \in \mathbb{R}^m$, we have

$$p(u, v_1, \ldots, v_m) \geq \max(u, a_1 v_1, \ldots, a_m v_m)\,. \qquad (9)$$

Theorem 3 *Assume that p is an increasing convolution function that possesses properties (8) and (9). Let perturbation function β of problem P(f,g) be lower semi-continuous at the origin. Then the zero duality gap property with respect to p holds.*

Remark 1 The perturbation function β depends on $P(f,g)$ and doesn't depend on the exogenous function p. It is worth noting that Theorems 2 and 3 establish equivalence relations between the zero duality gap property with respect to different p from a broad class of convolution functions.

Remark 2 If p is a linear function, then the lower semicontinuity does not imply the zero duality gap property, so we need to impose a condition that does not hold for linear functions. This is the role of (9). The results similar to Theorem 2 and Theorem 3 hold also for penalty type functions, where $p(u,v)$ is a function defined on $\mathbb{R}_+ \times \mathbb{R}^m_+$ and $L(x,\omega) = p(f(x), \omega_1 g_1(x)^+, \ldots, g_m(x)^+)$. In such a case (9) should be valid only for $u > 0$, $v \in \mathbb{R}^m_+$. This requirement is very weak and is valid for many increasing functions including the function $p(u,v) = u + \sum_{i=1}^m v_i$.

Let the Lagrange-type function be of the following form

$$L(x,\omega) = f(x) + \chi(g(x);\omega), \quad x \in X, \omega \in \Omega\,.$$

Consider set K of functions $\chi\colon \mathbb{R}^m \times \Omega \to \mathbb{R}$ with the following two properties

(i) $\chi(\cdot,\omega)$ is lower semi-continuous for all $\omega \in \Omega$;

(ii) $\sup_{\omega\in\Omega} \chi(v;\omega) = 0$, for all $v \in \mathbb{R}^m_-$.

Consider a point $(x_*, \omega_*) \in X \times \Omega$ such that

$$L(x_*, \omega_*) = \min_{x\in X} L(x, \omega_*)\,, \qquad (10)$$

$$\chi(g(x_*);\omega_*) = 0\,. \qquad (11)$$

Theorem 4 *Let $\chi \in K$. If (10) and (11) hold for $x_* \in X_0$ and $\omega_* \in \Omega$, then ω_* is an exact Lagrange parameter.*

The most advanced theory has been developed for two special classes of Lagrange-type functions. One of them is augmented Lagrangians (see article in encyclopedia). The other class consists of penalty-type functions for problems with a positive objective and a single constraint. This penalty-type functions are composed by convolutions functions of the form (2) with IPH functions p.

Remark 3 Consider problem $P(f,g)$ with m constraints $g_1, \ldots, g_m$. We can convert these constraints to a single one by many different ways. In particular, the system $g_1(x) \leq 0, \ldots, g_m(x) \leq 0$ is equivalent to the single inequality $f_1(x) := \sum_{i=1}^m g_i^+(x) \leq 0$. The function f_1 is non-smooth. If all functions $g_i(x)$ are smooth then a smoothing procedure can be applied to f_1 (see [13] for details). Problems with a single constraint are convenient to be dealt with from many points of view.

Let $P(f,f_1)$ be a problem with a positive objective f and a single constraint f_1. We consider here only IPH functions s_k defined on $\mathbb{R}^2_+$ by:

$$s_k(u,v) = (u^k + v^k)^{1/k}, \quad u, v \geq 0\,. \qquad (12)$$

(Many results that are valid for s_k can be extended also for IPH functions $p\colon \mathbb{R}^2_+ \to \mathbb{R}_+$ with properties $p(1,0) = 1$, $\lim_{u\to+\infty} p(1,u) = +\infty$.)

A penalty-type function L_k^+ corresponding to s_k has the form $L_k^+(x,d) = (f(x)^k + d^k f_1^+(x)^k)^{1/k}$. Here d^k is a penalty parameter. It can be shown that the exact parameter does not exist if $k > 1$ for the 'regular' problems in a certain sense, so we will here consider only the classical penalty function with k=1 and lower order penalty functions with $k < 1$. It can be shown that the existence of an exact parameter for $k \leq 1$ implies the existence of exact parameters for k' with $0 < k' < k$. One of the main questions that can be studied in the framework of this class of penalty-type functions is the size of exact penalty parameters. Generally speaking, we can diminish the size of exact parameter using the choice of k and some simple reformulations of the problem $P(f,f_1)$ in hand.

For the function L_k^+ an explicit value of the least exact penalty parameter can be expressed through the

perturbation function. Let $\beta(y)$ be the perturbation function of the problem $P(f,f_1)$. Note that $\beta(0) = M(f,f_1)$ and β is a decreasing function, so $\beta(y) \le M(f,f_1)$. For the sake of simplicity, we assume that $\beta(y) < M(f,f_1)$ for all $y > 0$. Let

$$\bar{d}_k = \sup_{y>0} \frac{(M(f,f_1)^k - \beta^k(y))^{1/k}}{y}. \tag{13}$$

Then the least exact parameter exists if and only if the supremum in (13) is finite and the least exact parameter is equal to $\bar{d}_k$. For $k=1$ the existence easily follows from the calmness results of Burke [1].

Let $f^c(x) = f(x) + c$ with $c > 0$ and $d_{c,k}$ be the least exact parameter for problem $P(f^c, f_1)$. Then it can be proved that $d_{c,k} \to 0$ as $c \to +\infty$.

Assume that functions f and f_1 are Lipschitz. Since $k < 1$ the function L_k^+ is not locally Lipschitz at points x where $f_1(x) = 0$, so we need to have a special smoothing procedure in order to apply numerical method for the unconstrained minimization of this function. Such a procedure is described in [14]. This procedure can be applied for different types of lower order penalty functions.

Another approach for constructing a Lipschitz penalty function with a small exact parameter is also of interest (see [11] and references therein).

Let σ be a strictly increasing continuous concave function defined on $[a, +\infty)$ where $a > 0$. Assume that $\sigma(a) \ge 0$ and $\lim_{y\to+\infty} \sigma(y) = 0$ where σ'_+ is the right derivative of the concave function σ. Consider the function $f^{\sigma,c}(x) = \sigma(f(x) + c)$ and the classical penalty function for $L^+_{1,\sigma,c}(x,d) = \sigma(f(x)+c) + df_1(x)$ for the problem $P(f^{\sigma,c}, f_1)$. Let $d_{\sigma,c}$ be the least exact parameter of $L^+_{1,\sigma,c}$ (assuming that this parameter exists). Then we can assert that $d_{\sigma,c} \to 0$ as $c \to 0$ under very mild assumptions.

References

1. Burke JV (1991) Calmness and exact penalization. SIAM J Control Optim 29:493–497
2. Burke JV (1991) An exact penalization viewpoint of constrained optimization. SIAM J Control Optim 29:968–998
3. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
4. Giannessi F (2005) Constrained optimization and image space analysis, vol 1. Separation of sets and optimality conditions. Mathematical concepts and methods in science and engineering, vol 49. Springer, New York, p 395
5. Huang XX, Yang XQ (2003) A unified augmented lagrangian approach to duality and exact penalization. Math Oper Res 28(3):533–552
6. Morrison DD (1968) Optimization by least squares. SIAM J Numer Anal 5:83–88
7. Rockafellar RT, Wets RJ-B (1998) Variational analysis. Springer, Berlin
8. Rubinov AM (2000) Abstract convexity and global optimization. Kluwer, Dordrecht
9. Rubinov AM, Glover BM, Yang XQ (2000) Decreasing functions with application to penalization. SIAM J Optim 10:289–313
10. Rubinov AM, Glover BM, Yang XQ (1999) Modified lagrangian and penalty functions in continuous optimization. Optimization 46:327–351
11. Rubinov AM, Yang XQ (2003) Lagrange-type functions in constrained non-convex optimization. Kluwer, Dordrecht
12. Rubinov AM, Yang XQ, Bagirov AM (2002) Nonlinear penalty functions with a small penalty parameter. Optim Methods Softw 17:931–964
13. Teo KL, Goh CJ, Wong KH (1991) A unified computational approach to optimal control problems. In: Pitman monographs and surveys in pure and applied mathematics, vol 55. Longman Scientific and Technical, Harlow, p 329
14. Wu ZY, Bai FS, Yang XQ, Zhang LS (2004) An exact lower order penalty function and its smoothing in nonlinear programming. Optimization 53:51–68
15. Yang XQ, Huang XX (2001) A nonlinear lagrangian approach to constrained optimization problems. SIAM J Optim 14:1119–1144
16. Yevtushenko YG, Zhadan VG (1990) Exact auxiliary functions in optimization problems. USSR Comput Math Math Phys 30:31–42

Lagrangian Duality: BASICS

Donald W. Hearn[1], Timothy J. Lowe[2]

[1] University Florida, Gainesville, USA

[2] University Iowa, Iowa City, USA

MSC2000: 90C30

Article Outline

Keywords

The Primal Problem and the Lagrangian Dual Problem

Weak and Strong Duality

Properties of the Lagrangian Dual Function

Geometrical Interpretations
of Lagrangian Duality
The Resource-Payoff Space
Gap Function
Summary
See also
References

Keywords

Primal optimization problem; Dual optimization problem; Subgradient; Duality gap; Constraint qualification

The Primal Problem and the Lagrangian Dual Problem

For a given *primal optimization problem* (P) it is possible to construct a related *dual* problem which depends on the same data and often facilitates the analysis and solution of (P). This section focuses on the *Lagrangian dual*, a particular form of dual problem which has proven to be very useful in many optimization applications.

A general form of *primal problem* is

$$(\mathrm{P}) \quad \begin{cases} \min & f(x) \\ \text{s.t.} & g(x) \le 0, \\ & h(x) = 0, \\ & x \in S, \end{cases}$$

where f is a scalar function of the n-dimensional vector x, and g and h are vector functions of x. S is a nonempty subset of $\mathbf{R}^n$. It is convenient to associate *dual variables* with the constraints as follows: components of the *dual vector* u correspond to components of the vector constraint $g(x) \le 0$, and similarly the components of v are associated with components of the constraint $h(x) = 0$.

There is a great deal of flexibility in defining problem (P). For example, any or all of the explicit constraints $g(x) \le 0$ and $h(x) = 0$ could be incorporated in the definition of the set S. This, of course, governs the number and type of dual variables. As will be seen in the examples, defining S is the first step in defining a Lagrangian dual of (P). To illustrate the basic duality results, certain assumptions regarding the functions f, g and h and the set S will be made to simplify the presentation below. For more thorough treatments, see the references.

Given (P), define the *Lagrangian function* $L(x, u, v) = f(x) + u^{\mathsf{T}} g(x) + v^{\mathsf{T}} h(x)$. The *Lagrangian dual problem* is then

$$(\mathrm{D}) \quad \begin{cases} \max & \theta(u, v) \\ \text{s.t.} & u \ge 0, \end{cases}$$

where, for fixed (u, v), the dual function θ is defined in terms of the *infimum of the Lagrangian function* with respect to $x \in S$:

$$\theta(u, v) = \inf_{x \in S} L(x, u, v).$$

Below are five examples of primal problems and their duals. The first is a geometrical example, three are classes of optimization problems: linear programs, convex programs, and quadratic programs, and the final example is an integer program.

Example 1 (geometrical problem) In this two-variable example, a linear function is to be minimized over the intersection of the unit disk and the nonnegative orthant. The optimal solution is at the origin with objective value zero.

$$(\mathrm{P1}) \quad \begin{cases} \min & x_1 + x_2 \\ \text{s.t.} & x_1^2 + x_2^2 \le 1, \\ & -x_1 \le 0, \\ & -x_2 \le 0, \end{cases}$$

Letting $S = \{(x_1, x_2): x_1^2 + x_2^2 \le 1\}$, the dual problem is

$$(\mathrm{D1}) \quad \begin{cases} \max & \theta(u) \\ \text{s.t.} & u \ge 0, \end{cases}$$

where

$$\theta(u) = \min_{x_1^2 + x_2^2 \le 1} (1 - u_1)x_1 + (1 - u_2)x_2.$$

Note that min replaces inf in the definition of θ since it is clear that the infimum exists and is finite for this example.

Example 2 (linear programming) Duality is an important topic in any treatment of linear programming. This example shows that the Lagrangian dual of a primal linear program is equivalent to the *dual linear program* as

it is usually formulated in textbooks. Letting the primal be

$$\text{(P2)} \quad \begin{cases} \min & c^\top x \\ \text{s.t.} & b - Ax \le 0, \\ & x \ge 0, \end{cases}$$

and choosing $S = \{x: x \ge 0\}$, the Lagrangian dual is

$$\text{(D2)} \quad \begin{cases} \max & \theta(u) \\ \text{s.t.} & u \ge 0 \end{cases}$$

where

$$\theta(u) = \inf_{x \ge 0} c^\top x + u^\top (b - Ax).$$

This reduces to

$$\theta(u) = b^\top u + \begin{cases} 0 & \text{if } (c - A^\top u) \ge 0, \\ -\infty & \text{otherwise.} \end{cases}$$

Assuming there are nonnegative values of u such that $c \ge A^\intercal u$, these would be the only viable choices for the maximization of $\theta(u)$ and therefore (D2) takes the form familiar from linear programming duality:

$$\begin{cases} \max & b^\top u \\ \text{s.t.} & A^\top u \le c, \\ & u \ge 0. \end{cases}$$

Example 3 (differentiable convex programming) One of the first nonlinear duals was developed by P. Wolfe [27] for the primal problem

$$\text{(P3)} \quad \begin{cases} \min & f(x) \\ \text{s.t.} & g(x) \le 0, \\ & x \in S, \end{cases}$$

where S is an open convex set, and f and g are differentiable convex functions defined on S. The Lagrangian function is $L(x, u) = f(x) + u^\intercal\, g(x)$, and it is further assumed that $\theta(u) \ne -\infty$ for all $u \ge 0$. With these assumptions the Lagrangian function is convex in x and has a minimum where its gradient is zero. That is, the requirement $\theta\ (u) = \min_{x \in S} L(x, u)$ is the same as requiring $\nabla_x L(x, u) = 0$. Thus the dual problem may be written

$$\text{(D3)} \quad \begin{cases} \max & L(x, u) \\ \text{s.t.} & \nabla_x L(x, u) = 0, \\ & u \ge 0. \end{cases}$$

Example 4 (convex quadratic programming) An important special case of the preceding example is the problem

$$\text{(P4)} \quad \begin{cases} \min & \frac{1}{2} x^\top H x + d^\top x \\ \text{s.t.} & Ax \le b, \\ & x \in \mathbb{R}^n, \end{cases}$$

where H is a given symmetric positive definite $n \times n$ matrix and d is a given vector in $\mathbf{R}^n$. Applying the results for (P3) above and using the equality constraints of (D3) to eliminate x, the dual of can be written

$$\text{(D4)} \quad \begin{cases} \max & \theta(u) = -\frac{1}{2} u^\top (AH^{-1}A^\top) u \\ & \quad -u^\top (b + AH^{-1}d) - \frac{1}{2} d^\top H^{-1} d \\ \text{s.t.} & u \ge 0. \end{cases}$$

Thus, the dual of (P4) is also a quadratic program in the dual variables u.

Example 5 (integer program) The following numerical example of a linear problem with binary variables will be used to illustrate various dual properties in the following sections.

$$\text{(P5)} \quad \begin{cases} \min & 20 - x_1 - 5x_2 - 7x_3 \\ \text{s.t.} & x_1 + 3x_2 + 4x_3 \le 5, \\ & x_j \in \{0, 1\}, \quad j = 1, 2, 3. \end{cases}$$

For this problem, let S be defined by the binary restrictions on the components of x. Then $L(x, u) = 20 - x_1 - 5x_2 - 7x_3 + u(x_1 + 3x_2 + 4x_3 - 5)$ and the dual problem is

$$\text{(D5)} \quad \begin{cases} \max & \theta(u) \\ \text{s.t.} & u \ge 0, \end{cases}$$

where

$$\begin{cases} \theta(u) = \min & (u - 1)x_1 + (3u - 5)x_2 \\ & \quad +(4u - 7)x_3 - 5u + 20 \\ \text{s.t.} & x_j \in \{0, 1\}, \quad j = 1, 2, 3. \end{cases}$$

Weak and Strong Duality

For a given primal problem (P) and associated dual problem (D), a fundamental relationship showing that

the two objective function values bound each other is given by the following *weak duality* result:

Theorem 6 *If $\overline{x}$ is feasible to (P) and $(\overline{u}, \overline{v})$ is feasible to (D), then*

$$f(\overline{x}) \geq \theta(\overline{u}, \overline{v}).$$

Proof

$$\begin{aligned}\theta(u, v) &= \inf_{x \in S} L(x, u, v) \\ &\leq f(\overline{x}) + \overline{u}^{\top} g(\overline{x}) + \overline{v}^{\top} h(\overline{x}) \leq f(\overline{x}).\end{aligned}$$

The first inequality follows since $\overline{x} \in S$ and the second from $\overline{u}^{\top} g(\overline{x}) \leq 0$ and $h(h(\overline{x}) = 0$. □

If the optimal primal and dual objective values are equal, *strong duality* is said to hold for the primal and dual pair. The following theorem illustrates such a result for the the pair (P3) and (D3).

Theorem 7 *Let x^* be an optimal solution for (P3) and assume the function g satisfies some constraint qualification. Then there exists a vector u^* such that (x^*, u^*) solves (D3) and*

$$f(x^*) = L(x^*, u^*).$$

Proof Under the assumptions there exists a $u^* \geq 0$ such that (x^*, u^*) satisfies the *Karush–Kuhn–Tucker conditions*:

$$\begin{aligned}&\nabla_x L(x^*, u^*) = 0, \\ &u^{*T} g(x^*) = 0,\end{aligned}$$

from which it follows that

$$f(x^*) = L(x^*, u^*)$$

and that (x^*, u^*) is feasible to (D3). Using this and the weak duality theorem gives

$$L(x^*, u^*) \geq L(x, u)$$

for any (x, u) satisfying the constraints of (D3). The results of the theorem follow. □

The references contain additional strong duality results, including cases where differentiability is not required. However, as will be seen in examples below, it often happens that there is a difference, known as the *duality gap*, between the optimal values of the primal and dual objective functions.

Properties of the Lagrangian Dual Function

The Lagrangian dual function enjoys two useful properties: it is concave and, although it is not necessarily differentiable, it is relatively straightforward to compute a *subgradient* at any dual feasible point.

Theorem 8 *θ (u, v) is concave.*

Proof For fixed x, $L(x, u, v)$ is linear in (u, v) and thus $\theta(u, v)$ is the infimum of a (possibly infinite) collection of functions linear in (u, v). □

It is important to note that the above result is true under very general conditions. In particular, it is true when the set S is discrete.

Since θ (u, v) is concave, it is known that at least one *linear supporting function* exists at each (u, v). Collectively, the gradients of all linear supports at (u, v) is called the set of *subgradients* of θ at (u, v).

For any (u, v) for which $\theta(u, v)$ is finite, denote $S(u, v)$ as the solution set of the minimization defining $\theta(u, v)$.

Theorem 9 *For fixed $(\overline{u}, \overline{v})$, let $\overline{x} \in S(\overline{u}, \overline{v})$. Then $(g(\overline{x}), h(\overline{x}))$ is a subgradient of θ at $(\overline{u}, \overline{v})$.*

Proof For any (u, v)

$$\begin{aligned}\theta(u, v) &= \inf_{x \in S} f(x) + u^{\top} g(x) + v^{\top} h(x) \\ &\leq f(\overline{x}) + u^{\top} g(\overline{x}) + v^{\top} h(\overline{x}) \\ &= f(\overline{x}) + (u - \overline{u})^{\top} g(\overline{x}) + \overline{u}^{\top} g(\overline{x}) \\ &\quad + (v - \overline{v})^{\top} h(\overline{x}) + \overline{v}^{\top} h(\overline{x}).\end{aligned}$$

Hence

$$\theta(u, v) \leq \theta(\overline{u}, \overline{v}) + g(\overline{x})^{\top}(u - \overline{u}) + h(\overline{x})^{\top}(v - \overline{v}).$$

□

If $S(\overline{u}, \overline{v})$ is a single point $\overline{x}$, then there is only one subgradient of θ at $(\overline{u}, \overline{v})$ in which case θ is differential at $(\overline{u}, \overline{v})$, i. e., $\nabla\theta(\overline{u}, \overline{v}) = (g(\overline{x}), h(\overline{x}))$.

From the above, θ is always concave and it is relatively easy to calculate a slope at any point. Much use of this is made in algorithms for large scale integer programs. Also, the fact that the maximum value of the dual provides a lower bound to the optimal objective function value in methods (such as *branch and bound*) for solving the primal problem. While strong duality

generally holds for convex programs, this is rarely true for integer programs.

Revisiting the examples of the first section, for Example 1 the Karush–Kuhn–Tucker conditions can be employed to derive

$$\theta(u) = -\sqrt{(1-u_1)^2 + (1-u_2)^2}.$$

There is no duality gap for this problem, the dual maximum occurs at $(u_1, u_2) = (1, 1)$ where θ is zero, in agreement with the primal minimum. The dual function is differentiable except at its maximizing point. The dual function of Example 2, a linear program, is linear and thus it is concave and differentiable everywhere. Similarly, in Example 4, since H is positive definite, H^{-1} is also positive definite and the dual function is again concave and differentiable everywhere. For Example 5, the integer program, values of u feasible to the dual problem, $S(u)$ and $\theta(u)$ are given in Table 1. $S(u)$ is the triple $(x_1(u), x_2(u), x_3(u))$.

Figure 1 is a graph of the function $\theta(u)$. Again, $\theta(u)$ is a concave function and it is differentiable except at

$$u = 1, \frac{5}{3}, \frac{7}{4}.$$

The maximum dual value is

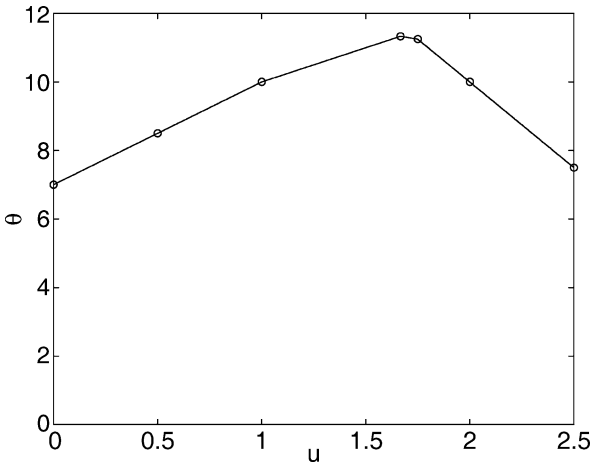
$$\theta\left(\frac{5}{3}\right) = 11\frac{1}{3},$$

which indicates a duality gap of size 2/3 since the optimal value of (P5) is $f(1, 0, 1) = 12$.

By contrast, Theorem 8 does not apply in Example 3 because the objective of (D3), a Lagrangian function, depends on both x and u, rather than the dual variables alone. Lagrangian functions are generally not concave.

Lagrangian Duality: BASICS, Table 1
Values of the dual function for Example 5

u	$S(u)$	$\theta(u)$
$0 < u < 1$	$(1, 1, 1)$	$7 + 3u$
1	$\{(1, 1, 1) \cup (0, 1, 1)\}$	$8 + 2u$
$1 < u < 5/3$	$(0, 1, 1)$	$8 + 2u$
5/3	$\{(0, 1, 1) \cup (0, 0, 1)\}$	$13 - u$
$5/3 < u < 7/4$	$(0, 0, 1)$	$13 - u$
7/4	$\{(0, 0, 1) \cup (0, 0, 0)\}$	$20 - 5u$
$7/4 < u$	$(0, 0, 0)$	$20 - 5u$



Lagrangian Duality: BASICS, Figure 1
***θ (u)* for Example 5**

Geometrical Interpretations of Lagrangian Duality

The Resource-Payoff Space

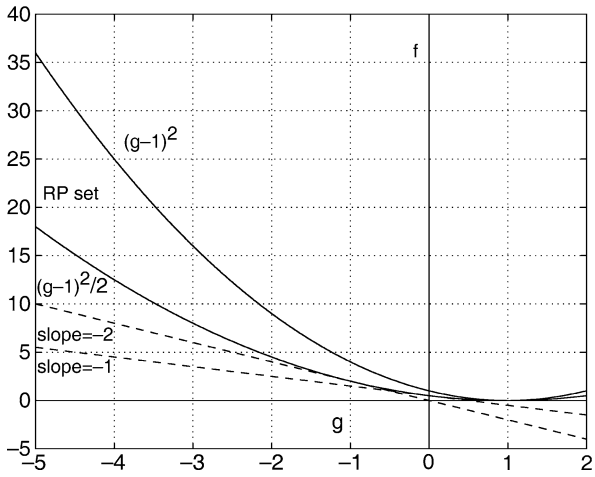
One interpretation of the dual problem is provided via the resource-payoff set RP for problem (P). To illustrate geometrically, assume that (P) has just one inequality constraint $g(x) \leq 0$ and there are no explicit equality constraints. Then the resource-payoff set for the problem is the set of points defined by

$$\mathrm{RP} = \{(g(x), f(x)) : \ x \in S)\}.$$

That is, RP is a mapping of all $x \in S$ into the (g, f)-plane. In this plane, the Lagrangian equated to a constant θ has the form $f + ug = \theta$, which defines a line of slope $-u$ and intercept θ. For any $u \geq 0$, the dual function $\theta(u)$ is defined by minimizing $f(x) + ug(x)$ over $x \in S$. Thus $\theta\ (u)$ is the intercept of a linear support to the resource-payoff set at $\{(g(x), f(x)) : x \in S(u)\}$. To illustrate, consider the problem

$$\text{(P6)} \quad \begin{cases} \min & x_1^2 + x_2^2 \\ \text{s.t.} & 1 - x_1 - x_2 \leq 0, \\ & -x_1 \leq 0, \\ & -x_2 \leq 0. \end{cases}$$

The optimal solution is $x_1^* = x_2^* = 1/2$ and $f(x_1^*, x_2^*) = 1/2$. Letting $S = \{(x_1, x_2) : x_1 \geq 0, x_2 \geq 0\}$, and $g(x_1, x_2) = 1 - x_1 - x_2$, the resource-payoff set is a subset of $\mathbf{R}^2$ defined by $\mathrm{RP} = \{(g(x_1, x_2), f(x_1, x_2)) : x_1 \geq 0, x_2 \geq$



Lagrangian Duality: BASICS, Figure 2
RP set for (P6)

0}. It can be verified that RP consists of all points in $\mathbf{R}^2$ between the curves $(g-1)^2$ and $(g-1)^2/2$ for $g \leq 1$ as shown in Fig. 2.

The two linear supports of RP shown have slopes of -2 and -1, corresponding to u values of 2 and 1. With $u = 2$, $(x_1(u), x_2(u))$ is the singleton (1, 1) and $(g(x(u)), f(x(u))) = (-1, 2)$. The line with slope -2 passing through the point $(g, f) = (-1, 2)$ intersects the f-axis at the origin. Thus $\theta(2) = 0 < f(x^*)$, illustrating the weak duality theorem.

For $u = 1$, $(x_1(u), x_2(u)) = (1/2, 1/2)$ and $(g(x(u)), f(x(u))) = (0, 1/2)$. Since this point lies on the f-axis it follows that $\theta(1) = 1/2 = f(x^*)$. This illustrates the strong duality theorem.

As an alternative consider Example 5, the *binary linear programming* problem. Since S is discrete, RP consists of the eight points in $\mathbf{R}^2$ listed in the last two columns of Table 2. The optimal solution to the problem is $x^* = (1, 0, 1), f(x^*) = 12$. The resource-payoff set for this example is shown in Fig. 3. The lines in the figure trace out the lower envelope of the resource-payoff set and are found by minimizing the Lagrangian function over S using $u^1 = 7/4$, $u^2 = 5/3$ and $u^3 = 1$. The lines with slope $-7/4$, $-5/3$, and -1 intersect the f-axis at

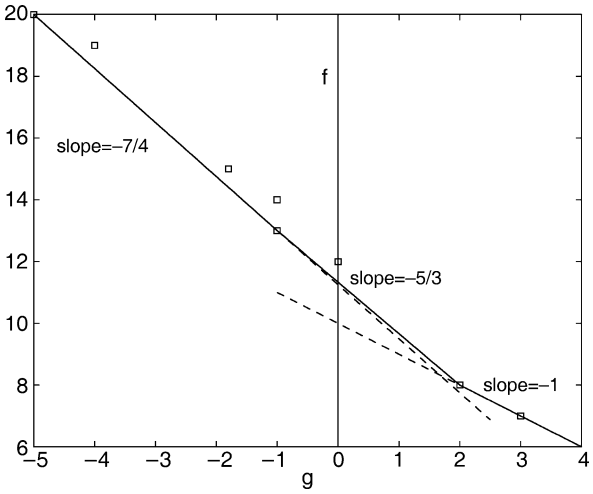
$$11\frac{1}{4}, 11\frac{1}{3}, 10,$$

respectively. Thus

$$\theta\left(\frac{7}{4}\right) = 11\frac{1}{4},\ \theta(1) = 10,\ \theta\left(\frac{5}{3}\right) = 11\frac{1}{3}.$$

Lagrangian Duality: BASICS, Table 2
Values of *g* and *f* for Example 5

x_1	x_2	x_3	$g(x_1, x_2, x_3)$ $x_1 + 3x_2 + 4x_3 - 5$	$f(x_1, x_2, x_3)$ $20 - x_1 - 5x_2 - 7x_3$
0	0	0	−5	20
0	0	1	−1	13
0	1	0	−2	15
0	1	1	2	8
1	0	0	−4	19
1	0	1	0	12
1	1	0	−1	14
1	1	1	3	7



Lagrangian Duality: BASICS, Figure 3
The set RP for Example 5

The duality gap for this problem, as noted earlier, is

$$f(x^*) - \theta(u^*) = 12 - 11\frac{1}{3} = \frac{2}{3}.$$

These two examples illustrate a sufficient condition for there to be no duality gap. There is no gap if the point $(g(x^*), f(x^*))$ lies on the *lower envelope* of the resource-payoff set, and there is a *linear support* of slope $-u \leq 0$ at that point with intercept $f(x^*)$.

This condition is satisfied for (P6), but not for (P5). However, if the constraint in (P5) is replaced by $g(x) = x_1 + 2x_2 + 4x_3 - 4 \leq 0$, the condition is satisfied. The effect of this constraint change can be seen in Table 2 and Fig. 3. In the table, the g column entries would be

increased by 1, and thus $f(x^*) = 13$, with $x^* = (x_1^*, x_2^*, x_3^*) = (0, 0, 1)$. In Fig. 3, the f-axis would be shifted one unit to be left. In this case, note that $(g(x^*), f(x^*))$ now lies on the lower envelope of the set RP. Furthermore, an optimal dual variable is any value $u^* \in [5/3, 7/4]$.

Gap Function

Another geometrical interpretation can be given for the primal problem

$$(\text{P7}) \quad \begin{cases} \min & f(x) \\ \text{s.t.} & Ax = b, \\ & x \geq 0, \end{cases}$$

where f is assumed convex and differentiable. In what follows, let $S = \{x \in \mathbf{R}^n : Ax = b, x \geq 0\}$ which is assumed to be a compact subset of $\mathbf{R}^n$.

For any feasible x, define the *gap function* by

$$\begin{aligned} G(x) &= \max_{y \in S} \nabla f(x)^\top (x - y) \\ &= -\min_{y \in S} \nabla f(x)^\top (y - x). \end{aligned}$$

The gap function has several interesting properties. Letting $y(x)$ be the solution of the linear program defining $G(x)$, note first that the gap function at x is the negative of the *directional derivative* of f at x in the direction $(y(x) - x)$. Second, it can be used to construct a lower bound on the optimal solution $f(x^*)$ of (P7). To see this, consider the convexity inequality

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x), \quad \forall y \in S.$$

Minimizing both sides over $y \in S$ implies

$$f(x^*) \geq f(x) - G(x), \quad \forall x \in S.$$

By the weak duality result, a lower bound for $f(x^*)$ can also be obtained by evaluating the dual objective at any dual feasible solution. The next theorem employs the Wolfe dual of (P7) to show that the bound given above is equivalent to obtaining the maximum dual objective value for a given x.

Let v and u be the dual variables associated with $Ax = b$ and $x \geq 0$, respectively. The *Lagrangian function* of (P7) is $L(x, v, u) = f(x) + v^\intercal(b - Ax) - u^\intercal x$. Then, for the given x, the maximum dual objective value is

$$d(x) = \max_{(v,u) \in D(x)} L(x, v, u),$$

where $D(x)$ is the set of all multipliers such that (x, v, u) is feasible to the Wolfe dual:

$$D(x) = \{(v, u) : \nabla_x L(x, v, u) = 0 \text{ and } u \geq 0\}.$$

Theorem 10 *For any $x \in S$, $G(x) = f(x) - d(x)$.*

Proof First it is verified that $D(x)$ is nonempty so that $d(x)$ is well defined. This will be true if there exists a v such that $A^\intercal v \leq \nabla f(x)$. By adaptation of Farkas' lemma (cf. also ► Farkas lemma; ► Farkas lemma: Generalizations) such a v exists if and only if the alternative system

$$\nabla f(x)^\top z < 0, \quad Az = 0, \quad z \geq 0$$

has no solution. However $Az = 0$, $z \geq 0$ imply that $x + \lambda z \in S$ for all $\lambda \geq 0$. Since S is assumed to be compact, the only possibility is $z = 0$ and the alternative system has no solution. Thus $D(x)$ is nonempty.

The dual constraints imply that $u^\intercal x = \nabla f(x)^\intercal x - v^\intercal Ax$, so

$$d(x) = \begin{cases} \max\limits_{v} & f(x) - \nabla f(x)x + v^\top b \\ \text{s.t.} & A^\top v \leq \nabla f(x). \end{cases}$$

By linear programming duality

$$\begin{cases} \max & b^\top v \\ \text{s.t.} & A^\top v \leq \nabla f(x) \end{cases} = \begin{cases} \min & \nabla f(x)^\top y \\ \text{s.t.} & Ay = b \\ & y \geq 0, \end{cases}$$

and it follows that

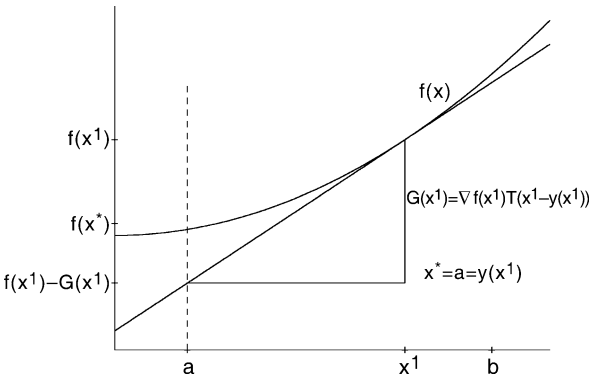
$$\begin{aligned} d(x) &= f(x) + \min_{y \in S} \nabla f(x)^\top (y - x) \\ &= f(x) - G(x). \end{aligned}$$

□

Expressing the duality gap in terms of x allows a simple interpretation of weak and strong duality in the convex case. Figure 4 illustrates the gap function in one variable with S being the interval $[a, b]$. Let $x = x^1$. The linear function

$$f(x^1) + \nabla f(x^1)^\top (y - x^1)$$

is the tangent line shown. It has a minimum in S at $y(x^1) = a$ which, by convexity, must lie below $f(x^*)$. Hence the weak duality result holds: $f(x^*) \geq f(x^1) - G(x^1) =$



Lagrangian Duality: BASICS, Figure 4
A one variable interpretation of weak and strong duality

$d(x^1)$. Strong duality occurs when $x^1 = x^*$ and the minimum of the linear function (i. e., the tangent at x^*) has the value $f(x^*)$. In this case $G(x^*) = 0$. If x^* were at an interior point of S, and/or if x^1 is infeasible to S, this same interpretation holds provided only that $f(x^1)$ and $\nabla f(x^1)$ are defined.

Summary

This section has illustrated basic results and geometrical interpretations of Lagrangian duality. The reference list below is a selection of texts and journal articles on this topic for further reading.

See also

- ► Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- ► First Order Constraint Qualifications
- ► Inequality-constrained Nonlinear Optimization
- ► Kuhn–Tucker Optimality Conditions
- ► Rosen's Method, Global Convergence, and Powell's Conjecture
- ► Saddle Point Theory and Optimality Conditions
- ► Second Order Constraint Qualifications
- ► Second Order Optimality Conditions for Nonlinear Optimization

References

1. Balinski ML, Baumol WJ (1968) The dual in nonlinear programming and its economic interpretation. Rev Economic Stud 35:237–256
2. Bazaraa MS, Goode JJ (1979) A survey of various tactics for generating Lagrangian multipliers in the context of Lagrangian duality. Europ J Oper Res 3:322–338
3. Bertsekas DP (1975) Nondifferentiable optimization. North-Holland, Amsterdam
4. Bertsekas DP (1982) Constrained optimization and Lagrange multiplier methods. Acad. Press, New York
5. Bertsekas DP (1995) Nonlinear programming. Athena Sci., Belmont, MA
6. Brooks R, Geoffrion A (1966) Finding Everett's Lagrange multipliers by linear programming. Oper Res 16:1149–1152
7. Everett H (1973) Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Oper Res 4:72–97
8. Falk JE (1967) Lagrange multipliers and nonconvex programming. J Math Anal Appl 19:141–159
9. Fiacco AV, McCormick GP (1968) Nonlinear programming: Sequential unconstrained minimization techniques. Wiley, New York
10. Fisher ML, Northup WD, Shapiro JF (1975) Using duality to solve discrete optimization problems: theory and computational experience. In: Balinski ML, Wolfe P (eds) Nondifferentiable Optimization. North-Holland, Amsterdam
11. Fletcher R (ed) (1969) Optimization. Acad. Press, New York
12. Geoffrion AM (1970) Elements of large- scale mathematical programming I-II. Managem Sci 16:652–675; 676–691
13. Geoffrion AM (1971) Duality in nonlinear programming: A simplified application-oriented development. SIAM Rev 13:1–7
14. Hearn DW (1982) The gap function of a convex program. Oper Res Lett 1:67–71
15. Hearn DW, Lawphongpanich S (1989) Lagrangian dual ascent by generalized linear programming. Oper Res Lett 8:189–196
16. Hearn DW, Lawphongpanich S (1990) A dual ascent algorithm for traffic assignment problems. Transport Res B 248(6):423–430
17. Kiwiel KC (1985) Methods of descent for nondifferentiable optimization. Springer, Berlin
18. Lasdon LS (1970) Optimization theory for large systems. MacMillan, New York
19. Luenberger DG (1969) Optimization by vector space methods. Wiley, New York
20. Luenberger DG (1973) Introduction to linear and nonlinear programming. Addison-Wesley, Reading, MA
21. Mangasarian OL (1969) Nonlinear programming. McGraw-Hill, New York
22. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
23. Powell MJD (1978) Algorithms for nonlinear constraints that use Lagrangian functions. Math Program 14:224–248
24. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton

25. Rockafellar RT (1975) Lagrange multipliers in optimization. In: Cottle RW, Lemke CE (eds) Nonlinear Programming, SIAM-AMS Proc. vol IX, pp 23–24
26. Whittle P (1971) Optimization under constraints. Wiley, New York
27. Wolfe P (1961) A duality theorem for nonlinear programming. Quart Appl Math 19:239–244
28. Zangwill WI (1969) Nonlinear programming: A unified approach. Prentice-Hall, Englewood Cliffs, NJ

Lagrangian Multipliers Methods for Convex Programming

LMM

MARC TEBOULLE
School Math. Sci., Tel-Aviv University,
Ramat-Aviv, Tel-Aviv, Israel

MSC2000: 90C25, 90C30

Article Outline

Keywords
Augmented Lagrangians
Quadratic Lagrangian
Proximal Minimization
Modified Lagrangians
See also
References

Keywords

Augmented Lagrangians; Convex optimization; Lagrangian multipliers; Primal-dual methods; Proximal algorithms

Optimization problems concern the minimization or maximization of functions over some set of conditions called constraints. The original treatment of constrained optimization problems was to deal only with equality constraints via the introduction of *Lagrange multipliers* which found their origin in basic mechanics. Modeling real world situations often requires using inequality constraints leading to more challenging optimization problems. Lagrange multipliers are used in optimality conditions and play a key role to devise algorithms for constrained problems. What will be summarized here are the basic elements of various algorithms based on Lagrangian multipliers to solve constrained optimization problems, and particularly convex optimization problems. A standard formulation of an optimization problem is:

$$(\mathrm{O}) \quad \min\{f(x):\ x \in X \cap C\},$$

where X is a certain subset of $\mathbf{R}^n$ and C is the set of constraints described by equality and inequality constraints

$$C = \left\{ x \in \mathbb{R}^n : \begin{array}{c} g_i(x) \le 0,\ i = 1, \ldots, m, \\ h_i(x) = 0,\ i = 1, \ldots, p \end{array} \right\}.$$

All the functions in problem (O) are real valued functions on $\mathbf{R}^n$, and the set X can described more abstract constraints of the problem. A point $x \in X \cap C$ is called a *feasible solution* of the problem, and an *optimal solution* is any feasible point where the local or global minimum of f relative to $X \cap C$ is actually attained. By a *convex problem* we mean the case where X is a convex set, the functions $f, g_1, \ldots, g_m$ are convex and $h_1, \ldots, h_p$ are affine. Recall that a set $S \subset \mathbf{R}^n$ is *convex* if the line segment joining any two different points of S is contained in it.

Let S be a convex subset of $\mathbf{R}^n$. A real valued function $f\colon S \to \mathbf{R}$ is convex if for any $x, y \in S$ and any $\lambda \in [0, 1]$,

$$f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y).$$

Convexity plays a fundamental role in optimization (even in nonconvex problems). One of the key fact is that when a convex function is minimized over a convex set, every local optimal solution is global. Another, fundamental point is that a powerful duality theory can be developed for convex problems, which as we shall see, is also at the root of the development and analysis of Lagrangian multiplier methods.

Augmented Lagrangians

The basic idea of augmented Lagrangian methods for solving constrained optimization problems, also called *multiplier methods*, is to transform a constrained problem into a sequence of unconstrained problems. The approach differs from the penalty-barrier methods, [13] from the fact that in the functional defining the unconstrained problem to be solved, in addition to a penalty parameter, there are also multipliers associated with the

constraints. Multiplier methods can be seen as a combination of penalty and dual methods. The motivation for these methods came from the desire of avoiding ill-conditioning associated with the usual penalty-barrier methods. Indeed, in contrast to penalty methods, the penalty parameter need not to go to infinity to achieve convergence of the multiplier methods. As a consequence, the augmented Lagrangian has a 'good' conditioning, and the methods are robust for solving nonlinear programs. Augmented Lagrangians methods were proposed independently by M.R. Hestenes [16] and M.J.D. Powell [26] for the case of equality constraints, and extended for the case of inequality constraints by R.T. Rockafellar [27]. Many other researchers have contributed to the development of augmented Lagrangian methods, and for an excellent treatment and comprehensive study of multiplier methods, see [7] and references therein.

Quadratic Lagrangian

We start by briefly describing the basic steps involved in generating a multiplier method for the equality constrained problem

$$\text{(E)} \quad \min\{f(x):\ h_i(x) = 0,\ i = 1, \ldots, p\}.$$

Here f and h_i are real valued functions on $\mathbf{R}^n$ and no convexity is assumed (which will not help anyway because of the nonlinear equality constraints). Also for simplicity we let $X = \mathbf{R}^n$. The ordinary Lagrangian associated with (E) is

$$l(x, y) = f(x) + \sum_{i=1}^{p} y_i h_i(x).$$

One of the oldest and simplest way to solve (E) is by sequential minimization of the Lagrangian ([2]). Namely, we start with an initial multiplier y_k and minimize $l(x, y^k)$ over $x \in \mathbf{R}^n$ to produce x^k. We then update the multiplier sequence via the formula:

$$y_i^{k+1} = y_i^k + s^k h_i(x^k), \quad i = 1, \ldots, p,$$

where s_k is a stepsize parameter. The rational behind the above method is that it can be simply interpreted as a gradient-type algorithm to solve an associated dual problem. Unfortunately, such a method while simple requires too many assumptions on the problem's data to generate points converging rapidly toward an optimal solution. Thus this *primal-dual framework* is not in general particularly attractive. However, combining the primal-dual idea to the one of penalty leads to another class of algorithms called *multiplier methods*. In these methods one uses instead of the classical Lagrangian $l(x, y)$ a 'penalized' Lagrangian of the form:

$$P_c(x, y) = f(x) + \sum_{i=1}^{p} y_i h_i(x) + \frac{c}{2} \sum_{i=1}^{p} h_i^2(x),$$

where $c > 0$ is a penalty parameter. Then, starting with an initial multiplier y^k and penalty parameter c^k, the augmented Lagrangian P_c is minimized with respect to x and at the end of each minimization, the multipliers (and sometimes also the penalty parameter) are updated according to some scheme and we continue the process until convergence. More precisely, the method of multipliers generates the sequences $\{y^k\} \subset \mathbf{R}^m$, $\{x^k\} \subset \mathbf{R}^n$ as follows. Given a sequence of nondecreasing scalars $c_k > 0$, compute

$$x^{k+1} \in \arg\min\left\{L_{c_k}(x, y^k):\ x \in \mathbb{R}^n\right\},$$
$$y_i^{k+1} = y_i^k + c_k h_i(x^{k+1}), \quad i = 1, \ldots, p.$$

The rational behind the updating of the multipliers y^k is that if the generated sequence x^k converges to a local minimum then the sequence y^k will converge to the corresponding Lagrange multiplier y^*. Under reasonable assumptions, this happens without increasing the parameter c^k to infinity and thus avoids the difficulty with ill-conditioning. The above scheme provides with the key steps in devising a multiplier method for equality constrained optimization problems. We now turn to the case of problems with inequality constraints:

$$\text{(I)} \quad \min\{f(x):\ g_i(x) \le 0,\ i = 1, \ldots, m\}.$$

One simple way to treat this case is to transform the inequality constraints to equality using squared variables and then apply the multiplier framework previously outlined. Thus, we convert problem (I) to the equality constrained problem in the variables (x, z):

$$\begin{cases} \min & f(x) \\ \text{s.t.} & g_i(x) + z_i^2 = 0, \quad i = 1, \ldots, m, \end{cases}$$

where $z \in \mathbf{R}^m$ are additional variables. The quadratic augmented Lagrangian to be minimized with respect to

(x, z) thus takes the form:

$$Q_c(x, z, y) = f(x) + \sum_{i=1}^{m} y_i(g_i(x) + z_i^2) + \frac{c}{2}\sum_{i=1}^{m}(g_i(x) + z_i^2)^2.$$

The key observation here is that the minimization with respect to z can be carried out analytically. One can verify via simple calculus that for fixed (x, y), $\min_{z\in\mathbb{R}^m} Q_c(x, z, y) = L_c(x, y)$, with

$$L_c(x, y) = f(x) + \frac{1}{2c}\sum_{i=1}^{m}\left[\max^2\{0, y_i + cg_i(x)\} - y_i^2\right].$$

Summarizing, the multiplier method for the inequality constrained problem (I) consists of the following two steps:

$$x^{k+1} \in \arg\min\left\{L_{c_k}(x, y^k):\ x \in \mathbb{R}^n\right\},$$
$$y^{k+1} = \max\{0, y^k + c_k g(x^{k+1})\}.$$

For the general optimization problem (O), namely the case of mixed equality and inequality constraints, Lagrangian multiplier methods can be developed in a similar fashion. Convergence results to a local minimum for the above schemes can be established under second order sufficiency assumptions, ([7,28]). In the case of convex programs, namely when in problem (I) the functions f, $g_1, \ldots, g_m$ are assumed convex functions, (or more generally in problem (O), if we also assume h_i affine and X convex), much stronger convergence results can be established under mild assumptions ([29]). A typical result is as follows.

Assumption 1 The set of optimal solutions of the convex problem (I) is nonempty and compact and the set of multiplier is nonempty and compact.

The assumption on the optimal set of multipliers is guaranteed under the standard *Slater constraint qualification*:

$$\exists\hat{x}:\ g_i(\hat{x}) \le 0, \quad i = 1, \ldots, m.$$

Under assumption 1, one can prove that the sequence y^k converges to some Lagrange multiplier y^* and any limit point of the sequence x^k is an optimal solution of the convex program. Note that we do not require that c_k is sufficiently large and convergence is obtained from any starting point $y^0 \in \mathbf{R}^m$.

The multiplier method for inequality constrained problems was derived by using slack variables in the inequality constraints and then by applying the multiplier method which was originally devised for problems having only equality constraints. An alternative way of constructing an augmented Lagrangian method is via the *proximal framework*.

Proximal Minimization

Consider the convex optimization problem

$$\text{(C)} \quad \min\{F(x):\ x \in \mathbb{R}^n\},$$

where F: $\mathbf{R}^n \to (-\infty, +\infty]$ is a proper, lower semicontinuous convex function. One method to solve (C) is to 'regularize' the objective function using the *proximal map* of J.-J. Moreau [22]. Given a real positive number c, a *proximal approximation* of f is defined by:

$$F_c(x) = \inf_u\left\{F(u) + (2c)^{-1}\|x - u\|^2\right\}. \tag{1}$$

The resulting function F_c enjoys several important properties: it is convex and differentiable with gradient which is Lipschitz with constant (c^{-1}) and when minimized possesses the same set of minimizers and the same optimal value than problem (C). The quadratic regularization process of the function f leads to an iterative procedure for solving problem (C), called the *proximal point algorithm* [21,30]. The method is as follows: given an initial point $x_0 \in \mathbf{R}^n$ a sequence $\{x_k\}$ is generated by solving:

$$x^{k+1} = \arg\min\left\{F(x) + \frac{1}{2c_k}\left\|x - x^k\right\|^2\right\}, \tag{2}$$

where $\{c_k\}_{k=1}^{\infty}$ is a sequence of positive numbers.

One of the most powerful application of the proximal algorithm is when applied to the dual of an optimization problem. Indeed, as shown by Rockafellar [27,29], a direct calculation shows that L_c can be written as

$$L_c(x, y) = \max_{\lambda\in\mathbb{R}_+^m}\left\{l(x, \lambda) - \frac{1}{2c}\|\lambda - y\|^2\right\}, \tag{3}$$

where the maximum is attained uniquely at $\lambda_i = \max\{0, y_i + c\, g_i(x)\}$, $i = 1, \ldots, m$. Here l: $\mathbf{R}^n \times \mathbf{R}^m_+ \to \mathbf{R}$ denotes the usual Lagrangian associated with the inequality constrained problem (I) and $\mathbf{R}^m_+$ stands for the nonnegative orthant. This shows that the quadratic augmented Lagrangian is nothing else but the Moreau proximal regularization of the ordinary Lagrangian, and the quadratic multiplier method can be interpreted as applying the proximal minimization algorithm on the dual problem associated with (I):

$$\text{(D)} \quad \sup\{d(y):\ y \geq 0\},$$

where $d(y) := \inf_x l(x, y)$ is the dual functional. This interplay between the proximal algorithm and multiplier methods is particularly interesting since it offers the possibility of designing and analyzing the convergence properties of the later from the former, and also leads to consider useful potential extensions of multiplier methods which are discussed next.

Modified Lagrangians

One of the main disadvantages of the quadratic multiplier methods for inequality constrained problems is that even when the original problem is given twice continuously differentiable, the corresponding functional L_c is not. Indeed, note that with twice continuously differentiable data $\{f, g_i\}$, the augmented Lagrangian L_c is continuously differentiable in x. However, the Hessian matrix of L_c is discontinuous for all x such that $g_i(x) = -c^{-1} y_i$. This may cause difficulties in designing an efficient unconstrained minimization algorithm for L_c and motivates the search for alternative augmented Lagrangian to handle inequality constrained problems, which we call here *modified Lagrangians*. These Lagrangians possess better differentiability properties to allow the use of efficient Newton-like methods in the minimization step. Modified Lagrangians can be found in several works, [1,15,19,20]. An approach originally developed in [19] proposed a class of methods which uses instead of L_c a modified Lagrangian of the form:

$$B_c(x, y) := f(x) + c^{-1} \sum_{i=1}^{m} y_i \psi(c g_i(x)),$$

where ψ is a scalar penalty function which is at least C^2 and satisfies some other technical conditions. For each choice of ψ we then have a multiplier method which consists of the sequence of unconstrained minimization problems

$$x^{k+1} \in \arg\min_{x \in \mathbb{R}^n} B_{c^k}(x, y^k),$$

followed by the multiplier updates

$$y_i^{k+1} = y_i^k \psi'(c_k g_i(x^{k+1})), \quad i = 1, \ldots, m.$$

The multiplier updating formula can be simply explained as follows. Suppose the functions in problem (I) are given differentiable, then x^{k+1} minimizes $B_{c_k}(x, y^k)$ means that $\nabla_x B_{c_k}(x^{k+1}, y^k) = 0$, i. e.,

$$\nabla f(x^{k+1}) + \sum_{i=1}^{m} y_i^k \psi'(c_k g_i(x^{k+1})) \nabla g_i(x^{k+1}) = 0,$$

and using the multiplier updates defined above the equation reduces to:

$$\nabla f(x^{k+1}) + \sum_{i=1}^{m} y_i^{k+1} \nabla g_i(x^{k+1}) = 0,$$

showing that (x^{k+1}, y^{k+1}) also satisfies the optimality conditions for minimizing the classical Lagrangian, namely $\nabla_x l(x^{k+1}, y^{k+1}) = 0$. Interesting special cases of the generic method described above includes the exponential method ([23,35]) with the choice $\psi(t) = e^t - 1$ and the modified barrier method [24] which is based on the choice $\psi(t) = -\ln(1 - t)$. More examples and further analysis of these methods can be found in [25].

Another way of constructing modified Lagrangians is in view of the results from the previous section, to try alternative proximal regularization terms which could lead to better differentiability properties of the corresponding augmented Lagrangian functional. This approach was considered in [32], who suggested new classes of proximal approximation of a function given by

$$F_\lambda(x) := \inf_u \{f(u) + \lambda^{-1} D(u, x)\}. \tag{4}$$

Here, $D(\cdot, \cdot)$, which replaces the quadratic proximal term in (1), is a measure of 'closeness' between x, y satisfying $D(x, y) \geq 0$ with equality if and only if $x = y$. One generic form for D is the use of a 'proximal-like'

term defined by

$$D(x, y) := d_\varphi(x, y) := \sum_{i=1}^{n} y_i \varphi(y_i^{-1} x_i),$$

where φ is a given convex function defined on the nonnegative real line and which satisfies some technical conditions ([33]). The motivation of using such functional emerges from the desire of eliminating nonnegativity constraints such as the ones present in the dual problem. Thus, by mimicking (2) and (3) with the proximal term d_φ, one can design a wide variety of modified Lagrangians methods with an appropriate choice of φ. The basic steps of the modified multipliers method then emerging can be described as follows: Given a sequence of positive numbers $\{c_k\}$, and initial points $x^k \in \mathbf{R}^n$, $y^k \in \mathbf{R}^m_+$ (the positive orthant) generate iteratively the next points by solving

$$x^{k+1} \in \arg\min \left\{ M_{c_k}(x, y^k) : \; x \in \mathbb{R}^n \right\}, \tag{5}$$

followed by the multiplier updates

$$y^{k+1} \in \arg\max_{y \geq 0} \{ y' g(x^{k+1}) - c_k^{-1} d_\varphi(y, y^k) \}, \tag{6}$$

where M_c is the modified Lagrangian defined by

$$M_c(x, y) = \sup_{\mu \in \mathbb{R}^m_+} \{ l(x, \mu) - c^{-1} d_\varphi(\mu, y) \} \tag{7}$$

i. e., the proximal-like regularization of the usual Lagrangian $l(x, \mu)$ associated with problem (I). In the equation (6), $g(x)$ denotes the column vector $(g_1(x), \ldots, g_m(x))' \in \mathbf{R}^m$ and the prime denotes transposition. The method is viable since both (6) and (7) can be solved analytically, and the computational analysis and effort should concentrate on (5). This method of multipliers is nothing else but a proximal-like algorithm applied to the dual problem (D) ([17]) i. e., starting with $y^0 \in \mathbf{R}^m_+$, generate a sequence $\{y^k\}$ by solving

$$y^{k+1} = \arg\max_{y \geq 0} \{ d(y) - c_k^{-1} d_\varphi(y, y^k) \}.$$

The above scheme gives rise to a rich family of numerical methods, which includes (with an appropriate choice of φ) several classes of nonquadratic multiplier methods ([7,24,35]). One of the main advantage of using these modified multiplier methods is that in contrast with the usual quadratic augmented Lagrangian function, the modified Lagrangian for various choices of d_φ is twice continuously differentiable if the problem's data f, g are. Thus, this opens the possibility of using Newton methods for solving efficiently (5).

Under assumption 1 and appropriate condition on the kernel φ one can prove convergence results for these modified multiplier methods similar to the one obtains in the quadratic case ([17]). There has been considerable recent research on modified Lagrangian methods and for further results see [3,4,5,11,18,25].

The Lagrangian functional plays a central role in the analysis and algorithmic development of constrained optimization problems. Lagrangian based methods and the related proximal framework have been used in other optimization contexts, such as convexification of nonconvex optimization problems [6,28], decomposition algorithms [9,12,31,34], semidefinite programming [10] and in many other applications, see e. g., [8,14] where more references can be found.

See also

- Convex Max-functions
- Decomposition Techniques for MILP: Lagrangian Relaxation
- Integer Programming: Lagrangian Relaxation
- Lagrange, Joseph-Louis
- Multi-objective Optimization: Lagrange Duality

References

1. Arrow KJ, Gould FJ, Howe SM (1973) A general saddle point result for constrained optimization. Math Program 5:225–234
2. Arrow KJ, Hurwicz L, Uzawa H (1958) Studies in linear and nonlinear programming. Stanford Univ. Press, Palo Alto, CA
3. Auslender AA, Cominetti R, Haddou M (1997) Asymptotic analysis of penalty and barrier methods in convex and linear programming. Math Oper Res 22:43–62
4. Auslender AA, Teboulle M, Ben-Tiba S (1999) Interior proximal and multiplier methods based on second order homogeneous kernels. Math Oper Res 24:645–668
5. Ben-Tal A, Zibulevsky M (1997) Penalty-barrier methods for convex programming problems. SIAM J Optim 7:347–366
6. Bertsekas D (1979) Convexification procedures and decomposition methods for nonconvex optimization problems. J Optim Th Appl 29:169–197
7. Bertsekas D (1982) Constrained optimization and Lagrangian multipliers. Acad. Press, New York

8. Bertsekas D, Tsitsiklis JN (1989) Parallel and distributed computation: Numerical methods. Prentice-Hall, Englewood Cliffs, NJ
9. Chen G, Teboulle M (1994) A proximal-based decomposition method for convex minimization problems. Math Program 64:81–101
10. Doljanski M, Teboulle M (1998) An interior proximal algorithm and the exponential multiplier method for semidefinite programming. SIAM J Optim 9:1–13
11. Eckstein J (1993) Nonlinear proximal point algorithms using Bregman functions with applications to convex programming. Math Oper Res 18:202–226
12. Eckstein J, Bertsekas. DP (1992) On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math Program 55:293–318
13. Fiacco AV, McCormick GP (1990) Nonlinear programming: Sequential unconstrained minimization techniques. Classics Appl Math. SIAM, Philadelphia
14. Glowinski R, Le Tallec P (1989) Augmented Lagrangians and operator-splitting methods in nonlinear mechanics. Stud Appl Math. SIAM, Philadelphia
15. Golshtein EG, Tretyakov NV (1996) Modified Lagrangians and monotone maps in optimization. Discrete Math and Optim. Wiley, New York
16. Hestenes MR (1969) Multiplier and gradient methods. J Optim Th Appl 4:303–320
17. Iusem A, Teboulle M (1995) Convergence analysis of nonquadratic proximal methods for convex and linear programming. Math Oper Res 20:657–677
18. Kiwiel KC (1997) Proximal minimization methods with generalized Bregman functions. SIAM J Control Optim 35:1142–1168
19. Kort KBW, Bertsekas DP (1972) A new penalty function method for constrained minimization. In: Proc. IEEE Conf. Decison Control, 162–166
20. Mangasarian OL (1975) Unconstrained Lagrangians in nonlinear programming. SIAM J Control 13:772–791
21. Martinet B (1978) Perturbation des méthodes D, optimisation application. RAIRO Anal Numer/Numer Anal 93(12):152–171
22. Moreau JJ (1965) Proximité and dualité dans un espace Hilbertien. Bull Soc Math France 93:273–299
23. Nguyen VH, Strodiot JJ (1979) On the convergence rate of a penalty function method of the exponential type. J Optim Th Appl 27:495–508
24. Polyak RA (1992) Modified barrier functions: Theory and methods. Math Program 54:177–222
25. Polyak RA, Teboulle M (1997) Nonlinear rescaling and proximal-like methods in convex optimization. Math Program 76:265–284
26. Powell MJD (1969) A method for nonlinear constraints in minimization problems. In: Fletcher R (ed) Optimization. Acad. Press, New York, 283–298
27. Rockafellar RT (1973) A dual approach to solving nonlinear programming problems by unconstrained optimization. Math Program 5:354–373
28. Rockafellar RT (1974) Augmented Lagrange multiplier functions and duality in nonconvex programming. SIAM J Control 12:268–285
29. Rockafellar RT (1976) Augmented Lagrangians and applications of the proximal point algorithm in convex programming. Math Oper Res 1:97–116
30. Rockafellar RT (1976) Monotone operators and the proximal point algorithm. SIAM J Control Optim 14:877–898
31. Spingarn JE (1985) Applications of the method of partial inverses to convex programming: Decomposition. Math Program 32:199–223
32. Teboulle M (1992) Entropic proximal mappings in nonlinear programming and applications. Math Oper Res 17:670–690
33. Teboulle M (1997) Convergence of proximal-like algorithms. SIAM J Optim 7:1069–1083
34. Tseng P (1991) Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. SIAM J Control Optim 29:119–138
35. Tseng P, Bertsekas DP (1993) On the convergence of the exponential multiplier method for convex programming. Math Program 60:1–19

Laplace Method and Applications to Optimization Problems

PANOS PARPAS, BERÇ RUSTEM
Department of Computing, Imperial College, London, GB

Article Outline

Abstract
Background
  Heuristic Foundations of the Method
Applications
  Stochastic Methods for Global Optimization
  Phase Transitions in Combinatorial Optimization
  Worst Case Optimization
References

Abstract

The Laplace method has found many applications in the theoretical and applied study of optimization problems. It has been used to study: the asymptotic behavior of stochastic algorithms, 'phase transitions' in combinatorial optimization, and as a smoothing technique

for non–differentiable minimax problems. This article describes the theoretical foundation and practical applications of this useful technique.

Background

Laplace's method is based on an ingenious trick used by Laplace in one his papers [19]. The technique is most frequently used to perform asymptotic evaluations to integrals that depend on a scalar parameter t, as t tends to infinity. Its use can be theoretically justified for integrals in the following form:

$$I(t) = \int_{\mathcal{A}} \exp\left\{\frac{-f(x)}{T(t)}\right\} \mathrm{d}\Lambda(x) .$$

Where $f : \mathbb{R}^n \to \mathbb{R}$, $T : \mathbb{R} \to \mathbb{R}$, are assumed to be smooth, and $T(t) \to 0$ as t tends to ∞. $\mathcal{A}$ is some compact set, and Λ is some measure on $\mathcal{B}$ (the $\sigma-$field generated by $\mathcal{A}$). We know that since $\mathcal{A}$ is compact, the continuous function f will have a global minimum in $\mathcal{A}$. For simplicity, assume that the global minimum x^* is unique, and that it occurs in the interior $\mathcal{A}$. Under these conditions, and as t tends to infinity, only points that are in the immediate neighborhood of x^* contribute to the asymptotic expansion of $I(t)$ for large t. The heuristic argument presented above can be made precise. The complete argument can be found in [2], and in [4]. Instead we give a heuristic but didactic argument that is usually used when introducing the method.

Heuristic Foundations of the Method

For the purpose of this subsection only, assume that f is a function of one variable, and that $\mathcal{A}$ is given by some interval $[a, b]$. It will be instructive to give a justification of the method based on the one dimensional integral:

$$K(t) = \int_a^b \exp\left\{-\frac{f(x)}{t}\right\} \mathrm{d}x .$$

Suppose that f has a unique global minimum, say c, such that $c \in (a, b)$. As t is assumed to be large, we only need to take into account points near c when evaluating $K(t)$. We therefore approximate $K(t)$ by $K(t;\epsilon)$. The latter quantity is given by:

$$K(t;\epsilon) = \int_{c-\epsilon}^{c+\epsilon} \exp\left\{-\frac{f(x)}{t}\right\} \mathrm{d}x .$$

Expanding f to second order, and by noting that $f'(c) = 0$, we obtain the following approximation:

$$\begin{aligned} K(t;\epsilon) &\approx \int_{c-\epsilon}^{c+\epsilon} \exp\left\{-\frac{f(c) + \frac{1}{2}f''(c)(x-c)^2}{t}\right\} \mathrm{d}x \\ &= \exp\left\{-\frac{f(c)}{t}\right\} \int_{c-\epsilon}^{c+\epsilon} \exp\left\{-\frac{f''(c)(x-c)^2}{2t}\right\} \mathrm{d}x . \end{aligned}$$

The limits of the integral above can be extended to infinity. This extension can be justified by the fact only points around c contribute to the asymptotic evaluation of the integral.

$$\begin{aligned} &K(t;\epsilon) \\ &\approx \exp\left\{-\frac{f(c)}{t}\right\} \int_{-\infty}^{+\infty} \exp\left\{-\frac{f''(c)(x-c)^2}{2t}\right\} \mathrm{d}x \\ &= \exp\left\{-\frac{f(c)}{t}\right\} \sqrt{\frac{2\pi t}{f''(c)}} . \end{aligned}$$

In conclusion we have that:

$$\lim_{t\to\infty} K(t) = \exp\left\{-\frac{f(c)}{t}\right\} \sqrt{\frac{2\pi t}{f''(c)}} .$$

Rigorous justifications of the above arguments can be found in [4]. These types of results are standard in the field of asymptotic analysis. The same ideas can be applied to optimization problems.

Applications

Consider the following problem:

$$\begin{aligned} F^* &= \min f(x) \\ &\quad \text{s.t } g_i(x) \le 0 \qquad i = 1, \ldots, l . \end{aligned} \tag{1}$$

Let S denote the feasible region of the problem above, and assume that it is nonempty, and compact, then:

$$\lim_{t\downarrow 0} -\epsilon \ln c(t) = F^* . \tag{2}$$

Where,

$$\begin{aligned} c(t) &\triangleq \int_S \exp\left\{\frac{-f(x)}{t}\right\} \mathrm{d}\Lambda \\ &= \int_{\mathbb{R}^n} \exp\left\{\frac{-f(x)}{t}\right\} I_x(S)\mathrm{d}\Lambda . \end{aligned} \tag{3}$$

Λ is any measure on $(\mathbb{R}^n, \mathcal{B})$. A proof of Eq. (2) can be found in [16].

The relationship in Eq. (3) can be evaluated using the Laplace method. The link between the Laplace method and optimization has been explored in:

- Stochastic methods for global optimization.
- Phase transitions in combinatorial optimization.
- Algorithms for worst case analysis.

These application areas will be explored next.

Stochastic Methods for Global Optimization

Global optimization is concerned with the computation of global solutions of Eq. (1). In other words, one seeks to compute F^*, and if possible obtaining points from the following set:

$$S^* = \{x \in S \mid f(x) = F^*\} \ .$$

Often the only way to solve such problems is by using a stochastic method. Deterministic methods are also available but are usually applicable to low dimensional problems. When designing stochastic methods for global optimization, it is often the case that the algorithm can be analyzed as a stochastic process. Then in order to analyze the behavior of the algorithm we can examine the asymptotic behavior of the stochastic process. In order to perform this analysis we need to define a probability measure that has its support in S^*. This strategy has been implemented in [3,6,7,8,9,10,16].

A well known method for obtaining a solution to an unconstrained optimization problem is to consider the following Ordinary Differential Equation (ODE):

$$\mathrm{d}X(t) = -\nabla f(X(t))\mathrm{d}t \ . \tag{4}$$

By studying the behavior of $X(t)$ for large t, it can be shown that $X(t)$ will eventually converge to a stationary point of the unconstrained problem. A review of, so called, continuous-path methods can be found in [22]. More recently, application of this method to large scale problems was considered by Li-Zhi et al. [13]. A deficiency of using Eq. (4) to solve optimization problems is that it will get trapped in local minima. In order to allow the trajectory to escape from local minima, it has been proposed by various authors (e. g. [1,3,7,8,12,16]) to add a stochastic term that would allow the trajectory to "climb" hills. One possible augmentation to Eq. (4) that would enable us to escape from local minima is to add noise. One then considers the *diffusion process*:

$$\mathrm{d}X(t) = -\nabla f(X(t))\mathrm{d}t + \sqrt{2T(t)}\mathrm{d}B(t) \ . \tag{5}$$

Where $B(t)$ is the standard Brownian motion in $\mathbb{R}^n$. It has been shown in [3,7,8], under appropriate conditions on f, that if the *annealing schedule* is chosen as follows:

$$T(t) \triangleq \frac{c}{\log(2+t)} \ , \quad \text{for some} \quad c \geq c_0 \ , \tag{6}$$

where c_0 is a constant positive scalar (the exact value of c_0 is problem dependent). Under these conditions, as $t \to \infty$, the transition probability of $X(t)$ converges (weakly) to a probability measure Π. The latter, has its support on the set of global minimizers. A characterization of Π was given by Hwang in [11]. It was shown that Π is the weak limit of the following, so called, *Boltzmann density*:

$$p(t,x) = \left[\exp\left\{-\frac{f(x)}{T(t)}\right\}\right]\left[\int_{\mathbb{R}^n} \exp\left\{-\frac{f(x)}{T(t)}\right\} \mathrm{d}x\right]^{-1} . \tag{7}$$

Discussion of the conditions for the existence of Π, can be found in [11]. A description of Π in terms of the Hessian of f can also be found in [11]. Extensions of these results to constrained optimization problems appear in [16].

Phase Transitions in Combinatorial Optimization

The aim in combinatorial optimization is to select from a finite set of configurations of the system, the one that minimizes an objective function. The most famous combinatorial problem is the Travelling Salesman Problem (TSP). A large part of theoretical computer science is concerned with estimating the complexity of combinatorial problems. Loosely speaking, the aim of computational complexity theory is to classify problems in terms of their degree of difficulty. One measure of complexity is time complexity, and worst case time complexity has been the aspect that received most attention. We refer the interested reader to [15] for results in this direction. We will briefly summarize results that have to do with average time complexity, the Laplace method, and phase transitions.

Most of complexity theory is concerned with worst case complexity. However, many useful methods (e. g. the simplex method) will require an exponential amount of time to converge only in pathological cases. It is therefore of great interest to estimate average case complexity. The physics community has recently proposed the use of tools from statistical mechanics as one way of estimating average case complexity. A review in the form of a tutorial can be found in [14]. Here we just briefly adumbrate the main ideas.

The first step in the statistical mechanics approach is to define a probability measure on the configuration of the system. This definition is done with the Boltzmann density:

$$p_t(C) = \frac{\exp\left\{-\frac{1}{t}f(C)\right\}}{\sum_C \exp\left\{-\frac{1}{t}f(C)\right\}} .$$

The preceding equation is of course the discrete version of Eq. (7). Using the above definition, the average value of the objective function is given by:

$$\langle f_t \rangle = \sum_C p_t(C) f(C) .$$

Tools and techniques of statistical mechanics can be used to calculate 'computational phase transitions'. A computational phase transition is an abrupt change in the computational effort required to solve a combinatorial optimization problem. It is beyond the scope of this article to elaborate on this interesting area of optimization. We refer the interested reader to the review in [14]. The book of Talagrand [20] presents some rigorous results on this subject.

Worst Case Optimization

In many areas where optimization methods can be fruitfully applied, worst case analysis can provide considerable insight into the decision process. The fundamental tool for worst case analysis is the continuous minimax problem:

$$\min_{x \in X} \Phi(x) ,$$

where $\Phi(x) = \max_{y \in Y} f(x, y)$. The continuous minimax problem arises in numerous disciplines, including n–person games, finance, economics and policy optimization (see [18] for a review). In general, they are used by the decision maker to assess the worst-case strategy of the opponent and compute the optimal response. The opponent can also be interpreted as nature choosing the worst-case value of the uncertainty, and the solution would be the strategy which ensures the optimal response to the worst–case. Neither the robust decision maker nor the opponent would benefit by deviating unilaterally from this strategy. The solution can be characterized as a saddle point when $f(x, \cdot)$ is convex in x and $f(\cdot, y)$ is concave in y. A survey of algorithms for computing saddle points can be found in [5,18].

Evaluating $\Phi(x)$ is extremely difficult due to the fact that global optimization is required over Y. Moreover, this function will in general be non-differentiable. For this reason, it has been suggested by many researchers (e. g. [17,21]) to approximate $\Phi(x)$ with $\Phi(x; t)$ given by:

$$\Phi(x; t) = \int_Y \exp\left\{-\frac{f(x, y)}{t}\right\} \mathrm{d}y .$$

This is of course another application of the Laplace method, and it can easily be seen that:

$$\lim_{t \downarrow 0} -t \ln \Phi(x; t) = \Phi(x) .$$

This idea has been implemented in [17,21] with considerable success.

References

1. Aluffi-Pentini F, Parisi V, Zirilli F (1985) Global optimization and stochastic differential equations. J Optim Theory Appl 47(1):1–16
2. Bender CM, Orszag SA (1999) Advanced mathematical methods for scientists and engineers I. Asymptotic methods and perturbation theory, Reprint of the 1978 original. Springer, New York
3. Chiang TS, Hwang CR, Sheu SJ (1987) Diffusion for global optimization in $\mathbf{R}^n$. SIAM J Control Optim 25(3):737–753
4. de Bruijn NG (1981) Asymptotic methods in analysis, 3rd edn. Dover Publications Inc., New York
5. Dem'yanov VF, Malozëmov VN (1990) Introduction to minimax. Translated from the Russian by Louvish D, Reprint of the 1974 edn. Dover Publications Inc., New York
6. Gelfand SB, Mitter SK (1991) Recursive stochastic algorithms for global optimization in $\mathbf{R}^d$. SIAM J Control Optim 29(5):999–1018
7. Geman S, Hwang CR (1986) Diffusions for global optimization. SIAM J Control Optim 24(5):1031–1043
8. Gidas B (1986) The Langevin equation as a global minimization algorithm. In: Disordered systems and biological organization (Les Houches 1985). NATO Adv Sci Inst Ser F Comput Systems Sci, vol 20. Springer, Berlin, pp 321–326

9. Gidas B (1987) Simulations and global optimization. In: Random media (Minneapolis, MN, 1985), IMA Vol Math Appl, vol 7. Springer, New York, pp 129–145
10. Gidas B (1985) Metropolis-type Monte Carlo simulation algorithms and simulated annealing. In: Topics in contemporary probability and its applications. Probab Stochastics Ser. CRC, Boca Raton, FL, pp 159–232
11. Hwang CR (1980) Laplace's method revisited: weak convergence of probability measures. Ann Probab 8(6):1177–1182
12. Kushner HJ (1987) Asymptotic global behavior for stochastic approximation and diffusions with slowly decreasing noise effects: global minimization via Monte Carlo. SIAM J Appl Math 47(1):169–185
13. Li-Zhi L, Liqun Q, Hon WT (2005) A gradient-based continuous method for large-scale optimization problems. J Glob Optim 31(2):271
14. Martin OC, Monasson R, Zecchina R (2001) Statistical mechanics methods and phase transitions in optimization problems. Theoret Comput Sci 265(1–2):3–67
15. Papadimitriou CH (1994) Computational complexity. Addison-Wesley, Reading, MA
16. Parpas P, Rustem B, Pistikopoulos E (2006) Linearly constrained global optimization and stochastic differential equations. J Glob Optim 36(2):191–217
17. Polak E, Royset JO, Womersley RS (2003) Algorithms with adaptive smoothing for finite minimax problems. J Optim Theory Appl 119(3):459–484
18. Rustem B, Howe M (2002) Algorithms for worst-case design and applications to risk management. Princeton University Press, Princeton, NJ
19. Stigler SM (1986) Laplace's 1774 memoir on inverse probability. Statist Sci 1(3):359–378
20. Talagrand M (2003) Spin glasses: a challenge for mathematicians. Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge (Results in Mathematics and Related Areas. 3rd Series). A Series of Modern. Surveys in Mathematics, vol 46. Springer, Berlin
21. Xu S (2001) Smoothing method for minimax problems. Comput Optim Appl 20(3):267–279
22. Zirilli F (1982) The use of ordinary differential equations in the solution of nonlinear systems of equations. In: Nonlinear optimization (Cambridge 1981). NATO Conf Ser II: Systems Sci. Academic Press, London, pp 39–46

Large Scale Trust Region Problems

LSTR

LAURA PALAGI
DIS, Universitá Roma 'La Sapienza', Rome, Italy

MSC2000: 90C30

Article Outline

Keywords
Algorithms Based on Successive Improvement of KKT Points
  Exact Penalty Function Based Algorithm (EPA)
  D.C. Decomposition Based Algorithm (DCA)
Parametric Eigenvalue Reformulation Based Algorithms
  Inverse Interpolation Parametric Eigenvalue Formulation (IPE)
  Semidefinite Programming Approach (SDP)
Conclusion
See also
References

Keywords

Large scale trust region problem; Exact penalty function; D.C. programming; Eigenvalue problem; Semidefinite programming

The *trust region* (TR) problem consists in minimizing a general quadratic function q: $\mathbf{R}^n \to \mathbf{R}$ of the type

$$q(x) = \frac{1}{2} x^\top Q x + c^\top x$$

subject to an ellipsoidal constraint $x^\intercal Hx \le r^2$ with the symmetric matrix H positive definite and r a positive scalar. By rescaling and without loss of generality, it can be assumed for sake of simplicity $H = I$, hence the TR problem is

$$\begin{cases} \min & q(x) \\ \text{s.t.} & \|x\|^2 \le r^2, \end{cases} \tag{1}$$

where $\|\cdot\|$ denotes the ℓ_2 norm.

The interest in this problem initially arose in the context of unconstrained optimization when $q(x)$ is a local quadratic model of the objective function which is 'trusted' to be valid over a restricted ellipsoidal region centered around the current iterate. However, it has been shown later that problems with the same structure of (1) are at the basis of algorithms for solving general constrained nonlinear programming problems (e. g. [2,14,19,21,27,28] and references therein), and for obtaining bounds for integer programming problems

(e. g. [10,11,12,17,18,26]; cf. also ► Integer programming).

Many papers have been devoted to study the specific features of Problem (1). It is well known [7,22] that a feasible point x^* is a global solution for (1) if and only if there exists a scalar $\lambda^* \geq 0$ such that the following KKT conditions are satisfied:

$$(Q + \lambda^* I)x^* = -c,$$
$$\lambda^*(\|x^*\|^2 - r^2) = 0,$$

and furthermore $Q + \lambda^* I \succcurlyeq 0$, where $\succcurlyeq$ denotes positive semidefinitness of the matrix.

Note that a complete characterization of global minimizers is given without requiring any convexity assumption on the matrix Q. Moreover, it has been proved that an approximation to the global solution can be computed in polynomial time (see, for example, [1,24,25]). Hence Problem (1) can be considered an 'easy' problem from a theoretical point of view. These peculiarities led to the development of 'ad hoc' algorithms for finding a global solution of Problem (1). The first ones proposed in [7,16,22] were essentially based on the solution of a sequence of linear system of the type $(Q + \lambda_k I)\, x = -\, c$ for a sequence $\{\lambda_k\}$. These algorithms produce an approximate global minimizer of Problem (1), but rely on the ability to compute a Cholesky factorization of the matrix $(Q + \lambda_k I)$ at each iteration k, and hence these methods are appropriate when forming a factorization for different values of λ_k is realistic in terms of both memory and time requirements. Indeed, they are appropriate for large scale problems with special structure, but in the general case, when no sparsity pattern is known, one cannot rely on factorizations of the matrices involved.

Thus one concentrates on iterative methods of conjugate gradient type (cf. ► Conjugate-gradient methods) that require only matrix-vector products. Among the methods that have been proposed to solve *large scale trust region* problems, the following two main categories can be identified:

- methods that produce a sequence of KKT points of (1) with progressive improvement of the objective function;
- methods that solve (1) via a sequence of parametric eigenvalue problems.

Algorithms Based on Successive Improvement of KKT Points

Methods in this class are based on special properties of KKT points of Problem (1). Indeed one can prove the following properties:

1) given a KKT point that is not a global minimizer, it is possible to find a new feasible point with a lower value of the objective function [5,13];
2) the number of distinct values of the objective function $q(x)$ at KKT points is bounded from above by $2m + 2$ where m is the number of negative eigenvalues of Q [13].

Exploiting these properties, a global minimizer of Problem (1) can be found, by applying a finite number of times an algorithm that, starting from a feasible point, locates a KKT point with a lower value of the objective function.

An algorithmic scheme of methods in this framework is summarized in the pseudocode of Table 1. The procedure described above is well-posed in the sense that it enters the 'DO cycle' a finite number of steps, since by Property 2, the function can assume at most a finite number of values at a KKT point.

To complete the scheme of Table 1 and obtain an efficient algorithm for the solution of Problem (1), it remains to specify how to move from a non global KKT

Large Scale Trust Region Problems, Table 1
A pseudocode for TR problem based on successive improvement of KKT points

```
procedure TR-IMPROVE-KKT()
   input instance (Q, c, r, x^0);
   Set k = 0; x = x^k; (starting point)
   find a KKT point x̂^k s.t. q(x̂^k) ≤ q(x^k);
   DO (until a global minimizer is found)
     (escape from a nonglobal KKT point)
       find x s.t. ‖ x ‖≤ r, q(x) < q(x̂^k);
     (update starting point)
       set k = k + 1, x^k = x;
     (find a 'better' KKT point)
        find a KKT point x̂^k s.t.
        q(x̂^k) ≤ q(x^k);
   OD;
   RETURN (solution)
END TR-IMPROVE-KKT;
```

point to a feasible point while improving the objective function, and how to define a globally and 'fast' convergent algorithm to locate a KKT point.

To check global optimality of a KKT point (i. e. to check if $Q + \lambda I \succcurlyeq 0$), one needs an estimate of the KKT multiplier λ corresponding to the point x, and has to verify whether $\lambda \geq -\lambda_{\min}(Q)$. To obtain λ the following multiplier function can be used

$$\lambda(x) = -\frac{1}{2r^2} x^\top (Qx + c), \tag{2}$$

which is consistent, namely at a KKT point $\lambda(x) = \lambda$. If $\lambda < -\lambda_{\min}(Q)$, then (x, λ) is a nonglobal KKT point and a negative curvature direction for the matrix $Q + \lambda I$ exists, namely a vector z such that $z^\intercal (Q + \lambda I)\, z < 0$. To perform the step 'escape from a non global KKT point', one can use such a direction. Roughly speaking and without discussing the details (see [5,13]), a new feasible point can be obtained by moving from x along z itself or along a direction easily obtainable from z of a computable quantity α. The efficiency of this step depends on the ability of finding efficiently such a vector z. Hence a procedure that finds an approximation of the minimum eigenvalue of $(Q + \lambda I)$ and of the corresponding eigenvector is needed. In the large scale setting, this can be done efficiently by using a Lanczos method [3,23] which meets the requirement of limited storage and needs only matrix-vector products.

In the algorithmic scheme of Table 1, it remains to define how to find efficiently a KKT point for Problem (1). Two different approaches have been recently (1998) proposed to perform this step; one is based on a continuously differentiable exact penalty function approach, the other is based on a difference of convex function approach. In both cases, the basic idea is to reformulate the constrained Problem (1) in a different form that allows one to use ideas typical of other fields of mathematical programming. Both approaches, which are described briefly in the sequel, treat indifferently the so called 'easy and hard' cases of Problem (1) and require only matrix vector products.

Exact Penalty Function Based Algorithm (EPA)

The main idea at the basis of a *continuously differentiable exact penalty function approach* is the reformulation of the constrained Problem (1) as an unconstrained one. In particular, a continuously differentiable function $P(x)$ can be defined [13] such that Problem (1) is 'equivalent' to the unconstrained problem

$$\min_{x \in \mathbb{R}^n} P(x).$$

The merit function takes full advantage of the structure of Problem (1) and it is a piecewise quartic function, whose definition relies on the particular multiplier function (2). The analytic expression of P is

$$P(x) = q(x) - \frac{\varepsilon}{4}\lambda(x)^2 + \frac{\varepsilon}{4} \max\left(0, \frac{2}{\varepsilon}(\|x\|^2 - r^2) + \lambda(x)\right)^2,$$

where $0 < \varepsilon < 2r^4/[r^2(\| Q \| + 1) + \|c\|^2]$. The function $P(x)$ has the following features:

- it has compact level sets;
- stationary (global minimum) points of $P(x)$ are KKT (global minimum) points of Problem (1) and vice versa; moreover $P(x) = q(x)$ at these points;
- the penalty parameter ε need not be updated;
- for points such that $\|x\|^2 \leq r^2$ it results $P(x) \leq q(x)$;
- $P(x)$ is twice continuously differentiable in a neighborhood of a KKT point that satisfies strict complementarity.

The unconstrained reformulation of Problem (1) can be exploited to define an algorithm for finding a KKT point while improving the value of objective function with respect to the initial one. Indeed any unconstrained method for the minimization of $P(x)$ can be used. Starting from a point x_0, any of these algorithms produce a sequence of the type

$$x^{k+1} = x^k + \alpha^k d^k, \tag{3}$$

where d^k is a suitable direction, α^k is a stepsize along d^k. The sequence $\{x^k\}$ need not to be feasible for Problem (1). The boundedness of the level sets of $P(x)$ guarantees the boundedness of the iterates and that any convergent unconstrained method obtains a stationary point $\overline{x}$ for P such that $P(\overline{x}) < P(x_0)$. Furthermore a stationary point of $P(x)$ is a KKT point of Problem (1) and $P(\overline{x}) = q(\overline{x})$. If, in addition, x_0 is a feasible point, the following relation holds:

$$q(\overline{x}) = P(\overline{x}) < P(x_0) \leq q(x_0),$$

which means that $\overline{x}$ is a KKT point of Problem (1) with a value of the objective function lower than the value at the starting point.

As regard the efficiency of the algorithms, in terms of rate of convergence and computational requirement, a 'good' direction d^k can be defined, by further exploiting the features of the unconstrained reformulation. Indeed, in a neighborhood of points satisfying the strict complementarity assumption, $P(x) \in C^2$ and therefore any unconstrained truncated Newton algorithm [4] can be easily adapted in order to define globally convergent methods which show a superlinear rate of convergence. Methods in this class include conjugate gradient based iterative method that requires only matrix-vector products and hence are suitable for large scale instances.

The resulting algorithmic scheme is reported in Table 2.

In the nonconvex case ($Q \not\geq 0$) strict complementarity holds in a neighborhood of every global minimizer of Problem (1) [13]. However, this may not be true in a neighborhood of a KKT point and the function $P(x)$ may be not twice differentiable there. Nevertheless algorithms which exhibit superlinear rate of convergence can be defined. In fact, drawing inspiration from the results in [6], the direction d^k is defined as the approximate solution of one of the following linear systems:

$$\begin{cases} \text{if} \quad \|x^k\|^2 - r^2 < -\varepsilon\frac{\lambda^k}{2}, \text{then} \\ \quad (Q + \lambda^k I)d^k = -(Qx^k + c), \\ \text{if} \quad \|x^k\|^2 - r^2 \geq -\varepsilon\frac{\lambda^k}{2}, \text{then} \\ \quad \begin{pmatrix} Q + \lambda^k I & x^k \\ (x^k)^\top & 0 \end{pmatrix} \begin{pmatrix} d^k \\ z^k \end{pmatrix} = \begin{pmatrix} -Qx^k - c \\ r^2 - \|x^k\|^2 \end{pmatrix}. \end{cases} \tag{4}$$

The solution of the linear systems (4), can be determined approximately by using the truncated Newton method proposed in [8]. The direction d^k satisfies suitable descent conditions with respect to the penalty function P, which can be used to measure the progressive improvement of the iterate. The stepsize α^k can be determined by any Armijo-type line search [9] that uses P as merit function.

It is possible to prove that the sequence $\{x^k\}$ produced by (3) with d^k obtained by (4) and $\{\lambda(x^k)\}$ by (2) converges to a KKT point $(\widehat{x}, \widehat{\lambda})$. Moreover if the KKT point $(\widehat{x}, \widehat{\lambda})$ satisfies $z^\top(Q + \widehat{\lambda}I)z > 0$ for all z: $z^\top\widehat{x} = 0$ whenever $\|\widehat{x}\|^2 = r^2$ and $\widehat{\lambda} > 0$, then there exists a neighborhood of $\widehat{x}$ where the rate of convergence of the algorithm is superlinear.

Large Scale Trust Region Problems, Table 2
A pseudocode for finding a KKT point by EPA

```
procedure KKT point by EPA()
   Given x^0 :|| x^0 ||^2 ≤ r^2 and ε > 0;
   set λ^0 = λ(x^0) and k = 0;
   DO (until a KKT point (x^k, λ^k) is found)
      set x^{k+1} = x^k + α^k d^k
      and λ^{k+1} = λ(x^{k+1});
      k = k + 1;
   OD;
   RETURN(KKT point);
END KKT point by EPA;
```

D.C. Decomposition Based Algorithm (DCA)

This algorithm is based on an appropriate reformulation of Problem (1) as the minimization of the difference of convex functions [5]. DCA has been proposed for solving large scale d.c. programming problems. The key aspect in d.c. optimization (cf. ► D.C. programming) relies on the particular structure of the objective function to be minimized on $\mathbf{R}^n$ that is expressed as $f(x) = g(x) - h(x)$, with g and h being convex. One uses the tools of convex analysis applied to the two components g and h of the d.c. function. In particular d.c. duality plays a fundamental role to understand how DCA works. Indeed for a generic d.c. problem, DCA constructs two sequences $\{x^k\}$ and $\{y^k\}$ and it can be viewed as a sort of decomposition approach of the primal and dual d.c. problems. It must be pointed out that a d.c. function has infinitely many d.c. decompositions that give rise to different primal dual pairs of d.c. problems and so to different DCA relative to these d.c. decompositions. Thus, choosing a d.c. decomposition may have an important influence on the qualities (such as robustness, stability, rate of convergence) of the DCA. This aspect is related to regularization techniques in d.c. programming.

In the special case of Problem (1), a quite appropriate d.c. decomposition has been proposed, so that DCA becomes very simple and it requires only matrix-vector products. To apply DCA to Problem (1), a *d.c. decomposition* of the objective function $f(x) = q(x) + \chi_F(x)$

must be defined, where $\chi_F(x)$ is the indicator function for the feasible set, namely

$$\chi_F(x) = \begin{cases} 0 & \text{if } \|x\|^2 \le r^2, \\ \infty & \text{otherwise.} \end{cases}$$

From the computational point of view, the most efficient decomposition that has been proposed is

$$g(x) = \frac{1}{2}\rho\|x\|^2 + c^\top x + \chi_F(x),$$
$$h(x) = \frac{1}{2}x^\top(\rho I - Q)x,$$

with $\rho > 0$ and such that $(\rho\, I - Q) \succcurlyeq 0$. In this case the sequence $\{y^k\}$ is obtained by the following rule $y^k = (\rho I - Q)\, x^k$ and x^{k+1} is obtained as the solution of the problem

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\rho\|x\|^2 + x^\top(c - y^k) + \chi_F(x).$$

Thus x^{k+1} is the projection of $(y^k - c)/\rho$ onto the feasible region $\|x\|^2 \le r^2$. The scheme for obtaining KKT points by DCA is reported in Table 3.

It has been proved [5] that algorithm DCA generates a sequence of feasible points $\{x^k\}$ with strictly decreasing value of the objective function and such that $\{x^k\}$ converges to a KKT point.

In practice the convergence rate depends on the choice of the parameter ρ. A possible choice (the best one according to some numerical experimentations performed in [5]) consists in taking ρ as close as possible to the largest eigenvalue of the matrix Q, namely $\rho = \max\{\lambda_{\max}(Q) + \varepsilon, 10^{-3}\}$ with $\varepsilon > 0$ and sufficiently small. Actually only a low accuracy estimate of $\lambda_{\max}(Q)$, which can be found by using a Lanczos method, is needed.

Large Scale Trust Region Problems, Table 3
A pseudocode for finding a KKT point by DCA

```
procedure KKT POINT by DCA()
  Given x^0, ρ > 0 such that (ρI − Q) ⪰ 0;
  DO (until a KKT point is found)
    IF ‖ (ρI − Q)x^k − c ‖≤ ρr THEN
      x^{k+1} = (1/ρ)[(ρI − Q)x^k − c]
    ELSE x^{k+1} = r ((ρI − Q)x^k − c) / ‖ (ρI − Q)x^k − c ‖
    END IF;
    IF ‖ x^{k+1} − x^k ‖≤ tol exit;
    set k = k + 1;
  OD;
  RETURN (KKT point);
END KKT POINT by DCA;
```

Parametric Eigenvalue Reformulation Based Algorithms

The algorithms in this framework are based on the reformulation of the TR problem into a parametric eigenvalue problem of a bordered matrix. It must be noted that, if the linear term is not present in the function $q(x)$, i. e. $c = 0$, Problem (1) is a pure quadratic problem that corresponds to finding the smallest eigenvalue of the matrix Q. Indeed the intuitive observation behind this idea is that given a real number t, one can write

$$\frac{1}{2}t + q(x) = \frac{1}{2}\begin{pmatrix}1\\ x\end{pmatrix}^\top \begin{pmatrix} t & c^\top \\ c & Q\end{pmatrix}\begin{pmatrix}1\\ x\end{pmatrix}$$

and for a fixed t the goal is to minimize the function $q(x)$ over the set $\{x\colon \|x\|^2 + 1 = r^2 + 1\}$, that is to minimize a pure quadratic form $z^\top D(t)\, z/2$ over a spherical region where

$$D(t) = \begin{pmatrix} t & c^\top \\ c & Q\end{pmatrix}.$$

This suggests that a solution of (1) may be found using eigenpairs of the matrix $D(t)$ where t is a parameter to be adjusted. Indeed, in both the algorithms proposed in this framework a key role is played by eigenpairs of the matrix $D(t)$. At each iteration the main computational step is the calculation of the smallest eigenvalue and a corresponding normalized eigenvector of the parametric matrix $D(t)$. The evaluation of the eigenvalue-eigenvector pair can be done by using Lanczos method as a black box. Therefore methods can exploit sparsity in the matrices and requires only matrix-vector multiplications. Moreover, only one element of the matrix $D(t)$ is changed at each iteration of both the algorithms and so consecutive steps of Lanczos algorithm become cheaper.

Both algorithms have to distinguish between the easy and hard case of Problem (1). The hard case is said to occur when the vector c is orthogonal to the eigenspace associated to the smallest eigenvalue of Q,

i. e. $c^\mathsf{T} y = 0$, for all $y \in S_{\min}$ with

$$S_{\min} = \{x \in \mathbb{R}^n : \; Qx = \lambda_{\min}(Q)x\}\,.$$

Depending on whether the easy or the hard case occurs, eigenpairs of the perturbed matrix $D(t)$ satisfies different properties. In the easy case, the smallest eigenvalue $\mu_{\min}\,(D(t))$ is simple and such that $\mu_{\min}\,(D(t)) < \lambda_{\min}\,(Q)$ for all values t. Moreover in this case the corresponding eigenvector has the first component not equal to zero and this plays a fundamental role in defining the iteration of both the algorithms. In the hard case caution should be used, due to the fact that the first component of the eigenvector corresponding to the smallest eigenvalue of $D(t)$ may be zero. Actually, any vector of the form $(0, y^\mathsf{T})^\mathsf{T}$ with $y \in S_{\min}$ is an eigenvector of the matrix $D(t)$ if and only if $c \perp S_{\min}$.

The two algorithms in this framework are briefly described below. Although the basic idea behind both the algorithms is the same, namely inverse interpolation for a parametric eigenvalue problem, the second one is embedded in a semidefinite programming framework. So the first one is referred to as 'inverse interpolation parametric eigenvalue' (IPE) approach and the second one as 'semidefinite programming approach' (SDP).

Inverse Interpolation Parametric Eigenvalue Formulation (IPE)

In [23] it is observed that if an eigenvector z of $D(t)$ corresponding to a given eigenvalue μ can be normalized so that its first component is one, that is $z = (1, x^\mathsf{T})^\mathsf{T}$, then a solution of the TR problem can be found in terms of eigenpairs of $D(t)$. This corresponds to the easy case and indeed the pair (x, μ) satisfies

$$\begin{pmatrix} t & c^\top \\ c & Q \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} = \mu \begin{pmatrix} 1 \\ x \end{pmatrix},$$

from which we get:

$$\begin{pmatrix} t - \mu = -c^\top x, \\ (Q - \mu I)x = -c. \end{pmatrix}$$

For $\mu < \lambda_{\min}(Q)$, that holds in the easy case with $\mu = \mu_{\min}(D(t))$, the matrix $(Q - \mu\, I)$ is positive definite and hence one can define the function

$$\phi(\mu) = -c^\top x = c^\top (Q - \mu I)^{-1} c,$$

whose derivative is

$$\phi'(\mu) = c^\top (Q - \mu I)^{-2} c = \|x\|^2\,.$$

For a given value of t, finding the smallest eigenvalue $\mu(t) := \mu_{\min}\,(D(t)) < \lambda_{\min}(Q)$ and the corresponding eigenvector of $D(t)$ and then normalizing the eigenvector to have its first component equal to one $(1, x_{\mu(t)}^\mathsf{T})^\mathsf{T}$ will provide a mean to evaluate the function $\phi(\mu)$ and its derivative. If t can be adjusted so that the corresponding $x_{\mu(t)}$ satisfies $\phi'(\mu(t)) = \| x_{\mu(t)} \|^2 = r^2$ with $t - \mu(t) = -\, c^\mathsf{T} x_{\mu(t)}$, and $\mu(t) \le 0$ then $(x, -\mu(t))$ satisfies the optimality conditions for Problem (1). Whereas if, during the course of adjusting t, it happens that $\mu(t) > 0$ with $\| x_{\mu(t)} \|^2 < r^2$ then the optimal solution of Problem (1) is actually unconstrained and can be found by solving the system $Qx = -\, c$ with any iterative method.

Hence using the parametric eigenvalue formulation, the optimal value of (x^*, λ^*) of Problem (1) can be found by solving a sequence of eigenvalue problems adjusting iteratively the parameter t. In order to make this observation useful, a modified Lanczos methods, the *implicit restarted Lanczos method* [23], is used for computing the smallest eigenvalue and the corresponding eigenvector of $D(t)$. Moreover a rapidly convergent iteration to adjust t has been developed, based on a two-point interpolant method. Recalling that the goal is to adjust t so that $\phi(\mu) = t - \mu$ and $\phi'(\mu) = r^2$, an interpolation based iteration that exploits the structure of the problem is proposed. The method is based upon an interpolant $\widehat{\phi}(\mu)$ of $\phi(\mu)$ of the form

$$\widehat{\phi}(\mu) = \frac{\gamma^2}{\alpha - \mu} + \beta(\alpha - \mu) + \delta.$$

The values of the parameters $\alpha, \beta, \gamma, \delta$ appearing in the interpolant function $\widehat{\phi}(\mu)$ are determined using the values of two iterations (x^k, μ^k), (x^{k-1}, μ^{k-1}) according to the following rules. The value δ is chosen so as to provide the current estimate $\delta_{\min}$ of $\lambda_{\min}(Q)$. In particular, if $\| x^k \| < r$ or $\| x^{k-1} \| < r$

$$\delta = \min\left(\delta_{\min}, \frac{(x^k)^\top Q x^k}{\left\|x^k\right\|^2}\right);$$

if $\| x^k \| > r$ and $\| x^{k-1} \| > r$ then

$$\delta = \min\left(\frac{(x^k)^\top Q x^k}{\left\|x^k\right\|^2}, \frac{(x^{k-1})^\top Q x^{k-1}}{\left\|x^{k-1}\right\|^2}\right)$$

Large Scale Trust Region Problems, Table 4
A pseudocode for TR based on (IPE)

```
procedure TR INTERPOL-PARAM-EIG()
  input instace (Q, c, r, x^0);
  (initialization)
  Find λ_min(Q) and its eigenvector x;
  set k = 0, t^k = 0, x^k = x, μ^k = λ_min(Q).
  DO ( until | (|| x^k ||^2 − r^2) / r^2 | ≤ tol )
    construct the interpolar φ̂(μ);
    find μ̂ : μ̂'(μ̂) = r^2, that is:
      μ̂ = α − ( γ^2 / (r^2 + β) )^{1/2};
    set t^{k+1} = μ̂ + φ̂(μ̂), that is:
      t^{k+1} = λ̂ + δ + β(α − μ̂) + γ^2/(α − μ̂);
    compute μ^{k+1} = μ_min(D(t^{k+1}))
      and the corresponding normalized eigenvec-
      tor ( 1, (x^{k+1})^⊤ )^⊤ ;
    set k = k + 1;
  OD;
  RETURN(solution)
END TR INTERPOLATION-PARAM-EIG;
```

and $\delta_{\min} = \min(\delta_{\min}, \delta) \geq \lambda_{\min}(Q)$. The other coefficient are chosen to satisfy $\widehat{\phi}(\mu^k) = -c^\top x^k$, $\widehat{\phi}'(\mu^k) = \|x^k\|^2$, $\widehat{\phi}'(\mu^{k-1}) = \|x^{k-1}\|^2$.

An algorithmic scheme for finding the global minimizer of Problem (1) in the easy case, is reported in Table 4.

It has been proved in [23] that there exists a neighborhood of $-\lambda^*$ such that if μ^0, μ^1 are in this neighborhood, all the sequence $\{\mu^k\}$ is well defined, remains in the neighborhood and converge superlinearly to $-\lambda^*$ with the corresponding iterates x^k converging superlinearly to x^*.

Unfortunately, the iteration described above can break down in the hard case. Indeed the iteration is based on the ability to normalize the eigenvector of the bordered matrix $D(t)$. This is not possible when the first component is equal to zero, that is in the hard case. From the computational point of view, also a near-hard case can be difficult and it is important to detect these cases and to define alternative rules so as to obtain a convergent iteration. This can be done, by using again eigenpairs of the bordered matrix and additional information such as the value of an upper bound λ_U on the optimal value λ^*. When the hard case is detected the new iteration should be used. The convergence of this new iteration can be established but unfortunately the rate of convergence is no longer superlinear.

Semidefinite Programming Approach (SDP)

In [20] a primal-dual simplex type method for Problem (1) has been proposed, which is essentially based on a primal dual pair of semidefinite programming problems. Primal-dual pairs of SDP provide a general framework for TR problem. The idea arises from the fact that Problem (1) enjoys strict *duality*, that is there is no duality gap and

$$q(x^*) = \min_x \max_\lambda L(x, \lambda) = \max_\lambda \min_x L(x, \lambda),$$

where $L(x, \lambda) = q(x) + \lambda(\|x\|^2 - r^2)$ denotes the Lagrangian function. By exploiting this feature it is possible to define a primal-dual pair of linear SDP problems that are strictly connected with the TR problem. In particular, a dual for Problem (1) is

$$\begin{cases} \max & (r^2+1)\mu_{\min}(D(t)) - t, \\ \text{s.t.} & \mu_{\min}(D(t)) \leq 0. \end{cases} \tag{5}$$

The objective function in (5) is a real valued concave function. When the constraint in Problem (1) is an equality one, its dual problem (5) is an unconstrained problem, and as an immediate consequence, the non convex constrained TR problem is transformed into a convex problem and hence it can be solved in polynomial time by the results for general convex programs.

Problem (5) can be easily reformulated as a SDP problem, by introducing an additional variable $\mu \in \mathbf{R}$:

$$\begin{cases} \max & (r^2+1)\mu - t, \\ \text{s.t.} & D(t) - \mu I \succeq 0, \\ & \mu \leq 0. \end{cases} \tag{6}$$

Slater's condition holds for Problem (6), and it is possible to write its Lagrangian dual that is:

$$\begin{cases} \min & \text{trace}(D(0)X), \\ \text{s.t.} & \text{trace}(X) \leq r^2 + 1, \\ & X_{11} = 1 \\ & X \succeq 0. \end{cases} \tag{7}$$

The algorithm parallels the dual simplex method for linear programming. At each iteration it maintains dual feasibility for Problem (6) and complementary slackness, while iterating to get primal feasibility of Problem (7) ($X_{11} = 1$) and reduce the duality gap.

Essentially these steps can be summarized as follows:

1) find a basic solution $(t, \mu_{\min}(D(t)))$ of Problem (6);
2) find an approximate solution of Problem (7), by using the complementary slackness relation

$$\text{trace}((D(t) - \mu I)X) = 0;$$

an eigenvector $z(t) = (z_0(t), v(t)^\intercal)^\intercal$ corresponding to $\mu_{\min}(D(t))$ is used and $X = (r^2 + 1)\, zz^\intercal$ so that the constraint on the trace of X in Problem (7) is satisfied;
3) use inverse interpolation to predict a value of the parameter t such that $X_{11} = 1$ and/or the duality gap

$$\text{trace}(D(0)X) - ((r^2 + 1)\mu - t)$$

is decreasing.

Some differences occur depending on whether the easy or the hard case happens. Let us denote by $z(t) = (z_0(t), v(t)^\intercal)^\intercal$ the eigenvector of $D(t)$ corresponding to $\mu_{\min}(D(t))$.

In the easy case, the first component $z_0(t) \neq 0$ and the vector $v(t)/z_0(t)$ is the unique optimal solution of

$$\begin{cases} \min & q(x) \\ \text{s.t.} & \|x\|^2 = \frac{1-z_0(t)^2}{z_0(t)^2}. \end{cases}$$

Hence, a value t^* such that $(1-z_0(t^*)^2)/z_0(t^*)^2 = r^2$ must be found and then the point $x^* = v(t^*)/z_0(t^*)$ with multiplier $\lambda^* = -\mu_{\min}(D(t^*))$ is the unique solution of Problem (1). The correct value of t can be found by standard search procedures and the algorithm produces an interval containing t^* that is iteratively updated.

In the hard case, $z_0(t)$ may be zero. However there is still a value t_0 such that $\mu_{\min}(D(t_0)) = \lambda_{\min}(Q)$ and a corresponding eigenvector $z(t_0)$ exists with first component not equal to zero. In order to obtain the value t_0, consider, without loss of generality, a diagonal Q with elements λ_i in increasing order, so that $\lambda_1 = \lambda_{\min}(Q)$. Assume that p is the multiplicity of $\lambda_{\min}(Q)$, and define

$$t_0 = \lambda_{\min}(Q) + \sum_{k=p+1}^{n} \frac{c_k}{\lambda_k - \lambda_{\min}(Q)}.$$

Then the smallest eigenvalue $\mu_{\min}(D(t_0)) = \lambda_{\min}(Q)$ with multiplicity $p + 1$.

Two cases can occur. If $(1-z_0(t_0)^2)/z_0(t_0)^2 > r^2$ then the value $t^* < t_0$. This case can be treated as the preceding easy case since there exists $t < t_0$ such that the eigenvalue $\mu_{\min}(D(t))$ is simple, it results $\mu_{\min}(D(t)) < \lambda_{\min}(Q)$, and the corresponding eigenvector satisfies $(1-z_0(t)^2)/z_0(t)^2 = r^2$. On the other hand, if $z_0^2(t_0) \geq 1/(r^2 + 1)$, then a primal step to the boundary of the feasible region of Problem (7) is taken while improving the objective function. In particular, let $w \in S_{\min}$ with $\|w\| = 1$, then the vector

$$x^* = \frac{v}{z_0(t_0)} + \left[\left(r^2 - \frac{1 - z_0^2(t_0)}{z_0^2(t_0)}\right) w\right]^{\frac{1}{2}}$$

together with $\lambda^* = -\lambda_{\min}(Q))$ satisfy the optimality conditions for Problem (1) and $t^* = t_0$. Hence in the hard case, a vector is found that allows to move to the correct radius while improving the objective function.

Inverse interpolation on the value of the first component z_0 of the eigenvector corresponding to $\mu_{\min}(D(t^k))$ is used to predict a new value for t^{k+1}.

A brief scheme of the algorithm is in Table 5.

Large Scale Trust Region Problems, Table 5
A pseudocode for TR based on SDP

```
procedure TR PRIMAL-DUAL-SDP()
    input instance (Q, c, r, x^0);
    (initialization)
    Find λ_min(Q); set k = 0.
    Set the interval of uncertainty
        [t_l^k, t_u^k] for t*, and [μ_l^k, μ_u^k] for q(x*);
    DO (until a solution is found)
        improve the parameter t^{k+1}
            using inverse interpolation
        update the iterate
            using μ_min(D(t^k)) and its corresponding
            eigenvector;
        update the intervals
            [t_l^{k+1}, t_u^{k+1}] and [μ_l^{k+1}, μ_u^{k+1}];
        set k = k + 1;    OD;
    RETURN(solution)
END TR PRIMAL-DUAL-SDP;
```

Conclusion

All the algorithms described above appear to be potentially equivalent from the computational point of view. They have been implemented in MATLAB [15] codes and the results of the numerical testing are reported in the corresponding papers.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- Best Approximation by Bounded or Continuous Functions
- Cholesky Factorization
- Conjugate-gradient Methods
- Interval Linear Systems
- Large Scale Unconstrained Optimization
- Linear Programming
- Local Attractors for Gradient-related Descent Iterations
- Nonlinear Least Squares: Newton-type Methods
- Nonlinear Least Squares: Trust Region Methods
- Orthogonal Triangularization
- Overdetermined Systems of Linear Equations
- QR Factorization
- Solving Large Scale and Sparse Semidefinite Programs
- Symmetric Systems of Linear Equations

References

1. Ben-tal A, Teboulle M (1996) Hidden convexity in some nonconvex quadratically constrained quadratic programming. Math Program 72(1):51–63
2. Coleman TF, Li Y (1996) An interior trust region approach for nonlinear minimization subject to bounds. SIAM J Optim 6(2):418–445
3. Cullum JK, Willoughby RA (1985) Lanczos algorithms for large symmetric eigenvalue computation. Birkhäuser, Basel
4. Dembo RS, Steihaug T (1983) Truncated-Newton methods algorithms for large-scale unconstrained optimization. Math Program 26(2):190–212
5. DinhTao Pham, HoaiAn LeThi (1998) D.C. optimization algorithm for solving the trust region subproblem. SIAM J Optim 8(2):476–505
6. Facchinei F, Lucidi S (1995) Quadratically and superlinear convergent algorithms for the solution of inequality constrained optimization problems. J Optim Th Appl 85(2):265–289
7. Gay DM (1981) Computing optimal locally constrained steps. SIAM J Sci Statist Comput 2(2):186–197
8. Grippo L, Lampariello F, Lucidi S (1989) A truncated Newton method with nonmonotone linesearch for unconstrained optimization. J Optim Th Appl 60(3):401–419
9. Grippo L, Lampariello F, Lucidi S (1991) A class of nonmonotone stabilization methods in unconstrained optimization. Numerische Math 59:779–805
10. Kamath A, Karmarkar N (1991) A continuous approach to compute upper bounds in quadratic maximization problems with integer constraints. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton Univ. Press, Princeton, pp 125–140
11. Karmarkar N (1990) An interior-point approach to {NP}-complete problems. In: Proc. Math. Program. Soc. Conf. Integer Programming and Combinatorial Optimization, pp 351–366
12. Karmarkar N, Resende MGC, Ramakrishnan KG (1991) An interior point algorithm to solve computationally difficult set covering problems. Math Program 52(3):597–618
13. Lucidi S, Palagi L, Roma M (1998) On some properties of quadratic programs with a convex quadratic constraint. SIAM J Optim 8(1):105–122
14. Martínez JM, Santos SA (1995) Trust region algorithms on arbitrary domains. Math Program 68(3):267–302
15. Matlab (1995) Reference guide. MathWorks
16. Moré JJ, Sorensen DC (1983) Computing a trust region step. SIAM J Sci Statist Comput 4(3):553–572
17. Pardalos PM (1996) Continuous approches to discrete optimization problems. In: Di Pillo G, Giannessi F (eds) Nonlinear Optimization and Applications. Plenum, New York, pp 313–328
18. Pardalos PM, Ye Y, Han C-G (1991) Algorithms for the solution of quadratic knapsack problems. LAA 25:69–91
19. Powell MJD, Yuan Y (1991) A trust region algorithm for equality constrained optimization. Math Program 49:189–211
20. Rendl F, Wolkowicz H (1997) A semidefinite framework to trust region subproblems with applications to large scale minimization. Math Program 77(2):273–299
21. Sartenaer A (1995) A class of trust region methods for nonlinear network optimization problems. SIAM J Optim 5(2):379–407
22. Sorensen DC (1982) Newton's method with a model trust region modification. SIAM J Sci Statist Comput 19(2):409–427
23. Sorensen DC (1997) Minimization of a large-scale quadratic function subject to an ellipsoidal constraint. SIAM J Optim 7(1):141–161
24. Vavasis SA (1991) Nonlinear optimization. Oxford Univ. Press, Oxford
25. Ye Y (1991) A new complexity result on minimization of a quadratic function with a sphere constraint. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton Univ. Press, Princeton, pp 19–31

26. Ye Y (1992) On affine scaling algorithms for nonconvex quadratic programming. Math Program 56:285–300
27. Ye Y, Tse E (1989) An extension of Karmarkar's projective algorithm for convex quadratic programming. Math Program 44:157–179
28. Yuan Y (1990) On a subproblem of trust region algorithms for constrained optimization. Math Program 47:33–63

Large Scale Unconstrained Optimization

LSUO

MASSIMO ROMA
Dip. Inform. e Sistemistica,
Universitá Roma 'La Sapienza', Roma, Italy

MSC2000: 90C06

Article Outline

Keywords
See also
References

Keywords

Large scale problem; Unconstrained optimization

A large scale unconstrained optimization problem can be formulated as the problem of finding a local minimizer of a real valued function $f: \mathbf{R}^n \to \mathbf{R}$ over the space $\mathbf{R}^n$, namely to solve the problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1}$$

where the dimension n is large. The notion of 'large scale' is machine dependent and hence it could be difficult to state a priori when a problem is of large size. However, today an unconstrained problem with more than one thousand variables is usually considered a *large scale problem*.

Besides its own theoretical importance, the growing interest in the last years in solving problems of large size derives from the fact that problems with a larger and larger number of variables are arising very frequently from real world as a result of modeling systems with a very complex structure.

The main difficulty in dealing with large scale problems is the fact that effective algorithms for small scale problems do not necessarily translate into efficient algorithms when applied to solve large problems. Therefore in most cases it is improper to tackle a problem with a large number of variables by using one of the many existing algorithms for the small scale case relying on the growing powerful of the modern computers (see, e. g., [11,13,34] for a review on the existing methods for small scale unconstrained optimization).

A basic feature of an algorithm for large scale problems is a low storage overhead needed to make practicable its implementation. Moreover, whenever a large scale problem has some structure it should be exploited to define reliable algorithms; in fact, often the structure of a problem reflects in the sparsity of the Hessian matrix of the function f which can be efficiently exploited.

Methods for unconstrained optimization differ according to how much information on the function f is available. In the framework of large scale unconstrained optimization it is usually required that the user provides at least subroutines which evaluate the objective function and its gradient for any point x. More effective methods can be obtained if second order derivatives are known. When the derivatives are not available they can be obtained by finite difference or by using automatic differentiation. Throughout we assume that the function f is twice continuously differentiable, i. e. that the gradient $g(x) = \nabla f(x)$ and the Hessian matrix $H(x) = \nabla^2 f(x)$ of the function f exist and are continuous. Moreover, we denote by $\|v\|$ the Euclidean norm of a vector $v \in \mathbf{R}^n$.

As in the small scale case, most of the large scale unconstrained algorithms are iterative methods which generate a sequence of points according to the scheme

$$x_{k+1} = x_k + \alpha_k d_k \tag{2}$$

where $d_k \in \mathbf{R}^n$ is a search direction and $\alpha_k \in \mathbf{R}$ is a steplength obtained by means of a one-dimensional search. Obviously, also in large scale optimization it is important that an algorithm presents both the *global convergence* (i. e. convergence of the sequence $\{x_k\}$ towards a stationary point from any starting point) and a good *convergence rate*.

A basic method for solving large scale unconstrained optimization problems can be considered the *steepest descent method* obtained by setting $d_k = -g(x_k)$ in (2). This method is based on the linear approximation of the objective function f and hence only first order information are need. Due to its very limited storage required by a standard implementation, steepest descent method could be considered very attractive in the large scale setting; moreover the global convergence can also be ensured. However, its convergence rate is only linear and therefore it is too slow to be used. A particular rule for computing the stepsize α_k has been proposed [39] and this led to a significant improvement of the efficiency of the steepest descent method.

One of the most effective methods for solving unconstrained problems is the Newton method (cf. ► Unconstrained nonlinear optimization: Newton–Cauchy framework). It is based on the quadratic approximation of $f(x_k + w)$ given by

$$\phi_k(w) = f(x_k) + g(x_k)^\top w + \tfrac{1}{2} w^\top H(x_k) w \qquad (3)$$

and it is defined by iterations of the form

$$x_{k+1} = x_k + s_k \qquad (4)$$

where the search direction s_k is obtained by minimizing the quadratic model of the objective function (3) over $\mathbf{R}^n$. On the one hand, Newton method presents quadratic convergence rate and it is scale invariant, but, on the other hand, in its pure form it is not globally convergent. Globally convergent modifications of the Newton method has been defined following the line search approach and the trust region approach (see, e. g. [11,12,27]; cf. also ► Large scale trust region problems), but the main difficulty, in dealing with large scale problems, is represented by the possibility to efficiently solve, at each iteration, linear systems which arise in computing the search direction s_k. In fact, the problem dimension could be too large for any explicit use of the Hessian matrix and iterative methods must be used to solve systems of linear equations instead of factorizations of the matrices involved. Indeed, whereas in the small scale setting the Newton direction s_k is usually determined by using direct methods for solving the linear system

$$H(x_k)s = -g(x_k), \qquad (5)$$

when n is large, it is impossible to store or factor the full $n \times n$ Hessian matrix unless it is a sparse matrix. Moreover the exact solution, at each iteration, of the system (5) could be too burdensome and not justified when x_k is far from a solution. In fact, since the benefits of using the Newton direction are mainly local (i. e. in the neighborhood of a solution), it should not be necessary a great computational effort to get an accurate solution of system (5) when $g(x_k)$ is large.

On the basis of these remarks, in [8] the *inexact Newton methods* were proposed. They represent the basic approach underlying most of the Newton-type large scale unconstrained algorithms. The main idea is to approximately solve the system (5) still ensuring a good convergence rate of the method by using a particular trade-off rule between the computational burden required to solve the system (5) and the accuracy with which it is solved. The measure of this accuracy is the relative residual

$$\frac{\|r_k\|}{\|g(x_k)\|}, \quad \text{where } r_k = H(x_k)s_k + g(x_k) \qquad (6)$$

and s_k is an approximate solution of (5). The analysis given in [8] shows that if the sequence $\{x_k\}$ generated by (4) converges to a point $x_\star$ and if

$$\lim_{k\to\infty} \frac{\|r_k\|}{\|g(x_k)\|} = 0, \qquad (7)$$

then $\{x_k\}$ converges superlinearly to $x_\star$. This result is at the basis of the *truncated Newton methods* which represent one of the most effective approach for solving large scale problems. This class of methods was introduced in [9] within the line search based Newton-type methods. They are based on the fact that whenever the Hessian matrix $H(x_k)$ is positive definite, to solve the Newton equation (5) is equivalent to determine the minimizer of the quadratic model (3). Therefore, in these methods, a Newton-type direction, i. e. an approximate solution of (5), is computed by applying the (linear) conjugate gradient (CG) method (cf. ► Conjugate-gradient methods) [23] to approximately minimize the quadratic function (3). A scheme of a line search based truncated Newton algorithm is the following:

Line search based truncated Newton algorithm

OUTER iterations
  For $k = 0, 1, \ldots$
  Compute $g(x_k)$
  Test for convergence
INNER iterations
  (Computation of the direction s_k)
  Iterate CG algorithm until
  a termination criterion is satisfied
  Compute a stepsize α_k by a line search procedure
  Set $x_{k+1} = x_k + \alpha_k s_k$

A scheme for a truncated Newton algorithm

Given a starting point x_0, at each iteration k, a Newton-type direction s_k is computed by truncating the CG iterates – the inner iterations – whenever a required accuracy is obtained. The definition of an effective truncation criterion represents a key aspect of any truncated Newton method and a natural choice is represented by monitoring when the relative residual (6) is sufficiently small. Moreover, by requiring that $\|r_k\| / \|g(x_k)\| \leq \eta_k$ with $\lim_{k\to\infty} \eta_k \to 0$, the condition given by (7) is satisfied and hence the superlinear convergence is guaranteed [9]. In particular η_k can be chosen to ensure that, as a critical point is approached, more accuracy is required. Other truncation criteria based on the reduction of the quadratic model can be defined [31]. Numerical experiences showed that a relatively small number of CG iterations is needed, in most cases, for obtaining a good approximation of the Newton direction and this is one the main advantage of the truncated Newton methods since a considerable computational savings can be obtained still ensuring a good convergence rate. The performance of the CG algorithm used in the inner iterations can be improved by using a preconditioning strategy based either on the information gained during the outer iterations or on some scaling of the variables. Several different preconditioning schemes have been proposed and tested [29,40]. Truncated Newton methods can be modified to enable their use whenever the Hessian matrix is not available; in fact, the CG method only needs the product of the Hessian matrix with a displacement vector, and this product can be approximated by finite difference [35]. The resulting method is called *discrete truncated Newton method*. In [41] a Fortran package (TNPACK) implementing a line search based (discrete) truncated Newton algorithm which uses a preconditioned conjugate gradient is proposed. However, additional safeguard is needed within truncated Newton algorithms since the Hessian matrix could be not positive definite. In fact, the CG inner iterations may break down before satisfying the termination criterion when the Hessian matrix is indefinite. To handle this case, whenever a *direction of negative curvature* (i. e. a direction d_k such that $d_k^\top H(x_k)\, d_k < 0$) is encountered, the inner iterations are usually terminated and a *descent direction* (i. e. a direction d_k such that $g(x_k)^\intercal\, d_k < 0$) is computed [9]. More sophisticated strategies can be applied for iteratively solving the system (5) when it is indefinite [6,15,36,43]. In particular, the equivalent characterization of the linear conjugate gradient algorithm via the Lanczos method can be exploited to define a truncated Newton algorithm which can be used to solve problems with indefinite Hessian matrices [28]. In fact, the Lanczos algorithm does not requires the Hessian matrix to be positive definite and hence it enables to obtain an effective Newton-type direction.

A truncated Newton method which uses a *nonmonotone line search* (i. e. which does not enforce the monotone decrease of the objective function values) was proposed in [20] and the effectiveness of this approach was shown especially in the solution of ill-conditioned problems. Moreover in the CG-truncated scheme proposed in [20] an efficient strategy to handle the indefinite case is also proposed.

A new class of truncated Newton algorithms for solving large scale unconstrained problems has been defined in [25]. In particular, a nonmonotone stabilization framework is proposed based on a *curvilinear line search*, i. e. a line search along the curvilinear path

$$x(\alpha) = x_k + \alpha^2 s_k + \alpha d_k,$$

where s_k is a Newton-type direction and d_k is a particular negative curvature direction which has some resemblance to an eigenvector of the Hessian matrix corresponding to the minimum eigenvalue. The use of the combination of these two directions enables, also in the large scale case, to define a class of line search based algorithms which are globally convergent towards points which satisfy second order necessary optimality conditions, i. e. stationary points where the Hessian matrix is

positive semidefinite. Besides satisfying this important theoretical property, this class of algorithms was also shown to be very efficient in solving large scale unconstrained problems [25,26]. This is also due to the fact that a Lanczos based iterative scheme is used to compute both the directions without terminating the inner iterations when indefiniteness is detected and, as result, more information about the curvature of the objective function are conveyed.

Truncated Newton methods have been also defined within the trust region based methods. These methods are characterized by iterations of the form (4) where, at each iteration k, the search direction s_k is determined by minimizing the quadratic model of the objective function (3) in a neighborhood of the current iterate, namely by solving the problem

$$\min_{\|s\| \le \Delta} \phi_k(s), \tag{8}$$

where Δ is the trust region radius. Also in this framework most of the existing algorithms require the solution of systems of linear equations. Some approaches are the *dogleg methods* [10,38] which aim to solve problem (8) over a one-dimensional arc and the method proposed in [5] which solves problem (8) over a two-dimensional subspace. However, whenever the problem dimension is large, it is impossible to rely on matrix factorizations, and iterative methods must be used. If the quadratic model (3) is positive definite and the trust region radius is sufficiently large that the trust region constraint is inactive at the unconstrained minimizer of the model, problem (8) can be solved by using the preconditioned conjugate gradient method [42,44]. Of course, a suitable strategy is needed whenever the unconstrained minimizer of the quadratic model is no longer lying within the trust region and the desired solution belongs to the trust region boundary. A simple strategy to handle this case was proposed in [42] and [44] and it considers the piecewise linear path connecting the CG iterates, stopping at the point where this path leaves the trust region. If the quadratic model (3) is indefinite, the solution must also lie on the trust region boundary and the piecewise linear path can be again followed until either it leaves the trust region, or a negative curvature direction is found. In this latter case, two possibilities have been considered: in [42] the path is continued along this direction until the boundary is reached; in [44] the minimizer of the quadratic model within the trust region along the steepest descent direction (the *Cauchy point*) is considered. This class of algorithms represents a trust region version of truncated Newton methods and an efficient implementation is carried out within the LANCELOT package [7]. These methods have become very important in large scale optimization, due to both their strong theoretical convergence properties and good efficiency in practice, but they are known to possess some drawbacks. Indeed, they are essentially unconcerned with the trust region until they blunder into its boundary and stop. Moreover, numerical experiences showed that very frequently this untimely stop happens during the first inner iterations when a negative curvature is present and this could deteriorate the efficiency of the method. In order to overcome this drawback an alternative strategy is proposed in [16] where ways of continuing the process once the boundary of the trust region is reached are investigated. The key point of this approach is the use of the Lanczos method and the fact that preconditioned conjugate gradient and Lanczos methods generate different bases for the same Krylov space. Several other large scale trust region methods (cf. ► Large scale trust region problems) have been proposed.

Another class of methods which can be successfully applied to solve large scale unconstrained optimization problems is the wide class of the nonlinear conjugate gradient methods [14,23]. They are extensions to the general (nonquadratic) case of the already mentioned linear conjugate gradient method. They represent a compromise between steepest descent method and Newton method and they are particularly suited for large scale problems since there is never a need to store a full Hessian matrix. They are defined by the iteration scheme (2) where the search direction is of the form

$$d_k = -g(x_k) + \beta_k d_{k-1} \tag{9}$$

with $d_0 = -g(x_0)$ and where β_k is a scalar such that the algorithm reduces to the linear conjugate gradient method if the objective function f is a strictly convex quadratic function and α_k in (2) is obtained by means of an exact line search (i. e., α_k is the one-dimensional minimizer of $f(x_k + \alpha\, d_k)$ with respect to α). The most widely used formulas for β_k are *Fletcher–Reeves* (FR)

and *Polak–Ribière* (PR) formulas given by

$$\beta_k^{\mathrm{FR}} = \frac{\|g(x_k)\|^2}{\|g(x_{k-1})\|^2},$$
$$\beta_k^{\mathrm{PR}} = \frac{g(x_k)^\top \left[g(x_k) - g(x_{k-1})\right]}{\|g(x_{k-1})\|^2}.$$

Many efforts have been devoted to investigate the global convergence for nonlinear conjugate gradient methods. A widespread technique to enforce the global convergence is the use of a regular *restart* along the steepest descent direction every n iterations obtained by setting $\beta_k = 0$. However, computational experiences showed that this restart can have a negative effect on the efficiency of the method; on the other hand, in the large scale setting, restarting does not play a significant role since n is large and very few restarts can be performed. Global convergence results have been obtained for the *Fletcher–Reeves method* without restart both in the case of exact line search [46] and when α_k is computed by means of an inexact line search [1]; then, the global convergence was extended to methods with $|\beta_k| \le \beta_k^{FR}$ [14]. As regards the global convergence of the *Polak–Ribière method*, for many years it was proved with exact line search only under strong convexity assumptions [37]. Global convergence both for exact and inexact line search can also be enforced by modifying the Polak–Ribière method by setting $\beta_k = \max\{\beta_k^{PR}, 0\}$ [14]; this strategy correspond to restart the iterations along the steepest descent direction whenever a negative value of β_k occurs. However, an inexact line search which ensures global convergence of the Polak–Ribière method for nonconvex function has been obtained in [21]. As regards the numerical performance of these two methods, extensive numerical experiences showed that, in general, Polak–Ribière method is usually more efficient than the Fletcher–Reeves method. An efficient implementation of the Polak–Ribière method (with restarts) is available as routine VA14 within the Harwell subroutine library [22]. See, e. g., [34] for a detailed survey on the nonlinear conjugate gradient methods.

Another effective approach to large scale unconstrained optimization is represented by the *limited-memory BFGS method* (L-BFGS) proposed in [32] and then studied in [24,30]. This method resembles the BFGS quasi-Newton method, but it is particularly suited for large scale (unstructured) problems because the storage of matrices is avoided. It is defined by the iterative scheme (2) with the search direction given by

$$d_k = -H_k g(x_k)$$

and where H_k is the approximation to the inverse Hessian matrix of the function f at the kth iteration. In the BFGS method the approximation H_k is updated by means of the BFGS correction given by

$$H_{k+1} = V_k^\top H_k V_k + \rho_k s_k s_k^\top$$

where $V_k = I - \rho_k\, y_k\, s_k^\top$, $s_k = x_{k+1} - x_k$, $y_k = g(x_{k+1}) - g(x_k)$, and $\rho_k = 1/y_k^\top\, s_k$. In the L-BFGS method, instead of storing the matrices H_k, a prefixed number (say m) of vectors pairs $\{s_k, y_k\}$ that define them implicitly are stored. Therefore, during the first m iterations the L-BFGS and the BFGS methods are identical, but when $k > m$ only information from the m previous iterations are used to obtain H_k. The number m of BFGS corrections that must be kept can be specified by the user. Moreover, in the L-BFGS the product $H_k\, g(x_k)$ which represents the search direction is obtained by means of a recursive formula involving $g(x_k)$ and the most recent vectors pairs $\{s_k, y_k\}$. An implementation of L-BFGS method is available as VA15 routine within the Harwell subroutine library [22]. An interesting numerical study of L-BFGS method and a comparison of its numerical performance with the discrete truncated Newton method and the Polak–Ribière conjugate gradient method are reported in [30]. The results of a numerical experience with limited-memory quasi-Newton and truncated Newton methods on standard library test problems and on two real life large scale unconstrained optimization applications can be found in [45]. A method which combines the discrete Newton method and the L-BFGS method is proposed in [4] to produce an efficient algorithm able to handle also ill-conditioned problems.

Limited memory quasi-Newton methods represent an adaptation of the quasi-Newton methods to large scale *unstructured optimization*. However, the quasi-Newton approach can be successfully applied to large scale problems with a particular structure. In fact, fre-

quently, an optimization problem has some structure which may be reflected in the sparsity of the Hessian matrix. In this framework, the most effective method is the *partitioned quasi-Newton method* proposed in [18,19]. It is based on the fact that a function f with a sparse Hessian is a *partially separable function*, i. e. it can be written in the form

$$f(x) = \sum_{i=1}^{n_e} f_i(x)$$

where the element functions f_i depends only on a few variables. Many practical problems can be formulated (or recasted) in this form showing a wide range of applicability of this approach. The basic idea of the partitioned quasi-Newton method is to decompose the Hessian matrix into a sum of Hessians of the element functions f_i. Each approximation to the Hessian of f_i is then updated by using dense updating techniques. These small matrices are assembled to define an approximation to the Hessian matrix of f used to compute the search direction. However, the element Hessian matrices may not be positive definite and hence BFGS formula cannot be used, and in this case a symmetric rank one formula is used. Global convergence results have been obtained under convexity assumption of the function f_i [17]. An implementation of the partitioned quasi-Newton method is available as VE08 routine of the Harwell subroutine library [22]. A comparison of the performance of partitioned quasi-Newton, L-BFGS, CG Polak–Ribière and truncated discrete Newton methods is reported in [33].

Another class of methods which has been extended to large sparse unconstrained optimization are *tensor methods* [3]. Tensor methods are based on fourth order model of the objective function and are particularly suited for problems where the Hessian matrix has a small rank deficiency.

To conclude, it is worthy to outline that in dealing with large scale unconstrained problems with a very large number of variables (more than 10^4) high performance computer architectures must be considered. See e. g. [2] for the solution of large scale optimization problems on vector and parallel architectures.

The reader can find the details of the methods mentioned in this brief survey in the specific cited references.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- Broyden Family of Methods and the BFGS Update
- Cholesky Factorization
- Conjugate-gradient Methods
- Continuous Global Optimization: Models, Algorithms and Software
- Interval Linear Systems
- Large Scale Trust Region Problems
- Linear Programming
- Modeling Languages in Optimization: A New Paradigm
- Nonlinear Least Squares: Trust Region Methods
- Optimization Software
- Orthogonal Triangularization
- Overdetermined Systems of Linear Equations
- QR Factorization
- Solving Large Scale and Sparse Semidefinite Programs
- Symmetric Systems of Linear Equations
- Unconstrained Nonlinear Optimization: Newton–Cauchy Framework
- Unconstrained Optimization in Neural Network Training

References

1. Al-Baali M (1985) Descent property and global convergence of the Fletcher–Reeves method with inexact line search. IMA J Numer Anal 5:121–124
2. Averick BM, Moré JJ (1994) Evaluation of large-scale optimization problems on vector and parallel architectures. SIAM J Optim 4:708–721
3. Bouaricha A (1997) Tensor methods for large, sparse unconstrained optimization. SIAM J Optim 7:732–756
4. Byrd RH, Nocedal J, Zhu C (1995) Towards a discrete Newton method with memory for large-scale optimization. In: Di Pillo G, Giannessi F (eds) Nonlinear Optimization and Applications. Plenum, New York, pp 1–12
5. Byrd RH, Schnabel RB, Shultz GA (1988) Approximate solution of the trust region problem by minimization over two-dimensional subspaces. Math Program 40:247–263
6. Chandra R (1978) Conjugate gradient methods for partial differential equations. PhD Thesis Yale Univ.
7. Conn AR, Gould NIM, Toint PhL (1992) LANCELOT: A Fortran package for large-scale nonlinear optimization (release A). Springer, Berlin
8. Dembo RS, Eisenstat SC, Steihaug T (1982) Inexact Newton methods. SIAM J Numer Anal 19:400–408

9. Dembo RS, Steihaug T (1983) Truncated-Newton algorithms for large-scale unconstrained optimization. Math Program 26:190–212
10. Dennis JE, Mei HHW (1979) Two new unconstrained optimization algorithms which use function and gradient values. J Optim Th Appl 28:453–482
11. Dennis JE, Schnabel RB (1989) A view of unconstrained optimization. In: Nemhauser GL, Rinnooy Kan AHG, Tood MJ (eds) Handbook Oper. Res. and Management Sci., vol 1. North-Holland, Amsterdam, pp 1–72
12. Fletcher R (1987) Practical methods of optimization. Wiley, New York
13. Fletcher R (1994) An overview of unconstrained optimization. In: Spedicato E (ed) Algorithms for continuous optimization. The state of the art. Kluwer, Dordrecht, pp 109–143
14. Gilbert JC, Nocedal J (1992) Global convergence properties of conjugate gradient methods for optimization. SIAM J Optim 2:21–42
15. Gill PE, Murray W, Ponceleon DB, Saunders MA (1992) Preconditioners for indefinite systems arising in optimization. SIAM J Matrix Anal Appl 13:292–311
16. Gould NIM, Lucidi S, Roma M, Toint PhL (1999) Solving the trust-region subproblem using the Lanczos method. SIAM J Optim 9:504–525
17. Griewank A (1991) The global convergence of partitioned BFGS on problems with convex decomposition and Lipschitzian gradients. Math Program 50:141–175
18. Griewank A, Toint PhL (1982) Local convergence analysis of partitioned quasi-Newton updates. Numerische Math 39:429–448
19. Griewank A, Toint PhL (1982) Partitioned variable metric updates for large structured optimization problems. Numerische Math 39:119–137
20. Grippo L, Lampariello F, Lucidi S (1989) A truncated Newton method with nonmonotone linesearch for unconstrained optimization. J Optim Th Appl 60:401–419
21. Grippo L, Lucidi S (1997) A globally convergent version of the Polak–Ribière conjugate gradient method. Math Program 78:375–391
22. Harwell Subroutine Library (1998) A catalogue of subroutines. AEA Techn.
23. Hestenes MR (1980) Conjugate direction methods in optimization. Springer, Berlin
24. Liu DC, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Math Program 45:503–528
25. Lucidi S, Rochetich F, Roma M (1998) Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization. SIAM J Optim 8:916–939
26. Lucidi S, Roma M (1997) Numerical experiences with new truncated Newton methods in large scale unconstrained optimization. Comput Optim Appl 7:71–87
27. Moré JJ, Sorensen DC (1984) Newton's method. In: Golub GH (ed) Studies in Numerical Analysis. Math. Assoc. Amer., Washington, DC, pp 29–82
28. Nash SG (1984) Newton-type minimization via the Lanczos method. SIAM J Numer Anal 21:770–788
29. Nash SG (1985) Preconditioning of truncated-Newton methods. SIAM J Sci Statist Comput 6:599–616
30. Nash SG, Nocedal J (1991) A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization. SIAM J Optim 1:358–372
31. Nash SG, Sofer A (1990) Assessing a search direction within a truncated-Newton method. Oper Res Lett 9:219–221
32. Nocedal J (1980) Updating quasi-Newton matrices with limited storage. Math Comput 35:773–782
33. Nocedal J (1990) The performance of several algorithms for large-scale unconstrained optimization. In: Coleman TF, Li Y (eds) Large-scale Numerical Optimization. SIAM, Philadelphia, pp 138–151
34. Nocedal J (1992) Theory and algorithms for unconstrained optimization. Acta Numer 1:199–242
35. O'Leary DP (1982) A discrete Newton algorithm for minimizing a function of many variables. Math Program 23:20–33
36. Paige CC, Saunders MA (1975) Solution of sparse indefinite systems of linear equations. SIAM J Numer Anal 12:617–629
37. Polak E, Ribière G (1969) Note sur la convergence de methodes de directions conjugées. Revue Franc Inform et Rech Oper 16:35–43
38. Powell MJD (1970) A new algorithm for unconstrained optimization. In: Mangasarian OL, Ritter K (eds) Nonlinear programming. Acad. Press, New York, pp 31–65
39. Raydan M (1997) The Barzilai and Borwein gradient method for large scale unconstrained minimization problems. SIAM J Optim 7:26–33
40. Schlick T (1993) Modified Cholesky factorization for sparse preconditioners. SIAM J Sci Comput 14:424–445
41. Schlick T, Fogelson A (1992) TNPACK – A truncated Newton package for large-scale problems: I. Algorithm and usage. ACM Trans Math Softw 18:46–70
42. Steihaug T (1983) The conjugate gradient method and trust regions in large-scale optimization. SIAM J Numer Anal 20:626–637
43. Stoer J (1983) Solution of large linear systems of equations by conjugate gradient type methods. In: Bachem A, Grötschel M, Korte B (eds) Mathematical Programming. The State of the Art. Springer, Berlin, pp 540–565
44. Toint PhL (1981) Towards an efficient sparsity exploiting Newton method for minimization. In: Duff IS (ed) Sparse Matrices and Their Uses. Acad. Press, New York, pp 57–88
45. Zou X, Navon IM, Berger M, Phua KH, Schlick T, Dimet FX (1993) Numerical experience with limited-memory quasi-Newton and truncated Newton methods. SIAM J Optim 3:582–608

46. Zoutendijk G (1970) Nonlinear programming computational methods. In: Abadie J (ed) Integer and Nonlinear Programming. North-Holland, Amsterdam, pp 37–86

L-convex Functions and M-convex Functions

KAZUO MUROTA
Res. Institute Math. Sci. Kyoto University, Kyoto, Japan

MSC2000: 90C27, 90C25, 90C10, 90C35

Article Outline

Keywords
Definitions of L- and M-Convexity
L-Convex Sets
M-Convex Sets
Properties of L-Convex Functions
Properties of M-Convex Functions
$L^\natural$- and $M^\natural$-Convexity
Duality
Network Duality
Subdifferentials
Algorithms
Applications
See also
References

Keywords

L-convexity; M-convexity; Discrete convex analysis; Submodular function; Matroid

In the field of nonlinear programming (in continuous variables), convex analysis [20,21] plays a pivotal role both in theory and in practice. An analogous theory for discrete optimization (nonlinear integer programming), called 'discrete convex analysis' [15,16], is developed for L-convex and M-convex functions by adapting the ideas in convex analysis and generalizing the results in matroid theory. The L- and M-convex functions are introduced in [15] and [12,18], respectively.

Definitions of L- and M-Convexity

Let V be a nonempty finite set and $\mathbf{Z}$ be the set of integers. For any function $g: \mathbf{Z}^V \to Z \cup \{+\infty\}$ define dom $g = \{p \in \mathbf{Z}^V : g(p) < +\infty\}$, called the *effective domain* of g.

A function $g: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ with dom $g \neq \emptyset$ is called *L-convex* if

$$g(p) + g(q) \geq g(p \vee q) + g(p \wedge q) \quad (p, q \in \mathbb{Z}^V),$$
$$\exists r \in \mathbb{Z} : \; g(p + \mathbf{1}) = g(p) + r \quad (p \in \mathbb{Z}^V),$$

where $p \vee q = (\max(p(v), q(v)) \,|v \in V) \in \mathbf{Z}^V$, $p \wedge q = (\min p(v), q(v))|v \in V) \in \mathbf{Z}^V$, and $\mathbf{1}$ is the vector in $\mathbf{Z}^V$ with all components being equal to 1.

A set $D \subseteq \mathbf{Z}^V$ is said to be an *L-convex set* if its indicator function δ_D (defined by $\delta_D(p) = 0$ if $p \in D$, and $= +\infty$ otherwise) is an L-convex function, i. e., if

i) $D \neq \emptyset$;
ii) $p, q \in D \Rightarrow p \vee q, p \wedge q \in D$; and
iii) $p \in D \Rightarrow p \pm \mathbf{1} \in D$.

A function $f: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ with dom $f \neq \emptyset$ is called *M-convex* if it satisfies

- M-EXC) For x, $y \in$ dom f and $u \in \text{supp}^+(x - y)$, there exists $v \in \text{supp}^-(x - y)$ such that

$$f(x) + f(y) \geq f(x - \chi_u + \chi_v) + f(y + \chi_u - \chi_v),$$

where, for any $u \in V$, χ_u is the *characteristic vector* of u (defined by $\chi_u(v) = 1$ if $v = u$, and $= 0$ otherwise), and

$$\text{supp}^+(z) = \{v \in V : \; z(v) > 0\} \quad (z \in \mathbb{Z}^V),$$
$$\text{supp}^-(z) = \{v \in V : \; z(v) < 0\} \quad (z \in \mathbb{Z}^V).$$

A set $B \subseteq \mathbf{Z}^V$ is said to be an *M-convex set* if its indicator function is an M-convex function, i. e., if B satisfies

- B-EXC) For $x, y \in B$ and for $u \in \text{supp}^+(x - y)$, there exists $v \in \text{supp}^-(x - y)$ such that $x - \chi_u + \chi_v \in B$ and $y + \chi_u - \chi_v \in B$.

This means that an M-convex set is the same as the set of integer points of the base polyhedron of an integral submodular system (see [8] for submodular systems).

L-convexity and M-convexity are conjugate to each other under the *integral Fenchel–Legendre transformation* $f \longmapsto f^\bullet$ defined by

$$f^\bullet(p) = \sup\{\langle p, x\rangle - f(x) : \; x \in \mathbb{Z}^V\}, \quad p \in \mathbb{Z}^V,$$

where $\langle p, x\rangle = \sum_{v \in V} p(v)\, x(v)$. That is, for L-convex function g and M-convex function f, it holds [15] that $g^\bullet$ is M-convex, $f^\bullet$ is L-convex, $g^{\bullet\bullet} = g$, and $f^{\bullet\bullet} = f$.

Example 1 (Minimum cost flow problem) L-convexity and M-convexity are inherent in the integer minimum-cost flow problem, as pointed out in [12,15]. Let $G = (V, A)$ be a graph with vertex set V and arc set A, and let $T \subseteq V$ be given. For $\xi: A \to Z$ its *boundary* $\partial\xi: V \to \mathbf{Z}$ is defined by

$$\partial\xi(v) = \sum \{\xi(a): a \in \delta^+ v\} - \sum \{\xi(a): a \in \delta^- v\} \quad (v \in V),$$

where $\delta^+ v$ and $\delta^- v$ denote the sets of out-going and incoming arcs incident to v, respectively. For $\widetilde{p}: V \to \mathbb{Z}$ its *coboundary* $\delta\widetilde{p}: A \to \mathbb{Z}$ is defined by

$$\delta\widetilde{p}(a) = \widetilde{p}(\partial^+ a) - \widetilde{p}(\partial^- a) \quad (a \in A),$$

where $\partial^+ a$ and $\partial^- a$ mean the initial and terminal vertices of a, respectively. Denote the class of one-dimensional discrete convex functions by

$$C_1 = \{\varphi: \mathbb{Z} \to \mathbb{Z} \cup \{+\infty\} \mid \operatorname{dom}\varphi \neq \emptyset, \varphi(t-1) + \varphi(t+1) \geq 2\varphi(t) \quad (t \in \mathbb{Z})\}.$$

For $\varphi_a \in \mathcal{C}_1$ $(a \in A)$, representing the arc-cost in terms of flow, the *total cost function* $f: \mathbf{Z}^T \to \mathbf{Z} \cup \{+\infty\}$ defined by

$$f(x) = \inf_{\xi} \left\{ \sum_{a \in A} \varphi_a(\xi(a)) : \begin{array}{c} \partial\xi(v) = -x(v) \\ (v \in T), \\ \partial\xi(v) = 0 \\ (v \in V \setminus T) \end{array} \right\} \quad (x \in \mathbb{Z}^T)$$

is M-convex, provided that $f > -\infty$ (i.e., f does not take the value of $-\infty$). For $\psi_a \in \mathcal{C}_1$ $(a \in A)$, representing the arc-cost in terms of tension, the total cost function $g: \mathbf{Z}^T \to \mathbf{Z} \cup \{+\infty\}$ defined by

$$g(p) = \inf_{\widetilde{p}} \left\{ \sum_{a \in A} \psi_a(\eta(a)) : \begin{array}{c} \eta = -\delta\widetilde{p}, \\ \widetilde{p}(v) = p(v) \\ (v \in T) \end{array} \right\} \quad (p \in \mathbb{Z}^T)$$

is L-convex, provided that $g > -\infty$. The two cost functions $f(x)$ and $g(p)$ are conjugate to each other in the sense that, if $\psi_a = \varphi_a^\bullet$ $(a \in A)$, then $g = f^\bullet$.

Example 2 (Polynomial matrix) Let $A(s)$ be an $m \times n$ matrix of rank m with each entry being a polynomial in a variable s, and let $\mathcal{B} \subseteq 2^V$ be the family of bases of $A(s)$ with respect to linear independence of the column vectors; namely, $J \subseteq V$ belongs to $\mathcal{B}$ if and only if $|J| = m$ and the column vectors with indices in J are linearly independent. Then $f: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ defined by

$$f(x) = \begin{cases} -\deg_s \det A[J] & (x = \chi_J,\ J \in \mathcal{B}) \\ +\infty & (\text{otherwise}) \end{cases}$$

is M-convex, where $\chi_J \in \{0, 1\}^V$ is the characteristic vector of J (defined by $\chi_J(v) = 1$ if $v \in J$, and $= 0$ otherwise), $A[J]$ denotes the $m \times m$ submatrix with column indices in $J \in \mathcal{B}$, and $\deg_s(\cdot)$ means the degree as a polynomial in s. The Grassmann–Plücker identity implies the exchange property of f. This example was the motivation of valuated matroids in [2,3], which in turn can be identified with the negative of M-convex functions f with $\operatorname{dom} f \subseteq \{0, 1\}^V$.

For $p = (p(v))_{v \in V} \in \mathbf{Z}^V$ denote by $D(p)$ the diagonal matrix of order $n = |V|$ with diagonal elements $s^{p(v)}$ $(v \in V)$. Then the function $g: \mathbf{Z}^V \to \mathbf{Z}$ defined by

$$g(p) = \max \{\deg_s \det(A \cdot D(p))[J]: J \in \mathcal{B}\}$$

is L-convex [16], where $(A \cdot D(p))[J]$ means the $m \times m$ submatrix of $A \cdot D(p)$ with column indices in J. We have $g = f^\bullet$.

L-Convex Sets

An L-convex set $D \subseteq \mathbf{Z}^V$ has 'no holes' in the sense that $D = \overline{D} \cap \mathbb{Z}^V$, where $\overline{D}$ denotes the convex hull of D in $\mathbf{R}^V$. Hence it is natural to consider the polyhedral description of $\overline{D}$, 'L-convex polyhedron' (see [15,16]). For any function $\gamma: V \times V \to \mathbf{Z} \cup \{+\infty\}$ with $\gamma(v, v) = 0$ $(v \in V)$, define

$$\mathbf{D}(\gamma) = \left\{ p \in \mathbb{R}^V : \begin{array}{c} p(v) - p(u) \leq \gamma(u, v) \\ (\forall u, v \in V) \end{array} \right\}.$$

If $\mathbf{D}(\gamma) \neq \emptyset$, $\mathbf{D}(\gamma)$ is an integral polyhedron and $D = \mathbf{D}(\gamma) \cap \mathbf{Z}^V$ is an L-convex set. If γ satisfies triangle inequality:

$$\gamma(u, v) + \gamma(v, w) \geq \gamma(u, w) \quad (u, v, w \in V),$$

then $\mathbf{D}(\gamma) \neq \emptyset$ and

$$\gamma(u, v) = \sup \{p(v) - p(u): p \in \mathbf{D}(\gamma)\} \quad (u, v \in V).$$

Conversely, for any nonempty $D \subseteq \mathbf{Z}^V$,

$$\gamma(u, v) = \sup\{p(v) - p(u) : p \in D\} \quad (u, v \in V),$$

satisfies triangle inequality as well as $\gamma(v, v) = 0$ ($v \in V$), and if D is L-convex, then $\overline{D} = \mathbf{D}(\gamma)$. Thus there is a one-to-one correspondence between L-convex set D and function γ satisfying triangle inequality. In particular, $D \subseteq \mathbf{Z}^V$ is L-convex if and only if $D = \mathbf{D}(\gamma) \cap \mathbf{Z}^V$ for some γ satisfying triangle inequality. For L-convex sets $D_1, D_2 \subseteq \mathbf{Z}^V$, it holds that $D_1 + D_2 = \overline{D_1 + D_2} \cap \mathbb{Z}^V$ and $\overline{D_1} \cap \overline{D_2} = \overline{D_1 \cap D_2}$.

It is also true that a function γ satisfying triangle inequality corresponds one-to-one to a positively homogeneous M-convex function f (i. e., $f(\lambda\, x) = \lambda\, f(x)$ for $x \in \mathbf{Z}^V$ and $0 \leq \lambda \in \mathbf{Z}$). The correspondence $f \longmapsto \gamma$ is given by

$$\gamma(u, v) = f(\chi_v - \chi_u) \quad (u, v \in V),$$

whereas $\gamma \longmapsto f$ by

$$f(x) = \inf_{\lambda} \left\{ \sum_{u,v \in V} \lambda_{uv} \gamma(u, v) : \begin{array}{c} \sum_{u,v \in V} \lambda_{uv}(\chi_v - \chi_u) = x, \\ 0 \leq \lambda_{uv} \in \mathbb{Z} \\ (u, v \in V) \end{array} \right\} \quad (x \in \mathbb{Z}^V).$$

The correspondence between L-convex sets and positively homogeneous M-convex functions via functions with triangle inequality is a special case of the conjugacy relationship between L- and M-convex functions.

M-Convex Sets

An M-convex set $B \subseteq \mathbf{Z}^V$ has 'no holes' in the sense that $B = \overline{B} \cap \mathbb{Z}_V$. Hence it is natural to consider the polyhedral description of $\overline{B}$, 'M-convex polyhedron'. A set function $\rho: 2^V \to \mathbf{Z} \cup \{+\infty\}$ is said to be *submodular* if

$$\rho(X) + \rho(Y) \geq \rho(X \cup Y) + \rho(X \cap Y) \quad (X, Y \subseteq V),$$

where the inequality is satisfied if $\rho(X)$ or $\rho(Y)$ is equal to $+\infty$. It is assumed throughout that $\rho(\emptyset) = 0$ and $\rho(V) < +\infty$ for any set function $\rho: 2^V \to \mathbf{Z} \cup \{+\infty\}$. For a set function ρ, define

$$\mathbf{P}(\rho) = \left\{ x \in \mathbb{R}^V : \begin{array}{c} x(X) \leq \rho(X) \\ (\forall X \subset V), \\ x(V) = \rho(V) \end{array} \right\},$$

where $x(X) = \sum_{v \in X} x(v)$. If ρ is submodular, $\mathbf{P}(\rho)$ is a nonempty integral polyhedron, $B = \mathbf{P}(\rho) \cap \mathbf{Z}^V$ is an M-convex set, and

$$\rho(X) = \sup\{x(X) : x \in \mathbf{P}(\rho)\} \quad (X \subseteq V).$$

Conversely, for any nonempty $B \subseteq \mathbf{Z}^V$, define a set function ρ by

$$\rho(X) = \sup\{x(X) : x \in B\} \quad (X \subseteq V).$$

If B is M-convex, then ρ is submodular and $\overline{B} = \mathbf{P}(\rho)$. Thus there is a one-to-one correspondence between M-convex set B and submodular set function ρ. In particular, $B \subseteq \mathbf{Z}^V$ is M-convex if and only if $B = \mathbf{P}(\rho) \cap \mathbf{Z}^V$ for some submodular ρ. The correspondence $B \leftrightarrow \rho$ is a restatement of a well-known fact [4,8]. For M-convex sets $B_1, B_2 \subseteq \mathbf{Z}^V$, it holds that $B_1 + B_2 = \overline{B_1 + B_2} \cap \mathbb{Z}^V$ and $\overline{B_1} \cap \overline{B_2} = \overline{B_1 \cap B_2}$.

It is also true that a submodular set function ρ corresponds one-to-one to a positively homogeneous L-convex function g. The correspondence $g \longmapsto \rho$ is given by the restriction

$$\rho(X) = g(\chi_X) \quad (X \subseteq V)$$

(χ_X is the characteristic vector of X), whereas $\rho \longmapsto g$ by the Lovász extension (explained below). The correspondence between M-convex sets and positively homogeneous L-convex functions via submodular set functions is a special case of the conjugacy relationship between M- and L-convex functions.

For a set function $\rho: 2^V \to \mathbf{Z} \cup \{+\infty\}$, the *Lovász extension* [11] of ρ is a function $\widehat{\rho}: \mathbb{R}^V \to \mathbb{R} \cup \{+\infty\}$ defined by

$$\widehat{\rho}(p) = \sum_{j=1}^{n} (p_j - p_{j+1}) \rho(V_j) \quad (p \in \mathbb{R}^V),$$

where, for each $p \in \mathbf{R}^V$, the elements of V are indexed as $\{v_1, \ldots, v_n\}$ (with $n = |V|$) in such a way that $p(v_1) \geq \cdots \geq p(v_n)$; $p_j = p(v_j)$, $V_j = \{v_1, \ldots, v_j\}$ for $j = 1, \ldots, n$, and

$p_{n+1} = 0$. The right-hand side of the above expression is equal to $+\infty$ if and only if $p_j - p_{j+1} > 0$ and $\rho(V_j) = +\infty$ for some j with $1 \le j \le n-1$. The Lovász extension $\widehat{\rho}$ is indeed an extension of ρ, since $\widehat{\rho}(\chi_X) = \rho(X)$ for $X \subseteq V$.

The relationship between submodularity and convexity is revealed by the statement [11] that a set function ρ is submodular if and only if its Lovász extension $\widehat{\rho}$ is convex.

The restriction to $\mathbf{Z}^V$ of the Lovász extension of a submodular set function is a positively homogeneous L-convex function, and any positively homogeneous L-convex function can be obtained in this way [15].

Properties of L-Convex Functions

For any $g: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ and $x \in \mathbf{R}^V$, define $g[-x]: \mathbf{Z}^V \to \mathbf{R} \cup \{+\infty\}$ by

$$g[-x](p) = g(p) - \langle p, x \rangle \quad (p \in \mathbb{Z}^V).$$

The set of the minimizers of $g[-x]$ is denoted as $\operatorname{argmin}(g[-x])$.

Let $g: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ be L-convex. Then $\operatorname{dom} g$ is an L-convex set. For each $p \in \operatorname{dom} g$,

$$\rho_p(X) = g(p + \chi_X) - g(p) \quad (X \subseteq V)$$

is a submodular set function with $\rho_p(\emptyset) = 0$ and $\rho_p(V) < +\infty$.

An L-convex function g can be extended to a convex function $\overline{g}: \mathbb{R}^V \to \mathbb{R} \cup \{+\infty\}$ through the Lovász extension of the submodular set functions ρ_p for $p \in \operatorname{dom} g$. Namely, for $p \in \operatorname{dom} g$ and $q \in [0, 1]^V$, it holds [15] that

$$\overline{g}(p + q) = g(p) + \sum_{j=1}^{n} (q_j - q_{j+1})(g(p + \chi_{V_j}) - g(p)),$$

where, for each q, the elements of V are indexed as $\{v_1, \ldots, v_n\}$ (with $n = |V|$) in such a way that $q(v_1) \ge \cdots \ge q(v_n)$; $q_j = q(v_j)$, $V_j = \{v_1, \ldots, v_j\}$ for $j = 1, \ldots, n$, and $q_{n+1} = 0$. The expression of $\overline{g}$ shows that an L-convex function is an integrally convex function in the sense of [5].

An L-convex function g enjoys *discrete midpoint convexity*:

$$g(p) + g(q) \ge g\left(\left\lceil \frac{p+q}{2} \right\rceil\right) + g\left(\left\lfloor \frac{p+q}{2} \right\rfloor\right)$$

for $p, q \in \mathbf{Z}^V$, where $\lceil p \rceil$ (or $\lfloor p \rfloor$) for any $p \in \mathbf{R}^V$ denotes the vector obtained by rounding up (or down) the components of p to the nearest integers.

The minimum of an L-convex function g is characterized by the local minimality in the sense that, for $p \in \operatorname{dom} g$, $g(p) \le g(q)$ for all $q \in Z^V$ if and only if $g(p + \mathbf{1}) = g(p) \le g(p + \chi_X)$ for all $X \subseteq V$.

The minimizers of an L-convex function, if nonempty, form an L-convex set. For any $x \in \mathbf{R}^V$, $\operatorname{argmin}(g[-x])$, if nonempty, is an L-convex set. Conversely, this property characterizes L-convex functions under an auxiliary assumption.

A number of operations can be defined for L-convex functions [15,16]. For $x \in \mathbf{Z}^V$, $g[-x]$ is an L-convex function. For $a \in \mathbf{Z}^V$ and $\beta \in \mathbf{Z}$, $g(a + \beta p)$ is L-convex in p. For $U \subseteq V$, the projection of g to U:

$$g^U(p') = \inf\left\{g(p', p'') : p'' \in \mathbb{Z}^{V \setminus U}\right\} \quad (p' \in \mathbb{Z}^U)$$

is L-convex in p', provided that $g^U > -\infty$. For $\psi_v \in \mathcal{C}_1$ $(v \in V)$,

$$\widetilde{g}(p) = \inf_{q \in \mathbb{Z}^V}\left[g(q) + \sum_{v \in V} \psi_v(p(v) - q(v))\right]$$

is L-convex in $p \in \mathbf{Z}^V$, provided that $\widetilde{g} > -\infty$. The sum of two (or more) L-convex functions is L-convex, provided that its effective domain is nonempty.

Properties of M-Convex Functions

Let $f: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ be M-convex. Then $\operatorname{dom} f$ is an M-convex set. For each $x \in \operatorname{dom} f$,

$$\gamma_x(u, v) = f(x - \chi_u + \chi_v) - f(x) \quad (u, v \in V)$$

satisfies [16] triangle inequality.

An M-convex function f can be extended to a convex function $\overline{f}: \mathbb{R}^V \to \mathbb{R} \cup \{+\infty\}$, and the value of $\overline{f}(x)$ for $x \in \mathbf{R}^V$ is determined by $\{f(y): y \in \mathbf{Z}^V, \lfloor x \rfloor \le y \le \lceil x \rceil\}$. That is, an M-convex function is an integrally convex function in the sense of [5].

The minimum of an M-convex function f is characterized by the local minimality in the sense that for $x \in \operatorname{dom} f$, $f(x) \le f(y)$ for all $y \in \mathbf{Z}^V$ if and only if $f(x) \le f(x - \chi_u + \chi_v)$ for all $u, v \in V$ [12,15,18].

The minimizers of an M-convex function, if nonempty, form an M-convex set. Moreover, for any

$p \in \mathbf{R}^V$, $\operatorname{argmin}(f[-p])$, if nonempty, is an M-convex set. Conversely, this property characterizes M-convex functions, under an auxiliary assumption that the effective domain is bounded or the function can be extended to a convex function over $\mathbf{R}^V$ (see [12,15]).

The level set of an M-convex function is not necessarily an M-convex set, but enjoys a weaker exchange property. Namely, for any $p \in \mathbf{R}^V$ and $\alpha \in \mathbf{R}$, $S = \{x \in \mathbf{Z}^V : f[-p](x) \leq \alpha\}$ (the level set of $f[-p]$) satisfies: For x, $y \in S$ and for $u \in \operatorname{supp}^+(x-y)$, there exists $v \in \operatorname{supp}^-(x-y)$ such that either $x - \chi_u + \chi_v \in S$ or $y + \chi_u - \chi_v \in S$. Conversely, this property characterizes M-convex functions [25].

A number of operations can be defined for M-convex functions [15,16]. For $p \in \mathbf{Z}^V$, $f[-p]$ is an M-convex function. For $a \in \mathbf{Z}^V$, $f(a-x)$ and $f(a+x)$ are M-convex in x. For $U \subseteq V$, the restriction of f to U:

$$f_U(x') = f(x', \mathbf{0}_{V \setminus U}) \quad (x' \in \mathbf{Z}^U)$$

(where $\mathbf{0}_{V \setminus U}$ is the zero vector in $\mathbf{Z}^{V \setminus U}$) is M-convex in x', provided that $\operatorname{dom} f_U \neq \emptyset$. For $\varphi_v \in \mathcal{C}_1$ $(v \in V)$,

$$\widetilde{f}(x) = f(x) + \sum_{v \in V} \varphi_v(x(v)) \quad (x \in \mathbf{Z}^V)$$

is M-convex, provided that $\operatorname{dom} \widetilde{f} \neq \emptyset$. In particular, a separable convex function $\widetilde{f}(x) = \sum_{v \in V} \varphi_v(x(v))$ with $\operatorname{dom} \widetilde{f}$ being an M-convex set is an M-convex function. For two M-convex functions f_1 and f_2, the integral convolution

$$(f_1 \square f_2)(x) = \inf\left\{ f_1(x_1) + f_2(x_2) : \begin{array}{c} x = x_1 + x_2 \\ x_1, x_2 \in \mathbf{Z}^V \end{array} \right\} \quad (x \in \mathbf{Z}^V)$$

is either M-convex or else $(f_1 \square f_2)(x) = \pm\infty$ for all $x \in \mathbf{Z}^V$.

Sum of two M-convex functions is not necessarily M-convex; such function with nonempty effective domain is called *M_2-convex*. Convolution of two L-convex functions is not necessarily L-convex; such function with nonempty effective domain is called *L_2-convex*. M_2- and L_2-convex functions are in one-to-one correspondence through the integral Fenchel–Legendre transformation.

$L^\natural$- and $M^\natural$-Convexity

$L^\natural$- and $M^\natural$-convexity are variants of, and essentially equivalent to, L- and M-convexity, respectively. $L^\natural$- and $M^\natural$-convex functions are introduced in [9] and [19], respectively.

Let v_0 be a new element not in V and define $\widetilde{V} = \{v_0\} \cup V$. A function $g: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ with $\operatorname{dom} g \neq \emptyset$ is called *$L^\natural$-convex* if it is expressed in terms of an L-convex function $\widetilde{g}: \mathbb{Z}^{\widetilde{V}} \to \mathbb{Z} \cup \{+\infty\}$ as $g(p) = \widetilde{g}(0, p)$. Namely, an $L^\natural$-convex function is a function obtained as the restriction of an L-convex function. Conversely, an $L^\natural$-convex function determines the corresponding L-convex function up to the constant r in the definition of L-convex function.

An $L^\natural$-convex function is essentially the same as a submodular integrally convex function of [5], and hence is characterized by discrete midpoint convexity [9]. An L-convex function, enjoying discrete midpoint convexity, is an $L^\natural$-convex function.

Quadratic function

$$g(p) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} p_i p_j \quad (p \in \mathbb{Z}^n)$$

with $a_{ij} = a_{ji} \in \mathbf{Z}$ is $L^\natural$-convex if and only if $a_{ij} \leq 0$ $(i \neq j)$ and $\sum_{j=1}^{n} a_{ij} \geq 0$ $(i = 1, \cdots, n)$. For $\{\psi_i \in \mathcal{C}_1 : i = 1, \ldots, n\}$, a separable convex function

$$g(p) = \sum_{i=1}^{n} \psi_i(p_i) \quad (p \in \mathbb{Z}^n)$$

is $L^\natural$-convex.

The properties of L-convex functions mentioned above are carried over, mutatis mutandis, to $L^\natural$-convex functions. In addition, the restriction of an $L^\natural$-convex function g to $U \subseteq V$, denoted g_U, is $L^\natural$-convex.

A subset of $\mathbf{Z}^V$ is called an *$L^\natural$-convex set* if its indicator function is an $L^\natural$-convex function. A set $E \subseteq \mathbf{Z}^V$ is an $L^\natural$-convex set if and only if

$$p, q \in E \quad \Rightarrow \quad \left\lceil \frac{p+q}{2} \right\rceil, \left\lfloor \frac{p+q}{2} \right\rfloor \in E.$$

A function $f: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ with $\operatorname{dom} f \neq \emptyset$ is called *$M^\natural$-convex* if it is expressed in terms of an M-convex function $\widetilde{f}: \mathbb{Z}^{\widetilde{V}} \to \mathbb{Z} \cup \{+\infty\}$ as

$$\widetilde{f}(x_0, x) = \begin{cases} f(x) & \text{if } x_0 + \sum_{u \in V} x(u) = 0 \\ +\infty & \text{otherwise.} \end{cases}$$

Namely, an $M^\natural$-convex function is a function obtained as the projection of an M-convex function. Conversely, an $M^\natural$-convex function determines the corresponding M-convex function up to a translation of dom f in the direction of v_0. A function $f: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ with dom $f \neq \emptyset$ is $M^\natural$-convex if and only if (see [19]) it satisfies

- $M^\natural$-EXC) For $x, y \in \mathrm{dom}\, f$ and $u \in \mathrm{supp}^+(x-y)$,

$$f(x)+f(y) \geq \min\Big[f(x-\chi_u)+f(y+\chi_u), \min_{v\in \mathrm{supp}^-(x-y)}\{f(x-\chi_u+\chi_v)+f(y+\chi_u-\chi_v)\}\Big].$$

Since M-EXC) implies $M^\natural$-EXC), an M-convex function is an $M^\natural$-convex function.

Quadratic function

$$f(x)=\sum_{i=1}^{n} a_i {x_i}^2 + b\sum_{i<j} x_i x_j \qquad (x \in \mathbb{Z}^n)$$

with $a_i \in \mathbf{Z}$ $(1 \leq i \leq n)$, $b \in \mathbf{Z}$ is $M^\natural$-convex if $0 \leq b \leq 2 \min_{1\leq i\leq n} a_i$ (cf. [19]). For $\{\varphi_i \in \mathcal{C}_1 : i = 0, \ldots, n\}$, a function of the form

$$f(x)=\varphi_0\left(\sum_{i=1}^{n} x_i\right)+\sum_{i=1}^{n}\varphi_i(x_i) \quad (x \in \mathbb{Z}^n)$$

is $M^\natural$-convex [19]; a separable convex function is a special case of this (with $\varphi_0 = 0$). More generally, for $\{\varphi_X \in \mathcal{C}_1 : X \in \mathcal{T}\}$ indexed by a laminar family $\mathcal{T} \subseteq 2^V$, the function

$$f(x)=\sum_{X\in\mathcal{T}} \varphi_X(x(X)) \quad (x \in \mathbb{Z}^V)$$

is $M^\natural$-convex [1], where $\mathcal{T}$ is called *laminar* if for any X, $Y \in \mathcal{T}$, at least one of $X \cap Y$, $X \setminus Y$, $Y \setminus X$ is empty.

The properties of M-convex functions mentioned above are carried over, mutatis mutandis, to $M^\natural$-convex functions. In addition, the projection of an $M^\natural$-convex function f to $U \subseteq V$, denoted f^U, is $M^\natural$-convex.

A subset of $\mathbf{Z}^V$ is called an *$M^\natural$-convex set* if its indicator function is an $M^\natural$-convex function. A set $Q \subseteq \mathbf{Z}^V$ is an $M^\natural$-convex set if and only if Q is the set of integer points of an integral generalized polymatroid (cf. [7] for generalized polymatroids).

As a consequence of the conjugacy between L- and M-convexity, $L^\natural$-convex functions and $M^\natural$-convex functions are conjugate to each other under the integral Fenchel–Legendre transformation.

Duality

Discrete duality theorems hold true for L-convex/concave and M-convex/concave functions. A function $g: \mathbf{Z}^V \to \mathbf{Z} \cup \{-\infty\}$ is called L-concave (respectively, $L^\natural$-, M-, or $M^\natural$-concave) if $-g$ is L-convex (respectively, $L^\natural$-, M-, or $M^\natural$-convex); dom g means the effective domain of $-g$. The concave counterpart of the discrete Fenchel–Legendre transform is defined as

$$g^\circ(p) = \inf\{\langle p, x\rangle - g(x) : x \in \mathbb{Z}^V\} \quad (p \in \mathbb{Z}^V).$$

A discrete separation theorem for L-convex/concave functions, named *L-separation theorem* [15] (see also [9]), reads as follows. Let $f: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ be an $L^\natural$-convex function and $g: \mathbf{Z}^V \to \mathbf{Z} \cup \{-\infty\}$ be an $L^\natural$-concave function such that dom $f \cap$ dom $g \neq \emptyset$ or dom $f^\bullet \cap$ dom $g^\circ \neq \emptyset$. If $f(p) \geq g(p)$ $(p \in \mathbf{Z}^V)$, there exist $\beta^* \in \mathbf{Z}$ and $x^* \in \mathbf{Z}^V$ such that

$$f(p) \geq \beta^* + \langle p, x^*\rangle \geq g(p) \quad (p \in \mathbb{Z}^V).$$

Since a submodular set function can be identified with a positively homogeneous L-convex function, the L-separation theorem implies *Frank's discrete separation theorem* for a pair of sub/supermodular functions [6], which reads as follows. Let $\rho: 2^V \to \mathbf{Z} \cup \{+\infty\}$ and $\mu: 2^V \to \mathbf{Z} \cup \{-\infty\}$ be submodular and supermodular functions, respectively, with $\rho(\emptyset) = \mu(\emptyset) = 0$, $\rho(V) < +\infty$, $\mu(V) > -\infty$, where μ is called *supermodular* if $-\mu$ is submodular. If $\rho(X) \geq \mu(X)$ $(X \subseteq V)$, there exists $x^* \in \mathbf{Z}^V$ such that

$$\rho(X) \geq x^*(X) \geq \mu(X) \quad (X \subseteq V).$$

Another discrete separation theorem, *M-separation theorem* [12,15] (see also [9]), holds true for M-convex/concave functions. Namely, let $f: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ be an $M^\natural$-convex function and $g: \mathbf{Z}^V \to \mathbf{Z} \cup \{-\infty\}$ be an $M^\natural$-concave function such that dom $f \cap$ dom $g \neq \emptyset$ or dom $f^\bullet \cap$ dom $g^\circ \neq \emptyset$. If $f(x) \geq g(x)$ $(x \in \mathbf{Z}^V)$, there exist $\alpha^* \in \mathbf{Z}$ and $p^* \in \mathbf{Z}^V$ such that

$$f(x) \geq \alpha^* + \langle p^*, x\rangle \geq g(x) \quad (x \in \mathbb{Z}^V).$$

The L- and M-separation theorems are conjugate to each other, while a self-conjugate statement can be made in the form of the *Fenchel-type duality* [12,15], as follows. Let $f: \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ be an $L^\natural$-convex function and $g: \mathbf{Z}^V \to \mathbf{Z} \cup \{-\infty\}$ be an $L^\natural$-concave function

such that $\mathrm{dom}\, f \cap \mathrm{dom}\, g \neq \emptyset$ or $\mathrm{dom}\, f^\bullet \cap \mathrm{dom}\, g^\circ \neq \emptyset$. Then it holds that

$$\inf\{f(p) - g(p) \colon\ p \in \mathbb{Z}^V\} = \sup\{g^\circ(x) - f^\bullet(x) \colon\ x \in \mathbb{Z}^V\}.$$

Moreover, if this common value is finite, the infimum is attained by some $p \in \mathrm{dom}\, f \cap \mathrm{dom}\, g$ and the supremum is attained by some $x \in \mathrm{dom}\, f^\bullet \cap \mathrm{dom}\, g^\circ$.

Example 3 Here is a simple example to illustrate the subtlety of discrete separation for discrete functions. Functions $f\colon \mathbf{Z}^2 \to \mathbf{Z}$ and $g\colon \mathbf{Z}^2 \to \mathbf{Z}$ defined by $f(x_1, x_2) = \max(0, x_1 + x_2)$ and $g(x_1, x_2) = \min(x_1, x_2)$ can be extended respectively to a convex function $\overline{f}\colon \mathbb{R}^2 \to \mathbb{R}$ and a concave function $\overline{g}\colon \mathbb{R}^2 \to \mathbb{R}$ according to the defining expressions. With $\overline{p} = (\frac{1}{2}, \frac{1}{2})$, we have $\overline{f}(x) \geq \langle \overline{p}, x\rangle \geq \overline{g}(x)$ for all $x \in \mathbf{R}^2$, and a fortiori, $f(x) \geq \langle \overline{p}, x\rangle \geq g(x)$ for all $x \in \mathbf{Z}^2$. However, there exists no integral vector $p \in \mathbf{Z}^2$ such that $f(x) \geq \langle p, x\rangle \geq g(x)$ for all $x \in \mathbf{Z}^2$. Note also that f is $\mathrm{M}^\natural$-convex and g is L-concave.

Network Duality

A conjugate pair of M- and L-convex functions can be transformed through a network ([12,16]; see also [23]). Let $G = (V, A)$ be a directed graph with arc set A and vertex set V partitioned into three disjoint parts as $V = V^+ \cup V^0 \cup V^-$. For $\varphi_a \in \mathcal{C}_1$ $(a \in A)$ and M-convex $f\colon \mathbf{Z}^{V^+} \to \mathbf{Z} \cup \{+\infty\}$, define $\widetilde{f}\colon \mathbb{Z}^{V^-} \to \mathbb{Z} \cup \{\pm\infty\}$ by

$$\widetilde{f}(y) = \inf_{\xi, x} \left\{ f(x) + \sum_{a \in A} \varphi_a(\xi(a)) \colon \begin{array}{c} \partial\xi = (x, 0, -y) \\ \in \mathbb{Z}^{V^+ \cup V^0 \cup V^-} \\ \xi \in \mathbb{Z}^A \end{array} \right\}.$$

For $\psi_a \in \mathcal{C}_1$ $(a \in A)$ and L-convex $g\colon \mathbf{Z}^{V^+} \to \mathbf{Z} \cup \{+\infty\}$, define $\widetilde{g}\colon \mathbb{Z}^{V^-} \to \mathbb{Z} \cup \{\pm\infty\}$ by

$$\widetilde{g}(q) = \inf_{\eta, p, r} \left\{ g(p) + \sum_{a \in A} \psi_a(\eta(a)) \colon \begin{array}{c} \eta = -\delta(p, r, q) \\ \eta \in \mathbb{Z}^A \\ (p, r, q) \in \mathbb{Z}^{V^+ \cup V^0 \cup V^-} \end{array} \right\}.$$

Then $\widetilde{f}$ is M-convex, provided that $\widetilde{f} > -\infty$, and $\widetilde{g}$ is L-convex, provided that $\widetilde{g} > -\infty$. If $g = f^\bullet$ and $\psi_a = \varphi_a^\bullet$ $(a \in A)$, then $\widetilde{g} = \widetilde{f}^\bullet$. A special case $(V^+ = V)$ of the last statement yields the network duality:

$$\inf \left\{ \Phi(x, \xi) \colon \begin{array}{c} \partial\xi = x, \\ x \in \mathbb{Z}^V, \\ \xi \in \mathbb{Z}^A \end{array} \right\} = \sup \left\{ \Psi(p, \eta) \colon \begin{array}{c} \eta = -\delta p, \\ p \in \mathbb{Z}^V, \\ \eta \in \mathbb{Z}^A \end{array} \right\},$$

where $\Phi(x, \xi) = f(x) + \sum_{a \in A} \varphi_a(\xi(a))$, $\Psi(p, \eta) = -g(p) - \sum_{a \in A} \psi_a(\eta(a))$ and the finiteness of $\inf \Phi$ or $\sup \Psi$ is assumed. The network duality is equivalent to the Fenchel-type duality.

Subdifferentials

The subdifferential of $f\colon \mathbf{Z}^V \to \mathbf{Z} \cup \{+\infty\}$ at $x \in \mathrm{dom}\, f$ is defined by $\{p \in \mathbf{R}^V \colon f(y) - f(x) \geq \langle p, y - x\rangle\ (\forall y \in \mathbf{Z}^V)\}$. The subdifferential of an L_2- or M_2-convex function forms an integral polyhedron. More specifically:

- The subdifferential of an L-convex function is an integral base polyhedron (an M-convex polyhedron).
- The subdifferential of an L_2-convex function is the intersection of two integral base polyhedra (M-convex polyhedra).
- The subdifferential of an M-convex function is an L-convex polyhedron.
- The subdifferential of an M_2-convex function is the Minkowski sum of two L-convex polyhedra.

Similar statements hold true with L and M replaced respectively by $\mathrm{L}^\natural$ and $\mathrm{M}^\natural$.

Algorithms

On the basis of the equivalence of $\mathrm{L}^\natural$-convex functions and submodular integrally convex functions, the minimization of an L-convex function can be done by the algorithm of [5], which relies on the ellipsoid method. The minimization of an M-convex function can be done by purely combinatorial algorithms; a greedy-type algorithm [2] for valuated matroids and a domain reduction-type polynomial time algorithm [24] for M-convex functions. Algorithms for duality of M-convex functions (in other words, for M_2-convex functions) are also developed; polynomial algorithms [14,22] for valuated matroids, and a finite primal algorithm [18] and a polynomial time conjugate-scaling algorithm [10] for the submodular flow problem.

Applications

A discrete analog of the conjugate duality framework [21] for nonlinear optimization is developed in [15]. An application of M-convex functions to engineering system analysis and matrix theory is in [13,17]. M-convex functions find applications also in mathematical economics [1].

See also

- ► Generalized Concavity in Multi-objective Optimization
- ► Invexity and its Applications
- ► Isotonic Regression Problems

References

1. Danilov V, Koshevoy G, Murota K (May 1998) Equilibria in economies with indivisible goods and money. RIMS Preprint Kyoto Univ 1204
2. Dress AWM, Wenzel W (1990) Valuated matroid: A new look at the greedy algorithm. Appl Math Lett 3(2):33–35
3. Dress AWM, Wenzel W (1992) Valuated matroids. Adv Math 93:214–250
4. Edmonds J (1970) Submodular functions, matroids and certain polyhedra. In: Guy R, Hanani H, Sauer N, Schönheim J (eds) Combinatorial Structures and Their Applications. Gordon and Breach, New York, pp 69–87
5. Favati P, Tardella F (1990) Convexity in nonlinear integer programming. Ricerca Oper 53:3–44
6. Frank A (1982) An algorithm for submodular functions on graphs. Ann Discret Math 16:97–120
7. Frank A, Tardos É (1988) Generalized polymatroids and submodular flows. Math Program 42:489–563
8. Fujishige S (1991) Submodular functions and optimization, vol 47. North-Holland, Amsterdam
9. Fujishige S, Murota K (2000) Notes on L-/M-convex functions and the separation theorems. Math Program 88:129–146
10. Iwata S, Shigeno M (1998) Conjugate scaling technique for Fenchel-type duality in discrete optimization. IPSJ SIG Notes 98-AL-65
11. Lovász L (1983) Submodular functions and convexity. In: Bachem A, Grötschel M, Korte B (eds) Mathematical Programming – The State of the Art. Springer, Berlin, pp 235–257
12. Murota K (1996) Convexity and Steinitz's exchange property. Adv Math 124:272–311
13. Murota K (1996) Structural approach in systems analysis by mixed matrices – An exposition for index of DAE. In: Kirchgässner K, Mahrenholtz O, Mennicken R (eds) ICIAM 95. Math Res. Akad. Verlag, Berlin, pp 257–279
14. Murota K (1996) Valuated matroid intersection, I: optimality criteria, II: algorithms. SIAM J Discret Math 9:545–561, 562–576
15. Murota K (1998) Discrete convex analysis. Math Program 83:313–371
16. Murota K (1998) Discrete convex analysis. In: Fujishige S (ed) Discrete Structures and Algorithms, vol V. Kindai-Kagaku-sha, Tokyo, pp 51–100 (In Japanese.)
17. Murota K (1999) On the degree of mixed polynomial matrices. SIAM J Matrix Anal Appl 20:196–227
18. Murota K (1999) Submodular flow problem with a nonseparable cost function. Combinatorica 19:87–109
19. Murota K, Shioura A (1999) M-convex function on generalized polymatroid. Math Oper Res 24:95–105
20. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton
21. Rockafellar RT (1974) Conjugate duality and optimization. SIAM Regional Conf Appl Math, vol 16. SIAM, Philadelphia
22. Shigeno M (1996) A dual approximation approach to matroid optimization problems. PhD Thesis Tokyo Inst. Techn.
23. Shioura A (1998) A constructive proof for the induction of M-convex functions through networks. Discrete Appl Math 82:271–278
24. Shioura A (1998) Minimization of an M-convex function. Discrete Appl Math 84:215–220
25. Shioura A (2000) Level set characterization of M-convex functions. IEICE Trans Fundam Electronics, Commun and Comput Sci E83-A:586–589

LCP: Pardalos–Rosen Mixed Integer Formulation

PANOS M. PARDALOS
Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 90C33, 90C11

Article Outline

Keywords
See also
References

Keywords

Linear complementarity problem; Mixed integer programming; Bimatrix games; Mixed integer problem; Minimum norm solution

In this article we consider the general *linear complementarity problem* (LCP) of finding a vector $x \in \mathbf{R}^n$ such that

$$Mx + q \geq 0, \quad x \geq 0, \quad x^\top Mx + q^\top x = 0$$

(or proving that such an x does not exist), where M is an $n \times n$ rational matrix and $q \in \mathbf{R}^n$ is a rational vector. For given data M and q, the problem is generally denoted by LCP(M, q). The LCP unifies a number of important problems in operations research. In particular, it generalizes the primal-dual linear programming problem, convex quadratic programming, and *bimatrix games* [1,2].

For the general matrix M, where $S = \{x: Mx + q \geq 0, x \geq 0\}$ can be bounded or unbounded, the LCP can always be solved by solving a specific zero-one, linear, *mixed integer problem* with n zero-one variables. Consider the following mixed zero-one integer problem:

$$\text{(MIP)} \quad \begin{cases} \max\limits_{\alpha,y,z} & \alpha \\ \text{s.t.} & 0 \leq My + \alpha q \leq e - z, \\ & \alpha \geq 0, \quad 0 \leq y \leq z, \\ & z \in \{0,1\}^n. \end{cases}$$

Theorem 1 *Let (α^*, y^*, z^*) be any optimal solution of (MIP). If $\alpha^* > 0$, then $x^* = y^*/\alpha^*$ solves the LCP. If in the optimal solution $\alpha^* = 0$, then the LCP has no solution.*

The equivalent mixed integer programming formulation (MIP) was first given in [3]. Every feasible point (α, y, z) of (MIP), with $\alpha > 0$, corresponds to a solution of LCP. Therefore, solving (MIP), we may generate several solutions of the corresponding LCP. J.B. Rosen [4] proved that the solution obtained by solving (MIP) is the *minimum norm solution* to the linear complementarity problem.

See also

- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Convex-simplex Algorithm
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Equivalence Between Nonlinear Complementarity Problem and Fixed Point Problem
- ▶ Generalized Nonlinear Complementarity Problem
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ Lemke Method
- ▶ Linear Complementarity Problem
- ▶ Linear Programming
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Order Complementarity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Principal Pivoting Methods for Linear Complementarity Problems
- ▶ Sequential Simplex Method
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem
- ▶ Topological Methods in Complementarity Theory

References

1. Cottle RW, Dantzig GB (1968) Complementarity pivot theory of mathematical programming. In: Dantzig GB, Veinott AF (eds) Mathematics of the Decision Sci., Part 1. Amer. Math. Soc., Providence, RI, pp 115–136
2. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
3. Pardalos PM, Rosen JB (1988) Global optimization approach to the linear complementarity problem. SIAM J Sci Statist Comput 9(2):341–353
4. Rosen JB (1990) Minimum norm solution to the linear complementarity problem. In: Leifman LJ (ed) Functional Analysis, Optimization and Mathematical Economics. Oxford Univ. Press, Oxford, pp 208–216

Least-index Anticycling Rules

LindAcR

TAMÁS TERLAKY
Department Comput. & Software,
McMaster University, West Hamilton, Canada

MSC2000: 90C05, 90C33, 90C20, 05B35

Article Outline

Keywords
Consistent Labeling For the Max-Flow Problem
Linear Optimization
  Least-Index Rules for Feasibility Problem
  The Linear Optimization Problem
  Least-Index Pivoting Methods for LO
Linear Complementarity Problems
Least-Index Rules and Oriented Matroids
See also
References

Keywords

Pivot rules; Anticycling; Least-index; Recursion; Oriented matroids

From the early days of mathematical optimization people were looking for simple rules that ensure that certain algorithms terminate in a finite number of steps. Specifically, on combinatorial structures the lack of finite termination imply that the algorithm cycles, i. e. periodically visits the same solutions. That is why rules ensuring finite termination of algorithms on finite structures are frequently referred to as *anticycling rules*.

One frequently used anticycling rule in linear optimization (cf. ► Linear programming) is the so-called *lexicographic pivoting rule* [9]. The other large class of anticycling procedures, the 'least-index' rules, is the subject of this paper. least-index rules were designed for network flow problems, linear optimization problems, linear complementarity problems and oriented matroid programming problems. These classes will be considered in the sequel.

Consistent Labeling For the Max-Flow Problem

The *maximal flow problem* (see e. g. [11]; [24]) is one of the basic problems of mathematical programming. The problem is given as follows. A *directed capacitated network* (N, A, u) is given, where N, the set of nodes, is a finite set; $A \subset N \times N$ is the set of directed arcs; finally, $u \in \mathbf{R}^A$ denotes the nonnegative capacity upper bound for flows through the arcs. Let further $s, t \in N$ be specified as the source and the sink in the network. A vector $f \in \mathbf{R}^A$ is a *flow* in the network, if the incoming flow at each node, different from the source and the sink, is equal to the flow going out from the node. The goal is to find a maximal flow, namely a flow for which the total flow flowing out of the source or, equivalently, flowing in to the sink is the largest possible. The *Ford–Fulkerson algorithm* is the best known algorithm to find such a maximal flow. It is based on generating augmenting path's subsequently. A *path* P connecting the source s and the sink t is a finite subset of arcs, where the source is the tail of the first arc; the sink is the head of the last arc; finally, the tail of an arc is always equal to the head of its predecessor. For ease of simplicity let us assume that if $(v^1, v^2) \in A$, then $(v^2, v^1) \in A$ as well. If the opposite arc were not present, we can introduce it with zero capacity.

0	Initialization. Let f be equal to zero. Let a free capacity network $(N, \overline{A}, \overline{u})$ be defined. Initially let $\overline{A} = \{a \in A : u_a > 0\}$ and $\overline{u} = u$.
1	Augmenting path. Let P be a path from s to t in the free capacity network. IF no such path exists, THEN STOP; A maximal flow is obtained.
2	Augmenting the flow. Let ϑ be the minimum of the arc capacities along the path P. Clearly $\vartheta > 0$. Increase the flow f on each arc of P by ϑ.
3	Update the free-capacity network. Decrease (increase) $\overline{u}_a$ by ϑ if the (opposite) of arc a is on the path P. Let $\overline{A} = \{a \in A : \overline{u}_a > 0\}$. Go to Step 1.

The Ford–Fulkerson max-flow algorithm

At each iteration cycle the flow value strictly increases. Thus, if the vector u is integral and the max-flow problem is bounded, then the Ford–Fulkerson algorithm provides a maximal flow in a finite number of steps. However, if the vector u contains irrational com-

ponents, then the algorithm does not terminate in a finite number of steps and, even worse, it might converge to a nonoptimal flow. For such an example see [11,24]. An elegant solution for this problem is the *consistent labeling algorithm* of A.W. Tucker [28]. This most simple refinement reads as follows:

> Be consistent at any time during the algorithm, specifically when building the augmenting path by using the *labeling procedure*. Whenever a labeled but unscanned subset of nodes is given during the procedure pick always the same from the same subset to be scanned.
> Particularly, if we assign an index to each node, then we are supposed to choose always the least-indexed node among the possibilities.

Tucker writes [28]: 'Fulkerson (unpublished) conjectured that a consistent labeling procedure would be polynomially bounded; a proof of this conjecture appears to be very difficult.'

Linear Optimization

Before discussing the general LO problem, first the linear *feasibility problem* is considered.

0	Initialization. Let $T(B)$ be an arbitrary basis tableau and fix an arbitrary ordering of the variables.
1	Leaving variable selection. Let K_P be the set of the indices of the infeasible variables in the basis. IF $K_P = \emptyset$, THEN STOP; the feasibility problem is solved. ELSE, let p be the least-index in K_P and then x_p will leave the basis.
2	Entering variable selection. Let K_D be the set of the column indices of the negative elements in row p of $T(B)$. IF $K_D = \emptyset$, THEN STOP; Row p of the tableau $T(B)$ gives an evidence that the feasibility problem is inconsistent and row p of the inverse basis is a solution of the alternative system. ELSE, let q be the least-index in K_D and then x_q will enter the basis.
3	Basis transformation. Pivot on (p, q). Go to Step 1.

Pivot rule

Least-Index Rules for Feasibility Problem

The feasibility problem

$$Ax = b, \quad x \geq 0,$$

and its alternative pair

$$b^\top y > 0, \quad A^\top y \leq 0,$$

can be solved by a very simple least-index pivot algorithm. A fundamental result, the so-called *Farkas lemma* (cf. also ▶ Farkas lemma; ▶ Farkas lemma: Generalizations) [10] says that exactly one of the two alternative systems has a solution. This result is also known as the *theorem of the alternatives*. When a simple finite pivot rule gives a solution to either of the two alternatives, an elementary constructive proof for the Farkas lemma and its relatives is obtained. The above simple finite least-index pivot rule for the feasibility problem is a special case (see below) of Bland's algorithm [5]. It is taken from [19] where the role of pivoting, and specifically the role of finite, least-index pivot rules in linear algebra is explored.

The Linear Optimization Problem

The general linear optimization (LO), linear programming (cf. ▶ Linear programming), problem will be considered in the standard primal form

$$\min \left\{ c^\top x : \; Ax = b, \; x \geq 0 \right\},$$

together with its standard dual

$$\max \left\{ b^\top y : \; A^\top y \leq c \right\}.$$

One of the most efficient, and for a long time the only, practical method to solve LO problems was the simplex method of G.B. Dantzig. The *simplex method* is a *pivot algorithm* that traverses through feasible *basic solutions* while the objective value is improving. The simplex method is in practice one of the most efficient algorithms but it is theoretically a finite algorithm only for *nondegenerate problems*.

A basis is called *primal degenerate* if at least one of the basic variables is zero; it is called *dual degenerate* if the reduced cost of at least one nonbasic variable is zero. In general, the basis is degenerate if it is either primal or dual, or both primal and dual degenerate. The LO problem is degenerate, if it has a degenerate ba-

sis. A pivot is called degenerate when after the pivot the objective remains unchanged. When the problem is *degenerate* the objective might stay the same in subsequent iterations and the simplex algorithm may *cycle*, i. e. starting from a basis, after some iterations the same basis is revisited and this process is repeated endlessly. Because the simplex method produces a sequence with monotonically improving objective values, the objective stays constant in a cycle, thus each pivot in the cycle must be degenerate. The possibility of cycling was recognized shortly after the invention of the simplex algorithm. Cycling examples were given by E.M.L. Beale [2] and by A.J. Hoffman [17]. Recently (1999) a scheme to construct cycling LO examples is presented in [15]. These examples made evident that extra techniques are needed to ensure finite termination of simplex methods. The first and widely used such tool is the class of lexicographic pivoting rules (cf. ► Lexicographic pivoting rules). Other, more recent techniques are the *least-index anticycling rules* and some more general recursive schemes.

Least-Index Pivoting Methods for LO

Cycling of the simplex method is possible only when the LO problem is degenerate. In that case not only many variables might be eligible to enter, but also to leave the basis. The least-index primal simplex rule makes the selection of both the entering and the leaving variable uniquely determined. Least-index rules are based on consistent selection among the possibilities. The first such rule for the simplex method was published by R.G. Bland [4,5].

The least-index simplex method is finite. The finiteness proofs are quite elementary. All are based on the simple fact that there is a finite number of different basis tableaus. Further, orthogonality of the primal and dual spaces on some recursive argumentation is used [4, 5,27]

It is straightforward to derive the least-index dual simplex algorithm. The only restriction relative to the dual simplex algorithm is, that when there are more candidates to leave or to enter the basis, always the least-indexed candidate has to be selected.

An interesting use of least index-resolution is used in [18] by designing finite primal-dual type Hungarian methods for LO. Note that finite criss-cross rules (cf. also ► Criss-cross pivoting rules) [14,26] make maximum possible use of least-index resolution.

0	Initialization Let $T(B)$ be a given primal feasible basis tableau and fix an arbitrary ordering of the variables.
1	Entering variable selection. Let K_D be the set of the indices of the dual infeasible variables, i.e. those with negative reduced cost. IF $K_D = \emptyset$, THEN STOP; The tableau $T(B)$ is optimal and this way a pair of solutions is obtained. ELSE, let q be the least-index in K_D and x_q, will enter the basis.
2	Leaving variable selection. Let K_P be the set of the indices of those candidate pivot elements in column q that satisfy the usual pivot selection conditions of the primal simplex method. IF $K_P = \emptyset$, THEN STOP; the primal problem is unbounded, and so the dual problem is infeasible. ELSE, let p be the least-index in K_P and then x_p will leave the basis.
3	Basis transformation. Pivot on (p, q). Go to Step 1.

The least-index primal simplex rule

Least-index simplex methods are not polynomial, they might require exponential number of steps to solve a LO problem, as it was shown by D. Avis and V. Chvátal [1]. Their example is essentially the Klee–Minty polytope [21]. Another example, again on the Klee–Minty polytope, is Roos's exponential example [25] for the least-index criss-cross method. Here the initial basis is feasible and, although it is not required, feasibility happens to be preserved, thus the criss-cross method reduces to a least index simplex method.

Linear Complementarity Problems

A linear complementarity problem (cf. ► Linear complementarity problem) (LCP) is given as follows:

$$-Mx + s = t, \quad x, s \geq 0, \quad x^\top s = 0.$$

Pivot algorithms are looking for a *complementary basis* solution of the LCP. A basis is called complementary, if

exactly one of the complementary variables x_i and s_i for all i is in the basis.

The solvability of LCP depends on the properties of the matrix M. One of the simplest case is when M is a *P-matrix*. The matrix M is a P-matrix if all of its principal minors are positive. K.G. Murty [22] presented an utmost simple finite pivot algorithm for solving the *P*-matrix LCP. This algorithm is a least-index *principal pivot algorithm*.

Two extremal behaviors, exponential in the worst case and polynomial in average, of this finite pivot rule is studied in [13].

Finite least-index pivot rules are developed for larger classes of LCPs. All are least-index principal pivoting methods, some more classical feasibility preserving simplex type methods [7,8,23], others are least-index criss-cross pivoting rules (cf. ► Criss-cross pivoting rules) [6,16,20]. More details are given in ► Principal pivoting methods for linear complementarity problems.

0	Initialization. Let $T(B)$ be complementary basis tableau and fix an arbitrary ordering of the variables. (We can choose $x = 0$, $s = t$ i.e., x nonbasic, s basic.)
1	Leaving variable selection. Let K be the set of the infeasible variables. IF $K = \emptyset$, THEN STOP; a complementary solution for LCP is obtained. ELSE, let p be the least-index in K.
2	Basis transformation. Pivot on (p, p), i.e. replace the least-indexed infeasible variable in the basis by its complementary pair. Go to Step 1.

Murty's Bard-type schema

Least-Index Rules and Oriented Matroids

The least-index simplex method was originally designed for oriented matroid linear programming (cf. also ► Oriented matroids) [3,4]. It turned soon out, that this is *not* a finite algorithm in the oriented matroid context. The reason is the possibility of *nondegenerate cycling* [3,12], a phenomenon what is impossible in the linear case. An apparent difference between the linear and the oriented matroid context is that for oriented matroids none of the finite-, recursive- or least-index-type rules yield a simplex method, i. e. a pivot method that preserves feasibility of the basis throughout. This discrepancy is also due to the possibility of nondegenerate cycling.

See also

- ► Criss-cross Pivoting Rules
- ► Lexicographic Pivoting Rules
- ► Linear Programming
- ► Pivoting Algorithms for Linear Programming Generating Two Paths
- ► Principal Pivoting Methods for Linear Complementarity Problems
- ► Probabilistic Analysis of Simplex Algorithms

References

1. Avis D, Chvátal V (1978) Notes on Bland's rule. Math Program Stud 8:24–34
2. Beale EML (1955) Cycling in the dual simplex algorithm. Naval Res Logist Quart 2:269–275
3. Bjorner A, Las Vergnas M, Sturmfels B, White N, Ziegler G (1993) Oriented matroids. Cambridge Univ. Press, Cambridge
4. Bland RG (1977) A combinatorial abstraction of linear programming. J Combin Th B 23:33–57
5. Bland RG (1977) New finite pivoting rules for the simplex method. Math Oper Res 2:103–107
6. Chang YY (1979) Least index resolution of degeneracy in linear complementarity problems. Techn Report Dept Oper Res Stanford Univ 79-14
7. Chang YY, Cottle RW (1980) Least index resolution of degeneracy in quadratic programming. Math Program 18:127–137
8. Cottle R, Pang JS, Stone RE (1992) The linear complementarity problem. Acad. Press, New York
9. Dantzig GB (1963) Linear programming and extensions. Princeton Univ. Press, Princeton
10. Farkas J (1902) Theorie der Einfachen Ungleichungen. J Reine Angew Math 124:1–27
11. Ford LR Jr, Fulkerson DR (1962) Network flows. Princeton Univ. Press, Princeton
12. Fukuda K (1982) Oriented matroid programming. PhD Thesis Waterloo Univ.
13. Fukuda K, Namiki M (1994) On extremal behaviors of Murty's least index method. Math Program 64:365–370
14. Fukuda K, Terlaky T (1997) Criss-cross methods: A fresh view on pivot algorithms. In: Mathematical Programming, (B) Lectures on Mathematical Programming, ISMP97, vol 79. Lausanne, pp 369–396

15. Hall J, McKinnon KI (1998) A class of cycling counter-examples to the EXPAND anti-cycling procedure. Techn. Report Dept. Math. and Statist. Univ. Edinburgh
16. Den Hertog D, Roos C, Terlaky T (1993) The linear complementarity problem, sufficient matrices and the criss-cross method. LAA 187:1–14
17. Hoffman AJ (1953) Cycling in the simplex method. Techn Report Nat Bureau Standards 2974
18. Klafszky E, Terlaky T (1989) Variants of the Hungarian method for solving linear programming problems. Math Oper Statist Ser Optim 20:79–91
19. Klafszky E, Terlaky T (1991) The role of pivoting in proving some fundamental theorems of linear algebra. LAA 151:97–118
20. Klafszky E, Terlaky T (1992) Some generalizations of the criss-cross method for quadratic programming. Math Oper Statist Ser Optim 24:127–139
21. Klee V, Minty GJ (1972) How good is the simplex algorithm? In: Shisha O (ed) Inequalities-III. Acad. Press, New York, pp 1159–1175
22. Murty KG (1974) A note on a Bard type scheme for solving the complementarity problem. Oper Res 11(2–3):123–130
23. Murty KG (1988) Linear complementarity, linear and nonlinear programming. Heldermann, Berlin
24. Murty KG (1992) Network programming. Prentice-Hall, Englewood Cliffs, NJ
25. Roos C (1990) An exponential example for Terlaky's pivoting rule for the criss-cross simplex method. Math Program 46:78–94
26. Terlaky T (1985) A convergent criss-cross method. Math Oper Statist Ser Optim 16(5):683–690
27. Terlaky T, Zhang S (1993) Pivot rules for linear programming: A survey on recent theoretical developments. Ann Oper Res 46:203–233
28. Tucker A (1977) A note on convergence of the Ford–Fulkerson flow algorithm. Math Oper Res 2(2):143–144

Least Squares Orthogonal Polynomials

Claude Brezinski, Ana C. Matos
Lab. d'Anal. Numérique et d'Optimisation,
Université Sci. et Techn. Lille Flandres–Artois,
Lille, France

MSC2000: 33C45, 65K10, 65F20, 65F22

Article Outline

Keywords
Existence and Uniqueness
Computation
Location of the Zeros
Applications
See also
References

Keywords

Orthogonal polynomials; Least squares; Padé-type approximation; Quadrature methods

Let c be the linear functional on the space of complex polynomials defined by

$$c(x^i) = \begin{cases} c_i \in \mathbb{C}, & i = 0, 1, \dots, \\ 0, & i < 0. \end{cases}$$

It is said that $\{P_k\}$ forms a family of (formal) *orthogonal polynomials* with respect to c if $\forall k$:

- P_k has the exact degree k,
- $c(x^i P_k(x)) = 0$ for $i = 0, \dots, k-1$.

Such a family exists if, $\forall k$, the Hankel determinant

$$H_k^{(0)} = \begin{vmatrix} c_0 & c_1 & \cdots & c_{k-1} \\ c_1 & c_2 & \cdots & c_k \\ \cdots & \cdots & \cdots & \cdots \\ c_{k-1} & c_k & \cdots & c_{2k-2} \end{vmatrix}$$

is different from zero. Such polynomials enjoy most of the properties of the usual orthogonal polynomials, when the functional c is given by

$$c(x^i) = \int_a^b x^i \, d\alpha(x),$$

where α is bounded and non decreasing in $[a, b]$ (see [1] for these properties). In this paper we study the polynomials R_k such that

$$\sum_{i=0}^{m} \left[c(x^i R_k(x))\right]^2$$

is minimized, where m is an integer strictly greater than $k-1$ (since, for $m = k-1$, we recover the previous formal orthogonal polynomials) and which can possibly depend on k. They will be called *least squares (formal) orthogonal polynomials*. They depend on the value of m but for simplicity this dependence will not be indicated in our notations.

Such polynomials arise naturally in problems of *Padé approximation* for power series with perturbed

coefficients, and in *Gaussian quadrature* (as described in the last section). Some properties of these polynomials are derived, together with a recursive scheme for their computation.

Existence and Uniqueness

Since the polynomials R_k will be defined apart from a multiplying factor, and since it is asked that the degree of R_k is exactly k we shall write

$$R_k(x) = b_0 + b_1 x + \cdots + b_k x^k$$

with $b_k = 1$. We set

$$\Phi(b_0, \ldots, b_{k-1}) = \sum_{i=0}^{m} [c(x^i R_k(x))]^2$$

and we seek for the values of $b_0, \ldots, b_{k-1}$ that minimize this quantity. That is, such that

$$\frac{\partial \Phi}{\partial b_j} = 0 \quad \text{for } j = 0, \ldots, k-1. \tag{1}$$

Setting $\gamma_n = (c_n, \ldots, c_{n+m})^\mathsf{T}$, this system can be written

$$b_0(\gamma_0, \gamma_j) + \cdots + b_{k-1}(\gamma_{k-1}, \gamma_j) = -(\gamma_k, \gamma_j) \tag{2}$$

for $j = 0, \ldots, k-1$. Thus R_k exists and is unique if and only if the matrix A_k of this system is non singular. Setting $X = (1, x, \ldots, x^{k-1})$ and calling the right-hand side of the preceding system γ we see that

$$R_k(x) = \frac{\begin{vmatrix} A_k & \gamma \\ X & x^k \end{vmatrix}}{|A_k|}.$$

If we set

$$B_k = \begin{pmatrix} c_0 & \cdots & c_{k-1} \\ \cdots & \cdots & \cdots \\ c_m & \cdots & c_{m+k-1} \end{pmatrix},$$

then $A_k = B_k^\top B_k$, $\gamma = B_k^\top \gamma_k$ and we recover the usual solution of a system of linear equations in the *least squares* sense.

Computation

The polynomials R_k can be recursively computed by inverting the matrix A_k of the above system (2) by the *bordering method*, see [5]. This method is as follows. Set

$$A_{k+1} = \begin{pmatrix} A_k & u_k \\ v_k & a_k \end{pmatrix}$$

where u_k is a column vector, v_k a row vector and a_k a scalar. We then have

$$A_{k+1}^{-1} = \begin{pmatrix} A_k^{-1} + A_k^{-1} u_k \beta_k^{-1} v_k A_k^{-1} & -A_k^{-1} u_k \beta_k^{-1} \\ -\beta_k^{-1} v_k A_k^{-1} & \beta_k^{-1} \end{pmatrix},$$

where $\beta_k = a_k - v_k A_k^{-1} u_k$.

Instead of choosing the normalization $b_k = 1$ we could impose the condition $b_0 = 1$. In that case we have the system

$$b_1'(\gamma_1, \gamma_j) + \cdots + b_k'(\gamma_k, \gamma_j) = -(\gamma_0, \gamma_j) \tag{3}$$

for $j = 1, \ldots, k$, and the bordering method can be used not only for computing the inverses of the matrices of the system recursively but also for obtaining its solution, since the new right-hand side contains the previous one.

Let $A_k{}'$ be the matrix of (3) and $d_k{}'$ be the right-hand side. We then have

$$A'_{k+1} = \begin{pmatrix} A'_k & u'_k \\ v'_k & a'_k \end{pmatrix}, \qquad d'_{k+1} = \begin{pmatrix} d'_k \\ f'_k \end{pmatrix}$$

with

$$u'_k = ((\gamma_{k+1}, \gamma_1), \ldots, (\gamma_{k+1}, \gamma_k))^\top;$$
$$v'_k = ((\gamma_1, \gamma_{k+1}), \ldots, (\gamma_k, \gamma_{k+1}));$$
$$a'_k = (\gamma_{k+1}, \gamma_{k+1});$$
$$d'_k = ((\gamma_0, \gamma_1), \ldots, (\gamma_0, \gamma_k))^\top;$$
$$f'_k = (\gamma_0, \gamma_{k+1}).$$

Setting $z_k{}' = (b_1{}', \ldots, b_k{}')^\mathsf{T}$ we have

$$z'_{k+1} = \begin{pmatrix} z'_k \\ 0 \end{pmatrix} + \frac{f'_k - v'_k z'_k}{\beta'_k} \begin{pmatrix} -A_k'^{-1} u'_k \\ 1 \end{pmatrix}$$

with $\beta_k{}' = a_k{}' - v_k{}' A_k'^{-1} u_k{}'$.

Of course the bordering method can only be used if β_k (or $\beta_k{}'$ in the second case) is different from zero. If it is not the case, instead of adding one new row and one new column to the system it is possible to add several rows and columns until a non singular β_k (which is now a square matrix) has been found (see [3] and [4]).

Location of the Zeros

We return to the normalization $b_k = 1$. As

$$c(x^i R_k(x)) = b_0 c_i + \cdots + b_k c_{i+k}$$

and $\partial c(x^i R_k(x))/\partial b_j = c_{i+j}$, from (1) we obtain

$$\sum_{i=0}^{m} c(x^i R_k(x)) c_{i+j} = 0 \quad \text{for } j = 0, \ldots, k-1. \quad (4)$$

This relation can be written as

$$c(R_k(x)(c_i + c_{i+1}x + \cdots + c_{i+m}x^m)) = 0,$$

for $i = 0, \ldots, k-1$. Let us now assume that

$$c_i = \int_a^b x^i d\alpha(x), \quad i = 0, 1, \ldots,$$

with α bounded and nondecreasing in $[a, b]$. We have

$$\begin{aligned} & c_i + c_{i+1}x + \cdots + c_{i+m}x^m \\ &= \sum_{j=0}^{m} \left[\int_a^b y^{i+j}\, d\alpha(y) \right] x^j \\ &= \int_a^b y^i \left(\sum_{j=0}^{m} x^j y^j \right) d\alpha(y). \end{aligned}$$

Set

$$w(x, \mu) = \int_a^b y^\mu \left(\sum_{j=0}^{m} x^j y^j \right) d\alpha(y).$$

Thus

$$w(x, i) = c_i + c_{i+1}x + \cdots + c_{i+m}x^m$$

and it follows that

$$c(R_k(x)w(x, i)) = \int_a^b R_k(x)w(x, i)\, d\alpha(x) = 0$$

for $i = 0, \ldots, k-1$, which shows that the polynomial R_k is *biorthogonal* in the sense of [7,8]. Let us now study the location of the zeros of R_k. For that purpose we shall apply [7, Thm. 3], also given as [8, Thm. 5]. Set

$$d\Phi(x, \mu) = w(x, \mu) d\alpha(x)$$

and

$$I_k(\mu) = \int_a^b x^k\, d\Phi(x, \mu), \quad k = 0, 1, \ldots.$$

In our case, μ takes the values $\mu_i = i - 1$, $i = 1, 2, \ldots$. Thus

$$\det\left[I_i(\mu_j)\right] = \det\left[(\gamma_{j-1}, \gamma_i)\right]$$

and the condition of regularity of [7,8] is equivalent to our condition for the existence and uniqueness of R_k. According to [7,8], we now have to look at the interpolation property of w. We have

$$w(x_i, \mu_j) = (\gamma_{j-1}, X_i)$$

where $X_i = (1, x_i, \ldots, x_i^m)^\top$, the x_i's being arbitrary distinct points in $[a, b]$, and thus

$$\begin{vmatrix} (\gamma_0, X_1) & \cdots & (\gamma_{k-1}, X_1) \\ \cdots & \cdots & \cdots \\ (\gamma_0, X_k) & \cdots & (\gamma_{k-1}, X_k) \end{vmatrix} = \det(\mathcal{X}_k \Gamma_k)$$

with

$$\mathcal{X}_k = \begin{pmatrix} X_1^\top \\ \vdots \\ X_k^\top \end{pmatrix} \quad \text{and} \quad \Gamma_k = (\gamma_0, \ldots, \gamma_{k-1}).$$

The interpolation property holds if and only if $\det(\mathcal{X}_k \Gamma_k) \neq 0$, that is, if and only if the matrix $\mathcal{X}_k \Gamma_k$ has rank k. Thus, using the theorem of [7,8], we have proved the following result:

Theorem 1 *If A_k is regular and if $\mathcal{X}_k \Gamma_k$ has rank k, then R_k exists and has k distinct zeros in $[a, b]$.*

Remark 2 When $0 \leq a < b$, it can be proved that $\det(\mathcal{X}_k \Gamma_k) \neq 0$ (see [2] for the details).

Applications

Our first application deals with *Padé-type approximation*. Let v_k be an arbitrary polynomial of degree k and let $w_k(t) = a_0 + \cdots + a_{k-1}t^{k-1}$ be defined by

$$a_i = c(x^{-i-1} v_k(x)), \quad i = 0, \ldots, k-1.$$

We set

$$\widetilde{v}_k(t) = t^k v_k(t^{-1}) \quad \text{and} \quad \widetilde{w}_k(t) = t^{k-1} w_k(t^{-1}).$$

Let f be the formal power series

$$f(t) = \sum_{i=0}^{\infty} c_i t^i.$$

Then it can be proved that

$$f(t) - \frac{\widetilde{w}_k(t)}{\widetilde{v}_k(t)} = O(t^k) \quad (t \to 0).$$

The rational function $\frac{\widetilde{w}_k(t)}{\widetilde{v}_k(t)}$ is called a Padé-type approximant of f and it is denoted by $(k-1/k)_f(t)$, [1]. Moreover it can also be proved that

$$f(t) - \frac{\widetilde{w}_k(t)}{\widetilde{v}_k(t)} = \frac{t^k}{\widetilde{v}_k(t)} c\left(\frac{v_k(x)}{1-xt}\right) = \frac{t^k}{\widetilde{v}_k(t)} \cdot c\left(\left(1 + xt + \cdots + x^{k-1}t^{k-1} + \frac{x^k t^k}{1-xt}\right) v_k(x)\right).$$

That is,

$$f(t)\widetilde{v}_k(t) - \widetilde{w}_k(t) = t^k \sum_{i=0}^{\infty} c(x^i v_k(x)) t^i.$$

Thus if the polynomial v_k, which is called the *generating polynomial* of $(k-1/k)$, satisfies

$$c(x^i v_k(x)) = 0 \quad \text{for } i = 0, \ldots, k-1,$$

then

$$f(t) - \frac{\widetilde{w}_k(t)}{\widetilde{v}_k(t)} = O(t^{2k}).$$

In this case v_k is the formal orthogonal polynomial P_k of degree k with respect to c and $\frac{\widetilde{w}_k(t)}{\widetilde{v}_k(t)}$ is the usual Padé approximant $[k-1/k]$ of f.

As explained in [10], Padé approximants can be quite sensitive to perturbations on the coefficients c_i of the series f. Hence the idea arises to take as v_k the least squares orthogonal polynomial R_k of degree k instead of the usual orthogonal polynomial, an idea which in fact motivated our study. Of course such a choice decreases the degree of approximation, since the approximants obtained are only of the Padé-type, but it can increase the stability properties of the approximants and also their precision since $\sum_{i=0}^{m} [c(x^i v_k(x))]^2$ is minimized by the choice $v_k = R_k$. We give a numerical example that illustrates this fact.

We consider the function

$$f(z) = \frac{\ln(1+z)}{z} = \sum_{i=0}^{\infty} c_i z^i$$

and we assume that we know the coefficients c_i with a certain precision. For example, we know approximate values c_i^* such that

$$\left|c_i - c_i^*\right| \le 10^{-8}, \quad i = 0, 1, \ldots.$$

In the following table we compare the number of exact figures given by the Padé approximant with those of the least squares Padé-type approximant, both computed with the same number of coefficients c_i^*. We can see that the least squares Padé-type approximant has better stability properties.

z	Padé approx [7/8]	LS Padé-type approx [6/7] ($m = 8$)
1.5	6.7	7.7
1.9	5.7	7.0
2.1	5.2	6.7

Another application concerns quadrature methods. We have already shown that if the functional c is given by

$$c_i = \int_a^b x^i \, d\alpha(x), \quad i = 0, 1, \ldots, \quad 0 \le a < b,$$

with α bounded and nondecreasing, then the corresponding least squares orthogonal polynomial of degree k, R_k, has k distinct zeros in $[a, b]$. We can then construct quadrature formulas of the interpolatory type.

If $\lambda_1, \ldots, \lambda_k$ are the zeros of R_k, we can approximate the integral

$$I = \int_a^b f(x) \, d\alpha(x)$$

by

$$I_k = A_1 f(\lambda_1) + \cdots + A_k f(\lambda_k) \tag{5}$$

where

$$A_i = \int_a^b \frac{\pi(x)}{\pi'(\lambda_i)(x - \lambda_i)} \, d\alpha(x)$$

and

$$\pi(x) = \prod_{j=1}^{k} (x - \lambda_j).$$

This corresponds to replacing the function f by its interpolating polynomial at the knots $\lambda_1, \ldots, \lambda_k$. The truncation error of (5) is given by

$$I - I_k = E_T = \int_a^b f[\lambda_1, \cdots, \lambda_k, x] R_k(x) \, d\alpha(x).$$

Expanding the divided difference we see

$$f[\lambda_1,\ldots,\lambda_k,x] = \sum_{i=1}^{k} f[\lambda_1,\cdots,\lambda_{k+i}](x-\lambda_{k+1})\cdots(x-\lambda_{k+i-1}) + f[\lambda_1,\ldots,\lambda_{k+m+1},x](x-\lambda_{k+1})\cdots(x-\lambda_{k+m+1})$$

for $\lambda_{k+1},\ldots,\lambda_{k+m+1}$ any points in the domain of definition $\mathcal{D}_f$ of f. If $0\in\mathcal{D}_f$, then we can choose

$$\lambda_{k+1}=\cdots=\lambda_{k+m+1}=0.$$

Setting

$$M_i = f[\lambda_1,\ldots,\lambda_{k+i}]$$

we get

$$f[\lambda_1,\ldots,\lambda_k,x] = \sum_{i=1}^{m+1} M_i x^{i-1} + x^{m+1} f[\lambda_1,\ldots,\lambda_{k+m+1},x]$$

and hence, for the truncation error

$$E_T = \sum_{i=0}^{m} M_{i+1}\left(\int_a^b R_k(x)x^i\,d\alpha(x)\right) + \int_a^b f[\lambda_1,\ldots,\lambda_{k+m+1},x]x^{m+1}R_k(x)\,d\alpha(x)$$

with

$$\sum_{i=0}^{m}\left(\int_a^b R_k(x)x^i\,d\alpha(x)\right)^2$$

minimised. Moreover, if $f\in C^{k+m+1}([a,b])$ and, since x^{m+1} is positive over $[a,b]$, we obtain

$$\int_a^b f[\lambda_1,\ldots,\lambda_{k+m+1},x]x^{m+1}R_k(x)\,d\alpha(x) = \frac{c_{m+1}}{(k+m+1)!}R_k(\lambda)f^{(k+m+1)}(\eta)$$

with $\lambda,\eta\in[a,b]$, and, for the error,

$$E_T = \sum_{i=0}^{m}\frac{f^{(k+i)}(\eta_i)}{(k+i)!}\left(\int_a^b R_k(x)x^i d\alpha(x)\right) + \frac{c_{m+1}}{(k+m+1)!}R_k(\lambda)f^{(k+m+1)}(\eta) \quad (6)$$

with $\eta_i\in[a,b]$, $i=0,\ldots,m$, $\lambda,\eta\in[a,b]$. We remark that in the case where $m=k-1$, R_k is the orthogonal polynomial with respect to the functional c and so (5) corresponds to a Gaussian quadrature formula. An advantage of the quadrature formulas (5) is that they are less sensitive to perturbations on the sequence of moments c_i, as is shown in the following numerical example. Such a case can arise in some applications where the formula giving the moments c_i is sensitive to rounding errors, see [11] for example.

Consider the functional c defined by

$$c_i = \int_0^1 x^i\,dx = \frac{1}{i+1}$$

and perturb the coefficients in the following way

i	c_i^*	i	c_i^*
0	1.00000011	6	0.14285700
1	0.50000029	7	0.12500000
2	0.33333340	8	0.11111109
3	0.25000101	9	0.10000000
4	0.20000070	10	0.09090899
5	0.16666600	11	0.08333300

We can construct from these coefficients the least squares orthogonal polynomials and the corresponding quadrature formulas (5). The precision of the numerical approximations of $I=\int_0^1 f(x)\,dx$ is given in the following table

$f(x)$	$k=5;m=4$ Gauss quad.	$k=5;m=6$ least sq. quad.
$1/(x+0.5)$	$-2.2*10^{-5}$	$-6.2*10^{-6}$
$1/(x+0.3)$	$-2.1*10^{-4}$	$-1.2*10^{-5}$

We can obtain other applications from the following generalization. Instead of minimizing $\sum_{i=0}^{m}[c(x^iR_k(x))]^2$ we can introduce weights and minimize

$$\Phi^*(b_0,\ldots,b_{k-1}) = \sum_{i=0}^{m} p_i\left[c(x^iR_k^*(x))\right]^2$$

with $p_i>0$, $i=0,\ldots,m$. If we choose the inner product

$$(\gamma_i,\gamma_j)^* = \sum_{k=0}^{m} p_k c_{i+k}c_{j+k}$$

the solution of this problem can be computed as in the previous case and all the properties of the polynomials are still true. It can be seen, from numerical examples, that if the sequence of moments c_i has a decreasing precision, we can expect that the least squares Padé-type approximants constructed with a decreasing sequence of weights will give a better result. In the same way, for the quadrature formulas (5), from the expression (6) of the truncation error and the knowledge of the magnitude of the derivatives, we can reduce this error by choosing appropriate weights. Some other possible applications of least squares orthogonal polynomials will be studied in the future.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- ABS Algorithms for Optimization
- Gauss–Newton Method: Least Squares, Relation to Newton's Method
- Generalized Total Least Squares
- Least Squares Problems
- Nonlinear Least Squares: Newton-type Methods
- Nonlinear Least Squares Problems
- Nonlinear Least Squares: Trust Region Methods

References

1. Brezinski C (1980) Padé type approximation and general orthogonal polynomials. ISNM, vol 50. Birkhäuser, Basel
2. Brezinski C, Matos AC (1993) Least squares orthogonal polynomials. J Comput Appl Math 46:229–239
3. Brezinski C, Redivo Zaglia M (1991) Extrapolation methods. Theory and practice. North-Holland, Amsterdam
4. Brezinski C, Redivo Zaglia M, Sadok H (1992) A breakdown-free Lanczos type algorithm for solving linear systems. Numer Math 63:29–38
5. Faddeeva VN (1959) Computational methods of linear algebra. Dover, Mineola, NY
6. Gantmacher FR (1959) The theory of matrices. Chelsea, New York
7. Iserles A, Nørsett SP (1985) Bi-orthogonal polynomials. In: Brezinski C, Draux A, Magnus AP, Maroni P, Ronveaux A (eds) Orthogonal Polynomials and Their Applications. Lecture Notes Math. Springer, Berlin, pp 92–100
8. Iserles A, Nørsett SP (1988) On the theory of biorthogonal polynomials. Trans Amer Math Soc 306:455–474
9. Karlin S (1968) Total positivity. Stanford Univ. Press, Palo Alto, CA
10. Mason JC (1981) Some applications and drawbacks of Padé approximants. In: Ziegler Z (ed) Approximation Theory and Appl. Acad. Press, New York, pp 207–223
11. Morandi Cecchi M, Redivo Zaglia M (1991) A new recursive algorithm for a Gaussian quadrature formula via orthogonal polynomials. In: Brezinski C, Gori L, Ronveaux A (eds) Orthogonal Polynomials and Their Applications. Baltzer, Basel, pp 353–358

Least Squares Problems

Åke Björck
Linköping University, Linköping, Sweden

MSC2000: 65Fxx

Article Outline

Keywords
Synonyms
Introduction
  Historical Remarks
  Statistical Models
  Characterization of Least Squares Solutions
Pseudo-inverse and Conditioning
  Singular Value Decomposition and Pseudo-inverse
  Conditioning of the Least Squares Problem
Numerical Methods of Solution
  The Method of Normal Equations
  Least Squares by QR Factorization
  Rank-Deficient and Ill-Conditioned Problems
  Rank Revealing QR Factorizations
Updating Least Squares Solutions
  Recursive Least Squares
  Modifying Matrix Factorizations
Sparse Problems
  Banded Least Squares Problems
  Block Angular Form
  General Sparse Problems
See also
References

Keywords

Least squares

Synonyms

LSP

Introduction

Historical Remarks

The linear least squares problem originally arose from the need to fit a linear mathematical model to given observations. In order to reduce the influence of errors in the observations one uses a greater number of measurements than the number of unknown parameters in the model.

The algebraic procedure of the method of least squares was first published by A.M. Legendre [25]. It was justified as a statistical procedure by C.F. Gauss [13]. A famous example of the use of the least squares principle is the prediction of the orbit of the asteroid Ceres by Gauss in 1801. After this success, the method of least squares quickly became the standard procedure for analysis of astronomical and geodetic data.

Gauss gave the method a sound theoretical basis in two memoirs: 'Theoria Combinationis' [11,12]. In them, Gauss proves the optimality of the least squares estimate without any assumptions that the random variables follow a particular distribution.

Statistical Models

In the *general univariate linear model* the vector $b \in \mathbf{R}^m$ of observations is related to the unknown parameter vector $x \in \mathbf{R}^n$ by a linear relation

$$Ax = b + \epsilon, \tag{1}$$

where $A \in \mathbf{R}^{m\times n}$ is a known matrix of full column rank. Further, ϵ is a vector of random errors with zero means and covariance matrix $\sigma^2\ W \in \mathbf{R}^{m\times m}$, where W is known but $\sigma^2 > 0$ unknown. The standard linear model is obtained for $W = I$.

Theorem 1 (*Gauss–Markoff theorem*) *Consider the standard linear model (1) with $W = I$. The best linear unbiased estimator of any linear function $c^\mathsf{T}x$ is $c^\top\widehat{x}$, where $\widehat{x}$ is obtained by minimizing the sum of the squared residuals,*

$$\|r\|_2^2 = \sum_{i=1}^{m} r_i^2, \tag{2}$$

where $r = b - Ax$ and $\|\cdot\|_2$ denotes the Euclidean vector norm. Furthermore, $E(s^2) = \sigma^2$, where s^2 is the quadratic form

$$s^2 = \frac{1}{m-n}(b - A\widehat{x})^\top(b - A\widehat{x}). \tag{3}$$

The variance-covariance matrix of the least squares estimate $\widehat{x}$ is given by

$$V(\widehat{x}) = \sigma^2(A^\top A)^{-1}. \tag{4}$$

The residual vector $\widehat{r} = b - A\widehat{x}$ satisfies $A^\top\widehat{r} = 0$, and hence there are n linear relations among the m components of $\widehat{r}$. It can be shown that the residuals $\widehat{r}$, and therefore also the quadratic form s^2, are uncorrelated with $\widehat{x}$, i. e., $\mathrm{cov}(\widehat{r},\widehat{x}) = 0$, $\mathrm{cov}(s^2,\widehat{x}) = 0$.

If the errors in ϵ are uncorrelated but not of equal variance, then the covariance matrix W is diagonal. Then the least squares estimator is obtained by solving the *weighted least squares problem*

$$\min \|D(Ax - b)\|_2\,, \qquad D = W^{-\frac{1}{2}}. \tag{5}$$

For the general case with no restrictions on A and W, see [23].

The assumption that A is known made in the linear model is frequently unrealistic since sampling or modeling errors often also affect A. In the *errors-in-variables model* one instead assumes a linear relation

$$(A + E)x = b + r, \tag{6}$$

where (E, r) is an error matrix whose rows are independently and identically distributed with zero mean and the same variance. An estimate of the parameters x in the model (6) is obtained from the total least squares (TLS) problem.

Characterization of Least Squares Solutions

Let $\mathcal{S}$ be set of all solutions to a least squares problem,

$$S = \{x \in \mathbb{R}^n\colon\ \|Ax - b\|_2 = \min\}\,. \tag{7}$$

Then $x \in \mathcal{S}$ if and only if $A^\mathsf{T}(b - Ax) = 0$ holds. Equivalently, $x \in \mathcal{S}$ if and only if x satisfies the *normal equations*

$$A^\top Ax = A^\top b. \tag{8}$$

Since $A^\mathsf{T}\,b \in \mathcal{R}(A^\mathsf{T}) = \mathcal{R}(A^\mathsf{T}\,A)$ the normal equations are always consistent. It follows that $\mathcal{S}$ is a nonempty, convex subset of $\mathbf{R}^n$. Any least squares solution x uniquely decomposes the right-hand side b into two orthogonal components

$$b = Ax + r, \quad Ax \in \mathcal{R}(A) \perp r \in \mathcal{N}(A^\top),$$

where $\mathcal{R}(A)$ and $\mathcal{N}(A^\mathsf{T})$ denote the range of A and the nullspace of A^T, respectively.

When rank $A < n$ there are many least squares solutions x, although the residual $b - Ax$ is still uniquely determined. There is always a unique least squares solution in $\mathcal{S}$ of minimum length. The following result applies to both overdetermined and underdetermined linear systems.

Theorem 2 *Consider the linear least squares problem*

$$\min_{x\in S} \|x\|_2, \quad S = \{x \in \mathbb{R}^n : \ \|b - Ax\|_2 = \min\}, \tag{9}$$

where $A \in \boldsymbol{R}^{m\times n}$ and rank(A) = r ≤ min(m, n). This problem always has a unique solution, which is distinguished by the property that

$$x \perp \mathcal{N}(A).$$

Pseudo-inverse and Conditioning

Singular Value Decomposition and Pseudo-inverse

A matrix decomposition of great theoretical and practical importance for the treatment of least squares problems is the singular value decomposition (SVD) of A,

$$A = U\Sigma V^\top = \sum_{i=1}^{n} u_i \sigma_i v_i^\top. \tag{10}$$

Here σ_i are the singular values of A and u_i and v_i the corresponding left and right singular vectors.

Using this decomposition the solution to problem (9) can be written $x = A^\dagger b$, where

$$A^\dagger = V \begin{pmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^\top \in \mathbb{R}^{n\times m}. \tag{11}$$

Here $A^\dagger$ is called the *pseudo-inverse* of A. It is the unique matrix which minimizes $\|AX - I\|_F$, where $\|\cdot\|_F$ denotes the Frobenius norm. Note that the pseudo-inverse $A^\dagger$ is not a continuous function of A, unless one allows only perturbations which do not change the rank of A.

The pseudo-inverse was first introduced by E.H. Moore in 1920. R. Penrose [30] later gave the following elegant algebraic characterization.

Theorem 3 (*Penrose's conditions*) *The pseudo-inverse $X = A^\dagger$ is uniquely determined by the four conditions:*

1) $AXA = A$;
2) $XAX = X$;
3) $(AX)^\mathsf{T} = AX$;
4) $(XA)^\mathsf{T} = XA$.

It can be directly verified that $A^\dagger$ given by (11) satisfies these four conditions.

The *total least squares problem* (TLS problem) involves finding a perturbation matrix (E, r) having minimal Frobenius norm, which lowers the rank of the matrix (A, b). Consider the singular value decomposition of the augmented matrix (A, b):

$$(A, b) = U\Sigma V^\top, \quad \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_{n+1}),$$

where $\sigma_1 \geq \cdots \geq \sigma_{n+1} \geq 0$. Then, in the generic case, $(x, -1)^\mathsf{T}$ is a right singular vector corresponding to σ_{n+1} and $\min \|(E, r)\|_F = \sigma_{n+1}$.

An excellent survey of theoretical and computational aspects of the total least squares problem is given in [22].

Conditioning of the Least Squares Problem

Consider a *perturbed least squares problem* where $\widetilde{A} = A + \delta A$, $\widetilde{b} = b + \delta b$, and let the perturbed solution be $\widetilde{x} = x + \delta x$. Then, assuming that rank(A) = rank$(A + \delta A) = n$ one has the first order bound

$$\|\delta x\|_2 \leq \frac{1}{\sigma_n}\left(\|\delta b_1\|_2 + \|\delta A\|_2 \|x\|_2\right) + \frac{1}{\sigma_n^2}\|\delta A\|_2 \|r\|_2.$$

The *condition number of a matrix* $A \in \mathbf{R}^{m\times n}$ $(A \neq 0)$ is defined as

$$\kappa(A) = \|A\|_2 \left\|A^\dagger\right\|_2 = \frac{\sigma_1}{\sigma_r}, \tag{12}$$

where $\sigma_1 \geq \cdots \geq \sigma_r > 0$, are the nonzero singular values of A. Hence, the *normwise relative condition number* of the least squares problem can be written as

$$\kappa_{LS}(A, b) = \kappa(A) + \kappa(A)^2 \frac{\|r\|_2}{\|A\|_2 \|x\|_2}. \tag{13}$$

For a *consistent problem* $(r = 0)$ the last term is zero. However, in general the condition number depends on the size of r and involves a term proportional to $\kappa(A)^2$.

A more refined perturbation analysis, which applies to both overdetermined and underdetermined systems, has been given in [34]. In order to prove any meaningful result it is necessary to assume that rank$(A + \delta A)$ = rank(A). If rank(A) = min(m, n), the condition $\eta \equiv \|A^\dagger\|_2 \|\delta A\|_2 < 1$ suffices to ensure that this is the case.

Numerical Methods of Solution

The Method of Normal Equations

The first step in the method of normal equations for the least squares problem is to form the cross-products

$$C = A^\top A \in \mathbb{R}^{n\times n}, \quad d = A^\top b \in \mathbb{R}^n. \tag{14}$$

Since the matrix C is symmetric, it is only necessary to compute and store its upper triangular part. When $m \gg n$ this step will result in a great reduction in the amount of data.

The computation of C and d can be performed either using an inner product form (operating on columns of A) or an outer product form (operating on rows of A). Row-wise accumulation of C and d is advantageous if the matrix A is sparse or held in secondary storage. Partitioning A by rows, one has

$$C = \sum_{i=1}^{m} a_{i.} a_{i.}^\top, \quad d = \sum_{i=1}^{m} b_i a_{i.}, \tag{15}$$

where $\widetilde{a}_i^\top$ denotes the ith row of A. This expresses C as a sum of matrices of rank.

Gauss solved the symmetric positive definite system of normal equation by elimination, preserving symmetry, and solving for x by back-substitution. A different sequencing of this algorithm is to compute the *Cholesky factorization*

$$C = R^\top R, \tag{16}$$

where R is upper triangular with positive diagonal elements, and then solve the two triangular systems

$$R^\top z = d, \quad Rx = z, \tag{17}$$

by forward- and back-substitution, respectively. The Cholesky factorization, named after the French officer A.L. Cholesky, who worked on geodetic survey problems in Africa, was published by C. Benoit [1]. (In statistical applications this method is often known as the *square-root method*, although the proper square root of A should satisfy $B^2 = A$.)

The method of normal equations is suitable for moderately ill-conditioned problems but is not a backward stable method. The accuracy can be improved by using fixed precision iterative refinement in solving the normal equations.

Set $x_0 = 0$, $r_0 = 0$, and for $s = 0, 1, \ldots$ until convergence do

$$\begin{aligned} r_s &= b - Ax_s, \\ R^\top (R\delta x_s) &= A^\top r_s, \\ x_{s+1} &= x_s + \delta x_s. \end{aligned}$$

(Here, x_1 corresponds to the unrefined solution of the normal equations.)

The method of normal equations can fail when applied to weighted least squares problems. To see this consider a problem with two different weights γ and 1,

$$\min_x \left\| \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix} x - \begin{pmatrix} \gamma b_1 \\ b_2 \end{pmatrix} \right\|_2, \tag{18}$$

for which the matrix of normal equations is $A^\intercal A = \gamma^2 A_1^\top A_1 + A_2^\top A_2$. When $\gamma \gg 1$ this problem is called *stiff*. In the limit $\gamma \to \infty$ the solution will satisfy the subsystem $A_1 x = b_1$ exactly. If $\gamma > u^{-1/2}$ (u is the unit roundoff), the information in the matrix A_2 may completely disappear when forming $A^\intercal A$. For possible ways around this difficulty, see [4, Chap. 4.4].

Least Squares by QR Factorization

The *QR factorization* and its extensions are used extensively in modern numerical methods for solving least squares problems. Let $A \in \mathbf{R}^{m\times n}$ with $\operatorname{rank}(A) = n$. Then there are an orthogonal matrix $Q \in \mathbf{R}^{m\times m}$ and an upper triangular $R \in \mathbf{R}^{n\times n}$ such that

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \tag{19}$$

Since orthogonal transformations preserve the Euclidean length, it follows that

$$\|Ax - b\|_2 = \left\| Q^\top (Ax - b) \right\|_2 \tag{20}$$

for any orthogonal matrix $Q \in \mathbf{R}^{m\times m}$. Hence using the QR factorization (19) the solution to the least squares problem can be obtained from

$$Q^\top b = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}, \quad Rx = d_1. \tag{21}$$

An algorithm based on the QR decomposition by Householder transformations was first developed in a seminal paper by G.H. Golub [18]. Here, Q is compactly represented as a product of Householder ma-

trices $Q = P_1 \cdots P_n$, where $P_k = I - \beta_k u_k u_k^\top$. Only the Householder vectors u_k are stored, and advantage is taken of the fact that the first $k-1$ components of u_k are zero.

Golub's method for solving the standard least squares problem is normwise backward stable, see [24, pp. 90ff]. Surprisingly, this method is stable also for solving the weighted least squares problems (5) provided only that the equations are sorted after decreasing row norms in A, see [8].

Due to storage considerations the matrix Q in a QR decomposition is often discarded when A is large and sparse. This creates a problem, since then it may not be possible to form $Q^\top b$. If the original matrix A is saved one can use the *corrected seminormal equations* (CSNE)

$$R^\top R\overline{x} = A^\top b, \quad \overline{r} = b - A\overline{x},$$
$$\overline{R}^\top \overline{R}\delta x = A^\top \overline{r}, \quad x_c = \overline{x} + \delta x.$$

(Note that unless the correction step is carried out the numerical stability of this method is no better than the method of normal equations.) An error analysis of the CSNE method is given in [2]. A comparison with the bounds for a backward stable method shows that in most practical applications the corrected seminormal equations is forward stable.

Applying the *Gram–Schmidt orthogonalization* process to the columns of A produces Q_1 and R in the factorization

$$A = (a_1, \ldots, a_n) = Q_1 R, \quad Q_1 = (q_1, \ldots, q_n),$$

where Q_1 has orthogonal columns and R is upper triangular. There are two computational variants of Gram–Schmidt orthogonalization, the *classical Gram–Schmidt orthogonalization* (CGS) and the *modified Gram–Schmidt orthogonalization* (MGS). In CGS there may be a catastrophic loss of orthogonality unless reorthogonalization is used. In MGS the loss of orthogonality can be shown to occur in a predictable manner.

Using an equivalence between MGS and Householder QR applied to A with a square matrix of zeros on top, backward stable algorithm based on MGS for solving least squares problems have been developed, see [3].

Rank-Deficient and Ill-Conditioned Problems

The mathematical notion of rank is not always appropriate in numerical computations. For example, if a matrix $A \in \mathbf{R}^{n\times n}$, with (mathematical) rank $k < n$, is randomly perturbed by roundoff, the perturbed matrix most likely has full rank n. However, it should be considered to be 'numerically' rank deficient.

When solving rank-deficient or ill-conditioned least squares problems, correct assignment of the 'numerical rank' of A is often the key issue. The *numerical rank* should depend on a tolerance which reflects the error level. Overestimating the rank may lead to a computed solution of very large norm, which is totally irrelevant. This behavior is typical in problems arising from discretizations of ill-posed problems, see [21].

Assume that the 'noise level' δ in the data is known. Then a numerical rank k, such that $\sigma_k > \delta \geq \sigma_{k+1}$, can be assigned to A, where σ_i are the singular values of A. The approximate solution

$$x = \sum_{i=1}^{k} \frac{c_i}{\sigma_i} v_i, \quad c = U^\top b,$$

is known as the *truncated singular value decomposition solution* (TSVD). This solution solves the related least squares problem $\min_x \|A_k x - b\|_2$, where

$$A_k = \sum_{i=1}^{k} u_i \sigma_i v_i^\top, \quad \|A - A_k\|_2 \leq \delta,$$

is the best rank k approximation of A, The subspace

$$\mathcal{R}(V_2), \quad V_2 = (v_{k+1}, \ldots, v_n),$$

is called the *numerical nullspace* of A.

An alternative to TSVD is *Tikhonov regularization*, where one considers the regularized problem

$$\min_x \|Ax - b\|_2^2 + \tau^2 \|Dx\|_2^2, \tag{22}$$

for some positive diagonal matrix $D = \mathrm{diag}(d_1, \ldots, d_n)$. The problem (22) is equivalent to the least squares problem

$$\min_x \left\| \begin{pmatrix} \tau D \\ A \end{pmatrix} x \begin{pmatrix} \tau 0 \\ b \end{pmatrix} \right\|_2, \tag{23}$$

where the matrix A has been modified by appending the matrix $\tau\ D$ on top. An advantage of using the regularized problem (23) instead of the TSVD is that its solution can be computed from a QR decomposition. When $\tau > 0$ this problem is always of full column rank and has

a unique solution. For $D = I$ it can be shown that $x(\tau)$ will approximately equal the TSVD solution for $\tau = \delta$.

Problem (23) also appears as a subproblem in trust region algorithms for solving nonlinear least squares, and in interior point methods for constrained linear least squares problems. A more difficult case is when the noise level δ is unknown and has to be determined in the solution process. Such problems typically arise in the treatment of discrete ill-posed problems, see [21].

Rank Revealing QR Factorizations

In some applications it is too expensive to compute the SVD. In such cases so called 'rank revealing' QR factorizations, often are a good substitute.

It can be shown that for any $0 < k < n$ a column permutation Π exists such that the QR decomposition of $A\,\Pi$ has the form

$$A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}, \tag{24}$$

where

$$\sigma_k(R_{11}) \geq \frac{1}{c}\sigma_k, \quad \|R_{22}\|_2 \leq c\sigma_{k+1}, \tag{25}$$

and $c < (n+1)/2$. In particular, if A has numerical δ-rank equal to k, then there is a column permutation such that $\| R_{22} \|_2 \leq c\,\delta$. Such a QR factorization is called a *rank revealing QR factorization* (RRQR). No efficient numerical method is known which can be guaranteed to compute an RRQR factorization satisfying (25), although in practice *Chan's method* [7] often gives satisfactory results.

A related rank revealing factorization is the complete orthogonal decomposition of the form

$$A = U \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} V^{\top}, \tag{26}$$

where U and V are orthogonal matrices, $R_{11} \in \mathbf{R}^{k\times k}$, $\sigma_k(R_{11}) \geq \sigma_k/c$, and

$$\left(\|R_{12}\|_F^2 + \|R_{22}\|_F^2\right)^{\frac{1}{2}} \leq c\sigma_{k+1}.$$

This is also often called a *rank revealing URV factorization*. (an alternative lower triangular form ULV is sometimes preferable to use.) If $V = (V_1 V_2)$ is partitioned conformably the orthogonal matrix V_2 can be taken as an approximation to the numerical nullspace $\mathcal{N}(A)$.

Updating Least Squares Solutions

It is often desired to solve a sequence of modified least squares problems

$$\min_x \|Ax - b\|_2, \quad A \in \mathbb{R}^{m\times n}, \tag{27}$$

where in each step rows of data in $(A,\ b)$ are added, deleted, or both. This need arises, e. g., when data are arriving sequentially. In various time-series problems a window moving over the data is used; when a new observation is added, an old one is deleted as the window moves to the next step in the sample. In other applications columns of the matrix A may be added or deleted. Such modifications are usually referred to as updating (downdating) of least squares solutions.

Important applications where modified least squares problems arise include statistics, optimization, and signal processing. In statistics an efficient and stable procedure for adding and deleting rows to a regression model is needed; see [6]. In regression models one may also want to examine the different models, which can be achieved by adding or deleting columns (or permuting columns).

Recursive Least Squares

Applications in signal processing often require near real-time solutions. It is then critical that the modification should be performed with as few operations and as little storage requirement as possible.

Methods based on the normal equations and/or updating of the Cholesky factorization are still often used in statistics and signal processing, although these algorithms lack numerical stability. Consider a least squares problem where an observation $w^{\intercal}x = \beta$ is added. The updated solution $\widetilde{x}$ then satisfies the modified normal equations

$$(A^{\top}A + ww^{\top})\widetilde{x} = A^{\top}b + \beta w. \tag{28}$$

A straightforward method for computing $\widetilde{x}$ is based on updating the (scaled) covariance matrix $C = (A^{\intercal}\,A)^{-1}$. By the *Sherman–Morrison formula* one obtains $\widetilde{C}^{-1} = C^{-1} + ww^{\top}$, and

$$\widetilde{C} = C - \frac{1}{1 + w^{\top}u}uu^{\top}, \quad u = Cw. \tag{29}$$

From this follows the updating formula

$$\widetilde{x} = x + (\beta - w^{\top}x)\widetilde{u}, \quad \widetilde{u} = \widetilde{C}w. \tag{30}$$

The equations (29), (30) define a *recursive least squares* (RLS) algorithm. They can, with slight modifications, also be used for 'deleting' observations. The simplicity of this updating algorithm is appealing, but a disadvantage is its serious sensitivity to roundoff errors.

Modifying Matrix Factorizations

The first area where algorithms for *modifying matrix factorizations* seems to have been systematically used is optimization. Numerous aspects of updating various matrix factorizations are discussed in [17].

There is a simple relationship between the problem of updating matrix factorizations and that of updating least squares solutions. If A has full column rank and the R-factor of the matrix (A, b) is

$$\begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix}, \tag{31}$$

then the solution to the least squares problem (27) is given by

$$Rx = z, \quad \|Ax - b\|_2 = \rho. \tag{32}$$

Hence updating algorithms for the QR or Cholesky factorization can be applied to (A, b) in order to give updating algorithms for least squares solutions.

Backward stable algorithms, which require $O(m^2)$ multiplications, exist for updating the QR decomposition for three important kinds of modifications:

- General rank one change of A.
- Deleting (adding) a column of A.
- Adding (deleting) a row of A.

In these algorithms, $Q \in \mathbf{R}^{m\times m}$ is stored explicitly as an $m \times m$ matrix. In many applications it suffices to update the 'Gram–Schmidt' QR decomposition

$$A = Q_1 R, \quad Q_1 \in \mathbb{R}^{m\times n}, \tag{33}$$

where $Q_1 \in \mathbf{R}^{m\times n}$ consists of the first n columns of Q, [10,31]. These only require $O(mn)$ storage and operations.

J.R. Bunch and C.P. Nielsen [5] have developed methods for updating the SVD

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^{\top},$$

where $U \in \mathbf{R}^{m\times m}$ and $V \in \mathbf{R}^{n\times n}$, when A is modified by adding or deleting a row or column. However, their algorithms require $O(mn^2)$ flops.

Rank revealing QR factorizations can be updated more cheaply, and are often a good alternative to use. G.W. Stewart [33] has shown how to compute and update a rank revealing complete orthogonal decomposition from an RRQR decomposition.

Most updating algorithms can be modified in a straightforward fashion to treat cases where a block of rows/columns are added or deleted. which are more amenable to efficient implementation on vector and parallel computers.

Sparse Problems

The gain in operations and storage in solving the linear least squares problems where the matrix A is sparse can be huge, making otherwise intractable problems possible to solve. *Sparse least squares problems* of huge size arise in a variety of applications, such as geodetic surveys, photogrammetry, molecular structure, gravity field of the earth, tomography, the force method in structural analysis, surface fitting, and cluster analysis and pattern matching.

Sparse least squares problems may be solved either by direct or iterative methods. Preconditioned iterative methods can often be considered as hybrids between these two classes of solution. Below direct methods are reviewed for some classes of sparse problems.

Banded Least Squares Problems

A natural distinction is between sparse matrices with regular zero pattern (e. g., banded structure) and matrices with an irregular pattern of nonzero elements.

A rectangular *banded matrix* $A \in \mathbf{R}^{m\times n}$ has the property that the nonzero elements in each row lie in a narrow band. A is said to have *row bandwidth* w if

$$w(A) = \max_{1\le i\le m} (l_i(A) - f_i(A) + 1). \tag{34}$$

where

$$f_i(A) = \min\{j\colon\ a_{ij} \neq 0\},$$
$$l_i(A) = \max\{j\colon\ a_{ij} \neq 0\}$$

are column subscripts of the first and last nonzeros in the ith row of A. For this structure to have practical significance one needs to have $w \ll n$. Note that, although the row bandwidth is independent of the row ordering, it will depend on the column ordering. To permute the

columns in A so that a small bandwidth is achieved the method of choice is the reverse Cuthill–McKee ordering, see [15].

It is easy to see that if the row bandwidth of A is w then the matrix of normal equations $C = A^{\mathsf{T}}A$ has at most upper bandwidth $p = w - 1$, i. e.,

$$|j - k| \geq w \quad \Rightarrow \quad (A^{\top}A)_{jk} = \sum_{i=1}^{m} a_{ij}a_{ik} = 0.$$

If advantage is taken of the band structure, the solution of a least squares problem where A has bandwidth w by the method of normal equations requires a total of

$$\tfrac{1}{2}\,(mw(w+3) + n(w-1)(w+2)) + n(2w-1)$$

flops.

Similar savings can be obtained for methods based on Givens QR decomposition used to solve banded least squares problem. However, then it is essential that the rows of A are sorted so that the column indices $f_i(A)$, $i = 1, \ldots, m$, of the first nonzero element in each row form a nondecreasing sequence, i. e.,

$$i \leq k \quad \Rightarrow \quad f_i(A) \leq f_k(A).$$

A matrix whose rows are sorted in this way is said to be in *standard form*. Since the matrix R in the QR factorization has the same structure as the Cholesky factor, it must be a banded matrix with nonzero elements only in the first $p = w - 1$ superdiagonals. In the *sequential row orthogonalization scheme* an upper triangular matrix R is initialized to zero. The orthogonalization then proceeds row-wise, and R is updated by adding a row of A at a time.

If A has constant bandwidth and is in standard form then in the ith step of reduction the last $(n - l_i(A))$ columns of R have not been touched and are still zero as initialized. Further, the first $(f_i(A) - 1)$ rows of R are already finished at this stage and can be read out to secondary storage. Thus, as with the Cholesky method, very large problems can be handled since primary storage is needed only for the active part of R. The complete orthogonalization requires about $2mw^2$ flops, and can be performed in $w(w + 3)/2$ locations of primary storage.

The Givens rotations could also be applied to one or several right-hand sides b. Only if right-hand sides which are not initially available are to be treated, need the Givens rotations be saved. The algorithm can be modified to also handle problems with variable row bandwidth w_i.

For the case when $m \gg n$ a more efficient schemes uses Householder transformations, see [24, Chap. 11]. Let A_k consist of the rows of A for which the first nonzero element is in column k. Then, in step k of this algorithm, the A_k is merged with R_{k-1}, by computing the QR factorization

$$Q_k^{\top}\begin{pmatrix} R_{k-1} \\ A_k \end{pmatrix} = R_k.$$

Note that this and later steps will not involve the first $k - 1$ rows and columns of R_{k-1}. Hence the first $k - 1$ rows of R_{k-1} are rows in the final matrix R.

The reduction using this algorithm takes about $w(w + 1)(m + 3n/2)$ flops, which is approximately half as many as for the Givens method. As in the Givens algorithm the Householder transformations can also be applied to one or several right-hand sides b to produce $c = Q^{\mathsf{T}}b$. The least squares solution is then obtained from $Rx = c_1$ by back-substitution.

It is essential that the Householder transformations be subdivided as outlined above, otherwise intermediate fill will occur and the operation count increase greatly, see the example in [32].

Block Angular Form

There is often a substantial similarity in the structure of large sparse least squares problems. The matrices possess a block structure, perhaps at several levels, which reflects a 'local connection' structure in the underlying physical problem. In particular, the problem can often be put in the following bordered block diagonal or *block angular form*:

$$A = \begin{pmatrix} A_1 & & & B_1 \\ & \ddots & & \vdots \\ & & A_M & B_M \end{pmatrix}, \tag{35}$$

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_{M+1} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_M \end{pmatrix}. \tag{36}$$

From (35) it follows that the variables $x_1, \ldots, x_M$ are coupled only to the variables x_{M+1}. Some applications

where the form (35) arises naturally are photogrammetry, Doppler radar positioning [27], and geodetic survey problems [20].

Problems of block angular form can be efficiently treated either by using normal equations of by QR factorization. It is easily seen that the matrix R from Cholesky or QR will have a block structure similar to that of A,

$$R = \begin{pmatrix} R_1 & & & S_1 \\ & \ddots & & \vdots \\ & & R_M & S_M \\ & & & R_{M+1} \end{pmatrix}, \tag{37}$$

where the $R_i \in \mathbf{R}^{n_i \times n_i}$ are upper triangular. This factor can be computed by first performing a sequence of orthogonal transformations yielding

$$Q_i^\top (A_i, B_i) = \begin{pmatrix} R_i & S_i \\ 0 & T_i \end{pmatrix}, \quad Q_i^\top b_i = \begin{pmatrix} c_i \\ d_i \end{pmatrix}.$$

Any sparse structure in the blocks A_i should be exploited. The last block row R_{M+1}, c_{M+1} is obtained by computing the QR decomposition

$$\widetilde{Q}_{M+1}^\top \begin{pmatrix} T & d \end{pmatrix} = \begin{pmatrix} R_{M+1} & c_{M+1} \\ 0 & d_{M+1} \end{pmatrix},$$

where

$$T = \begin{pmatrix} T_1 \\ \vdots \\ T_M \end{pmatrix}, \quad d = \begin{pmatrix} d_1 \\ \vdots \\ d_M \end{pmatrix}.$$

The unknown x_{M+1} is determined from the triangular system $R_{M+1}\, x_{M+1} = c_{M+1}$. Finally $x_M, \ldots, x_1$ are computed by back-substitution in the sequence of triangular systems $R_i x_i = c_i - S_i\, x_{M+1}$, $i = M, \ldots, 1$. Note that a large part of the computations can be performed in parallel on the M subsystems.

Several modifications of this basic algorithm have been suggested in [19] and [9].

General Sparse Problems

If A is partitioned by rows, then (15) can be used to compute the matrix $C = A^\mathsf{T} A$. Make the 'no-cancellation assumption' that whenever two nonzero numerical quantities are added or subtracted, the result is nonzero. Then it follows that the nonzero structure of $A^\mathsf{T} A$ is the direct sum of the nonzero structures of $a_i . a_i^\top$, $i = 1, \ldots, m$, where $a_i^\top$ denotes the ith row of A. Hence the undirected graph $G(A^\mathsf{T} A)$ representing the structure of $A^\mathsf{T} A$ can be constructed as the direct sum of all the graphs $G(a_i . a_i^\top)$, $i = 1, \ldots, m$. The nonzeros in row $a_i^\top$ will generate a subgraph, where all pairs of nodes are connected. Such a subgraph is called a *clique*.

From the graph $G(A^\mathsf{T} A)$ the structure of the Cholesky factor R can be predicted by using a graph model of Gaussian elimination. The fill under the factorization process can be analyzed by considering a sequence of undirected graphs $G_i = G(A^{(i)})$, $i = 0, \ldots, n-1$, where $A^{(0)} = A$. These *elimination graphs* can be recursively formed in the following way. Form G_i from $G_{(i-1)}$ by removing the node i and its incident edges and adding fill edges. The fill edges in eliminating node v in the graph G are

$$\{(j,k) \colon\ (j,k) \in \mathrm{Adj}_G(v),\ j \neq k\}.$$

Thus, the fill edges correspond to the set of edges required to make the adjacent nodes of v pairwise adjacent. The filled graph $G_F(A)$ of A is a graph with n vertices and edges corresponding to all the elimination graphs G_i, $i = 0, \ldots, n-1$. The filled graph bounds the structure of the Cholesky factor R,

$$G(R^\top + R) \subset G_F(A). \tag{38}$$

This also give an upper bound for the structure of the factor R in the QR decomposition.

A reordering of the columns of AP of A corresponds to a symmetric reordering of the rows and columns of $A^\mathsf{T} A$. Although this will not affect the number of nonzeros in $A^\mathsf{T} A$, only their positions, it may greatly affect the number of nonzeros in the Cholesky factor R. Before carrying out the Cholesky or QR factorization numerically, it is therefore important to find a permutation matrix P such that $P^\mathsf{T} A^\mathsf{T} AP$ has a sparse Cholesky factor R.

By far the most important local ordering method is the *minimum degree ordering* In terms of the Cholesky factorization this ordering is equivalent to choosing the ith pivot column as one with the minimum number of nonzero elements in the unreduced part of $A^\mathsf{T} A$. This will minimize the number of entries that will be modified in the next elimination step. Remarkably fast

symbolic implementations of the minimum degree algorithm exist, which use refinements of the elimination graph model of the Cholesky factorization. See [16] for a survey of the extensive development of efficient versions of the minimum degree algorithm.

Another important ordering method is *substructuring* or *nested dissection*, which results in a nested block angular form. Here the idea is to choose a set of nodes $\mathcal{B}$ in the graph $G(A^\intercal A)$, which separates the other nodes into two sets $\mathcal{A}_1$ and $\mathcal{A}_2$ so that node variables in $\mathcal{A}_1$ are not connected to node variables in $\mathcal{A}_2$. The variables are then ordered so that those in $\mathcal{A}_1$ appear first, those in $\mathcal{A}_2$ second, and those in $\mathcal{B}$ last. Finally the equations are ordered so that those including $\mathcal{A}_1$ come first, those including $\mathcal{A}_2$ next, and those only involving variables in $\mathcal{B}$ come last. This dissection can be continued recursively, first dissecting the regions $\mathcal{A}_1$ and $\mathcal{A}_2$ each into two subregions, and so on.

An algorithm using the normal equations for solving sparse linear least squares problems is usually split in a symbolical and a numerical phase as follows.

1) Determine symbolically a column permutation P_c such that $P_c^\top A^\intercal A P_c$ has a sparse Cholesky factor R.
2) Perform the Cholesky factorization of $P_c^\top A^\intercal A P_c$ symbolically to generate a storage structure for R.
3) Compute $B = P_c^\top A^\intercal A P_c$ and $c = P_c^\top A^\intercal b$ numerically, storing B in the data structure of R.
4) Compute the Cholesky factor R numerically and solve $R^\intercal z = c$, $Ry = z$, giving the solution $x = P_c y$.

Here, steps 1 and 2 involve only symbolic computation and apply also to a sparse QR algorithm. For details of the implementation of the numerical factorization see [15, Chap. 5]. For moderately ill-conditioned problems a sparse Cholesky factorization, possibly used with iterative refinement, is a satisfactory choice.

Orthogonalization methods are potentially more accurate since they work directly with A. The number of operations needed to compute the QR decomposition depends on the row ordering, and the following heuristic row ordering algorithm should be applied to A before the numerical factorization takes place:

First sort the rows after increasing $f_i(A)$, so that $f_i(A) \le f_k(A)$ if $i < k$. Then for each group of rows with $f_i(A) = k$, $k = 1, \ldots, \max_i f_i(A)$, sort all the rows after increasing $L_i(A)$.

In the sparse case, applying the usual sequence of Householder reflections may cause a lot of *intermediate fill-in*, with consequent cost in operations and storage. In the row sequential algorithm by J.A. George and M.T. Heath [14], this problem is avoided by using a row-oriented method employing Givens rotations. Even more efficient are *multifrontal methods*, in which Householder transformations are applied to a sequence of small dense subproblems.

Note that in most sparse QR algorithms the orthogonal factor Q is not stored. The corrected seminormal equations are used for treating additional right-hand sides. The reason is that for rectangular matrices A the matrix Q is usually much less sparse than R. In the multifrontal algorithm, however, Q can efficiently be represented by the Householder vectors of the frontal orthogonal transformations, see [26].

A Fortran multifrontal sparse QR subroutine, called QR27, has been developed by P. Matstoms [28]. He [29] has also developed a version of this to be used with MATLAB, implemented as four M-files and available from netlib.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- ABS Algorithms for Optimization
- Gauss, Carl Friedrich
- Gauss–Newton Method: Least Squares, Relation to Newton's Method
- Generalized Total Least Squares
- Least Squares Orthogonal Polynomials
- Nonlinear Least Squares: Newton-type Methods
- Nonlinear Least Squares Problems
- Nonlinear Least Squares: Trust Region Methods

References

1. Benoit C (1924) Sur la méthode de résolution des, équationes normales, etc. (Procédés du commandant Cholesky). Bull Géodésique 2:67–77
2. Björck Å (1987) Stability analysis of the method of seminormal equations for least squares problems. Linear Alg & Its Appl 88/89:31–48
3. Björck Å (1994) Numerics of Gram–Schmidt orthogonalization. Linear Alg & Its Appl 197-198:297–316
4. Björck Å (1996) Numerical methods for least squares problems. SIAM, Philadelphia
5. Bunch JR, Nielsen CP (1978) Updating the singular value decomposition. Numer Math 31:111–129

6. Chambers JM (1971) Regression updating. J Amer Statist Assoc 66:744–748
7. Chan TF (1987) Rank revealing {QR}-factorizations. LAA 88/89:67–82
8. Cox AJ, Higham NJ (1997) Stability of Householder QR factorization for weighted least squares problems. Numer Anal Report Manchester Centre Comput Math, Manchester, England, 301
9. Cox MG (1990) The least-squares solution of linear equations with block-angular observation matrix. In: Cox MG, Hammarling SJ (eds) Reliable Numerical Computation. Oxford Univ. Press, Oxford, pp 227–240
10. Daniel J, Gragg WB, Kaufman L, Stewart GW (1976) Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization. Math Comput 30:772–95
11. Gauss CF (1880) Theoria combinationis observationum erroribus minimis obnoxiae, pars posterior. In: Werke, IV. Königl. Gesellschaft Wissenschaft, Göttingen, pp 27–53, First published in 1823.
12. Gauss CF (1880) Theoria combinationis observationum erroribus minimis obnoxiae, pars prior. In: Werke, IV. Königl. Gesellschaft Wissenschaft. Göttingen, Göttingen, pp 1–26, First published in 1821.
13. Gauss CF (1963) Theory of the motion of the heavenly bodies moving about the Sun in conic sections. Dover, Mineola, NY (Translation by Davis CH); first published in 1809
14. George JA, Heath MT (1980) Solution of sparse linear least squares problems using Givens, rotations. Linear Alg & Its Appl 34:69–83
15. George JA, Liu JW-H (1981) Computer solution of large sparse positive definite systems. Prentice-Hall, Englewood Cliffs, NJ
16. George JA, Liu JW-H (1989) The evolution of the minimum degree ordering algorithm. SIAM Rev 31:1–19
17. Gill PE, Golub GH, Murray W, Saunders MA (1974) Methods for modifying matrix factorizations. Math Comput 28:505–535
18. Golub GH (1965) Numerical methods for solving least squares problems. Numer Math 7:206–216
19. Golub GH, Manneback P, Toint P (1986) A comparison between some direct and iterative methods for large scale geodetic least squares problems. SIAM J Sci Statist Comput 7:799–816
20. Golub GH, Plemmons RJ (1980) Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. Linear Alg & Its Appl 34:3–28
21. Hansen PC (1998) Rank-deficient and discrete ill-posed problems. Numerical aspects of linear inversion. SIAM, Philadelphia
22. Van Huffel S, Vandewalle J (1991) The total least squares problem: Computational aspects and analysis. Frontiers in Appl Math, vol 9. SIAM, Philadelphia
23. Kourouklis S, Paige CC (1981) A constrained approach to the general Gauss–Markov, linear model. J Amer Statist Assoc 76:620–625
24. Lawson CL, Hanson RJ (1974) Solving least squares problems. Prentice-Hall, Englewood Cliffs, NJ
25. Legendre AM (1805) Nouvelles méthodes pour la détermination des orbites des comètes. Courcier, Paris
26. Lu S-M, Barlow JL (1996) Multifrontal computation with the orthogonal factors of sparse matrices. SIAM J Matrix Anal Appl 17:658–679
27. Manneback P, Murigande C, Toint PL (1985) A modification of an algorithm by Golub and Plemmons for large linear least squares in the context of Doppler positioning. IMA J Numer Anal 5:221–234
28. Matstoms P (1992) QR27-specification sheet. Techn. Report Dept. Math. Linköping Univ.
29. Matstoms P (1994) Sparse QR factorization in MATLAB. ACM Trans Math Softw 20:136–159
30. Penrose R (1955) A generalized inverse for matrices. Proc Cambridge Philos Soc 51:406–413
31. Reichel L, Gragg WB (1990) FORTRAN subroutines for updating the QR decomposition. ACM Trans Math Softw 16:369–377
32. Reid JK (1967) A note on the least squares solution of a band system of linear equations by Householder reductions. Computer J 10:188–189
33. Stewart GW (1992) An updating algorithm for subspace tracking. IEEE Trans Signal Processing 40:1535–1541
34. Wedin P-Å (1973) Perturbation theory for pseudo-inverses. BIT 13:217–232

Leibniz, Gottfried Wilhelm

Sandra Duni Eksioglu
Industrial and Systems Engineering Department, University Florida, Gainesville, USA

MSC2000: 01A99

Article Outline

Keywords
See also
References

Keywords

Gottfried Wilhelm Leibniz; Integration; Differentiation; Theory of envelops; Infinitesimal calculus

G.W. Leibniz (1646–1716) was a well-known German philosopher and mathematician. He is considered a de-

scendant of German idealism and a pioneer of the Enlightenment. Leibniz is known as the inventor of the differential and integral calculus [7].

Leibniz's contribution in philosophy is as significant as in mathematics. In philosophy Leibniz is known for his fundamental philosophical ideas and principles including truth, necessary and contingent truths, possible worlds, the principle of sufficient reason (i. e., there is a reason behind everybody's action), the principle of pre-established harmony (i. e., the universe is created in such a way that corresponding mental and physical events occur simultaneously), and the principle of noncontradiction (i. e., if a contradiction can be derived from a proposition, this proposition is false). Leibniz was fond on the idea that the principles of reasoning could be organized into a formal symbolic system, an algebra or calculus of thought, where disagreements could be settled by calculations [4].

Leibniz was the son of a professor of moral philosophy at Leipzig Univ. Leibniz learned to read from his father before going to school. He taught himself Latin and Greek by age 12, so that he could read the books in his father's library. He studied law at the Univ. of Leipzig from 1661 to 1666. In 1666 he was refused the degree of doctor of laws at Leipzig. He went to the Univ. of Altdorf, which awarded him doctorate in jurisprudence in 1667 [1].

Leibniz started his career at the courts of Mainz where he worked until 1672. The Elector of Mainz promoted him to diplomatic services. In 1672 he visited Paris to try to dissuade Louis XIV from attacking German areas. Leibniz remained in Paris until 1676, where he continued to practice law. In Paris he studied mathematics and physics under Chr. Huygens. During this period he developed the basic features of his version of the calculus. He spent the rest of his life, from 1676 until his death (November 14, 1716) at Hannover [6].

Leibniz's most important achievement in mathematics was the discovery of *infinitesimal calculus*. The significance of calculus is so important that it was marked as the starting point of modern mathematics. Leibniz's formulations were different from previous investigation by I. Newton. Newton was mainly concentrated in the geometrical representation of calculus, while Leibniz took it towards analysis. Newton considered variables changing with time. Leibniz thought of variables x, y as ranging over sequences of infinitely close values. For Newton integration and differentiation were inverses, while Leibniz used *integration* as a summation. At that time, neither Leibniz nor Newton thought in terms of functions, both always thought in terms of graphs.

In November 1675 he wrote a manuscript using the notation $\int f(x)\,dx$ for the first time [5]. In the same manuscript he presented the product rule for *differentiation*. The quotient rule first appeared two years later, in July 1677. In 1676 Leibniz arrived in the conclusion that he was in possession of a method that was highly important because of its generality. Whether a function was rational or irrational, algebraic or transcendental (a word that Leibniz coined), his operations of finding sums and differences could always be applied.

In November 1676 Leibniz discovered the familiar notation $d(x^n) = nx^{n-1}\,dx$ for both integral and fractional n. Newton claimed that: 'not a single previously unsolved problem was solved here', but the formalism of Leibniz's approach proved to be vital in the development of the calculus. Leibniz never thought of the derivative as a limit. This does not appear until the work of J. d'Alembert. Leibniz was convinced that good mathematical notations were the key to progress so he experimented with different notation for coefficient systems. His language was fresh and appropriate, incorporating such terms as *differential*, *integral*, *coordinate* and *function* [8]. His notations which we still use today, were clear and elegant. His unpublished manuscripts contain more than 50 different ways of writing coefficient systems, which he worked on during a period of 50 years beginning in 1678.

Leibniz used the word *resultant* for certain combinatorial sums of terms of a determinant. He proved various results on resultants including what is essentially Cramer's rule. He also knew that a determinant could be expanded using any column, what is now called Laplace expansion. As well as studying coefficient systems of equations which led him to determinants, Leibniz also studied coefficient systems of quadratic forms which led naturally towards matrix theory [9]. He thought about continuity, space and time [2].

In 1684 Leibniz published details of his differentiable calculus in 'Acta Eruditorum', a journal established in Leipzig two years earlier. He described a general method for finding *maxima* and *minima*, and drawing tangents to curves. The paper contained the

rules for computing the derivatives of powers, products and quotient.

In 1686 Leibniz published a paper on the principles of new calculus [3] in 'Acta Eruditorum'. Leibniz emphasized the inverse relationship between differentiation and integration in the fundamental theorem of calculus.

In 1692 Leibniz wrote a paper that set the basis of the *theory of envelopes*. This was further developed in another paper published on 1694 where he introduced for the first time the terms *coordinates* and *axes of coordinates*.

Leibniz published many papers on mechanical subjects as well [1]. In 1700 Leibniz founded the Berlin Academy and was its first president.

Leibniz's principal works are:

1) 'De Arte Combinatoria' (On the Art of Combination), 1666;
2) 'Hypothesis Physica Nova' (New Physical Hypothesis), 1671;
3) 'Dicours de Metaphysique' (Discourse on Metaphysics), 1686;
4) Unpublished Manuscripts on the Calculus of Concepts, 1690;
5) 'Nouveaux Essais sur L'entendement Humaine' (New Essays on Human Understanding), 1705;
6) 'Theodicee' (Theodicy), 1710;
7) 'Monadologia' (The Monadology), 1714.

See also

► History of Optimization

References

1. Aiton EJ (1985) Leibniz, A biography. Adam Hilger Ltd, Bristol
2. Anapolitanos D (1999) Leibniz: Representation, continuity and the spatiotemporal. Kluwer, Dordrecht
3. Boyer BB (1968) A history of mathematics. Wiley, New York
4. MacDonald GR (1984) Leibniz. Oxford Univ. Press, Oxford
5. O'Connor JJ (Oct. 1998) Gottfried Wilhelm von Leibniz. Dept. Math. and Statist. Univ. St. Andrews, Scotland), http://www.history.mcs.st-andrews.ac.uk/history/Mathematicians/Leibniz.html
6. Pereira ME (2000) Gottfried Wilhelm von Leibniz. http://www.geocities.com/Athens/Delphi/6061/Leibniz.html
7. Wingereid B (2000) Gottfried Wilhelm von Leibniz. http://www.phs.princeton.k12.oh.us/Public/Lessons/enl/wing.html
8. Woolhouse RS (ed) (1981) Leibniz: Metaphysics and philosophy of science. Oxford Univ. Press, Oxford
9. Zalta EN (2000) Gottfried Wilhelm von Leibniz. http://mally.stanford.edu/leibniz

Lemke Method

Lemke Algorithm

MICHAEL M. KOSTREVA
Department Math. Sci., Clemson University, Clemson, USA

MSC2000: 90C33

Article Outline

Keywords
Lemke's Algorithm
See also
References

Keywords

Linear complementarity; Pivoting

The *linear complementarity problem* (LCP) is a well known problem in mathematical programming. Applications of the LCP to engineering, game theory, economics, and many other scientific fields have been found. The monograph of K.G. Murty [8] is a compendium of LCP developments. One of the most significant approaches to the solution of the linear complementarity problem is called Lemke's method or Lemke's algorithm. Two descriptions of the algorithm [6,7] provide many algorithmic proofs and details for the interested reader. Our treatment here is a sketch of the algorithm, together with pointers to related work in the literature.

There are some important related works for those who wish to solve LCP. A. Ravindran [10] provided a FORTRAN implementation of Lemke's algorithm in a set-up similar to the revised simplex method. C.B. Garcia [2] described some classes of matrices for which the associated LCPs can be solved by Lemke's algorithm. J.J.M. Evers [1] enlarged the range of application

of Lemke's algorithm, and showed that it could solve the bimatrix game. P.M. Pardalos and J.B. Rosen [9] presented a global optimization approach to LCP. D. Solow and P. Sengupta [11] proposed a finite descent theory for the linear complementarity problem. M.M. Kostreva [4] showed that without the *nondegeneracy assumption*, Lemke's algorithm may *cycle*, and showed that the minimum length of such a cycle is four.

The linear complementarity problem considered is: Given an $(n \times n)$-matrix M and an $(n \times 1)$ column vector q, problem LCP(q, M) is to find x (or prove that no such x exists) in $\mathbf{R}^n$ satisfying

$$y = Mx + q, \tag{1}$$

$$y_i \geq 0, \tag{2}$$

$$x_i \geq 0, \tag{3}$$

$$y_i \cdot x_i = 0, \tag{4}$$

for all i, $i = 1, \ldots, n$.

Clearly these conditions are equivalent to $y^\intercal x = 0$. The variables (y_i, x_i) are called a *complementary pair of variables*. Lemke's algorithm is organized relative to the following extended system of equations:

$$y = Mx + q + x_0 d, \tag{5}$$

where d is an $(n \times 1)$ column vector, and $x_0 \geq 0$. Relative to the vector d, it is only required that $(q + x_0 d) \geq 0$ for some $x_0 \geq 0$. It is assumed that the system of equations (5) is nondegenerate, that is, any solution has at most $n + 1$ zero values among the variables (y, x, x_0).

Lemke's Algorithm

If $q > 0$, terminate with a complementary feasible solution, $y = q$, $x = 0$.

If q has some negative component, then on the first pivot x_0 is increased until for the first time $y = q + x_0 d \geq 0$. When this occurs, some y variable, say y_r becomes zero. The first pivot is to exchange the variables x_0 and y_r. Now the variable x_0 is basic, and the variables y_r and x_r are two complementary non basic variables. If a pivot can be made on variable x_r (complement of the most recently pivoted member of the complementary pair), then it leads to another similar situation with another pair of complementary variables. If a pivot cannot be made, the sequence is terminated. If the variable x_0 becomes non basic (zero), a solution is at hand. If not, the pivoting continues uniquely, with each new set of equations containing a non basic complementary pair of variables, one of which is most recently made non basic. Due to the unique choices of pivot row and pivot column, finite termination must occur.

Under certain conditions, including the positive semidefinite matrices, the condition of termination without finding a pivot (also called secondary ray termination) can be shown to imply that the set $\{x: y = Mx + q \geq 0, \quad x \geq 0\}$ is empty. Under such conditions, Lemke's algorithm is said to *process the LCP*: either it is solved, or it is shown not to have a feasible solution. The set of all LCPs which Lemke's algorithm will process is unknown, but some recent papers shed light on its processing domain. Kostreva and M.M. Wiecek [5] use a multiple objective optimization approach which eventually results in a larger dimensioned LCP, while G. Isac, Kostreva and Wiecek [3] point out a set of problems which is impossible for Lemke's method to process.

Example 1 Consider the LCP corresponding to the quadratic programming problem

$$\begin{cases} \min & x_1^2 - 2x_1x_2 + x_2^2 + 3x_1 + x_2 \\ \text{s.t.} & 3x_1 + x_2 \geq 4 \\ & x_1 \geq 0,\ x_2 \geq 0. \end{cases}$$

Then $q = (-4, 3, 1)^\intercal$ and $M = [(0, -3, -1)^\intercal, (3, 2, -2)^\intercal, (1, -2, 2)^\intercal]$, and Lemke's algorithm requires four pivots to obtain the solution $x^* = (1, 1)^\intercal$, using the vector $d = (1, 1, 1)^\intercal$. It is noteworthy that the nondegeneracy assumption is not satisfied in this example, but Lemke's algorithm works anyway.

See also

- ► Convex-simplex Algorithm
- ► Linear Complementarity Problem
- ► Linear Programming
- ► Parametric Linear Programming: Cost Simplex Algorithm
- ► Sequential Simplex Method

References

1. Evers JJM (1978) More with the Lemke complementarity algorithm. Math Program 15:214–219
2. Garcia CB (1973) Some classes of matrices in linear complementarity theory. Math Program 5:299–310
3. Isac G, Kostreva MM, Wiecek MM (1995) Multiple objective approximation of feasible but unsolvable linear complementarity problems. J Optim Th Appl 86:389–405
4. Kostreva MM (1979) Cycling in linear complementarity problems. Math Program 16:127–130
5. Kostreva MM, Wiecek MM (1993) Linear complementarity problems and multiple objective programming. Math Program 60:349–359
6. Lemke CE (1965) Bimatrix equilibrium points and mathematical programming. Managem Sci 11:681–689
7. Lemke CE (1968) On complementary pivot theory in mathematics of the decision sciences. In: Dantzig GB, Veinott AF (eds) Amer. Math. Soc.
8. Murty KG (1988) Linear complementarity, linear and nonlinear programming. Heldermann, Berlin
9. Pardalos PM, Rosen JB (1988) Global optimization approach to the linear complementarity problem. SIAM J Sci Statist Comput 9:341–353
10. Ravindran A (1972) Algorithm 431-H. A computer routine for quadratic and linear programming problems. Comm ACM 15:818–820
11. Solow D, Sengupta P (1985) A finite descent theory for linear programming, piecewise linear minimization and the linear complementarity problem. Naval Res Logist Quart 32:417–431

Lexicographic Pivoting Rules

LexPr

TAMÁS TERLAKY
Department Comput. & Software,
McMaster University, West Hamilton, Canada

MSC2000: 90C05, 90C20, 90C33, 05B35, 65K05

Article Outline

Keywords
Lexicographic Simplex Methods
- Lexicographic Ordering
- The Lexicographic Primal Simplex Method
- The Use of Lexicographic Ordering
- Lexicographic Ordering and Perturbation
- Lexicographic Dual Simplex Method
- Extensions

Lexicography and Oriented Matroids
See also
References

Keywords

Pivot rules; Anticycling; Lexicographic ordering; LP; LCP; Oriented matroids

The general linear optimization (LO), linear programming (cf. ► Linear programming), problem will be considered in the standard primal form

$$\min\left\{c^\top x:\ Ax = b,\ x \geq 0\right\},$$

together with its standard dual

$$\max\left\{b^\top y:\ A^\top y \leq c\right\}.$$

One of the most efficient, and for a long time the only, practical method to solve LO problems was the simplex method of G.B. Dantzig. The *simplex method* is a *pivot algorithm* that traverses through feasible *basic solutions* while the objective value is improving. The simplex method is practically one of the most efficient algorithms but it is theoretically a finite algorithm only for *nondegenerate problems*.

A basis is called *primal degenerate* if at least one of the basic variables is zero; it is called *dual degenerate* if the reduced cost of at least one nonbasic variable is zero. In general, the basis is degenerate if it is either primal or dual, or both primal and dual degenerate. The LO problem is degenerate, if it has a degenerate basis. A pivot is called degenerate when after the pivot the objective remains unchanged. When the problem is *degenerate* the objective might stay the same in subsequent iterations and the simplex algorithm may *cycle*, i. e. starting from a basis, after some iterations the same basis is revisited and this process is repeated endlessly. Because the simplex method produces a sequence with monotonically improving objective values, the objective stays constant in a cycle, thus each pivot in the cycle must be degenerate. The possibility of cycling was recognized shortly after the invention of the simplex algorithm. Cycling examples were given by E.M.L. Beale [2] and by A.J. Hoffman [10]. Recently (1999) a scheme to construct cycling LO examples is presented in [9]. These examples made evident that extra techniques are needed to ensure finite termination of simplex methods. The first and widely used such tool is the *lexico-*

graphic simplex rule. Other techniques, like the least-index anticycling rules (cf. ► Least-index anticycling rules) and more general recursive schemes were developed more recently.

Lexicographic Simplex Methods

First we need to define an ordering, the so-called *lexicographic ordering* of vectors.

Lexicographic Ordering

An n-dimensional vector $u = (u_1, \dots, u_n)$ is called *lexicographically positive* or, in other words, *lexico-positive* if its first nonzero coordinate is positive, i. e. for a certain $j \leq n$ one has $u_i = 0$ for $i < j$ and $x_u > 0$. Observe, that the zero vector is the only lexico-nonnegative vector which is not lexico-positive. The vector u^0 is said to be lexicographically smaller than a vector u^1 when the difference $u^1 - u^0$ of the two vectors is lexico-positive. Further, if a finite set of vectors $\{u^0, \dots, u^k\}$ is given, then the vector u^0 is said to be lexico-minimal in the given set, when u^0 is lexicographically smaller than u^i for all $1 \leq i \leq k$.

The Lexicographic Primal Simplex Method

Cycling of the simplex method is possible only when the LO problem is degenerate. In that case possibly many variables are eligible to enter and to leave the basis. The lexicographic primal simplex rule makes the selection of the leaving variable uniquely determined when the entering variable is already chosen.

The Use of Lexicographic Ordering

At start a feasible *lexico-positive basis tableau* is given. A basis tableau is called lexico-positive if, except the reduced cost row, all of its row vectors are lexico-positive. Any feasible basis tableau can be made lexico-positive by a simple rearrangement of its columns. Specifically, we can take the solution column as the first one, and then take the current basic variables, in an arbitrary order, followed by the nonbasic variables, again in an arbitrary ordering.

The following lexicographic simplex pivot selection rule was first proposed by Dantzig, A. Orden and P. Wolfe [7].

0	Initialization. Let $T(B)$ be a given primal feasible lexico-positive basis tableau. (Fix the order of the variables.)
1	Entering variable selection. Choose a dual infeasible variable, i.e. one with negative reduced cost. Let its index be q. IF no such variable exists, THEN STOP; The tableau $T(B)$ is optimal and this way a pair of optimal solutions is obtained.
2	Leaving variable selection. Collect in column q all the candidate pivot elements that satisfy the usual pivot selection conditions of the primal simplex method. Let $K = \{i_1, \dots, i_k\}$ be the set of the indices of the candidate leaving variables. IF there is no pivot candidate, THEN STOP; The primal problem is unbounded, and so the dual problem is infeasible. IF there is a unique pivot candidate $\{p\} = K$ to leave the basis, THEN go to Step 3. IF there are more pivot candidates, THEN look at the row vectors t^i, $i \in K$, of the basis tableau (note that by construction x_i is the first coordinate of t^i). Let p be the pivot row if t^p is lexico-minimal in this set of row vectors.
3	Basis transformation. Pivot on (p, q). Go to Step 1.

The lexicographic primal simplex rule

The following two observations are important. First note that lexicographic selection plays role only when the leaving variable is selected. In that case some rows of the tableau are compared in the lexicographic ordering. If the basis variables were originally out right after the solution column, as proposed in order to get a lexico-positive initial tableau, then this comparison is already decided when one considers only the columns corresponding to the initial basis. This claim holds, because those columns form a basis, thus the related row vectors are linearly independent as well.

On the other hand, when the initial basis is the unit matrix, then at each pivot the basis inverse can be found, in the place of the initial unit matrix. When these

two observations are put together, it can be concluded that instead of using the rows of the basis tableau, the rows of the basis inverse headed by the corresponding solution coordinate, can be used in Step 2. to determine the unique leaving variable. As a consequence one do not need to calculate and store the complete basis tableau when implementing the lexicographic pivot rule. The solution and the basis inverse provide all the necessary information.

The lexicographic simplex method is finite. The finiteness proof is based on the following simple properties: There is a finite number of different basis tableaus. The first row of the tableau, i. e. the vector, having the objective value as its first coordinate followed by the reduced cost vector, strictly increases lexicographically at each iteration. This fact ensures that no basis can be revisited, thus cycling is impossible.

Lexicographic Ordering and Perturbation

Independent of [7], A. Charnes [4] developed a technique of *perturbation*, that resulted in a finite simplex algorithm. This algorithm turned out to be equivalent to the lexicographic rule. The perturbation technique is as follows. Let ϵ be a sufficiently small number. Let us replace b_i by $b_i + \sum_j a_{ij}\epsilon^j$ for all i. If ϵ is small enough then the resulted problem is nondegenerate. Moreover, starting from a given primal feasible basis, the primal simplex method applied to the new problem produces exactly the same pivot sequence as the lexicographic simplex method on the original problem.

In particular, when the problem is initialized with a feasible basis solution, it suffices to use the perturbation $b_i + \epsilon^i$. This way only the basis part of the coefficient matrix is used in Charnes' perturbation technic.

An appealing property of the perturbation technique is that actually it is not needed to perform the perturbation with a concrete ϵ. It can be done symbolically.

Lexicographic Dual Simplex Method

The dual simplex method is nothing else, than the primal simplex method applied to the dual problem, when the dual problem is brought in the primal standard form. This way it is straightforward to develop the lexicographic, or the equivalent perturbation technique for the dual simplex method.

Extensions

The lexicographic rule is extensively used in proving finiteness of pivot algorithms, see e. g. [1] for an application in a monotonic build-up scheme, [14] for further references in LO and [5] for references when lexicographic degeneracy resolution is applied for complementarity problems.

Lexicography and Oriented Matroids

Based on the perturbation interpretation, analogous lexicographic techniques and lexicographic pivoting rules were developed for oriented matroid programming [3] (cf. also ► Oriented matroids). These techniques were particularly interesting, because nondegenerate cycling [3,8] is possible in oriented matroids. An apparent difference between the linear and the oriented matroid context is that for oriented matroids none of the finite – recursive or least index type – rules yield a simplex method, i. e. a pivot method that preserves feasibility of the basis throughout. This discrepancy is also due to the possibility of nondegenerate cycling.

Interestingly, in the case of oriented matroid programming the finite lexicographic method of M.J. Todd [15,16] is the only one which preserves feasibility of the basis and therefore yields a finite simplex algorithm for oriented matroids.

The equivalence of Dantzig's self—dual parametric algorithm [6] and Lemke's complementary pivot algorithm [11,12] applied to the linear complementarity problem (cf. also ► Linear complementarity problem) defined by the primal and dual LO problem was proved by I. Lustig [13]. Todd's lexicographic pivot rule is essentially a lexicographic Lemke method (or the parametric perturbation method), when applied to the specific linear complementary problem defined by the primal-dual pair of LO problems. Hence, using the equivalence mentioned above a simplex algorithm for LO can be derived. However, it is more complicated to present this method in the linear optimization than in the complementarity context. Now Todd's rule will be sketched for the linear case.

0	Initialization. Let a lexico-positive feasible tableau $T(B)$ be given.
1	Entering variable selection. Collect all the dual infeasible variables as the set of candidate entering variables. Let their set of indices be denoted by K_D. IF no such variable exists, THEN STOP; The tableau $T(B)$ is optimal and this way a pair of optimal solutions is obtained. IF there is a unique $\{q\} = K_D$ candidate to enter the basis, THEN go to Step 2. IF there are more pivot candidates, THEN let q be the index of that variable whose column is lexico-minimal in the set K_D. (Analogous to the dual lexicographic simplex selection rule).
2	Leaving variable selection. Collect in column q all the candidate pivot elements that satisfy the usual pivot selection conditions of the primal simplex method. Let K_P be the set of the indexes of the candidate leaving variables. IF there is no pivot candidate, THEN STOP; the primal problem is unbounded, and so the dual problem is infeasible. IF there is a unique $\{p\} = K_P$ pivot candidate to leave the basis, THEN go to Step 3. IF there are more pivot candidates, THEN let p be the index of that variable whose row is lexico-minimal in the set K_P. (Analogous to the primal lexicographic simplex selection rule.)
3	Basis transformation. Pivot on (p, q). Go to Step 1.

Todd's lexicographic Lemke rule (Phase II)

In Todd's rule the perturbation is done first in the right-hand side and then in the objective (with increasing order of the perturbation parameter ϵ). It finally gives a two phase simplex method. For illustration only the second phase [14] is presented here. Complete description of the algorithm can be found in [3,16].

This algorithm is not only a unique simplex method for oriented matroids, but it is a novel application of lexicography in LO as well.

See also

- ▶ Criss-cross Pivoting Rules
- ▶ Least-index Anticycling Rules
- ▶ Linear Programming
- ▶ Pivoting Algorithms for Linear Programming Generating Two Paths
- ▶ Principal Pivoting Methods for Linear Complementarity Problems
- ▶ Probabilistic Analysis of Simplex Algorithms

References

1. Anstreicher KM, Terlaky T (1994) A monotonic build-up simplex algorithm. Oper Res 42:556–561
2. Beale EML (1955) Cycling in the dual simplex algorithm. Naval Res Logist Quart 2:269–275
3. Bjorner A, Las Vergnas M, Sturmfels B, White N, Ziegler G (1993) Oriented matroids. Cambridge Univ. Press, Cambridge
4. Charnes A (1952) Optimality and degeneracy in linear programming. Econometrica 20(2):160–170
5. Cottle R, Pang JS, Stone RE (1992) The linear complementarity problem. Acad. Press, New York
6. Dantzig GB (1963) Linear programming and extensions. Princeton Univ. Press, Princeton
7. Dantzig GB, Orden A, Wolfe P (1955) Notes on linear programming: Part I – The generalized simplex method for minimizing a linear form under linear inequality restrictions. Pacific J Math 5(2):183–195
8. Fukuda K (1982) Oriented matroid programming. PhD Thesis Waterloo Univ.
9. Hall J, McKinnon KI (1998) A class of cycling counterexamples to the EXPAND anti-cycling procedure. Techn. Report Dept. Math. Statist. Univ. Edinburgh
10. Hoffman AJ (1953) Cycling in the simplex method. Techn Report Nat Bureau Standards 2974
11. Lemke CE (1965) Bimatrix equilibrium points and mathematical programming. Managem Sci 11:681–689
12. Lemke CE (1968) On complementary pivot theory. In: Dantzig GB, Veinott AF (eds) Mathematics of the Decision Sci. Part I. Lect Appl Math 11. Amer. Math. Soc., Providence, RI, pp 95–114
13. Lustig I (1987) The equivalence of Dantzig's self-dual parametric algorithm for linear programs to Lemke's algorithm for linear complementarity problems applied to linear programming. SOL Techn Report Dept Oper Res Stanford Univ 87(4)
14. Terlaky T, Zhang S (1993) Pivot rules for linear programming: A survey on recent theoretical developments. Ann Oper Res 46:203–233
15. Todd MJ (1984) Complementarity in oriented matroids. SIAM J Alg Discrete Meth 5:467–485
16. Todd MJ (1985) Linear and quadratic programming in oriented matroids. J Combin Th B 39:105–133

Linear Complementarity Problem

RICHARD W. COTTLE
Stanford University, Stanford, USA

MSC2000: 90C33

Article Outline

Keywords
Synonyms
Definition
Sources of Linear Complementarity Problems
Equivalent Formulations
The Importance of Matrix Classes
Algorithms for Solving LCPs
Software
Some Generalizations
See also
References

Keywords

Quadratic programming; Bimatrix games; Matrix classes; Equilibrium problems

Synonyms

LCP

Definition

In its standard form, a linear complementarity problem (LCP) is an inequality system stated in terms of a mapping $f\colon \mathbf{R}^n \to \mathbf{R}^n$ where $f(x) = q + Mx$. Given f, one seeks a vector $x \in \mathbf{R}^n$ such that for $i = 1, \ldots, n$,

$$x_i \geq 0, \quad f_i(x) \geq 0, \quad \text{and} \quad x_i f_i(x) = 0. \tag{1}$$

Because the affine mapping f is specified by the vector $q \in \mathbf{R}^n$ and the matrix $M \in \mathbf{R}^{n\times n}$, the problem is ordinarily denoted LCP(q, M) or sometimes just (q, M). A system of the form (1) in which f is not affine is called a *nonlinear complementarity problem* and is denoted NCP(f). The notation CP(f) is meant to cover both cases.

If $\overline{x}$ is a solution to (1) satisfying the additional *nondegeneracy condition* $\overline{x}_i + f_i(\overline{x}) > 0$, $i = 1, \ldots, n$, the indices i for which $\overline{x}_i > 0$ or $f_i(\overline{x}) > 0$ form complementary subsets of $\{1, \ldots, n\}$. This is believed to be the origin of the term *complementary slackness* as used in linear and nonlinear programming. It was this terminology that inspired the name *complementarity problem*.

Sources of Linear Complementarity Problems

The linear complementarity problem is associated with the Karush–Kuhn–Tucker necessary conditions of local optimality found in quadratic programming. This connection (as well as the more general connection of nonlinear complementarity problems with other types of nonlinear programs) was brought out in [1,2] and later in [3]. Finding solutions to such systems was one of the original motivations for studying the subject. Another was the finding of equilibrium points in bimatrix and polymatrix games. This kind of application was emphasized in [16] and [22]. These early contributions also included essentially the first algorithms for this class of problems. There are numerous applications of the linear and nonlinear complementarity problems in computer science, economics, various engineering disciplines, finance, game theory, and mathematics. One application of the LCP is in algorithms for the nonlinear complementarity problem. Descriptions of (and references to) these applications can be found in [5,27] and [17]. The survey article [10] is a rich compendium on engineering and economic applications of linear and nonlinear complementarity problems.

Equivalent Formulations

Whether linear or nonlinear, the complementarity problem expressed by the system (1) can be formulated in several equivalent ways. An obvious one calls for a solution (x, y) to the system

$$y - f(x) = 0, \quad x \geq 0, \quad x^\top y = 0. \tag{2}$$

Another is to find a zero x of the mapping

$$g(x) = \min\{x, f(x)\}, \tag{3}$$

where the symbol min $\{a, b\}$ denotes the componentwise minimum of the two n-vectors a and b. A third equivalent formulation asks for a fixed point of the mapping

$$h(x) = x - g(x),$$

that is, a vector $x \in \mathbf{R}^n$ such that $x = h(x)$.

The formulation given in (3) is related to the (often nonconvex) optimization problem:

$$\begin{cases} \min & x^\top f(x) \\ \text{s.t.} & f(x) \geq 0 \\ & x \geq 0. \end{cases} \tag{4}$$

In such a problem, the objective is bounded below by zero, thus any feasible solution of (4) for which the objective function $x^\intercal f(x) = 0$ must be a global minimum as well as a solution of (1). As it happens, there are circumstances (for instance, the monotonicity of the mapping f) under which all the local minima for the mathematical programming problem (4) must in fact be solutions of (3). See [28] for an extended discussion of this matter.

Also noteworthy is a result in [8] showing that the LCP is equivalent to solving a system of equations $y = \varphi(x)$ where the mapping $\varphi: \mathbf{R}^n \to \mathbf{R}^n$ is piecewise linear. In particular, LCP(q, M) is equivalent to finding a vector u such that

$$q + Mu^+ - u^- = 0,$$

where (for $i = 1, \ldots, n$, $u_i^+ = \max\{0, u_i\}$ and $u_i^- = -\min\{0, u_i\}$.

The Importance of Matrix Classes

The extensive literature of the LCP exhibits several main directions of study: the existence and uniqueness (or number of) solutions, mathematical properties of the problem, generalizations of the problem, algorithms, applications, and implementations.

Much of the theory of the linear complementarity problem is intimately linked in various ways to matrix classes. For instance, one of the earliest theorems on the existence of solutions to LCPs is due H. Samelson, R.M. Thrall and O. Wesler [30]. Motivated by a problem in structural mechanics, they showed that the LCP(q, M) has a unique solution for every $q \in \mathbf{R}^n$ if and only if the matrix M has positive principal minors. (That is, the determinant of every principal submatrix of M is positive.) The class of such matrices has come to be known as **P**, and its members are called **P**-*matrices*. (The Samelson–Thrall–Wesler theorem characterizes this class of matrices in terms of the LCP.) The class **P** includes all *positive definite* (**PD**) matrices, i. e., those square matrices M for which $x^\intercal Mx > 0$ for all $x \neq 0$. In the context of the LCP, the term **PD** does not require symmetry. An analogous definition (and usage) holds for *positive semidefinite* (**PSD**) matrices, namely, M is **PSD** if $x^\intercal Mx \geq 0$ for all x. Some authors refer to such matrices as *monotone* because of their connection with monotone mappings. **PSD**-matrices have the property that associated LCPs (q, M) are solvable whenever they are feasible, whereas LCPs (q, M) in which $M \in$ **PD** are always feasible and (since **PD**$\subset$**PSD**) are always solvable. This distinction is given a more general matrix form in [25,26]. There **Q** is defined as the class of all square matrices for which LCP(q, M) has a solution for all q and $\mathbf{Q}_0$ as the class of all square matrices for which LCP(q, M) has a solution whenever it is feasible. Although the goal of usefully characterizing the classes **Q** and $\mathbf{Q}_0$ has not yet been realized, much is known about some of their special subclasses. Indeed, there are now literally dozens of matrix classes for which LCP existence theorems have been established. See [5,27] and [17] for an abundance of information on this subject.

From the theoretical standpoint, the class of 'sufficient matrices' [6] illustrates the intrinsic role of matrix classes in the study of the LCP. A matrix $M \in \mathbf{R}^{n\times n}$ is *column sufficient* if

$$[x_i(Mx)_i \leq 0 \quad \forall i] \quad \Rightarrow \quad [x_i(Mx)_i = 0 \quad \forall i]$$

and *row sufficient* if $M^\intercal$ is column sufficient. When M is both row and column sufficient, it is called *sufficient*. Row sufficient matrices always have nonnegative principal minors, hence so do (column) sufficient matrices. These classes include both **P** and **PSD** as distinct subclasses. The row sufficient matrices form a subclass of $\mathbf{Q}_0$; this is not true of column sufficient matrices however. The column sufficient matrices $M \in \mathbf{R}^{n\times n}$ are characterized by the property that the solution set of LCP(q, M) is convex for every $q \in \mathbf{R}^n$. In the same spirit, a real $n \times n$ matrix M is row sufficient if and only if for every $q \in \mathbf{R}^n$, the solutions of the LCP(q, M) are precisely the optimal solutions of the associated quadratic program (4). Rather surprisingly, the class of sufficient matrices turns out to be identical to the matrix class $\mathbf{P}_*$ introduced in [19]. See [13] and [34].

Algorithms for Solving LCPs

The algorithms for solving linear complementarity problems are of two major types: pivoting (or, direct)

and iterative (or, indirect). Algorithms of the former type are finite procedures that attempt to transform the problem (q, M) to an equivalent system of the form (q', M') in which $q' \geq 0$. Doing this is not always possible; it depends on the problem data, usually on the matrix class (such as **P**, **PSD**, etc.) to which M belongs. When this approach works, it amounts to carrying out a *principal pivotal transformation* on the system of equations

$$w = q + Mz.$$

To such a transformation there corresponds an index set α (with complementary index set $\overline{\alpha} = \{1, \ldots, n\} \setminus \alpha$) such that the *principal submatrix* $M_{\alpha\alpha}$ is nonsingular. When this (block pivot) operation is carried out, the system

$$w_\alpha = q_\alpha + M_{\alpha\alpha} z_\alpha + M_{\alpha\overline{\alpha}} z_{\overline{\alpha}},$$
$$w_{\overline{\alpha}} = q_{\overline{\alpha}} + M_{\overline{\alpha}\alpha} z_\alpha + M_{\overline{\alpha}\overline{\alpha}} z_{\overline{\alpha}}$$

becomes

$$z_\alpha = q'_\alpha + M'_{\alpha\alpha} w_\alpha + M'_{\alpha\overline{\alpha}} z_{\overline{\alpha}},$$
$$w_{\overline{\alpha}} = q'_{\overline{\alpha}} + M_{\overline{\alpha}'\alpha} w_\alpha + M_{\overline{\alpha}'\overline{\alpha}} z_{\overline{\alpha}},$$

where

$$q'_\alpha = -M^{-1}_{\alpha\alpha} q_\alpha,$$
$$q'_{\overline{\alpha}} = q_{\overline{\alpha}} - M_{\overline{\alpha}\alpha} M^{-1}_{\alpha\alpha} q_\alpha,$$
$$M'_{\alpha\alpha} = M^{-1}_{\alpha\alpha},$$
$$M'_{\overline{\alpha}\alpha} = M_{\overline{\alpha}\alpha} M^{-1}_{\alpha\alpha},$$
$$M'_{\alpha\overline{\alpha}} = -M^{-1}_{\alpha\alpha} M_{\alpha\overline{\alpha}},$$
$$M'_{\overline{\alpha}\overline{\alpha}} = M_{\overline{\alpha}\overline{\alpha}} - M_{\overline{\alpha}\alpha} M^{-1}_{\alpha\alpha} M_{\alpha\overline{\alpha}}.$$

There are two main pivoting algorithms used in processing LCPs. The more robust of the two is due to C.E. Lemke [21]. *Lemke's method* embeds the LCP (q, M) in a problem having an extra 'artificial' nonbasic (independent) variable z_0 with coefficients specially chosen so that when z_0 is sufficiently large, all the basic variables become nonnegative. At the least positive value of z_0 for which this is so, there will (in the nondegenerate case) be (exactly) one basic variable whose value is zero. That variable is exchanged with z_0. Thereafter the method executes a sequence of (almost complementary) simple pivots. In each case, the variable *becoming* basic is the complement of the variable that became nonbasic in the previous exchange. The method terminates if either z_0 decreases to zero (in which case the problem is solved) or else there is no basic variable whose value decreases as the incoming nonbasic variable is increased. The latter outcome is called *termination on a secondary ray*. For certain matrix classes, termination on a secondary ray is an indication that the given LCP has no solution. Lemke's method is studied from this point of view in [7].

The other pivoting algorithm for the LCP is called the *principal pivoting method* (PPM), expositions of which are given in [3] and [5]. The algorithm two versions: symmetric and asymmetric. The former executes a sequence of principal (block) pivots or order 1 or 2, whereas the latter does sequences of almost complementary pivots, each of which results in a block principal pivot or order potentially larger than 2.

Iterative methods are often favored for the solution of very large linear complementarity problems. In such problems, the matrix M tends to be sparse (i. e., to have a small percentage of nonzero elements) and structured. Since iterative methods do not modify the problem data, these features of large scale problems can be used to advantage. Ordinarily, however, an iterative method does not terminate finitely; instead, it generates a convergent sequence of trial solutions. The older iterative LCP algorithms are based on equation-solving methods (e. g., *Gauss–Seidel*, *Jacobi*, and *successive overrelaxation*); the more contemporary ones are varieties of the *interior point* type. In addition to the usual concerns about practical performance, considerable interest attaches to the development of polynomial time algorithms. Not unexpectedly, the allowable analysis and applicability of iterative algorithms depend heavily on the matrix class to which M belongs. Details on several such algorithms are presented in [36,37], and the monographs [5,27] and [17].

Software

For decades researchers have experimented with computer codes for various linear (and nonlinear) complementarity algorithms. By the late 1990s, this activity reached the stage where the work could be distributed as something approaching commercial software. An overview of available software for complementarity problems (mostly nonlinear), is available as [35].

Some Generalizations

Both linear and nonlinear complementarity problems have been generalized in numerous ways. One of the earliest generalizations, given in [14] and [18], is the problem CP(K, f) of finding a vector x in the closed convex cone K such that $f(x) \in K^*$ (the dual cone) and $x^\intercal f(x) = 0$. Through this formulation, a connection can be made between complementarity problems and *variational inequality problems*, that is, problems VI(X, f) wherein one seeks a vector $x^* \in X$ (a nonempty subset of $\mathbf{R}^n$) such that

$$f(x^*)^\top (y - x^*) \geq 0 \quad \text{for all } y \in X.$$

It was established in [18] that when X is a closed convex cone, say K, with dual cone K^*, then CP(K, f) and VI(X, f) have exactly the same solutions (if any). See [15] for connections with variational inequalities, etc.

In [29] the generalized complementarity problem CP(K, f) defined above is considered as an instance of a *generalized equation*, namely to find a vector $x \in \mathbf{R}^n$ such that

$$0 \in f(x) + \partial \psi_K(x),$$

where ψ_K is the indicator function of the closed convex cone K and ∂ denotes the subdifferential operator as used in convex analysis.

Among the diverse generalizations of the linear complementarity problem, the earliest appears in [30]. There, for given $n \times n$ matrices A and B and n-vector c, the authors considered the problem of the finding n-vectors x and y such that

$$Ax + By = c, \quad x, y \geq 0 \quad \text{and} \quad x^\top y = 0.$$

A different generalization was introduced in [4]. In this sort of problem, one has an affine mapping $f(x) = q + Nx$ where N is of order $\sum_{j=1}^{k} p_j \times n$ partitioned into k blocks; the vectors q and $y = f(x)$ are partitioned conformably. Thus,

$$y^j = q^j + N^j x \quad \text{for } j = 1, \ldots, k.$$

The problem is to find a solution of the system

$$\begin{aligned} &y = q + Nx, \\ &x, y \geq 0, \\ &x_j \prod_{i=1}^{p_j} y_i^j = 0, \quad j = 1, \ldots, k. \end{aligned}$$

In recently years, many publications, e. g. [9] and [24], have further investigated this *vertical linear complementarity problem* (VLCP). Interest in the model which is at the heart of [30] and is now called the *horizontal linear complementarity problem* (HLCP) was revived in [38] where it is used as the conceptual framework for the convergence analysis of infeasible interior point methods. (The problem also comes up in [20].) In some cases, HLCPs can be reduced to ordinary LCPs. This subject is explored in [33] which gives an algorithm for doing this when it is possible. A further generalization called *extended linear complementarity problem* (ELCP) was introduced in [23] and subsequently developed in [11,12] and [32]. To this collection of LCP variants can be added the ELCP presented in [31]. The form of this model captures the previously mentioned HLCP, VLCP and ELCP.

See also

- ▶ Convex-simplex Algorithm
- ▶ Equivalence Between Nonlinear Complementarity Problem and Fixed Point Problem
- ▶ Generalized Nonlinear Complementarity Problem
- ▶ Integer Linear Complementary Problem
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Lemke Method
- ▶ Linear Programming
- ▶ Order Complementarity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Principal Pivoting Methods for Linear Complementarity Problems
- ▶ Sequential Simplex Method
- ▶ Splitting Method for Linear Complementarity Problems
- ▶ Topological Methods in Complementarity Theory

References

1. Cottle RW (1964) Nonlinear programs with positively bounded Jacobians. Univ. Calif., Berkeley, CA
2. Cottle RW (1966) Nonlinear programs with positively bounded Jacobians. SIAM J Appl Math 14:147–158
3. Cottle RW, Dantzig GB (1968) Complementary pivot theory of mathematical programming. Linear Alg & Its Appl 1:103–125
4. Cottle RW, Dantzig GB (1970) A generalization of the linear complementarity problem. J Combin Th 8:79–90

5. Cottle RW, Pang JS, Stone RE (1992) The linear complementarity problem. Acad. Press, New York
6. Cottle RW, Pang JS, Venkateswaran V (1989) Sufficient matrices and the linear complementarity problem. Linear Alg & Its Appl 114/115:231–249
7. Eaves BC (1971) The linear complementarity problem. Managem Sci 17:612–634
8. Eaves BC, Lemke CE (1981) Equivalence of LCP and PLS. Math Oper Res 6:475–484
9. Ebiefung AA (1995) Existence theory and Q-matrix characterization for generalized linear complementarity problem. Linear Alg & Its Appl 223/224:155–169
10. Ferris MC, Pang JS (1997) Engineering and economic applications of complementarity problems. SIAM Rev 39:669–713
11. Gowda MS (1995) On reducing a monotone horizontal LCP to an LCP. Appl Math Lett 8:97–100
12. Gowda MS (1996) On the extended linear complementarity problem. Math Program 72:33–50
13. Guu S-M, Cottle RW (1995) On a subclass of P0. Linear Alg & Its Appl 223/224:325–335
14. Habetler GJ, Price AJ (1971) Existence theory for generalized nonlinear complementarity problems. J Optim Th Appl 7:223–239
15. Harker PT, Pang JS (1990) Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications. Math Program B 48:161–220
16. Howson JT Jr (1963) Orthogonality in linear systems. Rensselaer Inst. Techn., Troy, NY
17. Isac G (1992) Complementarity problems. Lecture Notes Math, vol 1528. Springer, Berlin
18. Karamardian S (1971) Generalized complementarity problem. J Optim Th Appl 8:161–168
19. Kojima M, Megiddo N, Noma T, Yoshise A (1991) A unified approach to interior point algorithms for linear complementarity problems. Lecture Notes Computer Sci, vol 538. Springer, Berlin
20. Kuhn D, Löwen R (1987) Piecewise affine bijections of $\mathbf{R}^n$ and the equation $Sx^+ - Tx^- = y$. Linear Alg & Its Appl 96:109–129
21. Lemke CE (1965) Bimatrix equilibrium points and mathematical programming. Managem Sci 11:681–689
22. Lemke CE, Howson JT Jr (1964) Equilibrium points of bimatrix games. SIAM J Appl Math 12:413–423
23. Mangasarian OL, Pang JS (1995) The extended linear complementarity problem. SIAM J Matrix Anal Appl 16:359–368
24. Mohan SR, Neogy SK (1997) Vertical block hidden Z-matrices and the generalized linear complementarity problem. SIAM J Matrix Anal Appl 18:181–190
25. Murty KG (1968) On the number of solutions to the complementarity problem and spanning properties of complementary cones. Univ. Calif., Berkeley, CA
26. Murty KG (1972) On the number of solutions to the complementarity problem and spanning properties of complementary cones. Linear Alg & Its Appl 5:65–108
27. Murty KG (1988) Linear complementarity: linear and nonlinear programming. Heldermann, Berlin
28. Pang JS (1995) Complementarity problems. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 271–338
29. Robinson SM (1979) Generalized equations and their solutions, Part I: Basic theory. Math Program Stud 10:128–141
30. Samelson H, Thrall RM, Wesler O (1958) A partition theorem for Euclidean n-space. Proc Amer Math Soc 9:805–807
31. De Schutter B, De Moor B (1996) The extended linear complementarity problem. Math Program 71:289–326
32. Sznajder R, Gowda MS (1995) Generalizations of P_0- and P-properties; Extended vertical and horizontal LCPs. Linear Alg & Its Appl 223/224:695–715
33. Tütüncü RH, Todd MJ (1995) Reducing horizontal linear complementarity problems. Linear Alg & Its Appl 223/224:717–730
34. Väliaho H (1996) P_*-matrices are just sufficient. Linear Alg & Its Appl 239:103–108
35. Website: www.cs.wisc.edu/cpnet
36. Ye Y (1993) A fully polynomial-time approximation algorithm for computing a stationary point of the general linear complementarity problem. Math Oper Res 18:334–346
37. Yoshise A (1996) Complementarity problems. In: Terlaky T (ed) Interior point methods of mathematical programming. Kluwer, Dordrecht, pp 297–367
38. Zhang Y (1994) On the convergence of a class of infeasible interior-point algorithm for the horizontal linear complementarity problem. SIAM J Optim 4:208–227

Linear Optimization: Theorems of the Alternative

ThAlt

KEES ROOS
Department ITS/TWI/SSOR,
Delft University Technol.,
Delft, The Netherlands

MSC2000: 15A39, 90C05

Article Outline

Keywords
See also
References

Keywords

Inequality systems; Duality; Certificate; Transposition theorem

If one has two systems of *linear relations*, where each relation is either an *linear equation* (or *linear equality relation*) or a *linear inequality relation* (of type $>$, $\geq$, $<$, $\leq$ or $\neq$), and exactly one of the two systems has a solution, then one says that the two given systems are each others *alternative*. A mathematical theorem stating that two systems are alternative systems is called a *theorem of the alternative*, or also a *transposition theorem*. Many such theorems are known. The table lists ten results of this type, with their inventors and dates. The table is a modified version of tables of H. Greenberg [16] and in [8]. In each case the alternative systems are labelled by *a* and *b*, respectively.

Consider by way of example the two systems 4*a* and 4*b* in the table. The corresponding theorem of the alternative is known as *Farkas' lemma*. Assume that 4*a* has a solution x, so $Ax \leq b$. Then we have for each nonnegative vector y that $y^\top Ax \leq y^\top b$. Hence, if $y^\top A = 0$ then we will have $y^\top b \geq 0$. Thus it follows that if 4*a* has a solution then 4*b* does not have a solution. This is the easy part of the proof of Farkas' lemma. The proof of the other implication is much harder. For a discussion of several proof techniques, see ► Farkas lemma.

In the above example we used that for $y \geq 0$ the inequality $y^\top Ax \leq y^\top b$ is implied by the system $Ax \leq b$. Note that the *implied inequality* $y^\top Ax \leq y^\top b$ is obtained from the separate inequalities in $Ax \leq b$ by combining them in a linear fashion. Fixing y, one easily understands that the implied inequality has no solution x if and only if $y^\top A = 0$ and $y^\top b < 0$. Together with $y \geq 0$ these are precisely the relations in the alternative system 4*b*. Thus, it may be concluded that Farkas' lemma can be restated by saying that the system $Ax \leq b$ is feasible if and only if it does not imply (in a linear fashion) the 'contradiction' $0^\top x < 0$. The 'if'-part is obvious: if the system has an implied inequality $0^\top x < 0$ then it must be inconsistent. But the 'only if'-part is a very deep result: it states that if the system has no contradictory implied inequality then it has a solution. The other theorems of the alternative in the table admit a similar interpretation.

The relevance of a theorem of the alternative is the following. Given some system S of relations the crucial question is whether the system has a solution or not. Knowing the answer to this question one is able to answer many other questions. For example, if one has a *linear optimization problem LO* in the *standard form*

$$\min_x \left\{ c^\top x : \; Ax = b, \; x \geq 0 \right\},$$

a given real number $\underline{z}$ is a strict lower bound for the optimal value of the problem if and only if the system

$$Ax = b, \quad c^\top x \leq \underline{z}, \quad x \geq 0,$$

has no solution, i. e. is *infeasible*. On the other hand, a given real number $\overline{z}$ is an upper bound for the optimal

1	J.B.J. Fourier (1826) [4]
a	$Ax \leq 0, \; Bx < 0, \; Cx = 0$
b	$y^\top A + v^\top B + w^\top C = 0,$ $y \geq 0, \; 0 \neq v \geq 0$
2	P. Gordan (1873) [7]
a	$Ax > 0$
b	$y^\top A = 0, \; 0 \neq y \geq 0$
3	J.Farkas (1902) [3]
a	$Ax = b, \; x \geq 0$
b	$y^\top A \geq 0, \; y^\top b < 0$
4	Farkas (1902) [3]
a	$Ax \leq b$
b	$y \geq 0, \; y^\top A = 0, \; y^\top b < 0$
5	E. Stiemke (1915) [13]
a	$Ax = 0, \; x > 0$
b	$y^\top A \geq 0, \; y^\top A \neq 0$
6	W.B. Carver (1912) [2]
a	$Ax < b$
b	$y^\top A = 0, \; y \geq 0, \; y^\top b \leq 0, \; y \neq 0$
7	T.S. Motzkin (1936) [10]
a	$Ax \leq 0, \; Bx < 0$
b	$y^\top A + v^\top B = 0, \; y \geq 0, \; v \geq 0, \; v \neq 0$
8	J. Ville (1938) [15]
a	$Ax > 0, \; x > 0$
b	$y^\top A \leq 0, \; y \geq 0, \; y \neq 0,$ or $A^\top y \neq 0$
9	A.W. Tacket (1956) [14]
a	$Ax \geq 0, \; Ax \neq 0, \; Bx \geq 0, \; Cx = 0$
b	$y^\top A + v^\top B + w^\top C = 0, \; y > 0, \; v \geq 0$
10	D. Gale (1960) [5]
a	$Ax \leq b$
b	$y^\top A = 0, \; y^\top b = -1, y \geq 0$

Ten pairs of alternative systems

value of the problem if and only if the system

$$Ax = b, \quad c^\top x \geq z, \quad x \geq 0,$$

has a solution, i. e. is *feasible*.

If a system S has a solution then this is easy to certify, namely by giving a solution of the system. The solution then serves as a *certificate* for the feasibility of S. If S is infeasible, however, it is more difficult to give an easy certificate. One is then faced with the problem of how to certify a negative statement. This is in general a very nontrivial problem that also occurs in many real life situations. For example, when accused for murder, how should one prove his innocence? In circumstances like these it may be impossible to find an easy to verify certificate for the negative statement 'not guilty'. A practical solution is the rule 'a person is innocent until his/her guilt is certified'. Clearly, from the mathematical point of view this approach is unsatisfactory.

Now suppose that there is an alternative system T and there exists a theorem of the alternative for S and T. Then we know that exactly one of the two systems has a solution. Therefore, S has a solution if and only if T has no solution. In that case, any solution of T provides a certificate for the unsolvability of S. Thus it is clear that a theorem of the alternative provides an easy to verify certificate for the unsolvability of a system of linear relations.

The proof of any theorem of the alternative consists of two parts. Assuming the existence of a solution of one system one needs to show that the other system is infeasible, and vice versa. It has been demonstrated above for Farkas' lemma that one of the two implications is easy to prove. This seems to be true for each theorem of the alternative: in all cases one of the implications is almost trivial, but the other implication is highly nontrivial and very hard to prove. On the other hand, having proved one theorem of the alternative the other theorems of the alternative easily follow. In this sense one might say that all the listed theorems of the alternative are equivalent: accepting one of them to be true, the validity of each of the other theorems can be verified easily. The situation resembles a number of cities on a high plateau. Travel between them is not too difficult; the hard part is the initial ascent from the plains below [1].

It should be pointed out that Farkas' lemma, or each of the other theorems of the alternative, is equivalent to the most deep result in linear optimization, namely the duality theorem for linear optimization: this theorem can be easily derived from Farkas' lemma, and vice versa (cf. also ► Linear programming). In fact, in many textbooks on linear optimization the duality theorem is derived in this way [5,17], whereas in other textbooks the opposite occurs: the duality theorem is proved first and then Farkas' lemma follows as a corollary [11]. This phenomenon is a consequence of a simple, and basic, logical principle that any duality theorem is actually equivalent to a theorem of the alternative, as has been shown in [9].

Both the Farkas' lemma and the duality theorem for linear optimization can be derived from a more general result which states that for any *skew-symmetric matrix* K (i. e., $K = -K^\top$) there exists a vector x such that

$$Kx \geq 0, \quad x \geq 0, \quad x + Kx > 0.$$

This result is due to Tucker [14] who also derives Farkas' lemma from it, whereas A.J. Goldman and Tucker [6] show how this result implies the duality theorem for linear optimization. For recent proofs, see [12].

See also

- ► Farkas Lemma
- ► Linear Programming
- ► Motzkin Transposition Theorem
- ► Theorems of the Alternative and Optimization
- ► Tucker Homogeneous Systems of Linear Relations

References

1. Broyden CG (1998) A simple algebraic proof of Farkas' lemma and related theorems. Optim Methods Softw 3:185–199
2. Carver WB (1921) Systems of linear inequalities. A-MATH 23(2):212–220
3. Farkas J (1902) Theorie der Einfachen Ungleichungen. J Reine Angew Math 124:1–27
4. Fourier JBJ (1826) Solution d'une question particulière du calcul des inégalités. Nouveau Bull. Sci. Soc. Philomath. Paris, 99–100
5. Gale D (1960) The theory of linear economic models. McGraw-Hill
6. Goldman AJ, Tucker AW (1956) Theory of linear programming. In: Kuhn HW, Tucker AW (eds) Linear Inequalities and Related Systems. Ann Math Stud. Princeton Univ. Press, Princeton, 53–97

7. Gordan P (1873) Über die Auflösung Linearer Gleichungen mit Reelen Coefficienten. Math Ann 6:23–28
8. Mangasarian OL (1994) Nonlinear programming. No. 10 in Classics Appl Math. SIAM, Philadelphia
9. McLinden L (1975) Duality theorems and theorems of the alternative. Proc Amer Math Soc 53(1):172–175
10. Motzkin TS Beiträge zur Theorie der Linearen Ungleichungen, PhD Thesis, Baselxs
11. Padberg M (1995) Linear optimization and extensions. Algorithms and Combinatorics, vol 12. Springer, Berlin
12. Roos C, Terlaky T, Vial J-Ph (1997) Theory and algorithms for linear optimization. An interior approach. Wiley, New York
13. Stiemke E (1915) Über Positive Lösungen Homogener Linearer Gleichungen. Math Ann 76:340–342
14. Tucker AW (1956) Dual systems of homogeneous linear relations. In: Kuhn HW, Tucker AW (eds) Linear Inequalities and Related Systems. Ann Math Stud. Princeton Univ. Press, Princeton, 3–18
15. Ville J (1938) Sur la thèorie gènèrale des jeux où intervient l'habileté des joueurs. In: Ville J (ed) Applications aux Jeux de Hasard. Gauthier-Villars, Paris, pp 105–113
16. Website: www-math.cudenver.edu/~hgreenbe
17. Zoutendijk G (1976) Mathematical programming methods. North-Holland, Amsterdam

Linear Ordering Problem

LOP

PAOLA FESTA
Dip. Mat. e Inform., Universitá Salerno, Baronissi (SA), Italy

MSC2000: 90C10, 90C11, 90C20

Article Outline

Keywords
Problem Description
Review of Exact and Approximation Algorithms
  Branch and Bound Algorithms
  Linear Programming Algorithms
See also
References

Keywords

Combinatorial optimization; Greedy technique; Graph optimization; Branch and bound; Linear programming

The linear ordering problem (LOP) has a wide range of applications in several fields, such as scheduling, sports, social sciences, and economics. Due to its combinatorial nature, it has been shown to be *NP*-hard [5]. Like many other computationally hard problems, the linear ordering problem has captured the researcher attention for developing efficient solution procedures. A comprehensive treatment of the state-of-art approximation algorithms for solving the linear order problem is contained in [15]. The scope of this article is to introduce the reader to this problem, providing its definition and some of the algorithms proposed in literature for solving it efficiently.

Problem Description

The linear ordering problem (LOP) can be formulated as follows: Given a *complete digraph* $D_n = (V_n, E_n)$ on n nodes and given arc weights $c(i, j)$ for each arc $(i, j) \in E_n$, find a *spanning acyclic tournament* in D_n such that the sum of the weights of its arcs is as large as possible.

An equivalent mathematical formulation of LOP ([11]) is the following: Given a matrix of weights $E = \{e_{ij}\}_{m\times m}$, find a permutation p of the columns (and rows) in order to maximize the sum of the weights in the upper triangle. Formally, the problem is to maximize

$$C_E(p) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} e_{p_i p_j},$$

where p_i is the index of the column (and row) occupying the position i in the permutation.

The best known among the applications of LOP occurs in economics. In fact, it is equivalent to the so-called *triangulation problem for input-output tables*. In this economical application, the economy (regional or national) is subdivided into sectors. An $m \times m$ input-output matrix is then created, whose entry (i, j) represents the flow of money from the sector i to the sector j. The sectors have to be ordered so that suppliers tend to come first followed by costumers. This scope can be achieved by permuting the rows and the columns of the built matrix so that the sum of entries above the diagonal is maximized, which is exactly the objective of the linear ordering problem.

Review of Exact and Approximation Algorithms

The pioneer heuristic method for solving LOP has been proposed by H.B. Chenery and T. Watanabe [3]. Their method tries to obtain plausible rankings of the sectors of an input-output table in the triangulation problem by ranking first those sectors that have a small share of inputs from other sectors and of outputs to final demand. An extensive discussion about the heuristics proposed until 1981 can be found in [16], while more recent work has been done in [2,11]. In [11] a heuristic algorithm is proposed based on the *tabu search methodology* and incorporating strategies for search intensification and diversification are given. For search intensification M. Laguna and others experimented with *path relinking*, a strategy proposed in connection with tabu search by F. Glover and Laguna [6] and still rarely used in actual implementations. In [2] an algorithm is presented implementing a *scatter search* strategy, which is a population-based method that has been shown to lead to promising outcomes for solving combinatorial and nonlinear difficult problems.

The development of exact algorithms for LOP can be seen connected to the development of methods for solving general integer programming problems, since any such method can be slightly modified to solve the triangulation problem. Most of those exact algorithms belong either to the branch and bound family or to the linear programming methods.

Branch and Bound Algorithms

One of the earliest published computational results using a branch and bound strategy is due to J.S. DeCani in 1972 [4]. He originally studied how to rank n objects on the basis of a number of paired comparisons. Since k persons have to pairwise compare n objects according to some criterion, a matrix $E = \{e_{ij}\}$ is built, where e_{ij} is the number of persons that prefer object i to object j. The problem is to find a linear ranking of the objects reflecting the outcome of the experiment as closely as possible. In the branch and bound strategy proposed by DeCani partial rankings are built up and each branching operation in the tree corresponds to inserting a further object at some position in the partial ranking. At level n of the tree a complete ranking of the objects is found. The upper bounds are exploited in the usual way for backtracking and excluding parts of the tree from further consideration.

A further method for solving LOP is the *lexicographic search algorithm* proposed in [9,10]. It lexicographically enumerates all permutations of the n sectors by fixing at level k of the enumeration tree the kth position of the permutations. In more detail, if at level k a node is generated, then the first k positions $\sigma(1)$, …, $\sigma(k)$ are fixed. Based on this fixing several Helmstädter's conditions can be tested. If one of them is violated, then there is no relatively optimum having $\sigma(1)$, …, $\sigma(k)$ in the first k positions. Therefore, the node currently under consideration can be ignored and a backtracking is performed. By using this method all relatively optimum solutions are enumerated, since there is no bounding according to objective function values. At the end the best one among them is kept. Starting from lexicographic search, [8] proposed a lexicographic branch and bound scheme.

Other authors have proposed branch and bound methods, such as [7,12], and [14].

Linear Programming Algorithms

All linear programming approaches are based on the consideration that the triangulation problem can be formulated as a 0–1 integer programming problem using the 3-dicycle inequalities. In [13] the LP relaxation using the tournament polytope P_C^n is proposed and the corresponding full linear program is solved in its dual version. In [1] LP relaxation is used for solving scheduling problems with precedence constraints. It is easy to see that the scheduling problem of minimizing the total weighted completion time of a set of processes on a single processor can be formulated as a linear ordering problem.

Other possibilities for theoretically solving linear ordering problems are methods as *dynamic programming* or by formulating the problem as *quadratic assignment problem* ([10]).

See also

- ► Assignment and Matching
- ► Assignment Methods in Clustering
- ► Bi-objective Assignment Problem
- ► Communication Network Assignment Problem
- ► Complexity Theory: Quadratic Programming

- ▶ Feedback Set Problems
- ▶ Frequency Assignment Problem
- ▶ Generalized Assignment Problem
- ▶ Graph Coloring
- ▶ Graph Planarization
- ▶ Greedy Randomized Adaptive Search Procedures
- ▶ Maximum Partition Matching
- ▶ Quadratic Assignment Problem
- ▶ Quadratic Fractional Programming: Dinkelbach Method
- ▶ Quadratic Knapsack
- ▶ Quadratic Programming with Bound Constraints
- ▶ Quadratic Programming Over an Ellipsoid
- ▶ Quadratic Semi-assignment Problem
- ▶ Standard Quadratic Optimization Problems: Algorithms
- ▶ Standard Quadratic Optimization Problems: Applications
- ▶ Standard Quadratic Optimization Problems: Theory

References

1. Boenchendorf K (1982) Reihenfolgenprobleme/Mean-flow-time sequencing. Math Systems in Economics. Athenäum–Hain–Scriptor–Hanstein, Königstein/Ts.
2. Campos V, Glover F, Laguna M, Martí R (1999) An experimental evaluation of a scatter search for the linear ordering problem. Manuscript Apr
3. Chenery HB, Watanabe T (1958) International comparisons of the structure of production. Econometrica 26(4):487–521
4. DeCani JS (1972) A branch &bound algorithm for maximum likelihood paired comparison ranking. Biometrika 59:131–135
5. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, New York
6. Glover F, Laguna M (1997) Tabu search. Kluwer, Dordrecht
7. Hellmich K (1970) Ökonomische Triangulierung. Heft 54. Rechenzentrum Graz, Graz
8. Kaas R (1981) A branch&bound algorithm for the acyclic subgraph problem. Europ J Oper Res 8:355–362
9. Korte B, Oberhofer W (1968) Zwei Algorithmen zur Lösung eines Komplexen Reihenfolgeproblems. Unternehmensforschung 12:217–231
10. Korte B, Oberhofer W (1969) Zur Triangulation von Input-Output Matrizen. Jahrbuch f Nat Ok u Stat 182:398–433
11. Laguna M, Martí R, Campos V (1999) Intensification and diversification with elite tabu search solutions for the liner ordering problem. Comput Oper Res 26:1217–1230
12. Lenstra jr. HW (1973) The acyclic subgraph problem. Techn Report Math Centrum Amsterdam BW26
13. Marcotorchino JF, Mirchaud P (1979) Optimisation en analyse ordinale des donnees. Masson, Paris
14. Poetsch G (1973) Lösungsverfahren zur Triangulation von Input-Output Tabellen. Heft 79. Rechenzentrum Graz, Graz
15. Reinelt G (1985) The linear ordering problem: Algorithms and applications. In: Hofmann HH, Wille R (eds) Res. and Exposition in Math., vol 8. Heldermann, Berlin
16. Wessels H (1981) Computers and intractability: A guide to the theory of NP-completeness. Beiträge zur Strukturforschung, vol 63. Deutsches Inst. Wirtschaftsforschung, Berlin
17. Whitney H (1935) On the abstract properties of linear dependence. Amer J Math 57:509–533

Linear Programming

LP

PANOS M. PARDALOS
Center for Applied Optim., Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 90C05

Article Outline

Keywords
Problem Description
  The Simplex Method
See also
References

Keywords

Linear programming; Basic solution; Simplex method; Pivoting; Nondegenerate

Linear programming (LP) is a fundamental optimization problem in which a linear objective function is to be optimized subject to a set of linear constraints. Due to the wide applicability of linear programming models, an immense amount of work has appeared regarding theory and algorithms for LP, since G.B. Dantzig proposed the simplex algorithm in 1947. It is not surprizing that in a recent survey of Fortune 500 companies, 85% of those responding said that they had used linear programming. The history, theory, and applications of linear programming may be found in [3]. Several books

have been published on the subject (see the references section).

Problem Description

Consider the linear programming problem (in standard form):

$$\begin{cases} \min & c^\top x \\ \text{s.t.} & Ax = b, \\ & x \geq 0 \end{cases} \tag{1}$$

where $c \in \mathbf{R}^n$, $b \in P^m$ and A is an $m \times n$ matrix of rank m (i. e. we do not have any redundant constraints). The feasible domain

$$P = \{x \in \mathbb{R}^n : \ Ax = b, \ x \geq 0\}$$

is a polytope. We assume that (1) has a finite optimal solution. Let B be a submatrix of A formed by m linearly independent columns. We may assume that $A = [B, N]$, i. e. the first columns of A are linearly independent. Then the linear system $Bx_B = b$ has a unique solution. If $x = (x_B, 0)$ then $Ax = b$ and $x = (x_B, 0)$ is called a *basic solution*. The components of x associated with the columns of B are called *basic variables*. If one of the basic variables in a basic solution is zero, that solution is called a *degenerate basic solution*. A basic solution that is feasible (i. e. $x \geq 0$) is called a *basic feasible solution*.

The following theorem identifies the special importance of the basic feasible solutions.

Theorem 1 *Assume that P in (1) is nonempty. Then a feasible point $x \in P$ is a vertex of P if and only if x is a basic feasible solution.*

Existence of basic feasible solutions is established by the following fundamental theorem of linear programming.

Theorem 2 *Given the linear programming problem (1), the following statements are true:*

1) *If P is nonempty, there exists a basic feasible solution.*
2) *If (1) has an optimal solution, then there is an optimal basic feasible solution.*

Therefore, the linear programming problem can be solved by searching among its basic feasible solutions (i. e. vertices of P). Since there are at most

$$\binom{m}{n}$$

basic solutions, the above theorem gives a finite, but a very inefficient algorithm. A more systematic search among the basic feasible solutions, is given by the simplex method, which was developed by Dantzing in 1947.

The Simplex Method

The simplex method has a simple geometric motivation which is described by the following two phases.

I	An initial vertex x_0 of P (basic feasible solution) is computed.
II	Starting from the vertex x_0, a sequence of vertices $x_0, \ldots, x_N$ is computed such a way that x_{i+1} is adjacent to x_i, $i = 0, \ldots, N-1$, and such that $c^\top x_{i+1} < c^\top x_i$. The method terminates if either none of the edges adjacent to x_N is decreasing the objective function (i.e., x_N is the solution) or if an unbounded edge adjacent to x_N is found, improving the objective function (i.e. the problem is unbounded).

Each step of the simplex method, moving from one vertex to an adjacent one, is called *pivoting*. The integer N gives the number of pivot steps in the simplex method. Phase I can be solved in a similar way to Phase II. In problems of the canonical form:

$$\begin{cases} \min & c^\top x \\ \text{s.t.} & Ax = b, \\ & x \geq 0, \quad b \geq 0, \end{cases} \tag{2}$$

there is no need for Phase I, because an initial vertex ($x_0 = 0$) is at hand. We start by considering Phase II of the simplex method, by assuming that an initial vertex (basic feasible solution) is available. Let x_0 be a basic feasible solution with $x_{10}, \ldots, x_{m0}$ its basic variables, and let $B = \{A_{B(i)} : i = 1, \ldots, m\}$ the corresponding *basis*. If A_j denotes the jth column of A, ($A_j \notin B$), then

$$\sum_{i=1}^{m} x_{ij} A_{B(i)} = A_j. \tag{3}$$

In addition,

$$\sum_{i=1}^{m} x_{i0} A_{B(i)} = b. \tag{4}$$

Multiply (3) by $\theta > 0$ and subtract the result from (4) to obtain:

$$\sum_{i=1}^{m}(x_{i0} - \theta x_{ij})A_{B(i)} + \theta A_j = b. \tag{5}$$

Assume that x_0 is nondegenerate. How much can we increase θ and still have a solution? We can increase θ until the first component of $(x_{i0} - \theta x_{ij})$ becomes zero or equivalently

$$\theta_0 = \min_i \left\{ \frac{x_{i0}}{x_{ij}} :\ x_{ij} > 0 \right\}. \tag{6}$$

If $\theta_0 = x_{l0}/x_{lj}$, then column A_l leaves the basis and A_j enters the basis.

If a tie occurs in (6), then the new solution is degenerate. In addition, if all $x_{ij} \le 0$, then we move arbitrarily far without becoming infeasible. In that case the problem is unbounded.

Define the new point $x_0{}'$ by

$$x'_{i0} = \begin{cases} x_{i0} - \theta x_{ij}, & i \neq l, \\ \theta_0, & i = l, \end{cases} \tag{7}$$

and

$$B'(i) = \begin{cases} B(i), & i \neq l, \\ j, & i = l. \end{cases}$$

It is easy to see that the m columns $A_{B'(i)}$ are linearly independent. Let

$$\sum_{i=1}^{m} x_i A_{B'(i)} = x_l A_j + \sum_{\substack{i=1 \\ i \neq l}}^{m} x_i A_{B(i)=0}.$$

Using (3) we have:

$$\sum_{\substack{i=1 \\ i \neq l}}^{m}(a_i x_{ij} + a_i)A_{B(i)} + a_l x_{lj} A_{B(l)} = 0$$

and by linear independence of the columns $A_{B(i)}$ we have

$$a_l = 0, \quad a_i(l + x_{ij}) = 0 \to a_1, \ldots, a_m = 0.$$

Hence, the new point $x_0{}'$ whose basic variables are given by (7) is a new basic feasible solution. When the basic feasible solution x_0 is degenerate then some of the basic variables are zero. Therefore more than $n-m$ of the constraints $x_j \ge 0$ are satisfied as equations (are active) and so x_0 satisfies more than n equations. From (6) it follows that if $x_{i0} = 0$ and the corresponding $x_{ij} > 0$, then $\theta_0 = 0$ and therefore we remain at the same vertex.

Note that when a basic feasible solution x_0 is degenerate, there can be an enormous number of basis associated with it. In fact, if x_0 has $k > m$ positive components, then there may be as many as $\binom{n-k}{n-m}$ different bases. In that case we may compute x_0 as many times as there are basis, but the set of variables that we label basic and nonbasic are different.

The cost (value of objective function) as a basic feasible solution x, with corresponding basis B is:

$$z_0 = \sum_{l=1}^{n} x_{i_0} c_{B(i)}$$

Suppose we bring column A_j into the new basis. The following economic interpretation can be used to select the pivot column A_j: For every unit of the variable x_j that enters the basis, an amount x_{ij} of each of the variables $x_{B(i)}$ must leave. Hence, a unit increase in the variable x_j results in a net change in the cost, equal to:

$$\bar{c}_j = c_j - z_j$$

(relative cost of column j), where $z_j = \sum_{i=1}^{m} x_{ij} c_{B(i)}$. It is profitable to bring column j into the basis exactly when $\bar{c}_j < 0$. Choosing the most negative $\bar{c}_j$ corresponds to a kind of steepest descent. However, many other selection criteria can be used (e. g., Blad's rule, etc).

If all reduced costs satisfy $\bar{c}_j \ge 0$, then we are at an optimal solution and the simplex method terminates. Note that relations (1) can be expressed in matrix notation by:

$$BX = A \quad \text{or} \quad X = B^{-1}A,$$

that is, the matrix $X = (x_{ij})$ is obtained by diagonalizing the basic columns of A. Then

$$z_j = \sum_{l=1}^{m} x_{ij} c_{B(i)} \quad \text{or} \quad z^\top = c_B^\top X = c_B^\top B^{-1} A.$$

Suppose $\bar{c} = c - z \ge 0$. Let y be a feasible point. Then,

$$c^\top y \ge z^\top y \ge c_B^\top B^{-1} A y = c_B^\top B^{-1} b = c^\top x_0$$

and therefore x_0 is an optimal solution.

Under the assumption of nondegeneracy with our pivot selection, $x_{l0} > 0$ (see (6)) and

$$\bar{z}_0 = z_0 - \frac{x_{l0}}{x_{lj}}(z_j - c_j) > z_0 \quad (z_j - c_j < 0).$$

Note that corresponding to any basis there is a unique z_0, and hence, we can never return to a previous basis. Therefore, each iteration gives a different basis and the simplex method terminator after $N \leq \binom{n}{m}$ pivots.

See also

- Affine Sets and Functions
- Carathéodory Theorem
- Convex-simplex Algorithm
- Criss-cross Pivoting Rules
- Farkas Lemma
- Gauss, Carl Friedrich
- Global Optimization in Multiplicative Programming
- History of Optimization
- Kantorovich, Leonid Vitalyevich
- Krein–Milman Theorem
- Least-index Anticycling Rules
- Lemke Method
- Lexicographic Pivoting Rules
- Linear Complementarity Problem
- Linear Optimization: Theorems of the Alternative
- Linear Space
- Motzkin Transposition Theorem
- Multiparametric Linear Programming
- Multiplicative Programming
- Parametric Linear Programming: Cost Simplex Algorithm
- Pivoting Algorithms for Linear Programming Generating Two Paths
- Principal Pivoting Methods for Linear Complementarity Problems
- Probabilistic Analysis of Simplex Algorithms
- Sequential Simplex Method
- Simplicial Pivoting Algorithms for Integer Programming
- Tucker Homogeneous Systems of Linear Relations

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms and applications. Prentice-Hall, Englewood Cliffs, NJ
2. Bertsimas D, Tsistklis JN (1997) Introduction to linear optimization. Athena Sci., Belmont, MA
3. Dantzig GB (1963) Linear programming and extensions. Princeton Univ. Press, Princeton
4. Fang S-C, Puthenpura S (1993) Linear optimization and extensions. Prentice-Hall, Englewood Cliffs, NJ
5. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: Algorithms and complexity. Prentice-Hall, Englewood Cliffs, NJ
6. Roos C, Terlaky T, Vial J-Ph (1998) Theory and algorithms for linear optimization: An interior point approach. Wiley, New York

Linear Programming: Interior Point Methods

KURT M. ANSTREICHER
University Iowa, Iowa City, USA

MSC2000: 90C05

Article Outline

Keywords
See also
References

Keywords

Linear programming; Interior point methods; Polynomial time algorithm

An enormous amount of research on interior point algorithms for linear programming (LP) has been conducted since N.K. Karmarkar [8] announced his celebrated *projective algorithm* in 1984. Interior point algorithms for LP are interesting for two different reasons. First, many interior point methods are *polynomial time algorithms* for LP. Consider a standard form problem:

$$\text{LP} \quad \begin{cases} \min & c^\top x \\ \text{s.t.} & Ax = b \\ & x \geq 0, \end{cases}$$

where A is an $m \times n$ matrix. For the purpose of characterizing the complexity of algorithms it is common to assume that the data of LP is integral. If L is the number of bits required to encode the data, then an algorithm for LP is polynomial time if the number of

operations required to solve LP is a polynomial function of n, m, and L. Throughout we use 'operations' to refer to arithmetic operations in infinite precision, although for an algorithm to be rigorously polynomial time in the rational model of computation the number of bits required to perform all computations should also be polynomially bounded. Karmarkar's projective algorithm solves LP in $O(nL)$ iterations and $O(n^4L)$ total operations. This overall complexity is required to obtain a solution of LP whose objective is within a tolerance $2^{-O(L)}$ of optimality; an exact optimal solution can then be obtained using a 'rounding' procedure in $O(n^3)$ operations. Karmarkar also described a *partial updating* procedure that reduced the overall complexity of his algorithm to $O(n^{3.5}L)$ operations. The idea of partial updating is to allow for some error in the specification of the projection equations that are solved on each iteration of the algorithm.

Interior point algorithms are also interesting because they perform well in practice. When the projective algorithm was first announced Karmarkar made well-publicized claims that his algorithm was several times faster than the simplex method in solving large LP problems. It was eventually discovered that most of Karmarkar's claims were actually for an implementation of the *affine scaling algorithm*, a simplified version of Karmarkar's algorithm that avoids the use of projective transformations. Initial attempts to replicate Karmarkar's results were mainly failures, but eventually it was convincingly established that interior point algorithms are highly competitive with the simplex method on large scale problems.

The announcement of Karmarkar's algorithm led to the development of a variety of different types of interior point methods for LP. The simplest of these are affine scaling methods, which were independently devised by E. Barnes [2] and R.J. Vanderbei, M.J. Meketon, and B.A. Freedman [21]. It was eventually realized that in fact the affine scaling method was discovered by I.I. Dikin [3] in 1967. The affine scaling method is not a polynomial time algorithm for LP, and it is now known that the algorithm may not even converge if the stepsize is too long [12]. Nevertheless its practical performance is often quite good, as indicated by Karmarkar's early claims.

Another type of interior point method, the *path following algorithm*, was discovered by J. Renegar [17]. Renegar's algorithm requires only $O(\sqrt{n}L)$ iterations to solve LP, as opposed to $O(nL)$ iterations for Karmarkar's algorithm. By adapting Karmarkar's partial updating technique to the path following framework, C.C. Gonzaga [5] and P.M. Vaidya [19] devised the first algorithms for LP with overall complexities of $O(n^3L)$ operations. The iterates of path following algorithms lie in a small neighborhood of the *central path* or *central trajectory*, which is defined to be the set of minimizers of the *logarithmic barrier function*

$$f_\mu(x) = \frac{c^\top x}{\mu} - \sum_{i=1}^{n} \ln(x_i),$$

over $\{x: Ax = b, x > 0\}$, for $\mu \in (0, \infty)$. Later C. Roos and J.-Ph. Vial [18], and Gonzaga [6] developed 'long step' path following algorithms. These algorithms are based on properties of the central path, but the iterates are not constrained to remain in a small neighborhood of the path. Long step path following algorithms are very closely related to the classical sequential unconstrained minimization technique (SUMT) of A.V. Fiacco and G.P. McCormick [4].

A different class of interior point algorithms is based on Karmarkar's use of a *potential function*, a surrogate for the original objective, to monitor the progress of his projective algorithm. Gonzaga [7] and Y. Ye [23] devised the first *potential reduction algorithms*. These algorithms are based on reducing a potential function but do not employ projective transformations. Ye's potential reduction algorithm requires $O(\sqrt{n}L)$ iterations, like path following algorithms, and provides an $O(n^3L)$ algorithm for LP when implemented with partial updating.

All of the algorithms mentioned to this point are based on solving LP, or alternatively the dual problem:

$$\text{LD} \quad \begin{cases} \max & b^\top y \\ \text{s.t.} & A^\top y + s = c \\ & s \geq 0. \end{cases}$$

Algorithms for solving LP typically generate feasible solutions to LD, and vice versa, but the algorithms are not symmetric in their treatment of the two problems. A different class of interior point methods, known as *primal-dual algorithms*, is completely symmetric in the

variables x and s. Primal-dual algorithms are based on applying Newton's method directly to the system of equations:

$$\mathrm{PD}(\mu) \quad \begin{cases} Ax = b \\ A^\top y + s = c \\ x \circ s = \mu e, \end{cases}$$

where $e \in \mathbf{R}^n$ is the vector of ones, μ is a positive scalar, and $x \circ s$ is the vector whose ith component is $x_i s_i$. Solutions $x > 0$ and $s > 0$ to PD(μ) are exactly on the central paths for LP and LD, respectively. Most primal-dual algorithms fit into the path following framework. The idea of a primal-dual path following algorithm was first suggested by N. Megiddo [13], and complete algorithms were first devised by R.C. Monteiro and I. Adler [15] and M. Kojima, S. Mizuno, and Y. Yoshise [10]. It is widely believed that primal-dual methods are in practice the best performing interior point algorithms for LP.

One advantage of the system PD(μ) is that Newton's method can be applied even when the current $x > 0$ and $s > 0$ are not feasible in LP and LD. This *infeasible interior point* (IIP) strategy was first employed in the OB1 code of I.J. Lustig, M.E. Marsten, and D.F. Shanno [11]. The solution to the Newton equations with $\mu = 0$ is referred to as the predictor, or primal-dual affine scaling direction, while the solution with $\mu = x^\top s/n$, for the current solutions x and s, is called the *corrector*, or *centering direction*. The primal-dual *predictor-corrector algorithm* alternates between the use of these two directions. One implementation of the IIP predictor-corrector strategy, due to S. Mehrotra [14], has worked particularly well in practice. Despite the fact that primal-dual IIP algorithms were very successfully implemented it proved to be quite difficult to characterize the convergence of these methods. The first such analyses, by Kojima, Megiddo, and Mizuno [9], and Y. Zhang [25], were followed by a large number of papers giving convergence/complexity results for various IIP algorithms. Ye, M.J. Todd, and Mizuno [24] devised a 'selfdual homogeneous' interior point method that has many of the practical features of IIP methods but at the same time has stronger convergence properties. An implementation of the homogeneous algorithm [22] exhibits excellent behavior, particularly when applied to infeasible or near-infeasible problems.

Many interior point algorithms for LP can be extended to more general optimization problems. Primal-dual algorithms generalize very naturally to the monotone linear complementarity problem (LCP; cf. ► linear complementarity problem); in fact many papers on primal-dual algorithms (for example [25]) are written in terms of the LCP. As a result these algorithms immediately provide interior point solution methods for convex quadratic programming (QP) problems. Interior point algorithms can also be generalized to apply to quadratically constrained quadratic programming (QCQP), optimization over *second order cone* (SOC) constraints, and semidefinite programming (SDP); for details on these and other extensions see [16]. The application of interior point methods to SDP has particularly rich applications, as described in [1], and [20], and remains the topic of extensive research.

See also

- ► Entropy Optimization: Interior Point Methods
- ► Homogeneous Selfdual Methods for Linear Programming
- ► Interior Point Methods for Semidefinite Programming
- ► Linear Programming: Karmarkar Projective Algorithm
- ► Potential Reduction Methods for Linear Programming
- ► Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ► Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Alizadeh F (1995) Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM J Optim 5:13–51
2. Barnes ER (1986) A variation on Karmarkar's algorithm for solving linear programming problems. Math Program 36:174–182
3. Dikin II (1967) Iterative solution of problems of linear and quadratic programming. Soviet Math Dokl 8:674–675
4. Fiacco AV, McCormick GP (1990) Nonlinear programming, sequential unconstrained minimization techniques. SIAM, Philadelphia
5. Gonzaga CC (1989) An algorithm for solving linear programming problems in O(n3L) operations. In: Megiddo N

(ed) Progress in Mathematical Programming. Springer, Berlin, pp 1–28
6. Gonzaga CC (1991) Polynomial affine algorithms for linear programming. Math Program 49:7–21
7. Gonzaga CC (1991) Large-step path-following methods for linear programming, Part I: Barrier function method. SIAM J Optim 1:268–279
8. Karmarkar N (1984) A new polynomial-time algorithm for linear programming. Combinatorica 4:373–395
9. Kojima M, Megiddo N, Mizuno S (1993) A primal-dual infeasible-interior-point algorithm for linear programming. Math Program 61:263–280
10. Kojima M, Mizuno S, Yoshise A (1989) A primal-dual interior point algorithm for linear programming. In: Megiddo N (ed) Progress in Mathematical Programming. Springer, Berlin, 29–47
11. Lustig IJ, Marsten RE, Shanno DF (1991) Computational experience with a primal-dual interior point method for linear programming. Linear Alg Appl 152:191–222
12. Mascarenhas WF (1997) The affine scaling algorithm fails for stepsize 0.999. SIAM J Optim 7:34–46
13. Megiddo N (1989) Pathways to the optimal set in linear programming. In: Megiddo N (ed) Progress in Mathematical Programming. Springer, Berlin, pp 131–158
14. Mehrotra S (1992) On the implementation of a primal-dual interior point method. SIAM J Optim 2:575–601
15. Monteiro RC, Adler I (1989) Interior path following primal-dual algorithms. Part I: linear programming. Math Program 44:27–41
16. Nesterov Y, Nemirovskii A (1994) Interior-point polynomial algorithms in convex programming. SIAM, Philadelphia
17. Renegar J (1988) A polynomial-time algorithm, based on Newton's method, for linear programming. Math Program 40:59–93
18. Roos C, Vial J-Ph (1990) Long steps with the logarithmic penalty barrier function in linear programming. In: Gabszewicz J, Richard J-F, Wolsey L (eds) Economic Decision Making: Games, Economics, and Optimization. Elsevier, Amsterdam, pp 433–441
19. Vaidya PM (1990) An algorithm for linear programming which requires O(((m+n)n2 + (m+n)1.5n)L) arithmetic operations. Math Program 47:175–201
20. Vandenberghe L, Boyd S (1996) Semidefinite programming. SIAM Rev 38:49–95
21. Vanderbei RJ, Meketon MJ, Freedman BA (1986) A modification of Karmarkar's linear programming algorithm. Algorithmica 1:395–407
22. Xu X, Hung P-F, Ye Y (1996) A simplified homogeneous self-dual linear programming algorithm and its implementation. Ann Oper Res 62:151–171
23. Ye Y (1991) An O(n3L) potential reduction algorithm for linear programming. Math Program 50:239–258
24. Ye Y, Todd MJ, Mizuno S (1994) An $O(nL)O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm. Math Oper Res 19:53–67
25. Zhang Y (1994) On the convergence of a class of infeasible interior-point algorithms for the horizontal linear complementarity problem. SIAM J Optim 4:208–227

Linear Programming: Karmarkar Projective Algorithm

Karmarkar Algorithm

KURT M. ANSTREICHER
University Iowa, Iowa City, USA

MSC2000: 90C05

Article Outline

Keywords
See also
References

Keywords

Linear programming; Interior point methods; Projective transformation; Potential function; Polynomial time algorithm

In his groundbreaking paper [6], N.K. Karmarkar described a new *interior point method* for linear programming (LP). As originally described by Karmarkar, his algorithm applies to a LP problem of the form:

$$\text{KLP} \quad \begin{cases} \min & c^\top x \\ \text{s.t.} & Ax = 0 \\ & x \in S, \end{cases}$$

where $x \in \mathbf{R}^n$, A is an $m \times n$ matrix, and S is the simplex $S = \{x \in \mathbf{R}^n : x \geq 0,\ e^\top x = n\}$. Throughout e denotes the vector with each component equal to one. It is assumed that e is feasible in KLP, and that the optimal objective value in KLP is exactly zero. These assumptions may seem restrictive, but it is easy to show that a standard form LP problem:

$$\begin{cases} \min & c^\top x \\ \text{s.t.} & Ax = b \\ & x \geq 0, \end{cases} \tag{1}$$

can be converted into a problem of the form KLP by combining the problem with its dual, and minimizing the gap between the two problems.

In addition to the special form of the LP problem, Karmarkar employed two novel ingredients in the specification of his algorithm. The first was the use of a *projective transformation* in the construction of the algorithm's iterative process. The algorithm is initialized at $x^0 = e$. For an iterate $x^k > 0$, $k \geq 0$, let X^k be the diagonal matrix with $X^k_{ii} = x^k_i$, $i = 1, \ldots, n$. On the kth iteration, the algorithm uses a projective change of coordinates $T^k: S \to S$,

$$T_k(x) = \frac{n(X^k)^{-1}x}{e^\top (X^k)^{-1}x},$$

to map the point x^k to e. Under the assumption that the optimal value in KLP is zero, KLP is equivalent to the transformed problem:

$$\begin{cases} \min & \overline{c}^\top \overline{x} \\ \text{s.t.} & \overline{A}\overline{x} = 0 \\ & \overline{x} \in S, \end{cases}$$

where $\overline{x} = T^k(x)$, $\overline{c} = X^k c$ and $\overline{A} = AX^k$. The algorithm then takes a projected gradient step in the transformed problem, and uses the inverse projective transformation to define the next iterate in the original coordinates:

$$x^{k+1} = T_k^{-1}\left(e - \alpha \frac{\overline{c}_p}{\|\overline{c}_p\|}\right), \tag{2}$$

where α is a positive steplength and $\overline{c}_p$ is the projection of $\overline{c}$ onto the nullspace of $\overline{A}$ and $e^\top$.

Karmarkar's second innovation was the use of a *potential function* to monitor the algorithm's progress. Karmarkar's potential function is:

$$f(x) = n \ln(c^\top x) - \sum_{i=1}^{n} \ln(x_i).$$

Karmarkar proved that on each iteration, the steplength α in (2) can be chosen so that $f(\cdot)$ is reduced by an absolute constant δ. It is then easy to show that the iterates satisfy $c^\intercal x^k \leq e^{-k\delta/n} c^\intercal x^0$ for all k. For any positive L, it follows that if $c^\intercal x^0 \leq 2^{O(L)}$, then the algorithm obtains an iterate x^k having $c^\intercal x^k \leq 2^{-O(L)}$ in $k = O(nL)$ iterations, each requiring $O(n^3)$ operations. For a problem of the form KLP with integer data, it can be shown that if $c^\intercal x^k \leq 2^{-O(L)}$, where L is the number of bits required to represent the problem, then an exact optimal solution can be obtained from x^k via a 'rounding' procedure. These facts together imply that Karmarkar's algorithm is a *polynomial time algorithm* for linear programming, requiring $O(n^4L)$ operations for a problem with n variables, and integer data of bit size L. Karmarkar also described a *partial updating* technique that reduces the total complexity of his algorithm to $O(n^{3.5}L)$ operations. Partial updating is based on using a scaling matrix $\widetilde{X}^k$ which is an approximation of X^k, and only 'updating' components $\widetilde{X}^k_{ii}$ which differ from X^k_{ii} by more than a fixed factor.

Karmarkar's algorithm created a great deal of interest for two reasons. First, the algorithm was a polynomial time method for LP. Second, Karmarkar claimed that unlike the *ellipsoid algorithm*, the other well-known polynomial time method for LP, his method performed extremely well in practice. There was some controversy at the time regarding these claims, and eventually it was discovered that most of Karmarkar's computational results were based on the *affine scaling algorithm*, a simplified version of his algorithm that avoids the use of projective transformations. In any case it soon became clear that the performance of interior point algorithms for LP could be highly competitive with the *simplex method*, the usual solution technique, on large problems.

There is a great deal of research connected with Karmarkar's algorithm. Several authors ([1,3,4,5,9]) showed that the special form of KLP was unnecessary, and instead the projective algorithm could be directly applied to a standard form problem (1). This 'standard form variant' adds logic which maintains a lower bound on the unknown optimal value in (1). Later it was shown that the projective transformations could also be eliminated, giving rise to so-called *potential reduction algorithms* for LP. The best known potential reduction algorithm, due to Y. Ye [8], requires only $O(\sqrt{n}L)$ iterations, and with an adaptation of Karmarkar's partial updating technique has a total complexity of $O(n^3L)$ operations. The survey articles [2] and [7] give extensive references to research connected with Karmarkar's algorithm, and related potential reduction methods.

See also

- ▶ Entropy Optimization: Interior Point Methods
- ▶ Homogeneous Selfdual Methods for Linear Programming
- ▶ Interior Point Methods for Semidefinite Programming
- ▶ Linear Programming: Interior Point Methods
- ▶ Potential Reduction Methods for Linear Programming
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Anstreicher KM (1986) A monotonic projective algorithm for fractional linear programming. Algorithmica 1:483–498
2. Anstreicher KM (1996) Potential reduction algorithms. In: Terlaky T (ed) Interior point methods of mathematical programming. Kluwer, Dordrecht, pp 125–158
3. de Ghellinck G, Vial J-Ph (1986) A polynomial Newton method for linear programming. Algorithmica 1:425–453
4. Gay DM (1987) A variant of Karmarkar's linear programming algorithm for problems in standard form. Math Program 37:81–90
5. Gonzaga CC (1989) Conical projection algorithms for linear programming. Math Program 43:151–173
6. Karmarkar N (1984) A new polynomial-time algorithm for linear programming. Combinatorica 4:373–395
7. Todd MJ (1997) Potential-reduction methods in mathematical programming. Math Program 76:3–45
8. Ye Y (1991) An O(n3L) potential reduction algorithm for linear programming. Math Program 50:239–258
9. Ye Y, Kojima M (1987) Recovering optimal dual solutions in Karmarkar's polynomial algorithm for linear programming. Math Program 39:305–317

Linear Programming: Klee–Minty Examples

Konstantinos Paparrizos[1], Nikolaos Samaras[1], Dimitrios Zissopoulos[2]

[1] Department Applied Informatics, University Macedonia, Thessaloniki, Greece

[2] Department Business Admin., Techn. Institute West Macedonia, Kozani, Greece

MSC2000: 90C05

Article Outline

Keywords
Introduction
Simplex Algorithm
Klee–Minty Examples
Applications
  Smallest Index Rule
  Largest Coefficient Rule
See also
References

Keywords

Klee–Minty examples; Linear programming; Simplex algorithm; Pivoting rules

The problem of determining the worst-case behavior of the simplex algorithm remained an outstanding open problem for more than two decades. In the beginning of the 1970s, V. Klee and G.J. Minty [9] solved this problem by constructing linear examples on which an exponential number of iterations is required before optimality occurs. In this article we present the Klee–Minty examples and show how they can be used to show exponential worst-case behavior for some well known *pivoting rules*.

Introduction

The problem of determining the worst-case behavior of the simplex algorithm remained an outstanding open problem for more than two decades. In the beginning of the 1970s, Klee and Minty in their classical paper [9] showed that the most commonly used pivoting rule, i. e., *Dantzig's largest coefficient pivoting rule* [5], performs exponentially bad on some specially constructed linear problems, known today as Klee–Minty examples. Later on, R.G. Jeroslow [8] showed similar behavior for the maximum improvement pivoting rule. He showed this result by slightly modifying Klee–Minty examples.

The Klee–Minty examples have been used by several researchers to show exponential worst-case behavior for the great majority of the practical pivoting rules. D. Avis and V. Shvatal [1] and independently, K.G. Murty [10, p. 439] showed exponential behavior for *Bland's least index pivoting rule* [2] and D. Goldfarb and W. Sit [7] for the *steepest edge simplex method* [5]. Recently,

C. Roos [13] established exponential behavior for *Terlaky's criss-cross method* [14] and K. Paparrizos [11] for a number of pivoting rules some of which use past history. Similar results have been derived by Paparrizos [12] for his *dual exterior point algorithm* and K. Dosios and Paparrizos [6] for a new primal dual pivoting rule [3].

In this paper we present the Klee–Minty examples and show some of their properties that are used in deriving complexity results of the simplex algorithm. These properties are then used to show exponential behavior for two pivoting rules; the least index and the maximum coefficient pivoting rule.

The paper is self contained. Next section describes a particular form of the *simplex algorithm*. The Klee–Minty examples and their properties are presented in Section 3. Section 4 is devoted to complexity results.

Simplex Algorithm

In describing our results we find it convenient to use the dictionary form [4] of the simplex algorithm. We will see in the next section that this form exhibits some advantages in describing the properties of the Klee–Minty examples.

Consider the linear problem in standard form

$$\begin{cases} \max & z = c^\top x \\ \text{s.t.} & Ax = b, \\ & x \geq 0, \end{cases} \tag{1}$$

where $c, x \in \mathbf{R}^n$, $b \in \mathbf{R}^m$, $A \in \mathbf{R}^{m\times n}$ and superscript T denotes transposition. Without loss of generality we may assume that A is of full row rank, i. e., $\operatorname{rank}(A) = m$ ($m < n$).

A basis for problem (1) is a set of indices $B \subset \{1, \ldots, n\}$ containing exactly m indices. The element of B, the components of c and x and the columns of A indexed by B are called *basic* while the remaining ones are called *nonbasic*. The set of nonbasic indices is denoted by N, $N = \{1, \ldots, n\} \sim B$. We also denote by $B(N)$ the submatrix of A containing the columns indexed by $B(N)$. The components of a vector x indexed by $B(N)$ are denoted by $x_B(x_N)$.

With this notation at hand the equality constraints of (1) are written in the form

$$Bx_B + Nx_N = b. \tag{2}$$

If B is a nonsingular matrix we can set $x_N = 0$ and compute x_B from (2). Then, we find $x_B = B^{-1}b$. The non singular matrix B is called *basic matrix* or *basis*. The solution $x_N = 0$ and $x_B = B^{-1}b$ is called *basic solution*. If, in addition, it is $x_B = B^{-1}b \geq 0$, then x_B, x_N is a *basic feasible solution*. Geometrically, a basic feasible solution of (1) corresponds to a vertex of the polyhedral set of the feasible region.

If B is nonsingular, we can express the basic variables x_B as a function of the non basic variable x_N. We have from (2) that

$$x_B = -B^{-1}Nx_N + B^{-1}b. \tag{3}$$

Using (3), the objective function of problem (1) is written in the form

$$\begin{aligned} z &= c_B^\top x_B + c_N^\top x_N \\ &= c_B^\top(-B^{-1}Nx_N + B^{-1}b) + c_N^\top x_N \\ &= (-c_B^\top B^{-1}N + c_N^\top)x_N + c_B^\top B^{-1}b. \end{aligned} \tag{4}$$

At every iteration the simplex algorithm constructs the system of equations (3) and (4).

Let the current feasible basis be B. The corresponding system of equations is written in the form

$$\begin{aligned} z &= (-c_B^\top B^{-1}N + c_N^\top)x_N + c_B^\top B^{-1}b, \\ x_B &= -B^{-1}Nx_N + B^{-1}b. \end{aligned} \tag{5}$$

We denote the coefficients of x_N and the constant terms of (5) by H, i. e.,

$$\begin{pmatrix} c_N^\top - c_B^\top B^{-1}N & c_B^\top B^{-1}b \\ -B^{-1}N & B^{-1}b \end{pmatrix} = H.$$

The top row of H, row zero, is devoted to the objective function. Some times we call it *cost row*. The remaining rows are numbered $1, \ldots, m$. The ith row, $1 \leq i \leq m$, corresponds to the basic variable $x_{B[i]}$, where $B[i]$ denotes the ith element of B. Similarly, the jth column of H, $1 \leq j \leq n-m$, corresponds to the nonbasic variable $x_{N[j]}$. The last column of H corresponds to the constant terms. We denote the entries of H by h_{ij}.

It is well known that if $h_{0j} \leq 0$, for $j = 1, \ldots, n-m$, then x_B, x_N is an optimal solution to (1). In this case the algorithm terminates. Otherwise, a nonbasic variable $x_{N[q]} = x_l$ such that $h_{0,N[q]} > 0$ is chosen. Variable x_l is called *entering variable*. If the condition

$$h_{i,N[q]} \geq 0, \quad \text{for } i = 1, \ldots, m,$$

holds, problem (1) is unbounded and the algorithm stops. Otherwise, the basic variable $x_{B[p]} = x_k$, is determined by the following *minimum ratio test*

$$\frac{x_{B[p]}}{-h_{r,N[q]}} = \min\left\{\frac{h_{i,n-m+1}}{-h_{i,N[q]}}:\ 1 \le i \le m, h_{i,N[q]} < 0\right\}.$$

The basic variable x_k is called *leaving variable*. Then, the entering variable x_l takes the place of the leaving variable and vice versa, i. e., it is set

$$B[p] \leftarrow N[q] \quad \text{and} \quad N[q] \leftarrow B[p].$$

Thus, a new basis $\overline{B}$ is constructed and the procedure is repeated. Let $\overline{H}$ be the tableau corresponding to the new basis $\overline{B}$. It is easily seen that

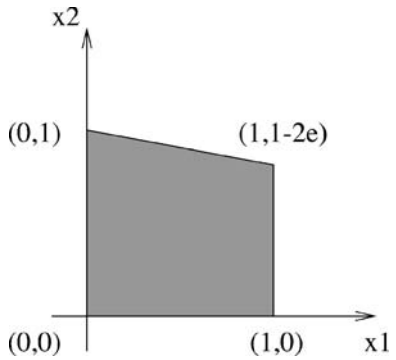
$$\overline{h}_{ij} = \begin{cases} -\frac{h_{pj}}{h_{pq}} & \text{if } i = p, \\ \frac{h_{iq}}{h_{pq}} & \text{if } i \ne p, j = q, \\ h_{ij}\frac{h_{pj}}{h_{pq}} & \text{otherwise.} \end{cases} \tag{6}$$

Klee–Minty Examples

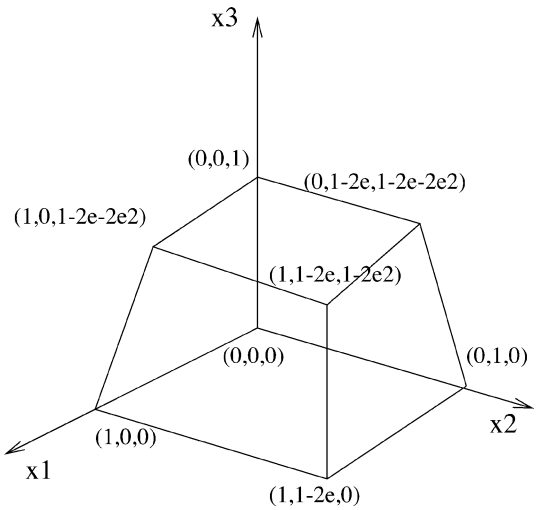
The Klee–Minty examples of order n are the linear problems of the form

$$\begin{cases} \max & \sum_{j=1}^{n} \varepsilon^{n-j} x_j \\ \text{s.t.} & x_1 \le 1 \\ & 2\sum_{j=1}^{i-1} \varepsilon^{i-j} x_j + x_i \le 1, \\ & \quad i = 2, \ldots, n, \\ & x_j \ge 0, \quad j = 1, \ldots, n, \end{cases} \tag{7}$$

where $0 < \varepsilon \le 1/3$. In this section we will show that the feasible region of (7) is a slightly perturbed cube of dimension n, see Fig. 1 and Fig. 2. The optimal solution is $(0, 0, \ldots, 1) \in \mathbf{R}^n$ and the optimal value is 1. A cube of dimension n has 2^n vertices. In the next section we will describe pivoting rules that force the simplex method to pass through all the vertices of the Klee–Minty examples. These pivoting rules require $2^n - 1$ iterations before optimality is reached and, hence, they are exponential.



Linear Programming: Klee–Minty Examples, Figure 1
Feasible region of Klee–Minty example of order $n = 2$



Linear Programming: Klee–Minty Examples, Figure 2
Feasible region of Klee–Minty example of order $n = 3$

The standard form of linear problem (1) is

$$\begin{cases} \max & \sum_{j=1}^{n} \varepsilon^{n-j} x_j \\ \text{s.t.} & x_1 + x_{n+1} = 1, \\ & 2\sum_{j=1}^{i-1} \varepsilon^{i-j} x_j + x_{n+i} = 1, \\ & \quad i = 2, \ldots, n, \\ & x_j \ge 0, \quad j = 1, \ldots, 2n, \end{cases} \tag{8}$$

where x_{n+i}, $1 \le i \le n$, is the slack variable corresponding to the ith inequality constraint of problem (7).

We will be interested in basic solutions of (8) such that, for each $j = 1, \ldots, n$ either x_j or x_{n+j} is basic but

not both. Such a basic solution is called *distinguished*. A tableau corresponding to a dis- tinguished basis is called *distinguished tableau*. In order to facilitate the presentation it is convenient to introduce the set Q of all zero-one n-sequences $(a_1, \ldots, a_n)$ such that

$$a_j = \begin{cases} 0 & \text{if } x_j \text{ is nonbasic and } x_{n+j} \text{ is basic,} \\ 1 & \text{if } x_{n+j} \text{ is nonbasic and } x_j \text{ is basic.} \end{cases}$$

We denote the distinguished basis corresponding to the sequence $(0, \ldots, 0)$ by $\widehat{B}$ and the tableau corresponding to $\widehat{B}$ by $\widehat{H}$. We have $\widehat{B} = \{n+1, \ldots, 2n\}$. It is easily verified that

$$\widehat{h}_{ij} = \begin{cases} \varepsilon^{n-j} & \text{if } i = 0,\ j \le n, \\ -1 & \text{if } 1 \le i = j \le n, \\ 0 & \text{if } 1 \le i < j,\ j \ne n+1, \\ -2\varepsilon^{i-j} & \text{if } i > j, \\ 1 & \text{if } i \ge n \text{ and } j = n+1. \end{cases} \tag{9}$$

Tableau $\widehat{H}$ is sometimes called *initial*.

A distinguished tableau H corresponding to $(a_1, \ldots, a_n) \in Q$ is constructed by starting from $\widehat{H}$ and pivoting only on elements h_{pp} such that $a_p = 1$. Using this procedure and relations (6) and (9) we easily conclude that

$$\begin{aligned} &h_{pp} = -1 \text{ for } p = 1, \ldots, n, \\ &h_{ij} = 0 \text{ for } 1 \le i \le n-1,\ i \le j \le n, \end{aligned} \tag{10}$$

for each distinguished tableau H.

Lemma 1 *Let B be an arbitrary distinguished basis and H the corresponding tableau. Then*

$$h_{ij} + h_{pj}h_{ip} = -h_{ij},\ j < p < n,\ i \ge 2, \tag{11}$$

$$h_{0j} + h_{pj}h_{0j} = 0,\ j < p \le n,\ i = 0. \tag{12}$$

Proof It suffices to show the following induction hypothesis. If the distinguished tableau $\overline{H}$ satisfies (11) and (12) and a pivot operation is performed on h_{rr}, $1 \le r \le n$, resulting in tableau H, then H satisfies (11) and (12) as well. Observe that relations (11) and (12) are satisfied by the initial tableau $\widehat{H}$.

So, assume that H satisfies (11) and (12) and a pivot operation is performed on element $h_{rr} = -1$. Then, we have from (6) and (10)

$$h_{ij} = \begin{cases} \overline{h}_{ij} + \overline{h}_{rj}\overline{h}_{ir}, & i \ne r,\ j \ne r, \\ -\overline{h}_{ij}, & i \ne r,\ j = r, \\ \overline{h}_{ij}, & \text{otherwise.} \end{cases} \tag{13}$$

Combining (13) and the induction hypothesis we have

$$h_{ij} = \begin{cases} -\overline{h}_{ij} & \text{if } i = 0,\ j \le r, \\ -\overline{h}_{ij} & \text{if } i > r,\ j \le r, \\ -\overline{h}_{ij} + \overline{h}_{rj}\overline{h}_{ir} & \text{if } i > r,\ j = n+1, \\ -\overline{h}_{ii} + \overline{h}_{rj}\overline{h}_{ir} & \text{if } i = 0,\ j = n+1, \\ -\overline{h}_{ij}, & \text{otherwise.} \end{cases} \tag{14}$$

There are two cases to be considered, $p \le r$ and $p > r$. From relations (14) we have, for $p \le r$,

$$h_{ij} = -\overline{h}_{ij}, \quad h_{ip} = -\overline{h}_{ip}, \quad h_{pj} = \overline{h}_{pj}$$

and for $p > r$

$$h_{ij} = -\overline{h}_{ij}, \quad h_{ip} = \overline{h}_{ij}, \quad h_{pj} = -\overline{h}_{pj}.$$

In both cases,

$$\begin{aligned} h_{ij} + h_{pj}h_{ip} &= -\overline{h}_{ij} - \overline{h}h_{ip}h_{pj} \\ &= \overline{h}_{ij} = -h_{ij}. \end{aligned}$$

This proves (11). The proof of (12) is similar.

Lemma 1 shows that pivoting on element h_{pp} of a distinguished tableau H is very easily performed. Just change the signs of the entries h_{ij} such that $i = p$ and $j \le p$ or $i > p$ and $j \le p$ and set

$$h_{i,n+1} \leftarrow h_{i,n+1} + h_{ip}h_{p,n+1}$$

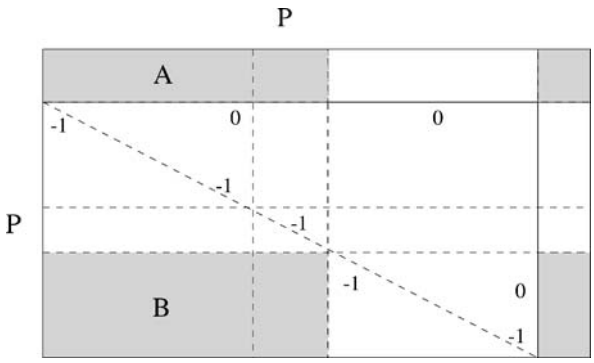
for $i = 0$ or $i > p$.

Figure 3 illustrates the entries of H that change value when pivoting on h_{pp}. In particular, the entries in areas A and B just change sign.

Theorem 2 *Let H be a distinguished tableau of problem (8) and $a = (a_1, \ldots, a_n)$ be the corresponding n-sequence. Then the following relations hold.*

For $i = 1, \ldots, n$ and $j = 1, \ldots, n$ we have

$$h_{ij} = \begin{cases} -1, & i = j, \\ 0, & i < j, \end{cases} \tag{15}$$



Linear Programming: Klee–Minty Examples, Figure 3
Entries of a distinguished tableau *H* that change value after pivoting on element h_{pp}

while for $i > j$,

$$h_{ij} = \begin{cases} -2\varepsilon^{i-j}, & \sum_{k=j}^{i-1} a_k \text{ even}, \\ 2\varepsilon^{i-j}, & \sum_{k=j}^{i-1} a_k \text{ odd}. \end{cases} \tag{16}$$

For $i = 0$ and $j = 1, \ldots, n$ we have

$$h_{0j} = \begin{cases} \varepsilon^{n-j}, & \sum_{k=j}^{n} a_k \text{ even}, \\ -\varepsilon^{n-j}, & \sum_{k=j}^{n} a_k \text{ odd}. \end{cases} \tag{17}$$

For $i = 1, \ldots, n$ and $j = n + 1$ we have

$$h_{i,n+1} = \begin{cases} 1, & i = 1, \\ 1 - \sum_{k=1}^{i-1} a_k h_{ik}, & 2 \le i \le n. \end{cases} \tag{18}$$

Proof The proof is by induction. We assume that distinguished tableau $\overline{H}$ satisfies (15)–(18), and show that tableau H computed by pivoting on $\overline{h}_{pp}$ satisfies (15)–(18) as well. Observe that initial tableau $\widehat{H}$ satisfies (15)–(18).

Let $\overline{a} = (\overline{a}_1, \ldots, \overline{a}_n)$ be the sequence corresponding to tableau $\overline{H}$. Then

$$\overline{a}_j = \begin{cases} a_j, & j \ne p, \\ 1 - a_j, & j = p. \end{cases}$$

Proof of (15)–(16)}. Relations (15) have already been shown. From Lemma 1 we have

$$\overline{h}_{ij} = h_{ij} \quad \text{and} \quad \sum_{k=j}^{i-1} a_k = \sum_{k=j}^{i-1} \overline{a}_k$$

for $i \le p$ or $i > p$ and $j > p$. For $i > p$ and $j \le p$ we have

$$\sum_{k=j}^{i-1} \overline{a}_k = \sum_{k=j}^{i-1} a_k - 2a_p.$$

Hence, if $\sum_{k=j}^{i-1} \overline{a}$ is odd (even), $\sum_{k=j}^{i-1} \overline{a}_k$ is even (odd). Also, from Lemma 1 we have $\overline{h}_{ij} = -h_{ij}$. Hence, (16) holds in all cases.

Proof of (17). It is easily seen that

$$\text{sign}(h_{0j}) = \text{sign}(h_{mj}), \quad \text{for } i \le n.$$

Now the proof comes from the proof of (16).

Proof of (18). If $i \le p$ we have $\overline{h}_{i,n+1} = h_{i,n+1}$. Hence, (18) holds trivially from the induction hypothesis. If $i > p$, then

$$\begin{aligned} \overline{h}_{i,n+1} &= h_{i,n+1} + h_{p,n+1} h_{ip} \\ &= 1 - \sum_{k=1}^{i-1} a_k h_{ik} + \left(1 - \sum_{k=1}^{p-1} a_k h_{pk}\right) h_{ip} \\ &= 1 - \sum_{k=1}^{p-1} a_k (h_{ik} + h_{pk} h_{ip}) \\ &\quad + h_{ip} - a_k h_{pp} h_{ip} - \sum_{k=1}^{i-1} a_k h_{ik} \\ &= 1 - \sum_{k=1}^{p-1} \overline{a} \overline{h}_{ik} - (1 - a_p) \overline{h}_{ip} - \sum_{k=p+1}^{i-1} \overline{a}_k \overline{h}_{ik} \\ &= 1 - \sum_{k=1}^{i-1} \overline{a}_k \overline{h}_{ik}. \end{aligned}$$

Theorem 3 *The feasible region of problem (8) is a slightly perturbed cube.*

Proof It suffices to show that the feasible region has precisely 2^n vertices. We show that each distinguished tableau, H, is feasible and all the adjacent tableaux are distinguished.

From Theorem 2 we have

$$\begin{aligned} h_{i,n+1} &= 1 - \sum_{k=1}^{i-1} a_k h_{ik} \\ &> 1 - 2\varepsilon(1 + \varepsilon + \varepsilon^2 + \cdots) \\ &= 1 - \frac{2\varepsilon}{1-\varepsilon} \geq 0. \end{aligned}$$

We show that if $x_{N[p]}$ is entering, then $x_{B[p]}$ is leaving variable. Let $h_{ip} < 0$ $(i < p)$. We show that

$$\frac{h_{p,n+1}}{h_{pp}} \leq \frac{h_{i,n+1}}{-h_{ip}}.$$

The last relation is equivalently written

$$-h_{ip}(1 - \sum_{k=1}^{p-1} a_k h_{pk}) < 1 - \sum_{k=1}^{i-1} a_k h_{ik}.$$

Using Lemma 1 we get

$$-h_{ip} - 2\sum_{k=1}^{p-1} a_k h_{ik} < 1 - \sum_{k=1}^{i-1} a_k h_{ik}$$

or

$$\begin{aligned} 0 &< 1 + h_{ip} + \sum_{k=1}^{p-1} a_k h_{ik} - \sum_{k=p}^{i-1} a_k h_{ik} \\ &= 1 + \sum_{k=1}^{p-1} a_k h_{ik} + (1 - a_k) h_{ip} + \sum_{k=p+1}^{i-1} a_k h_{ik}. \end{aligned}$$

We have already shown that the last relation holds.

Applications

Now, we are ready to show exponential behavior for some pivoting rules. Let $a \neq b$ be sequences of Q. We write $a < b$ if for the largest index j such that $a_j \neq b_j$ it is $\sum_{j=1}^{n} a_k$ even and $\sum_{k=1}^{n} b_k$ odd.

Let now $f(a)$ be the objective value at the vertex corresponding to $a \in Q$. It is easily seen that $f(a) < f(b)$, if $a < b$. The immediate successor of a sequence $a \in Q$ is the sequence $(a_1, \ldots, a_r, 1 - a_p, a_{p+1}, \ldots, a_n)$, where p is the smallest index such that $\sum_{j=p}^{n} a_j$ is even.

Given a distinguished tableau H, a nonbasic variable $x_{N[q]}$ is called *eligible* if $h_{0,\,N[p]} > 0$.

A pivoting rule that forces the simplex algorithm to pass through all vertices of Klee–Minty examples is the following. For the ease of reference we call it *generic pivoting rule*. Let $a \in Q$ be the sequence corresponding to H. The entering variable is $x_{N[p]}$, where p is the smallest index such that $\sum_{k=p}^{n} a_k$ is even. From Theorem 2 we see that $h_{0,\,N[p]} = \varepsilon^{n-p} > 0$. Hence, $x_{N[p]}$ is eligible, and the generic pivoting rule requires $2^n - 1$ iterations on Klee–Minty examples of order n.

Smallest Index Rule

In the *smallest index rule*, the entering variable is the eligible variable with the smallest index. We show that the smallest index rule, called also *Bland's rule*, performs exponentially on the slightly modified Klee–Minty examples

$$\begin{cases} \max & \displaystyle\sum_{j=1}^{n} \varepsilon^{n-j} x_{2j-1} \\ \text{s.t.} & x_1 \leq 1 \\ & \displaystyle 2\sum_{j=1}^{i-1} \varepsilon^{i-j} x_{2j-1} + x_{2i-1} \leq 1, \\ & \quad i = 2, \ldots, n, \\ & x_j \geq 0, \quad j = 1, \ldots, n. \end{cases} \tag{19}$$

We introduce the slack variable x_{2i} to the ith constraint of problem (19).

Theorem 4 *The least index pivoting rule performs exponentially on example (19).*

Proof We show that the simplex algorithm employing the least index pivoting rule requires $2^n - 1$ iterations when applied to problem (19) and initialized with the basis corresponding to the sequence $(0, \ldots, 0) \in Q$. Clearly, all the bases generated by the algorithm are distinguished i. e. for each i either x_{2i-1} or x_{2i} is basic but not both. Let H be the current distinguished tableau corresponding to the sequence $a \in Q$. Let also p be the smallest index such that $\sum_{k=p}^{n} a_k$ is even.

Then $h_{0,\,N[p]} > 0$ and, hence, $x_{N[p]}$ is eligible. Because of the indexing of the variable in problem (19), $N[p] = 2p$ or $2p - 1$. If q is another index such that $h_{0,\,N[q]} > 0$ ($\sum_{k=q}^{n} a_k$ is even), then $q > p$ and, hence, $N[q] > N[p]$. Hence, the next basis corresponds to the immediate successor of $a \in Q$. This completes the proof.

Largest Coefficient Rule

In the *largest coefficient rule* the entering variable $x_{N[p]}$ is chosen so that

$$h_{0,N[p]} = \max\left\{h_{0,N[j]}:\ h_{0,N[j]} > 0\right\}.$$

This rule solves problem (7) in one iteration when the initial basis is $(0,\ldots,0) \in Q$.

We modify problem (7) as follows. We set

$$\begin{aligned} \varepsilon &= \frac{1}{\mu}, \\ x_j &= y_j e^{2(j-1)}, \end{aligned} \tag{20}$$

and divide the ith constraint by $\varepsilon^{2(i-1)}$ and the objective function by $\varepsilon^{2(n-1)}$. Then, problem (7) is written in the equivalent form

$$\begin{cases} \max & \displaystyle\sum_{j=1}^{n} \mu^{n-j} y_j \\ \text{s.t.} & \displaystyle 2\sum_{j=1}^{i-1} \mu^{i-j} y_j + y_i \le \mu^{2(i-1)}, \\ & \quad i = 1,\ldots,n, \\ & y_j \ge 0, \quad j = 1,\ldots,n, \end{cases} \tag{21}$$

where $\mu = 1/\epsilon \ge 3$.

Theorem 5 *The largest coefficient rule performs exponentially on problem (21).*

Proof Problem (21) is a scaled version of problem (7). Let x_{n+i} be the slack of constraint i. Then, all the results of the previous section, except those involving the RHS, hold true for problem (21). Because of relation (20), $c_j \ge 0$ if and only if $y_j \ge 0$. Hence, every distinguished basis of (21) is feasible. Now, it suffices to show that the generic and the largest coefficient rule coincide when applied to problem (21). However, this statement holds because $\mu > 1$.

See also

- ▶ Criss-cross Pivoting Rules
- ▶ Least-index Anticycling Rules
- ▶ Lexicographic Pivoting Rules
- ▶ Linear Programming

References

1. Avis D, Chvátal V (1978) Notes on Bland's pivoting rule. Math Program Stud 8:24–34
2. Bland RG (1977) New finite pivoting rules for the simplex method. Math Oper Res 2:103–107
3. Chen H, Pardalos PM, Saunders MA (1994) The simplex algorithm with a new primal and dual pivot rule. Oper Res Lett 16:121–127
4. Chvátal V (1983) Linear programming. Freeman, New York
5. Dantzig GB (1963) Linear programming and extentions. Princeton Univ. Press, Princeton
6. Dosios K, Paparrizos K (1996) Resolution of the problem of degeneracy in a primal and dual simplex algorithm. Oper Res Lett 20:45–50
7. Goldfarb D, Sit W (1979) Worst case behavior of the steepest edge simplex method. Discrete Appl Math 1:277–285
8. Jeroslow RG (1973) The simplex algorithm with the pivot rule of maximizing improvement criterion. Discret Math 4:367–377
9. Klee V, Minty GJ (1972) How good is the simplex algorithm? In: Shisha O (ed) Inequalities: III. Acad. Press, New York
10. Murty KG (1983) Linear programming. Wiley, New York
11. Paparrizos K (1989) Pivoting rules directing the simplex method through all feasible vertices of Klee–Minty examples. Opsearch 26(2):77–95
12. Paparrizos K (1993) An exterior point simplex algorithm for (general) linear programming problems. Ann Oper Res 47:497–508
13. Roos C (1990) An exponential example for Terlaky's pivoting rule for the criss - cross simplex method. Math Program 46:78–94
14. Terlaky T (1985) A convergent criss - cross method. Math Oper Statist Ser Optim 16:683–690

Linear Programming Models for Classification

Paul A. Rubin
The Eli Broad Graduate School of Management, Michigan State University, East Lansing, USA

MSC2000: 62H30, 68T10, 90C05

Article Outline

Keywords
Introduction
Models
  Pathologies
  Multiple Group Problems
Methods

See also
References

Keywords

Classification; Discriminant analysis; Linear programming

Introduction

The *G-group classification problem (discriminant problem)* seeks to classify members of some population into one of G predefined groups based on the value of a *scoring function* f applied to a vector $\mathbf{x} \in \Re^p$ of observed attributes. The scoring function is constructed using *training samples* drawn from each group. Of several criteria available for selecting a scoring function, expected accuracy (measured either in terms of frequency of misclassification or average cost of misclassification) predominates. The scoring function f can be vector-valued, but when two groups are involved it is almost always scalar-valued, and scalar functions may be used even when there are more than two groups.

As discussed in [8], statistical methods for constructing scoring functions revolve around estimating, directly or indirectly, the density functions of the distributions of the various groups. In contrast, a number of approaches have been proposed that in essence ignore the underlying distributions and simply try to classify the training samples with maximal accuracy, hoping that this accuracy carries over to the larger population. The use of mathematical programming was suggested at least as early as 1965 by Mangasarian [11]; interest in it grew considerably with the publication of a pair of papers by Freed and Glover in 1981 [3,4], which led to parallel streams of research in algorithm development and algorithm analysis.

Though nonlinear scoring functions can be constructed, virtually all research into mathematical programming methods other than support vector machines [1] restricts attention to linear functions. This is motivated largely by tractability of the mathematical programming problems, but is bolstered by the fact that the Fisher linear discriminant function, the seminal statistically derived scoring function, is regarded as a good choice under a wide range of conditions. For the remainder of this article, we assume f to be linear. Directly maximizing accuracy on the training samples dictates the use of a mixed integer program to choose the scoring function (► Mixed Integer Classification Problems). The number of binary variables in such a formulation is proportional to the size of the training samples, and so computation time grows in a nonpolynomial manner as the sample sizes increase. It is therefore natural that attention turned to more computationally efficient linear programming classification models (LPCMs). Erenguc and Koehler [2] give a thorough survey of the spectrum of mathematical programming classification models as of 1989, and Stam [14] provides a somewhat more recent view of the field. Comparisons, using both "real-world" data and Monte Carlo experiments, of the accuracy of scoring functions produced by mathematical programming models with that of statistically-derived functions has produced mixed results [14], but there is evidence that LPCMs are more robust than statistical methods to large departures from normality in the population (such as populations with mixture distributions, discrete attributes, and outlier contamination).

Models

When $G = 2$ and f is linear and scalar-valued, classification of $\mathbf{x}$ is based without loss of generality on whether $f(\mathbf{x}) < 0$ or $f(\mathbf{x}) > 0$. (If $f(\mathbf{x}) = 0, \mathbf{x}$ can be assigned to either group with equal plausibility. This should be treated as a classification failure.) Barring the degenerate case $f \equiv 0$, the solution set to $f(\mathbf{x}) = 0$ forms a *separating hyperplane*. Ideally, though not often in practice, each group resides within one of the half-spaces defined by that hyperplane. An early precursor to linear programming models, the *perceptron algorithm* [12], constructs an appropriate linear classifier in finite time when the samples are separable, but can fail if the samples are not separable.

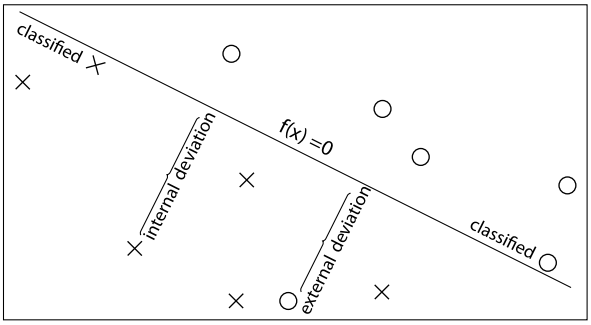
There being no way to count misclassifications in an optimization model without introducing integer variables, LPCMs must employ a surrogate criterion. A variety of criteria have been tried, all revolving around measurements of the displacement of the sample points from the separating hyperplane. Let $f(\mathbf{x}) = \mathbf{w}'\mathbf{x} + w_0$ for some non-null coefficient vector $\mathbf{w} \in \Re^p$ and some scalar w_0. The euclidean distance from $\mathbf{x}$ to the separating hyperplane is easily shown to be $|f(\mathbf{x})| \,/\, \|\mathbf{w}\|$. So the value of the scoring function at each training observa-

tion measures (to within a scalar multiple) how far the observation falls from the separating hyperplane. That distance is in turn identified as either an *internal deviation* or an *external deviation* depending on whether the observation falls in the correct or incorrect half-space. Figure 1 illustrates both types of deviation.

The "hybrid" model of Glover et al. [6] is sufficiently flexible to capture the key features of most two-group models. Let $\mathbf{X}_g$ be an $N_g \times p$ matrix of training observations from group g, and let $\mathbf{0}$ and $\mathbf{1}$ denote vectors of appropriate dimension, all of whose components are 0 and 1 respectively. The core of the hybrid model, to be expanded later, is:

$$\begin{aligned} \min & \sum_{g=1}^{2} \left(\alpha_g \cdot \mathbf{1}'\mathbf{e}_g - \beta_g \cdot \mathbf{1}'\mathbf{d}_g + \gamma_g e_{g0} - \delta_g d_{g0}\right) \\ \text{s.t. } & \mathbf{X}_1\mathbf{w} + w_0 \cdot \mathbf{1} + \mathbf{d}_1 - \mathbf{e}_1 + d_{10} \cdot \mathbf{1} - e_{10} \cdot \mathbf{1} \leq \mathbf{0} \\ & \mathbf{X}_2\mathbf{w} + w_0 \cdot \mathbf{1} - \mathbf{d}_2 + \mathbf{e}_2 - d_{20} \cdot \mathbf{1} + e_{20} \cdot \mathbf{1} \geq \mathbf{0} \\ & \mathbf{w}, w_0 \text{ free; } \mathbf{d}_g, \mathbf{e}_g, d_{g0}, e_{g0} \geq 0\,. \end{aligned}$$

Variables $\mathbf{d}_g$ and $\mathbf{e}_g$ are intended to capture the internal and external deviations respectively of individual observations from group g, while e_{g0} and d_{g0} are intended to capture the maximum (or minimum) external and internal deviations respectively across the sample from group g. (The original hybrid model had $d_{10} = d_{20}$ and $e_{10} = e_{20}$, which is unnecessarily restrictive.) The intent of Glover *et al.* in presenting the hybrid model was to subsume a number of previously proposed models, and so the hybrid model should be viewed as a framework. When applied, not all of the deviation variables need be present. For example, omission of $\mathbf{e}_g$ and $\mathbf{d}_g$ would yield a version of the "MMD" model [2], with e_{g0} the worst external deviation of any group g observation if any is misclassified (in which case $d_{g0} = 0$) and d_{g0} the minimum internal deviation of any group g observation if none is misclassified (in which case $e_{g0} = 0$). On the other hand, omission of e_{g0} and d_{g0} results in a variation of the "MSID" model [2], with the objective function penalizing individual external deviations ($\mathbf{e}_g$) and rewarding individual internal deviations ($\mathbf{d}_g$). The nonnegative objective coefficients α_g, β_g, γ_g, δ_g must be chosen so that the penalties for external deviations exceed the rewards for internal deviations; otherwise, the linear program becomes unbounded, as adding an equal amount to both e_{gn} and d_{gn} improves the objective value.



Linear Programming Models for Classification, Figure 1
Two-Group Problem with Linear Classifier

Pathologies

Due to their focus on minimizing error count, mixed integer classification models tend to be feasible (the trivial function $f \equiv 0$ is often a feasible solution) and bounded (one cannot do better than zero misclassifications). LPCMs, in contrast, tend to be "naturally" feasible but may require explicit bounding constraints. If the training samples are perfectly separable, a solution exists to the partial hybrid model with $e_{gn} = 0$ for all g and n and $d_{gn} > 0$ for some g and n; any positive scalar multiple of that solution is also feasible, and so the objective value is unbounded below. One way to correct this is to introduce bounds on the coefficients of the objective function, say

$$-\mathbf{1} \leq \mathbf{w} \leq +\mathbf{1}.$$

Another potential problem has to do with what is variously referred to as the "trivial" or "unacceptable" solution, namely $f \equiv 0$. Consider the partial hybrid model above. The trivial solution (all variables equal to zero) is certainly feasible, with objective value zero. Given the requirement that the objective coefficients of external deviation variables dominate those of internal deviation variables, any solution with a negative objective value must perfectly separate the training samples. Contrapositively, then, if the training samples cannot be separated, the objective value cannot be less than zero, in which case the trivial solution is in fact optimal. This is undesirable: the trivial function does not classify anything. The trick is to make the trivial solution suboptimal. Some authors try to accomplish this by fixing the

constant term w_0 of the classification function at some nonzero value (typically $w_0 = 1$). The trivial discriminant function $\mathbf{w} = \mathbf{0}$ with nonzero constant term now misclassifies one group completely, and is unlikely to be the model's optimal solution even when the training samples cannot be separated. There is the possibility, however, that the best linear classifier has $w_0 = 0$, in which case this approach dooms the model to finding an inferior solution.

Other approaches include various attempts to make $\mathbf{w} = \mathbf{0}$ infeasible, such as adding the constraint $\|\mathbf{w}\| = 1$. Unfortunately, trying to legislate the trivial solution out of existence results in a nonconvex feasible region, destroying the computational advantage of linear programming. Yet another strategy for weeding out trivial solutions is the introduction of a so-called *normalization* constraint. The normalization constraint proposed by Glover et al. for the hybrid model is

$$\sum_{g=1}^{2} \sum_{n=0}^{N_g} d_{gn} = 1.$$

Various pathologies have been connected to injudicious use of normalization constraints [9,10,13], including: unboundedness; trivial solutions; failure of the resulting discriminant function to adapt properly to rescaling or translation of the data (the optimal discriminant function after scaling or translating the data should be a scaled or translated version of the previously optimal discriminator, and the accuracy should be unchanged); and failure to find a discriminant function with perfect accuracy on the training samples when, in fact, they can be separated (which suggests that the discriminant function found will have suboptimal accuracy on the overall population). Indeed, Glover later changed the normalization of the hybrid model to [5]

$$-N_2 \cdot \mathbf{1}'\mathbf{X}_1\mathbf{w} + N_1 \cdot \mathbf{1}'\mathbf{X}_2\mathbf{w} = 1$$

to avoid some of these pathologies.

Multiple Group Problems

The use of a scalar-valued scoring function in an LPCM with $G > 2$ groups requires the a priori imposition of both a specific ordering and prescribed interval widths on the scores of the groups. This being impractical, attention turns to vector-valued functions. Whether using methods based on statistics or mixed integer programming, a common approach to the multiple group problem is to develop a separate scoring function for each group, and assign observations to the group whose scoring function yields the largest value at that observation. The linear programming analog would be to reward amounts by which the score $f_i(\mathbf{x})$ of an observation $\mathbf{x}$ from group i exceeds each $f_j(\mathbf{x}), j \neq i$ (or $\max_{j \neq i} f_j(\mathbf{x})$) and penalize differences in the opposite direction. This induces a proliferation of deviation variables (on the order of $(G-1)\sum_{g=1}^{G} N_g$). Other approaches may construct discriminant functions for all pairs of groups, or for each group versus all others, and then using a "voting" procedure to classify observations [15].

A good example of the use of a vector-valued scoring function is the work of Gochet et al. [7]. They begin with one scoring function per group, and in cases where two of those functions wind up identical, add additional functions to serve as tie-breakers. Their model is:

$$\begin{aligned}
\min \sum_{g=1}^{G} \sum_{g \neq h=1}^{G} & \mathbf{1}'\mathbf{e}_{gh} \\
\text{s.t. } & \mathbf{X}_g\left(\mathbf{w}_g - \mathbf{w}_h\right) + \left(w_{g0} - w_{h0}\right) \cdot \mathbf{1} + \mathbf{e}_{gh} - \mathbf{d}_{gh} = \mathbf{0} \\
& \sum_{g=1}^{G} \sum_{g \neq h=1}^{G} \mathbf{1}'\left(\mathbf{d}_{gh} - \mathbf{e}_{gh}\right) = q \\
& \mathbf{w}_g, w_{g0} \text{ free; } \mathbf{d}_{gh}, \mathbf{e}_{gh} \geq 0\,.
\end{aligned}$$

The scoring function corresponding to group g is $f_g(\mathbf{x}) = \mathbf{w}_g'\mathbf{x} + w_{g0}$. "Internal" and "external" deviations now represent amounts by which the scores of observations generated by the correct functions exceed or fall short of their scores from functions belonging to other groups. The first constraint is repeated for every pair of groups $g, h = 1, \ldots, G$, $g \neq h$. The second constraint, in which q is an arbitrary positive constant, is a normalization constraint intended to render infeasible both the trivial solution (all $\mathbf{w}_g$ identical) and solutions for which the total of the external deviations exceeds that of the internal deviations. If $\mathbf{w}_g = \mathbf{w}_h$ and $w_{g0} = w_{h0}$ for some $g \neq h$, the model is applied recursively to the subsamples from only those groups (possibly more than just g and h) that yielded identical scoring functions. The additional functions generated are used as tie-breakers.

Methods

The number of constraints in an LPCM approximately equals the number of training observations, while the number of variables can range from slightly more than the number of attributes to slightly more than the sum of the number of observations and the number of attributes, depending on which deviation variables are included in the model. In practice, the number of observations will exceed the number of attributes; indeed, if the difference is not substantial, the model runs the risk of overfitting the scoring function (in the statistical sense). When the number of deviation variables is small, then, the LPCM tends to have considerably more constraints than variables, and a number of authors have suggested solving its dual linear program instead, to reduce the amount of computation. Improvements in both hardware and software have lessened the need for this, but it may still be useful when sample sizes reach the tens or hundreds of thousands (which can happen, for example, when rating consumer credit, and in some medical applications).

See also

- ► Deterministic and Probabilistic Optimization Models for Data Classification
- ► Linear Programming
- ► Mixed Integer Classification Problems
- ► Statistical Classification: Optimization Approaches

References

1. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
2. Erenguc SS, Koehler GJ (1990) Survey of mathematical programming models and experimental results for linear discriminant analysis. Manag Decis Econ 11:215–225
3. Freed N, Glover F (1981) A linear programming approach to the discriminant problem. Decis Sci 12:68–74
4. Freed N, Glover F (1981) Simple but powerful goal programming models for discriminant problems. Eur J Oper Res 7:44–60
5. Glover F (1990) Improved linear programming models for discriminant analysis. Decis Sci 21:771–785
6. Glover F, Keene SJ, Duea RW (1988) A new class of models for the discriminant problem. Decis Sci 19:269–280
7. Gochet W, Stam A, Srinivasan V, Chen S (1997) Multigroup discriminant analysis using linear programming. Oper Res 45:213–225
8. Hand DJ (1997) Construction and assessment of classification rules, Wiley, Chichester
9. Koehler GJ (1989) Unacceptable solutions and the hybrid discriminant model. Decis Sci 20:844–848
10. Koehler GJ (1990) Considerations for mathematical programming models in discriminant analysis. Manag Decis Econ 11:227–234
11. Mangasarian OL (1965) Linear and nonlinear separation of patterns by linear programming. Oper Res 13:444–452
12. Rosenblatt F (1958) The perceptron: A probabilistic model for information storage and organization in the brain. Psychol Rev 65(6):386–408
13. Rubin PA (1991) Separation failure in linear pgogramming discriminant models. Decis Sci 22:519–535
14. Stam A (1997) Nontraditional approaches to statistical classification: Some perspectives on Lp-norm methods. Ann Oper Res 74:1–36
15. Witten IH, Frank E (2005) Data mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, Amsterdam

Linear Space

LEONIDAS PITSOULIS
Princeton University, Princeton, USA

MSC2000: 15A03, 14R10, 51N20

Article Outline

Keywords
See also

Keywords

Linear algebra

Let F be a field, whose elements are referred to as *scalars*. A *linear space* V over F is a nonempty set on which the operations of addition and scalar multiplication are defined. That is, for any $x, y \in V$, we have $x + y \in V$, and for any $x \in V$ and $\alpha \in F$ we have $\alpha\, x \in V$. Furthermore, the following properties must be satisfied:

1) $x + y = y + x$, $\forall\, x, y \in V$.
2) $(x + y) + z = x + (y + z)$, $\forall\, x, y, z \in V$.
3) There exists an element $0 \in V$, such that $x + 0 = x$, $\forall x \in V$.
4) $\forall\, x \in V$, there exists $-x \in V$ such that $x + (-x) = 0$.
5) $\alpha\,(x + y) = \alpha\, x + \alpha\, y$, $\forall\, \alpha \in F$, $\forall\, x, y \in V$.
6) $(\alpha + \beta)x = \alpha x + \beta x$, $\forall\, \alpha, \beta \in F$, $\forall\, x \in V$.

7) $(\alpha\beta)x = \alpha(\beta x)$, $\forall\ \alpha, \beta \in F, \forall\ x \in V$.
8) $1x = x$, $\forall x \in V$.

The elements of V are called *vectors*, and V is also called a *vector space*.

See also

► Affine Sets and Functions
► Linear Programming

Lipschitzian Operators in Best Approximation by Bounded or Continuous Functions

VASANT A. UBHAYA
Department Computer Sci. and Operations Research, North Dakota State University, Fargo, USA

MSC2000: 65K10, 41A30, 47A99

Article Outline

Keywords
Lipschitzian Selection Operators
Examples and Applications
See also
References

Keywords

Approximation problem; Minimum distance problem; Best approximation; Best estimate; Selection; Continuous selection operator; Lipschitzian selection operator; Bounded function; Continuous function; Uniform norm; Quasiconvex function; Convex function; Isotone functions; Majorants and minorants

Stated in simplest terms, this article considers, in an abstract mathematical framework, a curve fitting or estimation problem where a given set of data points f is approximated or estimated by an element from a set K so that the estimate of f is least affected by perturbations in f.

Let X be a normed linear space with norm $\|\cdot\|$ and K be any (not necessarily convex) nonempty subset of X. For any f in X, let

$$d(f, K) = \inf\{\|f - h\| :\ h \in K\} \tag{1}$$

denote the shortest distance from f to K. Let also, for f in X,

$$P(f) = P_K(f) = \{h \in K:\ \|f - h\| = d(f, K)\}.$$

The set-valued mapping P on X is called the *metric projection* onto K. It is also called the *nearest point mapping*, *best approximation operator*, *proximity map*, etc. If $P(f) \neq \emptyset$, then each element in it is called a *best approximation* to (or a *best estimate* of) f from K. In practical curve fitting or estimation problems, f represents the given data and the set K is dictated by the underlying process that generates f. Because of random disturbance or noise, f is in general not in K, and it is required to estimate f by an element of K. See [7,12] and other references given there for a discussion of such problems and the use of various norms or distance functions in approximation. An approximation problem or a minimum distance problem such as (1) involves finding a best approximation, investigating its uniqueness and other properties, and developing algorithms for its computation. If $P(f) \neq \emptyset$ (respectively, $P(f)$ is a singleton) for each $f \in X$, then K is called *proximinal* (respectively, *Chebyshev*).

If K is proximinal, then we define a *selection operator*, or simply a *selection*, to be any (single valued) function T on X into K so that $T(f) \in P(f)$ for every $f \in X$. If K is Chebyshev, then clearly $T = P$ and T is unique. A *continuous selection operator* is a selection T which is continuous. There is a vast literature available on the existence and properties of continuous selections including some survey papers. See, e. g., [1,2,3,6,8] and other references given there. A more difficult problem is finding a *Lipschitzian selection operator* (LSO) i. e., a selection T which satisfies

$$\|T(f) - T(h)\| \leq c(T)\,\|f - h\|, \quad \text{all } f, h \in X, \tag{2}$$

where $c(T)$ (a positive constant depending upon T) is the smallest value satisfying (2). An LSO T is called an *optimal Lipschitzian selection operator* (OLSO) if $c(T) \leq c(T')$ for all LSO T'. If the operator T in (2) is OLSO, then (2) shows that the estimate $T(f)$ of f is least sensitive to changes in the given data f. Consequently, $T(f)$ is the most desirable estimate of f. The concept of an OLSO was introduced in [12] and the existence of an LSO and OLSO was investigated in [13,14,15,16,17]. If X is a Hilbert space and $K \subset X$ is nonempty, closed and

convex, then K is Chebyshev. Then T, which maps f to its unique best approximation, is an LSO, i. e., T satisfies (2) with $c(T) = 1$. For a proof see [5, p. 100]. Since T is unique, it is also trivially OLSO. For other spaces, the results are not so straightforward.

In this paper we present several results which identify LSOs and OLSOs in approximation problems on the space of bounded or continuous functions. We illustrate these results by examples.

Lipschitzian Selection Operators

Let S be any set and B denote the Banach space of real bounded functions f on S with the uniform norm $\|f\| = \sup\{|f(s)|: s \in S\}$. Similarly, when S is topological, denote by $C = C(S)$, the space of real bounded and continuous functions on S, again, with the uniform norm $\|\cdot\|$. Let $X = B$ or C in what follows. We let $f \in X$, $K \subset X$ and $d(f, K)$ as above. We let $d(f) = d(f, K)$ for convenience. For f in X, define $K_f = \{k \in K : k \leq f\}$ and $K'_f = \{k \in K : k \geq f\}$. Let

$$\overline{f}(s) = \sup\left\{k(s):\ k \in K_f\right\}, \quad s \in S,$$
$$\underline{f}(s) = \inf\left\{k(s):\ k \in K'_f\right\}, \quad s \in S.$$

We state three conditions below, they are identical for $X = B$ or C.

1) If $k \in K$, then $k + c \in K$ for all real c.
2) If $f \in X$, then $\overline{f} \in K$.
3) If $f \in X$, then $\underline{f} \in K$.

If $\overline{f}$ and $\underline{f}$ are in K, then they are called the *greatest K-minorant* and the *smallest K-majorant* of f, respectively. Note that condition 2) (respectively, 3)) implies that the pointwise maximum (respectively, minimum) of any two functions in K is also in K. This can be easily established by letting $f = \max\{f_1, f_2\}$ (respectively, $f = \min\{f_1, f_2\}$) where $f_1, f_2 \in K$. We call a $g \in K$ the *maximal* (respectively, *minimal*) *best approximation* to $f \in X$ if $g \geq g'$ (respectively, if $g \leq g'$) for all best approximations g' to f.

Theorem 1 *Consider (1) with $X = B$ or C, and any $K \subset X$.*

a) Assume K is not necessarily convex. Suppose that conditions 1) and 2) hold for K. Then $d(f) = \|f - \overline{f}\|/2$ and $f' = \overline{f} + d(f)$ is the maximal best approximation to f. Also $\|f' - h'\| \leq 2\|f - h\|$ for all $f, h \in X$. The operator T defined $T(f) = f'$ is an LSO with $c(T) = 2$.

b) Assume K is not necessarily convex. Suppose that conditions 1) and 3) hold for K. Then a) holds with $\overline{f}$ replaced by $\underline{f}$ and with $f' = \underline{f} - d(f)$, which is the minimal best approximation to f.

c) Assume K is convex. Suppose that conditions 1), 2) and 3) hold for K. Then a) and b) given above apply. In addition, $d(f) = (\|\underline{f} - \overline{f}\|)/2$. A g in K is a best approximation to f if and only if $\underline{f} - d(f) \leq g \leq \overline{f} + d(f)$. Moreover, if $f' = (\underline{f} + \overline{f})/2$, then f' is a best approximation to f and $\|f' - h'\| \leq \|f - h\|$ for all $f, h \in X$. The operator T defined by $T(f) = f'$ is an OLSO with $c(T) = 1$.

The following theorem shows that the existence of a maximal (respectively, minimal) best approximation to (1) implies condition 2) (respectively, 3)).

Theorem 2 *Consider (1) with $X = B$ or C, and any $K \subset X$. Assume condition 1) holds for K. Assume that the pointwise maximum (respectively, minimum) of two function in K is also in K. Then condition 2) (respectively, 3)) holds if the maximal (respectively, minimal) best approximation to f exists. This best approximation then equals $\overline{f} + d(f)$ (respectively, $\mathrm{f} - d(f)$).*

The above theorems and the next one appear in [14,15]. Their proofs are available there. We now define another approximation problem, closely related to (1). Let

$$\overline{d}(f) = d(f, K_f) = \inf\left\{\|f - h\|:\ h \in K_f\right\}. \qquad (3)$$

The problem is to find a $g \in \{h \in K_f : \|f - h\| = d(f, K_f)\}$, called a *best approximation* to f from K_f.

Theorem 3 *Consider (3) with $X = B$ or C, and any $K \subset X$ which is not necessarily convex.*

a) Suppose that conditions 1) and 2) hold for K. Then $\overline{f}$ is the maximal best approximation to f and $\overline{d}(f) = \left\|f - \overline{f}\right\| = 2d(f)$. The operator T defined by $T(f) = \overline{f}$ is the unique OLSO with $c(T) = 1$.

b) Assume condition 1) holds for K. Assume that the pointwise maximum of two functions in K is also in K. Then condition 2) holds if the maximum best approximation to f exists. This best approximation then equals $\overline{f}$.

Examples and Applications

Example 4 (Approximation by quasiconvex functions.) Let $S \subset \mathbf{R}^n$ be nonempty convex and consider $B = B(S)$. For $C = C(S)$ assume S is nonempty, compact and convex. A function $h \in B$ is called *quasiconvex* if

$$h(\lambda s + (1-\lambda)t) \le \max\{h(s), h(t)\}, \quad \text{for all } s, t \in S, \quad 0 \le \lambda \le 1. \tag{4}$$

Equivalently, h in B is quasiconvex if one of the following conditions holds [9,10]:

- $\{h \le c\}$ is convex for all real c;
- $\{h < c\}$ is convex for all real c.

Let K be the set of all quasiconvex functions in B. It is easy to show that K and $K \cap C$ are closed cones which are not convex and both satisfy condition 1) above (K is a cone if $\lambda h \in K$ whenever $h \in K$ and $\lambda \ge 0$.) The greatest K-minorant of f is called the *greatest quasiconvex minorant* of f. Using (4) it is easy to show that if $f \in B$ then such a minorant $\overline{f}$ exits in B. The next proposition shows that if $f \in C$ then $\overline{f} \in C$.

Let Π be the set of all convex subsets of S. Clearly, φ, $S \in \Pi$. For any $A \subset \mathbf{R}^n$, we denote by $\mathrm{co}(A)$ the *convex hull* of A, i. e., the smallest convex set containing A.

Proposition 5 *Let $f \in X$ and let*

$$f^0(P) = \inf\{f(t):\ t \in S\backslash P\}, \quad P \in \Pi,$$
$$\overline{f}(s) = \sup\left\{f^0(P):\ P \in \Pi,\ s \in S\backslash P\right\},\ s \in S.$$

Then the following holds:

- *If $f \in B$ (respectively, C) then $\overline{f} \in B$ (respectively, C) and is quasiconvex. It is the greatest quasiconvex minorant of f.*
- *An $h \in B$ is the greatest quasiconvex minorant of $f \in B$ if and only if*

$$\{h < c\} = \mathrm{co}\{f < c\} \quad \text{for all real } c. \tag{5}$$

- *An $h \in B$ is the greatest quasiconvex minorant of $f \in C$ if and only if (5) holds or, equivalently, $\{h \le c\} = \mathrm{co}\{h \le c\}$ for all real c.*

This proposition and its proof appear in [15]. The proposition shows that condition 2) holds for K and $K \cap C$. Hence, Theorems 1a) and 3a) apply to $X = B$ and K, and also to $X = C$ and $K \cap C$. In particular, Theorem 1a) shows that in each of these two cases the operator T mapping f to $\overline{f}$ is LSO with $c(T) = 2$. Now the example given in [13, p. 332] shows that T is OLSO.

Example 6 (Approximation by convex functions.) Let $S \subset \mathbf{R}^n$ be nonempty convex and consider $B = B(S)$. A function $h \in B$ is called *convex* if $h(\lambda s + (1-\lambda)t) \le \lambda\, h(s) + (1-\lambda)\, h(t)$, for all $s, t \in S$ and all $0 \le \lambda \le 1$. Clearly, a convex function is quasiconvex. Let K be the set of all convex functions in B. It is easy to show that K is a closed convex cone and satisfies condition 1). The greatest K-minorant of f is called the *greatest convex minorant* of f. It follows at once from the definition of a convex function that if $f \in B$ then such a minorant $\overline{f}$ exists in B. Condition 2) therefore holds for K. Hence, Theorems 1a) and 3a) apply to $X = B$ and K. In particular, the LSO T of Theorem 1a) mapping f to $\overline{f}$ with $c(T) = 2$ can be shown to be an OLSO by using an example as in [13, p. 334].

Now consider approximation of a continuous function by continuous convex functions. For this case we let $S \subset \mathbf{R}^n$ be a polytope which is defined to be the convex hull of a finitely many points in $\mathbf{R}^n$. It is compact, convex and locally simplicial [11]. Let $K \subset C = C(S)$ be the set of continuous convex functions. It is easy to show that K is a closed convex cone. Again condition 1) holds for K. We assert that if $f \in C$, then $\overline{f}$ is convex and continuous. This will establish that $\overline{f}$ is the greatest convex minorant of f. To establish the assertion note that $\overline{f}$ is convex since it is the pointwise supremum of convex functions. Since S is locally simplicial, [11, Corol. 17.2.1; Thm. 10.2] show that $\overline{f}$ is continuous on S. Thus, condition 2) holds for K. Hence Theorems 1a) and 3a) apply to $X = C$ and K. In particular, the LSO T of Theorem 1a) mapping f to $\overline{f}$ with $c(T) = 2$ can be shown to be an OLSO by using the same example as in the bounded case above since the sequence used in that example consists of continuous functions [13].

Example 7 (Approximation by isotone functions.) Let S be any set with partial order $\le$. A *partial order* is a relation $\le$ on S satisfying [4, p. 4]:

- *reflexivity*, i. e., $s \le s$ for all $s \in S$; and
- *transitivity*, i. e., if $s, t, v \in S$, and $s \le t$ and $t \le v$, then $s \le v$.

A partial order is *antisymmetric* if $s \le t$ and $t \le s$ imply $s = t$. We do not include this antisymmetry condition in the partial order for sake of generality. We consider $B = B(S)$ as before, and define a function k in B to be *isotone*

if $k(s) \leq k(t)$ whenever $s, t \in S$ and $s \leq t$. Let $K \subset B$ be the set of all isotone functions. It is easy to see that K is a closed convex cone. It is nonempty since the zero function is in K. It is easy to verify that conditions 1), 2) and 3) apply to K. Thus the greatest isotone minorant $\overline{f}$ and the smallest isotone majorant $\underline{f}$ of an f in B exist. Theorem 1c) and 3a) apply and we conclude that the operator T of Theorem 1c), mapping f to $(\underline{f} + \overline{f})/2$, is OLSO with $c(T) = 1$ [15].

The next proposition gives explicit expressions for $\underline{f}$ and $\overline{f}$. We call a subset E of S a *lower* (respectively, *upper*) set if whenever $t \in E$ and $v \leq t$ (respectively, $t \leq v$), then $v \in E$. For s in S, let $L_s = \{t \in S,\ t \leq s\}$ and $U_s = \{t \in S,\ s \leq t\}$. Then, L_s (respectively, U_s) is the smallest lower (respectively, upper) set containing s, as may be easily seen.

Proposition 8

$$\underline{f}(s) = \sup\{f(t):\ t \in L_s\},$$
$$\overline{f}(s) = \inf\{f(t):\ t \in U_s\}.$$

For a proof, see [15].

Now we consider an application to C. Define $S = \times \{[a_i, b_i]: 1 \leq i \leq n\} \subset \mathbf{R}^n$, where $a_i < b_i$, and let $\leq$ be the usual partial order on vectors. Let $C = C(S)$ and let K be the set of isotone functions in C. It is easy to verify that K is a closed convex cone. Furthermore, if $f \in C$, then $\underline{f}, \overline{f} \in C$. We conclude, as before, that Theorems 1c) and 3a) apply. Various generalizations of this problem exist. See, for example, [12, Sect. 5], [15, Ex. 4.3], and [17].

As was observed in [16], the dual cone of K plays an important role in duality and approximation from K. Some properties of the cone K of isotone functions on a finite partially ordered set S and its dual cone are obtained in [18].

See also

► Convex Envelopes in Optimization Problems

References

1. Deutsch F (1983) A survey of metric selections. RC Sine (ed) Fixed points and Nonexpansive Mappings. In: Contemp. Math., vol 18. Amer Math Soc, Providence, pp 49–71
2. Deutsch F (1992) Selections for metric projections. SP Singh (ed) Approximation Theory, Spline Functions and Applications. Kluwer, Dordrecht, pp 123–137
3. Deutsch F, Li W, Park S-H (1989) Characterization of continuous and Lipschitz continuous metric selections in normed linear spaces. J Approx Theory 58:297–314
4. Dunford N, Schwartz JT (1958) Linear operators, Part I. Interscience, New York
5. Goldstein AA (1967) Constructive real analysis. Harper and Row, New York
6. Li W (1991) Continuous selections for metric projections and interpolating subspaces. In: Brosowski B, Deutsch F, Guddat J (eds) Approximation and Optimization, vol 1. P. Lang, Frankfurt, pp 1–108
7. Liu M-H, Ubhaya VA (1997) Integer isotone optimization. SIAM J Optim 7:1152–1159
8. Nurnberger G, Sommer M (1984) Continuous selections in Chebyshev approximation. In: Brosowski B, Deutsch F (eds) Parametric Optimization and Approximation. Internat Ser Numer Math, vol 72. Birkhäuser, Boston, pp 248–263
9. Ponstein J (1967) Seven kinds of convexity. SIAM Rev 9:115–119
10. Roberts AW, Varberg DE (1973) Convex functions. Acad. Press, New York
11. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton
12. Ubhaya VA (1985) Lipschitz condition in minimum norm problems on bounded functions. J Approx Theory 45:201–218
13. Ubhaya VA (1988) Uniform approximation by quasi-convex and convex functions. J Approx Theory 55:326–336
14. Ubhaya VA (1989) Lipschitzian selections in approximation from nonconvex sets of bounded functions. J Approx Theory 56:217–224
15. Ubhaya VA (1990) Lipschitzian selections in best approximation by continuous functions. J Approx Theory 61:40–52
16. Ubhaya VA (1991) Duality and Lispschitzian selections in best approximation from nonconvex cones. J Approx Theory 64:315–342
17. Ubhaya VA (1992) Uniform approximation by a nonconvex cone of continuous functions. J Approx Theory 68:83–112
18. Ubhaya VA (2001) Isotone functions, dual cones, and networks. Appl Math Lett 14:463–467

Load Balancing for Parallel Optimization Techniques

LBDOP

Ananth Grama[1], Vipin Kumar[2]

[1] Purdue University, West Lafayette, USA

[2] University Minnesota, Minneapolis, USA

MSC2000: 68W10, 90C27

Article Outline

Keywords
Parallel Depth-First Tree Search
Parallel Best-First Tree Search
Searching State Space Graphs
Anomalies in Parallel Search
Applications of Parallel Search Techniques
See also
References

Keywords

Parallel algorithm; Load balancing; Tree search; Graph search

Discrete optimization problems are solved using a variety of state space search techniques. The choice of technique is influenced by a variety of factors such as availability of heuristics and bounds, structure of state space, underlying machine architecture, availability of memory, and optimality of desired solution. The computational requirements of these techniques necessitates the use of large scale parallelism to solve realistic problem instances. In this chapter, we discuss parallel processing issues relating to state space search.

Parallel platforms have evolved significantly over the past two decades. Symmetric multiprocessors (SMPs), tightly coupled message passing machines, and clusters of workstations and SMPs have emerged as the dominant platforms. From an algorithmic standpoint, the key issues of locality of data reference and load balancing are key to effective utilization of all these platforms. However, message latencies, hardware support for shared address space and mutual exclusion, communication bandwidth, and granularity of parallelism all play important roles in determining suitable parallel formulations. A variety of metrics have also been developed to evaluate the performance of these formulations. Due to the nondeterministic nature of the computation, traditional metrics such as parallel runtime and speedup are difficult to quantify analytically. The scalability metric, Isoefficiency, has been used with excellent results for analytical modeling of parallel state space search.

The state spaces associated with typical optimization problems can be fashioned in the form of either a graph or a tree. Exploiting concurrency in graphs is more difficult compared to trees because of the need for replication checking. The availability of heuristics for best-first search imposes constraints on parallel exploration of states in the state space. For the purpose of parallel processing, we can categorize search techniques loosely into three classes: *depth-first tree search techniques* (a tree search procedure in which the deepest of the current nodes is expanded at each step), *best-first tree search techniques* (a tree search procedure in which nodes are expanded based on a global (heuristic) measure of how likely they are to lead to a solution), and *graph search techniques* (a search requiring additional computation for checking if a node has been encountered before, since a node can be reached from multiple paths). Many variants of these basic schemes fall into each of these categories as well.

Parallel Depth-First Tree Search

Search techniques in this class include ordered depth-first search, iterative deepening A∗ (IDA∗), and depth—first branch and bound (DFBB). In all of these techniques, the key ingredient is the depth-first search of a state space (cost-bounded in the case of IDA∗ and DFBB). DFS was among the first applications explored on early parallel computers. This is due to the fact that DFS is very amenable to parallel processing. Each subtree in the state space can be explored independently of other subtrees in the space. In simple DFS, there is no exchange of information required for exploring different subtrees. This implies that it is possible to device simple parallel formulations by assigning a distinct subtree to each processor. However, the space associated with a problem instance can be highly unstructured. Consequently, the work associated with subtrees rooted at different nodes can be very different. Therefore, a naive assignment of a subtree rooted at a distinct node to each processor can result in considerable idling overhead and poor parallel performance. The problem of designing efficient parallel DFS algorithms can be viewed in two steps: the partitioning problem and the assignment problem. The partitioning problem addresses the issue of breaking up a given search space into two subspaces. The assignment problem then maps subspaces to individual processors.

There are essentially two techniques for partitioning a given search space: node splitting and stack splitting. In node splitting, the root node of a subtree is expanded

to generate a set of successor nodes. Each of these nodes represents a distinct subspace. While node splitting is easy to understand and implement, it can result in search spaces of widely varying sizes. Since the objective of the assignment problem is to balance load while minimizing work transfers, widely varying subtask sizes are not desirable. An alternate technique called stack splitting attempts to partition a search space into two by assigning some nodes at all levels leading up to the specified node. Thus if the current node is at level 4, stack splitting will split the stack by assigning some nodes at levels 1, 2, and 3 to each partition. In general, stack splitting results in a more even partitioning of search spaces than node splitting.

We can now formally state the assignment problem for parallel DFS as a mapping of subtasks to processors such that:

- the work available at any processor can be partitioned into independent work pieces as long as it is more than some nondecomposable unit;
- the cost of splitting and transferring work to another processor is not excessive (i. e. the cost associated with transferring a piece of work is much less than the computation cost associated with it);
- a reasonable work splitting mechanism is available; i. e., if work w at one processor is partitioned in 2 parts ψw and $(1-\psi)w$, then $1-\alpha > \psi > \alpha$, where α is an arbitrarily small constant;
- it is not possible (or is very difficult) to estimate the size of total work at a given processor.

A number of mapping techniques have been proposed and analyzed in literature [5,7,8,9,11,16]. These mapping techniques are either initiated by a processor with work (*sender initiated*, the processor with work initiates the work transfer) or a processor looking for work (*receiver initiated*, an idle processor initiates the work transfer). In the *global round robin request* (GRR, idle processors in the global round robin scheme request processors for work in a round-robin fashion using a single (global) counter) receiver initiated scheme, a single counter specifies the processor that must receive the next request for work. This ensures that work requests are uniformly distributed across all processors. However, this scheme suffers from contention at the processor holding the counter. Consequently, the performance of this scheme is poor beyond a certain number of processors. A message combining variant of this scheme (*GRR-M*, a variant of the global round robin scheme in which requests for value of global counter are combined to alleviate contention overheads) relies on combining intermediate requests for the counter into single request. This alleviates the contention and performance bottleneck of the GRR scheme. The *asynchronous round robin balancing* (ARR, i. e. each processor selects a target for work request in a round robin manner using a local counter) uses one counter at each processor. Each processor uses its counter to determine the next processor to query for work. While this scheme balances work requests in a local sense, these requests may become clustered in a global sense. In the *random polling scheme* (RP, i. e. idle processors send work requests to a randomly selected target processor), each processor selects a random processor and requests work. In *near-neighbor load balancing scheme* (NN, i. e. an idle processor requests one of its immediate neighbors for work), processors request work from their immediate neighbors. This scheme has the drawback that localized hot-spots may take a long time to even out.

In sender initiated schemes a processor with work can give some of its work to a selected processor [6,16]. This class of schemes includes the master-slave (MS) and randomized allocation (RA) schemes. In the *MS scheme*, a processor, designated master, generates a fixed number of work pieces. These work-pieces are assigned to processors as they exhaust previously assigned work. The master may itself become the bottleneck when the number of processors is large. Multilevel master-slave algorithms have been used to alleviate this bottleneck. *Randomized allocation schemes* are sender initiated counterparts of RP schemes. In randomized allocation, a processor sends a part of its work to a randomly selected processor.

The performance and scalability of these techniques is often dependent on the underlying architecture. Many of these techniques are, in principle scalable, i. e., they result in linear speedup on increasing the number of processors p as long as the size of the search space grows fast enough with p. It is desirable that this required rate of growth of problem size (also referred to as the iso-efficiency metric [10]) be as small as possible since it allows the use of a larger number of processors effectively for solving a given problem instance. In Table 1, we summarize the iso-efficiency functions of various load balancing techniques.

Scalability results of receiver initiated load balancing schemes for various architectures

Arch Scheme	Shared	H-cube	Mesh (2D)	W/S Cluster
ARR	$p^2 \log p$	$p^2 \log^2 p$	$p^{2.5} \log p$	$p^3 \log p$
NN	$p^2 \log p$	$p^{\log \frac{1+1/\alpha}{2}}$	$k\sqrt{p}$	$p^3 \log p$
GRR	$p^2 \log p$	$p^2 \log p$	$p^2 \log p$	$p^2 \log p$
GRR-M	$p \log p$	$p \log^2 p$	$p^{1.5} \log p$	
RP	$p \log^2 p$	$p \log^2 p$	$p^{1.5} \log^2 p$	$p^2 \log^2 p$
Lower Bound	p	$p \log p$	$p^{1.5}$	p^2

IDA∗ and DFBB search techniques use this basic parallel DFS algorithm for searching state space. In *IDA∗* , each processor has a copy of the global cost bound. Processors perform parallel DFS with this cost bound. At the end of each phase, the cost is updated using a single global operation. Some schemes for allowing different processors to work with different cost bound have also been explored. In this case, a solution cannot be deemed optimal until search associated with all previous cost bounds has been completed. *DFBB* technique uses a global current best solution to bound parallel DFS. Whenever a processor finds a better solution, it updates this global current best solution (using a broadcast in message passing machines and a lock-set in shared memory machines). DFBB and IDA∗ using these parallel DFS algorithms has been shown to yield excellent performance for various optimization problems [3,13,19].

In many optimization problems, the successors of nodes tend to be strongly ordered. In such cases, naive parallel formulations that ignore this ordering information will perform poorly since they are likely to expand a much larger subspace than those that explore nodes in the right order. Parallel DFS formulations for such spaces associate priorities with nodes. Nodes with largest depth and highest likelihood of yielding a solution are assigned the highest priority. Parallel ordered DFS then expands these nodes in a prioritized fashion.

Parallel Best-First Tree Search

Best-first tree search algorithms rely on an *open list* (i. e. a list of unexplored configurations sorted on their quality) to sort available states on the basis of their heuristic solution estimate. If this heuristic solution estimate is guaranteed to be an underestimate (as is the case in the A∗ algorithm), it can be shown that the solution found by BFS is the optimal solution. The presence of a globally ordered open list makes it more difficult to parallelize BFS. In fact, at the first look, BFS may appear inherently serial since a node with higher estimated solution cost must be explored only after all nodes with lower costs have been explored. However, it is possible that there may be multiple nodes with the best heuristic cost. If the number of such nodes is less than the number of available processors, then some of the nodes with poorer costs may also be explored. Since it is possible that these nodes are never explored by the serial algorithm, this may result in excess work by the parallel formulation resulting in deceleration anomalies. These issues of speedup anomalies resulting from excess (or lesser) work done by the parallel formulations of state space search are discussed later.

A simple parallel formulation of BFS uses a global open list. Each processor locks the list, extracts the best available node and unlocks the list. The node is expanded and heuristic estimates are determined for each successor. The open list is locked again and all successors are inserted into the open list. Note that since the state space is a tree, no replication checking is required. The open list is typically maintained in the form of a global heap. The use of a global heap is a source of contention. If the time taken to lock, remove, and unlock the top element of the heap is t_{access} and time for expansion is t_{exp}, then the speedup of the formulation is bounded by $(t_{access} + t_{exp})/t_{access}$. A number of techniques have been developed to effectively reduce the access time [17]. These techniques support concurrent access to heaps stored in shared memory while maintaining strict insertion and deletion ordering. While these increase the upper bound on possible speedup, the performance of these schemes is still bounded.

The contention associated with the global data structure can be alleviated by distributing the open list across processors. Now, instead of p processors sharing a single list, they operate on k distinct open lists. In the limiting case, each processor has its own open list. A simple parallel formulation based on this framework starts off with the initial state in one heap. As additional states are generated, they are shipped off to

the other heaps. As nodes become available in other heaps, processors start exploring associated state space using local BFS. While it is easy to keep all processors busy using this framework, it is possible that some of the processors may expand nodes with poor heuristic estimates that are never expanded by the serial formulation. To avoid this, we must ensure that all open lists have a share of the best globally available nodes. This is also referred to as *quality equalization* (the process of ensuring that all processors are working on regions of state-space of high quality). Since the quality of nodes evolves with time, quality equalization must be performed periodically. Several triggering mechanisms have been developed to integrate quality equalization with load balancing [1,19]. A simple triggering mechanism tracks the best node in the system. The best node in the local heap is compared to the best node in the system and if it is considerably worse, an equalization process is initiated. Alternately, an equalization process may be initiated periodically. The movement of nodes between various heaps may itself be fashioned in a well defined topology. Lists may be organized into rings, shared blackboards, or hierarchical structures. These have been explored for several applications and architectures. Speedups in excess of 950 have been demonstrated on 1024 processor hypercubes in the context of TSPs formulated as best-first tree search problems [2].

Searching State Space Graphs

Searching state space graphs presents additional challenges since we must check for replicated states during search. The simplest strategy for dealing with graphs is to unroll them into trees. The overhead of unrolling a graph into a tree may range from a constant to an exponential factor. If the overhead is a small constant factor, the resulting tree may be searched using parallel DFS or BFS based techniques. However, for most graph search problems, this is not a feasible solution.

Graph search problems rely on a *closed list* (i. e. a list of all configurations that have been previously encountered) that keeps track of all nodes that have already been explored. Closed lists are typically maintained as hash tables for searching. In a shared memory context, insertion of nodes into the closed list requires locking of the list. If there is a single lock associated with the entire list, the list must be locked approximately as many times as the total number of nodes expanded. This represents a serial bottleneck. The bottleneck can be alleviated by associating multiple locks with the closed list. Processors lock only relevant parts of the closed list into which the node is being inserted.

Distributed memory versions of this parallel algorithm physically distribute the closed list across the processors. As nodes are generated, they are hashed to the appropriate processor that holds the respective part of the hash table. Search is performed locally at this processor and the node is explored further at this processor if required. This has two effects: if the hash function associated with the closed list is truly randomized, this has the effect of load balancing using randomized allocation. Furthermore, since nodes are randomly allocated to processors, there is a probabilistic quality equalization for heuristic search techniques. These schemes have been studied by many researchers [14,15]. Assuming a perfectly random hash function, it has been shown that if the number of nodes originating at each processor grows as $O(\log p)$, then each processor will have asymptotically equal number of nodes after the hash operation [15]. Since each node is associated with a communication, this puts constraints on the architecture bandwidth. Specifically, the bisection width of the underlying architecture must increase linearly with the number of processors for this formulation to be scalable.

A major drawback of graph search techniques such as BFS is that its memory requirement grows linearly with the search space. For large problems, this memory requirement becomes prohibitive. Many limited-memory variants of heuristic search have been developed. These techniques rely on retraction or delayed expansion of less promising nodes to reduce memory requirement. In the parallel processing context, retractions lead to additional communication and indexing for parent-child relationships [4].

Anomalies in Parallel Search

As we have seen above, it is possible for parallel formulations to do more or less work than the serial search algorithm. The ratio of nodes searched by the parallel and serial algorithms is called the *search overhead factor* (i. e. the ratio of excess work done by a parallel search formulation with respect to its serial formula-

tion). A search overhead factor of greater than one indicates a deceleration anomaly and less than one indicates an acceleration anomaly. An acceleration anomaly manifests itself in a speedup greater than p on p processors. It can be argued however that in these cases, the base sequential algorithm is suboptimal and a time-multiplexed serialization of the parallel algorithm is in fact a superior serial algorithm.

In DFS and related techniques, parallel formulations might detect solutions available close to the root on alternate branches, whereas serial formulations might search large parts of the tree to the left before reaching this node. Conversely, parallel formulations might also expand a larger number of nodes than the serial version. There are situations, in which parallel DFS can have a search overhead factor of less than 1 on the average, implying that the serial search algorithm in the situation is suboptimal. V. Kumar and V.N. Rao [18] show that if no heuristic information is available to order the successors of a node, then on the average, the speedup obtained by parallel DFS is superlinear if the distribution of solutions is nonuniform.

In BFS, the strength of the heuristic determines the search overhead factor. When strong heuristics are available, it is likely that expanding nodes with lower heuristic values will result in wasted effort. In general, it can be shown that for any given instance of BFS, there exists a number k such that expanding more than k nodes in parallel from a global open list leads to wasted computation [12]. This situation gets worse with distributed open lists since expanded nodes have locally minimum heuristics that are not the best nodes across all open lists. In contrast, the search overhead factor can be less than one if there are multiple nodes with identical heuristic estimates and one of the processors picks the right one.

Applications of Parallel Search Techniques

Parallel search techniques have been applied to a variety of problems such as integer and mixed integer programming, and quadratic assignment for applications ranging from path planning and resource location to VLSI packaging. Quadratic assignment problems from the Nugent–Eschermann test suites with up to 4.8×10^{10} nodes have been solved on parallel machines in days. Traveling salesman problems with thousands of cities and mixed integer programming problems with thousands of integer variable are within the reach of large scale parallel machines. While the use of parallelism increases the range of solvable problems, designing effective heuristic functions is critical. This has the effect of reducing effective branching factor and thus inter-node concurrency. However, the computation of the heuristic can itself be performed in parallel. The use of intra-node parallelism in addition to inter-node parallelism has also been explored. While significant amounts of progress has been made in effective use of parallelism in discrete optimization, with the development of new heuristic functions, opportunities for significant contributions abound.

See also

- ▶ Asynchronous Distributed Optimization Algorithms
- ▶ Automatic Differentiation: Parallel Computation
- ▶ Heuristic Search
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Parallel Computing: Complexity Classes
- ▶ Parallel Computing: Models
- ▶ Parallel Heuristic Search
- ▶ Stochastic Network Problems: Massively Parallel Solution

References

1. Cun BL, Roucairol C (1995) BOB: A unified platform for implementing branch-and-bound like algorithms. Techn. Report Univ. Versailles Saint Quentin 16
2. Dutt S, Mahapatra NR (1994) Scalable load-balancing strategies for parallel A* algorithms. J Parallel Distributed Comput 22(3):488–505, Special Issue on Scalability of Parallel Algorithms and Architectures (Sept. 1994)
3. Eckstein J (1997) Distributed versus centralized storage and control for parallel branch and bound: Mixed integer programming on the CM-5. Comput Optim Appl 7(2):199–220
4. Evett M, Hendler J, Mahanti A, Nau D (1990) PRA*: A memory-limited heuristic search procedure for the connection machine. Proc. Third Symp. Frontiers of Massively Parallel Computation, pp 145–149
5. Finkel RA, Manber U (Apr. 1987) DIB - A distributed implementation of backtracking. ACM Trans Program Languages and Systems 9(2):235–256
6. Furuichi M, Taki K, Ichiyoshi N (1990) A multi-level load balancing scheme for OR-parallel exhaustive search programs on the multi-PSI. Proc. Second ACM SIGPLAN Symp. Principles and Practice of Parallel Programming, pp 50–59

7. Janakiram VK, Agrawal DP, Mehrotra R (1988) A randomized parallel backtracking algorithm. IEEE Trans Comput C-37(12):1665–1676
8. Karp R, Zhang Y (1993) Randomized parallel algorithms for backtrack search and branch-and-bound computation. J ACM 40:765–789
9. Karypis G, Kumar V (oct. 1994) Unstructured tree search on SIMD parallel computers. IEEE Trans Parallel and Distributed Systems 5(10):1057–1072
10. Kumar V, Grama A, Gupta A, Karypis G (1994) Introduction to parallel computing: Algorithm design and analysis. Benjamin Cummings and Addison-Wesley, Redwod City, CA/Reading, MA
11. Kumar V, Grama A, Rao VN (July 1994) Scalable load balancing techniques for parallel computers. J Parallel Distributed Comput 22(1):60–79
12. Lai TH, Sahni S (1984) Anomalies in parallel branch and bound algorithms. Comm ACM 27(6):594–602
13. Lee EK, Mitchell JE (1997) Computational experience of an interior-point algorithm in a parallel branch-and-cut framework. Proc. SIAM Conf. Parallel Processing for Sci. Computing,).
14. Mahapatra NR, Dutt S (July 1997) Scalable global and local hashing strategies for duplicate pruning in parallel A* graph search. IEEE Trans Parallel and Distributed Systems 8(7):738–756
15. Manzini G, Somalvico M (1990) Probabilistic performance analysis of heuristic search using parallel hash tables. Proc. Internat. Symp. Artificial Intelligence and Math.,
16. Ranade AG (1991) Optimal speedup for backtrack search on a butterfly network. Proc. Third ACM Symp. Parallel Algorithms and Architectures,
17. Rao VN, Kumar V (1988) Concurrent access of priority queues. IEEE Trans Comput C-37(12):1657–1665
18. Rao VN, Kumar V (Apr 1993) On the efficicency of parallel backtracking. IEEE Trans Parallel and Distributed Systems 4(4):427–437. Also available as: Techn. Report 90–55, Dept. Computer Sci. Univ. Minnesota
19. Tschvke S, L-ling R, Monien B (1995) Solving the traveling salesman problem with a distributed branch-and-bound algorithm on a 1024 processor network. Proc. 9th Internat. Parallel Processing Symp. (April 1995), 182–189

Local Attractors for Gradient-related Descent Iterations

JOSEPH C. DUNN
Math. Department, North Carolina State University, Raleigh, USA

MSC2000: 49M29, 65K10, 90C06

Article Outline

Keywords
Differentials and Gradients
Gradient-Related Descent Methods
Descent Method Prototypes
The Armijo Steplength Rule
Fixed Points
Local Attractors: Necessary Conditions
Local Attractors: Sufficient Conditions
Nonsingular Attractors
Singular Attractors and Local Convexity
Local Convexity and Convergence Rates
Concluding Remarks
See also
References

Keywords

Unconstrained minimization; Gradient-related descent; Newtonian descent; Singular local attractors; Asymptotic convergence rates

In the classic unconstrained minimization problem, a continuously differentiable real-valued function f is given on a normed vector space $\mathbf{X}$ and the goal is to find points in $\mathbf{X}$ where the *infimum* of f is achieved or closely approximated. Descent methods for this problem start with some nonoptimal point x^0, search for a neighboring point x^1 where $f(x^1) < f(x^0)$, and so on ad infinitum. At each stage, the search is typically guided by a *local* model based on derivatives of f.

If f is convex and every local minimizer is therefore automatically a global minimizer, then well-designed descent methods can indeed generate *minimizing sequences*, i. e., sequences $\{x^k\}$ for which

$$\lim_{k\to\infty} f(x^k) = \inf_{x\in\mathbf{X}} f(x). \tag{1}$$

On the other hand, nonconvex cost functions can have multiple *local minimizers* and any of these may attract the iterates of the standard descent schemes. This behavior is examined here for a large class of gradient-related descent methods, and for local minimizers that need not satisfy the usual nonsingularity hypotheses. In addition, the analytical formulation adopted yields nontrivial local convergence theorems in infinite-dimensional normed vector spaces $\mathbf{X}$. Such theorems are not without computational significance since

they often help to explain emerging trends in algorithm behavior for increasingly refined finite-dimensional approximations to underlying infinite-dimensional optimization problems.

Differentials and Gradients

In a general normed vector space $\mathbf{X}$, the first (Fréchet) differential of f at a point x is a linear function $f'(x)$: $\mathbf{X} \to \mathbf{R}^1$ that satisfies the following conditions:

$$\left|\left\| f'(x) \right\|\right| \stackrel{\text{def}}{=} \sup_{\|u\|=1} \left| f'(x)u \right| < \infty, \tag{2}$$

$$\lim_{\|d\|\to 0} \frac{|f(x+d) - f(x) - f'(x)d|}{\|d\|} = 0. \tag{3}$$

Since $f'(x)d$ is linear in d, condition (2) holds if and only if $f'(x)d$ is continuous in d. Condition (2) is automatically satisfied in any finite-dimensional space $\mathbf{X}$. The remaining condition (3) asserts that $f(x)+ f'(x)d$ asymptotically approximates $f(x+d)$ with an $o(\|d\|)$ error as d approaches zero. At most one linear function can satisfy these conditions in some norm on $\mathbf{X}$. If conditions (2) and (3) do hold in the norm $\|\cdot\|$, then f is said to be (Fréchet) *differentiable* at x (relative to the norm $\|\cdot\|$). If f is differentiable near $x \in \mathbf{X}$ and if

$$\lim_{\|y-x\|\to 0} \left|\left\| f'(y) - f'(x) \right\|\right| = 0, \tag{4}$$

then f is continuously differentiable at x. Note that in finite-dimensional spaces, all norms are equivalent and conditions (2)–(4) hold in any norm if they hold in some norm. However, two norms on the same infinite-dimensional space need not be equivalent, and continuity and differentiability are therefore *norm-dependent properties* at this level of generality.

In the Euclidean space $\mathbf{X} = \mathbf{R}^n$, f is continuously differentiable if and only if the partial derivatives of f are continuous; moreover, when f has continuous partial derivatives, $f'(x)$ is specified by the familiar formula,

$$f'(x)d = \langle \nabla f(x), d \rangle, \tag{5}$$

where $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner product and $\nabla f(x)$ is the corresponding *gradient* of f at x, i. e.,

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$$

and

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right).$$

When $\nabla f(\cdot)$ is continuous, conditions (2)–(4) can be proved for the linear function in (5) with a straightforward application of the chain rule, Cauchy's inequality and the one-dimensional mean value theorem. In addition, it can be shown that $d = \nabla f(x)$ is the unique solution of the equations,

$$\|d\| = \left|\left\| f'(x) \right\|\right| \tag{6}$$

and

$$f'(x)d = \left|\left\| f'(x) \right\|\right| \, \|d\|, \tag{7}$$

where $\|\cdot\|$ and $|||\cdot|||$ are induced by the Euclidean inner product on $\mathbf{R}^n$.

The circumstances in the Euclidean space $\mathbf{R}^n$ suggest a natural extension of the gradient concept in general normed vector spaces $\mathbf{X}$. Let f be differentiable at $x \in \mathbf{X}$. Then any vector $d \in \mathbf{X}$ that satisfies conditions (6)–(7) will be called a *gradient vector* for f at x. Note that the symbols $\|\cdot\|$ and $|||\cdot|||$ in (6)–(7) now signify the norm provided on $\mathbf{X}$ and the corresponding operator norm in (2). Depending on the space $\mathbf{X}$, its norm $\|\cdot\|$ and the point x, conditions (6)–(7) may have no solutions for d, or a unique solution, or infinitely many solutions.

In any finite-dimensional space $\mathbf{X}$, linear functions are continuous, the unit sphere $\{u \in \mathbf{X}: \|u\| = 1\}$ is compact, the supremum in (2) is therefore attained at some unit vector u, and the existence of solutions d for (6)–(7) is consequently guaranteed. On the other hand, f may have infinitely many distinct gradients at a point x if the norm on $\mathbf{X}$ is not strictly convex. For example, if $\mathbf{X} = \mathbf{R}^n$ and $\|x\| = \max_{1 \le i \le n} |x_i|$, then $f'(x)$ is prescribed by (5), and

$$\left|\left\| f'(x) \right\|\right| = \sum_{i=1}^{n} \left| \frac{\partial f}{\partial x_i}(x) \right|.$$

Moreover, d is a gradient vector for f at x if and only if

$$d = \left|\left\| f'(x) \right\|\right| u$$

and

$$u_i \in \operatorname{sgn}\left(\frac{\partial f}{\partial x_i}(x) \right),$$

where $\mathrm{sgn}(t) = \{-1\}$ or $[-1, 1]$ or $\{1\}$ for $t < 0$, $t = 0$ and $t > 0$, respectively.

The existence of gradients can also be proved in reflexive infinite-dimensional spaces $\mathbf{X}$ where bounded linear functions are weakly continuous and closed unit balls are weakly compact. In nonreflexive spaces, conditions (6)–(7) may not have solutions d; however, in any normed vector space and for any fixed arbitrarily small $\nu \in (0, 1)$, the relaxed conditions,

$$\|d\| = \left|\left\|f'(x)\right\|\right| \tag{8}$$

and

$$f'(x)d \geq (1-\nu)\left|\left\|f'(x)\right\|\right| \ \|d\|, \tag{9}$$

always have solutions d. This follows easily from (2) and the meaning of sup. The solutions of (8)–(9) will be called *ν-approximate gradients* of f at x. They occupy a central position in the present formulation of the subject algorithms.

Gradient-Related Descent Methods

If $f'(x) = 0$, then x is called a *stationary point* of f. If $f'(x) \neq 0$, then x is not stationary and the set $\{d \in \mathbf{X}: f'(x)d < 0\}$ is a nonempty open half-space. An element d in this half-space is called a *descent vector* since condition (3) immediately implies that $f(x + td) < f(x)$ when t is positive and sufficiently small. If d is a ν-approximate gradient at a nonstationary point x, then according to (8)–(9),

$$f'(x)(-d) \leq -(1-\nu)\left|\left\|f'\right\|\right|^2 < 0.$$

Hence $-d$ is a descent vector. In particular, if d is a gradient at a nonstationary point x, then $-d$ is a *steepest descent vector* in the sense that

$$f'(x)(-d) \leq f'(x)v, \tag{10}$$

for all $v \in \mathbf{X}$ such that $\|v\| = \|d\|$.

Suppose that ν, μ_1, and μ_2 are fixed positive numbers, with $\nu \in (0, 1)$ and $\mu_2 \geq \mu_1 > 0$. At each $x \in \mathbf{X}$, let $G^\nu(x)$ denote the nonempty set of ν-approximate gradients for f at x and let $G(x)$ be a nonempty subset of the set of all multiples μd with $\mu \in [\mu_1, \mu_2]$ and $d \in G^\nu(x)$, i. e.,

$$\emptyset \neq G(x) \subset \bigcup_{\mu \in [\mu_1, \mu_2]} \mu G^\nu(x). \tag{11}$$

The corresponding set-valued mapping $G(\cdot)$ is referred to here as a *gradient-related set function* with parameters ν, μ_1 and μ_2. In the present development, a gradient-related iterative descent method consists of a gradient-related set function $G(\cdot)$, and a rule that selects a vector $d^k \in G(x^k)$ at each iterate x^k, and another rule that determines the *steplength* parameter $s^k \in (0, 1]$ in the recursion,

$$x^{k+1} = x^k - s^k d^k, \tag{12}$$

once d^k has been chosen. The sequences $\{x^k\}$ generated by this recursion are called gradient-related successive approximations. (For related formulations, see [3,4,10].) The convergence theorems described later in this article depend only on basic properties of gradient-related set functions and the steplengths s^k, hence the precise nature of the rule for selecting d^k in $G(x^k)$ is not important here. This rule may refer to prior iterates $\{x^i\}_{i \leq k}$, or may even be random in nature. There are also many alternative steplength rules that achieve sufficient reductions in f at each iteration in (12) and move the successive approximations x^k toward regions in the domain of f that are interesting in at least a local sense [3,4,10].

Descent Method Prototypes

When gradients of f exist and f attains its infima on lines in $\mathbf{X}$, the steepest descent and exact line minimization rules for d and s yield the prototype *steepest descent method*,

$$x^{k+1} = x^k - s^k d^k, \tag{13}$$

where

$$s^k \in \arg\min_{t>0} f(x^k - td^k) \tag{14}$$

and d^k is any solution of (6)–(7) for $x = x^k$. Note that the actual reduction in f achievable on a steepest descent half-line $\{y \in \mathbf{X}: \exists t > 0,\ y = x - td\}$ may be *smaller* than that attainable on other half-lines, since (10) merely refers to norm-dependent local directional rates of change for f at x. Thus the name of this method is somewhat misleading.

Newtonian descent algorithms also amount to special gradient-related descent methods near a certain type of *nonsingular local minimizer* x^*. These schemes

employ variants of the restricted line minimization steplength rule,

$$s^k \in \arg\min_{t\in(0,1]} f(x^k - td^k), \tag{15}$$

and replace the gradients d^k in a steepest descent iteration by descent vectors that approximate the Newton increment,

$$d^N(x^k) = f''(x^k)^{-1} f'(x^k). \tag{16}$$

Gradient-related descent vector approximations to $d^N(x^k)$ are generated in some neighborhood of x^* by various quasi-Newton auxiliary recursions, provided that the following (interdependent) nonsingularity conditions hold:

i) f is twice continuously (Fréchet) differentiable at x^*;
ii) $f''(x^*)$ satisfies the coercivity condition

$$(f''(x^*)v)v \geq c\,\|v\|^2$$

for some $c > 0$ and all $v \in \mathbf{X}$;
iii) A bounded inverse map $f''(x)^{-1}$ exists for all x sufficiently near x^*;
iv) $f''(\cdot)^{-1}$ is continuous at x^*.

Near a nonsingular local minimizer, the local convergence rates for Newtonian descent methods are generally much faster than the steepest descent convergence rate [8,10]. On the other hand, near singular local minimizers the Newton increments $d^N(x^k)$ and their quasi-Newton approximations are typically not confined to the image sets $G(x^k)$ of some gradient-related set function $G(\cdot)$, and may actually be *undefined* on continuous manifolds in $\mathbf{X}$ containing x^*. Under these circumstances, the unmodified Newtonian scaling principles can degrade or even destroy local convergence. In any case, the convergence properties of Newtonian descent methods near singular local minimizers x^* are not well-understood, and are likely to depend on the higher order structure of the singularity at x^*.

The Armijo Steplength Rule

The line minimization steplength rules in (14) and (15) can be very effective in special circumstances; however, they are more often difficult or impossible to implement, and are not intrinsically 'optimal' in any general sense when coupled with standard descent direction rules based on local models of f. By their very nature, such schemes do not anticipate the effect of current search direction and steplength decisions on the reductions achievable in f in later stages of the calculation. Therefore, over many iterations, the exact line minimization rule may well produce *smaller* total reductions in f than other much simpler steplength rules that merely aim for local reductions in f that are 'large enough' compared with $\|\|f'(x)\|\|$ at each iteration. A. Goldstein and L. Armijo proposed the first practical steplength rules of this kind in [1,8,9] for steepest descent and Newtonian descent methods in $\mathbf{R}^n$. These rules and other related schemes described in [10] and [4] are easily adapted to general gradient-related iterations. The present development focusses on the local convergence properties of the simple Armijo rule described below; however, with minor modifications, the theorems set forth here extend readily to the Goldstein rule and other similar line search formulations.

Let $G(\cdot)$ be a gradient-related set function with parameters ν, μ_1 and μ_2. Fix β in (0, 1) and δ in (0, 1), and for each x in $\mathbf{X}$ and d in $G(x)$ construct $s(x, d) \in (0, 1]$ with the *Armijo steplength rule*,

$$s(x, d) = \max t \tag{17}$$

subject to

$$t \in \{1, \beta, \beta^2, \ldots\}$$

and

$$f(x) - f(x - td) \geq \delta t f'(x)d.$$

When x is not stationary and $-d$ is any descent vector, the rule (17) admits *precisely one* associated steplength $s(x, d) \in (0, 1]$. This is true because β^k converges to zero as $k \to \infty$ and

$$\begin{aligned} &f(x) - f(x - td) \\ &= \delta f'(x)td + (1 - \delta)f'(x)td + o(t) \\ &\geq \delta f'(x)td \end{aligned}$$

for t positive and sufficiently small, in view of (3). When x is stationary, (17) yields $s(x, d) = 1$ trivially for every vector d.

Fixed Points

Descent methods based on gradient-related set functions and Armijo's rule generate sequences $\{x^k\}$ that sat-

isfy

$$x^{k+1} \in T(x^k), \quad k = 0, 1, \ldots, \tag{18}$$

where

$$T(x) \stackrel{\text{def}}{=} \{y\colon\ \exists d \in G(x),\ y = x - s(x,d)d\}\,. \tag{19}$$

The convergence theory outlined in the following sections addresses the behavior of all such Armijo gradient-related sequences near fixed points of the set-valued map $T(\cdot)\colon \mathbf{X} \to 2^{\mathbf{X}}$. The roots of this theory lie in Bertsekas' convergence proof for steepest descent iterates near nonsingular local minimizers in $\mathbf{R}^n$ [2], and subsequent modifications of this proof strategy for gradient projection methods and singular local minimizers in finite-dimensional or infinite-dimensional vector spaces with inner products [6,7]. For related nonlocal theories, see [10] and [4].

By definition, x^* is a *fixed point* of $T(\cdot)$ if and only if

$$x^* \in T(x^*).$$

Since Armijo's rule produces nonzero steplengths $s(x, d)$, it follows that x^* is a fixed point of $T(\cdot)$ if and only if x^* is a stationary point of f. More precisely,

Proposition 1 *Let $T(\cdot)$ be an Armijo gradient-related iteration map in (19). Then for all $x \in \boldsymbol{X}$,*

$$x \in T(x) \Leftrightarrow T(x) = \{x\} \Leftrightarrow G(x) = \{0\} \Leftrightarrow f'(x) = 0. \tag{20}$$

According to Proposition 1, any Armijo gradient-related sequence $\{x^k\}$ that intercepts a fixed point x^* of $T(\cdot)$ must terminate in x^*. Conversely, if $\{x^k\}$terminates in a vector x^*, then x^* is a fixed point of $T(\cdot)$, and hence a stationary point of f. On the other hand, Armijo gradient-related sequences that merely pass *near* some stationary point x^* may or may not converge to x^*.

Local Attractors: Necessary Conditions

A vector x^* is said to be a *local attractor* for an Armijo gradient-related iteration (18) if and only if there is a nonempty open ball,

$$B(x^*, \rho) = \{x \in \mathbf{X}\colon\ \|x - x^*\| < \rho\}$$

with center x^* and radius $\rho > 0$ such that every sequence $\{x^k\}$ which satisfies (18) and enters the ball $B(x^*, \rho)$ must converge to x^*, i. e.,

$$\exists l,\ x^l \in B(x^*, \rho) \Rightarrow \lim_{k\to\infty} \left\| x^k - x^* \right\| = 0. \tag{21}$$

With Proposition 1 and another rudimentary result for gradient-related set functions and Armijo steplengths, it is readily shown that a local attractor must be a *strict local minimizer* of f and an *isolated stationary point* of f.

Proposition 2 *Let $\nu \in (0, 1)$, $\mu_1 > 0$, and $\delta \in (0, 1)$ be fixed parameter values in the gradient-related set function $G(\cdot)$ and Armijo rule (17), and put $c_1 = \delta\,(1 - \nu)\mu_1 > 0$. Then for all $x \in \boldsymbol{X}$ and $d \in G(x)$,*

$$f(x) - f(x - s(x,d)d) \geq c_1 s(x,d) \left\| \left\| f'(x) \right\| \right\|^2. \tag{22}$$

Corollary 3 *Let $T(\cdot)$ be the Armijo gradient-related iteration map in (19). If $\{x^k\}$ is generated by the corresponding gradient-related iteration (18), then for all $k = 0, 1, \ldots,$*

$$f(x^{k+1}) \leq f(x^k) \tag{23}$$

and

$$f'(x^k) \neq 0 \Rightarrow f(x^{k+1}) < f(x^k). \tag{24}$$

Since f is continuous, the claimed necessary conditions for local attractors are now immediate consequences of Proposition 1 and Corollary 3.

Theorem 4 *A vector x^* is a local attractor for an Armijo gradient-related iteration (18) only if x^* is an isolated stationary point and a strict local minimizer of f, i. e., only if there is a nonempty open ball $B(x^*, \rho^*)$ that excludes every other stationary point $x \neq x^*$, and also excludes points $x \neq x^*$ at which $f(x) \leq f(x^*)$.*

The conclusion in Theorem 4 actually applies more generally to set-valued iteration maps $T(\cdot)$ prescribed by any steplength rule that guarantees the fixed-point characterization (20) and the descent property (23)–(24). On the other hand, related converse assertions are tied more closely to special properties of the Armijo rule and its variants, and to certain local uniform growth conditions on f and $\|\|f'(\cdot)\|\|$. If $\mathbf{X}$ is a finite-dimensional space, and x^* is a strict local minimizer

and an isolated stationary point, then the requisite uniform growth conditions automatically hold near x^* and the full converse of Theorem 4 can be proved. If $\mathbf{X}$ is an infinite-dimensional space, the growth conditions become hypotheses in a weaker but still nontrivial partial converse of Theorem 4. This is explained in greater detail below.

Local Attractors: Sufficient Conditions

If x^* is a strict local minimizer of f, then for some $\rho^* > 0$ and all x in the closed ball,

$$\overline{B}(x^*, \rho^*) = \{x \in \mathbf{X}\colon\ \|x - x^*\| \le \rho^*\},$$

the quantity $f(x) - f(x^*)$ is strictly positive when $x \neq x^*$. In finite-dimensional spaces, it is possible to say more. If $\dim \mathbf{X} < \infty$, then for each $t \in (0, \rho^*]$ the corresponding closed annulus,

$$A(t, \rho^*) = \{x\colon\ t \le \|x - x^*\| \le \rho^*\}, \tag{25}$$

is *compact*. Since the function $f(\cdot) - f(x^*)$ is continuous and positive in $A(t, \rho^*)$, it must attain a *positive minimum value* in this set, i. e.,

$$\alpha(t) \stackrel{\text{def}}{=} \min_{x \in A(t,\rho^*)} f(x) - f(x^*) > 0, \tag{26}$$

for each $t \in (0, \rho^*]$. Put $\alpha(0) = 0$ and note that for all t_1, t_2, $0 < t_1 < t_2 \le \rho^* \Rightarrow A(t_1, \rho^*) \supset A(t_2, \rho^*) \Rightarrow \alpha(t_1) \le \alpha(t_2)$. This establishes the following uniform growth property for strict local minimizers in finite-dimensional spaces.

Lemma 5 *Let $\mathbf{X}$ be a finite-dimensional normed vector space. If x^* is a strict local minimizer for f, then there is a positive number ρ^* and a positive definite nondecreasing real-valued function $\alpha(\cdot)$ on $[0, \rho^*]$ such that,*

$$f(x) - f(x^*) \ge \alpha(\|x - x^*\|), \tag{27}$$

for all $x \in \overline{B}(x^*, \rho^*)$.

In infinite-dimensional spaces, the uniform growth condition (27) need not hold at every strict local minimizer; however, when this condition is satisfied, the minimizer x^* has a crucial stability property for gradient-related descent methods. More specifically, suppose that (27) holds and $T(\cdot)$ is an Armijo iteration map (19) with associated parameter $\mu_2 > 0$. Since descent directions can not exist at a local minimizer, the vector x^* must be a stationary point. Fix $\epsilon \in (0, \rho^*]$ and note that since $f'(\cdot)$ is continuous and $f'(x^*) = 0$, there is a $\tau_\epsilon \in (0, \epsilon]$ for which

$$\|x - x^*\| + \mu_2 \left|\left\|f'(x)\right\|\right| < \epsilon \tag{28}$$

for all $x \in B(x^*, \tau_\epsilon)$. Now construct the corresponding set,

$$I(\epsilon) = \{x \in B(x^*, \epsilon)\colon\ f(x) - f(x^*) < \alpha(\tau_\epsilon)\}. \tag{29}$$

By Proposition 2, the simple descent property,

$$f(x - s(x, d)d) \le f(x) \tag{30}$$

holds for all x and all $d \in G(x)$, hence the restriction (28) and the properties of $\alpha(\cdot)$ insure that $I(\epsilon)$ is an *invariant set* for $T(\cdot)$, i. e., $T(x) \subset I(\epsilon)$ for all $x \in I(\epsilon)$. Moreover, since f is continuous, the minimizer x^* is clearly an *interior point* of the set $I(\epsilon)$, and this proves the following stability lemma for Armijo gradient-related iterations (or indeed, any gradient-related method with the descent property (30)).

Lemma 6 *Suppose that the uniform growth condition (27) holds near a local minimizer x^* for f. Let $T(\cdot)$ be an Armijo gradient-related iteration map in (19). Then for every $\epsilon > 0$ there is a corresponding $\rho \in (0, \epsilon]$ such that for all sequences $\{x^k\}$ satisfying (18), and all indices l,*

$$x^l \in B(x^*, \rho) \Rightarrow \forall k \ge l\ x^k \in B(x^*, \epsilon). \tag{31}$$

According to Lemma 6, the uniform growth condition (27) guarantees that an Armijo gradient-related sequence $\{x^k\}$ will remain in any specified arbitrarily small open ball $B(x^*, \epsilon)$ provided $\{x^k\}$ enters a sufficiently small sub-ball of $B(x^*, \epsilon)$. This property alone does not imply that $\{x^k\}$ *converges* to x^*; however, it is an essential ingredient in the local convergence proof outlined below. This proof requires two additional technical estimates for the Armijo rule and gradient-related set functions, a local uniform growth condition for $\left|\left\|f'(\cdot)\right\|\right|$ analogous to (27), and a local uniform continuity hypothesis on $f'(\cdot)$. The first pair of estimates are straightforward consequences of the Armijo rule and the one-dimensional mean value theorem. The last two requirements are automatically satisfied in finite-dimensional spaces, once again because closed bounded sets are compact in these spaces.

Proposition 7 *Let $\nu \in (0, 1)$, $\mu_2 > 0$, $\beta \in (0, 1)$, and $\delta \in (0, 1)$ be fixed parameter values in the gradient-related set function $G(\cdot)$ and Armijo rule (17), and put $c_2 = \delta(1-\nu)\mu_2^{-1} > 0$. Then for all $x \in \boldsymbol{X}$ and $d \in G(x)$,*

$$f(x) - f(x - s(x, d)d) \geq c_2 s(x, d)^2 \|d\|^2 . \tag{32}$$

Moreover, if $s(x, d) < 1$ and $c_3 = (1 - \delta)(1 - \nu)$, then there is a vector ξ in the line segment joining x to $x - \beta^{-1}s(x, d)d$ such that

$$\left|\left\|f'(\xi) - f'(x)\right\|\right| \geq c_3 \left|\left\|f'(x)\right\|\right| \tag{33}$$

and

$$\|\xi - x\| \leq \beta^{-1} s(x, d) \|d\| . \tag{34}$$

Lemma 8 *Let $\boldsymbol{X}$ be a finite-dimensional normed vector space. If x^* is an isolated stationary point for f, then there is a positive number ρ^* and a positive definite nondecreasing real-valued function $\beta(\cdot)$ on $[0, \rho^*]$ such that,*

$$\left|\left\|f'(x)\right\|\right| \geq \beta(\|x - x^*\|), \tag{35}$$

for all $x \in \overline{B}(x^*, \rho^*)$.

The proof of Lemma 8 is similar to the proof of Lemma 5.

Now suppose that the growth conditions (27) and (35) both hold in the ball $\overline{B}(x^*, \rho^*)$, and that $f'(\cdot)$ is uniformly continuous in this ball. By Lemma 6, there is a positive number $\rho \in (0, \rho^*/2]$ such that every sequence $\{x^k\}$ which satisfies (18) and enters the ball $B(x^*, \rho)$, thereafter remains in the larger ball $B(x^*, \rho^*/2)$. But if $\{x^k\}$ is eventually confined to the ball $B(x^*, \rho^*/2)$, then the mean value theorem insures that the nonincreasing real sequence $\{f(x^k)\}$ is bounded below and therefore converges to some finite limit. In this case, the differences $f(x^k) - f(x^{k+1})$ converge to zero and Propositions 2 and 7 therefore yield,

$$\lim_{k\to\infty} s(x^k, d^k) \left|\left\|f'(x^k)\right\|\right|^2 = 0, \tag{36}$$

and

$$\lim_{k\to\infty} s(x^k, d^k) \left\|d^k\right\| = 0, \tag{37}$$

where $d^k \in G(x^k)$ and $s(x^k, d^k)\, d^k = x^{k+1} - x^k$ for all k. It follows easily from the remainder of Proposition 7 and the growth condition (35) that

$$\lim_{k\to\infty} \left|\left\|f'(x^k)\right\|\right| = 0 \tag{38}$$

and therefore

$$\lim_{k\to\infty} \left\|x^k - x^*\right\| = 0. \tag{39}$$

To see that (38) must hold, construct the index sets, $\psi = \{k\colon s(x^k, d^k) = 1\}$ and $\phi = \{k\colon s(x^k, d^k) < 1\}$. If ψ is an infinite set, then,

$$\lim_{\substack{k\in\psi \\ k\to\infty}} \left|\left\|f'(x^k)\right\|\right| = 0,$$

by (36). On the other hand, if ϕ is an infinite set, then

$$\lim_{\substack{k\in\psi \\ k\to\infty}} \left|\left\|f'(x^k)\right\|\right| = 0,$$

by (37), (33), (34), and the local uniform continuity of $f'(\cdot)$. This establishes (38) and proves the following local convergence results.

Theorem 9 *If the uniform growth conditions (27) and (35) hold simultaneously in the closed ball $\overline{B}(x^*, \rho^*)$ for some $\rho^* > 0$, and if $f'(\cdot)$ is uniformly continuous in $\overline{B}(x^*, \rho^*)$ then x^* is a local attractor for Armijo gradient-related iterations (18).*

Corollary 10 *If $\boldsymbol{X}$ is a finite-dimensional normed vector space and x^* is a strict local minimizer and an isolated stationary point for f, then x^* is a local attractor for Armijo gradient-related iterations (18).*

Nonsingular Attractors

The nonsingularity conditions i) and ii) and Taylor's formula imply that in some neighborhood of x^*, the objective function f is convex and satisfies the local growth condition (27) with

$$\alpha(t) = a\, t^2 \tag{40}$$

for some $a > 0$. But if f is locally convex near x^*, then

$$\begin{aligned} f(x) - f(x^*) &\leq f'(x)(x - x^*) \\ &\leq \left|\left\|f'(x)\right\|\right| \|x - x^*\| \end{aligned} \tag{41}$$

near x^*, and therefore (27) and (40) imply (35) with

$$\beta(t) = a\, t. \tag{42}$$

These observations and Theorem 9 immediately yield the following extension of the convergence result in [2] for steepest descent processes in $\mathbf{R}^n$.

Corollary 11 *Every nonsingular local minimizer x^* is a local attractor for Armijo gradient-related iterations (18).*

Singular Attractors and Local Convexity

The growth condition (27) alone does not imply local convexity of f, or condition (35), or the local attractor property. In fact, (27) can hold even if x^* is the limit of some infinite sequence of local minimizers for f. This is readily demonstrated by the following simple function $F: \mathbf{R}^1 \to \mathbf{R}^1$:

$$F(x) = x^2 \left[\sqrt{2} - \sin\left(\frac{5\pi}{6} - \sqrt{3} \ln x^2 \right) \right]. \tag{43}$$

This function has a strict absolute minimizer at $x^* = 0$, with

$$(\sqrt{2} - 1)x^2 \le F(x) \le (\sqrt{2} + 1)x^2$$

for all $x \in \mathbf{R}^1$. However, F also has infinitely many (nonsingular) local minimizers,

$$x_m^{\pm} = \pm \exp\left[\frac{(1 - 8m)\pi}{8\sqrt{3}} \right]$$

for $m = 1, 2, \ldots$, and these local minimizers *accumulate* at 0. Since each $x_m^{\pm}$ is a stationary point and not an absolute minimizer, it follows that F is not convex in any neighborhood of the absolute minimizer at $x^* = 0$, that (35) cannot hold at x^*, and that x^* is not a local attractor for gradient-related descent processes. Evidently, $x^* = 0$ is a singular minimizer for F; in fact, $F''(x)$ does not exist at $x = 0$. (Apart from a minor alteration in one of its constants, (43) is taken directly from [6, Example 1.1]. The erroneous constant in [6] was kindly called to the author's attention by D. Bertsekas.)

The growth conditions (27) and (35) *together* still do not imply convexity of f near x^*, and indeed f may not be convex in any neighborhood of a singular local attractor. This is shown by another function $F: \mathbf{R}^2 \to \mathbf{R}^1$ from [6, Example 1.2], viz.

$$F(x) = x_1^2 - 1.98 x_1 \|x\|^2 + \|x\|^4, \tag{44}$$

where $x = (x_1, x_2)$ and $\|\cdot\|$ is the Euclidean norm in $\mathbf{R}^2$. This function has a singular absolute minimizer at $x^* = 0$, and $F(x)$ and $\|F'(x)\|$ grow like $\|x\|^4$ and $\|x\|^3$, respectively, near 0. On the other hand, since every neighborhood of 0 contains points x where $F'(x)(x - 0)$ is *negative*, it follows that F is not convex (or even pseudoconvex) near 0. Nevertheless, $x^* = 0$ is a local attractor for Armijo gradient-related iterations, according to Corollary 10.

Although f need not be convex near a singular local attractor x^*, there are many instances where some sort of local convexity property is observed. (The function $f(x) = x^4$ provides a simple illustration.) If the local pseudoconvexity condition,

$$\kappa(f(x) - f(x^*)) \le f'(x)(x - x^*), \tag{45}$$

is satisfied for some $\kappa > 0$ and all x in the ball $\overline{B}(x^*, \rho^*)$, then

$$\kappa(f(x) - f(x^*)) \le \|f'(x)\| \, \|x - x^*\|$$

near x^*, and condition (35) follows at once from (27), with

$$\beta(t) = \kappa(\rho^*)^{-1} \alpha(t)$$

for all $t \in [0, \rho^*]$. These considerations immediately yield two additional corollaries of Theorem 9.

Corollary 12 *Suppose that the uniform growth condition (27) holds in the closed ball $\overline{B}(x^*, \rho^*)$ for some $\rho^* > 0$. In addition, suppose that in $\overline{B}(x^*, \rho^*)$, $f'(\cdot)$ is uniformly continuous and f satisfies the pseudoconvexity condition (45). Then x^* is a local attractor for Armijo gradient-related iterations (18).*

Corollary 13 *If $\mathbf{X}$ is a finite-dimensional normed vector space, if x^* is a strict local minimizer for f, and if f satisfies the pseudoconvexity condition (45), then x^* is a local attractor for Armijo gradient-related iterations (18).*

Local Convexity and Convergence Rates

A local version of the convergence rate proof strategy in [5] also works in the present setting when $f'(\cdot)$ is locally Lipschitz continuous and f satisfies the pseudoconvexity condition (45) and the growth condition (27) near x^*. Under these circumstances, the worst-case convergence rate estimate,

$$f(x^k) - f(x^*) = O(k^{-1}), \tag{46}$$

can be proved for Armijo gradient-related sequences $\{x^k\}$ that pass sufficiently near x^*. More refined order estimates are possible if the first two hypotheses hold and

$$f(x) - f(x^*) \ge a \|x - x^*\|^r \tag{47}$$

for some $a > 0$ and $r \in (1, \infty)$, and all $x \in \overline{B}(x^*, \rho^*)$. In such cases, it it can be shown that

$$f(x^k) - f(x^*) = O(k^{-\frac{r}{(r-2)}}) \tag{48}$$

for $r \in (2, \infty)$, and

$$\exists \lambda \in [0, 1) \quad f(x^k) - f(x^*) = O(\lambda^k) \tag{49}$$

for $r \in (1, 2]$. (The latter estimate is comparable to the basic geometric convergence rate theorem for steepest descent iterates near nonsingular local minimizers [2].) The proof strategy in [5] can also produce still more precise local convergence rate estimates that relate the constants implicit in the order estimates (48) and (49) to local Lipschitz constants for $f'(\cdot)$ and parameters in the gradient-related set functions $G(\cdot)$, the growth condition (47), the pseudoconvexity condition (45), and the Armijo steplength rule (17).

In the absence of local convexity assumptions, it is harder to establish analogous asymptotic convergence rate theorems; however, the analysis in [6] and [7] has established $O(k^{-2})$ rate estimates for Hilbert space steepest descent iterations and a class of nonlinear functions f that contains the example (44).

Concluding Remarks

In a finite-dimensional space any two norms are equivalent and it can be seen that the gradient-related property and the local attractor property are therefore norm-invariant qualitative features of set-valued maps $G(\cdot)$: $\mathbf{X} \to 2^{\mathbf{X}}$ and local minimizers x^*. On the other hand, even in finite-dimensional spaces, the Lipschitz constants, growth rate constants, and gradient-related set function parameters in the present formulation are *not* norm-invariant, and this is reflected in norm-dependent convergence rates and norm-dependent size and shape parameters for the domains that are sent to a local attractor x^* by gradient-related iterations. These facts have potentially important computational manifestations when gradient-related methods are applied to large scale finite-dimensional problems that approximate some limiting problem in an infinite-dimensional space. Note that infinite-dimensional spaces can support multiple nonequivalent norms, and a set-valued function $G(\cdot)$ that is gradient-related in one norm need not be gradient-related relative to some other nonequivalent norm. Similarly, the local attractor property for a minimizer x^*, and indeed local optimality itself, are also typically norm-dependent at this level of generality.

See also

- ▶ Conjugate-gradient Methods
- ▶ Large Scale Trust Region Problems
- ▶ Nonlinear Least Squares: Newton-type Methods
- ▶ Nonlinear Least Squares: Trust Region Methods

References

1. Armijo L (1966) Minimization of functions having continuous partial derivatives. Pacific J Math 16:1–3
2. Bertsekas DP (1982) Constrained optimization and Lagrange multiplier methods. Acad. Press, New York
3. Bertsekas DP (1995) Nonlinear programming. Athena Sci., Belmont, MA
4. Daniel JW (1971) Approximate minimization of functionals. Prentice-Hall, Englewood Cliffs, NJ
5. Dunn JC (1981) Global and asymptotic convergence rate estimates for a class of projected gradient processes. SIAM J Control Optim 12:659–674
6. Dunn JC (1987) Asymptotic decay rates from the growth properties of Liapunov functions near singular attractors. J Math Anal Appl 125:6–21
7. Dunn JC (1987) On the convergence of projected gradient processes to singular critical points. J Optim Th Appl 55:203–216
8. Goldstein A (1965) On Newton's method. Numer Math 7:391–393
9. Goldstein A (1965) On steepest descent. SIAM J Control 3:147–151
10. Ortega JM, Rheinboldt WC (1970) Iterative solution of nonlinear equations in several variables. Acad. Press, New York

Location Routing Problem

YANNIS MARINAKIS
Department of Production Engineering and Management, Decision Support Systems Laboratory, Technical University of Crete, Chania, Greece

MSC2000: 90B06, 90B80

Article Outline

Introduction
Variants of the Location Routing Problem

Exact Algorithms for the Solution of the Location Routing Problem
Heuristic Algorithms for the Solution of the Location Routing Problem
Metaheuristic Algorithms for the Solution of the Location Routing Problem
References

Introduction

In the last few years, the need for an integrated logistic system has become a primary objective of every company manager. Managers recognize that there is a strong relation between the location of facilities, the allocation of suppliers, vehicles, and customers to the facilities, and the design of routes around the facilities. In a **location routing problem (LRP)**, the optimal number, the capacity, and the location of facilities are determined, and the optimal set of vehicle routes from each facility is also sought.

In most location models, it is assumed that the customers are served directly from the facilities being located. Each customer is served on his or her own route. In many cases, however, customers are not served individually from the facilities. Rather, customers are consolidated into routes that may contain many customers. One of the reasons for the added difficulty in solving these problems is that there are far more decisions that need to be made by the model. These decisions include:

- How many facilities to locate,
- Where the facilities should be,
- Which customers to assign to which depots,
- Which customers to assign to which routes,
- In what order customers should be served on each route.

In the LRP, a number of facilities are located among candidate sites and delivery routes are established for a set of users in such a way that the total system cost is minimized. As Perl and Daskin [51] pointed out, LRPs involve three interrelated, fundamental decisions: where to locate facilities, how to allocate customers to facilities, and how to route vehicles to serve customers.

The difference between the LRP and the classic vehicle routing problem is that not only routing must be designed but the optimal depot location must be simultaneously determined as well. The main difference between the LRP and the classical location-allocation problem is that, once the facility is located, the former requires a visitation of customers through tours while the latter assumes that the customer will be visited from the vehicle directly, and then the vehicle will return to the facility without serving any other customer ([47]). In general terms, the combined location routing model solves the joint problem of determining the optimal number, capacity, and location of facilities serving more than one customer and finding the optimal set of vehicle routes. In the LRP, the distribution cost is decreased due to the assignment of the customers to vehicles while the main objective is the design of the appropriate routes of the vehicles.

Variants of the Location Routing Problem

Laporte et al. [39] considered three variants of LRPs, including (1) capacity-constrained vehicle routing problems, (2) cost-constrained vehicle routing problems, and (3) cost-constrained location routing problems. The authors examined multidepot, asymmetrical problems and developed an optimal solution procedure that enables them to solve problems with up to 80 nodes. Chan et al. [11] solved a multidepot, multivehicle location routing problem with stochastically processed demands, which are defined as demands that are generated upon completing site-specific service on their predecessors. Min et al. [47] synthesized the past research and suggested some future research directions for the LRP. An extended recent literature review is included in the survey paper published by Nagy and Salhi [48]. They proposed a classification scheme and looked at a number of problem variants. The most important exact and heuristic algorithms were presented and analyzed in this survey paper.

Exact Algorithms for the Solution of the Location Routing Problem

A number of exact algorithms for the problem was presented by Laporte et al. [38]. Applications and formulations and exact and approximation algorithms for LRPs under capacity and maximum cost restrictions are studied in the survey of Laporte [34]. Nonlinear programming exact algorithms for the solution of the LRP have been proposed in [20,61]. Dynamic programming exact algorithms for the solution of the LRP have been proposed in [5]. Integer programming exact al-

gorithms for the solution of the LRP have been proposed in [35,37,46]. Mixed integer goal programming exact algorithms for the solution of the LRP have been proposed in [65]. Two branching strategies have been proposed in [36]. An iterative exact procedure has been proposed in [9]. A branch-and-bound technique on the LP relaxation has been proposed in [17].

Heuristic Algorithms for the Solution of the Location Routing Problem

The LRP is very difficult to solve using exact algorithms, especially if the number of customers or the candidate for location facilities is very large due to the fact that this problem belongs to the category of ***NP-hard problems***, i. e. there are no known polynomial-time algorithms that can be used to solve them. Madsen [43] presented a survey of heuristic methods. Christofides and Eilon [16] were the first to consider the problem of locating a depot from which customers are served by tours rather than individual trips. They proposed an approximation algorithm for the solution of the problem. Watson-Gandy and Dohrn [63] proposed an algorithm where the problem is solved by transforming its location part into an ordinary location problem using the Christofides–Eilon approximation algorithm. The routing part of the algorithm is solved using the Clarke and Wright algorithm. Jacobsen and Madsen [31] proposed three algorithms. The first is called a tree-tour heuristic. The second is called ALA–SAV and is a three-phase heuristic, where in the first phase a location–allocation problem is solved and in the second and third phases a Clarke and Wright heuristic is applied for solving the problem. Finally, the third proposed algorithm is called SAV–DROP and is a heuristic algorithm that combines the Clarke–Wright method and the DROP algorithm. A two-phase heuristic is presented in [4], where in the first phase the set of open plants is determined and a priori routes are considered, while in the second phase the routes are optimized. Other two-phase heuristics have been proposed in [7,12,13,30,33,42,49,50,58]. Cluster analysis algorithms are presented in [6,18,60]. Iterative approaches have been proposed by [27,59]. Min ([46]) considered a two-level location–allocation problem of terminals to customer clusters and supply sources using a hierarchical approach consisting of both exact and heuristic procedures. Insertion methods have been proposed in [15]. A partitioning heuristic algorithm is proposed in [35], and a sweep heuristic is proposed in [21].

Metaheuristic Algorithms for the Solution of the Location Routing Problem

Several metaheuristic algorithms have been proposed for the solution of the LRP. In what follows, an analytical presentation of these algorithms is given.

- **Tabu search (TS)** was introduced by Glover [22,23] as a general iterative metaheuristic for solving combinatorial optimization problems. Computational experience has shown that TS is a well-established approximation technique that can compete with almost all known techniques and that, by its flexibility, can beat many classic procedures. It is a form of local neighbor search. Each solution S has an associated set of neighbors $N(S)$. A solution $S' \in N(S)$ can be reached from S by an operation called a *move*. TS can be viewed as an iterative technique that explores a set of problem solutions by repeatedly making moves from one solution S to another solution S' located in the neighborhood $N(S)$ of S [24]. TS moves from a solution to its best admissible neighbor, even if this causes the objective function to deteriorate. To avoid cycling, solutions that have been recently explored are declared *forbidden or tabu* for a number of iterations. The tabu status of a solution is overridden when certain criteria (*aspiration criteria*) are satisfied. Sometimes, *intensification* and *diversification* strategies are used to improve the search. In the first case, the search is accentuated in the promising regions of the feasible domain. In the second case, an attempt is made to consider solutions in a broad area of the search space. Tuzun and Burke [62] proposed a two-phase tabu search architecture for the solution of the LRP. TS algorithms for the LRP are also presented in [10,14,41,45,57].
- **Simulated annealing (SA)** [1,3,32] plays a special role within local search for two reasons. First, SA appears to be quite successful when applied to a broad range of practical problems. Second, some threshold accepting algorithms such as SA have a stochastic component, which facilitates a theoretical analysis of their asymptotic convergence. SA [2] algorithms are stochastic algorithms that allow random

uphill jumps in a controlled fashion in order to provide possible escapes from poor local optima. Gradually the probability allowing the objective function value to increase is lowered until no more transformations are possible. SA owes its name to an analogy with the annealing process in condensed-matter physics, where a solid is heated to a maximum temperature at which all particles of the solid randomly arrange themselves in the liquid phase, followed by cooling through careful and slow reduction of the temperature until the liquid is frozen with the particles arranged in a highly structured lattice and minimal system energy. This ground state is reachable only if the maximum temperature is sufficiently high and the cooling sufficiently slow. Otherwise a metastable state is reached. The metastable state is also reached with a process known as quenching, in which the temperature is instantaneously lowered. Its predecessor is the so-called Metropolis filter. Wu et al. [64] proposed an algorithm that divides the original problem into two subproblems, i. e., the location–allocation problem and the general vehicle routing problem, respectively. Each subproblem is, then, solved in a sequential and iterative manner by the SA algorithm embedded in the general framework for the problem-solving procedure. SA algorithms for the LRP are presented in [8,40,41].

- **Greedy randomized adaptive search procedure (GRASP)** [56] is an iterative two-phase search method that has gained considerable popularity in combinatorial optimization. Each iteration consists of two phases, a construction phase and a local search procedure. In the construction phase, a randomized greedy function is used to build up an initial solution. This randomized technique provides a feasible solution within each iteration. This solution is then exposed for improvement attempts in the local search phase. The final result is simply the best solution found over all iterations. Prins et al. [52] proposed a GRASP with a path-relinking phase for the solution of the capacitated location routing problem.
- **Genetic algorithms (GAs)** are search procedures based on the mechanics of natural selection and natural genetics. The first GA was developed by John H. Holland in the 1960s to allow computers to evolve solutions to difficult search and combinatorial problems such as function optimization and machine learning [28]. Genetic algorithms offer a particularly attractive approach to problems like location routing problems since they are generally quite effective for the rapid global search of large, nonlinear, and poorly understood spaces. Moreover, GAs are very effective in solving large-scale problems. GAs [25] mimic the evolution process in nature. They are based on an imitation of the biological process in which new and better populations among different species are developed during evolution. Thus, unlike most standard heuristics, GAs use information about a population of solutions, called individuals, when they search for better solutions. A GA is a stochastic iterative procedure that maintains the population size constant in each iteration, called a generation. Their basic operation is the mating of two solutions to form a new solution. To form a new population, a binary operator called a crossover and a unary operator called a mutation are applied [54,55]. Crossover takes two individuals, called parents, and produces two new individuals, called offspring, by swapping parts of the parents. Marinakis and Marinaki [44] proposed a bilevel GA for a real-life LRP. A new formulation based on bilevel programming was proposed. Based on the fact that in the LRP decisions are made at a strategic level and at an operational level, we formulate the problem in such a way that in the first level, the decisions of the strategic level are made, namely, the top manager finds the optimal location of the facilities, while in the second level, the operational-level decisions are made, namely, the operational manager finds the optimal routing of vehicles. Other evolutionary approaches for the solution of the LRP have been proposed in [29,53].
- **Variable neighborhood search (VNS)** is a metaheuristic for solving combinatorial optimization problems whose basic idea is systematic change of a neighborhood within a local search [26]. VNS algorithms for the LRP are presented in [45].
- The **ant colony optimization (ACO)** metaheuristic is a relatively new technique for solving combinatorial optimization problems (COPs). Based strongly on the ant system (AS) metaheuristic developed by Dorigo, Maniezzo, and Colorni [19], ACO is derived

from the foraging behavior of real ants in nature. The main idea of ACO is to model the problem as the search for a minimum cost path in a graph. Artificial ants walk through this graph looking for good paths. Each ant has a rather simple behavior so that it will typically only find rather poor-quality paths on its own. Better paths are found as the emergent result of the global cooperation among ants in the colony. An ACO algorithm consists of a number of cycles (iterations) of solution construction. During each iteration a number of ants (which is a parameter) construct complete solutions using heuristic information and the collected experiences of previous groups of ants. These collected experiences are represented by a digital analog of trail pheromone that is deposited on the constituent elements of a solution. Small quantities are deposited during the construction phase while larger amounts are deposited at the end of each iteration in proportion to solution quality. Pheromone can be deposited on the components and/or the connections used in a solution depending on the problem. ACO algorithms for the LRP are presented in [8].

References

1. Aarts E, Korst J (1989) Simulated Annealing and Boltzmann Machines – A Stochastic Approach to Combinatorial Optimization and Neural Computing. John Wiley and Sons, Chichester
2. Aarts E, Korst J, Van Laarhoven P (1997) Simulated Annealing. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, Chichester, pp 91–120
3. Aarts E, Ten Eikelder HMM (2002) Simulated Annealing. In: Pardalos PM, Resende MGC (eds) Handbook of Applied Optimization. Oxford University Press, New York, pp 209–221
4. Albareda-Sambola M, Diaz JA, Fernandez E (2005) A Compact Model and Tight Bounds for a Combined Location-Routing Problem. Comput Oper Res 32(3):407–428
5. Averbakh I, Berman O (1994) Routing and Location-Routing p-Delivery Men Problems on a Path. Transp Sci 28(2):162–166
6. Barreto S, Ferreira C, Paixao J, Santos BS (2007) Using Clustering Analysis in a Capacitated Location-Routing Problem. Eur J Oper Res 179(3):968–977
7. Bookbinder JH, Reece KE (1988) Vehicle Routing Considerations in Distribution System Design. Eur J Oper Res, 37:204-213
8. Bouhafs L, Hajjam A, Koukam A (2006) A Combination of Simulated Annealing and Ant Colony System for the Capacitated Location-Routing Problem. Knowl-Based Intelligent Inf Eng Syst, LNCS 4251:409–416
9. Burness RC, White JA (1976) The Traveling Salesman Location Problem. Transp Sci 10(4):348–360
10. Caballero R, Gonzalez M, Guerrero FM, Molina J, Paralera C (2007) Solving a Multiobjective Location Routing Problem with a Metaheuristic Based on Tabu Search. Application to a Real Case in Andalusia. Eur J Oper Res 177(3):1751–1763
11. Chan Y, Carter WB, Burnes MB (2001) A Multiple-Depot, Multiple-Vehicle, Location-Routing Problem with Stochastically Processed Demands. Comput Oper Res 28:803–826
12. Cappanera P, Gallo G, Maffioli F (2003) Discrete Facility Location and Routing of Obnoxious Activities. Discret Appl Math 133(1–3):3–28
13. Chan Y, Baker SF (2005) The Multiple Depot, Multiple Traveling Salesmen Facility-Location Problem: Vehicle Range, Service Frequency, Heuristic Implementations. Math Comput Model 41(8-9):1035–1053
14. Chiang WC, Russell RA (2004) Integrating Purchasing and Routing in a Propane Gas Supply Chain. Eur J Oper Res 154(3):710–729
15. Chien TW (1993) Heuristic Procedures for Practical-sized Uncapacitated Location-Capacitated Routing Problems. Decis Sci 24(5):995–1021
16. Christofides N, Eilon S (1969) Expected Distances for Distribution Problems. Oper Res Q 20:437–443
17. Daskin MS (1987) Location, Dispatching, Routing Models for Emergency Services with Stochastic Travel Times. In: Ghosh A, Rushton G (eds) Spatial Analysis and Location-Allocation Models. Von Nostrand Reinhold Company, NY, pp 224–265
18. Dondo R, Cerda J (2007) A Cluster-Based Optimization Approach for the Multi-Depot Heterogeneous Fleet Vehicle Routing Problem with Time Windows. Eur J Oper Res 176(3):1478–1507
19. Dorigo M, Stutzle T (2004) Ant Colony Optimization, A Bradford Book. MIT Press Cambridge, MA, London
20. Ghosh JK, Sinha SB, Acharya D (1981) A Generalized Reduced Gradient Based Approach to Round-trip Location Problem. In: Jaiswal NK (eds) Scientific Management of Transport Systems. Amsterdam, Holland, pp 209–213
21. Gillett B, Johnson J (1976) Multi-Terminal Vehicle-Dispatch Algorithm. Omega 4(6):711–718
22. Glover F (1989) Tabu Search I. ORSA J Compu 1(3):190–206
23. Glover F (1990) Tabu Search II. ORSA J Compu 2(1):4–32
24. Glover F, Laguna M, Taillard E, de Werra D (eds) (1993) Tabu Search. JC Baltzer AG, Science Publishers, Basel
25. Goldberg DE (1989) Genetic Algorithms in Search, Optimization, Machine Learning. Addison-Wesley, Reading Massachussets
26. Hansen P, Mladenovic N (2001) Variable Neighborhood Search: Principles and Applications. Eur J Oper Res 130:449–467

27. Hansen PH, Hegedahl B, Hjortkajaer S, Obel B (1994) A Heuristic Solution to the Warehouse Location-Routing Problem. Eur J Oper Res 76:111–127
28. Holland JH (1975) Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI
29. Hwang HS (2002) Design of Supply-Chain Logistics System Considering Service Level. Comput Ind Eng 43(1–2):283–297
30. Jacobsen SK, Madsen OBG (1978) On the Location of Transfer Points in a Two-Level Newspaper Delivery System – A Case Study. Presented at The International Symposium on Locational Decisions. The Institute of Mathematical Statistics and Operations Research, The Technical University of Denmark, Lyngbyn Denmark, pp 24–28
31. Jacobsen SK, Madsen OBG (1980) A Comparative Study of Heuristics for Two Level Routing Location Problem. Eur J Oper Res 5:378–387
32. Kirkpatrick S, Gelatt CD, Vecchi MP (1982) Optimization by Simulated Annealing. Science 220:671–680
33. Laoutaris N, Zissimopoulos V, Stavrakakis I (2005) On the Optimization of Storage Capacity Allocation for Content Distribution. Comput Netw 47(3):409–428
34. Laporte G (1988) Location Routing Problems. In: Golden BL et al (eds) Vehicle Routing: Methods and Studies. North-Holland, Amsterdam, pp 163–198
35. Laporte G, Dejax PJ (1989) Dynamic Location-Routing Problems. J Oper Res Soc 40(5):471–482
36. Laporte G, Nobert Y (1981) An Exact Algorithm for Minimizing Routing and Operating Costs in Depot Location. Eur J Oper Res 6:224–226
37. Laporte G, Nobert Y, Arpin D (1986) An Exact Algorithm for Solving a Capacitated Location-Routing Problem. Ann Oper Res 6:293–310
38. Laporte G, Nobert Y, Pelletier P (1983) Hamiltonian Location Problems. Eur J Oper Res 12(1):82–89
39. Laporte G, Nobert Y, Taillefer S (1988) Solving a Family of Multi-depot Vehicle Routing and Location Routing Problems. Transp Sci 22:161–172
40. Lin CKY, Chow CK, Chen A (2002) A Location-Routing-Loading Problem for Bill Delivery Services. Comput Ind Eng 43(1–2):5–25
41. Lin CKY, Kwok RCW (2006) Multi-Objective Metaheuristics for a Location-Routing Problem with Multiple Use of Vehicles on Real Data and Simulated Data. Eur J Oper Res 175(3):1833–1849
42. Liu SC, Lee SB (2003) A Two-Phase Heuristic Method for the Multi-Depot Location Routing Problem Taking Inventory Control Decisions Into Consideration. Int J Adv Manuf Technol 22(11-12):941–950
43. Madsen OBG (1983) Methods for Solving Combined Two Level Location Routing Problems of Realistic Dimension. Eur J Oper Res 12(3):295–301
44. Marinakis Y, Marinaki M (2008) A Bilevel Genetic Algorithm for a Real Life Location Routing Problem. Int J Logist 11(1):49–65
45. Melechovsky J, Prins C, Calvo RW (2005) A Metaheuristic to Solve a Location-Routing Problem with Non-Linear Costs. J Heurist 11(5-6):375–391
46. Min H (1996) Consolidation Terminal Location-Allocation and Consolidated Routing Problems. J Bus Logist 17(2): 235–263
47. Min H, Jayaraman V, Srivastava R (1998) Combined Location-Routing Problems: A Synthesis and Future Research Directions. Eur J Oper Res 108:1–15
48. Nagy G, Salhi S (2007) Location-Routing: Issues, Models and Methods. Eur J Oper Res 177:649–672
49. Nambiar JM, Gelders LF, Van Wassenhove LN (1981) A Large Scale Location-Allocation Problem in the Natural Rubber Industry. Eur J Oper Res 6:183–189
50. Perl J, Daskin MS (1984) A Unified Warehouse Location-Routing Methodology. J Bus Logist 5(1):92–111
51. Perl J, Daskin MS (1985) A Warehouse Location Routing Model. Transp Res B 19:381–396
52. Prins C, Prodhon C, Calvo RW (2006) Solving the Capacitated Location-Routing Problem by a GRASP Complemented by a Learning Process and a Path Relinking, 4OR 4:221–238
53. Prins C, Prodhon C, Calvo RW (2006) A Memetic Algorithm with Population Management (MA|PM) for the Capacitated Location-Routing Problem. Evol Comput Combinatorial Optim, LNCS 3906:183–194
54. Reeves CR (1995) Genetic Algorithms. In: Reeves CR (eds) Modern Heuristic Techniques for Combinatorial Problems. McGraw-Hill, London, pp 151–196
55. Reeves CR (2003) Genetic Algorithms. In: Glover F, Kochenberger GA (eds) Handbooks of Metaheuristics. Kluwer, Dordrecht, pp 55–82
56. Resende MGC, Ribeiro CC (2003) Greedy Randomized Adaptive Search Procedures. In: Glover F, Kochenberger GA (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 219–249
57. Russell R, Chiang WC, Zepeda D (2006) Integrating Multi-Product Production and Distribution in Newspaper Logistics. Comput Oper Res 35(5): 1576–1588
58. Simchi-Levi D, Berman O (1988) A Heuristic Algorithm for the Traveling Salesman Location Problem on Networks. Eur J Oper Res 36:478–484
59. Srisvastava R (1993) Alternate Solution Procedures for the Location-Routing Problem. Omega 21(4):497–506
60. Srivastava R, Benton WC (1990) The Location-Routing Problem: Consideration in Physical Distribution System Design. Comput Oper Res 6:427–435
61. Stowers CL, Palekar US (1993) Location Models with Routing Considerations for a Single Obnoxious Facility. Transp Sci 27(4):350–362
62. Tuzun D, Burke LI (1999) A Two-Phase Tabu Search Approach to the Location Routing Problem. Eur J Oper Res 116:87–99
63. Watson-Gandy CTD, Dohrn PJ (1973) Depot Location with Van Salesman – A Practical Approach. Omega 1:321–329

64. Wu TH, Low C, Bai JW (2002) Heuristic Solutions to Multi-Depot Location-Routing Problems. Comput Oper Res 29:1393–1415
65. Zografos KG, Samara S (1989) Combined Location-Routing Model for Hazardous Waste Transportation and Disposal. Transp Res Record 1245:52–59

Logconcave Measures, Logconvexity

ANDRÁS PRÉKOPA
RUTCOR, Rutgers Center for Operations Research, Piscataway, USA

MSC2000: 90C15

Article Outline

Keywords
See also
References

Keywords

Logconcave function; Logconcave measure; Logconvex function; Logconvex measure; α-concave function; α-concave measure; Quasiconcave function; Quasiconcave measure

A nonnegative function $f\colon \mathbf{R}^n \to \mathbf{R}^1$ is called a *logconcave* (point) function if for every $\mathbf{x}, \mathbf{y} \in \mathbf{R}^n$ and $0 < \lambda < 1$ we have the inequality

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \geq \left[f(\mathbf{x})\right]^{\lambda}\left[f(\mathbf{y})\right]^{1-\lambda}.$$

A probability measure P defined on the Borel sets of $\mathbf{R}^n$ is called *logconcave* if for any Borel sets $A, B \subset \mathbf{R}^n$ and $0 < \lambda < 1$ we have the inequality

$$\mathsf{P}(\lambda A + (1-\lambda)B) \geq \left[\mathsf{P}(A)\right]^{\lambda}\left[\mathsf{P}(B)\right]^{1-\lambda},$$

provided that $\lambda A + (1-\lambda)B$ is also a Borel set. If P is a logconcave measure in $\mathbf{R}^n$ and $A \subset \mathbf{R}^n$ is a convex set, then $\mathsf{P}(A + \mathbf{z})$ is a logconcave point function in $\mathbf{R}^n$. In particular, the probability distribution function $F(\mathbf{z}) = \mathsf{P}(\{\mathbf{x}\colon \mathbf{x} \leq \mathbf{z}\}) = \mathsf{P}(\{\mathbf{x}\colon \mathbf{x} \leq \mathbf{0}\} + \mathbf{z})$, of the probability measure P, is a logconcave point function. If $n = 1$, then also $1 - F(\mathbf{z})$ is logconcave.

The basic theorem concerning logconcave measures [5,6] states that if the probability measure P is generated by a logconcave probability density function f, i. e.,

$$\mathsf{P}(C) = \int_C f(\mathbf{x})\, d\mathbf{x}$$

for every Borel set $C \subset \mathbf{R}^n$, then P is a logconcave measure.

Examples for logconcave probability distributions are the multivariate normal, the uniform (on a convex set) and for special parameter values the Wishart, the beta, the univariate and some multivariate gamma distributions.

A closely related theorem [5] states that if $f\colon \mathbf{R}^{n+m} \to \mathbf{R}^1$ is a logconcave function, then

$$\int_{\mathbb{R}^m} f(\mathbf{x}, \mathbf{y})\, d\mathbf{y}$$

is a logconcave function in $\mathbf{R}^n$. This implies that the convolution of two logconcave functions is also logconcave [3,5].

Logconcave probability distributions play important role in probabilistic constrained stochastic programming problems. If the problem is:

$$\begin{cases} \min & \mathbf{c}^{\top}\mathbf{x} \\ \text{s.t.} & \mathsf{P}(T\mathbf{x} \geq \boldsymbol{\xi}) \geq p, \\ & A\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}, \end{cases}$$

and the random vector $\boldsymbol{\xi}$ has continuous distribution with logconcave probability density function, then the set of feasible solutions is convex (for more general results see [6]). On the other hand, if the problem is solved by a barrier function method with logarithmic penalty function, then the function, to be minimized in each step, is convex.

The basic theorem of logconcave measures has the following generalization [1,2]: If $-\infty \leq \alpha \leq \infty$, $0 < \lambda < 1$, and the probability density function $f\colon \mathbf{R}^n \to \mathbf{R}^1$ satisfies ($\mathbf{x}, \mathbf{y} \in \mathbf{R}^n$):

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \geq \left[\lambda f^{\alpha}(\mathbf{x}) + (1-\lambda)f^{\alpha}(\mathbf{y})\right]^{\frac{1}{\alpha}},$$

then for any Borel sets $A, B \subset \mathbf{R}^n$ such that $\lambda A + (1-\lambda)B$ is also a Borel set, we have

$$\mathsf{P}(\lambda A + (1-\lambda)B) \qquad (1)$$

$$\geq \left\{\lambda \left[\mathsf{P}(A)\right]^{\gamma} + (1-\lambda)\left[\mathsf{P}(B)\right]^{\gamma}\right\}^{\frac{1}{\gamma}}, \qquad (2)$$

where $\gamma = \alpha/(1 + n\alpha)$. The cases $\alpha, \gamma = -\infty, 0, \infty$ are interpreted by continuity. Logconcavity corresponds to the case $\alpha = \gamma = 0$. If f, P satisfy the above inequalities, then f is called an *α-concave function* and P a *γ-concave probability measure*. If $\alpha, \gamma = -\infty$, then f and P are called *quasiconcave*.

A nonnegative function $f: \mathbf{R}^n \to \mathbf{R}^1$ is called *logconvex* in the convex set $D \in \mathbf{R}^n$ if for every $\mathbf{x}, \mathbf{y} \in D$ and $0 < \lambda < 1$ we have the inequality

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \left[f(\mathbf{x})\right]^{\lambda}\left[f(\mathbf{y})\right]^{1-\lambda}.$$

Similarly, the probability measure P defined on the Borel subsets of the convex set $D \subset \mathbf{R}^n$ is called *logconvex* if for any Borel sets $A, B \subset D$ we have the inequality

$$\mathsf{P}(\lambda A + (1-\lambda)B) \leq \left[\mathsf{P}(A)\right]^{\lambda}\left[\mathsf{P}(B)\right]^{1-\lambda}.$$

It follows, by Hölder's inequality, that the sum of logconvex functions is also logconvex. This fact, in turn, implies that if f is logconvex in D, then the function of the variable $\mathbf{t} \in \mathbf{R}^n$

$$g(\mathbf{t}) = \int_{C+\mathbf{t}} f(\mathbf{x})\, d\mathbf{x}$$

is logconvex for any fixed Borel set $C \subset \mathbf{R}^n$ in the sense that $g(\lambda\mathbf{t}_1 + (1-\lambda)\,\mathbf{t}_2) \leq [g(\mathbf{t}_1)]^{\lambda}\,[g(\mathbf{t}_2)]^{1-\lambda}$ provided that $C + \mathbf{t}_1 \subset D$, $C + \mathbf{t}_2 \subset D$ and $0 < \lambda < 1$.

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Borell C (1975) Convex set functions in d-space. Periodica Math Hungarica 6:111–136
2. Brascamp HJ, Lieb EH (1976) On extensions of the Brunn–Minkowski and Prékopa–Leindler theorems, including inequalities for log-concave functions, and with an application to the diffusion equations. J Funct Anal 22:366–389
3. Davidovich YS, Korenblum BL, Hacet BI (1969) A property of logarithmically concave functions. Soviet Math Dokl 10:477–480
4. Prékopa A (1971) Logarithmic concave measures with applications to stochastic programming. Acta Sci Math (Szeged) 32:301–316
5. Prékopa A (1973) On logarithmic concave measures and functions. Acta Sci Math (Szeged) 34:335–343
6. Prékopa A (1995) Stochastic programming. Kluwer, Dordrecht

Logconcavity of Discrete Distributions

ANDRÁS PRÉKOPA
RUTCOR, Rutgers Center for Operations Research, Piscataway, USA

MSC2000: 90C15

Article Outline

Keywords
See also
References

Keywords

Discrete logconcave distributions; Poisson distribution; Binomial distribution; Hypergeometric distribution; Geometric distribution; Trinomial distribution

The univariate discrete probability distribution $\{p_k: k \in \mathbf{Z}\}$ is called *logconcave* if for every k we have the inequality $p_k^2 \geq p_{k-1}\, p_{k+1}$. This inequality implies that if $k = \lambda i + (1-\lambda)j$, where i, j, k are integers and $0 < \lambda < 1$, then we have the inequality $p_k \geq p_i^{\lambda}\, p_j^{(1-\lambda)}$. Examples are the binomial, Poisson, hypergeometric, geometric distributions.

A classical theorem by M. Fekete [3] states that the convolution of two logconcave univariate discrete distributions is also logconcave.

The multivariate discrete logconcavity [2] is not a direct generalization of its univariate counterpart. The discrete probability distribution $\{\mathsf{P}(\mathbf{x}): \mathbf{x} \in \mathbf{Z}^m\}$ is said to be *logconcave* if there exists a convex function $g: \mathbf{R}^m \to \mathbf{R}$ such that

$$-\log \mathsf{P}(\mathbf{x}) = g(\mathbf{x}) \quad \text{for } \mathbf{x} \in \mathbb{Z}^m.$$

If $\mathsf{P}(\mathbf{x}) = 0$, then by definition $-\log \mathsf{P}(\mathbf{x}) = +\infty$.

The convolution theorem, mentioned above in connection with logconcave univariate distributions, does not carry over to the multivariate case. We know, however, that the trinomial distribution:

$$\mathsf{P}(k_1, k_2) = \frac{n!}{k_1!k_2!(n-k_1-k_2)!} \times p_1^{k_1} p_2^{k_2} (1-p_1-p_2)^{n-k_1-k_2}$$

is logconcave and the convolution of trinomial distributions is also logconcave [5]. For the use of discrete logconcavity in stochastic programming consult [6].

Other definitions and results, concerning multivariate discrete logconcavity, can be found in [1,4].

See also

- Approximation of Extremum Problems with Probability Functionals
- Approximation of Multivariate Probability Integrals
- Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- Extremum Problems with Probability Functions: Kernel Type Solution Methods
- General Moment Optimization Problems
- Logconcave Measures, Logconvexity
- L-shaped Method for Two-stage Stochastic Programs with Recourse
- Multistage Stochastic Programming: Barycentric Approximation
- Preprocessing in Stochastic Programming
- Probabilistic Constrained Linear Programming: Duality Theory
- Probabilistic Constrained Problems: Convexity Theory
- Simple Recourse Problem: Dual Method
- Simple Recourse Problem: Primal Method
- Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- Static Stochastic Programming Models
- Static Stochastic Programming Models: Conditional Expectations
- Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- Stochastic Integer Programs
- Stochastic Linear Programming: Decomposition and Cutting Planes
- Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- Stochastic Network Problems: Massively Parallel Solution
- Stochastic Programming: Minimax Approach
- Stochastic Programming Models: Random Objective
- Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- Stochastic Programming with Simple Integer Recourse

- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Bapat RB (1988) Discrete multivariate distributions and generalized log-concavity. Sankhyā Ser. A 50:98–100
2. Barndorff-Nielsen O (1978) Information and exponential families in statistical theroy. Wiley, New York
3. Fekete M, Polya G (1912) Uber ein Problem von Laguerre. Rend Circ Mat Palermo 23:89–120
4. Karlin S, Rinott Y (1981) Entropy inequalities for classes of probability distributions II. The multivariate case. Adv Appl Probab 13:325–351
5. Pedersen JG (1975) On strong unimodality and Mancillarity with applications to contingency tables. Scandinavian J Statist 2:127–137
6. Prékopa A (1995) Stochastic programming. Kluwer, Dordrecht

Logic-Based Outer Approximation

IGNACIO E. GROSSMANN
Department of Chemical Engineering,
Carnegie Mellon University, Pittsburgh, USA

Article Outline

Keywords
Introduction
NLP and Master Subproblems
Steps of Algorithm
Example
References

Keywords

Generalized Disjunctive Programming; Disjunctive Programming; Mixed-Integer Programming; Outer-Approximation Method; Logic-based Optimization

Introduction

Turkay and Grossmann [7] proposed a logic version of the outer-approximation algorithm for MINLP by Duran and Grossmann [2] for solving a special class of generalized disjunctive programming (GDP) problems. The problem arises in the optimization of process networks and involves two-term disjunctions in which the first term is activated when a unit or node is selected, while the second term enforces zero values to a subset of the continuous variables. The specific form of the GDP problem is as follows:

$$
\begin{aligned}
&\min Z = \sum_{k \in K} c_k + f(x) \\
&s.t. \quad r(x) \le 0 \\
&\begin{bmatrix} Y_k \\ g_k(x) \le 0 \\ c_k = \gamma_k \end{bmatrix} \vee \begin{bmatrix} \neg Y_k \\ B^k x = 0 \\ c_k = 0 \end{bmatrix} \quad k \in K \qquad \text{(GDP)} \\
&\Omega(Y) = \text{True} \\
&x \in R^n, \quad c \in R^m, \quad Y \in \{\text{true,false}\}^m ,
\end{aligned}
$$

where Y_k are the Boolean variables that decide whether the first term or second term in a disjunction $k \in K$ is true or false, and x are the continuous variables. The objective function involves the term $f(x)$ for the continuous variables and the charges c_k that depend on the discrete choices in each disjunction $k \in K$. The constraints $r(x) \le 0$ must hold regardless of the discrete choices. In contrast, $g_k(x) \le 0$ are conditional constraints that must hold when Y_k is true in the kth disjunction; otherwise $(\neg Y_i)$ a subset of the x variables is set to zero with the proper definition of the matrix B^i. In particular, we define $B^i = \left[b^{\mathrm{T}}\right]$ such that $b_j^{\mathrm{T}} = e^{\mathrm{T}}$ if $x_j = 0$, and $b_j^{\mathrm{T}} = 0^{\mathrm{T}}$ if $x_j \neq 0$. In this way only a subset of the variables x is forced to zero (typically flows). The cost variables c_k correspond to the fixed charges, and their value equals γ_k if the Boolean variable Y_k is true; otherwise they are zero. $\Omega(Y) = \text{True}$ are logical relations for the Boolean variables expressed as propositional logic. It is assumed for the derivation of basic methods that the functions are convex, although in practical applications these often correspond to nonconvex functions.

NLP and Master Subproblems

Following the original algorithm [2], the logic-based outer-approximation algorithm consists of solving NLP subproblems and disjunctive or MILP master problems.

As described in Turkay and Grossmann [7], for fixed values of the Boolean variables, Y_k = true and Y_k = false, the corresponding NLP subproblem is as follows:

$$\begin{aligned}
&\min Z = \sum_{k\in K} c_k + f(x)\\
&s.t. \quad r(x) \le 0\\
&\qquad \left.\begin{array}{l} g_k(x) \le 0 \\ c_k = \gamma_k \end{array}\right\} \text{for } Y_k = \text{true} \quad k \in K\\
&\qquad \left.\begin{array}{l} B^k x = 0 \\ c_k = 0 \end{array}\right\} \text{for } Y_k = \text{false} \quad k \in K\\
&\quad x \in R^n, \quad c_i \in R^m,
\end{aligned} \tag{NLPD}$$

Note that for every disjunction $k \in K$ only constraints corresponding to the Boolean variable Y_k that is true are imposed, thus leading to a reduction in the size of the problem. Also, fixed charges γ_k are only applied to these terms. Assuming that *NF* subproblems (NLPD) are solved in which sets of linearizations $l = 1, \ldots NF$ are generated for subsets of disjunction terms $L_k = \{l | Y_k^l = \text{true}\}$, one can define the following disjunctive OA master problem:

$$\begin{aligned}
&\text{Min} Z = \sum_k c_k + \alpha\\
&s.t. \quad \left.\begin{array}{l} \alpha \ge f(x^l) + \nabla f(x^l)^{\mathrm{T}}(x - x^l) \\ r(x^l) + \nabla r(x^l)^{\mathrm{T}}(x - x^l) \le 0 \end{array}\right\} l = 1, \ldots, L\\
&\left[\begin{array}{l} Y_k \\ g_k(x^l) + \nabla g_k(x^l)^{\mathrm{T}}(x - x^l) \le 0 \\ l \in L_k \\ c_k = \gamma_k \end{array}\right] \vee \left[\begin{array}{l} \neg Y_k \\ B^k x = 0 \\ c_k = 0 \end{array}\right]\\
&\quad k \in K\\
&\quad \Omega(Y) = \text{True}\\
&\alpha \in R, \quad x \in R^n, \quad c \in R^m, \quad Y \in \{\text{true, false}\}^m
\end{aligned} \tag{MGDP}$$

It should be noted that before applying the above master problem it is necessary to solve various subproblems (NLPD) so as to produce at least one linear approximation of each of the terms in the disjunctions. As shown by Turkay and Grossmann [7], selecting the smallest number of subproblems amounts to solving a set covering problem, which is of small size and easy to solve. In the context of a process flowsheet synthesis problem, another way of generating the linearizations in (MGDP) is by starting with an initial flowsheet and suboptimizing the remaining subsystems.

The above problem (MGDP) can be solved by the methods described by Beaumont [1], Raman and Grossmann [6], and Hooker [4]. Turkay and Grossmann [4] have shown that if the convex hull representation of the disjunctions is used in (MGDP), then converting the logic relations $\Omega(Y)$ into the inequalities $Ay \le a$ leads to the following MILP problem:

$$\begin{aligned}
&\text{Min} Z = \sum_k y_k y_k + \alpha\\
&s.t. \quad \left.\begin{array}{l} \alpha \ge f(x^l) + \nabla f(x^l)^{\mathrm{T}}(x - x^l) \\ r(x^l) + \nabla r(x^l)^{\mathrm{T}}(x - x^l) \le 0 \end{array}\right\} l = 1, \ldots L\\
&\nabla_{x_{Z_k}} g_k(x^l)^{\mathrm{T}} x_{Z_k} + \nabla_{x_{N_k}} g_k(x^l)^{\mathrm{T}} x_{N_k}^1\\
&\quad \le \left[-g_k(x^l) + \nabla_x g_k(x^l)^{\mathrm{T}} x^l\right] y_k\\
&l \in L_k, \quad k \in K\\
&x_{N_k} = x_{N_k}^1 + x_{N_k}^2\\
&0 \le x_{N_k}^1 \le x_{N_k}^U y_k\\
&0 \le x_{N_k}^2 \le x_{N_k}^U (1 - y_k)\\
&Ay \le a\\
&x \in R^n, \quad x_{N_k}^1 \ge 0, \quad x_{N_k}^2 \ge 0, \quad y \in \{0,1\}^m
\end{aligned} \tag{MIPDF}$$

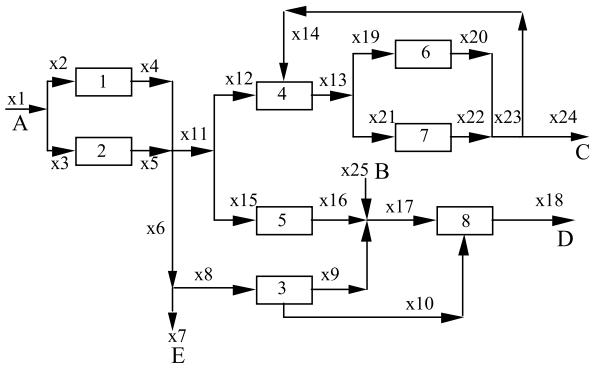
where the vector x is partitioned into the variables (x_{z_k}, x_{N_k}) for each disjunction k according to the definition of the matrix B^i (i. e., x_z refers to nonzero rows of this matrix). It is interesting to note that the logic-based outer-approximation algorithm represents a generalization of the modeling/decomposition strategy of Kocis and Grossmann [5] for the synthesis of process flowsheets.

Steps of Algorithm

Assuming feasible NLP subproblems, the steps of the proposed logic-based auter-approximation method are as follows:

Step 1: Model the problem in generalized disjunctive form as in (GDP).

Step 2: Identify the NF subproblems to be solved either from inspection or from set covering problems.



Logic-Based Outer Approximation, Figure 1
Process network example

Step 3: Solve NLP subproblems (NLPD) for the NF subproblems determined in step 2. The lowest-cost solution of these NLPs yields an upper bound, Z_U, for the problem.
Step 4: Linearize the objective function and constraints of the current NLP subproblem(s) and set up the MILP master problem (MIPDF). The solution of this problem gives the lower bound, Z_L, for the problem.
Step 5: If $|Z_U - Z_L| \leq \varepsilon$, where ε is a tolerance, then stop. The solution with the current Z_U is the optimal solution. Otherwise go to step 6.
Step 6: Solve NLP subproblem (NLPD) by fixing the Boolean variables predicted by the master problem. The objective function value of the solution is Z_{NLP}. If $Z_{NLP} < Z_U$, then set $Z_U = Z_{NLP}$.
Step 7: Compare the upper bound Z_U with the lower bound Z_L. If $|Z_U - Z_L| \leq \varepsilon$, then stop; the solution with the current Z_U is the optimal solution. Otherwise go to step 4.

It should be noted that one can also derive a logic-based version of Generalized Benders Decomposition as described in [7]. The logic outer-approximation algorithm described above has been implemented in the computer code LOGMIP by Vecchietti and Grossmann [8], which can be accessed from http://www.logmip.ceride.gov.ar

Example

Consider the following (GDP) problem from [7] that deals with a simplified version of the synthesis of a process network shown in Fig. 1.

The GDP model is as follows:

1. Objective function:

$$\begin{aligned}\min Z = {} & c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7 + c_8 + x_2 \\ & - 10x_3 + x_4 - 15x_5 - 40x_9 + 15x_{10} + 15x_{14} \\ & + 80x_{17} - 65x_{18} + 25x_{19} - 60x_{20} \\ & + 35x_{21} - 80x_{22} - 35x_{25} + 122\end{aligned}$$

2. Material balances at mixing/splitting points:

$$\begin{aligned} x_1 - x_2 - x_3 &= 0 \\ x_4 + x_5 - x_6 - x_{11} &= 0 \\ x_{13} - x_{19} - x_{21} &= 0 \\ x_{17} - x_9 - x_{16} - x_{25} &= 0 \\ x_{11} - x_{12} - x_{15} &= 0 \\ x_6 - x_7 - x_8 &= 0 \\ x_{23} - x_{20} - x_{22} &= 0 \\ x_{23} - x_{14} - x_{24} &= 0 \end{aligned}$$

3. Specifications on the flows:

$$\begin{aligned} x_{10} - 0.8x_{17} &\leq 0 \\ x_{10} - 0.4x_{17} &\geq 0 \\ x_{12} - 5x_{14} &\leq 0 \\ x_{12} - 2x_{14} &\geq 0 \end{aligned}$$

4. Disjunctions:

$$\text{Unit 1:} \begin{bmatrix} Y_1 \\ \exp(x_4) - 1 - x_2 \leq 0 \\ c_1 = 5 \end{bmatrix} \vee \begin{bmatrix} \neg Y_1 \\ x_2 = x_4 = 0 \\ c_1 = 0 \end{bmatrix}$$

$$\text{Unit 2:} \begin{bmatrix} Y_2 \\ \exp(x_5/1.2) - 1 - x_3 \leq 0 \\ c_2 = 8 \end{bmatrix} \vee \begin{bmatrix} \neg Y_2 \\ x_4 = x_3 = 0 \\ c_2 = 0 \end{bmatrix}$$

$$\text{Unit 3:} \begin{bmatrix} Y_3 \\ 1.5x_9 - x_8 + x_{10} = 0 \\ c_3 = 6 \end{bmatrix} \vee \begin{bmatrix} \neg Y_3 \\ x_8 = x_9 = x_{10} = 0 \\ c_3 = 0 \end{bmatrix}$$

$$\text{Unit 4:} \begin{bmatrix} Y_4 \\ 1.5(x_{12}+x_{14})-x_{13}=0 \\ c_4=10 \end{bmatrix} \vee \begin{bmatrix} \neg Y_4 \\ x_{12}=x_{13}=x_{14}=0 \\ c_4=0 \end{bmatrix}$$

$$\text{Unit 5:} \begin{bmatrix} Y_5 \\ x_{15}-2x_{16}=0 \\ c_5=6 \end{bmatrix} \vee \begin{bmatrix} \neg Y_5 \\ x_{15}=x_{16}=0 \\ c_5=0 \end{bmatrix}$$

$$\text{Unit 6:} \begin{bmatrix} Y_6 \\ \exp(x_{20}/1.5)-1-x_{19}\le 0 \\ c_6=7 \end{bmatrix} \vee \begin{bmatrix} \neg Y_6 \\ x_{19}=x_{20}=0 \\ c_6=0 \end{bmatrix}$$

$$\text{Unit 7:} \begin{bmatrix} Y_7 \\ \exp(x_{22})-1-x_{21}\le 0 \\ c_7=4 \end{bmatrix} \vee \begin{bmatrix} \neg Y_7 \\ x_{21}=x_{22}=0 \\ c_7=0 \end{bmatrix}$$

$$\text{Unit 8:} \begin{bmatrix} Y_8 \\ \exp(x_{18})-1-x_{10}-x_{17}\le 0 \\ c_8=5 \end{bmatrix} \vee \begin{bmatrix} \neg Y_8 \\ x_{10}=x_{17}=x_{18}=0 \\ c_8=0 \end{bmatrix}$$

5. Propositional Logic $[\Omega = (Y_i)]$:

$$Y_1 \Rightarrow Y_3 \vee Y_4 \vee Y_5$$
$$Y_2 \Rightarrow Y_{13} \vee Y_4 \vee Y_5$$
$$Y_3 \Rightarrow Y_1 \vee Y_2, \quad Y_3 \Rightarrow Y_8$$
$$Y_4 \Rightarrow Y_1 \vee Y_2, \quad Y_4 \Rightarrow Y_6 \vee Y_7$$
$$Y_5 \Rightarrow Y_1 \vee Y_2, \quad Y_5 \Rightarrow Y_8$$
$$Y_6 \Rightarrow Y_4$$
$$Y_7 \Rightarrow Y_4$$
$$Y_8 \Rightarrow Y_3 \vee Y_5 \vee (\neg Y_3 \wedge \neg Y_5)$$

Logic-Based Outer Approximation, Table 1
Progress of iterations

Subproblem	Objective value
NLPD1	73.277
NLPD2	103.584
NLPD3	113.789
MGDP	67.948
NLPD4	68.009

6. Specifications:

$$Y_1 \veebar Y_2$$
$$Y_4 \veebar Y_5$$
$$Y_6 \veebar Y_7$$

7. Variables:

$$x_j, \quad c_i \ge 0, \quad Y_i = \{\text{True,False}\}$$
$$i = 1, 2, \ldots, 8, \; j = 1, 2, \ldots, 25$$

Applying LOGMIP to solve this problem, and starting with three NLP subproblems at

$$\text{NLPD1}: Y_2 = Y_3 = Y_4 = Y_5 = Y_8 = \text{True}$$
$$\text{NLPD2}: Y_1 = Y_3 = Y_4 = Y_7 = Y_8 = \text{True}$$
$$\text{NLPD2}: Y_2 = Y_4 = Y_6 = Y_7 = \text{True}$$

the predicted optimum solution is given by $Z = 68.009$. Table 1 shows the progress of the iterations.

References

1. Beaumont N (1991) An Algorithm for Disjunctive Programs. Eur J Oper Res 48:362–371
2. Duran MA, Grossmann IE (1986) An Outer-Approximation Algorithm for a Class of Mixed-integer Nonlinear Programs. Math Program 36:307
3. Geoffrion AM (1972) Generalized Benders Decomposition. J Optim Theory Appl 10(4):237-260
4. Hooker JN (1999) Logic-Based Methods for Optimization. Wiley, New York
5. Kocis GR, Grossmann IE (1989) A Modeling and Decomposition Strategy for the MINLP Optimization of Process Flowsheets. Comput Chem Eng 13:797
6. Raman R, Grossmann IE (1994) Modelling and Computational Techniques for Logic Based Integer Programming. Comput Chem Eng 18(7):563–578
7. Turkay M, Grossmann IE (1996) Logic-based MINLP Algorithms for the Optimal Synthesis of Process Networks. Comput Chem Eng 20(8):959–978
8. Vecchietti A, Grossmann IE (1999) LOGMIP: A Disjunctive 0-1 Nonlinear Optimizer for Process Systems Models. Comput Chem Eng 23:555–565

Lovász Number

STANISLAV BUSYGIN
Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA

MSC2000: 05C69, 05C15, 05C17, 05C35, 90C35, 90C22

Article Outline

Synonyms
Introduction
Formulation
Lovász Number as an Upper Bound of Shannon Capacity
The Sandwich Theorem
Lovász Number as a Dual Bound of Quadratic Maximization
Applications
Perfect Graphs
Improving Upper Bounds for Independence Number
See also
References

Synonyms

ϑ-function

Introduction

Let $G(V,E)$ be a simple undirected graph, $V = \{1,2,\ldots,n\}$. The *adjacency matrix* of G is a matrix $A_G = (a_{ij})_{n\times n}$, where $a_{ij} = 1$ if $(i,j) \in E$ and $a_{ij} = 0$ otherwise. The set of vertices *adjacent* to a vertex $i \in V$ will be denoted by $N(i) = \{j \in V : (i,j) \in E\}$ and called the *neighborhood* of the vertex i. We will also consider the *complementary graph* $\bar{G}(V,\bar{E})$ having the same set of vertices V, but an edge $(i,j) \in \bar{E}$ if and only if i and j are not adjacent in G.

An *independent set* S is a subset of V such that no two vertices of S are adjacent, i. e., $\forall i \in S N(i) \cap S = \varnothing$. The set S is called a *maximal* independent set if any vertex $i \in V \setminus S$ has at least one adjacent vertex in S, i. e., $\forall i \in V \setminus S\ N(i) \cap S \neq \varnothing$. Finally, the set S is called a *maximum* independent set if it has the largest cardinality among all independent sets of the graph. This cardinality will be denoted by $\alpha(G)$ and called the *independence* (or *stability*) *number* of the graph G.

In addition to the maximum cardinality stable sets, we will consider the *maximum weight independent sets*. Let there be a given vector $w = (w_1, w_2, \ldots, w_n)^{\mathrm{T}}$ of nonnegative *vertex weights*. A maximum weight independent set is such an independent set $S \subseteq V$ that has the largest weight $\alpha(G,w) = \max_S \sum_{i\in S} w_i$.

Similarly, a *clique* Q of the graph G is a subset of V such that any two vertices in it are adjacent, i. e., $\forall i \in Q\ N(i) \cap Q = Q \setminus \{i\}$. The clique Q is called *maximal* if for any vertex $i \in V \setminus Q$ there is at least one vertex in Q non-adjacent to i, i. e., $\forall i \in V \setminus Q\ N(i) \cap Q \neq Q$. If Q has the largest cardinality among all cliques of the graph, it is called a *maximum* clique. The cardinality of a maximum clique will be denoted by $\omega(G)$ and called the *clique number* of the graph G. A *maximum weight clique* is a clique having the largest weight $\omega(G,w) = \max_Q \sum_{i\in Q} w_i$.

It is easy to see that independent sets of the graph G correspond to cliques of $\bar{G}$, and vise versa.

We will denote by $\chi(G)$ the *chromatic number* of the graph G (i. e. the minimum number of colors to which the graph vertices can be colored without using one color for any two adjacent vertices.) The number $\chi(\bar{G})$, giving the minimum number of cliques of G to which the vertex set V can be partitioned, will be also denoted by $\bar{\chi}(G)$ and called the *clique partition number* of the graph G.

Next, for two graphs $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$ we define their *strong product* $G_1 \cdot G_2$ as the graph, whose vertex set is the Cartesian product $V_1 \times V_2$ and in which a vertex (i,j) is adjacent to a vertex (i',j') if and only if $(i,i') \in E_1$ and $(j,j') \in E_2$. The strong product of k copies of G will be denoted by G^k.

Formulation

Lovász Number as an Upper Bound of Shannon Capacity

Let us consider the set $V = \{1,2,\ldots,n\}$ to be an alphabet in which the adjacency means that the two letters can be confused. Then any set of one-letter messages that cannot be confused with each other corresponds to an independent set of the graph and vise versa. Furthermore, the maximum number of one-letter messages that cannot be confused with each other is equal to $\alpha(G)$, and the maximum number of k-letter messages that cannot be confused with each other is equal to $\alpha(G^k)$. It is easy to see that there are at least $\alpha(G)^k$ k-letter messages that cannot be confused with each other,

so $\alpha(G^k) \geq \alpha(G)^k$. So,

$$\Theta(G) = \sup_k \sqrt[k]{\alpha(G^k)} = \lim_{k\to\infty} \sqrt[k]{\alpha(G^k)} \geq \alpha(G)\,. \quad (1)$$

The value $\Theta(G)$ is called the *Shannon zero-error capacity* of the graph G [14]. Generally, it is extremely hard to compute, and nowadays $\Theta(G)$ is not even known for the graph C_7 (cycle of 7 vertices).

Thus, the independence number $\alpha(G)$ gives a *lower* bound on $\Theta(G)$. In 1979, L. Lovász defined a new nontrivial *upper* bound on the Shannon zero-error capacity of a graph in his seminal paper [11]. This function was named later *Lovász number* (or ϑ*-function*) of a graph.

First, define an *orthonormal representation* of the graph G as a system $(u_1, u_2, \ldots, u_n)$ of unit vectors in a Euclidean space such that whenever two vertices i and j are not adjacent, the vector u_i is orthogonal to the vector u_j. It is easy to see that such systems of vectors do exist, e. g., any n orthonormal vectors from the space $\mathbb{R}^n$. The ϑ-function is defined as the following minimax value:

$$\vartheta(G) = \min_{\{c,(u_i)\}} \max_{i\in V} \frac{1}{(c^{\mathrm{T}}u_i)^2}\,, \quad (2)$$

where c ranges over unit vectors of the same dimensionality that the vectors u_i are. The vector c was called by Lovász the *handle* of the representation.

It can be shown that for a strong product of graphs, $\vartheta(G\cdot H) \leq \vartheta(G)\vartheta(H)$. To show that $\alpha(G) \leq \vartheta(G)$ one needs to observe that if S is a maximum independent set of G, then $1 = c^2 \geq \sum_{i\in S}(c^{\mathrm{T}}u_i)^2 \geq \alpha(G)/\vartheta(G)$. From here it is obvious that $\Theta(G) \leq \vartheta(G)$ as $\alpha(G^k) \leq \vartheta(G^k) \leq \vartheta(G)^k$.

Similarly, we introduce the weighted ϑ-function:

$$\vartheta(G,w) = \min_{\{c,(u_i)\}} \max_{i\in V} \frac{w_i}{(c^{\mathrm{T}}u_i)^2}\,, \quad (3)$$

which gives an upper bound for $\alpha(G,w) \leq \vartheta(G,w)$.

In contrast to $\Theta(G)$ and $\alpha(G,w)$, which are hard to compute, $\vartheta(G,w)$ can be computed with an arbitrary precision in a polynomial time by either the ellipsoid method or an interior point method due to its semidefinite programming formulation considered below (see also [7,9,13]). This makes ϑ-function attractive for estimating these intractable numbers.

The Sandwich Theorem

Other equivalent formulations implying a number of interesting properties of $\vartheta(G)$ were established in [7] (see also the extensive survey [9]). To introduce them, let us define three specific convex sets in $\mathbb{R}^n$ associated with the graph:

$$\begin{aligned}
\mathcal{STAB}(G) &= \mathrm{hull}(\{x\in\{0,1\}^n | x_j + x_k \leq 1,\\
&\qquad \forall (j,k)\in E\}),\\
\mathcal{TH}(G) &= \{x\geq 0 | \sum_{j\in V}(c^{\mathrm{T}}u_j)^2 x_j \leq 1,\\
&\qquad \forall \text{ ort. lab. } (u_j) \text{ of } G,\ \|c\| = 1\},\\
\mathcal{QSTAB}(G) &= \{x\geq 0 \mid \sum_{j\in Q} x_j \leq 1,\\
&\qquad \forall \text{ cliques } Q \text{ of } G\}\,.
\end{aligned}$$

Let $x^S \in \{0,1\}^n$ be the *incidence vector* of an independent set S, that is, $x_i^S = 1$ if $i\in S$, and $x_i = 0$ otherwise. Then, obviously, for any orthonormal representation (u_i) and a unit vector c,

$$\sum_{j\in V}(c^{\mathrm{T}}u_j)^2 x_j^S = \sum_{j\in S}(c^{\mathrm{T}}u_j)^2 \leq 1\,.$$

So, any $x\in\mathcal{STAB}(G)$ satisfy the constraints of $\mathcal{TH}(G)$. Let Q be any clique of G. Then we can construct an orthonormal representation as follows. Let all vectors $(u_i)_{i\in V\setminus Q}$ be mutually orthogonal, and also each of them be orthogonal to another unit vector c. We set all $(u_i)_{i\in Q}$ to be equal to c. If we consider the constraint $\sum_{j\in V}(c^{\mathrm{T}}u_j)^2 x_j \leq 1$ over only such orthonormal representations, we obtain the clique constraints defining the set $\mathcal{QSTAB}(G)$. Hence, we have

$$\mathcal{STAB}(G) \subseteq \mathcal{TH}(G) \subseteq \mathcal{QSTAB}(G)\,. \quad (4)$$

Obviously,

$$\alpha(G,w) = \max_x\{w^{\mathrm{T}}x \mid x\in\mathcal{STAB}(G)\}. \quad (5)$$

Let us also denote

$$\kappa(G,w) = \max_x\{w^{\mathrm{T}}x \mid x\in\mathcal{QSTAB}(G)\}. \quad (6)$$

We will prove that

$$\vartheta(G,w) = \max_x\{w^{\mathrm{T}}x \mid x\in\mathcal{TH}(G)\} \quad (7)$$

and henceforth conclude that

$$\alpha(G,w) \le \vartheta(G,w) \le \kappa(G,w). \tag{8}$$

The double inequality (8) constitutes the famous *sandwich theorem*.

Let us denote by S_n the set of all $n \times n$ symmetric matrices, and by S_n^+ the set of all *positive semidefinite* $n \times n$ matrices:

$$S_n^+ = \{A \in S_n |\, x^{\mathrm{T}} A x \ge 0 \ \forall x \in \mathbb{R}^n\}.$$

We also denote by $z = (\sqrt{w_1}, \sqrt{w_2}, \ldots, \sqrt{w_n})^{\mathrm{T}}$ the vector of square roots of the vertex weights. Consider the following functions of the graph and vertex weights:

$$\begin{aligned}\vartheta_2(G,w) &= \min_{A \in S_n} \lambda_{\max}(A),\\ \text{s.t. } a_{ij} &= \sqrt{w_i w_j}, \quad \forall (i,j) \notin E,\end{aligned}$$

where $\lambda_{\max}(A)$ denotes the largest eigenvalue of A;

$$\begin{aligned}\vartheta_3(G,w) &= \max_{X \in S_n^+} z^{\mathrm{T}} X z,\\ \text{s.t. } x_{ij} &= 0, \quad \forall (i,j) \in E, \quad \mathrm{tr}(X) = 1,\end{aligned}$$

where $\mathrm{tr}(X) = \sum_{i=1}^n x_{ii}$ denotes the *trace* of the matrix X;

$$\vartheta_4(G,w) = \max_{\{d,(v_i)\}} \sum_{i \in V} (d^{\mathrm{T}} v_i)^2 w_i,$$

where $(v_i)_{i \in V}$ range over all orthonormal representations of the complementary graph $\bar{G}$ and $\|d\| = 1$.

Theorem 1

$$\begin{aligned}\vartheta(G,w) &= \vartheta_2(G,w) = \vartheta_3(G,w) = \vartheta_4(G,w)\\ &= \max_x \{w^T x \mid x \in \mathcal{TH}(G)\}.\end{aligned}$$

Proof First we show that $\vartheta(G,w) \le \vartheta_2(G,w)$. Consider a matrix $A \in S_n$ such that $a_{ij} = \sqrt{w_i w_j}$, $\forall (i,j) \notin E$, and let $t = \lambda_{\max}(A)$. Then $tI - A \in S_n^+$, and hence there exists $X \in \mathbb{R}^{n \times n}$ such that $tI - A = X^{\mathrm{T}} X$. Let $x_i \in \mathbb{R}^n$ be the i-th column of X. Then

$$x_i^{\mathrm{T}} x_i = t - w_i, \quad \forall i \in V$$

and

$$x_i^{\mathrm{T}} x_j^{\mathrm{T}} = -\sqrt{w_i w_j}, \quad \forall i, j \text{ nonadjacent in } G.$$

Note that $\mathrm{rank}(X) < n$ since the matrix $tI - A$ has a zero eigenvalue. This implies that there exists a unit vector $c \in \mathbb{R}^n$ orthogonal to all x_i, $i \in V$. Consider the vectors

$$u_i = (\sqrt{w_i} c + x_i)/\sqrt{t}, \quad i \in V.$$

It is easy to see that

$$u_i^{\mathrm{T}} u_i = \frac{(w_i c^{\mathrm{T}} c + x_i^{\mathrm{T}} x_i)}{t} = 1$$

and for any two nonadjacent vertices $i, j \in V$,

$$u_i^{\mathrm{T}} u_j = \frac{(\sqrt{w_i w_j} c^{\mathrm{T}} c + x_i^{\mathrm{T}} x_j)}{t} = 0.$$

Hence, the vectors (u_i) form an orthonormal representation of G and

$$\vartheta(G,w) \le \max_{i \in V} \frac{w_i}{(c^{\mathrm{T}} u_i)^2} = \max_{i \in V} \frac{w_i}{w_i/t} = t = \lambda_{\max}(A).$$

Now, we show that $\vartheta_2(G,w) \le \vartheta_3(G,w)$. We have $z^{\mathrm{T}} X z \le \vartheta_3 \cdot \mathrm{tr}(X)$ for any $X \in S_n^+$ such that $x_{ij} = 0\ \forall (i,j) \in E$. This inequality is equivalent to $(W - \vartheta_3 I) \bullet X \le 0$, where $W = (\sqrt{w_i w_j})_{n \times n}$ and "$\bullet$" denotes the Euclidian inner product in $\mathbb{R}^{n \times n}$, i. e., $A \bullet B = \sum_{i,j} a_{ij} b_{ij}$. From here it can be inferred that the matrix $\vartheta_3 I - W$ is a sum of some positive semidefinite matrix $D \in S_n^+$ and another symmetric matrix $A = (a_{ij}) \in S_n$ such that if $(i,j) \notin E$, then $a_{ij} = 0$. This implies that $\vartheta_3 I - W - A \in S_n^+$ and hence $\vartheta_3 \ge \lambda_{\max}(W + A) \ge \vartheta_2$.

Next, we show that $\vartheta_3(G,w) \le \vartheta_4(G,w)$. Let $X = (x_{ij}) \in \mathbb{R}^{n \times n}$ be an optimum matrix for the program defining ϑ_3. Since $X \in S_n^+$, there exists a matrix $Y \in \mathbb{R}^{n \times n}$ such that $X = Y^{\mathrm{T}} Y$. Let x_i denote the i-th column of X and y_i denote the i-th column of Y. Construct an orthonormal system of vectors $(u_i)_{i \in V}$ in $\mathbb{R}^n$ such that there is the vector $u_i = y_i / \|y_i\|$ whenever $y_i \ne 0$. Since $y_i^{\mathrm{T}} y_j = x_{ij} = 0 \forall (i,j) \in E$, the system (u_i) is an orthonormal representation of $\bar{G}$. Furthermore, $z^{\mathrm{T}} Y^{\mathrm{T}} Y z = z^{\mathrm{T}} X z = \vartheta_3$ and hence $d = Yz/\sqrt{\vartheta_3}$ is a unit vector. Whenever $y_i \ne 0$,

$$d^{\mathrm{T}} v_i = \frac{z^{\mathrm{T}} Y^{\mathrm{T}} y_i}{(\sqrt{\vartheta_3} \|y_i\|)} = \frac{z^{\mathrm{T}} x_i}{(\sqrt{\vartheta_3} \|y_i\|)}.$$

Thus, $\|y_i\| d^{\mathrm{T}} v_i = z^{\mathrm{T}} x_i / \sqrt{\vartheta_3}$, $\forall i \in V$, and

$$\begin{aligned}\sum_{i \in V} \|y_i\| \sqrt{w_i} d^{\mathrm{T}} v_i &= \frac{1}{\sqrt{\vartheta_3}} \sum_{i \in V} z^{\mathrm{T}} x_i \sqrt{w_i}\\ &= \frac{1}{\sqrt{\vartheta_3}} z^{\mathrm{T}} X z = \sqrt{\vartheta_3}.\end{aligned}$$

Using the Cauchy–Schwarz inequality,

$$\begin{aligned}\vartheta_3 &= \left(\sum_{i\in V}\|y_i\|\sqrt{w_i}d^{\mathrm{T}}v_i\right)^2\\ &\le \left(\sum_{i\in V}\|y_i\|^2\right)\left(\sum_{i\in V}w_i(d^{\mathrm{T}}v_i)^2\right)\\ &= \left(\sum_{i\in V}x_{ii}\right)\left(\sum_{i\in V}w_i(d^{\mathrm{T}}v_i)^2\right)\\ &= \sum_{i\in V}w_i(d^{\mathrm{T}}v_i)^2 \le \vartheta_4(G,w)\,.\end{aligned}$$

Next, we prove that $\vartheta_4(G,w) \le \max_x\{w^{\mathrm{T}}x \mid x \in \mathcal{TH}(G)\}$. Let $(v_i)_{i\in V}$ and d be correspondingly an optimum orthogonal representation of $\bar{G}$ and its handle for the program defining ϑ_4. We show that the vector $\left((d^{\mathrm{T}}v_i)^2\right)_{i\in V}$ belongs to $\mathcal{TH}(G)$. Consider some orthonormal representation of G, $(u_i)_{i\in V}$, $u_i \in \mathbb{R}^n$, and let $c \in \mathbb{R}^n$ be some unit vector. The matrices $u_iv_i^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ are mutually orthogonal and have the unit norm with respect to the inner product "$\bullet$", i. e.,

$$(u_iv_i^{\mathrm{T}})\bullet(u_jv_j^{\mathrm{T}}) = (u_i^{\mathrm{T}}u_j)(v_i^{\mathrm{T}}v_j) = \begin{cases}1, \text{if } i=j,\\ 0, \text{if } i\ne j.\end{cases}$$

Now we may conclude that

$$\begin{aligned}\sum_{i\in V}(c^{\mathrm{T}}u_i)^2(d^{\mathrm{T}}v_i)^2 &= \sum_{i\in V}((cd^{\mathrm{T}})\bullet(u_iv_i^{\mathrm{T}}))^2\\ &\le (cd^{\mathrm{T}})\bullet(cd^{\mathrm{T}}) = 1\,.\end{aligned}$$

Hence $\left((d^{\mathrm{T}}v_i)^2\right)_{i\in V} \in \mathcal{TH}(G)$ and

$$\begin{aligned}\vartheta_4(G,w) &= \sum_{i\in V}w_i(d^{\mathrm{T}}v_i)^2\\ &\le \max_x\{w^{\mathrm{T}}x \mid x\in\mathcal{TH}(G)\}\,.\end{aligned}$$

The final step is to show that $\max_x\{w^{\mathrm{T}}x \mid x \in \mathcal{TH}(G)\}$
$\le \vartheta(G,w)$. Let x^* be a vector maximizing w^Tx over $\mathcal{TH}(G)$. Choosing any orthonormal representation $(u_i)_{i\in V}$ and a unit vector c, we have

$$\begin{aligned}w^{\mathrm{T}}x^* &\le \left(\max_i\frac{w_i}{(c^{\mathrm{T}}u_i)^2}\right)\sum_{i\in V}(c^{\mathrm{T}}u_i)^2x_i^*\\ &\le \max_i\frac{w_i}{(c^{\mathrm{T}}u_i)^2} \le \vartheta(G,w)\,.\end{aligned}$$

The four inequalities established above can hold if and only if all the ϑ's are equal. QED. □

Let us denote by $\kappa(G)$ the value of $\kappa(G,w)$ when all vertex weights are 1's. It is easy to see that $\kappa(G) \le \bar{\chi}(G)$. Indeed, if we take a minimum clique partition $\{Q_1, Q_2, \ldots, Q_{\bar{\chi}}\}$ of G, then $\bar{\chi}(G)$ is equal to the optimum value of the program:

$$\begin{aligned}&\max_{x\in\mathbb{R}^n}\sum_{i\in V}x_i,\\ &\text{s.t.}\sum_{i\in Q_k}x_i \le 1, \quad k=1,2,\ldots,\bar{\chi}, \quad x\ge 0\,.\end{aligned}$$

This program has the same objective as the program for $\kappa(G)$, but its set of constraints is a subset of constraints of $\mathcal{QSTAB}(G)$. So, we may extend the sandwich inequality to

$$\alpha(G) \le \vartheta(G) \le \kappa(G) \le \bar{\chi}(G)\,.$$

Omitting $\kappa(G)$ and applying the inequalities to the complementary graph, we obtain

$$\omega(G) \le \vartheta(\bar{G}) \le \chi(G)\,, \tag{9}$$

which expresses another famous form of the sandwich theorem stating that a polynomial-time computable number $\vartheta(\bar{G})$ lies in between the two *NP*-hard numbers: the clique number and the chromatic number.

Lovász Number as a Dual Bound of Quadratic Maximization

Consider the following quadratic formulation of the maximum weight independent set problem:

$$\begin{aligned}\alpha(G,w) &= \max w^{\mathrm{T}}x\\ &\text{s.t. } x_ix_j = 0, \quad \forall(i,j)\in E,\\ &\quad x_i^2 - x_i = 0,\ i=1, \quad \ldots,n\,.\end{aligned} \tag{10}$$

It has been shown by N.Z. Shor that the optimal Lagrangian dual bound of program (10) is equal to $\vartheta(G,w)$ [15]. One can compute this bound for a maximization problem with a quadratic objective subject to a set of linear and quadratic constraints minimizing a convex non-differentiable function defined over a parametric (linearly dependent on Lagrangian multipliers) set of negative semidefinite symmetric matrices.

Indeed, the Lagrangian of program (10) is

$$L(x, \Lambda) = w^{\mathrm{T}}x + \sum_{i=1}^{n} \lambda_{ii}(x_i^2 - x_i) + \sum_{(i,j)\in E} \lambda_{ij} x_i x_j \,.$$

The considered optimization problem is equivalent to

$$\max_{x} \min_{\Lambda} L(x, \Lambda) \,,$$

while the optimal Lagrangian dual bound is derived as

$$\min_{\Lambda} \max_{x} L(x, \Lambda) \,.$$

From here it follows that in the dual problem Λ should always be chosen in such a way that the quadratic form of $L(x, \Lambda)$ is negative semidefinite with respect to x (otherwise the inner maximization with respect to x will deliver infinity), while minimization with respect to Λ turns out to be convex non-differentiable [15].

Applications

Perfect Graphs

A graph is called *perfect* if, for all its vertex-induced subgraphs, the clique number is equal to the chromatic number. In this case the inequalities (9) become the equalities:

$$\omega(G) = \vartheta(\bar{G}) = \chi(G) \,.$$

So, both the clique number and the chromatic number can be computed in a polynomial time by means of the ϑ-function. It is also easy to see how the ϑ-function can be used to actually find a maximum clique of a perfect graph. Indeed, a vertex i of a perfect graph G belongs to some maximum clique if and only if ϑ-function of the subgraph induced by the neighborhood $N(i)$ is equal to $\vartheta(G) - 1$. Hence, we can obtain a maximum clique of the graph successively selecting a vertex satisfying this condition and repeating the procedure with the subgraph induced by its neighborhood. Moreover, this simple algorithm can be improved [1,17]. Coloring a perfect graph can be also performed in a polynomial time (see, e. g., [7]).

A graph is perfect if and only if its complementary graph is perfect [7,10]. This means that for any vertex-induced subgraph of a perfect graph there is also the equality between the independence number and the clique partition number. The *strong perfect graph theorem*, proved recently [5], states that a graph is perfect if and only if it does not include an odd hole or an odd antihole as a vertex-induced subgraph. A polynomial-time algorithm for recognizing perfect graphs was also derived on the basis of the strong perfect graph theorem [4].

Improving Upper Bounds for Independence Number

It is worth to consider how well does $\vartheta(G)$ approximate the independence number $\alpha(G)$ for general graphs and whether this approximation can be improved without breaking the polynomial-time computability. It turns out that for random graphs $\vartheta(G)/\alpha(G)$ grows as $O(\sqrt{n}/\log n)$ [2,9]. So, $\vartheta(G)$ does not allow for a fixed approximation guarantee for $\alpha(G)$. Moreover, the maximum independent set problem is known to be hard to approximate (see, e. g., [8]). However, there is a number of approaches to formulate increasingly tight approximations of $\alpha(G)$ based on the ϑ-function. The first one is the "lift-and-project" method by Lovász and Shrijver [12]. The second approach is to express $\alpha(G)$ as a *copositive* program and to use its approximations via semidefinite programming [6]. Finally, we may try to improve the dual bound of (10) and make it closer to the optimum generating *superfluous* quadratic constraints [15,16].

In first two cases one obtains a sequence of semidefinite programs increasing in size, but having non-increasing optimum values, and at some point the optimum value becomes equal to $\alpha(G)$. It comes as no surprise that before achieving the value $\alpha(G)$, in the general case, the size of the program increases exponentially (otherwise it would imply $P = NP$). What is more surprising is that any provable polynomial-time improvement of $\vartheta(G)$ (i. e. a polynomial-time computable function of a graph having a value less than $\vartheta(G)$ whenever $\alpha(G) < \vartheta(G)$) would also imply $P = NP$ [3]. Hence, unless $P = NP$, neither method can deliver, in general case, a value closer to $\alpha(G)$ than $\vartheta(G)$ before the size of the program becomes exponential.

See also

► Copositive Programming

References

1. Alizadeh F (1991) A sublinear-time randomized parallel algorithm for the maximum clique problem in perfect

graphs. In: Proc 2nd ACM-SIAM SODA, San Francisco, CA, pp 188–194
2. Bollobás B (1985) Random graphs. Academic Press, London
3. Busygin S, Pasechnik DV (2006) On NP-hardness of the clique partition – independence number gap recognition and related problems. Discret Math 304(4):460–463
4. Chudnovsky M, Cornuéjols G, Liu X, Seymour P, Vušković K (2006) Recognizing Berge graphs. Combinatorica 25(2):143–186
5. Chudnovsky M, Robertson N, Seymour P, Thomas R (2006) The strong perfect graph theorem. Ann Math 164(1):51–229
6. de Klerk E, Pasechnik D (2002) Approximation of the stability number of a graph via copositive programming. SIAM J Optim 12(4):875–892
7. Grötschel M, Lovász L, Schrijver A (1988) Geometric algorithms and combinatorial optimization. Springer, Berlin
8. Khot S (2001) Improved inapproximability results for max-clique, chromatic number and approximate graph coloring. In: Proc 42nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS), pp 600–609
9. Knuth DE (1994) The sandwich theorem. Elec J Comb 1: 1–48
10. Lovász L (1972) Normal hypergraphs and the perfect graph conjecture. Discret Math 2:253–267
11. Lovász L (1979) On the Shannon capacity of a graph. IEEE Trans Inform Theory IT-25(1):1–7
12. Lovász L, Schrijver A (1991) Cones of matrices and set-functions and 0-1 optimization. SIAM J Optim 1(2):166–190
13. Nesterov Y, Nemirovskii A (1994) Interior point polynomial methods in convex programming. SIAM, Philadelphia
14. Shannon CE (1956) The zero-error capacity of a noisy channel. IRE Trans Inform Theory IT-2(3):8–19
15. Shor NZ (1998) Nondifferentiable optimization and polynomial problems. Kluwer, Dordrecht
16. Shor NZ, Stetsyuk PI (2002) Lagrangian bounds in multi-extremal polynomial and discrete optimization problems. J Glob Optim 23:1–41
17. Yildirim EA, Fan-Orzechowski X (2006) On extracting maximum stable sets in perfect graphs using Lovász's theta function. Comput Optim Appl 33(2–3):229–247

LP Strategy for Interval-Newton Method in Deterministic Global Optimization

YOUDONG LIN, MARK A. STADTHERR
Department of Chemical and Biomolecular Engineering, University of Notre Dame, Notre Dame, USA

Article Outline

Background
Methods
Interval-Newton
Solution of Linear Interval Equation Systems
LP Strategy for Interval-Newton Method
Cases
Problem 1
Problem 2
Problems 3 and 4
Problem 5
Conclusions
References

Background

The interval-Newton method is a tool for solving a system of nonlinear algebraic equations. It provides the capability to enclose *all* solutions of the equation system that occur within a specified search interval, and to do so with *mathematical and computational certainty*. In the context of optimization, it is generally applied to the deterministic global optimization of bound-constrained problems:

$$\min_{\boldsymbol{x}} \phi(\boldsymbol{x}) \tag{1}$$

$$\boldsymbol{x} \in \boldsymbol{X}^{(0)}. \tag{2}$$

The objective $\phi(\boldsymbol{x})$ is in general a nonconvex function that may have multiple local minima. The interval vector (box) $\boldsymbol{X}^{(0)}$ provides upper and lower bounds on each component of the decision-variable vector $\boldsymbol{x}$. It is assumed here that these bounds are sufficiently wide that the global minimum of $\phi(\boldsymbol{x})$ will occur in the interior of $\boldsymbol{X}^{(0)}$. This means that the stationarity condition $\nabla\phi(\boldsymbol{x}) = \boldsymbol{0}$ can be used in the search for the global minimum. The interval-Newton method can then be applied to enclose all stationary points, one of which is the global minimizer. If only the global minimizer is sought, then interval-Newton is typically combined with some branch-and-bound scheme, so that all stationary points need not actually be found. However, in other applications, such as transition state analysis [22,38] and computation of phase equilibrium [13,37], it may be useful to know all of the stationary points, and the interval-Newton approach provides this capability. For situations in which it is possible that the global minimum will lie on a boundary

of $\boldsymbol{X}^{(0)}$, then the "peeling" process described by Kearfott [19], in which interval-Newton is applied to each of the lower dimensional subspaces that constitute the boundary of $\boldsymbol{X}^{(0)}$, can be used. For more general constrained problems, the interval-Newton method can be applied to the solution of the Karush–Kuhn–Tucker (KKT) conditions or the Fritz-John conditions. A thorough discussion of the use of the interval-Newton approach in global optimization has been given by Hansen and Walster [11]. In recent years, this approach has been used in a number of applications, including computation of fluid phase equilibrium from activity coefficient models [27,36,37,40], cubic equation-of state models [5,12,13,14,40] and statistical associating fluid theory [39], computation of solid-fluid equilibrium [35,41], parameter estimation using standard least squares [7,25] and error-in-variables [8,9], calculation of adsorption in nanoscale pores from a density function theory model [26], transition state analysis [22] and determination of molecular structures [24].

A drawback to the interval-Newton approach, as well as to other approaches for deterministic global optimization, is the potentially high computational cost. One way to improve the efficiency of the interval-Newton method is to more tightly bound the solution set of the linear interval equation system that is at the core of this approach. In this article, we discuss the solution of this linear interval system and describe a bounding strategy [21,23] based on the use of linear programming (LP) techniques. Using this approach it is possible to exactly (within round out) determine the desired bounds on the solution set of the linear interval system. By providing tight interval bounds on the solution set, the goal is to more quickly contract intervals that may contain stationary points, as well as to more quickly identify intervals that contain a unique stationary point or no stationary point, thus leading to an overall improvement in computational efficiency.

Methods

Interval-Newton

Several good introductions to interval computations are available [11,17,19,28]. A real interval X is defined as the set of real numbers lying between (and including) given upper and lower bounds; that is, $X = [a, b] = \{x \in \Re \mid a \le x \le b\}$. A real interval vector $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)^{\mathrm{T}}$ has n real interval components and can be interpreted geometrically as an n-dimensional rectangle or box. Note that in this context uppercase quantities are intervals, and lowercase quantities are real numbers. Basic arithmetic operations with intervals are defined by $X \,\mathrm{op}\, Y = \{x \,\mathrm{op}\, y \mid x \in X, y \in Y\}$, where $\mathrm{op} \in \{+, -, \times, \div\}$. Interval versions of the elementary functions can be similarly defined. It should be emphasized that, when machine computations with interval arithmetic operations are done, as in the procedures outlined below, the endpoints of an interval are computed with a directed (outward) rounding. That is, the lower endpoint is rounded down to the next machine-representable number and the upper endpoint is rounded up to the next machine-representable number. In this way, through the use of interval, as opposed to floating-point arithmetic, any potential rounding error problems are avoided and rigorous enclosures are maintained. Implementations of interval arithmetic and elementary functions are readily available, and recent compilers from Sun Microsystems directly support interval arithmetic and an interval data type.

For an arbitrary function $f(\boldsymbol{x})$, the *interval extension*, $F(\boldsymbol{X})$, encloses all values of $f(\boldsymbol{x})$ for $\boldsymbol{x} \in \boldsymbol{X}$; that is, it encloses the range of $f(\boldsymbol{x})$ over $\boldsymbol{X}$. It is often computed by substituting the given interval $\boldsymbol{X}$ into the function $f(\boldsymbol{x})$ and then evaluating the function using interval arithmetic. This so-called "natural" interval extension may be wider than the actual range of function values, though it always includes the actual range. For the case in which the function is a single-use expression, that is, an expression in which each variable occurs only once, natural interval arithmetic will always yield the true function range. For cases in which such rearrangements are not possible, there are a variety of other approaches that can be used to try to tighten interval extensions [11,17,19,28,29].

Of interest here is the interval-Newton method and its application to the stationarity condition $\nabla\phi(\boldsymbol{x}) = \boldsymbol{0}$. Given an $n \times n$ nonlinear equation system $\boldsymbol{f}(\boldsymbol{x}) = \nabla\phi(\boldsymbol{x}) = \boldsymbol{0}$ with a finite number of real roots in some initial interval, this technique provides the capability to find tight enclosures of *all* the roots of the system that lie within the given initial interval. An outline of the interval-Newton methodology is given here. More details are available elsewhere [11,19,34]. It should be emphasized that this technique is *not* equivalent to simply

implementing the routine "point" Newton method in interval arithmetic.

Given some initial interval $\boldsymbol{X}^{(0)}$, the interval-Newton algorithm is applied to a sequence of subintervals, which arises due to a bisection process. Consider a subinterval $\boldsymbol{X}^{(k)}$ in the sequence. Before application of interval-Newton, measures are usually taken first to try to eliminate, or at least shrink, this subinterval. For example, one may apply a function range test. An interval extension $\boldsymbol{F}(\boldsymbol{X}^{(k)})$ of the function $\boldsymbol{f}(\boldsymbol{x})$ is calculated. If there is any component of the interval extension $\boldsymbol{F}(\boldsymbol{X}^{(k)})$ that does not include zero, then the interval can be discarded, since no solution of $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ can exist in this interval. The next subinterval in the sequence may then be considered. Otherwise, testing of $\boldsymbol{X}^{(k)}$ continues. A variety of other interval-based techniques (e. g., constraint propagation) may also be applied to try to shrink $\boldsymbol{X}^{(k)}$ before proceeding to the interval-Newton procedure.

In the interval-Newton procedure, the linear interval equation system

$$\boldsymbol{F}'(\boldsymbol{X}^{(k)})(\boldsymbol{N}^{(k)} - \boldsymbol{x}^{(k)}) = -\boldsymbol{f}(\boldsymbol{x}^{(k)}), \tag{3}$$

is solved for a new interval $\boldsymbol{N}^{(k)}$, where $\boldsymbol{F}'(\boldsymbol{X}^{(k)})$ is an interval extension of the Jacobian of $\boldsymbol{f}(\boldsymbol{x})$, and $\boldsymbol{x}^{(k)}$ is an arbitrary point in $\boldsymbol{X}^{(k)}$. It has been shown [11,19,28] that any root contained in $\boldsymbol{X}^{(k)}$ is also contained in the *image* $\boldsymbol{N}^{(k)}$. This implies that if the intersection between $\boldsymbol{X}^{(k)}$ and $\boldsymbol{N}^{(k)}$ is empty, then no root exists in $\boldsymbol{X}^{(k)}$, and also suggests the iteration scheme $\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} \cap \boldsymbol{N}^{(k)}$. In addition, it has also been shown [11,19,28] that, if $\boldsymbol{N}^{(k)}$ is in the interior of $\boldsymbol{X}^{(k)}$, then there is a *unique* root contained in $\boldsymbol{X}^{(k)}$ and thus in $\boldsymbol{N}^{(k)}$. Thus, after computation of $\boldsymbol{N}^{(k)}$ from Eq. (3), there are three possibilities: (1) $\boldsymbol{X}^{(k)} \cap \boldsymbol{N}^{(k)} = \varnothing$, meaning there is no root in the current interval $\boldsymbol{X}^{(k)}$ and it can be discarded; (2) $\boldsymbol{N}^{(k)}$ is in the interior of $\boldsymbol{X}^{(k)}$, meaning that there is *exactly* one root in the current interval $\boldsymbol{X}^{(k)}$; (3) neither of the above, meaning that no conclusion can be drawn. In the last case, if $\boldsymbol{X}^{(k)} \cap \boldsymbol{N}^{(k)}$ is sufficiently smaller than $\boldsymbol{X}^{(k)}$, then the interval-Newton test can be reapplied to the resulting intersection, $\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} \cap \boldsymbol{N}^{(k)}$. Otherwise, the intersection $\boldsymbol{X}^{(k)} \cap \boldsymbol{N}^{(k)}$ is bisected, and the resulting two subintervals are added to the sequence of subintervals to be tested. If an interval containing a unique root has been identified, then this root can be tightly enclosed by continuing the interval-Newton iteration, which will converge quadratically to a desired tolerance (on the enclosure diameter).

At termination, when the subintervals in the sequence have all been tested, either all the real roots of $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ have been tightly enclosed, or it is determined that no root exists. Applied to nonlinear equation solving problems, this can be regarded as a type of branch-and-prune scheme on a binary tree. It should be emphasized that the enclosure, existence, and uniqueness properties discussed above, which are the basis of the method, can be derived without making any strong assumptions about the function $\boldsymbol{f}(\boldsymbol{x})$ for which roots are sought. The function must have a *finite* number of roots over the search interval of interest; however, no special properties such as convexity or monotonicity are required, and $\boldsymbol{f}(\boldsymbol{x})$ may have transcendental terms.

Solution of Linear Interval Equation Systems

Clearly, the solution of the linear interval system given by Eq. (3) is essential to the interval-Newton approach. To see the issues involved in solving such a system, consider the general linear interval system $A\boldsymbol{z} = \boldsymbol{B}$, where the matrix A and the right hand side vector $\boldsymbol{B}$ are interval-valued. The solution set S of this system is defined by $S = \left\{\boldsymbol{z} \,\middle|\, \tilde{A}\boldsymbol{z} = \boldsymbol{b}, \tilde{A} \in A, \boldsymbol{b} \in \boldsymbol{B}\right\}$. However, in general this set is not an interval and may have a very complex polygonal geometry. Thus to "solve" the linear interval system, one instead seeks an interval $\boldsymbol{Z}$ containing S. Computing the interval hull (the tightest interval containing S) is NP-hard [33], but there are several methods for determining an interval $\boldsymbol{Z}$ that contains but overestimates S. Various interval-Newton methods differ in how they solve Eq. (3) for $\boldsymbol{N}^{(k)}$ and thus in the tightness with which the solution set is enclosed. By obtaining bounds that are as tight as possible, the overall performance of the interval-Newton approach can be improved, since with a smaller $\boldsymbol{N}^{(k)}$ the volume of $\boldsymbol{X}^{(k)} \cap \boldsymbol{N}^{(k)}$ is reduced, and it is also more likely either that $\boldsymbol{X}^{(k)} \cap \boldsymbol{N}^{(k)} = \varnothing$ that $\boldsymbol{N}^{(k)}$ is in the interior of $\boldsymbol{X}^{(k)}$ will be satisfied. Thus, intervals that may contain solutions of the nonlinear equation system are more quickly contracted, and intervals that contain no solution or that contain a unique solution may be more quickly identified, all of which leads to a likely reduction in the number of bisections needed.

Frequently, $\boldsymbol{N}^{(k)}$ is computed component-wise using an interval Gauss–Seidel approach, preconditioned with an inverse-midpoint matrix. Though the inverse-midpoint preconditioner is a good general-purpose preconditioner, it is not always the most effective approach [18,19]. A hybrid preconditioning approach (HP/RP) [10], which combines a simple pivoting preconditioner with the standard inverse-midpoint scheme, has been shown to be significantly more efficient than the inverse-midpoint preconditioner alone on some applications. However, it still may not yield the tightest enclosure of the solution set, which, as noted above, is in general an NP-hard problem. Nevertheless, it is possible, using an LP-based strategy, to compute exact component-wise bounds on the solution set required in the context of the interval-Newton method, while avoiding exponential time complexity. This method is described next.

LP Strategy for Interval-Newton Method

Many types of methods have been proposed for bounding the solution set of a system of linear interval equations. One such method is based on the use of LP techniques [1,3,15,17]. Consider again the linear interval system $A\boldsymbol{z} = \boldsymbol{B}$. Oettli and Prager [31] showed that the solution set S is determined by the constraints:

$$\left|\hat{A}\boldsymbol{z} - \hat{\boldsymbol{B}}\right| \leq \Delta A\,|\boldsymbol{z}| + \Delta\boldsymbol{B}, \tag{4}$$

where $\hat{A}$ is the component-wise midpoint matrix of the interval matrix A, ΔA is the component-wise half-width (radius) matrix of A, and similarly $\hat{\boldsymbol{B}}$ and $\Delta\boldsymbol{B}$ are the midpoint and radius of $\boldsymbol{B}$. Eq. (4) is not directly useful for computing bounds on the solution set because of the absolute value operation on the right-hand side. In general, the solution may lie in all 2^n orthants for an n-dimensional problem. In each orthant, each component of $\boldsymbol{z}$ keeps a constant sign, and thus the absolute value can be dropped. For a given orthant, define the diagonal matrix D_α by

$$(D_\alpha)_{jj} = \begin{cases} 1 & z_j \geq 0 \\ -1 & z_j \leq 0 \end{cases} \quad j = 1, 2, \ldots, n. \tag{5}$$

Thus $|\boldsymbol{z}| = D_\alpha \boldsymbol{z}$ and $\boldsymbol{z} = D_\alpha\,|\boldsymbol{z}|$. Eq. (4) becomes:

$$\left|\hat{A}\boldsymbol{z} - \hat{\boldsymbol{B}}\right| \leq \Delta A D_\alpha \boldsymbol{z} + \Delta\boldsymbol{B}. \tag{6}$$

This can be rearranged to the set of linear inequalities

$$\begin{pmatrix} \hat{A} - \Delta A D_\alpha \\ -\hat{A} - \Delta A D_\alpha \end{pmatrix} \boldsymbol{z} \leq \begin{pmatrix} \overline{\boldsymbol{B}} \\ -\underline{\boldsymbol{B}} \end{pmatrix}, \tag{7}$$

where the underline and overline denote lower and upper interval bounds, respectively. To determine the tightest interval enclosing the solution set, one can then solve, in each orthant, the set of $2n$ optimization problems

$$\max_{\boldsymbol{z}} z_j, \qquad j = 1, 2, \ldots, n, \tag{8}$$

$$\min_{\boldsymbol{z}} z_j, \qquad j = 1, 2, \ldots, n, \tag{9}$$

each with the $2n$ linear inequality constraints given by Eq. (7). These can be solved using linear programming (LP) techniques. However, in general, there are 2^n orthants and so the solution time complexity will be exponential, as expected since this problem is known to be NP-hard.

In the context of the interval-Newton method, however, the exponential time complexity can be avoided. This is because only that part of the solution set of Eq. (3) that intersects $\boldsymbol{X}^{(k)}$ needs to be found. Consider the choice of the real point $\boldsymbol{x}^{(k)}$ in Eq. (3). Here $\boldsymbol{x}^{(k)}$ is an arbitrary point in $\boldsymbol{X}^{(k)}$ typically taken to be the midpoint. However, if $\boldsymbol{x}^{(k)}$ is chosen to be a corner of $\boldsymbol{X}^{(k)}$ instead, then the part of the solution set for $\boldsymbol{N}^{(k)} - \boldsymbol{x}^{(k)}$ of Eq. (3) that intersects $\boldsymbol{X}^{(k)}$ lies in just one orthant. Thus, in the context of interval-Newton, only $2n$ LP subproblems, each with $2n$ constraints, needs to be solved. Furthermore, the LP subproblems have properties that can be exploited. First, all the $2n$ subproblems share the same constraints; that is, they all have the same feasible region. Thus, an initial feasible basis for the LP subproblems needs to be found only once. Second, the objective function of each subproblem consists of just one variable. This makes the problem much simpler since it is not necessary, as it is in the general case, to calculate the gain in objective value when choosing variables to enter and exit the basis.

Lin and Stadtherr [23] have implemented the approach outlined above in the procedure LISS_LP (Linear Interval System Solver by Linear Programming), and incorporated it in an interval-Newton method for global optimization. Two key aspects of the implementation are:

1. The choice of a corner of $\boldsymbol{X}^{(k)}$ to be used in the LP problem. For this purpose, a heuristic approach [23] was developed that incorporates ideas from the pivoting preconditioner approach of Gau and Stadtherr [10].
2. Determination of rigorous error bounds on the solution of the LP problems. This is done using a procedure based on primal/dual relationships [16,23,30]. Complete details of the implementation are given by Lin and Stadtherr [23]. We turn next to some examples that demonstrate the performance of the LP-based strategy as implemented in LISS_LP.

Cases

Lin and Stadtherr [21,23] have tested the performance of the LP-based interval-Newton strategy for global optimization on a variety of problems. In this section, we summarize the results on a group of parameter estimation problems. Each parameter estimation case used was formulated using the error-in-variables approach, with complete details given by Gau and Stadtherr [8,9]. Comparisons are made to an interval Gauss–Seidel method with a hybrid preconditioning approach (HP/RP), which has been shown [10] to provide a substantial improvement in computational performance relative to standard implementations of the interval-Newton approach. Comparisons are made in terms of the number of interval-Newton (I-N) tests required, i. e., the number of times Eq. (3) must be solved, and in terms of the CPU time on a Sun Blade 1000 Model 1600 workstation. On a current (early 2007) workstation, these CPU times would be approximately an order of magnitude less.

Problem 1

This problem [6,20] involves estimation of binary parameters in the van Laar equation for activity coefficients. These two parameters are estimated from vapor-liquid equilibrium data for the binary system of methanol and 1,2-dichloroethane. Computational performance results are shown in Table 1. When the LP-based strategy (LISS_LP) is applied, the number of I-N tests is substantially reduced relative to HP/RP, indicating its effectiveness in reducing the number of intervals that must be tested. Essentially, by reducing the size of $\boldsymbol{N}^{(k)}$, the LP approach is able to more quickly identify and discard intervals that do not contain a stationary point. However, the percent reduction in overall CPU time is less than the percent reduction in I-N tests. This occurs due to the overhead in solving the LP subproblems.

LP Strategy for Interval-Newton Method in Deterministic Global Optimization, Table 1
Computational performance of LP-based method (LISS_LP) and preconditioned interval Gauss–Seidel method (HP/RP) on a Sun Blade 1000 Model 1600

Problem	Variables (n)	HP/RP		LISS_LP	
		I-N Tests	CPU time (s)	I-N Tests	CPU time (s)
1	12	303589	664.4	156182	496.7
2	264	220	1357.3	81	504.9
3	22	9505	24.0	1258	12.7
4	32	144833	976.2	24817	837.2
5	59	55255	2315.9	9757	1692.4

Problem 2

In this problem [4], the rating parameters are estimated for a steady-state heat exchanger network, which consists of four heat exchangers. The four parameters can be estimated from experimental measurements, including six flow measurements and thirteen temperature measurements. In the version of the problem considered here, 20 data points were considered, leading to an optimization problem involving 264 independent variables. Due to the large number of variables, sparse linear programming routines were implemented in LISS_LP for this problem. In this case, both I-N tests and CPU time are substantially reduced when the LP-based method is used, as shown in Table 1, indicating that the LP overhead is less significant on this relatively large problem.

Problems 3 and 4

Both of these problems involve the estimation of kinetic parameters for an irreversible, first-order reaction $A \rightarrow B$. In Problem 3 [6,20], data from an adiabatic continuous-stirred-tank reactor (CSTR) is used, and in Problem 4 [2] data from an isothermal batch reactor is used. In both cases, the reaction rate constant k is given by an Arrhenius expression

$$k = \theta_1 \exp\left(-\frac{\theta_2}{T}\right), \tag{10}$$

in which two parameters, θ_1 and θ_2, must be determined from experimental measurements. Again, the computational performance results (Table 1) show that use of the LP-based strategy substantially reduces the number of I-N tests required relative to HP/RP, but that a comparable reduction in CPU time is not achieved. For example, on Problem 4, the number of I-N tests is reduced by nearly a factor of 6, but there is only about a 15% reduction in CPU time. This reflects the fact that an I-N test performed using the LP method requires greater computational effort than an I-N test using the HP/RP approach. However, on problems studied by Lin and Stadtherr [21,23], this overhead was always offset by a large reduction in the number of I-N tests, resulting in computational savings on all but very small problems.

Problem 5

In this problem, parameters are estimated in a model of an isothermal pseudo-differential reactor for the catalytic hydrogenation of phenol on a palladium catalyst [32]. There are 28 experimental kinetic data points of the partial pressure of phenol (P_1), the partial pressure of hydrogen (P_2), and the initial reaction rate (r). It is desired to fit this kinetic data to a semi-empirical model of the form

$$r = \frac{\theta_1 \theta_2^2 \theta_3 P_1 P_2^2}{(1 + \theta_1 P_1 + \theta_2 P_2)^3}, \tag{11}$$

where θ_1, θ_2 and θ_3 are the parameters to be estimated. This global optimization problem has 59 independent variables. Due to the relatively large number of variables in this problem, a sparse linear programming routine was used in LISS_LP. As seen in Table 1, both I-N tests and CPU time are reduced nicely compared to HP/RP when the LP-based method is used. As in the case of Problem 2, on this relatively large problem the impact of the LP overhead appears to be less significant.

Conclusions

In this article, we have described an LP-based strategy [21,23] for solving the linear interval equation system arising in the context of the interval-Newton approach for deterministic global optimization. The method can obtain tighter bounds on the solution set of the linear interval system than the preconditioned interval Gauss–Seidel approach, and thus leads to a large reduction in the number of subintervals that must be tested during the interval-Newton procedure. However, the difference between the overhead required to solve the LP subproblems and that required to perform the preconditioned Gauss–Seidel method may lead to relatively smaller or larger improvements in overall computational time, depending on the size of the problem. With sparse linear algebra in the LP subproblems, the method can be successfully applied to deterministic global optimization problems involving over two hundred variables, providing a rigorous guarantee of global optimality.

References

1. Aberth O (1997) The solution of linear interval equations by a linear programming method. Lin Algebr Appl 259:271–279
2. Bard Y (1974) Nonlinear Parameter Estimation. Academic Press, New York
3. Beaumont O (1998) Solving interval linear systems with linear programming techniques. Lin Algebr Appl 281:293–309
4. Biegler LT, Tjoa IB (1980) A parallel implementation for parameter estimation with implicit models. Anns Opns Res 42:1–23
5. Burgos-Solórzano GI, Brennecke JF, Stadtherr MA (2004) Validated computing approach for high-pressure chemical and multiphase equilibrium. Fluid Phase Equilib 219:245–255
6. Esposito WR, Floudas CA (1998) Parameter estimation of nonlinear algebraic models via the error-in-variables approach. Ind Eng Chem Res 37:1841–1858
7. Gau CY, Brennecke JF, Stadtherr MA (2000) Reliable nonlinear parameter estimation in VLE modeling. Fluid Phase Equilib 168:1–18
8. Gau CY, Stadtherr MA (2000) Reliable nonlinear parameter estimation using interval analysis: Error-in-variable approach. Comput Chem Eng 24:631–637
9. Gau CY, Stadtherr MA (2002) Deterministic global optimization for error-in-variables parameter estimation. AIChE J 48:1192–1197
10. Gau CY, Stadtherr MA (2002) New interval methodologies for reliable chemical process modeling. Comput Chem Eng 26:827–840
11. Hansen ER, Walster GW (2004) Global Optimization Using Interval Analysis. Marcel Dekker, New York
12. Hua JZ, Brennecke JF, Stadtherr MA (1996) Reliable prediction of phase stability using an interval-Newton method. Fluid Phase Equilib 116:52–59
13. Hua JZ, Brennecke JF, Stadtherr MA (1998) Enhanced interval analysis for phase stability: Cubic equation of state models. Ind Eng Chem Res 37:1519–1527

14. Hua JZ, Brennecke JF, Stadtherr MA (1998) Reliable computation of phase stability using interval analysis: Cubic equation of state models. Comput Chem Eng 22:1207–1214
15. Jansson C (1997) Calculation of exact bounds for the solution set of linear interval systems. Lin Algebr Appl 251:321–340
16. Jansson C (2004) A rigorous lower bound for the optimal value of convex optimization problems. J Glob Optim 28:121–137
17. Jaulin L, Kieffer M, Didrit O, Walter É (2001) Applied Interval Analysis. Springer, London
18. Kearfott RB (1990) Preconditioners for the interval Gauss–Seidel method. SIAM J Numer Anal 27:804–822
19. Kearfott RB (1996) Rigorous Global Search: Continuous Problems. Kluwer, Dordrecht
20. Kim I, Leibman M, Edgar T (1990) Robust error-in-variables estimation using nonlinear programming techniques. AIChE J 36:985–993
21. Lin Y, Stadtherr MA (2004) Advances in interval methods for deterministic global optimization in chemical engineering. J Glob Optim 29:281–296
22. Lin Y, Stadtherr MA (2004) Locating stationary points of sorbate-zeolite potential energy surfaces using interval analysis. J Chem Phys 121:10159–10166
23. Lin Y, Stadtherr MA (2004) LP strategy for the interval-Newton method in deterministic global optimization. Ind Eng Chem Res 43:3741–3749
24. Lin Y, Stadtherr MA (2005) Deterministic global optimization of molecular structures using interval analysis. J Comput Chem 26:1413–1420
25. Lin Y, Stadtherr MA (2006) Deterministic global optimization for parameter estimation of dynamic systems. Ind Eng Chem Res 45:8438–8448
26. Maier RW, Stadtherr MA (2001) Reliable density-functional-theory calculations of adsorption in nanoscale pores. AIChE J 47:1874–1884
27. McKinnon KIM, Millar CG, Mongeau M (1996) Global optimization for the chemical and phase equilibrium problem using interval analysis. In: Floudas CA Pardalos PM (eds) State of the Art in Global Optimization: Computational Methods and Applications. Kluwer, Dordrecht, pp 365–382
28. Neumaier A (1990) Interval Methods for Systems of Equations. Cambridge University Press, Cambridge
29. Neumaier A (2003) Taylor forms – Use and limits. Reliable Comput 9:43–79
30. Neumaier A, Shcherbina O (2004) Safe bounds in linear and mixed-integer programming. Math Prog 99:283–296
31. Oettli W, Prager W (1964) Compatibility of approximate solution of linear equation with given error bounds for coefficients and right-hand sides. Numer Math 6:405–408
32. Rod V, Hancil V (1980) Numerical methods for estimating parameters in nonlinear models with errors in the variables. Technometrics 27:33
33. Rohn J, Kreinovich V (1995) Computing exact componentwise bounds on solution of linear systems with interval data is NP-hard. SIAM J Matrix Anal 16:415–420
34. Schnepper CA, Stadtherr MA (1996) Robust process simulation using interval methods. Comput Chem Eng 20(2):187–199
35. Scurto AM, Xu G, Brennecke JF, Stadtherr MA (2003) Phase behavior and reliable computation of high-pressure solid-fluid equilibrium with cosolvents. Ind Eng Chem Res 42:6464–6475
36. Stadtherr MA, Schnepper CA, Brennecke JF (1995) Robust phase stability analysis using interval methods. AIChE Symp Ser 91(304):356
37. Tessier SR, Brennecke JF, Stadtherr MA (2000) Reliable phase stability analysis for excess Gibbs energy models. Chem Eng Sci 55:1785–1796
38. Westerberg KM, Floudas CA (1999) Locating all transition states and studying the reaction pathways of potential energy surfaces. J Chem Phys 110:9259–9295
39. Xu G, Brennecke JF, Stadtherr MA (2002) Reliable computation of phase stability and equilibrium from the SAFT equation of state. Ind Eng Chem Res 41:938–952
40. Xu G, Haynes WD, Stadtherr MA (2005) Reliable phase stability analysis for asymmetric models. Fluid Phase Equilib 235:152–165
41. Xu G, Scurto AM, Castier M, Brennecke JF, Stadtherr MA (2000) Reliable computational of high-pressure solid-fluid equilibrium. Ind Eng Chem Res 39:1624–1636

L-shaped Method for Two-stage Stochastic Programs with Recourse

FRANCOIS LOUVEAUX[1], JOHN R. BIRGE[2]
[1] Université Namur, Namur, Belgium
[2] Northwestern University, Evanston, USA

MSC2000: 90C15

Article Outline

Keywords
L-Shaped Method for Two-Stage Stochastic Program with Bounded, Complete Recourse
See also
References

Keywords

Stochastic programming; Recourse; Decomposition techniques

A *stochastic linear program with recourse* (SLP) is a mathematical program of the form

$$\begin{cases} \min & c \cdot x + Q(x) \\ \text{s.t.} & Ax = b, \quad x \geq 0, \end{cases}$$

where $Q(x) = \mathsf{E}_\xi Q(x, \xi)$,

$$Q(x, \xi) = \begin{cases} \min & q, y(\xi) \\ \text{s.t.} & W \cdot y(\xi) = h - T \cdot x, y \geq 0, \end{cases}$$

and E_ξ denotes the mathematical expectation with respect to ξ, x is an $(n_1 \times 1)$ decision vector, and for each ξ, y is $(n_2 \times 1)$. A is $(m_1 \times n_1)$ and for each ξ, h is $(m_2 \times 1)$. All other matrices and vectors have conformable dimensions. Transposes are omitted for simplicity. The random vector ξ is formed by the random components of q, h and $T \cdot Q(x, \xi)$ is the second-stage value function for a given ξ and $Q(x)$ the expected value-function or expected recourse.

In the case where random vectors are described by discrete distributions, $Q(x)$ is a piecewise linear convex function of x, so that classical decomposition techniques may apply. Let $k = 1, \ldots, K$ index the possible realizations of ξ, let p_k be their probabilities and y_k be the corresponding second stage decision variables. SLP is then equivalent to the *extensive form*

$$\text{(EF)} \quad \begin{cases} \min & cx + \sum_{k=1}^{K} p_k q_k y_k \\ \text{s.t.} & Ax = b, \\ & T_k x + W y_k = h_k, \\ & \quad k = 1, \ldots, K, \\ & x, y_k \geq 0. \end{cases}$$

This extensive form possesses a *dual block-angular structure*. It is thus well suited to application of *Benders decomposition*, which in the case of stochastic programming is known as the *L-shaped method*. An abbreviated presentation is as follows. It is restricted to the case where all second stage programs are bounded and feasible for any choice of first-stage decision.

L-Shaped Method for Two-Stage Stochastic Program with Bounded, Complete Recourse

Consider the master linear program

$$\text{(MLP)} \quad \begin{cases} \min & cx + \theta \\ \text{s.t.} & Ax = b, \\ & \mathsf{E}_j x + \theta \geq e_j, \quad j = 1, \ldots, s, \\ & x \geq 0, \end{cases}$$

with s *optimality cuts* (initially $s = 0$) and θ a lower bound on $Q(x)$, (θ is omitted when $s = 0$).

Using the solution x^s, θ^s of (MLP) at iteration s, find optimal solutions to the K subproblems,

$$\begin{cases} \min & w = q_k y \\ \text{s.t.} & Wy = h_k - T_k x^s, \\ & y \geq 0, \end{cases}$$

with optimal simplex multipliers π_k^s, $k = 1, \ldots, K$.

Define $E_{s+1} = \sum_{k=1}^{K} p_k \pi_k^s T_k$ and $e_{s+1} = \sum_{k=1}^{K} p_k \pi_k^s h_k$.

If $e_{s+1} - E_{s+1} x^s \leq \theta^s$, then stop as x^s is an optimal solution. Otherwise, set $s = s + 1$ and return to the master program.

Finite convergence of the L-shaped method is proved through classical convexity arguments. When the second stage does not have complete recourse, some first stage decisions may imply that no feasible recourse exists for some k. Then, a number of *feasibility cuts* are also needed. They are obtained through the optimal simplex multipliers of some phase-1 problem. Although these cuts should theoretically be generated for all realizations $k = 1, \ldots, K$, there are many situations where the search for feasibility cuts can be limited to one selected second-stage only [9].

The L-shaped method can be made more efficient by performing some bunching to obtain optimal multipliers for several realizations of ξ at once (see the experiments in [4]). Efficiency can sometimes be gained by sending disaggregated cuts (also called multicuts) instead of one fully aggregated cut at each iteration [2]. Another way of improving the efficiency of the L-shaped is to include a quadratic regularizing term in the first-stage objective function. This additional term is typically the square of the Euclidean distance between

the decision x and the previous iterate point x^s [8]. *L*-shaped methods have also been combined with statistical estimation, in particular to cope with continuous random variables (see [5] and ► Discretely distributed stochastic programs: Descent directions and efficient points).

A number of alternatives to the *L*-shaped techniques have been proposed to solve SLP. One is to use the *Lagrangian finite generation method*, also known as *scenario aggregation* [7]. Another is to use interior points techniques [1]. For a general presentation of stochastic programming, see [3] or [6].

See also

- ► Approximation of Extremum Problems with Probability Functionals
- ► Approximation of Multivariate Probability Integrals
- ► Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ► Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ► General Moment Optimization Problems
- ► Logconcave Measures, Logconvexity
- ► Logconcavity of Discrete Distributions
- ► Multistage Stochastic Programming: Barycentric Approximation
- ► Preprocessing in Stochastic Programming
- ► Probabilistic Constrained Linear Programming: Duality Theory
- ► Probabilistic Constrained Problems: Convexity Theory
- ► Simple Recourse Problem: Dual Method
- ► Simple Recourse Problem: Primal Method
- ► Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ► Static Stochastic Programming Models
- ► Static Stochastic Programming Models: Conditional Expectations
- ► Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ► Stochastic Integer Programs
- ► Stochastic Linear Programming: Decomposition and Cutting Planes
- ► Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ► Stochastic Network Problems: Massively Parallel Solution
- ► Stochastic Programming: Minimax Approach
- ► Stochastic Programming Models: Random Objective
- ► Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ► Stochastic Programming with Simple Integer Recourse
- ► Stochastic Programs with Recourse: Upper Bounds
- ► Stochastic Quasigradient Methods in Minimax Problems
- ► Stochastic Vehicle Routing Problems
- ► Two-stage Stochastic Programming: Quasigradient Method
- ► Two-stage Stochastic Programs with Recourse

References

1. Birge JR, Holmes DF (1992) Efficient solution of two-stage stochastic linear programming using interior point methods. Comput Optim Appl 1:245–276
2. Birge JR, Louveaux FV (1988) A multicut algorithm for two-stage stochastic linear programs. Europ J Oper Res 34:384–392
3. Birge JR, Louveaux FV (1997) Introduction to stochastic programming. Springer, Berlin
4. Gassmann HI (1990) MSLiP: A computer code for the multistage linear programming problem. Math Program 47:407–423
5. Higle EJ, Sen S (1991) Stochastic decomposition: An algorithm for two stage linear programs with recourse. Math Oper Res 16:650–669
6. Kall P, Wallace SW (1994) Stochastic programming. Wiley, New York
7. Rockafellar RT, Wets RJ-B (1986) A Lagrangian finite generation technique for solving linear-quadratic problems in stochastic programming. Math Program Stud 23:63–96
8. Ruszczyński A (1986) A regularized decomposition for minimizing a sum of polyhedral functions. Math Program 35:309–333
9. Walkup D, Wets RJ-B (1967) Stochastic programs with recourse. SIAM J Appl Math 15:1299–1314

M

Maritime Inventory Routing Problems

MARIELLE CHRISTIANSEN[1], KJETIL FAGERHOLT[1,2]
[1] Section of Managerial Economics, Finance and Operations Research, Norwegian University of Science and Technology, Trondheim, Norway
[2] Norwegian Marine Technology Research Institute (MARINTEK), Trondheim, Norway

Article Outline

Abstract
Introduction
The Basic ISRP
  Problem Description
  Mathematical Model
Extensions of the ISRP
  One Central Supplier or Consumer
  Inventory Constraints in Either Production Ports or Consumption Ports
  Variable Production or Consumption Rate
  Multiple Products
  Use of Spot Charters
  Combined Inventory Routing and Cargo Routing
  Combining Inventory Routing with Other Planning Aspects
Concluding Remarks
Acknowledgements
References

Abstract

Maritime transportation is a heavily utilized mode when large quantities of bulk products need to be transported over long distances. Often, inventories exist at the loading and/or unloading ports of the sailing legs. When the ship operator has the responsibility for both the transportation of the fleet and the inventories at the ports, the underlying planning problem is a maritime inventory routing problem. Here we introduce the reader to various applications within maritime inventory routing and present some examples of research contributions. First we consider and present a mathematical model for the basic problem where a single product is transported and denote this problem the *inventory ship routing problem*. There exist a lot of extensions and variants of the problem. These include, among others, problems with inventories at only one end, variable production/consumption rates, multiple products, use of spot charters and problems that combine inventory routing with other planning aspects. Maritime inventory routing problems are very complex and to the authors' knowledge there exist no commercial optimization-based systems for the shipping industry yet. However, it is probably just a question of time before they become available.

Introduction

In order to survive in a tough global market, many companies have been forced to change their focus from competition between companies to competition between supply chains. Supply chains of companies with foreign sources of raw materials or with overseas customers most often include maritime transportation. Supply chain management and optimization are active fields of research, and we can see applications in almost all industries. So far the focus of such applications has usually not been much on maritime transportation, so there is a great potential and need for research in the area.

A maritime inventory routing problem is defined here as a combined ship routing and scheduling prob-

lem and an inventory management problem. The basic *inventory ship routing problem* (ISRP) concerns the transportation of a single product. The product is produced and stored in inventories at given loading ports and is transported by sea to inventories at unloading or consumption ports. Inventory capacities are defined in all ports. Further, we assume that information about production and consumption rates is given in all ports. To transport the product between the given production and consumption ports, the planners control a heterogeneous fleet of ships. The planning problem is to find routes and schedules for the fleet that minimize the transportation costs without interrupting production or consumption at the storages. Depending on the segment the fleet is operating in, the typical planning period spans from 1 to 2 weeks up to several months.

Most ship scheduling problems studied in the literature are so-called *cargo routing problems* [1]. In cargo routing problems, each cargo is specified by a given loading and unloading port. The quantity of the cargo is given and normally there exist time windows for loading and/or unloading. When planning routes and schedules, the shipping company either seeks to minimize the transportation costs for carrying all contracted cargoes or in addition to maximize profit for optional spot cargoes that may be available. We refer to [4] for a survey on maritime cargo routing problems. The cargo routing problems deviate from the ISPRs in a number of ways. The number of calls at a given port during the planning horizon is not predetermined in the ISRP, neither is the quantity to be loaded or unloaded at each port call. There is also no predefined pickup and delivery pair in the ISRP. The combination of the inventory management and the ship routing and scheduling makes the ISRP a very complex problem.

The inventory routing problem has been focused on in the literature for a couple of decades. Dror and Ball [8] defined the problem as a distribution problem in which each customer maintains a local inventory of a product such as heating oil and consumes a certain amount of that product each day. Given a central supplier (depot), the objective is to minimize the annual delivery costs while attempting to ensure that no customer runs out of the product at any time. The asymmetry between each type of inventory (production and consumption) with only one central supply node (depot) will often be found in road-based inventory routing problems, and more seldom in maritime transportation (ISRP). In the road-based inventory routing problem, the amount unloaded at each customer is often small compared to the total capacity of the vehicle. This is also in contrast with the ISRP, where the ship is often fully loaded and unloaded.

The objective of this article is to introduce the reader to various real planning problems within maritime inventory routing. The purpose is not to give a comprehensive overview of such problems, but rather to present examples of applications and research in the area.

The rest of the article is organized as follows: The first section defines the basic inventory ship routing problem and the underlying mathematical model. Extensions of the basic ISRP are addressed next. Finally, concluding remarks and future research follow.

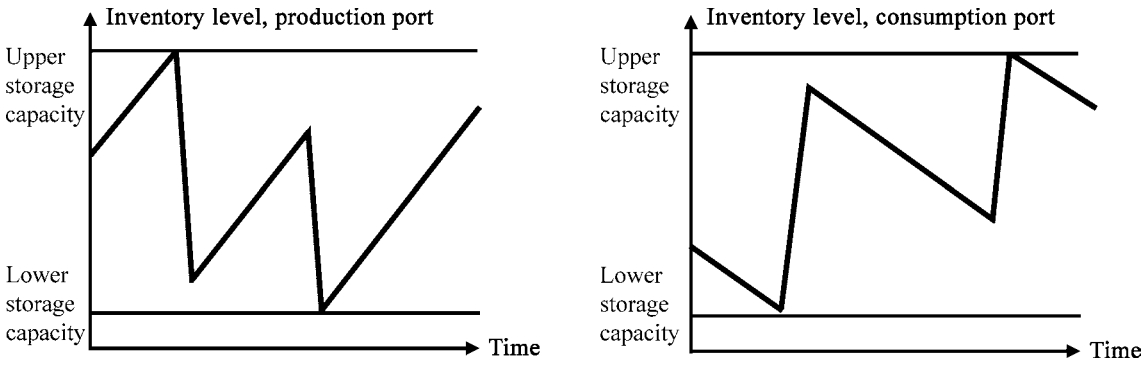
The Basic ISRP

In order to give an introduction to the various real planning problems within maritime inventory routing, we will start with a basic ISRP. First we describe the planning problem. Then we present an arc-flow formulation of the problem . The final section is devoted to real applications of the basic ISRP.

Problem Description

The products transported in maritime inventory routing problems are usually bulk products, where large quantities are transported and there are inventories at both the loading and the unloading ports. In these problems, the ship operators have a twofold responsibility: transportation and inventory management at the production and consumption sites. In such planning situations, the routing and scheduling of the fleet have to be synchronized with the inventory management at both production and consumption sites.

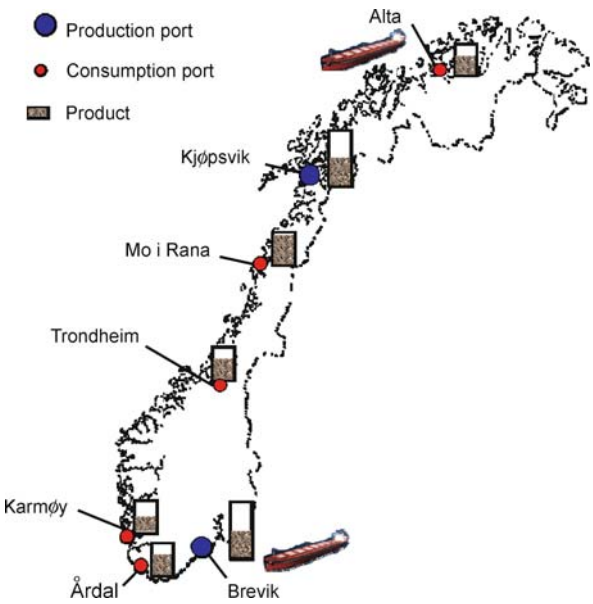
In the basic ISRP a single (homogeneous) product is transported. The product is produced at the sources, called loading ports, and consumed at the destinations, called unloading ports. Inventory storage capacities are given in all ports in addition to the production or consumption rate of the product. Here, the rate is assumed constant during the planning horizon.



Maritime Inventory Routing Problems, Figure 1
Inventory levels during a planning period for a production and a consumption port

The number of calls at a given port during the planning period is not predetermined, nor is the quantity to be loaded or unloaded at each port call. Figure 1 shows an example of a production/consumption and loading/unloading pattern. For both port types, the port is called at twice. However, the quantities loaded or unloaded differ at each call. The reason for this might be that the ports are visited by ships with different capacities loading/unloading full loads, or due to partial loading/unloading. In loading ports, it is important to ensure that the inventory level is not above the maximum inventory level when loading starts and not under the minimum inventory level when the loading has finished. In unloading ports, the opposite has to be ensured. The inventory level at the beginning of the planning period can be at any level, as indicated in Fig. 1.

Therefore, the planning problem is to design routes and schedules that minimize the transportation cost without interrupting production or consumption. We assume no inventory costs because the shipper owns both the producing sources and the consuming destinations. The ship operator controls a heterogeneous fleet of ships. We assume that partial loading and unloading is allowed, such that two ports of the same type (loading or unloading) may be called at in succession. The ship is not necessarily empty at the beginning of the planning horizon, but might have some load onboard. The ship can be either at a port or at sea at the beginning of the planning horizon. Figure 2 shows a simplified illustration of the planning problem for a cement producer in Norway with two production factories and five consumption ports with inventories. The fleet consists of two ships. Each port can be called at several times during the planning period by the same ship or different ships.



Maritime Inventory Routing Problems, Figure 2
A simplified planning situation with seven ports and two ships

Mathematical Model

The model of the ISRP will be presented in a compact and simplified way. In Sect. "Routing," we describe the flow network and the objective function. Then, the con-

ditions for the loading and unloading, the time aspects and the inventories are described in Sect. "Loading and Unloading", "Scheduling" and "Inventory Management," respectively. We base our notation and model on those of Christiansen et al. [3].

In the upcoming formulation, we have assumed that the ship may be partially loaded/unloaded, meaning that multiple cargoes may be onboard a ship simultaneously. The model could have been simplified if we had assumed full loads only or sailing between different port types (from loading to unloading and vice versa).

Routing In the mathematical description of the network each port is represented by an index i and the set of ports is given by $\mathcal{N}$. Let $\mathcal{V}$, indexed by v, be the set of available ships to be routed and scheduled. Not all ships can visit all ports, and $\mathcal{N}_v$ = {feasible ports for ship v} $\cup\{o(v),d(v)\}$ is the set of ports that can be visited by ship v. The terms $o(v)$ and $d(v)$ represent the artificial origin port and artificial destination port of ship v, respectively. Each port can be visited several times during the planning horizon, and $\mathcal{M}_i$ is the set of possible calls at port i, while $\mathcal{M}_{iv}$ is the set of calls at port i that can be made by ship v. The port call number is represented by an index m, and M_i is the last possible call at port i within the planning period. The set of nodes in the flow network represents the set of port calls, and each port call is specified by (i,m), $i \in \mathcal{N}, m \in \mathcal{M}_i$. In addition, we specify flow networks for each ship v with nodes (i,m), $i \in \mathcal{N}_v, m \in \mathcal{M}_{iv}$. $\mathcal{A}_v$ contains all feasible arcs for ship v, which is a subset of $\{i \in \mathcal{N}_v, m \in \mathcal{M}_{iv}\} \times \{i \in \mathcal{N}_v, m \in \mathcal{M}_{iv}\}$. Finally, C_{ijv} represents the variable costs for sailing between port i and port j with ship v. This includes port, channel and fuel costs.

In the network flow part of the formulation we use the following types of variables: the binary flow variable x_{imjnv}, $v \in \mathcal{V}$, $(i,m,j,n) \in \mathcal{A}_v$ equals 1, if ship v sails from node (i,m) directly to node (j,n), and 0 otherwise, and the slack variable w_{im}, $i \in \mathcal{N}$, $m \in \mathcal{M}_i$ is equal to 1 if no ship takes port call (i,m), and 0 otherwise. The routing formulation including the objective function is as follows:

$$\min \sum_{v\in\mathcal{V}} \sum_{(i,m,j,n)\in\mathcal{A}_v} C_{ijv} x_{imjnv}, \tag{1}$$

subject to

$$\sum_{v\in\mathcal{V}} \sum_{j\in\mathcal{N}_v} \sum_{n\in\mathcal{M}_{jv}} x_{imjnv} + w_{im} = 1, \forall i \in \mathcal{N}, m \in \mathcal{M}_i, \tag{2}$$

$$\sum_{j\in\mathcal{N}_v} \sum_{n\in\mathcal{M}_{jv}} x_{o(v)1jnv} = 1, \ \forall v \in \mathcal{V}, \tag{3}$$

$$\sum_{i\in\mathcal{N}_v} \sum_{m\in\mathcal{M}_{iv}} x_{imjnv} - \sum_{i\in\mathcal{N}_v} \sum_{m\in\mathcal{M}_{iv}} x_{jnimv} = 0, \quad \forall v \in \mathcal{V}, j \in \mathcal{N}_v \backslash \{o(v), d(v)\}, n \in \mathcal{M}_{jv}, \tag{4}$$

$$\sum_{i\in\mathcal{N}_v} \sum_{m\in\mathcal{M}_{iv}} x_{imd(v)1v} = 1, \ \forall v \in \mathcal{V}, \tag{5}$$

$$w_{im} - w_{i(m-1)} \geq 0, \ \forall i \in \mathcal{N}, m \in \mathcal{M}_i, \tag{6}$$

$$x_{imjnv} \in \{0,1\}, \ \forall v \in \mathcal{V}, (i,m,j,n) \in \mathcal{A}_v, \tag{7}$$

$$w_{im} \in \{0,1\}, \ \forall i \in \mathcal{N}, m \in \mathcal{M}_i. \tag{8}$$

The objective function (1) minimizes the total costs. Constraints (2) ensure that each port call is visited at most once. Constraints (3)–(5) describe the flow on the sailing route used by ship v. One or several of the calls in a specified port can be made by a dummy ship, and the highest call numbers will be assigned to dummy ships in constraints (6). These constraints reduce the number of symmetrical solutions in the solution approach. For the calls made by a dummy ship, we get artificial starting times and artificial inventory levels within the defined upper and lower limits. Finally, the formulation involves binary requirements (7) and (8) on the flow variables and port call slack variables, respectively.

Loading and Unloading The capacity of ship v is given by V_{CAPv}. Variable l_{imv}, $v \in V$, $i \in \mathcal{N}_v \backslash \{d(v)\}$, $m \in \mathcal{M}_{iv}$ gives the total load onboard ship v just after the service is completed at node (i,m), while variable q_{imv}, $v \in V$, $i \in \mathcal{N}_v \backslash \{d(v)\}$, $m \in \mathcal{M}_{iv}$ represents the quantity loaded or unloaded at port call (i,m), when ship v visits (i,m). It is assumed that nothing is loaded or unloaded at the artificial origin $o(v)$; $q_{o(v)1v} = 0$. However, the ships may have cargo onboard, L_{0v}, at the beginning of the planning horizon; $l_{o(v)1v} = L_{0v}$. Further, constant I_i is equal to 1 if port i is a loading port, -1 if port i is an unloading port and 0 if port i is $o(v)$ or $d(v)$. Constraints related to the quantity onboard a ship

can be formulated as follows:

$$x_{imjnv}(l_{imv} + I_j q_{jnv} - l_{jnv}) = 0, \quad \forall v \in \mathcal{V}, (i,m,j,n) \in \mathcal{A}_v | j \neq d(v), \tag{9}$$

$$q_{imv} \leq l_{imv} \leq \sum_{j \in \mathcal{N}_v} \sum_{n \in \mathcal{M}_{jv}} V_{CAPv} x_{imjnv}, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}_v, m \in \mathcal{M}_{iv} | I_i = 1, \tag{10}$$

$$0 \leq l_{imv} \leq \sum_{j \in \mathcal{N}_v} \sum_{n \in \mathcal{M}_{jv}} V_{CAPv} x_{imjnv} - q_{imv}, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}_v, m \in \mathcal{M}_{iv} | I_i = -1. \tag{11}$$

Constraints (9) give the relationship between the binary flow variables and the ship load at each port call. Constraints (10) and (11) give the ship capacity intervals at the port calls for loading and unloading ports, respectively.

Scheduling The time required to load or unload the ship may constitute a major part of the total time in many maritime transportation applications. It is therefore usual to calculate this as a function of the quantity loaded/unloaded. The time spent loading/unloading one unit of a cargo at port i is given by T_{Qi}. The term T_{Sijv} represents the sailing time from port i to port j with ship v. In some ports, there is a minimum required time, T_{Bi}, between the departure of one ship and the arrival of the next ship, due to small port area or narrow channels from the port to the pilot station. The time variable t_{im}, $(i \in \mathcal{N}, m \in \mathcal{M}_i) \cup (i \in o(v), \forall v, m = 1)$ represents the time at which service begins at node (i,m). It is assumed that the ship arrives at $o(v)$ at a given fixed time; $t_{o(v)1} = T_{0v}$. Finally, let T denote the planning horizon. The scheduling constraints can now be written as follows:

$$x_{imjnv}(t_{im} + T_{Qi} q_{imv} + T_{Sijv} - t_{jn}) \leq 0, \quad \forall v \in \mathcal{V}, (i,m,j,n) \in \mathcal{A}_v | j \neq d(v), \tag{12}$$

$$t_{im} - t_{i(m-1)} - \sum_{v \in \mathcal{V}} T_{Qi} q_{i(m-1)v} + T_{Bi} w_{im} \geq T_{Bi}, \quad \forall i \in \mathcal{N}, m \in \mathcal{M}_i \backslash \{1\}. \tag{13}$$

Constraints (12) take into account the timing or scheduling on the route. Note that waiting at a port is allowed. Constraints (13) prevent service overlap in the ports and ensure the order of real calls at the same port. A ship must complete its service before the next ship starts its service at the same port. If port i does not have constraints regarding the minimum time between departure of one ship and arrival of the next, $T_{Bi} = 0$. If port i also allows the service of several ships simultaneously, constraints (13) will simply be $t_{im} - t_{i(m-1)} \geq 0$, to ensure the order of calls at the port.

Inventory Management The levels of the inventory have to be within a given interval at each port $[S_{MNi}, S_{MXi}]$. The production rate R_i is positive if port i is producing the product, and negative if port i is consuming the product. At the beginning of the planning horizon, the inventory level at each port i is S_{0i} . Finally, s_{im}, $i \in \mathcal{N}, m \in \mathcal{M}_i$ represents the inventory level when service starts at port call (i,m). The inventory constraints of the formulation become

$$s_{i1} - R_i t_{i1} = S_{0i}, \ \forall i \in \mathcal{N}, \tag{14}$$

$$s_{i(m-1)} - \sum_{v \in \mathcal{V}} I_i q_{i(m-1)v} + R_i (t_{im} - t_{i(m-1)}) - s_{im} = 0, \quad \forall i \in \mathcal{N}, m \in \mathcal{M}_i \backslash \{1\}, \tag{15}$$

$$S_{MNi} \leq s_{im} \leq S_{MXi}, \ \forall i \in \mathcal{N}, m \in \mathcal{M}_i, \tag{16}$$

$$S_{MNi} \leq s_{im} - \sum_{v \in \mathcal{V}} I_i q_{imv} + R_i (T - t_{im}) \leq S_{MXi}, \quad \forall i \in \mathcal{N}, m = M_i. \tag{17}$$

The inventory level at the first call at each port is calculated in constraints (14). From constraints (15), we find the inventory level at any port call (i,m) from the inventory level upon arrival at the port in the previous call $(i,m\text{-}1)$, adjusted for the loaded/unloaded quantity at the port call and the production/consumption between the two arrivals. The general inventory limit constraints at each port call are given in (16). Constraints (17) ensure that the level of inventory at the end of the planning horizon is within its limits. It can easily be shown by substitution that constraints (17) ensure that the inventory at time T will be within the bounds even if ports are not visited at their last calls.

A Real Application An application that is close to the ISRP is a real ship planning problem for ammonia transportation. Norsk Hydro Agri (now named Yara) produces and consumes ammonia in its factories worldwide. The planners at the company are re-

sponsible for keeping the inventory levels within the predefined upper and lower limits at all Norsk Hydro Agri factories around the world where they produce and consume ammonia. This requires the planners to design routes and schedules for their fleet of heterogeneous ships transporting ammonia from production ports to consumption ports. The problem is described in detail in Christiansen [2]. The overall solution approach is based on a column generation approach with columns for both the ship routes and the inventory management sequences [5], where subproblems are solved by dynamic programming for each port and each ship [6]. Another solution approach to the same problem was developed by Flatberg et al. [9]. They used an iterative improvement heuristic combined with an linear program (LP) solver to solve this problem. The heuristic is used to solve the combinatorial problem of finding the ship routes, and an LP model is used to find the starting time of service at each call and the loading/unloading quantities.

Extensions of the ISRP

Most of the real applications of maritime inventory routing problems have a more complex structure than the basic ISRP. We present here various extensions of the ISRP that are described in the literature or have been experienced in our research group. In many maritime applications, several of the extensions are combined.

One Central Supplier or Consumer

As mentioned in the introduction, the road-based inventory routing problem often has a vehicle routing problem (VRP) structure, where a central supplier (or depot) serves a set of customers with a local inventory and a consumption rate. We can imagine a lot of real planning problems with such a structure, for instance, in the gasoline business, delivering gasoline to gas stations from a refinery or central storage. Milk collection at farms for transport to a dairy has the opposite structure, where the customers are producers and the depot consumes the milk.

In the maritime sector, we can also find this VRP structure for ship operators dealing with maritime inventory routing problems. The Norwegian oil company Statoil will start its production of natural gas from Snøhvit, Melkøya, north of Norway in 2008. Most of the gas will be cooled down and transported as liquefied natural gas (LNG) by LNG tankers. At the moment the planning problem concerns one source producing the gas and several consumption ports. Frich and Horgen [10] presented a mixed integer program (MIP) model of the planning problem where this special VRP structure is exploited.

Similar maritime inventory routing problems can be found, for instance, with the Arabian Gulf as the source for the transportation of both LNG and heavier oil products.

Inventory Constraints in Either Production Ports or Consumption Ports

For the ISRP, the inventory management is considered at both the loading and the unloading ports. However, many real planning problems concern the design of routes and schedules for a fleet of ships with inventory constraints at just one of the port types. There exist for instance ship operators engaged in vendor managed inventory (VMI) contracts. Here, the ship operator monitors its customers' inventories and must ensure that these are kept within predefined limits. Often, the customers are concerned about inventories at only the unloading ports, while the ship operator has entered into a contract to supply the product with given quantities and time windows from the loading ports. The opposite might also be the case, where the customers have inventories at only loading ports. Then, the ship operator must also engage in contracts to deliver these volumes with given quantities and time windows.

Variable Production or Consumption Rate

The production and consumption rates are assumed constant for all port inventories during the planning period in the ISRP. However, for many real planning problems this assumption is too coarse, and the production and consumption that may vary from day to day have to be taken into account in the modeling. Including this aspect into the basic ISRP model would result in a more complicated model and it would become harder to solve.

A maritime inventory routing problem for the LNG business was considered by Grønhaug et al. [11]. Here the production of LNG at the liquefaction plants and

the consumption of LNG at the regasification terminals have to be regarded as variable. In order to overcome these complicating factors, a time discretized model was developed, and it was solved by a column generation approach.

Also Ronen [16] used a time discretized model with a variable production and consumption rate for an inventory routing problem for refinery products. The model focuses on the inventory and not the routing part of the problem, as the model solution suggests shipment sizes that are assumed to be an input for a cargo routing problem at a later stage.

Multiple Products

Here we extend the ISRP to the multiproduct case. In the ISRP several cargoes may be transported simultaneously in one ship, but the product is assumed to be the same. This means that the product does not need to be transported in separated compartments onboard the ship or stored in separate stores at the ports.

The problem with multiple products is frequently encountered by chemical and oil product transport companies. Al-Khayyal and Hwang [1] gave a mathematical formulation for such a problem where the products are assumed to require dedicated compartments in the ship. For this problem there exist inventory limits and production/consumption rates for each product in each port. Hwang [13] used a combined Lagrangian relaxation and heuristic approach to solve test instances of the problem.

The problem described in Ronen [16] also includes multiple products. Sometimes the stowage onboard the ship must also be considered in the inventory routing problem; see, for instance, Haugen and Lund [12] for the transportation of cement products.

Use of Spot Charters

In some cases the dedicated fleet of ships has insufficient transportation capacity to provide continuous production at all sources and consumption at all destinations. In such a case some of the loads can be serviced by spot charters, which are ships chartered for a single voyage.

The cement company described by Haugen and Lund [12] is faced with limited vessel capacity. In some periods the company makes use of spot charters, while in peak periods additional road-based transportation is necessary. In their solution approach, the consumption inventories are sorted according to their importance and their location regarding what the cost effects for additional trucks will be. It is ensured that the inventories with highest priority are served by the fleet of ships.

Combined Inventory Routing and Cargo Routing

The cargo routing problem was introduced in the introduction. For this problem, there exist predefined cargoes with specified quantities and time windows. The cargoes may be contracted or optional spot ones. Often the companies facing an ISRP trade cargoes with other operators in order to better utilize the fleet and to ensure there is product balance at their own plants.

In the real problem described by Christiansen [2], the shipper trades ammonia with other operators. These traded volumes are determined by negotiations. The ship operator undertakes to load or unload ammonia within a determined quantity interval and to arrive at a particular port within a given time window. For these external ports, no inventory management problem exists.

There also exist shipping companies that have VMI contracts with some customers, but apart from that are involved in ordinary cargo routing. This will give these shipping companies a combined inventory and cargo routing planning problem.

Combining Inventory Routing with Other Planning Aspects

The ISRP concerns parts of a supply chain and focuses on sea transportation and the inventories at both ends of the sailing leg. In many real planning situations, it is sensible to consider larger parts of the supply chain. Persson and Göthe-Lundgren [15] studied a planning problem that integrates both the shipment planning of petroleum products from refineries to depots and the production scheduling at the refineries. Shih [17] and Liu and Sherali [14] presented two other maritime supply chain applications where coal is transported.

Rather than considering a larger part of the supply chain, the ISRP may be combined with other planning aspects. In Sect. "Multiple Products," we referred to the combined ISRP and stowage of different cement prod-

ucts in various compartments onboard the ships. See Haugen and Lund [12] for more information about the case and solution approach.

Concluding Remarks

We have described the maritime inventory routing problem, which is a combined inventory management and a ship routing and scheduling problem. The so-called basic ISRP and several extensions to the ISRP were presented. In practice, planners are more often faced with extensions of the ISRP and also the extensions described in combination with each other.

As far as we know, no generic commercial optimization-based decision support system exists for solving maritime inventory routing problems. However, the shipping industry is experiencing an increased need for such systems owing to extended planning responsibility and increased fleet sizes. We expect that such systems will be available on the market in the years to come.

The basic VRP is computationally very hard. The maritime inventory routing problem is even more demanding owing to the additional degrees of freedom. Many of the extensions discussed in this article are barely touched on in the operations research community. This means that there exist a lot of research challenges, in the development of both exact methods and heuristic solution methods.

Maritime transportation is faced with higher uncertainty in its operations compared with many other modes of transportation. This is due to greater dependence on weather conditions and technology. For the maritime inventory routing problem, we have also uncertainties in the production and consumption at the inventories. The consideration of these uncertainty aspects is another interesting topic of research.

Acknowledgements

This work was carried out with financial support from the Research Council of Norway through the INSUMAR project (Integrated supply chain and maritime transportation planning), the OPTIMAR project (Optimization in maritime transportation and logistics) and the DOMinant project (Discrete optimization methods in maritime and road-based transportation).

References

1. Al-Khayyal F, Hwang S-J (2007) Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, Part I: Applications and model. Eur J Oper Res 176:106–130
2. Christiansen M (1999) Decomposition of a combined inventory and time constrained ship routing problem. Transp Sci 33(1):3–16
3. Christiansen M, Fagerholt K, Nygreen B, Ronen D (2007) Maritime Transportation. In: Barnhart C, Laporte G (eds) Handbooks in Operations Research and Management Science: 14 Transportation. North-Holland, Amsterdam, pp 189–284
4. Christiansen M, Fagerholt K, Ronen D (2004) Ship routing and scheduling: Status and perspectives. Transp Sci 38(1):1–18
5. Christiansen M, Nygreen B (1998) A method for solving ship routing problems with inventory constraints. Ann Oper Res 81:357–378
6. Christiansen M, Nygreen B (1998) Modeling path flows for a combined ship routing and inventory management problem. Ann Oper Res 82:391–412
7. Christiansen M, Nygreen B (2005) Robust inventory ship routing by column generation. In: Desaulniers G, Desrosiers J, Solomon MM (eds) Chapter 7, Column generation. Springer, pp 197–224
8. Dror M, Ball M (1987) Inventory/Routing: Reduction from an Annual to a Short-Period Problem. Nav Res Logist 34: 891–905
9. Flatberg T, Haavardtun H, Kloster O, Løkketangen A (2000) Combining exact and heuristic methods for solving a vessel routing problem with inventory constraints and time windows. Ric Oper 29(91):55–68
10. Frich OT, Horgen R (2004) Supply chain optimization: Snøhvit LNG distribution. Master thesis, Norwegian University of Science and Technology, Trondheim, pp 144
11. Grønhaug R, Christiansen M, Desaulniers G, Desrosiers J (2008) A branch-and-price-and-cut method for a liquefied natural gas inventory routing problem. Working paper, Norwegian University of Science and Technology, Trondheim
12. Haugen Ø, Lund EH (2006) Optimization-based decision support for planning of cement distribution in maritime supply chains. Master thesis, Norwegian University of Science and Technology, Trondheim, pp 137
13. Hwang S-J (2005) Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk. Ph.d. Thesis, Georgia Institute of Technology, Atlanta
14. Liu C-M, Sherali HD (2000) A coal shipping and blending problem for an electric utility company. OMEGA 28:433–444
15. Persson JA, Göthe-Lundgren M (2005) Shipment planning at oil refineries using column generation and valid inequalities. Eur J Oper Res 163:631–652

16. Ronen D (2002) Marine inventory routing: Shipments planning. J Oper Res Soc 53:108–114
17. Shih L-H (1997) Planning of fuel coal imports using a mixed integer programming method. Int J Prod Econ 51:243–249

Mathematical Programming for Data Mining

Ioannis P. Androulakis[1],
W. Art Chaovalitwongse[2]
[1] Department of Biomedical Engineering, Rutgers University, Piscataway, USA
[2] Department of Industrial and Systems Engineering, Rutgers University, Piscataway, USA

Article Outline

Keywords
Introduction
Basic Definitions
Mathematical Programming Formulations
- Classification
- Clustering
- Support Vector Machines
- Multi-Class Support Vector Machines
- Data Mining in the Presence of Constraints
- Data Mining and Integer Optimization

Research Challenges
References

Keywords

Mathematical programming; Data mining; Optimization; Clustering; Classification

Introduction

Progress in digital data acquisition and storage technology has resulted in the growth of huge databases. This has occurred in a variety of scientific and engineering research applications [8] as well as medical domain [19,20]. Making sense out of these rapidly growing massive data sets gave birth to a "new" scientific discipline often referred to as *Data Mining*. Defining a discipline is, however, always a controversial task. The following working definition of the area was recently proposed [9]: Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.

Clearly the term data mining if often used as a synonym for the process of extracting useful information from databases. However, the overall knowledge discovery from databases (KDD) process is far more complicated and convoluted and involves a number of additional pre and post-processing steps [6]. Therefore, in our definition data mining refers to the ensemble of new, and existing, specific algorithms for extracting structure from data [8]. The exact definition of the knowledge extraction process and the expected outcomes are very difficult to characterize. However, a number of specific tasks can be identified and, by and large, define the key subset of deliverables from a data mining activity. Two such critical activities are classification and clustering.

A number of variants for these tasks can be identified and, furthermore, the specific structure of the data involved greatly impacts the methods and algorithms that are to be employed. Before we proceed with the exact definition of the tasks we need to provide working definitions of the nature and structure of the data.

Basic Definitions

For the purposes of our analysis we will assume that the data are expressed in the form of n-dimensional feature vectors $x \in X \subseteq \Re^n$. Appropriate pre-processing of the data may be required to transform the data into this form. Although in many cases this transformations can be trivial, in other cases transforming the data into a "workable" form is a highly non-trivial task. The goal of data mining is to estimate an explicit, or implicit, function that maps points of the feature vector from the input space, $X \subseteq \Re^n$, to an output space, C, given a finite sample. The concept of the finite sample is important because, in general, what we are given is a finite representative subset of the original space (training set) and we wish to make predictions on new elements of the set (testing set). The data mining tasks can thus de defined based on the nature of the mapping C and the extent to which the train set is characterized.

If the predicted quantity is a categorical value and if we know the value that corresponds to each elements of the training set then the question becomes how to

identify the mapping that connects the feature vector and the corresponding categorical value (class). This problem is known as the classification problem (supervised learning). If the class assignment is not known and we seek to: (*a*) identify whether a small, yet unknown, number of classes exist; (*b*) define the mapping assigning the features to classes then we have a clustering problem (unsupervised learning).

A related problem associated with superfluous information in the feature vector is the so-called feature selection problem. This is a problem closely related to over-fitting in regression. Having a minimal number of features leads to simpler models, better generalization and easier interpretation. One of the fundamental issues in data mining is therefore to identify the least number of features, sub-set of the original set of features, that best address the two issues previously defined. The concept of parsimony (Occam's razor) is often invoked to bias the search [1]: never do with more what can be done with fewer.

Although numerous methods exist for addressing these problems they will not be reviewed here. Nice reviews of classification and were recently presented in [8,9]. In this short introduction we will concentrate on solution methodologies based on reformulating the clustering, and classification questions as optimization problems.

Mathematical Programming Formulations

Classification and clustering, and for that matter most of the data mining tasks, are fundamentally optimization problems. Mathematical programming methodologies formalize the problem definition and make use of recent advances in optimization theory and applications for the efficient solution of the corresponding formulations. In fact, mathematical programming approaches, particularly linear programming, have long been used in data mining tasks.

The pioneering work presented in [13,14] demonstrated how to formulate the problem of constructing planes to separate linearly separable sets of points.

In this summary we will follow the formalism put forth in [2] since it presented one of the most comprehensive approaches to this problem. One of the major advantages of a formulation based on mathematical programming is the ease in incorporating explicit problem specific constraints. This will be discussed in greater detail later in this summary.

Classification

As discussed earlier the main goal in classification is to predict a categorical variable (class) based on the values of the feature vector. The general families of methods for addressing this problem include [9]:

- **i)** Estimation of the conditional probability of observing class C given the feature vector x.
- **ii)** Analysis of various proximity metrics and based the decision of class assignment based on proximity.
- **iii)** Recursive input space partitioning to maximize a score of class purity (tree-based methods).

The two-class classification problem can be formulated as the search of a function that assigns a given input vector x into two disjoint point sets A and B. The data are represented in the form of matrices. Assuming that the set A has m elements and the set B has k elements, then $A \in \Re^{m\times n}, B \in \Re^{k\times n}$, describe the two sets respectively. The discrimination in based on the derivation of hyperplane

$$P = \{x | x \in \Re^n, x^T\omega = \gamma\}$$

with normal and distance from the origin $\frac{|\gamma|}{||\omega||_2}$. The optimization problem then becomes to determine ω and γ such that the separating hyperplane P defines two open half spaces

$$\{x | x \in \Re^n, x^T\omega < \gamma\}$$
$$\{x | x \in \Re^n, x^T\omega > \gamma\}$$

containing mostly points in A and B respectively. Unless A and B are disjoint the separation can only be satisfied within some error. Minimization of the average violations provides a possible approximation of the separating hyperplane [2]:

$$\min_{\omega,\gamma} \frac{1}{m}\|(-A\omega + e\gamma + e)_+\|_1 + \frac{1}{k}\|(-B\omega + e\gamma + e)_+\|_1$$

In [2] a number of linear programming reformulations are discussed exploring the properties of the structure of the optimization problem. In particular an effective

robust linear programming (RLP) reformulation was suggested making possible the solution of large-scale problems:

$$\begin{aligned}\min_{\omega,\gamma,y,z}\ & \frac{e^T y}{m}+\frac{e^T z}{k}\\ \text{s.t.}\ & -A\omega+e\gamma+e\le y\\ & B\omega-e\gamma+e\le z\\ & y,z\ge 0.\end{aligned}$$

In [17] it was demonstrated how the above formulation can be applied repeatedly to produce complex space partitions similar to those obtained by the application of standard decision tree methods such as C4.5 [21] or CART [4].

Clustering

The goal of clustering is the segmentation of the raw data into groups that share a common, yet unknown, characteristic property. Similarity is therefore a key property in any clustering task. The difficulty arises from the fact that the process is unsupervised. That is neither the property nor the expected number of groups (clusters) are known ahead of time. The search for the optimal number of clusters is parametric in nature and the optimal point in an "error" vs. "number of clusters" curve is usually identified by a combined objective the weighs appropriately accuracy and number of clusters. Conceptually a number of approaches can be developed for addressing clustering problems:

i) Distance-based methods, by far the most commonly used, that attempt to identify the best k-way partition of the data by minimizing the distance of the points assigned to cluster k from the center of the cluster.

ii) Model-based methods assume the functional form of a model that describes each of the clusters and then search for the best parameter fit that models each cluster by minimizing some appropriate likelihood measure.

There are two different types of clustering: (1) hard clustering; (2) fuzzy clustering. The former assigns a data point to *exactly* one cluster while the latter assigns a data point to one of more clusters along with the likelihood of the data point belonging to one of those clusters.

The standard formulation of the hard clustering problem is:

$$\min_c \sum_{i=1}^{m} \min_l \|x^i - c^l\|_n$$

That is given m points, x, in an n-dimensional space, and a fixed number of cluster, k, determine the centers of the cluster, c, such that the sum of the distances of each point to a nearest cluster center is minimized. It was shown in [3] that this general non convex problem can be reformulated such that we minimize a bilinear functions over a polyhedral set by introducing a selection variable t_{il}:

$$\begin{aligned}\min_{c,d,t}\ & \sum_{i=1}^{m}\sum_{i=1}^{k} t_{il}(e^T d_{il})\\ \text{s.t.}\ & -d_{il}\le x^i-c^l\le d_{il}\\ & \sum_{l=1}^{k} t_{il}=1\\ & t_{il}\ge 0\\ & i=1,\ldots,m,\ l=1,\ldots,k.\end{aligned}$$

d is a dummy variable used to bound the components of the difference $x-c$. In the above formulation the 1-norm is selected [3].

The fuzzy clustering problem can be formulated as follows [5]:

$$\begin{aligned}\min_{w}\ & \sum_{i=1}^{m}\sum_{l=1}^{k} w_{il}^2\|x^i-c^l\|^2\\ \text{s.t.}\ & \sum_{l=1}^{k} w_{il}=1\\ & w_{il}\ge 1,\end{aligned}$$

where $x^i, i=1,\ldots,m$ is the location descriptor for the data point, $c^l, l=1,\ldots,k$ is the center of the cluster, w_{il} is the likelihood of a data point i being assigned to cluster l.

Support Vector Machines

This optimization formalism bares significance resemblance to the Support Vector Machines (SVM) framework [25]. SVM incorporate the concept of structural

risk minimization by determining a separating hyperplane that maximizes not only a quantity measuring the misclassification error but also maximizing the margin separating the two classes. This can be achieved by augmenting the objective of the RLP formulation earlier presented by an appropriately weighted measure of the separation between the two classes as $(1-\lambda)(e^T y + e^T z) + \frac{\lambda}{2}\|\omega\|_2^2$.

In [6] the concept of SVM is extended by introducing the Proximal support vector machines which classify points based on proximity to one of two parallel planes that are pushed as far apart as possible. Nonlinear transformations were also introduced in [6] to enable the derivation of non-linear boundaries in classifiers.

Multi-Class Support Vector Machines

Support vector machines were originally designed for binary classification. Extending to multi-class problems is still an open research area [10].

The earliest multi-class implementation is the *one against all* [22] by constructing k SVM models, where k is the number of classes. The ith SVM is classifies the examples of class i against all the other samples in all other classes. Another alternative builds *one against one* [12] classifiers by building $\frac{k(k-1)}{2}$ models where each is trained on data from two classes. The emphasis of current research is on novel methods for generating all the decision functions through the solution of a single, but much larger, optimization problem [10].

Data Mining in the Presence of Constraints

Prior knowledge about a system is often omitted in data mining applications because most algorithms do not have adequate provisions for incorporating explicitly such types of constrains. Prior knowledge can either encodes explicit and/or implicit relations among the features or models the existence of "obstacles" in the feature space [24].

One of the major advantages of a mathematical programming framework for performing data mining tasks is that prior knowledge can be incorporated in the definition of the various tasks in the form of (non)linear constraints. Efficient incorporation of prior knowledge in the form of nonlinear inequalities within the SVM framework was recently proposed by [15]. Reformulations of the original linear and nonlinear SVM classifiers to accommodate prior knowledge about the problem were presented in [7] in the context of approximation and in [16] in the context of classifiers.

Data Mining and Integer Optimization

Data mining tasks involve, fundamentally, discrete decisions:

- How many clusters are there?
- Which class does a record belong to?
- Which features are most informative?
- Which samples capture the essential information?

Implicit enumeration techniques such as branch-and-bound were used early on to address the problem of feature selection [18].

Mathematical programming inspired by algorithms for addressing various data mining problems are now being revisited and cast as integer optimization problems. Representative formulations include feature selection using Mixed-Integer Linear Programs [11] and in [23] integer optimization models are used to address the problem of classification and regression.

Research Challenges

Numerous issues can of course be raised. However, we would like to focus on three critical aspects

i) Scalability and the curse of dimensionality. Databases are growing extremely fast and problems of practical interest are routinely composed of millions of records and thousands of features. The computational complexity is therefore expected to grow beyond what is currently reasonable and tractable. Hardware advances alone will not address this problem either as the increase in computational complexity outgrows the increase in computational speed. The challenge is therefore two-fold: either improve the algorithms and the implementation of the algorithms or explore sampling and dimensionality reduction techniques.

ii) Noise and infrequent events. Noise and uncertainty in the data is a given. Therefore, data mining algorithms in general and mathematical programming formulations in particular have to account for the presence of noise. Issues from robustness and uncertainty propagation have to be incorporated.

However, an interesting issue emerges: how do we distinguish between noise and an infrequent, albeit interesting observation? This in fact maybe a question with no answer.

iii) Interpretation and visualization. The ultimate goal of data mining is understanding the data and developing actionable strategies based on the conclusions. We need to improve not only the interpretation of the derived models but also the knowledge delivery methods based on the derived models. Optimization and mathematical programming needs to provide not just the optimal solution but also some way of interpreting the implications of a particular solution including the quantification of potential crucial sensitivities.

References

1. Blumer A, Ehrenfeucht A, Haussler D, Warmuth MK (1987) Occam's razor. Inf Process Lett 24:377–380
2. Bradley PS, Fayyad U, Mangasarian OL (1999) Mathematical programming for data mining: Formulations and challenges. INFORMS J Comput 11:217–238
3. Bradley PS, Mangasarian OL, Street WN (1997) Clustering via concave minimization. In: Mozer MC, Jordan MI, Petsche T (eds) Advances in Neural Information Processing Systems. MIT Press, Cambridge, pp 368–374
4. Breiman L, Friedman J, Olsen R, Stone C (1993) Classification and Regression Trees. Wadsworth Inc., Boca Raton
5. Dunn JC (1973) A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. J Cybern 3:32–57
6. Fayyad U, Piatetsky-Shapiro G, Smyth P (1996) From data mining to knowledge discovery: An overview. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthursamy R (eds) Advances in knowledge discovery and data mining. AAAI Press, Cambridge, pp 229–248
7. Fung GM, Mangasarian OL (2001) Proximal Support Vector Machine Classifiers In: Provost F, Srikant R (eds) Proceedings KDD-2001: Knowledge Discovery and Data Mining, San Francisco, August 26–29 2001, Asscociation for Computing Machinery, pp 77–86, ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-02.ps. Accessed 2004
8. Grossman RL, Kamath C, Kegelmeyer P, Kumar V, Namburu EE (2001) Data mining for scientific and engineering applications. Kluwer, Dordrecht
9. Hand DJ, Mannila H, Smyth P (2001) Principle of data mining. Bradford Books, Cambridge
10. Hsu C-W, Lin C-J (2002) A comparison of methods multiclass support vector machines. IEEE Trans Neural Netw 13:415–425
11. Iannatilli FJ, Rubin PA (2003) Feature selection for multiclass discrimination via mized-integer linear programming. IEEE Trans Pattern Anal Mach Learn 25:779–783
12. Krebel U (1999) Pairwise classification and support vector machines: Advances in Kernel Methods – Support Vector Learning. MIT Press, Cambridge
13. Mangasarian OL (1965) Linear and nonlinear separation of pattern by linear programming. Oper Res 31:445–453
14. Mangasarian OL (1968) Multisurface method for pattern separation. IEEE Trans Inf Theory IT-14:801–807
15. Mangasarian OL, Shavlik JW, Wild EW (2003) Knowledge-based kernel approximations. Tech. rep., Data Mining Institute, University of Wisconsin, Madison
16. Mangasarian OL, Shavlik JW, Wild EW (2004) Knowledge-based kernel approximation. J Mach Learn Res 5:1127–1141
17. Mangasarian OL, Street WN, Wolberg WH (1995) Breast cancer diagnosis and prognosis via Linear Programming. Oper Res 43:570–577
18. Narendra P, Fukunaga K (1977) A branch and bound algorithm for feature subset selection. IEEE Trans Comput 26:917–922
19. Pardalos PM, Boginski V, Vazakopoulos A (2007) Data Mining in Biomedicine. Springer, Berlin
20. Pardalos PM, Principe J (2002) Biocomputing, Kluwer, Dordrecht
21. Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco
22. Scholkopf B, Burges C, Vapnik V (1995) Extracting support data for a given task: Proc. First International Conference on Knowledge Discovery and Data Mining. AAAI Press, Cambridge, pp 252–257
23. Shioda R (2003) Integer Optimization in Data Mining. PhD thesis, MIT, Cambridge
24. Tung AK, Hou J, Han J (2001) Spatial clustering in the presence of obstacles In: Proceedings ICDE-2001: 17th International Conference on Data Engineering. Heidelberg, pp 359–367
25. Vapnik VN (1995) The nature of statistical learning. Springer, Berlin

Mathematical Programming Methods in Supply Chain Management

Laura Di Giacomo
Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università di Roma "La Sapienza", Rome, Italy

Article Outline

Introduction
Definitions
Formulation
Methods and Applications
Models
Cases
Dynamic Supply Chain Management Problem for Extraction Activities
Dynamic Supply Chain Management Problem for Finance
Conclusions
See also
References

Introduction

Supply chain management (SCM) is the integration of key business processes from end users through the original suppliers to the customers that provides products, services and information that add value to all parties [13,14,17]. It is therefore concerned with the organization, the planning and the qualitative and quantitative determination of material and information flows both in and between facilities (vendors, plants, sites and distribution centres) and between these and the final consumers. It is a set of important activities in all producing facilities and in many organizations [6].

For some restricted production problems, such as determining an optimal control to a chemical plant, suitable experimental designs can be enacted, such as **EV**olutionary **OP**eration (**EVOP**) [4], Taguchi methods [19], or more complex experimental designs such as Latin squares, Greek squares and block designs [21].

In general, verification procedures, based on experimental replication and design, cannot be used in the applied sciences, as non-reversible and unpredictable changes in the environment occur [18], and the outcome of the plans cannot be imputed to the effect of the decision taken rather than to an environmental change, so there can be no evaluation of the relevance of a formulated supply chain plan.

Thus more complex methodologies than those based on experimental verification, such as intuition experimental design or anecdotal evidence, must be posited. The solution of any SCM problem must be undertaken with respect to a set of principles and procedures to ensure the formulation of expectationally valid plans, i. e. robust valid feasible policies are determined.

To enable management to formulate good SCM plans, the methodology proposed should be analysed for its logical consistency, its statistical correctness and its adequacy. Essentially, it must be shown that from acceptable premises or axioms, by suitable deductions a policy is formulated (syntactically correctness). Since this policy cannot be tested, but only applied, it must also be shown that in many other historical derivations the policies that were formulated by this methodology turned out to be applicable (semantically adequate).

A dynamic non-linear stochastic system formulation of an SCM model must be estimated and applied. Thus an optimization algorithm must be specified and solved which determines simultaneously the adequate functional form, its parameterization and the optimal control [6].

Definitions

In this section some fundamental definitions will be given.

A dynamical system is a precise mathematical object [16], and given the flows of the activities of the phenomenon, the input-output relationships must be determined by appropriate estimation methods.

Not every relationship can be modelled by mathematical system theory, since a representation which is non-anticipatory is required [16], while the condition that the functionals be sufficiently smooth which was previously required may be waived.

Dynamical systems have been defined at a high level of generality to refine concepts and perceive unity in a diversity of applications, and by appropriate modelling whole hierarchies of phenomena can be represented as systems defined at different levels.

Definition 1 ([16]) A dynamical system is a composite mathematical object defined by the following axioms:

1. There is a given time set T, a state set X, a set of input values U, a set of acceptable input functions $\Omega = \omega \colon \Omega \to U$, a set of output values Y and a set of output functions $\Gamma = \gamma \colon \Gamma \to Y$.
2. (Direction of time). T is an ordered subset of the reals.
3. The input space Ω satisfies the following conditions:
   (a) (Non-triviality). Ω is non-empty.

(b) (Concatenation of inputs). An input segment $\omega_{(t_1,t_2]}$, $\omega \in \Omega$ restricted to $(t_1, t_2] \cap T$. If $\omega, \omega' \in \Omega$ and $t_1 < t_2 < t_3$, there is an $\omega'' \in \Omega$ such that $\omega''_{(t_1,t_2]} = \omega_{(t_1,t_2]}$ and $\omega''_{(t_2,t_3]} = \omega'_{(t_2,t_3]}$.

4. There is a state transition function $\varphi\colon T \times T \times X \times \Omega \to X$ whose value is the state $x(t) = \varphi(t; \tau, x, \omega) \in X$ resulting at time $t \in T$ from the initial state $x = x(\tau) \in X$ at the initial time $\tau \in T$ under the action of the input $\omega \in \Omega$. ϕ has the following properties:
   (a) (Direction of time). ϕ is defined for all $t \geq \tau$, but not necessarily for all $t < \tau$.
   (b) (Consistency). $\varphi(t; t, x, \omega) = x$ for all $t \in T$, all $x \in X$ and all $\omega \in \Omega$.
   (c) (Composition property). For any $t_1 < t_2 < t_3$ there results:

$$\varphi(t_3; t_1, x, \omega) = \varphi(t_3; t_2, \varphi(t_2; t_1, x, \omega), \omega)$$

   for all $x \in X$ and all $\omega \in \Omega$.
   (d) (Causality). If $\omega, \omega' \in \Omega$ and $\omega_{(\tau,t]} = \omega'_{(\tau,t]}$ then $\varphi(t; \tau, x, \omega) = \varphi(t; \tau, x, \omega')$.
5. There is a given readout map $\eta\colon T \times X \to Y$ which defines the output $y(t) = \eta(t, x(t))$. The map $(\tau, t] \to Y$ given by $\sigma \mapsto \eta(\sigma, \varphi(\sigma, \tau, x, \omega))$, $\sigma \in (\tau, t]$, is an output segment, that is the restriction $\gamma_{(\tau,t]}$ of some $\gamma \in \Gamma$ to $(\tau, t]$.

The following mathematical structures in Definition 1 will be indicated by:

- The pair (t, x), $t \in T$, $x \in X$ $\forall t$ is called an event;
- The state transition function $\varphi(x_t, u_t)$ is called a trajectory.

Phenomena may also be modelled through dynamical systems in the input/output sense, which reflect an experimental design or a simulative approach, long applied in science.

Definition 2 A dynamical system in an input/output sense is a composite mathematical object defined as follows:

1. There are given sets T, U, Ω, Y and Γ satisfying all the properties required by Definition 1.
2. There is a set A indexing a family of functions

$$\mathcal{F} = \{f_\alpha\colon T \times \Omega \to Y, \alpha \in A\};$$

   each member of $\mathcal{F}$ is written explicitly as $f_\alpha(t, \omega) = y(t)$, which is the output resulting at time t from the input ω under the experiment α. Each f_α is called an input/output function and has the following properties:
   (a) (Direction of time). There is a map $\iota\colon A \to T$ such that $f_\alpha(t, \omega)$ is defined for all $t \geq \iota(\alpha)$.
   (b) (Causality). Let $\tau, t \in T$ and $\tau < t$. If $\omega, \omega' \in \Omega$ and $\omega_{(\tau,t]} = \omega'_{(\tau,t]}$, then $f_\alpha(t, \omega) = f_\alpha(t, \omega')$ for all α such that $\tau = \iota(\alpha)$.

While the input/output approach may determine a family of functions, which generally vary over the time interval of realization and across instances, the state-space approach represents the trajectories in the way indicated, through a unique function. The latter approach is intuitively more appealing, especially in applications.

The representations are equivalent. It is easy to transform a given system from a state space formulation into an input/output formulation and vice versa [2,16], so each may be used as convenience suggests.

It cannot be assumed generally that a dynamical system satisfies the conditions of smoothness, nor that it will meet the necessary and sufficient conditions for an optimal control to exist. Thus in general, the dynamical systems to be dealt with may have an awkward structure, but through the combined estimation and optimization approach a sufficiently good approximation may be obtained with the required characteristics [6].

A sufficiently general representation of a dynamical system may be formulated by applying Definition 1, recalling the equivalence of an input/output system and a system in state form:

$$x_{t+1} = \varphi(x_t, u_t)\,, \tag{1}$$

$$y_t = \eta(x_t)\,, \tag{2}$$

where $x_t \in X \subseteq R^r$ may simply be taken as an r-dimensional vector in a Euclidean space X, indicating the state of the system at time t, $u_t \in U \subseteq R^q$ may be taken as a q-dimensional vector in a Euclidean subspace U of control variables and $y_t \in Y \subseteq R^p$ is a p-dimensional vector in a Euclidean space Y of output variables, in line with Definitions 1 and 2.

The definition of a dynamical system is based on defining an intermediary set of states and a transition function or a family of functions. Neither of these constructions is unique, so if it is desired to represent

a SCM system by such structures, equivalence of the possible structures must be shown.

Definition 3 Given two states x_{t_0} and $\hat{x}_{t_0}$ belonging to systems S and $\hat{S}$ which may not be identical but have a common input space Ω and output space Y, the two states are said to be equivalent if and only if for all input segments $\omega_{[t_0,t)} \in \Omega$ the response segment of S starting in state x_{t_0} is identical with the response segment of $\hat{S}$ starting in state $\hat{x}_{t_0}$; that is

$$x_{t_0} \cong \hat{x}_{t_0} \Leftrightarrow \eta(t, \varphi(x_{t_0}, \omega_{[t_0,t)})) = \hat{\eta}(t, \hat{\varphi}(\hat{x}_{t_0}, \omega_{[t_0,t)})) \quad \forall t \in T, t_0 \leq t, \forall \omega_{[t_0,t)} \in S, \hat{S} . \tag{3}$$

Systems S and $\hat{S}$ may be two models of a SCM system solved with different control policies, or they may be various alternative models of the phenomenon.

Definition 4 A system is in reduced form if there are no distinct states in its state space which are equivalent to each other.

Definition 5 Systems S and $\hat{S}$ are equivalent $S \equiv \hat{S}$ if and only if to every state in the state space of S there corresponds an equivalent state in the state space of $\hat{S}$ and vice versa.

Some important conditions are required to make the representation of the SCM adequate.

The conditions of the system are:

- Reachability
- Controllability
- Observability
- Stability

These conditions are very important since they allow trajectories to be defined, the initial point of trajectories to be determined and their stability properties to be derived. Moreover they can be applied at any moment in time to determine if the goals of the SCM are still attainable and at what cost. Reachability, controllability and stability are seldom formally examined and yet at every period exogenous events can arise to nullify even the best formulated plan, so these are important instruments for SCM [6].

An important property which distinguishes dynamical systems from their counterparts derived in comparative statics is the distinction between systems which are simply equivalent and those which are multiply equivalent [6]. This distinction is crucial if dynamical systems are considered, while with comparative static models the distinction does not apply. This is one of the many reasons that one should insist on solving SCM dynamic estimation problems with a data-driven formulation [6].

The dynamical system representation of a SCM system permits one to verify its specification, whether the optimal control which determines the final event is reachable, if the system is controllable throughout the sequence of events comprising the trajectory, if the system is observable and finally if the given solution is stable, so that small perturbations will not give rise to explosive perturbations or to chaotic behaviour. In so doing crucial questions which are important to management can be answered.

If these conditions are not verified, this will suggest strategic changes to the SCM system or profound modification of policies, aspects which are difficult to determine in advance.

Computationally, these aspects are handled by adding appropriate constraints in the mathematical program [6].

Formulation

Consider the monitoring of a set of activities in time of a supply chain at a given level of aggregation, which may be at the department, plant or firm level, or a hierarchical system developed through all these organizational structures. Although the accuracy of the representation may depend on the sampling strategy and the time interval, these aspects will not be considered here.

Thus a given finite-dimensional estimation and optimization problem will be considered which may well be non-linear and dynamic.

Consider the data set of a phenomenon consisting of measurements (y^t, x^t, u^t) over $(t = 1, 2, \ldots, T)$ periods, where it is assumed that $y_t \in R^p$ is a p-dimensional vector, while $x_t \in R^r$ is an r-dimensional vector of explanatory or state variables of the dynamic process of dimension. Also, u_t is a q-dimensional vector of control variables. It is desired to determine functional forms $\varphi: R^{r+q} \to R^r$ and $\eta: R^r \to R^p$ and a set of suitable coefficients $\Theta \in R^m$ such that:

$$\text{Min} \quad J = \sum_{t=T+1}^{\mathcal{T}} c(x_t, u_t, y_t), \tag{4}$$

$$x^{t+1} = \varphi(x^t, u^t, y^t, w^t) \quad \forall t = T+1, \ldots, \mathcal{T}-1, \quad (5)$$

$$y^t = \eta(x^t, u^t, v^t) \quad \forall t = T+1, \ldots, \mathcal{T}, \quad (6)$$

where w^t and v^t are stochastic processes also to be determined.

Equation (4) is the objective function for the supply chain and (5) and (6) are the system equations in state space formulation and a similar representation may be adopted for the the input-output formulation [16,20].

The system (4)–(6) could be estimated by a maximum likelihood method so as to minimize the random errors, indicated by $w^t \in R^r$ and $v^t \in R^p$, such that they will have minimum variance and zero mean value, and then on the quantified model the optimal control problem could be solved, usually through an appropriate optimization problem.

However, for this type of model with serially correlated disturbances, which are also correlated with the control variables, its estimation will be biased and the necessary least-squares properties to ensure an asymptotically correct estimate may only be fulfilled in exceptional cases. Thus the two-stage approach, indicated above, is inappropriate [15].

It is important to apply a suitable data-driven statistical method to determine the most appropriate statistical form and the most precise values of the parameters, as when implemented correctly with regard to an accurately specified functional form. Such a method will provide estimates of parameters that satisfy the statistical properties [1,18].

Suppose that all the statistical properties that a given estimate must fulfil are set up as constraints to the maximum likelihood problem to be solved; then the parameters are defined implicitly by this optimization problem, which can be inserted into the optimal control system for policy determination, so that statistically correct estimates will always result. Thus the solution yielding the best policy can be chosen, where $T+1, \ldots, \mathcal{T}$ is the forecast period, by solving an optimization formulation of this complex problem. By recursing on the specifications, i. e. by changing the functional form, increasingly better fits can be obtained. At each iteration, the best combination of parameterization and policy is obtained.

The unknowns to be determined are the input and output variables considered and the parameters of the functional form specified in the current iteration, indicated as $\Theta = \{\theta_1, \theta_2\} \subset R^m$, respectively for (5) and (6). Note that m may be much larger than $2r + q + p + 1$, the number of variables present in each system, since the system is non-linear.

The mathematical program will be formulated with respect to the residual variables, but it is immediate that for a given functional form, the unknown parameters will be specified and thus the unknowns of the problem will also be defined and available. Thus the mathematical program is fully specified for each functional form to be considered.

Using the notation given above, the residual terms are given from Eqs. (5) and (6) as:

$$w_i = \hat{x}_{i+1} - \varphi(\hat{x}_i, \hat{u}_i, \hat{y}_i : \theta_1) \quad i = 1, 2, \ldots, N, \quad (7)$$

$$v_i = \hat{y}_{i+1} \eta(x_i, u_i, v_i : \theta_2) \quad i = 1, 2, \ldots, N, \quad (8)$$

where $\hat{\cdot}$, as usual, indicates the historical values of a variable, and thus suitable values of θ_1 and θ_2 must be determined by the mathematical program such that all the constraints expressed in terms of $w_i, v_i \forall i$ are specified.

Methods and Applications

Given an experimental data set obtained as a set of measurements of the operation of a phenomenon, it is desired to determine a suitable representation of it in the form of a model, so as to determine a suitable control law for the model which can then be extended to the phenomenon and thus obtain a better performance [3].

Except in some simple cases, the representation assumed by the model and the data that have been collected will condition the results obtainable by enacting the control law. For models that are non-linear in the parameters, the interaction between the estimation of these and the determination of an optimal control is much more complex than the linear case requiring the solution of constrained optimization problems which will determine simultaneously the best estimates and the optimal control.

Consider the availability of a given data set containing a number of sets of time series data or cross-sectional data. To determine from these data a suitable model, a functional form must be selected and a set of suitable parameters must be estimated which will satisfy

all the conditions on the model and permit the determination of a suitable set of control variables, which will define an optimal control with respect to a predefined merit function.

Thus from the data set it is desired to derive a sufficiently accurate model of the phenomenon, which can then be used in control and in prediction.

Statistical estimation methods are important because, when implemented correctly with regard to an accurately specified functional form, they will provide estimates of parameters that have the following properties [1,18]:

1. The parameter estimates are unbiased.
   - As the size of the data set grows larger, the estimated parameters tend to their true values.
2. The parameter estimates are consistent, which will then satisfy the following conditions:
   - The estimated parameters are aymptotically unbiased.
   - The variance of the parameter estimate must tend to zero as the data set tends to infinity.
3. The parameter estimates are asymptotically efficient.
   - The estimated parameters are consistent.
   - The estimated parameters have smaller asymptotic variance as compared to any other consistent estimator.
4. The residuals have minimum variance, which is ensured by the following factors:
   - The variance of the residuals must be minimum.
   - The residuals must be homoscedastic.
   - The residuals must not be serially correlated.
5. The residuals are unbiased (have zero mean).
6. The residuals have a non-informative distribution (usually, a Gaussian distribution).
   - If the distribution of the residuals is informative, the extra information could somehow be obtained, reducing the variance of the residuals, their bias etc., with the result that better estimates are obtained.

In short, through correct implementation of statistical estimation techniques the estimates are as close as possible to their true values, all the information that is available is applied and the uncertainty surrounding the estimates and the data fit is reduced to the maximum extent possible. Thus the estimates of the parameters, which satisfy all these conditions, are the 'best' possible in a 'technical' sense [1].

To ensure that all the statistical properties which the given estimates of the residuals must fulfil are satisfied at every iteration, instead of solving an unconstrained maximum likelihood or least-squares problem [15], the required statistical properties of the estimates are set up as constraints, together with the specification of the model of the phenomenon, and this global optimization problem is solved for all the undetermined variables.

The parameters of this model to be estimated are defined implicitly through those constraints which define the statistical conditions. On solving the global optimization problem, the parameter estimates that result will be defined for the optimal control system for the policy determination so that statistically correct estimates will always result.

The procedure adopted can be specified easily by using the same notation as above and by adding an additional set of constraints which express the statistical conditions that must be satisfied by the estimates.

Let

$$\gamma(x_{i+1}, x_i, u_i, y_{i+1}, y_i, w_i, v_i, \theta_1, \theta_2) \geq 0 \qquad i = 1, 2, \ldots, N, N+1, \ldots, \mathcal{T} \quad (9)$$

be the set of conditions to be satisfied to obtain estimates, if they exist, which satisfy the statistical properties indicated above. Then the optimization problem to be solved is:

$$\text{Min} \quad J = \sum_{i=N+1}^{\mathcal{T}} c(x_i, u_i, y_i), \quad (10)$$

$$x_{i+1} = \varphi(x_i, u_i, y_i, w_i : \theta_1) \quad i = 1, 2, \ldots, \mathcal{T}, \quad (11)$$

$$y_{i+1} = \eta(x_i, u_i, v_i : \theta_2) \quad i = 1, 2, \ldots, \mathcal{T}, \quad (12)$$

$$0 \leq \gamma(x_{i+1}, x_i, u_i, y_{i+1}, y_i, w_i, v_i, \theta_1, \theta_2) \qquad i = 1, 2, \ldots, \mathcal{T}. \quad (13)$$

Thus the solution yielding the best policy can be chosen by solving an optimization formulation of this complex problem. By recursing on the specifications, i. e. by changing the functional form, and increasing the number of independent variables considered, increasingly better fits can be obtained, with regard to both the historical data and the predicted optimal control policy.

Models

Many models of industrial, extractive and financial activities require the integration of key processes, but the most essential aspect is to formulate precise information where it is most needed [11,12].

Many optimization models solve satisfactorily supply chain problems, but apparently no model except this one integrates information and allocation of goods and operations dynamically.

This algorithm instead solves the combined problem, as has been shown elsewhere [6], while a theory-driven modelling approach to the problem, using models consisting of two stages, an identification stage and an optimization stage, can be shown to be dominated by this data-driven approach.

At present, this seems to be the only viable approach to solving such complex problems.

Cases

Some non-typical SCM problems are indicated here: dynamical supply chain management problems for perforation oil wells and for finance. Industrial SCM models are given elsewhere [3,8,9,10].

Dynamic Supply Chain Management Problem for Extraction Activities

The perforation of oil wells consists of a number of operations to drive the bit head lower and lower while ensuring normal functions on the equipment and the operations. To this end complex measurements are executed by software systems indicated as mudlogging systems. These measurements are designed to assist the operator in controlling the perforation rate of the bit head by monitoring a number of crucial operations periodically.

The settings of some of these operations affect the rate of perforation, and therefore it is considered extremely useful to dispose of measurements of these variables and have predictions over the next few periods of the possible advancement of the bit head, or of the rate of perforation, and so enable an optimal control of the process to be formulated [5].

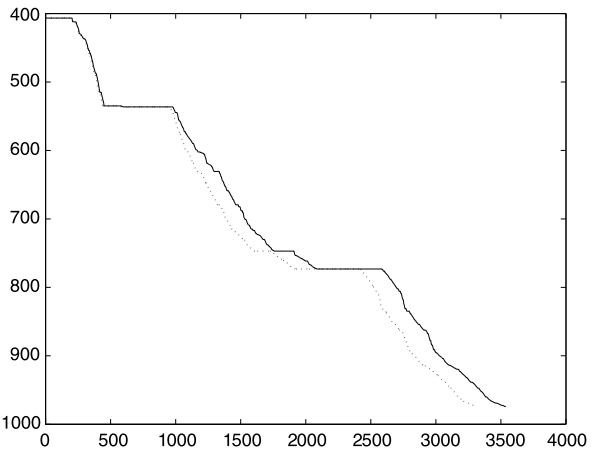
It should be mentioned that periodically the drilling process must be halted so that the boring can be lined with suitable materials. Also, one of the most important elements of the process is to keep circulating around the bit-head assembly a concentration of mud lubricants, indicated as mud, which gives the name to the measuring process. Recall that all these flows and operations occur in time, so it is considered crucial to specify dynamic models, unless it is desired to determine the steady-state rates of the eventual process.

In fact oil drilling processes can be considered as complex supply chain systems with many phases and many operations.

The determination of optimal control policies in processes for the extraction of oil from underground require that they be formulated as formal procedures, which are syntactically correct and semantically adequate, so as to permit management to make the necessary investments, not on hearsay or clever promotional activities, but on the basis of rational knowledge and confidence in the application.

Figure 1 shows an optimal SCM plan compared to the actual historical plan implemented. The predicted trajectory is superimposed on the actual time path of the perforation process, thus respecting all the interruptions and periods of halting.

In Table 1, six instances to determine optimal controls are indicated, and each entry reflects the drilling experience of the given well for that week with regard to the given period. From the active perforation intervals an intial period was selected randomly and the optimal



Mathematical Programming Methods in Supply Chain Management, Figure 1
Example of drilling for oil: real-time path (*continuous*) and optimal control path (*dashed*) for the well

Mathematical Programming Methods in Supply Chain Management, Table 1
Optimal predicted versus actual increment for 6 oil wells over 192 periods (8 h) in metres

Well and week	Optimal increment	Real increment	% difference
FT02D 9	114.0	73.3	35.70
FT02D 16	116.7	83.8	28.19
FT02D 23	73.7	13.45	81.75
GX01 3	94.8	72.2	23.84
GX01 11	57.9	18.8	67.53

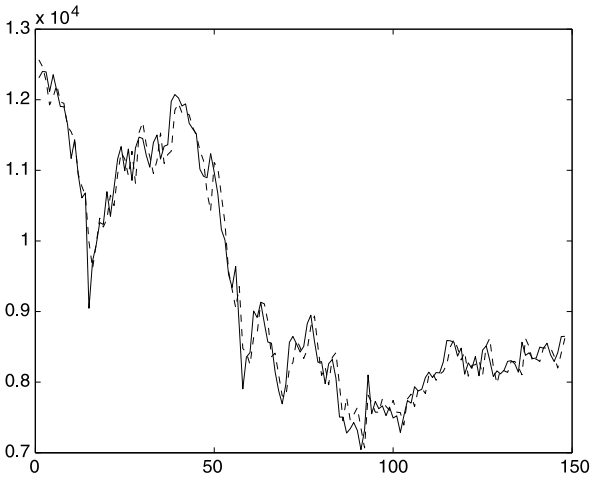
control was defined for the next 192 periods (8 h). The average predicted increment in depth attainable over the actual one was more than 30% on average.

Dynamic Supply Chain Management Problem for Finance

The prediction of future quotations on stock exchange indices is important and consists of the basic instrument to handle financial supply chain management systems. A financial supply chain system must consider many types of financial intermediaries, many types of stocks and stock indices and many types of operations. Further, there are many possibilities for managing the monetary holdings, so that a full SCM system is envisaged as defined above [7].

Consider the Dow-Jones Industrial Average (DJIA) stock exchange index over a period of 3 years starting in April 2001, as shown in Fig. 2, where the continuous line indicates the actual quotations, week by week over the period, while the 1-week-ahead predictions are given by the dashed line. As can be easily seen, the two curves almost coincide, which implies that the predictions 1 week ahead are very good.

Instead, in Table 2 a period of 5 weeks is considered from April 16, 2004 to May 14, 2004. The quotations are given every Friday evening at closing time, while the predictions are made on Fridays just after closing time. Thus on April 9 predictions were made for the next 5 weeks, as indicated in the second row of the table. After closing on April 16, 2004, predictions were made for 4 weeks only and are depicted in the third row of the table and so on for the subsequent weeks. Finally, in the last row the closing quotations for the week are given.



Mathematical Programming Methods in Supply Chain Management, Figure 2
Weekly time series of the Dow-Jones Industrial Average

Mathematical Programming Methods in Supply Chain Management, Table 2
Results for prediction of the Dow-Jones Industrial Average, 147 periods

Period	16/4	23/4	30/4	7/5	14/5
9/4	8652.86	8568.44	9304.81	13306.5	9958.15
16/4	..	8646.28	8552.54	11514.3	11000.3
23/4	..	..	8820.73	8806.51	4700.47
30/5	..	..	..	8518.12	8361.19
7/5	..	..	..	..	8343.35
Index	8712.88	8855.03	8538.03	8505.54	8432.25

This table allows one to determine with the appropriate portfolio model suitable financial policies to formulate optimal financial SCM plans [7].

Conclusions

Optimal dynamic SCM policies may be obtained by a correct application of statistical inference and mathematical programming techniques.

It has been indicated that these policies are expectationally valid, which implies that they are syntactically correct and semantically adequate.

Computational evidence has been presented and indicated in the references.

See also

- ► Generalizations of Interior Point Methods for the Linear Complementarity Problem
- ► Simultaneous Estimation and Optimization of Nonlinear Problems

References

1. Amemiya T (1985) Advanced Econometrics. Blackwell, Oxford
2. Aoki M (1976) Optimal Control and System Theory in Dynamic Economic Analysis. North-Holland, New York
3. Bilardo U, Di Giacomo L, Patrizi G (2007) Dynamic management of petroleum drilling operations. Submitted, June 2007, pp 1–35
4. Box GEP, Draper NR (1969) Evolutionary Operation: A Statistical Method for Process Improvement. Wiley, New York
5. Bradley HB (1987) Petroleum Engineering Handbook. Society of Petroleum Engineers, Richardson, TX
6. Di Giacomo L, Patrizi G (2006) Dynamic nonlinear modelization of operational supply chain systems. J Global Optim 34:503–534
7. Di Giacomo L, Patrizi G (2006) Optimal dynamic nonlinear prediction methods for management of financial instruments. Technical report, Dipartimento di Statistica, Probabilitá e Statistiche Applicate, Universitá di Roma, La Sapienza, Rome
8. Di Giacomo L, Patrizi G (2007) Methodological analysis of supply chain management applications: towards a normative theory of supply chain management. Submitted for publication, copy at: http://banach.sta.uniroma1.it/patrizi/, pp 1–41
9. Di Giacomo L, Patrizi G (2007) Optimal control of well drilling operations. Submitted, pp 1–31
10. Di Giacomo L, Patrizi G, Di Lena E, Pomaranzi L, Sensi F (2008) C.A.s.S.a.n.D.r. A computerized analysis for supply chain distribution activity. In: Bertazzi L, Speranza MG, van Nunen J (eds) Lectures Notes in Economics ans Mathematical Systems. Springer, Berlin
11. Eksioglu SD, Romeijn HE, Pardalos PM (2006) Cross-facility management of production and transportation planning problem. Comput Oper Res 33:3231–3251
12. Geunes J, Akçali E, Pardalos P, Romeijn H, Shen Z-J (2005) Applications of Supply Chain Management and E-Commerce Research. Springer, New York
13. Geunes J, Pardalos P (2005) Supply Chain Optimization. Springer, New York
14. Geunes J, Pardalos P, Romejin H (2005) Supply Chain Management: Models, Applications, and Research Directions. Springer, New York
15. Jennrich RI (1969) Asymptotic properties of non-linear least squares estimators. Ann Math Statist 40:633–643
16. Kalman RE, Falb PL, Arbib MA (1969) Topics in Mathematical System Theory. McGraw-Hill, New York
17. Lambert DM, Cooper MC, Pagh JD (1998) Supply chain management: Implementation issues and research opportunities. Int J Logist Manag 9:1–17
18. Malinvaud E (1978) Méthodes Statistiques de l'économétrie. Dunod, Paris
19. Ross PJ (1988) Taguchi techniques for quality engineering: loss function, orthogonal experiments, parameter and tolerance design. McGraw-Hill, New York
20. Söderström T, Stoica P (1989) System Identification. Prentice-Hall, New York
21. Vajda S (1967) Mathematics of Experimental Design: incomplete block designs and Latin squares. Griffin, London

Matrix Completion Problems

MONIQUE LAURENT
CWI, Amsterdam, The Netherlands

MSC2000: 05C50, 15A48, 15A57, 90C25

Article Outline

Keywords
Introduction
Positive Semidefinite Completion Problem
Euclidean Distance Matrix Completion Problem
Completion to Completely Positive and Contraction Matrices
Rank Completions
See also
References

Keywords

Partial matrix; Completion of matrices

Matrix completion problems are concerned with determining whether partially specified matrices can be completed to fully specified matrices satisfying certain prescribed properties. In this article we survey some results and provide references about these problems for the following matrix properties: positive semidefinite matrices, Euclidean distance matrices, completely positive matrices, contraction matrices, and matrices of given rank. We treat mainly optimization and combinatorial aspects.

Introduction

A *partial matrix* is a matrix whose entries are specified only on a subset of its positions; a *completion* of a partial matrix is simply a specification of the unspecified entries. *Matrix completion problems* are concerned with determining whether or not a completion of a partial matrix exists which satisfies some prescribed property. We consider here the following matrix properties: positive (semi) definite matrices, distance matrices, completely positive matrices, contraction matrices, and matrices of given rank; definitions are recalled below.

In what follows, x^*, A^* denote the conjugate transpose (in the complex case) or transpose (in the real case) of vector x and matrix A. A square real symmetric or complex Hermitian matrix A is *positive semidefinite* (psd) if $x^*Ax \geq 0$ for all vectors x and *positive definite* (pd) if $x^*Ax > 0$ for all vectors $x \neq 0$; then we write: $X \succcurlyeq 0$ ($X \succ 0$). Equivalently, A is psd (respectively, pd) if and only if all its eigenvalues are nonnegative (respectively, positive) and A is psd if and only if $A = BB^\intercal$ for some matrix B. A matrix A is said to be *completely positive* if $A = BB^\intercal$ for some nonnegative matrix B. An $n \times n$ real symmetric matrix $D = (d_{ij})$ is a *Euclidean distance matrix* (abbreviated as *distance matrix*) if there exist vectors $v_1, \ldots, v_n \in \mathbf{R}^k$ (for some $k \geq 1$) such that, for all $i, j = 1, \ldots, n$, d_{ij} is equal to the square of the Euclidean distance between v_i and v_j. Finally, a (rectangular) matrix A is a *contraction matrix* if all its singular values (that is, the eigenvalues of A^*A) are less than or equal to 1.

The set of positions corresponding to the specified entries of a partial matrix A is known as the *pattern* of A. If A is an $n \times m$ partial matrix, its pattern can be represented by a bipartite graph with node bipartition $[1, n] \cup [1, m]$ having an edge between nodes $i \in [1, n]$ and $j \in [1, m]$ if and only if entry a_{ij} is specified.

When asking about existence of a psd completion of a partial $n \times n$ matrix A, it is commonly assumed that all diagonal entries of A are specified (which is no loss of generality if we ask for a pd completion); moreover, it can obviously be assumed that A is *partial Hermitian*, which means that entry a_{ji} is specified and equal to $a_{ij}*$ whenever a_{ij} is specified. Hence, in this case, complete information about the pattern of A is given by the graph $G = ([1, n], E)$ with node set $[1, n]$ and whose edge set E consists of the pairs ij ($1 \leq i < j \leq n$) for which a_{ij} is a specified entry of A. The same holds when dealing with distance matrix completions (in which case diagonal entries can obviously be assumed to be equal to zero).

An important common feature of the above matrix properties is that they possess an 'inheritance structure'. Indeed, if a partial matrix A has a psd (pd, completely positive, distance matrix) completion, then every principal specified submatrix of A is psd (pd, completely positive, a distance matrix); similarly, if a partial matrix A admits a completion of rank $\leq k$, then every specified submatrix of A has rank $\leq k$. Hence, having a completion of a certain kind imposes certain 'obvious' necessary conditions. This leads to asking which are the patterns for the specified entries that insure that if the obvious necessary conditions are met, then there will be a completion of the desired type; therefore, this introduces a combinatorial aspect into matrix completion problems, as opposed to their analytical nature.

In this article we survey some results and provide references for the various matrix completion problems mentioned above, concerning optimization and combinatorial aspects of the problems. See [32,47] for more detailed surveys on some of the topics treated here.

Positive Semidefinite Completion Problem

We consider here the following *positive (semi) definite completion problem* (PSD): Given a partial Hermitian matrix $A = (a_{ij})_{ij \in S}$ whose entries are specified on a subset S of the positions, determine whether A has a psd (or pd) completion; if, yes, find such a completion. (Here, S is generally assumed to contain all diagonal positions.)

This problem belongs to the most studied matrix completion problems. This is due, in particular, to its many applications, e. g., in probability and statistics, systems engineering, geophysics, etc., and also to the fact that positive semidefiniteness is a basic property which is closely related to other matrix properties like being a contraction or distance matrix. Equivalently, (PSD) is the problem of testing feasibility of the following system (in variable $X = (x_{ij})$):

$$X \succeq 0, \qquad x_{ij} = a_{ij} \quad (ij \in S). \tag{1}$$

Therefore, (PSD) is an instance of the following *semidefinite programming problem* (P): Given Hermitian matrices $A_1, \ldots, A_m$ and scalars $b_1, \ldots, b_m$, decide

whether the following system is feasible:

$$X \succeq 0, \qquad A_j \cdot X = b_j \quad (j = 1, \ldots, m) \tag{2}$$

(where $A \cdot X := \sum_{i,j=1}^{n} a_{ij} * x_{ij}$ for two Hermitian $(n \times n)$-matrices A and X).

The exact complexity status of problems (PSD) and (P) is not known; in particular, it is not known whether they belong to the complexity class *NP*. However, it is shown in [60] that (P) is neither *NP*-complete nor co-*NP*-complete if $NP \neq$ co-*NP*. However, the semidefinite programming problem and, thus, problem (PSD) can be solved with an arbitrary precision in polynomial time. This can be done using the ellipsoid method (since one can test in polynomial time whether a rational matrix A is positive semidefinite and, if not, find a vector x such that $x^*Ax < 0$; cf. [24]), or interior point methods (cf. [3,27,56]). There has been a growing interest in semidefinite programming in the recent years (1994), which is due, in particular, to its successful application to the approximation of hard combinatorial optimization problems (cf. the survey [20]). This has prompted active research on developing interior point algorithms for solving semidefinite programming problems; the literature is quite large, see [64,65] for extensive information. Numerical tests are reported in [34] where an interior point algorithm is proposed for the approximate psd completion problem; it permits to find exact completions for random instances up to size 110.

Moreover, it is shown in [59] that problem (P) can be solved in polynomial time (for rational input data A_j, b_j) if either the number m of constraints, or the order n of the matrices X, A_j in (2) is fixed (cf. also [9]). Moreover, under the same assumption, one can test in polynomial time the existence of an integer solution and find one if it exists [39].

Call a partial Hermitian matrix A *partial psd* (respectively, *partial pd*) if every principal specified submatrix of A is psd (respectively, pd). As mentioned in the Introduction, being partial psd (pd) is an obvious necessary condition for A to have a psd (pd) completion. In general, this condition is not sufficient; for instance, the partial matrix:

$$A = \begin{pmatrix} 1 & 1 & ? & 0 \\ 1 & 1 & 1 & ? \\ ? & 1 & 1 & 1 \\ 0 & ? & 1 & 1 \end{pmatrix}$$

('?' indicates an unspecified entry) is partial psd, yet no psd completion exists; note that the pattern of A is a circuit of length 4. Call a graph *chordal* if it does not contain any circuit of length ≥ 4 as an induced subgraph; chordal graphs occur in particular in connection with the Gaussian elimination process for sparse pd matrices (cf. [21,61]). (An *induced subgraph* of a graph $G = (V, E)$ being of the form $H = (U, F)$ where $U \subseteq V$ and $F := \{ij \in E: i, j \in U\}$.) It is shown in [23] that every partial psd matrix with pattern G has a psd completion if and only if G is a chordal graph; the same holds for pd completions. This extends an earlier result from [16] which dealt with 'block-banded' partial matrices; in the Toeplitz case (all entries equal along a band), one finds the classical Carathéodory–Fejér theorem from function theory.

The proof from [23] is constructive and can be turned into an algorithm with a polynomial running time [48]. Moreover, it is shown in [48] that (PSD) can be solved in polynomial time when restricted to partial rational matrices whose pattern is a graph having a fixed minimum fill-in; the *minimum fill-in* of a graph being the minimum number of edges needed to be added in order to obtain a chordal graph. This result is based on the above mentioned results from [39,59] concerning the polynomial time solvability of (integer) semidefinite programming with a fixed number m of linear constraints in (2).

The result from [23] on psd completions of partial matrices with a chordal pattern has been generalized in various directions; for instance, considering general inertia possibilities for the completions ([17,35]), or considering completions with entries in a function ring [37].

If A is a partial matrix having a pd completion, then A has a unique pd completion with maximum determinant (this unique completion being characterized by the fact that its inverse has zero entries at all unspecified positions of A) [23]. In the case when the pattern of A is chordal, explicit formulas for this maximum determinant are given in [7]. The paper [52] considers the more general problem of finding a maximum determinant psd completion satisfying some additional linear constraints.

Further necessary conditions are known for the existence of psd completions. Namely, it is shown in [8] that if a partial matrix $A = (a_{ij})$ with pattern G and di-

agonal entries equal to 1 is completable to a psd matrix, then the associated vector $x := (\arccos(a_{ij})/\pi)_{ij \in E}$ satisfies the inequalities:

$$\sum_{e \in F} x_e - \sum_{e \in C \setminus F} x_e \leq |F| - 1 \quad \text{for all } F \subseteq C, \quad C \text{ circuit in } G, \ |F| \text{ odd.} \tag{3}$$

Moreover, any partial matrix with pattern G satisfying (3) is completable to a psd matrix if and only if G does not contain a homeomorph of K_4 as an induced subgraph (then, G is also known as *series-parallel graph*) [44]. (Here, K_4 denotes the complete graph on 4 nodes and a *homeomorph* of K_4 is obtained by replacing the edges of K_4 by paths of arbitrary length.) The patterns G for which every partial psd matrix satisfying (3) has a psd completion are characterized in [6]; they are the graphs G which can be made chordal by adding a set of edges in such a way that no new clique of size 4 is created. Although (3) can be checked in polynomial time for rational x [5], the complexity of problem (PSD) for series-parallel graphs (or for the subclass of circuits) is not known. A strengthening of condition (3) (involving cuts in graphs) is formulated in [44].

Another approach to problem (PSD) is considered in [1,28], which is based on the study of the cone

$$\mathcal{P}_G := \left\{ X = (x_{ij})_{i,j \in V} : \begin{array}{c} X \succeq 0, \ x_{ij} = 0 \\ \forall i \neq j, \quad ij \notin E \end{array} \right\}$$

associated to graph $G = (V, E)$. Indeed, it is shown there that a partial matrix A with pattern G has a psd completion if and only if

$$\sum_{i \in V} a_{ii} x_{ii} + \sum_{\substack{i \neq j, \\ ij \in E}} a_{ij} x_{ij} \geq 0, \quad \forall X \in \mathcal{P}_G. \tag{4}$$

Obviously, it suffices to check (4) for all X *extremal* in $\mathcal{P}_G$ (i. e., X lying on an extremal ray of the cone $\mathcal{P}_G$).

Define the *order* of G as the maximum rank of an extremal matrix in $\mathcal{P}_G$. The graphs of order 1 are precisely the chordal graphs [1,58] and the graphs of order 2 have been characterized in [46]. One might reasonably expect that problem (PSD) is easier for graphs having a small order. This is indeed the case for graphs of order 1; the complexity of (PSD) remains however open for the graphs of order 2 (partial results are given in [48]).

Euclidean Distance Matrix Completion Problem

We consider here the *Euclidean distance matrix completion problem* (abbreviated as *distance matrix completion problem*) (EDM): Given a graph $G = (V = [1, n], E)$ and a real partial symmetric matrix $A = (a_{ij})$ with pattern G and with zero diagonal entries, determine whether A can be completed to a distance matrix; that is, whether there exist vectors $v_1, \ldots, v_n \in \mathbf{R}^k$ for some $k \geq 1$ such that

$$a_{ij} = \|v_i - v_j\|^2 \quad \text{for all } ij \in E. \tag{5}$$

(here, $\|v\| = \sqrt{\sum_{h=1}^k v_h^2}$ denotes the Euclidean norm of $v \in \mathbf{R}^k$.) The vectors $v_1, \ldots, v_n$ are then said to form a *realization* of A. A variant of problem (EDM) is the *graph realization problem* (EDMk), obtained by letting the dimension k of the space where one searches for a realization of A be part of the input data.

Distance matrices are a central notion in the area of distance geometry; their study was initiated by A. Cayley in the 18th century and it was continued in particular by K. Menger and I.J. Schoenberg in the 1930s. They are, in fact, closely related to psd matrices. The following basic connection was established in [63]. Given a symmetric $(n \times n)$-matrix $D = (d_{ij})_{i,j=1}^n$ with zero diagonal entries, consider the symmetric $((n-1) \times (n-1))$-matrix $X = (x_{ij})_{i,j=1}^{n-1}$ defined by

$$x_{ij} = \frac{1}{2}(d_{in} + d_{jn} - d_{ij}) \quad \text{for all } i, j = 1, \ldots, n-1. \tag{6}$$

Then, D is a distance matrix if and only if X is psd; moreover, D has a realization in the k-space if and only if X has rank $\leq k$. Other characterizations are known for distance matrices. As the literature on this topic is quite large, see the monographs [11,13,14], where further references can be found.

Problems (EDM) and (EDMk) have many important applications; for instance, to multidimensional scaling problems in statistics (cf. [49]) and to position-location problems, i. e., problem (EDMk) mostly in dimension $k \leq 3$. A much studied instance of the latter problem is the molecular conformation problem in chemistry; indeed, nuclear magnetic resonance spectroscopy permits to determine some pairwise interatomic distances, the question being then to reconstruct

the global shape of the molecule from this partial information (cf. [13,41]).

In view of relation (6), problem (EDM) can be formulated as an instance of the semidefinite programming problem (P) and, therefore, it can be solved with an arbitrary precision in polynomial time. Exploiting this fact, some specific algorithms based on interior point methods are presented in [2] together with numerical tests. Moreover, problem (EDM) can be solved in polynomial time when restricted to partial rational matrices whose pattern is a chordal graph or, more generally, a graph with fixed minimum fill-in [48]; as in the psd case, this follows from the fact (mentioned below) that partial matrices that are completable to a distance matrix admit a good characterization when their pattern is a chordal graph.

While the exact complexity of problem (EDM) is not known, it has been shown in [62] that problem (EDMk) is *NP*-complete if $k = 1$ and *NP*-hard if $k \geq 2$ (even when restricted to partial matrices with entries in $\{1, 2\}$). Finding ϵ-optimal solutions to the graph realization problem is also NP-hard for small ϵ ([53]). The graph realization problem (EDMk) has been much studied, in particular in dimension $k \leq 3$, which is the case most relevant to applications. The problem can be formulated as a nonlinear global optimization problem: $f(v)$ such that $v = (v_1, \ldots, v_n) \in \mathbf{R}^{kn}$, where the cost function $f(\cdot)$ can, for instance, be chosen as

$$f(v) = \sum_{ij \in E} (\|v_i - v_j\|^2 - a_{ij})^2.$$

Hence, $f(\cdot)$ is zero precisely when the v_i's provide a realization of the partial matrix A. This optimization problem is hard to solve (as it may have many local optimum solutions). Several algorithms have been proposed in the literature; see, in particular, [13,19,26,29,31,41,54,57]. They are based on general techniques for global optimization like tabu and pattern search [57], the continuation approach (which consists of transforming the original function $f(\cdot)$ into a smoother function having fewer local optimizers, [53,54]), or divide-and-conquer strategies aiming to break the problem into a sequence of smaller or easier subproblems [13,29,31]. In [29,31], the basic step consist of finding principal submatrices having a unique realization, treating each of them separately and then trying to combine the solutions. Thus arises the problem of identifying principal submatrices having a unique realization, which turns out to be *NP*-hard [62]. However, several necessary conditions for unicity of realization are known, related with connectivity and generic rigidity properties of the graph pattern [30,67]. Generic rigidity of graphs can be characterized and recognized in polynomial time only in dimension $k \leq 2$ ([42,51]) (cf. the survey [43] for more references).

Call a partial matrix A a *partial distance matrix* if every specified principal submatrix of A is a distance matrix. Being a partial distance matrix is obviously a necessary condition for A to be completable to a distance matrix. It is shown in [4] that every partial distance matrix with pattern G is completable to a distance matrix if and only if G is a chordal graph; moreover, if all specified principal submatrices of the partial matrix A have a realization in the k-space, then A admits a completion having a realization in the k-space.

As noted in [33], if a partial matrix A with pattern G is completable to a distance matrix, then the associated vector $x := (\sqrt{a_{ij}})_{ij \in E}$ must satisfy the inequalities:

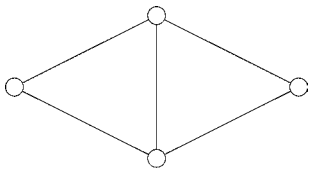
$$x_e - \sum_{f \in C \setminus \{e\}} x_f \leq 0 \quad \text{for all } e \in C,\ C \text{ circuit in } G. \tag{7}$$

The graphs G for which every partial matrix (respectively, partial distance matrix) A with pattern G for which (7) holds is completable to a distance matrix, are the graphs containing no homeomorph of K_4 as an induced subgraph [45] (respectively, the graphs that can be made chordal by adding edges in such a way that no new clique of size 4 is created [33]). Note the analogy with the corresponding results for the psd completion problem; some connections between the two problems (EDM) and (PSD) are exposed in [38,45].

Completion to Completely Positive and Contraction Matrices

Call a matrix *doubly nonnegative* if it is psd and entrywise nonnegative. Every completely positive (cp, for short) matrix is obviously doubly nonnegative. The converse implication holds for matrices of order $n \leq 4$ (cf. [22]) and for certain patterns of the nonzero entries in A (cf. [40]). The cp property is obviously inherited by principal submatrices; call a partial matrix A a *partial cp matrix* if every fully specified principal submatrix of A is cp. It is shown in [15] that every partial cp matrix

with graph pattern G is completable to a cp matrix if and only if G is a so-called block-clique graph. A *block-clique graph* being a chordal graph in which any two distinct maximal cliques overlap in at most one node or, equivalently, a chordal graph that does not contain an induced subgraph of the form:



Recall that an $(n \times m)$-matrix A is a contraction matrix if all eigenvalues of A^*A are less than or equal to 1 or, equivalently, if the matrix

$$\widetilde{A} = \begin{pmatrix} I_n & A \\ A^* & I_m \end{pmatrix} \tag{8}$$

is positive semidefinite. Call a partial matrix A a *partial contraction* if all specified submatrices of A are contractions. As every submatrix of a contraction is again a contraction, an obvious necessary condition for a partial matrix A to be completable to a contraction matrix is that A be a partial contraction. Thus arises the question of characterizing the graph patterns G for which every partial contraction with pattern G can be completed to a contraction matrix.

As we now deal with rectangular $n \times m$ partial matrices A, their pattern is the bipartite graph G with node set $U \cup V$, where U, V index the rows and columns of A and edges of G correspond to the specified entries of A. We may clearly assume to be dealing with partial matrices whose pattern is a connected graph (as the partial matrices associated with the connected components can be handled separately). Below is an example of a partial matrix A which is a partial contraction, but which is not completable to a contraction matrix:

$$A = \begin{pmatrix} ? & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & ? & \frac{1}{\sqrt{2}} \end{pmatrix}.$$

In fact, the graph pattern displayed in this example is in a sense present in every partial contraction which is not completable to a contraction. Namely, it is shown in [36] that the following assertions (i–iii) are equivalent for a connected bipartite graph G with node bipartition $U \cup V$:

i) Every partial contraction with pattern G can be completed to a contraction;
ii) G does not contain an induced matching of size 2 (i. e., if $e := uv$, $e' := u'v'$ are edges in G with $u \neq u' \in U$, $v \neq v' \in V$, then at least one of the pairs uv', $u'v$ is an edge in G; that is, G is nonseparable in the terminology of [21]);
iii) The graph $\widetilde{G}$ obtained from G by adding all edges uu' $(u \neq u' \in V)$ and vv' $(v \neq v' \in V)$ is chordal.

(Note that the implication iii) $\rightarrow$ i) is a consequence of the result on psd completions from [23] mentioned in the Section on the positive semidefinite completion problem above, as $\widetilde{G}$ is the graph pattern of the matrix $\widetilde{A}$ defined in (8).)

Rank Completions

In this section, we consider the problem of determining the possible ranks for the completions of a given partial matrix. For a partial matrix A, let $\mathrm{mr}(A)$ and $\mathrm{MR}(A)$ denote, respectively, the minimum and maximum possible ranks for a completion of A. If B, C are completions of A of respective ranks $\mathrm{mr}(A)$, $\mathrm{MR}(A)$, then changing B into C by changing one entry of B into the corresponding entry of C at a time permits to construct completions realizing all ranks in the range $[\mathrm{mr}(A), \mathrm{MR}(A)]$. Hence, the question is to determine the two extreme values $\mathrm{mr}(A)$ and $\mathrm{MR}(A)$. As we see below, the value $\mathrm{MR}(A)$ can, in fact, be expressed in terms of ranks of fully specified submatrices of A and it can be computed in polynomial time; this constitutes a generalization of the celebrated Frobenius–König theorem (corresponding to the case when specified entries are equal to 0). On the other hand, determining $\mathrm{mr}(A)$ seems to be a much more difficult task.

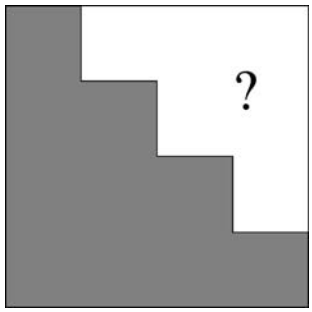
We first deal with the problem of finding *maximum rank completions*. Let A be an $n \times m$ partial matrix with graph pattern G, i. e., G is the bipartite graph $(U \cup V, E)$ where U, V index respectively the rows and columns of A, and the edges of G correspond to the specified entries of A, and let $\overline{G}$ denote the complementary bipartite graph whose edges correspond to unspecified entries of A. Note that computing $\mathrm{MR}(A)$ amounts to computing the generic rank of A when viewing the unspecified entries of A as independent variables over the field containing the specified entries. For a subset $X \subseteq U \cup V$, let A_X denote the submatrix of A with respective row

and column index sets $\{i \in [1, n]: u_i \notin X\}$ and $\{j \in [1, m]: v_j \notin X\}$. Call X a *cover* of $\overline{G}$ if every edge of $\overline{G}$ has at least one end node in X; that is, if A_X is a fully specified submatrix of A. Clearly, we have: $\mathrm{MR}(A) \leq \mathrm{rank}(A_X) + |X|$. In fact, the following equality holds:

$$\mathrm{MR}(A) = \min_{X \text{ cover of } \overline{G}} \mathrm{rank}(A_X) + |X| \tag{9}$$

as shown in [12]. A determinantal version of the result was given in [25]. In the special case when all specified entries of A are equal to 0, then $\mathrm{MR}(A)$ coincides with the maximum cardinality of a matching in $\overline{G}$ and, therefore, the minimax relation (9) reduces to the Frobenius–König theorem (cf. [50] for details on the latter result). Moreover, one can determine $\mathrm{MR}(A)$ and construct a maximum rank completion of A in polynomial time. This was shown in [55] by a reduction to matroid intersection and, more recently, in [18] where a simple greedy procedure is presented that solves the problem by perturbing an arbitrary completion.

We now consider *minimum rank completions*. To start with, note that $\mathrm{mr}(A)$ may depend, in general, on the actual values of the specified entries of A (and not only on the ranks of the specified submatrices of A). Indeed, consider the partial matrix $A = \begin{pmatrix} ? & a & b \\ d & ? & c \\ e & f & ? \end{pmatrix}$ where $a, b, c, d, e, f \neq 0$. Then, $\mathrm{mr}(A) = 1$ if $ace = bdf$ and $\mathrm{mr}(A) = 2$ otherwise, while all specified submatrices have rank 1 in both cases. Thus arises the question of identifying the bipartite graphs G for which $\mathrm{mr}(A)$ depends only on the ranks of the specified submatrices of A for every partial matrix A with pattern G; such graphs are called *rank determined*. The graph pattern of the above instance A is the circuit C_6. Hence, C_6 is not rank determined. Call a bipartite graph G *bipartite chordal* if it does not contain a circuit of length ≥ 6 as an induced subgraph. Then, if a bipartite graph is rank determined, it is necessarily bipartite chordal [12]. It is conjectured there that, conversely, every bipartite chordal graph is rank determined. The conjecture was shown to be true in [66] for the nonseparable bipartite graphs (i. e., the bipartite graphs containing no induced matching of size 2; they are obviously bipartite chordal). Note that a partial matrix A has a nonseparable pattern if and only if it has (up to row/column permutation) the following 'triangular' form:



Then, $\mathrm{mr}(A)$ can be explicitly formulated in terms of the ranks of the specified submatrices of A; in the simplest case, the formula for $\mathrm{mr}(A)$ reads:

$$\mathrm{mr}\begin{pmatrix} B & ? \\ C & D \end{pmatrix} = \mathrm{rank}\begin{pmatrix} B \\ C \end{pmatrix} + \mathrm{rank}\begin{pmatrix} C & D \end{pmatrix} - \mathrm{rank}(C).$$

It is shown in [12] that the above conjecture holds when the pattern G is a path, or when G is obtained by 'gluing' a collection of circuits of length 4 along a common edge.

See also

- ► Interior Point Methods for Semidefinite Programming
- ► Semidefinite Programming and Determinant Maximization

References

1. Agler J, Helton JW, McCullough S, Rodman L (1988) Positive semidefinite matrices with a given sparsity pattern. Linear Alg & Its Appl 107:101–149
2. Alfakih AY, Khandani A, Wolkowicz H (1998) Solving Euclidean distance matrix completion problems via semidefinite programming. Comput Optim Appl 12:13–30
3. Alizadeh F (1995) Interior point methods in semidefinite programming with applications in combinatorial optimization. SIAM J Optim 5:13–51
4. Bakonyi M, Johnson CR (1995) The Euclidian distance matrix completion problem. SIAM J Matrix Anal Appl 16:646–654
5. Barahona F, Mahjoub AR (1986) On the cut polytope. Math Program 36:157–173
6. Barrett WW, Johnson CR, Loewy R (1996) The real positive definite completion problem: cycle completability. Memoirs Amer Math Soc, vol 584. Amer. Math. Soc., Providence, RI

7. Barrett WW, Johnson CR, Lundquist M (1989) Determinantal formulas for matrix completions associated with chordal graphs. Linear Alg & Its Appl 121:265–289
8. Barrett W, Johnson CR, Tarazaga P (1993) The real positive definite completion problem for a simple cycle. Linear Alg & Its Appl 192:3–31
9. Barvinok AI (1993) Feasibility testing for systems of real quadratic equations. Discrete Comput Geom 10:1–13
10. Barvinok AI (1995) Problems of distance geometry and convex properties of quadratic maps. Discrete Comput Geom 13:189–202
11. Blumenthal LM (1953) Theory and applications of distance geometry. Oxford Univ. Press, Oxford
12. Cohen N, Johnson CR, Rodman L, Woederman HJ (1989) Ranks of completions of partial matrices. In: Dym H, et al (eds) The Gohberg Anniv. Coll. vol I, Birkhäuser, Basel, pp 165–185
13. Crippen GM, Havel TF (1988) Distance geometry and molecular conformation. Res. Studies Press, Taunton, UK
14. Deza M, Laurent M (1997) Geometry of cuts and metrics. Algorithms and Combinatorics, vol 15. Springer, Berlin
15. Drew JH, Johnson CR (1998) The completely positive and doubly nonnegative completion problems. Linear Alg & Its Appl 44:85–92
16. Dym H, Gohberg I (1981) Extensions of band matrices with band inverses. Linear Alg & Its Appl 36:1–24
17. Ellis RL, Lay DC, Gohberg I (1988) On negative eigenvalues of selfadjoint extensions of band matrices. Linear Alg & Its Appl 24:15–25
18. Geelen J (1999) Maximum rank matrix completion. Linear Alg & Its Appl 288:211–217
19. Glunt W, Hayden TL, Raydan M (1998) Molecular conformations from distance matrices. J Comput Chem 14:175–190
20. Goemans MX (1997) Semidefinite programming in combinatorial optimization. Math Program 79:143–161
21. Golumbic MC (1980) Algorithmic theory and perfect graphs. Acad. Press, New York
22. Gray LJ, Wilson DG (1980) Nonnegative factorization of positive semidefinite nonnegative matrices. Linear Alg & Its Appl 31:119–127
23. Grone R, Johnson CR, Sá EM, Wolkowicz H (1984) Positive definite completions of partial hermitian matrices. Linear Alg & Its Appl 58:109–124
24. Grötschel M, Lovász L, Schrijver A (1988) Geometric algorithms and combinatorial optimization. Springer, Berlin
25. Hartfiel DJ, Loewy R (1984) A determinantal version of the Frobenius-König theorem. Linear Multilinear Algebra 16:155–165
26. Havel TF (1991) An evaluation of computational strategies for use in the determination of protein structure from distance constraints obtained by nuclear magnetic resonance. Program Biophys Biophys Chem 56:43–78
27. Helmberg C, Rendl F, Vanderbei RJ, Wolkowicz H (1996) An interior-point method for semidefinite programming. SIAM J Optim 6:342–361
28. Helton JW, Pierce S, Rodman L (1989) The ranks of extremal positive semidefinite matrices with given sparsity pattern. SIAM J Matrix Anal Appl 10:407–423
29. Hendrickson B (1990) The molecule problem: Determining conformation from pairwise distances. PhD Thesis. Techn. Report Dept. Computer Sci. Cornell Univ. 90–1159
30. Hendrickson B (1992) Conditions for unique graph realizations. SIAM J Comput 21:65–84
31. Hendrickson B (1995) The molecule problem: exploiting structure in global optimization. SIAM J Optim 5:835–857
32. Johnson CR (1990) Matrix completion problems: A survey. In: Johnson CR (ed) Matrix Theory and Appl. Proc Symp Appl Math. Amer. Math. Soc., Providence, RI, pp 171–198
33. Johnson CR, Jones C, Kroschel B (1995) The distance matrix completion problem: cycle completability. Linear Multilinear Algebra 39:195–207
34. Johnson CR, Kroschel B, Wolkowicz H (1998) An interior-point method for approximate positive semidefinite completions. Comput Optim Appl 9:175–190
35. Johnson CR, Rodman L (1984) Inertia possibilities for completions of partial Hermitian matrices. Linear multilinear algebra 16:179–195
36. Johnson CR, Rodman L (1986) Completion of matrices to contractions. J Funct Anal 69:260–267
37. Johnson CR, Rodman L (1988) Chordal inheritance principles and positive definite completions of partial matrices over function rings. In: Gohberg I (ed) Contributions to Operator Theory and its Applications. Birkhäuser, Basel, pp 107–127
38. Johnson CR, Tarazaga P (1995) Connections between the real positive semidefinite and distance matrix completion problems. Linear Alg & Its Appl 223(4):375–391
39. Khachiyan L, Porkolab L (1997) Computing integral points in convex semi-algebraic sets. 38th Annual Symp. Foundations Computer Sci., pp 162–171
40. Kogan N, Berman A (1993) Characterization of completely positive graphs. Discret Math 114:297–304
41. Kuntz ID, Thomason JF, Oshiro CM (1993) Distance geometry. Methods in Enzymologie 177:159–204
42. Laman G (1970) On graphs and rigidity of plane skeletal structures. J Eng Math 4:331–340
43. Laurent M (1997) Cuts, matrix completions and graph rigidity. Math Program 79:255–283
44. Laurent M (1997) The real positive semidefinite completion problem for series-parallel graphs. Linear Alg & Its Appl 252:347–366
45. Laurent M (1998) A connection between positive semidefinite and Euclidean distance matrix completion problems. Linear Alg & Its Appl 273:9–22
46. Laurent M (1998) On the order of a graph and its deficiency in chordality. CWI Report PNA-R9801
47. Laurent M (1998) A tour d'horizon on positive semidefinite and Euclidean distance matrix completion problems.

In: Pardalos PM, Wolkowicz H (eds) Topics in Semidefinite and Interior-Point Methods. Fields Inst Res Math Sci Commun. Amer. Math. Soc., Providence, RI, pp 51–76
48. Laurent M (2000) Polynomial instances of the positive semidefinite and Euclidean distance matrix completion problems. SIAM J Matrix Anal Appl 22(3)(874–894
49. de Leeuw J, Heiser W (1982) Theory of multidimensional scaling. In: Krishnaiah PR, Kanal LN (eds) Handbook Statist., vol 2, North-Holland, Amsterdam, pp 285–316
50. Lovász L, Plummer MD (1986) Matching theory. Akad. Kiadó, Budapest
51. Lovász L, Yemini Y (1982) On generic rigidity in the plane. SIAM J Alg Discrete Meth 3:91–98
52. Lundquist ME, Johnson CR (1991) Linearly constrained positive definite completions. Linear Alg & Its Appl 150:195–207
53. Moré JJ, Wu Z (1996) ϵ-optimal solutions to distance geometry problems via global continuation. In: Pardalos PM, Shalloway D (eds) DIMACS, vol 23, Amer. Math. Soc., Providence, RI, 151–168
54. Moré JJ, Wu Z (1997) Global continuation for distance geometry problems. SIAM J Optim 7:814–836
55. Murota K (1993) Mixed matrices: Irreducibility and decomposition. In: Brualdi RA et al (eds) Combinatorial and graph-theoretical problems in linear algebra. IMA. Springer, Berlin, pp 39–71
56. Nesterov YE, Nemirovsky AS (1994) Interior point polynomial algorithms in convex programming: Theory and algorithms. SIAM, Philadelphia
57. Pardalos PM, Lin X (1997) A tabu based pattern search method for the distance geometry problem. In: Giannessis F et al (eds) Math. Program. Kluwer, Dordrecht
58. Paulsen VI, Power SC, Smith RR (1989) Schur products and matrix completions. J Funct Anal 85:151–178
59. Porkolab L, Khachiyan L (1997) On the complexity of semidefinite programs. J Global Optim 10:351–365
60. Ramana MV (1997) An exact duality theory for semidefinite programming and its complexity implications. Math Program 77:129–162
61. Rose DJ (1970) Triangulated graphs and the elimination process. J Math Anal Appl 32:597–609
62. Saxe JB (1979) Embeddability of weighted graphs in k-space is strongly NP-hard. In: Proc. 17th Allerton Conf. Communications, Control and Computing, pp 480–489
63. Schoenberg IJ (1935) Remarks to M. Fréchet's article 'Sur la définition axiomatique d'une classe d'espaces vectoriels distanciés applicables vectoriellement sur l'espace de Hilbert', Ann of Math 36:724–732
64. WEB: http://orion.math.uwaterloo.ca:80/~hwolkowi/henry/software/readme.html
65. WEB: http://www.zib.de/helmberg/semidef.html
66. Woerdeman HJ (1987) The lower order of lower triangular operators and minimal rank extensions. Integral Eq Operator Theory 10:859–879
67. Yemini Y (1979) Some theoretical aspects of position-location problems. Proc. 20th Annual Symp. Foundations Computer Sci., pp 1–8

Matroids

PAOLA FESTA
Dip. Mat. e Inform., Universita Salerno, Baronissi, Italy

MSC2000: 90C09, 90C10

Article Outline

Keywords
Historical Overview
Definition of a Matroid
Minor of Matroids: Restriction and Contraction
Representability of Matroids
Connectivity of Matroids
Examples of Matroids
  Uniform Matroid
  Graphic Matroid
  Transversal Matroid
  Partition Matroid
  Dual Matroids
Greedy Algorithms on Weighted Matroids
See also
References

Keywords

Combinatorial optimization; Greedy technique; Graph optimization

Matroids have been defined in 1935 as generalization of graphs and matrices. Starting from the 1950s they have had increasing interest and the theoretical results obtained have been used for solving several difficult problems in various fields such as civil, electrical, and mechanical engineering, computer science, and mathematics. A comprehensive treatment of matroids can not be contained in few pages or even in only one book. Thus, the scope of this article is to introduce the reader to this theory, providing the definitions of some different types of matroids and their main properties.

Historical Overview

In 1935, H. Whitney in [38] studied *linear dependence* and its important application in mathematics. A number of equivalent axiomatic systems for matroids is contained in his pioneering paper, that is considered the first scientific work about matroid theory.

In the 1950s and 1960s, starting from the Whitney's ideas, W. Tutte in [25,26,27,28,29,30,31,32,33] built a considerable body of theory about the structural properties of matroids, which became popular in the 1960s, when J. Edmonds in [5,6,7,8,9,10,11] introduced matroid theory in combinatorial optimization. From 1965 on, a growing number of researchers became interested in matroids. In 1976, D.J.A. Welsh ([34]) published the first book on matroid theory. In the 1970s, 1980s, and 1990s selected topics have been covered by a huge number of scientific publications, among them [1,2,3,12,13,15,17,18,20,21,23,24,35,36,37]. [16] provides an excellent historical survey, while [21] is a good book for students.

Definition of a Matroid

Matroids are combinatorial structures often treated in together with the *greedy technique*, which yields optimal solutions when applied for solving simple problems defined on matroids.

In order to provide the definition of a general matroid, some notation and further definitions are needed.

Definition 1 An ordered pair $S = (E, I)$, where $E = \{e_1, \ldots, e_n\}$ and $I \subseteq 2^E$, is an *independent system* (SI) if and only if

$$\forall A, B \subseteq E : \; B \subset A \in I \Rightarrow B \in I. \tag{1}$$

E is also called *ground set*.

Note that the empty set is necessarily a member of I.

Definition 2 The members of I are called *independent sets*.

Definition 3 The members of $D = 2^E \setminus I$ are called *dependent sets*.

Definition 4 The members of the set

$$B = \{A \subseteq E : \; A \in I, \; \forall f \in E \setminus A : \; B \cup \{f\} \notin I\}$$

are called *maximal independent sets* or *bases*.

In other words, a *basis* is an independent set which is maximal with respect to set inclusion operation.

Definition 5 The members of the set

$$C = \{C \subseteq E : \; C \in D, \; \forall f \in C : \; C \setminus \{f\} \in I\}$$

are called *minimal dependent sets* or *circuits*. A 1-element circuit is a *loop*.

Definition 6 A *matroid* M is an independent system (E, I) such that if $A, B \in I$, $|A| < |B|$, then there is some element $x \in \mathrm{B} \setminus A$ such that $A \cup \{x\} \in I$.

We say that M satisfies the *exchange property*.

Most combinatorial problems can be viewed as the problem of finding an element in one of the above defined sets corresponding to the optimal objective function value.

The word *matroid* is due to Whitney. He studied *matric matroids*, in which the elements of E are the rows of a given matrix and a set of rows is independent if they are linearly independent in the usual sense.

The following theorems express two equivalent axiomatic definitions of matroids in terms of bases and circuits.

Theorem 7 *A nonempty set B of subsets of E is the set of bases for a matroid $M = (E, I)$ if and only if for all B_1, $B_2 \in B$, $B_1 \neq B_2$, and $x \in B_1 \setminus B_2$, there exists an element $y \in B_2 \setminus B_1$ such that*

$$B_1 \cup \{y\} \setminus \{x\} \in B.$$

Theorem 8 *A set C of subsets of E is the set of circuits for a matroid $M = (E, I)$ if and only if the following two properties hold:*

1) *for all $X \neq Y \in C$, $X \not\subseteq Y$;*
2) *for all $X \neq Y \in C$ and $z \in X \cap Y$, there exists $Z \in C$ such that $Z \subseteq X \cup Y \setminus \{z\}$.*

Other alternative axiomatic characterizations of a matroid need some further definitions.

Let $M = (E, I)$ be a matroid.

Definition 9 For all $A \subseteq E$, let ρ: 2^E **N** be a function such that

$$\rho(A) = \max\{|X| : \; X \subseteq A, X \in I\}.$$

ρ is called *rank* of M.

Note that the rank of M is equal to the rank of E, which is given by the cardinality of the maximal independent subset of E. The rank is always well-defined, due to the following proposition.

Proposition 10 *If A is a subset of E and X and Y are maximal independent subsets of A, then $|X| = |Y|$.*

Proposition 10 claims that the maximal independent subsets contained in $A \subseteq E$ of a given matroid $M = (E, I)$ have the same cardinality. Choosing $A = E$, the following corollary holds.

Corollary 11 *The bases of any matroid have the same cardinality.*

Definition 12 A subset A of E is called a *closed* of M if

$$\rho(A \cup \{x\}) = \rho(A) + 1, \quad \forall x \in E \setminus A,$$

i. e. if it is not possible to add to A any element without increasing its rank.

Definition 13 The *closure operator* for M is a function $\sigma\colon 2^E \to 2^E$ such that for all $A \subseteq E$ $\sigma(A)$ is the closed of minimum cardinality that contains A, i. e.

$$\sigma(A) = A \cup \{x \in E \setminus A\colon\ \rho(A \cup \{x\}) = \rho(A)\}.$$

Definition 14 A subset A of E *covers* M if and only if it contains a basis of M, i. e.

$$\rho(A) = \rho(E).$$

With these further definitions at hand, the following theorems express three other equivalent axiomatic characterizations of a matroid in terms of its rank.

Theorem 15 *A function $\rho\colon 2^E \to \mathbf{N}$ is a rank function of a matroid $M = (E, I)$ if and only if for all $X \subseteq E$ and for all $y, z \in E$ the following three properties hold:*
1) $\rho(\emptyset) = 0$;
2) $\rho(X) \leq \rho(X \cup \{y\}) \leq \rho(X) + 1$;
3) $\rho(X) = \rho(X \cup \{y\}) = \rho(X \cup \{z\}) \Rightarrow \rho(X \cup \{y, z\}) = \rho(X)$.

Theorem 16 *A function $\rho\colon 2^E \to \mathbf{N}$ is a rank function of a matroid $M = (E, I)$ if and only if for all $X \neq Y \subseteq E$ the following three properties hold:*
1) $0 \leq \rho(X) \leq |X|$;
2) $X \subseteq Y \Rightarrow \rho(X) \leq \rho(Y)$;
3) $\rho(X \cup Y) + \rho(X \cap Y) \leq \rho(X) + \rho(Y)$.

Note that the second property of theorem 16 implies that ρ is a monotonic function, while the third property expresses its submodularity.

Theorem 17 *A function $\sigma\colon 2^E \to 2^E$ is a closure operator of a matroid $M = (E, I)$ if and only if for all $X \neq Y \subseteq E$ and for all $x, y \in E$ the following four properties hold:*
1) $X \subseteq \sigma(X)$;
2) $Y \subseteq X \Rightarrow \sigma(Y) \subseteq \sigma(X)$;
3) $\sigma(X) = \sigma(\sigma(X))$;
4) $y \notin \sigma(X)$, $y \in \sigma(X \cup \{x\}) \Rightarrow x \in \sigma(X \cup \{y\})$.

Definition 18 A matroid $M = (E, I)$ is *weighted* if there is an associated weight function w that assigns a strictly positive weight $w(x)$ to each element $x \in E$.

The *weight function* w extends to subsets A of E by summation:

$$w(A) = \sum_{x \in A} w(x).$$

Minor of Matroids: Restriction and Contraction

A *minor* of a matroid $M = (E, I)$ is a 'submatroid' obtained from *deleting* or *contracting* from the ground set E one or more elements.

A *loop* is an element y of a matroid such that $\{y\}$ is not independent. Equivalently, $\{y\}$ does not lie in any independent set, nor in maximal independent sets.

Definition 19 Let $M = (E, I)$ be a matroid. If an element $\{x\}$ is not a loop, the matroid M/x, called a *contraction* of M, is defined as follows:
1) the ground set of M/x is $E \setminus \{x\}$;
2) a set A is independent in M/x if and only if $A \cup \{x\}$ is independent in M.

The concept of matroid contraction can be dualized. In fact, an element y is a *coloop* if it is contained in every basis of M.

Definition 20 Let $M = (E, I)$ be a matroid. If an element $\{x\}$ is not a coloop, the matroid $M \setminus x$, called a *restriction* of M, is defined as follows:
1) the ground set of $M \setminus x$ is $E \setminus \{x\}$;
2) a set A is independent in $M \setminus x$ if and only if it is independent in M.

The above definitions have been given in terms of restriction and contraction of only one element, but they

can be easily extended to the restriction and contraction of a set X. The minors obtained will be denoted $M\backslash X$ and $M\backslash X$, respectively.

Representability of Matroids

One among the most common canonical examples of matroids is the *vectorial matroid*, whose ground set E is a finite set of vectors from a vector space, while the independent sets are the linearly independent subsets of vectors of E. A matroid $M = (E, I)$ is *representable* on a field F if there exists some vector space V over F, with some finite set E of vectors of V, so that M is isomorphic to the vectorial matroid of the set E. A *binary matroid* is a matroid representable over GF(2), while a *ternary matroid* is representable over GF(3).

In recent literature (as of 1999) the problem of classifying all the fields over which a given matroid is representable and the inverse problem of characterizing all the matroids that are representable on a given field have had growing interest. An important result for matroid representability is the following theorem.

Theorem 21 *A matroid M = (E, I) is representable over any field if and only if it is representable over GF(2) and over some field of characteristic other than two.*

A matroid as in the previous theorem is called *regular*.

Connectivity of Matroids

Connectivity is an important concept in matroid theory.

Definition 22 A matroid $M = (E, I)$ admits a *k-separation* if there exists a partition (X, Y) of the ground set E such that
1) $|X| \geq k, |Y| \geq k$;
2) $\rho(X) + \rho(Y) - \rho(E) \leq k - 1$.

Definition 23 The smallest k such that a matroid $M = (E, I)$ admits a k-separation is called the *connectivity* of M.

If $k \geq 2$, M is n-connected for any $n \leq k$; if $k = 1$, M is *disconnected*; if M admits any k-separations for all integers k, M has *infinite connectivity*.

An important result for matroid connectivity is the following theorem.

Theorem 24 *A matroid M = (E, I) is disconnected if and only if there exists a partition (X, Y) of the ground set E such that every circuit C of M is either a subset of X or a subset of Y.*

Examples of Matroids

In this section some of the most popular types of matroids involved in combinatorial optimization will be described.

Uniform Matroid

Let E be a set of n elements and let I be the family of subsets A of E such that $|A| \leq k < n$. Then $M = (E, I)$ is called the *uniform matroid* of rank k and is denoted by $U_{k,n}$.

The sets of the bases and the circuits of $U_{k,n}$ are

$$B = \{X \subseteq E\colon\ |X| = k\}$$

and

$$C = \{X \subseteq E\colon\ |X| = k + 1\},$$

respectively.

Moreover, for all $A \subseteq E$,

$$\rho(A) = \begin{cases} |A| & \text{if } |A| \leq K, \\ K & \text{otherwise,} \end{cases}$$

$$\sigma(A) = \begin{cases} A & \text{if } |A| \leq K, \\ E & \text{otherwise.} \end{cases}$$

Graphic Matroid

If F is the set of forests of a graph $G = (V, E)$, $M = (E, F)$ is called a *graphic matroid*. The circuits of M are the graph-theoretic circuits of G, while the rank of a subset E_1 of E is given by

$$\rho(E_1) = |V| - c(E_1),$$

where $c(E_1)$ is the number of connected components of $G_1 = (V, E_1)$.

Transversal Matroid

Let E be a finite set, $C = \{S_1, \ldots, S_m\}$ a collection of subsets of E, and let $T = \{e_1, \ldots, e_t\} \subseteq E$.

T is called a *transversal* of C if there exist distinct integers $j(1), \ldots, j(t)$ such that $e_i \in S_{j(i)}$, $i = 1, \ldots, t$. Let I be the set of all transversals of E, then $M = (E, I)$ is a *transversal matroid*.

Partition Matroid

Let E be a finite set, $\Pi = \{E_1, \ldots, E_p\}$ a *partition* of E, that is a collection of disjoint subsets of E covering E, and $d_1, \ldots, d_p$ p nonnegative integers. A subset A of E is *independent*, i. e. $A \in I$, if and only if $|A \cap E_j| \leq d_j$, $j = 1, \ldots, p$. The system $M = (E, I)$ is a matroid, called a *partition matroid*.

An example of a partition matroid can be obtained by considering any digraph $G = (V, E)$ and partitioning the edges of the set E according to which node is the head (or, equivalently, the tail) of each. Suppose that $d_j = 1, j = 1, \ldots, p$; then a set A of edges is independent if no two edges of A have the same head (or, equivalently, the same tail).

Dual Matroids

Let $M = (E, I)$ be a matroid, and let B be its set of bases.

The *dual matroid* $\overline{M}$ is the matroid on the ground set E, whose bases are the complements of the bases of M. Thus, a set A is independent in $\overline{M}$ if and only if A is disjoint from some basis of M. Note that $\overline{\overline{M}} = M$.

For a pair of matroids $(M, \overline{M})$ and their rank functions, the following propositions hold.

Proposition 25 *Let $M = (E, I)$ be a matroid, and let ρ be its rank function. Let $\overline{M} = (E, \overline{I})$ be the dual matroid of M; then*

$$\overline{\rho}(A) = |A| + \rho(E \setminus A) - \rho(E),$$

for each $A \subseteq E$.

Proposition 26 *Let $\overline{M}$ be the dual of the matroid $M = (E, I)$, let A be a subset of E and let $\overline{A} = E \setminus A$. If ρ and $\overline{\rho}$ are the rank functions of M and $\overline{M}$ respectively, then*

1) $\left|\overline{A}\right| - \overline{\rho}(\overline{A}) = \rho(E) - \rho(A)$;
2) $\overline{\rho}(\overline{E}) - \overline{\rho}(\overline{A}) = |A| - \rho(A)$.

Proposition 27 *Let $M = (E, I)$ be a matroid, then*

1) *x is a loop in M if and only if x is a coloop in $\overline{M}$ and vice versa;*
2) *If x is not a loop in M, then the dual of M/x is the matroid $\overline{M}\backslash x$;*
3) *If x is not a coloop in M, then the dual of $M \setminus x$ is the matroid $\frac{\overline{M}}{x}$.*

As example of the dual of a matroid, let us consider the vectorial matroid. Suppose that the vectors representing M are the columns of an $m \times n$ matrix A and that these vectors span F^m. Thus, A has rank m and is the matrix of a linear transformation T from F^n onto F^m. Let K be the kernel of T, and B the matrix of a linear embedding of U into F^n. Note that B is a $n \times (n - m)$ matrix (whose columns are the basis for U) and has rank $n - m$. Moreover, the columns of the $(n - m) \times n$ matrix $B^\intercal$ are indexed by the same set as the columns of A and $B^\intercal A = 0$. $B^\intercal$ is the dual matroid $\overline{M}$ of the vectorial matroid M.

Greedy Algorithms on Weighted Matroids

Many combinatorial problems for which the greedy technique gives an optimal solution can be formulated in terms of finding a maximum-weight independent subset in a weighted matroid. In more detail, there is given a weighted matroid $M = (E, I)$ and the objective is to find an independent set $A \in I$ such that $w(A)$ is maximized (also called an *optimal subset* of M). Since the weight $w(x)$ of any element $x \in E$ is positive, a maximum-weight independent subset is always a maximal independent subset.

In the *minimum spanning tree problem*, for example, there are given a connected undirected graph $G = (V, E)$ and a length function w such that $w(e)$ is the positive length of the edge e. The objective is to find an acyclic subset T of E that connects all of the vertices of G and whose total length

$$w(T) = \sum_{e \in T} w(e)$$

is minimized. This is a classical combinatorial problem and can be formulated as a problem of finding an optimal subset of a matroid. In fact, consider the graphic weighted matroid M_G with weight function w' such that $w'(e) = w_0 - w(e)$, where w_0 is larger than the maximum length of any edge. It can be easily seen that for each $e \in E$, $w'(e) \geq 0$ and that an optimal subset of M_G is a spanning tree of minimum total length in the original graph G. In more detail, each maximal independent subset A corresponds to a spanning tree and since

$$w'(A) = (|V| - 1) \cdot w_0 - w(A)$$

for any maximal independent subset A, the independent subset that maximizes $w'(A)$ must minimize $w(A)$.

J.B. Kruskal in [14] and R.C. Prim in [22] proposed two greedy strategies for solving efficiently the minimum spanning tree, but in the following is reported the

pseudocode of a greedy algorithm that works for any weighted matroid. The algorithm GREEDY takes as input a matroid $M = (E, I)$ and a weight function w and returns an optimal subset A.

```
set A = ∅
sort E[M] = {x_1, ..., x_k} into nonincreasing order
by weight w
FOR i = 1 to t
    IF A ∪ {x_i} ∈ I[M]
        set A = A ∪ {x_i}
return(A)
```

Greedy(*M*,*w*)

Like any other greedy algorithm, GREEDY always makes the choice that looks best at the moment. In fact, it considers in turn each element x_i belonging to $E[M]$, whose element are sorted into nonincreasing order by weight w and immediately adds x to the building set A if $A \cup \{x_i\}$ is still independent. Note that the returned set A is always independent, because it is initialized to the empty set, which is independent by definition of a matroid, and then at each iteration an element x_i is added to A while preserving the A's independence. A is also an optimal subset of the matroid M and therefore, a minimum spanning tree for the original graph G. To prove its optimality, it is enough to show that weighted matroids exhibit the two ingredients whose existence guarantee that a greedy strategy will solve optimally the given problem: the *greedy-choice property* and the *optimal substructure property*. The proof that matroids exhibit both these properties can be found in [4]. Generally speaking, the proof of the exhibition of the greedy-choice property consists of showing that a globally optimal solution can be obtained by making a locally optimal (greedy) choice. The proof examines a global optimal solution. It shows that the solution can be modified so that a greedy choice is made at the first step and that this choice reduces the original problem into an equivalent problem having smaller size. By induction, it is proved that a greedy choice can be made at each step. To show that making a greedy choice reduces the original problem into a similar but smaller problem reduces the proof of correctness to demonstrating that an optimal solution must exhibit optimal substructure. The optimal substructure property is exhibited by a given problem, if an optimal solution to the problem contains within it optimal solutions to subproblems. The validity of this property guarantee the applicability of greedy strategies as well as dynamic programming algorithms.

See also

► Oriented Matroids

References

1. Aigner M (1979) Combinatorial theory. Springer, Berlin
2. Bachem A, Kern W (1992) Linear programming duality: An introduction to oriented matroids. Springer, Berlin
3. Björner A, Las Vergnas M, Sturmfels B, White N, Ziegler GM (1993) Oriented matroids. Encycl Math Appl, vol 46. Cambridge Univ. Press, Cambridge
4. Cormen TH, Leiserson CE, Rivest RL (1990) Introduction to algorithms. MIT, Cambridge, MA
5. Edmonds J (1965) Lehman's switching game and a theorem of Tutte and Nash-Williams. J Res Nat Bureau Standards (B) 69:73–77
6. Edmonds J (1965) Maximum matching and a polyhedron with {0, 1} vertices. J Res Nat Bureau Standards (B) 69:125–130
7. Edmonds J (1965) Minimum partition of a matroid into independent subsets. J Res Nat Bureau Standards (B) 69:67–72
8. Edmonds J (1965) Paths, trees, and flowers. Canad J Math 17:449–467
9. Edmonds J (1967) Optimum branchings. J Res Nat Bureau Standards (B) 71:233–240
10. Edmonds J (1967) Systems of distinct representatives and linear algebra. J Res Nat Bureau Standards (B) 71:241–245
11. Edmonds J (1970) Submodular functions, matroids, and certain polyhedra. In: Guy R, Hanani H, Sauer N, Schönheim J (eds) Combinatorial Structures and Their Applications. Gordon and Breach, New York
12. Fujishige S (1991) Submodular functions and optimization. Ann Discret Math, vol 47. North-Holland, Amsterdam
13. Crapo HH, Rota GC (1970) On the foundations of combinatorial theory: Combinatorial geometries. preliminary MIT, Cambridge, MA
14. Kruskal JB (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. Proc Amer Math Soc 7:48–50
15. Kung JPS (1986) Numerically regular hereditary classes of combinatorial geometries. Geometriae Dedicata 21:85–105
16. Kung JPS (1986) A source book in matroid theory. Birkhäuser, Basel
17. Lawler EL (1976) Combinatorial optimization: Networks and matroids. Holt, Rinehart and Winston, New York

18. Lovász L, Plummer MD (1986) Matching theory. Akad. Kiadó, Budapest
19. Maffioli F (1990) Elementi di programmazione matematica. Masson, Paris
20. Murota K, Iri M, Nakamura M (1987) Combinatorial canonical form of layered mixed matrices and its application to block-triangularization of systems of linear/nonlinear equations. SIAM J Alg Discrete Meth 8:123–149
21. Oxley JG (1992) Matroid theory. Oxford Univ. Press, Oxford
22. Prim RC (1957) Shortest connection networks and some generalizations. Bell System Techn J 36:1389–1401
23. Recski A (1989) Matroid theory and its application in electrical networks and statics. Springer, Berlin
24. Schrijver A (1986) Theory of linear and integer programming. Wiley, New York
25. Tutte WT (1958) A homotopy theorem for matroids I–II. Trans Amer Math Soc 88:144–160; 161–174
26. Tutte WT (1959) Matroids and graphs. Trans Amer Math Soc 90:527–552
27. Tutte WT (1960) An algorithm for determining wheter a given binary matroid is graphic. Proc Amer Math Soc 11:905–917
28. Tutte WT (1961) On the problem of decomposing a graph into n connected factors. J London Math Soc 36:221–230
29. Tutte WT (1964) From matrices to graphs. Canad J Math 16:108–127
30. Tutte WT (1965) Lectures on matroids. J Res Nat Bureau Standards (B) 69:1–47
31. Tutte WT (1966) Connectivity in graphs. Univ. Toronto Press, Toronto
32. Tutte WT (1966) Connectivity in matroids. Canad J Math 18:1301–1324
33. Tutte WT (1971) Introduction to the theory of matroids. Amer. Elsevier, New York
34. Welsh DJA (1976) Matroid theory. Acad. Press, New York
35. White N (1986) Theory of matroids. Cambridge Univ. Press, Cambridge
36. White N (1987) Combinatorial geometries. Cambridge Univ. Press, Cambridge
37. White N (1991) Matroid applications. Cambridge Univ. Press, Cambridge
38. Whitney H (1935) On the abstract properties of linear dependence. Amer J Math 57:509–533

Maximum Constraint Satisfaction: Relaxations and Upper Bounds

MICHEL MINOUX[1], P. MAVROCORDATOS[2]
[1] Université Paris 6, Paris, France
[2] Algotheque and Université Paris 6, Paris, France

MSC2000: 90C10

Article Outline

Keywords
Introduction
CSP and MAX-CSP
MAX-CSP and the Radio Link Frequency Assignment Problem
A General Class of Relaxations for Computing MAX-CSP Bounds
Building Relaxations for RLFAP Using Maximum Cliques
Solving the Relaxed Problem $R[\mathcal{K}']$
Computational Results
See also
References

Keywords

Constraint satisfaction; Relaxation

Maximum constraint satisfaction problems (MAX-CSPs) generalize *maximum satisfiability* (MAX-SAT) to include cases where the variables are no longer restricted to binary (or Boolean) values.

MAX-CSP is *NP*-complete even in the special case of *binary CSPs*. Therefore designing procedures to compute upper bounds to the exact (unknown) optimum value (maximum number of satisfied constraints) is a relevant issue. Such bounds may be useful, in particular, to provide estimates of the quality of solutions obtained from various heuristic approaches.

This article describes a systematic way of computing upper bounds for large scale MAX-CSP instances such as those arising from the so-called *radio link frequency assignment problem* (RLFAP). After discussing the general relaxation principle and the basic procedure from which the bounds are derived, we present results of extensive computational experiments on series of 90 instances of RLFAP including both real test problems and randomly generated 'realistic' test problems (for sizes ranging from 396 variables and about 1700 constraints to 831 variables and about 4800 constraints).

These results clearly indicate that the proposed approach is practically useful to produce fairly accurate upper bounds for such large MAX-CSP problems.

Introduction

Constraint satisfaction problems (CSPs) may be viewed as a generalization of *satisfiability* (SAT) to include cases where, instead of taking binary values only (0–1 or true-false) the variables may take on a finite number (> 2) of given possible values.

For an infeasible CSP, a relevant question, both theoretically and practically, is to determine an assignment of values to variables such that the number of satisfied constraints is the *largest possible*. This is the so-called *maximum constraint satisfaction problem* (MAX-CSP), which generalizes in a natural way *maximum satisfiability* (MAX-SAT).

Since MAX-2SAT is *NP*-complete (see e. g. [12, pp. 259–260]) even the subclass of MAX-CSP corresponding to *binary CSPs* (those problems with constraints involving pairs of variables only) is *NP*-complete. Therefore, for very large instances such as those arising from practical applications (e. g. the RLFAP discussed below) one can only hope for approximate solutions using some of the currently available heuristic approaches such as: simulated annealing, tabu search, genetic algorithms, or local search of various kinds.

However, for many applications, getting an approximate solution without any information about the quality of this solution (e. g. measured by the difference between the cost of this solution and the optimal cost) may be of little value.

We address in this paper the problem of computing *upper bounds* to the optimum cost of MAX-CSP problems from which estimates on the quality of heuristic solutions can be derived.

The article is organized as follows. Basic definitions about CSPs and MAX-CSPs are recalled in the second section. Modeling the so-called radio link frequency assignment problem (RLFAP) in terms of CSP and MAX-CSP is addressed in the third section. Then we present a general class of relaxations for MAX-CSP problems and its specialization to the computation of MAX-CSP bounds for RLFAP. Finally results of extensive computational experiments carried out on series of both real test problems and realistic randomly generated test problems are presented. To our knowledge, this is the first time extensive computational results of this kind are reported for such large scale MAX-CSP problems.

CSP and MAX-CSP

A *constraint satisfaction problem* (CSP) is defined by specifying:

- a set of n variables $x_1, \ldots, x_n$;
- for each variable x_i, $i \in I = \{1, \ldots, n\}$ the domain of i, i. e. the (finite) set D_i of possible values for x_i;
- a set of K constraints φ_k, $k = 1, \ldots, K$. For each $k \in [1, K]$, constraint φ_k is defined by its *support set* (i. e. the subset $S_k = \text{supp}(\varphi_k)$ of indices of the variables involved in the constraint) and an *oracle* which, given any combination $\overline{x}_{[S_k]}$ of values for variables in S_k, answers TRUE if $\varphi_k(\overline{x}_{[S_k]}) =$ TRUE, i. e. if the combination is allowed, FALSE otherwise. (For any $S \subset \{1, \ldots, n\}$ and $x \in D_1 \times \cdots \times D_n$, $x_{[S]}$ denotes the vector x restricted to components in S.)

Given a CSP specified as above, we define a *free assignment* as any n-tuple $x \in D = D_1 \times \cdots \times D_n$. A *feasible assignment* (or *solution*) is a free assignment such that $\varphi_k(x_{[S_k]}) =$ TRUE for all $k = 1, \ldots, K$.

For simplicity, we restrict here to the case where each variable takes scalar values only (i. e. real or integer values), but we note that more general CSPs may be defined with variables taking, for instance, vector values.

The *arity of a constraint* φ_k is the cardinality of its support set: $|S_k| = |\text{supp}(\varphi_k)|$. A *binary CSP* is a constraint satisfaction problem in which $|\text{supp}(\varphi_k)| \leq 2$ for all $k = 1, \ldots, K$.

The constraint hypergraph associated with a given CSP is the hypergraph having vertex set $I = \{1, \ldots, n\}$ and edge set $\{S_1, \ldots, S_K\}$. In case of a binary CSP this is a graph.

The two examples below are interesting special cases of the general definition and show *NP* completeness of arbitrary CSPs.

Example 1 (Satisfiability) SAT is easily recognized as a special case of CSP where $\forall i$: $D_i = \{\text{TRUE, FALSE}\}$ and where there is a constraint φ_k corresponding to each clause C_k with $\varphi_k(x) =$ TRUE $\Leftrightarrow$ clause C_k is satisfied under truth assignment x.

Example 2 (Hypergraph q-coloring; see [2, Chap. 19]) Let $q > 1$ be a given integer and $H = [V, E]$ an hypergraph with vertex set V and edge set E. The problem is to assign one out of q colors to each vertex of H so that each edge of H has vertices of different colors. Clearly this may be formulated as a CSP problem where there is one variable x_i for each $v_i \in V$, with domain $D_i = \{1,$

$\ldots, q\}$, and one constraint φ_k for each edge $e_k = \{i_1, \ldots, i_p\} \in E$ such that $\varphi(\overline{x}_{i_1}, \ldots, \overline{x}_{i_p}) = \text{TRUE} \Leftrightarrow$ no two values in $\{\overline{x}_{i_1}, \ldots, \overline{x}_{i_p}\}$ are equal. Note that when H is a *graph* (i. e. $|e_k| = 2$ for all $e_k \in E$), the resulting CSP is a binary CSP.

For an infeasible CSP, one basic question is to determine a 'best possible' or 'least infeasible' assignment. If the criterion for quality (or degree of 'feasibility') of an assignment x is taken to be the number $\sigma(x)$ of constraints satisfied under that assignment, we are led to the so-called *MAX-CSP* problem:

- *Given*: a CSP defined by its variables $x_1, \ldots, x_n$, domains $D_1, \ldots, D_n$, and constraints $\varphi_1, \ldots, \varphi_K$.
- *Find*: $x \in D_1 \times \cdots \times D_n$ such that

$$\sigma(x) = \left|\left\{k \in [1, K]\colon\ \varphi_k(x_{[S_k]}) = \text{TRUE}\right\}\right|$$

is maximized.

Example 3 (MAX-SAT, MAX-2SAT) Clearly, MAX-SAT is a special case of MAX-CSP when the given CSP is a satisfiability problem. The associated decision problem is *NP*-complete even for the special case of MAX-2SAT ([13]), showing that MAX-CSP is *NP*-complete even for binary CSPs.

Heuristics for approximately solving the MAX-SAT problem have been proposed by [17,19,23,27]. A branch and bound algorithm for MAX-SAT based on probabilistic bounds is described in [3] with computational results up to 100 binary variables and 1000 clauses. The branch and cut algorithm described in [20] presents computational results for general Max-3SAT problems up to 100 binary variables and 575 clauses. For a recent survey on SAT and MAX-SAT, see [9].

For more general MAX-CSP problems, many heuristic approaches have been investigated such as *tabu search* ([4,7]), *simulated annealing* [5], *genetic algorithms* [18]. Exact Algorithms for random MAX-CSP problems were proposed in [11]. However in the computational experiments reported, the sizes of the problems for which exact optimal solutions were found are rather small (144 variables with domains of cardinality 4 and 646 constraints for the largest problems solved in [11]).

MAX-CSP and the Radio Link Frequency Assignment Problem

Operating large radio link telecommunication networks gives rise to the so-called *radio link frequency assignment problem* (RLFAP), which is to choose, for each transmission link, a specific operating frequency (among a given list of allowed values) while satisfying a list of noninterference constraints, (most constraints usually involving pairs of links). A CSP formulation of RLFAP is as follows: With n denoting the number of links, for each link $i = 1, \ldots, n$, there is an associated variable x_i representing the frequency to be assigned to link i. The domain D_i of x_i is the (finite) set of allowed frequencies for link i (frequencies are expressed in Hz, KHz, MHz or any other specified unit).

Any assignment $x \in S = D_1 \times \cdots \times D_n$ is not allowed because a number of constraints, called *noninterference constraints* have to be satisfied.

We will only consider here the case of *binary noninterference constraints* (i. e. involving only pairs of links), which is relevant to many applications of interest (see e. g. [15,16]). For a given pair of links i and j, two (exclusive) types of constraints are possible:

- equality constraints of the form

$$\text{(E)} \quad |x_i - x_j| = w_{ij};$$

- inequality constraints of the form

$$\text{(I)} \quad |x_i - x_j| \geq w_{ij}.$$

The real number w_{ij} which represents the requested slack or minimum requested slack between the two assigned frequencies will be called the *weight of the constraint*.

An instance of RLFAP is therefore specified by n (number of links), a list of domains $D_1, \ldots, D_n$ and a list of constraints i. e. a list of quadruples of the form (i, j, w_{ij}, T_{ij}) where: i, j are the indices of the two links involved, w_{ij} is the weight of the constraint, and T_{ij} its type ((E) or (I)). The *constraint graph* associated with an instance of RLFAP is defined as the undirected graph G with node set $\{1, \ldots, n\}$ and with an edge (i, j) for each constraint (i, j, w_{ij}, T_{ij}). We denote K the total number of constraints in an instance of RLFAP. Benchmarks of the RLFAP involving real instances up to 916 variables and 5744 constraints have been made publicly available in the context of the European Project CALMA

(see [8,15]). In those practical instances, the number of equality constraints (of type (E)) is never more than $n/2$, and assignments satisfying all of them can easily be found. We will denote $S' \subset S = D_1 \times \cdots \times D_n$ the set of all such assignments. All assignments $x \in S \setminus S'$ must be disregarded because they are physically meaningless, therefore, from now on, we will only consider assignments in S' as possible solution to RLFAP.

An assignment in S' which satisfies all constraints of type (I) will be called *feasible*. The feasibility version of RLFAP may therefore be stated as the following CSP:

- *Given*: an instance of RLFAP.
- *Question*: does there exist a feasible frequency assignment?
- *Answer*: yes or no and, if yes, output a feasible assignment x.

Efficient solution methods for RLFAP are of major interest to numerous practical applications in the context of civilian mobile communication networks as well as of military networks. Since the available spectrum is severely limited and the communication needs (traffic requirements) are continuously increasing, a high proportion of the instances of the RLFAP encountered in applications turn out to be *infeasible*.

When faced with an instance which is either infeasible or which is presumably infeasible (e. g. because running a heuristic solution method just failed to produce a feasible solution) a key question for the practitioner becomes to determine a 'best possible' or 'least infeasible' assignment.

This leads to the 'optimization version' of the RLFAP in the form of the following *MAX-CSP*:

- *Given*: an instance of RLFAP with n variables (links) and K constraints.
- *Question*: determine $x^* \in S'$ such that $\sigma(x^*)$ (number of satisfied constraints) is maximized:

$$\sigma(x^*) = \max_{x \in S'} \{\sigma(x)\}.$$

In view of the *NP*-completeness of MAX-CSP for binary CSPs, guaranteed optimal solutions to the above for large scale instances (such as those of the CELAR benchmarks) cannot be reasonably expected from currently available techniques in combinatorial optimization. A less ambitious, though practically relevant objective, addressed in the following section, is to try and obtain good *upper bounds* to an optimal solution value.

We note here that in the case where an upper bound $\widehat{\sigma}$ is found such that $\widehat{\sigma} < K$, then we can deduce that the given RLFAP has *no feasible solution*. Thus, an interesting by-product of computing bounds will be to produce *proofs of infeasibility* of a given instance of RLFAP. Clearly, such an information may be of considerable importance to the practitioner.

A General Class of Relaxations for Computing MAX-CSP Bounds

MAX-CSP may be reformulated as the discrete optimization problem

$$\begin{cases} \max & z = \sum_{k=1}^{K} y_k \\ \text{s.t.} & g_k(x) \geq y_k, \ \forall k = 1, \ldots, K, \\ & y_k = 0 \text{ or } 1, \ \forall k, \\ & x = (x_1, \ldots, x_n)^\top \in S'. \end{cases} \tag{1}$$

In the above, for all $k = 1, \ldots, K$, $g_k(x) \geq 1$ if $\varphi_k(x_{[S_k]})$ = TRUE, and $g_k(x) < 1$ if $\varphi_k(x_{[S_k]})$ = FALSE. Note that in the case of RLFAP, this specializes to: $g_k(x) = |x_i - x_j|/w_k$, where x_i and x_j are the two variables involved in constraint k, and w_k the weight of constraint k.

A *relaxation* of an optimization problem such as (1) is obtained by replacing its solution set by a larger solution set. Clearly if the relaxed problem can be solved exactly (i. e. to guaranteed optimality) then its optimal objective function value is an upper bound (in case of maximization) to the optimum objective function value of the original problem.

There exists a number of standard ways of relaxing an optimization problem such as (1), e. g. using Lagrangian relaxation (e. g. [10]) or considering the so-called *continuous relaxation* of some of the variables (e. g. relaxing the constraints on the y_k variables in (1) to $0 \leq y_k \leq 1$). However, in our treatment of RLFAP, those standard relaxations have not been considered because they do not give rise to easily solvable relaxed problems. We therefore investigated a different approach according to the following general principle.

The relaxations we consider are based on the identification of those parts of the constraint graph or hypergraph which are responsible for the infeasibility of the whole problem. Preliminary computational results obtained in [25] have shown that, at least for MAX-CSP

problems deriving from RLFAP, it is most often possible to identify in a given instance an infeasible induced subproblem of sufficiently reduced size to make the corresponding MAX-CSP bound computable in reasonable time.

This suggests to consider relaxations of (1) formed by subproblems induced by properly chosen subsets of constraints. Thus, if $\mathcal{K}' \subset \mathcal{K} = \{1, \ldots, K\}$ is the subset of constraints chosen, the induced relaxation considered is:

$$\begin{cases} \max & z = \sum_{k=1}^{K} y_k \\ \text{s.t.} & g_k(x) \geq y_k, \ \forall k \in \mathcal{K}', \\ & y_k = 0 \text{ or } 1, \ \forall k = 1, \ldots, K, \\ & x \in S'. \end{cases} \tag{2}$$

Note that, in an optimal solution to (2)

$$k \in \mathcal{K} \backslash \mathcal{K}' \Rightarrow y_k = 1.$$

Therefore $\bar{z}$, the optimum objective function value of (2), may be rewritten as:

$$\bar{z} = K - \left|\mathcal{K}'\right| + \bar{z}',$$

where $\bar{z}'$ is the optimum value of the problem:

$$R[\mathcal{K}'] \quad \begin{cases} \max & z' = \sum_{k \in \mathcal{K}'} y_k \\ \text{s.t.} & g_k(x) \geq y_k, \ \forall k \in \mathcal{K}', \\ & y_k = 0 \text{ or } 1, \ \forall k \in \mathcal{K}', \\ & x \in S'. \end{cases}$$

Clearly, the constraint graph or hypergraph G' corresponding to a relaxation $R[\mathcal{K}']$ is deduced from the constraint graph or hypergraph G by deleting all edges associated with the constraints in $\mathcal{K} \setminus \mathcal{K}'$. Also observe that if G' has several distinct connected components, then the solution of $R[\mathcal{K}']$ *decomposes into independent subproblems*, one for each connected component.

If the constraint graph or hypergraph G' is of sufficiently small size, then it is possible to solve $R[\mathcal{K}']$ exactly, and the optimum solution value obtained clearly leads to an upper bound to the optimum value of the original problem. When G' is too large to get the exact optimal solution value of $R[\mathcal{K}']$ then we will content ourselves with getting an *upper bound* to this exact optimal value (see the procedure SOLVE.RELAX below). Clearly, any such upper bound still provides a valid upper bound to the original problem. Of course, in the above approach, the quality of the bound derived from $R[\mathcal{K}']$ essentially depends on how to select the subset $\mathcal{K}'$. We now describe the selection procedure which has been used in our computational experiments.

Building Relaxations for RLFAP Using Maximum Cliques

We now specialize the general relaxation scheme described above to derive bounds for RLFAP. The presentation below improves and extends our preliminary work in [25].

The basic idea of our selection procedure for choosing $\mathcal{K}' \subset \mathcal{K}$ is that, for RLFAP, infeasibility is more likely to occur on subsets of links which are all mutually constrained, i. e. on subsets of links which induce a *clique* (complete subgraph) in the constraint graph. Since for RLFAP the constraint graphs arising from applications are always very sparse (less than 1% density for the CELAR instances), it is known that finding a clique of maximum cardinality can be efficiently done even using simple approaches such as implicit enumeration.

In [6] an efficient implicit enumeration based algorithm with good computational results for large sparse graphs up to 3000 vertices is described; however, it assumes very small maximum clique sizes (in the computational results presented in [6], *maximum clique* sizes do not exceed 11, and the running times seem to increase extremely fast with this parameter). Unfortunately, in view of the fact that, for our large RLFAP instances, the maximum clique sizes turned out to be commonly in the range [12, 25], the above algorithm could not be used.

We therefore worked out a different implementation of the implicit enumeration technique which allowed us to find guaranteed maximum cliques for all the test problems treated within acceptable computing times (see results at the end of the paper). Using this maximum clique algorithm, the procedure for building a relaxation to MAX-CSP for RFLAP is as follows.

The heuristic solution method used in our experiments to implement step b1) is a variant of *local search* consisting in iteratively improving an initial starting solution; at each iteration an exact tree search is car-

ried out to find an optimal solution to a subproblem involving only a few variables. In our computational experiments we observed that the impact of the quality of the heuristic solutions produced at step b1) on the quality of the relaxation obtained at the end of BUILD.RELAX was practically negligible (the main reason for this is that $\overline{\sigma}$ is only used as a stopping criterion in the process of successive extraction of maximum cliques). The computational results shown below confirm that bounds of good average quality indeed result from the above construction.

a Set: $\overline{G} = [\overline{X}, \overline{U}] \leftarrow G$ (the initial constraint graph), $i \leftarrow 0$

b Current step:

1 Apply a heuristic algorithm to get a good approximate solution to MAX-CSP on $\overline{G}$. Let $\overline{\sigma}$ denote the number of constraints satisfied in this solution.
IF $\overline{\sigma} = |\overline{U}|$ go to c) (end of the construction),
ELSE set: $i \leftarrow i + 1$.

2 Look for a maximum clique on $\overline{G}$. Let C_i be the clique obtained, with node set $N(C_i)$ and edge set $E(C_i)$.

3 Let $\overline{G'}$ denote the subgraph of $\overline{G}$ induced by $\overline{X} \setminus N(C_i)$ (obtained from $\overline{G}$ by deleting all edges having at least one endpoint in $N(C_i)$).
Set $\overline{G} \leftarrow \overline{G'}$ and return to b).

c IF $i = 0$, the problem is feasible and step b1) produces an assignment satisfying all the constraints. Terminate.
ELSE the relaxation $R[\mathcal{K}']$ obtained corresponds to the set $\mathcal{K}'$ of all constraints in $\cup_{j=1}^{i} E(C_j)$.

Procedure BUILD.RELAX

Solving the Relaxed Problem $R[\mathcal{K}']$

In order to solve the relaxed problem $R[\mathcal{K}']$ we use a basic procedure called FIND.SOLUTION($R[\mathcal{K}']$, θ) which, for any integer value $\theta \in [1, |\mathcal{K}'|]$, answers YES or NO depending on whether there exists a solution to $R[\mathcal{K}']$ with objective function value $z \geq \theta$ or not. In case of a YES answer, the procedure also exhibits the corresponding solution. We assume that this procedure is *exact* i. e. always finds the right answer. Clearly, any value of θ leading to a NO answer produces an *upper bound* to the optimal solution value of $R[\mathcal{K}']$.

The procedure SOLVE.RELAX($R[\mathcal{K}']$) determines a decreasing sequence of upper bounds to the optimal value of $R[\mathcal{K}']$ until either termination is obtained (at step c)) or the maximum computation time has been reached.

In the former case, the exact optimum solution value to $R[\mathcal{K}']$ is obtained; in the latter case, only an upper bound to this optimal value is produced.

a Initialization: Set $\theta \leftarrow | \mathcal{K}' |$.

b Current step:
Apply FIND.SOLUTION($R[\mathcal{K}']$, θ)
IF the answer is NO,
THEN set $\theta \leftarrow \theta - 1$ and return to b).
ELSE perform step c).

c A YES answer has been obtained at step b): θ is the optimal solution value to $R[\mathcal{K}']$. Terminate.

Procedure SOLVE.RELAX($R[\mathcal{K}']$)

When G', the constraint graph of $R[\mathcal{K}']$ has several distinct connected components corresponding to subsets of constraints, $\mathcal{K}'_1, \ldots, \mathcal{K}'_p$, then solving $R[\mathcal{K}']$ decomposes into the solution of several smaller subproblems $R[\mathcal{K}_1']$, …, $R[\mathcal{K}_p']$. In the procedure SOLVE.RELAX, this decomposability may be exploited in various possible ways. In our implementation, this is done by organizing the computation into *phases* numbered $t = 0, 1, \ldots$. The current upper bound value UB is initialized by: UB $\leftarrow |\mathcal{K}'|$. The current phase t consists in running the procedure FIND.SOLUTION on each of the subproblems $R[\mathcal{K}'_j]$, $j = 1, \ldots, p$, with the parameter $\theta = |\mathcal{K}'_j| - t$. Each time a NO answer is obtained, UB is updated by UB $\leftarrow$ UB $- 1$. Clearly with the above process, when a YES answer has been obtained for some subproblem $R[\mathcal{K}_j']$ during phase t, this subproblem should not be considered any more at later phases $t' > t$. The computation stops either at the end of a phase during which a YES answer has been obtained for all subproblems; or when a user-specified time limit has been reached.

The basic procedure FIND.SOLUTION has been implemented as a classical depth first tree search process of the implicit enumeration type, (achieved by

means of a recursive C function). Since getting the exact answer (YES or NO) is essential to the derivation of our bounds, the procedure FIND.SOLUTION is run until full completion of the tree search (i. e. when all the nodes of the tree have been explored implicitly or explicitly).

Computational Results

In order to validate the above described approach, systematic computational experiments have been carried out on two series of test problems.

The first set was composed of 15 infeasible real problems which arose from actual network engineering studies carried out on three distinct large radio link networks (one in the 2GHz frequency range, one in the 2, 5GHz frequency range and one in the 4GHz frequency range).

The second series concerned a set of $5 \times 15 = 75$ 'realistic' test problems generated by applying some random perturbation to the above 15 real problems. More precisely, each problem of the second series is generated from one problem of the first series by changing the weight w_{ij} of each inequality constraint of the form: $|x_i - x_j| \geq w_{ij}$ to: $\widetilde{w}_{ij} = w_{ij} \times (\alpha + \beta\Phi)$ where Φ is a pseudorandom number drawn from a uniform distribution on [0, 1] and α, β are chosen parameters (of course the pseudorandom drawing is assumed to be independent from one constraint to the next).

Maximum Constraint Satisfaction: Relaxations and Upper Bounds, Table 1

Prob. #	n	K	NF	Relaxation # var.	Relaxation # const.
1	680	2389	8	44	257
2	680	3367	16	38	339
3	680	4103	24	84	671
4	680	2725	8	74	490
5	680	2576	8	46	311
6	680	2470	8	44	284
7	831	3451	16	16	113
8	831	4802	24	33	248
9	396	1792	12	70	375
10	396	1792	12	70	375
11	396	1792	12	70	375
12	396	1792	12	70	375
13	396	1792	12	70	375
14	396	1792	12	70	375
15	396	1792	12	70	375

Maximum Constraint Satisfaction: Relaxations and Upper Bounds, Table 2

Prob. #	HS	Best upper bound obtained within 15s	5'	1 h
1	2376	2387	2385	2383
2	3358	3367	3366	3365
3	4090	4102	4098	4098
4	2700	2720	2713	2708
5	2559	2571	2569	2564
6	2457	2467	2464	2459
7	3440	3450	3450	3450
8	4781	4800	4800	4799
9	1762	1786	1780	1777
10	1759	1786	1780	1776
11	1761	1786	1780	1778
12	1764	1786	1780	1776
13	1761	1786	1780	1775
14	1757	1786	1780	1775
15	1764	1786	1783	1777

Table 1 presents the characteristics of the 15 real test problems treated, numbered 1 to 15 and provides for each problem: number of variables (n), number of constraints (K), number of distinct frequencies used (NF) and the main characteristics of the relaxed subproblem obtained from the procedure BUILD.RELAX: number of variables #var, and number of constraints #const.

Table 3 presents in a similar way the characteristics of the $5 \times 15 = 75$ test problems deduced from the previous ones by random perturbation. The 5 instances corresponding to each basic problem i are numbered $i_1, \ldots, i_5$. For each instance the values of the parameters α and β used to generate the instance are displayed together with the characteristics (number of variables, number of constraints) of the relaxed subproblem produced by BUILD.RELAX.

The computation times taken to construct the relaxed subproblems (using BUILD.RELAX) on the problems of Tables 1 and 3, are all between 5 minutes to 35 minutes with an average of about 12 minutes.

Table 2 shows the results obtained on the 15 real test problems of Table 1 and Table 4 shows the results for the 5×15 problems of Table 3. The computer used

Maximum Constraint Satisfaction: Relaxations and Upper Bounds, Table 3

Prob. #	α	β	Relaxation #var.	Relaxation #const.	Prob. #	α	β	Relaxation #var.	Relaxation #const.
1_1	0, 5	1	42	257	9_1	0, 2	1, 6	12	375
1_2	0, 5	1	42	261	9_2	0, 2	1, 6	12	66
1_3	0, 8	0, 4	42	261	9_3	0, 2	1, 6	48	66
1_4	0, 8	0, 4	42	261	9_4	0, 2	1, 6	24	264
1_5	0, 8	0, 4	42	261	9_5	0, 2	1, 6	36	132
2_1	0, 5	1	38	339	10_1	0, 2	1, 6	36	375
2_2	0, 5	1	38	339	10_2	0, 2	1, 6	12	198
2_3	0, 8	0, 4	38	339	10_3	0, 2	1, 6	48	66
2_4	0, 8	0, 4	38	339	10_4	0, 2	1, 6	24	264
2_5	0, 8	0, 4	38	339	10_5	0, 2	1, 6	36	132
3_1	0, 5	1	54	671	11_1	0, 2	1, 6	36	375
3_2	0, 5	1	70	460	11_2	0, 2	1, 6	12	198
3_3	0, 8	0, 4	84	480	11_3	0, 2	1, 6	48	66
3_4	0, 8	0, 4	84	671	11_4	0, 2	1, 6	24	264
3_5	0, 8	0, 4	54	671	11_5	0, 2	1, 6	36	132
4_1	0, 5	1	74	490	12_1	0, 2	1, 6	24	375
4_2	0, 5	1	74	490	12_2	0, 2	1, 6	12	132
4_3	0, 8	0, 4	74	490	12_3	0, 2	1, 6	48	66
4_4	0, 8	0, 4	74	490	12_4	0, 2	1, 6	24	264
4_5	0, 8	0, 4	74	490	12_5	0, 2	1, 6	36	162
5_1	0, 5	1	46	311	13_1	0, 2	1, 6	36	375
5_2	0, 5	1	46	311	13_2	0, 2	1, 6	12	198
5_3	0, 8	0, 4	46	311	13_3	0, 2	1, 6	48	66
5_4	0, 8	0, 4	46	311	13_4	0, 2	1, 6	24	264
5_5	0, 8	0, 4	46	311	13_5	0, 2	1, 6	36	132
6_1	0, 5	1	44	284	14_1	0, 2	1, 6	36	375
6_2	0, 5	1	44	284	14_2	0, 2	1, 6	12	198
6_3	0, 8	0, 4	44	284	14_3	0, 2	1, 6	48	66
6_4	0, 8	0, 4	44	284	14_4	0, 2	1, 6	24	264
6_5	0, 8	0, 4	44	284	14_5	0, 2	1, 6	36	132
7_1	0, 8	0, 4	16	113	15_1	0, 2	1, 6	24	375
7_2	0, 8	0, 4	16	113	15_2	0, 2	1, 6	12	132
7_3	0, 8	0, 4	16	113	15_3	0, 2	1, 6	48	66
7_4	0, 8	0, 4	16	113	15_4	0, 2	1, 6	24	264
7_5	0, 8	0, 4	16	113	15_5	0, 2	1, 6	36	132
8_1	0, 5	1	33	248					
8_2	0, 5	1	33	248					
8_3	0, 5	1	33	248					
8_4	0, 5	1	33	248					
8_5	0, 5	1	33	248					

was a PC Pentium 166 workstation with 32Mb RAM. For each problem we provide: HS, the best heuristic solution value obtained (number of satisfied constraints); the best upper bounds obtained after 15 seconds, 5 minutes and 1 hour. The results in Table 2 confirm that our approach is practical to consistently produce good bounds for real RLFAP instances within acceptable solution times.

Maximum Constraint Satisfaction: Relaxations and Upper Bounds, Table 4

Prob. #	HS	Best upper bound obtained within			Prob. #	HS	Best upper bound obtained within		
		15 s	5'	1 h			15 s	5'	1 h
1_1	2376	2386	2383	2378	9_1	1779	1791	1791	1791
1_2	2376	2386	2383	2378	9_2	1777	1791	1790	1789
1_3	2376	2386	2383	2378	9_3	1774	1788	1788	1785
1_4	2376	2386	2383	2378	9_4	1777	1790	1789	1789
1_5	2376	2386	2383	2378	9_5	1779	1789	1787	1787
2_1	3358	3366	3365	3365	10_1	1780	1789	1788	1787
2_2	3358	3367	3365	3365	10_2	1780	1791	1790	1788
2_3	3358	3367	3366	3365	10_3	1776	1788	1787	1785
2_4	3358	3367	3365	3365	10_4	1778	1790	1789	1789
2_5	3358	3367	3366	3365	10_5	1777	1789	1788	1788
3_1	4081	4103	4101	4101	11_1	1783	1789	1789	1789
3_2	4081	4102	4101	4101	11_2	1780	1791	1789	1788
3_3	4086	4102	4098	4098	11_3	1777	1788	1788	1786
3_4	4086	4102	4098	4098	11_4	1780	1790	1789	1789
3_5	4088	4102	4101	4101	11_5	1779	1789	1788	1787
4_1	2700	2720	2713	2708	12_1	1779	1790	1790	1789
4_2	2700	2720	2713	2708	12_2	1780	1791	1790	1789
4_3	2700	2720	2713	2708	12_3	1777	1788	1788	1786
4_4	2700	2720	2713	2708	12_4	1780	1790	1789	1787
4_5	2700	2720	2713	2708	12_5	1778	1789	1788	1787
5_1	2559	2571	2569	2564	13_1	1782	1789	1789	1788
5_2	2559	2572	2569	2564	13_2	1777	1790	1789	1789
5_3	2559	2572	2569	2564	13_3	1776	1788	1787	1786
5_4	2559	2573	2569	2564	13_4	1779	1790	1789	1789
5_5	2559	2573	2569	2564	13_5	1777	1789	1788	1788
6_1	2457	2467	2464	2459	14_1	1782	1789	1789	1788
6_2	2457	2467	2464	2459	14_2	1775	1791	1789	1789
6_3	2457	2467	2464	2459	14_3	1775	1788	1787	1786
6_4	2457	2467	2464	2459	14_4	1779	1791	1789	1789
6_5	2457	2467	2464	2459	14_5	1776	1789	1788	1788
7_1	3438	3450	3450	3450	15_1	1780	1790	1790	1789
7_2	3437	3450	3450	3450	15_2	1779	1791	1789	1788
7_3	3421	3430	3430	3430	15_3	1777	1788	1788	1788
7_4	3414	3424	3424	3424	15_4	1781	1790	1789	1788
7_5	3436	3450	3450	3450	15_5	1780	1789	1788	1788
8_1	4780	4800	4800	4799					
8_2	4783	4800	4800	4799					
8_3	4778	4800	4800	4799					
8_4	4781	4800	4800	4799					
8_5	4781	4800	4800	4799					

From Tables 2 and 4, it is seen that for all the instances treated, the difference between the heuristic solution values HS and the best upper bounds obtained are always quite small. More precisely for all the examples treated, the ratio $R = (\text{UB} - \text{HS})/\text{UB}$ is most often well below 1% (Problem 14 in Table 2 is the only

one for which $R > 1\%$). We note that since HS is only a *lower bound*, R is a pessimistic estimate of the relative difference between the best upper bound obtained and the optimal, unknown, solution value.

Also, from Table 4, it is seen that the results obtained appear to be fairly stable, in spite of the importance of the perturbations applied to generate the corresponding 75 instances. In addition to practical applicability, and efficiency, this clearly shows good stability and robustness in the behavior of our algorithms. To our knowledge, this is the first time a systematic way of deriving upper bounds to such large scale MAX-CSP problems has been implemented and fully tested.

To conclude, let us mention that, in view of the results obtained, the techniques described here have been included in an industrial software tool for radio network engineering developed by the French MOD (DGA/CELAR).

See also

► Frequency Assignment Problem
► Graph Coloring

References

1. Babel L, Tinhofer G (1990) A branch and bound algorithm for the maximum clique problem. ZOR: Methods and Models of Oper Res 34:207–217
2. Berge C (1973) Graphes et hypergraphes, 2nd edn. Dunod, Paris
3. Boros E, Prékopa A (1988) Probabilistic bounds and algorithms for the maximum satisfiability problem. RUTCOR Res. Report Rutgers Univ. New Jersey (USA) RRR#17-88
4. Bouju A, Boyce JF, Dimitropoulos CHD, von Scheidt G, Taylor JG (1995) Tabu search for the radio link frequency assignment problem. Proc. Conf. on Applied Decision Technologies: Modern Heuristic methods, Brumel Univ., Uxbridge (UK), Uxbridge, UK pp 233–250 work carried out in the CALMA PROJECT.
5. Bourret P (1995) Simulated annealing. Deliverable 2.3 of the CALMA Project. Report 3/3507.00/DERI-ONERA-CERT, CALMA
6. Carraghan R, Pardalos PM (1990) An exact algorithm for the maximum clique problem. Oper Res Lett 9:375–382
7. Castelino DJ, Hurley S, Stephens NM (1996) A tabu search algorithm for frequency assignment. Ann Oper Res 63:301–319
8. CELAR (1994) Radio link frequency assignment problem benchmark, CELAR, ftp.cs.unh.edu/pub/csp/archive/code/benchmarks
9. Du D-Z, Gu J, Pardalos PM (eds) (1997) Satisfiability problem: Theory and applications. DIMACS, vol 35. Amer Math Soc
10. Fisher ML (1981) The Lagrangian relaxation method for solving integer programming problems. Managem Sci 27:11–18
11. Freuder EC, Wallace RJ (1992) Partial constraint satisfaction. Artif Intell 58:21–70
12. Garey MR, Johnson DS (1979) Computers and intractability. A guide to the theory of NP-completeness. Freeman, New York
13. Garey MR, Johnson DS, Stokmeyer L (1976) Some simplified NP-complete graph problems. Theoret Comput Sci 1:237–267
14. Gondran M, Minoux M (1995) Graphes et algorithmes, 3rd edn. Eyrolles, Paris
15. Hajema W, Minoux M, West C (June 25, 1992) CALMA project specification. Statement of the Radio Link Frequency Assignment Problem. Appendix 3 to EUCLID RTP, 6-4 Implementing Arrangement
16. Hale WK (1980) Frequency assignment theory and applications. Proc IEEE 68(12):1497–1514
17. Hansen P, Jaumard B (1987) Algorithms for the maximum satisfiability problem. RUTCOR Res. Report Rutgers Univ. New Jersey (USA) RRR#43-87
18. Hurley S, Thiel SU, Smith DH (1996) A comparison of local search algorithms for radio link frequency assignment problems. Proc. ACM Symposium on Applied Computing, pp 251–257
19. Johnson DS (1974) Approximation algorithms for combinatorial problems. J Comput Syst Sci 9:256–278
20. Joy S, Mitchell J, Borcher SB (1997) A branch and cut algorithm for MAX-SAT and weighted MAX-SAT. In: Satisfiability Problem: Theory and Appl. In: DIMACS, vol 35. Amer Math Soc, pp 519–536
21. Kumar V (1992) Algorithms for constraint satisfaction problems: A survey. AI Magazine Spring:32–44
22. Lanfear TA (1989) Graph theory and radio frequency assignment. NATO EMC Analysis Project, vol 5. NATO, Brussels
23. Lieberherr KJ (1982) Algorithmic extremal problems in combinatorial optimization. J Algorithms 3:225–244
24. Lieberherr KJ, Specker E (1981) Complexity of partial satisfaction. J ACM 28(2):411–421
25. Mavrocordatos P, Minoux M (1995) Allocation de ressources dans les réseaux (frequency allocation in networks). Final Techn. Report CELAR contract #0114193. Résolution de problèmes d'optimisation combinatoire pour application à l'allocation optimisée de fréquences dans les grands réseaux.
26. Pardalos PM, Rodgers GP (1992) A branch and bound algorithm for the maximum clique problem. Comput Oper Res 19(5):363–375

27. Poljak S, Turzik D (1982) A polynomial algorithm for constructing a large bipartite subgraph, with an application to a satisfiability problem. Canad J Math XXXIV(3):519–524
28. Resende MGC, Pitsoulis LS, Pardalos PM (1997) Approximate solution of weighted MAX-SAT problems using GRASP. Satisfiability Problem: Theory and Appl. In: DIMACS, vol 35. Amer Math Soc, pp 393–405
29. Roberts FS (1991) T-colorings of graphs; recent results and open problems. Discret Math 93:229–245
30. Smith DH, Hurley S (1997) Bounds for the frequency assignment problem. Discret Math 167/168:571–582
31. Smith DH, Hurley S, Thiel SU (1998) Improving heuristics for the frequency assignment problem. Europ J Oper Res 107:76–86
32. de Werra D, Gay Y (1994) Chromatic scheduling and frequency assignment. Discrete Appl Math 49:165–174

Maximum Cut Problem, MAX-CUT

CLAYTON W. COMMANDER
Air Force Research Laboratory, Munitions Directorate, and Dept. of Industrial and Systems Engineering, University of Florida, Gainesville, USA

Article Outline

Introduction
  Organization
  Idiosyncrasies
Formulation
Methods
  Review of Solution Approaches
  C-GRASP Heuristic
Conclusions
See also
References

Introduction

The MAXIMUM CUT problem (MAX-CUT) is one of the simplest graph partitioning problems to conceptualize, and yet it is one of the most difficult combinatorial optimization problems to solve. The objective of MAX-CUT is to partition the set of vertices of a graph into two subsets, such that the sum of the weights of the edges having one endpoint in each of the subsets is maximum. This problem is known to be $\mathcal{NP}$-complete [18,27]; however, it is interesting to note that the inverse problem, i. e., that of looking for the minimum cut in a graph is solvable in polynomial time using network flow techniques [1]. MAX-CUT is an important combinatorial problem and has applications in many fields including VLSI circuit design [9,32] and statistical physics [5]. For other applications, see [16,21]. For a detailed survey of MAX-CUT, the reader can refer to [33].

Organization

In this paper, we introduce the MAXIMUM CUT problem and review several heuristic methods which have been applied. In Subsect. "C-GRASP Heuristic" we describe the implementation of a new heuristic based optimizing a quadratic over a hypercube. The heuristic is designed under the C-GRASP (Continuous Greedy Randomized Adaptive Search Procedure) framework. Proposed by Hirsch, Pardalos, and Resende [23], C-GRASP is a new stochastic metaheuristic for continuous global optimization problems. Numerical results are presented and compared with other heuristics from the literature.

Idiosyncrasies

We conclude this section by introducing the symbols and notations we will employ throughout this paper. Denote a graph $G = (V, E)$ as a pair consisting of a set of vertices V, and a set of edges E. Let the map $w\colon E \mapsto \mathbb{R}$ be a weight function defined on the set of edges. We will denote an edge-weighted graph as a pair (G,w). Thus we can easily generalize an un-weighted graph $G = (V, E)$ as an edge-weighted graph (G,w), by defining the weight function as

$$w_{ij} := \begin{cases} 1, & \text{if } (i,j) \in E\,, \\ 0, & \text{if } (i,j) \notin E\,. \end{cases} \tag{1}$$

We use the symbol "$b := a$" to mean "the expression a defines the (new) symbol b". Of course, this could be conveniently extended so that a statement like "$(1-\epsilon)/2 := 7$" means "define the symbol ϵ so that $(1-\epsilon)/2 = 7$ holds". We will employ the typical symbol S^c to denote the complement of the set S; further let $A \setminus B$ denote the set-difference, $A \cap B^c$. Agree to let the expression $x \leftarrow y$ mean that the value of the variable y is assigned to the variable x. Finally, to denote the cardinality of a set S, we use $|S|$. We will use **bold** for words

which we define, *italics* for emphasis, and SMALL CAPS for problem names. Any other locally used terms and symbols will be defined in the sections in which they appear.

Formulation

Consider an undirected edge-weighted graph (G,w), where $G = (V, E)$ is the graph, and w is the weight function. A **cut** is defined as a partition of the vertex set into two disjoint subsets S and $\bar{S} := V \setminus S$. The **weight** of the cut $(S, \bar{S})$ is given by the function $W\colon S \times \bar{S} \mapsto \mathbb{R}$ and is defined as

$$W(S, \bar{S}) := \sum_{i \in S, j \in \bar{S}} w_{ij} \,. \tag{2}$$

For an edge-weighted graph (G,w), a **maximum cut** is a cut of maximum weight and is defined as

$$MC(G, w) := \max_{\forall S \subseteq V} W(S, V \setminus S) \,. \tag{3}$$

We can formulate MAX-CUT as the following integer quadratic programming problem:

$$\max \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - y_i y_j) \tag{4}$$

subject to:

$$y_i \in \{-1, 1\}, \qquad \forall i \in V \,. \tag{5}$$

To see this, notice that each subset $V \supseteq S := \{i \in V : y_i = 1\}$ induces a cut $(S, \bar{S})$ with corresponding weight equal to

$$W(S, \bar{S}) = \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - y_i y_j) \,. \tag{6}$$

An alternative formulation of MAX-CUT based on the optimization of a quadratic over the unit hypercube was given by Deza and Laurent in [12].

Theorem 1 *Given a graph $G = (V, E)$ with $|V| = n$, the optimal objective function value of the* MAXIMUM CUT *problem is given by*

$$\max_{x \in [0,1]^n} x^{\mathrm{T}} W(e - x) \,, \tag{7}$$

where $W = [w_{ij}]_{i,j=1}^{n}$ is the matrix of edge weights, and $e := [1, 1, \ldots, 1]^{\mathrm{T}}$ is the unit vector.

Proof 1 Let

$$f(x) := x^{\mathrm{T}} W(e - x) \tag{8}$$

denote the objective function from Eq. (7). To begin with, notice that the matrix W has a zero diagonal, i. e., $w_{ii} = 0,\ \forall i \in 1, 2, \ldots, n$. This implies that $f(x)$ is linear with respect to each variable, and thus there always exists an optimal solution, x^* of (7) such that $x^* \in \{0, 1\}^n$. Therefore, we have shown that

$$\max_{x \in [0,1]^n} x^{\mathrm{T}} W(e - x) = \max_{x \in \{0,1\}^n} x^{\mathrm{T}} W(e - x). \tag{9}$$

The next step is to show that there is a bijection between binary vectors of length n and cuts in G. Consider *any* binary vector $\hat{x} \in \{0, 1\}^n$. Now suppose we partition the vertex set V into two disjoint subsets $V_1 := \{i | \hat{x}_i = 0\}$ and $V_2 := \{i | \hat{x}_i = 1\}$. Then, evaluating the objective function we have

$$f(\hat{x}) = \sum_{(i,j) \in V_1 \times V_2} w_{ij}, \tag{10}$$

which is equal to $W(V_1,V_2)$, the value of the cut defined by the partition of $V = V_1 \bigcup V_2$ (see Eq. (2) above).

Alternatively, consider *any* partition of V into two disjoint subsets $V_1, V_2 \subseteq V$. That is

$$V = V_1 \bigcup V_2 \quad \text{and} \quad V_1 \bigcap V_2 = \emptyset \,.$$

Now, we can construct the vector $\hat{x}$ as follows:

$$\hat{x}_i = \begin{cases} 1, & \text{if } i \in V_1 \\ 0, & \text{if } i \in V_2 \,. \end{cases} \tag{11}$$

Once again, evaluating the objective function on $\hat{x}$, we have

$$f(\hat{x}) = \sum_{(i,j) \in V_1 \times V_2} w_{ij} \,. \tag{12}$$

Hence $f(\hat{x}) = W(V_1, V_2)$ and we have the result.[1] Alas, we have shown the bijection between binary n-vectors

[1] Notice that the result holds even if (without the loss of generality) $V_1 = V$ and $V_2 = \emptyset$. In this case, a cut induced by (V_1, V_2) will be a maximum cut if $w_{ij} \le 0, \quad \forall i, j \in V$.

and cuts in G. In summary, we have

$$\begin{aligned}&\max_{x\in[0,1]^n} x^{\mathrm{T}} W(e-x)\\ &= \max_{x\in\{0,1\}^n} x^{\mathrm{T}} W(e-x)\\ &= \max_{V=V_1\bigcup V_2, V_1\bigcap V_2=\emptyset} \sum_{(i,j)\in V_1\times V_2} w_{ij}\,.\end{aligned}$$

□

There are several classes of graphs for which MAX-CUT is solvable in polynomial time [25]. These include planar graphs [11], weakly bipartite graphs with nonnegative edge weights [20], and graphs without K_5 minors [4]. The general problem however is known to be $\mathcal{APX}$-complete [31]. This implies that unless $\mathcal{P}=\mathcal{NP}$, MAX-CUT does not admit a polynomial time approximation scheme [30].

Methods

The MAXIMUM CUT problem is one of the most well-studied discrete optimization problems [27]. Since the problem is $\mathcal{NP}$-hard in general, there has been an incredible amount of research done in which heuristic techniques have been applied. Before we present the new heuristic approach, we review some of the prior work that has been done.

Review of Solution Approaches

There have been many semidefinite and continuous relaxations based on this formulation. This was first shown by Lovász in [28]. In 1995, Goemans and Williamson [19] used a semidefinite relaxation to achieve an approximation ratio of .87856. This implication of this work is significant for two reasons. The first is of course, the drastic improvement of the best known approximation ratio for MAX-CUT of 0.5 which had not been improved in over 20 years [36]. Secondly, and perhaps more significantly is that until 1995, research on approximation algorithms for nonlinear programming problems did not receive much attention. Motivated by the work of Goemans and Williamson, semidefinite programming techniques were applied to an assortment of combinatorial optimization problems successfully yielding the best known approximation algorithms for GRAPH COLORING [7,26], BETWEENNESS [10], MAXIMUM SATISFIABILITY [13,19], and MAXIMUM STABLE SET [2], to name a few [29].

As noted in [16], the use of interior point methods for solving the semidefinite programming relaxation have proven to be very efficient. This is because methods such as the one proposed by Benson, Ye, and Zhang in [6] exploit the combinatorial structure of the relaxed problem. Other algorithms based on the nonlinear semidefinite relaxation include the work of Helmberg and Rendl [22] and Homer and Peinado [24].

The work of Burer et al. in [8] describes the implementation of a rank-2 relaxation heuristic dubbed `circut`. This software package was shown to compute better solutions than the randomized heuristic of Goemans and Williamson, in general [16]. In a recent paper dating from 2002, Festa, Pardalos, Resende, and Ribeiro [16] implement and test six randomized heuristics for MAX-CUT. These include variants of Greedy Randomized Adaptive Search Procedures (GRASP), Variable Neighborhood Search, and path-relinking algorithms [35]. Their efforts resulted in improving the best known solutions for several graphs and quickly producing solutions that compare favorably with the method of Goemans and Williamson [19] and `circut` [8]. For several sparse instances, the randomized heuristics presented in [16] outperformed `circut`.

In [25], Butenko et al. derive a "worst-out" heuristic having an approximation ratio of at least 1/3 which they refer to as the *edge contraction method*. The also present a computational analysis of several greedy construction heuristics for MAX-CUT based on variations of the 0.5-approximation algorithm of Sahni and Gonzalez [36]. With this, we now move on and describe the implementation of a new heuristic for MAX-CUT based on the new metaheuristic Continuous GRASP [23].

C-GRASP Heuristic

The Continuous Greedy Randomized Adaptive Search Procedure (C-GRASP) is a new metaheuristic for continuous global optimization [23]. The method is an extension of the widely known discrete optimization algorithm Greedy Randomized Adaptive Search Procedure (GRASP) [15]. Preliminary results are quite promising, indicating that C-GRASP is able to quickly converge to the global optimum on standard benchmark test func-

```
procedure GRASP(MaxIter, RandomSeed)
1   f* ← 0
2   X* ← ∅
3   for i = 1 to MaxIter do
4      X ← ConstructionSolution(G, g, X, α)
5      X ← LocalSearch(X, N(X))
6      if f(X) ≥ f(X*) then
7         X* ← X
8         f* ← f(X)
9      end
10  end
11  return X*
end procedure GRASP
```

Maximum Cut Problem, MAX-CUT, Figure 1
GRASP for maximization

tions. The traditional GRASP is a two-phase procedure which generates solutions through the controlled use of random sampling, greedy selection, and local search. For a given problem Π, let F be the set of feasible solutions for Π. Each solution $X \in F$ is composed of k discrete components $a_1, \ldots, a_k$. GRASP constructs a sequence $\{X\}_i$ of solutions for Π, such that each $X_i \in F$. The algorithm returns the best solution found after all iterations. The GRASP procedure can be described as in the pseudo-code provided in Fig. 1. The *construction phase* receives as parameters an instance of the problem G, a ranking function $g\colon A(X) \mapsto \mathbb{R}$ (where $A(X)$ is the domain of feasible components $a_1, \ldots, a_k$ for a partial solution X), and a parameter $0 < \alpha < 1$. The construction phase begins with an empty partial solution X. Assuming that $|A(X)| = k$, the algorithm creates a list of the best ranked αk components in $A(X)$, and returns a uniformly chosen element x from this list. The current partial solution is augmented to include x, and the procedure is repeated until the solution is feasible, i. e., until $X \in F$.

The *intensification phase* consists of the implementation of a hill-climbing procedure. Given a solution $X \in F$, let $N(X)$ be the set of solutions that can found from X by changing one of the components $a \in X$. Then, $N(X)$ is called the neighborhood of X. The improvement algorithm consists of finding, at each step, the element X^* such that

$$X^* := \arg \max_{X' \in N(X)} f(X') ,$$

where $f\colon F \mapsto \mathbb{R}$ is the objective function of the problem. At the end of each step we make the assignment $X^* \leftarrow X$ if $f(X) > f(X^*)$. The algorithm will eventually achieve a local optimum, in which case the solution X^* is such that $f(X^*) \geq f(X')$ for all $X' \in N(X^*)$. X^* is returned as the best solution from the iteration and the best solution from all iterations is returned as the overall GRASP solution. GRASP has been applied to many discrete problems with excellent results. For an annotated bibliography of GRASP applications, the reader is referred to the work of Festa and Resende in [17].

Like GRASP, the C-GRASP framework is a multi-start procedure consisting of a construction phase and a local search [14]. Specifically, C-GRASP is designed to solve continuous problems subject to box constraints. The feasible domain is given as the n-dimensional rectangle $S := \{x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n : l \leq x \leq u\}$, where $l, u \in \mathbb{R}^n$ are such that $l_i \leq u_i$, for $i = 1, 2, \ldots, n$. Pseudo-code for the basic C-GRASP is provided in Fig. 2. Notice that the algorithm takes as input the dimension n, upper and lower bounds l and u, the objective function f, and parameters `MaxIters`, `MaxNumIterNoImprov`, `NumTimesToRun`, `MaxDirToTry`, and a number $\alpha \in (0, 1)$.

To begin with, the optimal objective function value f^* is initialized to $-\infty$. The procedure then enters the main body of the algorithm in the **for** loop from lines 2–21. The value `NumTimesToRun` is the total number of C-GRASP iterations that will be performed. To begin with, more initialization takes place as the current solution x is initialized as a random point inside the hyperrectangle, which is generated according to a function `UnifRand`$([l, u))$ which is uniform onto $[l,u)^2$. Furthermore, the parameter which controls the discretization of the search space, h, is set to 1. Next, the construction phase and local search phases are entered. In line 9, the new solution is compared to the current best solution. If the objective function value corresponding to the current solution dominates the incumbent, then the current solution replaces the incumbent and `NumIterNoImprov` is set to 0. This parameter

[2]This is the "typical" definition of a *Uniform* distribution. That is, $P : X \mapsto \mathbb{R}$ is uniform onto $[A,B)$, if, for any subinterval $I \subset [A, B)$, the measure of $P^{-1}(I)$ equals the length of I.

```
procedure C-GRASP(n, l, u, f(·), MaxIters, MaxNumIterNoImprov, NumTimesToRun,
          MaxDirToTry, α)
1   f* ← −∞
2   for j = 1 to NumTimesToRun do
3     x ← UnifRand([l, u))
4     h ← 1
5     NumIterNoImprov ← 0
6     for Iter = 1 to MaxIters do
7       x ← ConstructGreedyRandomized(x, f(·), n, h, l, u, α)
8       x ← LocalSearch(x, f(·), n, h, l, u, MaxDirToTry)
9       if f(x) ≥ f* then
10        x* ← x
11        f* ← f(x)
12        NumIterNoImprov ← 0
13      else
14        NumIterNoImprov ← NumIterNoImprov + 1
15      end if
16      if NumIterNoImprov ≥ MaxNumIterNoImprov then
17        h ← h/2
18        NumIterNoImprov ← 0
19      end if
20    end for
21  end for
22  return x*
end procedure C-GRASP
```

Maximum Cut Problem, MAX-CUT, Figure 2
C-GRASP pseudo-code adapted from [23]

controls when the discretization measure h is reduced. That is, after a total of MaxNumIterNoImprov iterations occur in which no solution better than the current best solution is found, h is set to $h/2$ and the loop returns to line 6. By adjusting the value of h, the algorithm is able to locate general areas of the search space which contain high quality solutions, and then narrow down the search in those particular regions. The best solution after a total of NumTimesToRun iterations is returned as the best solution.

The construction phase of the C-GRASP takes as input the randomly generated solution $x \in S$ (see Fig. 2, line 3). Beginning with all coordinates unfixed, the method then performs a line search on each unfixed coordinate direction of x holding the other $n-1$ directions constant. The objective function values resulting from the line search solution for each coordinate direction are stored in a vector, say V. An element $v_i \in V$ is then selected uniformly at random from the maximum $(1-\alpha)100\%$ elements of V, and the v_i coordinate direction is fixed. This process repeats until all n coordinates of x have been fixed. The resulting solution is returned as the C-GRASP solution from the current iteration. For a slightly more detailed explanation of this procedure, the reader is referred to [23].

As for the local search phase, this procedure *simulates* the role of calculating the gradient of the objective function $f(\cdot)$. As mentioned earlier, gradients are not used in C-GRASP because oftentimes, they are difficult to compute and result in slow computation times. Therefore, the gradient is approximated as follows. Given the construction phase solution x, the local search generates a set of directions and determines in which direction (if any) the objective function improves.

The directions are calculated according to a bijective function T which maps the interval of integers $[1, 3^n) \cap \mathbb{Z}$ onto their balanced ternary representation.

Maximum Cut Problem, MAX-CUT, Table 1
Parameters used for C-GRASP

$\alpha = 0.4$	`MaxDirToTry` = 20
`NumTimesToRun` = 20	`MaxIters` = 1000
`MaxNumIterNoImrpov` = 1	

Recall that n is the dimension of the problem under consideration. That is, $T\colon [1, 3^n) \cap \mathbb{Z} \mapsto \{-1, 0, 1\}^{\mathbb{N}}$. Clearly, as $n \to \infty$, the number of search directions grows exponentially. Therefore, only `MaxDirToTry` directions are generated[3] and tested on the current solution. For each direction d, the point $\hat{x} := x + hd$ is constructed and $f(\hat{x})$ is computed. Recall that h is the parameter which controls the density of the search space discretization. If the constructed point $\hat{x} \in S$ has a more favorable objective value than the current point x, then $\hat{x}$ replaces x, and the process continues. The phase terminates when a locally optimal point $x^* \in S$ is found. The point x^* is said to be locally optimal if $f(x^*) \geq f(x^* + hd) \forall d \in \{1, 2, \ldots,$ `MaxDirToTry`$\}$. Again, for a slightly more in depth description of this procedure, the reader should see the paper by Hirsch et al. [23].

Computational Results The proposed procedure was implemented in the C++ programming language and complied using Microsoft® Visual C++ 6.0. It was tested on a PC equipped with a 1700 MHz Intel® Pentium® M processor and 1 GB of RAM operating under the Microsoft® Windows® XP environment. The C-GRASP parameters used are provided in Table 1. First, we tested the C-GRASP on 10 instances produced by the Balasundarm–Butenko problem generator in [3]. Though these problems are relatively small, they have proven themselves to be quite formidable against the Multilevel Coordinate Search (MCS) black-box optimization algorithm. We also tested the C-GRASP on 12 instances from the TSPLIB [34] collection of test problems for the TRAVELING SALESMAN PROBLEM. These problems are also used as benchmark problems for testing MAX-CUT heuristics [19].

For further comparison, all instances were tested using the rank-2 relaxation heuristic `circut` [8], as well as with a simple 2-exchange local search heuristic which

[3]uniformly at random

```
procedure LocalSearch(G, MaxIter)
1   f* ← −∞
2   x* ← ∅
3   for j = 1 to MaxIter do
4     x ← KruskalMST(x, G)
5     x ← LocalImprove(x, G)
6     if f(x) ≥ f* then
7       x* ← x
8       f* ← f(x)
9     end if
10  end for
11  return x*
end procedure LocalSearch
```

Maximum Cut Problem, MAX-CUT, Figure 3
The 2-exchange local search routine

is outlined in the pseudo-code provided in Fig. 3. The method receives as input a parameter `MaxIter` indicating the maximum number of iterations to be performed and $G = (V, E)$ the instance of the problem whereupon a maximum spanning tree is found using Kruskal's algorithm [1]. The spanning tree, due to its natural bipartite structure provides a feasible solution to which a swap-based local improvement method is applied in line 5. The local improvement works as follows. For all pairs of vertices (u,v) such that $u \in S$ and $v \in \bar{S}$, a swap is performed. That is, we place $u \in \bar{S}$ and $v \in S$. If the objection function is improved, the swap is kept; otherwise, we undo the swap and examine the next (u,v) pair. The local search was tested on the same PC as the C-GRASP. The `circut` heuristic was compiled using Compaq® Visual Fortran on a PC equipped with a 3.60 GHz Intel® Xeon® processor and 3.0 GB of RAM operating under the Windows® XP environement.

Table 2 provides computational results of the algorithms on the 10 Balasundarum–Butenko instances from [3]. The first three columns provide the instance name, the number of vertices and the optimal solution. The solutions from the heuristics are provided next. The solutions from the Multilevel Coordinate Search algorithm were provided in [3]. For all of these instances, the time required by the C-GRASP, `circut`, and the local search to find their best solutions was fractions of a second. Computing times were not listed for the MCS algorithm in [3]. Notice that the 2-exchange

Maximum Cut Problem, MAX-CUT, Table 2
Comparative results from the Balasundaram–Butenko instances from [3]

Name	$\|V\|$	Opt	C-GRASP	MCS	circut	LS
G5-1	5	126	126	125	125	126
G5-2	5	40	40	39	40	40
G8-1	8	1987	1802	1802	1987	1987
G8-2	8	1688	1631	1671	1688	1688
G10-1	10	1585	1585	1513	1585	1585
G10-2	10	1377	1373	1373	1377	1377
G15-1	15	399	389	389	399	399
G15-2	15	594	594	593	594	594
G20-1	20	273	267	273	273	273
G20-2	20	285	285	282	285	285

local search computed optimal solutions for each of these instances, followed closely by `circut` which found optimal cuts for all but one problem. As for the continuous heuristics, the C-GRASP found optimal solutions for 5 of the 10 instances while the MCS procedure produced optimal cuts for only 1 instance. For the 5 instances where C-GRASP produced suboptimal solutions, the average deviation from the optimum was 3.54%.

Table 3 shows results of the C-GRASP, local search, and `circut` heuristics when applied to 12 instances from the TSPLIB collection of test problems for the TRAVELING SALESMAN problem [34]. The first two columns provide the instance name and the size of the vertex set $|V|$. Next the solutions are provided along with the associated computing time required by the respective heuristic. Notice that for all 12 instances, the three heuristics all found the same solutions. Notice that in terms of computation time, the simplest heuristic, the 2-exchange local search seems to be the best performing of the three methods tested. The rank-2 relaxation algorithm `circut` is also very fast requiring only 2.99 s on average to compute the solution. On the other hand, the C-GRASP method did not scale as well as the others. We see that there is a drastic increase in the solution time as the number of vertices increases beyond 48.

This is not particularly surprising. The philosophical reasoning behind the slow computation time of the C-GRASP relative to the discrete heuristics being that the C-GRASP is a *black-box* method and does not take into account any information about the problem other than the objective function. To the contrary, the local search and `circut` specifically exploit the combinatorial structure of the underlying problem. This allows them to quickly calculate high quality solutions.

Conclusions

In this paper, we implemented a new metaheuristic for the MAXIMUM CUT problem. In particular, we proposed the use of a continuous greedy randomized adaptive search procedure (C-GRASP) [23], for a continuous formulation of the problem. To our knowledge,

Maximum Cut Problem, MAX-CUT, Table 3
Comparative results from TRAVELING SALESMAN PROBLEM instances [34]

Name	$\|V\|$	C-GRASP	Time (s)	LS	Time (s)	circut	Time (s)
burma14	14	283	0.120	283	0.00	283	.046
gr17	17	24986	0.19	24986	0.00	24986	.047
bays29	29	53990	0.701	53990	0.01	53990	1.109
dantzig42	42	42638	1.832	42638	0.01	42638	1.75
gr48	48	320277	4.216	320277	0.00	320277	3.672
hk48	48	771712	2.804	771712	0.00	771712	2.516
gr96	96	105328	52.425	105328	0.01	105328	14.250
kroA100	100	5897368	66.445	5897368	0.01	5897368	2.359
kroB100	100	5763020	94.175	5763020	0.01	5763020	2.531
kroC100	100	5890745	66.545	5890745	0.01	5890745	2.500
kroD100	100	5463250	94.155	5463250	0.03	5463250	2.547
kroE100	100	5986587	69.64	5986587	0.03	5986587	2.500

this is the first application of C-GRASP to continuous formulations of discrete optimization problems. Numerical results indicate that the procedure is able to compute optimal solutions for problems of relatively small size. However, the method becomes inefficient on problems approaching 100 nodes. The main reason for this is the fact that C-GRASP is a black-box method, in that it does not take advantage of any information about the problem structure. Recall that the only input to the method is some mechanism to compute the objective function. A natural extension of the work presented here is to *enhance* the C-GRASP framework to take advantage of the structure of the problem at hand. Using a priori information about the problem being considered, one could modify the algorithm to include these properties which would presumably reduce the required computation time.

See also

- Combinatorial Test Problems and Problem Generators
- Continuous Global Optimization: Models, Algorithms and Software
- Derivative-Free Methods for Non-smooth Optimization
- Greedy Randomized Adaptive Search Procedures
- Heuristic Search
- Integer Programming
- NP-complete Problems and Proof Methodology
- Quadratic Integer Programming: Complexity and Equivalent Forms
- Random Search Methods
- Semidefinite Programming: Optimality Conditions and Stability
- Semidefinite Programming and the Sensor Network Localization Problem, SNLP
- Solving Large Scale and Sparse Semidefinite Programs
- Variable Neighborhood Search Methods

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network Flows: Theory, Algorithms, and Applications. Prentice-Hall
2. Alon N, Kahale N (1995) Approximating the independence number via the θ-function. Technical report. Tel Aviv University, Tel Aviv, Israel
3. Balasundaram B, Butenko S (2005) Constructing test functions for global optimization using continuous formulations of graph problems. J Optim Method Softw 20(4–5): 439–452
4. Barahona F (1983) The max-cut problem in graphs is not contractible to k_5. Oper Res Lett 2:107–111
5. Barahona F, Grötschel M, Jünger M, Reinelt G (1998) An application of combinatorial optimization to statisical physics and circuit layout design. Oper Res 36:493–513
6. Benso S, Ye Y, Zhang X (2000) Solving large-scale sparse semidefinite programs for combinatorial optimization. SIAM J Optim 10:443–461
7. Blum A (1994) New approximation algorithms for graph coloring. J ACM 41(3):470–516
8. Burer S, Monteiro RDC, Zhang Y (2001) Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs. SIAM J Optim 12:503–521
9. Chang KC, Du D-Z (1987) Efficient algorithms for layer assignment problems. IEEE Trans Computer-Aided Des 6: 67–78
10. Chor B, Sudan M (1995) A geometric approach to betweenness. In: Proc. of the 3rd Annual European Symposium Algorithms. pp 227–237
11. de la Vega WF (1996) MAX-CUT has a randomized approximation scheme in dense graphs. Random Struct Algoritm 8(3):187–198
12. Deza M, Laurent M (1997) Geometry of cuts and metrics. Springer-Verlag
13. Feige U, Goemans MX (1995) Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In: Proc. of the 3rd Israel Symposium on Theory of Computing and Systems. pp 182–189
14. Feo TA, Resende MGC (1989) A probabilistic heuristic for a computationaly difficult set covering problem. Oper Res Lett 8:67–71
15. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Glob Optim 6:109–133
16. Festa P, Pardalos PM, Resende MGC, Ribeiro CC (2002) Randomized heuristics for the MAX-CUT problem. Optim Method Softw 7:1033–1058
17. Festa P, Resende MGC (2002) GRASP: an annotated bibliography. In: Ribeiro C, Hansen P (eds) Essays and surveys in metaheuristics. Kluwer, pp 325–367
18. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman and Company
19. Goemans M, Williamson DP (1995) Improved approximation algorithms for MAX-CUT and satisfiability problems using semidefinite programming. J ACM 42:1115–1145
20. Grotschel M, Pulleyblank WR (1981) Weakly bipartite graphs and the max-cut problem. Oper Res Lett 1:23–27
21. Hager WW, Krylyuk Y (1999) Graph partitioning and continuous quadratic programming. SIAM J Discret Math 12:500–523

22. Helmberg C, Rendl F (2000) A spectral bundle method for semidefinite programming. SIAM J Optim 10:673–696
23. Hirsch MJ, Pardalos PM, Resende MGC (2006) Global optimization by continuous GRASP. Optim Lett (in press)
24. Homer S, Peinado M (1997) Two distributed memory parallel approximation algorithms for Max-Cut. J Parallel Distrib Comput 1:48–61
25. Kahruman-Anderoglu S, Kolotoglu E, Butenko S, Hicks IV (2007) On greedy construction heuristics for the MAX-CUT problem. Int J Comput Sci Eng (in press)
26. Karger D, Motwani R, Sudan M (1994) Approximate graph coloring by semidefinite programming. In: 35th Annual Symposium on Foundations of Computer Science. pp 2–13
27. Karp RM (1972) Reducability among combinatorial problems. In: Miller R, Thatcher J (eds) Complexity of computer computations. Plenum Press, pp 85–103
28. Lovász L (1979) On the shannon capacity of a graph. IEEE Trans Inf Theory IT-25(1):1–7
29. Lu H-I (1996) Efficient approximation algorithms for some semidefinite programs. Ph.d. dissertation, Brown University
30. Papadimitriou CH (1994) Computational complexity. Addison Wesley Longman
31. Papadimitriou CH, Yannakakis M (1991) Optimization, approximation, and complexity classes. J Comput Syst Sci 43(3):425–440
32. Pinter RY (1984) Optimal layer assignment for interconnect. J VLSI Comput Syst 1:123–137
33. Poljak S, Tuza Z (1995) The max-cut problem: a survey. In: Cook W, Lovász L, Seymour P (eds) Special year in combinatorial optimization. DIMACS Series in discrete mathematics and computer science. American Mathematical Society
34. Reinelt G (1991) TSPLIB - a traveling salesman problem library. ORSA J Comput 3:376–384
35. Resende MGC, Ribeiro CC (2003) Greedy randomized adaptive search procedures. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer, pp 219–249
36. Sahni S, Gonzalez T (1976) *P*-complete approximation problems. J ACM 23(3):555–565

Maximum Entropy and Game Theory

PANOS PARPAS, BERÇ RUSTEM
Department of Computing,
Imperial College, London, UK

Article Outline

Abstract
Background
Methods
Game Theory Approach
Maximum Entropy Formulation
Relationships Between Game Theory and Maximum Entropy
References

Abstract

In decision making under uncertainty an important step is uncertainty quantification. Game theory has been traditionally used since it injects robustness into the decision process. Another popular framework is that of maximum entropy. The purpose of this article is to briefly explain the two solution concepts and point out situations in which they are identical.

Background

Consider the following optimization problem:

$$\inf_{x\in X} F(x) \triangleq \int_{\Omega} f(x,\omega)\, \mathrm{d}P(\omega)\,, \tag{1}$$

where x is the decision vector, ω is a vector representing the random parameters that are distributed according to the probability measure P. We are not concerned here with the exact properties of $f(\cdot,\cdot)$ for the problem to be valid, the interested reader is referred to [2] where the properties of this type of problem are made more precise. Instead we give two well known and studied examples of this formulation. The first is the so called two-stage recourse problem and the second is the chance constrained formulation. The former can be formulated as in Eq. (1) with the following definition for the objective function:

$$f(x_1,\omega) \triangleq f_d(x_1) + f_u(x_1,\omega)\,, \tag{2}$$

and where

$$f_u(x_1,\omega) \triangleq \inf_{x_2(\omega)} \{f_2(x_1,x_2(\omega),\omega) \mid x_2(\omega)\in X_2(\omega,x_1)\}\,. \tag{3}$$

In Eq. (2) the objective function is split into two parts, the deterministic (f_d) and the uncertain part (f_u) of the problem . The decision to be taken is x_1. The full consequences of following a particular strategy are not known exactly since the true cost will depend on the

solution of Eq. (3). The objective is therefore to find the decision x_1 that is best on average.

A different decision model is given by chance constrained programming problems. These can be formulated as follows:

$$\inf_x \{f(x) \mid \Pr(g(x,\omega) \geq 0) \geq \alpha\} \ ,$$

where we optimize an objective function and impose constraints that need to be satisfied with a probability of above a certain threshold α.

These two models have been widely studied and have found many applications where traditional optimization is used. It is also evident that their usefulness revolves around our ability to provide a reasonable description of the uncertainties.

In order to provide a description of the uncertainties a technique based on moment matching can be used. Under this framework we assume that the decision maker can not provide an exact description of the distribution but only knows some of its moments. The problem is to recover a meaningful probability measure given this knowledge: suppose a vector of functions $m(\omega) = [m_1(\omega) \dots m_n(\omega)]$ and a vector of scalars $\mu = [\mu_1 \dots \mu_n]$ are given, the problem is to find a P such that:

$$\begin{aligned} \int_\Omega m_i(\omega)\, \mathrm{d}P(\omega) &= \mu_i, \quad i = 1, \dots, n \\ \int_\Omega \mathrm{d}P(\omega) &= 1 \ , \\ P(\omega) &\geq 0 \ , \quad \text{a.e} \ , \end{aligned} \tag{4}$$

where Ω is a compact subset of $\mathbb{R}^m$. By $\mathcal{P}$ we will denote the set of of all finite signed measures that are defined on the σ-field $\mathcal{F}$ of Ω. The vector $m(\omega)$ represents the (generalized) moments of the distribution. The problem in Eq. (4) is the so called generalized Hausdorff moment problem. The aim is to recover a compactly supported distribution from a finite number of its general, not necessarily power, moments. This is a variation of the classical moment problem formulated by Stieltjes. In [4] one can find a comprehensive summary of the main results when $\Omega = [0, 1]$. Prekopa [13] provides an excellent summary of results that are especially relevant in stochastic optimization problems.

Methods

Solving optimization problems where the uncertainty is only known through its moments requires some kind of regularization in order to fix the probability measure with which the optimization is to be done. Two popular frameworks are game theoretic and maximum entropy approaches. Under the game theory framework one selects the distribution with the worst case realization of the uncertainties. When a maximum entropy solution is sought, one optimizes with respect to the distribution with the maximum uncertainty.

Game Theory Approach

When P is unknown or not known exactly then the decision maker assumes that if strategy x is followed then the consequences of following this strategy will be decided by some law of Nature. Motivated by the application oriented requirement for robust decision making, we assume that Nature is antagonistic. If we decide to follow strategy x then Nature will follow strategy P^*. The latter is the solution of the following optimization problem:

$$\Phi(x) = \sup_{P \in \mathcal{P}} \int_\Omega f(x,\omega)\, \mathrm{d}P(\omega) \ , \tag{5}$$

where $\Phi(x)$ represents the value (outcome) of the game if strategy x is followed. Obviously $\Phi(x) \geq F(x)$ for given $x \in X$ and for all $P \in \mathcal{P}$. Therefore, after we minimize Eq. (5) for x we will be guaranteed to attain a value which is as good as $\Phi(x)$ whatever strategy nature decides to follow. The robustness property of the minimax strategy originates from the latter property. We thus reformulate Eq. (1) as follows:

$$\inf_{x \in X} \sup_{P \in \mathcal{P}} \int_\Omega f(x,\omega)\, \mathrm{d}P(\omega) \ . \tag{6}$$

This approach has its origins in game theory and has been used extensively in many areas of optimization. See for example [9] for an excellent introduction to game theory, applications of minimax especially in economics and finance can be found in [14]. Numerical algorithms to solve Eq. (6) have been proposed in [5,6]. The general idea of these algorithms is to solve the inner maximization problem using results for general Chebyshev inequalities [4,17]. However these methods require several global optimization steps to be performed

at each iteration in order to identify the support of the measure that maximizes Eq. (5). Moreover it is usually assumed that the max-function is convex. Algorithms based on stochastic quasi-gradient methods were proposed in [4,17]. Recently Shapiro et al. [15] proposed the use of a reference distribution in order to reformulate the minimax problem into a standard stochastic programming problem. Bertsimas et al. [1] and Lassere [11] proposed a semidefinite formulation of the inner maximization problem in Eq. (6).

Maximum Entropy Formulation

The formulation of the moment problem using the maximum entropy principle was initiated by Jaynes [8]. The derivations in this Section are more or less standard (see e. g. [12]).We will assume that a multivariate continuous density is postulated, discrete distributions share similar properties. Under these assumptions the maxent formulation is given by:

$$\begin{aligned} &\inf_{p\in\mathcal{P}_c} \; Q(p,h) = \int_\Omega p(\omega)\ln\frac{p(\omega)}{h(\omega)}\,\mathrm{d}\omega \\ &\text{s.t}\int_\Omega m_i(\omega)p(\omega)\,\mathrm{d}\omega = \mu_i \quad i = 0\ldots n\,, \end{aligned} \tag{7}$$

where $\mathcal{P}_c$ denotes the restriction on $\mathcal{P}$ to all absolutely continuous measures w.r.t $\mathrm{d}\omega$, we will write $\nu \ll \mu$ to mean that ν is absolutely continuous w.r.t μ. The function in the objective function is the so called Kullback Leibler divergence (see e. g. [3]) and serves as a kind of distance metric between $h(\omega)$ and $p(\omega)$; the former is a distribution that is assumed to be known. The objective is to find a p.d.f with the prescribed moments that is as close to h as possible. If such a function is not known then we take $h(\omega) \equiv 1$ (i. e. the uniform distribution) and the problem becomes the classical maximum entropy formulation. The convex functional defined by $Q(p,h)$ is always strictly positive and is zero if and only if $p = h$ a.e. It is also worth mentioning that in general $Q(p,h) \neq Q(h,p)$. These properties of $Q(p,h)$ are well known. We refer the interested reader to [3] for more properties of the entropy function. We assume that $m_0(\omega) = \mu_0 = 1$. Note that by considering general moments as opposed to power moments allows us to impose fractile constraints, this property is important in many applications.

While the problem in Eq. (7) is a convex optimization problem it cannot be handled using standard numerical algorithms. For this reason one considers the dual of Eq. (7). The Lagrangian associated with Eq. (7) is given by:

$$\begin{aligned} L(p,\lambda) = &\int_\Omega p(\omega)\ln\frac{p(\omega)}{h(\omega)}\,\mathrm{d}\omega \\ &+\sum_{i=0}^{n}\lambda_i\left(\int_\Omega m_i(\omega)p(\omega)\,\mathrm{d}\omega - \mu_i\right). \end{aligned}$$

The dual problem of Eq. (7) is given by:

$$\sup_\lambda D(\lambda) = \inf_{p\in\mathcal{P}_c} L(p,\lambda)\,. \tag{8}$$

It is well known that the inner minimization on Eq. (8) can be done explicitly using functional derivatives [12,16]:

$$\begin{aligned} &L(p+\delta p,\lambda) \\ &= \int_\Omega (p(\omega)+\delta p(\omega))\ln\left\{\frac{p(\omega)}{h(\omega)}\left(1+\frac{\delta p(\omega)}{p(\omega)}\right)\right\}\,\mathrm{d}\omega \\ &\quad+\sum_{i=0}^{n}\lambda_i\left(\int_\Omega m_i(\omega)(p(\omega)+\delta p(\omega))\,\mathrm{d}\omega - \mu_i\right) \\ &= L(p,\lambda) + \int_\Omega\left\{1+\ln\left(\frac{p(\omega)}{h(\omega)}\right)\right\}\delta p(\omega)\,\mathrm{d}\omega \\ &\quad+\sum_{i=0}^{n}\int_\Omega \lambda_i m_i(\omega)\delta p(\omega)\,\mathrm{d}\omega\,, \end{aligned}$$

where to get the last equality we assumed that δp is small, used the approximation $\ln(1+\epsilon) \approx \epsilon$ (which is valid for small ϵ) and ignored second order terms. The stationary points of the Lagrangian must therefore satisfy:

$$p(\omega) = h(\omega)\exp\left(-1-\sum_{i=0}^{n}\lambda_i m_i(\omega)\right). \tag{9}$$

Using the normalization condition we have:

$$\mathcal{Z} = \exp(1+\lambda_0) = \int_\Omega h(\omega)\exp\left(-\sum_{i=1}^{n}\lambda_i m_i(\omega)\right)\mathrm{d}\omega.$$

Using the equation above we can write Eq. (9) as follows:

$$p(\omega) = \frac{h(\omega)}{\mathcal{Z}}\exp\left(-\sum_{i=1}^{n}\lambda_i m_i(\omega)\right). \tag{10}$$

Finally, using Eq. (10) and the normalization constraint in Eq. (8) we find the following explicit form for the dual problem:

$$\sup_{\lambda} D(\lambda) = -\ln \mathcal{Z} - \sum_{i=1}^{n} \lambda_i \mu_i \,. \tag{11}$$

The dual formulation given above is more useful than the primal problem since the dual problem is amenable to conventional optimization algorithms.

Relationships Between Game Theory and Maximum Entropy

The minimax approach has proven to be a prudent method for problems where the nature of the uncertainty is not known exactly. We will approach the problem somewhat differently by dispensing the usual assumptions of convexity but allowing the decision maker to adopt mixed strategies. Such an approach (in the context of Stochastic Programming) has been described in Kolbin [10] but has not received much attention. The advantage of allowing mixed strategies is that the problem exhibits a saddle point. Topsøe [18] showed that if the decision problem has a specific structure (will be outlined below) then the solution of the maximum entropy problem and that of zero-sum games are dual to each other. Recently Grünwald et al. [7] has further developed this approach so that it can be applied to more general games. This generalization however has been done at the expense of defining more general entropy functionals; these do not, in general, render themselves to numerical algorithms. We believe this relationship to be very interesting and can under certain conditions be used as an additional motivation for adopting the maximum entropy principle.

Let Ω be a compact subset of $\mathbb{R}^n$ and let $\mathcal{F}$ be the σ-field generated by Ω. We will use ω to denote a random vector whose distribution is known to belong to a family $\mathcal{P}$. The following meaningless formulation of a stochastic programming problem:

$$\begin{aligned} &\inf_{x\in X} f(x,\omega) \\ &\text{s.t } g_i(x,\omega) \le 0\,, \quad i = 1,\ldots,k \end{aligned}$$

can be placed into a pertinent form by formulating it as a two-person zero sum game $G = (x, \omega, q)$. The first player is the Decision Maker (DM) that selects vectors $x \in X \subset \mathbb{R}^m$. The second player is Nature that selects an event $\omega \in \mathcal{F}$ with probability $P(\omega)$, it is further assumed that the exact probability measure of Nature is only known to belong to a certain family $\mathcal{P}$. The function q represents the outcome of the game given the strategies the two players decide to follow. Kolbin [10] suggested the following form:

$$q(x,\omega) = f(x,\omega) + \sum_{i=1}^{k} \beta_i(g_i(x,\omega))\,, \tag{12}$$

where $\beta_i(a)$ is a continuous non-decreasing penalty function that is 0 when $a \le 0$. An example of such a function is the max-penalty function given by: $c_i \max\{g_i(x,\omega), 0\}$ (c_i is a penalty parameter).

The DM would like to minimize the outcome of the game given by Eq. (12) whereas Nature being antagonistic would like to maximize this quantity:

$$\inf_{x\in X} \sup_{P\in\mathcal{P}} H(x,P) = \int_{\Omega} q(x,\omega)\, \mathrm{d}P(\omega)\,. \tag{13}$$

For the game above to exhibit a saddle point convexity assumptions need to be imposed on q. Many problems of interest do not have this property and it is necessary to resort to mixed strategies for the DM. Using our assumptions that q is continuous, and the compactness of Ω and X, it can be shown [10] that if we allow the DM to follow mixed strategies then the game in Eq. (13) will have a saddle point, i.e:

$$\begin{aligned} &\inf_{K\in\mathcal{K}} \sup_{P\in\mathcal{P}} \int_{\Omega\times X} q(x,\omega)\, \mathrm{d}P(\omega)\, \mathrm{d}K(x) \\ &\quad = \sup_{P\in\mathcal{P}} \inf_{K\in\mathcal{K}} \int_{\Omega\times X} q(x,\omega)\, \mathrm{d}P(\omega)\, \mathrm{d}K(x) \\ &\quad = H(K^*, P^*)\,, \end{aligned} \tag{14}$$

where the set $\mathcal{K}$ represents the family of randomized strategies of the DM.

Assume that the DM selects a probability measure $K \in \mathcal{K}$ and Nature selects $P \in \mathcal{P}$ and both have their support in Ω (or X). Moreover assume that the objective function of the game has the following functional form:

$$H(K,P) = \int_{\Omega} -p(\omega) \ln k(\omega)\, \mathrm{d}\omega\,, \tag{15}$$

where $p(\omega)$ and $k(\omega)$ are the Radon–Nikodym derivatives w.r.t $\mathrm{d}\omega$ of $P \in \mathcal{P}$ and $K \in \mathcal{K}$ respectively.

Topsøe [18] observed that under these conditions the maximum entropy solution and the minimax solution coincide. To see why this is the case, suppose that nature selects a probability measure from the following family:

$$\mathcal{P}_c = \left\{ \frac{\mathrm{d}P}{\mathrm{d}\omega} \mid P \in \mathcal{P}, \int_\Omega m_i(\omega)\, \mathrm{d}P(\omega) = \mu_i, \right. \\ \left. i = 1, \ldots, n \;\; P \ll \mathrm{d}\omega \right\} ,$$

where $P \ll \mathrm{d}\omega$ is used to denote that P is absolutely continuous w.r.t to $\mathrm{d}\omega$. $\mathcal{K}_c$, the family of admissible strategies for the DM is defined in an analogous manner. If Nature adopts a maximum entropy distribution, then its strategy can be found by solving:

$$\sup_{p \in \mathcal{P}_c} M(p) = \int_\Omega -p(\omega) \ln p(\omega)\, \mathrm{d}\omega .$$

The optimal solution will be given by:

$$p^*(\omega) = \exp\{-1 - \lambda_0^* - \sum_{i=1}^n \lambda_i^* m_i(\omega)\} .$$

The optimal strategy of the DM can then be obtained by solving:

$$\inf_{k \in \mathcal{K}_c} \int_\Omega -p^*(\omega) \ln k(\omega)\, \mathrm{d}\omega .$$

Using the information inequality [3] we have:

$$\int_\Omega -p^*(\omega) \ln k(\omega)\, \mathrm{d}\omega \;\geq\; \int_\Omega -p^*(\omega) \ln p^*(\omega)\, \mathrm{d}\omega , \tag{16}$$

the above inequality is satisfied as an inequality if and only if $k(\omega) = p^*(\omega)$ a.e. Consequently the optimal strategy for the DM is the same as Nature's strategy.

Conversely, assuming that the game has the functional form given in Eq. (15), then the minimax solution of the game in Eq. (14) is the same as the maximum entropy solution. Indeed, by using the information inequality we have:

$$\begin{aligned} &\sup_{p \in \mathcal{P}_c} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega \\ &\geq \int_\Omega -p^*(\omega) \ln k(\omega)\, \mathrm{d}\omega \\ &\geq \int_\Omega -p^*(\omega) \ln p^*(\omega)\, \mathrm{d}\omega = M(p^*) , \end{aligned}$$

where p^* is the distribution of maximum entropy. From the above relationship it follows that:

$$M(p^*) \leq \inf_{k \in \mathcal{K}_c} \sup_{p \in \mathcal{P}_c} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega , \tag{17}$$

if we choose $k = p^*$ as the minimizer of the left hand side of Eq. (14), then:

$$\begin{aligned} \sup_{p \in \mathcal{P}_c} \int_\Omega -p(\omega) \ln p^*(\omega)\, \mathrm{d}\omega &= -\lambda_0^* - \sum_{i=1}^n \lambda_i^* \mu_i \\ &= M(p^*) , \end{aligned}$$

it follows from above that:

$$\inf_{k \in \mathcal{K}} \sup_{p \in \mathcal{P}_c} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega \leq M(p^*) , \tag{18}$$

and therefore $k = p^*$ is indeed the minimizer of the left hand side of Eq. (14). From the well known property of minimax problems:

$$\begin{aligned} M(p^*) &= \inf_{k \in \mathcal{K}} \sup_{p \in \mathcal{P}_c} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega \\ &\geq \sup_{p \in \mathcal{P}_c} \inf_{k \in \mathcal{K}} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega , \end{aligned}$$

and from:

$$\begin{aligned} &\sup_{p \in \mathcal{P}_c} \inf_{k \in \mathcal{K}} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega \\ &\qquad \geq \inf_{k \in \mathcal{K}} \int_\Omega -p^*(\omega) \ln k(\omega)\, \mathrm{d}\omega = M(p^*) , \end{aligned}$$

we conclude that the game has a saddle point at $p = k = p^*$.

For games in the form of Eq. (15), the relationship between game theory and maximum entropy is most useful both theoretically and practically. For games not in the form described above can still be approached via maximum entropy methods but the definition of the entropy functional is given by a more general functional form. Grünwald et al. [7] defined the generalized entropy function as:

$$M(P) = \inf_{k \in \mathcal{K}} \int_{\Omega \times X} q(x, \omega)\, \mathrm{d}P(\omega)\, \mathrm{d}K(x) .$$

The maximum entropy problem becomes:

$$\max_{p \in \mathcal{P}_c} M(P) . \tag{19}$$

They showed that using the generalized definition of entropy one could find the same results for both game theory and maximum entropy problems. Even though the formulation in Eq. (19) is very general, unfortunately there is no general way to solve it. However, the relationship between the two principles is worth further investigation.

References

1. Bertsimas D, Sethuraman J (2000) Moment problems and semidefinite optimization. In: Handbook of semidefinite programming. Internat Ser Oper Res Manag Sci, vol 27. Kluwer, Boston, pp 469–509
2. Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer Series in Operations Research. Springer, New York
3. Cover TM, Thomas JA (1991) Elements of information theory. Wiley Series in Telecommunications. A Wiley-Interscience Publication. Wiley, New York
4. Dette H, Studden WJ (1997) The theory of canonical moments with applications in statistics, probability, and analysis. Wiley Series in Probability and Statistics: Applied Probability and Statistics. A Wiley-Interscience Publication. Wiley, New York
5. Ermoliev Y, Gaivoronski A, Nedeva C (1985) Stochastic optimization problems with incomplete information on distribution functions. SIAM J Control Optim 23(5):697–716
6. Gaivoronski A (1991) A numerical method for solving stochastic programming problems with moment constraints on a distribution function. Stochastic programming, Part II (Ann Arbor, MI, 1989). Ann Oper Res 31(1–4):347–369
7. Grünwald PD, Dawid AP (2004) Game theory, maximum entropy, minimum discrepancy and robust Bayesian decision theory. Ann Statist 32(4):1367–1433
8. Jaynes ET (1957) Information theory and statistical mechanics I. Phys Rev 106:620–630
9. Karlin S (1992) Mathematical methods and theory in games, programming, and economics. vol I: Matrix games, programming, and mathematical economics, vol II: The theory of infinite games, Reprint of the 1959 original. Dover, New York
10. Kolbin VV (1970) Stochastic programming. In: Theory of Probability. Mathematical Statistics. Theoretical Cybernetics. 1968 (Russian). Akad Nauk SSSR Vsesojuz Inst Naučn i Tehn Informacii, Moscow, pp 5–68
11. Lasserre JB (2002) Bounds on measures satisfying moment conditions. Ann Appl Probab 12(3):1114–1137
12. Mead LR, Papanicolaou N (1984) Maximum entropy in the problem of moments. J Math Phys 25(8):2404–2417
13. Prékopa A (1995) Stochastic programming. Mathematics and its Applications, vol 324. Kluwer, Dordrecht
14. Rustem B, Howe M (2002) Algorithms for worst-case design and applications to risk management. Princeton University Press, Princeton, NJ
15. Shapiro A, Ahmed S (2004) On a class of minimax stochastic programs. SIAM J Optim (electronic) 14(4):1237–1249
16. Smith DR (1998) Variational methods in optimization. Reprint of the 1974 original. Dover, Mineola, NY
17. Smith JE (1995) Generalized Chebychev inequalities: theory and applications in decision analysis. Oper Res 43(5):807–825
18. Topsøe F (1979) Information-theoretical optimization techniques. Kybernetika 15:8–27

Maximum Entropy Principle: Image Reconstruction

Entropy Optimization for Image Reconstruction

SHU-CHERNG FANG[1], JACOB H.-S. TSAO[2]

[1] North Carolina State University, Raleigh, USA

[2] San Jose State University, San Jose, USA

MSC2000: 94A17, 94A08

Article Outline

Keywords

See also

References

Keywords

Entropy optimization; Image reconstruction; Maximum entropy principle; Principle of maximum entropy

Images can be used to characterize the underlying distribution of certain physical properties, such as density, shape, and brightness, of an object under investigation. In many applications where an image is required, only a finite number of observations and/or indirect measurements can be made. Image reconstruction is a procedure for processing the measurement data to construct an image of the object. This section introduces the basic concept of *image reconstruction from projection data*. Two types of entropy optimization mod-

els, namely, the finite-dimensional model and vector-space model, and three classes of entropy optimization methodologies, namely, the discretization methods, Banach-space methods (e. g., MENT) and Hilbert-space methods (e. g., *finite element method*) are included. For more details about image reconstruction, the reader is referred to [2,7,13] and the references therein.

A very important scientific application of image reconstruction is in *computerized tomography* (CT) for medical diagnosis. Physicians need to know, for example, the location, shape, and size of a suspected tumor inside a patient's brain in order to plan a suitable course of treatment. With computerized tomography, images of cross-sections of a human body can be constructed from data obtained by measuring the attenuation of X-rays along a large number of straight lines (or strips) through each cross-section. For ease of introduction, we illustrate the basic ideas about image reconstruction with the example of two-dimensional X-ray CT, with the understanding that the discussion can be generalized to higher-dimensional settings.

In this example, the distribution to be determined is that of the X-ray linear attenuation coefficient of human body tissues. The total attenuation of the X-ray beam between a source and a detector is approximately the integral of the linear attenuation coefficient along the line between the source and the detector. The unknown distribution of the X-ray linear attenuation coefficient is represented by a density function f of two variables, which assumes zero-value outside a squared-shape region. The squared region is usually referred to as the support of the image.

Two basic types of entropy optimization models, namely, *finite-dimensional model* and *vector-space model*, are commonly used to decide the density function f. The finite-dimensional models approximate the density values over the support of the image at a finite number of grid points, while the density is approximated by a real-value function for the entire scanning region in the vector-space models. The latter models were motivated to reconstruct the image with only a small number of available projections.

In the finite-dimensional models, the support of the density f is represented by n (given by the users) regularly spaced grid points, and the values of the density function f at these points are denoted by $\mathbf{f} \equiv (f_1, \ldots, f_n)$. Assume that m projections are made and the measurement data $\mathbf{d} \equiv (d_1, \ldots, d_m)$ are obtained.

The relationship between the unknown density values $\mathbf{f}$ and the observed measurement $\mathbf{d}$ can be approximated by a linear relation

$$\mathbf{d} \approx \mathbf{A}\mathbf{f}, \tag{1}$$

where $\mathbf{A} = [a_{ij}]$ is a projection matrix.

Note that the approximation sign in (1) reflects possible errors in modeling and measurement. Also note that, in the classical square pixel model, the image is discretized by partitioning its support into a finite number of equi-sized square regions (called pixels or cells) whose centers are those n sample points. By assuming that the density function f is constant in each of the equi-sized pixels, i. e., $f = f_j$ throughout pixel j, the value of a_{ij} in the projection matrix is simply the length of the intersection of the line corresponding to the ith projection with the pixel surrounding the jth sample point.

Once the projection matrix $\mathbf{A}$ is defined and the measurement $\mathbf{d}$ is known, the problem is to find an $\mathbf{f}$ satisfying (1). To cope with the errors mentioned above, G.T. Herman [6] suggested that (1) be replaced by an 'interval constraint' and a nonnegativity constraint be added:

$$\mathbf{d} - \boldsymbol{\epsilon} \leq \mathbf{A}\mathbf{f} \leq \mathbf{d} + \boldsymbol{\epsilon}, \tag{2}$$

$$\mathbf{f} \geq \mathbf{O}, \tag{3}$$

where $\epsilon = (\epsilon_1, \ldots, \epsilon_m)$ is an m-vector of user-chosen tolerance levels. Note that (2) can be replaced by an equivalent system of inequalities

$$\mathbf{A}'\mathbf{f} \leq \mathbf{d}', \tag{4}$$

with twice as many one-sided inequalities [2,6].

For such an image reconstruction model, we can adopt either the *'feasibility approach'* to find a solution to (2) and (3) directly, or the *'optimization approach'* to find a solution that is not only feasible in the above sense but also optimal with respect to a certain criterion. In the literature, at least three different types of optimization problems have been proposed, namely, the entropy maximization problem, the quadratic minimization problem, and the maximum likelihood problem.

The entropy optimization problem seeks to optimize an entropic objective function subject to (2) and (3) as follows.

Model 1:

$$\begin{cases} \max & -\sum_{j=1}^{n} f_j \ln f_j \\ \text{s.t.} & \mathbf{d} - \boldsymbol{\epsilon} \le \mathbf{A}\mathbf{f} \le \mathbf{d} + \boldsymbol{\epsilon}, \\ & \mathbf{f} \ge \mathbf{O}. \end{cases} \tag{5}$$

Some researchers proposed models in which the f_j's are normalized in such a way that $\sum_{j=1}^{n} f_j = 1$, and the projection matrix and the measurement data differ from those of Model 1. See, e. g., [4]. In this way, a solution that is consistent with the measurement data but remains maximally noncommittal can be found. Note that an optimal solution to such models can also be interpreted as the most probable solution that is consistent with the measurement data [3].

Other variations of Model 1 exist. Despite possible modeling and measurement errors, one common practice is to replace (1) and inequalities (2), and (5) by a system of equations: $\mathbf{A}\mathbf{f} = \mathbf{d}$.

A different version of the finite-dimensional entropy optimization model begins with the definition of an error vector $\mathbf{e} = (e_1, \ldots, e_m)^\mathsf{T}$, where

$$e_i \equiv d_i - \sum_{j=1}^{n} a_{ij} f_j, \quad i = 1, \ldots, m.$$

Assume that errors $e_1, \ldots, e_m$ exist due to imprecise measurement and are independent noise terms with zero mean and known variance σ_i^2. S.F. Burch et al. [1] observed that the strong law of large numbers implies that

$$Q(\mathbf{f}) \equiv \frac{1}{m} \sum_{i=1}^{m} \frac{\left(\sum_{j=1}^{n} a_{ij} f_j - d_i\right)^2}{\sigma_i^2} \to 1,$$

as $m \to \infty$.

Thus, if m is sufficiently large, the following entropy optimization problem with quadratic constraints can be useful:

Model 2:

$$\begin{cases} \max & -\sum_{j=1}^{n} f_j \ln f_j \\ \text{s.t.} & \frac{1}{m}(\mathbf{A}\mathbf{f} - \mathbf{d})^\top \mathbf{S}^2 (\mathbf{A}\mathbf{f} - \mathbf{d}) = 1, \\ & f_j \ge 0, \quad j = 1, \ldots, n, \end{cases}$$

where $\mathbf{S}$ is a diagonal matrix with $1/\sigma_i$ being its ith diagonal element.

Concerns such as the smoothing effect, nonuniformity, peakness, and exactness [14] of a constructed image can also be addressed in this model with proper modification of the objective functions and constraints. So far, we have used the square pixel model to illustrate the idea of entropy optimization for image reconstruction. Other models exist [2].

For an introduction to the concept of *Shannon's entropy* and related entropy optimization principles, i. e., *principle of maximum entropy* and *principle of minimum cross-entropy*, see ▶ Entropy optimization: Shannon measure of entropy and its properties. A large amount of literature has been devoted to developing iterative methods for solving finite-dimensional *entropy optimization* problems with linear and/or quadratic constraints. For details and a unification of such methods, see [3].

The method currently employed in most CT systems is the 'filtered back-projection' method, which is based on a finite-dimensional model. (See [5,10] for details.) Compared to the iterative methods for solving entropy optimization problems, this method provides speed, which enables reconstruction of the image while X-ray transmission data are being collected. Hence the time between scanning and obtaining reconstructed images is reduced. However, there are situations where iterative methods produce comparable or better reconstructed images than the filtered back-projection method, e. g., in image reconstruction with few projections or in high-contrast image reconstruction. The ever increasing computer speed and its companion reduction in cost may increase the desirability of employing iterative methods in CT systems.

In many situations, e. g., conducting diagnostic experiments on plasma in magnetic confinement devices or laser target impositions with measurements on fusion reactor cores, only few projections are available, e. g., less than 10. When the finite-dimensional entropy optimization model is applied, it tends to produce 'streaking' artifacts. This motivated the use of the vector-space model.

Take the two-dimensional X-ray CT problem as an example. By assuming that the unknown density function $f(x, y)$ is continuous over a compact support D

such that

$$f(x, y) \geq 0 \text{ and } \int\int_D f(x, y)\, dx\, dy = 1, \tag{6}$$

G. Minerbo [9] defined the entropy of $f(x, y)$ as

$$\zeta(f) = -\int\int_D f(x, y) \ln[f(x, y)A]\, dx\, dy,$$

where A is the area of D. Denote the set of continuous, nonnegative functions with compact support in D by $C_+(D)$.

The scanning area is partitioned into parallel strips, each of which is penetrated by an X-ray beam. Let $\theta_j, j = 1, \ldots, J$, be the J distinct projection angles with respect to the X-axis of the scanning area. Also let $M(j)$ be the number of parallel beams associated with the jth projection or view, and $S_{j1} < \cdots < S_{jM(j)}$ be a set of abscissas for the jth view. The projection data are assumed to be in the form of the following 'strip integrals':

$$P_{jm}(f) \equiv \int_{S_{jm}}^{S_{j(m+1)}} \int_{-\infty}^{\infty} f(s\cos\theta_j - t\sin\theta_j, s\sin\theta_j + t\cos\theta_j)\, dt\, ds,$$

where $m = 1, \ldots, M(j)$ and $j = 1, \ldots, J$. It is assumed that, for $j = 1, \ldots, J$,

$$\int_{-\infty}^{\infty} f(s\cos\theta_j - t\sin\theta_j, s\sin\theta_j + t\cos\theta_j)\, dt = 0, \quad \text{for} \quad s < S_{j1} \quad \text{or} \quad s > S_{jM(j)}.$$

Let G_{jm} denote the observed values of $P_{jm}(f)$, for $m = 1, \ldots, M(j)$, and $j = 1, \ldots, J$. Note that (6) implies $G_{jm} \geq 0$ and $\sum_{m=1}^{M(j)} G_{jm} = 1$.

Then the vector-space model results in the following *optimization problem*:

Model 3:

$$\begin{cases} \sup\limits_{C_+(D)} & \zeta(f) \\ \text{s.t.} & P_{jm}(f) = G_{jm}, \\ & \quad m = 1, \ldots, M(j); \\ & \quad j = 1, \ldots, J. \end{cases} \tag{7}$$

A finite-dimensional unconstrained dual problem can be derived by using the technique of Lagrange multipliers. An algorithm known as MENT [9] was proposed. It was shown that the solutions produced by MENT converge to a density function f^* which satisfies the constraint (7) with $\zeta(f^*) = \sup_{C_+(D)} \zeta(f)$. However, the limiting density function f^* is not continuous. Actually, as pointed out in [8], f^* is piecewise constant and $f^* \notin C_+(D)$. When few projections are available and the object being scanned has a simple structure (or close to circular symmetry in density), some preliminary computational results indicated the potential of this approach.

Recognizing the fact that the supremum of Model 3 is not attained by any function $f \in C_+(D)$, M. Klaus and R.T. Smith [8] defined an alternative formulation in a richer class of functions than $C_+(D)$. More precisely, they replaced $C_+(D)$ by $L_+^2(D)$, the set of all nonnegative square integrable functions on D, as the setting. Note that all piecewise-constant functions over D are contained in $L_+^2(D)$. Also recognizing that measurements may not be consistent and even be flawed, they considered an optimization problem where the objective function is the original entropy functional $\zeta(f)$ minus a penalty term corresponding to the residual error in meeting the measurement constraints, and the constraint is that the maximizer lies in a weakly compact set that is determined by known physical information about the density function of the object to be scanned. A corresponding formulation becomes

Model 4:

$$\sup_{f \in \Omega} G(f) \equiv \zeta(f) - \gamma \sum_{j,m} [G_{jm} - P_{jm}(f)]^2,$$

where $\gamma > 0$ is an adjustable penalty parameter and Ω is a convex and weakly (sequentially) compact set of nonnegative functions in $L_+^2(D)$, with a compact support in D and containing physical information known *a priori* about the object to be scanned, e. g., upper and lower bounds on the density function. (A set Ω of nonnegative functions in $L_+^2(D)$ is weakly (sequentially) compact if and only if every sequence in Ω has a weakly convergent subsequence whose weak limit lies in Ω; a sequence $\{f_n(x, y)\}$ converges weakly to $f(x, y)$ if and only if the sequence $\{\langle f_n(x, y), g(x, y)\rangle\}$ converges to $\langle f(x, y), g(x, y)\rangle$ for every $g(x, y) \in L_+^2(D)$, where $\langle h_1, h_2\rangle \equiv \int\int h_1(x, y)\, h_2(x, y)\, dxdy$ denotes the inner product of h_1 and h_2 in the space of $L_+^2(D)$.)

With the aid of the theory of Hilbert space, it can be shown [8] that G has a unique maximizer in Ω, for any given data $G_{jm}, m = 1, \ldots, M(j), j = 1, \ldots, J$.

Based on this alternative formulation, the density function $f(x, y)$ can be approximated by using the *finite element method* [11]. For simplicity, assume that $D = [-1, 1] \times [-1, 1]$. First, we superimpose a fixed rectangular mesh on D, with uniform mesh size $h = 1/n$ in both the x and y directions. We also use the product of piecewise linear functions in x and y as the finite element space S^h. In this way, a basis for S^h has the form

$$\psi_k(x, y) = \psi_i(x)\psi_l(y), \text{for} \quad k = 1, \ldots, (2n + 1)^2,$$

where

$$l = \left\{ \frac{(k-1) - (k-1) \pmod{2n+1}}{2n+1} \right\} - n,$$
$$i = k - (l + n)(2n + 1) - n - 1,$$

and

$$\psi_j(t) = \begin{cases} 0 & \text{if} \quad t \le (j-1)h \\ & \text{or} \quad t \ge (j+1)h, \\ \frac{t-(j-1)h}{h}, & \text{if} \quad (j-1)h \le t \le jh, \\ \frac{(j+1)h-t}{h}, & \text{if} \quad jh \le t \le (j+1)h. \end{cases}$$

It is reasonable to expect that, in practice, one should know a priori the minimum and maximum densities of the object being examined. Hence we focus on a simple constraint set

$$\Omega = \left\{ f \in L^2_+(D): \quad \begin{array}{c} 0 < a \le f \le b < \infty \text{ a.e.}, \\ f = 0 \text{ a.e., in } \mathbb{R}^2 \setminus D \end{array} \right\}.$$

The density function $f(x, y)$ is then approximated in S^h by

$$f(x, y) = \sum_{k=1}^{N} c_k \psi_k(x, y),$$

where $N = (2n + 1)^2$ and c_k's are chosen as the optimal solution of the following finite-dimensional optimization problem:

$$\begin{cases} \sup\limits_{c \in \mathbb{R}^N} & G\left(\sum\limits_{k=1}^{N} c_k \psi_k(x, y) \right) \\ \text{s.t.} & 0 < a \le \sum\limits_{k=1}^{N} c_k \psi_k(x, y) \le b. \end{cases}$$

This problem can be further reduced to

$$\begin{cases} \sup_{c \in \mathbb{R}^N} & \zeta\left(\sum\limits_{k=1}^{N} c_k \psi_k(x, y) \right) \\ & -\gamma \sum\limits_{j,m} \left[G_{jm} - \sum\limits_{k=1}^{N} c_k P_{jm}(\psi_k(x, y)) \right]^2 \\ \text{s.t.} & 0 < a \le c_k \le b, \quad k = 1, \ldots, N. \end{cases}$$

Preliminary computational results reported in [11, 12] indicate some improvements of this alternative approach over the MENT algorithm when the object under investigation does not have circular symmetry in density and has a high density area near the edge of the scanning region.

See also

- ▶ Entropy Optimization: Interior Point Methods
- ▶ Entropy Optimization: Parameter Estimation
- ▶ Entropy Optimization: Shannon Measure of Entropy and its Properties
- ▶ Jaynes' Maximum Entropy Principle
- ▶ Optimization in Medical Imaging

References

1. Burch SF, Gull SF, Skilling JK (1983) Image restoration by a powerful maximum entropy method. Computer Vision, Graphics, and Image Processing 23:113–128
2. Censor Y, Herman GT (1987) On some optimization techniques in image reconstruction. Applied Numer Math 3:365–391
3. Fang S-C, Rajasekera JR, Tsao H-SJ (1997) Entropy optimization and mathematical programming. Kluwer, Dordrecht
4. Frieden BR (1972) Restoring with maximum likelihood and maximum entropy. J Optical Soc Amer 62:511–518
5. Hendee WR (1983) The physical principles of computed tomography. Little, Brown and Company, Boston, MA
6. Herman GT (1975) A relaxation method for reconstructing objects from noisy X-rays. Math Program 8:1–19
7. Herman GT (ed) (1979) Image reconstruction from projections: implementation and applications. Springer, Berlin
8. Klaus M, Smith RT (1988) A Hilbert space approach to maximum entropy reconstruction. Math Meth Appl Sci 10:397–406
9. Minerbo G (1979) MENT: A maximum entropy algorithm for reconstructing a source from projection data. Computer Graphics and Image Processing 10:48–68
10. Natterer F (1986) Mathematics of computerized tomography. Wiley, New York

11. Smith RT, Zoltani CK (1987) An application of the finite element method to maximum entropy tomographic image reconstruction. J Sci Comput 2(3):283–295
12. Smith RT, Zoltani CK, Klem GJ, Coleman MW (1991) Reconstruction of tomographic images from sparse data sets by a new finite element maximum entropy approach. Applied Optics 30(5):573–582
13. Stark H (ed) (1987) Image recovery: theory and application. Acad. Press, New York
14. Wang Y, Lu W (1992) Multi-criterion maximum entropy image reconstruction from projections. IEEE Trans Medical Imaging 11:70–75

Maximum Flow Problem

RAVINDRA K. AHUJA[1], THOMAS L. MAGNANTI[2], JAMES B. ORLIN[3]
[1] Department Industrial and Systems Engineering, University Florida, Gainesville, USA
[2] Sloan School of Management and Department Electrical Engineering and Computer Sci., Massachusetts Institute Technol., Cambridge, USA
[3] Sloan School of Management, Massachusetts Institute Technol., Cambridge, USA

MSC2000: 90C35

Article Outline

Keywords
Applications
  Capacity of Physical Networks
Feasible Flow Problem
  Matrix Rounding Problem
Preliminaries
  Residual Network
  Flow across an $s-t$-Cut
Generic Augmenting Path Algorithm
Generic Preflow-Push Algorithm
Combinatorial Implications
  of the Max–Flow Min–Cut Theorem
  Network Connectivity
Matchings and Covers
See also
References

Keywords

Network; Maximum flow problem; Minimum cut problem; Augmenting path algorithm; Preflow-push algorithm; Max-flow min-cut theorem

The *maximum flow problem* seeks the maximum possible flow in a capacitated network from a specified source node s to a specified sink node t without exceeding the capacity of any arc. A closely related problem is the *minimum cut problem*, which is to find a set of arcs with the smallest total capacity whose removal separates node s and node t. The maximum flow and minimum cut problems arise in a variety of application settings as diverse as manufacturing, communication systems, distribution planning, matrix rounding, and scheduling. These problems also arise as subproblems in the solution of more difficult network optimization problems. In this article, we study the maximum flow and minimum cut problems, briefly introducing the underlying theory and algorithms, and presenting some applications. See [2] for a wealth of additional material that amplifies on this discussion.

Let $G = (N, A)$ be a *directed network* defined by a set N of n *nodes* and a set A of m *directed arcs*. We refer to nodes i and j as *endpoints* of arc (i, j). A *directed path* $i_1 - i_2 - \cdots - i_k$ is a set of arcs $(i_1, i_2), \ldots, (i_{k-1}, i_k)$. Each arc (i, j) has an associated *capacity* u_{ij} denoting the maximum amount of flow on this arc. We assume that each arc capacity u_{ij} is an integer, and let $U = \max\{u_{ij}:(i, j) \in A\}$. The network has two distinguished nodes, a *source node* s and a *sink node* t. To help in representing a network, we use the arc adjacency list $A(i)$ of node i, which is the set of arcs emanating from it, that is, $A(i) = \{(i, j) \in A:j \in N\}$.

The maximum flow problem is to find the maximum flow from the source node s to the sink node t that satisfies the arc capacities and mass balance constraints at all nodes. We can state the problem formally as follows.

$$\max v \tag{1}$$

subject to

$$\sum_{\{j:\ (i,j)\in A\}} x_{ij} - \sum_{\{j:\ (j,i)\in A\}} x_{ji} = \begin{cases} v & \text{for } i = s, \\ 0 & \text{for } i \notin \{s, t\}, \\ -v & \text{for } i = t, \end{cases} \tag{2}$$

$$0 \le x_{ij} \le u_{ij} \text{ for all } (i, j) \in A. \tag{3}$$

We refer to a vector $x = \{x_{ij}\}$ satisfying (2) and (3) as a *flow* and the corresponding value of the scalar variable v as the *value* of the flow. We refer to the constraints (2) as the *mass balance constraints*, and refer to the constraints (3) as the *flow bound constraints*.

In examining the maximum flow problem, we impose two assumptions:

i) all arc capacities are integer; and
ii) whenever the network contains arc (i, j), then it also contains arc (j, i).

The second assumption is nonrestrictive since we allow arcs with zero capacity.

Sometimes the flow vector x might be required to satisfy lower bound constraints imposed upon the arc flows; that is, if $l_{ij} \geq 0$ specifies the lower bound on the flow on arc $(i, j) \in A$, we impose the condition $x_{ij} \geq l_{ij}$. We refer to this problem as the *maximum flow problem with nonnegative lower bounds*. It is possible to transform a maximum flow problem with nonnegative lower bounds into a maximum flow problem with zero lower bounds.

The minimum cut problem is a close relative of the maximum flow problem. A *cut* $[S, \overline{S}]$ partitions the node set N into two subsets S and $\overline{S} = N - S$ It consists of all arcs with one endpoint in S and the other in $\overline{S}$. We refer to the arcs directed from S to $\overline{S}$, denoted by $(S, \overline{S})$, as *forward arcs* in the cut and the arcs directed from $\overline{S}$ to S, denoted by $(S, \overline{S})$, as *backward arcs* in the cut. The cut $[S, \overline{S}]$ is called an $s - t$-cut if $s \in S$ and $t \in \overline{S}$. We define the *capacity of the cut* $[S, \overline{S}]$, denoted as $u[S, \overline{S}]$, as $\sum_{(i,j)\in(S,\overline{S})} u_{ij}$. A *minimum cut* in G is an $s - t$-cut of minimum capacity. We will show that any algorithm that determines a maximum flow in the network also determines a minimum cut in the network.

The remainder of this article is organized as follows. To help in understanding the importance of the maximum flow problem, we begin by describing several applications. In the next section we present some preliminary results concerning flows and cuts. We next discuss two important classes of algorithms for solving the maximum flow problem: augmenting path algorithms, and preflow-push algorithms. As described in the next section, augmenting path algorithms augment flow along directed paths from the source node to the sink node. The proof of the validity of the augmenting path algorithm yields the well-known max-flow min-cut theorem, which implies that the value of a maximum flow in a network equals the capacity of a minimum cut in the network. In the next section, we study preflow-push algorithms that 'flood' the network so that some nodes have excesses and then incrementally 'relieve' the flow from nodes with excesses by sending flow from excess nodes forward toward the sink node or backward toward the source node. In the final section, we study implications of the max-flow min-cut theorem and prove some max-min results in combinatorics.

We would like to design maximum flow algorithms that are guaranteed to be efficient in the sense that their worst-case running times, that is, the total number of multiplications, divisions, additions, subtractions, and comparisons in the worst-case grow slowly in some measure of the problem's size. We say that a maximum flow algorithm is an $O(n^3)$ algorithm, or has a worst-case complexity of $O(n^3)$, if it is possible to solve any maximum flow problem using a number of computations that is asymptotically bounded by some constant times the term n^3. We say that an algorithm is a *polynomial time algorithm* if it's worst-case running time is bounded by a polynomial function of the input size parameters. For a maximum flow problem, the input size parameters are n, m, and $\log U$ (the number of bits needed to specify the largest arc capacity). We refer to a maximum flow algorithm as a *pseudopolynomial time algorithm* if its worst-case running time is bounded by a polynomial function of n, m, and U. For example, an algorithm with worst-case complexity of $O(nm \log U)$ is a polynomial time algorithm, but an algorithm with worst-case complexity of $O(nmU)$ is a pseudopolynomial time algorithm.

Applications

The maximum flow problem arises in a variety of situations and in several forms. Sometimes, it arises directly in combinatorial applications that on the surface might not appear to be maximum flow problems at all; at other times, it occurs as a subproblem in the solution of more difficult network optimization problems. In this section, we describe three applications of the maximum flow problem.

Capacity of Physical Networks

An oil company needs to ship oil from a refinery to a storage facility using the pipelines of its underlying

distribution network. In this problem context, the refinery corresponds to a particular node s in the distribution network and the storage facility corresponds to another node t. The capacity of each arc is the maximum amount of oil per unit time that can flow along it. The value of a maximum $s-t$ flow determines the maximum flow rate from the source node s to the sink node t. Similar applications arise in other settings, for example, determining the transmission capacity between two nodes of a telecommunications network.

Feasible Flow Problem

The feasible flow problem consists of finding a feasible flow satisfying the following constraints:

$$\sum_{\{j:\ (i,j)\in A\}} x_{ij} - \sum_{\{j:\ (j,i)\in A\}} x_{ji} = b(i) \text{for all } i \in N, \quad (4)$$

$$0 \le x_{ij} \le u_{ij} \text{ for all } (i, j) \in A. \quad (5)$$

We assume that $\sum_{i \in N} b(i) = 0$. The following distribution scenario illustrates how the feasible flow problem arises in practice. Suppose that merchandise available at several seaports is desired by other ports. We know the stock of merchandise available at the 'supply' ports, the amount required at the other ports, and the maximum quantity of merchandise that can be shipped on a particular sea route. We wish to know whether we can satisfy all of the demands by using the available supplies.

We can solve the feasible flow problem by solving a maximum flow problem defined on an augmented network as follows. We introduce two new nodes, a source node s and a sink node t. For each node i with supply (that is, with $b(i) > 0$), we add an arc (s, i) with capacity $b(i)$, and for each node i with demand (that is, with $b(i) < 0$), we add an arc (i, t) with capacity $-b(i)$. We refer to the new network as the *transformed network*. We then solve a maximum flow problem from node s to node t in the transformed network. It is easy to show that the model (4)–(5) has a feasible solution if and only if the maximum flow saturates all the arcs emanating from the source node, that is, $x_{sj} = u_{sj}$ for all arcs $(s, j) \in A(s)$. Moreover, if each $b(i)$ and u_{ij} is integer, then model (4)–(5) always has an integer feasible solution whenever it has a feasible solution (see Theorem 3).

Sometimes in a feasible flow problem arcs have nonnegative lower bounds, that is, the flow bound constraints are $l_{ij} \le x_{ij} \le u_{ij}$ instead of $0 \le x_{ij} \le u_{ij}$, for some constants $l_{ij} \ge 0$ for each $(i, j) \in A$. By substituting $y_{ij} = x_{ij} - l_{ij}$ for x_{ij}, we can transform this problem to the formulation (4)–(5). Then (5) reduces to $0 \le y_{ij} \le (u_{ij} - l_{ij})$ and (4) reduces to the same set of equations, but with a different right-hand side vector b'.

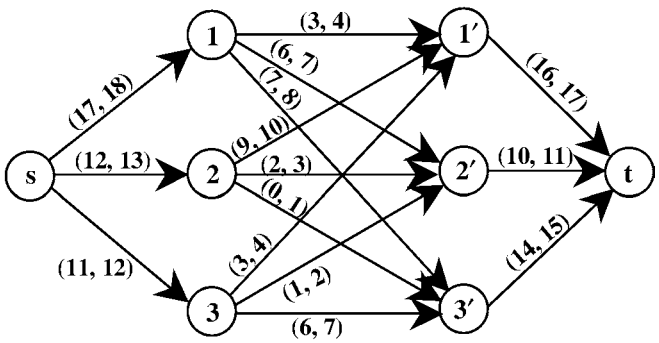
Matrix Rounding Problem

This application is concerned with consistent rounding of the elements, the row sums, and the column sums of a matrix. We are given a $p \times q$ matrix of real numbers $D = \{d_{ij}\}$, with row sums α_i and column sums β_j. We can round any real number d to the next smaller integer $\lfloor d \rfloor$ or to the next larger integer $\lceil d \rceil$, and the decision to round up or round down is entirely up to us. The matrix-rounding problem requires that we round the matrix elements, and the row and column sums of the matrix so that the sum of the rounded elements in each row equals the rounded row sum, and the sum of the rounded elements in each column equals the rounded column sum. We refer to such a rounding as a *consistent rounding*. The matrix-rounding problem arises is several application contexts, for example, the rounding of census data to disguise data on individuals.

Using a numerical example, we will show how to transform a matrix rounding problem into a maximum flow problem. Figure 1a) shows an instance of the matrix rounding problem and Fig. 1b) gives the maximum flow network G for this problem. The network G contains a node i corresponding to each row i of the matrix D, a node j corresponding to each column j of D, a source node s, and a sink node t. The network contains an arc (i, j) corresponding to the ijth element in the matrix, an arc (s, i) for each row i (this arc represents the sum of row i), an arc (j, t) for each column j (this arc represents the sum of column j). For any arc (i, j), we define its upper bound $u_{ij} = \lceil d_{ij} \rceil$ and lower bound $l_{ij} = \lfloor d_{ij} \rfloor$. Notice that the flow $x_{ij} = d_{ij}$ is a real-valued feasible flow x in the network. Since there is a one-to-one correspondence between the consistent roundings of the matrix and feasible integer flows in the corresponding network, we can find a consistent rounding

				Row Sum
	3.1	6.8	7.3	17.2
	9.6	2.4	0.7	12.7
	3.6	1.2	6.6	11.3
Column Sum	16.3	10.4	14.5	

a



b

Maximum Flow Problem, Figure 1
Network for the matrix rounding problem

by solving a feasible flow problem on the corresponding network. The feasible flow algorithm will produce an integer feasible flow (because of Theorem 3), which corresponds to a consistent rounding.

Preliminaries

In this section, we discuss some elementary properties of flows and cuts. We will use these properties to prove the celebrated max-flow min-cut theorem and to establish the correctness of the augmenting path algorithm described in the next section.

Residual Network

The concept of residual network plays a central role in the development of maximum flow algorithms. Given a flow x, the residual capacity r_{ij} of any arc $(i, j) \in A$ is the maximum additional flow that can be sent from node i to node j using the arcs (i, j) and (j, i). (Recall the assumption from the first Section that whenever the network contains arc (i, j), it also contains the arc (j, i).) The residual capacity r_{ij} has two components:

i) $u_{ij} - x_{ij}$, the unused capacity of arc (i, j);
ii) the current flow x_{ji} on arc (j, i), which we can cancel to increase the flow from node i to node j.

Consequently, $r_{ij} = u_{ij} - x_{ij} + x_{ji}$. We refer to the network $G(x)$ consisting of the arcs with positive residual capacities as the *residual network* (with respect to the flow x). Figure 2 gives an example of a residual network.

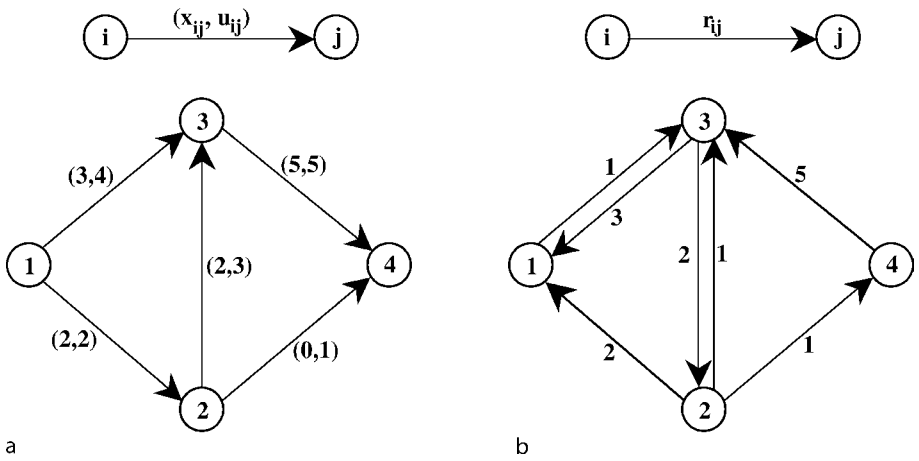
Flow across an $s - t$-Cut

Let x be a flow in the network. Adding the mass balance constraint (2) for the nodes in S, we obtain the equation

$$v = \sum_{i \in S} \left[\sum_{\{j:\ (i,j) \in A\}} x_{ij} - \sum_{\{j:\ (j,i) \in A\}} x_{ji} \right] = \sum_{(i,j) \in (S, \overline{S})} x_{ij} - \sum_{(i,j) \in (\overline{S}, S)} x_{ij}. \tag{6}$$

The second equality uses the fact that whenever both the nodes p and q belong to the node set S and $(p, q) \in A$, the variable x_{pq} in the first term within the bracket (for node $i = p$) cancels the variable $- x_{pq}$ in the second term within the bracket (for node $j = q$). The first expression in the right-hand side of (6) denotes the amount of flow from the nodes in S to nodes in $\overline{S}$, and the second expression denotes the amount of flow returning from the nodes in $\overline{S}$ to the nodes in S. Therefore, the right-hand side denotes the total (net) flow across the cut, and (6) implies that the flow across any $s - t$-cut $[S, \overline{S}]$ equals v. Substituting $x_{ij} \le u_{ij}$ in the first expression of (6) and $x_{ij} \ge 0$ in the second expression yields: $v \le \sum_{(i,j) \in (S, \overline{S})} u_{ij} = u[S, \overline{S}]$ implying that the value of any flow can never exceed the capacity of any cut in the network. We record this result formally for future reference.

Lemma *The value of any flow can never exceed the capacity of any cut in the network. Consequently, if the value of some flow x equals the capacity of some cut*



Maximum Flow Problem, Figure 2
Illustrating the construction of a residual network; a) the original network, with arc capacities and a flow *x*; b) the residual network

$[S, \overline{S}]$, *then x is a maximum flow and the cut* $[S, \overline{S}]$ *is a minimum cut.*

The *max-flow min-cut theorem*, to be proved in the next section, states that the value of some flow always equals the capacity of some cut.

Generic Augmenting Path Algorithm

In this section, we describe one of the simplest and most intuitive algorithms for solving the maximum flow problem, an algorithm known as the *augmenting path algorithm*.

Let x be a feasible flow in the network G, and let $G(x)$ denote the residual network corresponding to the flow x. We refer to a directed path from the source to the sink in the residual network $G(x)$ as an augmenting path. We define the residual capacity δ (P) of an augmenting path P as the maximum amount of flow that can be sent along it, that is, δ $(P) = \min setr_{ij}(i, j) \in P$. Since the residual capacity of each arc in the residual network is strictly positive, the residual capacity of an augmenting path is strictly positive. Therefore, we can always send a positive flow of δ units along it. Consequently, whenever the network contains an augmenting path, we can send additional flow from the source to the sink. (Sending an additional δ units of flow along an augmenting path decreases the residual capacity of each arc (i, j) in the path by δ units.) The generic augmenting path algorithm is essentially based upon this simple observation. The algorithm identifies augmenting paths in $G(x)$ and augments flow on these paths until the network contains no such path. The algorithm below describes the generic augmenting path algorithm.

We can identify an augmenting path P in $G(x)$ by using a graph search algorithm. A graph search algorithm starts at node s and progressively finds all nodes that are reachable from the source node using directed paths. Most search algorithms run in time proportional to the number of arcs in the network, that is, $O(m)$ time, and either identify an augmenting path or conclude that $G(x)$ contains no augmenting path; the latter happens when the sink node is not reachable from the source node.

```
BEGIN
    x := 0;
    WHILE G(x) contains a directed path from
    node s to node t DO
    BEGIN
        identify an augmenting path P from s to t;
        set δ := min{r_ij : (i, j) ∈ P};
        augment δ units of flow along P;
        update G(x);
    END;
END;
```

Generic augmenting path algorithm

For each arc $(i, j) \in P$, augmenting δ units of flow along P decreases r_{ij} by δ units and increases r_{ji} by δ units. The final residual capacities r_{ij} when the algorithm terminates specifies a maximum (arc) flow in the following manner. Since $r_{ij} = u_{ij} - x_{ij} + x_{ji}$, the arc flows satisfy the equality $x_{ij} - x_{ji} = u_{ij} - r_{ij}$. If $u_{ij} > r_{ij}$, we can set $x_{ij} = u_{ij} - r_{ij}$ and $x_{ji} = 0$; otherwise, we set $x_{ij} = 0$ and $x_{ji} = r_{ij} - u_{ij}$.

We use the maximum flow problem given in Fig. 3) to illustrate the algorithm. Fig 3a) shows the residual network corresponding to the starting flow $x = 0$, which is identical to the original network. The residual network contains three augmenting paths: $1 - 3 - 4$, $1 - 2 - 4$, and $1 - 2 - 3 - 4$. Suppose the algorithm selects the path $1 - 3 - 4$ for augmentation. The residual capacity of this path is $\delta = \min\{r_{13}, r_{34}\} = \min\{4, 5\} = 4$. This augmentation reduces the residual capacity of arc (1, 3) to zero (thus we delete it from the residual network) and increases the residual capacity of arc (3, 1) to 4 (so we add this arc to the residual network). The augmentation also decreases the residual capacity of arc (3, 4) from 5 to 1, and increases the residual capacity of arc (4, 3) from 0 to 4. Figure 3b) shows the residual network at this stage. In the second iteration, the algorithm selects the path $1 - 2 - 3 - 4$ and augments 1 unit of flow; Fig. 3c) shows the residual network after the augmentation. In the third iteration, the algorithm augments one unit of flow along the path $1 - 2 - 4$. Figure 3d) shows the corresponding residual network. Now the residual network contains no augmenting path and so the algorithm terminates.

Does the augmenting path algorithm always find a maximum flow? The algorithm terminates when the search algorithm fails to identify a directed path in $G(x)$ from node s to node t, indicating that no such path exists (we prove later that the algorithm would terminate finitely). At this stage, let S denote the set of nodes in N that are reachable in $G(x)$ from the source node using directed paths, and $\overline{S} = N - S$. Clearly, $s \in S$ and $t \notin \overline{S}$. Since the search algorithm cannot reach any node in $\overline{S}$ and it can reach each node in S, we know that $r_{ij} = 0$ for each $(i, j) \in (S, \overline{S})$. Recall that $r_{ij} = (u_{ij} - x_{ij}) + x_{ji}$, $x_{ij} \leq u_{ij}$, and $x_{ji} \geq 0$. If $r_{ij} = 0$, then $x_{ij} = u_{ij}$ and $x_{ji} = 0$. Since $r_{ij} = 0$ for each $(i, j) \in (S, \overline{S})$, by substituting these flow values in expression (6), we find that $v = u[S, \overline{S}]$. Therefore, the value of the current flow x equals the capacity of the cut. Lemma 1 implies that x is a maximum flow and $[S, \overline{S}]$ is a minimum cut. This conclusion establishes the correctness of the generic augmenting path algorithm and, as a byproduct, proves the following *max-flow min-cut theorem.*

Theorem 2 *The maximum value of the flow from a source node s to a sink node t in a capacitated network equals the minimum capacity among all $s - t$-cuts.*

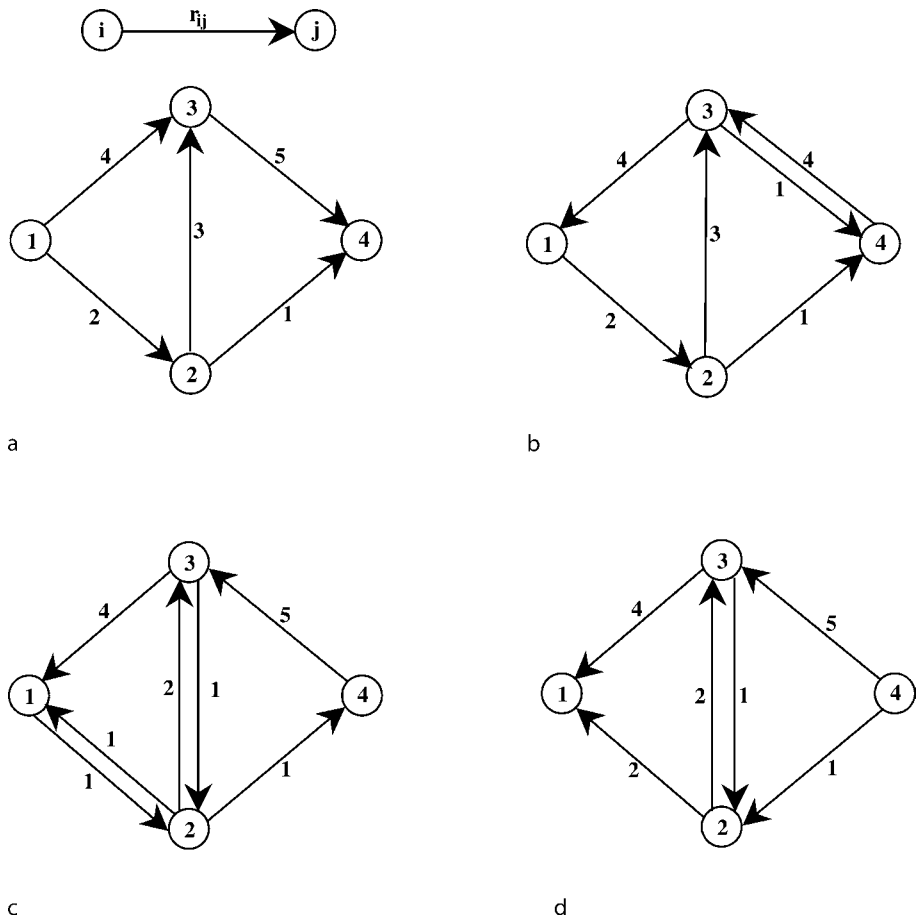
The proof of the max-flow min-cut theorem shows that when the augmenting path algorithm terminates, it also discovers a minimum cut $[S, \overline{S}]$, with S defined as the set of all nodes reachable from the source node in the residual network corresponding to the maximum flow. For our previous numerical example, the algorithm finds the minimum cut in the network, which is $[S, \overline{S}]$ with $S = \{1\}$.

The augmenting path algorithm also establishes another important result, the *integrality theorem*:

Theorem 3 *If all arc capacities are integer, then the maximum flow problem always has an integer maximum flow.*

This result follows from the facts that the initial (zero) flow is integer and all arc capacities are integer; consequently, all initial residual capacities will be integer. Since subsequently all arc flows change by integer amounts (because residual capacities are integer), the residual capacities remain integer throughout the algorithm. Further, the final integer residual capacities determine an integer maximum flow. The integrality theorem does not imply that every optimal solution of the maximum flow problem is integer. The maximum flow problem might have noninteger solutions and, most often, it has such solutions. The integrality theorem shows that the problem always has at least one integer optimal solution.

What is the worst-case running time of the algorithm? An augmenting path is a directed path in $G(x)$ from node s to node t. We have seen earlier that each iteration of the algorithm requires $O(m)$ time. In each iteration, the algorithm augments a positive integer amount of flow from the source node to the sink node. To bound the number of iterations, we will determine a bound on the maximum flow value. By definition, U denotes the largest arc capacity, and so the capacity of the cut $(\{s\}, S - \{s\})$ is at most nU. Since the value of any



Maximum Flow Problem, Figure 3
Illustrating the augmented path algorithm: a) the residual network $G(x)$ for $x = 0$; b) the residual network after augmenting four units along the path (1 — 3 — 4); c) the residual network after augmenting one unit along the path (1 — 2 — 3 — 4); d) the residual network after augmenting one unit along the path (1 — 2 — 4)

flow can never exceed the capacity of any cut in the network, we obtain a bound of nU on the maximum flow value and also on the number of iterations performed by the algorithm. Consequently, the running time of the algorithm is $O(nmU)$, which is a pseudopolynomial time bound. We summarize the preceding discussion with the following theorem.

Theorem 4 *The generic augmenting path algorithm solves the maximum flow problem in O(nmU) time.*

The augmenting path algorithm is possibly the simplest algorithm for solving the maximum flow problem. Empirically, the algorithm performs reasonably well. However, the worst-case bound on the number of iterations is poor for large values of U. For example, if $U = 2^n$, the bound is exponential in the number of nodes. Moreover, as shown by known examples, the algorithm can indeed perform these many iterations. A second drawback of the augmenting path algorithm is that if the capacities are irrational, the algorithm might not terminate. For some pathological instances of the maximum flow problem, the augmenting path algorithm does not terminate in a finite number of iterations and although the successive flow values converge to some value, they might converge to a value strictly less than the maximum flow value. (Note, however, that the max-flow min-cut theorem is valid even if arc capacities are irrational.) Therefore, if the augmenting path algorithm is to be guaranteed to be effective in all situations, it must select augmenting paths carefully.

Researchers have developed specific implementations of the generic augmenting path algorithms that overcome these drawbacks. Of these, the following three implementations are particularly noteworthy:

i) the maximum capacity augmenting path algorithm which always augments flow along a path in the residual network with the maximum residual capacity and can be implemented to run in $O(m^2 \log U)$ time;
ii) the capacity scaling algorithm which uses a scaling technique on arc capacities and can be implemented to run in $O(nm \log U)$ time;
iii) the shortest augmenting path algorithm which augments flow along a shortest path (as measured by the number of arcs) in the residual network and runs in $O(n^2 m)$ time.

These algorithms are due to J. Edmonds and R.M. Karp [6], H.N. Gabow [9], and E.A. Dinic [5], respectively. L.R. Ford and D.R. Fulkerson [8] and P. Elias, A. Fenstein and C.E. Shannon [7] independently developed the basic augmenting path algorithm.

Generic Preflow-Push Algorithm

Another class of algorithms for solving the maximum flow problem, known as *preflow-push algorithms*, is more decentralized than augmenting path algorithms. Augmenting path algorithms send flow by augmenting along a path. This basic operation further decomposes into the more elementary operation of sending flow along individual arcs. Sending a flow of δ units along a path of k arcs decomposes into k basic operations of sending a flow of δ units along each of the arcs of the path. We shall refer to each of these basic operations as a *push*. The preflow-push algorithms push flows on individual arcs instead of on augmenting paths.

A path augmentation has one advantage over a single push: it maintains conservation of flow at all nodes. The preflow-push algorithms violate conservation of flow at all steps except at the very end, and instead maintain a 'preflow' at each iteration. A *preflow* is a vector x satisfying the flow bound constraints and the following relaxation of the mass balance constraints (2):

$$\sum_{\{j:\ (i,j)\in A\}} x_{ij} - \sum_{\{j:\ (j,i)\in A\}} x_{ji} \geq 0 \quad \text{for all } i \in N - \{s,t\}. \tag{7}$$

Each element of a preflow vector is either a real number or equals $+\infty$. The preflow-push algorithms maintain a preflow at each intermediate stage. For a given preflow x, we define the excess for each node $i \in \mathrm{N} - \{s,t\}$ as

$$e(i) = \sum_{\{j:\ (j,i)\in A\}} x_{ji} - \sum_{\{j:\ (j,i)\in A\}} x_{ij}.$$

We refer to a node with positive excess as an active node. We adopt the convention that the source and sink nodes are never active. In a preflow-push algorithm, the presence of an active node indicates that the solution is infeasible. Consequently, the basic operation in this algorithm is to select an active node i and try to remove the excess by pushing flow out of it. When we push flow out of an active node, we need to do it carefully. If we just push flow to an adjacent node in an arbitrary manner and the other nodes do the same, then it is conceivable that some nodes keep pushing flow among themselves resulting in an infinite loop, which is not a desirable situation. Since ultimately we want to send the flow to the sink node, it seems reasonable for an active node to push flow to another node that is 'closer' to the sink. If all nodes maintain this rule, then the algorithm could never encounter an infinite loop. The concept of distance labels defined next allows us to implement this algorithmic strategy.

The preflow-push algorithms maintain a *distance label* $d(i)$ with each node in the network. The distance labels are nonnegative (finite) integers defined with respect to the residual network $G(x)$. We say that distance labels are valid with respect to a flow x if they satisfy the following two conditions:

$$d(t) = 0, \tag{8}$$

$$d(i) \leq d(j) + 1 \quad \text{for every arc } (i,j) \text{ in the residual network } G(x). \tag{9}$$

We refer to the conditions (8) and (9) as the *validity conditions*. It is easy to demonstrate that $d(i)$ is a lower bound on the length of any directed path (as measured by number of arcs) from node i to node t in the residual network, and thus is a lower bound on the length of the shortest path between nodes i and j. Let $i = i_1 - \cdots - i_k - t$ be any path of length k in the residual network

from node i to node t. The validity conditions (8), (9) imply that $d(i) = d(i_1) \le d(i_2) + 1$, $d(i_2) \le d(i_3) + 1, \ldots,$ $d(i_k) \le d(t) + 1 = 1$. Adding these inequalities shows that $d(i) \le k$ for any path of length k in the residual network, and therefore any (shortest) path from node i to node t contains at least $d(i)$ arcs. We say that an arc (i, j) in the residual network is *admissible* if it satisfies the condition $d(i) = d(j) + 1$; we refer to all other arcs as *inadmissible*.

The basic operation in the preflow-push algorithm is to select an active node i and try to remove the excess by pushing flow to a node with smaller distance label. (We will use the distance labels as estimates of the length of the shortest path to the sink node.) If node i has an admissible arc (i, j), then $d(j) = d(i) - 1$ and the algorithm sends flow on admissible arcs to relieve the node's excess. If node i has no admissible arc, then the algorithm increases the distance label of node i so that node i has an admissible arc. The algorithm terminates when the network contains no active nodes, that is, excess resides only at the source and sink nodes. The next algorithm describes the generic preflow-push algorithm.

```
BEGIN
    set x := 0 and d(j) := 0 for all j ∈ ℕ;
    set x_sj = u_sj for each arc (s, j) ∈ A(s);
    d(s) := n;
    WHILE residual network G(x) contains an active node
    DO
      BEGIN
        select an active node I;
        push/relabel(i);
      END;
END;
```

```
procedure push/relabel(i);
BEGIN
    IF network contains an admissible arc (i, j)
    THEN push δ := min{e(i), r_{i,j}} units of flow
    from node i to node j
    ELSE replace d(i) by
      min{d(j) + 1 : (i, j) ∈ A(i), r_ij > 0};
END;
```

The generic preflow-push algorithm

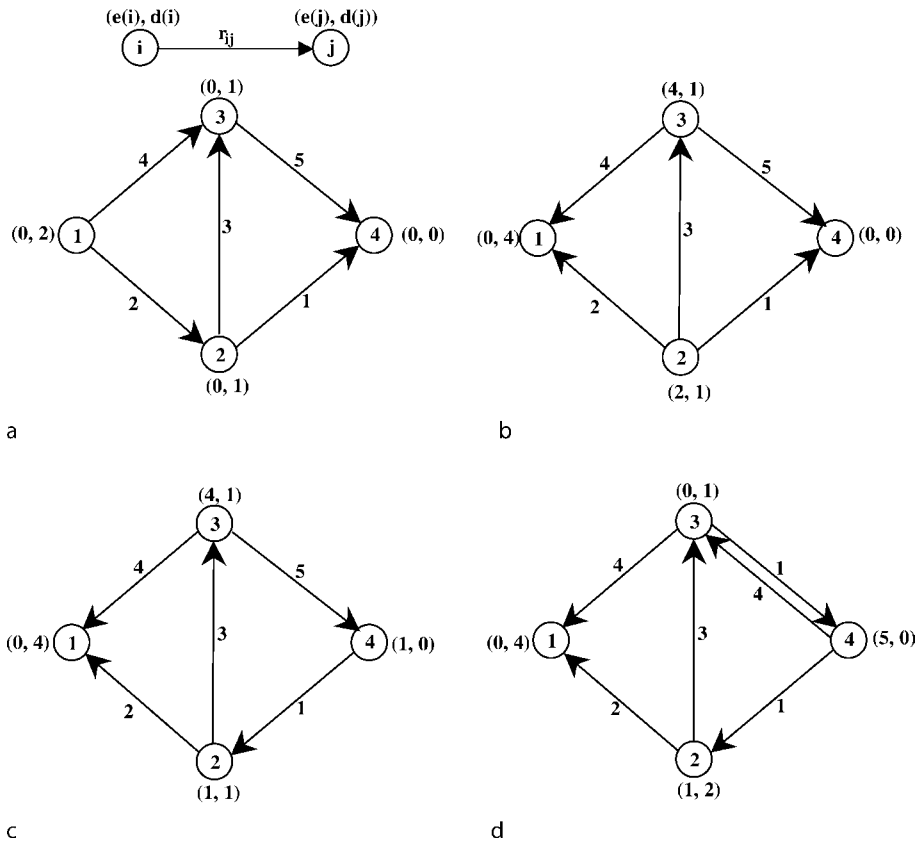
The algorithm first saturates all arcs emanating from the source node; then each node adjacent to node s has a positive excess, so that the algorithm can begin pushing flow from active nodes. Since the preprocessing operation saturates all the arcs incident to node s, none of these arcs is admissible and setting $d(s) = n$ will satisfy the validity condition (8), (9). But then, since $d(s) = n$, and a distance label is a lower bound on the length of the shortest path from that node to node t, the residual network contains no directed path from s to t. The subsequent pushes maintain this property and drive the solution toward feasibility. Consequently, when there are no active nodes, the flow is a maximum flow.

A push of δ units from node i to node j decreases both the excess $e(i)$ of node i and the residual r_{ij} of arc (i, j) by δ units and increases both $e(j)$ and r_{ji} by δ units. We say that a push of δ units of flow on an arc (i, j) is *saturating* if $d = r_{ij}$ and is *nonsaturating* otherwise. A nonsaturating push at node i reduces $e(i)$ to zero. We refer to the process of increasing the distance label of a node as a *relabel* operation. The purpose of the relabel operation is to create at least one admissible arc on which the algorithm can perform further pushes.

It is instructive to visualize the generic preflow-push algorithm in terms of a physical network: arcs represent flexible water pipes, nodes represent joints, and the distance function measures how far nodes are above the ground. In this network, we wish to send water from the source to the sink. We visualize flow in an admissible arc as water flowing downhill. Initially, we move the source node upward, and water flows to its neighbors. Although we would like water to flow downhill toward the sink, occasionally flow becomes trapped locally at a node that has no downhill neighbors. At this point, we move the node upward, and again water flows downhill toward the sink.

Eventually, no more flow can reach the sink. As we continue to move nodes upward, the remaining excess flow eventually flows back toward the source. The algorithm terminates when all the water flows either into the sink or flows back to the source.

To illustrate the generic preflow-push algorithm, we use the example given in Fig 4. Figure 4a) specifies the initial residual network. We first saturate the arcs emanating from the source node, node 1, and set $d(1) = n = 4$. Fig 4b) shows the residual graph at this stage. At



Maximum Flow Problem, Figure 4
Illustrating the preflow-push algorithm: a) the residual network $G(x)$ for $x = 0$; b) the residual network after saturating arcs emanating from the source; c) the residual network after pushing flow on arc (2, 4); d) the residual network after pushing flow on arc (3, 4)

this point, the network has two active nodes, nodes 2 and 3. Suppose that the algorithm selects node 2 for the push/relabel operation. Arc (2, 4) is the only admissible arc and the algorithm performs a saturating push of value $\delta = \min\{e(2), r_{24}\} = \min\{2, 1\} = 1$. Fig 4c) gives the residual network at this stage. Suppose the algorithm again selects node 2. Since no admissible arc emanates from node 2, the algorithm performs a relabel operation and gives node 2 a new distance label $d(2) = \min\{d(3)+ 1, d(1)+ 1\} = \min\{2, 5\} = 2$. The new residual network is the same as the one shown in Fig 4c) except that $d(2) = 2$ instead of 1. Suppose this time the algorithm selects node 3. Arc (3, 4) is the only admissible arc emanating from node 3, and so the algorithm performs a nonsaturating push of value $\delta = \min\{e(3), r_{34}\} = \min\{4, 5\} = 4$. Fig 4d) specifies the residual network at the end of this iteration. Using this process for a few more iterations, the algorithm will determine a maximum flow.

The analysis of the computational (worst-case) complexity of the generic preflow-push algorithm is somewhat complicated. Without examining the details, we might summarize the analysis as follows. It is possible to show that the preflow-push algorithm maintains valid distance labels at all steps of the algorithm and increases the distance label of any node at most $2n$ times. The algorithm performs $O(nm)$ saturating pushes and $O(n^2m)$ nonsaturating pushes. The nonsaturating pushes are the limiting computational operation of the algorithm and so it runs in $O(n^2m)$ time.

The preflow-push algorithm has several attractive features, particularly its flexibility and its potential for further improvements. Different rules for selecting active nodes for the push/relabel operations create many

different versions of the generic algorithm, each with different worst-case complexity. As we have noted, the bottleneck operation in the generic preflow-push algorithm is the number of nonsaturating pushes and many specific rules for examining active nodes can produce substantial reductions in the number of nonsaturating pushes. The following specific implementations of the generic preflow-push algorithms are noteworthy:

i) the FIFO preflow-push algorithm examines the active nodes in the first-in, first-out (FIFO) order and runs in $O(n^3)$ time;
ii) the highest label preflow-push algorithm pushes flow from an active node with the highest value of a distance label and runs in $O(n^2 m^{1/2})$ time; and
iii) the excess-scaling algorithm uses the scaling of arc capacities to attain a time bound of $O(nm + n^2 \log U)$.

These algorithms are due to A.V. Goldberg and R.J. Tarjan [10], J. Cheriyan and S.N. Maheshwari [4], and R.K. Ahuja and J.B. Orlin [3], respectively. These preflow-push algorithms are more general, more powerful, and more flexible than augmenting path algorithms. The best preflow-push algorithms currently outperform the best augmenting path algorithms in theory as well as in practice (see, for example, [1]).

Combinatorial Implications of the Max–Flow Min–Cut Theorem

The max-flow min-cut theorem has far reaching consequences. It can be used to prove several important results in combinatorics that appear to be difficult to prove using other means. We will illustrate the use of the max-flow min-cut theorem to prove two such important results.

Network Connectivity

Given a directed network $G = (N, A)$ and two specified nodes s and t, we are interested in the following two questions:

i) what is the maximum number of arc-disjoint (directed) paths from node s to node t; and
ii) what is the minimum number of arcs that we should remove from the network so that it contains no directed paths from node s to node t.

We will show that these two questions are closely related. The second question shows how robust a network, for example, a telecommunications network, is to the failure of its arcs.

In the network G, let us define the capacity of each arc as equal to one. Consider any feasible flow x of value v in the resulting unit capacity network. We can decompose the flow x into flows along v directed paths from node s to node t, each path carrying a unit flow. Now consider any $s - t$-cut $[S, \overline{S}]$ in the network. The capacity of this cut is $\left|(S, \overline{S})\right|$ that is, equals the number of forward arcs in the cut. Since each path joining nodes s and t contains at least one arc in the set $(S, \overline{S})$, the removal of all the arcs in $(S, \overline{S})$ disconnects all paths from node s to node t. Consequently, the network contains a disconnecting set of arcs of cardinality equal to the capacity of any $s - t$-cut $[S, \overline{S}]$. The max-flow min-cut theorem immediately implies the following result:

Corollary 5 *The maximum number of arc-disjoint paths from s to t in a directed network equals the minimum number of arcs whose removal will disconnect all paths from node s to node t.*

Matchings and Covers

The max-flow min-cut theorem also implies a max-min result concerning matchings and node covers in a directed bipartite network $G = (N_1 \cup N_2, A)$, with arc set $A \subseteq N_1 \times N_2$. In the network G, a subset $M \subseteq A$ is a *matching* if no two arcs in M have an endpoint in common. A subset $C \subseteq N_1 N_2$ is a *node cover* of G if every arc in A has at least one endpoint in the node set C. Suppose we create the network G' from G by adding two new nodes s and t, as well as arcs (s, i) of capacity 1 for each $i \in N_1$ and arcs (j, t) of capacity 1 for each $j \in N_2$. All other arcs in G' correspond to the arcs in G and have infinite capacity. It is possible to show that each matching of cardinality v defines a flow of value v in G', and each $s - t$ cut of capacity v induces a corresponding node cover with v nodes. Consequently, the max-flow min-cut theorem establishes the following result:

Corollary 6 *In a bipartite network $G = (N_1 \cup N_2, A)$, the maximum cardinality of any matching equals the minimum cardinality of any node cover of G.*

These two examples illustrate important relationships between maximum flows, minimum cuts, and many other problems in the field of combinatorics. The maximum flow problem is of interest because it provides

a unifying tool for viewing many such results, because it arises directly in many applications, and because it has been a rich arena for developing new results concerning the design and analysis of algorithms.

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Nonoriented Multicommodity Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Ahuja RK, Kodialam M, Mishra AK, Orlin JB (1997) Computational investigations of maximum flow algorithms. Europ J Oper Res 97:509–542
2. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs, NJ
3. Ahuja RK, Orlin JB (1989) A fast and simple algorithm for the maximum flow problem. Oper Res 37:748–759
4. Cheriyan J, Maheshwari SN (1989) Analysis of preflow-push algorithms for maximum network flow. SIAM J Comput 18:1057–1086
5. Dinic EA (1970) Algorithm for solution of a problem of maximum flow in networks with power estimation. Soviet Math Dokl 11:1277–1280
6. Edmonds J, Karp RM (1972) Theoretical improvements in algorithmic efficiency for network flow problems. J ACM 19:248–264
7. Elias P, Feinstein A, Shannon CE (1956) Note on maximum flow through a network. IRE Trans Inform Theory IT-2:117–119
8. Ford LR, Fulkerson DR (1956) Maximal flow through a network. Canad J Math 8:399–404
9. Gabow HN (1985) Scaling algorithms for network problems. J Comput Syst Sci 31:148–168
10. Goldberg AV, Tarjan RE (1988) A new approach to the maximum flow problem. J ACM 35:921–940, also: Proc. 19th ACM Symp. Theory of Computing, pp 136–146

Maximum Likelihood Detection via Semidefinite Programming

MIKALAI KISIALIOU, ZHI-QUAN LUO
Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, USA

MSC2000: 65Y20, 68W25, 90C27, 90C22, 49N15

Article Outline

Abstract
Keywords and Phrases
Introduction
Formulation
  System Model
  Connection with Unconstrained Optimization
  Semidefinite Relaxation Strategy
  Bit-Error-Rate Performance
Method
  SDP Solver
  Randomized Rounding Procedure
Cases
  Performance of SDR Detector
  Simulation Results
Conclusions
References

Abstract

Maximum-likelihood detection is a generic NP-hard problem in digital communications which requires efficient solution in practice. Some existing quasi-maximum-likelihood detectors achieve polynomial complexity with significant bit-error-rate performance degradation (e. g. LMMSE Detector), while others exhibit near-maximum-likelihood bit-error-rate performance with exponential complexity (e. g. Sphere Decoder and its variants). We present an efficient suboptimal detector based on a semidefinite relaxation, called SDR Detector, which enjoys near-maximum-likelihood bit-error-rate with worst-case polynomial complexity.

SDR Detector can be implemented with recently developed Interior-Point methods for convex optimization problems. For large systems SDR Detector provides a constant factor approximation for the maximum-likelihood detection problem. In high signal-to-noise ratio region SDR Detector can solve the maximum-likelihood detection problem exactly. Efficient implementations of SDR Detector empirically deliver a near-optimal bit-error-rate with running time that scales well to large problems and in any signal-to-noise ratio region.

Keywords and Phrases

Maximum-likelihood detection; Multiple-input multiple-output systems; Multiuser detection; Semidefinite relaxation

Introduction

Maximum-Likelihood (ML) detection is a fundamental problem in digital communications. Under the mild assumption of equiprobable transmitted signals ML Detector achieves the best Bit-Error-Rate (BER). In general, the ML detection problem is NP-hard due to the discrete nature of a signal constellation. The exhaustive search can be applied for small problem sizes, however this strategy is not practical for large systems. Large communication systems often arise in schemes with efficient rate and diversity utilization, e. g. the systems based on Linear Dispersion Codes [6]. Various suboptimal detectors that have been developed to approximate ML Detector can be divided into two major categories:

- Accelerated versions of ML Detector with exponential complexity (e. g. versions of Sphere Decoder [3,16]),
- Polynomial complexity detectors with significant degradation in the BER performance (e. g. Linear Minimum Mean Square Error (LMMSE) Detector, Matched Filter, Decorrelator, etc.).

We focus on an alternative detector which is based on a semidefinite relaxation of the ML detection problem. This detector, called SDR Detector hereafter, enjoys a worst-case polynomial complexity while delivering a near-optimal BER performance. In the next subsection we will introduce notations and a system model used throughout the text.

Formulation

System Model

Consider a vector communication channel with n transmit and m receive antennas. In wireless communications a Rayleigh fading model is widely used in scenarios with significantly attenuated line-of-sight signal component. An abundant research is based on this model which is used in profound theoretical results on channel capacity, diversity and multiplexing gain. Define a fading coefficient from the ith transmit antenna to the kth receive antenna to be a Gaussian zero-mean unit-variance, $\mathcal{N}(0,1)$, variable H_{ki}, with a Rayleigh distributed amplitude $|H_{ki}|$ and a uniformly distributed phase $\phi(H_{ki})$. The coefficients H_{ki} are assumed to be spatially and temporarily independent and identically distributed (i.i.d.). The transmitted signals $\mathbf{s} = [s_1, \ldots, s_n]^T$ are drawn from a discrete n-dimensional complex set C^n. The communication system is operating at an average Signal-to-Noise Ratio (SNR) denoted by ρ. Noise samples at each receive antenna, $\nu_k, k = 1, \ldots, m$, are modelled as i.i.d. $\mathcal{N}(0,1)$ random variables. With these notations a Rayleigh memoryless vector channel can be represented by:

$$\mathbf{y} = \sqrt{\rho/n}\,\mathbf{H}\,\mathbf{s} + \mathbf{v}\,. \tag{1}$$

The coefficient $\sqrt{\rho/n}$ ensures that the expected value of SNR at each receive antenna is equal to ρ independent of problem dimension n. Channel model (1) is quite generic and can be used to describe other communication systems, for example, a synchronous CDMA multi-access channel, where n denotes the number of users in the system.

In the sequel, we will assume that the receiver has perfect information of the fading matrix $\mathbf{H}$. In practice $\mathbf{H}$ is estimated by sending training signals which are known to the receiver. Given the vector of received signals $\mathbf{y}$ and the channel state $\mathbf{H}$, the optimal detector computes an estimate of transmitted signals such that the probability of an erroneous decision is minimized. For equiprobable input signals the minimal error probability is achieved by ML Detector given by:

$$\mathbf{s}_{\mathrm{ML}} = \arg\max_{\mathbf{s}\in C^n} p(\mathbf{y}|\mathbf{s},\mathbf{H})\,,$$

where $p(\cdot|\cdot)$ is a conditional probability density function and $\mathbf{s}_{\mathrm{ML}}$ denotes the ML estimate of transmitted

signals. For Gaussian noise this optimization problem can be stated in the form of the Integer Least Squares (ILS) problem:

$$\mathbf{s}_{\mathrm{ML}} = \arg\min_{\mathbf{s}\in C^n} \|\mathbf{y} - \sqrt{\rho/n}\,\mathbf{H}\,\mathbf{s}\|^2 . \tag{2}$$

In general, this optimization problem is NP-hard and the discrete constraint set C^n of dimension n is the source of intractability. We are interested in an efficient polynomial time approximation algorithm for (2) with theoretical performance guarantees. In the next section we will briefly discuss common approaches to solving problem (2).

Connection with Unconstrained Optimization

Several strategies have been developed to overcome high computational complexity of ML Detector. Some detectors achieve polynomial complexity by relaxing the integer constraint in the ML detection problem (2), e. g. LMMSE Detector, Decorrelator, and Matched Filter [5]. From the perspective of optimization theory these detectors can be jointly treated by dropping the discrete constraint in (2) and imposing a penalty function instead. For the BPSK constellation the relaxed problem can be written as:

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}\in\mathbb{R}^n} \|\mathbf{y} - \sqrt{\rho/n}\,\mathbf{H}\,\mathbf{s}\|^2 + \gamma\,\|\mathbf{s}\|^2 . \tag{3}$$

The modified optimization problem is usually followed by a rounding procedure which projects the optimal solution of the relaxed problem onto set C^n. Selecting proper values for γ, we can specialize (3) to LMMSE Detector, Decorrelator, or Matched Filter. An appealing advantage of this approach is that it can be solved analytically:

$$\hat{\mathbf{s}} = \mathrm{sign}\left(\left(\frac{\rho}{n}\,\mathbf{H}^T\mathbf{H} + \gamma\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{y}\right) . \tag{4}$$

This strategy achieves complexity $\mathcal{O}(n^3)$ while sacrificing the BER performance.

Another type of detectors preserves the near-ML BER while reducing the high complexity of the exhaustive search. The work originates in [3,16] with the algorithm to find the shortest vector on a lattice, known as the so-called Sphere Decoder. The algorithm reduces the exhaustive search to an ellipse centered at the zero-forcing estimate of the transmitted signals:

$$\mathbf{s}_{ZF} = \sqrt{n/\rho}\,\left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{y} .$$

Different variants of this approach use various intelligent strategies of the radius selection and ordering of points to be searched inside the ellipse. In high SNR region for small problem sizes Sphere Decoder empirically demonstrates fast running time [7]. However, a thorough theoretical analysis [9,10] has shown that both the worst-case and expected complexity of this algorithm is still exponential.

Semidefinite Relaxation Strategy

We consider an alternative approach to solve (2) which is based on a convex relaxation of the ML detection problem. Convexity of an optimization problem is a good indicator of problem tractability. Efficient and powerful algorithms with complexity $\mathcal{O}(n^{3.5})$ have recently been developed to solve convex optimization problems (e. g. Interior-Point methods). These algorithms make efficient use of theoretically computable stopping criteria, enjoy robustness, and offer the certificate of infeasibility when no solution exists. All these properties render convex optimization methods a primary tool for various fields of engineering.

There are several generic types of convex problems, the simplest one being a Linear Program (LP), i. e. the optimization problem with a linear objective function and linear constraints. An LP allows natural generalization of the notion of an inequality constraint to a so-called Linear Matrix Inequality (LMI). Instead of the regular componentwise meaning of the inequality in LP, LMI $\mathbf{X} \succeq 0$ implies that $\mathbf{X}$ belongs to the cone of symmetric positive semidefinite matrices, i. e. all eigenvalues of $\mathbf{X}$ are non-negative. Such generalization leads us to a generic class of Semi-Definite Programs (SDP), which can be written in the standard form as follows:

$$\begin{aligned} \min \quad & \mathbf{Q}\bullet\mathbf{X} \\ \text{s.t.} \quad & \mathbf{A}_k\bullet\mathbf{X} = b_k, \quad k = 1,\ldots,K, \\ & \mathbf{X}\succeq 0 , \end{aligned} \tag{5}$$

where $(\bullet)$ denotes inner product in the matrix space: $\mathbf{Q}\bullet\mathbf{X} = \mathrm{Tr}(\mathbf{QX})$. The class of SDP problems (5) includes Linear Programs as well as Second Order Cone Programs as special cases. It is quite remarkable that any problem (5) in the broad class of SDP problems can be solved in polynomial time, which makes it a valuable asset for solving engineering problems, includ-

ing filter design, control, VLSI circuit layout design, etc. [2].

In addition to application in numerical solvers, SDP formulation (5) is widely used for analysis and design of approximation algorithms for NP-hard problems. Traditional approaches involve relaxation of an NP-hard problem to an LP, which can be easily solved in polynomial time. With the invent of Interior-Point methods for non-linear convex optimization problems some approximation algorithms have been significantly improved [4]. Such advanced non-linear approximation algorithms use weaker relaxations, thereby preserving most of the structure of the original NP-hard problem. The class of SDP problems represents a perfect candidate for design of approximation algorithms since the SDP form is quite generic. The solution to the original NP-hard problem is generated from the solution of the relaxed SDP problem by a randomized or deterministic rounding procedure. For example, as will be shown later, the ML detection problem can be formulated as

$$\begin{aligned} f_{\mathrm{ML}} := \min \quad & \mathbf{Q} \bullet \mathbf{X} \\ \text{s.t.} \quad & X_{i,i} = 1, \quad i = 1, \ldots, n+1 \\ & \mathbf{X} \succeq 0 \\ & \mathbf{X} \text{ is rank-1} \,. \end{aligned} \tag{6}$$

Relaxing the rank constraint of $\mathbf{X}$ reduces the problem to the standard SDP form (5):

$$\begin{aligned} f_{\mathrm{SDP}} := \min \quad & \mathbf{Q} \bullet \mathbf{X} \\ \text{s.t.} \quad & X_{i,i} = 1, \quad i = 1, \ldots, n+1 \\ & \mathbf{X} \succeq 0 \,. \end{aligned} \tag{7}$$

A subsequent rounding procedure generates an estimate of the transmitted signals with an objective value denoted f_{SDR} based on the optimal solution $\mathbf{X}_{\mathrm{opt}}$ of this SDP problem.

Since SDR Detector outputs an estimate that belongs to the feasible set of the ML detection problem, the optimal objective value f_{SDR} of SDR Detector satisfies $f_{\mathrm{ML}} \leq f_{\mathrm{SDR}}$. Let f_{opt} (f_{apr}) denote the optimal objective value of an NP-hard problem (approximation algorithm) in minimization form, then the approximation algorithm with ratio $c \geq 1$ guarantees to provide a solution with objective value f_{apr} such that $f_{\mathrm{apr}} \leq c f_{\mathrm{opt}}$. The quality of SDR Detector can be measured in terms of approximation ratio c such that:

$$f_{\mathrm{ML}} \leq f_{\mathrm{SDR}} \leq c f_{\mathrm{ML}}, \quad c \geq 1 \,,$$

where c is independent of problem size.

Relaxation (5) was first applied to combinatorial optimization in [4] where the authors relaxed MAX-CUT problem to an SDP problem in the standard form (5). This strategy resulted in a substantial improvement of the approximation ratio for MAX-CUT problem, as compared to the classical relaxation to an LP. Unfortunately, we can not pursue this approach because the ML detection problem involves minimization instead of maximization (for a positive semidefinite matrix $\mathbf{Q}$) used in the formulation of MAX-CUT problem. Moreover, the ML detection problem does not allow a constant factor approximation algorithm for the worst case realizations of $\mathbf{H}$ and $\mathbf{v}$. However, from the perspective of digital communications we are interested in the average performance of SDR Detector over many channel and noise realizations. It turns out that SDR Detector allows a probabilistic approximation ratio for the random channel model (1). In high SNR region a typical behavior of the detection error probability is

$$P_e \simeq e^{-\gamma(\rho)} \,,$$

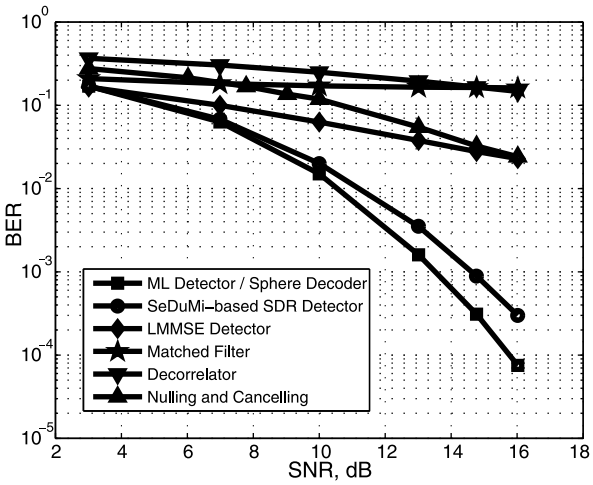
where function $\gamma(\rho)$ varies for different detectors. For example, $\gamma_{\mathrm{ml}}(\rho) = \mathcal{O}(\rho)$ for ML Detector, and $\gamma_{\mathrm{lmmse}}(\rho) = \mathcal{O}(\sqrt{\rho})$ for LMMSE Detector [5]. When a suboptimal detector is deployed instead of ML Detector, the incurred BER deterioration can be expressed in terms of the log-likelihood ratio:

$$\frac{\log(P_e(\mathrm{sdr}))}{\log(P_e(\mathrm{ml}))} = \frac{\gamma_{\mathrm{sdr}}(\rho)}{\gamma_{\mathrm{ml}}(\rho)} \leq c(\rho) \,.$$

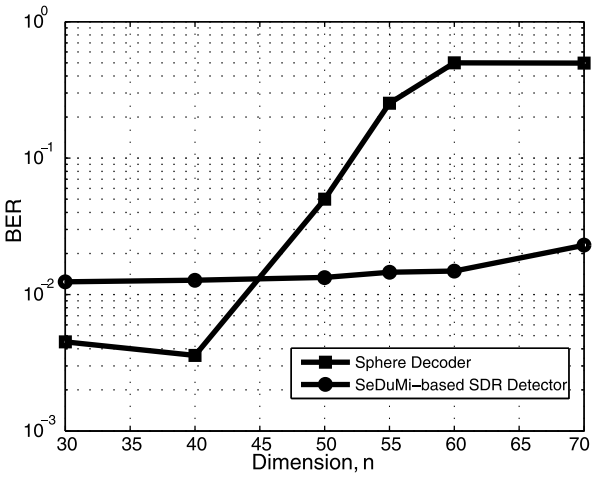
Therefore, the approximation ratio $c(\rho)$ is an essential step in bounding the SNR gap between two detectors. Before we proceed with the probabilistic analysis of the performance, let us consider the empirical BER performance of SDR Detector in numerical simulations for channel model (1).

Bit-Error-Rate Performance

The detector based on a semidefinite relaxation (SDR) consists of two parts: a solver of relaxation (7) and a randomized rounding procedure. The SDP in (7) can



Maximum Likelihood Detection via Semidefinite Programming, Figure 1
Bit-Error-Rate as a function of Signal-to-Noise Ratio for different detectors



Maximum Likelihood Detection via Semidefinite Programming, Figure 2
BER degradation due to the limit on detection time. Simulation parameters: BPSK modulation, SNR = 10 dB and time limit per bit = 6.3 ms

be efficiently solved using Interior Point (IP) methods with complexity $\mathcal{O}(n^{3.5})$. For this purpose we use SeDuMi optimization toolbox for Matlab. The randomized rounding procedure projects the solution of the SDP (7) onto the original discrete constraint set and will be discussed in details in the next section.

Figure 1 shows a comparison of the BER performance of the SeDuMi-based SDR Detector [13], LMMSE Detector, Matched Filter, Decorrelator, Nulling and Cancelling strategy, Sphere Decoder, and ML Detector. We observe a significant BER improvement of SDR Detector compared to other polynomial complexity detectors. Sphere Decoder with adjustable radius search [16] delivers the BER performance of ML Detector (with probability 1) with running time that scales exponentially [9] with problem size.

In many real-time/embedded applications a detection latency is upper bounded and, in general, premature decisions cause significant BER degradation. For simulation purposes we suppose that an engineering system is designed with BPSK modulation, operates at SNR = 10 dB and allows 6.3 ms per bit detection latency. Figure 2 demonstrates the BER performance of this system under the upper bound on the detection latency. The exponential complexity of Sphere Decoder reveals itself between dimensions 40 and 60 where we observe a rapid BER degradation because the running time of Sphere Decoder exceeds the fixed detection time threshold for most channel realizations. At the same time, the running time of SDR Detector scales gracefully with problem size and, in most cases, the detector completes detection in time. As a result, SDR Detector does not suffer any significant BER degradation even for large problem sizes. In fact, the number of late detections for SDR Detector does not exceed 1% for all dimensions shown in Fig. 2. For different values of SNR and latency per bit we obtain essentially similar curves for both detectors. Such behavior is indicative of the exponentially growing computational effort of Sphere Decoder and comparably modest computational power required by SDR Detector.

In the next section we will discuss the details of the SDP relaxation (11) and the randomized rounding procedure. After that we present theoretical guarantees that substantiate the observed empirical behavior of SDR Detector.

Method

SDR Detector consists of two components: an SDP solver and a randomized rounding procedure.

SDP Solver

A transformation of the original ML detection problem (2) into the standard SDP form (5) will help

us localize the place in (2) that makes the problem NP-hard. We start with homogenizing the objective function:

$$\begin{aligned}&\|\mathbf{y}-\sqrt{\rho/n}\,\mathbf{H}\mathbf{s}\|^2\\&=[\mathbf{s}\;\;1]^T\begin{bmatrix}(\rho/n)\,\mathbf{H}^T\mathbf{H} & -\sqrt{\rho/n}\,\mathbf{H}^T\mathbf{y}\\ -\sqrt{\rho/n}\,\mathbf{y}^T\mathbf{H} & \|\mathbf{y}\|^2\end{bmatrix}\begin{bmatrix}\mathbf{s}\\1\end{bmatrix}\\&=\rho\,\mathrm{Tr}(\mathbf{Q}\mathbf{x}\mathbf{x}^T)\,,\end{aligned}$$

where matrix $\mathbf{Q}\in\mathbb{R}^{(n+1)\times(n+1)}$ and vector $\mathbf{x}\in\mathbb{R}^{n+1}$ are defined as

$$\mathbf{Q}=\begin{bmatrix}(1/n)\,\mathbf{H}^T\mathbf{H} & -\sqrt{1/n\rho}\,\mathbf{H}^T\mathbf{y}\\ -\sqrt{1/n\rho}\,\mathbf{y}^T\mathbf{H} & \|\mathbf{y}\|^2/\rho\end{bmatrix},\mathbf{x}=\begin{bmatrix}\mathbf{s}\\1\end{bmatrix}\tag{8}$$

Notice, that matrix $\mathbf{Q}$ is composed of the parameters that are known at the receiver. We linearize the objective function by introducing a variable matrix $\mathbf{X}$ to comply with the standard SDP form (5):

$$\begin{aligned}f_{\mathrm{ML}}:=\min\quad&\mathrm{Tr}(\mathbf{Q}\mathbf{X})\\\text{s.t.}\quad&\mathbf{X}=\mathbf{x}\mathbf{x}^T\\&X_{i,i}=1,\quad i=1,\ldots,n+1\,.\end{aligned}\tag{9}$$

In this problem formulation we discarded constraint $x_{n+1}=1$ on the last entry of vector $\mathbf{x}$ because the problem is not sensitive to the sign of vector $\mathbf{x}$. If $\hat{x}_{n+1}=-1$ we output $-\hat{\mathbf{x}}$ as the solution to (9). Constraint $\mathbf{X}=\mathbf{x}\mathbf{x}^T$ is equivalent to the set $\{\mathbf{X}\succeq 0,\ \mathrm{rank}(\mathbf{X})=1\}$, where notation $\mathbf{X}\succeq 0$ implies that matrix $\mathbf{X}$ is symmetric positive semidefinite. Thus, we complete the transformation of the original ML Detection problem over BPSK constellation to the equivalent form stated in (6):

$$\begin{aligned}f_{\mathrm{ML}}:=\min\quad&\mathrm{Tr}(\mathbf{Q}\mathbf{X})\\\text{s.t.}\quad&X_{i,i}=1,\quad i=1,\ldots,n+1\\&\mathbf{X}\succeq 0\\&\mathbf{X}\text{ is rank-1}\,.\end{aligned}\tag{10}$$

The rank-1 constraint is the only non-convex constraint in (10) which makes the above problem intractable. SDR Detector relaxes the rank constraint and solves the following convex optimization problem:

$$\begin{aligned}f_{\mathrm{SDP}}:=\min\quad&\mathrm{Tr}(\mathbf{Q}\mathbf{X})\\\text{s.t.}\quad&X_{i,i}=1,\quad i=1,\ldots,n+1\\&\mathbf{X}\succeq 0\,.\end{aligned}\tag{11}$$

To reveal the difference between this relaxation and the one in (3) we can take one step further by relaxing the set of constraints $\{X_{i,i}=1,\ i=1,\ldots,n+1\}$ into $\{\mathrm{Tr}(\mathbf{X})=n+1\}$ while keeping constraint $\mathbf{X}\succeq 0$ intact. This extra relaxed problem can be solved analytically and leads to the solution

$$\hat{\mathbf{s}}=\frac{\rho}{n}\left(\frac{\rho}{n}\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{y}\,,$$

which is exactly the soft output of Decorrelator (4) with $\gamma=0$. The relaxation in (11) compares favorably to the relaxations in (3) because it requires less modifications of the ML problem, although complexity $\mathcal{O}(n^{3.5})$ of (11) is higher than $\mathcal{O}(n^3)$ for the detectors in (3).

Since we dropped the rank constraint in (11), a solution $\mathbf{X}_{\mathrm{opt}}$ of (11) is no longer rank-1, hence, we need to project $\mathbf{X}_{\mathrm{opt}}$ onto the feasible set of the original ML detection problem. Such projection is usually done by a rounding procedure which can be either deterministic like in (4) or randomized [13]. It can also vary depending on the processing power available for the algorithm. In the next section we will consider a randomized rounding procedure based on the principal eigenvector of matrix $\mathbf{X}_{\mathrm{opt}}$.

Randomized Rounding Procedure

There are various rounding procedures that can be used to extract a rank-1 approximation of $\mathbf{X}_{\mathrm{opt}}$. Widely used approaches and their analysis can be found in [4,13,14]. For our purposes we consider the randomized strategy based on the principal eigenvector of matrix $\mathbf{X}_{\mathrm{opt}}$. Notice that in the noise-free case, we have $\mathbf{v}=\mathbf{0}$ and a transmitted vector $\mathbf{s}$ belongs to the kernel of matrix $\mathbf{Q}$ which is defined in (8). The optimal objective function is 0 and is achieved by the vector of transmitted signals $\mathbf{s}$. Thus, in the noise-free case, the optimal solution of problem (11) is a rank-1 matrix:

$$\mathbf{X}_{\mathrm{opt}}=\begin{bmatrix}\mathbf{s}\\1\end{bmatrix}\begin{bmatrix}\mathbf{s}^T & 1\end{bmatrix}\,.$$

The structure of the optimal matrix $\mathbf{X}_{\mathrm{opt}}$ in the noise-free case suggests that the principal component of the eigen-decomposition contains most reliable information on the transmitted signals in high SNR region. It turns out that the optimal matrix $\mathbf{X}_{\mathrm{opt}}$ has a strong

principal component even in low SNR region, justifying the randomized rounding procedure presented below:

- *INPUT*: Solution $\mathbf{X}_{\text{opt}}$ of (11), and number D of randomized rounding tries.
- *OUTPUT*: Quasi-ML estimate $\mathbf{s}_{\text{SDR}}$ and the best achieved objective value f_{SDR}.
- *RANDOMIZED ROUNDING PROCEDURE*:

1. Take a spectral decomposition $\mathbf{X}_{\text{opt}} = \sum_{i=1}^{n+1} \lambda_i \mathbf{u}_i \mathbf{u}_i^T$ and set $\mathbf{v}_i = \sqrt{\lambda_i}\mathbf{u}_i$, $i = 1, \ldots, n+1$.
2. Pick $\mathbf{v}_i$ corresponding to the principal eigenvector $\mathbf{v}^{max} = \arg\max_{1 \le i \le n+1} \{\|\mathbf{v}_i\|\}$.
3. For each entry x_i define Bernoulli distribution:

$$\begin{aligned} \Pr\{x_i = +1\} &= (1 + v_i^{max})/2, \\ \Pr\{x_i = -1\} &= (1 - v_i^{max})/2 \,, \end{aligned} \tag{12}$$

   where v_i^{max} denotes the ith entry of vector $\mathbf{v}^{max}$.
4. Generate a fixed number D of i.i.d. $(n+1)$-dimensional vector samples $\bar{\mathbf{x}}_d$, $d = 1, \ldots, D$, such that each entry of $(\bar{\mathbf{x}}_d)_i$, $i = 1, \ldots, n+1$, is drawn from distribution (12).
5. For all D samples, set $\bar{\mathbf{x}}_d := -\bar{\mathbf{x}}_d$ if $(n+1)$-st entry of $\bar{\mathbf{x}}_d$ is equal to -1.
6. Pick $\mathbf{x}_{\text{SDR}} := \arg\min_d \bar{\mathbf{x}}_d^T \mathbf{Q} \bar{\mathbf{x}}_d$ and set the best achieved objective value $f_{\text{SDR}} := \mathbf{x}_{\text{SDR}}^T \mathbf{Q} \mathbf{x}_{\text{SDR}}$.
7. Return f_{SDR} and $\mathbf{s}_{\text{SDR}}$ which is given by vector $\mathbf{x}_{\text{SDR}}$ with the last bit discarded.

This randomized rounding procedure is designed to ensure that output $\mathbf{s}_{\text{SDR}}$ is equal to the vector of transmitted signals with high probability. Whenever there is an error, the procedure selects $\mathbf{s}_{\text{SDR}}$ to reduce the number of bits in error.

Cases

Performance of SDR Detector

Constant Factor Optimality of SDR Detector The core component of SDR Detector is an approximation algorithm based on the convex relaxation (11) of the original ML detection problem. In this section we analyze the approximation ratio of this algorithm.

A technique pioneered in [4] is widely used in optimization literature to derive a constant factor optimality for SDP-based relaxations. After the optimal solution $\mathbf{X}_{\text{opt}}$ of problem (11) has been obtained the randomized rounding procedure used in [4] defines Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{X}_{\text{opt}})$ (compare with (12)) and implements the n-dimensional sign$(\cdot)$ operator with uniformly generated cutting hyperplanes:

- Generate D i.i.d. samples $\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_D$ from Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{X}_{\text{opt}})$.
- Let $\mathbf{x}_i = \text{sign}(\bar{\mathbf{x}}_i)$ and set the solution $\mathbf{x}_{\text{SDR}}$ that achieves minimum:

$$f_{\text{SDR}} := \mathbf{x}_{\text{SDR}}^T \mathbf{Q} \mathbf{x}_{\text{SDR}} = \min_i \mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i \,.$$

The best objective value f_{SDR} achieved with this randomized rounding procedure can be upper bounded as follows [4]:

$$\begin{aligned} E\{f_{\text{SDR}}\} &= E\left\{\mathbf{x}_{\text{SDR}}^T \mathbf{Q} \mathbf{x}_{\text{SDR}}\right\} \\ &\le^P E\left\{\mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i\right\} \\ &= \text{Tr}\left(\mathbf{Q} E\left\{\mathbf{x}_i \mathbf{x}_i^T\right\}\right) \\ &= \frac{2}{\pi} \text{Tr}\left(\mathbf{Q} \arcsin(\mathbf{X}_{\text{opt}})\right) \,, \end{aligned} \tag{13}$$

where the inequality above holds in probability for sufficiently many samples D, and the last equality follows from that fact that for any scalar random samples $\bar{x}_i$ and $\bar{x}_j$ drawn from $\mathcal{N}(0, 1)$ we have:

$$E\{\text{sign}(\bar{x}_i)\,\text{sign}(\bar{x}_i)\} = \frac{2}{\pi} \arcsin\left(E\left\{\bar{x}_i \bar{x}_j\right\}\right) \,.$$

By taking Taylor expansion of $\arcsin(\mathbf{Y})$, we can see that for any matrix $\mathbf{Y}$, such that $\mathbf{Y} \succeq 0, Y_{ii} = 1$ the following inequality holds:

$$\arcsin(\mathbf{Y}) \succeq \mathbf{Y} \,. \tag{14}$$

Suppose that $\mathbf{Q} \preceq 0$, then we have the following upper bound:

$$\text{Tr}\left(\mathbf{Q} \arcsin(\mathbf{X}_{\text{opt}})\right) \le \text{Tr}(\mathbf{Q}\mathbf{X}_{\text{opt}}) \,, \tag{15}$$

which allows us to bound f_{SDR} as a constant factor away from f_{ML}:

$$\begin{aligned} f_{\text{ML}} \le E\{f_{\text{SDR}}\} &\le^P \frac{2}{\pi} \text{Tr}(\mathbf{Q}\mathbf{X}_{\text{opt}}) \\ &= \frac{2}{\pi} f_{\text{SDP}} \le \frac{2}{\pi} f_{\text{ML}} \,, \end{aligned}$$

where the first inequality holds because an output of SDR Detector belongs to the feasible set of the ML problem (10), the second inequality follows from (13)

combined with (15), the third equality is the definition of f_{SDP}, and the last inequality holds because the SDP problem (11) is a relaxation of the ML problem (10). Therefore, given $\mathbf{Q} \preceq 0$, we obtain a $2/\pi$-approximation ratio for the algorithm. Unfortunately, for ML detection problem the reverse inequality takes place (8):

$$\mathbf{Q} = \begin{bmatrix} (1/n)\,\mathbf{H}^T\mathbf{H} & -\sqrt{1/n\rho}\,\mathbf{H}^T\mathbf{y} \\ -\sqrt{1/n\rho}\,\mathbf{y}^T\mathbf{H} & \|\mathbf{y}\|^2/\rho \end{bmatrix} \succeq 0\,.$$

We can attempt to cure the problem with inequality similar to (14) in the reverse direction for some constant c:

$$\arcsin(\mathbf{Y}) \preceq c\mathbf{Y}, \text{ for all } \mathbf{Y} \succeq 0, \text{ with } Y_{ii} = 1\,.$$

For this inequality to hold, c must be growing linearly with problem dimension n. Hence, in the limit $n \to \infty$ the constant c together with the approximation ratio of the algorithm grow unbounded. That is, we can not obtain a constant factor approximation by applying the standard technique of [4] to the analysis of the SDP relaxation in (11).

The technique presented above applies to any negative semidefinite matrix $\mathbf{Q}$, hence, in the context of suboptimal detection it attempts to obtain a constant factor optimality for the worst-case channel realization. However, from the perspective of digital communications, we are interested in the average performance of SDR Detector over many channel realizations. Unlike the technique we have discussed above, a probabilistic analysis of Karush–Kuhn–Tucker (KKT) optimality conditions of the semidefinite problem (11) allows us to claim a constant factor optimality for SDR Detector in probability [11].

The optimal objective value f_{SDR} achieved by SDR Detector is within a constant factor $c(\rho,\gamma)$ away from the optimal ML objective value in probability:

$$\lim_{\substack{n,m\to\infty \\ m/n\to\gamma\geq 1}} P\left\{\frac{f_{\text{SDR}}}{f_{\text{ML}}} \leq c(\rho,\gamma)\right\} = 1, \qquad (16)$$

$$\text{where } c(\rho,\gamma) = 1 + \frac{2(1+\sqrt{\gamma})^2\beta}{\gamma\rho^{\alpha}-1}\,,$$

and $\{\alpha,\beta\}$ are given by

$$\alpha = \begin{cases} \frac{1}{3}, & \text{if } \gamma = 1 \\ \frac{1}{2}, & \text{if } \gamma > 1 \end{cases} \qquad \beta = \begin{cases} 4\sqrt[3]{4}, & \text{if } \gamma = 1 \\ 4\sqrt{\frac{\gamma}{\gamma-1}}, & \text{if } \gamma > 1 \end{cases}$$

The statement implies that the log-likelihood ratio of SDR and ML Detectors is bounded in probability by a constant which is fully specified by SNR only.

Performance of SDR Detector in High SNR Region We have argued in Sect. "Randomized Rounding Procedure" that the selected randomized rounding procedure provides the optimal solution in the noise-free case. The optimality condition can be extended to the case of large finite SNR: for sufficiently high SNR SDR Detector solves ML detection problem in polynomial time.

For given system dimension n and SNR ρ (both finite), the solution $\mathbf{X}_{\text{opt}}$ of the relaxed problem (11) is rank-1 if channel matrix $\mathbf{H}$ and noise $\mathbf{v}$ realizations satisfy:

$$\lambda_{min}(\mathbf{H}^T\mathbf{H}) > \sqrt{\frac{n}{\rho}}\,\|\mathbf{H}^T\mathbf{v}\|_1\,. \qquad (18)$$

Since random matrix $\mathbf{H}^T\mathbf{H}$ is full rank with probability 1, this claim can also be interpreted as follows: for any given n there exists a sufficiently high (finite) SNR level such that (18) holds and $\mathbf{X}_{\text{opt}}$ is rank-1. In general, if (18) does not hold $\mathbf{X}_{\text{opt}}$ may still be rank-1. Notice that if condition (18) is satisfied the solution of the SDP problem (11) belongs to the feasible set of (10), thus, $\mathbf{X}_{\text{opt}}$ is also the solution of the ML detection problem. Hence, under the specified conditions SDR Detector solves the original ML detection problem.

The asymptotic performance of SDR Detector for fixed problem size and $\rho \to \infty$ has been analyzed in [8], where it is shown that for Rayleigh fading $\mathbf{H}$ SDR Detector achieves maximum diversity, i. e.

$$\lim_{\rho\to\infty} \frac{\log P\{\mathbf{s}_{\text{sdr}} \neq \mathbf{s}\}}{\log\rho} = \lim_{\rho\to\infty} \frac{\log P\{\mathbf{s}_{\text{ml}} \neq \mathbf{s}\}}{\log\rho} = -\frac{n}{2}\,.$$

Simulation Results

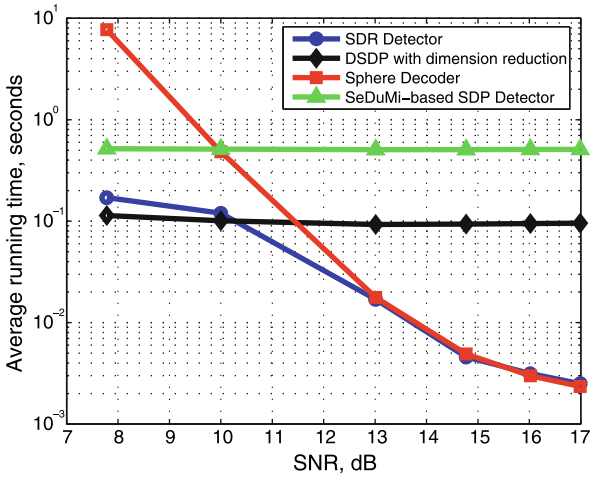
In this section we compare the running time and the BER performance of various implementations of the detectors based on the semidefinite relaxation (11) and that of Sphere Decoder:

- SDP detector [13] implemented with SeDuMi toolbox [15] for convex optimization problems.
- SDR Detector that is based on a dual-scaling interior-point method (DSDP implementation [1]) and a dimension reduction strategy [12].

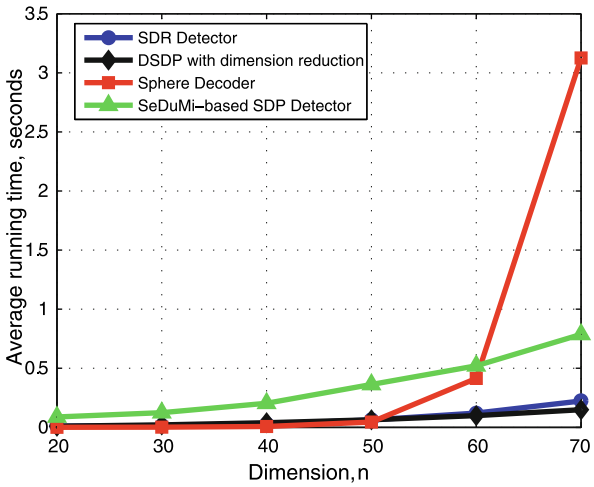
- SDR Detector [12], implemented with a dual-scaling interior-point method, a dimension reduction strategy, and warm start with a truncated version of Sphere Decoder.
- Sphere Decoder [16].

Figures 3 and 4 demonstrate the average running time and the BER performance achieved by the above detectors for problem size $n = 60$. Notice, the running time of DSDP-based (SeDuMi-based) detector is insensitive to SNR, and the BER performance shows 1 dB (2-dB) SNR loss. Sphere Decoder is faster than the semidefinite relaxation-based detectors in high SNR regime but becomes significantly slower for SNR lower than 10 dB. SDR Detector matches the speed of Sphere Decoder in high SNR region, matches the running time of other semidefinite relaxation-based detectors in low SNR regime, and enjoys the near-ML BER performance.

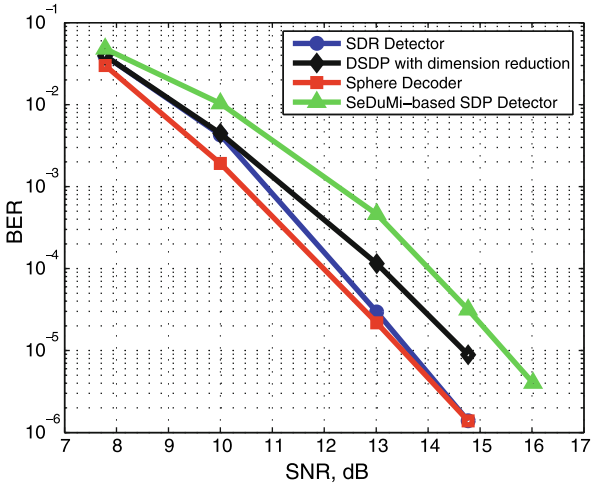
Figures 5 and 6 compare the average running time for large problems and in low SNR region. The running time of polynomial complexity detectors (SDR



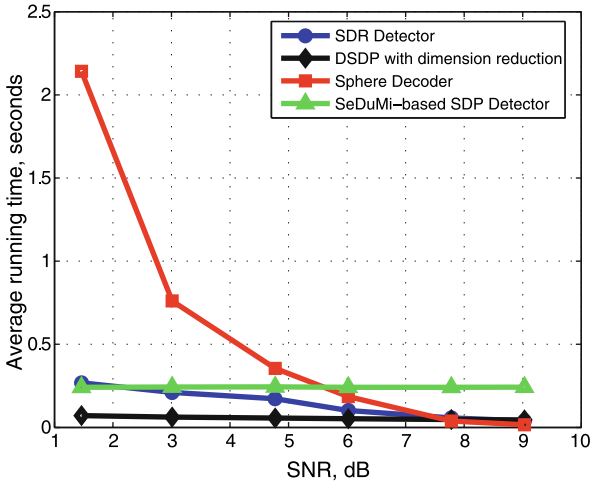
Maximum Likelihood Detection via Semidefinite Programming, Figure 3
Running time comparison, $n = 60$



Maximum Likelihood Detection via Semidefinite Programming, Figure 5
Running time for large problems, $\rho = 10$ dB



Maximum Likelihood Detection via Semidefinite Programming, Figure 4
Bit-error-rate comparison, $n = 60$



Maximum Likelihood Detection via Semidefinite Programming, Figure 6
Running time in low SNR regime, $n = 40$

Detector, SeDuMi and DSDP-based) scales well in both regimes, remaining in the sub-second region, while the running time of Sphere Decoder deteriorates in both scenarios.

Conclusions

We have considered the maximum likelihood detection problem. Among various quasi-ML detectors SDR Detector offers a near-optimal BER performance with the worst-case polynomial complexity. We have analyzed the underlying structure of the SDP relaxation which is the core of SDR Detector. For a given SNR SDR Detector delivers a constant factor approximation of the log-likelihood ratio for the original ML detection problem in probability, where the constant factor is independent of problem size. SDR Detector solves ML detection problem exactly in high SNR region. Numerical simulations of BER and running time empirically demonstrate the advantages of SDR Detector as compared to the computationally expensive ML Detector.

References

1. Benson SJ, Ye Y (2007) DSDP5: Software for semidefinite programming. ACM Trans Math Soft 34(3)
2. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, New York
3. Fincke U, Pohst M (1985) Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Math Comput 44:463–471
4. Goemans MX, Williamson DP (1995) Improved approximation algorithms for maximum cut and satisfiability problem using semidefinite programming. J ACM 42:1115–1145
5. Guo D, Verdú S (2005) Randomly spread CDMA: asymptotics via statistical physics. IEEE Trans Inf Theory 51:1982–2010
6. Hassibi B, Hochwald BM (2002) High-rate codes that are linear in space and time. IEEE Trans Inf Theory 48(7): 1804–1824
7. Hassibi B, Vikalo H (2001) On the expected complexity of sphere decoding. Thirty-Fifth Asilomar Conference on Signals. Syst Comput 2:1051–1055
8. Jalden J (2006) Detection for multiple input multiple output channels. Ph.D. Thesis, KTH, School of Electrical Engineering, Stockholm
9. Jalden J, Ottersten B (2004) An exponential lower bound on the expected complexity of sphere decoding. Proc ICASSP '04, vol 4, pp IV 393–IV 396
10. Jalden J, Ottersten B (2005) On the complexity of sphere decoding in digital communications. IEEE Trans Signal Process 53(4):1474–1484
11. Kisialiou M, Luo Z-Q (2005) Performance analysis of quasi-maximum-likelihood detector based on semi-definite programming. Proc ICASSP '05, vol 3, pp III 433–III 436
12. Kisialiou M, Luo Z-Q (2007) Efficient implementation of a quasi-maximum-likelihood detector based on semidefinite relaxation. Proc ICASSP '07, vol 4, pp IV 1329–IV 1332
13. Ma WK, Davidson TN, Wong KM, Luo Z-Q, Ching PC (2002) Quasi-maximum-likelihood multiuser detection using semidefinite relaxation. IEEE Trans Signal Process 50(4):912–922
14. Nesterov YE (1997) Quality of semi-difinite relaxation for nonconvex quadratic optimization. CORE Discussion Paper, no 9719
15. Sturm JF (1999) Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim Meth Soft 11–12:625–653
16. Viterbo E, Boutros J (1999) A universal lattice code decoder for fading channels. IEEE Trans Inf Theory 45(5): 1639–1642

Maximum Partition Matching

MPM

Jianer Chen
Texas A&M University, College Station, USA

MSC2000: 05A18, 05D15, 68M07, 68M10, 68Q25, 68R05

Article Outline

Keywords
Definitions and Motivation
Case I. Via Pre-Matching when $\|\mathbf{S}\|$ is Large
Case II. Via Greedy Method when $\|\mathbf{S}\|$ is Small
See also
References

Keywords

Maximum matching; Greedy algorithm; Star network; Parallel routing algorithm

The maximum partition matching problem was introduced recently in the study of routing schemes on interconnection networks [2]. In this article, we study the basic properties of the problem. An efficient algorithm for the maximum partition matching problem is presented.

Definitions and Motivation

Let $\mathbf{S} = \{C_1, \ldots, C_k\}$ be a collection of subsets of the universal set $U = \{1, \ldots, n\}$ such that $\cup_{i=1}^{k} C_i = U$, and $C_i \cap C_j = \emptyset$ for all $i \neq j$. A partition $(\mathbf{A}, \mathbf{B})$ of $\mathbf{S}$ pairs two elements a and b in U if a is contained in a subset in $\mathbf{A}$ and b is contained in a subset in $\mathbf{B}$. A *partition matching* (of order m) of $\mathbf{S}$ consists of two ordered subsets $L = \{a_1, \ldots, a_m\}$ and $R = \{b_1, \ldots, b_m\}$ of m elements of U (the subsets L and R may not be disjoint), together with a sequence of m distinct partitions of $\mathbf{S}$: $(\mathbf{A}_1, \mathbf{B}_1), \ldots, (\mathbf{A}_m, \mathbf{B}_m)$ such that for all $i = 1, \ldots, m$, the partition $(\mathbf{A}_i, \mathbf{B}_i)$ pairs the elements a_i and b_i. The *maximum partition matching problem* is to construct a partition matching of order m for a given collection $\mathbf{S}$ with m maximized.

The maximum partition matching problem arises in connection with the parallel routing problem in interconnection networks. In particular, in the study of the star networks [1], which are attractive alternatives to the popular hypercubes networks. It can be shown that constructing an optimal parallel routing scheme in the star networks can be effectively reduced to the maximum partition matching problem. Readers interested in this connection are referred to [2] for a detailed discussion.

The maximum partition matching problem can be formulated in terms of the *3-dimensional matching problem* as follows: given an instance $\mathbf{S} = \{C_1, \ldots, C_k\}$ of the maximum partition matching problem, we construct an instance M for the 3-dimensional matching problem such that a triple (a, b, P) is contained in M if and only if the partition P of $\mathbf{S}$ pairs the elements a and b. However, since the number of partitions of the collection $\mathbf{S}$ can be as large as 2^n and the 3-dimensional matching problem is *NP*-hard [4], this reduction does not hint a polynomial time algorithm for the maximum partition matching problem.

In the rest of this article, we study the basic properties for the maximum partition matching problem, and present an algorithm of running time $O(n^2 \log n)$ for the problem. We first introduce necessary terminologies that will be used in our discussion.

Let $\pi = \langle L, R, (\mathbf{A}_1, \mathbf{B}_1), \ldots, (\mathbf{A}_m, \mathbf{B}_m)\rangle$ be a partition matching of the collection $\mathbf{S}$, where $L = \{a_1, \ldots, a_m\}$ and $R = \{b_1, \ldots, b_m\}$. We will say that the partition $(\mathbf{A}_i, \mathbf{B}_i)$ *left-pairs* the element a_i and *right-pairs* the element b_i. An element a is said to be *left-paired* if it is in the set L. Otherwise, the element a is *left-unpaired*. Similarly we define *right-paired* and *right-unpaired* elements. The collections $\mathbf{A}_i$ and $\mathbf{B}_i$ are called the *left-collection* and *right-collection* of the partition $(\mathbf{A}_i, \mathbf{B}_i)$. The partition matching π may also be written as $\pi[(a_1, b_1), \ldots, (a_m, b_m)]$ if the corresponding partitions are implied.

For the rest of this paper, we assume that $U = \{1, \ldots, n\}$ and that $\mathbf{S} = \{C_1, \ldots, C_k\}$ is a collection of pairwise disjoint subsets of U such that $\cup_{i=1}^{k} C_i = U$.

Case I. Via Pre-Matching when ‖S‖ is Large

A necessary condition for two ordered subsets $L = \{a_1, \ldots, a_m\}$ and $R = \{b_1, \ldots, b_m\}$ of U to form a partition matching for the collection $\mathbf{S}$ is that a_i and b_i belong to different subsets in the collection $\mathbf{S}$, for all $i = 1, \ldots, m$. We say that the two ordered subsets L and R of U form a *pre-matching* $\sigma = \{(a_i, b_i): 1 \leq i \leq m\}$ if a_i and b_i do not belong to the same subset in the collection $\mathbf{S}$, for all $i = 1, \ldots, m$. The pre-matching σ is *maximum* if m is the largest among all pre-matchings of $\mathbf{S}$.

A maximum pre-matching can be constructed efficiently by the *algorithm pre-matching* given below, where we say that a set is *singular* if it consists of a single element. See [3] for a proof for the correctness of the algorithm.

Input : the collection $\mathbf{S} = \{C_1, \ldots, C_k\}$ of subsets of U
Output : a maximum pre-matching σ in $\mathbf{S}$
1. $\mathbf{T} = \mathbf{S}$; $\sigma = \emptyset$;
2. WHILE $\mathbf{T}$ contains more than one set but does not consist of exactly three singular sets
DO
  2.1. pick two sets C and C' of largest cardinality in $\mathbf{T}$;
  2.2. pick an element a in C and an element b in C';
  2.3. $\sigma = \sigma \cup \{(a, b), (b, a)\}$;
  2.4. $C = C - \{a\}$; $C' = C' - \{b\}$;
  2.5. if C or C' is empty now, delete it from $\mathbf{T}$;
3. IF $\mathbf{T}$ consists of exactly three singular sets $C_1 = \{a_1\}$, $C_2 = \{a_2\}$, and $C_3 = \{a_3\}$
THEN
$\sigma = \sigma \cup \{(a_1, a_2), (a_2, a_3), (a_3, a_1)\}$.

Algorithm pre-matching

In the following, we show that when the cardinality of the collection **S** is large enough, a maximum partition matching of **S** can be constructed from the maximum pre-matching σ produced by the algorithm pre-matching.

Suppose that the collection **S** consists of k subsets $C_1, \ldots, C_k$ and $2^k \geq 4n$. The pre-matching σ contains at most n pairs. Let (a, b) be a pair in σ and let C and C' be two arbitrary subsets in **S** such that C contains a and C' contains b. Note that the number of partitions $(\mathbf{A}, \mathbf{B})$ of **S** such that C is in **A** and C' is in **B** is equal to $2^{k-2} \geq n$. Therefore, at least one such partition can be used to left-pair a and right-pair b. This observation results in the following theorem.

Theorem 1 *Let $\mathbf{S} = \{C_1, \ldots, C_k\}$ be a collection of nonempty subsets of the universal set $U = \{1, \ldots, n\}$ such that $\cup_{i=1}^{k} C_i = U$ and $C_i \cap C_j = \emptyset$, for $i \neq j$. If $2^k \geq 4n$, then a maximum partition matching in **S** can be constructed in time $O(n^2)$.*

Proof Consider the following *algorithm partition-matching-I*.

Input:	the collection $\mathbf{S} = \{C_1, \ldots, C_k\}$ of subsets of U
Output:	a partition matching π in **S**
1.	construct a maximum pre-matching σ of **S**;
2.	FOR each pair (a, b) in σ DO use an unused partition of **S** to pair a and b.

Algorithm partition-matching-I

Suppose the pre-matching σ constructed in step 1 is $\sigma = \{(a_1, b_1), \ldots, (a_m, b_m)\}$. According to the above discussion, for each pair (a_i, b_i) in σ, there is always an unused partition of **S** that left-pairs a and right-pairs b. Therefore, step 2 of the algorithm partition-matching-I is valid and constructs a partition matching π for the collection **S**. Since each partition matching for **S** induces a pre-matching in **S** and σ is a maximum pre-matching, we conclude that the partition matching π is a maximum partition matching for the collection **S**.

By carefully organizing the elements in U and the partitions of **S**, we can show that the algorithm partition-matching-I runs in time $O(n^2)$. See [3].

Case II. Via Greedy Method when ‖S‖ is Small

Now we consider the case $2^k < 4n$. Since the number 2^k of partitions of the collection **S** is small, we can apply a greedy strategy that expands a current partition matching by trying to add each of the unused partitions to the partition matching. We show in this section that a careful use of this greedy method constructs a maximum partition matching for the given collection.

Suppose we have a partition matching $\pi = \pi[(a_1, b_1), \ldots, (a_h, b_h)]$ and want to expand it. The partitions of the collection **S** then can be classified into two classes: h of the partitions are used to pair the h pairs (a_i, b_i), $i = 1, \ldots, h$, and the rest $2^k - h$ partitions are unused. Now if there is an unused partition $P = (\mathbf{A}, \mathbf{B})$ such that there is a left-unpaired element a in **A** and a right-unpaired element b in **B**, then we simply pair the element a with the element b using the partition P, thus expanding the partition matching π.

Now suppose that there is no such unused partition, i. e., for all unused partitions $(\mathbf{A}, \mathbf{B})$, either **A** contains no left-unpaired elements or **B** contains no right-unpaired elements. This case may not necessarily imply that the current partition matching is the maximum. For example, suppose that $(\mathbf{A}, \mathbf{B})$ is an unused partition such that there is a left-unpaired element a in **A** but no right-unpaired elements in **B**. Assume further that there is a used partition $(\mathbf{A}', \mathbf{B}')$ that pairs elements (a', b'), such that the element b' is in **B** and there is a right-unpaired element b in $\mathbf{B}'$. Then we can let the partition $(\mathbf{A}', \mathbf{B}')$ pair the elements (a', b), and then let the partition $(\mathbf{A}, \mathbf{B})$ pair the elements (a, b'), thus expanding the partition matching π. An explanation of this process is that the used partitions have been incorrectly used to pair elements, thus in order to construct a maximum partition matching, we must re-pair some of the elements. To further investigate this relation, we need to introduce a few notations.

For a used partition P of **S**, we put an underline on a set in the left-collection (resp. the right-collection) of P to indicate that an element in the set is left-paired (resp. right-paired) by the partition P. The sets will be called the *left-paired set* and the *right-paired set* of the partition P, respectively.

Definition 2 A used partition P is *directly left-reachable* from a partition $P_1 = (\mathbf{A}_1, \mathbf{B}_1)$ if the left-paired set of P is contained in $\mathbf{A}_1$ (the partition P_1

can be either used or unused). The partition P is *directly right-reachable* from a partition $P_2 = (\mathbf{A}_2, \mathbf{B}_2)$ if the right-paired set of P is contained in $\mathbf{B}_2$. A partition P_s is *left-reachable* (resp. *right-reachable*) from a partition P_1 if there are partitions $P_2, \ldots, P_{s-1}$ such that P_i is directly left-reachable (resp. directly right-reachable) from P_{i-1}, for all $i = 2, \ldots, s$.

The left-reachability and the right-reachability are transitive relations.

Let $P_1 = (\mathbf{A}_1, \mathbf{B}_1)$ be an unused partition such that there are no left-unpaired elements in $\mathbf{A}_1$, and let $P_s = (\mathbf{A}_s, \mathbf{B}_s)$ be a partition left-reachable from P_1 and there is a left-unpaired element a_s in $\mathbf{A}_s$. We show how we can use a *chain justification* to make a left-unpaired element for the collection $\mathbf{A}_1$.

By the definition, there are used partitions $P_2, \ldots, P_{s-1}$ such that P_i is directly left-reachable from P_{i-1}, for $i = 2, \ldots, s$. We can further assume that P_i is not directly left-reachable from P_{i-2} for $i = 3, \ldots, s$ (otherwise we simply delete the partition P_{i-1} from the sequence). Thus, these partitions can be written as

$$
\begin{aligned}
&P_1 = (\{C_1\} \cup \mathbf{A}_1', \mathbf{B}_1),\\
&P_2 = (\{\underline{C_1}, C_2\} \cup \mathbf{A}_2', \mathbf{B}_2),\\
&P_3 = (\{\underline{C_2}, C_3\} \cup \mathbf{A}_3', \mathbf{B}_3),\\
&\ddots\\
&P_{s-1} = (\{\underline{C_{s-2}}, C_{s-1}\} \cup \mathbf{A}_{s-1}', \mathbf{B}_{s-1}),\\
&P_s = (\{\underline{C_{s-1}}, C_s\} \cup \mathbf{A}_s', \mathbf{B}_s),
\end{aligned}
$$

where $\mathbf{A}_1', \ldots, \mathbf{A}_s'$ are subcollections of $\mathbf{S}$ without an underlined set.

We can assume that the left-unpaired element a_s in $\mathbf{A}_s = \{\underline{C_{s-1}}, C_s\} \cup \mathbf{A}_s'$ is in a nonunderlined set C_s in $\mathbf{A}_s$ (otherwise we consider the sequence $P_1, \ldots, P_{s-1}$ instead). We modify the partition sequence into

$$
\begin{aligned}
&P_1 = (\{C_1\} \cup \mathbf{A}_1', \mathbf{B}_1),\\
&P_2 = (\{C_1, \underline{C_2}\} \cup \mathbf{A}_2', \mathbf{B}_2),\\
&P_3 = (\{C_2, \underline{C_3}\} \cup \mathbf{A}_3', \mathbf{B}_3),\\
&\vdots\\
&P_{s-1} = (\{C_{s-2}, \underline{C_{s-1}}\} \cup \mathbf{A}_{s-1}', \mathbf{B}_{s-1}),\\
&P_s = (\{C_{s-1}, \underline{C_s}\} \cup \mathbf{A}_s', \mathbf{B}_s).
\end{aligned}
$$

The interpretation is as follows: we use the partition P_s to left-pair the left-unpaired element a_s (the right-paired element in the right-collection $\mathbf{B}_s$ is unchanged). Thus, the element a_{s-1} in the set C_{s-1} of the partition P_s used to left-pair becomes left-unpaired. We then use the partition P_{s-1} to left-pair the element a_{s-1} and leave an element a_{s-2} in the set C_{s-2} left-unpaired, then we use the partition P_{s-2} to left-pair a_{s-2}, etc. At the end, we use the partition P_2 to left-pair an element a_2 in the set C_2 and leave an element a_1 in the set C_1 left-unpaired. Therefore, this process makes an element in the left-collection $\mathbf{A}_1 = \{C_1\} \cup \mathbf{A}_1'$ of the partition P_1 left-unpaired.

The above process will be called a *left-chain justification*. Thus, given an unused partition $P_1 = (\mathbf{A}_1, \mathbf{B}_1)$ in which the left-collection $\mathbf{A}_1$ has no left-unpaired elements and given a used partition $P_s = (\mathbf{A}_s, \mathbf{B}_s)$ left-reachable from P_1 such that the left-collection $\mathbf{A}_s$ of P_s has a left-unpaired element, we can apply the left-chain justification that keeps all used partitions in the partition matching π and makes a left-unpaired element for the partition P_1. A process called *right-chain justification* for right-collections of the partitions can be described similarly.

A greedy method based on the left-chain and right-chain justifications is presented in the following *algorithm greedy-expanding*.

Input:	the collection $\mathbf{S} = \{C_1, \ldots, C_k\}$ of subsets of U
Output:	a partition matching π_{exp} in $\mathbf{S}$
1.	$\pi_{\text{exp}} = \emptyset$;
2.	repeat until no more changes IF there is an unused partition $P = (\mathbf{A}, \mathbf{B})$ that has a left-unpaired element a in $\mathbf{A}$ and a right-unpaired element b in $\mathbf{B}$ THEN pair the elements (a, b) by the partition P and add P to the matching π_{exp} ELSE IF a left-chain justification or a right-chain justification (or both) is applicable to make an unused partition $P = (\mathbf{A}, \mathbf{B})$ to have a left-unpaired element in $\mathbf{A}$ and a right-unpaired element in $\mathbf{B}$ THEN apply the left-chain justification and/or the right-chain justification

Algorithm greedy-expanding

In case $2^k < 4n$, a careful organization of the elements and the partitions can make the running time

of the algorithm greedy-expanding bounded by $O(n^2 \log n)$. Briefly speaking, we construct a graph G of 2^k vertices in which each vertex represents a partition of **S**. The direct left- and right- reachabilities of partitions are given by the edges in the graph G, so that checking left- and right- reachabilities and performing left- and right- chain justifications can be done efficiently. Interested readers are referred to [3] for a detailed description.

After execution of the algorithm greedy-expanding, we obtain a partition matching π_{exp}. For each partition $P = (\mathbf{A}, \mathbf{B})$ not included in π_{exp}, either **A** has no left-unpaired elements and no used partition left-reachable from P has a left-unpaired element in its left-collection, or **B** has no right-unpaired elements and no used partition right-reachable from P has a right-unpaired element in its right-collection.

Definition 3 Define L_{free} to be the set of partitions P not used by π_{exp} such that the left-collection of P has no left-unpaired elements and no used partition left-reachable from P has a left-unpaired element in its left-collection, and define R_{free} to be the set of partitions P' not used by π_{exp} such that the right-collection of P' has no right-unpaired elements and no used partition right-reachable from P' has a right-unpaired element in its right-collection.

According to the algorithm greedy-matching, each partition not used by π_{exp} is either in the set L_{free} or in the set R_{free}. The sets L_{free} and R_{free} may not be disjoint.

Definition 4 L_{reac} to be the set of partitions in π_{exp} that are left-reachable from a partition in L_{free}, and define R_{reac} to be the set of partitions in π_{exp} that are right-reachable from a partition in R_{reac}.

According to the definitions, if a used partition P is in the set L_{reac}, then all elements in its left-collection are left-paired, and if a used partition P is in the set R_{reac}, then all elements in its right-collection are right-paired.

We first show that if L_{reac} and R_{reac} are not disjoint, then we can construct a maximum partition matching from the partition matching π_{exp} constructed by the algorithm greedy-expanding. For this, we need the following technical lemma.

Lemma 5 *If the sets L_{reac} and R_{reac} contain a common partition and the partition matching π_{exp} has less than n pairs, then there is a set C_0 in **S**, $|C_0| \le n/2$, such that either all elements in each set $C \neq C_0$ are left-paired and every used partition whose left-paired set is not C_0 is contained in L_{reac}, or all elements in each set $C \neq C_0$ are right-paired and every used partition whose right-paired set is not C_0 is contained in R_{reac}.*

For a proof, see [3].

Theorem 6 *If L_{reac} and R_{reac} have a common partition, then the collection **S** has a maximum partition matching of n pairs, which can be constructed in linear time from the partition matching π_{exp}.*

Proof If π_{exp} has n pairs, then π_{exp} is already a maximum partition matching. Thus we assume that π_{exp} has less than n pairs. According to the above lemma, we can assume, without loss of generality, that all elements in each set C_i, $i = 2, \ldots, k$, are left-paired, and that every used partition whose left-paired set is not C_1 is in L_{reac}. Moreover, $|C_1| \le \sum_{i=2}^{k} |C_i|$.

Let $t = \sum_{i=2}^{k} |C_i|$ and $d = |C_1|$. Then we can assume that the partition matching π_{exp} consists of the partitions

$$P_1, \ldots, P_t, P_{t+1}, \ldots, P_{t+h}$$

where $P_1, \ldots, P_t$ are used by π_{exp} to left-pair the elements in $\cup_{i=2}^{k} C_i$, and $P_{t+1}, \ldots, P_{t+h}$ are used by π_{exp} to left-pair the elements in C_1, $h < d$. Moreover, all partitions $P_1, \ldots, P_t$ are in the set L_{reac}. Thus, the set C_1 must be contained in the right-collection in each of the partitions $P_1, \ldots, P_t$.

We ignore the partitions $P_{t+1}, \ldots, P_{t+h}$ and use the partitions $P_1, \ldots, P_t$ to construct a maximum partition matching of n pairs. Note that $\{P_1, \ldots, P_t\}$ also forms a partition matching in the collection **S**.

For a partition $(\mathbf{A}, \mathbf{B})$ of **S**, we say that the partition $(\mathbf{B}, \mathbf{A})$ is obtained by *flipping* the partition $(\mathbf{A}, \mathbf{B})$. In the following *algorithm partition-flipping*, we show that a maximum partition matching of n pairs can be constructed by flipping d partitions in the partitions $P_1, \ldots, P_t$.

Input: a partition matching $\{P_1, \ldots, P_t\}$ that left-pairs all elements in $\cup_{i=2}^{k} C_i$, $t = \sum_{i=2}^{k} |C_i|$, and the set C_1 is contained in the right-collection of each partition P_i, $i = 1, \ldots, t$, $d = |C_1| \leq t$

Output: a maximum partition matching in **S** with n pairs.

1. if not all elements in the set C_1 are right-paired by $P_1, \ldots, P_t$, replace a proper number of right-paired elements in $\cup_{i=2}^{k} C_i$ by the right-unpaired elements in C_1 so that all elements in C_1 are right-paired by $P_1, \ldots, P_t$;
2. suppose that the partitions $P_1, \ldots, P_{t-d}$ right-pair $t - d$ elements $b_1, \ldots, b_{t-d}$ in $\cup_{i=2}^{k} C_i$, and that $P_{t-d+1}, \ldots, P_t$ right-pair the d elements in C_1;
3. suppose that $\overline{P}_1, \ldots, \overline{P}_{t-d}$ are the $t - d$ partitions in $\{P_1, \ldots, P_t\}$ that left-pair the elements $b_1, \ldots, b_{t-d}$;
4. flip each of the d partitions in $\{P_1, \ldots, P_t\} - \{\overline{P}_1, \ldots, \overline{P}_{t-d}\}$ to get d partitions $P_1', \ldots, P_d'$ to left-pair the d elements in C_1. The right paired element of each P_i' is the left-paired element before the flipping;
5. $\{P_1, \ldots, P_t, P_1', \ldots, P_d'\}$ is a partition matching of n pairs.

Algorithm partition-flipping

Step 1 of the algorithm is always possible: since C_1 is contained in the right-collection of each partition P_i, $i = 1, \ldots, t$, and $t \geq d$, for each right-unpaired element b in C_1, we can always pick a partition P_i that right-pairs an element in $\cup_{i=2}^{k} C_i$, and let P_i right-pair the element b. We keep doing this replacement until all d elements in C_1 get right-paired. At this point, the number of partitions in $\{P_1, \ldots, P_t\}$ that right-pair elements in $\cup_{i=2}^{k} C_i$ is exactly $t - d$. Step 3 is always possible since the partitions $P_1, \ldots, P_t$ left-pair all elements in $\cup_{i=2}^{k} C_i$.

Now we verify that the constructed sequence $\{P_1, \ldots, P_t, P_1', \ldots, P_d'\}$ is a partition matching in **S**. No two partitions P_i and P_j can be identical since $\{P_1, \ldots, P_t\}$ is supposed to be a partition matching in **S**. No two partitions P_i' and P_j' can be identical since they are obtained by flipping two different partitions in $\{P_1, \ldots, P_t\}$. No partition P_i is identical to a partition P_j' because P_i has C_1 in its right-collection while P_j' has C_1 in its left-collection. Therefore, the partitions $P_1, \ldots, P_t, P_1', \ldots, P_d'$ are all distinct.

Each of the partitions $P_1, \ldots, P_t$ left-pairs an element in $\cup_{i=2}^{k} C_i$, and each of the partitions $P_1', \ldots, P_d'$ left-pairs an element in C_1. Thus, all elements in the universal set U get left-paired in $\{P_1, \ldots, P_t, P_1', \ldots, P_d'\}$.

Finally, the partitions $P_1, \ldots, P_t$ right-pair all elements in C_1 and the elements $b_1, \ldots, b_{t-d}$ in $\cup_{i=2}^{k} C_i$. Now by our selection of the partitions, the partitions $P_1', \ldots, P_d'$ precisely right-pair all the elements in $\cup_{i=2}^{k} C_i - \{b_1, \ldots, b_{t-d}\}$. Thus, all elements in U also get right-paired in $\{P_1, \ldots, P_t, P_1', \ldots, P_d'\}$.

This concludes that the constructed sequence $\{P_1, \ldots, P_t, P_1', \ldots, P_d'\}$ is a maximum partition matching in the collection **S**. The running time of the algorithm partition-flipping is obviously linear.

Now we consider the case when the sets L_{reac} and R_{reac} have no common partitions.

Theorem 7 *If L_{reac} and R_{reac} have no common partitions, then the partition matching π_{exp} is a maximum partition matching.*

Proof Let W_{other} be the set of used partitions in π_{exp} that belong to neither L_{reac} nor R_{reac}. Then $L_{\text{free}} \cup R_{\text{free}} \cup L_{\text{reac}} \cup R_{\text{reac}} \cup W_{\text{other}}$ is the set of all partitions of the collection **S**, and $L_{\text{reac}} \cup R_{\text{reac}} \cup W_{\text{other}}$ is the set of partitions contained in the partition matching π_{exp}. Since all sets L_{reac}, R_{reac}, and W_{other} are pairwise disjoint, the number of partitions in π_{exp} is precisely $|L_{\text{reac}}| + |R_{\text{reac}}| + |W_{\text{other}}|$.

Now consider the set $W_L = L_{\text{free}} \cup L_{\text{reac}}$. Let U_L be the set of elements that appears in the left-collection of a partition in W_L. We have

- Every $P \in L_{\text{reac}}$ left-pairs an element in U_L;
- Every element in U_L is left-paired;
- If an element a in U_L is left-paired by a partition P, then $P \in L_{\text{reac}}$.

Therefore, the partitions in L_{reac} precisely left-pair the elements in U_L. This gives $|L_{\text{reac}}| = |U_L|$. Since there are only $|U_L|$ elements that appear in the left-collections in partitions in $L_{\text{free}} \cup L_{\text{reac}}$, we conclude that the partitions in $W_L = L_{\text{free}} \cup L_{\text{reac}}$ can be used to left-pair at most $|U_L| = |L_{\text{reac}}$ elements in any partition matching in **S**.

Similarly, the partitions in the set $W_R = R_{\text{free}} \cup R_{\text{reac}}$ can be used to right-pair at most $|R_{\text{reac}}|$ elements in any partition matching in **S**.

Therefore, any partition matching in the collection **S** can include at most $|L_{\text{reac}}|$ partitions in the set W_L, at most $|R_{\text{reac}}|$ partitions in the set W_R, and at most all partitions in the set W_{other}. Consequently, a maximum partition matching in **S** consists of at most $|L_{\text{reac}}| + |R_{\text{reac}}| + |W_{\text{other}}|$ partitions. Since the partition matching π_{exp} constructed by the algorithm greedy-expanding contains just this many partitions, π_{exp} is a maximum partition matching in the collection **S**.

Now it is clear how the maximum partition matching problem is solved.

Theorem 8 *The maximum partition matching problem is solvable in time $O(n^2 \log n)$.*

Proof Suppose that we are given a collection $\mathbf{S} = \{C_1, \ldots, C_k\}$ of pairwise disjoint subsets of $U = \{1, \ldots, n\}$.

In case $2^k \geq 4n$, we can call the algorithm partition-matching-I to construct a maximum partition matching in time $O(n^2)$.

In case $2^k < 4n$, we first call the algorithm greedy-expanding to construct a partition matching π_{exp} and compute the sets L_{reac} and R_{reac}. If L_{reac} and R_{reac} have no common partition, then according to the previous theorem, π_{exp} is already a maximum partition matching. Otherwise, we call the algorithm partition-flipping to construct a maximum partition matching. All these can be done in time $O(n^2 \log n)$. A detailed analysis of this algorithm can be found in [3].

See also

- ► Assignment and Matching
- ► Assignment Methods in Clustering
- ► Bi-objective Assignment Problem
- ► Communication Network Assignment Problem
- ► Frequency Assignment Problem
- ► Quadratic Assignment Problem

References

1. Akers SB, Krishnamurthy B (1989) A group-theoretic model for symmetric interconnection networks. IEEE Trans Comput 38:555–565
2. Chen C-C, Chen J (1997) Optimal parallel routing in star networks. IEEE Trans Comput 48:1293–1303
3. Chen C-C, Chen J (1999) The maximum partition matching problem with applications. SIAM J Comput 28:935–954
4. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, New York

Maximum Satisfiability Problem

MAX-SAT

Roberto Battiti
Dip. Mat., Universitá Trento, Povo (Trento), Italy

MSC2000: 03B05, 68Q25, 90C09, 90C27, 68P10, 68R05, 68T15, 68T20, 94C10

Article Outline

Keywords
See also
References

Keywords

Maximum satisfiability; Local search; Approximation algorithms; History-sensitive heuristics

In the maximum satisfiability (MAX-SAT) problem one is given a *Boolean formula in conjunctive normal form*, i. e., as a conjunction of clauses, each clause being a disjunction. The task is to find an assignment of truth values to the variables that satisfies the maximum number of clauses.

Let n be the number of variables and m the number of clauses, so that a formula has the following form:

$$\bigwedge_{1 \leq i \leq m} \left(\bigvee_{1 \leq k \leq |C_i|} l_{ik} \right),$$

where $|C_i|$ is the number of literals in clause C_i and l_{ik} is a *literal*, i. e., a propositional variable u_j or its negation $\overline{u_j}$, for $1 \leq j \leq n$. The set of clauses in the formula is denoted by $\mathcal{C}$. If one associates a weight w_i to each clause C_i one obtains the *weighted MAX-SAT problem*, denoted as MAX W-SAT: one is to determine the assignment of truth values to the n variables that maximizes the sum of the weights of the satisfied clauses. In

the literature one often considers problems with different numbers k of literals per clause, defined as MAX-k-SAT, or MAX W-k-SAT in the weighted case. In some papers MAX-k-SAT instances contain up to k literals per clause, while in other papers they contain exactly k literals per clause. We consider the second option unless otherwise stated.

MAX-SAT is of considerable interest not only from the theoretical side but also from the practical one. On one hand, the decision version SAT was the first example of an *NP*-complete problem [16], moreover MAX-SAT and related variants play an important role in the characterization of different approximation classes like APX and PTAS [5]. On the other hand, many issues in mathematical logic and artificial intelligence can be expressed in the form of satisfiability or some of its variants, like constraint satisfaction. Some exemplary problems are consistency in expert system knowledge bases [46], integrity constraints in databases [4,23], approaches to inductive inference [35,40], asynchronous circuit synthesis [32]. An extensive review of algorithms for MAX-SAT appeared in [9].

M. Davis and H. Putnam [19] started in 1960 the investigation of useful strategies for handling resolution in the satisfiability problem. Davis, G. Logemann and D. Loveland [18] avoid the memory explosion of the original DP algorithm by replacing the resolution rule with the *splitting rule*. A recent review of advanced techniques for resolution and splitting is presented in [31].

The MAX W-SAT problem has a natural integer linear programming formulation. Let $y_j = 1$ if Boolean variable u_j is 'true', $y_j = 0$ if it is 'false', and let the Boolean variable $z_i = 1$ if clause C_i is satisfied, $z_i = 0$ otherwise. The integer linear program is:

$$\max \sum_{i=1}^{m} w_i z_i$$

subject to the constraints:

$$\begin{cases} \sum_{j \in U_i^+} y_j + \sum_{j \in U_i^-} (1 - y_j) \geq z_i, \\ \qquad i = 1, \ldots, m, \\ y_j \in \{0,1\}, \qquad j = 1, \ldots, n, \\ z_i \in \{0,1\}, \qquad i = 1, \ldots, m, \end{cases}$$

where U_i^+ and U_i^- denote the set of indices of variables that appear unnegated and negated in clause C_i, respectively. If one neglects the objective function and sets all z_i variables to 1, one obtains an integer programming feasibility problem associated to the SAT problem [11].

The integer linear programming formulation of MAX-SAT suggests that this problem could be solved by a branch and bound method (cf. also ► Integer programming: Branch and bound methods). A usable method uses Chvátal cuts. In [35] it is shown that the resolvents in the propositional calculus correspond to certain *cutting planes* in the integer programming model of inference problems.

Linear programming relaxations of integer linear programming formulations of MAX-SAT have been used to obtained upper bounds in [27,33,55]. A linear programming and rounding approach for MAX-2-SAT is presented in [13]. A method for strengthening the generalized set covering formulation is presented in [47], where Lagrangian multipliers guide the generation of cutting planes.

The first approximation algorithms with a 'guaranteed' quality of approximation [5] were proposed by D.S. Johnson [38] and use greedy construction strategies. The original paper [38] demonstrated for both of them a performance ratio 1/2. In detail, let k be the minimum number of variables occurring in any clause of the formula, $m(x, y)$ the number of clauses satisfied by the feasible solution y on instance x, and $m^*(x)$ the maximum number of clauses that can be satisfied.

For any integer $k \geq 1$, the first algorithm achieves a feasible solution y of an instance x such that

$$\frac{m(x,y)}{m^*(x)} \geq 1 - \frac{1}{k+1},$$

while the second algorithm obtains

$$\frac{m(x,y)}{m^*(x)} \geq 1 - \frac{1}{2^k}.$$

Recently (1997) it has been proved [12] that the second algorithm reaches a performance ratio 2/3. There are formulas for which the second algorithm finds a truth assignment such that the ratio is 2/3. Therefore this bound cannot be improved [12].

One of the most interesting approaches in the design of new algorithms is the use of *randomization*.

During the computation, random bits are generated and used to influence the algorithm process. In many cases randomization allows to obtain better (expected) performance or to simplify the construction of the algorithm. Two randomized algorithms that achieve a performance ratio of 3/4 have been proposed in [27] and [55]. Moreover, it is possible to derandomize these algorithms, that is, to obtain deterministic algorithms that preserve the same bound 3/4 for every instance. The approximation ratio 3/4 can be slightly improved [28]. T. Asano [2] (following [3]) has improved the bound to 0.77. For the restricted case of MAX-2-SAT, one can obtain a more substantial improvement (performance ratio 0.931) with the technique in [21]. If one considers only satisfiable MAX W-SAT instances, L. Trevisan [54] obtains a 0.8 approximation factor, while H. Karloff and U. Zwick [41] claim a 0.875 performance ratio for satisfiable instances of MAX W-3-SAT. A strong negative result about the approximability can be found in [36]: Unless $P = NP$ MAX W-SAT cannot be approximated in polynomial time within a performance ratio greater than 7/8.

MAX-SAT is among the problems for which local search has been very successful: in practice, local search and its variations are the only efficient and effective method to address large and complex real-world instances. Different variations of local search with randomness techniques have been proposed for SAT and MAX-SAT starting from the late 1980s, see for example [30,52], motivated by previous applications of 'min-conflicts' heuristics in the area of artificial intelligence [44].

The general scheme is based on generating a starting point in the set of admissible solution and trying to improve it through the application of basic moves. The search space is given by all possible truth assignments. Let us consider the elementary changes to the current assignment obtained by changing a single truth value. The definitions are as follows.

Let U be the discrete search space: $U = \{0, 1\}^n$, and let f be the number of satisfied clauses. In addition, let $U^{(t)} \in U$ be the current configuration along the *search trajectory* at iteration t, and $N(U^{(t)})$ the neighborhood of point $U^{(t)}$, obtained by applying a set of basic moves μ_i $(1 \le i \le n)$, where μ_i complements the ith bit u_i of the string: $\mu_i\ (u_1, \ldots, u_i, \ldots, u_n) = (u_1, \ldots, 1 - u_i, \ldots, u_n)$:

$$N(U^{(t)}) = \left\{U \in U\colon\ U = \mu_i,\ U^{(t)},\ i = 1, \ldots, n\right\}.$$

The version of local search that we consider starts from a random initial configuration $U^{(0)} \in U$ and generates a search trajectory as follows:

$$V = \text{BESTNEIGHBOR}(N(U^{(t)})), \tag{1}$$

$$U^{(t+1)} = \begin{cases} V & \text{if } f(V) > f(U^{(t)}), \\ U^{(t)} & \text{if } f(V) \le f(U^{(t)}) \end{cases} \tag{2}$$

where BESTNEIGHBOR selects $V \in N(U^{(t)})$ with the best f value and ties are broken randomly. V in turn becomes the new current configuration if f improves. Other versions are satisfied with an improving (or nonworsening) neighbor, not necessarily the best one. Clearly, local search stops as soon as the first *local optimum* point is encountered, when no improving moves are available, see (2). Let us define as LS^+ a modification of LS where a specified number of iterations are executed and the candidate move obtained by BESTNEIGHBOR is always accepted even if the f value remains equal or worsens.

Properties about the number of clauses satisfied at a *local optimum* have been demonstrated. Let m^* be the best value and k the minimum number of literals contained in the problem clauses. Let m_{loc} be the number of satisfied clauses at a local optimum of any instance of MAX-SAT with at least k literals per clause. m_{loc} satisfies the following bound [34]:

$$m_{\text{loc}} \ge \frac{k}{k+1} m$$

and the bound is sharp. Therefore, if m_{loc} is the number of satisfied clauses at a local optimum, then:

$$m_{\text{loc}} \ge \frac{k}{k+1} m^*. \tag{3}$$

State-of-the-art heuristics for MAX-SAT are obtained by complementing local search with schemes that are capable of producing better approximations beyond the locally optimal points. In some cases, these schemes generate a sequence of points in the set of admissible solutions in a way that is fixed before the search

starts. An example is given by *multiple runs* of local search starting from different random points. The algorithm does not take into account the *history* of the previous phase of the search when the next points are generated. The term 'memory-less' denotes this lack of feedback from the search history.

In addition to the cited multiple-run local search, these techniques are based on Markov processes (simulated annealing; cf. also ► Simulated annealing methods in protein folding), 'plateau' search and 'random noise' strategies, or combinations of randomized constructions and local search. The use of a Markov process to generate a stochastic search trajectory is adopted, for example in [53].

The *Gsat algorithm* was proposed in [52] as a *model-finding procedure*, i. e., to find an interpretation of the variables under which the formula comes out 'true'. Gsat consists of multiple runs of LS^+, each run consisting of a number of iterations that is typically proportional to the problem dimension n. An empirical analysis of Gsat is presented in [24,25]. Different 'noise' strategies to escape from attraction basins are added to Gsat in [50,51].

A hybrid algorithm that combines a randomized greedy construction phase to generate initial candidate solutions, followed be a local improvement phase is the *GRASP* scheme proposed in [48] for the SAT and generalized for the MAX W-SAT problem in [49]. GRASP is an iterative process, with each iteration consisting of two phases, a construction phase and a local search phase.

Different *history-sensitive heuristics* have been proposed to continue local search schemes beyond local optimality. These schemes aim at intensifying the search in promising regions and at diversifying the search into uncharted territories by using the information collected from the previous phase (the history) of the search. Because of the internal feedback mechanism, some algorithm parameters can be modified and tuned in an *on-line* manner, to reflect the characteristics of the *task* to be solved and the *local* properties of the configuration space in the neighborhood of the current point. This tuning has to be contrasted with the *off-line* tuning of an algorithm, where some parameters or choices are determined for a given problem in a preliminary phase and they remain fixed when the algorithm runs on a specific instance.

Tabu search is a *history-sensitive heuristic* proposed by F. Glover [26] and, independently, by P. Hansen and B. Jaumard, that used the term 'SAMD' (steepest ascent mildest descent) and applied it to the MAX-SAT problem in [34]. The main mechanism by which the history influences the search in tabu search is that, at a given iteration, some neighbors are *prohibited*, only a nonempty subset $N_A(U^{(t)}) \subset N(U^{(t)})$ of them is *allowed*. The general way of generating the search trajectory that we consider is given by:

$$N_A(U^{(t)}) = \text{allow}(N(U^{(t)}), \ldots, U^{(t)}), \tag{4}$$

$$U^{(t+1)} = \text{BESTNEIGHBOR}(N_A(U^{(t)})). \tag{5}$$

The set-valued function allow selects a nonempty subset of $N(U^{(t)})$ in a manner that depends on the entire previous history of the search $U^{(0)}, \ldots, U^{(t)}$. A specialized tabu search heuristic is used in [37] to speed up the search for a solution (if the problem is satisfiable) as part of a branch and bound algorithm for SAT, that adopts both a relaxation and a decomposition scheme by using polynomial instances, i. e., 2-SAT and Horn-SAT.

Different methods to generate prohibitions produce *discrete dynamical systems* with qualitatively different *search trajectories*. In particular, prohibitions based on a list of *moves* lead to a faster escape from a locally optimal point than prohibitions based on a list of visited *configurations* [6]. In detail, the function allow can be specified by introducing a *prohibition parameter* T (also called *list size*) that determines how long a move will remain prohibited after its execution. The *fixed tabu search* algorithm is obtained by fixing T throughout the search [26]. A neighbor is allowed if and only if it is obtained from the current point by applying a move that has not been used during the last T iterations. In detail, if $\text{LU}(\mu)$ is the last usage time of move μ ($\text{LU}(\mu) = -\infty$ at the beginning):

$$N_A(U^{(t)}) = \left\{U = \mu U^{(t)} : \ \text{LU}(\mu) < (t - T)\right\}.$$

The *reactive tabu search* algorithm of [10], defines simple rules to determine the prohibition parameter by reacting to the repetition of previously-visited configurations. One has a repetition if $U^{(t+R)} = U^{(t)}$ for $R \geq 1$. The prohibition period T depends on the iteration t and

a *reaction equation* is added to the dynamical system:

$$T^{(t)} = \text{react}(T^{(t-1)}, U^{(0)}, \ldots, U^{(t)}).$$

An algorithm that combines local search and *nonoblivious local search* [8], the use of prohibitions, and a reactive scheme to determine the prohibition parameter is the *Hamming-reactive tabu search* algorithm proposed in [7], which contains also a detailed experimental analysis.

Given the hardness of the problem and the relevancy for applications in different fields, the emphasis on the experimental analysis of algorithms for the MAX-SAT problem has been growing in recent years (as of 2000).

In some cases the experimental comparisons have been executed in the framework of 'challenges,' with support of electronic collection and distribution of software, problem generators and test instances. An example is the the Second DIMACS algorithm implementation challenge on cliques, coloring and satisfiability, whose results have been published in [39]. Practical and industrial MAX-SAT problems and benchmarks, with significant case studies are also presented in [20]. Some basic problem models that are considered both in theoretical and in experimental studies of MAX-SAT algorithms are described in [31].

Different algorithms demonstrate a different degree of effort, measured by number of elementary steps or CPU time, when solving different kinds of instances. For example, in [45] it is found that some distributions used in past experiments are of little interest because the generated formulas are almost always very easy to satisfy. It also reports that one can generate very hard instances of k-SAT, for $k \geq 3$. In addition, it reports the following observed behavior for random fixed length 3-SAT formulas: if r is the ratio r of clauses to variables ($r = m/n$), almost all formulas are satisfiable if $r < 4$, almost all formulas are unsatisfiable if $r > 4.5$. A rapid transition seems to appear for $r \approx 4.2$, the same point where the computational complexity for solving the generated instances is maximized, see [17,42] for reviews of experimental results.

Let κ be the least real number such that, if r is larger than κ, then the probability of $\mathcal{C}$ being satisfiable converges to 0 as n tends to infinity. A notable result found independently by many people, including [22] and [14] is that

$$\kappa \leq \log_{\frac{8}{7}} 2 = 5.191.$$

A series of theoretical analyses aim at approximating the *unsatisfiability threshold* of random formulas [1,15,29,43].

See also

- ► Greedy Randomized Adaptive Search Procedures
- ► Integer Programming

References

1. Achlioptas D, Kirousis LM, Kranakis E, Krinzac D (1997) Rigorous results for random (2 + p)-SAT. In: Proc. Work. on Randomized Algorithms in Sequential, Parallel and Distributed Computing (RALCOM 97), Santorini, Greece, pp 1–10
2. Asano T (1997) Approximation algorithms for MAX-SAT: Yannakakis vs. Goemans–Williamson. In: Proc. 3rd Israel Symp. on the Theory of Computing and Systems, Ramat Gan, Israel, pp 24–37
3. Asano T, Ono T, Hirata T (1996) Approximation algorithms for the maximum satisfiability problem. Proc. 5th Scandinavian Work. Algorithms Theory, pp 110–111
4. Asirelli P, de Santis M, Martelli A (1985) Integrity constraints in logic databases. J Logic Programming 3:221–232
5. Ausiello G, Crescenzi P, Protasi M (1995) Approximate solution of NP optimization problems. Theoret Comput Sci 150:1–55
6. Battiti R (1996) Reactive search: Toward self-tuning heuristics. In: Rayward-Smith VJ, Osman IH, Reeves CR, Smith GD (eds) Modern Heuristic Search Methods. Wiley, New York, pp 61–83
7. Battiti R, Protasi M (1997) Reactive search, a history-sensitive heuristic for MAX-SAT. ACM J Experimental Algorithmics 2:2
8. Battiti R, Protasi M (1997) Solving MAX-SAT with nonoblivious functions and history-based heuristics. In: Du D-Z, Gu J, Pardalos PM (eds) Satisfiability Problem: Theory and Applications. DIMACS, vol 35. Amer. Math. Soc. and ACM, Providence, RI, pp 649–667
9. Battiti R, Protasi M (1998) Approximate algorithms and heuristics for MAX-SAT. In: Du D-Z and Pardalos PM (eds) Handbook Combinatorial Optim. Kluwer, Dordrecht, pp 77–148
10. Battiti R, Tecchiolli G (1994) The reactive tabu search. ORSA J Comput 6(2):126–140
11. Blair CE, Jeroslow RG, Lowe JK (1986) Some results and experiments in programming for propositional logic. Comput Oper Res 13(5):633–645
12. Chen J, Friesen D, Zheng H (1997) Tight bound on Johnson's algorithm for MAX-SAT. In: Proc. 12th Annual IEEE

Conf. Computational Complexity (Ulm, Germany), pp 274–281
13. Cheriyan J, Cunningham WH, Tuncel T, Wang Y (1996) A linear programming and rounding approach to MAX 2-SAT. In: Trick M, Johson DS (eds) Proc. Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability. DIMACS vol 26, pp 395–414
14. Chvátal V, Szemerédi E (1988) Many hard examples for resolution. J ACM 35:759–768
15. Chvátal V, Reed B (1992) Mick gets some (the odds are on his side). In: Proc. 33th Ann. IEEE Symp. on Foundations of Computer Sci., IEEE Computer Soc., pp 620–627
16. Cook SA (1971) The complexity of theorem-proving procedures. In: Proc. Third Annual ACM Symp. Theory of Computing, pp 151–158
17. Cook SA, Mitchell DG (1997) Finding hard instances of the satisfiability problem: A survey. In: Du D-Z Gu J, Pardalos PM (eds) Satisfiability Problem: Theory and Applications. DIMACS, vol 35. Amer. Math. Soc. and ACM, Providence, RI, pp 1–17
18. Davis M, Logemann G, Loveland D (1962) A machine program for theorem proving. Comm ACM 5:394–397
19. Davis M, Putnam H (1960) A computing procedure for quantification theory. J ACM 7:201–215
20. Du D-Z, Gu J, Pardalos PM (eds) (1997) Satisfiability problem: Theory and applications. DIMACS, vol 35. Amer. Math. Soc. and ACM, Providence, RI
21. Feige U, Goemans MX (1995) Approximating the value of two proper proof systems, with applications to MAX-2SAT and MAX-DICUT. In: Proc. Third Israel Symp. Theory of Computing and Systems, pp 182–189
22. Franco J, Paull M (1983) Probabilistic analysis of the Davis–Putnam procedure for solving the satisfiability problem. Discrete Appl Math 5:77–87
23. Gallaire H, Minker J, Nicolas JM (1984) Logic and databases: A deductive approach. Computing Surveys 16(2):153–185
24. Gent IP, Walsh T (1993) An empirical analysis of search in GSAT. J Artif Intell Res 1:47–59
25. Gent IP, Walsh T (1993) Towards an understanding of hill-climbing procedures for SAT. In: Proc. Eleventh Nat. Conf. Artificial Intelligence, AAAI Press/MIT, pp 28–33
26. Glover F (1989) Tabu search: Part I. ORSA J Comput 1(3):190–260
27. Goemans MX, Williamson DP (1994) New 3/4-approximation algorithms for the maximum satisfiability problem. SIAM J Discret Math 7(4):656–666
28. Goemans MX, Williamson DP (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J ACM 42(6):1115–1145
29. Goerdt A (1996) A threshold for unsatisfiability. J Comput Syst Sci 53:469–486
30. Gu J (1992) Efficient local search for very large-scale satisfiability problem. ACM SIGART Bull 3(1):8–12
31. Gu J, Purdom PW, Franco J, Wah BW (1997) Algorithms for the satisfiability (SAT) problem: A survey. In: Du D-Z Gu J, Pardalos PM (eds) Satisfiability Problem: Theory and Applications. DIMACS, vol 35. Amer. Math. Soc. and ACM, Providence, RI
32. Gu J, Puri R (1995) Asynchronous circuit synthesis with Boolean satisfiability. IEEE Trans Computer-Aided Design Integr Circuits 14(8):961–973
33. Hammer PL, Hansen P, Simeone B (1984) Roof duality, complementation and persistency in quadratic 0-1 optimization. Math Program 28:121–155
34. Hansen P, Jaumard B (1990) Algorithms for the maximum satisfiability problem. Computing 44:279–303
35. Hooker JN (1988) Resolution vs. cutting plane solution of inference problems: some computational experience. Oper Res Lett 7(1):1–7
36. Håstad J (1997) Some optimal inapproximability results. In: Proc. 28th Annual ACM Symp. on Theory of Computing, El Paso, Texas, pp 1–10
37. Jaumard B, Stan M, Desrosiers J (1996) Tabu search and a quadratic relaxation for the satisfiability problem. In: Trick M, Johson DS (eds) Proc. Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability. DIMACS, vol 26, pp 457–477
38. Johnson DS (1974) Approximation algorithms for combinatorial problems. J Comput Syst Sci 9:256–278
39. Johnson DS, Trick M (eds) (1996) Cliques, coloring, and satisfiability: Second DIMACS implementation challenge. DIMACS, vol 26. Amer Math Soc, Providence, RI
40. Kamath AP, Karmarkar NK, Ramakrishnan KG, Resende MG (1990) Computational exprience with an interior point algorithm on the satisfiability problem. Ann Oper Res 25:43–58
41. Karloff H, Zwick U (1997) A 7/8-approximation algorithm for MAX 3SAT? In: Proc. 38th Annual IEEE Symp. Foundations of Computer Sci., IEEE Computer Soc
42. Kirkpatrick S, Selman B (1994) Critical behavior in the satisfiability of random Boolean expressions. Science 264:1297–1301
43. Kirousis LM, Kranakis E, Krizanc D (Sept. 1996) Approximating the unsatisfiability threshold of random formulas. In: Proc. Fourth Annual European Symp. Algorithms. Springer, Berlin, pp 27–38
44. Minton S, Johnston MD, Philips AB, Laird P (1990) Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In: Proc. 8th Nat. Conf. Artificial Intelligence (AAAI-90), pp 17–24
45. Mitchell D, Selman B, Levesque H (1992) Hard and easy distributions of SAT problems. In: Proc. 10th Nat. Conf. Artificial Intelligence (AAAI-92), pp 459–465
46. Nguyen TA, Perkins WA, Laffrey TJ, Pecora D (1985) Checking an expert system knowledge base for consistency and completeness. In: Proc. Internat. Joint Conf. on Artificial Intelligence, pp 375–378

47. Nobili P, Sassano A (1996) Strengthening Lagrangian bounds for the MAX-SAT problem. Techn. Report Inst. Informatik Köln Univ., Germany, no. 96–230; Franco J, Gallo G, Kleine Buening H (eds) Proc. Work Satisfiability Problem, Siena, Italy
48. Resende MGC, Feo TA (1996) A GRASP for satisfiability. In: Trick M, Johson DS (eds) Proc. Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability. DIMACS, vol 26. Amer. Math. Soc., Providence, RI, pp 499–520
49. Resende MGC, Pitsoulis LS, Pardalos PM (1997) Approximate solution of weighted MAX-SAT problems using GRASP. In: Du D-Z Gu J, Pardalos PM (eds) Satisfiability Problem: Theory and Applications. DIMACS, vol 35. Amer Math Soc, Providence, RI
50. Selman B, Kautz H (1993) Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In: Proc. Internat. Joint Conf. Artificial Intelligence, pp 290–295
51. Selman B, Kautz HA, Cohen B (1996) Local search strategies for satisfiability testing. In: Trick M, Johson DS (eds) Proc. Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability. DIMACS, vol 26, pp 521–531
52. Selman B, Levesque H, Mitchell D (1992) A new method for solving hard satisfiability problems. In: Proc. 10th Nat. Conf. Artificial Intelligence (AAAI-92), pp 440–446
53. Spears WM (1996) Simulated annealing for hard satisfiability problems. In: Trick M, Johnson DS (eds) Proc. Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability. DIMACS vol 26, pp 533–555
54. Trevisan L (1997) Approximating satisfiable satisfiability problems. In: Proc. 5th Annual European Symp. Algorithms, Graz. Springer, Berlin, pp 472–485
55. Yannakakis M (1994) On the approximation of maximum satisfiability. J Algorithms 17:475–502

Medium-Term Scheduling of Batch Processes

STACY L. JANAK, CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering,
Princeton University, Princeton, USA

Article Outline

Introduction
  Problem Statement
Formulation
Models
  Short-Term Scheduling Model
Cases
  Case 1: Nominal Run without Campaign Mode Production
Conclusions
References

Introduction

In multiproduct and multipurpose batch plants, different products can be manufactured via the same or a similar sequence of operations by sharing available pieces of equipment, intermediate materials, and other production resources. They are ideally suited to manufacture products that are produced in small quantities or for which the production recipe or the customer demand pattern is likely to change. The inherent operational flexibility of this type of plant provides the opportunity for increased savings through the realization of an efficient production schedule which can reduce inventories, production and transition costs, and production shortfalls.

The problem of production scheduling and planning for multiproduct and multipurpose batch plants has received a considerable amount of attention during the last two decades. Extensive reviews have been written by Reklaitis [10], Pantelides [9], Shah [11] and more recently by Floudas and Lin [4,5]. Most of the work in the area of multiproduct batch plants has dealt with either the long-term planning problem or the short-term scheduling problem. Both planning and scheduling deal with the allocation of available resources over time to perform a set of tasks required to manufacture one or more products. However, long-term planning problems deal with longer time horizons (e. g., several months or years) and are focused on higher level decisions such as timing and location of additional facilities and levels of production. In contrast, short-term scheduling models address shorter time horizons (e. g., several days) and are focused on determining detailed sequencing of various operational tasks. The area of medium-term scheduling, however, which involves medium time horizons (e. g. several weeks) and still aims to determine detailed production schedules, can result in very large-scale problems and has received much less attention in the literature.

For medium-term scheduling, relatively little work has been presented in the literature. Medium-term

scheduling can be quite computationally complex, thus it is common for mathematical programming techniques to be used in their solution. The most widely employed strategy to overcome the computational difficulty is based on the idea of decomposition. The decomposition approach divides a large and complex problem, which may be computationally expensive or even intractable when formulated and solved directly as a single MILP model, to smaller subproblems, which can be solved much more efficiently. There have been a wide variety of decomposition approaches proposed in the literature. In addition to decomposition techniques developed for general forms of MILP problems, various approaches that exploit the characteristics of specific process scheduling problems have also been proposed. In most cases, the decomposition approaches only lead to suboptimal solutions, however, they substantially reduce the problem complexity and the solution time, making MILP based techniques applicable for large, real-world problems.

In this chapter, we propose an enhanced State-Task Network MILP model for the medium-term production scheduling of a multipurpose, multiproduct industrial batch plant. The proposed approach extends the work of Ierapetritou and Floudas [6] and Lin et al. [8] to consider a large-scale production facility and account for various storage policies (UIS, NIS, ZW), variable batch sizes and processing times, batch mixing and splitting, sequence-dependent changeover times, intermediate due dates, products used as raw materials, and several modes of operation. The methodology consists of the decomposition of the whole scheduling period into successive short horizons of a few days. A decomposition model is implemented to determine each short horizon and the corresponding products to be included. Then, a novel continuous-time formulation for short-term scheduling of batch processes with multiple intermediate due dates is applied to each short horizon selected, leading to a large-scale mixed-integer linear programming (MILP) problem. The scheduling model includes over 80 pieces of equipment and can take into account the processing recipes of hundreds of different products. Several characteristics of the production plant are incorporated into the scheduling model and actual plant data are used to model all parameters.

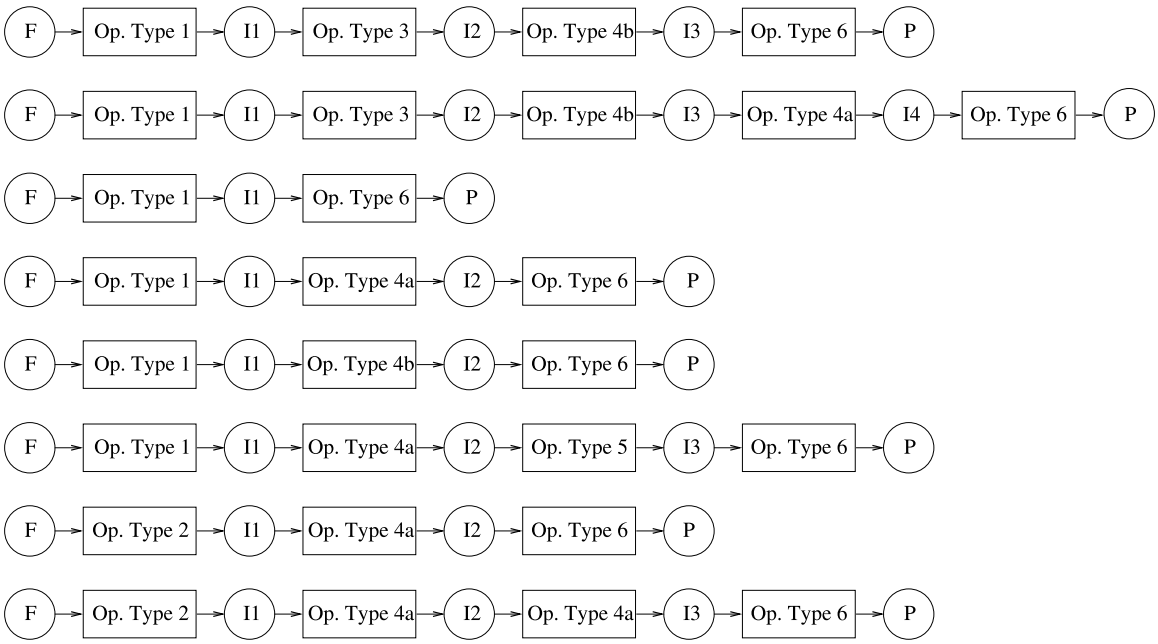
Problem Statement

In the multiproduct batch plant investigated, there are several different types of operations (or tasks) termed operation type 1 to operation type 6. The plant has many different types of units and over 80 are modeled explicitly. Hundreds of different products can be produced and for each of them, one of the processing recipes shown in Fig. 1 or a slight variation is applied. The recipes are represented in the form of State-Task Network (STN), in which the *state* node is denoted by a circle and the *task* node by a rectangle. The STN representation provides the flow of material through various tasks in the production facility to produce different types of final products and does not represent the actual connectivity of equipment in the plant.

For the first type of STN shown in Fig. 1, raw materials (or state F) are fed into a type 1 unit and undergo operation type 1 to produce an intermediate (or state I1). This intermediate then undergoes operation type 3 in a type 3 unit to produce another intermediate (or state I2). This second intermediate is then sent to a type 4b unit before the resulting intermediate material (or state I3) is sent to a type 6 unit to undergo an operation type 6 task to produce a final product (or state P). The information on which units are suitable for each product is given. All the units are utilized in a batch mode with the exception of the type 5 and 6 units, which operate in a continuous mode. The capacity limits of the type 1, type 2, and type 3 units vary from one product to another, while the capacity limits of the types 4a, 4b, 5 and 6 units are the same for all suitable products. The processing time or processing rate of each task in the suitable units is also specified. Also, some products require other products as their raw materials, creating very complicated state-task networks.

The time horizon considered for production scheduling is a few weeks or longer. Customer orders are fixed throughout the time horizon with specified amounts and due dates. There is no limitation on external raw materials and we apply the zero-wait storage condition or limited intermediate storage capacity for all materials based on actual plant data. There are two different types of products produced, category 1 and 2.

The sixth STN shown in Fig. 1 shows a special type of product, denoted as a campaign product. For this



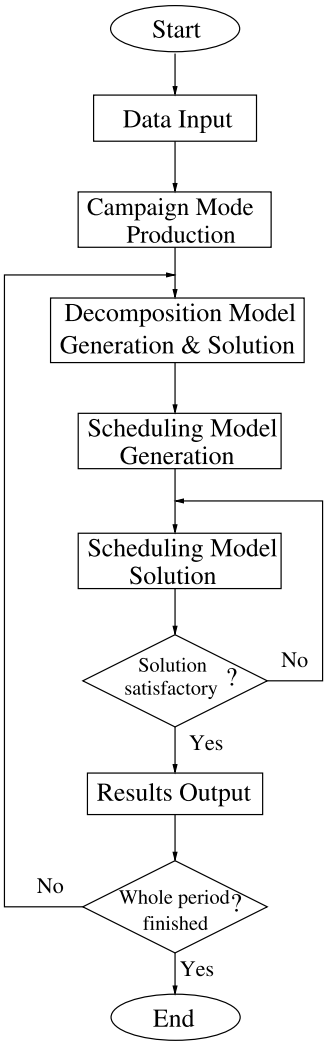
Medium-Term Scheduling of Batch Processes, Figure 1
State-task network (STN) representation of plant

type of product, raw materials are fed into up to three type 1 units and undergo operation type 1 to produce an intermediate, or state I1. This intermediate is then sent to one of two type 4a units before being processed in the type 5 unit, which is a continuous unit. Finally, the intermediate material (or state I3) is sent to a type 6 unit, producing a final campaign product (or state P). Because product changeovers in the type 5 unit can be undesirable, there was a need to introduce the ability to fix campaigns for continuous production of a single product in the type 5 unit, called campaign mode production.

Formulation

The overall methodology for solving the medium-range production scheduling problem is to decompose the large and complex problem into smaller short-term scheduling subproblems in successive time horizons [8]. The flowchart for this rolling horizon approach is shown in Fig. 2. The first step is to input relevant data into the formulation. Then, if necessary, campaign mode production is determined. Next, the overall medium-term scheduling problem is considered. A decomposition model is formulated and solved to determine the current time horizon and corresponding products that should be included in the current subproblem. According to the solution of the decomposition model, a short-term scheduling model is formulated using the information on customer orders, inventory levels, and processing recipes. The resulting MILP problem is a large-scale, complex problem which requires a large computational effort for its solution. When a satisfactory solution is determined, the relevant data is output and the next time horizon is considered. The above procedure is applied iteratively in an automatic fashion until the whole time horizon under consideration has been scheduled.

Note that the decomposition model determines how many days and products to consider in the shorter scheduling horizon subject to an upper limit on the complexity of the resulting mathematical model. Products are selected for the scheduling horizon if there is an order for the product, if the product has an order within a set amount of time into the future, if the product is used as a raw material for another product which is included, if the product was still processing in the previous scheduling horizon, or if the product is a campaign product and is included in a campaign for the current horizon.



Medium-Term Scheduling of Batch Processes, Figure 2
Flowchart of the rolling horizon approach

Models

A key component of the rolling horizon approach is the determination of the time horizon and the products which should be included for each short-term scheduling subproblem. We extend the two-level decomposition formulation of Lin et al. [8] which partitions the entire scheduling horizon into shorter subhorizons by taking into account the trade-off between demand satisfaction, unit utilization, and model complexity. In the first level, the number of days in the time horizon and the main products which should be included are determined. In the second level, additional products are added to the horizon so that each of the first-stage units, or type 1 units, are fully utilized.

Short-Term Scheduling Model

Once the decomposition model has determined the days in the time horizon and the products to be included, a novel continuous-time formulation for short-term scheduling with multiple intermediate due dates is applied to determine the detailed production schedule. This formulation is based on the models of Floudas and coworkers [6,7,8] and is expanded and enhanced in this work to take into account specific aspects of the problem under consideration. The proposed short-term scheduling formulation requires the following indices, sets, parameters and variables:

Indices:

- d days;
- i processing tasks;
- j units;
- k orders;
- n event points representing the beginning of a task;
- s states;

Sets:

- D days in the overall scheduling horizon;
- D^{in} days in the current scheduling horizon;
- I processing tasks;
- I_j tasks which can be performed in unit (j);
- I_k tasks which process order (k);
- I_s^c tasks which consume state (s);
- I_s^p tasks which produce state (s);
- I^{in} tasks which are included in the current scheduling horizon;
- I^{T5} tasks which are used to determine the type 5 unit campaign;
- I^{T6b} tasks which are used to perform operation type 6 for category 1 products;
- J units;
- J_i units which are suitable for performing task (i);
- J^p units which are suitable for performing only processing tasks, or operation type 1, 2, 3, and 5 tasks;
- J^{T1} units which are suitable for performing only operation type 1 tasks;

J^{T4} units which are suitable for performing only operation type 4a and 4b tasks;
J^{T5} units are which used to determine the type 5 unit campaign;
J^{T6} units which are suitable for performing only operation type 6 tasks;
K orders;
K_i orders which are processed by task (i);
K_s orders which produce state (s);
K^{in} orders which are included in the current scheduling horizon;
N event points within the time horizon;
S states;
S_k states which are used to satisfy order (k);
S^{cat1} states which are category 1 final products;
S^{cat2} states which are category 2 final products;
S^{cpm} states which have minimum or maximum storage limitations;
S^{f} states which are final products, after operation type 6;
S^{i} states which are intermediate products, before operation type 6;
S^{in} states which are included in the current scheduling horizon;
S^{p} states which are either final or intermediate products;
S^{rw} states which are products and are used as raw materials for other products;
S^{st} states which have no intermediate storage;
S^{T5} states which are used to determine the type 5 unit campaign;
S^{unl} states which have unlimited intermediate storage;
S^{0} states which are external raw materials;

Parameters:

B_s^{max} the maximum suitable batch size used to produce product state (s);
B_s^{min} the minimum suitable batch size used to produce product state (s);
C a large constant (e. g., 10000);
cap_{ij}^{max} maximum capacity for task (i) in unit (j);
cap_{ij}^{min} minimum capacity for task (i) in unit (j);
dem_s demand for state (s) in the current scheduling horizon;
dem_s^{rw} demand for raw material product state (s);
dem_s^{tot} total demand for state (s) in the overall horizon;
$duek_{ksd}$ due date of order (k) for state (s) on day (d);
$ExtraTime_i$ amount of time needed for operation type 3 task after processing task (i);
$FixedTime_{ij}$ constant term of processing time for task (i) in unit (j);
H time horizon;
$mintasks$ the minimum number of tasks that must occur in the first-stage processing units, J^{T1};
N^{max} the maximum number of event points in the scheduling horizon;
$praw_{ss'}$ 0-1 parameter to relate final product (s) to its raw material product (s');
$price_s$ price of state (s);
$prior_s$ priority of product state (s);
$prior_s^{raw}$ priority of raw material state (s);
$RateCT_{ij}$ variable term of processing time for task (i) in unit (j);
rk_{ksd} amount of order (k) for state (s) on day (d);
$start_j$ the time at which unit (j) first becomes available in the current scheduling horizon;
$stcap_s^{max}$ maximum capacity for storage of state (s);
$stcap_s^{min}$ minimum capacity for storage of state (s);
α coefficient for the demand satisfaction of individual orders term;
β coefficient for the due date satisfaction of individual orders term;
γ coefficient for the overall demand satisfaction slack variable term;
δ coefficient for the minimum inventory requirement in dedicated units term;
η coefficient for the artificial demands on raw material states term;
κ coefficient for the minimizing of binary variables term;
λ coefficient for the minimizing of active start times term;
μ a small constant (e. g., 0.01);
ρ_{is}^{c} proportion of state (s) consumed by task (i);
ρ_{is}^{p} proportion of state (s) produced by task (i);

$\tau_{ii'}$ sequence-dependent setup time between tasks (i) and (i');
ϕ coefficient for the satisfaction of orders term;
ω coefficient for the overall production term;

Continuous Variables:

$\boldsymbol{B(i,j,n)}$ amount of material undertaking task (i) in unit (j) at event point (n);
$\boldsymbol{D(s,n)}$ amount of state (s) delivered at event point (n);
$\boldsymbol{Df(s,n)}$ amount of state (s) delivered after the last event point;
$\boldsymbol{kD(k,s,n)}$ amount of state (s) delivered at event point (n) for order (k);
$\boldsymbol{kDf(k,s,n)}$ amount of state (s) delivered after the last event point for order (k);
$\boldsymbol{sla1(k,s,d)}$ amount of state (s) due on day (d) for order (k) that is not delivered;
$\boldsymbol{sla2(k,s,d)}$ amount of state (s) due on day (d) for order (k) that is over delivered;
$\boldsymbol{slcap(s,n)}$ amount of state (s) that is deficient in its dedicated storage unit at event point (n);
$\boldsymbol{sll(s)}$ amount of state (s) due in the current time horizon but not made;
$\boldsymbol{sll^{raw}(s)}$ amount of raw material product state (s) artificially due in the current time horizon but not made;
$\boldsymbol{slorder(k)}$ 0-1 variable indicating if order (k) was met;
$\boldsymbol{slt1(k,s,d)}$ amount of time state (s) due on day (d) for order (k) is late;
$\boldsymbol{slt2(k,s,d)}$ amount of time state (s) due on day (d) for order (k) is early;
$\boldsymbol{ST(s,n)}$ amount of state (s) at event point (n);
$\boldsymbol{STF(s)}$ final amount of state (s) at the end of the current time horizon;
$\boldsymbol{STO(s)}$ initial amount of state (s) at the beginning of the current time horizon;
$\boldsymbol{T^f(i,j,n)}$ time at which task (i) finishes in unit (j) at event point (n);
$\boldsymbol{T^s(i,j,n)}$ time at which task (i) starts in unit (j) at event point (n);
$\boldsymbol{tot(s)}$ total amount of state (s) made in the current time horizon;
$\boldsymbol{tt^s(i,j,n)}$ starting time of the active task (i) in unit (j) at event point (n);

Binary Variables:

$\boldsymbol{wv(i,j,n)}$ assigns the beginning of task (i) in unit (j) at event point (n);
$\boldsymbol{y(i,k,n)}$ assigns the delivery of order (k) through task (i) at event point (n);

On the basis of this notation, the mathematical model for the short-term scheduling of an industrial batch plant with intermediate due dates involves the following constraints:

$$\sum_{i\in I^{\text{in}},I_j} wv(i,j,n) \le 1\,, \quad \forall j\in J,\ \ n\in N,\ \ n\le N^{\max} \tag{1}$$

$$cap_{ij}^{\min}\cdot wv(i,j,n) \le B(i,j,n) \le cap_{ij}^{\max}\cdot wv(i,j,n)\,, \quad \forall i\in I^{\text{in}},\ \ j\in J_i,\ \ n\in N,\ \ n\le N^{\max} \tag{2}$$

$$st(s,n)=0,\ \ \forall s\in S^{\text{in}},S^{\text{st}},\ s\notin S^{\text{cpm}},S^{\text{unl}},\ \ n\in N,\ \ n\le N^{\max} \tag{3}$$

$$st(s,n)\ge stcap_s^{\min}-slcap(s)\,, \quad \forall s\in S^{\text{in}},S^{\text{cpm}},\ \ n\in N,\ \ n\le N^{\max} \tag{4}$$

$$st(s,n)\le stcap_s^{\max}\,, \quad \forall s\in S^{\text{in}},S^{\text{cpm}},\ \ n\in N,\ \ n\le N^{\max} \tag{5}$$

$$ST(s,n)=ST(s,n-1)-D(s,n)+\sum_{i\in I_s^{\text{p}}}\rho_{is}^{\text{p}}\sum_{j\in J_i}B(i,j,n-1)-\sum_{i\in I_s^{\text{c}}}\rho_{is}^{\text{c}}\sum_{j\in J_i}B(i,j,n)\,, \quad \forall s\in S^{\text{in}},\ \ n\in N,\ \ n>1,\ \ n\le N^{\max} \tag{6}$$

$$ST(s,n)=STO(s)-D(s,n)-\sum_{i\in I_s^{\text{c}}}\rho_{is}^{\text{c}}\sum_{j\in J_i}B(i,j,n)\,, \quad \forall s\in S^{\text{in}},\ \ n\in N,\ \ n=1 \tag{7}$$

$$STF(s)=ST(s,n)-Df(s,n)+\sum_{i\in I_s^{\text{p}}}\rho_{is}^{\text{p}}\sum_{j\in J_i}B(i,j,n)\,, \quad \forall s\in S^{\text{in}},\ \ n\in N,\ \ n=N^{\max} \tag{8}$$

$$T^{\mathrm{f}}(i,j,n) = T^{\mathrm{s}}(i,j,n) + FixedTime_{ij} \cdot wv(i,j,n) + RateCT_{ij} \cdot B(i,j,n)\,, \quad \forall i \in I^{\mathrm{in}},\ j \in J^{\mathrm{p}} \cup J^{\mathrm{T6}}, J_i,\ n \in N,\ n \le N^{\max} \tag{9}$$

$$T^{\mathrm{f}}(i,j,n) \ge T^{\mathrm{s}}(i,j,n)\,, \quad \forall i \in I^{\mathrm{in}},\ j \in J^{\mathrm{T4}}, J_i,\ n \in N,\ n \le N^{\max} \tag{10}$$

$$T^{\mathrm{f}}(i,j,n) = H,\ \forall s \in S^{\mathrm{in}}, S^{\mathrm{st}},\ s \notin S^{\mathrm{unl}}\,, \quad i \in I^{\mathrm{in}}, I_s^{\mathrm{p}},\ j \in J^{\mathrm{T4}}, J_i,\ n \in N,\ n = N^{\max} \tag{11}$$

$$T^{\mathrm{s}}(i,j,n+1) \ge T^{\mathrm{f}}(i,j,n) + ExtraTime_i \cdot wv(i,j,n)\,, \quad \forall i \in I^{\mathrm{in}},\ j \in J_i,\ n \in N,\ n < N^{\max} \tag{12}$$

$$T^{\mathrm{s}}(i,j,n+1) \ge T^{\mathrm{f}}(i',j,n) + (\tau_{i'i} + ExtraTime_{i'}) \cdot wv(i',j,n) - H[1 - w(i',j,n)]\,, \quad \forall j \in J,\ i,i' \in I^{\mathrm{in}}, I_j,\ i \ne i',\ n \in N,\ n < N^{\max} \tag{13}$$

$$T^{\mathrm{s}}(i,j,n+1) \ge T^{\mathrm{f}}(i',j',n) - H[1 - wv(i',j',n)]\,, \quad \forall s \in S^{\mathrm{in}},\ i \in I^{\mathrm{in}}, I_s^{\mathrm{c}},\ i' \in I^{\mathrm{in}}, I_s^{\mathrm{p}},\ j \in J_i,\ j' \in J_{i'},\ j \ne j',\ n \in N,\ n < N^{\max} \tag{14}$$

$$T^{\mathrm{s}}(i,j,n+1) \le T^{\mathrm{f}}(i',j',n) + H[2 - wv(i',j',n) - wv(i,j,n+1)]\,, \quad \forall s \in S^{\mathrm{in}}, S^{\mathrm{st}}, s \notin S^{\mathrm{unl}},\ i \in I^{\mathrm{in}}, I_s^{\mathrm{c}},\ i' \in I^{\mathrm{in}}, I_s^{\mathrm{p}},\ j \in J_i,\ j' \in J_{i'},\ j \ne j',\ n \in N,\ n < N^{\max} \tag{15}$$

The allocation constraints in (1) express the requirement that for each unit (j) and at each event point (n), only one of the tasks that can be performed in the unit (i. e., $i \in I_j$) should take place. The capacity constraints in (2) express the requirement for the batch-size of a task (i) processing in a unit (j) at event point (n), $B(i,j,n)$, to be greater than the minimum amount of material, $cap_{ij}^{\min}$, and less than the maximum amount of material, $cap_{ij}^{\max}$, that can be processed by task (i) in unit (j). The storage constraints in (3) enforce that those states with no intermediate storage have to be consumed by some processing task or storage task immediately after they are produced. Constraints (4) represent the minimum required storage for state (s) in a dedicated storage tank where this amount can be violated, if necessary, by an amount $slcap(s)$ which is penalized in the objective function. Constraints (5) represent the maximum available storage capacity for state (s) based on the maximum storage capacity of the dedicated storage tank. According to the material balance constraints in (6), the amount of material of state (s) at event point (n) is equal to that at event point $(n-1)$ increased by any amounts produced at event point $(n-1)$, decreased by any amounts consumed at event point (n), and decreased by the amount required by the market at event point (n), $D(s,n)$. Constraints (7)–(8) represent the material balance on state (s) at the first and last event points, respectively. The duration constraints in (9) represent the relationship between the starting and finishing times of task (i) in unit (j) at event point (n) for all processing tasks (i. e., J^{p}) and all operation type 6 tasks (i. e., J^{T6}) where $FixedTime_{ij}$ are the fixed processing times for batch tasks and zero for continuous tasks and $RateCT_{ij}$ are the inverse of processing rates for continuous tasks and zero for batch tasks, respectively. Constraints (10) also represent the relationship between the starting and finishing times of task (i) in unit (j) at event point (n), but for operation type 4a and 4b tasks (i. e., J^{T4}). They do not impose exact durations for tasks in these units but just enforce that all tasks must end after they start. Constraints (11) are written only for tasks in units which are processing a nonstorable state (i. e., S^{st} and not S^{unl}) and enforce that task (i) taking place at the last event point (n) must finish at the end of the horizon.

The sequence constraints in (12) state that task (i) starting at event point $(n+1)$ should start after the end of the same task performed in the same unit (j) which has finished at the previous event point, (n) where extra time is added after task (i) at event point (n), if necessary. The constraints in (13) are written for tasks (i) and (i') that are performed in the same unit (j) at event points $(n+1)$ and (n), respectively. If both tasks take place in the same unit, they should be at most consecutive. The third set of sequence constraints in (14) relate tasks (i) and (i') which are performed in different units (j) and (j') but take place consecutively according to the production recipe. The zero-wait constraints in (15) are written for different tasks (i) and (i') that take place consecutively with the intermediate state (s)

having no possible intermediate storage and thus subject to the zero-wait condition.

$$\sum_{\substack{i\in I^{\text{in}},\\ I_k,I^{\text{T6b}}}} \sum_{\substack{n\in N\\ n\le N^{\max}}} y(i,k,n) + slorder(k) \ge 1\,, \quad \forall k \in K^{\text{in}} \tag{16}$$

$$\sum_{i\in I^{\text{in}},I_k,I^{\text{T6b}}} \sum_{\substack{n\in N\\ n\le N^{\max}}} y(k,i,n) \le \sum_{\substack{s\in S^{\text{in}},S^{\text{cat1}}\\ sl_s>0}} \sum_{\substack{d\in D^{\text{in}}\\ rk_{ksd}>0}} \left\lceil \frac{rk_{ksd}}{B_s^{\min}} \right\rceil, \quad \forall k \in K^{\text{in}} \tag{17}$$

$$\sum_{k\in K^{\text{in}},K_i} \sum_{j\in J_i} suit_{ij} \cdot y(i,k,n) \ge \sum_{j\in J_i,J^{\text{T6}}} wv(i,j,n)\,, \quad \forall i \in I^{\text{in}},I^{\text{T6b}},\ n \in N,\ n \le N^{\max} \tag{18}$$

$$\sum_{k\in K^{\text{in}},K_i} y(i,k,n) \le \sum_{j\in J_i,J^{\text{T6}}} wv(i,j,n)\,, \quad \forall i \in I^{\text{in}},I^{\text{T6b}},\ n \in N,\ n \le N^{\max} \tag{19}$$

$$kD(k,s,n+1) + kDf(k,s,n+1) \ge \sum_{j\in J_i} B(i,j,n) - C\cdot(1-y(i,k,n))\,, \quad \forall s \in S^{\text{in}},S^{\text{cat1}},\ k \in K^{\text{in}},K_s,\ i \in I_k,I^{\text{T6b}},\ n \in N,\ n < N^{\max} \tag{20}$$

$$D(s,n) = \sum_{k\in K^{\text{in}},K_s} kD(k,s,n)\,, \quad \forall s \in S^{\text{in}},S^{\text{cat1}},\ n \in N,\ n \le N^{\max} \tag{21}$$

$$Df(s,n) = \sum_{k\in K^{\text{in}},K_s} kDf(k,s,n)\,, \quad \forall s \in S^{\text{in}},S^{\text{cat1}},\ n \in N,\ n = N^{\max} \tag{22}$$

$$\sum_{\substack{n\in N\\ n<N^{\max}}} \left[kD(k,s,n+1) + kDf(k,s,n+1)\right] + sla1(k,s,d) \ge rk_{ksd}\,, \quad \forall s \in S^{\text{in}},S^{\text{cat1}},\ k \in K^{\text{in}},K_s,\ d \in D^{\text{in}}, rk_{ksd} > 0 \tag{23}$$

$$\sum_{\substack{n\in N\\ n<N^{\max}}} \left[kD(k,s,n+1) + kDf(k,s,n+1)\right] + stf(s) - sla2(k,s,d) \le rk_{ksd}\,, \quad \forall s \in S^{\text{in}},S^{\text{cat1}},\ k \in K^{\text{in}},K_s,\ d \in D^{\text{in}}, rk_{ksd} > 0 \tag{24}$$

$$T^{\text{f}}(i,j,n) - slt1(k,s,d,n) \le duek(k,s,d) + H\cdot(2-wv(i,j,n)-y(i,k,n))\,, \quad \forall s \in S^{\text{in}},S^{\text{cat1}},\ k \in K^{\text{in}},K_s,\ i \in I^{\text{in}},I_k,I^{\text{T6b}},\ j \in J_i,\ n \in N, n \le N^{\max},\ d \in D^{\text{in}}, rk_{ksd} > 0 \tag{25}$$

$$T^{\text{f}}(i,j,n) + slt2(k,s,d,n) \ge (duek(k,s,d) - 24) - H\cdot(2-wv(i,j,n)-y(i,k,n))\,, \quad \forall s \in S^{\text{in}},S^{\text{cat1}},\ k \in K^{\text{in}},K_s,\ i \in I^{\text{in}},I_k,I^{\text{T6b}},\ j \in J_i,\ n \in N, n \le N^{\max},\ d \in D^{\text{in}}, rk_{ksd} > 0 \tag{26}$$

The order satisfaction constraints in (16)–(23) are written to ensure that all orders for category 1 products are met on-time and with the required amount. Both under and overproduction as well as early and late production are represented with slack variables that are penalized in the objective function. Note that these constraints can be modified to represent different requirements for production, if desired. Constraints (16) try to ensure that each order (k) is met at least one time with an operation type 6 task (i), where task (i) is suitable for order (k) if $i \in I_k$ and is a operation type 6 task for a category 1 product if $i \in I^{\text{T6b}}$. Similarly, constraints (17) enforce the condition that each order (k) for category 1 product state (s) on day (d) can be met with at most $\lceil rk_{ksd} / B_s^{\min} \rceil$ tasks. Constraints (18) and (19) link the delivery of order (k) through task (i) at event point (n) to the beginning of task (i) in any suitable unit (j) at event point (n) so that every category 1 operation type 6 task must be linked to at least one order delivery and vice versa. Thus, constraint (18) enforces that if a binary variable is activated for operation type 6 task (i), then at least one order delivery must be activated. Similarly, constraint (19) ensures that if no binary variables are activated for operation type 6 task (i) at event point (n), then no delivery variables can be activated. Constraints (20) relate the individual order delivery variables to the batch-size of the operation type 6 task used to satisfy the order. If an order (k) is met by task (i) at event point (n) (i. e., $y(i,k,n) = 1$), then at least one

operation type 6 task is active for task (i) at event point (n) and thus at least one $B(i,j,n)$ variable is greater than zero. Constraints (21) and (22) relate the individual order delivery variables to the overall delivery variables used in the material balance constraints.

Constraints (23) and (24) determine the under and overproduction, respectively, of order (k) for state (s) on day (d). Constraints (23) try to enforce the individual order delivery variables to exceed the amount due for order (k) (i. e., rk_{ksd}) where slack variables $sla1(k,s,d)$ are activated in the case of underproduction. Similarly, constraints (24) try to enforce the individual order delivery variables plus any amount of the product state left at the end of the horizon not to exceed the amount due for order (k) where slack variables $sla2(k,s,d)$ are activated in the case of overproduction. Constraints (25) and (26) determine the late and early production, respectively, of order (k) for state (s) on day (d). Constraints (25) try to enforce the finishing time of task (i) used to satisfy order (k) at event point (n) to be less than the due date of order (k) where slack variables $slt1(k,s,d,n)$ are activated in the case of late production. Similarly, constraints (26) try to enforce the finishing time of task (i) used to fulfill order (k) at event point (n) to be greater than the beginning of the day (d) on which the order is due (i. e., $duek(k,s,d)-24$). Otherwise, slack variables $slt2(k,s,d,n)$ are activated indicating early production.

$$tot(s) = stf(s) + \sum_{\substack{n\in N\\ n\le N^{\max}}} \left[D(s,n) + Df(s,n)\right], \quad \forall s \in S^{\text{in}} \tag{27}$$

$$\sum_{\substack{n\in N\\ n\le N^{\max}}} \left[D(s,n) + Df(s,n)\right] + sll(s) \ge dem_s, \quad \forall s \in S^{\text{in}}, S^{\text{cat1}} \tag{28}$$

$$tot(s) + sll(s) \ge dem_s, \quad \forall s \in S^{\text{in}}, S^{\text{cat2}} \tag{29}$$

$$\sum_{i\in I^{\text{in}}, I_s^{\text{p}}} \sum_{j\in J_i} \sum_{\substack{n\in N\\ n\le N^{\max}}} B(i,j,n) + sll^{\text{raw}}(s) \ge dem_s^{\text{raw}}, \quad \forall s \in S^{\text{in}}, S^{\text{rw}},\ s' \in S^{\text{in}}, S^{\text{f}},\ praw_{s's} > 0,\ dem_{s'} > 0,\ dem_s^{\text{raw}} > 0 \tag{30}$$

Constraints (27)–(29) are used to determine the overall underproduction for both category 1 and 2 products in the current time horizon. First, constraints (27) determine the total production for all product states (s) (i. e., $tot(s)$) in the current horizon. Then, constraints (28) sum the overall delivery variables for category 1 products and activate the slack variables $sll(s)$ if the sum does not exceed the demand for category 1 product state (s). Similarly, constraints (29) calculate the amount of underproduction (i. e., $sll(s)$) for category 2 product state (s) based on it's overall demand in the time horizon. The slack variable $sll(s)$ is then penalized in the objective function where category 1 and 2 products can be penalized at different weights. Constraints (30) determine the amount of underproduction for intermediate product states (s) that are needed as raw materials for final product states (s').

The bound constraints are used to impose lower and upper bounds on the continuous variables including slack variables. They are also used to fix some binary and continuous variables to be zero when necessary.

$$T^{\text{f}}(i,j,n) \ge start_j, \quad \forall i \in I, \quad j \in J_i, \quad n \in N$$
$$T^{\text{s}}(i,j,n) \ge start_j, \quad \forall i \in I, \quad j \in J_i, \quad n \in N$$
$$T^{\text{f}}(i,j,n) \le H, \quad \forall i \in I, \quad j \in J_i, \quad n \in N$$
$$T^{\text{s}}(i,j,n) \le H, \quad \forall i \in I, \quad j \in J_i, \quad n \in N$$
$$STO(s) = 0, \quad \forall s \notin S^0$$
$$STF(s) \le dem_s^{\text{tot}}, \quad \forall s \in S^{\text{f}}$$
$$tot(s) \le dem_s^{\text{tot}}, \quad \forall s \in S^{\text{f}}$$
$$D(s,n), Df(s,n) = 0, \quad \forall s \notin S^{\text{p}} \text{ or } n \in N, n > N^{\max}$$
$$D(s,n), Df(s,n) \le \sum_{d\in D^{\text{in}}} \sum_{k\in K^{\text{in}}} rk(k,s,d), \quad \forall s \in S^{\text{p}}, \quad n \in N, n \le N^{\max}$$
$$kD(k,s,n), kDf(k,s,n) = 0, \quad \forall k \notin K^{\text{in}} \text{ or } s \notin S_k \text{ or } n \in N, n > N^{\max}$$
$$kD(k,s,n), kDf(k,s,d) \le \sum_{d\in D^{\text{in}}} rk(k,s,d), \quad \forall s \in S_k, \quad n \in N, \quad n \le N^{\max}$$
$$slcap(s,n) \le stcap_s^{\min}, \quad \forall s \in S^{\text{cpm}}$$
$$sla1(k,s,n), sla2(k,s,n) = 0, \quad \forall k \notin K^{\text{in}} \text{ or } s \notin S_k \text{ or } d \notin D^{\text{in}} \text{ or } rk(k,s,d) = 0$$
$$sla1(k,s,n) \le rk(k,s,d), \quad \forall k \in K^{\text{in}}, \quad s \in S_k, \quad d \in D^{\text{in}}$$

$$
\begin{aligned}
& sla2(k,s,n) \le dem_s^{\text{tot}}, \quad \forall k \in K^{\text{in}}, \\
& \quad s \in S_k, \quad d \in D^{\text{in}} \\
& slt1(k,s,d,n), slt2(k,s,d,n) = 0, \quad \forall k \notin K^{\text{in}} \\
& \quad or\ s \notin S_k\ or\ d \notin D^{\text{in}}\ or \\
& rk(k,s,d) = 0\ or\ n \in N, \quad n > N^{\max} \\
& slt1(k,s,d,n) \le H - duek(k,s,d), \quad \forall k \in K^{\text{in}}, \\
& \quad s \in S_k, \quad d \in D^{\text{in}}, \quad n \in N, \quad n \le N^{\max} \\
& slt2(k,s,d,n) \le duek(k,s,d), \quad \forall k \in K^{\text{in}}, \\
& \quad s \in S_k, \quad d \in D^{\text{in}}, \quad n \in N, \quad n \le N^{\max} \\
& sll(s) \le dem_s, \quad \forall s \in S^{\text{p}} \\
& sll^{\text{raw}}(s) \le dem_s^{\text{raw}}, \quad \forall s \in S^{\text{rw}} \\
& wv(i,j,n), B(i,j,n) = 0, \quad \forall i \notin I^{\text{in}}\ or\ j \notin J_i \\
& \quad or\ n \in N, \quad n > N^{\max} \\
& D(s,n), Df(s,n) = 0, \quad \forall s \notin S^{\text{in}} \\
& \quad or\ n \in N, \quad n > N^{\max}
\end{aligned}
\tag{31}
$$

There are several different objective functions that can be employed with a general short-term scheduling problem. In this work, we maximize the sale of final products while penalizing several other terms including the slack variables introduced previously. The overall objective function is as follows:

$$
\begin{aligned}
Max\ \ & \omega \cdot \sum_{s \in S^{\text{in}}, S^{\text{p}}} price_s \cdot tot(s) \\
& - \lambda \cdot \sum_{i \in I^{\text{in}}} \sum_{j \in J_i} \sum_{\substack{n \in N \\ n \le N^{\max}}} tts(i,j,n) \\
& - \kappa \cdot \left[\sum_{i \in I^{\text{in}}} \sum_{j \in J_i} \sum_{\substack{n \in N \\ n \le N^{\max}}} wv(i,j,n) \right. \\
& \quad \left. + \sum_{k \in K^{\text{in}}} \sum_{i \in I_k} \sum_{\substack{n \in N \\ n \le N^{\max}}} y(i,k,n) \right] \\
& - \gamma \cdot \sum_{s \in S^{\text{f}}} prior_s \cdot sll(s) - \phi \cdot \sum_{k \in K^{\text{in}}} slorder(k) \\
& - \alpha \cdot \left[\sum_{k \in K^{\text{in}}} \sum_{s \in S^{\text{cat1}}} \sum_{d \in D^{\text{in}}} sla1(k,s,d) \right. \\
& \quad \left. + \mu \cdot sla2(k,s,d) \right] \\
& - \beta \cdot \left[\sum_{k \in K^{\text{in}}} \sum_{s \in S^{\text{cat1}}} \sum_{d \in D^{\text{in}}} \sum_{\substack{n \in N \\ n \le N^{\max}}} \mu \cdot slt1(k,s,d,n) \right. \\
& \quad \left. + slt2(k,s,d,n) \right] \\
& - \eta \cdot \sum_{s \in S^{\text{rw}}} prior_s^{\text{raw}} \cdot sll^{\text{raw}}(s) \\
& - \delta \cdot \sum_{s \in S^{\text{cpm}}} \sum_{\substack{n \in N \\ n \le N^{\max}}} slcap(s,n)
\end{aligned}
\tag{32}
$$

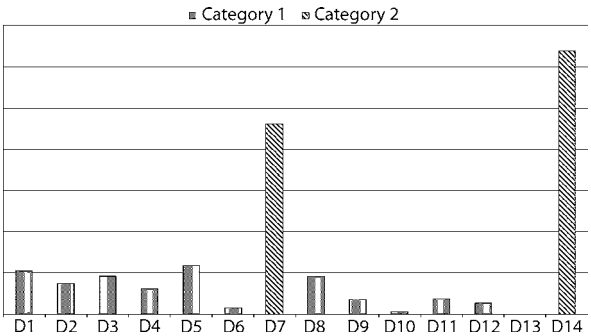
where each of the coefficients is used to balance the relative weight of each term in the overall objective function. The first term is the maximization of the value of the final products and is the main term of the objective function. The second term seeks to minimize the sum of the starting times of all active processing tasks. This is done to encourage all tasks to start as early as possible in the scheduling horizon. Note that this results in a bilinear term which can replaced with an equivalent linear term and set of constraints [3]. The third term seeks to minimize the number of active binary variables in the final production schedule. The fourth term seeks to minimize the slack variable that is activated when product state (s) does not meet its overall demand for the time horizon. Coefficient $prior_s$ allows the ability to assign different weights to different product states. The fifth term minimizes the number of category 1 orders (k) that are not filled in the time horizon. The sixth term minimizes the amount of over and underproduction of orders for category 1 products in the time horizon where the coefficient μ allows over and underproduction to be penalized by different amounts. The seventh term seeks to minimize the amount of early and late production of orders for category 1 products due in the time horizon where the coefficient μ allows early and late production to be penalized to different degrees. The eighth term minimizes the slack variables activated when insufficient raw material state (s) is produced during the time horizon where $prior_s^{\text{raw}}$ allows different states to be penalized by different amounts. The ninth, and final, term seeks to minimize the slack variables activated when insufficient intermediate state (s) is stored in its dedicated storage tank at each event point. Typical

values for each of the coefficients are as follows: $\omega = 1$, $\lambda = 1, \kappa = 10, \gamma = 1000, \phi = 1000, \alpha = 2000$, $\beta = 500, \mu = 0.01, \eta = 50, \delta = 10$.

Cases

In this section, an example problem is presented to demonstrate the effectiveness of the rolling horizon framework. The example utilizes the proposed framework to determine the medium range production schedule of an industrial batch plant for a two-week time period which satisfies customer orders for various products distributed throughout the time period. The example is implemented with GAMS 2.50 [1] and solved using CPLEX 9.0 [2] with a 3.20 GHz Linux workstation. The dual simplex method is used with best-bound search and strong branching. A relative optimality tolerance equal to 0.001% was used as the termination criterion along with a three hour time limit and an integer solution limit of 40.

The distribution of demands for the entire two-week time period is shown in Fig. 3 where the amounts are shown in relative terms. There are two categories of products, category 1 and 2, and a total of 67 different products have demands. There are two different campaign products that can be scheduled for campaign mode production and an additional eight intermediate products are used to make final products, even though they do not have demands. It is assumed that no final products are available at the beginning of the time horizon although some intermediate materials are available. Also, we assume no limitation on external raw materials and the zero-wait condition is applied to all intermediate materials unless they are used as raw materials for other final products. In this case, unlimited intermediate storage is allowed. Note that finite intermediate storage is effectively modeled for those intermediates that have a dedicated storage task with a given capacity limit. In addition, there are two types of connections made between each consecutive short-term scheduling horizon in the rolling horizon framework: the initial available time for each unit and the inventory of intermediate materials.



Medium-Term Scheduling of Batch Processes, Figure 3
Distribution of demands

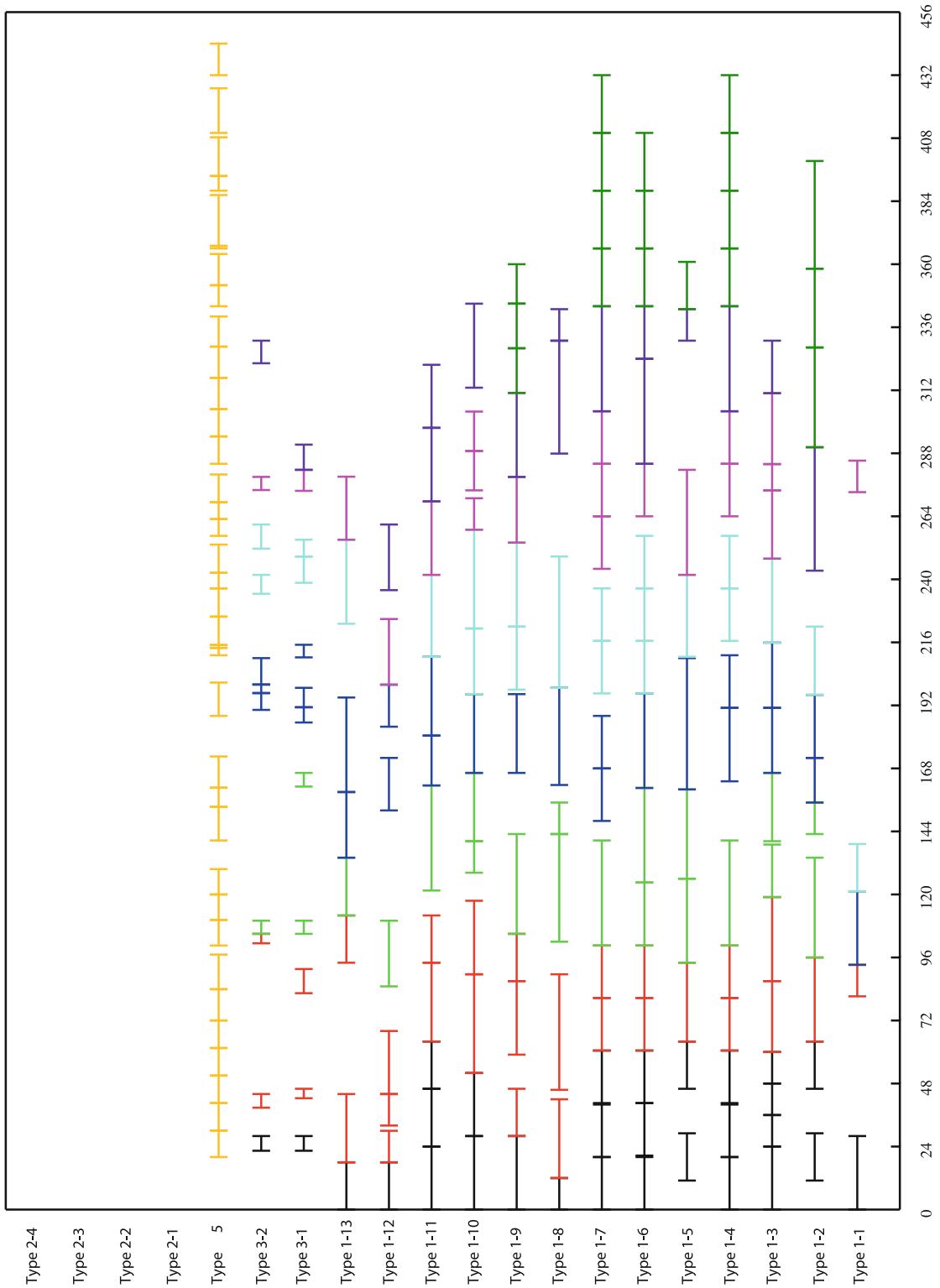
Case 1: Nominal Run without Campaign Mode Production

The example problem considers the production scheduling of an industrial batch plant where no type 5 unit campaign is imposed. Instead, demands for both campaign products are created throughout the time horizon with a total demand for each product equal to the production that would be imposed by a campaign. The total time period is 19 days, from D0 to D18. The rolling horizon framework decomposes the time horizon into 8 individual subhorizons, each with its own products and demands. The results of the decomposition for each time horizon can be seen in Table 1.

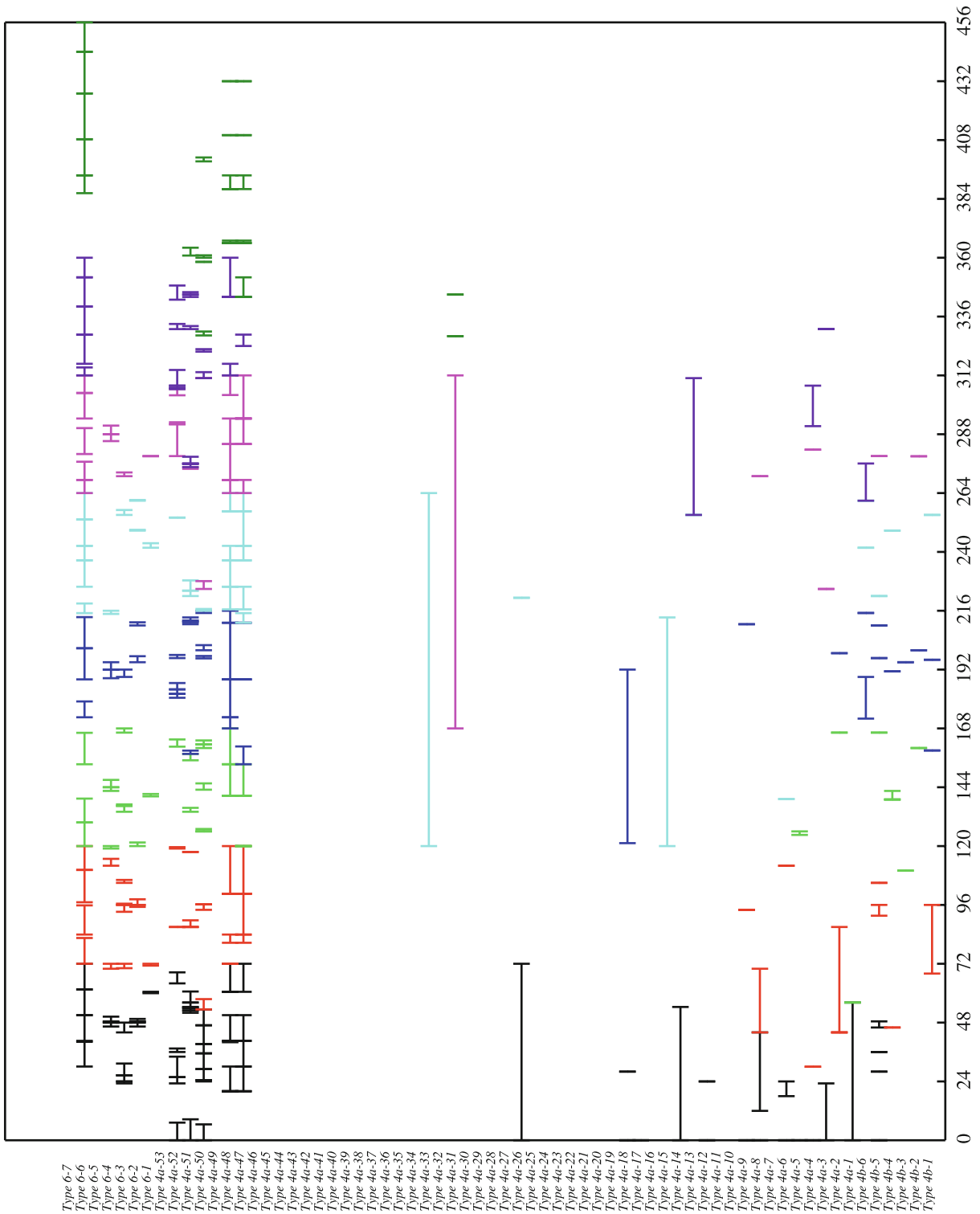
The final production schedule for the entire time period can be seen in Fig. 4 and 5 where the processing units (operation type 1, 2, 3, and 5) are shown in the first figure and the other units (operation type 4a, 4b, and 6) are shown in the second. Each short-term scheduling horizon is represented with a different color beginning with black for the first horizon, red for the second horizon, green for the third horizon, etc. The model and solution statistics for each short-term

Medium-Term Scheduling of Batch Processes, Table 1
Decomposition results for case 1

	Days	Main Products	Additional Products
H1	D0–D2	27	2
H2	D3–D4	31	0
H3	D5–D6	50	0
H4	D7–D8	49	0
H5	D9–D10	37	0
H6	D11–D12	49	0
H7	D13–D14	54	0
H8	D15–D18	45	0



Medium-Term Scheduling of Batch Processes, Figure 4
Overall production schedule for processing units for case 1



Medium-Term Scheduling of Batch Processes, Figure 5
Overall production schedule for non-processing units for case 1

Medium-Term Scheduling of Batch Processes, Table 2
Model and solutions statistics for case 1

	Days	Event Points	Objective Function	Binary Variables	Continuous Variables	Constraints
H1	D0–D2	8	14, 001.69	4880	33,064	187,833
H2	D3–D4	6	4135.24	3660	24,923	125,374
H3	D5–D6	6	−105, 854.81	5478	32,621	258,852
H4	D7–D8	6	−5496.19	5376	32,167	255,696
H5	D9–D10	6	−15, 352.37	4296	27,613	175,939
H6	D11–D12	6	−11, 326.13	5490	32,637	272,802
H7	D13–D14	6	−19, 401.39	5568	32,955	282,632
H8	D15–D18	10	−37, 054.00	7430	46,827	321,162

scheduling horizon can be seen in Table 2 where each horizon runs for the time limit of three hours.

The total demand for the entire 14-day period is 2323.545 and the total production is 2744.005, where 51.674 of the demands are not met. The production schedules obtained satisfy demands for almost all the products, though some due dates are relaxed, and also produce 18.10% more material than the demands require. Many of the processing units are not fully utilized, as shown in Table 3, indicating the potential for even more production in the given time period. Also, note that the processing units become more idle towards the end of the overall time horizon. This is because no demands are specified for the days following day D14 including days D15 to D18. Additional demands at the end of the overall time horizon or in the following days would generate a more heavily utilized production schedule.

Medium-Term Scheduling of Batch Processes, Table 3
Unit utilization statistics for case 1

Unit	Time Used (h)	TimeLeft (h)	Percent Utilized
Type 1–1	98.00	358.00	21.49%
Type 1–2	341.00	115.00	74.78%
Type 1–3	329.60	126.40	72.15%
Type 1–4	396.00	60.00	80.92%
Type 1–5	283.20	172.80	62.06%
Type 1–6	402.00	54.00	88.16%
Type 1–7	408.00	48.00	89.47%
Type 1–8	281.00	175.00	61.62%
Type 1–9	322.00	134.00	70.61%
Type 1–10	322.20	133.80	70.66%
Type 1–11	312.20	143.80	68.46%
Type 1–12	177.00	279.00	38.82%
Type 1–13	201.00	255.00	44.08%
Type 5	362.04	93.96	79.39%

Conclusions

In this paper, a *unit-specific* event-based continuous-time formulation is presented for the medium-term production scheduling of a large-scale, multipurpose industrial batch plant. The proposed formulation takes into account a large number of processing recipes and units and incorporates several features including various storage policies (UIS, NIS, ZW), variable batch sizes and processing times, batch mixing and splitting, sequence-dependent changeover times, intermediate due dates, products used as raw materials, and several modes of operation. The scheduling horizon is several weeks or longer, however longer time periods can be addressed with the proposed framework. A key feature of the proposed formulation is the use of a decomposition model to split the overall scheduling horizon into smaller subhorizons which are scheduled in a sequential fashion. Also, new constraints are added to the short-term scheduling model in order to model the delivery of orders at intermediate due dates. The effectiveness of the proposed approach is demonstrated with an industrial case study. Results indicate that the rolling horizon approach is effective at solving large-scale, medium-term production scheduling problems.

References

1. Brooke A, Kendrick D, Meeraus A, Raman R (2003) GAMS: A User's Guide. South San Franciso
2. CPLEX (2005) ILOG CPLEX 9.0 User's Manual
3. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization. Oxford University Press, Oxford

4. Floudas CA, Lin X (2004) Continuous-Time versus Discrete-Time Approaches for Scheduling of Chemical Processes: A Review. Comput Chem Eng 28:2109
5. Floudas CA, Lin X (2005) Mixed Integer Linear Prograunubg in Process Scheduling: Modeling, Algorithms, and Applications. Ann Oper Res 139:131
6. Ierapetritou MG, Floudas CA (1998) Effective Continuous-Time Formulation for Short-Term Scheduling: 1. Multipurpose Batch Processes. Ind Eng Chem Res 37:4341
7. Janak SL, Lin X, Floudas CA (2004) Enhanced Continuous-Time Unit-Specific Event-Based Formulation for Short-Term Scheduling of Multipurpose Batch Processes: Resource Constraints and Mixed Storage Policies. Ind Eng Chem Res 43:2516
8. Lin X, Floudas CA, Modi S, Juhasz NM (2002) Continuous-Time Optimization Approach for Medium-Range Production Scheduling of a Multiproduct Batch Plant. Ind Eng Chem Res 41:3884
9. Pantelides CC (1993) Unified Frameworks for Optimal Process Planning and Scheduling. In: Rippin DWT, Hale JC, Davis J (eds) Proceedings of the Second International Conference on Foundations of Computer-Aided Process Operations. CACHE. Crested Butte, Austin, pp 253–274
10. Reklaitis GV (1992) Overview of Scheduling and Planning of Batch Process Operations. In: Presented at NATO Advanced Study Institute – Batch Process Systems Engineering. Antalya
11. Shah N (1998) Single- And Multisite Planning and Scheduling: Current Status and Future Challenges. In: Pekny JF, Blau GE (eds) Proceedings of the Third International Conference on Foundations of Computer-Aided Process Operations. CACHE-AIChE. Snowbird, New York, pp 75–90

Metaheuristic Algorithms for the Vehicle Routing Problem

YANNIS MARINAKIS
Department of Production Engineering and Management, Decision Support Systems Laboratory, Technical University of Crete, Chania, Greece

MSC2000: 90B06, 90C59

Article Outline

Introduction
Metaheuristic Algorithms for the Vehicle Routing Problem
References

Introduction

The **vehicle routing problem (VRP)** or the **capacitated vehicle routing problem (CVRP)** is often described as a problem in which vehicles based at a central depot are required to visit geographically dispersed customers in order to fulfill known customer demands. Let $G = (V, E)$ be a graph where $V = \{i_0, i_1, i_2, \ldots i_n\}$ is the vertex set ($i_i = i_0$ refers to the depot and the customers are indexed $i_i = i_1, \ldots, i_n$) and $E = \{(i_l, i_{l_1})\colon i_l, i_{l_1} \in V\}$ is the edge set. Each customer must be assigned to exactly one of the k vehicles and the total size of deliveries for customers assigned to each vehicle must not exceed the vehicle capacity (Q_k). If the vehicles are homogeneous, the capacity for all vehicles is equal and denoted by Q. A demand q_{i_l} and a service time st_{i_l} are associated with each customer node i_l. The travel cost between customers i_l and i_{l_1} is $c_{i_l i_{l_1}}$. The problem is to construct a low cost, feasible set of routes – one for each vehicle. A route is a sequence of locations that a vehicle must visit along with the indication of the service it provides. The vehicle must start and finish its tour at the depot. The most important variants of the vehicle routing problem can be found in [12,13,39,54,84].

The vehicle routing problem was first introduced by Dantzig and Ramser [21]. As it is an NP-hard problem, the instances with a large number of customers cannot be solved in optimality within reasonable time. Due to the general inefficiency of the exact methods and their inability to solve large scale VRP instances, a large number of approximation techniques have been proposed. These techniques are classified into two main categories, the classical heuristics that were developed mostly between 1960 and 1990 and the metaheuristics that were developed in the last fifteen years.

In the 1960s and 1970s the first attempts to solve the vehicle routing problem focused on route building, route improvement and two-phase heuristics. In the 1980s a number of mathematical programming procedures were proposed for the solution of the problem. The most important of them can be found in [6,18,19,22,28,29,33,62,88].

Metaheuristic Algorithms for the Vehicle Routing Problem

The last fifteen years an incremental amount of metaheuristic algorithms have been proposed. Simulated

annealing, genetic algorithms, neural networks, tabu search, ant algorithms, together with a number of hybrid techniques are the main categories of the metaheuristic procedures. These algorithms have the ability to find their way out of local optima. Surveys in metaheuristic algorithms have been published by [27,31,32,49,50,79].

A number of metaheuristic algorithms have been proposed for the solution of the Capacitated Vehicle Routing Problem. The most important algorithms published for each metaheuristic algorithm are given in the following:

- **Simulated Annealing (SA)** [1,3,47,72] plays a special role within local search for two reasons. First, they appear to be quite successful when applied to a broad range of practical problems. Second, some threshold accepting algorithms such as SA have a stochastic component, which facilitates a theoretical analysis of their asymptotic convergence. Simulated Annealing [2] is a stochastic algorithm that allows random uphill jumps in a controlled fashion in order to provide possible escapes from poor local optima. Gradually the probability allowing the objective function value to increase is lowered until no more transformations are possible. Simulated Annealing owes its name to an analogy with the annealing process in condensed matter physics, where a solid is heated to a maximum temperature at which all particles of the solid randomly arrange themselves in the liquid phase, followed by cooling through careful and slow reduction of the temperature until the liquid is frozen with the particles arranged in a highly structured lattice and minimal system energy. This ground state is reachable only if the maximum temperature is sufficiently high and the cooling sufficiently slow. Otherwise a meta-stable state is reached. The meta-stable state is also reached with a process known as quenching, in which the temperature is instantaneously lowered. Its predecessor is the so-called Metropolis filter. Simulated Annealing algorithms for the VRP are presented in [14,31,63].
- **Threshold Accepting Method** is a modification of the Simulated Annealing, which together with **record to record travel** [25,26] are known as **Deterministic Annealing** methods. These methods leave out the stochastic element in accepting worse solutions by introducing a deterministic threshold denoted by $Th_m > 0$, and accept a worse solution if $\sigma = c(S') - c(S) \leq Th_m$, where c is the cost of the solution. This is the move acceptance criterion and the subscript m is an iteration index. Dueck and Scheurer [26] were the first to propose the Threshold Accepting Method for the VRP. Tarantilis et al. [81,82] proposed two very efficient algorithms belonging to this class: the **Backtracking Adaptive Threshold Accepting** (**BATA**) and the **List-Based Threshold Accepting** (**LBTA**). Other Deterministic Annealing methods were proposed by Golden et al. [40], the **Record-to-Record Travel Method** and by Li et al. [51].
- **Tabu search (TS)** was introduced by Glover [34,35] as a general iterative metaheuristic for solving combinatorial optimization problems. Computational experience has shown that TS is a well established approximation technique, which can compete with almost all known techniques and which, by its flexibility, can beat many classic procedures. It is a form of local neighbor search. Each solution S has an associated set of neighbors $N(S)$. A solution $S' \in N(S)$ can be reached from S by an operation called a *move*. TS can be viewed as an iterative technique which explores a set of problem solutions, by repeatedly making moves from one solution S to another solution S' located in the neighborhood $N(S)$ of S [37]. TS moves from a solution to its best admissible neighbor, even if this causes the objective function to deteriorate. To avoid cycling, solutions that have been recently explored are declared *forbidden or tabu* for a number of iterations. The tabu status of a solution is overridden when certain criteria (*aspiration criteria*) are satisfied. Sometimes, *intensification* and *diversification* strategies are used to improve the search. In the first case, the search is accentuated in the promising regions of the feasible domain. In the second case, an attempt is made to consider solutions in a broad area of the search space. Tabu Search algorithms for the VRP are presented in [7,9,20,30,63,70,71,77,85,89,90].
- **Genetic Algorithms (GAs)** are search procedures based on the mechanics of natural selection and natural genetics. The first GA was developed by John H. Holland in the 1960s to allow computers to evolve solutions to difficult search and combinatorial prob-

lems, such as function optimization and machine learning [44]. Genetic algorithms offer a particularly attractive approach for problems like vehicle routing problem since they are generally quite effective for rapid global search of large, non-linear and poorly understood spaces. Moreover, genetic algorithms are very effective in solving large-scale problems. Genetic algorithms [38,72] mimic the evolution process in nature. GAs are based on an imitation of the biological process in which new and better populations among different species are developed during evolution. Thus, unlike most standard heuristics, GAs use information about a population of solutions, called individuals, when they search for better solutions. A GA is a stochastic iterative procedure that maintains the population size constant in each iteration, called a generation. Their basic operation is the mating of two solutions in order to form a new solution. To form a new population, a binary operator called crossover, and a unary operator, called mutation, are applied [65,66]. Crossover takes two individuals, called parents, and produces two new individuals, called offsprings, by swapping parts of the parents. Genetic Algorithms for the VRP are presented in [4,5,8,11,45,56,53,60,64].

- **Greedy Randomized Adaptive Search Procedure – GRASP** [73] is an iterative two phase search method which has gained considerable popularity in combinatorial optimization. Each iteration consists of two phases, a construction phase and a local search procedure. In the construction phase, a randomized greedy function is used to build up an initial solution. This randomized technique provides a feasible solution within each iteration. This solution is then exposed for improvement attempts in the local search phase. The final result is simply the best solution found over all iterations. Greedy Randomized Adaptive Search Procedure algorithms for the VRP are presented in [17,42,55].
- The use of **Artificial Neural Networks** to find good solutions to combinatorial optimization problems has recently caught some attention. A neural network consists of a network [76] of elementary nodes (neurons) that are linked through weighted connections. The nodes represent computational units, which are capable of performing a simple computation, consisting of a summation of the weighted inputs, followed by the addition of a constant called the threshold or bias, and the application of a nonlinear response (activation) function. The result of the computation of a unit constitutes its output. This output is used as an input for the nodes to which it is linked through an outgoing connection. The overall task of the network is to achieve a certain network configuration, for instance a required input–output relation, by means of the collective computation of the nodes. This process is often called *self-organization*. Neural Networks algorithm for the VRP are presented in [61,83].
- The **Ant Colony Optimization (ACO)** metaheuristic is a relatively new technique for solving combinatorial optimization problems (COPs). Based strongly on the Ant System (AS) metaheuristic developed by Dorigo, Maniezzo and Colorni [24], ant colony optimization is derived from the foraging behaviour of real ants in nature. The main idea of ACO is to model the problem as the search for a minimum cost path in a graph. Artificial ants walk through this graph, looking for good paths. Each ant has a rather simple behavior so that it will typically only find rather poor-quality paths on its own. Better paths are found as the emergent result of the global cooperation among ants in the colony. An ACO algorithm consists of a number of cycles (iterations) of solution construction. During each iteration a number of ants (which is a parameter) construct complete solutions using heuristic information and the collected experiences of previous groups of ants. These collected experiences are represented by a digital analogue of trail pheromone which is deposited on the constituent elements of a solution. Small quantities are deposited during the construction phase while larger amounts are deposited at the end of each iteration in proportion to solution quality. Pheromone can be deposited on the components and/or the connections used in a solution depending on the problem. Ant Colony Optimization algorithms for the VRP are presented in [10,15,16,23,57,67,68,69].
- **Path Relinking** This approach generates new solutions by exploring trajectories that connect high-quality solutions – by starting from one of these solutions, called the *starting solution* and generating a path in the neighborhood space that leads

towards the other solution, called the *target solution* [36]. Two new metaheuristic algorithms using the path relinking strategy as a part first of Tabu Search Metaheuristic is proposed in [43] and second as a part of a Particle Swarm Optimization Metaheuristic is proposed in [52].

- **Guided Local Search (GLS)**, originally proposed by Voudouris and Chang [86,87], is a general optimization technique suitable for a wide range of combinatorial optimization problems. The main focus is on the exploitation of problem and search-related information to effectively guide local search heuristics in the vast search spaces of NP-hard optimization problems. This is achieved by augmenting the objective function of the problem to be minimized with a set of penalty terms which are dynamically manipulated during the search process to steer the heuristic to be guided. GLS augments the cost function of the problem to include a set of penalty terms and passes this, instead of the original one, for minimization by the local search procedure. Local search is confined by the penalty terms and focuses attention on promising regions of the search space. Iterative calls are made to local search. Each time local search gets caught in a local minimum, the penalties are modified and local search is called again to minimize the modification cost function. Guided Local Search algorithms for the VRP are presented in [58,59].
- **Particle Swarm Optimization (PSO)** is a population-based swarm intelligence algorithm. It was originally proposed by Kennedy and Eberhart as a simulation of the social behavior of social organisms such as bird flocking and fish schooling [46]. PSO uses the physical movements of the individuals in the swarm and has a flexible and well-balanced mechanism to enhance and adapt to the global and local exploration abilities. The first algorithm for the solution of the Vehicle Routing Problem was proposed by [52].
- One of the most interesting developments that have occurred in the area of TS in recent years is the concept of **Adaptive Memory** developed by Rochat and Taillard [74]. It is, mostly, used in TS, but its applicability is not limited to this type of metaheuristic. An adaptive memory is a pool of good solutions that is dynamically updated throughout the search process. Periodically, some elements of these solutions are extracted from the pool and combined differently to produce new good solutions. Very interesting and efficient algorithms based on the concept of Adaptive Memory have been proposed [74,78,79,80].
- **Variable Neighborhood Search (VNS)** is a metaheuristic for solving combinatorial optimization problems whose basic idea is systematic change of neighborhood within a local search [41]. Variable Neighborhood Search algorithms for the VRP are presented in [48].

References

1. Aarts E, Korst J (1989) Simulated Annealing and Boltzmann Machines – A stochastic Approach to Combinatorial Optimization and Neural Computing. Wiley, Chichester
2. Aarts E, Korst J, Van Laarhoven P (1997) Simulated Annealing. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, Chichester, pp 91–120
3. Aarts E, Ten Eikelder HMM (2002) Simulated Annealing. In: Pardalos PM, Resende MGC (eds) Handbook of Applied Optimization. Oxford University Press, Oxford, pp 209–221
4. Alba E, Dorronsoro B (2004) Solving the Vehicle Routing Problem by Using Cellular Genetic Algorithms, Conference on Evolutionary Computation in Combinatorial Optimization, EvoCOP'04. LNCS, vol 3004, 11–20, Portugal. Springer, Berlin
5. Alba E, Dorronsoro B (2006) Computing Nine New Best-So-Far Solutions for Capacitated VRP with a Cellular Genetic Algorithm. Inform Process Lett 98(6):225–230
6. Altinkemer K, Gavish B (1991) Parallel Savings Based Heuristics for the Delivery Problem. Oper Res 39(3): 456–469
7. Augerat P, Belenguer JM, Benavent E, Corberan A, Naddef D (1998) Separating Capacity Constraints in the CVRP Using Tabu Search. Eur J Oper Res 106(2–3):546–557
8. Baker BM, Ayechew MA (2003) A Genetic Algorithm for the Vehicle Routing Problem. Comput Oper Res 30(5): 787–800
9. Barbarosoglu G, Ozgur D (1999) A Tabu Search Agorithm for the Vehicle Routing Problem. Comput Oper Res 26:255–270
10. Bell JE, McMullen PR (2004) Ant Colony Optimization Techniques for the Vehicle Routing Problem. Adv Eng Inform 18(1):41–48
11. Berger J, Mohamed B (2003) A Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, Chicago, pp 646–656
12. Bodin L, Golden B (1981) Classification in Vehicle Routing and Scheduling. Networks 11:97–108

13. Bodin L, Golden B, Assad A, Ball M (1983) The State of the Art in the Routing and Scheduling of Vehicles and Crews. Comput Oper Res 10:63–212
14. Breedam AV (2001) Comparing Descent Heuristics and Metaheuristics for the Vehicle Routing Problem. Comput Oper Res 28(4):289–315
15. Bullnheimer B, Hartl RF, Strauss C (1997) Applying the Ant System to the Vehicle Routing Problem. Paper presented at 2nd International Conference on Metaheuristics, Sophia-Antipolis, France
16. Bullnheimer B, Hartl RF, Strauss C (1999) An Improved Ant System Algorithm for the Vehicle Routing Problem. Ann Oper Res 89:319–328
17. Chaovalitwongse W, Kim D, Pardalos PM (2003) GRASP with a new local search scheme for Vehicle Routing Problems with Time Windows. J Combin Optim 7(2):179–207
18. Christofides N, Mingozzi A, Toth P (1979) The Vehicle Routing Problem. In: Christofides N, Mingozzi A, Toth P, Sandi C (eds) Combinatorial Optimization. Wiley, Chichester
19. Clarke G, Wright J (1964) Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Oper Res 12:568–581
20. Cordeau JF, Gendreau M, Laporte G, Potvin JY, Semet F (2002) A Guide to Vehicle Routing Heuristics. J Oper Res Soc 53:512–522
21. Dantzig GB, Ramser RH (1959) The Truck Dispatching Problem. Manag Sci 6:80–91
22. Desrochers M, Verhoog TW (1989) A Matching Based Savings Algorithm for the Vehicle Routing Problem. Les Cahiers du GERAD G-89–04, Ecole des Hautes Etudes Commerciales de Montreal
23. Doerner K, Gronalt M, Hartl R, Reimman M, Strauss C, Stummer M (2002) Savings Ants for the Vehicle Routing Problem. In: Cagnoni S (ed) EvoWorkshops02. LNCS, vol 2279. Springer, Berlin, pp 11–20
24. Dorigo M, Stutzle T (2004) Ant Colony Optimization, A Bradford Book. MIT Press, London
25. Dueck G (1993) New Optimization Heuristics: The Great Deluge Algorithm and the Record-To-Record Travel. J Comput Phys 104:86–92
26. Dueck G, Scheurer T (1990) Threshold Accepting: A General Purpose Optimization Algorithm. J Comput Phys 90: 161–175
27. Fisher ML (1995) Vehicle routing. In: Ball MO, Magnanti TL, Momma CL, Nemhauser GL (eds) Network Routing. Handbooks in Operations Research and Management Science. North Holland, Amsterdam 8:1–33
28. Fisher ML, Jaikumar R (1981) A Generalized Assignment Heuristic for Vehicle Routing. Networks 11:109–124
29. Foster BA, Ryan DM (1976) An Integer Programming Approach to the Vehicle Scheduling Problem. Oper Res 27:367–384
30. Gendreau M, Hertz A, Laporte G (1994) A Tabu Search Heuristic for the Vehicle Routing Problem. Manag Sci 40:1276–1290
31. Gendreau M, Laporte G, Potvin J-Y (1997) Vehicle Routing: Modern Heuristics. In: Aarts EHL, Lenstra JK (eds) Local search in Combinatorial Optimization. Wiley, Chichester, pp 311–336
32. Gendreau M, Laporte G, Potvin JY (2002) Metaheuristics for the Capacitated VRP. In: Toth P, Vigo D (eds) The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, pp 129–154
33. Gillett BE, Miller LR (1974) A Heuristic Algorithm for the Vehicle Dispatch Problem. Oper Res 22:240–349
34. Glover F (1989) Tabu Search I. ORSA J Comput 1(3): 190–206
35. Glover F (1990) Tabu Search II. ORSA J Comput 2(1):4–32
36. Glover F, Laguna M, Marti R (2003) Scatter Search and Path Relinking: Advances and Applications. In: Glover F, Kochenberger GA (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 1–36
37. Glover F, Laguna M, Taillard E, de Werra D (eds) (1993) Tabu Search. Baltzer, Basel
38. Goldberg DE (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley, Reading
39. Golden BL, Assad AA (1988) Vehicle Routing: Methods and Studies. North Holland, Amsterdam
40. Golden BL, Wassil E, Kelly J, Chao IM (1998) The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithm, Problem Sets and Computational Results. In: Crainic TG, Laporte G (eds) Fleet Management and Logistics. Kluwer, Boston, pp 33–56
41. Hansen P, Mladenovic N (2001) Variable Neighborhood Search: Principles and Applications. Eur J Oper Res 130: 449–467
42. Hjorring C (1995) The Vehicle Routing Problem and Local Search Metaheuristics, PhD thesis, Department of Engineering Science, University of Auckland
43. Ho SC, Gendreau M (2006) Path Relinking for the Vehicle Routing Problem. J Heuristics 12:55–72
44. Holland JH (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor
45. Jaszkiewicz A, Kominek P (2003) Genetic Local Search with Distance Preserving Recombination Operator for a Vehicle Routing Problem. Eur J Oper Res 151:352–364
46. Kennedy J, Eberhart R (1995) Particle Swarm Optimization. In: Proceedings of (1995) IEEE International Conference on Neural Networks 4:1942–1948
47. Kirkpatrick S, Gelatt CD, Vecchi MP (1982) Optimization by Simulated Annealing. Science 220:671–680
48. Kytojoki J, Nuortio T, Braysy O, Gendreau M (2007) An Efficient Variable Neighborhood Search Heuristic for Very Large Scale Vehicle Routing Problems. Comput Oper Res 34(9):2743–2757
49. Laporte G, Gendreau M, Potvin J-Y, Semet F (2000) Classical and Modern Heuristics for the Vehicle Routing Problem. Int Trans Oper Res 7:285–300
50. Laporte G, Semet F (2002) Classical Heuristics for the Capacitated VRP. In: Toth P, Vigo D (eds) The Vehicle Routing

Problem, Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, pp 109–128
51. Li F, Golden B, Wasil E (2005) Very Large-Scale Vehicle Routing: New Test Problems, Algorithms and Results. Comput Oper Res 32(5):1165–1179
52. Marinakis Y, Marinaki M, Dounias G (2007) Nature Inspired Network Approaches in Management: Network Analysis and Optimization for the Vehicle Routing Problem Using NI-Techniques (submitted in AI Commun)
53. Marinakis Y, Marinaki M, Migdalas A (2006) A Hybrid Genetic – GRASP – ENS Algorithm for the Vehicle Routing Problem (submitted in Asia Pac J Oper Res)
54. Marinakis Y, Migdalas A (2002) Heuristic Solutions of Vehicle Routing Problems in Supply Chain Management. In: Pardalos PM, Migdalas A, Burkard R (eds) Combinatorial and Global Optimization. World Scientific, New Jersey, pp 205–236
55. Marinakis Y, Migdalas A, Pardalos PM (2006) Multiple Phase Neighborhood Search GRASP for the Vehicle Routing Problem (submitted in Comput Manag Sci)
56. Marinakis Y, Migdalas A, Pardalos PM (2007) A New Bilevel Formulation for the Vehicle Routing Problem and a Solution Method Using a Genetic Algorithm. J Global Optim 38:555–580
57. Mazzeo S, Loiseau I (2004) An Ant Colony Algorithm for the Capacitated Vehicle Routing. Electron Notes Discret Math 18:181–186
58. Mester D, Braysy O (2005) Active Guided Evolution Strategies for the Large Scale Vehicle Routing Problems with Time Windows. Comput Oper Res 32:1593–1614
59. Mester D, Braysy O (2007) Active-Guided Evolution Strategies for Large-Scale Capacitated Vehicle Routing Problems. Comput Oper Res 34(10):2964–2975
60. Mester D, Braysy O, Dullaert W (2007) A Multi-Parametric Evolution Strategies Algorithm for Vehicle Routing Problems. Expert Syst Appl 32(2):508–517
61. Modares A, Somhom S, Enkawa T (1999) A Self-Organizing Neural Network Approach for Multiple Traveling Salesman and Vehicle Routing Problems. Int Trans Oper Res 6(6):591–606
62. Mole RH, Jameson SR (1976) A Sequential Route-Building Algorithm Employing a Generalized Savings Criterion. Oper Res Q 27:503–511
63. Osman IH (1993) Metastrategy Simulated Annealing and Tabu Search Algorithms for Combinatorial Optimization Problems. Ann Oper Res 41:421–451
64. Prins C (2004) A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem. Comput Oper Res 31:1985–2002
65. Reeves CR (1995) Genetic Algorithms. In: Reeves CR (ed) Modern Heuristic Techniques for Combinatorial Problems. McGraw-Hill, London, pp 151–196
66. Reeves CR (2003) Genetic Algorithms. In: Glover F, Kochenberger GA (eds) Handbooks of Metaheuristics. Kluwer, Dordrecht, pp 55–82
67. Reimann M, Doerner K, Hartl RF (2003) Analyzing a Unified Ant System for the VRP and Some of Its Variants. In: Cagnoni S et al. (eds) EvoWorkshops 2003. LNCS, vol 2611: 300–310. Springer, Berlin
68. Reimann M, Doerner K, Hartl RF (2004) D-Ants: Savings Based Ants Divide and Conquer the Vehicle Routing Problem. Comput Oper Res 31(4):563–591
69. Reimann M, Stummer M, Doerner K (2002) A Savings Based Ant System for the Vehicle Routing Problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, New York, pp 1317–1326
70. Rego C (1998) A Subpath Ejection Method for the Vehicle Routing Problem. Manag Sci 44:1447–1459
71. Rego C (2001) Node-Ejection Chains for the Vehicle Routing Problem: Sequential and Parallel Algorithms. Parallel Comput 27(3):201–222
72. Rego C, Glover F (2002) Local Search and Metaheuristics. In: Gutin G, Punnen A (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 309–367
73. Resende MGC, Ribeiro CC (2003) Greedy Randomized Adaptive Search Procedures. In: Glover F, Kochenberger GA (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 219–249
74. Rochat Y, Taillard ED (1995) Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. J Heuristics 1:147–167
75. Shi Y, Eberhart R (1998) A Modified Particle Swarm Optimizer. In: Proceedings of (1998) IEEE World Congress on Computational Intelligence, pp 69–73
76. Sodererg B, Peterson C (1997) Artificial Neural Networks. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, Chichester, pp 173–214
77. Taillard ED (1993) Parallel Iterative Search Methods for Vehicle Routing Problems. Networks 23:661–672
78. Taillard ED, Gambardella LM, Gendreau M, Potvin JY (2001) Adaptive Memory Programming: A Unified View of Metaheuristics. Eur J Oper Res 135(1):1–16
79. Tarantilis CD (2005) Solving the Vehicle Routing Problem with Adaptive Memory Programming Methodology. Comput Oper Res 32(9):2309–2327
80. Tarantilis CD, Kiranoudis CT (2002) BoneRoute: An Adaptive Memory-Based Method for Effective Fleet Management. Ann Oper Res 115(1):227–241
81. Tarantilis CD, Kiranoudis CT, Vassiliadis VS (2002) A Backtracking Adaptive Threshold Accepting Metaheuristic Method for the Vehicle Routing Problem. Syst Anal Modeling Simul (SAMS) 42(5):631–644
82. Tarantilis CD, Kiranoudis CT, Vassiliadis VS (2002) A List Based Threshold Accepting Algorithm for the Capacitated Vehicle Routing Problem. Int J Comput Math 79(5): 537–553
83. Torki A, Somhon S, Enkawa T (1997) A Competitive Neural Network Algorithm for Solving Vehicle Routing Problem. Comput Indust Eng 33(3–4):473–476

84. Toth P, Vigo D (2002) The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia
85. Toth P, Vigo D (2003) The Granular Tabu Search (and its Application to the Vehicle Routing Problem). INFORMS J Comput 15(4):333–348
86. Voudouris C, Tsang E (1999) Guided Local Search and its Application to the Travelling Salesman Problem. Eur J Oper Res 113:469–499
87. Voudouris C, Tsang E (2003) Guided Local Search. In: Glover F, Kochenberger GA (eds) Handbooks of Metaheuristics. Kluwer, Dordrecht, pp 185–218
88. Wark P, Holt J (1994) A Repeated Matching Heuristic for the Vehicle Routing Problem. J Oper Res Soc 45:1156–1167
89. Willard JAG (1989) Vehicle Routing Using r-Optimal Tabu Search. Master thesis, The Management School, Imperial College, London
90. Xu J, Kelly JP (1996) A New Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem. Transp Sci 30:379–393

Metaheuristics

STEFAN VOSS
Institute of Information Systems (Wirtschaftsinformatik), University of Hamburg, Hamburg, Germany

MSC2000: 68T20, 90C59, 90C27, 68T99

Article Outline

Keywords and Phrases
Introduction
Definitions
  Local Search
  Metaheuristics
Metaheuristic Methods
  Simple Local Search Based Metaheuristics
  Simulated Annealing
  Tabu Search
  Evolutionary Algorithms
  Swarm Intelligence
  Miscellaneous
General Frames
  Adaptive Memory Programming
  A Pool Template
  Partial Optimization Metaheuristic
    Under Special Intensification Conditions
  Hybrids with Exact Methods
  Optimization Software Libraries
Applications
Conclusions
References

Keywords and Phrases

Heuristics; Metaheuristics; Greedy randomized adaptive search procedure; Pilot method; Variable neighborhood search; Simulated annealing; Tabu search; Genetic algorithm; Evolutionary algorithm; Scatter search; Hybridization; Optimization software library; POPMUSIC; Adaptive memory programming; Pool template

Introduction

Many decision problems in various areas such as business, engineering, economics, and science, including those in manufacturing, location, routing, and scheduling, may be formulated as optimization problems. Owing to the complexity of many of these optimization problems, particularly those of large sizes encountered in most practical settings, exact algorithms often perform very poorly, in some cases taking days or more to find moderately decent, let alone optimal, solutions even to fairly small instances. As a result, heuristic algorithms are conspicuously preferable in practical applications.

As an extension of simple heuristics, a large number of local search approaches have been developed to improve given feasible solutions. The main drawback of these approaches is their inability to continue the search upon becoming trapped in local optima. This leads to consideration of techniques for guiding known heuristics to overcome local optimality. Following this theme metaheuristics have become a most important class of approaches for solving optimization problems. They support managers in decision-making with robust tools that provide high-quality solutions to important applications in reasonable time horizons.

We describe metaheuristics mainly from an operations research perspective. Earlier survey papers on metaheuristics include those of Blum and Roli [14] and Voß [95]. Here we occasionally rely on the latter. The general concepts have not become obsolete, and many changes are mainly based upon an update to most recent references. A handbook on metaheuristics is available describing a great variety of concepts by various authors in a comprehensive manner [44].

Definitions

The basic concept of heuristic search as an aid to problem solving was first introduced in [76]. A *heuristic* is a technique (consisting of a rule or a set of rules) which seeks (and hopefully finds) *good* solutions at a reasonable computational cost. A heuristic is *approximate* in the sense that it provides (hopefully) a good solution for relatively little effort, but it does not guarantee optimality.

Heuristics provide simple means of indicating which among several alternatives seems to be best. That is, "Heuristics are criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. They represent compromises between two requirements: the need to make such criteria simple and, at the same time, the desire to see them discriminate correctly between good and bad choices. A heuristic may be a *rule of thumb* that is used to guide one's action" [73].

Greedy heuristics are simple iterative approaches available for any kind of (e. g., combinatorial) optimization problem. A good characterization is their *myopic* behavior. A greedy heuristic starts with a given feasible or infeasible solution. In each iteration there are a number of alternative choices (*moves*) that can be made to transform the solution. From these alternatives which consist in fixing (or changing) one or more variables, a *greedy choice* is made, i. e., the best alternative according to a given measure is chosen until no such transformations are possible any longer.

Usually, a greedy *construction heuristic* starts with an incomplete solution and completes it stepwise. Savings and dual algorithms follow the same iterative scheme: dual heuristics change an infeasible low-cost solution until reaching feasibility; savings algorithms start with a high-cost solution and realize the highest savings as long as possible. Moreover, in all three cases, once an element has been chosen this decision is (usually) not reversed throughout the algorithm, it is kept.

As each alternative has to be measured, in general we may define some sort of *heuristic measure* (providing, e. g., some priority values or some ranking information) which is iteratively followed until a complete solution is built. Usually this heuristic measure is applied in a greedy fashion.

For heuristics we usually have the distinction between finding initial feasible solutions and improving them. In that sense we first discuss local search before characterizing metaheuristics.

Local Search

The basic principle of local search is to successively alter solutions *locally*. Related transformations are defined by neighborhoods which for a given solution include all solutions that can be reached by one move. That is, neighborhood search usually is assumed to correspond to the process of iteratively moving from one solution to another one by performing some sort of operation. More formally, each solution of a problem has an associated set of neighbors called its *neighborhood*, i. e., solutions that can be obtained by a single operation called transformation or move. Most common ideas for transformations are, e. g., to add or drop some problem-specific individual components. Other options are to exchange two components simultaneously, or to swap them. Furthermore, components may be shifted from a certain position into other positions. All components involved within a specific move are called its elements or attributes.

Moves must be evaluated by some *heuristic measure* to guide the search. Often one uses the implied change of the objective function value, which may provide reasonable information about the (local) advantage of moves. Following a greedy strategy, *steepest descent* (SD) corresponds to selecting and performing in each iteration the best move until the search stops at a local optimum. Obviously, savings algorithms correspond to SD.

As the solution quality of local optima may be unsatisfactory, we need mechanisms which guide the search to overcome local optimality. A simple strategy called iterated local search is to iterate/restart the local search process after a local optimum has been obtained, which requires some perturbation scheme to generate a new initial solution (e. g., performing some random moves). Of course, more structured ways to overcome local optimality may be advantageous.

A general survey of local search can be found in [1] and the references from [2]. A simple template is provided in [90].

Starting in the 1970s (see Lin and Kernighan [66]), a variable way of handling neighborhoods is still a topic within local search. Consider an arbitrary neighborhood structure N, which defines for any solution s a set of neighbor solutions $N_1(s)$ as a neighborhood of depth $d = 1$. In a straightforward way, a neighborhood $N_{d+1}(s)$ of depth $d + 1$ is defined as the set $N_d(s) \cup \{s' | \exists s'' \in N_d(s) : s' \in N_1(s'')\}$. In general, a large d might be unreasonable, as the neighborhood size may grow exponentially. However, depths of two or three may be appropriate. Furthermore, temporarily increasing the neighborhood depth has been found to be a reasonable mechanism to overcome *basins of attraction*, e. g., when a large number of neighbors with equal quality exist.

Large-scale neighborhoods have become an important topic (see, e. g., [5] for a survey), especially when efficient ways are at hand for exploring them. Related research can also be found under various names; see, e. g., [75] for the idea of ejection chains.

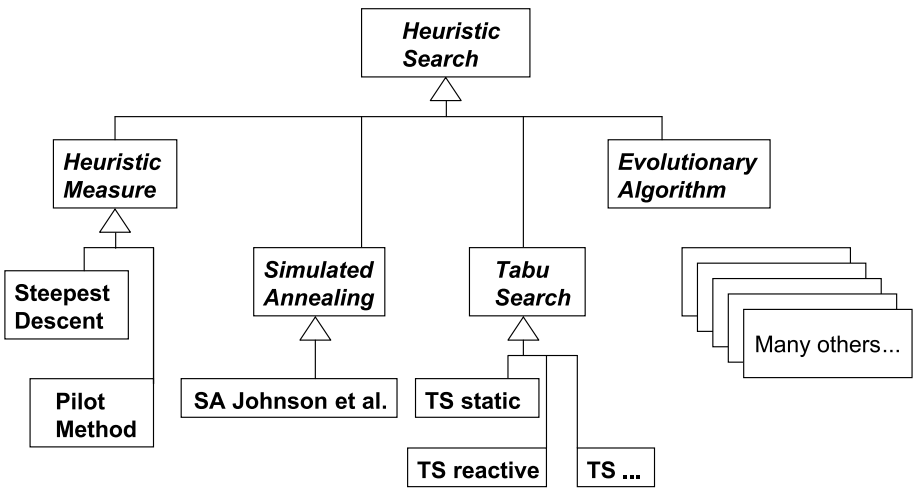
Stochastic local search is pretty much all we know about local search but is enhanced by randomizing choices. That is, a stochastic local search algorithm is a local search algorithm making use of randomized choices in generating or selecting candidate solutions for given instances of optimization problems. Randomness may be used for search initialization as well as the computation of search steps. A comprehensive treatment of stochastic local search is given in [58].

Metaheuristics

The formal definition of metaheuristics is based on a variety of definitions from different authors based on [39]. Basically, a metaheuristic is a top-level strategy that guides an underlying heuristic solving a given problem. In that sense we distinguish between a *guiding process* and an *application process*. The guiding process decides upon possible (local) moves and forwards its decision to the application process, which then executes the move chosen. In addition, it provides information for the guiding process (depending on the requirements of the respective metaheuristic) like the recomputed set of possible moves.

According to [43], "metaheuristics in their modern forms are based on a variety of interpretations of what constitutes *intelligent search*", where the term "intelligent search" has been made prominent by Pearl [73] (regarding heuristics in an artificial intelligence context; see also [92] regarding an operations research context). In that sense we may also consider the following definition: "A metaheuristic is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure information in order to find efficiently near-optimal solutions" [72].

To summarize, the following definition seems to be most appropriate: "A metaheuristic is an iterative



Metaheuristics, Figure 1
Simplified metaheuristics inheritance tree

master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method. The family of metaheuristics includes, but is not limited to, adaptive memory procedures, tabu search, ant systems, greedy randomized adaptive search, variable neighborhood search, evolutionary methods, genetic algorithms, scatter search, neural networks, simulated annealing, and their hybrids" (p. ix in [97]).

We describe the ingredients and basic concepts of various metaheuristic strategies like *tabu search* (TS), *simulated annealing* (SA), and *scatter search*. This is based on a simplified view of a possible inheritance tree for heuristic search methods, illustrating the relationships between some of the most important methods discussed below, as shown in Fig. 1.

We also emphasize advances including the important incorporation of exact methods into intelligent search. Furthermore, general frames are sketched that may subsume various approaches within the metaheuristics field.

Metaheuristic Methods

We survey the basic concepts of some of the most important metaheuristics. We shall see that adaptive processes originating from different settings such as psychology ("learning"), biology ("evolution"), physics ("annealing"), and neurology ("nerve impulses") have served as interesting starting points.

Simple Local Search Based Metaheuristics

To improve the efficiency of greedy heuristics, one may apply generic strategies to be used alone or in combination with each other, namely, changing the definition of alternative choices, look ahead evaluation, candidate lists, and randomized selection criteria bound up with *repetition*, as well as combinations with local search or other methods.

Greedy Randomized Adaptive Search Omitting a greedy choice criterion for a random strategy, one can run the algorithm several times and obtain a large number of different solutions. A combination of best and random choice seems to be appropriate: We define a *candidate list* as a list consisting of a number of (best, i. e., first best, second best, third best, etc.) alternatives. Out of this list one alternative is chosen randomly. The length of the candidate list is given either as an absolute value, a percentage of all feasible alternatives, or implicitly by defining an allowed quality gap (to the best alternative), which also may be an absolute value or a percentage.

Replicating a search procedure to determine a local optimum multiple times with different starting points has been given the acronym GRASP and investigated with respect to different applications. A comprehensive survey of GRASP and its applications is given in [32]. It should be noted that GRASP goes back to older approaches [52], which is frequently overlooked in many applications. The different initial solutions or starting points are found by a greedy procedure incorporating a probabilistic component. That is, given a candidate list to choose from, GRASP randomly chooses one of the best candidates from this list in a greedy fashion, but not necessarily the best possible choice.

The underlying principle is to investigate many good starting points through the greedy procedure and thereby to increase the possibility of finding a good local optimum on at least one replication. The method is said to be adaptive as the greedy function takes into account previous decisions when performing the next choice.

The Pilot Method Building on a simple greedy algorithm such as, e. g., a construction heuristic, the *pilot method* [29,30] is a metaheuristic not necessarily based on a local search in combination with an improvement procedure. It primarily *looks ahead* for each possible local choice (by computing a so-called "pilot" solution), memorizing the best result, and performing the respective move. (Very similar ideas have been investigated under the name *rollout method* [13].) One may apply this strategy by successively performing a greedy heuristic for all possible local steps (i. e., starting with all incomplete solutions resulting from adding some not yet included element at some position to the current incomplete solution). The look ahead mechanism of the pilot method is related to increased neighborhood depths as the pilot method exploits the evaluation

of neighbors at larger depths to guide the neighbor selection at depth one.

In most applications, it is reasonable to restrict the pilot process to some *evaluation depth*. That is, the method is performed up to an incomplete solution (e. g., partial assignment) based on this evaluation depth and is then completed by continuing with a conventional heuristic. For a recent study applying the pilot method to several combinatorial optimization problems obtaining very good results see [96]. Additional applications can be found, e. g., in [18,68].

Variable Neighborhood Search The basic idea of variable neighborhood search (VNS) is to change the neighborhood during the search in a systematic way. VNS usually explores increasingly distant neighborhoods of a given solution, and jumps from this solution to a new one if and only if an improvement has been made. In this way often favorable characteristics of incumbent solutions, e. g., that many variables are already at their appropriate value, will be kept and used to obtain promising neighboring solutions.

Moreover, a local search routine is applied repeatedly to get from these neighboring solutions to local optima. This routine may also use several neighborhoods. Therefore, to construct different neighborhood structures and to perform a systematic search, one needs to have a way for finding the distance between any two solutions, i. e., one needs to supply the solution space with some metric (or quasi-metric) and then induce neighborhoods from it. For an excellent treatment of various aspects of VNS see [51].

Simulated Annealing

Simulated annealing (SA) extends basic local search by allowing moves to inferior solutions [26,64]. A basic SA algorithm may be described as follows: Successively, a candidate move is randomly selected; this move is accepted if it leads to a solution with an improved objective function value compared to the current solution, otherwise, the move is accepted with a probability depending on the deterioration Δ of the objective function value. The acceptance probability is computed as $e^{-\Delta/T}$, using a temperature T as a control parameter. Usually, T is reduced over time for diversification at an earlier stage of the search and to intensify later.

Various authors have described a robust concretization of this general SA approach [60,62]. An interesting variant of SA is to strategically reheat the process, i. e., to perform a nonmonotonic acceptance function.

Threshold accepting [28] is a modification (or simplification) of SA accepting every move that leads to a new solution which is "not much worse" (i. e., deteriorates not more than a certain threshold which reduces with temperature) than the older one.

Tabu Search

The basic paradigm of *tabu search* (TS) is to use information about the search history to guide local search approaches to overcome local optimality (see [43] for a survey on TS). In general, this is done by a dynamic transformation of the local neighborhood. Based on some sort of memory, certain moves may be forbidden; we say they are set tabu. As for SA, the search may lead to performing deteriorating moves when no improving moves exist or when all improving moves of the current neighborhood are set tabu. At each iteration a best admissible neighbor may be selected. A neighbor, or a corresponding move, is called admissible if it is not tabu or if an aspiration criterion is fulfilled. An aspiration criterion is a rule to eventually override a possibly unreasonable tabu status of a move. For example, a move that leads to a neighbor with a better objective function value than encountered so far should be considered as admissible.

We briefly describe some TS methods that differ especially in the way in which tabu criteria are defined, taking into consideration the information about the search history (performed moves, traversed solutions).

The most commonly used TS method is based on a *recency-based* memory that stores moves, or attributes characterizing respective moves, of the recent past (*static TS*). The basic idea of such approaches is to prohibit an appropriately defined inversion of performed moves for a given period. For example, one may store the solution attributes that have been created by a performed move in a tabu list. To obtain the current tabu status of a move to a neighbor, one may check whether (or how many of) the solution attributes that would be destroyed by this move are contained in the tabu list.

Strict TS embodies the idea of preventing cycling to formerly traversed solutions. The goal is to provide ne-

cessity and sufficiency with respect to the idea of not revisiting any solution. Accordingly, a move is classified as tabu if and only if it leads to a neighbor that has already been visited during the previous search. There are two primary mechanisms to accomplish the tabu criterion: First, we may exploit logical interdependencies between the sequence of moves performed throughout the search process, as realized by, e. g., the reverse elimination method and the cancellation sequence method [40,94]. Second, we may store information about all solutions visited so far. This may be carried out either exactly or, for reasons of efficiency, approximately (e. g., by using hash codes).

Reactive TS aims at the automatic adaptation of the tabu list length of static TS [12]. The idea is to increase the tabu list length when the tabu memory indicates that the search is revisiting formerly traversed solutions. A possible specification can be described as follows: Starting with a tabu list length l of 1 it is increased every time a solution has been repeated. If there has been no repetition for some iterations, we decrease it appropriately. To accomplish the detection of a repetition of a solution, one may apply a trajectory-based memory using hash codes as for strict TS.

For reactive TS [12], it is appropriate to include means for diversifying moves whenever the tabu memory indicates that we are trapped in a certain region of the search space. As a trigger mechanism one may use, e. g., the combination of at least two solutions each having been traversed three times. A very simple escape strategy is to perform a number of random moves (depending on the average of the number of iterations between solution repetitions); more advanced strategies may take into account some long-term memory information (like the frequencies of appearance of specific solution attributes in the search history).

Of course there are a great variety of additional ingredients that may make TS work successfully, e. g., restricting the number of neighbor solutions to be evaluated (using candidate list strategies).

Evolutionary Algorithms

Evolutionary algorithms comprise a great variety of different concepts and paradigms, including genetic algorithms (GAs) [45,56], evolutionary strategies [55,83], evolutionary programs [36], scatter search [38,41], and memetic algorithms [71]. For surveys and references on evolutionary algorithms see also [9,37,69,78].

GAs are a class of adaptive search procedures based on principles derived from the dynamics of natural population genetics. One of the most crucial ideas for a successful implementation of a GA is the representation of an underlying problem by a suitable scheme. A GA starts, e. g., with a randomly created initial population of artificial creatures (strings), a set of solutions. These strings in whole and in part are the base set for all subsequent populations. They are copied and information is exchanged between the strings in order to find new solutions of the underlying problem. The mechanisms of a simple GA essentially consist of copying strings and exchanging partial strings. A simple GA uses three operators which are named according to the corresponding biological mechanisms: reproduction, crossover, and mutation. Performing an operator may depend on a *fitness function* or its value (fitness), respectively. As some sort of heuristic measure, this function defines a means of measurement for the profit or the quality of the coded solution for the underlying problem and often depends on the objective function of the given problem.

GAs are closely related to *evolutionary strategies*. Whereas the mutation operator in a GA serves to protect the search from premature loss of information, evolution strategies may incorporate some sort of local search procedure (such as SD) with self-adapting parameters involved in the procedure. On a simplified scale many algorithms may be coined evolutionary once they are reduced to the following frame [54]:

1. Generate an initial population of individuals.
2. While no stopping condition is met do.
3. Co-operation.
4. Self-adaptation.

Self-adaptation refers to the fact that individuals (solutions) evolve independently while co-operation refers to an information exchange among individuals.

Scatter search ideas established a link between early ideas from various sides – evolutionary strategies, TS, and GAs. As an evolutionary approach, scatter search originated from strategies for creating composite decision rules and surrogate constraints [38]. Scatter search is designed to operate on a set of points, called reference points, that constitute good solutions obtained

from previous solution efforts. The approach systematically generates linear combinations of the reference points to create new points, each of which is mapped into an associated point that yields integer values for discrete variables. Scatter search contrasts with other evolutionary procedures, such as GAs, by providing unifying principles for joining solutions based on generalized path constructions in Euclidean space and by utilizing strategic designs where other approaches resort to randomization. For a very comprehensive treatment of scatter search see [65].

Swarm Intelligence

Swarm intelligence is a relatively novel discipline interested in the study of self-organizing processes in nature and human artifacts [15,63]. While researchers in ethnology and animal behavior have proposed many models to explain various aspects of social insect behavior such as self-organization and shape formation, algorithms inspired by these models have been proposed to solve optimization problems. Successful examples are the so-called *ant system* or *ant colony* paradigm, the bee system, and swarm robotics, where the focus is on applying swarm intelligence techniques to the control of large groups of cooperating autonomous robots.

The *ant system* is a dynamic optimization process reflecting the natural interaction between ants searching for food [23]. The ants' ways are influenced by two different kinds of search criteria. The first one is the local visibility of food, i. e., the attractiveness of food in each ant's neighborhood. Additionally, each ant's way through its food space is affected by the other ants' trails as indicators for possibly good directions. The intensity of trails itself is time-dependent: With time passing, parts of the trails are diminishing, while the intensity may increase by new and fresh trails. With the quantities of these trails changing dynamically, an autocatalytic optimization process is started forcing the ants' search into most promising regions. This process of interactive learning can easily be modeled for most kinds of optimization problems by using simultaneously and interactively processed search trajectories.

A comprehensive treatment of the ant system paradigm can be found in [24]. To achieve enhanced performance of the ant system it is useful to hybridize it at least with a local search component.

Miscellaneous

Target analysis may be viewed as a general learning approach. Given a problem we first explore a set of sample instances and an extensive effort is made to obtain a solution which is optimal or close to optimality. The best solutions obtained provide *targets* to be sought within the next part of the approach. For instance, a TS algorithm may resolve the problems with the aim of finding what are the right choices to come to the already known solution (or as close to it as possible). This may give some information on how to choose parameters for other problem instances.

A different method in this context is *path relinking* (PR), which provides a useful means of intensification and diversification. Here new solutions are generated by exploring search trajectories that combine elite solutions, i. e., solutions that have proven to be better than others throughout the search. For references on target analysis and PR see [43].

Recalling local search based on *data perturbation* the *noising method* may be related to the following approach too. Given an initial feasible solution, the method performs some data perturbation [87] in order to change the values taken by the objective function of a respective problem to be solved. On the perturbed data a local search may be performed (e. g., following a SD approach). The amount of data perturbation (the noise added) is successively reduced until it reaches zero. The noising method is applied, e. g., in [19] for the clique partitioning problem.

The key issue in designing *parallel algorithms* is to decompose the execution of the various ingredients of a procedure into processes executable by parallel processors. In contrast to ant systems or GAs, metaheuristics like TS or SA, at first glance, have an intrinsic sequential nature owing to the idea of performing the neighborhood search from one solution to the next. However, some effort has been undertaken to define templates for parallel local search [20,90,91,93]. A comprehensive treatment with successful applications is provided in [6]. The discussion of parallel metaheuristics has also led to interesting hybrids such as the combination of a population of individual processes, agents, in a cooperative and competitive nature (see, e. g., the discussion of *memetic algorithms* in [71]) with TS.

Neural networks may be considered as metaheuristics, although we have not considered them here; see [85] for a comprehensive survey of these techniques for combinatorial optimization. In contrast, one may use metaheuristics to speed up the learning process regarding artificial neural networks; see [7] for a comprehensive consideration.

Furthermore, recent efforts on problems with multiple objectives and corresponding metaheuristic approaches can be found in [61]. See [82] for some ideas regarding GAs and fuzzy multiobjective optimization.

General Frames

An important avenue of metaheuristics research refers to general frames (to explain the behavior and the relationship between various methods) as well as the development of software systems incorporating metaheuristics (eventually in combination with other methods). Besides other aspects, this takes into consideration that in metaheuristics it has very often been appropriate to incorporate a certain means of diversification vs. intensification to lead the search into new regions of the search space. This requires a meaningful mechanism to detect situations when the search might be trapped in a certain area of the solution space. Therefore, within intelligent search the exploration of memory plays a most important role.

Adaptive Memory Programming

Adaptive memory programming (AMP) coins a general approach (or even thinking) within heuristic search focusing on exploiting a collection of memory components [42,89]. An AMP process iteratively constructs (new) solutions based on the exploitation of some memory, especially when combined with learning mechanisms supporting the collection and use of the memory. Based on the idea of initializing the memory and then iteratively generating new solutions (utilizing the given memory) while updating the memory based on the search, we may subsume various of the above-described metaheuristics as AMP approaches. This also includes exploiting provisional solutions that are improved by a local search approach.

The performance as well as the efficiency of a heuristic scheme strongly depends on its ability to use AMP techniques providing flexible and variable strategies for types of problems (or special instances of a given problem type) where standard methods fail. Such AMP techniques could be, e. g., dynamic handling of operational restrictions, dynamic move selection formulas, and flexible function evaluations.

Consider, as an example, adaptive memory within TS concepts. Realizing AMP principles depends on which specific TS application is used. For example, the reverse elimination method observes logical interdependencies between moves and infers corresponding tabu restrictions, and therefore makes fuller use of AMP than simple static approaches do.

To discuss the use of AMP in intelligent agent systems, one may use the simple model of ant systems as an illustrative starting point. Ant systems are based on combining local search criteria with information derived from the trails. This follows the AMP requirement for using flexible (dynamic) move selection rules (formulas). However, the basic ant system exhibits some structural inefficiencies when viewed from the perspective of general intelligent agent systems, as no distinction is made between successful and less successful agents, no time-dependent distinction is made, and there is no explicit handling of restrictions providing protection against cycling and duplication. Furthermore, there are possible conflicts between the information held in the adaptive memory (*diverging trails*).

A Pool Template

In [48] a *pool template* (PT) is proposed as can be seen in Fig. 2. The following notation is used. A pool of $p \geq 1$ solutions is denoted by P. Its input and output transfer is managed by two functions which are called *IF* and *OF*, respectively. S is a set of solutions with cardinality $s \geq 1$. A solution combination method (procedure *SCM*) constructs a solution from a given set S, and *IM* is an improvement method.

Depending on the method used, in step 1 a pool is either completely (or partially) built by a (randomized) diversification generator or filled with a single solution which has been provided, e. g., by a simple greedy approach. Note that a crucial parameter that deserves careful elaboration is the cardinality p of the pool. The main loop, executed until a termination criterion holds, consists of steps 2–5. Step 2 is the call of the output

1. Initialize P by an external procedure
WHILE termination=FALSE DO BEGIN
2. $S := OF(P)$
3. IF $s > 1$ THEN $S' := SCM(S)$
ELSE $S' := S$
4. $S'' := IM(S')$
5. $P := IF(S'')$
END
6. Apply a post-optimizing procedure to P

Metaheuristics, Figure 2
Pool template

function which selects a set of solutions, S, from the pool. Depending on the kind of method represented in the PT, these solutions may be assembled (step 3) to a working solution S' which is the starting point for the improvement phase of step 4. The outcome of the improvement phase, S'', is then evaluated by means of the input function, which possibly feeds the new solution into the pool. Note that a post-optimization procedure in step 6 is for facultative use. It may be a straightforward greedy improvement procedure if used for single-solution heuristics or a pool method on its own. As an example we quote a *sequential* pool method, the TS with PR in [11]. Here a PR phase is added *after* the pool has been initialized by a TS. A *parallel* pool method on the other hand uses a pool of solutions *while* it is constructed by the guiding process (e. g., a GA or scatter search).

Several heuristic and metaheuristic paradigms, whether they are obviously pool-oriented or not, can be summarized under the common PT frame. We provide the following examples:

a) Local search/SD: PT with $p = s = 1$.
b) SA: $p = 2, s = 1$ incorporating its probabilistic acceptance criterion in *IM*. (It should be noted that $p = 2$ and $s = 1$ seems to be unusual at first glance. For SA we always have a current solution in the pool for which one or more neighbors are evaluated and eventually a neighbor is found which replaces the current solution. Furthermore, at all iterations throughout the search the so far best solution is stored too (even if no real interaction between those two stored solutions takes place). The same is also valid for a simple TS. As for local search the current solution corresponds to the best solution of the specific search, we have $p = 1$.)
c) Standard TS: $p = 2, s = 1$ incorporating adaptive memory in *IM*.
d) GAs: $p > 1$ and $s > 1$ with population mechanism (crossover, reproduction, and mutation) in *SCM* of step 3 and without the use of step 4.
e) Scatter search: $p > 1$ and $s > 1$ with subset generation in *OF* of step 2, linear combination of elite solutions by means of *SCM* in step 3, e. g., a TS for procedure *IM* and a reference set update method in *IF* of step 5.
f) PR (as a parallel pool method): $p > 1$ and $s = 2$ with a PR neighborhood in *SCM*. Facultative use of step 4.

Partial Optimization Metaheuristic Under Special Intensification Conditions

A natural way to solve large optimization problems is to decompose them into independent subproblems that are solved with an appropriate procedure. However, such approaches may lead to solutions of moderate quality since the subproblems might have been created in a somewhat arbitrary fashion. Of course, it is not easy to find an appropriate way to decompose a problem a priori. The basic idea of POPMUSIC conditions is to locally optimize subparts of a solution, a posteriori, once a solution to the problem is available. These local optimizations are repeated until a local optimum is found. Therefore, POPMUSIC may be viewed as a local search working with a special, large neighborhood. While the acronym POPMUSIC was given by Taillard and Voß [88] other metaheuristics may be incorporated into the same framework too [84].

For large optimization problems, it is often possible to see the solutions as composed of parts (or chunks [102], cf. the term "vocabulary building"). Considering the vehicle routing problem, a part may be a tour (or even a customer). Suppose that a solution can be represented as a set of parts. Moreover, some parts are more in relation with some other parts, so a corresponding heuristic measure can be defined between two parts. The central idea of POPMUSIC is to select a so-called *seed part* and a set P of parts that are mostly related to the seed part to form a subproblem.

Then it is possible to state a local search optimization frame that consists of trying to improve all sub-

problems that can be defined, until the solution does not contain a subproblem that can be improved. In the POPMUSIC frame of [88], *P* corresponds precisely to seed parts that have been used to define subproblems that have been unsuccessfully optimized. Once *P* contains all the parts of the complete solution, all subproblems have been examined without success and the process stops.

Basically, the technique is a gradient method that starts from a given initial solution and stops in a local optimum relative to a large neighborhood structure. To summarize, both POPMUSIC as well as AMP may serve as a general frame encompassing various other approaches.

Hybrids with Exact Methods

Often a new idea or a new paradigm in metaheuristics is claimed to be *the* idea by the inventor, while others see it as useless in the first instance. However, once it has been hybridized, things begin to *fly*. Especially in population-based metaheuristics, many researchers have followed this trend. That is, we now see many hybrid approaches where the successful ingredients of various metaheuristics have been combined. The term "hybridization", however, goes further, as it also refers to combining metaheuristics with exact methods.

Traditionally, the structure of neighborhoods is determined by local transformations or moves. This usually refers to relatively small homogeneous neighborhoods. Different types of moves have been used in the construction of very large and diverse neighborhoods. In contrast, as a hybrid one may deploy neighborhoods that are *method-based*. By this we mean that the basic structure of a neighborhood is determined by the needs and requirements of a given (say, exact) optimization method used to search the neighborhood. That is, given an incumbent solution one may define the neighborhood so that an exact method can be efficiently used rather than defining a neighborhood and trying to find an appropriate method to explore it. This approach was called *corridor method* by Sniedovich and Voß [86] as it literally defines a neighborhood as a sufficiently sized corridor around a given solution so that a given exact method behaves well. Iteratively the corridor is moved through the search space for exploration.

Constraint programming (CP) is a paradigm for representing and solving a wide variety of problems expressed by means of variables, their domains, and constraints on the variables. Usually CP models are solved using depth-first search and branch and bound. Naturally, these concepts can be complemented by local search concepts and metaheuristics. This idea is followed by several authors; see [21] for TS and guided local search hybrids. Commonalities with the POPMUSIC approach can be deduced from [74].

Of course, the treatment of this topic is by no means complete and various ideas have been developed. One idea is to transform a greedy heuristic into a search algorithm by branching only in a few (i. e., limited number) cases when the choice criterion of the heuristic observes some borderline case or where the choice is least compelling, respectively. This approach may be called *limited discrepancy search* [17,53].

Independent from the CP concept, one may investigate hybrids of branch and bound and metaheuristics, e. g., for deciding upon branching variables or search paths to be followed within a branch and bound tree (see [103] for an application of reactive TS). Here we may also use the term "cooperative solver." Somewhat related is the local branching concept for solving mixed integer programs (MIP), which seeks to explore neighborhoods defined through (invalid) linear cuts. The neighborhoods are searched by means of a general purpose MIP solver [35].

Correspondingly, one of the current research issues refers to exploiting mathematical programming (MP) techniques in a (meta)heuristic framework or, correspondingly, granting to MP approaches the cross-problem robustness and computation time effectiveness which characterize metaheuristics. Discriminating landmark is some form of exploitation of a MP formulation, e. g., by means of MIP. In this respect various efforts have been made towards developing strategies for making a heuristic sequence of roundings to obtain feasible solutions for problems represented by means of appropriate MIP [3,34].

Optimization Software Libraries

Besides some well-known approaches for reusable software in the field of exact optimization (e. g., CPLEX or ABACUS; see http://www.ilog.com and http://www.

informatik.uni-koeln.de/abacus) some ready-to-use and well-documented component libraries in the field of local search based heuristics and metaheuristics have been developed; see especially the contributions in [98].

The most successful approaches documented in the literature are the heuristic optimization framework HOTFRAME of [33] and EASYLOCAL++ of [22]. HOTFRAME, as an example, is implemented in C++, which provides adaptable components incorporating different metaheuristics and an architectural description of the collaboration among these components and problem-specific complements. Typical application-specific concepts are treated as objects or classes: problems, solutions, neighbors, solution attributes, and move attributes. On the other hand, metaheuristic concepts such as the different methods described above and their building blocks such as tabu criteria or diversification strategies are also treated as objects. HOTFRAME uses genericity as the primary mechanism to make these objects adaptable. That is, common behavior of metaheuristics is factored out and grouped in generic classes, applying static type variation. Metaheuristics template classes are parameterized by aspects such as solution spaces and neighborhood structures.

Applications

Applications of metaheuristics are almost uncountable and appear in various journals (e. g., *Journal of Heuristics*), books, and technical reports every day. A helpful source for a subset of successful applications may be special issues of journals or compilations such as [25,77,79,97], just to mention some.

Specialized conferences like the Metaheuristics International Conference are devoted to the topic [25,59,72,80,81,97] and even more general conferences reveal that metaheuristics have become part of necessary prerequisites for successfully solving optimization problems [46]. Moreover, ready-to-use systems such as class libraries and frameworks have been developed, although they are usually restricted to application by the *knowledgeable* user.

Specialized applications also reveal research needs, e. g., in dynamic environments. One example refers to the application of metaheuristics for online optimization [49].

Conclusions

Over the last few decades metaheuristics have become a substantial part of the optimization stockroom with various applications in science and, even more important, in practice. Metaheuristics have been considered in textbooks, e. g., in operations research, and a wealth of monographs [27,43,70,92] are available. Most important in our view are general frames. AMP, an intelligent interplay of intensification and diversification (such as ideas from POPMUSIC), and the connection to powerful exact algorithms as subroutines for handable subproblems are avenues to be followed.

From a theoretical point of view, the use of most metaheuristics has not yet been fully justified. While convergence results regarding solution quality exist for most metaheuristics, once appropriate probabilistic assumptions are made [8,31,50] these turn out not to be very helpful in practice as usually a disproportionate computation time is required to achieve these results (usually convergence is achieved for a computation time tending to infinity, with a few exceptions, e. g., for the reverse elimination method within TS or the pilot method where optimality can be achieved with a finite, but exponential number of steps in the worst case). Furthermore, we have to admit that theoretically one may argue that none of the metaheuristics described are *on average* better than any other; there is *no free lunch* [101]. Basically this leaves the choice of a best possible heuristic or related ingredients to the ingenuity of the user/researcher. Some researchers related the term "hyperheuristics" to the question of which (heuristic) method among a given set of methods to choose for a given problem [16].

Moreover, despite the widespread success of various metaheuristics, researchers occasionally still have a poor understanding of many key theoretical aspects of these algorithms, including models of the high-level run-time dynamics and identification of search space features that influence problem difficulty [99]. Moreover, fitness landscape evaluations are considered to be in their infancy too.

From an empirical standpoint it would be most interesting to know which algorithms perform best under various criteria for different classes of problems. Unfortunately, this theme is out of reach as long as we do not

have any well-accepted standards regarding the testing and comparison of different methods.

While most papers on metaheuristics claim to provide "high-quality" results based on some sort of measure, we still believe that there is a great deal of room for improvement in testing existing as well as new approaches from an empirical point of view [10,57,67]. In a dynamic research process numerical results provide the basis for systematically developing efficient algorithms. The essential conclusions of finished research and development processes should always be substantiated (i. e., empirically and, if necessary, statistically proven) by numerical results based on an appropriate empirical test cycle. Furthermore, even when excellent numerical results are obtained, it may still be possible to compare with a simple random restart procedure and obtain better results in some cases [47]. However, this comparison is usually neglected.

Usually the ways of preparing, performing, and presenting experiments and their results are significantly different. The failing of a generally accepted standard for testing and reporting on the testing, or at least a corresponding guideline for designing experiments, unfortunately implies the following observation: Some results can be used only in a restricted way, e. g., because relevant data are missing, wrong environmental settings are used, or simply results are glossed over. In the worst case nonsufficiently prepared experiments provide results that are unfit for further use, i. e., any generalized conclusion is out of reach. Future algorithm research needs to provide effective methods for analyzing the performance of, e. g., heuristics in a more scientifically founded way (see [4,100] for some steps into this direction).

A final aspect that deserves special consideration is to investigate the use of information within different metaheuristics. While the AMP frame provides a very good entry into this area, this still provides an interesting opportunity to link artificial intelligence with operations research concepts.

References

1. Aarts EHL, Lenstra JK (eds) (1997) Local Search in Combinatorial Optimization. Wiley, Chichester
2. Aarts EHL, Verhoeven M (1997) Local search. In: Dell'Amico M, Maffioli F, Martello S (eds) Annotated Bibliographies in Combinatorial Optimization. Wiley, Chichester, pp 163–180
3. Achterberg T, Berthold T (2007) Improving the feasibility pump. Discret Optim 4:77–86
4. Adenso-Diaz B, Laguna M (2006) Fine-tuning of algorithms using fractional experimental designs and local search. Oper Res 54:99–114
5. Ahuja RK, Ergun O, Orlin JB, Punnen AB (2002) A survey of very large-scale neighborhood search techniques. Discret Appl Math 123:75–102
6. Alba E (ed) (2005) Parallel Metaheuristics. Wiley, Hoboken
7. Alba E, Marti R (eds) (2006) Metaheuristic Procedures for Training Neural Networks. Springer, New York
8. Althöfer I, Koschnick KU (1991) On the convergence of 'threshold accepting'. Appl Math Optim 24:183–195
9. Bäck T, Fogel DB, Michalewicz Z (eds) (1997) Handbook of Evolutionary Computation. Institute of Physics Publishing, Bristol
10. Barr RS, Golden BL, Kelly JP, Resende MGC, Stewart WR (1995) Designing and reporting on computational experiments with heuristic methods. J Heuristics 1:9–32
11. Bastos MB, Ribeiro CC (2002) Reactive tabu search with path relinking for the Steiner problem in graphs. In: Ribeiro CC, Hansen P (eds) Essays and Surveys in Metaheuristics. Kluwer, Boston, pp 39–58
12. Battiti R, Tecchiolli G (1994) The reactive tabu search. ORSA J Comput 6:126–140
13. Bertsekas DP, Tsitsiklis JN, Wu C (1997) Rollout algorithms for combinatorial optimization. J Heuristics 3:245–262
14. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: Overview conceptual comparison. ACM Comput Surv 35:268–308
15. Bonabeau E, Dorigo M, Theraulaz G (eds) (1999) Swarm Intelligence – From Natural to Artificial Systems. Oxford University Press, New York
16. Burke EK, Kendall G, Newall J, Hart E, Ross P, Schulenburg S (2003) Hyper-heuristics: An emerging direction in modern search technology. In: Glover FW, Kochenberger GA (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 457–474
17. Caseau Y, Laburthe F, Silverstein G (1999) A metaheuristic factory for vehicle routing problems. Lect Notes Comput Sci 1713:144–158
18. Cerulli R, Fink A, Gentili M, Voß S (2006) Extensions of the minimum labelling spanning tree problem. J Telecommun Inf Technol 4/2006:39–45
19. Charon I, Hudry O (1993) The noising method: A new method for combinatorial optimization. Oper Res Lett 14:133–137
20. Crainic TG, Toulouse M, Gendreau M (1997) Toward a taxonomy of parallel tabu search heuristics. INFORMS J Comput 9:61–72
21. de Backer B, Furnon V, Shaw P, Kilby P, Prosser P (2000) Solving vehicle routing problems using constraint programming and metaheuristics. J Heuristics 6:501–523

22. Di Gaspero L, Schaerf A (2003) EASYLOCAL++: An object-oriented framework for the flexible design of local-search algorithms. Softw Pr Experience 33:733–765
23. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: Optimization by a colony of cooperating agents. IEEE Trans Syst, Man Cybern B 26:29–41
24. Dorigo M, Stützle T (2004) Ant Colony Optimization. MIT Press, Cambridge
25. Dörner KF, Gendreau M, Greistorfer P, Gutjahr WJ, Hartl RF, Reimann M (eds) (2007) Metaheuristics: Progress in Complex Systems Optimization. Springer, New York
26. Dowsland KA (1993) Simulated annealing. In: Reeves C (ed) Modern Heuristic Techniques for Combinatorial Problems. Halsted, Blackwell, pp 20–69
27. Dreo J, Petrowski A, Siarry P, Taillard E (2006) Metaheuristics for Hard Optimization. Springer, Berlin
28. Dueck G, Scheuer T (1990) Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J Comput Phys 90:161–175
29. Duin CW, Voß S (1994) Steiner tree heuristics – a survey. In: Dyckhoff H, Derigs U, Salomon M, Tijms HC (eds) Operations Research Proceedings. Springer, Berlin, pp 485–496
30. Duin CW, Voß S (1999) The pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs. Netw 34:181–191
31. Faigle U, Kern W (1992) Some convergence results for probabilistic tabu search. ORSA J Comput 4:32–37
32. Festa P, Resende MGC (2004) An annotated bibliography of GRASP. Technical report, AT&T Labs Research, Florham Park
33. Fink A, Voß S (2002) HotFrame: A heuristic optimization framework. In: Voß S, Woodruff DL (eds) Optimization Software Class Libraries. Kluwer, Boston, pp 81–154
34. Fischetti M, Glover F, Lodi A (2005) The feasibility pump. Math Program A 104:91–104
35. Fischetti M, Lodi A (2003) Local branching. Math Program B 98:23–47
36. Fogel DB (1993) On the philosophical differences between evolutionary algorithms and genetic algorithms. In: Fogel DB, Atmar W (eds) Proceedings of the Second Annual Conference on Evolutionary Programming, Evolutionary Programming Society, La Jolla, pp 23–29
37. Fogel DB (1995) Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, New York
38. Glover F (1977) Heuristics for integer programming using surrogate constraints. Decis Sci 8:156–166
39. Glover F (1986) Future paths for integer programming links to artificial intelligence. Comput Oper Res 13:533–549
40. Glover F (1990) Tabu search – Part II. ORSA J Comput 2:4–32
41. Glover F (1995) Scatter search and star-paths: beyond the genetic metaphor. OR Spektrum 17:125–137
42. Glover F (1997) Tabu search and adaptive memory programming – Advances, applications challenges. In: Barr RS, Helgason RV, Kennington JL (eds) Interfaces in computer science and operations research: Advances in metaheuristics, optimization and stochastic modeling technologies. Kluwer, Boston, pp 1–75
43. Glover F, Laguna M (1997) Tabu Search. Kluwer, Boston
44. Glover FW, Kochenberger GA (eds) (2003) Handbook of Metaheuristics. Kluwer, Boston
45. Goldberg DE (1989) Genetic Algorithms in Search, Optimization, Machine Learning. Addison-Wesley, Reading
46. Golden BL, Raghavan S, Wasil EA (eds) (2005) The Next Wave in Computing, Optimization, Decision Technologies. Kluwer, Boston
47. Gomes AM, Oliveira JF (2006) Solving irregular strip packing problems by hybridising simulated annealing and linear programming. Eur J Oper Res 171:811–829
48. Greistorfer P, Voß S (2005) Controlled pool maintenance for meta-heuristics. In: Rego C, Alidaee B (eds) Metaheuristic optimization via memory evolution. Kluwer, Boston, pp 387–424
49. Gutenschwager K, Niklaus C, Voß S (2004) Dispatching of an electric monorail system: Applying meta-heuristics to an online pickup and delivery problem. Transp Sci 38:434–446
50. Hajek B (1988) Cooling schedules for optimal annealing. Math Oper Res 13:311–329
51. Hansen P, Mladenović N (1999) An introduction to variable neighborhood search. In: Voß S, Martello S, Osman IH, Roucairol C (eds) Meta-heuristics: Advances and trends in local search paradigms for optimization. Kluwer, Boston, pp 433–458
52. Hart JP, Shogan AW (1987) Semi-greedy heuristics: An empirical study. Oper Res Lett 6:107–114
53. Harvey W, Ginsberg M (1995) Limited discrepancy search. In: Proceedings of the 14th IJCAI. Morgan Kaufmann, San Mateo, pp 607–615
54. Hertz A, Kobler D (2000) A framework for the description of evolutionary algorithms. Eur J Oper Res 126:1–12
55. Hoffmeister F, Bäck T (1991) Genetic algorithms and evolution strategies: Similarities and differences. Lect Notes Comput Sci 496:455–469
56. Holland JH (1975) Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor
57. Hooker JN (1995) Testing heuristics: We have it all wrong. J Heuristics 1:33–42
58. Hoos HH, Stützle T (2005) Stochastic Local Search – Foundations and Applications. Elsevier, Amsterdam
59. Ibaraki T, Nonobe K, Yagiura M (eds) (2005) Metaheuristics: Progress as Real Problem Solvers. Springer, New York
60. Ingber L (1996) Adaptive simulated annealing (ASA): Lessons learned. Control Cybern 25:33–54
61. Jaszkiewicz A (2004) A comparative study of multiple-objective metaheuristics on the bi-objective set covering

problem and the pareto memetic algorithm. Ann Oper Res 131:215–235
62. Johnson DS, Aragon CR, McGeoch LA, Schevon C (1989) Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. Oper Res 37:865–892
63. Kennedy J, Eberhart RC (2001) Swarm Intelligence. Elsevier, Amsterdam
64. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
65. Laguna M, Martí R (2003) Scatter Search. Kluwer, Boston
66. Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. Oper Res 21:498–516
67. McGeoch C (1996) Toward an experimental method for algorithm simulation. INFORMS J Comput 8:1–15
68. Meloni C, Pacciarelli D, Pranzo M (2004) A rollout metaheuristic for job shop scheduling problems. Ann Oper Res 131:215–235
69. Michalewicz Z (1999) Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Berlin
70. Michalewicz Z, Fogel DB (2004) How to Solve It: Modern Heuristics, 2nd edn. Springer, Berlin
71. Moscato P (1993) An introduction to population approaches for optimization and hierarchical objective functions: A discussion on the role of tabu search. Ann Oper Res 41:85–121
72. Osman IH, Kelly JP (eds) (1996) Meta-Heuristics: Theory and Applications. Kluwer, Boston
73. Pearl J (1984) Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, Reading
74. Pesant G, Gendreau M (1999) A constraint programming framework for local search methods. J Heuristics 5:255–279
75. Pesch E, Glover F (1997) TSP ejection chains. Discret Appl Math 76:165–182
76. Polya G (1945) How to solve it. Princeton University Press, Princeton
77. Rayward-Smith VJ, Osman IH, Reeves CR, Smith GD (eds) (1996) Modern Heuristic Search Methods. Wiley, Chichester
78. Reeves CR, Rowe JE (2002) Genetic Algorithms: Principles and Perspectives. Kluwer, Boston
79. Rego C, Alidaee B (eds) (2005) Metaheuristic optimization via memory and evolution. Kluwer, Boston
80. Resende MGC, de Sousa JP (eds) (2004) Metaheuristics: Computer Decision-Making. Kluwer, Boston
81. Ribeiro CC, Hansen P (eds) (2002) Essays and Surveys in Metaheuristics. Kluwer, Boston
82. Sakawa M (2001) Genetic algorithms and fuzzy multiobjective optimization. Kluwer, Boston
83. Schwefel HP, Bäck T (1998) Artificial evolution: How and why? In: Quagliarella D, Périaux J, Poloni C, Winter G (eds) Genetic Algorithms and Evolution Strategy in Engineering and Computer Science: Recent Advances and Industrial Applications, Wiley, Chichester, pp 1–19
84. Shaw P (1998) Using constraint programming local search methods to solve vehicle routing problems. Working paper, ILOG SA, Gentilly
85. Smith K (1999) Neural networks for combinatorial optimisation: A review of more than a decade of research. INFORMS J Comput 11:15–34
86. Sniedovich M, Voß S (2006) The corridor method: A dynamic programming inspired metaheuristic. Control Cybern 35:551–578
87. Storer RH, Wu SD, Vaccari R (1995) Problem and heuristic space search strategies for job shop scheduling. ORSA J Comput 7:453–467
88. Taillard E, Voß S (2002) POPMUSIC - partial optimization metaheuristic under special intensification conditions. In: Ribeiro CC, Hansen P (eds) Essays and Surveys in Metaheuristics. Kluwer, Boston, pp 613–629
89. Taillard ÉD, Gambardella LM, Gendreau M, Potvin JY (2001) Adaptive memory programming: A unified view of meta-heuristics. Eur J Oper Res 135:1–16
90. Vaessens RJM, Aarts EHL, Lenstra JK (1998) A local search template. Comput Oper Res 25:969–979
91. Verhoeven MGA, Aarts EHL (1995) Parallel local search techniques. J Heuristics 1:43–65
92. Voß S (1993) Intelligent Search. Manuscript, TU Darmstadt
93. Voß S (1993) Tabu search: applications and prospects. In: Du DZ, Pardalos P (eds) Network Optimization Problems. World Scientific, Singapore, pp 333–353
94. Voß S (1996) Observing logical interdependencies in tabu search: Methods and results. In: Rayward-Smith VJ, Osman IH, Reeves CR, Smith GD (eds) Modern Heuristic Search Methods. Wiley, Chichester, pp 41–59
95. Voß S (2001) Meta-heuristics: The state of the art. Lect Notes Artif Intell 2148:1–23
96. Voß S, Fink A, Duin C (2004) Looking ahead with the pilot method. Ann Oper Res 136:285–302
97. Voß S, Martello S, Osman IH, Roucairol C (eds) (1999) Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer, Boston
98. Voß S, Woodruff DL (eds) (2002) Optimization Software Class Libraries. Kluwer, Boston
99. Watson JP, Whitley LD, Howe AE (2005) Linking search space structure, run-time dynamics, and problem difficulty: A step toward demystifying tabu search. J Artif Intell Res 24:221–261
100. Whitley D, Rana S, Dzubera J, Mathias KE (1996) Evaluating evolutionary algorithms. Artif Intell 85:245–276
101. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82
102. Woodruff DL (1998) Proposals for chunking and tabu search. Eur J Oper Res 106:585–598
103. Woodruff DL (1999) A chunking based selection strategy for integrating meta-heuristics with branch and

bound. In: Voß S, Martello S, Osman IH, Roucairol C (eds) Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer, Boston, pp 499–511

Metropolis, Nicholas Constantine

PANOS M. PARDALOS
Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 90C05, 90C25

Article Outline

Keywords
References

Keywords

Metropolis; Simulated annealing; Monte-Carlo method

Nicholas Constantine Metropolis was born in Chicago on June 11, 1915 and died on October 17, 1999 in Los Alamos. At Los Alamos, Metropolis was the main driving force behind the development of the MANIAC series of electronic computers. He was the first to code a problem for the ENIAC in 1945–1946 (together with S. Frankel), a task which consumed approximately 1,000,000 IBM punched cards.

Metropolis received his PhD in physics from the University of Chicago in 1941. He went to Los Alamos in 1943 as a member of the initial staff of fifty scientists of the Manhattan Project. He spent his entire career at Los Alamos, except for two periods (1946–1948 and 1957–1965), during which he was professor of Physics at the University of Chicago.

Metropolis is best known for the development (joint with S. Ulam and J. von Neumann) of the *Monte-Carlo method*. The Monte-Carlo method provides approximate solutions to a variety of mathematical problems by performing statistical sampling experiments on a computer. However, the real use of Monte-Carlo methods as a research tool stems from work on the atomic bomb during the second world war. This work involved a direct simulation of the probabilistic problems concerned with random neutron diffusion in fissile material. Metropolis and his collaborators, obtained Monte-Carlo estimates for the eigenvalues of Schrodinger equation.

In 1953, Metropolis co-authored the first paper on the technique that came to be known as *simulated annealing* [3,8]. Simulated annealing is a method for solving optimization problems. The name of the algorithm derives from an analogy between the simulation of the annealing of solids. Annealing refers to a process of cooling material slowly until it reaches a stable state.

Metropolis also made several early contributions to the use of computers in the exploration of nonlinear dynamics. In the Sixties and Seventies he collaborated with G.-C. Rota and others on significance arithmetic. Another contribution of Metropolis to numerical analysis is an early paper on the use of *Chebyshev's iterative method* for solving large scale linear systems [1].

References

1. Blair A, Metropolis N, von Neumann J, Taub AH, Tsingou M (1959) A study of a numerical solution to a two-dimensional hydrodynamical problem. Math Tables and Other Aids to Computation 13(67):145–184
2. Harlow F, Metropolis N (1983) Computing and computers: Weapons simulation leads to the computer era. Los Alamos Sci 7:132–141
3. Kirkpatrick S, Gelatt CD, Vecchi MP Jr (1983) Optimization by simulated annealing. Science 220(4598):671–680
4. Metropolis N (1987) The beginning of the Monte Carlo method. Los Alamos Sci 15:125–130
5. Metropolis N (1992) The age of computing: A personal memoir. Daedalus 121(1):87–103
6. Metropolis N, Howlett J, Rota G-C (eds) (1980) A history of computing in the twentieth century. Acad. Press, New York
7. Metropolis N, Nelson EC (oct. 1982) Early computing at Los Alamos. Ann Hist Comput 4(4):348–357
8. Metropolis N, Rosenbluth A, Teller A, Teller E (1953) Equation of state calculation by fast computing machines. J Chem Phys 21:1087–1092

Minimax: Directional Differentiability

VLADIMIR F. DEMYANOV
St. Petersburg State University, St. Petersburg, Russia

MSC2000: 90C30, 65K05

Article Outline

Keywords
A max-function
A Maximum Function
with Dependent Constraints
A Maxmin Function
Higher-Order Directional Derivatives
Hypodifferentiability of a Max Function
See also
References

Keywords

Minimax problem; Max-function; Maxmin function; Directional derivative; Higher-order derivatives; Hypodifferentiability; Support function

Minimax is a principle of optimal choice (of some parameters or functions). If applied, this principle requires to find extremal values of some max-type function. Since the operation of taking the pointwise maximum (of a finite or infinite number of functions) generates, in general, a nonsmooth function, it is important to study properties of such a function. Fortunately enough, though a max-function is not differentiable, in many cases it is still directionally differentiable. The directional differentiability provides a tool for formulating necessary (and sometimes sufficient) conditions for a minimum or maximum and for constructing numerical algorithms.

Recall that a function $f : \mathbf{R}^n \to \mathbf{R}$ is called *Hadamard directionally differentiable* (H.d.d.) at a point $x \in \mathbf{R}^n$ if for any $g \in \mathbf{R}^n$ there exists the finite limit

$$f'_{\mathrm{H}}(x, g) = \lim_{[\alpha, g'] \to [+0, g]} \frac{f(x + \alpha g') - f(x)}{\alpha}.$$

A function $f: \mathbf{R}^n \to \mathbf{R}$ is called *Dini directionally differentiable* (D.d.d.) at a point $x \in \mathbf{R}^n$ if for any $g \in \mathbf{R}^n$ there exists the finite limit

$$f'_{\mathrm{D}}(x, g) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha g) - f(x)}{\alpha}.$$

If f is H.d.d., then it is D.d.d. as well and ${f_{\mathrm{H}}}'(x, g) = {f_D}'(x, g)$.

Let $\Omega \subset \mathbf{R}^n$ be a convex compact set, $x \in \Omega$. The cone

$$N_x(\Omega) = \{v \in \mathbb{R}^n : \ (v, x) = \rho_\Omega(x)\}$$

is called *normal* to Ω at x. Here

$$\rho_\Omega(x) = \max_{y \in \Omega} (v, y)$$

is the *support function* of Ω at x.

A max-function

Let

$$f(x) = \max_{y \in G} \varphi(x, y), \tag{1}$$

where φ: $S \times G \to \mathbf{R}$ is continuous jointly in x, y on $S \times G$ and continuously differentiable in x there, $S \subset \mathbf{R}^n$ is an open set, G is a compact set of some space. Under the conditions stated, the function f is continuous on S.

Proposition 1 *The function f is H.d.d. at any point $x \in S$ and*

$$f'_H(x, g) = \max_{y \in R(x)} (\varphi'_x(x, y), g) = \max_{v \in \partial f(x)} (v, g), \tag{2}$$

where

$$R(x) = \{y \in G : \ f(x) = \varphi(x, y)\},$$

${\varphi_x}'(x, g)$ is the gradient of φ with respect to x for a fixed y, (a, b) is the scalar product of vectors a and b,

$$\partial f(x) = \operatorname{co} \left\{\varphi'_x(x, g) : \ y \in R(x)\right\} \subset \mathbb{R}^n.$$

The set $\partial f(x)$ is called the *subdifferential* of f at x. It is convex and compact. The mapping ∂f is, in general, discontinuous.

Remark 2 It turns out that a convex function can also be represented in the form (1) with φ being *affine* in x. For this special (convex) case the set $\partial f(x)$ is

$$\partial f(x) = \{v \in \mathbb{R}^n : \ f(z) - f(x) \geq (v, z - x), \ \forall z \in S\}.$$

The discovery of the directional differentiability of max-functions ([1,2,6]) and convex functions [10] was a breakthrough and led to the development of *minimax theory* and convex analysis ([4,9,10]).

A Maximum Function with Dependent Constraints

Let $x \subset \mathbf{R}^n$, $Y \subset \mathbf{R}^m$ be open sets and let

$$f(x) = \max_{y \in a(x)} \varphi(x, g), \tag{3}$$

where $a(x)$ is a multivalued mapping with compact images, φ: $X \times Y \to \mathbf{R}$ is Hadamard differentiable as a function of two variables, i. e. there exists the limit

$$\varphi'_{\mathrm{H}}([x, y], [g, v]) = \lim_{[\alpha, g', v'] \to [+0, g, v]} \frac{1}{\alpha} \cdot \left[\varphi(x + \alpha g', y + \alpha v') - \varphi(x, y)\right].$$

Then φ is continuous and $\varphi_{\mathrm{H}}{}'$ is continuous as a function of direction $[g, v]$.

The function f is called a *maximum function with dependent constraints*. Such functions are of great importance and have widely been studied (see [3,5,7,8]). To illustrate the results let us formulate one of them [5, Thm. I, 6.3].

Proposition 3 *Let a mapping a be closed and bounded, its images be convex and compact, the support function $a(x, l) = \max_{v \in a(x)} (v, l)$ be uniformly differentiable with respect to parameter l. Let, further, $x \in X$ and a function φ be concave in some convex neighborhood of the set $\{[x, y]: y \in R(x)\}$ (where $R(x) = \{y \in a(x): \varphi(x, y) = f(x)\}$). Then f (see (3)) is H.d.d. and*

$$f'(x, g) = \sup_{y \in R(x)} \min_{[l_1, l_2] \in V(x, y)} [(l_1, g) + a'(x, l_2; g)], \tag{4}$$

where

$$V(x, y) = \left\{l = [l_1, l_2] \in \overline{\partial}\varphi(x, y): \ l_2 \in N_{x,y}\right\},$$

$\overline{\partial}\varphi(x, y)$ is the superdifferential of φ at the point $[x, y]$, and $N_{x,y}$ is the cone normal to $a(x)$ at y.

Recall that if a function F: $\mathbf{R}^s \to \mathbf{R}$ is concave, $Z \subset \mathbf{R}^s$ is open, $z \in Z$, then the set

$$\overline{\partial}F(z) = \left\{v \in \mathbb{R}^s: \ \begin{array}{c} F(z') - F(z) \le (v, z' - z), \\ \forall z' \in Z \end{array}\right\}$$

is called the *superdifferential* of F at $z \in Z$. It is convex and compact.

A Maxmin Function

Let $\varphi(x, y, z)$: $S \times G_1 \times G_2 \to \mathbf{R}$ be continuous jointly in all variables, $S \subset \mathbf{R}^n$ be an open set, $G_1 \subset \mathbf{R}^m$, $G_2 \subset \mathbf{R}^p$ be compact. Put

$$f(x) = \max_{y \in G_1} \min_{z \in G_2} \varphi(x, y, z). \tag{5}$$

The function f is continuous on S.

Let

$$\Phi(x, y) = \min_{z \in G_2} \varphi(x, y, z),$$
$$R(x) = \{y \in G_1: \ \Phi(x, y) = f(x)\},$$
$$Q(x, y) = \{z \in G_2: \ \varphi(x, y, z) = \Phi(x, y)\}.$$

Fix $x \in S$, let $D_\varepsilon(\varepsilon > 0)$ be an ε-neighborhood of the set $\{x\} \times R(x) \times \cup_{y \in R(x)} Q(x, y)$. Assume that the derivatives

$$\frac{\partial\varphi}{\partial x}, \ \frac{\partial\varphi}{\partial y}, \ \frac{\partial^2\varphi}{\partial x^2}, \ \frac{\partial^2\varphi}{\partial x \partial y}, \ \frac{\partial^2\varphi}{\partial y^2}$$

exist and are continuous jointly in all variables on $D_\varepsilon(x)$ and that

$$\left(\frac{\partial^2\varphi(\overline{x}, y, z)}{\partial y^2} v, v\right) \le 0, \quad \forall [\overline{x}, y, z] \in D_\varepsilon(x), \quad v \in \mathbb{R}^m.$$

Assume also that G_1 is convex. Let $y \in G_1$. Put

$$\gamma(y) = \left\{v = \lambda(y' - y): \ \lambda > 0, \ y' \in G_1\right\},$$
$$\Gamma(y) = \mathrm{cl}\,\gamma(y).$$

Proposition 4 [3, Thm. 5.2] *Under the above assumptions the function f (see (5)) is Hadamard directionally differentiable and*

$$f'_H(x, g) = \sup_{y \in R(x)} \sup_{v \in \Gamma(y)} \min_{z \in Q(x, y)} \left[\left(\frac{\partial\varphi(x, y, z)}{\partial y}, v\right) + \left(\frac{\partial\varphi(x, y, z)}{\partial x}, g\right)\right].$$

Remark 5 More sophisticated results on the directional differentiability of max- and maxmin functions can be found, e. g., in [8].

Higher-Order Directional Derivatives

The results above are related to the first order directional derivatives. Using these derivatives, it is possible

to construct the following first order expansion:

$$f(x+\alpha g) = f(x) + \alpha f'(x,g) + o_{x,g}(\alpha), \tag{6}$$

where f' is either ${f_H}'$ or ${f_D}'$.

In some cases it is possible to get 'higher-order' expansions.

Let

$$f(x) = \max_{i\in I} f_i(x), \tag{7}$$

where $I = 1:N$, $x = (x_1, \ldots, x_n) \in \mathbf{R}^n$, the ${f_i}'$s are continuous and continuously differentiable up the lth order on an open set $S \subset \mathbf{R}^r$. Fix $x \in S$. Then for sufficiently small $\alpha > 0$

$$f_i(x+\alpha g) = f_i(x) + \sum_{k=1}^{l} \frac{\alpha^k}{k!} f_i^{(k)}(x,g) + o_i(g,\alpha^l), \tag{8}$$

where

$$f_i^{(k)}(x,g) = \sum_{j_1,\ldots,j_k=1}^{n} \frac{\partial^k f_i(x)}{\partial x_{j_1}\cdots\partial x_{j_k}} g_{j_1},\ldots,g_{j_k},$$

$$k \in 1,\ldots,l, \quad \frac{o_i(g,\alpha^l)}{\alpha^l} \underset{\alpha\downarrow 0}{\to} 0 \tag{9}$$

uniformly with respect to g, $\|g\| = 1$.

Let us use the following notation

$$f_i^0(x,g) = f_i(x),$$
$$\forall i \in I, \quad R_0(x,g) = I,$$
$$R_k(x,g) = \{i \in R_{k-1}(x,g) : f_i^{(k-1)}(x,g) = \max_{j\in R_{k-1}(x,g)} f_j^{(k-1)}(x,g)\},$$
$$k \in 1,\ldots,l.$$

Clearly

$$R_0(x,g) \supset R_1(x,g) \supset R_2(x,g) \supset \cdots.$$

Note that $R_0(x,g)$ does not depend on x and g, and $R_1(x,g)$ does not depend on g.

Proposition 6 [3, Thm. 9.1] *The following expansion holds:*

$$f(x+\alpha g) = f(x) + \sum_{k=1}^{l} \frac{\alpha^k}{k!} f^{(k)}(x,g) + o(g,\alpha^l), \quad \forall g \in \mathbb{R}^n, \tag{10}$$

where

$$f^{(k)}(x,g) = \max_{i\in R_k(x,g)} f_i^{(k)}(x,g), \quad \frac{o(g,\alpha^l)}{\alpha^l} \underset{\alpha\downarrow 0}{\to} 0 \tag{11}$$

uniformly with respect to g, $\|g\| = 1$.

The value $\partial^k f(x)/\partial g^k = f^{(k)}(x,g)$ is called the kth *derivative* of f at x in a direction g.

Remark 7 The mapping $R_1(x,g)$ is not continuous in x, while the mappings $R_k(x,g)$ $(k \geq 2)$ are not continuous in x as well as in g. Therefore the functions $f^{(k)}(x,g)$ in (11) are not continuous in x and (if $k \geq 2$) in g and, as a result, expansion (6) is also not 'stable' in x.

To overcome this difficulty we shall employ another tool.

Hypodifferentiability of a Max Function

Let us again consider the case where f is defined by (7). It follows from (8) that, for $\Delta = (\Delta_1, \ldots, \Delta_n) \in \mathbf{R}^n$,

$$f(x+\Delta) = \max_{i\in I}\left[f_i(x) + \sum_{k=1}^{l} \frac{1}{k!} f_i^{(k)}(x,\Delta)\right] + o(\|\Delta\|^k). \tag{12}$$

Let us use the notation (see (9))

$$f_i^{(k)}(x,\Delta) = A_{ik}\Delta^k.$$

The function $f_i^{(k)}(x,\Delta)$ is a *kth order form of coordinates* $\Delta_1,\ldots,\Delta_n$; A_{ik} being the set of coefficients of this form. Then (12) can be rewritten as

$$f(x+\Delta) = \max_{i\in I}\left[f_i(x) + \sum_{k=1}^{l} \frac{1}{k!} A_{ik}\Delta^k\right] + o(\|\Delta\|^k)$$
$$= f(x) + \max_{A\in d^l f(x)}\left[\sum_{k=1}^{l} \frac{1}{k!} A_k \Delta^k\right] + o(\|\Delta\|^k), \tag{13}$$

where

$$d^l f(x) = \operatorname{co}\left\{A^{(i)} = (A_{i0}, \ldots, A_{il}):\ i \in I\right\},$$
$$A_{i0} = f_i(x) - f(x), \quad A = (A_0, \ldots, A_l),$$
$$A_0 \in \mathbb{R},\ A_1 \in \mathbb{R}^n,$$
$$A_2 \in \mathbb{R}^{n\times n}, \ldots, A_k \in \overbrace{\mathbb{R}^{n\times\cdots\times n}}^{k \text{ times}}.$$

Here, $\overbrace{\mathbb{R}^{n\times\cdots\times n}}^{k \text{ times}}$ is the space of kth order real forms, e. g. $\mathbf{R}^{n\times n}$ is the space of real $(n \times n)$-matrices.

The set $d^l f(x)$ is called the *kth order hypodifferential* of f at x. It is an element of the space $\mathbb{R} \times \mathbb{R}^n \times \cdots \times \overbrace{\mathbb{R}^{n\times\cdots\times n}}^{l}$. The mapping $d^l f$ is continuous in x.

Remark 8 Expansion (13) can be extended to the case where f is given by (1) and φ is l times continuously differentiable in x.

Max functions represent a special case of the class of *quasidifferentiable functions* (see [5]).

See also

- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Minimax Theorems
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Quasigradient Methods in Minimax Problems

References

1. Danskin JM (1967) The theory of max-min and its application to weapons allocation problems. Springer, Berlin
2. Demyanov VF (1966) On minimizing the maximal deviation. Vestn Leningrad Univ 7:21–28
3. Demyanov VF (1974) Minimax: Directional differentiability. Leningrad Univ. Press, Leningrad
4. Demyanov VF, Malozemov VN (1974) Introduction to minimax. Wiley, New York. Second edition: Dover, New York (1990)
5. Demyanov VF, Rubinov AM (1995) Constructive nonsmooth analysis. P. Lang, Frankfurt am Main
6. Girsanov IV (1965) Differentiability of solutions of the mathematical programming problems. Abstracts Conf. Applications of Functional Analysis Methods to Solving Nonlinear Problems, pp 43–45
7. Levitin ES (1994) Perturbation theory in mathematical programming and its applications. Wiley, New York
8. Minchenko LI, Borisenko OF (1992) Differential properties of marginal functions and their applications to optimization problems. Nauka i Techn., Minsk
9. Pschenichny BN (1980) Convex analysis and extremal problems. Nauka, Moscow
10. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton

Minimax Game Tree Searching

CLAUDE G. DIDERICH[1], MARC GENGLER[2]

[1] Computer Sci. Department, Swiss Federal Institute Technology-Lausanne, Lausanne, Switzerland

[2] Ecole Sup. d'Ingénieurs de Luminy, Université Méditerrannée, Marseille, France

MSC2000: 49J35, 49K35, 62C20, 91A05, 91A40

Article Outline

Keywords
Minimax Trees
Sequential Minimax Game Tree Algorithms
  Minimax Algorithm
  Alpha-Beta Algorithm
  Optimal State Space Search Algorithm SSS*
  SCOUT: Minimax Algorithm of Theoretical Interest
  GSEARCH: Generalized Game Tree Search Algorithm
  SSS-2: Recursive State Space Search Algorithm
  Some Variations On The Subject
Parallel Minimax Tree Algorithms
  A Simple Way to Parallelize the Exploration of Minimax Trees
  A Mandatory Work First Algorithm
  Aspiration Search
  Tree-Splitting Algorithm
  PVSPLIT: Principal Variation Splitting Algorithm
  Synchronized Distributed State Space Search
  Distributed Game Tree Search Algorithm
  Parallel Minimax Algorithm with Linear Speedup
See also
References

Keywords

Algorithms; Games; Minimax; Searching

With the introduction of computers, also started the interest in having machines play games. Programming a computer such that it could play, for example chess, was seen as giving it some kind of intelligence. Starting in the mid fifties, a theory on how to play two player zero sum perfect information games, like chess or go, was developed. This theory is essentially based on traversing a tree called minimax or game tree. An edge in the tree represents a move by either of the players and a node a configuration of the game.

Two major algorithms have emerged to compute the best sequence of moves in such a minimax tree. On one hand, there is the alpha-beta algorithm suggested around 1956 by I. McCarthy and first published in [27]. On the other hand, G.C. Stockman [29] introduced the SSS∗ algorithm. Both methods try to minimize the number of nodes explored in the game tree using special traversal strategies and cut conditions.

Minimax Trees

A *two-player zero-sum perfect-information game*, also called *minimax game*, is a game which involves exactly two players who alternatively make moves. No information is hidden from the adversary. No coins are tossed, that is, the game is completely deterministic, and there is perfect symmetry in the quality of the moves allowed. Go, checker and chess are such minimax games whereas backgammon (the outcome of a die determines the moves available) or card games (cards are hidden from the adversary) are not.

A *minimax tree* or *game tree* is a tree where each node represents a state of the game and each edge a possible move. Nodes are alternatively labeled 'max' and 'min' representing either player's turn. A node having no descendants represents a final outcome of the game. The goal of a game is to find a winning sequence of moves, given that the opponent always plays his best move.

The quality of a node t in the minimax game tree, representing a configuration, is given by its value $e(t)$. The value $e(t)$, also called *minimax value*, is defined recursively as

$$e(t) = \begin{cases} f(t) & \text{if } t \text{ is a leave node,} \\ \max\limits_{s \in \text{sons}(t)} e(s) & \text{if } t \text{ is labeled 'max',} \\ \min\limits_{s \in \text{sons}(t)} e(s) & \text{if } t \text{ is labeled 'min'.} \end{cases}$$

If the considered minimax tree represents a *complete game*, that is, all possible board configurations, the function f may be defined as follows:

$$f(t) = \begin{cases} +1 & \text{if } t \text{ leads to a winning position,} \\ 0 & \text{if } t \text{ leads to a tie position,} \\ -1 & \text{if } t \text{ leads to a losing position;} \end{cases}$$

otherwise $f(t)$ represents an evaluation of the quality of a board position.

The relation between minimax trees and games is detailed in the following table.

Minimax tree notion	Minimax game notion
Minimax tree	All board configurations
Node in the tree	Board configuration
Edge from "max" to "min" node	Move by player "max"
Edge from "min" to "max" node	Move by player "min"
Node value	Quality of board position
Leave node	Outcome of a game
Solution path	Sequence of moves leading the best outcome

Sequential Minimax Game Tree Algorithms

Let t be a node of a minimax tree. Then the function first_son(t) returns the first son node s_1 of t and next_son(s_i, t) returns the i + 1th son of node t. The function no_more_sons(s, t) returns true of s is the last son of t. Otherwise it returns false. The ordering of the sons introduced by these functions is arbitrary. In practice it is given by some heuristic function. The function father(t) returns the father node of t, is_leave(t) whether or not t is a leave node and node_type(t) the type of node t.

Minimax Algorithm

The most basic minimax algorithm is called the *minimax algorithm*. It systematically traverses, in a depth first, left to right fashion, the complete minimax tree. All nodes are visited exactly once.

Alpha-Beta Algorithm

The first nontrivial algorithm introduced to compute the minimax value of a game tree was the

alpha-beta algorithm. According to D. Knuth and R. Moore, McCarthy's comments at the Dartmouth summer research conference on artificial intelligence led to the use of alpha-beta pruning in game playing programs since the late 1950s. The first published discussion of an algorithm for minimax tree pruning appeared in 1958 (see [11, p. 56]). Two early extensive studies of the algorithm may be found in [18] and [27].

The idea behind the alpha-beta algorithm is to traverse the minimax tree in a depth first, left to right fashion. It tries to prune sub-trees that can not influence the minimax value of the tree. The conditions used to prune sub-trees are called *cut conditions*. The idea behind the suggested cut conditions is to associate to each node a lower and an upper bound, called α and β bounds. The bounds of a node are passed to its sons and tightened during the execution of the algorithm. It is easy to see that if the lower bound of a node t of type 'max' is larger than its upper bound then all not visited sons of node t can be pruned, and similar for nodes of type 'min'.

```
FUNCTION AlphaBeta(n, α, β) IS
BEGIN
   IF is_leave(n) THEN RETURN f(n)
   s ← first_son(n)
   IF node_type(n)=max THEN
      LOOP
         α ← max{α, AlphaBeta(s, α, β)}
         IF α ≥ β THEN RETURN β
         EXIT LOOP WHEN no_more_sons(s, n)
         s ← next_son(s, n)
      END LOOP
      RETURN α
   ELSE
      LOOP
         β ← max{α, AlphaBeta(s, α, β)}
         IF β ≤ α THEN RETURN α
         EXIT LOOP WHEN no_more_sons(s, n)
         s ←next_sons(s, n)
      END LOOP
      RETURN β
   END IF
END AlphaBeta
```

Pseudocode for the alpha-beta algorithm

It has been proved in [18] that the alpha-beta algorithm correctly calculates the minimax value of a tree. The above pseudocode describes the alpha-beta algorithm.

The minimax value of a tree T is computed as follows.

$$e\,(\mathrm{root}(T)) \leftarrow \mathrm{AlphaBeta}\,(\mathrm{root}(T), -\infty, +\infty)\,.$$

Optimal State Space Search Algorithm SSS∗

It has been introduced by Stockman in 1979, [29]. It originates not in game playing but in systematic pattern recognition. The algorithm was first analyzed and criticized in [26].

The idea behind the SSS∗ algorithm is to use a tree traversal strategy that is, better than the depth first and left to right strategy found in the alpha-beta algorithm. The criteria used to order the nodes yet to visit is an upper bound of their value. Nodes are stored in non increasing order of their upper bound in a list called 'open'.

The SSS∗ algorithm first traverses the minimax tree from top to bottom. Nodes whose sons have not yet been visited and which cannot yet be pruned are marked 'live'. Nodes marked 'solved' have already been visited once and have therefore their best upper bound associated.

The operation purge(t, open) removes all nodes from the open list for which the node t is an ancestor. Due to the fact that the nodes in the open list are sorted in nonincreasing order of their associated upper bound, the pruning operation only eliminates nodes that need no further consideration.

The SSS∗ algorithm is described by the following pseudocode.

```
FUNCTION SSS∗ IS
BEGIN
   open ← ∅
   insert (root, live, +∞, open)
   LOOP
      (s, t, m) ← remove (open)
      IF s = root AND t = solved THEN RETURN m
      ⟨ Apply the Γ operator to node s ⟩
   END LOOP
END SSS∗
```

Pseudocode for the SSS∗ algorithm

The operator $\Gamma(s)$ is applied to each node s extracted from the 'open' list.

It is possible to define a dual version of the SSS∗, which may be called *SSS∗ -dual*, in which the computation of upper bounds is replaced by the computation of lower bounds. The SSS∗ -dual algorithm has been suggested in [21].

Stockman has shown that if the SSS∗ algorithm explores a node, then this node is also explored by the alpha-beta algorithm. In fact, the alpha-beta algorithm loses efficiency (in the number of nodes visited) against the SSS∗ algorithm when the value of the minimax tree is found towards the right of the tree. If the SSS∗ algorithm is applied to win-lose trees then it visits exactly the same nodes in the same order as would the alpha-beta algorithm.

```
⟨Apply the Γ operator to node s⟩ ≡
  IF t = live AND node_type = max
    AND NOT is_leave(t) THEN
    s ← first_son(t)
    LOOP
      insert (s, live, m, open)
      EXIT LOOP WHEN no_more_sons(s, t)
      s ← next_son(s, t)
    END LOOP
  END IF
  IF t = live AND node_type = min
    AND NOT is_leave(t) THEN
    insert(first_son(t), live, m, open)
  END IF
  IF t = live AND is_leave(t) THEN
    insert(t, solved, min{f(t), m}, open)
  END IF
  IF t = solved AND node_type = max
    AND NOT no_more_sons(t, father(t)) THEN
    insert(next_son(t, father(t)), live, m, open)
  END IF
  IF t = solved AND node_type = max
    AND no_more_sons(t, father(t)) THEN
    insert(father(t), solved, m, open)
  END IF
  IF t = solved AND node_type = min THEN
    insert(father(t), solved, m, open)
    purge(father(t), open)
  END IF
```

SCOUT: Minimax Algorithm of Theoretical Interest

In the previous sections, we have described the most common minimax algorithms. While trying to show the optimality of the alpha-beta algorithm, J. Pearl [23] introduced the SCOUT algorithm. His idea was to show that the SCOUT algorithm is dominated by the alpha-beta algorithm and to prove that SCOUT achieves an optimal performance. But counterexamples showed that the alpha-beta algorithm does not dominate the SCOUT algorithm because the conservative testing approach of the SCOUT algorithm may sometimes cut off nodes that would have been explored by the alpha-beta algorithm.

The SCOUT algorithm itself recursively computes the value of the first of its sons. Then it tests to see if the value of the first son is better that the value of the other sons. In case of a negative result, the son that failed the test is completely evaluated by recursively calling SCOUT.

Although the SCOUT algorithm is more of theoretical interest, there are some problem instances where it outperforms all other minimax algorithms. A last advantage of the SCOUT algorithm versus one of its major competitors, the SSS∗ algorithm, is that its storage requirements are similar to those of the alpha-beta algorithm.

GSEARCH: Generalized Game Tree Search Algorithm

In 1986, T. Ibaraki [16] proposed a generalization of the previously known algorithms to compute the minimax value of a game tree. His idea was to use a branch and bound like approach. Nodes of the considered tree which have not yet been evaluated are stored in a list which is ordered according to a given criteria. Different orderings give different traversal strategies. A lower and upper bound is associated to each node. These bounds generalize the α and β values found in the alpha-beta algorithm.

Finally Ibaraki showed how the algorithm GS}CH is related to other minimax algorithms like alpha-beta or SSS∗, and proved that his algorithm always surpasses the alpha-beta algorithm.

SSS-2: Recursive State Space Search Algorithm

The SSS-2 algorithm has been proposed by W. Pijls and A. de Bruin [24]. It is based on the idea of computing an upper bound for the root node and then repeatedly transforming this upper bound into a tighter one. They have shown that the SSS-2 algorithm exactly expands the same nodes as those to which the SSS∗ algorithm applies the Γ operator.

Some Variations On The Subject

Computing the minimax value of a game tree may be seen as aspiring the solution value from a leave node through the whole tree up to the root node. While moving closer to the root node, more and more useless subtrees will be eliminated, as we have already stated for the alpha-beta algorithm. The better the α and β bounds, the more subtrees may be pruned. If, for instance, one knows that the minimax value will, with high probability, be found in the subset $]a, b[$, then it may be worth calling the alpha-beta algorithm as

$$e \leftarrow \text{AlphaBeta}\,(\text{root}(T), a, b)$$

If, indeed, the minimax value $e(\text{root}(T))$ belongs to the set $]a, b[$, then the algorithm will correctly return that value. If the minimax value does not belong to the set $]a, b[$, then the value returned will be either a or b, depending on whether the minimax value belongs to $]-\infty, a]$ or $[b, +\infty[$. We then say that the alpha-beta algorithm *failed low*, respectively *high*. In the case where the algorithm failed low, the call

$$e \leftarrow \text{AlphaBeta}\,(\text{root}(T), -\infty, a + 1)$$

will return the correct value. But it would also be possible to reiterate this procedure on a subset $]a_1, a + 1[$.

The technique of limiting the interval in which the solution may be found is called *aspiration search*. If the minimax value belongs to the specified interval, then a much larger number of cut conditions are verified and the tree actually traversed is much smaller than the one traversed by the alpha-beta algorithm without initial alpha and beta bounds.

Furthermore it is interesting to note that aspiration search is at the bases of a technique called *iterative deepening* which is used in many game playing programs.

I. Althöfer [5] suggested an *incremental negamax algorithm* which uses estimates of all nodes in the minimax tree, rather than only those of the leave nodes, to determine the value of the root node. This algorithm is useful when dealing with erroneous leave evaluation functions. Under the assumption of independently occurring and sufficiently small errors, the proposed algorithm is shown to have exponentially reduced error probabilities with respect to the depth of the tree.

R.L. Rivest [25] proposed an algorithm for searching minimax trees based on the idea of approximating the min and the max operators by generalized mean value operators. The approximation is used to guide the selection of the next leave node to expand, since the approximation allows to select efficiently that leave node upon whose value the minimax value most highly depends. B.W. Ballard [6] proposed a similar algorithm where the value of some nodes (the chance node as he calls them) is a, possibly weighted, average of the values of its sons. In fact he considers one additional type of nodes called chance nodes.

Conspiracy numbers have been introduced by D.A. McAllester in [22] as a measurement of the accuracy of the minimax value of an incomplete tree. They measure the number of leave nodes whose value must change in order to change the minimax value of the root node by a given amount.

Parallel Minimax Tree Algorithms

Parallelizing the minimax algorithm is trivial over uniform trees. Even on irregular trees, the parallelization remains easy. The only additional problem arises from the fact that the size of the subtrees to explore may now vary. Different processors will be attributed problems of varying computational volume. All what is needed then to achieve excellent speedups, is a load-balancing scheme, that is, a mechanism by means of which processors may, during run-time, exchange problems so as to keep all processors busy all the time.

The parallelization of the alpha-beta and the SSS∗ algorithms are much more interesting than the more theoretical minimax algorithm. There exist basically two approaches or techniques to parallelize the alpha-beta algorithm. In the first approach, which has been one of the first techniques used, all processors explore the entire tree but using different search-intervals. This approach is at the basic of the algorithm called *parallel aspiration search* by G. Baudet [7]. The second one

consists in exploring simultaneously different parts of the minimax tree.

A Simple Way to Parallelize the Exploration of Minimax Trees

Exploring a minimax tree in parallel can very simply be obtained by generating the sons of the root node, and their sons and so on up to the point where one has as many son nodes waiting to be explored as there are processors. At this point, each processor explores the subtree rooted at one of these nodes, using any given sequential minimax algorithm. When all processors have completed their exploration, the solution for the entire tree is computed by using the partial results obtained from each of the processors.

In practice the sons of a node may be ordered in such a way that any son has a probability of yielding the locally optimal path that is no smaller than the corresponding probabilities for its right neighbors. The probability to find the optimum in the subtree rooted at a given son then always decreases when traversing the sons in a left to right order. Such ordering information is generally available in game-playing programs, the ordering function being a heuristic function based on the knowledge of the game to be played.

A Mandatory Work First Algorithm

R. Hewett and G. Krishnamurthy [15] proposed an algorithm that achieves an efficiency of roughly 50% for an number of processors in the range of 2 to 25. All the nodes that still need to be explored are maintained in a list called 'open' list. This list is ordered with respect to how the nodes have been reached. More precisely, the algorithm maintains two lists called 'open' and 'closed', and a tree called 'cut'. The 'open' list contains all the nodes yet to be explored, the 'closed' list contains the expanded nodes not yet pruned and the 'cut' tree contains the pruned nodes. The 'open' list initially contains only the root node. All processors fetch nodes from the 'open' list and process them if they cannot be discarded, that is, they do not have any of their ancestors in the 'cut' tree. Leave nodes are evaluated and their result is returned to the parent which may update its value and check for possible pruning by traversing the 'cut' tree up to the root node applying the usual alpha and beta cutoffs. If the node selected is not a leave node, it is expanded and its sons are inserted into the 'open' list and itself into the 'closed' list.

S.G. Akl et al. [1,2] proposed an algorithm that uses the same approach for exploring the minimax tree. Their priority function is computed as

$$p(n_i) = p(\text{father}(n_i)) - (b_{n_i} + 1 - i) \cdot 10^{(h-f-1)},$$

where n_i is the ith son of node father(n_i), b_{n_i} the branching of node father(n_i), h the search depth (the maximal depth of the minimax tree) and f the depth of node father(n_i) in the minimax tree.

K. Almquist et al. [3] also developed an algorithm based on the idea of having two categories of unexplored nodes which are ordered according to a given priority function. Furthermore they add to this concept parallel aspiration search as well as a novel scheduling algorithm.

In the same direction, V.-D. Cung and C. Roucairol [9] have proposed a shared memory parallel minimax algorithm which distinguishes between critical and non critical nodes. In their algorithm one processor is assigned to each node.

In the algorithm by I.R. Steinberg and M. Solomon [28], which is also a mandatory work first type algorithm, the list containing the speculative work or non critical nodes is dynamically ordered.

Aspiration Search

The parallel algorithm called *aspiration search* has been introduced by Baudet in 1978 [7]. In this algorithm the search interval $]-\infty, +\infty[$ used by the sequential alpha-beta algorithm is divided into a certain number of subintervals that cover the entire range $]-\infty, +\infty[$. Now, every processor explores the entire minimax tree using one subinterval, different processors being assigned different intervals. Any processor searching an interval $]a_i, a_{i+1}]$ may either fail low or high. The principle is the same as in the sequential version of the algorithm. Exactly one processor will neither fail low, nor fail high. The value computed by this processor is the value of the minimax tree to explore.

The implementation of the aspiration search algorithm is really simple. Furthermore, there is no information exchange needed between processors. If the nodes in the to explore minimax tree are ordered in such a way that the alpha-beta algorithm has to explore

the whole tree, then the speedup obtained by using the aspiration search algorithm is maximal. But, when the aspiration search algorithm is applied to randomly generated trees then Baudet has shown that the speedup is limited to about six and is independent of the number of processors used.

Tree-Splitting Algorithm

Among the early parallel minimax algorithms is the *tree-splitting algorithm* by R.A. Finkel and J.P. Fishburn [14]. This algorithm is based on the idea to look at the available processors as a tree of processors. Each processor, except for the ones representing leaves in the processor tree, have a fixed number p_b of son or slave processors. During the execution of the algorithm a non leave processor associated with a node n in the minimax tree spawns the exploration of the sons s_i of n to its p_b slaves. As soon as one slave returns the next unexplored son s_j is spawned to that slave or the current value is returned to the father processor if the cut condition is satisfied. If all the sons of a node have been spawned to its slaves, the father processor waits for the results of all its slaves. Leave processors simply compute the value of their associated node using the sequential alpha-beta algorithm.

An important advantage of the tree-splitting algorithm over other more elaborated algorithms is that it may be simply implemented as well on a shared memory parallel machine as on a distributed memories parallel machine.

The tree-splitting algorithm has been implemented and its execution has been simulated. On a 27 processor simulated machine, in which each processor has tree slave sons associated, the average speedup was 5.31 for trees of depth eight and a branching of three.

PVSPLIT: Principal Variation Splitting Algorithm

It has been proposed by T.A. Marsland and M.S. Campbell [19] and is by far the most often implemented algorithm, especially in chess playing programs. The algorithm is based on the structure of the sequential alpha-beta algorithm. The idea is to first explore in a sequential fashion a path from the root node to its leftmost leave. This path is called the *principal variation path*. The traversal is done to obtain alpha and beta bounds. If the minimax tree to explore is of type best first, then the explored principal variation path represents the solution path. In a second phase, for each level of the minimax tree all the yet to be visited sons are explored in parallel by using the bounds computed during the principal variation path computation and the traversal of the lower levels of the minimax tree.

The PVSPLIT algorithm is completely described by the following pseudocode using the negamax notation.

The PVSPLIT algorithm has been implemented in [20] on a network of Sun workstations. An acceleration of 3.06 has been measured on 4 processors when traversing minimax trees representing real chess games. The main problem of the PVSPLIT algorithm is that, during the second phase, the subtrees explored in parallel are not necessarily of the same size.

The PVSPLIT algorithm is most efficient when the iterative deepening technique is used, because with each iteration is is increasingly likely that the first move tried, that is, the one on the principal variation path, is the best one.

```
FUNCTION PVSplit(b, α, β) IS
BEGIN
  IF is_leave(n) THEN RETURN f(n)
  s ← first_son(n)
  α ← −PVSplit(s, −β, −α)
  IF α ≥ β THEN RETURN α
  FOR s′ ∈ sons(n) − {s} LOOP IN PARALLEL
    ⟨wait until a slave node is idle⟩
    v_i ← −TreeSplit(s′, −β, −α)
    IF v_i > α THEN
      α ← v_i
      ⟨Update the bounds according to α on all
      slaves⟩
    END IF
    IF α > β THEN
      ⟨Terminate all slave processors⟩
      RETURN α
    END IF
  END LOOP
  RETURN α
END PVSplit
```

Pseudocode for the PVSPLIT algorithm

Synchronized Distributed State Space Search

A completely different approach to parallelizing the SSS∗ algorithm has been taken by C.G. Diderich and

M. Gengler [10]. The algorithm proposed is called synchronized distributed state space search (SDSSS). It is an alternation of computation and synchronization phases. The algorithm has been designed for a distributed memory multiprocessor machine. Each processor manages its own local 'open' list of unvisited nodes.

The synchronization phase may be subdivided in three major parts. First, the processors exchange information about which nodes can be removed from the local 'open' lists. This corresponds to each processor sending the nodes for which the 'purge' operation may be applied by all the other processors. Next, all the processors agree on the globally lowest upper bound m^* for which nodes exist in some of the 'open' lists. Finally all the nodes having the same upper bound m^* are evenly distributed among all the processors. This operation concludes the synchronization phase.

The computation phase of the SDSSS algorithm may be described by the following pseudocode.

```
⟨Computation phase⟩ ≡
    WHILE ⟨there exists a node in the open list
    having an upper bound of m*⟩
        LOOP
        (s, t, m*) ← remove(open)
        IF s = root AND t = solved THEN
            BROADCAST 'the solution has been
            found'
            RETURN m*
        END IF
        ⟨Apply the Γ operator to node s⟩
    END LOOP
```

Pseudocode for the computation phase of the SDSSS algorithm

Experiments executing the SDSSS algorithm on an Intel iPSC/2 parallel machine have been conducted. Speedups of up to 11.4 have been measured for 32 processors.

Distributed Game Tree Search Algorithm

R. Feldman [12] parallelized the alpha-beta algorithm for massively parallel distributed memory machines. Different subtrees are searched in parallel by different processors. The allocation of processors to trees is done by imposing certain conditions on the nodes which are be selectable. They introduce the concept of *younger brother waits*. This concept essentially says that in the case of a subtree rooted at s_1, where s_1 is the first son node of a node n, is not yet evaluated, then the other sons $s_2, \ldots, s_b$ of node n are not selectable. Younger brothers may only be considered after their elder brothers, which has as a consequence that the value of the elder brothers may be used to give a tight search window to the younger brothers.

This concept is nevertheless not sufficient to achieve the same good search window as the alpha-beta algorithm achieves. Indeed when node s_1 is computed, then the younger brothers may all be explored in parallel using the value of node s_1. Thus the node s_2 has the same search window as it would have in the sequential alpha-beta algorithm, but this is not true anymore for s_i, where $i \geq 3$. Indeed if nodes s_2 and s_3 are processed in parallel, they only know the value of node s_1, while in the sequential alpha-beta algorithm, the node s_3 would have known the value of both s_1 and s_2. This fact forces the parallel algorithm to provide an information dissemination protocol.

In case the nodes s_2 and s_3 are evaluated on processors P and P', and processor P finishes its work before P', producing a better value than node s_1 did, then processor P will inform processor P' of this value, allowing it to continue with better information on the rest of its subtree or to terminate its work if the new value allows P' to conclude that its computation becomes useless. The load distribution is realized by means of a dynamic load balancing scheme, where idle processors ask other processors for work.

Speedups as high as 100 have been obtained on a 256 processor machines. In [13], a speedup of 344 on a 1024 transputer network interconnected as a grid and a speedup of 142 on a 256 processor transputer de Bruijn interconnected network have been shown.

Parallel Minimax Algorithm with Linear Speedup

In 1988, Althöfer [4] proved that it is possible, to develop a parallel minimax algorithm which achieves linear speedup in the average case. With the assumption that all minimax trees are binary win-loss trees, he exhibited such a parallel minimax algorithm.

M. Böhm and E. Speckenmeyer [8] also suggested an algorithm which uses the same basic ideas as Althöf-

fer. Their algorithm is more general in the sense that it needs only to know the distribution of the leave values and is independent of the branching of the tree explored.

In 1989, R.M. Karp and Y. Zhang [17] proved that it is possible to obtain linear speedup on every instance of a random uniform minimax tree if the number of processors is close to the height of the tree.

See also

- ► Bottleneck Steiner Tree Problems
- ► Directed Tree Networks
- ► Shortest Path Tree Algorithms

References

1. Akl SG, Barnard DT, Doran RJ (1979) Searching game trees in parallel. In: Proc. 3rd Biennial Conf. Canad. Soc. Computation Studies of Intelligence, pp 224–231
2. Akl SG, Barnard DT, Doran RJ (1982) Design, analysis, and implementation of a parallel tree search algorithm. IEEE Trans Pattern Anal Machine Intell PAMI-4(2):192–203
3. Almquist K, McKenzie N, Sloan K (1988) An inquiry into parallel algorithms for searching game trees. Techn. Report Univ. Washington, Seattle, WA 12(3)
4. Althöfer I (1988) On the complexity of searching game trees and other recursion trees. J Algorithms 9:538–567
5. Althöfer I (1990) An incremental negamax algorithm. Artif Intell 43:57–65
6. Ballard BW (1983) The ∗ -minimax search procedure for trees containing chance nodes. Artif Intell 21:327–350
7. Baudet GM (1978) The design and analysis of algorithms for asynchronous multiprocessors. PhD Thesis Carnegie-Mellon Univ. Pittsburgh, PA, CMU-CS-78-116
8. Böhm M, Speckenmeyer E (1989) A dynamic processor tree for solving game trees in parallel. Proc. SOR'89
9. Cung V-D, Roucairol C (1991) Parallel minimax tree searching. Res Report INRIA, vol 1549
10. Diderich CG (1992) Evaluation des performances de l'algorithme SSS∗ avec phases de synchronisation sur une machine parallèle à mémoires distribuées. Techn. Report Computer Sci. Dept. Swiss Federal Inst. Techn. Lausanne, Switzerland, LiTH-99 (In French.)
11. Feigenbaum EA, Feldman J (1963) Computers and thought. McGraw-Hill, New York
12. Feldmann R, Monien B, Mysliwietz P, Vornberger O (1989) Distributed game tree search. ICCA J 12(2):65–73
13. Feldmann R, Mysliwietz P, Monien B (1994) Game tree search on a massively parallel system. In: van den Herik HJ, Herschberg IS, Uiterwijk JWHM (eds) Advances in Computer Chess, vol 7. Univ. Limburg, Maastricht, pp 203–218
14. Finkel RA, Fishburn JP (1982) Parallelism in alpha-beta search. Artif Intell 19:89–106
15. Hewett R, Krishnamurthy G (1992) Consistent linear speedup in parallel alpha-beta search. Proc. ICCI'92, Computing and Information. IEEE Computer Soc Press, New York, pp 237–240
16. Ibaraki T (1986) Generalization of alpha-beta and {SSS*} search procedures. Artif Intell 29:73–117
17. Karp RM, Zhang Y (1989) On parallel evaluation of game trees. In: ACM Annual Symp. Parallel Algorithms and Architectures (SPAA'89). ACM, New York, pp 409–420
18. Knuth DE, Moore RW (1975) An analysis of alpha-beta pruning. Artif Intell, 6(4):293–326
19. Marsland TA, Campbell MS (1982) Parallel search of strongly ordered game trees. ACM Computing Surveys 14(4):533–551
20. Marsland TA, Popowich F (1985) Parallel game-tree search. IEEE Trans Pattern Anal Machine Intell PAMI-7(4):442–452
21. Marsland TA, Reinefeld A, Schaeffer J (1987) Low overhead alternatives to SSS*. Artif Intell 31:185–199
22. McAllester DA (1988) Conspiracy numbers for min-max searching. Artif Intell 35:287–310
23. Pearl J (1980) Asymptotical properties of minimax trees and game searching procedures. Artif Intell 14(2):113–138
24. Pijls W, de Bruin A (Aug. 1990) Another view of the SSS∗ algorithm. In: Proc. Internat. Symp. (SIGAL'90)
25. Rivest RL (1987) Game tree searching by min/max approximation. Artif Intell 34(1):77–96
26. Roizen I, Pearl J (1983) A minimax algorithm better than alpha-beta? Yes and no. Artif Intell 21:199–230
27. Slagle JH, Dixon JK (Apr. 1969) Experiments with some programs that search game trees. J ACM 16(2):189–207
28. Steinberg IR, Solomon M (1990) Searching game trees in parallel. Proc. IEEE Internat. Conf. Parallel Processing, III, III–9–III–17
29. Stockman GC (1979) A minimax algorithm better than alpha-beta? Artif Intell 12(2):179–196

Minimax Theorems

STEPHEN SIMONS
Department Math., University California,
Santa Barbara, USA

MSC2000: 46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05

Article Outline

Keywords
Von Neumann's Results
Infinite-Dimensional Results for Convex Sets

Functional-Analytic Minimax Theorems
Minimax Theorems that Depend on Connectedness
Mixed Minimax Theorems
A Metaminimax Theorem
Minimax Theorems and Weak Compactness
Minimax Inequalities for Two or More Functions
Coincidence Theorems
See also
References

Keywords

Minimax theorem; Fixed point theorem; Hahn–Banach theorem; Connectedness

We suppose that X and Y are nonempty sets and $f\colon X \times Y \to \mathbf{R}$. A *minimax theorem* is a theorem that asserts that, under certain conditions,

$$\inf_Y \sup_X f = \sup_X \inf_Y f,$$

that is to say,

$$\inf_{y\in Y} \sup_{x\in X} f(x,y) = \sup_{x\in X} \inf_{y\in Y} f(x,y).$$

The purpose of this article is to give the reader the flavor of the different kind of minimax theorems, and of the techniques that have been used to prove them. This is a very large area, and it would be impossible to touch on all the work that has been done in it in the space that we have at our disposal. The choice that we have made is to give the historical roots of the subject, and then go directly to the most recent results. The reader who is interested in a more complete narrative can refer to the 1974 survey article [35] by E.B. Yanovskaya, the 1981 survey article [8] by A. Irle and the 1995 survey article [31] by S. Simons.

Von Neumann's Results

In his investigation of *games of strategy*, J. von Neumann realized that, even though a two-person zero-sum game did not necessarily have a solution in *pure* strategies, it did have to have one in *mixed* strategies. Here is a statement of that seminal result ([19], translated into English in [21]):

Theorem 1 *(1928) Let A be an $m \times n$ matrix, and X and Y be the sets of nonnegative row and column vectors with unit sum. Then*

$$\min_{y\in Y} \max_{x\in X} xAy = \max_{x\in X} \min_{y\in Y} xAy.$$

Despite the fact that the statement of this result is quite elementary, the proof was quite sophisticated, and depended on an extremely ingenious induction argument. Nine years later, in [20], von Neumann showed that the bilinear character of Theorem 1 was not needed when he extended it as follows, using *Brouwer's fixed point theorem*:

Theorem 2 *(1937) Let X and Y be nonempty compact, convex subsets of Euclidean spaces, and $f\colon X \times Y \to \mathbf{R}$ be jointly continuous. Suppose that f is quasiconcave on X and quasiconvex on Y (see below). Then*

$$\min_Y \max_X f = \max_X \min_Y f.$$

When we say that f is *quasiconcave* on X, we mean that

- for all $y \in Y$ and $\lambda \in \mathbf{R}$, $GT(\lambda, y)$ is convex,

and when we say that f is *quasiconvex* on Y, we mean that

- for all $x \in X$ and $\lambda \in \mathbf{R}$, $LE(x, \lambda)$ is convex.

Here, $GT(\lambda, y)$ and $LE(x, \lambda)$ are 'level sets' associated with the function f. Specifically,

$$GT(\lambda, y) := \{x \in X\colon\ f(x,y) > \lambda\}$$

and

$$LE(x,\lambda) := \{y \in Y\colon\ f(x,y) \le \lambda\}.$$

In 1941, S. Kakutani [10] analyzed von Neumann's proof and, as a result, discovered the fixed point theorem that bears his name.

Infinite-Dimensional Results for Convex Sets

The first infinite-dimensional minimax theorem was proved in 1952 by K. Fan ([1]), who generalized Theorem 2 to the case when X and Y are compact, convex subsets of infinite-dimensional locally convex spaces, and the quasiconcave and quasiconvex conditions are somewhat relaxed. The result in this general line that has the simplest statement is that of M. Sion, who proved the following ([33]):

Theorem 3 *(1958) Let X be a convex subset of a linear topological space, Y be a compact convex subset of a linear topological space, and $f: X \times Y \to \boldsymbol{R}$ be upper semicontinuous on X and lower semicontinuous on Y. Suppose that f is quasiconcave on X and quasiconvex on Y. Then*

$$\min_Y \sup_X f = \sup_X \min_Y f.$$

When we say that f is 'upper semicontinuous on X' and 'lower semicontinuous on Y' we mean that, for all $y \in Y$, the map $x \longmapsto f(x, y)$ is *upper semicontinuous* and, for all $x \in X$, the map $y \longmapsto f(x, y)$ is *lower semicontinuous*. The importance of Sion's weakening of continuity to semicontinuity was that it indicated that many kinds of minimax problems have equivalent formulations in terms of subsets of X and Y, and led to Fan's 1972 work ([4]) on sets with convex sections and minimax inequalities, which has since found many applications in economic theory. Like Theorem 2, all these result relied ultimately on Brouwer's fixed point theorem (or the related *Knaster–Kuratowski–Mazurkiewicz lemma* (KKM lemma) on closed subsets of a finite-dimensional simplex).

Functional-Analytic Minimax Theorems

The first person to take minimax theorems out of the context of convex subsets of vector spaces, and their proofs (other than that of the matrix case discussed in Theorem 1) out of the context of fixed point theorems was Fan in 1953 ([2]). We present here a generalization of Fan's result due to H. König ([15]). König's proof depended on the Mazur–Orlicz version of the Hahn–Banach theorem (see Theorem 5 below).

Theorem 4 *(1968) Let X be a nonempty set and Y be a nonempty compact topological space. Let $f: X \times Y \to \boldsymbol{R}$ be lower semicontinuous on Y. Suppose that:*

- *for all $x_1, x_2 \in X$, there exists $x_3 \in X$ such that*

$$f(x_3, \cdot) \geq \frac{f(x_1, \cdot) + f(x_2, \cdot)}{2} \quad \text{on } Y;$$

- *for all $y_1, y_2 \in Y$, there exists $y_3 \in Y$ such that*

$$f(\cdot, y_3) \leq \frac{f(\cdot, y_1) + f(\cdot, y_2)}{2} \quad \text{on } X.$$

Then

$$\min_Y \sup_X f = \sup_X \min_Y f.$$

We give here the statement of the *Mazur–Orlicz version of the Hahn–Banach theorem*, since it is a very useful result and it not as well-known as it deserves to be.

Theorem 5 (*Mazur–Orlicz theorem*) *Let S be a sublinear functional on a real vector space E, and C be a nonempty convex subset of E. Then there exists a linear functional L on E such that $L \leq S$ on E and $\inf_C L = \inf_C S$.*

See [16,22] and [23] for applications of the Mazur–Orlicz theorem and the related 'sandwich theorem' to measure theory, Hardy algebra theory and the theory of flows in infinite networks.

The kind of minimax theorem discussed in this section (where X is not topologized) has turned out to be extremely useful in functional analysis, in particular in *convex analysis* and also in the theory of *monotone operators on a Banach space*. (See [32] for more details of these kinds of applications.)

Minimax Theorems that Depend on Connectedness

It was believed for some time that proofs of minimax theorems required either the fixed point machinery of algebraic topology, or the functional-analytic machinery of convexity. However, in 1959, W.-T. Wu proved the first minimax theorem in which the conditions of convexity were totally replaced by conditions related to *connectedness*. This line of research was continued by H. Tuy, L.L. Stachó, M.A. Geraghty with B.-L. Lin, and J. Kindler with R. Trost, whose results were all subsumed by a family of general topological minimax theorem established by König in [17]. Here is a typical result from [17]. In order to simplify the statements of this and some of our later results, we shall write $f_* := \sup_X \inf_Y f$. f_* is the 'lower value' of f. If $\lambda \in \mathbf{R}$, $V \subset Y$ and $W \subset X$, we write $GT(\lambda, V) := \bigcap_{y \in V} GT(\lambda, y)$ and $LE(W, \lambda) := \bigcap_{x \in W} LE(x, \lambda)$.

Theorem 6 (1992) *Let X be a connected topological space, Y be a compact topological space, and $f: X \times Y \to \boldsymbol{R}$ be upper semicontinuous on X and lower semicontinuous on Y. Let Λ be a nonempty subset of (f_*, ∞)*

such that $\inf \Lambda = f_*$ *and suppose that, for all* $\lambda \in \Lambda$, *for all nonempty subsets* V *of* Y, *and for all nonempty finite subsets* W *of* X,

$$GT(\lambda, V) \quad \textit{is connected in } X,$$

and

$$LE(W, \lambda) \quad \textit{is connected in } Y.$$

Then

$$\min_Y \sup_X f = \sup_X \min_Y f.$$

Mixed Minimax Theorems

In [34], F. Terkelsen proved the first *mixed minimax theorem*. We describe Terkelsen's result as 'mixed' since one of the conditions in it is taken from Theorem 4, and the other from Theorem 6:

Theorem 7 (1972) *Let* X *be a nonempty set and* Y *be a nonempty compact topological space. Let* $f: X \times Y \to \boldsymbol{R}$ *be lower semicontinuous on* Y. *Suppose that,*

- *for all* $x_1, x_2 \in X$ *there exists* $x_3 \in X$ *such that*

$$f(x_3, \cdot) \geq \frac{f(x_1, \cdot) + f(x_2, \cdot)}{2} \quad \textit{on } Y.$$

Suppose also that, for all $\lambda \in \boldsymbol{R}$ *and, for all nonempty finite subsets* W *of* X,

$$LE(W, \lambda) \textit{ is connected in } Y.$$

Then

$$\min_Y \sup_X f = \sup_X \min_Y f.$$

A Metaminimax Theorem

It was believed for some time that Brouwer's fixed point theorem or the Knaster–Kuratowski–Mazurkiewicz lemma was required to order to prove Sion's theorem, Theorem 3. However, in 1966, M.A. Ghouila-Houri ([7]) proved Theorem 3 using a simple combinatorial property of convex sets in finite-dimensional space. This was probably the first indication of the breakdown of the classification of minimax theorems as either of 'topological' or 'functional-analytic' type. Further indication of this breakdown was provided by Terkelsen's result, Theorem 7, and the subsequent 1982 results of I. Joó and Stachó ([9]), the 1985 and 1986 results of Geraghty and Lin ([5] and [6]), and the 1989 results of H. Komiya ([18]).

Kindler ([11]) was the first to realize (in 1990) that some abstract concept akin to connectedness might be involved in minimax theorems, even when the topological condition of connectedness was not explicitly assumed. This idea was pursued by Simons with the introduction in 1992 of the concept of *pseudoconnectedness*, which we will now describe. We say that sets H_0 and H_1 are *joined* by a set H if

$$H \subset H_0 \cup H_1, \quad H \cap H_0 \neq \emptyset$$

and

$$H \cap H_1 \neq \emptyset.$$

We say that a family $\mathcal{H}$ of sets is *pseudoconnected* if

$$H_0, H_1, H \in \mathcal{H} \quad \text{and} \quad H_0 \text{ and } H_1 \text{ joined by } H$$
$$\Downarrow$$
$$H_0 \cap H_1 \neq \emptyset.$$

Any family of closed connected subsets of a topological space is pseudoconnected. So also is any family of open connected subsets. However, pseudoconnectedness can be defined in the absence of any topological structure and, as we shall see in Theorem 8, is closely related to minimax theorems. Theorem 8 is the improvement of the result of [29] due to König (see [30]). We shall say that a subset W of X is *good* if

- W is finite; and
- for all $x \in X$, $LE(x, f_*) \cap LE(W, f_*) \neq \emptyset$.

Theorem 8 (1995) *Let* Y *be a topological space, and* Λ *be a nonempty subset of* $\boldsymbol{R}$ *such that* $\inf \Lambda = f_*$. *Suppose that, for all* $\lambda \in \Lambda$ *and for all good subsets* W *of* X,

- *for all* $x \in X$, $LE(x, \lambda)$ *is closed and compact;* $\{LE(x, \lambda) \cap LE(W, \lambda)\}_{x \in X}$ *is pseudoconnected; and*
- *for all* $x_0, x_1 \in X$, *there exists* $x \in X$ *such that* $LE(x_0, \lambda)$ *and* $LE(x_1, \lambda)$ *are joined by* $LE(x, \lambda) \cap LE(W, \lambda)$.

Then

$$\min_Y \sup_X f = \sup_X \min_Y f.$$

Theorem 8 is proved by induction on the cardinality of the good subsets of W. Given the obvious topological motivation behind the concept of pseudoconnectedness, it is hardly surprising that Theorem 8 implies Theorem 6. What is more unexpected is that Theorem 8 implies Theorems 4 and 7 also. We prefer to describe Theorem 8 as a *metaminimax theorem* rather than a *minimax theorem*, since it is frequently harder to prove that the conditions of Theorem 8 are satisfied in any particular case that it is to prove Theorem 8 itself. So Theorem 8 is really a device for obtaining minimax theorems rather than a minimax theorem in its own right.

More recent work by Kindler ([12,13] and [14]) on abstract intersection theorems has been at the interface between minimax theory and abstract set theory.

Minimax Theorems and Weak Compactness

There are close connections between minimax theorems and *weak compactness*. The following 'converse minimax theorem' was proved by Simons in [25]; this result also shows that there are limitations on the extent to which one can totally remove the assumption of compactness from minimax theorems.

Theorem 9 (1971) *Suppose that X is a nonempty bounded, convex, complete subset of a locally convex space E with dual space E^*, and*

$$\inf_{y\in Y}\sup_{x\in X}\langle x, y\rangle = \sup_{x\in X}\inf_{y\in Y}\langle x, y\rangle$$

whenever Y is a nonempty convex, equicontinuous, subset of E^*. Then X is weakly compact.

No compactness is assumed in the following, much harder, result (see [26]):

Theorem 10 (1972) *If X is a nonempty bounded, convex subset of a locally convex space E such that every element of the dual space E^* attains its supremum on X, and Y is any nonempty convex equicontinuous subset of E^*, then*

$$\inf_{y\in Y}\sup_{x\in X}\langle x, y\rangle = \sup_{x\in X}\inf_{y\in Y}\langle x, y\rangle .$$

If one now combines the results of Theorems 9 and 10, one can obtain a proof of the 'sup theorem' of R.C. James, one of the most beautiful results in functional analysis:

Theorem 11 (*James sup theorem*) *If C is a nonempty bounded closed convex subset of E, then C is $w(E, E^*)$-compact if and only if, for all $x^* \in E^*$, there exists $x \in C$ such that $\langle x, x^*\rangle = \max_C x^*$.*

James's theorem is not easy - the standard proof can be found in the paper [24] by J.D. Pryce.

See [31] for more details of the connections between minimax theorems and weak compactness.

Minimax Inequalities for Two or More Functions

Motivated by *Nash equilibrium* and the theory of *noncooperative games*, Fan generalized Theorem 2 to the case of more than one function. In particular, he proved in [3] the following two-function minimax inequality (since the compactness of X is not needed, this result can in fact be strengthened to include Sion's theorem, Theorem 3, by taking $g = f$):

Theorem 12 (1964) *Let X and Y be nonempty compact, convex subsets of topological vector spaces and f, g: $X \times Y \to \boldsymbol{R}$. Suppose that f is lower semicontinuous on Y and quasiconcave on X, g is upper semicontinuous on X and quasiconvex on Y, and*

$$f \le g \quad \text{on } X \times Y.$$

Then

$$\min_{Y}\sup_{X} f \le \sup_{X}\inf_{Y} g.$$

Fan (unpublished) and Simons (see [27]) generalized König's theorem, Theorem 4, with the following *two-function minimax inequality*:

Theorem 13 (1981) *Let X be a nonempty set, Y be a compact topological space and f, g: $X \times Y \to \boldsymbol{R}$. Suppose that f is lower semicontinuous on Y, and*

- *for all $y_1, y_2 \in Y$ there exists $y_3 \in Y$ such that*

$$f(\cdot, y_3) \le \frac{f(\cdot, y_1) + f(\cdot, y_2)}{2} \quad \text{on } X;$$

- *for all* $x_1, x_2 \in X$ *there exists* $x_3 \in X$ *such that*

$$g(x_3, \cdot) \geq \frac{g(x_1, \cdot) + g(x_2, \cdot)}{2} \quad \text{on } Y;$$

and

- $f \leq g$ *on* $X \times Y$.

Then

$$\min_Y \sup_X f \leq \sup_X \inf_Y g.$$

Theorems 12 and 13 both unify the theory of minimax theorems and the theory of *variational inequalities*. The curious feature about these two results is that they have 'opposite geometric pictures'. This question is discussed in [27] and [28]. The relationship between Theorem 12 and Brouwer's fixed point theorem is quite interesting. As we have already pointed out, Sion's theorem, Theorem 3, can be proved in an elementary fashion without recourse to fixed point related concepts. On the other hand, Theorem 12 can, in fact, be used to prove *Tychonoff's fixed point theorem*, which is itself a generalization of Brouwer's fixed point theorem. (See [3] for more details of this.)

A number of authors have proved minimax inequalities for more than two functions. See [31] for more details of these results.

Coincidence Theorems

A *coincidence theorem* is a theorem that asserts that if $S: X \to 2^Y$ and $T: Y \to 2^X$ have nonempty values and satisfy certain other conditions, then there exist $x_0 \in X$ and $y_0 \in Y$ such that $y_0 \in Sx_0$ and $x_0 \in Ty_0$. The connection with minimax theorems is as follows: Suppose that $\inf_Y \sup_X f \neq \sup_X \inf_Y f$. Then there exists $\lambda \in \mathbf{R}$ such that

$$\sup_X \inf_Y f < \lambda < \inf_Y \sup_X f.$$

Hence,

- for all $x \in X$ there exists $y \in Y$ such that $f(x, y) < \lambda$; and
- for all $y \in Y$ there exists $x \in X$ such that $f(x, y) > \lambda$.

Define $S: X \to 2^Y$ and $T: Y \to 2^X$ by

$$Sx := \{y \in Y: \ f(x, y) < \lambda\} \neq \emptyset$$

and

$$Tx := \{x \in X: \ f(x, y) > \lambda\} \neq \emptyset.$$

If S and T were to satisfy a coincidence theorem, then we would have $x_0 \in X$ and $y_0 \in Y$ such that

$$f(x_0, y_0) < \lambda \quad \text{and} \quad f(x_0, y_0) > \lambda,$$

which is clearly impossible. Thus this coincidence theorem would imply that

$$\inf_Y \sup_X f = \sup_X \inf_Y f.$$

The coincidence theorems known in algebraic topology consequently give rise to corresponding minimax theorems. There is a very extensive literature about coincidence theorems. See [31] for more details about this.

See also

- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Minimax: Directional Differentiability
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Quasigradient Methods in Minimax Problems

References

1. Fan K (1952) Fixed-point and minimax theorems in locally convex topological linear spaces. Proc Nat Acad Sci USA 38:121–126
2. Fan K (1953) Minimax theorems. Proc Nat Acad Sci USA 39:42–47
3. Fan K (1964) Sur un théorème minimax. CR Acad Sci Paris 259:3925–3928
4. Fan K (1972) A minimax inequality and its applications. In: Shisha O (ed) Inequalities, vol III. Acad. Press, New York, pp 103–113
5. Geraghty MA, Lin B-L (1985) Minimax theorems without linear structure. Linear Multilinear Algebra 17:171–180
6. Geraghty MA, Lin B-L (1986) Minimax theorems without convexity. Contemp Math 52:102–108
7. Ghouila-Houri MA (1966) Le théorème minimax de Sion. In: Theory of games. English Univ. Press, London, pp 123–129
8. Irle A (1981) Minimax theorems in convex situations. In: Moeschlin O, Pallaschke D (eds) Game Theory and Mathematical Economics. North-Holland, Amsterdam, pp 321–331
9. Joó I, Stachó LL (1982) A note on Ky Fan's minimax theorem. Acta Math Acad Sci Hung 39:401–407
10. Kakutani S (1941) A generalization of Brouwer's fixed-point theorem. Duke Math J 8:457–459

11. Kindler J (1990) On a minimax theorem of Terkelsen's. Arch Math 55:573–583
12. Kindler J (1993) Intersection theorems and minimax theorems based on connectedness. J Math Anal Appl 178:529–546
13. Kindler J (1994) Intersecting sets in midset spaces. I. Arch Math 62:49–57
14. Kindler J (1994) Intersecting sets in midset spaces. II. Arch Math 62:168–176
15. König H (1968) Über das Von Neumannsche Minimax-Theorem. Arch Math 19:482–487
16. König H (1970) On certain applications of the Hahn-Banach and minimax theorems. Arch Math 21:583–591
17. König H (1992) A general minimax theorem based on connectedness. Arch Math 59:55–64
18. Komiya H (1989) On minimax theorems. Bull Inst Math Acad Sinica 17:171–178
19. Neumann Jvon (1928) Zur Theorie der Gesellschaftspiele. MATH-A 100:295–320
20. Neumann Jvon (1937) Ueber ein ökonomisches Gleichungssystem und eine Verallgemeinerung des Brouwerschen Fixpunktsatzes. Ergebn Math Kolloq Wien 8:73–83
21. Neumann Jvon (1959) On the theory of games of strategy. In: Tucker AW, Luce RD (eds) Contributions to the Theory of Games, vol 4, Princeton Univ. Press, Princeton, pp 13–42
22. Neumann M (1989) Some unexpected applications of the sandwich theorem. In: Proc. Conf. Optimization and Convex Analysis, Univ. Mississippi
23. Neumann M (1991) Generalized convexity and the Mazur–Orlicz theorem. In: Proc. Orlicz Memorial Conf., Univ. Mississippi
24. Pryce JD (1966) Weak compactness in locally convex spaces. Proc Amer Math Soc 17:148–155
25. Simons S (1970/1) Critères de faible compacité en termes du théorème de minimax. Sém. Choquet 23:8
26. Simons S (1972) Maximinimax: minimax, and antiminimax theorems and a result of R.C. James. Pacific J Math 40:709–718
27. Simons S (1981) Minimax and variational inequalities: Are they or fixed point or Hahn–Banach type? In: Moeschlin O, Pallaschke D (eds) Game Theory and Mathematical Economics. North-Holland, Amsterdam, pp 379–388
28. Simons S (1986) Two-function minimax theorems and variational inequalities for functions on compact and noncompact sets with some comments on fixed-points theorems. Proc Symp Pure Math 45:377–392
29. Simons S (1994) A flexible minimax theorem. Acta Math Hungarica 63:119–132
30. Simons S (1995) Addendum to: A flexible minimax theorem. Acta Math Hungarica 69:359–360
31. Simons S (1995) Minimax theorems and their proofs. In: Du DZ, Pardalos PM (eds) Minimax and Applications. Kluwer, Dordrecht, pp 1–23
32. Simons S (1998) Minimax and monotonicity. Lecture Notes Math, vol 1693. Springer, Berlin
33. Sion M (1958) On general minimax theorems. Pacific J Math 8:171–176
34. Terkelsen F (1972) Some minimax theorems. Math Scand 31:405–413
35. Yanovskaya EB (1974) Infinite zero-sum two-person games. J Soviet Math 2:520–541

Minimum Concave Transportation Problems

MCTP

Bruce W. Lamar
Economic and Decision Analysis Center, The MITRE Corp., Bedford, USA

MSC2000: 90C26, 90C35, 90B06, 90B10

Article Outline

Keywords
See also
References

Keywords

Flows in networks; Global optimization; Nonconvex programming; Fixed charge transportation problem

The minimum concave transportation problem MCTP concerns the least cost method of carrying flow on a bipartite network in which the marginal cost for an arc is a nonincreasing function of the flow on that arc. A *bipartite network* contains source nodes and sink nodes, but no transshipment (i. e., intermediate) nodes. The MCTP can be formulated as

$$\min \sum_{(i,j)\in A} \phi_{ij}(x_{ij}) \tag{1}$$

subject to:

$$\sum_{j\in N} x_{ij} = s_i, \quad \forall i \in M, \tag{2}$$

$$\sum_{i\in M} x_{ij} = d_j, \quad \forall j \in N, \tag{3}$$

$$x_{ij} \geq 0, \quad \forall (i,j) \in A, \tag{4}$$

where M is the set of source nodes; N is the set of sink nodes; s_i is the supply at source node i, d_j is the demand at sink node j; $A = \{(i, j) : i \in M, j \in N\}$ is the (directed) arc set; x_{ij} is the flow carried on arc (i, j); and $\phi_{ij}(x_{ij})$ is the concave cost function for arc (i, j). Objective function (1) minimizes total costs; constraints (2) balance flow at the source nodes; and constraints (3) balance flow at the sink nodes. If $\sum_{i \in M} s_i$ is less (greater) than $\sum_{j \in N} d_j$, then a dummy source (sink) node can be added to set M (N).

MCTPs arise naturally in distribution problems involving shipments sent directly from supply points to demand points in which the transportation costs exhibit economies of scale [21]. However, the MCTP is not limited to this class of problems. Specifically, any network flow problem with arc cost functions that are not concave can be converted to a network flow problem on an expanded network whose arc cost functions are all concave [16]. Then, the expanded network can be converted to a bipartite network by replacing each transshipment node with a source node and a sink node. Arc flow capacities can be removed by adding additional source nodes, one for each capacitated arc [19,23].

The fixed charge transportation problem FCTP is a type of MCTP in which the cost function $\phi_{ij}(x_{ij})$ for each arc $(i, j) \in A$ is of the form

$$\phi_{ij}(x_{ij}) = \begin{cases} 0 & \text{if } x_{ij} = 0, \\ f_{ij} + g_{ij} \cdot x_{ij} & \text{if } x_{ij} > 0, \end{cases} \tag{5}$$

where f_{ij} and g_{ij} are coefficients with $f_{ij} \geq 0$. FCTPs are commonly used to model network flow problems involving setup costs [9]. Furthermore, a variety of combinatorial problems can be converted to FCTPs. For instance, consider the 0–1 knapsack problem KP. The KP is formulated as

$$\max \sum_{k=1}^{n} c_k \cdot y_k \tag{6}$$

subject to:

$$\sum_{k=1}^{n} a_k \cdot y_k \leq b, \tag{7}$$

$$y_k \in \{0, 1\}, \quad \text{for } k = 1, \ldots, n, \tag{8}$$

with $a_k \geq 0$ and $c_k \geq 0$ for $k = 1, \ldots, n$. The KP can be converted to a FCTP with two source nodes and $n + 1$ sink nodes. Define $a_{n+1} = b$ and $c_{n+1} = 0$. Then, the network is specified as $M = \{1, 2\}$, $N = \{1, \ldots, n + 1\}$, $s_1 = b$, $s_2 = \sum_{k=1}^{n} a_k$, and $d_j = a_j$ for $j = 1, \ldots, n + 1$; and the cost function is of the form of (5) where, for each arc $(i, j) \in A$, the coefficients f_{ij} and g_{ij} are given by

$$f_{ij} = \begin{cases} \sum_{k=1}^{n} c_k & \text{if } j = 1, \ldots, n, \\ 0 & \text{if } j = n + 1, \end{cases} \tag{9}$$

$$g_{ij} = \begin{cases} -\frac{c_j}{a_j} & \text{if } i = 1, \\ 0 & \text{if } i = 2. \end{cases} \tag{10}$$

For $j = 1, \ldots, n$ sink node j has two incoming arcs, exactly one of which will have nonzero flow in the optimal solution to the FCTP. If $x^*_{1j} > 0$ in the FCTP, then $y^*_j = 1$ in the KP. If $x^*_{2j} > 0$ in the FCTP, then $y^*_j = 0$ in the KP.

One consequence of this result is that *any* integer programming problem with integer coefficients can (in principle) be formulated and solved as a FCTP by first converting the integer program to a KP [10].

Exact solution methods for the MCTP are predominately branch and bound enumeration procedures [2,3,4,6,8,11,12,15]. Binary partitioning is used for the FCTP; and interval partitioning is used for the MCTP with arbitrary concave arc cost functions. Finite convergence of the method was shown by R.M. Soland [22]. The convex envelope of the cost function $\phi_{ij}(x_{ij})$ is an affine function. Hence, a subproblem in the branch and bound procedure can be solved efficiently as a linear transportation problem (LTP) [1]. Fathoming techniques (such as 'up and down penalties' and 'capacity improvement') based on post-optimality analysis of the LTP facilitate the branch and bound procedure for the MCTP [2,3,18,20]. The LTP is also used in approximate solution methods for the MCTP which rely on successive linearizations of the concave cost function, $\phi_{ij}(x_{ij})$ [5,13,14].

Test problems for the MCTP are given in [7,8,12,17,20].

See also

- ► Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ► Concave Programming
- ► Motzkin Transposition Theorem

► Multi-index Transportation Problems
► Stochastic Transportation and Location Problems

References

1. Balinski ML (1961) Fixed-cost transportation problems. Naval Res Logist 8:41–54
2. Barr RS, Glover F, Klingman D (1981) A new optimization method for large scale fixed charge transportation problems. Oper Res 29:448–463
3. Bell GB, Lamar BW (1997) Solution methods for nonconvex network problems. In: Pardalos PM Hearn DW, Hager WW (eds) Network Optimization. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 32–50
4. Cabot AV, Erenguc SS (1984) Some branch-and-bound procedures for fixed-cost transportation problems. Naval Res Logist 31:145–154
5. Diaby M (1991) Successive linear approximation procedure for generalized fixed-charge transportation problems. J Oper Res Soc 42:991–1001
6. Florian M, Robilland P (1971) An implicit enumeration algorithm for the concave cost network flow problem. Managem Sci 18:184–193
7. Floudas CA, Pardalos PM (1990) A collection of test problems for constrained global optimization Algorithms. Lecture Notes Computer Sci, vol 455. Springer, Berlin
8. Gray P (1971) Exact solution of the fixed-charge transportation problem. Oper Res 19:1529–1538
9. Guisewite GM, Pardalos PM (1990) Minimum concave-cost network flow problems: Applications, complexity, and algorithms. Ann Oper Res 25:75–100
10. Kendall KE, Zoints S (1977) Solving integer programming problems by aggregating constraints. Oper Res 25:346–351
11. Kennington J (1976) The fixed-charge transportation problem: A computational study with a branch-and-bound code. AIIE Trans 8:241–247
12. Kennington J, Unger VE (1976) A new branch-and-bound algorithm for the fixed charge transportation problem. Managem Sci 22:1116–1126
13. Khang DB, Fujiwara O (1991) Approximate solutions of capacitated fixed-charge minimum cost network flow problems. Networks 21:689–704
14. Kim D, Pardalos PM (1999) A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. Oper Res Lett 24:195–203
15. Lamar BW (1993) An improved branch and bound algorithm for minimum concave cost network flow problems. J Global Optim 3:261–287
16. Lamar BW (1993) A method for solving network flow problems with general nonlinear arc costs. In: Du D-Z and Pardalos PM (eds) Network Optimization Problems: Algorithms, Applications, and Complexity. World Sci., Singapore, pp 147–167
17. Lamar BW, Wallace CA (1996) A comparison of conditional penalties for the fixed charge transportation problem. Techn. Report Dept. Management Univ. Canterbury
18. Lamar BW, Wallace CA (1997) Revised-modified penalties for fixed charge transportation problems. Managem Sci 43:1431–1436
19. Lawler EL (1976) Combinatorial optimization: Networks and matroids. Holt, Rinehart and Winston, New York
20. Palekar US, Karwan MH, Zionts S (1990) A branch-and-bound method for the fixed charge transportation problem. Managem Sci 36:1092–1105
21. Rech P, Barton LG (1970) A non-convex transportation algorithm. In: Beale EML (ed) Applications of Mathematical Programming Techniques. English Univ. Press, London
22. Soland RM (1974) Optimal facility location with concave costs. Oper Res 22:373–382
23. Wagner HM (1959) On a class of capacitated transportation problems. Managem Sci 5:304–318

Minimum Cost Flow Problem

RAVINDRA K. AHUJA[1], THOMAS L. MAGNANTI[2], JAMES B. ORLIN[3]
[1] Department Industrial and Systems Engineering, University Florida, Gainesville, USA
[2] Sloan School of Management and Department Electrical Engineering and Computer Sci., Massachusetts Institute Technol., Cambridge, USA
[3] Sloan School of Management, Massachusetts Institute Technol., Cambridge, USA

MSC2000: 90C35

Article Outline

Keywords
Applications
  Distribution Problems
  Airplane Hopping Problem
  Directed Chinese Postman Problem
Preliminaries
  Assumptions
  Graph Notation
  Residual Network
  Order Notation
Cycle-Canceling Algorithm
Successive Shortest Path Algorithm
Network Simplex Algorithm
See also
References

Keywords

Network; Minimum cost flow problem; Cycle-canceling algorithm; Successive shortest path algorithm; Network simplex algorithm

The *minimum cost flow problem* seeks a least cost shipment of a commodity through a network to satisfy demands at certain nodes by available supplies at other nodes. This problem has many, varied applications: the distribution of a product from manufacturing plants to warehouses, or from warehouses to retailers; the flow of raw material and intermediate goods through various machining stations in a production line; the routing of automobiles through an urban street network; and the routing of calls through the telephone system. The minimum cost flow problem also has many less direct applications. In this article, we briefly introduce the theory, algorithms and applications of the minimum cost flow problem. [1] contains much additional material on this topic.

Let $G = (N, A)$ be a *directed network* defined by a set N of n *nodes* and a set A of m *directed arcs*. Each arc $(i, j) \in A$ has an associated *cost* c_{ij} that denotes the cost per unit flow on that arc. We assume that the flow cost varies linearly with the amount of flow. Each arc $(i, j) \in A$ has an associated *capacity* u_{ij} denoting the maximum amount that can flow on this arc, and a lower bound l_{ij} that denotes the minimum amount that must flow on the arc. We assume that the capacity and flow lower bound for each arc (i, j) are integers. We associate with each node $i \in N$ an integer $b(i)$ representing its supply/demand. If $b(i) > 0$, node i is a *supply node*; if $b(i) < 0$, then node i is a *demand node* with a demand of $-b(i)$; and if $b(i) = 0$, then node i is a *transshipment node*. We assume that $\sum_{i \in N} b(i) = 0$. The decision variables x_{ij} are arc flows defined for each arc $(i, j) \in A$.

The minimum cost flow problem is an optimization model formulated as follows:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

subject to

$$\sum_{\{j:\ (i,j)\in A\}} x_{ij} - \sum_{\{j:\ (j,i)\in A\}} x_{ji} = b(i), \quad \text{for all } i \in N, \tag{2}$$

$$l_{ij} \le x_{ij} \le u_{ij}, \quad \text{for all } (i, j) \in A. \tag{3}$$

We refer to the constraints (2) as the *mass balance constraints*. For a fixed node i, the first term in the constraint (2) represents the total outflow of node i and the second term represents the total inflow of node i. The mass balance constraints state that outflow minus inflow must equal the supply/demand of each node. The flow must also satisfy the lower bound and capacity constraints (3), which we refer to as *flow bound constraints*.

This article is organized as follows. To help in understanding the applicability of the minimum cost flow problem, we begin in Section 2 by describing several applications. In Section 3, we present preliminary material needed in the subsequent sections. We next discuss algorithms for the minimum cost flow problem, describing the cycle-canceling algorithm in Section 4 and the successive shortest path algorithm in Section 5. The cycle-canceling algorithm identifies negative cost cycles in the network and augments flows along them. The successive shortest path algorithm augments flow along shortest cost augmenting paths from the supply nodes to the demand nodes. In Section 6, we describe the network simplex algorithm.

Applications

Minimum cost flow problems arise in almost all industries, including agriculture, communications, defense, education, energy, health care, manufacturing, medicine, retailing, and transportation. Indeed, minimum cost flow problems are pervasive in practice. In this section, by considering a few selected applications that arise in distribution systems planning, capacity planning, and vehicle routing, we give a passing glimpse of these applications.

Distribution Problems

A large class of network flow problems center around distribution applications. One core model is often described in terms of shipments from plants to warehouses (or, alternatively, from warehouses to retailers). Suppose a firm has p plants with known supplies and q warehouses with known demands. It wishes to identify a flow that satisfies the demands at the warehouses from the available supplies at the plants and that minimizes

its shipping costs. This problem is a well-known special case of the minimum cost flow problem, known as the *transportation problem*. We next describe in more detail a slight generalization of this model that also incorporates manufacturing costs at the plants.

A car manufacturer has several manufacturing plants and produces several car models at each plant that it then ships to geographically dispersed retail centers throughout the country. Each retail center requests a specific number of cars of each model. The firm must determine the production plan of each model at each plant and a shipping pattern that satisfies the demand of each retail center while minimizing the overall cost of production and transportation.

We describe this formulation through an example. Figure 1 illustrates a situation with two manufacturing plants, two retailers, and three car models. This model has four types of nodes:

i) *plant nodes*, representing various plants;
ii) *plant/model nodes*, corresponding to each model made at a plant;
iii) *retailer/model nodes*, corresponding to the models required by each retailer; and
iv) *retailer nodes* corresponding to each retailer.

The network contains three types of arcs:

i) production arcs;
ii) transportation arcs; and
iii) demand arcs.

The production arcs connect a plant node to a plant/model node; the cost of this arc is the cost of producing the model at that plant. We might place lower and upper bounds on production arcs to control for the minimum and maximum production of each particular car model at the plants. Transportation arcs connect plant/model nodes to retailer/model nodes; the cost of any such arc is the total cost of shipping one car from the manufacturing plant to the retail center. The transportation arcs might have lower or upper bounds imposed upon their flows to model contractual agreements with shippers or capacities imposed upon any distribution channel. Finally, demand arcs connect retailer/model nodes to the retailer nodes. These arcs have zero costs and positive lower bounds that equal the demand of that model at that retail center.

The production and shipping schedules for the automobile company correspond in a one-to-one fashion with the feasible flows in this network model. Consequently, a minimum cost flow provides an optimal production and shipping schedule.

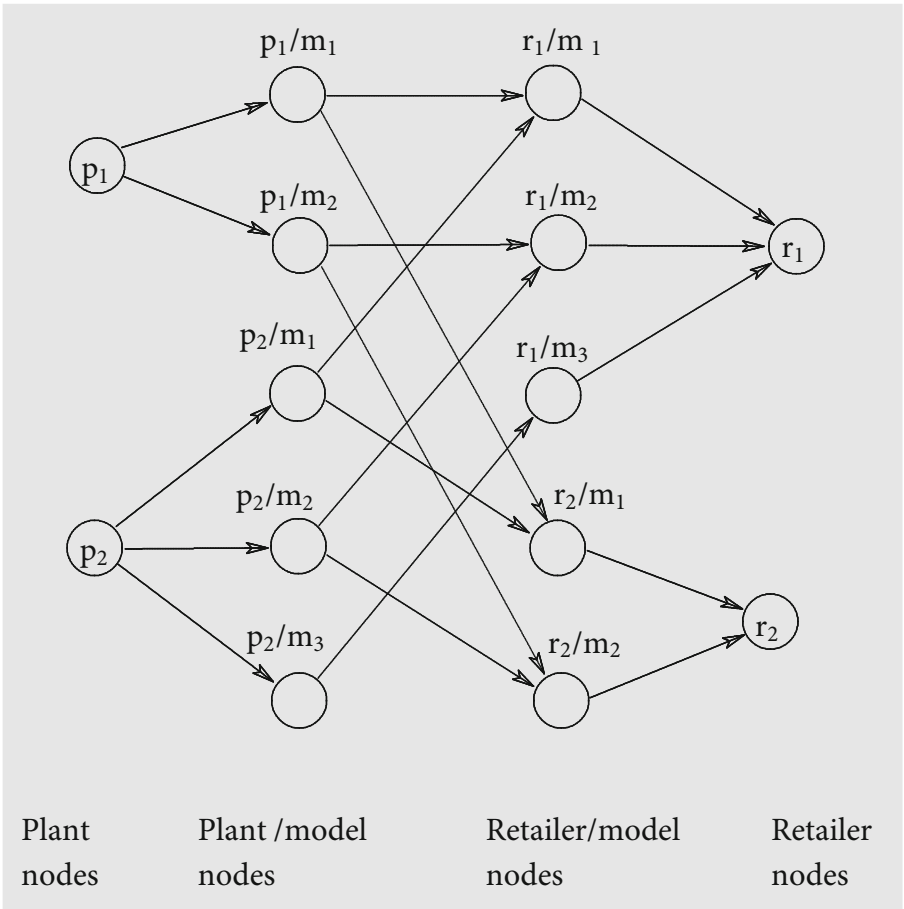
Airplane Hopping Problem

A small commuter airline uses a plane, with a capacity to carry at most p passengers, on a 'hopping flight' as shown in Fig. 2a). The hopping flight visits the cities $1, \ldots, n$, in a fixed sequence. The plane can pick up passengers at any node and drop them off at any other node. Let b_{ij} denote the number of passengers available at node i who want to go to node j, and let f_{ij} denote the fare per passenger from node i to node j. The airline would like to determine the number of passengers that the plane should carry between the various origins to destinations in order to maximize the total fare per trip while never exceeding the plane's capacity.

Figure 2b) shows a minimum cost flow formulation of this hopping plane flight problem. The network contains data for only those arcs with nonzero costs and with finite capacities: any arc listed without an associated cost has a zero cost; any arc listed without an associated capacity has an infinite capacity. Consider, for example, node 1. Three types of passengers are available at node 1: those whose destination is node 2, node 3 or node 4. We represent these three types of passengers in a new derived network by the nodes 1 – 2, 1 – 3 and 1 – 4 with supplies b_{12}, b_{13} and b_{14}. A passenger available at any such node, say 1 – 3, could board the plane at its origin node represented by flowing through the arc (1 – 3, 1) and incurring a cost of $-f_{13}$ units (or profit of f_{13} units). Or, the passenger might never board the plane, which we represent by the flow through the arc (1 – 3, 3). It is easy to establish a one-to-one correspondence between feasible flows in Fig. 2b) and feasible loading of the plane with passengers. Consequently, a minimum cost flow in Fig. 2b) will prescribe a most profitable loading of the plane.

Directed Chinese Postman Problem

The *directed Chinese postman problem* is a generic routing problem that can be stated as follows. In a directed network $G = (N, A)$ in which each arc (i, j) has an associated cost c_{ij}, we wish to identify a walk of minimum cost that starts at some node (the post office), visits each arc of the network at least once, and returns to the starting point (see the next Section for the def-



Minimum Cost Flow Problem, Figure 1
Formulating the production-distribution problem

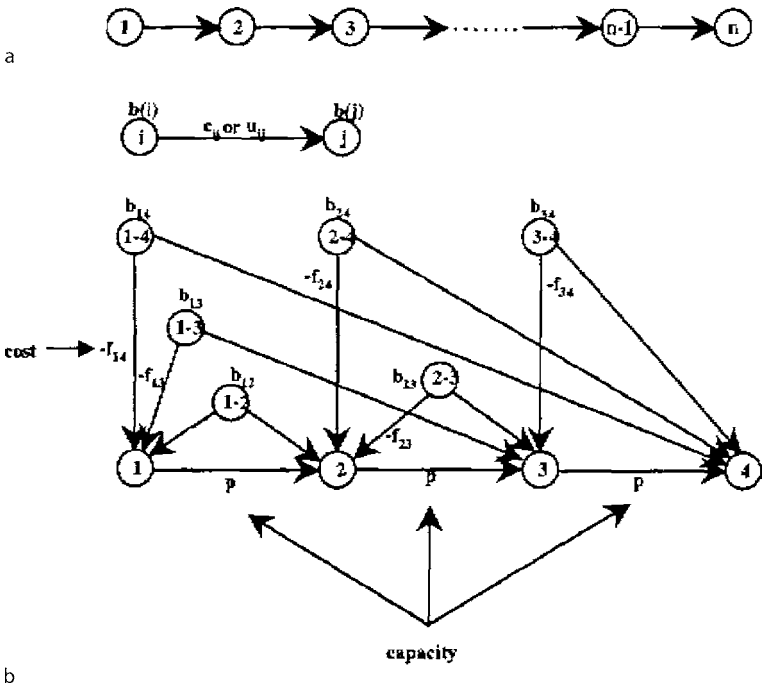
inition of a walk). This problem has become known as the Chinese postman problem because a Chinese mathematician, K. Mei-Ko, first discussed it. The Chinese postman problem arises in other settings as well; for instance, patrolling streets by police, routing street sweepers and household refuse collection vehicles, fuel oil delivery to households, and spraying roads with sand during snowstorms. The directed Chinese postman problem assumes that all arcs are directed, that is, the postal carrier can traverse an arc in only one direction (like one-way streets).

In the directed Chinese postman problem, we are interested in a closed (directed) walk that traverses each arc of the network at least once. The network might not contain any such walk. It is easy to show that a network contains a desired walk if and only if the network is *strongly connected*, that is, every node in the network is reachable from every other node via a directed path. Simple graph search algorithms are able to determine whether the network is strongly connected, and we shall therefore assume that the network is strongly connected.

In an optimal walk, a postal carrier might traverse arcs more than once. The minimum length walk minimizes the sum of lengths of the repeated arcs. Let x_{ij} denote the number of times the postal carrier traverses arc (i, j) in a walk. Any carrier walk must satisfy the following conditions:

$$\sum_{\{j:\ (i,j)\in A\}} x_{ij} - \sum_{\{j:\ (j,i)\in A\}} x_{ji} = 0 \quad \text{for all } i \in N, \qquad (4)$$

$$x_{ij} \geq 1 \quad \text{for all } (i, j) \in A. \qquad (5)$$



Minimum Cost Flow Problem, Figure 2
Formulation of the hopping plane flight problem as a minimum cost flow problem

The constraints (4) state that the carrier enters a node the same number of times that he or she leaves it. The constraints (5) state that the carrier must visit each arc at least once. Any solution x satisfying the system (4)–(5) defines a carrier's walk. We can construct a walk in the following manner. Given a flow x_{ij}, we replace each arc (i, j) with x_{ij} copies of the arc, each arc carrying a unit flow. In the resulting network, say $G' = (N, A')$, each node has the same number of outgoing arcs as it has the incoming arcs. It is possible to decompose this network into at most $m/2$ arc-disjoint directed cycles (by walking along an arc (i, j) from some node i with $x_{ij} > 0$, leaving an node each time we enter it until we repeat a node). We can connect these cycles together to form a closed walk of the carrier.

The preceding discussion shows that the solution x defined by a feasible walk for the carrier satisfies conditions (4)–(5), and, conversely, every feasible solution of system (4)–(5) defines a walk of the postman. The length of a walk defined by the solution x equals $\sum_{(i,j) \in A} c_{ij}x_{ij}$. This problem is an instance of the minimum cost flow problem.

Preliminaries

In this Section, we discuss some preliminary material required in the following sections.

Assumptions

We consider the minimum cost flow problem subject to the following six assumptions:

1) $l_{ij} = 0$ for each $(i, j) \in A$;
2) all data (cost, supply/demand, and capacity) are integral;
3) all arc costs are nonnegative;
4) for any pair of nodes i and j, the network does not contain both the arcs (i, j) and (j, i);
5) the minimum cost flow problem has a feasible solution; and
6) the network contains a directed path of sufficiently large capacity between every pair of nodes.

It is possible to show that none of these assumptions, except 2), restricts the generality of our development. We impose them just to simply our discussion.

Graph Notation

We use standard graph notation. A directed graph $G = (N, A)$ consists of a set N of nodes and a set A of arcs. A directed arc (i, j) has two *endpoints*, i and j. An arc (i, j) is *incident* to nodes i and j. The arc (i, j) is an *outgoing arc* of node i and an *incoming arc* of node j. A *walk* in a directed graph $G = (N, A)$ is a sequence of nodes and arcs $i_1, a_1, i_2, a_2, \ldots, i_r$ satisfying the property that for all $1 \leq k \leq r-1$, either $a_k = (i_k, i_{k+1}) \in A$ or $a_k = (i_{k+1}, i_k) \in A$. We sometimes refer to a walk as a *sequence of arcs* (or nodes) without any explicit mention of the nodes (or arcs). A *directed walk* is an oriented version of the walk in the sense that for any two consecutive nodes i_k and i_{k+1} on the walk, $a_k = (i_k, i_{k+1}) \in A$. A *path* is a walk without any repetition of nodes, and a *directed path* is a directed walk without any repetition of nodes. A *cycle* is a path $i_1, i_2, \ldots, i_r$ together with the arc (i_r, i_1) or (i_1, i_r). A directed cycle is a directed path $i_1, i_2, \ldots, i_r$ together with the arc (i_r, i_1). A spanning tree of a directed graph G is a subgraph $G' = (N, A')$ with $A' \subseteq A$ that is connected (that is, contains a path between every pair of nodes) and contains no cycle.

Residual Network

The algorithms described in this article rely on the concept of a *residual network* $G(x)$ corresponding to a flow x. For each arc $(i, j) \in A$, the residual network contains two arcs (i, j) and (j, i). The arc (i, j) has cost c_{ij} and *residual capacity* $r_{ij} = u_{ij} - x_{ij}$, and the arc (j, i) has cost $c_{ji} = -c_{ij}$ and residual capacity $r_{ji} = x_{ij}$. The residual network consists of arcs with positive residual capacity. If $(i, j) \in A$, then sending flow on arc (j, i) in $G(x)$ corresponds to decreasing flow on arc (i, j); for this reason, the cost of arc (j, i) is the negative of the cost of arc (i, j). These conventions show how to determine the residual network $G(x)$ corresponding to any flow x. We can also determine a flow x from the residual network $G(x)$ as follows. If $r_{ij} > 0$, then using the definition of residual capacities and Assumption 4), we set $x_{ij} = u_{ij} - r_{ij}$ if $(i, j) \in A$, and $x_{ji} = r_{ij}$ otherwise. We define the *cost of a directed cycle* W in the residual network $G(x)$ as $\sum_{(i,j) \in W} c_{ij}$.

Order Notation

In our discussion, we will use some well-known notation from the field of complexity theory. We say that an algorithm for a problem $\mathcal{P}$ is an $O(n^3)$ algorithm, or has a *worst-case complexity* of $O(n^3)$, if it is possible to solve any instance of $\mathcal{P}$ using a number of computations that is asymptotically bounded by some constant times the term n^3. We refer to an algorithm as a *polynomial time algorithm* if its worst-case running time is bounded by a polynomial function of the input size parameters, which for a minimum cost flow problem, are n, m, $\log C$ (the number of bits needed to specify the largest arc cost), and $\log U$ (the number of bits needed to specify the largest arc capacity). A *polynomial time algorithm* is either a *strongly polynomial time algorithm* (when the complexity terms involves only n and m, but not $\log C$ or $\log U$), or is a *weakly polynomial time algorithm* (when the complexity terms include $\log C$ or $\log U$ or both). We say that an algorithm is a *pseudopolynomial time algorithm* if its worst-case running time is bounded by a polynomial function of n, m and U. For example, an algorithm with worst-case complexity of $O(nm^2 \log n)$ is a strongly polynomial time algorithm, an algorithm with worst-case complexity $O(nm^2 \log U)$ is a weakly polynomial time algorithm, and an algorithm with worst-case complexity of $O(n^2 mU)$ is a pseudopolynomial time algorithm.

Cycle-Canceling Algorithm

In this Section, we describe the cycle-canceling algorithm, one of the more popular algorithms for solving the minimum cost flow problem. The algorithm sends flows (called *augmenting flows*) along directed cycles with negative cost (called *negative cycles*). The algorithm rests upon the following negative cycle optimality condition stated as follows.

Theorem 1 (Negative cycle optimality condition) *A feasible solution x^* is an optimal solution of the minimum cost flow problem if and only if the residual network $G(x^*)$ contains no negative cost (directed) cycle.*

It is easy to see the necessity of these conditions. If the residual network $G(x^*)$ contains a negative cycle (that is, a negative cost directed cycle), then by augmenting positive flow along this cycle, we can decrease the cost of the flow. Conversely, it is possible to show that if the residual network $G(x^*)$ does not contain any negative cost cycle, then x^* must be an optimal flow.

The negative cycle optimality condition suggests one simple algorithmic approach for solving the min-

```
BEGIN
  establish a feasible flow x in the network;
  WHILE G(x) contains a negative cycle DO
  BEGIN
    identify a negative cycle W;
    δ := min{r_ij : (i, j) ∈ W};
    augment δ units of flow in the cycle W
    and update G(x);
  END;
END
```

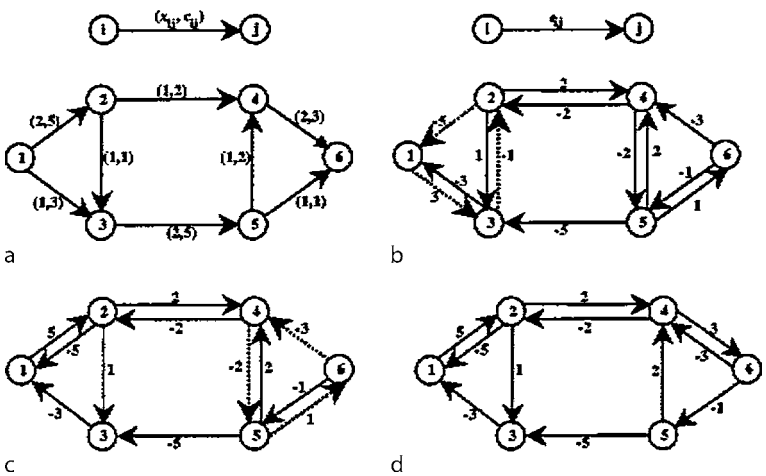
Minimum Cost Flow Problem, Figure 3
Cycle-canceling algorithm

imum cost flow problem, which we call the *cycle-canceling algorithm*. This algorithm maintains a feasible solution and at every iteration improves the objective function value. The algorithm first establishes a feasible flow x in the network by solving a related (and easily solved) problem known as the maximum flow problem. Then it iteratively finds negative cycles in the residual network and augments flows on these cycles. The algorithm terminates when the residual network contains no negative cost directed cycle. Theorem 1 implies that when the algorithm terminates, it has found a minimum cost flow. Figure 3a specifies this generic version of the cycle-canceling algorithm.

The numerical example shown in Fig. 4a) illustrates the cycle-canceling algorithm. This figure shows the arc costs and the starting feasible flow in the network. Each arc in the network has a capacity of 2 units. Figure 4b) shows the residual network corresponding to the initial flow. We do not show the residual capacities of the arcs in Fig. 4b) since they are implicit in the network structure. If the residual network contains both arcs (i, j) and (j, i) for any pair i and j of nodes, then both have residual capacity equal to 1; and if the residual network contains only one arc, then its capacity is 2 (this observation uses the fact that each arc capacity equals 2). The residual network shown in Fig. 4b) contains a negative cycle 1 – 3 – 2 – 1 with cost – 3. By augmenting a unit flow along this cycle, we obtain the residual network shown in Fig. 4c). The residual network shown in Fig. 4c) contains a negative cycle 6 – 4 – 5 – 6 with cost – 4. We augment unit flow along this cycle, producing the residual network shown in Fig. 4d), which contain no negative cycle. Given the optimal residual network, we can determine optimal flow using the method described in the previous Section.

A byproduct of the cycle-canceling algorithm is the following important result.

Theorem 2 (Integrality property) *If all arc capacities and supply/demands of nodes are integer, then the minimum cost flow problem always has an integer minimum cost flow.*



Minimum Cost Flow Problem, Figure 4
Illustration of the cycle-canceling algorithm. a) the original network with flow *x* and arc costs; b) the residual network *G*(*x*); c) the residual network after augmenting a unit of flow along the cycle 2 – 1 – 3 – 2; d) the residual network after augmenting a unit of flow along the cycle 4 – 5 – 6 – 4

This result follows from the fact that for problems with integer arc capacities and integer node supplies/demand, the cycle-canceling algorithm starts with an integer solution (which is provided by the maximum flow algorithm used to obtain the initial feasible flow) and at each iteration augments flow by an integral amount.

What is the worst-case computational requirement (complexity) of the cycle-canceling algorithm? The algorithm must repeatedly identify negative cycles in the residual network. We can identify a negative cycle in the residual network in $O(nm)$ time using a shortest path label-correcting algorithm [1]. How many times must the generic cycle-canceling algorithm perform this computation? For the minimum cost flow problem, mCU is an upper bound on the initial flow cost (since $c_{ij} \le C$ and $x_{ij} \le U$ for all $(i, j) \in A$) and $-mCU$ is a lower bound on the optimal flow cost (since $c_{ij} \ge -C$ and $x_{ij} \le U$ for all $(i, j) \in A$). Any iteration of the cycle-canceling algorithm changes the objective function value by an amount $\sum_{(i,j) \in W} c_{i,j})\, \delta$, which is strictly negative. Since we have assumed that the problem has integral data, the algorithm terminates within $O(mCU)$ iterations and runs in $O(nm^2\, CU)$ time, which is a pseudopolynomial running time.

The generic version of the cycle-canceling algorithm does not specify the order for selecting negative cycles from the network. Different rules for selecting negative cycles produce different versions of the algorithm, each with different worst-case and theoretical behavior. Two versions of the cycle-canceling algorithm are polynomial time implementations:

i) a version that augments flow in arc-disjoint negative cycles with the maximum improvement [2]; and
ii) a version that augments flow along a negative cycle with minimum mean cost, that is, the average cost per arc in the cycle [4]).

Successive Shortest Path Algorithm

The cycle-canceling algorithm maintains feasibility of the solution at every step and attempts to achieve optimality. In contrast, the successive shortest path algorithm maintains optimality of the solution at every step (that is, the condition that the residual network $G(x)$ contains no negative cost cycle) and strives to attain feasibility. It maintains a solution x, called a pseudoflow (see below), that is nonnegative and satisfies the arcs' flow capacity restrictions, but violates the mass balance constraints of the nodes. At each step, the algorithm selects a node k with excess supply (i. e., supply not yet sent to some demand node), a node l with unfulfilled demand, and sends flow from node k to node l along a shortest path in the residual network. The algorithm terminates when the current solution satisfies all the mass balance constraints.

To be more precise, a *pseudoflow* is a vector x satisfying only the capacity and nonnegativity constraints; it need not satisfy the mass balance constraints. For any pseudoflow x, we define the *imbalance* of node i as

$$e(i) = b(i) + \sum_{\{j,i) \in A\}} x_{ji} - \sum_{\{(i,j) \in A\}} x_{ij} \quad \text{for all } i \in N. \quad (6)$$

If $e(i) > 0$ for some node i, then we refer to $e(i)$ as the *excess* of node i; if $e(i) < 0$, then we refer to $-e(i)$ as the node's *deficit*. We refer to a node i with $e(i) = 0$ as *balanced*. Let E and D denote the sets of excess and deficit nodes in the network. Notice that $\sum_{i \in N} e(i) = \sum_{i \in N} b(i) = 0$, which implies that $\sum_{i \in E} e(i) = -\sum_{i \in D} e(i)$. Consequently, if the network contains an excess node, then it must also contain a deficit node. The residual network corresponding to a pseudoflow is defined in the same way that we define the residual network for a flow. The successive shortest path algorithm uses the following result.

Theorem 3 (Shortest augmenting path theorem) *Suppose a pseudoflow (or a flow) x satisfies the optimality conditions and we obtain x′ from x by sending flow along a shortest path from node k to some other node l in the residual network, then x′ also satisfies the optimality conditions.*

To prove this Theorem, we would show that if the residual network $G(x)$ contain no negative cycle, then augmenting flow along any shortest path does not introduce any negative cycle (we will not establish this result in this discussion). Figure 5 gives a formal description of the successive shortest path algorithm.

The numerical example shown in Fig. 6a) illustrates the successive shortest path algorithm. The algorithm starts with $x = 0$, and at this value of flow, the residual network is identical to the starting network. Just as we

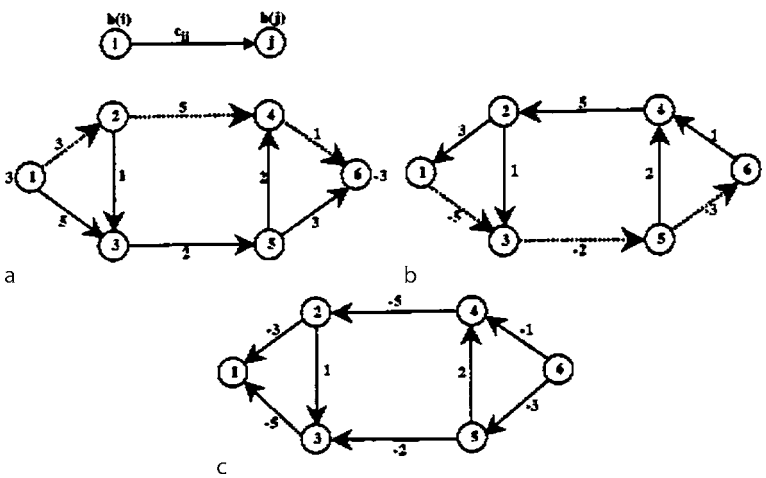
```
BEGIN
    x := 0;
    e(i) = b(i) for all i ∈ N;
    initialize the sets E and D;
    WHILE E ≠ ∅ DO
    BEGIN
        select a node k ∈ E and a node l ∈ D;
        identify a shortest path P in G(x) from
        node k to node l;
        δ := min[e(s), −e(t), min{r_ij : (i, j) ∈ P}];
        augment δ units of flow along the path P and
        update x and G(x);
    END
END
```

Minimum Cost Flow Problem, Figure 5
Successive shortest path algorithm

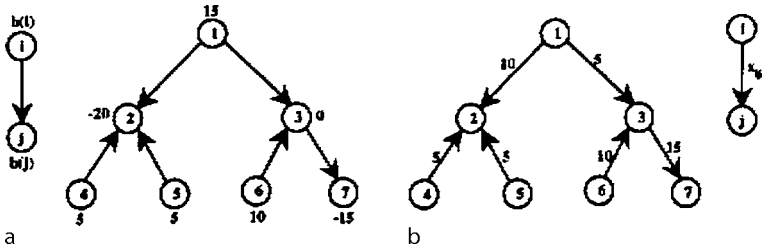
observed in Fig. 4, whenever the residual network contains both the arcs (i, j) and (j, i), the residual capacity of each arc is 1. If the residual network contains only one arc, (i, j) or (j, i), then its residual capacity is 2 units. For this problem, $E = \{1\}$ and $D = \{6\}$. In the residual network shown in Fig. 6a), the shortest path from node 1 to node 6 is 1 – 2 – 4 – 6 with cost equal to 9. The residual capacity of this path equals 2. Augmenting two units of flow along this path produces the residual network shown in Fig. 6b), and the next shortest path from node 1 to node 6 is 1 – 3 – 5 – 6 with cost equal to 10. The residual capacity of this path is 2 and we augment two unit of flow on it. At this point, the sets $E = D = \emptyset$, and the current solution solves the minimum cost flow problem.

To show that the algorithm correctly solves the minimum cost flow problem, we argue as follows. The algorithm starts with a flow $x = 0$ and the residual network $G(x)$ is identical to the original network. Assumption 3) implies that all arc costs are nonnegative. Consequently, the residual network $G(x)$ contains no negative cycle and so the flow vector x satisfies the negative cycle optimality conditions. Since the algorithm augments flow along a shortest path from excess nodes to deficit nodes, Theorem 3 implies that the pseudoflow maintained by the algorithm always satisfies the optimality conditions. Eventually, node excesses and deficits become zero; at this point, the solution maintained by the algorithm is an optimal flow.

What is the worst-case complexity of this algorithm? In each iteration, the algorithm reduces the excess of some node. Consequently, if U is an upper bound on the largest supply of any node, then the algorithm would terminate in at most nU iterations. We can determine a shortest path in $G(x)$ in $O(nm)$ time using a label-correcting shortest path algorithm [1]. Consequently, the running time of the successive shortest path algorithm is n^2mU.



Minimum Cost Flow Problem, Figure 6
Illustration of the successive shortest path algorithm. a) the residual network corresponding to $x = 0$; b) the residual network after augmenting 2 units of flow along the path 1 – 2 – 4 – 6; c) the residual network after augmenting 2 units of flow along the path 1 – 3 – 5 – 6



Minimum Cost Flow Problem, Figure 7
Computing flows for a spanning tree

The successive shortest path algorithm requires pseudopolynomial time to solve the minimum cost flow problem since it is polynomial in n, m and the largest supply U. This algorithm is, however, polynomial time for some special cases of the minimum cost flow problem (such as the assignment problem for which $U = 1$). Researchers have developed weakly polynomial time and strongly polynomial time versions of the successive shortest path algorithm; some notable implementations are due to [3] and [5].

Network Simplex Algorithm

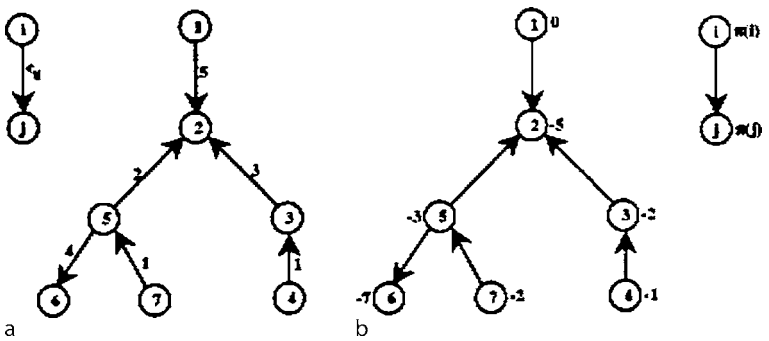
The network simplex algorithm for solving the minimum cost flow problem is an adaptation of the well-known simplex method for general linear programs. Because the minimum cost flow problem is a highly structured linear programming problem, when applied to it, the computations of the simplex method become considerably streamlined. In fact, we need not explicitly maintain the matrix representation (known as the simplex tableau) of the linear program and can perform all of the computations directly on the network. Rather than presenting the network simplex algorithm as a special case of the linear programming simplex method, we will develop it as a special case of the cycle-canceling algorithm described above. The primary advantage of our approach is that it permits the network simplex algorithm to be understood without relying on linear programming theory.

The network simplex algorithm maintains solutions called spanning tree solutions. A *spanning tree solution* partitions the arc set A into three subsets:

i) T, the arcs in the spanning tree;
ii) L, the nontree arcs whose flows are restricted to value zero;
iii) U, the nontree arcs whose flow values are restricted in value to the arcs' flow capacities.

We refer to the triple (T, L, U) as a *spanning tree structure*. Each spanning tree structure (T, L, U) has a unique solution that satisfies the mass balance constraints (2). To determine this solution, we set $x_{ij} = 0$ for all arcs $(i, j) \in L$, $x_{ij} = u_{ij}$ for all arcs $(i, j) \in U$, and then solve the mass balance equations (2) to determine the flow values for arcs in T.

To show that the flows on spanning tree arcs are unique, we use a numerical example. Consider the spanning tree T shown in Fig. 7a). Assume that $U = \varphi$, that is, all nontree arcs are at their lower bounds. Consider the leaf node 4 (a leaf node is a node with exactly one arc incident to it). Node 4 has a supply of 5 units and has only one arc (4, 2) incident to it. Consequently, arc (4, 2) must carry 5 units of flow. So we set $x_{42} = 5$, add 5 units to $b(2)$ (because it receives 5 units of flow sent from node 4), and delete arc (4, 2) from the tree. We now have a tree with one fewer node and next select another leaf node, node 5 with the supply of 5 units and the single arc (5, 2) incident to it. We set $x_{52} = 5$, again add 5 units to $b(2)$, and delete the arc (5, 2) from the tree. Now node 2 becomes a leaf node with modified supply/demand of $b(5) = -10$, implying that node 5 has an unfulfilled demand of 10 units. Node 2 has exactly one incoming arc (1, 2) and to meet the demand of 10 units of node 2, we must send 10 units of flow on this arc. We set $x_{12} = 10$, subtract 10 units from $b(1)$ (since node 1 sends 10 units), and delete the arc (1, 2) from the tree. We repeat this process until we have identified flow on all arcs in the tree. Figure 7b) shows the corresponding flow. Our discussion assumed that U is empty. If U were nonempty, we would first set $x_{ij} = u_{ij}$, add u_{ij} to $b(j)$, and subtract u_{ij} from $b(i)$ for each arc $(i, j) \in U$, and then apply the preceding method.



Minimum Cost Flow Problem, Figure 8
Computing node potentials for a spanning tree

We say a spanning tree structure is *feasible* if its associated spanning tree solution satisfies all of the arcs' flow bounds. We refer to a spanning tree structure as *optimal* if its associated spanning tree solution is an optimal solution of the minimum cost flow problem. We will now derive the optimality conditions for a spanning tree structure (T, L, U).

The network simplex algorithm augments flow along negative cycles. To identify negative cycles quickly, we use the concept of *node potentials*. We define node potentials $\pi(i)$ so that the reduced cost for any arc in the spanning tree T is zero. That is, that is, $c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j) = 0$ for each $(i, j) \in T$. With the help of an example, we show how to compute the vector π of node potentials. Consider the spanning tree shown in Fig. 8a) with arc costs as shown. The vector π has n variables and must satisfy $n - 1$ equations, one for each arc in the spanning tree. Therefore, we can assign one potential value arbitrary. We assume that $\pi(1) = 0$. Consider arc (1, 2) incident to node 1. The condition $c_{12}^{\pi} = c_{12} - \pi(1) + \pi(2) = 0$ yields $\pi(2) = -5$. We next consider arcs incident to node 2. Using the condition $c_{52}^{\pi} = c_{52} - \pi(5) + \pi(2) = 0$, we see that $\pi(5) = -3$, and the condition $c_{32}^{\pi} = c_{32} - \pi(3) + \pi(2) = 0$ shows that $\pi(3) = -2$. We repeat this process until we have identified potentials of all nodes in the tree T. Figure 8b) shows the corresponding node potentials.

Consider any nontree arc (k, l). Adding this arc to the tree T creates a unique cycle, which we denote as W_{kl}. We refer to W_{kl} as the *fundamental cycle* induced by the nontree arc (k, l). If $(k, l) \in L$, then we define the orientation of the fundamental cycle as in the direction of (k, l), and if $(k, l) \in U$, then we define the orientation opposite to that of (k, l). In other words, we define the orientation of the cycle in the direction of flow change permitted by the arc (k, l). We let $c(W_{kl})$ denote the change in the cost if we send one unit of flow on the cycle W_{kl} along its orientation. (Notice that because of flow bounds, we might not always be able to send flow along the cycle W_{kl}.) Let $\overline{W}_{kl}$ denote the set of forward arcs in W_{kl} (that is, those with the same orientation as (k, l)), and let $\underline{W}_{kl}$ denote the set of backward arcs in W_{kl} (that is, those with an opposite the orientation to arc (k, l)). Then, if we send one unit of flow along W_{kl}, then the flow on arcs in $\overline{W}_{kl}$ increases by one unit and the flow on arcs in $\underline{W}_{kl}$ decreases by one unit. Therefore,

$$c(W_{kl}) = \sum_{(i,j)\in W_{kl}} c_{ij} - \sum_{(i,j)\in \underline{W}_{kl}} c_{ij}.$$

Let $c^{\pi}(W_{kl})$ denote the change in the reduced costs if we send one unit of flow in the cycle W_{kl} along its orientation, that is,

$$c^{\pi}(W_{kl}) = \sum_{(i,j)\in \overline{W}_{kl}} c_{ij}^{\pi} - \sum_{(i,j)\in \underline{W}_{kl}} c_{ij}^{\pi}.$$

It is easy to show that $c^{\pi}(W_{kl}) = c(W_{kl})$. This result follows from the fact that when we substitute $c_{kl}^{\pi} = c_{ij} - \pi(i) + \pi(j)$ and add the reduced costs around any cycle, then the node potentials $\pi(i)$ cancel one another. Next notice that the manner we defined node potentials ensures that each arc in the fundamental cycle W_{kl} except the arc (k, l) has zero reduced cost. Consequently, if arc $(k, l) \in L$, then

$$c(W_{kl}) = c^{\pi}(W_{kl}) = c_{kl}^{\pi},$$

and if arc $(k, l) \in U$, then

$$c(W_{kl}) = c^{\pi}(W_{kl}) = -c^{\pi}_{kl}.$$

This observation and the negative cycle optimality condition (Theorem 1) implies that for a spanning tree solution to be optimal, it must satisfy the following necessary conditions:

$$c^{\pi}_{kl} \geq 0 \quad \text{for every arc } (i, j) \in L, \tag{7}$$

$$c^{\pi}_{kl} \leq 0 \quad \text{for every arc } (i, j) \in U. \tag{8}$$

It is possible to show that these conditions are also sufficient for optimality; that is, if any spanning tree solution satisfies the conditions (7)–(8), then it solves the minimum cost flow problem.

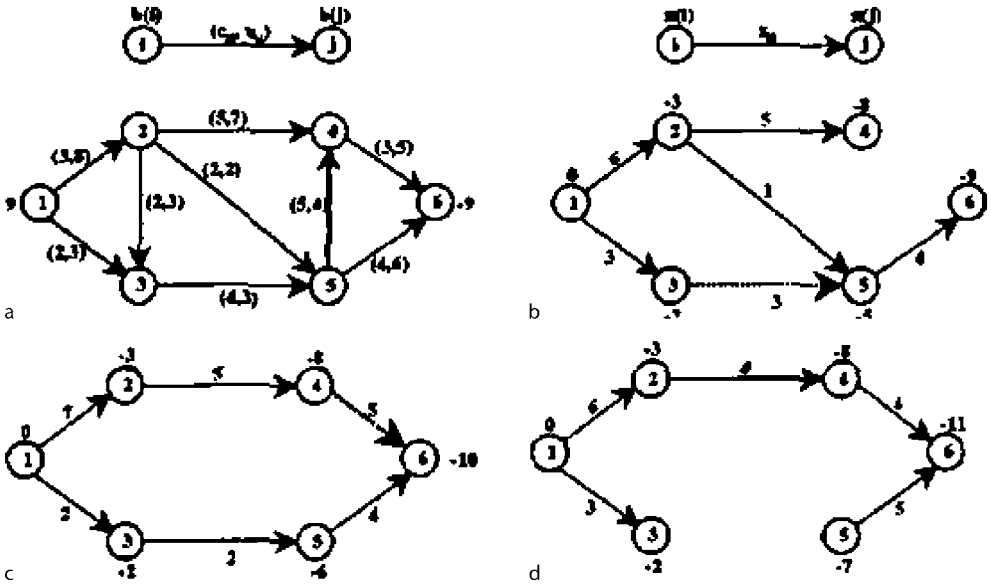
We now have all the necessary ingredients to describe the network simplex algorithm. The algorithm maintains a feasible spanning tree structure at each iteration, which it successively transforms it into an improved spanning tree structure until the solution becomes optimal. The algorithm first obtains an initial spanning tree structure. If an initial spanning tree structure is not easily available, then we could use the following method to construct one: for each node i with $b(i) \geq 0$, we connect node i to node 1 with an (artificial) arc of sufficiently large cost and large capacity; and for each node i with $b(i) < 0$, we connect node 1 to node i with an (artificial) arc of sufficiently large cost and capacity. These arcs define the initial tree T, all arcs in A define the set L, and $U = \emptyset$. Since these artificial arcs have large costs, subsequent iterations will drive the flow on these arcs to zero.

Given a spanning tree structure (T, L, U), we first check whether it satisfies the optimality conditions (7) and (8). If yes, we stop; otherwise, we select an arc $(k, l) \in L$ or $(k, l) \in U$ violating its optimality condition as an *entering arc* to be added to the tree T, obtain the fundamental cycle W_{kl} induced by this arc, and augment the maximum possible flow in the cycle W_{kl} without violating the flow bounds of the tree arcs. At this value of augmentation, the flow on some tree arc, say arc (p, q), reaches its lower or upper bound; we select this arc as an arc to leave the spanning tree T, adding it added to L or U depending upon its flow value. We next add arc (k, l) to T, giving us a new spanning tree structure. We repeat this process until the spanning tree structure satisfies the optimality conditions. Figure 9 specifies the essential steps of the algorithm.

```
BEGIN
    determine an initial feasible tree structure
    (T, L, U);
    let x be the flow and let π be the corresponding
    node potentials;
    WHILE (some nontree arc violates its optimality condition) DO
    BEGIN
        select an entering arc (k, l) violating the optimality conditions;
        add arc (k, l) to the spanning tree T, thus forming a unique cycle W_kl;
        augment the maximum possible flow δ in the cycle W_kl and
        identify a leaving arc (p, q) that reaches its lower or upper flow bound;
        update the flow x, the spanning tree structure (T, L, U) and the potentials π;
    END;
END
```

Minimum Cost Flow Problem, Figure 9
The network simplex algorithm

To illustrate the network simplex algorithm, we use the numerical example shown in Fig. 10a). Figure 10b) shows a feasible spanning tree solution for the problem. For this solution, $T = \{(1, 2), (1, 3), (2, 4), (2, 5), (5, 6)\}$, $L = \{(2, 3), (5, 4)\}$ and $U = \{(3, 5), (4, 6)\}$. We next compute $c^{\pi}_{35} = 1$. We introduce the arc (3, 5) into the tree, creating a cycle. Since (3, 5) is at its upper bound, the orientation of the cycle is opposite to that of (3, 5). The arcs (1, 2) and (2, 5) are forward arcs in the cycle and arcs (3, 5) and (1, 3) are backward arcs. The maximum increase in flow permitted by the arcs (3, 5), (1, 3), (1, 2), and (2, 5) without violating their upper and lower bounds is, respectively, 3, 3, 2, and 1 units. Thus, we augment 1 unit of flow along the cycle. The augmentation increases the flow on arcs (1, 2) and (2, 5) by one unit and decreases the flow on arcs (1, 3) and (3, 5) by one unit. Arc (2, 5) reaches its upper bound and we select it as the leaving arc. We update the spanning tree structure; Fig. 10c) shows the new spanning tree T and the new node potentials. The sets L and U become $L = \{(2, 3), (5, 4)\}$ and $U = \{(2, 5), (4, 6)\}$. In the next iter-



Minimum Cost Flow Problem, Figure 10
Numerical example for the network simplex algorithm

ation, we select arc (4, 6) since this arc violates the arc optimality condition. We augment one unit flow along the cycle 6 – 4 – 2 – 1 – 3 – 5 – 6 and arc (3, 5) leaves the spanning tree. Figure 10d) shows the next spanning tree and the updated node potentials. All nontree arcs satisfy the optimality conditions and the algorithm terminates with an optimal solution of the minimum cost flow problem.

The network simplex algorithm can select any nontree arc that violates its optimality condition as an entering arc. Many different rules, called pivot rules, are possible for choosing the entering arc, and these rules have different empirical and theoretical behavior. [1] describes some popular pivot rules. We call the process of moving from one spanning tree structure to another as a *pivot operation*. By choosing the right data structures for representing the tree T, it is possible to perform a pivot operation in $O(m)$ time.

To determine the number of iterations performed by the network simplex algorithm, we distinguish two cases. We refer to a pivot operation as *nondegenerate* if it augments a positive amount of flow in the cycle W_{kl} (that is, $\delta > 0$), and *degenerate* otherwise (that is, $\delta = 0$). During a degenerate pivot, the cost of the spanning tree solution decreases by $|c^{\pi}_{kl}|\delta$. When combined with the integrality of data assumption (Assumption 2) above), this result yields a pseudopolynomial bound on the number of nondegenerate iterations. However, degenerate pivots do not decrease the cost of flow and so are difficult to bound. There are methods to bound the number of degenerate pivots. Obtaining a polynomial bound on the number of iterations remained an open problem for quite some time; [6] suggested an implementation of the network simplex algorithm that runs in polynomial time. In any event, the empirical performance of the network simplex algorithm is very attractive. Empirically, it is one of the fastest known algorithms for solving the minimum cost flow problem.

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Maximum Flow Problem
- ▶ Multicommodity Flow Problems
- ▶ Network Design Problems

- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Nonoriented Multicommodity Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs, NJ
2. Barahona F, Tardos E (1989) Note of Weintraub's minimum cost circulation algorithm. SIAM J Comput 18:579–583
3. Edmonds J, Karp RM (1972) Theoretical improvements in algorithmic efficiency for network flow problems. J ACM 19:248–264
4. Goldberg AV, Tarjan RE (1988) Finding minimum-cost circulations by canceling negative cycles. Proc. 20th ACM Symposium on the Theory of Computing, pp 388–397. Full paper: J ACM (1989) 36:873–886
5. Orlin JB (1988) A faster strongly polynomial minimum cost flow algorithm. Proc. 20th ACM Symp. Theory of Computing, pp 377–387. Full paper: Oper Res (1989) 41:338–350
6. Orlin JB (1997) A polynomial time primal network simplex algorithm for minimum cost flows. Math Program 78B:109–129

MINLP: Application in Facility Location-allocation

Marianthi Ierapetritou[1], Christodoulos A. Floudas[2]

[1] Department Chemical and Biochemical Engineering, Rutgers University, Piscataway, USA

[2] Department Chemical Engineering, Princeton University, Princeton, USA

MSC2000: 90C26

Article Outline

Keywords
Location-allocation Models
Solution Approaches
Application: Development of Offshore Oil Fields
See also
References

Keywords

MINLP; Facility location-allocation

The *location-allocation problem* may be stated in the following general way: Given the location or distribution of a set of customers which could be probabilistic and their associated demands for a given product or service, determine the optimal locations for a number of service facilities and the allocation of their products or services to the costumers, so as to minimize total (expected) location and transportation costs. This problem finds a variety of applications involving the location of warehouses, distribution centers, service and production facilities and emergency service facilities. In the last section we are going to consider the development of an offshore oil field as a real-world application of the location-allocation problem. This problem involves the location of the oil platforms and the allocation of the oil wells to platforms.

It was shown in [25] that the joint location-allocation problem is *NP*-hard even with all the demand points located along a straight line. In the next section alternative location-allocation models will be presented based on different objectives and the incorporation of consumer behavior, price elasticity and system dynamics within the location-allocation decision framework.

Location-allocation Models

In developing location-allocation models different objectives alternatives are examined. One possibility is to follow the approach in [5], to minimize the number of centers required to serve the population. This objective is appropriate when the demand is exogenously fixed. A more general objective is to maximize demand by optimally locating the centers as proposed in [10]. The demand maximization requires the incorporation of *price elasticity* representing the dependence of the costumer preference to the distance from the center. The cost of establishing the centers can also be incorporated in the

model as proposed in [13]. An alternative objective towards the implementation of costumer preference towards the nearest center is the minimization of an aggregated weighted distance which is called the *median location-allocation* problem.

The simplest type of location-allocation problem is the *Weber problem*, as posed in [9], which involves locating a production center so as to minimize aggregate weighted distance from the different raw material sources. The extension of the Weber problem is the *p-median location-allocation problem*, which involves the optimal location of a set of p *uncapacitated centers* to minimize the total weighted distance between them and n demand locations. Here, each source is assumed to have infinite capacity. In continuous space, the p-median problem can be formulated as follows:

$$\begin{cases} \min & C = \sum_{i=1}^{n}\sum_{j=1}^{p} O_i \lambda_{ij} c_{ij} \\ \text{s.t.} & \sum_{j=1}^{p} \lambda_{ij} = 1, \quad i = 1, \ldots, n, \\ & \lambda_{ij} = 0, 1, \quad i = 1, \ldots, n,\ j = 1, \ldots, p, \end{cases}$$

where O_i is the quantity demanded at location i whose coordinates are (x_i, y_i); and λ_{ij} is the binary variables that is assigned the value of 1 if demand point i is located to center j and zero otherwise. The above formulation allocate the consumers to their nearest center while ensuring that only one center will serve each customer. This however, can lead to disproportionally sized facilities. In the more realistic situation where the capacities of the facilities are limited to supplies of $s_1, \ldots, s_n$ for $i = 1, \ldots, n$ facilities then the location-allocation problem takes the following form [24]:

$$\begin{cases} \min & C = \sum_{i=1}^{n}\sum_{j=1}^{p} w_{ij} c_{ij} \\ \text{s.t} & \sum_{j=1}^{p} w_{ij} = s_i, \quad i = 1, \ldots, n, \\ & \sum_{i=1}^{n} w_{ij} = d_j, \quad j = 1, \ldots, p, \\ & \lambda_{ij} = 0, 1, \quad i = 1, \ldots, n,\ j = 1, \ldots, p, \end{cases}$$

where w_{ij} is the amount shipped from facility i located at (x_i, y_i) to destination j. In the above formulations the distance (or the generalized transport cost, which is assumed to be proportional to distance) between the demand point i and the supply point j is represented by c_{ij}. The Euclidean metric:

$$c_{ij} = \sqrt{(x_i - \alpha_j)^2 + (y_i - \beta_j)^2}$$

or the rectilinear metric:

$$c_{ij} = |x_i - \alpha_j| + |y_i - \beta_j| .$$

The rectilinear metric is appropriate when the transportation is occurring along a grid of city streets (Manhattan norm) or along the aisles of a floor shop [8].

The aforementioned location-allocation models are based on the assumption that the consumers always prefer the nearest center to obtain service. In reality however, as reported in the literature from several empirical studies [11] there exist several services for which consumers choose their service facility center. The travel patterns of the consumers for example can produce a variety of allocations that differ from the nearest center rule. In order to accommodate such behavior a *spatial-interaction model* is incorporated within the uncapacitated p-median location-allocation model in the following manner:

$$\begin{cases} \min & \frac{1}{\beta}\sum_{j} Y_j \sum_{i} S_{ij} \log(S_{ij} - 1) \\ & \quad + \sum_{j}\sum_{i} Y_j S_{ij} c_{ij} \\ \text{s.t.} & \sum_{j} Y_j S_{ij} = O_i, \quad i = 1, \ldots, n, \\ & \sum_{j} Y_j = p \\ & S_{ij} \le Y_j, \quad i = 1, \ldots, n,\ j = 1, \ldots, p, \\ & Y_j = 0, 1, \quad j = 1, \ldots, p, \end{cases}$$

where the decision variables include Y_j which takes the value of one if the facility is located at J models. and zero otherwise;

$$S_{ij} = A_i O_i Y_j \exp(-\beta c_{ij})$$

that defines the interaction of facility i and consumer j.

$$A_i = \frac{1}{\sum_l Y_l \exp(-\beta c_{il})}, \quad i = 1, \ldots, m,$$

that ensures that the sum of all outflows from the origin i add up to the amount of demand at that location; β is either calibrated to match some known interaction data or is defined exogenously. The following relationship holds between the original p-median model and the spatial-interaction model as shown in [17]. The value of the optimal objective function at the solution of the p-median problem is given by:

$$\sum_i \sum_j O_i X_{ij} c_{ij},$$

where X_{ij} allocates demand to the nearest of p available centers. Turning to spatial-interaction model, as the impedance parameter β increases the term:

$$\frac{Y_j \exp(-\beta c_{ij})}{\sum_l Y_l \exp(-\beta c_{il})}$$

of the S_{ij} tends to X_{ij}, where $X_{ij} = 1$ if the travel time from i to j is smaller that the travel time from i to any other facility and zero otherwise. Therefore, the S_{ij} tends to $O_i X_{ij}$ and this model allocates the demand to the nearest facility as the original p-median problem.

All the models mentioned above consider the static location-allocation problem where all the activities take place at one instance. These formulations are sufficient if neither the level nor the location of demand alters over time. An important factor however, in any location-allocation problem is the dynamics of the system involving demand changes over time. Particularly, in the competitive environment, an optimal center location could become undesirable as new competing centers develop. Potential directions include the literature on *decision making under uncertainty*, [12]. A.J. Scott [18] proposed a general framework for the integration of the spatial and discrete temporal dimensions in the location-allocation models. He proposed a modification of the location-allocation so as to minimize an aggregate weighted transport cost over T time periods, during which time the number n_t, level O_{it} and the location (x_{it}, y_{it}) of the demand points change. If the locations were greatly different the center would be likely to relocate at some time and costs of relocation are included in the model. It was assumed that when a center relocates it incurs a fixed cost, α. Based on these ideas the formulation proposed for the uncapacitated location-allocation problem has the following form:

$$\begin{cases} \min \quad \alpha_1 + \sum_i^{n_1} O_{i1} c_{ij1} \\ \qquad\qquad + \sum_t^T \left(\alpha \lambda_t + \sum_{i=1}^{n_t} O_{it} c_{ijt} \right) \\ \text{s.t.} \quad \lambda_t = 0, 1, \quad t = 2, \ldots, T, \end{cases}$$

where the subscript t refers to different time periods, α_1 is the cost of establishing the center in the first time period. The problem as formulated above is to locate in the first period one center that takes into account future variations. Extending the aspects of this model allows the replacement of a truly dynamic model by a series of static problems as proposed in [3], thus outlining a multilayer approach, where the objective is to sequentially locate each period's facility given the previous period's facility locations in order to minimize the present period cost. This strategy is appropriate whenever the period durations are sufficiently long or under uncertainty regarding future data or decisions. An alternative approach proposed in [24] is a discounted present worth strategy which is appropriate whenever the foregoing conditions do not hold. In this case the facilities are being located one per period and the decisions are made in a rolling horizon framework.

Solution Approaches

For the uncapacitated location-allocation problem using Euclidean metric for the distances between each facility and the different demand points, R.F. Love and H. Juel [15] showed that this problem is equivalent to a concave minimization problem for which they used several heuristic procedures. For the capacitated problems assuming that the costs are proportional to l_p^q using l_p distances where $p \geq 1$ and $q \geq 1$ are integers, M. Avriel [1] developed a geometric programming approach. H.D. Sherali and C.M. Shetty [22] proposed a polar cutting plane algorithm for the case $p = q = 1$. For the case $p = q = 2$, Sherali and C.H. Tunc-

bilek [23] proposed a branch and bound algorithm (cf. ► MINLP: Branch and bound methods; ► MINLP: Branch and bound global optimization algorithm) that utilizes a specialized tight, linear programming representation to calculate strong upper bounds via a Lagrangian relaxation scheme. They exploit the special structure of the transportation constraints to derive a partitioning scheme. Additional cut-set inequalities are also incorporated to preserve partial solution.

For the uncapacitated location-allocation model using rectilinear distance metric Love and J.G. Morris [16] have developed an exact two-stage algorithm. R.E. Kuenne and R.M. Soland [14], have developed a branch and bound algorithm based on a constructive assignment of customers to sources. The capacitated problem has been addressed in [19,21] and utilize the discrete equivalence of the capacitated location-allocation problem. In particular, [8], and [26] showed that

a) the optimal values of x_i and y_i for each i must satisfy $x_i = \alpha_j$ for some j and $y_i = \beta_j$ for some j, which means that the rectilinear distance location problem always has an optimal solution with the sources located at the grid points of the vertical and horizontal lines drawn through the existing customer locations; and
b) the optimal source locations lie in the convex hull of the existing facility locations.

Based on these ideas and by denoting $k = 1, \ldots, K$ the intersection grid points that also belong to the convex hull of the existing facility locations, [21], introduced the decision binary variables z_{ik} that take the value of 1 if source i is located at point k and zero otherwise. This leads to the following discrete location-allocation problem:

$$
\begin{cases}
\min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{k=1}^{K} c_{ijk} w_{ij} z_{ik} \\
\text{s.t.} & \displaystyle\sum_{k=1}^{K} z_{ik} = 1, \quad i = 1, \ldots, n, \\
& \displaystyle\sum_{j=1}^{p} w_{ij} = s_i, \quad i = 1, \ldots, n, \\
& \displaystyle\sum_{i=1}^{n} w_{ij} = d_j, \quad j = 1, \ldots, p, \\
& w_{ij} \geq 0, \quad i = 1, \ldots, n, \quad j = 1, \ldots, p, \\
& z_{ik} = 0, 1, \quad i = 1, \ldots, n,
\end{cases}
$$

where $c_{ijk} = c_{ij}\ [|\alpha_k - \alpha_j| + |\beta_k - \beta_j|]$. The above model corresponds to a mixed integer bilinear programming problem. See [19] for a related version of this discrete-site location-allocation problem involving one-to-one assignment restriction and fixed charges. See [20] for the solution of the problem as a bilinear programming problem, since the binary variables z can be treated as positive variables because of the problem structure that preserves the binariness of z at optimality. However, in [21] it is proved that it is more useful to exploit the binary nature of z variables for the efficient solution of the above model. Before giving more details of this proposed branch and bound based approach we should mention the heuristic approach proposed in [4], which is very widely used. This so-called *alternating procedure* exploited the fundamental concepts of the location-allocation problem and simply involves allocating demand to centers and relocating centers until some convergence criterion is achieved. For the uncapacitated p-median problem, the alternating procedure involves iterating through the following equations:

$$
x_j = \frac{\sum_{i=1}^{n} \frac{O_i \lambda_{ij} x_i}{c_{ij}}}{\sum_{i=1}^{n} \frac{O_i \lambda_{ij}}{c_{ij}}},
$$

$$
y_j = \frac{\sum_{i=1}^{n} \frac{O_i \lambda_{ij} y_i}{c_{ij}}}{\sum_{i=1}^{n} \frac{O_i \lambda_{ij}}{c_{ij}}},
$$

which are derived from differentiating the objective function with respect to x_j and y_j and setting the partial derivatives to zero. The major drawback of this procedure is that it does not guarantee global optimality. This is in fact a concern because the spatial configuration of the local and the global optimum may be very different. As a rule, repeated runs using numerous starting values should be undertaken, although there is no guarantee that the repeatedly found solution would be the global optimum. Note however that the procedure is general to all different models of the location-allocation problem.

Returning to the approach proposed in [21] for the case of rectilinear capacitated location-allocation problem, the following linear reformulation of the problem

is used:

$$\begin{cases} \min & \sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{k=1}^{K} c_{ijk}X_{ijk} \\ \text{s.t.} & \sum_{k=1}^{K} X_{ijk} - w_{ij} = 0, \quad \forall(i,j), \\ & \sum_{j=1}^{p} X_{ijk} - s_i z_{ik} = 0, \quad \forall(i,k), \\ & -X_{ijk} + u_{ij} z_{ik} \geq 0, \quad \forall(i,j), \\ & \sum_{k=1}^{K} z_{ik} = 1, \quad \forall i, \\ & \sum_{j=1}^{p} w_{ij} = s_i, \quad \forall i, \\ & \sum_{i=1}^{n} w_{ij} = d_j, \quad \forall j, \\ & w_{ij} \geq 0, \quad \forall(i,j), \\ & z_{ik} = 0,1, \quad \forall i, \\ & X_{ijk} \geq 0, \quad \forall(i,j,k), \end{cases}$$

where $u_{ij} = \min\{s_i, d_j\}$. The above model corresponds to a mixed integer linear programming problem for which a special branch and bound algorithm is applied based on the derivation of tight lower bounds via a suitable Lagrangian dual formulation.

Briefly, for the location-allocation problems that have embedded spatial-interaction equations dual-based exact methods, [17], and heuristic approaches, [2], have been developed.

Application: Development of Offshore Oil Fields

In this section a real world application of the location-allocation problem is presented considering the minimum-cost development of offshore oil fields, [6]. The facilities to be located are the platforms and the demands to be allocated are the oil wells. For the initial information about an oil field, locations are decided upon the production wells which are specified by two map coordinates and a depth coordinate. The drilling is performed directionally from fixed platforms. The cost of drilling depends on the length and angle of the well from the platform. The platform cost depends on the water depth and on the number of wells to be drilled from the platform. Consequently for a large number of wells (25 to 300) an optimization problem that arises is to find the number, size and location of the platforms and the allocation of wells to platforms so as to minimize the sum of platform and drilling costs.

In order to formulate this problem the following indices, parameters and variables are introduced. Let m denote the number of wells and i the index of well, n the number of platforms and j the index for platform, z_{ij} are then the binary variables that represent the allocation of the well i to platform j if it takes the value of 1, otherwise it becomes 0, S_j the capacity of the platform j representing the number of wells drilled from this platform, (a_i, b_i) denote the location coordinates of well i and (x_j, y_j) the location of platform j, $d_{ij} = \sqrt{[(x_j - a_i)^2 + (y_j - b_i)^2]}$ is the horizontal Euclidean distance between well i and platform j, $g(d_{ij})$ denotes the drilling cost function that depends on distance d_{ij}, $P(S_j, x_j, y_j)$ is the platform cost which is a function of platform size S_j and its location. Based on this notation the location-allocation problem can be formulated as follows:

$$\begin{cases} \min & \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij} g(d_{ij}) + \sum_{j=1}^{n} P(S_j, x_j, y_j) \\ \text{s.t.} & \sum_{j=1}^{n} z_{ij} = 1, \quad \forall(i), \\ & \sum_{i=1}^{m} z_{ij} = S_j, \quad \forall j, \\ & z_{ij} = 0,1, \quad \forall(i,j), \end{cases}$$

where the first set of constraints guarantee that each well is assigned to exactly one platform and the second set guarantee that exactly S_j wells are assigned to each platform. Note that n is fixed in the problem and is usually small in the size of 3 to 5. The nature of the problem depends upon the form of the cost of the drilling function and the platform cost function. The approach taken in [6] is the alternating location-allocation method presented in the previous section. For the specific problem the approach involves the following steps:

a) given fixed platform locations find a minimum cost allocation of wells to platforms;
b) given fixed allocation of wells to platforms find the minimum total cost location for each platform.

The procedure alternates between steps a) and b) until convergence is achieved. The convergence criterion is the following: From the solution of step a) a set of

n subproblems are generated for each one of the platforms, the solution of these problems result in the relocation of the platforms. The iterations continue until no changes are possible. As mentioned above, the solution obtained from this algorithmic procedure is locally optimum in the sense that for a given assignment of wells to platforms the solution cannot be improved by changing locations and for given locations, the solution cannot be improved by altering the assignment of wells to platforms. The mathematical formulation of problem a), the *allocation subproblem* is the following:

$$\begin{cases} \min & \displaystyle\sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij} g(d_{ij}) + \sum_{j=1}^{n} P(S_j) \\ \text{s.t.} & \displaystyle\sum_{j=1}^{n} z_{ij} = 1, \quad \forall (i), \\ & \displaystyle\sum_{i=1}^{m} z_{ij} = S_j, \quad \forall j, \\ & z_{ij} = 0, 1, \quad \forall (i, j), \end{cases}$$

note that the platform cost now depends only on S_j since the location of the platforms are known. The solution procedure for this problem depends on the form of the platform cost $P(S_j)$. Five different forms are discussed in [6]:

1) *Single fixed cost with no capacity constraints*: $P(S_j) = a_j$ In this case the total cost for platforms is fixed and the optimal allocation corresponds to the assignment of the wells to the closest platform.
2) *Single fixed cost with capacity constraints*: $P(S_j) = a_j$ and capacity constraints are introduced as inequalities $\sum_{i=1}^{m} z_{ij} \leq S_j$, $\forall j$. In this case the problem corresponds to a linear programming model.
3) *Linear platform cost*: $P(S_j) = a_j + b_j S_j$ By considering the following transformation $c_{ij}' = c_{ij} + b_j$ the problem takes the form of case 1).
4) *Piecewise linear function*. In this case the problem has the structure of 'transshipment problem' which can be solved network flow techniques.
5) *Step function*: $P(S_j) = \sum_{k=1}^{K_j} r_j^k z_{ij}^k$, where K_j are the number of different size platforms available and r_j^k is the cost of kth size of platform j. The problem in this case is a mixed integer linear programming problem.

The mathematical formulation for problem b), the location problem, is the following. Assuming that A_j is the set of indices for the wells assigned to platform j, then $z_{ij} = 1$, for $i \in A_j$, $z_{ij} = 0$ otherwise and the problem for platform j takes the form:

$$\min \sum_{i=1}^{m} \sum_{i \in A_j} g(d_{ij}) + P(x_j, y_j).$$

Note that the platform cost is a function of platform location only since the size is assumed known. Since the drilling cost function is convex, if the platform cost is also convex then the problem corresponds to the minimization of a convex function that can be achieved through a local minimization algorithm. Of course if the platform cost is nonconvex then global optimality cannot be guaranteed and global optimization techniques should be considered, [7].

Finally, M.D. Devine and W.G. Lesso, [6], applied the aforementioned procedure to two test problems one involving 60 wells and 7 platforms and a second one involving 102 wells and 3 platforms. In both cases they reported large economic savings in the field development.

See also

- ▶ Chemical Process Planning
- ▶ Combinatorial Optimization Algorithms in Resource Allocation Problems
- ▶ Competitive Facility Location
- ▶ Extended Cutting Plane Algorithm
- ▶ Facility Location with Externalities
- ▶ Facility Location Problems with Spatial Interaction
- ▶ Facility Location with Staircase Costs
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ Global Optimization in Weber's Problem with Attraction and Repulsion
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis

- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming
- ▶ Multifacility and Restricted Location Problems
- ▶ Network Location: Covering Problems
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Production-distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Stochastic Transportation and Location Problems
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Avriel M (1980) A geometric programming approach to the solution of locational problems. J Reg Sci 20:239–246
2. Beaumont JR (1980) Spatial interaction models and the location-allocation problem. J Reg Sci 20:37–50
3. Cavalier TM, Sherali HD (1985) Sequential location-allocation problems on chains and trees with probabilistic link demands. Math Program 32:249–277
4. Cooper L (1964) Heuristic methods for location-allocation problems. SIAM Rev 6:37–53
5. Cristaller W (1966) Central places in southern Germany. Prentice-Hall, Englewood Cliffs, NJ
6. Devine MD, Lesso WG (1972) Models for the minimum cost development of offshore oil fields. Managem Sci 18:378–387
7. Floudas CA (1997) Deterministic global optimization in design, control, and computational chemistry. In: IMA Proc.: Large Scale Optimization with Applications. Part II: Optimal Design and Control 93:129–184
8. Francis RL, White JA (1974) Facility layout and location: An analytical approach. Prentice-Hall, Englewood Cliffs, NJ
9. Friedrich CJ (1929) Alfred Weber's theory of the location of industries. Univ. Chicago Press, Chicago
10. Getis A, Getis J (1966) Cristaller's central place theory. J Geography 65:200–226
11. Hubbard MJ (1978) A review of selected factors conditioning consumer travel behavior. J Consumer Res 5:1–21
12. Ierapetritou MG, Acevedo J, Pistikopoulos EN (1996) An optimization approach for process engineering problems under uncertainty. Comput Chem Eng 20:703–709
13. Koshaka RE (1983) A central-place model as a two-level location-allocation system. Environm Plan 15:5–14
14. Kuenne RE, Soland RM (1972) Exact and approximate solutions to the multisource Weber problem. Math Program 3:193–209
15. Love RF, Juel H (1982) Properties and solution mathods for large location-allocation problems. J Oper Res Soc 33:443–452
16. Love RF, Morris JG (1975) A computational procedure for the exact solution of location-allocation problems with rectangular distances. Naval Res Logist Quart 22:441–453
17. O'Kelly M (1987) Spatial interaction based location-allocation models. In: Ghosh A, Rushton G (eds) Spatial Analysis and Location Allocation Models. v. Nostrand, Princeton, NJ, pp 302–326
18. Scott AJ (1971) Dynamic location-allocation systems: Some basic planning strategies. Environm Plan 3:73–82
19. Sherali AD, Adams WP (1984) A decomposition algorithm for a discrete location-allocation problem. Oper Res 32:878–900
20. Sherali AD, Alameddine AR (1992) A new reformulation-linearization technique for the bilinear programming problems. J Global Optim 2:379–410
21. Sherali AD, Ramachandran S, Kim S (1994) A localization and reformulation discrete programming approach for the rectilinear discrete location-allocation problem. Discrete Appl Math 49:357–378
22. Sherali AD, Shetty CM (1977) The rectilinear distance location-allocation problem. AIIE Trans 9:136–143
23. Sherali AD, Tuncbilek CH (1992) A squared-Euclidean distance location-allocation problem. Naval Res Logist 39:447–469
24. Sherali HD (1991) Capacitated, balanced, sequential location-allocation problems on chain and trees. Math Program 49:381–396
25. Sherali HD, Nordai FL (1988) NP-hard, capacitated, balanced p-median problems on a chain graph with a continuum of link demands. Math Oper Res 13:32–49
26. Wendell RE, Hurter AP (1973) Location theory, dominance and convexity. Oper Res 21:314–320

MINLP: Applications in Blending and Pooling Problems

Viswanathan Visweswaran
SCA Technologies LLC, Pittsburgh, USA

MSC2000: 90C90, 90C30

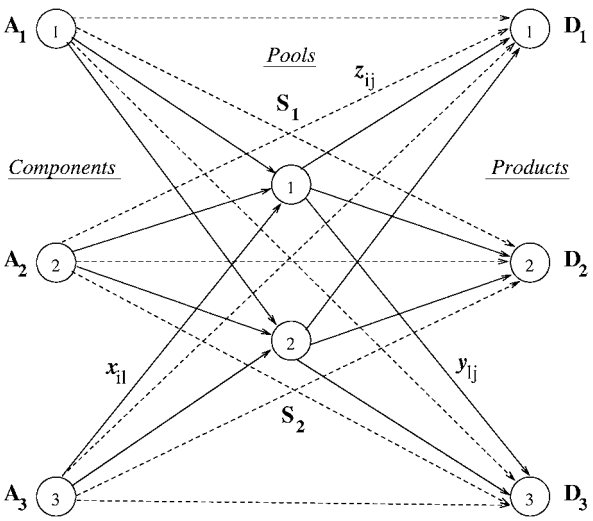
Article Outline

Keywords
Characteristics of Pooling and Blending Problems
Multiple Solutions
Nonlinear Blending
Single versus Multiperiod Models
Logical Constraints and MINLP Formulations
Complexity of Models
Solution Methods
Local Optimization Approaches
Global Optimization Approaches
Applications
See also
References

Keywords

Pooling; Blending; Multiperiod optimization

Pooling and blending is inherent in many manufacturing plants with limited tankage available to store the intermediate streams produced by various processes. Also, chemical products often need to be transported as a mixture, either in a pipeline, a tank car or a tanker. In each case, blended or pooled streams are then used in further downstream processing. In modeling these processes, it is necessary to model not only product flows but the properties of intermediate streams as well. The presence of these pools can introduce nonlinearities and nonconvexities in the model of the process, resulting in difficult problems with multiple local optima.



MINLP: Applications in Blending and Pooling Problems, Figure 1
General pooling and blending problem

Given a set of components i, a set of products j, a set of pools k and a set of qualities l, let x_{il} be the amount of component i allocated to pool l, y_{lj} be the amount going from pool l to product j, z_{ij} be the amount of component i going directly to product j and p_{lk} be the level of quality k in pool l. Furthermore, let A_i, D_j and S_l be upper bounds for component availabilities, product demands and pool sizes respectively, let C_{ik} be the level of quality k in component i, P_{jk} be upper bounds on product qualities, c_i be the unit price of component i and d_j be the unit price of product j. The general pooling and blending model can then be written as [1]:

$$
\begin{cases}
\max & -\sum_{i,l} c_i x_{il} + \sum_{l,j} d_j y_{lj} + \sum_{i,j} (d_j - c_i) z_{ij} \\
\text{s.t.} & \sum_{l} x_{il} + \sum_{j} z_{ij} \le A_i \\
& \sum_{l} y_{lj} + \sum_{i} z_{ij} \le D_j \\
& \sum_{i} x_{il} - \sum_{j} y_{lj} = 0 \\
& \sum_{i} x_{il} \le S_l \\
& -\sum_{i} C_{ik} x_{il} + p_{lk} \sum_{j} y_{lj} = 0 \\
& \sum_{l} (p_{lk} - P_{jk}) y_{lj} \\
& \quad + \sum_{i} (C_{ij} - P_{jk}) z_{ij} \le 0 \\
& x_{il}, y_{lj}, z_{ij}, p_{lk} \ge 0.
\end{cases}
$$

The first two sets of constraints ensure that the amount of components used and products made do not exceed the respective availabilities or demands. The third and fourth set of constraints are material balance constraints around each pool, which ensure that there is no accumulation or overflow of material in the pools. The fifth set of constraints relates the quality of each pool to the quality of the components going into the pool (in this case, the qualities are assumed to blend linearly, that is, the pool quality is an average of the qualities of the components). Finally, the sixth set of equations ensures that any upper bound specifications on product qualities are met. These last two sets of equations are

bilinear, and can cause significant problems in solving these models.

The general blending problem has a similar formulation as above, except that the pools need not be present; the components can be blended directly to make various products. It should also be noted that there are various other formulations possible, involving multiple time periods, tanks and inventories for components and products, and costs for pooling. Moreover, not all the components need go through all pools. One example of a simplified pooling model, due to C.A. Haverly [8,9], is given in Fig. 2, where three components with varying sulfur contents are to be blended to form two products. There is a maximum sulfur restriction on each product. The components have values of 6, 13 and 10, respectively, while the products have values of 9 and 15, respectively. The mathematical model for the problem consists of writing mass and sulfur balances for the various streams, and can be formulated as

$$
\begin{cases}
\max & 9\cdot(y_{11}+z_{31})+15\cdot(y_{12}+z_{32}) \\
 & -6x_{11}-13x_{21}-10\cdot(z_{31}+z_{32}) \\
\text{s.t.} & x_{11}+x_{21}-y_{11}-y_{12}=0 \\
 & p\cdot y_{11}+2z_{31}-2.5(y_{11}+z_{31})\le 0 \\
 & p\cdot y_{12}+2z_{32}-1.5(y_{12}+z_{32})\le 0 \\
 & p\cdot(y_{11}+y_{12})-3x_{11}-x_{21}=0 \\
 & y_{11}+z_{31}\le 100 \\
 & y_{12}+z_{32}\le 200.
\end{cases}
$$

The variable p represents the sulfur content of the pool (and of y_{11} and y_{12}) and is determined as an average of the sulfur contents of x_{11} and x_{21}.

Characteristics of Pooling and Blending Problems

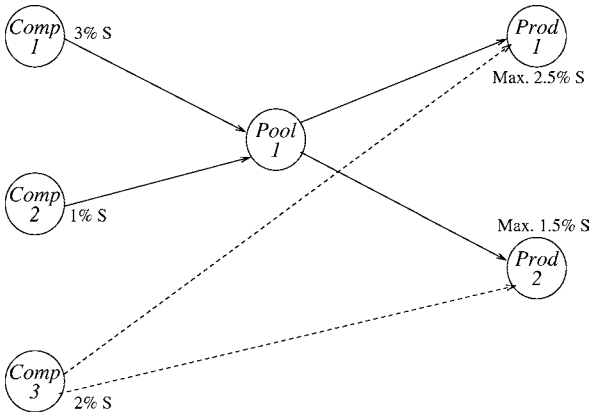
Multiple Solutions

The presence of nonconvex constraints needed to define pool and product qualities often results in multiple local solutions in these models. For example, consider the optimal solution of the Haverly pooling problem as a function of the pool quality p, as shown in Fig. 3.

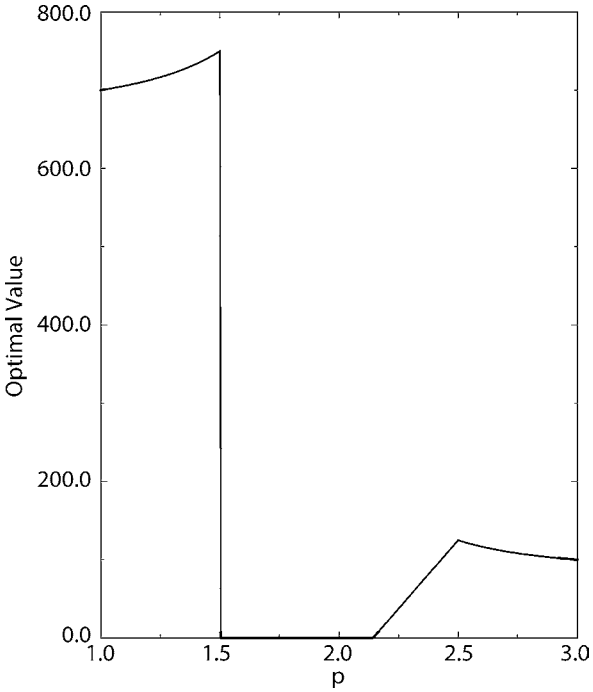
It can be seen that the problem has three solutions:

1) A local maximum of 125 at $p = 2.5$ with $x_{11} = 75$, $x_{21} = 25$, $y_{11} = 100$ and all other variables zero;
2) a saddle point region with $1 < p < 2$, all flows zero and profit of zero; and
3) a global maximum of 750 at $p = 1.5$ with $x_{11} = 50$, $x_{21} = 150$, $y_{12} = 200$ and all other variables zero.

It is not uncommon for a large pooling problem to have many dozen local optima, with the objective function varying by small amounts but with all the flow and quality variables taking on vastly different values.



MINLP: Applications in Blending and Pooling Problems, Figure 2
Haverly pooling problem



MINLP: Applications in Blending and Pooling Problems, Figure 3
Optimal solution to Haverly pooling problem

Nonlinear Blending

For the sake of simplicity, it is often assumed in formulating these models that the qualities to be tracked blend linearly by volume or weight of each component. In practice, however, this is rarely the case. For example, one of the properties commonly tracked in refinery blends is the *Reid vapor pressure* (RVP), which measures the volatility of a blend. The most commonly used blending rule for RVP is the *Chevron method*:

$$\left(\sum_i x_i\right) R^{1.25} = \sum_i x_i r_i^{1.25},$$

where r_i is the RVP of component i, x_i is its volume, and R is the RVP of the blend. Including such a nonlinear equation in the model can cause difficulty in its solution. Fortunately, this can be avoided by introducing a *blending index*, defined as

$$\overline{r_i} = r_i^{1.25}, \quad \overline{R} = R^{1.25}.$$

Then, all specifications on the blend RVP can be converted using the same index. For example, if there is a lower bound R^L on the blend RVP, then using the blending index results in the constraints as:

$$\left(\sum_i x_i\right) \overline{R} = \sum_i x_i \overline{r_i}, \qquad \overline{R} \geq (R^L)^{1.25}.$$

In some cases, the properties (such as octane number or pour point) can require complex blending rules which cannot be simplified using the blending index, and the full nonlinear blending equation must be included in the model as is.

Single versus Multiperiod Models

Since components are pooled or blended in the plants on a regular basis, it is often advantageous to model these processes using multiple periods. With *multiperiod models*, it is possible to accumulate material in the pools or blend tanks, thereby facilitating the allocation of stocks ahead of time in anticipation of a future lifting of a valuable product. This requires the model to incorporate inventories (carry-over stock) in each tank or pool, resulting in more complex models. It is important to note that each period does not need to be of the same duration. Often, the results of the multiperiod models will only be implemented for the first period, with results for future periods being used for planning purposes. Therefore, initial periods are typically of shorter duration (say a day each) while later periods might be as long as a month. This way, the same multiperiod model can be used as an operating tool for the present and a planning tool for the future.

Another important consideration in multiperiod models is the disposition of stocks at the end of the final period. If the final inventories/stocks are included simply as variables, the optimal solution will almost always set them to zero. In practice, however, this is unrealistic since it is not desired to run down stocks. This can be dealt with in several ways:

a) set the final inventory levels to reasonable values (say the same as inventory levels at the beginning of the first period);
b) assign a value to final inventory; this way the model can decide if it is worthwhile to produce stock to sell at the end of the final period.

Logical Constraints and MINLP Formulations

It is often necessary to impose additional logical constraints that dictate how various components are to be blended in relation to each other. Modeling such constraints often requires the addition of integer variables, as discussed below.

a) If a component is to be used in a particular blend, then it must be present in at least a certain amount in the blend. This arises from the fact that it is usually not practical to blend in infinitesimally small quantities.
   If x represents the volume of such a component, then introducing a new binary variable δ (i. e. δ is either 0 or 1) and the constraints

$$x - M\delta \leq 0,$$
$$x - m\delta \geq 0$$

   are sufficient to ensure this condition is satisfied. Here, M is a sufficiently large number, while m represents the threshold value below which a component should not be blended in.
b) Each product can have at most k components in its blend. This is typically imposed by limitations on how many streams can be physically blended in a reasonable amount of time. Again, introducing the

new variables and constraints as below:

$$x_1 - m\delta_1 \geq 0,$$
$$\cdots$$
$$x_n - m\delta_n \geq 0,$$
$$\delta_1 + \cdots + \delta_n \leq k,$$
$$\delta_1, \ldots, \delta_n \in \{0-1\}^n,$$

ensures this condition is met.

c) If component A is to be present in the blend, then component B must also be present:

$$x_A - m\delta_A \geq 0,$$
$$x_B - m\delta_B \geq 0,$$
$$\delta_B \geq \delta_A.$$

Each of these logical constraints results in a mixed integer nonlinear programming (MINLP) model (cf. also ► Mixed integer nonlinear programming). To date (2000), such models have not been used extensively in the practical solution of these problems in industry.

Complexity of Models

With the various options of single versus multiperiod and linear versus nonlinear blending, the models for pooling and blending can vary significantly in complexity. This is shown pictorially in Fig. 4.

Solution Methods

Pooling problems can be solved using a variety of solution algorithms. These can be broadly classified as local and global solution methods.

Local Optimization Approaches

Traditionally, pooling and blending problems have been solved using various recursion and successive linear programming (SLP) techniques. The first published approach for solving the pooling problem was due to Haverly [8], who proposed the following recursion approach for solving the problem given in Fig. 2:

1	Start with a guess for the pool quality p.
2	Solve the remaining linear problem for all other variables.
3	Calculate a new value for p from the solution in 2).

Unfortunately, this rather simple recursion will converge to a suboptimal solution regardless of the starting value for p. This can be partially addressed by using a 'distributed recursion' approach, where an additional recursion coefficient f and two additional 'correction vectors' are introduced, modifying the inequalities in the model as follows:

$$p \cdot y_{11} + 2z_{31} - 2.5(y_{11} + z_{31}) + f(\text{over} - \text{under}) \leq 0,$$
$$p \cdot y_{12} + 2z_{32} - 1.5(y_{12} + z_{32}) + (1-f)(\text{over} - \text{under}) \leq 0.$$

This formulation serves to distribute the error made in estimating the pool quality to the two pool destinations. Recursing on both p and f has a better likelihood of identifying the optimal solution.

SLP algorithms solve nonlinear models through a sequence of linear programs (LPs), each of which is a linearized version of the model around some base point. These methods consist of replacing nonlinear constraints of the form

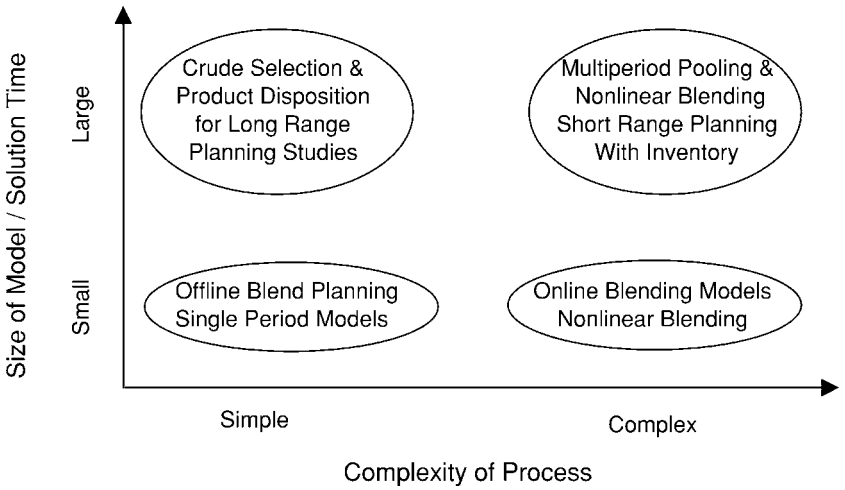
$$g(x) \leq 0, \qquad h(x) = 0,$$

with the linearizations

$$g(\overline{x}^k) + \nabla g(\overline{x}^k) \cdot (x - \overline{x}^k) \leq 0,$$
$$h(\overline{x}^k) + \nabla h(\overline{x}^k) \cdot (x - \overline{x}^k) = 0$$

around a base point $\overline{x}^k$ at the kth iteration. The linearized problems can be solved using standard LP methods. The solution to the problem is used to provide a value for $\overline{x}^{k+1}$. As long as there is an improvement in the objective function value as well as the feasibility of the original constraints, these methods can be shown to converge to a local optimum. They work well for largely linear problems and have therefore found widespread use in the refining industry for solving pooling, blending and general refinery planning problems [4,11]. However, when there are nonlinear blending constraints, the linearization in the SLP is often a bad approximation of the original problem, leading to poor convergence rates and large solution times.

Pooling and blending problems can also be solved using other nonlinear programming (NLP) methods such as generalized reduced gradient, successive quadratic programming or penalty function methods.



MINLP: Applications in Blending and Pooling Problems, Figure 4
Types of pooling problems

In general, these methods have not found large acceptance in solving these problems, mainly due to difficulties with convergence and stability.

Global Optimization Approaches

The recursive, SLP and conventional NLP techniques all suffer from the drawback that the solution found is highly dependent on the starting point, and in general cannot guarantee convergence to the global solution. In the last dozen years, numerous approaches have been proposed for the solution of quadratically constrained optimization problems (such as the pooling/blending problem). Surveys of these algorithms can be found in [10,12]. These approaches can generally be classified as either decomposition-based or branch and bound algorithms.

One of the common approaches to dealing with the nonconvexities in the pooling problem is to reduce the bilinear terms to linear terms over a convex envelope [2]. Noting that for any bilinear term $p \cdot y$,

$$
\begin{aligned}
&(p - p^L) \cdot (y - y^L) \geq 0,\\
&(p - p^U) \cdot (y - y^U) \geq 0,\\
&(p - p^L) \cdot (y - y^U) \leq 0,\\
&(p - p^U) \cdot (y - y^L) \leq 0,
\end{aligned}
$$

where $[p^L, p^U]$ and $[y^L, y^U]$ define the ranges for the variables p and y. This allows the term $p \cdot y$ to be replaced by a set of linear inequalities in the model, resulting in a linearized problem which provides an upper bound on the global solution to the original problem. After solving this problem, the rectangle defined by the bounds on p and y can be subdivided into smaller rectangles, and a new linearized problem can be solved over each of these subrectangles. By continuously subdividing these rectangles, the upper bound can be made to asymptotically approach the global solution. See [7] for the solution of several pooling problems using this approach.

Note that the pooling problem is a partially linear problem. That is, it can be formulated as

$$
\begin{cases}
\min\limits_{x,p} & c^\top x\\
\text{s.t.} & A(p)x \leq b,
\end{cases}
\tag{1}
$$

where p represents the pool quality and x represents all component flow rates. For such problems, decomposition approaches provide a natural solution mechanism. For a fixed value of p, this problem is linear, and provides an upper bound on the global solution. The solution to this linear problem (called the 'primal' problem) can be used to generate a *Lagrange function* of the form

$$
L(x, p) = c^\top x + \lambda \cdot (A(p)x - b)
$$

where λ represents the multipliers or marginal values for the constraints from the primal problem. Then, the

'dual' problem

$$\begin{cases} \min\limits_{x,p,\mu} & \mu \\ \text{s.t.} & \mu \geq L(x,p) \end{cases} \tag{2}$$

provides an upper bound on the global solution. Problem (2) contains bilinear terms of the form $A(p)x$, which can be underestimated in a variety of ways. C.A. Floudas and V. Visweswaran [5,6] have developed the GOP algorithm based on this approach. By alternating between the primal problem and a series of relaxed dual problems (developed by successively partitioning the feasible region), the GOP algorithm guarantees convergence to the global solution. In [13,14], they show that it is possible to develop properties that reduce the number of relaxed dual problems that need to be solved, thus speeding up the overall algorithm. They also report the solution of numerous pooling and blending problems using this approach.

Instead of fixing p for the primal problem, it is possible to solve (1) directly using local optimization techniques. For example, nonsmooth optimization techniques can be effective in finding local solutions to these problems [1]. The dual problem can also be solved this way, with the region for p being refined by partitioning. See [1] for the solution of several pooling problems using this approach.

It is important to note that these global optimization approaches (and others) for solving the pooling problem can be computationally intensive. Invariably, a large number of subproblems need to be solved before convergence to a global solution can be guaranteed. Because the subproblems are usually of the same structure, varying only slightly in the data for the problems, they can be solved in parallel. See [3] for an implementation of a distributed parallel version of the GOP algorithm and a successful application to solve pooling problems of medium size.

Applications

The most common application of pooling and blending models is in the *refining* and *petrochemical industries*. Crude oil from various sources is often brought into the refinery and stored in common tanks before being processed downstream. Similarly, intermediate streams from various refinery processes (alkylation, reforming, cracking) are usually sent to common pools from which finished products such as gasoline and diesel oil are made. In both cases, it is important to know various qualities of the stream coming out of the pool (such as chemical compositions like sulfur or physical properties such as vapor pressure).

In addition to refinery processes, blending is a feature of various other manufacturing processes. These include

- *agriculture*, where blending livestock feeds or fertilizers at minimum cost is very important;
- mining, where different ores are often mixed to achieve a desired quality;
- various aspects of food manufacturing; and
- pulp and paper, involving blending of raw materials used to produce paper.

See also

- ▶ Chemical Process Planning
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming

References

1. Ben-Tal A, Eiger G, Gershovitz V (1994) Global minimization by reducing the duality gap. Math Program 63:193
2. Al-Khayyal FA, Falk JE (1983) Jointly constrained biconvex programming. Math Oper Res 8(2):273
3. Androulakis IP, Isweswaran VV, Floudas CA (1995) Distributed decomposition-based approaches in global optimization. In: Floudas CA, Pardalos PM (eds) Proc. State

of the Art in Global Optimization: Computational Methods and Applications. Kluwer, Dordrecht, pp 285–301
4. Baker TE, Lasdon LS (1994) Successive linear programming at Exxon. Managem Sci 31(3):264
5. Floudas CA, Visweswaran V (1990) A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: I. Theory. Comput Chem Eng 14:1397
6. Floudas CA, Visweswaran V (1993) A primal-relaxed dual global optimization approach. J Optim Th Appl 78(2):187
7. Foulds LR, Haugland D, Jörnsten K (1990) A bilinear approach to the pooling problem. Chr. Michelsen Inst. Working Paper 90-3
8. Haverly CA (1978) Studies of the behaviour of recursion for the pooling problem. ACM SIGMAP Bull 25:19
9. Haverly CA (1979) Behaviour of recursion model–more studies. ACM SIGMAP Bull 26:22
10. Horst R, Tuy H (1993) Global optimization: Deterministic approaches. second Springer, Berlin
11. Lasdon LS, Waren AD, Sarkar S, Palacios-Gomez F (1979) Solving the pooling problem using generalized reduced gradient and successive linear programming algorithms. ACM SIGMAP Bull 27:9
12. Pardalos PM, Rosen JB (1987) Constrained global optimization: Algorithms and applications. Lecture Notes Computer Sci, vol 268. Springer, Berlin
13. Visweswaran V, Floudas CA (1996) Computational results for an efficient implementation of the GOP algorithm and its variants. In: Grossmann IE (ed) Global Optimization in Engineering Design. Nonconvex Optim Appl. Kluwer, Dordrecht, pp 111–154
14. Visweswaran V, Floudas CA (1996) New formulations and branching strategies for the GOP algorithm. In: Grossmann IE (ed) Global Optimization in Engineering Design. Kluwer, Dordrecht, pp 75–110

MINLP: Applications in the Interaction of Design and Control

Carl A. Schweiger, Christodoulos A. Floudas
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C11, 49M37

Article Outline

Keywords
Previous Work
Process Synthesis
Steady-State Modeling Approach
Dynamic Modeling Approach
MIOCP Solution Algorithm
  Primal Problem
Example: Reactor-Separator-Recycle System
See also
References

Keywords

Mixed integer nonlinear optimization; Parametric optimal control; Interaction of design and control

In the development of a process, the steady state design aspects and dynamic operability issues are usually handled sequentially. First, the design engineers develop and synthesize the structure of the flowsheet and determine the operating parameters and steady-state operating conditions. Then, the control engineer takes the fixed design and develops a control system to maintain the system at the desired specifications. During the first step, the dynamic operation of the process is generally not considered, and in the second step, changes to the flowsheet and operating conditions generally can not be made.

Process design seeks to determine the arrangement of processing units that will convert the given raw materials into the desired products. The idea is to develop a process flowsheet from the large number of possible design alternatives. Numerous process design methods and techniques exist for determining the best process flowsheet and operating conditions. This best design is determined by optimizing some economic criteria and the quality of the design is based on its economic value. Hence, the process is designed to operate at steady state and issues relating to the *process dynamics*, *operability*, and *controllability* are usually not considered.

Once the process has been designed, the plans are handed over to the *process control* engineer whose task is to ensure the stable dynamic performance of the process. The control engineer is concerned with developing a control system which maintains the operation of the process at the desired steady state in the presence ever-changing external influences. Issues such as disturbances, uncertainty, and changes in production rates must be addressed so as to maintain product quality and safe operation. By addressing the design and control sequentially, the inherent connection between the two is neglected. For instance, the steady-state design of

a process may appear to produce great economic profits. However, unfavorable dynamic operation may lead to a product which does not meet the required specifications. This may result in an economic loss due to disposal or reworking costs. Thus, a process design with good controllability aspects may have better economic value that an economically optimal steady state design when the dynamic operation is considered. This trade-off between the steady state design and the dynamic controllability motivates the treatment of the issues simultaneously.

There are additional incentives for employing a simultaneous approach. Due to economic and environmental reasons, the recent trend in process design has been towards more highly integrated process in terms of both material and energy flows. Processes are also required to operate under much tighter operating conditions due to environmental and safety issues. Both of these lead to designs with increased dynamic interactions and processes which are generally more difficult to control. Thus, the dynamic operation of the process must be considered at the early stages of the design.

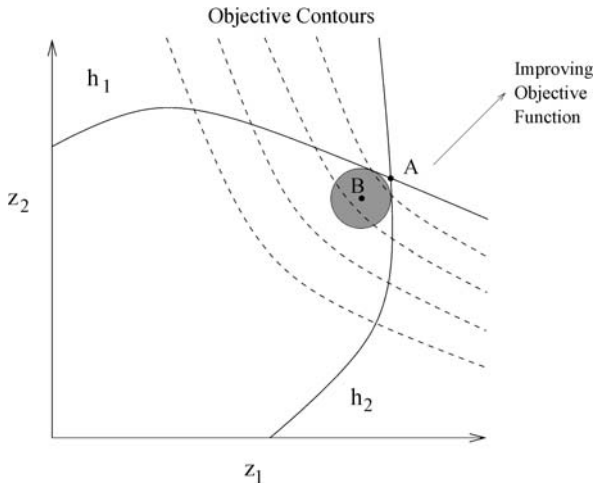
A systematic method for analyzing the *interaction of design and control* requires quantitative *controllability measures* of the process. Such measures have been derived to quantify certain qualitative concepts about the controllability of the process such as inversion, interaction effects, and directionality problems. A common measure for controllability is the integral squared error (ISE) between outputs and their desired levels. Although it is easy to measure, it is not of direct interest in practice. Other performance criteria such as maximum deviation of output variables, maximum magnitude of control variables, or time to return to steady state can also be used.

Most of the work in the development of controllability measures has focused on linear dynamic models. The control objective is the robust performance of the process without any restrictions on the controller structure [15]. One such measure is the structured singular value, σ, which indicates the performance in the presence of uncertainty. The condition number, γ, has been developed as an indicator of closed-loop sensitivity to model error while the disturbance conditions number, γ_d, indicates the sensitivity of the process to disturbances. The relative gain array (RGA), Λ, is used as an indicator of the relationship between control error and set point changes while the closed-loop disturbance gain (CLDG) is used to measure the relation between control error and disturbances. These measures have been used extensively in applications for controllability assessment; however, they can be misleading. While these indicators give ideas as to the closed loop performance of the process, their impact on the economics of the process is not clear.

Previous Work

In comparison to the amount of research on the controllability measures, relatively little work has been placed on methods for systematically determining the *trade-offs* between steady-state economics and dynamic controllability. Although economics continues to be the driving force in the design of a process, there is no straightforward method for evaluating the economics of the dynamic operation of the process. Several methods have been proposed to address these issues. M. Morari and J.D. Perkins [14] discuss the concept of controllability and emphasize that the design of a control system for a process is part of the overall design of the process. Noting that a great amount of effort has been placed on the assessment of controllability, particularly for linear dynamic models, they indicate that very little has been published on algorithmic approaches for determination of process designs where economics and controllability are traded off systematically.

In order to deal with the controllability issues on a economic level, a *back-off* method was presented in [18] to determine the economic impact of disturbances on the system. The basic idea is to determine the optimal steady-state operating point such that the feasible operation is maintained with respect to all constraints in the presence of uncertainties and disturbances. This operating point is compared to the optimal steady-state operating point determined in the absence of disturbances. The economic penalty incurred by backing away from the disturbances-free operating point to the feasible operating point can be determined and thus the cost of the disturbance can be evaluated. This concept is illustrated in Fig. 1. Point A indicates the nominal steady-state design, and point B is the back-off point which corresponds to the design which will not violate the constraints h_1 and h_2 in the presence of uncertainties and disturbances.



MINLP: Applications in the Interaction of Design and Control, Figure 1
Illustration of the back-off approach

The method is further developed in [17], where the control structure selection problem is analyzed. Perfect control assumptions are used along with a linearized model to formulate a mixed integer linear program (MILP) where the integer variables indicate the pairings between the manipulated and controlled variables. The back-off approach incorporated the dynamic operation of the process into the design, but it only ensures the feasible operation of the process and does not directly address controllability aspects.

An approach for determining process designs which are both steady-state and operationally optimal was presented in [2]. The controllability of potential designs is evaluated along with their economic performance by incorporating a model predictive control algorithm into the process design optimization algorithm. This coordinated approach uses an objective function which is a weighted sum of economic and controllability measures.

A multi-objective approach was proposed in [9,10] to simultaneously consider both controllability and economic aspects of the design. This approach incorporates both design and control aspects into a process synthesis framework where the trade-offs between various open-loop controllability measures and the economics of the process can be observed. The problem is formulated as a *mixed integer nonlinear program* (*MINLP*), where integer variables are utilized for structural alternatives in the process flowsheet. Through the application of multi-objective techniques, a process design which is both economic and controllable is determined.

A screening approach was proposed in [4], where the variability in the product quality is used to compare different steady-state process designs. The dynamic controllability is measured economically by calculating the amount of material produced that is off-specification and on-specification. The on-specification material leads to profits while the off-spec material results in costs for reworking or disposal.

A back-off technique was also developed in [1] for the design of steady-state and open-loop dynamic processes. Both uncertainties and disturbances are considered for determining the amount of back-off. In order to address the fact that back-off approaches address the feasible operation and do not address controllability aspects, [5] introduces a recovery factor which is defined as the ratio of the amount of penalty recovered with control to the penalty with no control. This ratio is then used to rank different control strategies.

The advantage of the back-off approaches is that they determine the cost increase associated with moving to the back-off position which is attributed to the uncertainties and disturbances. A limitation of this approach is that it can lead to rather conservative designs since the worst-case uncertainty scenario is considered. Although the probability of the worst-case uncertainty occurring may not be high, this is the basis for the final design. Also, the method has not been applied to the design/synthesis problem. A fixed design is considered and then the back-off is considered as a modification of this design.

The optimal design of dynamic systems under uncertainty was addressed in [13]. Flexibility aspects as well as the control design were considered simultaneously with the process design. The algorithm is used to find the economic optimum which satisfies all of the constraints for a given set of uncertainties and disturbances when the control system is included.

S. Walsh and Perkins [23] outline the use of optimization as a tool for the design/control problem. They note that the advances in computational hardware and optimization tools have made it possible to solve the complex problems that arise in design/control. Their assessment focuses on the control structure selection

problem where the economic cost of a disturbance is balanced against the performance of the controller.

The increasing importance of design and control issues had lead to more and more discussion on the topic. One contribution to the area has been [11]. The fundamental design and control concepts are described and several quantitative examples are given which illustrate the interaction of design and control.

Most of the previous work does not address synthesis issues and does not treat the problem quantitatively. Two methods employ the optimization approach in process synthesis to arrive at mathematical programming formulations which are solved to determine the trade-offs between the steady-state design and dynamic controllability. The first method [9,10] uses steady state linear controllability measures while the second method [20] uses full nonlinear dynamic models of the process.

Process Synthesis

Mathematical programming has been found to be a very useful tool for process synthesis. Its application in analyzing the interaction of design and control has followed directly along the process synthesis methodology.

The goal in process synthesis to determine the structure and operating conditions of the process flowsheet. The optimization approach to the synthesis problem involves three steps:

1) The representation of process design alternatives of interest through a process *superstructure*.
2) The *mathematical modeling* of the superstructure.
3) The *algorithmic development* of solution procedure to extract the optimal process flowsheet from the superstructure and solution of the optimization problem.

The key aspect is the postulation of a superstructure which contains all possible design alternatives of interest. The superstructure must be sufficiently rich so as to include the numerous design possibilities yet succinct enough to eliminate redundancies and reduce complexities.

The mathematical model is characterized by the variables and equations used in the model. Continuous variables are used to represent flowrates, compositions, temperatures, etc. Binary variables are used to represent structural alternatives such as the existence of process units. The modeling of steady-state processes leads to algebraic equations and constraints and results in an MINLP. When dynamic models are to be used, the continuous variables are partitioned into dynamic state variables, control variables, and time invariant variables, and the resulting formulation is classified as a *mixed integer optimal control problem* (MIOCP).

Steady-State Modeling Approach

This approach was outlined in [9,10] and follows the optimization approach for process synthesis. A systematic procedure is presented for incorporating open-loop steady-state controllability measures into the process synthesis problem. The problem is formulated mathematically as a MINLP and a *multi-objective optimization* problem is solved to quantitatively determine the best-compromise solution among the economic and control objectives. The ϵ-constraint method is used to determine the *noninferior solution set* where one objective can be improved only at the expense of another, and the best-compromise solution is determined using a *cutting plane algorithm*.

In order to apply the process synthesis approach, the controllability measure must be expressed as a function of the unknown design parameters. Steady-state controllability measures are used to simplify the problem and reduce implementation difficulties that arise when considering controllability measures as functions of frequency. The steady-state gains of the process can be written in an analytical form thus allowing for an algebraic representation.

The starting point for the controllability analysis is the linear multiple input/multiple output system written in the Laplace domain as

$$\mathbf{z}(s) = \mathbf{G}(s)\mathbf{u}(s) + \mathbf{G}_d(s)\mathbf{d}(s),$$

where $\mathbf{z}$ are the output variables, $\mathbf{u}$ are the control variables, $\mathbf{G}(s)$ is the process transfer function matrix, and $\mathbf{G}_d(s)$ is the disturbance transfer function matrix.

Closed-loop control can be considered by expressing the control variable $\mathbf{u}(s)$ as

$$\mathbf{u}(s) = \mathbf{G}_c(s)(\mathbf{z}^*(s) - \mathbf{z}(s)),$$

where $\mathbf{G}_c(s)$ is the controller transfer function and $\mathbf{z}^*$ is the desired set-point. This requires that the form of

controller transfer function be known as well as the method for calculating the parameters. Since this causes problems in the formulation of the optimization problem, the controllability is viewed as a property inherent to the process and independent of the particular control system design. The analysis thus considers only the open-loop controllability measures which depend only on the process itself.

Since both the process design and controllability measures can be expressed as functions of the unknown design parameters, the synthesis problem can be expressed as a multi-objective MINLP:

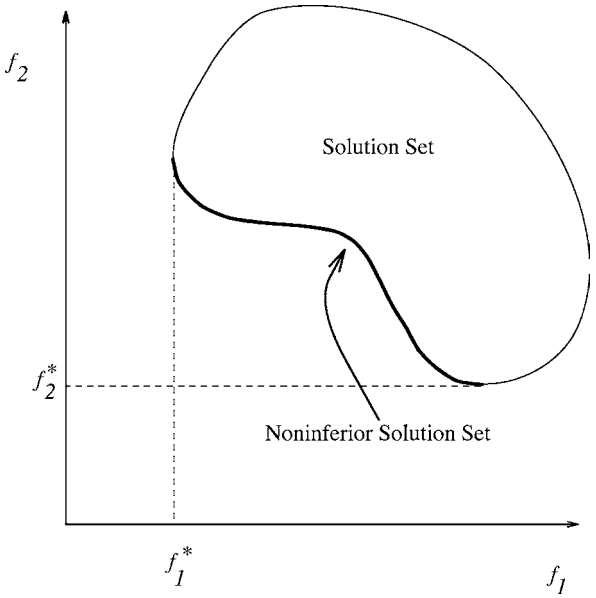
$$\begin{cases} \min & \mathbf{J}(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \boldsymbol{\eta} = \overline{\mathbf{h}}(\mathbf{x},\mathbf{y}) \\ & \mathbf{x} \in \mathsf{X} \subseteq \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q. \end{cases}$$

In this formulation, **J** is a vector of objectives which includes both the economic objectives and controllability objectives. The expressions **h** and **g** represent material and energy balances, thermodynamic relations, and other constraints. The controllability measures are included in the formulation as $\boldsymbol{\eta}$. The variables in this problem are partitioned as continuous **x** and binary **y**.

The problem is posed with multiple objectives representing the competing economic and open-loop controllability measures. Different techniques have been developed in order to assess the trade-offs among the objectives quantitatively. In this approach, the noninferior solution set is generated to determine the set of solutions in which one objective can be improved only at the expense of the other(s). The noninferior solution set for a two objective problem is visually depicted in Fig. 2.

This noninferior solution set is generated using an *ϵ-constraint method* where one objective is optimized and the others are included as constraints less than a parameter ϵ. The problem is reduced to a single objective optimization problem which is iteratively solved for varying values of ϵ to generate the noninferior solution set.

By reducing the problem to a single objective problem, MINLP optimization techniques can be applied



MINLP: Applications in the Interaction of Design and Control, Figure 2
Noninferior solution set for a problem with two objectives

to solve the problem. These MINLP techniques include *generalized Benders decomposition* (GBD) [7,19], outer approximation (OA) [3], outer approximation with equality relaxation (OA/ER) [8], and outer approximation with equality relaxation and augmented penalty [22]. These are discussed in detail in [6].

Once the noninferior solution set is determined, the best compromise solution is determined by applying a *cutting plane algorithm*. The trade-offs among the objectives are quantitatively assessed using weight factors which come from the slope of the noninferior solution set.

Dynamic Modeling Approach

The major limitation of the above approach is that is does not consider the dynamic behavior of the process. This approach considers the full dynamic model of the process and a dynamic controllability measure. An optimization approach is applied which involves a dynamic optimization problem.

One of the initial difficulties with this method is defining a controllability measure for nonlinear dynamic systems. As in the previous method, the controllability measure must be capable of being expressed as

a function of the unknown design parameters. One possible choice for the controllability measure is the integral square error (ISE). The benefit of this measure is that it is easy to calculate and and does reflect the dynamics of the process albeit only in the outputs of the process. One downside of this measure is that there is no one to one correspondence between the the control structure and the ISE measure. Thus, different dynamic characteristics of the process may not be reflected in the ISE.

The superstructure is the same as in the previous approach, but a dynamic model is used instead of a steady-state model. The dynamic modeling of the superstructure leads to a problem that includes *differential and algebraic equations* (DAEs) and the formulation is a multi-objective MIOCP. New algorithmic techniques must be developed for the solution of the formulation.

The general formulation for the multi-objective MIOCP is as follows:

$$
\begin{cases}
\min & \mathbf{J}(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}) \\
\text{s.t.} & \mathbf{f}_1(\dot{\mathbf{z}}_1(t), \mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{u}(t), \mathbf{x}, \mathbf{y}, t) = \mathbf{0} \\
& \mathbf{f}_2(\mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{u}(t), \mathbf{x}, \mathbf{y}, t) = \mathbf{0} \\
& \mathbf{z}_1(t_0) = \mathbf{z}_1^0 \\
& \mathbf{z}_2(t_0) = \mathbf{z}_2^0 \\
& \mathbf{h}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}) = \mathbf{0} \\
& \mathbf{g}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\
& \mathbf{h}''(\mathbf{x}, \mathbf{y}) = \mathbf{0} \\
& \mathbf{g}''(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\
& \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^p \\
& \mathbf{y} \in \{0, 1\}^q \\
& t_i \in [t_0, t_N] \\
& i = 0, \ldots, N.
\end{cases} \tag{1}
$$

Here, $\mathbf{z}_1(t)$ is a vector of n dynamic variables whose time derivatives, $\dot{\mathbf{z}}_1(t)$, appear explicitly, and $\mathbf{z}_2(t)$ is a vector of m dynamic variables whose time derivatives do not appear explicitly, $\mathbf{x}$ is a vector of p time invariant continuous variables, $\mathbf{y}$ is a vector of q binary variables, and $\mathbf{u}(t)$ is a vector of r control variables. Time t is the independent variable for the DAE system where t_0 is the fixed initial time, t_i are time instances, and t_N is the final time. The DAE system is represented by $\mathbf{f}_1$, the n differential equations, and $\mathbf{f}_2$, the m dynamic algebraic equations. The constraints $\mathbf{h}'$ and $\mathbf{g}'$ are point constraints where t_i represents the time instance at which the constraint is enforced and $\mathbf{h}''$ and $\mathbf{g}''$ are general constraints. The objective functions for the economic and controllability measures are represented by the vector $\mathbf{J}$.

The initial condition for the above system is determined by specifying n of the $2n + m$ variables $\mathbf{z}_1(t_0)$, $\dot{\mathbf{z}}_1(t_0)$, $\mathbf{z}_2(t_0)$. For DAE systems with index 0 or 1, the remaining $n + m$ values can be determined. In this work, DAE systems of index 0 or 1 are considered and the initial conditions for $\mathbf{z}_1(t)$ and $\mathbf{z}_2(t)$ are $\mathbf{z}_1^0$ and $\mathbf{z}_2^0$ respectively.

Note that in this general formulation, the $\mathbf{y}$ variables appear in the DAE system as well as in the point constraints and general constraints. This has implications on the solution strategy.

A similar approach to that of the previous approach is applied to address the multi-objective nature of the problem. An ϵ-constraint method is applied to reduce to problem to an iterative solution of single objective MIOCPs.

MIOCP Solution Algorithm

The strategy for solving the MIOCP is to apply iterative decomposition strategies similar to existing MINLP algorithms with extensions for handling the DAE system. The algorithm developed for the solution of the MIOCP closely parallels existing algorithms for MINLP optimization (GBD, OA, OA/ER, OA/ER/AP). The presence of the $\mathbf{y}$ variables in DAE system for the general case prohibits the use of Outer Approximation and its variants. For the special cases where the $\mathbf{y}$ variables do not appear in the DAEs and do participate in a linear and separable fashion, outer approximation and its variants can be applied to the problem. The GBD algorithm can be applied to the solution of the general problem, and the algorithmic development closely follows those of GBD.

The GBD algorithm is an iterative procedure which generates upper and lower bounds on the solution of the MINLP formulation. The upper bound results from the solution of an NLP primal problem and the lower bound from an MILP master problem. The bounds on the solution converge in a finite number of iterations to yield the solution to the MINLP model. A similar

methodology is applied to the MIOCP problem, but the forms of the primal and master problems have to be altered.

Primal Problem

The primal problem is obtained by fixing the $\mathbf{y}$ variables which leads to an optimal control problem. For fixed values of $\mathbf{y} = \mathbf{y}^k$, the MIOCP has the following form:

$$\begin{cases} \min & \mathbf{J}(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}^k) \\ \text{s.t.} & \mathbf{f}_1(\dot{\mathbf{z}}_1(t), \mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{u}(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0} \\ & \mathbf{f}_2(\mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{u}(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0} \\ & \mathbf{z}_1(t_0) = \mathbf{z}_1^0 \\ & \mathbf{z}_2(t_0) = \mathbf{z}_2^0 \\ & \mathbf{h}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}^k) \le \mathbf{0} \\ & \mathbf{h}''(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}''(\mathbf{x}, \mathbf{y}^k) \le \mathbf{0} \\ & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^p \\ & t_i \in [t_0, t_N] \\ & i = 0, \ldots, N. \end{cases} \tag{2}$$

The solution of this optimal control problem can be handled in several ways: complete discretization, solution of the necessary conditions, dynamic programming, and *control parameterization*. This work focuses on the control parameterization techniques which parameterize only the control variables $\mathbf{u}(t)$ in terms of time invariant parameters. At each step of the optimization procedure, the DAEs are solved for given values of the decision variables and a feasible path for $\mathbf{z}(t)$ is obtained. This solution is used to evaluate the objective function and remaining constraints. The control parameterization can either be open loop as described in [21] or closed-loop such as that described in [17] and [16] which also includes the control structure selection.

The basic idea behind the control parameterization is to express the control variables $\mathbf{u}(t)$ as functions of time invariant parameters. This parameterization can be done in terms of the independent variable t (open loop):

$$\mathbf{u}(t) = \phi(\mathbf{w}, t).$$

Alternatively, the parameterization can be done in terms of the state variables $\mathbf{z}(t)$ (closed-loop):

$$\mathbf{u}(t) = \psi(\mathbf{w}, \dot{\mathbf{z}}(t), \mathbf{z}(t)).$$

In both cases, $\mathbf{w}$ are the time invariant control parameters. The set of time invariant parameters, $\mathbf{x}$, is now expanded to include the control parameters:

$$\mathbf{x} = \{\mathbf{x}, \mathbf{w}\}.$$

The set of DAEs ($\mathbf{f}$) is expanded to include parameterization functions

$$\mathbf{f}(\cdot) = \{\mathbf{f}(\cdot), \phi(\cdot), \psi(\cdot)\}$$

and the control variables are converted to dynamic state variables:

$$\mathbf{z} = \{\mathbf{z}, \mathbf{u}\}.$$

Through the application of the control parameterization, the control variables are effectively removed from the problem and the following problem results:

$$\begin{cases} \min & J(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) \\ \text{s.t.} & \mathbf{f}_1(\dot{\mathbf{z}}_1(t), \mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0} \\ & \mathbf{f}_2(\mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0} \\ & \mathbf{z}_1(t_0) = \mathbf{z}_1^0 \\ & \mathbf{z}_2(t_0) = \mathbf{z}_2^0 \\ & \mathbf{h}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) \le \mathbf{0} \\ & \mathbf{h}''(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}''(\mathbf{x}, \mathbf{y}^k) \le \mathbf{0} \\ & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^p \\ & t_i \in [t_0, t_N] \\ & i = 0, \ldots, N. \end{cases} \tag{3}$$

This problem is a nonlinear program with differential and algebraic constraints (NLP/DAE). This problem is solved using a parametric method where the DAE system is solved as a function of the $\mathbf{x}$ variables. The solution of the DAE system is achieved through an integration routine which returns the values of the $\mathbf{z}$ variables at the time instances, $\mathbf{z}(t_i)$, along with their sensitivities with respect to the parameters, $\mathrm{d}\mathbf{z}/\mathrm{d}\mathbf{x}(t_i)$. The resulting problem is an NLP optimization over the space

of $\mathbf{x}$ variables which has the form:

$$\begin{cases} \min & J(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) \\ \text{s.t.} & \mathbf{h}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) \le \mathbf{0} \\ & \mathbf{h}''(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}''(\mathbf{x}, \mathbf{y}^k) \le \mathbf{0} \\ & \mathbf{x} \in \mathcal{X} \\ & t_i \in [t_0, \ldots, t_N] \\ & i = 0, \ldots, N, \end{cases} \tag{4}$$

where the variables $\dot{\mathbf{z}}_1(t_i)$, $\mathbf{z}_1(t_i)$, and $\mathbf{z}_2(t_i)$ are determined through the solution of the DAE system by integration:

$$\begin{cases} \mathbf{f}_1(\dot{\mathbf{z}}_1(t), \mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0}, \\ \mathbf{f}_2(\mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0}, \\ \mathbf{z}_1(t_0) = \mathbf{z}_1^0, \\ \mathbf{z}_2(t_0) = \mathbf{z}_2^0. \end{cases} \tag{5}$$

The functions $J(\cdot)$, $\mathbf{g}'(\cdot)$, and $\mathbf{h}'(\cdot)$ are functions of $\mathbf{z}(t_i)$ which are implicit functions of the $\mathbf{x}$ variables through the integration of the DAE system. For the solution of the NLP the objective and constraints evaluations, along with their gradients with respect to $\mathbf{x}$, are required. These are evaluated directly for the constraints $\mathbf{g}''(\mathbf{x})$ and $\mathbf{h}''(\mathbf{x})$. However, for the functions $J(\cdot)$, $\mathbf{g}'(\cdot)$, and $\mathbf{h}'(\cdot)$, the values $\mathbf{z}(t_i)$, and the gradients $d\mathbf{z}/d\mathbf{x}(t_i)$, as returned from the integration, are used. The functions $J(\cdot)$, $\mathbf{g}'(\cdot)$, and $\mathbf{h}'(\cdot)$ are evaluated directly and the gradients $dJ/d\mathbf{x}$, $d\mathbf{g}_i'/d\mathbf{x}$, and $d\mathbf{h}'/d\mathbf{x}$ are evaluated by using the chain rule:

$$\begin{cases} \frac{dJ}{d\mathbf{x}} = \left(\frac{\partial J}{\partial \mathbf{z}}\right)\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) + \left(\frac{\partial J}{\partial \mathbf{x}}\right), \\ \frac{d\mathbf{h}_i'}{d\mathbf{x}} = \left(\frac{\partial \mathbf{h}_i'}{\partial \mathbf{z}}\right)\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) + \left(\frac{\partial \mathbf{h}_i'}{\partial \mathbf{x}}\right), \\ \frac{d\mathbf{g}_i'}{d\mathbf{x}} = \left(\frac{\partial \mathbf{g}_i'}{\partial \mathbf{z}}\right)\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) + \left(\frac{\partial \mathbf{g}_i'}{\partial \mathbf{x}}\right). \end{cases} \tag{6}$$

Standard gradient based optimization techniques can be applied to solve this problem as an NLP. The solution of this problem provides values of the $\mathbf{x}$ variables and trajectories for $\mathbf{z}(t)$.

The master problem is formulated using *dual information* and the solution of the primal problem. Provided that the $\mathbf{y}$ variables participate linearly, the problem is an MILP whose solution provides a lower bound

MINLP: Applications in the Interaction of Design and Control, Table 1
Constraints and their corresponding dual variables

constraint	dual variable
$\mathbf{f}_1$	$\boldsymbol{\nu}_1(t)$
$\mathbf{f}_2$	$\boldsymbol{\nu}_2(t)$
$\mathbf{g}'$	$\boldsymbol{\mu}'$
$\mathbf{h}'$	$\boldsymbol{\lambda}'$
$\mathbf{g}''$	$\boldsymbol{\mu}''$
$\mathbf{h}''$	$\boldsymbol{\lambda}''$

and $\mathbf{y}$ variables for the next primal problem. Dual information is required from all of the constraints including the DAEs whose dual variables, or adjoint variables, are dynamic. The constraints and their corresponding dual variables are listed in Table 1.

The dual variables $\boldsymbol{\mu}'$, $\boldsymbol{\lambda}'$, $\boldsymbol{\mu}''$, and $\boldsymbol{\lambda}''$ are generally obtained from the solution technique for the primal problem. Dual information from the DAE system is obtained by solving the *adjoint problem* for the DAE system which has the following formulation:

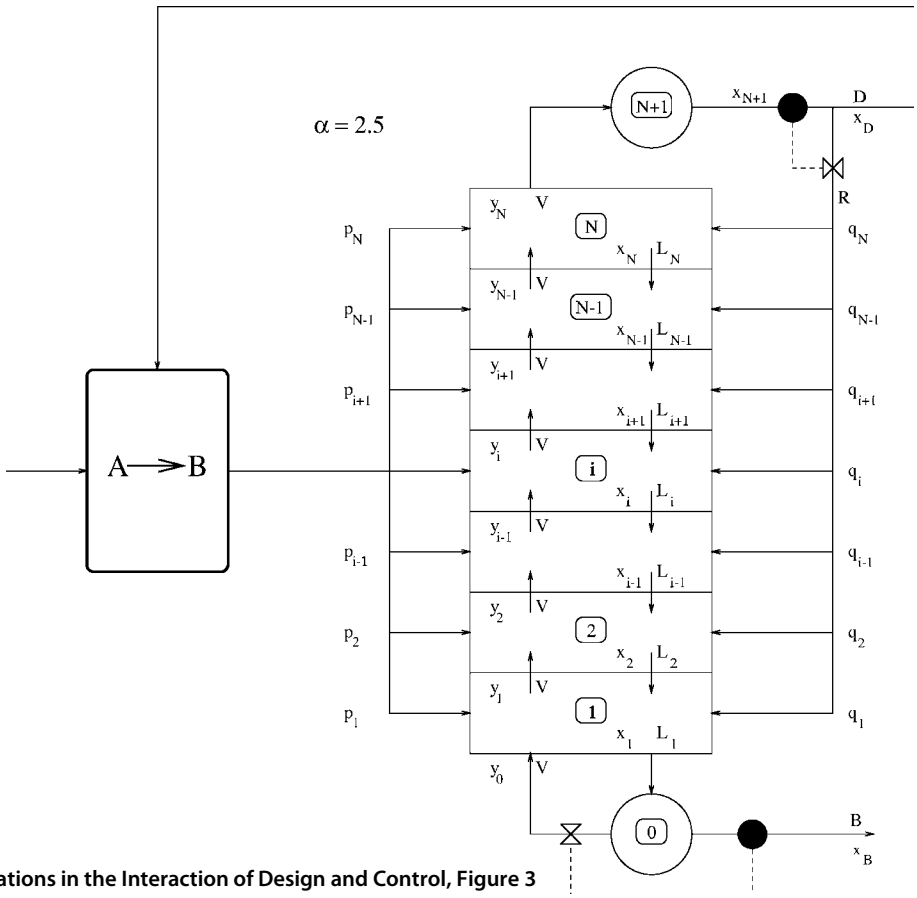
$$\begin{cases} \mathbf{p} = \boldsymbol{\nu}_1^\top \frac{d\mathbf{f}_1}{d\dot{\mathbf{z}}_1}, \\ \dot{\mathbf{p}} = \boldsymbol{\nu}_1^\top \frac{d\mathbf{f}_1}{d\mathbf{z}_1} + \boldsymbol{\nu}_2^\top \frac{d\mathbf{f}_2}{d\mathbf{z}_1}, \\ \mathbf{0} = \boldsymbol{\nu}_1^\top \frac{d\mathbf{f}_1}{d\mathbf{z}_2} + \boldsymbol{\nu}_2^\top \frac{d\mathbf{f}_2}{d\mathbf{z}_2}. \end{cases} \tag{7}$$

This is a set of DAEs where the solutions for $d\mathbf{f}_1/d\dot{\mathbf{z}}_1$, $d\mathbf{f}_1/d\mathbf{z}_1$, $d\mathbf{f}_2/d\mathbf{z}_1$, $d\mathbf{f}_1/d\mathbf{z}_2$, and $d\mathbf{f}_2/d\mathbf{z}_2$ are known functions of time obtained from the solution of the primal problem. The variables $\boldsymbol{\nu}_1(t)$ and $\boldsymbol{\nu}_2(t)$ are the adjoint variables and the solution of this problem is a backward integration in time with the following final time conditions:

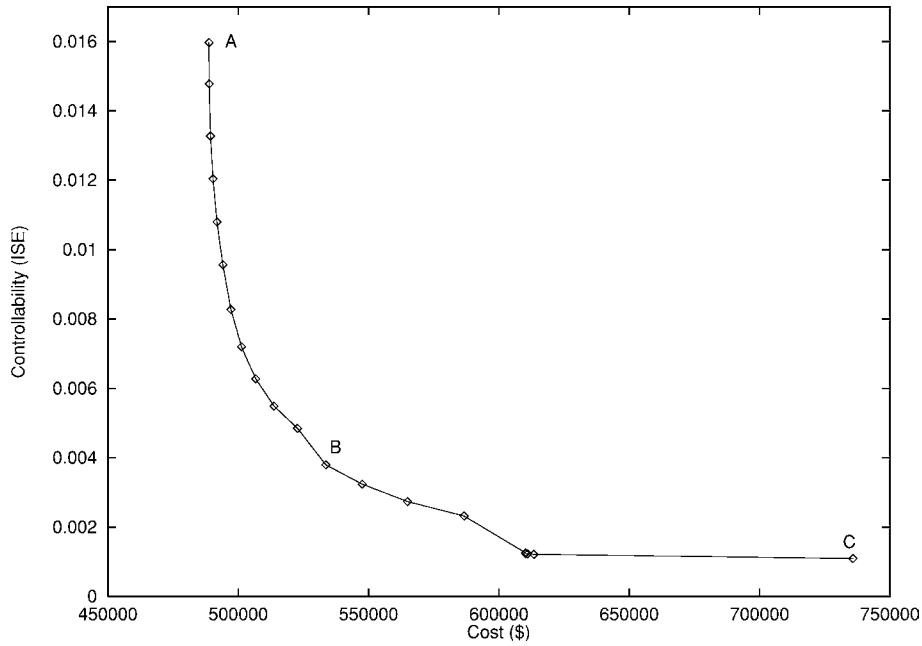
$$\frac{d\mathbf{J}}{d\mathbf{z}_1} + \boldsymbol{\lambda}' \frac{d\mathbf{h}'}{d\mathbf{z}_1} + \boldsymbol{\mu}' \frac{d\mathbf{g}'}{d\mathbf{z}_1} + \boldsymbol{\nu}_1^\top \frac{d\mathbf{f}_1}{d\dot{\mathbf{z}}_1} = 0.$$

Thus, the Lagrange multipliers for the end-time constraints are used as the final time conditions for the adjoint problem and are not included in the master problem formulation.

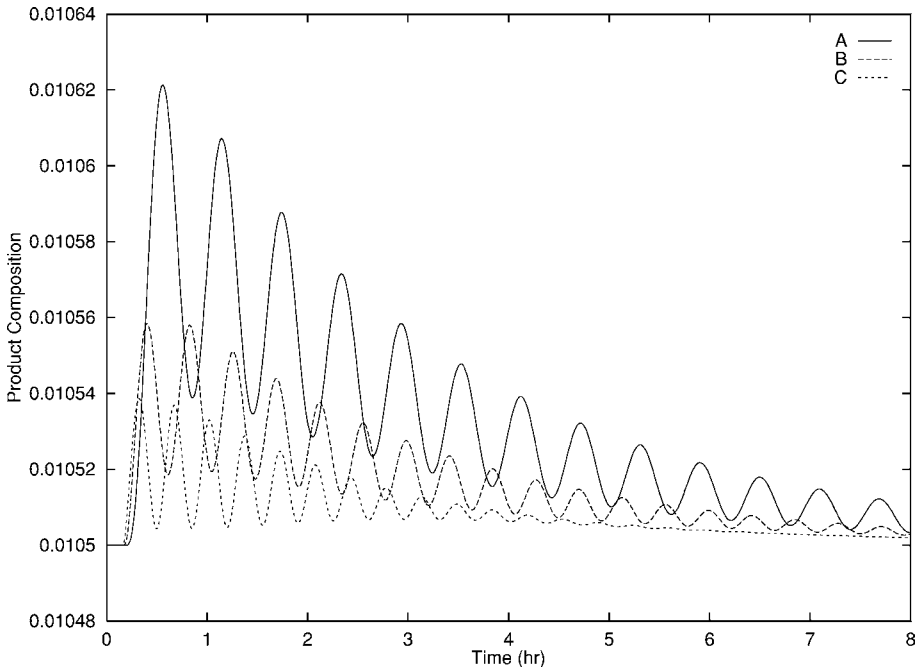
The master problem is formulated using the solution of the primal problem, $\mathbf{x}^k$ and $\mathbf{z}^k(t)$, along with



MINLP: Applications in the Interaction of Design and Control, Figure 3
Superstructure for reactor-separator-recycle system



MINLP: Applications in the Interaction of Design and Control, Figure 4
Noninferior solution set for the reactor-separator-recycle system



MINLP: Applications in the Interaction of Design and Control, Figure 5
Dynamic responses of product compositions for three designs

the dual information, $\boldsymbol{\mu}''^k$, $\boldsymbol{\lambda}''^k$, and $\boldsymbol{\nu}^k(t)$. The master problem has the following form:

$$
\begin{cases}
\min\limits_{\mathbf{y},\mu_b} & \mu_b \\
\text{s.t.} & \mu_b \geq J(\mathbf{x}^k, \mathbf{y}) \\
& \quad + \int_{t_0}^{t_N} \nu_1^k(t)\mathbf{f}_1(\dot{\mathbf{z}}_1^k(t), \mathbf{z}_1^k(t), \mathbf{z}_2^k(t), \mathbf{x}^k, \mathbf{y}, t)\, dt \\
& \quad + \int_{t_0}^{t_N} \nu_2^k(t) \\
& \qquad \mathbf{f}_2(\mathbf{z}_1^k(t), \mathbf{z}_2^k(t), \mathbf{x}^k, \mathbf{y}, t)\, dt \\
& \quad + \boldsymbol{\mu}''^k\mathbf{g}''(\mathbf{x}^k, \mathbf{y}) + \boldsymbol{\lambda}''^k\mathbf{h}''(\mathbf{x}^k, \mathbf{y}), \\
& \qquad k \in K_{\text{feas}}, \\
& \mathbf{0} \geq \int_{t_0}^{t_N} \nu_1^k(t) \\
& \quad \mathbf{f}_1(\dot{\mathbf{z}}_1^k(t), \mathbf{z}_1^k(t), \mathbf{z}_2^k(t), \mathbf{x}^k, \mathbf{y}, t)\, dt \\
& \quad + \int_{t_0}^{t_N} \nu_2^k(t)\mathbf{f}_2(\mathbf{z}_1^k(t), \mathbf{z}_2^k(t), \mathbf{x}^k, \mathbf{y}, t)\, dt \\
& \quad + \boldsymbol{\mu}''^k\mathbf{g}''(\mathbf{x}^k, \mathbf{y}) + \boldsymbol{\lambda}''^k\mathbf{h}''(\mathbf{x}^k, \mathbf{y}), \\
& \qquad k \in K_{\text{infeas}}, \\
& \mathbf{y} \in \{0, 1\}^q.
\end{cases}
\tag{8}
$$

The integral term can be evaluated since the profiles for $\mathbf{z}^k(t)$ and $\nu^k(t)$ both are fixed and known. Note that this formulation has no restrictions on whether or not $\mathbf{y}$ variables participate in the the DAE system.

Example: Reactor-Separator-Recycle System

The example problem considered here is the design of a process involving a reaction step, a separation step, and a recycle loop. Fresh feed containing A and B flow into a an isothermal reactor where the first order irreversible reaction $A \rightarrow B$ takes place. The product from the reactor is sent to a distillation column where the unreacted A is separated from the product B and sent back to the reactor. The superstructure is shown in Fig. 3.

The model equations for the reactor (CSTR) and the separator (ideal binary distillation column) can be found in [12]. The specific problem design follows the work in [10].

For this problem, the single output is the product composition. The bottoms (product) composition is controlled by the vapor boil-up and the distillate composition is controlled by the reflux rate. Since only the product composition is specified, the distillate compo-

sition set-point is free and left to be determined through the optimization.

The cost function includes column and reactor capital and utility costs.

$$\begin{aligned}
\text{cost}_{\text{reactor}} &= 17639 D_r^{1.066} (2D_r)^{0.802}, \\
\text{cost}_{\text{column}} &= 6802 D_c^{1.066} (2.4 N_t)^{0.802} \\
&\quad + 548.8 D_c^{1.55} N_t, \\
\text{cost}_{\text{exchangers}} &= 193023 V_{ss}^{0.65}, \\
\text{cost}_{\text{utilities}} &= 72420 V_{ss}, \\
\text{cost}_{\text{total}} &= \frac{\text{cost}_{\text{reactor}} + \text{cost}_{\text{column}} + \text{cost}_{\text{exchangers}}}{\beta_{\text{pay}}} \\
&\quad + \beta_{\text{tax}} [\text{cost}_{\text{utilities}}].
\end{aligned}$$

The controllability measure is the time weighted ISE for the product composition:

$$\frac{d\mu}{dt} = t(x_B - x_B^*)^2.$$

The noninferior solution set is shown in Fig. 4, and Table 2 lists the solution information for three of the designs in the noninferior solution set. The dynamic profile for these three designs are shown in Fig. 5.

All of the designs in the noninferior solution set are strippers. Since the feed enters at the top of the column, there is no reflux and thus no control loop for the distillate composition. The controllability of the process is increased by increasing the size of the reactor and decreasing the size of the column. The most controllable design has a large reactor and a single flash unit.

MINLP: Applications in the Interaction of Design and Control, Table 2
Solution information for three designs

Solution	A	B	C
Cost($)	489, 000	534, 000	736, 000
Capital($)	321, 000	364, 000	726, 000
Utility($)	168, 000	170, 000	10, 000
ISE	0.0160	0.00379	0.0011
Trays	19	8	1
Feed	19	8	1
V_r(kmol)	2057.9	3601.2	15000
V(kmol/hr)	138.94	141.25	85.473
K_V	90.94	80.68	87.40
τ_V (hr)	0.295	0.0898	0.0156

See also

- Chemical Process Planning
- Control Vector Iteration
- Duality in Optimal Control with First Order Differential Equations
- Dynamic Programming: Continuous-time Optimal Control
- Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- Dynamic Programming: Optimal Control Applications
- Extended Cutting Plane Algorithm
- Generalized Benders Decomposition
- Generalized Outer Approximation
- Hamilton–Jacobi–Bellman Equation
- Infinite Horizon Control and Dynamic Games
- MINLP: Application in Facility Location-allocation
- MINLP: Applications in Blending and Pooling Problems
- MINLP: Branch and Bound Global Optimization Algorithm
- MINLP: Branch and Bound Methods
- MINLP: Design and Scheduling of Batch Processes
- MINLP: Generalized Cross Decomposition
- MINLP: Global Optimization with αBB
- MINLP: Heat Exchanger Network Synthesis
- MINLP: Logic-based Methods
- MINLP: Outer Approximation Algorithm
- MINLP: Reactive Distillation Column Synthesis
- Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- Mixed Integer Nonlinear Programming
- Multi-objective Optimization: Interaction of Design and Control
- Optimal Control of a Flexible Arm
- Robust Control
- Robust Control: Schur Stability of Polytopes of Polynomials
- Semi-infinite Programming and Control Problems
- Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- Suboptimal Control

References

1. Bahri PA, Bandoni JA, Romagnoli JA (1996) Effect of disturbances in optimizing control: Steady-state open-loop backoff problem. AIChE J 42(4):983–994

2. Brengel DD, Seider WD (1992) Coordinated design and control optimization of nonlinear processes. Comput Chem Eng 16(9):861–886
3. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
4. Elliott TR, Luyben WL (1995) Capacity-based approach for the quantitative assessment of process controllability during the conceptual design stage. Industr Eng Chem Res 34:3907–3915
5. Figueroa JL, Bahri PA, Bandoni JA, Romagnoli JA (1996) Economic impact of disturbances and uncertain parameters in chemical processes – A dynamic back-off analysis. Comput Chem Eng 20(4):453–461
6. Floudas CA (1995) Nonlinear and mixed integer optimization: Fundamentals and applications. Oxford Univ. Press, Oxford
7. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10(4):237–260
8. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flow sheets. Industr Eng Chem Res 26(9):1869
9. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control–1. A multiobjective framework and application to binary distillation synthesis. Comput Chem Eng 18(10):933–969
10. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control–2. Reactor-separator-recycle system. Comput Chem Eng 18(10):971–994
11. Luyben ML, Luyben WL (1997) Essentials of process control. McGraw-Hill, New York
12. Luyben WL (1990) Process modeling, simulation, and control for chemical engineers, 2nd edn. McGraw-Hill, New York
13. Mohideen MJ, Perkins JD, Pistikopoulos EN (1996) Optimal design of dynamic systems under uncertainty. AIChE J 42(8):2251–2272
14. Morari M, Perkins J (1994) Design for operations. FOCAPD Conf Proc
15. Morari M, Zafiriou E (1989) Robust process control. Prentice-Hall, Englewood Cliffs, NJ
16. Narraway LT, Perkins JD (1993) Selection of control structure based on economics. Comput Chem Eng 18:S511–515
17. Narraway LT, Perkins JD (1993) Selection of process control structure based on linear dynamic economics. Industr Eng Chem Res 32:2681–2692
18. Narraway LT, Perkins JD, Barton GW (1991) Interaction between process design and process control: Economic analysis of process dynamics. J Process Control 1:243–250
19. Paules IV GE, Floudas CA (1989) APROS: Algorithmic development methodology for discrete-continuous optimization problems. Oper Res 37(6):902–915
20. Schweiger CA, Floudas CA (1997) Interaction of design and control: Optimization with dynamic models. In: Hager WW, Pardalos PM (eds) Optimal Control: Theory, Algorithms, and Applications. Kluwer, Dordrecht, pp 388–435
21. Vassiliadis VS, Sargent RWH, Pantelides CC (1994) Solution of a class of multistage dynamic optimization problems 1. Problems without path constraints. Industr Eng Chem Res 33:2111–2122
22. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer approximation method for MINLP optimization. Comput Chem Eng 14(7):769–782
23. Walsh S, Perkins JD (1996) Operability and control in process synthesis and design. In: Anderson JL (eds) Adv Chem Engin, vol 23. Acad. Press, New York, pp 301–402

MINLP: Branch and Bound Global Optimization Algorithm

Ioannis P. Androulakis
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 90C10, 90C26

Article Outline

Keywords
See also
References

Keywords

Mixed integer nonlinear programming; Global optimization; Branch and bound algorithms

A wide range of nonlinear optimization problems involve integer or discrete variables in addition to continuous ones. These problem are denoted as *mixed integer nonlinear programming* (MINLP) problems. Integer variables correspond to logical decision describing whether certain actions do or do not take place, or modeling the sequence according to which those decisions take place. The nonlinear nature of the MINLP models may arise from:

- nonlinear relations in the integer domain only
- nonlinear relations in the continuous domain only
- nonlinear relations in the joint domain, i. e., products of continuous and binary/integer variables.

The general mathematical formulation of the MINLP problems can be stated as follows:

$$\begin{cases} \min\limits_{x,y} & f(x,y) \\ \text{s.t.} & h(x,y) = 0 \\ & g(x,y) \le 0 \\ & x \in X \subset \mathbb{R}^n \\ & y \in Y \quad \text{(integer)}. \end{cases}$$

Here, x represents a vector of n continuous variables, y is a vector of integer variables, $f(x, y)$, $h(x, y)$, $g(x, y)$ represent the objective function, equality and inequality constraints, respectively. It should be noted, that every problem of the form just presented, can be transformed into one where all integer variables have been transformed into binary, i. e., 0–1, variables, by realizing that every integer $y^L \le y \le y^U$ can be expressed through 0–1 variables, $z = (z_1, \ldots, z_N)$, as:

$$y = y^L + z_1 + 2z_2 + 4z_3 + \cdots + 2^{N-1} z_N,$$
$$N = 1 + \text{INT}\left\{ \frac{\log(y^U - y^L)}{\log 2} \right\}.$$

Therefore, any MINLP problem can be written as:

$$\begin{cases} \min\limits_{x,y} & f(x,y) \\ \text{s.t.} & h(x,y) = 0 \\ & g(x,y) \le 0 \\ & x \in X \subset \mathbb{R}^n \\ & y \in Y = \{0,1\}^m. \end{cases}$$

In the analysis of MINLP problems two issues are of paramount importance:

- combinatorial explosion of computational requirements as the number of binary variables increases
- *NP*-hard nature of the problem of determining the *global minimum solution* of general nonconvex MINLP problems.

A complexity analysis of the former is presented in [16], while the complexity of determining global minimum solutions of MINLPs is discussed in [15].

Various methods exist for identifying a locally optimum solution of MINLP problems. These are discussed in great detail in [9] and in a recent thorough review paper, [6], which presents a comprehensive account of the various approaches for addressing issues related to the solution of mixed integer nonlinear optimization problems.

The main objective in a general branch and bound algorithm is to perform an enumeration of the alternatives without examining all 0–1 combinations of the binary variables. A key element in such an enumeration if the representation of alternatives via a *binary tree*. The basic ideas in a branch and bound algorithm are the following. First, a reasonable effort is made in solving the original problem, by considering for instance the continuous relaxation of it. If the relaxation does not result in an integer-feasible solution, i. e., one in which the binary variables achieve 0–1 at the optimal point, them the root node is separated into two candidate subproblems which are subsequently solved. The separation aims at creating simpler instances of the original problem. Until the problem is successfully solved this process of generating candidate subproblems is repeated. Branch and bound algorithms are also known as *divide-and-conquer* for that very reason. A basic principle common to all branch and bound algorithms is that the solution of the subproblems aims at generating *valid lower bounds* on the original MINLP through its relaxation to a continuous problem. The relaxation, in the case of MINLP, results in a nonlinear programming problem (NLP) which, in the general case, is nonconvex and needs to be solved to global optimality so as to provide a valid lower bound. If the NLP relaxation renders an integer solution, then this solution is referred to as *valid upper bound*. The generation of the sequence of valid upper and lower bounds is called *bounding step*. The way subproblems are created is by forcing some of the binary variables to take on a value of 0 or 1. This is known as the *branching step*. Nodes in the tree are pruned when the corresponding valid lower bound exceeds the valid upper bound, this stage is know as the *fathoming step*. The selection of the branching node, the branching variable and the generation of the lower bound are very crucial steps whose importance becomes even more pronounced when addressing nonconvex MINLP problems. Two basic strategies exists regarding the selection of the branching node depending on whether one designs a branch and bound based on a *depth-first* or a *breadth-first* approach. In the former, the last node created is selected for branching, in the latter the node that generated the best lower bound is selected. It is not clear which strategy is the

best and it is often that the one that minimizes the computational requirement is selected, [13]. Another alternative is to select nodes based on the deviation of the solution from integrality, [12]. The most common strategy for selecting a branching variable is to select the variable whose value at the solution of some relaxed problem is the farthest from integer, i. e., the most fractional variable, [17]. In [12] a method based on the concept of *pseudocosts* which quantifies the effect of binary variables is also proposed, which assigns essentially priorities on the order of branching variables. Finally, one of the most important computational step is the generation of the lower bound, in other words the solution of the relaxed problem. The effectiveness of a branch and bound depends of the quality of the lower bound that is generated. At every node of the branch and bound tree a nonlinear-nonconvex NLP is solved. Two issues are important: the lower bound must be valid, in other words the relaxation at a particular node must underestimate the solution of the original problem for this node, and the lower bounds must be tight so as to enhance the fathoming step. The key complexity when dealing with nonconvex MINLPs is that the relaxation solved at each node is, of course, a nonconvex NLP that has to be solved to global optimality. With the exception of problems which are convex in the x and relaxed y-space for which variants of the branch and bound algorithms will lead the correct solution, [18], in all other cases *global optimization* algorithms have to be employed for the generation of valid lower bounds.

In [19] the scope of branch and bound algorithms was extended to problems for which valid convex underestimating NLPs can be constructed for the convex relaxations. The problems included bilinear and separable problems for which convex underestimators can be build [14]. A number of very useful tests were proposed to accelerate the reduction of solution space. Namely:

1) Optimality based range reduction tests: For the first set of tests, an upper bound U on the nonconvex MINLP must be computed and a convex lower bounding NLP must be solved to obtain a lower bound L. If a bound constraint for variable x_i, with $x_i^L \leq x_i \leq x_i^U$, is active at the solution of the convex NLP and has multiplier $\lambda_i^* > 0$, the bounds on x_i can be updated as follows:
   a) If $x_i - x_i^U = 0$ at the solution of the convex NLP and $\kappa_i = x_i^U - (U - L)/\lambda_i^*$ is such that $\kappa_i > x_i^L$, then $x_i^L = \kappa_i$.
   b) If $x_i - x_i^L = 0$ at the solution of the convex NLP and $\kappa_i = x_i^L + (U - L)/\lambda_i^*$ is such that $\kappa_i < x_i^U$, then $x_i^U = \kappa_i$.

   If neither bound constraint is active at the solution of the convex NLP for some variable x_j, the problem can be solved by setting $x_j = x_j^U$ or $x_j = x_j^L$. Tests similar to those presented above are then used to update the bounds on x_j.
2) Feasibility based range reduction tests: In addition to ensuring that tight bounds are available for the variables, the constraint underestimators are used to generate new constraints for the problem. Consider the constraint $g_i(x, y) \leq 0$. If its underestimating function $\underline{g}_i(x, y) = 0$ at the solution of the convex NLP and its multiplier is $\mu_i^* > 0$, the constraint

$$\underline{g}_i(x, y) \geq -\frac{U - L}{\mu_i^*}$$

   can be included in subsequent problems.

A global optimization algorithm branch and bound algorithm has been proposed in [20]. It can be applied to problems in which the objective and constraints are functions involving any combination of binary arithmetic operations (addition, subtraction, multiplication and division) and functions that are either concave over the entire solution space (such as ln) or convex over this domain (such as exp).

The algorithm starts with an automatic reformulation of the original nonlinear problem into a problem that involves only linear, bilinear, linear fractional, simple exponentiation, univariate concave and univariate convex terms. This is achieved through the introduction of new constraints and variables. The reformulated problem is then solved to global optimality using a branch and bound approach. Its special structure allows the construction of a convex relaxation at each node of the tree. The integer variables can be handled in two ways during the generation of the convex lower bounding problem. The integrality condition on the variables can be relaxed to yield a convex NLP which can then be solved globally. Alternatively, the integer variables can be treated directly and the convex lower bounding MINLP can be solved using a branch and bound algorithm as described earlier. This second ap-

proach is more computationally intensive but is likely to result in tighter lower bounds on the global optimum solution. In order to obtain an upper bound for the optimum solution, several methods have been suggested. The MINLP can be transformed to an equivalent nonconvex NLP by relaxing the integer variables. For example, a variable $y \in \{0, 1\}$ can be replaced by a continuous variable $z \in [0, 1]$ by including the constraint $z - z \cdot z = 0$. The nonconvex NLP is then solved locally to provide an upper bound. Finally, the discrete variables could be fixed to some arbitrary value and the nonconvex NLP solved locally.

In [1] *SMIN* was proposed which is designed to address the following class of problems to global optimality:

$$\begin{cases} \min & f(x) + x^\top A_0 y + c_0^\top y \\ \text{s.t.} & h(x) + x^\top A_1 y + c_1^\top y = 0 \\ & g(x) + x^\top A_2 y + c_2^\top y \le 0 \\ & x \in X \subset \mathbb{R}^n \\ & y \in Y \quad \text{(integer)}, \end{cases}$$

where $c_0^\top$, $c_1^\top$ and $c_2^\top$ are constant vectors, A_0, A_1 and A_2 are constant matrices and $f(x)$, $h(x)$ and $g(x)$ are functions with continuous second order derivatives. The solution strategy is an extension of the αBB algorithm for twice-differentiable NLPs [4,5,7]. It is based on the generation of two converging sequences of upper and lower bounds on the global optimum solution. A rigorous underestimation and convexification strategy for functions with continuous second order derivatives allows the construction of a lower bounding MINLP problem with convex functions in the continuous variables. If no mixed-bilinear terms are present ($A_i = 0$, $\forall i$), the resulting MINLP can be solved to global optimality using the *outer approximation algorithm* (OA), [8]. Otherwise, the *generalized Benders decomposition* (GBD) can be used, [10], or the Glover transformations [11] can be applied to remove these bilinearities and permit the use of the OA algorithm. This convex MINLP provides a valid lower bound on the original MINLP. An upper bound on the problem can be obtained by applying the OA algorithm or the GBD to find a local solution. This bound generation strategy is incorporated within a branch and bound scheme: a lower and upper bound on the global solution are first obtained for the entire solution space. Subsequently, the domain is subdivided by branching on a binary or a continuous variable, thus creating new nodes for which upper and lower bounds can be computed. At each iteration, the node with the lowest lower bound is selected for branching. If the lower bounding MINLP for a node is infeasible or if its lower bound is greater than the best upper bound, this node is fathomed. The algorithm is terminated when the best lower and upper bound are within a pre-specified tolerance of each other.

Before presenting the algorithmic procedure, an overview of the underestimation and convexification strategy is given, and some of the options available within the algorithm are discussed.

In order to transform the MINLP problem of the form just described into a convex problem which can be solved to global optimality with the OA or GBD algorithm, the functions $f(x)$, $h(x)$ and $g(x)$ must be convexified. The underestimation and convexification strategy used in the αBB algorithm has previously been described in detail [3,4,5]. Its main features are exposed here.

In order to construct as tight an underestimator as possible, the nonconvex functions are decomposed into a sum of convex, bilinear, univariate concave and general nonconvex terms. The overall function underestimator can then be built by summing up the convex underestimators for all terms, according to their type. In particular, a new variable is introduced to replace each bilinear term, and is bounded by its convex envelope. The univariate concave terms are linearized. For each nonconvex term $nt(x)$ with Hessian matrix $H_{nt}(x)$, a convex underestimator $L(x)$ is defined as

$$L(x) = nt(x) - \sum_i \alpha_i (x_i^U - x_i)(x_i - x_i^L), \qquad (1)$$

where x_i^U and x_i^L are the upper and lower bounds on variable x_i, respectively, and the α parameters are nonnegative scalars such that $H_{nt}(x) + 2\,\mathrm{diag}(\alpha_i)$ is positive semidefinite over the domain $[x^L, x^U]$. The rigorous computation of the α parameters using interval Hessian matrices is described in [3,4,5].

The underestimators are updated at each node of the branch and bound tree as their quality strongly depends on the bounds on the variables. An unusual feature of the SMIN-αBB algorithm is the strategy used to select branching variables. It follows a hybrid approach where branching may occur both on the integer and the

continuous variables in order to fully exploit the structure of the problem being solved. After the node with the lowest lower bound has been identified for branching, the type of branching variable must be determined according to one of the following two criteria:

1) Branch on the binary variables first.
2) Solve a continuous relaxation of the nonconvex MINLP locally. Branch on a binary variable with a low degree of fractionality at the solution. If there is no such variable, branch on a continuous variable.

The first criterion results in the creation of an integer tree for the first q levels of the branch and bound tree, where q is the number of binary variables. At the lowest level of this integer tree, each node corresponds to a nonconvex NLP and the lower and upper bounding problems at subsequent levels of the tree are NLP problems. The efficiency of this strategy lies in the minimization of the number of MINLPs that need to be solved. The combinatorial nature of the problem and its nonconvexities are handled sequentially. If branching occurs on a binary variable, the selection of that variable can be done randomly or by solving a relaxation of the nonconvex MINLP and choosing the most fractional variable at the solution.

The second criterion selects a binary variable for branching only if it appears that the two newly created nodes will have significantly different lower bounds.Thus, if a variable is close to integrality at the solution of the relaxed problem, forcing it to take on a fixed value may lead to the infeasibility of one of the nodes or the generation of a high value for a lower bound, and therefore the fathoming of a branch of the tree. If no binary variable is close to integrality, a continuous variable is selected for branching.

A number of rules have been developed for the selection of a continuous branching variable. Their aim is to determine which variable is responsible for the largest separation distances between the convex underestimating functions and the original nonconvex functions. These efficient rules are exposed in [2]. Variable bound updates performed before the generation of the convex MINLP have been found to greatly enhance the speed of convergence of the αBB algorithm for continuous problems [2]. For continuous variables, the variable bounds are updated by minimizing or maximizing the chosen variable subject to the convexified constraints being satisfied. In spite of its computational cost, this procedure often leads to significant improvements in the quality of the underestimators and hence a noticeable reduction in the number of iterations required.

In addition to the update of continuous variable bounds, the SMIN-αBB algorithm also relies on binary variable bound updates. Through simple computations, an entire branch of the branch and bound tree may be eliminated when a binary variable is found to be restricted to 0 or 1. The bound update procedure for a given binary variable is as follows:

1) Set the variable to be updated to one of its bounds $y = y_B$.
2) Perform interval evaluations of all the constraints in the nonconvex MINLP, using the bounds on the solution space for the current node.
3) If any of the constraints are found infeasible, fix the variable to $y = 1 - y_B$.
4) If both bounds have been tested, repeat this procedure for the next variable to be updated. Otherwise, try the second bound.

In [1] *GMIN*, which operates within a classical branch and bound framework, was proposed. The main difference with similar branch and bound algorithms [12,17] is its ability to identify the global optimum solution of a much larger class of problems of the form

$$\begin{cases} \min\limits_{x,y} & f(x, y) \\ \text{s.t.} & h(x, y) = 0 \\ & g(x, y) \leq 0 \\ & x \in X \subset \mathbb{R}^n \\ & y \in N^q, \end{cases}$$

where N is the set of nonnegative integers and the only condition imposed on the functions $f(x, y)$, $g(x, y)$ and $h(x, y)$ is that their continuous relaxations possess continuous second order derivatives. This increased applicability results from the use of the αBB global optimization algorithm for continuous twice-differentiable NLPs [4,5,7].

At each node of the branch and bound tree, the nonconvex MINLP is relaxed to give a nonconvex NLP, which is then solved with the αBB algorithm. This allows the identification of rigorously valid lower bounds and therefore ensures convergence to the global optimum. In general, it is not necessary to let the αBB al-

gorithm run to completion as each one of its iterations generates a lower bound on global solution of the NLP being solved. A strategy of early termination leads to a reduction in the computational requirements of each node of the binary branch and bound tree and faster overall convergence.

The GMIN-αBB algorithm selects the node with the lowest lower bound for branching at every iteration. The branching variable selection strategy combines several approaches: branching priorities can be specified for some of the integer variables. When no variable has a priority greater than all other variables, the solution of the continuous relaxation is used to identify either the most fractional variable or the least fractional variable for branching.

Other strategies have been implemented to ensure a satisfactory convergence rate. In particular, bound updates on the integer variables can be performed at each level of the branch and bound tree. These can be carried out through the use of interval analysis. An integer variable, y^*, is fixed at its lower (or upper) bound and the range of the constraints is evaluated with interval arithmetic, using the bounds on all other variables. If the range of any constraint is such that this constraint is violated, the lower (or upper) bound on variable y^* can be increased (or decreased) by one. Another strategy for bound updates is to relax the integer variables, to convexify and underestimate the nonconvex constraints and to minimize (or maximize) a variable y^* in this convexified feasible region. The resulting lower (or upper) bound on relaxed variable y^* can then be rounded up (or down) to the nearest integer to provide an updated bound for y^*.

See also

- ▶ αBB Algorithm
- ▶ Chemical Process Planning
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Disjunctive Programming
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ Global Optimization in Batch Design Under Uncertainty
- ▶ Global Optimization in Generalized Geometric Programming
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization in Phase and Chemical Reaction Equilibrium
- ▶ Interval Global Optimization
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming
- ▶ Reformulation-linearization Methods for Global Optimization
- ▶ Smooth Nonlinear Nonconvex Optimization

References

1. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP probelms in process synthesis and design. Comput Chem Eng 21:S445–S450
2. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB for twice-differentiable NLP's – II. Implementation and computational results. Comput Chem Eng 22:1137–1158
3. Adjiman CS, Androulakis IP, Maranas CD, Floudas CA (1996) A global optimization method, αBB, for process design. Comput Chem Eng 20:S419–S424
4. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB for twice-differentiable NLP's – I. Theoretical Advances. Comput Chem Eng 22:1159–1179
5. Adjiman CS, Floudas CA (1996) Rigorous convex underestimators for general twice-differentiable problems. J Global Optim 9:23–40
6. Adjiman CS, Schweiger CA, Floudas CA (1998) Mixed-integer nonlinear optimization in process synthesis. In: Du D-Z and Pardalos PM (eds) Handbook Combinatorial Optim. Kluwer, Dordrecht

7. Androulakis IP, Maranas CD, Floudas CA (1995) αBB, a global optimization method for general constrained nonconvex problems. J Global Optim 7:337–363
8. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
9. Floudas CA (1995) Nonlinear and mixed-integer optimization: Fundamentals and applications. Oxford Univ. Press, Oxford
10. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
11. Glover F (1975) Improved linear integer programming formulations of nonlinear integer problems. Managem Sci 22:445–452
12. Gupta OK, Ravindran R (1985) Branch and bound experiments in convex nonlinear integer programming. Managem Sci 31:1533–1546
13. Lawler EL, Wood DE (1966) Branching and bound methods: A survey. Oper Res 14:699–719
14. McCormick GP (1976) Computatbility of global solutions to factorable nonconvex programs; Part I – convex underestimating problems. Math Program 10:147–175
15. Murty KG, Kabadi SN (1987) Some NP-complete problems in quadratic and nonlinear prgramming. Math Program 39:117–123
16. Neumhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
17. Ostrovsky GM, Mikhailov GW (1990) Discrete optimization of chemical processes. Comput Chem Eng 14:111–124
18. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput Chem Eng 16:937–947
19. Ryoo HS, Sahinidis NV (1995) Global optimization of nonconvex NLPs and MINLPs with applications in process design. Comput Chem Eng 19:551–566
20. Smith EMB, Pantelides CC (1997) Global optimization of nonconvex MINLPs. Comput Chem Eng 21:S333–S338

MINLP: Branch and Bound Methods

Brian Borchers
Department of Mathematics, New Mexico Tech., Socorro, USA

MSC2000: 90C11

Article Outline

Keywords
See also
References

Keywords

Mixed integer programming; Branch and bound; MINLP

A general *mixed integer nonlinear programming problem* (*MINLP*) can be written as

$$\text{(MINLP)} \quad \begin{cases} \min & f(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}) = 0 \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq 0 \\ & \mathbf{x} \in \mathbb{R}^n \\ & \mathbf{y} \in \mathbb{Z}^m. \end{cases}$$

Here $\mathbf{x}$ is a vector of n continuous variables and $\mathbf{y}$ is a vector of m *integer variables*. In many cases, the integer variables $\mathbf{y}$ are restricted to the values 0 and 1. Such variables are called *binary variables*. The function f is a scalar valued objective function, while the vector functions $\mathbf{h}$ and $\mathbf{g}$ express linear or nonlinear constraints. Problems of this form have a wide variety of applications, in areas as diverse as IR spectroscopy [6], finance [3], chemical process synthesis [9], topological design of transportation networks [12], and marketing [10].

The earliest work on *branch and bound* algorithms for mixed integer linear programming dates back to the early 1960s [7,13,15]. Although the possibility of applying branch and bound methods to mixed integer nonlinear programming problems was apparent from the beginning, actual work on such problems did not begin until later. Early papers on branch and bound algorithms for mixed integer nonlinear programming include [11,14].

A branch and bound algorithm for solving (MINLP) requires the following data structures. The algorithm maintains a list L of unsolved subproblems. The algorithm also maintains a record of the best integer solution that has been found. This solution, ($\mathbf{x}^*$, $\mathbf{y}^*$), is called the incumbent solution. The incumbent solution provides an upper bound, *ub*, on the objective value of an optimal solution to (MINLP).

The basic branch and bound procedure is as follows.

1) *Initialize*: Create the list L with (MINLP) as the initial subproblem. If a good integer solution is known,

then initialize $\mathbf{x}^*$, $\mathbf{y}^*$, and *ub* to this solution. If there is no incumbent solution, then initialize *ub* to $+\infty$.

2) *Select*: Select an unsolved subproblem, *S*, from the list *L*. If *L* is empty, then stop: If there is an incumbent solution, then that solution is optimal; If there is no incumbent solution, then (MINLP) is infeasible.
3) *Solve*: Relax the integrality constraints in *S* and solve the resulting nonlinear programming relaxation. Obtain a solution $\widehat{\mathbf{x}}$, $\widehat{\mathbf{y}}$, and a lower bound, *lb*, on the optimal value of the subproblem.
4) *Fathom*: If the relaxed subproblem was infeasible, then *S* will clearly not yield a better solution to (MINLP) than the incumbent solution. Similarly, if $lb \geq ub$, then the current subproblem cannot yield a better solution to (MINLP) than the incumbent solution. Remove *S* from *L*, and return to step 2.
5) *Integer Solution*: If $\widehat{\mathbf{y}}$ is integer, then a new incumbent integer solution has been obtained. Update $\mathbf{x}^*$, $\mathbf{y}^*$, and *ub*. Remove *S* from *L* and return to step 2.
6) *Branch*: At least one of the integer variables y_k takes on a fractional value in the solution to the current subproblem. Create a new subproblem, S_1 by adding the constraint

$$y_k \leq \lfloor \widehat{y}_k \rfloor.$$

Create a second new subproblem, S_2 by adding the constraint

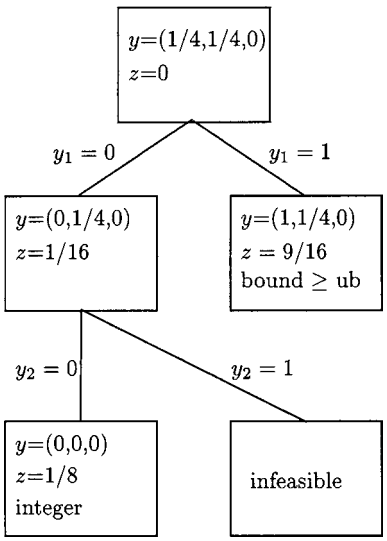
$$y_k \geq \lceil \widehat{y}_k \rceil.$$

Remove *S* from *L*, add S_1 and S_2 to *L*, and return to step 2.

The following example demonstrates how the branch and bound algorithm solves a simple (MINLP):

$$\begin{cases} \min & (y_1 - \frac{1}{4})^2 + (y_2 - \frac{1}{4})^2 + y_3^2 \\ & -2y_1 + 2y_2 \leq 1 \\ & \mathbf{y} \text{ binary.} \end{cases}$$

The optimal solution to the initial nonlinear programming relaxation is $y = (1/4, 1/4, 0)$, with an objective value of $z = 0$. Both y_1 and y_2 take on fractional values in this solution, so it is necessary to select a branching variable. The algorithm arbitrarily selects y_1 as the branching variable, and creates two new subproblems in which y_1 is fixed at 0 or 1. In the subproblem with y_1 fixed at 0, the optimal solution is $y = (0, 1/4, 0)$, with $z = 1/16$. Since the optimal value of y_2 is fractional, the algorithm again creates two new subproblems, with y_2 fixed at 0 and 1. The optimal solution to the subproblem with $y_1 = 0$ and $y_2 = 0$ is $y = (0, 0, 0)$, with $z = 1/8$. This establishes an incumbent integer solution. The subproblem with $y_1 = 0$ and $y_2 = 1$ is infeasible and can be eliminated from consideration. The subproblem with $y_1 = 1$ has an optimal solution with $y = (1, 1/4, 0)$ and objective value $z = 9/16$. Since 9/16 is larger than the objective value of the incumbent solution, this subproblem can be eliminated from consideration. Thus the optimal solution to the example problem is $y^* = (0, 0, 0)$ with objective value $z^* = 1/8$.



MINLP: Branch and Bound Methods, Figure 1
Branch and bound tree for a sample problem

Since each subproblem *S* creates at most two new subproblems, the set of subproblems considered by the branch and bound algorithm can be represented as a binary tree. The above figure shows the branch and bound tree for the example problem.

There are a number of important issues in the implementation of a branch and bound algorithm for (MINLP).

The first important issue is how to solve the nonlinear programming relaxations of the subproblems in step 3. If the objective function *f* and the constraint

functions **g** are convex, while the constraint functions **h** are linear, then the nonlinear programming subproblems in step 3 are convex and thus relatively easy to solve. A variety of methods have been used to solve these subproblems including generalized reduced gradient (GRG) methods [11], sequential quadratic programming (SQP) [4], active set methods for quadratic programming [8], and interior point methods [16].

However, if the nonlinear programming subproblems are nonconvex, then it can be extremely difficult to solve the nonlinear programming relaxation of S or even obtain a lower bound on the optimal objective function value. For some specialized classes of nonconvex optimization problems, including *indefinite quadratic programming*, *bilinear programming*, and *fractional linear programming*, convex functions which underestimate the nonconvex objective function are known. These *convex underestimators* are widely used in branch and bound algorithms for nonconvex nonlinear programming problems. Branch and bound techniques for nonconvex continuous optimization problems can also been used within a branch and bound algorithm for nonconvex mixed integer nonlinear programming problems. For instance, the *BARON* system uses this approach to solve a variety of nonconvex mixed integer nonlinear programming problems [17,18]. This approach is also used in the GMIN-αBB algorithm to solve nonconvex $0-1$ mixed integer nonlinear programming problems with twice differentiable objective and constraint functions [1].

The choice of the next subproblem to be solved in step 2 can have a significant influence on the performance of the branch and bound algorithm. In mixed integer linear programming, a variety of heuristics are employed to select the next subproblem [2]. One popular heuristic used in branch and bound algorithms for MILP is the *'best bound rule'*, in which the subproblem with the smallest lower bound is selected. The best bound rule is widely used within branch and bound algorithms for (MINLP) [4,11,18]

In step 6, there may be a choice of several variables with fractional values to be the branching variable. A simple approach is to select the variable whose value $\widehat{y}_k$ is furthest from being an integer [4,11]. In mixed integer linear programming, estimates of the increase in the objective function that will result from forcing a variable to an integer value are often made. These estimates, called *'pseudocosts'* or *'penalties'*, are used to select the branching variable. Penalties have also been used in branch and bound algorithms for mixed integer nonlinear programming problems [11,18].

The performance of the branch and bound algorithm can be improved by computing lower bounds on the optimal value of a subproblem without actually solving the subproblem. In [8], lower bounds on the optimal objective value of a subproblem are derived from an optimal dual solution to the subproblem's parent problem. If this lower bound is larger than the objective value of the incumbent solution, then the subproblem can be eliminated from consideration. In [4], Lagrangian duality is used to compute lower bounds during the solution of a subproblem. When the lower bound exceeds the value of the incumbent solution, the current subproblem can be discarded.

Another way to improve the performance of a branch and bound algorithm for (MINLP) is to tighten the formulation of the nonlinear programming subproblems before solving them. In the BARON package, dual information from the solution to a nonlinear programming subproblem is used to restrict the ranges of variables and constraints in the children of the subproblem [17,18].

In *branch and cut* approaches, constraints called cutting planes are added to the nonlinear programming subproblems [3,19]. These additional constraints are selected so that they reduce the size of the feasible region of nonlinear programming subproblems without eliminating any integer solutions from consideration. This tightens the formulations of the subproblems and thus increases the probability that a subproblem can be fathomed by bound. Furthermore, the use of cutting planes can make it more likely that an integer solution will be obtained early in the branch and bound process. A variety of cutting planes developed for use in branch and cut algorithms for integer linear programming have been adapted for use in branch and cut algorithms for nonlinear integer programming. These include *mixed integer rounding cuts* [3], *knapsack cuts* [3], *intersection cuts* [3], and *lift-and-project cuts* [19].

To date, little work has been done to compare the performance of branch and bound methods for (MINLP) with other approaches such as *outer approximation* and *generalized Benders decomposition*. B. Borchers and J.E. Mitchell (1997) compared an ex-

perimental branch and bound code with a commercially available outer approximation code on a number of test problems [5]. This study found that the branch and bound code and outer approximation code were roughly comparable in speed and robustness. R. Fletcher and S. Leyffer (1998) compared the performance of their branch and bound code for mixed integer convex quadratic programming problems with their implementations of outer approximation, generalized Benders decomposition, and an algorithm that combines branch and bound and outer approximation approaches [8]. Fletcher and Leyffer found that their branch and bound solver was consistently faster than the other codes by about an order of magnitude.

See also

- Chemical Process Planning
- Disjunctive Programming
- Extended Cutting Plane Algorithm
- Generalized Benders Decomposition
- Generalized Outer Approximation
- MINLP: Application in Facility Location-allocation
- MINLP: Applications in Blending and Pooling Problems
- MINLP: Applications in the Interaction of Design and Control
- MINLP: Branch and Bound Global Optimization Algorithm
- MINLP: Design and Scheduling of Batch Processes
- MINLP: Generalized Cross Decomposition
- MINLP: Global Optimization with αBB
- MINLP: Heat Exchanger Network Synthesis
- MINLP: Logic-based Methods
- MINLP: Outer Approximation Algorithm
- MINLP: Reactive Distillation Column Synthesis
- Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- Mixed Integer Nonlinear Programming
- Reformulation-linearization Methods for Global Optimization

References

1. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP problems in process synthesis and design. Comput Chem Eng 21(1001):S445–S450
2. Beale EML (1977) Integer programming. In: Jacobs D (ed) The State of the Art in Numerical Analysis. Acad. Press, New York p 409–448
3. Bienstock D (1996) Computational study of a family of mixed-integer quadratic programming problems. Math Program 74(2):121–140
4. Borchers B, Mitchell JE (1994) An improved branch and bound algorithm for mixed integer nonlinear programs. Comput Oper Res 21(4):359–367
5. Borchers B, Mitchell JE (1997) A computational comparison of branch and bound and outer approximation algorithms for 0–1 mixed integer nonlinear programs. Comput Oper Res 24(8):699–701
6. Brink A, Westerlund T (1995) The joint problem of model structure determination and parameter estimation in quantitative IR spectroscopy. Chemometrics and Intelligent Laboratory Systems 29:29–36
7. Dakin RJ (1965) A tree–search algorithm for mixed integer programming problems. Comput J 8:250–255
8. Fletcher R, Leyffer S (1998) Numerical experience with lower bounds for MIQP branch-and-bound. SIAM J Optim 8:604–616
9. Floudas CA (1995) Nonlinear and mixed-integer optimization: fundamentals and applications. Oxford Univ. Press, Oxford
10. Gavish B, Horsky D, Srikanth K (1983) An approach to the optimal positioning of a new product. Managem Sci 29(11):1277–1297
11. Gupta OK, Ravindran A (1985) Branch and bound experiments in convex nonlinear integer programming. Managem Sci 31(12):1533–1546
12. Hoang Hai Hoc (1982) Topological optimization of networks: A nonlinear mixed integer model employing generalized Benders decomposition. IEEE Trans Autom Control 27:164–169
13. Land A, Doig A (1960) An automatic method of solving discrete programming problems. Econometrika 28(3):497–520
14. Laughhunn DJ (1970) Quadratic binary programming with applications to capital-budgeting problems. Oper Res 18(3):454–461
15. Lawler EL, Wood DE (1966) Branch and bound methods: A survey. Oper Res 14(4):699–719
16. Lee EK, Mitchell JE (1997) Computational experience of an interior point algorithm in a parallel branch-and-cut framework. In: Proc. Eighth SIAM Conf. Parallel Processing for Sci. Computing, Minneapolis, March 1997. www.siam.org/catalog/mcc07/heath97.htm
17. Ryoo HS, Sahinidis NV (1996) A branch–and–reduce approach to global optimization. J Global Optim 8(2):107–139
18. Sahinidis NV (1996) BARON: A general purpose global optimization software package. J Global Optim 8(2):201–205

19. Stubbs RA, Mehrotra S (1999) A branch-and-cut method for 0–1 mixed convex programming. Math Program 80:515–532

MINLP: Design and Scheduling of Batch Processes

Christodoulos A. Floudas[1], S. T. Harding[2], Marianthi Ierapetritou[3]
[1] Department of Chemical Engineering, Princeton University, Princeton, USA
[2] Advanced Process Combinatorics, Inc., Purdue Technology Center, West Lafayette, USA
[3] Department Chemical and Biochemical Engineering, Rutgers University, Piscataway, USA

MSC2000: 90C26

Article Outline

Keywords
Continuous-Time Formulations
Problem Formulation
Discrete-Time Formulations
See also
References

Keywords

Batch process; Design; Scheduling; Continuous and discrete time models

The design of batch processes has been a major area of research for the past several decades. In conjunction with the design of batch plants, many different approaches have been proposed for the determination of an optimal schedule for the plant. It has been recognized for some time that in order to increase the efficiency of batch processes, the two tasks of design and scheduling should be considered simultaneously.

The problem is to design a batch process consisting of M processing steps, in which N products are made, where all materials follow the same path through the process. This is commonly known as a *multiproduct batch plant*, or a *flow-shop*.

There are two predominant methods for formulating the batch process design and scheduling problem. The first is a *continuous-time formulation* in which the scheduling information is incorporated through a *planning horizon* constraint. This problem can be formulated as a NLP or MINLP depending on whether the number of parallel units is fixed or variable. The solution of this problem does not give the actual schedule, but does guarantee that a feasible schedule exists. A separate problem, typically a MILP, must be solved to find the actual schedule.

The second method for formulating the batch process design and scheduling problem is based on a *state-task-network* (STN) representation. In this approach, the planning horizon is discretized into time steps. Each task must be assigned to both a unit and a time slot. The formulation results in a large MINLP whose solution provides both the plant design and the actual schedule.

Continuous-Time Formulations

The early work of [10] was based on the *single product campaign* (SPC) scheduling policy. In a single product campaign, all batches of one product are processed one after the other, followed by all of the batches of the next product, and so on.

In this approach, the scheduling information is incorporated by way of a planning horizon constraint. This constraint requires that all products must be completed before the planning horizon, H, is reached. In a single product campaign, the time between batches of product i is based on the maximum processing time over all of the stages,

$$t_{Li} = \max_j (t_{ij}),$$

where t_{Li} is the 'limiting' time for product i. The planning horizon constraint can be written as the sum over all of the products of the limiting time multiplied by the number of batches of each product

$$\sum_i \frac{Q_i}{B_i} T_{Li} \leq H,$$

where Q_i is the total production of i and B_i is the batch size for i. Because Q_i and B_i are variables, this results in a NLP.

In [4] the authors formulated the batch process design and scheduling problem as a MINLP. Their model was based on the SPC model of [10]. In this problem, more than one piece of equipment per stage is available for use in parallel. Rather than solve the MINLP rigorously, they relaxed the number of units per stage to be continuous and solved the resulting NLP. [5] formulated the MINLP using binary 0-1 variables and solved it with an outer approximation method. In addition to the combinatorial nature of the problem due to integer variables, the solution of the problem is complicated by the nonconvex form of the planning horizon constraint.

[2] developed extensions of the SPC formulation to allow more efficient utilization of the batch process equipment. They considered two *mixed-product campaign* (MPC) scheduling policies,

i) the *unlimited intermediate storage* (UIS) policy; and
ii) the *zero-wait* (ZW) policy.

As its name implies, a mixed product campaign allows batches of different products to be processed sequentially. For example, a SPC schedule for three batches each of two products A and B would be, AAABBB, while a MPC schedule could be ABABAB. In the zero-wait policy, when a product has completed processing in one stage, it must immediately begin processing in the next stage. Conversely, the UIS policy allows a product to be stored for a period of time before beginning the next processing step. [7] showed that for the case of zero cleanup times, the UIS policy is the most efficient mixed-product campaign policy, while the ZW policy is the most conservative. [2] incorporated the new scheduling policies into the batch process design problem by considering the characteristic cycle time for each policy. The cycle time becomes the basis upon which the planning horizon constraint is imposed.

[3] used the batch design formulation with mixed-product campaign schedules to formulate the batch synthesis, design and scheduling problem. In this formulation the number of stages, M, in the batch process is not fixed. Instead, each product is required to undergo the same sequence, T, of processing tasks. Units that each can perform one of the tasks are given, and in addition, 'superunits' are postulated that can combine two or more tasks. The problem is to assign tasks to units, size the units, and determine the number of parallel units in the batch process.

Problem Formulation

1) Binary variables

$$YEX_j = \begin{cases} 1 & \text{if unit } j \text{ exists} \\ 0 & \text{otherwise,} \end{cases}$$

$$YC_{cj} = \begin{cases} 1 & \text{if unit } j \text{ contains} \\ & \quad c \text{ parallel units} \\ 0 & \text{otherwise,} \end{cases}$$

$$Y_{tj} = \begin{cases} 1 & \text{if task } t \text{ is assigned} \\ & \quad \text{to unit } j \\ 0 & \text{otherwise,} \end{cases}$$

$$YF_{tj} = \begin{cases} 1 & \text{if } t \text{ is the first task} \\ & \quad \text{processed in unit } j \\ 0 & \text{otherwise.} \end{cases}$$

2) Design constraints
   - Task volume requirement, V_t^T, depends on batch size, B_i, of each product and size factor, S_{it}, for each product in each task.

$$V_t^T \geq B_i S_{it}.$$

   - The volume of a processing unit j must be large enough to accomodate task t if task t is assigned to unit j, $(Y_{tj} = 1)$.

$$V_j \geq V_t^T - V_j^U(1 - Y_{tj}).$$

   - The processing time, pt_{ij}, for each product in each unit is given by the corresponding time factor, t_{it}, for each product in task t if task t is assigned to unit j, $(Y_{tj} = 1)$.

$$pt_{ij} \geq \sum_t t_{it} Y_{tj}.$$

   - The number of batches, n_i, multiplied by the batch size must satisfy the production requirement, Q_i, for each product.

$$n_i B_i \geq Q_i.$$

3) Parallel equipment constraints
   - The number of parallel units in each stage j is determined by the binary variable YC_{cj} multiplied by the number c,

$$N_j = \sum_c c \cdot YC_{cj}.$$

4) Scheduling constraint
   - For the UIS policy with zero cleanup times, the planning horizon constraint derived by [2] is used,

$$\sum_i n_i pt_{ij} \le H \cdot N_j.$$

5) Logical constraints
   - If a stage j exists, then at least one processing task must be assigned to it,

$$\sum_t Y_{tj} \ge YEX_j.$$

   - If a stage j does not exist, there can be no tasks assigned to it,

$$Y_{tj} \le YEX_j.$$

   - If a stage j exists, then one of the tasks assigned to it must be the first task assigned to stage j,

$$\sum_t YF_{tj} = YEX_j.$$

   - There cannot be more than one first task assigned to each stage,

$$\sum_t YF_{tj} \le 1.$$

   - A task can be the first task assigned to a stage only if the task is among those assigned to the stage,

$$YF_{tj} \le Y_{tj}.$$

   - No tasks that occur before the first task assigned to stage j can be among those assigned to the stage,

$$Y_{t'j} \le 1 - YF_{tj} \quad \text{for } t' < t.$$

   - If multiple tasks are assigned to a unit, they must be consecutive tasks,

$$Y_{tj} \le YF_{tj} + Y_{t-1,j}.$$

   - One and only one binary variable that determines the number of parallel units in stage j must be active,

$$\sum_c YC_{cj} = 1.$$

6) Objective function
   - The objective is to minimize the cost of the plant. [3] used a fixed-charge cost for each unit, γ_j, plus a nonlinear cost function on the size of the unit,

$$\text{Cost} = \sum_j N_j \left[\gamma_j + \alpha_j V_j^{\beta_j}\right].$$

This formulation is a MINLP where all binary variables participate linearly and separably. However, it is a nonconvex problem due to the cost function, and the *bilinear* terms in the batch size constraints and the planning horizon constraints. [3] used the *outer approximation* method implemented in DICOPT ([11]) to solve a number of example problems. Due to the nonconvexities in the formulation, there is no guarantee of global optimality with the outer approximation method, but they report good results for the examples presented in the paper.

Two examples are briefly discussed to illustrate the proposed approach for multiproduct batch plants with a variety of scheduling policies. The first example consists of three products with four processing tasks and five potential units and superunits. The MINLP formulation with the SPC policy contains 33 binary variables and 54 continuous variables. With the ZW policy, the number of binary variables drops to 8, with 98 continuous variables. For the UIS policy, the formulation has 33 binary variables with 51 continuous variables.

The second example is larger and contains 6 products with 7 potential units and superunits. The SPC policy formulation contains 46 binary variables and 101 continuous variables. The MINLP formulation for the ZW policy has 11 binary and 374 continuous variables. The UIS policy formulation has 46 binary and 95 continuous variables. In all cases the examples were solved in less than 50 minutes using GAMS/DICOPT ++ on Microvax II.

Discrete-Time Formulations

A.P.F.D. Barbosa-Póvoa and S. Macchietto, [1], proposed a MILP formulation to address the problem of optimal batch design by simultaneously considering optimizing production schedule. They based their formulation on

a) an extended state-task-network (mSTN) representation of the batch plant; and

b) the discrete time representation using *uniform time discretization*.

In the STN representation, proposed in [6], all the materials are represented as states processed through a set of processing steps ('tasks'). In order to incorporate connectivity constraints the extended state-task-network (mSTN) is proposed involving the alternative design configurations considering all permitted equipment and connections allocations. Single campaign is assumed with a cyclic schedule of cycle time T repeated over a planning horizon H. A cycle represents a sequence of operations involving the production of all products and the utilization of all resources. The operational characteristics such as the allocation of equipments to tasks, batch sizes, task timings, transport of material and storage profiles are identical in each cycle. The mathematical formulation they proposed involves:

- allocation constraints for the assignment of the tasks to the units
- capacity constraints expressing the limiting equipment capability
- connectivity constraints for determining the connection of different units
- dedicated storage constraints
- mass balances
- production requirement constraints
- an objective function, which is chosen to be either the minimization of the capital cost or the maximization of plant profit.

The main variables of the formulation are:

a) binary structural variables representing the existence of an equipment;
b) binary allocation variables for the assignment of a task to a unit at the beginning of a time period;
c) continuous variables representing the capacity of a unit;
d) continuous variables corresponding to the batch size of a task to a unit at each time period;
e) amount of material delivered and received at each time period;
f) the amount of material transfered at each time period; and
g) the amount of material stored at each time period.

The proposed formulation correspond to a mixed integer linear programming (MILP) problem since they used linear cost functions to express the capital cost of equipments and time discretization to represent time. Three examples were solved illustrating:

a) the effect of limited connectivity and connection cost in the optimal design;
b) the advantages of considering simultaneously the plant design and plant connectivity rather than optimizing first the equipment sizes and then optimizing plant connectivity.

In later work, Barbosa-Póvoa and C.C. Pantelides, [1], proposed a new mathematical formulation for the optimization of batch plant design considering detailed operation characteristics (i. e., short term scheduling). This formulation also considers a uniform time discretization, the only difference lies in the plant representation. The resource-state-task (RTN) plant representation, [9], was used which corresponds to a more general and uniform description of all available production resources. However, the new formulation shares the main characteristics of the previous presented one with the same basic variables, and constraints.

Both formulations share the limitations of the discrete time formulations, which are that:

i) they correspond to an approximation of the time horizon; and
ii) they result in an unnecessary increase of the number of binary variables in particular, and in the overall size of the mathematical model.

A continuous-time formulation was proposed in [12], based on the STN representation and the scheduling formulation proposed in [13]. It gives rise to a mixed integer nonlinear programming problem which is solved using a stochastic MINLP optimizer based on an *evolutionary algorithm* (EA) with *simulated annealing* (SA) presented in [12]. The method is based on a guided stochastic generation of alternative vectors of decision variables, which explore promising areas of the search space through selection, crossover, and mutation operations applied to individuals in a population of solution candidates. It can be used to deal with nonconvex, nondifferentiable functions although it has no guarantee of convergence to even a local optimal solution. The proposed formulation involves the following basic variables:

- Main design variables representing the discrete decisions of selecting a unit (j), E_j, or a storage (s), E_s, or continuous decisions corresponding to the capacity of unit storage or utility, V_j, V_s, and U_u, respectively.

- Main operation variables corresponding to the discrete decision of allocation of task (i) in unit (j) at time T_l, W_{ijl}, and the decision of assigning task (i) in unit (j) between starting time T_l and end time $T_{l'}$, and continuous variables, the time of event (l), T_l, the batch size, the processing time and utility requirement of task (i) allocated to unit (j) starting at T_l, B_{ijl}, τ_{ijl}, U^u_{ijl}, respectively,

Based on these variables the proposed formulation involves:

1) Processing task models:

$$U^u_{ijl} = \alpha^u_{ijl} + \beta^u_{ij} B^{\gamma^u_{ij}}_{ijl},$$

expressing the consumption-generation of utilities as a function of batch size;

$$\tau_{ijl} = \alpha_{ijl} + \beta_{ij} B^{\gamma_{ij}}_{ijl} + \sum_u \mu^u_{ij} U^u_{ijl} + \sum_\alpha \mu^\alpha_{ij} A^\alpha_{jl},$$

expressing the dependence of processing time, τ_{ijl}, of batch size, B_{ijl}, utilities, U^u_{ijl}, and unit availabilities, A^α_{jl}.

2) Batch size constraints:

$$\phi^{\min}_{ij} V_j W_{ijl} \le B_{ijl} \le \phi^{\max}_{ij} V_j W_{ijl}$$

imposing the maximum and minimum capability of unit (j) when task (i) is performed.

3) Timing constraints:

$$W_{ijl}(\tau_{ijl} + T_l) = \sum_{l'>l} X_{ijll'} T_{l'},$$

which establish the relationship between processing time, τ_{ijl}, and time of event (l), T_l.

$$0 \le T_1 \le T_2 \le \cdots \le T_{l^{\max}} \le H,$$

expressing the monotonic increase in event times.

4) Allocation constraints:

$$0 \le \sum_{i\in I_j} \sum_{l''\le l'} W_{ijl''} - \sum_{i\in I_j} \sum_{l<l''} \sum_{l''\le l'} X_{ijll''} \le E_j,$$

$$\sum_{i\in I_j} \sum_{l''\le l^{\max}} W_{ijl''} = \sum_{i\in I_j} \sum_{l<l''} \sum_{l''\le l^{\max}} X_{ijll''},$$

$$W_{ijl} = \sum_{l'>l} X_{ijll'},$$

expressing the relationship between W_{ijl} and $X_{ijll'}$ operation variables, [13].

5) Material balances written for state s at event time T_l:

$$C_{sl'} = C_{sl'-1} + \sum_{i\in I_s} \sum_{j\in J_i} \rho^{in}_{sij} \sum_{l<l'} B_{ijl} X_{ijll'} - \sum_{i\in I_s} \sum_{j\in J_i} \rho^{out}_{sij} B_{ijl'},$$

$$0 \le C_{sl'} \le V_{s0} + V_s.$$

6) Utility constraints written for utility (u) at event time T_l:

$$U_{ul'} = U_{ul'-1} + \sum_{i\in I_u} \sum_{j\in J_i} \sum_{l<l'} U^u_{ijl} X_{ijll'} - \sum_{i\in I_u} \sum_{j\in J_i} U^u_{ijl'} W_{ijl'},$$

$$0 \le U_{ul'} \le U_u,$$

$$U_{uc} = \sum_{i\in I_u} \sum_{j\in J_i} \sum_l U^u_{ijl} T_{ijl} W_{ijl}.$$

7) Availability constraints written for unit (j) at event time T_l:

$$A^\alpha_{jl'+1} = \sum_{i\in I_j} A^\alpha_{jl'} \alpha^\alpha_{ij} W_{ijl'} - \sum_{i\in I_j} \beta^\alpha_{ij} W_{ijl'},$$

$$A^\alpha_{jl'} \le \sum_{i\in I_j} \gamma^\alpha_{ij} W_{ijl'}.$$

8) Existence constraints:

$$\sum_{i\in I_j} W_{ijl} \le E_j,$$

$$V^{\min}_j E_j \le V_j \le V^{\max}_j E_j,$$

$$V^{\min}_s E_s \le V_s \le V^{\max}_s E_s,$$

that correspond to logical restrictions on production unit and storage tank size if this unit-storage tank is present at the optimal design.

9) Production constraints:

$$C_{sl^{\max}} \ge R_s,$$

expressing the requirement of producing at least as much as the market demands for state (s).

10) Objective function:

$$\text{Profit} = \sum_{s\in S_p} p_s C_{sl^{\max}} + \sum_{s\in S_i} p_s (C_{sl^{\max}} - C_{s0}) - \sum_{s\in S_f} p_s C_{s0} - \sum_u c_u U_{uc};$$

the first two terms represent the revenue due to product and intermediate state production, respectively, whereas the last two terms express the cost of raw materials and utilities, respectively,

$$\text{Cost} = \sum_j (E_j \widetilde{\alpha}_j + \widetilde{\beta}_j V_j^{\widetilde{\gamma}_j}) + \sum_s (E_s \widetilde{\alpha}_s + \widetilde{\beta}_s V_s^{\widetilde{\gamma}_s});$$

the first term represent the cost of installing production unit (j), whereas the second term correspond to the cost of storage tank (s).

$$\text{Objective} = \text{Cost} - \text{Profit}.$$

This above formulation correspond to a MINLP problem with decision variables: W_{ijl}, $X_{ijll'}$, B_{ijl}, U^u_{ijl}, T_l that correspond to plant operation and E_j, E_s, V_j, V_s, U_u that represent design decisions. Nonconvexities appear in the timing constraints, material balances, utility constraints as bilinear products of binary and continuous variables and in the objective function in power form of the type $V_j^{\widetilde{\gamma}_j}$ and $V_s^{\widetilde{\gamma}_s}$. The authors proposed an evolutionary algorithm (EA) with simulated annealing (SA), [12], to solve this problem. They utilized simulated annealing to improve the poor local search ability of EA. A suitable encoding procedure is proposed which results in reduction in the number of constraints and variables by up to 50%. In particular, they explored the mathematical structure of the problem in the following sense. If $W_{ijl} = 1$ and $X_{ijll'} = 1$, unit j exists, it executes operation k which starts at $ST^j = l$ finishes at $FT^j_k = l'$ involving task $TS^j_k = i$ with batch size $BS^j_k = B_{ijl}$ and utility usage $U^j_{uk} = U^u_{ijl}$. So they proposed to replace W_{ijl}, $X_{ijll'}$, B_{ijl} and U^u_{ijl} by the operation sequence of tasks in units: task sequence $TS^j = (i_1, \ldots, i_{N_j})$, task batch size $BS^j = (B_1, \ldots, B_{N_j})$, task utility usage $U^j_u = (U^u_1, \ldots, U^u_{N_j})$, start time $ST^j = (l_1, \ldots, l_{N_j})$, finish time $FT^j = (l'_1, \ldots, l'_{N_j})$. In this way the decision variables become (E_j, V_j, E_s, V_s, U_u, T_l, TS^j, BS^j, U^j_u, ST^j, FT^j). The algorithm starts with an initial guess and evolves a number of candidate instances for these variables. The allocation and the capacity constraints are automatically satisfied by each candidate solution and T_l are chosen so that the timing constraints are also satisfied. Two examples are presented to illustrate the applicability of the proposed approach to solve batch design problem involving detailed scheduling constraints. Linear and nonlinear task processing times and unit cost models are considered for both the examples. For the first example considering linear functions for processing times and unit cost models the results obtained are compared with a discrete time formulation, [8], and found to outperform it in terms of number of variables which is expected since the formulation is based on the continuous time description and the computational requirement for the solution of their model. Considering nonlinear models for processing times and unit costs, the resulting model for a problem with 4 production units, 4 storage tanks, 5 tasks and 4 states, involves 62 integer and 34 continuous variables and 122 constraints. This example was the largest presented in this work, and required considerable computational effort, 7849.23 CPU seconds on a SUN ULTRAstation-1.

See also

- ▶ Chemical Process Planning
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ Job-shop Scheduling Problem
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis

- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming
- ▶ Stochastic Scheduling
- ▶ Vehicle Scheduling

References

1. Barbosa-Póvoa APFD, Macchietto S (1994) Detailed design of multipurpose batch plants. Comput Chem Eng 18:1014–1042
2. Birewar DB, Grossmann IE (1989) Incorporating scheduling in the optimal design of multiproduct batch plants. Comput Chem Eng 13:141–161
3. Birewar DB, Grossmann IE (1990) Simultaneous synthesis, sizing and scheduling of multiproduct batch plants. Industr Eng Chem Res 29:2242–2251
4. Grossmann IE, Sargent RWH (1979) Optimum design of multipurpose batch plants. Industr Eng Chem Process Des Developm 18:343–348
5. Kocis GR, Grossmann IE (1989) Computational experience with DICOPT solving MINLP problems in process synthesis engineering. Comput Chem Eng 13:307–315
6. Kondili E, Pantelides CC, Sargent RWH (1993) A general algorithm for short-term scheduling of batch operations - I. MILP formulation. Comput Chem Eng 17:211–227
7. Ku H, Karimi I (1986) Scheduling in multistage serial batch processes with finite intermediate storage - Part I. MILP formulation; Part II. Approximate algorithms. AIChE Annual Meeting, Miami
8. Manual gBSS, general batch scheduling system - User manual and language reference, Imperial College
9. Pantelides CC (1994) Unified frameworks for the optimal proces planning and scheduling. Proc. Second Conf. Foundations of Computer Aided Operations, pp 253–274
10. Sparrow RE, Forder GJ, Rippin DWT (1975) The choice of equipment sizes for multiproduct batch plants. Heuristics vs. branch and bound. Industr Eng Chem Process Des Developm 14:197–203
11. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer-approximation method for MINLP optimization. Comput Chem Eng 14:769–782
12. Xia Q, Macchietto S (1997) Design and synthesis of batch plants- MINLP solution based on a stochastic method. Comput Chem Eng 21:S697–S702
13. Zhang X, Sargent RWH (1994) The optimal operation of mixed production facilities - general formulation and some solution approaches for the solution. Proc. 5th Internat. Symp. Process Systems Engin. (Kyongju, Korea), pp 171–177

MINLP: Generalized Cross Decomposition

KAJ HOLMBERG
Department Math., Linköping Institute Technol., Linköping, Sweden

MSC2000: 90C11, 90C30, 49M27

Article Outline

Keywords
The Problem
The Primal Master Problem
The Dual Master Problem
The Subproblems
The Cross Decomposition Algorithm
The Convergence Tests
See also
References

Keywords

Decomposition; Primal-dual; Nonlinear; Mixed integer

Decomposition methods, such as the classical Benders decomposition (cf. ▶ Generalized Benders decomposition), [1], and Dantzig–Wolfe decomposition, [3], have been used to solve many different large structured optimization problems, by decomposing them with the help of relaxation of constraints or fixation of variables. The success of such an approach depends very much on the structure of the problem. In some cases these methods are very efficient, but in other cases they are not competitive with other techniques.

However, the simple elegance of these basic principles has inspired many researchers to propose modifications of the basic methods, mostly aimed at improving the efficiency of the methods, but also aimed at extending the applicability of the approaches.

Dantzig–Wolfe decomposition, originally for linear programming problems, [3], has been extended to convex nonlinear programming problems, [2], under several names, for example generalized linear programming. We will here simply use the term 'nonlinear Dantzig–Wolfe decomposition'.

Benders decomposition, originally for linear mixed integer programming problems, [1], has been extended to partly convex nonlinear programming problems, [5], under the name 'generalized Benders decomposition'.

On the other hand, among the numerous suggestions for modifications to increase the efficiency, there is one which in a way shares the simplicity and clear principle of the basic methods, namely *cross decomposition*, [11]. Usually described as a combination of Benders decomposition and Dantzig–Wolfe decomposition, simultaneously using the two methods in an iterative manner, the method borrows its basic convergence properties from these two methods. However, one can also view cross decomposition as the more general method, and Benders and Dantzig–Wolfe decomposition as modifications of cross decomposition, obtained by excluding one of the subproblems and one of the master problems.

Cross decomposition was originally developed for linear mixed integer programming problems, [11], but the approach is more general and not restricted to such problems. The first application of cross decomposition was to the capacitated facility location problem, [12], and produced a solution method which is recognized as one of the most efficient existing methods for that problem. However, another early application was to the stochastic transportation problem (a convex problem with linear parts), [10].

Here we will describe 'generalized cross decomposition', which was first proposed in [6], and more thoroughly treated in [7]. The generalization of the procedure, parallel to that in [5] for generalized Benders decomposition, enables the solving of nonlinear programming problems with convex parts, for example nonlinear mixed integer programming problems, see for example [4].

The Problem

Consider the following general optimization problem.

$$(\mathrm{P}) \quad \begin{cases} v^* = & \min f(x,y) \\ \text{s.t.} & G_1(x,y) \le 0 \\ & G_2(x,y) \le 0 \\ & x \in X \\ & y \in Y \end{cases}$$

where X and Y are compact, nonempty sets. Assume that X is convex and f, G_1 and G_2 are proper convex functions in x for any fixed $y \in Y$, i. e. that the problem is convex in x. Also assume that that f, G_1 and G_2 are bounded and Lipschitzian on (X, Y). Note that we do not assume any convexity in the y-variables. An important case is when Y is a (finite) set of integers.

Furthermore we assume the following (as was done in [5] for generalized Benders decomposition). The optimization with respect to x of the Lagrangian functions must be possible to do 'essentially independent' of y (called property P by A.M. Geoffrion). We therefore assume that the functions q_1, q_2, q_3 and q_4 exist, such that $f(x, y) + u_1^\top G_1(x, y) + u_2^\top G_2(x, y) = q_1(q_3(x, u), y, u)$, $\forall x, y, u$, and $\widetilde{u}_1^\top G_1(x, y) + \widetilde{u}_2^\top G_2(x, y) = q_2(q_4(x, \widetilde{u}), y, \widetilde{u})$, $\forall x, y, \widetilde{u}$, where q_3 and q_4 are scalar functions, q_1 and q_2 are increasing in their first argument, and $\widetilde{u}$ is assumed to belong to the set of all possible nonnegative, normalized directions $C = \{\widetilde{u} \ge 0 \colon e^\top \widetilde{u} = 1\}$, where e is a vector of ones. Since f, G_1 and G_2 are convex in x and bounded and Lipschitzian on (X, Y), the same applies to q_1 for any fixed $u \ge 0$, and to q_2 for any fixed $\widetilde{u} \in C$.

The optimal solution of P is denoted by (x^*, y^*). We will also mention the case when P is convex, i. e. where f, G_1 and G_2 are convex functions (in y too) and Y is a convex set. Lagrangian duality can be used to get a dual solution (the optimal Lagrange multipliers), denoted by $u^* = (u_1^*, u_2^*)$.

Let us for convenience introduce the following notation.

$$\begin{aligned} L(x,y,u) &= f(x,y) + u_1^\top G_1(x,y) + u_2^\top G_2(x,y), \\ \widetilde{L}(x,y,\widetilde{u}) &= \widetilde{u}_1^\top G_1(x,y) + \widetilde{u}_2^\top G_2(x,y), \\ L_1(x,y,u_1) &= f(x,y) + u_1^\top G_1(x,y), \\ \widetilde{L}_1(x,y,\widetilde{u}_1) &= \widetilde{u}_1^\top G_1(x,y). \end{aligned}$$

The Primal Master Problem

Using the primal structure of (P) we can rewrite it as

$$v^* = \min_{y \in V} h(y),$$

where $\forall y = \in V$,

$$\begin{cases} h(y) = & \min f(x, y) \\ \text{s.t.} & G_1(x, y) \leq 0 \\ & G_2(x, y) \leq 0 \\ & x \in X \end{cases}$$

and

$$V = \left\{ y \in Y\colon\ \exists x \in X\colon \begin{array}{l} G_1(x, y) \leq 0, \\ G_2(x, y) \leq 0 \end{array} \right\}.$$

The problem is convex in x, so we can use Lagrangian duality to get, $\forall y \in V$,

$$h(y) = \max_{u \geq 0} \min_{x \in X} L(x, y, u).$$

A similar expression can be obtained for V:

$$V = \left\{ y \in Y\colon\ \left(\max_{\widetilde{u} \in C} \min_{x \in X} \widetilde{L}(x, y, \widetilde{u}) \right) \leq 0 \right\}.$$

The full primal master problem is given below:

$$\begin{cases} v^* & = \min q \\ \text{s.t.} & q \geq \min\limits_{x \in X} L(x, y, u), \quad \forall u \geq 0, \\ & 0 \geq \min\limits_{x \in X} \widetilde{L}(x, y, \widetilde{u}), \quad \forall \widetilde{u} \in C, \\ & y \in Y. \end{cases}$$

This problem has an infinite number of constraints, one for each nonnegative dual point and one for each nonnegative dual direction. Each constraint contains an optimization problem (minimization with respect to x), which should in theory be solved for all $y \in Y$ before the main problem, $\min_{y \in Y} h(y)$, can be solved. However, we have

$$\min_{x \in X} L(x, y, u) = q_1 \left(\min_{x \in X} q_3(x, u), y, u \right)$$

and

$$\min_{x \in X} \widetilde{L}(x, y, \widetilde{u}) = q_2 \left(\min_{x \in X} q_4(x, \widetilde{u}), y, \widetilde{u} \right).$$

Since q_1 and q_2 are proper, convex, bounded and Lipschitzian on X, and X is compact and convex, the optima in x (for fixed u and $\widetilde{u}$) will be attained. q_1 and q_2 are increasing in their first argument, so the minimization in x can be made in q_3 and q_4 instead, and the value of y will thus not influence the result of this minimization. The minimization over x can be made once (for any y) and the result will then be true for all $y \in Y$.

The *relaxed primal master problem* only contains a finite number of cuts (with index sets P_U and R_U) which gives an approximate description of $h(y)$ and V, and an optimal objective function value, $v_{PM} \leq v^*$. Since the part of the problem that is described by the constraints is convex in x, v_{PM} will converge asymptotically towards v^* as the sets of constraints grow.

The constraints can now be expressed as

$$q \geq q_1 \left(\min_{x \in X} q_3(x, u^{(k)}), y, u^{(k)} \right), \quad \forall k \in P_U,$$
$$0 \geq q_2 \left(\min_{x \in X} q_4(x, \widetilde{u}^{(k)}), y, \widetilde{u}^{(k)} \right), \quad \forall k \in R_U.$$

The minimization in x can now be made independently in each constraint, since the other arguments in q_3 and q_4, namely u and $\widetilde{u}$, are fixed. Since the minima are attained, we use the notation $x^{(k)}$, $\forall k \in P_U$, and $\widehat{x}^{(k)}$, $\forall k \in R_U$, for the minimizers of q_3 and q_4.

Inserting this, we obtain the final form of the relaxed primal master problem.

$$\text{(PM)} \quad \begin{cases} v_{\text{PM}} & = \min q \\ \text{s.t.} & q \geq L(x^{(k)}, y, u^{(k)}), \quad \forall k \in P_U, \\ & 0 \geq \widetilde{L}(\widehat{x}^{(k)}, y, \widetilde{u}^{(k)}), \quad \forall k \in R_U, \\ & y \in Y. \end{cases}$$

The constraints in the first set are called *value cuts*, and those in the second set are called *feasibility cuts*.

The Dual Master Problem

Using Lagrangian duality on (P) yields a relaxation and a lower bound, v_L, on v^*:

$$v_L = \max_{u_1 \geq 0} g(u_1)$$

where, $\forall u_1 \geq 0$,

$$\begin{cases} g(u_1) & = \min L_1(x, y, u_1) \\ \text{s.t.} & G_2(x, y) \leq 0 \\ & x \in X \\ & y \in Y. \end{cases}$$

This leads to a dual master problem, which is a convexification of the problem. If (P) is not convex a duality gap might occur. We denote the subset of the solutions that are included by $(x^{(k)}, y^{(k)})$, $\forall k \in P_X$, and obtain the restricted dual master problem as

$$\text{(DM)} \quad \begin{cases} v_{DM} & = \max q \\ \text{s.t.} & q \leq L_1(x^{(k)}, y^{(k)}, u_1), \\ & \quad \forall k \in P_X, \\ & u_1 \geq 0. \end{cases}$$

The Subproblems

The primal subproblem is a convex problem in x, obtained by fixing y to $\overline{y}$.

$$\text{(PS)} \quad \begin{cases} h(\overline{y}) & = \min f(x, \overline{y}) \\ \text{s.t.} & G_1(x, \overline{y}) \leq 0 \\ & G_2(x, \overline{y}) \leq 0 \\ & x \in X. \end{cases}$$

A solution to (PS) is assumed to consist of both a primal solution, $x^{(k)}$, and a dual solution, $(u_1^{(k)}, u_2^{(k)})$. Due to the convexity we can use Lagrangian duality without creating a duality gap.

$$\text{(PSL)} \quad h(\overline{y}) = \sup_{u \geq 0} \min_{x \in X} L(x, \overline{y}, u).$$

If (PS) is infeasible, (PSL) will be unbounded in u, and a solution is represented by a direction, $\widetilde{u}^{(k)}$. A valid cut for the primal master problem also requires a corresponding primal solution, $\widehat{x}^{(k)}$, obtained by solving

$$\min_{x \in X} \widetilde{L}(x, \overline{y}, \widetilde{u}^{(k)}).$$

(Note that $\widehat{x}^{(k)}$ is not feasible in (PS).)

The dual subproblem is the following (nonconvex) problem, obtained by relaxing the first set of constraints in (P) and fixing the Lagrange multipliers u_1 to $\overline{u}_1$:

$$\text{(DS)} \quad \begin{cases} g(\overline{u}_1) & = \min L_1(x, y, \overline{u}_1) \\ \text{s.t.} & G_2(x, y) \leq 0 \\ & x \in X \\ & y \in Y \end{cases}$$

To handle unbounded dual solutions, $\widetilde{u}_1$, we can use the following subproblem:

$$\text{(UDS)} \quad \begin{cases} \widetilde{v}(\widetilde{u}_1) & = \min \widetilde{L}_1(x, y, \widetilde{u}_1) \\ \text{s.t.} & G_2(x, y) \leq 0 \\ & x \in X \\ & y \in Y. \end{cases}$$

(UDS) does not produce a bound on v^*, but if $\widetilde{v}(\widetilde{u}_1) \leq 0$ it yields a dual cut that will eliminate $\widetilde{u}_1$.

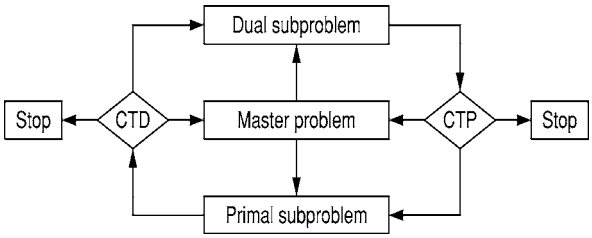
The Cross Decomposition Algorithm

In the subproblem phase of the cross decomposition method we iterate between the primal subproblem (PS) and the dual subproblem (DS) (or (UDS)).

The primal subproblem, (PS), supplies an upper bound, $h(\overline{y})$, on v^*, and $\overline{u}_1$ for the dual subproblem. The dual subproblem, (DS), supplies a lower bound, $g(\overline{u}_1)$, on v^*, and $\overline{y}$ for the primal subproblem. If (PS) has an unbounded solution, $\widetilde{u}_1$, we use (UDS) (instead of (DS)) to get $\overline{y}$.

Unfortunately, the lack of controllability for the important parts of the solutions, y and u_1, which occurs unless the problem is strictly convex, implies that this procedure alone cannot be expected to converge to the optimal solution.

We therefore need to use the master problems to ensure convergence. (PM) or (DM) can be solved with all the constraints generated by the subproblem solutions. We have all the known results for generalized Benders or nonlinear Dantzig–Wolfe decomposition to fall back on, so this technique is well known. After the solution of one master problem, the subproblem phase is reentered. (We do not switch to Benders or Dantzig–Wolfe decomposition completely.)



MINLP: Generalized Cross Decomposition, Figure 1

We will later describe convergence tests that tell us exactly when to use a master problem. The existence of such convergence tests is a very important aspect of cross decomposition. Let us, before getting any further, give below a short algorithm for cross decomposition algorithm.

Let us denote the convergence test in step 3 (before (PS)) by CTP and the convergence test in step 6 (before (DS)) by CTD. The optimality tests (step 2 and step 5) are included in the convergence tests, and the decision about where to go is based on the results of both tests. The algorithm is pictured in Fig. 1.

0	Get a starting $\overline{u}$.
1	Solve (DS) (or (UDS)).
2	IF optimal go to 8.
3	IF not convergence, go to 7A (or 7B).
4	Solve (PS).
5	IF optimal go to 8.
6	IF not convergence go to 7B (or 7A). ELSE go to 1.
7A	Solve (PM). Go to 4.
7B	Solve (DM). Go to 1.
8	Stop. The solution from (PS) is optimal.

We can start with either one of the subproblems, so a good primal starting solution can also be utilized.

If CTP indicates that (PS) will not give further convergence, we use (PM). If CTD indicates failure of convergence for (DS), we can use (DM) (which however gives certain convergence only if (P) is convex). After (PM) we go to (PS) and after (DM) we go to (DS), in order to make use of the output of the master problems. In the general nonconvex case, it is not necessary to use (DM). It is even possible to omit the convergence tests CTD if only (DM) is used.

The Convergence Tests

Returning to the question of convergence in the subproblem phase, we make the following definitions of ε-improvements.

'ε-bound-improvement' is an improvement of at least ε of the upper or lower bound.

'ε-cut-improvement' is a generation of a new, so far unknown cut, that is at least ε better (i. e. has a value of at least ε higher or lower) than all known cuts at some point.

Discussing *linear mixed integer problems*, as in [11], one can let $\varepsilon = 0$. In such a case we simply omit ε from the above notation.

Cut-improvement thus means that a new cut will be included in one of the restricted master problems and that the description of the functions $h(y)$ or $g(u_1)$ or the set Y is refined. By 'improvement' we will, in the rest of this paper mean bound-improvement and/or cut-improvement. When using unbounded solutions as input no finite bounds are obtained, so bound-improvement can not appear. Also, a cut giving a cut-improvement can be a value cut or a feasibility cut, i. e. generated by output in the form of unbounded as well as bounded solutions.

Let us by *primal cut-improvement* denote generation of a primal cut (for (PM)) and by *dual cut-improvement* denote generation of a dual cut (for (DM)). We also use the notation 'primal' or 'dual bound-improvement' to indicate which of the two subproblems that gave the improvement, i. e. *primal bound-improvement* means that $h(\overline{y}) < \overline{v}$ and *dual bound-improvement* means that $g(\overline{u}_1) > \underline{v}$. ($\overline{v}$ is the least upper bound known and $\underline{v}$ the largest lower bound known.)

The convergence tests are originally formulated to give the answers to the following questions.

- Can $\overline{y}$ give a bound-improvement in (PS)?
- Can $\overline{u}_1$ give a bound-improvement in (DS)?

Testing extreme rays, $\widetilde{u}_1$, for convergence, we note that the subproblem (UDS) can not give bound-improvement. We call the test of unbounded solutions CTDU.

We now give the convergence tests, CT, with strict inequalities, following [11]:

CTP	If $L(x^{(k)}, \overline{y}, u^{(k)}) < \overline{v}$, $\forall k \in P_U$, and $\widetilde{L}(\widehat{x}^{(k)}, \overline{y}, \widetilde{u}^{(k)}) < 0$, $\forall k \in R_U$, then $\overline{y}$ will give primal improvement. If not, use a master problem.
CTD	If $L_1(x^{(k)}, y^{(k)}, \overline{u}_1) > \underline{v}$, $\forall k \in P_X$, then $\overline{u}_1$ will give dual improvement. If not, use a master problem.
CTDU	If $\widetilde{L}_1(x^{(k)}, y^{(k)}, \widetilde{u}_1) > 0$, $\forall k \in P_X$, then $\widetilde{u}_1$ will give dual cut-improvement. If not, use a master problem.

We call CTD and the first part of CTP *value convergence tests* and CTDU and the second part of CTP *feasibility convergence tests*. This conforms to the notation of value and feasibility cuts in the master problems.

One can show that the convergence tests CTP and CTD are necessary for bound-improvement and sufficient for cut- or bound-improvement, see [7]. The convergence tests CTDU are sufficient for cut-improvement.

However, there can be an infinite number of primal and/or dual improvements, so one can not be certain that CT will fail within a finite number of steps. For this reason it is necessary to consider ε-improvements.

We need the following ε-convergence tests, CT ε:

CTPε	If $L(x^{(k)}, \overline{y}, u^{(k)}) \leq \overline{v} - \varepsilon$, $\forall k \in P_U$, and $\widetilde{L}(\widehat{x}^{(k)}, \overline{y}, \widetilde{u}^{(k)}) \leq -\varepsilon$, $\forall k \in R_U$, then $\overline{y}$ will give primal ε-improvement. If not, use a master problem.
CTDε	If $L_1(x^{(k)}, y^{(k)}, \overline{u}_1) \geq \underline{v} + \varepsilon$, $\forall k \in P_X$, then $\overline{u}_1$ will give dual ε-improvement. If not, use a master problem.
CTDUε	If $\widetilde{L}_1(x^{(k)}, y^{(k)}, \widetilde{u}_1) \geq \varepsilon$, $\forall k \in P_X$, then $\widetilde{u}_1$ will give dual ε-cut-improvement. If not, use a master problem.

The ε-value convergence tests correspond to the value cuts of the master problems, and the ε used corresponds directly to a change of ε of the bounds (ε-bound-improvement). The ε-feasibility convergence tests, on the other hand, correspond to feasibility cuts of the master problems, and the ε used corresponds to the 'infeasibility' it gives some previously feasible points, which is what we call ε-cut-improvement for feasibility cuts. While these ε-tests are sufficient for ε-improvement, they are not necessary. To prove necessity would require an inverse Lipschitz assumption, namely that for points a certain distance apart, the value of a function (the feasibility cut) should differ by at least a certain amount. The following result is proved in [7].

The ε-value convergence tests of CTP ε, the feasibility convergence tests of CTP and the ε-convergence tests CTD ε are necessary for ε-bound-improvement. The ε-convergence tests CT ε are sufficient for ε-bound- or ε-cut-improvement, in the sense that they are sufficient for one of the following.

I) ε-bound-improvement.
II) ε-cut-improvement.
III) ε_1-bound-improvement and ε_2-cut-improvement, where $\varepsilon_1 + \varepsilon_2 = \varepsilon$.

Now it is possible to verify finiteness of the convergence tests. A formal proof for this can be found in [7]. The following reasoning is used.

When the bounded set Y is completely described with an accuracy better than ε by either value cuts or feasibility cuts, the ε-convergence tests will fail (if not earlier). Each time the ε-convergence tests do not fail, we will get improvement according to one of the three cases mentioned above.

A finite number of ε-bound-improvements is obviously sufficient to decrease the finite distance between $\overline{v}$ and v^* to less than ε. After an ε-cut-improvement, the new cut describes $h(y)$ with an accuracy better than ε in the area around $\overline{y}$ where $h(y) < L(x^{(l)}, y, u^{(l)}) + \varepsilon$. Due to the Lipschitzian property of the functions f, G_1 and G_2, there is a least distance, δ, proportional to ε, from $\overline{y}$ to any point y violating this inequality, and the ε-convergence tests will fail for any point with a distance to $\overline{y}$ less then δ. The bounded set V can be completely covered by a finite number of such areas.

In the third case, an ε_1-bound-improvement together with an ε_2-cut-improvement, where $\varepsilon_1 + \varepsilon_2 = \varepsilon$, we can ignore the least of ε_1 and ε_2, leaving us with the other one greater or equal to $\varepsilon/2$. This yields one of the two cases above, so exchanging ε for $\varepsilon/2$ finiteness is still assured.

For unbounded solutions to (PS), any y satisfying $\widetilde{L}(\widehat{x}^{(l)}, y, \widetilde{u}^{(l)}) > -\varepsilon$ will make the ε-convergence tests fail, and because of the Lipschitzian property of G_1 and G_2 there is a least distance, δ (proportional to ε), from $\overline{y}$ to any y not making the ε-convergence tests fail. Thus an area of a certain least size is made 'infeasible', and

the bounded set $Y \setminus V$ can be covered by a finite set of such areas. Thus CTP ε will fail within a finite number of steps.

Note that it is enough that CTP ε fails. To obtain finiteness we do not need to use CTD ε, even if it might be useful in practice. We cannot show that CTD ε will fail within a finite number of steps. Dual ε-bound-improvement can only occur a finite number of times, but dual ε-cut-improvement can occur an infinite number of times, since the area to be covered by the cuts is the nonnegative orthant of u_1.

We therefore require that (PM) is used regularly. (One could even skip (DM) completely.) The following is our main result.

Theorem 1 *The generalized cross decomposition algorithm equipped with ε-convergence tests CT ε finds an ε-optimal solution to (P) in a finite number of steps, if the generalized Benders decomposition algorithm does.*

All the results for generalized Benders decomposition can be directly used for generalized cross decomposition, especially the following two.

In [5] it is shown that generalized Benders decomposition has finite exact convergence if Y is a finite discrete set. The worst case is solving the primal subproblem with each possible $y \in Y$, which will give a perfect description of $h(y)$ and V on Y.

Therefore we know that if Y is a finite discrete set, the generalized cross decomposition algorithm will solve P exactly in a finite number of steps.

It is also shown in [5] that generalized Benders decomposition terminates in a finite number of steps to an ε-optimal solution, i. e. where $\overline{v} - \underline{v} \leq \varepsilon$ for any given $\varepsilon > 0$, if the set of interesting (u_1, u_2)-solutions (possible optimal solutions to the primal subproblem) is bounded and $Y \subseteq V$. This makes the primal feasibility cuts (and the corresponding convergence tests) unnecessary. So for generalized cross decomposition, we know the following.

If $h(y)$ is bounded from above for all $y \in Y$, i. e. (PS) has a feasible solution for every $y \in Y$, then the cross decomposition algorithm (without UDS and the ε-feasibility convergence tests of CT ε) will yield finite ε-convergence, i. e. yield $\overline{v} - \underline{v} \leq \varepsilon$ in a finite number of steps, for any given $\varepsilon > 0$.

If $Y \not\subseteq V$ one might get asymptotic convergence of the feasibility cuts, i. e. solutions getting closer and closer to the feasible set, but never actually becomes feasible. If one is reluctant to base a stopping criterion on ε-feasible solutions, one could use penalty functions, which transforms feasibility cuts to value cuts and gives better possibilities of handling cases where $Y \not\subseteq V$. One could also use artificial variables for this purpose. As for nonlinear penalty function techniques, one should not forget the Lipschitzian assumption made.

The practical motivation behind cross decomposition is to replace the hard primal master problem with the easier dual subproblem to the largest possible extent. Therefore the theoretical result that generalized cross decomposition equipped with ε-convergence tests does not have asymptotically weaker convergence than generalized Benders decomposition, is quite satisfactory.

Finally one might mention that these approaches also has been applied to pure (not mixed) integer programming problems in [8] (nonlinear) and [9] (linear). In such cases, various duality gaps appear, and exact solution is not possible. However, the approach may be useful for obtaining good bounds on the objective function value, which are to be used in branch and bound methods.

See also

- ▶ Chemical Process Planning
- ▶ Decomposition Principle of Linear Programming
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis

- ► Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ► Mixed Integer Nonlinear Programming
- ► Simplicial Decomposition
- ► Simplicial Decomposition Algorithms
- ► Stochastic Linear Programming: Decomposition and Cutting Planes
- ► Successive Quadratic Programming: Decomposition Methods

References

1. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Math 4:238–252
2. Dantzig GB (1963) Linear programming and extensions. Princeton Univ. Press, Princeton
3. Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8:101–111
4. Floudas CA (1995) Nonlinear and mixed-integer optimization: Fundamentals and applications. Oxford Univ. Press, Oxford
5. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
6. Holmberg K (1985) Decomposition in large scale mathematical programming. PhD Thesis Dept. Math. Linköping Univ.
7. Holmberg K (1990) On the convergence of cross decomposition. Math Program 47:269–296
8. Holmberg K (1992) Generalized cross decomposition applied to nonlinear integer programming problems: Duality gaps and convexification in parts. Optim 23:341–356
9. Holmberg K (1994) Cross decomposition applied to integer programming problems: Duality gaps and convexification in parts. Oper Res 42(4):657–668
10. Holmberg K, Jörnsten K (1984) Cross decomposition applied to the stochastic transportation problem. Europ J Oper Res 17:361–368
11. Van Roy TJ (1983) Cross decomposition for mixed integer programming. Math Program 25:46–63
12. Van Roy TJ (1986) A cross decomposition algorithm for capacitated facility location. Oper Res 34:145–163

MINLP: Global Optimization with αBB

CLAIRE S. ADJIMAN, CHRISTODOULOS A. FLOUDAS
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 65K05, 90C11, 90C26

Article Outline

Keywords
The SMIN-αBB Algorithm
  Generation of Valid Upper and Lower Bounds
  Selection of Branching Node
  Selection of Branching Variable
  Variable Bound Updates
  Algorithmic Procedure
The GMIN-αBB Algorithm
  Branching Variable Selection
  Generation of a Valid Lower Bound
  Generation of a Valid Upper Bound
  Variable Bound Updates
Conclusions
See also
References

Keywords

Global optimization; Twice-differentiable MINLPs; Branch and bound; αBB algorithm

The αBB global optimization algorithm for continuous twice-differentiable NLPs (cf. ► αBB algorithm) [2,4,5,6,8,18] can be used to design global optimization algorithms for *mixed integer nonconvex problems* [1,3,7]. One such algorithm, the *special structure mixed integer αBB algorithm* (SMIN-αBB) is designed to address the class of *MINLPs* in which all the integer variables are binary variables that participate in linear or mixed-bilinear terms and in which the nonconvex functions in the continuous variables have continuous second order derivatives. This algorithm is an extension of the αBB algorithm and branching is performed on both the continuous and the binary variables. A second algorithm, the *general structure mixed integer αBB algorithm* (GMIN-αBB), guarantees convergence to the global optimum of a much broader class of problems. The integer variables may participate in the problem in a very general way, provided that the continuous relaxation of the MINLP is C^2 continuous. This article describes both algorithms.

The SMIN-αBB Algorithm

The *SMIN-αBB algorithm* [1,3,7] guarantees finite ϵ-convergence to the global solution of MINLPs belong-

ing to the class

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x}) + \mathbf{x}^\top A_f \mathbf{y} + c_f^\top \mathbf{y} \\ \text{s.t.} & g_i(\mathbf{x}) + \mathbf{x}^\top A_{g,i} \mathbf{y} + c_{g,i}^\top \mathbf{y} \le 0, \\ & \quad i = 1, \ldots, m, \\ & h(\mathbf{x}) + \mathbf{x}^\top A_{h,i} \mathbf{y} + c_{h,i}^\top \mathbf{y} = 0, \\ & \quad i = 1, \ldots, p, \\ & \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U] \\ & \mathbf{y} \in \{0,1\}^q \end{cases} \tag{1}$$

where $f(\mathbf{x})$, $g(\mathbf{x})$, and $h(\mathbf{x})$, are continuous, twice-differentiable functions, m is the number of inequality constraints, p is the number of equality constraints, q is the dimension of the binary variable vector, A_f, $A_{g,i}$ and $A_{h,i}$ are $n \times q$ matrices, and c_f, $c_{g,i}$ and $c_{h,i}$ are q-dimensional vectors.

The main features of any *branch and bound algorithm* are the strategy used to generate valid lower and upper bounds for the problem and the selection criteria for the branching node and the branching variable. Optionally, a procedure to tighten the variable bounds may be considered. Each one of these issues is examined in the context of the SMIN-αBB algorithm.

Generation of Valid Upper and Lower Bounds

A local solution of the nonconvex MINLP (1) using one of the algorithms described in [13] constitutes a valid upper bound on the global optimum solution of that problem. The *generalized Benders decomposition* (GBD) [10,14] or a standard MINLP branch and bound algorithm (B&B) [9,11,15,19,20] may be used to obtain such a solution. When there are no mixed-bilinear terms, the *outer approximation with equality relaxation* (OA/ER) [12,16] may also be used. Alternatively, the binary variables may be fixed to a combination of 0 and 1 values and the resulting nonconvex NLP may be solved locally.

A relaxed problem which can be solved to global optimality must be constructed from problem (1) in order to obtain a valid lower bound. The class of MINLPs in which the continuous functions $f(\mathbf{x})$, $g_i(\mathbf{x})$, and $h_i(\mathbf{x})$, are convex can be solved to global optimality using the GBD or B &B algorithms, and, when there are no mixed-bilinear terms, the OA/ER algorithm. To identify a *guaranteed lower bound* on the solution of the problem, it therefore suffices to construct convex underestimators for the nonconvex functions $f(\mathbf{x})$, $g_i(\mathbf{x})$, and $h_i(\mathbf{x})$, and to solve the resulting problem with one of these algorithms. The rigorous *convexification/relaxation strategy* used in the αBB algorithm for nonconvex continuous problem [2,4,5,6] allows the construction of the desired lower bounding MINLP. This scheme is based on a decomposition of the functions into a sum of terms with special mathematical structure, such as linear, convex, bilinear, trilinear, fractional, fractional trilinear, univariate concave and general nonconvex terms. A different convex relaxation technique is then applied for each class of term. The fact that a summation of convex functions is itself a convex function is then used to construct overall function underestimators and arrive at a convex lower bounding MINLP.

Selection of Branching Node

A list of the lower bounds on all the nodes that have not yet been explored during the branch and bound procedure is maintained. A number of approaches can be used to select the next branching node, such as depth-first, breadth-first or smallest lower bound first. Since the purpose of the algorithm is to identify the global solution of the problem, all promising regions, that is, all regions for which the lower bound is less than or equal to the best upper bound on the solution, must be explored. The strategy that usually minimizes the number of nodes to be examined and therefore the CPU requirements of the algorithm is used to choose the next branching node in the SMIN-αBB algorithm. Thus, the node with the smallest lower bound is selected.

Selection of Branching Variable

Several strategies can be used to select the next variable to be branched on. If a continuous variable is judiciously chosen, the partition results in an improvement of the lower bound on the problem through a tightening of the convex relaxation of the nonconvex continuous functions. Binary variables have an indirect effect

on the quality of the convex underestimators as they influence the range of values that the continuous variables can take on.

A first branching variable selection scheme exploits the direct relationship between the range of the continuous variables and the quality of the lower bounds and therefore branches only on these variables. One of the rules available for the αBB algorithm [2] is used for the selection. These are based on the size of the variable ranges, or on a measure of the quality of the underestimator for each term, or on a measure of each variable 's overall contribution to the quality of the underestimators.

A second approach aims to first tackle the combinatorial aspects of the problem by branching only on binary variables for the first q levels of the branch and bound tree, where q is the number of binary variables. The nonconvexities are dealt with on subsequent levels of the tree, by branching on the continuous variables. The specific binary variable used for branching is chosen randomly or from a priority assigned on the basis of its effect on the structure of the problem. In particular, the binary variables that influence the bounds on the greatest number of variables are given the highest priorities. Once all the binary variables have been fixed, the problems that must be considered are *continuous* nonconvex and convex problems for the upper and lower bound respectively. The bounding of the nodes below level q is therefore less computationally intensive than above that level.

A third approach also involves branching on the continuous and binary variables although the choice is no longer based on the level in the tree. To increase the impact of binary variable branching on the quality of the lower bound, such a variable is selected when a continuous relaxation of the problem indicates that the two children node will have significantly different lower bounds, and that one of them may even be infeasible. Thus, if one of the binary variables is close to 0 or 1 at a local solution of the continuous relaxation, it is branched on. The degree of closeness is an arbitrary parameter which can typically be set to 0.1 or 0.2. If no 'almost-integer' binary variable is found, a continuous variable is selected for branching. In general, this hybrid strategy results in a faster improvement in the lower bounds than the second approach, but it is more computationally intensive because a continuous relaxation must be solved before selecting a branching variable and a larger number of MINLP nodes may be encountered during the branch and bound search.

Variable Bound Updates

The tightening of variable bounds is a very important step because of its impact on the quality of the underestimators. For continuous variables, the strategies developed for the αBB algorithm may be used [2]. For the SMIN-αBB algorithm, they rely on the solution of several convex MINLPs in the optimization-based approach, or the iterative interval evaluation of the constraints in the interval-based approach. In this latter case, the binary variables are relaxed during the interval computation.

```
PROCEDURE binary variable bound update()
    Consider R = {(x, y) ∈ F : y_i = 0};
    Test interval feasibility of R;
    IF infeasible, set y_i^L = 1;
    Consider R = {(x, y) ∈ F : y_i = 1};
    Test interval feasibility of R;
    IF infeasible,
        IF y_i^L = 1, RETURN(infeasible node);
        ELSE, set y_i^U = 0;
    RETURN(new bounds y_i^L and y_i^U);
END binary variable bound update;
```

Procedure for binary variable bound updates

In the case of binary variables, successful bound updates are beneficial in two ways. First, they indirectly lead to the construction of tighter underestimators as they affect the continuous variable bounds. Second, they allow a binary variable to be fixed and therefore decrease the number of combinations that potentially need to be explored. An interval-based strategy can be used to carry out binary variable bound updates. Given the current upper bound $\overline{f}^*$ on the global optimum solution, the feasible region F is defined by the constraints appearing in the nonconvex problem, a new

constraint $f(\mathbf{x}) + \mathbf{x}^\top A_f \mathbf{y} + c_f^\top \mathbf{y} \le \overline{f}^*$, and the box $(\mathbf{x}, \mathbf{y}) \in [\mathbf{x}^L, \mathbf{x}^U] \times [\mathbf{y}^L, \mathbf{y}^U]$. Consider a variable $y_i \in \{0, 1\}$ whose bounds are being updated. The procedure above is used.

Algorithmic Procedure

The algorithmic procedure for the SMIN-αBB algorithm is as follows:

```
PROCEDURE SMIN-αBB algorithm()
  Decompose functions in problem;
  Set tolerance ϵ;
  Set f_*^* = f_*^0 = −∞ and f̄^* = f̄^0 = +∞;
  Initialize list of lower bounds {f_*^0};
  DO f̄^* − f_*^* > ϵ
    Select node k with smallest lower bound, f_*^k,
    from list of lower bounds;
    Set f_*^* = f_*^k;
    (Optional) Update binary and continuous variable bounds;
    Select binary or continuous branching variable
    Partition to create new nodes;
    DO for each new node i
      Generate convex lower bounding MINLP;
      Find solution f_*^i of convex lower bounding
      MINLP;
      IF infeasible or f_*^i > f̄^* + ϵ
        Fathom node;
      ELSE
        Add f_*^i to list of lower bounds;
        Find a solution f̄^i of nonconvex MINLP;
        IF f̄^i < f̄^* THEN Set f̄^* = f̄^i;
    OD;
  OD;
  RETURN(f̄^* and variables values at corresponding node);
END SMIN-αBB algorithm;
```

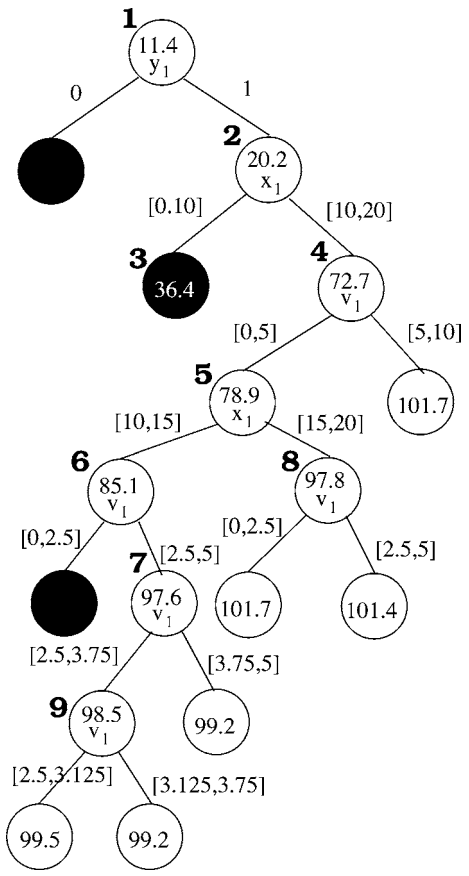
Pseudocode for the SMIN-αBB algorithm

In order to illustrate the algorithmic procedure, a small example proposed in [17] is used. It is a simple design problem where one of two reactors must be chosen to produce a given product at the lowest possible cost. It involves two binary variables, one for each reactor, and seven continuous variables. The formulation is:

$$
\begin{cases}
\min & 7.5y_1 + 5.5y_2 + 7v_1 + 6v_2 + 5x \\
\text{s.t.} & z_1 - 0.9\left(1 - e^{-0.5v_1}\right) x_1 = 0 \\
& z_2 - 0.8\left(1 - e^{-0.5v_2}\right) x_2 = 0 \\
& x_1 + x_2 - x = 0 \\
& z_1 + z_2 = 10 \\
& v_1 - 10y_1 \le 0 \\
& v_2 - 10y_2 \le 0 \\
& x_1 - 20y_1 \le 0 \\
& x_2 - 20y_2 \le 0 \\
& y_1 + y_2 = 1 \\
& 0 \le x_1, x_2 \le 20; \quad 0 \le z_1, z_2 \le 30 \\
& 0 \le v_1, v_2 \le 10; \quad 0 \le x \le 20 \\
& (y_1, y_2) \in \{0, 1\}^2
\end{cases}
$$

Because of the linear participation of the binary variables, the SMIN-αBB algorithm is well-suited to solve this nonconvex MINLP. It identifies the global solution of 99.2 after nine iterations, when bound updates are performed at every iteration and branching takes place on the binary variables first. Branching variable selection takes place randomly for the binary variables and according to the term measures for the continuous variables. At the global solution, the binary variable values are $y_1 = 1$ and $y_2 = 0$. The steps of the algorithm are shown in Fig. 1. The boldface numbers next to the nodes indicate the order in which the nodes were explored. The lower bound is computed by solving a convex relaxation of the nonconvex problem is indicated inside each node, and the branching variable selected for the node is also specified. The domain to which this branching variable is restricted is displayed along each branch. A black node indicates the lower bounding problem was found infeasible and a shaded node is fathomed because its lower bound is greater than the current upper bound on the solution.

At the first node, the initial lower bound is 11.4 and an upper bound of 99.2 is found. The binary variable y_1 is selected as a branching variable. The region $y_1 = 0$ is infeasible and can therefore be fathomed (black node), while an improved lower bound is found for $y_1 = 1$. This latter region is therefore chosen for exploration at the second iteration. Variable bound updates reveal that $y_2 = 1$ is infeasible so that y_2 can be fixed to zero. Branch-



MINLP: Global Optimization with αBB, Figure 1
SMIN-αBB branch and bound tree

ing on the continuous variables may now begin. The first selected variable is x_1 and regions $0 \leq x_1 \leq 10$ and $10 \leq x_1 \leq 20$ are created. Since the left region has the lowest lower bound (36.4), it is examined at iteration 3. Variable bound updates show that this region is in fact infeasible and it is therefore eliminated without further processing. The algorithm proceeds to node 4 for which v_1 is selected as a branching variable. The right region, $5 \leq v_1 \leq 10$, is fathomed since it has a lower bound greater than 99.2. The algorithm progresses along the branch and bound until, at iteration 9, two nodes are left open with lower bounds of 99.2. This is within the accuracy required for this run so the procedure is terminated. One more iteration would reveal that the only global optimum lies in the right child of node 9.

The SMIN-αBB algorithm is especially effective for chemical process synthesis problem such as distillation network or heat exchanger network synthesis [1,3].

The GMIN-αBB Algorithm

The *GMIN-αBB algorithm* is designed to address the broad class of problems represented by

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq 0 \\ & \mathbf{h}(\mathbf{x},\mathbf{y}) = 0 \\ & \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U] \\ & \mathbf{y} \in [\mathbf{y}^L, \mathbf{y}^U] \cap \mathbb{N}^q \end{cases} \tag{2}$$

where $f(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$, and $\mathbf{h}(\mathbf{x}, \mathbf{y})$, are functions whose continuous relaxation is twice continuously differentiable.

The GMIN-αBB algorithm [2,3,7] extends the applicability of the standard branch and bound approaches for MINLPs [9,11,13,15,19,20] by making use of the αBB-algorithm. The most crucial characteristics of the algorithm are the branching strategy, the derivation of a valid lower bound on problem (2), and the variable bound update strategies.

Branching Variable Selection

Branching in the GMIN-αBB algorithm is carried out on the integer variables only. When it is a bisection, the partition takes place either at the midpoint of the range of the selected variable, or at the value of that variable at the solution of the lower bounding problem. It is also possible to branch on more than one variable at a given node, or to perform k-section on one of the variables. More than two children node may be created from a parent node when the structure of the problem is such that the bounds on a small fraction of the integer variables affect the bounds on many of the other variables in the problem. As in the SMIN-αBB algorithm, an integer variable is chosen randomly or according to branching priorities. An additional rule consists of selecting the most or least fractional variable at the solution of a continuous relaxation of the problem.

Generation of a Valid Lower Bound

A guaranteed lower bound on the global solution of the current node of the branch and bound tree is obtained by solving a continuous relaxation of the nonconvex MINLP at that node. When the integer variables

that have not yet been fixed are allowed to vary continuously between their bounds, the problem becomes a nonconvex NLP. The validity of the lower bound can only be ensured if the global solution of this nonconvex NLP is identified or if a lower bound on this solution is found. On the other hand, when all integer variables have been fixed to integer values at a node, no additional partitioning of this node can take place and the global optimum solution of the nonconvex NLP is required to guarantee convergence of the GMIN-αBB. Based on these conditions, the αBB algorithm can be used as as subroutine to generate valid lower bounds:

- If at least one integer variable can be relaxed at the current node, run the αBB algorithm for a few iterations to obtain a valid lower bound on the global solution of the continuous relaxation *or* run the αBB algorithm to completion to obtain the global solution of the continuous relaxation.
- Otherwise, run the αBB algorithm to completion to obtain the global solution for the current node.

This strategy makes use of the convergence characteristics of the αBB algorithm to improve the performance of the GMIN-αBB algorithm. The rate of improvement of the lower bound on the global solution of a nonconvex NLP is usually very high at early iterations and then gradually tapers off. At later stages of an αBB run, the computationally expensive reduction of the gap between the bounds on the solution of the continuous relaxation does not result in a sufficiently significant increase in the lower bound to affect the performance of the GMIN-αBB algorithm and can therefore be bypassed.

Generation of a Valid Upper Bound

Because of the finite size of the branch and bound tree, it is not necessary to generate an upper bound on the nonconvex MINLP at each node in order to guarantee convergence of the GMIN-αBB algorithm. In the worst case, the integer variables are fixed at every node of the last level of the tree, and the solutions of the corresponding NLPs provide the upper bounds needed to identify the global optimum solution. However, upper bounds play a significant role in improving the convergence rate of the algorithm by allowing the fathoming of nodes whose lower bound is greater than the smallest upper bound and therefore reducing the final size of the branch and bound tree. An upper bound on the solution of a given node can be obtained in several ways. For example, if the solution of the continuous relaxation is integer-feasible, that is, all the relaxed integer variables have integer values at the solution, this solution is both a lower and an upper bound on the current node. If the αBB algorithm was run for only a few iterations and the relaxed integer variables are integer at the lower bound, they can be fixed to these integer values and the resulting nonconvex NLP can be solved locally to yield an upper bound on the solution of the node. Finally, a set of integer values satisfying the integer constraints can be used to construct a nonconvex NLP whose local solutions are upper bounds on the current node solution.

Variable Bound Updates

If the bounds on the integer variables at any given node can be tightened, the solution space can be significantly reduced due to the combinatorial nature of the problem. The allocation of computational resources for this purpose is therefore a potentially worthwhile investment. An optimization-based approach or an interval-based approach may be used to update the variable bounds. These approaches are similar to those developed for the αBB algorithm but they take advantage of the integrality of the variables. Thus, in the optimization approach, the lower or upper bound on variable y_i is improved by first relaxing the integer variables, and then solving the convex NLP

$$y^* = \begin{cases} \min \text{ or } \max_{\mathbf{x},\mathbf{y},\mathbf{w}} & y_i \\ \text{s.t.} & \breve{f}(\mathbf{x},\mathbf{y},\mathbf{w}) \le \overline{f}^* \\ & \mathbf{C}(\mathbf{x},\mathbf{y},\mathbf{w}) \\ & \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U] \\ & \mathbf{y} \in [\mathbf{y}^L, \mathbf{y}^U] \\ & \mathbf{w} \in [\mathbf{w}^L, \mathbf{w}^U] \end{cases} \tag{3}$$

where $\breve{f}(\mathbf{x},\mathbf{y},\mathbf{w})$ denotes the convex underestimator of objective function, $\overline{f}^*$ denotes the current best upper bound on the global optimum solution, $\mathbf{C}(\mathbf{x},\mathbf{y},\mathbf{w})$ denotes the set of convexified constraints, and $\mathbf{w}$ is the set of new variables introduced during the convexification/relaxation procedure. Finally, the improved lower or upper bound is obtained by setting $y_i^L = \lceil y^* \rceil$ or $y_i^U = \lfloor y^* \rfloor$.

In the interval-based approach, an iterative procedure is followed based on an interval test which provides sufficient conditions for the infeasibility of the original constraints and the 'bound improvement constraint' $f(\mathbf{x}, \mathbf{y}) \leq \overline{f}^*$, given the relaxed region $(\mathbf{x}, \mathbf{y}) \in [\mathbf{x}^L, \mathbf{x}^U] \times [\mathbf{y}^L, \mathbf{y}^U]$. This set of constraints defines a region denoted by F. The procedure to improve the lower (upper) bound on variable y_i is as follows:

```
PROCEDURE interval-based bound update()
    Set initial bounds L = y_i^L and U = y_i^U;
    Set iteration counter k = 0;
    Set maximum number of iterations K;
    DO k < K and L ≠ U
        Compute 'midpoint' M = ⌊(U + L)/2⌋;
        Set left region
            {(x, y) ∈ F : y_i ∈ [L, M]};
        Set right region
            {(x, y) ∈ F : y_i ∈ [M + 1, U]};
        Test interval feasibility of left(right) region;
        IF feasible,
            Set U = M (L = M);
        ELSE
            Test interval feasibility of right(left)
            region;
            IF feasible,
                Set L = M (U = M);
            ELSE
                IF k = 0,
                    RETURN(infeasible node);
                ELSE
                    Set L = U (U = L);
                    Set U = y_i^U (L = y_i^L);
        Set k = k + 1;
    OD;
    RETURN(y_i^L = L (y_i^U = U));
END interval-based bound update;
```

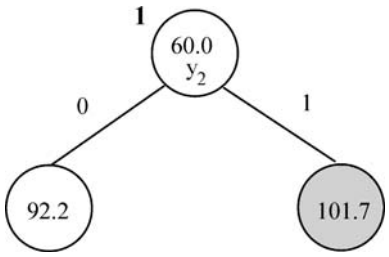
Interval-based bound update procedure

The variable bound tightening is performed before calling the αBB algorithm to obtain a lower bound on the solution of the current node. In many cases, during an αBB run, variable bound updates are also used to improve the quality of the generated lower bounds. Although the αBB algorithm treats the **y** variables as continuous, the bound update strategy within the αBB algorithm may be modified to account for the true nature of these variables. A larger reduction in the solution space can be achieved by adopting one of the *integer* bound update strategies described here for the relaxed **y** variables. This more stringent approach leads to a lower bound which is not necessarily a valid lower bound on the continuous relaxation, but which is always a lower bound on the global solution of the nonconvex MINLP.

The overall algorithmic procedure for the GMIN-αBB algorithm is shown below:

```
PROCEDURE GMIN-αBB algorithm()
    Set tolerance ε;
    Set f̲* = f̲^0 = −∞ and f̄* = f̄^0 = +∞;
    Initialize list of lower bounds {f̲^0};
    DO f̄* − f̲* > ε
        Select node k with smallest lower bound, f̲^k,
        from list of lower bounds;
        Set f̲* = f̲^k;
        (Optional) Update y variable bounds;
        Select integer branching variable(s);
        Create new nodes by branching;
        DO for each new node i
            Obtain lower bound f̲^i on node
                IF all integer variables are fixed,
                    Find global solution f̲^i of nonconvex
                    NLP with αBB algorithm;
                ELSE
                    Relax integer variables;
                    Run αBB algorithm to completion or
                    for a few iterations to get f̲^i
                        (Optional) Use integer bound
                        updates on y variables;
            IF f̲^i > f̄* + ε, THEN Fathom node;
            ELSE
                Add f̲^i to list of lower bounds;
                (Optional) Obtain upper bound f̄^i on
                nonconvex MINLP;
                IF f̄^i < f̄* THEN Set f̄* = f̄^i;
        OD;
    OD;
    RETURN(f̄* and variables values at corresponding node);
END GMIN-αBB algorithm;
```

Pseudocode for the GMIN-αBB algorithm



MINLP: Global Optimization with αBB, Figure 2
GMIN-αBB branch and bound tree

The algorithmic procedure for the GMIN-αBB algorithm is illustrated using the same example as for the SMIN-αBB algorithm. The branch and bound tree is shown in Fig. 2, using the same notation as previously.

At the first node, the continuous relaxation of the nonconvex MINLP is solved for 10 αBB iterations to yield a lower bound of 60. No upper bound is found. Next, the binary variable y_2 is chosen for branching and the continuous relaxation of the problem with $y_2 = 0$ is solved. A lower bound of 92.2 is found as the global solution to this nonconvex NLP. In addition, this solution is integer feasible and therefore provides an upper bound on the global optimum solution of the nonconvex MINLP. The region $y_2 = 1$ is then examined and the global solution of the NLP is found to be 101.7 after 10 αBB iterations. This node can therefore be fathomed and the procedure terminated.

The GMIN-αBB algorithm has been used to solve nonconvex MINLPs involving nonconvex terms in the integer variables and some mixed nonconvex terms. Branching priorities combined with variable bound updates and a small number of αBB iterations for relaxed nodes allow the identification of the global optimum solution after the exploration of a small fraction of the maximum number of nodes and with small CPU requirements. In particular, the algorithm has been used on a pump network synthesis problem [2,3]. Some nonconvex integer problems have also been tackled by the same approach. For instance, the minimization of trim loss, a problem taken from the paper cutting industry, has also been addressed for medium order sizes [3].

Conclusions

The αBB algorithm for nonconvex NLPs can be incorporated within more general frameworks to address broad classes of nonconvex MINLPs. One extension of the algorithm is the SMIN-αBB algorithm which identifies the global optimum solution of problems in which binary variables participate in linear or mixed-bilinear terms and continuous variables appear in twice continuously differentiable functions. The partitioning of the solution space takes place in both the continuous and binary domains. The GMIN-αBB algorithm is designed to locate the global optimum solution of problems involving integer and continuous variables in functions whose continuous relaxation is twice continuously differentiable. The algorithm is similar to traditional branch and bound algorithms for mixed integer problems in that branching occurs on the integer variables only and a continuous relaxation of the problem is constructed during the bounding step. It uses the αBB algorithm for the efficient and rigorous generation of lower bounds. Both algorithms are widely applicable and have been successfully tested on a variety of medium-size nonconvex MINLPs.

See also

- αBB Algorithm
- Chemical Process Planning
- Continuous Global Optimization: Models, Algorithms and Software
- Convex Envelopes in Optimization Problems
- Disjunctive Programming
- Extended Cutting Plane Algorithm
- Generalized Benders Decomposition
- Generalized Outer Approximation
- Global Optimization in Batch Design Under Uncertainty
- Global Optimization in Generalized Geometric Programming
- Global Optimization of Heat Exchanger Networks
- Global Optimization Methods for Systems of Nonlinear Equations
- Global Optimization in Phase and Chemical Reaction Equilibrium
- Interval Global Optimization
- MINLP: Application in Facility Location-allocation
- MINLP: Applications in Blending and Pooling Problems
- MINLP: Applications in the Interaction of Design and Control

- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Mass and Heat Exchanger Networks
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Heat Exchanger Network Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming
- ▶ Reformulation-linearization Methods for Global Optimization
- ▶ Smooth Nonlinear Nonconvex Optimization

References

1. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP problems in process synthesis. Comput Chem Eng 21:S445–S450
2. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB, for general twice-differentiable constrained NLPs -II. Implementation and computational results. Comput Chem Eng 22:1159
3. Adjiman CS, Androulakis IP, Floudas CA (2000) Global optimization of mixed-integer nonlinear problems. Comput Chem Eng 46:1769–1797
4. Adjiman CS, Androulakis IP, Maranas CD, Floudas CA (1996) A global optimization method, αBB, for process design. Comput Chem Eng 20:S419–S424
5. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB, for general twice-differentiable constrained NLPs -I. Theoretical advances. Comput Chem Eng 22:1137
6. Adjiman CS, Floudas CA (1996) Rigorous convex underestimators for twice-differentiable problems. J Global Optim 9:23–40
7. Adjiman CS, Schweiger CA, Floudas CA (1998) Mixed-integer nonlinear optimization in process synthesis. In: Du D-Z, Pardalos PM (eds) Handbook Combinatorial Optim. Kluwer, Dordrecht, pp 429–452
8. Androulakis IP, Maranas CD, Floudas CA (1995) αBB: A global optimization method for general constrained nonconvex problems. J Global Optim 7:337–363
9. Beale EML (1977) Integer programming. In: The State of the Art in Numerical Analysis. Acad. Press, New York, pp 409–448
10. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numer Math 4:238
11. Borchers B, Mitchell JE (1991) An improved branch and bound algorithm for mixed integer nonlinear programs. Techn. Report Renssellaer Polytechnic Inst. 200
12. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
13. Floudas CA (1995) Nonlinear and mixed integer optimization: Fundamentals and applications. Oxford Univ. Press, Oxford
14. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
15. Gupta OK, Ravindran R (1985) Branch and bound experiments in convex nonlinear integer programing. Managem Sci 31:1533–1546
16. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flow sheets. Industr Eng Chem Res 26:1869
17. Kocis GR, Grossmann IE (1989) A modelling and decomposition strategy for the MINLP optimization of process flowsheets. Comput Chem Eng 13:797–819
18. Maranas CD, Floudas CA (1994) Global minimum potential energy conformations of small molecules. J Global Optim 4:135–170
19. Ostrovsky GM, Ostrovsky MG, Mikhailow GW (1990) Discrete optimization of chemical processes. Comput Chem Eng 14:111
20. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput Chem Eng 16:937–947

MINLP: Heat Exchanger Network Synthesis

KEMAL SAHIN, KORHAN GÜRSOY, AMY CIRIC
Department Chemical Engineering,
University Cincinnati, Cincinnati, USA

MSC2000: 90C90

Article Outline

Keywords
Problem Statement
Heat Exchanger Network Superstructures
Mathematical Models
  for HEN Synthesis using MINLPs
  Match-Network Problem
  Heat Exchanger Network Synthesis
    Without Decomposition

Conclusions
See also
References

Keywords

MINLP; HEN synthesis; Network synthesis

Heat exchanger network synthesis problems arise in chemical process design when the heat released by hot process streams is used to satisfy the demands of cold process streams. These problems have been the subject of an intensive research effort, and over 400 publications have been written in the area. See [7,8,9] for reviews of the area, and [1,3] for detailed analysis of HEN synthesis.

The discovery by T. Umeda et al. [11] of a thermodynamic pinch point that limits heat integration in a heat exchanger network led to much of this research effort. They showed that setting minimum temperature approach, $\Delta T_{\min}$, places a lower bound on the utility consumption in a heat exchanger network and decomposed a heat exchanger network into independent subnetworks. This enables the heat exchanger network synthesis problem to be decomposed into four subproblems. The first subproblem finds the appropriate minimum temperature approach, the second subproblem minimizes the utility consumption, the third subproblem finds the minimum number of matches and identifies the matches and their heat duty, and the fourth finds and optimizes the actual network structure.

See [5] for a systematic scheme for solving these problems sequentially. First, the utility consumption is minimized using the linear programming (LP) transshipment model approach of [10]. Second, a set of process matches and their heat duties that minimize the total number of units is found with the mixed integer linear programming (MILP) strategy of [10]. Then, the network structure is found [5] by optimizing a superstructure that contains all possible network configurations embedded within it using a nonlinear programming (NLP) problem. When there is more than one combination of matches and heat duties that satisfies the minimum unit criterion, the best combination is found by exhaustive enumeration. The minimum temperature approach is optimized with a golden section search that solves all three of these optimization problems at each iteration.

In the late 1980s it was found, [4,12], that better network designs could be obtained by solving some of the heat exchanger network design subproblems simultaneously. C.A. Floudas and A.R. Ciric [4] combined the MILP stream matching problem with the NLP superstructure optimization problem formulated in [5], creating a mixed integer nonlinear programming problem (MINLP) that avoided the exhaustive search through all combinations of matches that minimize the number of units. In 1990, they [2] formulated the entire heat exchanger network design problem as a MINLP. The solution of this problem yields the optimal temperature approach, utility level, process matches, heat duties, and network structure, eliminating the need for a global section search for the optimum minimum temperature approach.

T.F. Yee and I.E. Grossmann [12] used a smaller superstructure proposed in [6] that embodies a sequential-parallel network structure to formulate an alternative MINLP for heat exchanger network synthesis. The solution of this MINLP yielded the utility consumption, matches and network structure and heat exchanger areas.

Problem Statement

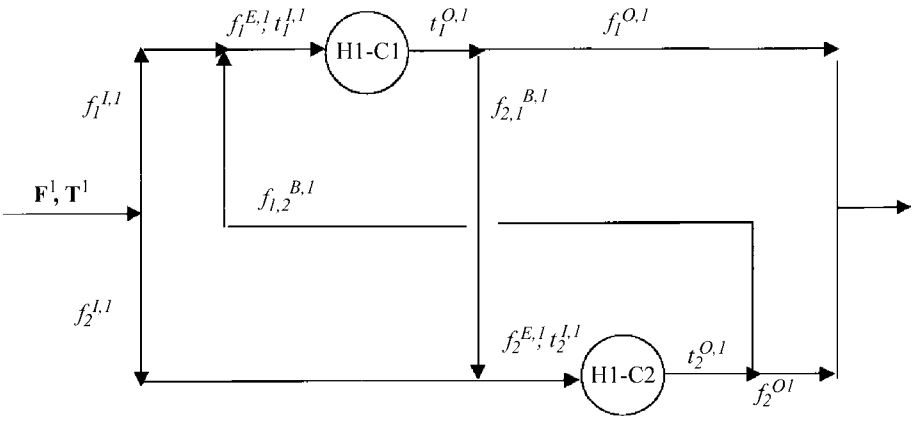
This article will explore two mixed integer nonlinear programming problems in heat exchanger network synthesis: combined match-network optimization and heat exchanger network synthesis without decomposition. The synthesis without decomposition problem can be stated as follows:

Given:

1) A set of hot process streams and hot utilities $i \in H$, their inlet and outlet temperatures $\mathbf{T}^i$, $\mathbf{T}^{O,i}$, and heat capacity flow rates $\mathbf{F}^i$;
2) A set of cold process streams and cold utilities $j \in C$, their inlet and outlet temperatures $\mathbf{T}^j$, $\mathbf{T}^{O,j}$, and heat capacity flow rates $\mathbf{F}^j$; and
3) Overall heat transfer coefficients $\mathbf{U}_{ij}$.

Determine:

A) The stream matches (ij), the heat duty $\mathbf{Q}_{ij}$ of match (ij), and the heat exchanger area $\mathbf{A}_{ij}$ of match (ij);
B) the piping structure for each stream in the network; and
C) the temperature and flowrate within each pipe of the network.



MINLP: Heat Exchanger Network Synthesis, Figure 1
A superstructure for one hot stream exchanging heat with two cold streams

In the match-network problem, one is also given

- the level of each utility; and
- a minimum temperature approach $\Delta T_{\min}$.

These problems can be solved using mixed integer nonlinear programming. The development and application of these approaches is described in more detail below.

Heat Exchanger Network Superstructures

Mixed integer nonlinear programming approaches to these problems begin with a superstructure that contains many alternative designs embedded within it. Two superstructures are particularly interesting.

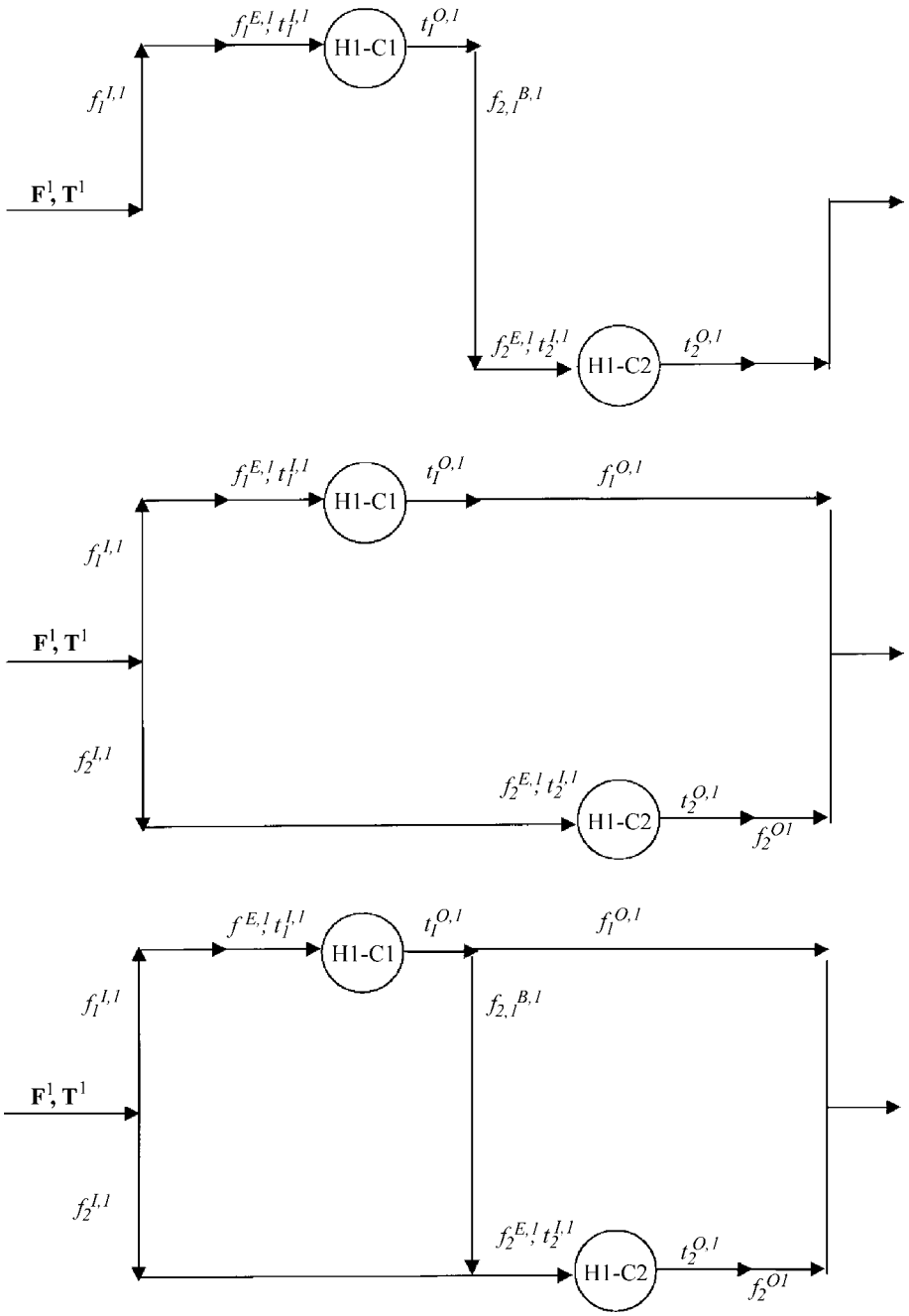
Figure 1 shows a superstructure of a hot stream, above the thermodynamic pinch point, that may exchange heat with two cold streams [5]. Notice that the stream can be piped in series, in parallel, and in split-mix-bypass configurations, as shown in Fig. 2. As we shall see, this richness leads to nonconvex constraints in the MINLP. The first network superstructure is created by constructing similar structures for every other stream above the pinch point.

Notice that in this subnetwork, streams H1 and C1, and all other pairs of hot and cold streams, can exchange heat no more than once. H1 and C1 may exchange heat again in the subnetwork below the thermodynamic pinch point. The thermodynamic pinch point has partitioned the temperature range into two intervals, and in each interval, individual process streams can only exchange heat once.

One could increase the number of times two streams can exchange heat by partitioning the temperature range further. This is the basic strategy behind the second superstructure [6,12] shown in Fig. 3. Here, the temperature range has been partitioned into many intervals, or stages. Within any particular stage, each hot stream may exchange heat with each cold stream; multiple intervals allow any particular match to take place many times in the network. Unlike the first superstructure, each stream in each stage is piped in a parallel configuration, and the inlet and outlet temperature of each parallel line is fixed by the temperature interval. Series piping structures arise when a stream exchanges heat only once per interval. The superstructure does not contain split-mix-bypass or series-parallel structures, but as we shall see that in exchange the nonconvex constraints that arise from the first superstructure have been eliminated.

Mathematical Models for HEN Synthesis using MINLPs

MINLP models of heat exchanger network synthesis arise when the process stream matches are selected while simultaneously optimizing the heat exchanger network; the former is a discrete decision modeled with integer variables, the latter, a nonlinear optimization problem. In this paper, we refer to this as the *match-network problem*. MINLPs may also be used to formulate an optimization problem that simultaneously minimizes the utility consumption, selects the stream matches, and optimizes the network layout, in *heat exchanger network synthesis without decomposition*.

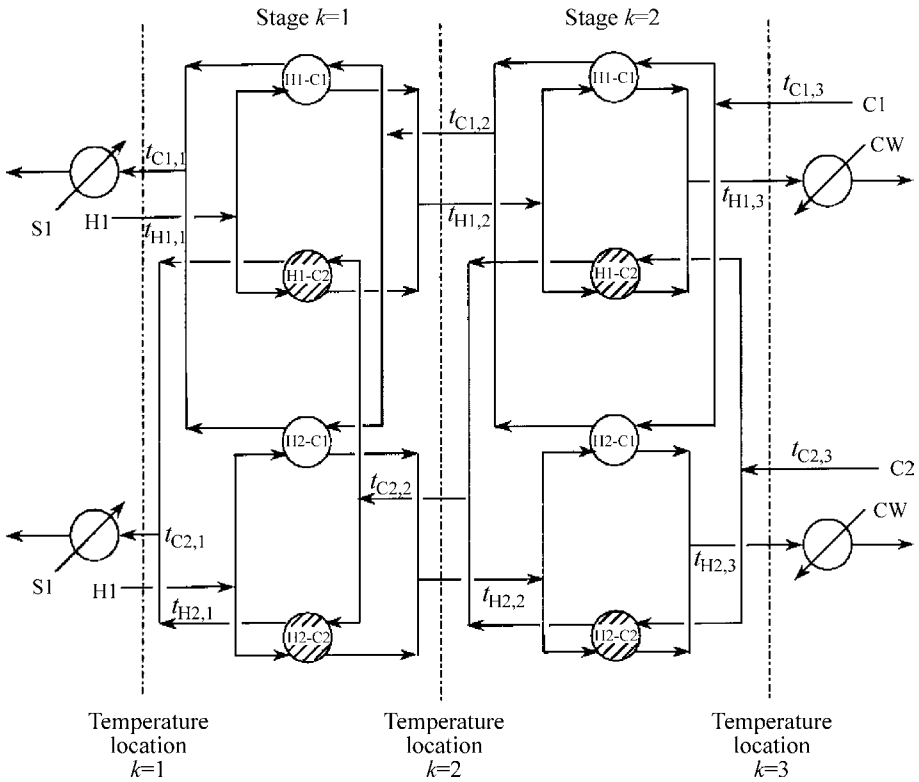


MINLP: Heat Exchanger Network Synthesis, Figure 2
Stream piping configurations embedded in the superstructure shown in Fig. 1

Match-Network Problem

The MINLP model of the match-network problem has three components: a *transshipment model* [10] that identifies feasible process stream matches and their heat duties, a *superstructure model* of all possible network structures, and an *objective function*.

The transshipment model partitions the temperature range into $t = 1, \ldots, T$ temperature intervals, using the inlet and outlet temperatures of the streams and



MINLP: Heat Exchanger Network Synthesis, Figure 3
Two-stage superstructure

the temperature interval approach temperature (TIAT). Hot streams release heat into the temperature intervals, where it either flows to the cold streams in the same interval or cascades down to the next colder interval. The binary variable Y_{ij} denotes the existence of a match between hot stream i and cold stream j, where heat loads are q_{ij} and Q_{ij}, and heat residuals are R_k. The model is composed by the following constraints:

$$\begin{cases} \sum_{j\in C_t} q_{ijt} + R_{i,t} - R_{i,t-1} = Q_{it}^H, \quad i \in H, \quad j \in C_i, \\ \sum_{i\in R_j} q_{ijt} = Q_{jt}^C, j \in C, \quad t = 1, \ldots, T, \\ \sum_{t=1}^{T} q_{ijt} = Q_{ij}, \quad i \in H, \quad j \in C_i, \\ Q_{ij} - UY_{ij} \le 0, \quad i \in H, \quad j \in C_i, \\ \sum_{i\in H}\sum_{j\in C_i} Y_{ij} \le N_{\max}. \end{cases}$$

The first two constraints in the transshipment model are the energy balances for each temperature interval. The total heat load in a match is given by the third constraint. The fourth constraint bounds the heat load using the binary variable Y_{ij} and a large fixed constant U. The last constraint in the above model puts an upper bound on the number of existing matches, which is the maximum number of units.

The second part of the match-network synthesis model is the hyperstructure topology model, which consists of mass and energy balances for the mixers and splitters, feasibility constraints, utility load constraint and bounds on the flow rate heat-capacities.

Mass balances for the splitters at the inlet of the superstructure:

$$\sum_{k'} f_{k'}^{I,k} = F^k, \quad k \in \mathrm{HCT}.$$

Here, HCT is the set of all process streams and utilities. Mass balances for the mixers at the inlets of the exchangers:

$$f_{k'}^{I,k} + \sum_{k''} f_{k',k''}^{B,k} - f_{k'}^{E,k} = 0, \quad k', k \in \mathrm{HCT}.$$

Mass balances for the splitters at the outlets of the exchangers:

$$f_{k'}^{O,k} + \sum_{k''} f_{k'',k'}^{B,k} - f_{k'}^{E,k} = 0, \quad k', k \in \text{HCT}.$$

Energy balances for the mixers at the inlets of the exchangers:

$$T^k f_{k'}^{I,k} + \sum_{k''} f_{k',k''}^{B,k} t_{k'}^{O,k} - f_{k'}^{E,k} t_{k'}^{I,k} = 0,$$

$$k', k \in \text{HCT}.$$

Energy balances over the heat exchangers:

$$Q_{ij} - f_j^{E,i}\left(t_j^{I,i} - t_j^{O,i}\right) = 0, \quad i \in H, \quad j \in C,$$

$$Q_{ij} - f_i^{E,j}\left(t_i^{O,j} - t_i^{I,j}\right) = 0.$$

The minimum temperature approach between a hot stream and a cold stream:

$$t_j^{I,i} - t_i^{O,j} \geq \Delta T_{\min},$$

$$t_j^{O,i} - t_i^{I,j} \geq \Delta T_{\min}.$$

Logical relations between the heat-capacity flow rates and the existence of a match:

$$f_j^{E,i} - F^i Y_{ij} \leq 0,$$

$$f_i^{E,j} - F^j Y_{ij} \leq 0.$$

Lower bounds on the heat-capacity flow rates through the exchanger:

$$f_j^{E,i} - \frac{Q_{ij}}{\Delta T_{ij,\max}} \geq 0,$$

$$f_i^{E,j} - \frac{Q_{ij}}{\Delta T_{ij,\max}} \geq 0,$$

where $\Delta T_{ij,\max}$ equals $T^i - T^j$. Lastly, the objective function minimizes the total investment cost:

$$\min \sum_{i \in H} \sum_{j \in C} \alpha \left(\frac{Q_{ij}}{U_{ij} \frac{t_j^{I,i} - t_i^{O,j} - t_j^{O,i} + t_i^{I,j}}{\ln \frac{t_j^{I,i} - t_i^{O,j}}{t_j^{O,i} - t_i^{I,j}}}} \right)^{\beta} Y_{ij}.$$

The model is a mixed integer nonlinear programming (MINLP) problem, as the objective function and the energy balances are nonlinear, and the decision variables Y_{ij} are binary. Notice that the energy balances are bilinear, creating a nonconvex feasible region.

MINLP: Heat Exchanger Network Synthesis, Table 1
Stream data for example problem

Stream	$T_{\text{in}}(C)$	$T_{\text{out}}(C)$	$FC_p(kW/C)$
H1	500	320	6
H2	480	380	4
H3	460	360	6
H4	380	360	20
H5	380	320	12
C1	290	660	18
F	700	700	
CW	300	320	

$U = 1.0kW/(m^2C)$
Annual cost=$1200A^{0.6}$ for all exchangers
$C_s = 140\$/kW$
$C_{cw} = 10\$/kW$

MINLP: Heat Exchanger Network Synthesis, Table 2
Match data for example problem; pseudo-pinch method [2]

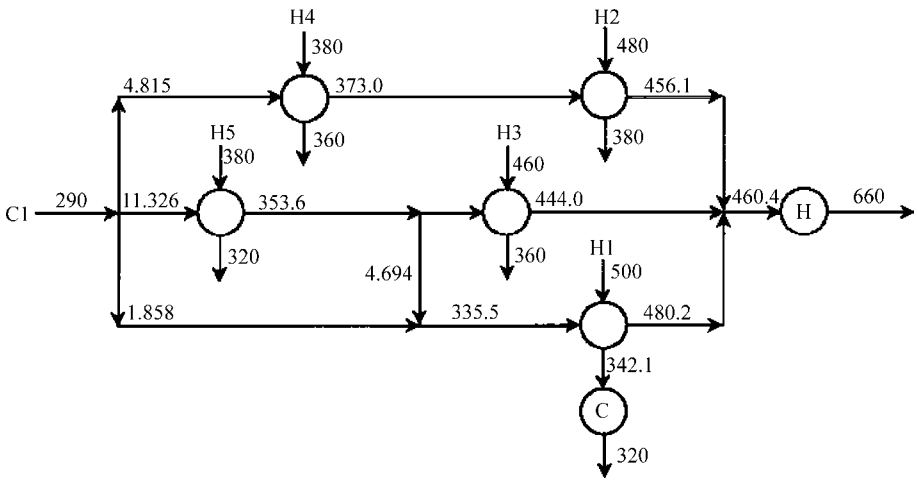
Match	$Q(kW)$	$A(m^2)$
H1-C1	948.454	79.391
H1-CW	131.546	6.280
H2-C1	400.000	29.057
H3-C1	600.000	57.488
H4-C1	400.000	14.880
H5-C1	720.000	25.509
S-C1	3591.546	32.112

Heat Exchanger Network Synthesis Without Decomposition

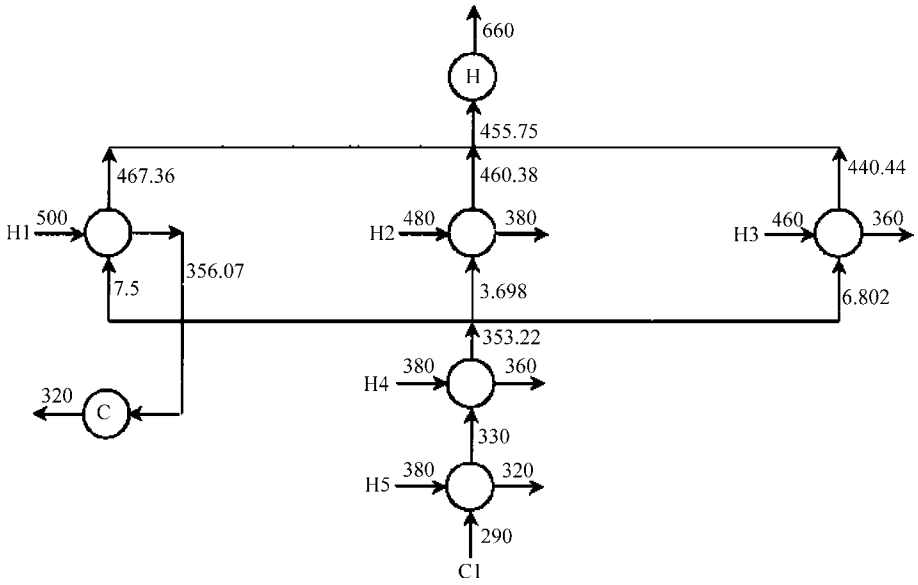
MINLP models that optimize utility consumption as well as process matches, heat duties, and network configurations can also be formulated. See [2] and [12] for pseudopinch approaches that set the TIAT to a small value and lets heat flow across the pinch. A strict decomposition at the pinch can also be maintained by letting TIAT vary, and using integer variables to model the changing structure of the temperature cascade.

Example 1 These techniques are demonstrated with a problem given in both [12] and [2]. The problem consists of two hot streams, two cold streams, one hot utility (steam), and one cold utility (cooling water). The stream data is given in Table 1.

Using the pseudopinch method with TIAT = 1C and $\Delta T_{\min} = 0.5C$, and allowing HRAT to vary between 1C



MINLP: Heat Exchanger Network Synthesis, Figure 4
Optimal network configuration for example problem; pseudopinch [2]



MINLP: Heat Exchanger Network Synthesis, Figure 5
Optimal network configuration for example problem; simultaneous approach [12]

and 30*C*, Ciric and Floudas [2] formulated the problem as a MINLP problem and solved it using the generalized Benders decomposition algorithm. The optimal network configuration is pictured in Fig. 3. The network consumes 3592.4*kW* of steam and 1312.4*kW* of cooling water, the HRAT is 8.42*C*. The annual cost of the network is \$571,080. The match data of this solution is given in Table 2. Yee and Grossmann [12] used the same problem to demonstrate the simultaneous optimization approach. The problem is again formulated as a MINLP problem. The optimal network configuration is given in Fig. 4. The annual cost of this network is \$576,640. HRAT is 13.1*C*. The match data of this network is given in Table 3.

Conclusions

Mixed integer nonlinear programming offer a powerful approach to heat exchanger network synthesis. Using these techniques, stream matching, the combina-

MINLP: Heat Exchanger Network Synthesis, Table 3
Match data for example problem; simultaneous approach [12]

Match	$Q(kW)$	$A(m^2)$
S-C1	3676.4	32.6
H1-C1	863.6	64.1
H2-C1	400.0	17.1
H3-C1	600.0	47.0
H4-C1	400.0	13.8
H1-CW	216.4	7.9
H5-C1	720.0	18.4

torial component of heat exchanger network synthesis, can be performed while simultaneously minimizing the utility consumption and selecting the cost-optimal heat exchanger network configuration. Merging these tasks leads to more cost-effective stream matches and lower exchanger costs.

See also

- Chemical Process Planning
- Extended Cutting Plane Algorithm
- Generalized Benders Decomposition
- Generalized Outer Approximation
- Global Optimization of Heat Exchanger Networks
- MINLP: Application in Facility Location-allocation
- MINLP: Applications in Blending and Pooling Problems
- MINLP: Applications in the Interaction of Design and Control
- MINLP: Branch and Bound Global Optimization Algorithm
- MINLP: Branch and Bound Methods
- MINLP: Design and Scheduling of Batch Processes
- MINLP: Generalized Cross Decomposition
- MINLP: Global Optimization with αBB
- MINLP: Logic-based Methods
- MINLP: Mass and Heat Exchanger Networks
- MINLP: Outer Approximation Algorithm
- MINLP: Reactive Distillation Column Synthesis
- Mixed Integer Linear Programming: Heat Exchanger Network Synthesis
- Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- Mixed Integer Nonlinear Programming

References

1. Biegler LT, Grossmann IE, Westerberg AW (1997) Systematic methods of chemical process design. Prentice-Hall, Englewood Cliffs, NJ
2. Ciric AR, Floudas CA (1990) Heat exchanger network synthesis without decomposition. Comput Chem Eng 15:385–396
3. Floudas CA (1995) Nonlinear and mixed-integer optimization. Oxford Univ. Press, Oxford
4. Floudas CA, Ciric AR (1989) Strategies for overcoming uncertainties in heat exchanger network synthesis. Comput Chem Eng 13(10):1133
5. Floudas CA, Ciric AR, Grossmann IE (1986) Automatic synthesis of optimum heat exchanger network configurations. AIChE J 32:276
6. Grossmann IE, Sargent RWH (1978) Optimum design of heat exchanger networks. Comput Chem Eng 2(1):1–7
7. Gundersen T, Naess L (1988) The synthesis of cost optimal heat exchanger networks: An industrial review of the state-of-the-art. Comput Chem Eng 12(6):503
8. Jezowski J (1994) Heat exchanger network grassroot and retrofit design: The review of the state-of-the-art: Part I. Hungarian J Industr Chem 22:279–294
9. Jezowski J (1994) Heat exchanger network grassroot and retrofit design: The review of the state-of-the-art: Part II. Hungarian J Industr Chem 22:295–308
10. Papoulias SA, Grossmann IE (1983) A structural optimization approach in process synthesis - II: Heat recovery networks. Comput Chem Eng 7:707
11. Umeda T, Harada T, Shiroko K (1979) A thermodynamic approach to the structure in chemical processes. Comput Chem Eng 3:373
12. Yee TF, Grossmann IE (1990) Simultaneous optimization models for heat integration - II: Heat exchanger network synthesis. Comput Chem Eng 14(10):1165

MINLP: Logic-based Methods

IGNACIO E. GROSSMANN
Carnegie Mellon University, Pittsburgh, USA

MSC2000: 90C10, 90C09, 90C11

Article Outline

Keywords
See also
References

Keywords

Generalized disjunctive programming; Disjunctive programming; Outer approximation method; generalized Benders decomposition; Mixed integer programming

There has been an increasing trend to representing linear and nonlinear discrete optimization problems by models consisting of algebraic constraints, logic disjunctions and logic relations ([1,7,8]). For instance, a mixed integer program can be formulated as a generalized disjunctive program as has been shown in [5]:

$$\text{(DP1)}\quad \begin{cases} \min & Z = \sum_k c_k + f(x) \\ \text{s.t.} & g(x) \le 0 \\ & \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{bmatrix}, \\ & k \in SD, \\ & \Omega(Y) = \text{true} \\ & x \in \mathbb{R}^n, \quad c \in \mathbb{R}^m, \\ & Y \in \{\text{true}, \text{false}\}^m, \end{cases}$$

in which Y_{ik} are the boolean variables that establish whether a given term in a disjunction is true ($h_{ik}(x) \le 0$), while $\Omega(Y)$ are logical relations assumed to be in the form of propositional logic involving only the boolean variables. Y_{ik} are auxiliary variables that control the part of the feasible space in which the continuous variables, x, lie, and the variables c_{ik} represent fixed charges which are set to a value γ_{ik} if the corresponding term of the disjunction is true. Finally, the logical conditions, $\Omega(Y)$, express relationships between the disjunctive sets. In the context of optimal synthesis of process networks, the disjunctions in (DP1) typically arise for each unit i in the following form:

$$\begin{bmatrix} Y_i \\ h_i(x) \le 0 \\ c_i = \gamma_i \end{bmatrix} \bigvee \begin{bmatrix} \neg Y_i \\ B^i x = 0 \\ c_i = 0 \end{bmatrix}, \quad i \in I, \tag{1}$$

in which the inequalities h_i apply and a fixed cost γ_i is incurred if the unit is selected (Y_i); otherwise ($\neg Y_i$) there is no fixed cost and a subset of the x variables is set to zero with the matrix B^i. An important advantage of the above modeling framework is that there is no need to introduce artificial parameters for the 'big-M' constraints that are normally used in MINLP to model disjunctions.

M. Turkay and I.E. Grossmann [9] proposed a logic version of the outer approximation algorithm for MINLP [3] for solving problem (DP1), and in which the disjunctions are given as in equation (1), and all the functions are assumed to be convex. The algorithm consists of solving a sequence of NLP subproblems and master problems, which are as follows.

For fixed values of the boolean variables, $Y_{\widehat{i}k} = \text{true}$ and $Y_{ik} = \text{false}$ for $\widehat{i} \neq i$, the corresponding NLP subproblem is as follows:

$$\text{(NLPD)}\quad \begin{cases} \min & Z = \sum_k c_k + f(x) \\ \text{s.t.} & g(x) \le 0 \\ & \text{for } Y_{\widehat{i}k} = \text{true}: \begin{cases} h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{cases} \\ & \text{for } Y_{ik} = \text{false}: \begin{cases} B^i x = 0 \\ c_k = 0 \end{cases} \\ & k \in SD, \\ & x \in \mathbb{R}^n, \quad c_i \in \mathbb{R}^1. \end{cases}$$

Note that for every disjunction $k \in SD$ only constraints corresponding to the boolean variable $Y_{\widehat{i}k}$ that is true are imposed. Also, fixed charges γ_{ik} are only applied to these terms. Assuming that K subproblems (NLPD) are solved in which sets of linearizations $l = 1, \ldots, K$ are generated for subsets of disjunction terms $L(ik) = \{l: Y^l_{ik} = \text{true}\}$, one can define the following *disjunctive OA master problem*:

$$\text{(MDP1)}\quad \min Z = \sum_k c_k + f(x)$$

such that

$$\begin{aligned} & \alpha \ge f(x^l) + \nabla f(x^l)^\top (x - x^l), \\ & g(x^l) + \nabla g(x^l)^\top (x - x^l) \le 0, \\ & l = 1, \ldots, L, \\ & \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x^\ell) + \nabla h_{ik}(x^\ell)^\top (x - x^\ell) \le 0 \\ c_k = \gamma_{ik} \end{bmatrix}, \end{aligned}$$

$$k \in SD,$$
$$\Omega(Y) = \text{true},$$
$$\alpha \in \mathbb{R}, \quad x \in \mathbb{R}^n, \quad c \in \mathbb{R}^m,$$
$$Y \in \{\text{true}, \text{false}\}^m.$$

It should be noted that before applying the above master problem it is necessary to solve various subproblems (NLPD) so as to produce at least one linear approximation of each of the terms in the disjunctions. Selecting the smallest number of subproblems amounts to the solution of a set covering problem, which is of small size and easy to solve [9].

The above problem (MDP1) can be solved by the methods described in [1] and [7]. It is also interesting to note that for the case of process networks Turkay and Grossmann [9] have shown that if the convex hull representation of the disjunctions in (1) is used in (MDP1), then assuming $B^i = I$ and converting the logic relations $\Omega(Y)$ into the inequalities $Ay \le a$, leads to the MILP problem,

$$\text{(MIPDF)} \quad \min Z = \sum_k c_k + f(x)$$

such that

$$\alpha \ge f(x^l) + \nabla f(x^l)^\top (x - x^l),$$
$$g(x^l) + \nabla g(x^l)^\top (x - x^l) \le 0,$$
$$l = 1, \ldots, L,$$
$$\nabla_{x_{z_i}} h_i(x^l)^\top x_{z_i} + \nabla_{x_{N_i}} h_i(x^l)^\top x^1_{N_i}$$
$$\le \left[-h_i(x^\ell) + \nabla_x h_i(x^\ell)^\top x^\ell\right] y_i,$$
$$\ell \in K^i_L, \quad i \in I,$$
$$x_{N_i} = x^1_{N_i} + x^2_{N_i},$$
$$0 \le x^1_{N_i} \le x^U_{N_i} y_i,$$
$$0 \le x^2_{N_i} \le x^U_{N_i}(1 - y_i),$$
$$Ay \le a,$$
$$x \in \mathbb{R}^n, \quad x^1_{N_i} \ge 0, x^2_{N_i} \ge 0,$$
$$y\{0, 1\}^m,$$

where the vector x is partitioned into the variables for each disjunction i according to the definition of the matrix B^i. The linearization set is given by $K^i_L = \{\ell\colon Y\ell_i = \text{true}, \ell = 1, \ldots, L\}$ that denotes the fact that only a subset of inequalities were enforced for a given subproblem ℓ. It is interesting to note that the logic-based outer approximation algorithm represents a generalization of the modeling/decomposition strategy [5] for the synthesis of process flowsheets.

Turkay and Grossmann [9] have also shown that while a logic-based *generalized Benders method* [4] cannot be derived as in the case of the OA algorithm, one can exploit the property for MINLP problems that performing one Benders iteration [2] on the MILP master problem of the OA algorithm, is equivalent to generating a generalized Benders cut. Therefore, a logic-based version of the generalized Benders method consists of performing one Benders iteration on the MILP master problem (MIPDF). It should also be noted that slacks can be introduced to (MDP1) and to (MIPDF) to reduce the effect of nonconvexities as in the augmented-penalty MILP master problem [10].

Finally, it should be noted that S. Lee and Grossmann [6] have developed a new branch and bound method and a MINLP reformulation that is based on the convex hull of each of the disjunctions in (DP1) with nonlinear inequalities.

See also

- ▶ Chemical Process Planning
- ▶ Decomposition Principle of Linear Programming
- ▶ Disjunctive Programming
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming

- ▶ Reformulation-linearization Methods for Global Optimization
- ▶ Simplicial Decomposition
- ▶ Simplicial Decomposition Algorithms
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Successive Quadratic Programming: Decomposition Methods

References

1. Beaumont N (1991) An algorithm for disjunctive programs. Europ J Oper Res 48:362–371
2. Benders JF (1982) Partitioning procedures for solving mixed-variables programming problems. Numer Math 4:238–252
3. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307
4. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10(4):237–260
5. Kocis GR, Grossmann IE (1989) A modeling and decomposition strategy for the MINLP optimization of process flowsheets. Comput Chem Eng 13:797
6. Lee S, Grossmann IE (2000) New algorithms for nonlinear generalized disjunctive programming. Comput Chem Eng 24:2125
7. Raman R, Grossmann IE (1993) Symbolic integration of logic in mixed integer linear programming techniques for process synthesis. Comput Chem Eng 17:909
8. Raman R, Grossmann IE (1994) Modelling and computational techniques for logic based integer programming. Comput Chem Eng 18:563
9. Turkay M, Grossmann IE (1996) A logic based outer-approximation algorithm for MINLP optimization of process flowsheets. Comput Chem Eng 20:959–978
10. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer-approximation method for MINLP optimization. Comput Chem Eng 14:769

MINLP: Mass and Heat Exchanger Networks

MEN, MHEN

KATERINA P. PAPALEXANDRI
bp Upstream Technol., Sunbury-on-Thames, Middlesex, UK

MSC2000: 93A30, 93B50

Article Outline

Keywords
MEN Superstructure
Modeling Mass Exchange
Minimizing Network Cost
See also
References

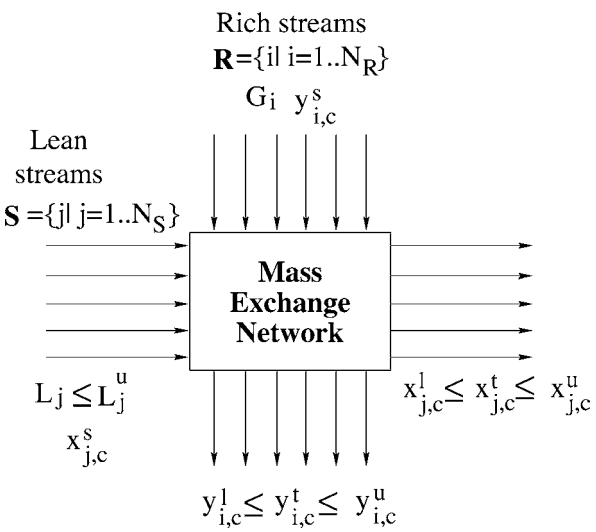
Keywords

MINLP; Mass and heat exchange; Separation

Mass integration in the form of *mass exchanger networks*, MEN, appears in the chemical industries as an economic alternative in waste treatment, feed preparation, product separation, recovery of valuable materials, etc. The MEN involves a set of rich streams, wherefrom one or more components are removed by means of lean streams (mass separating agents) in mass transfer operations that do not require energy (constant pressure and temperature).

The MEN synthesis/design problem is posed as a combinatorial problem, involving discrete and continuous decisions (e. g. the mass exchange operations/matches and the unit sizes, respectively), that both affect the overall mass integration cost.

When the mass transfer operations can take place at different temperatures, heat integration of the rich and



MINLP: Mass and Heat Exchanger Networks, Figure 1

lean streams is also considered within a combined *mass and heat exchanger network*, MHEN, synthesis problem.

In isothermal MEN synthesis, the simultaneous optimization of the mass exchange operations, the mass separating agent flows and the network configuration has been formulated by K.P. Papalexandri, E.N. Pistikopoulos and C.A. Floudas [9] as an MINLP problem based on:

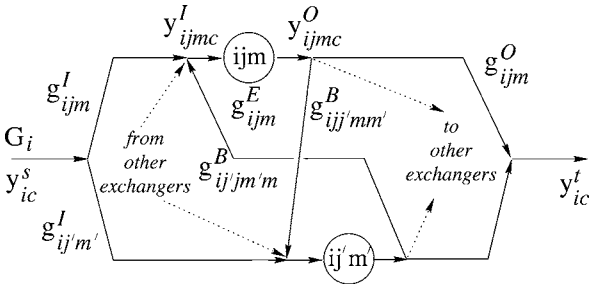
a) the MEN superstructure of synthesis/design alternatives;
b) modeling of mass exchange in each mass exchanger; and,
c) minimization of a total annualized network cost.

Details are given below

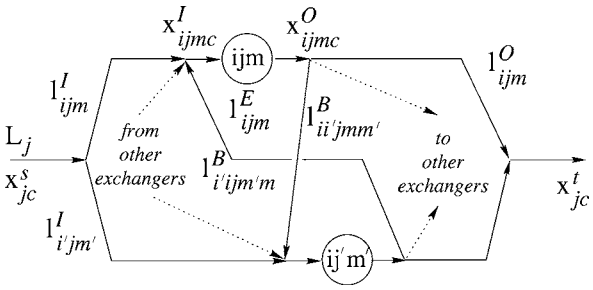
MEN Superstructure

The *MEN superstructure* for a given set of rich and lean streams includes all possible mass exchange operations (*mass exchange matches*) between the network streams in all possible network configurations. Its main features are:

- Each potential match between a rich and a lean stream corresponds to a potential mass exchanger (one-to-one correspondence).
  Multiple mass exchange matches between two streams may be considered (i. e. streams integrated at different points in the network), increasing thus the considered MEN structures and the combinatorial complexity of the synthesis problem. Note that, this is not similar to an a priori decomposition of the network into separable subnetworks.
- Each stream entering the network is split towards all its potential mass exchanger units.
- After each mass exchanger, a splitter is considered for each stream, where the stream is split towards its final mixer and all the other potential stream exchangers.
- Prior to each potential mass exchanger, a mixer is considered for each participating stream, where the flow from the initial splitter and connecting (bypass) flows from all the other exchangers of the stream are merged into the flow towards the exchanger.
- A mixer is considered at the network outlet of each stream, where flows from all the potential stream exchangers are merged into the outlet flow.



MINLP: Mass and Heat Exchanger Networks, Figure 2
Rich stream superstructure



MINLP: Mass and Heat Exchanger Networks, Figure 3
Lean stream superstructure

For example, for a rich stream i and its mth and m'th possible exchangers with lean streams j and j' respectively, we have Fig. 2

In Fig. 2, $c = 1, \ldots, C$ are the transferable components. All possible configurations for the two exchangers ((ijm) and $(ij'm')$ in series, or in parallel), result by 'deleting' appropriate connecting streams. Stream deletion corresponds to zero stream flows (e. g. $g^I_{ij'm'} = g^O_{ijm} = 0$ and $g^B_{ij'jm'm} = 0$ results in the exchangers in series).

For a lean stream j and its mth and m'th exchangers with rich streams i and i', we have Fig. 3.

The MEN superstructure is described by mass balances for the overall streams and each transferable component at the exchangers, splitters and mixers of the superstructure:

$$\begin{cases} g^E_{ijm}(y^I_{ijmc} - y^O_{ijmc}) = M_{ijmc}, \\ \quad i \in R,\ c = 1, \ldots, C, \\ l^E_{ijm}(x^O_{ijmc} - x^I_{ijmc}) = M_{ijmc}, \\ \quad i \in S,\ c = 1, \ldots, C, \end{cases} \tag{1}$$

$$\begin{cases} \sum_{j\in S,m} g^I_{ijm} - G_i = 0, \quad i \in R, \\ \sum_{i\in R,m} l^I_{ijm} - L_j = 0, \quad i \in S, \end{cases} \tag{2}$$

$$\begin{cases} g^O_{ijm} + \sum_{j'\in S,m'} g^B_{ijj'mm'} - g^E_{ijm} = 0, \\ l^O_{ijm} + \sum_{i'\in S,m'} l^B_{ii'jmm'} - l^E_{ijm} = 0, \\ i \in R,\ j \in S,\ m = 1,\ldots,M, \end{cases} \tag{3}$$

$$\begin{cases} g^I_{ijm} + \sum_{j'\in S,m'} g^B_{ij'jm'm} - g^E_{ijm} = 0, \\ l^I_{ijm} + \sum_{i'\in S,m'} l^B_{i'ijm'm} - l^E_{ijm} = 0, \\ i \in R,\ j \in S,\ m = 1,\ldots,M, \end{cases} \tag{4}$$

$$\begin{cases} g^I_{ijm} y^s_{ic} + \sum_{j'\in S,m'} g^B_{ij'jm'm} y^O_{ij'm'c} \\ \quad - g^E_{ijm} y^I_{ijmc} = 0, \\ l^I_{ijm} x^s_{jc} + \sum_{i'\in S,m'} l^B_{i'ijm'm} x^O_{i'jm'c} \\ \quad - l^E_{ijm} x^I_{ijmc} = 0, \\ \quad i \in R,\ j \in S, \\ c = 1,\ldots,C,\ m = 1,\ldots,M, \end{cases} \tag{5}$$

$$\begin{cases} \sum_{j\in S,m} g^O_{ijm} y^O_{ijmc} - G_i y^t_{ic} = 0, \\ \quad i \in R,\ c = 1,\ldots,C, \\ \sum_{i\in R,m} l^O_{ijm} x^O_{ijmc} - L_j x^t_{jc} = 0, \\ \quad i \in S,\ c = 1,\ldots,C, \end{cases} \tag{6}$$

where the inlet, outlet, exchanger and exchanger-connecting flows of the rich and lean streams (g^I, g^O, g^E, g^B and l^I, l^O, l^E, l^B, respectively) and the intermediate compositions of components (molar fractions x^I, x^O, y^I, y^O) are illustrated in the corresponding superstructure figures.

Modeling Mass Exchange

The existence of each potential mass exchanger in the network is denoted by a binary variable:

$$E_{ijm} = \begin{cases} 1, & \text{when the } m\text{th exchanger} \\ & \text{between streams } i \text{ and } j \text{ exists}, \\ 0, & \text{otherwise}, \end{cases}$$

and defined by

$$\begin{cases} g^E_{ijm} - E_{ijm} U \le 0, \\ l^E_{ijm} - E_{ijm} U \le 0, \\ M_{ijmc} - E_{ijm} U \le 0, \\ g^E_{ijm},\ l^E_{ijm},\ M_{ijmc} \ge 0, \end{cases} \tag{7}$$

where M_{ijmc} is the mass exchange load of component c in mass exchanger (ijm), and U a large positive number.

In each potential mass exchanger a component c is transferred from the rich to the lean stream when the rich composition is greater than the equilibrium composition with respect to the lean stream:

$$y_c \ge f(x_c),$$

where $f(x_c)$ is the mass transfer equilibrium relation, that may account for reactive mass transfer also.

Feasibility of mass transfer is ensured imposing the above constraint at the inlet and outlet of the streams, i. e. (for counter-current flows):

$$\begin{cases} -y^I_{ijmc} + f(x^O_{ijmc}) + \epsilon_{ijc} - (1 - E_{ijm})U \le 0, \\ -y^O_{ijmc} + f(x^I_{ijmc}) + \epsilon_{ijc} - (1 - E_{ijm})U \le 0, \end{cases} \tag{8}$$

where ϵ_{ijc} is a *minimum composition difference* that is required for feasible mass exchange in a unit of finite size (e. g. imposed from mechanical constraints). When $f(x_c)$ is not convex the constraints in (8) cannot guarantee feasible mass transfer throughout the exchanger. In this case $f(x_c)$ can be approximated by a set of convex functions and feasible mass transfer be ensured considering the constraints in (8) also for intermediate exchanger points, that define the convex parts. Note that, the mass-transfer feasibility or driving-force constraints in (8) are activated only when the corresponding exchanger exists ($E_{ijm} = 1$).

The size of each potential mass exchanger (number of mass transfer stages, N^{st}, etc.) is calculated as a function of the variable mass transfer, through appropriate design equations (e. g. for perforated-plate columns the *Kremser equation*):

$$N^{st}_{ijm} = N^{st}(g^E_{ijm}, l^E_{ijm}, x^I_{ijmc}, x^O_{ijmc}, y^I_{ijmc}, y^O_{ijmc}). \tag{9}$$

Minimizing Network Cost

The total network cost comprises

- the annualized capital cost of the mass exchangers, that may be discontinuous (involve a fixed charge cost factor), and
- the annualized operating cost, i. e. the cost of the mass separating agents.

Consequently, the MEN MINLP synthesis model is formulated as follows:

$$\text{(P1)} \quad \min \sum_{ijm} \left(AC^1_{ijm} E_{ijm} + AC^2_{ijm} (N^{st}_{ijm}) \right) + \sum_j AC^3_j L_j$$

such that

$$(2) - (9)$$
$$g^I_{ijm}, g^E_{ijm}, g^B_{ijj'mm'}, g^O_{ijm} \geq 0,$$
$$y^I_{ijmc}, y^O_{ijmc} \geq 0,$$
$$i \in R,\ j, j' \in S,$$
$$m, m' = 1, \ldots, M,$$
$$c = 1, \ldots, C,$$
$$l^I_{ijm}, l^E_{ijm}, l^B_{ii'jmm'}, l^O_{ijm} \geq 0,$$
$$x^I_{ijmc}, x^O_{ijmc} \geq 0,$$
$$i \in R,\ j, j' \in S,$$
$$m, m' = 1, \ldots, M,$$
$$c = 1, \ldots, C,$$
$$E_{ijm} = 0, 1,$$
$$i \in R,\ j, j' \in S,$$
$$m, m' = 1, \ldots, M.$$

(P1) is a nonconvex MINLP problem and global optimization methods are required to guarantee global optimal solutions.

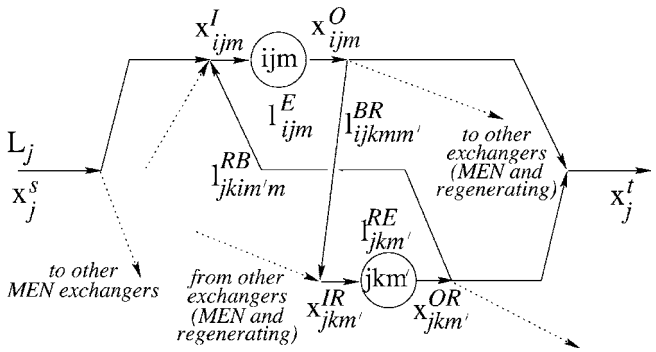
The main advantage of the simultaneous MEN synthesis model (P1), as opposed to the sequential MEN synthesis method, is that the trade-off between the capital and operating costs is systematically considered. Also,

- (P1) derives the optimal network with respect to all the transferable components, considering the mass transfer of each component separately within the calculated mass-transfer stages of each exchanger.
- Forbidden mass exchange matches, limited mass exchange and/or forbidden exchanger connections can be explicitly considered in (P1).
- Variable target compositions are straightforwardly handled.

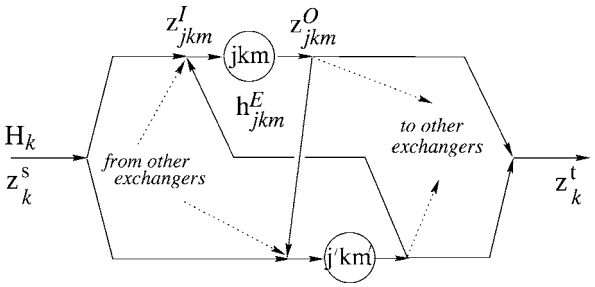
When the mass exchange matches and mass exchange loads are fixed (e. g. when these are determined within a sequential MEN synthesis framework), (P1) reduces to an NLP and can be solved to derive a network configuration and unit sizes with minimum capital cost.

Extending the concept of cost optimality of the mass exchanger network, two special cases have been studied:

- MEN and *regeneration networks*.
  When regenerating agents are available for some (or all) lean streams, the total mass integration cost involves also the regeneration cost. The regeneration network can be considered simultaneously within the MINLP MEN synthesis model [9], accounting for all the regeneration alternatives of the lean streams and employing binary variables to denote the existence of the regenerating exchangers. In this case, the mass separating agents behave as lean streams in the mass exchangers of the main MEN and as rich streams in the regenerating mass exchangers. The regeneration network is not necessarily separable from the main MEN, as a lean stream may be partly regenerated before being used as a separating agent in another mass exchanger. Thus, the lean stream superstructures involve all the possible interconnections between the exchangers of the main MEN and the regenerating exchangers. For example, for a lean stream j and its mth and m'th exchangers with rich stream i and regenerant k we have Fig. 4.
  The overall superstructure of mass exchange and regeneration alternatives involves also the superstructures of the regenerating agents, that have variable flows, while the overall network cost includes the main MEN and the regeneration cost (capital and operating cost).
- *Flexible mass exchange networks*.
  The ability of MEN to accommodate variations in the rich stream flows and inlet compositions in an efficient manner affects cost optimality. A *multiperiod MINLP MEN synthesis model* has been suggested in [7], to derive mass exchange networks, flexible to accommodate in an optimal manner different mass integration requirements. In the multiperiod MINLP model a weighted operating cost is optimized simultaneously with the capital cost



MINLP: Mass and Heat Exchanger Networks, Figure 4
Regenerable lean stream superstructure



MINLP: Mass and Heat Exchanger Networks, Figure 5
Regenerating stream superstructure

for mass exchangers that can operate feasibly under the different conditions. The MEN superstructure is extended to include control variables that enhance flexibility (as exchanger-bypassing streams and overall bypass streams that are accordingly penalized).

When the alternative mass transfer operations take place at different and/or variable temperatures, heat integration between the network streams can be simultaneously considered within a combined MEN and HEN synthesis problem [7]. The available rich and lean streams define hot, cold or hot-and-cold streams in the heat integration problem, depending on whether their supply and target compositions are above or below the mass exchange temperatures. Thus, their heat exchange alternatives include both hot- and cold-side matching. Inlet and outlet temperatures and compositions in mass and heat exchangers are variables. The combined mass and heat exchanger superstructure involves all the possible mass and heat exchangers of a stream and all the possible interconnections between them, Fig. 6.

The combined MEN and HEN superstructure is described by

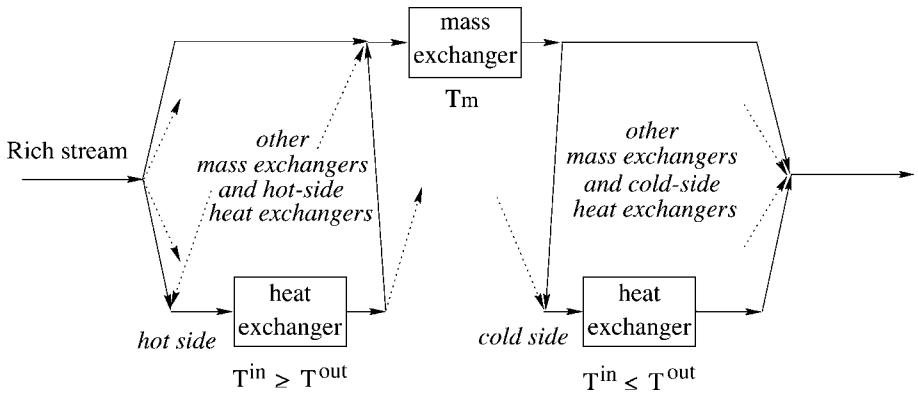
- mass balances at the superstructure splitters (i. e. the initial stream splitters and the splitters after each side of the possible mass and heat exchangers), similar to (2) and (3), and considering all the connecting flows;
- mass balances for overall flows and transferable components at the superstructure mixers (i. e. the final stream mixers and the mixers prior to each side of the potential mass and heat exchangers), similar to (4), (5) and (6), and considering all the connecting flows;
- energy balances at the superstructure mixers;
- mass balances at the mass exchangers, similar to (1), and
- energy balances at the heat exchangers.

The MHEN synthesis model also involves

- binary variables, to denote the existence of mass and heat exchangers, and their definition (mixed integer constraints),
- driving force constraints for mass exchange (8) at the potential mass exchangers, and for heat exchange at the potential heat exchangers (based on $\Delta T_{\min}$),
- design equations for the potential mass and heat exchangers, and
- a total annualized network cost.

and is formulated as a (nonconvex) MINLP.

The simultaneous MHEN synthesis model addresses systematically the trade-off between capital and operating cost of mass and heat integration. The MEN and HEN are not assumed separable. Thus, better integration can be achieved, as it is allowed for a stream to be partly heated for a particular mass exchange operation and then heated further for final purification.



MINLP: Mass and Heat Exchanger Networks, Figure 6
Combined MEN and HEN superstructure

In the simple case when the temperatures of the mass exchange operations are given or can be prepostulated, the rich and lean streams define hot (or cold) streams before participating to mass exchangers and cold (or hot) streams afterwards [11].

The final mass and heat exchanger network structure results from the flows of the superstructure substreams. Alternatively, the use of binary variables has been suggested in [7] to denote the existence of exchanger connections. This, although increasing the combinatorial complexity of the MINLP synthesis model, allows for:

i) explicit piping cost considerations,
ii) structural constraints to be easily modeled, and
iii) the solution of simple NLP subproblems within a decomposition-based MINLP solution method.

Mass exchange networks have been introduced as an end-of-pipe treatment alternative. However, the extent of mass recovery and the corresponding cost are closely related to the reactive and mixing operations in a process. A. Lakshmanan and L.T. Biegler [6] have suggested a MINLP model for the synthesis of optimal reactor networks, where the thermodynamic feasibility of mass integration and its implications are taken simultaneously into account, applying the first and second thermodynamic laws for mass exchange, i. e.

- Total mass balance for the mass integrated streams (resulting process and available rich and lean streams);

$$\sum_{i\in\mathbf{R}} G_i(y^s_{ic} - y^t_{ic}) = \sum_{j\in\mathbf{S}} L_j(x^t_{jc} - x^s_{jc}) \qquad (10)$$

and

- feasibility of mass exchange above (and below) each candidate mass exchange pinch:

$$\left\{\begin{array}{l}\text{Mass lost by all the rich}\\ \text{streams below each pinch}\\ \text{point candidate}\end{array}\right\} - \left\{\begin{array}{l}\text{Mass gained by all the lean}\\ \text{streams below each pinch}\\ \text{point candidate}\end{array}\right\} \le 0$$

i. e.

$$\begin{aligned}&\sum_{i\in\mathbf{R}} G_i \\ &\times \left[\max(0, y_p - y^t_{ic}) - \max(0, y_p - y^S_{ic})\right] \\ &- \sum_{j\in\mathbf{S}} L_j \\ &\times \left[\max(0, x_p - x^S_{jc}) - \max(0, x_p - x^t_{jc})\right] \\ &\le 0\end{aligned} \qquad (11)$$

Note that the thermodynamic feasibility requirements in (11) involve nondifferentiable terms if inlet and outlet compositions are variables (position of streams with respect to candidate pinch points). These can be handled either employing differentiable approximation functions [6], or introducing binary variables [2,3,5].

The main assumption in MEN is that mass transfer operations are isothermal. In the general case these can be followed (or caused) by heat transfer, as in distillation. Assuming constant counter-current molar flows, M.J. Bagajewicz and V. Manousiouthakis showed in [1] that distillation columns can be handled as pure

mass transfer operations and derived targets for energy consumption and separation of a 'key' component, employing the first and second thermodynamic laws in (10) and (11), within an MINLP-based MHEN sequential synthesis framework. The problem of energy-induced separations has been addressed by M.M. El-Halwagi, B.K. Srinivas and R.F. Dunn in [4], translating the energy-based separation tasks into simple energy-requiring operations (heating and cooling tasks) and deriving targets for energy consumption and the corresponding mass recovery, based on thermodynamic feasibility constraints.

Extending the concept of mass exchange to non-isothermal mass transfer operations Papalexandri and Pistikopoulos introduced a *mass/heat transfer module* [8], where mass is transferred between different phases or reacting species if that is thermodynamically feasible, i. e. if that decreases the total Gibbs free energy of the system. Mass and energy balances, taking into account possible reactions, and mass-transfer driving-force constraints based on total Gibbs free energy are employed to model the mass/heat transfer module as an aggregate of differential mass and energy transfer phenomena. Considering a superstructure of mass/heat and heat exchange modules in a process and all possible interconnections between them, process synthesis tasks can be formulated as mass/heat and heat exchange superstructure MINLP problems, where binary variables are employed to denote the existence of mass/heat and heat exchangers. Then, process operations (conventional and/or hybrid) and networks are derived as combinations of mass/heat and heat exchange phenomena [8,10].

See also

- ▶ Global Optimization of Heat Exchanger Networks
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ Mixed Integer Linear Programming: Heat Exchanger Network Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks

References

1. Bagajewicz MJ, Manousiouthakis V (1992) Mass/heat exchange network representation of distillation networks. AIChE J 38:1769–1800
2. El-Halwagi MM, Manousiouthakis V (1990) Simultaneous synthesis of mass-exchange and regeneration networks. AIChE J 36:1209–1219
3. El-Halwagi MM, Srinivas BK (1992) Synthesis of reactive mass exchange networks. Chem Eng Sci 47:2113–2119
4. El-Halwagi MM, Srinivas BK, Dunn RF (1995) Synthesis of optimal heat-induced separation networks. Chem Eng Sci 50:81–97
5. Gupta A, Manousiouthakis V (1993) Minimum utility cost of mass exchange networks with variable single component supplies and targets. Industr Eng Chem Res 32:1937–1950
6. Lakshmanan A, Biegler LT (1996) Synthesis of optimal chemical reactor networks with simultaneous mass integration. Industr Eng Chem Res 35:4523–4536
7. Papalexandri KP, Pistikopoulos EN (1994) A multiperiod MINLP model for the synthesis of flexible heat and mass exchange networks. Comput Chem Eng 18:1125–1139
8. Papalexandri KP, Pistikopoulos EN (1996) A generalized modular representation framework for process synthesis. AIChE J 42:1010–1032
9. Papalexandri KP, Pistikopoulos EN, Floudas CA (1994) Mass exchange networks for waste minimization: A simultaneous approach. Chem Eng Res Des 72:279–294
10. Sebastian P, Nadeau JP, Puiggali JR (1996) Designing dryers using heat and mass exchange networks: An application to conveyor belt dryers. Chem Eng Res Des 74:934–943
11. Srinivas BK, El-Halwagi MM (1994) Synthesis of combined heat and reactive mass exchange networks. Chem Eng Sci 49:2059–2074

MINLP: Outer Approximation Algorithm

IGNACIO E. GROSSMANN
Carnegie Mellon University, Pittsburgh, USA

MSC2000: 90C11

Article Outline

Keywords
See also
References

Keywords

Mixed integer nonlinear programming; Outer approximation method; Generalized Benders decomposition; Extended cutting plane method

The *outer approximation algorithm* (OA algorithm) ([1,2,9]) addresses mixed integer nonlinear programs of the form:

$$(\mathrm{P}) \quad \begin{cases} \min & Z = f(x,y) \\ \text{s.t.} & g_j(x,y) \le 0\,, \quad j \in J\,, \\ & x \in X\,,\ y \in Y\,, \end{cases}$$

where $f(\cdot)$, $g(\cdot)$ are convex, differentiable functions, J is the index set of inequalities, and x and y are the continuous and discrete variables, respectively. The set X is commonly assumed to be a convex compact set, e. g. $X = \{x\colon x \in \mathbf{R}^n, Dx \le d, x^L \le x \le x^U\}$; the discrete set Y corresponds to a polyhedral set of integer points, $Y = \{y\colon y \in \mathbf{Z}^m, Ay \le a\}$, and in most cases is restricted to $0-1$ values, $y \in \{0,1\}^m$. In most applications of interest the objective and constraint functions $f(\cdot)$, $g(\cdot)$ are linear in y (e. g. fixed cost charges and logic constraints).

The OA algorithm is based on the following theorem [1]:

Theorem 1 *Problem (P) and the following mixed-integer linear program (MILP) master problem (M-OA) have the same optimal solution (x^*, y^*),*

$$(\mathrm{M-OA}) \quad \min Z_L = \alpha$$

such that

$$\alpha \ge f(x^k,y^k) + \nabla f(x^k,y^k)\begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix}\,,$$

$$g_j(x^k,y^k) + \nabla g_j(x^k,y^k)\begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix} \le 0\,,$$

$$j \in J\,, \quad k \in K^*\,,$$

$$x \in X\,, \quad y \in Y\,,$$

where

$$K^* = \left\{ k\colon \begin{array}{c} (x^k,y^k)\ \textit{is the optimal} \\ \textit{solution to (NLP1)} \\ \textit{for all feasible}\ y^k \in Y \end{array} \right\}\,,$$

$$(\mathrm{NLP1}) \quad \begin{cases} \min & Z_U^k = f(x,y^k) \\ \text{s.t.} & g_j(x,y^k) \le 0\,, \quad j \in J\,, \\ & x \in X\,, \end{cases}$$

where Z_U^k is an upper bound to the optimum of problem (P).

Note that since the functions $f(x, y)$ and $g(x, y)$ are convex, the linearizations in (M-OA) correspond to outer approximations of the nonlinear feasible region in problem (P). Also, since the master problem (M-OA) requires the solution of all feasible discrete variables y^k, the following MILP relaxation is considered, assuming that the solution of K NLP subproblems is available:

$$(\mathrm{RM-OA}) \quad \min Z_L^K = \alpha$$

such that

$$\alpha \ge f(x^k,y^k) + \nabla f(x^k,y^k)\begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix}\,,$$

$$g_j(x^k,y^k) + \nabla g_j(x^k,y^k)\begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix} \le 0\,,$$

$$j \in J\,, \quad k = 1,\ldots,K\,,$$

$$x \in X\,, \quad y \in Y\,.$$

Given the assumption on convexity of the functions $f(x,y)$ and $g(x,y)$, the following property can be easily be established,

Property 2 The solution of problem (RM-OA), corresponds to a lower bound to the solution of problem (P).

Note that since function linearizations are accumulated as iterations proceed, the master problems (RM-OA) yield a nondecreasing sequence of lower bounds, $Z_L^1 \le \cdots \le Z_L^K$, since linearizations are accumulated as iterations k proceed.

The OA algorithm as proposed by M.A. Duran and I.E. Grossmann [1] consists of performing a cycle of major iterations, $k = 1,\ldots,K$, in which (NLP1) is solved for the corresponding y^k, and the relaxed MILP master problem (RM-OA) is updated and solved with the corresponding function linearizations at the point (x^k, y^k). The (NLP1) subproblems yield an upper bound that is used to define the best current solution, $\mathrm{UB}^K = \min(Z_U^k)$. The cycle of iterations is continued until this upper bound and the lower bound of the relaxed master problem, are within a specified tolerance.

It should be noted that for the case when the problem (NLP1) has no feasible solution, there are two major ways to handle this problem. The more general option is to consider the solution of the feasibility prob-

lem,

$$\text{(NLFP)} \quad \begin{cases} \min & u \\ \text{s.t.} & g_j(x, y^k) \le u\,, \quad j \in J\,, \\ & x \in X\,, \quad u \in \mathbb{R}^1\,. \end{cases}$$

R. Fletcher and S. Leyffer [2] have shown that for infeasible NLP subproblems, if the linearization at the solution of problem (NLFP) is included, this will guarantee convergence to the optimal solution.

For the case when the discrete set Y is given by 0–1 values in problem (P), the other option to ensure convergence of the OA algorithm without solving the feasibility subproblems (NLFP), is to introduce the following integer cut whose objective is to make infeasible the choice of the previous 0–1 values generated at the K previous iterations [1]:

$$\text{(ICUT)} \quad \begin{cases} \sum_{i \in B^k} y_i - \sum_{i \in N^k} y_i \le \left|B^k\right| - 1\,, \\ k = 1, \ldots, K\,, \end{cases}$$

where $B^k = \{i : y_i^k = 1\}$, $N^k = \{i y_i^k = 0\}$, $k = 1, \ldots, K$. This cut becomes very weak as the dimensionality of the 0–1 variables increases. However, it has the useful feature of ensuring that new 0–1 values are generated at each major iteration. In this way the algorithm will not return to a previous integer point when convergence is achieved. Using the above integer cut the termination takes place as soon as $Z_L^K \ge \mathrm{UB}^K$.

The OA method generally requires relatively few cycles or major iterations. One reason for this behavior is given by the following property:

Property 3 The OA algorithm trivially converges in one iteration if $f(x, y)$ and $g(x, y)$ are linear.

The proof simply follows from the fact that if $f(x, y)$ and $g(x, y)$ are linear in x and y the MILP master problem (RM-OA) is identical to the original problem (P).

It is also important to note that the MILP master problem need not be solved to optimality. In fact given the upper bound UB^K and a tolerance ε it is sufficient to generate the new (y^K, x^K) by solving,

$$\text{(M-OAF)} \quad \min Z_L^K = 0\alpha$$

such that

$$\alpha \ge \mathrm{UB}^k - \varepsilon\,,$$

$$\alpha \ge f(x^k, y^k) + \nabla f(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}\,,$$

$$g_j(x^k, y^k) + \nabla g_j(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \le 0\,,$$

$$j \in J\,, \quad k = 1, \ldots, K\,,$$

$$x \in X\,, \quad y \in Y\,.$$

While in (M-OA) the interpretation of the new point y^K is that it represents the best integer solution to the approximating master problem, in (M-OAF) it represents an integer solution whose lower bounding objective does not exceed the current upper bound UB^K; in other words it is a feasible solution to (M-OA) with an objective below the current estimate. Note that in this case the OA iterations are terminated when (M-OAF) is infeasible.

Another interesting point about the OA algorithm is the relationship of its master problem with the one of the *generalized Benders decomposition* method [3], which is given by:

$$\text{(RM-GBD)} \quad \min Z_L^K = \alpha$$

such that

$$\alpha \ge f(x^k, y^k) + \nabla_y f(x^k, y^k)^\top (y - y^k) + (\mu^k)^\top \left[g(x^k, y^k) + \nabla g(x^k, y^k)(y - y^k) \right]\,,$$

$$k \in KFS\,,$$

$$(\lambda^k)^\top \left[g(x^k, y^k) + \nabla g(x^k, y^k)(y - y^k) \right]\,,$$

$$k \in KIS\,,$$

$$x \in X\,, \quad \alpha \in \mathbb{R}^1\,,$$

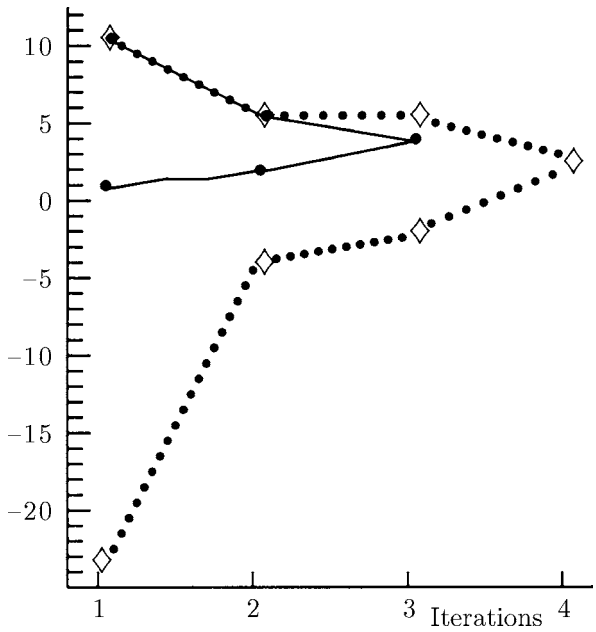
where *KFS* is the set of feasible subproblems (NLP1) and *KIS* the set of infeasible subproblems whose solution is given by (NLFP). Also $|\mathrm{KFS} \subset \mathrm{KIS}| = K$. The following property, holds between the two methods [1]:

Property 4 Given the same set of K subproblems, the lower bounds predicted by the relaxed master problem (RM-OA) are greater or equal to the ones predicted by the relaxed master problem (RM-GBD).

The above proof follows from the fact that the Lagrangian and feasibility cuts in (RM-GBD) are surrogates of the outer approximations in the master problem (M-OA). Given the fact that the lower bounds of

MINLP: Outer Approximation Algorithm, Table 1
Summary of computational results

Method	Subproblems	Master problems	LPs solved
BB	5 (NLP1)		
OA	3 (NLP2)	3 (M-PID)	19 LPs
GBD	4 (NLP2)	4 (M-GBD)	10 LPs
ECP	—	5 (M-MIP)	18LPs



MINLP: Outer Approximation Algorithm, Figure 1
Progress of iterations of OA and GBD for MINLP in MIP-EX

GBD are generally weaker, this method commonly requires a larger number of cycles or major iterations. As the number of 0–1 variables increases this difference becomes more pronounced. This is to be expected since only one new cut is generated per iteration. Therefore user-supplied constraints must often be added to the master problem to strengthen the bounds. As for the OA algorithm, the trade-off is that while it generally predicts stronger lower bounds than GBD, the computational cost for solving the master problem (M-OA) is greater since the number of constraints added per iteration is equal to the number of nonlinear constraints plus the nonlinear objective.

The OA algorithm is also closely related to the *extended cutting plane* (ECP) method by T. Westerlund and F. Peterssen [8]. The main difference lies that in the ECP method no NLP subproblem is solved, and that linerization simply takes place over the predicted continuous points from the MILP master problem, which in turn will normally only include linearizations of the most violated constraints.

Extension of the OA algorithm [4] include the LP/NLP based branch and bound [6], which avoids the complete solution of the MILP master problem (M-OA) at each major iteration. The method starts by solving an initial NLP subproblem which is linearized as in (M-OA). The basic idea consists then of performing an LP-based branch and bound method for (M-OA) in which NLP subproblems (NLP1) are solved at those nodes in which feasible integer solutions are found. By updating the representation of the master problem in the current open nodes of the tree with the addition of the corresponding linearizations, the need of restarting the tree search is avoided. Another important extension has been the method by Fletcher and Leyffer [2] who included a quadratic approximation based on the Hessian of the Lagrangian to the master problem (M-OAF) in order to capture nonlinearities in the 0–1 variables. Note that in this case the optimal solution of the mixed integer quadratic program (MIQP), Z^K, does not predict valid lower bounds in this case, and hence the constraint $\alpha \leq \text{UBK} - \varepsilon$ is added, with which the search is terminated when no feasible solution can be found in the MIQP master.

Finally, in order to handle equations in problem (P), G.R. Kocis and Grossmann [5] proposed the equality relaxation strategy, in which linearizations of equations are converted into inequalities for the MIP master problem according to the sign of the Lagrange multipliers of the corresponding NLP subproblem. J. Viswanathan and Grossmann [7], further proposed to add slack variables to this MILP master problem, and an augmented penalty function. Since in this generally nonconvex case the bounding properties do not apply, the algorithm was modified so as to start with the NLP relaxation of problem (P). If no integer solution is found, iterations between the MILP and NLP subproblems take place until there is no improvement in the objective function. This idea was precisely implemented in the commercial code DICOPT, which can also be modified to the original OA algorithm, if the user knows that the functions $f(x, y)$ and $g(x, y)$ are convex.

Example 5 In order to illustrate the performance of the OA algorithm, a simple numerical MINLP example is considered.

$$(\mathrm{MIP-EX}) \quad \begin{cases} \min & Z = y_1 + 1.5y_2 + 0.5y_3 \\ & \quad +x_1^2 + x_2^2 \\ \text{s.t.} & (x_1 - 2)^2 - x_2 \le 0 \\ & x_1 - 2y_1 \ge 0 \\ & x_1 - x_2 - 4(1 - y_2) \le 0 \\ & x_1 - (1 - y_1) \ge 0 \\ & x_2 - y_2 \ge 0 \\ & x_1 + x_2 \ge 3y_3 \\ & y_1 + y_2 + y_3 \ge 1 \\ & 0 \le x_1 \le 4, \quad 0 \le x_2 \le 4 \\ & y_1, y_2, y_3 = 0, 1\ . \end{cases}$$

The optimum solution to this problem corresponds to $y_1 = 0$, $y_2 = 1$, $y_3 = 0$, $x_1 = 1$, $x_2 = 1$, $Z = 3.5$. Figure 1 shows the progress of the iterations of the OA and GBD algorithm with the starting point $y_1 = y_2 = y_3 = 1$. As can be seen the lower bounds predicted by the OA algorithm are considerably stronger than the ones predicted by GBD. In particular at iteration 1, the lower bound of OA is 1.0 while the one of GBD is -23.5. Nevertheless, since this is a very small problem GBD requires only one more iteration than OA (4 versus 3). It is interesting to note that the NLP relaxation of this problem is 2.53, which is significantly lower than the optimal mixed integer solution. Also, as can be seen in Table 1, an NLP-based branch and bound method requires the solution of 5 NLP subproblems, while the ECP method requires 5 successive MILP problems.

See also

- ▶ Chemical Process Planning
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming

References

1. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307
2. Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Math Program 66:327
3. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10(4):237–260
4. Grossmann IE, Kravanja Z (1997) Mixed-integer nonlinear programming: A survey of algorithms and applications. In: Biegler LT, Coleman TF, Conn AR, Santosa FN (eds) Large-Scale Optimization with Applications. Part II: Optimal Design and Control. IMA vol Math Appl. Springer, Berlin, pp 73–100
5. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flow sheets. Industr Eng Chem Res 26(1869)
6. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput Chem Eng 16:937–947
7. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer-approximation method for MINLP optimization. Comput Chem Eng 14:769
8. Westerlund T, Pettersson F (1995) A cutting plane method for solving convex MINLP problems. Comput Chem Eng 19:S131–S136
9. Yuan X, Zhang S, Piboleau L, Domenech S (1988) Une methode d'optimisation nonlineare en variables pour la conception de procedes. Oper Res 22(331)

MINLP: Reactive Distillation Column Synthesis

Amy Ciric, Zeynep Gumus
Department Chemical Engineering,
University Cincinnati, Cincinnati, USA

MSC2000: 90C90

Article Outline

Keywords
Problem Statement
Distillation Based Superstructure Approaches
Heat and Mass Exchange Networks
Conclusions
See also
References

Keywords

Reactive distillation; Bilevel programming; MINLP

Reactive distillation (RD) occurs when a reaction takes place in the liquid holdup on the trays, in the reboiler, or in the condenser of a distillation column. Reactive distillation can increase the conversion of equilibrium limited reactions by continuously separating products and reactants, improve the selectivity in some kinetically limited reaction systems, and separate azeotropic and isomeric mixtures by converting one species into another that is easy to remove. It can also create a natural heat integration that uses an exothermic heat of reaction to create vapor boilup in a distillation column, and reduce capital costs by completing several processing steps in a single vessel. Reactive distillation is used commercially to produce methyl tert-butyl ether [13], esters including methyl acetate [1], and nylon 6, 6 [9]. It has also been proposed for hydrolysis reactions [7], ethyl- ene glycol synthesis [11], and cumene production [12]. See [7] for a review of the area.

As a result of increasing interest in the reactive distillation technique, systematic reactive distillation design methods have gained much importance. See [2,3,4,5] for residue curve maps, a powerful tool for visualizing distillation problems, to reactive distillation. In [7] this work was extended by including kinetic effects when the Damkohler number is fixed. In [14] synthesis of reactive distillation with multiple reactions is studied.

Reactive distillation poses a challenging problem for optimization based design techniques. Unlike in conventional distillation, holdup volume is an important design variable in reactive distillation, since the reaction generally takes place in the liquid body on the tray. The constant molar overflow assumption of conventional distillation design is not valid unless the reaction has thermal neutrality and is stoichiometrically balanced. For an optimal solution one should take into account that the feed to the column may be distributed. This, in addition to the holdup volume, liquid and vapor flows, composition and temperature profiles, number of trays and feed location(s) become major variables of an optimization problem which searches for a minimum of a cost function. The constraints of this optimization problem are material and energy balances, vapor-liquid equilibria, mole fraction summations, kinetic and thermodynamic relationships, and logical relationships between the variables. The resulting optimization model is a mixed integer nonlinear programming problem since it involves the optimum number of trays and feed tray locations which are integer variables. The cost function and the material and energy balances cause the nonlinearity of the problem.

There are two approaches to RD design via MINLP methods. One addresses reactive distillation through heat and mass exchanger networks [10], and the other addresses it through distillation column superstructures [6,8].

Problem Statement

The general problem of the reactive distillation column synthesis problem can be stated formally as follows.
Given:

- the chemical species, $i = 1, \ldots, I$, involved in the distillation; desired products, $i \in P$, and their production rates P_I;
- the set of chemical reactions, $j = 1, \ldots, J$;
- rate expressions r_j or an equilibrium constant K_j for each reaction j;
- heat of vaporization and vapor-liquid equilibrium data;
- cost of downstream separations;
- cost c_s and composition x_{is} of all feedstocks, $s = 1 \ldots S$;
- the cost of the column as a function of the number of trays and the internal vapor flow rate, $C(V, N)$;
- the form of the catalyst.

Determine:

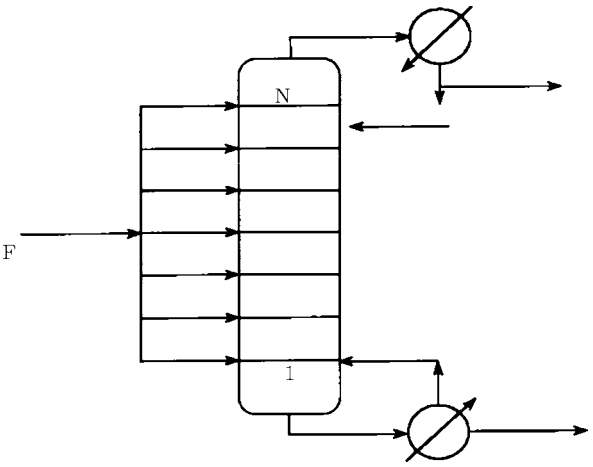
- the optimum number of trays;
- the trays where reactions take place;
- the holdup on each tray where a kinetically limited reaction takes place;

- the reflux ratio;
- the condenser and reboiler duties; and
- the feed location(s).

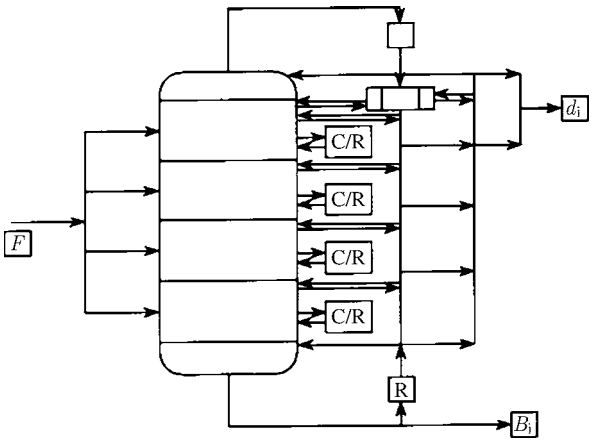
Such that the total cost is minimized while producing the correct amount of product.

Distillation Based Superstructure Approaches

One approach to MINLP based reactive distillation column design uses a superstructure that contains many different alternative designs embedded within it. Two different superstructures have been proposed; they differ in their treatment of the liquid reflux and vapor boilup, and in their heat management. See [6] for a structure that varies the number of trays and always recycles the liquid reflux to the top tray and the vapor boilup to the bottom tray (Fig. 1). More recently (1997), Z.H. Gumus and A.R. Ciric [8] modified the superstructure presented in [15] recycling vapor boilup and liquid reflux to each tray by adding a decanter to the distillate stream and side heaters and coolers to each tray (Fig. 2). In both of these superstructures, the number of trays may vary between 1 and some upper bound K. Each feed stream is split, and a portion is sent to each tray in the superstructure. In kinetically limited reactions, the hold-up volume may vary, and, in reactions systems with a solid catalyst, some trays will have reaction while others do not.



MINLP: Reactive Distillation Column Synthesis, Figure 1
Superstructure for optimum feed location(s) and number of trays [6]



MINLP: Reactive Distillation Column Synthesis, Figure 2
Tray-by-tray superstructure of [8]

The structure shown in Fig. 1 is appropriate for reactive distillation processes with a single liquid phase and kinetically limited reactions that are catalyzed with a solid catalyst. Representing the existence of each tray with an integer variable Y_k leads to a mixed integer nonlinear programming problem whose solution extracts a design with the number of trays, feed tray locations, reactive trays, holdup volumes, reflux ratio and boilup ratio that minimize the total cost. Assumed vapor liquid equilibrium on each tray, no reaction in the vapor phase, homogeneous liquid phase, negligible enthalpy of liquid streams, constant heat of vaporization leads to the MINLP shown below [6]:

$$\min Z = c_o + \sum_{sk} c_s F_{sk} + c_R Q_B + c_C Q_C + c_T D^{1.55} \times \sum \left(2Y_k + 1.27 \frac{W_k}{D^2} \right) + c_{SH} D \sum_k \left(H_o + \sum_{k' \le k} 2 + 1.27 \frac{W_{k'}}{D^2} \right)^{0.802} \times (Y_k - Y_{k+1})$$

subject to

$$\sum_s x_{is} F_{s1} - L_1 x_{i1}(1-\beta) + L_2 x_{i,2} - V_1 K_{i1} x_{i1} + \sum_j \nu_{ij} \xi_{1j} = 0, \quad (1)$$

$$\left[\sum_s x_{is}F_{sk} + V_{k-1}K_{i,k-1}x_{ik-1} + L_{k+1}x_{i,k+1} - L_k x_{ik} - V_k K_{ik} x_{ik} + \sum_j \nu_{ij}\xi_{ik}\right] Y_k = 0, \quad k = 2,\ldots,K, \tag{2}$$

$$\left[\lambda V_{k-1} - \lambda V_k - \sum_j \Delta H_j \xi_{jk}\right] Y_k = 0, \tag{3}$$

$$D_{ist} = \sum_k (V_k - L_{k+1})(Y_k - Y_{k+1}), \tag{4}$$

$$B_i = (1-\beta)L_1 x_{i1}, \tag{5}$$

$$B_i = P_i, \quad i \in P, \tag{6}$$

$$\left[\sum_i x_{ik} - 1\right](Y_k - Y_{k-1}) = 0, \tag{7}$$

$$\left[\sum_i K_{ik}x_{ik} - 1\right] Y_k = 0, \tag{8}$$

$$xd_i - K_{ik}x_{ik} - 1 + y_k - y_{k+1} \le 0, \tag{9}$$

$$x_{i,k+1} - xd_i - 1 + y_k - y_{k+1} \le 0, \tag{10}$$

$$\sum_i xd_i = 1, \tag{11}$$

$$\xi_{jk} = W_k f_j(x_{ik}, T_k), \tag{12}$$

$$K_{ik} = K_{ik}(x_{ik}, T_k), \tag{13}$$

$$V_k - F_{\max} Y_k \le 0, \tag{14}$$

$$\sum_s F_{sk} - F_{\max} Y_k \le 0, \tag{15}$$

$$L_{k+1} - F_{\max} Y_k \le 0, \tag{16}$$

$$W_k - W_{\max} Y_k \le 0, \tag{17}$$

$$Q_B = \beta\lambda L_1, \tag{18}$$

$$Q_C = \sum_k \lambda V_k (Y_k - Y_{k+1}), \tag{19}$$

$$D^4 \ge C_D \beta^2 L_1^2, \tag{20}$$

$$D \ge D_{\min}, \tag{21}$$

$$Y_{k+1} \le Y_k. \tag{22}$$

In this model, constraints (1) and (2) are the component balances of species i over the bottom tray and the remaining trays k; constraint (3) is the energy balance around tray k. The distillate flow is found with constraint (4). Distillate flow is calculated as the difference between the vapor flow leaving the top tray and the liquid flow entering it. Note that the term $Y_k - Y_{k+1}$ will be nonzero only for top tray, and zero for all others. Constraint (5) calculates the bottoms flow rate and constraint (6) specifies the production rate. Summation equations for the mole fractions are given in constraints (7) and (8). Constraints (9)–(11) identify the top tray and set the distillate and liquid reflux composition equal to the composition of the vapor leaving the top tray. Reaction rates are given in constraint (12), and the vapor liquid equilibrium constant is found by constraint (13). Constraints (14)–(17) ensure that when Y_k equals zero and tray k does not exist, the flows onto and off of the tray are zero. Constraints (18) and (19) calculate the reboiler and condenser duties, while constraints (20) and (21) find the column diameter. The last constraint ensures that tray $k+1$ does not exist if tray k does not exist.

In [6] this technique is demonstrated with the synthesis of a reactive distillation column that makes ethylene glycol from ethylene oxide and water. The main reaction is

$$C_2H_4O + H_2O \rightarrow C_2H_6O_2.$$

Further reaction of ethylene glycol gives the undesired byproduct diethylene glycol:

$$C_2H_4O + C_2H_6O_2 \rightarrow C_4H_{10}O_3$$

Ethylene glycol is produced using reactive distillation because the large volatility difference between the product and the reactants allows the continuous removal of EG from the reaction zone and absorption of the heat of reaction by the separation results in cost cuts.

The problem is solved using the reaction, physical property and cost data given in Table 1. The production rate is taken as 25 kg.mol/h of ethylene glycol. When the problem is solved without specifying the number of feed trays or their locations the solution obtained using GAMS is a 10-tray distillation column with a total

MINLP: Reactive Distillation Column Synthesis, Table 1
Ethylene glycol system; reaction, physical property and cost data

Reaction	Rate ($mol/cm^3.s$)	ΔH (kJ/mol)
1	$3.15\times10^9\exp\left[\frac{-9.547}{T(K)}\right]x_{EO}x_{H_2O}$	−80
2	$6.3\times10^9\exp\left[\frac{-9.547}{T(K)}\right]x_{EO}x_{EG}$	−13.1

Component K for P = 1 atm	
EO	$71.9\exp\left\{5.72\left[\frac{T-469}{T-35.9}\right]\right\}$
H_2O	$221.2\exp\left\{6.31\left[\frac{T-647}{T-52.9}\right]\right\}$
EG	$77\exp\left\{9.94\left[\frac{T-645}{T-71.4}\right]\right\}$
DEG	$47\exp\left\{10.42\left[\frac{T-681}{T-80.6}\right]\right\}$

EO feedstock: \$43.7/kmol	
Water feedstock: \$21.9/kmol	
Downstream separation: \$0.15/kmol H_2O in effluent	
C_{sh} =	\$222/yr
C_T =	\$15.7/yr
C_R =	\$146.8/kW.yr
C_C =	\$24.5/kW.yr
C_O =	\$10, 000/yr

annualized cost of 15.69×10^6/yr. The reaction zone is above tray 4 and the feed is distributed to each tray in the reaction zone. When the problem is slightly modified by adding constraints on the feed tray number, the solution changes to a 10-tray column with a total annualized cost of 15.73×10^6/yr. The reaction zone is between trays 4 and 10 and water is fed to tray 10 while ethylene glycol enters the column at tray 4. The selectivity reached by both columns is the same. (Fig. 3) shows the solutions. The column specifications are given in Table 2.

Heat and Mass Exchange Networks

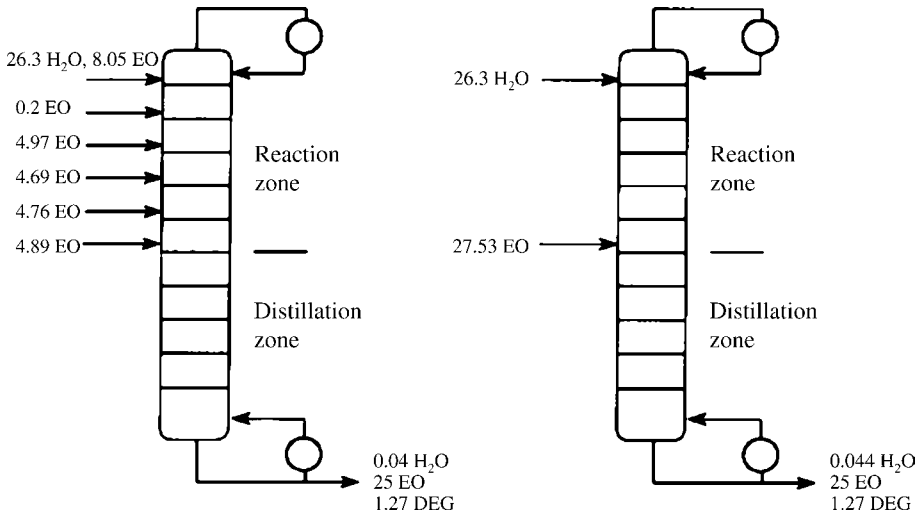
In this approach, process units are defined as combinations of heat and mass exchanger blocks, and the alternatives for the synthesis are explored simultaneously in a superstructure. A reactive distillation column can be described as a combination of mass/heat exchanger units with a condenser and a reboiler [10]. Heat and mass transfer takes place between the contacting vapor and liquid phases and from reactants to products. Multiple feeds and products and side heating and cooling tasks can be included in the description in the form of multiple mass and heat exchanger blocks between liquid and vapor streams. Its phase and quality define each stream. The quality indicator describes the leanness or richness of a stream in different components. Heat and mass transfer occurs between vapor and liquid streams of the same quality or between liquid and liquid (reactant and product) streams. For example, consider the reaction

$$A + B \rightarrow C + D.$$

Then there are liquid and vapor streams LABCD and VABCD in general notation. The streams lean in a component, for example in A, have that letter in parentheses, e. g. L(A)BCD or V(A)BCD. All possibilities of such streams, i. e. LAB(CD), VAB(CD), L(ABC)D, V(ABC)D, L(AB)C(D), V(AB)C(D), etc., and all the possible matches between them are considered within the structure. The possible matches are liquid-vapor matches of the same stream and all liquid-liquid matches.

This model describes exchangers with simple mass and energy balances and constraints defining phase and feasibility. Mass and heat generated or consumed by chemical reactions are included in the balances. Mass transfer is driven by a minimum concentration approach while a minimum temperature approach is the driving force for heat transfer. Concentration and temperature approach constraints are considered at each end of the exchanger. Equilibrium can be represented by a zero concentration approach, which means no driving force for mass transfer.

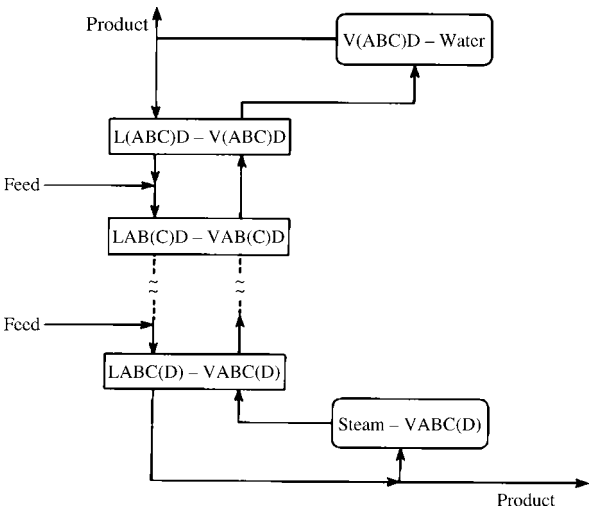
In the synthesis framework for an optimal process network, one should start with the construction of the stream sets containing all the initial, intermediate, and final process streams. The key is the availability of the physical and chemical property information on the streams. When the information is not enough to identify the individual streams, especially the intermediate streams, a general set of one vapor and one liquid stream is constructed, which contain all components involved in the process. The second step is to list all the possible stream matches. Engineering knowledge plays an important role in this step. One should be careful about not listing redundant or meaningless stream



MINLP: Reactive Distillation Column Synthesis, Figure 3
Optimal distributed and two-feed columns for ethylene glycol production

MINLP: Reactive Distillation Column Synthesis, Table 2
Column specifications for ethylene glycol production

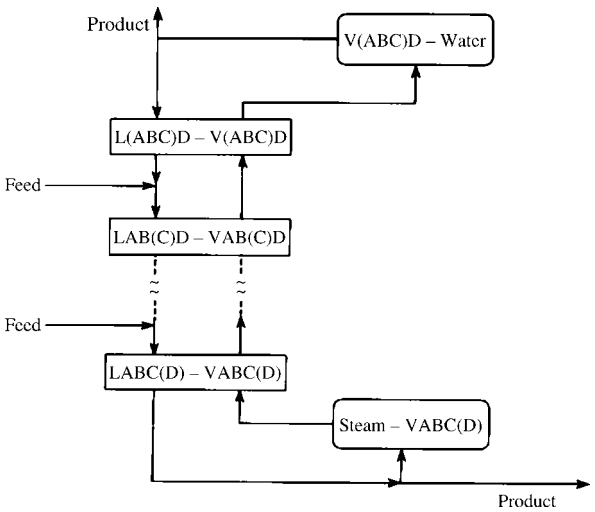
Feed type	Diam. (m)	Height (m)	Boilup ratio	Reboiler duty (MW)	Condenser duty (MW)
Distr.	1.3	12	0.958	6.7	7.31
Two-feed	1.3	12	0.96	6.9	7.5



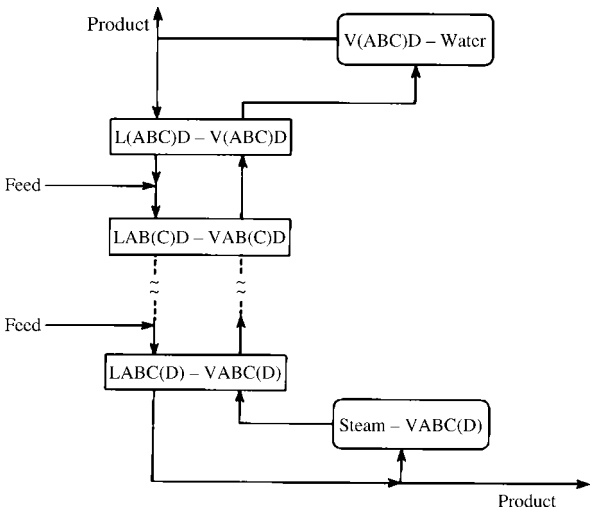
MINLP: Reactive Distillation Column Synthesis, Figure 4
Mass/heat exchange network representation of a multifeed reactive distillation column

matches since these will only make the problem more complex. Knowledge about the system is the key in this screening stage. Developing the mass/heat exchange network superstructure is the next step in the framework. All possible interconnections between the stream splitters and mixers should be taken into consideration. The last step is the optimization of the superstructure. Usually, the objective function of the optimization problem is a cost function. If the cost function includes only operating cost, which depends on the raw material and utility consumption, the objective function can be easily formulated from the superstructure. If, however, capital investment costs are involved in the objective cost function, the formulation is not straightforward from the superstructure, since process unit specifications are not considered in the superstructure. In this case, capital cost is to be approximated using cost functions that take operating conditions into account. Separation difficulty can be used in evaluating the capital cost of a distillation tray.

K.P. Papalexandri and E.N. Pistikopoulos [10] used the production of ethylene glycol from ethylene oxide and water to demonstrate this approach. The reactions involved in this production were given before. Physical properties, cost and reaction data are



MINLP: Reactive Distillation Column Synthesis, Figure 5
Steps of the synthesis framework



MINLP: Reactive Distillation Column Synthesis, Figure 6
Optimal reactive distillation column for ethylene glycol production

the same as given earlier in Table 1. The difference from the example problem studied in [6] is the objective, which is the minimization of operating cost only. The set of streams include the intermediate streams L{EO, H2O, EG, DEG} and V{EO, H2O, EG, DEG} and the product streams L(EG) and L(DEG). Five liquid-liquid mass/heat exchange matches and 15 liquid-vapor mass/heat exchange matches are considered. Representing each match with a binary variable, and considering all possible interactions between units, the problem is formulated as a mixed integer nonlinear programming problem with the objective of minimizing operating cost, which includes raw material cost, purification, and utility cost. The optimal reactive distillation column obtained is pictured in Fig. 6. The column has two reaction zones and multiple feeds, and the operating cost is 1.17×10^6 \$/yr.

Conclusions

This paper discussed the MINLP applications in reactive distillation design problems. Two main approaches are studied: distillation based superstructure approach that uses rigorous tray-by-tray method to model reactive distillation, and heat and mass exchanger network superstructure approach that realizes reactive distillation processes as combinations of several mass/heat exchangers with a condenser and a reboiler. Examples are included to demonstrate the approaches.

See also

- Chemical Process Planning
- Extended Cutting Plane Algorithm
- Generalized Benders Decomposition
- Generalized Outer Approximation
- MINLP: Application in Facility Location-allocation
- MINLP: Applications in Blending and Pooling Problems
- MINLP: Applications in the Interaction of Design and Control
- MINLP: Branch and Bound Global Optimization Algorithm
- MINLP: Branch and Bound Methods
- MINLP: Design and Scheduling of Batch Processes
- MINLP: Generalized Cross Decomposition
- MINLP: Global Optimization with αBB
- MINLP: Heat Exchanger Network Synthesis
- MINLP: Logic-based Methods
- MINLP: Outer Approximation Algorithm
- Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- Mixed Integer Nonlinear Programming

References

1. Agreda VH, Partin LR, Heise WH (1990) High purity methyl acetate via reactive distillation. Chem Eng Prog 86(2):40–46

2. Barbosa D, Doherty MF (1988) Design and minimum reflux calculations for double-feed multicomponent reactive distillation columns. Chem Eng Sci 43:2377
3. Barbosa D, Doherty MF (1988) Design and minimum reflux calculations for single-feed multicomponent reactive distillation columns. Chem Eng Sci 43:1523
4. Barbosa D, Doherty MF (1988) The influence of equilibrium chemical reactions on vapor-liquid phase diagrams. Chem Eng Sci 43:529
5. Barbosa D, Doherty MF (1988) The simple distillation of homogeneous reactive mixtures. Chem Eng Sci 43:541
6. Ciric AR, Gu D (1994) A mixed integer nonlinear programming approach to nonequilibrium reactive distillation synthesis. AIChE J 40(9):1479
7. Doherty MF, Buzad G (1992) Reactive distillation by design. Trans Inst Chem Eng 70:448–458
8. Gumus ZH, Ciric AR (1997) Reactive distillation column design with vapor/liquid/liquid equilibria. Comput Chem Eng 21:S983–988
9. Jacobs DB, Zimmermann J (1977) Chap. 12. In: Schildknecht CE (ed) Polymerization Processes. Wiley, New York
10. Papalexandri KP, Pistikopoulos EN (1996) A generalized modular representation framework for process synthesis based on mass/heat transfer principles. AIChE J 42(4): 1010
11. Parker AS (1958) Preparation of alkylene glycol. US Patent 2, 839, 588
12. Shoemaker JD, Jones EM (1987) Cumene by catalytic distillation. Hydrocarbon Proc June:57–58
13. Smith LA (1984) Method for the preparation of methyl tertiary butyl ether. US Patent 4, 978, 807
14. Ung S, Doherty MF (1995) Synthesis of reactive distillation systems with multiple equilibrium chemical reactions. Industr Eng Chem Res 34:2555–2565
15. Viswanathan J, Grossmann IE (1993) Optimal feed locations and number of trays for distillation columns with multiple feeds. I–EC Res 32:2942–2949

MINLP: Trim-loss Problem

Iiro Harjunkoski[1], Ray Pörn[2], Tapio Westerlund[3]
[1] Process Design Lab., Åbo Akad. University, Turku, Finland
[2] Department Math., Åbo Akad. University, Turku, Finland
[3] Process Design Lab., Åbo Akad. University, Åbo, Finland

MSC2000: 90C11, 90C90

Article Outline

Keywords
Problem Formulation
Linear Transformations
Parameterization Methods
Convex Transformations
  Exponential Transformation
  Square-Root Transformation
  Logarithmic and Square-Root Transformation
  Inverted Transformation
  Modified Square-Root Transformation
Example: A Numerical Problem
Conclusions
Notation
See also
References

Keywords

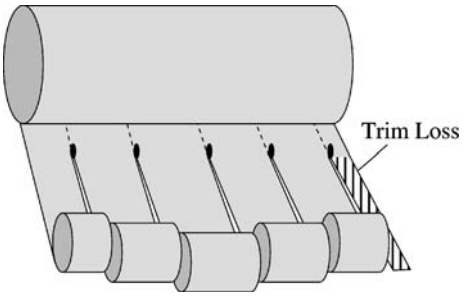
Scheduling; Paper converting; Trim-loss problem; Bilinear; Convex transformation

The trim-loss problem is one of the most demanding optimization problems in the paper-converting industry. It appears when an order specified by a customer is to be satisfied by cutting out a set of product reels from a wider raw paper reel.

The products in the order are characterized by width and quality. In a paper-converting mill the raw paper can be printed, coated and cut. In a typical paper-converting mill, there may be hundreds of different products to be produced. When considering the trim-loss problem, width is the most important property while the main problem is to determine such cutting patterns that minimize waste production, the *trim loss*.

In the optimization problem, beyond the number of cutting patterns needed, the appearance of each cutting pattern needs to be determined at the same time as having to decide how many times the cutting patterns ought to be repeated.

The customer widths and the raw paper widths are often more or less independent of each other. This makes it combinatorially very demanding to produce a cutting plan that minimizes the trim loss. Even if the trim-loss problem is in its basic form an integer problem, it has often been solved by linear programming (LP) methods [3] or some heuristic algorithms



MINLP: Trim-loss Problem, Figure 1
The cutting procedure

[4]. A good survey of widely used solution methods for trim-loss and assortment problems is given in [7].

When using an LP-approach to solve an integer problem the biggest difficulty is to convert the continuous solution such that the integer variables obtain integer values. The rounding methods are heuristic [8] and often fail to give the optimal integer solution even though the solution may be fairly good.

Problem Formulation

The trim-loss problem is a bilinear nonconvex integer nonlinear programming (INLP) problem. The appearance of a cutting pattern needs to be determined by integer variables and the bilinearity comes from the demand constraints.

A cutting pattern tells how many times a certain product is cut out from the raw paper. Let a cutting pattern have the index j and a product the index i. Assume a customer demand with I different products and further assume that the maximum allowed number of different cutting patterns is J. Further let m_j be the number of times a certain cutting pattern is repeated and n_{ij} be the number of times a product i appears in cutting pattern j. If the demand of a product i is expressed by $n_{i,\,\mathrm{order}}$, the demand constraints can be written as

$$\begin{aligned} & n_{i,\mathrm{order}} - \sum_{j=1}^{J} m_j \cdot n_{ij} \le 0, \\ & i = 1, \ldots, I, \\ & m_j, n_{ij} \in \mathbb{Z}^+. \end{aligned} \tag{1}$$

The negative bilinear terms make the problem nonconvex. Both of the variables in the term are integer variables and consequently the problem is a bilinear integer optimization problem. It is not possible to replace one of the variables n_{ij} with a continuous variable because this would violate the product specification. In theory it is possible to replace the m_j with a continuous variable but this may easily dissatisfy the desired product reel length and diameter requirements. Therefore, in the following study it is preferable to keep both m_j and n_{ij} as integers.

While raw paper reels of the same width are often glued together to form a continuous raw paper reel the problem can be simplified by omitting the raw paper length and assuming that the pattern lengths are equal.

Besides the demand constraint, certain constraints are needed to keep the problem feasible. Let the width of a product i be expressed by b_i and the width of the raw paper used for cutting pattern j by $B_{j,\,\max}$. The trim-loss width cannot exceed, for instance, 200mm owing to the machinery. This limit is represented by Δ_j. Furthermore, the maximum number of products that can be cut out from a pattern often has a physical restriction. The outcoming product reels have to form an angle big enough so that the reels do not attach together, yet with too big an angle between the outermost reels the paper may be torn off. Let this upper limit be $N_{j,\,\max}$.

Besides the total number of patterns, the pattern changes are also of interest when doing the optimization. This is due to the fact that the machinery normally needs to be stopped for a knife change which causes a production stop. Let therefore the variable y_j be 1 if the cutting pattern j exists and 0 if not. The sum of y_j variables then indicates how many different cutting patterns are needed to satisfy the production and the sum of m_j indicates the total number of all patterns which are related to the running metres of the raw material.

Now the basic formulation can be written in mathematical form. The objective is to minimize the total number of patterns and the number of pattern changes.

$$\min_{m_j, n_{ij}, y_j} \left\{ \sum_{j=1}^{J} c_j \cdot m_j + C_j \cdot y_j \right\} \tag{2}$$

subject to

$$\sum_{i=1}^{I} b_i \cdot n_{ij} - B_{j,\max} \le 0, \tag{3}$$

$$-\sum_{i=1}^{I} b_i \cdot n_{ij} + B_{j,\max} - \Delta_j \le 0, \tag{4}$$

$$\sum_{i=1}^{I} n_{ij} - N_{j,\max} \le 0, \tag{5}$$

$$y_j - m_j \le 0, \tag{6}$$

$$\begin{aligned} & m_j - M_j \cdot y_j \le 0, \\ & j = 1, \ldots, J, \end{aligned} \tag{7}$$

$$\begin{aligned} & n_{i,\text{order}} - \sum_{j=1}^{J} m_j \cdot n_{ij} \le 0, \\ & i = 1, \ldots, I, \\ & m_j,\ n_{ij} \in \mathbb{Z}, \quad y_j \in \{0, 1\}. \end{aligned} \tag{8}$$

The M_j gives the upper bound for corresponding m_j variables. When using an objective as in (2) the constraint (6) becomes irrelevant. The width constraints are given in (3)–(4) and the constraint (5) restricts the number of cuts in a pattern. The binary variables, y_j, are defined in (6)–(7).

The functionality of the variables are demonstrated in the following figure where the raw-paper width is $B_{j,\max}$. Note that the pattern length may typically be e. g. 6500m.

The last constraint, the demand constraint (8), is an integer bilinear constraint where both variables in bilinear terms are pure integers. This makes the problem a nonconvex MINLP problem where the nonconvexity appears in the integer variables.

There are very few methods available that are capable of solving similar nonconvex MINLP problems. Some heuristic methods such as simulated annealing [9] may find the global optimal solution within infinite time but algorithmic methods have not been proven to converge with such types of problems. Only recently (1999) some advancements have been reported in [1] and [11].

However, it is fully possible to transform the trim-loss problem into convex or linear form and use some established MINLP or MILP solver to solve the resulting problem to global optimality. Some linear transformations are presented in [6] and methods to transform the nonconvex problem into a convex form can be found in [10] and [5].

Linear Transformations

As can be seen from (2)–(8), all constraints but the last demand constraint are linear. This means that the problem should be fairly well bounded already by the linear part of the problem and thus a linear formulation strategy seems to be fully possible.

However, this linear transformation requires new variables and constraints that may complicate the problem. Using a standard approach, by rewriting one of the integer variables in the bilinear term by binary variables, the following is obtained.

$$\begin{aligned} & m_j = \sum_{k=1}^{K} 2^{k-1} \cdot \beta_{jk}, \\ & m_j \in \mathbb{R}, \quad \beta_{jk} \in \{0, 1\}. \end{aligned} \tag{9}$$

K is the number of binary variables needed. By defining L_{ij} to be the upper bound for respective n_{ij} variables and introducing a new slack-variable s_{ijk} the following constraints will create a necessary link between the n_{ij} and s_{ijk} variables:

$$s_{ijk} - n_{ij} \le 0, \tag{10}$$

$$-s_{ijk} + n_{ij} - L_{ij} \cdot (1 - \beta_{jk}) \le 0, \tag{11}$$

$$s_{ijk} - L_{ij} \cdot \beta_{jk} \le 0. \tag{12}$$

Using the above constraints the bilinear demand constraint can be written in linear form

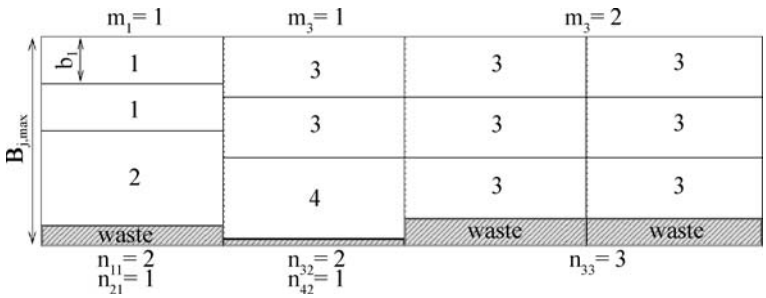
$$n_{i,\text{order}} - \sum_{j=1}^{J} \sum_{k=1}^{K} 2^{k-1} \cdot s_{ijk} \le 0. \tag{13}$$

The m_j could also be represented by special ordered sets (SOS) where at most, one of the binary variables are allowed to be nonzero.

$$m_j = \sum_{k=1}^{K} k \cdot \beta_{jk}\ , \qquad \sum_{k=1}^{K} \beta_{jk} \le 1\ . \tag{14}$$

It should be noted that the usage of this kind of transformation may enlarge the integrality gap unless for instance the n_{ij} variables in equations (3)–(5) are replaced with corresponding variables s_{ijk}.

The same transformation can be modified such that n_{ij} is replaced by a binary representation and m_j is defined through the slack-variables s_{ijk}.



MINLP: Trim-loss Problem, Figure 2
The integer variables

Parameterization Methods

Beyond the linear transformation, the problem can be written in linear form by simply parameterizing one of the variables in the bilinear term. This method though may lead to global optimality only in such cases where all the possible combinations have been considered. This strategy may be good for smaller problems but it may also generate far too many integer variables in solving larger trim-loss problems.

It is quite easy to generate all the possible combinations of n_{ij} variables satisfying the constraints (3)–(5). This strategy results in a problem where these constraints can be removed and where the n_{ij} variables in the resulting linear demand constraint are parameters:

$$n_{i,\text{order}} - m_j \cdot n'_{ij} \le 0\,, \qquad m_j \in \mathbb{Z}\,. \tag{15}$$

The same type of parameterization strategy may also be applied to the other variable m_j but in this case it may be more difficult to define the exact values of the parameters. One strategy is to use the upper bounds M_j or define all the m_j variables to be equal to one and make sure that a sufficient amount of the variables m_j are considered.

Another alternative is to combine the parameterization and transformation methods so that a proper amount of parameterized variables are combined with original variables. This strategy may be very efficient but often requires such information that may be difficult to obtain from a larger problem without any knowledge of the solution.

Convex Transformations

In the previous sections a number of methods were presented where the nonconvex problem can be transformed or parameterized into linear form. The main drawback for this linear transformation strategy is the large number of extra constraints and continuous variables. The parameterization strategy results in a formulation with a few constraints but many extra integer variables.

In the following a number of convexification methods are presented. Generally, the convex formulations need fewer extra constraints and continuous variables as the linear strategies and no extra integer variables as is the case with the parameterization methods. Thus, the convex transformation could be expected to result in formulations which are easier to solve especially for larger-scale orders. This creates an interesting problem, where the integer search space is reduced at the expense of more complex nonlinear functions, which could, in principle, be used as benchmarks for the performance of MINLP algorithms.

The basic principle for the convex transformation is to first expand the bilinearity in the demand constraint

$$m_j \cdot n_{ij} = (m_j + \tau)(n_{ij} + \tau) - \tau \cdot (m_j + n_{ij}) - \tau^2. \tag{16}$$

In the following text, the translation constant $\tau = 1$ is used for simplicity. The second step is to substitute the bilinear term in the original demand constraint

$$n_{i,\text{order}} - \sum_{j=1}^{J} (m_j + 1)(n_{ij} + 1) + \sum_{j=1}^{J} (m_j + n_{ij}) + J \le 0. \tag{17}$$

It should be noted that the transformations that follow need to consider the whole problem not only individual

functions, which makes the transformation techniques more demanding. A transformation of a single function may cause linear constraints to become nonlinear if one is unaware of this fact.

Exponential Transformation

The demand constraint is originally a negative bilinear constraint. The exponential transformation can only be applied to a positive bilinear constraint. Therefore, one of the variables in the bilinear term needs to be substituted with its reversed value.

$$r_{ij} = N_{j,\max} - n_{ij} \tag{18}$$

and the demand constraint is modified to

$$n_{i,\text{order}} - \sum_{j=1}^{J} m_j \cdot N_{j,\max} + \sum_{j=1}^{J} m_j \cdot r_{ij}. \tag{19}$$

Now the exponential transformation can be applied. The transformation is of the form

$$m_j + 1 = e^{M_j}, \quad r_{ij} + 1 = e^{R_{ij}} \tag{20}$$

and the variables are defined as

$$m_j = \sum_{l=1}^{L_j} \beta_{jl} \cdot l, \tag{21}$$

$$M_j = \sum_{l=1}^{L_j} \beta_{jl} \cdot \ln(l+1), \tag{22}$$

$$r_{ij} = \sum_{k=1}^{K_i} \beta_{ijk} \cdot k, \tag{23}$$

$$R_{ij} = \sum_{k=1}^{K_i} \beta_{ijk} \cdot \ln(k+1), \tag{24}$$

$$\sum_{l=1}^{L_j} \beta_{jl} \le 1, \quad \sum_{k=1}^{K_i} \beta_{ijk} \le 1, \tag{25}$$

$$\beta_{jl}, \beta_{ijk} \in \{0,1\}, \quad M_j, R_{ij} \in \mathbb{R}.$$

When combining these definitions, the demand constraint can be written in convex form

$$n_{i,\text{order}} - J + \sum_{j=1}^{J} e^{M_j + R_{ij}} - \sum_{j=1}^{J} \left((N_{j,\max} + 1) \cdot \sum_{l=1}^{L_j} \beta_{jl} \cdot l + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k \right) \le 0. \tag{26}$$

This transformation can also be achieved in slightly another way but using this strategy also requires updating some of the constraints in (3)–(7).

Square-Root Transformation

This transformation is almost equivalent to the previous one. A main difference is that it can be applied straight to the negative bilinear constraint and thus no r_{ij} variables need to be defined. The constraint (21) is valid but the constraint (23) needs to be modified to

$$n_{ij} = \sum_{k=1}^{K_i} \beta_{ijk} \cdot k. \tag{27}$$

Note that the equations in (25) are valid. The transformation is of the form

$$m_j + 1 = \sqrt{M_j}, \quad n_{ij} + 1 = \sqrt{N_{ij}}. \tag{28}$$

The transformation variables M_j and N_{ij} are defined as

$$M_j = 1 + \sum_{l=1}^{L_j} \beta_{jl} \cdot l(l+2), \tag{29}$$

$$N_{ij} = 1 + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k(k+2), \tag{30}$$

$$\beta_{jk}, \beta_{ijk} \in \{0,1\}, \quad M_j, N_{ij} \in \mathbb{R}.$$

and the resulting convex demand constraint is

$$n_{i,\text{order}} + J - \sum_{j=1}^{J} \sqrt{M_j \cdot N_{ij}} + \sum_{j=1}^{J} \left(\sum_{l=1}^{L_j} \beta_{jl} \cdot l + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k \right) \le 0. \tag{31}$$

Logarithmic and Square-Root Transformation

The square-root and the logarithmic functions can be combined, resulting in a third convex transformation. It is directly applicable to the negative bilinear function and the transformation can be written as

$$m_j + 1 = \sqrt{M_j}, \qquad n_{ij} + 1 = \ln N_{ij}. \tag{32}$$

The m_j, n_{ij} and M_j variables are defined as in the square-root transformation and the N_{ij} is defined as

$$N_{ij} = e + \sum_{k=1}^{K_i} \beta_{ijk} \cdot (e^{k+1} - e) \tag{33}$$

and the following convex demand constraint is obtained

$$n_{i,\text{order}} + J - \sum_{j=1}^{J} \sqrt{M_j} \cdot \ln N_{ij} + \sum_{j=1}^{J} \left(\sum_{l=1}^{L_j} \beta_{jl} \cdot l + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k \right) \leq 0. \tag{34}$$

It can be noted in equation (34) that the only difference to the former transformation is the third term of the demand constraint.

Inverted Transformation

The following transformation can be applied to a positive bilinear constraint. Thus the same definition of r_{ij} has to be done as for the exponential transformation. The transformation has the form

$$m_j + 1 = \frac{1}{M_j}, \quad r_{ij} + 1 = \frac{1}{R_{ij}}. \tag{35}$$

The definitions of the transformation variables follow:

$$M_j = 1 + \sum_{l=1}^{L_j} \beta_{jl} \cdot \left(\frac{1}{l+1} - 1 \right), \tag{36}$$

$$R_{ij} = 1 + \sum_{k=1}^{K_i} \beta_{ijk} \cdot \left(\frac{1}{k+1} - 1 \right). \tag{37}$$

The demand constraint is obtained exactly in the same way as before

$$n_{i,\text{order}} - J + \sum_{j=1}^{J} \frac{1}{M_j \cdot R_{ij}} - \sum_{j=1}^{J} \left((N_{\max} + 1) \cdot \sum_{l=1}^{L_j} \beta_{jl} \cdot l + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k \right) \leq 0. \tag{38}$$

Modified Square-Root Transformation

As the last transformation, a modification to the previously presented square-root transformation is introduced. In such cases where the variable m_j may take large values, it may be more efficient to use another type of binary representation.

$$m_j = \sum_{l=1}^{L'_j} 2^{l-1} \cdot \beta_{jl}, \tag{39}$$

where $L_j' = \lfloor \log_2(m_{j,\max}) \rfloor + 1$ if $m_{j,\max}$ is the upper bound for the respective m_j variable. This modification reduces the required number of binary variables and the transformation variable M_j needs to be redefined. The definition also requires additional slack-variables and constraints. In the following, the square-root transformation is used:

$$M_j = 1 + \sum_{l=1}^{L'_j} (s^{2l-2} + 2^l) \cdot \beta_{jl} + \sum_{l,m=1;m<l}^{L'_j} 2^{l+m-1} \cdot s_{jlm}, \tag{40}$$

$$-s_{jlm} - 1 + \beta_{jl} + \beta_{jm} \leq 0, \tag{41}$$

$$2 \cdot s_{jlm} - \beta_{jl} - \beta_{jm} \leq 0, \quad l, m = 1, \ldots, L'_j; \quad m < l. \tag{42}$$

By adding the extra constraints and defining N_{ij} as in the square-root strategy, the demand constraint can be

written in convex form as follows

$$n_{i,\text{order}} + J - \sum_{j=1}^{J} \sqrt{M_j \cdot N_{ij}} + \sum_{j=1}^{J} \left(\sum_{l=1}^{L_j} \beta_{jl} \cdot 2^{l-1} + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k \right) \leq 0. \quad (43)$$

Five methods for transforming the originally nonconvex trim-loss problem into convex form have been discussed. Three of them were directly applicable to a negative bilinear function but for two methods some operations were needed to change the demand constraint into a positive bilinear constraint.

Example: A Numerical Problem

In this last section a numerical example is solved with all of the presented methods. To improve the performance of the solution procedure some extra linear constraints need to be defined. They are, however, not specified here.

In the following example order an upper limit for products $n_{i,\,\max}$ that are allowed to be produced also has been defined. Here, the maximal possible overproduction of any product is 2. This limit is somewhat unnatural and is therefore not used as a constraint. However, the use of this type of upper bounds makes it possible to efficiently reduce the combinatorial space.

i	b_i(mm)	$n_{i,\text{order}}$	$n_{i,\max}$
1	330	8	10
2	360	16	18
3	380	12	14
4	430	7	9
5	490	14	16
6	530	16	18

Example order

The example demand is a mid-size customer order with a total weight of 27.5tons. Some important parameters need to be defined before optimization. The raw paper width of 2200mm is chosen and a maximal trim loss of 100mm is tolerated. At most 5 products may be cut out from a cutting pattern. Among the following parameters, the parameter M_j refers to the upper bound of the respective m_j variable and the parameter N_i to the n_{ij} variables. Note, that since the raw paper width is equal for every pattern the latter upper bound is independent of the index j.

$J = I = 6$	$N_{j,\max} = 5$
$c_j = 1$	$M_j = \{14, 12, 8, 7, 4, 2\}$
$C_j = 0.1$	$N_i = \{2, 3, 3, 5, 3, 4\}$
$B_{j,\max} = 2200$mm	$M_{\min} = 15$
$\Delta_j = 100$mm	

The problem parameters

The parameter $M_{\min}$ is the lower bound for the sum of the variables m_j. This sum can easily be calculated in advance and significantly enhances the optimization performance.

Before doing the actual optimization it should be pointed out that the results are not comparable. The main purpose for showing the numerical results is to demonstrate that the above presented strategies are fully usable and result in quite efficient solvable formulations. The transformation strategies can be directly applied to any problem where the bilinear terms contain integer variables.

The methods are divided into three groups of which the linear transformation and the parameterization strategies result into MILP formulations. The third group, the convex transformation strategy produces MINLP formulations that have in this case been solved using the extended cutting plane (ECP) algorithm by T. Westerlund and F. Peterssen [12].

In the parameterization strategies the problem is redefined by parameterizing certain variables which means that the resulting problem has already been partly solved. This may, however, not always be a benefit, especially in such problems where a huge number of parameters increases the integer search space for other variables.

The strategies are numbered as follows:

1.	binary representation of m_j
2.	binary representation of n_{ij}
3.	parameterization of n_{ij}
4.	parameterization of m_j
5.	exponential transformation
6.	square-root transformation
7.	logarithmic and square-root transformation
8.	inverted transformation
9.	modified square-root transformation

The strategies enlarge the problem both in terms of variables and constraints. In the following the number of variables and constraints are given. All the constraints are linear except in the convex transformation strategies where six of the constraints are nonlinear.

The strategies 1–4 are linear formulations of which 3–4 use the parameterization strategy to overcome the bilinearity. Strategies 5–9 are convex transformations. The field with combinations gives simply the number of unconstrained discrete variable combinations as a function of number of binary variables. This information is more informative than just the number of variables.

Strategy	Constraints	Variables (I/B/C)	Comb. 2^n
1.	408	36/23/120	2^{98}
2.	366	6/88/144	2^{105}
3.	59	51/51/—	2^{140}
4.	201	282/47/—	2^{634}
5.	199	—/169/84	2^{96}
6.	199	—/169/84	2^{96}
7.	185	—/169/84	2^{96}
8.	185	—/169/84	2^{96}
9.	225	—/208/84	2^{118}

The MILP problems 1–4 were solved with CPLEX-5.0 using default settings and the MINLP problems 5–9 were solved by 'mittlp', an ECP application written by H. Skrifvars. The optimization was done on a Pentium Pro 200MHz running the Linux operating system.

The optimization results can be seen in the following table.

Strategy	Nodes (MILP)	ECP-iter. (MINLP)	CPU-time (s)
1.	265	-	7.6
2.	51	-	0.51
3.	2174	-	3.2
4.	265	-	7.7
5.	-	4	8.6
6.	-	7	66.6
7.	-	9	138.6
8.	-	10	736.4
9.	-	6	49.9

The optimal result has two cutting patterns with the widths $B_1 = 2110$ mm and $B_2 = 2170$ mm and multiples $m_1 = 8$, $m_2 = 7$. The appearances of the patterns are given by the following variables: $n_{1,1} = 1$, $n_{2,1} = 2$, $n_{6,1} = 2$, $n_{3,2} = 2$, $n_{4,2} = 1$, $n_{5,2} = 2$

Conclusions

The study above is not a fair comparison. Experience has shown that the performance order is highly dependent on the specific problem. In order to get an idea of which of the methods is, in average the most efficient one, tens of problems of different sizes need to be solved. However, the study illustrates that it is fully possible to apply the transformation methods to a well explored real industrial problem.

In the present study the trim-loss problem was used as an example case but the transformation methods are general and can be applied to any problem with similar type of bilinear constraints.

Notation

i	product index
j	cutting pattern index
I	number of products in the order
J	number of possible cutting patterns
m_j	number of times the pattern j is used
n_{ij}	number of product i in pattern j
r_{ij}	reversed value of n_{ij}
$n_{i,\,\text{order}}$	number of product i ordered
b_i	width of product i
$B_{j,\,\max}$	width of raw paper of pattern j
Δ_j	max. trim-loss width
$N_{j,\,\max}$	max. number of products in pattern j
y_j	binary variable that is one if $m_j > 0$
c_j, C_j	cost coefficients
M_j	upper bound / transformation variable
β_{jl}, β_{jk}	binary variables for defining m_j
L_{ij}	upper bound
s_{ijk}	slack-variable for linear transformations
β_{ijk}	binary variables for defining n_{ij} or r_{ij}
n_{ij}'	fixed n_{ij} values
τ	translation constant
N_{ij}	transformation variable
R_{ij}	transformation variable
l, k, m	indices of binary variables
L_j, K_i	number of binary variables needed

See also

- Branch and Price: Integer Programming with Column Generation
- Decomposition Techniques for MILP: Lagrangian Relaxation
- Integer Linear Complementary Problem
- Integer Programming
- Integer Programming: Algebraic Methods
- Integer Programming: Branch and Bound Methods
- Integer Programming: Branch and Cut Algorithms
- Integer Programming: Cutting Plane Algorithms
- Integer Programming Duality
- Integer Programming: Lagrangian Relaxation
- LCP: Pardalos–Rosen Mixed Integer Formulation
- Mixed Integer Classification Problems
- Multi-objective Integer Linear Programming
- Multi-objective Mixed Integer Programming
- Multiparametric Mixed Integer Linear Programming
- Parametric Mixed Integer Nonlinear Optimization
- Set Covering, Packing and Partitioning Problems
- Simplicial Pivoting Algorithms for Integer Programming
- Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- Stochastic Integer Programs
- Time-dependent Traveling Salesman Problem

References

1. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP problems in process synthesis and design. Comput Chem Eng 21:S445–S450
2. Dakin RJ (1965) A tree search algorithm for mixed integer programming problems. Comput J 8:250–255
3. Gilmore PC, Gomory RE (1961) A linear programming approach to the cutting-stock problem. Oper Res 9:849–859
4. Haessler RW (1971) A heuristic programming solution to a non-linear cutting stock problem. Managem Sci 17:B793–B802
5. Harjunkoski I, Pörn R, Westerlund T, Skrifvars H (1997) Different strategies for solving bilinear integer problems with convex transformations. Comput Chem Eng 21:S487–S492
6. Harjunkoski I, Westerlund T, Isaksson J, Skrifvars H (1996) Different formulations for solving trim-loss problems in a paper converting mill with ILP. Comput Chem Eng 20:S121–S126
7. Hinxman AI (1980) The trim-loss and assortment problems: A survey. Europ J Oper Res 5:8–18
8. Johnston RE (1986) Rounding algorithms for cutting stock problems. Asia–Pacific J Oper Res 3:166–171
9. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
10. Skrifvars H, Harjunkoski I, Westerlund T, Kravanja Z, Pörn R (1996) Comparison of different MINLP methods applied on certain chemical engineering problems. Comput Chem Eng 20:S333–S338
11. Smith EMB, Pantelides CC (1997) Global optimization of nonconvex MINLPs. Comput Chem Eng 21:S791–S796
12. Westerlund T, Pettersson F (1995) An extended cutting plane method for solving convex MINLP problems. Comput Chem Eng 19:S131–S136

Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification

Han-Lin Li[1], Chang-Jui Fu[2]
[1] Institute of Information Management, National Chiao Tung University, Hsinchu, Taiwan
[2] Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

MSC2000: 90C08

Article Outline

Introduction/Background
Definitions
Formulation
Methods/Applications
Models
Program 1 (Nonlinear Mixed 0-1 Program)
Program 2 (Linear Mixed 0-1 Program)
Program 3
Cases
Finding CRP Binding Sites with a Given Pattern
Conclusions
References

Introduction/Background

In the past two decades, biologists have sequenced more and more complete genome sets for various species. To reveal secrets of life hiding in enormous genome data, the mechanism which conducts gene expression is continuously researched and discussed. Gene transcription, a primary gateway to gene function, is controlled by a complex regulatory mechanism. In this mechanism, many specific regulatory proteins bind to local regions of a gene upstream, called transcription factor binding sites (TFBS) or motifs, to control the gene expression. Therefore, the discrimination of TFBSs becomes an essential task for genome function analysis. Finding TFBSs is a challenging issue because motifs are mostly orientation- and position- independent to transcription starting points, and usually with some degree of ambiguity. Experimental methods like DNA microarray (DeRisi et al., 1997; Lockhart et al., 1996) and SAGE (Velculescu et al., 2000) are capable of precisely elucidating motifs, but too laborious and time consuming to analyze enormous genome data. More and more computer based methods - such as enumeration methods, probability models and heuristics - are being developed to help motif finding. The modeling of in silico motif finding has two parts: scoring function and algorithm. The simplest scoring function is given by summing up the number of base matches in a regulatory region. Generally it needs a predefined shared pattern for accuracy. Another scoring criterion is position-specific scoring matrices (PSSM) or its variant, information content (IC, Schneider et al., 1986), [44]. Though more computing is required, PSSM and IC are the most popular scoring functions, owing to their pattern-free property.

Current motif finding algorithms can generally be categorized as the probabilistic approaches and the deterministic approaches. Popular probabilistic algorithms are the expectation maximization [22], Gibbs sampling [21] and hidden Markov model (HMM). These are used to develop various sample-driven tools like MEME [3], CONSENSUS [17], AlignACE [19], ANN-spec [54], BioProspector [24], MotifSampler [48], GLAM [13], The Improbizer [1], QuickScore [38], SesiMCMC [11] and TFBSfinder [51].

There are many discrepancies among deterministic methods. A representative one is the consensus-based approach, [45] which tests all 4^m m-wide patterns and promises an optimal solution, but is very time consuming and impractical for large m [33,49]. Many heuristics are developed to prune the huge searching space, including testing only the substrings in the sequences [15,26], specifying a shared pattern to restrict the locations of mismatches [5,7,38,41], constructing suffix tree with fixed mismatches [30,31] and clustering approaches [6,23,34].

The methods for determining a consensus pattern can be split into two parts. The first part is the model for describing the shared pattern, and the second part is the algorithm for identifying the optimal consensus sequence according to its shared pattern. This study belongs to the second part. A consensus based motif finding problem is, given a set of sequences known to contain binding sites for a common factor but not knowing where the site are, to discover the location of the sites in each sequence [45].

Ecker et al. (2002) utilized optimization techniques to reformulate the maximum likelihood approach for motif finding problems. They adopted a probabilistic model and formulated a well-designed nonlinear model with reference to the expectation maximization algorithm of Lawrence and Reilly [22]. Their method, however, occasionally only finds a feasible solution or a local optimum, which means the best solution may not be found. Additionally, no further structural feature in the target motif can be embedded conveniently in their model.

Definitions

This study introduces a linear programming method for solving a motif finding problem to reach the globally optimal consensus sequence. Two examples of searching for CRP-binding sites and for FNR-binding sites in the *Escherichia coli* genome are used to illustrate the proposed method. The motif finding problem is firstly formulated as a nonlinear mixed 0-1 program for the alignment of DNA sequences; each of the four bases are coded with two binary variables and a matching score is designed. This nonlinear mixed 0-1 program is then converted into a linear mixed 0-1 program by linearization techniques. Owing to some special features of the binary relationships, this linear 0-1 program includes $2m$ binary variables where m is the number of active letters in the consensus. This method makes the number of binary variables independent of the number of sequences and the size of each sequence. That means the proposed method is computationally efficient in solving a motif finding problem with a large data size. Secondly, the proposed method is guaranteed to find the global optimum instead of a local optimum. Thirdly, many kinds of specific features accompanied with the target motif can be formulated as logical constraints and embedded into the linear program.

An example of searching CRP-binding sites, as discussed in Stormo et al. [44] and Ecker et al. (Ecker et al., 2002), is described as follows. Given eighteen letter sequences, each 105 positions long, where each position contains a letter from the set {A, T, C, G}, find a consensus sequence of length16 with the pattern

$$L_1L_2L_3L_4L_5 * * * * * * L_6L_7L_8L_9L_{10}$$

where $L_i \in${A, T, C, G} and $*$'s mean the positions of ignored letters.

Restated, the problem is to specify

(i) the L_i's of the consensus sequence pattern, and
(ii) the location of the site in each given sequence which can fit most closely the consensus sequence.

Formulation

This study firstly formulates a motif finding problem as a nonlinear mixed 0-1 program. This nonlinear mixed 0-1 program is then converted into a linear mixed 0-1 program using linearization techniques. To reduce the computational burden, many 0-1 variables in this linear mixed 0-1 program can actually be solved as continuous variables by an all or nothing assignment technique which greatly improves the computational efficiency of this program.

Here we use the example data in [44], as listed in Appendix, to describe the proposed method. First, we represent the data in Appendix as an 18*105 data matrix D:

$$D = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,105} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,105} \\ \vdots & \vdots & \ddots & \vdots \\ b_{18,1} & b_{18,2} & \cdots & b_{18,105} \end{bmatrix} \tag{1}$$

where $b_{l,p}$ is the letter in the position p of the sequence l.

Recall the example discussed in previous section: the consensus sequence we want to find has 16 positions (ten L_i's and six ignored letters). A sequence has 90 corresponding sites, so an 18*900 data matrix D' is generated from D.

$$D' = \begin{bmatrix} d^1_{1,1} & \cdots & d^{10}_{1,1} & d^1_{1,2} & \cdots & d^{10}_{1,2} & \cdots & d^1_{1,90} & \cdots & d^{10}_{1,90} \\ d^1_{2,1} & \cdots & d^{10}_{2,1} & d^1_{2,2} & \cdots & d^{10}_{2,2} & \cdots & d^1_{2,90} & \cdots & d^{10}_{2,90} \\ \vdots & & & \vdots & & & \ddots & & \vdots & \\ d^1_{18,1} & \cdots & d^{10}_{18,1} & d^1_{18,2} & \cdots & d^{10}_{18,2} & \cdots & d^1_{18,90} & \cdots & d^{10}_{18,90} \end{bmatrix} \tag{2}$$

where

$$d^i_{l,s} = \begin{cases} b_{l,i+s-1} & (\text{for } i = 1, 2, \ldots, 5) \\ b_{l,i+s+5} & (\text{for } i = 6, 7, \ldots, 10)\,, \end{cases}$$

and $s = 1 \ldots 90$ is the starting position of each candidate site.

Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification, Table 1
Base code in the determined consensus sequence

Base	u_i	v_i	a_i	t_i	c_i	g_i
A	0	0	1	0	0	0
T	1	1	0	1	0	0
C	0	1	0	0	1	0
G	1	0	0	0	0	1

For $L_i \in$ {A, T, C, G}, two binary variables u_i and v_i can be used to express L_i, an element of the consensus sequence, as shown in Table 1.

Table 1 indicates that if L_i is A, T, C, or G respectively, then $a_i = 1$, $t_i = 1$, $c_i = 1$ or $g_i = 1$, which implies following conditions.

$$\begin{aligned} a_i &= (1-u_i)(1-v_i) \\ t_i &= u_i v_i \\ c_i &= (1-u_i)v_i \\ g_i &= u_i(1-v_i) \end{aligned} \tag{3}$$

Now we let Score_l be the degree of fitting to the found consensus sequence, specified as

$$\text{Score}_l = \sum_{s=1}^{90} z_{l,s} \left(\theta_{l,s}^1 + \theta_{l,s}^2 + \ldots \cdots + \theta_{l,s}^{10}\right) \tag{4}$$

where $\theta_{l,s}^i$ is the element of candidate sites extracted from D'. The constraints associated with (4) are below:
(i)

$$\sum_{s=1}^{90} z_{l,s} = 1, \quad z_{l,s} \in \{0,1\} \text{ for all } l \text{ and } s\,. \tag{5}$$

(ii)

$$\theta_{l,s}^i = \begin{cases} a_i & \text{if } d_{l,s}^i = \text{A} \\ t_i & \text{if } d_{l,s}^i = \text{T} \\ c_i & \text{if } d_{l,s}^i = \text{C} \\ g_i & \text{if } d_{l,s}^i = \text{G}\,. \end{cases} \tag{6}$$

Clearly, $0 \le \text{Score}_l \le 10$, and the objective is to maximize the total sum of Score_l.

Methods/Applications

Consider the sample data in Fig. 1 for instance:

$$\begin{aligned} &\text{Score}_1 = \\ &z_{1,1}(a_1 + a_2 + g_3 + a_4 + c_5 + t_6 + t_7 + t_8 \\ &\quad + g_9 + a_{10}) \\ &+ z_{1,2}(a_1 + g_2 + a_3 + c_4 + t_5 + t_6 + t_7 + g_8 \\ &\quad + a_9 + t_{10}) \\ &z_{1,3}(g_1 + a_2 + c_3 + t_4 + g_5 + t_6 + g_7 + a_8 \\ &\qquad + t_9 + c_{10}) \end{aligned} \tag{7}$$

$$\begin{aligned} &\text{Score}_2 = \\ &z_{2,1}(g_1 + a_2 + t_3 + t_4 + a_5 + c_6 + g_7 + g_8 \\ &\quad + c_9 + g_{10}) \\ &+ z_{2,2}(a_1 + t_2 + t_3 + a_4 + t_5 + g_6 + g_7 + c_8 \\ &\quad + g_9 + t_{10}) \\ &+ z_{2,3}(t_1 + t_2 + a_3 + t_4 + t_5 + g_6 + c_7 + g_8 \\ &\qquad + t_9 + c_{10}) \end{aligned} \tag{8}$$

All $z_{l,s}$ in (4) are binary variables. Equation (5) implies that for a sequence l, only one site is chosen to contribute to Score_l. Suppose the kth site is selected, then $z_{l,k} = 1$ and $z_{l,s} = 0$ for all $s \in \{1, 2, \ldots, 90\}$, $s \ne k$. Since a huge amount of $z_{l,s}$ (i. e., $|l| * |s|$) are involved, to treat $z_{l,s}$ as binary variables would cause a heavy computational burden. Therefore $z_{l,s}$ should be resolved as continuous variables rather than binary variables. An important proposition is introduced below:

Proposition 1 (All or nothing assignment) *Let $z_{l,s} \ge 0$ be continuous variables instead of binary variables. If there is a k, $k \in \{1, 2, \ldots, 90\}$, such that $\sum_{i=1}^{10} \theta_{l,k}^i = \max\left\{\sum_{i=1}^{10} \theta_{l,s}^i \text{ for } s = 1, 2, \ldots, 90\right\}$, then assigning $z_{l,k} = 1$ and $z_{l,s} = 0$ for all $s \ne k$, $s \in \{1, 2, \ldots, 90\}$, can maximize the value of Score_l.*

Proof Since $\sum_s z_{l,s} = 1$ and $z_{l,s} \ge 0$, it is true that $\max\left\{\sum_s (z_{l,s} \sum_i \theta_{l,s}^i)\right\} \le \max\left\{\sum_i \theta_{l,s}^i \text{ for } s = 1, 2, \ldots, 90\right\} = \sum_i \theta_{l,k}^i$. □

Remark 1 The objective function $\sum_l Score_l$ can be rewritten as

$$f(x) = \sum_{i=1}^{10} \left\{ a_i \sum_{(l,s)\in SA_i} z_{l,s} + t_i \sum_{(l,s)\in ST_i} z_{l,s} + c_i \sum_{(l,s)\in SC_i} z_{l,s} + g_i \sum_{(l,s)\in SG_i} z_{l,s} \right\} \quad (9)$$

where $SA_i = \{(l,s)|d^i_{l,s} = A\}$, $ST_i = \{(l,s)|d^i_{l,s} = T\}$, $SC_i = \{(l,s)|d^i_{l,s} = C\}$, and $SG_i = \{(l,s)|d^i_{l,s} = G\}$ for $i = 1, 2, \ldots, 10$.

This result implies that SA_i (or ST_i, SC_i, SG_i) is a set composed of (l,s) in which the product term $z_{l,s}a_i$ (or $z_{l,s}t_i$, $z_{l,s}c_i$, $z_{l,s}g_i$ respectively) appears on the right hand side of (4) because $\theta^i_{l,s} = a_i$.

For instance, the sum of Score_1 and Score_2 in (7) and (8) becomes

$$\text{Score}_1 + \text{Score}_2 = a_1(z_{1,1} + z_{1,2} + z_{2,2}) + \ldots + a_{10}z_{1,1} + \cdots + g_1(z_{1,3} + z_{2,1}) + \cdots + g_{10}z_{2,1}\,. \quad (10)$$

Some logical constraints can be conveniently expressed by binary variables. For instance, the constraint that a CRP dimer binds a symmetrical site requires that

$$\text{if } L_i = \begin{cases} \text{A} & \text{then } L_{11-i} = \text{T}\,, \\ \text{C} & \text{then } L_{11-i} = \text{G}\,. \end{cases}$$

Such a logical structure can be conveniently formulated with the following constraints:

$$\left.\begin{aligned} u_i + u_{11-i} &= 1 \\ v_i + v_{11-i} &= 1 \end{aligned}\right\} \text{ for } i = 1, 2, 3, 4, 5 \quad (11)$$

where $u_i, v_i, u_{11-i}, v_{11-i} \in \{0, 1\}$.

With reference to Table 1, clearly if $L_i = A$ (i. e., $u_i = 0$ and $v_i = 0$) then $L_{11-i} = T$ (i. e., $u_{11-i} = 1$ and $v_{11-i} = 1$) and vice versa; (ii) if $L_i = C$ (i. e., $u_i = 0$ and $v_i = 1$) then $L_{11-i} = G$ (i. e., $u_{11-i} = 1$ and $v_{11-i} = 0$) and vice versa.

Models

A motif finding problem can be formulated as a nonlinear mixed 0-1 program based on these constraints:

Program 1 (Nonlinear Mixed 0-1 Program)

Maximize

$$\sum_{l=1}^{18} \text{Score}_l = \sum_{i=1}^{10} \left\{ a_i \sum_{(l,s)\in SA_i} z_{l,s} + t_i \sum_{(l,s)\in ST_i} z_{l,s} + c_i \sum_{(l,s)\in SC_i} z_{l,s} + g_i \sum_{(l,s)\in SG_i} z_{l,s} \right\} \quad (12)$$

subject to $\sum_{s=1}^{90} z_{l,s} = 1\,, \quad z_{l,s} \geq 0$ for all l, s

$$\left.\begin{aligned} a_i &= (1-u_i)(1-v_i) \\ t_i &= u_i v_i \\ c_i &= (1-u_i)v_i \\ g_i &= u_i(1-v_i) \end{aligned}\right\} \text{Conservative constraints for } i = 1, 2, \ldots, 10$$

$$\left.\begin{aligned} u_i + u_{11-i} &= 1 \\ v_i + v_{11-i} &= 1 \end{aligned}\right\} \text{Logical constraints for } i = 1, 2, \ldots, 5$$

$$u_i, v_i \in \{0, 1\} \quad \text{for } i = 1, 2, \ldots, 5$$

$$0 \leq u_i, v_i \leq 1 \quad \text{for } i = 6, 7, \ldots, 10$$

$$0 \leq a_i, t_i, c_i, g_i \leq 1 \quad \text{for } i = 1, 2, \ldots, 10\,.$$

This program intends to solve $\{a_i, t_i, c_i, g_i\}$ for $i = 1, 2, \ldots, 10$ thus to maximize the total degree of fitting to the consensus sequence for the given 18 sequences, subjected to a possible logical constraint. A very important feature of Program 1 is that we can treat $z_{l,s}$ as continuous variables rather than binary variables, which can improve the computational efficiency dramatically. We can ensure all found $z_{l,s}$ still have binary values as discussed in the next section.

Linearization of Program 1 Program 1 is a mixed nonlinear 0-1 program where $q_i \sum z_{l,s}$ for $q_i \in \{a_i, t_i, g_i, c_i\}$ and $u_i v_i$ are product terms. These product terms can be linearized directly by the following propositions:

Proposition 2 *The product term $\lambda_i = q_i \sum z_{l,s}$, where λ_i is to be maximized and $q_i \in \{0, 1\}$, can be linearized*

as follows:

$$\begin{aligned}
&\lambda_i \geq \sum z_{l,s} + M(q_i - 1)\\
&\lambda_i \geq 0\\
&\lambda_i \leq \sum z_{l,s}\\
&\lambda_i \leq M\, q_i
\end{aligned} \tag{13}$$

where M is a big constant larger than or equal to the number of sequences.

Proof If $q_i = 1$ then $\lambda_i = \sum z_{l,s}$; and otherwise $\lambda_i = 0$.

□

Proposition 3 *The product term $w_i = u_i v_i$, where $u_i, v_i \in \{0, 1\}$, can be linearized as follows:*

$$\begin{aligned}
&w_i \leq u_i\\
&w_i \ \leq v_i\\
&w_i \geq 0\\
&w_i \geq u_i + v_i - 1\,.
\end{aligned} \tag{14}$$

Denote $Z(a_i) = a_i \sum_{(l,s)\in SA_i} z_{l,s}$, $Z(t_i) = t_i \sum_{(l,s)\in ST_i} z_{l,s}$, $Z(c_i) = c_i \sum_{(l,s)\in SC_i} z_{l,s}$, and $Z(g_i) = g_i \sum_{(l,s)\in SG_i} z_{l,s}$. Program 1 is then linearized into Program 2 based on Proposition 2 and Proposition 3.

Program 2 (Linear Mixed 0-1 Program)

$$\begin{aligned}
\text{Maximize} \quad & \sum_{l=1}^{18} \text{Score}_l\\
&= \sum_{i=1}^{10} \big(Z(a_i) + Z(t_i) + Z(c_i)\\
&\quad + Z(g_i)\big)
\end{aligned} \tag{15}$$

subject to

$$\sum_{s=1}^{90} z_{l,s} = 1, \qquad z_{l,s} \geq 0 \quad \text{for all } l, s$$

$$\left.\begin{aligned}
&a_i = 1 - u_i - v_i + w_i\\
&t_i = w_i\\
&c_i = v_i - w_i\\
&g_i = u_i - w_i\\
&w_i \leq u_i\\
&w_i \leq v_i\\
&w_i \geq 0\\
&w_i \geq u_i + v_i - 1
\end{aligned}\right\} \text{Conservative constraints for } i = 1, 2, \ldots, 10$$

$$\left.\begin{aligned}
&u_i + u_{11-i} = 1\\
&v_i + v_{11-i} = 1
\end{aligned}\right\} \text{Logical constraints}$$

for $i = 1, 2, \ldots, 5$

$$\left.\begin{aligned}
&\sum_{(l,s)\in SA_i} z_{l,s} + M(a_i - 1) \leq Z(a_i) \leq \sum_{(l,s)\in SA_i} z_{l,s}\\
&0 \leq Z(a_i) \leq M\, a_i\\
&\sum_{(l,s)\in ST_i} z_{l,s} + M(t_i - 1) \leq Z(t_i) \leq \sum_{(l,s)\in ST_i} z_{l,s}\\
&0 \leq Z(t_i) \leq M\, t_i\\
&\sum_{(l,s)\in SC_i} z_{l,s} + M(c_i - 1) \leq Z(c_i) \leq \sum_{(l,s)\in SC_i} z_{l,s}\\
&0 \leq Z(c_i) \leq M\, c_i\\
&\sum_{(l,s)\in SG_i} z_{l,s} + M(g_i - 1) \leq Z(g_i) \leq \sum_{(l,s)\in SG_i} z_{l,s}\\
&0 \leq Z(g_i) \leq M\, g_i
\end{aligned}\right\} \text{Constraints for linearizing product terms}$$

$u_i, v_i \in \{0, 1\}$ for $i = 1, 2, \ldots, 5$

$0 \leq u_i, v_i \leq 1$ for $i = 6, 7, \ldots, 10$

$0 \leq a_i, t_i, c_i, g_i \leq 1$ for $= 1, 2, \ldots, 10$

$z_{l,s}$'s are treated as non-negative continuous variables for $l = 1, 2, \ldots, 18$ and $s = 1, 2, \ldots, 90$ where M can be any value greater than or equal to 18 .

In Program 2, since u_i and v_i are binary variables, a_i, t_i, c_i, and g_i should have binary values following (3). Although $z_{l,s}$ are treated as continuous variables, the values of $z_{l,s}$ should be 0 or 1. This is because the optimal solution of a linear program should be a vertex point satisfying $\sum_s z_{l,s} = 1$ for all l.

Consider the following proposition.

Proposition 4 *Let the optimal solution of Program 2 be $x^* = (Z^*, u^*, v^*)$ and $\sum_s z_{l,s} = 1$. Assume that a sequence l contains sites $s_1, s_2, \ldots, s_k$ such that*

$0 < z^*_{l,s_j} < 1$ *for* $j = 1, 2, \ldots, k$, *then,*

$$\sum_i \theta^i_{l,s_1} = \sum_i \theta^i_{l,s_2} = \cdots = \sum_i \theta^i_{l,s_k} = \max\left\{\sum_i \theta^i_{l,s}\right\},$$

where θ^i_{l,s_j} *are specified in (6) .*

Proof For $\sum_s z_{l,s} = 1$, if $s_p, s_q \in \{s_1, s_2, \ldots s_k\}$ where $\sum_i \theta^i_{l,s_p} > \sum_i \theta^i_{l,s_q}$, then to maximize $\text{Score}_l = \sum_{l,j} z_{l,s_j} \sum_i \theta^i_{l,s_j}$ requires $z_{l,s_q} = 0$. This conflicts with the observation that $0 < z_{l,s_q} < 1$, therefore $\sum_i \theta^i_{l,s_1} = \sum_i \theta^i_{l,s_2} = \cdots = \sum_i \theta^i_{l,s_k}$. □

After solving Program 2 we can obtain the globally optimum solution "TGTGA******TCACA" with objective value 147. The related nonzero $z_{l,s}$ values indicate the starting positions of the binding sites in the 18 sequences, as listed below:

$$z_{1,64} = z_{2,58} = z_{3,79} = z_{4,66} = z_{5,53} = z_{6,63} = z_{7,27} = z_{8,42} = z_{9,12} = z_{10,17} = z_{11,64} = z_{12,44} = z_{13,51} = z_{14,74} = z_{15,20} = z_{16,56} = z_{17,87} = z_{18,81} = 1$$

All other $z_{l,s}$'s have value 0.

In Program 2 the total number of 0-1 variables is $2m$ and the total number of the continuous variables is $20m + |l| * |s|$. Since the number of 0-1 variables is independent of the lengths of l and s, a motif finding problem with many long sequences can be solved effectively.

Suboptimal Consensus Sequences Program 2 can find the exact global optimum solution. Sometimes the second best and the third best solution may also be useful. It is very convenient for the proposed method to find a complete set of consensus sequences by adding some extra constraints. For instance, the second best solution of Program 2 can be obtained conveniently by solving the following program:

$$\text{Maximize} \quad \sum_{l=1}^{18} \text{Score}_l \tag{16}$$

subject to (i) The same constraints in Model 1

(ii) $t_1 + g_2 + t_3 + g_4 + a_5 + t_6 + c_7 + a_8 + c_9 + a_{10} \le 9$ (new constraint)

AAGACTGTTTTTTTGATC
GATTATTTGCACGGCGTC

a

$l = 1, s = 1$ AAGACTGTTTTTTTGATC
$l = 1, s = 2$ AAGACTGTTTTTTTGATC
$l = 1, s = 3$ AAGACTGTTTTTTTGATC
$l = 2, s = 2$ GATTATTTGCACGGCGTC
$l = 2, s = 3$ GATTATTTGCACGGCGTC

b

$$\begin{bmatrix} AAGACTTTGA & AGACTTTGAT & GACTGTGATC \\ GATTACGGCG & ATTATGGCGT & TTATTGCGTC \end{bmatrix}$$

c

Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification, Figure 1
A small example of finding consensus sequence: a two sequences to be compared; b Schematic representation of the candidate sites; c The associated *D′* matrix

The new constraint is used to force the program to find a new solution different from the solution of Program 2. The found second best consensus sequence is "TTTGA******TCAAA" with score 129. Similarly we can find another solution by adding following constraint into (16).

$$t_1 + t_2 + t_3 + g_4 + a_5 + t_6 + c_7 + a_8 + a_9 + a_{10} \le 9$$

The found third best consensus sequence is "AAATT******AATTT" with score 129.

Extend to Find Unknown Binding Sites A more complicated motif finding problem is to search for the consensus sequence with an uncertain pattern format where the number of ignored letters between the two half sites is unknown. An example is to find a consensus sequence of length $2 * 5 + k$ with the pattern

$$L_1 L_2 L_3 L_4 L_5 * \cdots * L_6 L_7 L_8 L_9 L_{10}$$

where k, the number of $*$'s, is an unknown integer between 0 and 10.

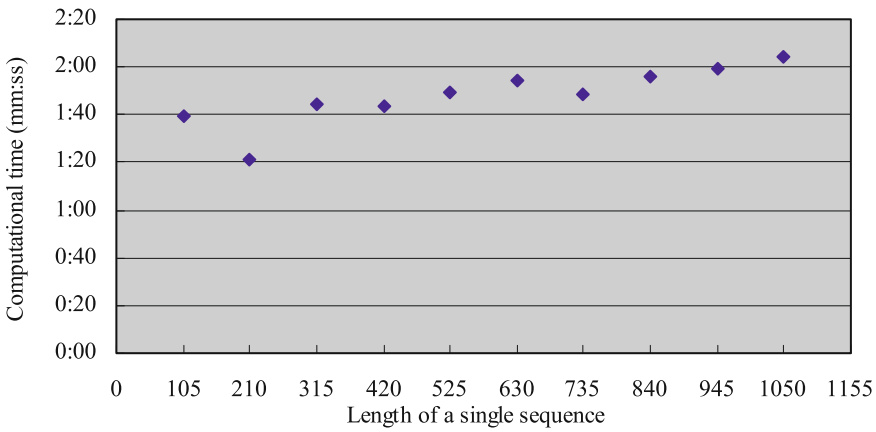
Program 2 can be modified slightly to treat this type of motif finding problem. Firstly we expand D in (1) as D' below:

$$D' = [D'(0)D'(1)D'(2) \ldots\ldots D'(10)]$$

Computational time versus sequence length

Sequence Length	Solving Time (mm:ss)
105	1:39
210	1:21
315	1:44
420	1:43
525	1:48
630	1:54
735	1:48
840	1:56
945	1:59
1050	2:04

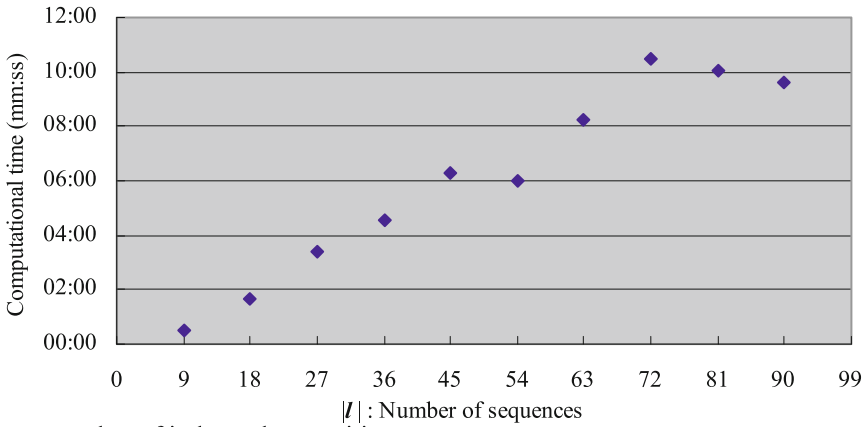
a



Computational time versus number of sequences

Number of Sequences	Solving Time (mm:ss)
9	0:30
18	1:39
27	3:21
36	4:32
45	6:15
54	6:01
63	8:16
72	10:29
81	10:01
90	9:37

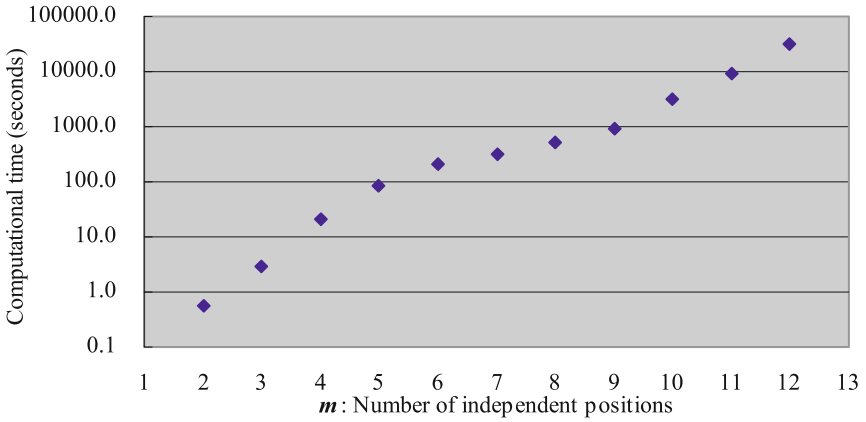
b



Computational time versus number of independent positions

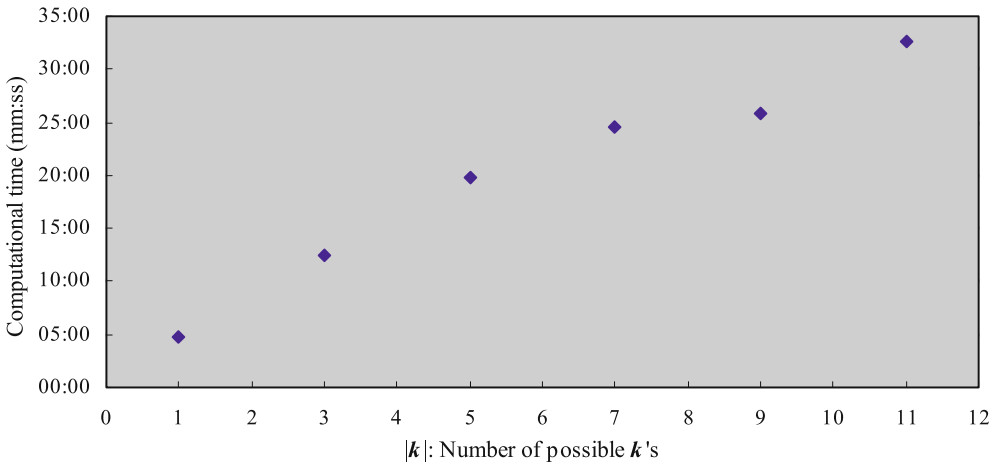
Number of Indep Pos	Solving Time (h:mm:ss)
2	0:00:01
3	0:00:03
4	0:00:21
5	0:01:23
6	0:03:38
7	0:05:18
8	0:08:25
9	0:15:52
10	0:53:27
11	2:33:20

c



Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification, Figure 2
The relationship between computational time and various factors involved in a consensus based motif finding problem. This figure illustrates the computational time of solving Program 2 with a various sequences sizes; b various number of sequences and c various independent positions

$\bar{k}$	$\|k\|$	Common Site	Score	Computational Time
0	1	**TGTTT(0)AAACA**	126	4:51
2	3	**TGAAA(2)TTTCA**	129	12:32
4	5	**GTGAA(4)TTCAC**	134	19:46
6	7	**TGTGA(6)TCACA**	147	24:28
8	9	**TGTGA(6)TCACA**	147	25:49
10	11	**TGTGA(6)TCACA**	147	32:35



Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification, Figure 3
Computational time of Program 3 with various numbers of possible *k*'s. The number enclosed in the common site is the solution of *k*

in which

$$D'(k) = \begin{bmatrix} d^1_{1,1,k} \cdots d^{10}_{1,1,k} & d^1_{1,2,k} \cdots d^{10}_{1,2,k} & \cdots & d^1_{1,90,k} \cdots d^{10}_{1,90,k} \\ d^1_{2,1,k} \cdots d^{10}_{2,1,k} & d^1_{2,2,k} \cdots d^{10}_{2,2,k} & \cdots & d^1_{2,90,k} \cdots d^{10}_{2,90,k} \\ \vdots & \vdots & \ddots & \vdots \\ d^1_{18,1,k} \cdots d^{10}_{18,1,k} & d^1_{18,2,k} \cdots d^{10}_{18,2,k} & \cdots & d^1_{18,90,k} \cdots d^{10}_{18,90,k} \end{bmatrix}$$

where $k \in \{0, 1, \ldots, 10\}$.

$$d^i_{l,s,k} = \begin{cases} b_{l,i+s-1} & (\text{for } i = 1, 2, 3, 4, 5) \\ b_{l,i+s+k-1} & (\text{for } i = 6, 7, 8, 9, 10) \end{cases}$$

$\theta^i_{l,s.k} = a_i$, t_i, c_i or g_i when $d^i_{l,s.k} =$ 'A', 'T', 'C', or 'G' respectively .

The cases with k larger than 10 are not considered since they are relatively rare. A linear mixed 0-1 program for solving this example is formulated below:

Program 3

$$\text{Maximize} \quad \sum_{i=1}^{2m} \left(Z(a_i) + Z(t_i) + Z(c_i) + Z(g_i)\right) \quad (15)$$

subject to

(i) $$\sum_{k=0}^{10} \sum_{s=1}^{96-k} z_{l,s,k} = 1 \,,$$
$$z_{l,s,k} \geq 0 \text{ for all } l, s, k$$

(ii) $$\sum_s z_{1,s,k} = \sum_s z_{2,s,k} = \ldots = \sum_s z_{18,s,k} \text{ for } k \in \{0, 1, \ldots, 10\}$$

(iii) the same conservative and logical constraints in Program 2

(iv) the same constraints for linearizing product terms in Program 2 but replace $z_{l,s}$ by $z_{l,s.k}$.

Constraints (i) and (ii) are used to ensure that when a specific k is chosen then $\sum_s z_{l,s,k} = 1$ and $\sum_s z_{l,s,k'} = 0$ for $k' \neq k$.

Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification, Table 2
FNR binding sites found by Program 3

Operon	Seq. length	Site seq. found by Program 3	Predicted Position	Score	Site seq. listed in RegulonDB*	Center Position
Common site: TTGAT----ATCAA						
narK	338	ATGAT----ATCAA	-86	9	actatgGGTAATGATAAATATCAATGATagataa	-79.5
		TTGAT----ATCAA	-48	10	atcttaTCGTTTGATTTACATCAAATTGccttta	-41.5
ansB	345	TTGTT----GTCAA	-48	8	acgttgTAAATTGTTTAACGTCAAATTTcccata	-41.5
		TTGTA----TCCAA	-81	6	gcctctAACTTTGTAGATCTCCAAAATAtattca	-74.5
		TTTAT----TTTAA	-123	7		
narG	525	TTGAT----ATCAA	-55	10	ctcttgATCGTTATCAATTCCCACGCTGtttcag	-41.5
dmsA	325	TTGAT----AACAA	-48	9	cttttgaTACCGAACAATAATTACTCCTCacttac	-33
frd	781	TTCAG----ATCCA	-37	7	AAAAATCGATCTCGTCAAATTTcagacttatcca	-47
		TTAAT----TTCAG	-98	7		
nirB	262	TTGAT----ATCAA	-48	10	aaaggtGAATTTGATTTACATCAATAAGcggggt	-41.5
sodA	284	TTGAT----ATTTT	-42	7	agtacgGCATTGATAATCATTTTCAATAtcattt	-34
fnr**	96	TTGAC----ATCAA	-7	9	atgttaAAATTGACAAATATCAATTACGgcttga	1
					ccttaaCAACTTAAGGGTTTTCAAATAGatagac	-103.5
(cyoA)	599	CTTCT----ATCAA	-113	7	N/A	N/A
		TTGTT----TTCAC	-198	7		
(icdA)	290	ATGAC----AACAA	16	7	N/A	N/A
		TTGCT----AGCAT	73	7		
(sdhC)	708	TTGAT----AATAA	-330	8	N/A	N/A
(ulaA)	346	TCAAT----ATCAA	-278	8	N/A	N/A
		TTGGT----ATTAA	-257	8		

* For visualizing the comparison, the letters in uppercase represent the binding site listed in RegulonDB; the letter in bold face is the center of the site sequence; and the encompassed letters represent the exact binding site obtained by Program 3.
** The second site listed in RegulonDB is not contained in the sequence data, which is only 96 bases long, from GenBank.

Cases

Finding CRP Binding Sites with a Given Pattern

Several experiments are tested here, using the example in the Appendix, to analyze the effect of sequence length and number of sequences on the computational time. All examples are solved by LINGO [40], a widely used optimization software, on a personal computer with a Pentium 4 2.0G CPU. A software package named "Global Site Seer" is developed based on Program 2 for finding DNA motifs. This software is available from http://www.iim.nctu.edu.tw/~cjfu/gss.htm.

Figure 2 illustrates the experimental results for analyzing the time complexity. Figure 2a is the computational time given various sequence lengths, where the number of sequences is fixed at 18. The results show that the computational time changes only slightly even if the sequence length is increased from 105 to 1050. Figure 2b is the computational time with various numbers of sequences. It shows that the solving time is roughly proportional to the number of sequences. The proposed model is quite promising for finding DNA motifs in a dataset with a large sequence length and a large number of sequences. Figure 2c shows that the computational time rises exponentially as the number of independent positions increases.

Using Program 3 to search CRP binding sites, we obtain the globally optimal solution "TGTGA******TCACA" with score 147, which is exactly the same solution found in Program 2. The second best solution is "GTGAA****TTCAC" with score 134. The relationship between the computational time and the number of possible k's (i. e. $|k|$) is linear, as shown in the experiment result listed in Fig. 3. The number of ignored letters k is between 0 and $\bar{k}$, the upper bound of k, and thus we have $|k| = \bar{k} + 1$ in this experiment.

Finding FNR-binding Sites with an Ambiguous Shared Pattern Program 3 is also applied to solve an example of searching for binding sites of fumarate and nitrate reduction regulatory protein (FNR) in *E. coli*. Both CRP and FNR belong to the CRP/FNR helix-turn-helix transcription factor superfamily [47]. The sequence data, which is taken from GenBank, contains 12 DNA sequences with lengths varied from 96 to 781. Owing to the dimer structure of the binding protein, the consensus sequence in this example also has a constraint of inverse symmetry. The RegulonDB database [18] lists the found regulatory binding sites for eight of these twelve sequences while the exact positions of the other four sequences are not listed yet. Solving this example by Program 3 we obtained the global optimal consensus sequence as "TTGAT****ATCAA" with score 107, which is the same consensus sequence as indicated by [47]. Table 2 illustrates the result including the consensus sequence and the predicted binding sites for all of the 12 sequences. Some sites downstream of the transcription start (i. e. with positive indices) are also listed because there are a few known cases in which regulatory sites appear within transcription units [47]. The proposed method has found some sites not listed in RegulonDB, but which have scores higher than those listed in RegulonDB (e. g. the third solution in the Operon *ansB* row of Table 2). The best predicted sites in the four undetermined sequences are also listed in Table 2.

Conclusions

This study proposes a linear mixed 0-1 programming approach for finding DNA motifs. Compared to the widely used maximum likelihood methods, the proposed method can reach a global optimum rather than finding a local optimum or a feasible solution. Additionally, by utilizing binary variables, some logical constraints can be embedded into the models. It is also convenient to find the complete set of the second, third, etc. best consensus sequences. Since the number of binary variables is fully independent of the number of sequences and the length of a sequence, the proposed method can treat motif finding problems with many long sequences. For finding motifs with many independent positions in an acceptable time, this study also proposes a method for distributed computing.

The proposed method can also be conveniently extended to treat more complicated motif finding problems. In this study an extension of the linear program is designed to find DNA motifs with an unknown number of ignored letters between the two half sites. The result of searching for FNR-binding sites shows that the extended model can find not only the locations of known binding sites listed in the RegulonDB database but also those not yet delimitated.

References

1. Ao W, Gaudet J, Kent WJ, Muttumu S, Mango SE (2004) Environmentally induced foregut remodeling by PHA-4/FoxA and DAF-12/NHR. Science 305(5691):1743–1746
2. Bailey T, Elkan C (1995) Unsupervised learning of multiple motifs in biopolymers using expectation maximization. Mach Learn 21(1–2):51–80
3. Bailey T, Elkan C (1995) The value of prior knowledge in discovering motifs with MEME. In: Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology, AAAI Press, Menlo Park, pp 21–29
4. Blanchette M, Schwikowski B, Tompa M (2000) An exact algorithm to identify motifs in orthologous sequences from multiple species. In: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, San Diego, pp 37–45
5. Brāzma A, Jonassen I, Eidhammer I, Gilbert D (1998) Approaches to the automatic discovery of patterns in biosequences. J Comput Biol 5(2):279–305
6. Buhler J, Tompa M (2002) Finding Motifs Using Random Projections. J Comput Biol 9(2):225–242
7. Califano A (2000) SPLASH: structural pattern localization analysis by sequential histograms. Bioinformatics 16(4):341–357
8. DeRisi J, Iyer V, Brown P (1997) Exploring the metabolic and genetic control of gene expression on a genomic scale. Science 278(5338):680–686
9. Ecker JG, Kupferschmid M, Lawrence CE, Reilly AA, Scott ACH (2002) An application of nonlinear optimization in molecular biology. Eur J Oper Res 138(2):452–458
10. Eskin E, Pevzner P (2002) Finding composite regulatory patterns in DNA sequences. Bioinformatics (Supplement 1) 18(1):S354–S363
11. Favorov AV, Gelfand MS, Gerasimova AV, Mironov AA, Makeev VJ (2004) Gibbs sampler for identification of symmetrically structured, spaced DNA motifs with improved estimation of the signal length and its validation on the ArcA binding sites. In: Proceedings of BGRS 2004, BGRS, Novosibirsk
12. Fratkin E, Naughton BT, Brutlag DL, Batzoglou S (2006) MotifCut: Regulatory motifs finding with maximum density subgraphs. Bioinformatics 22(14):e150–157

13. Frith MC, Hansen U, Spouge JL, Weng Z (2004) Finding functional sequence elements by multiple local alignment. Nucl Acids Res 32(1):189–200
14. Galas D, Eggert M, Waterman M (1985) Rigorous pattern-recognition methods for DNA sequences: analysis of promoter sequences from Escherichia coli. J Mol Biol 186(1):117–128
15. Gelfand M, Koonin E, Mironov A (2000) Prediction of transcription regulatory sites in archaea by a comparative genomic approach. Nucl Acids Res 28(3):695–705
16. Hertz GZ, Hartzell GW, Stormo GD (1990) Identification of consensus patterns in unaligned DNA sequences known to be functionally related. Comput Appl Biosci 6(2):81–92
17. Hertz GZ, Stormo GD (1999) Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. Bioinformatics 15(7–8):563–577
18. Huerta AM, Salgado H, Thieffry D, Collado-Vides J (1998) RegulonDB: a database on transcriptional regulation in Escherichia coli. Nucl Acids Res 26(1):55–59
19. Hughes JD, Estep PW, Tavazoie S, Church GM (2000) Computational identification of cis-regulatory elements associated with functionally coherent groups of genes in Saccharomyces cerevisiae. J Mol Biol 296(5):1205–1214
20. Krause M, Park M, Zhang JM, Yuan J, Harfe B, Xu SQ, Greenwald I, Cole M, Paterson B, Fire A (1997) A C. elegans E/Daughterless bHLH protein marks neuronal but not striated muscle development. Development 124(11):2179–2189
21. Lawrence CE, Altschul S, Boguski M, Liu J, Neuwald A, Wootton J (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. Science 262(5131):208–214
22. Lawrence CE, Reilly AA (1990) An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. PROTEINS: Struct, Funct Genet 7(1):41–51
23. Liang S, Samanta M, Biegel B (2004) cWINNOWER algorithm for finding fuzzy DNA motifs. J Bioinform Comput Biol 2(1):47–60
24. Liu XS, Brutlag DL, Liu JS (2001) BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. Pacific Symposium on Biocomputing, pp 127–138
25. Li HL, Fu CJ (2005) A linear programming approach for identifying a consensus sequence on DNA sequences. Bioinformatics 21(9):1838–1845
26. Li M, Ma B, Wang L (1999) Finding similar regions in many strings. In: Proceedings of the 31st ACM Annual Symposium on Theory of Computing, pp 473–482
27. Lockhart DJ, Dong H, Byme MC, Follettie MT, Gallo MV, Chee MS, Mittmann M, Wang C, Kobayashi M, Horton H, Brown EL (1996) Expression monitoring by hybridization to high-density oligonucleotide arrays. Nat Biotechnol 14(13):1675–1680
28. Matys V, Fricke E, Geffers R, Gößling E, Haubrock M, Hehl R, Hornischer K, Karas D, Kel AE, Kel-Margoulis OV, Kloos DU, Land S, Lewicki-Potapov B, Michael H, Münch R, Reuter I, Rotert S, Saxel H, Scheer M, Thiele S, Wingender E (2003) TRANSFAC: transcriptional regulation, from patterns to profiles. Nucl Acids Res 31(1):374–378
29. Needleman S, Wunsch C (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48(3):443–453
30. Pavesi G, Mauri G, Pesole G (2001) An algorithm for finding signals of unknown length in DNA sequences. ISMB 2001 Bioinformatics 17(Suppl 1):S207–214
31. Pavesi G, Mereghetti P, Mauri G, Pesole G (2004) Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. Nucl Acids Res 32(Web Server Issue):W199–W203
32. Peng CH, Hsu JT, Chung YS, Lin YJ, Chow WY, Hsu DF, Tang CY (2006) Identification of Degenerate Motifs Using Position Restricted Selection and Hybrid Ranking Combination. Nucl Acids Res 34:6379–6391
33. Pesole G, Prunella N, Liuni S, Attimonelli M, Saccone C (1992) WORDUP: an efficient algorithm for discovering statistically significant patterns in DNA sequences. Nucl Acids Res 20(11):2871–2875
34. Pevzner P, Sze H (2000) Combinatorial approaches to finding subtle signals in DNA sequences. In: Proceedings International Conference on Intelligent Systems for Molecular Biology, La Jolla, 20–23 August 2000, pp 269–278
35. Portman DS, Emmons SW (2000) The basic helix-loop-helix transcription factors LIN-32 and HLH-2 function together in multiple steps of a C. elegans neuronal sublineage. Development 127(24):5415–5426
36. Rajasekaran S, Hu Y, Luo J, Nick H, Sahni S, Shaw S (2001) Efficient algorithms for similarity search. J Comb Optim 5(1):125–132
37. Rajasekaran S, Sahni S, Shaw S (2001) Efficient algorithms for local alignment search. J Comb Optim 5(1):117–124
38. Régnier M, Denise A (2004) Rare events and conditional events on random strings. Discret Math Theor Comput Sci 6(2):191–214
39. Schneider TD, Stormo GD, Gold L, Ehrenfeucht A (1986) Information content of binding sites on nucleotide sequences. J Mol Biol 188(3):415–431
40. Schrage L (1999) Optimization Modeling With Lingo. LINDO Systems Inc., Chicago
41. Sinha S, Tompa M (2003) Performance comparison of algorithms for finding transcription factor binding sites. In: Bourbakis NG (ed) 3rd IEEE Symposium on Bioinformatics and Bioengineering, IEEE Computer Society, New York, 2003, pp 214–220
42. Sinha S, Tompa M (2003) YMF: a program for discovery of novel transcription factor binding sites by statistical overrepresentation. Nucl Acids Res 31(13):3586–3588
43. Stein L, Sternberg P, Durbin R, Thierry-Mieg J, Spieth J (2001) WormBase: network access to the genome and bi-

ology of Caenorhabditis elegans. Nucl Acids Res 29(1): 82–86
44. Stormo GD, Hartzell GW (1989) Identifying protein-binding sites from unaligned DNA fragments. In: Proceedings of the National Academy of Sciences of the USA, 86(4), pp 1183–1187
45. Stormo GD (2000) DNA binding sites: representation and discovery. Bioinformatics 16(1):16–23
46. Swoboda P, Adler HT, Thomas JH (2000) The RFX-type transcription factor DAF-19 regulates sensory neuron cilium formation in C. elegans. Mol Cell 5(3):411–421
47. Tan K, Moreno-Hagelsieb G, Collado-Vides J, Stormo GD (2001) A comparative genomics approach to prediction of new members of regulons. Genome Res 11(4):566–584
48. Thijs G, Lescot M, Marchal K, Rombauts S, De Moor B, Rouzé P, Moreau Y (2001) A higher-order background model improves the detection of promoter regulatory elements by Gibbs sampling. Bioinformatics 17(12):1113–1122
49. Tompa M (1999) An exact method for finding short motifs in sequences with application to the Ribosome Binding Site problem. In: Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology, Heidelberg, 6–August 2000, pp 262–271
50. Tompa M, Li N, Bailey TL, Church GM, De Moor B, Eskin E, Favorov AV, Frith MC, Fu Y, Kent WJ, Makeev VJ, Mironov AA, Noble WS, Pavesi G, Pesole G, Regnier M, Simonis N, Sinha S, Thijs G, van Helden J, Vandenbogaert M, Weng Z, Workman C, Ye C, Zhu Z (2005) Assessing Computational Tools for the Discovery of Transcription Factor Binding Sites. Nat Biotechnol 23(1):137–144
51. Tsai HK, Huang GTW, Chou MY, Lu HHS, Li WH (2006) Method for identifying transcription factor binding sites in yeast. Bioinformatics 22(14):1675–1681
52. Velculescu VE, Zhang L, Vogelstein B, Kinzler KW (1995) Serial analysis of gene expression. Science 270(5235): 484–487
53. Waterman M, Galas D, Arratia R (1984) Pattern recognition in several sequences: consensus and alignment. Bull Math Biol 46(4):512–527
54. Workman CT, Stormo GD (2000) ANN-Spec: a method for discovering transcription factor binding sites with improved specificity. In: Altman R, Dunker AK, Hunter L, Klein TE (eds) Pacific Symposium on Biocomputing. Stanford University, Stanford, pp 467–478

Mixed Integer Classification Problems

PAUL A. RUBIN
The Eli Broad Graduate School of Management, Michigan State University, East Lansing, USA

MSC2000: 62H30,68T10,90C11

Article Outline

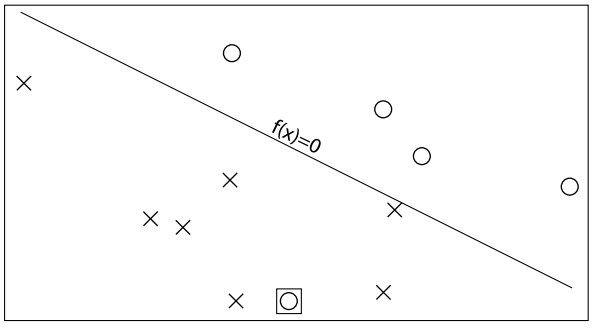
Keywords and Phrases
Introduction
Formulation
  Multiple Groups
Methods
See also
References

Keywords and Phrases

Classification; Discriminant analysis; Integer programming

Introduction

The *G-group classification problem*, also known as the *G-group discriminant problem*, involves a population partitioned into G distinct (and predefined) groups. The object is to construct a scalar- or vector- valued *scoring function* $f : \Re^p \to \Re$ so that the group to which a population member with observed attributes $\mathbf{x} \in \Re^p$ belongs can be determined, with best possible accuracy, from its score $f(\mathbf{x})$. The scoring function f is usually restricted to a particular class (most commonly, linear). By a wide margin, the majority of studies have focused on the two-group case. Construction of f is based on *training samples* from the various groups. The most reasonable criterion for choosing f may be expected misclassification cost, but many studies make the simplifying assumptions that all misclassifications are equally expensive and that groups are represented in the training samples in proportion to their prior probability of being encountered, in which case the criterion reduces



Mixed Integer Classification Problems, Figure 1
Optimal linear classifier (□ = misclassified)

to minimizing the number of misclassifications in the combined training samples. Figure 1 illustrates an optimal choice of linear classifier f in a two-group problem.

Classical discriminant analysis relies on distributional assumptions. In the two-group case with normally distributed attributes, the scalar-valued discriminant function that minimizes expected misclassification cost is known to be linear if the two groups have identical covariance structures and quadratic if not. In both cases, direct estimation of f is straightforward. Nonparametric approaches, making no distributional assumptions, have utilized an eclectic assortment of techniques, among them neural networks, metaheuristics, and mathematical programming. Although some consideration has been given to nonlinear programming methods, the bulk of the work involving mathematical programming has utilized either linear or mixed integer linear programming models, or support vector machines (quadratic programs) [8]. See [10] and ► linear programming models for classification (elsewhere in this volume) for an overview of the subject.

When f is linear, the problem of minimizing the number of misclassifications is a special case of the slightly more general problem of dropping the smallest (or least costly) set of constraints necessary to render an inconsistent set of linear inequalities consistent. This problem crops up in a variety of contexts, including pattern recognition [18], machine learning/data mining [5] and the analysis of infeasible linear programs [6]. Thus methods from those areas may be applicable to discriminant problems. For instance, Soltysik and Yarnold [15] applied the algorithm of Warmack and Gonzalez [18] to the two-group linear discriminant problem.

Formulation

The following is a typical mixed integer programming model for the two-group case, using a scalar linear discriminant function:

$$\begin{aligned} \min \quad & \sum_{g=1}^{2} \frac{\pi_g C_g}{N_g} \sum_{n=1}^{N_g} z_{gn} \\ \text{s.t.} \quad & \mathbf{X}_1\mathbf{w} + w_0 \cdot \mathbf{1} - M \cdot \mathbf{z}_1 \quad \leq \mathbf{0} \\ & \mathbf{X}_2\mathbf{w} + w_0 \cdot \mathbf{1} + M \cdot \mathbf{z}_2 \quad \geq \mathbf{0} \\ & \mathbf{w}, w_0 \text{ free}; \mathbf{z}_g \in \{0,1\}^{N_g} . \end{aligned} \tag{1}$$

Matrix $\mathbf{X}_g$ is an $N_g \times p$ training sample from group g, while π_g and C_g are respectively the prior probability of group g and the cost of misclassifying a member of that group. M is a sufficiently large positive constant, and $\mathbf{0}$ and $\mathbf{1}$ denote vectors, all of whose entries are respectively 0 or 1. The discriminant function $f(\mathbf{x}) = \mathbf{w}'\mathbf{x} + w_0$ is intended to produce negative scores for members of the first group and positive scores for members of the second group. Bivalent indicator variable z_{gn} takes value 1 if the nth training observation from group g is classified incorrectly and 0 if it is classified correctly.

The discriminant function is linear as written, but various nonlinear functions can be generated by embedding the attribute space $\Re^p$ in a higher-dimensional space. Support vector machines are particularly adept at this. Polynomial functions, for instance, are easily accommodated in (1) by expanding the sample matrices to include powers and products of attributes.

A score of zero results in an ambiguous classification. Some authors deal with this by changing the first two constraints of (1) to

$$\begin{aligned} \mathbf{X}_1\mathbf{w} + w_0 \cdot \mathbf{1} - M \cdot \mathbf{z}_1 \quad &\leq \quad -\varepsilon \cdot \mathbf{1} \\ \mathbf{X}_2\mathbf{w} + w_0 \cdot \mathbf{1} + M \cdot \mathbf{z}_2 \quad &\geq \quad +\varepsilon \cdot \mathbf{1} , \end{aligned}$$

where ε is a small positive constant. This formulation is nearly as general, although it is mathematically possible that infelicitous choices of ε and M could rule out an otherwise desirable solution.

Problem (1) is known to be *NP*-hard [1]. At the same time, using the finite VC-dimension of linear classifiers [16,17], it can be shown that the error rate of the solution to (1) converges in probability to the optimal error rate as sample size grows [2]. Assuming availability of sufficient data, a key question is whether the problem remains tractable when the training sample is large enough to provide a suitably accurate solution. There is grounds for (cautious) optimism, in that progress in hardware, software and algorithms advances the boundaries of what is tractable, while for a given problem instance the sample size needed for accuracy is static.

While there will often be a unique best choice of training observations to misclassify (i. e., unique optimal values of $\mathbf{z}_1$ and $\mathbf{z}_2$), there commonly will be infinitely many choices for the coefficients $\mathbf{w}$, w_0 of a discriminant function that misclassifies those observations

only. To select from among those coefficient solutions, authors often introduce additional terms in the objective function. As an example, Bajgier and Hill [4] used a formulation similar to the following:

$$\begin{aligned}
\min \quad & \sum_{g=1}^{2} \frac{\pi_g}{N_g} \sum_{n=1}^{N_g} \left[C_g z_{gn} + \varepsilon_1 d_{gn}^- - \varepsilon_2 d_{gn}^+ \right] \\
\text{s.t.} \quad & \mathbf{X}_1 \mathbf{w} + w_0 \cdot \mathbf{1} + \mathbf{d}_1^+ - \mathbf{d}_1^- - M \cdot \mathbf{z}_1 \quad \leq \mathbf{0} \\
& \mathbf{X}_2 \mathbf{w} + w_0 \cdot \mathbf{1} - \mathbf{d}_2^+ + \mathbf{d}_2^- + M \cdot \mathbf{z}_2 \quad \geq \mathbf{0} \\
& \mathbf{w}, w_0 \text{ free}; \ \mathbf{d}_g^+, \mathbf{d}_g^- \ \geq \ \mathbf{0}; \ \mathbf{z}_g \in \{0,1\}^{N_g} .
\end{aligned}$$

The deviation variables $\mathbf{d}_g^+$ and $\mathbf{d}_g^-$ measure the amount by which each score falls on the correct and incorrect side of the zero cutoff, respectively. The objective functions rewards the former and penalizes the latter, using small positive objective coefficients ε_1 and ε_2 to prevent improvements in these terms from inducing unnecessary misclassifications.

The motivation for formulation (1) is simple: if the training samples are representative of the overall population, the discriminant function that minimizes misclassification costs on the training samples should come close to minimizing expected misclassification cost on the overall population. Models like (1) tend to be computationally expensive, however. As is typical with mixed integer programming models, computation time increases modestly with the number of attributes (p) but more dramatically with the number of zero-one variables ($N_1 + N_2$, the combined sample size). Moreover, the constant M must be chosen large enough that the best choice of $\boldsymbol{w}$ and w_0 is not rendered infeasible by a misclassified observation with score larger than M in magnitude; but the larger M is, the weaker the bounds in a branch-and-bound solution of the problem, and thus the longer the solution time. Codato and Fischetti [7] reported success using a form of Benders cut to eliminate M.

In the special case where all attribute variables are discrete, it is likely that some observation vectors will appear more than once in the training samples. When that occurs, the number of zero-one variables can be reduced from one per observation to one per *distinguishable* observation, yielding a variation of (1) in which the objective function is replaced with

$$\min \quad \sum\nolimits_{g=1}^{2} \frac{\pi_g C_g}{N_g} \sum\nolimits_{k=1}^{K_g} N_{gk} z_{gk} .$$

In this formulation [3], K_g is the number of distinct attribute vectors $\mathbf{x}$ in the training sample from group g, N_{gk} is the number of repetitions of the kth distinct observation from group g, and the matrices $\mathbf{X}_g$ contain only one copy of each such observation.

Multiple Groups

When $G > 2$ groups are involved, the problem becomes considerably more complicated. In a practical application with multiple groups, it is plausible that misclassification costs would depend not only on the group to which a misclassified point belonged but also the one into which it was classified. Thus an appropriate objective function might look like

$$\sum_{g=1}^{G} \frac{\pi_g}{N_g} \sum_{\substack{h=1 \\ h \neq g}}^{G} C_{gh} \sum_{n=1}^{N_g} z_{ghn} ,$$

where C_{gh} is the cost of classifying a point from group g into group h and z_{ghn} is 1 if the nth observation of group g is classified into group h and 0 otherwise. This represents a substantial escalation of the number of indicator variables. As a consequence, most research on the multiple group problem assumes that misclassification costs depend only on the correct group.

Few models, and fewer computational results, have been published for the multiple group problem. Gehrlein [9] presented one of the earliest scalar-valued mixed integer models for the case $G > 2$. The range of his discriminant function is partitioned into separate intervals corresponding to the groups. His model, adapted to the preceding notation, is

$$\begin{aligned}
\min \quad & \sum_{g=1}^{G} \frac{\pi_g C_g}{N_g} \sum_{n=1}^{N_g} z_{gn} \\
\text{s.t.} \quad & \mathbf{X}_g \mathbf{w} + w_0 \cdot \mathbf{1} - M \cdot \mathbf{z}_g - U_g \cdot \mathbf{1} \quad \leq \ \mathbf{0} \\
& \mathbf{X}_g \mathbf{w} + w_0 \cdot \mathbf{1} + M \cdot \mathbf{z}_g - L_g \cdot \mathbf{1} \quad \geq \ \mathbf{0} \\
& U_g - L_g \quad \geq \ 0 \\
& L_h - U_g + M y_{hg} \quad \geq \ \varepsilon \\
& y_{gh} + y_{hg} \quad = \ 1 \\
& \mathbf{w}, w_0, \mathbf{L}, \mathbf{U} \text{ free}; \\
& \mathbf{z}_g \in \{0,1\}^{N_g}; \ \mathbf{y} \in \{0,1\}^{G(G-1)} .
\end{aligned}$$

The first three constraints are repeated for $g = 1, \ldots, G$ while the next two are repeated for all pairs g,

$h = 1, \ldots, G$ such that $g \neq h$. Observations are classified into group g if their scores fall in the interval $[L_g, U_g]$. Variable $y_{gh} = 1$ if the scoring interval for group g precedes that for group h. Parameter $\varepsilon > 0$ dictates a minimum separation between intervals.

Using a single scalar-valued discriminant function with $G > 2$ groups is restrictive; it assumes that the groups project onto some line in an orderly manner. In [9], Gehrlein also suggested a model using a vector-valued discriminant function $\boldsymbol{f}()$ of dimension G. Observation $\mathbf{x}$ would be classified into the group corresponding to the largest component of $\mathbf{f}(\mathbf{x})$. The model increases the number of coefficient variables and the number of constraints but not the number of 0-1 variables, the primary determinant of execution time. The model is:

$$\begin{aligned} \min \quad & \sum_{g=1}^{G} \frac{\pi_g C_g}{N_g} \sum_{n=1}^{N_g} z_{gn} \\ \text{s.t.} \quad & \mathbf{X}_g \mathbf{w}_g + w_{g0} \cdot \mathbf{1} - \mathbf{X}_g \mathbf{w}_h - w_{h0} \cdot \mathbf{1} + M \cdot \mathbf{z}_g \\ & \geq \varepsilon \cdot \mathbf{1} \\ & \mathbf{w}_g, w_{g0} \text{ free}; \ \mathbf{z}_g \in \{0,1\}^{N_g}. \end{aligned}$$

Here $\mathbf{w}_g'\mathbf{x} + w_{g0}$ is the gth component of $\mathbf{f}(\mathbf{x})$ and $\varepsilon > 0$ is the minimum acceptable difference between the correct component of the scoring function and the largest incorrect component. The sole constraint is repeated once for each pair $g, h = 1, \ldots, G$ such that $g \neq h$.

Methods

Advances in computer hardware, optimization software and algorithms for the mixed integer classification problem have allowed progressively larger training samples to be employed: where Koehler and Erenguc [11] were restricted to combined training samples of 100 in 1990 (on a mainframe), Rubin [13] was able to handle over 600 observations in 1997 (on a personal computer). Nonetheless, a variety of heuristics have been developed to find near optimal solutions to the problem. Several revolve around this property of the problem: if the training samples can be classified with perfect accuracy by a linear function, then problem (1) can be solved as a linear program, with the z_{gn} deleted, to obtain a discriminant function. Deletion of the z_{gn} reduces the objective function to a constant 0. Although this is perfectly acceptable, heuristics may substitute an objective function from one of the linear programming classification models, to encourage the chosen discriminant function to separate scores of the two groups as much as possible. This often also necessitates inclusion of a normalization constraint, to keep the resulting linear program from being unbounded. Alternatively, (1) may be solved heuristically to determine which training observations to misclassify, and then a linear programming model using the remaining observations may be employed to select the final discriminant function.

The BPMM heuristic of [11] solves the linear program dual to a relaxation of the mixed integer problem, notes which observations would be misclassified by the resulting discriminant function, and then solves the dual of each linear relaxation obtainable by deleting one of those observations. Solving the dual problem tends to be more efficient than solving the primal, since there will typically be more observations than attributes ($N_1 + N_2 \gg p$). The heuristics presented in [14] also operate on the dual of the linear relaxation of the mixed integer problem, restricting basis entry to force certain dual variables to take value zero (equivalent to relaxing the corresponding primal constraints, thus allowing the associated observations to be misclassified).

As noted earlier, comparatively few computational studies involve mixed integer models for multiple groups. Pavur proposed a sequential mixed integer method to handle multiple groups [12], constructing a vector-valued scoring function from a sequence of scalar functions. An initial mixed integer model similar to Gehrlein's is solved to obtain the first scalar function. Thereafter, a sequence of similar mixed integer models is solved, with each model bearing additional constraints compelling the scores produced by the next scoring function to have sample covariance zero with the scores of each of the preceding functions. The covariance constraints impose a sort of probabilistic "orthogonality" on the dimensions of the composite (vector-valued) scoring function.

See also

- ▶ Deterministic and Probabilistic Optimization Models for Data Classification
- ▶ Integer Programming

► Linear Programming Models for Classification
► Optimization in Boolean Classification Problems
► Statistical Classification: Optimization Approaches

References

1. Amaldi E, Kann V (1995) The complexity and approximability of finding maximum feasible subsystems of linear relations. Theor Comput Sci 147:181–210
2. Asparouhov O, Rubin PA (2004) Oscillation heuristics for the two-group classification problem. J Classif 21:255–277
3. Asparoukhov OK, Stam A (1997) Mathematical programming formulations for two-group classification with binary variables. Ann Oper Res 74:89–112
4. Bajgier SM, Hill AV (1982) An experimental comparison of statistical and linear programming approaches to the discriminant problem. Decis Sci 13:604–618
5. Bennett KP, Bredensteiner EJ (1997) A parameteric optimization method for machine learning. INFORMS J Comput 9(3):311–318
6. Chinneck JW (2001) Fast heuristics for the maximum feasible subsystem problem. INFORMS J Comput 13(3):210–223
7. Codato G, Fischetti M (2006) Combinatorial Benders' cuts for mixed-integer linear programming. Oper Res 54(4): 756–766
8. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
9. Gehrlein WV (1986) General mathematical programming formulations for the statistical classification problem. Oper Res Lett 5:299–304
10. Hand DJ (1997) Construction and assessment of classification rules. Wiley, Chichester
11. Koehler GJ, Erenguc SS (1990) Minimizing misclassifications in linear discriminant analysis. Decis Sci 21:63–85
12. Pavur R (1997) Dimensionality representation of linear discriminant function space for the multiple-group problem: An MIP approach. Ann Oper Res 74:37–50
13. Rubin PA (1990) Heuristic solution procedures for a mixed-integer programming discriminant model. Manag Decis Econ 11:255–266
14. Rubin PA (1997) Solving mixed integer classification problems by decomposition. Ann Oper Res 74:51–64
15. Soltysik R, Yarnold P (1994) The Warmack–Gonzalez algorithm for linear two-category multivariable optimal discriminant analysis. Comput Oper Res 21:735–745
16. Vapnik V (1999) An overview of statistical learning theory. IEEE Trans Neural Netw 10:988–999
17. Vapnik V, Chervonenkis A (1971) On the uniform convergence of relative frequencies of events to their probabilities. Theor Probab Appl 16:264–280
18. Warmack R, Gonzalez R (1973) An algorithm for the optimal solution of linear inequalities and its application to pattern recognition. IEEE Trans Comput C22:1065–1075

Mixed Integer Linear Programming: Heat Exchanger Network Synthesis

KEMAL SAHIN, KORHAN GÜRSOY, AMY CIRIC
Department Chemical Engineering,
University Cincinnati, Cincinnati, USA

MSC2000: 90C90

Article Outline

Keywords
Using MILP Models
to Find the Minimum Number of Units
Conclusions
See also
References

Keywords

MILP; HEN synthesis; Transshipment model

Heat exchanger networks use the waste heat released by hot process streams to heat the cold process streams of a chemical manufacturing plant, reducing utility costs by as much as 80%. *Heat exchanger network synthesis* has been an active area of process research ever since the energy crisis of the 1970s, and over 400 research papers have been published in the area. See [1,2,4,5,6], for recent reviews.

In 1979, T. Umeda et al. [8] discovered a thermodynamic pinch point that limits the energy savings of a heat exchanger network, establishes minimum utility levels, and partitions the heat exchanger network into two independent subnetworks. This discovery revolutionized heat exchanger network synthesis: with it, designers could compute utility levels a priori, then seek the heat exchanger network structure that uses the minimum utility consumption while also minimizing the total investment cost. This remaining problem requires matching the hot utilities and process streams that release heat with the cold process streams and utilities that require heat, choosing the network structure of each stream, and designing the individual heat exchanger networks. In general, this is a *mixed integer nonlinear programming problem* (MINLP), but can be decomposed into two smaller problems by first selecting the matches between hot and cold process streams

and utilities by minimizing the total number of units, then optimizing the network structure. The first problem is a mixed integer linear programming problem that will be discussed in detail here.

Using MILP Models to Find the Minimum Number of Units

Stated formally, the *minimum-units problem* is:
Given

1) A set of hot process streams and utilities $i \in H$, and for each hot stream i:
   a) the inlet and outlet temperatures $\mathbf{T}_i^I$ and $\mathbf{T}_i^O$;
   b) either the heat capacity flow rate $\mathbf{FC_{Pi}}$ or the heat duty $\mathbf{Q_i}$.
2) A set of cold process streams $j \in C$, and for each cold stream j:
   a) the inlet and outlet temperatures $\mathbf{T}_j^I$ and $\mathbf{T}_j^O$;
   b) either the heat capacity flow rate $\mathbf{FC}_{pj}$ or the heat duty $\mathbf{Q}_j$,
3) The minimum temperature difference between hot and cold streams exchanging heat, $\Delta\, T_{\min}$.

Identify a set of stream matches (ij) and their heat duties Q_{ij} that

a) meets the heating and cooling needs of each stream; and
b) minimizes the total number of matches.

S.A. Papoulias and I.E. Grossmann [7] formulated this as a mixed integer programming problem using a *transshipment model*, by making an analogy between heat exchanger networks and transportation networks. In the transshipment analogy, hot process streams, the sources of heat, are similar to manufacturing plants, the sources of goods, while cold process streams, the heat sinks, are akin to stores and shopping malls, the sinks of manufactured goods.

The analogy is not perfect, as heat only flows from a high temperature to a lower one, in obedience to the second law of thermodynamics. Partitioning the temperature range of the heat exchanger network into intervals can capture this heat flow pattern. Each interval sends excess, or residual, heat to the interval below it, just as excess manufactured goods are sent to a discount warehouse.

The hot side of this *temperature cascade* is created by ordering $\mathbf{T}_i^I$ and $\mathbf{T}_j^I + \Delta\, T_{\min}$ from the highest to the lowest value, creating $t = 1, \ldots, TI$ temperature intervals. Temperatures on the cold side of the cascade equal the temperature on the hot side minus $\Delta T_{\min}$. Hot stream i releases $\mathbf{Q}_{it}^H$ units of heat to temperature interval t. $\mathbf{Q}_{it}^H$ is equal to

$$Q_{it}^H = \begin{cases} FCP_i(T_{t-1} - T_t) \\ \quad \text{if } T_i^I \geq T_{t-1} \text{ and } T_i^O \leq T_t, \\ FCP_i(T_{t-1} - T_i^O) \\ \quad \text{if } T_i^I \geq T_{t-1} \text{ and } T_i^O \geq T_t, \\ Q \\ \quad \text{if } T_i^I = T_i^O \text{ and } T_{t-1} = T_i^I. \end{cases}$$

Cold stream j absorbs $\mathbf{Q}_{jt}^C$ units of heat from temperature interval t. $\mathbf{Q}_{jt}^C$ equals

$$Q_{jt}^C = \begin{cases} FCP_j(T_{t-1} - T_t) \\ \quad \text{if } T_j^I \leq T_t - \Delta T_{\min} \text{ and} \\ \quad\quad T_j^O \geq T_{t-1} - \Delta T_{\min}, \\ FCP_j(T_j^0 - T_{t-1}) \\ \quad \text{if } T_j^I \leq T_t - \Delta T_{\min} \text{ and} \\ \quad\quad T_j^O \leq T_{t-1} - \Delta T_{\min}, \\ Q_j \\ \quad \text{if } T_j^I = T_j^O \text{ and } T_j^I = T_{t-1} - \Delta T_{\min}. \end{cases}$$

Any excess heat sent to interval t from hot stream i cascades down to interval t+1 through the residual flow R_{it}. Process utilities may be treated as process streams, or may be placed at the top or bottom of the cascade.

This transshipment model of heat flow leads to the following mixed integer linear programming problem:

$$\min \sum_{i,j} y_{ij}$$

Mixed Integer Linear Programming: Heat Exchanger Network Synthesis, Table 1
Stream data. $Q_{CW} = 8395.2\ kW$, $\Delta T_{\min} = 10°C$

Stream	$T^{\text{in}}(°)$	$T^{\text{out}}(°)$	$FC_p(kW/K)$
H1	159	77	228.5
H2	159	88	20.4
H3	159	90	53.8
C1	26	127	93.3
C2	118	149	196.1

subject to

$$R_{i,t} - R_{i,t-1} + \sum_j q_{ijt} = Q_{it}^H, \quad i = 1, \ldots, H, \quad t = 1, \ldots, TI, \tag{1}$$

$$\sum_i q_{ijt} = Q_{jt}^C, \quad j = 1, \ldots, C, \; t = 1, \ldots, TI, \tag{2}$$

$$Q_{ij} = \sum_{t \in TI} q_{ijt}, \quad i = 1, \ldots, H, \; j = 1, \ldots, C, \tag{3}$$

$$Q_{ij} \le U_{ij} y_{ij}, \quad i = 1, \ldots, H, \; j = 1, \ldots, C, \tag{4}$$

$$\begin{cases} q_{ijt} \ge 0, \\ R_t \ge 0, \end{cases} \quad i = 1, \ldots, H, \; j = 1, \ldots, C, \; t = 1, \ldots, TI, \tag{5}$$

$$R_0 = R_T = 0, \tag{6}$$

$$y_{ij} = \{0, 1\}, \quad i = 1, \ldots, H, \; j = 1, \ldots, C. \tag{7}$$

In this formulation, y_{ij} is a binary variable which is one if a match between hot process stream i and cold process stream j occurs, and zero otherwise; q_{ijt} is the amount of heat exchanged between hot stream i and cold stream j in temperature interval t, R_{it} is the residual heat flow associated with hot stream i that cascades down from temperature interval t to temperature interval t+1, and Q_{ij} is the heat duty of match (i, j). The overall objective function minimizes the total number of units. Constraint (1) is the energy balance for hot stream i around temperature interval t and constraint (2) is the energy balance for cold stream j around temperature interval t. Constraint (3) finds the overall heat duty of match (ij). Constraint (4) sets this heat duty to zero when match (ij) does not exist. The nonnegativity constraints prevents heat flow from a low temperature to a higher one. Note that the residual heat flows into the first temperature interval and out of the last temperature interval are zero when there are no utilities above or below the cascade. The objective function and the constraints are linear, and the formulation involves both continuous and integer variables, making this a mixed integer linear programming problem.

Lower bounds on the solution of this problem are given by linear programming problems where some integer variables are fixed to either zero or one and the remainder are treated as continuous variables. The accuracy of these bounds depends upon the parameters U_{ij} is the fourth constraint. When these parameters are very large, the lower bounds will be quite far from the solution of the MILP.

The smallest acceptable value of U_{ij} is the minimum of the cooling requirements of stream i and the heating requirements of stream j:

$$U_{ij} = \min \left\{ \sum_{t \in TI} Q_{it}^H, \sum_{t \in TI} QC_{jt} \right\}.$$

Example 1 This example is from [3] and features three hot streams, two cold streams, and a cold utility. Table 1 gives the inlet and outlet stream temperatures and the flowrate heat capacities of each process stream and the cooling water duty.

Temperatures on the hot side of the cascade are 159°C, 128°C, and 36°C, while temperatures on the cold side are 149°C, 118°C and 26°C. There are two temperature intervals. Table 2 gives the heat released from hot streams to the temperature intervals, while Table 3 gives the heat absorbed by the cold streams from the temperature intervals.

Mixed Integer Linear Programming: Heat Exchanger Network Synthesis, Table 2
Q_{it}^H, heat released from hot stream *i* to temperature interval *t*

Stream	Temperature	Interval
	TI-1	TI-2
H1	7083.5	11635.5
H2	632.4	816.0
H3	1667.8	2044.4

Mixed Integer Linear Programming: Heat Exchanger Network Synthesis, Table 3
Q_{it}^C, heat absorbed by cold stream *i* from temperature interval *t*

Stream	Temperature	Interval
	TI-1	TI-2
C1	839.7	8583.6
C2	6079.1	
CW		8395.2

Mixed Integer Linear Programming: Heat Exchanger Network Synthesis, Table 4
Four solutions which satisfy minimum number of matches

Solution 1 Match Duty (Q_{ij})		Solution 2 Match Duty (Q_{ij})		Solution 3 Match Duty (Q_{ij})		Solution 4 Match Duty (Q_{ij})	
H1−C1	9423.3	H1−C1	7974.9	H1−C1	5711.1	H1−C1	4262.7
H1−C2	6079.1	H1−C2	6079.1	H1−C2	6079.1	H1−C2	6079.1
H1−CW	3234.6	H1−CW	4683.0	H1−CW	6946.8	H1−CW	8395.2
H2−CW	1448.4	H2−C1	1448.4	H2−CW	1448.4	H2−C1	1448.4
H3−W	3712.2	H3−CW	3712.2	H3−C1	3712.2	H3−C1	3712.2

In this example, the minimum number of units is 5, and there are four solutions to this MILP that meet this minimum (cf. Table 4).

Conclusions

Mixed integer linear programs are used in heat exchanger network synthesis to identify the minimum number of units, and a set of matches and their heat loads meeting the minimum. These MILPs are based upon a transshipment model of heat flow.

See also

- ▶ Chemical Process Planning
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ Global Optimization of Heat Exchanger Networks
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming

References

1. Biegler LT, Grossmann IE, Westerberg AW (1997) Systematic methods of chemical process design. Prentice-Hall, Englewood Cliffs
2. Floudas CA (1995) Nonlinear and mixed-integer optimization. Oxford Univ. Press, Oxford
3. Gundersen T, Grossmann IE (1990) Improved optimization strategies for automated heat exchanger synthesis through physical insights. Comput Chem Eng 14(9):925
4. Gundersen T, Naess L (1988) The synthesis of cost optimal heat exchanger networks: An industrial review of the state-of-the-art. Comput Chem Eng 12(6):503
5. Jezowski J (1994) Heat exchanger network grassroot and retrofit design: The review of the state-of-the-art: Part I. Hungarian J Industr Chem 22:279–294
6. Jezowski J (1994) Heat exchanger network grassroot and retrofit design: The review of the state-of-the-art: Part II. Hungarian J Industr Chem 22:295–308
7. Papoulias SA, Grossmann IE (1983) A structural optimization approach in process synthesis - II: Heat recovery networks. Comput Chem Eng 7:707
8. Umeda T, Harada T, Shiroko K (1979) A thermodynamic approach to the structure in chemical processes. Comput Chem Eng 3:373

Mixed Integer Linear Programming: Mass and Heat Exchanger Networks

MEN, MHEN

KATERINA P. PAPALEXANDRI
bp Upstream Technol., Middlesex, UK

MSC2000: 93A30, 93B50

Article Outline

Keywords
See also
References

Keywords

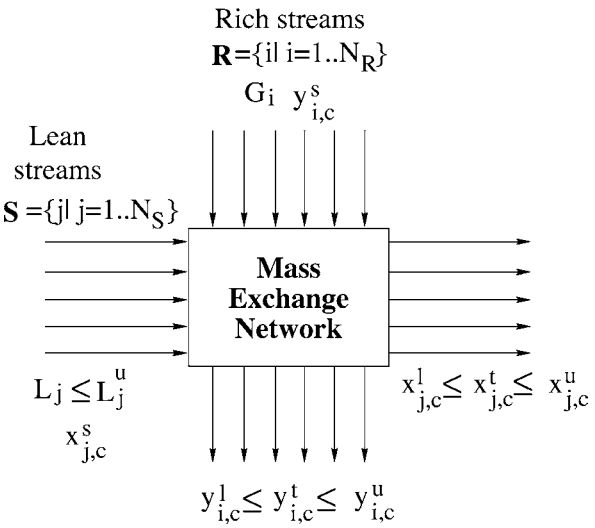
MILP; Mass and heat exchange; Separation

Separation networks involving mass transfer operations that do not require energy (e. g. absorption, liquid-liquid extraction, ion-exchange etc.) are characterized as mass exchange networks (MEN). These appear in the chemical industries mostly in waste treatment, but also, in feed preparation, product separation, recovery of valuable materials, etc. A *mass exchanger*, in this context, is any counter-current, direct-contact mass transfer unit, where one or more components are transferred at constant temperature and pressure from one process stream, which is characterized as *rich stream*, to another process or utility stream, characterized as *lean stream*. Mass integration aims to the purification of the rich streams and the recovery of valuable or hazardous materials at the minimum total cost (investment and operating cost of auxiliary streams). In the specific case, when the mass transfer operations take place at the same temperature, or heating/cooling requirements are negligible, the integration problem is limited to the synthesis of a mass exchanger network (MEN) only. When mass exchange operations at different temperature levels are encountered, mass and heat exchanger networks (MHEN) may be considered simultaneously.

MEN synthesis involves a set of rich streams, in terms of one or more components, $\mathbf{R} = \{i: i = 1, \ldots, N_R\}$, with known flowrates, G_i, inlet and outlet compositions for the components of interest, y^s_{ic}, y^t_{ic} (exact values or bounds) respectively, and a set of process or auxiliary lean streams (*mass separating agents*, MSAs), $\mathbf{S} = \{j: j = 1, \ldots, N_S\}$ with known cost, inlet and outlet compositions for the same components, x^s_{jc}, x^t_{jc} (exact values or bounds), as shown in Fig. 1.

The *synthesis problem* refers to the selection of the appropriate lean streams and their flowrates, L_j, the mass exchange operations (*mass exchange matches*), the mass transfer load for each separator and its required size, and the configuration of the overall network.

Mass transfer in each mass exchanger is governed by the first and second thermodynamic laws, as is heat transfer in heat exchangers. Mass transfer of a component c from a rich to a lean stream is feasible if the composition of c in the rich phase is greater than the equilibrium composition with respect to the lean phase:



Mixed Integer Linear Programming: Mass and Heat Exchanger Networks, Figure 1

$$y_c \geq f(x_c) + \epsilon, \tag{1}$$

where $f(x_c)$ is the equilibrium relation and ϵ is a *minimum composition difference* that ensures feasible mass transfer in a separator of finite size, in analogy to $\Delta T_{\min}$ in heat exchangers. This analogy led to the development of synthesis methods for mass exchanger networks employing mixed integer optimization techniques, similar to heat exchanger networks (cf. Mixed Integer Linear Programming: Mass and Heat Exchanger Networks; ► MINLP: Mass and Heat Exchanger Networks), that are categorized into the *sequential synthesis* and the *simultaneous synthesis* methods.

The *sequential MEN synthesis method*, introduced in [3] and [4] involves the following steps:

1) Minimum cost of mass separating agents (minimum utility problem), to determine the optimal flows of the mass separating agents.
2) Minimum number of mass exchanger units, for fixed MSA flows, to determine the mass exchange matches.
3) Network configuration and separator sizes for fixed mass exchange operations.

The first two synthesis steps involve the solution of linear and mixed integer linear problems.

A useful tool of the sequential MEN synthesis method is the *composition interval diagram*, CID,

where thermodynamic feasibility of mass transfer is explored mapping the rich and the lean streams on equivalent composition scales, that are derived from the mass transfer feasibility requirements in (1). In general, the composition equivalent scales and the minimum composition difference, ϵ, are defined for each component of interest and each pair of rich and lean streams. In the simple case of a single component, where mass transfer is independent of the presence of other components in the rich streams, the CID is constructed as illustrated in Fig. 2.

Feasible rich-to-lean mass transfer is guaranteed within a composition interval when the equilibrium relation $f(x_c)$ is convex within the interval. When $f(x_c)$ is convex in the whole composition range, only inlet compositions are required to construct the CID [8].

The minimum cost of mass separating agents is found employing a *transshipment model*, where the components of interest are the transferred commodities, the rich and the lean streams are considered as sources and sinks respectively, and the composition intervals define the intermediate nodes [4]. The model involves energy balances around the temperature intervals (intermediate nodes):

$$
\text{(TP1)} \quad \begin{cases} \min & \displaystyle\sum_j c_j L_j \\ \text{s.t.} & \delta_{k-1} + \displaystyle\sum_{i \in R_k} WR_k^i \\ & \quad = \delta_k + \displaystyle\sum_{j \in S_k} WS_k^j \\ & 0 \le L_j \le L_j^{\text{up}}, \ j \in S, \\ & \delta_0 = \delta_{N_{\text{int}}} = 0; \\ & \delta_k \ge 0, \ k = 1, \ldots, N_{\text{int}} - 1, \end{cases}
$$

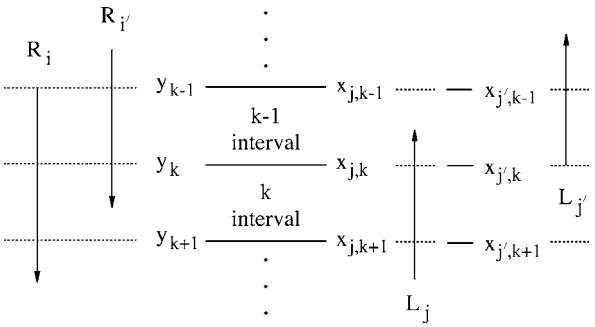
where

- R_k is the set of rich streams, present in interval k;
- S_k is the set of lean streams, present in interval k;
- N_{int} is the number of composition intervals;
- WR_k^i is the mass exchange load of rich stream i in interval k,

$$WR_k^i = G_i(y_k - \max(y_{k+1}, y_i^t));$$

- WS_k^j is the mass exchange load of lean stream j in interval k,

$$WS_k^j = L_j(\min(x_j^t, x_{jk}) - x_{jk+1});$$



Mixed Integer Linear Programming: Mass and Heat Exchanger Networks, Figure 2
Composition interval diagram

- δ_k is the residual mass exchange load in interval k.

Problem (TP1) results in the optimal flows of the mass separating agents and the identification of the *pinch points*, i. e. the thermodynamic bottlenecks in mass transfer. The pinch points are defined by zero residual flows and divide the mass exchange network into subnetworks. Mass transfer between different subnetworks (i. e. across the pinch) increases the cost of mass separating agents.

An assumption in (TP1) is that molar flows of the rich and the lean streams are constant. If significant flowrate variations take place, compositions and mass exchange loads are calculated based on nontransferable components.

The following cases are distinguished:

- Fixed inlet and outlet compositions.
  Then, (TP1) is an LP problem.
  When multiple components are considered, the CID is defined for all the components of interest and (TP1) corresponds to the multicommodity transshipment model. The pinch points are then determined by the component that requires the greater MSA flows.
- Variable outlet compositions.
  Then, the mass exchange loads of the rich and lean streams in their final intervals (defined by the upper and lower bounds on their outlet compositions) are variables. Problem (TP1) can still be solved as an LP [9], considering the variable mass exchange loads explicitly in the model.

Variable inlet compositions usually require flexible mass exchange networks to accommodate the variations and define a different problem. For a single com-

ponent it has been shown that the minimum MSA cost corresponds to the lower bounds of the inlet compositions [8].

For nonconvex equilibrium relations, (TP1) cannot guarantee feasible mass transfer throughout the composition range, while the predicted MSA cost is a lower bound to the actual minimum one. B.K. Srinivas and M.M. El-Halwagi suggested in [14] an iterative procedure to calculate the minimum required MSA cost, that involves two major steps:

i) a 'feasibility problem', where 'critical' composition levels are identified and included in the CID (nonconvex NLP step, that requires global optimization methods), and
ii) (TP1) with updated intervals, which calculates increasing lower bounds to the minimum MSA cost.

Instead of target outlet compositions for the rich streams, it may be of interest to remove a certain total mass load of pollutants. Then, (TP1) is solved with variable rich outlets and a fixed total mass exchange load [10]:

$$M_c = \sum_i G_i (y_i^s - y_i^t)$$

The minimum-utility-cost problem has been alternatively formulated as an LP or MINLP problem, based on total mass balances and the following property:

$$\left\{ \begin{array}{l} \text{Mass lost by all the rich} \\ \text{streams below each} \\ \text{pinch point candidate} \end{array} \right\} - \left\{ \begin{array}{l} \text{Mass gained by all the lean} \\ \text{streams below each} \\ \text{pinch point candidate} \end{array} \right\} \leq 0 \quad (2)$$

and employing binary variables to denote the relative position of variable outlet compositions with respect to each pinch point candidate in the CID [5,6,8,9].

The minimum number of mass exchange operations (units) for fixed MSA cost is determined in each subnetwork in a second step, in an attempt to minimize the fixed cost of the separators. The minimum number of mass exchangers is found employing the *expanded transshipment model*, where the existence of a mass exchange match-separator in a subnetwork is denoted by a binary variable:

$$E_{ijm} = \begin{cases} 1, & \text{when streams } i, j \\ & \text{exchange mass} \\ & \text{in subnetwork } m \\ 0, & \text{otherwise.} \end{cases}$$

For a single component, the minimum number of mass exchanger units is given by the following MILP problem [4]:

$$\text{(TP2)} \quad \begin{cases} \min & \displaystyle\sum_m \sum_{i \in R_m} \sum_{j \in S_m} E_{ijm} \\ \text{s.t.} & \delta_{ik} - \delta_{ik-1} + \displaystyle\sum_{j \in S_{mk}} M_{ijk} = WR_k^i, \\ & k \in I_m,\ i \in R_{mk},\ m \in M, \\ & \displaystyle\sum_{i \in R_{mk}} M_{ijk} = WS_k^j, \\ & k \in I_m,\ j \in S_{mk},\ m \in M \\ & \displaystyle\sum_{k \in I_m} M_{ijk} - E_{ijm} U_{ijm} \leq 0 \\ & \delta_{ik} \geq 0,\ k \in I_m,\ i \in R_m, \\ & M_{ijk} \geq 0,\ k \in I_m, \\ & i \in R_{km},\ j \in S_{km} \\ & E_{ijk} = 0, 1,\ k \in I_m, \\ & i \in R_{km},\ j \in S_{km}, \end{cases}$$

where

- R_m is the set of rich streams, present in subnetwork m,
- S_m is the set of lean streams, present in subnetwork m,
- I_m is the set of intervals in subnetwork m,
- R_{km} is the set of rich streams, present in interval k of subnetwork m, or above,
- S_{km} is the set of lean streams, present in interval k of subnetwork m,
- WR_k^i is the mass exchange load of rich stream i in interval k,
- δ_{ik} is the residual mass exchange load of rich stream i in interval k,
- WS_k^j is the mass exchange load of lean stream j in interval k, as determined by (TP1),
- M_{ijk} is the mass exchange load between i and j in interval k,

- U_{ijm} is an upper bound to the possible mass exchange load between i and j in subnetwork m,

$$U_{ijm} = \min\left(\sum_{k\in I_m} WR_k^i, \sum_{k\in I_m} WS_k^j\right).$$

Srinivas and El-Halwagi have shown [14] that, when the equilibrium relations around a pinch point are not convex, a mass exchanger can straddle the pinch and still be thermodynamically feasible. To account for such cases, exchangers across the pinch points can be considered introducing extra binary variables:

$$I_{ijp} \le M_{ijp},$$
$$I_{ijp+1} \le M_{ijp+1},$$
$$I_{ijp} + I_{ijp+1} \ge 2B_{ijp},$$
$$I_{ijp}, I_{ijp+1} \in \{0,1\},$$
$$B_{ijp} \in \{0,1\},$$

where

- I_{ijp} denotes that streams i and j exchange mass at the interval directly above pinch point p,
- I_{ijp+1} denotes that streams i and j exchange mass at the interval directly below pinch point p,
- B_{ijp} denotes the existence of an exchanger between streams i and j, across the pinch p.

Then, the number of required units to minimize is given by:

$$\sum_m \left(\sum_{i\in R_m}\sum_{j\in S_m} E_{ijm} - \sum_p B_{ijp}\right).$$

Note, that I_{ijp}-variables can be relaxed to continuous, due to total unimodularity of the model with respect to these variables:

$$0 \le I_{ijp},\ I_{ijp+1} \le 1$$

Problem (TP2) may not have a unique solution. Alternative combinations of mass exchange matches, featuring the minimum MSA cost, may be generated by solving (TP2) iteratively and including integer cuts. These do not necessarily correspond to networks of the same overall cost.

The expanded transshipment model can also be employed to determine the minimum MSA cost, considering variable mass loads for the lean streams. Then, forbidden or restricted mass exchange operations can be explicitly accounted for.

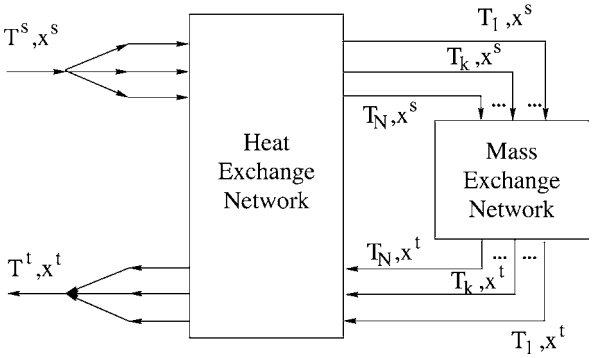
Although (TP2) does not determine the network structure, stream splitting and exchanger connectivity may be guided by the resulting mass exchange load distribution in each composition interval [4]. The actual network configuration is found in a next step, employing heuristic methods [3,5] or superstructure methods (NLP models).

Special cases of mass exchange networks have been studied:

- MEN and *regeneration networks* [5,11].
  The regeneration of mass separating agents by auxiliary streams can be considered simultaneously with the main MEN, in another mass exchanger network, where the MSAs behave as the rich streams. In this case, the CID is extended to include the equivalent composition scales of the regenerating agents. The inlet and outlet compositions of the lean streams in the main MEN are in general variables.
- Reactive mass exchange networks [6,11,14]
  Rich-to-lean mass transfer may involve interphase mass transfer and chemical reaction in the lean phase, at constant temperature. Mass exchange operations of this kind are considered deriving the equilibrium relations based on chemical equilibrium.

The main advantage of the sequential synthesis method for mass exchange networks is that simple optimization models are solved. However, unless the MSA cost is dominant, as synthesis decisions are fixed from one step to the next, important trade-offs between operating and capital cost are not exploited and overall cost optimality cannot be guaranteed. Furthermore, the minimum composition difference, ϵ that defines the mass recovery levels in (TP1) and (TP2), is in general, an optimization variable for each mass exchanger separately. In the sequential synthesis method this is fixed arbitrarily to a possibly conservative value for the construction of the CID. El-Halwagi and V. Manousiouthakis [4] suggested a two-level optimization procedure to select a unique ϵ for all mass exchange operations, based on the impact of ϵ on the final MEN cost, still, not exploiting the overall cost trade-offs.

When isothermal mass exchange operations take place at different temperature levels, the operating and overall mass integration costs are affected by the heating and cooling requirements of the system. Energy integration between the rich and lean streams can be



Mixed Integer Linear Programming: Mass and Heat Exchanger Networks, Figure 3

considered within a mass and heat exchanger network synthesis problem (MHEN) to reduce the total cost. The overall problem is addressed combining MEN and HEN synthesis tools. The optimal temperature of mass exchange is defined for each pair of rich and lean streams by the equilibrium relations that limit mass transfer

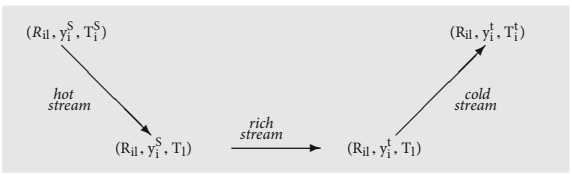
$$y_i \geq K_{ij}(T)x_j,$$

where $K_{ij}(T)$ is a known function of temperature.

In the sequential synthesis framework, the overall minimum operating cost for the network (cost of mass separating agents and heating/cooling utilities) may be calculated from a combined mass and heat transshipment model. Each stream is considered to consist of substreams, of the same inlet and outlet composition and temperature, each of which participates to isothermal mass exchange operations at a different temperatures. Srinivas and El-Halwagi proved [13], that, for monotonic dependence of the equilibrium constant on temperature, the overall utility cost of the combined MHEN is independent of such a stream decomposition, see Fig. 3.

Although the mass exchange temperatures ($T_1, \ldots, T_N$) are variables, their relative position with respect to inlet and outlet stream temperatures (greater or less) can be prepostulated. Thus, the rich and lean substreams define hot (or cold) streams before their mass exchange operations and cold (or hot) streams afterwards, cf. Fig. 4.

A CID is constructed, similarly to the simple MEN case, involving the several substreams with variable flows, and thus, variable mass loads in each composition interval. Mass exchange is permitted between substreams of the same temperature. A *temperature interval diagram*, TID, is also constructed, involving the hot and cold substreams and the available heating and cooling utilities, with variable heat loads per interval, due to the variable substream flows. In order to avoid discrete decisions (i. e. presence or not of streams in temperature intervals with variable limits), the temperature range for each mass transfer operation is discretized and a substream is associated with each candidate temperature [13].



Mixed Integer Linear Programming: Mass and Heat Exchanger Networks, Figure 4
Rich substream with $T_i^s \leq T_l \leq T_i^t$

The minimum utility cost is found from the solution of the combined LP transshipment model, which, for a single component is as follows:

$$\text{(TP3)} \quad \left\{ \min \left(\sum_{j \in S} c_j L_j + \sum_{n \in TI} \sum_{h \in HU_n} c_h QHU_{hn} + \sum_{n \in TI} \sum_{c \in CU_n} c_c QCU_{cn} \right) \right.$$

such that

$$\delta_{l_i k} - \delta_{l_i k-1} + \sum_{j \in S} \sum_{l'_j \in SS_{jk}} M_{l_i l'_j k} = WR_k^{l_i},$$

$$k \in CI,\ i \in R,\ l_i \in RS_{ik},$$

$$\sum_{i \in R} \sum_{l'_i \in RS_{ik}} M_{l'_i l_j k} = WS_k^{l_j},$$

$$k \in CI,\ j \in S, l_j \in SS_{jk},$$

$$\theta_{l_s n} - \theta_{l_s n-1} + \sum_{s' \in R \cup S} \sum_{l'_{s'} \in CS_{s'n}} Q_{l_s l'_{s'} n}$$

$$+ \sum_{c \in CU_n} QC_{l_s cn} = QS_{l_s n},$$
$$n \in TI,\ s \in R \cup S,\ l_s \in HS_{sn}, \qquad \theta^h_{hn} - \theta^h_{hn-1}$$
$$+ \sum_{s \in R \cup S} \sum_{l_s \in CS_{sn}} QH_{hl_s n} = QHU_{hn},$$
$$n \in TI,\ h \in HU_n,$$

$$\sum_{s' \in R \cup S} \sum_{l'_{s'} \in HS_{s'n}} Q_{l'_{s'} l_s n}$$
$$+ \sum_{h \in HU_n} QH_{hl_s n} = QS_{l_s n},$$
$$n \in TI,\ s \in R \cup S,\ l_s \in CS_{sn},$$

$$\sum_{s \in R \cup S} \sum_{l_s \in HS_{sn}} QC_{l_s cn} = QCU_{cn},$$
$$n \in TI,\ c \in CU_n,$$

$$\delta_{l_i k} \geq 0,\ k \in CI,\ i \in R,\ l_i \in RS_{ik},$$

$$\theta_{l_s n} \geq 0,\ n \in TI,\ s \in R \cup S,\ l_s \in HS_{sn},$$

$$\theta^h_{hn} \geq 0, n \in TI,\ h \in HU_n,$$

$$M_{l_i l'_j k} \geq 0,$$
$$k \in CI,\ i \in R,\ j \in S,$$
$$l_i \in RS_{ik},\ l'_j \in SS_{jk},$$

$$M_{l_i l'_j k} = 0,$$
$$k \in CI,\ i \in R,\ j \in S,$$
$$l_i \in RS_{ik},\ l'_j \in SS_{jk},$$
$$(l_i l'_j) \in FM,$$

$$\delta_{l_i 0} = \delta_{l_i N_{CI}} = 0,$$
$$i \in R,\ l_i \in RS_i,$$

$$\theta_{l_s 0} = \theta_{l_s N_{TI}} = 0,$$
$$s \in R \cup S,\ l_s \in HS_s,$$

$$\theta^h_{h0} = \theta^h_{hN_{TI}} = 0,$$
$$h \in HU,$$

where

- CI is the set of composition intervals k,
- TI is the set of temperature intervals n,
- RS_{ik} is the set of substreams of rich stream i, of variable flow, G_{l_i}, such that

$$\sum_l G_{l_i} = G_i,$$

present in interval k, or above,

- SS_{jk} is the set of substreams of lean stream j, of variable flow, L_{l_j}, such that

$$\sum_l L_{l_j} = L_j,$$

present in interval k,

- HS_{sn} is the set of hot substreams of stream s, present in interval n, or above,
- CS_{sn} is the set of cold substreams of stream s, present in interval n,
- HU_n is the set of hot utilities, present in interval n, or above,
- CU_n is the set of cold utilities, present in interval n,
- $WR^{l_i}{}_k$ is the mass exchange load of substream l_i, in interval k,

$$WR^{l_i}_k = G_{l_i}(y_k - \max(y_{k+1}, y_i spt)),$$

- $WS^{l_j}{}_k$ is the mass exchange load of substream l_j, in interval k,

$$WS^{l_j}_k = L_{l_j}(\min(x^t_j, x_{jk}) - x_{jk+1}),$$

- $\delta_{l_i k}$ is the residual mass load of substream l_i in interval k,
- $M_{l_i l\prime_j k}$ is the mass exchange load between l_i and l'_j, in k,
- FM is the set of mass exchanging substreams that are at different temperatures,
- $QS_{l_s n}$ is the heat load of substream l_s in interval n,
- $Q_{l_s l_{s\prime} \prime n}$ is the heat exchange load between l_s and $l_{s'}{}'$ in interval n,
- $\theta_{l_s n}$ is the residual heat load of hot substream l_s in interval n,
- QHU_{hn} is the heat load of hot utility h in interval n,
- QCU_{cn} is the heat load of cold utility c in interval n,
- $QH_{hl_s n}$ is the heat exchange load between hot utility h and l_s in interval n,
- θ^h_{hn} is the residual heat load of hot utility h in interval n,
- $QC_{l_s cn}$ is the heat exchange load between l_s and cold utility c in interval n.

Problem (TP3) results in the minimum utility cost and the corresponding flows of separating agents and heating/cooling utility streams, the optimal decomposition of each stream into substreams of fixed mass exchange temperature and the mass and heat exchange pinch points and corresponding subnetworks.

The minimum operating cost of the combined MHEN can alternatively be found applying the first and second thermodynamic laws (property in (2)) on the composition and temperature interval diagrams [13].

The minimum number of mass and heat exchangers is determined in a second step through the expanded MILP transshipment model, separately in each mass and heat exchanger subnetwork. The final network configurations and unit sizes are determined in a final step, applying heuristic rules or superstructure models.

Additional disadvantages of the sequential MHEN synthesis method, compared to the synthesis of simple MEN, are that:

i) the mass and heat exchange networks are assumed separable and
ii) the intermediate mass exchange temperatures are decided in the first step; this forbids full exploitation of the mass/heat integration trade-offs, as capital cost implications of such decision is not accounted for.

Modeling concepts from the sequential mass and heat exchanger network synthesis methods, employing LP and MILP optimization models, have been extended to explore distillation networks [1], pervaporation systems [12] and other energy-requiring separation networks [2,7].

See also

- Chemical Process Planning
- Extended Cutting Plane Algorithm
- Generalized Benders Decomposition
- Generalized Outer Approximation
- Global Optimization of Heat Exchanger Networks
- MINLP: Application in Facility Location-Allocation
- MINLP: Applications in Blending and Pooling Problems
- MINLP: Applications in the Interaction of Design and Control
- MINLP: Branch and Bound Global Optimization Algorithm
- MINLP: Branch and Bound Methods
- MINLP: Design and Scheduling of Batch Processes
- MINLP: Generalized Cross Decomposition
- MINLP: Global Optimization with αBB
- MINLP: Heat Exchanger Network Synthesis
- MINLP: Logic-based Methods
- MINLP: Outer Approximation Algorithm
- MINLP: Reactive Distillation Column Synthesis
- Mixed Integer Linear Programming: Heat Exchanger Network Synthesis
- Mixed Integer Nonlinear Programming

References

1. Bagajewicz MJ, Manousiouthakis V (1992) Mass/heat exchange network representation of distillation networks. AIChE J 38:1769–1800
2. El-Halwagi MM, Hamad AA, Garrison GW (1996) Synthesis of waste interception and allocation networks. AIChE J 42:3087–3101
3. El-Halwagi MM, Manousiouthakis V (1989) Synthesis of mass exchange networks. AIChE J 35:1233–1243
4. El-Halwagi MM, Manousiouthakis V (1990) Automatic synthesis of mass exchange networks with single component targets. Chem Eng Sci 45:2813–2831
5. El-Halwagi MM, Manousiouthakis V (1990) Simultaneous synthesis of mass-exchange and regeneration networks. AIChE J 36:1209–1219
6. El-Halwagi MM, Srinivas BK (1992) Synthesis of reactive mass exchange networks. Chem Eng Sci 47:2113–2119
7. El-Halwagi MM, Srinivas BK, Dunn RF (1995) Synthesis of optimal heat-induced separation networks. Chem Eng Sci 50:81–97
8. Gupta A, Manousiouthakis V (1993) Minimum utility cost of mass exchange networks with variable single component supplies and targets. Industr Eng Chem Res 32:1937–1950
9. Gupta A, Manousiouthakis V (1996) Variable target mass-exchange network synthesis through linear programming. AIChE J 42:1326–1340
10. Kiperstok A, Sharrat PN (1995) On the optimization of mass exchange networks for removal of pollutants. Chem Eng Res Des 73:271–277
11. Papalexandri KP, Pistikopoulos EN, Floudas CA (1994) Mass exchange networks for waste minimization: A simultaneous approach. Chem Eng Res Des 72:279–294
12. Srinivas BK, El-Halwagi MM (1993) Optimal design of pervaporation systems for waste reduction. Comput Chem Eng 17:957–970
13. Srinivas BK, El-Halwagi MM (1994) Synthesis of combined heat and reactive mass exchange networks. Chem Eng Sci 49:2059–2074

14. Srinivas BK, El-Halwagi MM (1994) Synthesis of reactive mass exchange networks with general nonlinear equilibrium relations. AIChE J 40:463–472

Mixed Integer Nonlinear Bilevel Programming: Deterministic Global Optimization

ZEYNEP H. GÜMÜŞ[1,2],
CHRISTODOULOS A. FLOUDAS[3]
[1] Department of Physiology and Biophysics, Weill Medical College, Cornell University, New York, USA
[2] The HRH Prince Alwaleed Bin Talal Bin Abdulaziz Institute for Computational Biomedicine, Weill Medical College, Cornell University, New York, USA
[3] Department of Chemical Engineering, Princeton University, Princeton, USA

Article Outline

Keywords and Phrases
Introduction
Formulation
Classes
BLPs with Inner Integer Variables
Reformulation/Linearization
Inner Problem KKT Conditions and Complementarity
Active Set Strategy
Transformed BLPP Global Optimization
Global Optimization Algorithm
Illustrative Example
Conclusions
References

Keywords and Phrases

Bilevel programming; Bilevel nonlinear; Bilevel optimization; Mixed integer optimization; Global optimization; Two-level optimization; Mixed integer nonlinear

Introduction

The global optimization of classes of mixed integer nonlinear bilevel optimization problems is addressed. For problems where the integer variables participate in both the inner and the outer problems, the outer level may involve general mixed-integer nonlinear functions. The inner level may involve functions that are mixed-integer nonlinear in outer variables, linear, polynomial, or multilinear in inner integer variables and linear in inner continuous variables. The technique is based on reformulating the mixed-integer inner problem as continuous by its convex hull representation [11,12] and solving the resulting nonlinear bilevel optimization problem by a novel deterministic global optimization framework.

Formulation

The general mixed-integer nonlinear Bilevel Programming Problem (BLP) formulation is:

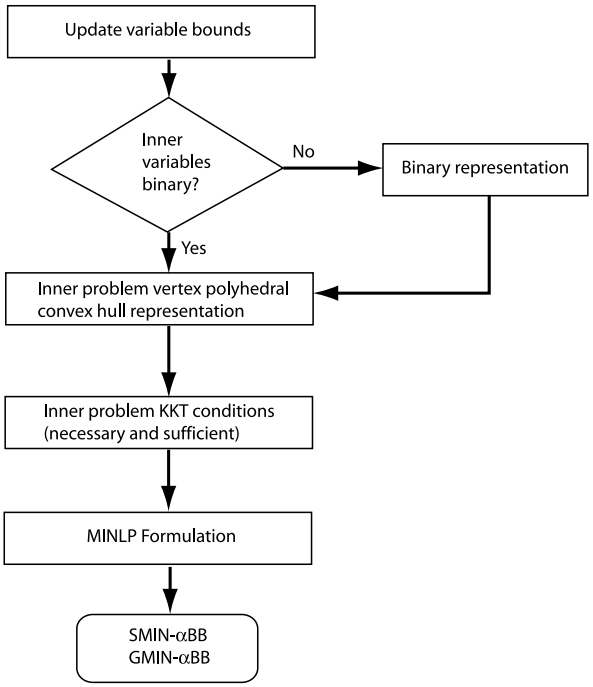
$$\begin{aligned} &\min_{x} F(\boldsymbol{x}, \boldsymbol{y}) \\ &\text{s.t. } \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y}) \geq 0 \\ &\quad \boldsymbol{H}(\boldsymbol{x}, \boldsymbol{y}) = 0 \\ &\quad \min_{y} f(\boldsymbol{x}, \boldsymbol{y}) \\ &\quad \text{s.t. } \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \geq 0 \\ &\qquad \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) = 0 \\ &\quad x_1, .., x_i \in \Re^{n_1}, y_1, .., y_j \in \Re^{n_2}, \\ &\quad x_{i+1}, .., x_{n_1} \in Z^+, y_{j+1}, .., y_{n_2} \in Y_{\text{IN}} \subseteq Z^+ \,. \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is a vector of outer problem variables, of which i are continuous and $n_1 - i$ are integer, y is a vector of inner problem variables, of which j are continuous and $n_2 - j$ are integer, $F(\boldsymbol{x}, \boldsymbol{y})$ is the outer objective function, $\boldsymbol{H}(\boldsymbol{x}, \boldsymbol{y})$ are outer equality constraints, $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y})$ are outer inequality constraints, $f(\boldsymbol{x}, \boldsymbol{y})$ is the inner objective function, $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y})$ are inner equality constraints, and $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y})$ are inner inequality constraints. The applications of BLP are many and diverse [4,6,7]; if these problems involve discrete decisions in addition to continuous ones, then the mixed-integer BLP models arise.

Classes

The nonlinear mixed integer BLP can be classified into four different categories, depending on the existence of integer variables in the outer or the inner problems:

(I). Integer Upper, Continuous Lower BLP;
(II). Purely Integer BLP;
(III). Continuous Upper, Integer Lower BLP;
(IV). Mixed-Integer Upper and Lower BLP.



Mixed Integer Nonlinear Bilevel Programming: Deterministic Global Optimization, Figure 1
Algorithm flowsheet for type II,III,IV BLPs

The existence of both integer and nonlinear terms in the above problem classes require special solution techniques. The specific mathematical structure of the mixed integer nonlinear BLP is of great import in developing corresponding solution strategies. Problems of Type I can be addressed with existing BLP solution approaches. For problems of Type II, enumeration methods can be applied. However, BLPs of Type III and IV are the most difficult to solve.

BLPs with Inner Integer Variables

The conventional solution method of the continuous BLP is to transform it into a single level problem by replacing the inner problem with the set of equations that define its Karush–Kuhn–Tucker (KKT) optimality conditions. However, the KKT optimality conditions use gradient information, so the conventional approach is not applicable when integer inner variables exist. Furthermore, if the integrality constraint is relaxed on the inner integer variables, the solution of this relaxed BLP does not provide a valid lower bound on the solution of the mixed-integer BLP [9]. Note that even if the optimal solution of the relaxed BLP is integral in $\boldsymbol{y}$, this may not be a globally optimal solution of the original BLP [9]. Thus, the conventional KKT-based methods inherently fail in locating the global optimum.

The BLP with inner mixed-integer variables can be transformed into an equivalent BLP with inner mixed-binary (0-1) variables as follows. Every inner problem integer variable y_j, with upper and lower bounds $y_j^{\mathrm{L}} \le y_j \le y_j^{\mathrm{U}}$ is converted into a set of binary variables using the formula [2]:

$$y_j = y_j^{\mathrm{L}} + z_{j1} + 2z_{j2} + 4z_{j3} + \ldots + 2^{(N_j-1)} z_{jN} \quad (2)$$

where $\boldsymbol{z}_j$ is a vector of (0-1) variables and N_j is the minimum number of (0-1) variables needed:

$$N_j = 1 + \mathrm{INT}\left(\frac{\log(y_j^{\mathrm{U}} - y_j^{\mathrm{L}})}{\log(2)}\right) \quad (3)$$

such that INT truncates its real argument to an integer.

The only time that the KKT optimality conditions are applicable to solve the BLP with mixed-binary $\boldsymbol{y}$ is when the following property is satisfied [8]:

Property 1 *If the inner problem constraint set, Y_{IN}, defines a vertex polyhedral convex hull and all the vertices of the convex hull lie in Y_{IN}, then the optimal inner problem integer solution is equivalent to its linear programming relaxation. As a result, the Karush–Kuhn–Tucker, KKT, conditions of relaxed inner linear problem are necessary and sufficient to define the optimal inner problem integer solution.*

The property is also satisfied when outer variables exist in the inner problem constraints, such that the inner problem vertex polyhedral convex envelope is defined *parametrically in* $\boldsymbol{x}$. Hence, the integer solution of the inner problem lies at a vertex of the inner solution set and the KKT optimality conditions locate the true optimal solution [8].

Here, a global optimization procedure is presented for BLPs of Type II, III and IV that is based on a reformulation/linearization scheme combined with a global optimization framework. The idea is that if the inner problem constraint set has a vertex polyhedral convex envelope, then *Property 1* is satisfied and the mixed-integer inner problem can be converted into a continuous problem of equivalent form. The application of the reformulation/linearization technique results in the

convex hull representation for several classes of inner problems.

Reformulation/Linearization

The mixed-binary inner problem constraint set is transformed into the continuous domain by converting it into a polynomial programming problem and then relinearizing it into an extended linear problem by a method based on [11]. First, a polynomial factor is defined as follows:

$$F_n(J_1, J_2) = \left\{ \begin{array}{l} \left(\prod_{j\in J_1} y_j\right)\left(\prod_{i\in J_2}(1-y_j)\right), \\ J_1, J_2 \subseteq N_y \equiv 1, .., n_y, \\ \text{s.t. } J_1 \cap J_2 = \emptyset,\ |J_1 \cup J_2| = n_y \end{array} \right\} . \tag{4}$$

Using this polynomial factor, the convex hull of the inner problem, Y_{IN}, is obtained. If the inner optimization problem is linear, then the 2-step process is as follows:

Step 1 Reformulation. Multiply every constraint, including $0 \le y \le 1$, with every factor defined as above and use the relationship $y_j^2 = y_j$, $\forall j = 1, .., n_y$ to linearize terms polynomial in $\boldsymbol{y}$. Include in the inner problem constraint set the nonnegativities on all possible factors of degree n_y (i. e. $F_n(J_1, J_2) \ge 0$ for all (J_1, J_2) of order n_y).

Step 2 Linearization. Linearize the inner constraints that are multilinear in $\boldsymbol{y}$, such as $\Pi_{j\in J} y_j$ by substituting a z_J for each set J with $|J| \ge 2$, with the elements of J in increasing order. (i. e. a new variable z_{ij} is introduced to substitute for a bilinear term $(y_i y_j = z_{ij})$ and further substitution is performed for multilinear terms). At constant $\boldsymbol{x}$, the resulting inner constraint set describes a polytope with all vertices defined by binary values and characterizes the convex hull of feasible solutions for any inner problem that is linear or polynomial binary in $\boldsymbol{y}$-variables.

If the inner optimization problem is mixed-binary linear or polynomial, the problem constraints are again multiplied by n_y-degree polynomial factors composed of the n_y binary variables and their complements and the resulting nonlinear problem is linearized by a substitution of new variables. Additional nonlinear terms arise from the multiplication of the n_y-degree polynomial factors with the inner problem linear continuous terms in $\boldsymbol{y}$, that are also linearized through a redefinition [8,12]. This transformation is applicable when in the mixed-binary inner problem, the continuous $\boldsymbol{y}$ are $0 \le y \le 1$. Note that there are no such restrictions on the outer problem $\boldsymbol{x}$-variables in both inner and outer problems.

Inner Problem KKT Conditions and Complementarity

After reformulation/relinearization, the inner problem is replaced by the set of equations that define its necessary and sufficient KKT optimality conditions:

$$\begin{aligned} h_i^r(x, y^*) &= 0, \quad i \in I\ , \\ \frac{\partial f(x, y^*)}{\partial y^*} &+ \sum_{j=1}^{J} \lambda_j \frac{\partial g_j^r}{\partial y^*} + \sum_{j=1}^{J} \mu_i^* \frac{\partial h_i^r}{\partial y^*} = 0\ , \\ g_j^r(x, y^*) + s_j^* &= 0\ , \quad j \in J \\ \lambda_j^* s_j^* &= 0\ , \quad j \in J \quad \text{(CS)} \\ \lambda_j^* s_j^* &\ge 0\ , \quad j \in J \end{aligned} \tag{5}$$

where f^r, $\boldsymbol{h}^r$ and $\boldsymbol{g}^r$ are the reformulated inner objective, equality and inequality constraints, $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the Lagrange multipliers of the inner inequality and equality constraints, and $\boldsymbol{s}$ are the slack variables associated with the complementarity constraints.

Active Set Strategy

The complementarity condition constraints, (CS) involve discrete decisions on the choice of the inner problem active constraint set. The set changes when at least one inequality function and its Lagrange multiplier are equal to zero. This imposes a major difficulty in the solution of the transformed problem. To overcome this difficulty, the Active Set Strategy [5,8] is employed, such that the complementarity constraints are reformulated as:

$$\begin{aligned} \lambda_j - U Y_j &\le 0\ , \quad j \in J \\ s_j - U(1 - Y_j) &\le 0\ , \quad j \in J \\ \lambda_j, s_j &\ge 0\ , \quad j \in J \\ Y_j &\in \{0, 1\}\ . \end{aligned} \tag{6}$$

where U is an upper bound on the slack variables $\boldsymbol{s}$ and $\boldsymbol{Y}$ are the additional binary variables introduced. If constraint j is active, $(Y_j = 1)$, and if inactive, $(Y_j = 0)$. Note that now the integer variable set includes the binary variables $\boldsymbol{Y}$ in addition to the outer problem integer variables.

Transformed BLPP Global Optimization

The problem that results after the reformulated/linearized inner problem is replaced with its KKT optimality conditions and active set strategy is applied can still have nonlinear terms due to complementarity and stationarity conditions. Further, nonlinear terms in the outer problem variables may exist in either the inner or outer problem constraints. Hence, the resulting problem is a mixed-integer (nonlinear) optimization problem and should be solved by a global optimization procedure. If the integer variables are all binary and only appear in linear or mixed-bilinear terms, the Special structure MINLP-αBB, SMIN-αBB [1,3] approach is employed. If the outer integer variables are not restricted to binary and/or participate in nonlinear terms, the General structure MINLP-αBB, GMIN-αBB [1,3] approach is employed. The steps of the proposed framework are given below.

Global Optimization Algorithm

Step 1 Establish variable bounds by solving the problems:

$\boldsymbol{y}^{\mathrm{L}}$, $\boldsymbol{y}^{\mathrm{U}}$ = min $\boldsymbol{y}$, $-\boldsymbol{y}$ s.t. inner problem constraint setprotect

to obtain simple lower and upper bounds on $\boldsymbol{y}$,

$\boldsymbol{y}^{\mathrm{L}} \le \boldsymbol{y} \le \boldsymbol{y}^{\mathrm{U}}$.

Step 2 If the inner integer variables are integer, convert into a set of binary variables by Eq. (2) and Eq. (3).

Step 3 Obtain the vertex polyhedral convex envelope of the inner problem feasible region via the reformulation/ linearization [11]. The inner problem is now linear in both inner binary and continuous variables and parametric in outer problem variables, $\boldsymbol{x}$.

Step 4 Replace the inner problem with the set of equations that define its necessary and sufficient KKT optimality conditions. The resulting problem is single level.

Step 5 Solve the resulting single level optimization problem to global optimality. The inner integer variables are all separable, linear and binary at the beginning of this step. If the final problem is a Mixed Integer Linear Problem, (MILP), then use CPLEX. Notice that the problem will be an MILP only for the simplest cases. If there are nonlinear continuous variables, but the integer variables are all binary, linear and separable, use SMIN-αBB [1,3] global optimization procedure. If the outer problem has nonlinear integer terms, then use GMIN-αBB [1,3] global optimization procedure.

Illustrative Example

The following problem [10] can not be solved to global optimality using current deterministic solution approaches for integer bilevel programming problems in the literature.

$$
\begin{aligned}
&\min -\left(-\frac{2}{5}x_1^2x_2 + 4x_2^2\right) y_1y_2 \\
&\quad -\left(-x_2^3 + 3x_1^2x_2\right)(1-y_1)y_2 - \left(2x_2^2 - x_1\right)(1-y_2) \\
&\text{s.t. } \min -\left(x_1x_2^2 + 8x_2^3 - 14x_1^2 - 5x_1\right) y_1y_2 \\
&\quad -\left(-x_1x_2^2 + 5x_1x_2 + 4x_2\right)(1-y_1)y_2 \\
&\quad -8x_1y_1(1-y_2) \\
&\text{s.t. } y_1 + y_2 \ge 1 \\
&\quad 0 \le x_1 \le 10 \\
&\quad 0 \le x_2 \le 10 \\
&\quad y_1, y_2 \in \{0,1\}^2\,, \quad x \in \Re\,.
\end{aligned}
\tag{7}
$$

Steps 1–2 Variable bounds are already given in this problem, with $0 \le x \le 10$, and y_1 and y_2 are defined as binary.

Step 3 Determination of the vertex polyhedral convex hull: The inner problem is $Ny = 2$ degrees, second degree factors y_1y_2, $y_1(1-y_2)$, $(1-y_1)y_2$, $(1-y_1)(1-y_2)$ multiply the inner problem constraint $y_1 + y_2 - 1 \ge 0$ and result in:

$$
\begin{aligned}
y_1y_2 &\ge 0 \\
y_1 + y_2 - y_1y_2 - 1 &\ge 0\,.
\end{aligned}
\tag{8}
$$

Linearization: Assign a new variable for the bilinear term $z_{12} = y_1y_2$ that leads to the additional constraints:

$$
\begin{aligned}
z_{12} &\geq 0 \\
y_1 - z_{12} &\geq 0 \\
y_2 - z_{12} &\geq 0 \\
-y_1 - y_2 + z_{12} &\geq -1 \, .
\end{aligned} \tag{9}
$$

From Eqs. (8) and (9), $y_1 + y_2 - z_{12} - 1 = 0$. Substituting the definition of z_{12} into Eq. (8), a linear relaxation of the inner problem constraint set leads to the original set of constraints:

$$
\begin{aligned}
y_1 + y_2 - 1 &\geq 0 \\
1 - y_2 &\geq 0 \\
1 - y1 &\geq 0 \, .
\end{aligned} \tag{10}
$$

Hence, the continuous relaxation of the original problem constraints define the convex hull and no additional constraints are necessary. The inner problem is continuous and linear in y_1 and y_2, and parametric in the outer problem variables $\boldsymbol{x}$.

Step 4 Replace the relaxed inner problem with the equivalent set of equations that define its necessary and sufficient KKT optimality conditions:

$$
\begin{aligned}
&\min \left(\frac{17}{5} x_1^2 x_2 - 4x_2^2 - x_2^3 \right) y_1 + \left(\frac{2}{5} x_1^2 x_2 - 2x_2^2 - x_1 \right) \\
&\cdot y_2 \left(-\frac{17}{5} x_1^2 x_2 + 2x_2^2 + x_2^3 + x_1 \right) \\
&0 \leq x_1 \leq 10 \\
&0 \leq x_2 \leq 10 \\
&- x_1^2 x_2^2 - 8x_2^3 + 14x_1^2 + 5x_1 - x_1 x_2^2 + 5x_1 x_2 + 4x_2 \\
&\quad - \lambda_1 + \lambda_2 = 0 \\
&- \lambda_1 - x_1^2 x_2^2 - 8x_2^3 + 14x_1^2 + 13x_1 + \lambda_3 = 0 \\
&- y_1 - y_2 + 1 + s_1 = 0 \\
&y_1 + s_2 = 1 \\
&y_2 + s_3 = 1 \\
&\lambda_1 - UY \leq 0 \\
&s_1 + UY \leq U \\
&Y \in \{0, 1\}; \;\; s_1, \lambda_1, y_1, y_2 \geq 0; \\
&x_1, x_2 \in \Re; \;\; y_1, y_2 \leq 1 \, .
\end{aligned} \tag{11}
$$

Step 5 The single level problem constraint contains the following nonlinear terms: $x_1^2 x_2 y_1$, $x_2^2 y_1$, $x_1^2 x_2 y_2$, $x_2^2 y_2$, $x_1 y_2$, $x_1^2 x_2$, $x_2{}^2$, $x_2{}^3$, $x_1 x_2^2$, $x_1 x_2$, $x_1^2 x_2^2$ that should be underestimated. All integer variables are binary, linear and separable. Solve the resulting single level problem to global optimality using SMIN-αBB [1,3]. The global optimal solution reported in [10] is at $(x_1^*, x_2^*, y_1^*, y_2^*) = (6.038, 2.957, 0, 1)$. We identify the lower global solution at (0, 10, 1, 1). Note that the solution of this problem by enumeration methods could be labor intensive due to the presence of continuous variables.

Conclusions

The global optimization framework addresses the solution of several classes of mixed integer nonlinear bilevel optimization problems. The outer problem may be mixed-integer nonlinear in both inner and outer variables; the inner problem may be mixed-integer nonlinear in outer variables, linear, polynomial or multilinear in inner integer variables and linear in inner continuous variables. This is based on the reformulation of the mixed-integer inner problem feasible space to generate its convex hull, where the vertices correspond to binary solutions. This allows the equivalence of the inner optimization problem to the set of equations that define its KKT optimality conditions, with which it is replaced. The resulting single level optimization problem is solved to global optimality. This is arguably the first deterministic global optimization technique that can solve several classes of mixed-integer nonlinear bilevel optimization problems. Note that if the central decision maker wants to locate the second-best inner or outer integer solutions, simple integer cuts [2] can be added prior to applying the relevant solution strategy.

References

1. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αbb, for general twice-differentiable constrained nlps i. theoretical advances. Comput Chem Eng 22:1137–1158
2. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. In: Topics in Chemical Engineering. Oxford University Press, New York
3. Floudas CA (2000) Deterministic Global Optimization: Theory, Methods and Applications. Nonconvex Optimization and its Applications, vol 37. Kluwer, Dordrecht
4. Floudas CA, Gümüş ZH, Ierapetritou M (2001) Global optimization in design under uncertainty: feasibility test and flexibility index problems. Ind Eng Chem Res 40(20): 4267–82
5. Grossmann IE, Floudas CA (1987) Active constraint strategy for flexibility analysis in chemical processes. Comp Chem Eng 11(6):675

6. Gümüş ZH, Ciric AR (1997) Reactive distillation column design with vapor/liquid/liquid equilibria. Comp Chem Eng 21(S):S983-S988
7. Gümüş ZH, Floudas CA (2001) Global optimization of nonlinear bilevel optimization problems. J Glob Optim 20:1–31
8. Gümüş ZH, Floudas CA (2005) Global optimization of mixed-integer bilevel programming problems. Comp Man Sci 2:181–212
9. Moore JT, Bard JF (1990) The mixed integer linear bilevel programming problem. Oper Res 38:911
10. Sahin KH, Ciric AR (1998) A dual temperature simulated annealing approach for solving bilevel programming problems. Comp Chem Eng 23:11–25
11. Sherali HD, Adams WP (1990) A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J Discret Math 3:411–430
12. Sherali HD, Adams WP (1994) A hierarchy of relaxations and convex-hull characterizations for mixed-integer zero-one programming problems. Discret Appl Math 52:83–106

Mixed-Integer Nonlinear Optimization: A Disjunctive Cutting Plane Approach

YUSHAN ZHU
Department of Chemical Engineering,
Tsinghua University, Beijing, China

MSC2000: 49M37, 90C11

Article Outline

Keywords
Introduction
Formulation
Branch-And-Cut Procedure
Linear Approximation of NLP Relaxation
Disjunctive Cut Generation
Cut Lifting
Conclusion
References

Keywords

Mixed-integer nonlinear programming; Branch-and-cut; Disjunctive cutting plane; Lift-and-project; MINLP

Introduction

The mixed-integer nonlinear programming (MINLP) approach was widely used to model and solve the process synthesis problems in chemical engineering filed during the last two decades within the superstructure framework that always involves discrete and continuous variables [3,4,5,6,10]. Recently, the successful employment of the branch-and-cut method for 0–1 integer programming [7,8] and 0–1 mixed-integer linear programming [1,2] has spurred great interest in its application for 0–1 mixed-integer nonlinear optimization due to the significant progress of interior point algorithm for convex optimization problems. Stubbs and Mehrotra [9] generalized the lift-and-project cut or the disjunctive cut for 0–1 integer or mixed-integer linear programming proposed in [1,2,7,8], and extended their method into a branch-and-cut algorithm for the 0–1 mixed-integer nonlinear optimization problem. The disjunctive cutting plane presented by Stubbs and Mehrotra [9] is obtained by solving a convex projection problem, so it is computationally expensive. In [11], a valid disjunctive cutting plane for mixed-integer nonlinear optimization problems was constructed by solving a linear programming problem implemented in an algorithmic package named MINO, i. e., Mixed-Integer Nonlinear Optimizer.

Formulation

The general 0–1 mixed-integer nonlinear optimization problems can be formulated as

$$(P)\left|\begin{array}{ll}\min\limits_{x,y} & dx\\ s.t. & Ax+Gy\le b\\ & g_i(x,y)\le 0,\quad i=1,\ldots,l\\ & x\in\Re^n,\quad y\in\{0,1\}^q\end{array}\right.$$

where the constant vectors and matrices are defined as $d\in\Re^n, A\in\Re^{m\times n}, G\in\Re^{m\times q}, b\in\Re^m$. Let the feasible region of the standard continuous relaxation of problem (P) be defined as

$$C=\left\{(x,y)\in\Re^{n+q}\left|\begin{array}{l}Ax+Gy\le b,\\ g_i(x,y)\le 0,\quad i=1,\ldots,l,\\ 0\le y\le 1,\end{array}\right.\right\}$$

Hence, the feasible set of (P) can be formulated as

$$C^0=\left\{(x,y)\in C:\ y_j\in\{0,\ 1\},\quad j=1,\ldots,q\right\}.$$

At a generic step of the branch-and-cut algorithm, let $(\overline{x}, \overline{y})$ be a solution to the current NLP relaxation of (P). If any of the components of the binary variables $\overline{y}$ are not in $\{0, 1\}$, we can add a valid inequality into the current feasible set such that this inequality is violated by $(\overline{x}, \overline{y})$. We denote by c the family of inequalities to describe the current feasible set and the newly incorporated inequalities. We denote by $F_0, F_1 \subseteq \{1, \ldots, q\}$ the sets of binary variables that have been fixed at 0 and 1, respectively. Let

$$K(C, F_0, F_1) = \left\{ (x, y) \in C \;\middle|\; \begin{array}{l} y_j = 0 \text{ for} j \in F_0 \\ y_j = 1 \text{ for} j \in F_1 \end{array} \right\}$$

And let $NLP(C, F_0, F_1)$ denote the nonlinear program

$$\begin{array}{ll} \min\limits_{x,y} & dx \\ \text{s.t.} & (x, y) \in K(C, F_0, F_1) \end{array}$$

The active nodes of the enumeration tree are represented by a list of S with ordered pairs (F_0, F_1). Let UBD represent the current upper bound, i. e., the value of the best-known solution to problem (P).

Branch-And-Cut Procedure

Input of d, n, q, A, G, b, g_i $(i = 1, \ldots, l)$:

(1) Initialization. Set $S = \{(F_0 = \phi, F_1 = \phi)\}$, and let C consist of the nonlinear programming relaxation of (P) and $UBD = \infty$.

(2) Node Selection. If $S = \phi$, stop. Otherwise, choose an ordered pair $(F_0, F_1) \in S$ and remove it from S.

(3) Lower Bounding Step. Solve the nonlinear program $NLP(C, F_0, F_1)$. If the problem is infeasible, go to *Step 2*. Otherwise, let $(\overline{x}, \overline{y})$ denote its optimal solution. If $d\overline{x} \geq UBD$, go to *Step 2*. If $\overline{y}_j \in \{0,\ 1\}$, $j = 1, \ldots, q$, let $(x^*, y^*) = (\overline{x}, \overline{y})$, $UBD = d\overline{x}$, and go to *Step 2*.

(4) Branching versus cutting decision. Should cutting planes be generated? If yes, go to *Step 5*, else go to *Step 6*.

(5) Cut generation. Generate cutting plane $\alpha x + \beta y \leq \gamma$ valid for (P) but violated by $(\overline{x}, \overline{y})$. Add the cuts into C and go to *Step 3*.

(6) Branching Step. Pick an index $j \in \{1, \ldots, q\}$ such that $0 < \overline{y}_j < 1$. Generate the subproblems corresponding to $(F_0 \cup \{j\}, F_1)$ and $(F_0, F_1 \cup \{j\})$, add them into the node set S. Go to *Step 2*.

When the algorithm terminates, if $UBD < \infty$, (x^*, y^*) is an optimal solution to (P), otherwise (P) is infeasible.

Linear Approximation of NLP Relaxation

The continuous relaxation of problem (P) at some node in an enumeration tree can be described by

$$(NLP) \left| \begin{array}{l} \min\limits_{x,y} dx \\ s.t.\ \tilde{A}x + \tilde{G}y \leq \tilde{b}, \\ \quad g_i(x, y) \leq 0, \quad i = 1, \ldots, l, \\ \quad y_j = 0, \quad j \in F_0, \\ \quad y_j = 1, \quad j \in F_1, \\ \quad (x, y) \in \Re^{n+q}, \end{array} \right.$$

where the reformulated linear constraint set consists of the original linear constraint set and the upper and lower bound constraints for binary variables, so we have $\tilde{A} \in \Re^{(m+2q)\times n}$, $\tilde{G} \in \Re^{(m+2q)\times q}$, $\tilde{b} \in \Re^{m+2q}$. Assume that the above NLP continuous problem is feasible and has a finite minimum at $(\overline{x}, \overline{y})$, since otherwise the node is done. A linear approximation problem at $(\overline{x}, \overline{y})$ for the above NLP problem can be obtained by

$$(LP) \left| \begin{array}{l} \min\limits_{x,y} dx \\ s.t.\ \tilde{A}x + \tilde{G}y \leq \tilde{b}, \\ \quad g_i(\overline{x}, \overline{y}) + \nabla g_i(\overline{x}, \overline{y}) \begin{pmatrix} x - \overline{x} \\ y - \overline{y} \end{pmatrix} \leq 0, \\ \quad i = 1, \ldots, l, \\ \quad y_j = 0, \quad j \in F_0, \\ \quad y_j = 1, \quad j \in F_1, \\ \quad (x, y) \in \Re^{n+q}, \end{array} \right.$$

where the original convex and differentiable functions are replaced by their first-order Taylor approximation at $(\overline{x}, \overline{y})$. Accordingly, a MILP problem corresponding to the MINLP problem at the current node can be described by

$$(MILP) \left| \begin{array}{l} \min\limits_{x,y} dx \\ s.t.\ \tilde{A}x + \tilde{G}y \leq \tilde{b}, \\ \quad g_i(\overline{x}, \overline{y}) + \nabla g_i(\overline{x}, \overline{y}) \begin{pmatrix} x - \overline{x} \\ y - \overline{y} \end{pmatrix} \leq 0, \\ \quad i = 1, \ldots, l, \\ \quad y_j = 0, \quad j \in F_0, \\ \quad y_j = 1, \quad j \in F_1, \\ \quad x \in \Re^n, y \in \{0, 1\}^q, \end{array} \right.$$

[11] proved that if the above NLP achieves its optimal solution at $(\overline{x}, \overline{y})$. Then, $(\overline{x}, \overline{y})$ is also an optimal solution to the aforementioned LP. The geometrical explanation of the above linear approximation is presented in Fig. 1, and it is obvious that the mixed-integer set is expanded after linear approximation.

Disjunctive Cut Generation

For problem (P), it is very attractive to construct the lift-and-project cut in terms of the approximated LP instead of the NLP, but the cut still can cut away the fraction point $(\overline{x}, \overline{y})$. Such cut can be derived by imposing the 0–1 integral condition on a binary variable y_j while $0 < \overline{y}_j < 1$. In Fig. 1, the short dashed line represents the cut generated directly by using the convex hull of the mixed-integer convex set presented by [9], and the long dashed line stands for the cut to be generated in [11] based on the linear approximation. For cut generation, the node sets F_0 and F_1 can be expanded to include additional binary variables whose optimal solutions are taken at 0 or 1 for the NLP problem at the current node. Then, we redefine these two sets as $\overline{F}_0 = \{i : \overline{y}_i = 0\}$ and $\overline{F}_1 = \{i : \overline{y}_i = 1\}$. It is not difficult to verify that the above NLP and LP problems have the same optimal solutions if we change the original node sets to be the expanded ones. Let the feasible region of the above LP be defined as

$$K = K\left(C, \overline{F}_0, \overline{F}_1\right) = \left\{ (x, y) \in \Re^{n+q} \;\middle|\; \begin{array}{l} \overline{A}x + \overline{G}y \le \overline{b} \\ y_i = 0, i \in \overline{F}_0 \\ y_i = 1, i \in \overline{F}_1 \end{array} \right\}$$

where $\overline{A} \in \Re^{(m+2q+l)\times n}, \overline{G} \in \Re^{(m+2q+l)\times q}, \overline{b} \in \Re^{m+2q+l}$, i. e., the newly reformulated linear constraint set consists of the linear approximation set besides the original one, as

$$\overline{A}x + \overline{G}y \le \overline{b} \quad \equiv \begin{cases} Ax + Gy \le b, \\ \nabla g^x\left(\overline{x}, \overline{y}\right) x + \nabla g^y\left(\overline{x}, \overline{y}\right) y \\ \le \nabla g\left(\overline{x}, \overline{y}\right) \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} - g\left(\overline{x}, \overline{y}\right), \\ -y_i \le 0 \quad i = 1, \ldots, q, \\ y_i \le 1 \quad i = 1, \ldots, q, \end{cases}$$

It should be noted that the node sets have already been changed to the expanded ones in the above formulation. If we impose the integrality condition on a binary variable y_j for which $0 < \overline{y}_j < 1$, the disjunctive cut can be obtained by choosing a valid inequality for

$$P_j(K) = conv\left(K \cap \left\{(x, y) \in \Re^{n+q} : y_j \in \{0, 1\}\right\}\right),$$

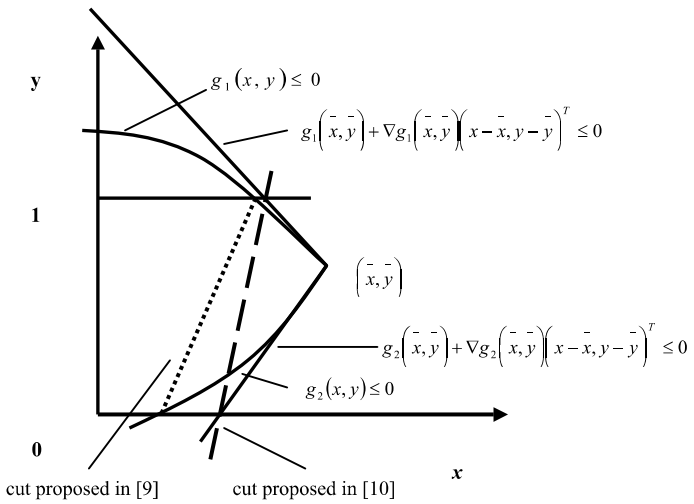
The convex hull of this union set can be further described by its disjunctive form as

$$P_j(K) = conv\left(\left\{K \cap \left\{(x, y) \in \Re^{n+q} : y_j \le 0\right\}\right\} \cup \left\{K \cap \left\{(x, y) \in \Re^{n+q} : -y_j \le -1\right\}\right\}\right),$$

Let $F = \{1, \ldots, q\} \setminus \left(\overline{F}_0 \cup \overline{F}_1\right)$ denote the set of free variables at node $\left(\overline{F}_0, \overline{F}_1\right)$, and the vector corresponding to those free variables can be defined as $y^F = y \setminus \{y_i : i \in \left(\overline{F}_0 \cup \overline{F}_1\right)\}$. The columns of matrix $\overline{G}$ corresponding to the fixed binary variables can be removed from the constraint set by defining $\overline{G}^F = \overline{G} \setminus \{\overline{G}_i : i \in \left(\overline{F}_0 \cup \overline{F}_1\right)\}$, and the right-hand side can be calculated accordingly as $\overline{b}^F = \overline{b} - \sum_{i \in F_1} \overline{G}_i$. Finally, the rows in matrices $\overline{A}$ and $\overline{G}^F$, and vector $\overline{b}^F$ that correspond to the upper and lower bounds of the fixed binary variables are removed. After doing the above operations, we can assume without loss of generality, that $F_1 = \phi$. Since if $F_1 \neq \phi$, all the variables y_k in F_1 can be complemented by 1-y_k which amounts replacing the columns G_k and the right-hand side with $-G_k$ and $b - G_k$, respectively. The reduced LP constraint set after removing the fixed binary variables becomes

$$\overline{A}^F x + \overline{G}^F y^F \le \overline{b}^F \equiv \begin{cases} Ax + \sum_{i \in F} G_j y_j \le b \\ - \sum_{i \in F_1} G_j, \\ \nabla g^x\left(\overline{x}, \overline{y}\right) x \\ + \sum_{i \in F} \nabla g_i^y\left(\overline{x}, \overline{y}\right) y_j \\ \le \nabla g\left(\overline{x}, \overline{y}\right) \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} \\ -g\left(\overline{x}, \overline{y}\right) \\ - \sum_{i \in F_1} \nabla g_i^y\left(\overline{x}, \overline{y}\right), \\ -y_i \le 0 \quad i \in F, \\ y_i \le 1 \quad i \in F, \end{cases}$$

where $\overline{A}^F \in \Re^{(m+2|F|+l)\times n}$, $\overline{G}^F \in \Re^{(m+2|F|+l)\times|F|}$, $\overline{b}^F \in \Re^{m+2|F|+l}$. Then, the feasible region of the above



Mixed-Integer Nonlinear Optimization: A Disjunctive Cutting Plane Approach, Figure 1
Linear approximation of the mixed-integer nonlinear optimization problem, where the mixed-integer convex set is described by a continuous variable and a binary variable

LP can be reformulated as

$$K = \left\{ \left(x, y^F\right) \in \Re^{n+|F|} \,\middle|\, \overline{A}^F x + \overline{G}^F y^F \le \overline{b}^F \right\}$$

Let $\left(\overline{x}, \overline{y}\right)$ be the optimal solution when solving $NLP\left(C, \overline{F}_0, \overline{F}_1\right)$. First, we assume that $\left(\overline{x}, \overline{y}\right)$ is not feasible to problem (P), and let j be the binary variable index such that $0 < \overline{y}_j < 1$. In [11], a disjunctive cut generation linear programming is given by

$$(LP(F)) \left| \begin{array}{l} \min \sum_{i \in N} z_i + \sum_{i \in F} w_i \\ s.t.\ x = u_0 + u_1,\ y^F = v_0 + v_1, \\ \quad \overline{A}^F u_0 + \overline{G}^F v_0 - \overline{b}^F \lambda_0 \le 0, \\ \quad v_{0,j} \le 0, \\ \quad \overline{A}^F u_1 + \overline{G}^F v_1 - \overline{b}^F \lambda_1 \le 0, \\ \quad -v_{1,j} \le -\lambda_1, \\ \quad \lambda_0 + \lambda_1 = 1; \\ \quad -z + x \le \overline{x}, \quad -z - x \le -\overline{x}, \\ \quad -w + y^F \le \overline{y}^F, \quad -w - y^F \le -\overline{y}^F, \\ \quad \lambda_0, \lambda_1 \ge 0, \quad x, u_0, u_1, z \in \Re^n, \\ \quad y^F, v_0, v_1, w \in \Re^{|F|}. \end{array} \right.$$

This linear program has $4n + 4|F| + 2$ variables with $2m + 3n + 7|F| + 2l + 3$ equality or inequality constraints. After solving this linear program, we get its solutions denoted by $\left(\tilde{x}, \tilde{y}^F, \tilde{u}_0, \tilde{v}_0, \tilde{u}_1, \tilde{v}_1, \tilde{z}, \tilde{w}, \tilde{\lambda}_0, \tilde{\lambda}_1\right)$ as well as the dual multipliers. Denote by $\pi_x^F, \pi_y^F, \delta_\lambda^F$ the multipliers for the equality constraints, δ_0^F, δ_1^F for the disjunctive inequality constraints, μ_0^F, μ_1^F for the inequality original constraints, and $\varepsilon_+^F, \varepsilon_-^F, \varphi_+^F, \varphi_-^F$ for the additional constraints in LP(F), the cut generated in Theorem 2, i. e. $\alpha x + \beta^F y^F \le \gamma$, can be reformulated by the dual multipliers and the primal solutions to LP(F), as $\pi_x^F x + \pi_y^F y^F \le \pi_x^F \tilde{x} + \pi_y^F \tilde{y}^F$. Since the feasible set of problem (P) at the node (F_0, F_1) is contained in the feasible set of the MILP at that node. Therefore, the inequality $\alpha x + \beta^F y^F \le \gamma$ is valid and proper for problem (P) at the current node denoted by (F_0, F_1), and its descendents where the variables in (F_0, F_1) remain fixed.

Cut Lifting

An important advantage of the cut generated by the lift-and-project technology is that the multipliers, i. e., $\mu_0^F, \delta_0^F, \mu_1^F, \delta_1^F$, obtained along with the solution $\left(\tilde{x}, \tilde{y}^F\right)$ by solving LP(F) can be used to calculate the closed form expressions of the coefficients β_i for the binary variables in the index set $F_0 \cup F_1$. First, we lift the inequality obtained at the current node into the complemented original space of the MILP problem, that is

$$\left\{ \left(x, y\right) \in \Re^{n+q} : \left| \begin{array}{l} \overline{A}x + \overline{G}^Q y \le \overline{b}^Q, \\ y \in \{0,1\}^q \end{array} \right. \right\}$$

Let $j \in \{1, \ldots, q\}$ be an index such that $0 < \overline{y}_j < 1$ and consider the inequality $\alpha^q x + \beta^q y \le \gamma^q$ generated over the complemented original space of the MILP

problem, this is to solve the linear program LP(Q), as

$$(LP(Q)) \left| \begin{array}{l} \min \sum_{i\in N} z_i + \sum_{i\in Q} w_i \\ s.t.\ x = u_0 + u_1, \quad y = v_0 + v_1, \\ \quad \overline{A}u_0 + \overline{G}^Q v_0 - \overline{b}^Q \lambda_0 \le 0, \\ \quad v_{0,j} \le 0, \\ \quad \overline{A}u_1 + \overline{G}^Q v_1 - \overline{b}^Q \lambda_1 \le 0, \\ \quad -v_{1,j} \le -\lambda_1, \\ \quad \lambda_0 + \lambda_1 = 1, \\ \quad -z + x \le -\overline{x}, \quad -z - x \le -\overline{x}, \\ \quad -w + y \le \overline{y}, \quad -w - y \le -\overline{y}, \\ \quad \lambda_0, \lambda_1 \ge 0, \\ \quad x, u_0, u_1, z \in \Re^n, y, v_0, v_1, w \in \Re^q . \end{array} \right.$$

Note that more variables and constraints are added into this linear program compared with LP(F). But, by using the solutions to LP(F) and its multipliers, we can obtain the optimal solution to the above LP(Q). Let $\left(\hat{x}, \hat{y}, \hat{u}_0, \hat{v}_0, \hat{u}_1, \hat{v}_1, \hat{z}, \hat{w}, \hat{\lambda}_0, \hat{\lambda}_1\right)$ be the solution to the linear program LP(Q), which can be constructed by the solution to the LP(F), as $\tilde{x} = \hat{x}$, $\hat{y} = (\tilde{y}^F, 0)$, $\hat{u}_0 = \tilde{u}_0$, $\hat{u}_1 = \tilde{u}_1$, $\hat{v}_0 = (\tilde{v}_0, 0)$, $\hat{v}_1 = (\tilde{v}_1, 0)$, $\hat{\lambda}_0 = \tilde{\lambda}_0$, $\hat{\lambda}_1 = \tilde{\lambda}_1$, $\hat{z} = \tilde{z}$, and $\hat{w} = (\tilde{w}, 0)$. Then, the corresponding dual multipliers of LP(Q) denoted by $(\hat{\pi}_x^Q, \hat{\pi}_y^Q, \hat{\mu}_0^Q, \hat{\delta}_0^Q, \hat{\mu}_1^Q, \hat{\delta}_1^Q, \hat{\delta}_\lambda^Q, \hat{\varepsilon}_+^Q, \hat{\varepsilon}_-^Q, \hat{\varphi}_+^Q, \hat{\varphi}_-^Q)$, which is also the solution to the dual linear program DLP(Q), can be constructed by those to DLP(F), as $\hat{\pi}_x^Q = \tilde{\pi}_x^F$, $\hat{\pi}_{y,i}^Q = \tilde{\pi}_{y,i}^F$ for $i \in F$, $\hat{\pi}_{y,i}^Q = \min\left\{\tilde{\mu}_0^F \overline{G}_i^Q, \tilde{\mu}_1^F \overline{G}_i^Q\right\}$ for $i \in F_0 \cup F_1$, $\hat{\delta}_0^Q = \tilde{\delta}_0^F$, $\hat{\delta}_1^Q = \tilde{\delta}_1^F$, $\hat{\delta}_\lambda^Q = \tilde{\delta}_\lambda^F$, $\hat{\mu}_0^Q = (\tilde{\mu}_0^F, 0)$, $\hat{\mu}_1^Q = (\tilde{\mu}_1^F, 0)$, $\hat{\varepsilon}_+^Q = \tilde{\varepsilon}_+^Q$, $\hat{\varepsilon}_-^Q = \tilde{\varepsilon}_-^Q$, $\hat{\varphi}_+^Q = (\tilde{\varphi}_+^F, 0)$, and $\hat{\varphi}_-^Q = \left(\tilde{\varphi}_-^F, 0\right)$. The inequality $\alpha^q x + \beta^q y \le \gamma^q$ described by $\hat{\pi}_x^Q x + \hat{\pi}_y^Q y \le \hat{\pi}_x^Q \hat{x} + \hat{\pi}_y^Q \hat{y}$ is valid for the entire enumeration tree, and cuts away $(\overline{x}, \overline{y})$.

Conclusion

A branch-and-cut algorithm is introduced in this section to solve 0–1 mixed-integer nonlinear optimization problem where the disjunctive cuts are generated and incorporated into an enumeration process. The lift-and-project cut generation is performed via linear programming, as opposed to the convex nonlinear approach used in [9]. This new approach has the advantage of making the cut generation computationally cheaper and overcoming the nondifferential problems.

References

1. Balas E, Ceria S, Cornuejols G (1993) A lift-and-project cutting plane algorithm for mixed-integer 0–1 programs. Math Program 58:295–324
2. Balas E, Ceria S Cornuejols G (1996) Mixed 0–1 programming by lift-and-project in a branch-and-cut framework. Manag Sci 42:1229–1246
3. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
4. Floudas CA (1995) Nonlinear and mixed-integer optimization, fundamentals and applications. Oxford Univ Press, New York
5. Floudas CA (2005) Research challenges, opportunities and synergism in systems engineering and computational biology. AIChE J 51:1872–1884
6. Grossman IE, Westerberg AW (2000) Research challenges in process systems engineering. AIChE J 46:1700–1703
7. Lovasz L, Schrijver A (1991) Cones of matrices and set functions and 0–1 optimization. SIAM J Optim 1:166–190
8. Sherali H, Adams W (1990) A hierarchy of relaxations between the continuous and convex hull representation for zero-one programming problems. SIAM J Discret Math 3:411–430
9. Stubbs RA, Mehrotra S (1999) A branch-and-cut method for 0–1 mixed convex programming. Math Program 86:515–532
10. Zhu Y, Kuno T (2003) Global optimization of nonconvex MINLP by a hybrid branch-and-bound and revised general Benders decomposition method. Ind Eng Chem Res 42:528–539
11. Zhu Y, Kuno T (2006) A disjunctive cutting plane based branch-and-cut algorithm for 0–1 mixed-integer nonlinear programs. Ind Eng Chem Res 45:187–196

Mixed Integer Nonlinear Programming

MINLP

CHRISTODOULOS A. FLOUDAS
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C11, 49M37

Article Outline

Keywords
Mathematical Description
Challenges in MINLP

Overview of Local Optimization Approaches for Convex MINLP Models
Overview of Global Optimization Approaches for Nonconvex MINLP Models
Branch and Reduce Algorithm
Optimality Based Range Reduction Tests
Feasibility Based Range Reduction Tests
Interval Analysis Based Approach
Node Fathoming Tests
Branching Strategies
Extended Cutting Plane for Pseudoconvex MINLPs
Reformulation/Spatial Branch and Bound Algorithm
Hybrid Branch and Bound and Outer Approximation
The SMIN-αBB Algorithm
The GMIN-αBB Algorithm
See also
References

Keywords

Decomposition; Outer approximation; Branch and bound; Global optimization

A wide range of nonlinear optimization problems involve integer or discrete variables in addition to the continuous variables. These classes of optimization problems arise from a variety of applications and are denoted as *mixed integer nonlinear programming* MINLP problems.

The integer variables can be used to model, for instance, sequences of events, alternative candidates, existence or non-existence of units (in their zero-one representation), while discrete variables can model, for instance, different equipment sizes. The continuous variables are used to model the input-output and interaction relationships among individual units/operations and different interconnected systems.

The nonlinear nature of these mixed integer optimization problems may arise from:

i) nonlinear relations in the integer domain exclusively (e. g., products of binary variables in the quadratic assignment model);
ii) nonlinear relations in the continuous domain only (e. g., complex nonlinear input-output model in a distillation column or reactor unit);
iii) nonlinear relations in the joint integer-continuous domain (e. g., products of continuous and binary variables in the scheduling/planning of batch processes and retrofit of heat recovery systems).

The book [88] studies mixed integer linear optimization and combinatorial optimization, while the [40] studies mixed integer nonlinear optimization problems.

The coupling of the integer domain and the continuous domain with their associated nonlinearities make the class of MINLP problems very challenging from the theoretical, algorithmic, and computational point of view. Mixed integer nonlinear optimization problems are encountered in a variety of applications in all branches of engineering and applied science, applied mathematics, and operations research. These represent very important and active research areas that include:

- *process synthesis*
  - heat exchanger networks
  - retrofit of heat recovery systems
  - distillation sequencing
  - mass exchange networks
  - reactor-based systems
  - reactor-separator-recycle systems
  - utility systems
  - total process systems
  - metabolic engineering
- *process design*
  - reactive distillation
  - design of dynamic systems
  - plant layout
  - environmental design
- process synthesis and *design under uncertainty*
  - uncertainty analysis
  - dynamic systems
  - batch plant design
- *molecular design*
  - solvent selection
  - design of polymers and refrigerants
  - property prediction under uncertainty
- *interaction of design, synthesis and control*
  - steady state operation
  - dynamic operation
- *process operations*
  - scheduling of multiproduct plants
  - design and retrofit of multiproduct plants
  - synthesis, design and scheduling of multipurpose plants

  - planning under uncertainty
- *facility location and allocation*
- *facility planning and scheduling*
- *topology of transportation networks*

The applications in the area of *process synthesis* in chemical engineering include:

i) the synthesis of grassroot heat recovery networks [24,25,43,138,139,140];
ii) the retrofit of heat exchanger systems [25,95];
iii) the synthesis of distillation-based separation systems [8,9,90,102,104,131];
iv) the synthesis of mass exchange networks [54,99];
v) the synthesis of complex reactor networks [71,73,74,119];
vi) the synthesis of reactor-separator-recycle systems [72];
vii) the synthesis of utility systems [65];
viii) the synthesis of total process systems [28,29,68,69,75,76,98]; and
ix) the analysis and synthesis of metabolic pathways [30,58,59,107].

Reviews of the mixed integer nonlinear optimization frameworks and applications in Process Synthesis are provided in [40,49,50], and [7], while algorithmic advances for logic and global optimization in Process Synthesis are reviewed in [44].

The MINLP applications in the area of *process design* include:

i) reactive distillation processes [26];
ii) design of dynamic systems [11,14,117,118];
iii) plant layout systems [47,105]; and
iv) environmentally benign systems [27,123].

The MINLP applications in the area of *process synthesis and design under uncertainty* include:

i) deterministic and stochastic uncertainty analysis [1,33,51];
ii) design of dynamic systems under uncertainty [31,85]; and
iii) design of batch processes under uncertainty [57,63,108,109].

In the area of *molecular design*, the MINLP applications include:

i) the computer-aided molecular design aspects of selecting the best solvents [91];
ii) design of polymers and refrigerants [21,22,23,35,80,111,126]; and
iii) property prediction under uncertainty [81].

The MINLP applications in the area of *interaction of design, synthesis and control* include:

i) studies under steady state operation of chemical processes [78,79,96,97]; and
ii) studies under dynamic operation [85,86,118].

Applications of MINLP approaches have also emerged in the area of *process operations* and include:

i) short term scheduling of batch and semicontinuous processes [85,143];
ii) the design of multiproduct plants [17,18,53];
iii) the synthesis, design and scheduling of multipurpose plants [13,36,37,93,94,116,127,128,132,133,137]; and
iv) planning under uncertainty [62,63,64,77,106].

Reviews of the advances in the design, scheduling and planning of batch plants can be found in [52,113], while a collection of recent contributions can be found in the proceedings of the 1998 FOCAPO meeting.

MINLP applications received significant attention in other engineering disciplines. These include

i) the facility location in a multi-attribute space [45];
ii) the optimal unit allocation in an electric power system [16];
iii) the facility planning of an electric power generation [19,114];
iv) the chip layout and compaction [32];
v) the topology optimization of transportation networks [60]; and
vi) the optimal scheduling of thermal generating units [48].

Mathematical Description

The general algebraic MINLP formulation can be stated as:

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \subseteq \mathbb{R}^n \\ & \mathbf{y} \in \mathbf{Y} \quad \text{integer}. \end{cases} \tag{1}$$

Here $\mathbf{x}$ represents a vector of n continuous variables (e. g., flows, pressures, compositions, temperatures, sizes of units), and $\mathbf{y}$ is a vector of integer variables (e. g., alternative solvents or materials); $\mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ denote the m equality constraints (e. g., mass, energy

balances, equilibrium relationships); $\mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}$ are the p inequality constraints (e. g., specifications on purity of distillation products, environmental regulations, feasibility constraints in heat recovery systems, logical constraints); and $f(\mathbf{x}, \mathbf{y})$ is the objective function (e. g., annualized total cost, profit, thermodynamic criteria).

Remark 1 The integer variables $\mathbf{y}$ with given lower and upper bounds

$$\mathbf{y}^L \leq \mathbf{y} \leq \mathbf{y}^U$$

can be expressed through 0–1 variables (i. e., binary), denoted as $\mathbf{z}$, by the following formula:

$$y = y^L + z_1 + 2z_2 + 4z_3 + \cdots + 2^{N-1} z_N,$$

where N is the minimum number of 0–1 variables needed. This minimum number is given by:

$$N = 1 + \text{INT}\left\{\frac{\log\left(y^U - y^L\right)}{\log 2}\right\},$$

where the INT function truncates its real argument to an integer value.

Then, formulation (1) can be written in terms of 0–1 variables:

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \subseteq \mathbb{R}^n \\ & \mathbf{y} \in \{0, 1\}^q, \end{cases} \tag{2}$$

where $\mathbf{y}$ now is a vector of q 0–1 variables (e. g., existence of a process unit ($y_i = 1$) or nonexistence ($y_i = 0$)).

Challenges in MINLP

Dealing with *mixed integer nonlinear optimization* models of the form (1) or (2) present two major challenges. These difficulties are associated with the nature of the problem, namely, the combinatorial domain ($\mathbf{y}$-domain) and the continuous domain ($\mathbf{x}$-domain).

As the number of binary variables $\mathbf{y}$ in (2) increase, one is faced with a large combinatorial problem, and the complexity analysis results characterize MINLP problems as NP-complete [88]. At the same time, due to the nonlinearities the MINLP problems are in general nonconvex which implies the potential existence of multiple local solutions. The determination of a global solution of the *nonconvex MINLP* problems is also NP-hard, since even the global optimization of constrained nonlinear programming problems can be *NP*-hard [100], and even quadratic problems with one negative eigenvalue are *NP*-hard [101]. An excellent book on complexity issues for nonlinear optimization is [129].

Despite the aforementioned discouraging results from complexity analysis, which are worst-case results, significant progress has been achieved in the MINLP area from the theoretical, algorithmic, and computational perspective. As a result, several algorithms have been proposed for convex and nonconvex MINLP models, their convergence properties have been investigated, and a large number of applications now exist that cross the boundaries of several disciplines. In the sequel, we will discuss these developments.

Overview of Local Optimization Approaches for Convex MINLP Models

A representative collection of *local MINLP* algorithms developed for solving *convex MINLP* models of the form (1) or restricted classes of (2) includes the following:

1) *generalized Benders decomposition*, GBD, [42,46,103];
2) *outer approximation*, OA, [34];
3) outer approximation with equality relaxation, OA/ER, [67];
4) outer approximation with equality relaxation and augmented penalty, OA/ER/AP, [131];
5) *generalized outer approximation*, GOA, [38];
6) *generalized cross decomposition*, GCD, [61];
7) *branch and bound*, BB, [15,20,39,55,92,110];
8) *feasibility approach*, FA, [82];
9) *extended cutting plane*, ECP, [134,135];
10) *logic-based approaches*, [124,130].

In the pioneering work [46] on the generalized benders decomposition, GBD, two sequences of updated upper(nonincreasing) and lower (nondecreasing) bounds are created that converge within ϵ in a finite number of iterations. The upper bounds correspond to solving subproblems in the $\mathbf{x}$ variables by fixing the $\mathbf{y}$ variables, while the lower bounds are based on duality theory.

The outer approximation, OA, addresses problems with nonlinear inequalities, and creates sequences of upper and lower bounds as the GBD, but it has the distinct feature of using primal information, that is the solution of the upper bound problems, so as to linearize the objective and constraints around that point. The lower bounds in OA are based upon the accumulation of the linearized objective function and constraints, around the generated primal solution points.

The OA/ER algorithm extends the OA to handle nonlinear equality constraints by relaxing them into inequalities according to the sign of their associated multipliers.

The OA/ER/AP algorithm introduces an augmented penalty function in the lower bound subproblems of the OA/ER approach.

The generalized outer approximation, GOA, extends the OA to the MINLP problems that the GBD addresses and introduces exact penalty functions.

The generalized cross decomposition, GCD, simultaneously utilizes primal and dual information by exploiting the advantages of Dantzig–Wolfe and generalized Benders decomposition.

An overview of these local MINLP algorithms and extensive theoretical, algorithmic, and applications of GBD, OA, OA/ER, OA/ER/AP, GOA, and GCD algorithms can be found in [40].

The branch and bound, BB, approaches start by solving the continuous relaxation of the MINLP and subsequently perform an implicit enumeration where a subset of the 0–1 variables is fixed at each node. The lower bound corresponds to the NLP solution at each node and it is used to expand on the node with the lowest lower bound or it is used to eliminate nodes if the lower bound exceeds the current upper bound. If the continuous relaxation, NLP in most cases with the exception of the algorithm of [110] where an LP problem is obtained, of the MINLP has a 0–1 solution for the **y** variables, then the BB algorithm will terminate at that node. With a similar argument, if a tight NLP relaxation results in the first node of the tree, then the number of nodes that would need to be eliminated can be low. However, loose NLP relaxations may result in having a large number of NLP subproblems to be solved. The algorithm terminates when the lowest lower bound is within a prespecified tolerance of the best upper bound.

The feasibility approach, FA, rounds the relaxed NLP solution to an integer solution with the least local degradation by successively forcing the superbasic variables to become nonbasic based on the reduced cost information. The premise of this approach is that the problems to be treated are sufficiently large so that techniques requiring the solution of several NLP relaxations, such as the branch and bound approach, have prohibitively large costs. They therefore wish to account for the presence of the integer variables in the formulation and solve the mixed integer problem directly. This is achieved by fixing most of the integer variables to one of their bounds (the *nonbasic variables*) and allowing the remaining small subset (the *basic variables*) to take discrete values in order to identify feasible solutions. After each iteration, the reduced costs of the variables in the nonbasic set are computed to measure their effect on the objective function. If a change causes the objective function to decrease, the appropriate variables are removed from the nonbasic set and allowed to vary for the next iteration. When no more improvement in the objective function is possible, the algorithm is terminated. This strategy leads to the identification of a local solution.

The *cutting plane* algorithm proposed in [66] for NLP problems has been extended to MINLPs [134,135]. The ECP algorithm relies on the linearization of one of the nonlinear constraints at each iteration and the solution of the increasingly tight MILP made up of these linearizations. The solution of the MILP problem provides a new point on which to base the choice of the constraint to be linearized for the next iteration of the algorithm. The ECP does not require the solution of any NLP problems for the generation of an upper bound. As a result, a large number of linearizations are required for the approximation of highly nonlinear problems and the algorithm does not perform well in such cases. Due to the use of linearizations, convergence to the global optimum solution is guaranteed only for problems involving inequality constraints which are convex in the **x** and relaxed **y**-space.

An alternative to the direct solution of the MINLP problem was proposed by [124]. Their approach stems from the work of [70] on a modeling/decomposition strategy which avoids the zero-flows generated by the nonexistence of a unit in a process network. The first stage of the algorithm is the reformulation of the

MINLP into a generalized disjunctive program. A vector of Boolean variables indicate the status of a disjunction (True or False) and are associated with the alternatives. The set of disjunctions allows the representation of several alternatives. A set of logical relationships between the Boolean variables is introduced. Instead of resorting to binary variables within a single model, the disjunctions are used to generate a different model for each alternative. Since all continuous variables associated with the nonexisting alternatives are set to zero, this representation helps to reduce the size of the problems to be solved. Two algorithms are suggested by [124]. They are logic-based variants of the outer approximation and generalized Benders decomposition. [130] introduced LOGMIP, a computer code for disjunctive programming and MINLP problems, and studied modeling alternatives and process synthesis applications.

Overview of Global Optimization Approaches for Nonconvex MINLP Models

the previous Section we discussed local MINLP algorithms which are applicable to convex MINLP models. While identification of the global solution for convex problems can be guaranteed, a local solution is often obtained for nonconvex problems. The recent book by [41] discusses the theoretical, algorithmic and applications oriented advances in the global optimization of mixed integer nonlinear models. A number of *global MINLP* algorithms that have been developed to address different types of nonconvex MINLPs are presented in this section. These include:

1) *Branch and reduce* approach, [115];
2) *interval analysis* based approach, [125];
3) *extended cutting plane* approach, [135,136];
4) *reformulation/spatial branch and bound* approach, [121,122];
5) *hybrid branch and bound and outer approximation* approach, [141,142];
6) The *SMIN-αBB* approach, [2,4];
7) The *GMIN-αBB* approach, [2,4].

In the sequel, we will briefly discuss the approaches 1)–7).

Branch and Reduce Algorithm

[115] extended the scope of branch and bound algorithms to problems for which valid convex underestimating NLPs can be constructed for the nonconvex relaxations. The range of application of the proposed algorithm encompasses bilinear problems and separable problems involving functions for which convex underestimators can be built [10,83]. Because the nonconvex NLPs must be underestimated at each node, convergence can only be achieved if the continuous variables are branched on. A number of tests are suggested to accelerate the reduction of the solution space. They are summarized in the following.

Optimality Based Range Reduction Tests

For the first set of tests, an upper bound U on the nonconvex MINLP must be computed and a convex lower bounding NLP must be solved to obtain a lower bound L. If a bound constraint for variable x_i, with $x_i^L \leq x_i \leq x_i^U$, is active at the solution of the convex NLP and has multiplier $\lambda_i^* > 0$, the bounds on x_i can be updated as follows:

1) If $x_i - x_i^U = 0$ at the solution of the convex NLP and $\kappa_i = x_i^U - (U - L)/\lambda_i^*$ is such that $\kappa_i > x_i^L$, then $x_i^L = \kappa_i$.
2) If $x_i - x_i^L = 0$ at the solution of the convex NLP and $\kappa_i = x_i^L + (U - L)/\lambda_i^*$ is such that $\kappa_i < x_i^U$, then $x_i^U = \kappa_i$.

If neither bound constraint is active at the solution of the convex NLP for some variable x_j, the problem can be solved by setting $x_j = x_j^U$ or $x_j = x_j^L$. Tests similar to those presented above are then used to update the bounds on x_j.

Feasibility Based Range Reduction Tests

In addition to ensuring that tight bounds are available for the variables, the underestimators of the constraints are used to generate new constraints for the problem. Consider the constraint $g_i(\mathbf{x}, \mathbf{y}) \leq 0$. If its underestimating function $\underline{g}_i(\mathbf{x}, \mathbf{y}) = 0$ at the solution of the convex NLP and its multiplier is $\mu_i^* > 0$, the constraint

$$\underline{g}_i(\mathbf{x}, \mathbf{y}) \geq -\frac{U - L}{\mu_i^*}$$

can be included in subsequent problems.

The branch and reduce algorithm has been tested on a set of small problems.

Interval Analysis Based Approach

An approach based on interval analysis was proposed by [125] to solve to global optimality problems with a twice-differentiable objective function and once-differentiable constraints. Interval arithmetic allows the computation of guaranteed ranges for these functions [87,89,112]. The approach relies on the same concepts of successive partitioning of the domain and bounding of the objective function, while the branching takes place on the discrete and continuous variables. The main difference with the branch and bound algorithms is that bounds on the problem solution in a given domain are not obtained through optimization. Instead, they are based on the range of the objective function in the domain under consideration, as computed with interval arithmetic. As a consequence, these bounds may be quite loose and efficient fathoming techniques are required in order to enhance convergence. [125] suggested node fathoming tests and branching strategies which are outlined in the sequel. Convergence is declared when best upper and lower bounds are within a prespecified tolerance and when the width of the corresponding region is below a prespecified tolerance.

Node Fathoming Tests

The *upper-bound test* is a classical criterion used in all branch and bound schemes: If the lower bound for a node is greater than the best upper bound for the MINLP, the node can be fathomed.

The *infeasibility test* is also used by all branch and bound algorithms. However, the identification of infeasibility using interval arithmetic differs from its identification using optimization schemes. An inequality constraint $g_i(\mathbf{x}, \mathbf{y}) \leq 0$ is declared infeasible if its interval inclusion over the current domain, is positive. If a constraint is found to be infeasible, the current node is fathomed.

The *monotonicity test* is used in interval-based approaches. If a region is feasible, the monotonicity properties of the objective function can be tested. For this purpose, the inclusions of the gradients of the objective with respect to each variable are evaluated. If all the gradients have a constant sign for the current region, the objective function is monotonic and only one point needs to be retained from the current node.

The *nonconvexity test* is used to test the existence of a solution (local or global) within a region. If such a point exists, the Hessian matrix of the objective function at this point must be positive semidefinite. A sufficient condition is the nonnegativity of at least one of the diagonal elements of its interval Hessian matrix.

[125] suggested two additional tests to accelerate the fathoming process. The first is denoted as *lower bound test*. It requires the computation of a valid lower bound on the objective function through a method other than interval arithmetic. If the upper bound at a node is less than this lower bound, the region can be eliminated. The second test, the *distrust region method*, aims to help the algorithm identify infeasible regions so that they can be removed from consideration. Based on the knowledge of an infeasible point, interval arithmetic is used to identify an infeasible hypercube centered on that point.

Branching Strategies

The variable with the widest range is selected for branching. It can be a continuous or a discrete variable. In order to determine where to split the chosen variable, a relaxation of the MINLP is solved locally.

- Continuous Branching Variable: If the optimal value of the continuous branching variable, x^*, is equal to one of the variable bounds, branch at the midpoint of the interval. Otherwise, branch at $x^* - \beta$, where β is a very small scalar.
- Discrete Branching Variable: If the optimal value of the discrete branching variable, y^*, is equal to the upper bound on the variable, define a region with $y = y^*$ and one with $y^L \leq y \leq y^* - 1$, where y^L is the lower bound on y. Otherwise, create two regions $y^L \leq y \leq \text{int}(y^*)$ and $\text{int}(y^*) + 1 \leq y \leq y^U$, where y^U is the upper bound on y.

This algorithm has been tested on a small example problem and a molecular design problem [125].

Extended Cutting Plane for Pseudoconvex MINLPs

The use of the ECP algorithm for nonconvex MINLP problems was suggested in [135], using a modified algorithmic procedure as described in [136]. The main changes occur in the generation of new constraints for the MILP at each iteration (Step 4). In addition to the

construction of the linear function $l_k(\mathbf{x}, \mathbf{y})$ at iteration k, the following steps are taken:

1) Remove all constraints for which $l_i(\mathbf{x}^k, \mathbf{y}^k) > g_{ji}(\mathbf{x}^k, \mathbf{y}^k)$. These correspond to linearizations which did not underestimate the corresponding nonlinear constraint at all points due to the presence of nonconvexities.
2) Replace all constraints for which $l_i(\mathbf{x}^k, \mathbf{y}^k) = g_{ji}(\mathbf{x}^k, \mathbf{y}^k) = 0$ by their linearization around $(\mathbf{x}^k, \mathbf{y}^k)$.
3) If constraint i is such that $g_i(\mathbf{x}^k, \mathbf{y}^k) > 0$, add its linearization around $(\mathbf{x}^k, \mathbf{y}^k)$.

The convergence criterion is also modified. In addition to the test used in Step 3, the following two conditions must be met:

1) $(\mathbf{x}^k - \mathbf{x}^{k-1})^{\mathsf{T}}(\mathbf{x}^k - \mathbf{x}^{k-1}) \leq \delta$, a pre-specified tolerance.
2) $\mathbf{y}^k - \mathbf{y}^{k-1} = 0$.

The ECP algorithm for pseudoconvex MINLPs has been used to address a trim loss problem arising in the paper industry [136]. A comparative study between the outer approximation, the generalized Benders decomposition and the extended cutting plane algorithm for convex MINLPs was presented in [120].

Reformulation/Spatial Branch and Bound Algorithm

A global optimization algorithm of the branch and bound type was proposed in [121]. It can be applied to problems in which the objective and constraints are functions involving any combination of binary arithmetic operations (addition, subtraction, multiplication and division) and functions that are either concave over the entire solution space (such as ln) or convex over this domain (such as exp).

The algorithm starts with an automatic reformulation of the original nonlinear problem into a problem that involves only linear, bilinear, linear fractional, simple exponentiation, univariate concave and univariate convex terms. This is achieved through the introduction of new constraints and variables. The reformulated problem is then solved to global optimality using a branch and bound approach. Its special structure allows the construction of a convex relaxation at each node of the tree. It should be noted that due to the introduction of many new constraints and variables the size of the convex relaxation of the reformulated problem increases substantially even for modest size problems. The integer variables can be handled in two ways during the generation of the convex lower bounding problem. The integrality condition on the variables can be relaxed to yield a convex NLP which can then be solved globally. Alternatively, the integer variables can be treated directly and the convex lower bounding MINLP can be solved using a branch and bound algorithm. This second approach is more computationally intensive but is likely to result in tighter lower bounds on the global optimum solution.

In order to obtain an upper bound on the optimum solution, a local MINLP algorithm can be used. Alternatively, the MINLP can be transformed to an equivalent nonconvex NLP by relaxing the integer variables. For example, a variable $y \in \{0, 1\}$ can be replaced by a continuous variable $z \in [0, 1]$ by including the constraint $z - z \cdot z = 0$.

This algorithm has been applied to reactor selection, distillation column design, nuclear waste blending, heat exchanger network design and multilevel pump configuration problems.

Hybrid Branch and Bound and Outer Approximation

[142] proposed a global optimization MINLP approach for the synthesis of heat exchanger networks without stream splitting. This approach is a hybrid branch and bound with outer approximation. It is based on two alternative convex underestimators for the heat transfer area. The first type of these convex underestimators along with the variable bounds and techniques for the bound contraction are based on a thermodynamic analysis. The second type is based on a relaxation and transformation so as to employ specific underestimation schemes. These convex underestimators result in a convex MINLP that is solved using the Outer Approximation approach and which provides valid lower bounds on the global solution. This approach has been applied to five heat exchanger network examples that employ the MINLP model of [138] that contains linear constraints and nonconvex objective function.

[141] introduced a deterministic branch and contract approach for structured process systems that have univariate concave, bilinear and linear fractional terms.

They proposed properties of the contraction operation and studied their effect on several applications.

The SMIN-αBB Algorithm

The SMIN-αBB global optimization algorithm, proposed by [2] is designed to solve to global optimality mathematical models where the binary/integer variables appear linearly and hence separably from the continuous variables and/or appear in at most bilinear terms, while nonlinear terms in the continuous variables appear separably from the binary/integer variables. These mathematical models become:

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x}) + \mathbf{x}^\top A_0 \mathbf{y} + \mathbf{c}_0^\top \mathbf{y} \\ \text{s.t.} & \mathbf{h}(\mathbf{x}) + \mathbf{x}^\top A_1 \mathbf{y} + \mathbf{c}_1^\top \mathbf{y} = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}) + \mathbf{x}^\top A_2 \mathbf{y} + \mathbf{c}_2^\top \mathbf{y} \le \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \subseteq \mathbb{R}^n \\ & \mathbf{y} \in \mathbf{Y} \quad \text{integer}, \end{cases} \tag{3}$$

where $\mathbf{c}_0^\top$, $\mathbf{c}_1^\top$ and $\mathbf{c}_2^\top$ are constant vectors, A_0, A_1 and A_2 are constant matrices and $f(\mathbf{x})$, $\mathbf{h}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ are functions with continuous second order derivatives.

The theoretical, algorithmic and computational studies of the SMIN-αBB algorithm are presented in detail in [41].

The GMIN-αBB Algorithm

The GMIN-αBB global optimization algorithm proposed in [2] operates within a branch and bound framework. The main difference with the algorithms of [56,92] and [20] is its ability to identify the global optimum solution of a much larger class of problems of the form

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \subseteq \mathbb{R}^n \\ & \mathbf{y} \in \mathbb{N}^q, \end{cases}$$

where $\mathbf{N}$ is the set of nonnegative integers and the only condition imposed on the functions $f(\mathbf{x},\mathbf{y})$, $\mathbf{g}(\mathbf{x},\mathbf{y})$ and $\mathbf{h}(\mathbf{x},\mathbf{y})$ is that their continuous relaxations possess continuous second order derivatives. This increased applicability results from the use of the αBB global optimization algorithm for continuous twice-differentiable NLPs [3,5,6,12].

The theoretical, algorithmic and computational studies of the GMIN-αBB Algorithm are presented in detail in [41].

See also

- Chemical Process Planning
- Complexity Classes in Optimization
- Complexity of Degeneracy
- Complexity of Gradients, Jacobians, and Hessians
- Complexity Theory
- Complexity Theory: Quadratic Programming
- Computational Complexity Theory
- Continuous Global Optimization: Applications
- Extended Cutting Plane Algorithm
- Fractional Combinatorial Optimization
- Generalized Benders Decomposition
- Generalized Outer Approximation
- Global Optimization in the Analysis and Management of Environmental Systems
- Information-based Complexity and Information-based Optimization
- Interval Global Optimization
- Kolmogorov Complexity
- MINLP: Application in Facility Location-allocation
- MINLP: Applications in Blending and Pooling Problems
- MINLP: Applications in the Interaction of Design and Control
- MINLP: Branch and Bound Global Optimization Algorithm
- MINLP: Branch and Bound Methods
- MINLP: Design and Scheduling of Batch Processes
- MINLP: Generalized Cross Decomposition
- MINLP: Global Optimization with αBB
- MINLP: Heat Exchanger Network Synthesis
- MINLP: Logic-based Methods
- MINLP: Outer Approximation Algorithm
- MINLP: Reactive Distillation Column Synthesis
- MINLP: Trim-loss Problem
- Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- Parallel Computing: Complexity Classes

References

1. Acevedo J, Pistikopoulos EN (1996) A parametric MINLP algorithm for process synthesis problems under uncertainty. IÈC Res 35(1):147–158
2. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP problems in process synthesis and design. Comput Chem Eng Suppl 21:S445–S450
3. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB, for general twice-differentiable NLPs – II. Implementation and computational results. Comput Chem Eng 22(9):1159–1179
4. Adjiman CS, Androulakis IP, Floudas CA (2000) Global optimization of mixed-integer nonlinear problems. AIChE J 46:1769–1797
5. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB, for general twice-differentiable NLPs – I. Theoretical advances. Comput Chem Eng 22(9):1137–1158
6. Adjiman CS, Floudas CA (1996) Rigorous convex underestimators for general twice-differentiable problems. J Global Optim 9:23–40
7. Adjiman CS, Schweiger CA, Floudas CA (1998) Mixed-integer nonlinear optimization in process synthesis. In: Du D-Z, Pardalos PM (eds) Handbook Combinatorial Optim. Kluwer, Dordrecht
8. Aggarwal A, Floudas CA (1990) Synthesis of general distillation sequences – Nonsharp separations. Comput Chem Eng 14(6):631
9. Aggarwal A, Floudas CA (1992) Synthesis of heat integrated nonsharp distillation sequences. Comput Chem Eng 16:89
10. Al-Khayyal FA (1990) Jointly constrained bilinear programs and related problems: An overview. Comput Math Appl 19:53
11. Allgor RI, Barton PI (1997) Mixed integer dynamic optimization. Comput Chem Eng 21:S451–S456
12. Androulakis IP, Maranas CD, Floudas CA (1995) αBB: A global optimization method for general constrained nonconvex problems. J Global Optim 7:337–363
13. Barbosa-Povoa AP, Macchietto S (1994) Detailed design of multipurpose batch plants. Comput Chem Eng 18(11–12):1013–1042
14. Barton PI, Allgor RJ, Feehery WF (1998) Dynamic optimization in a discontinuous world. I–EC Res 37(3):966–981
15. Beale EML (1977) Integer programming. The State of the Art in Numerical Analysis. Acad Press, New York, pp 409–448
16. Bertsekas DL, Lower GS, Sandell NR, Posbergh TA (1983) Optimal short term scheduling of large-scale power systems. IEEE Trans Autom Control AC-28:1
17. Birewar DB, Grossmann IE (1989) Incorporating scheduling in the optimal design of multiproduct batch plants. Comput Chem Eng 13:141–161
18. Birewar DB, Grossmann IE (1990) Simultaneous synthesis, sizing and scheduling of multiproduct batch plants. Comput Chem Eng 29(11):2242
19. Bloom JA (1983) Solving an electricity generating capacity expansion planning problem by generalized benders' decomposition. Oper Res 31(5):84
20. Borchers B, Mitchell JE (1991) An improved branch and bound algorithm for mixed integer nonlinear programs. Techn Report Renssellaer Polytechnic Inst 200
21. Camarda KV, Maranas CD (1999) Optimization in polymer design using connectivity indices. I–EC Res 38:1884–1892
22. Churi N, Achenie LEK (1996) Novel mathematical programming model for computer aided molecular design. Industr Eng Chem Res 35(10):3788–3794
23. Churi N, Achenie LEK (1997) The optimal design of refrigerant mixtures for a two-evaporator refrigeration system. Comput Chem Eng 21(13):349–354
24. Ciric AR, Floudas CA (1989) A retrofit approach of heat exchanger networks. Comput Chem Eng 13(6):703
25. Ciric AR, Floudas CA (1990) A mixed-integer nonlinear programming model for retrofitting heat exchanger networks. I–EC Res 29:239
26. Ciric AR, Gu DY (1994) Synthesis of nonequilibrium reactive distillation processes by MINLP optimization. AIChE J 40(9):1479–1487
27. Ciric AR, Huchette SG (1993) Multiobjective optimization approach to sensitivity analysis waste treatment costs in discrete process synthesis and optimization problems. I–EC Res 32(11):2636–2646
28. Daichendt MM, Grossmann IE (1994) Preliminary screening for the MINLP synthesis of process systems: I. Aggregation techniques. Comput Chem Eng 18:663
29. Daichendt MM, Grossmann IE (1994) Preliminary screening for the MINLP synthesis of process systems: II. Heat exchanger networks. Comput Chem Eng 18:679
30. Dean JP, Dervakos GA (1996) Design of process-compatible biological agents. Comput Chem Eng Suppl A 20:S67–S72
31. Dimitriadis VD, Pistikopoulos EN (1995) Flexibility analysis of dynamic systems. I–EC Res 34(12):4451–4462
32. Dorneigh MC, Sahinidis NV (1995) Global optimization algorithms for chip layout and compaction. Eng Optim 25(2):131–154
33. Dua V, Pistikopoulos EN (1998) Optimization techniques for process synthesis and material design under uncertainty. Chem Eng Res Des 76(A3):408–416
34. Duran MA, Grossmann IE (1986) An outer approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307
35. Duvedi A, Achenie LEK (1997) On the design of environmentally benign refrigerant mixtures: A Mathematical Programming Approach. Comput Chem Eng 21(8):915–923

36. Faqir NM, Karimi IA (1990) Design of multipurpose batch plants with multiple production routes. Proc FOCAPD '89, Snowmass, Colorado, p 451
37. Fletcher R, Hall JAJ, Johns WR (1991) Flexible retrofit design of multiproduct batch plants. Comput Chem Eng 15(12):843
38. Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Math Program 66(3):327–349
39. Fletcher R, Leyffer S (1998) Numerical experience with lower bounds for MIQP branch and bound. SIAM J Optim 8(2):604–616
40. Floudas CA (1995) Nonlinear and mixed integer optimization: Fundamentals and applications. Oxford Univ. Press, Oxford
41. Floudas CA (2000) Deterministic global optimization: Theory, methods and applications. Nonconvex Optim Appl. Kluwer, Dordrecht
42. Floudas CA, Aggarwal A, Ciric AR (1989) Global optimum search for nonconvex NLP and MINLP problems. Comput Chem Eng 13(10):1117–1132
43. Floudas CA, Ciric AR (1989) Strategies for overcoming uncertainties in heat exchanger network synthesis. Comput Chem Eng 13(10):1133
44. Floudas CA, Grossmann IE (1995) Algorithmic approaches to process synthesis: logic and global optimization. FOCAPD'94, 91 AIChE Symp Ser, pp 198–221
45. Ganish B, Horsky D, Srikanth K (1983) An approach to optimal positioning of a new product. Managem Sci 29:1277
46. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
47. Georgiadis MC, Rotstein GE, Macchietto S (1997) Optimal layout design in multipurpose batch plants. Industr Eng Chem Res 36(11):4852–4863
48. Geromel JC, Belloni MR (1986) Nonlinear programs with complicating variables: theoretical analysis and numerical experience. IEEE Trans Syst, Man Cybern SMC-16:231
49. Grossmann IE (1990) MINLP optimization strategies and algorithms for process synthesis. Proc 3rd Internat Conf on Foundations of Computer-Aided Process Design, p 105
50. Grossmann IE (1996) Mixed integer optimization techniques for algorithmic process synthesis. In: Anderson JL (ed) Advances in chemical engineering, process synthesis, vol 23. Acad Press, New York, pp 171–246
51. Grossmann IE, Floudas CA (1987) Active constraint strategy for flexibility analysis in chemical processes. Comput Chem Eng 11(6):675
52. Grossmann IE, Quesada I, Ramon R, Voudouris VT (1992) Mixed-integer optimization techniques for the design and scheduling of batch processes. Proc NATO Advanced Study. Inst Batch Process Systems Engineering
53. Grossmann IE, Sargent RWH (1979) Optimal design of multipurpose chemical plants. Industr Eng Chem Process Des Developm 18:343
54. Gupta A, Manousiouthakis V (1993) Minimum utility cost of mass exchange networks with variable single component supplies and targets. I–EC Res 32(9):1937–1950
55. Gupta OK (1980) Branch and bound experiments in nonlinear integer programming. PhD Thesis, Purdue Univ
56. Gupta OK, Ravindran R (1985) Branch and bound experiments in convex nonlinear integer programming. Managem Sci 31(12):1533–1546
57. Harding ST, Floudas CA (1997) Global optimization in multiproduct and multipurpose batch design under uncertainty. Industr Eng Chem Res 36(5):1644–1664
58. Hatzimanikatis V, Floudas CA, Bailey JE (1996) Analysis and design of metabolic reaction networks via mixed-integer linear optimization. AIChE J 42(5):1277–1292
59. Hatzimanikatis V, Floudas CA, Bailey JE (1996) Optimization of regulatory architectures in metabolic reaction networks. Biotechnol and Bioengin 52:485–500
60. Hoang HH (1982) Topological optimization of networks: A nonlinear mixed integer model employing generalized Benders decomposition. IEEE Trans Autom Control AC-27:164
61. Holmberg K (1990) On the convergence of the cross decomposition. Math Program 47:269
62. Ierapetritou MG, Pistikopoulos EN (1994) Simultaneous incorporation of flexibility and economic risk in operational planning under uncertainty. Comput Chem Eng 18(3):163–189
63. Ierapetritou MG, Pistikopoulos EN (1996) Batch plant design and operations under uncertainty. Industr Eng Chem Res 35(2):772–787
64. Ierapetritou MG, Pistikopoulos EN, Floudas CA (1996) Operational planning under uncertainty. Comput Chem Eng 20(12):1499–1516
65. Kalitventzeff B, Marechal F (1988) The management of a utility network. Process Systems Engineering. PSE '88, Sydney, Australia, p 223
66. Kelley JE (1960) The cutting plane method for solving convex programs. J SIAM 8(4):703–712
67. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flow sheets. I–EC Res 26(9):1869
68. Kocis GR, Grossmann IE (1988) Global optimization of nonconvex MINLP problems in process synthesis. I–EC Res 27(8):1407
69. Kocis GR, Grossmann IE (1989) Computational experience with DICOPT solving MINLP problems in process systems engineering. Comput Chem Eng 13:307
70. Kocis GR, Grossmann IE (1989) A modelling and decomposition strategy for the MINLP optimization of process flowsheets. Comput Chem Eng 13(7):797–819
71. Kokossis AC, Floudas CA (1990) Optimization of complex reactor networks-I. isothermal operation. Chem Eng Sci 45(3):595

72. Kokossis AC, Floudas CA (1991) Optimal synthesis of isothermal reactor-separator-recycle systems. Chem Eng Sci 46:1361
73. Kokossis AC, Floudas CA (1994) Optimization of complex reactor networks - II. Nonisothermal operation. Chem Eng Sci 49(7):1037
74. Kokossis AC, Floudas CA (1994) Stability in optimal design: Synthesis of complex reactor networks. AIChE 40(5):849–861
75. Kravanja Z, Grossmann IE (1990) PROSYN - An MINLP process synthesizer. Comput Chem Eng 14:1363
76. Kravanja Z, Grossmann IE (1996) A computational approach for the modeling/decomposition strategy in the MINLP optimization of process flowsheets with implicit models. I–EC Res 35(6):2065–2070
77. Liu ML, Sahinidis NV (1997) Process planning in a fuzzy environment. Europ J Oper Res 100(1):142–169
78. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control, Part 1: A multiobjective framework and application to binary distillation synthesis. Comput Chem Eng 18(10):933–969
79. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control, Part 2: Reactor–separator–recycle system. Comput Chem Eng 18(10):971–994
80. Maranas CD (1996) Optimal computer-aided molecular design: A polymer design case study. Industr Eng Chem Res 35(10):3403–3414
81. Maranas CD (1997) Optimal molecular design under property prediction uncertainty. AIChE J 43(5):1250–1264
82. Mawengkang H, Murtagh BA (1986) Solving nonlinear integer programs with large scale optimization software. Ann Oper Res 5:425
83. McCormick GP (1976) Computability of global solutions to factorable nonconvex programs: Part I – Convex underestimating problems. Math Program 10:147–175
84. Mockus L, Reklaitis GV (1996) A new global optimization algorithm for batch process scheduling. In: Floudas CA, Pardalos PM (eds) State of the art in global optimization. Kluwer, Dordrecht, pp 521–538
85. Mohideen MJ, Perkins JD, Pistikopoulos EN (1996) Optimal design of dynamic systems under uncertainty. AIChE J 42, no.8:2251–2272
86. Mohideen MJ, Perkins JD, Pistikopoulos EN (1997) Robust stability considerations in optimal design of dynamic systems under uncertainty. J Process Control 7(5):371–385
87. Moore RE (1979) Methods and applications of interval analysis. SIAM, Philadelphia
88. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Interscience Ser Discrete Math and Optim. Wiley, New York
89. Neumaier A (1990) Interval methods for systems of equations. Encycl Math Appl. Cambridge Univ. Press, Cambridge
90. Novak Z, Kravanja Z, Grossmann IE (1996) Simultaneous synthesis of distillation sequences in overall process schemes using an improved MINLP approach. Comput Chem Eng 20(12):1425–1440
91. Odele O, Macchietto S (1993) Computer aided molecular design: A novel method for optimal solvent selection. Fluid Phase Equilib 82:47
92. Ostrovsky GM, Ostrovsky MG, Mikhailow GW (1990) Discrete optimization of chemical processes. Comput Chem Eng 14:111–117
93. Papageorgaki S, Reklaitis GV (1990) Optimal design of multipurpose batch plants: 1. Problem formulation. I–EC Res 29(10):2054
94. Papageorgaki S, Reklaitis GV (1990) Optimal design of multipurpose batch plants: 2. A decomposition solution strategy. I–EC Res 29(10):2062
95. Papalexandri KP, Pistikopoulos EN (1993) An MINLP retrofit approach for improving the flexibility of heat exchanger networks. Ann Oper Res 42:119
96. Papalexandri KP, Pistikopoulos EN (1994) Synthesis and retrofit design of operable heat exchanger networks: 1. Flexibility and structural controllability aspects. I–EC Res 33:1718
97. Papalexandri KP, Pistikopoulos EN (1994) Synthesis and retrofit design of operable heat exchanger networks: 2. Dynamics and control structure considerations. I–EC Res 33:1738
98. Papalexandri KP, Pistikopoulos EN (1996) Generalized modular representation framework for process synthesis. AIChE J 42:1010
99. Papalexandri KP, Pistikopoulos EN, Floudas CA (1994) Mass exchange networks for waste minimization: A simultaneous approach. Chem Eng Res Developm 72:279
100. Pardalos PM, Schnitger G (1988) Checking local optimality in constrained quadratic programming is NP-Hard. Oper Res Lett 7(1):33
101. Pardalos PM, Vavasis SA (1991) Quadratic programming with one negative eigenvalue is NP-hard. J Global Optim 1:15
102. Paules IV GE, Floudas CA (1988) Synthesis of flexible distillation sequences for multiperiod operation. Comput Chem Eng 12(4):267
103. Paules IV GE, Floudas CA (1989) APROS: Algorithmic development methodology for discrete-continuous optimization problems. Oper Res 37(6):902
104. Paules IV GE, Floudas CA (1992) Stochastic programming in process synthesis: A two-stage model with MINLP recourse for multiperiod heat-integrated distillation sequences. Comput Chem Eng 16(3):189
105. Penteado FD, Ciric AR (1996) An MINLP approach for safe process plant layout. I–EC Res 35(4):1354–1361
106. Petkov SB, Maranas CD (1997) Multiperiod planning and scheduling of multiproduct batch plants under demand uncertainty. I–EC Res 36(11):4864–4881
107. Petkov SB, Maranas CD (1997) Quantitative assessment of uncertainty in the optimization of metabolic pathways. Biotechnol and Bioengin 56(2):145–161

108. Petkov SB, Maranas CD (1998) Design of single product campaign batch plants under demand uncertainty. AIChE J 44(4):896–911
109. Pistikopoulos EN, Ierapetritou MG (1995) Novel approach for optimal process design under uncertainty. Comput Chem Eng 19(10):1089–1110
110. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput Chem Eng 16:937–947
111. Raman VS, Maranas CD (1999) Optimization in product design with properties correlated with topological indices. Comput Chem Eng 45:997–1017
112. Ratschek H, Rokne J (1984) Computer methods for the range of functions. Ellis Horwood Ser Math Appl. Halsted Press, New York
113. Reklaitis GV (1991) Perspectives on scheduling and planning process operations. Proc. 4th. Internat. Symp. on Process Systems Engineering, Montreal, Canada
114. Rouhani R, Lasdon L, Lebow W, Warren AD (1985) A generalized Benders decomposition approach to reactive source planning in power systems. Math Program Stud 25:62
115. Ryoo HS, Sahinidis NV (1995) Global optimization of nonconvex NLPs and MINLPs with applications in process design. Comput Chem Eng 19(5):551–566
116. Sahinidis NV, Grossmann IE (1991) Convergence properties of generalized Benders decomposition. Comput Chem Eng 15(7):481
117. Schweiger CA, Floudas CA (1997) MINOPT: A software package for mixed-integer nonlinear optimization, user's guide. Manual Computer-Aided Systems Lab Dept Chemical Engin Princeton Univ
118. Schweiger CA, Floudas CA (1998) Interaction of design and control: optimization with dynamic models. In: Hager WW, Pardalos PM (eds) Optimal Control: Theory, Algorithms, and Applications. Kluwer, Dordrecht, pp 388–435
119. Schweiger CA, Floudas CA (1999) Optimization framework for the synthesis of complex reactor networks. I–EC Res 38:744–766
120. Skrifvars H, Harjunkoski I, Westerlund T, Kravanja Z, Pörn R (1996) Comparison of different MINLP methods applied on certain chemical engineering problems. Comput Chem Eng Suppl 20:S333–S338
121. Smith EMB, Pantelides CC (1996) Global optimisation of general process models. In: Grossmann IE (ed) Global Optimization in Engineering Design. Kluwer, Dordrecht, pp 355–386
122. Smith EMB, Pantelides CC (1999) A symbolic reformulation/spatial branch and bound algorithm for the global optimization of nonconvex MINLPs. Comput Chem Eng 23:457–478
123. Stefanis SK, Livingston AG, Pistikopoulos EN (1997) Environmental impact considerations in the optimal design and scheduling of batch processes. Comput Chem Eng 21(10):1073–1094
124. Turkay M, Grossmann IE (1996) Logic-based MINLP algorithms for the optimal synthesis of process networks. Comput Chem Eng 20(8):959–978
125. Vaidyanathan R, El-Halwagi M (1996) Global optimization of nonconvex MINLP's by interval analysis. In: Grossmann IE (ed) Global Optimization in Engineering Design. Kluwer, Dordrecht, pp 175–193
126. Vaidyaraman S, Maranas CD (1999) Optimal synthesis of refrigation cycles and selection of refrigerants. AIChE J 45:997–1017
127. Vaselenak J, Grossmann IE, Westerberg AW (1987) An embedding formulation for the optimal scheduling and design of multiproduct batch plants. I–EC Res 26(1):139
128. Vaselenak J, Grossmann IE, Westerberg AW (1987) Optimal retrofit design of multipurpose batch plants. I–EC Res 26(4):718
129. Vavasis S (1991) Nonlinear optimization: Complexity issues. Oxford Univ. Press, Oxford
130. Vecchietti A, Grossmann IE (1999) LOGMIP: A disjunctive 0-1 nonlinear optimizer for process system models. Comput Chem Eng 23:555–565
131. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer-approximation for MINLP optimization. Comput Chem Eng 14(7):769–782
132. Wellons MC, Reklaitis GV (1991) Scheduling of multipurpose batch chemical plants: 1. Formulation of single-product campaigns. I–EC Res 30(4):671
133. Wellons MC, Reklaitis GV (1991) Scheduling of multipurpose batch chemical plants: 1. Multiple product campaign formulation and production planning. I–EC Res 30(4):688
134. Westerlund T, Pettersson F (1995) An extended cutting plane method for solving convex MINLP problems. Comput Chem Eng 19:131–136
135. Westerlund T, Pettersson F, Grossmann IE (1994) Optimization of pump configuration problems as a MINLP problem. Comput Chem Eng 18(9):845–858
136. Westerlund T, Skrifvars H, Harjunkoski I, Pörn R (1998) An extended cutting plane method for a class of non-convex MINLP problems. Comput Chem Eng 22(3):357–365
137. Xia Q, Macchietto S (1997) Design and synthesis of batch plants: MINLP solution based on a stochastic method. Comput Chem Eng 21:S697–S702
138. Yee TF, Grossmann IE (1990) Simultaneous optimization models for heat integration - II. Heat exchanger network synthesis. Comput Chem Eng 14(10):1165–1184
139. Yee TF, Grossmann IE, Kravanja Z (1990) Simultaneous optimization models for heat integration - I. Area and energy targeting and modeling of multi-stream exchangers. Comput Chem Eng 14(10):1151–1164
140. Yee TF, Grossmann IE, Kravanja Z (1990) Simultaneous optimization models for heat integration - III. Area and energy targeting and modeling of multi-stream exchangers. Comput Chem Eng 14(11):1185–1200

141. Zamora JM, Grossmann IE (1998) Continuous global optimization of structured process systems models. Comput Chem Eng 22(12):1749–1770
142. Zamora JM, Grossmann IE (1998) A global MINLP optimization algorithm for the synthesis of heat exchanger networks with no stream splits. Comput Chem Eng 22(3):367–384
143. Zhang X, Sargent RWH (1994) The optimal operation of mixed production facilities: General formulation and some solution approaches. Proc. 5th Int. Symp. Process Systems Engineering, pp 171–177

Mixed Integer Optimization in Well Scheduling

DIMITRIOS I. GEROGIORGIS,
VASSILEIOS D. KOSMIDIS,
EFSTRATIOS N. PISTIKOPOULOS
Centre for Process Systems Engineering,
Imperial College London, London, UK

MSC2000: 76T30, 90C11, 90C90

Article Outline

Abstract
Introduction
Problem Statement
Optimization Model
  Wellbore Model
  Mass, Momentum and Energy Balances in Well, Manifold and Separator Nodes
  Network Logic Constraints
  Well and Flowline Momentum and Energy Balances
  Maximum Number of Well Switches
  Objective Function
An MINLP Formulation for the Well Scheduling Problem
Optimization Strategy
Solution Procedure
Example Problems
  Example 1
  Example 2
Integration of Reservoir Multiphase Flow Simulation and Optimization
Literature Review and Challenges for Integrated Modeling and Optimization
Problem Definition and Model Formulation
Reservoir Multiphase Flow Simulation Results
Oil Production Optimization Results
Conclusions
References

Abstract

This Chapter presents a novel, mixed integer nonlinear programming (MINLP) model for the well scheduling problem, where the nonlinear behavior of the reservoir, wells, pipelines and surface facilities has been incorporated into the mathematical formulation. The well scheduling problem is formulated as a snapshot optimization problem with an objective function that expresses the maximization of an economic index. Discrete decisions here include the operational status of the wells, the allocation of wells to manifold or separators and the allocation of surface flowlines to separators. Continuous decisions include the well oil flowrates, and the allocation of gas-to-gas lift wells.

A three-step solution strategy is proposed for the solution of this problem, where logic based relations and piecewise linear approximations of oil field wells are integrated in the MINLP formulation. The model is solved following an Outer Approximation (OA) class algorithm. A number of examples are presented to illustrate the performance and business value of the proposed strategy; a remarkable increase in oil production of up to 10% is demonstrated, compared to results obtained via widespread heuristic methods. A further increase of 2.9% can be achieved by dynamic optimization based on explicit consideration of the multiphase flow within the reservoirs of a particular oil field.

Introduction

In an era of globalized business operations, large and small oil and gas producers alike strive to foster profitability by improving the agility of exploration endeavors and the efficiency of oil production, storage and transport operations [7]. Consequently, they all face acute challenges: ever-increasing international production, intensified global competition, price volatility, operational cost reduction policies, aggressive financial goals (market share, revenue, cash flow and profitability) and strict environmental constraints (offshore extraction, low sulphur): all these necessitate a high level of oilfield modeling accuracy, so as to maximize recovery from certified reserves. Straightforward translation

of all considerations to explicit mathematical objectives and constraints can yield optimal oilfield network design, planning and operation policies. Therefore, the foregoing goals and constraints should be explicitly incorporated and easily revised if the generality of production optimization algorithms is to be preserved. This Chapter provides a summary of a new, efficient MINLP optimization formulation for well scheduling, and a novel strategy towards integration of equation-oriented process modeling and multiphase reservoir computational fluid dynamics (CFD), in order to include the dynamic behavior of reservoirs into oil and gas production models.

The problem of fuel production optimization subject to explicit oilfield constraints has attracted significant attention, documented in many petroleum engineering publications. A comprehensive literature review by Kosmidis [18] classifies previous algorithms in 3 broad categories (simulation, heuristic, and mathematical programming methods) and underlines that most are applied either to simple pipeline networks of modest size, relying on heuristic rules of limited applicability, or are suitable for special structures. Reducing the computational burden (focus on natural-flow wells or gas-lift wells only, or reducing well network connectivity discrete variables) is a crucial underlying pattern.

Dynamic oil and gas production systems simulation and optimization is a research trend which has the clear potential to meet the foregoing challenges of the international oil and gas industry and assist producers in achieving business goals and energy needs. Previous work [8,19,20,23] has addressed successfully research challenges in this field, using appropriate simplifying correlations [25] for two-phase flow of oil and gas in production wells and pipelines. A series of assumptions are routinely adopted to achieve manageable computational complexity: the fundamental one is the steady-state assumption for the reservoir model, based on the enormous timescale difference between different spatial levels (oil and gas reservoir dynamics evolve in the order of weeks, the respective timescales of pipeline networks are in the order of minutes, and the production optimization horizon is in the order of days). The decoupling of reservoir simulation from surface facilities optimization is based on these timescale differences among production elements [2,25]. While the surface and pipeline facilities are in principle no different from those found in any petrochemical plant, sub-surface elements (reservoirs, wells) induce complexity which must be addressed via a systematic strategy that has not been hitherto proposed.

In some petroleum fields, such as the Prudhoe Bay [22], a production well can be connected to different manifolds that lead to different separators. In such fields, switching a well from one manifold to another could be an effective way to increase oil production and/or reduce production cost by making optimal use of the existing resources such as the capacity of separators [9]. However, for best results, the well connection must be optimized simultaneously with the well oil rate and gas lift rate. The corresponding optimization problem is known as well scheduling problem.

Problem Statement

The well scheduling problem in integrated oil and gas production can be stated as follows: given are (i) a *set of wells*, which could be closed (shut in) or connected to manifolds or separators, (ii) a *set of flowlines* which could be connected to separators. The goal is to determine: (i) the *operational status of the wells*, i. e. closed or open, (ii) the *connection of wells* to manifolds or separators, (iii) the *connection of flow lines* to separators, (iv) the *well oil flowrate* and the (v) the *allocation of gas* to gas lift wells, which maximize the net revenue (oil sales minus the cost of gas compression), while satisfying physical laws and operational constraints such as: (i) a *well bore model*, (ii) *mass, energy and momentum balances* throughout the production network, (iii) *upper and lower well oil rate constraints* and minimum pressure constraints at the inlet and outlet of the flowlines, (iv) *maximum oil, gas and water capacity constraints* in the separators, (v) an *upper bound on gas lift availability* and (vi) a *maximum number of well switches*. The first step in order to determine the optimal well configuration that maximizes the revenue is to develop a suitable superstructure that includes all the possible pipeline network configurations. Such a production network superstructure is shown in Fig. 1 and includes the *reservoir (R)*, the *wells (W)*, the *manifolds (M)*, and the *separators (S)* nodes as well as the potential connection of wells to manifolds or separators and flowlines to separators. Two types of wells are considered: (i) *type A wells* that can be connected only

to manifolds, and (ii) *type B wells* that can be connected to separators. It must be noted that a feasible production network should satisfy the following requirements: (i) a *type A well* should be either shut in or else connected to one manifold, (ii) a *type B well* should be either shut in or else connected to one separator, and (iii) a manifold flowline must be connected to one separator.

Optimization Model

This section presents the MINLP optimization model for the well scheduling problem, based on the following assumptions:

- the system is under steady state conditions,
- a homogeneous slip model which is applied to determine the pressure drop in the pipelines,
- the temperature of the reservoir is known,
- the operating pressures of the separators are constant, and
- the thermodynamic description of the fluid is based on the *black oil model*.

For the development of the MINLP optimization model, the following sets, variables and parameters are defined:

Sets

I set of wells
I_A set of wells of type A
I_B set of wells of type B
M set of manifolds
S set of separators

Indices

i, i_A, i_B well in set I, I_A, I_B respectively
m manifold in set M
s separator in set S

Binary decision variables

$$y_i = \begin{cases} 1 & \text{if well } i \text{ is open} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{i,m} = \begin{cases} 1 & \text{if well } i \text{ is connected to manifold } m \\ 0 & \text{otherwise} \end{cases}$$

$$y_{m,s} = \begin{cases} 1 & \text{if manifold } m \text{ is connected to separator } s \\ 0 & \text{otherwise} \end{cases}$$

$$y_{i,s} = \begin{cases} 1 & \text{if well } i \text{ is connected to separator } s \\ 0 & \text{otherwise} \end{cases}$$

Continuous variables

$q_{p,i}$ flowrate in stock tank conditions of phase p from well i
$q_{p,i,m}$ flowrate in stock tank condition of phase p from well i to manifold m
$q_{p,m,s}$ flowrate in stock tank condition of phase p from manifold m to separator s
$q_{p,s}$ flowrate in stock tank condition of phase p in separator s
$P_{i,m}$ pressure of well i at the manifold level
P_m manifold pressure
$H_{i,m}$ total enthalpy of well i at the manifold level
H_m manifold enthalpy

The proposed model includes the following elements: (i) the well bore model, (ii) the mass, momentum and energy balances in well, manifold and separator nodes, (iii) the network logic constraints, (iv) the well and flowline momentum and energy balances, (v) the maximum number of allowable well switches, and (vi) the objective function.

Wellbore Model

The wellbore model describes the multiphase fluid flow from the reservoir to the wellbore and comprises the following equations:

$$q_{o,i} = PI_i(P_{R,i} - P_i^{\text{wf}}) \,, \ \forall i \in I \tag{1}$$

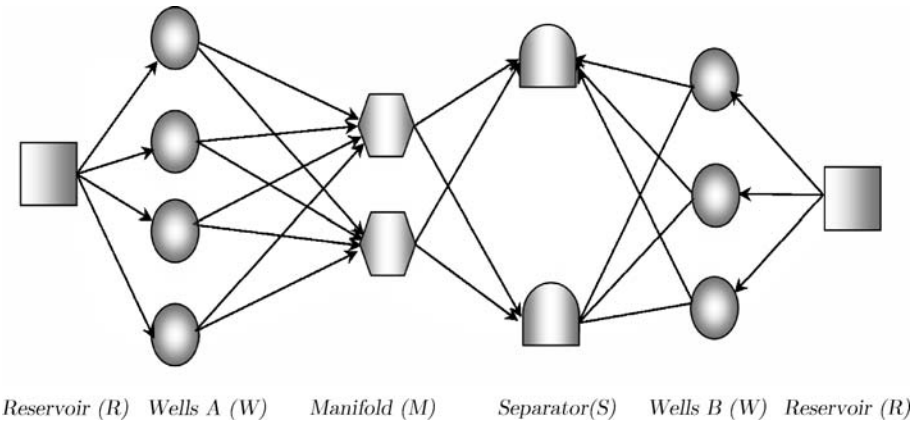
$$q_{g,i}^{\text{f}} = f_o(q_{o,i}) \,, \ \forall i \in I \tag{2}$$

$$q_{w,i} = f_w(q_{o,i}) \,, \ \forall i \in I \tag{3}$$

$$T_i = T_R \,, \ \forall i \in I \tag{4}$$

$$H_i = f_H(P_i^{\text{wf}}, T_i, q_{o,i}, q_{w,i}, q_{g,i}) \,, \ \forall i \in I \tag{5}$$

where P_i^{wf} is the bottomhole pressure and $q_{g,i}^{\text{f}}$ is the formation gas flowrate in stock tank conditions. Equations (2) and (3) can be nonlinear in order to model the case of gas and water coning wells. These nonlinear relations are generated either by using Addington's correlations [1] or by repetitively solving a well coning model for different values of well oil rate $q_{o,i}$, in order to calculate the corresponding water $q_{w,i}$ and gas



Mixed Integer Optimization in Well Scheduling, Figure 1
Production network superstructure for the well scheduling problem

flowrates $q^{\mathrm{f}}_{g,i}$. In the latter case, Eq. (2) and (3) are constructed via curve fitting to the data series $(q_{o,i},\ q_{o,i})$ and $(q_{o,i},\ q^{\mathrm{f}}_{g,i})$ respectively. For naturally flowing wells, the total gas flowrate is given by:

$$q_{g,i} = q^{\mathrm{f}}_{g,i}\ ,\ \forall i \in D = \{i \in I \mid \text{natural flow}\} \qquad (6)$$

while for gas lift wells the total gas flowrate is equal to:

$$q_{g,i} = q^{\mathrm{f}}_{g,i} + q^{\mathrm{inj}}_{g,i}\ ,\ \forall i \in F = \{i \in I \mid \text{gas lift}\}\ . \qquad (7)$$

Mass, Momentum and Energy Balances in Well, Manifold and Separator Nodes

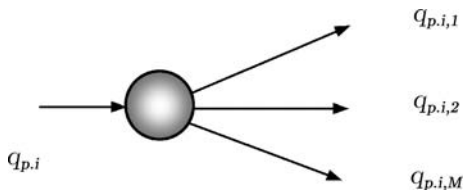
A well node of type A can be modeled as a splitter $i \in I_A$, which consists of an inlet stream that represents the fluid flow from the reservoir, and a set of outlet streams that represents the potential connections of a well to manifolds as shown in Fig. 2. The mass balances around the splitter for each phase are given by the following relations:

$$q_{p,i} = \sum_m q_{p,i,m}\ ,\ \forall p \in \{o, w, g\}\ ,\ i \in I_A\ . \qquad (8)$$

Similarly, the mass balances around a well node of type B are given by:

$$q_{p,i} = \sum_s q_{p,i,s}\ ,\ \forall p \in \{o, w, g\}\ ,\ i \in I_B\ . \qquad (9)$$

There is also an upper and a lower bound in the well oil flowrates. The upper bound is enforced to prevent



Mixed Integer Optimization in Well Scheduling, Figure 2
Splitter node

sand production [4], while the lower bound is imposed to satisfy stable flow [27]:

$$y_i q^L_{o,i} \le q_{o,i} \le q^U_{o,i} y_i\ ,\ \forall i \in I\ . \qquad (10)$$

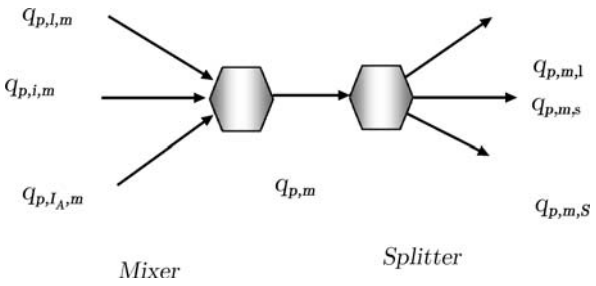
Equation (10) states that if well i is open ($y_i = 1$), then the well oil flowrate $q_{o,i}$ is constraint by an upper and a lower bound, while if well i is shut in ($y_i = 0$), then the well oil flowrate $q_{o,i}$ is zero.

Manifold Node A manifold node is shown in Fig. 3 and performs two tasks: (i) mixing and (ii) splitting. The mass balance of the mixer for each phase is given by:

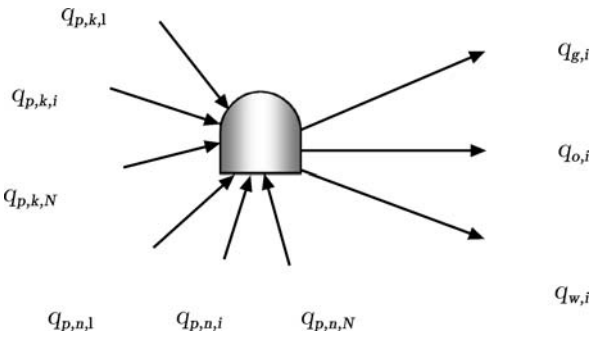
$$\sum_{i \in I_A} q_{p.i.m} = q_{p.m}\ ,\ \forall p \in \{o, w, g\}\ ,\ m \in M\ . \qquad (11)$$

The splitter allocates the manifold fluid $q_{p,m}$ to one separator s. The mass balances for each phase around the splitter are given by:

$$q_{p,m} = \sum_s q_{p,m,s}\ ,\ \forall p, m \in M\ . \qquad (12)$$



Mixed Integer Optimization in Well Scheduling, Figure 3
Manifold node



Mixed Integer Optimization in Well Scheduling, Figure 4
Separation node

All wells that are connected to manifold m must operate at the same pressure (P_m):

$$L^P(1 - y_{i,m}) \le P_i^m - P_m \le U^P(1 - y_{i,m}) , \quad \forall i \in I_A , \ m \in M \tag{13}$$

where $P_{i,m}$ is the pressure of well i at manifold level m, and L^P, U^P are the corresponding upper and lower bounds, respectively. Moreover, if manifold m is connected to separator s, then its inlet pressure must be greater than the separator pressure:

$$P_s y_{m,s} \le P_m , \ \forall m \in M , \ s \in S . \tag{14}$$

The pressure of the flowline at the separator level $P_{m,s}$ is equal to the separator pressure:

$$P_{m,s} = \sum_s y_{m,s} P_s . \tag{15}$$

The energy balance in the manifold is given by:

$$\sum_{i \in I_A} H_i^m = H_m , \ \forall m \in M \tag{16}$$

where H_i^m is the enthalpy of well i at manifold level m.

Separator Node Each separator s has a set of inlet streams coming from the flowlines and type B wells, as shown in Fig. 4. The mass balances for each phase are given by the following relation:

$$\sum_m q_{p,m,s} + \sum_{i \in I_B} q_{p,i,s} = q_{p,s} , \ \forall p, s \in S \tag{17}$$

while the separator capacity constraints must also be satisfied:

$$q_{p,s} \le C_{p,s} , \ \forall p, s \in S . \tag{18}$$

Finally, the total amount of gas available for gas lift is restricted by the compressor capacity (C_C):

$$\sum_i q_{g,i}^{\text{inj}} \le C_c . \tag{19}$$

Network Logic Constraints

A well of type A could either be shut in, or else connected to one manifold:

$$\sum_m y_{i,m} = y_i , \ \forall i \in I_A \tag{20}$$

$$y_{i,m} \le y_i , \ \forall i \in I_A , \ m \in M . \tag{21}$$

The integer Eq. (20) states that if the well is open ($y_i = 1$) then it should be connected to one manifold, while Eq. (21) states that if the well is shut in ($y_i = 0$) then all binary variables $y_{i,m}$ which represent the connection of well i to manifold m are zero.

Similarly, a well of type B could either be shut in, or else connected to one separator:

$$\sum_s y_{s,i} = y_i , \ \forall i \in I_B \tag{22}$$

$$y_{s,i} \le y_i , \ \forall i \in I_B , \ s \in S . \tag{23}$$

Furthermore, it is also necessary to enforce the condition that each manifold flowline is connected to one separator:

$$\sum_s y_{m,s} = 1 , \ \forall m \in M . \tag{24}$$

Moreover, if the connection of well i to manifold m or separator s does not exist, then its corresponding flowrates and enthalpies must be zero:

$$0 \le H_i^m \le H^U y_{i,m}\,,\ \forall i \in I_A \tag{25}$$

$$0 \le q_{p,i,m} \le q_{p,i,m}^U y_{i,m}\,,\ \forall p, i \in I_A\,,\ m \in M \tag{26a}$$

$$0 \le q_{p,i,s} \le q_{p,i,s}^U y_{i,s}\,,\ \forall p, i \in I_B\,,\ s \in S \tag{26b}$$

Well and Flowline Momentum and Energy Balances

1. *Naturally flowing wells of type A.* Kosmidis [18] discusses how naturally flowing wells of type A can be accurately approximated by piecewise linear functions:

$$q_{o,i}^{\max} = \sum_j \eta_{i,j} q_{o,i,j}^{d,\max}\,,\ \forall i \in I_A \tag{27a}$$

$$P_i^m = \sum_j \eta_{i,j} P_{i,j}^d\,,\ \forall i \in I_A \tag{27b}$$

$$H_i^m = \sum_j \sum_k \lambda_{i,j,k} H_{i,j,k}^d\,,\ \forall i \in I_A \tag{27c}$$

$$q_{o,i} = \sum_j \sum_k \lambda_{i,j,k} q_{o,i,j,k}^d\,,\ \forall i \in I_A \tag{27d}$$

$$q_{o,i} \le q_{o,i}^{\max}\,,\ \forall i \in I_A \tag{27e}$$

$$\sum_j \sum_k \lambda_{i,j,k} = y_i\,,\ \forall i \in I_A \tag{27f}$$

$$\begin{aligned} &\eta_{i,j} = \sum_k \lambda_{i,j,k}\,,\ \xi_{i,k} = \sum_j \lambda_{i,j,k}\,,\\ &\zeta_{i,t} = \sum_j \lambda_{i,j,j+t}\,,\ \forall i \in I_A \end{aligned} \tag{27g}$$

$$\eta_{i,j}\,,\ \xi_{i,k}\,,\ \zeta_{i,t} \ge 0\,,\ SOS\,,\ \forall i \in I_A\,. \tag{27h}$$

It must be noted that if well i is shut in ($y_i = 0$), then all continuous variables in constraint (27) are set equal to zero, as it can be observed from constraint (27f). The piecewise linear approximation of the well model is constructed in a pre-processing step by discretizing:

(i) the manifold pressure between the valid lower (L^P) and upper (U^P) bound, and

(ii) the well oil rate in the interval $[q_{o,i}^L,\ q_{o,i}^U]$.

The lower bound (L^P) is equal to the lowest operating pressure of the separators, while the upper bound (U^P) must be greater than the highest operating pressure of the separators.

2. *Naturally flowing wells of type B.* For the case of naturally flowing wells of type B, the oil flowrate $q_{o,i,s}$ of well i in separator s is given by:

$$q_{o,i,s} \le q_{o,i,s}^{d,\max} y_{i,s}\,,\ \forall i \in I_B \tag{28}$$

where $q_{o,i,s}^{d,\max}$ is calculated in a pre-processing step for each fixed pressure separator s by setting the choke fully open.

3. *Gas lift wells of type A.* These can be accurately approximated by the following set of mixed integer linear relations:

$$q_{o,i} = \sum_j \sum_k \lambda_{i,j,k} q_{o,i,j,k}^d\,,\ \forall i \in I_B \tag{29a}$$

$$q_{g,i}^{\text{inj}} = \sum_j \sum_k \lambda_{i,j,k} q_{g,i,k}^{d,\text{inj}}\,,\ \forall i \in I_B \tag{29b}$$

$$P_i^m = \sum_j \sum_k \lambda_{i,j,k} P_{i,j}^d\,,\ \forall i \in I_B \tag{29c}$$

$$H_i^m = \sum_j \sum_k \lambda_{i,j,k} H_{i,j,k}^d\,,\ \forall i \in I_B \tag{29d}$$

$$\sum_j \sum_k \lambda_{i,j,k} = y_i\,,\ \forall i \in I_B \tag{29e}$$

$$\begin{aligned} &\eta_{i,j} = \sum_j \lambda_{i,j,k}\,,\ \xi_{i,k} = \sum_k \lambda_{i,j,k}\,,\\ &\zeta_{i,t} = \sum_j \mu_{i,j,j+t}\,,\ \forall i \in I_B \end{aligned} \tag{29f}$$

$$\eta_{i,j}\,,\ \xi_{i,k}\,,\ \zeta_{i,t} \ge 0\,(SOS)\,,\ \forall i \in I_B\,. \tag{29g}$$

These relations are constructed in a pre-processing step by discretizing:

(iii) the manifold pressure in the interval [L^P , U^P], and

(iv) the well gas injection rate in the interval $[0,\ q_{g,i}^{\text{inj},U}]$, where $q_{g,i}^{\text{inj},U}$ is the gas injection rate at the upper bound pressure (U^P), where the well oil flowrate is reduced despite the increase in gas injection rate.

4. *Gas lift wells of type B.* These are connected to a fixed pressure separator and they can be accurately approximated as follows:

$$q_{o,i,s} = \sum_j \lambda_{i,j,s} q^d_{o,i,j,s} \,, \ \forall i \in I_B \,, \ s \in S \quad (30a)$$

$$q^{\text{inj}}_{g,i,s} = \sum_j \lambda_{i,j,s} q^{d,\text{inj}}_{g,i,j,s} \,, \ \forall i \in I_B \,, \ s \in S \quad (30b)$$

$$\sum_j \lambda_{i,j,s} = y_{i,s} \,, \ \forall i \in I_B \,, \ s \in S \quad (30c)$$

$$\lambda_{i,j,s} \geq 0 \,, \ SOS \,. \quad (30d)$$

Flowline Momentum Balance The momentum balance in the manifold flowlines is given by:

$$P_{m,s} - f_P(P_m, H_m, q_{o,m}, q_{g,m}, q_{w,m}) = 0 \,, \quad \forall m \in M \,, \ s \in S \quad (31)$$

where $P_{m,s}$ is the pressure of flowline m at separator level s.

Remark During construction of the piecewise linear approximations or calculation of $q^{\max}_{o,i,s}$, it is possible to identify naturally flowing wells of type A or type B which are unable to flow towards certain manifolds or separators. To exclude these infeasible connections, the following logic constraints are incorporated in the mathematical formulation:

$$y_{i,m} \leq 1 - y_{m,s} \,, \ \forall i \in I_A \quad (32)$$

$$y_{i,s} \leq 0 \,, \ \forall i \in I_B \,. \quad (33)$$

Constraint (32) states that if flowline m is connected to separator $s (y_{m,s} = 1)$, then well i cannot be connected to manifold m.

Maximum Number of Well Switches

There is an upper bound on the number of well switches (for wells of both types A and B) that can be performed within a day. This is an operational constraint and is applied to avoid huge flow variations which may eventually lead to a surface facility shut down. To consider and model this requirement, the following binary variables and parameters are introduced:

Binary variables

$$c^{\text{f}}_i = \begin{cases} 1 & \text{if the well } i \text{ is open and in the previous day was closed} \\ 0 & \text{otherwise} \end{cases}$$

$$c^{\text{nf}}_i = \begin{cases} 1 & \text{if the well } i \text{ is closed and in the previous day was open} \\ 0 & \text{otherwise} \end{cases}$$

$$c_{i,m} = \begin{cases} 1 & \text{if the well } i \text{ of type A is connected to a new manifold } m \text{ on this day} \\ 0 & \text{otherwise} \end{cases}$$

$$c_{i,s} = \begin{cases} 1 & \text{if the well } i \text{ of type B is connected to a new separator } s \text{ on this day} \\ 0 & \text{otherwise} \end{cases}$$

$$c_{m,s} = \begin{cases} 1 & \text{if the flowline } m \text{ is connected to a new separator } s \text{ on this day} \\ 0 & \text{otherwise} \,. \end{cases}$$

Parameters

$NC^{\max}_A$ maximum number of switches for the wells of type A.

$NC^{\max}_B$ maximum number of switches for the wells of type B.

y^b binary parameters representing the well structure of the previous day.

One switch is accounted for in the following cases:

(i) *A well i was closed and is currently open.* This case is modeled by incorporating the following constraint in the formulation:

$$y_i - y^b_i \leq c^{\text{f}}_i \,, \ \forall i \in I \,. \quad (34)$$

Thus, if well i is open $(y_i = 1)$ while it was previously closed $(y^b_i = 0)$, then one well switch is accounted for by forcing the binary variable c^{f}_i to be 1.

(ii) *A well i was open and is currently closed.* To incorporate this well switch in the formulation, the following constraint is used:

$$y^b_i - y_i \leq c^{\text{nf}}_i \,, \ \forall i \in I \,. \quad (35)$$

(iii) *A well of type A switches manifold.* If well i is currently connected to manifold $m (y_{i,m} = 1)$ and it was previously connected to manifold $m' \neq m$

(that implies $y^b_{i,m} = 0$), then there is one well switch, which is modeled by the following constraint in the formulation:

$$y_{i,m} - y^b_{i,m} \le c_{i,m}\,,\ \forall i \in B\,,\ m \in M \qquad (36)$$

where the set $B = \{i \in I_A \mid y^b_i \ne 0\}$ is the set of wells of type A that were open during the previous day. Constraint (36) is applicable only for wells of type A that were previously open ($y^b_i \ne 0$), to avoid double counting a well switch; the case of a well that was closed ($y^b_i = 0$) and is currently open ($y_i = 1$) is considered by constraint (34).

(iv) *A well of type B switches to a new separator.* If well i is currently connected to separator $s(y_{i,s} = 1)$ and was previously connected to separator $s' \ne s(y^b_{i,s} = 0)$, then there is one well switch, which is modeled by the following constraint in the formulation:

$$y_{i,s} - y^b_{i,s} \le c_{i,s}\,,\ \forall i \in C\,,\ s \in S \qquad (37)$$

where the set $C = \{i \in I_B \mid y^b_i \ne 0\}$ is the set of wells of type B that were open during the previous day.

(v) *A manifold flowline switches to a new separator.* If a manifold flowline m is currently connected to a separator $s(y_{m,s} = 1)$ and was previously connected to separator $s' \ne s(y^b_{m,s} = 0)$, then there is one switch, which is accounted for by forcing the binary variable $c_{m,s}$ to be 1:

$$y_{m,s} - y^b_{m,s} \le c_{m,s}\,,\ \forall m \in M\,,\ s \in S\,. \qquad (38)$$

The sum of switches for the wells of type A and B must be less then an upper bound:

$$\sum_{i\in I_A}(c^{\mathrm{f}}_i + c^{\mathrm{nf}}_i) + \sum_{i\in B}\sum_{m\in M} c_{i,m} + \sum_{m\in M}\sum_{s\in S} c_{m,s} \le NC^{\max}_A \qquad (39)$$

$$\sum_{i\in I_A}(c^{\mathrm{f}}_i + c^{\mathrm{nf}}_i) + \sum_{i\in C}\sum_{s\in S} c_{i,s} \le NC^{\max}_B\,. \qquad (40)$$

Objective Function

The objective function is the maximization of daily revenue:

$$\max\ w_o \sum_{i\in I} q_{o,i} - w_g \sum_{i\in I} q^{\mathrm{inj}}_{g,i} \qquad (41)$$

The control variables are:

(i) the well operational status (open or close),
(ii) the well connections to manifolds and separators,
(iii) the flowline connections to separators,
(iv) the well oil flowrates, and
(v) the gas injection rates into gas lift wells.

An MINLP Formulation for the Well Scheduling Problem

By defining the vectors $\mathbf{x} = [P, H]$, $\mathbf{q}_p = [q_o, q_g, q_w, q^{\mathrm{inj}}_g]$, $\mathbf{y}^s = [y_i, y_{i,m}, y_{m,s}, y_{i,s}]$, $\mathbf{c} = [c^{\mathrm{f}}_i, c^{\mathrm{nf}}_i, c_{i,m}, c_{i,s}, c_{m,s}]$, $\boldsymbol{\phi} = [\lambda_{i,j}, \xi_{i,k}, \zeta_{i,t}]$ and $\mathbf{y}^a$ (the vector of binary variables that are used to impose the adjacency condition in *SOS*-type variables), the mathematical programming formulation (P) for the well scheduling problem can be concisely expressed as:

$$P\colon\ \max\ \psi(q_{o,i}, q^{\mathrm{inj}}_{g,i}) \qquad (42)$$

subject to

$$m_1(\mathbf{x}_i,\ \mathbf{q}_{p,i}) = 0 \qquad (43)$$

$$m_2(\mathbf{q}_{p,i},\ \mathbf{q}_{p,i,m}, \mathbf{q}_{p,i,s}, \mathbf{q}_{p,s}, \mathbf{x}_i, \mathbf{x}_{i,m}, \mathbf{y}^s) \le 0 \qquad (44)$$

$$m_3(\mathbf{q}_{p,i,m}, \mathbf{q}_{p,i,s}, \mathbf{y}^{\mathbf{s}}) \le 0 \qquad (45)$$

$$m_4(q_{o,i}, q^{\mathrm{inj}}_{g,i}, q_{o,i,s}, q^{\max}_{o,i,s}, q^{\mathrm{inj}}_{g,i,s}\mathbf{x}_{i,m}, \phi, \mathbf{y}^{\mathbf{s}}, \mathbf{y}^a) = 0 \qquad (46)$$

$$m_5(\mathbf{x}_m, \mathbf{q}_{p,m}) = 0 \qquad (47)$$

$$m_6(\mathbf{y}^s, \mathbf{c}) \le 0\,. \qquad (48)$$

The equivalence of the equations within the above model (P) is explained as follows. Equation (42) is equivalent to the linear objective function (41). Equation (43) represents the nonlinear wellbore model Eq. (1)–(7). Equation (44) represents the mixed integer linear mass, momentum and energy balances in the wells, manifold and separator nodes and is equivalent to Eq. (8)–(19). Equation (45) represents the mixed integer linear network logic constraints and is equivalent to Eq. (20)–(26). Equation (46) represents the well piecewise linear approximation and is equivalent to Eq. (27)–(30). Equation (47) represents the nonlinear momentum balance in the flowlines and is equivalent

to Eq. (31). Finally, Eq. (48) represents the integer logic relations associated with the ability of naturally flowing wells to flow and with the relevant well switches, and is equivalent to Eq. (32)–(40).

The mathematical programming formulation P includes: (i) binary variables, and (ii) nonlinear equations. Therefore, it belongs to the class of mixed integer nonlinear programming (MINLP) problems. There are two categories of binary variables: the one that is associated with the structure of the production network and the well switches ($\mathbf{y}^s, \mathbf{c}$), and a second that is used to impose the adjacency condition on *SOS*-type variables ($\mathbf{y}^a$). Moreover, the number of nonlinear equations is equal to the number of coning wells plus the number of flowlines.

The most popular methods for solving MINLP problems are those that proceed by solving a sequence of nonlinear (NLP) and mixed integer linear programs (MILP) problems. These include Generalized Benders decomposition (GBD, Geoffrion [3]) and Outer Approximation (OA, Kocis and Grossmann [5]). The disadvantage of GBD is that it may require a significant number of major iterations of the NLP subproblem and the MILP master problem. The major advantage of OA is that it typically requires fewer iterations to achieve a solution, since its MILP master problem contains more information than the GBD formulation. Conversely, because the OA master problem is richer, it is also more time-consuming to solve. A detailed review of the various MINLP algorithms has been published by Floudas [10].

This Chapter considers an approach based on Outer Approximation (OA), since it typically requires fewer iterations when compared to other MINLP techniques. Also, its modified version (Outer Aproximation/Augmented Penalty (OA/AP), (Viswanathan and Grossmann, 1990)) has been found to be capable of handling mild nonconvexities present in the MINLP problems.

Optimization Strategy

The first NLP subproblem of the OA/AP algorithm involves solving an optimization problem where the structure of the pipeline network is the one of the previous day. The lth NLP subproblem ($l > 1$) involves fixing the discrete decisions $\mathbf{y}^s$ and $\mathbf{c}$ to a given set of values ($\mathbf{y}^{s(l)}, \mathbf{c}^{(l)}$). Therefore, there is no need to introduce the logic constraints (45), (48) and hence the NLP subproblem (P) is equivalent to the well operation and gas lift allocation problem. It must be noted that the solution of the NLP subproblem provides a lower bound on the solution of the MINLP problem since the binary variables $\mathbf{y}^s$ and $\mathbf{c}$ are fixed to values that are not necessarily optimal.

The master problem is formulated from the linearization of the nonlinear constraints (43) and (47) at the solution points of the subproblems ($l = 1, \ldots, L$) and relaxation of them to inequalities using the sign of the Lagrange multipliers [17]. It is therefore, a MILP problem. The master problem provides (i) an upper bound to the MINLP problem and (ii) a new set of binary variables $\mathbf{y}^s$ and $\mathbf{c}$. The master MILP problem is as follows:

$$PM\colon \ \max \ \psi(q_{o,i}, q_{g,i}^{\text{inj}}) - (\mathbf{w}_l^p)^{\mathrm{T}}\mathbf{p}_l - (\mathbf{w}_l^q)^{\mathrm{T}}\mathbf{q}_l \quad (49)$$

subject to

$$\mathbf{T}^l \left\{ [\nabla_{\mathbf{x}_i} m_1(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l)\, \nabla_{\mathbf{q}_{p,i}} m_1(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l)] \begin{bmatrix} (\mathbf{x}_i - \mathbf{x}_i^l) \\ (\mathbf{q}_{p.i} - \mathbf{q}_{p.i}^l) \end{bmatrix} \right\} \le \mathbf{p}_l, \ \forall l = 1, \ldots, L \quad (50)$$

$$m_2(\mathbf{q}_{p,i}, \mathbf{q}_{p,i,m}, \mathbf{q}_{p,i,s}, \mathbf{q}_{p,s}, \mathbf{x}_i, \mathbf{x}_{i,m}, \mathbf{y}^s) \le 0 \quad (51)$$

$$m_3(\mathbf{q}_{p,i,m}, \mathbf{q}_{p,i,s}, \mathbf{y}^s) \le 0 \quad (52)$$

$$m_4(q_{o,i}, q_{g,i}^{\text{inj}}, q_{o,i,s}, q_{o,i,s}^{\max}, q_{g,i,s}^{\text{inj}} \mathbf{x}_{i,m}, \phi, \mathbf{y}^s, \mathbf{y}^a) = 0 \quad (53)$$

$$\mathbf{T}^l \left\{ [\nabla_{\mathbf{x}_m} m_5(\mathbf{x}_m^l, \mathbf{q}_{p,m}^l)\, \nabla_{\mathbf{q}_{p,s}} m_5(\mathbf{x}_m^l, \mathbf{q}_{p,m}^l)] \begin{bmatrix} \mathbf{x}_m - \mathbf{x}_m^l \\ \mathbf{q}_{p,m} - \mathbf{q}_{p,m}^l \end{bmatrix} \right\} \le \mathbf{q}_l \ , \ l = 1, \ldots, L \quad (54)$$

$$m_6(\mathbf{y}^s, \mathbf{c}) \le 0 \quad (55)$$

$$\sum_{n \in G^l} y_n^s - \sum_{n \in NG^l} y_n^s \le |G^l| \ , \ \forall l = 1, \ldots, L \quad (56)$$

where $\mathbf{w}_l^p$ and $\mathbf{w}_l^q$ are both vectors whose dimension is equal to the number of equations in (50) and (54), respectively. Each element of these vectors is a positive

scalar which is greater than the absolute value of the Lagrange multiplier v_j^l associated with the jth constraint in (50) and (54) at the lth iteration. Moreover, $\mathbf{T}^l$ is a diagonal matrix whose elements are defined as follows:

$$t_{jj}^l = \begin{cases} -1 & \text{if } v_j^l < 0 \\ 1 & \text{if } v_j^l > 0 \\ 0 & \text{if } v_j^l = 0\,. \end{cases} \tag{57}$$

Furthermore, $\mathbf{p}^l$ and $\mathbf{q}l$ are vectors whose elements are positive slack variables associated with each of the constraints in Eq. (50) and (54), respectively. Finally, the constraint (56) is known as an *integer cut* and is applied to ensure that any pipeline configuration that has already been considered is not selected again. The notion $|G^l|$ denotes the cardinality of the set G^l whose elements are all the structural binary variables ${y_n}^s$ that have a value of 1 at the lth iteration, while NG^l is the set of structural binary variables that have a value of zero at lth iteration.

The MINLP problem terminates when the difference of the best lower bound from the NLP subproblems ($\max_l LB^l$) and the current upper bound from the MILP problem (UB^l) are within a prespecified tolerance ε:

$$\frac{\max_l LB^l - UB^l}{UB^l} \leq \varepsilon \tag{58}$$

or when the MILP problem is integer-infeasible. The optimal solution is the one given by the best NLP subproblem.

However, the solution of the MILP problem on the full space of both structural and interpolation binary variables is computationally intensive, since the number of interpolation binary variables becomes very large as the number of wells increases. For instance, a problem with 10 gas lift wells involves about 300 interpolation binary variables. This motivates the need to reformulate the MILP problem (PM), so as to involve only structural and switching binary variables $\mathbf{y}^s$ and $\mathbf{c}$. As mentioned, the master MILP problem is constructed from (i) linearization of the nonlinear constraints, and (ii) relaxation of the nonlinear equality constraints using the sign of Lagrange multipliers. Fortunately, information for both is available from the solution of the NLP subproblem. Consider for instance the case of a gas lift well of type A (29), where the subscript i has been dropped for simplicity. At the optimal point, three adjacent μ coefficients are active (Williams, 1990). The active triplet is assumed to be $(\mu_{j,k}, \mu_{j+1,k}, \mu_{j,k+1})$, without loss of generality. Then the gas lift model (29) can be written as:

$$q_o = \mu_{j,k} q_{o,j,k}^d + \mu_{j+1,k} q_{o,j+1,k}^d + \mu_{j,k+1} q_{o,j,k+1}^d \tag{59a}$$

$$q_g^{\text{inj}} = \mu_{j,k} q_{g,k}^{d,\text{inj}} + \mu_{j+1,k} q_{g,k}^{d,\text{inj}} + \mu_{j,k+1} q_{g,k+1}^{d,\text{inj}} \tag{59b}$$

$$P_m = \mu_{j,k} P_j^{d,m} + \mu_{j+1,k} P_{j+1}^{d,m} + \mu_{j,k+1} P_j^{d,m} \tag{59c}$$

$$\mu_{j,k} + \mu_{j+1,k} + \mu_{j,k+1} = 1 \tag{59d}$$

By substituting Eq. (59d) into (59b) and (59c), both $\mu_{j+1,k}$ and $\mu_{j,k+1}$ are given by:

$$\mu_{j+1,k} = \frac{P_m - P_j^{d,m}}{P_{j+1}^{d,m} - P_j^{d,m}} \tag{60a}$$

$$\mu_{j,k} = \frac{q_g^{\text{inj}} - q_{g,k}^{d,\text{inj}}}{q_{g,k+1}^{d,\text{inj}} - q_{g,k}^{d,\text{inj}}}\,. \tag{60b}$$

Substituting Eq. (60a) and (60b) into (59a), the following equation is obtained:

$$\begin{aligned} q_o = q_{o,j,k} &+ \frac{q_{o,j+1,k}^d - q_{o,j,k}^d}{P_{j+1}^{d,m} - P_j^{d,m}} (P_m - P_j^{d,m}) \\ &+ \frac{q_{o,j,k+1}^d - q_{o,j,k}^d}{q_{g,k+1}^{d,\text{inj}} - q_{g,k}^{d,\text{inj}}} (q_g^{\text{inj}} - q_{g,k}^{\text{inj}})\,. \end{aligned} \tag{61}$$

Equation (61) is the linearization of the nonlinear gas lift well model, where the derivatives are calculated by forward finite difference formulae. If Eq. (61) replaces Eq. (59), a new NLP subproblem is obtained; then, by applying KKT conditions to both NLP subproblems, it is easy to prove that the Lagrange multiplier of Eq. (59a) is equal to the Lagrange multiplier of Eq. (61). Consequently, the active triplet of μ's is obtained from the solution of the NLP subproblem, along with the Lagrange multiplier; moreover, the new MILP master problem (PM') is formulated:

$$\begin{aligned} PM'\colon \max\ & \psi(q_{o,i}, q_{g,i}^{\text{inj}}) \\ & - (\mathbf{w}_l^p)^{\mathrm{T}} \mathbf{p}_l - (\mathbf{w}_l^q)^{\mathrm{T}} \mathbf{q}_l - (\mathbf{w}_l^r)^{\mathrm{T}} \mathbf{r}_l \end{aligned} \tag{62}$$

subject to

$$\mathbf{T}^l \left\{ [\nabla_{\mathbf{x}_i} m_1(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l), \nabla_{\mathbf{q}_{p,i}} m_1(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l)] \begin{bmatrix} (\mathbf{x}_i - \mathbf{x}_i^l) \\ (\mathbf{q}_{p.i} - \mathbf{q}_{p.i}^l) \end{bmatrix} \right\} \leq \mathbf{p}_l \,, \ \forall l = 1, \ldots, L \quad (63)$$

$$m_2(\mathbf{q}_{p,i}, \mathbf{q}_{p,i,m}, \mathbf{q}_{p,i,s}, \mathbf{q}_{p,s}, \mathbf{x}_i, \mathbf{x}_{i,m}, \mathbf{y}^s) \leq 0 \quad (64)$$

$$m_3(\mathbf{q}_{p,i,m}, \mathbf{q}_{p,i,s}, \mathbf{y}^s) \leq 0 \quad (65)$$

$$\mathbf{T}^l \left\{ [\nabla_{\mathbf{x}_i} m_4(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l) \nabla_{\mathbf{q}_{p,i}} m_4(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l)] \begin{bmatrix} (\mathbf{x}_i - \mathbf{x}_i^l) \\ (\mathbf{q}_{p.i} - \mathbf{q}_{p.i}^l) \end{bmatrix} \right\} \leq \mathbf{r}_l \,, \ \forall l = 1, \ldots, L \quad (66)$$

$$\mathbf{T}^l \left\{ [\nabla_{\mathbf{x}_m} m_5(\mathbf{x}_m^l, \mathbf{q}_{p,s}^l) \nabla_{\mathbf{q}_{p,s}} m_5(\mathbf{x}_m^l, \mathbf{q}_{p,s}^l)] \begin{bmatrix} \mathbf{x}_m - \mathbf{x}_m^l \\ \mathbf{q}_{p,s} - \mathbf{q}_{p,s}^l \end{bmatrix} \right\} \leq \mathbf{q}_l \,, \ l = 1, \ldots, L \quad (67)$$

$$m_6(\mathbf{y}^s, \mathbf{c}) \leq 0 \quad (68)$$

$$\sum_{n \in G^l} y_n^s - \sum_{n \in NG^l} y_n^s \leq |G^l| \,, \ \forall l = 1, \ldots, L \,. \quad (69)$$

The MILP problem (PM') involves only structural ($\mathbf{y}^s$) and switching ($\mathbf{c}$) binary variables. Figure 5 depicts the linearization of the nonlinear gas lift model, according to the foregoing analysis.

Solution Procedure

Based on the foregoing sections, the steps of the proposed MINLP optimization strategy for the well scheduling problem are formally presented as follows:

(1) Pre-processing step
   1. The reservoir information (productivity index, *GOR* and *WOR*) is updated, using a reservoir simulator.
   2. For each naturally flowing well, the manifold pressure and the well oil rate are discretized between a lower and an upper bound. Then, the well model is simulated for each pair of discrete points, and the momentum and energy balances are approximated with piecewise linear functions.
   3. For each gas lift well, the manifold pressure and the gas injection rate are discretized between a lower and an upper bound. Then, the well model is simulated for each pair of discrete points, and the momentum and energy balances are approximated with piecewise linear functions.
   4. If a naturally flowing well cannot flow towards a separator, then the corresponding logic constraint is incorporated into the formulation.
   5. If Vertical Flowing Tables are used, then the approximation of momentum and energy balances in the wells is simpler: there is no need for well simulation using each pair of discrete points, and simple interpolation calculations are used to approximate the momentum and energy balances in the wells.

(2) Processing step

This step involves the solution of the MINLP problem:

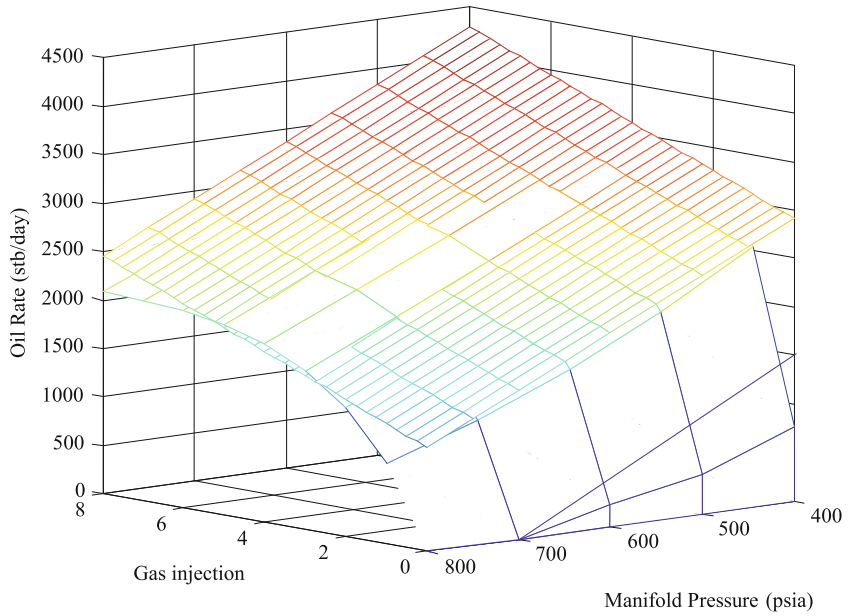
   1. Set the iteration counter at $l = 0$, and the upper bound at $UB^0 = +\infty$.
   2. Solve the NLP subproblem as a sequence of MILP problems, following the algorithm described by Kosmidis [18] 4 to obtain a lower bound (LB^l).
   3. Add linearizations and integer cuts cumulatively, and solve the MILP master problem (PM') and update the upper bound (UB^l).
   4. If $(UB^l - \max_l LB^l)/(LB^l) \leq \varepsilon$ or the MILP problem is integer-infeasible, then STOP. The optimal structure is the one which corresponds to the best lower bound $\max_l LB^l$. Else, set $l = l + 1$ and go to step (2).2.

(3) Post-processing step

For each well, fix the manifold pressure and the well oil flowrate and perform a a well simulation (based on the system of well equations) to calculate the precise well choke settings.

Example Problems

This section illustrates the performance of the proposed MINLP algorithm in two different example problems.



Mixed Integer Optimization in Well Scheduling, Figure 5
Linearization of a gas lift well model

The first example is a small production network which involves three wells connected to a manifold, and it is used to illustrate the economic impact of incorporating discrete decisions in the well scheduling problem. In the second example, the proposed MINLP optimization strategy is applied to a field consisting of 3 separators, 2 manifolds and 11 naturally flowing wells. To evaluate the economic benefits of the proposed MINLP optimization strategy, heuristic rules are also applied to the same problem for comparison. Finally, the proposed method is applied to an oil field, which consists of 22 (both naturally flowing and gas lift) wells.

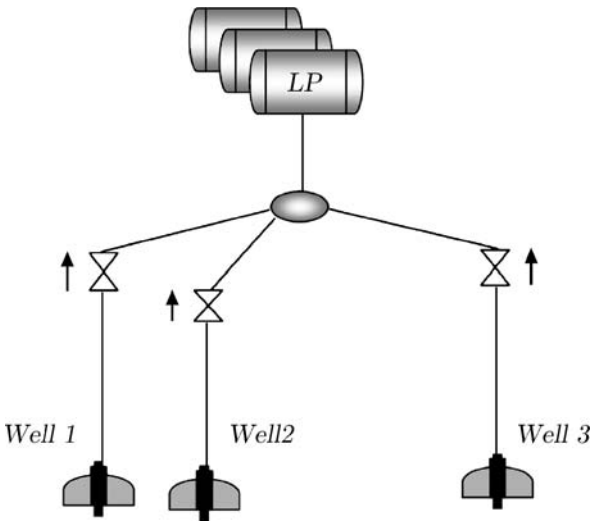
Example 1

The mathematical formulation and the solution procedure developed in this Chapter has been applied to the production network presented in Fig. 6. The well characteristics, separator pressures and capacities are given in Table 1 and 2, respectively. The problem has been formulated as an MINLP problem, where binary variables are used to model the operational status of each (closed or open) well. The MILP problems have been implemented in GAMS [5] and solved using CPLEX® as the MIP solver. The problem involves 3 binary variables, 26 interpolation binary variables and 81 constraints. Initially, the manifold pressure and the well oil flowrate are discretized to construct a piecewise linear approximation of the well model. Then, the initial structure (all wells tied to the manifold) has been evaluated by solving the corresponding NLP subproblem: the optimal solution has thus been determined equal to $LB^1 = 12010$ STB/day. The master MILP problem is then formulated and solved: the MILP problem solution generates a new production network structure, where well 1 is shut in. The new structure has then been evaluated in the NLP subproblem, and a new lower bound equal to $LB^2 = 12104.2$ STB/day has been determined. The algorithm terminates, since the MILP master problem is found to be integer-infeasible. Therefore, the optimal structure involves only wells 2 and 3 connected to the manifold, with their chokes fully open.

A typical heuristic rule for maximization of oil production states that the well chokes must be fully open for oil maximization. The application of this heuristic rule to this particular production network yields an oil production level equal to 11929.2 STB/day. The results from the application of heuristic rules and the proposed strategy are summarized in Table 3 and suggest that: (i) these heuristic rules may lead to suboptimal solutions, and (ii) an increase in oil production of 175 STB/day is observed when the proposed formal

Mixed Integer Optimization in Well Scheduling, Table 1
Well characteristics for a three-well production network (illustrative example)

Reservoir / pipeline parameters	Well 1	Well 2	Well 3	Flowline
Reservoir pressure (psia)	2370	4650	4250	
Productivity index (STB/psia day)	3.0	9.0	3.3	
GOR (SCF/STB)	5100	1900	1600	
WC	0.93	0.165	0.15	
Vertical length (ft)	8000	6000	7000	22000 ft
Horizontal length (ft)	6000	4000	3000	0
Diameter (in)	3 in	3 in	3 in	6 in
Roughness	0.0001	0.0001	0.001	0.0001
Flowrate upper bound (STB/day)	1600	10000	5300	
Flowrate lower bound (STB/day)	200	530	470	



Mixed Integer Optimization in Well Scheduling, Figure 6
Production network structure for Example 1 (illustrative example)

Mixed Integer Optimization in Well Scheduling, Table 2
Surface facilities: separator capacities for Example 1 (illustrative example)

Pressure (psia)	400
Oil Capacity (STB/day)	17000
Gas Capacity (MSCF/day)	33000
Water Capacity (STB/day)	22000

MINLP optimization technique is applied to the well scheduling problem. The above result can be explained by considering the interaction of wells that share a common flowline. This particular three-well network problem has a well with a very high water cut (well 1), as can

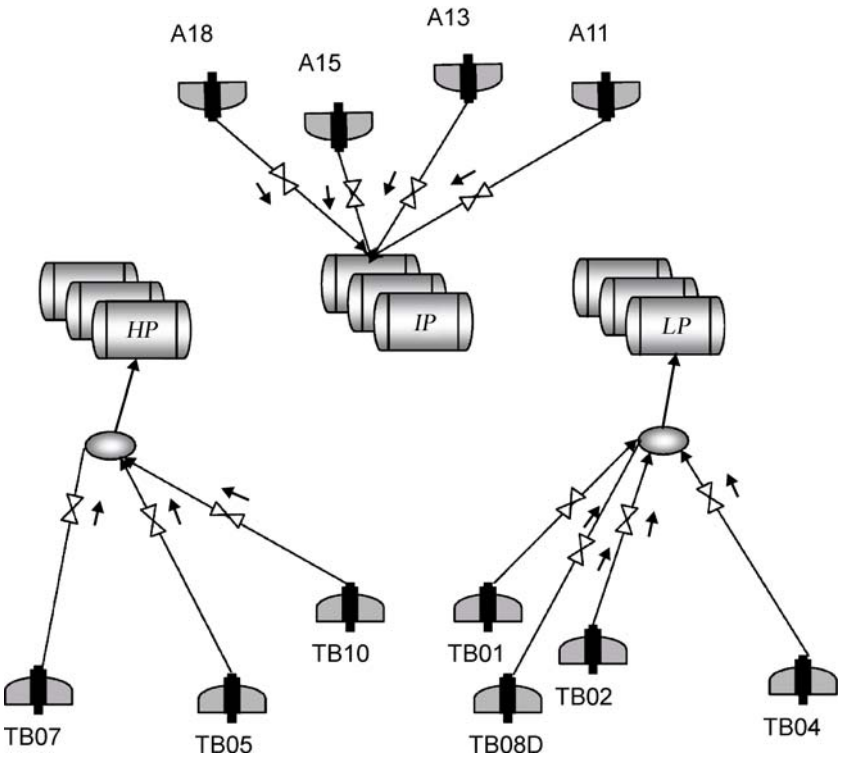
Mixed Integer Optimization in Well Scheduling, Table 3
Comparison of structure and oil production results: heuristics vs. optimization

Structure	Objective function (STB/day)
$(y_1, y_2, y_3) = (1, 1, 1)$	11929.2 (Heuristics)
$(y_1, y_2, y_3) = (0, 1, 1)$	12104.2 (Optimization)

be seen from Table 1: this results in increased back pressure in the manifold flowline, which restricts oil production from wells 2 and 3. By shutting in well 1, the pressure drop in the flowline is reduced: the increased production from wells 2 and 3 thus compensates losses in oil production by shutting in well 1.

Example 2

The proposed MINLP optimization strategy is also applied to an oil field that comprises 11 naturally flowing wells, 2 manifolds and 3 separators: this production network is depicted in Fig. 7. Two types of wells are considered: (i) *type A wells*, designated as TB01, TB02, TB04, TB05, TB07, TB08D and TB10, and (ii) *type B wells*, designated as A11, A13, A15 and A18. All these are naturally flowing wells and their well oil flowrate upper bounds are given in Table 4. The surface facilities consist of a high (HP), an intermediate (IP) and a low (LP) pressure separator, and the respective operating pressures and capacities are summarized in Table 5. Two case studies are considered: the first is a gas coning oil field, while the second is a water coning oil field. The well bore model is generated from a reservoir



Mixed Integer Optimization in Well Scheduling, Figure 7
Production network structure for Example 2a (initial pipeline configuration)

Mixed Integer Optimization in Well Scheduling, Table 4
Maximum flowrate values for wells

TB01	2300 STB/day	TB07	7000 STB/day	A11	4100 STB/day
TB02	4300 STB/day	TB08D	1000 STB/day	A13	4200 STB/day
TB04	2500 STB/day	TB10	7000 STB/day	A15	1800 STB/day
TB05	7500 STB/day			A18	1200 STB/day

Mixed Integer Optimization in Well Scheduling, Table 5
Operating pressures and capacities of separators for Example 2a (gas coning)

	HP separator		IP separator		LP separator	
	Capacity	Optimal	Capacity	Optimal	Capacity	Optimal
Pressure (psia)	1235		460		165	
Oil (STB/day)	15000	12541.9	10000	7191.2	10000	9584.1
Gas (MMSCF/day)	24000	24000	18000	18000	18000	18000
Water (STB/day)	2000	1236.5	4000	2057.8	8000	8000
Total oil production	29317					
NLP (LB)	28567	28910	29317	28735	29020	
MILP (UB)	32104	31580	31210	30920	30067	

Mixed Integer Optimization in Well Scheduling, Table 6
Optimal well flowrates by MINLP optimization (Example 2a)

Well	Q_o (STB/day)	Q_g (MSCF/day)	Q_w (STB/day)
TB01	2300	5574.83	3552.06
TB05	5685.67	10806.03	1123.67
TB08D	Shut in		
TB02	632.906	2223.614	0.0
TB04	836.88	1971.94	2055.46
TB07	6647.67	12543.21	109.594
TB10	5814.3	8229.61	2392.47
A11	4100	9151.44	71.62
A13	1291.18	3930.83	1553.64
A15	1800	4917.72	432.51
A18	208.552	650.76	3.213

simulator using a coning model. Details about this second example (oil flowrate as a function of bottomhole pressure, *GOR* and *WC* for both cases) are presented by Kosmidis [18].

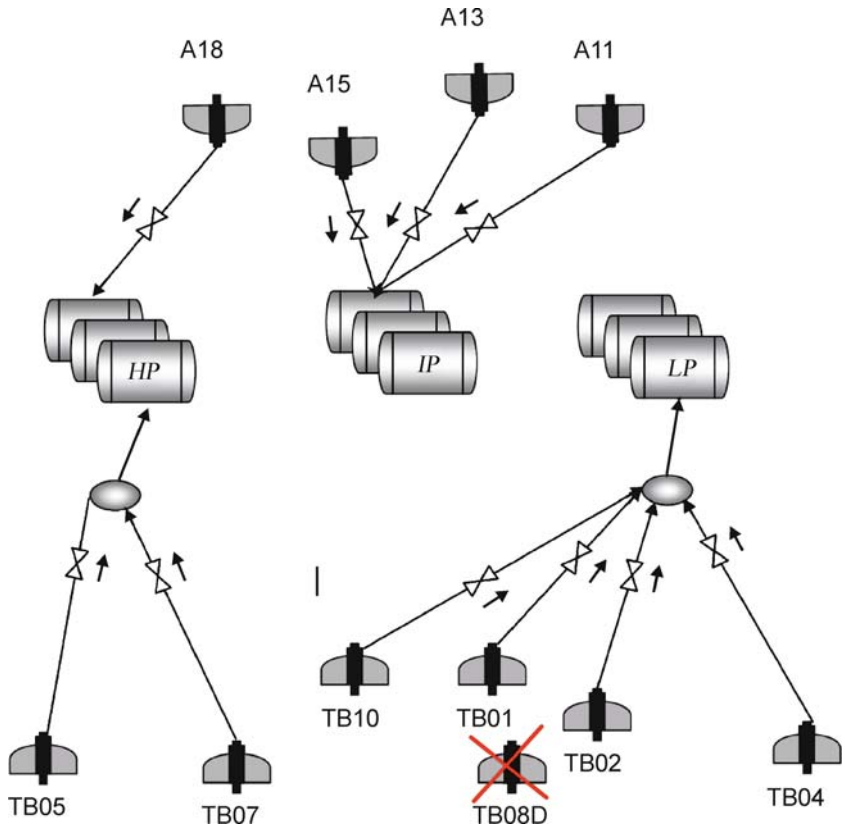
Example 2a (gas coning problem) The initial structure of the production network is shown in Fig. 7, and five (5) well interconnection changes are allowed for wells of type A and type B. The MINLP optimization problem involves 89 binary variables, 260 interpolation binary variables, 924 continuous variables, 1082 constraints and the objective is the maximization of oil production. The optimization requires 5 OA/AP iterations and the total oil production is 29317.2 STB/day; the optimal production network structure is presented in Fig. 8. Table 5 summarizes the amount of oil, gas and water in the separators and the convergence history of the MINLP algorithm; the individual well fluid flowrates are reported in Table 6. A remarkable observation is that the gas capacity of all separators is fully utilized at the optimal operating point, as can be observed from the results of Table 5.

Example 2b (water coning problem) This problem is again solved following the proposed MINLP optimization strategy. The initial structure of the field is presented in Fig. 8; the maximum number of allowable well interconnection changes is seven (7) for wells of type A and type B. The MINLP problem converges in 6 OA/AP iterations and the optimal structure is depicted in Fig. 9. Table 7 presents the amount of oil, gas and water in the separators and the convergence history of the MINLP problem, while well fluid flowrates are reported in Table 8. The results of Table 7 suggest that the production bottleneck of the oil field is the water separator capacity, and the proposed MINLP optimization method manages to allocate and operate the wells in such a way that the available water separator capacity is almost fully utilized. The manifold flowline that is connected to the HP separator in the initial structure (Fig. 7) is reallocated to the IP separator, since the latter has a larger water capacity compared to the HP separator (Table 7).

Heuristic Rules vs. Optimization Examples 2a and 2b are also both solved with heuristic rules, by applying the following procedure:
STEP 0. Consider an initial pipeline structure identical to that of the previous day.
STEP 1. Set the chokes fully open and solve the corresponding production network problem.
STEP 2. If some of the resulting well flowrates from Step 1 violate their upper bounds, then choke back these wells until the respective upper bounds are satisfied.
STEP 3. The following two heuristic rules are applied sequentially (one well at a time):

(i) Choke back the well according to the following heuristic rule: if gas and/or water capacity constraints are violated, then choke back the well with the highest *GOR* and/or *WC*, respectively, until the capacity constraints are satisfied. Terminate or else go to Step 3 (ii).
(ii) Allocate high *GOR* wells to the *HP* separator and high *WC* wells to the LP separator, and go back to Step 1.

It must be noted that: (i) the heuristic rules are applied sequentially, and (ii) the termination criterion is based on the satisfaction of the operator. The results from the application of heuristic rules are based on repetitively applying the procedure described, until the maximum number of allowable interconnection changes is reached. The production network structures resulting from the application of heuristic rules in Examples 2a and 2b are depicted in Fig. 10 and 11, respectively. Tables 9 and 10 summarize the results derived from both MINLP optimization and heuristic strategies. The comparison clearly demonstrates the economic bene-

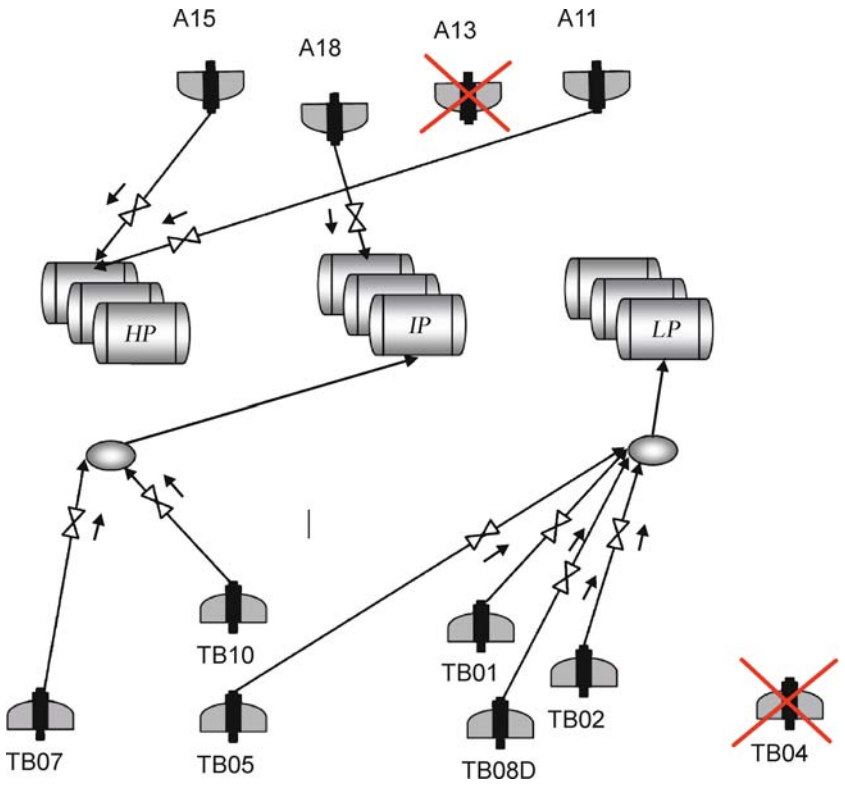


Mixed Integer Optimization in Well Scheduling, Figure 8
Optimal production network structure by MINLP optimization (Example 2a)

Mixed Integer Optimization in Well Scheduling, Table 7
Optimal surface separator capacities by MINLP optimization (Example 2b)

	HP separator		IP separator		LP separator	
	Capacity	Optimal	Capacity	Optimal	Capacity	Optimal
Pressure (psia)	1235		460		165	
Oil (STB/day)	15000	5900	10000	9714.5	10000	9684.4
Gas (MMSCF/day)	24000	14069.2	18000	18000	18000	18000
Water (STB/day)	2000	1926.9	4000	4000	8000	8000
Total oil production	25299					
NLP (LB)	23210	24820	25170	25299	24870	24320
MILP (UB)	29102	28670	27332	26703	26209	26023

fits from the application of the proposed MINLP optimization strategy, which in both examples achieves of up to 10% in oil production. There are many reasons which can explain these superior results: (i) the simplistic nature of heuristic rules, which consider only the individual well *GOR* and *WC*, and neglect other parameters (e. g. productivity index, pipeline length and diameter), (ii) heuristic strategies do not account directly for system interactions, which become important when the wells share a common flowline, and (iii) heuristic methods often have ad hoc or unclear termination criteria.



Mixed Integer Optimization in Well Scheduling, Figure 9
Optimal production network structure by MINLP optimization (Example 2b)

Mixed Integer Optimization in Well Scheduling, Table 8
Optimal well flowrates by MINLP optimization (Example 2b)

	Q_o (STB/day)	Q_g (MSCF/day)	Q_w (STB/day)
TB01	2300	2125.14	3552.1
TB05	5200	10429.3	1874.8
TB08D	757.308	2723.6	375.1
TB02	1427.04	2721.95	2198.01
TB04	Shut in		
TB07	6864.4	12937.2	1893.7
TB10	1691.2	2359.6	1573.4
A11	4100	9151.4	1494.4
A13	Shut in		
A15	1800	4917.7	432.5
A18	1158.8	2703.2	532.9

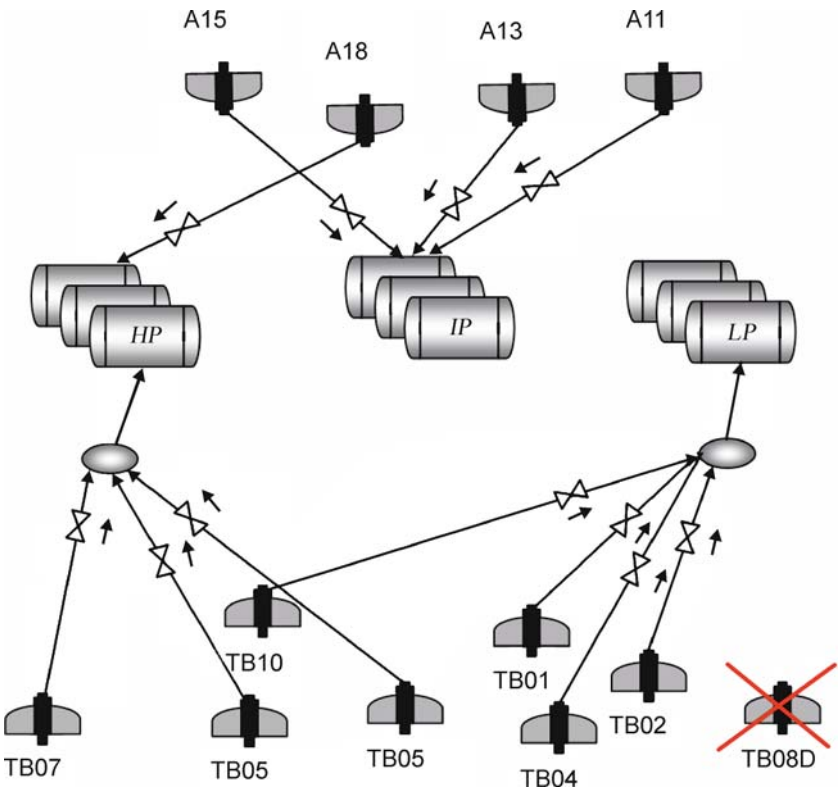
Integration of Reservoir Multiphase Flow Simulation and Optimization

Dynamic oil and gas production systems simulation and optimization is a research trend with a potential to meet the challenges faced by the international oil and gas industry, as has been already demonstrated in a wide variety of publications in the open literature. The multiphase flow in reservoirs and wells governs fuel transport and production, but is mostly handled by algebraic approximations in modern optimization applications: true reservoir state variable profiles (initial/boundary conditions) are generally not known. Nevertheless, oil reservoirs, wells, pipelines, manifolds and surface facilities are all equally important elements of a spatially and temporally distributed complex system, and the potential contribution of CFD methods has not been fully explored so far, even though it is generally recognized that computing accurate reservoir and well state variable profiles can be extremely useful for optimization. This section discusses a strategy for interfacing reservoir simulation (ECLIPSE®) with equation-oriented process optimization (gPROMS®) and presents a relevant application [13].

The complex multiphase flow in oil production fields is of paramount importance. Despite intensive

Mixed Integer Optimization in Well Scheduling, Table 9
Comparison of results: MINLP optimization vs. heuristics (Example 2a)

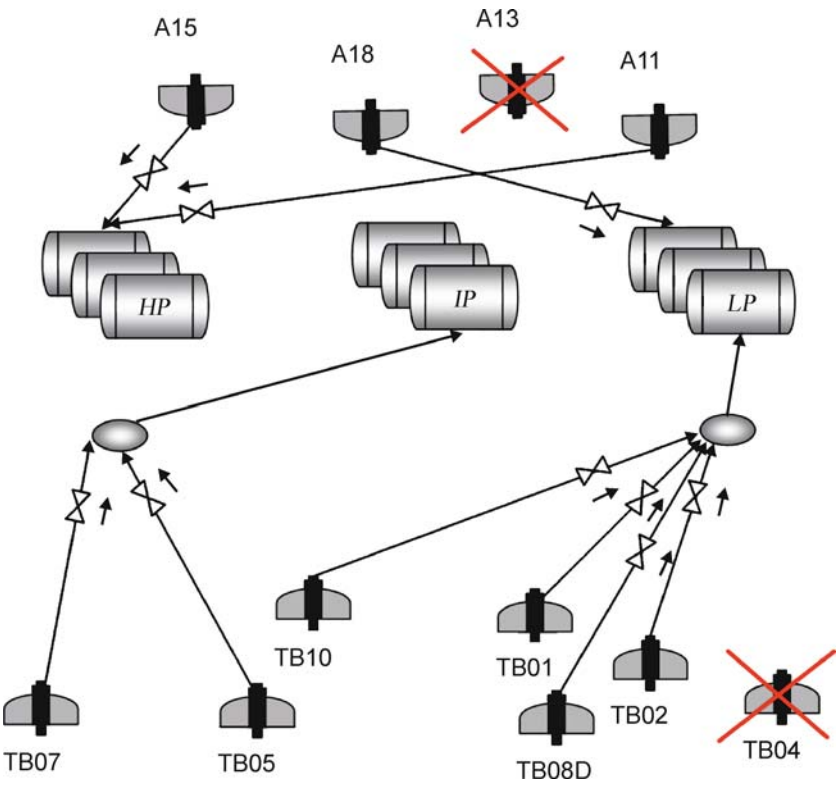
Example 2a (High GORs)	Capacity	Optimization	Heuristics
	Q_o (STB/day)	Q_o (STB/day)	Q_o (STB/day)
LP	10000	9584.15	9004.296
IP	10000	7191.18	7191.186
HP	15000	12541.894	12321.138
Total		29317.2	28516.6
Benefit (STB/day)		800.5 (+2.3%)	



Mixed Integer Optimization in Well Scheduling, Figure 10
Heuristic production network structure (Example 2a)

Mixed Integer Optimization in Well Scheduling, Table 10
Comparison of results: MINLP optimization vs. heuristics (Example 2b)

Example 2b (High WCs)	Capacity	Optimization	Heuristics
	Q_o (STB/day)	Q_o (STB/day)	Q_o (STB/day)
LP	10000	9684.4	7424.407
IP	10000	9714.5	9311.058
HP	15000	5900	5900
Total		25298.8	22635.5
Benefit (STB/day)		2663.3 (+11.8%)	



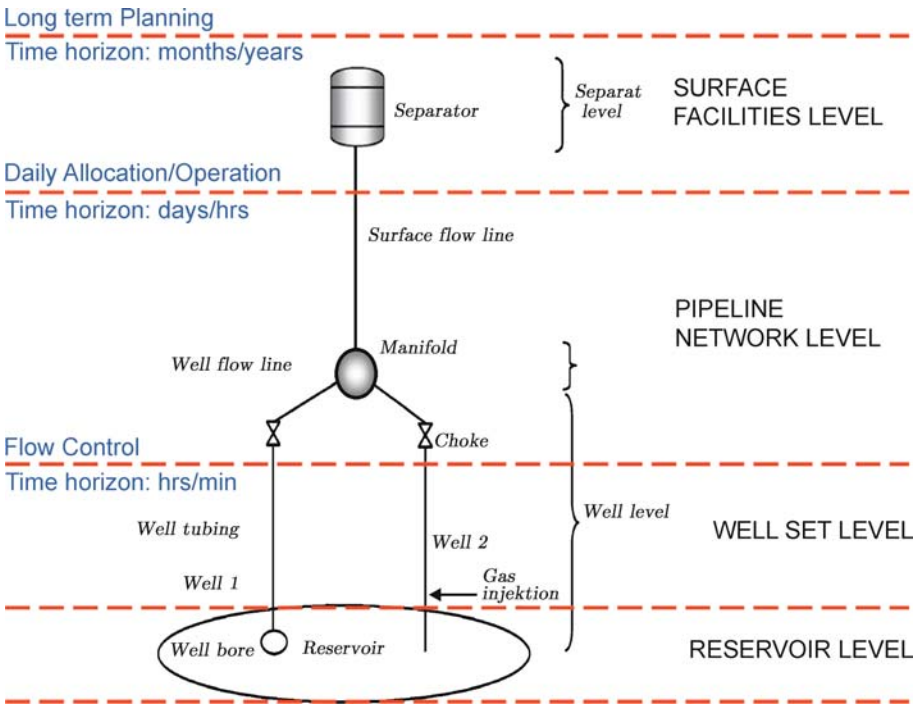
Mixed Integer Optimization in Well Scheduling, Figure 11
Heuristic production network structure (Example 2b)

experimentation and extensive CFD simulations towards improved understanding of flow and phase distribution, commercial optimization applications have not benefited adequately from accurate sub-surface multiphase CFD modeling, and knowledge from field data is not readily implementable in commercial software. Model integration can enable the employment of two-phase reservoir CFD simulation, towards enhanced oil or gas production from depleted or gas-rich reserves, respectively.

The concept of integrated modeling and optimization of oil and gas production treats oil reservoirs, wells and surface facilities as a single (albeit multiscale) system, and focuses on computing accurate reservoir state variable profiles (as initial/boundary conditions). The upper-level optimization can thus benefit from the low-level reservoir simulation of oil and gas flow, yielding flow control settings and production resource allocations. The components of this system are tightly interconnected (well operation, allocation of wells to headers and manifolds, gas lift allocation, control of unstable gas lift wells). These are only some of the problems that can be addressed via this unified framework. Figure 12 presents the concept of integrated modeling of oil and gas production systems.

Literature Review and Challenges for Integrated Modeling and Optimization

A number of scientific publications address modeling and simulation of oil extraction: they either focus on accurate reservoir simulation, without optimization considerations [15,22], or on optimal well planning and operations, with reduced [8,23,29,32] or absent [28,30] reservoir models. A recent paper [16] is the only considering a three-dimensional field topology (without additional flow constraints) for well placement optimization. Computational Field Dynamics (CFD) is a powerful technology, suitable for studying the dynamic behavior of reservoirs for efficient field operation [2]. The MINLP formulation for oilfield production optimization of Kosmidis [19] uses detailed well



Mixed Integer Optimization in Well Scheduling, Figure 12
Integrated modeling concept for oil and gas production systems optimization: illustration of the hierarchy of levels and all production circuit elements

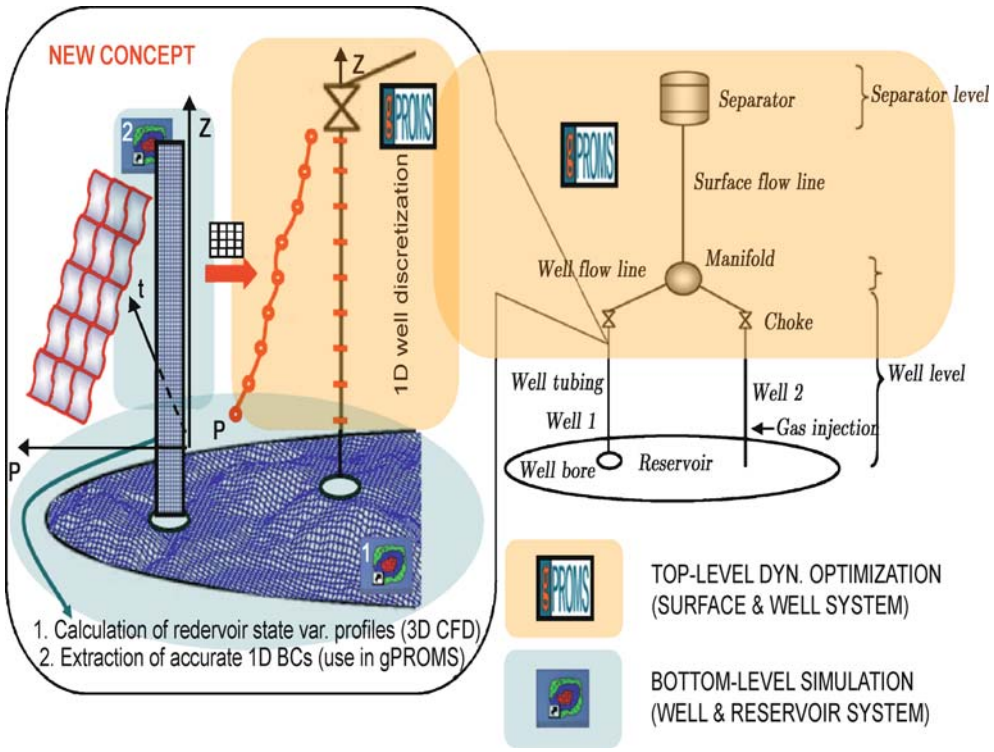
models and serves as a starting point in the case examined in this section. Therein, the nonlinear reservoir behavior, the multiphase flow in pipelines, and surface capacity constraints are *all* considered (multiphase flow is handled by DAE systems, which in turn comprise ODEs for flow equations and algebraics for phys. properties). The model uses a degrees-of-freedom analysis and well bounding, but most importantly approximates each well model with piecewise linear functions (via data preprocessing).

Here, explicit reservoir flow simulation via a dynamic reservoir simulator (ECLIPSE®) is combined with an equation-oriented process optimizer (gPROMS®), towards integrated modeling and optimization of a literature problem 13. An asynchronous fashion is employed: the first step is the calculation of state variable profiles from a detailed description of the production system (reservoir) via ECLIPSE®. This is possible by rigorously simulating the multiphase flow within the reservoir, with real-world physical properties (whose extraction is laborious [7]). These dynamic state variable profiles (pressure, oil, gas and water saturation, flows) are a lot more accurate than piecewise linear approximations [18], serving as initial conditions for the higher-level dynamic optimization model (within gPROMS®). Crucially, these profiles constitute major sources of uncertainty in simplified models. Considering the oil and gas pressure drop evolution within the reservoir and along the wells, one can solve single-period or multi-period dynamic optimization problems that yield superior optima, because piecewise linear pressure underestimation is avoided. While integrating different levels (sub-surface elements and surface facilities – Fig. 12) is vital, interfacing CFD simulation with MINLP optimization is here pursued in an asynchronous fashion (given the computational burden for CFD nested within MINLP).

The concept of integrated modeling and optimization is illustrated in Fig. 13.

Problem Definition and Model Formulation

Dynamic CFD modeling for explicit multiphase flow simulation in reservoirs and wells comprises a large



Mixed Integer Optimization in Well Scheduling, Figure 13
Integrated modeling and optimization of oil and gas production systems: illustration of the explicit consideration of multiphase flow within reservoirs and wells

number of conservation laws and constitutive equations for closure: Table 1 presents only the most important ones, which are implemented in ECLIPSE®. The *black-oil model* [25] is adopted in this study, to manage complexity. More complicated, *compositional models* are widely applied [2], accounting explicitly for different hydrocarbon real- or pseudo-species concentrations. A black-oil model allows for multiphase simulation via only 3 phases (oil, water, gas):

Multiphase flow CFD model equations (Nomenclature [19]):

Oil:

$$\nabla\left[\frac{k\,k_{ro}}{\mu_o B_o}\nabla(P_o+\rho g h)\right]+q_o=\frac{\partial}{\partial t}\left(\phi\frac{S_o}{B_o}\right) \quad (70)$$

Water:

$$\nabla\left[\frac{k\,k_{rw}}{\mu_w B_w}\nabla(P_w+\rho g h)\right]+q_w=\frac{\partial}{\partial t}\left(\phi\frac{S_w}{B_w}\right) \quad (71)$$

Gas:

$$\nabla\left[\frac{k\,k_{rg}}{\mu_g B_g}\nabla(P_g+\rho g h)\right] + \nabla\left[R_s\frac{k\,k_{ro}}{\mu_o B_o}\times\nabla(P_o+\rho g h)\right]+q_g = \frac{\partial}{\partial t}\left(\phi\frac{S_g}{B_g}+R_s\frac{\phi S_o}{B_o}\right) \quad (72)$$

Total pressure gradient:

$$\frac{\mathrm{d}P}{\mathrm{d}x}=-g\rho_m(x)\sin(\theta)-\frac{\tau_w(x)S}{A} \quad (73)$$

Capillary pressure (oil/gas):

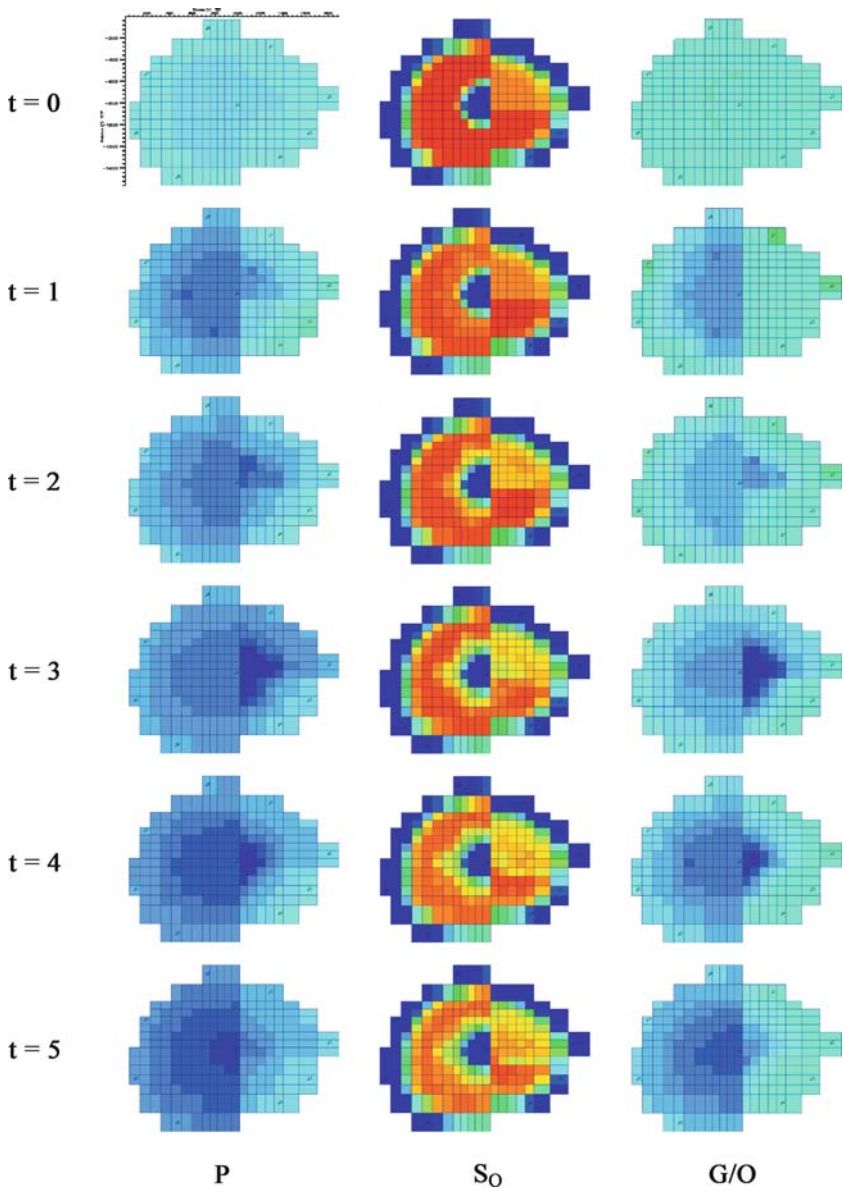
$$P_{\mathrm{cog}}(S_o,S_g)=P_o-P_g \quad (74)$$

Capillary pressure (oil/water):

$$P_{\mathrm{cow}}(S_o,S_w)=P_o-P_w \quad (75)$$

Multiphase mixture saturation:

$$S_o+S_w+S_g=1 \quad (76)$$



Mixed Integer Optimization in Well Scheduling, Figure 14
Temporal evolution of pressure, oil saturation and gas/oil ratio in an oilfield: the gradual depletion of oil in reservoirs is explicitly considered for optimization (t:yr)

Multiphase mixture density:

$$\rho_m(x) = \rho_l(x)E_l(x) + \rho_g(x)E_g(x) \tag{77}$$

Multiphase mixture viscosity:

$$\mu_m(x) = \mu_l(x)E_l(x) + \mu_g(x)E_g(x) \tag{78}$$

Multiphase mixture sup. velocity:

$$U_m(x) = \frac{\rho_l(x)}{\rho_m(x)}U_{sl}(x) + \frac{\rho_g(x)}{\rho_m(x)}U_{sg}(x) \tag{79}$$

Multiphase mixture holdup closure:

$$E_g(x) + E_l(x) = 1 \tag{80}$$

Drift flux model (gas holdup):

$$E_g = f_d(U_{sl}, U_{sg}, \text{ mixture properties}) \tag{81}$$

Choke model (for well & valve i):

$$q_{L,i} = f_c(d_i, P_i(x_{ch}^-), P_i(x_{ch}^+), c_i, q_{g,i}, q_{w,i}) \tag{82}$$

Mixed Integer Optimization in Well Scheduling, Table 11
Oil production optimization by explicit CFD simulation boundary conditions

Example 2a, Kosmidis et al.[20]	Total capacity	Via performance indices	With explicit reservoir simulation
Oil production (STB/day)	35000	29317.2	30193.7 (+2.9%)
Gas production (MSCF/day)	60000	60000	60000
Water production (STB/day)	14000	11294.3	11720.1 (+3.8%)

Choke setting (for well & valve i):

$$c_i = \max(c_c, P_i(x_{ch}^-), P_i(x_{ch}^+)) \tag{83}$$

Performance (flow vs. pressure):

$$q_{j,i} = f_j(P_{\mathrm{wf},j,i}),\ \forall i \in I,\ \forall j \in \{o, w, g\}\ . \tag{84}$$

Reduced (1D) multiphase flow balances were solved using a fully implicit formulation and Newton's method, but only for the wells and not for the reservoir [18]. The present section uses: (a) explicit reservoir and well 3D multiphase flow simulation, (b) elimination of Eq. (84) (performance relations/preprocessing obsolete due to CFD), (c) CFD profiles as initial conditions (asynchronous fashion) for dynamic optimization. The MINLP optimization objective (maximize oil production) and model structure is adopted from the literature [20] via a gPROMS®–SLP implementation. Adopting an SQP strategy can increase robustness as well as computational complexity.

Reservoir Multiphase Flow Simulation Results

Dynamic multiphase flow simulation results (ECLIPSE®) are presented in Fig. 14.

Oil Production Optimization Results

Dynamic optimization via explicit CFD simulation of a particular oil field problem can improve on results from MINLP optimization: the comparison is presented in Table 11.

Conclusions

A novel MINLP optimization formulation for the well scheduling problem has been proposed in this Chapter: the optimal connectivity of wells to manifolds and separators is treated simultaneously with the optimal well operation and gas lift allocation. The algorithm avoids examining infeasible connections of wells to manifolds or separators by incorporating appropriate integer cuts in the formulation: these, along with the incorporation of operational logic constraints pertinent to the maximum number of well switches, lead to satisfactory computational performance: convergence has been achieved in less then 6 iterations in all cases examined. The business value of the new MINLP formulation has been investigated by comparing the proposed method with established heuristic rules, and an increase of up to 10% in oil production has been observed for the cases studied [18].

The combination of dynamic multiphase CFD simulation and MINLP optimization has the potential to yield improved solutions towards efficiently maximizing oil production. This Chapter also addresses integrated oilfield modeling and optimization, treating the oil reservoirs, wells and surface facilities as a combined system: most importantly, it stresses the benefit of computing accurate state variable profiles for reservoirs via CFD. Explicit CFD simulations via a dynamic reservoir simulator (ECLIPSE®, Schlumberger) are combined with equation-oriented process optimization software (gPROMS®, PSE): the key idea is to use reduced-order copies of CFD profiles for dynamic optimization. The literature problem solved shows that explicit use of CFD results in optimization yields improved optima at additional cost (CPU cost *and* cost for efficient separation of the additional water; the percentage difference is due to accurate reservoir simulation). These can also be evaluated systematically for larger case studies under various conditions [14].

References

1. Addington DV (1981) An approach to gas coning correlations for a large grid cell reservoir simulator. J Pet Eng 33:2267–2274
2. Aziz K, Settari A (1979) Petroleum Reservoir Simulation. Appl Sci Publ, London
3. Benders JF (1962) Partitioning procedures for solving mixed variables programming problems. Numer Math 4:238–252

4. Brill JP, Mukherjee H (1999) Multiphase Flow in Wells. SPE Monograph, Henry L Doherty Series (17), Richardson
5. Brooke A, Kenderick D, Meeraus A (1992) GAMS: A User Guide. Scientific Press, Redwood City
6. Duran MA, Grossmann IE (1986) A mixed integer nonlinear programming algorithm for process system synthesis. AIChE J 32:592–606
7. Economides M, Hill AD, Ehlig-Economides CA (1994) Petroleum Production Systems. Prentice Hall, Englewood Cliffs
8. Fang WY, Lo KK (1996) A generalized well management scheme for reservoir simulation. Soc Pet Eng Pap SPE 29124:116–120
9. Fentor DJ (1984) A multilevel well management program for modeling offshore fields. Soc Pet Eng Pap SPE 12964:75–82
10. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, Oxford
11. Geoffrion AM (1972) Generalized Benders decomposition. Optim Theory App 10:237–260
12. GeoQuest (2000) Eclipse 300 Technical Description 2000A. GeoQuest, Schlumberger Information Solutions (SIS), Houston
13. Gerogiorgis DI, Georgiadis MC, Bowen G, Pantelides CC, Pistikopoulos EN (2006) Dynamic oil and gas production optimization via explicit reservoir simulation. In: Marquardt W, Pantelides C (eds) Proceedings of ESCAPE-16/PSE2006. Elsevier, Amsterdam, pp 179–184
14. Gerogiorgis DI, Pistikopoulos EN (2006) Wells-to-tankers: Dynamic oil and gas production optimization via explicit reservoir CFD simulation. In: Proceedings of the PRES-CHISA 2006 Conference (CD) Prague
15. Hepguler G, Barua S, Bard W (1997) Integration of field surface and production network with a reservoir simulator. Soc Pet Eng Pap SPE 38937:88–93
16. Ierapetritou MG, Floudas CA, Vasantharajan S, Cullick AS (1999) Optimal location of vertical wells: decomposition approach. AIChE J 45:844–859
17. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flowsheets. Ind Eng Chem Res 45:1869–1880
18. Kosmidis VD (2003) Integrated Oil and Gas Production Optimization. PhD Thesis, Department of Chemical Engineering, Imperial College London
19. Kosmidis VD, Perkins JD, Pistikopoulos EN (2004) Optimization of well oil rate allocations in petroleum fields. Ind Eng Chem Res 43:3513–3527
20. Kosmidis VD, Perkins JD, Pistikopoulos EN (2005) A mixed integer optimization formulation for the well scheduling problem on petroleum fields. Comput Chem Eng 29:1523–1541
21. Litvak ML, Darlow BL (1995) Surface network and well tubinghead pressure constraints in compositional simulation. Soc Pet Eng Pap SPE 29125:325–336
22. Litvak ML, Clark AJ, Fairchield JW, Fossum MP, McDonald CJ, Wood ARO (1997) Integration of Prudhoe Bay surface pipeline network and full field reservoir models. Soc Pet Eng Pap SPE 38885:435–443
23. Lo KK (1992) Optimum lift-gas allocations under multiple production constraints. Soc Pet Eng Pap SPE 26017
24. Lo KK, Starley GP, Holden CW (1995) Application of linear programming to reservoir development evaluations. Soc Pet Eng Pap SPE 26637:52–58
25. Peaceman DW (1977) Fundamentals of Numerical Reservoir Simulation. Elsevier, Amsterdam
26. Process Systems Enterprise (PSE) Ltd (2000) gPROMS® Advanced User Guide. PSE, London
27. Pucknell JK, Mason JNE, Vervest EG (1993) An evaluation of recent mechanistic models of multiphase flow for predicting pressure drops in oil and gas wells. Soc Pet Eng Pap SPE 26682:1–11
28. Saputelli L, Malki H, Canelon J, Nikolaou M (2002) A critical overview of artificial neural network applications in the context of continuous oil field optimization. Soc Pet Eng Pap SPE 77703:1–11
29. Stewart G, Clark AC, McBride SA (2001) Field-wide production optimization. Soc Pet Eng Pap SPE 59459:1–10
30. Van den Heever SA, Grossmann IE (2000) An iterative aggregation/disaggregation approach for solution of an MINLP oilfield infrastructure planning model. Ind Eng Chem Res 39:1955–1971
31. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer approximation method for MINLP optimization. Comput Chem Eng 14:769–782
32. Wang P, Litvak M, Aziz K (2002) Optimization of production from mature fields. Proceedings of the 17th World Petroleum Congress Rio de Janeiro, 2002
33. Williams HP (1990) Model Building in Mathematical Programming. Wiley, Chichester

Mixed Integer Programming/Constraint Programming Hybrid Methods

Abderrahmane Aggoun[1], Christos Maravelias[2], Alkis Vazacopoulos[3]
[1] KLS-OPTIM, Villebon sur Yvette, France
[2] University of Wisconsin – Madison, Madison, USA
[3] Dash Optimization, Englewood Cliffs, USA

Article Outline

Background
Mixed-Integer Programming
Constraint Programming
Methods

Applications
Conclusions
References

Background

Mixed-Integer Programming

Mixed-Integer Programming (MIP) [5] emerged in the mid 1950s as an extension of Linear Programming (LP) to include both integer and continuous variables. It was developed to address a variety of problems (facility location, scheduling, design of plants and networks, etc.) where discrete decisions needed to be made. There are two main algorithms used to solve MIP models: branch-and-bound [5,31] and cutting planes. When the two solution methods are combined we have the branch-and-cut algorithm, where cutting planes are added until either an integral solution is found or it becomes impossible or too expensive to find another cutting plane. In the latter case, a traditional branch operation is performed and the search for cutting planes continues for the subproblems. Balas developed an algorithm for 0–1 problems to obtain dual bounds and check primal feasibility [3]. The idea of cutting planes was originally proposed by Gomory in [17], and a cutting plane algorithm was presented by Gomory in [18]. A general procedure for bounded programs was proposed by Chvatal in [13].

The results of Edmonds and Fulkerson in the late 1960s led several authors to propose other, specific types of cutting planes: cover inequalities [4,5], flow cover inequalities [4], and GUB constraints [51]. Due to the incorporation of these theoretical results, the efficiency of the commercial solvers has greatly been enhanced during the last decade. Advances in preprocessing, more sophisticated branching and node selection rules, as well as the use of primal heuristics have also contributed to the improvement of MIP solvers. Special techniques have also been used extensively for the solution of MIP problems, when the set of constraints exhibits a special structure. The most popular of these schemes are Benders decomposition [9] and Lagrangean relaxation [16,19]. More information on MIP can be found in [38], and [52], while an exposition in recent progress in solution techniques for MIP models can be found in [30].

Constraint Programming

Constraint Programming [24,47] is a relatively new modeling and solution paradigm that was originally developed to solve feasibility problems, but it has been extended to solve optimization problems as well. Constraint Programming (CP) has emerged as a very interesting sub-field of logic programming that aims at combining the declarative aspect of logic programming and constraint solving in an efficient problem solving environment [29]. Optimization problems in Constraint Programming are solved as Constraint Satisfaction Problems (CSP), where we have a set of variables, a set of possible values for each variable (domain) and a set of constraints among the variables. Constraints are solved with methods and advanced techniques originating in various areas, from Artificial Intelligence, Operations Research and Discrete Mathematics. The computation domains handled by CP solvers are quite diverse, including Boolean algebra, linear programming, finite domains, and list and set handling. Successful industrial applications were implemented with CP solvers over finite domains in production planning, scheduling and resource applications [44]. Finite domain constraints are expressed over variables, which range over a finite set of possible values. Constraints may be arithmetic, symbolic or global constraints [1] that have been developed to efficiently model and solve complex problems. A CP program is usually structured as follows: (1) declaration of decision variables, (2) constraints and (3) the enumeration/optimization. The question to be answered is as follows: Is there an assignment of values to variables that satisfy all constraints? Constraint Programming is very expressive as continuous, integer, as well as boolean variables are permitted and moreover, variables can be indexed by other variables. A CP problem can be seen as a network of constraints. As soon as some information becomes available at some points in this network, constraints are invoked to check consistency and to remove inconsistent values by applying efficient handling methods. The new domain reductions are propagated through the network. The solution of CP models is based on performing constraint propagation at each node by reducing the domains of the variables. If an empty domain is found the node is pruned. Branching is performed whenever a domain of an integer, binary or boolean variable has more than one ele-

ment, or when the bounds of the domain of a continuous variable do not lie within a tolerance. Whenever a solution is found, or a domain of a variable is reduced, new constraints are added. The search terminates when no further nodes must be examined.

The effectiveness of CP depends on the propagation mechanism behind the constraints. Thus, even though many constructs and constraints are available, not all of them have efficient propagation mechanisms. For some problems, such as scheduling, propagation mechanisms have been proven to be very effective. Some of the most common propagation rules for scheduling are the "time-table" constraint [32], the "disjunctive-constraint" propagation [6,45], the "edge-finding" [12,39] and the "not-first, not-last" [7]. Constrained-based scheduling algorithms can be found in [8]. General information on CP can be found in [24,27,36,47].

Methods

Several authors have compared MIP and CP based approaches for solving a variety of problems [21,26], and the main findings are as follows:

- MIP based techniques are very efficient when the LP relaxation is tight and the models have a structure that can be effectively exploited.
- CP based techniques are better for highly constrained discrete optimization problems.

Since the two approaches appear to have complementary strengths, in order to solve difficult problems that are not effectively solved by either of the two, several researchers have proposed models that integrate the two paradigms. The integration between MIP and CP can be achieved in two ways [26,48]:

1 By combining MIP and CP constraints into one hybrid model. In this case a hybrid algorithm that integrates constraint propagation with linear programming in a single search tree is also needed for the solution of the model (e. g. see [21,42]).
2 By decomposing the original problem into two subproblems: one MIP and one CP subproblem. Each model is solved separately and information obtained while solving one subproblem is used for the solution of the other subproblem [11,28].

Bockmayr and Kasper [10] have presented a unifying framework, called *Branch and Infer*, which can be used for the development of various integration schemes. Hooker et al. [25] have proposed a new modeling paradigm to perform efficient integration of MIP and CP techniques. In general, it is not clear whether an integration strategy performs better than a standalone MIP or CP approach, especially when the problem at hand is solved effectively by one of the two approaches. For some problems, however, the integration of the two approaches has led to significant computational improvements. Common integration schemes include the derivation of cuts for MIP formulations using CP techniques, the use of CP to accelerate column generation, and the use of CP local search to solve MIP scheduling problems. Integration schemes are described in [21,23,26,27,37], and [48].

MIP/CP Hybrid Schemes are particularly successful for scheduling problems that often arise in manufacturing, chemical and food industry, in transportation industries and in computing environments. To solve a scheduling problem one has to (i) allocate limited resources to tasks, and (ii) sequence the tasks allocated to a single resource. We will refer to the first set of decisions as the *assignment* problem, and the second set of decisions as the *sequencing* problem.

While heuristic methods are widely used, rigorous optimization methods have also been studied. To solve some hard scheduling problems to optimality, several authors have proposed MIP/CP hybrid schemes that exploit the complementary strengths of Mathematical and Constraint Programming. The main idea behind these approaches is to solve a relaxed MIP model to determine the allocation of machines to tasks, and use CP to check the feasibility of a given assignment and to generate cuts that are added in the relaxed MIP model. Thus, the complementary strengths of the two methods are combined: Mathematical Programming is used for optimization (i. e. identify potentially good assignments) and Constraint Programming to check feasibility.

Applications

A scheduling problem that has been widely studied using hybrid schemes is the Multi-Machine Assignment Scheduling Problem (MMASP) with Release and Due Times. In this problem a set I of N jobs have to be processed on a set J of M machines; the processing of job

$i \in I = \{1, \ldots N\}$ on any machine $j \in J = \{1, \ldots M\}$, must start after its release time r_i and must be completed before its due time d_i; the processing time and processing cost of job $i \in I$ on machine $j \in J$ are P_{ij}, and C_{ij} respectively. The objective is to minimize the total processing cost. The MMSAP was first studied by Hooker et al. [22] in a hybrid optimization framework.

A MIP model (M) for the MMASP consists of constraints (1)–(6):

$$\min Z = \sum_{i \in I} \sum_{j \in J} C_{ij} x_{ij} \tag{1}$$

$$\sum_{j \in J} x_{ij} = 1 \quad \forall i \in I \tag{2}$$

$$s_i \geq r_i \quad \forall i \in I \tag{3}$$

$$s_i + \sum_{j \in J} P_{ij} x_{ij} \leq d_i \quad \forall i \in I \tag{4}$$

$$y_{ii'} + y_{i'i} \geq x_{ij} + x_{i'j} - 1 \quad \forall j \in J, \forall i \in I, \forall i' \in I | i < i' \tag{5}$$

$$s_i + \sum_{j \in J} P_{ij} x_{ij} \leq s_{i'} + M(1 - y_{ii'}) \quad \forall i \in I, \forall i' \in I | i \neq i' \tag{6}$$

where binary x_{ij} is 1 if job i is assigned to machine j, binary $y_{ii'}$ is 1 is job i is scheduled before job i' in the same machine, and s_i is the start time of processing of job i.

Constraint (2) ensures that each job is processed on exactly one machine. Constraints (3) and (4) restrict each job to start after its release, and finish before its due time, respectively. Constraint (5) imposes the condition that if both jobs i and i' are assigned to the same machine j (i. e. $x_{ij} + x_{i'j} - 1 = 1$), then jobs i and i' must be sequenced (i. e. $y_{ii'} = 1$ or $y_{i'i} = 1$). Constraint (6) is a big-M sequencing constraint that is active when $y_{ii'}$ is 1.

Hooker et al. [22] and Jain and Grossmann [28] showed that model (M) is not efficient, due to the poor LP relaxation caused by the big-M sequencing constraint (6). Furthermore, they showed that standalone CP models are not efficient either, due to the large number of different assignments. To overcome this, the authors proposed a scheme where an IP master problem and a CP subproblem are solved iteratively. The IP master problem is a relaxation of model (M) and it is used to determine an assignment. The CP subproblem is used to check feasibility of the current assignment; if infeasible, integer cuts are added and the IP master problem is re-solved; if feasible, the subproblem gives a feasible sequence, and the algorithm terminates. The IP master problem consists of constraints (1)–(2), (7) and the integer cuts that are added at each iteration. Constraint (7) is used to eliminate infeasible assignments:

$$\sum_{i \in I} P_{ij} x_{ij} \leq \max_{i \in I} \{d_i\} - \min_{i \in I} \{r_i\} \quad \forall j \in J \tag{7}$$

The IP master problem does not include the sequencing binary variables $y_{ii'}$ and big-M constraint (6), it is solved fast, and at iteration k, yields a complete job-machine assignment $\boldsymbol{x}^k$. The CP subproblem is then used to check whether the current assignment $\boldsymbol{x}^k$ is feasible. At each iteration k, the set I_j^k of jobs assigned on machine $j \in J$, the processing time $\tilde{P}_i^k$ of each job, and the domain D_i^k for the start time of job i (i. e. $s_i \in D_i^k$) are given by (8)–(10), respectively:

$$I_j^k = \{i | x_{ij}^k = 1\} \quad \forall j \in J \tag{8}$$

$$\tilde{P}_i^k = \sum_{j \in J} P_{ij} x_{ij}^k \quad \forall i \in I \tag{9}$$

$$D_i^k = [r_i, d_i - \tilde{P}_i^k] \quad \forall i \in I \tag{10}$$

Thus, the CP subproblem reduces to $|J|$ one-machine independent problems, and for each one of these problems we try to find a sequence of jobs in I_j^k that satisfies constraint (10) and the non-overlapping of jobs assigned to machine k (see (5) and (6)). This problem can be solved using the global constraint *cumulative* [1], and various propagation techniques (time-table, disjunctive, edge-finding, etc.).

$$\text{cumulative}_{i \in O} \left((s_i, d_i, r_i), \ l, \ e \right) \tag{11}$$

The basic version of cumulative, (see detailed examples in [2]) takes 3 arguments, argument 1 is the set of operations O where each operation is characterized by three parameters, which can be either domain variables or values; the starting time s_i, the duration d_i, and the amount of some resource r_i used by the operation. The second argument l is the upper bound on the resource

consumption. The third argument e is the completion time. In this case, Eq. (11) can be written as follows:

$$\text{cumulative}_{i\in I_j}\left(\left(s_i, \tilde{P}_i^k, 1\right),\ 1,\ \max_{i\in I_j}\{d_i\}\right) \tag{12}$$

The global cumulative constraint is satisfied if the following conditions hold:

$$\sum_{i\in O: s_i\le t\le s_i+d_i, t\in 1..l} r_i \le l \quad AND \quad \max_{i\in O}(s_i+d_i)\le e \tag{13}$$

If there is no sequence for machine j that satisfies constraint (12), then the current assignment $\boldsymbol{x}^k$ is infeasible. For every infeasible one-machine problem we add the following integer cut in the cut-pool of the master problem:

$$\sum_{i\in I_j^k} x_{ij} \le |I_j^k| - 1 \tag{14}$$

If the IP master problem is solved to optimality, the lower bound provided by the optimal solution Z^k of the IP is non-decreasing, and the first feasible assignment is the assignment that yields the optimal solution with a minimum assignment cost. A schematic of the proposed algorithm is given in Fig. 1. The hybrid iterative approach was shown to be considerably faster than standalone MIP and CP models.

The above hybrid decomposition can also be implemented in a branch-and-cut framework (B&C), where the IP master problem is not solved to optimality before adding cuts. In the B&C framework, cuts are added either at a (possibly suboptimal) integer solution to the master problem or a *partially* feasible node, i. e. a node with integer assignments for a subset of machines.

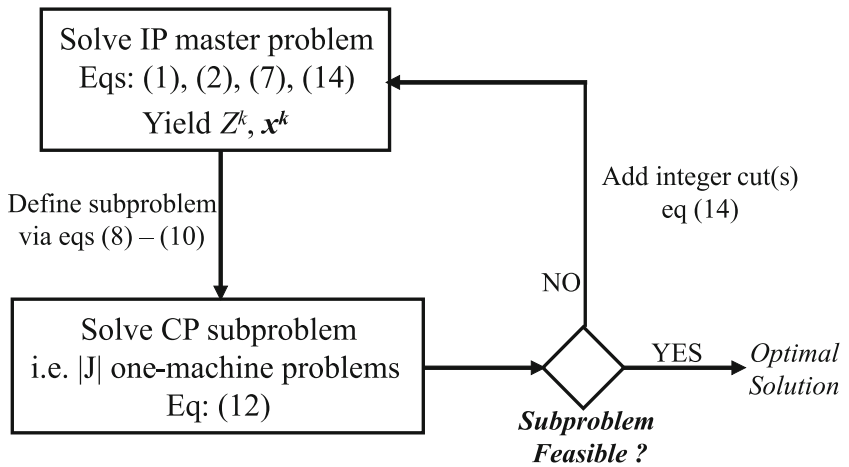
Bockmayr and Pisaruk [11] proposed a hybrid branch-and-cut scheme where the master problem is solved using an IP solver and the CP solver is called at a node of the tree, in order to generate integer cuts. The advantage of this method is that the IP model is not solved from scratch every time an integer solution (i. e. an assignment) is found. Furthermore, the authors were able to obtain cuts that are stronger than the ones proposed by Jain and Grossmann [28]. They were also able to generate cuts from fractional LP solutions of the IP model. The computational performance of the proposed hybrid branch-and-cut approach is better than the iterative IP/CP approach. Vazacopoulos and Verma [49] proposed certain Disjunctive and preemptive cuts to a priori forbid infeasible assignments and developed two hybrid MIP/CP algorithms for the MMASP. Sadykov and Wolsey [43] studied several hybrid approaches and developed two IP/CP hybrid schemes that appear to be better than those previously proposed. In the first, the authors were able to develop two classes of tightening inequalities, in the space of x_{ij} variables, which exclude many infeasible assignments and thus lead to smaller trees. The tightening inequalities are knapsack constraints, similar to constraint (7), but for subsets of set I. They also proposed a column generation algorithm using the tightening inequalities.

While the MMASP has been extensively studied due to its simple structure, hybrid schemes have also been developed for more complex scheduling problems. Harjunkoski and Grossmann [20], Timpe [46] and Constantino [14] presented hybrid schemes for complex chemical plants. Maravelias and Grossmann [33,34] proposed a general framework for integrating Mathematical and Constraint Programming methods for the solution of scheduling problems, while Maravelias [35] proposed the integration of MIP methods with heuristic algorithms. Hybrid methods that combine Mathematical and Constraint Programming have also been applied to transportation, inventory management and resource allocation problems.

Conclusions

While the computational efficiency of MIP/CP methods varies significantly, there is evidence that for some classes of problems they outperform existing methods. In general, if the structure of the problem at hand is exploited by efficient preprocessing and the generation of strong cuts, it is expected that hybrid schemes will be more effective because they combine the complementary strengths of two solution techniques.

The computational performance of hybrid methods relies on (i) the quality of the decomposition, (ii) the solution efficiency of the two subproblems, and (iii) the number of subproblems needed to be solved to prove optimality. Ideally, the original problem should be decomposed/reformulated into a tight MIP subproblem that is easily solved yielding potentially good feasible



Mixed Integer Programming/Constraint Programming Hybrid Methods, Figure 1
Iterative hybrid IP/CP scheme of Jain and Grossmann [28]

solutions, and a feasibility CP subproblem that is used to check feasibility and generate cuts.

In particular, MIP/CP methods have been shown to be very effective in tackling scheduling problems where both assignment and sequencing decisions have to be made. The key idea in these methods is the decomposition of the original problem into two sub problems; Mathematical Programming is used for the assignment of tasks to resources, while Constraint Programming is used for the sequencing of tasks on resources.

References

1. Aggoun A, Beldiceanu N (1993) Extending CHIP in order to solve complex scheduling problems. J Math Comput Model 17(7):57–73
2. Aggoun A, Vazacopoulos A (2004) Solving sports scheduling and timetabling problems with constraint programming. In: Butenko S, Gil-Lafuente J, Pardalos PM (eds) Economics, Management and Optimization in Sports. Springer, Dordrecht, pp 243–264
3. Balas E (1965) An Additive Algorithm for Solving Linear Programs with 0–1 Variables. Oper Res 13:517–546
4. Balas E (1975) Facets of the Knapsack Polytope. Math Program 8:146–164
5. Balas E (2001) Integer Programming. In: Floudas T, Pardalos P (eds) Encyclopedia of Optimization, vol 2. Kluwer, pp 492–499
6. Baptiste P, Le Pape C (1996) Disjunctive Constraints for Manufacturing Scheduling: Principles and Extensions. Int J Comput Integr Manuf 9(4):306–3410
7. Baptiste P, Le Pape C (1996) Edge-Finding Constraint Propagation Algorithms for Disjunctive and Cumulative Scheduling. In: Proc 15th Workshop of the UK Planning Special Interest Group
8. Baptiste P, Le Pape C, Nuijten W (2001) Constrained-Based Scheduling: Applying Constraint Programming to Scheduling Problems. Kluwer
9. Benders JF (1962) Partitioning Procedures for Solving Mixed Variables Programming Problems. Numer Math 4:238–252
10. Bockmayr A, Kasper T (1998) Branch and Infer: A Unifying Framework for Integer and Finite Domain Constraint Programming. INFORMS J Comput 10(3):287–300
11. Bockmayr A, Pisaruk N (2003) Detecting Infeasibility and Generating Cuts for MIP Using CP. In: Proc CP-AI-OR 2003. Montreal, pp 24–34
12. Caseau Y, Laburthe F (1994) Improved CLP Scheduling with Task Intervals. In: Proc 11th Int Conf Log Program
13. Chvatal V (1973) Edmonds Polytops and a Hierarchy of Combinatorial Problems. Discret Math 4:305–337
14. Constantino M (2003) Integrated Lot-sizing and Scheduling of Barbot's Paint Production Using Combined MIP/CP. LISCOS Project Summary Meeting
15. Dakin RJ (1965) A Tree Search Algorithm for Mixed Integer Programming Problems. Comput J 8:250–255
16. Fisher ML (1981) The Lagrangean Relaxation Method for Solving Integer Programming Problems. Manag Sci 27: 1–18
17. Gomory RE (1958) Outline of an Algorithm for Integer Solutions to Linear Programs. Bull Am Math Soc 64:275–278
18. Gomory RE (1963) An Algorithm for Integer Solutions to Linear Programs. In: Graves R, Wolfe P (eds) Recent Advances in Mathematical Programming. McGraw Hill, New York, pp 269–302
19. Guignard M, Kim S (1987) Lagrangean Decomposition: A Model Yielding Stronger Lagrangean Bounds. Math Program 39:215–228

20. Harjunkoski I, Grossmann IE (2002) Decomposition Techniques for Multistage Scheduling Problems Using Mixed-Integer and Constrained Programming Methods. Comp Chem Eng 26:1533–1552
21. Heipcke S (1999) An Example of Integrating Constraint Programming and Mathematical Programming. Electron Note Discret Math 1(1):84
22. Hooker JN, Ottosson G, Thorsteinsson ES, Kim HJ (1999) On integrating constraint propagation and linear programming for combinatorial optimization. In: Proc 16th National Conf Artif Intell (AAAI-99), AAAI, The AAAI Press/MIT Press, Cambridge, pp 136–141
23. Hooker JN, Osorio MA (1999) Mixed Logic / Linear Programming. Discret Appl Math 97:395–442
24. Hooker JN (2000) Logic Based Methods for Optimization: Combining Optimization and Constraint Satisfaction. Wiley, New York
25. Hooker JN, Ottosson G, Thornsteinsson ES, Kim HJ (2000) A Scheme for Unifying Optimization and Constraint Satisfaction Methods. Knowl Eng Rev 15:11–30
26. Hooker JN (2002) Logic, Optimization, and Constraint Programming. INFORMS J Comput 14(4):295–321
27. Hooker JN (2006) Integrated Methods for Optimization. Springer, Dordrecht
28. Jain V, Grossmann IE (2001) Algorithms for Hybrid MIP/CP Model for a Class of Optimization Problems. INFORMS J Comput 13:258–276
29. Jaffar J, Maher M (1994) Constraint Logic Programming. Surv J Log Program 19(20):503–581
30. Johnson EL, Nemhauser GL, Savelsbergh MWP (2000) Progress in Linear Programming Based Branch-and-Bound Algorithms: Exposition. INFORMS J Comput 12:2–23
31. Land AH, Doig AG (1960) An automatic Method for Solving Discrete Programming Models. Econom 28:83–97
32. Le Pape C (1998) Implementation of Resource constraints in ILOG SCHEDULE: A Library for the Development of Constrained-Based Scheduling Systems. Intell Syst Eng 3(2):55–66
33. Maravelias CT, Grossmann IE (2004) A Hybrid MIP/CP Decomposition Approach for the Short Term Scheduling of Multipurpose Plants. Comput Chem Eng 28:1921–1949
34. Maravelias CT, Grossmann IE (2004) Using MILP and CP for the Scheduling of Batch Chemical Processes. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Lect Note Comput Sci 3011:1–20
35. Maravelias CT (2006) A Decomposition Framework for the Scheduling of Single- and Multi-stage Processes. Comput Chem Eng 30(3):407–420
36. Marriott K, Stuckey PJ (1999) Introduction to Constraint Logic Programming. MIT Press, Cambridge, MA
37. Milano M (2003) Constraint and Integer Programming: Toward a Unified Methodology. Springer, Dordrecht
38. Nemhauser GL, Wolsey LA (1989) Integer and Combinatorial Optimization. Wiley, New York
39. Nuijten WPM (1994) Time and Resource Constrained Scheduling: A Constraint Satisfaction Approach. PhD Thesis, Eindhoven University of Technology
40. Padberg MW, van Roy TJ, Wolsey LA (1985) Valid Linear Inequalities for Fixed Charged Problems. Oper Res 33:842–861
41. Pinedo M (2001) Scheduling: Theory, Algorithms, and Systems. Prentice Hall, Englewood Cliffs
42. Rodosek R, Wallace MG, Hajian MT (1999) A New Approach to Integrating Mixed Integer Programming and Constraint Logic Programming. Ann Oper Res 86:63–87
43. Sadykov R, Wolsey L (2006) Integer and Constraint Programming in Solving a Multi-Machine Assignment Scheduling Problem with Deadlines and Release Dates. INFORMS J Comput 18(2):209–217
44. Simonis H (1995) Application Development with the CHIP System. In: Proc ESPRIT WG CONTESSA Workshop on Constraint Databases and Applications. Lect Note Comput Sci 1034:1–21
45. Smith SF, Cheng C-C (1993) Slack-based Heuristics for Constrained Satisfaction Scheduling. In: Proc 11th National Conf Artif Intell
46. Timpe C (2003) Solving BASF's Plastics Production and Lot-sizing Problem Using Combined CP/MIP. LISCOS Project Summary Meeting
47. van Hentenryck P (1989) Constraint Satisfaction in Logic Programming. MIT Press, Cambridge, MA
48. van Hentenryck P (2002) Constraint and Integer Programming in OPL. INFORMS J Comput 14(4):345–372
49. Vazacopoulos A, Verma N (2005) Hybrid MIP-CP techniques to solve the Multi-Machine Assignment and Scheduling Problem in Xpress-CP. In: Geunes J, Pardalos PM (eds) Supply Chain Optimization I. Springer, Dordrecht, pp 391–413
50. Wolsey LA (1975) Faces for a Linear Inequality in 0–1 Variables. Math Program 8:165–178
51. Wolsey LA (1990) Valid Inequalities for Mixed-Integer Programs with Generalized and Variable Upper Bound Constraints. Discret Appl Math 25:251–261
52. Wolsey L (1998) Integer Programming. Wiley, New York

Model Based Control for Drug Delivery Systems

PINKY DUA[1,2], VIVEK DUA[3], EFSTRATIOS N. PISTIKOPOULOS[4]

[1] Centre for Process Systems Engineering, Department of Chemical Engineering, Imperial College London, London, UK

[2] GlaxoSmithKline Research & Development Limited, Harlow, UK

[3] Centre for Process Systems Engineering, Department of Chemical Engineering, University College London, London, UK
[4] Centre for Process Systems Engineering, Department of Chemical Engineering, Imperial College London, London, UK

Article Outline

Abstract
Introduction
Surgery Under Anesthesia
Modeling Anesthesia
Control of Anesthesia
Results
Blood Glucose Control for Type 1 Diabetes
Model for Type 1 Diabetes
Parametric Controller
Results
Concluding Remarks
See also
References

Abstract

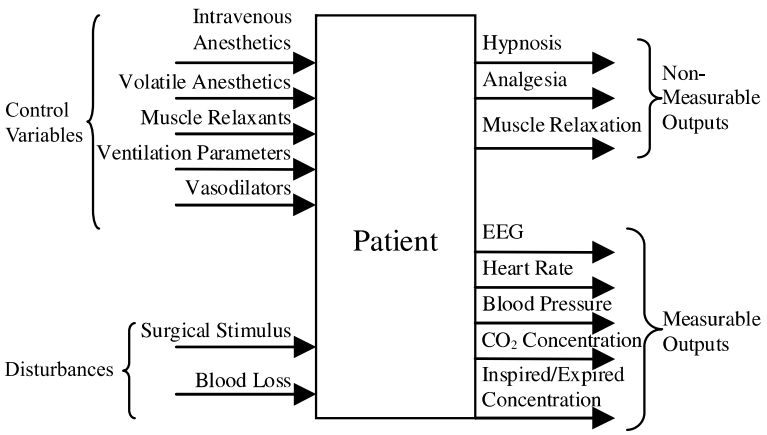
This chapter presents model based controllers for two drug delivery systems: (i) surgery under anesthesia and (ii) insulin delivery for type 1 diabetes. For anesthesia, a compartmental model is presented and then used for deriving model predictive controller for simultaneous control of mean arterial pressure (MAP), cardiac output (CO) and hypnosis. For type 1 diabetes, parametric control techniques are used for obtaining insulin delivery rate as an explicit function of the state of the patient. This reduces the implementation of the model based controller to function evaluations that can be carried out on a portable computational hardware.

Introduction

Drug delivery systems aim to provide effective therapy by minimizing side effects, reducing deviations from the desired state of the patient and increasing patient compliance and safety. Automation of a drug delivery system relies on the mathematical model of the patient that can take into account the pharmacokinetic and pharmacodynamic effects of the drugs on various organs of the body. To reduce the complexity of the mathematical model, some of the organs are lumped and then represented as interconnected compartments. This reduction in complexity is quite important especially for models that are used for controlling the amount of drugs to be infused. In this chapter, models and advanced model based controllers for two drug delivery systems are presented. In Sect. "Surgery Under Anesthesia", the first system which is concerned with the delivery of anesthetics for patients undergoing surgery is discussed. A compartmental model is presented that considers a choice of three drugs, isoflurane, dopamine and sodium nitroprusside, and therefore allows simultaneous control of mean arterial pressure, cardiac output and hypnosis. This model is then used for designing model predictive controller and the performance of the controller is tested for its set-point tracking capabilities. In Sect. "Blood Glucose Control for Type 1 Diabetes" model based parametric controller for the regulation of the blood glucose concentration for people with type 1 diabetes is derived. The key advantage of this controller is that the optimal drug infusion rate is obtained as an explicit function of the state of the patient and therefore requires simple function evaluations for its implementation. Concluding remarks are presented in Sect. "Concluding Remarks".

Surgery Under Anesthesia

Anesthesia is defined as the absence or loss of sensation. In order to provide safe and adequate anesthesia, the anesthesiologist must guarantee analgesia, provide hypnosis, muscle relaxation and maintain vital functions of the patient. Anesthesiologists administer anesthetics and monitor a wide range of vital functions, such as mean arterial pressure (MAP), heart rate, cardiac output (CO). These vital functions need to be monitored and maintained within tolerable operating ranges by infusing various drugs and/or intravenous fluids as shown in Fig. 1. Automation of anesthesia is desirable as it will provide more time and flexibility to the anesthesiologist to focus on critical issues, monitor the conditions that cannot be easily measured and overall improve patient's safety. Also, the cost of the drugs will be reduced and shorter time will be spent in the post-operative care unit. There is a significant amount of research in the area of developing models and control strategies for anesthesia [10,14,15,17]. Gentilini et



Model Based Control for Drug Delivery Systems, Figure 1
Anesthesia control system (adapted from [6])

al. [6] proposed a model for the regulation of MAP and hypnosis with isoflurane. It was observed that controlling both MAP and hypnosis simultaneously with isoflurane was difficult. Yu et al. [16] proposed a model for regulating MAP and CO using dopamine (DP) and sodium nitroprusside (SNP), but the control of hypnosis was not considered.

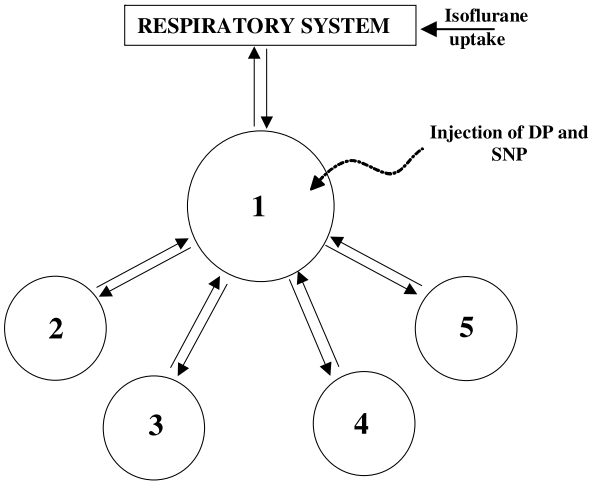
In the next section, a compartmental model is presented, which allows the simultaneous regulation of the MAP and the unconsciousness of the patients. The model is characterized by: (i) pharmacokinetics for the uptake and distribution of the drugs, (ii) pharmacodynamics which describes the effect of the drugs on the vital functions and (iii) baroreflex for the reaction of the central nervous system to changes in the blood pressure. The model involves choice of three drugs, isoflurane, DP and SNP. This combination of drugs allows simultaneous regulation of MAP and hypnosis.

Modeling Anesthesia

The model is based on the distribution of isoflurane in the human body [15]. It consists of five compartments organized as shown in Fig. 2.

The compartments 1–5 represent lungs, vessel rich organs (e. g. liver), muscles, other organs and tissues and fat tissues respectively.

The distribution of the drugs occurs from the central compartment to the peripheral compartments by the arteries and from the peripheral to the central by the veins. The first compartment in Fig. 2 is the central



Model Based Control for Drug Delivery Systems, Figure 2
Compartmental model

compartment and heart can be considered to be belonging to the central compartment, whereas compartments 2–5 are the peripheral compartments.

Pharmacokinetic Modeling The uptake of isoflurane in central compartment via the respiratory system is modeled as:

$$V\frac{dC_{insp}}{dT} = Q_{in}C_{in} - (Q_{in} - \Delta Q)C_{insp} - f_R(V_T - \Delta)(C_{insp} - C_{out}),$$

where C_{insp} is the concentration of isoflurane inspired by the patient (g/ml), C_{in} is the concentration of isoflu-

rane in the inlet stream (g/ml), C_{out} is the concentration of isoflurane in the outlet stream (g/ml), Q_{in} is the inlet flow rate (ml/min), ΔQ is the losses (ml/min), V is the volume of the respiratory system (l), f_R is the respiratory frequency (l/min), V_T is the tidal volume (l) and Δ is the physiological dead space (ml). For the central compartment, the concentration of isoflurane is given by:

$$V_1 \frac{dC_1}{dt} = \sum_{i=2}^{5} \left(Q_i \left(\frac{C_i}{R_i} - C_1 \right) \right) + f_R (V_T \Delta)(C_{insp} C_1) ,$$

where C_i is the concentration of the drug in compartment i (g/ml), R_i is the partition coefficient between blood and tissues in compartment i, Q_i is the blood flow in compartment i (ml/min). The concentration of DP and SNP in the central compartment is modeled as follows:

$$V_1 \frac{dC_1}{dt} = \sum_{i=2}^{5} \left(Q_i \left(\frac{C_i}{R_i} - C_1 \right) \right) + C_{inf} - \frac{1}{\tau_{\frac{1}{2}}} C_1 V_1 ,$$

where C_{inf} is the concentration of the drug infused (g/min), V_i is the volume of compartment i (ml) and $\tau_{1/2}$ is the half-life of the drug (min). Isoflurane is eliminated by exhalation and metabolism in liver, the 2nd compartment, as follows:

$$V_2 \frac{dC_2}{dt} = Q_2 \left(C_1 - \frac{C_2}{R_2} \right) - k_{20} C_2 V_2 ,$$

where k_{20} is the rate of elimination of isoflurane in the 2nd compartment (min^{-1}). The distribution of isoflurane in compartments 3 to 5 is given by:

$$V_i \frac{dC_i}{dt} = Q_i \left(C_1 - \frac{C_i}{R_i} \right) , \quad i = 3, \ldots, 5 .$$

The natural decay of DP and SNP in the body, for compartment 2 to 5, is given by:

$$V_i \frac{dC_i}{dt} = Q_i \left(C_1 - \frac{C_i}{R_i} \right) - \frac{1}{\tau_{\frac{1}{2}}} C_i V_i , \quad i = 2, \ldots, 5 .$$

Pharmacodynamic Modeling The effect of DP and SNP on two of the heart's characteristic parameters: maximum elastance (E_{max}) and systemic resistance (R_{sys}) is given by:

$$\frac{dEff}{dt} = k_1 C_1^N (Eff_{max} - Eff) - k_2 Eff$$
$$E_{max} = E_{max,0} \left(1 + Eff_{DP-E_{max}} \right)$$
$$R_{sys} = R_{sys,0} \left(1 - Eff_{DP-R_{sys}} - Eff_{SNP-R_{sys}} \right) ,$$

where Eff is the measure of the effect of drug on the parameters of interest, R_{sys} is the systemic resistance (mmHg/(ml/min)), E_{max} is the maximum elastance (mmHg/ml), $E_{max,0}$ is nominal maximum elastance, $R_{sys,0}$ is nominal systemic resistance, $Eff_{DP-Emax}$ is effect of DP on E_{max}, $Eff_{DP-Rsys}$ is effect of DP on R_{sys}, $Eff_{SNP-Rsys}$ is the effect of SNP on R_{sys}, k_1, k_2 are the rate constants and N is the non-linearity constant. MAP can then be expressed as a function of E_{max} and R_{sys} as:

$$MAP^2 \frac{1}{R_{sys}^2} + 2K^2 MAP - 2K^2 V_{LV} E_{max} = 0$$
$$K = \frac{A_{aorta} A_{LV}}{\sqrt{\rho} \sqrt{A_{LV}^2 - A_{aorta}^2}} ,$$

where MAP is the mean arterial pressure (mmHg), A_{aorta} is the cross sectional area of the aorta (cm^2), A_{LV} is the cross sectional area of the left ventricle (cm^2), V_{LV} is the mean volume of the left ventricle (ml) and ρ is the blood density (g/ml). Isoflurane affects MAP as follows:

$$MAP = \frac{Q_1}{\sum_{i=2}^{5} \left(g_{i,0} \left(1 + b_i C_i \right) \right)} ,$$

where, $g_{i,0}$ is the baseline conductivities (ml/(min.mmHg)) and b_i is the variation coefficient of conductivity (ml/g). There is experimental evidence that a transportation delay exists between the lungs and the site of effect of isoflurane on the unconsciousness of the patient. In order to model this, an effect compartment is linked to the central compartment. The concentration of isoflurane within this compartment is related to the central compartment, which is given by:

$$\frac{dC_e}{dt} = k_{e0} (C_1 - C_e) ,$$

where C_e is the concentration of isoflurane in the effect compartment (g/ml), and k_{e0} is the kinetics in the effect compartment (min^{-1}). The action of isoflurane can be

then expressed as follows:

$$\Delta\text{BIS} = \Delta\text{BIS}_{\text{MAX}} \frac{C_e^{\gamma}}{C_e^{\gamma} + \text{EC}_{50}^{\gamma}}$$
$$\Delta\text{BIS} = \text{BIS} - \text{BIS}_0$$
$$\Delta\text{BIS}_{\text{MAX}} = \text{BIS}_{\text{MAX}} - \text{BIS}_0 ,$$

where BIS_0 is the baseline value of BIS (assumed to be 100), BIS_{MAX} is the maximum value of BIS (assumed to be 0), EC_{50} is the patient's sensitivity to the drug and γ is the measure of the degree of non-linearity.

Baroreflex Baroreflex is obtained from a set of transfer functions relating the mean arterial pressure to the maximum elastance and the systemic resistance and is given by:

$$\text{bfc} = \frac{e^{c(\text{MAP}-\text{MAP0})}}{1 + e^{c(\text{MAP}-\text{MAP0})}} ,$$

where c is the empirical constant (mmHg).

Control of Anesthesia

The model presented in the previous section was validated by carrying out a number of dynamic simulations for different amounts of drug dosages and disturbances using gPROMS [7]. For designing controllers, this model was linearized at the nominal values of inputs: 0.6% vol. of isoflurane, 2 µg/kg/min of DP and 4 µg/kg/min of SNP and outputs: 57.38 mmHg of MAP, 61.1 BIS and 1.21 l/min of CO, to obtain a state-space model of the following form:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t + Du_t , \end{aligned} \tag{1}$$

subject to the following constraints:

$$\begin{aligned} x_{\min} \le x_t \le x_{\max} \\ y_{\min} \le y_t \le y_{\max} \\ u_{\min} \le u_t \le u_{\max} , \end{aligned} \tag{2}$$

where $x_t \in R^n$, $y_t \in R^l$, $u_t \in R^m$, are the state, output and input vectors respectively and the subscripts min and max denote lower and upper bounds respectively.

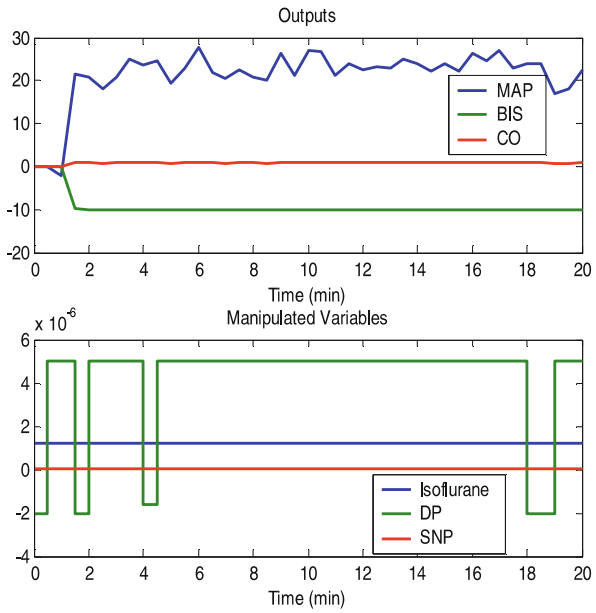
Model predictive control (MPC) [5] problem can then be posed as the following optimization problem:

$$\begin{aligned} \min_U \; & J(U, x(t)) = x_{t+N_y|t}^T P x_{t+N_y|t} \\ & + \sum_{k=0}^{N_y-1} \left[x_{t+k|t}^T Q x_{t+k|t} + u_{t+k}^T R u_{t+k} \right] \\ s.t. \; & x_{\min} \le x_{t+k|t} \le x_{\max} , \quad k = 1, \ldots, N_c \\ & y_{\min} \le y_{t+k|t} \le y_{\max} , \quad k = 1, \ldots, N_c \\ & u_{\min} \le u_{t+k} \le u_{\max} , \quad k = 1, \ldots, N_c \\ & x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k} , \quad k \ge 0 \\ & y_{t+k|t} = Cx_{t+k} + Du_{t+k} , \quad k \ge 0 \\ & u_{t+k} = Kx_{t+k|t} , \quad N_u \le k \le N_y , \end{aligned} \tag{3}$$

where $U = [u_t^T, \ldots, u_{t+N_u-1}^T]^T$, Q and R are constant, symmetric and positive definite matrices, P is given by the solution of the Riccati or Lyapunov equation, N_y, N_u and N_c are the prediction, control and constraint horizons respectively and the superscript T denotes transpose of the vector. Problem (3) is solved at each time t for the current state x_t and the vector of predicted state variables, $x_{t+1|t}, \ldots, x_{t+Ny|t}$ at time $t+1, \ldots, t+k$ respectively and corresponding control actions $u_t, \ldots, u_{t+k}$ are obtained.

Results

The model for anesthesia consists of 23 states, 3 outputs and 3 inputs. This state-space form of the model is then adapted for designing model predictive controller by using the MATLAB Model Predictive Control Toolbox™ [11]. For designing the MPC controller, the following input: $0 \le \text{DP} \le 7$ µg/kg.min, $0 \le \text{SNP} \le 10$ µg/kg.min, $0 \le \text{Isoflurane} \le 5\%$vol., and output constraints: $40 \le \text{MAP} \le 150$mmHg, $40 \le \text{BIS} \le 65$, $1 \le \text{CO} \le 6.5$ l/min are used. A prediction horizon of 5, control horizon of 3 and sampling time of 0.5 minutes are considered. A set point of [20–10 1]′ deviation from the nominal point of the output variables is given and the performance of the controller is shown in Fig. 3. It is observed that the MPC tracks the set point quite well. The performance of the MPC was also tested by reducing the model to 15 states and was observed to be very good. From the above results it can be inferred that the model based control technology provides a promising platform for the automation of anesthesia.

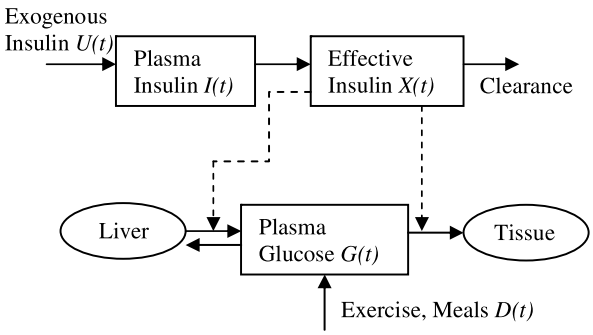


Model Based Control for Drug Delivery Systems, Figure 3
MPC performance for anesthesia

Note that MPC solves a quadratic program at regular time intervals. In the next section a parametric programming approach for control of blood glucose for type 1 diabetes is presented that does not require repetitively solving quadratic programs.

Blood Glucose Control for Type 1 Diabetes

Diabetes is a disease that affects the body's ability to regulate glucose. In Type 1 diabetes, the pancreas produces insufficient insulin, and exogenous insulin is required to be infused at an appropriate rate to maintain blood sugar levels within the range of 60–120 mg/dl [2]. If insulin is supplied in excess, the blood glucose level can go well below normal (< 60 mg/dl), a condition known as *hypoglycemia*. On the other hand, if insulin is not supplied sufficiently, the blood glucose level is elevated above normal (> 120 mg/dl), a condition known as *hyperglycemia*. Both hypo- and hyperglycemia can be harmful to an individual's health. Hence, it is very important to control the level of blood glucose in the body to within a reasonable range [9,12]. In the following sections, advanced model based controllers for regulating the blood glucose concentration for type 1 diabetes are presented.



Model Based Control for Drug Delivery Systems, Figure 4
Schematic representation of the Bergman model

Model for Type 1 Diabetes

The Bergman model [1] is used in this study, which presents a 'minimal' model comprising 3 equations to describe the dynamics of the system. The schematic representation of the model is shown in Fig. 4. The modeling equations are:

$$\frac{dG}{dt} = -P_1 G - X(G + G_b) + D(t) \tag{4}$$

$$\frac{dI}{dt} = -n(I + I_b) + U(t)/V_1 \tag{5}$$

$$\frac{dX}{dt} = -P_2 X + P_3 I\,. \tag{6}$$

The states in this model are: G, plasma glucose concentration (mg/dl) relative to basal value, I, plasma insulin concentration (mU/l) relative to basal value, and X, proportional to I in remote compartment (min^{-1}). The inputs are: $D(t)$, meal glucose disturbance (mg/dl/min), $U(t)$, manipulated insulin infusion rate (mU/min) and G_b, I_b, nominal values of glucose and insulin concentration (81 mg/dl; 15 mU/l). The parameter values for a Type 1 diabetes are: $P_1 = 0\,\text{min}^{-1}$, $P_2 = 0.025\,\text{min}^{-1}$, $P_3 = 0.000013\,\text{l/mUmin}^2$, $V_1 = 12\,\text{l}$ and $n = 5/54\,\text{min}$ [4].

The model, (4)–(6) is linearized about the steady-state values of $G_b = 81$ mg/dl, $I_b = 15$ mU/l, $X_b = 0$ and $U_b = 16.66667$ mU/min to obtain the state space model of the form: $x_{t+1} = Ax_t + Bu_t + B_d d_t$ where the term d_t represents the input disturbance glucose meal. The sampling time considered is 5 minutes, which is reasonable for the current glucose sensor technology. The discrete state-space matrices A, B, C and B_d

are as follows:

$$A = \begin{bmatrix} 1 & -0.000604 & -21.1506 \\ 0 & 0.6294 & 0 \\ 0 & 0.00004875 & 0.8825 \end{bmatrix}$$
$$B = \begin{bmatrix} -0.000088 \\ 0.3335 \\ 0.0000112 \end{bmatrix},$$
$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \quad B_d = \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix}$$

The constraints imposed are $60 \leq G + G_b \leq 180$ and $0 \leq U + U_b \leq 100$.

Parametric Controller

Parametric programming can be used in the MPC framework to obtain U as a function of x_t by treating U as optimization variables and x_t as parameters as described next [3,13]. For simplicity in presentation assume that $N_y = N_u = N_c$, the theory presented is however valid for the case when N_y, N_u and N_c are not equal. The equalities in formulation (3) are eliminated by making the following substitution:

$$x_{t+k|t} = A^k x_t + \sum_{j=0}^{k-1} A^j B u_{t+k-1-j} \tag{7}$$

to obtain the following Quadratic Program (QP):

$$\begin{aligned} &\min_U \frac{1}{2} U^T H U + x_t^T F U + \frac{1}{2} x_t^T Y x_t \\ &s.t. GU \leq W + E x_t \,, \end{aligned} \tag{8}$$

where, $U = [u_t^T, \ldots, u_{t+N_u-1}^T]^T \in R^s$, is the vector of optimization variables, $s = mN_u$, H is a constant, symmetric and positive definite matrix and H, F, Y, G, W, E are obtained from Q, R and (1) and (2).

The QP problem in (8) can now be reformulated as a multi-parametric quadratic program (mp-QP):

$$\begin{aligned} &V_z(x) = \min_z \frac{1}{2} z^T H z \\ &s.t. Gz \leq W + S x_t \,, \end{aligned} \tag{9}$$

where, $z = U + H^{-1} F^T x_t, z \in R^s$, and $S = E + GH^{-1}F^T$.

This mp-QP is solved by treating z as the vector of optimization variables and x_t as the vector of parameters to obtain z as an explicit function of x_t. U is then obtained as an explicit function of x_t by using $U = z - H^{-1}F^T x_t$.

Results

A prediction horizon $N_y = 5$ and Q/R ratio of 1000 is considered for deriving the control law – this results in partitioning of the state-space into 31 polyhedral regions. These regions are known as Critical Regions (CR). Associated with each CR is a control law that is an affine function of the state of the patient. For example, one of the CRs is given by the following state inequalities:

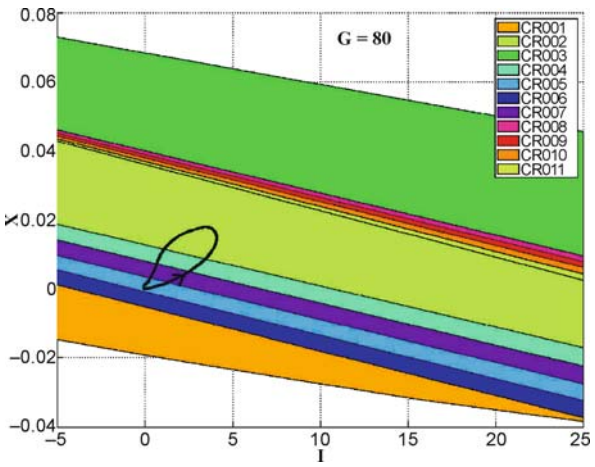
$$\begin{aligned} &-5 \leq I \leq 25 \\ &0.0478972G - 0.0002712I - X \leq 0.104055 \\ &0.0261386G - 0.0004641I - X \leq 0.0576751 \\ &-0.00808846G + 0.00119685I + X \leq 0 \\ &-0.00660123G + 0.00130239I + X \leq 0 \\ &0.00609435G - 0.00134362I - X \leq 0 \end{aligned} \tag{10}$$

where the insulin infusion rate as a function of the state variables for the next five time intervals is given as follows:

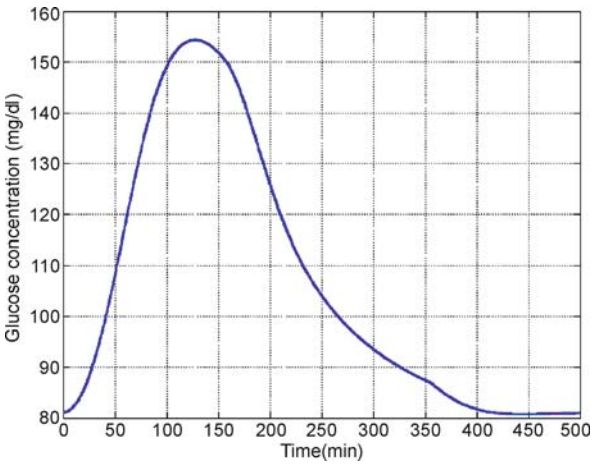
$$\begin{aligned} U(1) &= 30.139G - 0.44597I - 3726.2X \\ U(2) &= 24.874G - 0.40326I - 3280.4X \\ U(3) &= 20.16G - 0.35946I - 2842.8X \\ U(4) &= 16.002G - 0.31571I - 2424.1X \\ U(5) &= 0 \end{aligned} \tag{11}$$

The complete partitioning of the state-space for G = 80 mg/dl into CRs is shown in Fig. 5. The performance of the parametric controller for a 50 mg meal disturbance [8] is as shown in Figs. 6 and 7. The corresponding trajectory of the state variables is also shown in Fig. 5.

The model based parametric controller of the form given in (10) and (11) can be stored and implemented on a simple computational hardware and therefore can provide effective therapy at low on-line computational costs.



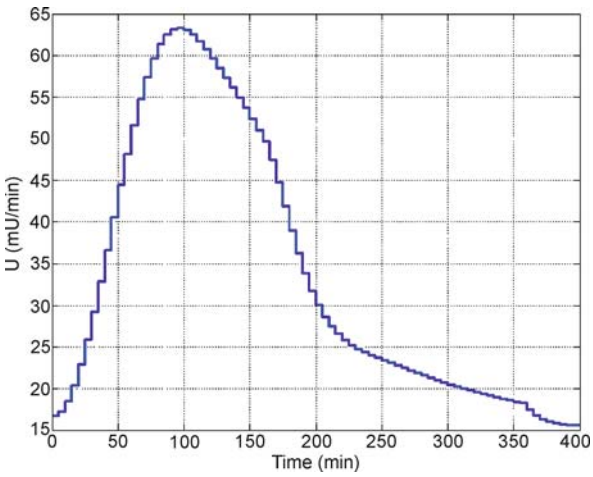
Model Based Control for Drug Delivery Systems, Figure 5
Critical regions for type 1 diabetes



Model Based Control for Drug Delivery Systems, Figure 6
Glucose concentration vs. time

Concluding Remarks

Automation of drug delivery systems aims at reducing patient inconvenience by providing better and personalized healthcare. The automation can be achieved by developing detailed models and by deriving advanced controllers that can take into account the model as well as the constraints on state and control variables. In this chapter, a compartmental model incorporating pharmacokinetic and pharmacodynamic aspects for delivery of anesthetic agents has been presented. This model was then used for the derivation of model predictive controller. For type 1 diabetes, implementation of advanced model based controllers through a simple computational hardware was demonstrated by deriving insulin delivery rate as an explicit function of the state of patient. The developments presented in this chapter highlight the importance of modeling and control techniques for biomedical systems.



Model Based Control for Drug Delivery Systems, Figure 7
Insulin infusion vs. time

See also

► Nondifferentiable Optimization: Parametric Programming

References

1. Bergman RN, Phillips LS, Cobelli C (1981) Physiologic evaluation of factors controlling glucose tolerance in man. J Clin Invest 68:1456–1467
2. DCCT-The Diabetes Control and Complications Trial Research Group (1993) The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus. New Engl J Med 329:977–986
3. Dua P, Doyle FJ III, Pistikopoulos EN (2006) Model based glucose control for type 1 diabetes via parametric programming. IEEE Trans Biomed Eng 53:1478–1491
4. Fisher ME (1991) A semiclosed loop algorithm for the control of blood glucose levels in diabetics. IEEE Trans Biomed Eng 38:57–61
5. Garcia CE, Prett DM, Morari M (1989) Model predictive control: theory and practice – a survey. Automatica 25: 335–348
6. Gentilini A, Frei CW, Glattfelder AH, Morari M, Sieber TJ, Wymann R, Schnider TW, Zbinden AM (2001) Multi-tasked closed-loop control in anesthesia. IEEE Eng Med Biol 20:39–53

7. gPROMS (2003) Introductory user guide, Release 2.2. Process Systems Enterprise Ltd, London
8. Lehmann ED, Deutsch T (1992) A physiological model of glucose-insulin interaction in type 1 diabetes mellitus. J Biomed Eng 14:235–242
9. Lynch SM, Bequette BW (2002) Model predictive control of blood glucose in type I diabetics using subcutaneous glucose measurements. In: Proc. American Control Conf., Anchorage, AK, pp 4039–4043
10. Mahfouf M, Asbury AJ, Linkens DA (2003) Unconstrained and constrained generalized predictive control of depth of anaesthesia during surgery. Control Eng Pract 11: 1501–1515
11. MATLAB (1998) MPC Toolbox manual. The MathWorks, Natick
12. Parker RS, Doyle FJ III, Peppas NA (2001) The intravenous route to blood glucose control. IEEE Eng Med Biol 20: 65–73
13. Pistikopoulos EN, Dua V, Bozinis NA, Bemporad A, Morari M (2002) On line optimization via off-line parametric optimization tools. Comput Chem Eng 26:175–185
14. Rao RR, Palerm CC, Aufderhiede B, Bequette BW (2001) Automated regulation of hemodynamic variables. IEEE Eng Med Biol 20:24–38
15. Yasuda N, Lockhart SH, Eger EI, Weiskopf RB, Laster M, Taheri S, Peterson NA (1991) Comparison of kinetics of sevoflurane and isoflurane in humans. Anesthesia Analg 72:316–324
16. Yu C, Roy RJ, Kaufman H (1990) A circulatory model for combined nitroprusside-dopamine therapy in acute heart failure. Med Prog Technol 16:77–88
17. Zwart AN, Smith NT, Beneken JEW (1972) Multiple model approach to uptake and distribution of halothane: the use of an analog computer. Comput Biomed Res 5:228–238

Modeling Difficult Optimization Problems

JOSEF KALLRATH[1,2]
[1] GVC/S (Scientific Computing) - B009, BASF Aktiengesellschaft, Ludwigshafen, Germany
[2] Astronomy Department, University of Florida, Gainesville, USA

MSC2000: 90C06, 90C10, 90C11, 90C30, 90C57, 90C90

Article Outline

Introduction
Models and the Art of Modeling
Tricks of the Trade for Monolithic Models
Decomposition Techniques
Column Generation
Column Enumeration
Branch-and-Price
Rolling Time Decomposition
An Exhaustion Method
Indices and Sets
Variables
The Idea of the Exhaustion Method
Computing Lower Bounds
Primal Feasible Solutions and Hybrid Methods
Summary
References

Introduction

We define difficult optimization problems as problems that cannot be solved to optimality or to any guaranteed bound by any standard solver within a reasonable time limit. The problem class we have in mind are mixed-integer programming (MIP) problems. Optimization, and especially MIP, is often appropriate and frequently used to model real-world optimization problems. While it started in the 1950s, models have become larger and more complicated.

A reasonable general framework is mixed-integer nonlinear programming (MINLP) problems. They are specified by the augmented vector $\mathbf{x}_{\oplus}^{\mathrm{T}} = \mathbf{x}^{\mathrm{T}} \oplus \mathbf{y}^{\mathrm{T}}$ established by the vectors $\mathbf{x}^{\mathrm{T}} = (x_1, \ldots, x_{n_c})$ and $\mathbf{y}^{\mathrm{T}} = (y_1, \ldots, y_{n_d})$ of n_c continuous and n_d discrete variables, an objective function $f(\mathbf{x}, \mathbf{y})$, n_e equality constraints $\mathbf{h}(\mathbf{x}, \mathbf{y})$, and n_i inequality constraints $\mathbf{g}(\mathbf{x}, \mathbf{y})$. The problem

$$\min \left\{ f(\mathbf{x}, \mathbf{y}) \;\middle|\; \begin{array}{l} \mathbf{h}(\mathbf{x}, \mathbf{y}) = 0,\ \mathbf{h} : X \times U \to \mathbb{R}^{n_e}, \\ \mathbf{x} \in X \subseteq \mathbb{R}^{n_c} \\ \mathbf{g}(\mathbf{x}, \mathbf{y}) \geq 0,\ \mathbf{g} : X \times U \to \mathbb{R}^{n_i}, \\ \mathbf{y} \in U \subseteq \mathbb{Z}^{n_d} \end{array} \right\} \tag{1}$$

is called a *mixed-integer nonlinear programming* (MINLP) problem if at least one of the functions $f(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$, or $\mathbf{h}(\mathbf{x}, \mathbf{y})$ is nonlinear. The vector inequality, $\mathbf{g}(\mathbf{x}, \mathbf{y}) \geq 0$, is to be read componentwise. Any vector $\mathbf{x}_{\oplus}^{\mathrm{T}}$ satisfying the constraints of (1) is called a *feasible point* of (1). Any feasible point whose objective function value is less than or equal to that of all other feasible points is called an *optimal solution*. From this

definition it follows that the problem might not have a unique optimal solution.

Depending on the functions $f(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$, and $\mathbf{h}(\mathbf{x}, \mathbf{y})$ in (1) we get the following structured problems known as

Acronym	*Type of optimization*	$f(\mathbf{x}, \mathbf{y})$	$\mathbf{h}(\mathbf{x}, \mathbf{y})$	$\mathbf{g}(\mathbf{x}, \mathbf{y})$	n_d
LP	Linear programming	$\mathbf{c}^T\mathbf{x}$	$\mathsf{A}\mathbf{x} - \mathbf{b}$	$\mathbf{x}$	0
QP	Quadratic programming	$\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}$	$\mathsf{A}\mathbf{x} - \mathbf{b}$	$\mathbf{x}$	0
NLP	Nonlinear programming				0
MILP	Mixed-integer LP	$\mathbf{c}^T\mathbf{x}_\oplus$	$\mathsf{A}\mathbf{x}_\oplus - \mathbf{b}$	$\mathbf{x}_\oplus$	≥ 1
MIQP	Mixed-integer QP	$\mathbf{x}_\oplus^T Q\mathbf{x}_\oplus + \mathbf{c}^T\mathbf{x}_\oplus$	$\mathsf{A}\mathbf{x}_\oplus - \mathbf{b}$	$\mathbf{x}_\oplus$	≥ 1
MINLP	Mixed-integer NLP				≥ 1

with a matrix A of m rows and n columns, i. e., $\mathsf{A} \in \mathcal{M}(m \times n, \mathbb{R})$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$, and $n = n_c + n_d$. Real-world problems lead much more frequently to LP and MILP than to NLP or MINLP problems. QP refers to quadratic programming problems. They have a quadratic objective function but only linear constraints. QP and MIQP problems often occur in applications of the financial services industry.

While LP problems as described in [31] or [1] can be solved relatively easily (the number of iterations, and thus the effort to solve LP problems with m constraints, grows approximately linearly in m), the computational complexity of MILP and MINLP grows exponentially with n_d but depends strongly on the structure of the problem. Numerical methods to solve NLP problems work iteratively, and the computational problems are related to questions of convergence, getting stuck in bad local optima and availability of good initial solutions. Global optimization techniques can be applied to both NLP and MINLP problems, and its complexity increases exponentially in the number of all variables entering nonlinearly into the model.

While the word *optimization*, in nontechnical or colloquial language, is often used in the sense of *improving*, the mathematical optimization community sticks to the original meaning of the word related to finding the *best value* either globally or at least in a local neighborhood. For an algorithm being considered as a (mathematical, strict, or exact) optimization algorithm in the mathematical optimization community there is consensus that such an algorithm computes feasible points proven globally (or locally) optimal for linear (nonlinear) optimization problems. Note that this is a definition of a mathematical optimization algorithm and *not* a statement saying that computing a local optimum is sufficient for nonlinear optimization problems. In the context of mixed-integer linear problems an optimization algorithm [12] and [13] is expected to compute a proven optimal solution or to generate feasible points and, for a maximization problem, to derive a reasonably tight, nontrivial upper bound. The quality of such bounds is quantified by the integrality gap – the difference between the upper and lower bound. What one considers to be a good-quality solution depends on the problem, the purpose of the model, and the accuracy of the data. A few percent, say 2 to 3%, might be acceptable for the example discussed by Kallrath (2007, Encyclopedia: Planning). However, discussion based on percentage gaps become complicated when the objective function includes penalty terms containing coefficients without a strict economic interpretation. In such cases scaling is problematic. Goal programming as discussed in ([23], p. 294) might help in such situations to avoid penalty terms in the model. The problem is first solved with respect to the highest-priority goal, then one is concerned with the next level goal, and so on.

For practical purposes it is also relevant to observe that solving mixed-integer linear problems and the problem of finding appropriate bounds is often $\mathcal{NP}$-complete, which makes these problems hard to solve. A consequence of this structural property is that these problems scale badly. If the problem can be solved to optimality for a given instance, this might not be so if the size is increased slightly. While tailor-made optimization algorithms such as column generation and branch-and-price techniques can often cope with this

situation for individual problems, it is very difficult for standard software.

We define difficult optimization problems as problems that cannot be solved to optimality or within a reasonable integrality gap by any standard MIP solver within a reasonable time limit. Problem structure, size, or both could lead to such behavior. However, in many cases these problems (typically MIP or nonconvex optimization problems fall into this class) can be solved if they are individually treated, and we resort to the art of modeling.

The art of modeling includes choosing the right level of detail implemented in the model. On the one hand, this needs to satisfy the expectations of the owner of the real-world problem. On the other hand, we are limited by the available computational resources. We give reasons why strict optimality or at least safe bounds are essential when dealing with real-world problems and why we do not accept methods that do not generate both upper and lower bounds.

Mapping the reality also forces us to discuss whether deterministic optimization is sufficient or whether we need to resort to optimization under uncertainty. Another issue is to check whether one objective function suffices or whether multiple-criterion optimization techniques need to be applied.

Instead of solving such difficult problems directly as, for example, a standalone MILP problem, we discuss how problems can be solved equivalently by solving a sequence of models.

Efficient approaches are as follows:

- Column generation with a master and subproblem structure,
- Branch-and-price,
- Exploiting a decomposition structure with a rolling time horizon,
- Exploiting auxiliary problems to generate safe bounds for the original problem, which then makes the original problems more tractable,
- Exhaustion approaches,
- Hybrid methods, i. e., constructive heuristics and local search on subsets of the difficult discrete variables leaving the remaining variables and constraints in tractable MILP or MINLP problems that can be solved.

We illustrate various ideas using real-world planning, scheduling, and cutting-stock problems.

Models and the Art of Modeling

We are here concerned with two aspects of modeling and models. The first one is to obtain a reasonable representation of the reality and mapping it onto a mathematical model, i. e., an optimization problem in the form of (1). The second one is to reformulate the model or problem in such equivalent forms that is is numerically tractable.

Models The terms *modeling* and *model building* are derived from the word *model*. Its etymological roots are the Latin word *modellus* (scale, [diminutive of modus, measure]) and what was to be in the 16th century the new word *modello*. Nowadays, in a scientific context the term is used to refer to a simplified, abstract, or well-structured part of the reality one is interested in. The idea itself and the associated concept is, however, much older. Classical geometry, and especially Pythagoras around 600 B.C., distinguish between wheel and circle and field and rectangle. Around A.D. 1100 a wooden model of the later Speyer cathedral was produced; the model served to build the real cathedral. Astrolabs and celestial globes have been used as models to visualize the movement of the moon, planets, and stars on the celestial sphere and to compute the times of rises and settings. Until the 19th century mechanical models were understood as pictures of reality. Following the principles of classical mechanics the key idea was to reduce all phenomena to the movement of small particles. Nowadays, in physics and other mathematical sciences one will talk about models if

- For reasons of simplification, one restricts oneself to certain aspects of the problem (*example*: if we consider the movement of the planets, in a first approximation the planets are treated as point masses);
- For reasons of didactic presentation, one develops a simplified picture for more complicated reality (*example*: the planetary model is used to explain the situation inside atoms);
- One uses the properties in one area to study the situation in an analogous problem.

A model is referred to as a mathematical model of a process or a problem if it contains typical mathematical objects (variables, terms, relations). Thus, a (mathematical) model represents a real-world problem in

the language of mathematics using mathematical symbols, variables, equations, inequalities, and other relations.

It is very important when building a model to define and state precisely the purpose of the model. In science, we often encounter epistemological arguments. In engineering, a model might be used to construct some machines. In operations research and optimization, models are often used to support strategic or operative decisions. All models enable us to

- Learn and understand situations that do not allow easy access (very slow or fast processes, processes involving a very small or very large region);
- Avoid difficult, expensive, or dangerous experiments; and
- Analyze case studies and *what-if-when scenarios*.

Tailored optimization models can be used to support decisions (that is, the overall purpose of the model). It is essential to have a clear objective describing what a good decision is. The optimization model should produce, for instance, optimal solutions in the following sense:

- To avoid unwanted byproducts as much as possible,
- To minimize costs, or
- to maximize profit, earnings before interest and taxes (EBIT), or contribution margin.

The purpose of a model may change over time.

To solve a real-world problem by mathematical optimization, at first we need to represent our problem by a *mathematical model*, that is, a set of mathematical relationships (e. g., equalities, inequalities, logical conditions) representing an abstraction of our real-world problem. This translation is part of the model-building phase (which is part of the whole modeling process) and is not trivial at all because there is nothing we could consider an exact model. Each model is an acceptable candidate as long as it fulfills its purpose and approximates the real world accurately enough. Usually, a model in mathematical optimization consists of four key objects:

- Data, also called the *constants* of a model;
- Variables (continuous, semicontinuous, binary, integer), also called decision variables;
- Constraints (equalities, inequalities), also called *restrictions*; and
- Objective function (sometimes even several of them).

The data may represent costs or demands, fixed operation conditions of a reactor, capacities of plants, and so on. The variables represent the degrees of freedom, i. e., what we want to decide: how much of a certain product is to be produced, whether a depot is closed or not, or how much material we will store in the inventory for later use. Classical optimization (calculus, variational calculus, optimal control) treats those cases in which the variables represent continuous degrees of freedom, e. g., the temperature in a chemical reactor or the amount of a product to be produced. Mixed-integer optimization involves variables restricted to integer values, for example counts (numbers of containers, ships), decisions (yes-no), or logical relations (if product A is produced, then product B also needs to be produced). The constraints can be a wide range of mathematical relationships: algebraic, analytic, differential, or integral. They may represent mass balances, quality relations, capacity limits, and so on. The objective function expresses our goal: minimize costs, maximize utilization rate, minimize waste, and so on. Mathematical models for optimization usually lead to structured problems such as:

- *Linear programming* (LP) problems,
- *Mixed-integer linear programming* (MILP) problems,
- *Quadratic* (QP) and *mixed-integer quadratic programming* (MIQP),
- *Nonlinear programming* (NLP) problems, and
- *Mixed-integer nonlinear programming* (MINLP) problems.

The Art of Modeling How do we get from a given problem to its mathematical representation? This is a difficult, nonunique process. It is a compromise between the degree of detail required to model a problem and the complexity, which is tractable. However, simplifications should not only be seen as an unavoidable evil. They could be useful for developing understanding or serve as a platform with the client, as the following three examples show.

1. At the beginning of the modeling process it can be useful to start with a "down-scaled" version to develop a feeling for the structure and dependencies of the model. This enable a constructive dialog between the modeler and the client. A vehicle fleet with 100 vehicles and 12 depots could be analyzed with

only 10 vehicles and 2 depots to let the *model world* and the *real world* find each other in a sequence of discussions.
2. In partial or submodels the modeler can develop a deep understanding of certain aspects of the problem which can be relevant to solve the whole problem.
3. Some aspects of the real world problem could be too complicated to model them complete or exactly. During the modeling process it can be clarified, using a smaller version, whether partial aspects of the model could be neglected or whether they are essential.

In any case it is essential that the simplifications be well understood and documented.

Tricks of the Trade for Monolithic Models

Using state-of-the-art commercial solvers, e. g., `XPressMP` [`XPressMP` is by Dash Optimization, http://www.dashoptimization.com] or `CPLEX` [`CPLEX` is by ILOG, http://www.ilog.com], MILP problems can be solved quite efficiently. In the case of MINLP and using global optimization techniques, the solution efficiency depends strongly on the individual problem and the model formulation. However, as stressed in [21] for both MILP and MINLP problem, it is recommended that the full mathematical structure of a problem be exploited, that appropriate reformulations of models be made, and that problem-specific valid inequalities or cuts be used. Software packages may also differ with respect to the ability of *presolving techniques*, *default strategies* for the branch-and-bound algorithm, *cut generation* within the branch-and-cut algorithm, and, last but not least, *diagnosing and tracing infeasibilities*, which is an important issue in practice.

Here we collect a list of recommendation tricks that help to improve the solution procedure of monolithic MIP problems, i. e., standalone models that are solved by one call to a MILP or MINLP solver. Among them are:

- Use bounds instead of constraints if the dual values are not necessarily required.
- Apply one's own presolving techniques. Consider, for instance, a set of inequalities

$$B_{ijk}\delta_{ijk} \le A_{ijk}\,; \quad \forall\{i,j,k\} \tag{2}$$

on binary variables δ_{ijk}. They can be replaced by the bounds

$$\delta_{ijk} = 0\,; \quad \forall\{(i,j,k)\,|A_{ijk} < B_{ijk}\}$$

or, if one does not trust the $<$ in a modeling language, the bounds

$$\delta_{ijk} = 0\,; \quad \forall\{(i,j,k)\,|A_{ijk} \le B_{ijk} - \varepsilon\}$$

where $\varepsilon > 0$ is a small number, say, of the order of 10^{-6}. If $A_{ijk} \ge B_{ijk}$, then (2) is redundant. Note that, due to the fact that we have three indices, the number of inequalities can be very large.
- Exploit the *presolving techniques* embedded in the solver; cf. [28].
- Exploit or eliminate symmetry: sometimes, symmetry can lead to degenerate scenarios. There are situations, for instance, in scheduling where orders can be allocated to identical production units. Another example is the capacity design problem of a set of production units to be added to a production network. In that case, symmetry can be broken by requesting that the capacities of the units be sorted in descending order, i. e., $c_u \ge c_{u+1}$. [29] exploit symmetry in order allocation for stock cutting in the paper industry; this is a very enjoyable paper to read.
- Use special types of variables for which tailor-made branching rules exist (this applies to semicontinuous and partial-integer variables as well as special ordered sets).
- Experiment with the various *strategies* offered by the commercial branch-and-bound solvers for the branch-and-bound algorithm.
- Experiment with the *cut generation* within the commercial branch-and-cut algorithm, among them Gomory cuts, knapsack cuts, or flow cuts; cf. [28].
- Construct one's own valid inequalities for certain substructures of problems at hand. Those inequalities may be added a priori to a model, and in the extreme case they would describe the complete convex hull. As an example we consider the mixed-integer inequality

$$x \le C\lambda\,, \quad 0 \le x \le X\,; \quad x \in \mathbb{R}_0^+\,, \quad \lambda \in \mathbb{N} \tag{3}$$

which has the valid inequality

$$x \le X - G(K-\lambda) \quad \text{where}$$
$$K := \left\lceil \frac{X}{C} \right\rceil \quad \text{and} \quad G := X - C\,(K-1)\,. \tag{4}$$

This valid inequality (4) is the more useful, the more K and X/C deviate. A special case arising is often the situation $\lambda \in \{0, 1\}$. Another example, taken from [39], p. 129 is

$$A_1\alpha_1 + A_2\alpha_2 \le B + x \qquad x \in \mathbb{R}_0^+ \qquad \alpha_1, \alpha_2 \in \mathbb{N} \tag{5}$$

which for $B \notin \mathbb{N}$ leads to the valid inequality

$$\lfloor A_1 \rfloor \alpha_1 + \left(\lfloor A_2 \rfloor \alpha_2 + \frac{f_2 - f}{1 - f} \right) \le \lfloor B \rfloor + \frac{x}{1 - f} \tag{6}$$

where the following abbreviations are used:

$$f := B - \lfloor B \rfloor \,, \qquad f_1 := A_1 - \lfloor A_1 \rfloor \,, \quad f_2 := A_2 - \lfloor A_2 \rfloor \,. \tag{7}$$

The dynamic counterpart of valid inequalities added a priori to a model leads to cutting-plane algorithms that avoid adding a large number of inequalities a priori to the model (note, this can be equivalent to finding the complete convex hull). Instead, only those useful in the vicinity of the optimal solution are added dynamically. For the topics of valid inequalities and cutting-plane algorithms the reader is referred to books by Nemhauser and Wolsey [30], Wolsey [39], and Pochet and Wolsey [32].

- Try disaggregation in MINLP problems. Global optimization techniques are often based on convex underestimators. Univariate functions can be treated easier than multivariate terms. Therefore, it helps to represent bilinear or multilinear terms by their disaggregated equivalences. As an example we consider $x_1 x_2$ with given lower and upper bounds X_i^- and X_i^+ for x_i; $i = 1, 2$. Wherever we encounter $x_1 x_2$ in our model we can replace it by

$$x_1 x_2 = \frac{1}{2}(x_{12}^2 - x_1^2 - x_2^2)$$

and

$$x_{12} = x_1 + x_2 \,.$$

The auxiliary variable is subject to the bounds $X_{12}^- := X_1^- + X_2^-$ and

$$X_{12}^- \le x_{12} \le X_{12}^+ \,, \qquad X_{12}^- := X_1^- + X_2^- \,, \quad X_{12}^+ := X_1^+ + X_2^+ \,.$$

This formulation has another advantage. It allows us to construct easily a relaxed problem which can be used to derive a useful lower bound. Imagine a problem $\mathcal{P}$ with the inequality

$$x_1 x_2 \le A \,. \tag{8}$$

Then

$$x_{12}^2 - X_1^- x_1 - X_2^- x_2 \le 2A \tag{9}$$

is a relaxation of $\mathcal{P}$ as each point (x_1, x_2) satisfying (8) also fulfills (9). Note that an alternative disaggregation avoiding an additional variable is given by

$$x_1 x_2 = \tfrac{1}{4}\left[(x_1 + x_2)^2 - (x_1 - x_2)^2\right] \,.$$

However, all of the creative attempts listed above may not suffice to solve the MIP using one monolithic model. That is when we should start looking at solving the problem by a sequence of problems. We have to keep in mind that to solve a MIP problem we need to derive tight lower and upper bounds with the gap between them approaching zero.

Decomposition Techniques

Decomposition techniques decompose a problem into a set of smaller problems that can be solved in sequence or in any combination. Ideally, the approach can still compute the global optimum. There are standardized techniques such as Benders Decomposition [cf. Floudas ([9], Chap. 6). But often one should exploit the structure of an optimization to construct tailor-made decompositions. This is outlined in the following subsections.

Column Generation

In linear programming parlance, the term *column* usually refers to variables. In the context of column-generation techniques it has wider meaning and stands for any kind of objects involved in an optimization problem. In vehicle routing problems a column might, for instance, represent a subset of orders assigned to a vehicle. In network flow problems a column might represent a feasible path through the network. Finally, in cutting-stock problems [10,11] a column represents a pattern to be cut.

The basic idea of column generation is to decompose a given problem into a master and subproblem. Problems that might otherwise be nonlinear can be completely solved by solving only linear problems. The critical issue is to generate master and subproblems that can both be solved efficiently. One of the most famous examples is the elegant column-generation approach of Gilmore and Gomory [10] for computing the minimal number of rolls to satisfy a requested demand for smaller sized rolls. This problem, if formulated as one monolithic problem, leads to a MINLP problem with a large number of integer variables. In simple cases, such as those described by Schrage ([35], Sect. 11.7), it is possible to generate all columns explicitly, even within a modeling language. Often the decomposition has a natural interpretation. If not all columns can be generated, the columns are added dynamically to the problem. Barnhart et al. [2] give a good overview on such techniques. A more recent review focusing on selected topics of column generation is [25]. In the context of vehicle routing problems, feasible tours contain additional columns as needed by solving a shortest-path problem with time windows and capacity constraints using dynamic programming [7].

More generally, column-generation techniques are used to solve well-structured MILP problems involving a huge number, say, several hundred thousand or millions, of variables, i. e., columns. Such problems lead to large LP problems if the integrality constraints of the integer variables are relaxed. If the LP problem contains so many variables (columns) that it cannot be solved with a direct LP solver (revised simplex, interior point method), one starts solving this so-called *master problem* with a small subset of variables yielding the *restricted master problem*. After the restricted master problem has been solved, a pricing problem is solved to identify new variables. This step corresponds to the identification of a nonbasic variable to be taken into the basis of the simplex algorithm and the term *column generation*. The restricted master problem is solved with the new number of variables. The method terminates when the pricing problems cannot identify any new variables. The simplest version of column generation is found in the Dantzig–Wolfe decomposition [6].

Gilmore and Gomory [10,11] were the first to generalize the idea of dynamic column generation to an integer programming (IP) problem: the cutting-stock problem. In this case, the pricing problem, i. e., the subproblem, is an IP problem itself – and one refers to this as a *column-generationalgorithm*. This problem is special as the columns generated when solving the relaxed master problem are sufficient to get the optimal integer feasible solution of the overall problem. In general this is not so. If not only the subproblem, but also the master problem involves integer variables, then the column-generation part is embedded into a branch-and-bound method; this is called *branch-and-price*. Thus, branch-and-price is integer programming with column generation. Note that during the branching process new columns are generated; therefore the name *branch-and-price*.

Column Generation in cutting-stock Problems This section describes the mathematical model for minimizing the number of roles or trimloss and illustrates the idea of column generation.

Indices used in this model:

$p \in \mathcal{P} := \{p_1, \ldots, p_{N^p}\}$ for cutting patterns (formats).
Either the patterns are directly generated according to a complete enumeration or they are generated by column generation.
$i \in \mathcal{I} := \{i_1, \ldots, i_{N^I}\}$ given orders or widths.

Input Data We arrange the relevant input data size here:

B [L] width of the rolls (raw material roles)
D_i [-] number of orders for the width i
W_i [L] width of order type i

Integer Variables used in the different model variants:

$\mu_p \in \mathbb{N}_0 := \{0, 1, 2, 3, \ldots\}$ [−] indicates how often pattern p is used.
If cutting pattern p is not used, then we have $\mu_p = 0$.
$\alpha_{ip} \in \mathbb{N}_0$ [−] indicates how often width i is contained in pattern p.
This variable can take values between 0 and D_i depending on the order situation.

Model The model contains a suitable object function

$$\min f(\alpha_{ip}, \mu_p)\,,$$

as well as the boundary condition (fulfillment of the demand)

$$\sum_p \alpha_{ip}\mu_p = D_i\,, \quad \forall i \tag{10}$$

and the integrality constraints

$$\begin{aligned} \alpha_{ip} &\in \mathbb{N}_0\,, \quad \forall\{ip\}\,, \\ \mu_p &\in \mathbb{N}_0\,, \quad \forall\{p\}\,. \end{aligned} \tag{11}$$

General Structure of the Problem In this form it is a mixed-integer nonlinear optimization problem (MINLP). This problem class is difficult in itself. More serious is the fact that we may easily encounter several million variables α_{ip}. Therefore the problem cannot be solved in this form.

Solution Method The idea of dynamic column generation is based on the fact that one must decide in a master problem for a predefined set of patterns how often every pattern must be used as well as calculate suitable input data for a subproblem. In this subproblem new patterns are calculated.

The master problem solves for the multiplicities of existing patterns and has the shape

$$\min \sum_p \mu_p\,,$$

with the demand-fulfill inequality (note that it is allowed to produce more than requested)

$$\sum_i N_{ip}\mu_p \geq D_i\,, \quad \forall i \tag{12}$$

and the integrality constraints

$$\mu_p \in \mathbb{N}_0\,, \quad \forall\{p\}\,. \tag{13}$$

The subproblem generates new patterns. Structurally it is a knapsack problem with object function

$$\min_{\alpha_i} 1 - \sum_p P_i\alpha_i\,,$$

where P_i are the dual values (pricing information) of the master problem (pricing problem) associated with (12) and α_i is an integer variable specifying how often width i occurs in the new pattern. We add the knapsack constraint with respect to the width of the rolls

$$\sum_i W_i\alpha_i \leq B\,, \quad \forall i \tag{14}$$

and the integrality constraints

$$\alpha_i \in \mathbb{N}_0\,, \quad \forall\{i\}\,. \tag{15}$$

In some cases, α_i could be additionally bounded by the number, K, of knives.

Implementation Issues The critical issues in this method, in which we alternate in solving the master problem and the subproblem, are the initialization of the procedure (a feasible starting point is to have one requested width in each initial pattern, but this is not necessarily a good one), excluding the generation of the existing pattern by applying integer cuts, and the termination.

Column Enumeration

Column enumeration is a special variant of column generation and is applicable when a small number of columns is sufficient. This is, for instance, the case in real-world cutting-stock problems when it is known that the optimal solution has only a small amount of trimloss. This usually eliminates most of the pattern. Column enumeration naturally leads to a type of selecting columns or partitioning models. A collection of illustrative examples contained in ([35], Sect. 11.7) covers several problems of grouping, matching, covering, partitioning, and packing in which a set of given objects has to be grouped into subsets to maximize or minimize some objective function. Despite the limitations with respect to the number of columns, column enumeration has some advantages:

- No pricing problem,
- Easily applied to MIP problems,
- Column enumeration is much easier to implement.

In the online version of the vehicle routing problem described in [22] it is possible to generate the complete set, C_r, of all columns, i. e., subsets of orders $i \in \mathcal{O}$, $r = |\mathcal{O}|$, assigned to a fleet of n vehicles, $v \in \mathcal{V}$. Let C_r be the union of the sets, C_{rv}, i. e., $C_r = \cup_{v=1\ldots n} C_{rv}$ with $C_r = |C_r| = 2^r n$, where C_{rv}

contains the subsets of orders assigned to vehicle v. Note that C_{rv} contains all subsets containing 1, 2, or r orders assigned to vehicle v. The relevant steps of the algorithm are:

1. Explicitly generate all columns C_{rv}, followed by a simple feasibility test *w.r.t.* the availability of the cars.
2. Solve the routing-scheduling problem for all columns C_{rv} using a tailor-made branch-and-bound approach (the optimal objective function values, $Z(\tau_c)$ or $Z(\tau_{cv})$, respectively, and the associated routing-scheduling plan are stored).
3. Solve the partitioning model:

$$\min_{\gamma_{cv}} \sum_{c=1}^{C_{rv}} \sum_{v=1}^{N^V} Z(\tau_{cv})\gamma_{cv} \,, \tag{16}$$

s.t.

$$\sum_{c=1}^{C_r} \sum_{v=1}^{N^V} I_i(\tau_{cv})\gamma_{cv} = 1 \,, \quad \forall i = 1, \ldots, r \tag{17}$$

ensures that each order is contained exactly once, the inequality

$$\sum_{c=1}^{C_r} \gamma_{cv} \leq 1 \,, \quad \forall v \in \mathcal{V} \,, \tag{18}$$

ensuring that at most one column can exist for each vehicle, and the integrality conditions

$$\gamma_{cv} \in \{0, 1\} \,, \quad \forall c = 1, \ldots, C_r \,. \tag{19}$$

Note that not all combinations of index pairs $\{c, v\}$ exist; each c corresponds to exactly one v, and vice versa. This formulation allows us to find optimal solutions with the defined columns for a smaller number of vehicles. The objective function and the partitioning constraints are just modified by substituting

$$\sum_{v=1|v\in\mathcal{V}}^{N^V} \longrightarrow \sum_{v=1|v\in\mathcal{V}_*}^{N^V} \,,$$

the equations

$$\sum_{c=1}^{C_{rv}} \sum_{v=1|v\in\mathcal{V}_*}^{N^V} I_i(\tau_{cv})\gamma_{cv} = 1 \,, \quad \forall i = 1, \ldots, r \,,$$

and the inequality

$$\sum_{c=1}^{C_{rv}} \gamma_{cv} \leq 1 \,, \quad \forall v \in \mathcal{V}_* \,,$$

where $\mathcal{V}_* \subset \mathcal{V}$ is a subset of the set $\mathcal{V}$ of all vehicles. Alternatively, if it is not prespecified which vehicles should be used but it is only required that not more than N_*^V vehicles be used, then the inequality

$$\sum_{c=1}^{C_r} \sum_{v=1|v\in\mathcal{V}}^{N^V} \gamma_{cv} \leq N_*^V \tag{20}$$

is imposed.

4. Reconstruct the complete solution and extract the complete solution from the stored optimal solutions for the individual columns.

Branch-and-Price

Branch-and-price (often coupled with branch-and-cut) refers to a tailor-made algorithm exploiting the decomposition structure of the problem to be solved. This efficient method for solving MIP problems with column generation has been well described by Barnhart et al. [2] and has been covered by Savelsbergh [34] in the first edition of the *Encyclopedia of Optimization*. Here, we give a list of more recent successful applications in various fields.

- *Cutting stock*: [3,38]
- *Engine routing and industrial in-plant railroads*: [26]
- *Network design*: [16]
- *Lot sizing*: [38]
- *Scheduling (staff planning)*: [8]
- *Scheduling of switching engines*: [24]
- *Supply chain optimization* (pulp industry): [5]
- *Vehicle routing*: [7,15]

Rolling Time Decomposition

The overall methodology for solving the medium-range production scheduling problem is to decompose the large and complex problem into smaller short-term scheduling subproblems in successive time horizons, i. e., we decompose according to time. Large-scale industrial problems have been solved by Janak et al. [18,19]. A decomposition model is formulated and solved to determine the current horizon and

corresponding products that should be included in the current subproblem. According to the solution of the decomposition model, a short-term scheduling model is formulated using the information on customer orders, inventory levels, and processing recipes. The resulting MILP problem is a large-scale complex problem that requires a large computational effort for its solution. When a satisfactory solution is determined, the relevant data are output and the next time horizon is considered. The above procedure is applied iteratively in an automatic fashion until the whole scheduling period under consideration is finished.

Note that the decomposition model determines automatically how many days and products to consider in the small scheduling horizon subject to an upper limit on the complexity of the resulting mathematical model.

An Exhaustion Method

This method combines aspects of a constructive heuristics and of exact model solving. We illustrate the exhausting method by the cutting-stock problem described in Sect. "Column Generation in cutting-stock Problems"; assigning orders in a scheduling problem would be another example. The elegant column generation approach by Gilmore and Gomory [10] is known for producing minimal trimloss solutions with *many* patterns. Often this corresponds to setup changes on the machine and therefore is not desirable. A solution with a minimal number of patterns minimizes the machine setup costs of the cutter. Minimizing simultaneously trimloss and the number of patterns is possible for a small case of a few orders only exploiting the MILP model by Johnston and Salinlija [20]. It contains two conflicting objective functions. Therefore one could resort to goal programming. Alternatively, we could produce several parameterized solutions leading to different numbers of rolls to be used and patterns to be cut from which the user would extract the one he likes best.

As the table above indicates, we compute tight lower bounds on both trimloss and the number of patterns. Even for up to 50 feasible orders, near-optimal solutions are constructed in less than a minute.

Note that it would be possible to use the branch-and-price algorithm described in [38] or [3] to solve the one-dimensional cutting-stock problem with minimal numbers of patterns. However, these methods are not easy to implement. Therefore, we use the following approaches, which are much easier to program:

- V1: Direct usage of the model by Johnston and Salinlija [20] for a small number, say, $N^I \leq 14$, of orders and $D_{\max} \leq 10$. In a preprocessing step we compute valid inequalities as well as tight lower and upper bounds on the variables.
- V2: Exhaustion procedure in which we generate successively new patterns with maximal multiplicities. This method is parameterized by the permissible percentage waste $W_{\max}$, $1 \leq W_{\max} \leq 99$. After a few patterns have been generated with this parameterization, it could happen that is is not possible to generate any more patterns with waste restriction. In this case the remaining unsatisfied orders are generated by V1 without the $W_{\max}$ restriction.

Indices and Sets

In this model we use the indices listed in Johnston and Salinlija [20]:

$i \in \mathcal{I} := \{i_1, \ldots, i_{N^I}\}$ denotes the sets of width.

$j \in \mathcal{J} := \{j_1, \ldots, j_{N^P}\}$ denotes the pattern; $N^J \leq N^I$.
The patterns are generated by V1, or dynamically by maximizing the multiplicities of a used pattern.

$k \in \mathcal{K} := \{k_1, \ldots, k_{N^P}\}$ denotes the multiplicity index to indicate how often a width is used in a pattern.
The multiplicity index can be restricted by the ratio of the width of the orders and the width of the given rolls.

Variables

The following integer or binary variables are used:

$a_{ijk} \in \mathbb{N}$ [−] specifies the multiplicity of pattern j.
The multiplicity can vary between 0 and $D_{\max} := \max\{D_i\}$. If pattern j is not used, we have $r_j = p_j = 0$.

$p_j \in \{0, 1\}$ [−] indicates whether pattern j is used at all.

$r_j \in \mathbb{N}$ [−] specifies how often pattern j is used.
The multiplicity can vary between 0 and $D_{\max} := \max\{D_i\}$. If pattern j is not used, we have $r_j = p_j = 0$.

$\alpha_{ip} \in \mathbb{N}$ [−] specifies how often width i occurs in pattern p.

```
 # of  #  output file  flag Wmax           comment
rolls pat
-------------------------------------------------------------------
  0    5                 8   99   lower bound: minimal # of patterns
  30  10   pat00.out     9   99   lower bound: minimal # of rolls
  34   7   pat01.out     0   20
  31   9   pat02.out     1   15
  30   8   pat03.out     0   10   minimal number of rolls
  32   9   pat04.out     1    8
  30   8   pat05.out     0    6   minimal number of rolls
  31   8   pat06.out     1    4

 The best solution found contains 7 patterns!
 The solution with minimal trimloss contain 30 rolls!

 Improvement in the lower bound of pattern: 6!
 Solutions with 6 patterns are minimal w.r.t.
 to the number of patterns.

 A new solution was found with only 6 patterns and 36 rolls: patnew.out
  36   6   patnew.out  0  99
```

This width-multiplicity variable can take all values between 0 and D_i.

$x_{ijk} \in \{0,1\}$ [−] indicates whether width i appears in pattern j at level k.

Note that $x_{ijk} = 0$ implies $a_{ijk} = 0$.

The Idea of the Exhaustion Method

In each iteration we generate m at most two or three new patterns by maximizing the multiplicities of these patterns, allowing no more than a maximum waste, $W_{\max}$. The solution generated in iteration m is preserved in iteration $m+1$ by fixing the appropriate variables. If the problem turns out to be infeasible (this may happen if $W_{\max}$ turns out to be restrictive), then we switch to a model variant in which we minimize the number of patterns subject to satisfying the remaining unsatisfied orders.

The model is based on the inequalities (2,3,5,6,7,8,9) in [20], but we add a few more additional ones or modify the existing ones. We exploit two objective functions: maximizing the multiplicities of the patterns generated

$$\max \sum_{j=1}^{\pi_u} r_j \,,$$

where π_u specifies the maximal number of patterns (π_u could be taken from the solution of the column-generation approach, for instance), or minimizing the number of patterns generated

$$\min \sum_{j=1}^{\pi_u} p_j \,.$$

The model is completed by the integrality conditions

$$r_j, a_{ijk} \in \{0,1,2,3,\ldots\} \tag{21}$$

$$p_j, x_{ijk}, y_{jk} \in \{0,1\} \,. \tag{22}$$

The model is applied several times with $a_{ijk} \le \tilde{D}_i$, where $\tilde{D}_i$ is the number of remaining orders of width i. In particular, the model has to fulfill the relationships

$$k a_{ijk} > \tilde{D}_i \quad \Longrightarrow \quad a_{ijk} = 0 \quad , \quad x_{ijk} = 0$$

and

$$a_{ijk} \le \left\lceil \frac{\tilde{D}_i}{k} \right\rceil \quad \text{or} \quad a_{ijk} \le \left\lceil \frac{\tilde{D}_i + S_i}{k} \right\rceil \,,$$

where S_i denotes the permissible overproduction.

The constructive method described so far provides an improved upper bound, π'_u, on the number of pattern.

Computing Lower Bounds

To compute a lower bound we apply two methods. The first method is to solve a bin-packing problem, which is equivalent to minimizing the number of rolls in the original cutting-stock problem described in the Sect. "Column Generation in Cutting-Stock Problems" for equal demands $D_i = 1$. If solved with the column-generation approach, this method is fast and cheap, but the lower bound, π'_l, is often weak. The second method is to exploit the upper bound, π'_u, on the number of patterns obtained and to call the exact model as in V1. It is impressive how quickly the commercial solvers `CPLEX` and `XpressMP` improve the lower bound yielding π''_l. For most examples with up to 50 orders we obtain $\pi'_u - \pi'_l \leq 2$, but in many cases $\pi'_u - \pi'_l = 1$ or even $\pi'_u = \pi''_l$.

Primal Feasible Solutions and Hybrid Methods

We define hybrid methods as methods based on any combination of exact MIP methods with constructive heuristics, local search, metaheuristics, or constraint programming that produces primal feasible solutions. Dive-and-fix, near-integer-fix, and fix-and-relax are such hybrid methods. They are user-developed heuristics exploiting the problem structure. In their kernel they use a declarative model solved, for instance, by `CPLEX` and `XpressMP`.

In constructive heuristics we exploit the structure of the problem and compute a feasible point. Once we have a feasible point we can derive safe bounds on the optimum and assign initial values to the critical discrete variable, which could be exploited by the `GAMS/CPLEX` *mipstart* option. Feasible points can sometimes be generated by appropriate sequences of relaxed models. For instance, in a scheduling problem $\mathcal{P}$ with due times one might relax these due times obtaining the relaxed model $\mathcal{R}$. The optimal solution, or even any feasible point of $\mathcal{R}$, is a feasible point of $\mathcal{P}$ if the due times are models with appropriate unbounded slack variables.

Constructive heuristics can also be established by systematic approaches of fixing critical discrete variables. Such approaches are *dive-and-fix* and *relax-and-fix*. In dive-and-fix the LP relaxation of an integer problem is to be solved followed by fixing a subset of fractional variables to suitable bounds. *Near-integer-fix* is a variant of dive-and-fix that fixes variables with fractional values to the nearest integer point. Note that these heuristics are subject to the risk of becoming infeasible.

The probability of becoming infeasible is less likely in relax-and-fix. In relax-and-fix, following Pochet and Wolsey ([32], pp. 109) we suppose that the binary variables δ of a MIP problem $\mathcal{P}$ can be partitioned into R disjoint sets $S^1; \ldots; S^R$ of decreasing importance. Within these subsets U^r with $U \subseteq \cup_{u=r+1}^{R} S^u$ for $r = 1; \ldots; R-1$ can be chosen to allow for somewhat more generality. Based on these partitions, R MIP problems are solved, denoted $\mathcal{P}^r$ with $1 \leq r \leq R$ to find a heuristic solution to $\mathcal{P}$. For instance in a production planning problem, S^1 might be all the δ variables associated with time periods in $\{1, \ldots, t_1\}$, S^u those associated with periods in $\{t_u + 1, \ldots, t_{u+1}\}$, whereas U^r would would be the δ variables associated with the periods in some set $\{t_r + 1, \ldots, u_r\}$.

In the first problem, $\mathcal{P}^1$, one only imposes the integrality of the important variables in $S^1 \cup U^1$ and relaxes the integrality on all the other variables in S. As $\mathcal{P}^1$ is a relaxation of $\mathcal{P}$, for a minimization problem, the solution of $\mathcal{P}^1$ provides a lower bound of $\mathcal{P}$. The solution values, δ^1, of the discrete variables are kept fixed when solving $\mathcal{P}^r$. This continues and in the subsequent $\mathcal{P}^r$, for $2 \leq r \leq R$, we additionally fix the values of the δ variables with index in S^{r-1} at their optimal values from $\mathcal{P}^{r-1}$ and add the integrality restriction for the variables in $S^r \cup U^r$.

Either $\mathcal{P}^r$ is infeasible for some $r \in \{1, \ldots, R\}$, and the heuristic failed, or else (x^R, δ^R) is a relax-and-fix solution. To avoid infeasibilities one might apply a smoothed form of this heuristic that allows for some overlap of U^{r-1} and U^r. Additional free binary variables in horizon $r-1$ allow one to link the current horizon r with the previous one. Usually this suffices to ensure feasibility. Relax-and-fix comes in various flavors exploiting time-decomposition or time-partitioning structures. Other decompositions, for instance plants, products, or customers, are possible as well.

A local search can be used to improve the solution obtained by the relax-and-fix heuristic. The main idea is to solve repeatedly the subproblem on a small number of binary variables reoptimizing, for instance, the production of some products. The binary variables for resolving could be chosen randomly or by a metaheuristic

such as simulated annealing. All binary variables related to them are released; the others are fixed to the previous best values.

Another class of MIP hybrid method is established by algorithms that combine a MIP solver with another algorithmic method. A hybrid method obtained by the combination of mixed-integer and constraint logic programming strategies has been developed and applied by Harjunkoski et al. [14] as well as Jain and Grossmann [17] for solving scheduling and combinatorial optimization problems. Timpe [37] solved mixed planning and scheduling problems with mixed MILP branch-and-bound and constraint programming. Maravelias and Grossmann [27] proposed a hybrid/decomposiiton algorithm for the short-term scheduling of batch plants, and Roe et al. [33] presented a hybrid MILP/CLP algorithm for multipurpose batch process scheduling in which MILP is used to solve an aggregated planning problem while CP is used to solve a sequencing problem. Other hybrid algorithms combine evolutionary and mathematical programming methods; see, for instance, the heuristics by Till et al. [36] for stochastic scheduling problems and by Borisovsky et al. [4] for supply management problems.

Finally, one should not forget to add some algorithmic component that, for the minimization problem at hand, would generate some reasonable bounds to be provided in addition to the hybrid method. The hybrid methods discussed above provide upper bounds by constructing feasible points. In favorite cases, the MIP part of the hybrid solver provides lower bounds. In other case, lower bounds can be derived from auxiliary problems, which are relaxations of the original problem, and which are easier to solve.

Summary

If a given MIP problem cannot be solved by an available MIP solver exploiting all its internal presolving techniques, one might reformulate the problem and obtain an equivalent or closely related representation of reality. Another approach is to construct MIP solutions and bounds by solving a sequence of models. Alternatively, individual tailor-made exact decomposition techniques could help as well as primal heuristics such as relax-and-fix or local search techniques on top of a MIP model.

References

1. Anstreicher KM (2001) Linear Programming: Interior Point Methods. In: Floudas CA, Pardalos P (eds) Encyclopedia of Optimization, vol 3. Kluwer, Dordrecht, pp 189–191
2. Barnhart C, Johnson EL, Nemhauser GL, Savelsberg MWP, Vance PH (1998) Branch-and-price: column generation for solving huge integer programs. Oper Res 46(3):316–329
3. Belov G, Scheithauer G (2006) A Branch-and-Crice Algorithm for One-Dimensional Stock Cutting and Two-Dimensional Two-Stage Cutting. Eur J Oper Res 171:85–106
4. Borisovsky P, Dolgui A, Eremeev A (2006) Genetic Algorithms for Supply Management Problem with Lower-bounded Demands. In: Dolgui A, Morel G, Pereira C (eds) Information Control Problems in Manufacturing 2006: A Proceedings volume from the 12th IFAC International Symposium. vol. 3., St Etienne, France, 17-19 May 2006. North-Holland, Dordrecht, pp 521–526
5. Bredström D, Lundgren JT, Rönnqvist M, Carlsson D, Mason A (2004) Supply Chain Optimization in the Pulp Mill Industry – IP models, Column Generation and Novel Constraint Branches. Eur J Oper Res 156:2–22
6. Dantzig B, Wolfe P (1960) The decomposition algorithm for linear programming. Oper Res 8:101–111
7. Desrochers M, Desrosiers J, Solomon MM (1992) A New Optimization Algorithm for the Vehicle Routing Problem with time Windows. Oper Res 40(2):342–354
8. Eveborn P, Ronnqvist M (2004) Scheduler – A System for Staff Planning. Ann Oper Res 128:21–45
9. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, Oxford
10. Gilmore PC, Gomory RE (1961) A Linear Programming Approach to the Cutting Stock Problem. Oper Res 9:849–859
11. Gilmore PC, Gomory RE (1963) A Linear Programming Approach to the Cutting Stock Problem, Part II. Oper Res 11:863–888
12. Grötschel M (2004) Mathematische Optimierung im industriellen Einsatz. In: Lecture at Siemens AG, Munich, Germany, 7 Dec 2004
13. Grötschel M (2005) Private communication
14. Harjunkoski I, Jain V, Grossmann IE (2000) Hybrid Mixed-integer constraint Logic Programming Strategies for Solving Scheduling and Combinatorial Optimization Problems. Comput Chem Eng 24:337–343
15. Irnich S (2000) A Multi-Depot Pickup and Delivery Problem with a Single Hub and Heterogeneous Vehicles. Eur J Oper Res 122:310–328
16. Irnich S (2002) Netzwerk-Design für zweistufige Transportsysteme und ein Branch-and-Price-Verfahren für das gemischte Direkt- und Hubflugproblem. Dissertation, Fakultät für Wirtschaftswissenschaften, RWTH Aachen, Aachen, Germany

17. Jain V, Grossmann IE (2001) Algorithms for hybrid MILP/CP models for a class of optimization problems. IFORMS J Comput 13:258–276
18. Janak SL, Floudas CA, Kallrath J, Vormbrock N (2006) Production Scheduling of a Large-Scale Industrial Batch Plant: I. Short-Term and Medium-Term Scheduling. Ind Eng Chem Res 45:8234–8252
19. Janak SL, Floudas CA, Kallrath J, Vormbrock N (2006) Production Scheduling of a Large-Scale Industrial Batch Plant: II. Reactive Scheduling. Ind Eng Chem Res 45:8253–8269
20. Johnston RE, Sadinlija E (2004) A New Model for Complete Solutions to One-Dimensional Cutting Stock Problems. Eur J Oper Res 153:176–183
21. Kallrath J (2000) Mixed Integer Optimization in the Chemical Process Industry: Experience, Potential and Future Perspectives. Chem Eng Res Des 78(6):809–822
22. Kallrath J (2004) Online Storage Systems and Transportation Problems with Applications: Optimization Models and Mathematical Solutions, vol 91 of Applied Optimization. Kluwer, Dordrecht
23. Kallrath J, Maindl TI (2006) Real Optimization with SAP-APO. Springer, Berlin
24. Lübbecke M, Zimmermann U (2003) Computer aided scheduling of switching engines. In: Jäger W, Krebs HJ (eds) Mathematics – Key Technology for the Future: Joint Projects Between Universities and Industry. Springer, Berlin, pp 690–702
25. Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. Oper Res 53(6):1007–1023
26. Lübbecke ME, Zimmermann UT (2003) Engine routing and scheduling at industrial in-plant railroads. Transp Sci 37(2):183–197
27. Maravelias CT, Grossmann IE (2004) A Hybrid MILP/CP Decomposition Approach for the Continuous Time Scheduling of Multipurpose Batch Plants. Comput Chem Eng 28:1921–1949
28. Martin A (2001) General Mixed Integer Programming: Computational Issues for Branch-and-Cut Algorithms. In: Naddef D, Juenger M (eds) Computational Combinatorial Optimization. Springer, Berlin, pp 1–25
29. Menon S, Schrage L (2002) Order Allocation for Stock Cutting in the Paper Industry. Oper Res 50(2):324–332
30. Nemhauser GL, Wolsey LA (1988) Integer and Combinatorial Optimization. Wiley, New York
31. Pardalos PM (2001) Linear Programming. In: Floudas CA, Pardalos P (eds) Encyclopedia of Optimization, vol 3. Kluwer, Dordrecht, pp 186–188
32. Pochet Y, Wolsey LA (2006) Production Planning by Mixed Integer Programming. Springer, Berlin
33. Roe B, Papageorgiou LG, Shah N (2005) A hybrid MILP/CLP Algorithm for Multipurpose Batch Process schedulin. Comput Chem Eng 29:1277–1291
34. Savelsbergh MWP (2001) Branch-and-Price: Integer Programming with Column Generation. In: Floudas CA, Pardalos P (eds) Encyclopedia of Optimization. Kluwer, Dordrecht, pp 218–221
35. Schrage L (2006) Optimization Modeling with LINGO. LINDO Systems, Chicago
36. Till J, Sand G, Engell S, Emmerich M, Schönemann L (2005) A New Hybrid Algorithm for Solving Two-Stage Stochastic Problems by Combining Evolutionary and Mathematical Programming Methods. In: Puigjaner L, Espuña A (eds) Proc. European Symposium on Computer Aided Process Engineering (ESCAPE) - 15. Dordrecht, North-Holland, pp 187–192
37. Timpe C (2002) Solving mixed planning and scheduling problems with mixed branch and bound and constraint programming. OR Spectrum 24:431–448
38. Vanderbeck F (2000) Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem. Oper Res 48(5):915–926
39. Wolsey LA (1998) Integer Programming. Wiley, New York

Modeling Languages in Optimization: A New Paradigm

Tony Hürlimann
Institute Informatics, University Fribourg, Fribourg, Switzerland

MSC2000: 90C10, 90C30

Article Outline

Keywords
Why Declarative Representation
Algebraic Modeling Languages
Second Generation Modeling Languages
Modeling Language and Constraint Logic Programming
Modeling Examples
  Sorting
  The *n*-Queens Problem
  A Two-Person Game
  Equal Circles in a Square
  The (Fractional) Cutting-Stock Problem
Conclusion
See also
References

Keywords

Algorithmic language; Declarative language; Modeling language; Solver

In this paper, modeling languages are identified as a new computer language paradigm and their applications for representing optimization problems is illustrated by examples.

Programming languages can be classified into three paradigms: imperative, functional, and logic programming [14]. The *imperative programming paradigm* is closely related to the physical way of how (the von Neumann) computer works: Given a set of memory locations, a program is a sequence of well defined instructions on retrieving, storing and transforming the content of these locations. The *functional paradigm* of computation is based on the evaluation of functions. Every program can be viewed as a function which translates an input into a unique output. Functions are first-class values, that is, they must be viewed as values themselves. The computational model is based on the λ-calculus invented by A. Church (1936) as a mathematical formalism for expressing the concept of a computation. The *paradigm of logic programming* is based on the insight that a computation can be viewed as a kind of (constructive) proof. Hence, a program is a notation for writing logical statements together with specified algorithms for implementing inference rules.

All three programming paradigms concentrate on problem representation as a *computation*, that is, the problem is stated in a way that describes the process of solving it. The computation on how to solve a problem 'is' its representation. One may call such a notational system an *algorithmic language*.

Definition 1 An *algorithmic language* describes (explicitly or implicitly) the computation of solving a problem, that is, 'how' a problem can be processed using a machine. The computation consists of a sequence of well-defined instructions which can be executed in a finite time by a Turing machine. The information of a problem which is captured by an algorithmic language is called *algorithmic knowledge* of the problem.

Algorithmic knowledge to describe a problem is very common in our everyday life – one only need to look at cookery-books, or technical maintenance manuals – that one may ask whether the human brain is 'predisposed' to preferably present a problem in describing its solution recipe.

However, there exists at least one different way to capture knowledge about a problem; it is the method which describes 'what' the problem is by defining its properties, rather than saying 'how' to solve it. Mathematically, this can be expressed by a set $\{x \in X: R(x)\}$, where X is a continuous or discrete state space and $R(x)$ is a Boolean relation, defining the properties or the *constraints* of the problem; x is called the *variable*(s). A notational system that represents a problem in this way is called a *declarative language*.

Definition 2 A *declarative language* describes the problem as a set using mathematical variables and constraints defined over a given state space. This space can be finite or infinite, countable or noncountable. The information of a problem which is captured by a declarative language is called *declarative knowledge* of the problem.

The declarative representation, in general, does not give any indication on how to solve the problem. It only states what the problem is. Of course, there exists a trivial algorithm to solve a declaratively stated problem, which is to enumerate the state space and to check whether a given $x \in X$ violates the constraint $R(x)$. The algorithm breaks down, however, whenever the state space is infinite. But even if the state space is finite, it is – for most nontrivial problems – so large that a full enumeration is practically impossible.

Algorithmic and declarative representations are two fundamentally different kinds of modeling and representing knowledge. Declarative knowledge answers the question 'what is?', whereas algorithmic knowledge asks 'how to?' [4]. An algorithm gives an exact recipe of how to solve a problem. A mathematical model, i. e. its declarative representation, on the other hand, (only) defines the problem as a subspace of the state space. No algorithm is given to find all or a single element of the feasible subspace.

Why Declarative Representation

The question arises, therefore, why to present a problem using a declarative way, since one must solve it anyway and, hence, represent as an algorithm? The reasons are, first of all, *conciseness*, *insight*, and *documentation*. Many problems can be represented declaratively in a very concise way, while the representation of their computation is long and complex. Concise writings favor also the insight of a problem. Furthermore, in many scientific papers a problem is stated in a declarative

way using mathematical equations and inequalities for documentational purposes. This gives a clear statement of the problem and is an efficient way to communicate it to other scientists. However, documentation is by no means limited to human beings. One can imagine declarative languages implemented on a computer like algorithmic languages, which are parsed and interpreted by a compiler. In this way, an interpretative system can *analyse the structure* of a declarative program, can *pretty-print* it on a printer or a screen, can *classify* it, or *symbolically transform* it in order to view it as a diagram or in another textual form.

Of course, the most interesting question is whether the declarative way of representing a problem could be of any help in *solving* the problem.

Indeed, for certain classes of problems the computation can be obtained directly from a declarative formulation. This is true for all recursive definitions. A classical example is the algorithm of Euclid to find the greatest common divisor (gcd) of two integers. One can proof that

$$\gcd(a, b) = \begin{cases} \gcd(b, a \bmod b), & b > 0 \\ a, & b = 0, \end{cases}$$

which is clearly a declarative statement of the problem. In *Scheme*, a functional language, this formula can be implemented directly as a function in the following way:

```
(define (gcd a b)
    (if(= b 0) a
        (gcd b (remainder a b))))
```

Similar formulations can be given for any other language which includes recursion as a basic control structure. This class of problems is surprisingly rich. The whole paradigm of dynamic programming can be subsumed under this class.

A class of problems of a very different kind are linear programs, which can be represented declaratively in the following way:

$$\{\min cx\colon\ Ax \geq b\}$$

From this formulation – in contrast to the class of recursive definitions – nothing can be deduced that would be useful in solving the problem. However, there exists well-known methods, for example the simplex method, which solves almost all instances in a very efficient way. Hence, to make the declarative formulation of a linear program useful for solving it, one only needs to translate it into a form, the simplex algorithm accepts as input. The translation from the declarative formulation $\{\min cx\colon Ax \geq b\}$ to such an input-form can be automated. This concept can be extended to nonlinear and discrete problems.

Algebraic Modeling Languages

The idea to state the mathematical problem in a declarative way and to translate it into an 'algorithmic' form by a standard procedure led to a new language paradigm emerged basically in the community of operations research at the end of the 1980s, the *algebraic modeling languages* (AIMMS [1], AMPL [7], GAMS [2], LINGO [18], and LPL [12] and others). These languages are becoming increasingly popular even outside the community of operations research. Algebraic modeling languages represent a problem in a purely declarative way, although most of them include computational facilities to manipulate the data as well as certain control structures.

One of their strength is the complete separation of the problem formulation as a declarative model from finding a solution, which is supposed to be computed by an external program called a *solver*. This allows the modeler not only to separate the two main tasks of model formulation and model solution, but also to switch easily between several solvers. This is an invaluable benefit for many difficult problems, since it is not uncommon that a model instance can be solved using one method, and another instance is solvable only using another method. Another advantage of such languages is to separate clearly between model structure, which only contains parameters (place-holder for data) but no data, and model instance, in which the parameters are replaced by a specific data set. This leads to a natural separation between model formulation and data gathering stored in databases. Hence, the main features of these algebraic modeling languages are:

- purely declarative representation of the problem;
- clear separation between formulation and solution;
- clear separation between model structure and model data.

It is, however, naive to think that one only needs to formulate a problem in a concise declarative form and to link it somehow to a solver in order to solve it. First of all, the 'linking process' is not so straightforward as it seems initially. Second, a solver may not exist which could solve the problem at hand in an efficient way. One only needs to look at Fermat's last conjecture which can be stated in a declarative way as $\{a, b, c, n \in \mathbf{N}^+: a^n + b^n = c^n, a, b, c \geq 1, n > 2\}$ to convince oneself of this fact. Even worse, one can state a problem declaratively for which no solver can exist. This is true already for the rather limited declarative language of first order logic, for which no algorithm exists which decides whether a formula is true or false in general (see [5]).

In this sense, efforts are under way actually in the design of such languages which focus on flexibly linking the declarative formulation to a specific solver to make this paradigm of purely declarative formulation more powerful. This language-solver-interface problem has different aspects and research goes in many directions. A main effort is to integrate symbolic model transformation rules into the declarative language in order to generate formulations which are more useful for a solver. AMPL, for example, automatically detects partially separable structure and computes second derivatives [8]. This information are also handed over to a nonlinear solver. LPL, to cite a very different undertaking, has integrated a set of rules to translate *symbolically* logical constraints into 0–1 constraint [11]. To do this in an intelligent way is all but easy, because the resulting 0–1 formulation should be as sharp as possible. This translation is useful for large mathematical models which must be extended by a few logical conditions. For many applications the original model becomes straightforward while the transformed is complicated but still relatively easy to solve (examples were given in [11]). Even if the resulting formulation is not solvable efficiently, the modeler can gain more insights into the structure of the model from such a symbolic translation procedure, and eventually modify the original formulation.

Second Generation Modeling Languages

Another research activity, actually under way, goes in the direction of extending the algebraic modeling languages in order to express also algorithmic knowledge. This is necessary, because even if one could link an purely declarative language to any solver, it remains doubtful of whether this can be done efficiently in all cases. Furthermore, for many problems it is not useful to formulate them in a declarative way: the algorithmic way is more straightforward and easier to understand. For still other problems a mixture of declarative and algorithmic knowledge leads to a superior formulation in terms of understandability as well as in terms of efficiency, (examples are given below to confirm this findings).

Therefore, AIMMS integrates control structures and procedure definitions. GAMS, AMPL and LPL also allow the modeler to write algorithms powerful enough to solves models repeatedly.

A theoretical effort was undertaken in [10] to specify a modeling language which allows the modeler (or the programmer) to combine algorithmic and declarative knowledge within the same language framework without intermingle them. The overall syntax structure of a model (or a program) in this framework is as follows:

```
MODEL ModelName
    ⟨declarative part of the model⟩
BEGIN
    ⟨algorithmic part of the model⟩
END ModelName.
```

Declarative and algorithmic knowledge are clearly separated. Either part can be empty, meaning that the problem is represented in a purely declarative or in a purely algorithmic form. The declarative part consists of the basic building blocks of declarative knowledge: variables, parameters, constraints, model checking facilities, and sets (that is a way to 'multiply' basic building blocks). This part may also contain 'ordinary declarations' of an algorithmic language (e. g., type and function declarations). Furthermore, one can declare whole models within this part, leading to nested model structures, which is very useful in decomposing a complex problem into smaller parts. The algorithmic part, on the other hand, consists of all control structures which make the language Turing complete. One may imagine his or her favorite programming language being implemented in this part. A language which combines declar-

ative and algorithmic knowledge in this way is called *modeling language*.

Definition 3 A *modeling language* is a notational system which allows one to combine (not to merge) declarative and algorithmic knowledge in the same language framework. The content captured by such a notation is called a *model*.

Such a language framework is very flexible. Purely declarative models are linked to external solvers to be solved; purely algorithmic models are programs, that is algorithms + data structures, in the ordinary sense.

Modeling Language and Constraint Logic Programming

Merging declarative and algorithmic knowledge is not new, although it is not very common in language design. The only existing language paradigm doing it is *constraint logic programming* (CLP), a refinement of logic programming [13]. There are, however, important differences between the CLP paradigm and the paradigm of modeling language as defined above.

1) In CLP the algorithmic part – normally a search mechanism – is behind the scene and the computation is intrinsically coupled with the declarative language itself. This could be a strength because the programmer does not have to be aware of how the computation is taking place, he or she only writes the rules in a descriptive, that is declarative, way and triggers the computation by a request. In reality, however, it is an important drawback, because – for most nontrivial problem – the programmer 'must' be aware on how the computation is taking place. Therefore, to guide the computation in CLP, the declarative program is interspersed with additional rules which have nothing to do with the description of the original problem. In a modeling language, the user either links the declarative part to an external solver or writes the solver within the language. In either case, both parts are strictly separated. Why is this separation so important? Because it allows the modeler to 'plug in' different solvers without touching the overall model formulation.
2) The second difference is that the modeling language paradigm lead automatically to modular design. This is probably to hottest topic in software engineering: building components. Software engineering teaches us that a complex structure can be only managed efficiently by break it down into many relatively independent components. The CLP approach leads more likely to programs that are difficult to survey and hard to debug and to maintain, because such considerations are entirely absent within the CLP paradigm.
3) On the other hand, the community of CLP has developed methods to solve specific classes of combinatorial problems which seems to be superior to other methods. This is because they rely on propagation, simplification of constraints, and various consistency techniques. In this sense, CLP solvers could be used and linked with modeling languages. Such a project is actually under way between the AMPL language and the ILOG solver [6,17].

Hence, while the *representation of models* is probably best done in the language framework of modeling languages, the solution process can taken place in a CLP solver for certain problems.

Modeling Examples

Five modeling examples are chosen from very different problem domains to illustrate the highlights of the presented paradigm of modeling language. The first two examples show that certain problems are best formulated using algorithmic knowledge, the next two examples show the power of a declarative formulation, and a last example indicates that mixing both paradigms is sometimes more advantageous.

Sorting

Sorting is a problem which is preferably expressed in an algorithmic way. Declaratively, the problem could be formulated as follows: Find a permutation π such that $A_{\pi i} \leq A_{\pi i+1}$ for all $i \in \{1, \ldots, n-1\}$ where $A_1, \ldots, n$ is an array of objects on which an order is defined. It is difficult to imagine a 'solver' that could solve this problem as efficiently as the best known sorting algorithms such as Quicksort, of which the implementation is straightforward.

The reason why the sorting problem is best formulated as an algorithm is probably that the state space is exponential in the number of items, however, the best algorithm only has complexity $O(n \log n)$.

The *n*-Queens Problem

The n-queens problem is to place n queens on a chessboard of dimension $n \times n$ in such a way, that they cannot beat each other. This problem can be formulated declarative as follows: $\{x_i, x_j \in \{1, \ldots, n\}: x_i \neq x_j, x_i + i \neq x_j + j, x_i - i \neq x_j - j\}$, where x_i is the column position of the ith queen (i. e. the queen in row i).

Using the LPL [12] formulation:

```
MODEL nQueens;
    PARAMETER n; SET i ALIAS j::={1, ..., n};
    DISTINCT VARIABLE x{i}[1, ..., n];
    CONSTRAINT S{i, j : i < j}:
        x[i] + i <> x[j] + j AND x[i] - i <> x[j] - j;
END
```

the author was able to solve problems for $n \leq 8$ using a general MIP solver. The problem is automatically translated into a 0–1 problem by LPL. Replacing the MIP-solver by a tabu search heuristic, problems with $n \leq 50$ were solvable within the LPL framework. Using the constraint language OZ [19] problems of $n \leq 200$ are efficiently solvable using techniques of propagation and variable domain reductions. However, the success of all these methods seems to be limited compared to the best we can attain. In [20,21], Sosic Rok and Gu Jun presented a polynomial time local heuristic that can solve problems of $n \leq 3\,000\,000$ in less than one minute. The presented algorithm is very simple. The conclusion seems to be for the n-queens problem that an algorithmic formulation is advantageous.

A Two-Person Game

Two players choose at random a positive number and note it on a piece of paper. They then compare them. If both numbers are equal, then neither player gets a payoff. If the difference between the two numbers is one, then the player who has chosen the higher number obtains the sum of both; otherwise the player who has chosen the smaller number obtains the sum of both. What is the optimal strategy for a player, i. e. which numbers should be chosen with what frequencies to get the maximal payoff? This problem was presented in [9] and is a typical two-person zero-sum game. In LPL, it can be formulated as follows:

```
ODEL Game 'finite two-person zero-sum game';
    SET i ALIAS j := /1 : 50/;
    PARAMETER p{i, j} := IF(j > i, IF(j = i + 1,
    -i - j, MIN(i, j)), IF(j < i, -p[j, i], 0));
    VARIABLE x{i};
    CONSTRAINT R : SUM{i} x[i] = 1;
    MAXIMIZE gain: MIN{j}(SUM{i}p[j, i] * x[i]);
END Game.
```

This is an very compact way to declaratively formulate the problem and it is difficult to imagine how this could be achieved using algorithmic knowledge alone. It is also an efficient way to state the problem, because large instances can be solved by an linear programming solver. LPL automatically transforms it into an linear program. (By the way, the problem has an interesting solution: Each player should only choose number smaller than six.)

Equal Circles in a Square

The problem is to find the maximum diameter of n equal mutually disjoint circles packed inside a unit square.

In LPL, this problem can be compactly formulated as follows:

```
MODEL circles 'pack equal circles in a square';
PARAMETER n 'number of circles';
SET i ALIAS j = 1, ..., n;
VARIABLE
    t 'diameter of the circles';
    x{i}[0, 1] 'x-position of the center';
    y{i}[0, 1] 'y-position of the center';
CONSTRAINT
    R{i, j : i < j} 'circles must be disjoint':
        (x[i] - x[j])^2 + (y[i] - y[j])^2 >= t;
MAXIMIZE obj 'maximize diameter': t;
END
```

C.D. Maranas et al. [15] obtained the best known solutions for all $n \leq 30$ and, for $n = 15$, an even better one

using an equivalent formulation in GAMS and linking it to MINOS [16], an well-known nonlinear solver.

The (Fractional) Cutting-Stock Problem

Paper is manufactured in rolls of width B. A set of customers W orders d_w rolls of width b_w(with $w \in W$). Rolls can be cut in many ways, every subset $P' \subseteq W$ such that $\sum_{i \in P'} y_i\, b_i \leq B$ is a possible cut-pattern, where y_i is a positive integer. The question is how the initial roll of width B should be cut, that is, which patterns should be used, in order to minimize the overall paper waste. A straightforward formulation of this problem is to enumerate all patterns, each giving a variable, then to minimize the number of used patterns while fulfilling the demands. The resulting model is a very large linear program which cannot be solved.

A well-known method in operations research to solve such kind of problems is to use a column generation method (see [3] for details), that is, a small instance with only a few patterns is solved and a rewarding column – a pattern – is added repeatedly to the problem. The new problem is then solved again. This process is repeated, until no pattern can be added. To find a rewarding pattern, another problem – named a knapsack problem – must be solved.

The problem can be formulated partially be algorithmic partially by declarative knowledge. It consists of two declaratively formulated problems (a linear program and an knapsack problem), which are both repeatedly solved. In a pseudocode one could formulate the algorithmic knowledge as follows:

```
SOLVE the small cutting-stock problem
SOLVE the knapsack problem
WHILE a rewarding pattern was found DO
    add pattern to the cutting-stock problem
    SOLVE the cutting-stock problem again
    SOLVE the knapsack problem again
ENDWHILE
```

The two models (the cutting-stock problem and the knapsack problem) can be formulated declaratively. In the proposed framework of modeling language, the complete problem can now be expressed as in the program below.

```
MODEL CuttingStock;
  MODEL Knapsack(i, w, p, K, x, obj);
    SET i;
    PARAMETER w{i}; p{i}; K;
    INTEGER VARIABLE x{i};
    CONSTRAINT R: SUM{i} w * x ≤ K;
    MAXIMIZE obj: SUM{i} p * x;
  END Knapsack.
  SET
    w 'rolls ordered'; p 'possible patterns';
  PARAMETER
    a{w, p} 'pattern table';
    d{w} 'demands';
    b{w} 'widths of ordered rolls';
    B 'initial width';
    INTEGER y{w} 'new added pattern';
    C 'contribution of a cut';
  VARIABLE
    X{p} 'number of rolls cut according to p';
  CONSTRAINT
    Dem{w}: SUM{p} a * X ≥ d;
  MINIMIZE obj: SUM{p} X;
BEGIN
  SOLVE;
  SOLVE Knapsack(w, b, Dem.dual, B, y, C);
  WHILE (C > 1) DO
    p := p + {'pattern_' + str(#p)};
    a{w, #p} := y[w];
    SOLVE;
    SOLVE Knapsack(w, b, Dem.dual, B, y, C);
  END;
END CuttingStock.
```

This formulation has several remarkable properties:

1) It is short and readable. The declarative part consists of the (small) linear cutting-stock problem, it also contains, as a submodel, a knapsack problem. The algorithmic part implements thecolumn generation method. Both parts are entirely separated.
2) It is a complete formulation, except from the data. No other code is needed; both models can be solved using a standard MIP solver (since the knapsack problem is small in general).
3) It has a modular structure. The knapsack problem is an independent component with its own name space; there is no interference with the surrounding model. It could even be declared outside the cutting-stock problem.

4) The cutting-stock problem is only one problem of a large class of relevant problems which are solved using a column generation or, alternatively, a row-cut generation.

Conclusion

It has been shown that certain problems are best formulated as algorithms, others in a declarative way, still others need both paradigms to be stated concisely. Computer science made available many algorithmic languages; they can be contrasted to the algebraic modeling languages which are purely declarative. A language, called modeling language, which combines both paradigms was defined in this paper and examples were given showing clear advantages of doing so. Its is more powerful than both paradigms separated.

However, the integration of algorithmic and declarative knowledge cannot be done in an arbitrary way. The language design must follow certain criteria well-known in computer science. The main criteria are: *reliability* and *transparency*. Reliability can be achieved by a unique notation to code models, that is, by a modeling language, and by various checking mechanisms (type checking, unit checking, data integrity checking and others). Transparency can be obtained by flexible decomposition techniques, like modular structure as well as access and protection mechanisms of these structure, well-known techniques in language design and software engineering.

Solving efficiently and relevant optimization problems using present desktop machine not only asks for fast machines and sophisticated solvers, but also for formulation techniques that allow the modeler to communicate the model easily and to build it in a readable and maintainable way.

See also

- ► Continuous Global Optimization: Models, Algorithms and Software
- ► Large Scale Unconstrained Optimization
- ► Optimization Software

References

1. Bisschop J (1998) AIMMS, the modeling system. Paragon Decision Techn, Haarlem, www.paragon.nl
2. Brooke A, Kendrick D, Meeraus A (1988) GAMS. A user's guide. Sci Press, Marrickville
3. Chvátal V (1973) Linear programming. Freeman, New York
4. Feigenbaum EA (1996) How the 'what' becomes the 'how'. Comm ACM 39(5):97–104
5. Floyd RW, Beigel R (1994) The language of machines, an introduction to computability and formal languages. Computer Sci Press, Rockville
6. Fourer R (1998) Extending a general-purpose algebraic modeling language to combinatorial optimization: A logic programming approach. In: Woodruff DL (ed) Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Sci and Oper Res. Kluwer, Dordrecht, pp 31–74
7. Fourer R, Gay DM, Kernighan BW (1993) AMPL, a modeling language for mathematical programming. Sci Press, Marrickville
8. GAY DM (1996) Automatically finding and exploiting partially separable structure in nonlinear programming problems. AT&T Bell Lab Murray Hill, New Jersey
9. Hofstadter DR (1988) Metamagicum, Fragen nach der Essenz von Geist und Struktur. Klett-Cotta, Stuttgart
10. Hürlimann T (1997) Computer-based mathematical modeling. Habilitations Script. Fac Economic and Social Sci, Inst Informatics, Univ Fribourg
11. Hürlimann T (1998) An efficient logic-to-IP translation procedure. Working Paper, Inst Informatics, Univ Fribourg, ftp://ftp-iiuf.unifr.ch/pub/lpl/doc/APMOD1.pdf
12. Hürlimann T (1998) Reference manual for the LPL modeling language. Working Paper, version 4.30. Inst Informatics, Univ. Fribourg, Fribourg, ftp://ftp-iiuf.unifr.ch/pub/lpl/doc/Manual.ps
13. Jaffar J, Maher MJ (1995) Constraint logic programming: A survey. Handbook Artificial Intelligence and Logic Programming. Oxford Univ Press, Oxford
14. Louden KC (1993) Programming languages – Principles and practice. PWS/Kent Publ, Boston
15. Maranas CD, Floudas CA, Pardalos PM (1993) New results in the packing of equal circles in a square. Dept Chemical Engin, Princeton Univ, Princeton
16. Murtagh BA, Saunders MA (1987) MINOS 5.0, user guide. Systems Optim Lab, Dept Oper Res, Stanford Univ, Stanford
17. ILOG SA (1997) ILOG solver 4.0 user's manual; ILOG solver 4.0 reference manual. ILOG, Mountain View
18. Schrage L (1998) Optimization modeling with LINGO. Lindo Systems, Chicago, www.lindo.com
19. Smolka G (1995) The Oz programming model. In: van Leeuwen J (ed) Computer Sci Today, 1000 of Computer Sci. Springer, Berlin, pp 324–343
20. Sosic R, Gu J (1990) A polynomial time algorithm for the n-queens problem. SIGART Bull 1(3):7–11
21. Sosic R, Gu J (1991) 3,000,000 queens in less than one minute. SIGART Bull 2(1):22–24

Molecular Distance Geometry Problem

Carlile Lavor[1], Leo Liberti[2], Nelson Maculan[3]
[1] State University of Campinas (IMECC-UNICAMP), Campinas, Brazil
[2] École Polytechnique, LIX, Palaiseau, France
[3] Federal University of Rio de Janeiro (COPPE-UFRJ), Rio de Janeiro, Brazil

MSC2000: 46N60

Article Outline

Introduction
ABBIE Algorithm
Global Continuation Algorithm
D.C. Optimization Algorithms
Geometric Build-up Algorithm
BP Algorithm
Conclusion
Acknowledgements
References

Introduction

This article presents a general overview of some of the most recent approaches for solving the molecular distance geometry problem, namely, the ABBIE algorithm, the Global Continuation Algorithm, d.c. optimization algorithms, the geometric build-up algorithm, and the BP algorithm.

The determination of the three-dimensional structure of a molecule, especially in the protein folding framework, is one of the most important problems in computational biology. That structure is very important because it is associated to the chemical and biological properties of the molecule [7,11,46]. Basically, this problem can be tackled in two ways: experimentally, via nuclear magnetic resonance (NMR) spectroscopy and X-ray crystallography [8], or theoretically, through potential energy minimization [19].

The Molecular Distance Geometry Problem (MDGP) arises in NMR analysis. This experimental technique provides a set of inter-atomic distances d_{ij} for certain pairs of atoms (i,j) of a given protein [23,24,33,56,57]. The MDGP can be formulated as follows:

Given a set S of atom pairs (i,j) on a set of m atoms and distances d_{ij} defined over S, find positions $x_1, \ldots, x_m \in \mathbb{R}^3$ of the atoms in the molecule such that

$$||x_i - x_j|| = d_{ij}\,, \quad \forall (i,j) \in S. \tag{1}$$

When the distances between all pairs of atoms of a molecule are given, a unique three-dimensional structure can be determined by a linear time algorithm [16]. However, because of errors in the given distances, a solution may not exist or may not be unique. In addition to this, because of the large scale of problems that arise in practice, the MDGP becomes very hard to solve in general. Saxe [51] showed that the MDGP is NP-complete even in one spatial dimension.

The exact MDGP can be naturally formulated as a nonlinear global minimization problem, where the objective function is given by

$$f(x_1, \ldots, x_m) = \sum_{(i,j)\in S} (||x_i - x_j||^2 - d_{ij}^2)^2\,. \tag{2}$$

This function is everywhere infinitely differentiable and has an exponential number of local minimizers. Assuming that all the distances are correctly given, $x \in \mathbb{R}^{3m}$ solves the problem if and only if $f(x) = 0$.

Formulations (1) and (2) correspond to the exact MDGP. Since experimental errors may prevent solution existence (e. g. when the triangle inequality

$$d_{ij} \le d_{ik} + d_{kj}$$

is violated for atoms i,j,k), we sometimes consider an ϵ-optimum solution of (1), i. e. a solution $x_1, \ldots, x_m$ satisfying

$$|||x_i - x_j|| - d_{ij}| \le \epsilon\,, \quad \forall (i,j) \in S\,. \tag{3}$$

Moré and Wu [41] showed that even obtaining such an ϵ-optimum solution is NP-hard for ϵ small enough.

In practice, it is often just possible to obtain lower and upper bounds on the distances [4]. Hence a more practical definition of the MDGP is to find positions $x_1, \ldots, x_m \in \mathbb{R}^3$ such that

$$l_{ij} \le ||x_i - x_j|| \le u_{ij}\,, \quad \forall (i,j) \in S\,, \tag{4}$$

where l_{ij} and u_{ij} are lower and upper bounds on the distance constraints, respectively.

The MDGP is a particular case of a more general problem, called the distance geometry problem [6,13,14,15], which is intimately related to the Euclidean distance matrix completion problem [1,28,38].

Several methods have been developed to solve the MDGP, including the EMBED algorithm by Crippen and Havel [12,25], the alternating projection algorithm by Glunt et al. [20], spectrial gradient methods by Glunt et al. [21,22], the multi-scaling algorithm by Trosset et al. [29,52], a stochastic/perturbation algorithm by Zou, Byrd, and Schnabel [58], variable neighborhood search-based algorithms by Liberti, Lavor, and Maculan [35,39], the ABBIE algorithm by Hendrickson [26,27], the Global Continuation Algorithm by Moré and Wu [41,42,43,44,45], the d.c. optimization algorithms by An and Tao [2,3], the geometric build-up algorithm by Dong, Wu, and Wu [16,17,54], and the BP algorithm by Lavor, Liberti, and Maculan [37]. Two completely different approaches for solving the MDGP are given in [34] (based on quantum computation) and [53] (based on algebraic geometry).

The wireless network sensor positioning problem is closely related to the MDGP, the main difference being the presence of fixed anchor points with known positions: results derived for this problem can often be applied to the MDGP. Amongst the most notable, [18] shows that the MDGP associated to a trilateration graph (a graph with an order on the vertices such that each vertex is adjacent to the preceding 4 vertices) can be solved in polynomial time; [40] provides a detailed study of Semi Definite Programming (SDP) relaxations applied to distance geometry problems.

ABBIE Algorithm

In [26,27], Hendrickson describes an approach to the exact MDGP that replaces a large optimization problem, given by (2), by a sequence of smaller ones. He exploits some combinatorial structure inherent in the MDGP, which allows him to develop a divide-and-conquer algorithm based on a graph-theoretic viewpoint.

If the atoms and the distances are considered as nodes and edges of a graph, respectively, the MDGP can be described by a distance graph and the solution to the problem is an embedding of the distance graph in an Euclidean space. When some of the atoms can be moved without violating any distance constraints, there may be many embeddings. The graph is then called flexible or otherwise rigid.

If the graph is rigid or does not have partial reflections, for example, then the graph has a unique embedding. These necessary conditions can be used to find subgraphs that have unique embeddings. The problem can then be solved by decomposing the graph into such subgraphs, in which the minimization problems associated to the function (2) are solved. The solutions found for the subgraphs can then be combined into a solution for the whole graph.

This approach to the MDGP has been implemented in a code named ABBIE and tested on simulated data provided by the bovine pancreatic ribonuclease A, a typical small protein consisting of 124 amino acids, whose three-dimensional structure is known [47]. The data set consists of all distances between pairs of atoms in the same amino acid, along with 1167 additional distances corresponding to pairs of hydrogen atoms that were within 3.5 Å of each other. It was used fragments of the protein consisting of the first 20, 40, 60, 80 and 100 amino acids as well as the full protein, with two sets of distance constraints for each size corresponding to the largest unique subgraphs and the reduced graphs. These problems have from 63 up to 777 atoms.

Global Continuation Algorithm

In [43], Moré and Wu formulated the exact MDGP in terms of finding the global minimum of a similar function to (2),

$$f(x_1, \ldots, x_m) = \sum_{(i,j)\in S} w_{ij}(||x_i - x_j||^2 - d_{ij}^2)^2 , \quad (5)$$

where w_{ij} are positive weights (in numerical results $w_{ij} = 1$ was used).

Following the ideas described in [55], Moré and Wu proposed an algorithm, called Global Continuation Algorithm, based on a continuation approach for global optimization. The idea is gradually transform the function (5) into a smoother function with fewer local minimizers, where an optimization algorithm is then applied to the transformed function, tracing their minimizers back to the original function. For other works based on continuation approach, see [9,10,30,31,32,49].

The transformed function $\langle f\rangle_\lambda$, called the Gaussian transform, of a function $f\colon \mathbb{R}^n \to \mathbb{R}$ is defined by

$$\langle f\rangle_\lambda(x) = \frac{1}{\pi^{n/2}\lambda^n}\int_{\mathbb{R}^n} f(y)\exp\left(-\frac{||y-x||^2}{\lambda^2}\right)\mathrm{d}y\,, \tag{6}$$

where the parameter λ controls the degree of smoothing. The value $\langle f\rangle_\lambda(x)$ is a weighted average of $f(x)$ in a neighborhood of x, where the size of the neighborhood decreases as λ decreases: as $\lambda \to 0$, the average is carried out on the singleton set $\{x\}$, thus recovering the original function in the limit. Smoother functions are obtained as λ increases.

This approach to the MDGP has been implemented and tested on two artificial models of problems, where the molecule has $m = s^3$ atoms located in the three-dimensional lattice

$$\{(i_1, i_2, i_3)\colon 0 \le i_1 < s, 0 \le i_2 < s, 0 \le i_3 < s\}$$

for an integer $s \ge 1$. In numerical results, it was considered $m = 27, 64, 125, 216$.

In the first model, the ordering for the atoms is specified by letting i be the atom at the position (i_1,i_2,i_3),

$$i = 1 + i_1 + si_2 + s^2 i_3,$$

and the set of atom pairs whose distances are known, S, is given by

$$S = \{(i,j) : |i-j| \le r\}\,, \tag{7}$$

where $r = s^2$. In the second model, the set S is specified by

$$S = \{(i,j) : ||x_i - x_j|| \le \sqrt{r}\}\,, \tag{8}$$

where $x_i = (i_1, i_2, i_3)$ and $r = s^2$. For both models, s is considered in the interval $3 \le s \le 6$.

In (7), S includes all nearby atoms, while in (8), S includes some of nearby atoms and some relatively distant atoms.

It was shown that the Global Continuation Algorithm usually finds a solution from any given starting point, whereas the local minimization algorithm used in the multistart methods is unreliable as a method for determining global solutions. It was also showed that the continuation approach determines a global solution with less computational effort that is required by the multistart approach.

D.C. Optimization Algorithms

In [2,3], An and Tao proposed an approach for solving the exact MDGP, based on the d.c. (difference of convex functions) optimization algorithms. They worked in $\mathcal{M}_{m,3}(\mathbb{R})$, the space of real matrices of order $m \times 3$, where for $X \in \mathcal{M}_{m,3}(\mathbb{R})$, X_i (resp., X^i) is its ith row (resp., ith column). By identifying a set of positions of atoms $x_1, \ldots, x_m$ with the matrix X, $X_i^T = x_i$ for $i = 1, \ldots, m$, they expressed the MDGP by

$$0 = \min\left\{\sigma(X) := \frac{1}{2}\sum_{(i,j)\in S, i<j} w_{ij}\theta_{ij}(X)\colon X \in \mathcal{M}_{m,3}(\mathbb{R})\right\}, \tag{9}$$

where $w_{ij} > 0$ for $i \ne j$ and $w_{ii} = 0$ for all i. The pairwise potential $\theta_{ij}\colon \mathcal{M}_{m,3}(\mathbb{R}) \to \mathbb{R}$ is defined for problem (1) by either

$$\theta_{ij}(X) = \left(d_{ij}^2 - ||X_i^T - X_j^T||^2\right)^2 \tag{10}$$

or

$$\theta_{ij}(X) = \left(d_{ij} - ||X_i^T - X_j^T||\right)^2, \tag{11}$$

and for problem (4) by

$$\theta_{ij}(X) = \min^2\left\{\frac{||X_i^T - X_j^T||^2 - l_{ij}^2}{l_{ij}^2}, 0\right\} + \max^2\left\{\frac{||X_i^T - X_j^T||^2 - u_{ij}^2}{u_{ij}^2}, 0\right\}. \tag{12}$$

Similarly to (2), X is a solution if and only if it is a global minimizer of problem (9) and $\sigma(X) = 0$.

While the problem (9) with θ_{ij} given by (9) or (12) is a nondifferentiable optimization problem, it is a d.c. optimization problem.

An and Tao demonstrated that the d.c. algorithms can be adapted for developing efficient algorithms for solving large-scale exact MDGPs. They proposed various versions of d.c. algorithms that are based on different formulations for the problem. Due its local character, the global optimality cannot be guaranteed for a general d.c. problem. However, the fact that the global optimality can be obtained with a suitable starting point

motivated them to investigate a technique for computing good starting points for the d.c. algorithms in the solution of (9), with θ_{ij} defined by (11).

The algorithms have been tested on three sets of data: the artificial data from Moré and Wu [43] (with up to 4096 atoms), 16 proteins in the PDB [5] (from 146 up to 4189 atoms), and the data from Hendrickson [27] (from 63 up to 777 atoms). Using these data, they showed that the d.c. algorithms can efficiently solve large-scale exact MDGPs.

Geometric Build-up Algorithm

In [17], Dong and Wu proposed the solution of the exact MDGP by an algorithm, called the geometric build-up algorithm, based on a geometric relationship between coordinates and distances associated to the atoms of a molecule. It is assumed that it is possible to determine the coordinates of at least four atoms, which are marked as fixed; the remaining ones are non-fixed. The coordinates of a non-fixed atom a can be calculated by using the coordinates of four non-coplanar fixed atoms such that the distances between any of these four atoms and the atom a are known. If such four atoms are found, the atom a changes its status to fixed. More specifically, let b_1, b_2, b_3, b_4 be the four fixed atoms whose Cartesian coordinates are already known. Now suppose that the Euclidean distances among the atom a and the atoms b_1, b_2, b_3, b_4, namely $d_{a,bi}$, for $i = 1, 2, 3, 4$, are known. That is,

$$
\begin{aligned}
||a - b_1|| &= d_{a,b_1},\\
||a - b_2|| &= d_{a,b_2},\\
||a - b_3|| &= d_{a,b_3},\\
||a - b_4|| &= d_{a,b_4}.
\end{aligned}
$$

Squaring both sides of these equations, we have:

$$
\begin{aligned}
||a||^2 - 2a^T b_1 + ||b_1||^2 &= d^2_{a,b_1},\\
||a||^2 - 2a^T b_2 + ||b_2||^2 &= d^2_{a,b_2},\\
||a||^2 - 2a^T b_3 + ||b_3||^2 &= d^2_{a,b_3},\\
||a||^2 - 2a^T b_4 + ||b_4||^2 &= d^2_{a,b_4}.
\end{aligned}
$$

By subtracting one of these equations from the others, it is obtained a linear system that can be used to determine the coordinates of the atom a. For example, subtracting the first equation from the others, we obtain

$$Ax = b\,, \tag{13}$$

where

$$A = -2\begin{bmatrix} (b_1 - b_2)^T \\ (b_1 - b_3)^T \\ (b_1 - b_3)^T \end{bmatrix}, \quad x = a\,,$$

and

$$b = \begin{bmatrix} \left(d^2_{a,b_1} - d^2_{a,b_2}\right) - \left(||b_1||^2 - ||b_2||^2\right) \\ \left(d^2_{a,b_1} - d^2_{a,b_3}\right) - \left(||b_1||^2 - ||b_3||^2\right) \\ \left(d^2_{a,b_1} - d^2_{a,b_4}\right) - \left(||b_1||^2 - ||b_4||^2\right) \end{bmatrix}.$$

Since b_1, b_2, b_3, b_4 are non-coplanar atoms, the system (13) has a unique solution. If the exact distances between all pairs of atoms are given, this approach can determine the coordinates of all atoms of the molecule in linear time [16].

Dong and Wu implemented such an algorithm, but they verified that it is very sensitive to the numerical errors introduced in calculating the coordinates of the atoms. In [54], Wu and Wu proposed the updated geometric build-up algorithm showing that, in this algorithm, the accumulation of the errors in calculating the coordinates of the atoms can be controlled and prevented. They have been tested the algorithm with a set of problems generated using the known structures of 10 proteins downloaded from the PDB data bank [5], with problems from 404 up to 4201 atoms.

BP Algorithm

In [37], Lavor, Liberti, and Maculan propose an algorithm, called branch-and-prune (BP), based on a discrete formulation of the exact MDGP. They observe that the particular structures of proteins makes it possible to formulate the MDGP applied to protein backbones as a discrete search problem. They formalize this by introducing the discretizable molecular distance geometry problem (DMDGP), which consists of a certain subset of MDGP instances (to which most protein backbones belong) for which a discrete formulation can be supplied. This approach requires that the bond lengths and angles, as well as the distances between atoms separated by three consecutive bond lengths are known.

In order to describe a backbone of a protein with m atoms, in addition to the bond lengths $d_{i-1,i}$, for $i = 2, \ldots, m$, and the bond angles $\theta_{i-2,i}$, for $i = 3, \ldots, m$,

it is necessary to consider the torsion angles $\omega_{i-3,i}$, for $i = 4, \ldots, m$, which are the angles between the normals through the planes defined by the atoms $i-3$, $i-2$, $i-1$ and $i-2$, $i-1$, i.

It is known that [48], given all the bond lengths $d_{1,2}, \ldots, d_{m-1,m}$, bond angles $\theta_{13}, \ldots, \theta_{m-2,m}$, and torsion angles $\omega_{1,4}, \ldots, \omega_{m-3,m}$ of a molecule with m atoms, the Cartesian coordinates $(x_{i_1}, x_{i_2}, x_{i_3})$ for each atom i in the molecule can be obtained using the following formulae:

$$\begin{bmatrix} x_{i_1} \\ x_{i_2} \\ x_{i_3} \\ 1 \end{bmatrix} = B_1 B_2 \ldots B_i \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \forall i = 1, \ldots, m,$$

where

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$B_2 = \begin{bmatrix} -1 & 0 & 0 & -d_{1,2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$B_3 = \begin{bmatrix} -\cos\theta_{1,3} & -\sin\theta_{1,3} & 0 & -d_{2,3}\cos\theta_{1,3} \\ \sin\theta_{1,3} & -\cos\theta_{1,3} & 0 & d_{2,3}\sin\theta_{1,3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

and

$$B_i = \begin{bmatrix} -\cos\theta_{i-2,i} & -\sin\theta_{i-2,i} & 0 & -d_{i-1,i}\cos\theta_{i-2,i} \\ \sin\theta_{i-2,i}\cos\omega_{i-3,i} & -\cos\theta_{i-2,i}\cos\omega_{i-3,i} & -\sin\omega_{i-3,i} & d_{i-1,i}\sin\theta_{i-2,i}\cos\omega_{i-3,i} \\ \sin\theta_{i-2,i}\sin\omega_{i-3,i} & -\cos\theta_{i-2,i}\sin\omega_{i-3,i} & \cos\omega_{i-3,i} & d_{i-1,i}\sin\theta_{i-2,i}\sin\omega_{i-3,i} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

for $i = 4, \ldots, m$.

Since all the bond lengths and bond angles are assumed to be given in the instance, the Cartesian coordinates of all atoms of a molecule can be completely determined by using the values of $\cos\omega_{i-3,i}$ and $\sin\omega_{i-3,i}$, for $i = 4, \ldots, m$.

For instances of the DMDGP class, for all $i = 4, \ldots, m$, the value of $\cos\omega_{i-3,i}$ can be computed by the formula

$$\begin{aligned} \cos\omega_{i-3,i} &= a/b \\ \text{where} \quad a &= d_{i-3,i-2}^2 + d_{i-2,i}^2 - 2d_{i-3,i-2}d_{i-2,i} \\ &\quad \cdot \cos\theta_{i-2,i}\cos\theta_{i-1,i+1} - d_{i-3,i}^2 \\ \text{and} \quad b &= 2d_{i-3,i-2}d_{i-2,i}\sin\theta_{i-2,i}\sin\theta_{i-1,i+1}, \end{aligned} \tag{14}$$

which is just a rearrangement of the cosine law for torsion angles [50] (p. 278), and all the values in the expression (14) are given in the instance. This allows to express the position of the i-th atom in terms of the preceding three, giving 2^{m-3} possible conformations, which characterizes the discretization of the problem.

The idea of the BP algorithm is that at each step the ith atom can be placed in two possible positions. However, either of both of these positions may be infeasible with respect to some constraints. The search is branched on all atomic positions which are feasible with respect to all constraints; by contrast, if a position is not feasible the search scope is pruned.

The algorithm has been tested on the artificial data from Moré and Wu [43] (with up to 216 atoms) and on the artificial data from Lavor [36] (a selection from 10 up to 100 atoms).

Conclusion

This paper surveys some of the methods to solve the Molecular Distance Geometry Problem, with particular reference to five existing algorithms: ABBIE algorithm, global continuation algorithm, d.c. optimization algorithms, the geometric build-up algorithm and the BP algorithm.

Acknowledgements

The authors would like to thank CNPq, FAPESP and FAPERJ for their financial support.

References

1. Alfakih AY, Khandani A, Wolkowicz H (1999) Solving Euclidean distance matrix completion problems via semidefinite programming. Comput Optim Appl 12:13–30
2. An LTH (2003) Solving large-scale molecular distance geometry problems by a smoothing technique via the Gaussian transform and d.c. programming. J Global Optim 27:375–397

3. An LTH, Tao PD (2003) Large-scale molecular optimization from distance matrices by a d.c. optimization approach. SIAM J Optim 14:77–114
4. Berger B, Kleinberg J, Leighton T (1999) Reconstructing a three-dimensional model with arbitrary errors. J ACM 46:212–235
5. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) The protein data bank. Nucl Acids Res 28:235–242
6. Blumenthal LM (1953) Theory and Applications of Distance Geometry. Oxford University Press, London
7. Brooks III CL, Karplus M, Pettitt BM (1988) Proteins: a theoretical perspective of dynamics, structure, and thermodynamics. Wiley, New York
8. Brünger AT, Nilges M (1993) Computational challenges for macromolecular structure determination by X-ray crystallography and solution NMR-spectroscopy. Q Rev Biophys 26:49–125
9. Coleman TF, Shalloway D, Wu Z (1993) Isotropic effective energy simulated annealing searches for low energy molecular cluster states. Comput Optim Appl 2:145–170
10. Coleman TF, Shalloway D, Wu Z (1994) A parallel build-up algorithm for global energy minimizations of molecular clusters using effective energy simulated annealing. J Global Optim 4:171–185
11. Creighton TE (1993) Proteins: structures and molecular properties. Freeman and Company, New York
12. Crippen GM, Havel TF (1988) Distance geometry and molecular conformation. Wiley, New York
13. Dattorro J (2005) Convex optimization and euclidean distance geometry. Meboo Publishing USA, Palo Alto
14. De Leeuw J (1977) Applications of convex analysis to multidimensional scaling. In: Barra JR, Brodeau F, Romier G, van Cutsem B (eds) Recent developments in statistics. North-Holland, Amsterdam, pp 133–145
15. De Leeuw J (1988) Convergence of the majorization method for multidimensional scaling. J Classif 5:163–180
16. Dong Q, Wu Z (2002) A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances. J Global Optim 22:365–375
17. Dong Q, Wu Z (2003) A geometric build-up algorithm for solving the molecular distance geometry problem with sparse distance data. J Global Optim 26:321–333
18. Eren T, Goldenberg DK, Whiteley W, Yang YR, Morse AS, Anderson BDO, Belhumeur PN (2004) Rigidity, computation, and randomization in network localization. In: Proc IEEE Infocom 2673–2684, Hong Kong
19. Floudas CA, Pardalos PM (eds)(2000) Optimization in computational chemistry and molecular biology. Nonconvex optimization and its applications, vol 40. Kluwer, The Netherlands
20. Glunt W, Hayden TL, Hong S, Wells J (1990) An alternating projection algorithm for computing the nearest euclidean distance matrix. SIAM J Matrix Anal Appl 11:589–600
21. Glunt W, Hayden TL, Raydan M (1993) Molecular conformations from distance matrices. J Comput Chem 14:114–120
22. Glunt W, Hayden TL, Raydan M (1994) Preconditioners for distance matrix algorithms. J Comput Chem 15:227–232
23. Gunther H (1995) NMR Spectroscopy: basic principles, concepts, and applications in chemistry. Wiley, New York
24. Havel TF (1991) An evaluation of computational strategies for use in the determination of protein structure from distance geometry constraints obtained by nuclear magnetic resonance. Prog Biophys Mol Biol 56:43–78
25. Havel TF (1995) Distance geometry. In: Grant DM, Harris RK (eds) Encyclopedia of nuclear magnetic resonance. Wiley, New York, pp 1701–1710
26. Hendrickson BA (1991) The molecule problem: determining conformation from pairwise distances. Ph.D. thesis. Cornell University, Ithaca
27. Hendrickson BA (1995) The molecule problem: exploiting structure in global optimization. SIAM J Optim 5:835–857
28. Huang HX, Liang ZA (2003) Pardalos PM Some properties for the euclidean distance matrix and positive semidefinite matrix completion problems. J Global Optim 25:3–21
29. Kearsley AJ, Tapia RA, Trosset MW (1998) The solution of the metric STRESS and SSTRESS problems in multidimensional scaling by Newton's method. Comput Stat 13:369–396
30. Kostrowicki J, Piela L (1991) Diffusion equation method of global minimization: performance for standard functions. J Optim Theor Appl 69:269–284
31. Kostrowicki J, Piela L, Cherayil BJ, Scheraga HA (1991) Performance of the diffusion equation method in searches for optimum structures of clusters of Lennard-Jones atoms. J Phys Chem 95:4113–4119
32. Kostrowicki J, Scheraga HA (1992) Application of the diffusion equation method for global optimization to oligopeptides. J Phys Chem 96:7442–7449
33. Kuntz ID, Thomason JF, Oshiro CM (1993) Distance geometry. In: Oppenheimer NJ, James TL (eds) Methods in Enzymology, vol 177. Academic Press, New York, pp 159–204
34. Lavor C, Liberti L, Maculan N (2005) Grover's algorithm applied to the molecular distance geometry problem. In: Proc. of VII Brazilian Congress of Neural Networks, Natal, Brazil
35. Lavor C, Liberti L, Maculan N (2006) Computational experience with the molecular distance geometry problem. In: Pintér J (ed) Global optimization: scientific and engineering case studies. Springer, New York, pp 213–225
36. Lavor C (2006) On generating instances for the molecular distance geometry problem. In: Liberti L, Maculan N (eds) Global optimization: from theory to implementation. Springer, Berlin, pp 405–414
37. Lavor C, Liberti L, Maculan N (2006) The discretizable molecular distance geometry problem. arXiv:q-bio/0608012

38. Laurent M (1997) Cuts, matrix completions and a graph rigidity. Math Program 79:255–283
39. Liberti L, Lavor C, Maculan N (2005) Double VNS for the molecular distance geometry problem. In: Proc. of MECVNS Conference, Puerto de la Cruz, Spain
40. Man-Cho So A, Ye Y (2007) Theory of semidefinite programming for sensor network localization. Math Program 109:367–384
41. Moré JJ, Wu Z (1996) ϵ-Optimal solutions to distance geometry problems via global continuation. In: Pardalos PM, Shalloway D, Xue G (eds) Global minimization of non-convex energy functions: molecular conformation and protein folding. American Mathematical Society, Providence, IR, pp 151–168
42. Moré JJ, Wu Z (1996) Smoothing techniques for macromolecular global optimization. In: Di Pillo G, Gianessi F (eds) Nonlinear Optimization and Applications. Plenum Press, New York, pp 297–312
43. Moré JJ, Wu Z (1997) Global continuation for distance geometry problems. SIAM J Optim 7:814–836
44. Moré JJ, Wu Z (1997) Issues in large scale global molecular optimization. In: Biegler L, Coleman T, Conn A, Santosa F (eds) Large scale optimization with applications. Springer, Berlin, pp 99–122
45. Moré JJ, Wu Z (1999) Distance geometry optimization for protein structures. J Global Optim 15:219–234
46. Neumaier A (1997) Molecular modeling of proteins and mathematical prediction of protein structure. SIAM Rev 39:407–460
47. Palmer KA, Scheraga HA (1992) Standard-geometry chains fitted to X-ray derived structures: validation of the rigid-geometry approximation. II. Systematic searches for short loops in proteins: applications to bovine pancreatic ribonuclease A and human lysozyme. J Comput Chem 13:329–350
48. Phillips AT, Rosen JB, Walke VH (1996) Molecular structure determination by convex underestimation of local energy minima. In: Pardalos PM, Shalloway D, Xue G (eds) Global minimization of non-convex energy functions: molecular conformation and protein folding. American Mathematical Society, Providence, IR, pp 181–198
49. Piela L, Kostrowicki J, Scheraga HA (1989) The multiple-minima problem in the conformational analysis of molecules: deformation of the protein energy hypersurface by the diffusion equation method. J Phys Chem 93:3339–3346
50. Pogorelov A (1987) Geometry. Mir Publishers, Moscow
51. Saxe JB (1979) Embeddability of weighted graphs in k-space is strongly NP-hard. In: Proc. of 17th Allerton Conference in Communications, Control, and Computing, 480–489, Allerton, USA
52. Trosset M (1998) Applications of multidimensional scaling to molecular conformation. Comput Sci Stat 29:148–152
53. Wang L, Mettu RR, Donald BR (2005) An algebraic geometry approach to protein structure determination from NMR data. In: Proc. of the 2005 IEEE Computational Systems Bioinformatics Conference, Stanford, USA
54. Wu D, Wu Z (2007) An updated geometric build-up algorithm for solving the molecular distance geometry problem with sparse distance data. J Global Optim 37:661–673
55. Wu Z (1996) The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation. SIAM J Optim 6:748–768
56. Wütrich K (1989) The development of nuclear magnetic resonance spectroscopy as a technique for protein structure determination. Acc Chem Res 22:36–44
57. Wütrich K (1989) Protein structure determination in solution by nuclear magnetic resonance spectroscopy. Science 243:45–50
58. Zou Z, Byrd RH, Schnabel RB (1997) A stochastic/perturbation global optimization algorithm for distance geometry problems. J Global Optim 11:91–105

Molecular Structure Determination: Convex Global Underestimation

ANDREW T. PHILLIPS
Computer Sci. Department, University Wisconsin–Eau Claire, Eau Claire, USA

MSC2000: 65K05, 90C26

Article Outline

Keywords
Molecular Model
The Convex Global Underestimator
The CGU Algorithm
See also
References

Keywords

Protein folding; Molecular structure determination; Convex global underestimation

An important class of difficult global minimization problems arise as an essential feature of molecular structure calculations. The determination of a stable molecular structure can often be formulated in terms of calculating the global (or approximate global) minimum of a potential energy function (see [6]). Computing the global minimum of this function is very difficult because it typically has a very large number of local

minima which may grow exponentially with molecule size.

One such application is the well known protein folding problem. It is widely accepted that the folded state of a protein is completely dependent on the one-dimensional linear sequence (i. e., 'primary' sequence) of amino acids from which the protein is constructed: external factors, such as enzymes, present at the time of folding have no effect on the final, or native, state of the protein. This led to the formulation of the *protein folding problem*: given a known primary sequence of amino acids, what would be its native, or folded, state in three-dimensional space.

Several successful predictions of folded protein structures have been made and announced before the experimental structures were known (see [3,9]). While most of these have been made with a blend of a human expert's abilities and computer assistance, fully automated methods have shown promise for producing previously unattainable accuracy [2].

These machine based prediction strategies attempt to lessen the reliance on experts by developing a completely computational method. Such approaches are generally based on two assumptions. First, that there exists a potential energy function for the protein; and second that the folded state corresponds to the structure with the lowest potential energy (minimum of the potential energy function) and is thus in a state of thermodynamic equilibrium. This view is supported by in vitro observations that proteins can successfully refold from a variety of denatured states. Evolutionary theory also supports a folded state at a global energy minimum. Protein sequences have evolved under pressure to perform certain functions, which for most known occurrences requires a stable, unique, and compact structure. Unless specifically required for a certain function, there was no biochemical need for proteins to hide their global minimum behind a large kinetic energy barrier. While kinetic blocks may occur, they should be limited to special proteins developed for certain functions (see [1]).

Molecular Model

Unfortunately, finding the 'true' energy function of a molecular structure, if one even exists, is virtually impossible. For example, with proteins ranging in size up to 1, 053 amino acids (a collagen found in tendons), exhaustive conformational searches will never be tractable. Practical search strategies for the protein folding problem currently require a simplified, yet sufficiently realistic, molecular model with an associated potential energy function representing the dominant forces involved in protein folding [4]. In a one such simplified model, each residue in the primary sequence of a protein is characterized by its backbone components $\mathrm{NH} - C_\alpha H - \mathbf{C}'\mathrm{O}$ and one of 20 possible amino acid sidechains attached to the central C_α atom. The three-dimensional structure of the chain is determined by internal molecular coordinates consisting of bond lengths l, bond angles θ, sidechain torsion angles χ, and the backbone dihedral angles ϕ, ψ, and ω. Fortunately, these $10r - 6$ parameters (for an r-residue structure) do not all vary independently. Some of these ($7r - 4$ of them) are regarded as fixed since they are found to vary within only a very small neighborhood of an experimentally determined value. Among these are the $3r - 1$ backbone bond lengths l, the $3r - 2$ backbone bond angles θ, and the $r - 1$ peptide bond dihedral angles ω (fixed in the trans conformation). This leaves only the r sidechain torsion angles χ, and the $r - 1$ backbone dihedral angle pairs (ϕ, ψ). In the reduced representation model presented here, the sidechain angles χ are also fixed since sidechains are treated as united atoms (see below) with their respective torsion angles χ fixed at an 'average' value taken from the Brookhaven Protein Databank. Remaining are the $r - 1$ backbone dihedral angles pairs. These also are not completely independent; they are severely constrained by known chemical data (the Ramachandran plot) for each of the 20 amino acid residues. Furthermore, since the atoms from one C_α to the next C_α along the backbone can be grouped into rigid planar peptide units, there are no extra parameters required to express the three-dimensional position of the attached O and H peptide atoms. Hence, these bond lengths and bond angles are also known and fixed.

Another key element of this simplified polypeptide model is that each sidechain is classified as either hydrophobic or polar, and is represented by only a single 'virtual' center of mass atom. Since each sidechain is represented by only the single center of mass 'virtual atom' C_s, no extra parameters are needed to define the position of each sidechain with respect to the backbone

mainchain. The twenty amino acids are thus classified into two groups, hydrophobic and polar, according to the scale given by S. Miyazawa and R.L. Jernigan [7].

Corresponding to this simplified polypeptide model is a simple energy function. This function includes four components: a contact energy term favoring pairwise hydrophobic residues, a second contact term favoring hydrogen bond formation between donor NH and acceptor $C' = O$ pairs, a steric repulsive term which rejects any conformation that would permit unreasonably small interatomic distances, and a main chain torsional term that allows only certain preset values for the backbone dihedral angle pairs (ϕ, ψ). Since the residues in this model come in only two forms, hydrophobic and polar, where the hydrophobic monomers exhibit a strong pairwise attraction, the lowest free energy state involves those conformations with the greatest number of hydrophobic 'contacts' [4] and intrastrand hydrogen bonds. Simplified potential functions have been successful in [10,11], and [12]. Here we use a simple modification of the energy function from [11].

The Convex Global Underestimator

One practical means for finding the global minimum of the polypeptide's potential energy function is to use a convex global underestimator to localize the search in the region of the global minimum. The idea is to fit all known local minima with a convex function which underestimates all of them, but which differs from them by the minimum possible amount in the discrete L1 norm. The minimum of this underestimator is used to predict the global minimum for the function, allowing a more localized conformer search to be performed based on the predicted minimum.

More precisely, given an r-residue structure with $n = 2r - 2$ backbone dihedral angles, denote a conformation of this simplified model by $\phi \in \mathbf{R}^n$, and the corresponding simplified potential energy function value by $F(\phi)$. Then, assuming that $k \geq 2n + 1$ local minimum conformations $\phi^{(j)}$, for $j = 1, \ldots, k$, have been computed, a convex quadratic underestimating function $U(\phi)$ is fitted to these local minima so that it underestimates all the local minima, and normally interpolates $F(\phi^{(j)})$ at $2n + 1$ points. This is accomplished by determining the coefficients in the function $U(\phi)$ so that

$$\delta_j = F(\phi^{(j)}) - U(\phi^{(j)}) \geq 0 \tag{1}$$

for $j = 1, \ldots, k$, and where $\sum_{j=1}^{n} \delta_j$ is minimized. That is, the difference between $F(\phi)$ and $U(\phi)$ is minimized in the discrete L_1 norm over the set of k local minima $\phi^{(j)}$, $j = 1, \ldots, k$. Of course, this 'underestimator' only underestimates known local minima. The specific underestimating function $U(\phi)$ used in this convex global underestimator (CGU) method is given by

$$U(\phi) = c_0 + \sum_{i=1}^{n} \left(c_i \phi_i + \frac{1}{2} d_i \phi_i^2 \right). \tag{2}$$

Note that c_i and d_i appear linearly in the constraints of (1) for each local minimum $\phi^{(j)}$. Convexity of this quadratic function is guaranteed by requiring that $d_i \geq 0$ for $i = 1, \ldots, n$. Other linear combinations of convex functions could also be used, but this quadratic function is the simplest.

Additionally, in order to guarantee that $U(\phi)$ attains its global minimum $U_{\min}$ in the hyperrectangle $H\phi = \{\phi_i : \ 0 \leq \underline{\phi}_i \leq \phi_i \leq \overline{\phi}_i \leq 2\pi\}$, an additional set of constraints are imposed on the coefficients of $U(\phi)$:

$$\begin{cases} c_i + \underline{\phi}_i d_i \leq 0, \\ c_i + \overline{\phi}_i d_i \geq 0, \end{cases} \qquad i = 1, \ldots, n. \tag{3}$$

Note that the satisfaction of (3) implies that $c_i \leq 0$ and $d_i \geq 0$ for $i = 1, \ldots, n$.

The unknown coefficients c_i, $i = 0, \ldots, n$, and d_i, $i = 1, \ldots, n$, can be determined by a linear program which may be considered to be in the dual form. For reasons of efficiency, the equivalent primal of this problem is actually solved, as described below. The solution to this primal linear program provides an optimal dual vector, which immediately gives the underestimating function coefficients c_i and d_i. Since the convex quadratic function $U(\phi)$ gives a global approximation to the local minima of $F(\phi)$, then its easily computed global minimum function value $U_{\min}$ is a good candidate for an approximation to the global minimum of the correct energy function $F(\phi)$.

An efficient linear programming formulation and solution satisfying (1)–(3) will now be summarized. Let $f^{(j)} = F(\phi^{(j)})$, for $j = 1, \ldots, k$, and let $f \in \mathbf{R}^k$ be the vector with elements $f^{(j)}$. Also let $\omega^{(j)} \in \mathbf{R}^n$ be the vector with elements $\frac{1}{2}(\phi_i^{(j)})^2$, $i = 1, \ldots, n$, and let $e_k \in \mathbf{R}^k$ be the vector of ones. Now define the following two matrices

$\Phi \in \mathbf{R}^{(n+1)\times k}$ and $\Omega \in \mathbf{R}^{n\times k}$:

$$\begin{cases} \Phi = \begin{pmatrix} e_k^\top \\ \phi^{(1)} \cdots \phi^{(k)} \end{pmatrix}, \\ \Omega = \begin{pmatrix} \omega^{(1)} \cdots \omega^{(k)} \end{pmatrix}. \end{cases} \tag{4}$$

Finally, let $c \in \mathbf{R}^{n+1}$, $d \in \mathbf{R}^n$, and $\delta \in \mathbf{R}^k$ be the vectors with elements c_i, d_i, and δ_i, respectively. Then (1)–(3) can be restated as the linear program (with free variables c, d, and δ):

- minimize $e_k^\top \delta$
- such that

$$\begin{pmatrix} \Phi^\top & \Omega^\top & 0 \\ -\Phi^\top & -\Omega^\top & -I_k \\ I'_n & \underline{D} & 0 \\ -I'_n & -\overline{D} & 0 \end{pmatrix} \begin{pmatrix} c \\ d \\ \delta \end{pmatrix} \le \begin{pmatrix} f \\ -f \\ 0 \\ 0 \end{pmatrix}, \tag{5}$$

where $\underline{D} = \operatorname{diag}(\underline{\phi}_1, \ldots, \underline{\phi}_n)$, $\overline{D} = \operatorname{diag}(\overline{\phi}_1, \ldots, \overline{\phi}_n)$, I_k is the identity matrix of order k, and I'_n is the $n \times (n+1)$ 'augmented' matrix $(0 : I_n$ where I_n is the identity matrix of order n.

Since the matrix in (5) has more rows than columns ($2(k+n)$ rows and $k+2n+1$ columns, where $k \ge 2n+1$), it is computationally more efficient to consider it as a dual problem, and to solve the equivalent primal. After some simple transformations, this primal problem reduces to:

$$\begin{cases} \min & f^\top y_1 - f^\top e_k \\ \text{s.t.} & \begin{pmatrix} \Phi & I'^\top_n & -I'^\top_n \\ \Omega & \underline{D} & -\overline{D} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \Phi e_k \\ \Omega e_k \end{pmatrix} \\ & y_1, y_2, y_3 \ge 0 \end{cases} \tag{6}$$

which has only $2n+1$ rows and $k+2n \ge 4n+1$ columns, and the obvious initial feasible solution $y_1 = e_k$ and $y_2 = y_3 = 0$. Furthermore, since the first of the $2n+1$ constraints in (6) in fact requires that $e_k^\top y_1 = 1$, then the function $f^\top y_1 - f^\top e_k$ is also bounded below, and so this primal linear program always has an optimal solution. This optimal solution gives the values of c, d, and δ via the dual vectors, and also determines which values of $f^{(j)}$ are interpolated by the potential function $U(\phi)$. That is, the basic columns in the optimal solution to (6) correspond to the conformations $\phi^{(j)}$ for which $F(\phi^{(j)}) = U(\phi^{(j)})$.

Note that once an optimal solution to (6) has been obtained, the addition of new local minima is very easy. It is done by simply adding new columns to Φ and Ω, and therefore to the constraint matrix in (6). The number of primal rows remains fixed at $2n+1$, independent of the number k of local minima.

The convex quadratic underestimating function $U(\phi)$ determined by the values $c \in \mathbf{R}^{n+1}$ and $d \in \mathbf{R}^n$ now provides a global approximation to the local minima of $F(\phi)$, and its easily computed global minimum point $\phi_{\min}$ is given by $(\phi_{\min})_i = -c_i/d_i$, $i = 1, \ldots, n$, with corresponding function value $U_{\min}$ given by $U_{\min} = c_0 - \sum_{i=1}^n c_i^2/d_i$. The value $U_{\min}$ is a good candidate for an approximation to the global minimum of the correct energy function $F(\phi)$, and so $\phi_{\min}$ can be used as an initial starting point around which additional configurations (i. e., local minima) should be generated. These local minima are added to the constraint matrix in (6) and the process is repeated. Before each iteration of this process, it is necessary to reduce the volume of the hyperrectangle $H\phi$ over which the new configurations are produced so that a tighter fit of $U(\phi)$ to the local minima 'near' $\phi_{\min}$ is constructed.

The rate and method by which the hyperrectangle size is decreased, and the number of additional local minima computed at each iteration must be determined by computational testing. But clearly the method depends most heavily on computing local minima quickly and on solving the resulting linear program efficiently to determine the approximating function $U(\phi)$ over the current hyperrectangle.

If E_c is a cutoff energy, then one means for decreasing the size of the hyperrectangle $H\phi$ at any step is to let $H\phi = \{\phi: U(\phi) \le E_c\}$. To get the bounds of $H\phi$, consider $U(\phi) \le E_c$ where $U(\phi)$ satisfies (2). Then limiting ϕ_i requires that

$$\sum_{i=1}^n \left(c_i\phi_i + \frac{1}{2} d_i \phi_i^2 \right) \le E_c - c_0. \tag{7}$$

As before, the minimum value of $U(\phi)$ is attained when $\phi_i = -c_i/d_i$, $i = 1, \ldots, n$. Assigning this minimum value to each ϕ_i, except ϕ_k, then results in

$$c_k\phi_k + \frac{1}{2} d_k \phi_k^2 \le E_c - c_0 + \frac{1}{2} \sum_{i \ne k} \frac{c_i^2}{d_i} \equiv \beta_k. \tag{8}$$

The lower and upper bounds on ϕ_k, $k = 1, \ldots, n$, are given by the roots of the quadratic equation

$$c_k\phi_k + \frac{1}{2}d_k\phi_k^2 = \beta_k. \tag{9}$$

Hence, these bounds can be used to define the new hyperrectangle $H\phi$ in which to generate new configurations.

Clearly, if E_c is reduced, the size of $H\phi$ is also reduced. At every iteration the predicted global minimum value $U_{\min}$ satisfies $U_{\min} \le F(\phi^*)$, where ϕ^* is the smallest known local minimum conformation. Therefore, $E_c = F(\phi^*)$ is often a good choice. If at least one improved point ϕ, with $F(\phi) < F(\phi^*)$, is obtained in each iteration, then the search domain $H\phi$ will strictly decrease at each iteration, and may decrease substantially in some iterations.

The CGU Algorithm

Based on the preceding description, a general method for computing the global, or near global, energy minimum of the potential energy function $F(\phi)$ can now be described.

1) Compute $k \ge 2n + 1$ distinct local minima $\phi^{(j)}$, for $j = 1, \ldots, k$, of the function $F(\phi)$.
2) Compute the convex quadratic underestimator function given in (2) by solving the linear program given in (6). The optimal solution to this linear program gives the values of c and d via the dual vectors.
3) Compute the predicted global minimum point $\phi_{\min}$ given by $(\phi_{\min})_i = -c_i/d_i$, $i = 1, \ldots, n$, with corresponding function value $U_{\min}$ given by $U_{\min} = c_0 - \sum_{i=1}^{n} c_i^2/(2d_i)$.
4) If $\phi_{\min} = \phi^*$, where $\phi^* = \text{argmin}\{F(\phi^{(j)}): j = 1, 2, \ldots\}$ is the best local minimum found so far, then stop and report ϕ^* as the approximate global minimum conformation.
5) Reduce the volume of the hyperrectangle $H\phi$ over which the new configurations will be produced, and remove all columns from Φ and Ω which correspond to the conformations which are excluded from $H\phi$.
6) Use $\phi_{\min}$ as an initial starting point around which additional local minima $\phi^{(j)}$ of $F(\phi)$ (restricted to the hyperrectangle $H\phi$) are generated. Add these new local minimum conformations as columns to the matrices Φ and Ω.
7) Return to step 2.

The number of new local minima to be generated in step 6 is unspecified since there is currently no theory to guide this choice. In general, a value exceeding $2n + 1$ would be required for the construction of another convex quadratic underestimator in the next iteration (step 2). In addition, the means by which the volume of the hyperrectangle $H\phi$ is reduced in step 5 may vary. One could use the two roots of (7) to define the new bounds of $H\phi$. Another method would be simply to use $H\phi = \{\phi_i: (\phi_{\min})_i - \delta_i \le \phi_i \le (\phi_{\min})_i + \delta_i\}$ where $\delta_i = |(\phi_{\min})_i - (\phi^*)_i|$, $i = 1, \ldots, n$.

For complete details of the CGU method and its computational results, see [5,8].

See also

- Adaptive Simulated Annealing and its Application to Protein Folding
- Genetic Algorithms
- Global Optimization in Lennard–Jones and Morse Clusters
- Global Optimization in Protein Folding
- Monte-Carlo Simulated Annealing in Protein Folding
- Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach
- Packet Annealing
- Phase Problem in X-ray Crystallography: Shake and Bake Approach
- Protein Folding: Generalized-ensemble Algorithms
- Simulated Annealing
- Simulated Annealing Methods in Protein Folding

References

1. Abagyan RA (1993) Towards protein folding by global energy optimization. Federation of Europ Biochemical Soc: Lett 325:17–22
2. Androulakis IR, Maranas CD, Floudas CA (1997) Prediction of oligopeptide conformations via deterministic global optimization. J Global Optim 11:1–34
3. Benner SA, Gerloff DL (1993) Predicting the conformation of proteins: man versus machine. Federation of Europ Biochemical Soc: Lett 325:29–33
4. Dill KA (1990) Dominant forces in protein folding. Biochemistry 29(31):7133–7155

5. Dill KA, Phillips AT, Rosen JB (1997) Protein structure and energy landscape dependence on sequence using a continuous energy function. J Comput Biol 4(3):227–239
6. Merz K, Grand S Le (1994) The protein folding problem and tertiary structure prediction. Birkhäuser, Basel
7. Miyazawa S, Jernigan RL (1993) A new substitution matrix for protein sequence searches based on contact frequencies in protein structures. Protein Eng 6:267–278
8. Phillips AT, Rosen JB, Walke VH (1995) Molecular structure determination by global optimization. In: Pardalos PM, Xue GL, Shalloway D (eds) DIMACS. Amer Math Soc, Providence, pp 181–198
9. Richards FM (1991) The protein folding problem. Scientif Amer:54–63
10. Srinivasan R, Rose GD (1995) LINUS: A hierarchic procedure to predict the fold of a protein. PROTEINS: Struct Funct Genet 22:81–99
11. Sun S, Thomas PD, Dill KA (1995) A simple protein folding algorithm using binary code and secondary structure constraints. Protein Eng 8(8):769–778
12. Yue K, Dill KA (1996) Folding proteins with a simple energy function and extensive conformational searching. Protein Sci 5:254–261

Monotonic Optimization

SAED ALIZAMIR
Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA

MSC2000: 90C26, 65K05, 90C30

Article Outline

Introduction
Normal Sets and Polyblocks
Normal Sets
Polyblocks
Solution Method
Generalizations
Optimization of the Difference of Monotonic Functions
Discrete Monotonic Optimization
Applications
Conclusions
References

Introduction

The role of convexity in optimization theory has increased significantly over the last few decades. Despite this fact, a wide variety of global optimization problems are usually encountered in applications in which nonconvex models need to be tackled. For this reason, developing solution methods for specially structured nonconvex problems has become one of the most active areas in recent years. Although these problems are difficult by their nature, promising progress is achieved for some special mathematical structures. Among the solution methods developed for these special structures, *monotonic optimization*, first proposed by Tuy [9], is presented in this study.

Problems of optimizing monotonic functions of n variables under monotonic constraints arise in the mathematical modeling of a broad range of real-world systems, including in economics and engineering. The original difficulties of these problems can be reduced by a number of principles derived from their monotonicity properties. For example, in nonconvex problems in general, a solution which is known to be feasible or even locally optimal, does not provide any information about global optimality and the search should be continued on the entire feasible space, while for an increasing objective function, a feasible solution like z, would exclude the cone $z + R^n_+$ from the search procedure (for a minimization objective function). In a similar way, if $g(x)$ in a constraint like $g(x) \leq 0$ is increasing, then by knowing that z is infeasible for this constraint, the whole cone $z + R^n_+$ can be discarded from further consideration. This kind of information would obviously restrict the search space and may result in more efficient solution methods.

To formally present the general framework of the monotonic optimization problem, consider two vectors $x, x' \in R^n$. We say $x' \geq x$ (x' dominates x) if $x'_i \geq x_i \; \forall i = 1, \ldots, n$. We say $x' > x$ (x' strictly dominates x) if $x'_i > x_i \; \forall i = 1, \ldots, n$. Let $R^n_+ = \{x \in R^n | x \geq 0\}$ and $R^n_{++} = \{x \in R^n | x > 0\}$. If $a, b \in R^n$ and $a \leq b$, we define the box $[a, b]$ as the set of all $x \in R^n$ such that $a \leq x \leq b$. Similarly, let $[a, b) = \{x | a \leq x < b\}$ and $(a, b] = \{x | a < x \leq b\}$. A function $f : R^n \to R$ is called *increasing* on a box $[a, b] \in R^n$ if $f(x) \leq f(x')$ for $a \leq x \leq x' \leq b$. A function f is called *decreasing* if $-f$ is increasing. Any increasing or decreasing function is referred to as *monotonic*. It can be easily shown that the pointwise supremum of a bounded-above family of increasing functions and the pointwise infimum of a bounded-below family of increasing functions are increasing.

In monotonic optimization, the following problem is considered:

$$\begin{aligned} \text{Maximize (minimize)} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \le 1 \quad \forall i = 1, \ldots, m_1, \\ & h_j(x) \ge 1 \quad \forall j = 1, \ldots, m_2, \\ & x \in R_+^n, \end{aligned} \tag{1}$$

in which $f(x)$, $g_i(x)$, and $h_j(x)$ are increasing functions on R^n. A more general definition of this problem is presented in Sect. "Normal Sets and Polyblocks". Heuristically, $f(x)$ may be a cost function (profit function for the maximize problem), $g_i(x)$ may express some resource availability constraints, while $h_j(x)$ may be a family of utility functions which have to take a value at least as big as a goal.

The remainder of this article is organized as follows. We first describe the theory of normal sets and polyblocks in Sect. "Normal Sets and Polyblocks". Monotonic optimization algorithms are presented in Sect. "Solution Method". Section "Generalizations" contains two generalizations of monotonic optimization. Different class of applications for which monotonic optimization is adapted are discussed in Sect. 5 and finally conclusions are made in Sect. "Conclusions".

Normal Sets and Polyblocks

The theory of normal sets and polyblocks is the underlying principle for monotonic optimization. In this section, the definitions are presented as well as the main concepts and properties to help the reader to understand the upcoming algorithms. For more details and proofs see [5,9,10].

Normal Sets

A set $G \subset R_+^n$ is called *normal* if for any two points $x, x' \in R_+^n$ such that $x \le x' > x' \in G$ implies $x \in G$. Given any set $D \subset R_+^n$, the set $N[D]$, which is called the *normal hull* of D, is the smallest normal set containing D. In other words, $N[D]$ can be interpreted as the intersection of all normal sets that contain D. The intersection and the union of a family of normal sets are normal. If the normal set contains a point $u \in R_{++}^n$ we say it has a *nonempty interior*. Suppose that $g(x)$ is an increasing function over R_+^n. Define the *level set* of $g(x)$ as the set $G = \left\{x \in R_+^n | g(x) \le 1\right\}$. It can be shown that the level set of an increasing function is a normal set and it is closed if the function is lower semicontinuous.

Define $I(x) = \{i | x_i = 0\}$, $K_x = \{x' \in R_+^n | x'_i > x_i \ \forall i \notin I(x)\}$, and $\mathrm{cl}K_x = \left\{x' \in R_+^n | x' \ge x\right\}$. Then a point $y \in R_+^n$ is called an *upper boundary point* of a bounded normal set G if $y \in \mathrm{cl}G$ while $K_y \subset R_+^n \backslash G$. The set of upper boundary points of G is called the *upper boundary* of G and is denoted by $\partial^+ G$.

For a compact normal set $G \subset [0, b]$ with nonempty interior and for every point $z \in R_+^n \backslash \{0\}$, the half line from 0 through z meets $\partial^+ G$ at a unique point denoted by $\pi_G(z)$, which is defined as $\pi_G(z) = \lambda z$, $\lambda = \max\{\alpha > 0 | \alpha z \in G\}$.

A set $H \subset R_+^n$ is called a *reverse normal set* (also known as *conormal*) if $x' \ge x$ and $x \in H$ implies $x' \in H$. A reverse normal set in a box $[0, b]$ is defined as a set like $H \in R_+^n$ for which $0 \le x \le x' \le b$ and $x \in H$ implies $x' \in H$. As before, $rN[D]$ is the smallest reverse normal set containing $D \subset R_+^n$ and is called a *reverse normal hull* of set D. Define $H = \left\{x \in R_+^n | h(x) \ge 1\right\}$ for the increasing function $h(x)$. Then it can be shown that H is reverse normal and it is closed if $h(x)$ is upper semicontinuous.

A point $y \in R_+^n$ is called a *lower boundary point* of a reverse normal set H if $y \in \mathrm{cl}H$ and $x \notin H \ \forall x < y$. The set of lower boundary points of H is called the *lower boundary* of H and is denoted by $\partial^- H$.

For the closed reverse normal set H and $b \in \mathrm{int}H$ and every point $z \in [0, b] \backslash H$, the half line from b through z meets $\partial^- H$ at a unique point $\rho_H(z)$, which is defined as $\rho_H(z) = b + \mu(z - b)$, $\mu = \max\{\alpha > 0 | b + \alpha(z - b) \in H\}$.

Now consider the set of constraints imposed by increasing functions $g_i(x)$ and $h_j(x)$ in problem (1). The feasible space characterized by these sets of constraints can properly be presented by normal sets and reverse normal sets. Define the sets $G, H \subset R_+^n$ as $G = \left\{x \in R_+^n | g_i(x) \le 1 \ \forall i = 1, \ldots, m_1\right\}$ and $H = \left\{x \in R_+^n | h_j(x) \ge 1 \ \forall i = 1, \ldots, m_2\right\}$. Then by the basic properties of normal and reverse normal sets which were described above, G is the intersection of a finite number of normal sets which is normal. In a similar way, H is the intersection of a finite number of reverse normal sets which is reverse normal. Now we can redefine the fundamental problem of monotonic optimization, also called the *canonical monotonic optimization*

problem, as optimizing a monotonic function on the intersection of a family of normal and reverse normal sets as follows:

$$\begin{aligned} &\text{Maximize (minimize)} \quad f(x) \\ &\qquad\qquad \text{subject to} \quad x \in G \cap H\,, \end{aligned} \tag{2}$$

in which $G \subset [0,\, b] \subset R^n_+$ is a compact normal set, H is a close reverse normal set, and $f(x)$ is an increasing function on $[0, b]$. Tuy [9] proved that if G has a nonempty interior (if $b \in \text{int}H$), then the maximum (minimum) of $f(x)$ over $G \cap H$, if it exists, is attained on $\partial^+ G \cap H$ ($G \cap \partial^- H$). On the basis of this essential result, it can be shown that for every arbitrary compact set $D \subset R^n_+$, $\max\{f(x)|x \in D\} = \max\{f(x)|x \in N[D]\}$. Analogously, for the minimization version of the objective function, for any arbitrary set $E \subset R^n_+$, we have $\min\{f(x)|x \in E\} = \min\{f(x)|x \in rN[E]\}$.

It is worth mentioning that the minimization problem can be converted to the maximization case by making a simple set of transformations. So it can be either transformed to the maximization problem or treated separately.

Polyblocks

The role of *polyblocks* in monotonic optimization is the same as that of the polytope in convex optimization. As the polytope is the convex hull of finitely many points in R^n, a polyblock is the normal hull of finitely many points in $R^n{}_+$. A set $P \subset R^n_+$ is a polyblock in $[a,\, b] \subset R^n_+$ if it is the union of a finite number of boxes $[a, z], z \in T \subset [a, b]$. The set T is called the *vertex set* of the polyblock. We call the vertex $z \in T$ a *proper* vertex if $z \notin [0,\, z'] \; \forall z' \in T \backslash \{z\}$, i. e., by removing the vetex z from T, the new polyblock created by T is not equivalent to P. A vertex which is not proper is called an *improper* vertex. A polyblock can be defined by the set of its proper vertices.

A polyblock is a closed normal set and the intersection of a set of polyblocks is again a polyblock. Now suppose that $x \in [a,\, b]$ and consider the set $P = [a,\, b] \backslash (x,\, b]$. Then it is easy to verify that P is a polyblock with vertices $z^i = b + (x - b)e^i,\ \forall i = 1, \ldots, n$ in which e^i is the ith unit vector. Using this property, we can approximate an arbitrary compact normal set $\Omega \subset R^n_+$ (with any desired accuracy) by a nested sequence of polyblock approximation. At each iteration, a point $x \notin \Omega$ is found and a new polyblock is constructed based on that which is a subset of the previous polyblock but still contains the set Ω.

To present the main idea of the *polyblock approximation method* in monotonic optimization, we need one more result on optimizing an increasing function over a polyblock. Tuy [9] proved that the increasing function $f(x)$ achieves its maximum over a polyblock at a proper vertex.

Now consider the problem of maximizing the increasing function $f(x)$ over the arbitrary compact set $\Omega \subset R^n_+$. As mentioned before, we can substitute Ω by its normal hull. So without loss of generality, we assume that Ω is normal. The idea is to construct a nested sequence of polyblock outer approximation $P_1 \supset P_2 \supset \ldots \supset \Omega$ in such a way that $\max\{f(x)|x \in P_k\} \searrow \max\{f(x)|x \in \Omega\}$.

At iteration k, assume z^k is the proper vertex of P_k which maximizes $f(x)$, i. e., $z^k = \arg\max\{f(z)|z \in T_k\}$, where T_k is the set of proper vertices of P_k. Then if z^k is feasible in Ω, the initial feasible space, it also solves the problem. Otherwise, we are interested in a new polyblock $P_{k+1} \subset P_k \backslash \{z^k\}$ which still contains Ω as a subset.

To obtain P_{k+1} from P_k, the box $[0,\, z^k]$ is replaced by $[0,\, z^k] \backslash K_{x^k}$, in which x^k is defined as $\pi(z^k)$. Mathematically, $P_{k+1} = ([0, z^k] \backslash K_{x^k}) \bigcup_{z \in T_k \backslash \{z^k\}} [0, z]$, which clearly satisfies the desired property of $\Omega \subset P_{k+1} \subset P_k \backslash \{z^k\}$.

The vertex set of the established polyblock P_{k+1}, denoted by V_{k+1}, contains the proper vertices of P_k excluding z^k and a set of n new vertices, $z^{k,1}$, $z^{k,2}, \ldots, z^{k,n}$, defined as $z^{k,i} = z^k + (x^k_i - z^k_i)e^i$. This result is directly followed by the earlier-mentioned property of polyblocks about the vertices of $[a,\, b] \backslash (x,\, b]$. Finally, the proper vertex set of P_{k+1}, T_{k+1}, is obtained from V_{k+1} by removing its improper vertices [9,10].

A set $P \subset R^n_+$ is called a *reverse polyblock* in $[0, b]$ if it is the union of a finite number of boxes $[z,\, b], z \in T,\ T \subset [0,\, b]$. The set T is called the *vertex set* of the reverse polyblock. As before, z is a *proper* vertex if by removing it from T, the new reverse polyblock created by T is not equivalent to P. A reverse polyblock can be defined by the set of its proper vertices. An increasing function $f(x)$ achieves its minimum over a re-

verse polyblock at a proper vertex. Similar results to what we had for polyblocks can be developed for reverse polyblocks in the very same way. For more details see [9,10].

Solution Method

Consider problem (2) (in the maximization form) as discussed in Sect. "Normal Sets and Polyblocks" with the additional assumptions that $f(x)$ is semicontinuous on H and $G \cap H \subset R^n_{++}$. The latter assumption implies the existence of a vector a such that $0 < a \leq x,\ \forall x \in G \cap H$. Let $H_a = \{x \in H | x \geq a\}$. For $\epsilon \geq 0$ as a given tolerance, the solution x' is called ϵ-optimal if $f(x') \geq f(x) - \epsilon,\ \forall x \in G \cap H$. We attempt to design an algorithm which is capable of finding an ϵ-optimal solution for any given ϵ.

Obviously, $b \in H$ because otherwise the problem is infeasible. Let $P_1 = [0,\ b]$ be the initial polyblock and $T_1 = \{b\}$ its corresponding proper vertex set. If we apply the polyblock approximation method described in Sect. "Normal Sets and Polyblocks" to this problem, at each iteration k, P_k and its proper vertex set, T_k, are obtained from the last iteration. We should notice that every vertex $z \in T_k \backslash H_a$ can be removed since they do not belong to the initial feasible space. Also suppose that $f(x^k)$ is the best value found for the objective function so far. Then any vertex z for which $f(z) \leq f(x^k) + \epsilon$ is discarded because no ϵ-optimal solution happens to be in box $[0, z]$. These two rules can be applied at each iteration to refine the proper vertex set T_k and delete some of the vertices from further consideration.

If $T_k = \emptyset$ in some iteration k, it means there is no solution x for which $f(x) > f(x^k) + \epsilon$. So, x^k, the best solution found so far, is ϵ-optimal and the procedure terminates. Otherwise, let $z^k = \arg\max \{f(z) | z \in T_k\}$. If z^k is feasible in $G \cap H$, it solves the problem. Since $z^k \in H$ is always true, it is feasible if it belongs to G and infeasible otherwise. In the case of infeasibility, we find $x^k = \pi_G(z^k)$ and construct the polyblock P_{k+1} as described in Sect. "Normal Sets and Polyblocks" which excludes z^k while still containing a global optimal solution of the problem. This procedure is repeated until the termination criteria are satisfied or the problem is known to be infeasible. This procedure, first proposed by Tuy [9], is called the *polyblock algorithm*. Tuy [9] discussed the convergence of this method and showed that as $k \to \infty$, the sequence x^k converges to a global optimal solution of the problem.

Now consider the minimization case of problem (2) in Sect. "Normal Sets and Polyblocks" with additional assumptions that $f(x)$ is semicontinuous on G and there exists a vector c such that $0 < c < b$ and $0 \leq x \leq c,\ \forall x \in G \cap H$. A nested sequence of reverse polyblock outer approximation of $G \cap H$ (or a subset of $G \cap H$ in which the existence of at least one optimal solution is guaranteed) is called the *reverse polyblock algorithm* (*copolyblock algorithm*) which is devised to solve this problem [9].

The polyblock approximation algorithm works properly for relatively small dimension n, typically $n = 10$. However, the algorithm converges slowly as it gets closer to the global optimal solution and needs a large number of iterations even for a value of n as small as 5. Tuy et al. [12] presented two main reasons for this drawback of the algorithm. First, the speed of convergence depends on the way in which we construct the current polyblock from the previous one. Obviously, we prefer to remove a larger portion of the previous polyblock to have a smaller search space and a higher speed of convergence. This goal is achieved by employing more complex rules of constructing the polyblocks, which imposes some additional computational effort. The second source of the slowness of the algorithm is how it selects the solution x_k in each iteration. These solutions are basically derived from the monotonicity properties of the problem, while sometimes there may exist some amount of convexity which can be used to speed up the algorithm.

Tuy and Al-Khayyal [11] introduced the concept of *reduced box* and *reduced polyblock*. It involves tightening the box in which we are interested to find the upper bound of $f(x)$, in such a way that the reduced box still contains an optimal solution of the problem. Then based on that, a new procedure is developed to produce tighter polyblocks. They also redefined the proper vertex set of polyblocks in the algorithm and suggested that instead of selecting x^k as the last point of G on the halfline from a through z^k, as the original algorithm does, a more complex way can be implemented by incorporating some of the convexity properties of the problem. This is by solving the convex relaxation of the problem $\max \{f(x) | x \in G \cap H, x \in [a,\ z^k]\}$ which gives us an upper bound of $f(x)$ over the feasible solu-

tion x in box $[a, x^k]$. Similar ideas were applied to the reverse polyblock algorithm as well. Using these two new modifications and improvements, they developed new algorithms and discussed their convergence properties, namely, the *revised polyblock algorithm* and the *revised reverse polyblock (copolyblock) algorithm*.

Most of the outer approximation procedures, including the polyblock algorithm, encounter storage and numerical problems while solving problems in high dimensions. By using branch-and-bound strategies, one can tackle these difficulties. Bounding is performed on the basis of the polyblock approximation. As before, monotonicity cuts and convex relaxation can be combined to enhance the quality of the bounds in the corresponding portion of the feasible space. In this branch-and-bound approach, branching is performed as partitioning the feasible space into cones pairwise having no common interior point. The logic behind using conical partitioning instead of rectangular partitioning is the fact that the optimal solution of the monotonic optimization problem, as discussed before, is always achieved on the upper boundary of the feasible normal set. Using conical partitioning is more efficient and less expensive in terms of the computational time.

The algorithm starts with initial cone R^n_+ and partitions it into subcones. For each of these subcones, an upper bound for the value of the objective function over the feasible solutions contained in it is derived. Those cones which are known to not contain an optimal solution are fathomed and the remaining ones are subdivided again and the procedure is repeated until the termination criteria are satisfied. Among the remaining cones, the one having the maximal bound is the first candidate for branching. This algorithm, suggested by Tuy and Al-Khayyal [11], is called the *conical algorithm*.

For those problems having partial monotonicity and partial convexity, this branch-and-bound scheme can be extended to devise a more general method. In this method, branching is performed on the nonconvex variables and bounds are computed by Lagrangian or convex relaxation [6].

To further exploit the monotonic structure of the problem, *reduction cuts* are combined with original monotonicity cuts and a more efficient method is developed [13]. This method creates branch-and-cut algorithms to solve monotonic optimization problems by systematic use of these cuts.

Finally, it is worth mentioning that a new concept of the *essential ϵ-optimal solution* can be applied to monotonic optimization problems. The advantage of the method developed on the basis of this concept is the finding of an approximate optimal solution which is more appropriate and more stable than that which is found by the ϵ-optimal method. For details see [8].

Generalizations

The essential approach used in monotonic optimization can be further generalized to cover a wider class of non-convex general optimization problems. Among these generalizations, optimization of the difference of monotonic functions and discrete monotonic optimization are presented here.

Optimization of the Difference of Monotonic Functions

The underlying idea of monotonic optimization can be extended to deal with problems including the *difference of monotonic functions*. A function $f : R^n_+ \to R$ is said to be a difference of monotonic functions if it is representable as the difference of two increasing functions: $f_1 : R^n_+ \to R$ and $f_2 : R^n_+ \to R$. Similar to functions presented as the difference of convex functions, the class of difference of monotonic functions is a linear space. The pointwise minimum and pointwise maximum of a family of difference of monotonic functions (difference of convex functions) is still a difference of monotonic functions (difference of convex functions). The linear combination of a set of difference of monotonic functions is a difference of monotonic functions. Obviously, any polynomial function can be presented as the difference of two increasing functions, the first one includes all terms having positive coefficients and the second one includes all terms having negative coefficients.

Consider the problem:

$$\begin{aligned} \text{Maximize (minimize)} \quad & f(x) - g(x) \\ \text{subject to} \quad & x \in G \cap H\,, \end{aligned} \tag{3}$$

in which G and H are as before and $f(x)$ and $g(x)$ are increasing functions on $[0, b]$. Tuy [9] extended the original polyblock algorithm to solve this problem. By introducing t as the difference between $g(b)$

and $g(x)$ for $x \in [0, b]$ and regarding the fact that t is always positive owing the function $g(x)$ being increasing, we rewrite the model as (maximization case) $\max\{f(x) + t - g(b)|x \in G \cap H, t = g(b) - g(x)\}$. Now $g(b)$ is a constant and can be removed from the objective function. In the resulting problem, $\max\{f(x) + t|x \in G \cap H, 0 \le t \le g(b) - g(x)\}$, consider the set of constraints. By incrementing the dimension of the problem by one, the feasible space can be presented as $D \cap E$ such that $D = \{(x, t)|x \in G, t + g(x) \le g(b), 0 \le t \le g(b) - g(0)\}$ and $E = \{(x, t)|x \in H, 0 \le t \le g(b) - g(0)\}$. It is easy to verify that D is a normal set and H is a reverse normal set in the box $[0, b] \times [0, g(b) - g(0)]$. Also the function $F(x, t) = f(x) + t$ is an increasing function on $[0, b] \times [0, g(b) - g(0)]$. So problem (3) is reduced to problem (2) in Sect. "Normal Sets and Polyblocks" and can be treated by the original polyblock algorithm. The additional cost that the presence of difference of monotonic functions has incurred is the dimension of the problem incremented by one.

For the minimization case of problem (3), a similar transformation can be applied to convert this problem to the minimization case of problem (2).

To make the problem even more general, suppose that all constraints are also difference of monotonic functions. Specifically, consider the problem:

$$\begin{aligned} \text{Maximize (minimize)} \quad & f_1(x) - f_2(x) \\ \text{subject to} \quad & g_i(x) - h_i(x) \le 0 \\ & \forall i = 1, \ldots, m, \\ & x \in \Omega \subset [0, b] \subset R_+^n, \end{aligned} \tag{4}$$

in which $f_1(x)$, $f_2(x)$, $g_i(x)$, and $h_i(x)$ are increasing functions and Ω is a normal set. By the above argument, first we can make a proper transformation and convert the objective function to an increasing function. So without loss of generality, let us assume that $f_2(x) = 0$. Now consider the set of m constraints. This set of constraints can be rewritten as $\max_i \{g_i(x) - h_i(x)\} \le 0$. Since the pointwise maximum of a family of difference of monotonic functions is still a difference of monotonic functions, we can represent the space imposed by these constraints by $g(x) - h(x) \le 0$, where both $g(x)$ and $h(x)$ are increasing. By introducing the new variable $t \ge 0$ and assuming $g(b) \ge 0$ (this assumption is not restrictive), the set of the following two constraints fully defines the space mentioned: $g(x) + t \le g(b)$, $h(x) + t \ge g(b)$. The first constraint gives us the upper bound of $g(b) - g(0)$ for t.

Finally the problem reduces to (maximization case): $\max\{f_1(x)|g(x) + t \le g(b), h(x) + t \ge g(b), x \in \Omega, 0 \le t \le g(b) - g(0)\}$. This problem is the same as problem (2) by defining $G = \{(x, t)| x \in \Omega, g(x) + t \le g(b), 0 \le t \le g(b) - g(0)\}$, which is a subset of the box $[0, b] \times [0, g(b) - g(0)]$ and $H = \{(x, t)|h(x) + t \ge g(b)\}$ is defined in R_+^{n+1}.

Increasing the dimension of the problem is the main drawback of the above mentioned approach. Tuy and Al-Khayyal [11] presented a direct approach for the difference of monotonic functions optimization problem requiring no additional dimension. This method is referred to as the branch-reduce-and-bound (BRB) algorithm. As the name of the algorithm suggests, it contains three main steps, which are branching upon nonconvex variables, reducing any partition set before bounding, and bounding over each partition set.

The branching phase is performed by rectangular subdivision. Every box is divided into two subboxes by a hyperplane. The reduction phase contains a set of operations by which the box $[p, q]$ is tightened without losing any feasible solution. This is called a *proper reduction* of [p,q]. This approach takes advantage of the monotonicity properties of the problem and increases the rate of convergence in the algorithm. In the bounding phase, for a properly reduced box $[p, q]$, an upper bound like β is obtained such that $\beta \ge \max\{f_1(x) - f_2(x)|g_i(x) - h_i(x) \le 0, \forall i = 1, \ldots, m; x \in [p, q]\}$. As mentioned before, stronger bounds are obtained by a sequence of polyblock approximations or by combining monotonicity with convexity present in the problem. Furthermore, more complex methods can be applied to improve the quality of the bounds in the bounding phase.

Discrete Monotonic Optimization

A class of monotonic optimization problems containing the additional discrete constraints are called *discrete monotonic optimization problems*. Specifically, given a finite set S of points in the box [a,b], the constraint $x \in S$ is added to the model. So the problem can be represented as $\max\{f(x)|x \in G \cap H \cap S\}$ (all the assumptions are as in problem (2).

The original polyblock algorithm is not practical for these problems. Since the polyblock algorithm is an iterative procedure, it does not have the capability to produce the optimal solution in a finite number of iterations. However, by making suitable modifications, one can use this algorithm to obtain the exact optimal solution of the problem in a finite number of steps [1,14]. In the new method, monotonicity cuts are adjusted on the basis of a special procedure to cope with discrete requirements. This adjustment consists in updating the vertex of the monotonicity cut by pushing it deeper inside the polyblock to obtain a tighter space while keeping all discrete points which are not proven to be nonoptimal, unaffected.

The algorithm first constructs the normal hull of $G \cap S$, denoted by $\tilde{G}$, and then tries to solve the problem $\max\{f(x)|x \in \tilde{G} \cap H\}$ in continuous space. This method is called the *discrete polyblock algorithm*. For large-scale instances, a similar BRB algorithm was developed by Tuy et al. [14].

Applications

Although monotonic optimization is a new approach in global optimization and there is not a broad literature on its applications, it can be applied to numerous problems. In most of these applications, first some transformations are performed and the problems are reformulated in the proper way. Then monotonic optimization is applied and other approaches are employed to enhance the quality of the bounds. Some of these applications are briefly introduced in this section.

Polynomial programming: The problem of minimizing or maximizing a polynomial function under a set of polynomial constraints, which is encountered in a multitude of applications, is called *polynomial programming*. Tuy [9] reformulated this problem as a difference of monotonic functions problem which can be solved by the methods described before. Tuy [7] proposed a robust solution approach for polynomial programming based on a monotonic optimization scheme. He developed a BRB procedure to tackle the polynomial optimization problems of higher dimensions.

Polynomial optimization contains nonconvex quadratic programming as a special case. So every polynomial optimization method can be applied to solve this important class of problems [4,16].

Fractional programming: In *fractional programming*, we are dealing with functions which are represented by ratios of other functions. Phuong and Tuy [3] considered a generalized linear fractional programming problem. In this problem, the objective function consists of an arbitrary continuous increasing function of m linear fractional functions and the feasible set is the polytope D. Linear fractional functions are defined as the ratio of two linear affine functions. They proposed a new unified approach which reformulates the problem and solves it as a monotonic optimization problem.

Tuy [17] considered a more general class of fractional programming problems which is optimizing a polynomial fractional function (the ratio of two polynomial functions) under polynomial constraints. His method to solve the problem is again based on reformulating the problem as a monotonic optimization problem. A branch-and-bound scheme was presented for problems of higher ranks. Clearly, polynomial programming is a special case of this class of problems.

Multiplicative programming: *Multiplicative programming* problems are optimization problems containing products of a number of convex or concave functions in the objective function or constraints. Tuy [9] showed that these classes of problems are essentially monotonic optimization problems. Tuy and Nghia [15] devised a new approach based on the reverse polyblock approximation method for a broad class of problems including generalized linear multiplicative and linear fractional programming as special cases.

For more applications, including *Lipschitz optimization*, *optimization under network constraints*, the *Fekete points problem*, and the *Lennard-Jones potential energy function*, see [9].

Conclusions

We have discussed the recently developed theory of monotonic optimization as well as its generalizations and applications. This noble scheme which is capable of solving a wide range of nonconvex problems is based on an polyblock outer approximation procedure.

The approach that monotonic optimization uses to deal with optimization problems is analogous to convex optimization in several respects. Just as we approx-

imate convex sets by polyhedrons, normal sets, defined as the level sets of increasing functions, can be approximated by a set of polyblocks in monotonic optimization. As the difference of convex functions plays an essential role in convex analysis (because any arbitrary continuous function can be represented as the difference of two convex functions), optimization problems representable as the difference of monotonic functions can be treated in monotonic optimization.

The performance of this method can be significantly improved by incorporating some other techniques like convex relaxation to exploit other properties present in the problem. In high dimensions, branch-and-bound or branch-and-cut extensions of the algorithm can be applied to overcome storage difficulties and increase the convergence speed.

References

1. Minoux M, Tuy H (2001) Discrete Monotonic Global Optimization. preprint. Institute of of Mathematics, Hanoi
2. Pardalos PM, Romeijn HE, Tuy H (2000) Recent developments and trends in global optimization. J Comput Appl Math 124:209–228
3. Phuong NTH, Tuy H (2003) A Unified Monotonic Approach to Generalized Linear Fractional Programming. J Global Optim 26:229–259
4. Phuong NTH, Tuy H (2002) A Monotonicity Based Approach to Nonconvex Quadratic Minimization. Vietnam J Math 30:373–393
5. Rubinov A, Tuy H, Mays H (2001) An Algorithm for Monotonic Global Optimization Problems. Optimization 49:205–221
6. Tuy H (2005) Partly Convex and Convex-Monotonic Optimization Problems. preprint, Institute of Mathematics, Hanoi
7. Tuy H (2005) Polynomial Optimization: A Robust Approach. preprint, Institute of Mathematics, Hanoi
8. Tuy H (2005) Robust Solution of Nonconvex Global Optimization Problems. J Global Optim 32:307–323
9. Tuy H (2000) Monotonic Optimization: Problems and Solution Approaches. SIAM J Optim 11:464–494
10. Tuy H (1999) Normal sets, Polyblocks, and Monotonic Optimizatin. Vietnam J Math 27:277–300
11. Tuy H, Al-Khayyal F (2003) Monotonic Optimization Revisited. preprint, Institute of Mathematics, Hanoi
12. Tuy H, Al-Khayyal F, Ahmed S (2001) Polyblock Algorithms Revisited. preprint, Institute of Mathematics, Hanoi
13. Tuy H, Al-Khayyal F, Thach PT (2005) Monotonic Optimization: Branch and Cut Methods. In: Audet C, Hansen P, Savard G (eds) Essays and Surveys in Global Optimization. Springer US, pp 39–78
14. Tuy H, Minoux M, Phuong NTH (2006) Discrete Monotonic Optimization with Application to a Discrete Location Problem. SIAM J Optim 17:78–97
15. Tuy H, Nghia ND (2001) Reverse Polyblock Approximation for Generalized Multiplicative/Fractional Programming. preprint, Institute of Mathematics, Hanoi
16. Tuy H, Phuong NTH (2007) A robust algorithm for quadratic optimization under quadratic constraints. J Global Optim 37:557–569
17. Tuy H, Thach PT, Konno H (2004) Optimization of Polynomial Fractional Functions. J Global Optim 29:19–44

Monte-Carlo Simulated Annealing in Protein Folding

YUKO OKAMOTO
Department Theoret. Stud. Institute Molecular Sci. and Department Functional Molecular Sci., Graduate University Adv. Stud., Okazaki, Japan

MSC2000: 92C40

Article Outline

Keywords
Introduction
Energy Functions of Protein Systems
Methods
Results
Conclusions
See also
References

Keywords

Simulated annealing; Protein folding; Tertiary structure prediction; α-helix; β-sheet

We review uses of Monte-Carlo simulated annealing in the protein folding problem. We will discuss the strategy for tackling the protein folding problem based on all-atom models. Our approach consists of two elements: the inclusion of accurate solvent effects and the development of powerful simulation algorithms that can avoid getting trapped in states of energy local minima. For the former, we discuss several models varying in nature from crude (distance-dependent dielectric function) to rigorous (reference interaction site model).

For the latter, we show the effectiveness of Monte-Carlo simulated annealing.

Introduction

Proteins under their native physiological conditions spontaneously fold into unique three-dimensional structures (tertiary structures) in the time scale of milliseconds to minutes. Although protein structures appear to be dependent on various environmental factors within the cell where they are synthesized, it was inferred by experiments 'in vitro' that the three-dimensional structure of a protein is determined solely by its amino-acid sequence information [12]. Hence, it has been hoped that once the correct Hamiltonian of the system is given, one can predict the native protein tertiary structure from the first principles by computer simulations. However, this has yet to be accomplished. There are two reasons for the difficulty. One reason is that the inclusion of accurate solvent effects is nontrivial, because the number of solvent molecules that have to be considered is very large. The other reason for the difficulty comes from the fact that the number of possible conformations for each protein is astronomically large [30,60]. Simulations by conventional methods such as Monte-Carlo or molecular dynamics algorithms in canonical ensemble will necessarily be trapped in one of many local-minimum states in the energy function. In this article, I will discuss a possible strategy to alleviate these difficulties. The outline of the article is as follows. In Sect. "Energy Functions of Protein Systems" we summarize the energy functions of protein systems that we used in our simulations. In Sect. "Methods" we briefly review our simulation methods. In Sect. "Results" we present the results of our protein folding simulations. Section "Conclusions" is devoted to conclusions.

Energy Functions of Protein Systems

The energy function for the protein systems is given by the sum of two terms: the conformational energy E_{P} for the protein molecule itself and the solvation free energy E_{S} for the interaction of protein with the surrounding solvent. The conformational energy function E_{P} (in kcal/mol) for the protein molecule that we used is one of the standard ones. Namely, it is given by the sum of the electrostatic term E_{C}, 12-6 Lennard–Jones term E_{LJ}, and hydrogen-bond term E_{HB} for all pairs of atoms in the molecule together with the torsion term E_{tor} for all torsion angles:

$$\begin{cases} E_{\mathrm{P}} = E_{\mathrm{C}} + E_{\mathrm{LJ}} + E_{\mathrm{HB}} + E_{\mathrm{tor}}, \\ E_{\mathrm{C}} = \sum_{(i,j)} \dfrac{332 q_i q_j}{\epsilon r_{ij}}, \\ E_{\mathrm{LJ}} = \sum_{(i,j)} \left(\dfrac{A_{ij}}{r_{ij}^{12}} - \dfrac{B_{ij}}{r_{ij}^{6}} \right), \\ E_{\mathrm{HB}} = \sum_{(i,j)} \left(\dfrac{C_{ij}}{r_{ij}^{12}} - \dfrac{D_{ij}}{r_{ij}^{10}} \right), \\ E_{\mathrm{tor}} = \sum_{i} U_i \left(1 \pm \cos(n_i \chi^i) \right). \end{cases} \tag{1}$$

Here, r_{ij} is the distance (in Å) between atoms i and j, ϵ is the dielectric constant, and χ^i is the torsion angle for the chemical bond i. Each atom is expressed by a point at its center of mass, and the partial charge q_i (in units of electronic charges) is assumed to be concentrated at that point. The factor 332 in E_{C} is a constant to express energy in units of kcal/mol. These parameters in the energy function as well as the molecular geometry were adopted from ECEPP/2 [37,41,57]. The computer code KONF90 [23,46] was used for all the Monte-Carlo simulations. For gas phase simulations, we set the dielectric constant ϵ equal to 2. The peptide-bond dihedral angles ω were fixed at the value 180° for simplicity. So, the remaining dihedral angles ϕ and ψ in the main chain and χ in the side chains constitute the variables to be updated in the simulations. One Monte-Carlo (MC) sweep consists of updating all these angles once with Metropolis evaluation [36] for each update.

Solvation free energy of interactions between a solute molecule and solvent molecules, in general, can be divided into three contributions: hydrophobic term that corresponds to the work required to create a cavity of the shape of the solute molecule in solution (the term 'hydrophobic' used in this article is different from a more standard one; see [11] for clarification on various definitions), the electrostatic term (including the hydrogen-bond energy) between solute and solvent molecules, and the Lennard–Jones term between solute and solvent molecules.

One of the simplest ways to represent solvent effects is by the sigmoidal, distance-dependent dielectric function [20,54]. The explicit form of the function we used

is given by [43]

$$\epsilon(r) = D - \frac{D-2}{2}\left[(sr)^2 + 2sr + 2\right]e^{-sr}, \qquad (2)$$

which is a slight modification of the one used in [9]. Here, we use $s = 0.3$ and $D = 78$. It approaches 2 (the value inside a protein) in the limit the distance r going to zero and 78 (the value for bulk water) in the limit r going to infinity. The distance-dependent dielectric function is simple and also computationally only slightly more demanding than the gas-phase case. But it only involves the electrostatic interactions. Other solvent contributions are hydrophobic interactions and Lennard–Jones interactions between protein and solvent.

Another commonly used term that represents solvent contributions is the term proportional to the solvent-accessible surface area of protein molecule. The solvation free energy E_S in this approximation is given by

$$E_S = \sum_i \sigma_i A_i, \qquad (3)$$

where A_i is the solvent-accessible surface area of ith functional group, and σ_i is the proportionality constant. There are several versions of the set of the proportionality constants and functional groups. Five parameter sets were compared for the systems of peptides and a small protein, and we found that the parameter sets of [52,59] are valid ones [33]. The term in (3) includes all the contributions from solvent (namely, hydrophobic, electrostatic, and Lennard–Jones interactions), and it is therefore more accurate than the distance-dependent dielectric function. It is, however, an empirical representation, and its validity has to be eventually tested with a rigorous solvation theory.

The most widely-used and rigorous method of inclusion of solvent effects is probably the one that deals with the explicit solvent molecules with all-atom representations. Many molecular dynamics simulations of protein systems now directly include these explicit solvent molecules (for a review, see, for instance, [4]). Another rigorous method is based on the statistical mechanical theory of liquid and solution and is called the reference interaction site model (RISM) [7,21]. The RISM integral equation for solute-solvent (p-s) correlation functions in Fourier k-space is given by

$$\widetilde{\mathbf{h}}^{\mathbf{ps}} = \widetilde{\mathbf{w}}^{\mathbf{pp}}\widetilde{\mathbf{c}}^{\mathbf{ps}}\left(\widetilde{\mathbf{w}}^{\mathbf{ss}} + \boldsymbol{\rho}\widetilde{\mathbf{h}}^{\mathbf{ss}}\right), \qquad (4)$$

where $\widetilde{\mathbf{h}}^{\mathbf{ps}}$ and $\widetilde{\mathbf{h}}^{\mathbf{ss}}$ are the matrices of the solute-solvent and the solvent-solvent total correlation functions, respectively, $\widetilde{\mathbf{c}}^{\mathbf{ps}}$ is the matrix of the solute-solvent direct correlation functions, $\widetilde{\mathbf{w}}^{\mathbf{pp}}$ and $\widetilde{\mathbf{w}}^{\mathbf{ss}}$ are the intramolecular correlation matrices for solute and solvent, respectively, and $\boldsymbol{\rho}$ is the number density matrix of the solvent. The solvation free energy is given by

$$E_S = 4\pi\rho k_B T \int_0^\infty r^2 F(r)\,\mathrm{d}r, \qquad (5)$$

where $F(r)$ is defined by

$$F(r) \equiv \sum_{a,b}\left\{\frac{1}{2}h_{ab}^{ps}(r)^2 - c_{ab}^{ps}(r) - \frac{1}{2}h_{ab}^{ps}(r)c_{ab}^{ps}(r)\right\}. \qquad (6)$$

Here, the summation indices a and b run over the solute and the solvent sites, respectively. A robust and fast algorithm for solving RISM equations was recently (as of 1999) developed [24], which made folding simulations of peptides a feasible possibility [25]. Although this method is computationally much more time-consuming than the first two methods (terms with distance-dependent dielectric function and those proportional to surface area), it gives the most accurate representation of the solvation free energy.

Methods

Once the appropriate energy function of the protein system is given, we have to employ a simulation method that does not get trapped in states of energy local minima. We have been advocating the use of Monte-Carlo simulated annealing [27].

In the regular canonical ensemble with a given inverse temperature $\beta \equiv 1/k_B T$, the probability weight of each state with energy E is given by the Boltzmann factor:

$$W_B(E) = \exp(-\beta E). \qquad (7)$$

The probability distribution in energy is then given by

$$\mathsf{P}_B(T, E) \propto n(E) W_B(E), \qquad (8)$$

where $n(E)$ is the number of states with energy E. Since the number of states $n(E)$ is an increasing function of energy and the Boltzmann factor $W_B(E)$ decreases exponentially with E, the probability distribution $\mathsf{P}_B(T, E)$ has a bell-like shape in general. When the temperature is high, β is small, and $W_B(E)$ decreases slowly with E. So, $\mathsf{P}_B(T, E)$ has a wide bell-shape. On the other hand, at low temperature β is large, and $W_B(E)$ decreases rapidly with E. So, $\mathsf{P}_B(T, E)$ has a narrow bell-shape (and in the limit $T \to 0$ K, $\mathsf{P}_B(T, E) \propto \delta(E - E_{\mathrm{GS}})$, where E_{GS} is the global-minimum energy). However, it is very difficult to obtain canonical distributions at low temperatures with conventional simulation methods. This is because the thermal fluctuations at low temperatures are small and the simulation will certainly get trapped in states of energy local minima. Simulated annealing [27] is based on the process of crystal making. Namely, by starting a simulation at a sufficiently high temperature (much above the melting temperature), one lowers the temperature gradually during the simulation until it reaches the global-minimum-energy state (crystal). If the rate of temperature decrease is sufficiently slow so that thermal equilibrium may be maintained throughout the simulation, only the state with the global energy minimum is obtained (when the final temperature is 0 K). However, if the temperature decrease is rapid (quenching), the simulation will get trapped in a state of energy local minimum in the vicinity of the initial state.

Simulated annealing was first successfully used to predict the global-minimum-energy conformations of polypeptides and proteins [22,61,63] and to refine protein structures from X-ray and NMR data [5,42] almost a decade ago. Since then this method has been extensively used in the protein folding and structure refinement problems (for reviews, see [45,62]). Our group has been testing the effectiveness of the method mainly in oligopeptide systems. The procedure of our approach is as follows. While the initial conformations in the protein simulations are usually taken from the structures inferred by the experiments, our initial conformations are *randomly generated*. Each Monte-Carlo sweep updates every dihedral angle (in both the main chain and side chains) once. Our annealing schedule is as follows: The temperature is lowered exponentially from T_I = 1000 K to T_F = 250 K (the final temperature T_F was sometimes set equal to 100 K, 50 K, or 1 K) [23,46]. The temperature for the nth MC sweep is given by

$$T_n = T_I \gamma^{n-1}, \tag{9}$$

where γ is a constant which is determined by T_I, T_F, and the total number of MC sweeps of the run. Each run consists of $10^4 \sim 10^6$ MC sweeps, and we usually made 10 to 20 runs from different initial conformations.

Results

We now present the results of our simulations based on Monte-Carlo simulated annealing. All the simulations were started from randomly-generated conformations.

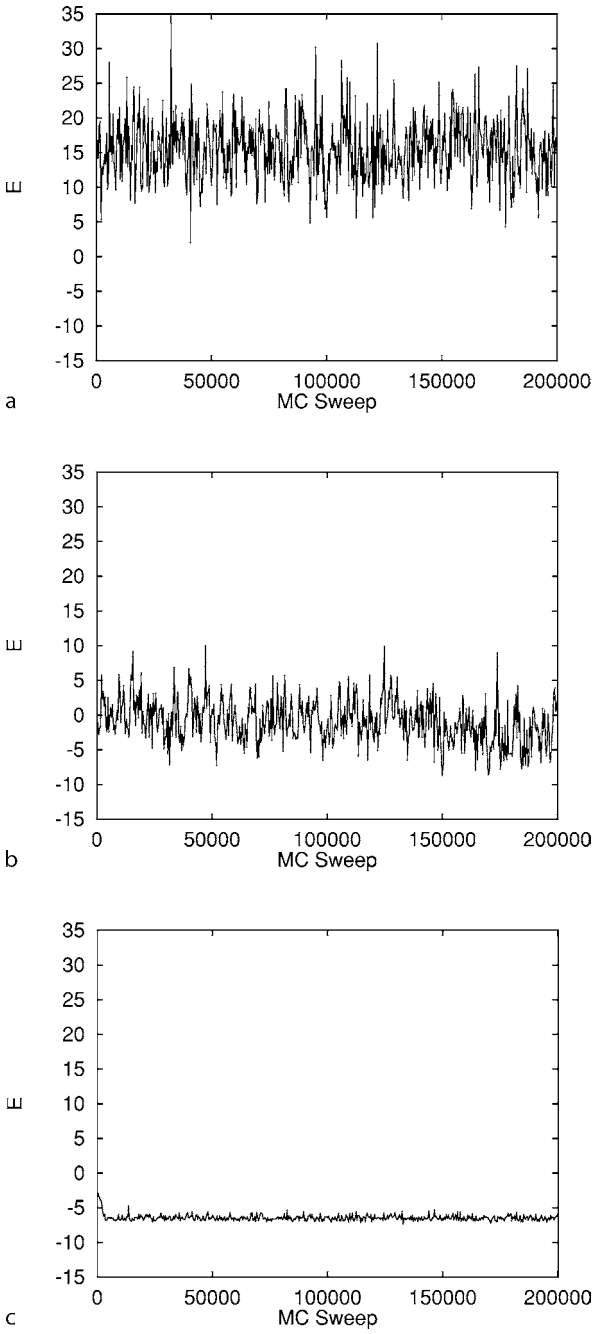
The first example is Met-enkephalin. This brain neuro peptide consists of 5 amino acids with the amino-acid sequence: Tyr-Gly-Gly-Phe-Met. Because it is one of the smallest peptides that have biological functions, it has served as a bench mark for testing a new simulation method. The global minimum conformation of this peptide for ECEPP/2 energy function in gas phase (ϵ = 2) is known [31,49]. For KONF90 realization of ECEPP/2 energy, the peptide is essentially in the ground state for $E_{\mathrm{P}} \leq -11$ kcal/mol [15,49] and the lowest value is −12.2 kcal/mol [16,17].

In Fig. 1, we show the 'time series' of the total conformational energy E_{P} (in (1)) obtained by conventional canonical Monte-Carlo simulations at T = 1000, 300, and 50 K.

The thermal fluctuations for the run at T = 50 K in Fig. 1c are very small and this run has apparently gotten trapped in states of energy local minima (because the average energy at 50 K is about −11 kcal/mol [15,16]). In Fig. 2 we display the time series of energy obtained by a Monte-Carlo simulated annealing simulation.

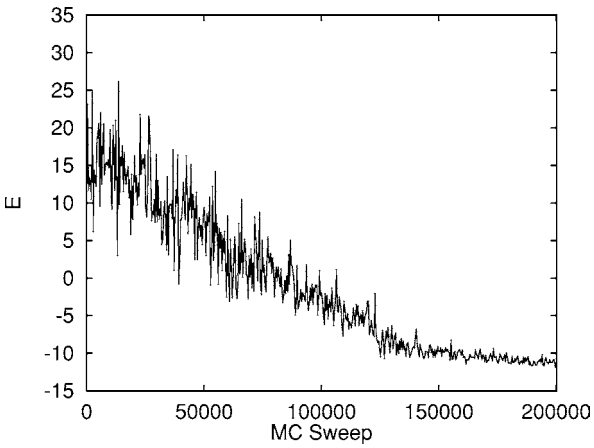
This run reaches the global minimum region ($E_{\mathrm{P}} \leq -11$ kcal/mol) as the temperature is decreased during the simulation from 1000 K to 50 K.

We have up to now presented the results in gas phase (ϵ = 2). In Fig. 3 we compare the superposed structures of lowest-energy conformations from 8 Monte-Carlo simulated annealing runs in gas phase, simple-repulsive solvent, and water (the latter two contributions were calculated by the RISM theory) [26] with those of 5 structures inferred from NMR experiments ([13, Fig. 2]). The figures were created with RasMol [55].



Monte-Carlo Simulated Annealing in Protein Folding, Figure 1

Series of energy E_P (kcal/mol) of Met-enkephalin from conventional canonical Monte-Carlo runs at T = 1000 K (a), 300 K (b), and 50 K (c)



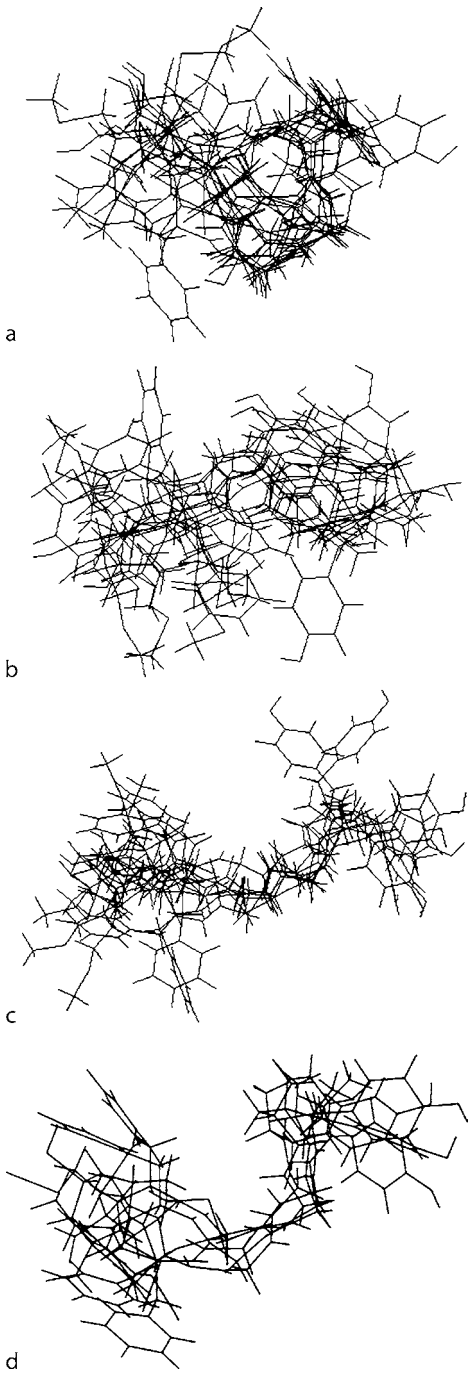
Monte-Carlo Simulated Annealing in Protein Folding, Figure 2

Time series of energy E_P (kcal/mol) of Met-enkephalin from a Monte-Carlo simulated annealing run

We see a striking similarity between simulation results in water Fig. 3c and those of NMR experiments (Fig. 3d). The simulation results in Fig. 3 are from the same number of MC sweeps. It seems that the presence of water speeds up the convergence of the backbone structures in the sense that it requires less number of MC sweeps for convergence [26].

The solvation free energy based on the RISM theory is very accurate, but it is also computationally very demanding. We are currently trying to solve this problem making the algorithm more efficient and robust [24]. Hereafter, we discuss how well other solvation theories can still describe the effects of solvent in the prediction of three-dimensional structures of oligopeptides and small proteins.

Next systems we discuss are those of homo-oligomers with length of 10 amino acids. From the structural data base of X-ray experiments of protein structures [8] and CD experiments [6], it is known that certain amino acids have more tendency of α-helix formation than others. For instance, alanine is a helix former and glycine is a helix breaker, while phenylalanine has intermediate helix-forming tendency. We have performed 20 Monte-Carlo simulated annealing runs of 10,000 MC sweeps in gas phase (ϵ = 2) with each of $(Ala)_{10}$, $(Leu)_{10}$, $(Met)_{10}$, $(Phe)_{10}$, $(Ile)_{10}$, $(Val)_{10}$, and $(Gly)_{10}$ [44]. These amino acids are nonpolar and we can avoid the complications of electrostatic and



Monte-Carlo Simulated Annealing in Protein Folding, Figure 3
Superposition of the eight conformations of Met-enkephalin obtained as the lowest-energy structures by Monte-Carlo simulated annealing in gas phase (a), simple-repulsive solvent (b), and water (c) together with superposition of five conformations deduced from the NMR experiment (d)

hydrogen-bond interactions of side chains with each other, with main chain, and with the solvent.

In order to analyze how much α-helix formation is obtained by simulations, we first define α-helix state of a residue. We consider that a residue is in the α-helix state when the dihedral angles (ϕ, ψ) fall in the range $(-60 \pm 45°, -50 \pm 45°)$ (Definition I) [23,46]. The length ℓ of a helical segment is then defined by the number of successive residues that are in the α-helix state. The number n of helical residues in a conformation is defined by the sum of ℓ over all helical segments in the conformation. Note that $\ell = 3$ corresponds to roughly one turn of α-helix. We therefore consider a conformation as helical if it has a segment with helix length $\ell \geq 3$.

The average values of the dihedral angles ϕ and ψ for the helical segments based on Definition I (with helix length $\ell \geq 3$) are $-70°$ and $-37°$, respectively, and the standard deviation is $\sim 10°$ for ECEPP/2 energy function [44,46]. Hence, for detailed analyses of the data we adopt a more stringent criterion for α-helix state (Definition II): The range is $(\phi, \psi) = (-70 \pm 20°, -37 \pm 20°)$ [44].

We likewise consider that a residue is in the β-strand state when the dihedral angles (ϕ, ψ) fall in the range $(-130 \pm 50°, 135 \pm 45°)$ [44]. The β-strand length m is then defined to be the number of successive residues that are in the β-strand state. We consider a conformation as β-stranded if it has a segment with β-strand length $m \geq 3$.

In Table 1 we summarize the α-helix formation in the 20 Monte-Carlo simulated annealing runs [44]. The results are for Definition II of the α-helix state.

We see that $(Met)_{10}$, $(Ala)_{10}$, and $(Leu)_{10}$ gave many helical conformations: 15, 9, and 9 (out of 20), respectively. In particular, $(Met)_{10}$ and $(Ala)_{10}$ produced long helices, some conformations being almost entirely helical ($\ell \geq 8$). On the other hand, $(Val)_{10}$, $(Ile)_{10}$, and $(Gly)_{10}$ gave few helical conformations: 2, 2, and 1 (out of 20), respectively. We obtained not only a smaller number of helices but also shorter helices for these homo-oligomers than the above three homo-oligomers. Finally, the results for $(Phe)_{10}$ indicate that Phe has intermediate helix-forming tendency between these two groups. We thus have the following rank order of helix-forming tendency for the seven amino acids [44]:

$$\text{Met} > \text{Ala} > \text{Leu} > \text{Phe} > \text{Val} > \text{Ile} > \text{Gly}. \quad (10)$$

Monte-Carlo Simulated Annealing in Protein Folding, Table 1
α-Helix formation in homo-oligomers from 20 Monte-Carlo simulated annealing runs

Peptide ℓ	$(Met)_{10}$	$(Ala)_{10}$	$(Leu)_{10}$	$(Phe)_{10}$	$(Val)_{10}$	$(Ile)_{10}$	$(Gly)_{10}$
3	1	0	4	1	0	2	1
4	2	0	2	2	2	0	0
5	0	1	1	1	0	0	0
6	2	3	2	1	0	0	0
7	2	1	0	0	0	0	0
8	7	4	0	0	0	0	0
9	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0
Total	15/20	9/20	9/20	5/20	2/20	2/20	1/20

This can be compared with the experimentally determined helix propensities [6,8]. Our rank order (10) is in good agreement with the experimental data.

We then analyzed the relation between helix-forming tendency and energy. We found that the differences $\Delta E = E_{NH} - E_H$ between minimum energies for nonhelical (NH) and helical (H) conformations is large for homo-oligomers with high helix-forming tendency (9.7, 10.2, 21.5 kcal/mol for $(Met)_{10}$, $(Ala)_{10}$, $(Leu)_{10}$, respectively) and small for those with low helix-forming tendency (0.5, 1.6, −3.2 kcal/mol for $(Val)_{10}$, $(Ile)_{10}$, $(Gly)_{10}$, respectively). Moreover, we found that the large ΔE for the former homo-oligomers are caused by the Lennard–Jones term ΔE_{LJ} (13.3, 8.0, 17.5 kcal/mol for $(Met)_{10}$, $(Ala)_{10}$, $(Leu)_{10}$, respectively). Hence, we conjecture that the differences in helix-forming tendencies are determined by the following factors [44]. A helical conformation is energetically favored in general because of the Lennard–Jones term E_{LJ}. For amino acids with low helix-forming tendency except for Gly, however, the steric hindrance of side chains raises E_{LJ} of helical conformations so that the difference ΔE_{LJ} between nonhelical and helical conformations are reduced significantly. The small ΔE_{LJ} for these amino acids can be easily overcome by the entropic effects and their helix-forming tendencies are small. Note that such amino acids (Val and Ile here) have two large side-chain branches at C^β, while the helix forming amino acids such as Met and Leu have only one branch at C^β and Ala has a small side chain.

We now study the β-strand forming tendencies of these seven homo-oligomers. In Table 2 we summarize the β-strand formation in 20 Monte-Carlo simulated annealing runs [44].

The implications of the results are not as obvious as in the α-helix case. This is presumably because a short, isolated β-strand is not very stable by itself, since hydrogen bonds between β-strands are needed to stabilize them. However, we can still give a rough estimate for the rank order of strand-forming tendency for the seven amino acids [44]:

$$\text{Val} > \text{Ile} > \text{Phe} > \text{Leu} > \text{Ala} > \text{Met} > \text{Gly}. \quad (11)$$

Here, we considered Val as more strand-forming than Ile, since the longer the strand segment is, the harder it is to form by simulation. Our rank order (11) is again in good agreement with the experimental data [8].

By comparing (11) with (10), we find that the helix-forming group is the strand-breaking group and vice versa, except for Gly. Gly is both helix and strand breaking. This reflects the fact that Gly, having no side chain, has a much larger (backbone) conformational space than other amino acids.

The helix-coil transitions of homo-oligomer systems were further analyzed by multicanonical algorithms [3] in [47,48]. The obtained results gave quantitative support to those by Monte-Carlo simulated annealing described above [44].

We have so far studied peptides with nonpolar amino acids each of which is electrically neutral as a whole. We now discuss the helix-forming tendencies of peptides with polar amino acids where side chains are charged by protonation or deprotonation. One example is the C-peptide, residues 1–13 of ribonuclease A.

Monte-Carlo Simulated Annealing in Protein Folding, Table 2
β-Strand formation in homo-oligomers from 20 Monte-Carlo simulated annealing runs

Peptide	$(Met)_{10}$	$(Ala)_{10}$	$(Leu)_{10}$	$(Phe)_{10}$	$(Val)_{10}$	$(Ile)_{10}$	$(Gly)_{10}$
m							
3	0	0	2	5	1	7	0
4	0	0	0	1	0	4	0
5	0	0	0	0	2	1	0
6	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
Total	0/20	0/20	2/20	6/20	5/20	12/20	0/20

It is known from the X-ray diffraction data of the whole enzyme that the segment from Ala-4 to Gln-11 exhibits a nearly 3-turn α-helix [58,64]. It was also found by CD [56] and NMR [53] experiments that the isolated C-peptide also has significant α-helix formation in aqueous solution at temperatures near 0°C.

Furthermore, the CD experiment of the isolated C-peptide showed that the side-chain charges of residues Glu-2^- and His-12^+ enhance the stability of the α-helix, while the rest of the charges of other side chains do not [56]. The NMR experiment [53] of the isolated C-peptide further observed the formation of the characteristic salt bridge between Glu-2^- and Arg-10^+ that exists in the native structure determined by the X-ray experiments of the whole protein [58,64].

In order to test whether our simulations can reproduce these experimental results, we made 20 Monte-Carlo simulated annealing runs of 10,000 MC sweeps with several C-peptide analogues [23,46]. The amino-acid sequences of four of the analogues are listed in Table 3.

Monte-Carlo Simulated Annealing in Protein Folding, Table 3
Amino-acid sequences of the peptide analogues of C-peptide studied by Monte-Carlo simulated annealing

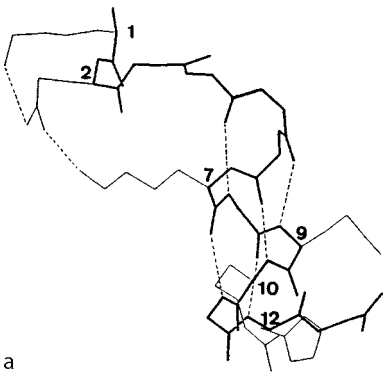
Peptide	I	II	III	IV
Sequence				
1	Lys^+			
2	Glu^-			Glu
3	Thr			
4	Ala			
5	Ala			
6	Ala			
7	Lys^+			
8	Phe			
9	Glu^-	Glu	Leu	
10	Arg^+			
11	Gln			
12	His^+			His
13	Met			

The simulations were performed in gas phase (ϵ = 2). The temperature was decreased exponentially from 1000 K to 250 K for each run. As usual, all the simulations were started from random conformations.

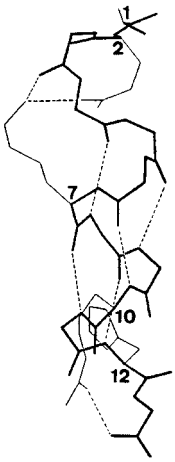
In Table 4 we summarize the helix formation of all the runs [46]. Here, the number of conformations with segments of helix length $\ell \geq 3$ are given with Definition I of the α-helix state. From this table one sees that α-helix was hardly formed for Peptide IV where Glu-2 and His-12 are neutral, while many helical conformations were obtained for the other peptides. This is in accord with the experimental results that the charges of Glu-2^- and His-12^+ are necessary for the α-helix stability [56].

Peptides II and III had conformations with the longest α-helix (ℓ = 7). These conformations turned out to have the lowest energy in 20 simulation runs for each peptide. They both exhibit an α-helix from Ala-5 to Gln-11, while the structure from the X-ray data has an α-helix from Ala-4 to Gln-11. These three conformations are compared in Fig. 4.

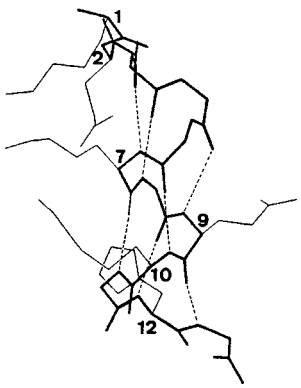
As mentioned above, the agreement of the backbone structures is conspicuous, but the side-chain



a



b



c

Monte-Carlo Simulated Annealing in Protein Folding, Figure 4

The lowest-energy conformations of Peptide II (a) and Peptide III (b) of C-peptide analogues obtained from 20 Monte-Carlo simulated annealing runs in gas phase, and the corresponding *X*-ray structure (c)

Monte-Carlo Simulated Annealing in Protein Folding, Table 4
α-Helix formation in C-peptide analogues from 20 Monte-Carlo simulated annealing runs

Peptide	I	II	III	IV
ℓ				
3	4	2	3	1
4	3	2	3	0
5	1	1	0	0
6	0	1	0	0
7	0	1	1	0
Total	8/20	7/20	7/20	1/20

structures are not quite similar. In particular, while the *X*-ray [58,64] and NMR [53] experiments imply the formation of the salt bridge between the side chains of Glu-2^- and Arg-10^+, the lowest-energy conformations of Peptides II and III obtained from the simulations do not have this salt bridge.

The disagreement is presumably caused by the lack of solvent in our simulations. We have therefore made multicanonical Monte-Carlo simulations of Peptide II with the inclusion of solvent effects by the distance-dependent dielectric function (see (2)) [18,19]. It was found that the lowest-energy conformation obtained has an α-helix from Ala-4 to Gln-11 and does have the characteristic salt bridge between Glu-2^- and Arg-10^+ [18,19].

Similar dependence of α-helix stability on side-chain charges was observed in Monte-Carlo simulated annealing runs of a 17-residue synthetic peptide [43]. The pH difference in the experimental conditions was represented by the corresponding difference in charge assignment of the side chains, and the agreement with the experimental results (stable α-helix formation at low pH and low helix content at high pH) was observed in the simulations by Monte-Carlo simulated annealing with the distance-dependent dielectric function [43].

Considering our simulation results on homo-oligomers of nonpolar amino acids, C-peptide, and the synthetic peptide, we conjecture that the helix-forming tendencies of oligopeptide systems are controlled by the following factors [43]. An α-helix structure is generally favored energetically (especially, the Lennard–Jones term). When side chains are uncharged, the steric hindrance of side chains is the key factor for the difference in helix-forming tendency. When some of the

side chains are charged, however, these charges play an important role in the helix stability in addition to the above factor: Some charges enhance helix stability, while others reduce it.

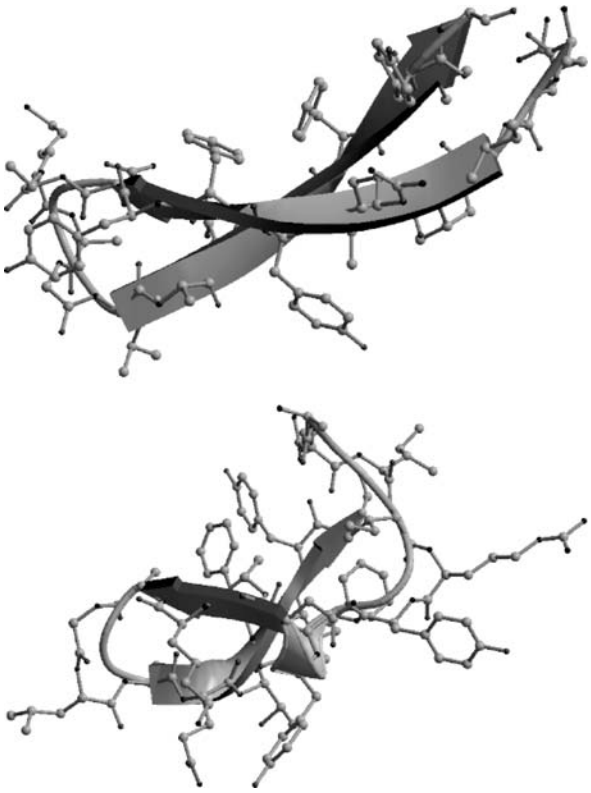
We have up to now discussed α-helix formations in our simulations of oligopeptide systems. We have also studied β-sheet formations by Monte-Carlo simulated annealing [38,39,51]. The peptide that we studied is the fragment corresponding to residues 16–36 of bovine pancreatic trypsin inhibitor (BPTI) and has the amino-acid sequence: Ala^{16}-Arg^{+}-Ile-Ile-Arg^{+}-Tyr-Phe-Tyr-Asn-Ala-Lys^{+}-Ala-Gly-Leu-Cys-Gln-Thr-Phe-Val-Tyr-Gly^{36}. An antiparallel β-sheet structure in residues 18–35 is observed in *X*-ray crystallographic data of the whole protein [10].

We first performed 20 Monte-Carlo simulated annealing runs of 10,000 MC sweeps in gas phase ($\epsilon = 2$) with the same protocol as in the previous simulations [38]. Namely, the temperature was decreased exponentially from 1000 K to 250 K for each run, and all the simulations were started from random conformations. The difference of the present simulation and the previous ones comes only from that of the amino-acid sequences.

The most notable feature of the obtained results is that α-helices, which were the dominant motif in previous simulations of C-peptide and other peptides, are absent in the present simulation. Most of the conformations obtained consist of stretched strands and a 'turn' which connects them. The lowest-energy structure indeed exhibits an antiparallel β-sheet [38].

We next made 10 Monte-Carlo simulated annealing runs of 100,000 MC sweeps for BPTI(16–36) with two dielectric functions: $\epsilon = 2$ and the sigmoidal, distance-dependent dielectric function of (2) [39]. The results with $\epsilon = 2$ reproduced our previous results: Most of the obtained conformations have β-strand structures and no extended α-helix is observed. Those with the sigmoidal dielectric function, on the other hand, indicated formation of α-helices. One of the low-energy conformations, for instance, exhibited about a four-turn α-helix from Ala-16 to Gly-28 [39]. This presents an example in which a peptide with the same amino-acid sequence can form both α-helix and β-sheet structures, depending on its electrostatic environment.

NMR experiments suggest that this peptide actually forms a β-sheet structure [40]. The representation of solvent by the sigmoidal dielectric function (which gave α-helices instead) is therefore not sufficient. Hence, the same peptide fragment, BPTI(16–36), was further studied in aqueous solution that is represented by solvent-accessible surface area of (3) by Monte-Carlo simulated annealing [51]. Twenty simulation runs of 100,000 MC sweeps were made. It was indeed found that the lowest-energy structure obtained has a β-sheet structure (actually, type II′ β-turn) at the very location suggested by the NMR experiments [40]. This structure and that deduced from the *X*-ray experiments [10] are compared in Fig. 5. The figures were created with Molscript [29] and Raster3D [2,35].



Monte-Carlo Simulated Annealing in Protein Folding, Figure 5

The structure of BPTI(16–36) deduced from X-ray experiments (a) and the lowest-energy conformation of BPTI(16–36) obtained from 20 Monte-Carlo simulated annealing runs in aqueous solution represented by solvent-accessible surface area (b)

Although both conformations are β-sheet structures, there are important differences between the two: The positions and types of the turns are different. Since

the X-ray structure is taken from the experiments on the whole BPTI molecule, it does not have to agree with that of the isolated BPTI(16–36) fragment. It was found [51] that the simulated results in Fig. 5b have remarkable agreement with those in the NMR experiments of the isolated fragment [40].

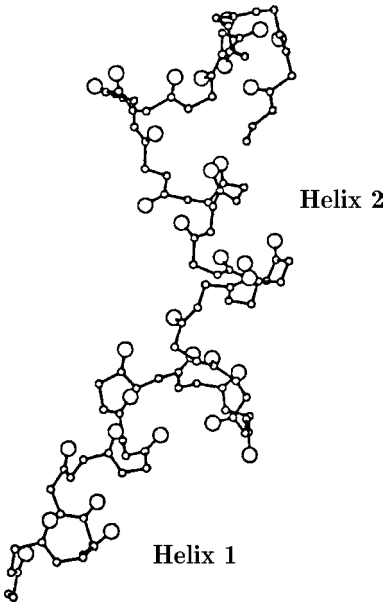
We have so far dealt with peptides with small number of amino acids (up to 21) with simple secondary structural elements: a single α-helix or β-sheet. The native proteins usually have more than one secondary structural elements. We now discuss our attempts on the first-principles tertiary structure predictions of larger and more complicated systems.

The first example is the fragment corresponding to residues 1–34 of human parathyroid hormone (PTH). An NMR experiment of PTH(1–34) suggested the existence of two α-helices around residues from Ser-3 to His-9 and from Ser-17 to Leu-28 [28]. Another NMR experiment of a slightly longer fragment, PTH(1–37), in aqueous solution also suggested the existence of the two helices [32]. One of the determined structures, for instance, has α-helices in residues from Gln-6 to His-9 and from Ser-17 to Lys-27 [32].

For PTH(1–34) we performed 20 Monte-Carlo simulated annealing runs of 10,000 MC sweeps in gas phase ($\epsilon = 2$) with the same protocol as in the previous simulations [50]. Many conformations among the 20 final conformations obtained exhibited α-helix structures (especially in the N-terminus area). In Fig. 6 we show the lowest-energy conformation of PTH(1–34) [50].

This conformation indeed has two α-helices around residues from Val-2 to Asn-10 (Helix 1) and from Met-18 to Glu-22 (Helix 2), which are precisely the same locations as suggested by experiment [28], although Helix 2 is somewhat shorter (5 residues long) than the corresponding one (12 residues long) in the experimental data.

A slightly larger peptide fragment, PTH(1–37), was also studied by Monte-Carlo simulated annealing [34] to compare with the results of the recent NMR experiment in aqueous solution [32]. Ten simulation runs of 100,000 MC sweeps were made in gas phase ($\epsilon = 2$) and in aqueous solution that is represented by the terms proportional to the solvent-accessible surface area (see (3)). Although the results are preliminary, the simulations in gas phase did not produce two helices this time in contrast to the previous work [50], where a short

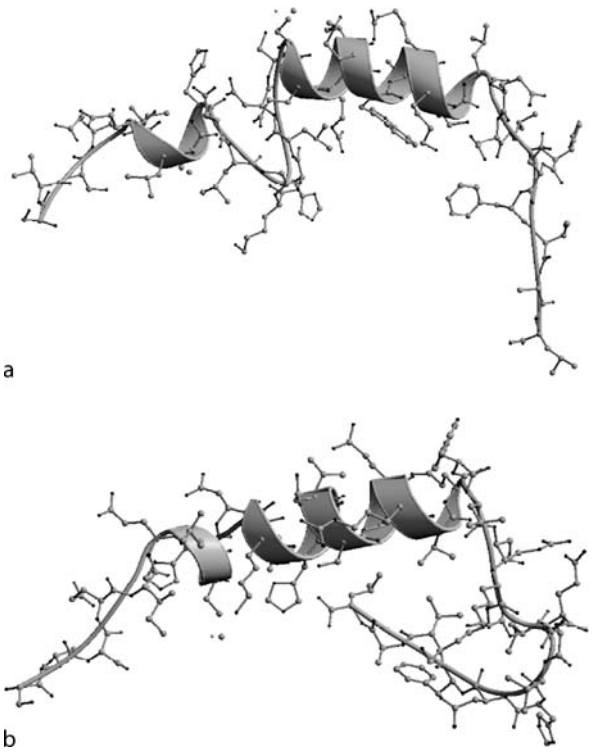


Monte-Carlo Simulated Annealing in Protein Folding, Figure 6
Lowest-energy conformation of PTH(1–34) obtained from 20 Monte-Carlo simulated annealing runs in gas phase

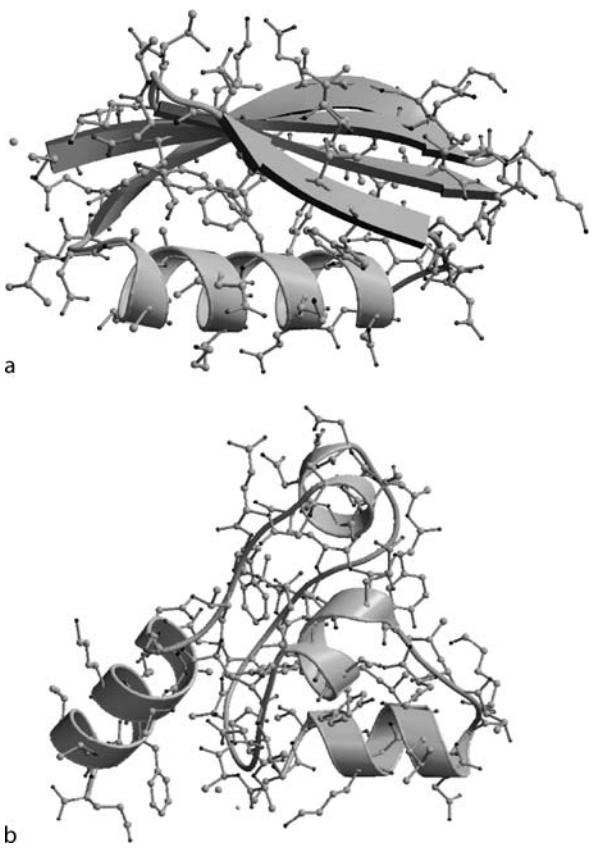
second helix was observed, as discussed in the previous paragraph. The lowest-energy conformation has an α-helix from Val-2 to Asn-10. The simulations in aqueous solution, on the other hand, did observe the two α-helices. The lowest-energy conformation obtained has α-helices from Gln-6 to His-9 and from Gly-12 to Glu-22. Note that the second helix is now more extended than the first one in agreement with experiments. This structure together with one of the NMR structure [32] is shown in Fig. 7. The figures were again created with Molscript [29] and Raster3D [2,35].

Generalized-ensemble simulations of PTH(1–37) are now in progress in order to obtain more quantitative information such as average helicity as a function of residue number, etc.

The second example of more complicated system is the immunoglobulin-binding domain of streptococcal protein G. This protein is composed of 56 amino acids and the structure determined by an NMR experiment [14] and an X-ray diffraction experiment [1] has an α-helix and a β-sheet. The α-helix extends from residue Ala-23 to residue Asp-36. The β-sheet is made of four β-strands: from Met-1 to Gly-9, from Leu-12 to Ala-20, from Glu-42 to Asp-46, and from Lys-50 to Glu-56.



Monte-Carlo Simulated Annealing in Protein Folding, Figure 7
A structure of PTH(1–37) deduced from NMR experiments (a) and the lowest-energy conformation of PTH(1–37) obtained from 10 Monte-Carlo simulated annealing runs in aqueous solution represented by solvent-accessible surface area (b)



Monte-Carlo Simulated Annealing in Protein Folding, Figure 8
A structure of protein G deduced from an X-ray experiment (a) and the lowest-energy conformation of protein G obtained from Monte-Carlo simulated annealing runs with the distance-dependent dielectric function (b)

This structure is shown in Fig. 8a). The figures in Fig. 8 were again created with Molscript [29] and Raster3D [2,35].

We have performed eight Monte-Carlo simulated annealing runs of 50,000 to 400,000 MC sweeps with the sigmoidal, distance-dependent dielectric function of (2). The lowest-energy conformation so far obtained has four α-helices and no β-sheet in disagreement with the *X*-ray structure. This structure is shown in Fig. 8b).

The disagreement of the lowest-energy structure (Fig. 8b) so far obtained with the *X*-ray structure (Fig. 8a) is presumably caused by the poor representation of the solvent effects. As can been seen in Fig. 8a), the *X*-ray structure has both interior where a well-defined hydrophobic core is formed and exterior where it is exposed to the solvent. The distance-dependent dielectric function, which mimics the solvent effects only in electrostatic interactions, is therefore not sufficient to represent the effects of the solvent here.

Conclusions

In this article we have reviewed theoretical aspects of the protein folding problem. Our strategy in tackling this problem consists of two elements: 1) inclusion of accurate solvent effects, and 2) development of powerful simulation algorithms that can avoid getting trapped in states of energy local minima.

We have shown the effectiveness of Monte-Carlo simulated annealing by showing that direct folding of α-helix and β-sheet structures from randomly-generated initial conformations are possible.

As for the solvent effects, we considered several methods: a distance-dependent dielectric function, a term proportional to solvent-accessible surface area, and the reference interaction site model (RISM). These methods vary in nature from crude but computationally inexpensive (distance-dependent dielectric function) to accurate but computationally demanding (RISM theory). In the present article, we have shown that the inclusion of some solvent effects is very important for a successful prediction of the tertiary structures of small peptides and proteins.

See also

- ▶ Adaptive Simulated Annealing and its Application to Protein Folding
- ▶ Bayesian Global Optimization
- ▶ Genetic Algorithms
- ▶ Genetic Algorithms for Protein Structure Prediction
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Lennard–Jones and Morse Clusters
- ▶ Global Optimization in Protein Folding
- ▶ Molecular Structure Determination: Convex Global Underestimation
- ▶ Monte-Carlo Simulations for Stochastic Optimization
- ▶ Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach
- ▶ Packet Annealing
- ▶ Phase Problem in X-ray Crystallography: Shake and Bake Approach
- ▶ Protein Folding: Generalized-ensemble Algorithms
- ▶ Random Search Methods
- ▶ Simulated Annealing
- ▶ Simulated Annealing Methods in Protein Folding
- ▶ Stochastic Global Optimization: Stopping Rules
- ▶ Stochastic Global Optimization: Two-phase Methods

References

1. Achari A, Hale SP, Howard AJ, Clore GM, Gronenborn AM, Hardman KD, Whitlow M (1992) 1.67- Å X-ray structure of the B2 immunoglobulin-binding domain of streptococcal protein G and comparison to the NMR structure of the B1 domain. Biochemistry 31:10449–10457
2. Bacon D, Anderson WF (1988) A fast algorithm for rendering space-filling molecular pictures. J Mol Graphics 6:219–220
3. Berg BA, Neuhaus T (1991) Multicanonical algorithms for first order phase transitions. Phys Lett B267:249–253
4. Brooks III CL (1998) Simulations of protein folding and unfolding. Curr Opin Struct Biol 8:222–226
5. Brünger AT (1988) Crystallographic refinement by simulated annealing: Application to a 2.8 Å resolution structure of aspartate aminotransferase. J Mol Biol 203:803–816
6. Chakrabartty A, Kortemme T, Baldwin RL (1994) Helix propensities of the amino acids measured in alanine-based peptides without helix-stabilizing side-chain interactions. Protein Sci 3:843–852
7. Chandler D, Andersen HC (1972) Optimized cluster expansions for classical fluids. Theory of molecular liquids. J Chem Phys 57:1930–1937
8. Chou PY, Fasman GD (1974) Prediction of protein conformation. Biochemistry 13:222–245
9. Daggett V, Kollman PA, Kuntz ID (1991) Molecular dynamics simulations of small peptides: dependence on dielectric model and pH. Biopolymers 31:285–304
10. Deisenhofer J, Steigemann W (1975) Crystallographic refinement of the structure of bovine pancreatic trypsin inhibitor at 1.5 Å resolution. Acta Crystallogr B31:238–250
11. Dill K (1990) The meaning of hydrophobicity. Science 250:297–297
12. Epstain CJ, Goldberger RF, Anfinsen CB (1963) The genetic control of tertiary protein structure: studies with model systems. Cold Spring Harbor Symp Quant Biol 28: 439–449
13. Graham WH, Carter ES, II, Hicks RP (1992) Conformational analysis of Met-enkephalin in both aqueous solution and in the presence of sodium dodecyl sulfate micelles using multidimensional NMR and molecular modeling. Biopolymers 32:1755–1764
14. Gronenborn AM, Filpula DR, Essig NZ, Achari A, Whitlow M, Wingfield PT, Clore GM (1991) A novel, highly stable fold of the immunoglobulin binding domain of streptococcal protein G. Science 253:657–661
15. Hansmann UHE, Okamoto Y (1993) Prediction of peptide conformation by multicanonical algorithm: new approach to the multiple-minima problem. J Comput Chem 14:1333–1338
16. Hansmann UHE, Okamoto Y (1994) Comparative study of multicanonical and simulated annealing algorithms in the protein folding problem. Phys A 212:415–437
17. Hansmann UHE, Okamoto Y (1994) Sampling ground-state configurations of a peptide by multicanonical annealing. J Phys Soc Japan 63:3945–3949
18. Hansmann UHE, Okamoto Y (1998) Tertiary structure prediction of C-peptide of ribonuclease A by multicanonical algorithm. J Phys Chem B 102:653–656
19. Hansmann UHE, Okamoto Y (1999) Effects of side-chain charges on α-helix stability in C-peptide of ribonuclease

A studied by multicanonical algorithm. J Phys Chem B 103:1595–1604
20. Hingerty BE, Ritchie RH, Ferrell T, Turner JE (1985) Dielectric effects in biopolymers: the theory of ionic saturation revisited. Biopolymers 24:427–439
21. Hirata F, Rossky PJ (1981) An extended RISM equation for molecular polar fluids. Chem Phys Lett 83:329–334
22. Kawai H, Kikuchi T, Okamoto Y (1989) A prediction of tertiary structures of peptide by the Monte Carlo simulated annealing method. Protein Eng 3:85–94
23. Kawai H, Okamoto Y, Fukugita M, Nakazawa T, Kikuchi T (1991) Prediction of α-helix folding of isolated C-peptide of ribonuclease A by Monte Carlo simulated annealing. Chem Lett:213–216
24. Kinoshita M, Okamoto Y, Hirata F (1997) Calculation of hydration free energy for a solute with many atomic sites using the RISM theory: robust and efficient algorithm. J Comput Chem 18:1320–1326
25. Kinoshita M, Okamoto Y, Hirata F (1997) Solvation structure and stability of peptides in aqueous solutions analyzed by the reference interaction site model theory. J Chem Phys 107:1586–1599
26. Kinoshita M, Okamoto Y, Hirata F (1998) First-principle determination of peptide conformations in solvents: combination of Monte Carlo simulated annealing and RISM theory. J Amer Chem Soc 120:1855–1863
27. Kirkpatrick S, Gelatt, CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
28. Klaus W, Dieckmann T, Wray V, Schomburg D, Wingender E, Mayer H (1991) Investigation of the solution structure of the human parathyroid hormone fragment (1–34) by ^{1}H NMR spectroscopy, distance geometry, and molecular dynamics calculations. Biochemistry 30:6936–6942
29. Kraulis PJ (1991) MOLSCRIPT: A program to produce both detailed and schematic plots of protein structures. J Appl Crystallogr 24:946–950
30. Levinthal C (1968) Are there pathways for protein folding? J Chem Phys 65:44–45
31. Li Z, Scheraga HA (1987) Monte Carlo-minimzation approach to the multiple-minima problem in protein folding. Proc Natl Acad Sci USA 84:6611–6615
32. Marx UT, Austermann S, Bayer P, Adermann K, Ejchart A, Sticht H, Walter S, Schmid F-X, Jaenicke R, Forssmann W-G, Rösch P (1995) Structure of human parathyroid hormone 1–37 in solution. J Biol Chem 270:15194–15202
33. Masuya M, Okamoto Y, in preparation
34. Masuya M, Okamoto Y, in preparation
35. Merritt EA, Murphy MEP (1994) Raster3D version 2.0. A program for photorealistic molecular graphics. Acta Crystallogr D50:869–873
36. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:1087–1092
37. Momany FA, McGuire RF, Burgess AW, Scheraga HA (1975) Energy parameters in polypeptides. VII. Geometric parameters, partial atomic charges, nonbonded interactions, hydrogen bond interactions, and intrinsic torsional potentials for the naturally occurring amino acids. J Phys Chem 79:2361–2381
38. Nakazawa T, Kawai H, Okamoto Y, Fukugita M (1992) β-sheet folding of bovine pancreatic trypsin inhibitor fragment (16–36) as predicted by Monte Carlo simulated annealing. Protein Eng 5:495–503
39. Nakazawa T, Okamoto Y (1999) Electrostatic effects on the α-helix and β-strand folding of BPTI(16–36) as predicted by Monte Carlo simulated annealing. J Peptide Res 54:230–236
40. Nakazawa T, Okamoto Y, Kobayashi Y, Kyogoku Y, Aimoto S, in preparation
41. Némethy G, Pottle MS, Scheraga HA (1983) Energy parameters in polypeptides. 9. Updating of geometrical parameters, nonbonded interactions, and hydrogen bond interactions for the naturally occurring amino acids. J Phys Chem 87:1883–1887
42. Nilges M, Clore GM, Gronenborn AM (1988) Determination of three-dimensional structures of proteins from interproton distance data by hybrid distance geometry-dynamical simulated annealing calculations. FEBS Lett 229:317–324
43. Okamoto Y (1994) Dependence on the dielectric model and pH in a synthetic helical peptide studied by Monte Carlo simulated annealing. Biopolymers 34:529–539
44. Okamoto Y (1994) Helix-forming tendencies of nonpolar amino acids predicted by Monte Carlo simulated annealing. PROTEINS: Struct Funct Genet 19:14–23
45. Okamoto Y (1998) Protein folding problem as studied by new simulation algorithms. Recent Res Developm Pure Appl Chem 2:1–23
46. Okamoto Y, Fukugita M, Nakazawa T, Kawai H (1991) α-helix folding by Monte Carlo simulated annealing in isolated C-peptide of ribonuclease A. Protein Eng 4:639–647
47. Okamoto Y, Hansmann UHE (1995) Thermodynamics of helix-coil transitions studied by multicanonical algorithms. J Phys Chem 99:11276–11287
48. Okamoto Y, Hansmann UHE, Nakazawa T (1995) α-Helix propensities of amino acids studied by multicanonical algorithm. Chem Lett 391–392
49. Okamoto Y, Kikuchi T, Kawai H (1992) Prediction of low-energy structures of Met-enkephalin by Monte Carlo simulated annealing. Chem Lett 1275–1278
50. Okamoto Y, Kikuchi T, Nakazawa T, Kawai H (1993) α-Helix structure of parathyroid hormone fragment (1–34) predicted by Monte Carlo simulated annealing. Internat J Peptide Protein Res 42:300–303
51. Okamoto Y, Masuya M, Nabeshima M, Nakazawa T (1999) β-Sheet formation in BPTI(16–36) by Monte Carlo simulated annealing. Chem Phys Lett 299:17–24
52. Ooi T, Oobatake M, Némethy G, Scheraga HA (1987) Accessible surface areas as a measure of the thermodynamic parameters of hydration of peptides. Proc Natl Acad Sci USA 84:3086–3090

53. Osterhout JJ, Baldwin RL, York EJ, Stewart JM, Dyson HJ, Wright PE (1989) ^{1}H NMR studies of the solution conformations of an analogue of the C-peptide of ribonuclease A. Biochemistry 28:7059–7064
54. Ramstein J, Lavery R (1988) Energetic coupling between DNA bending and base pair opening. Proc Natl Acad Sci USA 85:7231–7235
55. Sayle RA, Milner-White EJ (1995) RasMol: biomolecular graphics for all. TIBS 20:374–376
56. Shoemaker KR, Kim PS, Brems DN, Marqusee S, York EJ, Chaiken IM, Stewart JM, Baldwin RL (1985) Nature of the charged-group effect on the stability of the C-peptide helix. Proc Natl Acad Sci USA 82:2349–2353
57. Sippl MJ, Némethy G, Scheraga HA (1984) Intermolecular potentials from crystal data. 6. Determination of empirical potentials for O-H $\cdots$ O = C hydrogen bonds from packing configurations. J Phys Chem 88:6231–6233
58. Tilton RF Jr, Dewan JC, Petsko GA (1992) Effects of temperature on protein structure and dynamics: X-ray crystallographic studies of the protein ribonuclease-A at nine different temperatures from 98 to 320 K. Biochemistry 31:2469–2481
59. Wesson L, Eisenberg D (1992) Atomic solvation parameters applied to molecular dynamics of proteins in solution. Protein Sci 1:227–235
60. Wetlaufer DB (1973) Nucleation, rapid folding, and globular intrachain regions in proteins. Proc Natl Acad Sci USA 70:697–701
61. Wilson C, Doniach S (1989) A computer model to dynamically simulate protein folding: studies with crambin. PROTEINS: Struct Funct Genet 6:193–209
62. Wilson SR, Cui W (1994) Conformation searching using simulated annealing. In: The Protein Folding Problem and Tertiary Structure Prediction. In: Lecture Notes. Birkhäuser, Basel, pp 43–70
63. Wilson SR, Cui W, Moskowitz JW, Schmidt KE (1988) Conformational analysis of flexible molecules: location of the global minimum energy conformation by the simulated annealing method. Tetrahedron Lett 29:4373–4376
64. Wychoff HW, Tsernoglou D, Hanson AW, Knox JR, Lee B, Richards FM (1970) The three-dimensional structure of ribonuclease-S. J Biol Chem 245:305–328

Monte-Carlo Simulations for Stochastic Optimization

DAVID P. MORTON, ELMIRA POPOVA
Oper. Res. and Industrial Engineering,
University Texas at Austin, Austin, USA

MSC2000: 90C15, 65C05, 65K05, 90C31, 62F12

Article Outline

Keywords
Solution Procedures
Establishing Solution Quality
Variance Reduction Techniques
Theoretical Justification for Sampling
See also
References

Keywords

Stochastic programming; Simulation-based optimization; Monte-Carlo method

Many important real-world problems contain stochastic elements and require optimization. *Stochastic programming* and *simulation-based optimization* are two approaches used to address this issue. We do not explicitly discuss other related areas including stochastic control, stochastic dynamic programming, and Markov decision processes. We consider a stochastic optimization problem of the form

$$\text{(SP)} \quad z^* = \min_{x \in X} \mathsf{E} f(x, \xi),$$

where x is a vector of decision variables with deterministic feasible region $X \subset \mathbf{R}^d$, ξ is a random vector, and f is a real-valued function with finite expectation, $\mathsf{E} f(x, \xi)$, for all $x \in X$. We use x^* to denote an optimal solution to (SP). Note that the decision x must be made prior to observing the realization of ξ.

A wide variety of types of problems can be expressed as (SP) depending on the definitions of f and X. Two of the most commonly-used approaches are rooted in mathematical programming and in discrete-event simulation modeling.

In a two-stage stochastic linear program with recourse [6,14], X is a polyhedral set and f is defined as the optimal value of a linear program, given x and ξ, i. e.,

$$f(x, \xi) = cx + \begin{cases} \min\limits_{y \geq 0} & qy \\ \text{s.t.} & Wy = Tx + h. \end{cases} \tag{1}$$

Here, ξ is the vector of random elements from h, q, T, and W. A prototypical problem of this nature is a capacity allocation model under uncertain demand and/or

capacity availabilities. x is a strategic decision allocating resources while y represents an operational recourse decision that is made *after* observing the demand and availabilities. Example applications of this type include capacity expansion planning in an electric power system [16] and in a telecommunications network [61]. The two-stage model generalizes to a more dynamic, multistage model (see, e. g., [10]) in which decisions are made, and random events unfold, over time. For multistage applications in asset-liability management see [13] and in hydro-electric scheduling see [39].

In the context of a simulation model, $f(x, \xi)$ could represent a performance measure under a design specified by x. For example, $f(x, \xi)$ might represent the number of hours in a workday that a critical machine is blocked in a queueing network model of a manufacturing system in which buffer sizes are determined by x. In another application, E.L. Plambeck et al. [53] allocate constrained processing rates to unreliable machines with buffers in a fluid serial queueing network in order to maximize steady-state throughput. In nonterminating simulations, the expectation in $\mathsf{E}f(x, \xi)$ is typically with respect to a steady-state distribution.

Note that $\mathsf{E}f(x, \xi)$ can capture objectives not usually thought of as a 'mean'. For example, if c represents random rates of return and x investment amounts, we might want to maximize the probability of exceeding a return threshold, T. We can write $\mathsf{P}(cx \geq T) = \mathsf{E}I(cx \geq T)$ where $I(\cdot)$ is the *indicator function* that takes value one if its argument is true and zero otherwise. For more on probability maximization models (and generalizations of (SP) in which X contains probabilistic constraints) see [54]. See [45] for a discussion of risk modeling in stochastic optimization.

A more general model than (SP) allows the distribution of ξ to depend on x. Some simple types of dependencies can effectively be captured in (SP) via modeling tricks, such as the x scaling random elements of T in (1). General dependencies, however, are difficult to handle. For work on decision-dependent distributions when there are a finite number of possibilities see [26,40].

Regardless of whether it is defined as the expected value of a mathematical program or as a long-run average performance measure of a discrete-event simulation model, it is usually impossible to calculate $\mathsf{E}f(x, \xi)$ exactly- even for a fixed value of x. When the dimension of the random vector ξ is relatively low, one approach is to obtain deterministic approximations of $\mathsf{E}f(x, \xi)$ using numerical quadrature or related ideas. In stochastic programming, this corresponds to generating and refining bounds on $\mathsf{E}f(x, \xi)$ within a sequential approximation algorithm [20,24,43]. For problems in which ξ is of moderate-to-high dimension and is continuous or has a large number of realizations, *Monte-Carlo simulation* is widely regarded as the method of choice for estimating $\mathsf{E}f(x, \xi)$. As a result, it is not surprising that Monte-Carlo techniques play a fundamental role in solving (SP).

In recent years (1999), considerable progress has been made in solving realistically-sized problems with a significant number of stochastic parameters and decision variables. The telecommunications model considered in [61] has 86 random point-to-point demand pairs and 89 links on which capacity may be installed. In [53] queueing networks with up to 50 nodes are studied. Each node represents a machine with random failures and has a decision variable denoting its assigned cycle time. [53] also solves a stochastic PERT (program evaluation and review technique) problem with 70 nodes and 110 stochastic arcs. The arcs model the times required to complete activities and a decision variable associated with each arc influences (parameterizes) the distribution of the random activity duration. These problems contain objectives with high-dimensional expectations and all were solved using Monte-Carlo methods.

In this article we discuss:

i) several types of Monte-Carlo-based solution procedures that can be used for solving (SP);
ii) methods for testing the quality of a candidate solution $\widehat{x} \in X$;
iii) variance reduction techniques used in stochastic optimization; and
iv) theoretical justification for using sampling.

Solution Procedures

Monte-Carlo methods for approximately solving stochastic optimization problems can typically be classified on the basis of whether the sampling is external to, or internal to, the optimization algorithm. Solution procedures of both types are driven by estimates of objective function values and/or gradients. Before turn-

ing to solution procedures we briefly discuss *gradient estimation*.

In stochastic programming, gradient (or subgradient) estimates of $\mathsf{E}f(x,\xi)$ are typically available via duality. In simulation-based optimization, the primary methods for obtaining gradient estimates are finite differences, the *likelihood ratio* (LR) method (also called the *score function method*) [29,57], and *infinitesimal perturbation analysis* (IPA) [27,35]. *Finite-difference approximations* require minimal structure, needing only estimates of $\mathsf{E}f(x,\xi)$; however, they result in solution procedures that can converge slowly. The LR method is more widely applicable than IPA, but when both apply the IPA approach tends to produce estimators with lower variance. See, for example, [28] for a discussion of these issues.

In the simplest form of 'external sampling' (also called *'sample-path optimization'* [55] and the *'stochastic counterpart' method* [57]) we generate independent and identically distributed (i.i.d.) replicates $\xi^1, \ldots, \xi^n$ from the distribution of ξ and form the approximating problem

$$(\mathrm{SP}_n) \quad z_n^* = \min_{x\in X} \frac{1}{n}\sum_{i=1}^{n} f(x,\xi^i).$$

Even when it is possible to construct (SP_n) using i.i.d. variates, it may be preferable to use another sampling scheme in order to reduce the variance of the resulting estimators. Moreover, in nonterminating simulation models, generating i.i.d. replicates from a stationary distribution is often impossible (for exceptions see recent work on *exact sampling*, e. g., [3,22]), but under appropriate conditions we may run the simulation for a length n and replace the objective function in (SP_n) with a consistent estimate of the desired long-run average performance measure.

After constructing an instance of (SP_n) we employ a (deterministic) optimization algorithm to obtain a solution x_n^*. In the case of stochastic linear programming, (SP_n) is a large scale linear program. The cutting plane algorithm of R.M. Van Slyke and R.J-B. Wets [64], its variant with a quadratic proximal term [58], and its multistage version [7,9] are powerful tools for solving such problems. A cutting plane algorithm with a proximal term and IPA-based gradients is used in an external sampling method for solving the queueing network problem in [53]. See [8] for a recent survey of computational methods for stochastic programming instances of (SP_n).

Intuitively, we might expect solutions of (SP_n) to more accurately approximate solutions of (SP) as n increases. We discuss results supporting this in Sect. "Theoretical Justification for Sampling". In addition, after having solved (SP_n) to obtain x_n^* it would be desirable to know whether n was 'large enough'. More generally, we would like to be able to test the quality of a candidate solution (such as x_n^*). This is discussed in the next section.

We now turn to solution procedures based on internal sampling. These algorithms adapt deterministic optimization algorithms by replacing exact function and gradient evaluations with Monte-Carlo estimates. The sampling is internal because new observations of ξ are generated on an as-needed basis at each iteration of the algorithm. We briefly discuss stochastic adaptations of steepest descent and cutting plane methods.

A deterministic *steepest descent algorithm* for (SP) forms iterates $\{x^\ell\}$ using the recursion

$$x^{\ell+1} \leftarrow \Pi_X\left[x^\ell - \rho^\ell \nabla E f(x^\ell,\xi)\right].$$

Π_X performs a projection onto X and $\{\rho^\ell\}$ are steplengths. It is usually impossible to calculate $\nabla\mathsf{E}f(x,\xi)$ exactly and it must be estimated. *Stochastic approximation* (SA) and *stochastic quasigradient* (SQG) algorithms are stochastic variants of a steepest descent search. The *Keifer-Wolfowitz* SA method uses unbiased estimates of $\mathsf{E}f(x,\xi)$ to form finite-difference approximations of the gradient. The *Robbins–Monro* SA procedure requires unbiased estimates of $\nabla\mathsf{E}f(x,\xi)$. SQG methods do not require that $\mathsf{E}f(x,\xi)$ be differentiable and work under more general assumptions concerning the estimates of (sub)gradients of $\mathsf{E}f(x,\xi)$. In particular, the estimates need not be unbiased but the bias must effectively shrink to zero as the algorithm proceeds. For convergence properties of SA methods see [49] and for SQG procedures see [23].

Cutting plane methods are applicable when $\mathsf{E}f(x,\xi)$ is convex. The iterates $\{x^\ell\}$ are found by solving a sequence of optimization problems of the form

$$\min_{x\in X} \max_{\ell=1,\ldots,L} \mathsf{E}f(x^\ell,\xi) + \nabla\mathsf{E}f(x^\ell,\xi)(x-x^\ell),$$

where L grows as the algorithm proceeds. At each iteration a first order Taylor approximation of $\mathsf{E}f(x,\xi)$, i. e., a cutting plane, is computed at the current iterate x^{ℓ} and is used to refine the piecewise-linear outer approximation of $\mathsf{E}f(x,\xi)$. The key idea is that this approximation need only be accurate in the neighborhood of an optimal solution. For stochastic linear programs, G.B. Dantzig, P.W. Glynn [15], and G. Infanger [37,38] and J.L. Higle and S. Sen [32,34] have developed Monte-Carlo-based cutting plane methods by using statistical estimates for the cut intercepts and gradients. Dantzig, Glynn, and Infanger use separate streams of observations of ξ to estimate each cut. The *stochastic decomposition* algorithm of Higle and Sen uses common random number streams to calculate each cut and employs an updating procedure to ensure that the statistical cuts are asymptotically valid (i. e., lie below $\mathsf{E}f(x,\xi)$). Relative to SA and SQG methods, cutting plane procedures avoid potentially difficult projections and, in practice, have a reputation for converging more quickly, particularly when X is high dimensional.

Grid search and optimization of *metamodels* are two common approaches to optimizing system performance in discrete-event simulation models. In grid search, X is replaced by a 'grid' of points $X_m = \{x^1, \ldots, x^m\}$ and sample-mean estimates

$$\overline{f}_n(x) = 1/n \sum_{i=1}^{n} f(x,\xi^i)$$

are formed at each $x \in X_m$. (SP) is then approximately solved by $z_n^* = \min_{x\in X_m} \overline{f}_n(x)$ with x_n^* being the associated minimizer. Grid search is attractive because it requires minimal structure, but in implementing this procedure, we must exercise care in selecting m and n. With independent sampling at each grid point, K.B. Ensor and Glynn [21] consider the rate at which n must grow relative to m in order to achieve consistency and they also discuss the method's limiting behavior when the rate of growth is at (and slower than) the critical rate.

A metamodel can be used to approximate a more complex simulation model which, in turn, is an approximation of the real system. In such a metamodel, estimates of $\mathsf{E}f(x,\xi)$ are formed at each point in a set specified by an experimental design, and the parameters of the postulated *response surface* are fit to these observed values. The resulting function is then optimized with respect to x. For more on metamodels see, e. g., [11,47]. The review in [25] includes optimization using response surfaces, and metamodeling has also been applied in stochastic programming [5].

The grid-search and metamodel approaches are classified as external sampling procedures if the procedure is executed once. However, it may be desirable to refine the grid (or the region covered by the experimental design) in the neighborhood of promising values of x and repeat the methodology. When it is adaptively repeated in this fashion the procedure is classified as an internal sampling method.

We have not explicitly discussed approaches for when X is discrete. These range from methods for selecting the best design in simulation to those for solving stochastic integer programming models. Finally, sampling-based procedures for multistage stochastic programs have been proposed in [17].

Establishing Solution Quality

Establishing solution quality is a key concept when using an approximation scheme to solve an optimization problem. When applying Monte-Carlo techniques to (SP), the best we can expect are probabilistic quality statements. In the context of external sampling, there has been significant work on studying the behavior of solutions to (SP_n) for large sample sizes (see the last section). There are analogous convergence results for algorithms based on internal sampling. Such results take a number of forms but perhaps the most fundamental is to show that limit points of the sequence of solutions are, say, almost surely optimal to (SP). Next, it is desirable to have a statement regarding the rate of convergence and an associated asymptotic distribution. These consistency and limiting distribution results are aimed at justifying sampling-based methods and may be viewed as establishing solution quality. However, the approach discussed in this section centers on the question: Given a candidate solution $\widehat{x} \in X$, what can be said regarding its quality? Because candidate solutions may be obtained by internal or external sampling schemes or via another, heuristic, method, procedures that can directly test the quality of $\widehat{x}$, regardless of its origin, are very attractive.

One natural way of defining solution quality is by the optimality gap, $\mathsf{E}f(\widehat{x},\xi) - z^*$. An optimal solution

has an optimality gap of zero, but in our setting we hope to make probabilistic statements such as

$$\mathsf{P}\{\mathsf{E}f(\widehat{x},\xi) - z^* \le \epsilon\} \ge \alpha, \tag{2}$$

where ϵ is a random confidence interval width and α is a confidence level, e. g., $\alpha = 0.95$. Unfortunately, exact confidence intervals such as (2) can be difficult to obtain even in relatively simple statistical settings so we attempt to construct approximate confidence intervals

$$\mathsf{P}\{\mathsf{E}f(\widehat{x},\xi) - z^* \le \epsilon\} \approx \alpha. \tag{3}$$

To form a *confidence interval* (3) for $\mathsf{E}f(\widehat{x},\xi) - z^*$ we estimate the mean of a gap random variable $G_n = U_n - L_n$ that is expressed as the difference between upper and lower bound estimators and satisfies $\mathsf{E}G_n \ge \mathsf{E}f(\widehat{x},\xi) - z^*$.

In many problems it is relatively straightforward to estimate the performance of a suboptimal decision $\widehat{x}$ via simulation. For example, the standard sample mean estimator, $U_n = 1/n \sum_{i=1}^{n} f(\widehat{x},\xi^i)$, provides an unbiased estimate of the expected cost of using decision $\widehat{x}$, i. e., $\mathsf{E}f(\widehat{x},\xi)$.

To construct a confidence interval for the optimality gap we also want an estimate of z^*. However, unbiased estimates of z^* are difficult to obtain so an estimator L_n that satisfies $\mathsf{E}L_n \le z^*$ is used. In [51] it is shown that if the objective in (SP_n) is an unbiased estimate of $\mathsf{E}f(x,\xi)$ then $\mathsf{E}z_n^* \le z^*$, i. e., z_n^* is one possible lower bound estimator L_n. Higle and Sen [33] perform a Lagrangian relaxation of a reformulation of (SP_n) which uses explicit 'nonanticipativity' constraints. The resulting lower bound is weaker in expectation than z_n^* but has the computational advantage that the optimization problem separates by scenario.

Once observations of G_n can be formed, we can appeal to the batch means method and use the central limit theorem [51], or a nonparametric approach [31,33], to construct approximate confidence intervals (3). Another approach to examining solution quality is to test the null hypothesis that the (generalized) Karush-Kuhn-Tucker (KKT) optimality conditions are satisfied; see [63]. Higle and Sen [31] also consider the KKT conditions but use them to derive bounds on the optimality gap.

Variance Reduction Techniques

When applying the 'crude' Monte-Carlo method to estimate $\mathsf{E}f(x,\xi)$ for fixed x, we use the standard sample mean estimator based on i.i.d. terms,

$$\frac{1}{n}\sum_{i=1}^{n} f(x,\xi^i).$$

The error associated with this estimate is proportional to

$$\left[\frac{\operatorname{var} f(x,\xi)}{n}\right]^{1/2}. \tag{4}$$

This error can be decreased by increasing the sample size. However, obtaining an additional digit of accuracy requires increasing the sample size by a factor of 100. If f is defined as the optimal value of a mathematical program or as the performance measure of a simulation model, increasing the number of evaluations of f in this fashion can be prohibitively expensive. *Variance reduction techniques* (VRTs) effectively decrease the numerator in (4) instead of increasing the denominator. Many problems for which crude Monte-Carlo would yield useless results are instead made computationally tractable via VRTs. As described in Sect. "Solution Procedures", sampling is also used to estimate $\nabla Ef(x,\xi)$, but for simplicity we primarily restrict our attention to VRTs for estimating $\mathsf{E}f(x,\xi)$.

Some VRTs, including *control variates* (CVs) and *importance sampling* (IS), exploit special structures of $f(x,\xi)$. Suppose that we have $\Gamma_x(\xi)$, with known mean μ_Γ, which is believed to approximate (be positively correlated with) $f(x,\xi)$. In CVs we attempt to 'subtract out' variation by generating observations of $[f(x,\xi) - \Gamma_x(\xi)] + \mu_\Gamma$, which has the same expectation as $f(x,\xi)$. (It is common to incorporate a multiplicative factor with the control variate $\Gamma_x(\xi)$ and also possible to use multiple controls.) In IS we attempt to reduce variance by generating observations of $\mu_\Gamma\,[f(x,\xi)/\,\Gamma_x(\xi)]$. In CVs observations of ξ are generated from its original distribution. However, in IS the expected value of the ratio is not the ratio of expectations and, as a result, there is a change of measure induced by Γ_x that is required to yield an unbiased estimate. Under the new IS distribution, we are more likely to sample ξ where $\Gamma_x(\xi)$ is large, i. e., scenarios that our approximation function predicts have high cost. In an IS scheme for

stochastic linear programs, [15,37] use an approximation function that is separable in the components of ξ while [48] utilizes a piecewise-linear approximation. See [12] for the solution of a stochastic optimization problem to price American-style financial options using the simpler European option as a control variate. These papers report significant variance reduction in computational results.

Other VRTs exploit correlation structures in the solution methodology.*Common random numbers* (CRNs) are often used in simulation when comparing the performance of two systems. The use of CRNs has been suggested in a stochastic approximation method with finite differences where the same stream is used for the forward and backward point estimates [50]. The upper and lower bounds used to determine solution quality (see the previous section) may be viewed as two 'systems' and the use of CRNs in estimating their difference has been advocated in [34,51]. In order to reduce the error in the resulting response surface, various methods have been proposed for generating the streams of observations of ξ at each point in the experimental design. The *Schruben–Margolin* scheme [59] uses a mixture of CRNs and antithetic variates and an extension [65] also incorporates CVs.

Another group of VRTs attempts to more regularly spread the sampled observations over the support of ξ. Such techniques include stratified sampling and Latin hypercube sampling as well as quasi-Monte-Carlo techniques in which the sequence of observations is deterministic. Empirical results in [30] for two-stage stochastic linear programming compare the variance reduction obtained by stratified sampling, antithetic variates, IS, and CVs and suggest that a CV procedure performs relatively well, particularly on high-variance problems.

Theoretical Justification for Sampling

In Sect. "Solution Procedures" we formed an approximating problem for external sampling procedures by using the sample mean estimator of $\mathsf{E}f(x,\xi)$. Here we redefine (SP_n) as

$$(\mathrm{SP}_n) \quad z_n^* = \min_{x\in X} \mathsf{E}_n f(x,\xi),$$

with x_n^* again denoting an optimal solution. In (SP) the expected value operator E is with respect to the 'true' probability measure P while in (SP_n), E_n is with respect to a measure P_n that is a statistical estimate of P. If Monte-Carlo methods are used to generate i.i.d. replicates from P then P_n is the associated (random) *empirical measure*.

Since z_n^* is an estimator of z^* and x_n^* an estimator of an optimal solution to (SP), it is natural to study the behavior of these estimators for large sample sizes. For example, under what conditions do we obtain consistency and what can be said concerning rates of convergence? Positive answers to such questions provide theoretical justification for employing external Monte-Carlo sampling techniques to solve (SP).

In general, (SP_n) and (SP) may have multiple optimal solutions and so we cannot expect $\{x_n^*\}$ to converge. Instead, establishing consistency of x_n^* amounts to showing that the accumulation points of the sequence are almost surely optimal to (SP). If, for example, the samples are i.i.d. then by the strong law of large numbers we have $\mathsf{E}_n f(x,\xi) \to \mathsf{E}f(x,\xi)$, a.s., for all x. Unfortunately, this does not ensure that $\{x_n^*\}$ has accumulation points that are optimal to (SP) and that $z_n^* \to z^*$, a.s. [4].

The notion of *epiconvergence* plays a fundamental role in establishing consistency results for x_n^* and z_n^*; see [4]. A sequence of functions $\{\phi_n\}$ is said to epiconverge to ϕ (written $\phi_n \overset{\text{epi}}{\to} \phi$) if the *epigraphs* of ϕ_n, $\{(x,\beta)\colon \beta \geq \phi_n(x)\}$, converge to that of ϕ. Epiconvergence is weaker than classical uniform convergence. P. Kall [41] provides an excellent review of various types of convergence, their relations, and their implications for approximations of optimization models. Epiconvergence is a valuable property because of the following result:

Theorem 1 *Suppose* $\phi_n \overset{epi}{\to} \phi$. *If* $\widehat{x}$ *is an accumulation point of* $\{x_n^*$, *where* $x_n^* \in$ *argmin* $\phi_n(x)$, *then* $\widehat{x} \in \operatorname{argmin} \phi(x)$.

Constrained optimization is captured in this result because ϕ_n and ϕ are defined to be extended-real-valued functions that take value $+\infty$ at infeasible points. While it is possible that the sequence of optimizers $\{x_n^*\}$ has no accumulation points, this potential difficulty is avoided if the feasible region X is compact (i. e., closed and bounded).

Because of the implications of epiconvergence, there is considerable interest in determining sufficient

conditions on f, P_n, and P under which $\mathsf{E}_n f(x,\xi) \overset{\text{epi}}{\to} \mathsf{E} f(x,\xi)$, a.s. Note that because $\{\mathsf{P}_n\}$ are random measures, the epiconvergence of the approximating functions is with probability one (also called *epiconsistency*). Under this hypothesis the accumulation points of $\{x_n^*\}$ are almost surely optimal to (SP); see [19].

Sufficient conditions for achieving $\mathsf{E}_n f(x,\xi) \overset{\text{epi}}{\to} \mathsf{E} f(x,\xi)$, a.s. are examined in [19,42,55], and [56]. Roughly speaking, we will obtain epiconsistency if f is sufficiently smooth, P_n converges weakly to P with probability one, and the tails of the distributions are well-behaved relative to f. See [2,60] for results when f is discontinuous.

For two-stage stochastic programming in which the recourse matrix W in (1) is deterministic and P_n is the empirical measure, [46] contains consistency results under modest assumptions. We note that is possible to develop consistency results using other (stronger) types of convergence of $\mathsf{E}_n f(x,\xi)$ to $\mathsf{E} f(x,\xi)$; see, for example, [52].

There is a large literature on consistency, stability, and rates of convergence for solutions of (SP_n). Much of this work may be viewed as generalizing earlier results on constrained *maximum likelihood estimation* in [1] and [36]. Under restrictive assumptions, asymptotic normality for $\sqrt{n}(z_n^* - z^*)$ and $\sqrt{n}(x_n^* - x^*)$ may be obtained, e. g., [19]. However, when inequality constraints in X play a nontrivial role we cannot, in general, expect to obtain limiting distributions that are normal [18,44,62]. See [44] for a limiting distribution for $\sqrt{n}(x_n^* - x^*)$ that is the solution of a (random) quadratic program.

See also

► Monte-Carlo Simulated Annealing in Protein Folding

References

1. Aitchison J, Silvey SD (1958) Maximum-likelihood estimation of parameters subject to restraints. Ann Math Statist 29:813–828
2. Artstein Z, Wets RJ-B (1994) Stability results for stochastic programs and sensors, allowing for discontinuous objective functions. SIAM J Optim 4:537–550
3. Asmussen S, Glynn PW, Thorisson H (1992) Stationary detection in the initial transient problem. ACM Trans Modeling and Computer Simulation 2:130–157
4. Attouch H, Wets RJ-B (1981) Approximation and convergence in nonlinear optimization. In: Mangasarian O, Meyer R, Robinson S (eds) Nonlinear Programming, vol 4. Acad Press, New York, pp 367–394
5. Bailey TG, Jensen PA, Morton DP (1999) Response surface analysis of two-stage stochastic linear programming with recourse. Naval Res Logist 46:753–778
6. Beale EML (1955) On minimizing a convex function subject to linear inequalities. J Royal Statist Soc 17B:173–184
7. Birge JR (1985) Decomposition and partitioning methods for multistage stochastic linear programs. Oper Res 33:989–1007
8. Birge JR (1997) Stochastic programming computation and applications. INFORMS J Comput 9:111–133
9. Birge JR, Donohue CJ, Holmes DF, Svintsitski OG (1996) A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. Math Program 75:327–352
10. Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer, Berlin
11. Box GEP, Draper NR (1987) Empirical model-building and response surfaces. Wiley, New York
12. Broadie M, Glasserman P (1997) Pricing American-style options using simulation. J Econom Dynam Control 21:1323–1352
13. Cariño DR, Kent T, Meyers DH, Stacy C, Sylvanus M, Turner AL, Watanabe K, Ziemba WT (1994) The Russell-Yasuda Kasia model: An asset/liability model for a Japanese insurance company using multistage stochastic programming. Interfaces 24:29–49
14. Dantzig GB (1955) Linear programming under uncertainty. Managem Sci 1:197–206
15. Dantzig GB, Glynn PW (1990) Parallel processors for planning under uncertainty. Ann Oper Res 22:1–21
16. Dantzig GB, Glynn PW, Avriel M, Stone JC, Entriken R, Nakayama M (1989) Decomposition techniques for multiarea generation and transmission planning under uncertainty. Report Electric Power Res Inst EPRI 2940-1
17. Dempster MAH, Thompson RT (1999) EVPI-based importance sampling solution procedures for multistage stochastic linear programmes on parallel MIMD architectures. Ann Oper Res 90:161–184
18. Dupačová J (1991) On non-normal asymptotic behavior of optimal solutions for stochastic programming problems and on related problems of mathematical statistics. Kybernetika 27:38–51
19. Dupačová J, Wets RJ-B (1988) Asymptotic behavior of statistical estimators and of optimal solutions of stochastic optimization problems. Ann Statist 16:1517–1549
20. Edirisinghe C, Ziemba WT (1996) Implementing bounds-based approximations in convex-concave two-stage stochastic programming. Math Program 75:295–325
21. Ensor KB, Glynn PW (1997) Stochastic optimization via grid search. In: Yin GG, Zhang Q (eds) Mathematics of Stochas-

tic Manufacturing Systems, vol 33. Lect Appl Math Amer Math Soc, Providence, pp 89–100
22. Ensor KB, Glynn PW (2000) Simulating the maximum of a random walk. J Statist Planning Inference 85:127–135
23. Ermoliev Y (1988) Stochastic quasigradient methods. In: Ermoliev Y, Wets RJ-B (eds) Numerical Techniques for Stochastic Optimization. Springer, Berlin, pp 141–185
24. Frauendorfer K (1992) Stochastic two-stage programming. of Lecture Notes Economics and Math Systems, vol 392. Springer, Berlin
25. Fu MC (1994) Optimization via simulation: A review. Ann Oper Res 53:199–248
26. Futschik A, Pflug GCh (1997) Optimal allocation of simulation experiments in discrete stochastic optimization and approximative algorithms. Europ J Oper Res 101:245–260
27. Glasserman P (1991) Gradient estimation via perturbation analysis. Kluwer, Dordrecht
28. Glynn PW (1989) Optimization of stochastic systems via simulation. In: Proc 1989 Winter Simulation Conf, pp 90–105
29. Glynn PW (1990) Likelihood ratio gradient estimation for stochastic systems. Comm ACM 33(10):75–84
30. Higle JL (1998) Variance reduction and objective function evaluation in stochastic linear programs. INFORMS J Comput 10:236–247
31. Higle JL, Sen S (1991) Statistical verification of optimality conditions for stochastic programs with recourse. Ann Oper Res 30:215–240
32. Higle JL, Sen S (1991) Stochastic decomposition: An algorithm for two-stage linear programs with recourse. Math Oper Res 16:650–669
33. Higle JL, Sen S (1996) Duality and statistical tests of optimality for two stage stochastic programs. Math Program 75:257–275
34. Higle JL, Sen S (1996) Stochastic decomposition: A statistical method for large scale stochastic linear programming. Kluwer, Dordrecht
35. Ho YC, Cao XR (1991) Perturbation analysis of discrete event dynamic systems. Kluwer, Dordrecht
36. Huber PJ (1967) The behavior of maximum likelihood estimates under nonstandard conditions. In: Proc Fifth Berkeley Symp Math Stat Probab, pp 221–233
37. Infanger G (1992) Monte Carlo (importance) sampling within a Benders decomposition algorithm for stochastic linear programs. Ann Oper Res 39:69–95
38. Infanger G (1993) Planning under uncertainty: Solving large-scale stochastic linear programs. Sci Press Ser. Boyd & Fraser, Danvers
39. Jacobs J, Freeman G, Grygier J, Morton D, Schultz G, Staschus K, Stedinger J (1995) SOCRATES: A system for scheduling hydroelectric generation under uncertainty. Ann Oper Res 59:99–133
40. Jonsbråten TW, Wets RJ-B, Woodruff DL (1998) A class of stochastic programs with decision dependent random elements. Ann Oper Res 82:83–106
41. Kall P (1986) Approximation to optimization problems: An elementary review. Math Oper Res 11:9–18
42. Kall P (1987) On approximations and stability in stochastic programming. In: Guddat J, Jongen HTh, Kummer B, Nožička F (eds) Parametric Optimization and Related Topics. Akad Verlag, Berlin, pp 387–407
43. Kall P, Ruszczyński A, Frauendorfer K (1988) Approximation techniques in stochastic programming. In: Ermoliev Y, Wets RJ-B (eds) Numerical Techniques for Stochastic Optimization. Springer, Berlin, 33–64
44. King AJ, Rockafellar RT (1993) Asymptotic theory for solutions in statistical estimation and stochastic programming. Math Oper Res 18:148–162
45. King AJ, Takriti S, Ahmed S (1997) Issues in risk modeling for multi-stage systems. IBM Res Report RC 20993
46. King AJ, Wets RJ-B (1991) Epi-consistency of convex stochastic programs. Stochastics 34:83–91
47. Kleijnen JPC, Groenendaal WVan (1992) Simulation: A statistical perspective. Wiley, New York
48. Krishna AS (1993) Enhanced algorithms for stochastic programming. SOL Report Dept Oper Res Stanford Univ 93–8
49. Kushner HJ, Yin GG (1997) Stochastic approximation algorithms and applications. Springer, Berlin
50. L'Ecuyer P, Giroux N, Glynn PW (1994) Stochastic optimization by simulation: numerical experiments with the M/M/1 queue in steady-state. Managem Sci 40:1245–1261
51. Mak WK, Morton DP, Wood RK (1999) Monte Carlo bounding techniques for determining solution quality in stochastic programs. Oper Res Lett 24:47–56
52. Pflug GCh, Ruszczyński A, Schultz R (1998) On the Glivenko–Cantelli problem in stochastic programming: Linear recourse and extensions. Math Oper Res 23:204–220
53. Plambeck EL, Fu B-R, Robinson SM, Suri R (1996) Sample-path optimization of convex stochastic performance functions. Math Program 75:137–176
54. Prékopa A (1995) Stochastic programming. Kluwer, Dordrecht
55. Robinson SM (1996) Analysis of sample-path optimization. Math Oper Res 21:513–528
56. Robinson SM, Wets RJ-B (1987) Stability in two-stage stochastic programming. SIAM J Control Optim 25:1409–1416
57. Rubinstein RY, Shapiro A (1993) Discrete event systems: Sensitivity and stochastic optimization by the score function method. Wiley, New York
58. Ruszczyński A (1986) A regularized decomposition method for minimizing a sum of polyhedral functions. Math Program 35:309–333
59. Schruben LW, Margolin BH (1978) Pseudorandom number assignment in statistically designed simulation and distribution sampling experiments. J Amer Statist Assoc 73:504–525

60. Schultz R (1995) On structure and stability in stochastic programs with random technology matrix and complete integer recourse. Math Program 70:73–89
61. Sen S, Doverspike RD, Cosares S (1994) Network planning with random demand. Telecommunication Systems 3:11–30
62. Shapiro A (1989) Asymptotic properties of statistical estimators in stochastic programming. Ann Statist 17:841–858
63. Shapiro A, Homem-de-Mello T (1998) A simulation-based approach to two-stage stochastic programming with recourse. Math Program 81:301–325
64. Slyke RM Van, Wets RJ-B (1969) L-Shaped linear programs with applications to optimal control and stochastic programming. SIAM J Appl Math 17:638–663
65. Tew JD (1995) Simulation metamodel estimation using a combined correlation-based variance reduction technique for first and higher-order metamodels. Europ J Oper Res 87:349–367

Motzkin Transposition Theorem

MTT

ARKADI NEMIROVSKI[1], KEES ROOS[2]
[1] Fac. Industrial Engineering and Management Technion, Israel Institute Technol., Haifa, Israel
[2] Department ITS/TWI/SSOR, Delft University Technol., Delft, The Netherlands

MSC2000: 15A39, 90C05

Article Outline

Keywords
See also
References

Keywords

Inequality systems; Duality; Certificate; Transposition theorem

Motzkin's transposition theorem (MTT) [1] is a so-called *theorem of the alternative* (cf. ► Linear Optimization: Theorems of the Alternative). It deals with the question whether or not a given system of *linear inequalities* has a solution. In the most general case such a system has the form

$$\text{(S)} \quad Ax \geq a, \quad Bx > b,$$

where A and B are matrices of size $m \times n$ and $p \times n$, respectively, and where $Ax \geq a$ contains the 'larger than or equal' inequalities and $Bx > b$ the 'larger than' inequalities. Note that inequalities of the opposite type ('smaller than or equal' or 'smaller than') can be turned into the appropriate form by multiplying them by -1.

The Motzkin transposition theorem states that the system (S) has no solution if and only if at least one of the systems (T1) and (T2) has a solution, where the latter systems are given by

$$\text{(T1)} \quad \begin{cases} y^\top A + v^\top B = 0, & y^\top a + v^\top b > 0, \\ y \geq 0, & v \geq 0, \end{cases}$$

and

$$\text{(T2)} \quad \begin{cases} y^\top A + v^\top B = 0, & y^\top a + v^\top b \geq 0, \\ y \geq 0, & v \geq 0, \quad v \neq 0, \end{cases}$$

respectively.

In other words, when one has a solution of (T1) or of (T2) this solution is a *certificate* for the fact that the given system (S) is *infeasible*, i. e., has no solution.

It makes sense to formulate two most useful principles following from the theorem.

Theorem 1 (*Principle A*) *The system (S) is infeasible if and only if one can combine the inequalities in (S) in a linear fashion (i. e., multiply each inequality with a nonnegative number and add the results) to get the contradictory inequality $0 > 0$ (or $0 \geq 1$).*

To see that this is exactly what the MTT says, let y and v denote nonnegative vectors of appropriate sizes. Then the inequality

$$\left(y^\top A + v^\top B\right) x \geq y^\top a + v^\top b \tag{1}$$

is a consequence of the inequalities in (S), and if the vector v is not the zero vector, then also the stronger inequality

$$\left(y^\top A + v^\top B\right) x > y^\top a + v^\top b \tag{2}$$

is a consequence of (S). The inequalities (1) and (2) have certainly solutions if $y^\top A + v^\top B \neq 0$. But if $y^\top A + v^\top B = 0$ then (1) yields a contradiction if $y^\top a + v^\top b > 0$ and (2) if $y^\top a + v^\top b \geq 0$. The first case occurs if (T1) has a solution and the second case if (T2) has a solution.

The second principle is:

Theorem 2 (*Principle B*) *If (S) is feasible, then a linear inequality is a consequence of the inequalities in (S) if and only if it can be obtained by combining, in a linear fashion, the inequalities in (S) and the trivial inequality* $0 \geq -1$.

This principle can be understood in a similar way: If (S) is feasible, then $c^\intercal x \geq z$ is an implied inequality if and only if

$$Ax \geq a,\ Bx > b \quad \Rightarrow \quad c^\top x \geq z,$$

which is equivalent to the system

$$Ax \geq a, \quad Bx > b, \quad -c^\top x > -z$$

being infeasible. By Principle A this happens if and only if there exist nonnegative vectors y and v and a nonnegative scalar λ such that

$$\left(y^\top A + v^\top B - \lambda c\right) x \geq y^\top a + v^\top b - \lambda z$$

is a contradictory inequality. Hence $y^\intercal A + v^\intercal B - \lambda c = 0$ and $y^\intercal a + v^\intercal b - \lambda z > 0$. Since (S) is feasible, we must have $\lambda > 0$. Without loss of generality we may assume $\lambda = 1$. Then $c = y^\intercal A + v^\intercal B$ and $z \geq y^\intercal a + v^\intercal b$. This proves the claim.

The above principles are highly nontrivial and very deep. Consider, e. g., the following system of 4 inequalities with two variables u, v:

$$-1 \leq u \leq 1,$$
$$-1 \leq v \leq 1.$$

From these inequalities it follows that

$$u^2 + v^2 \leq 2,$$

which in turn implies, by the *Cauchy inequality*, the inequality $u + v \leq 2$:

$$u + v = 1 \cdot u + 1 \cdot v \leq \sqrt{1^2 + 1^2}\sqrt{u^2 + v^2} \leq 2.$$

The concluding inequality is linear, and is a consequence of the original system, but the above derivation is 'highly nonlinear'. It is absolutely unclear a priori why the same inequality can also be obtained from the given system in a linear manner as well, as stated by Principle B. Of course, it can – it suffices to add the inequalities $u \leq 1$ and $v \leq 1$.

The MTT is one of the deepest result in the part of mathematics dealing with linear inequalities and, in fact, is logically equivalent to other deep results in this discipline. For example, it is equivalent to the duality theorem for linear optimization (cf. ▶ Linear Programming). To demonstrate this, consider the *linear optimization problem*

$$\text{(P)} \quad \min\left\{c^\top x\colon\ Ax \geq b\right\}.$$

Let z^* denote the optimal value of (P), where we take $z^* = -\infty$ if (P) is unbounded and $z^* = \infty$ if (P) is infeasible. Now, a real z is a lower bound on the optimal value of (P) if and only if $c^\intercal x \geq z$ is a consequence of $Ax \geq b$, or, which is the same, if and only if the system of linear inequalities

$$(\mathrm{S}_z) \quad Ax \geq b, \quad -c^\top x > -z$$

has no solutions. By the MTT this is the case if and only if at least one of the systems

$$(\mathrm{T1}_z) \quad \begin{cases} y^\top A - y_0 c = 0, & y^\top b - y_0 z > 0, \\ y \geq 0, & y_0 \geq 0 \end{cases}$$

and

$$(\mathrm{T2}_z) \quad \begin{cases} y^\top A - y_0 c = 0, & y^\top b - y_0 z \geq 0, \\ y \geq 0, & y_0 > 0 \end{cases}$$

has a solution. Note that the only difference between these two systems is that $(\mathrm{T1}_z)$ requires $y_0 \geq 0$ whereas $(\mathrm{T2}_z)$ requires $y_0 > 0$. Also, since the system $(\mathrm{T2}_z)$ is homogeneous, without loss of generality we may take $y_0 = 1$. Thus it follows that z is a lower bound on the optimal value of (P) if and only if one of the following two systems

$$(\mathrm{T1}'_z) \quad y^\top A = 0, \quad y^\top b > 0, \quad y \geq 0$$

and

$$(\mathrm{T2}'_z) \quad y^\top A = c, \quad y^\top b \geq z, \quad y \geq 0$$

has a solution. Observe that z does not appear in $(\mathrm{T1}'_z)$. Therefore, if this system has a solution then each real z is a lower bound on the optimal value of (P), but this occurs if and only the problem (P) is infeasible. Assuming that (P) is feasible, it follows that z is a lower bound on the optimal value of (P) if and only if the system $(\mathrm{T2}'_z)$

has a solution. Given a solution y of (T2$'_z$) any z satisfying $y^\top b \geq z$ is a lower bound and the largest lower bound provided in this way is $y^\top b$. Hence, the largest possible lower bound on the optimal value of (P) is the optimal value of the problem

$$\text{(D)} \quad \max \left\{ b^\top y \colon \; y^\top A = c, \; y \geq 0 \right\}.$$

If the problem (P) is unbounded, i. e., if there does not exists a lower bound on the optimal value of (P), then the problem (D) must be infeasible. Otherwise the optimal value of (D) must coincide with the optimal value of (P).

The problem (D) is called the *dual problem* of the *primal problem* (P). The above findings can be summarized as follows:

> if one of the two problems (P) and (D) is unbounded then the other is infeasible; if both problems are feasible then they have both an optimal solution and the optimal values are the same.

This is the *duality theorem for linear optimization*. Note that one other case may occur, namely that both problems are infeasible. It became clear above that (P) is infeasible if and only if (T1$'_z$) has a solution, so

> the primal problem (P) is infeasible if and only if there exists a *dual ray* y, i. e., a vector y such that
>
> $$y^\top A = 0, \quad y^\top b > 0, \quad y \geq 0. \tag{3}$$

In fact, the latter statement is equivalent to the statement that (3) and $Ax \geq b$ are *alternative systems*, which is the special case of the MTT occurring when B is vacuous and which is known as *Farkas' lemma*. (See ► Linear Optimization: Theorems of the Alternative and ► Farkas Lemma.) In just the same way it can be derived from a variant of Farkas' lemma that:

> the dual problem (D) is infeasible if and only if there exists a *primal ray* x, i. e., a vector x such that
>
> $$Ax \geq 0, \quad c^\top x < 0. \tag{4}$$

It has been shown above that the MTT implies the duality theorem for linear optimization. The converse is also true: Assuming the duality theorem for linear optimization, the MTT easily can be proved, showing that the two results are logically equivalent. This goes in two steps. Assuming the duality theorem for linear optimization, first one derives Farkas' lemma and then it is shown that the MTT follows. To derive Farkas' lemma, consider the problem

$$\min \left\{ 0^\top x \colon \; Ax \geq b \right\}.$$

Clearly, the system $Ax \geq b$ has a solution if and only if the optimal value of this problem is zero. By the duality theorem this holds if and only if the optimal value of the dual problem

$$\max \left\{ b^\top y \colon \; y^\top A = 0, \; y \geq 0 \right\}$$

is also zero. This holds if and only

$$y^\top A = 0, \; y \geq 0 \quad \Rightarrow \quad b^\top y \leq 0,$$

which is true if and only if the system

$$y^\top A = 0, \quad y \geq 0, \quad b^\top y > 0$$

has no solution, proving Farkas' lemma.

To prove the MTT, one derives from Farkas' lemma that the 'weaker' system

$$\text{(S1)} \quad Ax \geq a, \quad Bx \geq b$$

is infeasible if and only if the system (T1) has a solution. If (S1) is feasible then one easily verifies that (S) has no solution if and only if the optimal value of the problem

$$\text{(P1)} \quad \min \left\{ v \colon \; Ax \geq a, \; Bx + ve \geq b \right\}$$

is a nonnegative real. Here e denotes the all-one vector. Since (P1) is feasible and below bounded, by the duality theorem this happens if and only if the optimal value of the dual problem

$$\text{(D1)} \quad \max \left\{ a^\top y + b^\top v \colon \begin{array}{c} y^\top A + v^\top B = 0, \\ e^\top v = 1, \\ y \geq 0, \quad v \geq 0 \end{array} \right\}$$

is a nonnegative real and, finally, this occurs if and only if (T2) has a solution. Thus it has been shown that the MTT is logically equivalent to the duality theorem for linear optimization.

So far the issue of how to prove the MTT has not been touched. One possible approach is to prove the duality theorem for linear optimization and then derive the MTT in the above described way. This approach is now quite popular in text books. For a recent example see, e. g., [2]. The easiest way for a direct proof is to prove first the Farkas' lemma and then derive the MTT from this lemma. The latter step uses the easy to verify statement that (S) has no solution if and only if the system

$$
\begin{aligned}
&Ax - ta \geq 0,\\
&Bx - tb - se \geq 0,\\
&t - s \geq 0,\\
&-s < 0
\end{aligned}
$$

has no solution. Application of a suitable variant of Farkas' lemma to this system yields the MTT. Farkas' lemma and its proof have a rich history; for a nice and detailed survey one might consult [3].

See also

- ► Farkas Lemma
- ► Linear Optimization: Theorems of the Alternative
- ► Linear Programming
- ► Minimum Concave Transportation Problems
- ► Multi-index Transportation Problems
- ► Stochastic Transportation and Location Problems
- ► Tucker Homogeneous Systems of Linear Relations

References

1. Motzkin TS (1936) Beiträge zur Theorie der Linearen Ungleichungen. PhD Thesis Azriel, Jerusalem
2. Padberg M (1995) Linear optimization and extensions. of Algorithms and Combinatorics, vol 12. Springer, Berlin
3. Schrijver A (1986) Theory of linear and integer programming. Wiley, New York

Multi-Class Data Classification via Mixed-Integer Optimization

Metin Türkay, Fadime Üney Yüksektepe
Department of Industrial Engineering,
Koç University, Istanbul, Turkey

Article Outline

Introduction
MILP Formulation
Training Problem Formulation
Testing Problem Formulation
Application
Conclusion
References

Introduction

Data classification is a supervised learning strategy that analyzes the organization and categorization of data in distinct classes [14]. Generally, a training set, in which all objects are already associated with known class labels, is used by classification methods. The data classification algorithm works on this set by using input attributes and builds a model to classify new objects. In other words, the algorithm predicts output attribute values. Output attribute of the developed model is categorical [4]. There are many applications of data classification in finance [6,14], health care [14], sports [14], engineering [10,14] and science [10]. Data classification is an important problem that has applications in a diverse set of areas ranging from finance to bioinformatics.

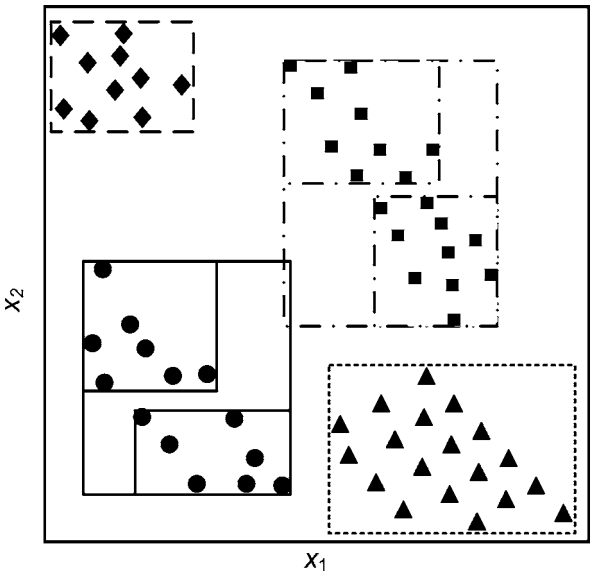
A broad range of methods exists for data classification problem including Decision Tree Induction [14], Bayesian Classifier [14], Neural Networks (NN) [10], Support Vector Machines (SVM) [10] and Mathematical Programming (MP) [1]. An overall view of classification methods is published by Weiss and Kulikowski [21]. A neural network is a data structure that attempts to simulate the behavior of neurons in a biological brain [14]. A major shortcoming of the neural network approach is a lack of explanation of the constructed model. The possibility of obtaining a nonconvergent solution due to the wrong choice of initial weights and the possibility of resulting in a non-optimal solution due to the local minima problem are important handicaps of neural network-based methods. SVM approach operates by finding a hyper surface that will split the classes so that the distance between the hyper surface and the nearest of the points in the groups has the largest value [19]. The main goal is to generate a separating hyper surface which maximizes the margin and produces good generalization ability [10]. In recent years, SVM has been considered one of the most effi-

cient methods for two-class classification problems [5]. SVM method has two important drawbacks in multi-class classification problems; a combination of SVM has to be used in order to solve the multi-class classification problems and some approximation algorithms are used in order to reduce the computational time for SVM while learning the large scale of data.

There have been numerous attempts to solve classification problems using mathematical programming. A survey of classification methods using mathematical programming is published by Joachimsthaler and Stam [11]. The mathematical programming approach to data classification was first introduced in early 1980's. Since then, numerous mathematical programming models have appeared in the literature. As an extension of complement to these, Erenguc and Koehler provide a comprehensive review [7]. Many distinct mathematical programming methods with different objective functions are developed in the literature. These include; minimizing the maximum exterior deviation, minimizing the weighted sum of exterior deviations, minimizing a measure of exterior deviations while maximizing a measure of interior deviations, minimizing the number of misclassifications, and minimizing a generalized distance measure. Most of these methods modeled data classification as linear programming (LP) problems to optimize a distance function. Contrary to LP problems, mixed-integer linear programming (MILP) problems that minimize the misclassifications on the design data set are also widely studied [7]. Mathematical programming methods have certain advantages over the parametric ones. For instance, they are free from parametric assumptions and weights to be adjusted. Moreover, varied objectives and more complex problem formulations can easily be accommodated. On the other hand, obtaining a solution without any discriminating power, unbounded solutions and excessive computational effort requirement are some of the problems in mathematical programming based methods. Koehler [12] surveys the potential problems in mathematical programming formulations. There have been several attempts to formulate data classification problems as MILP problems [2,8,13,15]. Since MILP methods suffer from computational difficulties, the efforts are mainly focused on efficient solutions for two-group supervised classification problems. Although ways to solve a multi-class data classification problem exist by means of solving several two-group problems, such approaches also have drawbacks including computational complexity resulting in long computational times [16].

MILP Formulation

The objective in data classification is to assign data points that are described by several attributes into a predefined number of classes. The use of hyper-boxes for defining boundaries of the sets that include all or some of the points in that set as shown in Fig. 1 can be very accurate on multi-class problems. If it is necessary, more than one hyper-box could be used in order to represent a class as shown in Fig. 1. When the classes that are indicated by square and circle data points are both represented by a single hyper-box respectively, the boundaries of these hyper-boxes will overlap. Thus, two boxes are constructed in order to eliminate this overlapping. A very important consideration in using hyper-boxes is the number of boxes used to define a class. If the total number of hyper-boxes is equal to the number of classes, then the data classification is very efficient. On the other hand; if there are as many hyper-boxes of a class as the number of data points in a class, then the data classification is inefficient.



Multi-Class Data Classification via Mixed-Integer Optimization, Figure 1
Schematic representation of multi-class data classification using hyper-boxes

The data classification problem based on this new idea is built in two parts: training and testing. Determination of the characteristics of the data points that belong to a certain class and differentiation them from the data points that belong to other classes are the targets done during the training part. Thus, boundaries of the classes are formed by the construction of hyper-boxes in the training step. After the distinguishing characteristics of the classes are determined, then the effectiveness of the classification must be tested. Predictive accuracy of the developed model is performed on a test data set during the test part.

Training Problem Formulation

Training part studies are performed on a training data set composed of a number of instances i. The data points are represented by the parameter a_{im} that denotes the value of attribute m for the instance i. The class k that the data point i belongs to are given by the set D_{ik}. Each existing hyper-box l encloses a number of data points belonging to the class k. Moreover, bounds n (*lower, upper)* of each hyper-box is determined by solving the training problem.

Given these parameters and the sets, the following variables are sufficient to model the data classification problem with hyper-boxes. The binary variable ybl is indicates whether the box l is used or not. The position (inside or outside) of the data point i with regard to box l is represented by ypb_{il}. The assigned class k of box l and data point i is symbolized by ybc_{lk} and ypc_{ik}, respectively. If the data point i is within the bound n with respect to attribute m of box l, then the binary variable $ypbn_{ilmn}$ takes the value of 1, otherwise 0. Similarly, $ypbm_{ilm}$ indicates whether the data point i is within the bounds of attribute m of box l or not. Finally, yp_{ik} indicate the misclassification of data points. In order to define the boundaries of hyper-boxes, two continuous variables are required: X_{lmn} is the one that models bounds n for box l on attribute m. Correspondingly, bounds n for box l of class k on attribute m are defined with the continuous variable $XD_{l,k,m,n}$.

The following MILP problem models the training part of data classification method using hyper-boxes:

$$\min z = \sum_i \sum_k yp_{ik} + c \sum_l yb_l \tag{1}$$

subject to

$$XD_{lkmn} \le a_{im} ypb_{il} \quad \forall i,k,l,m,n \,|\, n = lo \tag{2}$$

$$XD_{lkmn} \ge a_{im} ypb_{il} \quad \forall i,k,l,m,n \,|\, n = up \tag{3}$$

$$XD_{lkmn} \le Q ybc_{lk} \quad \forall k,l,m,n \tag{4}$$

$$\sum_k XD_{lkmn} = X_{lmn} \quad \forall l,m,n \tag{5}$$

$$ypbn_{ilmn} \ge (1/Q)(X_{lmn} - a_{im}) \quad \forall i,l,m,n \,|\, n = up \tag{6}$$

$$ypbn_{ilmn} \ge (1/Q)(a_{im} - X_{lmn}) \quad \forall i,l,m,n \,|\, n = lo \tag{7}$$

$$\sum_l ypb_{il} = 1 \quad \forall i \tag{8}$$

$$\sum_k ypc_{ik} = 1 \quad \forall i \tag{9}$$

$$\sum_l ypb_{il} = \sum_k ypc_{ik} \quad \forall i \tag{10}$$

$$\sum_k ybc_{lk} \le yb_l \quad \forall l \tag{11}$$

$$ybc_{lk} - \sum_i ypb_{il} \le 0 \quad \forall l,k \tag{12}$$

$$ybc_{lk} - \sum_i ypc_{ik} \le 0 \quad \forall l,k \tag{13}$$

$$\sum_n ypbn_{ilmn} - ypbm_{ilm} \le \mathrm{N} - 1 \quad \forall i,l,m \tag{14}$$

$$\sum_m ypbm_{ilm} - ypb_{il} \le \mathrm{M} - 1 \quad \forall i,l \tag{15}$$

$$ypc_{ik} - yp_{ik} \le 0 \quad \forall i,k \notin D_{ik} \tag{16}$$

$$X_{lmn}, XD_{lkmn} \ge 0\,,\ yb_l, ybc_{lk}, ypb_{il}, ypc_{ik}, ypbn_{ilmn}, ypbm_{ilm}, yp_{ik} \in \{0,1\} \tag{17}$$

The objective function of the MILP problem (Eq. (1)) is to minimize the misclassifications in the data set with the minimum number of hyper-boxes. In order to eliminate unnecessary use of hyper-boxes, the unnecessary existence of a box is penalized with a small scalar c in

the objective function. The lower and upper bounds of the boxes are given in Eqs. (2) and (3), respectively. The lower and upper bounds for the hyper-boxes are determined by the data points that are enclosed within the hyper-box. Eq. (4) enforces the bounds of hyper-boxes exist if and only if this hyper-box is assigned to a class. Eq. (5) is used to relate the two continuous variables that represent the bounds of the hyper-boxes. The position of a data point with respect to the bounds on attribute m for a hyper-box is given in Eqs. (6) and (7). The binary variable $ypbn_{ilmn}$ helps to identify whether the data point i is within the hyper-box l. Two constraints, one for the lower bound and one for the upper bound, are needed for this purpose (Eqs. (6) and (7)). Since these constraints establish a relation between continuous and binary variables, an arbitrarily large parameter, Q, is included in these constraints. The Eqs. (8) and (9) state that every data point must be assigned to a single hyper-box, l, and a single class, k, respectively. The equivalence between Eqs. (8) and (9) is given in Eq. (10); indicating that if there is a data point in the class k, then there must be a hyper-box l to represent the class k and vice versa. The existence of a hyper-box implies the assignment of that hyper-box to a class as shown in Eq. (11). If a class is represented by a hyper-box, there must be at least one data point within that hyper-box as in Eq. (12). In the same manner, if a hyper-box represents a class, there must be at least a data point within that class as given in Eq. (13). The Eq. (14) represents the condition of a data point being within the bounds of a box in attribute m. If a data point is within the bounds of all attributes of a box, then it must be in the box as shown in Eq. (15). When a data point is assigned to a class that it is not a member of, a penalty applies as indicated in Eq. (16). Finally, last constraint gives non-negativity and integrality of decision variables. By using this MILP formulation, a training set can be studied and the bounds of the classes are determined for a data classification problem.

Testing Problem Formulation

The testing problem for multi-class data classification using hyper-boxes is straight forward. If a new data point whose membership to a class is not known arrives, it is necessary to assign this data point to one of the classes. There are three possibilities for a new data point when determining its class:

i. the new data point is within the boundaries of a single hyper-box,
ii. the new data point is within the boundaries of more than one hyper-box,
iii. the new data point is not enclosed in any of the hyper-boxes determined in the training problem.

When the first possibility is realized for the new data point, the classification is made by directly assigning this data to the class that was represented by the hyper-box enclosing the data point. Since eliminating the shared areas between the constructed hyper-boxes introduces new constraints into the training problem that makes it computationally very difficult to be solved, there exists a possibility for a new data point to be within the boundaries of more than one hyper-box. In that case, the data point is assigned to the classes of the hyper-boxes that enclose this specific data point. The proportion of the number of correct classes to the number of total assigned classes to that data point determines the effect of that data point to the accuracy of the model. In the case when the third possibility applies, the assignment of the new data point to a class requires some analysis. If the data point is within the lower and upper bounds of all but not one of the attributes (i. e., m') defining the box, then the shortest distance between the new point and the hyper-box is calculated using the minimum distance between hyper-planes defining the hyper-box and the new data point. The minimum distance between the new data point j and the hyper-box is calculated using Eq. (18) considering the fact that the minimum distance is given by the normal of the hyper-plane.

$$\min_{l,m,n} \left\{ \left| \left(a_{jm} - X_{lmn} \right) \right| \right\} \tag{18}$$

When the data point is between the bounds of smaller than or equal to M-2 attributes, then the smallest distance between the point and the hyper-box is obtained by calculating the minimum distance between edges of the hyper-box and the new point. An edge is a finite segment consists of the points of a line that are between two extreme points X_{lmn} and $X_{lm'n}$. The data point j is represented by the vector $\vec{A}_j$ which is composed of a_{jm} values and $\vec{P0}_{lmn}$ and $\vec{P1}_{lmn}$ are the vector forms of two extreme points. The minimum distance between the new data point j and one of the segments of the

hyper-box determined by two extreme points is calculated using Eq. (25) where $(\cdot)$ indicates the dot product of the matrices in Eq. (22) and (23).

$$\vec{W}_{jlmn} = \vec{A}_j - \vec{P0}_{lmn} \tag{19}$$

$$\vec{V}_{jlmn} = \vec{P1}_{lmn} - \vec{P0}_{lmn} \tag{20}$$

$$C1_{jlmn} = \frac{(W_{jlmn} \cdot V_{jlmn})}{\|W_{jlmn}\| \, \|V_{jlmn}\|} \tag{21}$$

$$C2_{jlmn} = \frac{(V_{jlmn} \cdot V_{jlmn})}{\|V_{jlmn}\| \, \|V_{jlmn}\|} \tag{22}$$

$$b_{jlmn} = \frac{C1_{jlmn}}{C2_{jlmn}} \tag{23}$$

$$Pb_{jlmn} = P0_{jlmn} + b_{jlmn} V_{jlmn} \tag{24}$$

$$\min_{l,n} \left\{ \sqrt{\sum_m (a_{jm} - p_{b_{jlmn}})^2} \right\} \tag{25}$$

When data point is not within the lower and upper bounds of any attributes defining the box, then the shortest distance between the new point and the hyper-box is calculated using the minimum distance between extreme points of the hyper-box and the new data. The minimum distance between the new data point j and one of the extreme points of the hyper-box is calculated using Eq. (26).

$$\min_{l,n} \left\{ \sqrt{\sum_m (a_{jm} - X_{lmn})^2} \right\} \tag{26}$$

The following algorithm assign a new data point j with attribute values a_{jm} to class k:

Step 0: Initialize $inAtt(l,m) = 0$.

Step 1: For each l and m, if

$$X_{lmn} \le a_{jm} \le X_{lmn'} \quad \forall n = lo, n' = up \tag{27}$$

Set $inAtt(l,m) = inAtt(l,m) + 1$.

Step 2: If $inAtt(l,m) = M$, then go to Step 3. Otherwise, continue. If $inAtt(l,m) \le M - 1$, then go to Step 4.

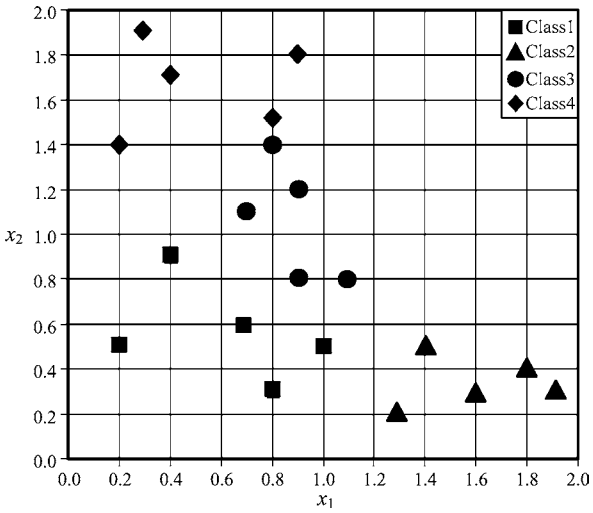
Step 3: Assign the new data point to class k where ybc_{lk} is equal to 1 for the hyper-box in Step 2. Stop.

Step 4: Calculate the minimum given by Eq. (18) and set the minimum as *min1(l)*. Calculate the minimum given by Eq. (25) and set the minimum as *min2(l)*. Calculate the minimum given by Eq. (26) and set the minimum as *min3(l)*. Select the minimum between *min1(l)*, *min2(l)* and *min3(l)* to determine the hyper-box l that is closest to the new data point j. Assign the new data point to class k where ybc_{lk} is equal to 1 for the hyper-box l. Stop.

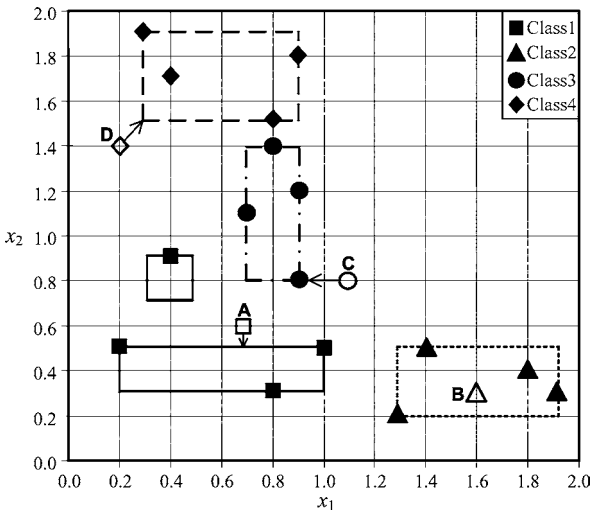
Application

We applied the mathematical programming method on a set of 16 data points in 4 different classes given in Fig. 2. The data points can be represented by two attributes, 1 and 2.

There are a total of 20 data points; 16 of these points were used in training and 4 of them used in testing. The training problem classified the data into 4 four classes using 5 hyper-boxes as shown in Fig. 3. It is interesting to note that Class1 requires two hyper-boxes while the other classes are represented with a single hyper-box only. The reason for having two hyper-boxes for Class1 is due to the fact that a single hyper-box for this



Multi-Class Data Classification via Mixed-Integer Optimization, Figure 2
Data points in the illustrative example and their graphical representation



Multi-Class Data Classification via Mixed-Integer Optimization, Figure 3
Hyper-boxes that classify the data points in the illustrative example

class would include one of the data points that belong to Class3. In order to eliminate inconsistencies in training data set, the method included one more box for Class1.

After the training is successfully completed, the test data is processed to assign them to hyper-boxes that classify the data perfectly. The assignment of the test data point B to Class2 is straightforward since it is included in the hyper-box that classifies Class2 (i. e., $inAtt(l,m) = M$ for this data point). The test data in Class1 is assigned to one of the hyper-boxes that classify Class1. Similarly, the test data in Class3 is also assigned to the hyper-box that classifies Class3. Since the test data in these classes are included within the bounds of one of the two attributes, the minimum distance is calculated as the normal to the closest hyper-plane to these data points. In the case of data point that belongs to Class4, it is assigned to its correct class since the closest extreme point of a hyper-box classifies Class4. This extreme point of the hyper-box 5 classifying Class4 is given by $(X_{5,1,lo}, X_{5,2,lo})$. The test problem also classified the data points with 100% accuracy as shown in Fig. 3.

This illustrative example is also tested by different data classification models existing in the literature in order to compare the results and to measure the performance of the proposed model. Table 1 shows the examined models and their outcomes for this small illustrative example.

Multi-Class Data Classification via Mixed-Integer Optimization, Table 1
Comparison of different classification models for the illustrative example

Classification Model	Prediction Accuracy	Misclassified Sample(s)
Neural Networks[a]	75%	A
Support Vector Machines[b]	75%	D
Bayesian Classifier[c]	75%	C
K-nearest Neighbor Classifier[c]	75%	A
Statistical Regression Classifiers[c]	75%	C
Decision Tree Classifier[c]	50%	A, C
MILP approach	100%	–

[a] iDA implementation in MS Excel [9] [b] SVM implementation in Matlab [3] [c] WEKA [20]

Neural Networks, Support Vector Machines, Bayesian, K-nearest Neighbor and Statistical Regression classifiers have only one misclassified instance which leads to 75% accuracy value as shown in Table 1. Neural Networks and K-nearest Neighbor classifier predicts the class of test sample A as Class3. Support Vector Machine method misclassifies test sample D and assigns it to Class1 while Bayesian and Statistical Regression classifier classifies test sample C as belonging to Class2. On the other hand, Decision Tree classifier gives the lowest accuracy value (50%) with two misclassifications. Sample A and sample C is classified as Class3 and Class2, respectively. Consequently, MILP approach in this thesis classifies all of the test samples accurately and achieves 100% accuracy. As a result, the MILP approach performs better than other data classification methods that are listed in Table 1 for the illustrative example. The accuracy of the MILP approach is tested on IRIS dataset and protein folding type dataset. The results indicate that the MILP approach has better accuracy than other methods on these datasets [17,18].

Conclusion

Multi-class data classification problem can be very effectively modeled as an MILP problem. One of the

most important characteristics of the MILP approach is allowing the use of hyper-boxes for defining the boundaries of the classes that enclose all or some of the points in that set. In other words, if necessary, more than one hyper-box is constructed for a specific class through the training part studies. Moreover, well-construction of the boundaries of each class provides the lack of misclassifications in the training set and indirectly improves the accuracy of the model. The model does not require the underlying distribution of the training data set and learns from the training set in a reasonable time. With only one parameter (c: the penalty parameter to minimize the total number of hyper-boxes), the suggested model is simple and very effective. Furthermore, the proposed model can be used for both binary and multi-class data classification problems without any modifications. Hence, the performance of the model does not depend on the class related changes.

References

1. Adem J, Gochet W (2006) Mathematical programming based heuristics for improving LP-generated classifiers for the multi-class supervised classification problem. Eur J Oper Res 168:181–199
2. Bajgier SM, Hill AV (1982) An experimental comparison of statistical and linear programming approaches to the discriminant problem. Decis Sci 13:604–618
3. Cawley G (2000) Matlab Support Vector Machine Toolbox. School of Information Systems, University of East Anglia
4. Chen MS, Han J, Yu PS (1996) Data Mining: An overview from a database perspective. IEEE Trans Knowl Data Eng 8:866–883
5. Cortes C, Vapnik V (1995) Support vector network. Mach Learn 20:273–297
6. Edelstein H (2003) Building Profitable Customer Relationships with Data Mining. Two Crows Corporation, Maryland
7. Erenguc SS, Koehler GJ (1990) Survey of mathematical programming models and experimental results for linear discriminant analysis. Manag Decis Econ 11:215–225
8. Gehrlein WV (1986) General mathematical programming formulations for the statistical classification problem. Oper Res Lett 5(6):299–304
9. iData Analyzer, Version 2.0, Information Acumen Corporation.
10. Jagota A (2000) Data Analysis and Classification for Bioinformatics, Bay Press, New York
11. Joachimsthaler EA, Stam A (1990) Mathematical programming approaches for the classification problem in two-group discriminant analysis. Multivar Behav Res 25:427–454
12. Koehler GJ (1990) Considerations for mathematical programming models in discriminant analysis. Manag Decis Econ 11:227–234
13. Littschwager JM, Wang C (1978) Integer programming solution of a classification problem. Manag Sci 24(14):1515–1525
14. Roiger RJ, Geatz MW (2003) Data Mining – A Tutorial Based Primer. Addison Wesley Press, Boston
15. Stam A, Joachimsthaler EA (1990) A comparison of a robust mixed-integer approach to existing methods for establishing classification rules for the discriminant problem. Eur J Oper Res 46(1):113–122
16. Tax D, Duin R (2002) Using two-class classifiers for multi class classification, Proc 16th Int Conference Pattern Recogn, Quebec City, Canada, vol II, IEEE Computers Society Press, Los Alamitos, pp 124–127
17. Turkay M, Uney F, Yilmaz O (2005) Prediction of Folding type of Proteins Using Mixed-Integer Linear Programming. In: Puigjaner L, Espuna A (eds) Computer-Aided Chem. Eng. vol 20A: ESCAPE-15. Elsevier, Amsterdam, pp 523–528
18. Uney F, Turkay M (2006) A Mixed-Integer Programming Approach to Multi-Class Data Classification Problem. Eur J Oper Res 173(3):910–920
19. Vapnik VN (1998) Statistical Learning Theory. Wiley, New York
20. WEKA (Waikato Environment for Knowledge Analysis) (199–2005) Version 3.4.5, University of Waikato, New Zealand
21. Weiss SM, Kulikowski CA (1991) Computer systems that learn: classification and prediction methods from statistics, neural networks, machine learning and expert systems. Morgan Kaufmann, San Mateo, CA

Multicommodity Flow Problems

Cynthia Barnhart[1], Niranjan Krishnan[1], Pamela H. Vance[2]

[1] Center for Transportation Studies, Massachusetts Institute Technol., Cambridge, USA

[2] Goizueta Business School, Emory University, Atlanta, USA

MSC2000: 90C35

Article Outline

Keywords
MCF Example Applications
  IMCF Example Applications
Formulations
LP Solution Methods

IP Solution Methods
See also
References

Keywords

Multicommodity network flows; Column generation; Decomposition; Transportation

Linear multicommodity flow problems (MCF) are linear programs (LPs) that can be characterized by a set of commodities and an underlying network. A *commodity* is a good that must be transported from one or more origin nodes to one or more destination nodes in the network. In practice these commodities might be telephone calls in a telecommunications network, packages in a distribution network, or airplanes in an airline flight network. Each commodity has a unique set of characteristics and the commodities are not interchangeable. That is, you cannot satisfy demand for one commodity with another commodity. The objective of the MCF problem is to flow the commodities through the network at minimum cost without exceeding arc capacities. A comprehensive survey of linear multicommodity flow models and solution procedures are presented in [2].

Integer multicommodity flow (IMCF) problems, a constrained version of the linear multicommodity flow problem in which flow of a commodity (specified in this case by an origin-destination pair) may use only one path from origin to destination.

MCF and IMCF problems are prevalent in a number of application contexts, including transportation, communication and production.

MCF Example Applications

- *Routing* vehicles in traffic networks (*dynamic traffic assignment*). This involves the determination of minimum delay routes for vehicles from their origins to their respective destinations over the traffic network. The allowable congestion levels determine the arc capacities. Alternatively, there are no capacities but the cost on an arc is a function of the amount of flow on the arc. In the former case, the objective function is linear while in the latter it is nonlinear.
- *Distribution systems planning*. In this problem there are different products (or, commodities) produced at several plants with known production capacities. Each commodity has a certain demand in each customer zone. The demand is satisfied by shipping via regional distribution centers with finite storage capacities. A.M. Geoffrion and G.W. Graves [28] model this problem of routing the commodities from the manufacturing plants to the customer zones through the distribution centers as a MCF problem.
- *Import* and *export models*. One of the factors that may affect export is handling capacity at ports. D. Barnett, J. Binkley and B. McCarl [8] use a MCF model to analyze the effect of US port capacities on the export of wheat, corn and soybean.
- Optimization of *freight operations*. T. Crainic, J.A. Ferland and J.M. Rousseau [20] develop a MCF-based routing and scheduling optimization model that considers the planning issues for the railroad industry. More recently, H.N. Newton [48] and C. Barnhart, H. Jin and P.H. Vance [13] study the railroad blocking problem using multicommodity based formulations.
- Freight Assignment in the *Less-than-Truckload* (LTL) industry. An LTL carrier has to consolidate many shipments to make economic use of the vehicles. This requires the establishment of a large number of terminals to sort freight. Trucking companies use forecasted demands to define routes for each vehicle to carry freight to and from the terminals. Once the routes are fixed, the problem is to deliver all the shipments with minimum total service time or cost. This problem is formulated as a MCF problem in [17] and [24].
- *Express Shipment Delivery*. D. Kim [40] models the shipment delivery problem faced by express carriers like Federal Express, United States Postal Service, United Parcel Service, etc. as a MCF problem on a network in space and time.
- Routing messages in a *telecommunications* or computer network. The network consists of transmission lines. Each message request is a commodity. The problem is to route the messages from origins to the respective destinations at a minimum cost. T.L. Magnanti et al. [42] and others provide MCF-based formulations for this problem.

- Long-term *hydro-generation* optimization. The task in this case is to determine the amount of hydro-generation at a reservoir in an interval of time, that minimizes the expected cost of power generation over a period of time, divided into several intervals. N. Nabonna [47] showed that this problem can be modeled as a MCF problem with inflows given as probabilistic density functions.
- *Forest management*. For each planning period, forest managers have to make decisions concerning the land areas to be harvested, the volume of timber to be harvested from these areas, the land areas to be developed for recreation and the road network to be built and maintained in order to support both the timber haulers and recreationists. This problem has been formulated as a MCF problem in [33].
- *Street planning*. L.R. Foulds [26] introduced this problem and modeled it as a MCF problem. The objective is to identify a set of two-way streets such that making these streets one-way minimizes the total congestion cost in the network.
- *Spatial price equilibrium* (SPE) problem. This problem requires modeling consumer flows within a general network. The SPE problem determines the optimum levels of production and consumption at each market and the optimal flows satisfy the equilibrium property. R.S. Segall [59] models and solves the SPE problem as a MCF problem.

For a more comprehensive description of MCF applications, see [2,37,57].

IMCF Example Applications

- *Airline fleet assignment*. Given a time table of flight arrivals and departures, the expected demand on the flights and a set of aircraft, the objective is to arrive at a minimum cost assignment of aircraft to the flights. This problem has been extensively studied in [1,31].
- *Airline crew scheduling*. This problem deals with the minimum cost scheduling of crews. Factors such as hours of work limitations and Federal Aviation Administration regulations must be taken into account while solving the problem. For an in-depth study see [5,14].
- *Airline maintenance routing problems* require that single aircraft be routed such that maintenance requirements are satisfied and each flight is assigned to exactly one aircraft. This problem has been studied in [10,19,25].
- *Bandwidth packing problems* require that bandwidth be allocated in telecommunications networks to maximize total revenue. The demands, or calls, on the networks are the commodities and the objective is to route the calls from their origin to their destination. In the case of video teleconferencing, since call splitting is not allowed, each call must be routed on exactly one network path. This IMCF problem is described in [49].
- *Package flow problems*, such as those arising in express package delivery operations, require that shipments, each with a specific origin and destination, be routed over a transportation network. Each set of packages with a common origin-destination pair can be considered as a commodity and often, to facilitate operations and ensure customer satisfaction, must be assigned to a single network path. These problems are cast as IMCF problems in [12].

Formulations

Multicommodity flow problems can be modeled in a number of ways depending how one defines a commodity. There are three major options: a commodity may originate at a subset of nodes in the network and be destined for another subset of nodes, or it may originate at a single node and be destined for a subset of the nodes, or finally it may originate at a single node and be destined for a single node. K.L. Jones et al. [34] present models for each of these different cases. In the interest of space, we will only consider models for the last case. The other cases can also be modeled using variants of the models presented here.

We present two different formulations of the MCF problem: the *node-arc* or *conventional* formulation and the *path* or *column generation formulation*. The MCF is defined over the network G comprised of node set N and arc set A. MCF contains decision variables x, where x_{ij}^k is the fraction of the total quantity (denoted q^k) of commodity k assigned to arc ij. In the IMCF problem these variables are restricted to be binary. The cost of assigning commodity k in its entirety to arc ij equals q^k times the unit flow cost for arc ij, denoted c_{ij}^k. Arc ij has capacity d_{ij}, for all $ij \in A$. Node i has supply of

commodity k, denoted b_i^k, equal to 1 if i is the origin node for k, equal to -1 if i is the destination node for k, and equal to 0 otherwise.

The node-arc MCF formulation is:

$$\text{minimize} \sum_{k\in K}\sum_{ij\in A} c_{ij}^k q^k x_{ij}^k \tag{1}$$

such that

$$\sum_{ij\in A} x_{ij}^k - \sum_{ji\in A} x_{ji}^k = b_i^k, \quad \forall i \in N, \forall k \in K, \tag{2}$$

$$\sum_{k\in K} q^k x_{ij}^k \le d_{ij}, \quad \forall ij \in A, \tag{3}$$

$$x_{ij}^k \ge 0, \quad \forall ij \in A, \quad \forall k \in K. \tag{4}$$

Note that without restricting generality of the problem, we model the arc flow variables x having values between 0 and 1. To do this, we scale the demand for each commodity to 1 and accordingly adjust the coefficients in the objective function (1) and in constraints (3). Also note the block-angular structure of this model. The conservation of flow constraints (2) form nonoverlapping blocks, one for each commodity. Only the arc capacity constraints (3) link the values of the flow variables of different commodities.

To contrast, the path-based or column generation MCF formulation has fewer constraints, and far more variables. Again, the underlying network G is comprised of node set N and arc set A, with q^k representing the quantity of commodity k. $P(k)$ represents the set of all origin-destination paths in G for k, for all $k \in K$. In the column generation model, the binary decision variables are denoted y_p^k, where y_p^k is the fraction of the total flow of commodity k assigned to path $p \in P(k)$. The cost of assigning commodity k in its entirety to path p equals q^k times the unit flow cost for path p, denoted c_p^k. c_p^k represents the sum of the c_{ij}^k costs for all arcs ij contained in path p. As before, arc ij has capacity d_{ij}, for all $ij \in A$. Finally, δ_{ij}^p is equal to 1 if arc ij is contained in path $p \in P(k)$, for all $k \in K$; and is equal to 0 otherwise.

The path or column generation IMCF formulation is then:

$$\text{minimize} \sum_{k\in K}\sum_{p\in P(k)} c_p^k q^k y_p^k \tag{5}$$

such that

$$\sum_{k\in K}\sum_{p\in P(k)} q^k y_p^k \delta_{ij}^p \le d_{ij}, \quad \forall ij \in A, \tag{6}$$

$$\sum_{p\in P(k)} y_p^k = 1, \quad \forall k \in K, \tag{7}$$

$$y_p^k \ge 0, \quad \forall p \in P(k), \quad \forall k \in K. \tag{8}$$

LP Solution Methods

Comprehensive surveys of the available multicommodity network flow solution techniques are provided in [6,37]. Descriptions of these approaches are also provided in [2,38].

Price-directive decomposition techniques use the path-based MCF model. To limit the number of variables considered in finding an optimal solution, *column generation* techniques are used. Further details of price-directive *decomposition* and column generation are provided in [18,22,41,45,61].

Resource-directive decomposition techniques attempt to solve MCF problems by allocating arc capacity by commodity and solving the resulting decoupled minimum cost flow problems for each commodity. Additional descriptions of this technique can be found in [27,30,35,37,39,41,52,60,61].

Computational comparisons of the performance of price- and resource-directive decomposition methods can be found in [3,4]. A. Ali, R.V. Helgason, J.L. Kennington, and H. Lall [4] report that specialized decomposition codes can be expected to run from three to ten times faster than a general linear programming package. Furthermore, A.A. Assad [7] reports that resource-directive algorithms converge quickly for small problems but are outperformed by the price-directive method for larger MCF problems.

G. Saviozzi [56] uses *subgradient* techniques on the *Lagrangian relaxation* of the bundle constraints and proposes a method of arriving at an advanced starting basis for the minimum cost multicommodity flow problem.

Partitioning methods specialize the simplex method by partitioning the current basis to exploit the underlying network structure. Experiences with primal partitioning techniques have been reported in [24,32,

36,43,51,53,54,55], among others. J.B. Rosen [53] develops a partitioning strategy for angular problems. J.K. Hartman and L.S. Lasdon [32] develop a generalized upper bounding algorithm for multicommodity network flow problems in which the special structure of the MCF problem is exploited. Their primal partitioning procedure, a specialization of the generalized upper bounding procedure developed by G.B. Dantzig and R.M. Van Slyke [21], involves the determination at each iteration of the inverse of a basis containing only one row for each saturated arc. Similarly, C.J. McCallum [44] developed a generalized upper bounding algorithm for a communications network planning problem. All of these procedures exploit the block-diagonal problem structure and perform all steps of the simplex method on a reduced working basis of dimension m, where m represents the size of set A.

Interior point methods and parallel computing techniques have also been applied to MCF problems. Interior point methods provide polynomial time algorithms for the MCF problems. The best time bound is due to P.M. Vaidya [62]. G.L. Schultz and R.R. Meyer [58] provide an interior point method with massive parallel computing to solve multicommodity flow problems.

Development of new heuristic procedures for MCF problems include the primal and dual-ascent heuristics described in [17] and [9], respectively. A. Gersht and A. Shulman [29] use a barrier-penalty method to find nearly optimal solutions for multicommodity problems, while R. Schneur [62] describes a scaling algorithm to determine nearly feasible MCF solutions.

Recently, price-directive decomposition or column generation approaches, such as those presented in [2,11,23,34] have been the most extensively used method for solving large versions of the linear MCF problem. The general idea of column generation is that optimal solutions to large LP's can be obtained without explicitly including all columns (i. e., variables) in the constraint matrix (called the *Master Problem* or MP). In fact, only a very small subset of all columns will be in an optimal solution and all other (nonbasic) columns can be ignored. In a minimization problem, this implies that all columns with positive reduced cost can be ignored. The multicommodity flow column generation strategy, then, is:

0) RMP Construction. Include a subset of columns in a restricted MP, called the *Restricted Master Problem*, or RMP;
1) RMP Solution. Solve the RMP LP;
2) Pricing Problem Solution. Use the dual variables obtained in solving the RMP to solve the pricing problem. The pricing problem either identifies one or more columns with negative reduced cost (i. e., columns that price out) or determines that no such column exists.
3) Optimality Test. If one or more columns price out, add the columns (or a subset of them) to the RMP and return to Step 1; otherwise stop, the MP is solved.

For any RMP in Step 1, let $-\pi_{ij}$ represent the nonnegative dual variables associated with constraints (6) and σ^k represent the unrestricted dual variables associated with constraints (7). Since c_p^k can be represented as $\sum_{ij\in A} c_{ij}^k\, \delta_{ij}^p$, the reduced cost of column p for commodity k, denoted $\bar{c}_p^k q^k$, is:

$$\bar{c}_p^k q^k = \sum_{ij\in A} q^k (c_{ij}^k + \pi_{ij}) \delta_{ij}^p - \sigma^k, \quad \forall p \in P(k), \forall k \in K. \quad (9)$$

For each RMP solution generated in Step 1, the pricing problem in Step 2 can be solved efficiently. Columns that price out can be identified by solving one shortest path problem for each commodity $k \in K$ over a network with arc costs equal to $c_{ij}^k + \pi_{ij}$, for each $ij \in A$. Let $p*$ represent a resulting shortest path $p*$ for commodity k. Then, if for all $k \in K$,

$$c_{p*}^k q^k \geq 0,$$

the MP is solved. Otherwise, the MP is not solved and, for each $k \in K$ with

$$c_{p*}^k q^k < 0,$$

path $p* \in P(k)$ is added to the RMP in Step 3.

IP Solution Methods

The ability to solve large MCF LP's enables the solution of large IMCF problems. Successful approaches for solving large IMCF problems use the path-based or

column generation formulation of the problem. Column generation IP's can be solved to optimality using a procedure known as *branch and price*, detailed in [15,23,64]. Branch and price, a generalization of branch and bound with LP relaxations, allows column generation to be applied at each node of the branch and bound tree. Branching occurs when no columns price out to enter the basis and the LP solution does not satisfy the integrality conditions.

Applying a standard branch and bound procedure to the final restricted master problem with its existing columns will not guarantee an optimal (or feasible) solution. After the branching decision modifies RMP, it may be the case that there exists a column for MP that prices out favorably, but is not present in RMP. Therefore, to find an optimal solution we must maintain the ability to solve the pricing problem after branching. The importance of generating columns after the initial LP has been solved is demonstrated for airline crew scheduling applications in [63]. Although they were unable to find even feasible IP solutions using just the columns generated to solve the initial LP relaxation, they were able to find quality solutions using a branch and price approach for crew scheduling problems in which they generated additional columns whenever the LP bound at a node exceeded a preset IP target objective value.

The difficulty of performing column generation with branch and bound is that conventional integer programming branching on variables may not be effective because fixing variables can destroy the structure of the pricing problem. For the multicommodity flow application, a branching rule is needed that ensures that the pricing problem for the LP with the branching decisions included can be solved efficiently with a shortest path procedure. To illustrate, consider branching based on variable dichotomy in which one branch forces commodity k to be assigned to path p, i. e., $y_p^k = 1$, and the other branch does not allow commodity k to use path p, i. e., $y_p^k = 0$. The first branch is easy to enforce since no additional paths need to be generated once k is assigned to path p. The latter branch, however, cannot be enforced if the pricing problem is solved as a shortest path problem. There is no guarantee that the solution to the shortest path problem is not path p. In fact, it is likely that the shortest path for k is indeed path p. As a result, to enforce a branching decision, the pricing problem solution must be achieved using a *next shortest path procedure*. In general, for a subproblem, involving a set of a branching decisions, the pricing problem solution must be achieved using a kth shortest path procedure.

The key to developing a branch and price procedure is to identify a branching rule that eliminates the current fractional solution without compromising the tractability of the pricing problem. In general, J. Desrosiers et al [23] argue this can be achieved by basing branching rules on variables in the original formulation, and not on variables in the column generation formulation. This means that branching rules should be based on the arc flow variables x_{ij}^k from the node-arc formulation of the problem. Barnhart et al. [15] develop branching rules for a number of different master problem structures. They also survey specialized algorithms that have appeared in the literature for a broad range of applications.

M. Parker and J. Ryan [49] present a branch and price algorithm for the bandwidth packing problem. in which the objective is to choose which of a set of commodities to send in order to maximize revenue. They use a path-based formulation. Their branching scheme selects a fractional path and creates a number of new subproblems equal to the length of the path (measured in the number of arcs it contains) plus one. On one branch, the path is fixed into the solution and on each other branch, one of the arcs on the path is forbidden. To limit time spent searching the tree they use a dynamic optimality tolerance. They report the solution of 14 problems with as many as 93 commodities on networks with up to 29 nodes and 42 arcs. All but two of the instances are solved to within 95% of optimality.

K. Ziarati et al. [16] consider the problem of assigning railway locomotives to trains. They model the problem as an integer multicommodity flow problem with side constraints and solve using a Dantzig–Wolfe decomposition technique, where subproblems are formulated as constrained or unconstrained shortest path problems.

P. Raghavan and C.D. Thompson [50] illustrate the use of randomized algorithms to solve some integer multicommodity flow problems. They use randomized rounding procedures that give provably good solutions in the sense that they have a very high probability of being close to optimality.

Barnhart et al. [12] present a *branch and price and cut* algorithm for general IMCF problems where each commodity is represented by an origin-destination pair and flow volume. *Branch and cut*, another variant of branch and bound, allows valid inequalities, or cuts, to be added throughout the branch and bound tree. *Branch and price and cut* combines column and row generation to yield very strong LP relaxations at nodes of the branch and bound tree.

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Nonoriented Multicommodity Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Abara J (1989) Applying integer linear programming to the fleet assignment problem. Interfaces 19:20–28
2. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs
3. Ali AI, Barnett D, Farhangian K, Kennington JL, Patty B, Shetty B, McCarl B, Wong P (1984) Multicommodity network problems: Applications and computations. IIE Trans 16:127–134
4. Ali A, Helgason R, Kennington J, Lall H (1980) Computational comparison among three multicommodity network flow algorithms. Oper Res 28:995–1000
5. Anbil R, Gelman E, Patty B, Tanga R (1991) Recent advances in crew-pairing optimization at American Airlines. Interfaces 21:62–64
6. Assad AA (1978) Multicommoditynetwork flows - A survey. Networks 8:37–91
7. Assad AA (1980) Solving linear multicommodity flow problems. Proc IEEE Internat Conf Circuits and Computers 1, pp 157–161
8. Barnett D, Binkley J, McCarl B (1982) The effects of US port capacity constraints on national and world grain shipments. Techn Paper, Purdue Univ
9. Barnhart C (1993) Dual-ascent methods for large-scale multi-commodity flow problems. Naval Res Logist 40:305–324
10. Barnhart C, Boland NL, Clarke LW, Johnson EL, Nemhauser GL, Shenoi RG (1998) Flight string models for aircraft fleeting and routing. Transport Sci 32(3):208–220, Focused Issue on Airline Optimization
11. Barnhart C, Hane CA, Johnson EL, Sigismondi G (1995) A column generation and partitioning approach for multicommodity flow problems. Telecommunication Systems 3:239–258
12. Barnhart C, Hane CA, Vance PH (2000) Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. Oper Res 48(2):318–326
13. Barnhart C, Jin H, Vance PH (1997) Railroad blocking: A network design application. Working Paper Center Transport Stud, MIT
14. Barnhart C, Johnson EL, Anbil R, Hatay L (1994) A column generation technique for the long-haul crew-assignment problem. In: Ciriani TA, Leachman RC (eds) Optimization in Industry: Math. Programming and Optimization Techniques 2. Wiley, New York, pp 7–24
15. Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWF, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. Oper Res 46(3):316–329
16. Barnhart C, Johnson EL, Nemhauser GL, Sigismondi G, Vance P (1993) Formulating a mixed integer programming problem to improve solvability. Oper Res 41:1013–1019
17. Barnhart C, Sheffi Y (1993) A network-based primal-dual heuristic for the solution of multi-commodity network flow problems. Transport Sci 27:102–117
18. Bazaraa MS, Jarvis JJ (1977) Linear programming and network flows. Wiley, New York
19. Clarke LW, Johnson GL, Nemhauser GL, Zhu Z (1997) The aircraft rotationproblem. Ann Oper Res: Math Industr Systems II 69:33–46
20. Crainic T, Ferland JA, Rousseau JM (1984) A tactical planning model for Rail freight transportation. Transport Sci 18:165–184
21. Dantzig GB, Van Slyke RM (1967) Generalized upper bounding techniques. J Comput Syst Sci 1:213–226
22. Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8:108–111
23. Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995) Time constrained routing and scheduling. In: Ball ME, Magnanti TL, Monma C, Nemhauser GL (eds) Handbook Oper. Res. and Management Sci. 8 Elsevier, Amsterdam, pp 35–139

24. Farvolden JM, Powell WB, Lustig IJ (1993) A primal partitioning solution for the arc-chain formulation of a multicommodity network flow problem. Oper Res 4(4):669–693
25. Feo TA, Bard JF (1989) Flight scheduling and maintenence base planning. Managem Sci 35:1415–1432
26. Foulds LR (1981) A multicommodity flow network design problem. Transport Res B 15:273–283
27. Geoffrion AM (1970) Primal resource-directive approaches for optimizing non-linear decomposable systems. Oper Res 18:375–403
28. Geoffrion AM, Graves GW (1974) Multicommodity distribution systems design by Bender's decomposition. Managem Sci 20:822–844
29. Gersht A, Shulman A (1987) A new algorithm for the solution of the minimum cost multicommodity flow problem. Proc 26th IEEE Conf Decision and Control, pp 748–758
30. Grinold RC (1972) Steepest ascent for large scale linear program. SIAM Rev 14:447–464
31. Hane CA, Barnhart C, Johnson EL, Marsten RE, Nemhauser GL, Sigismondi G (1995) The fleet assignment problem: solving a large-scale integer program. Math Program 70:211–232
32. Hartman JK, Lasdon LS (1972) A generalized upper bounding algorithm for multicommodity network flow problems. Networks 1:333–354
33. Helgason R, Kennington J, Wong P (1981) An application of network programming for national forest planning. Techn Report, Dept Oper Res Southern Methodist Univ, Dallas
34. Jones KL, Lustig IJ, Farvolden JM, Powell WB (1993) Multicommodity network flows: The impact of formulation on decomposition. Math Program 62:95–117
35. Karkazis J, Boffey TB (1981) A subgradient based optimal solution method for the multicommodity problem. In: Burkard RE, Ellinger T (eds) Methods Oper Res 40. Anton Hain Verlag, pp 339–344
36. Kennington JL (1977) Solving multicommodity transportation problems using a primal partitioning simplex technique. Naval Res Logist Quart 24:309–325
37. Kennington JL (1978) A survey of linear cost network flows. Oper Res 26:209–236
38. Kennington JL, Helgason RV (1980) Algorithms for network programming. Wiley, New York
39. Kennington JL, Shalaby M (1977) An effective subgradient procedure for minimal cost multicommodity flow problems. Managem Sci 23:994–1004
40. Kim D (1997) Large scale transportation service network design: Models, algorithms and applications. PhD Thesis, Dept Civil Engin, MIT
41. Lasdon L (1970) Optimization theory for large systems. MacMillan, New York
42. Magnanti TL, Mirchandani P, Vachani R (1995) Modeling and solving the two-facility capacitated network loading problem. Oper Res 43(1)
43. Maier SF (1974) A compact inverse scheme applied to a multicommodity network with resource constraints. In: Cottle R, Krarup J (eds) Optimization Methods for Resource Allocation. English Univ Press, London, pp 179–203
44. McCallum CJ (1977) A generalized upper bounding approach to a commununications network planning problem. Networks 7:1–23
45. Minoux M (1986) Mathematical programming: Theory and algorithms. Wiley, New York
46. Moore E (1957) The shortest path through a maze. In: Proc Internat Symposium on the Theory of Switching, Harvard Univ Press, pp 282–292
47. Nabonna N (1993) Multicommodity network flow model for long term hydro-generation optimization. IEEE Trans Power Systems 8:395–404
48. Newton HN (1996) Network design under budget constraints with application to the railroad blocking problem. PhD Thesis, Auburn Univ, Alabama
49. Parker M, Ryan J (1994) A column generation algorithm for bandwidth packing. Telecommunication Systems 2:185–195
50. Raghavan P, Thompson CD (1987) Randomized rounding: A technique for provably good algorithms and algorithmic proofs. Combinatorica 4:365–374
51. Ritter K (1967) A decomposition method for linear programming problems with coupling constraints and variables. Techn Report, Math Res Center Univ Wisconsin 739
52. Robacker JT (1956) Notes on linear programming: Part XXXVII concerning multicommodity networks. Techn Report, The Rand Corp RM-1799
53. Rosen JB (1964) Primal partition programming for block diagonal matrices. Numerische Math 6:250–260
54. Saigal R (1967) Multicommodity flows in directed networks. Techn Report Oper Res Center, Univ Calif ORC 67–38
55. Sakarovitch M, Saigal R (1967) An extension of generalized upper bounding techniques for structured linear programs. SIAM J Appl Math 15:906–914
56. Saviozzi G (1986) Advanced start for the multicommodity network flow problem. Math Program Stud 26:221–224
57. Schneur R (1991) Scaling algorithms for multi-commodity flow problems and network flow problems with side constraints. PhD Diss, Massachusetts Inst Techn
58. Schultz GL, Meyer RR (1991) An interior point method for block angular optimization. SIAM J Optim 1:583–602
59. Segall RS (1995) Mathematical modeling of spatial price equilibrium for multicommodity consumer flows of large markets using variational inequalities. Appl Math Modeling 19(2):112–122
60. Shetty B, Muthukrishnan R (1990) A parallel projection for the multicommodity network model. J Oper Res Soc 41:837–842
61. Swoveland C (1971) Decomposition algorithms for the multicommodity distribution problem. Working Paper Western Management Sci Inst Univ Calif, Los Angeles 184

62. Vaidya PM (1989) Speeding up linear programming using fast matrix multiplication. Proc 30th Annual Symp Foundations of Computer Sci, pp 332–337
63. Vance PH, Barnhart C, Johnson EL, Nemhauser GL, Mahidara D, Krishna A, Rebello R (1994) Exceptions in crew planning. ORSA/TIMS Detroit, Michigan
64. Vanderbeck F, Wolsey LA (1996) An exact algorithm for IP column generation. Oper Res Lett 19:151–159
65. Zenios SA (1991) On the fine-grain decomposition of multicommodity transportation problems. SIAM J Optim 1:643–669
66. Ziarati K, Soumis F, Desrosiers J, Gelinas S, Saintonge A (1995) Locomotive assignment with heterogeneous consists at CN North America. Working Paper GERAD and École Polytechnique de Montréal

Multicriteria Decision Support Methodologies for Auditing Decisions

CHRYSOVALANTIS GAGANIS,
CONSTANTIN ZOPOUNIDIS
Financial Engineering Laboratory, Department of Production Engineering and Management, Technical University of Crete, University Campus, Chania, Greece

MSC2000: 90C90, 90C11, 91B28

Article Outline

Keywords
Introduction
Sample
  Financial Variables
  Non-financial Variables
Method
Empirical Results
Conclusions and Further Research
References

Keywords

Qualified audit reports; Multicriteria decision aid; Auditing; Classification

Introduction

References to Financial Statements Fraud (FSF) and earnings manipulation have attracted the attention of market participants, academics, and regulators all over the world especially in recent years and following the collapse of Enron. During these years there have also been several cases of financial statement fraud which have been undetected by the auditors.

Using normal audit procedures, the detection of falsified financial statements is a difficult task [18,73]). There are numerous reasons for these difficulties such as a shortage of knowledge concerning the characteristics of management fraud, the efforts of managers to deceive auditors, and difficulties in collecting, analyzing and synthesizing large quantities of data from several different sources.

Models of audit reporting have several uses (i. e. prediction, determination, bankruptcy), as described by Dopuch et al. [26]. For example, they can provide a benchmark representing the probability that an auditor would issue a modified audit report on a given company. Furthermore, these models can be imperative in an auditing system that enables the users to take preventive or corrective actions [30,57,80]).

Most of the earlier studies of FSF have used discrete choice models in which the dependent variable was dichotomous. Mutchler [66] and Levitan and Knoblett [60] used discriminant analysis, Dopuch et al. [26] and Lennox [59] used probit models, Keasey et al. [48], Bell and Tabor [9], Monroe and Teh [65], Louwers [62], DeFond et al. [25], Citron and Taffler [17], Menon and Schwartz [63], and Spathis [80] used logit models, Krishnan [55] used an ordered probit model, Spathis et al. [79,81]), and Pasiouras et al. [71] used multicriteria decision aid (UTADIS) and multivariate statistical techniques (e. g. discriminant and logit analysis), Gaganis and Pasiouras [35] used discriminant and logit models, Gaganis et al. [36] used probabilistic neural network models, Gaganis et al. [33] used nearest neighbor models, Fanning et al. [31] and Fanning and Cogger [30] used artificial neural networks, and Doumpos et al. [27] used support vector machines.

In the present study, a multicriteria approach was followed through the application of the nonparametric **M**ulti-group **H**ierarchical **DIS**crimination (MHDIS) method with the aim of developing a sorting model to detect those firms that issue FSF in Greece. The MHDIS model was compared with logit analysis in order to test its efficiency against a benchmark that has been com-

monly used in previous studies. MHDIS is not based on statistical assumptions, which often cause problems during the application of statistical methods (logit and probit analysis), and furthermore, it can easily incorporate qualitative data.

Although there have recently been a few attempts to develop models to detect falsified audit statements in Greece [15,51,79,81]. The present study differs in several respects. First, we have used a more recent and larger dataset than in previous studies, which contains more detailed information. The data used corresponds to the fiscal years 2001–2004 and covers 398 companies. Spathis et al. [79,81] and Kirkos et al. [51] examined the same random sample of 76 manufacturing companies covering the period 1997–1999, and Caramanis and Spathis [15] examined a sample of 182 companies. Second, we examined both listed and unlisted companies from the manufacturing, trade, and services sectors in contrast to Spathis et al. [79,81] and Kirkos et al. [51] who examined only manufacturing listed companies and Caramanis and Spathis [15] who considered only listed companies. Third, we used out-of-time and out-of-sample testing samples. When evaluating the classification ability of a model, it is important to ensure that it has not been over-fitted to the training (estimation) dataset. As Stein [83] mentions "a model without sufficient validation may only be a hypothesis". Previous research has shown that when classification models are used to reclassify the observations of the training sample, the classification accuracies are biased upward. Thus, it is necessary to classify a set of observations which were not used during the development of the model, using some kind of testing sample.

The rest of the paper is organized as follows: Section "Sample" describes the sample used in this study. Section "Method" describes the methodology. Section "Empirical Results" presents the empirical results, and the Sect. "Conclusions and Further Research" discusses the concluding remarks and suggests some possible future research directions.

Sample

The data used in the study consisted of financial statement information (i. e. balance sheet, income statement, auditors' opinions, and the notes to financial statements) of a sample of companies obtained from ICAP[1] database and Athens Stock Exchange (ASE). Our analysis was restricted to Greek limited (société anonyme) and limited liability companies, which are obliged by law to have their financial statements audited, and we focused on the period between 2001 and 2004.

We obtained 199 qualified cases which were distributed over various sectors[2]. The next step was to select unqualified firms. We used a pair-matching method by sector. Matching of firms is common practice when conducting classification studies in auditing as well as in other areas of finance, such as bankruptcy or acquisitions prediction (e. g. [11,34,50,58,71]). There are two primary reasons for following this procedure, which is known as choice-based sampling. The first is the lower cost of collecting data in comparison with an unmatched sample [6,46,90]). The second and most important is that a choice-based sample provides greater information content than a random sample [19,45,69]). Hence, our sample consisted of the same number of qualified and unqualified cases.

Most of the previous studies concerning the development of models to replicate (or predict) auditors' opinion used training and testing samples from the same period, or re-sampling techniques such as jackknife and bootstrap (e. g. [57,79,81]). However, as Espahbodi and Espahbodi [29] point out, the real test of a classification model and its practical usefulness is its ability to classify objects correctly in the future. The main reason, as stated by Barnes [5], is that given inflationary effects, technological and other reasons, such as accounting policies, it is not reasonable to expect financial ratios to be stable over time. To account for this *population drifting*, in the present study, we split our sample of 398 companies into two distinct samples. The training sample, used for the development of the models, consisted of 234 companies and covered the period 2001–2003. The validation sample consisted of the remaining 164 companies and used data from 2004.

[1] ICAP is the largest company providing Business Information and Consulting Services in Greece.

[2] The sample for this study consists of 164 manufacturing, 122 trade and 110 services companies.

Financial Variables

Similar to previous studies we used financial ratios as the indicators of FSF. One of the problems with the selection of suitable ratios is that not only are there many financial variables which could be potential candidates for inclusion in the model, but also that previous studies have, in general, failed to select variables that reflect theoretical models of FFS as well. We therefore selected variables that were regarded as important in previous studies, such as Albrecht and Romney [1], Palmrose [70], Dopouch et al. [26], Loebbecke et al. [61], Green [38], Stice [84], Davia et al. [23], Bell et al. [10], Schilit [75], Arens and Loebbecke [3], Beasley [7], Bologna et al. [12], Krishnan and Krishnan [54], Green and Choi [39], Hoffman [42], Hollman and Patton [43], Zimbelman [89], Laitinen and Laitinen [57], Spathis et al. [79,81], Spathis [80], Doumpos et al. [27], Gaganis et al. [33,36], Gaganis and Pasiouras [35], and Pasiouras et al. [71]. Table 1 present a full list of the variables considered in this study. There are 28 financial variables covering all aspects of the performance of the selected companies, such as liquidity, leverage, profitability, managerial activity, and annual changes in basic accounts [20].

An examination of previous research indicates that the prediction variables range between 6 and 20. Most of these studies selected the effective independent variables using a statistical method, in an attempt to reduce the number of independent variables and the impact of potential multicollinearity. There is, however, little relevant theory about the selection of independent variables for the nonlinear methods. From a practical point of view, developing a model that considers a large number of variables poses problems for the applicability of the model on a daily basis by the auditor. This is because any application of the model requires that the auditor collects all necessary data, which leads to increased time and cost for data collection and management [79]. In the present study, we used a combination of two statistical analysis methods to examine whether there was an association between our variables and auditors' opinions and hence to select our final set.

First, we used the Kruskal–Wallis non-parametric test to examine the differences between qualified and unqualified companies. Table 1 present the results of the Kruskal–Wallis test for the training sample. Only three variables: 365*Stock/Cost of Sales; Logarithm of Debt; and Inventories/Total Assets were not statistically significant at the 10% level. We then reduced the number of variables to a manageable size using factor analysis.

This approach can be used to uncover the latent structure of a set of variables by reducing the attribute space from a larger number of variables to a smaller number of factors. The factor loadings were then used to select a limited set of financial variables.

Finally, seven financial variables were selected, each being the variable with the highest loading in each factor. These were: Receivable/Sales; Current Assets/Current Liabilities; Current Assets/Total Assets; Cash/Total Assets; Profit before tax/Total Assets; Inventories/Total Assets; and Annual Change in Sales.

Non-financial Variables

In addition to the seven financial variables discussed above, six non-financial ones were also used. Dopuch et al. [26] presented a predictive model of audit opinion qualifications in which the variables with greatest predictive power were categorical ones.

Previous studies mainly dealt with the construction of bankruptcy models for making audit opinions relative to going concern (e. g. [44,52,53]). Prior research (e. g. [17,30,41,61,74,79,84]) suggests that financial distress is very important in the issuing of an audit qualification. Although most of the previous studies used Altman's z-score [2] or credit risk assessment of a rating agency [71] as a proxy of default, such an approach may not be appropriate in our case. The reason is that Altman's z-score was developed for a particular industry (i. e. manufacturing), under different economic conditions (i. e. in the 1960s) and for a specific country (i. e. USA). In the present study, we used a score (UTADISCR) estimated from the UTADIS bankruptcy prediction model of Zopounidis et al. [92]. We anticipated that the use of this measure, which indicates the likelihood of default of Greek firms over the 12 months following the date of its calculation, might provide more accurate results.

Spathis [80] found that audit qualification decision was positively associated with company litigation. In this study, the client litigation variable was coded as zero if a company had litigation in the year preceding

Multicriteria Decision Support Methodologies for Auditing Decisions, Table 1
Descriptive statistics

	Unqualified		Qualified		Kruskal–Wallis
	Mean	St.Dev	Mean	St.Dev	
INVREC / TA	0.58	0.26	0.53	0.22	2.83 ***
REC / SAL	0.31	0.18	0.52	0.59	16.84 *
CA / CL	1.52	0.60	1.00	0.69	66.07 *
CA / TA	0.75	0.22	0.65	0.22	14.48 *
CASH / CL	0.22	0.28	0.08	0.18	26.59 *
CASH / TA	0.12	0.16	0.05	0.11	13.85 *
ROA	0.19	0.12	−0.12	0.16	157.80 *
PRBT / FA	5.04	11.16	−0.62	1.29	149.09 *
INV / SAL	0.12	0.12	0.27	0.41	6.28 **
SAL / TA	1.52	0.83	1.09	0.84	22.70 *
TD / TA	0.59	0.21	0.90	0.34	66.26 *
PRBT / CL	0.37	0.25	−0.15	0.24	157.60 *
CL / TA	0.56	0.23	0.82	0.34	40.98 *
WC / TA	0.20	0.18	−0.17	0.34	80.45 *
EBIT MARGIN	0.15	0.12	−0.19	0.39	157.17 *
GP / SAL	0.36	0.18	0.13	0.21	72.76 *
GP / TA	0.52	0.33	0.16	0.22	83.78 *
(CA − ST) / CL	1.18	0.47	0.74	0.56	65.87 *
365 * AREC / SAL	126.70	66.49	238.46	330.86	21.81 *
365 * ST / CS	73.29	79.92	113.16	163.22	0.41
SAL / EQ	7.36	10.81	0.05	32.71	31.89 *
SAL / TD	2.84	1.63	1.24	0.87	78.28 *
365 * AP / SAL	139.63	550.32	155.07	249.57	65.98 *
TACH	0.22	0.46	0.10	0.39	11.76 *
SALCH	0.15	0.39	0.10	0.44	5.79 **
LOGTA	7.45	0.53	7.28	0.46	7.50 *
LOGDEPT	7.19	0.54	7.19	0.45	0.05
INV / TA	0.16	0.13	0.16	0.15	0
UTADISCR	0.74	0.10	0.49	0.12	126.51**

Notes: INVREC / TA: (Inventories + Receivable) / Total Assets, REC / SAL: Receivable / Sales, CA / CL: Current Assets / Current Liabilities, CA / TA: Current Assets / Total Assets, CASH / CL: Cash / Current Liabilities, CASH / TA: Cash / Total Assets, ROA: Profit before tax × 100 / Total assets, PRBT / FA: Profit (Loss) before tax × 100 / Fixed Assets, INV / SAL: Inventories/Sales, SAL\TA: Sales / Total Assets, TD / TA: Total Dept / Total Assets, PRBT / CL : Profit (Loss) before tax × 100 / Current Liabilities, CL / TA: Current Liabilities / Total Assets, WC / TA: Working Capital / Total Assets, EBIT Margin: Profits before interest and taxes × 100 / Turnover, GP / SAL: Gross Profit / Sales, GP / TA: Gross Profit / Total Assets, (CA − ST) / CL: (Current assets − Stock) / Current liabilities, 365 * AREC / SAL:365 Accounts Receivable / Sales, 365 * ST / CS: 365 * Stock / Cost of Sales, SAL / EQ: Sales / Equity, SAL / TD: Sales / Total Dept, 365 * AP / SAL: 365* Accounts Payable / Sales, TACH: (Total Assets in year t − Total Assets in year t−1) × 100 / Total Assets in year t−1, SALCH: (Sales in year t − Sales in year t−1) × 100 / Sales in year t−1, LOGTA: Logarithm of Total Assets, LOGDEPT: Logarithm of Dept, INV / SAL : Inventories / Total Assets. The Kruskal–Wallis test indicates whether there are statistically significant differences between the two groups. ** Significant at the 1% level, * Significant at the 5% level, *** Significant at the 10% level

the audit opinion and as one otherwise. Skinner [77] considered companies having litigation in the following cases: (a) a lawsuit had been filed in a Greek court; (b) there had been an allegation of common stock price fraud; or (c) there had been an allegation of stock exchange violation under Greek law.

The most consistent result in all previous research has been that auditor size can explain the supply of a higher level of audit quality, defined as the joint probability of detecting and reporting material financial errors (i. e., [4,8,22,24,25,28,47,48,64,72]). The evidence concerning the relationship between audit firms and audit report is mixed. Whereas Warren [88] did find a significant association between large audit firms and qualified audit reports, Shank and Murdock [76] found otherwise.

Previous studies also examined whether the auditor is one of the Big Four (namely PricewaterhouseCoopers, Deloitte and Touche, KPMG, Ernst and Young) or not [35]. We use a dummy variable set to zero (Domestic = 0) if the auditor was one of the domestic audit firms and one if the auditor was one of the foreign audit firms in Greece (Foreign = 1). In Greece the auditing profession was liberalized in 1992 by enabling legislation [37], see Caramanis [13,14]. The competition between the local Greek and foreign audit companies has increased since 1992. Nowadays, Greek companies audit a greater percentage of companies than foreign audit companies. It is possible that smaller companies avoid paying the premium price levied by the large audit companies, since Krishnan et al. [54] found that smaller firms in the US are less likely to be audited by the 'Big Four' companies. Furthermore, it is possible that the partners of domestic audit companies are more likely to develop close personal relationships with the directors of Greek client companies. On the other hand, there is a chance that the domestic audit companies will be more familiar with the 'small acceptable standards of control' in Greece.

Various papers examine the relationship between the audit opinion before and after the chance of auditors (switching). Chow and Rice [16], Craswell [21], Gul et al. [40], and Krishnan et al. [56] found a significant positive association between qualified opinions and subsequent auditor switching. As Nieves [68] points out, two effects may obscure the influence of the audit report in motivating a change: (a) many auditor changes may be unrelated to audit opinion; and (b) the reasons for auditor changes are an internal state which is not directly observable. We tested the importance of auditor changes to detection of FFS over a 3-year period. The 3-year period included the first year of the financial statements and auditors' opinions and the 2 years before this first year. We used a dummy variable (Prior 2 year auditor) that takes a value of one (yes = 1) if the auditor had been retained and a value of zero if the firm had switched auditors (no = 0).

Finally, we used two other variables, LOSS and STOCK. LOSS is an indicator variable whose value is zero if an auditee experienced a loss in the year of audit opinion and one (profit) otherwise. Spathis [80] found a significant difference between qualified and non-qualified audit reports for this variable. STOCK is a dummy variable that takes a value of zero (yes = 0) for companies listed in the Athens Stock Exchange and one for unlisted companies (no = 1). Ireland [46] reported that whether a company is listed or unlisted may influence the auditor's independence. Listed companies may have greater supervision and training of their stock exchange authority. Furthermore, as Ireland [46] points out, large companies are more likely to have good accounting systems and internal controls, thus reducing disagreements and limitations on scope while, at the same time, auditors are more likely to waive earnings management attempts (resulting in mis-statements) in large clients, even after controlling for the materiality of such attempts [67].

Method

Multicriteria decision making (MCDM) provides the methodological basis for the combination of qualitative and quantitative data. MCDA has been applied in finance as a sophisticated tool to improve the decision-taking in the turbulent and complex financial environment that exists nowadays. Spronk et al. [82] thoroughly investigated the application of this technique in the financial field. In the present study we used the MHDIS method [91].

The problem considered in this case study falls within the classification problematic that in general involves the assignment of a finite set of alternatives $A = \{a_1, a_2, \ldots, a_n\}$ to a set of q ordered classes $C_1 \succ C_2 \succ \cdots \succ C_q$. Each alternative was evaluated

along a set of m criteria $g_1, g_2, \ldots, g_m$. In the present case study the alternatives involved the companies in the sample, the criteria correspond to the set of seven financial variables, and six non-financial variables and there were two classes, the unqualified financial statements (class C_1) and the qualified financial statements (class C_2).

MHDIS distinguishes the groups progressively, starting by discriminating the first group from the others, and then proceeds to the discrimination between the alternatives belonging to the other group. To accomplish this task two additive utility functions are developed in each one of the $q-1$ steps, where q is the number of groups. The first function $U_k(a)$ describes the alternatives of group C_1, and the second function $U_{\sim k}(a)$ describes the remaining alternatives that are classified in lower groups $C_{k+1}, \ldots, C_q$.

$$U_k(a) = \sum_{i=1}^{m} p_{ki} u_{ki}(g_i)$$

$$\text{and } U_{\sim k}(a) = \sum_{i=1}^{m} p_{\sim ki} u_{\sim ki}(g_i),$$

$$k = 1, 2, \ldots, q-1.$$

The corresponding marginal utility functions for each criterion g_i are denoted as $u_{ki}(g_i)$ and, $u_{\sim ki}(g_i)$ which are normalized between 0 and 1, while the criteria weights p_{ki} and $p_{\sim ki}$ sum up to 1. As mentioned above, the model is developed in $q-1$ steps. In the first step, the method develops a pair of additive utility functions $U_1(a)$ and $U_{\sim 1}(a)$ to discriminate between the alternatives of group C_1 and the alternatives of the other groups $C_2, \ldots, C_q$. On the basis of the above function forms the rule to decide upon the classification of any alternative has the following form:

If $U_1(a) \geq U_{\sim 1}(a)$ then a belongs in C_1.

Else if $U_1(a) \leq U_{\sim 1}(a)$ then a belongs in (C_2, $C_3, \ldots, C_q$).

The alternatives that are found to belong in class C_1 (correctly or incorrectly) are excluded from further analysis. In the next step, another pair of utility functions $U_2(a)$ and $U_{\sim 2}(a)$ is developed to discriminate between the alternatives of group C_2 and the alternatives of the groups $C_3, \ldots, C_q$. Similarly to step 1, the alternatives that are found to belong in group C_2 are excluded from further analysis. This procedure is repeated up to the last stage $(q-1)$, when all groups have been considered. The estimation of the weights of the criteria in the utility functions as well as the marginal utility functions is accomplished through mathematical programming techniques. More specifically, at each stage of the hierarchical discrimination procedure, two linear programs and a mixed-integer one are solved to estimate the two additive utility functions optimally and to minimize the classification error. Further details of the mathematical programming formulations used in MHDIS can be found in Zopounidis and Doumpos [91].

Empirical Results

Table 1 presents descriptive statistics which indicate the magnitude of the difference in the independent variables between the qualified and unqualified reports over the period 2001–2003. A comparison between the mean value of UTADISCR for the qualified and unqualified companies in the training sample shows that the former had a lower average value, which was statistically significant at the 1% level. Hence, many companies that had manipulated their financial statements were in financial distress [30,36,78]). Statistically significant differences at the 1% level were also found for two others variables, namely Inventories/Sales and Sales Annual Change, between the two groups in the training sample. Thus, companies with lower sales are more likely to receive a qualified report than other companies in Greece.

Furthermore, the variable Profits before tax/Total Assets (ROA) had lower means for the qualified companies, which was consistent with most of the previous studies, indicating that firms which receive qualified opinions made less profit [7,61,78,81,86]).

Table 2 illustrates the contribution of each of the financial and non-financial criteria in our auditing model. As our study involved two groups (unqualified and qualified) the hierarchical discrimination process of MHDIS consisted of only one stage, during which two additive utility functions were developed. The utility function U_1 characterizes the unqualified companies whereas the utility function $U_{\sim 1}$ characterizes those that were qualified.

ROA is indicated as one of the important criteria in most cases. Particularly in the case of unquali-

Multicriteria Decision Support Methodologies for Auditing Decisions, Table 2
Average weights for the criteria in the 2 models

	MHDIS financial		MHDIS No financial	
	U1	U∼1	U1	U∼1
REC / SAL	2.28%	3.82%	0.02%	0.02%
CA / CL	7.17%	25.18%	3.99%	16.41%
CA / TA	21.49%	5.02%	15.04%	10.14%
CASH / TA	4.39%	11.23%	16.87%	1.11%
ROA	44.75%	25.71%	28.09%	11.57%
INV / SAL	12.54%	13.74%	10.55%	0.02%
SALCH	7.38%	15.30%	1.50%	3.94%
Profit or Loss in the year			0.00%	0.00%
Stock Exchange			0.00%	17.49%
Auditor			0.00%	0.00%
Prior 2 years Auditor			9.93%	0.00%
Litigation			0.00%	22.37%
UTADISCR			14.01%	16.94%

Notes: REC / SAL: Receivable / Sales, CA / CL: Current Assets / Current Liabilities, CA / TA: Current Assets / Total Assets, CASH / TA: Cash / Total Assets, ROA: Profit before Tax × 100 / Total Assets, INV / SAL: Inventories / Total Assets, SALCH: (Sales in year t − Sales in year $t-1$) × 100 / Sales in year $t-1$

Multicriteria Decision Support Methodologies for Auditing Decisions, Table 3
Classification results (accuracies in %) for the MHDIS and Logit models

	Unqualified	Qualified	Average
Panel A:Financial Variables			
Training(2001-2003)			
MHDIS	94.87	95.73	95.30
LA	95.7	95.7	95.7
Holdout (2004)			
MHDIS	80.49	87.80	84.15
LA	78.00	86.6	82.3
Panel B: Non-financial Variables			
Training (2001-2003)			
MHDIS	98.29	98.29	98.29
LA	97.43	95.73	96.58
Holdout (2004)			
MHDIS	84.15	91.46	87.81
LA	74.39	93.90	84.15

fied firms, it has a weight that is as high as 44.75% in the financial model and 28.09% in the non-financial one. Similar findings were observed in previous studies [7,61,71,78,81,86]).

The most important criteria that characterized the qualified firms in the case of the financial model are ROA and Current Assets/Current Liabilities (CA/CL) followed by Sales in year t − Sales in year $t-1$ (SALCH) with weights of 25.71, 25.18 and 15.30%, respectively. Pasiouras et al. [71] also found ROA to be statistically significant at the 1% level and one of the most important criteria for the models. In addition, Ireland [46] reported that companies with high liquidity (CA/CL) might increase the likelihood of a qualified audit opinion as assets may have been overstated.

From the non-financial criteria, litigation, STOCK, UTADISCR and CA/CL were the most important criteria with 22.37, 17.49, 16.94 and 16.41%, respectively, for qualified companies. A comparison with the results of previous studies showed that the results were similar. In particular, Spathis [80] found that litigation and financial distress were among the most important variables, and Spathis et al. [79] found CA/CL to be among the most important factors.

Table 3 presents the classification results obtained from the financial and non financial models. The classification ability of the models was tested further using the out-of-time and out-of-sample companies. The results indicated that the MHDIS models developed with the selected variables were able to provide a satisfactory distinction between qualified and unqualified statements.

The overall correct classifications at the training and holdout stages were 95.3 and 84.15%, respectively. The differences between the financial and the non-financial MHDIS models were significant. Overall, the non-financial MHDIS model provided higher overall classification accuracy in both the training (92.3%) and holdout samples (87.81%). This means that the inclusion of non-financial variables in the model yielded a more accurate distinction between qualified and unqualified companies than the inclusion of financial variables alone.

For benchmarking purposes, we developed additional models with logit analysis (LA). These models were developed with the same input variables. In the case of the financial model, the classification accuracy in the training sample was 78%, and the correspond-

ing figure for the holdout sample was 74.3%. In the case of the non-financial model, however, the classification accuracies were 82.3 and 84.15% in the training and holdout samples, respectively. It is therefore clear that MHDIS was more efficient than LA during both the training and holdout stages (for both financial and non-financial models).

MHDIS achieves more balanced results in terms of type I and type II errors in the holdout sample. Whereas Bell and Tabory [9] reported that type II errors are more costly than type I errors, Kida [49] argued that type I errors might result in: (a) a company changing its audit firm (switching), which means loss in audit firm revenue; (b) a lawsuit by a client against the accounting firm; (c) a negative effect on the auditor's reputation in the business community; (d) a deterioration in relations with the client; or (e) the so-called self-fulfilling prophecy – the qualification itself jeopardizes client survival, which in turn increases the probability of that consequence.

Conclusions and Further Research

This study investigated the extent to which MHDIS models based on financial and non-financial variables could predict auditors' decisions to issue qualified opinions in the Greek market.

The sample consisted of 199 companies operating in the Greek manufacturing, trade and service sectors with FSF between 2001 and 2004, matched by industry and total assets with 199 non-FSF ones, yielding a total of 398 companies. We used out-of-time and out-of-sample testing samples to evaluate the classification ability of the model and ensure that they were not overfitted to the training dataset. The sample was split into a training dataset of 234 companies using data from the period 2001–2003 and a validation dataset of 164 companies using data from the year 2004.

Seven financial and six non-financial variables, representing all dimensions of companies' performance, were selected for inclusion in the models that were developed through the MHDIS approach. The results indicated that ROA, CA/CL and Current Assets/Total Assets A were the important criteria for financial model. In additional, litigation, stock exchange, UTADISCR and CA/CL were the most important criteria for the non-financial model. Furthermore, the non-financial MHDIS model provides higher overall classification accuracy indicating that the inclusion of non-financial variables resulted in a more accurate distinction between qualified and unqualified companies.

By using such models, auditors can simultaneously screen a large number of firms and direct their attention to the ones that are more likely to contain misstatements, saving time or money. These models can also be used by policy-makers in an attempt to stop tax evasion (i. e. the tax evasion consisting of filing fraudulent tax declarations in Italy is estimated to be between 3 and 10% of GNP [87]). In addition, these models can be useful to investors, managers, banks and others companies to identify 'red flags'.

The current research could be extended in several directions. First, future research could be extended towards the inclusion of additional variables such as managers' experience, market characteristics (i. e. industry concentration, industry growth), audit fees and non-audit fees, subsidiaries, and stock prices. Second, companies could be classified into more specific groups. Third, the inclusion of data from a longer time period, could allow the consideration of industry and macroeconomic effects. Finally, future research could be directed towards the comparison and integration of alternative or additional classification techniques, such as neural networks, rough sets, expert systems, support vector machine, and others. The integration of the models through additional techniques, such as bagging and boosting, could also be examined.

References

1. Albrecht S, Romney M (1986) Red-flagging management: a validation. Adv Account 3:323–33
2. Altman E (1983) Corporate financial distress: A complete guide to predicting, avoiding, and dealing with bankruptcy. Wiley, New York
3. Arens A, Loebbecke J (1994) Auditing: An Integrated Approach, 6th edn. Prentice-Hall, Englewood Cliffs
4. Balvers R, McDonald B, Miller R (1988) Underpricing of New Issues and the Choice of Auditor as a Signal of Investment Banker Reputation. Account Rev 63:605–21
5. Barnes P (1990) The prediction of takeover targets in the U.K. by means of multiple discriminant analysis. J Bus Finance Account 17(1):73–84
6. Bartley JW, Boardman CM (1990) The relevance of inflation adjusted accounting data to the prediction of corporate takeovers. J Bus Finance Account 17(1):53–72

7. Beasley SM, Carcello JV, Hermanson DR (1999) fraudulent financial reporting: 1987–1997: an analysis of US public companies, research report, Committee on Sponsoring Organizations of the Treadway Commission, AICPA, New York
8. Beatty R (1989) Auditor Reputation and the Pricing of Initial Public Offerings. Account Rev 64:693–709
9. Bell T, Tabor R (1991) Empirical analysis of audit uncertainty qualifications. J Account Res 29:350–370
10. Bell T, Szykowny S, Willingham J (1993) Assessing the likelihood of fraudulent financial reporting: a cascaded logic approach, Working Paper. KPMG Peat Marwick, Montvale
11. Bhargava M, Dubelaar C, Scott T (1998) Predicting bankruptcy in the retail sector: an examination of the validity of key measures of performance. J Retailing Consumer Services 5(2):105–117
12. Bologna G, Lindquist R, Wells J (1996) The Accountant's Handbook of Fraud and Commercial Crime. Wiley, New York
13. Caramanis VC (1997) The enigma of the Greek auditing profession: some preliminary results concerning the impact of liberalization on auditor behaviour. Eur Account Rev 6(1):85–108
14. Caramanis VC (1999) International accounting firms versus indigenous auditors: Intra-professional conflict in the Greek auditing profession. Critical Persp Account 10:153–196
15. Caramanis C, Spathis C (2006) Auditee and audit firm characteristics as determinants of audit qualifications evidence from the Athens stock exchange. Managerial Auditing J 21(9):905–920
16. Chow CW, Rice SJ (1982) Notes: qualified audit opinions and auditor switching. Account Rev LVII(April):326–335
17. Citron DB, Taffler RJ (1992) The audit report under going concern uncertainties: an empirical analysis. Account Bus Res 22(88/Autumn):337–345
18. Coderre GD (1999) fraud detection, using Data Analysis Techniques to detect fraud. Global Audit Publications, Vancover
19. Cosslett SR (1981) Efficient estimation of discrete choice models. In: Manski CF, McFadden D (eds) Structural analysis of discrete data with econometric applications. MIT Press, Cambridge
20. Courtis JK (1978) Modelling a financial ratios categoric framework. J Bus Finance Account 5(4):371–386
21. Craswell AT (1988) The association between qualified opinions and auditor switches. Account Bus Res 19:23–31
22. Craswell AT, Francis R, Taylor SL (1995) Auditor Brand Name Reputations and Industry Specialization. J Account Econ 20:297–322
23. Davia H, Coggins P, Wideman J, Kastantin J (1992) Management Accountant's Guide to Fraud Discovery and Control. Wiley, New York
24. De Angelo LE (1981) Auditor size and Audit Quality. J Account Econ 3:183–199
25. DeFond ML, Wong TJ, Li S (2000) The Impact of Improved Auditor Independence on Audit Market Concentration in China. J Account Econ 28:269–305
26. Dopouch N, Holthausen R, Leftwich R (1987) Predicting audit qualifications with financial and market variables. Account Rev 62(3):431–454
27. Doumpos M, Gaganis C, Pasiouras F (2005) Explaining qualifications in audit reports using a support vector machine methodology. Intelligent Syst Account, Finance Manage 13:197–215
28. Dye R (1993) Auditing Standards, Legal Liability and Auditor Wealth. J Polit Econ 101:887–914
29. Espahbodi H, Espahbodi P (2003) Binary choice models for corporate takeover. J Banking Finance 27:549–574
30. Fanning K, Cogger KO (1998) Neural network detection of management fraud using published financial data. Int J Intelligent Syst Account, Finance Manage 7(1):21–41
31. Fanning K, Cogger K, Srivastava R (1995) Detection of management fraud: a neural network approach. Int J Intelligent Syst Account, Finance Manage 4(2):113–26
32. Fanning K, Cogger K (1998) Neural network detection of management fraud using published financial data. Int J Intelligent Syst Account, Finance Manage 7(1):21–24
33. Gaganis C, Pasiouras F, Spathis C, Zopounidis C (2005) Identifying qualified audit reports in UK firms: a nearest neighbours approach. In: 28th European Accounting Association Annual Congress, Göteborg, Sweden, 18–20 May 2005
34. Gaganis C, Pasiouras F, Tzanetoulakos A (2005) A Comparison and Integration of Classification Techniques for the Prediction of Small UK Firms Failure. J Financ Decis Making 1:55–69
35. Gaganis C, Pasiouras F (2006) Auditing models for the detection of qualified audit opinions in the UK public services sector, Int J Account, Auditing Perform Eval 3(4):471–493
36. Gaganis C, Pasiouras F, Doumpos M (2007) Probabilistic neural networks for the identification of qualified audit opinions. Expert Syst Appl 32:114–124
37. Government Gazette (1993) Law 2166/1993, A/137/24–8–1993. Ethniko Typographeio, Athens
38. Green B (1991) Identifying management irregularities through preliminary analytical procedures, unpublished doctoral dissertation. Kent State University, Kent
39. Green BP, Choi JH (1997) Assessing the risk of management fraud through neuralnetwork technology. J Pract Theory 16(1):14–28
40. Gul FA, Lee DS, Lynn M (1992) A note on audit qualification and switches: some further evidence from a small sample study. J Int Account, Auditing Taxation 1(1):111–120
41. Haskins M, Williams D (1990) A contingent model of intra-Big Eight auditor changes. J Pract Theory 3:55–74
42. Hoffman VB (1997) Discussion of the effects of SAS No. 82 on auditors attention to fraud risk factors and audit planning decisions. J Account Res 35(5):99–104

43. Hollman VP, Patton JM (1997) Accountability, the dilution effect and conservatism in auditors fraud judgments. J Account Res 35(2):227–237
44. Hopwood W, McKeown J, Mutchler J (1994) A Reexamination of Auditor versus Model Accuracy within Xontext of the Going-Concern Opinion Decision. Contemp Account Res 2(10):409–431
45. Imbens W (1992) An efficient method of moments estimator for discrete choice models with choice-base sampling. Econ 60:1187–1214
46. Ireland J (2003) An empirical investigation of determinants of audit reports in the UK. J Bus Finance Account 30(7&8):975–1015
47. Ireland JC, Lennox CS (2002) The Large Audit Firm Fee Premium: A Case of Selectivity Bias? J Account, Auditing Finance 17(1/Winter):73–91
48. Keasey K, Watson R, Wynarzcyk P (1988) The Small Company Audit Qualification: A reliminary Investigation. Account Bus Res 18:323–33
49. Kida T (1980) An investigation into auditor's continuity and related qualification judgments. J Account Res Autumn:506–523
50. Kira D, Morin D (1993) Prediction of takeover targets of Canadian Firms. In: Janssen J, Skiadas CH (eds) Applied Stochastic Models and Data Analysis, pp 507–518. World Scientific Publ Co, Singapore
51. Kirkos E, Spathis C (2006) Data mining techniques for the detection of fraudulent financial statements. Expert Systems with Application, Forthcoming
52. Koh HC, Killough LN (1990) The Use of Multiple Discriminant Analysis in the assessment of the Going Concern status of an audit client. J Bus Finance Account 17(2):179–192
53. Koh HC (1991) Model predictions and auditor assessments of going concern status. Account Bus Res 21(84):331–338
54. Krishnan J, Krishnan J (1996) The role of economic tradeoffs in the audit opinion decision: an empirical analysis. J Account, Auditing Finance 11(4):565–586
55. Krishnan J (1994) Auditor Switching and Conservatism. Account Rev 69:200–16
56. Krishnan J, Krishnan J, Stephens RG (1996) The simultaneous relation between auditor switching and audit opinion: an empirical analysis'. Account Bus Res 26(Summer):224–236
57. Laitinen EK, Laitinen T (1998) Qualified audit reports in Finland: evidence from large companies. Eur Account Rev 7(4):639–653
58. Laitinen T, Kankaanpaa M (1999) Comparative analysis of failure prediction methods: the Finnish case. Eur Account Rev 8:67–92
59. Lennox CS (1999) The Accuracy and Incremental Information Content of Audit Reports in Predicting Bankruptcy. J Bus, Finance Account 26(May–June):757–70
60. Levitan AS, Knoblett JA (1985) Indicators of Expectations to the Going-Concern Assumption. J Pract Theory 5(1):26–39
61. Loebbecke J, Eining M, Willingham J (1989) Auditor's experience with material irregularities: frequency, nature and detectability. J Pract Theory 9:1–28
62. Louwers TJ (1998) The Relation Between Going-Concern Opinions and the Auditor's Loss Function. J Account Res 36(1):143–56
63. Menon K, Schwartz KB (1987) An empirical investigation of audit qualification decisions in the presence of going-concern uncertainties. Contemp Account Res 3(2):302–15
64. Menon K, Williams D (1991) Auditor Credibility and Initial Public Offerings. Account Rev 66:313–32
65. Monroe G, The S (1993) Predicting Uncertainty Audit Qualifications in Australia Using Publicly Available Information. Account Finance 33(2):79–106
66. Mutchler JF (1985) A Multivariate Analysis of the Auditor's Going-Concern Opinion Decision. J Account Res 23(2): 668–82
67. Nelson M, Elliot JA, Tarpley RL (2000) Where do companies attempt earnings management, and when do auditors prevent it? Working Paper, Cornell University, New York
68. Nieves GA, Eiliano RB (2003) Do Spanish firms changes auditor to avoid a qualified audit report? Int J Auditing 7:37–53
69. Palepu KG (1986) Predicting Takeover Targets: A Methodological and Empirical Analysis. J Account Econ 8:3–35
70. Palmrose Z (1987) Litigation and independent auditors: the role of business failures and management fraud. J Pract Theory 6(2):90–102
71. Pasiouras F, Gaganis C, Zopounidis C (2007) Multicriteria decision support methodologies for auditing decisions: the case of qualified audit reports in the UK. Eur J Oper Res 180(3):1317–1330
72. Pong CM, Whittington G (1994) The Determinants of Audit Fees: Some Empirical Models. J Bus Finance Account 21(8):1071–95
73. Porter B, Cameron A (1987) Company fraud-what price the auditor? Account J December:44–47
74. Reynolds J, Francis J (2001) Does size matter? The influence of large clients on office-level auditor reporting decisions. J Account Econ 30:375–400
75. Schilit H (1993) Financial Shenanigans: How to Detect Accounting Gimmicks and Fraud in Financial Reports. McGraw-Hill, New York
76. Shank J, Murdock R (1978) Comparability in the Application of Reporting Standards: Some Further Evidence. Account Rev (October):824–35
77. Skinner JD (1997) Earnings disclosures and stockholder law suits. J Account Econ 23(3):249–282
78. Spathis C (2002) Detecting false financial statements using published data: some evidence from Greece. Managerial Auditing J 17(4):174–191
79. Spathis C, Doumpos M, Zopounidis C (2003) Using client performance measures to identify pre-engagement fac-

tors associated with qualified audit reports in Greece. Int J Account 38:267–284
80. Spathis C (2003) Audit qualification, firm litigation, and financial information: an empirical analysis in Greece. Int J Auditing 7(1):71–85
81. Spathis C, Doumpos M, Zopounidis C (2002) Detecting falsified financial statements: a comparative study using multicriteria analysis and multivariate statistical techniques. Eur Account Rev 11(3):509–535
82. Spronk J, Steuer RE, Zopounidis C (2005) Multicriteria Decision Aid/Analysis in Finance. In: Figueira, Greco, Ehrgott (eds) Multiple Criteria Decision Analysis: State of the Art Surveys. Springer, Boston, Dordrecht, London, pp 799–858
83. Stein RM (2002) Benchmarking default prediction models: Pitfalls and Remedies in Model Validation Moody's KMV Technical Report #030124, June 13
84. Stice J, Albrecht S, Brown L (1991) Lessons to be learned-ZZZZBEST Regina and Lincoln saving. CPA J April:52–54
85. Stice J (1991) Using financial and market information to identify pre-engagement market factors associated with lawsuits against auditors. Account Rev 66(3):516–33
86. Summers SL, Sweeney JT (1998) Fraudulently misstated financial statements and insider trading: an empirical analysis. Account Rev 73(1):131–146
87. Tanzi V, Shome P (1993) A primer on tax evasion. In: International Monetary Fund, Staff Papers 40(4)807–828
88. Warren C (1980) Uniformity of auditing standards: a replication. J Account Res (Spring):312–24
89. Zimbelman MF (1997) The effects of SAS No. 82 on auditors attention to fraud risk-factors and audit planning decisions. J Account Res 35(5):75–9
90. Zmijewski ME (1984) Methodological Issues Related to the Estimation of Financial Distress Prediction Models. J Account Res 22(Supplement):59–86
91. Zopounidis C, Doumpos M (2000) Building additive utilities for multi-group hierarchical discrimination: the M.H.D.I.S method. Optim Methods Softw 14:219–240
92. Zopounidis C, Doumpos M, Gaganis C (2006) Prediction of firms' Bankruptcy: a Multicriteria Methodology. Rev Econ Sci (9):283–296

Multicriteria Methods for Mergers and Acquisitions

Fotios Pasiouras[1], Constantin Zopounidis[2]
[1] School of Management, University of Bath, Bath, UK
[2] Financial Engineering Laboratory, Department of Production Engineering and Management, Technical University of Crete, Chania, Greece

MSC2000: 91B28, 90C05, 90C90

Article Outline

Keywords and Phrases
Introduction/Background
Methods/Applications
Formulation
Cases
Conclusions
References

Keywords and Phrases

Acquisitions; Additive utility functions; Mergers; Multicriteria decision aid; Takeovers

Introduction/Background

Over the last 35 years there have been several studies which have attempted to develop classification models to predict takeover targets in various countries and regions of the world, such as the USA [7,8,10,11,12,15,23,30,34,37], the UK [2,3,4,5,13,28,36], Canada [9,20,29], Greece [31,38,39], and more recently the EU [26,27] and Asia [25]. This is not surprising, since the prediction of acquisitions can be of major interest to stockholders, investors, creditors, and generally anyone who has established a relationship with the acquired firm [35].

Most of these studies have used multivariate statistical and econometric techniques such as discriminant analysis (DA) and logit analysis and only more recently the parametric nature and the statistical assumptions and restrictions of those approaches have led researchers to the application of alternative techniques such as artificial neural networks (ANN) [10], rough sets (RS) [31], recursive partitioning algorithm [15], support vector machines [26], and nearest neighbors [26].

A few recent studies have also used multicriteria decision aid (MCDA, which is the designation usually used in Europe, or multiple criteria decision making, MCDM, which is the one usually used in the USA) techniques [13,25,26,27,36,39] which over the last few years have gained significant recognition among researchers and have been employed in several studies in banking, finance, accounting, and management. For example, Steuer and Na [33] identified 256 applications that combine MCDM and finance. One of the characteris-

tics of these techniques is that they are well suited for analyzing complex decision problems that involve multiple and usually conflicting criteria and/or goals. They can therefore prove particularly useful in the prediction of acquisitions, since there is often not one single reason but a number of reasons that lead management to the decision to merge with or acquire another firm. A further advantage of MCDA techniques is that they do not make any assumptions, as do the traditional techniques (see Barniv and McDonald [6] for a summary of the problems related to the use of discriminant, logit, and probit), about the normality of the variables or the group dispersion matrices (e. g., DA) and they are not sensitive to multicollinearity (e. g., logit analysis). In this paper we first present a brief review of the studies that have applied MCDA in the prediction of acquisition targets (Sect. "Methods/Applications"). Then, in Sect. "Formulation", we outline one of the MCDA techniques, namely, *utilités* additives discriminantes (UTADIS), which is used for the development of our classifications models. "Cases" describes a case study and presents the results. Finally, Sect. "Conclusions" concludes our paper.

Methods/Applications

Zopounidis and Doumpos [39] were the first to propose the use of MCDA in the prediction of acquisition targets. They developed a classification model with the multigroup hierarchical discrimination (MHDIS) method using a sample of 30 acquired and 30 nonacquired Greek firms and ten financial ratios covering various aspects of a firm's financial condition. Data from 1 year prior to the acquisition (year −1) were used for the development of the model, while years 2 (year −2) and 3 (year −3) before the acquisition were used to test its discrimination ability. The model classifies correctly 58.33 and 61.67% of the firms for years 2 and 3 prior to the acquisition, respectively. The authors argue that this poor classification could be attributed to the difficulty of predicting acquisition targets in general, and not necessarily to the inability of the proposed approach as a discrimination method. To test further the proposed technique, its classification accuracy was compared with that of DA and UTADIS. The correct classification accuracy obtained using the proposed method is better for all years than that obtained using DA. As opposed to the UTADIS method, the classification accuracy under the proposed approach is significantly higher for year −1, the same for for year −2, and slightly higher for year −3. On the basis of these results the authors conclude that the iterative binary segmentation procedure is able to provide results that are at least favorably comparable with those provided by UTADIS and outperforms DA.

Tartari et al. [35] also used UTADIS in their study along with linear DA (LDA), probabilistic neural networks (PNN), and RS in an attempt to examine whether the integration of different methods using a stacked generalization approach could result in higher classification accuracies. Their sample consisted of 48 UK firms, selected from 19 industries/sectors, acquired during 2001, and 48 nonacquired firms matched by principal business activity, asset size, sales volume, and number of employees. Twenty-three financial ratios measuring profitability, liquidity and solvency, and managerial performance were initially calculated for each firm for up to 3 years prior to the acquisition (1998–2000); however, they finally used a set of nine ratios, selected on the basis of factor analysis. Their exercise consisted of two stages. First, UTADIS, LDA, PNN, and RS were used to develop individual models. The most recent year (i. e., 2000) was used as a training sample, while data from the other two years (i. e., 1998 and 1999) were used to test the generalizing performance of the proposed integration approach. An eightfold cross-validation approach was employed to develop the base models using the four methods. The classifications of the firms obtained were then used as a training sample for the development of a stacked generalization model. Finally, the development of the stacked model was performed using the UTADIS method that combines (at a metalevel) the group assignments of all the four methods considered in the analysis. The use of other methods to develop the combined model was also examined; nevertheless the results are inferior to those obtained with UTADIS. The stacked model performs better (in terms of the overall correct accuracy rate) than any of the four methods upon which it is based, throughout all the years of the analysis. Furthermore, the results indicate that the stacked model provides significant reductions in the overall error rate compared with LDA, RS, and UTADIS, although they were less significant compared with PNN.

In another UK study, Doumpos et al. [13] compared the classification ability of UTADIS against one of the models developed using DA, logistic regression (LR), and ANN. The sample included 76 UK firms acquired during 2000–2002, matched by industry and size with 76 nonacquired firms. Twenty-nine financial ratios were initial candidates for model development, representing profitability, efficiency, activity, financial leverage, liquidity, and growth; however, the authors finally selected six variables on the basis of a t test and correlation analysis. The UTADIS model was first developed using data drawn from the most recent year prior to the acquisition (i. e., year −1). The model developed was then applied to data from 2 and 3 years prior to the acquisition (years −2 and −3). The average accuracies were 74.34 and 78.95%, respectively. These accuracies are higher than the ones obtained by both DA and LR, and are found to comparable to or better than those of ANN when tested using data from years −2 and −3.

Pasiouras et al. [26] used both UTADIS and MHDIS, among several other classification techniques, to develop models specifically designed for the EU banking industry. They developed several models on the basis of equal and unequal training samples from the period 1998–2000, using both raw and country-adjusted variables. The models were tested in equal and unequal datasets from a future period (2001–2002). They also developed models that combine the predictions of the individual models developed in the first stage, using two integration techniques, namely, stacked generalization and majority voting. Their results were mixed and depended on the form of the variables used, the datasets, and the evaluation measure considered. Hence, they concluded that there is no clear winner technique that dominates all the others under all circumstances. However, UTADIS appears several times as one of the best techniques. Furthermore, the stacked model developed through UTADIS also performs relatively well.

Pasiouras et al. [27] also focused on the EU banking sector, but differentiated their study in two ways from that of Pasiouras et al. [26]. First, they considered an additional MCDA technique, namely, PAIRCLAS, that was applied for the first time in the prediction of acquisitions. Second, they followed a tenfold cross-validation resampling procedure for the development and evaluation of the models. Their sample consisted of 168 banks acquired between 1998 and 2002 matched with 168 nonacquired banks. MHDIS achieved the highest overall accuracy in the validation dataset, with 68% of the acquired and 63.3% of the nonacquired banks classified correctly (implying an overall classification rate of 65.7%). PAIRCLAS also achieves marginally better classification accuracies than UTADIS, and its ability to classify correctly the nonacquired banks (75%) is even higher than that of MHDIS (72.2%).

In another study, Pasiouras et al. [25] concentrated on the Asian banking sector. They used a sample of 52 targets and 47 acquirers that were involved in acquisitions in nine Asian banking markets during 1998–2004 and matched them by country and time with an equal number of banks not involved in acquisitions. The models were developed and validated through a tenfold cross-validation approach using UTADIS and MHDIS. In each case three versions of the model were developed. The first one distinguished between acquired and noninvolved banks. The second one distinguished between acquirers and noninvolved banks. The last one, was a three-outcome model that simultaneously distinguished between targets, acquirers, and noninvolved banks. For comparison purposes they also developed models through DA. The results indicate that the MCDA models are more efficient that the ones developed through DA. Furthermore, in all cases the models are more efficient in distinguishing between acquirers and noninvolved banks than between targets and noninvolved banks. Finally, the models with a binary outcome achieve higher accuracies than the ones which simultaneously distinguish between acquirers, targets, and noninvolved banks.

Formulation

The problem considered in the present study is a classification one that in general involves the assignment of a set of m alternatives $A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m\}$, evaluated along a set of n criteria $g_1, g_2, \ldots, g_n$, to a set of q classes $C_1, C_2, \ldots, C_q$. In the case of acquisitions, the alternatives are the firms in the sample, the criteria can correspond to financial and nonfinancial variables, and there are usually two classes, the nonacquired firms (class C_1) and the acquired firms (class C_2). Hence, in what follows we consider the simple two-class case, while details

on the multiclass case can be found in Doumpos and Zopounidis [14] and Zopounidis and Doumpos [39].

The UTADIS approach, used in the present study, implies the development of an additive utility function that is used to score the firms and decide upon their classification. The utility function has the following general form:

$$U(\mathbf{a}) = \sum_{i=1}^{n} w_i u_i'(g_i) \in [0,\ 1]\ , \tag{1}$$

where w_i is the weight of criterion g_i (the criteria weights sum up to 1) and $u'_i(g_i)$ is the corresponding marginal utility function normalized between 0 and 1. The marginal utility functions provide a mechanism for decomposing the aggregate result (global utility) in terms of individual assessment to the criterion level. To avoid the estimation of both the criteria weights and the marginal utility functions, it is possible to use the transformation $u_i(g_i) = w_i u'_i(g_i)$. Since $u'_i(g_i)$ is normalized between 0 and 1, it becomes obvious that $u_i(g_i)$ ranges in the interval $[0, w_i]$. In this way, the additive utility function is simplified to the following form:

$$U(\mathbf{a}) = \sum_{i=1}^{n} u_i(g_i) \in [0,\ 1]\ . \tag{2}$$

The utility function developed provides an aggregate score $U(\mathbf{a})$ of each firm along all criteria. In the case of acquisitions prediction, this score provides the basis for determining whether the firm could be classified in either the group of nonacquired ones or in the group of acquired ones. The classification rule in this case is the following (C_1 and C_2 denote the group of nonacquired and acquired firms, respectively, while u_1 is a cutoff utility point defined on the global utility scale, i. e., between 0 and 1):

$$\left.\begin{array}{ll} U(\mathbf{a}) \geq u_1 & \Rightarrow \mathbf{a} \in C_1 \\ U(\mathbf{a}) < u_1 & \Rightarrow \mathbf{a} \in C_2 \end{array}\right\}\ . \tag{3}$$

The estimation of the additive value function and the cutoff threshold is performed using linear programming techniques so that the sum of all violations of the classification rule (3) for all the firms in the training sample is minimized. A detailed description and derivation of this mathematical programming formulation can be found in Doumpos and Zopounidis [14].

Cases

In this section, our method is illustrated by a case study from the work of Pasiouras et al. [24]. The dataset considered in the study consists of 76 firms acquired between 2000 and 2002, and 76 nonacquired firms, which operate in manufacturing, construction, and mining–quarrying–extraction industries in the UK. The sample was constructed as follows. The acquired firms were first identified in the Hemscott M&As database and the financial data were collected from the Financial Analysis Made Easy database of Bureau van Dijk. After screening for data availability in FAME, 59 manufacturing, six construction, five production and six mining–quarrying–extraction firms had complete financial data for the 3 years prior to the acquisition and were included in the sample.

Although the year of acquisition is not common for all firms in the sample, they were all thought to be acquired in the "zero" year, considered as the year of reference. The years of activity prior to "zero" are coded as "year −1" (1 year prior), "year −2" (2 years prior), and "year −3" (3 years prior).

After the sample described above had been obtained, nonacquired firms were chosen to match the acquired firms. The firms were matched by industry and size (total assets) and financial data for the nonacquired companies were taken from the same calendar years as for the corresponding acquired companies.

Barnes [5] mentions that the problem for the analyst who attempts to forecast targets is simply a matter of identifying the best predictive (i. e., explanatory) variables. Unfortunately, financial theories do not offer much in selecting specific variables among the numerous ones regarded as potential candidates in model development. Given the large number of possible ratios, it is important to reduce the list of ratios that enter the final model selection process. Hence, a question that emerges when attempting to select accounting ratios for empirical research is which ones, among the hundreds, should be used? However, there is no easy way to determine how many ratios a particular model should contain. Too few and the model will not capture all the relevant information. Too many and the model will overfit the training sample, but underperform in a holdout sample, and will most likely have onerous data input requirements [21]. As Hamer [17]

points out, the variable set should be constructed on the basis of (1) minimizing the cost of data collection and (2) maximizing the model applicability. Huberty [18] suggests three variable screening techniques that could be used: logical screening (e. g., financial theory and human judgment), statistical screening (e. g., test of differences of two group means such as the *t* test), and dimension reduction (e. g., factor analysis). In the present study we follow the latermost approach as in Stevens [34], Barnes [2], Kira and Morin [20], Zanakis and Zopounidis [38], and Tartari et al. [35]. Hence, a total of 25 variables are initially considered on the basis of data availability and previous studies, covering several aspects of firms' performances such as profitability, efficiency, activity, financial leverage, liquidity, and growth. Factor analysis is then used to reduce the number of variables to a smaller number of factors that are linear combinations of the initial variables. The analysis results in the extraction of seven factors, with eigenvalues higher than 1. The variable with the highest loading is selected from each one of the seven components for inclusion in the classification models. Consequently, we use the following seven variables:

1. X1: Current assets/current liabilities,
2. X2: Total liabilities/shareholders' equity,
3. X3: Annual change of total assets,
4. X4: Annual change of current liabilities,
5. X5: Profits before taxes/total assets,
6. X6: Sales/stock,
7. X7: Sales/debtors.

X1 is an indicator of liquidity that has been used in many previous studies [2,12,20,38]. The views about liquidity are somewhat mixed. It is possible that firms with excess liquidity are more likely to be acquired because of their good short-term financial position and the availability of cash or near-cash assets [36]. In this case, there is also an opportunity for the acquirers to finance the acquisition with the target's own resources [32]. On the other hand, it can be argued that a firm in need of funds to finance its working capital requirements is likely to be an acquisition target because the acquirer, after the acquisition, expects to bring additional funds into the firm to improve its liquidity [29].

X2 is a measure of financial leverage that has been used as a proxy for financial leverage in Rege [29], Palepu [23], and Kim and Arbel [19] among others. According to the *financial leverage hypothesis* the likelihood of being acquired decreases with the increase in company debt. There are two reasons why firms with lower preexisting levels of debt are considered attractive acquisition targets. The first is that the low debt ratio of the target decreases the probability of future default of the joint firm, while at the same time it increases the debt capacity of the new firm. The second is that in some cases a firm has extremely low debt ratios, the value of the firm may not be maximized, and low leverage can be seen as a sign of inefficient management.

X3 and X4 are measures of annual changes in two basic elements of the firms (i. e., assets, liabilities). Firms whose growth rates, as measured by X3, are relatively high can experience problems because their management and/or structure will not able to deal with and sustain exceptional growth. It is therefore possible that a firm which is constrained in this way will become an acquisition target of a firm with surplus resources or management available to help [14]. Furthermore, a firm with high levels of growth might be acquired by firms that what to take advantage of this increase in assets, and boost their own growth. Turning to X4, exceptional increases may indicate that the firm has problems in meeting its short-term liabilities and can therefore be acquired to avoid solvency.

Variables X5, X6, and X7 are related to the *inefficient management hypothesis*. This hypothesis argues that if the managers of a firm fail to maximize its market value, then the firm is likely to be an acquisition target and inefficient managers will be replaced. Thus, these takeovers are motivated by a belief that the acquiring firm's management can manage better the target's resources. This view is supported by two specific arguments. First, the firm might be poorly run by its current management, partly because the objectives of the management are at variance with those of the shareholders. In this case, the takeover threat can serve as a control mechanism limiting the degree of variance between management's pursuits for growth from shareholders' desire for wealth maximization. A merger may not be the only way to improve management, but if disappointed shareholders cannot accomplish a change in management that will increase the value of their investment within the firm, either because it is too costly or too slow, then a merger may be a simpler and more practical way of achieving their desired goals. Second,

Multicriteria Methods for Mergers and Acquisitions, Table 1 Weights of the variables (percent) in the *utilités* additives discriminantes (UTADIS) model (averages over the ten replications)

	Year −1	Year −2	Year −3
X1	11.15	2.45	10.40
X2	23.28	34.04	31.99
X3	5.15	7.43	8.80
X4	15.79	5.71	8.59
X5	14.67	25.08	12.22
X6	14.40	11.31	14.77
X7	15.56	13.97	13.23

the acquirer may simply have better management experience than the target. There are always firms with unexploited opportunities to cut costs and increase sales and earnings, and that makes them natural candidates for acquisition by other firms with better management [1]. Therefore, if the management of the acquirer is more efficient than the management of the target firm, a gain could result through a merger if the management of the target is replaced.

Table 1 presents the contribution of the seven criteria in the UTADIS model. To ensure the proper development and validation of the models, we follow a tenfold cross-validation. Hence, the total sample of 152 firms is randomly split into ten mutually exclusive subsamples (folds) of approximately equal size. Then ten models are developed, using each fold in turn for validation and the remaining folds for training. Therefore, in each of the ten replications, the training sample consists of 137 firms and the validation of 15 firms. The figures presented are the averages over the ten replications.

X2 (total liabilities/shareholders' equity) appears to be the most important criterion in all 3 years with an average weight that ranges between 23.28 (year −1) and 34.04% (year −3). The profitability and efficiency indicators (X5, X6, X7) also appear to be important in classifying firms within the two groups, with average weights between 11.31 and 25.08%. X1 (i. e., current assets ratio) carries a weight above 10% in years −1 and −3 but it is considerably reduced to 2.45% in year −2. Finally, X3, which corresponds to the annual growth of the firm in terms of total assets, is the least important criterion in all years.

Multicriteria Methods for Mergers and Acquisitions, Table 2 Classification accuracies in percent (averages over ten replications)

	Acquired firms	Nonacquired firms	Overall accuracy
Classification accuracies of the UTADIS model in the development stage			
Year −1	80.1	71.8	75.9
Year −2	81.9	71.1	76.5
Year −3	81.1	70.3	75.7
Classification accuracies in the validation stage			
Year −1			
UTADIS	76.2	63.9	70.1
DA	77.9	54.0	65.9
Year −2			
UTADIS	75.3	65.5	70.4
DA	77.4	47.2	62.3
Year −3			
UTADIS	77.3	63.1	70.2
DA	65.5	50.4	58.0

By comparing the score $U(\mathbf{a})$ of each firm with the cutoff threshold that was calculated through the estimation of the UTADIS model and rule (3), we can decide whether a firm can be classified as acquired or not acquired. Table 2 presents the classification results obtained by UTADIS during the development and validation process. In Table 2 we also present the classification results obtained by DA, used for benchmarking purposes.

The overall classification accuracy of the UTADIS model during the development stage is around 75%. Furthermore, the model appears to be quite robust, with classifications that do not deviate significantly from one year to another. Unsurprisingly, consistent with previous studies, the classification accuracy decreases in the validation stag; however, the decrease is relatively small and the overall classification accuracy is now around 70%. It should be mentioned that while our model misclassifies around 30% of the firms in the validation dataset, this is not uncommon for studies on the prediction of acquisitions targets.

Other studies that used resampling techniques obtained similar results. Bartley and Boardman [7] reported a classification accuracy of 64%, while in a later

study [8] they obtained classification accuracies between 69.9 and 79.9%. Similarly, the classification accuracy in the study of Kira and Morin [20] was 66.17%. The study of Pasiouras et al. [27] that focused on the EU banking industry also reported classification accuracies between 61.6 and 65.7%. As Barnes [14] notes, perfect prediction models are difficult to develop even for bankruptcy prediction, where failing firms have definitely inferior or abnormal performance compared with healthy firms. The problem with the identification of acquisition targets is not only that there are potentially many reasons for acquisitions, but also that at the same time managers do not always act in a manner which maximizes shareholders' returns owing to hybris or agency motives.

While the comparison of the results obtained in the current study with the ones of previous studies gives a first indication for the performance of the model, a direct comparison is not appropriate because of differences in the datasets [16,21], the industry under investigation, the methods used to validate the models, and so on. Hence, the comparison of the UTADIS model with the one developed with DA using exactly the same dataset, variables, and development and validation procedures might provide a more accurate indication of the efficiency (in terms of classification accuracy) of the MCDA model. Looking at the results in Table 2, we see that UTADIS clearly achieves higher classification accuracies than DA. Furthermore, while the classification accuracies of DA decrease as we move back in time, the accuracies of UTADIS remain quite robust, even when we use data from 3 years prior to the acquisition. Finally, with the exception of acquired firms in year −1, UTADIS outperforms DA in classifying correct firms of both groups (i. e., acquired, nonacquired).

Conclusions

In this paper we first discussed why MCDA could be useful in the prediction of acquisition targets and provided a review of relevant studies. Then, we presented the UTADIS technique and its application on a dataset of acquired and nonacquired UK firms.

The application indicates that UTADIS not only outperforms a model developed by DA, but it also achieves quite robust results, as we use data that move away from the period of the event.

Future applications of MCDA in the area of acquisitions prediction could focus on the incorporation of nonfinancial and qualitative data (e. g. managers' experience, managers' educational background) in the analysis. Although this has been mentioned in the literature in the past [22,38], there is still a lack of studies that use such variables in the analysis, usually owing to data availability. MCDA techniques, like UTADIS, can easily incorporate qualitative data, and it would be therefore interesting to perform such an exercise. Furthermore, it would also be worthwhile to investigate the classification of firms in more than two groups (e. g., acquired, acquirers, noninvolved) as in the study of Pasiouras et al. [25]. While the results of the later study were not promising, the study focused on the banking industry, which is a special case. Hence, results from nonfinancial sectors might lead to different conclusions. MCDA techniques, like MHDIS, which was developed with the multigroup discrimination in mind, might be useful in such applications.

References

1. Arnold G (1998) Corporate Financial Management, 1st edn. Financial Times Prentice Hall, London
2. Barnes P (1990) The prediction of takeover targets in the U.K. by means of multiple discriminant analysis. J Bus Finance Account 17:73–84
3. Barnes P (1998) Can takeover targets be identified by statistical techniques?: Some UK evidence. Stat 47:573–591
4. Barnes P (1999) Predicting UK Takeover Targets: Some Methodological Issues and an Empirical Study. Rev Quant Finance Account 12:283–301
5. Barnes P (2000) The identification of U.K. takeover targets using published historical cost accounting data. Some empirical evidence comparing logit with linear discriminant analysis and raw financial ratios with industry-relative ratios. Int Rev Financial Analys 9:147–162
6. Barniv R, McDonald JB (1999) Review of Categorical Models for Classification Issues in Accounting and Finance. Rev Quant Finance Account 13:39–62
7. Bartley JW, Boardman CM (1986) Replacement-Cost-Adjusted Valuation Ratio as a Discriminator Among Takeover Target and Nontarget Firms. J Econom Bus 38:41–55
8. Bartley JW, Boardman CM (1990) The relevance of inflation adjusted accounting data to the prediction of corporate takeovers. J Bus Finance Account 17:53–72
9. Belkaoui A (1978) Financial ratios as predictors of Canadian takeovers. J Bus Finance Account 5:93–107

10. Cheh JJ, Weinber RS, Yook KC (1999) An Application Of An Artificial Neural Network Investment System To Predict Takeover Targets. J Appl Bus Res 15:33–45
11. Cudd M, Duggal R (2000) Industry Distributional Characteristics of Financial Ratios: An Acquisition Theory Application. Financial Rev 41:105–120
12. Dietrich JM, Sorensen E (1984) An Application of Logit Analysis to Prediction of Merger Targets. J Bus Res 12:393–402
13. Doumpos M, Kosmidou K, Pasiouras F (2004) Prediction of Acquisition Targets in the UK: A Multicriteria Approach. Oper Res Int J 4:191–211
14. Doumpos M, Zopounidis C (2002) Multicriteria Decision Aid Classification Methods. Kluwer, Dordrecht
15. Espahbodi H, Espahbodi P (2003) Binary choice models for corporate takeover. J Bank Finance 27:549–574
16. Gupton GM, Stein RM (2002) LossCalc^{TM}: Moody's Model for Predicting Loss Given Default (LGD). Moody's Investors Service Global Credit Research, Special Comment, February
17. Hamer MM (1983) Failure prediction: Sensitivity of classification accuracy to alternative statistical methods and variable sets. J Account Public Policy 2:289–307
18. Huberty CJ (1984) Applied Discriminant Analysis. Wiley, New York
19. Kim WG, Arbel A (1998) Predicting merger targets of hospitality firms (a Logit model). Int J Hosp Manag 17:303–318
20. Kira D, Morin D (1993) Prediction of takeover targets of Canadian Firms. In: Janssen J, Skiadas CH (eds) Applied Stochastic Models and Data Analysis. World Scientific Publ. Co., Singapore, pp 507–518
21. Kocagil AE, Reyngold A, Stein RM, Ibarra E (2002) Moody's RiskCalc^{TM} Model For Privately-Held U.S. Banks, Moody's Investors Service. Global Credit Research, July
22. Meador AL, Church PH, Rayburn LG (1996) Development of prediction models for horizontal and vertical mergers. J Financial Strateg Decis 9:11–23
23. Palepu KG (1986) Predicting Takeover Targets: A Methodological and Empirical Analysis. J Account Econ 8:3–35
24. Pasiouras F, Doumpos M, Zopounidis C (2006) Predicting acquisitions: Methodological Framework and Applications. Klidarithmos Publications (In Greek), Athens
25. Pasiouras F, Gaganis C, Zopounidis C (2007) Classification models for the detection of acquisition targets in the Asian banking sector, Paper presented at the 22nd European Conference on Operational Research, Prague, July 8–11
26. Pasiouras F, Tanna S, Zopounidis C (2005) Application of Quantitative Techniques for the Prediction of Bank Acquisition Targets. World Scientific Publishing Co, Singapore
27. Pasiouras F, Tanna S, Zopounidis C (2007) The identification of acquisition targets in the EU banking industry: an application of multicriteria approaches. Int Rev Financial Analys 16:262–281
28. Powell RG (2001) Takeover Prediction and Portfolio Performance: A Note. J Bus Finance Account 28:993–1011
29. Rege UP (1984) Accounting ratios to locate take-over targets. J Bus Finance Account 11:301–311
30. Simkowitz M, Monroe RJ (1971) A discriminant analysis function for conglomerate mergers. South J Bus 38:1–16
31. Slowinski R, Zopounidis C, Dimitras AI (1997) Prediction of company acquisition in Greece by means of the rough set approach. Eur J Oper Res 100:1–15
32. Song MH, Walkling RA (1993) The impact of managerial ownership on acquisition attempts and target shareholder wealth. J Financial Quant Analys 12:439–457
33. Steuer RE, Na P (2003) Multiple criteria decision making combined with finance: A categorized bibliographic study. Eur J Oper Res 150:496–515
34. Stevens DL (1973) Financial Characteristics of Merged Firms: A Multivariate Analysis. J Financial Quant Analys 8:149–158
35. Tartari E, Doumpos M, Baourakis G, Zopounidis C (2003) A stacked generalization framework for the prediction of corporate acquisitions. Foundat Comput Decis Sci 28:41–61
36. Tzoannos J, Samuels JM (1972) Mergers and takeovers: the financial characteristics of companies involved. J Bus Finance 4:5–16
37. Walter RM (1994) The Usefulness of Current Cost Information for Identifying Takeover Targets and Earning Above-Average Stock Returns. J Account Auditing Finance 9:349–377
38. Zanakis SH, Zopounidis C (1997) Prediction of Greek company takeovers via multivariate analysis of financial ratios. J Oper Res Soc 48:678–687
39. Zopounidis C, Doumpos M (2002) Multi-group discrimination using multi-criteria analysis: Illustrations from the field of finance. Eur J Oper Res 139:371–389

Multicriteria Sorting Methods

CONSTANTIN ZOPOUNIDIS, MICHAEL DOUMPOS
Department Production Engineering and Management, Financial Engineering Lab. Techn. University Crete University Campus, Chania, Greece

MSC2000: 90C29

Article Outline

Keywords
Multicriteria Sorting Methods
Goal Programming Approaches
Outranking Relations Approaches
Preference Disaggregation Approaches

A Multigroup Hierarchical Discrimination Method
Model Formulation
The Hierarchical Discrimination Process
Estimation of Utility Functions
LP1: Minimizing the Overall Classification Error
LP2: Minimizing the Number of Misclassifications
LP3: Maximizing the Minimum Distance
An Illustrative Example
Distinguishing Between C_1 and C_2-C_3
Distinguishing Between C_2 and C_3
Concluding Remarks and Future Perspectives
See also
References

Keywords

Sorting; Multicriteria analysis; Goal programming; Outranking relation; Preference disaggregation

Decision making problems, according to their nature, the policy of the decision maker, and the overall objective of the decision may require the choice of an alternative solution, the ranking of the alternatives from the best to the worst ones or the *sorting* of the alternatives in predefined homogeneous classes [30]. For instance, a decision regarding the location of a new power plant can be considered as a choice problem, since the objective is to select the most appropriate location according to environmental, social and investment criteria. On the other hand, an evaluation of the efficiency of the different units of a firm can be considered as a ranking problem, since the objective is to estimate the relative performance of each unit compared to the others. Finally, a credit granting decision is a sorting problem: a credit application can be accepted, rejected or submitted for further consideration, according to the business and personal profile of the applicant. Actually, a wide variety of decision problems, including financial and investment decisions, environmental decisions, medical decisions, etc., are better formulated and studied through the sorting approach.

The sorting problem, generally stated, involves the assignment of a set of observations (objects, alternatives) described over a set of attributes or criteria into predefined homogeneous classes. This type of problem can also referred to as the *'discrimination'* problem or the *'classification'* problem. Although any of these three terms can be used to describe the general objective of the problem (i. e. the assignment of observations into groups), actually, they refer to two slightly different situations: the discrimination or classification problem refers to the assignment of observations into classes which are not necessarily ordered. On the other hand, sorting refers to the problem in which the observations should be classified into classes which are ordered from the best to the worst ones. For instance, in medical diagnosis the classification of patients according to their symptoms into several possible diseases is a discrimination (classification) problem, since it is impossible to establish a preference ordering between the diseases. On the contrary, the evaluation of bankruptcy risk is a sorting problem, since the non-bankrupt firms are preferred to the bankrupt ones. In this paper the terms 'discrimination', 'classification', and 'sorting' will be used without distinction to refer to the general problem of assigning observations, objects or alternatives into classes.

The major practical interest of the sorting problem, has motivated researchers in developing an arsenal of methods for studying such problems, with the aim being the development of quantitative models achieving the higher possible classification accuracy and predicting ability. In 1936, R.A. Fisher [8] was the first to propose a framework for studying classification problems taking into account their multidimensional nature. The linear discriminant analysis (LDA) that Fisher proposed has been used for decades as the main classification technique and it is still being used at least as a reference point for comparing the performance of new techniques that are developed. C. Smith in 1947 [34] extended Fisher's linear discriminant analysis proposing quadratic discriminant analysis (QDA) in order to overcome the restrictive assumption underlying LDA that groups have equal dispersion matrices. Later on, several other statistical classification approaches have been proposed. Among them logit and probit analysis are the most widely used techniques overcoming the multivariate normality assumption of discriminant analysis (both linear and quadratic). Although these techniques overcome most of the statistical restrictions imposed in discriminant analysis, their parameters are difficult to explain, especially in multigroup discriminant problems.

The continuous advances in other fields including operations research and artificial intelligence led many scientists and researchers to exploit the new capabili-

ties of these fields, in developing more efficient classification techniques. Among the attempts made one can mention neural networks, machine learning, fuzzy sets as well as *multicriteria decision aid* (MCDA). This article will focus on MCDA and its application in the study of classification problems with or without ordered classes. MCDA provides an arsenal of powerful and efficient nonparametric classification methods and approaches, which are free of statistical assumptions and restrictions, while furthermore they are able to incorporate the decision maker's preferences in a flexible and realistic way.

The remainder of the article is organized as follows. Section 2 provides a review of MCDA sorting approaches and techniques, outlining their basic characteristics, concepts and limitations. In section 3, a new MCDA sorting method is described and its operation is depicted through a simple illustrative example. Finally, section 4 concludes the paper and outlines some possible future research directions concerning the application of MCDA in sorting problems.

Multicriteria Sorting Methods

The MCDA methods which have been proposed for the study of sorting problems can be distinguished either according to the approach from which they are originated (multi-objective/goal programming, multiattribute utility theory, outranking relations, preference disaggregation), or according to the type of problem that they address (ordered or non-ordered classes). The review presented in this section will distinguish the methods according to their origination, but in the same time the type of problems that they address will also be discussed.

Goal Programming Approaches

The work of A. Charnes and W.W. Cooper [4] set the foundations on goal/multi-objective programming, but it can also be considered as one or the pioneering studies in the field of MCDA in general. Since then, both multi-objective and *goal programming* constitute two major fields of interest from the theoretical and practical points of view in the MCDA and operations research communities. In particular, goal programming approaches, during the 1960s and the 1970s have been used to elicit attribute weights in multiple criteria ranking decision problems [15,27,35,36]. N. Freed and F. Glover [9] were among the first to investigate the potentials of goal programming techniques in the discriminant problem. Their aim was to develop a linear discriminant model so that the minimum distance of the score of each alternative from a predefined cut-off point is maximized (maximize the minimum distance-MMD). To develop this model, they proposed the following goal programming formulation:

$$\begin{cases} \max & d \\ \text{s.t.} & \sum w_i x_{ij} + d \le c, \quad \forall i \in \text{Group 1}, \\ & \sum w_i x_{ij} - d \ge c, \quad \forall i \in \text{Group 2}, \end{cases}$$

where w_i is the weight of attribute i, x_{ij} is the evaluation of alternative j on attribute i, and c is the cut-off score (w_i and d are unrestricted in sign). Soon after proposing this model, the same authors proposed a variety of similar goal programming formulations incorporating several other discrimination criteria, such as the sum of deviations (optimize the sum of deviations-OSD), the sum of interior deviations (minimize the sum of interior deviations-MSID) and the maximum deviation [10].

These two studies attracted the interest of several operational researchers and management scientists. S.M. Bajgier and A.V. Hill [2] proposed a new goal programming approach in order to minimize the number of misclassifications using a mixed integer programming formulation (MIP) and conducted a first experimental study to compare the MMD model, the OSD model, and their MIP formulation with LDA. They concluded that the goal programming formulations are generally superior to LDA, except for the case of moderate to low overlap between groups and equal dispersion matrices, where LDA outperforms all the examined goal programming formulations.

The performance of goal programming approaches compared to statistical techniques was an issue that several researchers tried to investigate using mainly experimental data sets. Freed and Glover [11] compared MMD, MSID, OSD and LDA and they concluded that although the presence of outliers pose a greater problem for the two simpler goal programming formulations (MMD and MSID) than for LDA, generally the goal programming approaches outperform LDA.

E.A. Joachimsthaler and A. Stam [18] compared the LDA, QDA, logistic regression and OSD procedures and they concluded that these methodologies produce similar results although the misclassification rates for LDA and QDA tended to increase with highly kurtosis data and increased dispersion heterogeneity. C.A. Markowski and E.P. Markowski [22] examined the influence of qualitative attributes on the discriminating performance of MMD and LDA. Although the incorporation of qualitative attributes in LDA violates the normality assumption, the experimental study of the authors showed that the incorporation of qualitative variables improved the performance of LDA, while on the other hand MMD did not appear to be particularly well-suited for use with qualitative variables. In another experimental study conducted by P.A. Rubin [32], QDA outperformed 15 goal programming approaches, leading the author to indicate that 'if LP models are to be considered seriously as an alternative to conventional procedures, they must be shown to outperform QDA under plausible conditions, presumably involving non-Gaussian data'. These experimental studies clearly indicate the confusion concerning the discriminating performance of the goal programming formulations as opposed to well known multivariate statistical techniques. Except for this issue, the research on the field of goal programming approaches for discriminant problems, was also focus on the theoretical drawbacks which were often meet. Markowski and Markowski [23] were the first to identify two major drawbacks of the goal programming formulations (MMD and OSD) proposed by Freed and Glover [9,10]. More specifically, they proved that if each quadrant contains at least one case from the second group, unacceptable solutions will result in MMD (all coefficients in the discriminant function are zeros which leads all the observations to be classified in the same group), while furthermore they showed that the solutions (discriminant functions) obtained through the MMD and the OSD models are not stable when the data are transformed (when there is a shift from the origin). Except for these two problems, many goal programming formulations were found to suffer from two additional theoretical shortcomings [29]:

a) they produce unbounded solutions, and
b) they produce improper solutions.

A solution is considered *unbounded* if the objective function can be increased or decreased without limit, in which case the discrimination rule (function) may be meaningless, whereas a solution is improper if all observations fall on the classification hyperplane.

To overcome these problems new goal programming formulations were proposed, including hybrid models [12,13], nonlinear programming formulations [37], as well as several mixed integer programming formulations [1,3,5,20,33,38,39].

In the light of this review of goal programming approaches for discriminant problems it is possible to identify the following three characteristics of the research in this field:

1) The majority of the proposed models aim at developing a linear discrimination rule (function). The extension of the models to develop a nonlinear discriminant function leads to nonlinear programming formulations which are generally computationally intensive and difficult to solve. Among the few alternative approaches is the MSM method (multisurface method) proposed by O.L. Mangasarian [21] that leads to the construction of a piecewise linear discrimination surface between two groups (see also [26] for a revision of the method using multi-objective programming and fuzzy mathematical programming techniques).
2) Little research has been made on extending the existing framework on the multigroup discriminant problem. E.-U. Choo and W.C. Wedley [5], W. Gochet et al. [14], as well as J.M. Wilson [39] applied goal programming approaches in multigroup discriminant problems, but generally most of the studies in this field were focused on two-group discrimination trying to extend the original goal programming models of Freed and Glover [9,10] in order to achieve higher classification accuracy and predicting ability.
3) The models based on the goal programming approach can be applied in any classification problem with or without ordered classes.

Outranking Relations Approaches

In contrast to the goal programming approaches, *outranking relations* procedures study the classification problem on a completely different basis. The aim of such procedures is not to develop a discriminant function (linear or nonlinear), but instead their aim is to

model the decision makers' preferences and develop a global preference model which can be used to assign the alternatives (observations) into the predefined classes. To achieve the classification of the alternatives some reference profiles are determined which can be considered as representative examples of each class. Through the comparison of each alternative with these reference profiles the classification of the alternatives is accomplished.

A representative example of MCDA sorting method based on the outranking relations approach is the ELECTRE TRI method proposed by W. Yu [40]. The aim of ELECTRE TRI is to provide a sorting of the alternatives under consideration into two or more ordered categories. In order to define the categories ELECTRE TRI uses some reference alternatives (reference profiles) r_i, $i = 1, \ldots, k-1$, which can be considered as fictitious alternatives different from the alternatives under consideration. The profile r_i is the theoretical limit between the categories C_i and C_{i+1} (C_{i+1} is preferred to C_i) and r_i is strictly better than r_{i-1} for each criterion. To provide a sorting of the alternatives in categories ELECTRE TRI makes comparisons of each alternative with the profiles.

For an alternative a and a profile r_i the concordance index $c_j(a, r_i)$ is calculated. This index expresses the strength of the affirmation 'alternative a is at least as good as profile r_i on criterion j'. In order to compare the alternative to a reference profile on the basis of more than one criteria, a global concordance index $C(a, r_i)$ is calculated. This index expresses the strength of the affirmation 'a is at least as good as r_i according to all criteria'. Setting w_j as the weight of the criterion j, $C(a, r_i)$ is constructed as the weighted average of all $c_i(a, r_i)$.

In contrast to the concordance index, the discordance index $D_j(a, r_i)$ expresses the strength of the opposition to the affirmation 'alternative a is at least as good as profile r_i according to criterion g_j'. The calculation of the discordance index is based on the definition of a veto threshold $v_j(r_i)$ for criterion j and the profile r_i. The veto threshold $v_j(r_i)$ for criterion j defines the minimum accepted difference between the values of the profile r_i and alternative a on the specific criterion so that we can say that they have totally different preference according to criterion j.

Let $\overline{F}(a, r_i)$ be the set consisted of all criteria for which the discordance index value is greater than the value of global concordance index. For each affirmation of the type: 'alternative a outranks profile r_i according to all criteria', the credibility index $\sigma_s(a, r_i)$ is calculated. If $\overline{F}(a, r_i)$ is empty then $\sigma_s(a, r_i) = C(a, r_i)$, otherwise the credibility index is calculated as follows:

$$\sigma_s(a, r_i) = C(a, r_i) \cdot \prod_{j \in \overline{F}} \frac{1 - D_j(a, r_i)}{1 - C(a, r_i)}.$$

If the value of the credibility index of the affirmation 'alternative a outranks profile r_i according to all criteria} exceeds a predefined cut-off value λ, then the proposition 'a outranks r_i' can be considered to be valid. Denoting the outranking relation as **S**, the preference (**P**), indifference (**I**) and incomparability (**R**) relations between alternative a and profile r_i can be defined as follows:

- $a\mathbf{I}r_i$ if and only if $a\mathbf{S}r_i$ and $r_i\mathbf{S}a$;
- $a\mathbf{P}r_i$ if and only if $a\mathbf{S}r_i$ and no $r_i\mathbf{S}a$;
- $r_i\mathbf{P}a$ if and only if no $a\mathbf{S}r_i$ and $r_i\mathbf{S}a$;
- $a\mathbf{R}r_i$ if and only if no $a\mathbf{S}r_i$ and no $r_i\mathbf{S}a$.

According to these relations two sorting procedures are applied: the pessimistic and the optimistic one. The sorting procedure starts by comparing alternative a to the worst profile r_1 and in the case where $a\mathbf{P}r_1$, a is compared to the second profile r_2, etc., until one of the following two situations appears:

i) $a\mathbf{P}r_i$ and $r_{i+1}\mathbf{P}a$ or $a\mathbf{I}r_{i+1}$;

ii) $a\mathbf{P}r_i$ and $a\mathbf{R}r_{i+1}, \ldots, a\mathbf{R}r_{i+k}, r_{i+k+1}\mathbf{P}a$.

If situation i) appears, then alternative a is assigned to category $i+1$ by both pessimistic and optimistic procedures. If situation ii) appears, then a is assigned to category $i+1$ by the pessimistic procedure and to category $i+k+1$ by the optimistic procedure.

It is clear that the ELECTRE TRI method is a powerful tool for analyzing the decision maker's preference in sorting problems involving multiple criteria where the classes are ordered. However, the major drawback of the method is the significant amount of information that it requires by the decision maker (weights of the criteria, preference and indifference thresholds, veto thresholds, etc.). This problem can be overcame using decision instances (assignment examples) as proposed in [25].

Other MCDA sorting methods based on the outranking relations approach have been proposed in [24] (N-TOMIC method), [31] and the PROMETHEE

method as it has been modified in [19]. Furthermore, P. Perny [28] extended the existing framework of the sorting methods based on the outranking relations approach in the case in which the groups are not ordered. More specifically, he proposed the construction of a fuzzy outranking relation in order to estimate the membership of each alternative for each group, and suggested two assignment procedures:

a) filtering by strict preference (the assignment rule consists of testing whether an alternative is preferred or not to a reference profile reflecting the lower limit of a group), and
b) filtering by indifference (the assignment rule consists of testing whether an alternative is indifferent or not to a reference profile representing a prototype of a group).

Overall the main characteristics of sorting methods based on the outranking relations approach of MCDA include their application to both sorting (ordered classes) as well as discrimination (non ordered classes) problems, and the significant amount of information that they require by the decision maker.

Preference Disaggregation Approaches

The *preference disaggregation* approach refers to the analysis (disaggregation) of the global preferences of the decision maker to deduce the relative importance of the evaluation criteria, using ordinal regression techniques based mainly on linear programming formulations.

In contrast to the outranking relations approach the global preference model of the decision maker is not constructed through a direct interrogation procedure between the decision analyst and the decision maker. Instead, decision instances (e. g. past decisions) are used in order to analyze the decision policy of the decision maker, to specify his/her preferences and construct the corresponding global preference model as consistently as possible.

A well known preference disaggregation method is the UTA method (UTilités Additives) proposed in [17]. Given a predefined ranking of a reference set of alternatives, the aim of the UTA method is to construct a set of additive utility functions which are as consistent as possible with the pre-ordering of the alternatives (and consequently with the decision maker's preferences). The form of the additive utility function is the following:

$$U(\overline{g}) = \sum_j u_j(g_j),$$

where $U(\overline{g})$ denotes the global utility of an alternative described over a vector of criteria $\overline{g}$, while $u_j(g_j)$ is the partial or marginal utility of an alternative on criterion g_j.

Except for the study of ranking problems, the methodological framework of the preference disaggregation approach using the UTA method is also applicable in sorting problems. The UTADIS method (UTilités Additives DIScriminantes) [6,16,17,42] is a representative example. In the UTADIS method, the sorting of the alternatives is accomplished by comparing the global utility (scores) of each alternative a, denoted as $U(a)$, with some thresholds $(u_1, \ldots, u_{q-1})$ which distinguish the classes $C_1, \ldots, C_q$ (the classes are ordered, so that C_1 is the class of the best alternatives and C_q is the class of the worst alternatives).

$$U(a) \geq u_1 \Rightarrow a \in C_1$$
$$u_2 \leq U(a) < u_1 \Rightarrow a \in C_2$$
$$\cdots$$
$$u_k \leq U(a) < u_{k-1} \Rightarrow a \in C_k$$
$$\cdots$$
$$U(a) < u_{q-1} \Rightarrow a \in C_q.$$

The objective of the UTADIS method is to estimate an additive utility function and the utility thresholds in order to minimize the classification error. The classification error is measured through two error functions denoted as $\sigma^+(a)$ and $\sigma^-(a)$, representing the deviations of a misclassified alternative from the utility threshold. The estimation of both the additive utility model and the utility thresholds is achieved through linear programming techniques [6,42].

See [7] and [41] for three variants of the UTADIS method to improve the classification accuracy of the obtained additive utility models as well as their predicting ability. The first variant (UTADIS I) except for the classification errors also incorporates the distances of the correctly classified alternatives from the utility thresholds which have to be maximized. The second variant (UTADIS II) is based on a mixed integer programming formulation minimizing the number of

misclassifications instead of their magnitude, while the third variant (UTADIS III) combines UTADIS I and II, and its aim is to minimize the number of misclassifications and maximize the distances of the correctly classified alternatives from the utility thresholds.

Overall the main characteristics of the application of the preference disaggregation approach in the study of sorting problems, can be summarized in the following three aspects.

1) The information that is required is minimal, since, similarly to the goal programming approaches, only a predefined classification of a reference set of alternatives is required.
2) The preference disaggregation approach is focused only on decision problems where the classes are ordered, since it is assumed that there is a strict preference relation between the classes.
3) The classification/sorting models which are developed have a nonlinear form, since the marginal utilities of the evaluation criteria are piecewise linear and consequently the global utility model is also nonlinear, in contrast to the linear discriminant models used in the goal programming approaches.

A Multigroup Hierarchical Discrimination Method

In this section a new method is presented for the study of discrimination problems with two or more ordered groups (multigroup discrimination). The proposed method is called M.H.DIS (Multigroup Hierarchical DIScrimination) and differs from most of the aforementioned MCDA approaches in two major aspects.

1) It employs a hierarchical discrimination approach: the method does not aim on the development of an overall global preference model (discriminant function) which will characterize all the observations (alternatives or objects). Instead the method is trying to distinguish the groups progressively, starting by discriminating the first group (best alternatives) from all the others, and then proceeding to the discrimination between the objects which belong to the other groups.
2) It accommodates three different discrimination criteria in a very flexible and efficient way. The most common discrimination criterion in the previous approaches is the minimization of the classification error which is measured as the deviations of the scores of the misclassified alternatives from some cut-off points. However, such an objective does not necessarily yield the optimal classification rule. For instance, consider that in a discrimination problem, three alternatives are misclassified with the following deviations from the cut-off point: [0.25, 0.25, 0.25], with the overall objective of minimizing the total classification error being 0.75. It is obvious, that this classification result is not optimal, since a classification result [0, 0, 0.75] yields the same value for the overall classification error (0.75), but there is only one misclassified alternative instead of three. Several mixed integer programming formulations have been proposed to confront this issue, but their application in real world problems is prohibited by the significant amount of time required to solve such problems. M.H.DIS employs an efficient *mixed integer programming* (MIP) formulation for minimizing the number of misclassifications, once the minimization of the classification error has been achieved. Furthermore, M.H.DIS also considers a third criterion in order to achieve the higher possible discrimination. These three discrimination criteria have been used in previous studies separately, or in hybrid models [12,13], but they have never been used through a sequential procedure. Instead, in M.H.DIS initially the classification error is minimized. Then considering only the misclassified alternatives M.H.DIS tries to 're-arrange' their classification error in order to minimize the number of misclassifications, and finally the maximum discrimination between the alternatives is attempted.

Model Formulation

Let $A = \{a_1, \ldots, a_n\}$ be a set of n alternatives which should be classified into q ordered classes $C_1, \ldots, C_q$.(C_1 is preferred to C_2, C_2 is preferred to C_3, etc.) Each alternative is described (evaluated) along a set $G = \{g_1, \ldots, g_m\}$ of m evaluation criteria. The evaluation of each alternative a on criterion g_i is denoted as $g_i(a)$. According to the set A of alternatives, p_i different values for each criterion g_i can be distinguished. These p_i values are rank-ordered from the smallest value g_i^1 to the largest value $g_i^{p_i}$. Furthermore, among the set of cri-

teria it is possible to distinguish two subsets: a subset G_1 consisting of m_1 criteria for which higher values indicate higher preference, and a second subset G_2 consisting of m_2 criteria for which the decision maker's preference is a decreasing function of the criterion's scale. For instance, in an investment decision problem G_1 may include criteria related to the return of an investment project (projects with higher return are preferred), while G_2 may include criteria related to the risk of the investment (projects with lower risk are preferred).

The Hierarchical Discrimination Process

The method proceeds progressively in the classification of the alternatives into the predefined classes, starting from class C_1 (best alternatives). Initially, the aim is to identify which alternatives belong in class C_1. The alternatives which are found to belong in class C_1 (either correctly or incorrectly) are excluded from further consideration. In a second stage the objective is to identify which alternatives belong in class C_2. The alternatives which are found to belong in this class (either correctly or incorrectly) are excluded from further consideration, and the same procedure continues until all alternatives have been classified in the predefined classes.

Throughout this hierarchical classification procedure, it is assumed that the decision maker's preferences are monotone functions (increasing or decreasing) on the criteria's scale. This assumption implies that in the case of a criterion $g_i \in G_1$, as the evaluation of an alternative on this criterion increases, then the decision of classifying this alternative into a higher (better) class is more favorable to a decision of classifying the alternative into a lower (worst) class. For instance, in the credit granting problem as the profitability of a firm increases, the credit analyst will be more favorable in classifying the firm as a healthy firm, rather than classifying it as a risky one. A similar implication is also made for each criterion $g_i \in G_2$.

This preference relation between the several possible decisions of classifying a specific alternative a into one of the predefined classes, imposes the following general classification rule:

> The decision concerning the classification of an alternative a into one of the predefined classes should be made in such a way that the utility (value) of such a decision for the decision maker is maximized.

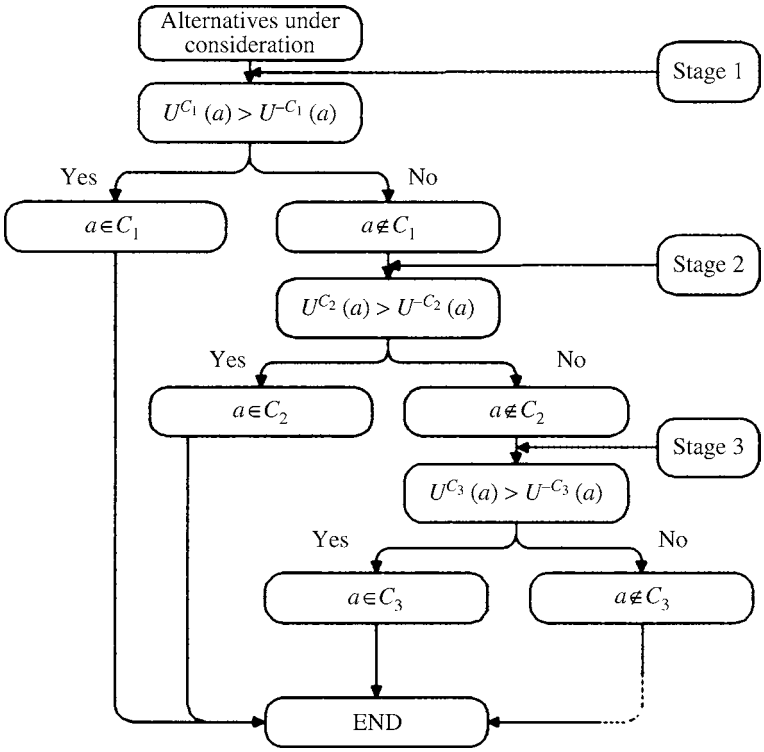
The utility of a decision concerning the classification of an alternative a into group C_j can be expressed in the form of additive utility function:

$$U^{C_j}(a) = \sum_{i=1}^{m} u_i^{C_j}[g_i(a)] \in [0, 1],$$

where $u_i^{C_j}[g_i(a)]$ denotes the marginal (partial) utility of the decision concerning the classification of an alternative a into group C_j according to criterion g_i. If $g_i \in G_1$, then $u_i^{C_j}(g_i)$ will be an increasing function on the criterion's scale. On the contrary, the marginal utility of a criterion $g_i \in G_2$ regarding the classification of an alternative into a lower (worse) class C_k $(k > j)$ will be a decreasing function on the criterion's scale. For instance, consider once again the credit granting problem: since healthy firms are generally characterized by high profitability, the marginal utility for a profitability criterion for the group of healthy firms will be an increasing function, indicating that as profitability increases the preference of decision concerning the classification of a firm in the group of healthy firms in also increasing. On the other hand, for the group of risky firms the marginal utility will be a decreasing function of the criterion's (profitability) values, indicating that as profitability increases the preference of the decision concerning the classification of a firm in the group of risky firms is decreasing.

Consequently, at each stage of the hierarchical classification procedure that was described above, two utility functions are constructed. The first one corresponds to the utility of a decision concerning the classification of an alternative a into class C_k (denoted as $U^{C_k}(a)$), while the second one corresponds to the utility of a decision concerning the nonclassification of an alternative a into class C_k (denoted as $U^{-C_k}(a)$). Based on these two utility functions the aforementioned general classification rule can be expressed as follows:

$$\begin{cases} \text{if } U^{C_k}(a) > U^{-C_k}(a), & \text{then } a \in C_k, \\ \text{if } U^{C_k}(a) < U^{-C_k}(a), & \text{then } a \notin C_k. \end{cases} \quad (1)$$



Multicriteria Sorting Methods, Figure 1
The hierarchical classification procedure

Following this rule, the overall hierarchical discrimination procedure is presented in Fig. 1.

Estimation of Utility Functions

According to the hierarchical discrimination procedure which was described above, to achieve the classification of the alternatives in q classes, the number of utility functions which must be estimated is $2(q-1)$. The estimation of these utility functions in M.H.DIS is accomplished through linear programming techniques. More specifically, at each stage of the hierarchical discrimination procedure, two linear programs and one mixed integer program are solved to estimate 'optimally' the two utility functions.

LP1: Minimizing the Overall Classification Error

According to the classification rule (1), to achieve the correct classification of an alternative $a \in C_k$ at stage k (cf. Fig. 1), the estimated utility functions should satisfy the following constraint:

$$U^{C_k}(a) > U^{-C_k}(a).$$

Since, in linear programming it is not possible to use strict inequality constraints, a small positive real number s may be used as follows:

$$U^{C_k}(a) - U^{-C_k}(a) \geq s.$$

If for an alternative $a \in C_k$ the classification rule at stage k yields $U^{C_k}(a) < U^{-C_k}(a)$, then this alternative is misclassified, since it should be classified in one of the lower classes (the specific classification of the alternative will be determined in the next stages of the hierarchical discrimination process). The classification error in this case is:

$$e(a) = U^{-C_k}(a) - U^{C_k}(a) + s.$$

Similarly, to achieve the correct classification of an alternative $b \notin C_k$ at stage k, the estimated utility func-

tions should satisfy the following constraint:

$$U^{-C_k}(b) - U^{C_k}(b) \geq s.$$

If this constraint is not satisfied for an alternative $b \notin C_k$ at stage k, then this fact implies that this alternative should be classified in class C_k and the classification error in this case is $e(b) = U^{C_k}(b) - U^{-C_k}(b) + s$.

Moreover, to achieve the monotonicity of the marginal utilities, the following constraints are imposed:

$$\text{if } g_i \in G_1 \begin{cases} u_i^{C_k}(g_i^1) = 0 \\ u_i^{-C_k}(g_i^{p_i}) = 0 \\ u_i^{C_k}(g_i^{j+1}) > u_i^{C_k}(g_i^j) \\ u_i^{-C_k}(g_i^{j+1}) < u_i^{-C_k}(g_i^j) \end{cases} \tag{2}$$

$$\text{if } g_i \in G_2 \begin{cases} u_i^{C_k}(g_i^{p_i}) = 0 \\ u_i^{-C_k}(g_i^1) = 0 \\ u_i^{C_k}(g_i^{j+1}) < u_i^{C_k}(g_i^j) \\ u_i^{-C_k}(g_i^{j+1}) > u_i^{-C_k}(g_i^j) \end{cases} \tag{3}$$

where g_i^j and g_i^{j+1} are two consecutive values of criterion $g_i (g_i^{j+1} > g_i^j$ for all $g_i \in G)$. These constraints can be simplified by setting:

$$\text{if } g_i \in G_1 \begin{cases} w_{ij,j+1}^{C_k} = u_i^{C_k}(g_i^{j+1}) - u_i^{C_k}(g_i^j) \\ w_{ij,j+1}^{-C_k} = u_i^{-C_k}(g_i^j) - u_i^{-C_k}(g_i^{j+1}) \end{cases} \tag{4}$$

$$\text{if } g_i \in G_2 \begin{cases} w_{ij,j+1}^{C_k} = u_i^{C_k}(g_i^j) - u_i^{C_k}(g_i^{j+1}) \\ w_{ij,j+1}^{-C_k} = u_i^{-C_k}(g_i^{j+1}) - u_i^{-C_k}(g_i^j) \end{cases} \tag{5}$$

The marginal utility of criterion g_i at point g_i^j can then be calculated through the following formulas:

$$\begin{aligned} u_i^{C_k}(g_i^j) &= \sum_{l=1}^{j-1} w_{il,l+1}^{C_k}, \\ u_i^{-C_k}(g_i^j) &= \sum_{l=j}^{p_i-1} w_{il,l+1}^{-C_k}. \end{aligned} \tag{6}$$

Using these transformations, constraints (2) and (3) can be rewritten as follows (a small positive number t is used to ensure the strict inequality):

$$w_{ij,j+1}^{C_k} \geq t, \quad w_{ij,j+1}^{-C_k} \geq t, \qquad \forall g_i.$$

Consequently, the initial linear program (LP1) to be solved can be formulated as follows:

$$\begin{cases} \min & F = \sum_{a \in A} e(a) \\ \text{s.t.} & U^{C_k}(a) - U^{-C_k}(a) + e(a) \geq s, \\ & \quad \forall a \in C_k, \\ & U^{-C_k}(b) - U^{C_k}(b) + e(b) \geq s, \\ & \quad \forall b \notin C_k, \\ & w_{ij,j+1}^{C_k} \geq t \\ & w_{ij,j+1}^{-C_k} \geq t \\ & \sum_i \sum_j w_{ij,j+1}^{C_k} = 1 \\ & \sum_i \sum_j w_{ij,j+1}^{-C_k} = 1 \\ & e(a), s, t \geq 0. \end{cases}$$

LP2: Minimizing the Number of Misclassifications

If after the solution of (LP1), there exist some alternatives $a \in A$ for which $e(a) > 0$, then obviously these alternatives are misclassified. However, as it has been already illustrated during the discussion of the main characteristics of M.H.DIS, it may be possible to achieve a 're-arrangement' of the classification errors which may lead to the reduction of the number of misclassifications.

In M.H.DIS this is achieved through a mixed integer programming (MIP) formulation. However, since MIP formulations are difficult to solve, especially in cases where the number or integer or binary variables is large, the MIP formulation used in M.H.DIS considers only the misclassifications occurred by solving (LP1), while retaining all the correct classifications. Let C be the set of alternatives which have been correctly classified after solving (LP1), and M be the set of misclassified alternatives for which $e(a) > 0$. The MIP formulation used in

M.H.DIS is the following (LP2):

$$\begin{cases} \min & F = \sum_{a \in A} I(a) \\ \text{s.t.} & U^{C_k}(a) - U^{-C_k}(a) \geq s, \\ & \quad \forall a \in C_k \cap C, \\ & U^{-C_k}(b) - U^{C_k}(b) \geq s, \\ & \quad \forall b \notin C_k, b \in C, \\ & U^{C_k}(a) - U^{-C_k}(a) + I(a) \geq s, \\ & \quad \forall a \in C_k \cap M, \\ & U^{-C_k}(b) - U^{C_k}(b) + I(a) \geq s, \\ & \quad \forall b \notin C_k, b \in M, \\ & w_{ij,j+1}^{C_k} \geq t \\ & w_{ij,j+1}^{-C_k} \geq t \\ & \sum_i \sum_j w_{ij,j+1}^{C_k} = 1 \\ & \sum_i \sum_j w_{ij,j+1}^{-C_k} = 1 \\ & s, t, I(a) \text{ integer.} \end{cases}$$

The first set of constraints is used to ensure that all the correct classifications achieved by solving (LP1) are retained. The second set of constraints is used only for the alternatives which were misclassified by (LP1). Their meaning is similar to the constraints in LP1, with the only difference being the transformation of the continuous variables $e(a)$ of LP1 (classification errors) into integer variables $I(a)$ which indicate whether an alternative is misclassified or not. The meaning of the final two constraints has already been illustrated in the discussion of the LP1 formulation. The objective of LP2 is to minimize the number of misclassifications occurred through the solution of LP1.

LP3: Maximizing the Minimum Distance

Solving LP1 and LP2 the 'optimal' classification of the alternatives has been achieved, where the term 'optimal' refers to the minimization of the number of misclassified alternatives. However, the correct classification of some alternatives may have been 'marginal', that is although they are correctly classified, their global utilities according to the two utility functions developed may have been very close. The objective of LP3 is to maximize the minimum difference between the global utilities of the correctly classified alternatives achieved according to the two utility functions.

Similarly to LP2, let C be the set of alternatives which have been correctly classified after solving LP1 and LP2, and M be the set of misclassified alternatives. LP3 can be formulated as follows:

$$\begin{cases} \max & d \\ \text{s.t.} & U^{C_k}(a) - U^{-C_k}(a) - d \geq s, \\ & \quad \forall a \in C_k \cap C, \\ & U^{-C_k}(b) - U^{C_k}(b) - d \geq s, \\ & \quad \forall b \notin C_k, b \in C, \\ & U^{C_k}(a) - U^{-C_k}(a) \geq s, \\ & \quad \forall a \in C_k \cap M, \\ & U^{-C_k}(b) - U^{C_k}(b) \geq s, \\ & \quad \forall b \notin C_k, b \in M, \\ & w_{ij,j+1}^{C_k} \geq t \\ & w_{ij,j+1}^{-C_k} \geq t \\ & \sum_i \sum_j w_{ij,j+1}^{C_k} = 1 \\ & \sum_i \sum_j w_{ij,j+1}^{-C_k} = 1 \\ & d, s, t \geq 0. \end{cases}$$

The first set of constraints involves only the correctly classified alternatives. In these constraints d represents the minimum absolute difference between the global utilities of each alternative in the two utility functions. The second set of constraints involves the misclassified alternatives and it is used to ensure that they will be retained as misclassified.

An Illustrative Example

To illustrate the application of the method, consider a simple example consisting of six alternatives evaluated along three evaluation criteria [25] for which higher values are preferred. The alternatives must be classified in three ordered classes. Table 1, illustrates the evaluation of the alternatives on the criteria as well as the predefined classification.

Distinguishing Between C_1 and C_2-C_3

In the first stage of the hierarchical discrimination procedure, the aim is to distinguish the alternatives belonging in class C_1 from the alternatives belonging in

Multicriteria Sorting Methods, Table 1
Data of the illustrative example (Source: [25])

	g_1	g_2	g_3	Class
a_1	70	64.75	46.25	C_1
a_2	61	62	60	C_1
a_3	40	50	37	C_2
a_4	66	40	23.125	C_2
a_5	20	20	20	C_3
a_6	15	15	30	C_3

classes C_2 and C_3. To achieve this classification two utility functions are developed, denoted as $U^{C_1}(a)$ and $U^{-C_1}(a)$.

The utility of the decision of classifying the alternative a_1 in class C_1 can be expressed as follows:

$$U^{C_1}(a_1) = u_1^{C_1}(70) + u_2^{C_1}(64.75) + u_3^{C_1}(46.25). \quad (7)$$

Since for all criteria higher values are preferred, it is possible to define the following rank-order on each criterion's scale ($p_1 = p_2 = p_3 = 6$).

$g_1)$ $g_1^1 = 15 < 20 < 40 < 61 < 66 < 70 = g_1^{p_1}$;
$g_2)$ $g_2^1 = 15 < 20 < 40 < 50 < 62 < 64.75 = g_2^{p_2}$;
$g_3)$ $g_3^1 = 20 < 23.125 < 30 < 37 < 46.25 < 60 = g_3^{p_3}$.

According to relation (4), the following transformations are then applied (criterion g_1):

$$\begin{aligned}
w_{11,2}^{C_1} &= u_1^{C_1}(20) - u_1^{C_1}(15),\\
w_{12,3}^{C_1} &= u_1^{C_1}(40) - u_1^{C_1}(20),\\
w_{13,4}^{C_1} &= u_1^{C_1}(61) - u_1^{C_1}(40),\\
w_{14,5}^{C_1} &= u_1^{C_1}(66) - u_1^{C_1}(61),\\
w_{15,6}^{C_1} &= u_1^{C_1}(70) - u_1^{C_1}(66).
\end{aligned}$$

The same transformations are also applied to criteria g_2 and g_3. Then, according to (6), relation (7) can be re-written in the following way:

$$\begin{aligned}
U^{C_1}(a) = &(w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1} + w_{14,5}^{C_1} + w_{15,6}^{C_1})\\
&+ (w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1} + w_{24,5}^{C_1} + w_{25,6}^{C_1})\\
&+ (w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1} + w_{34,5}^{C_1}).
\end{aligned}$$

On the other hand, if a_1 is classified in class C_2 then the utility of the decision maker will be:

$$\begin{aligned}
&U^{-C_1}(a_1) = u_1^{-C_1}(70) + u_2^{-C_1}(64.75) + u_3^{-C_1}(46.25)\\
&\Updownarrow\\
&U^{-C_1}(a_1) = w_{35,6}^{-C_1}.
\end{aligned}$$

Following the same methodology, the utilities concerning the classification of the rest of the alternatives are also formulated.

- Alternative a_2:

$$\begin{aligned}
&U^{-C_1}(a_2) = u_1^{C_1}(61) + u_2^{C_1}(62) + u_3^{C_1}(60)\\
&\Updownarrow\\
&U^{C_1}(a_2) = (w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1})\\
&\quad + (w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1} + w_{24,5}^{C_1})\\
&\quad + (w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1} + w_{34,5}^{C_1} + w_{35,6}^{C_1}),\\
&U^{-C_1}(a_2) = u_1^{-C_1}(61) + u_2^{-C_1}(62) + u_3^{-C_1}(60)\\
&\Updownarrow\\
&U^{-C_1}(a_2) = (w_{14,5}^{-C_1} + w_{15,6}^{-C_1}) + (w_{25,6}^{-C_1}).
\end{aligned}$$

- Alternative a_3:

$$\begin{aligned}
&U^{-C_1}(a_3) = u_1^{C_1}(40) + u_2^{C_1}(50) + u_3^{C_1}(37)\\
&\Updownarrow\\
&U^{C_1}(a_3) = (w_{11,2}^{C_1} + w_{12,3}^{C_1})\\
&\quad + (w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1})\\
&\quad + (w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1}),\\
&U^{-C_1}(a_3) = u_1^{-C_1}(40) + u_2^{-C_1}(50) + u_3^{-C_1}(37)\\
&\Updownarrow\\
&U^{-C_1}(a_3) = (w_{13,4}^{-C_1} + w_{14,5}^{-C_1} + w_{15,6}^{-C_1})\\
&\quad + (w_{24,5}^{-C_1} + w_{25,6}^{-C_1}) + (w_{34,5}^{-C_1} + w_{35,6}^{-C_1}).
\end{aligned}$$

- Alternative a_4:

$$\begin{aligned}
&U^{-C_1}(a_4) = u_1^{C_1}(66) + u_2^{C_1}(40) + u_3^{C_1}(23.125)\\
&\Updownarrow\\
&U^{C_1}(a_4) = (w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1} + w_{14,5}^{C_1})
\end{aligned}$$

$$+ (w_{21,2}^{C_1} + w_{22,3}^{C_1}) + (w_{31,2}^{C_1}),$$
$$U^{-C_1}(a_4) = u_1^{-C_1}(66) + u_2^{-C_1}(40) + u_3^{-C_1}(23.125)$$
$$\Updownarrow$$
$$U^{-C_1}(a_4) = (w_{15,6}^{-C_1}) + (w_{23,4}^{-C_1} + w_{24,5}^{-C_1} + w_{25,6}^{-C_1}) + (w_{32,3}^{-C_1} + w_{33,4}^{-C_1} + w_{34,5}^{-C_1} + w_{35,6}^{-C_1}).$$

- Alternative a_5:

$$U^{-C_1}(a_5) = u_1^{C_1}(20) + u_2^{C_1}(20) + u_3^{C_1}(20)$$
$$\Updownarrow$$
$$U^{C_1}(a_5) = (w_{11,2}^{C_1}) + (w_{21,2}^{C_1}),$$
$$U^{-C_1}(a_5) = u_1^{-C_1}(20) + u_2^{-C_1}(20) + u_3^{-C_1}(20)$$
$$\Updownarrow$$
$$U^{-C_1}(a_5) = (w_{12,3}^{-C_1} + w_{13,4}^{-C_1} + w_{14,5}^{-C_1} + w_{15,6}^{-C_1}) + (w_{22,3}^{-C_1} + w_{23,4}^{-C_1} + w_{24,5}^{-C_1} + w_{25,6}^{-C_1}) + (w_{31,2}^{-C_1} + w_{32,3}^{-C_1} + w_{33,4}^{-C_1} + w_{34,5}^{-C_1} + w_{35,6}^{-C_1}).$$

- Alternative a_6:

$$U^{-C_1}(a_6) = u_1^{C_1}(15) + u_2^{C_1}(15) + u_3^{C_1}(30)$$
$$\Updownarrow$$
$$U^{C_1}(a_6) = (w_{31,2}^{C_1} + w_{32,3}^{C_1}),$$
$$U^{-C_1}(a_6) = u_1^{-C_1}(15) + u_2^{-C_1}(15) + u_3^{-C_1}(15)$$
$$\Updownarrow$$
$$U^{-C_1}(a_6) = (w_{11,2}^{-C_1} + w_{12,3}^{-C_1} + w_{13,4}^{-C_1} + w_{14,5}^{-C_1} + w_{15,6}^{-C_1}) + (w_{21,2}^{-C_1} + w_{22,3}^{-C_1} + w_{23,4}^{-C_1} + w_{24,5}^{-C_1} + w_{25,6}^{-C_1}) + (w_{33,4}^{-C_1} + w_{34,5}^{-C_1} + w_{35,6}^{-C_1}).$$

According to these expressions of the global utility of the decision to classify an alternative into class C_1 or into one of the classes C_2 and C_3, the LP1 formulation is used to minimize the classification error ($s = 0.001$, $t = 0.0001$).

$$\begin{cases} \min & F = e(a_1) + e(a_2) + e(a_3) + e(a_4) \\ & \quad + e(a_5) + e(a_6) \\ \text{s.t.} & U^{C_1}(a_1) - U^{-C_1}(a_1) + e(a_1) \geq 0.001 \\ & U^{C_1}(a_2) - U^{-C_1}(a_2) + e(a_2) \geq 0.001 \\ & U^{-C_1}(a_3) - U^{C_1}(a_3) + e(a_3) \geq 0.001 \\ & U^{-C_1}(a_4) - U^{C_1}(a_4) + e(a_4) \geq 0.001 \\ & U^{-C_1}(a_5) - U^{C_1}(a_5) + e(a_5) \geq 0.001 \\ & U^{-C_1}(a_6) - U^{C_1}(a_6) + e(a_6) \geq 0.001 \\ & w_{ij,j+1}^{C_1} \geq 0.0001, \quad w_{ij,j+1}^{-C_1} \geq 0.0001, \\ & \sum_{i=1}^{3} \sum_{j=1}^{5} w_{ij,j+1}^{C_1} = 1, \\ & \sum_{i=1}^{3} \sum_{j=1}^{5} w_{ij,j+1}^{-C_1} = 1, \\ & \forall i = 1, 2, 3, \quad \forall j = 1, \ldots, 6, \\ & e(a_1), e(a_2), e(a_3) \geq 0, \\ & e(a_4), e(a_5), e(a_6) \geq 0. \end{cases}$$

The obtained solution is presented in Table 2. According to this solution, the marginal utilities are calculated.

- Criterion g_1:
  - $u_1^{C_1}(15) = 0$,
  - $u_1^{-C_1}(15) = w_{11,2}^{-C_1} + w_{12,3}^{-C_1} + w_{13,4}^{-C_1} + w_{14,5}^{-C_1} + w_{15,6}^{-C_1} = 0.25937$,

Multicriteria Sorting Methods, Table 2
Results obtained through the solution of LP1

$w_{11,2}^{C_1}$	0.00010	$w_{11,2}^{-C_1}$	0.03708
$w_{12,3}^{C_1}$	0.00010	$w_{12,3}^{-C_1}$	0.03708
$w_{13,4}^{C_1}$	0.09872	$w_{13,4}^{-C_1}$	0.07406
$w_{14,5}^{C_1}$	0.00010	$w_{14,5}^{-C_1}$	0.03708
$w_{15,6}^{C_1}$	0.09872	$w_{15,6}^{-C_1}$	0.07406
$w_{21,2}^{C_1}$	0.00010	$w_{21,2}^{-C_1}$	0.03708
$w_{22,3}^{C_1}$	0.00010	$w_{22,3}^{-C_1}$	0.03708
$w_{23,4}^{C_1}$	0.09872	$w_{23,4}^{-C_1}$	0.07406
$w_{24,5}^{C_1}$	0.13570	$w_{24,5}^{-C_1}$	0.11104
$w_{25,6}^{C_1}$	0.09872	$w_{25,6}^{-C_1}$	0.07406
$w_{31,2}^{C_1}$	0.00010	$w_{31,2}^{-C_1}$	0.03708
$w_{32,3}^{C_1}$	0.09872	$w_{32,3}^{-C_1}$	0.07406
$w_{33,4}^{C_1}$	0.09872	$w_{33,4}^{-C_1}$	0.07406
$w_{34,5}^{C_1}$	0.13570	$w_{34,5}^{-C_1}$	0.11104
$w_{35,6}^{C_1}$	0.13570	$w_{35,6}^{-C_1}$	0.11104

- $u_1^{C_1}(20) = w_{11,2}^{C_1} = 0.0001$,
- $u_1^{-C_1}(20) = w_{12,3}^{-C_1} + w_{13,4}^{-C_1} + w_{14,5}^{-C_1} + w_{15,6}^{-C_1} = 0.22229$,
- $u_1^{C_1}(40) = w_{11,2}^{C_1} + w_{12,3}^{C_1} = 0.0002$,
- $u_1^{-C_1}(40) = w_{13,4}^{-C_1} + w_{14,5}^{-C_1} + w_{15,6}^{-C_1} = 0.18521$,
- $u_1^{C_1}(61) = w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1} = 0.09892$,
- $u_1^{-C_1}(61) = w_{14,5}^{-C_1} + w_{15,6}^{-C_1} = 0.11114$,
- $u_1^{C_1}(66) = w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1} + w_{14,5}^{C_1} = 0.09902$,
- $u_1^{-C_1}(66) = w_{15,6}^{-C_1} = 0.07406$,
- $u_1^{C_1}(70) = w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1} + w_{14,5}^{C_1} + w_{15,6}^{C_1} = 0.19773$,
- $u_1^{-C_1}(70) = 0$;

- Criterion g_2:
  - $u_2^{C_1}(15) = 0$,
  - $u_2^{-C_1}(15) = w_{21,2}^{-C_1} + w_{22,3}^{-C_1} + w_{23,4}^{-C_1} + w_{24,5}^{-C_1} + w_{25,6}^{-C_1} = 0.33333$,
  - $u_2^{C_1}(20) = w_{21,2}^{C_1} = 0.0001$,
  - $u_2^{-C_1}(20) = w_{22,3}^{-C_1} + w_{23,4}^{-C_1} + w_{24,5}^{-C_1} + w_{25,6}^{-C_1} = 0.29625$,
  - $u_2^{C_1}(40) = w_{21,2}^{C_1} + w_{22,3}^{C_1} = 0.0002$,
  - $u_2^{-C_1}(40) = w_{23,4}^{-C_1} + w_{24,5}^{-C_1} + w_{25,6}^{-C_1} = 0.25917$,
  - $u_2^{C_1}(50) = w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1} = 0.09892$,
  - $u_2^{-C_1}(50) = w_{24,5}^{-C_1} + w_{25,6}^{-C_1} = 0.18511$,
  - $u_2^{C_1}(62) = w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1} + w_{24,5}^{C_1} = 0.23462$,
  - $u_2^{-C_1}(62) = w_{25,6}^{-C_1} = 0.07406$,
  - $u_2^{C_1}(64.75) = w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1} + w_{24,5}^{C_1} + w_{25,6}^{C_1} = 0.33333$,
  - $u_2^{-C_1}(64.75) = 0$;
- Criterion g_3:
  - $u_3^{C_1}(20) = 0$,
  - $u_3^{-C_1}(20) = w_{31,2}^{-C_1} + w_{32,3}^{-C_1} + w_{33,4}^{-C_1} + w_{34,5}^{-C_1} + w_{35,6}^{-C_1} = 0.40730$,
  - $u_3^{C_1}(23.125) = w_{31,2}^{C_1} = 0.0001$,
  - $u_3^{-C_1}(23.125) = w_{32,3}^{-C_1} + w_{33,4}^{-C_1} + w_{34,5}^{-C_1} + w_{35,6}^{-C_1} = 0.37021$,
  - $u_3^{C_1}(30) = w_{31,2}^{C_1} + w_{32,3}^{C_1} = 0.09882$,
  - $u_3^{-C_1}(30) = w_{33,4}^{-C_1} + w_{34,5}^{-C_1} + w_{35,6}^{-C_1} = 0.29615$,
  - $u_3^{C_1}(37) = w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1} = 0.19753$,
  - $u_3^{-C_1}(37) = w_{34,5}^{-C_1} + w_{35,6}^{-C_1} = 0.22209$,
  - $u_3^{C_1}(46.25) = w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1} + w_{34,5}^{C_1} = 0.33323$,
  - $u_3^{-C_1}(46.25) = w_{35,6}^{-C_1} = 0.11104$,
  - $u_3^{C_1}(60) = w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1} + w_{34,5}^{C_1} + w_{35,6}^{C_1} = 0.46893$,
  - $u_3^{-C_1}(60) = 0$;

Multicriteria Sorting Methods, Table 3
Global utilities obtained through the solution of LP1 (stage 1)

	$U^{C_1}(a)$	$U^{-C_1}(a)$
a_1	0.8643	0.1110
a_2	0.8025	0.1852
a_3	0.2967	0.5924
a_4	0.0993	0.7034
a_5	0.0002	0.9258
a_6	0.0988	0.8889

According to these marginal utilities, the global utilities are calculated based on the expressions that have already been presented. Table 3, illustrates the obtained global utilities according to the two utility functions that were developed.

It is clear that a_1 and a_2 are classified in class C_1, since the global utility of a decision concerning the classification of these two alternatives in class C_1 is greater than the utility concerning their classification in classes C_2 or C_3. Similarly, alternatives a_3, a_4, a_5 and a_6 are not classified in class C_1, but instead they belong in one of the classes C_2 or C_3 (their specific classification will be determined in the next stage of the hierarchical discrimination process).

Since the correct discrimination between the alternatives belonging in class C_1 and the alternative not belonging in this class has been achieved through LP1, it is not necessary to proceed in LP2 (minimization of the number of misclassifications). Hence, the procedure proceeds in the formulation and solution of LP3 in order to achieve the higher possible discrimination:

$$
\begin{cases}
\max & d \\
\text{s.t.} & U^{C_1}(a_1) - U^{-C_1}(a_1) - d \geq 0.001 \\
& U^{C_1}(a_2) - U^{-C_1}(a_2) - d \geq 0.001 \\
& U^{-C_1}(a_3) - U^{C_1}(a_3) - d \geq 0.001 \\
& U^{-C_1}(a_4) - U^{C_1}(a_4) - d \geq 0.001 \\
& U^{-C_1}(a_5) - U^{C_1}(a_5) - d \geq 0.001 \\
& U^{-C_1}(a_6) - U^{C_1}(a_6) - d \geq 0.001 \\
& w_{ij,j+1}^{C_1} \geq 0.0001, \quad w_{ij,j+1}^{-C_1} \geq 0.0001 \\
& \sum_{i=1}^{3} \sum_{j=1}^{5} w_{ij,j+1}^{C_1} = 1, \quad \sum_{i}^{3} \sum_{j=1}^{5} w_{ij,j+1}^{-C_1} = 1, \\
& \forall i = 1, 2, 3, \quad \forall j = 1, \ldots, 6, \quad d \geq 0.
\end{cases}
$$

According to the obtained solution and following the same procedure for calculating the marginal utili-

Multicriteria Sorting Methods, Table 4
Global utilities obtained through the solution of LP3 (stage 1)

	$U^{C_1}(a)$	$U^{-C_1}(a)$
a_1	0.9985	0.0001
a_2	0.9987	0.0003
a_3	0.0008	0.9992
a_4	0.0009	0.9993
a_5	0.0002	0.9998
a_6	0.0002	0.9998

ties, the global utilities of Table 4 are obtained. Obviously, this new solution provides a better discrimination of the alternatives, compared to the initial solution obtained by LP1.

Distinguishing Between C_2 and C_3

After the solution of LP3, the first stage of the hierarchical discrimination process is completed, with the correct classification of a_1 and a_2 in class C_1. Consequently, these two alternatives are excluded from further consideration (second stage). In the second stage, the aim is to determine the specific classification of the alternatives a_3, a_4, a_5 and a_6. The following rank-order is defined on the scale of the three evaluation criteria ($p_1 = p_2 = p_3 = 4$).

$g_1)$ $g_1^1 = 15 < 20 < 40 < 66 = g_1^{p_1}$;
$g_2)$ $g_2^1 = 15 < 20 < 40 < 50 = g_2^{p_2}$;
$g_3)$ $g_3^1 = 20 < 23.125 < 30 < 37 = g_2^{p_2}$.

Then, following the procedure illustrated in the previous stage, the variables $w_{ij,j+1}^{C_1}$ and $w_{ij,j+1}^{-C_1}$ are formulated, and the new form of the LP1 problem is the following ($s = 0.001$, $t = 0.0001$):

$$
\begin{cases}
\min & F = e(a_3) + e(a_4) + e(a_5) + e(a_6) \\
\text{s.t.} & U^{C_2}(a_3) - U^{-C_2}(a_3) + e(a_3) \geq 0.001 \\
& U^{C_2}(a_4) - U^{-C_2}(a_4) + e(a_4) \geq 0.001 \\
& U^{-C_2}(a_5) - U^{C_2}(a_5) + e(a_5) \geq 0.001 \\
& U^{-C_2}(a_6) - U^{C_2}(a_6) + e(a_6) \geq 0.001 \\
& w_{ij,j+1}^{C_2} \geq 0.0001, \quad w_{ij,j+1}^{-C_2} \geq 0.0001 \\
& \sum_{i=1}^{3}\sum_{j=1}^{3} w_{ij,j+1}^{C_2} = 1, \\
& \sum_{i}^{3}\sum_{j=1}^{3} w_{ij,j+1}^{-C_2} = 1, \\
& \forall i = 1,2,3, \quad \forall j = 1,\ldots,4, \\
& e(a_3), e(a_4), e(a_5), e(a_6) \geq 0.
\end{cases}
$$

Table 5 presents the global utilities of the alternatives according to the solution obtained by LP1 in this second stage.

The alternatives are correctly classified in their original classes, and therefore, it is not necessary to proceed with LP2 (similarly to the first stage). Instead, the method proceeds in solving LP3 to achieve better discrimination of the alternatives.

$$
\begin{cases}
\max & d \\
\text{s.t.} & U^{C_2}(a_3) - U^{-C_2}(a_3) - d \geq 0.001 \\
& U^{C_2}(a_4) - U^{-C_2}(a_4) - d \geq 0.001 \\
& U^{-C_2}(a_5) - U^{C_2}(a_5) - d \geq 0.001 \\
& U^{-C_2}(a_6) - U^{C_2}(a_6) - d \geq 0.001 \\
& w_{ij,j+1}^{C_2} \geq 0.0001, \quad w_{ij,j+1}^{-C_2} \geq 0.0001, \\
& \sum_{i=1}^{3}\sum_{j=1}^{3} w_{ij,j+1}^{C_2} = 1 \\
& \sum_{i=1}^{3}\sum_{j=1}^{3} w_{ij,j+1}^{-C_2} = 1, \\
& \forall i = 1,2,3, \quad \forall j = 1,\ldots,4, \\
& d \geq 0.
\end{cases}
$$

Table 6 presents the global utilities calculated according to the solution of LP3.

In this point the hierarchical discrimination procedure ends, since all the alternatives have been classified in the three predefined classes. Moreover, this classification is correct. In particular, in stage 1 a_1 and a_2 have

Multicriteria Sorting Methods, Table 5
Global utilities obtained through the solution of LP1 (stage 2)

	$U^{C_2}(a)$	$U^{-C_2}(a)$
a_3	0.8944	0.1000
a_4	0.7333	0.2501
a_5	0.2111	0.8000
a_6	0.1612	0.7500

Multicriteria Sorting Methods, Table 6
Global utilities obtained through the solution of LP3 (stage 2)

	$U^{C_2}(a)$	$U^{-C_2}(a)$
a_3	0.9999	0.0005
a_4	0.9997	0.0003
a_5	0.0002	0.9996
a_6	0.0005	0.7949

been correctly classified in class C_1, while in stage 2 a_3 and a_4 have been correctly classified in class C_2, and a_5 and a_6 have been classified into the final class C_3 (cf. Table 6).

Concluding Remarks and Future Perspectives

The focal point of interest in this article was the application of MCDA in the study of sorting or more generally discrimination (classification) problems. Such types of problems have major practical interest in several fields including finance, environmental and energy policy and planning, marketing, medical diagnosis, robotics (pattern recognition), etc. The multivariate statistical classification techniques have been used for decades to study such problems. However, their inability to provide a realistic and flexible approach to support real world decision making problems in situations where classification is required, led operational researchers, management scientists as well as practitioners towards the exploitation of the recent advances in the fields of operations research, management science, and artificial intelligence.

Among these 'alternative' approaches for the study of classification problem, MCDA provides an arsenal of tools and methods to develop classification (sorting) models within a realistic and flexible context. This article outlined the main MCDA classification techniques, both from the specific type of classification problems that they address (ordered or non-ordered classes), as well as from the MCDA approach that they employ (goal programming, outranking relations, preference disaggregation).

Furthermore, a new MCDA approach has been proposed. The M.H.DIS method, extends the common two-group classification framework, through a hierarchical multigroup discrimination procedure, taking into account three main discrimination criteria through a sequential process. In this way the classification problem is studied globally, in order to achieved the higher possible classification accuracy. Except for the illustrative example used in this paper, the M.H.DIS method has already been used in several financial classification problems, including the evaluation of bankruptcy risk, portfolio selection and management, the evaluation of bank branches efficiency, the assessment of country risk, company mergers and acquisitions, etc. [43], providing very encouraging results compared to well known statistical techniques (discriminant analysis, logit and probit analysis), and MCDA preference disaggregation techniques (family of UTADIS methods).

An interesting further research direction would be the exploration of a possible combination of M.H.DIS with artificial intelligence techniques such as fuzzy sets, in order to consider the fuzziness which may exist on the evaluation of alternatives on each evaluation criterion, or on the classification of the alternatives.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Abad PL, Banks WJ (1993) New LP based heuristics for the classification problem. Europ J Oper Res 67:88–100
2. Bajgier SM, Hill AV (1982) A comparison of statistical and linear programming approaches to the discriminant problem. Decision Sci 13:604–618
3. Banks WJ, Abad PL (1991) An efficient optimal solution algorithm for the classification problem. Decision Sci 22:1008–1023
4. Charnes A, Cooper WW (1961) Managem. models and industrial applications of linear programming. Wiley, New York
5. Choo E-U, Wedley WC (1985) Optimal criterion weights in repetitive multicriteria decision-making. J Oper Res Soc 36(11):983–992
6. Devaud JM, Groussaud G, Jacquet–Lagrèze E (1980) UTADIS: Une méthode de construction de fonctions d'utilité additives rendant compte de jugements globaux. Europ Working Group on Multicriteria Decision Aid, Bochum
7. Doumpos M, Zopounidis C (1998) The use of the preference disaggregation analysis in the assessment of financial risks. Fuzzy Economic Rev 3(1):39–57
8. Fisher RA (1936) The use of multiple measurements in taxonomic problems. Ann Eugenics 7:179–188
9. Freed N, Glover F (1981) A linear programming approach to the discriminant problem. Decision Sci 12:68–74
10. Freed N, Glover F (1981) Simple but powerful goal programming models for discriminant problems. Europ J Oper Res 7:44–60
11. Freed N, Glover F (1986) Evaluating alternative linear programming models to solve the two-group discriminant problem. Decision Sci 17:151–162
12. Glover F (1990) Improved linear programming models for discriminant analysis. Decision Sci 21:771–785
13. Glover F, Keene S, Duea B (1988) A new class of models for the discriminant problem. Decision Sci 19:269–280
14. Gochet W, Stam A, Srinivasan V, Chen S (1997) Multigroup discriminant analysis using linear programming. Oper Res 45(2):213–225
15. Horsky D, Rao MR (1984) Estimation of attribute weights from preference comparisons. Managem Sci 30(7):801–822
16. Jacquet–Lagrèze E (1995) An application of the UTA discriminant model for the evaluation of R&D projects. In: Pardalos PM, Siskos Y, Zopounidis C (eds) Advances in Multicriteria Analysis. Kluwer, Dordrecht, pp 203–211
17. Jacquet–Lagrèze E, Siskos Y (1982) Assessing a set of additive utility functions for multicriteria decision making, the UTA method. Europ J Oper Res 10:151–164
18. Joachimsthaler EA, Stam A (1988) Four approaches to the classification problem in discriminant analysis: An experimental study. Decision Sci 19:322–333
19. Khoury NT, Martel JM (1990) The relationship between risk-return characteristics of mutual funds and their size. Finance 11(2):67–82
20. Koehler GJ, Erenguc SS (1990) Minimizing misclassifications in linear discriminant analysis. Decision Sci 21:63–85
21. Mangasarian OL (1968) Multisurface method for patter separation. IEEE Trans Inform Theory IT-14(6):801–807
22. Markowski CA, Markowski EP (1987) An experimental comparison of several approaches to the discriminant problem with both qualitative and quantitative variables. Europ J Oper Res 28:74–78
23. Markowski EP, Markowski CA (1985) Some difficulties and improvements in applying linear programming formulations to the discriminant problem. Decision Sci 16:237–247
24. Massaglia M, Ostanello A (1991) N-TOMIC: A decision support for multicriteria segmentation problems. In: Korhonen P (ed) Internat. Workshop Multicriteria Decision Support, vol 356. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 167–174
25. Mousseau V, Slowinski R (1998) Inferring an ELECTRE-TRI model from assignment examples. J Global Optim 12(2):157–174
26. Nakayama H, Kagaku N (1992) Pattern classification by linear goal programming and its extensions. J Global Optim 12(2):111–126
27. Pekelman D, Sen SK (1974) Mathematical programming models for the determination of attribute weights. Managem Sci 20(8):1217–1229
28. Perny P (1998) Multicriteria filtering methods based on concordance and non-discordance principles. Ann Oper Res 80:137–165
29. Ragsdale CT, Stam A (1991) Mathematical programming formulations for the discriminant problem: An old dog does new tricks. Decision Sci 22:296–307
30. Roy B (1985) Méthodologie multicritère d'aide à la décision. Economica, Paris
31. Roy B, Moscarola J (1977) Procédure automatique d'examem de dossiers fondée sur une segmentation trichotomique en présence de critéres multiples. RAIRO Rech Opérat 11(2):145–173
32. Rubin PA (1990) A comparison of linear programming and parametric approaches to the two- group discriminant problem. Decision Sci 21:373–386
33. Rubin PA (1990) Heuristic solution procedures for a mixed-integer programming discriminant model. Managerial and Decision Economics 11:255–266
34. Smith C (1947) Some examples of discrimination. Ann Eugenics 13:272–282
35. Srinivasan V, Shocker AD (1973) Estimating the weights for multiple attributes in a composite criterion using pairwise judgements. Psychometrika 38(4):473–493
36. Srinivasan V, Shocker AD (1973) Linear programming techniques for multidimensional analysis of preferences. Psychometrika 38(3):337–396

37. Stam A, Joachimsthaler EA (1989) Solving the classification problem via linear and nonlinear programming methods. Decision Sci 20:285–293
38. Stam A, Joachimsthaler EA (1990) A comparison of a robust mixed-integer approach to existing methods for establishing classification rules for the discriminant problem. Europ J Oper Res 46:113–122
39. Wilson JM (1996) Integer programming formulation of statistical classification problems. OMEGA Internat J Management Sci 24(6):681–688
40. Yu W (1992) ELECTRE TRI: Aspects methodologiques et manuel d'utilisation. Document du Lamsade (Univ Paris-Dauphine) 74
41. Zopounidis C, Doumpos M (1997) A multicriteria decision aid methodology for the assessment of country risk. In: Zopounidis C, Garcia Vázquez JM (eds) Managing in Uncertainty. Proc VI Internat Conf AEDEM, AEDEM Ed, pp 223–236
42. Zopounidis C, Doumpos M (1997) Preference disaggregation methodology in segmentation problems: The case of financial distress. In: Zopounidis C (ed) New Operational Approaches for Financial Modelling. Physica Verlag, Heidelberg, pp 417–439
43. Zopounidis C, Doumpos M (1998) A multi-group hierarchical discrimination method for managerial decision problems: The M.H.DIS method. In: Paper Presented at the EURO XVI Conf.: Innovation and Quality of Life, Brussels, 12–15 July

Multidimensional Assignment Problem

MAP

Alla R. Kammerdiner
Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA

MSC2000: 90C10, 90C27

Article Outline

Keywords and Phrases
Introduction
Formulation
Cases
Methods
Applications
See also
References

Keywords and Phrases

Multidimensional assignment problem; Zero-one integer programming; Combinatorial optimization

Introduction

The multidimensional assignment problem (MAP) can be viewed as a higher-dimensional extension of the linear assignment problem (LAP). While the LAP is often explained as assigning each person in a group a specific job so that for each job there is only one person who does it, and for each person there is only one assigned job. The MAP generalizes evidently two-dimensional (people, jobs) LAP by allowing additional dimensions (space, time, etc.) Hence, the previous example of scheduling people to jobs can be extended to scheduling people to jobs at various time intervals in different locations, so that each specific parameter (say, time interval) is coupled with its own unique three other parameters (person, job, location) and none of them are in any other assignment (of a person, a job, a time slot and a person). Such a modified assignment problem is an example of a MAP in four dimensions.

Obviously, the LAP is a special case of the MAP in two dimensions. On the other hand, the MAP (sometimes referred to as multi-index assignment problem) is a special case of the multi-index transportation problem, just like the LAP is a particular instance of the more general transportation problem.

Interestingly, a broader class of multidimensional transportation problems was originally considered about a decade before the LAP was first given its multidimensional generalization. In fact, a three-dimensional case of the multi-index transportation problem was first introduced by Schell in 1955 [33], and later by Haley [19] in 1963. The MAP was initially presented by Pierskalla [26] in 1966, through first extending the LAP to its three-dimensional case, and then (in 1968) as a general formulation of MAP in n dimensions [27].

Despite the fact that the LAP can be solved in polynomial time, the MAP of dimensionality $d \geq 3$ is known to be NP-hard in general (the latter statement follows from a reduction of the matching problem in three dimensions) [16]. In fact, the size of the MAP increases extremely fast with an increase in di-

mensions. To be more precise, the size of problem grows by products of factorials. As a result of the inherent complexity of the problem, only small to medium-sized instances of the MAP can be solved routinely at the moment. Most of the exact and heuristic algorithms developed for this problem are enumerative in nature and/or utilize some form of local neighborhood search. Although many real-life applications of the MAP, including the data association problem in target tracking, require solving general problems of dimensions higher than three, most of the proposed solution methods deal with widely studied three-dimensional versions such as axial 3-MAP and planar 3-MAP.

Formulation

Several alternative formulations of the MAP have been given since Pierskalla introduced it as a 0-1 integer programming problem as follows.

Given $1 \le p_1 \le \ldots \le p_d \le n$, a finite sequence of positive integers, we want to

$$
\begin{aligned}
\text{minimize} \quad & \sum_{1\le i_1\le p_1} \cdots \sum_{1\le i_d\le p_d} c_{i_1\ldots i_d} \cdot x_{i_1\ldots i_d} \\
\text{subject to} \quad & \sum_{1\le i_2\le p_2} \cdots \sum_{1\le i_d\le p_d} x_{i_1\ldots i_d} \\
& \qquad = 1, \ 1 \le i_1 \le p_1 , \\
& \sum_{1\le i_1\le p_1} \cdots \sum_{1\le i_{k-1}\le p_{k-1}} \sum_{1\le i_{k+1}\le p_{k+1}} \cdots \\
& \qquad \sum_{1\le i_d\le p_d} x_{i_1\ldots i_d} = 1 , \\
& \qquad 1 \le i_k \le p_k , \quad 2 \le k \le d-1 , \\
& \sum_{1\le i_1\le p_1} \cdots \sum_{1\le i_{d-1}\le p_{d-1}} x_{i_1\ldots i_d} = 1 , \\
& \qquad 1 \le i_d \le p_d , \\
& x_{i_1\ldots i_d} \in \{0, 1\}, \quad 1 \le i_k \le p_k, \ 1 \le k \le d ,
\end{aligned}
\tag{1}
$$

where $c_{i_1\ldots i_d}$ are the cost coefficients.

By introducing dummy variables, we can assume without loss of generality that $p_1 = \ldots = p_d = n$; then the d-dimensional assignment problem can be reformulated as follows:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{1\le i_1\le n} \cdots \sum_{1\le i_d\le n} c_{i_1\ldots i_d} \cdot x_{i_1\ldots i_d} \\
\text{subject to} \quad & \sum_{1\le i_2\le n} \cdots \sum_{1\le i_d\le n} x_{i_1\ldots i_d} = 1 , \quad 1 \le i_1 \le n , \\
& \sum_{1\le i_1\le n} \cdots \sum_{1\le i_{k-1}\le n} \\
& \sum_{1\le i_{k+1}\le n} \cdots \sum_{1\le i_d\le n} x_{i_1\ldots i_d} = 1 , \\
& 1 \le i_k \le n , \quad 2 \le k \le d-1 , \\
& \sum_{1\le i_1\le n} \cdots \sum_{1\le i_{d-1}\le n} x_{i_1\ldots i_d} = 1 , \quad 1 \le i_d \le n , \\
& x_{i_1\ldots i_d} \in \{0, 1\} , \quad 1 \le i_k \le n , \quad 1 \le k \le d .
\end{aligned}
\tag{2}
$$

The MAP (2) also has an interesting interpretation as a problem of combinatorial optimization:

Given a d-dimensional cubic matrix, one must find the permutation of its columns and rows with the minimum sum of the diagonal elements. In other words, this is an equivalent characterization of (2) in terms of $d-1$ permutations $\pi_1, \pi_2, \ldots, \pi_{d-1}$ of the set $\{1, 2, \ldots n\}$:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{1\le i\le n} c_{i\pi_1(i)\ldots\pi_{d-1}(i)} , \\
\text{subject to} \quad & \pi_1, \ \pi_2 , \ \ldots , \ \pi_{d-1} \in \Pi^n ,
\end{aligned}
\tag{3}
$$

where Π^n is the set of all permutations of $\{1, 2, \ldots n\}$.

Spieksma [34] gives an alternative compact formulation of the MAP as follows:

Given d sets $A_1, A_2, \ldots, A_d$, each of size n, let $A = \otimes_{i=1}^{d} A_i = A_1 \times A_2 \times \ldots \times A_d$. In other words, A is a set of all d-tuples $a = (a(1), a(2), \ldots, a(d)) \in A$. Let x_a denote a variable for each $a \in A$. Then, given assignment costs c_a for all $a \in A$, the objective function is written as $\sum_{a\in A} c_a x_a$.

Given a positive integer k, such that $1 \le k \le d-1$, let Q denote the set of all $(d-k)$-element subsets of $\{1, 2, \ldots, d\}$. Each subset F from Q corresponds to the set of "fixed" indices. Given such F, let $A_F = \otimes_{f\in F} A_f$. Next, given some $g \in A_F$, let $A(F, g) = \{a \in A | a(f) = g(f), \forall f \in F\}$ denote the set of all d-tuples that coincide with g on the set F of "fixed" indices.

Then the multi-index assignment problem can be written as:

$$\begin{aligned} \text{maximize/minimize} \quad & \sum_{a \in A} c_a x_a \\ \text{subject to} \quad & \sum_{a \in A(F,g)} x_a = 1, \\ & \text{for all } g \in A_F,\ F \in Q\,, \\ & x_a \in \{0,\ 1\},\ \text{for all } a \in A\,. \end{aligned} \tag{4}$$

Similarly to the linear assignment formulation by means of a bipartite graph, the MAP can also be stated using the graph theory terminology in the subsequent fashion [7]:

Given a complete d-partite graph $G = (V_1, V_2, \ldots, V_d; E)$, where V_i, $|V_i| = n$, $i \in \{1, 2, \ldots, d\}$, denote mutually disjoint vertex sets, and E is the set of edges in the graph, a subset of the vertex set $V = \cup_{i=1}^{d} V_i$ is said to be a *clique* if it meets every set V_i in exactly one vertex. A d-dimensional assignment is a partition of V into n pairwise disjoint cliques. Given a real-valued cost function c defined on the set of cliques of d-partite graph G, the d-dimensional assignment problem asks for a d-dimensional assignment, which minimizes c.

Cases

A special case of the MAP that is based on the graph theory formulation for MAP was considered by Bandelt et al. [6]. The cost function in this particular case can be represented using some type of function of elementary costs defined on the edges of the d–partite graph, whereas a general formulation of the MAP using graphs allows for the cost function to be defined arbitrarily on the set of cliques. In particular, the clique costs can be decomposed using such functions of edge costs as a sum of costs (i. e., a sum of the lengths of all the edges in a given clique), a tour cost (i. e., minimum cost of a traveling salesman tour in a given clique), a star cost (i. e., minimum length of a spanning star in a given clique), and a tree cost (i. e., minimum cost of a spanning tree). By using the decomposed costs, one can construct the worst-case bounds on the ratio between the solution costs found by a simple heuristic, as well as find the cost of the optimal solution. Specifically, Crama and Spieksma [10] considered a case of three-dimensional assignment problem, where the lengths of the edges of the underlying three-partite graph satisfy the triangle inequality, and the objective function is defined as the cost of the triangle formed by three vertices (each from a different mutually disjoint vertex subsets of the three-partite graph). When the triangle cost is defined as the length of the triangle (i. e., sum of the lengths of all its sides), then there exists a heuristic that gives a feasible solution that is within 3/2 from the optimum. The latter bound is decreased to 4/3 in the case when the triangle cost is defined as the sum of the two shortest sides.

As mentioned earlier, owing to the exponential increase in the size of the problem with an increase in the number of parameters, it becomes computationally difficult to solve MAP instances of higher dimensionality. As a result most solution methods for the MAP are constructed for three-dimensional versions of the problem. Two important types of the three-dimensional assignment problem are the *axial* three-dimensional assignment problem and the *planar* three-dimensional assignment problem. The distinction between two types lies in constraints and can be easily explained using the following simple geometric interpretation [7].

Let each solution be represented by a three-dimensional 0-1 array of size $n \times n \times n$. To visualize such an array of zeros and ones, let us fix a vertex and draw lines or axes along three dimensions. Next, we partition each axis onto n intervals. This partition splits the array into n^3 cells so that each cell contains either a 0 or a 1. Given an axis, say j, each of n intervals on j has a corresponding two-dimensional level surface that consists of $n \times n$ cells and goes through a given interval of j. Alternatively, the interval partition of each axis divides a three-dimensional solution array into n two-dimensional surfaces or "slices" corresponding to each interval on the axis. The constraints imposed in the axial case guarantee that for each axis and all of its intervals, the $n \times n$ cells in each two-dimensional slice through the interval sum up to 1. In other words, each axial interval is assigned a value of 1, which constitutes the sum of all cells that can be projected on that axial interval. This explains the name "axial."

In contrast, the constraints of the planar MAP deal with three planes formed by each possible pair of axes. For example, consider the plane formed by axes j and k. Using the above partition, this plane is divided into $n \times n$ squares. For each square on the plane, there is a corresponding stack of cells that goes along the i axis. Each cell in the stack is projected onto its square (j^*, k^*)

by axis i. The planar constraints require that for each plane and every square the sum of the cells in an associated stack is equal to 1.

Integer programming formulations for each of type of three-index MAP are given below.

Given a set of n^3 cost coefficients c_{ijk}, the axial three-dimensional assignment problem is defined as follows:

$$\begin{aligned}
&\text{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}x_{ijk} \\
&\text{subject to} \quad \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1\,, \quad 1 \le i \le n\,, \\
&\qquad\qquad \sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1\,, \quad 1 \le j \le n\,, \\
&\qquad\qquad \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk} = 1\,, \quad 1 \le k \le n\,, \\
&\qquad\qquad x_{ijk} \in \{0,\,1\}\,, \quad 1 \le i,\,j,\,k \le n\,.
\end{aligned} \tag{5}$$

Given n^3 cost coefficients c_{ijk}, the planar three-dimensional assignment problem can be written in the following fashion:

$$\begin{aligned}
&\text{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}x_{ijk} \\
&\text{subject to} \quad \sum_{i=1}^{n} x_{ijk} = 1, \quad 1 \le j,k \le n\,, \\
&\qquad\qquad \sum_{j=1}^{n} x_{ijk} = 1\,, \quad 1 \le i,\,k \le n\,, \\
&\qquad\qquad \sum_{k=1}^{n} x_{ijk} = 1\,, \quad 1 \le i,\,k \le n\,, \\
&\qquad\qquad x_{ijk} \in \{0,\,1\}\,, \quad 1 \le i,\,j,\,k \le n\,.
\end{aligned} \tag{6}$$

The axial three-index MAP given by (5) can also be formulated using n permutations σ and π as a combinatorial optimization problem:

$$\text{minimize} \quad \sum_{i=1}^{n} c_{i\sigma(i)\pi(i)}\,, \ \text{subject to } \sigma,\,\pi \in \Pi_n\,. \tag{7}$$

Note that the planar three-dimensional assignment problem has a different combinatorial interpretation in terms of Latin squares of order n.

Although both axial and planar three-dimensional assignment problems (just as the general MAP) are generally NP-hard, there exist a number of polynomially solvable special cases. Particularly, in the case when the cost coefficients form a so-called Monge array [8], the MAP is solved by $d-1$ identity-n permutations $\{1,\,2,\,\ldots,\,n\} \to \{1,\,2,\,\ldots,\,n\}$. Another case of the polynomially solvable MAP is the axial three-dimensional assignment problem, where the cost coefficient can be represented as a product of nonnegative index factors $c_{ijk} = p_i \cdot q_j \cdot r_k$, and the objective function is maximized [9].

Methods

All known exact methods for solving this generally NP-hard problem are enumerative in nature, and as a result of the inherent complexity of the problem such methods are too slow for practical applications of the MAP. Hence, researchers often use heuristic approaches to find suboptimal solutions of different MAPs. In fact, one of the earliest solution methods for the MAP was a suboptimal method of trisubstitution proposed by Pierskalla [26] in 1966 to solve a three-dimensional assignment problem. Later Frieze and Yadegar [15] developed a suboptimal procedure for the three-index MAP using Lagrangian relaxation. Their technique utilized information contained in the relaxed solution to recover a feasible solution. The key advantage of the Lagrangian relaxation approach is that it allows for computing both upper and lower bounds on the optimum solution, and therefore this method can be employed to evaluate solution quality. Consequently, the Lagrangian relaxation technique was widely used to propose numerous modifications of the original three-dimensional method by extending it to the general multidimensional case [12,28,29]. For example, one of such algorithms presented by Poore and Robertson [29] in 1997 works by relaxing a d-dimensional assignment problem to a two-dimensional problem, then maximizing with regard to the relaxed Lagrangian multipliers, and next formulating the recovery procedure as a $(d-1)$-dimensional problem. These three steps are repeated successively until the recovery procedure can be formulated as a two-dimensional problem, which is solved optimally in polynomial time, and the algorithm terminates.

Most exact methods for solving the MAP are devised primarily for its three-dimensional case. One of

the earliest exact approaches for the axial three-dimensional assignment problem was suggested by Pierskalla [27] in 1968. His approach works by enumerating all feasible solutions using a tree structure, and utilizing the branch and bound method as follows. For a given node of the feasible solutions tree, a lower bound is calculated from the corresponding dual subproblem, before proceeding further on outgoing branches from the node. If the lower bound is greater than the known lowest bound, then the outgoing branches are eliminated, since it is impossible to obtain a better solution along such branches. Otherwise when the lower bound obtained is less than the known lowest bound, we continue further from this node, because it might still be possible to improve our solution in that direction. Although this branch and bound algorithm can easily be generalized to the multidimensional case, it is too slow to work effectively for the general MAP.

Since Pierskalla introduced his branch and bound procedure for the axial three-dimensional assignment problem, many other branch and bound based approaches have been developed. Most of them branch the current problem onto two subproblems by setting one variable $x_{ijk} = 0$ or $x_{ijk} = 1$. Then the size of the subproblems is decreased. In contrast, a branch and bound scheme proposed by Balas and Saltzman [5] permits fixing several variables at once at each branching node by incorporating a special branching strategy that takes advantage of the problem structure.

The planar three-index MAP can also be solved using variations of branch and bound. One of the first applications of this method to the planar case was given by Vlach [35] in 1967. The algorithm obtains lower bounds by means of row and column reductions that are similar to the ones in the axial case. A method for solving the planar three-dimensional assignment problem based on a clever combination of branch and bound with a relaxation heuristic and Lagrangian relaxation was developed by Magos and Miliotis [24]. The upper bounds are calculated by first applying the relaxation heuristic and then decomposing the remaining problem into n linear sum assignment problems. The lower bounds are computed by either a heuristic or a Lagrangian relaxation depending on the current problem.

The method introduced by Hansen and Kaufman [20] for solving the axial three-dimensional assignment problem employs a primal-dual method comparable to the well-known Hungarian method for the LAP.

There have been a number of investigations of a convex hull of feasible solutions of the three-dimensional assignment problem. Euler et al. [14] examined the polyhedral structure of the solution polytope for the planar three-index MAP through its connection to Latin squares. Euler [13] also studied the axial polytope by investigating the role of odd cycles for a class of facets of the polytope. The structure of the axial three-index assignment polytope was also analyzed by Balas et al. [3,4,32]. They developed linear-time separation algorithms for different classes of facets induced by specific cliques, and then constructed a polyhedral procedure for solving the axial three-index MAP.

Clemons et al. [11] applied a simulated annealing algorithm for solving the MAP. Several local neighborhood search procedures were implemented for the MAP. Greedy randomized adaptive search procedures (GRASP) were applied by Murphey et al. [25] for solving the general MAP and later by Lidstrom et al. [22] and by Aiex et al. [1] for finding solutions of the axial three-dimensional assignment problem. A tabu search for the planar three-dimensional assignment problem was employed by Magos [23] to obtain suboptimal solutions of the planar thee-index assignment problem.

Grundel and Pardalos [18] developed a test problem generator for testing exact and suboptimal solution methods for the axial MAP. Several recent studies investigated various asymptotic properties of the MAPs with randomly generated assignment cost coefficients. In particular, Grundel et al. [17] established the lower and upper bounds for the expected number of local minima of the MAPs with random costs.

Applications

The MAP can be used to solve various real-life problems arising in such important areas as capital investment, dynamic facility location, and satellite launching [30]. Other applications of the MAP include circuit board assembly and production planning of goods, which can be modeled using the axial three-dimensional assignment problem [34]. The planar three-dimensional assignment problem has also found many interesting applications, for instance, school timetables

and experimental design [21], as well as modeling of satellite launching [2].

Furthermore, it was shown that such a complex problem as tracking elementary particles can be investigated using the five-dimensional assignment problem as a mathematical model [31]. By solving this complex case of the MAP, one can reconstruct the paths of charged elementary particles produced by the Large Electron–Positron Collider.

Many important applications of the general MAP arise in data association, resource allocation, air traffic control, surveillance, etc. In particular, Poore [28] has shown that the data association problem arising in a large class of multiple target tracking and sensor fusion problems can be formulated as a MAP by partitioning the set of observations into false reports and tracks, and then maximizing the likelihood of selecting the true partition.

See also

- ► Assignment and Matching
- ► Integer Programming: Branch and Bound Methods
- ► Multi-index Transportation Problems

References

1. Aiex RM, Resende MGC, Pardalos PM, Toraldo G (2005) GRASP with path relinking for the three-index assignment. INFORMS J Comput 17(2):224–247
2. Balas E, Landweer P (1983) Traffic Assignment in Communication Satelites. Oper Res Lett 2:141–147
3. Balas E, Qi L (1993) Linear-time separation algorithms for three-index assignment polytope. Discr Appl Math 43: 1–12
4. Balas E, Saltzman MJ (1989) Facets of the three-index assignment polytope. Discr Appl Math 23:201–229
5. Balas E, Saltzman MJ (1991) An algorithm for three-index assignment problem. Oper Res 39:150–161
6. Bandelt HJ, Crama Y, Spieksma FCR (1994) Approximation algorithms for multidimensional assignment problems with decomposable costs. Discr Appl Math 49:25–50
7. Burkard RE, Çela E (1999) Linear Assignment Problems and Extensions. In: Pardalos P, Du D-Z (eds) The Handbook of Combinatorial Optimization. Kluwer, Dordrecht, pp 75–149
8. Burkard RE, Klinz B, Rudolf R (1996) Perspectives of Monge properties in optimization. Discr Appl Math 70:95–161
9. Burkard RE, Rudolf R, Woeginger GJ (1996) Three-dimensional axial problem with decomposable cost coefficients. Discr Appl Math 65:123–169
10. Crama Y, Spieksma FCR (1992) Approximation algorithms for three-dimensional assignment problems with triangle inequalities. Eur J Oper Res 60:273–279
11. Clemons W, Grundel D, Jeffcoat D (2004) Applying simulated annealing to the multidimensional assignment problem. In: Grundel DA, Pardalos PM, Murphey R (eds) Theory and Algorithms for Cooperative Systems. World Scientific, Singapore
12. Deb S, Pattipatti KR, Bar-Shalom Y (1992), A s-dimensional assignment algorithm for track initiation. Proceedings of the IEEE International Systems Conference, Kobe, Japan, September 127–130
13. Euler R (1987) Odd cycles and a class of facets of the axial 3-index assignment polytope. Applicationes mathimaticae (Zastowania Matematyki) 19:375–386
14. Euler R, Burkard RE, Grommes R (1986) On Latin squares and facial structure of related polytopes. Discr Math 62:155–181
15. Frieze AM, Yadegar J (1981) An algorithm for solving 3-dimensional assignment problems with application to scheduling a teaching practice. J Oper Res Soc 32:969–995
16. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco
17. Grundel D, Krokhmal P, Oliveira C, Pardalos P (2007) On the number of local minima for the multidimensional assignment problem. J Comb Optim 13(1):1–18
18. Grundel D, Pardalos P (2005) Test Problem Generator for the Multidimensional Assignment Problem. Comput Optim Appl 30(2):133–146
19. Haley KB (1963) The Multi-index Problem. Oper Res 11:368–379
20. Hansen P, Kaufman L (1973) A primal-dual algorithm for three-dimensional assignment problem. Cahiers du CERO 15:327–336
21. Hilton A (1980) The reconstruction of Latin Squares with Applications to School Timetabling and to Experimental Desin
22. Lidstrom N, Pardalos P, Pitsoulis L, Toraldo G (1996) An approximation algorithm for the three-index assignment problem, Technical Report
23. Magos D (1996) Tabu search for the planar three-dimensional assignment problem. J Global Optim 8:35–48
24. Magos D, Miliotis P (1994) An algorithm for the planar three-index assignment problem. Eur J Oper Res 77:141–153
25. Murphey R, Pardalos P, Pitsoulis L (1998) A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. DIMACS Ser Am Math Soc 40:277–302
26. Pierskalla WP (1967) The tri-substitution method for the three-multidimensional assignment problem. CORS J 5:71–81
27. Pierskalla WP (1968) The multidimensional assignment problem. Oper Res 16:422–431

28. Poore AB (1994) Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking. Comput Optim Appl 3:27–54
29. Poore AB, Robertson AJ (1997) A New Lagrangian Relaxation Based Algorithm for a Class of Multidimensional Assignment Problems. Comput Optim Appl 8:129–150
30. Pierskalla WP (1967) The tri-substitution method for the three-dimensional assignment problem. J Canadian Oper Res Soc 5:71–81
31. Pusztaszeri JF, Rensing PE, Liebling TM (1996) Tracking Elementary Particles Near Their Primary Vertex: A Combinatorial Approach. J Global Optim 9(1):41–64
32. Qi L, Balas E, Gwan G (1994) A new facet class and a polyhedral method for three-index assignment problem. In: Du D-Z (ed) Advances in Optimization. Kluwer, Dordrecht, pp 256–274
33. Schell E (1955) Distribution of a Product by Several Properties, Proc. Second Symposium in Linear Programming, Washington, D.C., January 27-29, vol 1–2, pp 615–642
34. Spieksma FCR (2000) Multi index assignment problems: complexity, approximation, applications. In: Pitsoulis L, Pardalos P (eds) Nonlinear Assignment Problems, Algorithms and Applications. Kluwer, Dordrecht, pp 1–12
35. Vlach M (1967) Branch and bound method for the three-index assignment problem. Economicko-Matematicky Obzor 3:181–191

Multidimensional Knapsack Problems

JOHN E. BEASLEY
The Management School, Imperial College, London, England

MSC2000: 90C27, 90C10

Article Outline

Keywords
Exact Algorithms
Statistical/Asymptotic Analysis
Early Heuristic Algorithms
Bound Based Heuristics
Tabu Search Heuristics
Genetic Algorithm Heuristics
Analysed Heuristics
Other Heuristics
Multiple–Choice Problems
See also
References

Keywords

Multidimensional knapsack; Multiconstraint knapsack; Multiple choice knapsack; Combinatorial optimization

The *multidimensional knapsack problem* (MKP) can be formulated as:

$$\begin{cases} \max & \sum_{j=1}^{n} p_j x_j \\ \text{s.t.} & \sum_{j=1}^{n} r_{ij} x_j \le b_i, \quad i = 1, \ldots, m, \\ & x_j \in \{0,1\}, \quad j = 1, \ldots, n, \end{cases} \tag{1}$$

where $b_i \ge 0$, $i = 1, \ldots, m$, and $r_{ij} \ge 0$, $i = 1, \ldots, m$, $j = 1, \ldots, n$.

Each of the m constraints in (1) is called a *knapsack constraint*, so the MKP is also called the *m-dimensional knapsack problem*.

Other names given to this problem in the literature are the *multiconstraint knapsack problem*, the *multi-knapsack problem* and the *multiple knapsack problem*. Some authors also include the term 'zero-one' in their name for the problem, e. g., the *multidimensional zero-one knapsack problem*. Historically the majority of authors have used the name *multidimensional knapsack problem* and so we also use that phrase to refer to the problem. The special case corresponding to $m = 2$ is known as the *bidimensional knapsack problem* or the *bi-knapsack problem*.

Many practical problems can be formulated as a MKP, for example, the capital budgeting problem where project j has profit p_j and consumes r_{ij} units of resource i. The goal is to find a subset of the n projects such that the total profit is maximised and all resource constraints are satisfied. Other applications of the MKP include allocating processors and databases in a distributed computer system [24], project selection and cargo loading [53], and cutting-stock problems [26].

The MKP can be regarded as a general statement of any zero-one integer programming problem with non-negative coefficients. Indeed much of the early work on the MKP (e. g., [32,35,52,59]) viewed the problem in this way.

Most of the research on knapsack problems deals with the much simpler single constraint version ($m = 1$). For the single constraint case the problem is not

strongly *NP*-hard and effective approximation algorithms have been developed for obtaining near-optimal solutions. A good review of the single constraint knapsack problem and its associated exact and heuristic algorithms is given by S. Martello and P. Toth [42].

Below we give a very brief overview of the literature relating to the MKP. A more detailed literature review can be found in [10].

Exact Algorithms

There have been relatively few exact algorithms presented in the literature.

W. Shih [53] presented a branch and bound algorithm (cf. also ► Integer programming: Branch and bound methods) for the MKP with an upper bound obtained by computing the objective function value associated with the optimal fractional solution for each of the m single constraint knapsack problems separately and selecting the minimum objective function value among those as the upper bound.

Another branch and bound algorithm was presented in [25] with various relaxations of the problem, including Lagrangian, surrogate and composite relaxations being used to compute bounds. Y. Crama and J.B. Mazzola [11] showed that although the bounds derived from these relaxations are stronger than the bounds obtained from the linear programming (LP) relaxation, the improvement in the bound that can be realized using these relaxations is limited.

Statistical/Asymptotic Analysis

There have been a few papers considering a statistical/asymptotic analysis of the MKP.

An asymptotic analysis was presented by K.E. Schilling [51] who computed the asymptotic ($n \to \infty$ with m fixed) objective function value for the MKP where the r_{ij}'s and p_j's were uniformly (and independently) distributed over the unit interval and where $b_i = 1$. K. Szkatula [54] generalized that analysis to the case where $b_i \neq 1$ (see also [55]).

A statistical analysis was conducted by J.F. Fontanari [18], who investigated the dependence of the objective function on b_i and on m, in the case when $p_j = 1$ and the r_{ij}'s were uniformly distributed over the unit interval.

Early Heuristic Algorithms

Early heuristic algorithms for the MKP were typically based upon simple constructive heuristics.

S.H. Zanakis [59] gave detailed results comparing three algorithms from [32,35] and [52]. R. Loulou and E. Michaelides [40] presented a greedy-like method based on *Toyoda's primal heuristic* [57]. Primal heuristics start with a zero solution, after which a succession of variables are assigned the value one, according to a given rule, as long as the solution remain feasible.

Bound Based Heuristics

Bound based heuristics make use of an upper bound on the optimal solution to the MKP.

M.J. Magazine and O. Oguz [41] presented a heuristic algorithm that combines the ideas of S. Senju and Toyoda's dual heuristic [52] with *Everett's generalized Lagrange multiplier approach* [17]. Dual heuristics start with the all-ones solution, variables are then successively set to zero according to heuristic rules until a feasible solution is obtained. Their algorithm computes an approximate solution and uses the multipliers generated to obtain an upper bound.

H. Pirkul [45] presented a heuristic algorithm which makes use of surrogate duality. The m knapsack constraints were transformed into a single knapsack constraint using surrogate multipliers. A feasible solution was obtained by packing this single knapsack in decreasing order of profit/weight ratios. These ratios were defined as $p_j / \sum_{i=1}^{m} \omega_i r_{ij}$, where ω_i is the surrogate multiplier for constraint i. Surrogate multipliers were determined using a descent procedure.

J.S. Lee and M. Guignard [36] presented a heuristic that combined Toyoda's primal heuristic [57] with variable fixing, LP and a complementing procedure from [6].

A. Volgenant and J.A. Zoon [58] extended the heuristic in [41] in two ways:

1) in each step, not one, but more, multiplier values are computed simultaneously; and
2) at the end of the procedure the upper bound is sharpened by changing some multiplier values.

A. Freville and G. Plateau [21] presented an efficient preprocessing algorithm for the MKP, based on [20], which provided sharp lower and upper bounds on the optimal value, and also a tighter equivalent represen-

tation by reducing the continuous feasible set and by eliminating constraints and variables.

They also [22] presented a heuristic for the bidimensional knapsack problem which includes problem reduction, a bound based upon surrogate relaxation and partial enumeration.

Tabu Search Heuristics

Tabu search (TS) heuristics are based on tabu search concepts (see [1,29,46]).

F. Dammeyer and S. Voß [12] presented a TS heuristic based on reverse elimination. R. Aboudi and K. Jörnsten [2] combined TS with the pivot and complement heuristic [6] in a heuristic that they applied to the MKP (see also [39]). R. Battiti and G. Tecchiolli [7] presented a heuristic based on reactive TS (essentially TS but with the length of the tabu list varied over the course of the algorithm).

F. Glover and G.A. Kochenberger [28] presented a TS heuristic with a flexible memory structure that integrates recency and frequency information keyed to 'critical events' in the search process. Their method was enhanced by a strategic oscillation scheme that alternates between constructive (current solution feasible) and destructive (current solution infeasible) phases. See also [30].

A. Løkketangen and Glover [37] presented a heuristic based on probabilistic TS (essentially TS but with the acceptance/rejection of a potential move controlled by a probabilistic process). They also [38] presented a TS heuristic designed to solve general zero-one mixed integer programming problems which they applied to the MKP.

Genetic Algorithm Heuristics

Genetic algorithm (GA) heuristics are based on genetic algorithm concepts (see [1,8,43,46]).

In the GA of [34] infeasible solutions were allowed to participate in the search and a simple fitness function which uses a graded penalty term was used. In [56] simple heuristic operators based on local search algorithms were used, and a hybrid algorithm based on combining a GA with a TS heuristic was suggested.

In [48,49] a GA was presented where parent selection is not unrestricted (as in a standard GA) but is restricted to be between 'neighboring' solutions. Infeasible solutions were penalized as in [34]. An adaptive threshold acceptance schedule (motivated by [14,15]) for child acceptance was used.

In the GA of [33] only feasible solutions were allowed. P.C. Chu and J.E. Beasley [10] presented a GA based upon a simple repair operator to ensurethat all solutions were feasible.

Analysed Heuristics

Analysed heuristics have some theoretical underlying analysis relating to their worst-case or probabilistic performance.

A.M. Frieze and M.R.B. Clarke [23] described a polynomial approximation scheme based on the use of the dual simplex algorithm for LP, and analysed the asymptotic properties of a particular random model.

In [47] a class of generalized greedy algorithms is proposed in which items are selected according to decreasing ratios of their p_j's and a weighted sum of their r_{ij}'s. These heuristics were subjected to both a worst-case, and a probabilistic, performance analysis.

I. Averbakh [5] investigated the properties of several dual characteristics of the MKP for different probabilistic models. He also presented a fast statistically efficient approximate algorithm with linear running time complexity for problems with random coefficients.

Other Heuristics

G.E. Fox and G.D. Scudder [19] presented a heuristic based on starting from setting all variables to zero(one) and successively choosing variables to set to one(zero). See [13] for a heuristic based upon simulated annealing (SA). See [27] for a heuristic based on ghost image processes. S. Hanafi and others [31] presented a simple multistage algorithm within which a number of different local search procedures (such as greedy, SA, threshold accepting [14,15] and noising [9]) can be used. They also presented two TS heuristics.

Multiple–Choice Problems

One problem that is related to the MKP is the *multidimensional multiple-choice knapsack problem* (MMKP). Suppose that $\{1, \ldots, n\}$ is divided up into K sets S_k, $k = 1, \ldots, K$, which are mutually exclusive $S_k \cap S_l = \emptyset$, $\forall k \neq l$, and exhaustive $\cup_{k=1}^{K} S_k = \{1, \ldots, n\}$. If we then

add to the formulation of the MKP given previously the constraint

$$\sum_{j \in S_k} x_j = 1, \quad k = 1, \ldots, K, \tag{2}$$

we obtain the MMKP. Equation (2) ensures that exactly one variable is chosen from each of the sets S_k, $k = 1, \ldots, K$.

See [44] for a heuristic for MMKP based on the MKP heuristic of Magazine and Oguz [41].

The special case of the MMKP corresponding to $m = 1$ is known as the *multiple-choice knapsack problem* (MCKP) and its LP relaxation as the *linear multiple-choice knapsack problem* (LMCKP). Work on MCKP includes [16], which presented a hybrid dynamic programming tree search algorithm incorporating a Lagrangian relaxation bound; [4], which presented a heuristic based upon SA; and [3], which presented a tree search algorithm incorporating a Lagrangian relaxation bound. For work on LMCKP see [50]. Earlier work on MCKP and LMCKP is cited in [3,4,16,50].

See also

- ► Integer Programming
- ► Quadratic Knapsack

References

1. Aarts EHL, Lenstra J (eds) (1997) Local search in combinatorial optimization. Wiley, New York
2. Aboudi R, Jörnsten K (1994) Tabu search for general zero-one integer programs using the pivot and complement heuristic. ORSA J Comput 6:82–93
3. Aggarwal V, Deo N, Sarkar D (1992) The knapsack problem with disjoint multiple-choice constraints. Naval Res Logist 39:213–227
4. Al-Sultan K (1994) A new approach to the multiple-choice knapsack problem. Proc 16th Internat Conf Computers and Industr Engineering, pp 548–550
5. Averbakh I (1994) Probabilistic properties of the dual structure of the multidimensional knapsack problem and fast statistically efficient algorithms. Math Program 65:311–330
6. Balas E, Martin CH (1980) Pivot and complement – a heuristic for 0–1 programming. Managem Sci 26:86–96
7. Battiti R, Tecchiolli G (1995) Local search with memory: Benchmarking RTS. OR Spektrum 17:67–86
8. Bäck T, Fogel DB, Michalewicz Z (eds) (1997) Handbook of evolutionary computation. Oxford Univ. Press, Oxford
9. Charon I, Hudry O (1993) The noising method: A new method for combinatorial optimization. Oper Res Lett 14:133–137
10. Chu PC, Beasley JE (1998) A genetic algorithm for the multidimensional knapsack problem. J Heuristics 4:63–86
11. Crama Y, Mazzola JB (1994) On the strength of relaxations of multidimensional knapsack problems. INFOR 32:219–25
12. Dammeyer F, Voss S (1993) Dynamic tabu list management using reverse elimination method. Ann Oper Res 41:31–46
13. Drexl A (1988) A simulated annealing approach to the multiconstraint zero-one knapsack problem. Computing 40:1–8
14. Dueck G (1993) New optimization heuristics: the grand deluge algorithm and the record-to-record travel. J Comput Phys 104:86–92
15. Dueck G, Scheuer T (1990) Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. J Comput Phys 90:161–175
16. Dyer ME, Riha WO, Walker J (1995) A hybrid dynamic programming/branch-and-bound algorithm for the multiple-choice knapsack problem. J Comput Appl Math 58:43–54
17. Everett H (1963) Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Oper Res 11:399–417
18. Fontanari JF (1995) A statistical analysis of the knapsack problem. J Phys A: Math Gen 28:4751–4759
19. Fox GE, Scudder GD (1985) A heuristic with tie breaking for certain 0–1 integer programming models. Naval Res Logist Quart 32:613–623
20. Freville A, Plateau G (1986) Heuristics and reduction methods for multiple constraints 0–1 linear programming problems. Europ J Oper Res 24:206–215
21. Freville A, Plateau G (1994) An efficient preprocessing procedure for the multidimensional 0–1 knapsack problem. Discrete Appl Math 49:189–212
22. Freville A, Plateau G (1997) The 0–1 bidimensional knapsack problem: toward an efficient high-level primitive tool. J Heuristics 2:147–167
23. Frieze AM, Clarke MRB (1984) Approximation algorithms for the m-dimensional 0–1 knapsack problem: worst-case and probabilistic analysis. Europ J Oper Res 15:100–109
24. Gavish B, Pirkul H (1982) Allocation of databases and processors in a distributed computing system. In: Akoka J (ed) Managem. of Distributed Data Processing. North-Holland, Amsterdam, pp 215–231
25. Gavish B, Pirkul H (1985) Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. Math Program 31:78–105
26. Gilmore PC, Gomory RE (1966) The theory and computation of knapsack functions. Oper Res 14:1045–1075
27. Glover F (1994) Optimization by ghost image processes in neural networks. Comput Oper Res 21:801–822
28. Glover F, Kochenberger GA (1996) Critical event tabu search for multidimensional knapsack problems. In: Os-

man IH, Kelly JP (eds) Meta–Heuristics: Theory and Applications. Kluwer, Dordrecht, pp 407–427
29. Glover FW, Laguna M (1997) Tabu search. Kluwer, Dordrecht
30. Hanafi S, Freville A (1998) An efficient tabu search approach for the 0–1 multidimensional knapsack problem. Europ J Oper Res 106:659–675
31. Hanafi S, Freville A, AbedellaouiEl A (1996) Comparison of heuristics for the 0–1 multidimensional knapsack problem. In: Osman IH, Kelly JP (eds) Meta–Heuristics: Theory and Applications. Kluwer, Dordrecht, pp 449–465
32. Hillier FS (1969) Efficient heuristic procedures for integer linear programming with an interior. Oper Res 17:600–637
33. Hoff A, Løkketangen A, Mittet I (1996) Genetic algorithms for 0/1 multidimensional knapsack problems. Working Paper, Molde College, Molde
34. Khuri S, Bäck T, Heitkötter J (1994) The zero/one multiple knapsack problem and genetic algorithms. Proc. 1994 ACM Symp. Applied Computing (SAC'94), ACM, New York, pp 188–193
35. Kochenberger GA, McCarl BA, Wymann FP (1974) A heuristic for general integer programming. Decision Sci 5:36–44
36. Lee JS, Guignard M (1988) An approximate algorithm for multidimensional zero-one knapsack problems - a parametric approach. Managem Sci 34:402–410
37. Løkketangen A, Glover F (1996) Probabilistic move selection in tabu search for zero-one mixed integer programming problems. In: Osman IH, Kelly JP (eds) Meta-Heuristics: Theory and Applications. Kluwer, Dordrecht, pp 467–487
38. Løkketangen A, Glover F (1997) Solving zero-one mixed integer programming problems using tabu search. Europ J Oper Res 106:624–658
39. Løkketangen A, Jörnsten K, Storøy S (1994) Tabu search within a pivot and complement framework. Internat Trans Oper Res 1:305–316
40. Loulou R, Michaelides E (1979) New greedy-like heuristics for the multidimensional 0–1 knapsack problem. Oper Res 27:1101–1114
41. Magazine MJ, Oguz O (1984) A heuristic algorithm for the multidimensional zero-one knapsack problem. Europ J Oper Res 16:319–326
42. Martello S, Toth P (1990) Knapsack problems: Algorithms and computer implementations. Wiley, New York
43. Mitchell M (1996) An introduction to genetic algorithms. MIT, Cambridge
44. Moser M, Jokanovic DP, Shiratori N (1997) An algorithm for the multidimensional multiple-choice knapsack problem. IEICE Trans Fundam Electronics, Commun and Computer Sci E80A:582–589
45. Pirkul H (1987) A heuristic solution procedure for the multiconstraint zero-one knapsack problem. Naval Res Logist 34:161–172
46. Reeves CR (1993) Modern heuristic techniques for combinatorial problems. Blackwell, Oxford
47. Rinnooy Kan AHG, Stougie L, Vercellis C (1993) A class of generalized greedy algorithms for the multi-knapsack problem. Discrete Appl Math 42:279–290
48. Rudolph G, Sprave J (1995) A cellular genetic algorithm with self-adjusting acceptance threshold. Proc. First IEE/IEEE Internat. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications, IEEE, New York, pp 365–372
49. Rudolph G, Sprave J (1996) Significance of locality and selection pressure in the grand deluge evolutionary algorithm. In: Voigt HM, Ebeling W, Rechenberg I, Schwefel HP (eds) Parallel Problem Solving from Nature IV Proc Internat Conf Evolutionary Computation, Lecture notes Computer Sci. Springer, Berlin, pp 686–694
50. Sarin S, Karwan MH (1989) The linear multiple choice knapsack problem. Oper Res Lett 8:95–100
51. Schilling KE (1990) The growth of m-constraint random knapsacks. Europ J Oper Res 46:109–112
52. Senju S, Toyoda Y (1968) An approach to linear programming with 0–1 variables. Managem Sci 15:196–207
53. Shih W (1979) A branch and bound method for the multiconstraint zero-one knapsack problem. J Oper Res Soc 30:369–378
54. Szkatula K (1994) The growth of multi-constraint random knapsacks with various right-hand sides of the constraints. Europ J Oper Res 73:199–204
55. Szkatula K (1997) The growth of multi-constraint random knapsacks with large right-hand sides of the constraints. Oper Res Lett 21:25–30
56. Thiel J, Voss S (1994) Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithms. INFOR 32:226–242
57. Toyoda Y (1975) A simplified algorithm for obtaining approximate solutions to zero-one programming problems. Managem Sci 21:1417–1427
58. Volgenant A, Zoon JA (1990) An improved heuristic for multidimensional 0–1 knapsack problems. J Oper Res Soc 41:963–970
59. Zanakis SH (1977) Heuristic 0–1 linear programming: An experimental comparison of three methods. Managem Sci 24:91–104

Multidisciplinary Design Optimization

MDO

Layne T. Watson
Virginia Polytechnic Institute and State University, Blacksburg, USA

MSC2000: 65F10, 65F50, 65H10, 65K10

Article Outline

Keywords
MDO Paradigm Example
See also
References

Keywords

Collaborative; Concurrent subspace; Multidisciplinary design; Multipoint approximation; Response surface; DACE

Modern large scale vehicle design (aircraft, ships, automobiles, mass transit) requires the interaction of multiple disciplines, traditionally processed in a sequential order. *Multidisciplinary optimization* (MDO), a formal methodology for the integration of these disciplines, is evolving toward methods capable of replacing the traditional sequential methodology of vehicle design by concurrent algorithms, with both an overall gain in product performance and a decrease in design time. The obstacles to MDO becoming a production methodology, in the same sense as quality control, are numerous and formidable. In aircraft design, for instance, typical disciplines involved would be aerodynamics, structures, thermodynamics, controls, propulsion, manufacture, and economics. Detailed analyses in each of these disciplines could involve tens to hundreds of subroutines and tens of thousands of lines of code. Managing the software libraries and data alone is a daunting task.

Codes from different disciplines typically are grossly incompatible, but even within disciplines, data structures and solution representations may be incompatible, requiring 'translation' routines or recoding. This incompatibility is particularly acute when stand-alone packages with interactive interfaces are involved. Most disciplinary codes, designed years ago for small serial computers, are very ill-suited to modern parallel architectures, even with a coarse grained approach.

Detailed, highly accurate disciplinary analyses are very expensive, requiring sometimes hours on a supercomputer, even when run in parallel. The import of this is that, regardless of the dimension of the design space, it can be sampled for accurate function values at only a relatively small number of points. Other obstacles to achieving true MDO include model verification, noisy function values, and flawed parallel optimization methodologies.

Almost every conceivable strategy for MDO has been proposed. A good recent summary of hierarchical approaches can be found in [4], and [9] pioneered nonhierarchical or concurrent approaches. The basic idea of concurrent methods, and a particular variant known as *concurrent subspace optimization* (CSSO), is to simultaneously and independently optimize each of the disciplines (or 'contributing analyses', as they are called), and then perform a global coordination that brings the entire system closer to a globally feasible and optimal point. *Collaborative optimization* differs from CSSO in how the global coordination is managed. An excellent discussion of these approaches is in the proceedings [2]. While concurrent methods are intuitively appealing and naturally parallelizable, they are not guaranteed to converge [8].

Trust region model management [1] is a rigorous approach to MDO that shows promise, and aspects of CSSO when combined with an extended Lagrangian and response surface approximations, can lead to a provably convergent MDO method (J.F. Rodríguez, J.E. Renaud and L.T. Watson, [6]). A noteworthy aspect of the *Rodríguez method* [6] is that the convergence proof covers variable fidelity data, which is crucial in practice.

In a taxonomy of MDO approaches, one distinction would be between hierarchic or nonhierarchic. Another distinction is whether parallelism is achieved between disciplines (concurrent disciplinary computation) or within disciplines (multipoint, response surface, local/global computation). If response surface approximations are used, two prevalent approximation methods are classical least squares and *DACE* (Design and Analysis of Computer Experiments).

S. Burgee, A.A. Giunta, V. Balabanov, B. Grossman, W.H. Mason, R. Narducci, R.T. Haftka, and Watson [3] has a detailed discussion of the multipoint, classical least squares approach to response surface construction, and of the use of parallelism within disciplines (the pipelined MDO paradigm of Burgee is also provably convergent). The tack of this approach is to use classical design of experiments theory, regression statistics, and low order polynomial approximation models.

The DACE [7] model posits that the output of a computer analysis program is

$$Y(x) = \beta + Z(x),$$

where $Z(x)$ is a zero mean stationary Gaussian process. (This is clearly a fiction since computer output is deterministic. The issue is whether the model has predictive power.) Using Bayesian statistics, the best unbiased predictor is

$$\widehat{Y}(x) = \widehat{\beta} + r(x, S)R^{-1}(Y_S - 1 \cdot \widehat{\beta}),$$

where S is a set of observation sites, Y_S is the vector of observations at S, $r(x, S)$ is the correlation of x with sites S, R is the correlation matrix between sites S, and $\widehat{\beta}$ is the estimate of the mean. Some parametrized functional form for the correlation is assumed, and then these correlation parameters and $\widehat{\beta}$ are computed as maximum likelihood estimates.

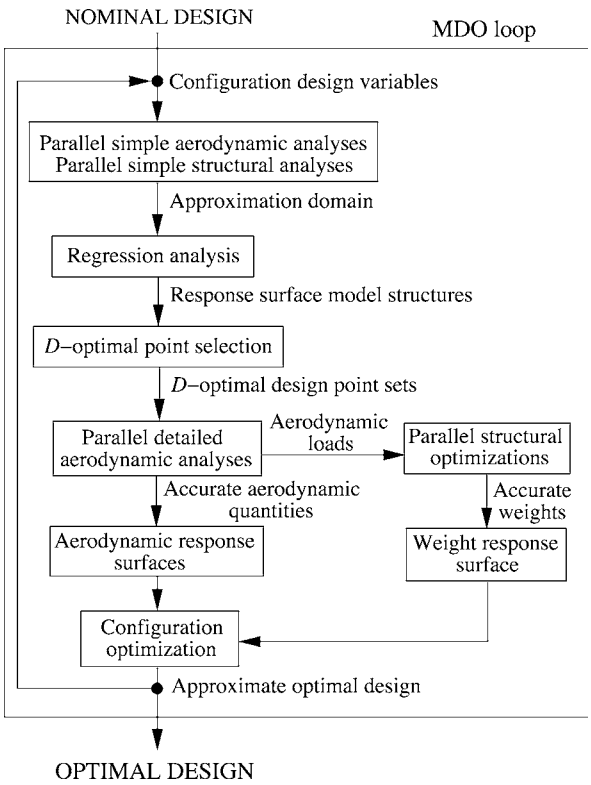
DACE models are more flexible than polynomial models, but with sparse data in high dimensions neither DACE nor polynomial models have much predictive power. To appreciate the problem, observe that a cube in 30 dimensions has $2^{30} \approx 10^9$ vertices, and to even evaluate an algebraic formula at each vertex requires supercomputer power.

MDO Paradigm Example

As an illustration, an MDO paradigm for aircraft design is presented here. The MDO algorithm is a repeat loop, with a nominal design as its starting point, approximate optimal designs as loop iterates, and an optimal design as its ending point (see Fig. 1). At the start of each loop, aerodynamic shape and mission variables are obtained from either the nominal starting design or the intermediate approximate optimal design. These shape and mission variables are then used in the parallel simple aerodynamic and structural analyses.

The simple aerodynamic analyses are performed on a regular grid of points in the design space. Simple aerodynamic calculations evaluate the (aerodynamic) feasibility of each grid point using tolerances on the constraints and move limits on the objective function, eliminating grossly infeasible points, and generating an approximation domain. The simple structural analyses use the aerodynamic shape and mission variables in basic weight equations to calculate approximate weights needed by the objective function and constraints, further refining the approximation domain.

Using the relatively abundant data from the simple analyses, regression analysis and analysis of variance



Multidisciplinary Design Optimization, Figure 1
MDO paradigm

are used to identify less important terms in the polynomial response surface models. Once the less important terms are eliminated, the structure of the reduced-term polynomial regression models is known, and can be used later in the generation of response surface approximations of the optimal weight and necessary aerodynamic quantities over the approximation domain.

A genetic algorithm (GA; cf. ► Genetic Algorithms) is used to find sets of approximate D-optimal design points in the approximation domain obtained from the parallel simple analyses. The structure of a response surface model is embodied in the regression matrix $\mathbf{X}$, which defines the GA merit function $|\mathbf{X}^{\mathrm{T}}\mathbf{X}|$ (maximized by a set of points called D-optimal). These D-optimal design points are input to the detailed aerodynamic analysis code, which performs detailed analyses at each of the D-optimal design points in parallel. The analyses result in accurate aerodynamic quantities, such as wave drag and other drag components, and accurate aerodynamic loads.

The accurate aerodynamic quantities are used to generate reduced-term polynomial response surface models for each of the expensive quantities (such as wave drag). An aerodynamic load calculated in the detailed aerodynamic analyses is used in a detailed structural optimization to calculate an accurate optimal weight for that particular aerodynamic load. This structural optimization is done (in parallel) for each aerodynamic load generated in the detailed aerodynamic analyses. The accurate optimal weights calculated in the structural optimization are used to generate a reduced-term polynomial response surface model for the optimal weight.

All the response surface models are then used in a configuration optimization to generate an approximate optimal design, which will be used as the starting design for the next iteration of the MDO loop. The grid spacing may possibly be refined for the simple analyses. When some convergence criterion is satisfied, the MDO loop exits with an optimal design.

Note that the *source of parallelism in the present MDO paradigm is the multipoint approximations within each discipline*, where the disciplines are visited sequentially in a pipeline. This contrasts sharply with CSSO MDO paradigms, where the *source of the parallelism is processing the disciplines in parallel.*

See also

- ► Bilevel Programming: Applications in Engineering
- ► Design Optimization in Computational Fluid Dynamics
- ► Interval Analysis: Application to Chemical Engineering Design Problems
- ► Multilevel Methods for Optimal Design
- ► Optimal Design of Composite Structures
- ► Optimal Design in Nonlinear Optics
- ► Structural Optimization: History

References

1. Alexandrov N (1996) Robustness properties of a trust region framework for managing approximations in engineering optimization. Proc. 6th AIAA/NASA/USAF Multidisciplinary Analysis and Optimization Symposium, 96-4102 AIAA, 1056–1059
2. Alexandrov N, Hussaini MY (eds) (1997) Multidisciplinary design optimization, state-of-the-art. SIAM, Philadelphia
3. Burgee S, Giunta AA, Balabanov V, Grossman B, Mason WH, Narducci R, Haftka RT, Watson LT (1996) A coarse grained parallel variable-complexity multidisciplinary optimization paradigm. Internat J Supercomputer Appl High Performance Comput 10:269–299
4. Cramer EJ, Dennis JE Jr, Frank PD, Lewis RM, Shubin GR (1994) Problem formulation for multidisciplinary optimization. SIAM J Optim 4:754–776
5. Renaud JE, Gabriele GA (1991) Sequential global approximation in non-hierarchic system decomposition and optimization. Adv Design Automation 1:191–200
6. Rodríguez JF, Renaud JE, Watson LT (1998) Convergence of trust region augmented Lagrangian methods using variable fidelity approximation data. Structural Optim 15:141–156
7. Sacks J, Welch WJ, Mitchell TJ, Wynn HP (1989) Design and analysis of computer experiments. Statistical Sci 4:409–435
8. Shankar J, Ribbens CJ, Haftka RT, Watson LT (1993) Computational study of a nonhierarchical decomposition algorithm. Comput Optim Appl 2:273–293
9. Sobieszczanski-Sobieski J (1988) Optimization by decomposition: A step from hierarchic to non-hierarchic systems. Second NASA/Air Force Symposium on Recent Advances in Multidisciplinary Analysis and Optimization

Multifacility and Restricted Location Problems

MFR

Horst W. Hamacher, Stefan Nickel
Fachbereich Math., Universität Kaiserslautern, Kaiserslautern, Germany

MSC2000: 90B85

Article Outline

Keywords
See also
References

Keywords

Location theory; Weber problem; Weber–Rawls problem; multifacility Weber problem; multiWeber problem; Gauge; Convex polytope; Linear programming; NP-hard; Voronoi diagram; Restricted location problem; Finite dominating set; Discretization; Barrier location problems

In location planning one is typically concerned with finding a good location for one or several new facilities with respect to a given set of existing facilities (clients). The two most common models in planar location theory are the *Weber problem*, where the average (weighted) distance of the new to the existing facilities is taken into account and the *Weber–Rawls problem*, where the maximum (weighted) distance of the new to the existing facilities is taken into account.

More precisely, one is given a finite set $Ex = \{Ex_1, \ldots, Ex_M\}$ of existing facilities (represented by their geographical coordinates) in the plane $\mathbf{R}^2$ and distance functions d_m assigned to each existing facility $m \in \mathcal{M} := \{1, \ldots, M\}$. The set of locations for the N new facilities one is looking for is denoted $X = \{X_1, \ldots, X_N\}$. The distance between the new facilities is measured by a common distance d. Additionally, a value w_{mn} is assigned to each pair (Ex_m, X_n), for $m \in \mathcal{M}$, $n \in \mathcal{N} := \{1, \ldots, N\}$ and a value v_{rs} assigned to each pair (X_r, X_s), for $r, s \in \mathcal{N}$, $s > r$, reflecting the *level of interaction*.

With these definitions the *multifacility Weber objective function* can be written as

$$\sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} w_{mn} d_m(Ex_m, X_n) + \sum_{\substack{r,s\in\mathcal{N}\\ s>r}} v_{rs} d(X_r, X_s) := f(X_1, \ldots, X_N)$$

and the *multifacility Weber–Rawls objective function* can be written as

$$\max\left\{ \max_{\substack{m\in\mathcal{M}\\ n\in\mathcal{N}}} w_{mn} d_m(Ex_m, X_n),\ \max_{\substack{r,s\in\mathcal{N}\\ s>r}} v_{rs} d(X_r, X_s) \right\} := g(X_1, \ldots, X_N).$$

In the corresponding optimization problems we may additionally assume a feasible region $\mathcal{F}$ and we look for

$$\min_{\{X_1,\ldots,X_N\}\subset\mathcal{F}} f(X_1, \ldots, X_N),$$

and

$$\min_{\{X_1,\ldots,X_N\}\subset\mathcal{F}} g(X_1, \ldots, X_N).$$

In the first part of this survey it is assumed that $\mathcal{F} = \mathbf{R}^2$ whereas $\mathcal{F}$ will be a restricted set later on.

The models above implicitly assume that the new facilities can be distinguished, that the amount of interaction between each new and existing facility is known and that the new facilities have mutual communication. Note, that problems without communication between the new facilities can be separated into N independent 1-facility problems which can be easily solved by suitable algorithms. Also, in many applications we want to locate a number of indistinguishable facilities to serve the overall demand. This implies that we are not only locating facilities, but we are also allocating existing facilities (clients) to the new ones. This variation of the problem is called *multiWeber* or *multiWeber–Rawls problem* and the objective functions can be written as

$$\sum_{m\in\mathcal{M}} w_m d_m(Ex_m, \{X_1, \ldots, X_N\}) = \widehat{f}(X)$$

and

$$\max_{m\in\mathcal{M}} \{w_m d_m(Ex_m, \{X_1, \ldots, X_N\})\} = \widehat{g}(X),$$

respectively, where $d_m(Ex_m, \{X_1, \ldots, X_N\}) := \min_{Y\in X_1,\ldots,X_n\}} d_m(Ex_m, Y)$.

In order to discuss solution methods, suitable types of distance functions d_m, $m \in \mathcal{M}$, are specified next.

Let B be a compact convex set in the plane containing the origin in its interior and let Y be a point in the plane. The *gauge* of Y (with respect to B) is then defined as

$$\gamma_B(Y) := \inf\{\lambda > 0 :\ Y \in \lambda B\}.$$

This definition dates back to [25]. The distance from Ex_m to Y induced by γ_B is

$$d_m(Ex_m, Y) := \gamma_{B_m}(Y - Ex_m) \quad \text{for } m \in \mathcal{M}.$$

In the case where all B_m are convex polytopes with extreme points $\mathrm{Ext}(B_m) := \{e_1^m, \ldots, e_G^m\}$ we can define halflines l_i^m starting at Ex_m and going through e_i^m. For the 1-facility case it was proved in [6] for the Weber problem that there always exists an optimal solution in the set of intersection points of the halflines l_i^m for $i = 1, \ldots, G^m$ and $m \in \mathcal{M}$. This result carries over to multi-(facility) Weber problems when each B_m has no more than 4 extreme points [24]. For more than 4 extreme points it is in general wrong (see [24] for a counterexample).

In the case where all B_m are polytopes we can give linear programming formulations for the multifacility

Weber as well as the multifacility Weber–Rawls problem [34] using B_m^0, the polar set of B_m, $m \in \mathcal{M}$.

$$\begin{cases} \min & \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} w_{mn} z_{mn} + \sum_{\substack{r,s\in\mathcal{N}\\ s>r}} v_{rs} z'_{rs} \\ \text{s.t.} & \left\langle Ex_m - X_n, e_m^0 \right\rangle \le z_{mn}, \\ & \quad \forall m \in \mathcal{M},\ n \in \mathcal{N}\, e_m^0 \in \operatorname{Ext}(B_m^0), \\ & \left\langle X_s - X_r, e^0 \right\rangle \le z'_{rs}, \\ & \quad \forall s, r \in \mathcal{N},\ s > r,\ e^0 \in \operatorname{Ext}(B^0), \end{cases}$$

$$\begin{cases} \min & z \\ \text{s.t.} & w_{mn} \left\langle Ex_m - X_n, e_m^0 \right\rangle \le z, \\ & \quad \forall m \in \mathcal{M},\ n \in \mathcal{N},\ e_m^0 \in \operatorname{Ext}(B_m^0), \\ & v_{rs} \left\langle X_s - X_r, e^0 \right\rangle \le z, \\ & \quad \forall s, r \in \mathcal{N},\ s > r,\ e^0 \in \operatorname{Ext}(B^0). \end{cases}$$

Even without polyhedral structure we still have a convex optimization problem for which several solution techniques are available (see [11,12,21,32] and references therein).

In the case where we also have to deal with the allocation problem we still can apply discretization results from the 1-facility case. The allocation part makes the problem however *NP*-hard (see [22,23]; cf. also ► Complexity Theory; ► Complexity Classes in Optimization). Nevertheless, constructs from computational geometry (e. g. Voronoi diagrams; cf. also ► Voronoi Diagrams in Facility Location) can be used to tackle the allocation part efficiently and allow iterative heuristics producing in general satisfactory results (see [2,30]).

Further extensions are possible and already investigated including location with attraction and repulsion, hub location, etc. (see [32] for further references).

A problem common to all forms of multi-(facility) location problems is, that in an optimal solution locations of different new facilities may coincide with each other or with existing facilities. This raises at least two issues:

- A priori detection of coincidences which result in a reduction of the dimension of the problem and allow the exploitation of differentiability are discussed in [7,20,31].
- If coincidence is excluded, the theory of restricted location can be used which is discussed next.

So far, the set $\mathcal{F}$ for placing new facilities was the whole plane $\mathbf{R}^2$. Now, the feasibility set $\mathcal{F} = \mathbf{R}^2 \setminus \operatorname{int}(\mathcal{R})$ is considered, where $\mathcal{R} \subseteq \mathbf{R}^2$ is the restricting set assumed to be connected in $\mathbf{R}^2$. This problem is more complicated than the unrestricted one, since $\mathcal{F}$ is in general not convex. But from a practical point of view it is a necessary extension of the classical location model, since forbidden regions appear everywhere: nature reserves, lakes, exclusion of coincidence in multifacility, etc. These problems are called *restricted location problems* and have been developed in [1,12,14,15] and [26]. In the following we exclude the trivial case and assume that none of the optimal solutions of the unrestricted problem is a feasible solution of the restricted one.

If the objective function h of the location problem is convex it can be shown that optimal solutions of the restricted problem can be found on the boundary of $\mathcal{R}$. Therefore, level curves

$$L_=(z) := \{X \in \mathbb{R}^n :\ h(X) = z\}$$

and level sets

$$L_\le(z) := \{X \in \mathbb{R}^n :\ h(X) \le z\}$$

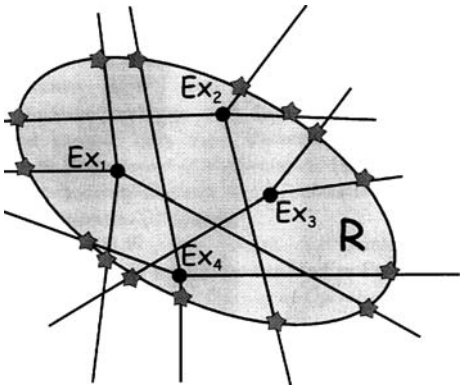
can be used to reformulate the restricted location problem as

$$\min \{z :\ L_=(z) \cap \partial\mathcal{R} \ne \emptyset \text{ and } L_\le(z) \subseteq \mathcal{R}\}.$$

A resulting search algorithm was formulated in [11], but proved to be inefficient in practical applications.

An efficient approach originally presented in [12,14,15] identifies *finite dominating set* (FDS) on the boundary $\mathcal{R}$, i. e. a finite set of locations on $\partial\mathcal{R}$ which contains an optimal solution. Using this discretization, problems with gauge distance and convex forbidden region can be solved by considering as FDS the intersection points of l_i^m and the boundary of $\mathcal{R}$ (see [15,26,28] and the illustration in the following figure).

The discretization also works for restricted center problems [16] and can be extended to nonconvex forbidden regions (see [15,26]) and also to the case of attraction and repulsion (negative weights are allowed), see [29]. The concept of forbidden regions has been successfully applied to a problem in PCB assembly, where the bins holding the parts to be inserted into the PCB have to be stored [10]. Of course, the PCB itself has to be forbidden for placing a bin. A solution approach, where also the issue of space requirements in a multifacility setting is addressed can be found in [9,15]. A more gen-



Multifacility and Restricted Location Problems, Figure 1
Example of a restricted location problem with 4 existing facilities and an elliptic forbidden region

eral case where the new facility is a line has been considered in [33]. Algorithms for multifacility problems with forbidden regions can be found in [8,15,27].

Another type of restricted location problem is one, where not only placement, but also tresspassing of regions is forbidden. These problems are called *barrier location problems*. The corresponding models are mathematically challenging, since the distance functions (and thus also the objective functions) are no longer convex. [17] considers Euclidean distances and one circle as forbidden region. [1] and [4] develop heuristics for l_p distances and barriers that are closed polygons. [19] and [3] obtain discretization results for l_1 distances and arbitrary shaped barriers by showing an equivalence of the barrier problem to a network location problem. In the more general context of gauge distances an FDS is given in [13] for median problems and in [5] for center problems. Finally, [18] considers barrier problems if the distance is an arbitrary norm and the barrier consists of a line with finitely many passages.

See also

- ▶ Combinatorial Optimization Algorithms in Resource Allocation Problems
- ▶ Competitive Facility Location
- ▶ Facility Location with Externalities
- ▶ Facility Location Problems with Spatial Interaction
- ▶ Facility Location with Staircase Costs
- ▶ Global Optimization in Weber's Problem with Attraction and Repulsion
- ▶ MINLP: Application in Facility Location-allocation
- ▶ Network Location: Covering Problems
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Production-distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Stochastic Transportation and Location Problems
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Aneja YP, Parlar M (1994) Algorithms for Weber facility location in the presence of forbidden regions and/or barriers to travel. Transport Sci 28:70–76
2. Aurenhammer F (1991) Voronoi diagrams – A survey of a fundamental geometric data structure. ACM Computing Surveys 23:345–405
3. Batta R, Ghose A, Palekar US (1989) Locating facilities on the Manhattan metric with arbitrarily shaped barriers and convex forbidden regions. Transport Sci 23:26–36
4. Butt SE, Cavalier TM (1996) An efficient algorithm for facility location in the presence of forbidden regions. Europ J Oper Res 90:56–70
5. Dearing PM, Hamacher HW, Klamroth K (1998) Center problems with barriers. Techn Report Depts Math Univ Kaiserslautern and Clemson Univ
6. Durier R, Michelot C (1985) Geometrical properties of the Fermat–Weber problem. Europ J Oper Res 20:332–343
7. Fliege J (1997) Nondifferentiability detection and dimensionality reduction in minisum multifacility location problems. J Optim Th Appl 94
8. Fliege J, Nickel S (2000) An interior point method for multifacility location problems with forbidden regions. Studies in Location Anal 14:23–45
9. Foulds LR, Hamacher HW (1993) Optimal bin location and sequencing in printed circuit board assembly. Europ J Oper Res 66:279–290
10. Francis RL, Hamacher HW, Lee C-Y, Yeralan S (1994) Finding placement sequences and bin locations for Cartesian robots. Trans Inst Industr Eng (IIE):47–59
11. Francis RL, McGinnis LF Jr, White JA (1992) Facility layout and location: An analytical approach, 2nd edn. Prentice-Hall, Englewood Cliffs
12. Hamacher HW (1995) Mathematische Lösungsverfahren für planare Standortprobleme. Vieweg, Braunschweig, Wiesbaden
13. Hamacher HW, Klamroth K (1997 2000) Planar location problems with barriers under polyhedral gauges. Report in Wirtschaftsmath 21, Dept Math, Univ Kaiserslautern

14. Hamacher HW, Nickel S (1994) Combinatorial algorithms for some 1-facility median problems in the plane. Europ J Oper Res 79:340–351
15. Hamacher HW, Nickel S (1995) Restricted planar location problems and applications. Naval Res Logist 42:967–992
16. Hamacher HW, Schöbel A (1997) A note on center problems with forbidden polyhedra. Oper Res Lett 20:165–169
17. Katz IN, Cooper L (1981) Facility location in the presence of forbidden regions, I: Formulation and the case of Euclidean distance with one forbidden circle. Europ J Oper Res 6:166–173
18. Klamroth K (1996) Planar location problems with line barriers. Report in Wirtschaftsmath 13, Dept Math, Univ Kaiserslautern
19. Larson RC, Sadiq G (1983) Facility locations with the Manhattan metric in the presence of barriers to travel. Oper Res 31:652–669
20. Lefebvre O, Michelot C, Plastria F (1991) Sufficient conditions for coincidence in minisum multifacility location problems with a general metric. Oper Res 39:437–442
21. Love RF, Morris JG, Wesolowsky GO (1988) Facilities location: Models and methods. North-Holland, Amsterdam
22. Masuyama S, Ibaraki T, Hasegawa T (1981) The computational complexity of the M-center problems on the plane. Trans IECE Japan E64:57–64
23. Megiddo N, Supowit KJ (1984) On the complexity of some common geometric location problems. SIAM J Comput 13:182–196
24. Michelot C (1987) Localization in multifacility location theory. Europ J Oper Res 31:177–184
25. Minkowski H (1967) Gesammelte Abhandlungen, vol 2. Chelsea, New York
26. Nickel S (1995) Discretization of planar location problems. Shaker, Aachen
27. Nickel S (1997) Bicriteria and restricted 2-facility Weber problems. Math Meth Oper Res 45(2):167–195
28. Nickel S (1998) Restricted center problems under polyhedral gauges. Europ J Oper Res 104(2):343–357
29. Nickel S, Dudenhöffer E-M (1997) Weber's problem with attraction and repulsion under polyhedral gauges. J Global Optim 11:409–432
30. Okabe A, Boots B, Sugihara K (1992) Spatial tesselations. Concepts and applications of Voronoi diagrams. Wiley, New York
31. Plastria F (1992) When facilities coincide: exact optimality conditions in multifacility Location. J Math Anal Appl 169:476–498
32. Plastria F (1995) Continuous location problems. In: Drezner Z (ed) Facility Location – A Survey of Applications and Methods. Springer, Berlin, pp 225–262
33. Schöbel A (1999) Locating lines and hyperplanes: Theory and algorithms. Kluwer, Dordrecht
34. Ward JE, Wendell RE (1985) Using block norms for location modeling. Oper Res 33:1074–1090

Multi-index Transportation Problems

MITP

MAURICE QUEYRANNE[1], FRITS SPIEKSMA[2]
[1] University British Columbia, Vancouver, Canada
[2] Maastricht University, Maastricht, The Netherlands

MSC2000: 90C35

Article Outline

Keywords
Formulations
Applications
  Transportation and Logistics
  Timetabling
  Multitarget Tracking
  Tables with Given Marginals
  Other Applications
Solution Methods
  A Greedy Algorithm for Axial MITPs
  A Monge Property
  Hub Heuristics for Axial MITPs
See also
References

Keywords

Transportation problem; Three-dimensional transportation problem; Greedy algorithm; Monge property; Approximation algorithms

An ordinary *transportation problem* has variables with two indices, typically corresponding to sources (or origins, or supply points) and destinations (or demand points). A *multi-index transportation problem* (MITP) has variables with three or more indices, corresponding to as many different types of points or resources or other factors. Multi-index transportation problems were considered by T. Motzkin [22] in 1952; an application involving the distribution of different types of soap was presented by E. Schell [35] in 1955. MITPs are also known as *multidimensional transportation problems* [4]. There are several versions and special cases of MITPs:

- The number k of dimensions may be fixed to a small value; the resulting MITP is called a *k-index transportation problem*, k ITP. Quite naturally, the

best studied cases are the *three-index transportation problems* (3ITPs), also known as *three-dimensional*, or *3D transportation problems*.

- The type of constraints is determined by an integer m with $0 < m < k$, defining m-fold k ITPs (called *symmetric MITPs* in [16]; see also [41, Chapt. 8]). The most common cases are *axial MITPs*, when $m = k-1$; and *planar MITPs*, when $m = 1$; see below for details.
- Integer solutions may or may not be required. Integrality requirements, which give rise to *integer MITPs*, may be necessary since MITPs lack the integrality property enjoyed by ordinary transportation problems (but see [22] for an exception).
- Unit right-hand sides, in conjunction with integrality requirements, give rise to *multi-index assignment problems* (MIAPs). (Some authors use this term for integer MITPs with integer right-hand sides; the present terminology, consistent with that for ordinary assignment and transportation problems, seems preferable.) MIAPs are hard to solve: the 3IAP is already *NP*-hard by reduction from the 3-dimensional matching problem [17]. Even worse [6]: no polynomial time algorithm for the 3IAP can achieve a constant performance ratio, unless $P = NP$.
- The objective function is usually a simple linear combination of the variables, normally a total cost to be minimized as in equation (1) below. Alternatives, not considered in this article, may include bottleneck objectives [11,36], more general nonlinear objectives such as in [34], or multicriteria problems [38].
- There may be additional constraints, such as upper bounds on the variables, (capacitated MITPs), variables fixed to the value zero (MITPs with forbidden cells), or constraints on certain partial sums of variables (MITPs with generalized capacity constraints).

MITPs with linear objectives and without integrality restrictions are linear programming problems with a special structure. The most extensively studied *integer MITPs* are *three-index assignment problems* (3IAPs); see also Three-index Assignment Problem.

Formulations

The following compact notation [31,34] avoids multiple summations and multiple layers of subindices. Let $k \geq 3$ denote the number of dimensions or indices, and $K = \{1, \ldots, k\}$. For $i \in K$ let A_i denote the set of values of the ith index. Let $A = \otimes_{i \in K} A_i = A_1 \times \cdots \times A_k$ denote the Cartesian product of these index sets, that is, the set of all joint indices (k-tuples) $a = (a(1), \ldots, a(k))$ with $a(i) \in A_i$ for all $i \in K$. One variable x_a is associated with each joint index $a \in A$. Thus, for example in a 3ITP with index sets I, J and L, the variable x_a stands for $x_{ij\ell}$ when the joint index is $a = (i, j, \ell)$.

Given unit costs $c_a \in \mathbf{R}$ for all $a \in A$, a linear objective function is

$$\min \sum_{a \in A} c_a x_a \tag{1}$$

and the variables are usually restricted to be nonnegative:

$$x_a \geq 0 \quad \text{for all } a \in A. \tag{2}$$

Given the integer m with $0 < m < k$, the demand constraints of the m-fold k ITP are defined as follows. Let $\binom{K}{k-m}$ denote the set of all $(k-m)$-element subsets of K; an $F \in \binom{K}{k-m}$ is interpreted as a set of $k-m$ 'fixed indices'. Given such an F and a $(k-m)$-tuple $g \in A_F = \otimes_{f \in F} A_f$ of 'fixed values', let

$$A(F, g) = \{a \in A:\ a(f) = g(f),\ \forall f \in F\}$$

be the set of k-tuples which coincide with g on the fixed indices. The m-fold demand constraints are

$$\sum_{a \in A(F,g)} x_a = d_{Fg} \quad \text{for all } F \in \binom{K}{k-m},\ g \in A_F, \tag{3}$$

where the right-hand sides d_{Fg} are given positive demands associated with the values g for fixed index subset F. These 'demands' may also denote supplies or capacities when the indices represent sources or some other resource type. When some of these resources are in excess, the equality in constraints (3) may be replaced with inequalities. Problem (1)–(3) is a k ITP. Adding the integrality restrictions

$$x_a \in \mathbb{N} \quad \text{for all } a \in A, \tag{4}$$

yields an integer MITP.

As mentioned above, the most common cases are $m = k-1$, defining axial MITPs; and $m = 1$, defining planar MITPs. For the axial problems, the notation may simplified by letting $d_{ig} = d_{Fg}$ when $F = \{ i \}$. Note that each variable x_a appears in the same number k of axial and planar demand constraints; however there are only $\sum_{i \in K} |A_i|$ axial constraints, versus $\sum_{i \in K} \prod_{f \in K \setminus \{ i \}} |A_f|$ planar constraints. Of course, it is possible to combine demand constraints with different values of m, so as to formulate different types of restrictions (e. g., see [5] and [16]).

Reductions between MITPs are presented in [16], where it is shown in particular that an m-fold k ITP can be reduced to a 1-fold k ITP for any m (with $0 < m < k$), thereby generalizing a result in [14]. Thus, an algorithm that solves planar k ITPs is in principle capable of solving m-fold k ITPs for any m (with $0 < m < k$).

Notice that any MITP with arbitrary right hand sides can be transformed to a MITP with right hand sides 1. This is a (pseudopolynomial) transformation and simply involves duplicating a resource with a supply of q units by q unit-supply resources. There seems to be little advantage in doing so, except perhaps in converting an integer MITP into one with 0–1 variables.

Another issue is the existence of feasible solutions. For an axial MITP the requirement of equal total demands $\sum_g d_{ig} = \sum_g d_{jg}$ for all $i, j \in K$ is a necessary and sufficient condition for the existence of feasible solutions. Feasibility conditions are more complicated for nonaxial problems; see [40] for a review of results for planar problems. See also [41, Chapt. 8] for properties of polytopes associated with (integer) MITPs, including issues of degeneracy.

Applications

Transportation and Logistics

MITPs are used to model transportation problems that may involve different goods; such resources as vehicles, crews, specialized equipment; and other factors such as alternative routes or transshipment points. Thus index sets A_1 and A_2 may represent destinations and sources, respectively, and the other sets A_3, A_4, …these additional factors. The type of 'demand' constraints used will reflect the availability of these factors and their interactions. Thus, for example, an axial demand constraint (3) with right-hand side d_{3i} will be used for a vehicle type $i \in A_3$ of which d_{3i} units are globally available (at identical cost) to all sources and destinations, while a constraint with $F = \{2, 3\}$ will be used if there are d_{Fg} vehicles of type $g(3)$ available at the different sources $g(2)$.

Interesting cases arise when each resource or factor $\ell \in A_i$ corresponds to a point $P_{i,\ell}$ in a *metric space*, i. e., a set with a distance δ, and the unit costs c_a are 'decomposable' as defined below. Each joint index $a \in A$ may be interpreted as a *cluster* of points among which transportation and other activities are conducted. The unit cost c_a reflects the within-cluster transportation costs associated with these activities; it is *decomposable* if it can be expressed as a function of the distances between pairs of points in the cluster a. Examples include the *diameter* $\max_{i,j} \delta(P_{i,a(i)}, P_{j,a(j)})$, when all these activities are performed simultaneously; the sum costs $\sum_{i,j} \delta(P_{i,a(i)}, P_{j,a(j)})$ when all activities are performed sequentially; and the *Hamiltonian path* or *path* costs, when all points $P_{i\ell}$ in the cluster have to be visited in a shortest sequence.

Other interesting cases arise when one of the indices denotes time. A simple *dynamic location problem* [27] may be modeled as an axial k ITP, where index set A_1 may denote the set of facilities (say, warehouses) to be located; A_2 that of candidate locations; and A_3 that of time periods. The costs c_{ijt} may include discounted construction and operating costs of these facilities. See [38] and [33] for other applications of this type.

Timetabling

Other problems involving time and which can be formulated as MITPs arise in timetabling or staffing applications. To illustrate, consider the following generic situation. Given are N employees (index i), each of which can be assigned to one of M tasks (index j) during each of T time periods (index k). Moreover, for each pair consisting of a task and a time period a number r_{jk} is given denoting the number of employees required for task j in period k. Also, a number r_{ij} is given denoting the number of periods that task j requires employee i. An employee can only be assigned to one task during each time period. Finally, there is a cost-coefficient c_{ijk} which gives the cost of employee i performing task j in period k. This problem is called the *multiperiod assignment problem* in [21] (see also the references contained

therein). To model this as a planar 3ITP, let A_1 be the set of employees; A_2 the set of tasks; A_3 the set of time periods;

$$d_{Fg} = \begin{cases} r_{jk} & \text{for } F = \{2,3\}, \quad \forall g = (j,k); \\ 1 & \text{for } F = \{1,3\}, \quad \forall g = (i,k); \\ r_{ij} & \text{for } F = \{1,2\}, \quad \forall g = (i,j); \end{cases}$$

and require the decision variables to be in $\{0, 1\}$. A special case arises when $r_{jk} = 1$ for all j, k and $N = M$. The polyhedral structure of the resulting planar 3ITP is investigated in [7]. Other references dealing with timetabling problems formulated as MITPs are [10,15] and [12].

Multitarget Tracking

Consider the following (idealized) situation. N objects move along straight lines in the plane. At each of T time instants a *scan* has been made, and the approximate position of each object is observed and recorded. From such a scan it is not possible to deduce which object generated which observation. Also, a small error may be associated with each observation. A *track* is defined as a T-tuple of observations, one from each scan. For each possible track a cost is computed based on a least squares criterion associated with the observations in the track. The problem is now to identify N tracks while minimizing the sum of the costs of these tracks. This problem is called the *data-association problem* in [25]. It can be modeled as an axial integer TIAP as follows: let A_i be the set of observations in scan i, $i = 1, \ldots, T$, and let $d_{ig} = 1$, $i = 1, \ldots, T$, $g = 1, \ldots, N$. Not surprisingly, this problem is NP-hard already for $T = 3$ (see [37]; notice however that this does not follow from the NP-hardness of 3IAP due to the structure present in the cost-coefficients in the objective function of multitarget tracking problems). Other references dealing with target tracking problems formulated as axial MIAPs are [23] and [24]; see also [20].

Tables with Given Marginals

Other statistical applications of MITPs require finding multidimensional tables with given sums across rows or higher-dimensional planes, as specified in constraints (3). The right-hand sides d_{Fg} of such constraints are often known as *marginals*. In a simple application [3] arising in the *integration of surveys* and *controlled selection*, each index set represents a population from which a sample is to be drawn. A (joint) sample is a k-tuple, one from each population. The marginals are specified marginal probability distributions over each population, giving rise to axial demand constraints. Given sample costs c_a, the problem is to find a joint probability distribution, defined by (x_a), of all the samples, consistent with these marginal distributions and of minimum expected cost (1).

In contrast, problems of *updating input-output matrices* (see [34] and references therein) typically have nonlinear objectives. In such problems, given are a k-dimensional array B of data (for example, past input-output coefficients) and arrays d of marginals (for example, forecast aggregate coefficients) with appropriate dimensions. The problem is to determine values x_a, the updated array entries, satisfying the demand constraints corresponding to the given marginals, and such that the resulting updated array $X = (x_a)$ differs as little as possible from the given array B, as specified by an appropriate (nonlinear) objective function. A (nonlinear) MITP arises when the values x_a are constrained to be nonnegative, a natural requirement in many contexts.

Other Applications

include an axial integer 3ITP model for planning the launching of weather satellites [27], and an axial integer 5IAP arising in routing meshes in circuit design [9].

Solution Methods

As noted above, MITPs are linear programming problems with a special structure. There are several proposals for extensions of LP (transportation) algorithms to MITPs (e. g., [4,13] for 3ITPs and [1] for a 4ITP).

As also mentioned earlier, integer MITPS are hard to solve. Exact algorithms have been proposed for the axial integer 3IAP (see Three-index Assignment Problem) and for the planar integer 3IAP (see [39] and [19]). Other exact approaches for integer MITPs rely on structure that is present in the particular application considered (see, e. g., [12]).

Several methods have been proposed to obtain good approximate solutions to integer MITPs. In [21]results

are reported for a *rounding heuristic* on some medium-sized planar integer 3ITPs. A *tabu search* algorithm for this problem is described in [18]. Heuristic solution approaches based on *Lagrangian relaxation* are proposed in [26,28] and [29] for multitarget tracking problems.

One major difficulty with these exact or approximate solution methods may be the sheer size of MITP formulations; if, for example, all $|A_i| = n$ then an m-fold k ITP has n^k variables and $\binom{k}{m} n^{k-m}$ constraints. In contrast, the two approaches sketched below yield feasible solutions to *axial* MITPs much more quickly than simply writing down all the cost coefficients. In particular, these algorithms only produce the nonzero variables x_a and their values; all other variables are zero in the solution. In addition, this solution is integral if all demands are integral. Of course, the effectiveness of these methods relies on some assumptions on the cost coefficients c_a, assumptions which are verified in several applications.

A Greedy Algorithm for Axial MITPs

The *greedy algorithm* below (a multi-index extension of the *North–West corner rule*) finds a feasible solution to axial MITPs in $O(k \sum_i |A_i|)$ time, which is (for fixed k) linear in the size of the demand data d_{ig}. This solution is in fact optimal if the cost coefficients are known to satisfy a 'Monge property' [3,31,32] defined below. (For $k = 3$, this greedy algorithm is already described in [4] to obtain a basic feasible solution).

Consider the axial k ITP with equality constraints (3) and assume that each $A_i = \{1, \ldots, |A_i|\}$. Recalling that the demands are denoted d_{ig}, assume that $\sum_{g \in A_i} d_{ig} = \sum_{g \in A_1} d_{1g}$ for all $i \in K$, a necessary and sufficient condition for the problem to be feasible.

```
PROCEDURE greedy MITP algorithm
    WHILE (∑_{g∈A_i} d_ig > 0 for all i ∈ K) DO
        let a(i) = min{g ∈ A_i : d_ig > 0};
        let Δ = min{d_{i,a(i)} : i ∈ K};
        let x_a = Δ;
        FOR i ∈ K DO let d_{i,a(i)} = d_{i,a(i)} − Δ;
RETURN x
END
```

A greedy algorithm for axial MITPs

A Monge Property

The *join* $a \vee b$ and *meet* $a \wedge b$ of $a, b \in A$ are

$$(a \vee b)_i = \max\{a(i), b(i)\},$$
$$(a \wedge b)_i = \min\{a(i), b(i)\} \quad \text{for all } i \in K.$$

The cost coefficients (c_a) satisfy the *Monge property* if

$$c_{a \vee b} + c_{a \wedge b} \leq c_a + c_b \quad \text{for all } a, b \in A.$$

Note that this is just the *submodularity* of the function $c: A \to \mathbf{R}$ defined on the product lattice A, see [3,31,32]. These references show that the above greedy algorithm returns an optimal solution for *all* feasible demands if and only if the cost function satisfies the Monge property. The latter two references also extend the greedy algorithm

i) to the case of forbidden cells when the nonforbidden cells form a *sublattice* of A; and
ii) so that it returns an optimal dual solution.

They also show that optimizing a linear function over a *submodular polyhedron* is special case of the dual problem. It is shown in [32] that the primal problems are equivalent to the 'submodular linear programs on forests' of [8].

Cost functions c with the Monge property include typical decomposable costs (as defined above) when all the points are located on a same line or on parallel lines (one line for each factor type A_i). For these problems, the greedy algorithm above amounts to a 'left to right sweep' across the points.

Hub Heuristics for Axial MITPs

The basic idea ([30], extending earlier work on axial 3IAPS [6] and MIAPs [2] with decomposable costs) is to solve a small number of ordinary transportation problems and to expand their solutions into a feasible solution to the original MITP. For a large collection of decomposable costs arising from applications, the objective value of this feasible solution is provably within a constant factor of the optimum.

Given an index h, called the *hub*, determine, for each index $i \neq h$, a feasible solution to the ordinary transportation problem defined by supplies $(d_{ij})_{j \in A(i)}$ and $(d_{hg})_{g \in A(h)}$. The Expand procedure below then takes as inputs these solutions $y^{(h)} = (y^i)_{i \neq h}$ and expands them into a feasible solution $x^{(h)}$ to the axial MITP. Its running time is $O(|A_h| \sum_{i \neq h} |A_i|)$.

FOR $k = 1, \ldots, n$,
PROCEDURE Expand(h, $y^{(h)}$)
FOR $g := 1$ TO n_h DO
  $q := 0$;
  $a(i) := 1$ for $i \in K \setminus h$;
  WHILE($q < d_{h,g}$) DO
    let ℓ be such that
    $y^{\ell}_{a(\ell),g} = \min\{y^{r}_{a(r),g} : r \neq h\}$;
    $x^{(h)}_{a} := y^{\ell}_{a(\ell),g}$;
    $y^{r}_{a(r),g} := y^{r}_{a(r),g} - x^{(h)}_{a}$ for all $r \in K \setminus h$;
    $a(\ell) := a(\ell) + 1$;
    $q := q + x^{(h)}_{a}$;
RETURN $x^{(h)}$
END

The Expand procedure for axial MITPs

In the hub heuristics for decomposable costs, the ordinary transportation problems use as cost coefficients the distances $\delta(P_{ij}, P_{hg})$ between the corresponding points P_{ij} and P_{hg} in the metric space. The expanded MITP solution x^h would be optimum if the cost function was that of the star with center h, namely if $c_a = \sum_{i \neq h} \delta(P_{i,a(i)}, P_{h,a(h)})$. The *triangle-inequality* property of the distance δ allows one to bound the cost penalty from using this h-star cost function instead of the actual decomposable cost function.

In the *single hub heuristic*, one chooses a hub $h \in K$; solves these $k-1$ transportation problems; inputs their solutions $y^{(h)}$ to Expand; and simply outputs the resulting MITP solution $x^{(h)}$. If the distance δ satisfies the triangle inequality, the cost of this solution $x^{(h)}$ is no more than $k-1$ times the optimal cost, in the worst case, for many common decomposable cost functions. The *multiple-hub heuristic* is an obvious extension whereby one performs the single-hub heuristic k times, once for each $h \in K$, and retains the best solution. This amounts to solving $\binom{K}{2}$ ordinary transportation problems. Under the same assumptions as above and for many common decomposable cost functions, the cost of the resulting solution is less than twice the optimum cost in the worst case.

See also

- ► Generalized Assignment Problem
- ► Stochastic Transportation and Location Problems

References

1. Bammi D (1978) A generalized-indices transportation problem. Naval Res Logist Quart 25:697–710
2. Bandelt H-J, Crama Y, Spieksma FCR (1994) Approximation algorithms for multidimensional assignment problems with decomposable costs. Discrete Appl Math 49:25–50
3. Bein WW, Brucker P, Park JK, Pathak PK (1995) A Monge property for the d-dimensional transportation problem. Discrete Appl Math 58:97–109
4. Corban A (1964) A multidimensional transportation problem. Rev Roumaine Math Pures et Appl IX:721–735
5. Corban A (1966) On a three-dimensional transportation problem. Rev Roumaine Math Pures et Appl XI:57–75
6. Crama Y, Spieksma FCR (1992) Approximation algorithms for three-dimensional assignment problems with triangle inequalities. Europ J Oper Res 60:273–279
7. Euler R, Le Verge H (1996) Time-tables, polyhedra and the greedy algorithm. Discrete Appl Math 65:207–222
8. Faigle U, Kern W (1996) Submodular linear programs on forests. Math Program 72:195–206
9. Fortin D, Tusera A (1994) Routing in meshes using linear assignment. In: Bachem A, Derigs U, Jünger M, Schrader R (eds) Operation Research 93, pp 169–171
10. Frieze AM, Yadegar J (1981) An algorithm for solving 3-dimensional assignment problems with application to scheduling a teaching practice. J Oper Res Soc 32:989–995
11. Geetha S, Vartak MN (1994) The three-dimensional bottleneck assignment problem with capacity constraints. Europ J Oper Res 73:562–568
12. Gilbert KC, Hofstra RB (1987) An algorithm for a class of three-dimensional assignment problems arising in scheduling applications. IIE Trans 8:29–33
13. Haley KB (1962) The solid transportation problem. Oper Res 10:448–463
14. Haley KB (1963) The multi-index problem. Oper Res 11:368–379
15. Junginger W (1972) Zurückführung des Stundenplanproblems auf einen dreidimensionales Transportproblem. Z Oper Res 16:11–25
16. Junginger W (1993) On representatives of multi-index transportation problems. Europ J Oper Res 66:353–371
17. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations. Plenum, New York, pp 85–103
18. Magos D (1996) Tabu search for the planar three-index assignment problem. J Global Optim 8:35–48
19. Magos D, Miliotis P (1994) An algorithm for the planar three-index assignment problem. Europ J Oper Res 77:141–153
20. Mavridou T, Pardalos PM, Pitsoulis L, Resende MGC (1998) A GRASP for the biquadratic assignment problem. Europ J Oper Res 105/3:613–621

21. Miller JL, Frank LS (1996) A binary-rounding heuristic for multi-period variable-task duration assignment problems. Comput Oper Res 23:819–828
22. Motzkin T (1952) The multi-index transportation problem. Bull Amer Math Soc 58:494
23. Murphey R, Pardalos PM, Pitsoulis L (1998) A GRASP for the multitarget multisensor tracking problem. In: DIMACS, vol 40. Amer Math Soc, Providence, pp 277–302
24. Murphey R, Pardalos PM, Pitsoulis L (1998) A parallel GRASP for the data association multidimensional assignment problem. In: IMA Vol Math Appl, vol 106. Springer, Berlin, pp 159–180
25. Pattipatti KR, Deb S, Bar-Shalom Y, Washburn RB Jr (1990) Passive multisensor data association using a new relaxation algorithm. In: Bar-Shalom Y (ed) Multitarget-multisensor tracking: Advances and applications, p 111
26. Pattipatti KR, Deb S, Bar—Shalom Y, Washburn RB Jr (1992) A new relaxation algorithm passive sensor data association. IEEE Trans Autom Control 37:198–213
27. Pierskalla WP (1968) The multidimensional assignment problem. Oper Res 16:422–431
28. Poore AB (1994) Multidimensional assignment formulation of data-association problems arising from multitarget and multisensor tracking. Comput Optim Appl 3:27–57
29. Poore AB, Rijavec N (1993) A Lagrangian relaxation algorithm for multidimensional assignment problems arising from multitarget tracking. SIAM J Optim 3:544–563
30. Queyranne M, Spieksma FCR (1997) Approximation algorithms for multi-index transportation problems with decomposable costs. Discrete Appl Math 76:239–253
31. Queyranne M, Spieksma FCR, Tardella F (1993) A general class of greedily solvable linear programs. In: Rinaldi G, Wolsey L (eds) Proc Third IPCO Conf (Integer Programming and Combinatorial Optimization), pp 385–399
32. Queyranne M, Spieksma FCR, Tardella F (1998) A general class of greedily solvable linear programs. Math Oper Res 23(4):892–908
33. Rautman CA, Reid RA, Ryder EE (1993) Scheduling the disposal of nuclear waste material in a geologic repository using the transportation model. Oper Res 41:459–469
34. Romero D (1990) Easy transportation-like problems on K-dimensional arrays. J Optim Th Appl 66:137–147
35. Schell E (1955) Distribution of a product by several properties. In: Directorate of Management Analysis (ed) Second Symposium in Linear Programming 2. DCS/Comptroller HQ, US Air Force, Washington DC, pp 615–642
36. Sharma JK, Sharup K (1977) Time-minimizing multidimensional transportation problem. J Eng Production 1:121–129
37. Spieksma FCR, Woeginger GJ (1996) Geometric three-dimensional assignment problems. Europ J Oper Res 91:611–618
38. Tzeng G, Teodorović D, Hwang M (1996) Fuzzy bicriteria multi-index transportation problems for coal allocation planning of Taipower. Europ J Oper Res 95:62–72
39. Vlach M (1967) Branch and bound method for the three-index assignment problem. Ekonomicko–Matematicky Obzor 3:181–191
40. Vlach M (1986) Conditions for the existence of solutions of the three-dimensional planar transportation problem. Discrete Appl Math 13:61–78
41. Yemelichev VA, Kovalev MM, Kratsov MK (1984) Polytopes, graphs and optimization. Cambridge Univ Press, Cambridge

Multilevel Methods for Optimal Design

Natalia M. Alexandrov
NASA Langley Res. Center, Hampton, USA

MSC2000: 49M37, 65K05, 65K10, 90C30, 93A13

Article Outline

Keywords
Problem Formulation
Bilevel Optimization
  Penalty-Based Methods
  KKT-Based Methods
  Descent-Based Methods
Examples: Collaborative Optimization
Example: MAESTRO,
  a Class of Multilevel Algorithms
Summary
See also
References

Keywords

Nonlinear optimization; Multilevel; Bilevel; Hierarchical; Multidisciplinary design

Multilevel, or *hierarchical*, programming problems (MLP) are constrained optimization programs in which subsets of the solution set are themselves solution sets of other, lower-level optimization programs. Several general MLP problem statements exist. They differ from one another in the specifics of optimization variable distribution among the levels and the definition of the objectives and constraints at particular levels.

Given a set of objectives $\{f_i\}_{i=1,\ldots,M}$ with $f_i: \mathbf{R}^n \to \mathbf{R}$ and a vector of variables $x \in \mathbf{R}^n$, partitioned into subsets

$x = (x_1, \ldots, x_M)$ for some integer M denoting the number of subsystems, a prototypical form of MLP may be stated as follows:

$$\begin{cases} \min_{x_1 \in S_1} & f_1(x) \\ \text{s.t.} & x_2 \in \operatorname{argmin}_{x_2 \in S_2}\{f_2(x)\} \\ & \vdots \\ & x_M \in \operatorname{argmin}_{x_M \in S_M}\{f_M(x)\}, \end{cases}$$

where the optimization problem at each level i controls its own subset of variables x_i, while the other subsets of variables $x_1, \ldots, x_{i-1}, x_{i+1}, x_M$ serve as parameters. The constraint set for each level is $S_i \equiv \{x\colon h_i(x) = 0, g_i(x) \geq 0\}$ with $h_i\colon \mathbf{R}^n \to \mathbf{R}^{m_{h_i}}$ and $g_i\colon \mathbf{R}^n \to \mathbf{R}^{m_{g_i}}$ for some integers m^{h_i}, m^{g_i}.

This form of MLP inspired by the work of H. Stackelberg [92] can be viewed as an M-player Stackelberg game [18,84]. Its interpretation is that of M autonomous players or decision makers seeking to minimize their (possibly constrained) objective functions while manipulating subsets of decision or design variables disjoint from those of other decision makers. The higher-level problems are implicit in the variables of the lower-level problems. This formulation has been studied widely in the bilevel case. See, for example, [15] and the references therein. In general, all problem levels, but the outermost one, may contain a number of concurrent optimization problems.

A related variant of the problem, known as the *generalized bilevel programming problem*, represents the reaction of the lower-level problem to decisions made by the upper-level problem via a solution of an equilibrium problem stated as a variational inequality:

$$\begin{cases} \min\limits_{\substack{x \in X, \\ y \in Y(x)}} & f_1(x, y) \\ \text{s.t.} & \langle f_2(x, y), y - z\rangle \leq 0 \quad \text{for all } z \in Y(x), \end{cases}$$

where the upper-level domain X is such that the lower-level domain $Y(x)$ is not empty. This formulation was introduced by P. Marcotte in [63] and studied in [45,64], and [71].

Multilevel problems may be partitioned into two classes with respect to another criterion [100]. In one of the classes, upper-level optimization problems depend on the corresponding lower-level ones through the *optimal value functions* (or the *marginal functions*) of the lower-level problems. An optimal value function represents the value of a lower-level objective function at a solution of that lower-level problem. In the other class, upper-level problems depend on the corresponding lower-level problems through the actual optimal solutions of the latter. An example of two such formulations in engineering design optimization will be given further.

Multilevel programming problems arise in numerous applications where the structure of the application involves hierarchical *decision making* or where the sheer size and complexity of the problem necessitates partitioning of the system and processing the subsystems in a hierarchical fashion. Information on applications of multilevel optimization in such varied areas as power systems, water resource systems, urban traffic systems, and river pollution control can be found in [36,50,51,52,62,69,70,85], and many other references. The use of multilevel algorithms in engineering control is well documented, for instance, in [46] and [57].

The broad area of *multidisciplinary design optimization*(MDO) – a term that denotes a large set of research subjects and practical techniques for the design of complex coupled engineering systems – is particularly amenable to the use of multilevel methods, due to the extreme computational expense and the organizational complexity of the field. For instance, the design of aircraft involves aerodynamics, structural analysis, control, weights, propulsion, and cost, to list a few disciplines. The complexity and expense of each discipline have assured that most disciplines have developed into vast, autonomous fields of study, so that practically feasible optimization methods that involve the contributing disciplines must take into account such an *autonomy* and the hierarchical organization. Maintaining disciplinary autonomy while accounting for interdisciplinary subsystem couplings and allowing for integrated system optimization with respect to system and interdisciplinary objectives is one of the tasks of MDO. Overviews of multidisciplinary optimization may be found in [6] and [90].

Practitioners of engineering have been using multilevel methods, in some form, since optimization algorithms made their appearance in engineering problems. The seminal works [60,65], and [98] contributed

to a systematic development and understanding of hierarchical optimization. Multilevel methods have been studied extensively in application to multidisciplinary design ([16,17,22,96,97]) and single-discipline design areas that give rise to large problems, such as structural optimization (e. g., [74,87,91]). Engineering multilevel optimization has always had a strong connection with *multi-objective optimization* (e. g., [53]).

Problem Formulation

The procedure of formulating an engineering design problem as a multilevel or a bilevel problem is difficult and depends on the complexity and size of the problem. The general components in formulating a multilevel optimization problem are as follows:

- The original problem is studied to determine its structure. Structure is of paramount importance in deciding to adopt a particular formulation. For instance, most formulations assume that the problem subsystems share only a relatively small number of variables, i. e., that the *bandwidth of interdisciplinary coupling* is relatively small.
- The problem is partitioned into a system (or upper-level) problem and subsystem (or lower-level) problems. Decisions are made on inclusions of particular variables and constraints into the system and subsystems. Decisions are also made on the form of the system and subsystem objectives.
- Finally, algorithms are selected for solving the system and subsystem optimization problems. One must distinguish a formulation of the problem from the algorithm used to solve that formulation. While some of the multilevel formulations can be easily shown to be mathematically equivalent to the original problem with respect to solution sets, they may not be equivalent with respect to other attributes, such as constraint qualifications and optimality conditions. Hence the numerical properties of algorithms applied to different formulations vary widely [8,9,10].

Problem *decomposition* constitutes a special area of study. In general, decomposition techniques take advantage of the problem structure and depend on the strength and bandwidth of couplings among the subsystems. *Separable* and partially separable problems are particularly amenable to decomposition.

Two types of decomposition may be considered in design optimization. Coarse-grained decomposition with respect to disciplines presents no difficulty, because the design problem initially consists of autonomous parts. The difficulty at this level of problem formulation is in *integration* or *synthesis*. However, in realistic applications, even though the coarse-grained decomposition is frequently obvious, the complexity of the problem requires that a *dependence analysis* be performed in order to determine the most advantageous arrangement or sequencing of the disciplinary subsystems in the optimization procedure. Automatic techniques based on graph-theoretic foundations may be found in [78] and [79], for instance.

Finer-grained decomposition within a particular discipline may be addressed by a multitude of techniques for decomposition of mathematical programs. Extensive references on decomposition in general mathematical programming, beginning with [19] and [31], and extended in [49] and many others, can be found in [42] and [43]. Further references to decomposition techniques aimed specifically at design problems can be found in [95].

General multilevel programming presents an exceedingly difficult problem, and many multilevel formulations and algorithms of engineering design rely more on heuristics than on theoretically substantiated foundations. There are exceptions, for instance, such as those in [12,29,68], and [75]. While many engineering multilevel approaches have enjoyed success when applied to specific problems, insufficient analytical foundation and the difficulty of the problem usually mean that the approaches are not robust, and extensive 'fine-tuning' of heuristic parameters is required for each new problem or instance of a problem. Hence, recent years have seen renewed interest in systematic, analytically substantiated approaches to MLP. Many such developments have taken place in *bilevel optimization*.

Bilevel Optimization

Although bilevel optimization problems (BLP) form the simplest case of multilevel optimization, they are very difficult to solve and constitute a fertile research area. A survey of the field can be found in [28]. A large bibliography with an emphasis on theoretical developments is also provided in [94].

The conventional general bilevel problem may be posed as follows:

$$\begin{cases} \min_{x \in X} & f_1(x, y) \\ \text{s.t.} & h_1(x, y) = 0 \\ & g_1(x, y) \geq 0, \end{cases}$$

where y solves for fixed x:

$$\begin{cases} \min_{y \in Y} & f_2(x, y) \\ \text{s.t.} & h_2(x, y) = 0 \\ & g_2(x, y) \geq 0. \end{cases}$$

The cases of linear and convex problem functions have been studied widely. A popular class of methods for the linear bilevel problem (extreme point algorithms) computes global solutions by enumerating extreme points of the lower-level feasible set (e. g., [27]). Convex bilevel problems are often solved by branch and bound methods (e. g., [15]). A survey of methods for linear and convex bilevel programming can be found in [11].

The considerably more difficult case of nonlinear and nonconvex problem functions has inspired much research activity as well but has, to date, led to few computationally successful algorithms. The existing approaches to nonlinear bilevel optimization can be classified into several categories.

Penalty-Based Methods

This category uses penalty methods. In some algorithms (e. g., [1]), a barrier function penalizes the lower-level objective. In double-penalty methods, both the lower-level problem and the upper-level problem are approximated by sequences of unconstrained optimization problems [56,61,64]. Single or double-penalty methods are, in general, expected to converge slowly, especially for highly nonlinear problems. Thus using these methods for the usually large and nonlinear design optimization problems may be difficult.

KKT-Based Methods

The algorithms of this category convert the bilevel problem into a nonconvex, single-level optimization problem by using the *Karush—Kuhn—Tucker conditions* (KKT conditions) of the lower-level problem as constraints on the upper-level problem [14,15,20,37,44]. If the lower-level problem is convex, the KKT formulation is equivalent to the original formulation [14]. However, even in this case, the KKT conditions on the lower-level problem include the *complementarity slackness* condition as a constraint. The form of the complementarity condition makes the single-level problem difficult to solve. The KKT formulation suffers from an additional difficulty. Namely, it is well known from the study of the sensitivity and stability of nonlinear programming (e. g., [40]) that even if the lower-level problem behaves exceedingly well in that it satisfies such stringent assumptions as *strong second order sufficiency* and *regularity* as a *constraint qualification*, the *feasible set* of the single-level problem will generally not be differentiable with respect to x. Hence, the performance of gradient-based solvers on the transformed problem may be adversely affected.

Descent-Based Methods

Another category of algorithms is based on solving subproblems that result in descent for the upper-level problem with gradient information of the lower-level problem used in a number of ways [34,39,59,83].

The remainder of the article will be devoted to a more detailed description of two specific approaches to nonlinear, nonconvex problems that arose from the need to solve engineering design problems. One approach is a bilevel formulation, the other is an algorithm for solving multilevel formulations.

Examples: Collaborative Optimization

Collaborative optimization (CO) is a general approach to solving multidisciplinary design optimization problems by formulating them as nonlinear bilevel programs of special structure. CO comprises a number of methods. Its antecedents can be traced to earlier hierarchical approaches, as in [60] and [98]. The underlying idea of CO appeared in [13,80,81,82,88] and [96,97]. The approach has recently received attention under the name of *collaborative optimization* [22,23,86,93].

Given that MDO problems are naturally partitioned into subsystems along disciplinary lines, CO suggests an intuitively attractive way to formulate the optimization problem so that the autonomy of the disciplinary subsystem computations is preserved. However, the approach presents a problem that is difficult to solve by

means of conventional nonlinear programming software [7,58]. The analytical and computational aspects of CO were addressed in [9], of which the following discussion is an abstract. As a complete description of CO is lengthy, only an abbreviated version is considered here.

It is assumed that the original system is composed of a number, say M, of interdependent but autonomous systems, each of which is described by a disciplinary analysis A_i, $i = 1, \ldots, M$, expressed in the form

$$A_i(x_i, y_i(x_i)) = 0,$$

where, given a vector of disciplinary design variables x_i, the analysis (frequently represented by a numerical differential equation solver or simulator) is performed to yield the vector of state variables or responses $y_i(x_i)$. The sets of disciplinary variables x_i are not necessarily disjoint. The disciplinary constraints are usually represented by inequalities

$$c_i(x_i, y_i(x_i)) \geq 0.$$

Once the system objective and variables and the subsystem constraints and variables are identified, the bilevel problem is formed as follows:

The constraints of the system problem comprise the 'consistency' (or 'coupling' or 'matching') conditions that are used to drive the discrepancy among the inputs and outputs shared by the subsystems to zero. The values of the constraints are computed by solving the subsystem optimization problems, and the number of *consistency constraints* is related to the number of subsystems and variables shared among the subsystems. The form of the consistency constraints determines a particular implementation of CO.

Let ξ and η represent system-level variables corresponding to inputs and outputs of subsystems, respectively. Then, given M subsystems, the abbreviated system program is

$$\begin{cases} \min & F(\xi, \eta) \\ \text{s.t.} & G(\xi, \eta) = 0, \end{cases} \tag{1}$$

where

$$G(\xi, \eta) = \begin{pmatrix} g_1(\xi, \eta) \\ \vdots \\ g_M(\xi, \eta) \end{pmatrix}$$

is the set of system consistency constraints obtained by solving lower-level subproblems, each of which is of the form

$$\begin{cases} \min & \frac{1}{2}\left[\|\xi_i - x_i\|^2 + \|\eta_i - y(x_i)\|^2\right] \\ \text{s.t.} & c_i(x_i, y(x_i)) \geq 0, \end{cases} \tag{2}$$

where i is the number of the subsystem. Thus, the objective of a subsystem optimization problem is always to minimize the discrepancy between the shared variables of the subsystems, in a least squares sense, subject to satisfying the disciplinary constraints, which do not depend explicitly on the system variables passed down to the subsystems as parameters. The subsystems remain feasible during optimization, while *interdisciplinary feasibility* is gradually attained at the system level via the consistency constraints. Maintaining disciplinary feasibility is extremely important from the design perspective.

The problem now consists of a set of decoupled subproblems that can be solved independently and in parallel.

One instance of the system-level consistency conditions gives rise to the form in which CO is usually presented: namely, the consistency condition is intended to drive to zero the value function of the subproblem (2). That is,

$$g_i(\xi, \eta) = \frac{1}{2}\|\xi - x_*\|^2 + \|\eta - y(x_*)\|^2, \tag{3}$$

where x_* solves the subsystem optimization problem.

Another instance of system-level consistency conditions matches the system-level variables with their subsystem counterparts computed in subproblem

$$g_i(\xi, \eta) = (\xi - x_*, \eta - y(x_*)). \tag{4}$$

The behavior of optimization algorithms applied to the original and CO formulations will differ greatly, as the formulations are not equivalent with respect to constraint qualifications or optimality conditions.

In general, value functions are not differentiable, and this may cause difficulties for optimization algorithms applied to the system-level problem. However, under a number of strong assumptions, the constraints are locally differentiable and can usually be computed.

Derivatives of the system-level constraints with respect to the system-level design variables are the sensi-

tivities of the minima or the solutions of the subsystem-level optimization problems to parameters. The area of *sensitivity in nonlinear programming* has been studied extensively. Relevant results can be found in [40] and [41]. In particular, under the assumptions of sufficient smoothness, *second order sufficiency*, regularity as constraint qualification, and *strict complementarity slackness*, the basic sensitivity theorem (BST) proves the existence of a unique, local, continuously differentiable solution-multiplier triple for the perturbed problem. Moreover, locally, the set of active constraints remains unchanged and regularity and strict complementarity hold, allowing one to compute derivatives locally. In fact, under a number of assumptions, stronger statements can be made about the differentiability of the value function [30,77].

Under the conditions of BST, local first order derivatives of the consistency constraints (3) have a particularly simple form because, in the case of CO, the constraints of the lower-level problems do not depend on parameters. On the other hand, the first order sensitivities of solutions of the lower-level problem that form the derivatives of the consistency constraints (4), while of closed form, are expensive to compute and involve second order derivatives of the subsystem Lagrangians.

There is another feature of the CO formulation with compatibility constraints (3) that will cause difficulties for nonlinear programming algorithms applied to the system-level problem: *Lagrange multipliers* will almost never exist for the equality constrained system level problem, with all the ensuing consequences. The nonexistence of Lagrange multipliers is due to the description of the feasible region that causes the Jacobian of the system-level constraints to vanish at a solution. The formulation with compatibility constraints (4) aims to address this problem. However, the computation of derivatives for this formulation is clearly expensive, as it not only involves solving a system of equations, but also requires the computation of second order information for the subsystems. The difficulties are addressed in detail in [9].

In summary, CO is an appealing approach to design optimization; however, the bilevel nature of the problem formulation will cause difficulties for conventional nonlinear programming algorithms applied to the system-level problem. Variations, special algorithms for solution, and alternatives can be found in, e. g. [33,54,55].

Example: MAESTRO, a Class of Multilevel Algorithms

As mentioned earlier, most multilevel formulations and algorithms for engineering design problems assume that the bandwidth of coupling among the subsystems comprised by the multilevel system is small. While many problems may be stated in this way, it is becoming increasingly important to consider problems with large bandwidth of coupling where, to use an MDO expression, 'everything affects everything else'. MAESTRO (a class of multilevel algorithms for constrained optimization; [2]) is intended for solving large nonlinear programming problems with arbitrary couplings among the naturally occurring subsystems, i. e., a particular instance of MDO problems with a single objective. The class was extended in, e. g., [5] to include a large class of steps for the nonlinear programming problem and in [3,4] to incorporate general nonlinear objectives. The class makes no assumptions on the structure of the problem, such as convexity or separability.

The algorithms of the class are based on *trust region methodology* (see, e. g., [35,38,67]) and are proven to converge under reasonable assumptions.

The idea of the MAESTRO algorithms is to attain sequential predicted *sufficient decrease conditions* for all the constrained objectives, and is a direct extension of the multilevel ideas for the equality constrained optimization problem. The approach can be summarized as follows. Given an initial approximation to the solution of the multilevel problem, the *trial step* for the multilevel problem is computed as a sum of a sequence of substeps, each of which predicts sufficient (or optimal) decrease in the quadratic model of the objective of a given subproblem, subject to maintaining predicted decrease in the models of the previous objectives. For instance, in the case of the unconstrained bilevel problem, the trial step for the bilevel problem is a sum of two substeps. The first substep is computed to predict sufficient decrease, via the quadratic model of the innermost objective f_2, for the subproblem of approximately optimizing

$$m_{f_2}(s) \equiv f_2(x_c) + \nabla f_2(x_c)^\top s + \frac{1}{2} s^\top H_2(x_c) s,$$

in the trust region of size δ_{f_2} to produce the substep s_{f_2}, where x_c is the current approximation to the solution and H_2 is the current approximation to the Hessian of f_2. The second step s_{f_1} would then approximately minimize the quadratic model of the outermost objective f_1, constructed at $x_c + s_{f_2}$, in the trust region of size δ_{f_1}, subject to constraints that enforce the preservation of the predicted sufficient or optimal decrease for f_1. The total trial step is evaluated by using the merit function designed to account for the sequential processing of the objectives. The algorithm is shown to converge to critical points of the bilevel or multilevel problem. Thus, the essential difference between this approach and the classical approaches to bilevel optimization is that instead of starting from the optimality conditions for the bilevel or multilevel problem, the approach attempts to obtain decrease on the sequence of subproblem models, while preserving predicted decrease for the previously processed subproblems, and to measure progress via the use of an appropriate merit function with rigorously updated penalty parameters. It is important to emphasize that the merit function is used only to evaluate the steps, and not to compute them.

The ongoing work is concerned with practical implementation issues and applications to engineering design problems.

Summary

Multilevel optimization has been an active research field, both in applied mathematics and in engineering design. Many open questions remain, in particular, in the area of practical computational algorithms for bilevel and multilevel problems. Overviews of some recent developments can be found in [66].

Understanding the behavior of specific, nonlinear programming algorithms applied to the system-level problem of the bilevel or multilevel formulations will present an interesting and difficult area of inquiry, and would benefit from the techniques of *nonsmooth analysis and optimization* [32,36,47], unconventional notions of constraint qualifications [24,25], and optimality [99,100].

To facilitate research and testing in the area of algorithms, one may find automatic bilevel and multilevel problem generators, as well as other sources of multilevel problems, described in [26,72,73].

See also

- ▶ Bilevel Fractional Programming
- ▶ Bilevel Linear Programming
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Bilevel Programming
- ▶ Bilevel Programming: Applications
- ▶ Bilevel Programming: Applications in Engineering
- ▶ Bilevel Programming: Implicit Function Approach
- ▶ Bilevel Programming: Introduction, History and Overview
- ▶ Bilevel Programming in Management
- ▶ Bilevel Programming: Optimality Conditions and Duality
- ▶ Design Optimization in Computational Fluid Dynamics
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Multidisciplinary Design Optimization
- ▶ Multilevel Optimization in Mechanics
- ▶ Optimal Design of Composite Structures
- ▶ Optimal Design in Nonlinear Optics
- ▶ Stochastic Bilevel Programs
- ▶ Structural Optimization: History

References

1. Aiyoshi E, Shimizu K (1981) Hierarchical decentralized system and its new solution by a barrier method. IEEE Trans Syst, Man Cybern 11:444–449
2. Alexandrov NM (1993) Multilevel algorithms for nonlinear equations and equality constrained optimization. PhD Thesis Rice Univ.
3. Alexandrov NM (1996) Multilevel and multiobjective optimization in multidisciplinary design. AIAA Paper no. 96-4122
4. Alexandrov NM (2000) A trust-region algorithm for nonlinear bilevel optimization. in preparation
5. Alexandrov NM, Dennis JE (1995) Multilevel algorithms for nonlinear optimization. In: Borggaard J, Burkardt J, Gunzburger M, Peterson J (eds) Optimal Design and Control. Birkhäuser, Basel, pp 1–22
6. Alexandrov NM, Hussaini MY (eds) (1997) Multidisciplinary design optimization: State of the art. SIAM, Philadelphia
7. Alexandrov NM, Kodiyalam S (1998) Initial results of an MDO method evaluation study. AIAA Paper no. 98-4884

8. Alexandrov NM, Lewis RM (2000) Algorithmic perspectives on problem formulations in MDO. AIAA Paper no. 2000-4719
9. Alexandrov NM, Lewis RM (2002) Analytical and computational aspects of collaborative optimization for multidisciplinary design. AIAA J 40:301–309
10. Alexandrov NM, Lewis RM (2000) Analytical and computational properties of distributed approaches to MDO. AIAA Paper no. 2000-4718
11. Anadalingam G, Friesz TL (1992) Hierarchical optimization: An introduction. Ann Oper Res 34:1–11
12. Badhrinath K, Rao JRJ (1994) Bilevel models for optimum designs which are insensitive to perturbations in variables and parameters. Techn Report Univ Houston UH-ME-SDL-94-03
13. Balling RJ, Sobieszczanski-Sobieski J (1994) An algorithm for solving the system-level problem in multilevel optimization. In: Fifth AIAA/USAF/NASA/ISSMO Symp. Multidisciplinary Analysis and Optimization (Panama Beach, Florida, Sept. 7-9, 1994), AIAA Paper no. 94-4333 (1994)
14. Bard JF (1983) An algorithm for solving the general bilevel programming program. Math Oper Res 8:260–272
15. Bard JF, Falk JE (1982) An explicit solution to the multilevel programming problem. Comput Oper Res 9:77–100
16. Barthelemy J-FM (1988) Engineering applications of heuristic multilevel optimization methods. NASA TM-101504
17. Barthelemy J-FM, Riley MF (1988) Improved multilevel optimization approach for the design of complex engineering systems. AIAA J 26:353–360
18. Basar T, Selbuz H (1979) Closed loop Stackelberg strategies with applications in optimal control of multilevel systems. IEEE Trans Autom Control AC-24:166–178
19. Benders JF (1962) Partitioning procedures for solving mixed variables programming problems. Numerische Math 4:238–252
20. Bialas WF, Karwan MH (1982) On two-level optimization. IEEE Trans Autom Control AC-27:211–214
21. Bracken J, McGill J (1973) Mathematical programs with optimization problems in the constraints. Oper Res 21:37–44
22. Braun RD (1996) Collaborative optimization: An architecture for large-scale distributed design. PhD Thesis Stanford Univ.
23. Braun RD, Moore AA, Kroo IM (1997) Collaborative approach to launch vehicle design. J Spacecraft and Rockets 34:478–486
24. Burke JV (1991) Calmness and exact penalization. SIAM J Control Optim 29:493–497
25. Burke JV (1991) An exact penalization viewpoint of constrained optimization. SIAM J Control Optim 29:968–998
26. Calamai PH, Vicente LN (1993) Generating linear and linear-quadratic bilevel programming problems. SIAM J Sci Comput 14:770–782
27. Candler W, Townsley R (1982) A linear two-level programming problem. Comput Oper Res 9:59–76
28. Chen Y (1994) Bilevel programming problems: Analysis, algorithms and applications. PhD Thesis Univ. Montreal
29. Chidambaram B, Rao JRJ (1994) A study of constraint activity in bilevel models of optimal design. Techn Report Univ Houston UH-ME-SDL-94-01
30. Clarke FH (ed) (1990) Optimization and nonsmooth analysis. SIAM, Philadelphia
31. Danzig GB, Wolfe P (1960) Decomposition principle for linear programming. Oper Res 8:101–111
32. De Luca A, Di Pillo G (1987) Exact augmented Lagrangian approach to multilevel optimization of large-scale systems. Internat J Syst Sci 18:157–176
33. De Miguel W, Murray W (2006) A Local Convergence Analysis of Bilevel Decomposition Algorithms. Optim Eng 7:99–133
34. De Silva AH, McCormick GP (1992) Implicitly defined optimization problems. Ann Oper Res 34:107–124
35. Dennis JE Jr, Schnabel RB (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs, NJ
36. Dirickx YMI, Jennergren LP (1979) Systems analysis by multilevel methods. Wiley, New York
37. Edmunds TA, Bard JF (1991) Algorithm for nonlinear bilevel mathematical programs. IEEE Trans Syst, Man Cybern 21:83–89
38. El-Alem MM (1991) A global convergence theory for the {Celis–Dennis–Tapia} trust region algorithm for constrained optimization. SIAM J Numer Anal 28:266–290
39. Falk JE, Liu J (1995) On bilevel programming, Part I: General nonlinear cases. Math Program 70:47–72
40. Fiacco AV (ed) (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Acad. Press, New York
41. Fiacco AV, McCormick GP (eds) (1990) Nonlinear programming, sequential unconstrained minimization techniques. SIAM, Philadelphia
42. Flippo OE (1989) Stability, duality and decomposition in general mathematical programming. PhD Thesis, Erasmus Univ. Rotterdam
43. Flippo OE, Rinnooy Kan AHG (1993) Decomposition in general mathematical programming. Math Program 60:361–382
44. Fortuny-Amat J, McCarl B (1981) A representation of a two-level programming problem. J Oper Res Soc 32:783–792
45. Friesz T, Tobin R, Cho H, Mehta N (1990) Sensitivity analysis based heuristic algorithms for mathematical programs with variational inequality constraints. Math Program 48:265–284
46. Gahutu DWH, Looze DP (1985) Parametric coordination in hierarchical control. Large Scale Systems 8:33–45
47. Gauvin J (1978) The method of parametric decomposition in mathematical programming: the nonconvex case.

In: Lemarechal C, Griffin R (eds) Nonsmooth optimization. Pergamon, Oxford, pp 131–149
48. Gauvin J, Dubeau F (1983) Some examples and counterexamples for stability analysis of nonlinear programming problems. In: Fiacco AV (ed) Math. Program. Stud., vol 21. North-Holland, Amsterdam, pp 69–78
49. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
50. Goulbeck B, Brdys M, Orr CH, Rance JP (1988) A hierarchical approach to optimized control of water distribution systems: Part I, decomposition. Optimal Control Appl Meth 9:51–61
51. Goulbeck B, Brdys M, Orr CH, Rance JP (1988) A hierarchical approach to optimized control of water distribution systems: part II, lower-level algorithm. Optimal Control Appl Meth 9:109–126
52. Haimes YY (1977) Hierarchical analyses of water resources systems. McGraw-Hill, New York
53. Haimes YY, Tarvainen K, Shima T, Thadathil J (1990) Hierarchical multiobjective analysis of large-scale systems. Hemisphere, Washington, DC
54. Hafka RT, Watson LT (2005) Multidisciplinary Design Optimization Problems with Quasiseparable Subsystems. Optim Eng 6:9–20
55. Hafka RT, Watson LT (2006) Decomposition Theory for Multidisciplinary Design Optimization Problems with Mixed Integer Quasiseparable Subsystems. Optim Eng 7:135–149
56. Ishizuka Y, Aiyoshi E (1992) Double penalty method for bilevel optimization problems. Ann Oper Res 34:73–88
57. Kirsch U (1989) Improved optimum structural design by passive control. Engin with Comput 5:13–22
58. Kodiyalam S (1998) Evaluation of methods for multidisciplinary design optimization (MDO), phase I. NASA Contractor Report
59. Kolstad CD, Lasdon LS (1990) Derivative evaluation and computational experience with large bilevel mathematical programs. J Optim Th Appl 65:485–499
60. Lasdon LS (1970) Optimization theory for large systems. MacMillan, New York
61. Loridan P, Morgan J (1988) Quasiconvex lower level problems and applications in two-level optimization. Techn Report Univ Montreal CRM-1578
62. Mahmoudi MS (1977) Multilevel systems control and applications: A survey. IEEE Trans Syst, Man Cybern SMC-7:125–143
63. Marcotte P (1985) A new algorithm for solving variational inequalities with application to the traffic assignment problem. Math Program 33:339–351
64. Marcotte P, Zhu DL (1996) Exact and inexact penalty methods for the generalized bilevel programming problem. Math Program 74:141–157
65. Mesarović MD, Macko D, Takahara Y (1970) Theory of hierarchical, multilevel, systems. Acad. Press, New York
66. Migdalas A, Pardalos PM, Värbrand P (eds) (1998) Multilevel optimization: Algorithms and applications. Kluwer, Dordrecht
67. Moré JJ (1991) Recent developments in software for trust region methods. In: Bachem A, Grötschel M, Korte B (eds) Mathematical Programming: The State of the Art. Springer, Berlin, pp 266–290
68. Muralidhar R, Rao JRJ, Badhrinath K, Kalagatla A (1995) Multilevel formulations and limit analysis and design of structures with bilateral contact constraints. Techn Report Univ Houston UH-ME-SDL-95-02
69. Nachane DM (1984) Optimization methods in multilevel systems: A methodological survey. Europ J Oper Res 21:25–38
70. Nicholls MG (1995) Aluminium production modelling – a non-linear bi-level programming approach. Oper Res 43:208–218
71. Outrata J (1994) On optimization problems with variational inequality constraints. SIAM J Optim 4:340–357
72. Padula SL, Alexandrov NM, Green LL (1996) MDO test suite at NASA Langley Research Center. In: Proc. Sixth AIAA/NASA/ISSMO Symp. Multidisciplinary Analysis and Optimization, AIAA
73. Padula SL, Young KC (1986) Simulator for multilevel optimization research. NASA Techn Memorandum TM-87751
74. Rao JRJ, Badhrinath K (1996) Solution of multilevel structural design problems using a nonsmooth algorithm. In: Proc. Sixth AIAA/NASA/ISSMO Symp. Multidisciplinary Analysis and Optimization, AIAA, AIAA-96-3986-CR.
75. Rao JRJ, Chidambaram B (1993) Parametric deformation and model optimality in concurrent design. Techn Report Univ Houston UH-ME-SDL-93-01
76. Reddy SY, Fertig KW, Smith DE (Aug. 1996) Constraint management methodology for conceptual design tradeoff studies. In: Proc. DETC'96, ASME Paper 96-DETC/DTM-1228
77. Rockafellar RT (1984) Directional differentiability of the optimal value function in a nonlinear programming problem. Math Program Stud 21:312–226
78. Rogers JL (1996) DeMAID/GA – An enhanced design manager's aid for intelligent decomposition. Proc. Sixth AIAA/NASA/ISSMO Symp. Multidisciplinary Analysis and Optimization,), AIAA Paper 96-4157.
79. Rogers JL (1996) DeMAID/GA user's guide – Design manager's aid for intelligent decomposition with a genetic algorithm. NASA Langley Res Center TM-110241
80. Schmitt LA, Chang KJ (1984) A multilevel method for structural synthesis. In: A Collection of Technical Papers: AIAA/ASME/ASCE/AHS 25th Structures, Structural Dynamics and Materials Conf., AIAA
81. Schmitt LA Jr, Mehrinfar M (1982) Multilevel optimum design of structures with fiber-composite stiffened panel components. AiAA J 20(1):138–147

82. Schmitt LA Jr, Ramanathan RK (1978) Multilevel approach to minimum weight design inclusing buckling constraints. AiAA J 16(2):97–104
83. Shimizu K, Aiyoshi E (1981) A new computational method for Stackelberg and min-max problems by use of a penalty method. IEEE Trans Autom Control AC-26:460–466
84. Simaan M, Cruz JB Jr (1973) On the Stackelberg strategy in nonzero-sum games. J Optim Th Appl 11(5):535–555
85. Singh MG, Mahmoud MS, Titli A (1981) A survey of recent developments in hierarchical optimization and control. Proc. IFAC Control Sci. and Techn. 8th Triennial World Congress, Kyoto, Japan, IFAC
86. Sobieski I, Kroo I (1996) Aircraft design using collaborative optimization. In: AIAA paper 96-0715 Presented at the 34th AIAA Aerospace Sci. Meeting, Reno, Nevada, Jan. 15-18, 1996, AIAA
87. Sobieszczanski-Sobieski J (1993) Optimization by decomposition. In: Kamat MP (ed) Structural Optimization: Status and Promise. Progress in Astronautics and Aeronautics, vol 150. AIAA, pp 487–515
88. Sobieszczanski-Sobieski J (1993) Two alternative ways for solving the coordination problem in multilevel optimization. Structural Optim 6:205–215
89. Sobieszczanski-Sobieski J (1996) Multidisciplinary aerospace design optimization: Survey of recent developments. In: Proc. 34-th Aerospace Sci. Meeting and Exhibit, Reno, Nevada, AIAA, AIAA paper 96-0711.
90. Sobieszczanski-Sobieski J, Haftka RT (1997) Multidisciplinary aerospace design optimization: survey of recent developments. Structural Optim 14:1–23
91. Sobieszczanski-Sobieski J, James BB, Dovi AR (1985) Structural optimization by generalized multilevel optimization. AIAA J 23:1775–1782
92. Stackelberg H (ed) (1952) The theory of the market economy. Oxford Univ. Press, Oxford
93. Tappeta RV, Renaud JE (1997) Multiobjective collaborative optimization. J Mechanical Design 119:403–411
94. Vicente LN, Calamai PH (1994) Bilevel and multilevel programming: A bibliographic review. J Global Optim 5:291–306
95. Wagner TC (1993) A general decomposition methodology for optimal system design. PhD Thesis, Univ. Michigan
96. Walsh JL, LaMarsh WJ, Adelman HM (1992) Fully integrated aerodynamic/dynamic/structural optimization of helicopter rotor blades. NASA TM-104226
97. Walsh JL, Young KC, Pritchard JI, Adelman HM, Mantay WR (1995) Integrated aerodynamic/dynamic/structural optimization of helicopter rotor blades using multilevel decomposition. NASA TP-3465
98. Wismer DA (ed) (1971) Optimization methods for large-scale systems. McGraw-Hill, New York
99. Ye JJ, Zhu DL, Zhu QJ (1997) Exact penalization and necessary optimality conditions for generalized bilevel programming problems. SIAM J Optim 7:481–507
100. Zhang R (1994) Problems of hierarchical optimization in finite dimensions. SIAM J Optim 4:521–536

Multilevel Optimization in Mechanics

Georgios E. Stavroulakis[1], Euripidis Mistakidis[2], Olympia Panagouli[3]
[1] Carolo Wilhelmina Techn. Universität, Braunschweig, Germany
[2] University Thessaly, Volos, Greece
[3] Aristotle University, Thessaloniki, Greece

MSC2000: 49Q10, 74K99, 74Pxx, 90C90, 91A65

Article Outline

Keywords
Large Cable Structures
Large Elastoplastic Structures
Validation and Improvements of Simplified Models
Decomposition Algorithms for Nonconvex Minimization Problems
Structures with Fractal Interfaces
See also
References

Keywords

Multilevel optimization; Computational mechanics; Parallel computation in mechanics

Multilevel optimization methods have been developed first in the period after 1960. The main scope was to facilitate the optimization of large scale systems in industrial processes and to solve trajectory determination and prediction problems using trajectory decomposition techniques. The reader may refer in this respect to the corresponding articles [3] and [26] and to the references given there but also to the books [27] and [12]. More recent works on this subject have been published in [4,14]. It should be mentioned that certain sources concerning the ideas of multilevel optimization may be found in well-known treatises of calculus of variations and theoretical mechanics, cf. e. g. [5,10]. Indeed, the well-known procedure of variational methods in Mechanics of 'frozen' variables or constraints has a great

relationship with the ideas of multilevel optimization. Also the well-known iterative methods of H. Cross and G. Kany of linear structural analysis used after 1940 and before the development of computer codes based on the *finite element method* (FEM), for the calculation of framed structures, are nothing than a formulation in the 'language' of structural analysis of a multilevel optimization algorithm for the minimization problem of the complementary energy of the structure, expressed in terms of the bending moments of the beam and column connections.

Among the pioneers in the application of the multilevel optimization methods in mechanics and especially concerning the calculation of structures involving inequality constraints was P.D. Panagiotopoulos [19,20]. The idea was the following: Most mechanical problems can be expressed as the minimum problems of an appropriately formulated energy function. The decomposition of this initial optimization problem into smaller subproblems corresponds to the energetic decomposition of the initial mechanical problem into smaller fictitious subproblems. The mutual interaction of these subproblems yields, after an iterative procedure, the solution of the initial problem. The aforementioned method leads to the following three main applications of the multilevel optimization techniques in the framework of Mechanics and more generally in engineering sciences.

a) Calculation of large structures.
b) Validation of the simplifying assumptions used for the calculation of complex structures. Accuracy testing.
c) Accuracy improvement of simplified models used for the estimation of the behavior of complex structures.

Note that in the above, the term 'structure' can be replaced with the term 'systems', meaning systems whose behavior is characterized by the solution of a minimax problem.

Since most of the multilevel techniques developed in the early sixties for the trajectory determination problems in space science are also applicable to stationarity problems, and since recently it has been proved that in the dynamic problems involving impact phenomena the functional of the action is stationary [22,23] it results that there is also a further application of the multilevel optimization methods:

d) Calculation of the dynamic behavior of structures involving impact effects.

To the aforementioned applications the following, classical one, can be added.

e) Solution of optimal control (minimum of weight or cost, maximum of strength) in dynamic structural analysis problems.

This article deals mainly with static systems. Concerning the application d) and e) the reader is referred to [12,27] in relation with [22,23]. In dynamic problems analogous methods to the static problems can be developed.

The classical *decomposition techniques* which are applied to optimization problems (cf. in this respect also [20, pp. 355ff]) have been extended and they can be applied also to *substationarity problems* [25], i. e. to problems of the type

$$0 \in \overline{\partial} f(x),$$

where f is a nonconvex nonsmooth energy function and $\overline{\partial}$ denotes the generalized gradient of F.H. Clarke [7] as it has been extended by R.T. Rockafellar [25] for nonLipschitzian functionals. In this case the *variational inequalities* of the convex energy problems are replaced by *hemivariational inequalities* (cf. e. g. [8,17,20,21]) and instead of a global minimum of the convex potential or complementary energy functionals, the local minima and maxima are searched and among them the global minimum as well. For the numerical treatment of hemivariational inequalities certain numerical methods have been developed (cf. e. g. [21]) and among them, the two methods described in [15] are extensions of the multilevel optimization methods to substationarity problems.

It should also be noted that most of the domain decomposition methods are special cases of the multilevel optimization algorithms, as it results easily if one considers the energy functionals corresponding to the partial differential equations studied. Then the domain decomposition leads to energy functionals which have to be minimized on the decomposed parts of the domain.

Finally, it should be mentioned that fractal geometries in optimization problems arising in Mechanics are treated by means of appropriate multilevel transformations of the problem as is will be shown further. It is evident that an optimization problem with many variables cannot always directly be decomposed into indepen-

dent optimization subproblems. The aim of the multilevel optimization is to define with respect to an optimization problem, appropriate mutually independent subproblems. Each of these when solved independently yields the optimum of the overall problem after an iterative procedure which is called second-level controller. The decomposition into subproblems is achieved by choosing some variables, called coordinating variables, which are freely manipulated by the second-level controller in such a way that the subproblems (first-level of the problem) have solutions which in fact yield the optimum of the initial problem, i. e. before its decomposition into subproblems. Here, the ideas of [3] are closely followed.

There are several different methods of transforming a given constrained optimization problem into a multilevel optimization problem. All these methods are basically combination of two methods: the *feasible decomposition method* or *model coordination method* and the *nonfeasible decomposition method* or *goal coordination method*.

Let us consider the problem

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{u}} & \Pi(\mathbf{x},\mathbf{u}) \\ \text{s.t.} & \mathbf{f}(\mathbf{x},\mathbf{u}) = 0 \\ & \mathbf{R}(\mathbf{x},\mathbf{u}) \geq 0, \end{cases} \tag{1}$$

where $\mathbf{x}$ is a vector in E_n, $\mathbf{u}$ is a vector in E_m, $\mathbf{f}$ is an n vector of C^2 functions, Π is a twice continuously differentiable (C^2) function, and $\mathbf{R}$ is an r vector of C^2 functions. To decompose, coordinating variables $\mathbf{s}$ may be substituted not only for a single variable but also, for functions $\mathbf{g}(\mathbf{x},\mathbf{u})$, so that Π is splitted into mutually disjoint parts and the $\mathbf{f}$ and $\mathbf{R}$ equations contain no common $\mathbf{x}$, $\mathbf{u}$, or $\mathbf{s}$ variables between the subproblems. Thus the following problem results:

$$\Pi(\mathbf{x},\mathbf{u},\mathbf{s}) = \sum_{i=1}^{N} \Pi^{(i)}(\mathbf{x}^{(i)},\mathbf{u}^{(i)},\mathbf{s}^{(i)})$$
$$\mathbf{f}^{(i)}(\mathbf{x}^{(i)},\mathbf{u}^{(i)},\mathbf{s}^{(i)}) = 0, \quad i = 1,\ldots,N,$$
$$\mathbf{R}^{(i)}(\mathbf{x}^{(i)},\mathbf{u}^{(i)},\mathbf{s}^{(i)}) \geq 0, \quad i = 1,\ldots,N.$$

The (i) denotes to the ith subproblem or subsystem which must be optimized. For example in a control problem $\mathbf{x}$ denotes the state, $\mathbf{u}$ denotes the control and $\mathbf{x}^{(1)}$ is the state vector for the first subsystem. Also the coupling equations must be added:

$$\mathbf{s}^{(i)} = \mathbf{g}^{(i)}(\mathbf{x}^{(j)},\mathbf{u}^{(j)}) \quad \text{for all } j \neq i.$$

The Lagrangian of the new problem reads

$$\widetilde{\Pi}(\mathbf{x},\mathbf{u},\mathbf{s};\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\rho}) = \sum_{i=1}^{N} \Pi^{(i)} + \sum_{i=1}^{N} \boldsymbol{\lambda}^{(i)\top}\mathbf{f}^{(i)} + \sum_{i=1}^{N} \boldsymbol{\mu}^{(i)\top}(\mathbf{R}^{(i)} - \boldsymbol{\sigma}^{(i)}) + \sum_{i=1}^{N} \boldsymbol{\rho}^{(i)\top}(\mathbf{g}^{(i)} - \mathbf{s}^{(i)}), \tag{2}$$

where $\boldsymbol{\sigma}^{(i)} \geq 0$ are additional slack variables such that

$$\mathbf{R}^{(i)} - \boldsymbol{\sigma}^{(i)} = 0.$$

$\widetilde{\Pi}$ is immediately separable into N individual subsystems, except for its last term.

In the method of nonfeasible decomposition it is assumed that $\boldsymbol{\rho}^{(i)}$ has a known value. The term $\boldsymbol{\rho}^{(i)\top}\mathbf{s}^{(i)}$ is put in the ith subsystem and all of the $\boldsymbol{\rho}^{(i)\top}\mathbf{g}^{(i)}(\mathbf{x}^{(j)},\mathbf{u}^{(j)})$ terms associated with the jth variables are put in the jth subsystem. On the other hand, in the feasible decomposition method it is assumed that $\mathbf{s}^{(i)}$ has a known value. Moreover, all of the $\boldsymbol{\rho}^{(i)\top}[\mathbf{g}^{(i)}(\mathbf{x}^{(j)},\mathbf{u}^{(j)}) - \mathbf{s}^{(i)}]$ terms associated with the jth variables are put in the jth subsystem. In both cases, the optimization problem is separable and each subsystem can be optimized independently. Equation (2) is rewritten in more compact form as

$$\widetilde{\Pi}(\mathbf{x},\mathbf{v};\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\rho}) = F(\mathbf{x},\mathbf{v}) + \boldsymbol{\lambda}^{\top}\mathbf{f}(\mathbf{x},\mathbf{v}) + \boldsymbol{\mu}^{\top}[\mathbf{R}(\mathbf{x},\mathbf{v}) - \boldsymbol{\sigma}] + \boldsymbol{\rho}^{\top}\mathbf{h}(\mathbf{x},\mathbf{v}), \tag{3}$$

where $\boldsymbol{\sigma} \geq 0$, $\mathbf{v}$ represents $\mathbf{u}$ and $\mathbf{s}$ and $\mathbf{h}(\mathbf{x},\mathbf{v})$ denotes all $\mathbf{g}^{(i)} - \mathbf{s}^{(i)}$, $\boldsymbol{\rho}$ is a Lagrange multiplier vector of the same dimension as $\mathbf{g}$, $\boldsymbol{\mu}$ is an r vector including all Lagrange multipliers, and $\boldsymbol{\lambda}$ is an n vector including all Lagrange multipliers.

The Kuhn–Tucker theory of nonlinear programming [9] implies that if $\Pi(\mathbf{x},\mathbf{v})$ has a critical point at $(\mathbf{x}^0,\mathbf{v}^0)$ such that the constraint equations in (1), are satisfied, and if the rank of

$$\left[\left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)^{\top}\left(\frac{\partial \mathbf{R}}{\partial \mathbf{y}}\right)^{\top}\left(\frac{\partial \mathbf{h}}{\partial \mathbf{y}}\right)^{\top}\right]$$

is full and equals the rank of

$$\left[\left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)^{\top}\left(\frac{\partial \mathbf{R}}{\partial \mathbf{y}}\right)^{\top}\left(\frac{\partial \mathbf{h}}{\partial \mathbf{y}}\right)^{\top}\left(\frac{\partial \Pi}{\partial \mathbf{y}}\right)^{\top}\right], \tag{4}$$

where

$$\mathbf{y} \equiv \begin{pmatrix}\mathbf{x}\\ \mathbf{v}\end{pmatrix}$$

at $(\mathbf{x}^0, \mathbf{v}^0)$, then a set of unique Lagrange multipliers $\boldsymbol{\lambda}^0$, $\boldsymbol{\mu}^0$ and $\boldsymbol{\rho}^0$ exist at the critical point. The necessary conditions for a critical point (local minimum) are

$$\frac{\partial \widetilde{\Pi}}{\partial \mathbf{x}} = \frac{\partial \widetilde{\Pi}}{\partial \mathbf{v}} = 0, \quad \mu_i R_i = 0, \quad \mathbf{R} \geq 0, \quad \boldsymbol{\mu} \leq 0, \tag{5}$$

$$\frac{\partial \widetilde{\Pi}}{\partial \boldsymbol{\lambda}} = \mathbf{f}^{\top} = 0, \quad \frac{\widetilde{\Pi}}{\partial \boldsymbol{\rho}} = \mathbf{h}^{\top} = 0. \tag{6}$$

If $\Pi(\mathbf{x}, \mathbf{v})$ is convex, if $f_i(\mathbf{x}, \mathbf{v})$ and $h_i(\mathbf{x}, \mathbf{v})$ are convex for λ_i^0 and ρ_i^0 positive, or if $f_i(\mathbf{x}, \mathbf{v})$, $h_i(\mathbf{x}, \mathbf{v})$, $R_i(\mathbf{x}, \mathbf{v})$ are concave for λ_i^0, ρ_i^0, μ_i^0 negative, and the above necessary conditions are satisfied, then $\Pi(\mathbf{x}^0, \mathbf{v}^0)$ is the absolute minimum of (1) and $\widetilde{\Pi}$ has a global saddle point at $(\mathbf{x}^0, \mathbf{v}^0)$; that is,

$$\begin{aligned}\widetilde{\Pi}(\mathbf{x}, \mathbf{v}; \boldsymbol{\lambda}^0, \boldsymbol{\mu}^0, \boldsymbol{\rho}^0) &\geq \widetilde{\Pi}(\mathbf{x}^0, \mathbf{v}^0; \boldsymbol{\lambda}^0, \boldsymbol{\mu}^0, \boldsymbol{\rho}^0)\\ &\geq \widetilde{\Pi}(\mathbf{x}^0, \mathbf{v}^0; \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\rho})\end{aligned}$$

for all $\mathbf{x}$, $\mathbf{v}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, and $\boldsymbol{\rho}$. These conditions can be relaxed to local convexity and concavity such that only a local minimum and saddle point are assured.

The *nonfeasible gradient controller* of L.C. Lasdon and J.D. Schoeffler [11] has the following form: Given (1), suppose that

a) $\widetilde{\Pi}$ has a global saddle point at $(\mathbf{x}^0, \mathbf{v}^0; \boldsymbol{\lambda}^0, \boldsymbol{\mu}^0, \boldsymbol{\rho}^0)$; and
b) for any given $\boldsymbol{\rho}$, a finite constrained (unique) minimum (constrained by $\mathbf{f}$ and $\mathbf{R}$) exists.

Then the iterative procedure given by

$${}^{i+1}\boldsymbol{\rho} = {}^{i}\boldsymbol{\rho} + \Delta\boldsymbol{\rho},$$

where

$$\Delta\boldsymbol{\rho} = +\kappa \mathbf{h}(\mathbf{x}^*, \mathbf{v}^*), \quad \text{with } \kappa > 0,$$

will converge to $\boldsymbol{\rho}^0$ and the absolute minimum of (1). Note that a local saddle point can replace a), then the initial guess on $\boldsymbol{\rho}$ must be within this saddle region. However, then the algorithm leads only to a local minimum. This Lasdon gradient controller can be considered as a variant of the modified *Arrow–Hurwicz gradient method* of K. Arrow, L. Hurwicz and H. Uzawa [1].

The *feasible gradient controller* of C.B. Brosilow et al. [6] has the following form: Given (1), suppose that

a) a finite minimum exists at $(\mathbf{x}^0, \mathbf{v}^0)$; and
b) all the conditions of (5) and (6) are fulfilled except for $\partial\widetilde{\Pi}/(\partial \mathbf{s}) = 0$, (where $\mathbf{v}$ denotes all $\mathbf{s}$ and $\mathbf{u}$).

Then the iterative procedure given by

$${}^{i+1}\mathbf{s} = {}^{i}\mathbf{s} + \Delta\mathbf{s},$$

where

$$\Delta\mathbf{s} = -\kappa\left(\frac{\partial \widetilde{\Pi}}{\partial \mathbf{s}}\right), \quad \text{with } \kappa > 0,$$

will converge to $\mathbf{s}^0 = \mathbf{x}^0$ and the minimum of (1).

The good choice of κ is important for the gradient calculations. Then at the second level of the feasible method, we may write ([3, p. 142]) that

$$\mathrm{d}\boldsymbol{\Pi}^* = \frac{\partial \widetilde{\Pi}}{\partial \mathbf{s}}\mathrm{d}\mathbf{s} = -\kappa\frac{\partial \widetilde{\Pi}}{\partial \mathbf{s}}\left(\frac{\partial \widetilde{\Pi}}{\partial \mathbf{s}}\right)^{\top}, \quad \kappa > 0.$$

An estimate of the expected improvement is written as $-\alpha\overline{\boldsymbol{\Pi}}^*$, $\alpha > 0$, where α is usually 10% or so. Then

$$\kappa = \frac{\alpha\widetilde{\Pi}^*}{\left(\partial\widetilde{\Pi}/\partial \mathbf{s}\right)\left(\partial\widetilde{\Pi}/\partial \mathbf{s}\right)^{\top}}. \tag{7}$$

In the case of nonfeasible decomposition a similar equation may be obtained [3]:

$$\kappa = \frac{\alpha\widetilde{\Pi}^*}{\mathbf{g}^{\top}\mathbf{g}}. \tag{8}$$

Note that $\Delta\mathbf{s}$ and $\Delta\boldsymbol{\rho}$ become singular at the optimum if (7) and (8) are used, respectively, and therefore these values of $\Delta\mathbf{s}$ and $\Delta\boldsymbol{\rho}$ are not appropriate to obtain exact solutions.

There is also the possibility to apply a Newton–Raphson controller both for the feasible and for the nonfeasible method in the second level (cf. in this context [3, p. 173]).

For instance examining (5) and (6), it is obvious that the only necessary condition not satisfied by the subsystems is $\mathbf{g} = 0$ in the nonfeasible decomposition method. Thus the Newton–Raphson method has as task to solve $\mathbf{g} = 0$ by an iterative method at the second level.

Note that the main characteristic of the aforementioned methods, i. e. the decomposition into subsystems and the separable optimization applies also to nonsmooth convex or nonconvex optimization problems.

Large Cable Structures

Here a possibility offered in structural analysis by the multilevel optimization algorithms is presented. Certain subproblems do not contain inequalities, i. e. are bilateral, and thus they can be treated by the available classical (i. e. based only on inequalities) FEM programs.

In the majority of *cable structures* the number of cables and nodes is large, and so an optimization problem with a large number of unknowns and constraints must be solved. Here, a multilevel optimization technique suitable for the solution of this kind of optimization problem is proposed. The initial optimization problem is decomposed into a number of subproblems. In the 'first level' of the calculation, each subproblem is optimized separately, and in the 'second level' the solutions of these subproblems are combined to yield the overall optimum.

It is interesting to note that some of these subproblems constitute minimization problems without inequality constraints (corresponding to classical bilateral structures), and the algorithms for their numerical treatment are much faster. The initial problem is decomposed into two subproblems: the first involves only the displacement terms and corresponds to a structure resulting from the given one by considering that all the cables act as bars (capable of having compressive forces), and the second, including only the slackness terms, corresponds to a hypothetical slack structure. In order to perform the decomposition, the potential energy of the structure is written in the form

$$\Pi(\mathbf{u}, \mathbf{v}) = \Pi'(\mathbf{u}) + \Pi''(\mathbf{v}) + \mathbf{u}^\top \mathbf{G}\mathbf{K}_0\mathbf{v}, \tag{9}$$

where

$$\Pi'(\mathbf{u}) = \frac{1}{2}\mathbf{u}^\top \mathbf{K}\mathbf{u} - \mathbf{u}^\top(\mathbf{G}\mathbf{K}_0\mathbf{e}_0 + \mathbf{p}) \tag{10}$$

and

$$\Pi''(\mathbf{v}) = \frac{1}{2}\mathbf{v}^\top \mathbf{K}_0\mathbf{v}^\top + \mathbf{v}^\top(\mathbf{a} - \mathbf{K}_0\mathbf{e}_0). \tag{11}$$

In the above equations $\mathbf{u}$, $\mathbf{v}$, $\mathbf{p}$, $\mathbf{e}_0$ are the displacements, slackness, loading and initial strain vectors respectively, $\mathbf{K}_0$ is the natural stiffness matrix, $\mathbf{K}$ is the stiffness matrix of the assembled structure and $\mathbf{G}$ is the equilibrium matrix. Introducing the variable $\mathbf{w}$ the minimization problem (9) takes the form

$$\min \Pi(\mathbf{u}, \mathbf{v}, \mathbf{w}) = \Pi'(\mathbf{u}) + \Pi''(\mathbf{v}) + \mathbf{u}^\top \mathbf{G}\mathbf{K}_0\mathbf{w}.$$

The Lagrangian of this problem is

$$\Pi_1(\mathbf{u}, \mathbf{v}, \mathbf{w}) = \Pi(\mathbf{u}, \mathbf{v}, \mathbf{w}) + \boldsymbol{\rho}^\top(\mathbf{v} - \mathbf{w}),$$

where $\boldsymbol{\rho}$ is the vector of the Lagrange multipliers. The decomposition can be performed by means of two methods: the nonfeasible gradient controller method of Lasdon and Schoeffler and the feasible gradient controller method of Brosilow, Lasdon and Pearson [11]. In the nonfeasible gradient controller method the value of $\boldsymbol{\rho}$ is supposed to be constant in the first level, say $\boldsymbol{\rho}_1$, and the minimization problem decomposes into the two subproblems

$$\min_{\mathbf{u},\mathbf{w}} \{\Pi'(\mathbf{u}) + \mathbf{u}^\top \mathbf{G}\mathbf{K}_0\mathbf{w} - \boldsymbol{\rho}_1^\top \mathbf{w}\}$$

and

$$\min_{\mathbf{v}} \left\{\Pi''(\mathbf{v}) + \boldsymbol{\rho}_1^\top \mathbf{v}\colon\ \mathbf{v} + \mathbf{p} \geq 0\right\}.$$

After performing the optimization, the values of $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$, e. g. $\mathbf{u}_1$, $\mathbf{v}_1$ and $\mathbf{w}_1$, result. It is obvious that $\mathbf{v}_1 \neq \mathbf{w}_1$. The task of the second level is to estimate a new value of $\boldsymbol{\rho}$, e. g. $\boldsymbol{\rho}_2$ by means of the equation

$$\boldsymbol{\rho}_2 = \boldsymbol{\rho}_1 + \kappa(\mathbf{v}_1 - \mathbf{w}_1), \quad \kappa > 0,$$

where κ is a properly chosen constant (see, e. g., [11]), and to transmit this value to the first level. The optimization is performed again, new values $\mathbf{u}_2$, $\mathbf{v}_2$ and $\mathbf{w}_2$ result, etc., until the differences $\mathbf{v}_i - \mathbf{w}_i$ are made negligible. The algorithm converges in a finite number of steps, provided that the minima exist [11].

In the feasible gradient controller method, the value of $\mathbf{w}$ is taken as constant in the first level, e. g. $\mathbf{w}_1$, and

thus the initial problem decomposes into the two subproblems

$$\min_{\mathbf{u}}\{\Pi'(\mathbf{u}) + \mathbf{u}^\top \mathbf{G}\mathbf{K}_0\mathbf{w}_1\}$$

and

$$\min_{\mathbf{v},\boldsymbol{\rho}} \left\{\Pi_1''(\mathbf{v}) + \boldsymbol{\rho}^\top(\mathbf{v} - \mathbf{w}_1''):\ \mathbf{v} + b \geq 0\right\}.$$

As a result of the optimization, the values of $\mathbf{u}$, $\mathbf{v}$ and $\boldsymbol{\rho}$, e. g. $\mathbf{u}_1$, $\mathbf{v}_1$ and $\boldsymbol{\rho}_1$ are calculated. By means of the second level a new value of $\mathbf{w}$, e. g. $\mathbf{w}_2$, is estimated and transmitted to the first level. This value is given by the equation

$$\mathbf{w}_2 = \mathbf{w}_1 - \kappa \left(\frac{\partial \Pi_1(\mathbf{u}, \mathbf{v}, \mathbf{w})}{\partial \mathbf{w}}\right)_{\mathbf{w}=\mathbf{w}_1}, \quad \kappa > 0,$$

where κ is a properly chosen constant (see, e. g., [11]). The optimization yields a new set of values $\mathbf{u}_2$, $\mathbf{v}_2$ and $\boldsymbol{\rho}_2$ and the procedure is continued until the difference between the consecutive values of vector $\mathbf{w}$ becomes sufficiently small.

For numerical applications the reader is referred to [20].

Large Elastoplastic Structures

We consider here the holonomic *plasticity* model [13], (extension to nonholonomic plasticity problems is straightforward) described by the following equations:

$$\begin{aligned}
&\mathbf{e} = \mathbf{F}_0\mathbf{s},\\
&\mathbf{e} = \mathbf{e}_0 + \mathbf{e}_E + \mathbf{e}_P,\\
&\mathbf{e}_P = \mathbf{N}\boldsymbol{\lambda},\\
&\boldsymbol{\phi} = \mathbf{N}^\top\mathbf{s} - \mathbf{k},\\
&\boldsymbol{\lambda} \geq 0, \quad \boldsymbol{\phi} \leq 0, \quad \boldsymbol{\phi}^\top\boldsymbol{\lambda} = 0,
\end{aligned}$$

where $\mathbf{F}_0$ is the natural flexibility matrix of the structure, $\mathbf{e}$ the respective strain vector consisting of three parts, the initial strain $\mathbf{e}_0$, the elastic strain $\mathbf{e}_E$ and the plastic strain $\mathbf{e}_P$, $\boldsymbol{\lambda}$ are the plastic multipliers vector, $\boldsymbol{\phi}$ the yield functions, $\mathbf{N}$ is the matrix of the gradients of the yield functions with respect to the stresses and $\mathbf{k}$ is a vector of positive constants. The potential energy of the structure is written in the form

$$\Pi(\mathbf{u}, \boldsymbol{\lambda}) = \Pi'(\mathbf{u}) + \Pi''(\boldsymbol{\lambda}) - \mathbf{u}^\top\mathbf{G}\mathbf{K}_0\mathbf{N}\boldsymbol{\lambda}$$

where

$$\begin{aligned}
\Pi'(\mathbf{u}) &= \tfrac{1}{2}\mathbf{u}^\top\mathbf{K}\mathbf{u} - \mathbf{e}_0^\top\mathbf{K}_0\mathbf{G}^\top\mathbf{u} - \mathbf{p}^\top\mathbf{u},\\
\Pi''(\boldsymbol{\lambda}) &= \tfrac{1}{2}\boldsymbol{\lambda}^\top\mathbf{N}^\top\mathbf{K}_0\mathbf{N}\boldsymbol{\lambda} + \mathbf{e}_0^\top\mathbf{K}_0\mathbf{N}\boldsymbol{\lambda} - \mathbf{k}\boldsymbol{\lambda}.
\end{aligned}$$

Again, $\mathbf{K}$ is the stiffness matrix of the structure and $\mathbf{K}_0$ is the inverse of $\mathbf{F}_0$.

The solution of the problem can be obtained by minimizing the potential energy of the structure:

$$\min\{\Pi(\mathbf{u}, \boldsymbol{\lambda}):\ \boldsymbol{\lambda} \geq 0\}. \tag{12}$$

By introducing a new variable $\mathbf{w}$, (12) takes the form

$$\begin{aligned}
\min\big\{\Pi(\mathbf{u}, \boldsymbol{\lambda}, \mathbf{w}) = \Pi'(\mathbf{u}) + \Pi''(\boldsymbol{\lambda})\\
-\mathbf{u}^\top\mathbf{G}\mathbf{K}_0\mathbf{N}\mathbf{w}:\ \mathbf{w} = \boldsymbol{\lambda},\ \boldsymbol{\lambda} \geq 0\big\}.
\end{aligned} \tag{13}$$

As in the previous section, the decomposition can be performed by the two methods of the feasible and the nonfeasible gradient controller respectively. For the sake of brevity only the nonfeasible gradient method will be shown here. The Lagrangian of (13) is first considered

$$\Pi(\mathbf{u}, \boldsymbol{\lambda}, \mathbf{w}) = \Pi(\mathbf{u}, \boldsymbol{\lambda}, \mathbf{w}) + \boldsymbol{\rho}^\top(\boldsymbol{\lambda} - \mathbf{w})$$

and the minimization problem is decomposed in the following two subproblems

$$\min_{\mathbf{u},\mathbf{w}} \left\{\Pi'(\mathbf{u}) - \mathbf{u}^\top\mathbf{G}\mathbf{K}_0\mathbf{N}\mathbf{w} - \boldsymbol{\rho}^\top\mathbf{w}\right\} \tag{14}$$

and

$$\min_{\boldsymbol{\lambda}} \left\{\Pi_1''(\boldsymbol{\lambda}) + \boldsymbol{\rho}^\top\boldsymbol{\lambda}:\ \boldsymbol{\lambda} \geq 0\right\}. \tag{15}$$

In the first step it is supposed that the value of $\boldsymbol{\rho}$ is constant (say $\boldsymbol{\rho}_1$) and we take as a result from (14) and (15) the values $\mathbf{u}_1$, $\boldsymbol{\lambda}_1$ and $\mathbf{w}_1$. Obviously $\boldsymbol{\lambda}_1 \neq \mathbf{w}_1$. Then the second level controller estimates the new value of $\boldsymbol{\rho}$ from the equation

$$\boldsymbol{\rho}_2 = \boldsymbol{\rho}_1 + \kappa(\boldsymbol{\lambda}_1 - \mathbf{w}_1), \quad \kappa > 0,$$

and transmits it to the first level, and the procedure is continued until the differences $\boldsymbol{\lambda}_i - \mathbf{w}_i$ become appropriately small.

The same procedure can be applied also to holonomic models including hardening and to nonholonomic plasticity models [13].

Validation and Improvements of Simplified Models

In mechanics and engineering sciences as well as in economy, simplified models are often considered for the treatment of complicated problems, e. g. concerning the calculation of stresses in complex structures. In these models it is assumed that certain quantities do not influence considerably the solution of the problem. By means of the multilevel decomposition, a method which permits the *validation* of these models and the improvement of their accuracy can be developed. This idea is explained in the sequel.

A. Consider a large structure involving also some cables and assume that due to the pretension of the cables the structure is calculated as if the cables are rods, i. e. by ignoring the fact that a cable may become slack and then it has zero stresses. Then in the equations (9)–(11) $\mathbf{v} = 0$ and the solution of the minimum problem is obtained by solving an unconstrained minimization problem, i. e. by a linear system solver. In order to check whether the solution of the simplified model is close to the solution of the initial problem, in which some cables, say r, may become *slack*, i. e. $v_i > 0$, $i = 1, \ldots, r$, it is enough to verify whether the second level controller which gives a value of the slackness of the cables causes a significant change in the solution of the first level problem which corresponds to the simplified structure. Also the algorithm offers an improvement of the solution of the simplified model.

B. Here, the investigation of the mutual influence of two subsystems is presented. Consider two substructures connected together, for instance a cylindrical shell with a hemispherical shell covering the one end of the cylinder. The solution of the whole linear elastic structural compound minimizes, for a given external loading, the potential (or the complementary) energy of the whole structure. Let $\mathbf{x}_1$ (respectively, $\mathbf{x}_2$) be the variables of the cylindrical (respectively, the hemispherical) shell and let $\mathbf{z}$ be the common variables at the contact line which are common in both structures. In order to decompose the potential energy into two minimum problems, one containing the unknowns of the cylindrical shell and the other of the hemispherical shell, the common variables for the cylindrical (respectively, hemispherical) shell are denoted by $\mathbf{z}_1$ (respectively, $\mathbf{z}_2$) and thus the initial problem

$$\min_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{z}} \{\Pi(\mathbf{x}_1, \mathbf{x}_2, \mathbf{z}) = \Pi_1(\mathbf{x}_1, \mathbf{z}) + \Pi_2(\mathbf{x}_2, \mathbf{z})\}$$

is written as

$$\min_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{z}_1,\mathbf{z}_2} \{\Pi_1(\mathbf{x}_1, \mathbf{z}_1) + \Pi_2(\mathbf{x}_2, \mathbf{z}_2) \colon \ \mathbf{z}_1 - \mathbf{z}_2 = 0\}\,.$$

Here Π_1 (respectively, Π_2) denotes the potential or the complementary energy of the cylindrical (respectively, the hemispherical) shell. Thus it can be tested by the nonfeasible controller method how the difference $\mathbf{z}_1 - \mathbf{z}_2$ influences the solution of the problem. The procedure is similar in the case of elastoplastic structures with the difference that the minimum is constrained by inequalities.

The above procedure may find applications in estimating the influence of saddles on pipelines of rigidity rings on long tubes etc.

C. Note that in all the above cases the Lagrange multipliers have a precise meaning: they correspond in the sense of energy to the chosen coordinating variables, i. e., if the coordinating variables are stresses (respectively, strains) or forces (respectively, displacements) then the coordinating Lagrange multipliers are strains (respectively, stresses) or displacements (respectively, forces). Thus the feasible and the nonfeasible decomposition method have a precise mechanical meaning. In the first case the Lagrange multipliers, i. e. the strains (respectively, the stress) are controlled while in the second one the coordinating variables, i. e. the stress (respectively, the strain) of the links between the two substructures are controlled, in order to achieve the position of equilibrium of the whole structure.

D. Some of the resulting substructures may have a known analytical solution. Then this fact facilitates the calculation and may be applied as a test for the accuracy of the resulting solution via a numerical technique, e. g. by the FEM model. The procedure is described in [24].

E. The multilevel decomposition method can be used also as estimator of the sensitivity of the final solution to small changes of the system to be optimized [24]. This method may be used for example in estimating how a partial change in a structure influences the stress

and strain field of the structure without solving twice the structure.

Decomposition Algorithms for Nonconvex Minimization Problems

In unilateral contact problems with friction, Panagiotopoulos proposed in 1975 an algorithm [18] called later PANA-algorithm for the decomposition of the *quasivariational inequality* problems into two classical variational inequality problems which are equivalent to two minimization problems. Analogous decomposition methods of complicated problems using an analogous to [18] fixed point procedure can be applied to the treatment of much more complicated problems today involving nonconvex energy functions. This section is devoted to the study of multilevel decomposition algorithms for problems belonging to the general framework of the substationarity problems.

It is known that the equilibrium of an elastic body Ω in adhesive contact with a support Γ is governed by the following problem [17,21]: Find $u \in V$ such as to satisfy the hemivariational inequality

$$\alpha(u, v-u) + \int_\Gamma j_N^0(u_N, v_N - u_N)\mathrm{d}\Gamma + \int_\Gamma j_T^0(u_T, v_T - u_T)\mathrm{d}\Gamma \geq (f, v-u), \quad \forall v \in V. \tag{16}$$

Here u, v are the displacement fields, f are all the applied forces, (f, v) – usually a L^2 internal product – is the work of the applied forces, $\alpha(u, v)$ is the elastic strain energy which is usually a coercive form, j_N (respectively, j_T) denote the nonconvex, locally Lipschitz generally nonsmooth energy density functions of the adhesive forces in the normal (respectively, the tangential) direction to the interface Γ. It is assumed that the normal adhesive action is independent of the tangential adhesive action. Moreover, j_N^0, j_T^0 denote the directional derivative in the sense of Clarke [7], and u_N, v_N (respectively, u_T, v_T) denote the normal (respectively, tangential) component of the displacement with respect to Γ. The solution of the above problem can be obtained in most cases of practical interest (cf. [21]) under certain mild hypotheses which guarantee this equivalence, by solving the substationarity problem

$$0 \in \overline{\partial} I(u) = \overline{\partial}\left\{\frac{1}{2}\alpha(u,u) + \int_\Gamma j_N(u_N)\mathrm{d}\Gamma + \int_\Gamma j_T(u_T)\mathrm{d}\Gamma - (f,u)\right\},$$

where $\overline{\partial}$ denotes the generalized gradient of Clarke.

In engineering problems the nonconvex superpotentials (cf. e. g. [16]) j_N and j_T are not independent but they depend j_N (respectively, j_T) on the vectors S_T (respectively, S_N), where S_T, S_N are the reactions corresponding to u_T, u_N respectively. In this case a hemivariational inequality cannot be formulated. In order to solve this problem numerically one may apply the following procedure: In the first step it is assumed that S_N is given, say, $S_N^{(0)}$ and the problem ($S_N^{(0)}$ enters with its work into $(f_1^{(0)}, u)$)

$$0 \in \overline{\partial}\left\{\frac{1}{2}\alpha(u,u) + \int_\Gamma j_T(S_N^{(0)}, u_T)\mathrm{d}\Gamma - (f_1^{(0)}, u)\right\} \tag{17}$$

is solved. The above problem yields a value of S_T, say $S_T^{(1)}$. Then the problem

$$0 \in \overline{\partial}\left\{\frac{1}{2}\alpha(u,u) + \int_\Gamma j_N(S_T^{(1)}, u_N)\mathrm{d}\Gamma - (f_2^{(1)}, u)\right\} \tag{18}$$

is solved ($S_T^{(1)}$ enters with its work into $(f_2^{(1)}, u)$) yielding a new value of S_N, say $S_N^{(1)}$, and so on until the differences $\| S_N^{(i)} - S_N^{(i+1)} \|$ and $\| S_T^{(i)} - S_T^{(i+1)} \|$ at each point of the discretized interface Γ become appropriately small. Here $\| \cdot \|$ denotes the $\mathbf{R}^3$-norm because the values are checked pointwise. The first (respectively, second) problem with $j_N = 0$ (respectively, with $j_T = 0$) corresponds to the first level (respectively, to the second level). Applications of the above procedure can be found in [15,20,21].

Structures with Fractal Interfaces

In this section the attention is focused on the fractal geometry of interfaces where their behavior is modeled by means of an appropriate nonmonotone contact and friction mechanism. The interfaces of fractal geometry are analyzed here as a sequence of classical interface

subproblems. These classical subproblems result from the consideration of the *fractal interface* as the unique 'fixed point' of a given *iterative function system* (IFS), which consists of N contractive mappings w_i: $\mathbf{R}^2 \to \mathbf{R}^2$ with contractivity factors $0 \le s_i < 1$, $i = 1, \ldots, N$ [2]. According to this procedure, a fractal set A is the 'fixed point' of a transformation W i. e.

$$A = W(A) = \bigcup_{i=1}^{N} W_i(A),$$

where W_i is defined

$$W_i(B) = \{w_i(x):\ x \in B\}, \quad \forall B \in H(\mathbb{R}^2).$$

Generally a *fractal set* A is given by the relation:

$$A = \lim_{n\to\infty} W^{(n)}(B), \quad \forall B \in H(\mathbb{R}^2),$$

where $H(\mathbf{R}^2)$ is the space of all compact subsets of $\mathbf{R}^2$. Thus each level corresponds to a classical geometry approximating the fractal geometry. Within each level a new optimization problem is solved with the new data. Thus the multilevel character of the optimization problem results from the necessity to take into account the fractal geometry.

In the sequel a linear elastic structure occupying a subset Ω of $\mathbf{R}^3$ is considered. In its undeformed state the structure has a boundary Γ which is decomposed into two mutually disjoint parts Γ_U and Γ_F. It is assumed that on Γ_U(respectively, Γ_F) the displacements (respectively, the tractions) are given. In the structure Ω some cracks with interfaces Φ of fractal type are formed. These cracks in brittle materials frequently propagate along one or more irregular ways. In this case the fracture system may be considered to be a cluster of branches propagating in such a way that new branches in the $n + 1$ step are successively created from a former branch at the n step. In other words the fracture system can be modeled by an IFS procedure. Regarding now the boundary conditions on Φ, it is assumed that nonmonotone, possibly multivalued laws describe the behavior of each interface in the normal and tangential directions. More specifically, it is assumed that the following boundary conditions hold:

$$-S_N \in \overline{\partial} j_N(u_N, x),$$
$$-S_T \in \overline{\partial} j_T(u_T, x).$$

Then according to the previous section, an equilibrium position of Ω is characterized by the hemivariational inequality (16).

In this case, where the fractured body Ω with fractal interfaces Φ is studied, it is necessary to substitute in (16) the domain Γ with Φ. As it has been mentioned above, Φ is the fixed point of a given transformation denoted by W, i. e.

$$\Phi = W\Phi,$$
$$\Phi^{(n+1)} = W\Phi^{(n)},$$
$$\Phi^{(n)}_{n\to\infty} \to \Phi.$$

Thus, for each approximation $\Phi^{(n)}$ of the fractal interface Φ a structure $\Omega^{(n)}$ must be solved. Since $\Phi^{(n)}$ is an interface set with classical geometry the solutions $u^{(n)}$ and $\sigma^{(n)}$ (where $u^{(n)}$ and $\sigma^{(n)}$ are the corresponding displacement and stress fields) are obtained using numerical procedures for the solution of (17) and (18). This procedure is repeated several times by increasing n; at the limit $n \to \infty$, $u^{(n)}$ and $\sigma^{(n)}$ give the solution of the fractal interface problem.

See also

- ▶ Bilevel Fractional Programming
- ▶ Bilevel Linear Programming
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Bilevel Programming
- ▶ Bilevel Programming: Applications
- ▶ Bilevel Programming: Applications in Engineering
- ▶ Bilevel Programming: Implicit Function Approach
- ▶ Bilevel Programming: Introduction, History and Overview
- ▶ Bilevel Programming in Management
- ▶ Bilevel Programming: Optimality Conditions and Duality
- ▶ Multilevel Methods for Optimal Design
- ▶ Stochastic Bilevel Programs

References

1. Arrow KJ, Hurwicz L, Uzawa H (1985) Studies in linear and nonlinear programming. Stanford Univ Press, Palo Alto
2. Barnsley M (1988) Fractals everywhere. Acad Press, New York

3. Bauman EJ (1971) Trajectory decomposition. In: Leondes CT (ed) Optimization methods for large scale systems with applications. McGraw-Hill, New York
4. Bertsekas DP, Tsitsiklis JN (1989) Parallel and distributed computation: numerical methods. Prentice-Hall, Englewood Cliffs, last edition: Athena Sci, Belmont Mass, 1997.
5. Bolza O (1904) Lectures on the calculation of variations. Chicago, Chicago
6. Brosilow CB, Lasdon LS, Pearson JD (1965) A multi-level technique for optimization. In: Proc Joint Autom Control Conf. Rensselaer Polytech Inst, Troy
7. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
8. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
9. Hadley G (1964) Non-linear and dynamic programming. Addison-Wesley, Reading
10. Hamel G (1967) Theoretische Mechanik. Springer, Berlin
11. Lasdon LC, Schoeffler JD (1965) A multi-level technique for optimization. In: Proc Joint Autom Control Conf. Rensselaer Polytech Inst, Troy
12. Leondes CT (ed) (1968) Advances in control systems. Theory and applications. Acad Press, New York
13. Maier G (1968) A quadratic programming approach for certain classes of non linear structural problems. Meccanica 2:121–130
14. Migdalas A, Pardalos PM (1996) Special issue on hierarchical and bilevel programming. J Global Optim 8(3)
15. Mistakidis ES, Stavroulakis GE (1998) Nonconvex optimization in mechanics. Algorithms, heuristics and engineering applications by the FEM. Kluwer, Dordrecht
16. Moreau JJ, Panagiotopoulos PD, Strang G (eds) (1988) Topics in nonsmooth mechanics. Birkhäuser, Basel
17. Naniewicz Z, Panagiotopoulos PD (1995) Mathematical theory of hemivariational inequalities and applications. M. Dekker, New York
18. Panagiotopoulos PD (1975) A nonlinear programming approach to the unilateral contact – and friction – boundary value problem in the theory of elasticity. Ingen Archiv 44:421–432
19. Panagiotopoulos PD (1976) A variational inequality approach to the inelastic stress-unilateral analysis of cable structures. Comput Structures 6:133–139
20. Panagiotopoulos PD (1985) Inequality problems in mechanics and applications. Convex and nonconvex energy Functions. Birkhäuser, Basel, Russian Translation: MIR, Moscow 1989
21. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Berlin
22. Panagiotopoulos PD (1995) Modelling of nonconvex nonsmooth energy problems. Dynamic hemivariational inequalities with impact effects. J Comput Appl Math 63:123–138
23. Panagiotopoulos PD (1995) Variational principles for contact problems including impact phenomena. In: Raous M (ed) Contact Mechanics. Plenum, New York
24. Panagiotopoulos PD, Mistakidis ES, Stavroulakis GE, Panagouli OK (1998) Multilevel optimization methods in mechanics. In: Multilevel Optimization: Algorithms, Complexity and Applications. Kluwer, Dordrecht, pp 51–90
25. Rockafellar RT (1979) La théorie des sous-gradients et ses applications à l'optimization. Fonctions convexes et nonconvexes. Les Presses de l'Univ Montréal, Montréal
26. Schoeffler JD (1971) Static multilevel systems. In: Leondes CT (ed) Optimization methods for large scale systems with applications. McGraw-Hill, New York
27. Wismer DA (ed) (1971) Optimization methods for large scale systems with applications. McGraw-Hill, New York

Multi-objective Combinatorial Optimization

MOCO

JACQUES TEGHEM
Lab. Math. & Operational Research Fac., Polytechn. Mons, Mons, Belgium

MSC2000: 90C29, 90C27

Article Outline

Keywords
Generation of $E(P)$
  Direct Methods
  Two-Phase Method
  Heuristic Methods
Interactive Determination of a Good Compromise
  Goal Programming
  Interactive Two-Phase Methods and MOSA Method
See also
References

Keywords

Multi-objective programming; Combinatorial optimization

It is well known that, on the one hand, *combinatorial optimization* (CO) provides a powerful tool to formulate and model many optimization problems, on the

other hand, a multi-objective (MO) approach is often a realistic and efficient way to treat many real world applications. Nevertheless, until recently, Multi-objective combinatorial optimization (MOCO) did not receive much attention in spite of its potential applications. One of the reason is probably due to specific difficulties of MOCO models. We can distinguish three main difficulties. The first two are the same as those existing for multi-objective integer linear programming (MOILP) problem (cf. ► Multi-objective Integer Linear Programming), i. e.

- the number of *efficient solutions* may be very large;
- the nonconvex character of the feasible set requires to device specific techniques to generate the so-called 'nonsupported' efficient solutions (cf. ► Multi-objective Integer Linear Programming).

A particular single CO problem is characterized by some specificities of the problem, generally a special form of the constraints; the existing methods for such problem use these specificities to define efficient ways to obtain an optimal solution. For MOCO problem, it appears interesting to do the same to obtain the set of efficient solutions. Consequently, and contrary to what is often done in MOLP and MOILP methods, a third difficulty is to elaborate methods avoiding to introduce additional constraints so that we preserve during all the procedure the particular form of the constraints.

The general form of a MOCO problem is

$$(\mathrm{P}) \quad \begin{cases} \text{'}\min_{X\in S}\text{'} & z_k(X) = c_k X, \\ & \qquad k = 1, \ldots, K, \\ \text{where} & S = D \cap B^n \\ \text{with} & X(n \times 1), \\ & B = \{0, 1\} \end{cases}$$

and D is a specific polytope characterizing the CO problem: assignment problem, knapsack problem, traveling salesman problem, etc.

There exists several surveys on MOCO; some are devoted to specific problems (i. e., the particular form of D): the shortest path problem [8], transportation networks [2], and the scheduling problem [6,7]; the survey [9] is more general examining successively the literature on MO assignment problems, knapsack problems, network flow problems, traveling salesman problems, location problems, set covering problems.

In the present article we put our attention on the existing methodologies for MOCO. First we examine how to determine the set $E(\mathrm{P})$ of all the efficient solutions and we distinguish three approaches: direct methods, two-phase methods and heuristic methods. Subsequently we analyse interactive approaches to generate a 'good compromise' satisfying the decision maker.

Generation of $E(\mathrm{P})$

Direct Methods

The first idea is to use intensively classical methods for single objective problem (P) existing in the literature to determine $E(\mathrm{P})$. Of course, each time a feasible solution is obtained the k values $z_k(X)$ are calculated and compared with the list $\widehat{E(\mathrm{P})}$ containing all the feasible solutions already obtained and non dominated by another generated feasible solution. Clearly, $\widehat{E(\mathrm{P})}$, called the *set of potential efficient solutions*, plays the role of the so-called 'incumbent solution' in single objective methods. At each step, $\widehat{E(\mathrm{P})}$ is updated and at the end of the procedure $E(\mathrm{P}) = \widehat{E(\mathrm{P})}$. Such extension of single objective method is specially designed for enumerative procedure based on a *branch and bound* approach. Unfortunately, in a MO framework, a node of the branch and bound tree is less often fathomed than in the single objective case, so that logically such MO procedure is less efficient.

We describe below an example of such direct method, extending the well known Martello–Toth procedure, for the multi-objective *knapsack problem* formulated as

$$\begin{cases} \text{'}\max\text{'} & z_k(X) = \sum_{j=1}^{n} c_j^{(k)} x_j, \quad k = 1, \ldots, K, \\ & \sum_{j=1}^{n} w_j x_j \le W \quad x_j = (0, 1). \end{cases}$$

The following typical definitions are used ($k = 1, \ldots, K$):

- O_k: variables order according to decreasing values of c_j^k / w_j.
- $r_j^{(k)}$: the rank of variable j in order O_k.
- Θ: variables order according to increasing values of $\sum_{k=1}^{K} r_j^{(k)} / K$.

We assume that variables are indexed according to ordinal preference Θ.

At any node of the branch and bound tree, variables are set to 0 or 1; let B_0 and B_1 denote the index sets of variables assigned to the values 0 and 1, respectively. Let F be the index set of free variables which always follow, in the order Θ, those belonging to $B_1 \cup B_0$. If $i-1$ is the last index of fixed variables, we have $B_1 \cup B_0 = \{1, \ldots, i-1\}$; $F = \{i, \ldots, n\}$.

Initially, $i = 1$. Let

- $\overline{W} = W - \sum_{j \in B_1} w_j \geq 0$ be the leftover capacity of the knapsack.
- $\underline{Z} = \left(\underline{z}_k = \sum_{j \in B_1} c_j^{(k)}\right)_{k=1,\ldots,K}$ be the criteria values vector obtained with already fixed variables.
  $\widehat{E(\mathrm{P})}$ contains nondominated feasible values $\underline{Z}$ and is updated at each new step.
  Initially, $\underline{z}_k = 0$, $\forall\, k$, and $\widehat{E(\mathrm{P})} = \emptyset$.
- $\overline{Z} = (\overline{z}_k)$ be the vector whose components are upper bounds of feasible values respectively for each objective at considered node. These upper bounds are evaluated separately, for instance as in the Martello–Toth method.
  Initially, $\overline{z}_k = \infty$, $\forall\, k$.

A node is fathomed in the following two situations:

i) if $\{j \in F:\ w_j < \overline{W}\} = \emptyset$; or

ii) $\overline{z}$ is dominated by $z^* \in \widehat{E(\mathrm{P})}$.

When the node is fathomed, the backtracking procedure is performed: a new node is build up by setting to zero the variable corresponding to the last index in B_1. Let t be this index:

$$B_1 \leftarrow B_1 \backslash \{t\},$$
$$B_0 \leftarrow (B_0 \cap \{1, \ldots, t-1\}) \cup \{t\},$$
$$F \leftarrow \{t+1, \ldots, n\}.$$

When the node is nonfathomed, a new node of the branch and bound tree is build up for next iteration, as follows:

- Define s to be the index variable such that

$$\max \left\{ l \in F:\ \sum_{j=i}^{l} w_j < \overline{W} \right\}.$$

  If $w_i > \overline{W}$, set $s = i - 1$.
- If $s \geq i$:

$$B_1 \leftarrow B_1 \cup \{i, \ldots, s\},$$
$$B_0 \leftarrow B_0,$$
$$F \leftarrow F \backslash \{i, \ldots, s\}.$$

If $s = i - 1$,

$$B_1 \leftarrow B_1 \cup \{r\},$$
$$B_0 \leftarrow B_0 \cup \{i, \ldots, r-1\},$$
$$F \leftarrow F \backslash \{i, \ldots, r\},$$

with $r = \min \left\{ j \in F:\ w_j < \overline{W} \right\}$.

The procedure stops when the initial node is fathomed and then $E(\mathrm{P}) = \widehat{E(\mathrm{P})}$. An illustration is given in [10].

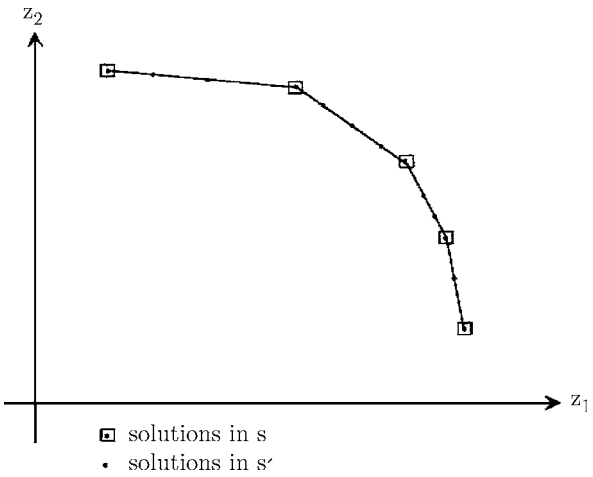
Two-Phase Method

Such an approach is particularly well designed for bi-objective MOCO problems. The first phase consists to determine the set $SE(\mathrm{P})$ of supported efficient solutions (see ► Multi-objective Integer Linear Programming). Let $S \cup S'$ be the list of supported efficient solutions already generated; S is initialized with the two efficient optimal solutions respectively of objectives z_1 and z_2. Solutions of S are ordered by increasing value of criterion 1; let X_r and X_s be two consecutive solutions in S, thus with $z_{1r} < z_{1s}$ and $z_{2r} > z_{2s}$, where $z_{kl} = z_k(X_l)$. The following single-criterion problem is considered:

$$(\mathrm{P}_\lambda) \quad \begin{cases} \min & z_\lambda(X) = \lambda_1 z_1(X) + \lambda_2 z_2(X) \\ & X \in S = D \cap B^{(n)} \\ & \lambda_1 \geq 0, \quad \lambda_2 \geq 0. \end{cases}$$

This problem is optimized with a classical single objective CO algorithm for the values $\lambda_1 = z_{2r} - z_{2s}$ and $\lambda_2 = z_{1s} - z_{1r}$; with these values the search direction $z_\lambda(X)$ corresponds in the objective space to the line defined by Z_r and Z_s. Let $\{X^t: t = 1, \ldots, T\}$ be the set of optimal solutions obtained in this manner and $\{Z_t: t = 1, \ldots, T\}$ their images in the objective space. There are two possible cases:

- $\{Z_r, Z_s\} \cap \{Z_t: t = 1, \ldots, T\} = \emptyset$: Solutions X^t are new supported efficient solutions. X^1 and X^T, provided $T > 1$, are put in S and, if $T > 2$, $X^2, \ldots, X^{T-1}$ are put in S'. It will be necessary at further steps to consider the pairs (X^r, X^1) and (X^T, X^s)
- $\{Z_r, Z_s\} \subset \{Z_t: t = 1, \ldots, T\}$: Solutions $\{X^t: t = 1, \ldots, T\} \setminus \{X^r, X^s\}$ are new supported efficient solutions giving the same optimal value as X^r and X^s for $z_\lambda(X)$; we put them in list S'.



Multi-objective Combinatorial Optimization, Figure 1
SE*(P) = *S* ∪ *S'

This first phase is continued until all pairs (X^r, X^s) of S have been examined without extension of S.

Finally, we obtain $SE(P) = S \cup S'$ as illustrated in Fig. 1.

The purpose of the second phase is to generate the set $NSE(P) = E(P) \setminus SE(P)$ of nonsupported efficient solutions. Each nonsupported efficient solution has its image inside the triangle $\Delta\ Z_r Z_s$ determined by two successive solutions X^r and X^s of $SE(P)$ (see Fig. 1). So each of the $|SE(P)| - 1$ triangles $\Delta\ Z_r Z_s$ are successively analysed. This phase is more difficult to manage and is dependent of the particular MOCO problem analysed; in general, this second phase is achieved using partly a classical single objective CO method. An example of such second phase is given in ► Bi-objective Assignment Problem and in [14] for the bi-objective knapsack problem.

Heuristic Methods

As pointed out in [9,10,14], it is unrealistic to extend the exact methods describe above to MOCO problems with more than two criteria or more than a few hundred variables; the reason is that these methods are too consuming time. Because a *metaheuristic*, simulating annealing (SA), tabu search (TS), genetic algorithms (GA), etc., provide, for the single objective problem, excellent solutions in a reasonable time, it appeared logical to try to adapt these metaheuristics to a multi-objective framework.

The seminal work in this direction is the 1993 Ph.D. thesis of E.L. Ulungu, which gave rise to the so-called *MOSA method* to approximate $E(P)$ (see, in particular, [11]). After this pioneer study, this direction has been tackled by other research teams: P. Czyzak and A. Jaszkiewicz ([3]) proposed another way to adapt simulating annealing to a MOCO problem; independently, [4,5] and [1] did the same with tabu search, the later combining also tabu search and genetic algorithms; genetic algorithms are also used in [13].

The principle idea of MOSA method can be resumed in short terms. One begins with an initial iterate X_0 and initializes the set of potentially efficient points PE to just contain X_0. One then samples a point Y in the neighborhood of the current iterate. But instead of accepting Y if it is better than the current iterate on an objective: we now accept it if it is *not dominated* by any of the points currently in the set PE. If it is not dominated, we make Y the current iterate, add it to PE, and throw out any point in PE that are dominated by Y. On the other hand, if Y is dominated, we still make it the current iterate with some probability. In this way, as we move the iterate through the space, we simultaneously build up a set PE of potentially efficient points. The only complicated aspect of this scheme is the method for computing the acceptance probability for Y when it is dominated by a point in PE. The MOSA method is described in details in [11] and in ► Bi-objective Assignment Problem.

Interactive Determination of a Good Compromise

The general idea of *interactive methods* is described in ► Multi-objective Integer Linear Programming. Two types of methods can be distinguished, which we treat in the following subsections.

Goal Programming

As pointed out in [9], this methodology is often used by American researchers to treat several case studies. The general idea of *goal programming* method is to introduce for each objective k deviation variables d^+ and d^-, respectively by excess and by default, with respect to a certain a priori goal g_k, so that goal constraints are defined. If some priorities expressed by some weights p_k are given, this results in a single-objective problem (P_g)

defined by the global weighted deviation function:

$$(\mathrm{P}_g) \quad \begin{cases} \min & \displaystyle\sum_{k=1}^{K} p_k d_k^- \\ \text{s.t.} & z_k(X) + d_k^+ - d_k^- = g_k, \quad \forall k, \\ & X \in S = D \cap B^n. \end{cases}$$

When a solution is obtained, the decision maker can possibly modify the values of the goals g_k before a new iteration is performed. One drawback is that the additional goal constraints induce the loss of the particular structure of the initial CO problem, so that a general ILP software must be used to solve problem (P_g).

Interactive Two-Phase Methods and MOSA Method

The two-phase methodology described above can easily be adapted to build interactively a good compromise. At each step of the first phase, the decision maker can indicate which pair (X_r, X_s) he prefers so that only a small subset of $SE(\mathrm{P})$ is generated in the direction given by the decision maker; at the second phase, only one (or a few number of) triangles $\Delta\, Z_r Z_s$ is (are) analysed to verify if there exists in it a more satisfying non-supported efficient solution. In the same spirit, an interactive MOSA method can be designed (see also [12]): the decision maker gives some goals g_k and only the solutions satisfying $z_k(X) \leq g_k$ are putting in the list of potential efficient solutions. When this list contains a certain a priori fixed number of solutions, the decision maker indicates which one is preferred, modifies the goals g_k in a more restrictive sense before to continue the search with MOSA.

An example of such interactive procedure is given in [12] for a real case study.

See also

- Bi-objective Assignment Problem
- Combinatorial Matrix Analysis
- Combinatorial Optimization Algorithms in Resource Allocation Problems
- Combinatorial Optimization Games
- Decision Support Systems with Multiple Criteria
- Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- Evolutionary Algorithms in Combinatorial Optimization
- Financial Applications of Multicriteria Analysis
- Fractional Combinatorial Optimization
- Fuzzy Multi-objective Linear Programming
- Multicriteria Sorting Methods
- Multi-objective Integer Linear Programming
- Multi-objective Optimization and Decision Support Systems
- Multi-objective Optimization: Interaction of Design and Control
- Multi-objective Optimization: Interactive Methods for Preference Value Functions
- Multi-objective Optimization: Lagrange Duality
- Multi-objective Optimization: Pareto Optimal Solutions, Properties
- Multiple Objective Programming Support
- Neural Networks for Combinatorial Optimization
- Outranking Methods
- Portfolio Selection and Multicriteria Analysis
- Preference Disaggregation
- Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- Preference Modeling
- Replicator Dynamics in Combinatorial Optimization

References

1. Ben Abdelaziz F, Chaouachi J, Krichen S (1997) A hybrid heuristic for multiobjective knapsack problems. Techn Report Inst Sup Gestion, Tunisie, submitted
2. Current JR, Min H (1986) Multiobjective design of transportation networks: taxonomy and annotation. Europ J Oper Res 26(2):187–201
3. Czyzak P, Jaszkiewicz A (1998) Pareto simulated annealing – A metaheuristic technique for multiple objective combinatorial optimization. J Multi-Criteria Decision Anal 7:34–47
4. Gandibleux X, Mezdaoui N, Fréville A (1997) A tabu search procedure to solve multiobjective combinatorial optimisation problems. In: Caballero R, Steuer R (eds) Proc. Volume of MOPGP'96. Springer, Berlin
5. Hansen MP (1996) Tabu search for multiobjective optimization: MOTS. Techn Report Inst Math Modelling, Techn Univ Denmark
6. Hoogeveen H (1992) Single machine bicriteria scheduling. PhD Diss, Univ Eindhoven
7. Köksalan M, Köksalan–Konda CkiS (1997) Multiple criteria scheduling on single machine: A review and a general approach. In: Karwan M, et al. (eds) Essays in Decision Making. Springer, Berlin

8. Ulungu EL, Teghem J (1991) Multi-objective shortest problem path: A survey. In: Cerny M, Glackaufova D, Loula D (eds) Proc Internat Workshop on MCDM, Liblice, pp 176–188
9. Ulungu EL, Teghem J (1994) Multi-objective combinatorial optimization problems: A survey. J Multi-Criteria Decision Anal 3:83–104
10. Ulungu EL, Teghem J (1997) Solving multi-objective knapsack problem by a branch and bound procedure. In: Climaco J (ed) Multicriteria Analysis. Springer, Berlin, pp 269–278
11. Ulungu EL, Teghem J, Fortemps Ph, Tuyttens D (1999) MOSA method: A tool for solving MOCO problems I. Multi-Criteria Decision Anal 8:221–236
12. Ulungu EL, Teghem J, Ost Ch (1998) Efficiency of interactive multi-objective simulated annealing through a case study. J Oper Res Soc 49:1044–1050
13. Viennet R, Fontex M (1996) Multi-objective combinatorial optimization using a genetic algorithm for determining a Pareto set. Internat J Syst Sci 27(2):255–260
14. Visée M, Teghem J, Pirlot M, Ulungu EL (1998) Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. J Global Optim 12:139–155

Multi-objective Fractional Programming Problems

ZHIAN LIANG
Department of Applied Mathematics,
Shanghai University of Finance and Economics,
Shanghai, P.R. China

MSC2000: 90C29

Article Outline

Keywords and Phrases
Introduction
Efficiency Conditions
Duality
  Mond-Weir Dual
  Schaible Dual
  Extended Bector Type Dual
Concluding Remarks
References

Keywords and Phrases

Multiobjective fractional programming problem; (F, α, ρ, d)-convex functions; Efficient solution; Efficiency condition; Duality

Introduction

A number of optimization problems are actually multiobjective optimization problems (MOPs), where the objectives are conflicting. As a result, there is usually no single solution which optimizes all objectives simultaneously. A number of techniques have been developed to find a compromise solution to MOPs. The reader is referred to the recent book by Miettinen [16] about the theory and algorithms for MOPs. Fractional programming problems(FPPs) arise from many applied areas such as portfolio selection, stock cutting, game theory, and numerous decision problems in management science. Many approaches for FPPs have been exploited in considerable details. See, for example, Avriel et al. [3], Craven [5], Schaible [24,25], Schaible and Ibaraki [26] and Stancu-Minasian [27,28].

In this paper, we consider the following multiobjective fractional programming problem:

$$\begin{aligned}(\mathrm{MFP})\quad &\min \frac{f(x)}{g(x)} \triangleq \left(\frac{f_1(x)}{g_1(x)}, \frac{f_2(x)}{g_2(x)}, \ldots, \frac{f_p(x)}{g_p(x)}\right)^{\mathrm{T}},\\ &\text{s.t.}\\ &h(x) \le 0, \quad x \in X,\end{aligned}$$

where $X \subset R^n$ is an open set, f_i, g_i $(i = 1, 2, \ldots, p)$ are real-valued functions defined on X, and h is an m-dimensional vector-valued function defined on X. Suppose that $f_i(x) \ge 0$ and $g_i(x) > 0$ for $x \in X$ and $i = 1, 2, \ldots, p$. Moreover, let f_i, g_i $(i = 1, 2, \ldots, p)$ and h_j $(j = 1, 2, \ldots, m)$ be continuously differentiable over X and denote the gradients of f_i, g_i and h_j at x by $\nabla f_i(x), \nabla g_i(x)$ and $\nabla h_j(x)$, respectively.

If the parameter p in the problem (MFP) is equal to 1, then (MFP) corresponds to the following single-objective fractional programming problem:

$$\begin{aligned}(\mathrm{FP})\quad &\min \frac{f(x)}{g(x)},\\ &\text{s.t. } h(x) \le 0, \quad x \in X,\end{aligned}$$

where $X \subset R^n$ is an open set, f, g are real-valued functions defined on X, and h is an m-dimensional vector-valued function defined on X, $f(x) \ge 0$ and $g(x) > 0$ for all $x \in X$. Moreover, assume that $f(x)$, $g(x)$ and $h_j(x)$ $(j = 1, 2, \ldots, m)$ are continuously differentiable over X.

Khan and Hanson [10], and Reddy and Mukherjee [21] considered the optimality conditions and duality for (FP) with respect to the following generalized concepts of convexity, respectively.

Definition 1 [6] Let f be a real function defined on an open set $X \subseteq R^n$ and differentiable at x_0. Given a mapping $\eta : X \times X \to R^n$, the function f is said to be invex at x_0 with respect to η if, $\forall x \in X$, the following inequality holds:

$$f(x) - f(x_0) \geq \nabla f(x_0)^{\mathrm{T}} \eta(x, x_0) .$$

Definition 2 [7] Let f be a real function defined on an open set $X \subseteq R^n$ and differentiable at x_0. Given a real number ρ, a mapping $\eta : X \times X \to R^n$ and a scalar function $d : X \times X \to R$, the function f is said to be ρ-invex at x_0 with respect to η and d if, $\forall x \in X$, the following inequality holds:

$$f(x) - f(x_0) \geq \nabla f(x_0)^{\mathrm{T}} \eta(x, x_0) + \rho d^2(x, x_0) .$$

The authors of references [10,21] imposed the corresponding generalized convexity on the numerator and denominator individually for the objective function in the problem (FP), and then derived some optimality conditions and duality results. How to extend these methods to the multiobjective case is still an open problem [21].

As far as the multiobjective fractional problem (MFP) is concerned, Jeyakumar and Mond [8] introduced a concept of v-invexity as follows.

Definition 3 Let $f : X \to R^p$ be a real vector function defined on an open set $X \subseteq R^n$ and each component of f be differentiable at x_0. The function f is said to be v-invex at $x_0 \in X$ if there exist a mapping $\eta : X \times X \to R^n$ and a function $\alpha_i : X \times X \to R_+ \setminus \{0\}$ $(i = 1, 2, \ldots, p)$ such that, $\forall x \in X$,

$$f_i(x) - f_i(x_0) \geq \alpha_i(x, x_0) \nabla f_i(x_0)^{\mathrm{T}} \eta(x, x_0) .$$

Jeyakumar and Mond [8] obtained some weak efficiency conditions and duality results for a nonconvex multiobjective fractional programming problem via the concept of v-invexity, v-pseudoinvexity and v-quasiinvexity.

Motivated by various concepts of generalized convexity, Liang et al. [12] introduced a unified formulation of the generalized convexity, which was called (F, α, ρ, d)-convexity, and obtained some corresponding optimality conditions and duality results for the single-objective fractional problem (FP). In this paper, we will extend the methods adopted for the single-objective problem (FP) in [12] to the multiobjective problem (MFP).

Definition 4 A function $F : R^n \to R$ is said to be sublinear if for any $\alpha_1, \alpha_2 \in R^n$,

$$F(\alpha_1 + \alpha_2) \leq F(\alpha_1) + F(\alpha_2) , \tag{1}$$

and for any $r \in R_+$, $\alpha \in R^n$,

$$F(r\alpha) = rF(\alpha) . \tag{2}$$

Note that the concept of the sublinear function was given in Preda [20]. Now, a sublinear function is defined simply as a function that is subadditive and positively homogeneous, which is free of extraneous symbols in Preda [20]. It follows from (2) that $F(0) = 0$.

Based upon the concept of the sublinear function, we recall the unified formulation about generalized convexity, i. e., (F, α, ρ, d)-convexity, which was introduced in [12] as follows.

Definition 5 Given an open set $X \subset \Re^n$, a number $\rho \in R$, and two functions $\alpha : X \times X \to R_+ \setminus \{0\}$ and $d : X \times X \to R$, a differentiable function f over X is said to be (F, α, ρ, d)-convex at $x_0 \in X$ if for any $x \in X$, $F(x, x_0; \cdot) : \Re^n \to \Re$ is sublinear, and $f(x)$ satisfies the following condition:

$$\begin{aligned} f(x) - f(x_0) \geq & F(x, x_0; \alpha(x, x_0) \nabla f(x_0)) \\ & + \rho d^2(x, x_0) . \end{aligned} \tag{3}$$

The function f is said to be (F, α, ρ, d)-convex over X if, $\forall x_0 \in X$, it is (F, α, ρ, d)-convex at x_0; f is said to be strongly $(F, \alpha, \rho, d) - convex$ or $(F, \alpha) - convex$ if $\rho > 0$ or $\rho = 0$, respectively.

From Definition 5, there are the following special cases:

(i) If $\alpha(x, x_0) = 1$ for all $x, x_0 \in X$, then (F, α, ρ, d)-convexity is (F, ρ)-convexity [20].

(ii) If $F(x, x_0; \alpha(x, x_0) \nabla f(x_0)) = \nabla f(x_0)^{\mathrm{T}} \eta(x, x_0)$ for a certain mapping $\eta : X \times X \to R^n$, then (F, α, ρ, d)-convexity is ρ-invexity defined in [7].

(iii) If $\rho = 0$ or $d(x, x_0) \equiv 0$ for all $x, x_0 \in X$ and $F(x, x_0; \alpha(x, x_0) \nabla f(x_0)) = \nabla f(x_0)^{\mathrm{T}} \eta(x, x_0)$

for a certain mapping $\eta : X \times X \to R^n$, then (F, α, ρ, d)-convexity reduces to invexity [6].

In the following, ρ, α and d are referred to as parameters of (F, α, ρ, d)-convexity. Furthermore, we will adopt the following conventions.

Let R^n_+ denote the nonnegative orthant of R^n and x^{T} denote the transpose of the vector $x \in R^n$. For any two vectors $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}, y = (y_1, y_2, \ldots, y_n)^{\mathrm{T}} \in R^n$, we denote:

$$
\begin{aligned}
&x = y \text{ implying} && x_i = y_i, && i = 1, 2, \ldots, n;\\
&x \prec y \text{ implying} && x_i \le y_i, && i = 1, 2, \ldots, n,\\
& && && \text{but } x \ne y;\\
&x < y \text{ implying} && x_i < y_i, && i = 1, 2, \ldots, n;\\
&x \not< y \text{ implying} && y_i < x_i && \text{for at leastone } i\,.
\end{aligned}
$$

A solution of the problem (MFP) is referred to as an efficient (Pareto optimal) solution, which is defined as follows.

Definition 6 A feasible solution $x_0 \in X$ of (MFP) is called an efficient solution of (MFP) if there exists no other feasible solution $x \in X$ such that

$$\frac{f(x)}{g(x)} \prec \frac{f(x_0)}{g(x_0)}\,.$$

In [14], Maeda gave a kind of constraint qualification, which was called generalized Guignard constraint qualification (GGCQ), under which he derived the following Kuhn–Tucker type necessary conditions for a feasible solution x_0 to be an efficient solution to the problem (MFP):

If x_0 is an efficient solution of (MFP) and (GGCQ) holds at x_0 [14], then there exist $\tau = (\tau_1, \tau_2, \ldots, \tau_p)^{\mathrm{T}} \in R^p_+, \tau > 0, \sum_{i=1}^p \tau_i = 1$ and $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^{\mathrm{T}} \in R^m_+$ such that

$$
\begin{aligned}
&\sum_{i=1}^{p} \tau_i \nabla \frac{f_i(x_0)}{g_i(x_0)} + \sum_{j=1}^{m} \lambda_j \nabla h_j(x_0) = 0,\\
&\lambda_j h_j(x_0) = 0, \qquad j = 1, 2, \ldots, m\,.
\end{aligned}
$$

This paper is organized as follows. In Sect. "Efficiency Conditions", efficiency conditions for the multiobjective fractional problem (MFP) involving (F, α, ρ, d)-convexity are presented. The duality properties of the problem (MFP) are studied in Sect. "Duality", including several duals for (MFP) and some weak and strong duality theorems. Concluding remarks are given in the last section.

Efficiency Conditions

First, we present a lemma which indicates that (F, α, ρ, d)-convexity can be preserved after taking division.

Lemma 1 *Let $X \subset R^n$ be an open set. Assume that p, q are real-valued differentiable functions defined on X and $p(x) \ge 0, q(x) > 0$ for all $x \in X$. If p and $-q$ are (F, α, ρ, d)-convex at $x_0 \in X$, then p/q is $(F, \overline{\alpha}, \overline{\rho}, \overline{d})$-convex at x_0, where $\overline{\alpha}(x, x_0) = \frac{\alpha(x, x_0) q(x_0)}{q(x)}, \overline{\rho} = \rho\left(1 + \frac{p(x_0)}{q(x_0)}\right)$ and $\overline{d}(x, x_0) = \frac{d(x, x_0)}{q^{\frac{1}{2}}(x)}$.*

In the following, we present some sufficient efficiency conditions for (MFP) under appropriate (F, α, ρ, d)-convexity assumptions.

Theorem 1 *Let x_0 be a feasible solution of (MFP). Suppose that there exist $\tau = (\tau_1, \tau_2, \ldots, \tau_p)^T \in R^p_+, \tau > 0$, $\sum_{i=1}^p \tau_i = 1$ and $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T \in R^m_+$ such that*

$$\sum_{i=1}^{p} \tau_i \nabla \frac{f_i(x_0)}{g_i(x_0)} + \sum_{j=1}^{m} \lambda_j \nabla h_j(x_0) = 0\,, \tag{4}$$

$$\lambda_j h_j(x_0) = 0, \qquad j = 1, 2, \ldots, m\,. \tag{5}$$

If f_i and $-g_i$ $(i = 1, 2, \ldots, p)$ are $(F, \alpha_i, \rho_i, d_i)$-convex at x_0, h_j $(j = 1, 2, \ldots, m)$ is $(F, \beta_j, \zeta_j, c_j)$-convex at x_0, and

$$\sum_{i=1}^{p} \tau_i \overline{\rho}_i \frac{\overline{d}_i^2(x, x_0)}{\overline{\alpha}_i(x, x_0)} + \sum_{j=1}^{m} \lambda_j \zeta_j \frac{c_j^2(x, x_0)}{\beta_j(x, x_0)} \ge 0\,, \tag{6}$$

where $\overline{\alpha}_i(x, x_0) = \frac{\alpha_i(x, x_0) g_i(x_0)}{g_i(x)}, \overline{\rho}_i = \rho_i\left(1 + \frac{f_i(x_0)}{g_i(x_0)}\right)$, and $\overline{d}_i(x, x_0) = \frac{d_i(x, x_0)}{g_i^{\frac{1}{2}}(x)}$, then x_0 is a global efficient solution for (MFP).

Corollary 1 *Let x_0 be a feasible solution of (MFP). Suppose that there exist $\tau = (\tau_1, \tau_2, \ldots, \tau_p)^T \in R^p_+$, $\tau > 0, \sum_{i=1}^p \tau_i = 1$, and $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T \in R^m_+$ such that*

$$
\begin{aligned}
&\sum_{i=1}^{p} \tau_i \nabla \frac{f_i(x_0)}{g_i(x_0)} + \sum_{j=1}^{m} \lambda_j \nabla h_j(x_0) = 0,\\
&\lambda_j h_j(x_0) = 0, \qquad j = 1, 2, \ldots, m\,.
\end{aligned}
$$

If f_i and $-g_i$ $(i = 1, 2, \ldots, p)$ are strongly $(F, \alpha_i, \rho_i, d_i)$-convex (or (F, α_i)-convex) at x_0, h_j $(j = 1, 2, \ldots, m)$ is strongly $(F, \beta_j, \zeta_j, c_j)$-convex (or (F, β_j)-convex) at x_0, then x_0 is a global efficient solution for (MFP).

For $i = 1, 2, \ldots, p$, if $g_i(x) = 1$ for all $x \in X$, $f_i(x)$ need not be nonnegative, and the functions involved are assumed to be invex, ρ-invex with respect to $\eta\colon X \times X \to R^n$, $d\colon X \times X \to R$, (F, ρ)-convex, or generalized (F, ρ)-convex, respectively, then we can obtain the corresponding results presented in [1,2,9].

Next, we consider a special case of (MFP), in which the fractional objective functions have the same denominator. For $i = 1, 2, \ldots, p$, let $g_i(x) = g(x)$ in (MFP). The property about the efficient solution of this special (MFP) can be obtained similarly as that in Theorem 1, so we state the following theorem:

Theorem 2 *Let x_0 be a feasible solution of (MFP). Suppose that there exist $\tau = (\tau_1, \tau_2, \ldots, \tau_p)^T \in R_+^p$, $\tau > 0$, $\sum_{i=1}^p \tau_i = 1$, and $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T \in R_+^m$ such that*

$$\sum_{i=1}^{p} \tau_i \nabla \frac{f_i(x_0)}{g(x_0)} + \sum_{j=1}^{m} \lambda_j \nabla h_j(x_0) = 0,$$
$$\lambda_j h_j(x_0) = 0, \quad j = 1, 2, \ldots, m.$$

If $-g$ is (F, α, ρ, d)-convex at x_0, f_i $(i = 1, 2, \ldots, p)$ is (F, α, ρ_i, d)-convex at x_0, h_j $(j = 1, 2, \ldots, m)$ is $(F, \overline{\alpha}, \zeta_j, \overline{d})$-convex at x_0, and $\sum_{i=1}^p \tau_i \overline{\rho}_i + \sum_{j=1}^m \lambda_j \zeta_j \geq 0$, where $\overline{\alpha}(x, x_0) = (\alpha(x, x_0) g(x_0))/g(x)$, $\overline{\rho}_i = \rho_i + \rho(f_i(x_0))/g(x_0)$ and $\overline{d}(x, x_0) = (d(x, x_0))/(g^{\frac{1}{2}}(x))$, then x_0 is a global efficient solution for (MFP).

Finally, we present an equivalent formulation of the problem (MFP). Let $G(x) = \prod_{i=1}^p g_i(x)$, $G_i(x) = \frac{G(x)}{g_i(x)}$ $(i = 1, 2, \ldots, p)$. Then (MFP) can be written in the following form:

$$(\overline{\text{MFP}})$$
$$\min \left(\frac{G_1(x) f_1(x)}{G(x)}, \frac{G_2(x) f_2(x)}{G(x)}, \ldots, \frac{G_p(x) f_p(x)}{G(x)} \right)^{\mathrm{T}},$$
$$\text{s.t.} \quad h(x) \leq 0, \; x \in X.$$

By Theorem 2, we have the following corollary:

Corollary 2 *Let x_0 be a feasible solution of (MFP). Suppose that there exist $\tau = (\tau_1, \tau_2, \ldots, \tau_p)^T \in R_+^p$, $\tau > 0$, $\sum_{i=1}^p \tau_i = 1$, and $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T \in R_+^m$ such that*

$$\sum_{i=1}^{p} \tau_i \nabla \frac{f_i(x_0)}{g_i(x_0)} + \sum_{j=1}^{m} \lambda_j \nabla h_j(x_0) = 0,$$
$$\lambda_j h_j(x_0) = 0, \quad j = 1, 2, \ldots, m.$$

If $-G$ is (F, α, ρ, d)-convex at x_0, $G_i f_i (i = 1, 2, \ldots, p)$ is (F, α, ρ_i, d)-convex at x_0, $h_j (j = 1, 2, \ldots, m)$ is $(F, \overline{\alpha}, \zeta_j, \overline{d})$-convex at x_0, and $\sum_{i=1}^p \tau_i \overline{\rho}_i + \sum_{j=1}^m \lambda_j \zeta_j \geq 0$, where $\overline{\rho}_i = \rho_i + \rho(f_{i(x_0)})/(g_i(x_0))$, $\overline{\alpha}(x, x_0) = (\alpha(x, x_0) G(x_0))/G(x)$, and $\overline{d}(x, x_0) = (d(x, x_0))/(G^{1/2}(x))$, then x_0 is a global efficient solution for (MFP).

Under the assumptions of Theorem 2 or Corollary 2, if $\rho \geq \max\limits_{1 \leq i \leq p} \rho_i$, $\overline{\rho}_i = \rho_i(1 + f_i(x_0)/g(x_0))$, or $\overline{\rho}_i = \rho_i(1 + f_i(x_0)/g_i(x_0))$, respectively, then the corresponding results still hold.

Duality

Many types of duals for a given mathematical programming problem. Two well-known duals are the Wolfe type dual [29] and the Mond-Weir type dual [17]. Recently, the mixed (or general type) dual has been considered for various optimization problems [1,2,11,13,18,19,20,30,31,32]. The mixed dual includes the Wolfe type dual and the Mond-Weir type dual as special cases. In the sequel, the generalized Mond-Weir dual are discussed first, and then three other types of duals are presented, which are based on (F, α, ρ, d)-convexity for the problem (MFP).

Let $M = \{1, 2, \ldots, m\}$ and $M_0, M_1, \ldots, M_q$ be a partition of M, i. e., $\bigcup_{k=1}^q M_k = M$, $M_k \bigcap M_l = \emptyset$ for $k \neq l$. The generalized Mond-Weir dual of (MFP) is as follows:

$$\max \frac{f(u)}{g(u)} + \lambda_{M_0}^{\mathrm{T}} h_{M_0}(u)\, e \triangleq \left(\frac{f_1(u)}{g_1(u)} + \lambda_{M_0}^{\mathrm{T}} h_{M_0}(u), \ldots, \frac{f_p(u)}{g_p(u)} + \lambda_{M_0}^{\mathrm{T}} h_{M_0}(u) \right)^{\mathrm{T}},$$
$$\text{s.t.} \sum_{i=1}^{p} \tau_i \nabla \frac{f_i(u)}{g_i(u)} + \sum_{j=1}^{m} \lambda_j \nabla h_j(u) = 0,$$

$$\lambda_{M_k}^{\mathrm{T}} h_{M_k}(u) \geq 0, k = 1, 2, \ldots, q,$$
$$\tau = (\tau_1, \tau_2, \ldots, \tau_p)^{\mathrm{T}} \in R_+^p, \tau > 0,$$
$$\sum_{i=1}^{p} \tau_i = 1,$$
$$\lambda_{M_k} \in R_+^{|M_k|}, k = 0, 1, 2, \ldots, q,$$
$$u \in X,$$

where $e = (1, 1, \ldots, 1)^{\mathrm{T}}$ and λ_{M_k} denotes the column vector whose subscripts of components belong to M_k. In particular, if $M_0 = M, M_k = \emptyset, k = 1, 2, \ldots, q$, then the above dual becomes the Wolfe type dual; if $M_0 = \emptyset$ and $q = 1, M_1 = M$, the Mond-Weir type dual is obtained. Since the Wolfe type dual is unsuitable for single objective fractional programming problems [15,22,23], the duals with $M_0 \neq \emptyset$ are certainly unsuitable for (MFP). For the generalized Mond-Weir type dual, we only consider the case $M_0 = \emptyset, M_1 = M$, i. e., the Mond-Weir dual.

Mond-Weir Dual

The Mond-Weir dual of the problem (MFP) has the following form:

(MFD1)

$$\max \frac{f(u)}{g(u)} = \left(\frac{f_1(u)}{g_1(u)}, \frac{f_2(u)}{g_2(u)}, \ldots, \frac{f_p(u)}{g_p(u)}\right)^{\mathrm{T}}$$
$$\text{s.t.} \sum_{i=1}^{p} \tau_i \nabla \frac{f_i(u)}{g_i(u)} + \sum_{j=1}^{m} \lambda_j \nabla h_j(u) = 0,$$
$$\lambda^{\mathrm{T}} h(u) \geq 0,$$
$$\tau = (\tau_1, \tau_2, \ldots, \tau_p)^{\mathrm{T}} \in R^p, \tau > 0, \sum_{i=1}^{p} \tau_i = 1,$$
$$\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^{\mathrm{T}} \in R_+^m, u \in X.$$

Theorem 3 *(Weak Duality) Assume that $\overline{x}$ is a feasible solution of (MFP) and $(\overline{u}, \overline{\tau}, \overline{\lambda})$ is a feasible solution of (MFD1). If f_i and $-g_i$ $(i = 1, 2, \ldots, p)$ are $(F, \alpha_i, \rho_i, d_i)$-convex at $\overline{u}$, h_j $(j = 1, 2, \ldots, m)$ is (F, β, ζ_j, c_j)-convex at $\overline{u}$, and the inequality*

$$\sum_{i=1}^{p} \overline{\tau}_i \overline{\rho}_i \frac{\overline{d}_i^2(\overline{x}, \overline{u})}{\overline{\alpha}_i(\overline{x}, \overline{u})} + \sum_{j=1}^{m} \overline{\lambda}_j \zeta_j \frac{c_j^2(\overline{x}, \overline{u})}{\beta(\overline{x}, \overline{u})} \geq 0 \qquad (7)$$

holds, where $\overline{\alpha}_i(\overline{x}, \overline{u}) = \alpha_i(\overline{x}, \overline{u})(g(\overline{u}))/(g(\overline{x}))$, $\overline{\rho}_i = \rho_i(1 + (f_i(\overline{u}))/(g_i(\overline{u})))$, and $\overline{d}_i(\overline{x}, \overline{u}) = (d_i(\overline{x}, \overline{u}))/(g_i^{\frac{1}{2}}(\overline{x}))$, then we have

$$\frac{f(\overline{x})}{g(\overline{x})} \not< \frac{f(\overline{u})}{g(\overline{u})}.$$

Corollary 3 *(Weak Duality) Assume that $\overline{x}$ is a feasible solution of (MFP), and $(\overline{u}, \overline{\tau}, \overline{\lambda})$ is a feasible solution of (MFD1). If f_i and $-g_i$ $(i = 1, 2, \ldots, p)$ are strongly $(F, \alpha_i, \rho_i, d_i)$-convex (or (F, α_i)-convex) at $\overline{u}$, and h_j $(j = 1, 2, \ldots, m)$ is strongly (F, β, ζ_j, c_j)-convex (or (F, β)-convex) at $\overline{u}$, then*

$$\frac{f(\overline{x})}{g(\overline{x})} \not< \frac{f(\overline{u})}{g(\overline{u})}.$$

Theorem 4 *(Strong Duality) Assume that $\overline{x}$ is an efficient solution of (MFP) and the constraint qualification (GGCQ) holds at $\overline{x}$ [14]. Then there exists $(\overline{\tau}, \overline{\lambda}) \in R_+^p \times R_+^m$ such that $(\overline{x}, \overline{\tau}, \overline{\lambda})$ is a feasible solution of (MFD1), and the objective function values of (MFP) and (MFD1) at the corresponding points are equal. If the assumptions about the generalized convexity and the inequality (7) in Theorem 3 are also satisfied, then $(\overline{x}, \overline{\tau}, \overline{\lambda})$ is an efficient solution of (MFD1).*

Schaible Dual

In this subsection, we shall consider the following extended form of the Schaible dual for (MFP) [22,23]:

(MFD2)

$$\max \lambda = (\lambda_1, \lambda_2, \ldots, \lambda_p)^{\mathrm{T}}$$
$$\text{s.t.} \sum_{i=1}^{p} \tau_i \nabla_u (f_i(u) - \lambda_i g_i(u)) + \sum_{j=1}^{m} \nu_j \nabla h_j(u) = 0,$$
$$f_i(u) - \lambda_i g_i(u) \geq 0, \ i = 1, 2, \ldots, p$$
$$\nu^{\mathrm{T}} h(u) \geq 0,$$
$$\tau > 0, \quad \sum_{i=1}^{p} \tau_i = 1,$$
$$\lambda \in R_+^p, \quad \tau \in R_+^p, \quad \nu \in R_+^m, \quad u \in X.$$

Theorem 5 *(Weak Duality). Assume that $\overline{x}$ is a feasible solution of (MFP) and $(\overline{u}, \overline{\tau}, \overline{\lambda}, \overline{\nu})$ is a feasible solution of (MFD2). If one of the following holds:*

- *(I) f_i and $-g_i$ $(i = 1, 2, \ldots, p)$ are $(F, \alpha_i, \rho_i, d_i)$-convex at $\overline{u}$, h_j $(j = 1, 2, \ldots, m)$ is (F, β, ζ_j, c_j)-convex at $\overline{u}$, and*

$$\sum_{i=1}^{p} \overline{\tau}_i \rho_i (1 + \overline{\lambda}_i) \frac{d_i^2(\overline{x}, \overline{u})}{\alpha_i(\overline{x}, \overline{u})} + \sum_{j=1}^{m} \overline{v}_j \zeta_j \frac{c_j^2(\overline{x}, \overline{u})}{\beta(\overline{x}, \overline{u})} \geq 0; \tag{8}$$

- *(II) f_i and $-g_i$ $(i = 1, 2, \ldots, p)$ are (F, α, ρ_i, d)-convex at $\overline{u}$, h_j $(j = 1, 2, \ldots, m)$ is (F, α, ζ_j, d)-convex at $\overline{u}$, and the vectors $\overline{\tau}, \overline{\lambda}, \overline{v}$ satisfy:*

$$\sum_{i=1}^{p} \overline{\tau}_i \rho_i (1 + \overline{\lambda}_i) + \sum_{j=1}^{m} \overline{v}_j \zeta_j \geq 0\,, \tag{9}$$

then

$$\frac{f(\overline{x})}{g(\overline{x})} \not< \overline{\lambda}.$$

Theorem 6 *(Strong Duality). Assume $\overline{x}$ is an efficient solution of (MFP), and the constraint qualification (GGCQ) holds at $\overline{x}$ [14]. Then there exist $\overline{\tau} \in R_+^p$, $\overline{\lambda} \in R_+^p$, $\overline{v} \in R_+^m$ such that $(\overline{x}, \overline{\tau}, \overline{\lambda}, \overline{v})$ is a feasible solution of (MFD2) and $\overline{\lambda} = \frac{f(\overline{x})}{g(\overline{x})}$. Furthermore, if all assumptions in Theorem 5 are satisfied, then the corresponding $(\overline{x}, \overline{\tau}, \overline{\lambda}, \overline{v})$ is an efficient solution of (MFD2).*

Extended Bector Type Dual

For a single-objective fractional programming problem in [4], Bector used the positivity of the denominator to transform the inequality constraints and add them to the objective by Lagrangian mulitipliers for establishing a kind of dual. Since the denominators in (MFP) need not be the same, we use the equivalent form $(\overline{MFP})$ of (MFP) to establish the following dual, which is called the extended Bector type dual of (MFP):

(MFD3)

$$\max \begin{pmatrix} \frac{G_1(u) f_1(u) + v_{M_0}^{\mathrm{T}} h_{M_0}(u)}{G(u)} \\ \vdots \\ \frac{G_p(u) f_p(u) + v_{M_0}^{\mathrm{T}} h_{M_0}(u)}{G(u)} \end{pmatrix}^{\mathrm{T}}$$

$$\text{s.t.} \sum_{i=1}^{p} \tau_i \nabla_u \frac{G_i(u) f_i(u) + v_{M_0}^{\mathrm{T}} h_{M_0}(u)}{G(u)}$$

$$+ \sum_{k=1}^{q} \nabla_u v_{M_k}^{\mathrm{T}} h_{M_k}(u) = 0,$$

$$v_{M_k}^{\mathrm{T}} h_{M_k}(u) \geq 0, \quad k = 1, 2, \ldots, q,$$

$$G_i(u) f_i(u) + v_{M_0}^{\mathrm{T}} h_{M_0}(u) \geq 0,$$

$$i = 1, 2, \ldots, p,$$

$$\sum_{i=1}^{p} \tau_i = 1, \tau = (\tau_1, \tau_2, \ldots, \tau_p)^{\mathrm{T}} \in R_+^p,$$

$$\tau > 0,$$

$$u \in X, \quad v_{M_k} \in R_+^{|M_k|}, \quad k = 0, 1, 2, \ldots, q\,.$$

Theorem 7 *(Weak Duality) Let x be a feasible solution of (MFP) and (u, τ, v) be a feasible solution of (MFD3). Assume that $-G$ is (F, α, ρ, d)-convex at u, $G_i f_i$ $(i = 1, \ldots, p)$ is (F, α, ρ_i, d)-convex at u and h_j $(j = 1, \ldots, m)$ is (F, α, ζ_j, d)-convex at u. If $\rho \geq \max_{1 \leq i \leq p} \rho_i$ and the following inequality holds:*

$$\sum_{i=1}^{p} \tau_i \rho_i \left(1 + \frac{G_i(u) f_i(u) + v_{M_0}^T h_{M_0}(u)}{G(u)}\right) + \sum_{j \in M_0} v_j \zeta_j + G(u) \sum_{k=1}^{q} \sum_{j \in M_k} v_j \zeta_j \geq 0\,, \tag{10}$$

then we have

$$\frac{f(x)}{g(x)} \not< \frac{\overline{G}(u) f(u) + v_{M_0}^T h_{M_0}(u)\, e}{G(u)}\,,$$

where $\overline{G}(u) = diag\{G_1(u), \ldots, G_p(u)\}$ and each component in $e \in R^p$ is equal to 1.

Theorem 8 *(Strong Duality) Assume that $\overline{x}$ is an efficient solution of (MFP) and the constraint qualification (GGCQ) holds at $\overline{x}$ [14]. Then there exists $(\overline{\tau}, \overline{v})$ such that $(\overline{x}, \overline{\tau}, \overline{v})$ is a feasible solution of (MFD3), and the objective function values of (MFP) and (MFD3) at $\overline{x}$ and $(\overline{x}, \overline{\tau}, \overline{v})$, respectively, are equal. If the assumptions and conditions in Theorem 7 are also satisfied, then $(\overline{x}, \overline{\tau}, \overline{v})$ is an efficient solution of (MFD3).*

Concluding Remarks

In this paper, a unified formulation of the generalized convexity defined in [12] is adopted, which includes many other generalized convexity concepts in optimization theory as special cases. Our concept of generalized convexity is suitable to analyze the efficiency

conditions and duality of multiobjective fractional programming problems. Efficiency conditions and duality for a class of multiobjective fractional programming problems are presented. We extend the methods, which were adopted for single-objective fractional programming problems in [10,12,21], to the case with multiple fractional objectives. We also present the extended Bector type dual by using an equivalent formulation of the primal problem. Note that we only consider (MFP) from a viewpoint of the efficient solution in this paper. The methods used here can be extended to the study of (MFP) from a viewpoint of the weak efficient solution.

References

1. Aghezzaf B, Hachimi M (2000) Generalized convexity and duality in multiobjective programming problems. J Glob Optim 18:91–101
2. Aghezzaf B, Hachimi M (2001) Sufficiency and duality in multiobjective programming involving generalized (F, ρ)-convexity. J Math Anal Appl 258:617–628
3. Avriel M, Diewert WE, Schaible S, Zang I (1988) Generalized Concaveity. Plenum Press, New York
4. Bector CR (1973) Duality in nonlinear fractional programming. Z Oper Res 17:183–193
5. Craven BD (1988) Fractional Programming. Heldermann, Berlin
6. Hanson MA (1981) On sufficiency of the Kuhn–Tucker conditions. J Math Anal Appl 80:544–550
7. Jeyakumar V (1985) Strong and weak invexity in mathematical programming. Methods Oper Res 55:109–125
8. Jeyakumar V, Mond B (1992) On generalized convex mathematical programming. J Aust Math Soc Series B 34:43–53
9. Kaul RN, Suneja SK, Srivastava MK (1994) Optimality criteria and duality in multiple-objective optimization involving generalized invexity. J Optim Theor Appl 80(3):465–482
10. Khan Z, Hanson MA (1997) On ratio invexity in mathematical programming. J Math Anal Appl 205:330–336
11. Li Z (1993) Duality theorems for a class of generalized convex multiobjective programming problems. Acta Scientiarum Naturalium Universitatis NeiMongol 24(2):113–118
12. Liang ZA, Huang HX, Pardalos PM (2001) Optimality conditions and duality for a class of nonlinear fractional programming problems. J Optim Theor Appl 110(3):611–619
13. Liang Z, Ye Q (2001) Duality for a class of multiobjective control problems with generalized invexity. J Math Anal Appl 256:446–461
14. Maeda T (1994) Constraint qualifications in multiobjective optimization problems: differentiable case. J Optim Theor Appl 80(3):483–500
15. Mangasarian OL (1969) Nonlinear Programming. McGraw-Hill, New York
16. Miettinen KM (1999) Nonlinear Multiobjective Optimization. Kluwer, Dordrecht
17. Mond B, Weir T (1981) Generalized concavity and daulity. In: Schaible S, Ziemba WT (eds) Generalized Convexity in Optimization and Economics. Academic Press, New York, pp 263–280
18. Mond B, Weir T (1982) Duality for fractional programming with generalized convexity conditions. J Inf Optim Sci 3(2):105–124
19. Mukherjee RN, Rao CP (2000) Mixed type duality for multiobjective variational problems. J Math Anal Appl 252:571–586
20. Preda V (1992) On efficiency and duality for multiobjective programs. J Math Anal Appl 166:365–377
21. Reddy LV, Mukherjee RN (1999) Some results on mathematical programming with generalized ratio invexity. J Math Anal Appl 240:299–310
22. Schaible S (1976) Duality in fractional proramming: a unified approach. Oper Res 24:452–461
23. Schaible S (1976) Fractional programming, I. Duality Manag Sci 22:858–867
24. Schaible S (1981) Fractional programming: applications and algorithms. Eur J Oper Res 7:111–120
25. Schaible S (1995) Fractional Programming. In: Horst R, Pardalos PM (eds) Handbook of Global Optimization. Kluwer, Dordrecht, pp 495–608
26. Schaible S, Ibaraki T (1983) Fractional programming. Eur J Oper Res 12:325–338
27. Stancu-Minasian IM (1997) Fractional programming: theory, methods and applications. Kluwer, Dordrecht
28. Stancu-Minasian IM (1999) A fifth bibliograpy of fractional programming. Optimization 45:343–367
29. Wolfe P (1961) A duality theorem for nonlinear programming. Q Appl Math 19:239–244
30. Xu Z (1996) Mixed type duality in multiobjective programming problems. J Math Anal Appl 198:621–635
31. Yang XM, Teo KL, Yang XQ (2000) Duality for a class of nondifferentiable multiobjective programming problems. J Math Anal Appl 252:999–1005
32. Zhang Z, Mond B (1997) Duality for a nondifferentiable programming problem. Bull Aust Math Soc 55:29–44

Multi-objective Integer Linear Programming

MOILP

JACQUES TEGHEM
Lab. Math. & Operational Research Fac.,
Polytechn. Mons, Mons, Belgium

MSC2000: 90C29, 90C10

Article Outline

Keywords
Generation of E(P)
Klein–Hannan Method
Kiziltan–Yucaoglu Method
Interactive Methods
Gonzalez–Reeves–Franz Algorithm
Steuer–Choo Method
The MOMIX Method
See also
References

Keywords

Multi-objective programming; Integer; Linear programming

From the 1970s onwards, *multi-objective linear programming* (MOLP) methods with continuous solutions have been developed [8]. However, it is well known that discrete variables are unavoidable in the linear programming modeling of many applications, for instance, to represent an investment choice, a production level, etc.

The mathematical structure is then *integer linear programming* (ILP), associated with MOLP giving a MOILP problem. Unfortunately, MOILP cannot be solved by simply combining ILP and MOLP methods, because it has got its own specific difficulties.

The problem (P) considered is defined as

$$(\mathrm{P})\quad \begin{cases} \text{'}\max_{X\in D}\text{'} & z_k(X)=\sum_{j=1}^{n} c_j^{(k)} x_j, \\ & k=1,\ldots,K, \\ \text{where} & D=\left\{ X\in\mathbb{R}^n : \begin{array}{c} TX\le d, \\ X\ge 0, \\ x_j \text{ integer}, \\ j\in J \end{array} \right\} \\ \text{with} & T(m\times n), \\ & d(m\times 1), \\ & X(n\times 1), \\ & J\subset\{1,\ldots,n\}. \end{cases}$$

If we denote $LD=\{X\colon TX\le d, X\ge 0\}$, problem (LP) is the linear relaxation of problem (P):

$$(\mathrm{LP})\quad \begin{cases} \text{'}\max\text{'} & z_k(X), \quad k=1,\ldots,K, \\ & X\in LD \end{cases}$$

A solution $X^\star$ in D (or LD) is said to be *efficient* for problem (P) (or (LP)) if there does not exist any other solution in D (or LD) such that $z_k(X)\ge z_k(X^\star)$, $k=1,\ldots,K$, with at least one strict inequality.

Let $E(\cdot)$ denote the set of all efficient solutions of problem $(\cdot)$. It is well known (see [8]) that (LP) may be characterized by the optimal solutions of the single objective and parametrized problem:

$$(\mathrm{LP}_\lambda)\quad \begin{cases} \max & \sum_{k=1}^{K}\lambda_k z_k(X) \\ & X\in LD \\ \text{with} & \lambda_k>0, \quad \forall k, \\ & \sum_{k=1}^{K}\lambda_k=1 \end{cases}$$

This fundamental principle – often called *Geoffrion's theorem* – is no longer valid in presence of discrete variables because the set D is not convex. The set of optimal solutions of problem (P_λ), defined as problem (LP_λ) in which LD is replaced by D, is only a subset SE(P) of E(P); the solutions in SE(P) are called *supported efficient solutions*, while the solutions belonging to $NSE(\mathrm{P}) = E(\mathrm{P}) \setminus SE(\mathrm{P})$ are called *nonsupported efficient solutions*.

The breakdown of Geoffrion's theorem for problem (P) can be illustrated by the following obvious example:

$$\begin{aligned} &K=2, \\ &z_1(X)=6x_1+3x_2+x_3, \\ &z_2(X)=x_1+3x_2+6x_3, \\ &D=\{X\colon\ x_1+x_2+x_3\le 1,\ x_i\in\{0,1\}\}. \end{aligned}$$

For this problem,

$$E(\mathrm{P})=\{(1,0,0);(0,1,0);(0,0,1)\}$$

while $NSE(\mathrm{P})=\{(0,1,0)\}$.

Nevertheless, V.J. Bowman [1] has given a theoretical characterization of E(P): Setting

$$M_k = \max_{X \in D} z_k(X),$$
$$\overline{z}_k = M_k + \varepsilon_k, \quad \text{with } \varepsilon_k > 0,$$
$$\rho > 0,$$

then E(P) is characterized by the optimal solutions of the problem(P_λ^T):

$$\min_{X \in D} \max_k \left(\lambda_k \left(\overline{z}_k - z_k(X) \right) + \rho \left(\sum_{k=1}^{K} \left(\overline{z}_k - z_k(X) \right) \right) \right),$$

consisting of minimizing the augmented weighted Tchebychev distance between $z_k(X)$ and $\overline{z}_k$.

Let us note that another characterization of E(P) is given in [2] for the particular case of binary variables.

Two types of problems can be analysed:

- Generate E(P) explicitly. Several methods have been proposed; they are reviewed in [10]. below we will present two of them, which appear general, characteristic and efficient.
- To determine interactively with the decision maker a 'best compromise' in E(P) according to the preferences of the decision maker. Some of the existing approaches are reviewed in [11]; below we will describe three of these interactive methods.

Generation of *E*(P)

Klein–Hannan Method

See [5]. This is an iterative procedure for sequentially generating the complete set of efficient solutions for problem (P) (we suppose that the coefficients $c_j^{(k)}$ are integers); it consists in solving a sequence of progressively more constrained single objective ILP problems and can be implemented through use of any ILP algorithm.

- (Initialization: step 0) An objective function $l \in \{1, \ldots, K\}$ is chosen arbitrarily and the following single objective ILP problem is considered:

  $$(\mathrm{P}_0) \quad \max_{X \in D} z_l(X).$$

  Let $E(\mathrm{P}_0)$ be the set of all optimal solutions of (P_0) and let E_0(P) be the set of solutions defined as $E_0(\mathrm{P}) = E(\mathrm{P}_0) \cap E(\mathrm{P})$. Thus, E_0(P) is the subset of non-dominated solutions in $E(\mathrm{P}_0)$.
- (Step j, ($j \geq 1$)) The efficient solutions generated at the previous steps are denoted by X_r^*, $r = 1, \ldots, R$, i. e. $\cup_{i=1}^{j-1} E_i(P) = \{X_r^* : r = 1, \ldots, R\}$. In this jth step, the following problem is solved

  $$(\mathrm{P}_j) \quad \begin{cases} \max\limits_{X \in D} z_l(X) \\ \bigcap\limits_{r=1}^{R} \left(\bigcup\limits_{\substack{k=1 \\ k \neq l}}^{K} z_k(X) \geq z_k(X_r^*) + 1 \right). \end{cases}$$

  The new set of constraints represents the requirement that a solution to (P_j) be better on some objective $k \neq l$ for each efficient solution X_r^* generated during the previous steps; an example of implementation of theseconstraints is given in [5]. The set of solutions E_j(P) is then defined as $E_j(\mathrm{P}) = E(\mathrm{P}_j) \cap E(\mathrm{P})$, where $E(\mathrm{P}_j)$ is the set of all optimal solutions of (P_j).

The procedure continues until, at some iteration J, the problem (P_J) becomes infeasible; at this time $E(\mathrm{P}) = \cup_{j=0}^{J-1} E_j(\mathrm{P})$.

Kiziltan–Yucaoglu Method

See [4]. This is a direct adaptation to a multi-objective framework of the well-known *Balas algorithm* for the ILP problem with binary variables.

At node S^r of the *branch and bound* scheme, the following problem is considered:

$$\begin{cases} \text{'max'} & \sum\limits_{j \in F^r} c_j x_j + \sum\limits_{j \in B^r} c_j \\ \text{s.t.} & \sum\limits_{j \in F} t_j x_j \leq d^r \\ & x_j = (0, 1) \\ \text{where} & B^r \text{ is the index set of variables} \\ & \quad \text{assigned the value one} \\ & F^r \text{ is the index of free variables} \\ & d^r = d - \sum\limits_{j \in B^r} t_j \\ & t_j \text{ is the } j\text{th column of } T \\ & c_j \text{ is the vector of components } c_j^{(k)}. \end{cases}$$

The node S^r is called *feasible* when $d^r \geq 0$ and *infeasible* otherwise. The three *basic rules of the branch and bound* algorithm are:

- (bounding rule) A lower and upper bound vector, $\underline{Z}^r$ and $\overline{Z}^r$, respectively, are defined as

$$\underline{Z}^r = \sum_{j \in B^r} c_j,$$

$$\overline{Z}^r = \underline{Z}^r + Y^r,$$

where $Y^r_k = \sum_{j \in F^r} \max\{0, c^k_j\}$. The vector $\underline{Z}^r$ is added to a list $\widehat{E}$ of existing lower bounds if $\underline{Z}^r$ is not dominated by any of the existing vectors of $\widehat{E}$. At the same time, any vector of $\widehat{E}$ dominated by $\underline{Z}^r$ is discarded.
- (fathoming rules) In the multi-objective case, the feasibility of a node is no longer a sufficient condition for fathoming it. The three general fathoming conditions are:
  - $\overline{Z}^r$ is dominated by some vector of $\widehat{E}$;
  - the node S^r is feasible and $\underline{Z}^r = \overline{Z}^r$;
  - the node S^r is unfeasible and $\sum_{j \in F^r} \min(0, t_{ij}) > d^r_i$ for some $i = 1, \ldots, m$.

  The usual backtracking rules are applied.
- (branching rule) A variable $x_l \in F^r$ is selected to be the branching variable.
  - If the node S^r is feasible, $l \in \{j \in F^r : c_j \not\leq 0\}$.
  - Otherwise, index l is selected by the *minimum unfeasibility criterion*:

$$\min_{j \in F^r} \sum_{i=1}^{m} \max\left(0, -d^r_i + t_{ij}\right).$$

When the explicit enumeration is complete, $E(\mathrm{P}) = \widehat{E}$.

Interactive Methods

Such methods are particularly important to solve multi-objective applications. The general idea is to determine progressively a good compromise solution integrating the preferences of the decision maker.

The dialog with the decision maker consist of a succession of 'calculation phase' managed by the model and 'information phase' managed by the decision maker.

At each calculation phase, one or several new efficient solutions are determined taking into account the information given by the decision maker at the preceding information phase. At each information phase, a few number of easy questions are asked to the decision maker to collect information about its preferences in regard to the new solutions.

Gonzalez–Reeves–Franz Algorithm

See [3]. In this method a set $\widetilde{E}$ of K efficient solutions is selected and updated in each algorithm step according to the decision maker's preferences. At the end of the procedure, $\widetilde{E}$ will contain the most preferred solutions. The method is divided in two stages: in the first one, the supported efficient solutions are considered, while the second one deals with nonsupported efficient solutions.

- (Stage 1): Determination of the best supported efficient solutions. $\widetilde{E}$ is initialized with K optimal solutions of the K single objective ILP problems. Let us denote by $\widetilde{Z}$ the K corresponding points in the objective space of the solution of $\widetilde{E}$. At each iteration, a linear direction of search $G(X)$ is build:$G(X)$ is the inverse mapping of the hyperplane defined by the points of $\widetilde{Z}$ in the objective space into the decision space. A new supported efficient solution X^* is determined by solving the single objective ILP problem $\max_{X \in D} G(X)$ and Z^* is the corresponding point in the objective space. Then:
  - if $Z^* \notin \widetilde{Z}$ and the decision maker prefers solution X^* to at least one solution of $\widetilde{E}$: the least preferred solution is replaced in $\widetilde{E}$ by X^* and a new iteration is performed;
  - if $Z^* \notin \widetilde{Z}$ and X^* is not preferred to any solution in $\widetilde{E}$: $\widetilde{E}$ is not modified and the second stage is initiated;
  - if $Z * \widetilde{Z}$: $\widetilde{Z}$ defines a face of the efficient surface and the second stage is initiated.
- (Stage 2): Introduction of the best non supported solutions. We will not give details about this second stage (see [3] or [10]); letus just say that it is performed in the same spirit but considering the single objective problem

$$\begin{cases} \max & G(X) \\ & X \in D \\ & G(X) \leq \widetilde{G} - \varepsilon \quad \text{with } \varepsilon > 0 \end{cases}$$

where $\widetilde{G}$ is the optimal value obtained for the last function $G(X)$ considered.

Steuer–Choo Method

See [9]. Several interactive approaches of MOLP problems can also be applied to MOILP; among them, we mention only the Steuer–Choo method, which is a very

general procedure based on problem (P_λ^T) defined in the introduction.

The first iteration uses a widely dispersed group of λ weighting vectors to sample the set of efficient solutions. The sample is obtained by solving problem (P_λ^T) for each of the λ values in the set. Then the decision maker is asked to identify the most preferred solution $X^{(1)}$ among the sample. At iteration j, a more refined grid of weighting vectors λ is used to sample the set of efficient solution in the neighborhood of the point $z_k(X^{(j)})$ $(k = 1, \ldots, K)$ in the objective space. Again the sample is obtained by solving several problems (P_λ^T) and the most preferred solution $X^{(j+1)}$ is selected. The procedure continues using increasingly finer sampling until the solution is deemed to be acceptable.

The MOMIX Method

(See [6].) The main characteristic of this method is the use of an interactive branch and bound concept – initially introduced in [7] – to design the interactive phase.

- (First compromise): The following minimax optimization, with $m = 1$, is performed to determined the compromise $\widetilde{X}^{(1)}$:

$$(P^m) \quad \begin{cases} \min & \delta \\ \forall k & \Pi_k^{(m)}(M_k^{(m)} - z_k(X)) \le \delta, \\ & X \in D^{(m)} \end{cases}$$

  where
  - $D^{(1)} \equiv D$;
  - $[m_k^{(1)}, M_k^{(1)}]$ are the variation intervals of the criteria k, provided by the pay-off table (see [8]);
  - $\Pi_k^{(1)}$ are certain normalizing weights taking into account these variation intervals (see [8]).

Remark 1 If the optimal solution is not unique, an augmented weighted Tchebychev distance is required in order to obtain an efficient first solution.

- (Interactive phases): There are integrated in an interactive branch and bound tree; a first step (a *depth-first* progression in the tree) leads to the determination of a first good compromise; the second step (a *backtracking* procedure) confirms the degree of satisfaction achieved by the decision maker or it finds a better compromise if necessary.
  - (Depth first progression): For $m \ge 1$, let at the mth iteration
    1) $\widetilde{X}^{(m)}$ be the mth compromise;
    2) $z_k^{(m)}$ be the corresponding values of the criteria;
    3) $[m_k^{(m)}, M_k^{(m)}]$ be the variation intervals of the criteria; and
    4) $\Pi_k^{(m)}$ be the weight of the criteria.

    The decision maker has to choose, at this mth iteration, the criterion $l_m(1) \in \{k: k = 1, \ldots, K\}$ he is willing to improve in priority. Then a new constraint is introduced so that the feasible set becomes $D^{(m+1)} \equiv D^{(m)} \cap \{z_{lm}(1)(X) > z_{lm}(1)^{(m)}\}$ Further, the variation intervals $[m_k^{(m+1)}, M_k^{(m+1)}]$ and the weights $\Pi_k^{(m+1)}$ are updated on the new feasible set $D^{(m+1)}$. The new compromise $\widetilde{X}^{(m+1)}$ is obtained by solving the problem (P^{m+1}).

    Different tests allow to terminate this first step. The node $(m+1)$ is fathomed if one of the following conditions is verified:
    a) $D^{(m+1)} = \emptyset$;
    b) $M_k^{(m+1)} - m_k^{(m+1)} \le \epsilon_k \ \forall\, k$;
    c) the vector $\widehat{Z}$ of the *incumbent values* (values of the criteria for the best compromise already determined) is preferred to the new ideal point (of component $M_k^{(m+1)}$).

    The first step of the procedure is stopped if either more than q successive iterations do not bring an improvement of the incumbent point $\widehat{Z}$ or more than Q iterations have been performed.

    Note that the parameters ϵ_k, q and Q are fixed in the agreement with the decision maker.

c) (Backtracking procedure): It can be hoped that the appropriate choice of the criterion $z_{lm}(1)$, at each level m of the depth-first progression, has been made so that at the end of the first step, a good compromise has been found.

Nevertheless, it is worth examining some other parts of the tree to confirm the satisfaction of the decision maker. The complete tree is generated in the following manner: at each level, K subnodes are introduced by successively adding the constraints:

$$\begin{array}{llllll} z_{l_m(1)}(X) > z_{l_m(1)}^{(m)}, & & & & & \\ z_{l_m(2)}(X) > z_{l_m(2)}^{(m)}; & z_{l_m(1)}(X) \le z_{l_m(1)}^{(m)}, & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ z_{l_m(K)}(X) > z_{l_m(K)}^{(m)}; & z_{l_m(k)}(X) \le z_{l_m(k)}^{(m)}, & & & & \end{array}$$

for all $k = 1, \ldots, K-1$, where $l_m(k) \in \{k\colon k = 1, \ldots, K\}$ is the kth objective that the decision maker wants to improve at the mth level of the branch and bound tree.

At each level m, the criteria are thus ordered according to the priorities of the decision maker in regard with the compromise $\widetilde{X}^{(m)}$.

The usual backtracking procedure is applied; yet it seems unnecessary to explore the whole tree. Indeed, the subnode $k > \overline{K}$ of each branching correspond to a simultaneous relaxation of those criteria $l_m(k)$, $k \leq \overline{K}$, the decision maker wants to improve in priority!

Therefore, the subnodes $k > \overline{K} = 2$ or 3, for instance, do almost certainly not bring any improved solutions.

The fathoming tests and the stopping tests are again applied in this second step.

See also

- Bi-objective Assignment Problem
- Branch and Price: Integer Programming with Column Generation
- Decision Support Systems with Multiple Criteria
- Decomposition Techniques for MILP: Lagrangian Relaxation
- Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- Financial Applications of Multicriteria Analysis
- Fuzzy Multi-objective Linear Programming
- Integer Linear Complementary Problem
- Integer Programming
- Integer Programming: Algebraic Methods
- Integer Programming: Branch and Bound Methods
- Integer Programming: Branch and Cut Algorithms
- Integer Programming: Cutting Plane Algorithms
- Integer Programming Duality
- Integer Programming: Lagrangian Relaxation
- LCP: Pardalos–Rosen Mixed Integer Formulation
- Mixed Integer Classification Problems
- Multicriteria Sorting Methods
- Multi-objective Combinatorial Optimization
- Multi-objective Mixed Integer Programming
- Multi-objective Optimization and Decision Support Systems
- Multi-objective Optimization: Interaction of Design and Control
- Multi-objective Optimization: Interactive Methods for Preference Value Functions
- Multi-objective Optimization: Lagrange Duality
- Multi-objective Optimization: Pareto Optimal Solutions, Properties
- Multiparametric Mixed Integer Linear Programming
- Multiple Objective Programming Support
- Outranking Methods
- Parametric Mixed Integer Nonlinear Optimization
- Portfolio Selection and Multicriteria Analysis
- Preference Disaggregation
- Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- Preference Modeling
- Set Covering, Packing and Partitioning Problems
- Simplicial Pivoting Algorithms for Integer Programming
- Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- Stochastic Integer Programs
- Time-dependent Traveling Salesman Problem

References

1. Bowman VJ Jr (1976) On the relationship of the Tchebycheff norm of the efficient frontier of multi-criteria objectives. In: Thiriez H, Zionts S (eds) Multiple Criteria Decision Making. Springer, Berlin, pp 76–85
2. Burkard RE (1981) A relationship between optimality and efficiency in multiple criteria 0–1 programming problems. Comput Oper Res 8:241–247
3. Gonzalez JJ, Reeves GR, Franz LS (1985) An interactive procedure for solving multiple objective integer Linear programming problems. In: Haimes Y, Chankong V (eds) Decision Making with Multiple Objectives. Springer, Berlin, pp 250–260
4. Kiziltan G, Yucaoglu E (1983) An algorithm for multiobjective zero-one linear programming. Managem Sci 29(12):1444–1453
5. Klein D, Hannan E (1982) An algorithm for the multiple objective integer linear programming. EJOR 9(4):378–385
6. L'Hoir H, Teghem J (1995) Portfolio selection by MOLP using an interactive branch and bound. Found Computing and Decision Sci 20(3):175–185
7. Marcotte O, Soland R (1986) An interactive branch and bound algorithm for multiple criteria optimization. Managem Sci 32(1):61–75

8. Steuer RE (1986) Multiple criteria optimization theory, computation and applications. Wiley, New York
9. Steuer RE, Choo E-U (1983) An interactive method weighted Tchebycheff procedure for multiple objective programming. Math Program 26:326–344
10. Teghem J, Kunsch P (1986) Interactive method for multi-objective integer linear programming. In: Fandel G, et al. (eds) Large Scale Modelling and Interactive decision analysis. Springer, Berlin 75–87
11. Teghem J, Kunsch P (1986) A survey of techniques for finding efficient solutions to multi-objective integer linear programming. Asia–Pacific J Oper Res 3:1195–106

Multi-objective Mixed Integer Programming

MARIA JOÃO ALVES, JOÃO CLÍMACO
Fac. Economics, University Coimbra and INESC, Coimbra, Portugal

MSC2000: 90C29, 90C11

Article Outline

Keywords
Efficiency and Nondominance
Supported and Unsupported Nondominated Solutions
Characterization of the Nondominated Set
Interactive Versus Noninteractive Methods
Computing Processes and Their Use in Interactive Methods
Weighted-Sums Programs with Additional Constraints
Tchebycheff and Achievement Scalarizing Programs
Conclusions and Future Developments
See also
References

Keywords

Multi-objective mathematical programming; Multicriteria analysis; Interactive method

A *multi-objective (multicriteria) mixed integer programming*(*MOMIP*) problem is a mathematical programming problem that considers more than one objective function and some but not all the variables are constrained to be integer valued. The integer variables can either be binary or take on general integer values. The problem may be stated as follows:

$$\begin{cases} \max & z_1 = f_1(x) \\ & \vdots \\ \max & z_k = f_k(x) \\ \text{s.t.} & x \in X \end{cases}$$

where $X \subset \mathbf{R}^n$ denotes the nonconvex set of feasible solutions defined by a set of functional constraints, $x \geq 0$ and x_j integer $j \in J \subset \{1, \ldots, n\}$. It is assumed that X is compact (closed and bounded) and nonempty.

Although a MOMIP problem may be nonlinear, models with linear constraints and linear objective functions have been more often considered. In a multi-objective mixed integer linear programming (*MOMILP*) problem, the functional constraints can be defined as $Ax \leq b$, and the objective functions $f_i(x) = c_i x$, $i = 1, \ldots, k$, where A is a $m \times n$ matrix, b is a m-dimensional column vector and c_i, $i = 1, \ldots, k$, are n-dimensional row vectors.

Multi-objective mixed integer programming is very useful for many areas of application such as communication, transportation and location, among others. Integer variables are required in a real-world model whenever it is sought to incorporate discrete phenomena; for instance, investment choices, production levels, fixed charges, logical conditions or disjunctive constraints. However, research on MOMIP has been rather limited. Concerning multi-objective mathematical programming, most research efforts have been so far devoted to linear programming with continuous variables (MOLP). The introduction of discrete phenomena into multi-objective models leads to all-integer or mixed integer problems that are more difficult to tackle. They can not be handled by most MOLP approaches because the feasible set is no longer convex. Also, there are multi-objective approaches designed for all-integer problems that do not apply to the mixed integer case. Therefore, even for the linear case, techniques for dealing with multi-objective mixed integer programming involve more than the combination of MOLP with multi-objective integer programming techniques.

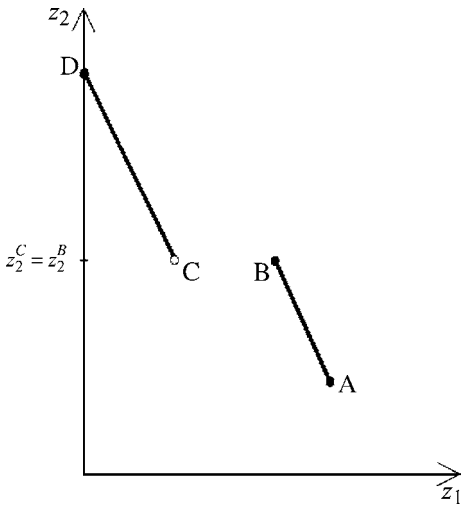
Efficiency and Nondominance

The concept of *efficiency* (or *nondominance*) in MOMIP is defined as usually for multi-objective mathematical programming: A solution $\overline{x} \in X$ is *efficient* if and only if it does not exist another $x \in X$ such that $f_i(x) \geq f_i(\overline{x})$ for all $i \in \{1, \ldots, k\}$ and $f_i(x) > f_i(\overline{x})$ for at least one i. A solution $\overline{x} \in X$ is *weakly efficient* if and only if it does not exist another $x \in X$ such that $f_i(x) > f_i(\overline{x})$ for all $i \in \{1, \ldots, k\}$.

Let $Z \subset \mathbf{R}^k$ be the image of the feasible region X in the criterion (objective function) space. A criterion point $\overline{z} \in Z$ corresponding to a (weakly) efficient solution $\overline{x} \in X$ is called (weakly) *nondominated*. The designations 'efficient', 'nondominated' and 'Pareto optimal' are often used as synonyms.

Supported and Unsupported Nondominated Solutions

Since the feasible region is nonconvex, unsupported nondominated points/solutions may exist in a MOMIP problem. A nondominated point $\overline{z} \in Z$ is *unsupported* if it is dominated by a convex combination (which does not belong to Z) of other nondominated criterion points (belonging to Z). In Fig. 1 the line segment from A to B plus D is the set of supported nondominated criterion points. The line segment from C to D excluding C and D is the set of unsupported nondominated criterion points. Note that convex combinations of B and D dominate the line segment from C to D, excluding D. C is a weakly nondominated solution.



Multi-objective Mixed Integer Programming, Figure 1
Nondominated criterion points of a MOMILP problem

Characterization of the Nondominated Set

Unlike MOLP, the nondominated (or efficient) set of MOMIP problems can not be fully determined by parameterizing on λ the weighted-sums program:

$$(P_\lambda) \quad \begin{cases} \max & \left\{ \sum_{i=1}^{k} \lambda_i f_i(x) : \; x \in X \right\} \\ \text{where} & \lambda \in \Lambda . \end{cases}$$

Here,

$$\Lambda = \left\{ \lambda \in \mathbb{R}^k : \begin{array}{l} \lambda_i > 0 \quad \forall i, \\ \sum_{i=1}^{k} \lambda_i = 1 \end{array} \right\} .$$

The unsupported nondominated solutions cannot be reached even if the complete parameterization on λ is attempted.

Researchers on multi-objective mathematical programming early recognized this fact and stated other characterizations for the nondominated set that fit MOMIP and, in particular, MOMILP problems. Basically, two main characterizations are defined. One consists of introducing additional constraints into the *weighted-sums program*. Generally, these constraints impose bounds on the objective function values. This form of characterization may be regarded as a particularization of the general characterization provided by R.M. Soland [13]. The other is based on the Tchebycheff theory whose theoretical foundation originated from V.J. Bowman [3]. More details about these characterizations and on how they provide the computation of nondominated solutions will be given later. Although providing very important theoretical results, the characterizations of the nondominated set do not offer an explicit means to provide decision support for MOMIP problems. However, some authors have developed decision support methods for these problems.

Interactive Versus Noninteractive Methods

Methods may be either *noninteractive* (in general, generating methods designed to find the whole or a subset of the nondominated solutions) or *inter-*

active (characterized by phases of human intervention alternated with phases of computation). Generating methods for MOMIP problems usually require an excessive amount of computational resources, both in processing time and storage capacity. Even specialized generating algorithms developed just for bi-objective problems, which profit from graphical representations on the criterion space, tend to be inadequate to deal with large problems. Nevertheless, the distinction between interactive and generating methods is not always clear. Some approaches attempt to find a representative subset of the nondominated set (generating methods according to the above definition) and would be easily embodied in an interactive framework. The bi-objective method of R. Solanki [14] may be regarded as an example of such an approach.

Taking into account the difficulties mentioned above, and the large number of nondominated solutions in many problems, special attention to interactive methods will be paid. First of all, a short remark is made about the major paradigms followed by the authors of interactive methods. Some authors admit that the decision maker's (DM) preferences can be represented by an *implicit utility function*. The interactive process consists in building a protocol of interaction aiming to discover the optimum (or an approximation of it) of that implicit utility function. The convergence to this optimum requires no contradictions in the DM's responses given throughout the interactive process.

In contrast with implicit utility function approaches, the *open communication* approaches are based on a progressive and selective learning of the nondominated set. The terminology of open communication is inspired on the concept of open exchange, defined by P. Feyerbend [6]. Such multi-objective approaches are not intended to converge to any 'best' compromise solution but to help the DM to avoid the search for nondominated solutions he/she is not at all interested in. There are no irrevocable decisions during the whole process and the DM is always allowed to go 'backwards' at a later interaction. So, at each interaction, the DM is only asked to give some indications on what direction the search for nondominated solutions must follow, or occasionally to introduce additional constraints. The process only finishes when the DM considers to have gained sufficient insight into the nondominated solution set. Using the terminology of B. Roy [12], 'convergence' must give place to 'creation'. The interactive process is a constructive process, not the search for something 'pre-existent'.

Although we personally prefer the open communication methods, we will include in the next section a tentative classification of both, drawing out some differences and similarities between them. We adopt this perspective because this question is not specific to mixed integer programming and arguments pro or against each approach, besides being subjective, are the same as in other multi-objective programming fields. Furthermore, since MOMIP is still in its early steps, no behavioral studies exist addressing the use of procedures within this context.

As we have mentioned before, research on MOMIP has been rather scarce in comparison to other fields of the multi-objective mathematical programming, namely in MOLP. We will mention herein some well-known methods specially designed for MOMIP or far more generally applicable.

Computing Processes and Their Use in Interactive Methods

Weighted-Sums Programs with Additional Constraints

The introduction of bounds on the objective function values into the *weighted-sums program* (P_λ) enables this program to also compute unsupported nondominated solutions:

$$(P_{\lambda,g}) \quad \max \left\{ \sum_{j=1}^{k} \lambda_i f_i(x) :\ x \in X,\ f(x) \geq g \right\},$$

where $f(x) = (f_1(x), \ldots, f_k(x))$, $\lambda \in \Lambda$ and g is a vector of objective bounds. Besides the fact that every solution obtained by $(P_{\lambda,g})$ is nondominated, there always exists a $g \in \mathbf{R}^k$ such that $(P_{\lambda,g})$ yields a particular nondominated solution. Other types of additional constraints can also be used.

A scalarizing program which consists of the weighted-sums program combined with additional constraints is used for computing nondominated solutions in the interactive branch and bound method of B. Villarreal et al. [18]. The additional constraints are bounds imposed on integer variables by the branching

process. This method, which is devoted to MOMILP problems, received later improvements in [8] and [11]. Starting by applying the well-known (MOLP) *Zionts–Wallenius procedure* to the linear relaxation of the MOMILP problem, the method then employs a branch and bound phase until an integer solution that satisfies the DM is achieved. An implicit utility function is assumed and the DM's preferences are assessed using pairwise evaluations of decision alternatives and trade-off analysis. In light of the DM's underlying utility function, decisions on whether to apply again the Zionts–Wallenius procedure to the linear relaxation of a candidate multi-objective subproblem, or to continue to branch by appending a constraint on a variable, are successively made.

Another method that uses particular forms of $(P_{\lambda,g})$ to compute nondominated solutions is due to Y. Aksoy [1]. This is an interactive method for bicriterion mixed integer programs that employs a branch and bound scheme to divide the subset of nondominated solutions considered at each node into two disjoint subsets. The branching process seeks to bisect the range of nondominated values for z_2 at the node under consideration, checking whether a nondominated point exists whose value for z_2 is in the middle of the range. If no such solution exists, that subset is divided using two nondominated points whose values for z_2 are the closest (one up and the other down) to the middle value. These nondominated solutions are obtained by solving $(P_{\lambda,g})$ optimizing one objective function and bounding the other. The interactive process requires the DM to make pairwise comparisons in order to determine the branching node and to adjust the incumbent solution to the preferred nondominated solution. It is assumed that the DM's preferences are consistent, transitive and invariant over the process aiming to optimize the DM's implicit utility function.

C. Ferreira et al. [5] proposed a decision support system for bicriterion mixed integer programs. The interactive process follows an open communication protocol asking the DM to specify bounds for the objective function values. These bounds are input into $(P_{\lambda,g})$ defining subregions to carry on the search for nondominated solutions. Some objective space regions are progressively eliminated either by dominance or infeasibility.

Tchebycheff and Achievement Scalarizing Programs

Bowman [3] proved that the parameterization on w of $\min_{x\in X} \|\overline{f} - f(x)\|_w$ generates the nondominated set, where $w_i \geq 0$ for all i, $\sum_{i=1}^{k} w_i = 1$, $\overline{f}$ is a criterion point such that $\overline{f} > f(x)$ for all $x \in X$ and $\|\overline{f} - f(x)\|_w$ denotes the *w-weighted Tchebycheff metric*, that is, $\max_{1\leq i\leq k}\{w_i|\overline{f}_i - f_i(x)|\}$. This scalarizing program is equivalent to

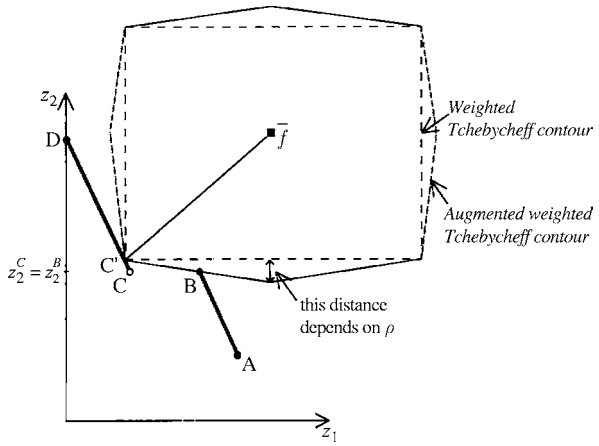
$$(T_w) \quad \begin{cases} \min & \alpha \\ \text{s.t.} & w_i\left(\overline{f}_i - f_i(x)\right) \leq \alpha, \quad 1 \leq i \leq k, \\ & x \in X, \quad \alpha \geq 0. \end{cases}$$

(T_w) may yield weakly nondominated solutions (for instance, point C in Fig. 1). Replacing the objective function in (T_w) by $\alpha - \rho \sum_{i=1}^{k} f_i(x)$ with ρ a small positive value, all the solutions returned by this augmented weighted Tchebycheff program are nondominated. R.E. Steuer and E.-U. Choo [16] proved that there are always ρ small enough that enable to reach all the nondominated set for the finite-discrete and polyhedral feasible region cases.

Concerning the MOMIP case, although there may be portions of the nondominated set that the program is unable to compute, even considering ρ very small (for example, the line segment from C to C' in Fig. 2, for a given ρ), this characterization is still possible in practice. Note that ρ can be set so small that the DM is unable to discriminate between those solutions and a nearby weakly nondominated solution (this corresponds to C' getting closer to C in Fig. 2).

In [16] and [15] a lexicographic weighted Tchebycheff program is proposed for the nonlinear and infinite-discrete feasible region cases to overcome this drawback of the augmented weighted Tchebycheff program. The lexicographic approach can also be applied to the mixed integer (linear) case. However, it is more difficult to implement since two stages of optimization are employed. At the first stage only α is minimized. When the first stage results in alternative optima, a second stage is required. It consists of minimizing $-\sum_{i=1}^{k} f_i(x)$ over the solutions that minimize α in order to eliminate the weakly nondominated solutions.

Besides (T_w) (either the augmented or the lexicographic forms), there are other similar approaches that also allow to characterize the nondominated set of



Multi-objective Mixed Integer Programming, Figure 2
Illustration of the augmented weighted Tchebycheff metric

multi-objective mixed integer programs. An approach of this type consists in discarding the w-vector or fixing it and varying $\overline{f}$, the criterion reference point that represents the DM's aspiration levels. This scalarizing program can be denoted by ($T_{\overline{f}}$). There always exist reference points satisfying $\overline{f} > f(x)$ for all $x \in X$, such that ($T_{\overline{f}}$) produces a particular nondominated solution $\overline{z} = f(\overline{x})$. The variation of $\overline{f}$ can be done according to a vector direction θ, leading to ($T_{\overline{f}+\theta}$). The reference points are thus projected onto the nondominated set. Reference points that do not satisfy the condition $\overline{f} > f(x)$ for all $x \in X$ may also be considered provided that the α variable is defined without sign restriction. This corresponds to the minimization of a distance from Z to the reference point if the latter is not attainable and to the maximization of such a distance if the reference point is attainable. If reference or aspiration levels are used as controlling parameters, the (weighted) Tchebycheff metric changes its form of dependence on controlling parameters and should be interpreted as an *achievement function* [9].

Like (T_w), the simplest form of ($T_{\overline{f}}$) may produce weakly nondominated solutions. The augmented form is a good substitute in practice and the lexicographic approach guarantees that all nondominated solutions can be reached. In what follows, let (T.) denote either the simplest, the augmented or the lexicographic form.

Scalarizing programs (T_w), ($T_{\overline{f}}$) and their extensions or slight different formulations are used to generate nondominated solutions in several (interactive) methods proposed in literature, namely in the following ones.

Steuer and Choo [16] proposed a general purpose multi-objective programming interactive method that assumes an implicit DM's utility function without any special restriction on shape. The strategy of the interactive procedure is to sample series of progressively smaller subsets of nondominated solutions. At each interaction, the DM selects his/her preferred solution from a sample of nondominated solutions obtained from (T_w) with several w-vectors and the ideal criterion point in the role of $\overline{f}$. The solution preferred by the DM provides information to tighten the set of w-vectors for the next interaction. The procedure terminates when a nondominated criterion point sufficiently close to the optimal criterion point of the underlying utility function is found.

Solanki's method [14], which is designed for bi-objective mixed integer linear programs, is an adaptation of the noninferior set estimation (NISE) method developed by J.L. Cohon for bi-objective linear programs. It seeks to generate a representative subset of nondominated solutions by combining the NISE's key features with weighted Tchebycheff scalarizing programs. At each iteration, a new nondominated solution, say z^3, is computed by solving (T_w) for specific w and $\overline{f}$, assuring that z^3 belongs to the region between a pair of nondominated criterion points previously determined, say (z^1, z^2). This pair is then replaced by (z^1, z^3) and (z^3, z^2). The approximation of the nondominated surface is progressively improved, thus decreasing the 'errors' associated with the approximate representation of the pairs. This 'error' is measured by the largest range of the two objectives for the points forming the pair. The algorithm finishes when the maximum 'error' is lower than a predefined maximum allowable 'error'.

Another interactive method capable of solving MOMIP problems was developed by A. Durso [4]. This method employs a branching scheme considering progressively smaller portions of the nondominated set by imposing lower bounds on the criterion values. At each interaction, the k nondominated solutions that define the (quasi)ideal criterion point for each new node are calculated. The DM is then asked to select the node for branching by choosing the preferred ideal point. The branching process begins by solving an equally weighted augmented Tchebycheff program to deter-

mine a 'centralized' nondominated point for the subset of the node under exploration. Once the DM chooses the most preferred of the $k+1$ nondominated points already known for this node, say $\widehat{z}$, up to k new nodes (children) are created. Each child inherits its parent's bounding constraints and uses $\widehat{z}$ to further restrict one of them. Thus, the ith child restricts the ith criterion by imposing $f_i(x) \geq \widehat{z}_i + \delta$ with δ small positive. This approach may be regarded as an open communication procedure that terminates when the DM is satisfied with the incumbent solution (the preferred nondominated solution obtained so far).

M.J. Alves and J. Clímaco [2] proposed a MOMILP open communication interactive approach. It combines the Tchebycheff theory with the traditional branch and bound technique for solving single-objective mixed integer programs. At each interaction, the DM specifies either a reference point $\overline{f}$, which is input in ($T_{\overline{f}}$) to compute a nondominated solution via branch and bound, or just selects an objective function, say f_j, he/she wants to improve with respect to the previous nondominated solution. In the latter case, the reference point is automatically adjusted by increasing the jth component of $\overline{f}$ keeping the others equal, in order to produce new nondominated solutions (directional search) more suited to the DM's preferences. This involves an iterative process of sensitivity analysis and operations to update the branch and bound tree. The sensitivity analysis takes advantage of the special behavior of the parametric scalarizing program ($T_{\overline{f}+\theta}$). It returns a value $\overline{\theta}_j > 0$ such that the structure of the previous branch and bound tree remains unchanged for variations in $\overline{f}_j$ up to $\overline{f}_j + \overline{\theta}_j$. Therefore, reference points $\overline{f} + \theta = (\overline{f}_1, \ldots, \overline{f}_j + \theta_j, \ldots, \overline{f}_k)$ with $\theta_j \leq \overline{\theta}_j$ lead to nondominated solutions that may be obtained in a straightforward way. If the DM wishes to continue the search in the same direction, a slight increase over $\overline{\theta}_j$, say $\overline{\theta}_j + \epsilon$, is first considered. In this case, the previous sensitivity analysis also returns the best candidate node, i. e., an ancestor of the node that will produce the next nondominated solution. The previous branch and bound tree is thus used to proceed to the next computations. Since further branching is usually required, an attempt is made to simplify the tree before enlarging it. The underlying idea is to avoid an evergrowing tree. This simplification means cutting off parts of the tree linked by branching constraints no longer active. In sum, this approach brings together sensitivity analysis phases meant to adjust the reference point and simplification/branching operations of the search tree to compute nondominated solutions. This process is repeated as long as the DM wishes to continue the directional search or if the reference point has not been adjusted enough to yield a nondominated solution different from the previous one (a situation that occurs more often in all-integer programs than in mixed integer models). Computational experiments have shown that this multi-objective approach succeeds in performing directional searches. The times of computing phases using simplification/branching operations have been significantly reduced by this strategy.

Some researchers have developed other methods for multi-objective integer programming that are also applicable to the mixed integer case. Good examples of such approaches are those in [10,17] and [7]. In our opinion, they all are open communication procedures that share some key features, namely the concept of projecting a reference direction onto the nondominated surface (although this procedure is used in different ways) and the type of information required about the DM's preferences. This information lies fundamentally in the specification of aspiration levels for the objective function values (reference points). Some of these approaches are continuous/integer ([7,10]) working almost all the time with nondominated continuous solutions of the linear relaxation of the problem. Whenever the DM finds a satisfactory continuous solution, an integer nondominated solution close to it is then computed.

Conclusions and Future Developments

Most methods developed so far for MOMIP problems require an excessive amount of computational effort, or require too much cognitive load from the DM, or only address bi-objective problems. In addition, computational experience with real-world applications is lacking. Although interesting or promising approaches have been developed, further research efforts must be made in order to build effective interactive methods able to handle real-sized problems.

See also

► Branch and Price: Integer Programming with Column Generation

- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Graph Coloring
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem

References

1. Aksoy Y (1990) An interactive branch-and-bound algorithm for bicriterion nonconvex/mixed integer programming. Naval Res Logist 37:403–417
2. Alves MJ, Clímaco J (2000) An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound. Europ J Oper Res 124(3):478–494
3. Bowman VJ (1976) On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. In: Thiriez H, Zionts S (eds) Multiple Criteria Decision Making. Lecture notes Economics and Math Systems. Springer, Berlin, pp 76–86
4. Durso A (1992) An interactive combined branch-and-bound/Tchebycheff algorithm for multiple criteria optimization. In: Goicoechea A, Duckstein L, Zionts S (eds) Multiple Criteria Decision Making, Proc 9th Internat Conf. Springer, Berlin, pp 107–122
5. Ferreira C, Santos BS, Captivo ME, Clímaco J, Silva CC (1996) Multiobjective location of unwelcome or central facilities involving environmental aspects: A prototype of a decision support system. Belgian J Oper Res Statist Comput Sci 36(2–3):159–172
6. Feyerabend P (1975) Against method. Verso, London
7. Karaivanova J, Korhonen P, Narula S, Wallenius J, Vassilev V (1995) A reference direction approach to multiple objective integer linear programming. Europ J Oper Res 81:176–187
8. Karwan MH, Zionts S, Villarreal B, Ramesh R (1985) An improved interactive multicriteria integer programming algorithm. In: Haimes Y, Chankong V (eds) Decision Making with Multiple Objectives. Lecture notes Economics and Math Systems. Springer, Berlin, pp 261–271
9. Lewandowski A, Wierzbicki A (1988) Aspiration based decision analysis and support. Part I: Theoretical and methodological backgrounds. WP-88-03, Internat Inst Appl Systems Anal (IIASA), Austria
10. Narula SC, Vassilev V (1994) An interactive algorithm for solving multiple objective integer linear programming problems. Europ J Oper Res 79:443–450
11. Ramesh R, Zionts S, Karwan MH (1986) A class of practical interactive branch and bound algorithms for multicriteria integer programming. Europ J Oper Res 26:161–172
12. Roy B (1987) Meaning and validity of interactive procedures as tools for decision making. Europ J Oper Res 31:297–303
13. Soland RM (1979) Multicriteria optimization: A general characterization of efficient solutions. Decision Sci 10:26–38
14. Solanki R (1991) Generating the noninferior set in mixed integer biobjective linear programs: An application to a location problem. Comput Oper Res 18(1):1–15
15. Steuer R (1986) Multiple criteria optimization: Theory, computation and application. Wiley, New York
16. Steuer R, Choo E-U (1983) An interactive weighted Tchebycheff procedure for multiple objective programming. Math Program 26:326–344
17. Vassilev V, Narula SC (1993) A reference direction algorithm for solving multiple objective integer linear programming problems. J Oper Res Soc 44(12):1201–1209
18. Villarreal B, Karwan H M, Zionts S (1980) An interactive branch and bound procedure for multicriterion integer linear programming. In: Fandel G, Gal T (eds) Multiple Criteria Decision Making: Theory and Application. Lecture notes Economics and Math Systems. Springer, Berlin, pp 448–467

Multi-objective Optimization and Decision Support Systems

SERPIL SAYIN
Koç University, İstanbul, Turkey

MSC2000: 90B50, 90C29, 65K05, 90C05, 91B06

Article Outline

Keywords
Traditional Classification
Multi-Objective Linear Programming
Working in the Outcome Space
Reflections on Optimization Trends
Nonlinear and Integer Problems
Applications
A Related Optimization Problem
Trends
See also
References

Keywords

Multiple criteria decision making; Vector optimization; Efficient solution; Decision support

Multiple criteria decision making (*MCDM*) refers to the explicit incorporation of more than one evaluation criteria into a decision problem. MCDM has been a very active field of research roughly since the 1970s. Although boundaries might be fuzzy and overlapping, multicriteria decision analysis (studying the problem of identifying the 'most-preferred' among a finite discrete set of alternatives), multi-attribute utility theory (using utility functions explicitly to model a decision maker's preferences) and multi-objective optimization (modeling the decision problem within a mathematical programming framework) have emerged as major fields of interest under MCDM. For more information on the general field of MCDM, see [21].

Multi-objective mathematical programming provides a flexible modeling framework that allows for simultaneous optimization of more than one objective function over a feasible set. Mathematically, the multi-objective optimization problem can be expressed as:

$$\text{(MOO)} \quad \begin{cases} \max & f(x), \\ \text{s.t.} & x \in X, \end{cases}$$

where $X \subseteq \mathbf{R}^n$ is the set of feasible alternatives and $f = (f_1, \ldots, f_p)$: $\mathbf{R}^n \to \mathbf{R}^p$, $p \geq 2$, is a vector-valued function. Note that X can be any set, continuous or discrete, expressed through constraints, and the objective function f can be of any form.

The increased flexibility provided by (MOO) also raises the question of what constitutes a solution to it. The definition of optimality is no longer valid, as each objective function would possibly yield a different optimal solution. Therefore solving the (MOO) problem is about studying the inherent trade-offs among conflicting objectives. Efficient solutions are the ones that possess the relevant trade-off information. An $x^o \in \mathbf{R}^n$ is called an *efficient solution* for the (MOO) problem if $x^o \in X$ and there exists no $x \in X$ such that $f(x) \geq f(x^o)$ with strict inequality holding for at least one component. The set of all efficient solutions of the (MOO) problem is usually denoted by X_E. As per the above definition, the most-preferred solution of the decision maker should belong to X_E, as solutions that are not efficient, the dominated ones, can be improved upon in at least one objective without worsening the others.

Since X_E is usually a big set, confining the most-preferred solution to X_E does not help identify the most-preferred solution immediately. In particular, the difficulty of defining and obtaining the most-preferred solution, the one that the decision maker would identify as the solution to the decision-making problem, and the need for the inevitable involvement of the decision maker in the solution procedure has resulted in very different solution approaches to the (MOO) problem.

Traditional Classification

The timing of the involvement of the decision maker in the solution procedure has been a crucial factor that distinguishes among various approaches to the (MOO) problem [13]. *A priori methods*, methods that use prior articulation of preferences, ask the decision maker to specify preference information prior to the application of an optimization routine. The elicitation of preference information can be directed towards deriving a utility function that describes the decision maker's preferences [14], or as in goal programming [7] and compromise programming [23], a standard model can be imposed upon the decision maker. As these methods reduce the (MOO) problem to a single-objective optimization problem and they aspire to find a single solution to it, they have received considerable recognition although their assumptions are usually restrictive.

The *interactive methods* require the interaction of the decision maker with the computer while solving a particular (MOO) problem. Usually, the idea is to construct a model that proposes solutions to the

(MOO) problem based on some initial input. The decision maker is then invited to reply to the solution by providing additional preference information. The interaction between the computer program and the decision maker continues until a satisfying solution is obtained.

Interactive methods are important in more than one way. First, they have introduced the means for practically solving a (MOO) problem [12]. Second, they help a decision maker learn about the inherent trade-offs of a problem during the solution process [5]. Third, the idea underlying the interactive methods constitutes the major motivation behind the contemporary *decision support systems*. Although interactive algorithms have encountered a certain level of acceptance from practitioners [1,20], they are not without disadvantages. They usually rely too much on the information provided by the decision maker, are not able to provide a global look at X_E, and thus at the trade-offs inherent in a problem, and they focus on finding a single solution whereas a number of solutions may be compatible with the decision maker's preferences. Moreover, their information requests may be overwhelming for the decision maker. It has been discussed that interactive methods need to address behavioral aspects of decision making [16] and concentrate on interfacing the decision maker[15] as well as broadening their model base [10]. Although they do not encompass all the raised issues, some of the interactive (MOO) algorithms have already evolved into decision support systems that provide a friendly environment for modeling as well as problem solving [17]. It can be expected that more decision support systems to solve problem (MOO) will appear in the near future.

Perhaps the most straight-forward way of approaching the (MOO) problem is as in *vector optimization* methods. Also referred to as *posterior methods*, these methods are based on the sole assumption that the decision maker prefers more to less in each objective function in (MOO) hence they propose identifying all of the efficient solutions of (MOO) and presenting them to the decision maker for the identification of the most-preferred solution. Along with theoretical findings [2,11], some vector optimization methods have been proposed; however, the methods have not gained practical recognition in general. The failure in the implementation of the proposed methods can be explained by the heavy computational requirements of these methods. Perhaps a more important factor is the difficulty of presenting the efficient set in a 'legible' way to the decision maker. Furthermore, as the efficient set is usually continuous when the feasible region is, the task of identifying the most-preferred solution is a monstrous one attributed to the decision maker.

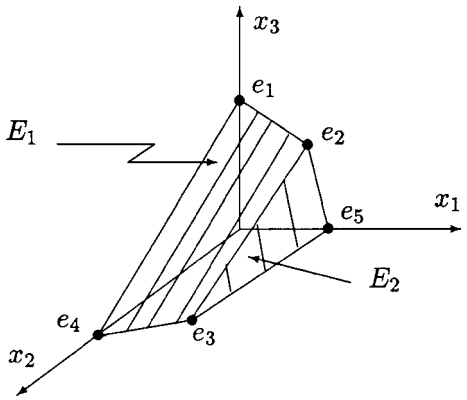
Multi-Objective Linear Programming

When (MOO) has linear objective functions and a polyhedral feasible set, the resulting problem is called a *multiple objective linear programming* (MOLP) problem. The MOLP problem has mathematical features that make it easier to characterize and obtain the efficient set compared to the more general case. More specifically, it has been shown that the efficient set of the MOLP problem consists of a collection of efficient faces of the feasible region. As faces of a polyhedron can be characterized in a number of ways, for instance as the convex hull of its extreme points if its compact, as the optimal solution set to a particular optimization problem, or as a polyhedron itself, it becomes possible to obtain and present the efficient set [9,18,22].

Yet the computational effort increases with problem size, and the (MOO) problem cannot be considered truly solved at this stage without some mechanism that helps the decision maker identify the most-preferred solution in this huge and hard-to-explore set. Most of the vector optimization methods have concentrated on finding the set of efficient extreme points of the multiple objective linear programming problem. These are usually methods that rely on simplex-like procedures or parametric searches that incorporate book-keeping mechanisms based on the fact that the set of efficient extreme points is connected. A well-known procedure that solves (MOLP) for all of its extreme points is ADBASE which was developed by R.E. Steuer [19].

Example 1 Consider the MOLP problem [18]:

$$\begin{cases} \max & x_1, x_2, x_3, \\ \text{s.t.} & 2x_1 + 3x_2 + 4x_3 \leq 12 \\ & 4x_1 + x_2 + x_3 \leq 8 \\ & x_1, x_2, x_3 \geq 0. \end{cases} \tag{1}$$



The efficient set is the union of the two shaded efficient faces E_1 and E_2. There are 5 efficient extreme points: $e_1 = (0, 0, 3)$, $e_2 = (10/7, 0, 16/7)$, $e_3 = (12/10, 32/10, 0)$, $e_4 = (0, 4, 0)$, $e_5 = (2, 0, 0)$. If X denotes the feasible region, The face marked E_1 can be characterized as the polyhedron that forces the first constraint in (1) to equality in the definition of X. It can also be defined as the convex hull of its four extreme points e_1, e_2, e_3, e_4. Finally, it is the optimal solution set to the optimization problem

$$\begin{cases} \max & \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 \\ \text{s.t.} & x \in X \end{cases}$$

for $(\lambda_1, \lambda_2, \lambda_3) = (2, 3, 4)$, and its positive multiples.

In large problems, the set of efficient extreme points may still contain too many points to be studied by the decision maker. Moreover, extreme efficient points may not carry the trade-off information well since some portions of the efficient set may end up being overemphasized whereas some regions are highly missed. Indeed, there is no reason for a decision maker to be solely interested in extreme point efficient solutions. The attractiveness of efficient extreme points mostly lies in their mathematical properties. With this motivation, a method that applies to a general set of (MOO) problems has been suggested to find globally-representative subsets of the efficient set [6].

Working in the Outcome Space

The outcome set $Y = \{ y \in \mathbf{R}^p : y = f(x) \; \exists \; x \in X \}$ helps redefine an equivalent problem to (MOO) in p-dimensional *outcome space*:

$$\text{(MOOO)} \quad \begin{cases} \max & y \\ \text{s.t.} & y \in Y. \end{cases}$$

As the number of objectives p is usually much less than the number of variables n, the structure of Y is simpler than that of X [4,8]. The ability to work directly with (MOOO) thus has the potential of providing significant computational benefits that vector optimization algorithms have tried to realize [3].

Reflections on Optimization Trends

As a field within the general field of optimization, multi-objective optimization is naturally affected by the trends that become dominant in optimization. Consequently, interior point methods, genetic algorithms, neural networks have been applied to the (MOO) problem in various ways. As there are difficult problems under (MOO) that cannot be yet practically solved, new developments in the general field of optimization constitute a potential to solve these problems.

Nonlinear and Integer Problems

Most of the algorithms proposed to solve problem (MOO) concentrate on the fully linear case. In general, when nonlinearities are introduced, the efficient solutions and the efficient set become difficult to characterize. There are some algorithms that allow for nonlinearities in the objective functions, and in the constraints that define the feasible region, but usually in a conservative way so as to retain some computational tractability. Similarly, the multiple objective integer programming problem is a very difficult one to solve due to the additional complications related to integrality.

Applications

Along with what one can call 'case studies', certain applications that are more generic than a case study but more specific than problem (MOO) itself have appeared. Typical examples include, but are not limited to, bicriteria network optimization problems, bicriteria knapsack problems, and multicriteria scheduling problems. Since usually these are problems that naturally involve multiple criteria, the methods developed for these problems have practical implications. Most of the methods developed can be categorized under a priori methods. A typical approach is to form a weighted combination of the objective functions. Recently, interactive and vector optimization approaches that deal with similar problems have also appeared.

A Related Optimization Problem

A related problem is the problem of optimizing a function g: $\mathbf{R}^n \rightarrow \mathbf{R}^p$ over the efficient set X_E. This can be a difficult *global optimization* problem depending on the properties of the objective function g. The problem is motivated in different ways. Sometimes, in certain settings, a function that is to serve as a pseudo utility function is available. Then optimizing this pseudo utility function over the efficient set in a sense corresponds to solving problem (MOO) itself. In addition, when g becomes one of the objective functions, then solving this problem provides the range of values the objective function takes over the efficient set. This information is valuable for a decision maker who is trying to make assessments to solve a problem and is used in some of the interactive algorithms. The difficulty of the problem has also resulted in heuristic solution approaches.

Trends

The advances in information technology affect the field of multiple criteria decision making heavily. Faster computers and parallel processing opportunities make it timewise feasible to solve optimization problems that would be deemed impractical in the past. Improved graphical capabilities make it feasible to accommodate sophisticated user interfaces to invite the decision maker in the problem solving process more actively and reliably. The developments in the World Wide Web present many opportunities to explore for individual and group decision support. At this point in time, there is still a need to solve the MOO problem in a rigorous, user-friendly and creative way. The decision support systems that enable the involvement of the decision maker in modeling and problem solving practically seem to be the way of solving (MOO) problems. The vector optimization approaches can also benefit from a decision support framework in their effort to help the decision maker identify a most-preferred solution.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Benayoun R, De Montgolfier J, Tergny J, Larichev O (1971) Linear programming with multiple objective functions: Step method (STEM). Math Program 1:366–375
2. Benson HP (1978) Existence of efficient solutions for vector maximization problems. J Optim Th Appl 26:569–580
3. Benson HP (1998) A hybrid approach for solving multiple objective linear programs in outcome space. J Optim Th Appl 98:17–35
4. Benson HP, Lee D (1996) Outcome-based algorithm for optimizing over the efficient set of a bicriteria linear programming problem. J Optim Th Appl 88(1):77–105
5. Benson HP, Lee D, McClure JP (1997) A multiple-objective linear programming model for the citrus rootstock selection problem in Florida. J Multi-Criteria Decision Anal 6:1–13
6. Benson HP, Sayin S (1997) Towards finding global representations of the efficient set in multiple objective mathematical programming. Naval Res Logist 44:47–67
7. Charnes A, Cooper WW (1977) Goal programming and multiple objective optimization-Part 1. Europ J Oper Res 1:39
8. Dauer JP, Liu Y-H (1990) Solving multiple objective linear programs in objective space. Europ J Oper Res 46:350–357
9. Ecker JG, Hegner NS, Kouada IA (1980) Generating all maximal efficient faces for multiple objective linear programs. J Optim Th Appl 30:353–381
10. Gardiner LR, Steuer RE (1994) Unified interactive multiple-objective programming - An open-architecture for accommodating new procedures. J Oper Res Soc 45(12):1456–1466
11. Geoffrion AM (1968) Proper efficiency and the theory of vector maximization. J Math Anal Appl 22:618–630

12. Geoffrion AM, Dyer JS, Feinberg A (1972) An interactive approach for multi-criterion optimization with an application to the operations of an academic department. Managem Sci 19:357–368
13. Hwang CL, Masud ASM (1979) Multiple objective decision making-methods and applications, A state of the art survey. Lecture Notes Economics and Math Systems. Springer, Berlin
14. Keeney RL, Raiffa H (1976) Decisions with multiple objectives: Preferences and value tradeoffs. Wiley, New York
15. Korhonen P, Laakso J (1986) A visual interactive method for solving the multiple criteria problem. Europ J Oper Res 24:277–287
16. Korhonen P, Moskowitz H, Wallenius J (1990) Choice behavior in interactive multiple criteria decision making. Ann Oper Res 23:161–179
17. Korhonen P, Wallenius J (1988) A Pareto race. Naval Res Logist 35:615–623
18. Sayin S (1996) An algorithm based on facial decomposition for finding the efficient set in multiple objective linear programming. Oper Res Lett 19:87–94
19. Steuer RE (1983) Operating manual for the ADBASE multiple objective linear programming package. Techn Report College Business Admin Univ Georgia, Athens
20. Steuer RE, Choo E-U (1983) An interactive weighted Tchebycheff procedure for multiple objective programming. Math Program 26:326–344
21. Steuer RE, Gardiner LR, Gray J (1996) A bibliographic survey of the activities and the international nature of multiple criteria decision making. J Multi-Criteria Decision Anal 5:195–217
22. Yu PL, Zeleny M (1975) The set of all nondominated solutions in linear cases and a multicriteria simplex method. J Math Anal Appl 49:430–468
23. Zeleny M (1973) Compromise programming. In: Cochrane JL, Zeleny M (eds) Multiple Criteria Decision Making. Univ South Carolina Press, Columbia

Multi-objective Optimization: Interaction of Design and Control

Carl A. Schweiger, Christodoulos A. Floudas
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C29, 90C11, 90C90

Article Outline

Keywords
Multi-objective Optimization
Multi-objective Framework
for the Interaction of Design and Control
General Mathematical Formulation
Solution of the MOP
Noninferior Solution Sets
Choosing the Best-Compromise Solution
Cutting Plane Algorithm
Multi-objective Optimization
in the Interaction of Design and Control
Conclusions
See also
References

Keywords

Interaction of design and control; Multi-objective optimization; Mixed integer nonlinear optimization; Pareto optimal solution

Traditionally, *process design* and *process control* are treated sequentially. Dynamics are not considered during the design phase, and flowsheet changes can not be made during the control phase. The problem with this approach is that the two are inherently connected as the design of the process affects its controllability. Thus, the steady state design and the dynamic operability issues should be treated simultaneously. Analyzing the *interaction of design and control* addresses the issue of quantitatively determining the *trade-offs* between the steady state economics and the dynamic controllability.

The interaction of design and control problem is to determine the process flowsheet which is both the economically optimal and controllable. There are different methods for addressing this problem. One common approach is to use overdesign where, once the economic steady state design is determined, surge tanks are added or equipment sizes are increased in order to handle any dynamic problems which may arise. This overdesign is usually based on heuristic rules and will likely move the design away from its economic optimum. There is no guarantee that the measures taken will even improve the controllability of the process. Other methods may examine the dynamic operation of several designs to determine which has the best controllability aspects.

There are very few methods which address the interaction of design and control in a quantitative manner.

The interaction of design and control can be addressed through a *process synthesis* approach involving optimization. This approach involves the representation of design alternatives through a process superstructure, the mathematical modeling of the superstructure, and the development of an algorithm to extract the optimal flowsheet from the superstructure. The simultaneous optimization of the design and control of the process is handled through multiple objectives representing the steady state economics and dynamic controllability. This naturally leads to a multi-objective framework.

Multi-objective Optimization

In any decision making process, the goal is to reach the best compromise solution among a number of competing objectives. Many examples of competing objectives exist in the field of engineering. For example, in the design of a process, one may have to consider safety and operational issues as well as economic issues. A decision making process is necessary when the most economic design is not the safest or most operable.

The best compromise solution depends on the relative importance of the conflicting objectives. This relative importance is not easily determined and is usually a subjective decision. The one responsible for making this decision is the decision maker (DM) whose choice can be based on a number of factors. Since subjective measures and decisions do not translate well into mathematics, a quantitative way of determining the tradeoffs and relative importance among the the objectives is necessary for a multi-objective optimization framework.

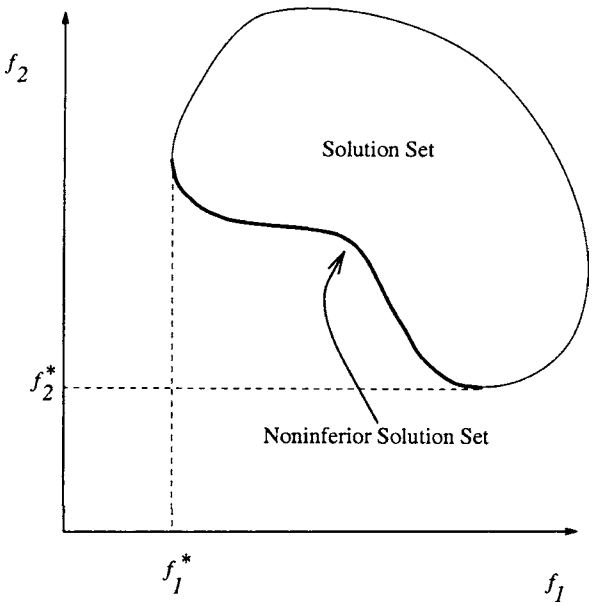
Multi-objective Framework for the Interaction of Design and Control

In analyzing the interaction of design and control, the objectives that are considered measure the steady state economics and the dynamic controllability of the process. The optimization approach in process synthesis serves as the basis for the multi-objective framework for the interaction of design and control. The procedure involves four steps:

1) Process representation;
2) Mathematical modeling;
3) Generation of noninferior solution set (determine trade-offs);
4) Best-compromise examination.

The first step is the representation of all the possible design alternatives through a process superstructure. In this step, all the units and possible connections of interest are incorporated into the superstructure such that all designs of interest are included as a subset of the superstructure.

Next, a mathematical model of the superstructure is developed for the superstructure as well as for for objective functions. The mathematical formulation is determined by the structure of the process flowsheet and must include all information needed to evaluate the objective functions. The objective functions must measure the economics of the process as well as the controllability of the process. Since the objective related to the economic performance is determined by steady state operation and the objective for the controllability is determined by its dynamic operation, the mathematical model most contain both steady state and dynamic information. The mathematical formulation involves both continuous and discrete variables where discrete variables are used to indicate the existence of units and connections within the flowsheet.



Multi-objective Optimization: Interaction of Design and Control, Figure 1
Noninferior solution set for a problem with two objectives

Once the model has been formulated, an algorithm is developed and used to determine the quantitative trade-offs among the competing objectives. Individually, each objective can be optimized, but together, they will be in conflict. This means that there is a set of solutions where one objective can be improved only at the expense of the other objectives. This set of solutions is called the *noninferior solution set* which is visually depicted for a two objective problem in Fig. 1. This solution set is also referred to as *nondominated* and *Pareto optimal* and the surface of noninferior solutions implicitly defines a function $G(\mathbf{J})$.

Using the information about the trade-offs among the competing objectives, a strategy for determining the best compromise solution is developed. This strategy is based on information from the DM and depends on the relative weights given to the objectives. These weights are varied systematically to locate the solution which the DM prefers the most. How to determine these weights is one of the more interesting aspects of the problem.

Note that the multi-objective problem can be reduced if some of the objectives (presumably those with very low weights) need not be optimized but simply brought to a satisfactory level. In this case, these objectives can be incorporated into the problem as constraints.

General Mathematical Formulation

The mathematical model is a multi-objective *mixed integer nonlinear programming* problem which has the following form:

$$\begin{cases} \text{OPTIMIZE} & \mathbf{J}(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q. \end{cases} \tag{1}$$

In this formulation, $\mathbf{J}$ is a vector of objectives which includes the economic and controllability objectives. The expressions $\mathbf{h}$ and $\mathbf{g}$ represent material and energy balances, thermodynamic relations, and other constraints. The controllability measures are included in the formulation as $\boldsymbol{\eta}$. The variables in this problem are partitioned as continuous $\mathbf{x}$ and binary $\mathbf{y}$.

Solution of the MOP

One way to address the solution of the MOP is to formulate it using a utility function U which implicitly relates the multiple objectives in terms of some common basis:

$$\begin{cases} \min & U[\mathbf{J}(\mathbf{x},\mathbf{y})] \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q. \end{cases} \tag{2}$$

By introducing the utility function, the vector optimization problem has been reduced to a scalar optimization problem and MINLP techniques can be applied to solve the problem. These MINLP techniques include *generalized Benders decomposition* (GBD) [4,14], *outer approximation* (OA) [2], *outer approximation with equality relaxation* (OA/ER) [8], and *outer approximation with equality relaxation and augmented penalty* (OA/ER/AP) [16]. These methods are discussed in detail in [3].

With the definition of the noninferior solution set, the optimization problem can be formulated as

$$\begin{cases} \min & U[\mathbf{J}(\mathbf{x},\mathbf{y})] \\ \text{s.t.} & G(\mathbf{J}) = 0. \end{cases} \tag{3}$$

The challenging aspect of the problem is determining the explicit form of the utility function. One possible form of the utility function is a weighted linear sum of the objectives:

$$U[\mathbf{J}(\mathbf{x},\mathbf{y})] = \sum_{i \in I} w_i J_i,$$

where I is the set of objective functions and w_i are the weights for the objective functions whose value is determined by the DM. The difficulty that arises is that the utility function is generally not known. It is, however, assumed to be convex and continuously differentiable.

The issues surrounding the solution of the multi-objective optimization problem are determining the noninferior solution set, determining the utility function based on information from the DM, and determining the best-compromise solution.

Different techniques have been developed in order to assess the trade-offs among the objectives quantita-

tively. See [7] for a tutorial in multi-objective optimization. A review is also available in [17]. Much of the fundamental aspects of multi-objective optimization can be found in [1].

Noninferior Solution Sets

The noninferior solution set can be determined in a number of ways. One approach is the formulate the problem as

$$\begin{cases} \min & \sum_{i\in I} w_i J_i(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q, \end{cases} \tag{4}$$

where the weights w_i are selected such that $w_i \ge 0$ for all i and $\sum_{i\in I} w_i = 1$. Through a suitable choice of the weights, the noninferior solution set can be found. This approach can miss some points in the noninferior solution set if the solution region is nonconvex. In order to address this problem, a weighted norm can be used as follows:

$$\begin{cases} \min & \left\{ \sum_{i\in I} \left[w_i J_i(\mathbf{x},\mathbf{y}) \right]^p \right\}^{1/p} \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q. \end{cases} \tag{5}$$

By increasing the size of p, the curvature of the supporting function is increased and more noninferior points can be found. In the extreme of $p = \infty$, all the noninferior points can be located. Using the ∞-norm, the problem becomes

$$\begin{cases} \min & \max_{i\in I} w_i J_i(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q. \end{cases} \tag{6}$$

The advantage of this formulation is that the weights have a physical meaning for the DM. If the DM knows the desired values for each objective for a given noninferior point, the weights can be set to the reciprocal of these values. The noninferior solution will be the one that is most like the one with the values specified by the DM. The disadvantage of this formulation is that it can be difficult to solve.

Another way to determine the noninferior solution set is through the *ϵ-constraint method* [6]. In this approach, all but one of the objectives is incorporated into the problem as a constraint less than ϵ. This results in the following formulation:

$$\begin{cases} \min & J_1(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & J_i(\mathbf{x},\mathbf{y}) \le \epsilon_i, \quad i = 2,\ldots,q, \\ & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q. \end{cases} \tag{7}$$

By varying the values of ϵ_i, the points of the noninferior solution set can be found.

Choosing the Best-Compromise Solution

To this point, the focus has been on determining the noninferior solution set. Only one of the points can be chosen as the best solution for the problem, and the task of the DM is to determine this point. Once the noninferior solution set is determined, it is presented to the DM who will choose the solution point he prefers. The selection of this point is based on the relative importance of the objectives in the eyes of the decision maker.

Instead of assigning arbitrary weights to the various objectives, a systematic approach can applied which uses the trade-off information in the noninferior solution set. The slope of the noninferior solution set at any point reveals how much one objective will be improved at the expense of another objective. This information is used in an interactive, iterative *cutting plane algorithm* to determine the best compromise solution.

Cutting Plane Algorithm

The cutting plane algorithm described in [11] is based on [5] and [10]. Marginal rates of substitution were used to solve problems of the form (2) where U is unknown, convex, and continuously differentiable. Due to

convexity, the partial derivatives of U with respect to each of the arguments in the objective space are positive. This is expressed mathematically as

$$\frac{\partial U(\mathbf{J})}{\partial J_i} > 0.$$

Thus, a decrease in J_i will lead to a decrease in U. In the interactive scheme, the DM is asked for the positive trade-off weights, w_i^k, for a given solution k. This weight is defined as the ratio of the change in the utility function with respect to one function divided by the change in the utility function with respect to another. This is expressed mathematically as

$$w_i^k = \frac{\partial U(\mathbf{J}^k)\,/\,\partial J_i}{\partial U(\mathbf{J}^k)\,/\,\partial J_1}$$

where $\mathbf{J}^k = [J_1(\mathbf{x}^k, \mathbf{y}^k), \ldots, J_1(\mathbf{x}^k, \mathbf{y}^k)]$. A line search along a feasible direction of steepest descent locates an improved solution for the next iteration.

By exploiting the fact that the utility function is convex, cutting planes can be introduced to reduce the search to improving directions [10]. Since U is convex,

$$\begin{aligned} 0 &\geq U(\mathbf{J}^*) - U(\mathbf{J}^k) \\ &\geq \nabla_f U(\mathbf{J}^k)(J^* - \mathbf{J}^k) \\ &\geq \begin{cases} \min & \nabla_f U(\mathbf{J}^k)(\mathbf{J} - \mathbf{J}^k) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0, 1\}^q. \end{cases} \end{aligned} \tag{8}$$

This involves the linearization in the objective space around the point J^k. If the solution to the minimization is zero, then the optimal solution $J^\star$ has been found. If the solution has a negative value, then the direction leads to an improvement in the objective space. This minimization can be performed over a number of points $k = 1, \ldots, K$ to find a direction which improves all of them. Cutting planes in the objective space are formed to find new values of the objectives which improve the utility function according to the trade-off weights, ∇U, which the DM provides. At each iteration of the algorithm, the following problem must be solved:

$$\begin{cases} \min & z \\ \text{s.t.} & z \geq \sum_{i=1}^{p} w_i^k (J_i(\mathbf{x}, \mathbf{y}) - J_i(\mathbf{x}^k, \mathbf{y}^k), \\ & \quad \forall k = 1, \ldots, K. \\ & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0, 1\}^q. \end{cases} \tag{9}$$

The steps of the cutting plane algorithm are the following:

1. Determine the initial solution point $k = 1$ and determine the values of all the objective functions.
   Assign the values of the weights w_i^k.
2. Solve (9) to find new values of $\mathbf{x}$ and $\mathbf{y}$.
   Determine the values of the objective functions for the new values of $\mathbf{x}$ and $\mathbf{y}$.
3. IF the solution to (9) is zero, THEN go to Step 4 ELSE set $k = k + 1$, update the values $\mathbf{x}^k$, $\mathbf{y}^k$, and $\mathbf{J}^k$, generate new weights, and go to Step 2.
4. Terminate with $\mathbf{x}^k$ and $\mathbf{y}^k$ as the best-compromise solution.

Cutting plane algorithm

This algorithm requires the DM to provide only trade-off weights at each iteration. These weights can be estimated by knowledge of the relative importance of the objectives or by information from the noninferior solution set.

Multi-objective Optimization in the Interaction of Design and Control

The interaction of design and control has been recognized as a multi-objective problem by many researchers as the objectives representing the steady-state economic design and dynamic controllability are regarded as noncommensurable. One of the first challenges in this problem is determining a suitable controllability objective. The choice of the controllability objective will dictate the required elements of the mathematical formulation of the problem.

One of the early works which addressed the multi-objective nature of the interaction of design and control was that of [9]. A given set of alternative steady-state designs was assumed to be known. Bounds on the dynamic measures of the designs were determined and used to screen designs and determine the noninferior solution set. No method was provided for determining the best-compromise solution.

In the work of [13], singular value decomposition is used to determine dynamic operability measures. The controllability is formulated through the linearization of the model and is given in terms of the singular values of the transfer function. This modeling leads to an infinite-dimensional problem as all frequencies must be considered for the controllability measure. For the multi-objective optimization, the ϵ-constraint method was used to determine the noninferior solution set. The scalar optimization was addressed by approximating the infinite-dimensional problem and using an gradient-based algorithm to solve the optimization problem and determine the operating parameters for the process.

The previous methods did not take into account that the structure of the process flowsheet as well as the design parameters determine its inherent controllability. In order to consider structural alternatives in the process flowsheet such as the existence of units in the flowsheet, discrete variables are used in the process modeling. This aspect of the process design was considered by [11,12] in the interaction of design and control by using the optimization approach to process synthesis. In this approach, the structure of the process flowsheet and the design parameters are considered simultaneously with the dynamic controllability of the process. The controllability measures employed were the open-loop linear controllability measures (singular value, condition number, relative gain array). The noninferior solution set was determined using the ϵ-constraint method, and the best-compromise solution was found using the cutting plane method described above.

Further development of the above technique was addressed by [15] where nonlinear dynamic models were considered. The problem was formulated as a multi-objective *mixed integer optimal control problem*. The multi-objective problem was again solved using the ϵ-constraint method. The mixed integer optimal control problem was solved by extending the methods for solving mixed integer nonlinear optimization to handle dynamic systems.

Conclusions

Analyzing the interaction of design and control leads to a multi-objective optimization problem. The key issue in solving this problem is quantitatively determining the trade-offs between the steady-state economics and the dynamic controllability. By using multi-objective optimization techniques, these characteristics of the process can be traded off in a systematic manner.

By following the optimization approach to process synthesis, a mathematical framework can be developed. This involves developing a superstructure of design alternatives and effective mathematical models for the different criteria. The algorithmic procedure for solving the multi-objective problem involves the successive solution of scalar optimization problems to determine the noninferior solution set. The final step in the approach is to determine the best-compromise solution from those in the noninferior solution set.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Control Vector Iteration
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming

- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Optimal Control of a Flexible Arm
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Clark PA, Westerberg AW (1983) Optimization for design problems having more than one objective. Comput Chem Eng 7(4):259–278
2. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
3. Floudas CA (1995) Nonlinear and mixed integer optimization: Fundamentals and applications. Oxford Univ Press, Oxford
4. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10(4):237–260
5. Geoffrion AM, Dyer JS, Feinberg A (1972) An interactive approach for multi-criterion optimization with an application to the operation of an academic department. Managem Sci 19:357–368
6. Haimes Y, Hall WA, Freedman HT (1975) Multi-objective optimization in water resource systems: The surrogate worth trade-off method. Elsevier, Amsterdam
7. Hwang CL, Paidy SR, Yoon K, Masud ASM (1980) Mathematical programming with multiple objectives: A tutorial. Comput Oper Res 7:5–31
8. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flow sheets. Industr Eng Chem Res 26(9):1869
9. Lenhoff AM, Morari M (1982) Design of resilient processing plants I: Process design under consideration of dynamic effects. Chem Eng Sci 37(2):245–258
10. Loganathan GV, Sherali HD (1987) A convergent interactive cutting-plane algorithm for multiobjective optimization. Oper Res 35:365–377
11. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control-1. A multiobjective framework and application to binary distillation synthesis. Comput Chem Eng 18(10):933–969
12. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control-2. Reactor-separator-recycle system. Comput Chem Eng 18(10):971–994
13. Palazoglu A, Arkun Y (1986) A multiobjective approach to design chemical plants with robust dynamic operability characteristics. Comput Chem Eng 10(6):567–575
14. Paules IV GE, Floudas CA (1989) APROS: Algorithmic development methodology for discrete-continuous optimization problems. Oper Res 37(6):902–915
15. Schweiger CA, Floudas CA (1997) Interaction of design and control: Optimization with dynamic models. In: Hager WW, Pardalos PM (eds) Optimal Control: Theory, Algorithms, and Applications. Kluwer, Dordrecht, pp 388–435
16. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer approximation method for MINLP optimization. Comput Chem Eng 14(7):769–782
17. Zionts S (1979) Methods for solving management problems involving multiple objectives. Working Paper SUNY at Buffalo

Multi-objective Optimization: Interactive Methods for Preference Value Functions

HAROLD P. BENSON
Department Decision and Information Sci., University Florida, Gainesville, USA

MSC2000: 90C29

Article Outline

Keywords
See also
References

Keywords

Multi-objective optimization; Multiple criteria decision making; Interactive method; Preference value function; Value function

The *multi-objective optimization* (*multiple criteria decision making*) problem is the problem of choosing a most preferred solution when two or more incommensurate, conflicting objective functions (criteria) are to be simultaneously maximized. Interest in multi-objective optimization has risen sharply during the past 30 years. There are at least three reasons for this. First, and most importantly, is the increasing recognition that most applied problems in both the private and public sectors involve multiple objectives rather than one objective. Second, a variety of solution algorithms for multi-objective optimization are now available. Finally, the enormous improvements in the speed and storage of computers make it practical to apply these algorithms to the solution of realistically-sized problem applications.

Formally, the statement of the multi-objective optimization problem of interest here is

$$\text{(V)} \quad \begin{cases} \text{VMAX} & f(x) = [f_1(x), \ldots, f_p(x)], \\ \text{s.t.} & x \in X. \end{cases}$$

Here, $p \geq 2$, X is a nonempty subset of $\mathbf{R}^n$, each f_j, $j = 1, \ldots, p$, is a real-valued function defined on X or on some suitable set containing X, and VMAX indicates that, in some unspecified sense, we are to 'vector maximize' the vector $f(x)$ of *objective functions* (*criteria*) over X. The set X is called the set of *decision alternatives* or the *decision set*, and $\{f(x) \in \mathbf{R}^p : x \in X\}$, is called the *outcome set*.

There are a large number of diverse solution algorithms for problem (V). All are intended to help the *decision maker* (DM) find a most preferred solution to the problem. In the majority of these algorithms, the notion of efficiency plays an indispensable role. An *efficient* (*nondominated*, *noninferior*, *Pareto optimal*) solution for problem (V) is a solution $\overline{x} \in X$ such that there exists no other solution $x \in X$ that satisfies $f(x) \geq f(\overline{x})$ and $f(x) \neq f(\overline{x})$. Let X_E denote the set of efficient solutions for problem (V). Notice that if $\overline{x} \in X_E$, then there is no other feasible solution for problem (V) that achieves at least as large a value as $\overline{x}$ in each criterion of the problem and a strictly larger value than $\overline{x}$ in at least one criterion of the problem.

In the great majority of instances of problem (V), the *preference value function* (*value function*) v of the DM is unknown. This is a function $v: \mathbf{R}^p \to \mathbf{R}$ that maps the outcomes of problem **V** to real numbers in such a way that for any two outcomes y^1 and y^2, the DM prefers y^1 to y^2 if and only if $v(y^1) > v(y^2)$. Although v is unknown, what *is* known is that for each objective function f_j, the DM prefers more of f_j to less of f_j. Mathematically, this means that v is *coordinatewise increasing*, i. e., that whenever $\overline{z}, z \in \mathbb{R}^p$ satisfy $\overline{z} \geq z$ and $\overline{z}_j > z_j$ for some $j = 1, \ldots, p$, then $v(\overline{z}) > v(z)$. It is easy to show that when v is coordinatewise increasing, any maximizer x^* of $v[f(x)]$ over X must satisfy $x^* \in X_E$. In other words, as long as the DM prefers more to less, the search for a most preferred solution to problem (V) can be confined to X_E. This is one of the key reasons that the concept of efficiency is so important to the majority of the algorithms for problem (V).

The interactive methods constitute one of the most popular categories of algorithms for solving problem (V). An *interactive method* for problem (V) consists of a sequence of DM-computer interactions designed to create a sequence of decision alternatives that terminates with a most preferred solution to the problem. In a majority of cases, the generated alternatives are efficient. Each iteration of the interactive process consists of three steps. First, an initial solution is found with the aid of the computer. Typically, this solution is found by solving a single-objective optimization problem that generates either an efficient point or, at worst, a feasible point. Next, the DM is asked to react to the generated point by answering one or more questions involving his preferences for it. Last, based upon the answers given, the computer generates a new point, typically by modifying parameters in the single-objective optimization problem. This process continues until either the computer or the DM identifies a most preferred solution. The value function v of the DM is never needed and, in fact, is assumed to be unavailable.

There are several advantages to using interactive methods as compared to other categories of methods for problem (V). For instance, the preference information asked of the DM at each iteration is not difficult to supply. Furthermore, the DM thereby learns about his value function, which is often initially vague or mostly unknown. As the search continues, the DM also learns about the decision or efficient decision alternatives available and the trade-offs in the objective functions across these decision alternatives. The optimizations required of the computer are also usually

not difficult to perform. Finally, because the DM is highly involved in the process, his confidence in the most preferred solution that is eventually found is enhanced.

A frequent criticism of the interactive methods is that, in practice, the work required of the DM during the iterations seems to be burdensome for him in many cases. This may cause the DM to prematurely terminate the search so that a most preferred solution is not found.

There are literally hundreds of interactive algorithms for problem (V). Many are limited to cases where problem (V) is a multiple objective linear programming problem. Others apply when problem (V) is a multiple objective convex, nonlinear programming problem, a multiple objective integer programming problem, or some other type of multiple objective optimization problem. Instead of examining these algorithms individually, we will describe them by groups according to the characteristics that they possess.

One of the key characteristics of the interactive algorithms concerns the type of information required of the DM at each iteration. For instance, at each iteration, the DM may be asked to intuitively assign or reassign *weights* to the criteria according to his current assessment of their relative importance. R.E. Steuer [13] has shown some important stumbling blocks to this approach, however. Other algorithms may instead elicit *relaxation quantities* from the DM. In these cases, the DM is asked how much he would be willing to relax the level of one objective function in order to obtain possible improvements in the levels of other objective functions. Some of the oldest interactive algorithms use this approach [1,9]. Still other types of algorithms ask the DM various types of *trade-off questions*. The trade-off questions are designed to obtain an estimate of the gradient of the value function of the DM at the current solution. This approach is also relatively old, but difficult for the DM to accomplish [5,14]. Finally, a number of algorithms call for the DM to make *paired comparisons* at each iteration. In a paired comparison, the DM is given two solutions to compare and must give his preference for one or the other. Usually, the DM can accomplish this. But when the two solutions are quite similar, difficulties can arise [15]. In addition, algorithms that use paired comparisons can sometimes call for excessive numbers of these comparisons [12].

A second dimension where the interactive algorithms differ is in the approach used to explore the feasible region X or the efficient set X_E. Some algorithms use *feasible direction methods* [2]. In these algorithms, at each iteration, the direction to move from a point that was last found and the distance to move along the direction are determined with the aid of the DM. By moving along the direction by the specified amount, the next solution point is found. In many algorithms, all such points are efficient. In another group of algorithms, *feasible region reduction* is used to explore X or X_E. As points in X or in X_E are examined in these methods, portions of X are removed, usually via linear cuts. Another set of algorithms uses *weighting space reduction*. In these algorithms, a weighted sum of f_j, $j = 1, \ldots, p$, is maximized at each iteration, thereby yielding a point in X_E. Based upon the DM's responses to these maximizations, portions of the weighting space are removed. Eventually, the portion of the weighting space remaining is so small that the DM can pick out the set of weights associated with a most preferred solution.

Other approaches used to explore X or X_E include the *trade-off cutting plane* method [10], *Lagrange multiplier* methods, *visual interactive* methods (see, e. g. [7]), and the *branch and bound* method [8], among others. For further reading concerning these methods, see [3,4,6,11,12,13].

Another way to group the interactive algorithms for problem (V) is according to whether or not they handle inconsistencies in the DM's preference responses. As human beings, DM's are prone to giving preference responses over the course of the solution procedure that imply inconsistencies such as asymmetries or intransitivities of preference. Some algorithms take no account of these possible inconsistencies and have been criticized for this [12]. Others attempt to reduce inconsistency by either minimizing the DM's cognitive burden or by incorporating tests for inconsistency that are used as the interactive solution process proceeds.

W.S. Shin and A. Ravindran [12] have compared various of the classes of interactive algorithms according to four criteria that are important in practice. These criteria are the DM's cognitive burden, the ease with which the single-objective optimizations called for can be used, implemented and solved, the handling of inconsistency, and the overall quality of the solution process and the answers obtained. Although preliminary,

these comparisons seem to show the relative superiority of the weighting space reduction and other criterion weight space search methods, and of the visual interactive methods. Readers should note, however, that the rankings in the study are subjectively-obtained by the authors [7].

For further general reading on interactive methods, see [2,3,4,6,11,12,13,14].

See also

- Bi-objective Assignment Problem
- Decision Support Systems with Multiple Criteria
- Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- Financial Applications of Multicriteria Analysis
- Fuzzy Multi-objective Linear Programming
- Multicriteria Sorting Methods
- Multi-objective Combinatorial Optimization
- Multi-objective Integer Linear Programming
- Multi-objective Optimization and Decision Support Systems
- Multi-objective Optimization: Interaction of Design and Control
- Multi-objective Optimization: Lagrange Duality
- Multi-objective Optimization: Pareto Optimal Solutions, Properties
- Multiple Objective Programming Support
- Outranking Methods
- Portfolio Selection and Multicriteria Analysis
- Preference Disaggregation
- Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- Preference Modeling

References

1. Benayoun R, Montgolfier J, Tergny J, Laritchev O (1971) Linear programming with multiple objective functions: Step method (STEM). Math Program 1:366–375
2. Benson HP, Aksoy Y (1991) Using efficient feasible directions in interactive multiple objective linear programming. Oper Res Lett 10:203–209
3. Buchanan JT, Daellenbach HG (1987) A comparative evaluation of interactive solution methods for multiple objective decision models. Europ J Oper Res 29:353–359
4. Evans GW (1984) An overview of techniques for solving multiobjective mathematical programs. Managem Sci 30:1268–1282
5. Geoffrion AM, Dyer JS, Feinberg A (1972) An interactive approach for multicriterion optimization, with an application to the operation of an academic department. Managem Sci 19:357–368
6. Goicoechea A, Hansen DR, Duckstein L (1982) Multiobjective decision analysis with engineering and business applications. Wiley, New York
7. Kohornen P, Laakso J (1986) A visual interactive method for solving the multiple criteria problem. Europ J Oper Res 24:277–287
8. Marcotte O, Soland R (1985) An interactive branch-and-bound algorithm for multiple criteria optimization. Managem Sci 32:61–75
9. Monarchi DE, Kisiel CC, Duckstein L (1973) Interactive multiobjective programming in water resources: A case study. Water Resources Res 9:837–850
10. Musselman K, Talavage J (1980) A tradeoff cut approach to multiple objective optimization. Oper Res 28:1424–1435
11. Rosenthal RE (1985) Concepts, theory and techniques: Principles of multiobjective optimization. Decision Sci 16:133–152
12. Shin WS, Ravindran A (1991) Interactive multiple objective optimization. Survey I: Continuous case. Comput Oper Res 18:97–114
13. Steuer RE (1986) Multiple criteria optimization: Theory, computation, and application. Wiley, New York
14. Wallenius J (1975) Comparative evaluation of some interactive approaches to multicriterion optimization. Managem Sci 21:1387–1396
15. Zionts S, Wallenius J (1983) An interactive multiple objective linear programming method for a class of underlying nonlinear utility functions. Managem Sci 29:519–529

Multi-objective Optimization: Lagrange Duality

HIROTAKA NAKAYAMA
Department Applied Math., Konan University, Kobe, Japan

MSC2000: 90C29, 90C30

Article Outline

Keywords
Linear Cases
Nonlinear Cases
Geometric Duality
Duality for Weak Efficiency
See also
References

Keywords

Vector inequality; Efficient point; Vector valued Lagrangian

As is well known, duality in mathematical programming is based on the property that any closed convex set can be also represented by the intersection of closed half spaces including it. Let the multi-objective optimization problem to be considered here be given by

$$(\mathrm{P}) \quad \begin{cases} \min & f(x) := (f_1(x), \ldots, f_p(x)) \\ \text{over} & x \in X, \end{cases}$$

where

$$X = \left\{ x \in X' : \begin{array}{l} g_i(x) \leqq 0, \\ i = 1, \ldots, m, X' \subset \mathbb{R}^n \end{array} \right\}.$$

Note here that vector inequalities are commonly used: for any n-vectors a and b, $a > b$ means $a_i > b_i$ $(i = 1, \ldots, n)$. Also, $a \geqq b$ means $a_i \geqq b_i$ $(i = 1, \ldots, n)$. On the other hand, $a \geq b$ means $a \geqq b$ but $a \neq b$. Hereafter, vector inequalities such as $g(x) \leqq 0$ will be used instead of $g_i(x) \leqq 0$ $(i = 1, \ldots, m)$.

Defining a dual problem (D) in some appropriate way associated with the problem (P), our aim is to show the property min(P) = max(D). Here min(P) denotes the set of efficient points of the problem (P) in the objective function space $\mathbf{R}^p$, and similarly max(D) the one of the dual problem (D).

Unlike the usual mathematical programming, the optimal value of the primal problem (and the dual problem) are not necessarily determined uniquely in multi-objective optimization. Hence, there have been developed several kinds of formulation of dual problem in order to get the desirable property min(P) = max(D). Regarding Lagrange duality, three typical dualizations can be seen in linear cases, nonlinear cases and geometric approaches [6].

Linear Cases

The first result on duality for multi-objective optimization seems the one given in [1] for linear cases. This is formulated as a matrix optimization including the vector optimization as a special case. Although there have been several related works, the probably most attractive one is given in [2] because it is formulated as a natural extension of traditional linear programming: Let A be an $m \times n$ matrix, C a $p \times n$ matrix, and b an m-vector. Then the *primal problem* (P) in linear cases is formulated as

$$(\mathrm{P_I}) \quad \begin{cases} \min & Cx \\ \text{s.t.} & Ax \geqq b \\ & x \geqq 0. \end{cases}$$

Associated with $(\mathrm{P_I})$, H. Iserman [2] defined the *dual problem* as

$$(\mathrm{D_I}) \quad \begin{cases} \max & \Lambda b \\ \text{s.t.} & \Lambda A \ngeqq C \\ & \Lambda \geqq 0. \end{cases}$$

Here, the multiplier $\Lambda \geqq 0$ is a $p \times m$ matrix whose elements are all nonnegative.

Then Isermann's duality is given by

i) $\Lambda\, b \ngeqq Cx$ for all feasible x and Λ.

ii) Suppose that $\overline{\Lambda} b = C\overline{x}$ for some feasible $\overline{x}$ and some feasible $\overline{\Lambda}$. Then $\overline{\Lambda}$ is an efficient solution to $(\mathrm{D_I})$ and $\overline{x}$ is an efficient solution to $(\mathrm{P_I})$.

iii) $\min(\mathrm{P_I}) = \max(\mathrm{D_I})$.

Nonlinear Cases

The most natural dualization in nonlinear multi-objective optimization seems to be the one given in [10].

Consider the problem (P), and assume the following:

i) X' is a nonempty compact convex set.

ii) f is continuous, and $f(X) + \mathbf{R}^n_+$ is convex in $\mathbf{R}^p$.

iii) $g_i (i = 1, \ldots, m)$ are continuous and convex.

Under these assumptions, it can be readily shown that for every $u \in \mathbf{R}^m$, both sets $X(u) = \{x \in X' : g(x) \leqq u\}$ and $Y(u) = f[X(u)] = \{y \in \mathbb{R}^p : y = f(x), x \in X', g(x) \leqq u\}$ are compact and convex.

The primal problem (P) can be embedded as $(\mathrm{P_0})$ in a family of perturbed problems (P_u) given by

$$(\mathrm{P_u}) \quad \min Y(u).$$

Defining $\Gamma = \{u \in \mathbf{R}^m : X(u) \neq \emptyset\}$, the set Γ is convex. Now in a similar fashion to the ordinary mathematical programming, the perturbed map can be defined by

$$W(u) = \min \{ f(x) : \ x \in X', \ g(x) \leqq u \}.$$

It is known that for every $u \in \Gamma$, $W(u) + \mathbf{R}^p_+$ is convex and

$$W(u) + \mathbb{R}^p_+ = Y(u) + \mathbb{R}^p_+.$$

In addition, the map W is monotone and convex on Γ.

Now, define the vector valued Lagrangian function with a $p \times m$ matrix multiplier Λ as

$$L(x, \Lambda) = f(x) + \Lambda g(x).$$

Associated with this definition, the dual map can be defined as

$$\Phi(\Lambda) = \min \Omega(\Lambda),$$

where

$$\Omega(\Lambda) = \left\{L(x, \Lambda):\ x \in X'\right\}.$$

Under the terminology, the dual problem associated with the primal problem (P) can be given by

$$(\mathrm{D_{TS}}) \quad \max \bigcup_{\Lambda \in \mathcal{L}} \Phi(\Lambda).$$

It can be shown that Φ is concave point-to-set map on Γ, namely

$$\Phi(\alpha\Lambda^1 + (1-\alpha)\Lambda^2) \subset \alpha\Phi(\Lambda^1) + (1-\alpha)\Phi(\Lambda^2) + \mathbb{R}^p_+$$

and $\Phi(\Lambda) + \mathbf{R}^p_+$ is a convex set in $\mathbf{R}^p$ for each $\Lambda \in \mathcal{L}$. Here $\mathcal{L}$ is the set of all $p \times m$ matrices whose components are all positive.

T. Tanino and Y. Sawaragi [10] presented the following as duality in multi-objective optimization:

Theorem 1

i) For any $x \in X$ and $y \in \Phi(\Lambda)$

$$y \not\geqq f(x).$$

ii) Suppose that $\widehat{x} \in X$, $\widehat{\Lambda} \in \mathcal{L}$ and $f(\widehat{x}) \in \Phi(\widehat{\Lambda})$. Then $\widehat{y} = f(\widehat{x})$ is an efficient point to the primal problem (P) and also to the dual problem (D_{TS}).

iii) Suppose that any efficient solutions to (P) are all proper and that Slater's constraint qualification is satisfied. Then

$$\min (P) \subset \max (D_{TS}).$$

Remark 2 The above theorem is not complete in the sense that the relation min(P) = max(D) does not hold. Regarding conjugate duality, there have been reports presenting w-min(P) = w-max(D) (see, e. g., [4] and [9]). Several studies based on geometric consideration have been made for deriving the relation min(P) = max(D) using vector valued Lagrangian. This will be stated in the following

Geometric Duality

Geometric considerations are made in [3], based on the supporting hyperplanes for epi$W$, and in [5], based on the supporting conical varieties for epi$W$, which is denoted by G here.

Define

$$G = \left\{(u, y) \in \mathbb{R}^p \times \mathbb{R}^p: \begin{array}{l} y \geqq f(x), \\ u \geqq g(x) \\ \text{for some } x \in X' \end{array}\right\},$$

$$Y_G = \left\{y:\ (0, y) \in G, 0 \in \mathbb{R}^m,\ y \in \mathbb{R}^p\right\}.$$

Associates with the primal problem (P), we consider the following two kinds of dual problems:

$$(\mathrm{D_N}) \quad \max \bigcup_{\Lambda \in \mathcal{L}} Y_{S(\Lambda)},$$

where

$$Y_{S(\Lambda)} = \left\{y \in \mathbb{R}^p:\ f(x) + \Lambda g(x) \not\leqq y,\ \forall x \in X'\right\}$$

and

$$(\mathrm{D_J}) \quad \bigcup_{\substack{\mu > 0 \\ \lambda \geqq 0}} Y_{H^-(\lambda, \mu)},$$

where

$$Y_{H^-(\lambda,\mu)} = \left\{y \in \mathbb{R}^p:\ \begin{array}{l} \langle \mu, f(x)\rangle + \langle \lambda, g(x)\rangle \geqq \langle \mu, y\rangle \\ \forall x \in X' \end{array}\right\}.$$

Theorem 3

i) For any feasible x in (P) and for any feasible y in (D_N) or (D_J),

$$y \not\geqq f(x).$$

ii) Assume that G is closed, that there exists at least an efficient solution to the primal problem, and that

these solutions are all proper. Then, under the condition of Slater's constraint qualification, the following holds:

$$\min(P) = \max(D_N) = \max(D_J).$$

Remark 4 In the above duality, we assumed that the convex set G is closed and that Slater's constraint qualification is satisfied, which seem relatively restrictive. Instead of these conditions, J. Jahn [3] assumed that Y_G is closed and some normality condition.

Define

$$A_{G(\mu)} = \{\alpha\colon\ (0,\alpha) \in G(\mu),\ 0 \in \mathbb{R}^m,\ \alpha \in \mathbb{R}^1\}$$
$$Y_G = \{y\colon\ (0,y) \in G,\ 0 \in \mathbb{R}^m,\ y \in \mathbb{R}^m\}.$$

Definition 5 The primal problem (P) is said to be *J-normal*, if for every $\mu > 0$

$$\mathrm{cl}(A_{G(\mu)}) = A_{\mathrm{cl}\,G(\mu)}.$$

The primal problem (P) is said to be *J-stable*, if it is J-normal and for an arbitrary $\mu > 0$ the problem

$$\sup_{\lambda \geqq 0} \inf_{x \in X} \langle \mu, f(x)\rangle + \langle \lambda, g(x)\rangle$$

has at least one solution.

On the other hand, J.W. Nieuwenhuis [7] suggested another normality condition:

Definition 6 The primal problem (P) is said to be *N-normal*, if

$$\mathrm{cl}\, Y_G = Y_{\mathrm{cl}\,G}.$$

Lemma 7 *Slater's constraint qualification* ($\exists \widehat{x}$, $g(\widehat{x}) > 0$) *yields J-stability and N-normality.*

Theorem 8 *Suppose that Y_G is closed, $\min_D(P) \neq \emptyset$, and the efficient solutions to (P) are all proper. Then, under the condition of J-stability,*

$$\min(P) = \max(D_N) = \max(D_J).$$

Duality for Weak Efficiency

Define

$$Y_{S'(\Lambda)} = \{y \in \mathbb{R}^p\colon\ f(x) + \Lambda g(x) \not< y,\ \forall x \in X'\}.$$

Theorem 9 *Suppose that Y_G is a nonempty subset in $\boldsymbol{R}^p$ and $Y_G + \boldsymbol{R}^p_+$ is bounded. Then under the condition of N-normality*

$$w\text{-}\min \mathrm{cl}\, Y_G = w\text{-}\max \mathrm{cl} \bigcup_{\Lambda \in \mathcal{L}} Y_{S'(\Lambda)}$$
$$= w\text{-}\max \mathrm{cl} \bigcup_{\substack{\mu \in \mathbb{R}^p_+ \setminus \{0\} \\ \lambda \geqq 0}} Y_{H^-(\lambda,\mu)}.$$

Remark 10 As can be readily seen, by defining $\inf A$, for a set $A \in \mathbf{R}^p$, as essentially $\min \mathrm{cl}(A + \mathbf{R}^p_+)$ and similarly $\sup A$ as essentially $\min \mathrm{cl}(A - \mathbf{R}^p_+)$, we can have $\inf(\mathrm{P}) = \sup(\mathrm{D_{TS}}) = \sup(\mathrm{D_N}) = \sup(\mathrm{D_J})$ under some appropriate stability condition [9].

See also

- Bi-objective Assignment Problem
- Decision Support Systems with Multiple Criteria
- Decomposition Techniques for MILP: Lagrangian Relaxation
- Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- Financial Applications of Multicriteria Analysis
- Fuzzy Multi-objective Linear Programming
- Integer Programming: Lagrangian Relaxation
- Lagrange, Joseph-Louis
- Lagrangian Multipliers Methods for Convex Programming
- Multicriteria Sorting Methods
- Multi-objective Combinatorial Optimization
- Multi-objective Integer Linear Programming
- Multi-objective Optimization and Decision Support Systems
- Multi-objective Optimization: Interaction of Design and Control
- Multi-objective Optimization: Interactive Methods for Preference Value Functions
- Multi-objective Optimization: Pareto Optimal Solutions, Properties
- Multiple Objective Programming Support
- Outranking Methods
- Portfolio Selection and Multicriteria Analysis
- Preference Disaggregation
- Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- Preference Modeling

References

1. Gale D, Kuhn HW, Tucker AW (1951) Linear programming and the theory of games. In: Koopmans TC (ed) Activity Analysis of Production and Allocation. Wiley, New York, pp 317–329
2. Isermann H (1978) On some relations between a dual pair of multiple objective linear programs. Z Oper Res 22:33–41
3. Jahn J (1983) Duality in vector optimization. Math Program 25:343–353
4. Kawasaki H (1982) A duality theorem in multiobjective nonlinear prgramming. Math Oper Res 7:95–110
5. Nakayama H (1984) Geometric consideration of duality in vector optimization. J Optim Th Appl 44:625–655
6. Nakayama H (1999) Duality in multi-objective optimization. In: Gal T, Stewart TJ, Hanne T (eds) Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory and Applications. Kluwer, Dordrecht, pp 3.1–3.29
7. Nieuwenhuis JW (1980) Supremal points and generalized duality. Math Operationsforsch Statist Ser Optim 11:41–59
8. Sawaragi Y, Nakayama H, Tanino T (1985) Theory of multiobjective optimization. Acad Press, New York
9. Tanino T (1988) Supremum of a set in a multi-dimensional space. J Math Anal Appl 130:386–397
10. Tanino T, Sawaragi Y (1979) Duality theory in multiobjective programming. J Optim Th Appl 27:509–529

Multi-objective Optimization: Pareto Optimal Solutions, Properties

HAROLD P. BENSON
Department Decision and Information Sci.,
University Florida, Gainesville, USA

MSC2000: 90C29

Article Outline

Keywords
See also
References

Keywords

Multi-objective optimization; Multiple criteria decision making; Efficient solution; Pareto optimal solution; Noninferior solution; Nondominated solution; Weakly efficient solution; Weakly Pareto optimal solution; Weakly noninferior solution; Weakly nondominated solution; Properly efficient solution

The *multi-objective optimization* (*multiple criteria decision making*) problem is the problem of choosing a most preferred solution when two or more incommensurate, conflicting objective functions (criteria) are to be simultaneously maximized. A central difficulty in such problems is that, unlike in single objective maximization problems, there is no obvious or simple way to define the concept of a most preferred solution. Nevertheless, because the applications of multi-objective optimization abound, there has been great interest during the past 30 years in seeking appropriate definitions for a most preferred solution and in developing algorithms that aid the decision maker (DM) to find such a solution. These applications are in a wide variety of areas, including, for example, production planning, finance, environmental conservation, academic planning, nutrition planning, advertising, facility location, auditing, blending techniques, transportation planning, and scheduling, to name just a few.

There are several alternate mathematical formulations of the multi-objective optimization problem [13]. For purposes of modeling the deterministic multiple objective optimization problems found in management science/operations research, however, the most popular form of the problem is denoted

$$(\mathrm{V}) \quad \begin{cases} \mathrm{VMAX} & [f_1(x), \ldots, f_p(x)] \\ \text{s.t.} & x \in X. \end{cases}$$

Here, $p \geq 2$, X is a nonempty subset of $\mathbf{R}^n$, each f_j, $j = 1, \ldots, p$, is a real-valued function defined on X or on a suitable set containing X, and VMAX indicates that we are to, in some as-yet unspecified sense, 'vector maximize' the vector

$$f(x) = [f_1(x), \ldots, f_p(x)]$$

of *objective functions* (*criteria*) over X. The set X is called the set of *alternatives* or the *decision set*.

Of all of the solution concepts proposed for helping the DM find a most preferred solution for problem (V), the concept of efficiency has proven to be of overriding importance. An *efficient* (*Pareto optimal*, *noninferior*, *nondominated*) *solution* for problem (V) is a point $\overline{x} \in X$ such that there exists no other point $x \in X$ that satisfies $f(x) \geq f(\overline{x})$ and $f(x) \neq f(\overline{x})$. Letting X_E denote the set of all efficient points for problem (V), we see that whenever $\overline{x} \in X_E$, there is no other feasible

point that does at least as well as $\overline{x}$ in all of the criteria for problem (V) and strictly better in at least one criterion. A point $\overline{x} \in X$ is called *dominated* when for some other point $x \in X$, $f(x) \geq f(\overline{x})$ and, for at least one $j = 1, \ldots, p$, $f_j(x) > f_j(\overline{x})$. Thus, we have the alternate definition for efficiency that states that a point $\overline{x}$ is an *efficient* solution for problem (V) when $\overline{x} \in X$ and there are no other points in X that dominate $\overline{x}$.

One of the reasons for the fundamental importance of the efficiency concept is that it has proven to be highly useful in a variety of algorithms for problem (V). Among these algorithms are the *satisficing* methods, *compromise programming*, most *interactive methods*, and the *vector maximization method*. The latter method, for instance, seeks to generate either all of X_E or key parts of X_E. The generated set is shown to the DM. Then, based upon the DM's internal utility (or value) function, the DM chooses from the generated set a most preferred solution. For details concerning these methods for problem (V), see [7,10,12,13,14].

In some cases, it is useful to consider a slightly relaxed concept of efficiency called weak efficiency. A point $\overline{x} \in X$ is called a *weakly efficient* (*weakly Pareto optimal*, *weakly noninferior*, *weakly nondominated*) *solution* for problem (V) when there is no other point $x \in X$ such that $f(x) > f(\overline{x})$. Let X_{WE} denote the set of all weakly efficient points for problem (V). Notice that X_E is a subset of X_{WE}. In some cases of problem (V), such as when the objective functions are ratios of linear functions, it is easier to analyze and generate points in X_{WE} than points in X_E.

Let U represent a utility function defined on the space $\mathbf{R}^p$ of the objective functions of problem (V). Suppose that U is *coordinatewise increasing*, i. e., that whenever $\overline{z}, z \in \mathbb{R}^p$ satisfy $\overline{z} \geq z$ and $\overline{z}_j > z_j$ for some $j = 1, \ldots, p$, then $U(\overline{z}) > U(z)$. Suppose that x^* is an optimal solution to the single objective problem

$$\text{(S)} \quad \max_{x \in X} U[f_1(x), \ldots, f_p(x)].$$

Then x^* must be an efficient solution for problem (V) (cf. [11]).

The property in the previous paragraph explains to a great extent why the concept of efficiency is of such fundamental value. The assumption that the utility function U in the above paragraph is coordinatewise increasing implies that in problem (S), for each $j = 1, \ldots, p$, more of f_j is preferred to less of f_j. Thus, if we imagine that U is the utility (or value) function of the DM over the objective function space of problem (V), then the previous paragraph implies that whenever the DM prefers more to less in each objective function of problem (V), any point that maximizes the DM's utility for $f(x)$ over X must be an efficient point in problem (V). In short, as long as we know that the DM prefers more to less, we can confine the search for a most preferred solution to X_E. Although the utility function of the DM is generally not actually available, in virtually all applications the DM does, indeed, prefer more to less in each objective function of problem (V). Thus, in essentially all cases, any most preferred solution for problem (V) will be found in X_E.

Because of the central importance of efficiency, a great deal of effort has been made by researchers to delineate the properties of the efficient points and of the efficient set for problem (V). In what follows, we shall briefly highlight some of the most important of these properties.

Consider the single-objective optimization problem

$$\text{(W)} \quad \begin{cases} \max & \sum_{j=1}^{p} w_j f_j(x), \\ \text{s.t.} & x \in X. \end{cases}$$

Here, w_j, $j = 1, \ldots, p$, are parameters, which are often thought of as weights associated with the objective functions f_j, $j = 1, \ldots, p$, of problem (V). A number of so-called *scalarization properties* for efficient points of problem (V) are expressed in terms of problem (W). To present some of these, another efficiency concept, called proper efficiency, is needed. A point x° is said to be a *properly efficient solution* for problem (V) when $x^\circ \in X_E$ and, for some sufficiently large number M, whenever $f_i(x) > f_i(x^\circ)$ for some $i = 1, \ldots, p$ and some $x \in X$, there exists some $j = 1, \ldots, p$ such that $f_j(x) < f_j(x^\circ)$ and

$$\frac{f_i(x) - f_i(x^\circ)}{f_j(x^\circ) - f_j(x)} \leq M.$$

In words, for each properly efficient solution of problem (V), for each criterion, the possible marginal gains in that criterion relative to the losses in the criteria that have losses *cannot all* be unbounded from above. Let

X_{PRE} denote the set of properly efficient solutions for problem (V), and let $w^{\mathsf{T}} = (w_1, \ldots, w_p)$. Then some key scalarization properties are as follows.

1) If $\overline{x}$ is the unique optimal solution to problem (W) for some $w \geq 0$, $w \neq 0$, then $\overline{x} \in X_E$.
2) If $\overline{x}$ is an optimal solution to problem (W) for some $w \geq 0$, $w \neq 0$, then $\overline{x} \in X_{WE}$.
3) Assume that for each $j = 1, \ldots, p$, f_j is a concave function on the convex set X. Then $\overline{x} \in X_{PRE}$ if and only if $\overline{x}$ is an optimal solution to problem (W) for some $w > 0$.
4) Under the assumptions in property 3), $\overline{x} \in X_{WE}$ if and only if $\overline{x}$ is an optimal solution to problem (W) for some $w \geq 0$, $w \neq 0$.
5) Under the assumptions of property 3), if $\overline{x} \in X_E$ but $\overline{x} \notin X_{PRE}$, then there exists a $w \geq 0$, $w \neq 0$ with $w_j = 0$ for at least one $j = 1, \ldots, p$ such that $\overline{x}$ is an optimal solution to problem (W).
6) If each f_j, $j = 1, \ldots, p$, is a linear function and X is a polyhedron, $X_{PRE} = X_E$.

The scalarization properties can be used for various purposes, including the generation of points in X_E, X_{WE} and X_{PRE}. For instance, when each $f_j, j = 1, \ldots, p$, is a linear function and X is a polyhedron, from properties 3) and 6), points in X_E, including, at least potentially, all of X_E, can be generated by solving problem (W) as the parameter $w > 0$ is varied. Under the assumptions of property 3), the same process will generate points in X_{PRE}, including, at least potentially, all of X_{PRE}. However, from properties 3)–5), it is apparent that no such simple process for generating X_E exists, even under the assumptions of property 3). This is another motivation for the proper efficiency concept.

Another important issue in efficiency concerns testing. One may want to test a given point for efficiency in problem (V), and one may want to test whether X_E and X_{PRE} are empty or not. We will present several of the properties of efficiency that provide some of the theory for these tests. These properties all utilize the single-objective problem

$$(\mathrm{T}) \quad \begin{cases} \max & \displaystyle\sum_{j=1}^{p} f_j(x), \\ \text{s.t.} & f_j(x) \geq f_j(x^\circ), \\ & j = 1, \ldots, p, \\ & x \in X. \end{cases}$$

Here, x° is an arbitrary element of $\mathbf{R}^n$. The properties are as follows.

7) The point $x^\circ \in \mathbf{R}^n$ belongs to X_E if and only if x° is an optimal solution to problem (T).
8) Suppose that $x^\circ \in X$ in problem (T), and that problem (T) has no finite maximum value. Then $X_{PRE} = \emptyset$ [1].
9) Suppose that the assumptions of property 3) hold, that $x^\circ \in X$ in problem (T), and that problem (T) has no finite maximum value. Then, if the set
$$Z = \{z \in \mathbb{R}^p : \ z \leq f(x) \text{ for some } x \in X\}$$
is closed, $X_E = \emptyset$.
10) Assume that each $f_j, j = 1, \ldots, p$, is a linear function and that X is a polyhedron. Suppose that $x^\circ \in X$ in problem (T), and that problem (T) has no finite maximum value. Then $X_E = \emptyset$.
11) Any optimal solution to problem (T) belongs to X_E.

Notice from these properties that solving problem (T) is a useful tool for both testing a point for efficiency and for investigating the issues of whether X_E and X_{PRE} are empty or not. In the case of testing a point x° for efficiency, property 7) shows that problem (T) can be used to obtain a definitive answer, i. e., using property 7), we will always detect whether or not $x^\circ \in X_E$. Furthermore, when property 7) shows that $x^\circ \notin X_E$, but problem (T) has an optimal solution x^*, then, by property 11), $x^* \in X_E$. Notice also that in this case, x^* dominates x°.

In the case of investigating whether or not X_E and X_{PRE} are empty, however, definitive answers cannot usually be obtained by using these properties. This is because none of the properties addresses the issue of whether or not X_E and X_{PRE} are empty when, instead of having an optimal solution or having no finite maximum value, problem (T) has a finite but unattained maximum value. The one case where the properties *can* be used to definitely detect whether or not X_E and X_{PRE} are empty is the case where the objective functions of problem (V) are all linear and X is a polyhedron. In that case, problem (T) cannot have a finite but unattained maximum value. Therefore, properties 7), 10) and 11) can be used to detect whether or not $X_E = X_{PRE}$ is empty in such cases.

One of the main challenges computationally to generating all or parts of X_E or X_{WE} for the DM to consider

is that both X_E and X_{WE} are, except for trivial cases, *nonconvex sets*. Although some researchers have suggested ways to mitigate this problem [5], it generally remains a major stumbling block for algorithm development. In many common cases, however, X_E or X_{WE} possesses a useful, although less valuable, property than convexity upon which algorithms can be based. This property is called *connectedness*. In particular, a set $Z \subseteq \mathbf{R}^n$ is *connected* if, whenever A and B are nonempty subsets of $\mathbf{R}^n$ such that A has no points in common with the closure of B, and B has no points in common with the closure of A, $Z \neq A \cup B$. Some common cases of problem (V) where X_E or X_{WE} is connected are given in the following properties.

12) Assume that for each $j = 1, \ldots, p$, f_j is a quasiconcave function on X, and that X is a compact convex set. Then X_{WE} is connected.
13) Assume that for each $j = 1, \ldots, p$, f_j is a concave function on $\mathbf{R}^n$, and that X is a compact convex set. Then X_E is connected.

Recall that a concave function on a convex set is also quasiconcave on the set. Therefore, from property 12), it follows that X_{WE} is connected when each objective function in problem (V) is a concave function on X, and X is a compact, convex set.

There are a variety of other properties of efficient points and of the efficient set for problem (V). These include, for instance, density properties, stability-related properties, the *domination property* [2,3,8], and complete efficiency-related properties [4,6]. For further reading, see [5,7,9,10,12,13,14].

See also

- Bi-objective Assignment Problem
- Decision Support Systems with Multiple Criteria
- Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- Financial Applications of Multicriteria Analysis
- Fuzzy Multi-objective Linear Programming
- Multicriteria Sorting Methods
- Multi-objective Combinatorial Optimization
- Multi-objective Integer Linear Programming
- Multi-objective Optimization and Decision Support Systems
- Multi-objective Optimization: Interaction of Design and Control
- Multi-objective Optimization: Interactive Methods for Preference Value Functions
- Multi-objective Optimization: Lagrange Duality
- Multiple Objective Programming Support
- Outranking Methods
- Portfolio Selection and Multicriteria Analysis
- Preference Disaggregation
- Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- Preference Modeling

References

1. Benson HP (1979) An improved definition of proper efficiency for vector maximization with respect to cones. J Math Anal Appl 71:232–241
2. Benson HP (1983) On a domination property for vector maximization with respect to cones. J Optim Th Appl 39:125–132
3. Benson HP (1984) Errata corrige. J Optim Th Appl 43:477–479
4. Benson HP (1991) Complete efficiency and the initialization of algorithms for multiple objective programming. Oper Res Lett 10:481–487
5. Benson HP, Sayin S (1997) Towards finding global representations of efficient sets in multiple objective mathematical programming. Naval Res Logist 44:47–67
6. Benveniste M (1977) Testing for complete efficiency in a vector maximization problem. Math Program 12:285–288
7. Goicoechea A, Hansen DR, Duckstein L (1982) Multiobjective decision analysis with engineering and business applications. Wiley, New York
8. Henig MI (1986) The domination property in multicriteria optimization. J Math Anal Appl 114:7–16
9. Luc DT (1989) Theory of vector optimization. Springer, Berlin
10. Sawaragi Y, Nakayama H, Tanino T (1985) Theory of multiobjective optimization. Acad Press, New York
11. Soland RM (1979) Multicriteria optimization: A general characterization of efficient solutions. Decision Sci 10:26–38
12. Steuer R (1986) Multiple criteria optimization: Theory, computation and application. Wiley, New York
13. Yu PL (1985) Multiple-criteria decision making. Plenum, New York
14. Zeleny M (1982) Multiple criteria decision making. McGraw-Hill, New York

Multiparametric Linear Programming

VIVEK DUA, EFSTRATIOS N. PISTIKOPOULOS
Imperial College, London, UK

MSC2000: 90C31, 90C05

Article Outline

Keywords
See also
References

Keywords

Sensitivity analysis with respect to right-hand side changes; Critical region

In this article we will describe some results for sensitivity analysis and parametric programming for linear models. The solution approach that is described here is based upon the extension of simplex algorithm for linear programs (LP) [3,5]. Here we mention some references ([1,2,6,7,8,9,10,11,12,13,14,15,16,18,19,20,21], and [17]); however [3] is recommended for an extensive list of references and [4] for a historical outline on parametric linear programming.

We will consider right-hand side (RHS) multiparametric linear programming problems, where uncertain parameters are assumed to be bounded in a convex region. The solution algorithm is based upon characterizing the given initial convex region by a number of nonoverlapping smaller convex regions and obtaining optimal solutions associated with each of these regions. The basic assumptions for the application of the algorithm are:

- The given region must be finite and connected.
- One should be able to characterize at least one (smaller) region.
- One should be able to identify all regions that are adjacent to a given region.

Consider the following multiparametric linear programming problem, when parameters are present on the right-hand side of the constraints:

$$\begin{cases} z(\theta) = \min\limits_{x} & c^\top x \\ \text{s.t.} & Ax = b + F\theta \\ & x \geq 0 \\ & x \in \mathbb{R}^n, \quad \theta \in \mathbb{R}^s, \end{cases} \tag{1}$$

where x is a vector of continuous variables; A and F are constant matrices, and c and b are constant vectors of appropriate dimensions; θ is a vector of uncertain parameters, such that for each $\theta \in K$, $\theta \in \mathbf{R}^s$, (1) has a finite optimal solution, and has no optimal solution for $\theta \in \mathbf{R}^s - K$. Further, consider the following restriction on $\theta \in \Xi$, $\Xi = \{\theta : G\theta \leq g\}$, where G is a constant matrix and g is a constant vector; see Fig. 1 for a graphical interpretation for the two parametric case where θ is bounded in the region given by PQRST.

The simplex tableau associated with (1) is given as follows:

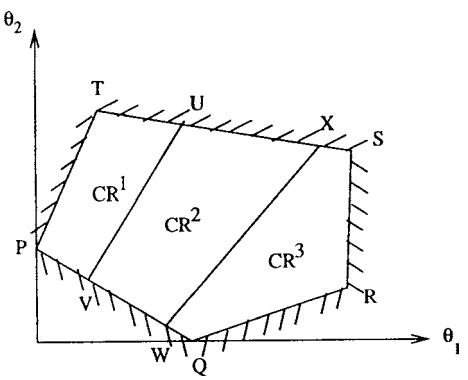
$$Yx - {}^{\rho}F\theta = x_B,$$
$$z + {}^{\rho}z^\top x - f_{m+1}\theta = z^{(\rho)},$$

where

$$\begin{aligned} &Y = B^{-1}A, \quad {}^{\rho}F = B^{-1}F, \quad x_B = B^{-1}b, \\ &z = c^\top x, \quad {}^{\rho}z = c_B^\top Y - c^\top, \\ &f_{m+1}^\top = c_B^\top {}^{\rho}F, \quad z^{(\rho)} = c_B^\top x_B, \end{aligned} \tag{2}$$

where ρ corresponds to the index of basic variables and B is the corresponding matrix. The (critical) region within which the above (optimal) tableau is valid can then be derived as follows. The *critical region*, CR, where an optimal solution, $z^{(\rho)}(\theta) = c_B^\top x_B(\theta)$, preserves its optimality, is given by the initial conditions on θ:

$$G\theta \leq g \tag{3}$$



Multiparametric Linear Programming, Figure 1
Definition of critical regions

together with the conditions of *primal feasibility*. The conditions of primal feasibility are derived as follows. The basis B is said to be *primal feasible* if the condition:

$$B^{-1}b(\theta) = x_B(\theta) \geq 0, \tag{4}$$

where $b(\theta) = b + F\theta$ and $x_B(\theta) = x_B + {}^{\rho}F\theta$, is satisfied. Then using (2) and (4), the condition of primal feasibility is given by:

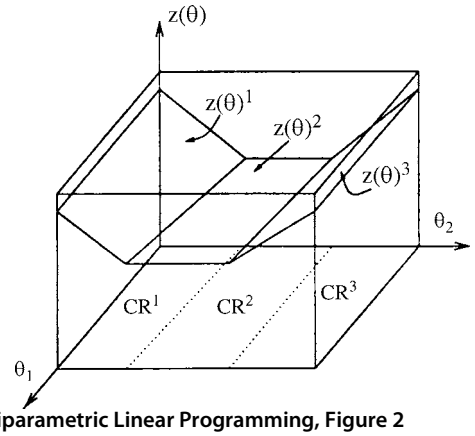
$$-{}^{\rho}F\theta \leq x_B. \tag{5}$$

Thus, the critical region corresponding to ρ is given by (5) and (3). For illustration purposes, say in Fig. 1, the initial region of θ (condition (3)) is given by PQRST and the condition of primal feasibility is given by UVWX (condition (5)), then CR^2 is the corresponding critical region. Note that CR^2 is obtained by removing the redundant constraints, PT, QR and RS. In order to devise a procedure to obtain 'all' the critical regions (CR^1 and CR^3), and optimal solutions associated with them, we first state the following:

- Two optimal bases are said to be *neighbors* if
  - there exists some $\theta^* \in K$ such that both the bases are optimal, and,
  - it is possible to pass from one basis to another by one dual step.
- The critical regions associated with two different optimal bases are said to be *neighbors* if their corresponding bases are neighbors.
- Two neighboring critical regions lie in opposite half spaces.
- The optimal value function, $z(\theta)$, is continuous and convex; see Fig. 2 for a graphical interpretation for the case of two parameters.

Based upon the above statements, the solution algorithm for identifying all the critical regions can now be described. The algorithm consists of two major parts. In the first part, an initial feasible solution is obtained and the critical region which corresponds to the initial solution is characterized. The second part then starts with this critical region and identifies all the regions and corresponding optimal solutions. The major steps of the algorithm are as follows:

1) Find a feasible solution:
   - Solve (1) by treating θ as a free variable to obtain θ^*. If no feasible solution exists, stop; (1) is infeasible.



Multiparametric Linear Programming, Figure 2
z*(*θ*)is a continuous and convex function of *θ

   - Fix $\theta = \theta^*$ and solve (1) to obtain an initial basis B and corresponding critical region.

2) Find all optimal solutions:
   - Construct two lists V and W, where V consists of those optimal bases whose neighboring bases have been identified, and W consists of those bases whose neighbors have yet not been identified.
   - Select any basis from W and identify all its neighboring bases. From all the identified bases, insert in W those bases which are neither in V nor in W. The optimal solutions (and corresponding critical regions) are then determined by moving from the basis to its neighbors by one dual step.
   - Repeat the procedure until $W = \{\emptyset\}$.

See also

- ▶ Bounds and Solution Vector Estimates for Parametric NLPs
- ▶ Global Optimization in Multiplicative Programming
- ▶ Linear Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Multiplicative Programming
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Parametric Global Optimization: Sensitivity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Parametric Optimization: Embeddings, Path Following and Singularities
- ▶ Selfdual Parametric Method for Linear Programs

References

1. Gal T (1992) Putting the LP survey into perspective. OR/MS Today 19(6):93
2. Gal T (1992) Weakly redundant constraints and their impact on postoptimal analysis in linear programming. Europ J Oper Res 60:315–336
3. Gal T (1995) Postoptimal analyses, parametric programming, and related topics. de Gruyter, Berlin
4. Gal T (1997) Advances in sensitivity analysis and parametric programming. Kluwer, Dordrecht
5. Gal T, Nedoma J (1972) Multiparametric linear programming. Managem Sci 18:406–422
6. Granot D, Granot F, Johnson EL (1982) Duality and pricing in multiple right-hand choice linear programming problems. Math Oper Res 7:545–556
7. Greenberg HJ (1986) An analysis of degeneracy. Naval Res Logist Quart 33:635–655
8. Greenberg HJ (1993) How to analyze the results of linear programs - Part 1: Preliminaries. Interfaces 23(4):56–67
9. Greenberg HJ (1993) How to analyze the results of linear programs - Part 2: Price Interpretation. Interfaces 23(5):97–114
10. Greenberg HJ (1993) How to analyze the results of linear programs - Part 3: Infeasibility Diagnosis. Interfaces 23(6):120–139
11. Greenberg HJ (1994) How to analyze the results of linear programs - Part 4: Forcing Structures. Interfaces 24(1):121–130
12. Greenberg HJ (1994) The use of optimal partition in linear programming solution for postoptimal analysis. Oper Res Lett 15:179–185
13. Greenberg HJ (1996) The ANALYZE rulebase for supporting LP analysis. Ann Oper Res 65:91–126
14. Hansen PM, Labbe M, Wendell RE (1989) Sensitivity analysis in multiple objective linear programming: the tolerance approach. Europ J Oper Res 38(1):63–69
15. Magnati TL, Orlin JB (1988) Parametric linear programming and anti-cycling pivoting rules. Math Program 41:317–325
16. Murty K (1980) Computational complexity of parametric linear programming. Math Program 19:213–219
17. Roos C, Terlaky T, Vial -Ph J (1997) Theory and algorithms for linear optimization, an interior point approach. Wiley, New York
18. Wang H-F, Huang C-S (1993) Multiparametric analysis of the maximum tolerance in a linear programming problem. Europ J Oper Res 67(1):75–87
19. Ward JE, Wendell RE (1990) Approaches to sensitivity analysis in linear programming. Ann Oper Res 27:3–38
20. Wendell RE (1985) The tolerance approach to sensitivity analysis in linear programming. Managem Sci 31:564–578
21. Wendell RE (1997) Linear programming 3: The tolerance approach. In: Gal T, Greenberg HJ (eds) Advances in Sensitivity Analysis and Parametric Programming. Kluwer, Dordrecht

Multiparametric Mixed Integer Linear Programming

VIVEK DUA, EFSTRATIOS N. PISTIKOPOULOS
Imperial College, London, UK

MSC2000: 90C31, 90C11

Article Outline

Keywords
Mixed Integer Linear Programming Problems
  Involving a Single Uncertain Parameter
  in Objective Function Coefficients
See also
References

Keywords

Parametric bounds; Branch and bound; Comparison of parametric solutions

In this article we describe theoretical and algorithmic developments in the field of parametric programming for linear models involving 0–1 integer variables. We will consider two cases of the problem: single parametric (when a single uncertain parameter is present) and multiparametric (when more than one uncertain parameters are present in the model). For the case when a single uncertain parameter is present, solution approaches are based upon

a) enumeration [11,12,13];
b) cutting planes [6]; and
c) branch and bound techniques [8,10].

For the multiparametric case, solution algorithm that has been proposed is based upon branch and bound fundamentals [1,2]. While most of the work on single parametric problems has been reviewed in the two excellent papers [5] and [7], and has been borrowed here for the sake of completeness, the work on multiparametric problems, the focus of this article, is quite recent and is described in detail. It may be mentioned that while solution approaches for single parametric case are available for uncertainty in objective function coefficients or right-hand side of constraints, for the case of more than one uncertain parameter the solution ap-

proach is available only for the right-hand side case. Next we will describe solution approaches for

a) single parametric mixed integer linear programs for objective function coefficients parametrization; and
b) single parametric pure integer programs when the uncertain parameter is present on the right-hand side of the constraints.

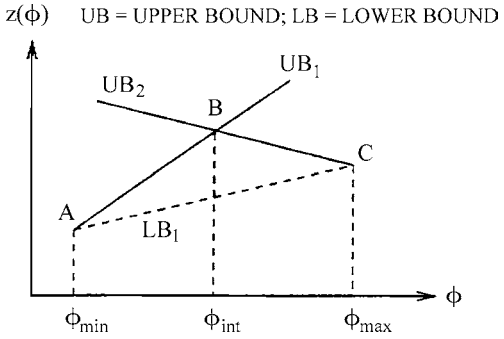
These illustrate some concepts which are based upon some basic observations. For other solution approaches, see the literature cited above. Finally we will present a solution approach for right-hand side multiparametric mixed integer linear programs.

Mixed Integer Linear Programming Problems Involving a Single Uncertain Parameter in Objective Function Coefficients

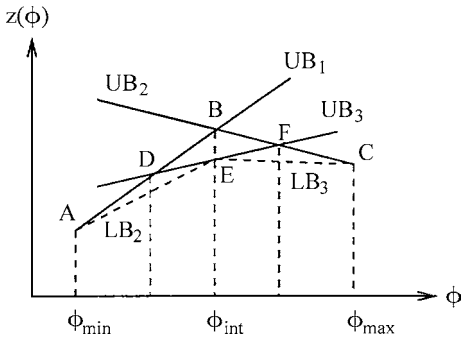
These can be stated as follows:

$$\begin{cases} z(\phi) & = \min\limits_{x,y} (c^\top + c'\phi)x + d^\top y \\ \text{s.t.} & Ax + Ey \le b, \\ & x \in \mathbb{R}^n, \quad y \in \{0,1\}^l, \\ & \phi_{\min} \le \phi \le \phi_{\max}, \end{cases} \tag{1}$$

where x is a vector of continuous variables; y is the vector of 0–1 integer variables; ϕ is a scalar uncertain parameter bounded between its lower and upper bounds $\phi_{\min}$ and $\phi_{\max}$ respectively; A is an $(m \times n)$ matrix; E is an $(m \times l)$ matrix; c, c', d and b are vectors of appropriate dimensions. Solution procedure for (1) is based upon following two features of the formulation in (1). First feature of this formulation is that, since the uncertain parameter is present in the objective function only, the feasible region of (1) remains constant for all the fixed values of ϕ in $[\phi_{\min}, \phi_{\max}]$. And the second feature is that, the optimal value of (1) for $\phi_{\min} \le \phi \le \phi_{\max}$ is piecewise linear, continuous, and concave on its finite domain. The solution is then approached by deriving valid upper and lower bounds, using the concavity property of the objective function value, and sharpening these bounds until they converge to the same value, as described next. Solving (1) for ϕ fixed at its endpoints $\phi_{\min}$ and $\phi_{\max}$, gives upper bounds AB and BC respectively (see Fig. 1); and a linear interpolation, AC, between the endpoints provides a lower bound to the solution. The region ABC within which the solution will



Multiparametric Mixed Integer Linear Programming, Figure 1
Derivation of bounds



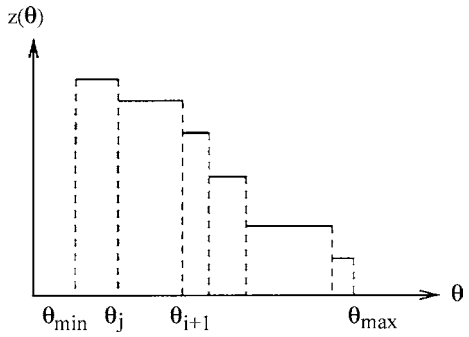
Multiparametric Mixed Integer Linear Programming, Figure 2
Sharpening of bounds

lie is then reduced by solving (1) at ϕ_{int}, the intersection point of two upper bounds AB and BC. This results (see Fig. 2) in two smaller regions, ADE and EFC, within which the solution will exist. This procedure is continued until the difference between upper and lower bounds becomes zero.

Integer programming problem involving a single uncertain parameter on the right-hand side of the constraints can be stated as follows:

$$\begin{cases} z(\theta) & = \min\limits_{y} d^\top y \\ \text{s.t.} & Ey \le b + r\theta, \\ & \theta_{\min} \le \theta \le \theta_{\max} \\ & y \in \{0,1\}^l, \end{cases} \tag{2}$$

where r is a scalar constant and θ is a scalar uncertain parameter bounded between $\theta_{\min}$ and $\theta_{\max}$ respectively. For a special case of (2) when $r \ge 0$, it may be



Multiparametric Mixed Integer Linear Programming, Figure 3
Step function nature of objective function value

noted that as θ is increased from $\theta_{\min}$ to $\theta_{\max}$, the feasible region will enlarge, and hence the objective function value will decrease or remain the same, i. e., $z(\theta_i) \geq z(\theta_{i+1})$ for $\theta_i \leq \theta_{i+1}$. Further, since only integer variables are present in (2), a solution will remain optimal for some interval of θ and then suddenly another solution will become optimal, and remain so for the next interval (see Fig. 3). The problem thus reduces to solving (2) at an end point, say $\theta_{\min}$, and then finding a point θ_i at which the current solution becomes infeasible. Solving (2) at $\theta_i + \epsilon$ will give another integer solution. This procedure is continued until we hit the other end point, $\theta_{\max}$.

Consider a multiparametric mixed integer linear programming problem (mp-MILP) of the following form:

$$\begin{cases} z(\theta) & = \min\limits_{x,y} c^\top x + d^\top y \\ \text{s.t.} & Ax + Ey \leq b + F\theta, \\ & G\theta \leq g, \\ & x \in \mathbb{R}^n, \quad y \in \{0,1\}^l, \quad \theta \in \mathbb{R}^s, \end{cases} \tag{3}$$

where θ is a vector of uncertain parameters; F is an $(m \times s)$ matrix, G is an $(r \times s)$ matrix, and g is a constant vector. Solving (3) implies obtaining the optimal solution to (3) for every θ that lies in $\Xi = \{\theta : G\theta \leq g, \theta \in \mathbf{R}^s\}$. The algorithm for the solution of (3) proposed in [1] is based upon simultaneously using the concepts of

- branch and bound method for solving mixed integer linear programming (MILP) problems (see, e. g., [9]); and,
- simplex algorithm for solving multiparametric linear programming (mp-LP) problems [4].

While a solution of (3) by relaxing the integrality condition on y (at the root node) represents a *parametric lower bound*, a solution where all the y variables are fixed (e. g., at a terminal node) represents a *parametric upper bound*. The algorithm proceeds from the root node (lower bound) towards terminal nodes (upper bound) by fixing y variables at the intermediate nodes. The complete enumeration of the tree is avoided by fathoming those intermediate nodes which guarantee a suboptimal solution.

At the root node, by relaxing the integrality condition on y, i. e., considering y as a continuous variable bounded between 0 and 1, (3) is transformed to an mp-LP of the following form:

$$\begin{cases} \check{z}(\theta) & = \min\limits_{x,\check{y}} c^\top x + d^\top \check{y} \\ \text{s.t.} & Ax + E\check{y} \leq b + F\theta, \\ & G\theta \leq g, \\ & 0 \leq \check{y} \leq 1, \\ & x \in \mathbb{R}^n, \quad \theta \in \mathbb{R}^s. \end{cases} \tag{4}$$

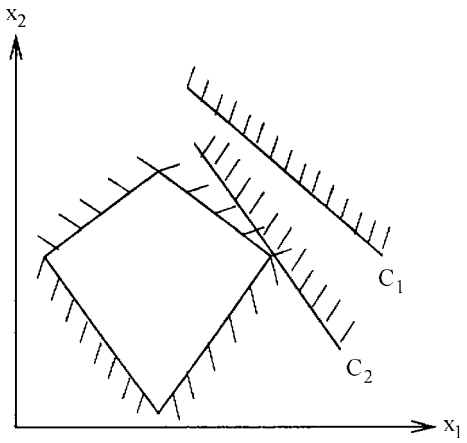
The solution of (4), given by linear parametric profiles, $\check{z}(\theta)^i$, valid in their corresponding *critical regions*, $\check{\text{CR}}^i$, represents a parametric lower bound.

Similarly, at a node where all y are fixed, $y = \widehat{y}$, (3) is transformed to an mp-LP of the following form:

$$\begin{cases} \widehat{z}(\theta) & = \min\limits_{x,\widehat{y}} c^\top x + d^\top \widehat{y} \\ \text{s.t.} & Ax + E\widehat{y} \leq b + F\theta, \\ & G\theta \leq g, \\ & \widehat{y} = \{0,1\}^l, \\ & x \in \mathbb{R}^n, \quad \theta \in \mathbb{R}^s. \end{cases} \tag{5}$$

The solution of (5), $\widehat{z}(\theta)^i$, valid in its corresponding critical regions, $\widehat{\text{CR}}^i$, represents a parametric upper bound.

Starting from the root node, some of the y variables are systematically fixed (to 0 and 1) to generate intermediate nodes of the branch and bound tree. At an intermediate node, where some y are fixed and some are



Multiparametric Mixed Integer Linear Programming, Figure 4
Redundant constraints

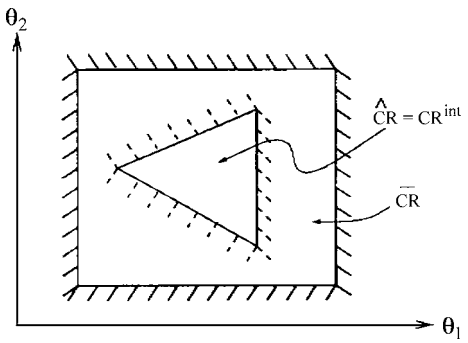
relaxed, an mp-LP of the following form is formulated:

$$\begin{cases} \overline{z}(\theta) & = \min\limits_{x,y} c^\top x + d_j^\top \widehat{y}_j + d_k^\top \check{y}_k \\ \text{s.t.} & Ax + E_j \widehat{y}_j + E_k \check{y}_k \le b + F\theta, \\ & G\theta \le g, \\ & \widehat{y}_j = \{0,1\}, \\ & 0 \le \check{y}_k \le 1, \\ & x \in \mathbb{R}^n, \quad \theta \in \mathbb{R}^s, \end{cases} \tag{6}$$

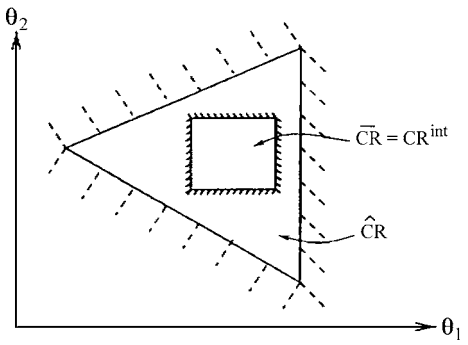
where the subscripts j and k correspond to y that are fixed and y that are free, respectively. The solution at an intermediate node, $\overline{z}(\theta)^i$, valid in its corresponding critical regions, $\overline{\text{CR}}^i$, is then analyzed, to decide whether to explore subnodes of this intermediate node or not, by using the following fathoming criteria. A given space in any node can be discarded if one of the following holds:

- (*infeasibility criterion*) Problem (6) is infeasible in the given space.
- (*integrality criterion*) An integer solution is found in the given space.
- (*dominance criterion*) The solution of the node is greater than the current upper bound in the same space.

If all the regions of a node are discarded the node can be fathomed. While the first two fathoming criteria (Infeasibility and Integrality) are easy to apply, in order to apply the third one (dominance criteria) we need a comparison procedure, which is described next.



Multiparametric Mixed Integer Linear Programming, Figure 5
Definition of CR^{int}; Case 1

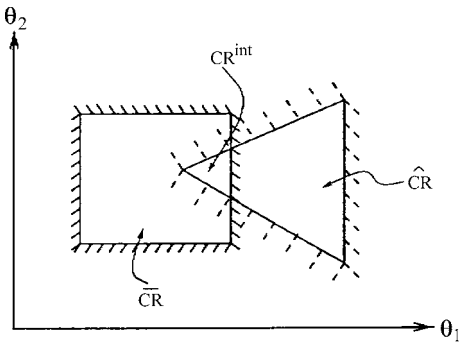


Multiparametric Mixed Integer Linear Programming, Figure 6
Definition of CR^{int}; Case 2

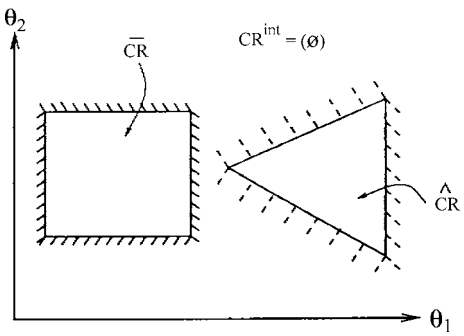
The comparison procedure consists of two steps. In the first step, a region, $\text{CR}^{\text{int}} = \widehat{\text{CR}} \cap \overline{\text{CR}}$, where the solution of the intermediate node and the current upper bound are valid is defined. This is achieved by removing the redundant constraints from the set of constraints which define $\widehat{\text{CR}}$ and $\overline{\text{CR}}$ (for a procedure to eliminate redundant constraints see [3]); graphical interpretation of redundant constraints is given in Fig. 4, where C_1 is a strongly redundant constraint and C_2 is a weakly redundant constraint.

The results of this *redundancy test*, which belong to one of the following 4 cases, are then analyzed as follows:

- (case 1; Fig. 5) All constraints from $\overline{\text{CR}}$ are redundant. This implies that $\overline{\text{CR}} \supseteq \widehat{\text{CR}}$, and therefore $\text{CR}^{\text{int}} = \widehat{\text{CR}}$.



Multiparametric Mixed Integer Linear Programming, Figure 7
Definition of CR^{int}; Case 3



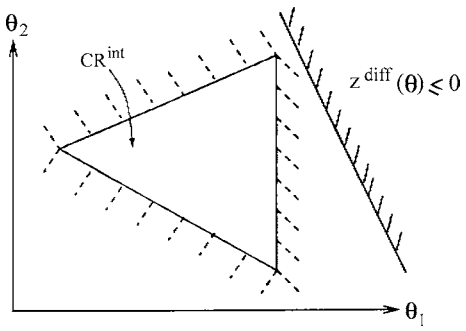
Multiparametric Mixed Integer Linear Programming, Figure 8
Definition of CR^{int}; Case 4

- (case 2; Fig. 6) All constraints from $\widehat{CR}$ are redundant. This implies that $\widehat{CR} \supseteq \overline{CR}$, and therefore $CR^{int} = \overline{CR}$.
- (case 3; Fig. 7) Constraints from both regions are nonredundant. This implies that two spaces intersect with each other, and CR^{int} is given by the space delimited by the nonredundant constraints.
- (case 4; Fig. 8) The problem is infeasible. This implies that two spaces are apart from each other and $CR^{int} = \{\emptyset\}$.

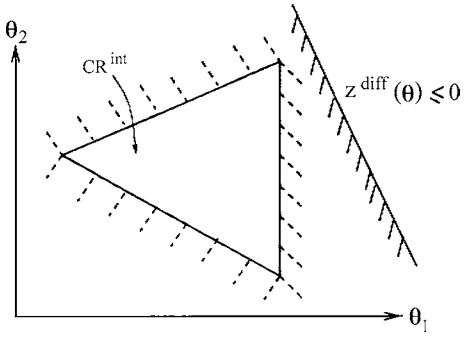
Once CR^{int} has been defined, the second step is to compare $\overline{z}$ to $\widehat{z}$, so as to find which of the two is lower. This is achieved by defining a new constraint:

$$z^{diff}(\theta) = \overline{z}(\theta) - \widehat{z}(\theta) \leq 0$$

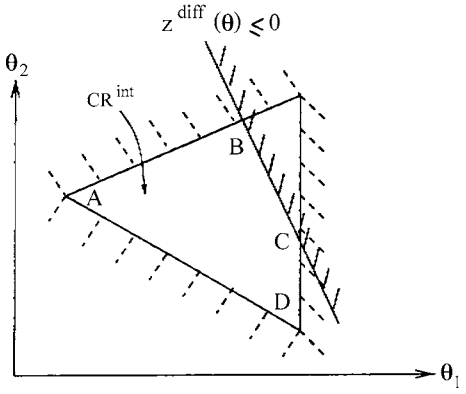
and checking for redundancy of this constraint in CR^{int}. This redundancy test results in following 3 cases:



Multiparametric Mixed Integer Linear Programming, Figure 9
Compare $\overline{z}(\theta) : \widehat{z}(\theta)$; Case 1



Multiparametric Mixed Integer Linear Programming, Figure 10
Compare $\overline{z}(\theta) : \widehat{z}(\theta)$; Case 2



Multiparametric Mixed Integer Linear Programming, Figure 11
Compare $\overline{z}(\theta) : \widehat{z}(\theta)$; Case 3

- (case 1; Fig. 9) The new constraint is redundant. This implies that $\overline{z}(\theta) \leq \widehat{z}(\theta)$ and therefore the space must be kept for further analysis.

- (case 2; Fig. 10) The problem is infeasible. This implies that $\overline{z}(\theta) \geq \widehat{z}(\theta)$ and therefore the space can be discarded from further analysis.
- (case 3; Fig. 11) The new constraint is nonredundant. This implies that $\overline{z}(\theta) \leq \widehat{z}(\theta)$ in *ABCD*, and therefore the rest of the space can be discarded from further analysis.

Based upon the above theoretical framework, the steps of the algorithm can be summarized as follows:

1 Set an upper bound of $\widehat{z}(\theta) = \infty$.
2 Solve the fully relaxed problem (4).
IF an integer solution is found in a critical region, THEN update the upper bound and discard the region from further analysis.
3 Fix one of the y variables to 0 and 1 to create two new nodes.
IF no new nodes can be generated, THEN stop.
4 Solve the resulting problem (6).
IF the problem is infeasible THEN go back to Step 3,
ELSE compare the solution to the current upper bound.
5 IF all regions from a node have been analyzed, THEN go to Step 3.

See also

- ▶ Bounds and Solution Vector Estimates for Parametric NLPs
- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Linear Programming
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Parametric Global Optimization: Sensitivity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Parametric Optimization: Embeddings, Path Following and Singularities
- ▶ Selfdual Parametric Method for Linear Programs
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem

References

1. Acevedo J, Pistikopoulos EN (1997) A multiparametric programming approach for linear process engineering problems under uncertainty. Industr Eng Chem Res 36:717–728
2. Acevedo J, Pistikopoulos EN (1999) An algorithm for multiparametric mixed integer linear programming problems. Oper Res Lett 24:139–148
3. Gal T (1995) Postoptimal analyses, parametric programming, and related topics. de Gruyter, Berlin
4. Gal T, Nedoma J (1972) Multiparametric linear programming. Managem Sci 18:406–422
5. Geoffrion AM, Nauss R (1977) Parametric and postoptimality analysis in integer linear programming. Managem Sci 23(5):453–466
6. Holm S, Klein D (1984) Three methods for postoptimal analysis in integer linear programming. Math Program Stud 21:97–109
7. Jenkins L (1990) Parametric methods in integer linear programming. Ann Oper Res 27:77–96
8. Marsten RE, Morin TL (1977) Parametric integer programming: The right-hand side case. Ann Discret Math 1:375–390
9. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
10. Ohtake Y, Nishida N (1985) A branch-and-bound algorithm for 0-1 parametric mixed-integer programming. Oper Res Lett 4(1):41–45
11. Piper CJ, Zoltners AA (1976) Some easy postoptimality analysis for zero-one programming. Managem Sci 22(7):759–765
12. Roodman GM (1972) Postoptimality analysis in zero-one programming by implicit enumeration. Naval Res Logist Quart 19:435–447

13. Roodman GM (1974) Postoptimality analysis in zero-one programming by implicit enumeration: The mixed-integer case. Naval Res Logist Quart 21:595–607

Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach

John L. Klepeis, Christodoulos A. Floudas
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 92C40, 65K10

Article Outline

Keywords
Motivation
Mathematical Description
Potential Energy Modeling
Solvation Energy Modeling
Problem Formulation
Global Minimization Using αBB
Algorithmic Description
See also
References

Keywords

Protein folding; Multiple minima; Global optimization; αBB

Motivation

Proteins are arguably the most complex molecules in nature. This complexity arises from an intricate balance of intra- and inter-molecular interactions that define the native three-dimensional structure of the system, and subsequently its biological functionality. The underlying goal of *protein folding* research is to understand the formation of these native tertiary structures. Genetic engineering can be used to produce proteins with specific amino acid sequences. The next step involves developing the link between the primary protein sequence and the native structure. The ability to predict the folding of proteins promises to have important practical and theoretical ramifications, especially in the areas of medicinal and biophysical chemistry.

Experimental studies have shown that proteins, under native physiological conditions, spontaneously refold to their unique, native structure after denaturation. This implies that the formation of the native structure is controlled primarily by the amino acid sequence. According to Anfinsen's hypothesis the native structure is in a state of thermodynamic equilibrium corresponding to the conformation with the lowest free energy. Through mathematical modeling of protein interaction energies, the protein folding problem can be addressed as a *conformational search* for the global minimum energy.

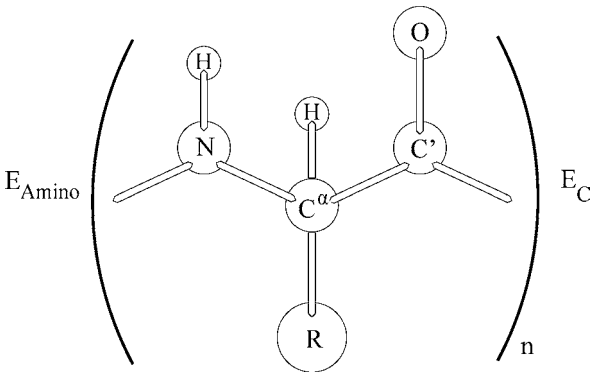
There exists two fundamental problems associated with protein folding in the context of a conformational search. The first is the ability to correctly model protein interactions using detailed mathematical equations. The second is associated with searching the highly nonconvex energy hypersurface that describes a given protein. This complexity, coupled with an exponential growth in the number of local minima as the size of protein increases, has become known as the *multiple minima* problem. There exists an obvious need for the development of efficient global optimization techniques. An efficient method which has been successfully applied to detailed atomistic models of protein folding is the *αBB* [1,2,3,17] global optimization algorithm.

Mathematical Description

Proteins are essentially polymer chains composed of a predefined set of amino acid residues in which neighboring residues are linked by peptidic bonds. Naturally occurring proteins consist of only 20 different amino acid residues, and the form of the side chain R (e. g., methyl, butyl, benzoic, etc.) defines the differences between these constituent groups. The chemical structure of a generic protein is illustrated in Fig. 1. The repeating unit $-NC^{\alpha}C'-$ defines the backbone of the protein. The protein also possesses amino and carboxyl end groups, denoted by E_{Amino} and E_{Carboxyl}, respectively.

The geometry of a protein can be fully described by assigning a three-dimensional coordinate vector r_i:

$$r_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}.$$



Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 1
Generic primary protein structure

These r_i specify the position of each atom in the protein molecule. The bond vector between two atoms (i, j) connected with a covalent bond is defined as:

$$r_{ij} = \begin{pmatrix} x_j - x_i \\ y_j - y_i \\ z_j - z_i \end{pmatrix}.$$

The corresponding bond length is then equal to the Euclidean distance between these two atoms:

$$|r_{ij}| = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}$$

A covalent bond angle, θ_{ijk}, formed by the two adjacent bond vectors r_{ij} and r_{jk} can be computed by the following formulas:

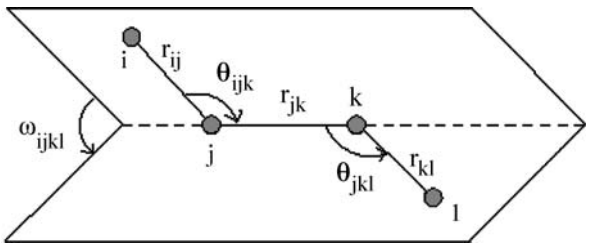
$$\cos(\theta_{ijk}) = \frac{r_{ij} \cdot r_{jk}}{|r_{ij}|\,|r_{jk}|}, \quad \sin(\theta_{ijk}) = \frac{r_{ij} \times r_{jk}}{|r_{ij}|\,|r_{jk}|}.$$

Here, $r_{ij} \cdot r_{jk}$ is the dot product of the bond vectors r_{ij} and r_{jk} and $r_{ij} \times r_{jk}$ is the cross product.

The dihedral angle ω_{ijkl} measures the relative orientation of two adjacent covalent angles θ_{ijk} and θ_{jkl}. This angle is defined as the angle between the normals through the planes defined by atoms i, j, k and j, k, l respectively, and can be calculated from the following relations:

$$\cos(\omega_{ijkl}) = \frac{(r_{ij} \times r_{jk}) \cdot (r_{jk} \times r_{kl})}{|r_{ij} \times r_{jk}|\,|r_{jk}, \times r_{kl}|},$$

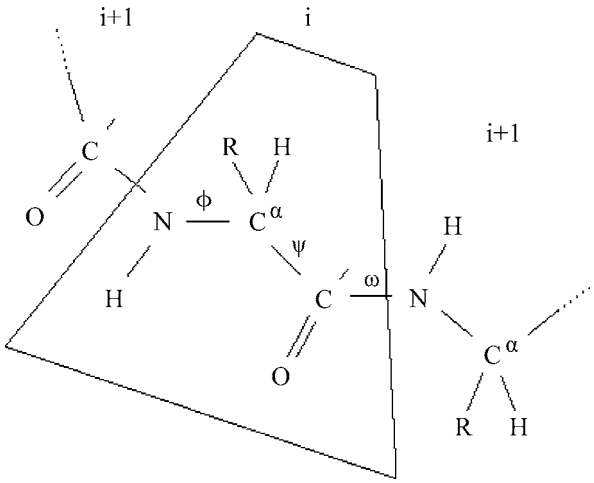
$$\sin(\omega_{ijkl}) = \frac{(r_{kl} \times r_{ij}) \cdot r_{jk}\,|r_{jk}|}{|r_{ij} \times r_{jk}|\,|r_{jk} \times r_{kl}|}.$$



Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 2
Illustration of dihedral angle

An alternative to specifying the coordinate vector for all atoms in a protein molecule is to set bond lengths, covalent bond angles and independent dihedral angles. A common approximation is to assume rigid bond lengths and bond angles so that the dihedral angles can be used to fully characterize the shape of the protein molecule.

The names of the dihedral angles of a protein chain follow a standard nomenclature. The dihedral angle between the normals of the planes formed by atoms $C_{i-1}{}'N_iC_i^{\alpha}$ and $N_iC_i^{\alpha}C_i'$ respectively, is called ϕ_i, where $i-1$ and i are two adjacent amino acid residues. The angle defined by the planes $N_iC_i^{\alpha}C_i'$ and $C_i^{\alpha}C_i'N_{i+1}$, respectively, is called ψ_i, where i and $i+1$ are two adjacent amino acid residues. Also, ω_i is the dihedral angle defined by the planes $C_i^{\alpha}\,C_i'\,N_{i+1}$ and $C_i'N_{i+1}C_{i+1}^{\alpha}$.



Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 3
Dihedral angle conventions

The letter χ is utilized to denote the dihedral angles which are associated with the side groups R_i. Finally, the letter θ is used to name the dihedral angles associated with the two end groups. These conventions are illustrated in Fig. 3.

Potential Energy Modeling

A number of empirically based *molecular mechanics* models have been developed for protein systems, including AMBER [24], CHARMM [7], ECEPP/3 [19], GROMOS [11], MM3 [4]. These models, also known as *force fields*, are typically expressed as summations of several potential energy components, with the mathematical form of individual energy terms based on the phenomenological nature of that term. A general total potential energy equation should include terms for bond stretching (E_{bond}), angle bending (E_{angle}), torsion (E_{tor}) and nonbonded (E_{nb}) interactions:

$$E_{\text{potential}} = E_{\text{bond}} + E_{\text{angle}} + E_{\text{tor}} + E_{\text{nb}}$$

When rigid body approximations are employed, bond stretching and angle bending energies can be neglected. For these force fields, torsion angles define a set of independent variables that effectively describe any protein conformation. This approximately reduces the number of variables by a factor of 3 over those force fields that use a Cartesian coordinate system to describe flexible molecular geometries.

One example of a rigid body atomistic level potential energy model is the ECEPP/3 force field. In this case, the nonbonded energy terms, E_{nb}, include electrostatic, E_{elec}, van der Waals, E_{vdw}, and hydrogen bonding, E_{hbond}, interactions. These energies are calculated for those atoms that are separated by more than two atoms; that is, the atoms possess at least a 1–4 relationship. Electrostatic energies, E_{elec}, are calculated as Coulombic forces based on atomic point charges:

$$E_{\text{elec}} = \frac{Q_i Q_j}{\epsilon R_{ij}}$$

Here, Q_i and Q_j represent the two point charges, while R_{ij} equals the distance between these two points. The ϵ term describes the dielectric nature of the protein environment.

General nonbonded van der Waals interactions, E_{vdw}, are modeled using a 6–12 Lennard–Jones potential energy term, which consists of a repulsion and attraction term:

$$E_{\text{vdw}} = \epsilon_{ij}\left[\left(\frac{R^*_{ij}}{R_{ij}}\right)^{12} - 2\left(\frac{R^*_{ij}}{R_{ij}}\right)^{6}\right].$$

The energy minimum for a given atomic pair is described by the potential depth, ϵ_{ij}, and position, R^*_{ij}. For those atomic pairs that may form a hydrogen bond, the 6–12 potential energy term is replaced by a modified 10–12 Lennard–Jones type term:

$$E_{\text{hbond}} = \epsilon_{ij}\left[5\left(\frac{R^*_{ij}}{R_{ij}}\right)^{12} - 6\left(\frac{R^*_{ij}}{R_{ij}}\right)^{10}\right].$$

Finally, corrective torsional energies, E_{tor}, which are represented by a three term Fourier series expansion, are also added:

$$E_{\text{tor}} = \frac{E_1}{2}(1 - \cos\phi) + \frac{E_2}{2}(1 - \cos 2\phi) + \frac{E_3}{2}(1 - \cos 3\phi).$$

Each term can be interpreted physically. The 1-x (cos ϕ) symmetry term accounts for those nonbonded interactions not included in general nonbonded terms. The 2-x (cos 2 ϕ) symmetry term is related to the interactions of orbitals, while the 3-x (cos 3 ϕ) symmetry term describes steric contributions.

Other specific potential energy terms may also be added to the general energy equation depending on the exact protein sequence. For example, the formation of disulfide bridges can be enforced by adding a penalty term to constrain the values of particular atomic distances. Correction terms have also been used to adjust conformational energies according to the configurations of proline and hydroxyproline residues.

Solvation Energy Modeling

In general, the energetic description of a protein must also include *solvation effects*. A theoretically simple approach would be to explicitly surround the peptide with solvent molecules and compute potential energy contributions for intra-and inter-molecular interactions. These explicit calculations tend to greatly increase the computational cost of the simulation. In addition, solvent configurations are not rigid, so these calcula-

tions must consider an average solvent-peptide configuration, which is typically generated by a number of *Monte-Carlo* (MC) or molecular dynamics (MD) simulations [14]. Therefore, most simulations of this type are limited to restricted conformational searches.

An alternative way for effectively considering average solvent effects is to use implicit solvation models. One complication involves the solvent's influence on electrostatic interaction energies because of the implicit relationship between dielectric effects and solvation. A simple solution has been to modify the representation of the dielectric term. In reality, however, the rigorous treatment of electrostatic interactions involves the solution of the Poisson–Boltzmann equation.

Other simple and computationally feasible implicit solvation models are based on empirical representations of the solvation energy. In these cases, the solvation energy of each functional group is related to the interaction of the solvent with a hydration shell for the particular group. The individual terms are then summed together to provide a total solvation energy for the system. These solvation contributions can be described by the following general equation:

$$E_{\text{solv}} = \sum_{i=1}^{N} S_i \sigma_i .$$

Typically, S_i represents either the solvent-accessible surface area, A_i, or the solvent-accessible volume of hydration layer, VHS_i, for the functional group, and σ_i is an empirically derived free energy density parameter.

A number of algorithms have been developed for calculating solvent-accessible surface areas [8,9,22]. Although several of these are relatively efficient, the appearance of discontinuities has been one complication in considering solvent accessible surface areas. In addition, a large number of parameterization strategies (JRF, OONS, WE, etc.) have been used to derive appropriate σ_i parameters [21,23,25]. In the case of the JRF parameter set, discontinuities can be avoided because the surface-accessible solvation energies are only included at local minimum conformations [23]. This is because the parameters were derived from low energy solvated configurations of actual tetrapeptides.

Several methods have also been developed for calculating the hydration volumes and corresponding free energy parameters [6,12]. A recent and computationally inexpensive method, RRIGS, is based on a Gaussian approximation for the volume of a hydration layer [6]. This method also inherently avoids numerical problems associated with possible discontinuities so that the solvation energy contributions can easily be added at every step of local minimizations.

Problem Formulation

For protein folding, the *energy minimization* problem can be formulated as a nonconvex, nonlinear global optimization problem in which the energy, E, must be globally minimized with respect to the dihedral angles of the protein:

$$\begin{cases} \min & E(\phi_i, \psi_i, \omega_i, \chi_i^k, \theta_j^N, \theta_j^C) \\ \text{subject to} & -\pi \le \phi_i \le \pi \\ & -\pi \le \psi_i \le \pi \\ & -\pi \le \omega_i \le \pi \\ & -\pi \le \chi_i^k \le \pi \\ & -\pi \le \theta_j^N \le \pi \\ & -\pi \le \theta_j^C \le \pi . \end{cases}$$

The index $i = 1, \ldots, N_{\text{RES}}$ defines the number of residues, N_{RES}, in the protein. In addition, $k = 1, \ldots, K^i$ denotes the number of dihedral angles in the side chain of the ith residue, and $j = 1, \ldots, J^N$ and $j = 1, \ldots, J^C$ indicates the indices of the amino and carboxyl end groups, respectively. The energy, E, represents the total potential energy function, $E_{\text{potential}}$, plus the free energy of solvation, E_{solv}. In most cases, this is the exact formulation; that is, energetic and gradient contributions can be added at each step of the minimization. However, in the case of surface-accessible hydration using the JRF parameters, the potential energy function is minimized before adding the hydration energy contributions. In other words, gradient contributions from solvation are not considered.

Even after reducing this optimization problem to a function of internal variables, the multidimensional surface that describes the energy function possesses an astronomically large number of local minima. In addition, evaluation of the energy, especially with the addition of solvation, is computationally expensive, which makes even local minimization slow. A large number of techniques have been developed to search this nonconvex conformational space. Many methods employ

stochastic search procedures, while others rely on simplifications of the potential model and/or mathematical transformations. In addition, the use of statistical and/or heuristic conformational information is often required. In general, the major limitation is that there is no guarantee for convergence to the global minimum energy structure. A number of recent reviews have focused on global optimization issues for these systems [10,20].

The αBB global optimization approach has been extremely effective in identifying global minimum energy conformations of peptides described by detailed atomistic models. The development of this deterministic *branch and bound* method was motivated by the need for an algorithm that could guarantee convergence to the global minimum of nonlinear optimization problems with twice-differentiable functions. The application of this algorithm to the minimization of potential energy functions was first introduced for microclusters [16]. The algorithm has also been shown to be successful for isolated [5,15], as well as solvated peptide systems [13].

Global Minimization Using αBB

The αBB global optimization algorithm effectively brackets the global minimum solution by developing converging sequences of lower and upper bounds. These bounds are refined by iteratively partitioning the initial domain. Upper bounds on the global minimum are obtained by local minimizations of the original energy function, E. Lower bounds belong to the set of solutions of the convex lower bounding functions, which are constructed by augmenting E with the addition of separable quadratic terms. By using ϕ_i^L, ψ_i^L, ω_i^L, $\chi_i^{k,L}$, $\theta_j^{N,L}$, $\theta_j^{C,L}$ and ϕ_i^U, ψ_i^U, ω_i^U, $\chi_i^{k,U}$, $\theta_j^{N,U}$, $\theta_j^{C,U}$ to refer to lower and upper bounds on the corresponding dihedral angles, the lower bounding function, L, of the energy hypersurface can be expressed in the following manner:

$$
\begin{aligned}
L = E
&+ \sum_{i=1}^{N_{\mathrm{RES}}} \alpha_{\phi,i} \left(\phi_i^L - \phi_i\right)\left(\phi_i^U - \phi_i\right) \\
&+ \sum_{i=1}^{N_{\mathrm{RES}}} \alpha_{\psi,i} \left(\psi_i^L - \psi_i\right)\left(\psi_i^U - \psi_i\right) \\
&+ \sum_{i=1}^{N_{\mathrm{RES}}} \alpha_{\omega,i} \left(\omega_i^L - \omega_i\right)\left(\omega_i^U - \omega_i\right) \\
&+ \sum_{i=1}^{N_{\mathrm{RES}}} \sum_{k=1}^{K^i} \alpha_{\chi,i,k} \left(\chi_i^{k,L} - \chi_i^k\right)\left(\chi_i^{k,U} - \chi_i^k\right) \\
&+ \sum_{j=1}^{J^N} \alpha_{j,\theta^N} \left(\theta_j^{N,L} - \theta_j^N\right)\left(\theta_j^{N,U} - \theta_j^N\right) \\
&+ \sum_{j=1}^{J^C} \alpha_{j,\theta^C} \left(\theta_j^{C,L} - \theta_j^C\right)\left(\theta_j^{C,U} - \theta_j^C\right).
\end{aligned}
$$

The α represent nonnegative parameters which must be greater or equal to the negative one-half of the minimum eigenvalue of the Hessian of E over the defined domain. The overall effect of these terms is to overpower the nonconvexities of the original nonconvex terms by adding the value of 2 α to the eigenvalues of the Hessian of E. The convex lower bounding functions, L, possess a number of important properties which guarantee global convergence [18]:

i) L is a valid underestimator of E;
ii) L matches E at all corner points of the current box constraints;
iii) L is convex in the current box constraints;
iv) the maximum separation between L and E is bounded. This property ensures that feasibility and convergence tolerances can be reached for a finite size partition element;
v) the underestimators L constructed over supersets of the current set are always less tight than the underestimator constructed over the current box constraints for every point within the current box constraints.

Once solutions for the upper and lower bounding problems have been established, the next step is to modify the problem for the next iteration. This is accomplished by successively partitioning the initial domain into smaller subdomains. One obvious strategy is to subdivide the original hyper-rectangle by bisecting the longest dimension. In order to ensure nondecreasing lower bounds, the hyper-rectangle to be bisected is chosen by selecting the region which contains the infimum of the minima of lower bounds. A nonincreasing sequence for the upper bound is found by solving the nonconvex problem locally and selecting it to be the minimum over all the previously recorded up-

per bounds. If the single minimum of L for any hyper-rectangle is greater than the current upper bound, this hyper-rectangle can be discarded because the global minimum cannot be within this subdomain (fathoming step).

The computational requirement of the αBB algorithm depends on the number of variables (global) on which branching occurs. Therefore, these global variables need to be chosen carefully. Qualitatively, the branching variables should correspond to those variables which substantially influence the nonconvexity of the surface and the location of the global minimum. In terms of the protein folding problem, it is generally accepted that the backbone dihedral angles (ϕ and ψ) are the most influential variables. Therefore, in larger problems, the global variable set should include only the set of ϕ and ψ variables. In this case, the dihedral angles associated with the peptide bond (ω) and the side chains (χ) are treated as local variables.

Algorithmic Description

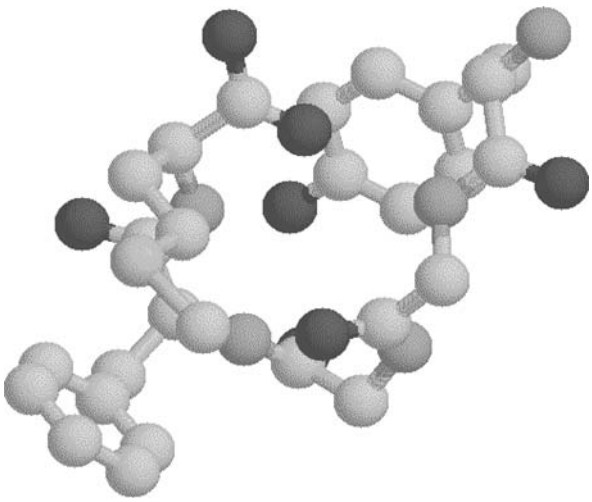
The basic steps of the algorithm are as follows:

1) The initial best upper bound is set to an arbitrarily large value. The original domain is partitioned along one of the global variable dimensions.
2) A convex function L is constructed in each hyper-rectangle and minimized using a local nonlinear solver, with function calls to potential and solvation models. If a solution is greater than the best upper bound the entire subregion can be fathomed, otherwise the solution is stored.
3) The local minima for L are used as initial starting points for local minimizations of the upper bounding function E in each hyper-rectangle. In solving the upper bounding problems, all variable bounds are expanded to $(-\pi, \pi)$ domain. These solutions are upper bounds on the global minimum solution in each hyper-rectangle.
4) The current best upper bound is updated to be the minimum of those thus far stored. If a new upper bound (from step 3) is selected, a separate module is called to ensure that the absolute value of each gradient in the objective function gradient vector is below a specified tolerance (kcal/mol/deg). The second derivative matrix is also evaluated to verify that the upper bound solution is a local minimum.
5) The hyper-rectangle with the current minimum value for L is selected and partitioned along one of the global variables.
6) If the best upper and lower bounds are within an ϵ tolerance the program will terminate, otherwise it will return to Step 2.

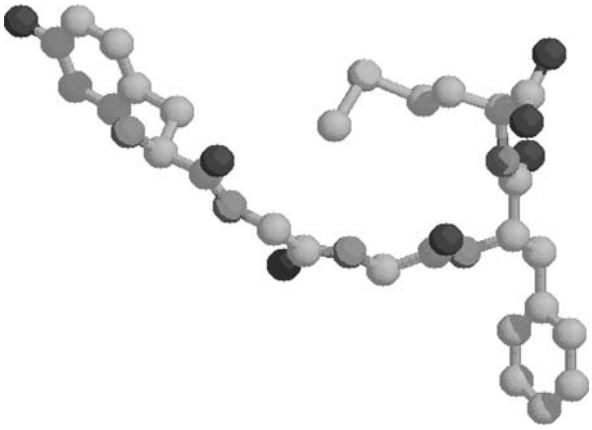
A novel approach has also been proposed for the initialization of the αBB algorithm [5]. Specifically, an analysis of 98 proteins from the Brookhaven X-ray data bank was used to develop dihedral angle distributions in the form of histograms from $-\pi$ to π for each dihedral angle of each of the naturally occurring amino acids. Using this information, a set of reduced domains can be defined for every dihedral angle of every residue in the peptide sequence. Overall initialization domains correspond to the Cartesian products of all the sub-domains of individual residues in the protein. This approach maintains the guarantee of global optimality over the considered search space of the reduced domains, and is deterministic in those subdomains that possess convex underestimators. In addition, all variable bounds are expanded to the $[-\pi, \pi]$ when solving the upper bounding problem. Therefore, although the initial point of an upper bounding minimization is restricted to the search space of the corresponding lower bounding problem, the solution may lie outside the original subdomain.

Example 1 Met-enkephalin (H-Tyr-Gly-Gly-Phe-Met-OH) is an endogenous opioid pentapeptide found in the human brain, pituitary, and peripheral tissues. Its biological function involves a large variety of physiological processes, most notably the endogenous response to pain. The peptide consists of 24 dihedral angles and a total of 75 atoms, and has played the role of a benchmark molecular conformation problem. The energy hypersurface is extremely complex with the number of local minima estimated on the order of 10^{11}. The unsolvated global minimum energy conformation, which is efficiently located using the αBB algorithm, has been shown to exhibit a type II' β-bend along the N-C' peptidic bond of Gly^3 and Phe^4 [5], as shown in Fig. 4.

The algorithm has also successfully predicted global minimum energy structures of met-enkephalin using both solvent-accessible surface area (JRF) and volume of hydration (RRIGS) models [13]. In both cases, extended structures were identified, which qualitatively



Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 4
Global minimum energy structure of unsolvated met-enkephalin

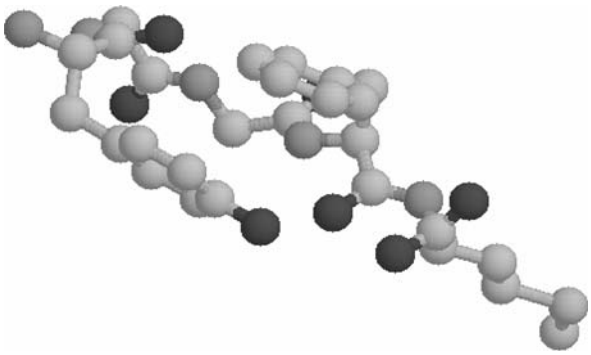


Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 5
Global minimum energy structure of met-enkephalin using area based hydration

agrees with experimental results. However, differences in the role of nonbonded energies and the side chain conformations have been identified. The global minimum energy conformations of the surface area and volume of hydration models are shown in Fig. 5 and Fig. 6, respectively.

See also

- ► Adaptive Simulated Annealing and its Application to Protein Folding



Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 6
Global minimum energy structure of met-enkephalin using volume based hydration

- ► Genetic Algorithms
- ► Global Optimization in Lennard–Jones and Morse Clusters
- ► Global Optimization in Protein Folding
- ► Molecular Structure Determination: Convex Global Underestimation
- ► Monte-Carlo Simulated Annealing in Protein Folding
- ► Packet Annealing
- ► Phase Problem in X-ray Crystallography: Shake and Bake Approach
- ► Protein Folding: Generalized-ensemble Algorithms
- ► Simulated Annealing
- ► Simulated Annealing Methods in Protein Folding

References

1. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB, for general twice-differentiable NLPs-II. Implementation and computational results. Comput Chem Eng 22:1159–1179
2. Adjiman CS, Androulakis IP, Maranas CD, Floudas CA (1996) A global optimization method, αBB, for process design. Comput Chem Eng 20:S419–S424
3. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB, for general twice-differentiable NLPs-I. Theoretical advances. Comput Chem Eng 22:1137–1158
4. Allinger NL, Yuh YH, Lii J-H (1989) Molecular mechanics. The MM3 force field for hydrocarbons. J Amer Chem Soc 111:8551–8566
5. Androulakis IP, Maranas CD, Floudas CA (1997) Global minimum potential energy conformations of oligopeptides. J Global Optim 11:1–34

6. Augspurger JD, Scheraga HA (1996) An efficient, differentiable hydration potential for peptides and proteins. J Comput Chem 17:1549–1558
7. Brooks BR, Bruccoleri RE, Olafson BD, States DJ, Swaminathan S, Karplus M (1983) CHARMM: A program for macromolecular energy minimization and dynamics calculations. J Comput Chem 4:187–217
8. Connolly ML (1983) Analytical molecular surface calculation. J Appl Crystallogr 16:548–558
9. Eisenhaber F, Argos P (1993) Improved strategy in analytic surface calculation for molecular systems: Handling of singularities and computational efficiency. J Comput Chem 14:1272–1280
10. Floudas CA, Klepeis JL, Pardalos PM (1998) Global optimization approaches in protein folding and peptide docking. In: DIMACS, vol 47. Amer Math Soc, Providence, pp 141–171
11. van Gunsteren WF, Berendsen HJC (1987) GROMOS. Groningen Mol Sim
12. Kang YK, Némethy G, Scheraga HA (1987) Free energies of hydration of solute molecules 1. Improvement of hydration shell model by exact computations of overlapping volumes. J Phys Chem 91:4105–4109
13. Klepeis JL, Androulakis IP, Ierapetritou MG, Floudas CA (1998) Predicting solvated peptide conformations via global minimization of energetic atom-to-atom interactions. Comput Chem Eng 22:765–788
14. Kollman PA (1993) Free energy calculations: Applications to chemical and biochemical phenomena. Chem Rev 93:2395–2417
15. Maranas CD, Androulakis IP, Floudas CA (1996) A deterministic global optimization approach for the protein folding problem. In: DIMACS, vol 23. Amer Math Soc, Providence, pp 133–150
16. Maranas CD, Floudas CA (1992) A global optimization approach for Lennard–Jones microclusters. J Chem Phys 97:7667–7677
17. Maranas CD, Floudas CA (1994) A deterministic global optimization approach for molecular structure determination. J Chem Phys 100:1247–1261
18. Maranas CD, Floudas CA (1994) Global minimum potential energy conformations of small molecules. J Global Optim 4:135–170
19. Némethy G, Gibson KD, Palmer KA, Yoon CN, Paterlini G, Zagari A, Rumsey S, Scheraga HA (1992) Energy parameters in polypeptides 10. Improved geometrical parameters and nonbonded interactions for use in the ECEPP/3 algorithm with application to proline containing peptides. J Phys Chem 96:6472–6484
20. Neumaier A (1997) Molecular modeling of proteins and mathematical prediction of protein structure. SIAM Rev 39:407–460
21. Oobatake M, Némethy G, Scheraga HA (1987) Accessible surface areas as a measure of the thermodynamic parameters of hydration of peptides. Proc Nat Acad Sci USA 84:3086–3090
22. Perrot G, Cheng B, Gibson KD, Vila J, Palmer KA, Nayeem A, Maigret B, Scheraga HA (1992) MSEED: A program for the rapid analytical determination of accessible surface areas and their derivatives. J Comput Chem 13:1–11
23. Vila J, Williams RL, Vásquez M, Scheraga HA (1991) Empirical solvation models can be used to differentiate native from near-native conformations of bovine pancreatic trypsin inhibitor. PROTEINS: Struct Funct Genet 10:199–218
24. Weiner SJ, Kollman PA, Case DA, Singh UC, Ghio C, Alagona G, Profeta S, Weiner P (1984) A new force field for molecular mechanical simulation of nucleic acids and proteins. J Amer Chem Soc 106:765–784
25. Wesson L, Eisenberg D (1992) Atomic solvation parameters applied to molecular dynamics of proteins in solution. Protein Sci 1:227–235

Multiple Objective Dynamic Programming

Michael M. Kostreva, Laura C. Lancaster
Department Math. Sci., Clemson University, Clemson, USA

MSC2000: 90C39, 90C31

Article Outline

Keywords
See also
References

Keywords

Dynamic programming; Multiple objective programming; Efficient set

Dynamic programming has been an area of active research since its introduction by R. Bellman [1]. More recently, with the recognition that many applied optimization problems require more than one objective, the study of multicriteria optimization has become a growing area of research. Included in this area of multicriteria optimization is the study of multiple objective dynamic programming (*MODP*). MODP was

first used to replace multiple objective linear programming (MOLP) where it was not applicable, such as in problems with discrete variables. Many of the techniques used are extensions of classical dynamic programming. The following is a discussion of some of the research that has been developed in the area of MODP.

Using multiple objective dynamic programming to find the 'shortest' path through a network with constant costs is one of the more straightforward uses of MODP. Work has been done on both forward and backward MODP in this area. First, we consider a general network containing a set of nodes $N = \{1, \ldots, n\}$ and a set of arcs $A = \{(i_0, i_1), (i_2, i_3), (i_4, i_5), \ldots\} \subset N \times N$ which indicates connections between nodes. Each arc (i, j) has an associated cost vector, $c_{ij} = (c_{ij1}, \ldots, c_{ijm}) \subset \mathbf{R}^m$. A path from node i_0 to i_p is the sequence of arcs $P = \{(i_0, i_1), \ldots, (i_{p-1}, i_p)\}$ where the first node of each arc is the same as the terminal node of the preceding arc and each node in the path is unique. Let Π_i be the set of all paths from node 1 to node i. The cost to traverse a path p in Π_i is $[c(p)] = \sum_{(i,j) \in p}[c_{ij}]$. A path in Π_i is *nondominated* if there is no other path p^* in Π_i with $[c(p^*)]_r \leq [c(p)]_r$ for $r = 1, \ldots, m$ and $[c(p^*)]r < [c(p)]_r$ for some $r \in \{1, \ldots, m\}$.

0	$k = 1$.
1	Evaluate S_i^k for all nodes using $S_i^k = \{c_{ij} + S_i^{k-1}\}$.
2	If $k < N$, set $k = k + 1$ and return to step 1; otherwise:
3	For each nondominated solution at each node determined in step 1 and for each r, $r = 1, \ldots, m$, define T^r as $T^r = \min_{i_N, \ldots, i_0} \sum_n c^r_{i_n j_{n-1}}$, where i_n is the originated node at stage n and I_n is the set of nodes that can be reached from node n.
4	Given weights $W^m \in \mathbf{R}^m_+$, compute the MINSUM as $\min \left[\sum_{r=1}^{m} \left\{ W^r \frac{\sum_{k=1}^{N} c_{ij} - T^r}{T^r} \right\} \right]$.

H.G. Daellenbach and C.A. DeKluyver [5] gave one of the earliest algorithms for backward MODP with constant costs, which finds nondominated paths from all nodes to the destination node. Their method is basically an extension of the principle of optimality to a multicriteria context. They state a *principle of Pareto optimality of MODP*: 'A nondominated policy has the property that regardless of how the process entered a given state, the remaining decisions must belong to a nondominated subpolicy.' Let S_i^k be the nondominated vector of objective values for a node i, exactly k links from its destination, t. Then the algorithm is given above.

The resulting S_i^k vectors give nondominated solutions for the network, but maybe not all of them. They solve an example in which the weights are not specified.

A few years later, R. Hartley [6] proposed a similar algorithm that also uses backward MODP to find all Pareto paths from all nodes in the network to a specified node. The algorithm is as follows:

Let $V_0(i) = \{\infty, \ldots, \infty\}$ for $k = 0, 1, \ldots$, and let $V_k(t) = \{0, \ldots, 0\}$.

$V_k(i) = \text{eff}[\cup \{c_{ij} + V_{k-1}(j): j \in \Gamma(i)\}]$ for $i \in N$ $(i \neq t)$ and $k = 1, 2, \ldots$, where $\Gamma(i)$ is the set of nodes such that $(i, j) \in A$. The 'eff' operator finds all nondominated vectors in the set. The associated paths must be handled separately.

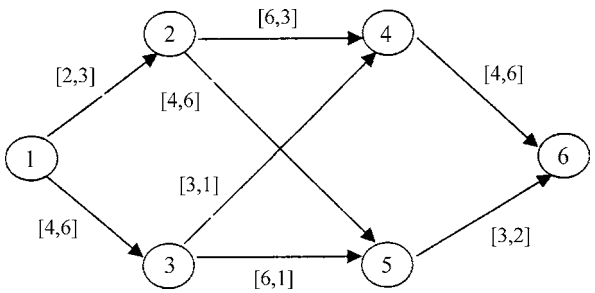
H.W. Corley and I.D. Moon [4] used forward MODP to find all nondominated paths from a specified node to all other nodes in a network with multiple constant costs. They assumed that the network contains no loops and that $c_{ij} \neq \{0, \ldots, 0\}$ for any $(i, j) \in A$. Letting $G_i^{(k)}$ be the set of vector costs of all Pareto paths from node 1 to node i containing k or fewer arcs, the algorithm follows:

1	Set $c_{ii} = (0, \ldots, 0)$, $i = 1, \ldots, n$ and $c_{ij} = (\infty, \ldots, \infty)$, $i \neq j$, if no arc exists from i to j. Set $k = 1$ and let $G_i^{(1)} = \{c_{1i}\}$, $i = 1, \ldots, n$.
2	For $i = 1, \ldots, n$, set $G_i^{k+1} = \text{Vmin} \cup_{j=1}^{n} \{c_{ij} + g_j^k : g_j^k \in G_j^{(k)}\}$.
3	If $G_i^{(k+1)} = G_i^{(k)}$, $i = 1, \ldots, n$, stop, otherwise go to step 4.
4	If $k = n - 1$, stop. Else, $k = k + 1$ and go to step 2.

Vmin is an operation that computes the vector costs of all nondominated paths in a set of vector costs. An algorithm for Vmin is given in their paper.

Multiple Objective Dynamic Programming, Table 1

	$k = 1$	$k = 2$	$k = 3$
$G_1^{(k)}$	$\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\end{pmatrix}$
$G_2^{(k)}$	$\begin{pmatrix}2\\3\end{pmatrix}$	$\begin{pmatrix}2\\3\end{pmatrix}$	$\begin{pmatrix}2\\3\end{pmatrix}$
$G_3^{(k)}$	$\begin{pmatrix}4\\6\end{pmatrix}$	$\begin{pmatrix}4\\6\end{pmatrix}$	$\begin{pmatrix}4\\6\end{pmatrix}$
$G_4^{(k)}$	$\text{V min}\left\{\begin{pmatrix}8\\6\end{pmatrix}, \begin{pmatrix}7\\7\end{pmatrix}\right\}$	$\left\{\begin{pmatrix}8\\6\end{pmatrix}, \begin{pmatrix}7\\7\end{pmatrix}\right\}$	$\left\{\begin{pmatrix}8\\6\end{pmatrix}, \begin{pmatrix}7\\7\end{pmatrix}\right\}$
$G_5^{(k)}$	$\text{V min}\left\{\begin{pmatrix}6\\9\end{pmatrix}, \begin{pmatrix}10\\7\end{pmatrix}\right\}$	$\begin{pmatrix}6\\9\end{pmatrix}, \begin{pmatrix}10\\7\end{pmatrix}$	$\begin{pmatrix}6\\9\end{pmatrix}, \begin{pmatrix}10\\7\end{pmatrix}$
$G_6^{(k)}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}$	$\text{V min}\left\{\begin{pmatrix}12\\12\end{pmatrix}, \begin{pmatrix}11\\13\end{pmatrix}, \begin{pmatrix}9\\11\end{pmatrix}, \begin{pmatrix}13\\9\end{pmatrix}\right\}$	$\begin{pmatrix}9\\11\end{pmatrix}, \begin{pmatrix}13\\9\end{pmatrix}$



Multiple Objective Dynamic Programming, Figure 1

Table 1 gives the results of the algorithm. The resulting Pareto optimal paths from node 1 to node 6 are {(1, 2), (2, 5), (5, 6)} and {(1, 3), (3, 5), (5, 6)}.

The following example uses the *Corley–Moon algorithm* to solve a dynamic routing problem for the network in Fig. 1.

Using multiple objective dynamic programming to find the shortest path through a network with time-dependent costs is considerably more complicated than MODP with constant costs. The monotonicity assumptions necessary for the principle of optimality in dynamic programming can easily be broken when dealing with time-dependent costs. Reaching a node later may be less costly than reaching it earlier. M.M. Kostreva and M.M. Wiecek [7] extended the work done by K.L. Cooke and E. Halsey [3] on dynamic programming with one time-dependent cost (travel time) to dynamic programming with multiple time-dependent costs. This method uses backward dynamic programming on a discrete time grid to find all nondominated paths from every node in the network to the destination node.

Assume the discrete time grid $S_T = \{t_0, \ldots, t_0 + \mathrm{T}\}$, $t_0 > 0$ and the cost functions $[c_{ij}(t)]_k > 0$, $(i, j) \in A$, for all $t \in S_T$. T is the upper bound on total time to travel from any node in the network to the destination node, N_d. Also assume that $[c_{ij}(t)]_1$ is the time to travel from node i to node j when the arrival time at node i is time t. For all $i \in N \setminus N_d$ and all $t \in S_T$, define $\{[F_i(t)]\}$ as the set of nondominated vectors associated with the paths that leave node i at time t and reach node N_d and define $\{[F_i(t)(k)]\}$ as the set of nondominated vectors associated with the paths that leave node i at time t and reach node N_d in at most $k + 1$ links before time $t_0 + T$, where $k = 0, 1, \ldots$. The following is the principle of optimality used for this algorithm: 'A nondominated path p, leaving node i at time $t \in S_T$ and reaching node N at or before time $t_0 + T$, has the property that for each node j lying on this path, a subpath p_1, that leaves node j at time $t_j \in S_T$, $t_j > t$, and arrives at node N at or before

time $t_0 + T$, is nondominated.' The algorithm is as follows:

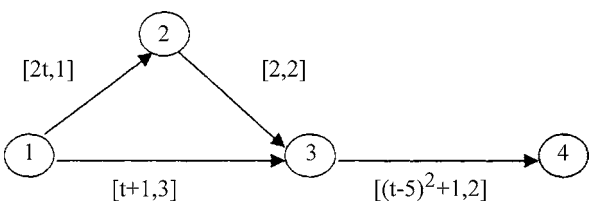
1. Find a time grid of discrete values $S_T = \{t_0, \ldots, t_0 + T\}$, $t_0 > 0$ and compute $[c_{ij}(t)]$ for all $t \in S_T$ and all $(i, j) \in A$.
2. Modify $[c_{ij}(t)]$ for all $t \in S_T$ and all $(i, j) \in A$ as follows:
$$[c_{ij}(t)]' = \begin{cases} [c_{ij}(t)] & \text{if } t+[c_{ij}(t)]_l \le t_0+T, \\ \infty & \text{if } t+[c_{ij}(t)]_l > t_0+T. \end{cases}$$
3. Find the initial array $[\{F_i(t)(0)\}]$, $i = 1, \ldots, N$, for all $t \in S_T$, where $\{[F_{N_d}(t)^{(0)}]\} = \{0\}$, and $\{[F_i(t)^{(0)}]\} = [c_{iN_d}(t)]'$ for $i \in N \backslash N_d$.
4. Find the arrays $[\{[F_i(t)^{(k)}]\}]$, $i = 1, \ldots N$, for all $t \in S_T$, for $k = 1, 2, \ldots$ as follows:
$$\{[F_i(t)^{(k)}]\} = \text{VMIN}\{[c_{ij}(t)]' + \{[F_j(t+[c_{ij}(t)]'_1)^{(k-1)}]\}\}, \quad i \in N \backslash \{N_d\}, \quad \{[F_i(t)^{(k)}]\} = \{0\}.$$
5. The sequence of sets $\{[F_i(t_0)^{(k)}]\}$, $k = 1, 2, \ldots$, converges to the set $\{[F_i(t_0)]\}$, the set of nondominated vectors associated with the paths that leave node i at time t_0 and reach node N_d.

The following example uses Algorithm One [7] to solve a dynamic routing problem for the network in Fig. 2. A grid of discrete values of time $S_{19} = \{1, 2, \ldots, 20\}$ for $t_0 = 1$ is established.

Table 2 shows the initial array and the two subsequent arrays. So, the set $\{\text{Eff}(E_I(t_0))\}$ of all nondominated paths that leave node 1, 2, and 3 at time $t_0 = 1$ are $\{(1, 2), (2, 3), (3, 4)\}$, $\{(2, 3), (3, 4)\}$, and $\{(3, 4)\}$.

Kostreva and Wiecek [7] also developed an algorithm which uses forward dynamic programming to find all nondominated paths from an origin node to every other node in the network without using a time grid. Thus, assume t is a continuous variable, $t \ge 0$, and $[c_{ij}(t)]_1 > 0$. An assumption must be made about the cost functions so that the principle of optimality will hold for these networks: For any arc $(i, j) \in A$ and all $t_1, t_2 \ge 0$, if $t_1 \le t_2$, then:

a) $t_1 + [c_{ij}(t_1)]_1 \le t_2 + [c_{ij}(t_2)]$ 1, and
b) $[c_{ij}(t_1)]_r \le [c_{ij}(t_2)]_r$ for all $r \in \{2, \ldots, m\}$. Assuming the cost functions are monotone increasing with respect to time satisfies this assumption.



Multiple Objective Dynamic Programming, Figure 2

1. Find the initial vector $\{[G_j^{(0)}]\}$, $j = 1, \ldots, N$, where $\{[G_1^{(0)}]\} = \{0\}$ and $\{[G_j^{(0)}]\} = [c_{1j}(0)]$, $j = 2, \ldots, N$.
2. Calculate the vectors $\{[G_j^{(k)}]\}$, $j = 1, \ldots, N$, for $k = 1, 2, \ldots$, as follows:
$$\{[G_j^l(t_l)^{(k)}],\ l =, \ldots, N_j\} = \text{VMIN}\{[G_i^n(t^n)^{(k-1)}] + [c_{ij}(t^n)], \quad n = 1, \ldots, N_i\}, \quad j = 2, \ldots, N, \quad \{[G_1^l(t^l)^{(k)}], l = 1\} = \{0\}.$$
3. $\{[G_j^{(k)}]\}$, $k = 1, 2, \ldots$, converges to $\{[G_j]\}$, the set of vector costs of all nondominated paths which leave the origin node at time $t = 0$ and lead to node j.

Assume that node 1 is the origin node. For nodes $j = 2, \ldots, N$, let $[G_j^u(t^u)^{(k)}]$ be the vector cost of the nondominated path u which is of at most k links leaving the origin node at time $t = 0$ and leading to node j, where t^u is the arrival time of this path at node j. Also, let $\{[G_j^{(k)}]\}$ be the set of vector costs of all nondominated paths which are of at most k links leaving the origin node at time $t = 0$ and leading to node j, where N_j is the number of nondominated paths. Let $\{[G_j]\}$ be the set of vector costs of all nondominated paths which leave the origin node at time $t = 0$ and lead to node j. The algorithm is as listed above.

Another way to get around the monotonicity assumption of dynamic programming is to use generalized dynamic programming techniques. See [2] for

Multiple Objective Dynamic Programming, Table 2
Sequence of arrays

Time	$[\{[F_I(t)^{(0)}]\}]$	$[\{[F_I(t)^{(1)}]\}]$	$[\{[F_I(t)^{(2)}]\}]$
1	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{17}{2}\binom{0}{0}$	$\binom{7}{5}\binom{7}{4}\binom{17}{2}\binom{0}{0}$	$\binom{5}{5}\binom{7}{4}\binom{17}{2}\binom{0}{0}$
2	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{10}{2}\binom{0}{0}$	$\binom{4}{5}\binom{4}{4}\binom{10}{2}\binom{0}{0}$	$\binom{4}{5}\binom{4}{4}\binom{10}{2}\binom{0}{0}$
3	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{5}{2}\binom{0}{0}$	$\binom{9}{5}\binom{3}{4}\binom{5}{2}\binom{0}{0}$	$\binom{9}{5}\binom{3}{4}\binom{5}{2}\binom{0}{0}$
4	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{2}{2}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{4}{4}\binom{2}{2}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{4}{4}\binom{2}{2}\binom{0}{0}$
5	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{1}{2}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{7}{4}\binom{1}{2}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{7}{4}\binom{1}{2}\binom{0}{0}$
6	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{2}{2}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{12}{4}\binom{2}{2}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{12}{4}\binom{2}{2}\binom{0}{0}$
7	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{5}{2}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{5}{2}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{5}{2}\binom{0}{0}$
8	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{10}{2}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{10}{2}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{10}{2}\binom{0}{0}$
9	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{0}{0}$
...	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{0}{0}$	$\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{\infty}{\infty}\binom{0}{0}$

a way to use generalized DP with a multicriteria preference function. Basically, generalized DP uses a weaker principle of optimality than Bellman's famous version [1]. Generalized DP finds partial solutions that may lead to optimal solutions even though locally they are not optimal solutions according to the preference function.

In [2] generalized DP is applied to the multicriteria best path problem. Assuming node 1 to be the origin and node N to be the destination, let Π be the set of all paths in the network. Let

$$P(j) = \{p \in \Pi\colon\ i_1 = 1,\ i_n = j\}$$

be the set of all paths from the origin to node j. Let

$$X(j) = \{p \in \Pi\colon\ i_1 = j,\ i_n = N\}$$

be the set of all paths from node j to the destination node. The vector cost along each arc is called an *arc length vector*, $l_{ij} = (l_{ij}^1, \ldots, l_{ij}^m) \in \mathbf{R}^m$. A path length function $z\colon \Pi \to \mathbf{R}^m$ assigns a path length vector to every path $p \in \Pi$ where $\circ$ is a binary operator on $\mathbf{R}^m$:

$$z(p) = l_{1,2} \circ \cdots \circ l_{i_n - 1, i_n}.$$

Thus, each different objective can have a different binary operation. For example, distance would have an additive binary operator and probabilities would have

a multiplicative binary operator. Let

$$Z(j) = \{z(p):\ p \in P(j)\}$$

be the set of all length vectors of all paths from the origin to node j. A multicriteria preference function u: $\mathbf{R}^m \to \mathbf{R}$ is defined on the set of path length vectors. The objective is to maximize this preference function. The monotonicity assumption says that for all $z, z' \in Z(j), u(z) \leq u(z') \Rightarrow u(z \circ l_{jk}) \leq u(z' \circ l_{jk})$ for all j, $k \in S$ such that $(j, k) \in T$. Unfortunately, with multiobjective problems this assumption is easily violated. Generalized DP tries to get around this monotonicity assumption by having local preference relations defined as $\rho_j \subseteq Z(j) \times Z(j)$: for $z, z' \in Z(j)$, where $z \rho_j z'$ implies that any subpath from the origin to node j whose length is z cannot be used in a path to produce a better overall path from the origin to the destination node than using the subpath from the origin to node j whose length is z'. So, subpath length vector z' is more locally preferred even though subpath length vector z may be globally preferred, $u(z') \leq u(z)$. So, for $z, z' \in Z(j)$, $z \rho_j z'$ if and only if $\exists p' \in X(j)$ such that $u(z \circ z(p)) \leq u(z' \circ z(p'))$ for all $p \in X(j)$. These local preference relations are used to form the *weak principle of optimality*. An optimal path must be composed of subpaths that can be part of an optimal path.

Unfortunately, in order to get these preference relations one would have to complete all paths from every node in the network. Since this is too computationally intense, the preference relations are relaxed to the refining local preference relations $\prec_j$ where $z \prec_j z'$ implies $z \rho_j z'$. Using $\prec_j$ avoids having to find the entire relation ρ_j. Using this relation means that a larger set of maximal path length vectors will be kept by using ρ_j than if ρ_j were used. A maximal path length vector is a vector where there does not exist another vector at that state that is strictly more preferred. Let

$$\text{maxl}(X, \rho) = \left\{x \in X:\ \exists x' \in X:\ x \rho x' \text{ and } x' \rho x\right\}.$$

The following are the equations of generalized DP:

$$f(1) = \{z_1\},$$
$$f(j) = \text{maxl}\left(\cup_{(i,j)\in A}(f(i) \circ l_{ij}) \prec_j\right)$$
$$\text{for } j = 2, \ldots, N,$$

where $\{f(i) \circ l_{ij}\} = \{z \circ l_{ij}:\ z \in f(i)\}$.

When the monotonicity assumptions are satisfied, the $\prec_j$ relation can be replaced with the multicriteria preference function, u, thus reducing to the conventional DP problem. However, when the monotonicity assumption does not hold the $\prec_j$ relation must be defined by trying exploit any special structures of each individual problem. Also, using dynamic programming to find the entire Pareto optimal set can be seen as another special case of generalized DP where $z_k \geq z_k{}'$ for all $k = 1, \ldots, m \Rightarrow z \prec_j z'$ (assuming minimization of each criteria).

The subject of multiple objective dynamic programming has developed into a viable body of knowledge capable of providing solutions to applied problems in which trade-offs among objectives is important. Among the multiple objective techniques, it is distinctive in its ability to provide the entire Pareto optimal set. To gain such an advantage, one must be willing to perform computationally intensive operations on large sets of vectors.

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Neuro-dynamic Programming

References

1. Bellman RE (1957) Dynamic programming. Princeton University Press, Princeton

2. Carraway RL, Morin TL, Moskowitz H (1990) Generalized dynamic programming for multicriteria optimization. Europ J Oper Res 44:95–104
3. Cooke KL, Halsey E (1966) The shortest route through a network with time-dependent internodal transit times. J Math Anal Appl 14:493–498
4. Corley HW, Moon ID (1985) Shortest paths in networks with vector weights. J Optim Th Appl 46:79–86
5. Daellenbach HG, DeKluyver CA (1980) Note on multiple objective dynamic programming. J Oper Res Soc 31:591–594
6. Hartley R (1984) Vector optimal routing by dynamic programming. In: Serafini P (ed) Mathematics of Multiobjective Optimization, pp 215–224
7. Kostreva MM, Wiecek MM (1993) Time dependency in multiple objective dynamic programming. J Math Anal Appl 173:289–307

Multiple Objective Programming Support

Pekka Korhonen[1,2]
[1] Internat. Institute Applied Systems Analysis, Laxenburg, Austria
[2] Helsinki School Economics and Business Adm., Helsinki, Finland

MSC2000: 90C29

Article Outline

Keywords
Introduction
A Multiple Objective Programming Problem
Generating Nondominated Solutions
  A Linear Scalarizing Function
  A Chebyshev-Type Scalarizing Function
Solving Multiple Objective Problems
Example of a Decision Support System: VIG
Numerical Illustrations
Conclusion
See also
References

Keywords

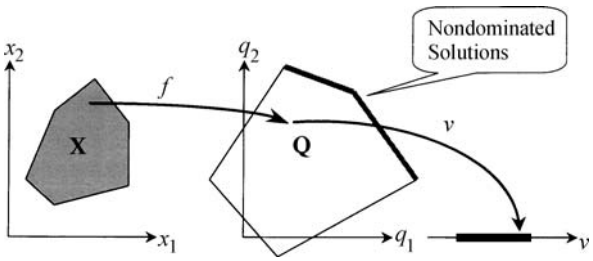
Multiple criteria decision making; Multiple objective programming; Multiple objective programming support; Scalarizing function; Value function

This article gives a brief introduction into multiple objective programming support. We will overview basic concepts, formulations, and principles of solving multiple objective programming problems. To solve those problems requires the intervention of a decision-maker. That's why behavioral assumptions play an important role in multiple objective programming. Which assumptions are made affects which kind of support is given to a decision maker. We will demonstrate how a free search type approach can be used to solve multiple objective programming problems.

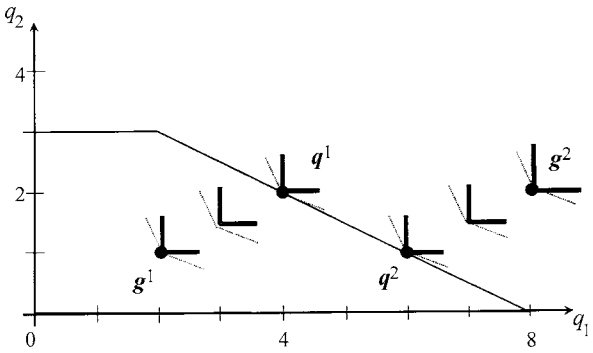
Introduction

Before we can consider the concept of *multiple objective programming support* (MOPS), we have to first explain the concept of *multiple criteria decision making* (MCDM). Even if there is a variation of different definitions, most researchers working in the field might accept the following general definition: Multiple Criteria Decision Making (MCDM) refers to the solving of decision and planning problems involving multiple (generally conflicting) criteria. 'Solving' means that a decision-maker (DM) will choose one 'reasonable' alternative from among a set of available ones. It is also meaningful to define that the choice is irrevocable. For an MCDM problem it is typical that no unique solution for the problem exists. Therefore to find a solution for MCDM problems requires the intervention of a decision-maker (DM). In MCDM, the word 'reasonable' is replaced by the words 'efficient/nondominated'. They will be defined later on.

Actually the above definition is a strongly simplified description of the whole (multiple criteria) decision making process. In practice, MCDM problems are not often so well-structured, that they can be considered just as a choice problem. Before a decision problem is ready to be 'solved', the following questions require a lot of preliminary work: How to structure the problem? How to find essential criteria? How to handle uncertainty? These questions are by no means outside the interest area of MCDM-researchers. The outranking method by B. Roy [17] and the AHP (the analytic hierarchy process) developed by T.L. Saaty [18] are examples of the MCDM-methods, in which a lot of effort is devoted to problem structuring. Both methods are well known and widely used in practice. In both methods,



Multiple Objective Programming Support, Figure 1
A variable, criterion, and value space



Multiple Objective Programming Support, Figure 2
Illustrating the projection of a feasible and an infeasible aspiration level point onto the nondominated surface

the presence of multiple criteria is an essential feature, but the structuring of a problem is an even more important part of the solution process.

When the term 'support' is used in connection with MCDM, we may adopt a broad perspective and refer with the term to all research associated with the relationship between the problem and the decision-maker. In this article we take a narrower perspective and focus on a very essential supporting problem in multiple criteria decision making: How to assist a DM to find the 'best' solution from among a set of available 'reasonable' alternatives, when the alternatives are evaluated by using several criteria? Available alternatives are assumed to be defined explicitly or implicitly by means of a mathematical model. The term *multiple objective programming* is usually used to refer to dealing with this kind of model.

The following considerations are general in the sense that usually it is not necessary to specify how the alternatives are defined. It is enough to assume that they belong to set Q. However, in Fig. 1 and Fig. 2 and the numerical example we consider a multiple objective linear programming model in which all constraints and objectives are defined using linear functions.

The article consists of seven sections. In Sect. "A Multiple Objective Programming Problem", we give a brief introduction to some foundations of multiple objective programming. How to generate potential 'reasonable' solutions for a DM's evaluation is considered in Sect. "Generating Nondominated Solutions", and in Sect. "Solving Multiple Objective Problems", we will review general principles to solve a multiple objective programming problem. In Sect. "Example of a Decision Support System: VIG", a multiple criteria decision support system VIG is introduced, and a numerical example is solved in Sect. "Numerical Illustrations". Concluding remarks are given in Sect. "Conclusion".

A Multiple Objective Programming Problem

A multiple objective programming (MOP) problem in a so-called *criterion space* can be defined as follows:

$$\begin{cases} \text{'max'} & q \\ \text{s.t.} & q \in Q, \end{cases} \tag{1}$$

where set $Q \subset \mathbf{R}^k$ is a so-called *feasible region* in a k-dimensional criterion space $\mathbf{R}^k$. The set Q is of special interest. Most considerations in multiple objective programming are made in a criterion space.

Set Q may be convex/nonconvex, bounded/unbounded, precisely known or unknown, consist of finite or infinite number of alternatives, etc. When Q consists of a finite number of elements which are explicitly known in the beginning of the solution process, we have an important class of problems which may be called e. g. (multiple criteria) *evaluation problems*. Sometimes those problems are referred to as *discrete multiple criteria problems* or *selection problems* (for a survey see for example [16]).

When the number of alternatives in Q is infinite and not countable, the alternatives are usually defined using a mathematical model formulation, and the problem is called continuous. In this case we say that the alternatives are only implicitly known. This kind of problem is referred as a *multiple criteria design problem* (the terms 'evalution' and 'design' are adopted from A. Arbel) or a *continuous multiple criteria problem*. In this case, the

set Q is not specified directly, but by means of decision variables as usually done in single optimization problems:

$$\begin{cases} \max & q = f(x) = (f_1(x), \ldots, f_k(x)) \\ \text{s.t.} & x \in X, \end{cases} \tag{2}$$

where $X \subset \mathbf{R}^n$ is a *feasible set* and $f : \mathbf{R}^n \to \mathbf{R}^k$. The space $\mathbf{R}^n$ is called a *variable space* (see Fig. 1). The functions f_i, $i = 1, \ldots, k$, are *objective functions*. The feasible region Q can now be written as $Q = \{q: q = f(x), x \in X\}$.

The MOP-problem has seldom a unique solution, i. e. an optimal solution that simultaneously maximizes all objectives. Conceptually the multiple objective mathematical programming problem may be regarded as a *value* (*utility*) *function* maximization program:

$$\begin{cases} \max & v(q) \\ \text{s.t.} & q \in Q, \end{cases} \tag{3}$$

where v is a real-valued function, which is strictly increasing in the criterion space and defined at least in the feasible region Q. It is mapping the feasible region into a one-dimensional value space (see Fig. 1). Function v specifies the DM's preference structure over the feasible region. However, the key assumption in multiple objective programming is that v is unknown. Generally, if the value function is estimated explicitly, the system is considered to be in the MAUT category, see for example [7], (MAUT stands for multiple attribute utility theory) and can then be solved without any interaction of the DM. Typically, MAUT-problems are not even classified under the MCDM-category. If the value function is implicit (assumed to exist but is otherwise unknown) or no assumption about the value function is made, the system is usually classified under MCDM [2] or MOP.

Solutions of the MOP-problems are all those alternatives which can be the solutions of some value function v: $Q \to \mathbf{R}$. Those solutions are called *efficient* or *nondominated* depending on the space where the alternatives are considered. The term nondominated is used in the criterion space and efficient in the variable space. (Some researchers use the term efficient to refer to efficient and nondominated solutions without making any difference.) Any choice from among the set of efficient (nondominated) solutions is an acceptable and 'reasonable' solution, unless we have no additional information about the DM's preference structure.

Nondominated solutions are defined as follows:

Definition 1 In (1), $q^* \in Q$ is *nondominated* if and only if there does not exist another $q \in Q$ such that $q \geq q^*$ and $q \neq q^*$.

Definition 2 In (1), $q^* \in Q$ is *weakly nondominated* if and only if there does not exist another $q \in Q$ such that $q > q^*$.

Correspondingly, efficient solutions are defined as follows:

Definition 3 In (2), $x^* \in X$ is *efficient* if and only if there does not exist another $x \in X$ such that $f(x) \geq f(x^*)$ and $f(x) \neq f(x^*)$.

Definition 4 In (2), $x^* \in X$ is *weakly efficient* if and only if there does not exist another $x \in X$ such that $f(x) > f(x^*)$.

The final ('best') solution $q \in Q$ of the problem (1) is called the *most preferred solution*. It is a solution preferred by the DM to all other solutions. At the conceptual level, we may think it is the solution maximizing an (unknown) value function in problem (3). How to find it? That is the problem we now proceed to consider.

Unfortunately, the above characterization of the most preferred solution is not very operational, because no system can enable the DM to simultaneously compare the final solution to all other solutions with an aim to check if it is really the most preferred or not. It is also as difficult to maximize a function we do not know. Some properties for a good system are, for example, that it makes the DM convinced that the final solution is the most preferred one, does not require too much time from the DM to find the final solution, to give reliable enough information about alternatives, etc. Even if it is impossible to say which system provides the best support for a DM for his multiple criteria problem, all proper systems have to be able to recognize, generate and operate with nondominated solutions. To generate nondominated solutions for the DM's evaluation is thus one key issue in multiple objective pro-

gramming. In the next section, we will consider some principles.

Generating Nondominated Solutions

Despite many variations among different methods of generating nondominated solutions, the ultimate principle is the same in all methods: a single objective optimization problem is solved to generate a new solution or solutions. The objective function of this single objective problem may be called a *scalarizing function*, according to [25]. It typically has the original objectives and a set of parameters as its arguments. The form of the scalarizing function as well as what parameters are used depends on the assumptions made concerning the DM's preference structure and behavior.

Two classes of parameters are widely used in multiple objective optimization:

1) weighting coefficients for objective functions; and
2) reference/aspiration/reservation levels for objective function values.

Based on those parameters, there exist several ways to specify a scalarizing function. An important requirement is that this function completely characterizes the set of nondominated solutions:

> for each parameter value, all solution vectors are nondominated, and for each nondominated criterion vector, there is at least one parameter value, which produces that specific criterion vector as a solution

(see, for theoretical considerations, e. g. [26]).

A Linear Scalarizing Function

A classic method to generate nondominated solutions is to use the weighted-sums of objective functions, i. e. to use the following linear scalarizing function:

$$\max \left\{ \lambda' f(x) \colon \; x \in X \right\}. \tag{4}$$

If $\lambda > 0$, then the solution vector x of (4) is efficient, but if we allow that $\lambda \geq 0$, then the solution vector is weakly-efficient. (see, e. g. [21, p. 215; 221]). Using the parameter set $\Lambda = \{\lambda \colon \lambda > 0\}$ in the weighted-sums linear program we can completely characterize the efficient set provided the constraint set is convex. However, Λ is an open set, which causes difficulties in a mathematical optimization problem. If we use $\mathrm{cl}(\Lambda) = \{\lambda \colon \lambda \geq 0\}$ instead, the efficiency of x cannot be guaranteed anymore. It is surely weakly-efficient, and not necessarily efficient. When the weighted-sums are used to specify a scalarizing function in multiple objective linear program (MOLP) problems, the optimal solution corresponding to nonextreme points of X is never unique. The set of optimal solutions always consists of at least one extreme point, or the solution is unbounded. In early methods, a common feature was to operate with weight vectors $\lambda \in \mathbf{R}^k$, limiting considerations to efficient extreme points (see, e. g., [29]).

A Chebyshev-Type Scalarizing Function

Currently, most solution methods are based on the use of a so-called Chebyshev-type scalarizing function first proposed by A. Wierzbicki [25]. We will refer to this function by the term *achievement* (*scalarizing*) *function*. The achievement (scalarizing) function projects any given (feasible or infeasible) point $g \in \mathbf{R}^k$ onto the set of nondominated solutions. Point g is called a *reference point*, and its components represent the desired values of the objective functions. These values are called *aspiration levels*.

The simplest form of achievement function is:

$$s(g, q, w) = \max_{k \in K} \frac{g_k - q_k}{w_k}, \tag{5}$$

where $w > 0 \in \mathbf{R}^k$ is a (given) vector of weights, $g \in \mathbf{R}^k$, and $q \in Q = \{f(x) \colon x \in X\}$. By minimizing $s(g, q, w)$ subject to $q \in Q$, we find a weakly nondominated solution vector q^* (see, e. g. [25,26]). However, if the solution is unique for the problem, then q^* is nondominated. If $g \in \mathbf{R}^k$ is feasible, then $q^* \in Q$, $q^* \geq g$. To guarantee that only nondominated (instead of weakly nondominated) solutions will be generated, more complicated forms for the achievement function have to be used, for example:

$$s(g, q, w, \rho) = \max_{k \in K} \left[\frac{g_k - q_k}{w_k} \right] + \rho \sum_{i=1}^{k} (g_i - q_i), \tag{6}$$

where $\rho > 0$. In practice, we cannot operate with a definition 'any positive value'. We have to use a prespecified value for ρ. Another way is to use a lexicographic formulation [10].

The applying of the scalarizing function (6) is easy, because given $g \in \mathbf{R}^k$, the minimum of $s(g, v, w, \rho)$ is found by solving the following LP-problem:

$$\begin{cases} \min & \epsilon + \rho \sum_{i=1}^{k} (g_i - q_i) \\ \text{s.t.} & x \in X \\ & \epsilon \geq \frac{g_i - q_i}{w_i}, \quad i = 1, \ldots, k. \end{cases} \tag{7}$$

Problem (7) can be further written as:

$$\begin{cases} \min & \epsilon + \rho \sum_{i=1}^{k} (g_i - q_i) \\ \text{s.t.} & x \in X \\ & q + \epsilon w - z = g \\ & z \geq 0. \end{cases} \tag{8}$$

To illustrate the use of the achievement scalarizing function, consider a two-criteria problem with a feasible region having four extreme points (0, 0), (0, 3), (2, 3), (8, 0), as shown in Fig. 2. In Fig. 2, the thick solid lines describe the indifference curves when $\rho = 0$ in the achievement scalarizing function. The thin dotted lines stand for the case $\rho > 0$. Note that the line from (2, 3) to (8, 0) is nondominated and the line from (0, 3) to (2, 3) (excluding the point (2, 3)) is weakly-nondominated, but dominated. Let us assume that the DM first specifies a feasible aspiration level point $g^1 = (2, 1)$. Using a weight vector $w = [2, 1]'$, the minimum value of the achievement scalarizing function (−1) is reached at a point $v^1 = (4, 2)$ (cf. Fig. 2). Correspondingly, if an aspiration level point is infeasible, say $g^2 = (8, 2)$, then the minimum of the achievement scalarizing function (+ 1) is reached at point $v^2 = (6, 1)$. When a feasible point dominates an aspiration level point, then the value of the achievement scalarizing function is always negative; otherwise it is nonnegative. It is zero, if an aspiration level point is weakly-nondominated.

Solving Multiple Objective Problems

Several dozen procedures and computer implementations have been developed from the 1970s onwards to address both *multiple criteria evaluation* and *design problems*. The multiple objective decision procedures always requires the intervention of a DM at some stage in the solution process. A popular way to involve the DM in the solution process is to use an interactive approach.

The specifics of these procedures vary, but they have several common characteristics. For example, at each iteration, a solution, or a set of solutions, is generated for a DM's examination. As a result of the examination, the DM inputs information in the form of trade-offs, pairwise comparisons, aspiration levels, etc. (see [20] for a more detailed discussion). The responses are used to generate a presumably, improved solution. The ultimate goal is to find the most preferred solution of the DM. Which search technique and termination rule is used is heavily dependent on the underlying assumptions postulated about the behavior of the DM and the way in which these assumptions are implemented. In MCDM-research there is a growing interest in the behavioral realism of such assumptions.

Based on the role that the value function (3) is supposed to play in the analysis, we can classify the assumptions into three categories:

1) Assume the existence of a value function v, and assess it explicitly.
2) Assume the existence of a stable value function v, but do not attempt to assess it explicitly. Make assumptions of the general functional form of the value function.
3) Do not assume the existence of a stable value function v, either explicitly, or implicitly.

The first assumption is adopted in multi-attribute utility or decision analysis (see, e. g. [7]). Interactive software implementing such approaches on personal computers exists.

The second assumption was a basic paradigm used in interactive multiple criteria approaches in the 1970s. A classical example is the GDF-method [3]. DM's responses to specific questions were used to guide the solution process towards an 'optimal' or 'most preferred' solution (in theory), assuming that the DM behaves according to some specific (but unknown) underlying value function (see for surveys, e. g. [5,20,21], and [24]). Interactive software that implements such systems for a computer have often been developed by the authors of the above procedures for experimental purposes.

The approaches based on the assumption on 'no stable value/utility function' typically operate with a DM's aspiration levels regarding the objectives on the feasible region. The aspiration levels are projected via minimizing so called achievement scalarizing functions (6) [23,25]. No specific behavioral assumptions e. g. transitivity are necessary.

In essence, this approach seeks to help the DM more or less freely to search the set of efficient solutions. Interactive software that implements such systems for a computer have been developed like ADBASE [22], DIDAS [14], VIG [8], and VIMDA [9]. For an excellent review of several interactive multiple criteria procedures, see [21]. Other well-known books that provides a deeper background and additional references especially in the field of multiple objective optimization include [1,4,5,6,19,27] and [28].

Multiple objective linear programming (MOLP) is the most commonly studied problem in *multiple criteria decision making* (MCDM). Most solution methods are developed for this problem.

Example of a Decision Support System: VIG

Today, many systems use aspiration level projections, where the projection is performed using Chebyshev-type achievement scalarizing functions as explained above. These functions can be controlled either by varying weights (keeping aspiration levels fixed) or by varying the aspiration levels (keeping weights fixed). Instead of aspiration levels, some algorithms asks the DM to specify the reservation levels for the criteria (see, e. g. [15]).

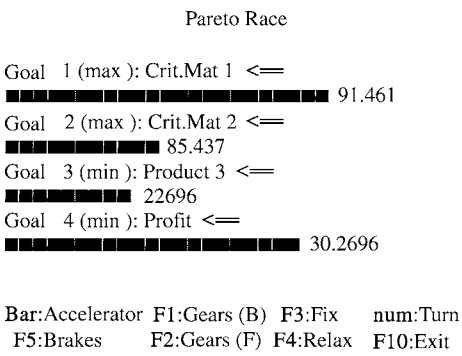
An achievement scalarizing function projects one aspiration (reservation) level point at a time onto the nondominated frontier. By parametrizing the function, it is possible to project the whole vector onto the nondominated frontier as originally proposed by [11]. The vector to be projected is called a *reference direction vector* and the method *reference direction method*, correspondingly. When a direction is projected onto the nondominated frontier, a curve traversing across the nondominated frontier is obtained. Then an interactive line search is performed along this curve. The idea enables the DM to make a continuous search on the nondominated frontier. The corresponding mathematical model is a simple modification from the original model (8) developed for projecting one point:

$$\begin{cases} \min & \epsilon + \rho \sum_{i=1}^{k} (g_i - q_i) \\ \text{s.t.} & x \in X \\ & q + \epsilon w - z = g + tr, \quad z \geq 0, \end{cases} \tag{9}$$

where t: $0 \to \infty$ and $r \in \mathbf{R}^k$ is a reference direction. In the original approach, a reference direction was specified as a vector starting from the current solution and passing through the aspiration levels. The DM was asked to give aspiration levels for criteria.

The original reference direction approach has been further developed into many directions. First, [12] improved upon the original procedure by making the specification of a reference direction dynamic. The dynamic version was called *Pareto race*. In Pareto race, the DM can freely move in any direction on the nondominated frontier he/she likes, and no restrictive assumptions concerning the DM's behavior are made. Furthermore, the objectives and constraints are presented in a uniform manner. Thus, their role can also be changed during the search process. The method and its implementation is called Pareto race. The whole software package consisting of Pareto race is called VIG.

In Pareto race, a reference direction r is determined by the system on the basis of preference information received from the DM. By pressing number keys corresponding to the ordinal numbers of the objectives, the DM expresses which objectives he/she would like to improve and how strongly. In this way he/she implicitly specifies a reference direction. Figure 3 shows the



Multiple Objective Programming Support, Figure 3
Example Pareto race screen

Pareto race interface for the search, embedded in the VIG software [8].

Thus Pareto race is a visual, dynamic, search procedure for exploring the nondominated frontier of a multiple objective linear programming problem. The user sees the objective function values on a display in numeric form and as bar graphs, as he/she travels along the nondominated frontier. The keyboard controls include an accelerator, gears, brakes, and a steering mechanism. The search on the nondominated frontier is like driving a car. The DM can, e. g., increase/decrease the speed, make a turn and brake at any moment he/she likes.

To implement those features, Pareto race uses certain control mechanisms, which are controlled by the following keys:

- (SPACE) BAR, an 'accelerator': Proceed in the current direction at constant speed.
- F1, 'gears (backward)': Increase speed in the backward direction.
- F2, 'gears (forward)': Increase speed in the forward direction.
- F3, 'fix': Use the current value of objective i as the worst acceptable value.
- F4, 'relax': Relax the 'bound' determined with key F3.
- F5, 'brakes': Reduce speed.
- F10, 'exit'.
- num, 'turn': Change the direction of motion by increasing the component of the reference direction corresponding to the goal's ordinal number $i \in [1, k]$ pressed by DM.

An example of the Pareto race screen is given in Fig. 3. The screen is associated with the numerical example described in the next section.

Pareto race does not specify restrictive behavioral assumptions for a DM. He/she is free to make a search on the nondominated surface, until he/she believes that the solution found is his/her most preferred one.

Pareto race is only suitable for solving moderate size problems. When the size of the problem becomes large, computing time makes the interactive mode inconvenient. To solve large scale problems [13] proposed a method based on Pareto race. An (interactive) free search is performed to find the most preferred direction. Based on the direction, an nondominated curve can be generated in a batch mode if desired.

Numerical Illustrations

For illustrative purposes, we will consider the following production planning problem, where a decision maker (DM) tries to find the 'best' product-mix for three products: Product 1, Product 2, and Product 3. The production of these products requires the use of one machine (mach. hours), man-power (man hours), and two critical materials (crit. mat. 1 and crit. mat. 2). Selling the products results in profit (profit). Assume that the DM describes his/her decision problem as follows:

> Of course, I would like to make as much profit as possible. Because it is difficult and quite expensive to obtain critical materials, I would like to use them as little as possible, but never more than I have presently in storage (96 units of each). Only one machine is used to produce the products. It operates without any problems for at least 9 hours. The length of the regular working day is 10 hours. People are willing to work overtime which is costly and they are tired the next day. Therefore, if possible, I would like to avoid it. Finally, product 3 is very important to a major customer, and I cannot totally exclude it from the production plan.

The traditional single objective programming considers the problem as a profit maximization problem. The other 'requirements' are taken as constraints. The multiple objective programming takes a 'softer' perspective. We may, for instance, consider the problem as a four objective problem. The DM would like to make as much profit as possible, but simultaneously, he/she would like to use those two critical materials as little as possible, and in addition to maximize the use of product 3. Machine hours and man hours are considered as constraints, but during the search process the role of

Multiple Objective Programming Support, Table 1
The coefficient matrix of the production planning problem

	Prod. 1	Prod. 2	Prod. 3
mach. hours	1.5	1	1.6
man hours	1	2	1
crit. mat. 1	9	19.5	7.5
crit. mat. 2	7	20	9
profit	4	5	3

Multiple Objective Programming Support, Table 2
A sample of solutions for the multiple criteria problem

	I	II	III	IV
Objectives:				
crit. mat. 1	91.46	94.50	93.79	90.00
crit. mat. 2	85.44	88.00	89.15	84.62
profit	30.27	31.00	30.42	29.82
product 3	0.23	0.00	0.50	0.44
Constraints:				
mach. hours	9.00	9.00	9.00	9.00
man hours	9.73	10.00	10.00	9.62
Decision Variables:				
product 1	3.88	4.00	3.45	3.71
product 2	2.81	3.00	3.03	2.74
product 3	0.23	0.00	0.50	0.44

constraints and objectives may also be changed, if necessary.

We assume that the problem can be modeled as an MOLP-model. The coefficient matrix of the problem is given in Table 1.

Thus, we have the following multiple objective linear programming model:

$$\begin{aligned}
&\text{crit. mat. 1:} && 9P_1 + 19.5P_2 + 7.5P_3 && \to \min\\
&\text{crit. mat. 2:} && 7P_1 + 20P_2 + 9P_3 && \to \min\\
&\text{profit:} && 4P_1 + 5P_2 + 3P_3 && \to \max\\
&\text{product 3:} && P_3 && \to \max
\end{aligned}$$

subject to:

$$\begin{aligned}
&\text{mach. hours:} && 1.5P_1 + P_2 + 1.6P_3 && \le 9\\
&\text{man hours:} && P_1 + 2P_2 + P_3 && \le 10
\end{aligned}$$

The problem has no unique solution. Using the Pareto race (see Fig. 3) or any other software developed for multiple objective programming enables a DM to search nondominated solutions. Which solution he/she will choose as a final one depends entirely on his/her own preferences. Actually, all sample solutions except solution II are somehow consistent with his/her statement above. In solution II, product 3 is excluded from the production plan.

Conclusion

In this article, we have provided an overview on multiple objective programming support. The emphasis was how to find the most preferred alternative from among a set of reasonable (nondominated) alternatives. This kind of the approach is unique for the multiple criteria decision making. We have left other features like structuring the problem, finding relevant criteria etc. beyond this presentation. They are important, but also relevant in the considerations of any decision support system.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Cohon J (1978) Multiobjective programming and planning. Acad. Press, New York
2. Dyer J, Fishburn P, Wallenius J, Zionts S (1992) Multiple criteria decision making, multiattribute utility theory – The next ten years. Managem Sci 38:645–654
3. Geoffrion A, Dyer J, Feinberg A (1972) An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. Managem Sci 19:357–368
4. Haimes Y, Tarvainen K, Shima T, Thadathil J (1990) Hierarchical multiobjective analysis of large-scale systems. Hemisphere, Washington

5. Hwang C, Masud A (1979) Multiple objective decision making – Methods and applications: A state-of-the-art survey. Springer, Berlin
6. Ignizio J (1976) Goal programming and extensions. D.C. Heath, Lexington
7. Keeney RL, Raiffa H (1976) Decisions with multiple objectives: Preferences and value tradeoffs. Wiley, New York
8. Korhonen P (1987) VIG - A visual interactive support system for multiple criteria decision making. Belgian J Oper Res Statist Comput Sci 27:3–15
9. Korhonen P (1988) A visual reference direction approach to solving discrete multiple criteria problems. Europ J Oper Res 34:152–159
10. Korhonen P, Halme M (1996) Using lexicographic parametric programming for searching a nondominated set in multiple objective linear programming. J Multi-Criteria Decision Anal 5:291–300
11. Korhonen P, Laakso J (1986) A visual interactive method for solving the multiple criteria problem. Europ J Oper Res 24:277–287
12. Korhonen P, Wallenius J (1988) A Pareto race. Naval Res Logist 35:615–623
13. Korhonen P, Wallenius J, Zionts S (1992) A computer graphics-based decision support system for multiple objective linear programming. Europ J Oper Res 60:280–286
14. Lewandowski A, Kreglewski T, Rogowski T, Wierzbicki A (1989) Decision support systems of DIDAS family (Dynamic Interactive Decision Analysis and Support. In: Lewandowski A, Wierzbicki A (eds) Aspiration Based Decision Support Systems. Springer, Berlin, pp 21–47
15. Michalowski W, Szapiro T (1992) A bi-reference procedure for interactive multiple criteria programming. Oper Res 40:247–258
16. Olson D (1996) Decision aids for selection problems. Ser Oper Res. Springer, Berlin
17. Roy B (1973) How outranking relation helps multiple criteria decision making. In: Cochrane J, Zeleny M (eds) Multiple Criteria Decision Making. University South Carolina Press, Columbia, pp 179–201
18. Saaty T (1980) The analytic hierarchy process. McGraw-Hill, New York
19. Sawaragi Y, Nakayama H, Tanino T (1985) Theory of multiobjective optimization. Acad. Press, New York
20. Shin W, Ravindran A (1991) Interactive multiple objective optimization: Survey I – Continuous case. Comput Oper Res 18:97–114
21. Steuer RE (1986) Multiple criteria optimization: Theory, computation, and application. Wiley, New York
22. Steuer R (1992) Manual for the ADBASE multiple objective linear programming package. Department Management Sci, University Georgia, Athens
23. Steuer R, Choo E-U (1983) An interactive weighted Tchebycheff procedure for multiple objective programming. Math Program 26:326–344
24. White D (1990) A bibliography on the applications of mathematical programming multiple-objective methods. J Oper Res Soc 41:669–691
25. Wierzbicki A (1980) The use of reference objectives in multiobjective optimization. In: Fandel G, Gal T (eds) Multiple Objective Decision Making, Theory and Application. Springer, Berlin
26. Wierzbicki A (1986) On the completeness and constructiveness of parametric characterizations to vector optimization problems. OR Spektrum 8:73–87
27. Yu PL (1985) Multiple criteria decision making: Concepts, techniques, and extensions. Plenum, New York
28. Zeleny M (1982) Multiple criteria decision making. McGraw-Hill, New York
29. Zionts S, Wallenius J (1976) An interactive programming method for solving the multiple criteria problem. Managem Sci 22:652–663

Multiplicative Programming

TAKAHITO KUNO
University Tsukuba, Ibaraki, Japan

MSC2000: 90C26, 90C31

Article Outline

Keywords
See also
References

Keywords

Optimization; Nonconvex minimization; Low-rank nonconvexity

Multiplicative programming refers to a class of optimization problems containing products of real-valued functions in the objective and/or in the constraints. A product of convex functions is called a *convex multiplicative function*; similar definitions hold for *concave* and *linear* multiplicative functions. Multiplicative functions appear in various areas, including microeconomics [4], VLSI chip design [10] and modular design [2]. Especially in multiple objective decision making, they play important roles [3]. A typical example is a bond portfolio optimization studied in [7],

where a number of performance indices such as average coupon rate, average terminal yields and average length of maturity associated with a portfolio (a bundle of assets) are to be optimized (either minimized or maximized) subject to a number of constraints. One handy approach to simultaneously optimizing multiple objectives without a common scale is to optimize the geometric mean, or equivalently the product of these objectives. Thus, we are led to consider a multiplicative programming problem.

The simplest subclass of multiplicative programming problems is a *linear multiplicative program*, which is a quadratic program of minimizing a product of two affine functions $\mathbf{c}_1^\top \mathbf{x} + c_{10}$, $i = 1, 2$, over a polytope $D \subset \mathbf{R}^n$:

$$\begin{cases} \min & f(\mathbf{x}) = (\mathbf{c}_1^\top \mathbf{x} + c_{10})(\mathbf{c}_2^\top \mathbf{x} + c_{20}) \\ \text{s.t.} & \mathbf{x} \in D. \end{cases} \tag{1}$$

This problem was first studied by K. Swarup [13] many years ago, but had attracted little attention until the late 1980s when an intensive research was undertaken [8,12,14]. In general, the objective function f is indefinite; it is quasiconcave on a region where the signs of $c_i^\top \mathbf{x} + c_{i0}s$ are the same, but quasiconvex on a region where the signs are different [1,8]. Therefore, to solve (1), we need to solve a quasiconcave minimization problem:

$$\begin{cases} \min & f(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D \cap S, \end{cases} \tag{2}$$

and a quasiconcave maximization problem:

$$\begin{cases} \max & f(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D \cap S, \end{cases} \tag{3}$$

where $S = \{\mathbf{x} \in \mathbf{R}^n : \mathbf{c}_i^\top \mathbf{x} + c_{i0} \geq 0, i = 1, 2\}$. While (2) belongs to multi-extremal global optimization [6] and is known to be *NP*-hard [11] (cf. also ► Complexity Classes in Optimization; ► Computational Complexity Theory), problem (3) can be solved using a standard convex minimization technique because maximizing $f(\mathbf{x})$ amounts to minimizing a convex function $-\log(\mathbf{c}_1^\top \mathbf{x} + c_{10}) - \log(\mathbf{c}_2^\top \mathbf{x} + c_{20})$. For the same reason as (3), certain linear programs with additional linear multiplicative constraints, e. g. the modular design problem with $x_i\, y_j \geq b_{ij}$ [2], can be handled within the framework of convex programming, if $x_i, y_j \geq 0$.

A generalization of (1) is a *convex multiplicative program*, which minimizes a product of several convex functions $f_i(\mathbf{x})$, $i = 1, \ldots, p$, over a compact convex set $D \subset \mathbf{R}^n$:

$$\begin{cases} \min & f(\mathbf{x}) = \prod_{i=1}^{p} f_i(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D. \end{cases} \tag{4}$$

In most of the existing solutions to (4), the convex functions f_i are assumed to be nonnegative-valued on D. When $f_i(\mathbf{x}') = 0$ for some i and for some $\mathbf{x}' \in D$, the minimum value of (4) is zero; and $\mathbf{x}'$ is a globally optimal solution. We may therefore assume for each i that $f_i(\mathbf{x}) > 0$ for all $\mathbf{x} \in D$. If f is a concave multiplicative function instead of a convex one, the problem is equivalent to a concave minimization problem because $\log f(\mathbf{x}) = \sum_i^{p=1} \log f_i(\mathbf{x})$ is concave. The convex multiplicative program (4) itself can also be transformed into a concave minimization problem (cf. ► Concave Programming), though f is not a concave function. For example, introducing additional variables y_i, $i = 1, \ldots, p$, we have an equivalent problem:

$$\begin{cases} \min & \sum_{i=1}^{p} \log y_i \\ \text{s.t.} & \mathbf{x} \in D \\ & f_i(\mathbf{x}) \leq y_i, \quad i = 1, \ldots, p, \\ & \mathbf{y} \geq 0. \end{cases} \tag{5}$$

The number p of $f_i s$ is often very small in comparison with the dimension n of $\mathbf{x}$; e. g. five or so in applications to multiple objective optimization. Owing to this *low-rank nonconvexity* [9], problem (5) can be solved far more efficiently than the usual concave minimization problem of the same size.

In addition to (1) and (4), there are a number of studies on problems with *generalized convex multiplicative functions* of the forms $f(\mathbf{x}) = \prod_{i=1}^{p} f_i(\mathbf{x}) + g(\mathbf{x})$ and $f(\mathbf{x}) = \sum_{i=1}^{p} f_{2i-1}(\mathbf{x})\, f_{2i}(\mathbf{x}) + g(\mathbf{x})$, where the $f_i s$ and g are convex functions. These are all nonconvex minimization problems, each of which has an enormous number of local minima. Nevertheless, algorithms de-

veloped in the 1990s can locate a globally optimal solution in a reasonable amount of time, by exploiting special structures of f such as low-rank nonconvexity. A comprehensive review of the algorithms are given by H. Konno and T. Kuno in [5].

See also

- ► Global Optimization in Multiplicative Programming
- ► Linear Programming
- ► Multiparametric Linear Programming
- ► Parametric Linear Programming: Cost Simplex Algorithm

References

1. Avriel M, Diewert WE, Schaible S, Zang I (1988) Generalized concavity. Plenum, New York
2. Evans DH (1963) Modular design: A special case in nonlinear programming. Oper Res 11:637–647
3. Geoffrion M (1967) Solving bicriterion mathematical programs. Oper Res 15:39–54
4. Henderson JM, Quandt RE (1971) Microeconomic theory. McGraw-Hill, New York
5. Horst R, Pardalos PM (1995) Handbook of global optimization. Kluwer, Dordrecht
6. Horst R, Tuy H (1993) Global optimization: deterministic approaches. second Springer, Berlin
7. Konno H, Inori M (1989) Bond portfolio optimization by bilinear fractional programming. J Oper Res Japan 32:143–158
8. Konno H, Kuno T (1992) Linear multiplicative programming. Math Program 56:51–64
9. Konno H, Thach PT, Tuy H (1997) Optimization on low rank nonconvex structures. Kluwer, Dordrecht
10. Maling K, Mueller SH, Heller WR (1982) On finding most optimal rectangular package plans. Proc 19th Design Automation Conf, pp 663–670
11. Matsui T (1996) NP-hardness of linear multiplicative programming and related problems. J Global Optim 9:113–119
12. Pardalos PM (1990) Polynomial time algorithms for some classes of constrained nonconvex quadratic problems. Optim 21:843–853
13. Swarup K (1966) Programming with indefinite quadratic function with linear constraints. Cahiers CERO 8:132–136
14. Thoai NV (1991) A global optimization approach for solving the convex multiplicative programming problem. J Global Optim 1:341–357

Multi-Quadratic Integer Programming: Models and Applications

W. Art Chaovalitwongse[1], Xiaozheng He[2], Anthony Chen[3]

[1] Department of Industrial and Systems Engineering, Rutgers University, Piscataway, USA

[2] Department of Civil Engineering, University of Minnesota, Minneapolis, USA

[3] Department of Civil and Environmental Engineering, Utah State University, Logan, USA

MSC2000: 65K05, 90C11, 90C20

Article Outline

Keywords and Phrases
Introduction
Multi-Quadratic Integer Program
Applications
  Bilinear Problem
  Minimax Problem
  Mixed Integer Problem
Solution Techniques
Linear Forms of MQIP
References

Keywords and Phrases

Quadratic programming; Quadratic constraints; Mixed-integer program; Linearization

Introduction

In this contribution, we consider Multi-Quadratic Programming (MQP) problems, where the objective function is a quadratic function and the feasible region is defined by a finite set of quadratic and linear constraints. They can be formulated as follows:

$$\begin{aligned} &\min x^T Q x + c^T x \\ &\text{s.t. } x^T A_j x + B_j x \le b_j\,, \quad j = 1, \ldots, m \quad x \ge 0\,, \end{aligned} \tag{1}$$

where A_j is an $(n \times n)$ matrix corresponding to the mth quadratic constraint, and B_j is the jth row of the $(m \times n)$ matrix B.

MQP plays an important role in modeling many diverse problems. The MQP encompasses many other optimization problems since it provides a much improved model compared to the simpler linear relaxation of a problem. Indeed, linear mixed 0-1, fractional, bilinear, bilevel, generalized linear complementarity, and many more programming problems are or can easily be reformulated as special cases of MQP. However, there are theoretical and practical difficulties in the process of solving such problems. However, very large linear models can be solved efficiently; whereas MQP problems are in general $\mathcal{NP}$-hard and numerically intractable. The problem of finding a feasible solution is also $\mathcal{NP}$-hard. This is because MQP is a generalization of the linear complementarity problem [29]. The nonlinear constraints in MQP define a feasible region which is in general neither convex nor connected. Moreover, even if the feasible region is a polyhedron, optimizing the quadratic objective function is strongly $\mathcal{NP}$-hard as the resulting problem is considered to be the disjoint bilinear programming problem. Therefore, finding a finite and exact algorithm that solves large MQP problems is impractical. Even for the convex case (when Q and A_j are positive semidefinite), there are very few algorithms for solving MQP problems. However, the MQP constitutes an important part of mathematical programming problems, arising in various practical applications including facility location, production planning, VLSI chip design, optimal design of water distribution networks, and most problems in chemical engineering design.

The MQP was first introduced in the seminal paper of Kuhn and Tucker [31]. Later on, the case of MQP with a single quadratic constraint in the problem was discussed in [55,56]. The first general approach for solving MQP problems was proposed in [12], where the following two Lagrange functions for MQP are considered:

$$L_1(x,\mu) = x^T Q x + c^T x + \sum_{j=1}^{m} \mu_j (x^T A_j x - B_j x - b_j)\,,$$

$$L_2(x,\mu,\lambda) = L_1(x,\mu) - \lambda_i x_i\,,$$

where μ and λ are the multipliers for the quadratic and bound constraints respectively. A cutting plane algorithm was applied to solve this problem; that is, the algorithm solves a sequence of linear master problems that minimize a piecewise linear function constructed from the Lagrange functions for constant x, and a primal problem with either an unconstrained quadratic function (using $L_2(x,\mu,\lambda)$) or a quadratic function over the nonnegative orthant (using $L_1(x,\mu)$) [21].

Multi-Quadratic Integer Program

In this contribution we consider a multi-quadratic integer programming (MQIP) problem with bilevel variables. This problem is a more specific case of MQP. Recently, multi-quadratic zero-one programming problems were proved equivalent to mixed-integer programming problems [16]. In that work, a quadratic zero-one programming was initially proved equivalent to a mixed integer programming problem. Then, the result was extended to the case multi-quadratic programming case.

Throughout this paper, we consider a multi-quadratic zero-one programming problem, which has following form:

$$P_1 :: \min f(x) = x^T A x\,,\ \text{s.t. } Bx \geq b\,,\ x^T C x \geq \alpha\,,$$
$$x \in \{0,1\}^n\,,\ \alpha \text{ is a constant}\,.$$

Notice $x^T C x \geq \alpha$ essentially represent the same the quadratic constraints as $x^T A_j x + B_j x \leq b_j$ in problem (1), due to the binary variables' property $x_i x_i = x_i$.

Applications

Bilinear Problem

Each n-dimensional MQP problem can be easily transformed to a $2n$-dimensional bilinear problem. A strategy for reducing the necessary dimension of the resulting bilinear program is also proposed [7,28]. However, on the other hand, bilinear optimization problems are nothing else but a special instance of MQP. Pooling problems in petrochemistry, the modular design problem introduced in [17], in particular the multiple modular design problem [7,18] or the more general modularization of product sub-assemblies [46], and special classes of structured stochastic games [20] are only some examples of the wide range of applications of bilinear programming problems. Another large class of optimization problems are problems with linear or quadratic functions additionally involving Boolean

variables (i. e., variables $x_i \in \mathbb{R}$ with the constraint $x_i \in \{0, 1\}$). Another widely explored problem is the problem of packing $n \in \mathbb{N}$ equal circles in a square, which can be transformed to a MQP problem. One looks for the maximum radius r of n non-overlapping circles contained in the unit square. This problem is equivalent to a MQP problem with a linear objective function and concave quadratic constraints.

Minimax Problem

A related class of global optimization problems are minimax location problems [42], which also lead to quadratic constraints. Production planning and portfolio optimization are examples where so-called chance constrained linear programs occur. These are problems, looking similar to linear programs. However, the matrix describing the linear constraints of such problems is not deterministic, it is a stochastic one. Under certain restrictive assumptions it is possible to transform these stochastic constraints to deterministic quadratic constraints [42], such that in general a problem of type MQP is obtained. In [8] it is shown that nonconvex MQP problems can be used for the examination of special instances of nonlinear bilevel programming problems. Other applications of MQP include the fuel mixture problem encountered in the oil industry [43] and also placement and layout problems in integrated circuit design [9,10].

Mixed Integer Problem

As described in the previous section, MQP problem can be easily linearized to a mixed integer zero-one problem with the same problem size. In theory and practice, the linearization technique proposed in [16] has been shown to be superior than other conventional linearization techniques. In medical applications, multi-quadratic zero-one problems were used to model epileptic brains for electrode selection problems. Basically, multi-quadratic zero-one problems were solved to identify the location (electrode) sites of the brain that can detect seizure pre-cursors (predict seizures) [30,34,36]. In order to operate in real time, multi-quadratic zero-one problems were linearized to a mixed integer zero-one problem, which is much faster to solve in practice.

Hence there are many applications of MQP. Whether the MQP is in practice applicable for solving, for example, problems resulting from integer programming problems, depends on the numerical efficiency of the solution method that is used. Up to now only few methods for solving the considered general case of MQP were proposed in the literature. Most of them result from methods being developed for other more general problem classes. In the next section we will discuss some of the solution techniques.

Solution Techniques

There are many different techniques proposed for solving this type of problems, most of them are of branch and bound type or some type of linearization techniques [4,25,26,27,37,38,39,57]. A disadvantage of the standard linearization technique is the additional variables for each product $x_i x_j$, in which the number of new variables is $O(n^2)$, where n is the number of initial 0-1 variables [4,25,26,57]. The method proposed in [16] needs only $O(kn)$ additional continuous variables, where k is the number of quadratic constraints, and the number of initial 0-1 variables remains the same. A branch-and-bound algorithm for solving MQP problems (and other more general problems), when the objective function is separable and the constraint set is linear, was introduced in [19]. The method evolves solving bounding convex envelope approximating problems over successive partitions of the feasible region. This method was later extended to deal with nonconvex constraints but it generates a number of infeasible solutions and does not, in general, converge in a finite number of iterations [53]. An algorithm for the solution to linear problems with an additional reverse convex constraint was proposed in [15]. The algorithm involves partitioning the feasible region into subsets contained in cones originating at an infeasible vertex of the polytope formed by the linear constraints while ensuring that an interior point of the feasible region is contained in each partition. Later on, an algorithm for the solution to problems with concave objective functions and separable quadratic constraint was proposed in [8]. The algorithm uses piecewise linear approximation for the quadratic constraints and solves a MQP problem as a mixed 0-1 linear problem. This algorithm is similar to the solu-

tion approaches for concave quadratic problems [40] and for indefinite quadratic problems [35]. During the last decade, several authors are interested in some special cases of MQP. Also, many extensions of MQP have been discussed in the literature. The problem of minimizing an indefinite quadratic objective subject to two-sided indefinite quadratic constraints was discussed in [54]. Under suitable assumptions, they derived necessary and sufficient optimality conditions and gave some conditions for the existence of solutions for this nonconvex program. While several methods have been suggested for solving MQP problems, numerical solutions of the general problem are still rarely available in the literature. By using a double duality argument, under suitable assumptions, the MQP is proved to be equivalent to a convex program [6]. In addition, a problem with a concave quadratic function is proved to be equivalent to a minimax convex problem, and thus can be solved in polynomial time via interior-points methods. The property is no longer true when Q is an indefinite quadratic function [6].

Linear Forms of MQIP

As aforementioned, MQP problems have a close relationship with mixed integer zero-one problem by applying linearization schemes, which have been explored for decades. Although the existing linearization schemes originally were developed for QP instead of MQP, they could be easily applied to MQP, since the quadratic constraints in MQP could be reformulated by using the same technique in linearization considered for the quadratic objective. This section will provide a brief view of major linearization schemes and their applications on MQP problems.

No matter what specific reformulation of the linearization schemes, the ideas are the same as replacing the quadratic product $x_i x_j$ by additional variables. Currently existing linearization schemes were developed in four phases.

The prototype of linearization technique arose in 1960s, proposed by Watters [57] and Zangwill [58] (see also Fortet [22,23]). This approach introduces additional binary variables w_{ij} for replacement of the products $x_i x_j$ and additional constraints, $x_i + x_j - w_{ij} \le 1$ and $x_i + x_j \ge 2w_{ij}, \forall i, j$, for a guarantee of correct replacement. Taken this approach, the MQP P_1 is transformed as following form:

$$
\begin{aligned}
MIP_1 :: \min f(x,w) &= \sum_i a_{ii} x_i \\
&+ \sum_i \sum_{j>i} (a_{ij} + a_{ji}) w_{ij}, \text{ s.t. } Bx \ge b, \\
&\sum_i c_{ii} x_i + \sum_i \sum_{j>i} (c_{ij} + c_{ji}) w_{ij} \ge \alpha, \\
&x_i + x_j - w_{ij} \le 1, \forall i, j, \; x_i + x_j \ge 2w_{ij}, \\
&\forall i, j, \; x \in \{0,1\}^n, w_{ij} \in \{0,1\} \; \alpha \text{ is a constant} \\
&\text{and } A = (a_{ij}), C = (c_{ij}) .
\end{aligned}
$$

In this formulation, the quadratic products $x_i x_j$ in objective and constraints, $x^T Ax$ and $x^T Cx$, have been similarly replaced by additional binary variables w_{ij}, and the formulation is consistent with original P_1 by additional constraints, $x_i + x_j - w_{ij} \le 1$ and $x_i + x_j \ge 2w_{ij}, \forall i, j$. Following this seminal work, Glover and Woolsey [25] provided more concise zero-one linear programming formulations, where reformulation rules are given under difference conditions to reduce the numbers of additional constraints and additional variables.

In the second phase development of linearization techniques, researchers recognized the additional binary variables w_{ij} in MIP_1 could be relaxed by continuous ones. Such linearization schemes include the models developed in Glover [24], Glover and Woolsey [26], and Rhys [45]. One scheme with close relationship of the linearization prototype was provided in Glover and Woolsey [26], which introduces additional cut constraints $x_i \ge w_{ij}$ and $x_j \ge w_{ij}, \forall i, j$ enforcing the additional continuous variables w_{ij} to be binary. However, this technique doubles the number of additional constraints added and thereafter enlarges the size of original MQP problems. A straightforward generalization of $x_i \ge x_i x_j, \forall j$ generated their further improvement of this technique in [26] that used alternative concise constraints $(n-i)x_j \ge \sum_{(j>i)} w_{ij}, \forall i$ to enforce the additional variables to be binary, with somewhat fewer constraints. Applying such linearization technique, the MQP P_1 has the following representation:

$$
\begin{aligned}
MIP_2 :: \min f(x,w) &= \sum_i a_{ii} x_i \\
&+ \sum_i \sum_{j>i} (a_{ij} + a_{ji}) w_{ij}, \text{ s.t. } Bx \ge b,
\end{aligned}
$$

$$\sum_i c_{ii}x_i + \sum_i \sum_{j>i}(c_{ij}+c_{ji})w_{ij} \geq \alpha,$$
$$x_i + x_j - w_{ij} \leq 1, \forall i, j, \ (n-i)x_j$$
$$\geq \sum_{(j>i)} w_{ij}, \forall i, \ x \in \{0,1\}^n, w_{ij} \geq 0$$
$$\alpha \text{ is a constant and } A = (a_{ij}), C = (c_{ij}).$$

The main difference between MIP_1 and MIP_2 is the continuity of w_{ij} and the smaller number of additional constraints.

Beyond the linearization technique in [26], Glover first noticed Petersen's work [41], where the cross products in the model are considered by their upper and lower bounds. Following this idea, Glover, in [24], firstly proposed a linearization technique introducing different continuous variables w_i to the pioneer research. In his linearization scheme, the additional continuous variables are defined by $w_i = x_i \sum_j a_{ij}x_j$, where x_i are binary variables in original model and a_{ij} are quadratic coefficients in the objective function. Further define the lower and upper bounds of $\sum_j a_{ij}x_j$ by $A_i^- = \sum_j \{a_{ij} | a_{ij} < 0\}$ and $A_i^+ = \sum_j \{a_{ij} | a_{ij} > 0\}$, respectively. Taking the cross products and binary variables into consideration, the additional inequalities $A_i^+ x_i \geq w_i \geq A_i^- x_i$ and $\sum_j a_{ij}x_j - A_i^-(1-x_i) \geq w_i \geq \sum_j a_{ij}x_j - A_i^+(1-x_i)$ provide the equivalence of original QP model. Applied such linearization technique, the MQP P_1 has a different structure as follows:

$$MIP_3 :: \min f(w) = \sum_i w_i, \text{ s.t. } Bx \geq b,$$
$$A_i^+ x_i \geq w_i \geq A_i^- x_i, \forall i, \sum_j a_{ij}x_j - A_i^-(1-x_i)$$
$$\geq w_i \geq \sum_j a_{ij}x_j - A_i^+(1-x_i) \forall i \sum_i v_i \geq \alpha,$$
$$C_i^+ x_i \geq v_i \geq C_i^- x_i, \forall i, \sum_j c_{ij}x_j - C_i^-(1-x_i)$$
$$\geq v_i \geq \sum_j c_{ij}x_j - C_i^+(1-x_i) \forall i, \ x \in \{0,1\}^n,$$
$$\alpha \text{ is a constant}.$$

Notice, in MIP_3, the quadratic constraint $x^T Cx \geq \alpha$ is replaced by a series of inequalities $\sum_i v_i \geq \alpha, C_i^+ x_i \geq v_i \geq C_i^- x_i, \forall i, \ \sum_j c_{ij}x_j - C_i^-(1-x_i) \geq v_i \geq \sum_j c_{ij}x_j - C_i^+(1-x_i) \forall i$, which follow the definitions in [24] as: $v_i = x_i \sum_j c_{ij}x_j$, $C_i^- = \sum_j \{c_{ij} | c_{ij} < 0\}$, and $C_i^+ = \sum_j \{c_{ij} | c_{ij} > 0\}$. Compared MIP_3 with MIP_1 and MIP_2, the most important improvement of this linearization technique is that the numbers of additional variables and constraints reduce from $O(n^2)$ to $O(n)$. Some recent papers [1,2] proposed further-improved linearization techniques based on the strategy of Glover's technique, either providing concise formulation or generating tighter upper/lower bounds.

The linearization techniques in the third phase development considered the transformation from the direction of tightness instead of problem size. One typical technique included in this category is the famous Sherali–Adams Reformulation-Linearization Technique (as RLT in short) [3,4,5,50,51,52] which provides wide-range applications. The development milestones of RLT can be found in a recent memorial paper written by Sherali [47]. Interested readers could follow this paper to find the development details of the linearization scheme.

Some practical applications of linearization technique in early 1980s (e. g. [13] considered for solving notorious quadratic assignment problem) generated the experiences that the linearization techniques are practically inefficient although they may have small problem size. Such experience intrigued some researchers to provide better LP structures with tighter bounds, which offer better computational efficiencies, rather than to pursue smaller problem size. The linearization technique shown in [3] provides a structure having tighter bounds for zero-one QP. The transformation happens not only replacing the cross products $x_i x_j$ in the model but also reconstructing the constraints to obtain the tightness. The example given in [3] not only includes the additional constraints and continuous variables, but reconstructs the linear constraints by multiplying x_j and $1 - x_j$, respectively. Applying this linearization technique to MQP, P_1 is transformed as follows:

$$MIP_4 :: \min f(x,w) = \sum_i a_{ii}x_i$$
$$+ \sum_i \sum_{j>i}(a_{ij}+a_{ji})w_{ij}, \text{ s.t. } \sum_i B_{ki}x_i - \beta_k$$
$$\geq \sum_{i<j} B_{ki}w_{ij} + \sum_{i>j} B_{ki}w_{ji} + (B_{kj} - \beta_k)x_j$$
$$\geq 0,$$

$$\forall j,k,\ \sum_i c_{ii}x_i + \sum_i \sum_{j>i}(c_{ij}+c_{ji})w_{ij} \geq \alpha,$$
$$x_i + x_j - w_{ij} \leq 1, \forall i,j,\ x_i \geq w_{ij}, x_j \geq w_{ij}$$
$$\forall i,j,\ x \in \{0,1\}^n, w_{ij} \geq 0,$$
where α is a constant and $A = (a_{ij})$,
$C = (c_{ij})$,
$B = (B_{ij}), b = (\beta_k)$.

Notice the linear constraints $Bx \geq b$ are reconstructed as by multiplying x_j and $1 - x_j$ and then have much more complicated but tighter representations. The authors also provided the rigorous proof that the construction is tighter than the linearizaiton provided by Glover [24]. Other than that, this formulation uses the inequalities $x_i \geq w_{ij}$ and $x_j \geq w_{ij}$ instead of $(n-i)x_j \geq \sum_{(j>i)} w_{ij}$ which will weaken the model's tightness as pointed out by the authors.

Using this idea of multiplying x_j and $1 - x_j$ to the feasible set Adams and Sherali [4] provided a linearization strategy to more general MIP with cross products between continuous and binary variables. Comparisons were also provided between the RLT strategy and the linearization techniques in Watters [57], Zangwill [58], Petersen [41], Glover and Woolsey [25,26], and Glover [24]. Along with this direction, Sherali and Adams [49,50,51] generated a hierarchy of relaxations for zero-one polynomial problems. This relaxation strategy generalizes the idea in [3] by introducing a select set of d-degree polynomial terms or factors, where d is an integer less than the number of binary variables. Multiplying the feasible set by d-degree polynomial terms, as the authors showed, obtains an equivalent reformulation, for each $d = 1, \ldots, n$, which can enforce the binary restrictions on the original x variables. And these papers also proved that, when $d = n$, the resulting linear system characterizes the convex hull of feasible solutions, and therefore is tighter than any other linearization techniques.

The most recent development, as the final phase, of linearization technique is proposed by Chaovalitwongse et al. [16]. The authors took the dual variables into account, and proposed a new linearization technique based on KKT optimality conditions. Their approach was originally considered for MQP, and is not hard to be utilized for zero-one QP problems. The transformation of MQP P_1 using this linearization strategy can be shown as follows.

$$MIP_5 :: \min g(s,x) = e^T s - Me^T x, \text{ s.t.}$$
$$Ax - y - s + Me = 0,\ Bx \geq b,$$
$$y \leq 2M(e-x), Cx - z + M'e \geq 0,$$
$$e^T z - M'e^T x \geq \alpha,\ z \leq 2M'x,$$
$$x \in \{0,1\}^n,\ y_i,\ s_i,\ z_i \geq 0,$$
where $M' = \|C\|_\infty$ and $M = \|A\|_\infty$.

Notice the additional variables including s, y and z, which are introduced from the Lagrangian function of MQP.

Theorem 1 *P_1 has an optimal solution x^0 iff there exist y^0, s^0, z^0 such that (x^0, y^0, s^0, z^0) is an optimal solution of MIP_5.*

Proof 1 See [16]. □

To conclude all the linearization schemes shown herein, we provide a table aggregating the numbers of additional variables and constraints for these techniques as a brief comparison. Assuming we have k linear constraints $\sum_i B_{ji}x_i \geq b_j, j = 1, \ldots, k$ and m quadratic constraints $x^T C_j x \geq \alpha_j, j = 1, \ldots, m$, in an MQP. Also assume that the number of binary variables $n \gg k$ and $n \gg m$. Then the number of additional variables and constraints of the linearized forms applying different techniques can be shown in the table as follows:

Models	P_1	MIP_1	MIP_2	MIP_3	MIP_4	MIP_5
Additional constraints	0	$O(n^2)$	$O(n^2)$	$O(nm)$	$O(n^2)$	$O(nm)$
Addiontal variables	0	$O(n^2)$	$O(n^2)$	$O(nm)$	$O(n^2)$	$O(nm)$
Total constraints	$O(m+k)$	$O(n^2)$	$O(n^2)$	$O(nm)$	$O(n^2)$	$O(nm)$
Total variables	$O(n)$	$O(n^2)$	$O(n^2)$	$O(nm)$	$O(n^2)$	$O(nm)$

Notice that the problem size is not the only reason of computational efficiencies. The tightness of linearization schemes, as pointed out by [47], may significantly change the effectiveness of the techniques for MQP.

In terms of solution methods, there are many studies in the literature dealing with the MQP. Most of them apply a technique called semidefinite programming to

solve the problem. Specifically, these approaches include special branch-and-bound [9,10,32,44], branch-and-cut [11], lift-and-project [11], and the state-of-the-art Interior Point method [14,33]. Some of them have been applied in the commercial software package, e. g., the solvers BARON and CPLEX in GAMS.

References

1. Adams WP, Forrester RJ (2005) A simple recipe for concise mixed 0-1 linearizations. Oper Res Lett 33:55–61
2. Adams WP, Forrester RJ, Glover FW (2004) Comparison and enhancement strategies for linearizing mixed 0-1 quadratic programs. Discret Optim 1:99–120
3. Adams WP, Sherali HD (1986) A tight linearization and an algorithm for zero-one quadratic programming problems. Manag Sci 32:1274–1290
4. Adams WP, Sherali HD (1990) Linearization strategies for a class of zero-one mixed integer programming problems. Oper Res 38(2):217–226
5. Adams WP, Sherali HD (1993) Mixed-integer bilinear programming problems. Math Program 59(3):279–305
6. Aharon B-T, Teboulle M (1996) Hidden convexity in some nonconvex quadratically constrained quadratic programming. Math Program 72:51-639
7. Al-Khayyal FA (1992) Generalized bilinear programming: Part I. Models, applications and linear programming relaxation. Eur J Oper Res 60:306–314
8. Al-Khayyal FA, Horst R, Pardalos PM (1992) Global optimization of concave functions subject to separable quadratic constraints: An application to bilevel programming. Ann Oper Res 34:125–147
9. Al-Khayyal FA, Larsen C, van Voorhis T (1995) A relaxation method for nonconvex quadratically constrained quadratic programs. J Glob Optim 6:215–230
10. Al-Khayyal FA, van Voorhis T (1996) Accelerating convergence of branch-and-bound algorithms for quadratically constrained optimization problems. In: Floudas CA (ed) State of the art in global optimization: computational methods and applications. Kluwer, Dordrecht
11. Audet C, Hansen P, Jaumard B, Savard G (2000) A branch and cut algorithms for nonconvex quadratically constrained program. Math Program 87:131–152
12. Baron DP (1972) Quadratic programming with quadratic constraints. Nav Res Logist Q 19:253–260
13. Bazaraa MS, Sherali HD (1980) Benders' partitioning applied to a new formulation of the quadratic assignment problem. Nav Res Logist Q 27(1):28–42
14. Ben-Tal A, Zibulevsky M (1997) Penalty/Barrier multiplier methods for convex programming problems. SIAM J Optim 7(2):347–366
15. Ben-Saad S (1989) An algorithm for a class of nonlinear convex optimization problems. PhD thesis. University of California, Los Angeles
16. Chaovalitwongse WA, Pardalos PM, Prokoyev OA (2004) Reduction of Multi-Quadratic 0-1 Programming Problems to Linear Mixed 0-1 Programming Problems. Oper Res Lett 32(6):517–522
17. Evans DH (1963) Modular design – A special case in nonlinear programming. Oper Res 11:637–647
18. Evans DH (1970) A note on modular design – A special case in nonlinear programming. Oper Res 18:562–564
19. Falk JE, Soland RM (1969) An algorithm for separable nonvox programming problems. Manag Sci 15(9):550–569
20. Filar JA, Schultz TA (1987) Bilinear programming and structured stochastic games. J Optim Theor Appl 53(1):85–104
21. Floudas CA, Visweswaran V (1995) Quadratic optimization. In: Horst R, Pardalos PM (eds) Handbook of Global Optimization. Kluwer, Dordrecht, pp 217–269
22. Fortet R (1959) L'algèbre de Boole et ses Applications en Recherche Opérationnelle. Cah Cent Etudes Rech Oper 1:5–36
23. Fortet R (1960) Applications de l'algèbre de Boole et Recherche Opérationnelle. Rev Fr Informat Rech Oper 4:17–26
24. Glover F (1975) Improved linear integer programming formulation of nonlinear integer programs. Manag Sci 22:455–460
25. Glover F, Woolsey E (1973) Futher reduction of zero-one polynomial programs to zero-one linear programming. Oper Res 21(1):156–161
26. Glover F, Woolsey E (1974) Converting the 0-1 Polynomial Programming Program to a 0-1 Linear Program. Oper Res 22(1):180-182
27. Hansen P (1979) Methods of nonlinear 0-1 programming. Ann Discret Math 5:53–70
28. Hansen P, Jaumard B (1992) Reduction of indefinite quadratic programs to bilinear programs. J Glob Optim 2:41–60
29. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
30. Iasemidis LD, Pardalos PM, Shiau D-S, Chaovalitwongse WA, Narayanan K, Kumar S, Carney PR, Sackellares JC (2003) Prediction of Human Epileptic Seizures based on Optimization and Phase Changes of Brain Electrical Activity. Optim Methods Softw 18(1):81–104
31. Kuhn HW, Tucker AW (1951) Nonlinear Programming. In: Nayman J (ed) Proceedings of the Second Berkeley Symposium on Math. Stat. and Prob. University of California Press, Berkeley, pp 481–492
32. Linderoth J (2005) A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. Math Program 103(2):251–282
33. Nesterov YE, Nemirovskii AA (1994) Interior Point Polynomial Methods in Convex Programming. In: SIAM Series in Applied Mathematics. SIAM, Philadelphia
34. Pardalos PM, Chaovalitwongse WA, Iasemidis LD, Sackellares JC, Shiau D-S, Carney PR, Prokopyev OA, Yatsenko VA (2004) Seizure Warning Algorithm Based on Spa-

tiotemporal Dynamics of Intracranial EEG. Math Program 101(2):365–385
35. Pardalos PM, Glick JH, Rosen JB (1987) Global minimization of indefinite quadratic problems. Computing 39:281–291
36. Pardalos PM, Iasemidis LD, Shiau D-S, Sackellares JC (2001) Quadratic binary programming and dynamic system approach to determine the predictability of epileptic seizures. J Comb Optim 5(1):9–26
37. Pardalos PM, Jha S (1991) Graph separation techniques for quadratic zero-one programming. Comput Math Appl 21(6/7):107–113
38. Pardalos PM, Rodgers GP (1990) Computational aspects of a branch and bound algorithm for quadratic 0-1 programming. Computing 45:131–144
39. Pardalos PM, Rodgers GP (1990) Parallel branch and bound algorithm for quadratic zero-one on a hypercube architecture. Ann Oper Res 22:271–292
40. Pardalos PM, Rosen JB (1986) Methods for global concave minimization: A bibliographic survey. SIAM Rev 28(3):367–379
41. Petersen CC (1971) A note on transforming the product of variables to linear form in linear programs. Working Paper, Purdue University
42. Phan Huy Hao E (1982) Quadratically constrained quadratic programming: Some applications and a method for solution. Z Oper Res 26:105–119
43. Phing TQ, Tao PD, Hoai An LT (1994) A method for solving D.C. programming problems, application to fuel mixture nonconvex optimization problems. J Glob Optim 6:87–105
44. Raber U (1998) A simplicial branch-and-bound method for solving nonconvex all-quadratic programs. J Glob Optim 13:417–432
45. Rhys JMW (1970) A selection problem of shared fixed costs and network flows. Manag Sci 17:200–207
46. Rutenberg DP, Shaftel TL (1971) Product design: Subassemblies for multiple markets. Manag Sci 18(4):B220–B231
47. Sherali HD (2007) RLT: A unified approach for discrete and continuous nonconvex optimization. Ann Oper Res 147:185–193
48. Sherali HD, Adams WP (1984) A decomposition algorithm for a discrete location-allocation problem. Oper Res 32(4):878–900
49. Sherali HD, Adams WP (1990) A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J Discret Math 3(3):411–430
50. Sherali HD, Adams WP (1994) A hierarchy of relaxations and convex hull Characterizations for mixed-integer zero-one programming problems. Discret Appl Math 52:83–106
51. Sherali HD, Adams WP (1999) A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Kluwer, Dordrecht
52. Sherali HD, Adams WP, Driscoll PJ (1998) Exploiting special structures in constructing a hierarchy of relaxations for 0-1 mixed integer problems. Oper Res 46(3):396–405
53. Soland RM (1971) An algorithm for separable nonvox programming problems II: Nonconvex constraints. Manag Sci 17(11):159
54. Stern RJ, Wolkowicz H (1995) Indefinite trust region subproblems and nonsymmetric eigenvalue perturbations. SIAM J Optim 5:286–313
55. Swarup K (1966) Indefinite quadratic programming with a quadratic constraint. Ekonom Obz 4:69–75
56. Van De Panne C (1966) Programming with a quadratic constraint. Manag Sci 12:709–815
57. Watters LJ (1967) Reduction of integer polynomial programming to zero-one linear programming problems. Oper Res 15:1171–117
58. Zangwill WI (1965) Media selection by decision programming. J Advert Res 5(3):30–36

Multi-Scale Global Optimization Using Terrain/Funneling Methods

ANGELO LUCIA
Dept. of Chemical Engineering,
University of Rhode Island, Kingston, USA

MSC2000: 65H20

Article Outline

Introduction
Formulation
Methods
A Global Terrain Method for Optimization at the Small Length Scale
A Funneling Method for Optimization at the Large Length Scale
A Multi-Scale Global Optimization Method
Application
Small Scale Terrain Optimization
Large Scale Funneling Optimization
Robustness
Reliability – Avoiding Traps at Local Minima
References

Introduction

Many problems in optimization involve multiple length and time scales. Perhaps the most commonly stud-

ied problems in this area are configurational molecular modeling problems such as phase transitions in wax formation, the structure of Lennard-Jones (LJ) clusters, and protein folding. This class of optimization problems is generally characterized by the presence of many, many stationary points (i. e., minima, first-order saddles, second-order saddles, etc.) that give the appearance of roughness at the small length scale and quite different geometric structure at the large length scale. A good example of the disparity in different length scales is described in Onuchic et al. [28] who illustrate the small-scale geometry of the protein free energy landscape showing many stationary points (or roughness or frustration) at the small length scale and a funnel shaped geometry for the large length scale. This is the multi-scale description we adopt in this expose. It is often acknowledged that finding all stationary points on these multi-scale objective function surfaces is all but impossible [8] for many problems of practical interest and in most cases it is irrelevant. What is of primary interest from a computational chemistry perspective is finding the relatively few important stationary points that describe important physical phenomena – without finding everything else. These important stationary points include global minima, strong local minima and important transitions states that describe rate limiting behavior.

There are many deterministic and/or stochastic methods that can be used to solve multi-scale global optimization problems. See, for example, [1,2,3,4,5,6,7,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,29,30,31,32,33,34,35,36]. These methodologies run the gambit from branch and bound methods, to homotopy-continuation and interval methods, to simulated annealing and genetic algorithms, to terrain and funneling methods, to specialized techniques.

In this chapter, a deterministic terrain/funneling approach for multi-scale global optimization is described. This approach considers two distinct length scales and assumes that the geometry of the larger length scale is funnel shaped. Terrain methods are used to explore the objective function landscape at small length scales and gather point-wise and average gradient and curvature information while funneling methods are used to make large-scale, monotonically decreasing moves at the large length scale and 'funnel' iterates to the global minimizer.

Formulation

The formulation of the problem under consideration is straightforward. It is to find the

$$\text{global min } f(z): \quad \text{subject to } z \leq c(z) \tag{1}$$

where $f = f(z)$ is a C^3 objective function defined on R^n subject to bounds on variables, $c(z)$, and where z are the optimization variables. It is assumed that f has two distinct length scales – a small length scale of considerable roughness and a large length scale where f has non-quadratic behavior. For the discussions that follow, it is convenient to denote the gradient of f by $g = g(z)$ and the Hessian matrix of f by $h = h(z)$.

Generally, formulations based on second-order Taylor series expansion are adequate to describe behavior at the small length scale, and methods based on quadratic approximations of f are well known. However, since it is assumed that the behavior of the objective function, f, is non-quadratic at the large length scale, funneling methods are used to build approximations to the large-scale geometry of f using the funnel function given by

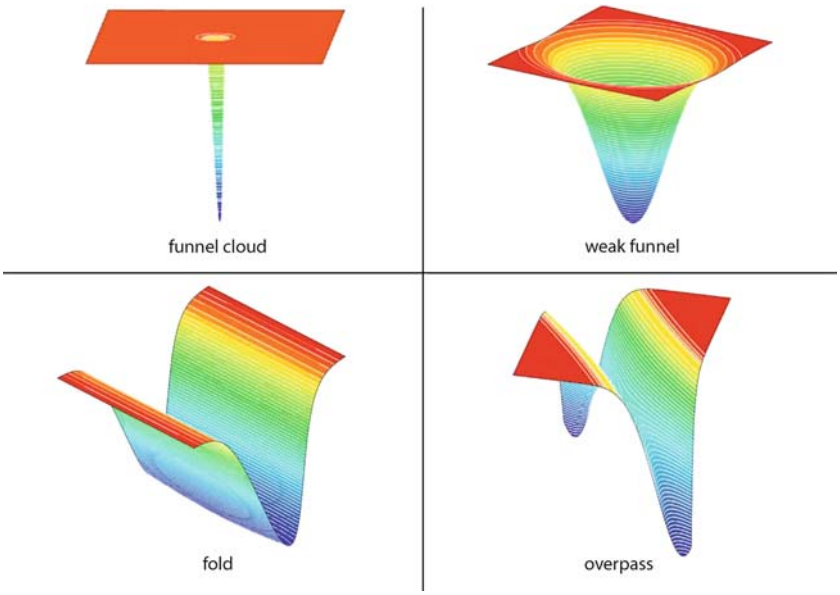
$$\mathrm{F(z)} = \mathrm{F_0} - \Gamma \exp[-q(z)] \tag{2}$$

where $q(z) = \frac{1}{2} z^{\mathrm{T}} A z + b^{\mathrm{T}} z + c$, and where $\Gamma > 0$, $\mathrm{F_0}$ and c are scalar parameters, b is an n-dimensional vector, and A is an n x n symmetric matrix. The functional form of Eq. (2) is interesting because it is non-convex, has a unique global minimum when A is positive definite, and contains certain inherent self-scaling characteristics. Figure 1 gives an illustration of the funnel geometries that can be represented by the functionality of Eq. (2).

The funnels at the top and the bottom left in Fig. 1 each have unique minimum because A is positive definite while the one at the bottom right has two minima since A is indefinite.

Methods

In this sub-section, the global terrain method, a funneling algorithm, and a multi-scale global optimization method are described.



Multi-Scale Global Optimization Using Terrain/Funneling Methods, Figure 1
Various Funnel Geometries

A Global Terrain Method for Optimization at the Small Length Scale

Terrain methods [18,19,20] have described in detail in a separate chapter in this encyclopedia and are briefly summarized in this chapter for the purpose of continuity of presentation. Terrain methods are used to locate sets of stationary and singular points of the objective function. They do this by following valleys and moving up and down the landscapes of $g^{\mathrm{T}}g$ and f. Key among the equations used in terrain methods is the characterization of valleys as solutions, V, to a sequence of general nonlinearly constrained optimization problems

$$\mathrm{V} = \{\min h^{\mathrm{T}} g^{\mathrm{T}} hg \text{ such that } g^{\mathrm{T}} g = L, \text{ for all } L\varepsilon\ \Lambda\} \tag{3}$$

where g and h are defined as before, where L is any given value (or level) of the least-squares objective function, and where Λ is some collection of contours. Terrain methods require

1) Reliable downhill equation solving
2) Reliable and efficient computation of singular points
3) Efficient uphill movement comprised of predictor-corrector calculations
4) Reliable and efficient eigenvalue-eigenvector computations
5) Effective bookkeeping
6) A termination criterion to decide when the computations have finished.

In this chapter, a terrain methodology is used to find sets of stationary and singular points and to determine *average* gradient and *average* curvature (or Hessian matrix) information along a given terrain path. Average gradient and curvature information is calculated from the mean value theorem using the following equations.

$$\langle g \rangle = (1/\alpha) \quad \int\ g[z(\alpha)]d\alpha \tag{4}$$

$$\langle h \rangle = (1/\alpha) \quad \int\ h[z(\alpha)]d\alpha \tag{5}$$

where α is some relevant length of the smooth terrain path connecting any set of stationary and singular points. It is important for the reader to understand that it is the set of stationary and singular points as well as average gradient and Hessian matrix information that are communicated from the small length scale to the large length scale. This will be discussed again a little later in this chapter.

A Funneling Method for Optimization at the Large Length Scale

To build iterative global funnel approximations of the objective function we match function, gradient, and second derivative information of the true objective function, f, g and h, with the function, gradient, and Hessian matrix information, F, G and H respectively, of the funnel function at various points, where $G = G(z)$ and $H = H(z)$ are given by

$$G(z) = \Gamma \exp[-q(z)][Az + b] \tag{6}$$

$$H(z) = \Gamma \exp[-q(z)][A - (Az + b)(Az + b)^{\mathrm{T}}] \tag{7}$$

Note that if $f(z)$ is used in place of $F(z)$ in Eq. (2), then it follows that

$$\gamma(z) = F_0 - f(z) = \Gamma \exp[-q(z)] \tag{8}$$

where $\gamma > 0$ is a positive scaling factor that depends on a single numerical measurement, $f(z)$, and the scalar parameter, F_0. Moreover, replacing $G(z)$ with $g(z)$ and $H(z)$ with $h(z)$ in Eqs. (6) and (7) respectively give the equations

$$[Az + b] = g(z)/\gamma \tag{9}$$

$$A = [\gamma h + gg^{\mathrm{T}}]/\gamma^2 \tag{10}$$

Equations (8), (9), and (10) show that it is possible to estimate A and b from values of $f(z)$, $g(z)$, and $h(z)$ using interpolation formula at two or more iterates.

Interpolating Formulae Let z_k be any value of the unknown optimization variables with corresponding objective function, gradient and Hessian matrix values f_k, g_k and h_k respectively. Also let z_{k+1} be some other arbitrary but not necessarily nearby or successive iterate with corresponding function, gradient and Hessian matrix values f_{k+1}, g_{k+1} and h_{k+1}. Writing Eq. (8) for z_k and z_{k+1} and then subtracting the latter from the former, eliminates F_0 and gives

$$\gamma_{k+1} - \gamma_k = f_k - f_{k+1} \tag{11}$$

Repeating the same algebra using Eq. (10) yields

$$\gamma_k^2[\gamma_{k+1}h_{k+1} + g_{k+1}g_{k+1}^{\mathrm{T}}] - \gamma_{k+1}^2[\gamma_k h_k + g_k g_k^{\mathrm{T}}] = 0 \tag{12}$$

Equations (11) and (12) form a set of $[1 + n(n + 1)/2]$ nonlinear equations in the two unknowns γ_k and γ_{k+1} when the symmetry of the associated matrices is taken into account. This together with Eq. (11) gives a total of $[1 + n(n + 1)/2]$ nonlinear equations. For $n = 1$, there are two equations and two unknowns. When $n > 1$ there are more equations than unknown variables. However, irrespective of this, two equations for which γ_k and $\gamma_{k+1} > 0$ can be determined using the Routh criterion.

Estimating Funnel Parameters Calculated values of γ_k and γ_{k+1} can be used to determine the matrix A from Eq. (10) – using gradient and Hessian matrix information either at z_k or z_{k+1}. Following this, the parameter b can be computed by simply rearranging Eq. (9) to give

$$b = g(z)/\gamma - Az \tag{13}$$

while F_0 can be calculated from $F_0 = f(z) + \gamma$. Like A, the values of $\boldsymbol{b}$ and F_0 can be determined using function and gradient values at either z_k or z_{k+1}.

Finding the Funnel Minimum It is then straightforward to estimate the unique global minimizer of the *funnel approximation*, say y, by simply solving

$$Ay = -b \tag{14}$$

Note that Eq. (10) shows that the matrix A is generated from a rank-one, positive semi-definite correction to $h(z)$, that the sign definiteness of A can be controlled by the parameter γ, and that γ appears to also play the role of a self-scaling factor.

Communication Between Length Scales One of the keys to success in any multi-scale global optimization methodology is the communication between length scales. In the terrain/funneling approach to multi-scale global optimization, small-scale calculations communicate *average* gradient and Hessian matrix information at two distinct points to the large length scale. The large length scale optimizations, on the other hand, communicate an estimate of the values of the optimization variables at a converged minimum of the funnel approximation to the small length scale and identify the next region on the objective function surface on which small-scale optimizations should be conducted.

A Multi-Scale Global Optimization Method

The details of a multi-scale global optimization methodology based on the terrain and funneling methods is as follows.

1 Perform two sets of small-scale optimization calculations using the terrain methodology starting from two different points on the objective function surface. Calculate *average* gradient, and *average* Hessian information along the resulting terrain paths. Thus at the kth funnel iteration the following information is available – z_k, f_k, g_k and h_k and z_{k+1}, f_{k+1}, g_{k+1} and h_{k+1} such that $f_{k+1} < f_k$.

2 Conduct iterative large-scale optimization calculations with the funneling methodology initialized using the objective function, average gradient and average Hessian information from the small-scale optimization calculations to find a funnel minimum that also corresponds to a stationary point on the true objective function surface. To do this,
   a) Solve Eqs. (11) and (12) for γ_k and γ_{k+1}.
   b) Using γ_{k+1}, calculate A and b from Eqs. (10) and (13) respectively.
   c) Determine an estimate of funnel minimum, y, from Eq. (14).
   d) Evaluate $f(y)$, $g(y)$ and $h(y)$.
   e) Test $f(y)$ against f_{k+1}. If $f(y) < f_{k+1}$, then go to step 2f for the next funnel iteration. Else set $\gamma_k = \gamma_k/2$ and return to step 2a.
   f) Set $z_{k+1} = y$, $f_{k+1} = f(y)$, $g_{k+1} = g(y)$, and $h_{k+1} = h(y)$.
   g) If $||g(z_{k+1})|| < \varepsilon$, set $y = y^*$, and go to step 3; else go to step 2a.

3 Conduct a new set of small-scale terrain calculations using the funnel minimum from step 2. Calculate *average* gradient and *average* Hessian information along the resulting terrain path such that new values of z_{k+1}, f_{k+1}, g_{k+1} and h_{k+1} satisfy the condition $f_{k+1} < f_k$.

4 Repeat step 2 using the new small-scale information and z_k, f_k, g_k and h_k from step 1.

5 Repeat steps 2 and 3 until there is no further decrease in the objective function.

Here we describe step 2 of the multi-scale algorithm. The most effective way to determine γ_{k+1} in step 2a is to rearrange Eq. (11) for γ_{k+1} in terms of γ_k and then substitute the resulting expression into Eq. (12) This gives a cubic polynomial equation in γ_k and shows that there are *three* possible values of γ_k and thus three possible sets of scaling factors (γ_k, γ_{k+1}). Using an equation solver like Newton's method, it is easy to find one solution for γ_k. The other two values of γ_k can be determined by deflation of the cubic equation to a quadratic equation and by using the quadratic formula. The correct value of γ_k is the smallest *real* valued $\gamma_k > 0$ such that $\gamma_{k+1} > \gamma_k$, where $\gamma_{k+1} = f_k - f_{k+1} + \gamma_k$. Step 2b is straightforward and step 2c requires the solution of a system of linear equations. Step 2d evaluates the *actual* function, gradient, and Hessian matrix at the funnel iterate y. Step 2e is used to ensure monotonic decreasing objective function values by halving γ_{k+1} until $f(y) < f_{k+1}$ while step 2f replaces the information associated with z_{k+1} with that for the funnel iterate y. Finally, step 2g checks the norm of the gradient of the objective function and terminates the funnel iterations once that norm of the gradient falls below the specified tolerance. Note that any point, y^*, that satisfies the convergence condition in step 2g is simultaneously a stationary point of $f(z)$ and a minimizer of the funnel function $F(z)$.

The proposed multi-scale optimization algorithm is very robust. The reason for this is because if the funneling algorithm gets trapped at a local minimum, the methodology returns to the small-scale terrain calculations to get average gradient and average Hessian information around that local minimum. Replacing point-wise zero-valued gradient information at a local minimum with averaged non-zero valued gradient information forces the optimizer to look for a minimizer that is deeper in funnel. By forcing movement further down the funnel in this way, the multi-scale algorithm will continue to improve the value of the objective function in a monotonic fashion until it reaches the global minimum at the bottom of the funnel.

Application

In this sub-section, a simple illustration of the multi-scale global optimization methodology is given using the classical thirteen-particle Lennard-Jones (LJ_{13}) problem. This example was selected because it is the first in the series of Lennard-Jones clusters that truly has a single funnel based on the Mackay icosahedron structure with the global minimum lying at the bot-

tom of the funnel [8]. The LJ_{13} cluster has 33 unknown Cartesian coordinates, 1509 minima, 116,835 first-order and second-order transition states, and many higher order transition states [9].

Small Scale Terrain Optimization

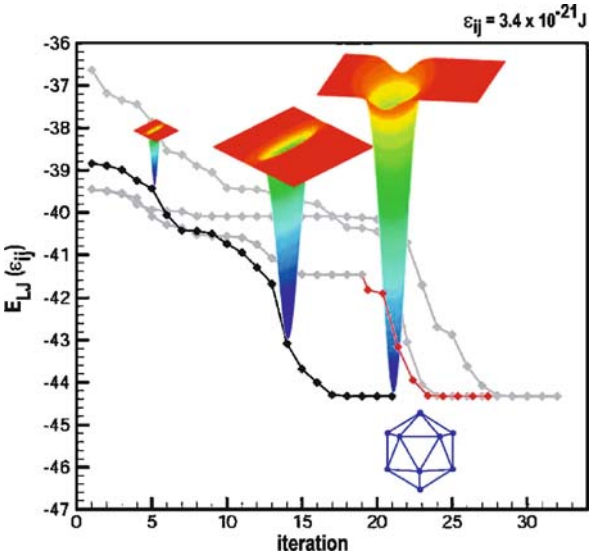
Small-scale optimization calculations were performed using the terrain methodology from two different starting points. From the first starting point, which corresponds to an energy of E = −38.8572, six (6) stationary and singular points along a terrain path were calculated requiring 4832 function and gradient evaluations and 616.68 s of computer time. Average gradient and Hessian information was accumulated along the terrain path defined the path integrals given by Eqs. (4) and (5). From a second and quite different starting point on the objective function surface another set of small-scale optimization calculations was performed using the terrain methodology. The energy at this second starting point was E = −38.4246, and five (5) stationary and singular points were calculated requiring 1415 function and gradient evaluations and 214.25 s of computer time. Again, average gradient and Hessian information was accumulated along the terrain path using the path integrals using Eqs. (4) and (5).

Large Scale Funneling Optimization

To build an approximation of the large-scale geometry, two singular points – one from each of the two sets of terrain calculations – were selected. Using the function values at these singular points as well as average gradient and average Hessian information, funnel parameters were determined and a first estimate of the funnel minimum was calculated. We then replaced one of the singular points with this estimate of the funnel minimum and repeated the funnel optimization calculations until $||g|| < \varepsilon = 1 \times 10^{-4}$. Figure 2 summarizes these results, which are shown by the curve of dark or black diamonds. The resulting structure for LJ_{13} is also shown in Fig. 2. Twenty (20) funneling iterations were required for convergence to the global minimum at E = −44.3268 on the LJ_{13} energy surface.

Robustness

Terrain/funnel calculations were performed using many other arbitrary starting points on the potential



Multi-Scale Global Optimization Using Terrain/Funneling Methods, Figure 2
Multi-Scale Terrain/Funneling Calculations for the LJ_{13} Cluster

energy surface for the LJ_{13} cluster to illustrate the robustness of this multi-scale optimization method. Results for some of these calculations are also shown in Fig. 2 by the two light gray curves with diamonds. Note that these multi-scale optimization calculations using the terrain/funneling approach also converge easily and monotonically to the global minimum in 32 and 27 funnel iterations respectively from these starting points. In all cases, the funneling portion of the overall multi-scale algorithm finds the global minimum in less than 0.35 s.

Reliability – Avoiding Traps at Local Minima

To show that the proposed optimization approach does not get trapped at local minima, terrain/funneling calculations were repeated from a different set of initial terrain calculations. In particular, small-scale terrain calculations starting from points on the energy surface corresponding to energy values of E_1 = −39.1597 and E_2 = −38.4246 were performed and average gradient and average Hessian information gathered along the resulting terrain paths. Using this information, the funneling algorithm converged to a *local* minimum on the potential energy surface at E_3 = −41.4445 in 18 funnel iterations. The results of these calculations correspond to the gray curve that terminates at E = −41.4445 in

Fig. 2. This local minimum was then used as a starting point to perform a third set of small-scale terrain calculations and to gather average gradient and average Hessian information in the valley around the local minimum. Using this average information around the local minimum, E_3, together with average information around E_1, a second set of iterative funneling calculations was performed. In this case, the funneling calculations located the global minimum at $E = -44.3268$ on the energy surface in 16 funnel iterations. This second set of funnel iterations is shown by the red curve in Fig. 2.

Those readers interested in the numerical details of the LJ_{13} illustration are encouraged to contact the author, who is quite willing to provide all computer-generated numerical results for this example.

References

1. Aluffi-Pentini F, Parisi V, Zirilli F (1985) Global optimization and stochastic differential equations. J Opt Theory Appl 47:1–16
2. Bahren J, Protopopescu V (1996) Generalized TRUST algorithms for global optimization. In: Floudas CA, Pardalos PM (eds) State of the Art in Global Optimization. Kluwer, Dordrecht, pp 163–180
3. Baker J (1986) An algorithm for the location of transition states. J Comput Chem 7:385–395
4. Bilbro GL (1994) Fast stochastic global optimization. IEEE Trans Sys Man Cyber 4:684–689
5. Cerjan CJ, Miller WH (1981) On finding transition states. J Chem Phys 75:2800–2806
6. Deaven DM, Tit N, Morris JR, Ho KM (1996) Structural optimization of Lennard-Jones clusters by a genetic algorithm. Chem Phys Lett 256:195–200
7. DeJong K (1975) An analysis of the behavior of a class of genetic adaptive systems. Ph.D. Thesis, Univ. of Michigan, Ann Arbor, Princeton
8. Doye JPK, Miller MA, Wales DJ (1999) Evolution of the potential energy surface with size for Lennard-Jones clusters. J Chem Phys 111:8417–8428
9. Doye JPK, Wales DJ (2002) Saddle points and dynamics of Lennard-Jones clusters, solids and supercooled liquids. J Chem Phys 116:3777–3788
10. Gibson KD, Scheraga HA (1987) Revised algorithm for the build-up procedure for predicting protein conformation by energy minimization. J Comput Chem 8:826–834
11. Gregurick SK, Alexander MH, Hartke B (1996) Global geometry optimization of $(Ar)_n$ and $B(Ar)_n$ clusters using a modified genetic algorithm. J Chem Phys 104:2684–2691
12. Hansen ER (1980) Global optimization using interval analysis – The multidimensional case. Numer Math 34:247–270
13. Henkelman G, Johannesson G, Jonsson H (2000) Methods for finding saddle points and minimum energy paths. In: Schwartz SD (ed) Progress in Theoretical Chemistry and Physics. Kluwer, Dordrecht, 5:269–300
14. Holland JH (1992) Genetic algorithms. Sci Am 267:66
15. Jones DT, Taylor WR, Thornton JM (1992) A new approach to protein folding recognition. Nature 358:86–89
16. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
17. Levy AV, Montalvo A (1985) The tunneling algorithm for the global minimization of functions. SIAM J Sci Stat Comp 6:15–29
18. Lucia A, Yang F (2002) Global terrain methods. Comput Chem Eng 26:529–546
19. Lucia A, Yang F (2003) Multivariable terrain methods. AIChE J 49:2553–2563
20. Lucia A, DiMaggio PA, Depa P (2004) A geometric terrain methodology for global optimization. J Global Optim 29:297–314
21. Lucia A, DiMaggio PA, Depa P (2004) Funneling algorithms for multi-scale optimization on rugged terrains. Ind Eng Chem Res 43:3770–3781
22. Maranas CD, Floudas CA (1992) A global optimization approach to Lennard-Jones microclusters. J Chem Phys 97:7667–7678
23. Maranas CD, Floudas CA (1995) Finding all solutions to nonlinearly constrained systems of equations. J Global Optim 7:143–182
24. Matro A, Freeman DL, Doll JD (1994) Locating transition states using double-ended classical trajectories. J Chem Phys 101:10458–10463
25. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:1087–1092
26. Muller K, Brown LD (1979) Location of saddle points and minimum energy paths by a constrained simplex optimization procedure. Theoret Chim Acta 53:75–93
27. Niesse JA, Mayne HR (1996) Global geometry optimization of atomic clusters using a modified genetic algorithm in space-fixed coordinates. J Chem Phys 105:4700–4706
28. Onuchic JN, Luthey-Schulten Z, Wolynes PG (1997) Theory of protein folding: the energy landscape perspective. Annu Rev Phys Chem 48:545–600
29. Piela L, Kostrowicki J, Scheraga HA (1989) The multiple minima problem in conformational analysis of molecules. Deformation of the potential energy hypersurface by the diffusion equation method. J Phys Chem 93:3339–3346
30. Pincus MR, Klausner RD, Scheraga HA (1982) Calculation of the three dimensional structure of the membrane-bound portion of melittin from its amino acid sequence. Proc Natl Acad Sci USA 79:5107–5110
31. Scheraga HA (1974) Prediction of protein conformation. In: Anfinsen CB, Schechter AN (eds) Current Topics in Biochemistry. Acad Press, New York, p 1

32. Schnepper CA, Stadtherr MA (1996) Robust process simulation using interval methods. Comput Chem Eng 20:187–199
33. Sevick EM, Bell AT, Theodorou DN (1993) A chain of states method for investigating infrequent events in processes occurring in multistate, multidimensional systems. J Chem Phys 98:3196–3212
34. Sun AC, Seider WD (1992) Homotopy-continuation algorithm for global optimization. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton Univ Press, Princeton, pp 561–592
35. Wales DJ, Doye JPK (1997) Global optimization by basin hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. J Phys Chem A 101:5111–5116
36. Westerberg KM, Floudas CA (1999) Locating all transition states and studying the reaction pathways of potential energy surfaces. J Chem Phys 110:9259–9295

Multistage Stochastic Programming: Barycentric Approximation

KARL FRAUENDORFER, MICHAEL SCHÜRLE
Institute Operations Res., University St. Gallen, St. Gallen, Switzerland

MSC2000: 90C15

Article Outline

Keywords
See also
References

Keywords

Stochastic programming; Approximation

Many problems in finance, economics and other applications require that decisions $x_t \in \mathbf{R}_t^n$ are made periodically over time, depending on observations of uncertain data (η_t, ξ_t) in future periods $t = 1, \ldots, T$. Here, it is distinguished between random data $\eta_t \in \Theta_t \subset \mathbf{R}_t^K$ that influence prices in the objective function and random data $\xi_t \in \Xi_t \subset \mathbf{R}_t^L$ that affect the demand on the right-hand side of constraints in an optimization problem.

Once an observation (η_t, ξ_t) becomes available, the decision maker has to determine a policy x_t that minimizes the costs $\rho_t(x^{t-1}, x_t, \eta^t)$ in t plus the expected costs in the subsequent periods $t+1, \ldots, T$, subject to a set of constraints $f_t(x^{t-1}, x_t) \leq h(\xi^t)$. Both the objective function and the constraints may depend on the sequences of observations $\eta^t = (\eta_1, \ldots, \eta_t)$, $\xi^t = (\xi_1, \ldots, \xi_t)$ up to t and earlier decisions $x^{t-1} = (x_0, \ldots, x_{t-1})$. Obviously, an action x_t must be selected after (η_t, ξ_t) is observed but before the future outcomes $\eta_{t+1}, \ldots, \eta_T$ and $\xi_{t+1}, \ldots, \xi_T$ are known, i. e. the decision is based only on information available at time t. Hence, one obtains a sequence of decisions with the property $x_0, x_1(\eta^1, \xi^1), \ldots, x_T(\eta^T, \xi^T)$, called *nonanticipativity*. This results in a *multistage stochastic program*, which may be written in its dynamic representation as a series of nested two-stage programs (with $\phi_{T+1}(\cdot) := 0$, see [4]):

$$\phi_t(x^{t-1}, \eta^t, \xi^t) := \min \left\{ \rho_t(x^{t-1}, x_t, \eta^t) + \int \phi_{t+1}(x^{t-1}, x_t, \eta^t, \eta_{t+1}, \xi^t, \xi_{t+1}) \, \mathrm{d}P_{t+1} \right\}, \quad t = 0, \ldots, T, \tag{1}$$

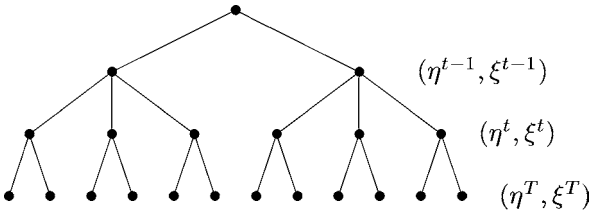
where the expectation is taken w.r.t. the probability measure $P_{t+1}(\eta_{t+1}, \xi_{t+1} | \eta^t, \xi^t)$ of the joint distribution of (η_{t+1}, ξ_{t+1}), subject to

$$f_t(x^{t-1}, x_t) \leq h(\xi^t), \quad x_t \geq 0. \tag{2}$$

In case of discrete distributions, it is well known that one can immediately transform the stochastic multistage program given by (1) and (2) into a (large) deterministic equivalent problem which can be solved by standard optimization tools, possibly combined with *decomposition techniques* to exploit the special structure of the problem (see e. g. [1,9,10]). However, if the distribution is continuous with some density function, it is in general impossible to do the integration in (1) exactly. One way to overcome this difficulty is to approximate the (continuous) probability measure P_t by a discrete one Q_t. In MSP, this is usually done by constructing a *scenario tree* which can be illustrated as follows:

Together with the associated scenario probabilities, this tree is defined formally as

$$\mathcal{A} := \left\{ (\eta^T, \xi^T) : \begin{array}{c} (\eta_t, \xi_t) \in \mathcal{A}_t(\eta^{t-1}, \xi^{t-1}) \\ \forall t > 0 \end{array} \right\}, \qquad q(\eta^T, \xi^T) := \prod_{t=1}^{T} q_t(\eta_t, \xi_t | \eta^{t-1}, \xi^{t-1}). \tag{3}$$



Multistage Stochastic Programming: Barycentric Approximation, Figure 1

The scenario tree represents an approximation of the discrete-time process $(\eta_t, \xi_t; t = 1, \ldots, T)$, and $\mathcal{A}_t(\cdot)$ denotes the set of finitely many outcomes for (η_t, ξ_t) conditioned on the history (η^{t-1}, ξ^{t-1}). Again, this results in a sparse large scale program. Naturally, the question arises how good the accuracy of the associated (deterministic) optimization problem is, and if the set of scenarios can be improved w.r.t. the accuracy.

For convex optimization problems where the random data are decomposable in two groups, one that determines the cost function and the second one affecting the demand, it can be shown (see [4] for details) that the *value function* (1) is a saddle function for all $t = 1, \ldots, T$ under the following conditions:

i) $\rho_t(\cdot)$ is concave in η_t,
ii) the left-hand sides of the constraints are deterministic, and
iii) the distribution function of $P_t(\cdot | \eta^{t-1}, \xi^{t-1})$ depends linearly on the past.

Then, (1) is concave in η_t and convex in (x_t, ξ_t). The situation where assumptions i)–iii) are fulfilled is called the entire convex case.

This underlying saddle property of the value function motivates the application of *barycentric approximation* which derives two scenario trees $\mathcal{A}^u$ and $\mathcal{A}^l$. The associated approximate deterministic programs provide upper and lower bounds to the original problem. In this sense, barycentric approximation is a generalization of the inequalities due to H.P. Edmundson [2] and A. Madansky [8] (see e. g. ▶ Stochastic Programs with Recourse: Upper Bounds) and J.L. Jensen [6] that is applicable to saddle functions of correlated random data. Here, it is assumed that $\Theta_t \subset R_t^K$ and $\Xi_t \subset R_t^L$ are regular simplices whose vertices are denoted by u_{ν_t}, $\nu_t = 0, \ldots, K_t$, and $v_{\mu t}$, $\mu_t = 0, \ldots, L_t$. Both Θ_t and Ξ_t may depend on prior observations (η^{t-1}, ξ^{t-1}) although this is not stressed in the notation for simplicity.

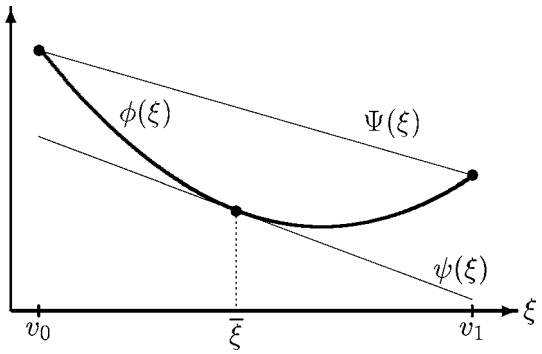
To illustrate the way the discretization is performed, assume that a two-stage problem is given (the time index is omitted here) with deterministic objective, i. e. only the right-hand side coefficients $h(\xi)$ are random (see e. g. [7]). For any $\xi \in \Xi$, the *barycentric weights* $\tau_0(\xi), \ldots, \tau_L(\xi)$ w.r.t. the simplex Ξ are given by

$$\begin{aligned} \tau_0 + \cdots + \tau_L &= 1, \\ \tau_0 v_0 + \cdots + \tau_L v_L &= \xi. \end{aligned} \tag{4}$$

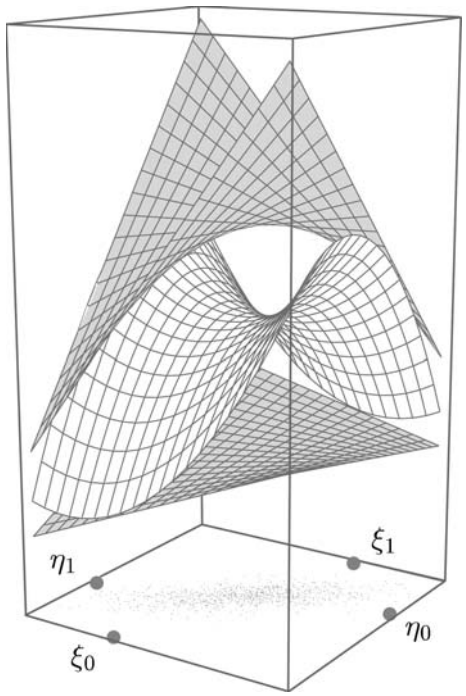
Since $\phi(x, \xi)$ is convex in ξ for all x, $\varphi(\xi) := \phi(\widehat{x}, \xi)$ is a convex function for any fixed first-stage decision $\widehat{x}$. Due to convexity, $\varphi(\xi)$ is bounded from above for all $\xi \in \Xi$ by a linear function $\Psi(\xi) = \sum_{\mu=0}^{L} \tau_\mu(\xi)\, \nu_\mu$. To construct the 'classical' *Edmundson–Madansky upper bound* (EM) for $\int \varphi(\xi)\, dP$ over the simplex Ξ, ξ is replaced by a discrete random variable with the same expectation, attaining values $v_0, \ldots, v_L$. To obtain the corresponding probabilities, ξ has to be replaced by $overline\xi = \int \xi\, dP$ in (4), and the system must be solved for $\tau_0, \ldots, \tau_L$. Then, $\int \varphi(\xi)\, dP \leq \sum_{\mu=0}^{L} \tau_\mu(\overline{\xi}) \nu_\mu$, and the weights may be interpreted as the probabilities of the discrete outcomes.

On the other hand, a lower bound can be found using *Jensen's inequality*: $\varphi(\overline{\xi}) \leq \int \varphi(\xi)\, dP$, i. e. by evaluation of the function for the expectation of ξ, and the tangent $\psi(\xi)$ to $\varphi(\xi)$ at $\overline{\xi}$ is a lower bound to the original function. Both linear approximations $\psi(\xi)$ and $\Psi(\xi)$ to the convex value function for a given policy are shown in Fig. 2.

From a computational viewpoint, the original function $\varphi(\xi)$ is replaced by two linear affine functions. Clearly, $\psi(\xi)$ and $\Psi(\xi)$ can be integrated easily over the support of ξ. If there is only randomness in the objec-



Multistage Stochastic Programming: Barycentric Approximation, Figure 2



Multistage Stochastic Programming: Barycentric Approximation, Figure 3

tive with deterministic right-hand sides, a lower and an upper bound can be constructed by applying the same procedure to the dual concave (maximization) problem, deriving an upper bound from Jensen's inequality and a lower approximation with the EM-rule.

Barycentric approximation combines these concepts for stochastic objective and right-hand sides [3] and extends them to the multistage case [4,5]. It derives distinguished points, so-called *generalized barycenters*, where the value function (1) must be supported by two bilinear functions to minimize the error induced by the approximation. This is shown in Fig. 3 for $K_t = L_t = 1$, where the minorant is supported at ξ_0 and ξ_1 and the majorant at η_0 and η_1.

Let $\lambda_{t,0}(\eta_t), \ldots, \lambda_t, K_t(\eta_t), \tau_t, 0(\xi_t), \ldots, \tau_t, L_t(\xi_t)$ be the barycentric weights w.r.t. Θ_t and Ξ_t defined analogously to (4). For both simplices, the generalized barycenters and their probabilities are given by

$$\eta_{\mu_t} = \left[q(\eta_{\mu_t})\right]^{-1} \cdot \sum_{\nu_t=0}^{K_t} u_{\nu_t} \int \lambda_{\nu_t}(\eta_t)\tau_{\mu_t}(\xi_t)\,\mathrm{d}P_t,$$

$$q(\eta_{\mu_t}) = \int \tau_{\mu_t}(\xi_t)\,\mathrm{d}P_t, \quad \mu_t = 0, \ldots, L_t,$$

$$\xi_{\nu_t} = \left[q(\xi_{\nu_t})\right]^{-1} \cdot \sum_{\mu_t=0}^{L_t} \nu_{\mu_t} \int \lambda_{\nu_t}(\eta_t)\tau_{\mu_t}(\xi_t)\,\mathrm{d}P_t,$$

$$q(\xi_{\nu_t}) = \int \lambda_{\nu_t}(\eta_t)\,\mathrm{d}P_t, \quad \nu_t = 0, \ldots, K_t.$$

Note that the integrand $\lambda_{\mu t}(\eta_t)\cdot\tau_{\nu_t}(\xi_t)$ is a bilinear function in (η_t, ξ_t) since the barycentric weights $\lambda_{\mu t}$ and τ_{ν_t} are linear in their components. Obviously, a bilinear function is easy to integrate which was the intention of the approximation.

The generalized barycenters ξ_{ν_t}, $\nu_t = 0, \ldots, K_t$, are supporting points of the minorant. They are combined with the vertices u_{ν_t} and weighted with the corresponding probabilities $q(\xi_{\nu_t})$ to obtain discrete outcomes for the lower approximation of the original measure P_t. This way, one derives a discrete probability measure Q_t^l with support

$$\operatorname{supp} Q_t^l = \{(u_{\nu_t}, \xi_{\nu_t}):\ \nu_t = 0, \ldots, K_t\}.$$

Analogously, $\eta_{\mu t}$, $\mu_t = 0, \ldots, L_t$, are supporting points for the majorant with assigned probabilities $q(\eta_{\mu t})$. This induces a discrete measure Q_t^u for the upper approximation with

$$\operatorname{supp} Q_t^u = \{(\eta_{\mu_t}, \nu_{\mu_t}):\ \mu_t = 0, \ldots, L_t\}.$$

Both measures represent the solutions of two corresponding moment problems. The advantageous feature from a computational viewpoint is that the generalized barycenters and their probabilities are completely determined by the first moments of η_t and ξ_t, and by the bilinear cross moments $E(\eta_{\nu_t}\cdot\xi_{\mu t})$, $\nu_t = 0, \ldots, K_t$, $\mu_t = 0, \ldots, L_t$. Note that the covariance of two random variables is derived from the first moments and the corresponding cross moments. Therefore, the measures Q_t^u and Q_t^l incorporate implicitly a correlation between η_t and ξ_t. However, cross moments (or covariances, respectively) between different elements of η_t are not taken into account (the same holds for the components of ξ_t). Hence, the formulae given above are applicable without the assumption of independent random variables.

Applying the approximation scheme dynamically over time, one obtains two *barycentric scenario trees* $\mathcal{A}^l$ and $\mathcal{A}^u$ with their path probabilities of type (3). The set of outcomes at stage $t = 1, \ldots, T$ is given by $\mathcal{A}^l(\eta^{t-1}, \xi^{t-1}) = \operatorname{supp} Q_t^l(\cdot|\eta^{t-1}, \xi^{t-1})$ and $\mathcal{A}^u(\eta^{t-1}, \xi^{t-1}) = \operatorname{supp} Q_t^u(\cdot|\eta^{t-1}, \xi^{t-1})$. Substituting P_t in (1) by the discrete measures Q_t^l and Q_t^u yields two value functions

$$\psi_t(x^{t-1}, \eta^t, \xi^t) := \min \left\{ \rho_t(x^{t-1}, x_t, \eta^t) + \int \psi_{t+1}(x^{t-1}, x_t, \eta^t, \eta_{t+1}, \xi^t, \xi_{t+1})\, \mathrm{d}Q_{t+1}^l \right\},$$

$$\Psi_t(x^{t-1}, \eta^t, \xi^t) := \min \left\{ \rho_t(x^{t-1}, x_t, \eta^t) + \int \Psi_{t+1}(x^{t-1}, x_t, \eta^t, \eta_{t+1}, \xi^t, \xi_{t+1})\, \mathrm{d}Q_{t+1}^u \right\}$$

for $t = 0, \ldots, T$ with $\psi^{T+1}(\cdot) = \Psi^{T+1}(\cdot) := 0$. According to [4], these are lower and upper bounds to the original value function, i. e.

$$\psi_t(x^{t-1}, \eta^t, \xi^t) \le \phi_t(x^{t-1}, \eta^t, \xi^t) \le \Psi_t(x^{t-1}, \eta^t, \xi^t).$$

In the entire convex case, the accuracy of the approximation is quantifiable by the difference between the upper and lower bound. If required, the approximation can be improved by partitioning the simplices Θ_t and Ξ_t. In case that the subsimplices become arbitrarily small, the extremal measures converge to P_t, and the convergence of the upper and lower bounds to the expectation of the value function is guaranteed (see [5] for details).

See also

- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Two-stage Stochastic Programs with Recourse
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ General Moment Optimization Problems
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Minimax Approach
- ▶ Simple Recourse Problem: Primal Method
- ▶ Simple Recourse Problem: Dual Method
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Integer Programs
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Preprocessing in Stochastic Programming
- ▶ Stochastic Network Problems: Massively Parallel Solution

References

1. Birge JR, Donohue CJ, Holmes DF, Svintsitski OG (1996) A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. Math Program 75:327–352
2. Edmundson HP (1957) Bounds on the expectation of a convex function of a random variable. Techn Report RAND Corp, 982
3. Frauendorfer K (1992) Stochastic two-stage programming. Springer, Berlin
4. Frauendorfer K (1994) Multistage stochastic programming: Error analysis for the convex case. Math Meth Oper Res 39:93–122

5. Frauendorfer K (1996) Barycentric scenario trees in convex multistage stochastic programming. Math Program 75:277–293
6. Jensen JL (1906) Sur les fonctions convexes et les inégalitées entre les valeurs moyennes. Acta Math 30:175–193
7. Kall P (1998) Bounds for and approximations to stochastic linear programs with recourse. In: Marti K, Kall P (eds) Stochastic Programming Methods and Technical Applications. Springer, Berlin, pp 1–21
8. Madansky A (1959) Bounds on the expectation of a convex function of a multivariate random variable. Ann Math Statist 30:743–746
9. Mulvey JM, Ruszczyński A (1995) A new scenario decomposition method for large-scale stochastic optimization. Oper Res 41:477–490
10. Rosa CH, Ruszczyński A (1996) On augmented Lagrangian decomposition methods for multistage stochastic programs. Ann Oper Res 64:289–309

N

Nested Partitions Optimization

LEYUAN SHI[1], SIGURDUR OLAFSSON[2]
[1] University of Wisconsin, Madison, USA
[2] Iowa State University, Ames, USA

MSC2000: 90C59, 90C11

Article Outline

Introduction
Formulation
Methods
Cases
Resource-Constrained Project Scheduling
Feature Selection
Radiation Treatment Planning
Conclusions
References

Introduction

The *nested partitions* (NP) method is a powerful optimization method that has been found to be very effective for solving large-scale discrete optimization problems. Such problems are common in many practical applications and the NP method is hence useful in diverse application areas. It can be applied to both operational and planning problems and has been demonstrated to effectively solve complex problems in both manufacturing and service industries.

The NP method was first introduced in [4] and application examples include diverse areas such as optimization of beam orientation in radiation therapy [1], feature selection in data mining [3], and product design [5].

Formulation

The NP method is particularly well suited for complex large-scale discrete optimization problems where traditional methods experience difficulty. It is, however, very broadly applicable and can be used to solve any optimization problem that can be stated mathematically in the following generic form:

$$\min_{x \in X} f(x)\,, \tag{1}$$

where the solution space or feasible region X is either a discrete or a bounded set of feasible solutions.

An important special type of problem that can be effectively addressed using the NP method is *mixed integer programs* (MIP) [6]. For such problems there may be one set of discrete variables and one set of continuous variables and the objective function and constraints are both linear. A general MIP can be stated as follows:

$$z_{\text{MIP}} = \min_{x,y \in X} c^1 x + c^2 y\,, \tag{2}$$

where $X = \left\{x \in Z_+^n, y \in R^n : A^1 x + A^2 y \le b\right\}$ and we use z_{MIP} to denote any linear objective function, that is, $z_{\text{MIP}} = f(x) = cx$. While some large-scale MIPs can be solved efficiently using exact mathematical programming methods, complex applications often give rise to MIPs where exact solutions can only be found for relatively small problems. When dealing with such complex large-scale problems the NP method provides an attractive alternative. However, even in such cases it may be possible to take advantage of exact mathematical programming methods by incorporating them into the NP framework. The NP method therefore provides a framework for combining the complementary bene-

fits of two optimization approaches that have traditionally been studied separately, namely, mathematical programming and metaheuristics.

Another important class of problems are *combinatorial optimization problems* (COP) where the feasible region is finite but its size typically grows exponentially with the number of input parameters of the problem. A general COP can be stated as follows:

$$\min_{x \in X} f(x) , \tag{3}$$

where $|X| < \infty$, but the objective function $f: X \to \mathbf{R}$ may be a complex nonlinear function. Sometimes it may have no analytic expression and must be evaluated through a model, such as a simulation model, a data mining model, or other application-dependent models. One advantage of the NP method is that it is effective for optimization when f is known analytically (deterministic optimization), when it is noisy (stochastic optimization), or even when it must be evaluated using an external process.

Methods

The NP method is best viewed as a metaheuristic framework, and it has similarities to branching methods in that like branch-and-bound it creates partitions of the feasible region. However, it also has some unique features that make it well suited for very hard large-scale optimization problems.

Metaheuristics have emerged as the most widely used approach for solving difficult large-scale combinatorial optimization problems [2]. A metaheuristic provides a framework to guide application-specific heuristics, such as a greedy local search, by restricting which solution or set of solutions should or can be visited next. For example, the tabu search metaheuristic disallows certain moves that might otherwise be appealing by making the reverse of recent moves tabu or forbidden. At the same time it always forces the search to take the best nontabu move, which enables the search to escape local optima. Similar to tabu search, most metaheuristics guide the search from solution to solution or possibly from a set of solutions to another set of solutions. In contrast, the NP method guides the search by determining where to concentrate the search effort. Any optimization method, such as an application-specific local search, other general purpose heuristic, or a mathematical programming method, can then be integrated within this framework.

The development of metaheuristics and other heuristic search methods has been made largely in isolation from the recent advancements in the use of mathematical programming methods for solving large-scale discrete problems. It is a very important and novel characteristic of the NP method that it provides a natural metaheuristic framework for combining the use of heuristics and mathematical programming and taking advantage of their complementary nature. Indeed, as far as we know, the NP method is the first systematic search method that enables users to simultaneously realize the full benefits of incorporating lower bounds through various mathematical programming methods and using any domain knowledge or heuristic search method for generating good feasible solutions. It is this flexibility that makes the NP method so effective for practical problems.

To concentrate the search effort, the NP method employs a decomposition approach similar to that of branch-and bound. Specifically, in each step the method partitions the space X of feasible solutions into the *most promising region* and the *complementary region*, namely, the set of solutions not contained in the most promising region. The most promising region is then partitioned further into subregions. The partitioning can be done exactly as branching for a branch-and-bound algorithm, but instead of focusing on obtaining lower bounds and comparing those bounds with a single primal feasible solution, the NP methods focuses on generating primal feasible solution from each of the subregions and the complementary region. This results in an upper bound on the performance of each of these regions. The region with the best feasible solution is judged the most promising and the search is focused accordingly. A best upper bound does not guarantee that the corresponding subset contains the optimal solution, but since the NP method also finds primal feasible solutions for the complementary region it is able to recover from incorrect moves. Specifically, if the best solution is found in one of the subregions this becomes the new most promising region, where if it is in the complementary region the NP method backtracks. This focus on generating primal feasible solutions and the global perspective it achieves

through backtracking are distinguishing features of the NP method that set it apart from similar branching methods.

Unlike exact optimization methods such as branch-and-bound the NP method does not guarantee that the correct region is selected in each move of the algorithm. Incorrect moves can be corrected through backtracking, but for the method to be both effective and efficient, the correct move must be made frequently. How this is accomplished depends on how the feasible solutions are generated.

In what we refer to as the *pure NP method* feasible solutions are generated using simple uniform random sampling. To increase the probability of making the correct move the number of samples should be increased. A purely uniform random sampling is rarely efficient, however, and the strength of the NP method is that it can incorporate application-specific methods for generating feasible solutions. In particular, for practical applications domain knowledge can often be utilized to very effectively generate good feasible solutions. We call such implementations *knowledge-based NP methods*. We will also see examples of what we refer to as *hybrid NP methods* where feasible solutions are generated using either general heuristic methods such as greedy local search, genetic algorithm, or tabu search, or mathematical programming methods. If done effectively, incorporating such methods into the NP framework makes it more likely that the correct move is made and hence makes the NP method more efficient. Indeed, such hybrid and knowledge-based implementations are often an order of magnitude more efficient than uniform random sampling.

In addition to the method for generating feasible solutions, the probability of making the correct move depends heavily on the partitioning approach. A generic method for partitioning is usually straightforward to implement but by taking advantage of special structure and incorporating this into intelligent partitioning the efficiency of the NP method may be improved by an order of magnitude. The strength of the NP method is indeed in this flexibility. Special structure, local search, any heuristic search, and mathematical programming can all be incorporated into the NP framework to develop optimization algorithms that are more effective in solving large-scale optimization problems than when these methods are used alone.

Cases

Here we introduce three application examples that illustrate the type of optimization problems for which the NP method is particularly effective. For each application the optimization problem has a complicating aspect that makes it difficult for traditional optimization methods. For the first of these problems, resource-constrained project scheduling, the primary difficulty is in a set of complicating constraints. For the second problem, the feature selection problem, the difficulty lies in a complex objective function. The third problem, radiation treatment planning, has both difficult-to-satisfy constraints and a complex objective function that cannot be evaluated through an analytical expression. Each of the three problems can be solved effectively by the NP method by incorporating our understanding of the application into the framework.

Resource-Constrained Project Scheduling

Planning and scheduling problems arise as critical challenges in many manufacturing and service applications. One such problem is the resource-constrained project scheduling problem that can be described as follows. A project consists of a set of tasks to be performed and given precedence requirements between some of the tasks. The project scheduling problem involves finding the starting time of each task so that the overall completion time of the project is minimized. It is well known that this problem can be solved efficiently using what is called the critical path method that uses forward recursion to find the earliest possible completion time for each task. The completion time of the last task defines the makespan or the completion time of the entire project.

Now assume that one or more resources are required to complete each task. The resources are limited so if a set of tasks requires more than the available resources they cannot be performed concurrently. The problem now becomes NP-hard and cannot be solved efficiently to optimality using any traditional methods. To state the problem we need the following notation: V is the set of all tasks, E is the set of precedence constraints, p_i is the processing time of task $i \in V$, R is the set of resources, R_k are the available resources of type $k \in R$, and r_{ik} are resources of type k required by task i.

The decision variables are the starting times for each task: x_i is the starting time of task $i \in V$.

Finally, for notational convenience we define the set of tasks processed at time t as

$$V(t) = \{i : x_i \le t \le x_i + p_i\} \ . \tag{4}$$

With this notation, we now formulate the resource-constrained project scheduling problem mathematically as follows:

$$\min \max_{i \in V} x_i + p_i \ , \tag{5}$$

$$x_i + p_i \le x_j, \quad \forall (i,j) \in E \ , \tag{6}$$

$$\sum_{i \in V(t)} r_{ik} \le R_k, \quad \forall k \in R, t \in \mathbf{Z}_+^1 \ , \ x_i \in \mathbf{Z}_+^1 \ . \tag{7}$$

Here the precedence constraints (6) are easy, whereas the resource constraints (7) are hard. By this we mean that if the constraints (7) are dropped then the problem becomes easy to solve. Such problems, where complicating constraints transform the problem from easy to very hard, are common in large-scale optimization. Indeed the classic job shop scheduling problem can be viewed as a special case of the resource-constrained project scheduling problem where the machines are the resources. Without the machine availability constraints the job shop scheduling problem reduces to a simple project scheduling problem. Other well-known combinatorial optimization problems have similar properties. For example, without the subset elimination constraints the classic traveling salesman problem reduces to a simple assignment problem that can be solved efficiently.

The flexibility of the NP method allows us to address such problems effectively by taking advantage of special structure when generating feasible solutions. It is important to note that it is very easy to use sampling to generate feasible solutions that satisfy very complicated constraints. Therefore, when faced with a problem with complicating constraints we want to use random sampling to generate partial feasible solutions that resolve the difficult part of the problem and then complete the solution using the appropriate efficient optimization method.

For example, when a feasible solution for the resource-constrained-project scheduling problem is generated, the resource allocation should be generated using random sampling and the solution can then be completed by applying the critical path method to determine the starting times for each task. This requires reformulating the problem so that the resource and precedence constraints can be separated, but such a reformulation is rather easily achieved by noting that the resource constraints can be resolved by determining a sequence between the tasks that require the same resource(s) at the same time. Once this sequence has been determined it can be added as precedence constraints and the remaining solution can be generated using the critical path method. Feasible solutions can therefore be generated in the NP method by first randomly sampling a sequence to resolve resource conflicts and then applying the critical path method. Both procedures are very fast, so complete sample solutions can be generated rapidly.

We also note that constraints that are difficult for optimization methods such as mathematical programming are sometimes very easily addressed in practice by incorporating domain knowledge. For example, a domain expert may easily be able to specify priorities among tasks requiring the same resource(s) in the resource-constrained project scheduling problem. The domain expert can therefore, perhaps with some assistance from an interactive decision support system, specify some priority rules to convert a very complex problem into an easy-to-solve problem. The NP method can effectively incorporate such domain knowledge into the optimization framework by using the priority rules when generating feasible solutions. This is particularly effective because the domain expert would not need to specify priority rules to resolve all resource conflicts. Rather, any available priority rule or other domain knowledge can be incorporated to guide the sampling.

The same structure can be used to partition intelligently. Instead of partitioning directly using the decision variables (x_i), we note that it is sufficient to partition to resolve the resource conflicts. Once those are resolved then the problem is solved. This approach is applicable to any problem that can be decomposed in a similar manner.

Feature Selection

Knowledge discovery and data mining is a relatively new field that has experienced rapid growth owing to

its ability to extract meaningful knowledge from very large databases. One of the problems that must usually be solved as part of practical data mining projects is the feature selection problem, which involves selecting a good subset of variables to be used by subsequent inductive data mining algorithms. The problem of selecting a best subset of variables is well known in the statistical literature as well as in machine learning. The recent explosion of interest in data mining for addressing various business problems has led to a renewed interest in this problem. From an optimization point of view, feature selection can clearly be formulated as a COP where binary decision variables determine if a feature (variable) is included or excluded. The solution space can therefore be stated very simply as all permutations of a binary vector of length n, where n is the number of variables. The size of this feasible region is 2^n, so it experiences exponential growth, but typically there are no additional constraints to complicate its structure.

On the other hand, there is no consensus objective function that measures the quality of a feature or a set of features. Tens of alternatives have been proposed in the literature, including both functions that measure the quality of individual features and functions that measure the quality of a set of features. However, no single measure is satisfactory in all cases and the ultimate measure is therefore: Does it work? In other words, when the features selected are used for learning does it result in a good model being induced? The most effective feature selection approach in terms of solution quality is therefore the wrapper approach, where the quality of a set of features is evaluated by applying a learning algorithm to the set and evaluating its performance. Specifically, an inductive learning algorithm, such as decision tree induction, support vector machines or neural networks, are applied to training data containing only the features selected. The performance of the induced model is evaluated and this performance is used to measure the quality of the feature subset. This objective function is not only nonlinear, but since a new model must be induced for every feature subset it is also very expensive to evaluate.

Mathematically, the feature selection can be stated as the following COP:

$$\min_{x\in\{0,1\}^n} f(x)\,, \tag{8}$$

that is, $X = \{0,1\}^n$. Feature selection is therefore a very hard COP not because of the complexity of the feasible region, although it does grow exponentially, but owing to the complexity of an objective function that is very expensive to evaluate. However, this is also an example where application-specific heuristics can be effectively exploited by the NP method.

Significant research has been devoted to methods for measuring the quality of features. This includes information-theoretic methods such as using Shannon's entropy to measure the amount of information contained in each feature: the more information, the more valuable the feature. The entropy is measured for each feature individually and it can hence be used as a very fast local search or a greedy heuristic, where the features with the highest information gain are added one at a time. While such a purely entropy based feature selection will rarely lead to satisfactory results, the NP method can exploit this by using the entropy measure to define an intelligent partitioning.

We let $X(k) \subseteq X$ denote the most promising region in the kth iteration and partition the set into two disjoint subsets (note that $X(0) = X$):

$$X_1(k) = \{x \in X(k)\colon x_i = 1\}\,, \tag{9}$$

$$X_2(k) = \{x \in X(k)\colon x_i = 0\}\,. \tag{10}$$

Hence, a partition is defined by a sequence of features $x_1, x_2, \ldots, x_n$, which determines the order in which the features are either included ($x_i = 1$) or excluded ($x_i = 0$).

We calculate the information gain $\mathrm{Gain}(i)$ of feature i, which is the expected reduction in entropy that would occur if we knew the value of feature i, that is,

$$\mathrm{Gain}(i) = I - E(i)\,, \tag{11}$$

where I is the expected information that is needed to classify a given instance and $E(i)$ is the entropy of each feature. The maximum information gain, or equivalently the minimum entropy, determines a ranking of the features. Thus, we select

$$i_1 = \arg \min_{i\in\{1,2,\ldots,n\}} E(i)\,,$$

$$i_2 = \arg \min_{i \in \{1,2,\ldots,n\} \setminus \{i_1\}} E(i) ,$$

$$\vdots$$

$$i_n = \arg \min_{i \in \{1,2,\ldots,n\} \setminus \{i_1,\ldots,i_{n-1}\}} E(i) .$$

The feature order $i_1, i_2, \ldots, i_n$ defines an intelligent partition for the NP method and this has been found to be an order of magnitude more efficient than an average arbitrary partitioning [3]. We can use a similar idea to generate feasible solutions from each region using a sampling strategy that is biased towards including features with high information gain. A very fast greedy heuristic can thus greatly increase the efficiency of the NP method while resulting in much higher quality solutions that the greedy heuristic is able to achieve on its own.

Radiation Treatment Planning

Health care delivery is an area of immense importance where optimization techniques have been used increasingly in recent years. Radiation treatment planning is an important example of this and intensity-modulated radiation therapy (IMRT) is a recently developed complex technology for such treatment. It employs a multileaf collimator to shape the beam and to control, or modulate, the amount of radiation that is delivered from each of the delivery directions (relative to the patient). The planning of the IMRT is very important because it needs to achieve the treatment goal while incurring the minimum possible damage to other organs. Because of its complexity the treatment planning problem is generally divided into several subproblems. The first of these is termed the *beam angle selection* (BAS) problem. In essence, BAS requires the determination of roughly four to nine angles from 360 possible angles subject to various spacing and opposition constraints.

Designing an optimal IMRT plan requires the selection of beam orientations from which radiation is delivered to the patient. These orientations, called beam angles, are currently manually selected by a clinician on the basis of his/her judgment. The planning process proceeds as follows. A dosimetrist selects a collection of angles and waits 10–30 min while a dose pattern is calculated. The resulting treatment is likely to be unacceptable, so the angles and dose constraints are adjusted, and the process is repeated. Finding a suitable collection of angles often takes several hours. The goal of using optimization methods to identify quality angles is to provide a better decision support system to replace the tedious repetitive process just described. An integer programming model of the problem contains a large number of binary variables and the objective value of a feasible point is evaluated by solving a large, continuous optimization problem. For example, in selecting five to ten angles, there are between 4.9^{10} and 8.9×10^{19} subsets of $0, 1, 2, \ldots, 359$.

The BAS problem is complicated by both an objective function with no analytical expression and constraints that are hard to satisfy. In the end an IMRT plan is either acceptable or not and the considerations for determining acceptability are too complex for a simple analytical model. Thus, the acceptability and hence the objective function value for each plan must be evaluated by a qualified physician. This makes evaluating the objective not only expensive in terms of time and effort, but also introduces noise into the objective function because two physicians may not agree on the acceptability of a particular plan. The constraints of the BAS problem are also complicated since each beam angle will result in radiation of organs that are not the target of the treatment. There are therefore two types of constraints: the target should receive the minimum amount of radiation and other organs should receive no more than some maximum amount of radiation. Since these bounds need to be specified tightly the constraints are hard to satisfy.

The BAS problem illustrates how mathematical programming can be effectively incorporated into the NP framework. Since the evaluation of even a single IMRT plan must be done by an expert and is hence both time-consuming and expensive, it is imperative to impose a good structure on the search space that reduces the number of feasible solutions that need to be generated. This can be accomplished through an intelligent partitioning, and specifically by computing the optimal solution of an integer program with a much simplified objective function [1]. The output of the integer program then serves to define an intelligent partitioning. For example, suppose a good angle set (50°, 80°, 110°, 250°, 280°, 310°, 350°) is found by solving the integer program. We can then partition on the first angle in the set, which is 50° in this example. Then one subregion includes angle 50°, and the other excludes 50°.

Conclusions

The NP method is a powerful metaheuristic for solving large-scale discrete optimization problems. The method systematically partitions the feasible region into subregions and moves from one region to another on the basis of information obtained by randomly generating feasible sample solutions from each of the current regions. The method keeps track of which part of the feasible region is the most promising in each iteration and the number of feasible solutions generated, and hence the computational effort is always concentrated in this most promising region.

The efficiency of the NP algorithm depends on making the correct move frequently. This success probability depends in turn on both the partitioning and the method for generating feasible solutions. For any practical application it is therefore important to increase the success probability by developing intelligent partitioning methods, incorporating special structure into weighted sampling, and applying randomized heuristics to generate high-quality feasible solutions.

The NP method has certain connections to standard mathematical programming techniques such as branch-and-bound. However, the NP method is primarily useful for problems that are either too large or too complex for mathematical programming to be effective. But even for such problems mathematical programming methods can often be used to solve either a relaxed problem or a subproblem of the original and these solutions can be effectively incorporated into the NP framework.

The three application examples presented here illustrate the broad usefulness of the NP method in both manufacturing and service industries, and how it can take advantage of special structure and application-specific heuristics to improve the efficiency of the search.

References

1. D'Souza WD, Meyer RR, Shi L (2004) Selection of beam orentations in intensity-modulated radiation therapy using single-beam indices and integer programming. Phys Med Biol 49:3465–3481
2. Gendreau M, Potvin J (2005) Metaheuristics in Combinatorial Optimization. Annu Oper Res 140:189–213
3. Ólafsson S, Yang J (2005) Intelligent Partitioning for Feature Selection. INFORMS J Comput 17(3):339–355
4. Shi L, Ólafsson S (2000) Nested Partitions Method for Global Optimization. Oper Res 48:390–407
5. Shi L, Ólafsson S, Chen Q (2001) An Optimization Framework for Product Design. Manag Sci 47:1681–1692
6. Wolsey L (1998) Integer Programming. Wiley, New York

Network Design Problems

NDP

DIETMAR CIESLIK
University Greifswald, Greifswald, Germany

MSC2000: 05C05, 05C40, 68R10, 90C35

Article Outline

Keywords
See also
References

Keywords

Network; Shortest path; Minimum spanning trees; Steiner minimal tree; Network flows

Scientific or engineering applications usually require solving mathematical problems. Such applications in accordance with networks span a wide range, from modeling the evolution of species in biology to modeling soap films for grids of wires; from the design of collections of data to the design of heating or air-conditioning systems in buildings; and from the creation of oil and gas pipelines to the creation of communication networks, road and railway lines. These are all network design problems of significant importance and nontrivial complexity. The network topology and design characteristics of these systems are classical examples of optimization problems.

I. Intuitively speaking, a *network* is a set of points and a set of connections where each connection joins one point to another and has a certain length. The combinatorial structure of such a network is described as a graph G which is defined to be a pair (V, E) where

- V is any finite set of elements, called *vertices*, and
- E is a finite family of elements which are unordered pairs of vertices, called *edges*.

Additionally, assume that a function $l\colon E \to \mathbf{R}$ is given for the edges of the graph G. Usually, assume that l

has only positive values and call it a length-function. A (connected) graph equipped with a length-function is called a *network*.

The *length of a subgraph* $G' = (V', E')$, $V' \subseteq V$, $E' \subseteq E \cap \binom{V'}{2}$, of the network G is defined as

$$L(G') := \sum_{e \in E'} l(e).$$

Each network $G = (V, E)$ with the length-function l: $E \to \mathbf{R}$ is a metric space (V, ρ) by defining the distance function in the way that $\rho(v, v')$ is the length of a shortest path between the vertices v and v' in G. That means that ρ: $V \times V \to \mathbf{R}$ is a real-valued function satisfying:

i) $\rho(v, v') \geq 0$ for all v, v' in V;
ii) $\rho(v, v') = 0$ if and only if $v = v'$;
iii) $\rho(v, v') = \rho(v', v)$ for all v, v' in V; and
iv) $\rho(v, v') \leq \rho(v, w) + \rho(w, v')$ for all v, v', w in V (*triangle inequality*).

The *problem of finding shortest paths* in a graph with a length-function is an important and well-studied problem. Such a path is easy to find by an algorithm created by E.W. Dijkstra [7]:

```
PROCEDURE shortest path
    InputInstance();
    Start with the vertex v;
    Label the vertex v with 0 :  L(v) = 0;
    REPEAT
        Select an edge ww' between a labeled vertex w and an unlabeled vertex w' such that the quantity L(w)+l(ww') is minimal as possible;
        Label w' :  L(w') = L(w) + l(ww')
    UNTIL v' is achieved
END shortest path;
```

A pseudocode for a procedure finding a shortest path between two vertices v and v' in a network

Assume that the procedure runs if all vertices of the network are achieved then the procedure creates a spanning tree rooted at the vertex v containing shortest paths from v to every vertex. Moreover, the label $L(w)$ denotes the distance from v to w, in other terms, $\rho(v, w) = L(w)$.

Conversely, each finite metric space can be represented as a graph with a nonnegative length-function [23].

Graphs lend themselves as natural models of transportation as well as communication networks. Consequently, it is natural to study network design problems such as optimal facility location problems for graphs and as graphs in metric spaces.

The core network design problem is the *minimum spanning tree problem*, where one wish to design a minimum cost network contains a path from each vertex to each other. Such a network must be a tree, which is called a *minimum spanning tree* (MST). Creating a minimum spanning tree is the problem one has the longest history of all ND problems, starting with O. Borůvka [2] in 1926. See [19] for an excellent historical survey.

All the known efficient minimum spanning tree algorithms are special cases of a general greedy method, in which one builds up an MST edge-by-edge, including appropriate short edges and excluding appropriate long edges ones. Perhaps the simplest method to find an MST is due to J.B. Kruskal [30]:

```
PROCEDURE minimum spanning tree
    InputInstance();
    Start without any edge;
    REPEAT
        Choose the shortest edge that does not form
        form a circle with edges already chosen
    UNTIL | V | −1 edges are chosen
END minimum spanning tree;
```

A pseudocode for a procedure finding a minimum spanning tree in a network $G = (V, E)$

About an efficient implementation of Kruskal's algorithm and several more effective procedures compare [3].

II. A general ND problem is for a given configuration of vertices and/or edges to find a network which contains these objects, fulfilling some predetermined requirements and minimizes a given objective function. This is quite general and models a wide variety of problems.

J. McGregor Smith [40] presents a classification of applications for network design problems. Generalizing this is

- *Large region networks*. The metric in large geographic regions is given by the shortest great circle distance between the points on the (Euclidean)

sphere. Large region location problems arise when the difference between the Euclidean and the great circle metrics is considerable. Location of international headquarters or distribution centers and planning oil or natural gas pipelines or long distance telephone lines are examples [27] and [38].

- *Regional networks*. Consider inter-urban networks, like communication networks, railway lines and interstate highway networks. R.F. Love and J.G. Morris [33] study a variety of mathematical forms as samples for intercity, urban and rural road distances. Moreover, they found that the p-norms, and linear combinations of these, are the best possible. Recent surveys are [4] and [34].
- *Macro scale networks*. Chemical processing plants, urban arterial systems and similar intra-urban systems are typical applications. In these situations the rectilinear metric is often used. If the structure of the possible connections is predetermined it is also possible to formulate the problem as a network design problem in graphs.
- *Intermediate scale networks*. Electric, heating and air-conditioning systems in buildings are examples of network optimization problems where Steiner points can reduce the overall minimum cost solution of the network. The rectilinear metric is the most frequent measure of the distance in these applications. For models and methods see [26].
- *Micro scale networks*. The design of very large scale integration (VLSI) networks is an example of Steiner's problem where the overall interconnecting length of the network is crucial for the solution. In this class of applications, the rectilinear metric is again the most frequent metric. It is also conceivable to use a linear combination of rectilinear and maximum norm [26] and [32].
- *Evolutionary networks*. Molecular sequences are used to reconstruct the course of the evolution. Since the evolution is assumed to have proceed from a common ancestral species in a tree-like branching of species, this process is generally modeled by a tree. The key question is the reconstruction of this tree based on the contemporary data. Molecular data comes as either DNA sequences (composed of nucleotides from an alphabet of four letters; namely the four nucleic acids Adenine, Guanine, Thymine and Cytosine) or sequences of proteins (composed of amino acids from an alphabet of 20 letters; namely Alanine, Arginine, Asparagine, …, Valine). As the metric often the Hamming distance is used. Surveys are given in [13] and [39].

III. Considering a group of ND problems which are the problems with connectivity requirements one finds:

- The *bounded degree minimum spanning tree problem* (abbreviated: BDMST-problem), which is defined as follows: Given a finite set N of points in a metric space and an integer $\beta > 1$. Find a spanning tree $T = (N, E)$ such that T has minimal length among all candidates with a maximum degree of the vertices less or equal than β.

  Finding a BDMST of maximum degree $\beta = 2$ is equivalent to solving the *traveling salesman problem*, which is known to be NP-hard. The borderline which divides the classes NP-hard and $\mathcal{P}$ for the BDMST problem depending on the space and the quantity β are described in [5] and [37].
- Many modified (minimum) spanning tree problems are presented in the literature, for instance
  - find a spanning tree interconnecting all vertices with minimal maximum degree;
  - find a spanning tree interconnecting all vertices with at least k leaves in the tree;
  - find a spanning tree $T = (V, E)$ such that the quantity

    $$\sum_{v,v' \in V} L(\text{the path from } v \text{ to } v' \text{ in } T)$$

    is minimal;
  - find a spanning tree isomorphic to a given tree;

  These problems are NP-complete in general, but it is shown that they can be solved more easily in several specific cases, [15,17], and [28].
- The *Steiner minimal tree problem*, where one seek a minimum network that connects a set N of designated terminal points. Any network solving this problem must be a tree, which is called a *Steiner minimal tree* (SMT). It may contain vertices different from the points which are to be connected. Such points are called *Steiner points*. In other terms; an SMT for N is a minimum spanning tree on $N \cup Q$, where Q is a set of additional vertices inserted into the metric space in order to achieve a minimal solution. In general, however, it is impossible to compute the number of Steiner points in an easy way

independently from the determination of an SMT. Additionally, Steiner point locations in the space are not prespecified from a candidate list of point locations.

Steiner's problem for graphs was originally formulated by S.B. Hakimi [22] in 1971. Since then, the problem has received considerable attention in the literature. Several exact algorithms and heuristics have been suggested and discussed.

R.M. Karp [29] showed that Steiner's problem is NP-complete. An algorithm which finds a solution in exponential time is given in [8]. The algorithm is based on the dynamic programming methodology using a decomposition property. On the other hand, Hakimi [22] remarked that an SMT for N in a network $G = (V, E)$ can be found by the enumeration of minimum spanning trees of subgraphs of G induced by supersets of N. E.L. Lawler [31] suggested a modification of this algorithm, using the fact that the number of Steiner points is bounded by $|N| - 2$ which shows that not all subsets V' must be considered.

A recent survey about Steiner minimal trees in networks is given in [26].

Algorithmic problems on arbitrary graphs often remain NP-complete even when restricted to special classes of graphs, yet may becomes solvable in polynomially bounded time on others. For instance, Steiner's problem remains NP-complete even for planar graphs [14]; yet, it can be solved in linear time in several specific graphs [41,43,44].

The geometric version of the Steiner minimal tree problem was originated by P. Fermat [10] early in the 17th century and by C.F. Gauss [16] in 1836. Perhaps starting with the book [6], in 1941, the *Gauss problem* became popularized under the name of *Steiner's problem*. That is: Given finite set of points in a Euclidean space, find a network which connects all points of the set with minimal length.

A classical survey of Steiner's problem in the Euclidean plane is [18] and is termed 'Steiner minimal tree' for the shortest interconnecting network and 'Steiner points' for the additional vertices.

Without loss of generality, the following is true for any SMT for a finite set N of points:

1) the degree of each vertex is at most three;
2) the degree of each Steiner point equals three; and two edges is incident to a Steiner point meet at as angle of 120°;
3) there are at most $|N| - 2$ Steiner points.

In the Euclidean plane one has a geometric construction by ruler and compass originated in [35] and [42]. Recent (1999) surveys about Steiner minimal trees, also in other geometries than the Euclidean ones, are given in [4] and [26].

Clearly, an MST is an approximation of an SMT. More exactly: One can find a tree interconnecting a finite set of points in a metric space in fast time (namely, $O(n^2)$-time, where n is the number of given points or the number of vertices in N, respectively) with a length at most twice the length of a shortest possible tree, namely an SMT. Hence, it is of interest to consider the quantity

$$\inf\left\{\frac{L(\text{SMT for } N)}{L(\text{SMT for } N)}:\ \begin{array}{c} N \text{ a finite set} \\ \text{of points}\end{array}\right\},$$

which is called the *Steiner ratio* of the space and says how much the total length of an MST can be decreased by allowing Steiner points.

space	Steiner ration	source
Plane with rectilinear norm	$2/3 = 0.66666\cdots$	[25]
Euclidean norm	$\sqrt{3}/2 = 0.86602\cdots$	[9]
Plane with p-norm	$2/3 \le m \le \sqrt{3}/2$	[4]

IV. Most of the ND problems can be modeled by an integer program.

Let $G = (V, E$ be a graph. A cut S in G is a partition of the vertex-set V into two nonempty parts, S and $V \setminus S$. An edge e crosses the cut S, written by $e \in \gamma(S)$, if it has exactly one endpoint in each part. Now, let l: $E \to \mathbf{R}$ be a length-function. Define the following integer linear program:

$$\begin{cases} \min & \displaystyle\sum_{e\in E} l(e)\cdot x_e \\ \text{s.t.} & \displaystyle\sum_{e\in\gamma(S)} x_e \ge p(S), \quad \emptyset \ne S \subset V, \\ & x_e \in \{0,1\}, \quad e \in E, \end{cases}$$

whereby p: $2^V \to \mathbf{N}$ is a function parametrizing the ND problem which is to solve. If one has two vertices v and v', and sets $p(S) = 1$ when S contains v but not v', then the program models the shortest path problem (between v and v'). If $p(S) = 1$ for all cuts S, then the program models the minimum spanning tree problem. Other ND problems with connectivity constraints discussed by integer linear programs are mentioned in [20] and [21].

V. The second group of ND problems are the problems with capacity constraints.

Let $G = (V, E)$ be a directed graph, usually called a *digraph*, that is

- V is a finite set of elements, called *vertices*, and
- $E \subseteq V \times V$ is a finite family, called the *set of edges*.

Assume, that there are two distinguished vertices, a source v_0 and a sink v_1 in G, and that there is a (directed) path from v_0 to v_1. Additionally, assume that there is a nonnegative capacity-function c: $E \to \mathbf{R}$.

A flow f on the digraph G is a nonnegative function on the edges such that

1) f does not exceed the capacities: $0 \leq f(e) \leq c(e)$ for every edge e; and
2) f satisfies the so-called *Kirchhoff-condition*

$$\sum_{(u,v)\in E} f(u,v) = \sum_{(v,w)\in E} f(v,w)$$

for every vertex $v \in V \setminus \{v_0, v_1\}$.

The quantity

$$\sum_{(v_0,v)\in E} f(v_0,v) = \sum_{(v,v_1)\in E} f(v,v_1)$$

is called the *value of the flow* f. The problem is to find a flow of maximum value, called a *maximum flow*.

The fundamental theory of network flows was developed by L.R. Ford Jr. and D.R. Fulkerson [11,12]: Similarly as above, a cut is defined to be a vertex partition S and $V \setminus S$ such that $v_0 \in S$ and $v_1 \in V \setminus S$. The capacity of the cut is

$$c(S) = \sum_{e\in\gamma(S)} c(e).$$

Ford and Fulkerson's main result, the *max-flow min-cut theorem*, states that the maximum flow value equals the minimum cut capacity.

Ford and Fulkerson proved this theorem by devising an algorithm that, given a flow f, either finds a cut whose capacity equals the flow value or finds a way to increase the the flow value along an augmenting path from v_0 to v_1.

```
PROCEDURE maximum flow
    InputInstace();
    Start with the zero flow;
    REPEAT
        find an augmenting path from v0 to v1;
        increase the flow value by altering the flows
        along the edges of the path
    UNTIL it no longer applies
END maximum flow;
```

A pseudocode for a procedure finding a maximum flow from a source v_0 to a sink v_1 in a graph $G = (V, E)$ equipped with a capacity-function

If all capacities are integers this procedure produces a maximum flow. Note, if the capacities are arbitrary real numbers the algorithm need never terminate, and successive flow values, though they will converge, need not converge to the maximum flow value.

Surveys for network flow algorithms, including classical work and a discussion of complexity, are [1] and [36].

VI. Combining ND problems with connectivity and capacity constraints there is the minimum cost flow problem, which is to determine a least cost shipment of a commodity through a network in order to satisfy the network possibilities.

Let $G = (V, E)$ be a digraph. Associated with each vertex v there is a number $b(v) \in \mathbf{R}$, satisfying $\sum_{v\in V} b(v) = 0$. A vertex v is called a *source*, a *sink* or a *transshipment vertex* if $b(v)$ is positive, negative or zero, respectively. Additionally, assume that there is a nonnegative capacity-function c: $E \to \mathbf{R}$ and a positive cost-function (i. e., length-function) l: $E \to \mathbf{R}$. Then the *minimum cost flow problem* is formulated as

$$\begin{cases} \min & \displaystyle\sum_{e\in E} l(e)\cdot x_e \\ \text{s.t.} & \displaystyle\sum_{(w,v)\in E} x_{(w,v)} - \sum_{(v,w)\in E} x_{(v,w)} = b(v), \\ & \quad v \in V, \\ & 0 \leq x_e \leq c(e), \qquad e \in E, \\ & x_e \quad \in \{0,1\}, \quad e \in E. \end{cases}$$

This problem is discussed in the literature many times; one of several available good sources is [1] which includes several polynomial time algorithms. A general background is [24].

See also

- Auction Algorithms
- Communication Network Assignment Problem
- Directed Tree Networks
- Dynamic Traffic Networks
- Equilibrium Networks
- Evacuation Networks
- Generalized Networks
- Maximum Flow Problem
- Minimum Cost Flow Problem
- Network Location: Covering Problems
- Nonconvex Network Flow Problems
- Piecewise Linear Network Flow Problems
- Shortest Path Tree Algorithms
- Steiner Tree Problems
- Stochastic Network Problems: Massively Parallel Solution
- Survivable Networks
- Traffic Network Equilibrium

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms and applications. Prentice-Hall, Englewood Cliffs
2. Borůvka O (1926) O jistém problému minimálnim. Práca Moravské Prirodovědecké Spolnečnosti 3:37–58
3. Cheriton D, Tarjan RE (1976) Finding minimum spanning trees. SIAM J Comput 5:724–742
4. Cieslik D (1998) Steiner minimal trees. Kluwer, Dordrecht
5. Cieslik D (1998) Using Hadwiger numbers in network design. In: DIMACS, 40. Am Math Soc, Providence, pp 59–78
6. Courant R, Robbins H (1941) What is mathematics? Oxford Univ. Press, Oxford
7. Dijkstra EW (1959) A note on two problems in connection with graphs. Numer Math 1:269–271
8. Dreyfus SE, Wagner RA (1972) The Steiner problem in graphs. Networks 1:195–207
9. Du D-Z, Hwang FK (1992) A proof of the Gilbert–Pollak conjecture on the Steiner ratio. Algorithmica 7:121–136
10. Fermat P (1934) Abhandlungen über Maxima und Minima. Oswalds Klassiker der exakten Wissenschaft, vol 238. H. Miller, reprint from original.
11. Ford LR Jr, Fulkerson DR (1956) Maximal flow through a network. Canad J Math 8:399–404
12. Ford LR Jr, Fulkerson DR (1962) Network flow theory. Princeton Univ. Press, Princeton
13. Foulds LR (1994) Graph theory applications. Springer, Berlin
14. Garey MR, Johnson DS (1977) The rectilinear Steiner tree problem is NP-complete. SIAM J Appl Math 32:826–834
15. Garey MR, Johnson DS (1979) Computers and intractibility. Freeman, New York
16. Gauss CF (1917) Briefwechsel Gauss–Schuhmacher. In: Werke, vol. X. pp 459–468
17. Gavish B (1982) Topological design of centralized computer networks - Formulations and algorithms. Networks 12:355–377
18. Gilbert EN, Pollak HO (1968) Steiner minimal trees. SIAM J Appl Math 16:1–29
19. Graham RL, Hell P (1985) On the history of the minimum spanning tree problem. Ann Hist Comput 7:43–57
20. Grötschel M, Monma CL (1990) Integer polyhedra arising from certain network design problems with connectivity constraints. SIAM J Discret Math 3:502–523
21. Grötschel M, Monma CL, Stoer M (1994) Design of survivable networks. In: Handbook Oper Res and Management Sci. North-Holland, Amsterdam
22. Hakimi SB (1971) Steiner's problem in graphs and its implications. Networks 1:113–133
23. Hakimi SL, Yau SS (1964) Distance matrix of a graph and its realizability. Quart Appl Math 22:305–317
24. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
25. Hwang FK (1976) On Steiner minimal trees with rectilinear distance. SIAM J Appl Math 30:104–114
26. Hwang FK, Richards DS, Winter P (1992) The Steiner tree problem. North-Holland, Amsterdam
27. Ivanov AO, Tuzhilin AA (1994) Minimal networks - The Steiner problem and its generalizations. CRC Press, Boca Raton
28. Jungnickel D (1994) Graphen, Netzwerke und Algorithmen. BI Wissenschaftsverlag, Mannheim
29. Karp RM (1962) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations. Springer, New York, pp 85–103
30. Kruskal JB (1956) On the shortest spanning subtree of a graph and the travelling salesman problem. Proc 7:48–50
31. Lawler EL (1976) Combinatorial optimization - Networks and matroids. Holt, Rinehart and Winston, New York
32. Lengauer T (1990) Combinatorial algorithms for integrated circuit layout. Teubner and Wiley, Stuttgart
33. Love RF, Morris JG (1972) Modelling inter-city road distances by mathematical function. J Oper Res Soc 23:61–71
34. Love RF, Morris JG, Wesolowsky G (1989) Facilities location - Models and methods. North-Holland, Amsterdam
35. Melzak ZA (1961) On the problem of Steiner. Canad Math Bull 4:143–148

36. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization. Prentice-Hall, Englewood Cliffs
37. Robins G, Salowe JS (1995) Low-degree minimum spanning trees. Discrete Comput Geom 14:151–165
38. Rubinstein JH, Weng JF (1997) Compression theorems and Steiner ratios on spheres. J Combin Optim 1:67–78
39. Setubal J, Meidanis J (1997) Introduction to computational molecular biology. PWS, Boston, MA
40. Smith JM (1985) Generalized Steiner network problems in engineering design. In: Design Optimization. pp 119–161
41. Wald JA, Colbourn CJ (1983) Steiner trees, partial 2-trees, and minimum IFI networks. Networks 13:159–167
42. Winter P (1985) An algorithm for the Steiner problem in the Euclidean plane. Networks 15:323–345
43. Winter P (1986) Generalized Steiner problem in series-parallel networks. J Algorithms 7:549–566
44. Winter P (1987) Steiner problems in networks: A survey. Networks 17:129–167

Network Location: Covering Problems

TIMOTHY J. LOWE
University Iowa, Iowa City, USA

MSC2000: 90C35, 90B10, 90B80

Article Outline

Keywords
See also
References

Keywords

Maximum coverage location problem; Uncapacitated facility location problem

The *covering problem on a network* involves the decision problem of determining the location of one or more 'facilities' or 'centers' to provide service to several clients located at known points on the network. To provide service to a client, a facility must be located 'close enough' to the client. In a more general version of the problem, there may be fixed costs associated with locating facilities, and also there may be penalty costs associated with not serving clients (not locating a facility close enough to the client). In this situation, to minimize total cost there is a trade-off between establishing facilities and not serving clients.

Let $N(V, A)$ be a connected undirected network with node set $V = \{v_1, \ldots, v_m\}$ and arc set A. Each arc $a \in A$ has a given length $l_a \geq 0$. If we consider $a = [v_i, v_j]$ as a line segment with length l_a, then any point on a can be defined by its distance from v_i (or from v_j). We denote by $d(x, y)$ the shortest path distance on the arcs of N between the points x and y on N. If X is a set of points on N, then $D(X, y)$ denotes the distance between y and a point in X which is closest to y.

Without loss of generality, we assume that the clients are located at the nodes of N. Furthermore, we assume that each node $v_i \in V$ is the site of exactly one client. To specify the notion of 'coverage', for $i = 1, \ldots, m$, let the 'covering radius' $r_i \geq 0$ be associated with v_i. In order for client i to be covered, we require that at least one facility x be located so that $d(x, v_i) \leq r_i$. Equivalently, if X is a set of located facilities, $D(X, v_i) \leq r_i$.

In our version of the problem, we assume that facilities can only be located at members of a subset V' of V, where $V' = \{v_1, \ldots, v_n\}$, $n \leq m$. Since both clients and facilities are located at nodes of N, we can define an $m \times n$ (0–1) covering matrix A, where $a_{ij} = 1$ if and only if $d(v_j, v_i) \leq r_i$. Thus if $a_{ij} = 1$ and if there is a facility located at v_j, then client i is covered. Let $c_j > 0$ be the cost of locating a facility at node v_j, $j = 1, \ldots, n$, and let $b_i > 0$ be the penalty cost associated with not covering client i. Finally, let z_i, $i = 1, \ldots, m$ be a (0–1) variable. Also, let x_j, $j = 1, \ldots, n$ be a (0–1) variable. Then with the above, we can formulate the covering problem as the following (0–1) integer programming problem:

$$\text{(P)} \quad \min \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} b_i z_i$$

subject to:

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j + z_i \geq 1, & i = 1, \ldots, m, \\ x_j \in \{0, 1\}, & j = 1, \ldots, n, \\ z_i \in \{0, 1\}, & i = 1, \ldots, m. \end{cases}$$

In an optimal solution to problem (P), note that $z_i = 1$ only when client i is not covered. That is, only when $x_j = 0$ for every j where $d(v_j, v_i) \leq r_i$.

For some versions of the problem there is a 'budget constraint' imposed on the facilities. This constraint usually takes the form:

$$\sum_{j=1}^{n} c_j x_j \leq M. \tag{1}$$

In the special case where each $c_j = 1$, the budget constraint simply limits the number of facilities that can be located (to a total of M). This problem, when each $b_i = 1$ and the cost of locating facilities is eliminated from the objective function, is often called the *maximum coverage location problem*, since when the objective function is minimized, a minimum number of clients are not covered and so a maximum number of clients are covered. Henceforth, we will only consider versions of problem (P) that do not include (1).

Several applications of problem (P) and its variants have been reported in the literature (see [11] for a partial list of applications). Besides direct applications of the problem, in [9] it is shown that the *p-center problem on a network* (the problem of locating no more than p facilities to minimize the maximum distance between any client and its closest facility) can be solved by solving a sequence of covering problems. As further evidence of its applicability, in [7] it is shown that the *uncapacitated facility location problem* can be formulated as a covering problem. Thus, the covering problem is one of the very fundamental network location problems.

On a general network, the covering problem is known to be *NP*-hard. Heuristics for general (0–1) covering problems have been well-studied ([3,4,5,8,10]).

A popular heuristic is the greedy heuristic which works as follows. (For notational convenience, in the following discussion of the heuristic we assume that each client must be covered. Thus the z_i variables can be removed from (P). If this is not the case, modifications to the heuristic are obvious.) At each iteration t, let A^t be the submatrix of A that consists of rows and columns of A that have not been removed thus far. For each column j of A^t compute $r_j^t = c_j/I_j^t$, where I_j^t is the number of nonzero entries in column j of A^t. Set x_{j^*} to 1 where j^* is an index j for which r_j^t is minimum.

Remove j^* from A^t as well as every row i where $a_{ij^*} = 1$. The resulting matrix is A^{t+1} and the heuristic continues until there are no nonzero entries remaining in the matrix.

Thus the heuristic chooses the column of A^t for which the corresponding facility covers uncovered clients at 'least cost per coverage'. The procedure continues until all clients are covered.

There is a performance guarantee of the greedy heuristic that is independent of the cost coefficients, but does depend upon the entries in the matrix A. It can be shown that the ratio of the objective function value derived via the greedy approach to the optimal objective function value will not exceed $H(d) = \sum_{k=1}^{d} 1/k$, where d is the maximum number of nonzero entries in any column of A.

When the underlying network, $N(V, A)$, is a tree, T (a connected undirected network without cycles), problem (P) can be solved in polynomial time. A. Kolen and A. Tamir [7] have shown that in the case of a tree network, the rows and columns of A can be permuted (in polynomial time) so that the matrix A is in *standard greedy form* (does not contain the submatrix $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$). They then show that any covering problem (P) where A is in standard greedy form can be solved in $O(mn)$. See [1] for the existence of covering problems where A is standard greedy, but where the problem cannot be derived from a tree network location problem. Thus, the Kolen–Tamir approach may be applicable to a broader class of covering problems on networks.

In the special case of (P) on a tree, where each $c_j = 1$ and every client must be covered (b_i is sufficiently large to prohibit noncoverage), even more efficient solution methods are possible (see [2,6], and [12]). With each $c_j = 1$, problem (P) reduces to the problem of minimizing the number of facilities located subject to the condition that every client is covered.

The procedure of B.C. Tansel, R.L. Francis, T.J. Lowe, and M.L. Chen [12] involves a physical 'string model'. Their approach does not involve direct use of the formulation (P) and facilities can be located anywhere (on nodes or arcs) on the tree T. The procedure begins by inscribing straight-line segments on a planar surface such that each segment represents an arc of T. Next, a string of length r_i is fastened to each node v_i of the inscribed tree.

The procedure involves pulling the strings toward the interior of the tree by first considering strings fas-

tened to nodes at the tips of the tree. Facilities are located at points on the tree where the ends of tight strings reach. Once a facility is located each remaining string that reaches the facility is removed from the model, since the corresponding client is covered by that facility.

See also

- ▶ Auction Algorithms
- ▶ Combinatorial Optimization Algorithms in Resource Allocation Problems
- ▶ Communication Network Assignment Problem
- ▶ Competitive Facility Location
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Facility Location with Externalities
- ▶ Facility Location Problems with Spatial Interaction
- ▶ Facility Location with Staircase Costs
- ▶ Generalized Networks
- ▶ Global Optimization in Weber's Problem with Attraction and Repulsion
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ MINLP: Application in Facility Location-Allocation
- ▶ Multifacility and Restricted Location Problems
- ▶ Network Design Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Piecewise Linear Network Flow Problems
- ▶ Production-Distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Shortest Path Tree Algorithms
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Transportation and Location Problems
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Broin M, Lowe TJ (1986) A dynamic programming algorithm for covering problems with (greedy) totally balanced constraint matrices. SIAM J Alg Discrete Meth 7:348–357
2. Chandrasekaran R, Tamir A (1982) Polynomially bounded algorithms for locating p-centers on a tree. Math Program 22:304–315
3. Chvátal V (1979) A greedy heuristic for the set-covering problem. Math Oper Res 4:233–235
4. Hochbaum DS (1997) Approximating covering and packing problems: Set cover, vertex cover, independent set and related problems. In: Hochbaum DS (ed) Approximation Algorithms for NP-Hard Problems. PWS, Boston
5. Johnson DS (1974) Approximation algorithms for combinatorial problems. J Comput Syst Sci 9:256–278
6. Kariv O, Hakimi SL (1979) An algorithmic approach to network location problems. Part 1. The p-centers. SIAM J Appl Math 37:513–538
7. Kolen A, Tamir A (1990) Covering problems. In: Discrete Location Theory. Wiley, New York
8. Lovász L (1975) On the ratio of optimal integral and fractional covers. Discret Math 13:383–390
9. Minieka E (1970) The m-center problem. SIAM Rev 12:138–139
10. Nemhauser G, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
11. Schilling DA, Jayaraman V, Barkhi R (1993) A review of covering problems in facility location. Location Sci 1:25–55
12. Tansel BC, Francis RL, Lowe TJ, Chen ML (1982) Duality and distance constraints for the nonlinear *p*-center problem and covering problem on a tree network. Oper Res 30:725–744

Neural Networks for Combinatorial Optimization

THEODORE B. TRAFALIS, SUAT KASAP
School of Industrial Engineering,
University Oklahoma, Norman, USA

MSC2000: 90C27, 90C30

Article Outline

Keywords
Neural Networks
Neural Networks and Combinatorial Optimization Problems

Combinatorial Optimization Problems
Quadratic Assignment Problem
Graph Partitioning Problem
Traveling Salesman Problem
Conclusions
See also
References

Keywords

Combinatorial optimization; Neural networks; Nonlinear programming; Global optimization

Most of the engineering design problems and applications can be formulated as a nonlinear programming problem in which the objective function to be optimized is nonlinear and has many local optima in its feasible region. It is desirable to find a local optimum that corresponds to a near global optimum. The problem of finding a global minimum or maximum is known as the *global optimization problem*. The main difficulties in finding a global optimum are that there are no operationally useful optimality conditions for identifying whether a point is indeed a global optimum, except in cases of special structured problems [16].

In this article, one class of global optimization problems, specifically combinatorial optimization problems will be mainly discussed. The combinatorial optimization problems deal with problems of maximizing or minimizing an objective function subject to inequality and/or equality constraints over a set of combinatorial alternatives. Finding optimal solutions to such problems where the decision variables are combinatorial or discrete is known as *combinatorial optimization problems*. A naive way to solve combinatorial optimization problems is to list all of the feasible solutions of a given problem, then evaluate its objective function, and choose the best one as an optimum solution. However, even though it is possible in principle to solve the problem in this way, in practice it is not, because of the large number of possible feasible solutions to any problem of a reasonable size. Most of the combinatorial optimization problems are *NP*-complete [34]. Because of the combinatorial structure of these problems the time needed to solve them grows exponentially with the size of the problem. For *NP*-complete problems, there is no algorithm that provides an exact solution to the problem in polynomial time.

Over the last three decades, the combinatorial optimization problem is one of the most challenging problems that has received considerable attention. The traditional solution methodologies for combinatorial optimization problems can be categorized into three groups as exact, heuristic, and approximation methods [50]. Exact methods guarantee to obtain an optimum solution, but the computational time for obtaining an optimum solution is usually an exponential function of the number of variables. The simplex method and branch and bound methods are some examples of exact methods. The heuristic methods are problem-specific methods based on success without formal analysis of performance. Simulated annealing (SA), tabu search, and genetic algorithms are some examples of heuristic methods [45]. On the other hand, the approximation methods generate feasible solutions that are near optimal solutions. Neural networks based methods are perceived as approximation methods.

This article reviews the use of NNs for combinatorial optimization problems and provides a survey of most of the NN approaches that have been applied to combinatorial optimization problems. In Section 2, basic definitions, classifications and applications of NNs are introduced. In Section 3, the mathematical basis of problem formulation for NNs based on an energy function is explained. In Section 4, various combinatorial optimization problems studied on NNs are presented. Finally, Section 5 is the conclusion.

Neural Networks

Neural Network (NN) models are algorithms for intellectual tasks such as learning and optimization that are based on the concept of how the human brain works. A NN model is composed of a large number of processing elements called *neurons*. Each neuron is connected to other neurons by links, each with an associated weight. Neurons without links toward them are called *input neurons* and with no link leaving away from them are called *output neurons*. The neurons are represented by state variables. State variables are functions of the weighted-sum of input variables and other state variables. Each neuron performs a simple transformation at the same time in a parallel-distributed manner. The input-output relation of the transformation in a neuron is characterized by an *activation function*.

Some examples of activation functions are the threshold function, linear, and sigmoidal functions. The combination of input neurons, output neurons, and links between neurons with associated weights constitute the architecture of the NN.

Mathematically, a NN model is a directed graph with the following properties [12]:

1) Each neuron (node) is associated with a state variable S_i and an activation threshold (bias) v_i.
2) Each link (edge) between two neurons i and j associates with it a weight w_{ij}.
3) Each neuron (node) defines a transfer function $f_i(S_i, w_{ij}, v_i)$, which determines the state of the neuron.

The classification of NNs can be done in different ways. According to the nature of activation, they can be categorized as deterministic and stochastic NNs. *Deterministic NNs* activate their states by using deterministic activation functions such as the threshold, linear, or sigmoidal functions. Most of the existing NN models are deterministic. *Stochastic NNs* activate their states according to a probability distribution. A typical example of stochastic NNs is the Boltzmann Machine (BM). According to the nature of the connectivity, NNs can be categorized as feed-forward and recurrent NNs. If a directed graph has no closed paths, then it is called a *feed-forward NN*. A typical example of the feed-forward NN is the popular multilayer perceptron. Conversely, if a directed graph has closed paths, then it is called a *recurrent NN*. A typical example of the recurrent NN is the Hopfield NN ([9,12]).

Modern era of NNs is said to have begun with the introductory work of W.S. McCullough and W. Pitts [28]. They have proposed a general theory of information processing based on networks of neurons. Each one of these neurons can only take the output values 1 or 0 by representing the active and resting states of neurons respectively. NNs have come a long way from the early days of McCullough and Pitts. NNs have established themselves as an interdisciplinary subject with rich connections in the neuroscience, psychology, physical sciences, mathematics and engineering.

NNs have very close ties with optimization and the ties are manifested mainly into two aspects. On one aspect, a lot of learning algorithms have been developed based on optimization techniques to train NNs to perform modeling tasks ([2,9,12,37,39]). On the other aspect, NNs have been developed for solving optimization problems ([4,6,12,26,36,40,44,50,51]). Because of the inherent nature of parallel and distributed information processing in NNs, they are promising computational models for solving large scale optimization problems in real-time ([3,5,21,22,23,49]).

Neural Networks and Combinatorial Optimization Problems

NNs have been proposed as a model of computation to solve combinatorial optimization problems. To solve combinatorial optimization problems using NNs requires a mapping of the problem onto the NNs in such a way that one can identify a solution from outputs of the neurons. In other words, to solve the combinatorial optimization problems using NNs, the key is to map the problem into the architecture of the NN for which the stable state represents the solution of the combinatorial optimization problem.

Combinatorial optimization problems can be solved using NNs by following two approaches.

1) By formulating a combinatorial optimization problem in terms of minimizing an energy function of either discrete or continuous variables ([1,3,4,5,8,13,15,35,36,40]).
2) By designing competition based NNs in which neurons are allowed to compete to become active under certain conditions ([7,9]).

These two approaches suggest that NNs are an alternative for solving combinatorial optimization problems as compared to other optimization techniques. Motivations for using NNs include the improvement in the speed of the operation through massively parallel computation and possible hardware design advantages. One of the main advantages of NN approaches to classical optimization approaches is the inherently parallel and distributed nature of the dynamic solution procedure. Therefore, NNs are capable to solve large scale optimization problems in real time ([3,5,21,22,23,49]).

NNs have been used to solve many combinatorial optimization problems since the pioneering work of J.J. Hopfield and D.W. Tank [15]. They formulated the traveling salesman problem (TSP) on a highly interconnected NN and made exploratory numerical studies on a 10-city and 30-city TSP. They showed that an energy function can be defined for an analog Hopfield NN and the NN always converges to a local min-

imum of this function. To find the (near) global minimum of the energy function, the Boltzmann Machine (BM) was proposed [2]. The BM was the first attempt to combine SA and NN architectures to solve combinatorial optimization problems. However, BM is designed for discrete variables with the disadvantage of being so slow. Another approach by combining SA and NNs was proposed in [24]. In this study, a noise term has been added to the neural dynamics. The intensity of that noise term has been shown to depend on the state of the neuron as well as on a temperature parameter T. By selecting an appropriate temperature schedule $T(t)$, the resulting NN converged to a near global minimum of the NN energy function that is in the neighborhood of the global minimum of the combinatorial optimization problem. The Hopfield NN was extended to handle inequality constraints and simplified its convergence characteristics by the eigenvalue analysis [1]. The mathematical basis of the behavior of the Hopfield NN by means of an idealization of the Hopfield network was given in [47]. This study helped to give a better understanding of the relationship between NNs and the combinatorial optimization. The analog neural solution of the combinatorial optimization problem was considered [48]. The solution method was analyzed based on the Lagrange multiplier for the continuous relaxation problem of 0-1 integer programming. The theory and methodology of the deterministic NNs for the combinatorial optimization were presented in [50]. This study was extended to the convex programming [51]. The use of NN methods for the combinatorial optimization problems was reviewed ([13,26,44]). A systematic approach to design competition based NNs for the combinatorial optimization was presented in [7]. The competition based NNs were studied in detail [9].

The procedure of NN approaches to the combinatorial optimization mostly begins with the formulation of an energy function. Ideally, the minimum of this energy function corresponds to the optimal solution of the combinatorial optimization problem. Most of the existing NN approaches to combinatorial optimization problems formulate an energy function by incorporating an objective function and constraints through functional transformation and numerical weighting. A functional transformation is usually used to convert constraints to a penalty function to penalize the violations of constraints. Numerical weighting is often used to balance constraint satisfaction and objective minimization. In the NN formulation for combinatorial optimization problem, constraint violations enter to the energy function in an explicit way. In general, such an energy function will have the following form:

$$E = \sum_i A_i(\text{constraint violation})_i + \text{cost} \tag{1}$$

where $A_i > 0$ and cost is an objective function that is independent from the constraint violations. By minimizing the energy function E, we attempt to minimize the cost while at the same time minimize the constraint violations.

The second step in designing NNs for combinatorial optimization problems is to derive a dynamic equation (also known as state equation or motion equation). The dynamic equation of the NN prescribes the motion of the activation states of the NN. A properly derived dynamic equation can ensure that the state of the NN reaches equilibrium and the equilibrium-state of the NN satisfies the constraints and optimizes the objective function of the problem. Currently, the dynamic equations of most NNs for optimization problems are derived by letting the time derivative of a state vector to be directly proportional to the negative gradient of an energy function. The dynamic equations and energy functions for a wide variety of combinatorial optimization problems were studied [28]. The last step is to determine the architecture of the NN in terms of neurons and connections based on the derived dynamic equation in such a way that one can identify a solution from the outputs of the neurons.

The success of optimization using NNs lies in the formulation of an energy function, based on the objective function and constraints of the given optimization problem, and the derivation of a dynamic equation of NNs, based on the formulated energy function.

The NN approaches to optimization problems have been started with [14]. He has pioneered an approach for solving minimization problems by utilizing the collective computational capabilities of NNs. He has mapped the objective function and the problem constraints onto a quadratic energy function of the neural states that presents the energy of the system of neurons.

The energy function had the following form:

$$E(S) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}S_iS_j - \sum_{i=1}^{N} S_iI_i \ , \tag{2}$$

where S_i represents the output state of neurons. C_{ij} represents the strength of the link between neurons i and j. I_i represents the input bias; the input of neurons is denoted as x_i. The motion equations of neurons is defined by

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = -x_i + \sum_{j\neq i}^{N} C_{ij}S_i + I_i \ , \quad \text{where} \quad S_i = f_i(x_i) \ . \tag{3}$$

Hopfield showed that for a NN with symmetric connections and nonnegative elements on the diagonal of the C matrix, the NN will always converge by performing the gradient descent to a stable state, which is a local minimum of the energy function. Many difficult optimization problems can be formulated as the minimization of a quadratic form and thus can be mapped onto the NN model. Hopfield and Tank [15] extended the discrete model to an analog model where $0 \leq S_i \leq 1$. The modified energy function is:

$$E(S) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}S_iS_j - \sum_{i=1}^{N} S_iI_i + \sum_{i=1}^{N}\frac{1}{R_i}\int_0^{v_i} g_i^{-1}(S)\,\mathrm{d}S \ , \tag{4}$$

where R_i is the input resistance to unit i, and g_i is the activation function. The dynamics of the neuron update is defined by

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = -x_i + \sum_{j\neq i}^{N} C_{ij}S_i + I_i \ , \quad \text{where} \quad S_i = \frac{1}{2}(1+\tanh(x_i)) \ . \tag{5}$$

This model is synchronous, continuous and deterministic.

NNs to minimize the energy function are composed of either discrete or continuous neurons. The possibilities of using different types of NNs such as discrete-state and discrete-time, continuous-state and discrete-time, and continuous-state and continuous-time networks to solve combinatorial optimization problems were investigated [29]. One usually prefers to use continuous neurons rather than discrete ones for two reasons. First, a continuous network tends to avoid oscillations between stable states. Second, solutions are much better than those provided by discrete NNs, because the valley of energy landscapes are wider and neuron outputs are not restricted to corners of the hypercube during convergence. NNs do not guarantee globally optimal solutions but they compute locally optimal solutions.

In the last two decades, spin glass theories of statistical physics have been extensively studied because of their applications to other areas like optimization and neural networks ([10,30,46]). A similarity between an NN of the type proposed in [28] and a system of elementary magnetic spins was pointed in [25]. These ideas were developed further [14] by studying how such a NN or a spin system can store and retrieve information. The idea of an energy function was used to formulate a new way of understanding the computation performed by NNs.

To solve combinatorial optimization problems using NNs and the mean field theory (MFT) of the statistical physics ([3,4,5,17,18,20,31,40]) was introduced by C. Peterson and J.R. Anderson [38]. This was the first detailed attempt to use the MFT and NNs for solving combinatorial optimization problems after a brief description by Hopfield and Tank [15]. The problem was mapped onto the NN such that a neuron being 'on' corresponded to a certain decision and then relaxed the system using the MFT in order to escape from local minima. In this study, the problem was mapped onto an *Ising glass model* ([10,11,31,40]). Another method for finding approximate solutions to combinatorial optimization problems within NNs and the MFT concept was presented [40]. The problem was mapped onto a *Potts glass model* ([19,31,40,46,52]) in this study. A mean field annealing (MFA) was described based on the MFT and SA. Similar studies based on the MFT of statistical physics to solve combinatorial optimization problems have been studied ([4,5,49]). A general framework for the MFA algorithm was derived and its relationship with to Hopfield NNs was shown [4]. A NN mapping and the MFT solution method for finding good solutions to combinatorial optimization prob-

lems containing inequality constraints like the knapsack problem were developed ([32,33]).

A method to solve combinatorial optimization problems was proposed ([17,18]) based on 'two-layer random field model' using the technique of the MFA. This method determined the appropriate values of the weights of two kinds of terms in objective function; a cost term that should be minimized and a constraint term that expresses constraints imposed on solutions. A parallel MFT method and a new temperature scheduling method named as maximum entropy cooling schedule were proposed [43]. The relationship between the MFT NN model and the continuous-time Hopfield NN was analyzed [20] by using the theory of dynamical systems. This study showed that the asynchronous MFT model was equivalent to the continuous-time Hopfield NN on the nature of the fixed points and hence guaranteed theoretically usage of the MFT model for solving combinatorial optimization problems in place of the continuous-time Hopfield NN.

Combinatorial Optimization Problems

In this section, we list some of the combinatorial optimization problems that have been studied to be solved using NNs. Most of these problems are *NP*-complete [34]. The mapping of combinatorial optimization problems onto NNs by focusing graph problems such as graph K-partitioning, maximum graph matching, and maximum clique problems was studied [13]. The use of NNs for combinatorial optimization problems was reviewed [26]. A number of interesting combinatorial optimization problems such as graph partitioning, traveling salesman problem, vertex cover, maximum clique, maximum independent set, number partitioning, maximum matching, set cover, and graph coloring were studied to show how to map these problems into NNs [44]. A NN mapping and the MFT solution method for finding good solutions to combinatorial optimization problems containing inequality constraints like the knapsack problem were developed [32]. The MFT approach to the knapsack problem was extended to multiple knapsacks and generalized assignment problems [33]. The following three problems have been studied mostly by several researchers using different approaches.

Quadratic Assignment Problem

The quadratic assignment problem (QAP) represents a large class of combinatorial optimization problems arising in a variety of planning and designing contexts. The QAP seeks to minimize a quadratic cost function for assignment of a number of objects to positions that can be mathematically described as follows [50]:

$$\begin{cases} \min & f(x) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} c_{ijkl}x_{ij}x_{kl} \\ \text{s.t.} & \sum_{i=1}^{N} x_{ij} = 1, \quad j = 1,\ldots,N, \\ & \sum_{j=1}^{N} x_{ij} = 1, \quad i = 1,\ldots,N, \\ & x_{ij} \in \{0,1\}, \quad i,j = 1,\ldots,N, \end{cases} \tag{6}$$

where c_{ijkl} denotes the cost associated with assigning object i to position j and object k to position l for $i, j, k, l = 1, \ldots, N$.

An energy function for this problem can be written as follows [8]:

$$E(S) = \frac{A}{2}\sum_{i=1}^{n}\sum_{j\neq i}\sum_{k=1}^{m}\sum_{l\neq k} c_{ijkl}S_{ik}S_{jl} + \frac{B}{2}\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{l\neq k} S_{ik}S_{il} + \frac{C}{2}\sum_{k=1}^{m}\left(1-\sum_{i=1}^{n} S_{ik}\right)^2. \tag{7}$$

The first term is the just cost function. The second term specifies the constraint that at most one position can be assigned to each object and the third term specifies the constraint that every object must be assigned by exactly one assignment. This term prevents the solution of no assignments. The QAP was studied in ([6,7,8,50]).

Graph Partitioning Problem

The graph partitioning (GP) problem can be defined as follows: Given a set of N nodes with a given connectivity, partition them into K sets each with N/K nodes such that the net connectivity (cut-size) is minimal between each set. If $K = 2$, then this problem is known as the *graph bipartitioning problem*.

An energy function for this problem can be written as follows [42]: For each vertex i, a binary unit $S_i = +1$ or -1 is assigned, and for each pair of vertices S_i, S_j, $i \neq$

j, a value $T_{ij} = 1$ if they are connected, and $T_{ij} = 0$ if they are not connected is assigned:

$$E(S) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}S_iS_j + \frac{\alpha}{2}\left(\left(\sum_{i=1}^{N} S_i\right)^2 - \sum_{i=1}^{N} S_i^2\right), \tag{8}$$

where α is an imbalance parameter. The first term corresponds to the cost function and the second term corresponds to the constraints of the problem that guarantee equal partition. $T_{ij}S_iS_j$ is 0 whenever vertices i and j are not connected at all, positive whenever connected vertices i and j are in the same partition and negative when they are in separate partition. The GP was studied in ([5,6,38,40,41,42,44,49]). The graph bipartitioning was studied in ([4,6,36,40,41,42]). The graph K-partitioning was studied in ([13,44]).

Traveling Salesman Problem

The traveling salesman problem (TSP) can be defined as follows: Given a list of N cities and the distances, d_{ij}, among them, find the shortest possible tour through a set of N cities, visiting each one exactly once. Note that the TSP is a special case of the GP problem where $K = N$. For the n-city TSP, the number of possible tours are $n!$. However, a tour describes an order in which cities are visited. For the n-city TSP, there are $2n$ tours of equal path-length. Therefore, there are $2!/2n$ distinct paths for closed TSP tours. The total of $N = n^2$ neurons required to map the n-city TSP to NNs. To enable the N neurons in the TSP network, to compute a solution to the problem, the network must be described by an energy function in which the lowest energy state correspond to the best path. Hopfield and Tank [15] used the following energy function to solve the TSP problem. The output S_{ij} indicates whether city i is assigned to position j in the tour or not.

$$E(S) = \frac{A}{2}\sum_{X}^{N}\sum_{i}^{N}\sum_{j\neq i}^{N} S_{Xi}S_{Xj} + \frac{B}{2}\sum_{i}^{N}\sum_{X}^{N}\sum_{X\neq Y}^{N} S_{Xi}S_{Yi} + \frac{C}{2}\left(\sum_{X}^{N}\sum_{i}^{N} S_{Xi} - N\right)^2 + \frac{D}{2}\sum_{X}^{N}\sum_{X\neq Y}^{N}\sum_{i}^{N} d_{XY}S_{Xi}(S_{Y,i+1} + S_{Y,i-1}). \tag{9}$$

The first three terms characterize the general problem constraints. The first triple sum is zero if and only if each city row X contains no more than '1', (the rest of the entries being zero). The second triple sum is zero if and only if each 'position in tour' column contains no more than '1', (the rest of the entries being zero). The third triple sum is zero if and only if there are n entries of '1' in the entire matrix. The last triple sum is the cost term, the length of the path corresponding to a given tour. The TSP was studied in ([6,15,17,18,26,27,35,40,41,42,43,50]).

In addition to these problems, the following problems have been also studied:

- The Airline Crew Scheduling [21].
- The Constraint Satisfaction Problem [26].
- The Generalized Assignment Problem ([7,33]).
- The Graph Coloring Problem [44].
- The Job Scheduling [7].
- The Knapsack Problem ([1,7,32,33,36,41,42]).
- The Maximum Clique Problem ([13,22,23,44]).
- The Maximum Independent Set Problem [44].
- The Maximum Matching Problem ([13,44]).
- The Morphism Problems [13].
- The Number Partitioning [44].
- The Satellite Broadcasting Scheduling [3].
- The School Scheduling ([41,42]).
- The Set Cover Problem [44].
- The Vertex Cover Problem [44].

Conclusions

The use of NNs for the combinatorial optimization problems and the NN approaches that have been applied to combinatorial optimization problems were reviewed. These approaches suggest that NNs are an alternative for solving combinatorial optimization problems as compared to other optimization techniques. Motivations for using NNs include the improvement in the speed of the operation through massively parallel computation, and possible hardware design advan-

tages. One of the main advantages of NN approaches to classical optimization approaches is the inherently parallel and distributed nature of the dynamic solution procedure. Therefore, NNs are capable to solve large scale optimization problems in real time. This is essential in many engineering design, control, and optimization problems. Because of the inherent nature of parallel and distributed information processing in NNs, they are promising computational models for solving large scale optimization problems in real-time.

See also

- ► Neuro-dynamic Programming
- ► Replicator Dynamics in Combinatorial Optimization
- ► Unconstrained Optimization in Neural Network Training

References

1. Abe S, Kawakami J, Hirasawa K (1992) Solving inequality constrained combinatorial optimization problems by the Hopfield neural networks. Neural Networks 5:663–670
2. Ackley DH, Hinton GE, Sejnowski TJ (1985) A learning algorithm for Boltzman machines. Cognitive Sci 9:147–169
3. Ansari N, Hou ESH, Yu Y (1995) A new method to optimize the satellite broadcasting schedules using the mean field annealing of a Hopfield neural network. IEEE Trans Neural Networks 6(2):470–482
4. Bilbro G, Mann R, Miller TK, Snyder WE, VanDenBout DE, White M (1988) Optimization by mean field annealing. In: Touretzky DS (eds) Proc. Annual Conf. Advances in Neural Information Processing Systems. Morgan Kaufmann, San Mateo, pp 91–98
5. Bilbro GL, Snyder WE, Garnier SJ, Gault JW (1992) Mean field annealing formalism for constructing GNC-like algorithms. IEEE Trans Neural Networks 3(1):131–138
6. Cichocki A, Unbehauen R (1993) Neural networks for optimization and signal processing. Wiley, New York
7. Fang L, Li T (1990) Design of competition-based neural networks for combinatorial optimization. Internat J Neural Systems 1(3):221–235
8. Fang L, Wilson WH, Li T (1990) Mean field annealing neural net for quadratic assignment. In: Internat. Neural Network Conf., Paris, July 9–13. Kluwer, Dordrecht, pp 282–286
9. Fausett L (1994) Fundamentals of neural networks: Architectures, algorithms, and applications. Prentice-Hall, Englewood Cliffs
10. Fischer KH, Hertz JA (1991) Spin glasses. Cambridge Univ Press, Cambridge
11. Glauber RJ (1963) Time-dependent statistics of the Ising model. J Math Phys 4(2):294–307
12. Haykin S (1994) Neural networks: A comprehensive foundation. MacMillan College Publ., New York
13. Herault L, Niez JJ (1991) Neural networks and combinatorial optimization: A study of NP-complete graph problems. In: Gelenbe E (eds) Neural Networks: Advances and Applications. Elsevier, Amsterdam, pp 165–213
14. Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. In: Proc Natl Acad Sci USA, Biophysics, pp 2554–2558
15. Hopfield JJ, Tank DW (1985) Neural computation of decisions in optimization problems. Biol Cybern 52:141–152
16. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
17. Igarashi H (1993) A solution to combinatorial optimization problems using a two-layer random field model: Mean-field approximation. In: World Congress on Neural Networks, Portland, July 11-15. Lawrence Erlbaum Ass., Philadelphia, pp 283–286
18. Igarashi H (1994) A solution for combinatorial optimization problems using a two-layer random field model: Mean-field approximation. Syst Comput Jpn 25(8):61–71
19. Kanter I, Sompolinsky H (1987) Graph optimization problems and the Potts glass. J Phys A: Math Gen 20:L673–L679
20. Kurita N, Funahashi K-I (1996) On the Hopfield neural networks and mean field theory. Neural Networks 9(9):1531–1540
21. Lagerholm M, Peterson C, Söderberg B (1997) Airline crew scheduling with Potts neurons. Neural Comput 9:1589–1599
22. LeeK-C, Takefuji Y (1996) Maximum clique problems: Part 1. In: Takefuji Y, Wang J (eds) Neural Computing for Optimization and Combinatorics. World Sci., Singapore, pp 31–61
23. LeeK-C, Takefuji Y (1996) Maximum clique problems: Part 2. In: Takefuji Y, Wang J (eds) Neural Computing for Optimization and Combinatorics. World Sci., Singapore, pp 63–77
24. Levy BC, Adams MB (1987) Global optimization with stochastic neural networks. In: Neural Networks for Optimization and Signal Processing, Proc. First Internat. Conf. on Neural Networks, San Diego, pp 681–690
25. Little WA (1974) The existence of persistent states in the brain. Math Biosci 19:101–120
26. LooiC-K (1992) Neural network methods in combinatorial optimization. Comput Oper Res 19(3–4):191–208
27. Matsuda S (1996) Set-theoretic comparison of mapping of combinatorial optimization problems to Hopfield neural networks. Syst Comput Jpn 27(6):45–59
28. McCullough WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophysics 5:115–133
29. Melamed II (1994) Neural networks and combinatorial optimization. Automation and Remote Control 55(11):1553–1584

30. Mezard M, Parisi G, Virasoro MA (1987) Spin glass theory and beyond. World Sci., Singapore
31. Muller B, Reinhardt J (1990) Neural networks: An introduction. Springer, Berlin
32. Ohlsson M, Peterson C, Söderberg B (1993) Neural networks for optimization problems with inequality constraints: The knapsack problem. Neural Computation 5:331–339
33. Ohlsson M, Pi H (1997) A study of the mean field approach to knapsack problems. Neural Networks 10(2):263–271
34. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: Algorithms and complexity. Prentice-Hall, Englewood Cliffs
35. Peterson C (1990) Parallel distributed approaches to combinatorial optimization: Benchmark studies on traveling salesman problem. Neural Computation 2:261–269
36. Peterson C (1993) Solving optimization problems with mean field methods. Phys A 200:570–580
37. Peterson C, Anderson JR (1987) A mean field theory learning algorithm for neural networks. Complex Systems 1:995–1019
38. Peterson C, Anderson JR (1988) Neural networks and NP-complete optimization problems; a performance study on the graph bisection problem. Complex Systems 2:59–89
39. Peterson C, Hartman E (1989) Explorations of the mean field theory learning algorithm. Neural Networks 2:475–494
40. Peterson C, Söderberg B (1989) A new method for mapping optimization problems onto neural networks. Internat J Neural Systems 1(1):3–22
41. Peterson C, Söderberg B (1993) Artificial neural networks. In: Reeves CR (eds) Modern heuristic techniques for combinatorial problems. Wiley, New York, pp 197–242
42. Peterson C, Söderberg B (1997) Artificial neural networks. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, New York, pp 177–213
43. Qian F, Hirata H (1994) A parallel computation based on mean field theory for combinatorial optimization and Boltzman machines. Syst Comput Jpn 25(12):86–97
44. Ramanujam J, Sadayappan P (1995) Mapping combinatorial optimization problems onto neural networks. Inform Sci 82:239–255
45. Reeves CR (1993) Modern heuristic techniques for combinatorial problems. Wiley, New York
46. Reichl LE (1980) A modern course in statistical physics. Univ Texas Press, Austin
47. Uesaka Y (1993) Mathematical basis of neural networks for combinatorial optimization problems. Optoelectronics - Devices and Technologies 8(1):1–9
48. Urahama K (1995) Mathematical programming formulation for neural combinatorial optimization algorithms. Electron Comm Jpn 78(9):67–75
49. VanDenBout DE, Miller TK (1990) Graph partitioning using annealed networks. IEEE Trans Neural Networks 1(2):192–203
50. Wang J (1994) Deterministic neural networks for combinatorial optimization. In: Omidvar OM (eds) Progress in Neural Networks. Ablex, Stamford, CT, pp 319–340
51. Wang J (1996) Recurrent neural networks for optimization. In: Chen CH (eds) Fuzzy Logic and Neural Network Handbook. McGraw-Hill, New York
52. Wu FY (1982) The Potts model. Rev Modern Phys 54(1):235–268

Neuro-Dynamic Programming

NDP

DIMITRI P. BERTSEKAS
Labor. Information and Decision Systems,
Massachusetts Institute Technol., Cambridge, USA

MSC2000: 90C39

Article Outline

Keywords
Cost Approximations in Dynamic Programming
Approximation Architectures
Simulation and Training
Neuro-Dynamic Programming
See also
References

Keywords

Dynamic programming; Optimization; Reinforcement learning; Simulation; Neural networks

Neuro-dynamic programming (NDP for short) is a relatively new class of dynamic programming methods for control and sequential decision making under uncertainty. These methods have the potential of dealing with problems that for a long time were thought to be intractable due to either a large state space or the lack of an accurate model. The methods discussed combine ideas from the fields of neural networks, artificial intelligence, cognitive science, simulation, and approximation theory. In this article, we delineate the major conceptual issues, we survey a number of recent developments, we describe some computational experience, and we address a number of open questions.

We consider systems where decisions are made in stages. The outcome of each decision is not fully predictable but can be anticipated to some extent before the next decision is made. Each decision results in some immediate cost but also affects the context in which future decisions are to be made and therefore affects the cost incurred in future stages. *Dynamic programming* (DP for short) provides a mathematical formalization of the trade-off between immediate and future costs.

Generally, in DP formulations there is a discrete-time dynamic system whose state evolves according to given transition probabilities that depend on a decision/control u. In particular, if we are in state i and we choose decision u, we move to state j with given probability $p_{ij}(u)$. Simultaneously with this transition, we incur a cost $g(i, u, j)$. In comparing, however, the available decisions u, it is not enough to look at the magnitude of the cost $g(i, u, j)$; we must also take into account how desirable the next state j is. We thus need a way to rank or rate states j. This is done by using the optimal cost (over all remaining stages) starting from state j, which is denoted by $J^*(j)$. These costs can be shown to satisfy some form of *Bellman's equation*

$$J^*(i) = \min_u \mathsf{E}\left\{g(i, u, j) + J^*(j) | i, u\right\} ,$$
$$\text{for all } i ,$$

where j is the state subsequent to i, and $\mathsf{E}\{\cdot|i, u\}$ denotes expected value with respect to j, given i and u. Generally, at each state i, it is optimal to use a control u that attains the minimum above. Thus, decisions are ranked based on the sum of the expected cost of the present period, and the optimal expected cost of all subsequent periods.

The objective of DP is to calculate numerically the optimal cost function J^*. This computation can be done off-line, i. e., before the real system starts operating. An optimal policy, that is, an optimal choice of u for each i, is computed either simultaneously with J^*, or in real time by minimizing in the right-hand side of Bellman's equation. It is well known, however, that for many important problems the computational requirements of DP are overwhelming, mainly because of a very large number of states and controls (Bellman's 'curse of dimensionality'). In such situations a suboptimal solution is required.

Cost Approximations in Dynamic Programming

NDP methods are suboptimal methods that center around the approximate evaluation of the optimal cost function J^*, possibly through the use of neural networks and/or simulation. In particular, we replace the optimal cost $J^*(j)$ with a suitable approximation $\widetilde{J}(j, r)$, where r is a vector of parameters, and we use at state i the (suboptimal) control $\widetilde{\mu}(i)$ that attains the minimum in the (approximate) right-hand side of Bellman's equation

$$\widetilde{\mu}(i) = \arg\min_u \mathsf{E}\left\{g(i, u, j) + \widetilde{J}(j, r) | i, u\right\} .$$

The function $\widetilde{J}$ will be called the *scoring function*, and the value $\widetilde{J}(j, r)$ will be called the *score* of state j. The general form of $\widetilde{J}$ is known and is such that once the parameter vector r is determined, the evaluation of $\widetilde{J}(j, r)$ of any state j is fairly simple.

We note that in some problems the minimization over u of the expression

$$\mathsf{E}\left\{g(i, u, j) + \widetilde{J}(j, r) | i, u\right\}$$

may be too complicated or too time-consuming for making decisions in real-time, even if the scores $\widetilde{J}(j, r)$ are simply calculated. In such problems we may use a related technique, whereby we approximate the expression minimized in Bellman's equation,

$$Q(i, u) = \mathsf{E}\left\{g(i, u, j) + J^*(j) | i, u\right\} ,$$

which is known as the Q-factor corresponding to (i, u). In particular, we replace $Q(i, u)$ with a suitable approximation $\widetilde{Q}(i, u, r)$, where r is a vector of parameters. We then use at state i the (suboptimal) control that minimizes the approximate Q-factor corresponding to i:

$$\widetilde{\mu}(i) = \arg\min_u \widetilde{Q}(i, u, r) .$$

Much of what will be said about approximation of the optimal cost function also applies to approximation of Q-factors. We thus focus primarily on approximation of the optimal cost function J^*.

We are interested in problems with a large number of states and in scoring functions $\widetilde{J}$ that can be described with relatively few numbers (a vector r of small dimension). Scoring functions involving few parameters are called *compact representations*, while the tabular description of J^* are called the *lookup table represen-*

tation. Thus, in a lookup table representation, the values $J^*(j)$ are stored in a table for all states j. In a typical compact representation, only the vector r and the general structure of the scoring function $\widetilde{J}(\cdot, r)$ are stored; the scores $\widetilde{J}(j, r)$ are generated only when needed. For example, $\widetilde{J}(j, r)$ may be the output of some neural network in response to the input j, and r is the associated vector of weights or parameters of the neural network; or $\widetilde{J}(j, r)$ may involve a lower-dimensional description of the state j in terms of its 'significant features', and r is the associated vector of relative weights of the features. Thus determining the scoring function $\widetilde{J}(j, r)$ involves two complementary issues:

1) deciding on the general structure of the function $\widetilde{J}(j, r)$; and
2) calculating the parameter vector r so as to minimize in some sense the error between the functions $J^*(\cdot)$ and $\widetilde{J}(\cdot, r)$.

Approximations of the optimal cost function have been used in the past in a variety of DP contexts. Chess playing programs represent a successful example. A key idea in these programs is to use a position evaluator to rank different chess positions and to select at each turn a move that results in the position with the best rank. The *position evaluator* assigns a numerical value to each position, according to a heuristic formula that includes weights for the various features of the position (material balance, piece mobility, king safety, and other factors). Thus, the position evaluator corresponds to the scoring function $\widetilde{J}(j, r)$ above, while the weights of the features correspond to the parameter vector r. Usually, some general structure of position evaluator is selected (this is largely an art that has evolved over many years, based on experimentation and human knowledge about chess), and the numerical weights are chosen by trial and error or (as in the case of the champion program Deep Thought) by 'training' using a large number of sample grandmaster games.

As the chess program paradigm suggests, intuition about the problem, heuristics, and trial and error are all important ingredients for constructing cost approximations in DP. However, it is important to supplement heuristics and intuition with more systematic techniques that are broadly applicable and retain as much as possible the nonheuristic aspects of DP.

NDP aims to develop a methodological foundation for combining dynamic programming, compact representations, and simulation to provide the basis for a rational approach to complex stochastic decision problems.

Approximation Architectures

An important issue in function approximation is the *selection of architecture*, that is, the choice of a parametric class of functions $\widetilde{J}(\cdot, r)$ or $\widetilde{Q}(\cdot, \cdot, r)$ that suits the problem at hand. One possibility is to use a neural network architecture of some type. We should emphasize here that in this article we use the term 'neural network' in a very broad sense, essentially as a synonym to 'approximating architecture'. In particular, we do not restrict ourselves to the classical multilayer perceptron structure with sigmoidal nonlinearities. Any type of universal approximator of nonlinear mappings could be used in our context. The nature of the approximating structure is left open in our discussion, and it could involve, for example, radial basis functions, wavelets, polynomials, splines, etc.

Cost approximation can often be significantly enhanced through the use of *feature extraction*, a process that maps the state i into some vector $f(i)$, called the *feature vector* associated with the state i. Feature vectors summarize, in a heuristic sense, what are considered to be important characteristics of the state, and they are very useful in incorporating the designer's prior knowledge or intuition about the problem and about the structure of the optimal controller. For example in a queueing system involving several queues, a feature vector may involve for each queue a three-value indicator, that specifies whether the queue is 'nearly empty', 'moderately busy', or 'nearly full'. In many cases, analysis can complement intuition to suggest the right features for the problem at hand.

Feature vectors are particularly useful when they can capture the 'dominant nonlinearities' in the optimal cost function J^*. By this we mean that $J^*(i)$ can be approximated well by a 'relatively smooth' function $\widehat{J}(f(i))$; this happens for example, if through a change of variables from states to features, the function J^* becomes a (nearly) linear or low-order polynomial function of the features. When a feature vector can be chosen to have this property, one may consider approximation architectures where both features and (relatively simple) neural networks are used together. In partic-

ular, the state is mapped to a feature vector, which is then used as input to a neural network that produces the score of the state. More generally, it is possible that both the state and the feature vector are provided as inputs to the neural network.

A simple method to obtain more sophisticated approximations, is to partition the statee space into several subsets and construct a separate cost function approximation in each subset. For example, by using a linear or quadratic polynomial approximation in each subset of the partition, one can construct piecewise linear or piecewise quadratic approximations over the entire state space. An important issue here is the choice of the method for partitioning the state space. Regular partitions (e. g., grid partitions) may be used, but they often lead to a large number of subsets and very time-consuming computations. Generally speaking, each subset of the partition should contain 'similar' states so that the variation of the optimal cost over the states of the subset is relatively smooth and can be approximated with smooth functions. An interesting possibility is to use features as the basis for partition. In particular, one may use a more or less regular discretization of the space of features, which induces a possibly irregular partition of the original state space. In this way, each subset of the irregular partition contains states with 'similar features'.

Simulation and Training

Some of the most successful applications of neural networks are in the areas of pattern recognition, nonlinear regression, and nonlinear system identification. In these applications the neural network is used as a universal approximator: the input-output mapping of the neural network is matched to an unknown nonlinear mapping F of interest using a least squares optimization. This optimization is known as *training the network*. To perform training, one must have some training data, that is, a set of pairs $(i, F(i))$, which is representative of the mapping F that is approximated.

It is important to note that in contrast with these neural network applications, in the DP context there is no readily available training set of input-output pairs $(i, J^*(i))$, which can be used to approximate J^* with a least squares fit. The only possibility is to evaluate (exactly or approximately) by simulation the cost functions of given (suboptimal) policies, and to try to iteratively improve these policies based on the simulation outcomes. This creates analytical and computational difficulties that do not arise in classical neural network training contexts. Indeed the use of simulation to evaluate approximately the optimal cost function is a key new idea, that distinguishes the methodology of this presentation from earlier approximation methods in DP.

Using simulation offers another major advantage: it allows the methods of this article to be used for systems that are hard to model but easy to simulate; that is, in problems where an explicit model is not available, and the system can only be observed, either as it operates in real time or through a software simulator. For such problems, the traditional DP techniques are inapplicable, and estimation of the transition probabilities to construct a detailed mathematical model is often cumbersome or impossible.

There is a third potential advantage of simulation: it can implicitly identify the 'most important' or 'most representative' states of the system. It appears plausible that if these states are the ones most often visited during the simulation, the scoring function will tend to approximate better the optimal cost for these states, and the suboptimal policy obtained will perform better.

Neuro-Dynamic Programming

The name 'neuro-dynamic programming' expresses the reliance of the methods of this article on both DP and neural network concepts. In the artificial intelligence community, where the methods originated, the name *reinforcement learning* is also used. In common artificial intelligence terms, the methods allow systems to 'learn how to make good decisions by observing their own behavior, and use built-in mechanisms for improving their actions through a reinforcement mechanism'. In less anthropomorphic DP terms, 'observing their own behavior' relates to simulation, and 'improving their actions through a reinforcement mechanism' relates to iterative schemes for improving the quality of approximation of the optimal cost function, or the Q-factors, or the optimal policy. There has been a gradual realization that reinforcement learning techniques can be fruitfully motivated and interpreted in terms of classical DP concepts such as value and policy iteration; see the survey [1], and the book [6], which

point out the connections between the artificial intelligence/reinforcement learning viewpoint and the control theory/DP viewpoint, and give many references.

The currently most popular methodology in NDP iteratively adjusts the parameter vector r of the scoring function $\widetilde{J}(j, r)$ as it produces sample state trajectories $(i_0, \ldots, i_{k+1}, \ldots,)$ by using simulation. These trajectories correspond to either a fixed stationary policy, or to a 'greedy' policy that applies, at state i, the control u that minimizes the expression

$$\mathsf{E}\left\{g(i,u,j)+\widetilde{J}(j,r)|i,u\right\},$$

where r is the current parameter vector. A central notion here is the notion of a *temporal difference*, defined by

$$d_k = g(i_k,u_k,i_{k+1})+\widetilde{J}(i_{k+1},r)-\widetilde{J}(i_k,r),$$

and expressing the difference between our expected cost estimate $\widetilde{J}(i_k, r)$ at state i_k and the predicted cost estimate $g(i_k, u_k, i_{k+1}) + \widetilde{J}(i_{k+1}, r)$ based on the outcome of the simulation. If the cost approximations were exact, the average temporal difference would be zero by Bellman's equation. Thus, roughly speaking, the values of the temporal differences can be used to make incremental adjustments to r so as to bring about an approximate equality (on the average) between expected and predicted cost estimates along the simulated trajectories. This viewpoint, formalized by R.S. Sutton in [5], can be implemented through the use of gradient descent/stochastic approximation methodology. Sutton proposed a family of methods of this type, called TD(λ), and parameterized by a scalar $\lambda \in [0, 1]$. One extreme, TD(1), is closely related to Monte-Carlo simulation and least squares parameter estimation, while the other extreme, TD(0), is closely related to stochastic approximation. A related method is Q-learning, introduced by C.J.C.H. Watkins [9], which is a stochastic approximation-like method that iterates on the Q-factors. While there is convergence analysis of TD(λ) and Q-learning for the case of lookup table representations (see [8,4]), the situation is much less clear in the case of compact representations. A number of results have been derived for approximate policy and value iteration methods, which are obtained from the traditional DP methods after compact representations of the various cost functions involved are introduced.

While the theoretical support for the NDP methodology is only now emerging, there have been quite a few reports of successes with problems too large and complex to be treated in any other way. A particularly impressive success is the development of a backgammon playing program as reported by G. Tesauro [7]. Here a neural network provided a compact representation of the optimal cost function of the game of backgammon by using simulation and TD(λ). The training was performed by letting the program play against itself. After training for several months, the program nearly defeated the human world champion. Variations of the method used by Tesauro have been used with success in a variety of applications.

The recent experience of several researchers, involving several engineering applications, has confirmed that NDP methods can be impressively effective in problems where traditional DP methods would be hardly applicable and other heuristic methods would have a limited chance of success. We note, however, that the practical application of NDP is computationally very intensive, and often requires a considerable amount of trial and error. Fortunately, all the computation and experimentation with different approaches can be done off-line. Once the approximation is obtained off-line, it can be used to generate decisions fast enough for use in real time. In this context, we mention that in the machine learning literature, reinforcement learning is often viewed as an 'on-line' method, whereby the cost approximation is improved as the system operates in real time. This is reminiscent of the methods of traditional adaptive control.

Extensive references for the material of this article are the research monographs [3,6]. A more limited textbook discussion is given in [2]. The survey [1], and the overviews [10,11], and other papers in the edited volume [12] point out the connections between the artificial intelligence/reinforcement learning viewpoint and the control theory/DP viewpoint, and give many references.

See also

- ► Dynamic Programming: Average Cost Per Stage Problems
- ► Dynamic Programming in Clustering

- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neural Networks for Combinatorial Optimization
- ▶ Replicator Dynamics in Combinatorial Optimization
- ▶ Unconstrained Optimization in Neural Network Training

References

1. Barto AG, Bradtke SJ, Singh SP (1995) Real-time learning and control using asynchronous dynamic programming. Artif Intell 72:81–138
2. Bertsekas DP (1995) Dynamic programming and optimal control, vol II, Athena Sci., Belmont
3. Bertsekas DP, Tsitsiklis JN (1996) Neuro-dynamic programming. Athena Sci., Belmont
4. Jaakkola T, Jordan MI, Singh SP (1994) On the convergence of stochastic iterative dynamic programming algorithms. Neural Computation 6:1185–1201
5. Sutton RS (1988) Learning to predict by the methods of temporal differences. Machine Learning 3:9–44
6. Sutton RS, Barto AG (1998) Reinforcement learning. MIT, Cambridge
7. Tesauro G (1992) Practical issues in temporal difference learning. Machine Learning 8:257–277
8. Tsitsiklis JN (1994) Asynchronous stochastic approximation and Q-learning. Machine Learning 16:185–202
9. Watkins CJCH (1989) Learning from delayed rewards. PhD Thesis Cambridge Univ, Cambridge
10. Werbös PJ (1992) Approximate dynamic programming for real-time control and neural modeling. In: White DA, Sofge DA (eds) Handbook of Intelligent Control. v. Nostrand, Princeton
11. Werbös PJ (1992) Neurocontrol and supervised learning: An overview and valuation. In: White DA, Sofge DA (eds) Handbook of Intelligent Control. v. Nostrand, Princeton
12. White DA, Sofge DA (eds) (1992) Handbook of Intelligent Control. v. Nostrand, Princeton

New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization

NECULAI ANDREI[1,2]

[1] Center for Advanced Modeling and Optimization, Research Institute for Informatics, Bucharest, Romania

[2] Academy of Romanian Scientists, Bucharest, Romania

MSC2000: 49M07, 49M10, 90C06, 65K

Article Outline

Keywords and Phrases
Introduction
New Hybrid Conjugate Gradient Algorithms
  The New Hybrid Algorithms (CCOMB, NDOMB)
Convergence Analysis
Numerical Experiments
Conclusion
References

Keywords and Phrases

Unconstrained optimization; Hybrid conjugate gradient method; Conjugacy condition; Numerical comparisons; Dolan–Moré profile

New hybrid conjugate gradient algorithms are proposed and analyzed. In these hybrid algorithms the famous parameter β_k is computed as a convex combination of the Polak–Ribière–Polyak and Dai–Yuan conjugate gradient algorithms. In one hybrid algorithm the parameter in convex combination is computed in such a way that the conjugacy condition is satisfied, independent of the line search. In the other, the parameter in convex combination is computed in such a way that the conjugate gradient direction is the Newton direction. The algorithm uses the standard Wolfe line search conditions. Numerical comparisons with conjugate gradient algorithms using a set of 750 unconstrained optimization problems, some of them from the CUTE library, show that the hybrid computational scheme based on the conjugacy condition outperforms the known hybrid conjugate gradient algorithms.

Introduction

Let us consider the nonlinear unconstrained optimization problem

$$\min \{f(x) : x \in R^n\} , \tag{1}$$

where $f : R^n \to R$ is a continuously differentiable function, bounded from below. For solving this problem, starting from an initial guess $x_0 \in R^n$, a nonlinear conjugate gradient method generates a sequence $\{x_k\}$ as

$$x_{k+1} = x_k + \alpha_k d_k , \tag{2}$$

where $\alpha_k > 0$ is obtained by line search, and the directions d_k are generated as

$$d_{k+1} = -g_{k+1} + \beta_k s_k , \quad d_0 = -g_0 . \tag{3}$$

In (3) β_k is known as the conjugate gradient parameter, $s_k = x_{k+1} - x_k$ and $g_k = \nabla f(x_k)$. Consider $\|.\|$ the Euclidean norm and define $y_k = g_{k+1} - g_k$. The line search in the conjugate gradient algorithms is often based on the standard Wolfe conditions:

$$f(x_k + \alpha_k d_k) - f(x_k) \le \rho \alpha_k g_k^{\mathrm{T}} d_k , \tag{4}$$

$$g_{k+1}^{\mathrm{T}} d_k \ge \sigma g_k^{\mathrm{T}} d_k , \tag{5}$$

where d_k is a descent direction and $0 < \rho \le \sigma < 1$. Plenty of conjugate gradient methods are known, and an excellent survey of these methods, with special attention to their global convergence, was given by Hager and Zhang [19]. Different conjugate gradient algorithms correspond to different choices for the scalar parameter β_k. Some of these methods, such those of Fletcher and Reeves (FR) [16], Dai and Yuan (DY) [12] and conjugate descent (CD) proposed by Fletcher [15], have strong convergence properties, but they may have modest practical performance owing to jamming:

$$\beta_k^{\mathrm{FR}} = \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} , \quad \beta_k^{\mathrm{DY}} = \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} ,$$
$$\beta_k^{\mathrm{CD}} = \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{-g_k^{\mathrm{T}} s_k} .$$

On the other hand, the methods of Polak and Ribière [23] and Polyak (PRP) [24], Hestenes and Stiefel (HS) [20] or Liu and Storey (LS) [22] in general may not be convergent, but they often have better computational performances:

$$\beta_k^{\mathrm{PRP}} = \frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k} , \quad \beta_k^{\mathrm{HS}} = \frac{g_{k+1}^{\mathrm{T}} y_k}{y_k^{\mathrm{T}} s_k} , \quad \beta_k^{\mathrm{LS}} = \frac{g_{k+1}^{\mathrm{T}} y_k}{-g_k^{\mathrm{T}} s_k} .$$

In this contribution we focus on hybrid conjugate gradient methods. These methods are combinations of different conjugate gradient algorithms; mainly they are proposed to avoid the jamming phenomenon. One of the first hybrid conjugate gradient algorithms was introduced by Touati-Ahmed and Storey [28], where the parameter β_k is computed as

$$\beta_k^{TS} = \begin{cases} \beta_k^{\mathrm{PRP}} = \frac{g_{k+1}^{\mathrm{T}} y_k}{\|g_k\|^2}, & \text{if } 0 \le \beta_k^{\mathrm{PRP}} \le \beta_k^{\mathrm{FR}}, \\ \beta_k^{\mathrm{FR}} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}, & \text{otherwise.} \end{cases}$$

The PRP method has a built-in restart feature that directly addresses jamming. Indeed, when the step s_k is small, then the factor y_k in the numerator of β_k^{PRP} tends to zero. Therefore, β_k^{PRP} becomes small and the search direction d_{k+1} is very close to the steepest descent direction $-g_{k+1}$. Hence, when the iterations jam, the method of Touati-Ahmed and Storey uses the PRP computational scheme.

Another hybrid conjugate gradient method was given by Hu and Storey [21], where β_k in (3) is

$$\beta_k^{\mathrm{HuS}} = \max \{0, \min \{\beta_k^{\mathrm{PRP}}, \beta_k^{\mathrm{FR}}\}\} .$$

As above, when the method of Hu and Storey is jamming, then the PRP method is used instead.

The combination of LS and CD conjugate gradient methods leads to the following hybrid method:

$$\beta_k^{\mathrm{LS-CD}} = \max \{0, \min \{\beta_k^{\mathrm{LS}}, \beta_k^{\mathrm{CD}}\}\} .$$

The CD method of Fletcher [15] is very similar to the FR method. With an exact line search, the CD method is identical to the FR method. Similarly, for an exact line search, the LS method is identical to the PRP method. Therefore, the hybrid LS–CD method with an exact line search has similar performances as the hybrid method of Hu and Storey.

Gilbert and Nocedal [17] suggested a combination between PRP and FR methods as

$$\beta_k^{\mathrm{GN}} = \max \{-\beta_k^{\mathrm{FR}}, \min \{\beta_k^{\mathrm{PRP}}, \beta_k^{\mathrm{FR}}\}\} .$$

Since β_k^{FR} is always nonnegative, it follows that β_k^{GN} can be negative. The method of Gilbert and Nocedal has the same advantage of avoiding jamming.

Using the standard Wolfe line search, the DY method always generates descent directions and if the gradient is Lipschitz-continuous the method is globally convergent. In an effort to improve their algorithm, Dai and Yuan [13] combined their algorithm with other conjugate gradient algorithms, and proposed the following two hybrid methods:

$$\beta_k^{\mathrm{hDY}} = \max\left\{-c\beta_k^{\mathrm{DY}}, \min\left\{\beta_k^{\mathrm{HS}}, \beta_k^{\mathrm{DY}}\right\}\right\} ,$$

$$\beta_k^{\mathrm{hDYz}} = \max\left\{0, \min\left\{\beta_k^{\mathrm{HS}}, \beta_k^{\mathrm{DY}}\right\}\right\} ,$$

where $c = (1-\sigma)/(1+\sigma)$. For the standard Wolfe conditions (4) and (5), under the Lipschitz continuity of the gradient, Dai and Yuan [13] established the global convergence of these hybrid computational schemes.

In the following we propose another hybrid conjugate gradient as a convex combination of PRP and DY conjugate gradient algorithms. We selected these two methods for combination in a hybrid conjugate gradient algorithm because the PRP algorithm has good computational properties, on one hand, and the DY algorithm has strong convergence properties, on the other hand. Often the PRP method performs better in practice than the DY method and we speculate this in order to have a good practical conjugate algorithm. The structure of this chapter is as follows. In Sect. "New Hybrid Conjugate Gradient Algorithms" we introduce our hybrid conjugate gradient algorithm and prove that it generates descent directions satisfying in some conditions the sufficient descent condition. Section "The New Hybrid Algorithms (CCOMB, NDOMB)" presents the algorithms and in Sect. "Convergence Analysis" we show the convergence analysis. In Sect. "Numerical Experiments" some numerical experiments and performance profiles of Dolan–Moré [14] corresponding to this new hybrid conjugate gradient algorithm and some other conjugate gradient algorithms are presented. The performance profiles corresponding to a set of 750 unconstrained optimization problems in the CUTE test problem library [6] as well as some other unconstrained optimization problems presented in [1] show that this hybrid conjugate gradient algorithm outperforms the known hybrid conjugate gradient algorithms.

New Hybrid Conjugate Gradient Algorithms

The iterates $x_0, x_1, x_2, \ldots$ of our algorithm are computed by means of recurrence (2) where the step size $\alpha_k > 0$ is determined according to the Wolfe conditions (4) and (5), and the directions d_k are generated by the rule:

$$d_{k+1} = -g_{k+1} + \beta_k^{\mathrm{N}} s_k , \qquad d_0 = -g_0 , \tag{6}$$

where

$$\begin{aligned}\beta_k^{\mathrm{N}} &= (1-\theta_k)\beta_k^{\mathrm{PRP}} + \theta_k \beta_k^{\mathrm{DY}} \\ &= (1-\theta_k)\frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k} + \theta_k \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k}\end{aligned} \tag{7}$$

and θ_k is a scalar parameter satisfying $0 \le \theta_k \le 1$, which needs to be determined. Observe that if $\theta_k = 0$, then $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{PRP}}$, and if $\theta_k = 1$, then $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{DY}}$. On the other hand, if $0 < \theta_k < 1$, then β_k^{N} is a convex combination of β_k^{PRP} and β_k^{DY}.

Referring to the PRP method, Polak and Ribière [23] proved that when function f is strongly convex and the line search is exact, then the PRP method is globally convergent. In an effort to understand the behavior of the PRP method, Powell [25] showed that if the step length $s_k = x_{k+1} - x_k$ approaches zero, the line search is exact and the gradient $\nabla f(x)$ is Lipschitz-continuous, then the PRP method is globally convergent. Additionally, assuming that the search direction is a descent direction, Yuan [29] established the global convergence of the PRP method for strongly convex functions and a Wolfe line search. For general nonlinear functions the convergence of the PRP method is uncertain. Powell [26] gave a three-dimensional example, in which the function to be minimized is not strongly convex, showing that even with an exact line search the PRP method may not converge to a stationary point. Later on Dai [7] presented another example, this time with a strongly convex function for which the PRP method fails to generate a descent direction. Therefore, theoretically the convergence of the PRP method is limited to strongly convex functions. For general nonlinear functions the convergence of the PRP method is established under restrictive conditions (Lipschitz continuity, exact line search and the step size tends to zero). However, the numerical experiments presented, for example, by Gilbert and Nocedal [17] proved that the PRP method is one of the best conjugate

gradient methods, and this was the main motivation to consider it in (7).

On the other hand, the DY method always generates descent directions, and in [8] Dai established a remarkable property for the DY conjugate gradient algorithm, relating the descent directions to the sufficient descent condition. It was shown that if there exist constants γ_1 and γ_2 such that $\gamma_1 \le \|g_k\| \le \gamma_2$ for all k, then for any $p \in (0,1)$ there exists a constant $c > 0$ such that the sufficient descent condition $g_i^{\mathrm{T}} d_i \le -c \|g_i\|^2$ holds for at least $\lfloor pk \rfloor$ indices $i \in [0,k]$, where $\lfloor j \rfloor$ denotes the largest integer $\le j$. Therefore, this property is the main reason we consider the DY method in (7).

It easy to see that

$$d_{k+1} = -g_{k+1} + (1-\theta_k)\frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} s_k + \theta_k \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} s_k . \tag{8}$$

Supposing that d_k is a descent direction ($d_0 = -g_0$), then for the algorithm given by (2) and (8) we can prove the following result.

Theorem 1 *Assume that α_k in algorithm (2) and (8) is determined by Wolfe line search (4) and (5). If $0 < \theta_k < 1$, and*

$$\left| \frac{g_k^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \right| \|g_{k+1}\|^2 \ge \frac{(g_{k+1}^{\mathrm{T}} y_k)(g_{k+1}^{\mathrm{T}} s_k)}{\|g_k\|^2} , \tag{9}$$

then direction d_{k+1} given by (8) is a descent direction.

Proof Since $0 < \theta_k < 1$, from (8) we get

$$\begin{aligned} g_{k+1}^{\mathrm{T}} d_{k+1} &= -\|g_{k+1}\|^2 + (1-\theta_k)\frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} g_{k+1}^{\mathrm{T}} s_k \\ &\quad + \theta_k \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} g_{k+1}^{\mathrm{T}} s_k \\ &\le -\|g_{k+1}\|^2 + \frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} g_{k+1}^{\mathrm{T}} s_k \\ &\quad + \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} g_{k+1}^{\mathrm{T}} s_k \\ &= \left(-1 + \frac{g_{k+1}^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \right) \|g_{k+1}\|^2 \\ &\quad + \frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} g_{k+1}^{\mathrm{T}} s_k \\ &= \frac{g_k^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \|g_{k+1}\|^2 + \frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} g_{k+1}^{\mathrm{T}} s_k . \end{aligned}$$

But, $y_k^{\mathrm{T}} s_k > 0$ by (5) and since $g_k^{\mathrm{T}} s_k \le 0$, it follows that

$$\frac{g_k^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \|g_{k+1}\|^2 \le 0 .$$

Therefore, from (9), it follows that $g_{k+1}^{\mathrm{T}} d_{k+1} \le 0$, i. e., the direction d_{k+1} is a descent one. □

Theorem 2 *Suppose that $(g_{k+1}^{\mathrm{T}} y_k)(g_{k+1}^{\mathrm{T}} s_k) \le 0$. If $0 < \theta_k < 1$, then the direction d_{k+1} given by (8) satisfies the sufficient descent condition*

$$g_{k+1}^{\mathrm{T}} d_{k+1} \le -\left(1 - \theta_k \frac{g_{k+1}^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \right) \|g_{k+1}\|^2 . \tag{10}$$

Proof From (8) we have

$$\begin{aligned} g_{k+1}^{\mathrm{T}} d_{k+1} &= -\|g_{k+1}\|^2 + (1-\theta_k)\frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k} g_{k+1}^{\mathrm{T}} s_k \\ &\quad + \theta_k \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} g_{k+1}^{\mathrm{T}} s_k \\ &= -\|g_{k+1}\|^2 + \theta_k \frac{g_{k+1}^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \|g_{k+1}\|^2 \\ &\quad + (1-\theta_k)\frac{(g_{k+1}^{\mathrm{T}} y_k)(g_{k+1}^{\mathrm{T}} s_k)}{g_k^{\mathrm{T}} g_k} \\ &\le -\left(1 - \theta_k \frac{g_{k+1}^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \right) \|g_{k+1}\|^2 \le 0 . \end{aligned}$$

Observe that since $y_k^{\mathrm{T}} s_k > 0$ by (5) and since $g_{k+1}^{\mathrm{T}} s_k = y_k^{\mathrm{T}} s_k + g_k^{\mathrm{T}} s_k < y_k^{\mathrm{T}} s_k$, then $y_k^{\mathrm{T}} s_k / g_{k+1}^{\mathrm{T}} s_k > 1$. Therefore, if $0 < \theta_k < 1$, it follows that $\theta_k < y_k^{\mathrm{T}} s_k / g_{k+1}^{\mathrm{T}} s_k$. Therefore,

$$1 - \theta_k \frac{g_{k+1}^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} > 0 ,$$

proving the theorem. □

To select the parameter θ_k we consider the following two possibilities. In the first hybrid conjugate gradient algorithm the parameter θ_k is selected in such a manner that the *conjugacy condition* $y_k^{\mathrm{T}} d_{k+1} = 0$ is satisfied at every iteration, independent of the line search. Hence,

from $y_k^{\mathrm{T}} d_{k+1} = 0$ after some algebra, using (8), we get

$$\theta_k = \theta_k^{\mathrm{CCOMB}} \equiv \frac{(y_k^{\mathrm{T}} g_{k+1})(y_k^{\mathrm{T}} s_k) - (y_k^{\mathrm{T}} g_{k+1})(g_k^{\mathrm{T}} g_k)}{(y_k^{\mathrm{T}} g_{k+1})(y_k^{\mathrm{T}} s_k) - \|g_{k+1}\|^2 \|g_k\|^2} . \tag{11}$$

In the second algorithm the parameter θ_k is selected in such a manner that the direction d_{k+1} from (8) is the Newton direction, i. e.,

$$-\nabla^2 f(x_{k+1})^{-1} g_{k+1} = -g_{k+1} + (1-\theta_k)\frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} s_k + \theta_k \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} s_k . \tag{12}$$

Having in view that $\nabla^2 f(x_{k+1}) s_k = y_k$, from (12) we get

$$\theta_k = \theta_k^{\mathrm{NDOMB}} \equiv \frac{(y_k^{\mathrm{T}} g_{k+1} - s_k^{\mathrm{T}} g_{k+1})\|g_k\|^2 - (g_{k+1}^{\mathrm{T}} y_k)(y_k^{\mathrm{T}} s_k)}{\|g_{k+1}\|^2 \|g_k\|^2 - (g_{k+1}^{\mathrm{T}} y_k)(y_k^{\mathrm{T}} s_k)} . \tag{13}$$

Observe that the parameter θ_k given by (11) or (13) can be outside the interval $[0, 1]$. However, in order to have a real convex combination in (7) the following rule is considered: if $\theta_k \le 0$, then set $\theta_k = 0$ in (7), i. e., $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{PRP}}$; if $\theta_k \ge 1$, then take $\theta_k = 1$ in (7), i. e., $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{DY}}$. Therefore, under this rule for θ_k selection, the direction d_{k+1} in (8) combines the properties of the PRP and DY algorithms.

The New Hybrid Algorithms (CCOMB, NDOMB)

Step 1 *Initialization.* Select $x_0 \in R^n$ and the parameters $0 < \rho \le \sigma < 1$. Compute $f(x_0)$ and g_0. Consider $d_0 = -g_0$ and set the initial guess: $\alpha_0 = 1/\|g_0\|$.

Step 2 *Test for continuation of iterations.* If $\|g_k\|_\infty \le 10^{-6}$, then stop.

Step 3 *Line search.* Compute $\alpha_k > 0$ satisfying the Wolfe line search conditions (4) and (5) and update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 4. θ_k *parameter computation.* If $(y_k^{\mathrm{T}} g_{k+1})(y_k^{\mathrm{T}} s_k) - \|g_{k+1}\|^2 \|g_k\|^2 = 0$, then set $\theta_k = 0$, otherwise compute θ_k as follows:

CCOMB algorithm (θ_k from the conjugacy condition): $\theta_k = \theta_k^{\mathrm{CCOMB}}$.

NDOMB algorithm (θ_k from the Newton direction): $\theta_k = \theta_k^{\mathrm{NDOMB}}$.

Step 5 β_k^{N} *conjugate gradient parameter computation.* If $0 < \theta_k < 1$, then compute β_k^{N} as in (7). If $\theta_k \ge 1$, then set $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{DY}}$. If $\theta_k \le 0$, then set $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{PRP}}$.

Step 6 *Direction computation.* Compute $d = -g_{k+1} + \beta_k^{\mathrm{N}} s_k$. If the restart criterion of Powell,

$$\left|g_{k+1}^{\mathrm{T}} g_k\right| \ge 0.2 \|g_{k+1}\|^2 , \tag{14}$$

is satisfied, then set $d_{k+1} = -g_{k+1}$; otherwise define $d_{k+1} = d$. Compute the initial guess $\alpha_k = \alpha_{k-1}\|d_{k-1}\| / \|d_k\|$, set $k = k+1$ and continue with step 2.

It is well known that if f is bounded along the direction d_k then there exists a step size α_k satisfying the Wolfe line search conditions (4) and (5). In our algorithm when the Powell restart condition is satisfied, we restart the algorithm with the negative gradient $-g_{k+1}$. More sophisticated reasons for restarting the algorithms have been proposed in the literature [10], but we are interested in the performance of a conjugate gradient algorithm that uses this restart criterion, associated with a direction satisfying the conjugacy condition or that is equal to the Newton direction. Under reasonable assumptions, conditions (4), (5) and (14) are sufficient to prove the global convergence of the algorithm. We consider this aspect in the next section.

The first trial of the step length crucially affects the practical behavior of the algorithm. At every iteration $k \ge 1$ the starting guess for step α_k in the line search is computed as $\alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. This selection was considered for the first time by Shanno and Phua [27] in CONMIN. It is also considered in the packages SCG by Birgin and Martínez [5] and in SCALCG by Andrei [2,3,4].

Convergence Analysis

Throughout this section we assume that

1. *The level set* $S = \{x \in R^n : f(x) \le f(x_0)\}$ *is bounded.*
2. *In a neighborhood* N *of* S, *the function* f *is continuously differentiable and its gradient is Lipschitz-continuous, i. e., there exists a constant* $L > 0$

such that $\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|$, *for all* $x, y \in N$.

Under these assumptions for f, there exists a constant $\Gamma \ge 0$ such that $\|\nabla f(x)\| \le \Gamma$, for all $x \in S$.

It was proved in [11] that for any conjugate gradient method with strong Wolfe line search the following lemma holds.

Lemma 1 *Suppose that assumptions 1 and 2 hold and consider any conjugate gradient method (2) and (3), where d_k is a descent direction and α_k is obtained by the strong Wolfe line search. If*

$$\sum_{k\ge 1} \frac{1}{\|d_k\|^2} = \infty\,, \tag{15}$$

then

$$\liminf_{k\to\infty} \|g_k\| = 0\,. \tag{16}$$

For uniformly convex functions which satisfy the above assumptions we can prove that the norm of d_{k+1} generated by (8) is bounded above. Thus, by Lemma 1 we have the following result.

Theorem 3 *Suppose that assumptions 1 and 2 hold. Consider the algorithm (2) and (8), where d_{k+1} is a descent direction and α_k is obtained by the strong Wolfe line search.*

$$f(x_k + \alpha_k d_k) - f(x_k) \le \rho\alpha_k g_k^{\mathrm{T}} d_k\,, \tag{17}$$

$$\left|g_{k+1}^{\mathrm{T}} d_k\right| \le -\sigma g_k^{\mathrm{T}} d_k\,. \tag{18}$$

If for $k \ge 0$, $\|s_k\|$ tends to zero and there exist the nonnegative constants η_1 and η_2 such that

$$\|g_k\|^2 \ge \eta_1 \|s_k\|^2 \text{ and } \|g_{k+1}\|^2 \le \eta_2 \|s_k\|\,, \tag{19}$$

and the function f is a uniformly convex function, i. e., there exists a constant $\mu \ge 0$ such that for all $x, y \in S$

$$(\nabla f(x) - \nabla f(y))^{\mathrm{T}}(x - y) \ge \mu\|x - y\|^2\,, \tag{20}$$

then

$$\lim_{k\to\infty} g_k = 0\,. \tag{21}$$

Proof From (20) it follows that $y_k^{\mathrm{T}} s_k \ge \mu\|s_k\|^2$. Now, since $0 \le \theta_k \le 1$, from uniform convexity and (19) we have

$$\left|\beta_k^{\mathrm{N}}\right| \le \left|\frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k}\right| + \left|\frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k}\right| \le \frac{\|g_{k+1}\|\,\|y_k\|}{\eta_1\|s_k\|^2} + \frac{\eta_2\|s_k\|}{\mu\|s_k\|^2}\,.$$

But $\|y_k\| \le L\|s_k\|$; therefore,

$$\left|\beta_k^{\mathrm{N}}\right| \le \frac{\Gamma L}{\eta_1\|s_k\|} + \frac{\eta_2}{\mu\|s_k\|}\,.$$

Hence,

$$\|d_{k+1}\| \le \|g_{k+1}\| + \left|\beta_k^{\mathrm{N}}\right|\|s_k\| \le \Gamma + \frac{\Gamma L}{\eta_1} + \frac{\eta_2}{\mu}\,,$$

which implies that (15) is true. Therefore, by Lemma 1 we have (16), which for uniformly convex functions is equivalent to (21). □

Powell [25] showed that for general functions the PRP method is globally convergent if the step lengths $\|s_k\| = \|x_{k+1} - x_k\|$ tend to zero, i. e., $\|s_k\| \le \|s_{k-1}\|$ is a condition of convergence. For convergence of our algorithms from (19) we see that along the iterations, for $k \ge 1$, the gradient must be bounded as $\eta_1\|s_k\|^2 \le \|g_k\|^2 \le \eta_2\|s_{k-1}\|$. If the Powell condition is satisfied, i. e., $\|s_k\|$ tends to zero, then $\|s_k\|^2 \ll \|s_{k-1}\|$ and therefore the norm of the gradient can satisfy (19). In the numerical experiments we observed that (19) is always satisfied in the last part of the iterations.

For general nonlinear functions the convergence analysis of our algorithm exploits insights developed by Gilbert and Nocedal [17], Dai and Liao [9] and Hager and Zhang [18]. Global convergence proof of these new hybrid conjugate gradient algorithms is based on the Zoutendijk condition combined with the analysis showing that the sufficient descent condition holds and $\|d_k\|$ is bounded. Suppose that level set S is bounded and function f is bounded from below.

Lemma 2 *Assume that d_k is a descent direction and ∇f satisfies the Lipschitz condition $\|\nabla f(x) - \nabla f(x_k)\| \le L\|x - x_k\|$ for all x on the line segment connecting x_k and x_{k+1}, where L is a constant. If the line search satisfies the second Wolfe condition (5), then*

$$\alpha_k \ge \frac{1-\sigma}{L}\frac{\left|g_k^{\mathrm{T}} d_k\right|}{\|d_k\|^2}\,. \tag{22}$$

Proof Subtracting $g_k^{\mathrm{T}} d_k$ from both sides of (5) and using the Lipschitz condition, we have

$$(\sigma - 1) g_k^{\mathrm{T}} d_k \le (g_{k+1} - g_k)^{\mathrm{T}} d_k \le L \alpha_k \|d_k\|^2 . \quad (23)$$

Since d_k is a descent direction and $\sigma < 1$, (22) follows immediately from (23). □

Theorem 4 *Suppose that assumptions 1 and 2 hold, $0 < \theta_k \le 1, (g_{k+1}^{\mathrm{T}} y_k)(g_{k+1}^{\mathrm{T}} s_k) \le 0$, for every $k \ge 0$ there exists a positive constant ω, such that $1 - \theta_k (g_{k+1}^{\mathrm{T}} s_k)/(y_k^{\mathrm{T}} s_k) \ge \omega > 0$, and there exist the constants γ and Γ, such that for all $k, \gamma \le \|g_k\| \le \Gamma$. Then for the computational scheme (2) and (8), where α_k is determined by the Wolfe line search (4) and (5), either $g_k = 0$ for some k or*

$$\liminf_{k \to \infty} \|g_k\| = 0 . \quad (24)$$

Proof By the Wolfe condition (5) we have

$$y_k^{\mathrm{T}} s_k = (g_{k+1} - g_k)^{\mathrm{T}} s_k \ge (\sigma - 1) g_k^{\mathrm{T}} s_k = -(1 - \sigma) g_k^{\mathrm{T}} s_k . \quad (25)$$

By Theorem 2, and the assumption $1 - \theta_k (g_{k+1}^{\mathrm{T}} s_k)/(y_k^{\mathrm{T}} s_k) \ge \omega$, it follows that

$$g_k^{\mathrm{T}} d_k \le -\left(1 - \theta_{k-1} \frac{g_k^{\mathrm{T}} s_{k-1}}{y_{k-1}^{\mathrm{T}} s_{k-1}}\right) \|g_k\|^2 \le -\omega \|g_k\|^2 .$$

Therefore,

$$-g_k^{\mathrm{T}} d_k \ge \omega \|g_k\|^2 . \quad (26)$$

Combining (25) with (26), we get

$$y_k^{\mathrm{T}} s_k \ge (1 - \sigma) \omega \alpha_k \gamma^2 .$$

On the other hand $\|y_k\| = \|g_{k+1} - g_k\| \le L \|s_k\|$; hence,

$$\left| g_{k+1}^{\mathrm{T}} y_k \right| \le \|g_{k+1}\| \, \|y_k\| \le \Gamma L \|s_k\| .$$

With these, from (7) we get

$$\left| \beta_k^{\mathrm{N}} \right| \le \left| \frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k} \right| + \left| \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} \right| .$$

But

$$\left| \frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k} \right| \le \frac{\|g_{k+1}\| \, \|y_k\|}{\gamma^2} \le \frac{\Gamma L \|s_k\|}{\gamma^2} \le \frac{\Gamma L D}{\gamma^2} ,$$

where $D = \max\{\|y - z\| : y, z \in S\}$ is the diameter of the level set S.

On the other hand,

$$\left| \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} \right| \le \frac{\Gamma^2}{(1 - \sigma) \omega \alpha_k \gamma^2} .$$

Therefore,

$$\left| \beta_k^{\mathrm{N}} \right| \le \frac{\Gamma L D}{\gamma^2} + \frac{\Gamma^2}{(1 - \sigma) \omega \alpha_k \gamma^2} \equiv E . \quad (27)$$

Now, we can write

$$\|d_{k+1}\| \le \|g_{k+1}\| + \left| \beta_k^{\mathrm{N}} \right| \|s_k\| \le \Gamma + E D . \quad (28)$$

Since the level set S is bounded and the function f is bounded from below, using Lemma 2, from (4) it follows that

$$0 < \sum_{k=0}^{\infty} \frac{(g_k^{\mathrm{T}} d_k)^2}{\|d_k\|^2} < \infty , \quad (29)$$

i. e., the Zoutendijk condition holds. Therefore, from Theorem 2 using (29), the descent property yields

$$\sum_{k=0}^{\infty} \frac{\gamma^4}{\|d_k\|^2} \le \sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} \le \sum_{k=0}^{\infty} \frac{1}{\omega^2} \frac{(g_k^{\mathrm{T}} d_k)^2}{\|d_k\|^2} < \infty ,$$

which contradicts (28). Hence, $\gamma = \liminf_{k \to \infty} \|g_k\| = 0$. □

Therefore, when $0 < \theta_k \le 1$ our hybrid conjugate gradient algorithms are globally convergent, meaning that either $g_k = 0$ for some k or (24) holds. Observe that in the conditions of Theorem 2 the direction d_{k+1} satisfies the sufficient descent condition independent of the line search.

Numerical Experiments

In this section we present the computational performance of a Fortran implementation of the CCOMB and NDOMB algorithms for a set of 750 unconstrained optimization test problems. The test problems are the unconstrained problems in the CUTE [6] library, along with other large-scale optimization problems presented in [1]. We selected 75 large-scale unconstrained optimization problems in extended or generalized form. Each problem was tested ten times for a gradually increasing number of variables: $n = 1000, 2000, \ldots,$

10000. At the same time we present comparisons with other conjugate gradient algorithms, including the performance profiles of Dolan and Moré [14].

All algorithms implement the Wolfe line search conditions with $\rho = 0.0001$ and $\sigma = 0.9$, and the same stopping criterion $\|g_k\|_\infty \leq 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector.

The comparisons of algorithms are given in the following context. Let f_i^{ALG1} and f_i^{ALG2} be the optimal values found by ALG1 and ALG2, for problem $i = 1, \ldots, 750$, respectively. We say that, in the particular problem i, the performance of ALG1 was better than the performance of ALG2 if

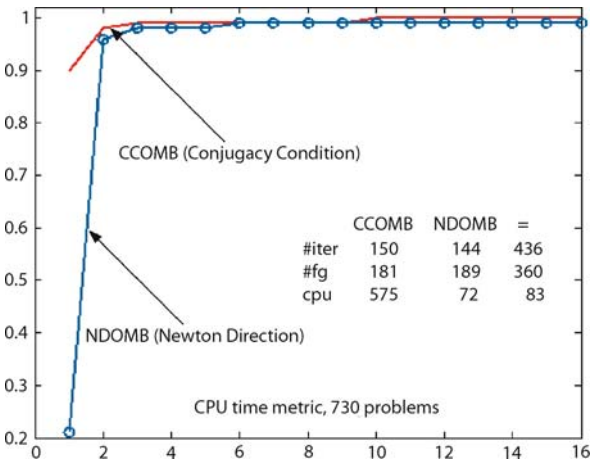
$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{30}$$

and the number of iterations, or the number of function-gradient evaluations, or the CPU time of ALG1 was less than the number of iterations, or the number of function-gradient evaluations, or the CPU time corresponding to ALG2, respectively.

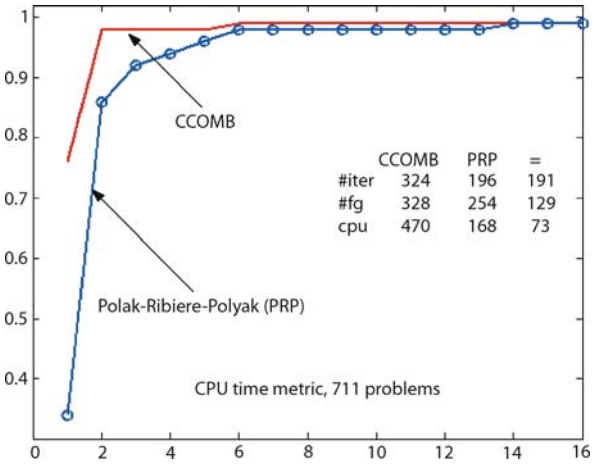
All codes were written in double-precision Fortran and compiled with f77 (default compiler settings) on an Intel Pentium 4, 1.8 GHz workstation. All these codes were authored by Andrei. The performances of these algorithms were evaluated using the profiles of Dolan and Moré [14]. That is, for each algorithm we plotted the fraction of problems for which the algorithm is within a factor of the best CPU time. The left side of these figures gives the percentage of the test problems, out of 750, for which an algorithm is more performant; the right side gives the percentage of the test problems that were successfully solved by each of the algorithms. Mainly, the right side represents a measure of an algorithm's robustness.

In the first set of numerical experiments, we compared the performance of the CCOMB algorithm with the performance of the NDOMB algorithm. Figure 1 shows the Dolan and Moré CPU performance profile of the CCOMB algorithm compared with that of the NDOMB algorithm.

Observe that the CCOMB algorithm outperforms the NDOMB algorithm in the vast majority of problems. Only 730 problems out of 750 satisfy criterion (30). Referring to the CPU time, the CCOMB algorithm was better in 575 problems, in contrast to the NDOMB algorithm, which solved only 72 problems in a better CPU time.



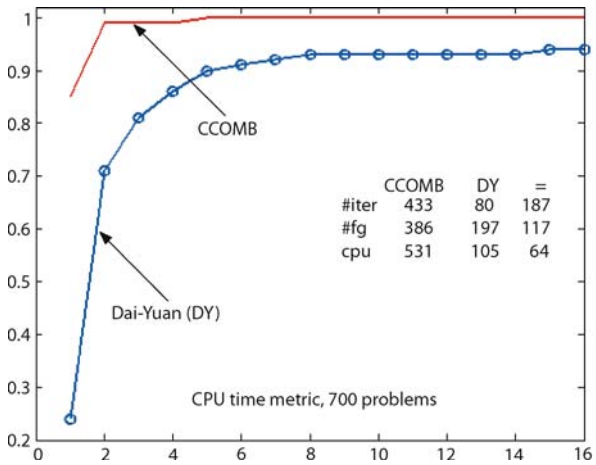
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 1
Performance based on CPU time: CCOMB algorithm compared with the NDOMB algorithm



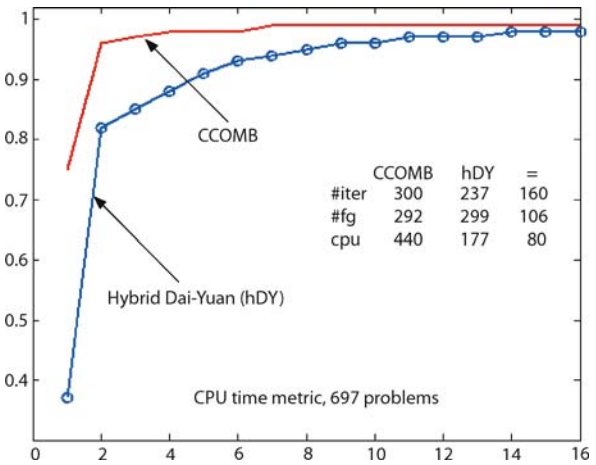
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 2
Performance based on CPU time: CCOMB algorithm compared with the Polak–Ribière–Polyak (*PRP*) algorithm

In the second set of numerical experiments we compared the performance of the CCOMB algorithm with the performances of the PRP and DY conjugate gradient algorithms. Figures 2 and 3 show the Dolan and Moré CPU performance profiles of the CCOMB algorithm compared with those of the PRP and DY algorithms, respectively.

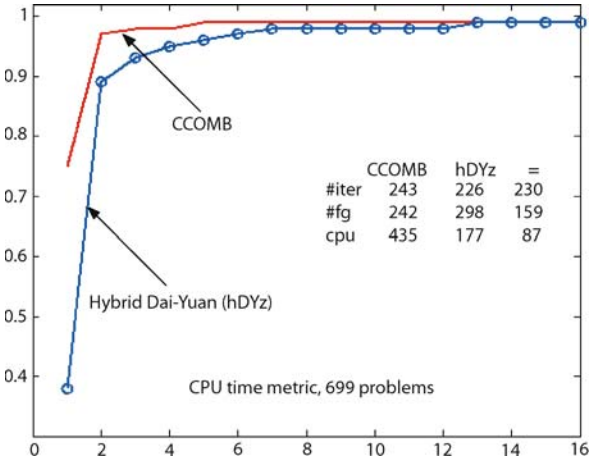
When comparing the CCOMB and PRP algorithms (Fig. 2), subject to the number of iterations, we see that the CCOMB algorithm was better in 324 problems (i. e.,



New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 3
Performance based on CPU time: CCOMB algorithm compared with the Dai–Yuan (*DY*) algorithm



New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 4
Performance based on CPU time: CCOMB algorithm compared with the hybrid Dai–Yuan (*hDY*) algorithm



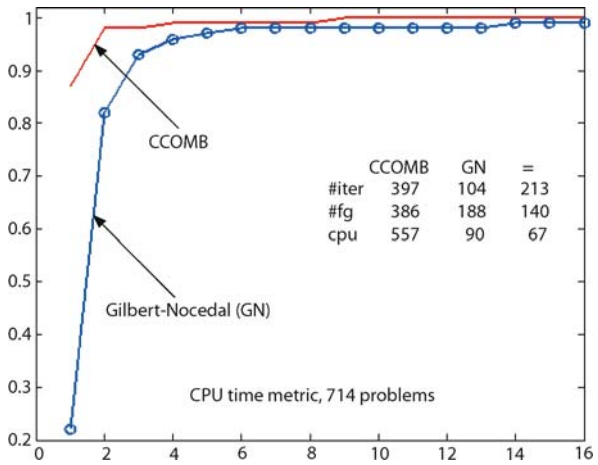
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 5
Performance based on CPU time: CCOMB compared with the hybrid Dai–Yuan (*hDYz*) algorithm

it achieved the minimum number of iterations in 324 problems), the PRP algorithm was better in 196 problems and they achieved the same number of iterations in 191 problems, etc. Out of 750 problems, only for 711 problems does criterion (30) holds. Similarly, in Fig. 3 we see the number of problems for which the CCOMB algorithm was better than the DY algorithm. Observe that the convex combination of the PRP and DY algorithms, expressed as in (7), is far more successful than the PRP or the DY algorithm.

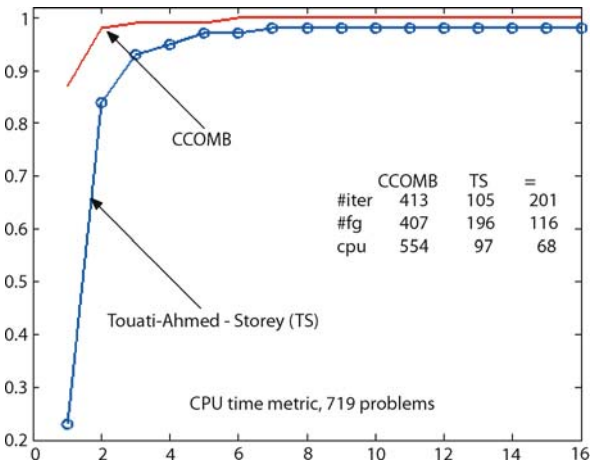
The third set of numerical experiments refers to the comparisons of the CCOMB algorithm with hybrid conjugate gradient algorithms: hDY, hDYz, GN, HuS, TS and LS-CD. Figures 4–9 presents the Dolan and Moré CPU performance profiles of these algorithms, as well as the number of problems solved by each algorithm in the minimum number of iterations, the minimum number of function evaluations and the minimum CPU time, respectively.

From these figures we see that the CCOMB algorithm is the top performer. Since these codes use the same Wolfe line search and the same stopping criterion they differ in their choice of the search direction. Hence, among these conjugate gradient algorithms we considered here, the CCOMB algorithm appears to generate the best search direction.

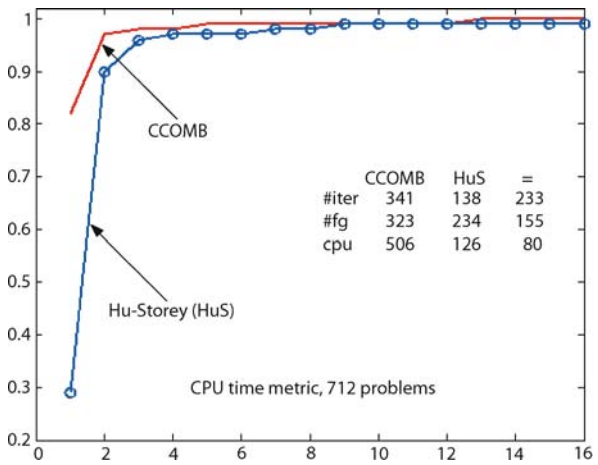
In the fourth set of numerical experiments we compared the CCOMB algorithm with the CG_DESCENT conjugate gradient algorithm of Hager and Zhang [18]. The computational scheme implemented in the CG_DESCENT algorithm is a modification of the HS method which satisfies the sufficient-descent condition, independent of the accuracy of the line search. The CG_DESCENT code, authored by Hager and Zhang, contains the variant CG_DESCENT (HZw) implementing the Wolfe line search and the variant CG_DESCENT (HZaw) implementing an approximate Wolfe line search. There are two main points associated



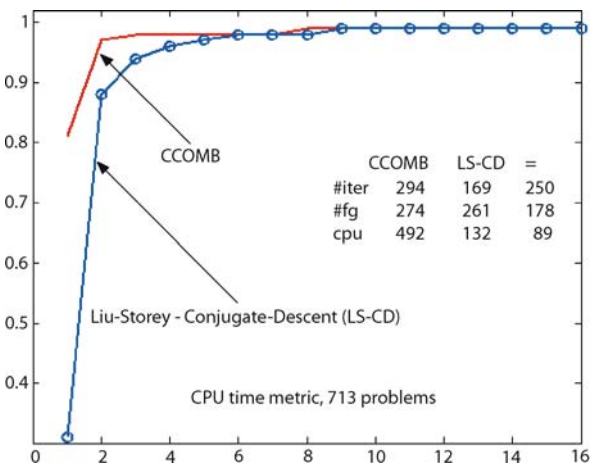
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 6
Performance based on CPU time: CCOMB compared with the Gilbert–Nocedal (*GN*) algorithm



New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 8
Performance based on CPU time: CCOMB algorithm compared with the Touati-Ahmed–Storey (*TS*) algorithm



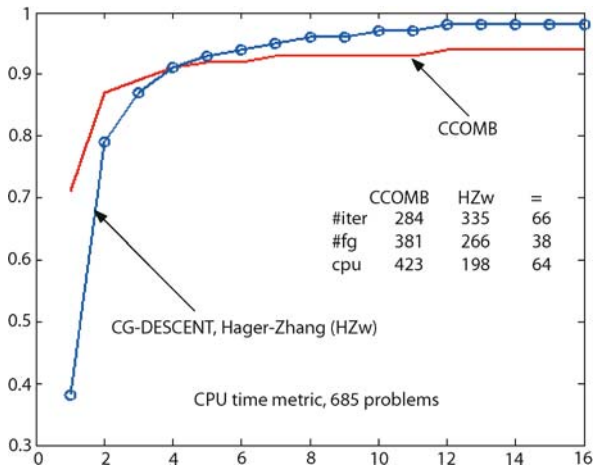
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 7
Performance based on CPU time: CCOMB algorithm compared with the Hu–Storey (*HuS*) algorithm



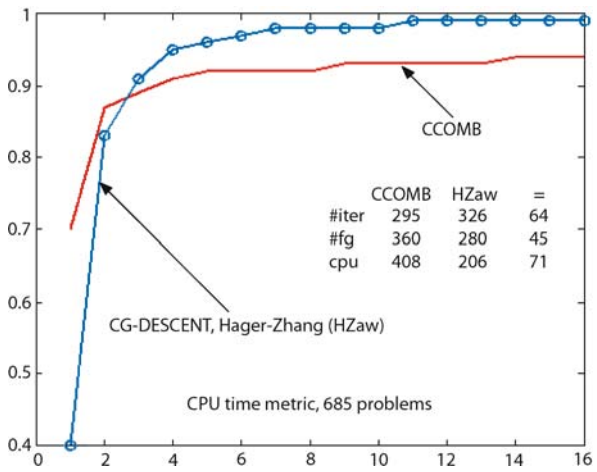
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 9
Performance based on CPU time: CCOMB algorithm compared with the Liu–Storey–conjugate descent (*LS-CD*) algorithm

with the CG_DESCENT algorithm. Firstly, the scalar products are implemented using the loop unrolling of depth 5. This is efficient for large-scale problems (over 10^6 variables). Secondly, the Wolfe line search is implemented using a very fine numerical interpretation of the first Wolfe condition (4). The Wolfe conditions implemented in the CCOMB and CG_DESCENT (HZw) algorithms can compute a solution with accuracy of the order of the square root of the machine epsilon.

In contrast, the approximate Wolfe line search implemented in the CG_DESCENT (HZaw) algorithm can compute the solution with accuracy of the order of machine epsilon. Figures 10 and 11 present the performance profile of these algorithms in comparison with that of the CCOMB algorithm. We see that the CG_DESCENT algorithm is more robust than the CCOMB algorithm.



New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 10
Performance based on CPU time: CCOMB algorithm compared with the CG_DESCENT algorithm with Wolfe line search (*HZw*)



New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 11
Performance based on CPU time: CCOMB algorithm compared with the CG_DESCENT algorithm with approximate Wolfe line search (*HZaw*)

Conclusion

We know a large variety of conjugate gradient algorithms. The known hybrid conjugate gradient algorithms are based on projection of classical conjugate gradient algorithms. In this chapter we have proposed new hybrid conjugate gradient algorithms in which the famous parameter β_k is computed as a convex combination of β_k^{PRP} and β_k^{DY}, i. e., $\beta_k = (1-\theta_k)\beta_k^{\mathrm{PRP}} + \theta_k\beta_k^{\mathrm{DY}}$. The parameter θ_k is computed in such a manner that the conjugacy condition is satisfied, or the corresponding direction in the hybrid conjugate gradient algorithm is the Newton direction. For uniformly convex functions if the step size s_k approaches zero, the gradient is bounded in the sense that $\eta_1 \|s_k\|^2 \leq \|g_k\|^2 \leq \eta_2 \|s_{k-1}\|$ and the line search satisfies the strong Wolfe conditions, then our hybrid conjugate gradient algorithms are globally convergent. For general nonlinear functions if the parameter θ_k from the β_k^{N} definition is bounded, i. e., $0 < \theta_k < 1$, then our hybrid conjugate gradient is globally convergent. The Dolan and Moré CPU performance profile of the hybrid conjugate gradient algorithm based on the conjugacy condition (CCOMB algorithm) is better than the performance profile corresponding to the hybrid algorithm based on the Newton direction (NDOMB algorithm). The performance profile of the CCOMB algorithm was better than those of the well-established conjugate gradient algorithms (hDY, hDYz, GN, HuS, TS and LS-CD) for a set consisting of 750 unconstrained optimization test problems, some of them from CUTE library. Additionally the proposed hybrid conjugate gradient algorithm CCOMB is more robust than the PRP and DY conjugate gradient algorithms. However, subject to robustness the CCOMB algorithm is outperformed by the CG_DESCENT algorithm.

References

1. Andrei N (2004) Test functions for unconstrained optimization. Available via http://www.ici.ro/camo/neculai/SCALCG/evalfg.for
2. Andrei N (2007) Scaled conjugate gradient algorithms for unconstrained optimization. Comput Optim Appl 38:401–416
3. Andrei N (2007) Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization. Optim Method Softw 22:561–571
4. Andrei N (2007) A scaled BFGS preconditioned conjugate gradient algorithm for unconstrained optimization. Appl Math Lett 20:645–650
5. Birgin E, Martínez JM (2001) A spectral conjugate gradient method for unconstrained optimization. Appl Math Optim 43:117–128
6. Bongartz I, Conn AR, Gould NIM, Toint PL (1995) CUTE: constrained and unconstrained testing environments. ACM Trans Math Softw 21:123–160

7. Dai YH (1997) Analysis of conjugate gradient methods. Ph.D. Thesis, Institute of Computational Mathematics and Scientific/Engineering Computing. Chinese Academy of Science, Beijing
8. Dai YH (2001) New properties of a nonlinear conjugate gradient method. Numer Math 89:83–98
9. Dai YH, Liao LZ (2001) New conjugacy conditions and related nonlinear conjugate gradient methods. Appl Math Optim 43:87–101
10. Dai YH, Liao LZ, Li D (2004) On restart procedures for the conjugate gradient method. Numer Algorithm 35:249–260
11. Dai YH, Han JY, Liu GH, Sun DF, Yin X, Yuan Y (1999) Convergence properties of nonlinear conjugate gradient methods. SIAM J Optim 10:348–358
12. Dai YH, Yuan Y (1999) A nonlinear conjugate gradient method with a strong global convergence property. SIAM J Optim 10:177–182
13. Dai YH, Yuan Y (2001) An efficient hybrid conjugate gradient method for unconstrained optimization. Annu Oper Res 103:33–47
14. Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. Math Program 91:201–213
15. Fletcher R (1987) Practical Methods of Optimization, vol 1: Unconstrained Optimization. Wiley, New York
16. Fletcher R, Reeves C (1964) Function minimization by conjugate gradients. Comput J 7:149–154
17. Gilbert JC, Nocedal J (1992) Global convergence properties of conjugate gradient methods for optimization. SIAM J Optim 2:21–42
18. Hager WW, Zhang H (2005) A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J Optim 16:170–192
19. Hager WW, Zhang H (2006) A survey of nonlinear conjugate gradient methods. Pac J Optim 2:35–58
20. Hestenes MR, Stiefel EL (1952) Methods of conjugate gradients for solving linear systems. J Res Nat Bur Stand 49:409–436
21. Hu YF, Storey C (1991) Global convergence result for conjugate gradient methods. J Optim Theory Appl 71:399–405
22. Liu Y, Storey C (1991) Efficient generalized conjugate gradient algorithms, Part 1: Theory. JOTA 69:129–137
23. Polak E, Ribière G (1969) Note sur la convergence de directions conjuguée. Rev Fr Inf Rech Oper, 3e Année 16:35–43
24. Polyak BT (1969) The conjugate gradient method in extreme problems. USSR Comp Math Math Phys 9:94–112
25. Powell MJD (1977) Restart procedures of the conjugate gradient method. Math Program 2:241–254
26. Powell MJD (1984) Nonconvex minimization calculations and the conjugate gradient method. In: Watson GA (ed) Numerical Analysis Proceedings, Dundee, 1983. Lecture Notes in Mathematics, vol 1066. Springer, Berlin, pp 122–141
27. Shanno DF, Phua KH (1976) Algorithm 500, Minimization of unconstrained multivariate functions. ACM Trans Math Softw 2:87–94
28. Touati-Ahmed D, Storey C (1990) Efficient hybrid conjugate gradient techniques. J Optim Theory Appl 64:379–397
29. Yuan Y (1993) Analysis on the conjugate gradient method. Optim Method Softw 2:19–29

Nonconvex Energy Functions: Hemivariational Inequalities

GEORGIOS E. STAVROULAKIS[1,2],
EURIPIDIS MISTAKIDIS[3]
[1] Carolo Wilhelmina Technische Universität, Braunschweig, Germany
[2] Technical University of Crete, Chania, Greece
[3] University Thessaly, Volos, Greece

MSC2000: 49J40, 70-XX, 80-XX, 49J52, 49Q10, 74K99, 74Pxx

Article Outline

Keywords
Clarke's Generalized Derivative
  Examples
Critical Points and Substationarity
Nonconvex Energy Functions
  Filling the Gaps in a Multifunction
  Superpotential Nonmonotone Laws
Hemivariational Inequalities
  Theoretical Studies
  Semicoercive Case
  Substationarity of the Energy
  Applications
  Numerical Algorithms
See also
References

Keywords

Hemivariational inequalities; Nonsmooth modeling; Generalized subdifferential of F.H. Clarke; Nonconvex energy function

A large number of problems in mechanics, engineering and economics can be defined by means of appropriate differentiation of an *energy function*. This function is sometimes called a *potential*. On the assumption of

differentiability one may write the first order optimality condition that the gradient of the function is equal to zero at some point. All points which satisfy this condition are called *critical points*. The set of nonlinear equations derived this way are the governing equations (the model) of the studied problem. Equivalently, one may consider the requirement that the directional derivative of the function at this critical point and in all directions emanating from this point is equal to zero, or that all possible small variations of the energy function around the critical point is equal to zero. This way the critical point is described by a *variational inequality* and one speaks about a variational formulation of the problem. Whether a critical point is also a minimum of the considered energy function, or, for example, a maximum or a saddle point requires the consideration of second order optimality conditions. One should mention in passing that (possibly local) minima are of outmost importance in applications. For instance, in mechanics they provide stable equilibria of the studied mechanical systems. Convexity and coercivity of the energy function guarantees that a critical point is a minimum while strict convexity ensures its uniqueness as well.

Lack of differentiability of the energy function, or the consideration of inequality constraints in the minimization problem changes the picture. As far as the convexity property holds, one may use the powerful tools of *convex analysis* to study the problem. The gradient of the nonsmooth but convex function is replaced by the subgradient, the differential by a set-valued *subdifferential* (in the sense of J.-J. Moreau and R.T. Rockafellar) and the critical point equation by the differential inclusion: zero must be an element of the subdifferential of the energy function at minimum. Accordingly, in the variational formulation a variational equation is replaced by a variational inequality. Analogous considerations hold true for the case of inequality constraints. Here, one has, in principle, two ways to study the problem. Either all admissible variations are taken into account in the derivation of the optimality conditions, or the inequality constraints are included by means of the indicator function of the admissible set in the set of the problem's variables. Following Moreau, who introduced and studied convex analysis and applied it for the solution of mechanical problems, a convex nondifferentiable potential energy function is called a *superpotential*.

Admittedly, convexity is a convenient assumption too good to be true in real life applications. The study of nonconvex energy functions requires new tools and methods, which are being developed within the area of nonconvex analysis. Among them, the notion of the generalized gradients in the sense of F.H. Clarke, Rockafellar [2] has found several applications. Concerning the search for minima of a nonconvex energy function one may formulate critical point problems. Under certain assumptions the generalized gradient of Clarke provides a useful tool for the formulation and the study of nonconvex and nonsmooth energy function problems. In the area of mechanics, P.D. Panagiotopoulos has been the first to introduce and use this notion. He called the resulting nonconvex variational inequalities *hemivariational inequalities*. Initially, the notion of Clarke, which is suitable for locally Lipschitz energy functions, has been used. Later, the extended notion of Rockafellar, which roughly speaking includes infinite vertical branches, has been considered. Later, he extended this analysis by incorporating inequality constraints or convex superpotential terms (as in the case of convex problems), which have been inspired from the engineering applications he studied. Thus, he formulated and studied *variational-hemivariational inequality* problems. On the analogy of convex analysis, the nonconvex and possibly nondifferentiable energy functions have been called by Panagiotopoulos *nonconvex superpotentials*. Once more, it should be emphasized here that hemivariational inequalities are not equivalent to minimum problems but to substationarity problems. Nevertheless, they constitute a consistent extension of variational inequalities, and they include them for the case of convex energy functions. Furthermore, and this is important for some numerical algorithms, as with the subdifferential of convex analysis the generalized subdifferential of Clarke involves one convex set in the set-valued approximation of the differential of a nonconvex and nonsmooth energy function. Consequently, the development of theory and algorithms runs in parallel with the ones developed for problems of convex analysis and of variational inequalities. It should be mentioned that propositions of substationary potential and complementary energy generalize the classical minimum energy propositions of mechanics (where, for historical reasons, they are known as principles). Moreover, following the example of *nonsmooth analy-*

sis, Panagiotopoulos named the whole area of mechanics which deals with nondifferentiable functions *nonsmooth mechanics*.

In this article the notions of generalized derivative of Clarke and Rockafellar and the definition of the substationarity points are first given. After that, the hemivariational and the variational-hemivariational inequalities of Panagiotopoulos are presented. Short comments on the theoretical tools which are used for their study, examples of applications and numerical algorithms which have been proposed for their solution complete the paper. Details on all previous issues can be found in the cited publications. See also ► hemivariational inequalities: applications in mechanics.

Clarke's Generalized Derivative

Let a function f be locally Lipschitz at $x \in X$ and let y be a vector in X. The *directional differential in the sense of Clarke* of f at x in the direction y, denoted by $f^0(x, y)$, is defined by the relation:

$$f^0(x, y) = \limsup_{\substack{\mu \to 0+ \\ h \to 0}} \frac{f(x + h + \mu y) - f(x + h)}{\mu}.$$

$f^0(x, y)$ is also called a *generalized directional differential*.

By means of the directional differential $f^0(x, y)$ one can now define the *generalized gradient* $\overline{\partial} f(x)$: $X \to X^*$:

$$\overline{\partial} f(x) = \left\{ x^* \in X^* : \begin{array}{c} f^0(x, x_1 - x) \geq (x^*, x_1 - x) \\ \forall x_1 \in X \end{array} \right\}. \tag{1}$$

One may also use the definition

$$\overline{\partial} f(x) = \left\{ x^* \in X^* : \ (x^*, -1) \in N_{\operatorname{epi} f}(x, f(x)) \right\}, \tag{2}$$

where $N_C(x)$ denotes the normal cone to a set C at point x and epi f is the epigraph of the function f.

Note that $\overline{\partial} f(\cdot)$ is a multivalued mapping; it is a nonempty convex, closed and bounded subset of $X^\star$ and the following relation holds true:

$$f^0(x, y) = \max \left\{ \langle y, x^* \rangle : \ x^* \in \overline{\partial} f(x) \right\}. \tag{3}$$

For didactical reasons the notation $\overline{\partial}$ is used here (and in most of the work of Panagiotopoulos). In honor of Clarke who proposed it, the notation ∂_{CL} is sometimes used. When misunderstanding is not expected, the notation ∂, which is usually reserved for the convex analysis subdifferential, is also used. It should also be noted here that $\overline{\partial}$ should not be confused with the superdifferential used in the theory of quasidifferentiability in the sense of V.F. Demyanov.

Relation (1) can be used to define the generalized gradient $\overline{\partial} f(x)$ for any type of function $f: X \to \overline{\mathbb{R}}$ which is finite at the point x. Note that $\overline{\partial} f(x)$ may be empty. The above definition of $\overline{\partial} f(x)$ for any function $f: X \to \overline{\mathbb{R}}$ makes sense, because the normal cone $N_C(x)$ can be defined with respect to any set epi f. The generalized directional differential $f^\uparrow(x; y)$ at x in the direction y is defined by the relation:

$$f^\uparrow(x, y) = \sup \left\{ \langle y, x^* \rangle : \ x^* \in \overline{\partial} f(x) \right\}. \tag{4}$$

Thus one may write the relation:

$$\overline{\partial} f(x) = \left\{ x^* \in X^* : \begin{array}{c} f^\uparrow(x, x_1 - x) \geq \langle x^*, x_1 - x \rangle \\ \forall x_1 \in X \end{array} \right\}. \tag{5}$$

The directional differential $f^\uparrow(x; y)$ is also called *directional differential in the sense of Rockafellar* [15]. Note that $\overline{\partial} f(x) = \emptyset$ if $f^\uparrow(x, 0) = -\infty$, while if $f^\uparrow(x, y)$ is finite for every y, then $\overline{\partial} f(x) \neq \emptyset$.

It should be noted that for a convex function f one has

$$f^\uparrow(x, y) = \liminf_{\widetilde{y} \to y} f'(x, \widetilde{y}), \quad \forall y \in X, \tag{6}$$

where $f'(\cdot, \cdot)$ denotes the one-sided directional Gâteaux differential. Moreover, for a locally Lipschitz function f at point x one has

$$f^\uparrow(x, y) = f^0(x, y), \quad \forall y \in X, \tag{7}$$

and for a continuously differentiable f:

$$\overline{\partial} f(x) = \{\operatorname{grad} f(x)\}. \tag{8}$$

Examples

For a convex function f one has

$$\overline{\partial} f(x) = \partial f(x), \tag{9}$$

and for a concave and bounded below on a neighborhood of x function f:

$$\overline{\partial} f(x) = -\partial(-f)(x) \tag{10}$$

at every point x where f is finite. The *indicator function* I_C of a (generally nonconvex) set C is defined by $I_C(x) = 0$ if $x \in C$, and $I_C(x) = \infty$ otherwise. It can be proved that

$$\overline{\partial} I_C(x) = N_C(x) \tag{11}$$

and

$$I_C^{\uparrow}(x, y) = I_{T_{C(x)}}(y). \tag{12}$$

In the finite-dimensional case $X \equiv \mathbf{R}^n$, for a locally Lipschitz function f at a point x, $\overline{\partial} f(x)$ is the convex hull of all points $y \in \mathbf{R}^n$ of the form

$$y = \lim_{i \to \infty} \operatorname{grad} f(x_i), \tag{13}$$

where x_i converges as $i \to \infty$ to x, avoiding the nondifferentiability points and any other points of a set of measure zero (in the sense of Lebesgue) and such that $\operatorname{grad} f(x_i)$ converges.

For a *maximum-type function* f, i. e., when the function is defined by means of continuously differentiable functions $\varphi_i = \varphi_i(x)$, $i = 1, \ldots, m$, $x \in \mathbf{R}^n$ by the relation: $f = \max\{\varphi_i, \ldots, \varphi_m\}$, one has

$$\overline{\partial} f(x) = \operatorname{co}\{\operatorname{grad} \varphi_i(x) : \ i \in I(x)\}, \tag{14}$$

where $I(x) = \{\mathrm{i}: \varphi_i(x) = f(x)\}$ is the active index set.

The *normal cone* to a set defined by: $C = \{x \in \mathbf{R}^n : f(x) \leq 0\}$ at a point x_0 with $f(x_0) = 0$ is described by the relation

$$N_C(x_0) \subset \left\{\lambda x^* : \ x^* \in \mathbb{R}^n, \ \lambda \geq 0, \ x^* \in \overline{\partial} f(x_0)\right\},$$

whenever f is Lipschitzian on a neighborhood of x_0 and $0 \notin \overline{\partial} f(x_0)$. The notion of $\overline{\partial}$-regularity assures that directional derivative information can be regained from the Clarke's notion. For a locally Lipschitz function one requires that $f^0(x, y) = f'(x, y)$, $\forall y \in X$, holds at x. This definition is equivalent to the statement that epi f is regular at $(x, f(x))$. For instance, a convex function and a maximum type function are $\overline{\partial}$-regular at a point x where they are finite. For example, for the max-type function $f = \max\{\varphi_1, \ldots, \varphi_m\}$ one has

$$N_C(x_0) = \overline{\partial} I_C(x_0) = \left\{ z = \sum_{i=1}^{m} \lambda_i \operatorname{grad} \varphi_i(x_0) : \begin{array}{c} \lambda_i \geq 0, \\ \varphi_i(x_0) \leq 0, \\ \lambda_i \varphi_i(x_0) = 0 \end{array} \right\}, \tag{15}$$

if $0 \notin \overline{\partial} f(x_0)$. The above relation permits the extension of the Lagrange multiplier rule for optimization problems subjected to the nonconvex inequality constraints $\varphi_i(x) \leq 0$, $i = 1, \ldots, m$. This becomes obvious, e. g. if one considers the search for a local minimum problem of a continuously differentiable function g: $\mathbf{R}^n \to \mathbf{R}$ over $C = \{x \in \mathbf{R}^n : \varphi_i(x) \leq 0, \ i = 1, \ldots, m\}$. A necessary condition is $0 \in \overline{\partial}(g + I_C)(x)$, which implies that

$$-\operatorname{grad} g(x) \in \overline{\partial} I_C(x), \tag{16}$$

which together with (15) leads to the Lagrange multiplier rule.

Critical Points and Substationarity

The notion of *substationarity* plays an important role in the theory of hemivariational inequalities because it permits the formulation of the propositions of substationary potential and complementary energy which generalize the corresponding classical minimum energy propositions in mechanics [12,13,14]. Point x_0 is a *substationarity point of a functional* $f : X \to \overline{\mathbb{R}}$ if

$$0 \in \overline{\partial} f(x_0). \tag{17}$$

Equivalently one has:

$$f^{\uparrow}(x_0, y) \geq 0, \quad \forall y \in X. \tag{18}$$

Substationarity points are all the classical stationarity points, all the local minima, a large class of local maxima, as well as all the saddle points. Point x is said to be a *substationarity point of f with respect to a set C*, if $f + I_C$ is substationary at x.

Nonconvex Energy Functions

The nonconvex superpotentials resulting by integrating discontinuous functions $\beta \in L^{\infty}_{\mathrm{loc}}(\mathbf{R})$ play an important

role in the formulation of hemivariational inequalities for several types of mechanical problems. After some technical details about filling in gaps in a multifunction, nonmonotone laws in mechanics which admit a nonconvex energy function will be introduced.

Filling the Gaps in a Multifunction

Suppose that β: $\mathbf{R} \to \mathbf{R}$ is a function such that $\beta \in L^{\infty}_{\text{loc}}(\mathbf{R})$, i. e. a function which is essentially bounded on any bounded interval of $\mathbf{R}$. For any $\rho > 0$ and $\xi \in \mathbf{R}$ let us define

$$\overline{\beta}_{\rho}(\xi) = \operatorname*{essinf}_{\|\xi_1 - \xi\| \le \rho} \beta(\xi_1) \tag{19}$$

and

$$\overline{\overline{\beta}}_{\rho}(\xi) = \operatorname*{esssup}_{\|\xi_1 - \xi\| \le \rho} \beta(\xi_1). \tag{20}$$

Obviously, the monotonicity properties of $\rho \to \overline{\beta}_{\rho}(\xi)$ and $\rho \to \overline{\overline{\beta}}_{\rho}(\xi)$ imply that the limits as $\rho \to 0_+$ exist. Therefore one may write that: $\overline{\overline{\beta}}(\xi) = \lim_{\rho \to 0_+} \overline{\overline{\beta}}_{\rho}(\xi)$. Furthermore, one defines the multivalued function:

$$\widetilde{\beta}(\xi) = \left[\overline{\beta}(\xi), \overline{\overline{\beta}}(\xi)\right], \tag{21}$$

where $[\cdot, \cdot]$ denotes an interval between the two given arguments. Then, a locally Lipschitz function j can be determined, up to an additive constant, by the relation

$$j(\xi) = \int_0^{\xi} \beta(\xi_1)\, d\xi_1, \tag{22}$$

such that $\overline{\partial} j(\xi) \subset \widetilde{\beta}(\xi)$. If moreover $\beta(\xi_{\pm})$ exist for each $\xi \in \mathbf{R}$, then one has $\overline{\partial} j(\xi) = \widetilde{\beta}(\xi)$.

Superpotential Nonmonotone Laws

Let us assume that one has an one-dimensional mechanical law which is described by the graph of a discontinuous function. For instance, a force-displacement law $(S - u)$ is considered, which may correspond to an one-dimensional nonlinear spring law, to a nonlinear boundary condition, etc. The law is considered to be of the form β: $u \to -S$ where $u \in \mathbf{R}$ and $S \in \mathbf{R}$. The procedure of (19)–(22) is used in order to define a locally Lipschitz nonconvex superpotential energy function $j(u)$. The mechanical law is produced, in turn, by using the previously introduced generalized subdifferential operator $\partial_{CL} = \overline{\partial}$ and the nonconvex superpotential j as follows:

$$-f \in \partial_{\text{CL}} j(u). \tag{23}$$

By definition, (23) is equivalent to the inequality

$$j^{\uparrow}(u, u^* - u) \ge \langle -f, u^* - u \rangle, \quad \forall u^* \in U, \tag{24}$$

for $u \in U$, which Panagiotopoulos has called a hemivariational inequality, and to the inclusion

$$(-f, -1) \in N_{\text{epi}\, j}(u, j(u)). \tag{25}$$

For j Lipschitzian, $j^{\uparrow}$ in (23) is replaced by j^0 Obviously, if j is a convex superpotential, one has superpotential laws which can be described by monotone graphs with complete vertical branches. The procedure would be identical in that case as well, where ∂_{CL} is replaced by the subdifferential of convex analysis. The result would be a variational inequality. Note also that extensions to multidimensional laws (e. g., material constitutive relations) can be considered within this formulation.

Hemivariational Inequalities

An abstract coercive hemivariational inequality is written first. Let V be a real Hilbert space with the property that $V \subset [L^2(\Omega)]^n \subset V^{\star}$, where $V^{\star}$ denotes the dual space of V, and the injections are continuous and dense. Let moreover a boundary value problem be defined in an open, bounded subset Ω of $\mathbf{R}^n$. Here $(\cdot, \cdot)$ denotes the $[L^2(\Omega)]^n$ inner product and the duality pairing, $\|\cdot\|$ is the norm of V and $|\cdot|_2$ is the $[L^2(\Omega)]^n$-norm. One should recall that the form $(\cdot, \cdot)$ extends uniquely from $V \times L^2\ [(\Omega)]^n$ to $V \times V^{\star}$. Moreover let L: $V \to L^2(\Omega)$, $Lu = \widehat{u}$, $\widehat{u}(x) \in \mathbb{R}$ be a linear continuous mapping. Further, assume that $l \in V^{\star}$, that L: $V \to L^2(\Omega)$ and that $\widetilde{V} = \left\{v \in V : \widehat{v} \in L^{\infty}(\Omega)\right\}$ is dense in V for the V-norm, and has a Galerkin base in V. It is also assumed that $a(\cdot, \cdot)$: $V \times V \to \mathbf{R}$ is a bilinear symmetric continuous form which is *coercive*, i. e. there exists a constant c

> 0 such that

$$a(v, v) \geq c \|v\|^2, \quad \forall v \in V. \tag{26}$$

A *coercive hemivariational inequality problem* reads: Find $u \in V$ such that

$$a(u, v-u) + \int_{\Omega} j^0(\widehat{u}, \widehat{v} - \widehat{u})\, d\Omega \geq (l, v-u), \quad \forall v \in V. \tag{27}$$

For example, a linear elastostatic structural analysis problem with additional nonlinear elements of nonmonotone type which admit a nonconvex superpotential $j(u)$ can be written in the hemivariational inequality form (25). In this context u are the displacements at the various points of the structure, which occupies Ω in its undeformed configuration. For a plate problem one has $\Omega \subset \mathbf{R}^2$, for a three-dimensional continuum $\Omega \subset \mathbf{R}^3$, etc. Moreover the functional space V is dictated from the kind of the assumed application and from the (natural, support) boundary conditions of the structure. The operator L and the energy form (u, u) depend on the mechanical theory used for the elastic part of the structure, while l denotes the external loading. Finally, coercivity usually means that a well-defined mechanical theory has been assumed and sufficient boundary conditions are assigned so that, for example, no rigid body motions of the structure are allowed. Finally, one should note that the familiar form of the principle of virtual work (i. e., a variational equality) can be obtained back from the hemivariational inequality (27) if the effect of the nonlinear terms is neglected, i. e., if the second term on the left-hand side of (27) is absent and if an equality is assumed in the place of the inequality.

Theoretical Studies

One recalls that the theory of the existence of solution of variational inequalities is a well-developed theory in mathematics which is closely connected with the convexity of the energy functionals involved. Indeed the existence theory of variational inequalities is based on monotonicity arguments. On the contrary the study of hemivariational inequalities, due to the absence of convexity is based on compactness arguments. Existence and approximation results for hemivariational inequalities have been studied for several applications. See [11,14] for details and citations to related references.

Semicoercive Case

Noncoercive problems arise in mechanics, for example, when the existing classical boundary conditions are not sufficient to prevent (even infinitesimal) rigid body displacements and rotations of the structure. In classical, equality mechanics, one may write conditions that guarantee the existence of a solution to the considered boundary value problem. Nevertheless, uniqueness of some quantities may be lost. One may consider, as an example, a free elastic body subjected to self-equilibrated external forces. In this case stresses and deformations of the elastic body can be determined, but its displacements are only determined modulo some rigid body displacements and rotations. In the presence of inequality (e. g., unilateral contact) constraints analogous relations have also been provided by G. Fichera (see, e. g., [13, Chap. 4]). It is interesting to observe that some inequality constraints may be activated and stabilize the body, a result that has certain applications in robotics [1].

For hemivariational inequalities, analogous results have been obtained by Panagiotopoulos and coworkers (see also [4]). One such result for an abstract hemivariational inequality is given here, without proof. Here $a(\cdot, \cdot)$ is assumed to be *semicoercive*, i. e., $a(\cdot, \cdot)$ is continuous and symmetric but it has a nonzero kernel, i. e. $\ker a(\cdot, \cdot) = \{q: a(q, q) = 0\} \neq \{0\}$. Moreover let $\ker a$ be finite dimensional. Let us assume that the norm $\|v\|$ on V is equivalent to $\|\|v\|\| = p(\widetilde{v}) + \|q\|_2$, where $v = \widetilde{v} + q$, $q \in \ker a$, $\widetilde{v} \in \ker a^{\perp}$ (i. e. $(\widetilde{v}, q) = 0$, $\forall\, q \in \ker a$) and $p(\widetilde{v})$ is a seminorm on V such that $p(v) = p(v + q)$, $\forall\, v \in V$, $q \in \ker a$, and let $a(v, v) \geq c(p(v))^2$, $\forall v \in V$, $c = \text{const} > 0$. This semicoercivity inequality replaces (26) for the coercive case. Moreover, let q_+ and q_- be the positive and negative parts of $\widehat{q}$, where $\widehat{q} = Lq$, i. e. $q_+ = \max\{0, \widehat{q}\}$, $q_- = \max\{0, -\widehat{q}\}$. The following quantities will also be used: $\beta(-\infty) = \lim \sup_{\xi \to -\infty} \beta(\xi)$ and $\beta(\infty) = \lim \inf_{\xi \to \infty} \beta(\xi)$. On the assumption that

$$\beta(-\infty) \leq \beta(\xi) \leq \beta(\infty), \quad \forall \xi \in \mathbb{R}, \tag{28}$$

a necessary condition for the existence of a solution $u \in V$ of a semicoercive hemivariational inequality problem

is the following inequality:

$$\int_\Omega [\beta(-\infty)q_+ - \beta(\infty)q_-]\, d\Omega \le (l, q) \le \int_\Omega [\beta(\infty)q_+ - \beta(-\infty)q_-]\, d\Omega, \quad \forall q \in \ker a. \tag{29}$$

If (28) holds strictly (with $<$ instead of $\le$), then (29) also holds strictly.

Substationarity of the Energy

Equivalent to the hemivariational inequality is the following *substationarity problem*: Find $u \in V$ such that the superpotential energy functional

$$\Pi(v) = \frac{1}{2}a(v, v) + \int_\Omega j(\widehat{v})\, d\Omega - (l, v) \tag{30}$$

is substationary at $v = u$. Here, the integral $\int_\Omega j(\widehat{v})\, d\Omega$ is set equal to ∞ if it is not defined.

Recall that the latter problem is, by definition, equivalent to: Find $u \in V$ which is a solution of the inclusion

$$0 \in \partial\Pi(u). \tag{31}$$

The equivalence of the above defined substationarity problem with the hemivariational inequality can be proved, on the assumption that j is locally Lipschitz and ∂-regular

Applications

Several types of hemivariational inequalities have already been studied (see, for example, [13,14], and the references given there) with respect to certain engineering problems, e. g. in nonmonotone semipermeability problems, in the theory of multilayered plates (delamination), in the theory of composite structures, in the theory of partial debonding of adhesive joints etc.

Numerical Algorithms

Till now (1998), the following methods have been investigated for the numerical solution of hemivariational inequality problems in mechanics.

- Nonsmooth optimization algorithms (in particular, of the *bundle algorithm* optimization type), for the solution of the inclusion (31). See [6,9] and, for the bundle nonsmooth optimization concept, e. g., [7].
- Decomposition into a series of convex subproblems (for instance, into a number of variational inequalities). For the numerical treatment of problems in elastostatics this approach has been the most fruitful, according to the experience of the authors. The decomposition has been based either on engineering motivated methods and heuristics (cf., [8]), or on mathematical programming techniques. In the last case results from the theory of quasidifferentiability [3], of difference-convex optimization [10] and of enumeration-type or branch and bound procedures [16] have been tested. More details can be found in [3,5,10,14].

See also

- ▶ Bariational Inequalities
- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex-Nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities

- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Al-Fahed AM, Stavroulakis GE, Panagiotopoulos PD (1992) A linear complementarity approach to the frictionless gripper problem. Internat J Robotics Res 11(2):112–122
2. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, Hoboken. reprinted as Classics Appl. Math. (1990) vol 5. SIAM, Philadelphia
3. Dem'yanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
4. Goeleven D (1996) Noncoercive variational problems and related results. Addison-Wesley, Indianapolis
5. Haslinger J, Stavroulakis GE (eds) (2006) Nonsmooth Mechanics of Solids. CISM Courses and Lectures, vol 485. Springer, Wien, New York
6. Haslinger J, Miettinen M, Panagiotopoulos PD (1999) Finite element method for hemivariational inequalities. Kluwer, Dordrecht
7. Hiriart-Urruty J-B, Lemaréchal C (1993) Convex analysis and minimization algorithms. Springer, Berlin
8. Koltsakis EK, Mistakidis ES, Tzaferopoulos MA (1995) On the numerical treatment of nonconvex energy problems in mechanics. J Glob Optim 6:427–448
9. Miettinen M, Mäkelä MM, Haslinger J (1995) On numerical solution of hemivariational inequalities by nonsmooth optimization methods. J Glob Optim 8(4):401–425
10. Mistakidis ES, Stavroulakis GE (1998) Nonconvex optimization in mechanics. Algorithms, heuristics and engineering applications by the F.E.M. Kluwer, Dordrecht
11. Naniewicz Z, Panagiotopoulos PD (1995) Mathematical theory of hemivariational inequalities and applications. Dekker, London
12. Panagiotopoulos PD (1983) Nonconvex energy functions. Hemivariational inequalities and substationary principles. Acta Mechanics 42:160–183
13. Panagiotopoulos PD (1985) Inequality problems in mechanics and applications. Convex and nonconvex energy functions. Birkhäuser, Basel. Russian translation, MIR, Moscow
14. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Heidelberg
15. Rockafellar RT (1981) The theory of subgradients and its applications to problems of optimization: Convex and nonconvex functions. Heldermann, Berlin
16. Tzaferopoulos MA, Liolios A (1997) On a branch and bound algorithm for the solution of a family of softening material problems of mechanics with applications to the analysis of metallic structures. J Glob Optim 11:133–149

Nonconvex Network Flow Problems

NNFP

BRUCE W. LAMAR
Economic and Decision Analysis Center,
The MITRE Corp., Bedford, USA

MSC2000: 90C26, 90C30, 90C35, 90B10

Article Outline

Keywords
See also
References

Keywords

Economy of scale; Global optimization; Nonconvex programming; Flows in networks; Fixed charge networks; Network design

The basic *network flow problem* can be formulated as

$$\begin{cases} \min & \displaystyle\sum_{(i,j)\in A} \phi_{ij}(x_{ij}) \\ \text{s.t.} & \displaystyle\sum_{(n,j)\in A} x_{nj} - \sum_{(i,n)\in A} x_{in} = b_n, \\ & \qquad \forall n \in N, \\ & l_{ij} \le x_{ij} \le u_{ij}, \quad \forall (i,j) \in A, \end{cases} \tag{1}$$

where A is the (directed) arc set with generic element (i, j); N is the node set with generic element n; b_n is the *net supply* (if $b_n > 0$) or *net demand* (if $b_n < 0$) at node n; u_{ij} (respectively, l_{ij}) is the flow upper (respectively, lower) bound for arc (i, j); x_{ij} is the *flow decision variable* for arc (i, j); and ϕ_{ij} (x_{ij}) is the *cost function* for arc (i, j).

It is assumed that $\sum_{n \in N} b_n = 0$ (otherwise, a dummy supply node or demand node can be added to N to enforce this condition). The *objective function* in (1) minimizes total costs in the network; the first constraints are *node flow balance equations*; and the second constraints are *arc flow bounds*. Extensions to the basic network flow problem given above include *multicommodity networks*, *generalized networks* (i. e., networks with arc flow gains or losses), and *augmented networks* (i. e., formulations with additional constraints and/or decision variables in addition to x_{ij}) (see, for example, [1]).

If the cost function $\phi_{ij}(x_{ij})$ is a *nonconvex function* for one or more of the arcs in set A, then (1) is referred to as a *nonconvex network flow problem* (NNFP). The most commonly used cost function in a NNFP is a *fixed charge function*, of the form

$$\phi_{ij}(x_{ij}) = \begin{cases} 0 & \text{if } x_{ij} = 0, \\ \alpha_{ij} + \beta_{ij} \cdot x_{ij} & \text{if } x_{ij} > 0, \end{cases} \tag{2}$$

where α_{ij} and β_{ij} are coefficients with $\alpha_{ij} \geq 0$ (and $l_{ij} = 0$). By incurring the quantum of cost α_{ij} before flow can be carried on arc (i, j), fixed charge NNFPs can be used to model a variety of network design and expansion [29], lot sizing [17], and facility location [6,30] problems. Classical combinatorial problems, such as traveling salesperson and 0–1 knapsack problems, can also be represented as fixed charge NNFPs. In fact, *any* mathematical programming problem that can be formulated as an integer program with integer coefficients can be recast as a fixed charge NNFP [28].

Another common form of cost function in a NNFP is a concave quadratic function, of the form

$$\phi_{ij}(x_{ij}) = \alpha_{ij} + \beta_{ij} \cdot \left(x_{ij} - \gamma_{ij}\right)^2, \tag{3}$$

where α_{ij}, β_{ij}, and γ_{ij} are coefficients with $\beta_{ij} \leq 0$. Concave quadratic NNFPs are used to model arc flow economies of scale which are present in many communication [32], water resource [10], and physical distribution [22] problems.

A third type of cost function used in NNFPs is the 'sawtooth' function. A simple two-piece sawtooth function is given by

$$\phi_{ij}(x_{ij}) = \begin{cases} \alpha_{ij} \cdot x_{ij} & \text{if } x_{ij} < \gamma_{ij}, \\ \beta_{ij} \cdot x_{ij} & \text{if } x_{ij} \geq \gamma_{ij}, \end{cases} \tag{4}$$

where α_{ij}, β_{ij}, and γ_{ij} are coefficients with $\alpha_{ij} \geq \beta_{ij}$ and $l_{ij} \leq \gamma_{ij} \leq u_{ij}$. Functions of the form (4) are used to represent price-breaks and other types of all-units discounting [7] that occur, for instance, in network representations of inventory [33] and cash flow management problems [24].

The NNFP is in the class of NP-hard problems [14]. Thus, determining the *global minimum* to an NNFP is challenging because of the existence of many feasible points that are locally, but not globally, optimal. G.M. Guisewite [18] distinguishes between NNFPs with concave cost functions (such as (2) and (3)) and indefinite cost functions (that is, functions that are neither concave nor convex, such as (4)). For concave NNFPs, if the constraints in (1) are feasible, then there exists an optimal extreme point solution. Further, if the coefficients b_n, l_{ij}, and u_{ij} in (1) are integer-valued, then the optimal arc flows x^*_{ij} will also be integer-valued [4].

One of the earliest solution methods for concave NNFPs was proposed by B. Yaged [32]. This method uses successive linearizations of the concave cost function. It quickly converges to a local (but not necessarily global) minimum. Other approximate methods for concave NNFPs involve dual ascent [2,9], Lagrangian relaxation [11], local extreme point search techniques [13,21], and tabu search methods [16].

Exact methods for concave NNFPs generally rely on the underlying network topology. For problems with an arbitrary network topology, branch and bound procedures using rectangular partitioning are the most widely used techniques [3,5,26,30]. If the number of supply nodes or demand nodes in the set N is limited, then dynamic programming approaches are tractable [8,17,20]. Alternatively, if there are only a small number of arcs in the set A with a nonlinear cost function, then parametric programming techniques may be employed [25].

For the second type of NNFP – with an indefinite form of cost function – the optimal solution to (1) is not necessarily at an extreme point of the feasible region. For cases where the indefinite cost function is piecewise-linear but not necessarily continuous (as is the case in (4)), then the problem can be formulated and solved as a mixed integer program. If the indefinite function is continuous (and satisfies certain regularity properties), then it can be converted to a difference convex function (d.c. function) and the indefinite NNFP

can be solved as a specialized difference convex programming problem [31]. For more general functions, the indefinite NNFP can be converted (at least in principle) to a concave NNFP on an expanded network. Then, the concave NNFP solution techniques can be applied the problem on the expanded network [27].

See [12,15,18,19,22,23] for additional discussion of applications and solution methods for NNFPs.

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Global Supply Chain Models
- ▶ Inventory Management in Supply Chains
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Multicommodity Flow Problems
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonoriented Multicommodity Flow Problems
- ▶ Operations Research Models for Supply Chain Management and Design
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs
2. Balakrishnan A, Magnanti TL, Wong RT (1989) A dual-ascent procedure for large scale uncapacitated network design. Oper Res 37:716–740
3. Bell GB, Lamar BW (1997) Solution methods for nonconvex network problems. In: Pardalos PM, Hearn DW, Hager WW (eds) Network Optimization. Lectures Notes Economics and Math Systems. Springer, Berlin, pp 32–50
4. Charnes A, Cooper WW (1961) Management models and industrial applications of linear programming. Wiley, New York
5. Cruz FRB, Macgregor Smith J, Mateus GR (1998) Solving to optimality the uncapacitiated fixed-charge network flow problem. Comput Oper Res 25:67–81
6. Daskin MS (1995) Network and discrete location. Wiley, New York
7. Dolan RJ (1987) Quantity discounts: Managerial issues and research opportunities. Marketing Sci 6:1–22
8. Erickson RE, Monna CL, Veinott AF (1987) Send-and-split method for minimum concave-cost network flows. Math Oper Res 12:634–664
9. Erlenkotter D (1978) A dual-based procedure for uncapacitated facility location. Oper Res 26:992–1009
10. Feltenmark S, Lindberg PO (1997) Network methods for head-dependent hydro power scheduling. In: Pardalos PM, Hearn DW, Hager WW (eds) Network Optimization, Lectures Notes Economics and Math Systems. Springer, Berlin, pp 249–264
11. Fisher ML (1985) An applications oriented guide to Lagrangian relaxation. Interfaces 15:10–21
12. Florian M (1986) Nonlinear cost network models in transportation analysis. In: Gallo G, Sandi C (eds) Netflow at Pisa, Math Program Stud. North-Holland, Amsterdam, pp 167–196
13. Gallo G, Sandi C, Sodini C (1980) An algorithm for the min concave cost flow problem. Europ J Oper Res 4:248–255
14. Garey MR, Johnson DS (1979) Computers and intractability. A guide to the theory of NP-completeness. Freeman, New York
15. Glover F, Klingman D, Phillips NV (1992) Network models in optimization and their applications in practice. Wiley, New York
16. Glover F, Laguna M (1997) Tabu search. Kluwer, Dordrecht
17. Graves SC, Orlin JB (1985) A minimum concave-cost dynamic network flow problem with applications to lot sizing. Networks 15:59–71
18. Guisewite GM (1995) Network models. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 609–648
19. Guisewite GM, Pardalos PM (1990) Minimum concave-cost network flow problems: Applications, complexity, and algorithms. Ann Oper Res 25:75–100
20. Guisewite GM, Pardalos PM (1991) Algorithms for the single-source uncapacitated minimum concave-cost network flow problem. J Global Optim 1:245–265
21. Guisewite GM, Pardalos PM (1992) Performance of local search in minimum concave-cost network flow problems. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton Univ. Press, Princeton, pp 50–75
22. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht

23. Jensen PA, Barnes JW (1980) Network flow programming. Wiley, New York
24. Jorjani S, Lamar BW (1994) Cash flow management network modelling with quantity discounting. OMEGA Internat J Management Sci 22:149–155
25. Klinz B, Tuy H (1993) Minimum concave-cost network flow problems with a single nonlinear arc cost. In: Du D-Z, Pardalos PM (eds) Network Optimization Problems: Algorithms, Applications, and Complexity. World Sci., Singapore, pp 125–146
26. Lamar BW (1993) An improved branch and bound algorithm for minimum concave cost network flow problems. J Global Optim 3:261–287
27. Lamar BW (1993) A method for solving network flow problems with general nonlinear arc costs. In: Du D-Z, Pardalos PM (eds) Network Optimization Problems: Algorithms, Applications, and Complexity. World Sci., Singapore, pp 147–167
28. Lamar BW (1996) A note on formulating nonconvex network optimisation problems. Techn Report Dept Management Univ Canterbury
29. Magnanti TL, Wong RT (1984) Network design and transportation planning: models and algorithms. Transport Sci 18:1–55
30. Soland RM (1974) Optimal facility location with concave costs. Oper Res 22:373–382
31. Tuy H (1995) D.C. optimization: Theory, methods, and algorithms. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 149–216
32. Yaged B (1971) Minimum cost routing for static network models. Networks 1:139–172
33. Zangwill WI (1968) Minimum concave cost flows in certain networks. Managem Sci 14:429–450

Nonconvex-nonsmooth Calculus of Variations

DANIEL GOELEVEN[1], DUMITRU MOTREANU[2]

[1] I.R.E.M.I.A. University de la Réunion, Saint-Denis, France

[2] Department Mat., University Al.I.Cuza, Iasi, Romania

MSC2000: 49J40

Article Outline

Keywords
See also
References

Keywords

Variational-hemivariational inequality; Generalized critical point; Unilateral mechanics

Problems in *unilateral mechanics* involving both monotone and nonmonotone unilateral boundary conditions have as variational formulations the so-called variational-hemivariational inequalities [5]. Solutions of these mathematical models can be determined as the critical points of some energy functionals which can usually be expressed as the sum of a locally Lipschitz function $\Phi\colon X \to \mathbf{R}$ and a proper, convex and lower semicontinuous function $\Psi\colon X \to \mathbf{R} \cup \{+\infty\}$. The critical points of $I := \Phi + \Psi$ are here defined as the solutions of the *variational-hemivariational inequality*:

$$\begin{aligned} &u \in X : \\ &\Phi^o(u; v-u) + \Psi(v) - \Psi(u) \geq 0, \quad \forall v \in X . \end{aligned} \tag{1}$$

For example, let us consider a linear elastic body identified with an open bounded subset $\Omega \subset \mathbf{R}^3$. The boundary Γ of the body Ω is assumed to consist of three open disjoint parts Γ_1, Γ_2 and Γ_3, i. e. $\Gamma = \overline{\Gamma}_1 \cup \overline{\Gamma}_2 \cup \overline{\Gamma}_3$. Let us denote by $u = (u_i)$, $\sigma = (\sigma_{ij})$, $\varepsilon = (\varepsilon_{ij})$, $S = (S_i)$ the displacement vector, the stress tensor, the strain tensor and the stress vector, respectively. It is supposed that the body is subject to a body density force $f \in L^2(\Omega; \mathbf{R}^3)$. Moreover, one compels the part Γ_1 of the body to satisfy the constraint

$$u(x) \in Q(x), \quad \forall x \in \Gamma_1 , \tag{2}$$

where $Q(x)$ is defined as follows:

$$Q(x) := \left\{ y \in \mathbf{R}^3 : \ g(x, y) \leq 0 \right\} , \tag{3}$$

where $g\colon \Gamma_1 \times \mathbf{R}^3 \to \mathbf{R}$ is continuous and convex in the second variable. Moreover, we assume that $g(x, 0) \leq 0$, $\forall x \in \Gamma_1$. Thus $Q(x)$ is a nonempty, closed and convex subset for all $x \in \Gamma_1$. The formulation of these unilateral constraints has to encompass the associated forces of constraints $r(x)$. We assume a normal reaction law of the form

$$-r(x) \in N_{Q(x)}(u(x)), \quad \forall x \in \Gamma_1 , \tag{4}$$

where for a vector $v \in \mathbf{R}^3$, $N_{Q(x)}(v)$ denotes the normal cone of $Q(x)$ at v. Note that the system (2–4) is equivalent to the variational inequality

$$\begin{aligned} &u(x) \in Q(x) : \\ &r(x)^\top (v - u(x)) \geq 0, \quad \forall v \in Q(x), \quad \forall x \in \Gamma_1 . \end{aligned} \tag{5}$$

The part Γ_2 (it is assumed that meas$(\Gamma_2) > 0$) of the boundary is assumed to be fixed, that is,

$$u(x) = 0, \quad \forall x \in \Gamma_2 . \tag{6}$$

On Γ_3, we consider conditions of a unilateral contact between the body and a Winkler-type support which may sustain only limited values of traction, that is (we consider the usual decompositions $v = v_N n + v_T$, $S = S_N n + S_T$, v_N, $S_N \in \mathbf{R}$, v_T, $S_T \in \mathbf{R}^3$, where n denotes the unit outward normal vector on Γ):

$$-S_N \in \partial j(x, u_N(x)), \quad \forall x \in \Gamma_3 , \tag{7}$$

where

$$j(x, y) := \begin{cases} 0 & \text{if} \quad y < 0, \\ \frac{k_o(x) y^2}{2} & \text{if} \quad 0 \le y < \varepsilon, \\ \frac{k_o(x) \varepsilon^2}{2} & \text{if} \quad y \ge \varepsilon, \end{cases}$$

and

$$S_T = C_T \quad \text{on} \quad \Gamma_3 \tag{8}$$

with C_T given in $L^2(\Gamma_3; \mathbf{R}^3)$. We refer the reader to [4] for the details concerning the formulation of the subdifferential law (7). From (7), (8) and the orthogonality between $S_N n$ and u_T and between S_T and $u_N n$, we obtain the hemivariational inequality

$$S^\top v + j_y^o(x, u_N(x); v_N) - C_T^\top v_T \ge 0,$$
$$\forall v \in \mathbf{R}^3, \quad \forall x \in \Gamma_3 .$$

Here $\partial j(x, \cdot)$ stands for the *Clarke subdifferential* of j with respect to the second variable while $j_y^o(x, u; v)$ denotes the generalized directional derivative of j at u in the direction v [2]. In the framework of a small deformation theory, one has the equilibrium equations

$$\sigma_{ij,j} + f_i = 0 , \tag{9}$$

$$\varepsilon_{ij}(u) = \frac{1}{2}(u_{i,j} + u_{j,i}) , \tag{10}$$

$$\sigma_{ij} = C_{ijkl} \varepsilon_{kl}(u) , \tag{11}$$

where $C_{ijkl} \in L^\infty(\Omega)$ denotes the elasticity tensor assumed to satisfy the usual symmetry and ellipticity properties. Suppose now that all the data are sufficiently smooth to justify the following computations. From (9) and for a virtual displacement $v - u$ we obtain the equation

$$-\int_\Omega \sigma_{ij,j}(v_i - u_i)\, dx = \int_\Omega f^\top (v - u)\, dx.$$

Using the Green–Gauss theorem and relations (10) and (11), we get

$$\int_\Omega C_{ijkl} \varepsilon_{ij}(u) \varepsilon_{kl}(v - u)\, dx$$
$$= \int_\Omega f^\top (v - u)\, dx + \int_\Gamma S^\top (v - u)\, ds .$$

Using now the boundary conditions, we derive for v and u satisfying (6):

$$\int_\Omega C_{ijkl} \varepsilon_{ij}(u) \varepsilon_{kl}(v - u)\, dx$$
$$= \int_\Omega f^\top (v - u)\, dx + \int_{\Gamma_3} S_N (v_N - u_N)\, ds$$
$$+ \int_{\Gamma_3} C_T^\top (v_T - u_T)\, ds + \int_{\Gamma_1} r^\top (v - u)\, ds .$$

A combination of this last expression expressing the principle of virtual work with the mechanical laws (4) and (7), we obtain the variational-hemivariational inequality problem: Find u satisfying (2) and (6) such that

$$\int_\Omega C_{ijkl} \varepsilon_{ij}(u) \varepsilon_{kl}(v - u)\, dx$$
$$+ \int_{\Gamma_3} j^o(x, u_N; v_N - u_N)\, ds$$
$$- \int_\Omega f^\top (v - u)\, dx - \int_{\Gamma_3} C_T^\top (v_T - u_T)\, ds \ge 0 ,$$

for all v satisfying (2) and (6). This mathematical model expresses the principle of virtual works in its inequality form. Let us now present a suitable framework to formulate this last inequality model. The boundary Γ of the body is assumed sufficiently regular and we denote by γ: $H^1(\Omega, \mathbf{R}^3) \to H^{1/2}(\Gamma; \mathbf{R}^3)$, γ_N: $H^1(\Omega; \mathbf{R}^3) \to H^{1/2}(\Gamma)$ and γ_T: $H^1(\Omega; \mathbf{R}^3) \to H_T$ (see e. g., [6] for the notations) the usual trace mapping, normal trace mapping and tangential trace mapping, respectively. We set

$$X = \{u \in H^1(\Omega; \mathbf{R}^3) : \gamma(u(x)) = 0 \text{ a.e. } x \in \Gamma_2\}$$

and

$$C = \{u \in X : \gamma(u(x)) \in Q(x) \text{ a.e. } x \in \Gamma_1\} .$$

We define the operator $A: X \to X'$, the element $h \in X'$ and the functional $J: X \to \mathbf{R}$ by the formulas:

$$\langle Au, v\rangle := \int_\Omega C_{ijkl}(x)\varepsilon_{ij}(u)\varepsilon_{kl}(v)\ \mathrm{d}x, \quad \forall u, v \in X,$$

$$\langle h, v\rangle := \int_\Omega f_i v_i\ \mathrm{d}x + \int_{\Gamma_3} C_T^\top v_T\ \mathrm{d}s, \quad \forall v \in X,$$

$$J(v) := \int_{\Gamma_3} j(x, \gamma_N(v(x)))\ \mathrm{d}s, \quad \forall v \in X\,.$$

Using the rules concerning the subdifferentiation of integral functionals and composite mappings [2], we can show that

$$J^o(u, v) \le \int_{\Gamma_3} j^o(x, \gamma_N(u(x)); \gamma_N(u(x)))\ \mathrm{d}s, \quad \forall u, v \in X\,. \tag{12}$$

Setting

$$\Phi(u) = \frac{1}{2}\langle Au, u\rangle - \langle h, u\rangle + J(u)$$

and using the relation (12) we see that the equilibriums of our system can be determined by means of the variational-hemivariational inequality:

$$u \in C: \quad \Phi^o(u; v - u) \ge 0, \quad \forall v \in C,$$

or equivalently

$$u \in X: \quad \Phi^o(u; v - u) + \Psi_C(v) - \Psi_C(u) \ge 0, \quad \forall v \in X\,, \tag{13}$$

where Ψ_C denotes the indicator function of C. It is now natural to introduce a concept of *generalized critical point* of the energy functional $I = \Phi + \Psi_C$ as a solution of the variational-hemivariational inequality (13) (which is a particular case of the general model given in (1)).

Various other examples in mechanics leading to a variational-hemivariational inequality like (1) are described in [5]. As a rule, the formulation of concrete problems involving both monotone and nonmonotone boundary conditions introduces a general nonsmooth and nonconvex energy functional (expressed as the sum of a locally Lipschitz function $\Phi: X \to \mathbf{R}$ and a proper, convex and lower semicontinuous function $\Psi: X \to \mathbf{R} \cup \{+\infty\}$) whose critical points are defined as the solutions of the variational-hemivariational inequality (1). If $\Phi \in C^1(X; \mathbf{R})$ then problem (1) reduces to a classical variational inequality. A critical point theory to deal with such case has been developed by A. Szulkin [7]. Defining a compactness condition of Palais–Smale type related to the variational inequality model and using the famous Ekeland principle, Szulkin provided a deformation lemma and extended the well-known mountain pass theorem, the saddle point theorem, the main results for even functionals, etc. Several examples including variational inequalities are given in [7]. If $\Psi = 0$, the problem (1) reduces to the problem studied by K.-C. Chang [1]. Motivated by the study of partial differential equations involving discontinuous nonlinearities, Chang extended the concept of critical point, the Palais–Smale condition and the deformation lemma so as to be applicable to locally Lipschitz functionals. The minimax method developed by Chang is now one of the most efficient and appreciated tools to deal with hemivariational inequalities arising in unilateral mechanics and partial differential equations involving discontinuous nonlinearities. It has been in particular extensively used by D. Goeleven, D. Motreanu, and P.D. Panagiotopoulos [3] so as to develop in several directions the theory of hemivariational inequalities (see for instance the references cited in [3]). So, the theory of Szulkin and the one of Chang has been shown very efficient to deal with problems in unilateral mechanics involving some given classes of boundary conditions. However, one has seen it above, to deal efficiently with problems in unilateral mechanics involving both monotone and nonmonotone boundary conditions it is necessary to deal with a critical point theory for functions which can be written as the sum of a locally Lipschitz function and a proper, convex and lower semicontinuous function. Such theory has now recently been developed by Goeleven, Motreanu, and Panagiotopoulos [3]. The approach combines the approach of Szulkin and the one of Chang in a nontrivial way. A general linking theorem is obtained and then used to generalize the mountain pass theorem, the saddle point theorem and the main results for even functionals. This last theory constitutes a contribution in the field of nonconvex-nonsmooth calculus of variations that unifies the theory of Szulkin

and Chang. Moreover it can be used to develop the theory of variational-hemivariational inequalities and consequently to study various concrete problems in mechanics involving different types of unilateral boundary conditions.

See also

- ▶ Composite Nonsmooth Optimization
- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonsmooth and Smoothing Methods for Nonlinear Complementarity Problems and Variational Inequalities
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Chang K-C (1981) Variational methods for non-differentiable functionals and their applications to partial differential equations. J Math Anal Appl 80:102–129
2. Clarke FH (1984) Nonsmooth analysis and optimization. Wiley, New York
3. Goeleven D, Motreanu D, Panagiotopoulos PD (1997) General minimax methods for variational-hemivariational inequalities. Preprint
4. Naniewicz Z (1989) On some nonmonotone subdifferential boundary conditions in elastostatics. Ingen Archiv 60:31–40
5. Naniewicz Z, Panagiotopoulos PD (1995) The mathematical theory of hemivariational inequalities and applications. M. Dekker, New York
6. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Berlin
7. Szulkin A (1986) Minimax principles for lower semicontinuous functions and applications to nonlinear boundary value problems. Ann Inst Henri Poincaré 3:77–109

Nondifferentiable Optimization

Nonsmooth Optimization, NDO

Samir Elhedhli[1], Jean-Louis Goffin[1], Jean-Philippe Vial[2]

[1] GERAD/Fac. Management, McGill University, Montreal, Canada

[2] Logilab/Department Management Stud., University Genève, Geneva, Switzerland

MSC2000: 90-00, 90C47, 46N10

Article Outline

Keywords
Basic Definitions
Sources of NDO Problems
Solution Approaches
  Subgradient Methods
Steepest Descent and *E*-Subgradient Methods
  Cutting Plane Methods

Conclusion
See also
References

Keywords

Nondifferentiable optimization; Nonsmooth optimization

Nondifferentiable, or nonsmooth, optimization (NDO) is concerned with problems where the smoothness assumption on the functions involved is relaxed. 'Nondifferentiability' means that the gradient does not exist, implying that the function may have kinks or corner points. Consequently, the function cannot be approximated locally by a tangent hyperplane, or by a quadratic approximation. In NDO, the smoothness assumption is usually replaced by weaker ones, which at least guarantees the existence of directional derivatives.

NDO problems arise in a variety of contexts, and methods designed for smooth optimization may fail to solve them. This justifies developing specialized theory and methods that are the object of this short introduction. In the sequel, we will often refer to *convex NDO*, a subclass of nondifferentiable optimization, in which functions are further assumed to be convex. Due to its global property, convexity allows stronger convergence results and finer analyses. Yet, the difficulties linked with the presence of kinks remain an important aspect, justifying special interest for this class of problems.

In the following Section, we give some basic definitions, then discuss examples of nondifferentiable optimization problems and finally, describe a few different solution techniques.

Basic Definitions

The basic nondifferentiable optimization problem takes the form

$$[\text{NDP}] \quad \min_{x \in \mathbf{R}^n} f(x), \tag{1}$$

where f is a real valued, continuous, nondifferentiable function. Convexity of f implies that it has at least one supporting hyperplane at every point of $\mathbf{R}^n$. The slopes of such hyperplanes form the set of *subgradients*, which is known as the *subdifferential set* or the *generalized gradient* [7]. At differentiable points there is a unique supporting hyperplane whose slope is the gradient. At nondifferentiable points, there is an infinite set of subgradients and, hence, an infinite set of supporting hyperplanes.

A supporting hyperplane to f at a point x_0 is given by

$$y = f(x_0) + \xi_0^\top (x - x_0),$$

where ξ_0 is any element of the subdifferential $\partial f(x_0)$ of f at x_0. Recalling the fact that it is a supporting hyperplane leads to the *subgradient inequality*

$$f(x_0) + \xi_0^\top (x - x_0) \le f(x). \tag{2}$$

Subgradients are defined by this inequality.

Determining the whole subdifferential set is generally an extremely difficult, or impossible, task. If the function f is polyhedral, the number of extreme points of the subdifferential may be exponential in the dimension of the underlying space. A complete description of the subdifferential can be accomplished for simple situations, such as the one when f is the maximum of a finite number of convex differentiable functions: $f(x) = \max_{i \in I} f_i(x)$. The subdifferential $\partial f(x_0)$ is then given by

$$\partial f(x_0) = \left\{ \sum_{i \in I(x_0)} \alpha_i \nabla f_i(x_0) : \begin{array}{c} \sum_{i \in I(x_0)} \alpha_i = 1, \\ \alpha_i \ge 0 \end{array} \right\}$$

$$I(x_0) = \{ i : \ f_i(x_0) = f(x_0) \} \ .$$

When f is a Lipschitz function, the subdifferential set can be defined as being the set of cluster points of the gradients $\nabla f(x_i)$ as a sequence of differentiable points x_i approaches x [7]. The precise definition of $\partial f(x_0)$ is given by

$$\text{conv} \{ \lim \nabla f(x_i) : x_i \to x_0 : \ \nabla f(x_i) \text{ exists} \} \ .$$

In nondifferentiable optimization, the whole subdifferential set is never calculated. Subgradients are calculated when needed and often a single element suffices. It is common practice to isolate the procedures for calculating subgradients into an oracle. The number of calls to the oracle can be a basis for comparing different NDO methods.

A natural solution method in nonsmooth analysis is an iterative method, where a search is done following descent directions. A descent direction is one along

which a small movement of f leads to a strict improvement. In other words

$$f'(x_0; d) = \lim_{t \to 0} \frac{f(x + td) - f(x)}{t}$$

should be strictly negative. $f'(x_0;d)$ is called the *directional derivative* and it is related to the subgradient through the relation

$$f'(x_0; d) = \sup \left\{ \xi^\top d : \ \xi \in \partial f(x_0) \right\}. \tag{3}$$

This relation implies that for d to be a descent direction, $-d$ has to make an acute angle with every subgradient of f at x_0.

Sources of NDO Problems

Nonsmooth problems are encountered in many disciplines. In some instances, they occur naturally and in others they result from mathematical transformations.

In statistics, for example, rectilinear data fitting, which was long discovered to be superior to the Euclidean one – it has the advantage of overcoming the effect of outliers, [25] – results directly in an NDO problem. Similarly, functions involving ℓ_1 or ℓ_∞ norms, Euclidean or Chebyshev distances, a maximum of convex functions are typical NDO problems. As an example, the ℓ_∞ solution of an overdetermined linear system is found by solving the nondifferentiable convex function:

$$\min_{x \in \mathbf{R}^n} \|Ax - b\|_\infty = \min_{x \in \mathbf{R}^n} \max_{i=1,\ldots,m} \left| a_i^\top x - b_i \right|, \tag{4}$$

where $x \in \mathbf{R}^n$, $b \in \mathbf{R}^m$ and $A \in \mathbf{R}^{m \times n}$ with rows $a_i^\top$. This problem can be traced back to the Russian mathematician P.L. Chebyshev who studied it in the 1850s [25].

Among the mathematical transformations that lead to NDO problems is the technique that changes constrained problems into unconstrained ones through the use of exact penalty functions [11] (cf. also ► Quasidifferentiable optimization: Exact penalty methods). Equality constraints, $\phi(x) = 0$ and inequality constraints $\varphi(x) \leq 0$ are placed in the objective using penalty parameters and nondifferentiable functions $|\phi(x)|$ and $\max\{0, \varphi(x)\}$, respectively. In other word, a solution to the constrained problem

$$\begin{cases} \min & f(x) \\ \text{s.t.} & \phi(x) = 0, \\ & \varphi(x) \leq 0, \end{cases} \tag{5}$$

is determined by solving

$$\min f(x) + t_1 |\phi(x)| + t_2 \max\{0, \varphi(x)\}$$

for large enough values of t_1 and t_2.

Still, the major source of optimization problems are master problems resulting from the application of relaxation/restriction techniques such as Lagrangian relaxation [14] (cf. also ► Integer programming: Lagrangian relaxation), [12], Benders decomposition (cf. also ► Generalized Benders decomposition) [4,13] and Dantzig–Wolfe decomposition [9,10].

These different approaches are conceptually similar, at least in the linear case, and end up solving the same NDO problem. To show that, let us consider the linear program

$$[LP] \begin{cases} \min & c^\top x \\ \text{s.t.} & Ax \geq b, \\ & Dx \geq d, \\ & x \geq 0. \end{cases}$$

Where, we assume for the ease of exposition that $\{x: Ax \geq b; x \geq 0\}$ is a bounded, nonempty polytope. The dual of [LP] is

$$[LD] \begin{cases} \max & b^\top u + d^\top v \\ \text{s.t.} & A^\top u + D^\top v \leq c, \\ & u, v \geq 0. \end{cases}$$

Applying Lagrangian relaxation to [LP] is equivalent to relaxing $Dx \geq d$ using positive dual multipliers v, leading to

$$\max_{v \geq 0} \left\{ \min_{x \geq 0} c^\top x + v^\top (d - Dx) : \ Ax \geq b \right\}. \tag{6}$$

Benders decomposition applied to [LD] results in

$$\max_{v \geq 0} \left\{ \max_{u \geq 0} b^\top u + d^\top v : \ A^\top u \leq c - D^\top v \right\},$$

where v is assumed to be the complicating variable. Replacing the inside problem by its dual leads to

$$\max_{v \geq 0} \left\{ \min_{x \geq 0} c^\top x + v^\top (d - Dx) : \ Ax \geq b \right\}. \tag{7}$$

Dantzig–Wolfe decomposition replaces [LP] by its convex representation in terms of the convex points of

$\{x: Ax \geq b; x \geq 0\}$ that is indexed by $\mathcal{E}$, to get

$$\begin{cases} \min & \sum_{h\in\mathcal{E}} \alpha_h (c^\top x^h) \\ \text{s.t.} & \sum_{h\in\mathcal{E}} \alpha_h (Dx^h) \geq d, \\ & \sum_{h\in\mathcal{E}} \alpha_h = 1, \\ & \alpha_h \geq 0, \quad h \in \mathcal{E}. \end{cases}$$

Taking the dual results in

$$\begin{cases} \max\limits_{v\geq 0, v_0} & v_0 + d^\top v \\ \text{s.t.} & c^\top x^h + v^\top D x^h \geq v_0, \quad \forall h \in \mathcal{E}, \end{cases}$$

which is equivalent to

$$\max_{v\geq 0} \left\{ \min_{h\in\mathcal{E}} c^\top x^h + v^\top (d - Dx^h) \right\}. \tag{8}$$

The equivalence between (6) and (7) is obvious. Using the fact that there is always an extreme point solution to a linear program, the equivalence between (6), (7) and (8) is established. Therefore, Lagrangian relaxation applied to the primal is exactly Benders decomposition applied to the dual, and is equivalent by duality to Dantzig–Wolfe decomposition. Furthermore, all three solve (8), which is the maximum of a concave piecewise linear function that is nondifferentiable at intersection points.

F.H. Clarke [7] and [8] discusses further examples from physics, engineering, economics and optimal control. Other mathematical problems leading to NDO optimization include semi-infinite programming, eigenvalue optimization and variational inequalities [16].

Solution Approaches

Due to the existence of successful solution methods for differentiable optimization, one other solution approach tries to transform nonsmooth problem into smooth ones. As an example, the absolute value function $|x|$, which is nondifferentiable at zero can be approximated by

$$\begin{cases} -x, & x \geq t, \\ \frac{x^2}{t}, & -t \leq x \leq t, \\ x, & x \geq t, \end{cases}$$

for small values of the parameter t. For these transformations to be successful, the right transformation should be found and the the nondifferentiable points should be known. A solution approach based on this transformation is discussed in [21].

Other solution approaches that try to eliminate nondifferentiability, do so by transforming an unconstrained nonsmooth problem into a constrained smooth one. This approach is highly efficient for problems that can be transformed into easily solvable constrained problems such as linear programs. The ℓ_∞ optimization problem described in (4) is equivalent to the linear program

$$\begin{cases} \min & y \\ \text{s.t.} & Ax - ye \leq b, \\ & Ax + ye \geq b, \end{cases}$$

where e is the appropriate dimension vector whose entries are all ones. Being a linear program with a special type of matrix, most linear programming techniques were modified to solve (4). This includes the simplex-like algorithm in [3] and the interior point algorithms in [23] and [27].

Subgradient Methods

The first methods for nondifferentiable optimization tried to extend the gradient-based methods that were successful for smooth optimization. The transition from gradients to subgradients is not straightforward as some subgradient-based search direction are not necessarily improving directions. P. Wolfe [26] gives an example where the extension of the steepest descent method fails. To overcome that, some designed methods [18,20] will only take a serious step when the next iterate is a better one.

Subgradient methods (cf. also ► Nondifferentiable optimization: Subgradient optimization methods) were developed by N.Z. Shor [24] in the 1960s. They are basically an iterative technique where iterates are updated using a current subgradient and a carefully-chosen stepsize. Applied to (1), iterates are given by

$$x_{k+1} = x_k + t_k \xi_k,$$

where x_k is the current point, ξ_k is a subgradient of f at x_k and t_k is a stepsize. Shor [24] states that a constant

stepsize does not converge, even for the simple function $|x|$. He proposes the use of a stepsize that satisfies

$$\sum_{k=0}^{\infty} t_k = \infty, \quad t_k \to 0.$$

In practice, the most widely used stepsize is $\theta[f(x_k) - f^*]/\| \xi_k \|$ where $\theta \in (0, 2]$ and f^* is the best estimate of the optimal value $f(x^*)$.

Steepest Descent and *E*-Subgradient Methods

Subgradient methods are not monotonic, as they do not guarantee to improve the value of the minimized function. Descent methods are designed to overcome this drawback. As an example we discuss the steepest descent method which chooses its search direction by solving

$$\min_{\|d\| \le 1} f'(x; d).$$

Using relation (3), the steepest descent direction, at a point x_k, is given by

$$d_k = \frac{\xi_k}{\|\xi_k\|}; \quad \xi_k = \arg \max_{\xi \in \partial f(x_0)} \|\xi\| .$$

The method proceeds iteratively, updating the iterates by

$$x_{k+1} = x_k + \alpha_k \xi_k$$

and choosing the steplength α_k so that $f(x_{k+1}) < f(x_k)$.

The main difficulty with the steepest descent resides in the calculation of the direction d_k which necessitates the knowledge of the whole differential set $\partial f(x)$. To overcome that, *ϵ-subgradient methods* prefer to calculate approximate steepest descent direction by searching through subgradients of neighboring points through the use of the *ϵ-subdifferential set*

$$\partial_\epsilon f(x) = \{\xi : \ f(x_0) + \xi(x - x_0) + \epsilon \le f(x), \ \forall x\} .$$

Details of the method can be found in [5].

Cutting Plane Methods

J.E. Kelley [17] and W. Cheney and A.A. Goldstein [6] were the first to realize the potential of such methods for convex programming. Applied to (1), cutting plane algorithms use the subgradient inequality to approximate f by

$$f(x) \cong \max_{i \in I} f(x_i) + \xi_i^\top (x - x_i),$$

where ξ_i^f, $i \in I$ are subgradients of f at x_i, $i \in I$. Thus, (1) is replaced by

$$\min_x \left\{ \max_{i \in I} f(x_i) + \xi_i^\top (x - x_i) \right\},$$

which is equivalent to,

$$\begin{cases} \min & v \\ \text{s.t.} & f(x_i) + \xi_i^\top (x - x_i) \le v, \quad \forall i \in I. \end{cases} \tag{9}$$

Problem (9) is a linear program that is easier to deal with than the original problem. It is to note, however, that this is only an approximation of (1), which gets better as more constraints are added. Let us denote by [MP$_k$] the relaxed master problem (9) with index set I_k.

By transforming (1) to (9), a nondifferentiable problem is replaced by a constrained problem having a large number of constraints. Cutting plane methods use only a subset of these constraints and generate the rest as needed. In fact, they would solve a series of relaxed master problems [MP$_k$] and stop when an optimal (satisfactory) solution to (1) is reached.

Various cutting plane methods were proposed over the years. Each variant generates cuts at a different point called the *query point*. *Kelley's classical cutting plane method* [17] chooses the minimum of the relaxed master problem [MP$_k$] as a query point. Although, it may work well for some problems, this method suffers from slow convergence [22]. The *analytic center cutting plane method* (ACCPM) [15,16], on the other hand, chooses the analytic center as its query point. Its calculation makes use of interior point concepts and has shown promising results for a number of applications [1,2]. *Bundle methods* [19,20], choose the query point by solving a quadratic program that contains a small number of cutting planes. The information (bundle of cutting planes) is updated regularly and kept moderately small.

Conclusion

Nondifferentiable optimization tackles a class of problems that are intractable to classical optimization meth-

ods. Most of the theory is based on the notion of subgradients and most of the work is done for the convex case. It has an abundance of applications in real life, because the nondifferentiability aspect captures some of the inherent complexity in real-life problems. Like all disciplines, favoring an easily implementable and understood method will not necessarily lead to a good solution method. This corresponds to the subgradient method in NDO. Although it is easily implementable, it has slow convergence. More sophisticated methods, such as bundle methods or ACCPM are more promising from a computational point of view but require more knowhow of the method and of numerical linear algebra.

See also

- ▶ Dini and Hadamard Derivatives in Optimization
- ▶ Discontinuous Optimization
- ▶ Global Optimization: Envelope Representation
- ▶ Nondifferentiable Optimization: Cutting Plane Methods
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Nondifferentiable Optimization: Newton Method
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Nondifferentiable Optimization: Relaxation Methods
- ▶ Nondifferentiable Optimization: Subgradient Optimization Methods

References

1. Bahn O, Du Merle O, Goffin JL, Vial JP (1995) A cutting plane method from analytic centers for stochastic programming. Math Program B 69:45–73
2. Bahn O, Goffin JL, Vial JP, Du Merle O (1994) Implementation and behavior of an interior point cutting plane algorithm for convex programming: An application to geometric programming. Discrete Appl Math 49:3–23
3. Barrodale I, Phillips C (1974) An improved algorithm for discrete Chebychev linear approximation. In: Proc. Fourth Manitoba conference on Numerical Mathematics. Univ Manitoba, Winnipeg
4. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Math 4:238–252
5. Bertsekas DP (1995) Nonlinear programming. Athena Sci., Belmont
6. Cheney W, Goldstein AA (1959) Newton's method for convex programming and Chebyshev approximation. Numerische Math 1–5:253–268
7. Clarke FH (1989) Optimization and nonsmooth analysis. Les Publ. CRM, Montreal
8. Clarke FH, Ledyaev Yu, Ledyaev S, Stern RJ, Wolenski PR (1998) Nonsmooth analysis and control theory. Grad Texts Math. Springer, Berlin
9. Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8:101–111
10. Dantzig GB, Wolfe P (1961) The decomposition algorithm for linear programming. Econometrica 29(4):767–778
11. Fiacco AV, McCormick GP (1968) Nonlinear programming: Sequential unconstrained minimization techniques. Wiley, New York, Reprint: (1990) SIAM Classics Appl Math, vol 4. SIAM, Philadelphia
12. Fisher ML (1981) The Lagrangian relaxation method for solving integer programming problems. Managem Sci 27:1–18
13. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
14. Geoffrion AM (1974) Lagrangean relaxation for integer programming. Math Program Stud 2:82–114
15. Goffin J-L, Haurie A, Vial J-P (1992) Decomposition and nondifferentiable optimization with the projective algorithm. Managem Sci 38(2):284–302
16. Goffin J-L, Vial J-P (1998) Interior point methods for nondifferentiable optimization. In: Kishka P, Lorenz HW, Derigs U, Domschke W, Kleinschmidt P, Moehring R (eds) Operations Research Proc. 1997. Springer, Berlin, pp 35–49
17. Kelley JE (1960) The cutting plane method for solving convex programs. J SIAM 8:703–712
18. Kiwiel KC (1985) Methods of descent for nondifferentiable optimization. Lecture Notes Math, vol 1133. Springer, Berlin
19. Kiwiel KC (1990) Proximity control in Bundle methods for convex nondifferentiable optimization. Math Program 46:105–122
20. Lemaréchal C (March 28-April 8, 1977 1978) Bundle methods in nonsmooth optimization. In: Lemaréchal C, Mifflin R (eds) Nonsmooth Optimization: Proc. IIASA Workshop. Pergamon, Oxford
21. Madsen K, Nielsen HB, Pnar MÇ (1993) New characterizations of $\ell 1$ solutions of over-determined linear systems. Oper Res Lett 16
22. Nemirovskii AS, Yudin DB (1983) Problem complexity and method efficiency in optimization. Wiley, New York
23. Ruzinsky SA, Olsen ET (1989) $\ell 1$ and $\ell\infty$ minimization via a variant of Karmarkar's algorithm. IEEE Trans Acoustics, Speech and Signal Processing 37
24. Shor NZ (March 28-April 8, 1977 1978) Subgradient methods: A survey of soviet research. In: Lemaréchal C, Mifflin R (eds) Nonsmooth optimization: Proc. IIASA workshop. Pergamon, Oxford

25. Watson GA (1980) Approximation theory and numerical methods. Wiley, New York
26. Wolfe P (1975) A method of conjugate subgradients for minimizing nondifferentiable functions. Math Program Stud 3:145–173
27. Zhang Y (1993) Primal-dual interior point approach for computing the $\ell 1$-solutions and $\ell\infty$-solutions of overdetermined linear systems. J Optim Th Appl 77(2)

Nondifferentiable Optimization: Cutting Plane Methods

Samir Elhedhli[1], Jean-Louis Goffin[1], Jean-Philippe Vial[2]
[1] McGill University, Montreal, Canada
[2] Logilab/Department Management Stud., University Genève, Geneva, Switzerland

MSC2000: 49M20, 90C25, 90-08

Article Outline

Keywords
A Generic Cutting Plane Algorithm
Kelley's Cutting Plane Method
Center of Gravity Method
Largest Inscribed Sphere Method
Volumetric Method
Bundle Methods
Analytic Center Cutting Plane Method (ACCPM)
Concluding Remarks
See also
References

Keywords

Nondifferentiable optimization; Nonsmooth optimization; Cutting planes

Cutting plane methods were proposed independently by J.E. Kelley [27] and W. Cheney and A.A. Goldstein [5] as a solution technique for constrained convex optimization problems. Although they may not compete with some of the efficient methods for smooth optimization, they are one of the first and fundamental solution approaches for nondifferentiable optimization (cf. ▶ Nondifferentiable optimization).

The fact that they rely on polyhedral approximations of convex functions, makes the technique suitable for nondifferentiable optimization. It forms with subgradient optimization [38] the two major solution approaches for nondifferentiable problems. The cutting plane algorithms that are being designed over the years are getting more sophisticated and ultimately, showing promising computational results and stable numerical properties.

In this article, we give an overview of cutting plane methods for nondifferentiable optimization problems. We start with a brief introduction to cutting planes. We then give a generic cutting plane algorithm, that will be used to describe some of the main variants.

A Generic Cutting Plane Algorithm

To describe the general cutting plane approach, we consider the following nondifferentiable problem

$$[NDP] \begin{cases} \min & f(x) \\ \text{s.t.} & g(x) \le 0\,, \end{cases}$$

where f and g are real-valued, continuous, nondifferentiable, convex functions. Convexity implies that there is at least one supporting hyperplane to f at every point x_0 of the domain, whose equation is given by

$$y = f(x_0) + \xi_0^f (x - x_0)\,,$$

where ξ_0^f is any element of the subdifferential $\partial f(x_0)$ of f at x_0. (For ease of notation we assume that subgradients are row vectors.) Recalling that a supporting hyperplane gives an under-estimate of f, the *subgradient inequality*

$$f(x_0) + \xi_0^f (x - x_0) \le f(x) \tag{1}$$

can be used to approximate f by the maximum of a set of piecewise linear functions. Therefore, given a set of points x_i, $i \in I$ and their corresponding subgradients ξ_i^f, f is tangentially approximated by

$$\overline{f}(x) = \max_{i \in I} \left\{ f(x_i) + \xi_i^f (x - x_i) \right\}\,. \tag{2}$$

Inequality (1) implies that $\overline{f}(x) \le f(x)$ for any index set I. Larger sets will give better approximations.

The same can be said about g, leading to the polyhedral approximation

$$\overline{g}(x) = \max_{j\in J} g(x_j) + \xi_j^g (x - x_j)\,,$$

where ξ_j^g are subgradients of g at a collection of points x_j indexed by J. Thus, [NDP] can be approximated by,

$$\begin{cases} \min & \left\{ \max\limits_{i\in I} f(x_i) + \xi_i^f (x - x_i) \right\} \\ \text{s.t.} & \max\limits_{j\in J} g(x_i) + \xi_j^g (x - x_j) \le 0\,, \end{cases}$$

which is equivalent to,

$$\begin{cases} \min & v \\ \text{s.t.} & f(x_i) + \xi_i^f (x - x_i) \le v, \quad \forall i \in I, \\ & g(x_j) + \xi_j^g (x - x_j) \le 0, \quad \forall j \in J. \end{cases} \tag{3}$$

Problem (3) is a linear program that is easier to deal with than the original problem. It is to note, however, that this is only an approximation of [NDP], that gets better as more constraints are added. Let us denote by [MP_k] the *relaxed master problem* (3) with index sets I_k and J_k whose optimal solution is denoted by (x_k, v_k).

By transforming [NDP] into (3), nondifferentiability is eliminated at the cost of having a problem with a very large number of constraints. To get around that, cutting plane methods use only a subset of these constraints and generate the rest as needed. In fact, they would handle a series of relaxed master problems [MP_k] and stop when an optimal (satisfactory) solution to [NDP] is reached. [MP_k] is a relaxation of [NDP] as, by convexity of g,

$$\left\{ x\colon \max_{j\in J_k} \left\{ g(x_j) + \xi_j^g (x - x_j) \right\} \le 0 \right\}$$

contains $\{x\colon g(x) \le 0\}$, while, by convexity of f,

$$\max_{i\in I_k} \left\{ f(x_i) + \xi_i^f (x - x_i) \right\} \le f(x)$$

for all $x \in \{x\colon g(x) \le 0\}$. This implies that the optimal value v_k is a lower bound on the optimal value $f(x^*)$. The relaxation gets tighter as more points are included in the index sets I and J. The hope is that termination occurs before the sets I and J get enormously big.

The idea described above can be put into the generic cutting plane algorithm below.

In the course of the cutting plane algorithm, upper bounds are usually achieved through the objective evaluation at a feasible point. Lower bounds are either given by the optimal solution of the relaxed master problem, or by evaluating the dual objective at a feasible dual solution. For the stopping criteria, the algorithm may be stopped if the bound on the duality gap drops below a certain threshold. This is given by the difference between the best known upper and lower bounds.

Initialize:

1 Get an initial upper bound v_u and a lower bound v_l for the optimal solution $f(x^*)$.
2 Get an initial relaxed master problem [MP_0].

Iterate (k)

1 Get a *query point* x_k and a corresponding lower bound v_l^+.
2 If x_k is infeasible to [NDP] ($g(x_k) > 0$), then the oracle of g generates a *feasibility cut* of type $g(x_k) + \xi_j^g (x - x_k) \le 0$.
3 If x_k is feasible to [NDP] ($g(x_k) \le 0$), then the oracle of f generates an *optimality cut* of type $f(x_k) + \xi_i^f (x - x_k) \le v$ and an upper bound v_u^+.
4 Update the bounds:
a) if x_k is feasible to [NDP], then $v_u = \max\{v_u, v_u^+\}$.
b) $v_l = \min\{v_l, v_l^+\}$
5 Update index sets:
a) If a feasibility cut is added then $I_{k+1} := I_k$ and $J_{k+1} := J_k \cap \{k\}$.
b) If an optimality cut is added then $I_{k+1} := I_k \cap \{k\}$ and $J_{k+1} := J_k \cap \{k\}$.
6 Either STOP or $k := k + 1$.

At each iteration of the method, we can construct a bounded polyhedral set, namely the *localization set*. Given an upper bound v_u^k to [MP_k], it is given by

$$F_k(v_u^k) = \left\{ (x, v)\colon \begin{array}{c} v \le v_u^k; \\ f(x_i) + \xi_i^f (x - x_i) \le v, \\ i \in I_k; \\ g(x_j) + \xi_j^g (x - x_j) \le 0, \\ j \in J_k \end{array} \right\}.$$

The localization set $F_k(v_u)$ is a bounded polyhedral subset of the feasible region of [MP_k] that contains any optimal solution $(x^*, f(x^*))$ to [NDP]. To see that, $f(x^*)$

$\leq v_u^k$ as v_u^k is an upper bound on the optimal value. Furthermore, by the feasibility of x^* and the subgradient inequality $g(x_j) + \xi_j^g(x^* - x_j) \leq g(x^*) \leq 0$ for all $j \in J_k$ and by the fact that [MP$_k$] is a relaxation of [NDP], $f(x_i) + \xi_i^f(x^* - x_i) \leq v_k \leq f(x^*)$ for all $i \in I_k$.

Although every cutting plane method follows the above general scheme, different query point choices produce different algorithms. Prior to discussing a few of them, we would like to stress that some methods are explicitly based on the localization set, and require that this set be bounded. This might not hold in particular in the initialization phase, since no cut is yet present, and no upper and lower bounds on the objective are known. Therefore, one has to introduce box constraints. This may not even suffice if the first generated cut is not an optimality one, because the localization set remains unbounded in the z variable. One must then proceed with a an auxiliary phase I problem.

Kelley's Cutting Plane Method

This is the classical cutting plane method that was originally described in [27] and is present in the original *Dantzig–Wolfe decomposition* [7,8] and Benders decomposition [4]. At iteration k, the method solves the relaxed master [MP$_k$] and uses its solution x_k to generate further cuts.

As [MP$_k$] is a relaxation of [NDP], v_k gives a lower bound to $f(x^*)$ which is monotonically increasing. In addition, when x_k is feasible, $f(x_k)$ gives an upper bound that, unfortunately, is not monotonically decreasing. The difference between the updated bounds gives an estimate of the duality gap and can be used as a practical stopping criteria. The exact optimal solution of [NDP] is detected when x_k is feasible and $f(x_k) = v_k$, which is equivalent to having $0 \in \partial f(x_k)$.

This optimal point strategy assumes that the relaxation is a good approximation of the original problem, but this is only true when a big number of cuts has been generated. The method is globally convergent [47], but in practice, it sometimes shows a slow pattern of convergence [35].

Center of Gravity Method

This method was first proposed in [32] as the first cutting plane method that generates query points at the center of the localization set. Choosing the *center of gravity* to generate query points seems to be the natural choice. In fact cutting through the center of gravity of the a localization set of volume V and dimension n produces two sets, each with a volume of at least

$$\left[1 - (1 - \frac{1}{n+1})^n\right] V.$$

Therefore, after k iterations the volume of the localization set shrinks at a constant rate of $1/(1 - \exp(1))$. This rate of convergence is the best that can be obtained [46]. Unfortunately, finding a single center of gravity could be as difficult as solving the original problem.

For some simple convex bodies such as ellipsoids, cubes or spheres, calculating the center of gravity is relatively easy. This idea is the motivation behind the largest inscribed sphere method of [11], the volumetric method of [43] and the analytic center cutting plane method (ACCPM) of [18,19,21].

Largest Inscribed Sphere Method

The query point of this method is chosen as the center of the largest inscribed sphere in the localization set. Its calculation is based on work of G.L. Nemhauser and W.B. Widhelm [34], who showed that the minimization of the simple linear program

$$\begin{cases} \min & \sigma \\ \text{s.t.} & a_i^\top x + \left\| a_i^\top \right\| \sigma \leq b_i, \quad i \in I, \end{cases}$$

gives the radius σ and the center x of the largest inscribed sphere in the bounded polyhedron $\{x\colon a_i^\top x \leq b_i, i \in I\}$. The method is detailed in [11].

Volumetric Method

P.M. Vaidya [43] proposed the volumetric center as a query point. It is the maximizer of the determinant of the Hessian matrix of the logarithmic barrier function. This choice is motivated by the observation that for every point of the localization set, an inscribed ellipsoid can be constructed. The point that gives the maximum-volume inscribed ellipsoid is the minimizer of

$$\frac{1}{2} \log \left(\det \left(\sum \frac{a_i a_i^\top}{(b_i - a_i^\top x)^2} \right) \right),$$

where a_i are the columns of the matrix defining the localization set.

Bundle Methods

These methods were first proposed by C. Lemaréchal [30]. The method has developed over the years, [28,31], building upon the pioneering work [30].

The method adds a regularization term to the estimation of f and solves

$$\begin{cases} \min & \left\{ \max\limits_{i\in I_k} f(x_i) + \xi_i^f (x - x_i) \right. \\ & \qquad \left. + \frac{1}{2\alpha_k} \|x - x_{k-1}\|^2 \right\} \\ \text{s.t.} & \max\limits_{j\in J_k} \left\{ g(x_j) + \xi_j^g (x - x_j) \right\} \le 0, \end{cases} \tag{4}$$

to get the next query point x_k. The main idea is based on the fact that (2) is a good approximation of f when the next iterate is not far from the previous ones. Problem (4) is equivalent to the quadratic program

$$\begin{cases} \min & v + \frac{1}{2\alpha_k} \|x - x_{k-1}\|^2 \\ \text{s.t.} & f(x_i) + \xi_i^f (x - x_i) \le v, \quad i \in I_k, \\ & g(x_j) + \xi_j^g (x - x_j) \le 0, \quad j \in J_k. \end{cases}$$

The nice feature of bundle methods is that they limit the number of hyperplanes that are used to approximate f. The set of subgradients (the bundle) is updated at each iteration and kept moderately small. As a result, (4) is solved very quickly.

Techniques to estimate α_k are treated in [28] where three different values are proposed.

Analytic Center Cutting Plane Method (ACCPM)

To overcome the difficulty associated with the calculation of centers of polyhedrons and still use a central point strategy, ACCPM uses concepts form the interior point literature that have proven to be highly efficient.

The query point for ACCPM is the *analytic center* of the localization set $F_k(v_u)$. This new notion of center was first introduced in [40,41,42] as the the unique pair (x_a^k, v_a^k) that minimizes

$$\log(v_u^k - v) + \sum_{i\in I_k} \log[v - f(x_i) - \xi_i^f (x - x_i)] + \sum_{i\in J_k} \log[-g(x_i) - \xi_i^g (x - x_i)].$$

This function is a *potential function* similar to the one used by N.K. Karmarkar [26] when presenting the first interior point algorithm. Its minimization is equivalent to the solution of a linear program by any interior point method. The primal projective algorithm (cf. also ► Linear programming: Karmarkar projective algorithm) is favored due to its ability to deal with dual infeasibilities when new cuts are added. It is, in principle, a modified Newton method applied to the minimization of the potential function [16].

The *Newton procedure* for computing a central point (x_a^k, v_a^k) of $F_k(v_u)$ identifies, upon termination, a dual feasible solution to [MP$_k$]. Evaluating the dual objective at this point, gives a lower bound to the optimal solution of [NDP]. In addition, if x_a^k is feasible to [NDP], i. e. $g(x_a^k) \le 0$, then $f(x_a^k)$ is an upper bound. Unfortunately, x_a^k never coincides with the original optimizer x^*, as it can never be a vertex, so the stopping criteria can only be based on the difference between the upper a lower bounds. The algorithm would stop if that gap is sufficiently small.

The convergence analysis of the method is done in [20] and [36]. ACCPM is readily implemented [25] and has shown promising results in solving different large scale problems [2,3,17,33]. The method was modified to use weighted analytic centers as a query point [22], to add multiple cuts at once [24], to use a primal-dual interior point algorithm for the calculation of analytic centers [9] and to accommodate quadratic cuts [10].

Concluding Remarks

Cutting plane methods are an effective solution approach for nondifferentiable optimization. This has proven to be true when advanced methods such as bundle and analytic center methods were designed. Not only they make use of recent advances in optimization theory, they also resulted in efficient computer implementations.

Manipulating the set of cuts is an important enhancement factor to cutting plane methods. In the addition of cuts, introducing a whole set at once is more effective than introducing single cuts as it allows the fast accumulation of information about the problem. This is possible for certain classes of problems where disaggregation is possible. This is true for Dantzig–Wolfe and

Benders decompositions with primal and dual block diagonal matrices respectively.

The second improvement is in the manipulation of number of cuts in the relaxed master problem. Keeping all cuts may lead to a better approximation but it also requires huge amounts of storage space and solution time. Thus well defined cut-dropping strategies will considerably improve the performance of the method.

A final remark concerns the use of heuristics with cutting planes. They can be used to initialize the method so that sufficient cutting planes are obtained to start the method, or take over the search for optimal solutions when the method stalls.

See also

- ► Dini and Hadamard Derivatives in Optimization
- ► Global Optimization: Envelope Representation
- ► Nondifferentiable Optimization
- ► Nondifferentiable Optimization: Minimax Problems
- ► Nondifferentiable Optimization: Newton Method
- ► Nondifferentiable Optimization: Parametric Programming
- ► Nondifferentiable Optimization: Relaxation Methods
- ► Nondifferentiable Optimization: Subgradient Optimization Methods

References

1. Atkinson DS, Vaidya PM (1995) A cutting plane algorithm that uses analytic centers. Math Program B 69:1–43
2. Bahn O, Goffin JL, Vial JP, du Merle O (1994) Implementation and behavior of an interior point cutting plane algorithm for convex programming: An application to geometric programming. Discrete Appl Math 49:3–23
3. Bahn O, Merle OD, Goffin JL, Vial JP (1995) A cutting plane method from analytic centers for stochastic programming. Math Program B 69:45–73
4. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Math 4:238–252
5. Cheney W, Goldstein AA (1959) Newton's method for convex programming and Chebyshev approximation. Numerische Math 1–5:253–268
6. Clarke FH (1989) Optimization and nonsmooth analysis. Les Publ. CRM, Montreal
7. Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8:101–111
8. Dantzig GB, Wolfe P (1961) The decomposition algorithm for linear programming. Econometrica 29(4):767–778
9. Denault M, Goffin J-L (1997) On a primal-dual analytic center cutting plane method for variational inequalities. GERAD Techn Report G-97-56
10. Denault M, Goffin J-L (1998) Variational inequalities with quadratic cuts. GERAD Techn Report G-98-69
11. Elzinga J, Moore TG (1975) A central cutting plane algorithm for the convex programming problem. Math Program 8:134–145
12. Fiacco AV, McCormick GP (1968) Nonlinear programming: Sequential unconstrained minimization techniques. Wiley, New York. Reprint: SIAM Classics Appl Math, vol 4, 1990
13. Fisher ML (1981) The Lagrangian relaxation method for solving integer programming problems. Managem Sci 27:1–18
14. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
15. Geoffrion AM (1974) Lagrangean relaxation for integer programming. Math Program Stud 2:82–114
16. de Ghellinck G, Vial JP (1986) A polynomial Newton method for linear programming. Algorithmica 1:425–453
17. Goffin J-L, Gondzio J, Sarkissian R, Vial J-P (1997) Solving nonlinear multicommodity flow problem by the analytic centre cutting plane method. Math Program B 76:131–154
18. Goffin J-L, Haurie A, Vial J-P (1992) Decomposition and nondifferentiable optimization with the projective algorithm. Managem Sci 38(2):284–302
19. Goffin J-L, Haurie A, Vial J-P, Zhu DL (1993) Using central prices in the decomposition of linear programs. Europ J Oper Res 64:393–409
20. Goffin J-L, Luo Z-Q, Ye Y (1996) Complexity analysis of an interior cutting plane method for convex feasibility problems. SIAM J Optim 6:638–652
21. Goffin J-L, Vial J-P (1990) Cutting planes and column generation techniques with the projective algorithm. J Optim Th Appl 65:409–429
22. Goffin J-L, Vial J-P (1993) On the computation of weighted analytic centers and dual ellipsoids with the projective algorithm. Math Program 60:81–92
23. Goffin J-L, Vial J-P (1998) Interior point methods for nondifferentiable optimization. In: Kishka P, Lorenz HW, Derigs U, Domschke W, Kleinschmidt P, Moehring R (eds) Operations Research Proc. 1997. Springer, Berlin, pp 35–49
24. Goffin J-L, Vial J-P (1998) Multiple cuts in the analytic center cutting plane method. HEC/Logilab Techn Report 98.10 and G-98-26, Dept. Management Stud Univ Geneva
25. Gondzio J, du Merle O, Sarkissian R, Vial JP (1994) ACCPM-A library for convex optimization based on analytic centre cutting plane method. Europ J Oper Res 94:206–211
26. Karmarkar NK (1984) A new polynomial-time algorithm for linear programming. Combinatorica 4:373–395
27. Kelley JE (1960) The cutting plane method for solving convex programs. J SIAM 8:703–712
28. Kiwiel KC (1990) Proximity control in Bundle methods for convex nondifferentiable optimization. Math Program 46:105–122

29. Lasdon L (1970) Optimization theory for large scale systems. MacMillan, New York
30. Lemaréchal C (1978) Bundle methods in nonsmooth optimization. In: Lemarechal C, Mifflin R (eds) Nonsmooth Optimization, Proc. IIASA Workshop, March 28–April 8 1977. Pergamon, Oxford
31. Lemaréchal C, Nemrirovskii A, Nesterov Y (1995) New variants of bundle methods. Nonsmooth Optim, Math Program 69:111–147
32. Levin AY (1965) On an algorithm for the minimization of convex functions over convex functions. Soviet Math Dokl 6:286–290
33. du Merle O, Hansen P, Jaumard B, Mladenovic N (1997) An interior point algorithm for minimum sum of squares clustering. GERAD Techn Report G97-53
34. Nemhauser GL, Widhelm WB (1971) A modified linear program for columnar methods in mathematical programming. Oper Res 19:1051–1060
35. Nemirovskii AS, Yudin DB (1983) Problem complexity and method efficiency in optimization. Wiley, New York
36. Nesterov Y (1996) Cutting plane methods from analytic centers: efficiency estimates. Math Program B 69:149–176
37. Roos C, Terlaky T, Vial J-P (1997) Interior point methods: Theory and algorithms. Springer, Berlin
38. Shor NZ (1978) Subgradient methods: A survey of Soviet research. In: Lemaréchal C, Mifflin R (eds) Nonsmooth optimization: Proc. IIASA workshop, March 28–April 8 1977. Pergamon, Oxford
39. Shor NZ (1985) Minimization methods for nondifferentiable functions. Springer, Berlin
40. Sonnevend G (1985) An analytical center for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming. Lecture Notes Control Inform Sci, vol 84. Springer, Berlin, pp 866–876
41. Sonnevend G (1988) New algorithms in convex programming based on a notion of centre (for systems of analytic inequalities) and on rational extrapolation. In: Hoffmann KH, Hiriat-Urruty JB, Lemaréchal C, Zowe J (eds) Trends in Mathematical Optimization; Proc. 4th French–German Conf. Optimization, Irsee, April 1986. Internat Ser Numer Math. Birkhäuser, Basel, pp 311–327
42. Sonnevend G (1989) Applications of the notion of analytic center in approximation (estimation) problem. J Comput Appl Math 28:349–358
43. Vaidya P (1996) A new algorithm for minimizing convex functions over convex sets. Math Program 73:291–341
44. Wolfe P (1975) A method of conjugate subgradients for minimizing nondifferentiable functions. Math Program Stud 3:145–173
45. Ye Y (1992) A potential reduction algorithm allowing column generation. SIAM J Optim 2:7–20
46. Ye Y (1997) Interior point algorithms: Theory and analysis. Wiley, New York
47. Zangwill WI (1969) Nonlinear programming: A unified approach. Prentice-Hall, Englewood Cliffs

Nondifferentiable Optimization: Minimax Problems

VLADIMIR F. DEMYANOV
St. Petersburg State University, St. Petersburg, Russia

MSC2000: 90C30, 65K05

Article Outline

Keywords
Max-Type Functions
Algorithms for Unconstrained Minimization
  Method of Steepest Descent
  Hypodifferential Descent
  Newton-Type Method
A Convex Max-Function
  Extremal Basis Method
Constrained Minimax Problems
Method of Hypodifferential Descent
The Kelley Method
See also
References

Keywords

Minimax problem; Nondifferentiable optimization; Inf-stationary point; Extremal basis method; Hypodifferential; Subdifferential

Max-Type Functions

Let $S \subset \mathbf{R}^n$ be an open set. A standard *minimax problem* (MMP) is the problem of minimizing a function

$$f(x) = \max_{y \in G} \varphi(x, y), \tag{1}$$

where G is a compact set of some space Y, the function $\varphi: S \times G \to \mathbf{R}$ is continuous jointly in $[x, y]$ on $S \times G$ and continuously differentiable in x (the function $\varphi'_x(x, y)$ is continuous in $[x, y]$ on $S \times G$). The function f is, in general, nonsmooth though φ is smooth in x.

Remark 1 The function

$$f(x) = \max_{y \in G} |\varphi(x, y)|$$

can be rewritten in the form (1) since

$$\begin{aligned} f(x) &= \max_{y \in G} \max\{\varphi(x, y), -\varphi(x, y)\} \\ &= \max_{\overline{y} \in \overline{G}} \overline{\varphi}(x, \overline{y}), \end{aligned}$$

where

$$\overline{G} = \{[1, y]\colon y \in G\} \bigcup \{[-1, y]\colon y \in G\} \subset \mathbf{R} \times Y = \overline{Y},$$

$$\overline{\varphi}(x, \overline{y}) = \begin{cases} \varphi(x, y), & \overline{y} = [1, y], \\ -\varphi(x, y), & \overline{y} = -[1, y]. \end{cases}$$

The first serious study of minimax problems was performed by P.L. Chebyshev who may be considered as Godfather of nonsmooth analysis. He treated the problem of minimizing the function

$$f(x) = \max_{y \in [a,b]} |\Psi(y) - P(x, y)|, \tag{2}$$

where $x = (x_0, \ldots, x_{n-1}) \in \mathbf{R}^n$, $P(x, y) = \sum_{i=0}^{n-1} x_i y^i$. The function $P(x, y)$ is a polynomial (in y). If $x^* \in \mathbf{R}^n$ and

$$f(x^*) = \min_{x \in \mathbf{R}^n} f(x)$$

then $P(x^*, y)$ is called the *polynomial of best approximation* (for the function $\Psi(y)$). If $\Psi(y) = y^n$, $[a, b] = [-1, 1]$ then

$$P_n^*(y) = y^n - P(x^*, y) = \frac{1}{2^{n-1}} T_n(y),$$

where $T_n(y)$ is the famous *Chebyshev polynomial*:

$$T_0(y) = 1, \quad T_1(y) = y,$$
$$T_{m+1}(y) = 2yT_m(y) - T_{m-1}(y), \quad \forall m \geq 2.$$

On the interval $[-1, 1]$ the relation $T_n(y) = \cos(n \arccos y)$ holds.

Let $\varphi_1(x, y) = \Psi(y) - P(x, y)$. The following condition holds (see [4,9]): For a point $x^* \in \mathbf{R}^n$ to be a minimizer of f (see (2)) it is necessary and sufficient that $n + 1$ points $y_0, \ldots, y_n$ exist such that

$$y_i \in G, \quad \forall i \in 0, \ldots, n; \quad y_0 < \cdots < y_n,$$

$$\varphi_1(x^*, y_{i+1}) = -\varphi_1(x^*, y_i), \tag{3}$$

$$\forall i \in 0, \ldots, n-1, \tag{4}$$

$$|\varphi_1(x^*, y_{i+1})| = \varphi(x^*, y_i) = f(x^*). \tag{5}$$

The set $\{y_0, \ldots, y_n\}$ satisfying (3)–(5) is called a *Chebyshev alternation* (or *alternance*).

The following properties of a max-function will be used in the sequel (see [1,4,5]).

1) Let a function f be described by (1). Then it is directionally differentiable at any point $x \in S$ and

$$f'(x, g) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha g) - f(x)}{\alpha} = \max_{v \in \partial f(x)} (v, g), \tag{6}$$

where

$$\partial f(x) = \operatorname{co} \{\varphi'(x, y)\colon y \in R(x)\}, \quad g \in \mathbf{R}^n,$$
$$R(x) = \{y \in G\colon \varphi(x, y) = f(x)\}.$$

(6) means that f is subdifferentiable, $\partial f(x)$ is the *subdifferential* of f at x (it is a convex compact set). The subdifferential mapping ∂f is not, in general, Hausdorff continuous.

2) For a point $x^* \in S$ to be a (local or global) minimizer of f on S it is necessary that

$$0_n \in \partial f(x^*). \tag{7}$$

Here $0_n = (0, \ldots, 0) \in \mathbf{R}^n$.

A point $x^* \in S$ satisfying (7) is called an *inf-stationary point* of f.

Remark 2 Note that for a point $x^{**} \in S$ to be a (local or global) maximizer of f on S it is necessary that

$$\partial f(x^{**}) = \{0_n\}. \tag{8}$$

If the function f is convex (it is the case, e. g., if φ is convex in x for any $y \in G$), then condition (7) is also sufficient for x^* to be a global minimizer of f on an open set S. Therefore condition (8) implies that a convex function f does not attain its maximal value on an open convex set S.

3) If x_0 is not inf-stationary then the direction

$$g(x_0) = -\frac{v_0}{\|v_0\|}, \tag{9}$$

where

$$\|v_0\| = \max_{v \in \partial f(x_0)} \|v\|,$$

is the steepest descent direction of f at x_0, i. e.

$$f'(x_0, g(x_0)) = \min_{\|g\|=1} f'(x_0, g).$$

4) The following expansion holds

$$f(x+\Delta) = f(x) + \max_{y\in G}[\varphi(x,y) - f(x) + (\varphi'_x(x,y),\Delta)] + o_x(\Delta),$$

where

$$\frac{o_x(\Delta)}{\|\Delta\|} \overset{\|\Delta\|\to 0}{\longrightarrow} 0.$$

This expression can be rewritten as

$$f(x+\Delta) = f(x) + \max_{[a,v]\in \mathrm{d}f(x)} [a + (v,\Delta)] + o_x(\Delta), \tag{10}$$

where

$$\mathrm{d}f(x) = \mathrm{co}\left\{[a,v]\in \mathbf{R}\times\mathbf{R}^n : \begin{array}{c} a=\varphi(x,y)-f(x),\\ v=\varphi'_x(x,y),\\ y\in G\end{array}\right\} \tag{11}$$

is the *hypodifferential* of f at x. Note that $a \geq 0$ and $\max_{[a,v]\in df(x)} a = 0$. The mapping df is Hausdorff-continuous on S.

5) Necessary condition (7) is equivalent to

$$0_{n+1} \in \mathrm{d}f(x^*). \tag{12}$$

6) If x_0 is not inf-stationary, then let us find

$$\min_{z\in \mathrm{d}f(x^*)} \|z\| = \|z_0\|,$$

where $z_0 = [a_0, v_0]$. In this case $v_0 \neq 0_n$. The direction

$$\widetilde{g}(x_0) = -\frac{v_0}{\|v_0\|} \tag{13}$$

is a descent (not necessary steepest descent) direction of f at x_0.

The vector-function $\widetilde{g}(x)$ (see (13)) is continuous.

7) If the function φ in (1) is twice continuously differentiable at x then the following relation holds:

$$f(x+\Delta) = f(x) + \max_{y\in G}\left[\varphi(x,y) - f(x) + (\varphi'_x(x,y),\Delta) + \frac{1}{2}(\varphi_{xx}(x,y)\Delta,\Delta)\right] + o_x(\Delta^2), \tag{14}$$

where

$$\frac{o_x(\Delta^2)}{\|\Delta\|^2} \overset{\|\Delta\|\to 0}{\longrightarrow} 0.$$

(14) can be rewritten as

$$f(x+\Delta) = f(x) + \max_{[a,v,A]\in d^2 f(x)}\left[a + (v,\Delta) + \frac{1}{2}(A\Delta,\Delta)\right] + o_x(\Delta^2), \tag{15}$$

where

$$d^2 f(x) = \mathrm{co}\left\{[a,v,A]\in \mathbf{R}\times\mathbf{R}^n\times\mathbf{R}^{n\times n} : \begin{array}{c} a=\varphi(x,y)-f(x),\\ v=\varphi'_x(x,y),\\ A=\varphi_x(x,y),\\ y\in G\end{array}\right\}. \tag{16}$$

Here $\mathbf{R}^{n\times n}$ is the space of real $(n\times n)$-matrices. The set $d^2 f(x)$ is called the *second hypodifferential* of f at x. It is closed, bounded and convex. The mapping $d^2 f$ is Hausdorff continuous on S. In this case f is twice continuously hypodifferentiable.

Algorithms for Unconstrained Minimization

Assume that in (1) $S = \mathbf{R}^n$. Then the problem of minimizing f is an unconstrained minimax problem. There are a lot of numerical methods to solve this problem based on the properties of max-functions.

Method of Steepest Descent

Let $x_0 \in \mathbf{R}^n$ be arbitrary. Assume that x_k has already been defined. If $0_n \in \partial f(x_k)$ then x_k is an inf-stationary point and the process terminates. If $0_n \notin \partial f(x_k)$ then let us take $g_k = g(x_k)$ (see (9)) and find

$$\min_{\alpha\geq 0} f(x_k + \alpha g_k) = f(x_k + \alpha_k g_k). \tag{17}$$

Now, put $x_{k+1} = x_k + \alpha_k g_k$. Continuing in the same manner we construct a sequence $\{x_k\}$ such that

$$f(x_{k+1}) < f(x_k). \tag{18}$$

If this sequence $\{x_k\}$ is finite, then, by construction, the last point is a stationary one.

Let $\{x_k\}$ be not finite. Assume that the set $Q(x_0) = \{x \in \mathbf{R}^n: f(x) \le f(x_0)\}$ is bounded (then it is closed). Due to (18) $x_k \in Q(x_0)$ and, hence, there exist a point $x^* \in Q(x_0)$ and a subsequence $\{x_{k_s}\}$ such that $x_{k_s} \to x^*$. One may expect that x^* is inf-stationary. However, in general, this is not the case and the reason is the discontinuity of the mapping ∂f.

To ensure the convergence it is necessary to overcome the discontinuity of ∂f.

Let us introduce the set $R_\varepsilon(x) = \{y \in G: f(x) - \varphi(x, y) \le \varepsilon\}$ where $\varepsilon \ge 0$. Put

$$L_\varepsilon(x) = \operatorname{co}\left\{\varphi'_x(x, y): y \in R_\varepsilon(x)\right\}.$$

Find

$$\min_{v \in L_\varepsilon(x)} \|v\| = \|v_\varepsilon(x)\|.$$

If $\|v_\epsilon(x)\| = 0$, the point x is called an *ε-stationary point*. Choose any $\varepsilon > 0$. Let us construct the following method. Take any $x_0 \in \mathbf{R}^n$. Let x_k have already been found. If $\|v_\varepsilon(x_k)\| = 0$ then x_k is ε-stationary. If $\|v_\varepsilon(x_k)\| > 0$ then let us find

$$\min_{\alpha \ge 0} f(x_k - \alpha v_\varepsilon(x_k)) = f(x_k - \alpha_k v_\varepsilon(x_k))$$

and put $x_{k+1} = x_k - \alpha_k v_\varepsilon(x_k)$. Continuing analogously we get a sequence $\{x_k\}$.

Proposition 3 *If the set $Q(x_0)$ is bounded then in a finite number of steps we arrive at a point x_k such that*

$$0_n \in L_{2\varepsilon}(x_k).$$

Thus, in a finite number of steps we shall find a 2ε-stationary point.

Now it is not difficult to modify this method to get an inf-stationary point of f.

Choose any $\varepsilon_0 > 0$ and $x_0 \in \mathbf{R}^n$. Assume that $Q(x_0)$ is bounded. Applying the above method, in a finite number of steps we shall find a point $\overline{x}_0$ such that $0_n \in L_{2\varepsilon_0}(\overline{x}_0)$. Let $\overline{x}_k \in Q(x_0)$ be found such that $0_n \in L_{2\varepsilon_k}(\overline{x}_k)$ where $\varepsilon_k = 2^{-k}\varepsilon_0$. Take $\varepsilon_{k+1} = \varepsilon_k/2$ and $x_k = \overline{x}_k$. Applying the above method, in a finite number of steps a point $\overline{x}_{k+1} \in Q(x_0)$ will be found such that $0_n \in L_{2\varepsilon_{k+1}}(\overline{x}_{k+1})$. Clearly, $\overline{x}_{k+1} \in Q(x_0)$. The sequence $\{\overline{x}_k\}$ is bounded.

Proposition 4 *Any limit point of the sequence $\{x_k\}$ is an inf-stationary point of f.*

Hypodifferential Descent

Another method is based on expansion (10). Take $x_0 \in \mathbf{R}^n$. Let x_k have been found. Compute

$$\min_{z \in df(x_k)} \|z\| = \|z_k\|,$$

where $z_k = [a_k, v_k]$. If $\| z_k \| = 0$ then the point x_k is inf-stationary and the process terminates.

If $\| z_k \| > 0$ then $v_k \ne 0_n$ and let us find

$$\min_{\alpha \ge 0} f(x_k - \alpha v_k) = f(x_k - \alpha_k v_k)$$

and put $x_{k+1} = x_k - \alpha_k v_k$.

If the sequence $\{x_k\}$ thus constructed is finite then the last point is inf-stationary. If $\{x_k\}$ is infinite then the following result holds:

Proposition 5 *If the set $Q(x_0)$ is bounded then any limit point of the sequence $\{x_k\}$ is inf-stationary.*

Remark 6 The two algorithms described above are 'conceptual' (according to the terminology of E. Polak). These algorithms are computationally effective in the case where the set G (see (1)) contains only a finite number of points. Different practical implementations of the above ideas can be found in [2,4,10,11,12].

Newton-Type Method

If the function φ in (1) is twice continuously differentiable, one can employ the expansion (14)–(15).

Take any $x_0 \in \mathbf{R}^n$. Let x_k have already been defined. Find

$$\min_{\Delta \in \mathbf{R}^n} F_k(\Delta) = F_k(\Delta_k),$$

where

$$F_k(\Delta) = \max_{[a,v,A] \in d^2 f(x_k)} \left[a + (v, \Delta) + \frac{1}{2}(A\Delta, \Delta)\right].$$

Now put $x_{k+1} = x_k + \Delta_k$.

Under some additional conditions (see [3]) the sequence $\{x_k\}$ thus constructed converges at least to a local minimizer of f (and the rate of convergence is quadratic).

A Convex Max-Function

Extremal Basis Method

Now let us consider the case where f is described by (1) and the function φ in (1) is strongly convex in x with

a constant $m > 0$ for every fixed $y \in G$, i. e.

$$\varphi(x + \Delta, y) \geq \varphi(x, y) + (\varphi_x'(x, y), \Delta) + m\|\Delta\|^2, \quad \forall [x, y] \in \mathbf{R}^n \times G, \quad \forall \Delta \in \mathbf{R}^n. \tag{19}$$

The function f is also strongly convex and has a unique minimizer.

Choose an arbitrary set of $n + 2$ points from G:

$$\tau_0 = \{y_{01}, \ldots, y_{0,n+2}\}, \quad y_{0i} \in G, \quad \forall i \in 1, \ldots, n + 2.$$

The set τ_0 will be called a *basis*.

Assume that the points $x_0, \ldots, x_{k-1}$ and the basis $\tau_k = \{y_{k1}, \ldots, y_{k,n+2}\}$ have already been found.

Let us define the function

$$f_k(x) = \max_{i \in 1, \ldots, n+2} \varphi(x, y_{ki})$$

and choose a point $x_k \in \mathbf{R}^n$ such that

$$f_k(x_k) = \min_{x \in \mathbf{R}^n} f_k(x). \tag{20}$$

If $f(x_k) = f_k(x_k)$ then x_k is the minimizer of f, and the process terminates.

The minimization problem in (20) is simpler than that of minimizing f.

Consider the case $f(x_k) > f_k(x_k)$. By the necessary and (in our convex case) sufficient condition (7)

$$0_n \in L_k, \tag{21}$$

where $L_k = \operatorname{co} H_k$, $H_k = \{\varphi_x'(x_k, y_{ki}) : i \in R_k\}$, $R_k = \{i \in 1, \ldots, n + 2 : \varphi(x_k, y_{ki}) = f_k(x_k)\}$. By the Carathéodory theorem [7] every point of L_k can be represented as a convex combination of not more than $n + 1$ points of H_k. Therefore, there exists at least one index $i_k \in 1, \ldots, n+2$ such that either $i_k \notin R_k$ or the origin in (21) may be 'constructed' without the vector $\varphi_x'(x_k, y_{ki_k})$.

Let $\overline{y}_k \in G$ be such that

$$\varphi(x_k, \overline{y}_k) = f(x_k) = \max_{y \in G} \varphi(x_k, y).$$

Now let us construct a new basis

$$\tau_{k+1} = \{y_{k+1,1}, \ldots, y_{k+1,n+2}\}$$

where

$$y_{k+i,i} = \begin{cases} y_{ki}, & i \neq i_k, \\ \overline{y}_k, & i = i_k. \end{cases}$$

The basis τ_{k+1} differs from τ_k by one point and also contains $n + 2$ points. For the basis τ_{k+1} again define the function $f_{k+1}(x)$ and the point x_{k+1}.

As a result, a sequence $\{x_k\}$ is constructed. If this sequence is finite, its last point is the minimizer. If not, the following property holds.

Proposition 7 (See [6 Sect. III.10].) *The sequence $\{x_k\}$ converges to the minimum point of f.*

Remark 8 The extremal basis method can be extended (with necessary adjustments) to the case where φ is just convex at x, not necessarily strongly convex.

Remark 9 If the function φ in (1) is not convex then condition (7) and the Carathéodory theorem produce the following properties:

1) There exist points $y_1, \ldots, y_{k+1}$ such that $y_i \in Y$ for any $i \in 1, \ldots, k + 1$ and a minimizer x^* of f is an inf-stationary point of the function

$$F(x) = \max_{i \in 1, \ldots, k+1} \varphi(x, y_i).$$

2) There exist points $y_1, \ldots, y_{k+1}$ and coefficients $\alpha_1, \ldots, \alpha_{k+1}$ such that

$$y_i \in Y, \quad \alpha_i \geq 0, \quad \forall i \in 1, \ldots, k + 1, \quad \sum_{i \in 1, \ldots, k+1} \alpha_i = 1,$$

and a minimizer x^* is a stationary point of the smooth function

$$L(x) = \sum_{i \in 1, \ldots, k+1} \alpha_i \varphi(x, y_i).$$

These properties can be used to derive corresponding numerical algorithms.

Constrained Minimax Problems

Let f be defined by (1) on an open set $S \subset \mathbf{R}^n$ and $\Omega \subset S$ be a closed set. The problem is to find

$$\min_{x \in \Omega} f(x) = f^*.$$

Take $x \in \Omega$. The set

$$\Gamma(x, \Omega) = \left\{ g \in \mathbf{R}^n : \begin{array}{c} \exists \{[\alpha_k, g_k]\} : \\ [\alpha_k, g_k] \to [+0, g], \\ x + \alpha_k g_k \in \Omega, \\ \forall k \end{array} \right\}$$

is the *Bouligand cone* to Ω at x. It is nonempty and closed.

The function f is subdifferentiable (see (6)).

The following necessary condition holds ([4,5]):

For a point $x^* \in \Omega$ to be a minimizer of f on Ω it is necessary that

$$f'(x^*, g) \geq 0, \quad \forall g \in \Gamma(x^*, \Omega). \tag{22}$$

The cone $\Gamma(x^*, \Omega)$ may be represented in the form

$$\Gamma(x^*, \Omega) = \bigcup_{i \in I} A_i, \tag{23}$$

where the A_i are closed convex cones, I is some set (e. g., $\Gamma(x^*, \Omega)$ can always be given as the union of all its rays). Of course, from practical considerations we are interested to have as few elements in I as possible (the best case is the one where Ω is convex, then $\Gamma(x^*, \Omega)$ is also convex).

Taking into account (6), condition (22) can be rewritten in the equivalent form

$$\partial f(x^*) \cap A_i^+ \neq \emptyset, \quad \forall i \in I, \tag{24}$$

where A_i^+ is the cone conjugate to A_i:

$$A_i^+ = \{v \in = \mathbf{R}^n : \ (v, g) \geq 0, \ \forall g \in A_i\}.$$

A point $x^* \in \Omega$ satisfying (24) is called an *inf-stationary point* of f on Ω.

Let $x \in \Omega$ be not inf-stationary. For every $i \in I$ let us find

$$\min_{\substack{v \in \partial f(x) \\ w \in A_i^+}} \|v - w\| = \|v_i - w_i\| = \rho_i \tag{25}$$

and

$$\max_{i \in I} \rho_i = \rho_{i_0} = \|v_{i_0} - w_{i_0}\|. \tag{26}$$

Then the direction

$$g_{i_0}(x_0) = \frac{w_{i_0} - v_{i_0}}{\rho_{i_0}}$$

is a direction of steepest descent of the function f at the point x_0 (on the set Ω):

$$f'(x, g_{i_0}(x_0)) = \min_{\substack{g \in \Gamma(x_0, \Omega) \\ \|g\| = 1}} f'(x, g).$$

It may happen that there exist several steepest descent directions (s.d.d.). If $\Gamma(x_0, \Omega)$ is convex then such a direction is unique.

Many numerical methods for minimizing f on Ω employ condition (24) (see [2,6,10,11,12]).

Let us discuss in detail the case where Ω is described in the form

$$\Omega = \{x \in S : \ h(x) \leq 0\}, \tag{27}$$

where

$$h(x) = \max_{z \in G_1} \psi(x, z),$$

G_1 is a compact set of some space Z, $\psi : S \times G_1 \to \mathbf{R}^n$ is continuous jointly in x and z on $S \times G_1$ and is continuously differentiable in x. Assume that Ω is nonempty. Note that Ω is closed.

The function h is also subdifferentiable and

$$\partial h(x) = \operatorname{co}\{\psi'_x(x, z) : \ z \in Q(x)\},$$
$$Q(x) = \{z \in G_1 : \ \psi(x, z) = h(x)\}.$$

Note that if $h(x) = 0$ and $0_n \notin \partial h(x)$ then

$$\Gamma^+(x, \Omega) = \{v = \lambda w : \ \lambda \leq 0, \ w \in \partial h(x)\}.$$

If $h(x) < 0$ the $\Gamma^+(x, \Omega) = \{0_n\}$.

The cone $\Gamma(x, \Omega)$ is convex and closed and, hence, there exists only one steepest descent direction.

The following necessary condition is true: Let $x^* \in \Omega$, $h(x^*) = 0$. For a point x^* to be a minimizer of f on Ω it is necessary that

$$0_n \in \operatorname{co}\{\partial f(x^*), \partial h(x^*)\} = L(x^*). \tag{28}$$

If $0_n \notin \partial h(x^*)$ then conditions (22) and (28) are equivalent. If $0_n \in \partial h(x^*)$ then (28) holds automatically but (22) does not.

If $0_n \notin L(x_0)$, $h(x_0) = 0$ then the direction

$$g(x_0) = -\frac{v(x_0)}{\|v(x_0)\|}$$

where

$$\|v(x_0)\| = \min_{v \in L(x_0)} \|v\|$$

is a descent direction and it is an admissible direction, i. e. there exists $\alpha_0 > 0$ such that $x_0 + \alpha\, g(x_0) \in \Omega$, $\forall \alpha$

$\in [0, \alpha_0]$. Observe that the steepest descent direction is not necessarily admissible.

Note also that for any $\lambda > 0$ condition (28) is equivalent to

$$0_n \in \operatorname{co}\{\partial f(x^*), \lambda \partial h(x^*)\} = L_\lambda(x^*).$$

Method of Hypodifferential Descent

The function h (as well as f) is continuously hypodifferentiable with the hypodifferential (cf. (11))

$$dh(x) = \operatorname{co}\left\{ [a, v] : \begin{array}{c} a = \psi(x, z) - h(x), \\ v = \psi'_x(x, z), \\ z \in G_1 \end{array} \right\}. \quad (29)$$

Let

$$L(x) = \operatorname{co}\{df(x), dh(x) + [h(x), 0_n]\}.$$

Proposition 10 *For a point $x^* \in \Omega$ to be a minimizer of f on Ω it is necessary that*

$$0_{n+1} \in L(x^*). \quad (30)$$

A point $X^* \in \Omega$ satisfying (30) is an inf-stationary point of f on Ω.

If $x \in \Omega$ is not inf-stationary, then

$$\rho(x) = \min_{\overline{z} \in L(x)} \|z\| = \|\overline{z}(x)\| > 0.$$

Here

$$\overline{z}(x) = [\eta(x), z(x)],$$
$$\eta(x) \in \mathbf{R}, \quad z(x) \in \mathbf{R}^n, \quad z(x) \neq 0_n.$$

The direction $g(x) = -z(x)/\|z(x)\|$ is an admissible descent direction. The vector-function $z(x)$ is continuous on Ω.

Let us describe the following method (see [5, Chap. 5, Sect. 5]):

Take any $x_0 \in \Omega$. Let $x_k \in \Omega$ have already been defined. If $\rho(x_k) = 0$ then x_k is inf-stationary. If $\rho(x_k) > 0$ then let us find

$$\min_{\substack{\alpha \geq 0, \\ (x_k - \alpha z(x_k)) \in \Omega}} f(x_k - \alpha z(x_k)) = f(x_k - \alpha_k z(x_k)).$$

Now put

$$x_{k+1} = x_k - \alpha_k z(x_k).$$

By construction $x_k \in \Omega$, $\forall k$.

If the sequence $\{x_k\}$ is finite, its last point is an inf-stationary one. If it is infinite the following result holds:

Proposition 11 *If the set*

$$\{x \in \Omega : f(x) \leq f(x_0)\}$$

is bounded, then any limit point of the sequence $\{x_k\}$ is inf-stationary.

Remark 12 The method described is 'conceptual'. For its practical implementation it is necessary to avoid the computation of df and dh (by (11) and (29)) and take some smaller sets (since the hypodifferential mapping is not uniquely defined).

Remark 13 If both functions φ and ψ are convex in x then the extremal basis method (given above) can be extended for minimizing f on Ω (see [6]).

The Kelley Method

Let us consider the problem of minimizing a function

$$f(x) = \max_{y \in G} \varphi(x, y) = \max_{y \in G} [\varphi_1(x, y) + f_1(x)]$$

on a convex compact set $\Omega \in \mathbf{R}^n$, where $\varphi_1 : \Omega \times G \to \mathbf{R}$ is convex in x on Ω for any $y \in G$ and continuous in y on G and $f_1 : \Omega \mathbf{R}$ is continuous on Ω. Then the following modification of the *Kelley cutting plane method* [8] can be used:

Choose any $x_0 \in \Omega$ and find $y_0 \in G$ such that $\varphi(x_0, y_0) = f(x_0)$. Take any $v_0 \in \partial\varphi_1(x_0, y_0)$ (where $\partial\varphi_1(x_0, y_0)$ is the subdifferential of the convex function $\varphi_1(x, y_0)$ at x_0). Put

$$B_0(x) = f_1(x) + \varphi_1(x_0, y_0) + (v_0, x - x_0).$$

Let $x_k \in \Omega$ have already been defined. Find $y_k \in G$ such that $\varphi(x_k, y_k) = f(x_k)$, take any $v_k \in \partial\varphi_1(x_k, y_k)$ and put

$$B_k(x) = f_1(x) + \varphi_1(x_k, y_k) + (v_k, x - x_k).$$

Let

$$f_k(x) = \max_{i\in 0,\dots,k} B_i(x)$$
$$= f_1(x) + \max_{i\in 0,\dots,k} [\varphi_1(x_i, y_i) + (v_i, x - x_i)].$$

Find

$$\min_{x\in\Omega} f_k(x) = f_k(x_k^*).$$

If $f_k(x_k^*) = f(x_k^*)$ then x_k^* is a minimizer of f on Ω and the process terminates. Otherwise, take $x_k^* = x_{k+1}$ and proceed as above. If the sequence $\{x_k\}$ thus constructed is finite, its last point is a minimizer. If not, the following statement holds.

Proposition 14 *Any limit point of the sequence $\{x_k\}$ is a minimizer of f on Ω.*

See also

- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Dini and Hadamard Derivatives in Optimization
- ▶ Global Optimization: Envelope Representation
- ▶ Minimax: Directional Differentiability
- ▶ Minimax Theorems
- ▶ Nondifferentiable Optimization
- ▶ Nondifferentiable Optimization: Cutting Plane Methods
- ▶ Nondifferentiable Optimization: Newton Method
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Nondifferentiable Optimization: Relaxation Methods
- ▶ Nondifferentiable Optimization: Subgradient Optimization Methods
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Quasigradient Methods in Minimax Problems

References

1. Danskin JM (1967) The theory of max-min and its application to weapons allocation problems. Springer, Berlin
2. Daugavet VA, Malozemov VN (1981) Quadratic rate of convergence of one linearization method for solving descrete minimax problems. USSR J Comput Math Math Phys 21(4):835–843
3. Demyanov VF (1995) Fixed point theorem in nonsmooth analysis and its applications. Numer Funct Anal Optim 16(1–2):53–109
4. Demyanov VF, Malozemov VN (1974) Introduction to minimax. Wiley, New York
5. Demyanov VF, Rubinov AM (1995) Constructive nonsmooth analysis. P. Lang, Frankfurt am Main
6. Demyanov VF, Vasiliev LV (1986) Nondifferentiable optimization. Springer and Optim. Software, Berlin
7. Karlin S (1959) Mathematical methods and theory in games, programming and economics. Addison-Wesley, Reading
8. Kelley JF (1960) The cutting-plane method for solving convex problems. SIAM J Appl Math 8(4):703–712
9. Laurent P-J (1972) Approximation et optimisation. Hermann, Paris
10. Panin VM (1981) On some methods for solving convex programming problems. USSR J Comput Math Math Phys 21(2):315–328
11. Polak E (1987) On the mathematical foundations of nondifferentiable optimization in engineering design. SIAM Rev 29:21–89
12. Pschenichny BN (1983) The method of linearization. Nauka, Moscow (In Russian)

Nondifferentiable Optimization: Newton Method

A. M. Rubinov
School Inform. Techn. and Math. Sci. University Ballarat, Ballarat, Australia

MSC2000: 49J52, 90C30

Article Outline

Keywords
See also
References

Keywords

Newton method; Variational inequality problem; Nonlinear complementary problem; Smoothing; Approximation of nonsmooth mappings; Semismooth mapping; Superlinear convergence

One of the main approaches to solving equations and unconstrained optimization problems with differentiable functions involved is based on the *Newton*

method (NM). The classical version of this method (in a finite-dimensional setting) deals with an equation

$$F(x) = 0 \tag{1}$$

with a differentiable mapping F. Having an approximate solution x_k ($k = 0, 1, \dots$) we wish to find a new one $x_{k+1} = x_k + y$ which is 'better' than x_k. Since there exists the derivative ∇F, we can approximate the mapping $y \longmapsto F(x_k + y)$ by the mapping $y \longmapsto F(x_k) + \nabla F(x_k)y$ and hence approximate the equation $F(x_k + y) = 0$ by the equation

$$F(x_k) + \nabla F(x_k)y = 0\,. \tag{2}$$

It is assumed that the linear mapping $\nabla F(x_k)$ is invertible. A new approximation x_{k+1} to the solution has a form $x_{k+1} = x_k + y_k$ where $y_k = -(\nabla F(x_k))^{-1}(F(x_k))$ is a solution of the equation (2). (Sometimes it is more convenient to add a degree of freedom and consider a vector $x_{k+1} = x_k + t_k y_k$ with $t_k > 0$ as a new approximation.) Thus a classical smooth version of the NM has a form

$$x_{k+1} = x_k - (F'(x_k))^{-1}(F(x_k))\,. \tag{3}$$

There are many methods for solving an unconstrained minimization problem

$$f(x) \to \min \tag{4}$$

based on the scheme (3) and its modifications. The simplest scheme:

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \tag{5}$$

is suitable for twice-continuously differentiable (C^2) functions f. This scheme allows us to find critical points of the function f.

As it turns out many optimization and related problems even with smooth objective functions and constraints can be reduced to the solution of special kinds of *nonsmooth equations* (see, for example, [20]).

We shall consider as an example a *variational inequality problem* (VIP): find a vector $x \in D$ such that

$$(y - x)^\top F(x) \geq 0 \quad \text{for all } y \in K\,, \tag{6}$$

where K is a closed convex subset of $\mathbf{R}^n$ and F is a continuously differentiable (C^1) mapping defined on an open set $D \supset K$. Let P_K be the *metric projection* onto the set C (that is $\| x - P_K x \| = \min_{y \in K} \| x - y \|$ with the Euclidean norm $\| \cdot \|$) and

$$\begin{aligned} \widetilde{F}_K(x) &= x - P_K(x - F(x)), \\ F_K(z) &= F(P_K(z)) + (z - P_K(z)). \end{aligned} \tag{7}$$

It can be shown (see, for example, [20]) that a vector x satisfies (6) if and only if x is a solution of the equation $\widetilde{F}_K(x) = 0$ and if and only if $z = x - F(x)$ is a solution of the equation $F_K(z) = 0$. Since P_K is not necessarily a smooth mapping it follows that both mappings $\widetilde{F}_K$ and F_K are not necessarily smooth.

A special case of VIP (6) with $K = \mathbf{R}^n_+$ is a *nonlinear complementary problem* (NCP): find a vector $x \in D$ such that

$$x \geq 0, \quad F(x) \geq 0, \quad x^\top F(x) = 0\,, \tag{8}$$

where $D \subset \mathbf{R}^n$ is an open set containing the nonnegative orthant $\mathbf{R}^n_+$ and F is a continuously differentiable (C^1) function defined on D. Since $P_{\mathbb{R}^n_+}(x) = x^+$, the mappings (7) have the following form:

$$\begin{aligned} \widetilde{F}_K(x) &= \min(x, F(x)), \\ F_K(z) &= F(x^+) - x^-, \end{aligned} \tag{9}$$

where $x^+ = \max(x, 0)$ and $x^- = \min(x, 0)$; max and min stand for componentwise maximum and minimum, respectively. Thus NCP reduces to equations with max and min operators.

The following properties of the function $m(b, c) = \min(b, c)$ are important for application to NCP: m is positive homogeneous of the first degree and the set $\{x = (x_1, x_2)$: $x_1 \geq 0$, $x_2 \geq 0$, $m(x) = 0\}$ coincides with the union of two positive semi-axes. Sometimes it is more convenient to consider functions with the same properties and also being C^1 on $\mathbf{R}^2$ except the origin. One example from this large class of functions is the so-called *Fischer–Burmeister function* [8])

$$\phi(b, c) = (b^2 + c^2)^{1/2} - (b + c)\,. \tag{10}$$

If

$$H(x) = (\phi(x_1, F_1(x)), \dots, \phi(x_n, F_n(x)))^\top\,, \tag{11}$$

then x is a solution of NCP if and only if $H(x) = 0$.

Thus it became necessary to extend the NM to enable the efficient solution of nonsmooth equations.

Note that this extension is not generally possible. B. Kumer [15] has found an example of a function F defined on the real line, which enjoys very good properties (F is Lipschitz, strictly monotone, directionally differentiable everywhere and Fréchet differentiable at the solution of the equation (1)) and such that NM alternates for almost all starting points.

One of the first contributions to nonsmooth NM was given by S.M. Robinson [27] for solving feasibility problems involving inequality constraints and by M. Kojima and S. Shindo [14] for solving (1) with a piecewise smooth mapping F. Various versions and modifications of the NM for nonsmooth equations have been developed during the last decade. In particular *damped NM* and *smoothing NM* enjoy global convergence under certain monotonicity assumptions. There are finite-dimensional and infinite-dimensional settings of this problem. Here we consider only finite-dimensional versions of the nonsmooth NM. For infinite-dimensional problems see for example [7,16].

The first problem which arises in the study of nonsmooth NM is to find suitable *approximations of nonsmooth mappings*. Many authors suggested various versions of such approximations. Two types of approximations are mainly considered. One of them is an approximation by a certain set-valued mapping $x \to V(x)$ where $V(x)$ is a set of invertible matrices. In this case the equation (2) is replaced by a linear equation $F(x_k) + A_k y = 0$, where A_k is an arbitrary matrix from $V(x_k)$. Thus NM in such a setting has a form

$$x_{k+1} = x_k - A_k^{-1} F(x_k). \tag{12}$$

The second type is based on an approximation by means of a certain nonlinear mapping F'. In such a case the method can be presented in the following form: $x_{k+1} = x_k + y_k$, where y_k is a solution of the nonlinear equation

$$F(x_k) + F'(x_k)(y) = 0. \tag{13}$$

It is assumed that the auxiliary nonlinear equation (13) can be solved relatively easily.

There is a general approach [16] based on the set-valued approximation $\mathcal{F}(x, y)$ of a single-valued locally Lipschitz mapping F which includes both of the above mentioned types. This approach clarifies two conditions which lead to convergence of NM: first the uniform injectivity (invertibility) of the approximating mapping at a neighborhood Y of a solution x^*: there exists $c > 0$ such that

$$\|u\| \geq c\,\|y\| \quad \text{for all } y \in Y,\ u \in \mathcal{F}(x, y), \tag{14}$$

and secondly, that the mapping F should accomplish a relatively good approximation at this neighborhood:

$$F(x) + \mathcal{F}(x, y) \subset \mathcal{F}(x, y + (x - x^*) + o(\|x - x^*\|), \tag{15}$$

where $o(y)/\|y\| \to 0$ as $\|y\| \to 0$.

It can be shown (see [16]) that for many concrete situations (15) is equivalent to

$$F(x) + \mathcal{F}(x, x^* - x) \subset o(\|x - x^*\|)B, \tag{16}$$

where B is the unit ball. The local speed of convergence is determined by the number c and the function o in (15). Conditions (14) and (15) are necessary for convergence of NM under some additional assumptions ([16]).

We now turn to the first type of approximation. If F is a locally Lipschitz mapping then the *B-subdifferential* $\partial_B F(x)$ and *Clarke generalized Jacobian* $\partial_{\mathrm{Cl}} F(x)$ are considered as approximations. By definition

$$\partial_B F(x) = \left\{ \lim_{x_k \to x} \nabla F(x_k) : \begin{array}{c} F \text{ differentiable} \\ \text{at } x_k \end{array} \right\}$$

and $\partial_{\mathrm{Cl}} F(x)$ is the convex hull of $\partial_B F(x)$. L. Qi suggested another approximation, the *C-differential* [23], which can be applied to all continuous (not just Lipschitz) mappings. By definition the C-subdifferential T is a compact-valued upper-semicontinuous mapping such that $F(x + u) = F(x) + A(u) + o(u)$ for any $A \in T(x + u)$. (It is important that T is a mapping, not an individual set at the point x; approximation near the point x is accomplished by matrices from the sets $T(y)$ for all y sufficiently close to x.) Close construction of point-based set-valued approximation is studied in [30], where connections with constructions from [16,28] are mentioned. There exist exact calculus rules for C-differentials which allow their relatively easy computation. An interesting example of approximation is given by *approximate Jacobians* (see, for example, [10]). This construction is again applicable for all continuous mappings and allows the unification of various approaches to approximation.

Local convergence based on a matrix approximation $V(x)$ can be proved only if all matrices A belonging to $V(x)$ with x sufficiently close to a solution x^*, accomplish a sufficiently good approximation (compare with [16]). Convergence can be, in particular, proved if the mapping F is semismooth (see [25]) at the point x^* or enjoys some properties close to semismoothness (see, for example, [10]). The mapping F is called *semismooth* with respect to a matrix approximation $V(x)$ at a point x^* if the directional derivative $F'(x, \cdot)$ exists at all points x close to x^* and

$$\begin{aligned} &Au - F'(x,u) = o(\|u\|), \\ &A \in T(x+u), \quad u \to 0, \end{aligned} \tag{17}$$

that is, each matrix $A \in T(x+u)$ approximates the directional derivative at the point x in the direction u. (Semismoothness was originally introduced by R. Mifflin for real-valued functions in 1977.) In some applications strong semismoothness is required. The mapping F is called *strongly semismooth* at a point x if there exists a number K such that

$$\begin{aligned} &\|Au - F'(x,u)\| \le K\|u\|^2, \\ &A \in T(x+u), \quad u \to 0. \end{aligned} \tag{18}$$

Semismoothness and strongly semismoothness can be easily verified in many concrete situations. The simplest example of a strongly semismooth mapping is a coordinatewise maximum (or minimum) of a finite number of C^2 mappings.

It is well known that the NM produces for differentiable mappings a sequence which converges to a solution very fast. As it turns out this property holds also in nonsmooth setting under some regularity conditions. Qi and J. Sun demonstrated that for semismooth mappings the rate of convergence is *Q-superlinear* [25], that is

$$\lim_{k\to+\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0.$$

Under some additional assumptions (for example, strong semismoothness) [25] it can be shown that the rate of convergence is *Q-quadratic*:

$$\lim_{k\to+\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} < +\infty.$$

The so-called *Kantorovich scheme* [12] in the study of NM can also be extended for nonsmooth equations (see for example [16,25,27,29]). This scheme allows one to prove not only the convergence of the NM but also the existence of a solution in a small neighborhood of the starting point x_o if some combinations of such parameters as $\|F(x_o)\|$ and the uniform boundary of norm of inverse matrices, the value K in (18), the radius of neighborhood, are sufficiently small.

Various kinds of nonlinear approximation are used for setting the NM in the framework of the second approach.

The *B-derivative* [19,22], that is uniformly (with respect to direction) directional derivative, is often used as a nonlinear approximation. One more tool for such an approximation is the *codifferential* [5]. One of the difficult tasks arising under application of the NM based on nonlinear approximation is to solve the auxiliary subproblem (13). For B-derivative based NM, J.S. Pang [19] proposed solving this problem inexactly. A version of NM under the assumption that each matrix of $\partial_B F(x)$ is invertible (the so-called *BD*-regularity) was studied in [22]. This assumption is weaker than invertibility of B-derivative and also nonsingularity of all matrices of $\partial_{Cl} F(x)$. A different approach to nonlinear approximation was proposed by Robinson [28], who introduced the so-called *point-based approximation*. This approach is convenient in the study of equations with mappings F_K (see (7)) generated by the projection P_K on a convex set K. Q-superlinear convergence and (for strong approximations) Q-quadratic convergence can be proved for various kinds of nonlinear approximations.

An abstract version of the second approach, based on the Kantorovich scheme, was proposed in [29]. The mapping $F^{\odot}(x, v)$ is called an *approximator* for the mapping F at the point x in the direction v if $F(x+v) = F(x) + F^{\odot}(x, v) + o(v)$ where $\lim_{t\to+0} t^{-1}o(tv) = 0$. Assume that an approximator $F^{\odot}$ accomplishes a strong approximation: there exist a number K such that $\|F(x+u) - F(x) - F^{\odot}(x, u)\| \le K\|u\|^2$ for all x close to the starting point x_0, and $F^{\odot}$ enjoys the following property (which is weaker than the pseudo-Lipschitz property [1]): there exists a number L such that the equation $F^{\odot}(x, u) = y$ has a solution $\overline{u}$ with $\|\overline{u}\| \le L\|y\|$ for all sufficiently small y. Then the convergence of the NM based on $F^{\odot}$ can be proved [29] by means of the Kantorovich scheme under some typical for this scheme assumptions. The inequality $\|x^* - x_k\| \le \gamma$

2^{-k} holds for the Newton sequence (x_k) with a number $\gamma > 0$.

As a rule NM converges to a solution (even in the smooth setting) only if a starting point is sufficiently close to this solution. Thus the question arises: is it possible to find modifications of NM which provide *global convergence* for some equations, that is convergence from an arbitrary initial point? One of such modifications is the *damped Newton method*. For instance, if the NM uses a positively homogeneous approximation $F'(x_k, u)$, such as a Jacobian in the smooth case, or directional derivative [7,13,15,19], then x_{k+1} is chosen as $x_k + \lambda_k y_k$ for some $\lambda_k \in (0, 1]$ such that

$$\|F(x_{k+1})\| \cong (1 - \lambda_k)\,\|F(x_k)\| < \|F(x_k)\|\,. \qquad (19)$$

This can be generalized to nonpositively homogeneous approximations such as point based approximations [3,21] by setting up a path $p_k(\lambda)$ joining $x_k = p_k(0)$ to $x_k + y_k = p_k(1)$, where y_k is found by the NM, such that $F(p_k(\lambda)) = (1 - \lambda)\,F(x_k) + o(\lambda)$. Then it is easy to determine $\lambda_k \in (0, 1]$ such that for $x_{k+1} = p_k(\lambda_k)$, (19) holds.

There is a close connection between the damped NM and the so-called *list square merit function* of the operator F:

$$\theta(x) = \frac{1}{2}\,\|F(x)\|^2\,.$$

For some nonsmooth operators F arising from NCP and related problems the function θ is continuously differentiable. This property is very useful in the study of NM and damped NM [11].

One of modifications of the nonsmooth NM is the so-called *smoothing Newton method*, which is also called *splitting NM* or *homotopy NM*. This method based on an approximation of the operator F in (1) by a smooth function G. Usually a smoothing damped NM is studied. This method has the following form

$$x_{k+1} = x_k - t_k \nabla G_x(x_k, \varepsilon_k)^{-1} F(x_k), \qquad (20)$$

where $G(x, \varepsilon)$ is a smoothing approximation of the mapping F, that is for any $\varepsilon > 0$ the function $x \longmapsto G(x, \varepsilon)$ is continuously differentiable and $\|F(x) - G(x, \varepsilon)\| \to 0$ as $\varepsilon \to 0$.

Various types of *smoothing functions* are used in optimization for a long time (see, for example, [13]). Some of them can be constructed by means of the so-called *Chen–Harker–Kanzow–Smale function*:

$$\xi(u, \varepsilon) = \frac{1}{2}\left((u^2 + 4\varepsilon^2)^{\frac{1}{2}} + u\right), \quad (u, e) \in \mathbb{R}^2,$$

which serves for an approximation of $\max(0, u)$ (see, for example, [24]). In particular, it is possible to find smoothing operators for operators defined by (9) by means of the function ξ. An interesting approach to the smoothing methods can be found in [3]. Global convergence of the method (20) can be proved under some assumptions. If the gradient of the smoothing function G is fairly close to some approximations of the operator F then the method converges Q-superlinearly (quadratically) [4,24].

There is a close connection between smoothing NM and *interior point methods* (see, for example, [21]).

For some special classes of problems it is possible to obtain stronger results than in the general setting (see, for example, [6,11,18,26]). On the other hand there are modifications of the nonsmooth Newton method for the solution of generalized equations of the form $y \in F(x)$ with a set-valued mapping F (see, for example, [2,7,16]).

See also

- Automatic Differentiation: Calculation of Newton Steps
- Dini and Hadamard Derivatives in Optimization
- Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- Global Optimization: Envelope Representation
- Interval Newton Methods
- Nondifferentiable Optimization
- Nondifferentiable Optimization: Cutting Plane Methods
- Nondifferentiable Optimization: Minimax Problems
- Nondifferentiable Optimization: Parametric Programming
- Nondifferentiable Optimization: Relaxation Methods
- Nondifferentiable Optimization: Subgradient Optimization Methods
- Unconstrained Nonlinear Optimization: Newton–Cauchy Framework

References

1. Aubin J-P, Ekeland I (1984) Applied nonlinear analysis. Wiley, New York
2. Aze D, Chou CC (1995) On a Newton type iterative methods for solving inclusions. Math Oper Res 20:790–800
3. Chen C, Mangasarian OL (1996) A class of smoothing functions for nonlinear and mixed complementarity problems. Comput Optim Appl 5:97–138
4. Chen X, Qi L, Sun D (1998) Global and superlinear convergence of the smoothing Newton method and its application to general box constrained variational inequalities. Math Comput 67:519–540
5. Demyanov VF (1995) Fixed point theorem in nonsmooth analysis and its applications. Numer Funct Anal Optim 16:53–109
6. Dirkse SP, Ferris MC (1995) The path solver: A nonmonotone stabilization scheme for mixed complementary problems. Optim Methods Softw 5:123–156
7. Dontchev AL (1996) Local convergence of the Newton method for generalized equations. CR Acad Sci Paris Ser I 322:327–331
8. Fisher A (1992) A special Newton-type optimization method. Optim 24:269–284
9. Harker PT, Xiao B (1990) Newton's method for nonlinear complimentary problem: A B-differentiable equation approach. Math Program 48:339–357
10. Jeyakumar V, Luc DT (1998) Approximate Jacobian matrices for nonsmooth continuous maps and C^1-optimization. SIAM J Control Optim 36:1815–1832
11. Jiang H, Ralph D (1998) Global and local superlinear convergence analysis of Newton-type methods for semismooth equations with smooth least squares. In: Fukushima M, Qi L (eds) Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods. Kluwer, Dordrecht, pp 181–210
12. Kantorovich L, Akilov G (1964) Functional analysis in normed spaces. MacMillan, New York
13. Kaplan AA (1981) On an approach to the solution of convex programming problems. Soviet Math Dokl 23:588–591
14. Kojima M, Shindo S (1986) Extensions of Newton and quasi-Newton methods to system of PC^1 equations. J Oper Res Soc Jpn 29:352–374
15. Kummer B (1988) Newton's method for non-differentiable functions. In: Guddat J (ed) Adv Math Optim. Akademie, Berlin, pp 114–125
16. Kummer B (1992) Newton's method based on generalized derivatives for nonsmooth functions: convergence analysis. In: Oettli W, Pallaschke D (eds) Advances in Optimization. Springer, Berlin, pp 171–194
17. Kummer B (1995) Approximation of multifunctions and superlinear convergence. In: Durier R, Michelot C (eds) Recent Developments in Optimization. Springer, Berlin, pp 243–251
18. De Luca T, Facchinei F, Kanzow C (1996) A semismooth equation approach to the solution of nonlinear complementary problems. Math Program 75:407–439
19. Pang JS (1990) Newton's method for B-differentiable equations. Math Oper Res 15:311–341
20. Pang JS, Qi L (1993) Nonsmooth equations: motivation and algorithms. SIAM J Optim 3:443–465
21. Potra E, Ye Y (1996) Interior-point methods for nonlinear complementary problems. J Optim Th Appl 88:617–642
22. Qi L (1993) Convergence analysis of some algorithms for solving nonsmooth equations. Math Oper Res 18:227–244
23. Qi L (1996) C-differential operators, C-differentiability and generalized Newton methods. Applied Math Report. Univ New South Wales, Sydney
24. Qi L, Sun D (1999) Nonsmooth equations and smoothing Newton methods. In: Eberhard A et al (eds) Progress in Optimization: Contribution from Australasia. Kluwer, Dordrecht, pp 121–146
25. Qi L, Sun J (1993) A nonsmooth version of Newton's method. Math Program 58:353–367
26. Ralph D (1994) Global convergence of damped Newton's method for nonsmooth equations, via the path search. Math Oper Res 19:352–389
27. Robinson SM (1994) Newton's method for a class of nonsmooth functions. Set-Valued Anal 2:291–305
28. Robinson SM (1972) Extension of Newton's method to nonlinear functions with values in a cone. Numerische Math 19:341–347
29. Rubinov A, Zaffaroni A (1999) Continuous approximation of nonsmooth mappings. In: Eberhard A et al (eds) Progress in Optimization: Contribution from Australasia. Kluwer, Dordrecht, pp 27–58
30. Xu H (1999) Set-valued approximations and Newton's methods. Math Program 84:401–420

Nondifferentiable Optimization: Parametric Programming

SANJO ZLOBEC
Department Math. Statist., McGill University, West Montreal, Canada

MSC2000: 90C05, 90C25, 90C29, 90C30, 90C31

Article Outline

Keywords
Historical Outline
Stability
Optimality and Numerical Methods

Applications
See also
References

Keywords

Parametric programming model; Sensitivity; Stability; Optimal parameter; Parameter identification; Point-to-set mapping

Mathematical descriptions of real-life problems are typically stated in terms of *decision variables* x and parameters θ. These objects may be related through a system of equations and inequalities such as

$$x \in \mathcal{F}(\theta) = \left\{ x \in \mathbf{R}^n : \quad \begin{array}{l} g^i(x,\theta) \le 0,\ i \in I, \\ h^j(x,\ \theta) = 0,\ j \in J \end{array} \right\},$$

where g^i, h^j: $\mathbf{R}^n \times \mathbf{R}^p \to \mathbf{R}$, $i \in I$, $j \in J$, are some functions, and I and J are finite index sets. Problems of the form

$$\begin{cases} \min\limits_{(x)} & f(x,\theta) \\ \text{s.t.} & x \in \mathcal{F}(\theta), \end{cases}$$

where θ is allowed to vary over some set F in $\mathbf{R}^p$, are termed *parametric programming models* (PPM). *Parametric programming* (PP) is the study of parametric programming models. A local analysis of these models, around a fixed θ, is referred to as *sensitivity analysis* (SA). In particular, SA is concerned with changes of the minimal value subject to small perturbations of the parameter. Parametric programming is a huge area containing, or closely related to, many topics, such as path following methods (cf. ► Parametric optimization: Embeddings, path following and singularities), sensitivity in semi-infinite programming, constraint qualifications (cf. also ► First order constraint qualifications; ► Second order constraint qualifications), bilevel programming (cf. ► Bilevel programming: Introduction, history and overview), etc. We will refer to some of these topics hereby only in passing. Since every equality constraint can be replaced by two inequalities, one can assume that $J = \emptyset$. Then the model is said to be *linear* (resp. *convex*) if the functions $f(\cdot, \theta)$, $g^i(\cdot, \theta)$: $\mathbf{R}^n \to \mathbf{R}$, $i \in I$, are linear (resp. convex) for every $\theta \in \mathbf{R}^p$.

Historical Outline

Parametric programming has its roots in the study of linear programs:

$$\mathrm{LP} \begin{cases} \min & c^\top x \\ \text{s.t.} & Ax \le b, \quad x \ge 0, \end{cases}$$

where one or more coefficients of the vectors b and c, or the matrix A, are considered as parameters and allowed to vary. The study can be traced to the 1950s literature. According to [16], the right-hand side changes in a linear program were investigated by W. Orchard-Hays in his unpublished Master's thesis (1952). The term 'parametric' LP was used in [41]. The classical problems of SA and PPM dealt mainly with pivoting and the simplex method. A different approach that uses polyhedral structures rather than the simplex method was developed in [46,47,48,49]. A classical parametric problem in LP is to determine the range of perturbations for specific parameters in b and c, that preserve optimal bases. A related problem is to determine the range for which an optimal solution exists. This range is called the 'critical set'. Various approaches to solving these problems have been successfully implemented in commercial software packages and adjusted to particular situations, e. g., data envelopment analysis [45]. It is well known that difficulties may arise when the problem under consideration is degenerate (i. e., when optimal basis is not unique). In that case the commercial packages may provide essentially different results, that is to say, the information could be 'confusing and hardly allows a solid interpretation'; see, e. g., [5], where the claim is demonstrated on a transportation problem. The study of changes of the parameters in [5] is a departure from the classical approach. Instead of employing local analysis and pivoting, the authors make use of the strict complementarity condition and optimal partitioning in order to construct and study the behavior of the *optimal value function* for the right-hand side and objective-function perturbations. They do it for both LP and convex quadratic programming and obtain sharp intervals. Other approaches used to study the effect of perturbations of parameters in LP, including the 'tolerance approach' (where variations may occur simultaneously and independently), are described in [63,64]. The classical texts on sensitivity analysis and *parametric linear programming* include [9,15,37,48,49].

([15] lists 1031 references.) Almost 1000 items only on degeneracy in LP are listed in [17].

A major obstacle to using linear algebra and calculus methods in the study of linear models is that these methods are not suited to explain continuity properties of the model. Indeed, the model does not generally react 'continuously' to continuous changes of the parameters in the coefficient matrix. Let us illustrate such a situation.

Example 1 Consider the model

$$\begin{cases} \min & x \\ \text{s.t.} & \theta x = 0, \\ & -1 \le x \le 1. \end{cases}$$

When $\theta = 0$, then the feasible set $\mathcal{F}(\theta)$ is the segment $-1 \le x \le 1$, the optimal solution is $x = -1$, and the optimal value is -1. For any perturbation $\theta \neq 0$, all three objects jump to zero.

An example of a linear model where the critical set is disjoint and not closed is given in [9, pp. 114–116]. Another one with a matrix of full row rank where both the feasible set and the set of optimal solutions experience jumps under continuous perturbations of the *parameter* in the interior of the critical set is given in [42]. We extend this example to an LP in canonical form with a full row rank matrix below.

Example 2 Consider the model

$$\begin{cases} \min & -x_2, \\ \text{s.t.} & x_1 + x_2 + x_3 = 1, \\ & x_1 + \theta x_2 - x_4 = 1, \\ & x_i \ge 0, \quad i = 1, \dots, 4. \end{cases}$$

A unique optimal solution at $\theta = 1$ is $x = (x_i) = (0, 1, 0, 0)$ and the optimal value is -1. However, for any perturbation $\theta = 1 - \epsilon$, $\epsilon > 0$, the solution jumps to another unique optimal solution $(1, 0, 0, 0)$, and the optimal value becomes zero.

In order to study reactions of a model to continuous perturbations of data, the feasible set and the optimal solutions set can be viewed as images of *point-to-set mappings* with a domain in the space of parameters. Hence the study of continuity of these and related objects, such as *Lagrange multiplier sets*, requires basic tools of point-to-set topology. These tools have been used in mathematical programming sporadically and in different contexts, e. g., in an analysis of convergence of numerical algorithms in [50,65]. After papers such as [26], the point-to-set approach to the study of parametric programming has become standard. The first text on the theory of *nonlinear parametric optimization* is [4], written by several authors from the 'Berlin school of parametric optimization' initiated by F. Nožička. (Twenty nine students obtained doctorates under his supervision.) This text contains 30 pages of bibliography on PP. A classical text on methodology used in perturbation analyses in nonlinear programming is [13], and one on path following methods in PP is [25]. A unified approach to general perturbations with applications to system analysis and numerical optimization is given in [34]. Since the late 1970s there has been an outburst of research activities in PP. For instance, to date (1998) there have been 20 annual symposia on mathematical programming with data perturbations held at the George Washington Univ., and the International Conference on Parametric Optimization and Related Topics is being held bi-annually since 1985. There have been at least 15 books written on parametric programming.

The study of parametric programming can be roughly divided into three general areas: *stability*, optimality and numerical methods, and applications.

Stability

In this area one mainly studies continuity properties of the feasible set $\mathcal{F}(\theta)$, the set of all optimal solutions $\mathcal{F}^\circ(\theta) = \{x^\circ(\theta)\}$, the set of all Lagrange multipliers $U = U(\theta)$, and the optimal value $f^\circ(\theta) = f(x^\circ(\theta), \theta)$, as the parameter θ varies. Here f° is a function, while $\mathcal{F}$: $\theta \to \mathcal{F}(\theta)$, $\mathcal{F}^\circ$: $\theta \to \mathcal{F}^\circ(\theta)$, and $U \to U(\theta)$ are point-to-set mappings. If the constraints in a *parametric programming model* are continuous functions, then $\mathcal{F}$ is a closed mapping. In order to guarantee continuity of the mapping $\mathcal{F}$ one requires an extra condition, e. g., that $\mathcal{F}$ be lower-semicontinuous (or, equivalently, open). If the PPM is convex and $\mathcal{F}^\circ(\theta^*) \neq \emptyset$ and bounded, at some θ^*, then continuity of $\mathcal{F}$ locally implies both the existence of optimal solutions and continuity of the *optimal value function*, and the mapping $\mathcal{F}^\circ$ is closed. However, continuity of $\mathcal{F}$ does not imply continuity of U.

Conditions for the existence of a differentiable path of solutions $x^\circ(\theta)$ in a general model, as the parameter θ varies, are given in [14]. They are extended to a continuously differentiable path of saddle points in [13]; see also [19,20]. The results are established by applying an appropriate implicit function theorem to a necessary condition for optimality. The reference [14] also contains an explicit formula for the partial derivatives, as well as approximations based on classical penalty functions. For a state-of-the-art about *Lipschitz stability* for linear and quadratic (also nonconvex) programs up to 1987, see [30]. For a recent (1998) guided tour of sensitivity analysis that yields lower and upper estimates for the optimal value function see [6].

If the feasible set mapping $\mathcal{F}$ is continuous and the set of optimal solutions is nonempty and bounded, then such models are often termed stable. In *convex parametric programming* continuity of $\mathcal{F}$ is implied by *Slater's condition*. In the fundamental paper [51], stability for linear systems in the canonical form, subject to perturbations of all data, is characterized by the existence of a positive feasible solution $x > 0$. The result is stated for linear inequalities in partially ordered Banach spaces. It is extended in [52] to nonlinear inequalities over a closed convex set. When perturbations of specific coefficients are considered, then the existence of a positive solution is not a necessary, but it is rather a sufficient condition for stability. Also, in this case, chunks (regions) of the parameter space attached to a fixed parameter θ^*, where $\mathcal{F}$ is lower semicontinuous, are termed *regions of stability* at θ^* see [73,74]. (In the model from Example 2, a region of stability at $\theta^* = 1$ is $\theta \geq 1$.) Such regions can often be calculated globally; one of these is the set of all paths emanating from θ^* on which the constraints satisfy Slater's condition. For a list of regions of stability and a necessary condition for stability in convex PP see [69]. These regions are of independent interest in, e. g., the study of random decision systems with complete connections [66] and linear programming [7]. The radius of the largest ball centered at θ^*, with the property that the model is stable at its every interior point θ, is the *radius of stability* at θ^*, e. g., [69]. It is a measure of how much the system can be uniformly strained from θ^* before it starts breaking down. In a linear parametric programming model in canonical form with a full row rank coefficient matrix, stability is implied by the existence of a nondegenerate basic feasible solution. On the other hand, *instability* (i. e., loss of continuity of the feasible set mapping) typically occurs in situations of 'enforced optima' such as lexicographic and multilevel programming, including *von Stackelberg games* of market economy. For example, in bilevel programming, instability occurs when the optimal solutions set of the follower (lower level decision maker) loses its lower semicontinuity. The leader's (upper level decision maker's) model is then unstable, because its feasible set is the set of optimal solutions of the follower. In this situation the leader's optimal value function typically experiences a discontinuity even if the follower's model is globally stable; see [43, Chapt. 13; 16]. The notion of stability is not uniquely defined, see, e. g. [4,13,21,31,34]. *Structural stability* and continuous deformations of nonlinear programs have been studied in, e. g., [22,28,29,33]. In particular, the *'topological stability'* of the feasible set (i. e., homeomorphy with respect to all sufficiently small perturbations up to second order of the involved functions) is proved to be equivalent to the Mangasarian–Fromovitz constraint qualification being satisfied at the feasible points; see [24]. Characterizations of stability (local existence and uniqueness) of stationary points and Karush–Kuhn–Tucker points with respect to all sufficiently small perturbations up to second order are given, respectively, in [32,53]. For a study of stability with nondifferentiable data see [3,44], also [57]. Some difficulties with an abstract formulation of parametric programming are mentioned in [43, Chapt. 14]; see also [2,58,75]. Stability in optimal control is studied in, e. g., [38,39].

Optimality and Numerical Methods

A parameter θ is said to be *locally* (resp. *globally*) optimal if it locally (resp. globally) optimizes the optimal value function $f^\circ(\theta)$ over its feasible set $F = \{\theta: \mathcal{F}(\theta) \neq \emptyset\}$. Calculation and characterization of *optimal parameters* are basic problems of parametric programming. For convex PPM one can formulate necessary and sufficient conditions for local (and global) optimality of the parameter, e. g., [68,69,70,71]. These conditions are expressed in terms of *saddle-point inequalities* and local results typically require conditions such as uniqueness of the optimal solution in the x component and lower semicontinuity of the feasible set mapping. They

are simplified under *input constraint qualifications* [55]. The optimal value function is not generally known analytically, it is generally nondifferentiable, nonconvex (nonconcave), and discontinuous even for linear models. However, for the right-hand side and objective-function perturbations in linear models it can be constructed, e. g., [5].

Calculation of optimal parameters is often achieved by using the so-called *'marginal value formula'*. This formula gives the derivative of the optimal value function on a prescribed path in terms of the derivatives of the Lagrangian function relative to x and θ. At each iteration the calculation consists of two parts: first, an improvable feasible path is determined and then a step-size problem is solved on the path by a search method. Optimization of the optimal value function by stable perturbations of the parameters (also called *input optimization*) in convex PP is a challenging problem and no satisfactory theory or numerical methods presently seem to exist, e. g., [71]. In contrast, the theory of path following methods, based on nonlinear programming optimality conditions and used when data depend on one scalar parameter, is well developed, although 'not successful in every case' see, e. g., [23,25]. In order to improve the *path following approach* these authors propose jumps between connected components in the sets of local minimizers and generalized critical points.

Applications

Classical sensitivity analysis, when applied to the right-hand side perturbations in linear programming, provides interpretation of Lagrange multipliers as *shadow prices*. The results have been extended to convex programming using the marginal value formula, e. g., in [12]. Genuine applications of parametric programming extend far beyond classical sensitivity analysis. Some of them are related to the fundamental notion of a *well-posed problem* in the sense of Hadamard. These are problems in applied mathematics which have a unique solution and the solution changes continuously when the parameters (e. g., boundary conditions in differential equations) change continuously. Problems that are not well-posed are called *ill-posed*. According to Fritz John (see [62, p. ix]) 'the majority of applied problems are, and always have been, ill-posed, particularly when they require numerical answers'. There is a more general notion of well-posedness (for problems with nonunique solutions); see [10]. Many problems of mathematical physics can be formulated as optimization problems and continuous dependance of the solution on the boundary conditions can be studied using PP. In the context of mathematical programming, some of the first applications of PP included the study of convergence of numerical algorithms. The rate of convergence can be determined using continuity properties of point-to-set mappings; see, e. g., [14,50,54,65]. Some of the ambiguities that occur while solving ordinary LP can be understood and resolved by PP. For example, in some models describing real-life problems, such as the one reported in [62, pp. 212–213], significant *jumps of optimal solutions* occur when some data are perturbed, while the respective values of the objective function are comparatively close. This is a typical behavior of stable linear programming models when the optimal solutions mapping is not continuous. The authors of [62] suggest a method for stabilization of optimal solutions of such programs by 'Tikhonov regularization'.

The results on optimal parameters in parametric programming have many applications including machine scheduling [61,67], restructuring of the work force in a textile mill [71,72], ranking of efficiently administered university libraries by their robustness of data [40,72], the study of systems of differential equations under matrix perturbations in robust analysis and control [27], as well as approximation theory, especially in the problems of *best fitting to data*. These problems are formulated as follows: given a set of points, one wishes to determine a function (from a prescribed class of functions, e. g., linear) that best approximates these points. After forcing the points to satisfy the function, the problem generally reduces to an inconsistent system of equations for which one determines a best approximate solution. The solution is given in terms of *decision variables* (such as the slope of the line and its intersection with the y-axis, in the linear cases of two-dimensional data in the (x, y)-plane). If the Euclidean norm is used then the solutions are called the *least squares solutions*. However, the problem can also incorporate estimates of errors made in measuring data. The errors may fall within some known lower and upper bounds. One can consider the data vector as a pa-

rameter in which case the best approximation problem assumes a more general form. It consists of finding perturbations of data, within specified boundaries, for which one achieves the best fit. The problem is called the *generalized least squares problem* in the context of the Euclidean space. If the analytic form of the function that approximates data is known, and if it contains some 'parameters' to be determined, then the best approximation problem is called the *parameter identification problem*.

Example 3 One may wish to determine the constant of gravity g, the initial position s_0, and the initial velocity v_0 of a falling object. It is known that the object is governed by Newton's second law of motion $s = s_0 + v_0 t + gt^2/2$. Measuring $s = s(t)$ at various times t, one obtains a generally inconsistent system of linear equations in the 'parameters' s_0, v_0, and g. After minimizing a norm of the residual vector one identifies the parameters. However, one may also take into consideration relative errors that have been made in reading t and $s(t)$. The generalized parameter identification problem is to determine the best fit within the allowable errors. This approach typically gives more accurate results. In the context of parametric programming models, the optimal errors are optimal values of the parameter, while the 'parameters' to be identified, like s_0, v_0, and g, are actually decision variables.

Parameter identification problems are used in many areas. For example, in the biological sciences they are used in attempts to find kinetic constants which can quantitatively describe certain *biochemical processes*. Typically, experiments are performed with tracers (labeled with radioactive or stable isotopes), then experimental tissue radioactivity curves are fitted to a biological model to find kinetic parameters. On the basis of these kinetic parameters one can calculate regional glucose utilization in the brain [60], serotonin synthesis [8], aromatic amino acid activity [18] and receptor densities [36].

Optimal parameters are also important for *post-optimality analyses* of linear programs. Using PP one can answer basic questions like: Given an optimal solution of an LP, for what perturbations of data does the solution remain optimal? The following example exposes the problem.

Example 4 Consider the linear program in one variable with zero-value objective function:

$$\begin{cases} \min & 0 \cdot x \\ \text{s.t.} & -1 \le x \le 1. \end{cases}$$

A (global) *minimizer* is $x^* = -1$. Now suppose that this program belongs to the class of perturbed programs

$$\begin{cases} \min & \theta^2 \cdot x, \\ \text{s.t.} & -1 \le x \le 1, \end{cases}$$

when θ is fixed at $\theta^* = 0$. Then, for perturbations $\theta \neq 0$, the point (x^*, θ^*) is actually a local *maximizer*! In situations like these some optimal solutions of linear programs may be 'better' than others. Here, say, $x = 1$ is 'better' than $x^* = -1$, because its local optimality is not affected by perturbations of (x, θ).

The optimality-preserving problems, at the global optimality level, are solved for linear models using the saddle-point optimality conditions along θ-paths and after setting the terms corresponding to the components of the x variable equal to zero. Two closely related problems are:

i) given an infeasible point x (e. g., a prescribed profile of production that one wishes to achieve), find perturbations of data θ that make the point feasible; and
ii) given a feasible but nonoptimal point, find perturbations that make the point optimal.

Many results from convex parametric programming can be adjusted to work for partly convex programs (PC) and more general mathematical programs, e. g., [1,23,70]. Partly convex programs are programs which, after 'freezing' some of the coordinates in x, become convex programs in the remaining variables. The program from Example 4, with the objective function $x_2^2 \cdot x_1$ and the constraint $-1 \le x_1 \le 1$, is a PC program. (Identify $x_2 = \theta$.) Since every mathematical program with twice continuously differentiable functions can be formulated as a *partly convex program*, see [35], one can in principle study (and solve) many mathematical programs by studying PC programs and convex parametric programming models.

Numerous *applications of parametric programming* are mentioned in the classical texts [4,14,15,34]. Parametric programming has found many applications in discrete optimization, transportation problems (e. g.,

[59]), economics and finance (e. g., [11,56]), approximation, as well as in multi-objective, multilevel, stochastic and global optimization (e. g., ► Parametric global optimization: Sensitivity). Some of the recent research in parametric programming has been focused on connections between polynomial complexity and perturbation theory, stability results for nonunique solutions, control, semi-infinite programs (e. g., [28]), and nonsmooth problems.

See also

- ► Bounds and Solution Vector Estimates for Parametric NLPs
- ► Dini and Hadamard Derivatives in Optimization
- ► Global Optimization: Envelope Representation
- ► Multiparametric Linear Programming
- ► Multiparametric Mixed Integer Linear Programming
- ► Nondifferentiable Optimization
- ► Nondifferentiable Optimization: Cutting Plane Methods
- ► Nondifferentiable Optimization: Minimax Problems
- ► Nondifferentiable Optimization: Newton Method
- ► Nondifferentiable Optimization: Relaxation Methods
- ► Nondifferentiable Optimization: Subgradient Optimization Methods
- ► Parametric Global Optimization: Sensitivity
- ► Parametric Linear Programming: Cost Simplex Algorithm
- ► Parametric Mixed Integer Nonlinear Optimization
- ► Parametric Optimization: Embeddings, Path Following and Singularities
- ► Selfdual Parametric Method for Linear Programs

References

1. Antipin A (1966) Equilibrium programming problems: Prox-regularization and prox-methods. In: Gritzmann P (ed) Recent Advances in Optimization, Proc. 8th French–German Conf. Optimization, July 21–26 1996. Springer, Berlin, pp 1–18
2. Asgharian M, Zlobec S (2000) Convex parametric programming in abstract spaces. Techn Report, McGill Univ
3. Auslender A (1984) Stability in mathematical programming with nondifferentiable data. SIAM J Control Optim 22:239–254
4. Bank B, Guddat J, Klatte D, Kummer B, Tammer T (1982) Non-linear parametric optimization. Akademie, Berlin
5. Berkelaar AB, Roos K, Terlaky T (1997) The optimal set partition approach to linear and quadratic programming. In: Advances in Sensitivity Analysis and Parametric Programming. Kluwer., Dordrecht
6. Bonnans JF, Shapiro A (1998) Optimization problems with perturbations, A guided tour. SIAM Rev 40:228–264
7. Cojocaru I (1985) Régions de stabilité dans la programmation linéaire. An Univ Bucuresti Mat 34:12–21
8. Diksic M, Nagahiro S, Grdisa M (1995) The regional rate of serotonin synthesis estimated by the α-methy-tryptophan method in rat brain from a single time point. J Cerebral Blood Flow Metabolism 15:806–813
9. Dinkelbach W (1969) Sensitivitätsanalysen und parametrische Programmierung. de Gruyter, Berlin
10. Dontchev AL, Zollezzi T (1993) Well-posed optimization problems. Lecture Notes Math, vol 1543. Springer, Berlin, pp 239–254
11. Dupacova J Stability and sensitivity for a scenario-based bond portfolio management problem. In: Proc. Conf. Mathematical Methods in Economy and Industry, Liberec, June 1–5 1998
12. Eremin II, Astafiev NN (1976) Introduction to the theory of linear and convex programming. Nauka, Moscow (In Russian)
13. Fiacco AV (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Acad. Press, New York
14. Fiacco AV, McCormick GP (1968) Nonlinear programming: sequential unconstrained minimization techniques. Wiley, New York
15. Gal T (1979) Postoptimal analyses, parametric programming and related topics. McGraw-Hill, New York. Updated revision of German 1973 edn; 2nd revised edition: W. de Gruyter, New York, 1995
16. Gal T (1983) Letter on historiogramme on parametric programming. J Oper Res Soc 34:162–163
17. Gal T (1994) Selected bibliography on degeneracy. Ann Oper Res 46/47:1–10
18. Garnett ES, Firnau G, Nahmias C (1983) Dopamine visualized in the basal ganglia of living man. Nature 305:137–138
19. Gauvin J, Dubeau F (1982) Differential properties of the marginal function in mathematical programming. Math Program Stud 19:101–119
20. Gauvin J, Janin R (1988) Directional behaviour of optimal solutions in nonlinear mathematical programming. Math Oper Res 13:629–649
21. Greenberg HJ, Pierskalla WP (1972) Extensions of the Evans–Gould stability theorem for mathematical programs. Oper Res 20:143–153
22. Guddat J, Jongen HTh (1987) Structural stability in nonlinear optimization. Optim 18:617–631
23. Guddat J, Jongen HTh (1988) On global optimization based on parametric optimization. In: Guddat J et al (eds) Advances in Mathematical Optimization. Akademie, Berlin, 63–79

24. Guddat J, Jongen HTh, Rückmann J-J (1986) On stability and stationary points in nonlinear optimization. J Austral Math Soc (Ser B):36–56
25. Guddat J, Vasquez FGuerra, Jongen HTh (1991) Parametric optimization: singularities, pathfollowing and jumps. Teubner and Wiley, New York
26. Hogan WW (1973) Point-to-set maps in mathematical programming. SIAM Rev 15:591–603
27. Hu T, Huang L (1996) Real stability radii and quadratic matrix inequality. Chinese Sci Bull 41:2043–2046
28. Jongen HTh, Rückmann J-J (1998) On stability and deformation in semi-infinite optimization. In: Reemtsen R, Rückmann J-J (eds) Semi-Infinite Programming. Kluwer, Dordrecht, pp 29–67
29. Jongen HTH, Jonker P, Twilt F (1983) Nonlinear optimization in R^n: Transversality, flows, parametric aspects. P. Lang, Frankfurt am Main
30. Klatte D (1987) Lipschitz continuity of infima and optimal solutions in parametric optimization: The polyhedral case. In: Guddat J et al (eds) Parametric Optimization and Related Topics. Akademie, Berlin, pp 229–248
31. Klatte D, Tammer K (1990) Strong stability of stationary solutions and Karush–Kuhn–Tucker points in nonlinear optimization. Ann Oper Res 27:285–308
32. Kojima M (1980) Strongly stable stationary solutions in nonlinear programs. In: Robinson SM (ed) Analysis and Computation of Fixed Points. Acad. Press, New York, pp 93–138
33. Kojima M, Hirabayashi R (1984) Continuous deformations of nonlinear programs. Math Program Stud 21:150–198
34. Levitin ES (1994) Perturbation theory in mathematical programming and its applications. Wiley, New York, Russian edition: 1992.
35. Liu WB, Floudas CA (1993) A remark on the GOP algorithm for global optimization. J Global Optim 3:519–521
36. Logan J, Dewey SL, Fowler AP, Brodie JS, Angrist B, Volkow ND, Gatley SJ (1991) Effects of endogenous dopamine on measures of [^{18}F] -N- methylspiroperidol binding in basal ganglia: Comparison of simulations and experimental results from PET studies in baboons. Synapse 9:195–207
37. Lommatzsch K (ed) (1979) Anwendungen der linearen parametrischen Optimierung. Akademie, Berlin
38. Malanowski K (1987) Stability and sensibility to optimal control problems for systems with control appearing linearly. Appl Math Optim 16:73–91
39. Malanowski K (1992) Second order conditions and constraint qualifications in stability and sensitivity analysis of solutions to optimization problems in Hilbert spaces. Appl Math Optim 25:51–79
40. Mann G, Zlobec S (1998) Ranking efficiency by robustness in data envelopment analysis. 20th Internat. Symp. Math. Program. with Data Perturbations, Washington, D.C., May 1998
41. Manne AS (1953) Notes on parametric linear programming. RAND Report P-468
42. Martin DH (1975) On the continuity of the maximum in parametric linear programming. J Optim Th Appl 17:205–210
43. Migdalas A, Pardalos PM, Värbrand P (eds) (1998) Multilevel optimization: Algorithms and applications. Nonconvex Optim Appl., vol 20. Kluwer, Dordrecht
44. Mordukhovich BS (1991) Sensitivity analysis in nonsmooth optimization. In: Field D, Komkov V (eds) Theoretical Aspects of Industrial Design. SIAM, Philadelphia
45. Neralić L (1997) Sensitivity in data envelopment analysis for arbitrary perturbations of data. Glasnik Mat 32:315–335
46. Nožička F (1972) Lineare parametrische Optimierung – ein Problem der Stabilität der optimalen Lösung. In: Konf Mat Met v Ekonomii, Praha, pp 45 –82
47. Nožička F (1972) Über eine Klasse von linearen einparametrischen Optimierungsproblemen. Math Operationsforsch Statist 3:159–194
48. Nožička F, Guddat J, Hollatz H (1972) Theorie der linearen Optimierung. Akademie, Berlin
49. Nožička F, Guddat J, Hollatz H, Bank B (1974) Theorie der linearen parametrischen Optimierung. Akademie, Berlin
50. Robinson SM (1974) Perturbed Kuhn–Tucker points and rates of convergence for a class of nonlinear programming algorithms. Math Program 7:1–16
51. Robinson SM (1975) Stability theory for systems of inequalities. Part I: Linear systems. SIAM J Numer Anal 12:754–772
52. Robinson SM (1976) Stability theory for systems of inequalities. Part II: Differentiable nonlinear systems. SIAM J Numer Anal 13:497–513
53. Robinson SM (1980) Strongly regular generalized equations. Math Oper Res 5:43–62
54. Robinson SM (1987) Local epi-continuity and local optimization. Math Program 37:208–222
55. Van Rooyen M, Sears M, Zlobec S (1988) Constraint qualifications in input optimization. J Austral Math Soc (Ser B) 30:326–342
56. Rupnik V (1985) Some experiences with fluid modelling of economic systems. Systems Res 2(3):211–219
57. Shapiro A (1988) Sensitivity analysis of nonlinear programs and differentiability properties of metric projections. SIAM J Control Optim 26:628–645
58. Shapiro A (1992) Perturbation analysis of optimization problems in Banach spaces. Numer Funct Anal Optim 13:97–116
59. Shubert IS, Zimmerman U (1985/6) One-parametric bottleneck transportation problems. Ann Oper Res 4:343–369
60. Sokoloff L, Reivich M, Kennedy C, DesRosiers MH, Patlak CS, Pettigrew KD, Sakurada O, Shinohara M (1977) The 14 C-deoxyglucose method for the measurement of local glucose utilization: theory, procedure, and normal values in the conscious and anesthetized albino rat. J Neurochemistry 28:897–916
61. Tharwat A, Zimmermann K (1998) On optimal choice of parameters in machine-time scheduling problems. In: Proc.

Conf. Mathematical Methods in Economy and Industry, Liberec, June 1–5 1998
62. Tikhonov AV, Arsenin VY (1977) Solutions of ill-posed problems. Wiley, New York
63. Ward J, Wendell RE (1990) Approaches to sensitivity analysis in linear programming. Ann Oper Res 27:3–38
64. Wendell RE (1997) Linear programming 3: The tolerance approach. In: Gal T, Greenberg HJ (eds) Advances in Sensitivity Analysis and Parametric Programming. Kluwer, Dordrecht
65. Zangwill WJ (1969) Nonlinear programming: A unified approach. Prentice-Hall, Englewood Cliffs
66. Zidaroiu C (1985) Regions of stability for random decision systems with complete connections. An Univ Bucuresti Mat 34:87–97
67. Zimmermann K (1989) Optimal choice of parameters in a machine scheduling problem. Ekonomicko-Matematicky Obzor 25:73–83
68. Zlobec S (1983 1986) Characterizing an optimal input in perturbed convex programming. Math Program 25:109–121, Corrigendum: Ibid 35:368–371
69. Zlobec S (1988) Characterizing optimality in mathematical programming models. Acta Applic Math 12:342–349
70. Zlobec S (1996) Lagrange duality in partly convex programming. In: Floudas CA, Pardalos PM (eds) State of the Art in Global Optimization. Kluwer, Dordrecht, pp 1–17
71. Zlobec S (1998) Stable parametric programming. Optim 45:387–416
72. Zlobec S (2000) Parametric programming: An illustrative mini-encyclopaedia. Math Comunications 5:1–39
73. Zlobec S, Ben-Israel A (1979) Perturbed convex programs: Continuity of the optimal solutions and optimal values. In: Oettli W, Steffens F (eds) Methods Oper. Res.: Proc. III Symp. Operat. Res., 31. Athenäum–Hain–Scriptor–Hanstein, pp 737–749
74. Zlobec S, Gardner R, Ben-Israel A (1982) Regions of stability for arbitrarily perturbed convex programs. In: Fiacco AV (ed) Mathematical Programming With Data Perturbations, Lecture Notes Pure and Applied Math. M. Dekker, New York, pp 69–89
75. Zowe J, Kurcyusz S (1979) Regularity and stability for the mathematical programming problem in Banach spaces. Applied Math and Optim 5:49–62

Nondifferentiable Optimization: Relaxation Methods

Ulf Bränndlund
Kungliga Tekniska Högskolan, Stockholm, Sweden

MSC2000: 49J52, 90C30

Article Outline

Keywords
Constraints on the Variables
Finding a Minimum Point
See also
References

Keywords

Subgradient; Subdifferential; Convex function; Relaxation method; Lagrangian relaxation; Bundle methods

Relaxation methods for convex nondifferentiable optimization have their origin in relaxation methods for finding a solution to a system of linear inequalities, see [1] and [6]. In mathematical terms, one wants to find a vector $x \in \mathbf{R}^n$ such that $a_i^\top x - b_i \leq 0$ for $i = 1, \ldots, m$. In the simplest form, such a relaxation method iterates an initial guess $x_0 \in \mathbf{R}^n$ with the iteration scheme

$$x_{k+1} = x_k - \gamma^k \frac{a_{i_k}^\top x_k - b_{i_k}}{\|a_{i_k}\|_2^2} a_{i_k}, \tag{1}$$

where $i_k \in \operatorname{Argmax}_{i=1,\ldots,m} a_i^\top x_k - b_i$, and $\gamma^k \in [\delta, 2-\delta]$ and $\delta \in (0, 1)$. The iterations stop once a feasible solution is found. With $\gamma^k = 1$ the iteration formula (1) corresponds to a projection of x_k onto the most violated hyperplane at x_k.

The problem of finding a solution to an inequality system can be cast into a convex nondifferentiable optimization problem with known optimal value, namely the following

$$\min_x f(x) = \min_x \max \left\{ \max_{i=1,\ldots,m} a_i^\top x - b_i, 0 \right\},$$

which has optimal solution value equal to zero if the system is consistent.

The iteration scheme (1) can be generalized to convex nondifferentiable minimization with known optimal value in the following manner. Suppose that we want to find $x \in \mathbf{R}^n$ such that $f(x) \leq f_{\text{lev}}$, where $f_{\text{lev}} \geq f^* = \inf f(x)$ is known. Let $\mathfrak{m}_f$ denote the set of affine functions minorizing f,

$$\mathfrak{m}_f = \left\{ (a, b) \in \mathbb{R}^n \times \mathbb{R} : \begin{array}{c} a^\top y - b \leq f(y) \\ \text{for all } y \in \mathbb{R}^n \end{array} \right\}.$$

Assume as is customary in nondifferentiable optimization that we at $x \in \mathbf{R}^n$ can evaluate $f(x)$ and one arbitrarily chosen *subgradient*

$$g \in \partial f(x) = \left\{ g \in \mathbb{R}^n : \begin{array}{c} f(y) \geq f(x) + g^\top (y - x) \\ \text{for all } y \in \mathbb{R}^n \end{array} \right\} .$$

Then, $(g, g^\intercal x - f(x)) \in \mathfrak{m}_f$ and by definition of $\mathfrak{m}_f$,

$$f(x) \geq \sup_{(a,b)\in\mathfrak{m}_f} \{a^\top x - b\} \geq g^\top x - (g^\top x - f(x)) = f(x) .$$

Hence $f(x) = \sup_{(a,b)\in\mathfrak{m}_f}\{a^\top x - b\}$.

Thus, finding an x such that $f(x) \leq f_{\text{lev}}$ can be seen as solving an infinite system of inequalities, $a^\intercal x - b - f_{\text{lev}} \leq 0$ for all $(a, b) \in \mathfrak{m}_f$. The analog of (1) then becomes

$$x_{k+1} = x_k - \gamma^k \frac{f(x_k) - f_{\text{lev}}}{\|g_k\|_2^2} g_k , \tag{2}$$

where $g_k \in \partial f(x_k)$. It can be shown that if $X^* = \{x: f(x) \leq f_{\text{lev}} \neq \emptyset$ then $x_k \to x_{\text{lev}} \in X^*$.

This steplength rule has often in the literature been called the *Polyak II rule* perhaps since it appears as the second numbered equation in B.T. Polyak's classical paper [8], where its convergence properties are studied. It is shown there that, under certain mild conditions on f, the convergence rate is linear to an optimal solution, should it exist.

The step (2) corresponds to a projection onto a hyperplane. With $\gamma^k = 1$, x_{k+1} solves the quadratic programming problem

$$\begin{cases} \min & \|x - x_k\|_2^2 , \\ \text{s.t.} & f_{\text{lev}} \geq f(x_k) + g_k^T (x - x_k) . \end{cases} \tag{3}$$

One may in this framework add previously generated subgradients in order to obtain faster convergence, i. e. let x_{k+1} solve

$$\begin{cases} \min & \|x - x_k\|_2^2 , \\ \text{s.t} & f_{\text{lev}} \geq f(x_j) + g_j^T (x - x_j), \\ & j \in J^k , \end{cases} \tag{4}$$

where J^k is a subset of $\{1, \ldots, k\}$ including k.

Constraints on the Variables

The classical way to handle constraints on the variables, i. e. when x has to be in some closed and *convex set* X as well as in the set $\{x: f(x) \leq f_{\text{lev}}$, is to let

$$x_{k+1} = P_X \left(x_k - \gamma^k \frac{f(x_k) - f_{\text{lev}}}{\|g_k\|_2^2} g_k \right) , \tag{5}$$

where $P_X(y) = \text{argmin}_{x \in X} \|y - x\|_2^2$ denotes the *projection* of y onto the feasible set X.

However, it is often better to let x_{k+1} be the projection of x_k onto the set $X \cap \{x: \gamma^k (f_{\text{lev}} - f(x_k)) \geq g_k^T (x - x_k)\}$, i. e. for $\gamma^k = 1$ to add $x \in X$ to the constraint set of (3). If X consists of simple bounds or the unit-simplex then it can be shown that this latter way results in a new iteration point which is closer to the desired set $\{x \in X: f(x) \leq f_{\text{lev}}$ than does (5) (see [2]). In these two cases the computational burden of solving this one projection problem does not need to be larger, than solving the two very simple projections involved in (5).

Finding a Minimum Point

Suppose one wants to find a minimum point x^* of a convex function f. Then if the optimal value f^* is known a priori then the algorithms mentioned above can be successfully used with $f_{\text{lev}} = f^*$. But, prior knowledge of f^* is not usually the case. However, when solving the dual problem in applications of *Lagrangian relaxation* to obtain upper bounds on a primal maximization problem, it is often the case that a (good) lower bound, f_{low}, on f^* is known from a primal feasible solution. In these applications, iterations schemes as the ones above can be applied heuristically and often successfully by replacing f_{lev} in (2) by f_{low}.

To be specific, one such heuristic goes as follows. Suppose that at iteration k a lower bound f_{low}^k on f^* is known, which is used in place of f_{lev} in (2). Initially, $\gamma_0 = 2$ and at iteration k, γ_k is reduced by a constant factor $\alpha \in (0, 1)$, if there has been no decrease in terms of the function values for the last $K_{\max}$ iterations (which could indicate that too large steps are taken). Here α and $K_{\max}$ are parameters which need to be chosen appropriately for the application in question. Typical values are $\alpha = 0.5$ and $K_{\max} = 5$.

0 (Initialization)
Select a point $x_1 \in \mathbf{R}^n$ and $\delta^1 > 0$ and $\sigma_{\max} > 0$.
Set $\sigma_1 = 0$ and $f^0_{\text{rec}} = \infty$. Set $k = 1$, $l = 1$ and $k(1) = 1$.

1 (Function call)
Calculate $f(x_k)$ and $g_k \in \partial f(x_k)$.
IF $\| g_k \| = 0$
THEN terminate ($x_k \in$ Argmin f). Set $f^k_{\text{rec}} = \min\{f(x_k), f^{k-1}_{\text{rec}}\}$ and update x^k_{rec} correspondingly.

2 (Sufficient Descendent?)
If $f(x_k) \le f^{k(l)}_{\text{rec}} - \delta_l/2$, set $k(l+1) = k$, $\sigma_k = 0$, $\delta_{l+1} = \delta_l$, $l = l + 1$ and skip next step.

3 (Oscillation?)
If $\sigma_k > \sigma_{\max}$, set $k(l+1) = k$, $\sigma_k = 0$, $\delta_{l+1} = \delta_l/2$, replace x_k by x^k_{rec} and g_k correspondingly.

4 (New point)
Set $f^k_{\text{lev}} = f^{k(l)}_{\text{rec}} - \delta_l$,

$$x_{k+1} = x_k - \frac{f(x_k) - f^k_{\text{lev}}}{\| g_k \|_2^2} g_k.$$

5 (Path update)
Set $\sigma_k = \sigma_k + \| x_{k+1} - x_k \|$. Set $k = k + 1$ and return to Step 1.

A simple convergent method, [2] and [4], based on (2) for minimizing f with no assumption on prior knowledge of the optimal value is motivated as follows. It is a fact that if the set $\{x \in \mathbf{R}: f(x) \le f_{\text{lev}}\}$ is empty then the iteration scheme (2) generates a path whose length $\sum_{k=1}^{\infty} \|x_{k+1} - x_k\|$ is unbounded, and if the set is nonempty then the path length is bounded. If the former seems to be occurring then f_{lev} should be increased and if the algorithm seems to be converging to f_{lev} then a lower f_{lev} could be used. The mechanics of the algorithm should be clear from the following description. Let $f^k_{\text{rec}} = \min_{l=1,\ldots,k} f(x_l)$ denote the best function value found up to iteration k and let x^k_{rec} be the point at which this occurs. Let f^k_{lev} denote the 'level' which we are aiming for. The number $k(l)$ denotes the iteration of the lth change of f^k_{lev}. The number σ_k is the length of the path since the last update of f_{lev}.

Note that the level f^k_{lev} remains fixed in between the iterations $k(l)$ and $k(l+1) - 1$. For this algorithm it is possible to derive so called efficiency estimates. In particular, it can be shown that for 'small' $\epsilon > 0$ the algorithm produces $f^k_{\text{rec}} - f^* < \epsilon$ for $k \ge K/\epsilon^3$, where K is a positive constant.

Of course, one may in Step 4 use (4) instead of (2) to obtain a new iteration point. However, when using (4) in Step 4, more sophisticated schemes to adjust the aiming level f^k_{lev} are possible. A scheme suggested in [3] uses ideas from so called *proximal point bundle methods*, in which sufficient descent in terms of function values is enforced by means of so called null steps. Another method by C. Lemaréchal, A.S. Nemirovsky and Yu.E. Nesterov, [5], for the case when x is constrained to a compact set X, uses $f^k_{\text{lev}} = \alpha f^k_{\text{rec}} + (1-\alpha) f^k_{\text{low}}$, where f^k_{low} is the best lower bound possible, that is

$$f^k_{\text{low}} = \min_{x \in X} \max_{j=1,\ldots,k} f(x_j) + g_j^T (x - x_j).$$

This algorithm has a complexity estimate given by $f^k_{\text{rec}} - f^* < \epsilon$ for $k \ge K/\epsilon^2$, where K is a positive constant. Such a complexity estimate is optimal in the sense that it can not be improved uniformly with respect to the dimension by more than an absolute constant factor (for details, see [7]).

See also

- ▶ Dini and Hadamard Derivatives in Optimization
- ▶ Global Optimization: Envelope Representation
- ▶ Nondifferentiable Optimization
- ▶ Nondifferentiable Optimization: Cutting Plane Methods
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Nondifferentiable Optimization: Newton Method
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Nondifferentiable Optimization: Subgradient Optimization Methods

References

1. Agmon S (1954) The relaxation method for linear inequalities. Canad J Math 6:282–292
2. Brännlund U (1993) On relaxation methods for nonsmooth convex optimization. PhD Thesis, Kungliga Tekniska Högskolan, Stockholm
3. Brännlund U, Kiwiel KC, Lindberg PO (1995) A descent proximal level bundle method for convex nondifferentiable optimization. Oper Res Lett 17:121–126
4. Goffin J-L, Kiwiel KC (1999) Convergence of a simple subgradient level method. Math Program 85(1):207–211

5. Lemaréchal C, Nemirovskii AS, Nesterov YuE (1995) New variants of bundle methods. Math Program 69:111–147
6. Motzkin TS, Schoenberg IJ (1954) The relaxation method for linear inequalities. Canad J Math 6:393–404
7. Nemirovsky AS, Yudin DB (1983) Problem complexity and method efficiency in optimization. Ser Discret Math, Wiley, New York
8. Polyak BT (1969) Minimization of unsmooth functionals. USSR Comput Math Math Phys 9:14–29

Nondifferentiable Optimization: Subgradient Optimization Methods

Subgradient Methods

Ulf Brännlund
Kungliga Tekniska Högskolan, Stockholm, Sweden

MSC2000: 49J52, 90C30

Article Outline

Keywords
See also
References

Keywords

Subgradient; Subdifferential; Convex function

Subgradient methods for minimization of a *convex function* f: $\mathbf{R}^n \to \mathbf{R}$ over a closed *convex set* X have proven to be an efficient mean to solve large scale optimization problems. In particular, this is the case when X is a simple set, such as $\mathbf{R}^n$ or the positive orthant and when high accuracy of the solution is not required, e. g. in the context of Lagrangian relaxation of integer programming problems (cf. ► Integer programming: Lagrangian relaxation).

A *subgradient* at a point $x \in \mathbf{R}^n$ of a convex function f is a vector $g \in \mathbf{R}^n$ satisfying $f(y) \geq f(x) + g^\mathsf{T}(y - x)$ for all $y \in \mathbf{R}^n$. The set of subgradients at a point x is known as the *subdifferential* $\partial f(x)$. At points where the function is differentiable, the gradient is the sole member of the subdifferential.

Gradient based iterative methods, e. g. the steepest descent method designed for minimization of a smooth functions fail when one attempts to use their analogs, i. e. replacing gradients by subgradients, for minimization of convex nonsmooth functions. This is because the negative subgradient may not be a descent direction and even if it were at all points generated along the path, the sequence of iterates would not necessarily minimize f.

The basic subgradient algorithm takes the following form:

0 (Initialize)
Choose a point $x_0 \in \mathbf{R}^n$ and set $k = 0$.
1 (Function call)
Calculate $g_k \in \partial f(x_k)$.
IF $\| g_k \| = 0$ THEN terminate since $x_k \in \operatorname{Argmin} f$.
2 (New point)
Set
$$x_{k+1/2} = x_k - t_k \frac{g_k}{\|g_k\|},$$
and
$$x_{k+1} = \operatorname*{argmin}_{y \in X} \| y - x_{k+1/2} \|,$$
replace k by $k + 1$. Return to Step 1.

It can be shown, see [4], that if

$$t_k \downarrow 0 \quad \text{and} \quad \sum_{k=0}^{\infty} t_k = \infty, \tag{1}$$

then $\liminf f(x_k) = \inf_{x \in X} f(x) = f^*$ and furthermore, if also

$$\sum_{k=0}^{\infty} t_k^2 < \infty,$$

then x^k converges to a minimum point, if there is one (see [2]).

As can be expected, the rate of convergence for the above algorithm with the *divergent series rule* (1) is very slow. In fact, it can be shown that the algorithm can not have so-called geometric convergence rate. An algorithm is said to have *geometric convergence rate*, or *r-linear convergence rate*, if for any convex function and any starting point there exist M and $q \in (0, 1)$ such that $\|x_k - x^*\| \leq Mq^k$, where x^* is an optimal point. J.-L. Goffin has [3] shown that it is possible to obtain geometric convergence rate with the *geometric series rule*

$$t_k = M\rho^k \|g_k\|,$$

if M and $\rho \in (0, 1)$ are chosen large enough. In practice, though, for a particular f and starting point x_0 it is impossible to know what sufficiently large is.

Another often used steplength rule is the so-called *relaxation rule*, or *Polyak II rule*, where t^k is chosen according to

$$t_k = \gamma_k \frac{f(x_k) - f_{\text{lev}}^k}{\|g_k\|},$$

where f_{lev}^k is an estimate of the minimum value value f^* and $\gamma_k \in [\delta, 2 - \delta]$, and $\delta \in (0, 1)$. See also ► Nondifferentiable optimization: Relaxation methods.

Pure subgradient methods have a tendency to zigzag, i. e. a step in the direction $-g_k$ tends to be followed by a step which is almost parallel with g_k. Several solutions to overcome this behavior have been proposed. One such solution is the concept of *space dilation* of N.Z. Shor (see [5]). This approach is related to quasi-Newton methods for differentiable optimization. Another solution to the zig-zagging problem is the method proposed in [1], in which a step is taken in a direction which is a sum of the previous direction and the current subgradient. This approach is analogous to conjugate gradient methods for differentiable optimization.

See also

- ► Dini and Hadamard Derivatives in Optimization
- ► Global Optimization: Envelope Representation
- ► Nondifferentiable Optimization
- ► Nondifferentiable Optimization: Cutting Plane Methods
- ► Nondifferentiable Optimization: Minimax Problems
- ► Nondifferentiable Optimization: Newton Method
- ► Nondifferentiable Optimization: Parametric Programming
- ► Nondifferentiable Optimization: Relaxation Methods

References

1. Camerini PM, Fratta L, Maffioli F (1975) On improving relaxation methods by modified gradient techniques. Math Program Stud 3:79–97
2. Correa R, Lemaréchal C (1993) Convergence of some algorithms for convex minimization. Math Program 62(2):261–275
3. Goffin JL (1977) On convergence rates of subgradient optimization methods. Math Program 13:329–347
4. Polyak BT (1967) A general method for solving extremum problems. Soviet Math Dokl 8:593–597
5. Shor NZ (1970) Convergence rate of gradient descent method with dilation of the space. Cybernetics 6(2):102–108

Nonlinear Least Squares: Newton-type Methods

Chengxian Xu
Xian Jiaotong University, Xian, China

MSC2000: 49M37

Article Outline

Keywords
See also
References

Keywords

Newton method; Full-step Gauss–Newton method; Quasi-Newton methods; Adaptive methods; Global convergence; Local convergence rate

Nonlinear least squares problems can be phrased in terms of minimizing a real valued function that is a sum of some nonlinear functions of several variables. Efficient solution for unconstrained nonlinear least squares is important. Though some problems that arise in practical areas usually have constraints placed upon the variables and special techniques are required to handle these constraints, eventually the numerical techniques used rely upon the efficient solution of unconstrained nonlinear least squares problems.

The *unconstrained nonlinear least squares problems* have the form

$$\min f(x) = \frac{1}{2}\sum_{i=1}^{m}[r_i(x)]^2 = \frac{1}{2}r(x)^{\top}r(x),$$

where $r(x) = (r_1(x), \ldots, r_m(x))^{\top}$ and $r_i(x)$, $i = 1, \ldots, m$ are nonlinear functions of $x \in \mathbf{R}^n$. When $r(x)$, hence $f(x)$ is twice continuously differentiable, the gradient

and the Hessian matrix of $f(x)$ are given by

$$g(x) = \nabla f(x) = A(x)r(x),$$
$$G(x) = \nabla^2 f(x) = A(x)A(x)^\top + \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x)$$

where $A(x) = [\nabla r_1(x) \cdots \nabla r_m(x)]$. The special structures of these derivatives had been exploited in developing effective solution methods for nonlinear least squares.

As a general unconstrained minimization problem, the *Newton method* plays a central role in the development of numerical methods for nonlinear least squares solution. Most commonly used nonlinear least squares methods can be viewed as variations on Newton's method. The Newton method for general optimization is derived based upon the quadratic model

$$q_k(\delta) = f_k + g_k^\top \delta + \frac{1}{2}\delta^\top G_k \delta,$$

where $f_k = f(x^{(k)})$, $g_k = g(x^{(k)})$, $G_k = G(x^{(k)})$, $\delta = x - x^{(k)}$, and $x^{(k)}$ is an approximation to a local minimizer x^* of the objective function $f(x)$. $q_k(\delta)$ is a local approximation to $f(x)$ at $x^{(k)}$ obtained from the *truncated Taylor approximation*. If this approximation is appropriate, then a presumably better approximation $x^{(k+1)} = x^{(k)} + \delta^{(k)}$ can be obtained by requiring that the step $\delta^{(k)}$ be a minimizer of $q_k(\delta)$. Thus the Newton method takes an initial approximation $x^{(1)}$ to x^* and attempts successively to improve the approximation through the iteration

- Solve the system $G_k\delta = -g_k$ for $\delta = \delta^{(k)}$.
- Set $x^{(k+1)} = x^{(k)} + \delta^{(k)}$.

If $G_k = M_k + C_k$ with $M_k = A_k A_k^\top$, $C_k = C(x^{(k)}) = \sum_{i=1}^{m} r_i(x^{(k)})\nabla^2 r_i(x^{(k)})$ is positive definite, the solution $\delta^{(k)}$ of the system is the global minimizer of $q_k(\delta)$, and if the starting point $x^{(1)}$ is sufficiently close to x^* at which $g(x^*) = 0$ and $G(x^*)$ is positive definite, the Newton method is well defined and converges at a quadratic rate.

Unfortunately, the basic Newton method as it stands is not suitable for a general purpose use since G_k may not be positive definite when $x^{(k)}$ is remote from x^* and even if G_k is positive definite, the convergence may not occur since $\{f_k\}$ may not decrease. Though both the possibilities can be eliminated by incorporating either *trust region technique* for the former case or *line search technique* for the later case, the main disadvantage of the Newton method is the demand for evaluation of second order derivatives of problem functions.

Since $r(x)$ is being minimized in the least squares sense, it may be the case that $r_i(x^*)$, $i = 1, \ldots, m$ are zero or very small. Thus when $x^{(k)}$ is close to x^*, compared with M_k, the second part C_k in G_k may be negligible. This suggests that M_k is a good approximation to G_k and gives the well known *full-step Gauss–Newton method*

- Solve the system $M_k\delta = -A_k r_k$ for $\delta = \delta^{(k)}$.
- Set $x^{(k+1)} = x^{(k)} + \delta^{(k)}$.

An important feature of the Gauss–Newton method is that the approximation to the Hessian G_k is directly obtained only from the first order derivatives of the problem functions. The approximation will be exact when the functions $r_i(x)$, $i = 1, \ldots, m$, are all either linear or zeros. Since the Gauss–Newton method is obtained from the Newton method by neglecting the part C_k of G_k, the convergence property of the method greatly depends on the size of the omitted part. If $C(x^*)$ is large relative to $M(x^*)$ in the sense $\eta_m(x^*) = \|M(x^*)^{-1} C(x^*)\| > 1$ which is regarded as *combined relative measure* of the nonlinearity and residual size of the problem (assuming $A(x^*)$ is full rank), then the method does not converge. If $\eta_m(x^*) \leq \theta < 1$ and the initial point $x^{(1)}$ is close to x^* enough, the Gauss–Newton method converges. In case convergence occurs, the rate of convergence also depends upon the size of $\eta_m(x^*)$. If $C(x^*) = 0$, the method is rather satisfactory in the sense that the method converges quadratically for zero residual problems, and if $C(x^*) \neq 0$, the convergence rate of the method is linear and the speed of convergence decreases as the relative nonlinearity or the residual size increases.

Since the matrix M_k is at worst positive semidefinite, the solution, denoted now by $s^{(k)}$, determined in the Gauss–Newton system is not uphill on $f(x)$. A possible modification to force the Gauss–Newton method to converge is to incorporate line search techniques. If the matrix A_k has full rank, then M_k is positive definite and the solution $s^{(k)}$ in the Gauss–Newton system is a descent direction of $f(x)$ at $x^{(k)}$. A line search along the direction $s^{(k)}$ determines a steplength α_k such that $f(x^{(k)}$

$+\alpha_k s^{(k)}) < f(x^{(k)})$ and a new approximation to x^* is then obtained as $x^{(k+1)} = x^{(k)} + \alpha_k s^{(k)}$. This modification was first suggested by H.O. Hartley [7] and is generally referred as *damped Gauss–Newton method*. This method does prevent divergence and usually have a larger domain of convergence than the full-step Gauss–Newton method. In fact convergence follows if the *condition number* $\mathcal{K}(A_k A_k^\top)$ is uniformly bounded above. Since A_k will usually be bounded above, this essentially requires that A_k does not loss rank in limit. Unfortunately this can happen [11] and convergence to a noncritical point can occur. Also when A_k is rank deficient, the solution $s^{(k)}$ of the Gauss–Newton system becomes numerically orthogonal to g_k at some distance from a local minimizer x^* and no further progress can be made by line searches.

A further modification to the full-step Gauss–Newton method is to incorporate the trust region technique. In this modification, the trust region subproblem

$$\begin{cases} \min & q_k(\delta) = f_k + r_k^\top A_k^\top \delta + \frac{1}{2}\delta^\top M_k \delta \\ \text{s.t.} & \|\delta\| \le \Delta_k \end{cases}$$

is solved for $\delta^{(k)}$ with properly adjusted radius Δ_k and the new point is obtained as $x^{(k+1)} = x^{(k)} + \delta^{(k)}$. The trust region radius is adjusted in such a way that the model function $q_k(\delta)$ is believed to have adequately approximated the function $f(x)$ in the region $\| \delta \| \le \Delta_k$. This modification with a different form was first suggested by K. Levenberg [8] and D.W. Marquardt [9]. J.J. Moré used the modification in the above form [10].

An aspect of the Gauss–Newton method is the efficient solution of the Gauss–Newton system. It must be emphasized here that the solution of the system can not be done by first forming the product $A_k A_k^\top$ and then performing a factorization on the product, because this will worse the conditioning of the system and lead to substantial loss in precision. A stable and efficient way is to factorize the augmented matrix $[A_k^\top \; r_k]$ using either *Householder transformation* or *Given's transformation*.

Regarded as a general optimization problem, another way to obtain approximations to the Hessian matrix G_k from first order derivative information of problem functions is to use *quasi-Newton updates* and the resulting Newton type methods are called *quasi-Newton methods*. Let B_k denote an approximation to the Hessian G_k in the quadratic model $q_k(\delta)$. Any Newton type method with line searches has the following basic framework.

- Solve the system $B_k s = -g_k$ for a descent direction $s^{(k)}$.
- Determine a steplength α_k along the direction $s^{(k)}$ by line searches.
- Set $x^{(k+1)} = x^{(k)} + \alpha_k s^{(k)}$.

In order to ensure *global convergence* and to have a rapid *local convergence rate*, quasi-Newton updates require the matrix B_k to satisfy the following conditions:

- since $G_k \delta_k \approx \gamma_k$, B_k should satisfy the so-called quasi-Newton equation

  $$B_{k+1}\delta^{(k)} = \gamma^{(k)},$$

  where

  $$\delta^{(k)} = x^{(k+1)} - x^{(k)}, \gamma^{(k)} = g_{k+1} - g_k.$$

- B_k is symmetric.
- B_{k+1} is effectively obtained from B_k using lower rank updating

  $$B_{k+1} = B_k + E_k$$

  so that the calculation of B_{k+1} is less expensive.
- B_k is positive definite so that the solution $s^{(k)}$ obtained as the minimizer of $q_k(\delta)$ is a descent direction of $f(x)$ at $x^{(k)}$.

There exist numerous updating formulas to achieve these conditions and the *Broyden family* with single parameter

$$B_{k+1}(\theta) = B_k - \frac{B_k \delta^{(k)} {\delta^{(k)}}^\top B_k}{{\delta^{(k)}}^\top B_k \delta^{(k)}} + \frac{\gamma^{(k)} {\gamma^{(k)}}^\top}{{\delta^{(k)}}^\top \gamma^{(k)}} + \theta w^{(k)} {w^{(k)}}^\top$$

is the most important, where

$$w^{(k)} = ({\delta^{(k)}}^\top B_k \delta^{(k)})^{1/2} \times \left[\frac{\gamma^{(k)}}{{\delta^{(k)}}^\top \gamma^{(k)}} - \frac{B_k \delta^{(k)}}{{\delta^{(k)}}^\top B_k \delta^{(k)}} \right]$$

and θ is a parameter [3]. If ${\delta^{(k)}}^\top \gamma^{(k)} > 0$ for all k, then $B_{k+1}(\theta)$ preserves positive definiteness for all values

$$\theta > \overline{\theta} = \frac{1}{1 - a_k b_k} (< 0),$$

where

$$a_k = \frac{{\gamma^{(k)}}^\top H_k \gamma^{(k)}}{{\delta^{(k)}}^\top \gamma^{(k)}}, \quad b_k = \frac{{\delta^{(k)}}^\top B_k \delta^{(k)}}{{\delta^{(k)}}^\top \gamma^{(k)}}$$

and $H_k = B_k^{-1}$. The $\overline{\theta}$ is the value which causes $B_{k+1}(\theta)$ to be singular. The condition ${\delta^{(k)}}^\top \gamma^{(k)} > 0$ is realistic and can always be achieved when the steplength α_k either is determined from exact line search or satisfies the *Goldstein conditions*. The Broyden family includes the famous BFGS ($\theta = 0$) formula

$$B_{k+1}^{\mathrm{BFGS}} = B_k + \frac{\gamma^{(k)} {\gamma^{(k)}}^\top}{{\delta^{(k)}}^\top \gamma^{(k)}} - \frac{B_k \delta^{(k)} {\delta^{(k)}}^\top B_k}{{\delta^{(k)}}^\top B_k \delta^{(k)}},$$

DFP ($\theta = 1$) formula

$$B_{k+1}^{\mathrm{DFP}} = B_k + \left(1 + \frac{{\delta^{(k)}}^\top B_k \delta^{(k)}}{{\delta^{(k)}}^\top \gamma^{(k)}}\right) \times \frac{\gamma^{(k)} {\gamma^{(k)}}^\top}{{\delta^{(k)}}^\top \gamma^{(k)}} - \frac{\gamma^{(k)} {\delta^{(k)}}^\top B_k + B_k \delta^{(k)} {\gamma^{(k)}}^\top}{{\delta^{(k)}}^\top \gamma^{(k)}}$$

and the *selfdual rank one formula* ($\theta = 1/(1-b_k)$)

$$B_{k+1} = B_k + \frac{y_k y_k^\top}{{\delta^{(k)}}^\top y_k}, \quad y_k = \gamma^{(k)} - B_k \delta^{(k)}.$$

Both the DFP and BFGS methods was found to work well in practice and have been widely used. These two methods have a number of important properties as follows:

1) For quadratic functions (with exact line searches)
   - terminate in at most n iterations with ($B_{n+1} = G$);
   - preserve the *hereditary property*

$$B_i \delta^{(j)} = \gamma^{(j)}, \quad j = 1, \ldots, i-1;$$

   - generate conjugate directions, and conjugate gradients (when $B_1 = I$)
2) For general functions
   - preserve positive definite B_k;
   - global convergence for strictly convex functions (with exact line searches);
   - local superlinear convergence rate.

The BFGS method is even better than the DFP method and has usually been used with low accuracy line searches. In fact, global convergence and superlinear convergence rate of the BFGS method with inexact line searches have been proved [12]. The Broyden family is important in that many of the properties of the BFGS and DFP formulas are common to whole family. L.C.W. Dixon showed that when applied to any continuously differentiable function, all Broyden methods generate the same sequence $\{x^{(k)}\}$ from the same starting point, assuming that the multiple local minima in line search are resolved consistently, degenerate values of θ are avoided and the algorithm is well-defined.

When any above quasi-Newton updating formula is used in a method, the system $B_k s = -g_k$ needs solved to get the direction $s^{(k)}$ and this needs $O(n^3)$ arithmetic operations. In early versions of quasi-Newton methods, the search direction $s^{(k)}$ is obtained as a product of a matrix and a vector from

$$s^{(k)} = -H_k g_k$$

and the matrix H_k is an inverse approximation to G_k ($H_k \approx G_k^{-1}$ obtained from an *inverse quasi-Newton updating* formula. This avoids the solution of the system and only requires $O(n^2)$ arithmetic operations. The inverse updating formulas can be similarly derived from the quasi-Newton equation

$$H_{k+1} \gamma^{(k)} = \delta^{(k)}.$$

or can be derived from above updating formulas using the *Sherman–Morrison formula*. For example, the Broyden family of inverse updatings is given by

$$H_{k+1}(\phi) = H_k - \frac{H_k \gamma^{(k)} {\gamma^{(k)}}^\top H_k}{{\gamma^{(k)}}^\top H_k \gamma^{(k)}} + \frac{\delta^{(k)} {\delta^{(k)}}^\top}{{\delta^{(k)}}^\top \gamma^{(k)}} + \phi v^{(k)} {v^{(k)}}^\top,$$

where

$$v^{(k)} = \left(\gamma^{(k)\top} H_k \gamma^{(k)}\right)^{1/2} \times \left[\frac{\delta^{(k)}}{\delta^{(k)\top}\gamma^{(k)}} - \frac{H_k\gamma^{(k)}}{\gamma^{(k)\top} H_k \gamma^{(k)}}\right]$$

and ϕ is a parameter related to θ by

$$\phi = \frac{1-\theta}{1+\theta(a_k b_k - 1)}.$$

The corresponding inverse BFGS, DFP and rank one updating formulas can be obtained by setting $\phi = 1$, $\phi = 0$ and $\phi = 1/(1 - a_k)$. It can be seen that one kind of updating formulas is obtained from another kind by simply making interchanges $B_k \leftrightarrow H_k$ and $\delta^{(k)} \leftrightarrow \gamma^{(k)}$. Although, the matrix H_k preserves the positive definiteness in theory when $\delta^{(k)\top}\gamma^{(k)} > 0$, the loss of the positive definiteness may occur in practical calculations due to round-off errors and it can not be easily recognized whether the matrix H_k is positive definite or not. Now the widely used versions of quasi-Newton methods is to represent the matrix B_k in a factorized form $L_k D_k L_k^\top$ and to update the factors in $O(n^2)$ operations to obtain the factors $L_{k+1}D_{k+1}L_{k+1}^\top$ of the matrix B_{k+1} where L is a unit lower triangular matrix and D is a diagonal matrix with positive diagonals [5]. The solution of the system $B_k s = -g_k$ is then obtained in $O(n^2)$ operations by using forward and backward substitutions. The positive definiteness of the matrix B_k is indicated by the diagonal elements of D_k and can be maintained by controlling round-off errors. Methods with quasi-Newton updating in $LDL^\intercal$ form preserve the convergence properties of quasi-Newton methods and reduce the arithmetic operations at each iteration.

When quasi-Newton updating formulas are used to generate approximations to the Hessian matrices or their inverses for nonlinear least squares problems, $A_1A_1^\top$ can be selected as the initial matrix B_1 if $A_1A_1^\top$ is positive definite. One more modification to the quasi-Newton updatings for nonlinear least squares is the definition of the vector $\gamma^{(k)}$. Let $\gamma_o^{(k)}$ denote previous defined vector $\gamma^{(k)}$. From the Taylor expansion of $g(x)$, a new definition of $\gamma^{(k)}$

$$\gamma_n^{(k)} = M_{k+1}\delta^{(k)} + (A_{k+1} - A_k)r_{k+1}$$

can be used to replace $\gamma_o^{(k)}$ in any quasi-Newton updating formula. This idea is essentially suggested by M.C. Bartholomew-Biggs [2]. However, the condition $\delta^{(k)\top}\gamma_n^{(k)} > 0$, required for maintaining the positive definite approximations, may not be guaranteed by line searches. This difficulty can be avoided by using a safeguarded value

$$\max\left\{\delta^{(k)\top}\gamma_n^{(k)}, 0.01\delta^{(k)\top}\gamma_o^{(k)}\right\}$$

in place of $\delta^{(k)\top}\gamma^{(k)}$ in updating formulas [1].

For nonlinear least squares problems, both the quasi-Newton and Gauss–Newton methods generate approximations to Hessian matrices of objective functions only from their first order derivative information. The convergence rate is superlinear when any quasi-Newton, such as BFGS method is used to any differentiable nonlinear least squares problem, but the special structure of the problem functions is not taken into account and many iterations are required to build up a satisfactory approximation to the Hessian. The Gauss–Newton method takes the advantage of the special form of nonlinear least squares and better approximations to Hessian matrices are directly obtained from the Jacobian matrix $A(x)$ of $r(x)$ for zero and small residual problems. The damped Gauss-Newton method converges at a quadratic rate for zero residual problems and at a fast linear rate for small residual problems, which in limited precision may be preferable to superlinear convergent methods. However, the method converges slowly for large residual problems, even fails on singular problems. Hence, the damped Gauss–Newton method is generally preferred for zero and small residual problems, but should be avoided for large residual and singular problems. Attempts have been made to combine the best features of both kind of methods. These include *adaptive methods* [4,13], *hybrid methods* [1,6], and *factorized quasi-Newton methods* [14,15].

In adaptive methods approximations to Hessian $G(x)$ are obtained by approximating the nonlinear term C_k in G_k by a symmetric matrix S_k and keeping the Gauss–Newton matrix M_k unchanged, that is, to define

$$B_k = M_k + S_k$$

and to develop updating formulas for the matrix S_k. This idea was first suggested by K.M. Brown and J.E. Dennis in 1973. In their method $S_k = \sum_{i=1}^m r_i(x^{(k)})$ $S_i^{(k)}$ and $S_i^{(k)}$ approximates $\nabla^2 r_i(x^{(k)})$ obtained from $S_i^{(k-1)}$ using certain quasi-Newton updating formula. In

methods of C.G. Broyden and Dennis (1973), J.T. Bett (1976), Bartholomew-Biggs (1977), P.E. Gill and W. Murray (1978), and Dennis, D.M. Gay and R.E. Welsch (1981) the formulas for updating S_k are defined by using certain quasi-Newton-like updating formulas, for example, Gill and Murray used the BFGS-like formula

$$S_{k+1} = S_k + \frac{\gamma^{(k)}\gamma^{(k)\top}}{\delta^{(k)\top}\gamma^{(k)}} - \frac{W_k\delta^{(k)}\delta^{(k)\top}W_k}{\delta^{(k)\top}W_k\delta^{(k)}},$$

where $W_k = M_{k+1} + S_k$, while Dennis, Gay and Welsch developed an updating formula for S_k:

$$S_{k+1} = S_k - \frac{\eta^{(k)}\gamma^{(k)\top} + \gamma^{(k)}\eta^{(k)\top}}{\delta^{(k)\top}\gamma^{(k)}} + \frac{\eta^{(k)\top}\delta^{(k)}}{(\delta^{(k)\top}\gamma^{(k)})^2}\gamma^{(k)}\gamma^{(k)\top},$$

where $\eta^{(k)} = S_k\delta^{(k)} - y^{(k)}$ and $y^{(k)} = (A_{k+1} - A_k)\, r_{k+1}$. These methods attempt to get better approximations to G_k from first order derivative information by using the special structure of nonlinear least squares. However, since the positive definiteness of the resulting approximation $B_k = M_k + S_k$ can not be guaranteed, these methods must be incorporated with trust region techniques. This increases the complexity of methods. Theoretical analysis and practical calculations show that the Dennis–Gay–Welsch adaptive method is superlinearly convergent and effective.

At each iteration of a hybrid method, the approximation B_k is simply chosen either the Gauss–Newton matrix M_k or a quasi-Newton matrix, for example BFGS matrix, by defining a test T_k, that is,

$$B_{k+1} = \begin{cases} \text{BFGS}(B_k, \delta^{(k)}, \gamma^{(k)}) & \text{if } T_k \text{ holds,} \\ M_{k+1} & \text{otherwise,} \end{cases}$$

where BFGS(B_k, $\delta^{(k)}$, $\gamma^{(k)}$) denotes the BFGS updating formula. Thus each step of a hybrid method is either a Gauss–Newton step or a BFGS step. In developing a hybrid method, a test must be derived to distinguish between these two steps. A reasonable test should have the capability to differentiate problems so that the method ultimately takes the damped Gauss–Newton method for zero and small residual problems or the BFGS method for large residual and singular problems. M. Al-Baali and R. Fletcher [1] proposed a test

$$\text{T}_\text{k}: \ \Delta(B_k, \delta^{(k)}, \gamma^{(k)}) \le \Delta(M_{k+1}, \delta^{(k)}, \gamma^{(k)}),$$

based on the quantity

$$\Delta(A, \delta^{(k)}, \gamma^{(k)}) = \left(a_k^2 - 2\frac{1}{b_k} + 1\right)^{1/2},$$

which is regarded as a measure of the approximation error of the matrix A to G_{k+1}, where

$$a_k = \frac{\gamma^{(k)\top}A^{-1}\gamma^{(k)}}{\delta^{(k)\top}\gamma^{(k)}}, \quad b_k = \frac{\delta^{(k)\top}A\delta^{(k)}}{\delta^{(k)\top}\gamma^{(k)}}.$$

Though the resulting hybrid method is robust and effective, the test needs extra $O(n^2)$ arithmetic operations at each iteration and the resulting method is only linearly convergent [6]. Fletcher and C.X. Xu [6] proposed a simple test

$$\text{T}_\text{k}: \ \frac{f_k - f_{k+1}}{f_k} \le \epsilon,$$

where $\epsilon \in (0, 1)$ is a preset parameter. Numerical experiences show that the method is not sensitive to the choice of the value ϵ, but the value $\epsilon = 0.2$ is recommended. The test is simple and theoretical analysis shows that the method ultimately takes the Gauss–Newton method for zero residual problems and the BFGS method for nonzero residual problems. Therefore, the method maintains the convergence properties of BFGS method and combines the best features of both the Gauss–Newton and BFGS methods. Practical calculations show that the later hybrid method is better than the Fletcher–Al-Baali hybrid method.

Factorized quasi-Newton methods for nonlinear least squares take the approximations B_k in the form

$$B_k = (A_k + L_k)(A_k + L_k)^\top$$

and develop updating formulas for the matrix L_k such that $L_kL_k^\top + A_kL_k^\top + L_kA_k^\top$ approximates the second part C_k of G_k. If the matrix $(A_k + L_k)$ is of full rank, the search direction $s^{(k)}$ obtained from the system $B_ks = -g_k$ is guaranteed to be descent. The updating formulas for L_k can similarly derived from the quasi-Newton equation

$$(A_{k+1} + L_{k+1})(A_{k+1} + L_{k+1})^\top\delta^{(k)} = \gamma^{(k)}.$$

Using the theory of *generalized inverses* of matrices, Xu, X.F. Ma and M.Y. Kong derived a class of updating for-

mulas for L_k

$$L_{k+1} = L_k + (a-d)\frac{V_k \delta^{(k)} \gamma^{(k)\top}}{\delta^{(k)\top}\gamma^{(k)}} + b\frac{V_k W_k^{-1}\gamma^{(k)}\gamma^{(k)\top}}{\delta^{(k)\top}\gamma^{(k)}} - c\frac{V_k\delta^{(k)}\delta^{(k)\top}W_k}{\delta^{(k)\top}W_k\delta^{(k)}},$$

where

$$V_k = A_{k+1} + L_k, \quad W_k = V_k V_k^\top$$

a, b, c and d are parameters satisfying the following equations:

$$\frac{\delta^{(k)\top}W_k\delta^{(k)}}{\delta^{(k)\top}\gamma^{(k)}}a^2 + 2ab + \frac{\gamma^{(k)\top}W_k^{-1}\gamma^{(k)}}{\delta^{(k)\top}\gamma^{(k)}} = 1,$$
$$\frac{ad}{\delta^{(k)\top}\gamma^{(k)}} = \frac{bc}{\delta^{(k)\top}W_k\delta^{(k)}},$$
$$c + d = 1.$$

The resulting approximation $B_{k+1} = (A_{k+1} + L_{k+1})(A_{k+1} + L_{k+1})^\top$ are a class of Broyden-like updating formulas

$$B_{k+1} = \alpha B_{k+1}^{\mathrm{BFGS}} + \beta B_{k+1}^{\mathrm{DFP}},$$

where $\alpha = c^2 + 2cd$, $\beta = d^2$ and $\alpha + \beta = (c+d)^2 = 1$, B_{k+1}^{BFGS} and B_{k+1}^{DFP} are in previous forms with B_k being replaced by W_k. Both these formulas for BFGS-like and DFP-like are first obtained by H. Yabe and T. Takahashi [15] from their proposed updating formulas for L_{k+1} which can be obtained in the above updating formulas of L_{k+1} by setting $c = 0$, $d = 1$, $b = (\delta^{(k)\top}\gamma^{(k)}/\gamma^{(k)\top}W_k^{-1}\gamma^{(k)})^{1/2}$ for BFGS-like and $c = 1$, $d = 0$, $a = (\delta^{(k)\top}\gamma^{(k)}/\delta^{(k)\top}W_k\delta^{(k)})^{1/2}$ for DFP-like, respectively.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- ABS Algorithms for Optimization
- Automatic Differentiation: Calculation of Newton Steps
- Conjugate-Gradient Methods
- Contraction-mapping
- Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- Gauss–Newton Method: Least Squares, Relation to Newton's Method
- Generalized Total Least Squares
- Global Optimization Methods for Systems of Nonlinear Equations
- Interval Analysis: Systems of Nonlinear Equations
- Interval Newton Methods
- Large Scale Trust Region Problems
- Least Squares Orthogonal Polynomials
- Least Squares Problems
- Local Attractors for Gradient-related Descent Iterations
- Nondifferentiable Optimization: Newton Method
- Nonlinear Least Squares Problems
- Nonlinear Least Squares: Trust Region Methods
- Nonlinear Systems of Equations: Application to the Enclosure of All Azeotropes
- Unconstrained Nonlinear Optimization: Newton–Cauchy Framework

References

1. Al-Baali M, Fletcher R (1985) Variational methods for non-linear least squares. J Oper Res Soc 36:405–421
2. Bartholomew-Biggs MC (1977) The estimation of the Hessian matrix in nonlinear least squares problems with non-zero residuals. Math Program 12:67–80
3. Broyden CG (1965) A class of methods for solving nonlinear simultaneous equations. Math Comput 19:577–593
4. Dennis JE, Gay DM, Welsch RE (1981) An adaptive nonlinear least-squares algorithm. TOMS 7:348–368
5. Fletcher R, Powell MJD (1974) On the modification of LDL$^\mathsf{T}$ factorization. Math Comput 28:1067–1087
6. Fletcher R, Xu CX (1987) Hybrid methods for nonlinear least squares. IMA J Numer Anal 7:371–389
7. Hartley HO (1961) The modified Gauss–Newton method for the fitting of nonlinear regression function by least squares. Technometrics 3:269–280
8. Levenberg K (1944) A method for the solution of certain nonlinear problems in least squares. Quart Appl Math 2:164–168
9. Marquardt DW (1963) An algorithm for least squares estimation of nonlinear parameters. SIAM J 11:431–441
10. Moré JJ (1977) The Levenberg–Marquardt algorithm: implementation and theory. In: Watson GA (ed) Numerical Analysis: Dundee. Lecture Notes Math. Springer, Berlin
11. Powell MJD (1970) A hybrid method for nonlinear equations. In: Rabinowitz P (ed) Numerical Methods for Nonlinear Algebraic Equations. Gordon and Breach, New York
12. Powell MJD (1975) Some global convergence properties of a variable metric algorithm for minimization without exact line searches. AERE Harwell Report CSS15
13. Xu CX (1987) Hybrid methods for nonlinear least squares and related problems. PhD Thesis, Univ. Dundee

14. Xu CX, Ma XF, Kong MY (1996) A class of factorized quasi-Newton methods for nonlinear least squares problems. J Comput Math 14:143–158
15. Yabe H, Takahashi T (1991) Factorized quasi-Newton methods for nonlinear least squares problems. Math Program 51:75–100

Nonlinear Least Squares Problems

NLS

Chengxian Xu
Xian Jiaotong University, Xian, China

MSC2000: 90C30

Article Outline

Keywords
Solution of Simultaneous Equations
Curve (Data) Fitting
Optimal Designs
See also
References

Keywords

Nonlinear least squares; Residuals; Kuhn–Tucker optimality condition; Descent method; Line search; Trust region

Nonlinear least squares problems are among the most commonly occurring and important applications of optimization techniques. The problem is to find minima of a real valued function that has the form of a sum of some nonlinear functions of several independent variables

$$\min f(x) = \frac{1}{2}\sum_{i=1}^{m}[r_i(x)]^2 = \frac{1}{2}r(x)^\top r(x)\,,$$

where $r_i(x)$, $i = 1, \ldots, m$, are m nonlinear functions defined on $\mathbf{R}^n$, $r(x)$ is the vector representation of $r_i(x)$, $i = 1, \ldots, m$, $m \geq n$, and the real number 1/2 is generally placed for convenience.

Approaches for least squares can be traced back to more than two hundred years ago. It is well-known that C.F. Gauss proposed the method, called *Gauss–Newton method*, in 1809 to estimate motion orbits of planets from observation data. However, early in the 1750s some methods had been suggested by P.S. Laplace, L. Euler, etc. to deal with measuring data in astronomical observations. In 1805, A.M. Legendre proposed a method to determine the orbit of comets and the meridian of the earth. He called it least squares method, but did not demonstrate its optimality in theory. The advent of computer promoted the development and applications of numerical analysis including optimization methods and nonlinear least squares solutions.

Nonlinear least squares problems arise in various practical areas such as scientific computing, scientific experiments, engineering designs, survey and observations, geological prospecting, physical science, mathematics and so on. It is particularly useful in data processing and error estimations. Here are a few examples to illustrate the applications of nonlinear least squares problems.

Solution of Simultaneous Equations

It is frequently concerned with in scientific computing to find a solution of a system of nonlinear equations

$$\begin{aligned} &f_1(x_1, \ldots, x_n) = 0\,,\\ &\cdots \quad \cdots\\ &f_m(x_1, \ldots, x_n) = 0\,, \end{aligned}$$

where $x_1, \ldots, x_n$ are unknowns. The system is *underdetermined* if $m < n$, *well-determined* if $m = n$ and *overdetermined* if $m > n$. It is usually not possible to obtain an exact solution for an overdetermined system and one possibility is to seek a best least squares solution, that is, to find x^* such that the function

$$f(x) = \frac{1}{2}\sum_{i=1}^{m}[f_i(x)]^2$$

is minimized at x^*.

Curve (Data) Fitting

Perhaps the most frequently solved of all nonlinear least squares problems are data fitting problems. In scientific researches, for example chemical or physical experiments, it is often encountered that the dependence of some observable quantity y on some independent variable(s) t is predicted, based on theoretical ground, to

have the form

$$y = h(x, t)$$

and m values $y_1, \ldots, y_m$ are measured for points $t_1, \ldots, t_m$ where $x \in \mathbf{R}^n$ is an adjustable parameter vector. We are required to choose an optimal parameter vector x^* such that the function $h(x, t)$ best fits the data in the least squares sense, that is, define the function $f(x)$ by

$$f(x) = \frac{1}{2}\sum_{i=1}^{m}[h(x, t_i) - y_i]^2 = \frac{1}{2}r(x)^\top r(x)\,,$$

and x^* is found by minimizing the function $f(x)$ where $r_i(x) = h(x, t_i) - y_i$, $i = 1, \ldots, m$ are called *residuals*.

In practice, both the values $y_1, \ldots, y_m$ and $t_1, \ldots, t_m$ are usually subject to measured errors. In the above conventional approach, implicit assumptions are made that only the values $y_1, \ldots, y_m$ are subject to measured errors and that the values $t_1, \ldots, t_m$ are either exact or contain negligible errors. In many applications, however, this is an oversimplification and the use of the above nonlinear least squares problems may lead to bias in the estimated parameters and variance values. It is then necessary to take proper account of errors in all variable values. The function of the resulting problem has the form

$$\begin{aligned} f(x, \tau) &= \frac{1}{2}\sum_{i}^{m}[(h(x, \tau_i) - y_i)^2 + (\tau_i - t_i)^2] \\ &= \frac{1}{2}[r(x, \tau)^\top r(x, \tau) + e(\tau)^\top e(\tau)]\,, \end{aligned}$$

where $\tau = (\tau_1, \ldots, \tau_m)^\mathsf{T}$ and $r(x, \tau)$, $e(\tau)$ are m-vectors with components $r_i(x, \tau) = h(x, \tau_i) - y_i$ and $e_i(\tau) = \tau_i - t_i$, $i = 1, \ldots, m$, respectively. Taking errors in all variables into account increases the complexity of the problem. For example, the simplest linear problem $h(x, t) = x_1 + x_2 t$ is no longer a linear problem when errors in all variables are taken into account.

Optimal Designs

Let us consider the optimal design of a coaxial cable circuit. Let $x_1, \ldots, x_n$ be the design parameters. For a given circuit, its performance index is a function of $x_1, \ldots, x_n$ and ω

$$h(x_1, \ldots, x_n, \omega)\,,$$

where ω is the frequency of the circuit. The aim of the circuit optimal design is to determine the circuit parameters $x_1, \ldots, x_n$ such that the performance index of the circuit best approximates a given characteristic function $\psi(\omega)$ in a given interval $[\alpha, \beta]$ of the frequency ω. This can be expressed as to find $x^* \in \mathbf{R}^n$ which minimizes the function

$$\begin{aligned} f(x) &= \frac{1}{2}\sum_{i=1}^{m}[r_i(x)]^2 \\ &= \frac{1}{2}\sum_{i=1}^{m}[h(x_1, \ldots, x_n, \omega_i) - \psi(\omega_i)]^2 \end{aligned}$$

where $\alpha \le \omega_1 < \cdots < \omega_m \le \beta$. Meanwhile the circuit parameters are generally subject to some restricted factors of supplied cables such as geometric shapes, diameters, medium materials and so on. These subjects can be expressed as constrained conditions in the form

$$c_j(x_1, \ldots, x_n) \ge 0, \quad j = 1, \ldots, p\,.$$

Therefore, the mathematical model of the optimal design of a circuit generally has the form

$$\begin{cases} \min & f(x) = \frac{1}{2}\sum_{i=1}^{m}[r_i(x)]^2 \\ \text{s.t.} & c_j(x) \ge 0, \quad j = 1, \ldots, p\,. \end{cases}$$

Nonlinear least squares problems can be classified into unconstrained and constrained ones depending on whether there exist constraints on variables $x \in \mathbf{R}^n$. For example, the resulting problem in the optimal design of the coaxial cable circuit is a constrained nonlinear least squares problem while the problem formed in the solution of simultaneous equations is an unconstrained nonlinear least squares problem. In some situations, the solution of a unconstrained nonlinear least squares problem may be unacceptable. To avoid such a situation, some restrictions on parameter vector x can be imposed in the form of constraints. For example, some variables are required to be nonnegative, some may be bounded by lower and/or upper bound(s), and some dependent relationships among variables must be satisfied. This also results in constrained nonlinear least squares problem. When errors in all variables are taken into account, the resulting problem is generally called a *generalized nonlinear least squares problem*. This increases not only the problem complexity, but also the

problem dimension. An advantage of the generalized nonlinear least squares problems that can be exploited is that the problem variables are *separable*. An optimization problem is called separable if the optimization with respect to some of the variables is easier than with respect to the others. Of course, there are other separable nonlinear least squares problems such as arose in curve fitting, component analysis and orthogonal regression (see [14]).

When $r(x)$, and hence $f(x)$ is twice continuously differentiable, the gradient and the Hessian matrix of $f(x)$ are given by

$$\nabla f(x) = A(x)r(x),$$

$$\nabla^2 f(x) = A(x)A(x)^\top + \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x)\,,$$

where $A(x) = [\nabla r_1(x), \ldots, \nabla r_m(x)]$. Let x^* be a minimizer of a NLS problem. The problem is called a *zero residual problem* if $r(x^*) = 0$ and hence $f(x^*) = 0$ and a *nonzero residual problem* if $r(x^*) \neq 0$. A nonzero residual problem is called *small residual* if $r(x^*)$ is small or the second part of $\nabla^2 f(x^*)$ is relatively small compared with the first part of $\nabla^2 f(x^*)$, otherwise it is called *large residual*.

Methods for nonlinear least squares are iterative type and are based upon trying to find a point x^* at which the so-called *Kuhn–Tucker optimality condition* is satisfied. These methods are generally of *descent* type. From a given initial point, these methods generate a sequence $\{x^{(k)}\}$ such that either one point of the sequence or any accumulation point of $\{x^{(k)}\}$ is a *Kuhn–Tucker point* (referred to as a *KT point*). The typical behavior of a method is that if the iteration is not terminated at some point $x^{(k)}$, the values $\phi(x^{(k)})$ of a *merit function* for the problem are monotonically decreased so that the iterates $x^{(k)}$ move steadily towards a neighborhood of a KT point x^*, and then converges rapidly to the point x^*. The basic structure of the kth iteration of such a method has the form

- Check if $x^{(k)}$ is a KT point.
- If $x^{(k)}$ is not a KT point, determine a $\delta^{(k)}$ such that

$$\phi(x^{(k)} + \delta^{(k)}) < \phi(x^{(k)}).$$

- $x^{(k+1)} = x^{(k)} + \delta^{(k)}$.

The information of second order derivatives of problem functions are required to determine if a KT point is a local minimizer. Since the evaluation of second order derivatives are time consuming and sometimes the second order derivatives of functions are not available, most nonlinear least squares methods do not evaluate second order derivatives and just try to locate a KT point. A KT point may not be a local minimizer. However, there exist other features of methods such as the monotonic decreasing property of the sequence $\{\phi(x^{(k)})\}$, which usually imply that a KT point is a local minimizer, except in rare cases. A descent method with this property is called globally convergent, that is, the method does not require the initial point $x^{(1)}$ close to x^*.

A globally convergent method for optimization usually defines a merit function ϕ to force convergence from poor starting points. For unconstrained nonlinear least squares problems, the choice of ϕ is simple. It is natural to choose the objective function f as ϕ. For constrained nonlinear least squares problems, the choice of ϕ is complicated by the fact that $x^{(k)} + \delta^{(k)}$ should move closer to satisfying the constraints than $x^{(k)}$ and $\delta^{(k)}$ yields a reasonable decrease in the objective function. A number of merit functions are available for constrained nonlinear least squares problems solution. These include the ℓ_1 *penalty function* proposed by S.P. Han [11] and used by R. Fletcher [9], the *augmented Lagrange functions* of M.R. Hestenes [12] and Fletcher [7], the *merit function* of G. Di Pillo and L. Grippo [6] and the *merit function* of P.T. Boggs and J.W. Tolle [2].

Strategies are available to achieve the descent property in nonlinear least squares methods. These are: *line search* strategy and *trust region* strategy. Line search strategy transfers a multivariable minimization problem into a series of sub-minimization problems with single variable, and are the most commonly used in practice. In line search methods, $\delta^{(k)}$ is obtained in the form $\delta^{(k)} = \alpha_k s^{(k)}$ where $s^{(k)}$ is a descent direction of the merit function $\phi(x)$ at $x^{(k)}$ and $\alpha_k > 0$ is a steplength along the direction $s^{(k)}$. The descent direction $s^{(k)}$ is generally obtained as a minimizer of some model problem which is a local approximation to the original problem at $x^{(k)}$ and the steplength α_k is determined by some line search method so that $x^{(k)} + \alpha_k s^{(k)}$ gives a sufficient decrease in a chosen merit function from $x^{(k)}$. Specifically, α_k is determined such that $\phi(x^{(k)} + \alpha_k s^{(k)}) < \phi(x^{(k)})$. For methods with line search, the convergence and the convergence rate depend upon the reduction on the merit

function at each iteration, which relies on the choices of the descent direction $s^{(k)}$ and the steplength α_k. Theoretically, exact line searches satisfying $\alpha_k = \text{argmin}\, \phi(x^{(k)} + \alpha s^{(k)})$ usually gives the maximal reduction on the function $\phi(x^{(k)} + \alpha s^{(k)})$. But determining α_k by exact line search is not necessary and inefficient when iterate $x^{(k)}$ is remote from x^*. Practical line search methods search for a steplength satisfying certain conditions. These conditions are readily satisfied so that the steplength can be effectively determined and guarantee the convergence of a method when the search direction $s^{(k)}$ is sufficient descent [1]. Among various conditions, *Goldstein's conditions* are widely used in practice. These conditions require the steplength α_k to satisfy

$$\phi(x^{(k)} + \alpha s^{(k)}) \le \phi(x^{(k)}) + \rho\alpha\nabla\phi(x^{(k)})^\top s^{(k)},$$
$$\phi(x^{(k)} + \alpha s^{(k)}) \ge \phi(x^{(k)}) + \sigma\alpha\nabla\phi(x^{(k)})^\top s^{(k)},$$

where $0 < \rho < \sigma < 1$. Usually, these two conditions define an interval of acceptable α-values. A disadvantage of the second condition is that the minimum of $\phi(x^{(k)} + \alpha s^{(k)})$ may be excluded to the left of the interval. A remedy to avoid this case is to use

$$\nabla\phi(x^{(k)} + \alpha s^{(k)})^\top s^{(k)} \ge \sigma\nabla\phi(x^{(k)})^\top s^{(k)}$$

to replace that condition. Numerous line search methods are available. Strategies used in line search methods generally consist of *bracketing* and *sectioning*. The simplest line search methods are pattern searches such as *golden section search* and *Fibonnaci section search* [1]. These pattern searches use only evaluated function values. When the first order derivative information are available, the simplest line search is the *backtracking* which seeks the smallest integer i satisfying

$$\phi(x^{(k)} + t^i\alpha_0 s^{(k)}) \le \phi(x^{(k)}) + \rho t^i\alpha_0\nabla\phi(x^{(k)})^\top s^{(k)}$$

and set $\alpha_k = t^j\,\alpha_0$ if j is such an integer. Effective line search methods to determine α_k satisfying the Goldstein conditions are also available [1]. These methods employ sectioning schemes and interpolations.

Trust region strategy generates $\delta^{(k)}$ by solving some model subproblem with a trust region constraint. The trust region defined by $\| \delta \| \le \Delta_k$ is a neighborhood about the current point $x^{(k)}$ and is adjusted in such a way that the subproblem model is believed to have adequately approximated the chosen merit function in that region. For a given trust region radius Δ_k, the solution $\delta^{(k)}$ of minimizing the subproblem model within the region $\| \delta \| \le \Delta_k$ is sought. If a satisfactory reduction on the merit function is obtained at $x^{(k)} + \delta^{(k)}$, $x^{(k)} + \delta^{(k)}$ is accepted as $x^{(k+1)}$. If the computed step $\delta^{(k)}$ is not acceptable, the sub-problem model is not accurate enough in that region and the radius of the trust region is reduced to improve the accuracy of the approximation and the step is recomputed. The trust region radius may be increased after an acceptable step. Methods with trust region converge globally. An obstacle preventing the trust region methods from common use is the effective solution of trust region subproblems. Repeated solution of system of linear equations with modified coefficient matrices are required to obtain a satisfactory $\delta^{(k)}$ and increase the complexity of trust region methods. Now effective approximate solution methods for the solution of trust region subproblems exist. These include positive definite *dogleg path* method [5,13], indefinite dogleg path methods [19], optimal path method in two-dimensional subspaces [3] and conjugate gradient method [15].

Subproblem models are generally local approximations to a chosen merit function at current iterate point $x^{(k)}$. These approximations are generally linear or quadratic. A linear approximation is the simplest function and a quadratic approximation is usually more accurate than linear approximation in certain neighborhood of $x^{(k)}$. The quadratic function is one of the simplest smooth functions and the minimum of a model problem with quadratic function is well-determined and is relatively easy to determine.

As an important class of optimization problems, any method for general minimization can be used to solve nonlinear least squares problems. However, special methods which take the advantage of the special structure of the objective function in nonlinear least squares are available. Most special methods are based on the well-known Gauss–Newton (G-N) method which requires only the first order derivatives of problem functions. In the G-N method, the matrix $A(x^{(k)})\,A(x^{(k)})^\top$ is used to approximate the Hessian matrix $\nabla^2 f(x^{(k)})$. Since $r(x^*)$ may be small or zero as $f(x)$ is minimized, this may be a good approximation when $x^{(k)}$ is close to x^*. It is well known that for zero residual problems, the local convergence rate of the G-N method is quadratic, but for small residual problems,

the convergence of the method is at most linear and even not converge for large residual problems. According to the information used in a method, methods for nonlinear least squares can be divided into first order derivative methods, second order derivative methods and methods without derivatives. Among them the first order derivative methods are the most commonly used. These include (damped) G-N method, quasi-Newton methods, hybrid methods [1,10], adaptive method [4], factorized quasi-Newton methods [17,18] and methods with quasi-Newton corrections to Gauss–Newton matrix. As for second order derivative methods, the Newton method is the most famous. However, the disadvantages of the Newton method such as the evaluation of second order derivatives and convergence only local make the method rarely used in practice. As for methods without using derivatives, one can make a choice among the direction set methods [8] and the hybrid Gauss–Newton–Broyden method [16]. Theoretical analysis and numerical results show that these methods have good convergence properties and are robust and trust.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- ABS Algorithms for Optimization
- Gauss–Newton Method: Least Squares, Relation to Newton's Method
- Generalized Total Least Squares
- Least Squares Orthogonal Polynomials
- Least Squares Problems
- Nonlinear Least Squares: Newton-type Methods
- Nonlinear Least Squares: Trust Region Methods

References

1. Al-Baali M, Fletcher R (1985) Variational methods for nonlinear least-squares. J Oper Res Soc 36:405–421
2. Boggs PT, Tolle JW (1987) Merit functions and nonlinear programming. SIAM Conf. Optimization, Houston
3. Byrd RH, Schnabel RB, Shultz GA (1988) Approximate solution of the trust region problem by minimization over two-dimensional subspaces. Math Program 40:247–263
4. Dennis JE, Gay DM, Welsch RE (1981) An adaptive nonlinear least-squares algorithm. TOMS 7:348–368
5. Dennis JE, Mei HHW (1979) Two new unconstrained optimization algorithms which use function and gradient values. JOTA 28:453–482
6. Di Pillo G, Grippo L (1979) A new class of augmented Lagrangian in nonlinear programming. SIAM J Control Optim 17:618–628
7. Fletcher R (1973) An exact penalty function for nonlinear programming with inequalities. Math Program 5:129–150
8. Fletcher R (1980) Practical methods of optimization: Unconstrained optimization, vol 1. Wiley, New York
9. Fletcher R (1984) An ℓ^1 penalty method for nonlinear constraints. In: Boggs PT, Byrd RH, Schnabel RB (eds) Numerical Optimization. SIAM, Philadelphia, pp 26–40
10. Fletcher R, Xu CX (1987) Hybrid methods for nonlinear least squares. IMA J Numer Anal 7:371–389
11. Han SP (1977) A globally convergent method for nonlinear programming. JOTA 22:297–309
12. Hestenes MR (1977) Multiplier and gradient methods. JOTA 4:303–320
13. Powell MJD (1970) A hybrid method for nonlinear equations. In: Rabinowitz P (ed) Numerical Methods for Nonlinear Algebraic Equations. Gordon and Breach, New York, pp 87–114
14. Ruhe A, Wedin PA (1980) Algorithms for separable nonlinear least squares problems. SIAM Rev 22:318–337
15. Steihaug T (1983) The conjugate gradient method and trust regions in large scale optimization. SIAM J Numer Anal 20:626–637
16. Xu CX (1990) Hybrid methods for nonlinear least squares problems without calculating derivatives. JOTA 65:555–575
17. Xu CX, Ma XF, Kong MY (1996) A class of factorized quasi-Newton methods for nonlinear least squares problems. J Comput Math 14:143–158
18. Yabe H, Takahashi T (1991) Factorized quasi-Newton methods for nonlinear least squares problems. Math Program 51:75–100
19. Zhang JZ, Xu CX (1999) A class of indefinite dogleg methods for unconstrained minimization. SIAM J Optim 9:646–667

Nonlinear Least Squares: Trust Region Methods

CHENGXIAN XU
Xian Jiaotong University, Xian, China

MSC2000: 49M37

Article Outline

Keywords
See also
References

Keywords

Trust region method; Solution of a system; Optimal path; Dogleg path; Negative curvature

Descent methods for nonlinear least squares problems generate a sequence $\{x^{(k)}\}$ such that $f(x^{(k+1)}) < f(x^{(k)})$ and $x^{(k)}$ converges to a local minimizer of the objective function $f(x)$. Methods with *line searches* generate the sequence using the iteration

$$x^{(k+1)} = x^{(k)} + \alpha_k s^{(k)} ,$$

where $s^{(k)}$ is a descent direction of $f(x)$ at $x^{(k)}$ and α_k is a steplength along the direction $s^{(k)}$ determined by line searches. The search direction $s^{(k)}$ is obtained as the solution of the system

$$B_k s = -g_k ,$$

where g_k is the gradient of $f(x)$ at $x^{(k)}$ and B_k is either the Hessian G_k of $f(x)$ at $x^{(k)}$ or its approximation. The matrix B_k is required to be positive definite, so that the solution $s^{(k)}$ which is the unique minimizer of the quadratic model

$$q_k(\delta) = f_k + g_k^\top \delta + \frac{1}{2}\delta^\top B_k \delta$$

is guaranteed to be a descent direction, where $q_k(\delta)$ is a local approximation to $f(x)$ at $x^{(k)}$, $f_k = f(x^{(k)})$ and $\delta = x - x^{(k)}$.

When the matrix B_k is not positive definite that are often encountered in practical calculation, the quadratic model does not have a unique minimizer and methods with line searches may not be defined. A more realistic approach is to take $x^{(k+1)} = x^{(k)} + \delta^{(k)}$. The $\delta^{(k)}$ minimizes the quadratic model within a neighborhood $N(x^{(k)}, \Delta_k)$ of the point $x^{(k)}$ in which the quadratic function is believed adequately to approximate the function $f(x)$. The neighborhood $N(x^{(k)}, \Delta_k)$ is generally called a *trust region* and methods having this framework are called *trust region methods*. These methods can retain the rapid rate of convergence of Newton type methods, but are also generally applicable and *globally convergent*.

The development of trust region methods can be traced back to the work of K. Levenberg [8] and D.W. Marquardt [9] on unconstrained *nonlinear least squares* problems

$$\min f(x) = \frac{1}{2}\sum_{i=1}^{m}[r_i(x)]^2 = \frac{1}{2} r(x)^\top r(x) ,$$

where $r(x) = (r_1(x), \ldots, r_m(x))^\top$ and $r_i(x)$, $i = 1, \ldots, m$, are nonlinear functions of $x \in \mathbf{R}^n$. Assuming that $r_i(x)$, $i = 1 \cdots, m$, hence $f(x)$ is twice continuously differentiable, the gradient and the Hessian matrix of $f(x)$ are defined by

$$\begin{aligned} g(x) &= \nabla f(x) = A(x) r(x), \\ G(x) &= \nabla^2 f(x) \\ &= A(x)A(x)^\top + \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x) . \end{aligned}$$

The *Levenberg–Marquardt method* is a modification of the well-known *Gauss–Newton method* and is based upon the idea that when the full Gauss–Newton step fails, a proper bias towards the steepest descent direction may generate a satisfactory reduction on the function $f(x)$. Thus the step $\delta^{(k)}$ between iterates of the Levenberg–Marquardt method is a solution of the system

$$(A_k A_k^\top + \mu_k I)\delta = -g_k$$

for some properly chosen $\mu_k \geq 0$. The application of the Levenberg–Marquardt method to general unconstrained minimization is given by S.M. Goldfeld, R.E. Quandt and H.F. Trotter [6], in which the matrix $A_k A_k^\top$ in the above system is just replaced by the matrix B_k, that is,

$$(B_k + \mu_k I)\delta = -g_k .$$

Let $\delta^{(k)}$ be the solution of the system. Then $\delta^{(k)}$ solves the trust region subproblem

$$\begin{cases} \min & q_k(\delta) = f_k + g_k^\top \delta + \frac{1}{2}\delta^\top B_k \delta \\ \text{s.t.} & \|\delta\| \leq \Delta_k \end{cases}$$

with $\Delta_k = \|\delta^{(k)}\|$. Also for any given Δ_k, the solution $\delta^{(k)}$ of the trust region subproblem usually satisfies the system

$$(B_k + \mu_k I)\delta = -g_k, \quad \left\|\delta^{(k)}\right\| = \Delta_k ,$$

where $\mu_k \geq 0$ such that $B_k + \mu_k I$ is at least positive semidefinite, except for the case when B_k is positive definite and $\|B_k^{-1} g_k\| \leq \Delta_k$, then the solution is $\delta^{(k)} = -$

$B_k^{-1} g_k$. It can be seen that the solution $\delta^{(k)}$ can be obtained by controlling either the value μ_k or the radius Δ_k.

Early versions of the Levenberg–Marquardt and Goldfeld–Quandt–Trotter methods determined $\delta^{(k)}$ from the system by controlling μ_k and recently used trust region methods determined $\delta^{(k)}$ by controlling the radius Δ_k. Direct control of μ_k has a number of disadvantages. One is that there does not seem to have a reasonable and automatic initial choice for the value μ_1, while a reasonable value for initial Δ_1 can be a small fraction of the size $\|x^{(1)}\|$ of the starting point $x^{(1)}$. One more problem occurs when $x^{(k)} + \delta^{(k)}$ leads to an increase in the objective function $f(x)$. In this case, the function and derivative information evaluated at $x^{(k)}$ and $x^{(k)} + \delta^{(k)}$ can be used to estimate a required decrease in radius Δ_k, but it is not clear how these information are used to estimate a reasonable increase for μ_k.

In implementing recent forms of trust region methods, there are two main problems. One problem is how the trust region radius Δ_k shall be chosen. To prevent undue restriction of the step $\delta^{(k)}$, the trust region radius Δ_k should be as large as possible under the condition that $q_k(\delta)$ adequately approximates $f(x)$ in that region. Let $\delta^{(k)}$ be the solution of the trust region subproblem for given Δ_k, the agreement between $q_k(\delta)$ and $f(x)$ in the neighborhood $N(x^{(k)}, \Delta_k)$ can be measured by the comparison between the actual reduction in $f(x)$

$$\text{ared}(\delta^{(k)}) = f(x^{(k)}) - f(x^{(k)} + \delta^{(k)})$$

and the reduction predicted by the quadratic model

$$\begin{aligned}\text{pred}(\delta^{(k)}) &= f(x^{(k)}) - q_k(\delta^{(k)}) \\ &= -g_k^{\top}\delta^{(k)} - \frac{1}{2}\delta^{(k)\top} B_k \delta^{(k)} ,\end{aligned}$$

where B_k is either the Hessian matrix G_k of $f(x)$ or its approximation, for example $A_k A_k^{\top}$. If $\text{ared}(\delta^{(k)})$ is satisfactory compared with the pred $(\delta^{(k)})$, which implies a good reduction in f, the trust region is referred to as proper. Then $x^{(k)} + \delta^{(k)}$ is accepted as a new iterate point $x^{(k+1)}$ and the trust region radius may either keep unchanged or be increased in case the reduction in f is sufficient and the trust region constraint is active. If the computed step $\delta^{(k)}$ is not acceptable, which occurs when $q_k(\delta)$ is not accurate enough in $N(x^{(k)}, \Delta_k)$, then the radius Δ_k is reduced to improve accuracy of the approximation and the step is recomputed from the trust region subproblem with reduced Δ_k. A model trust region method with direct control of Δ_k can now be described as follows:

1) Give parameters $0 < \gamma_1 < 1 < \gamma_2$, $0 < \eta_1 < \eta_2 < 1$ and $\Delta_{\max} > 0$, initial point $x^{(1)}$ and $\Delta_1 (\leq \Delta_{\max})$ and set $k = 1$.
2) Calculate f_k, g_k. If $g_k = 0$ then terminate, else form B_k.
3) Solve the trust region subproblem for $\delta^{(k)}$.
4) Compute $\theta_k = \text{ared}(\delta^{(k)})/\text{pred}(\delta^{(k)})$.
5) If $\theta_k < \eta_1$, then $\Delta_k = \gamma_1 \Delta_k$ and go to step 3).
6) $x^{(k+1)} = x^{(k)} + \delta^{(k)}$,

$$\Delta_{k+1} = \begin{cases} \overline{\Delta} & \text{if } \theta_k \geq \eta_2 \\ & \text{and } \left\|\delta^{(k)}\right\| = \Delta_k, \\ \Delta_k & \text{otherwise,} \end{cases}$$

set $k = k + 1$, where $\overline{\Delta} = \min\{\gamma_2 \Delta_k, \Delta_{\max}\}$

The most important issue to implement a trust region method is the efficient solution of the trust region subproblem. There are three possible cases to determine the solution of the trust region subproblem:

a) *Newton step case*: B_k is positive definite and $\|B_k^{-1} g_k\| \leq \Delta_k$. The solution is $\delta^{(k)} = -B_k^{-1} g_k$.
b) *General case*: B_k is positive definite and $\|B_k^{-1} g_k\| > \Delta_k$ or B_k is indefinite and $\|(B_k - \mu_1^{(k)} I)^{+} g_k\| \geq \Delta_k$ where $(A)^{+}$ denotes the generalized inverse of the matrix A and $\mu_1^{(k)}$ is the smallest eigenvalue of B_k. The solution is $\delta^{(k)} = -(B_k + \mu_k I)^{-1} g_k$ with $\mu_k > \mu_\ell = \max\{0, -\mu_1^{(k)}\}$, $\|\delta^{(k)}\| = \Delta_k$.
c) *Hard case* [11]: B_k is indefinite and $\|(B_k - \mu_1^{(k)} I)^{+} g_k\| < \Delta_k$. The solution is $\delta^{(k)} = -(B_k - \mu_1^{(k)} I)^{+} g_k + t_k u_1^{(k)}$ where $u_1^{(k)}$ is the eigenvector of B_k corresponding to $\mu_1^{(k)}$ and t_k is determined from the equation $\|(B_k - \mu_1^{(k)} I)^{+} g_k + t u_1^{(k)}\| = \Delta_k$.

Since the hard case rarely occurs, the solution of trust region subproblem is closely related to the solution of the equation

$$\begin{aligned}\phi_k(\mu) &= \|\delta_k(\mu)\| - \Delta_k \\ &= \left\|(B_k + \mu I)^{-1} g_k\right\| - \Delta_k = 0 .\end{aligned}$$

This is a nonlinear equation with respect to μ and there is generally no finite method to find its exact solution. Since $\phi_k(\mu_\ell) > 0$ and $\phi_k(\infty) = -\Delta_k$, it is clear that the

solution can be found in the interval (μ_ℓ, ∞). Since $\phi_k(\mu)$ is a convex and continuous monotonic decreasing function in the interval (μ_ℓ, ∞), the solution of the equation is unique. M.D. Hebden [7] used the practical structure of the function $\phi_k(\mu)$ to propose a method that generates a sequence $\{\mu^{(j)}\}$ such that $\mu^{(j)} \to \mu_k$. Let $\mu^{(j)}$ be a current estimation to μ_k, a fractional function

$$\psi_k(\mu) = \frac{\alpha}{\mu + \beta} - \Delta_k$$

is used to approximate the function $\phi_k(\mu)$ such that

$$\psi_k(\mu^{(j)}) = \phi_k(\mu^{(j)}), \quad \psi_k'(\mu^{(j)}) = \phi_k'(\mu^{(j)})$$

where

$$\alpha = -\frac{(\phi_k(\mu^{(j)}) + \Delta_k)^2}{\phi_k'(\mu^{(j)})},$$

$$\beta = -\frac{\phi_k(\mu^{(j)}) + \Delta_k}{\phi_k'(\mu^{(j)})} - \mu^{(j)}.$$

Setting $\psi_k(\mu) = 0$ and substituting α and β give the iteration

$$\mu^{(j+1)} = \mu^{(j)} + \xi(\mu^{(j)}) \left[\frac{\|\delta_k(\mu^{(j)})\|}{\Delta_k} - 1 \right],$$

$$\xi(\mu^{(j)}) = \frac{\|\delta_k(\mu^{(j)})\|^2}{\delta_k(\mu^{(j)})^\top (B_k + \mu^{(j)} I)^{-1} \delta_k(\mu^{(j)})}.$$

J.J. Moré and D.C. Sorensen [11] also derived the iteration by applying the Newton method to the equation

$$h_k(\mu) = \frac{1}{\Delta_k} - \frac{1}{\|\delta_k(\mu)\|}.$$

The iteration can be calculated in the following way.

i) Factorize the matrix $(B_k + \mu^{(j)} I) = R^\top R$ with R an upper triangular matrix.

ii) Solve $R^\top R\delta = -g_k$ for $\delta_k(\mu^{(j)})$ using forward and backward substitutions.

iii) Solve $R^\top z = \delta_k(\mu^{(j)})$ for z_j using forward substitution.

iv)
$$\mu^{(j+1)} = \mu^{(j)} + \frac{\|\delta_k(\mu^{(j)})\|^2}{\|z_j\|^2} \left[\frac{\|\delta_k(\mu^{(j)})\|}{\Delta_k} - 1 \right].$$

To start the iteration, an initial value for $\mu^{(1)}$ is required. A natural choice is the value $\mu^{(1)} = \mu_\ell$. This choice needs the calculation of the smallest eigenvalue of the matrix B_k and will cause numerical difficulties when B_k is not positive definite. Moré [10] used the choice

$$\mu^{(1)} = \mu_{k-1} \frac{\Delta_{k-1}}{\Delta_k}$$

as the initial value of μ_k at the kth iteration where μ_{k-1} is the accepted value of the equation $\phi_{k-1}(\mu) = 0$ at the $(k-1)$th iteration. J.E. Dennis and R.B. Schnabel [4] proposed the choice

$$\mu^{(1)} = \mu_{k-1} + \xi(\mu_{k-1}) \left[\frac{\|\delta_{k-1}(\mu_{k-1})\|}{\Delta_k} - 1 \right],$$

$$\xi(\mu_{k-1}) = \frac{\|\delta_{k-1}(\mu_{k-1})\|^2}{\delta_{k-1}(\mu_{k-1})^\top (B_{k-1} + \mu_{k-1} I)^{-1} \delta_{k-1}(\mu_{k-1})},$$

which is an analog to the iteration for $\mu^{(j+1)}$. Of course, a safeguard strategy is imposed to force convergence, that is, lower and upper bounds for $\mu^{(j)}$ are provided and updated in iteration process. The iteration finds an approximate solution $\delta^{(k)}$ satisfying the *Hebden conditions* [7]

$$\text{pred}(\delta^{(k)}) \geq \tau(f_k - q_k^*),$$

$$\left| \left\| \delta^{(k)} \right\| - \Delta_k \right| \leq \rho \Delta_k,$$

where τ and ρ are positive constants and q_k^* is the optimal value of the trust region subproblem for given Δ_k. Then strong convergence result can be obtained (see [5]).

In the general case, the repeated *solution of the system* $(B_k + \mu^{(j)} I)\delta = -g_k$ for different values of $\mu^{(j)}$ to determine a satisfactory approximate solution $\delta^{(k)}$ of the trust region subproblem for given value Δ_k and the recomputation of $\delta^{(k)}$ for reduced values Δ_k may be required at each iteration. It is this complication that prevent trust region methods from wide use in past two decades. Most practical trust region methods attempt to find an approximate solution of the trust region subproblem in a reasonable amount of computational effort. G.A. Shultz, Schnabel and R.H. Byrd [13] proposed general conditions on the approximate solution $\delta^{(k)}$ to ensure a satisfactory reduction on $q_k(\delta)$ so that resulting trust region methods have strong convergence properties. These conditions are as follows:

A) There exist constants $\beta_1 > 0$ and $\sigma_1 > 0$ such that for all Δ_k and B_k

$$\text{pred}(\delta^{(k)}) \geq \beta_1 \|g_k\| \min\left\{\Delta_k, \sigma_1 \frac{\|g_k\|}{\|B_k\|}\right\}.$$

B) There exists a $\beta_2 > 0$ such that for all Δ_k and B_k

$$\text{pred}(\delta^{(k)}) \geq -\beta_2 \mu_1^{(k)} \Delta_k^2 .$$

C) If the matrix B_k is positive definite and $\|B_k^{-1} g_k\| \leq \Delta_k$, then

$$\delta^{(k)} = -B_k^{-1} g_k .$$

Condition A) ensures the global convergence of a trust region method to a point satisfying the first order necessary condition and a trust region method satisfying condition B) will converge to a point at which the second order necessary conditions are satisfied. When condition C) is satisfied, a convergent trust region method converges at a quadratic rate if $G(x^*)$ is positive definite at the limit point x^*.

Let Δ_k vary in the interval $(0, \infty)$, then the solution $\delta_k(\mu)$ of the trust region subproblem forms a continuous *optimal path* $\Gamma^{(k)}(\tau)$ in the space $\mathbf{R}^n$. The optimal path can be expressed as

$$\Gamma^{(k)}(\tau) = \Gamma_1^{(k)}(t(\tau)) + \Gamma_2^{(k)}(\theta(\tau))$$

where

$$\Gamma_1^{(k)}(t(\tau)) = -\sum_{i \in I} \frac{t(\tau)}{\mu_i^{(k)} t(\tau) + 1} v_i^{(k)} - t(\tau) \sum_{i \in N} v_i^{(k)},$$

$$\Gamma_2^{(k)}(\theta(\tau)) = \theta(\tau) u_1^{(k)},$$

$$t(\tau) = \begin{cases} \tau & \text{if } \tau < \frac{1}{\mu_\ell}, \\ \frac{1}{\mu_\ell} & \text{if } \tau \geq \frac{1}{\mu_\ell}, \end{cases}$$

$$\theta(\tau) = \begin{cases} 0 & \text{if } \mu_\ell = 0, \\ \max\left\{\tau - \frac{1}{\mu_\ell}, 0\right\} & \text{if } \mu_\ell > 0, \end{cases}$$

$$I = \left\{i:\ \mu_i^{(k)} \neq 0\right\}, \quad N = \left\{i:\ \mu_i^{(k)} = 0\right\}$$

$$v_i^{(k)} = {u_i^{(k)}}^\top g_k u_i^{(k)}, \quad i = 1, \ldots, n ,$$

and $\mu_1^{(k)} \leq \cdots \leq \mu_n^{(k)}$ are eigenvalues of B_k and $u_i^{(k)}$, $i = 1, \ldots, n$, are corresponding orthonormal eigenvectors. The optimal path has two properties. As a point x proceeds from $x^{(k)}$ along the path: i) the distance to $x^{(k)}$ is monotonically increasing, and ii) the value of $q_k(\delta)$ is monotonically decreasing. These properties guarantee that for any given Δ_k, the solution of the trust region subproblem is $\delta^{(k)} = \Gamma^{(k)}(\tau_k)$ where τ_k is uniquely determined from the equation $\| \Gamma^{(k)}(\tau) \| = \Delta_k$. The formulation of the path $\Gamma^{(k)}(\tau)$ needs the calculation of all eigenvalues and eigenvectors of the matrix B_k. This is time consuming and is unrealistic. J.P. Bulteau and J.-Ph. Vial [1] restrict the solution of the trust region subproblem in a two-dimensional subspace $S = \text{span}[a_1 a_2]$ and choose the solution of the problem

$$\begin{cases} \min & q_k(\delta) = f_k + g_k^\top \delta + \frac{1}{2} \delta^\top B_k \delta , \\ \text{s.t.} & \|\delta\| \leq \Delta_k,\ \delta \in S , \end{cases}$$

as an approximate solution of the subproblem. The vectors a_1 and a_2 are chosen in such a way that $A = [a_1 a_2]$, $A^\top A = I$. Then any $\delta \in S$ can be expressed as $\delta = Az$ for any $z \in \mathbf{R}^2$ and the restricted subproblem can be simplified as

$$\begin{cases} \min & q_k(z) = f_k + g_k^\top A z + \frac{1}{2} z^\top A^\top B_k A z , \\ \text{s.t.} & \|z\| \leq \Delta_k . \end{cases}$$

This is a trust region subproblem in the space $\mathbf{R}^2$ and can be easily solved using the optimal path method, since we only need calculate the whole eigensystem of a (2×2)-matrix $A^\top B_k A$ to form the optimal path $\Gamma_S^{(k)}(\tau)$. The subspace S is generally chosen in the following way:

$$S = \begin{cases} \text{span}[-g_k, -B_k^{-1} g_k] & \text{if } B_k \text{ P.D.}, \\ \text{span}[-g_k, u_1^{(k)}] & \text{if } B_k \text{ I.D.}, \end{cases}$$

where P.D. and I.D. denote positive definite and indefinite, respectively. In fact the path $\Gamma_S^{(k)}(\tau)$ can be regarded as a projection of the optimal path $\Gamma^{(k)}(\tau)$ in the subspace S and is an approximation of the path $\Gamma^{(k)}(\tau)$. Based on this idea, the recent efficient solution methods for trust region subproblems first form an approximate path $\Gamma_a^{(k)}(\tau)$ and then choose $\delta^{(k)} = \Gamma_a^{(k)}(\tau_k)$ with $\| \Gamma_a^{(k)}(\tau_k) \| = \Delta_k$. This greatly reduce the complication for the solution of the trust region subproblem, since for any given Δ_k the solution is obtained from the solution of the equation $\| \Gamma_a^{(k)}(\tau) \| = \Delta_k$ and the reformulation of the path for any reduced Δ_k is not required. When formulating an approximate path, it is required

to satisfy the two properties i) and ii) that the optimal path has.

Dogleg path methods are most effective and are first suggested by M.J.D. Powell and modified by Dennis and H.H.W. Mei [3] for positive definite matrices and extended to indefinite matrices by J.Z. Zhang and C.X. Xu [15]. Powell's method uses a single dogleg path in the form

$$\Gamma_{ps}^{(k)} = [0, \delta_{cp}^{(k)}, \delta_{np}^{(k)}]$$

to approximate the optimal path, where

$$\delta_{cp}^{(k)} = -\frac{g_k^\top g_k}{g_k^\top B_k g_k} g_k$$

is the minimizer of $q_k(\delta)$ in the steepest descent direction and

$$\delta_{np}^{(k)} = -B_k^{-1} g_k$$

is the global minimizer of $q_k(\delta)$. Then the solution $\delta^{(k)}$ of the problem

$$\min \left\{ q_k(\delta) \colon \ \|\delta\| \le \Delta_k,\ \delta \in \Gamma_{ps}^{(k)} \right\}$$

is taken as an approximate solution of the trust region subproblem. The solution $\delta^{(k)}$ can be obtained in the following way

$$\delta^{(k)} = \begin{cases} -B_k^{-1} g_k & \text{if } \left\| \delta_{np}^{(k)} \right\| \le \Delta_k \,, \\ -\frac{\Delta_k}{\|g_k\|} g_k & \text{if } \left\| \delta_{cp}^{(k)} \right\| \ge \Delta_k \,, \\ \delta_{cp}^{(k)} + t_k (\delta_{np}^{(k)} - \delta_{cp}^{(k)}) & \text{otherwise} \,, \end{cases}$$

where $t_k \in (0, 1)$ is determined from the equation $\| \delta_{cp}^{(k)} + t(\delta_{np}^{(k)} - \delta_{cp}^{(k)}) \| = \Delta_k$. Dennis and Mei modified the single dogleg path to form a double dogleg path

$$\Gamma_{dm}^{(k)} = [0, \delta_{cp}^{(k)}, \delta_{\eta}^{(k)}, \delta_{np}^{(k)}] \,,$$

where

$$\delta_{\eta}^{(k)} = \eta \delta_{np}^{(k)} \,, \quad \frac{\|g_k\|^4}{g_k^\top B_k g_k \cdot g_k^\top B_k^{-1} g_k} < \eta < 1 \,.$$

The solution $\delta^{(k)}$ in both the methods satisfies the general conditions A) and C). Hence both the methods are global convergent and convergence rate is quadratic if $G(x^*)$ is positive definite at limit point x^*.

Both the single and double dogleg path methods do work well when all the matrices B_k are positive definite. But they are unable to deal with the nonpositive definite case which occurs very often in practice. Zhang and Xu proposed indefinite dogleg paths for indefinite matrices B_k. One of these indefinite dogleg paths has the form

$$\Gamma_{ip}^{(k)} = [0, \delta_{\mu p}^{(k)}, \delta_{\mu}^{(k)}, d) \,,$$

where

$$\delta_{\mu p}^{(k)} = -\frac{g_k^\top g_k}{g_k^\top (B_k + \mu_k I) g_k} g_k \,,$$
$$\delta_{\mu}^{(k)} = -(B_k + \mu_k I)^{-1} g_k \,,$$

d is a *negative curvature* direction of the matrix B_k and μ_k is chosen such that $(B_k + \mu_k I)$ is positive definite. Since the optimal path $\Gamma^{(k)}(\tau)$ is infinite when B_k is indefinite, this is an infinite dogleg path. The solution $\delta^{(k)}$ obtained in the path has the following form

$$\delta^{(k)} = \begin{cases} -\frac{\Delta_k}{\|g_k\|} g_k & \text{if } \left\| \delta_{\mu p}^{(k)} \right\| \ge \Delta_k \,, \\ \delta_{\mu}^{(k)} + t_k d, \, t_k > 0 & \text{if } \left\| \delta_{\mu}^{(k)} \right\| \le \Delta_k \,, \\ \delta_{\mu p}^{(k)} + t_k (\delta_{\mu}^{(k)} - \delta_{\mu p}^{(k)}), & \\ t_k \in (0, 1) & \text{otherwise} \,, \end{cases}$$

where both t_k are determined from the equation $\|\delta^{(k)}(t)\| = \Delta_k$. When this indefinite dogleg path is combined with either the single or the double dogleg path, the resulting trust region method generates the solution $\delta^{(k)}$ satisfying all the three general conditions A)–C).

The negative curvature direction d can be effectively obtained from the Bunch–Parlett factorization

$$P B_k P^\top = L D L^\top$$

for symmetric matrices [2], where P is a permutation matrix, L a unit lower triangular matrix and D a block diagonal matrix with 1×1 and 2×2 diagonal blocks. Since matrices B_k and D have the same *inertia*, the negative curvature directions of B_k can be directly obtained from the eigenvectors of D corresponding to its negative eigenvalues. Let v_1 be the eigenvector corresponding to the smallest eigenvalue of D and $v = (v_1^\top L^\intercal P g_k) v_1$. Then

$$d = -\operatorname{sgn}(g_k^\top P^\top L^{-1} v) P^\top L^{-1} v$$

is a satisfactory negative curvature direction of B_k [15].

For large scale problems, methods with matrix factorizations generally are not suitable. T. Steihaug [14] proposed to use a *conjugate gradient method* for the system $B_k\delta = -g_k$ to find an approximate solution of the trust region subproblem. After δ_1, h_1 and d_1 are given, the iteration of the conjugate gradient method has the form:

$$\begin{aligned}
\delta_{j+1} &= \delta_j + \alpha_j d_j\,,\\
\alpha_j &= \frac{h_j^\top h_j}{d_j^\top B_k d_j}\,,\\
h_{j+1} &= h_j - \alpha_j B_k d_j\,,\\
d_{j+1} &= h_{j+1} + \beta_j d_j\,,\\
\beta_j &= \frac{h_{j+1}^\top h_{j+1}}{h_j^\top h_j}\,,\\
j &= 1,\ldots,n\,.
\end{aligned}$$

When this iteration is used to generate an approximate solution to the trust region subproblem, the solution is obtained in the following way. Assume that $d_j^\top B_k d_j > 0$ for $j = 1,\ldots,i-1$ and $\|\delta_i\| < \Delta_k$ and that h_i and d_i have been calculated. In case either $d_i^\top B_k d_i < 0$ or $d_i^\top B_k d_i > 0$ but $\|\delta_i + \alpha_i d_i\| \geq \Delta_k$, $\delta_i + t_i d_i$ is chosen as $\delta^{(k)}$ where the value t_i is determined from the equation $\|\delta_i + td_i\| = \Delta_k$. If $d_j^\top B_k d_j > 0$ for all $j = 1,\ldots,n$ and $\|\delta_{n+1}\| \leq \Delta_k$, then it follows from the properties of the conjugate gradient method that the matrix B_k is positive definite, $h_{n+1} = 0$ and $\delta_{n+1} = -B_k^{-1}g_k$ is the exact solution of the system $B_k\delta = -g_k$. Therefore, δ_{n+1} is the desired solution of trust region subproblem. The conjugate gradient method for the solution of the trust region subproblem has the form.

1) Give $\delta_1 = 0$, $h_1 = -g_k$, $d_1 = h_1$, $\eta_k \in (0,1)$, $j = 1$.
2) If $d_j^\top B_k d_j < 0$ then go to 6) below, else calculate $\alpha_j = h_j^\top h_j / d_j^\top B_k d_j$ and $\delta_{j+1} = \delta_j + \alpha_j d_j$.
3) If $\|\delta_{j+1}\| \geq \Delta_k$, go to 6) below.
4) Calculate $h_{j+1} = h_j - \alpha_j B_k d_j$. If $\|h_{j+1}\| \leq \eta_k\|g_k\|$, choose $\delta^{(k)} = \delta_{j+1}$.
5) $\beta_j = h_{j+1}^\top h_{j+1}/h_j^\top h_j$, $d_{j+1} = h_{j+1} + \beta_j d_j$, $j = j+1$, then go to 2).
6) Determine t_j such that $\|\delta_j + t_j d_j\| = \Delta_k$ and choose $\delta^{(k)} = \delta_j + t_j d_j$.

In fact, the conjugate gradient method for the approximate solution of the trust region subproblem is essentially a *multiple dogleg path* method. When the matrix B_k is positive definite, the multiple dogleg path is

$$\Gamma_{mp}^{(k)} = [0, \delta_1, \ldots, \delta_{n+1}]$$

and when the matrix B_k is indefinite, the multiple dogleg path is

$$\Gamma_{mi}^{(k)} = [0, \delta_1, \ldots, \delta_i, d_i)$$

where d_i is the first calculated negative curvature direction in the iteration process. Then the solution $\delta^{(k)}$ is obtained either in the path $\Gamma_{mp}^{(k)}$ or in the path $\Gamma_{mi}^{(k)}$.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- ABS Algorithms for Optimization
- Conjugate-gradient Methods
- Gauss–Newton Method: Least Squares, Relation to Newton's Method
- Generalized Total Least Squares
- Large Scale Trust Region Problems
- Least Squares Orthogonal Polynomials
- Least Squares Problems
- Local Attractors for Gradient-related Descent Iterations
- Nonlinear Least Squares: Newton-type Methods
- Nonlinear Least Squares Problems

References

1. Bulteau JP, Vial J-Ph (1985) A restricted trust region algorithm for unconstrained optimization. JOTA 47:413–434
2. Bunch JB, Parlett BN (1971) Direct methods for solving symmetric indefinite system of linear equations. SIAM J Numer Anal 8:639–655
3. Dennis JE Jr, Mei HHW (1979) Two new unconstrained optimization algorithms which use function and gradient values. JOTA 28:453–482
4. Dennis JE Jr, Schnabel RB (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs
5. Gay DM (1981) Computing optimal locally constrained step. SIAM J Sci Statist Comput 2:186–197
6. Goldfeld SM, Quandt RE, Trotter HF (1966) Maximisation by quadratic hill-climbing. Econometrica 34:541–551
7. Hebden MD (1973) An algorithm for minimization using exact second derivatives. Atomic Energy Res Establishment Harwell, Report TP 515

8. Levenberg K (1944) A method for the solution of certain nonlinear problems in least squares. Quart Appl Math 2:164–168
9. Marquardt DW (1963) An algorithm for least squares estimation of nonlinear parameters. SIAM J 11:111–115
10. Moré JJ (1978) The Levenberg–Marquardt algorithm: implementation and theory. In: Watson GA (ed) Numerical Analysis: Dundee. Lecture Notes Math. Springer, Berlin, pp 105–116
11. Moré JJ, Sorensen DC (1981) Computing a trust region step. SIAM J Sci Statist Comput 4:553–572
12. Powell MJD (1970) A hybrid method for nonlinear equations. In: Rabinowitz P (ed) Numerical methods for nonlinear algebraic equations. Gordon and Breach, New York, pp 87–114
13. Shultz GA, Schnabel RB, Byrd RH (1985) A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties. SIAM J Numer Anal 22:47–67
14. Steihaug T (1983) The conjugate gradient method and trust region in large scale optimization. SIAM J Numer Anal 20:626–637
15. Zhang JZ, Xu CX (1999) A class of indefinite dogleg methods for unconstrained minimization. SIAM J Optim 9:646–667

Nonlinear Systems of Equations: Application to the Enclosure of All Azeotropes

S. T. Harding, Christodoulos A. Floudas
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C30

Article Outline

Keywords
Nonlinear System of Equations
Condition 1: Phase Equilibrium
Condition 2: Equality of Phase Compositions
Solution Methodologies
Global Optimization Approach
Homotopy Continuation Approach
Heterogeneous Azeotropes
Reactive Azeotropes
See also
References

Keywords

Thermodynamics; Heterogeneous; Homogeneous; Azeotrope

An *azeotrope* occurs when a liquid mixture boils at a constant temperature, producing a vapor with the same composition as the liquid. This is a significant phenomenon in *distillation* processes. An azeotrope represents a barrier to distillation, beyond which further separation of the components in the mixture is not possible. Therefore, the problem of locating all azeotropes in a chemical mixture is more than simply a thermodynamic curiosity, it is important for a number of reasons. First, the task of locating *all* azeotropes in a chemical mixture is essential for the *synthesis of separation processes*. A separation process that is designed without complete knowledge about the azeotropes in the system is likely to be infeasible.

In addition, many thermodynamic models have been proposed which can predict the phase behavior of nonideal mixtures. Unfortunately, the accuracy of these models is not uniform over a wide range of mixtures. One useful way of testing the accuracy of a model for a given mixture is to compare the compositions of the azeotropes predicted by the model with those determined by experiment.

Despite the considerable interest in the area of predicting phase equilibria for chemical mixtures, relatively few methods for enumerating the azeotropes for a given system have been reported. The thermodynamic conditions for azeotropy constitute a nonlinear system of equations. This problem presents the additional challenge of finding all of the solutions to the nonlinear system of equations where the number of solutions is not known.

Nonlinear System of Equations

The most common type of azeotropes studied to date are called homogeneous azeotropes and occur when a single liquid phase is in equilibrium with the vapor phase. The thermodynamic conditions for homogeneous azeotropy are therefore gives,

1) equilibrium between the vapor phase and a single liquid phase, and
2) composition of the vapor phase is identical to the composition of the liquid phase.

Two other classifications of azeotropes, heterogeneous and reactive, will be discussed later in this article.

Condition 1: Phase Equilibrium

The equilibrium condition requires that the chemical potential of each component must be the same in the liquid and the vapor phases. If we consider a mixture of N components,

$$\mu_i^V = \mu_i^L , \quad \forall i \in N ,$$

where μ_i^V and μ_i^L represent the *chemical potential* of component i in the vapor and liquid phases. From the definition of the fugacity of component i in a mixture, $\hat{f}_i$,

$$\hat{f}_i^V = \hat{f}_i^L , \quad \forall i \in N ,$$

hence,

$$y_i \hat{\phi}_i^V P = x_i \gamma_i^L f_i^L , \quad \forall i \in N . \tag{1}$$

The symbol $\hat{\phi}_i^V$ represents the mixture *fugacity coefficient* of component i in the vapor phase. For the liquid phase, γ_i^L is the *activity coefficient*, and f_i^L is the fugacity of component i in the liquid phase. Rearranging (1) gives,

$$\frac{y_i}{x_i} = \frac{\gamma_i^L f_i^L}{\hat{\phi}_i^V P} , \quad \forall i \in N .$$

A common simplification is to assume that at low pressure the vapor phase can be modeled as an ideal gas, for which $\hat{\phi}_i^V = 1$. For the liquid phase the fugacity is equal to $f_i^L = \phi_i^{sat} P_i^{sat} (PF)_i$. But, for an ideal gas, $\phi_i^{sat} = 1$, and $(PF)_i = 1$. Therefore,

$$\frac{y_i}{x_i} = \frac{\gamma_i^L P_i^{\text{sat}}}{P} . \tag{2}$$

Condition 2: Equality of Phase Compositions

The azeotropy condition requires that the composition of the vapor phase is identical to the composition of the liquid phase.

$$y_i = x_i , \quad \forall i \in N . \tag{3}$$

Equations must also be added to require that the mole fractions in each phase sum to unity and have values between 0 and 1.

$$\begin{cases} \sum_{i \in N} y_i = \sum_{i \in N} x_i = 1 , \\ 0 \le y_i, x_i \le 1 , \qquad \forall i \in N . \end{cases} \tag{4}$$

Equations (2), (3), (4) constitute the nonlinear system of equations that are satisfied by a homogeneous azeotrope. Typical nonlinear equation solvers cannot be used robustly to find all of the solutions to this system of equations, since it is a nonlinear, constrained problem with multiple solutions. Many systems contain multiple azeotropes, each of which is a solution to the system. In addition, each pure component is a solution, giving at least N solutions.

Solution Methodologies

Most of the previous work reported in the literature has been limited to calculating homogeneous azeotropes using local nonlinear equation solvers. This means that the ability of these methods to predict azeotropes is dependent upon choosing good starting points for the solution technique. These methods cannot guarantee that all of the azeotropes have been located in a particular system. [1] calculated ternary homogeneous azeotropes using the Wilson model under isothermal conditions. [10] calculated homogeneous azeotropes of binary mixtures using an equation of state as the thermodynamic model. Their approach was to fix temperature and vary composition and volume until thermodynamic equilibrium conditions were satisfied. [12] also used an equation of state to calculate homogeneous azeotropes for binary mixtures.

[2] presented a search method for finding homogeneous and heterogeneous azeotropes which uses a *Levenberg–Marquardt algorithm* to find homogeneous azeotropes and then checks the stability of each solution with the tangent plane criterion described by [8]. A solution which is found to be unstable is then used as the starting point for a new search for an heterogeneous azeotrope. Again, this technique relies on local solution techniques and cannot guarantee that all azeotropes are located.

[4] proposed a method for locating all homogeneous azeotropes in multicomponent systems. This method uses a homotopy continuation technique for tracking branches of solutions to the nonlinear system of equations. They demonstrated that the technique performed robustly for systems containing up to five components using the Wilson activity coefficient equation. Recently, [3] have extended this method for heterogeneous azeotropes, and [9] have used it to predict

reactive azeotropes. [7] have formulated the problem of locating all homogeneous azeotropes as a global optimization problem in which each global minimum solution corresponds to an azeotrope. They applied this approach robustly using the Wilson, NRTL, UNIQUAC and UNIFAC activity coefficient equations for systems containing up to five components.

An excellent review on nonideal distillation, including a discussion on the computation of azeotropes has recently been published in [13].

Global Optimization Approach

In order to find all azeotropes, one must find all solutions to the system of nonlinear equations constituted by (2), (3), and (4). The method proposed in [6] reformulates the problem of enclosing all solutions of nonlinear systems of constrained equations into a global optimization problem in which the task is to enclose all global minimum solutions. In this approach, each nonlinear equality is replaced by two inequalities and a single slack variable is introduced. For the location of all homogeneous azeotropes, this corresponds to employing equations (2), (3), and (4) and reformulating them as the following global optimization problem:

$$\begin{cases} \min\limits_{\mathbf{x},T,s} & s \\ \text{s.t.} & \ln P - \ln P_i^{\text{sat}} - \ln \gamma_i - s \le 0, \quad \forall i, \\ & -\ln P + \ln P_i^{\text{sat}} + \ln \gamma_i - s \le 0, \quad \forall i, \\ & \sum\limits_{i \in N} x_i = 1, \\ & 0 < \mathbf{x} < 1, \\ & s \ge 0. \end{cases} \tag{5}$$

Problem (5) may have multiple global minima. Each global minimum of Problem (5) (where the solution $s^* = 0$) corresponds to an homogeneous azeotrope since when $s = 0$ the constraints (2), (3), and (4) are satisfied. Note that the first two sets of constraints of (5) correspond to the nonlinear equations (2) of the equilibrium constraint written as two inequalities. In addition, note that the nonlinear term $P_i^{sat}\gamma_i x_i$ appears as both a positive and a negative term. Thus, this term must be nonconvex in at least one of the two constraints. This means that if a local optimization approach is used to solve Problem (5), some or all of the global solutions may be missed.

For azeotropes in which less than N of the components participate (a k-ary azeotrope where $k \le N$), the case where $x_i = 0$ for one or more component must be accounted for. This can be done by multiplying the equilibrium constraints used in (5) by x_i. The general search for all k-ary homogeneous azeotropes is formulated as:

$$\begin{cases} \min\limits_{\mathbf{x},T,s} & s \\ \text{s.t.} & x_i(\ln P - \ln P_i^{\text{sat}} - \ln \gamma_i) - s \le 0, \\ & \quad \forall i \in N, \\ & x_i(-\ln P + \ln P_i^{sat} + \ln \gamma_i) - s \le 0, \\ & \quad \forall i \in N, \\ & \sum\limits_{i \in N} x_i = 1, \\ & 0 \le \mathbf{x} \le 1, \\ & s \ge 0. \end{cases} \tag{6}$$

0	Start with an initial guess for a solution (this does not affect the convergence of the algorithm).
1	Determine if the current solution satisfies the original nonlinear system of equation. IF it does, THEN store the solution, it corresponds to an azeotrope.
2	Is the current region smaller than the minimum size tolerance? IF it is, THEN fathom the region.
3	Partition the region into two subdomains.
4	Solve a lower bounding problem in each subdomain. IF the solution is greater than zero, THEN fathom the region. ELSE the solution is stored in the list of lower bounding solutions.
5	Choose among the active regions the one with the minimum lower bounding solution and update the current solution point. Return to Step 1. IF there are no regions remaining, THEN terminate.

Global optimization procedure

In [7], the authors have applied a global optimization approach to enclose all of the solutions to Problems (5) and (6). In this approach, each nonconvex term

is identified, and replaced by a convex underestimator. The solution of the convexified problem is then a lower bound on the solution to the original nonconvex problem. A *branch and bound* procedure is used to improve the lower bound. Regions that cannot contain an azeotrope have a positive lower bound and can be fathomed. Regions whose lower bound is less than or equal to zero are refined further until all of the azeotropes have been located. [7] have analyzed the Wilson, NRTL, UNIQUAC, and UNIFAC equations and have developed tight convex underestimators for all of the nonconvex terms for these thermodynamic functions.

Example 1 In this quaternary system (methanol, benzene, i-propanol, n-propanol) three binary azeotropes have been reported in the literature, as shown in Table 1. No experimental data was found for the ternary and quaternary systems.

Using the global optimization approach of [7], both the Wilson and NRTL equations predicted only the three reported azeotropes. The results for the Wilson equation are very close to the reported compositions and temperatures of the azeotropes. The results for the NRTL equation are also close to the reported values, with the exception of the Methanol-Benzene azeotrope.

Nonlinear Systems of Equations: Application to the Enclosure of All Azeotropes, Table 1
Solution for methanol (1) – benzene (2) – i-propanol (3) – n-propanol (4)

Azeo	x_1	x_2	x_3	x_4	T(deg C)
experimental data from [5]					
$1-2$	0.605	0.395	–	–	58.08
$1-3$	no azeotrope				
$1-4$	no azeotrope				
$2-3$	–	0.600	0.400	–	71.80
$2-4$	–	0.791	–	0.209	77.10
$3-4$	no azeotrope				
Wilson equation					
$1-2$	0.624	0.376	–	–	58.129
$2-3$	–	0.586	0.414	–	71.951
$2-4$	–	0.780	–	0.220	76.946
NRTL equation					
$1-2$	0.063	0.937	–	–	80.166
$2-3$	–	0.588	0.412	–	71.832
$2-4$	–	0.776	–	0.224	77.131

Homotopy Continuation Approach

The method proposed in [4] is based on tracking branches of solutions to nonlinear systems of equations that are perturbed from the original system. The key idea is to start with an equilibrium surface, on which all of the solutions are known a priori. The postulation is that every solution is connected to one of the branches of the initial surface. This surface is then gradually deformed, and the solution branches are tracked, ending with the actual nonideal equilibrium surface.

If the original equilibrium condition, (2), is rearranged so that the vapor mole fraction, y_i, is represented as a function of x_i, we obtain an equation of the form

$$y_i = \frac{\gamma_i^L P_i^{\text{sat}}}{P} x_i \,.$$

The ideal system can be represented by *Raoult's law*:

$$y_i^{\text{id}} = \frac{P_i^{\text{sat}}}{P} x_i \,.$$

One of the most convenient ways to represent the gradual deformation of the equilibrium surface is to use the linear convex combination of the ideal and nonideal equilibrium surfaces, where the homotopy parameter, t, determines the degree of deformation.

$$\widetilde{y}_i = \left[(1-t) + t\gamma_i^L\right] \frac{P_i^{\text{sat}}}{P} x_i \,.$$

The problem now is to find the roots of

$$\mathbf{h}(\mathbf{x}, t) = 0 \,, \tag{7}$$

where

$$\mathbf{h}(\mathbf{x}, t) = \widetilde{\mathbf{y}} - \mathbf{x} \,.$$

In [4], the authors use a *homotopy continuation* method to find the roots of (7). At $t = 0$, the pure components are used at the initial solutions to the ideal equilibrium equation. The homotopy parameter is then increased by a small amount and the solution branches are tracked. At each step, the determinant of $\mathbf{h}_\mathbf{x}$ is calculated, where

$$\mathbf{h}_\mathbf{x} = \left[\frac{\partial h_i}{\partial x_j}\right]$$
$$i = 1, \ldots, N-1, \quad j = 1, \ldots, N-1 \,.$$

Nonlinear Systems of Equations: Application to the Enclosure of All Azeotropes, Table 2
Solution for acetone (1) – chloroform (2) – methanol (3)

Azeo	x_1	x_2	x_3	T (deg C)
experimental data from [5]				
$1-2$	0.360	0.640	—	64.50
$1-3$	0.800	—	0.200	55.70
$2-3$	—	0.653	0.347	53.35
$1-2-3$	0.318	0.241	0.441	57.67
Wilson equation				
$1-2$	0.3437	0.6563	—	65.47
$1-3$	0.7944	—	0.2056	55.34
$2-3$	—	0.6481	0.3519	53.91
$1-2-3$	0.3234	0.2236	0.4530	57.58

If $\det[\mathbf{h}_\mathbf{x}] = 0$, then a bifurcation may occur, and an additional solution branch is started, which corresponds to an azeotrope.

[4] applied this method to a range of well-understood systems using the Wilson activity coefficient equation. Even though the homotopy continuation approach does not offer theoretical guarantee of locating all azeotropes, [4] observed that the method predicted all of the azeotropes that have been verified experimentally for these systems.

Example 2 [4] examined the ternary system of acetone, chloroform, and methanol They found that their homotopy continuation approach predicted all three binary azeotropes that are reported in the literature, using the Wilson equation. Their results are shown in Table 2.

Heterogeneous Azeotropes

Azeotropy is not limited to systems with a single liquid phase. In the more general case, where multiple liquid phases exist in equilibrium, a liquid mixture that boils at a constant temperature is called a heterogeneous azeotrope. Heterogeneous azeotropes can be of two different types. Type I heterogeneous azeotropes occur when the overall liquid composition is identical to the vapor composition. Conversely, Type II heterogeneous azeotropes occur when the overall liquid composition is not equal to the vapor composition. Type II heterogeneous azeotropes are possible theoretically, but have never been verified experimentally.

The phase equilibrium condition for a heterogeneous azeotrope in a system with M liquid phases is written:

$$\mu_i^V = \mu_i^{L_j}, \quad \forall i \in N, \ \forall j \in M .$$

When the definitions of the chemical potentials are applied and simplifications are made, the expression becomes,

$$\frac{y_i}{x_i^j} = \frac{\gamma_i^{L_j} P_i^{\text{sat}}}{P} .$$

In Type I heterogeneous azeotropes, the overall composition of the liquid is equal to the composition of the vapor,

$$y_i = \sum_{j=1}^{M} \psi^j x_i^j , \quad \forall i \in N ,$$

where ψ^j is the mole fraction of liquid phase j in the liquid mixture. The additional constraints are:

$$\sum_{i=1}^{N} y_i = 1 ,$$
$$\sum_{i=1}^{N} x_i^j = 1, \quad \forall j \in M ,$$
$$\sum_{j=1}^{M} \psi^j = 1 ,$$

An extension of the homotopy continuation method for homogeneous azeotropes of [4] was developed by [3] in order to examine the problem of finding all Type I heterogeneous azeotropes.

Reactive Azeotropes

Azeotropic behavior can also occur in reacting mixtures, [11]. The authors derived necessary and sufficient conditions for reactive azeotropes. These are:

$$Y_i - X_i = 0 , \quad i = 1, \ldots, N-1 ,$$

where

$$X_i = \frac{\nu_k x_i - \nu_i x_k}{\nu_k - \nu_T x_k} ,$$
$$Y_i = \frac{\nu_k y_i - \nu_i y_k}{\nu_k - \nu_T y_k} ,$$

where ν_i is the vector of reaction stoichiometric coefficients for component i, the index k refers to the reference component, and ν_T is the vector of the sum of the stoichiometric coefficients for each reaction.

In addition to this requirement, the system must also be in phase equilibrium, and in chemical equilibrium. The chemical equilibrium expression for each reaction r is written,

$$\left(\frac{K_{\text{eq}}^r}{K_{\text{eq}}^r+1}\right)\prod_{i\in\text{react}}(x_i\gamma_i)^{|\nu_{\text{react}}^r|} - \left(\frac{1}{K_{\text{eq}}^r+1}\right)\prod_{i\in\text{prod}}(x_i\gamma_i)^{|\nu_{\text{prod}}^r|} = 0\,,$$

where K_{eq}^r is the equilibrium coefficient for reaction r.

[9] have applied an arc-length continuation approach to search for all reactive azeotropes at fixed K_{eq} for systems with a single chemical reaction.

See also

- ► Contraction-Mapping
- ► Global Optimization Methods for Systems of Nonlinear Equations
- ► Interval Analysis: Systems of Nonlinear Equations
- ► Nonlinear Least Squares: Newton-type Methods

References

1. Aristovich VY, Stepanova EI (1970) Determination of the existence and composition of multicomponent azeotropes by calculation from data for binary systems. Zh Prikl Khim (Leningrad) 43:2192–2200
2. Chapman RG, Goodwin SP (1993) A general algorithm for the calculation of azeotropes in fluid mixtures. Fluid Phase Equilib 85:55–69
3. Eckert E, Kubicek M (1997) Computing heterogeneous azeotropes in multicomponent mixtures. Comput Chem Eng 21(3):347–350
4. Fidkowski ZT, Malone MF, Doherty MF (1993) Computing azeotropes in multicomponent mixtures. Comput Chem Eng 17(12):1141–1155
5. Gmehling J, Menke J, Krafczyk J, Fischer K (1994) Azeotropic data. Wiley-VCH, Weinheim
6. Harding ST, Floudas CA (1997) Global optimization in multiproduct and multipurpose batch design under uncertainty. Industr Eng Chem Res 36:1644–1664
7. Harding ST, Maranas CD, McDonald CM, Floudas CA (1997) Locating all azeotropes in homogeneous azeotropic systems. Industr Eng Chem Res 36:160–178
8. Michelsen ML (1981) The isothermal flash problem: Part I. Stability. Fluid Phase Equilib 9:1–19
9. Okasinski MJ, Doherty MF (1997) Thermodynamic behavior of reactive azeotropes. AIChE J 43(9):2227–2238
10. Teja AS, Rowlinson JS (1973) The prediction of the thermodynamic properties of fluids and fluid mixtures - IV. Critical and azeotropic states. Chem Eng Sci 28:529–538
11. Ung S, Doherty MF (1995) Necessary and sufficient conditions for reactive azeotropes in multireaction mixtures. AIChE J 41(11):2383–2392
12. Wang S, Whiting WB (1986) New algorithm for calculation of azeotropes from equations of state. Industr Eng Chem Process Des Developm 25:547–551
13. Widagdo S, Seider WD (1996) Azeotropic distillation. AIChE J 42:96–130

Nonlocal Sensitivity Analysis with Automatic Differentiation

LEIGH TESFATSION
Iowa State University, Ames, USA

MSC2000: 34-XX, 49-XX, 65-XX, 68-XX, 90-XX

Article Outline

Keywords
Basic Problem Formulation
Fully Automated Implementation
Incorporation of Automatic Differentiation
Automatic Initialization
via Adaptive Homotopy Continuation
See also
References

Keywords

Nonlocal sensitivity analysis; Nasa program; Automatic differentiation; Feed algorithm; Adaptive homotopy; Adaptive computational method

Basic Problem Formulation

Sensitivity analysis problems typically reduce to determining the response of a vector $x^* = (x_1^*, \ldots, x_n^*)$ to changes in a scalar α^*, where x^* and α^* are required to satisfy an n-dimensional system of nonlinear equations of the form

$$0 = \psi(x,\alpha) \equiv (\psi^1(x,\alpha),\ldots,\psi^n(x,\alpha))^\top\,. \qquad (1)$$

This problem formulation, with a scalar parameter α, is more general than it might first appear. For example, suppose an analyst wishes to investigate the surface of function values $x = f(z)$ taken on by some function f: $\mathbf{R}^m \to \mathbf{R}^n$ as z ranges over a specified region Z in $\mathbf{R}^m$. One approach is to consider a suitably smooth curve s: $[0, 1] \to Z$ which roughly fills this region, of the form $z = s(\alpha)$, and to define a new function of the form $\psi(x, \alpha) \equiv x - f(s(\alpha))$. Solving the system of equations $\psi(x, \alpha) = 0$ for x as a function of α as α ranges from 0 to 1 then yields a curve of points $x(\alpha)$ on the function surface which gives some idea of the shape of this surface over the region Z.

Assuming ψ: $\mathbf{R}^{n+1} \to \mathbf{R}^n$ is twice continuously differentiable and has a nonsingular Jacobian matrix $\psi_x(x^*, \alpha^*)$, the implicit function theorem guarantees the existence of a continuously differentiable function $x(\alpha)$ taking some neighborhood $N(\alpha^*)$ of α^* into $\mathbf{R}^n$ such that

$$0 = \psi(x(\alpha), \alpha)\,, \quad \alpha \in N(\alpha^*)\,, \tag{2}$$

with $x(\alpha^*) = x^*$. From (2) one obtains the fundamental equation for sensitivity analysis,

$$\frac{dx(\alpha)}{d\alpha} = -\psi_x(x(\alpha), \alpha)^{-1}\psi_\alpha(x(\alpha), \alpha)\,, \quad \alpha \in N(\alpha^*)\,. \tag{3}$$

As it stands, (3) is an analytically incomplete system of ordinary differential equations. That is, a closed form representation for the Jacobian inverse $J(\alpha)^{-1} \equiv \psi_x(x(\alpha), \alpha)^{-1}$ as a function of α is often not obtainable for $n \geq 3$. Thus, the integration of (3) from initial conditions would typically require the supplementary algebraic determination of the Jacobian inverse $J(\alpha)^{-1}$ at each step in the integration process.

Why not simply incorporate a linear equation solver to accomplish the needed matrix inversions? Two reasons can be given. First, the Jacobian matrix might have one or more eigenvalues which are small in absolute value. Consequently, as can be seen using a singular value decomposition, the inverse matrix can be highly ill-conditioned in the sense that its elements have large absolute values and take on both positive and negative values. In this case, small roundoff and truncation errors can cause large errors in the resulting numerically determined component values of the sensitivity vector $dx(\alpha)/d\alpha$. Second, there exists an alternative approach [14] that has proven its reliability and efficiency in numerous contexts over the past twenty years: replace the algebraic operation of matrix inversion by an initial value problem highly suited for modern digital computers.

The latter approach is taken in [10]. The differential system (3) is extended by the incorporation of ordinary differential equations for the Jacobian inverse. More precisely, letting $A(\alpha)$ and $\delta(\alpha)$ denote the adjoint and the determinant of the Jacobian matrix $J(\alpha)$, and recalling that the inverse of any nonsingular matrix can be represented as the ratio of its adjoint to its determinant, the following differential system is validated for $x(\alpha)$, $A(\alpha)$, and $\delta(\alpha)$:

$$\frac{\mathrm{d}x(\alpha)}{\mathrm{d}\alpha} = -A(\alpha)\frac{\psi_\alpha(x(\alpha), \alpha)}{\delta(\alpha)}\,; \tag{4}$$

$$\frac{\mathrm{d}A(\alpha)}{\mathrm{d}\alpha} = \frac{A(\alpha)\,\mathrm{Trace}(A(\alpha)B(\alpha)) - A(\alpha)B(\alpha)A(\alpha)}{\delta(\alpha)}\,; \tag{5}$$

$$\frac{\mathrm{d}\delta(\alpha)}{\mathrm{d}\alpha} = \mathrm{Trace}(A(\alpha)B(\alpha))\,. \tag{6}$$

The ijth component of the matrix $B(\alpha) \equiv dJ(\alpha)/d\alpha$ appearing in equations (5) and (6) is

$$\sum_{k=1}^{n}\left(\psi^i_{jk}(x(\alpha), \alpha)\frac{\mathrm{d}x_k(\alpha)}{\mathrm{d}\alpha}\right) + \psi^i_{j,n+1}(x(\alpha), \alpha)\,, \tag{7}$$

where ψ^i_{jk} denotes the second partial of ψ^i with respect to x_j and x_k, and $\psi^i_{j,n+1}$ denotes the second partial of ψ^i with respect to x_j and α. Given (4), note that each of the components (7) is expressible as a known function of $x(\alpha)$, $A(\alpha)$, $\delta(\alpha)$, and α. Initial conditions for equations (4)–(6) must be provided at a parameter point α^* by specifying values for $x(\alpha^*)$, $A(\alpha^*)$, and $\delta(\alpha^*)$ satisfying $0 = \psi(x(\alpha^*), \alpha^*)$, $A(\alpha^*) = \mathrm{Adj}(J(\alpha^*))$, and $\delta(\alpha^*) = \mathrm{Det}(J(\alpha^*)) \neq 0$.

In summary, the system of equations (4)–(6) provides an analytically complete system of ordinary differential equations for the *nonlocal sensitivity analysis* of the original system of interest, $0 = \psi(x, \alpha)$. That is, it permits the *tracking* of the solution vector $x(\alpha)$ and the sensitivity vector $dx(\alpha)/d\alpha$, together with the adjoint $A(\alpha)$ and the determinant $\delta(\alpha)$ of the Jacobian matrix $J(\alpha)$, over any α-interval $[\alpha^*, \alpha^{**}]$ where the determinant remains nonzero.

Fully Automated Implementation

The complete differential system (4)–(6) was initially implemented in [10] by means of a Fortran program incorporating a fourth-order Adams–Moulton integration method with a Runge–Kutta start and hand-coded partial derivatives. High numerical accuracy was obtained in illustrative applications, even near critical points α where the determinant $\delta(\alpha)$ became zero. Nevertheless, hand-coding of partial derivatives was clearly an undesirable feature of the program. The partial derivative expressions in (7) involve the second order partial derivatives of $\psi(\cdot)$; and $\psi(\cdot)$ in turn could involve the partial derivatives of some still more basic function, such as the criterion function for an optimization problem. This is indeed the typical case for economic problems (e. g., the profit maximization problem handled in [10]), since such problems invariably incorporate the decision-making processes of various types of economic agents.

In consequence, a more fully *automated Fortran program for nonlocal sensitivity analysis* was eventually developed in [11]. This program is referred to as *Nasa* (an acronym for Nonlocal Automated Sensitivity Analysis); it is available for downloading as freeware from the Web site http://www.econ.iastate.edu/tesfatsi/. Nasa incorporates a fairly substantial library for the forward-mode automatic evaluation of partial derivatives through order three [13] as well as an adaptive homotopy method [12] for automatically obtaining all required initial conditions. The following sections briefly describe these features. A recent example of how Nasa has been applied to an applied general equilibrium problem in economics is detailed in [2].

Incorporation of Automatic Differentiation

Four basic approaches (see Jerrell [8] for an interesting comparative discussion of these four alternative approaches) can be used to obtain computer-generated numerical values for derivatives: hand-coding; numerical differentiation; symbolic differentiation; and automatic derivative evaluation, or *automatic differentiation* for short. Recently, *computational differentiation* has come to be the preferred term for automatic differentiation; see [1]. To avoid confusion, the more traditional term is used here. Numerical differentiation methods substitute discrete approximate forms for derivative expressions. For example, finite difference methods involve the approximation of derivatives by ratios of discrete increments; e. g., $f'(t) \approx [f(t + h) - f(t)]/h$ for some suitably small h. Symbolic differentiation methods generate exact symbolic expressions for derivatives that can be manipulated algebraically as well as evaluated numerically. In contrast, automatic differentiation methods do not generate explicit derivative expressions, either approximate or symbolic. Rather, these methods focus on the generation of derivative evaluations by breaking down the evaluation of a derivative at a given point into a sequence of simpler evaluations for functions of at most one or two variables. These evaluations are exact up to roundoff and truncation error.

For the nonlocal sensitivity analysis problem outlined above, the primary requirement is for partial derivative evaluations through order three to be obtained in a reliable and efficient manner. The use of numerical differentiation methods such as finite difference introduces systematic approximation errors into applications that can be reduced but not eliminated entirely due to the risk of catastrophic floating point error. Symbolic differentiation software packages such as Macsyma, Mathematica, and Maple produce analytical expressions for derivatives but are notorious for 'expression swell', that is, for the great many lines of code they produce for the derivative expressions of even relatively simple functional forms despite repeated use of reduction routines; see [5] for explicit examples (note that automatic differentiation has now been introduced into Maple, see [6]). Thus, an automatic derivative evaluation routine would seem to be the preferred alternative for the application at hand.

Automatic differentiation appears to have been independently developed by R.E. Moore [16] and R. Wengert [20]. The key idea of Moore and Wengert was to decompose the evaluation of complicated functions of many variables into a sequence of simpler evaluations of special functions of one or two variables, referred to below as a 'function list'. Total differentials of the special functions could be automatically evaluated along with the special function values, and partial derivatives could then be recovered from the total differentials by solving certain associated sets of linear algebraic equations.

Nonlocal Sensitivity Analysis with Automatic Differentiation, Table 1
Illustrative application of the Feed algorithm

Function List	$\partial/\partial x$	$\partial/\partial y$	$\partial^2/\partial x^2$	$\partial^3/\partial x^3$
$a = x$	1	0	0	0
$b = y$	0	1	0	0
$c = ab$	$a_x b + a b_x$	$a_y b + a b_y$	$a_{xx} b + 2a_x b_x + a b_{xx}$	$a_{xxx} b + 3a_{xx} b_x + 3a_x b_{xx} + a b_{xxx}$
$d = \log(c)$	$c^{-1} c_x$	$c^{-1} c_y$	$-c^{-2} c_x^2 + c^{-1} c_{xx}$	$2c^{-3} c_x^3 - 3c^{-2} c_x c_{xx} + c^{-1} c_{xxx}$
$z = a + d$	$a_x + d_x$	$a_y + d_y$	$a_{xx} + d_{xx}$	$a_{xxx} + d_{xxx}$

As detailed in [1] and [4], great strides have been made over the past thirty years in developing fast and reliable automatic differentiation algorithms. The Nasa program incorporates one such algorithm, originally developed in [13], that is now referred to as *Feed* (an acronym for Fast Efficient Evaluation of Derivatives). A detailed discussion of the use of this automatic differentiation algorithm for both optimization and sensitivity analysis can be found in [9]. Total differentials are replaced by derivative arrays in order to avoid repeated function evaluations and the need to recover partial derivatives from total differentials for each successively higher-order level of differentiation.

As a simple illustration of Feed, consider the function F: $\mathrm{R}^2_{++} \to \mathrm{R}$ defined by

$$z = F(x, y) \equiv x + \log(xy)\,. \tag{8}$$

Suppose one wishes to evaluate the function value z and the partial derivatives z_x, z_y, z_{xx} and z_{xxx} at a given domain point (x, y). Consider Table 1.

The first column of Table 1 constitutes the function list for the function (8); it sequentially evaluates the function value $z = x + \log(xy)$ at the given domain point (x, y). The remaining entries in each row give the indicated derivative evaluations of the first entry in the row, using only algebraic operations. The first two rows initialize the algorithm, one row being required for each independent variable. The only input required for the first two rows is the domain point (x, y). Each subsequent row outputs a one-dimensional array of the form $(p, p_x, p_y, p_{xx}, p_{xxx})$, using the arrays obtained from previous row calculations as inputs. The final row yields the desired evaluations $(z, z_x, z_y, z_{xx}, z_{xxx})$. Note that the limitation to this collection of partial derivative evaluations is for expositional simplicity only. The evaluation of any additional desired partial derivative of z, say z_{xyy} or z_{xxxy}, can be obtained in a similar manner by suitably augmenting Table 1 with an additional column of algebraic operations.

The elements in each of the rows in Table 1 can be numerically evaluated by means of sequential calls to Feed calculus subroutines. These evaluations are exact up to roundoff and truncation error. For expositional simplicity, Table 1 only depicts evaluations for partial derivatives through order three. However, Feed calculus subroutines can in principle be constructed to evaluate the function value and the distinct partial derivatives through order k of any real-valued multivariable function that can be sequentially evaluated in a finite number of steps by means of the two-variable functions

$$\begin{aligned} &w = u + v, \quad w = u - v, \quad w = uv\,, \\ &w = \frac{u}{v}, \quad w = u^v \end{aligned} \tag{9}$$

and arbitrary nonlinear one-variable kth-order differentiable functions such as

$$\begin{aligned} &\cos(u), \quad \sin(u), \quad \exp(u), \quad c^u, \\ &\log(u), \quad au^b + c, \end{aligned} \tag{10}$$

for arbitrary constants a, b, and c. Systematic rules for constructing general kth-order calculus subroutines for special functions such as (10) are derived in [13]. References to other work focusing on recurrence relations for the derivatives of special functions such as (10) can be found, for example, in [15]. A detailed discussion of the library of Feed calculus subroutines currently incorporated into Nasa is given in [11].

The Feed algorithm thus envisions the successive transformation of arrays of partial derivatives through any specified order k into similarly-configured arrays as one forward sweep is taken through the function list for a specified kth-order differentiable function. A similar approach is proposed in [15] and [18, p. 280]. In

contrast, the partial derivative evaluation methods proposed in [17, Chapter VI, pp. 91–111] and [21] have a tree structure; that is, gradient operations are used to generate evaluations for each successively higher-order collection of partial derivatives using the results of previous gradient operations as inputs. Another approach that has attracted a great deal of interest is reverse-mode differentiation; see [3] and [7].

Automatic Initialization via Adaptive Homotopy Continuation

The initial conditions needed to integrate the complete differential system (4)–(6) from a given initial parameter point α^* consist of a solution vector $x(\alpha^*)$ together with evaluations for the adjoint $A(\alpha^*)$ and determinant $\delta(\alpha^*)$ of the Jacobian matrix $\psi_x(x(\alpha^*), \alpha^*)$. For many nonlinear problems, finding an initial solution vector is a difficult matter in and of itself.

Nasa incorporates an adaptive homotopy method [12] for automating these needed initializations. A standard (linear) homotopy method applied to the problem of finding a solution x^* for a system of equations $0 = F(x)$ proceeds by introducing a homotopy of the form

$$0 = tF(x) + [1 - t][x - c] \tag{11}$$

and solving for x as a function of t as t varies from 0 to 1 along the real line, where c represents any initial guess for the solution vector x^*. In contrast, an *adaptive homotopy* is a homotopy for which the usual continuation parameter t varying from 0 to 1 on the real line is replaced by an adaptive continuation 'agent' that makes its way by trial and error from $0 + 0i$ to $1 + 0i$ in the complex plane in accordance with certain stated objectives.

Specifically, the continuation agent designed in [12] adaptively selects a path of β values from $0 + 0i$ to $1 + 0i$ in the complex plane for the homotopy

$$0 = [F(x) - F(c)] + \beta F(c)\,, \tag{12}$$

where c again represents any initial guess for the solution vector x^*. The path for β is selected in accordance with the following multiple objective optimization problem: Reach the point $1 + 0i$ starting from the point $0 + 0i$ by taking as few steps as possible along a spider-web (spoke/hub) grid centered at $1 + 0i$ in the complex plane, but do so in a way that avoids regions where the Jacobian matrix becomes ill-conditioned.

The adaptive homotopy method introduced in [12] and incorporated into Nasa is thus an example of what might more generally be called an *adaptive computational method*, i. e., a computational method that embodies the following principle important for applied researchers: Let the computational algorithm adapt to the physical problem at hand instead of requiring users to reformulate their physical problems to conform to algorithmic requirements. For sufficiently smooth functions $F(\cdot)$, a properly constructed homotopy (e. g., a probability one homotopy as formulated in [19]) is theoretically guaranteed to have no singular points along the real continuation path from 0 to 1 for almost all initial starting points c. However, successful implementation of such homotopy methods can require a mathematically sophisticated reformulation of the user's original problem.

The homotopy (12) is solved for x as a function of β as β varies from $0 + 0i$ to $1 + 0i$ in the complex plane by making use of a complete system of ordinary differential equations analogous to the system set out in the basic problem formulation above. At each β point one obtains a solution vector $x^*(\beta)$ together with evaluations $A^*(\beta)$ and $\delta^*(\beta)$ for the adjoint and determinant of the homotopy Jacobian matrix $J^*(\beta) = F_x(x^*(\beta))$. Note that the homotopy Jacobian matrix coincides with the Jacobian matrix for the original function of interest $F(\cdot)$, implying that singularities are not artificially induced into the problem by the homotopy method per se. In principle, the solution vector $x^*(1 + 0i)$ obtained for (12) at $\beta = 1 + 0i$ yields a solution vector for the original system of interest, $0 = F(x)$. In particular, letting $F(x) \equiv \psi(x, \alpha^*)$, one obtains complete initial conditions for the original problem of interest, the nonlocal sensitivity analysis of the system $0 = \psi(x, \alpha)$ over an interval of α values starting at α^*.

See also

- ► Automatic Differentiation: Calculation of the Hessian
- ► Automatic Differentiation: Calculation of Newton Steps
- ► Automatic Differentiation: Geometry of Satellites and Tracking Stations

- ▶ Automatic Differentiation: Introduction, History and Rounding Error Estimation
- ▶ Automatic Differentiation: Parallel Computation
- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Point and Interval Taylor Operators
- ▶ Automatic Differentiation: Root Problem and Branch Problem
- ▶ Parametric Global Optimization: Sensitivity
- ▶ Sensitivity Analysis of Complementarity Problems
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Sensitivity and Stability in NLP
- ▶ Sensitivity and Stability in NLP: Approximation
- ▶ Sensitivity and Stability in NLP: Continuity and Differential Stability

References

1. Berz M, Bischof Ch, Corliss G, Griewank G (1996) Computational differentiation techniques, applications, and tools. SIAM, Philadelphia
2. Dakhlia S (1999) Testing for a unique equilibrium in applied general equilibrium models. J Econom Dynam Control 23:1281–1297
3. Griewank A (1989) On automatic differentiation. In: Iri M, Tanabe K (eds), Mathematical Programming: Recent Developments and Applications. Kluwer, Dordrecht, pp 83–108
4. Griewank A, Corliss G (1991) Automatic differentiation of algorithms: Theory, implementation, and application. SIAM, Philadelphia
5. Hayes K, Hirschberg J, Slottje D (1987) Computer algebra: Symbolic and algebraic computation in economic/econometric applications. In: Adv Econometrics, vol 6. JAI Press, pp 51–89
6. Heck A (1993) Introduction to Maple. Springer, Berlin
7. van Iwaarden R (1993) Automatic differentiation applied to unconstrained nonlinear optimization with result verification. Interval Comput 4:30–41
8. Jerrell ME (1997) Automatic differentiation and interval arithmetic for estimation of disequilibrium models. Comput Economics 10:295–316
9. Kagiwada H, Kalaba R, Rasakhoo N, Spingarn K (1985) Numerical derivatives and nonlinear analysis. Plenum, New York
10. Kalaba R, Tesfatsion L (1981) Complete comparative static differential equations. Nonlinear Anal Th Methods Appl 5:821–833
11. Kalaba R, Tesfatsion L (1990) Nonlocal automated sensitivity analysis. Comput Math Appl 20:53–65
12. Kalaba R, Tesfatsion L (1991) Solving nonlinear equations by adaptive homotopy continuation. Appl Math Comput 41:99–115
13. Kalaba R, Tesfatsion L, Wang J-L (1983) A finite algorithm for the exact evaluation of higher-order partial derivatives of functions of many variables. J Math Anal Appl 92:552–563
14. Kalaba R, Zagustin E, Holbrow W, Huss R (1977) A modification of Davidenko's method for nonlinear systems. Comput Math Appl 3:315–319
15. Kedem G (1980) Automatic differentiation of computer programs. ACM Trans Math Softw 6:150–165
16. Moore RE (1962) Interval arithmetic and automatic error analysis in digital computing. PhD Thesis, Dept. Computer Sci., Stanford Univ
17. Rall L (1981) Automatic differentiation: Techniques and applications. Springer, Berlin
18. Rall L (1986) The arithmetic of differentiation. Math Magazine 59:275–282
19. Watson LT, Sosonkina M, Melville RC, Morgan AP, Walker HF (1997) HOMPACK90: A suite of FORTRAN 90 codes for globally convergent homotopy algorithms. ACM Trans Math Softw 23:514–549
20. Wengert R (1964) A simple automatic derivative evaluation program. Comm ACM 7:463–464
21. Wexler A (1988) An algorithm for exact evaluation of multivariate functions and their derivatives to any order. Comput Statist Data Anal 6:1–6

Nonoriented Multicommodity Flow Problems

PIERRE CHARDAIRE[1], ABDEL LISSER[2]

[1] University East Anglia, Norwich, UK

[2] Recherche et Developpement, France Telecom, Issy les Moulineaux, France

MSC2000: 90B10, 90C05, 90C06, 90C35

Article Outline

Keywords
See also
References

Keywords

Network; Multicommodity flow; Decomposition algorithms

The routing of traffic requirements is one of the important problems that have to be solved when designing a telecommunication network. Another important

problem is the computation of standby capacities that must be set up to enable the rerouting of the traffic requirements affected by any network failure of some given types. Both problems can be formulated as nonoriented multicommodity network flow models sometimes also called *undirected multicommodity network flow models*. The routing problem captures the main aspects of such models and is therefore considered in more details hereafter. The reader interested in the computation of standby capacities is referred to [11,12].

A transmission network can be viewed as an undirected graph. An edge (i. e. a link of the network) is characterized by a pair of nodes, a transmission cost per circuit and a transmission capacity in the number of circuits. The *routing problem* is the determination of the most economical way of using the available transmission capacities to route a traffic matrix (a number of circuits between various pairs of nodes) through the network. This problem can be expressed as follows:

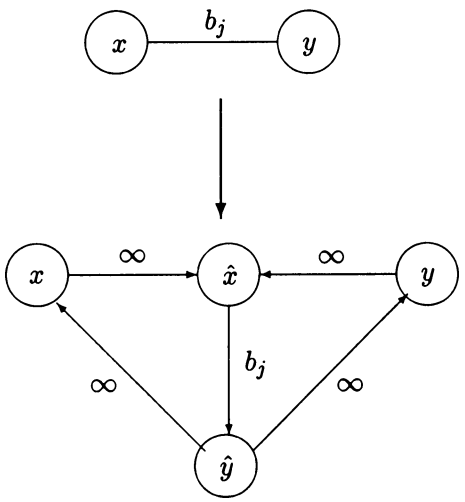
$$\begin{cases} \min & \sum_{k,j} c_j^k \left| x_j^k \right| \\ \text{s.t.} & Ax^k = r^k, \quad k \in K, \\ & \sum_k \left| x_j^k \right| \le b_j, \quad j \in J, \end{cases}$$

where:

- A is the node-arc incidence matrix corresponding to an arbitrary orientation of the network graph (i. e. $A_{ij} = +1$ if arc j is directed away from node i, $A_{ij} = -1$ if arc j is directed towards node i, $A_{ij} = 0$ otherwise);
- $J = \{1, \ldots, \overline{J}\}$ is the set of arcs of the network;
- k is a commodity characterized by a number of circuits, d^k, to be routed through the network between a given pair of nodes, s^k and t^k;
- $K = \{1, \ldots, \overline{K}\}$ is the set of commodities to be routed;
- r^k is the requirement vector for commodity k (i. e. $r^k_{s^k} = d^k$, $r^k_{t^k} = -d^k$ and $r^k_l = 0$ when $l \notin \{s^k, t^k\}$);
- x^k is the flow vector for a given commodity k (i. e. x_j^k is the flow on arc $j = (s, t)$ with $x_j^k > 0$ if the flow goes from s to t and $x_j^k < 0$ if the flow goes from t to s);
- b_j is the capacity of link j;
- c_j^k is the cost per unit of flow on link j for commodity k.

The first group of constraints contains one block of constraints per commodity. These so-called *flow constraints* specify that x is a routing of the traffic requirements. The second group of constraints ties together the various commodities. These *coupling constraints* specify that only a limited number of circuits can be routed through any given edge of the network. The above formulation is called a *node-arc formulation of the problem*.

If we assume that cost coefficient c_j^k are non negative, this nonlinear problem can be replaced with an equivalent standard (linear) multicommodity flow problem by using the transformation of the network graph depicted in the following figure where the figures over edges and arcs denote capacities.



The structure of the resulting model is similar to that of the original model except that absolute values are removed and nonnegativity restrictions on the flow variables are introduced. The transformation is justified by the fact that the support of the flow associated with any given commodity does not contain any directed cycle in the optimal solution of the transformed problem. The transformed model could be solved by any code designed for the solution of directed multicommodity flow problems. However, as this model is much larger than the original nonlinear model a better approach has to be devised when a computationally efficient solution method is sought for. Assuming that $c_j^k > 0$, one can introduce nonnegative variables, x_j^{k+} and x_j^{k-}, such that $x_j^k = x_j^{k+} - x_j^{k-}$ and $|x_j^k| = x_j^{k+} + x_j^{k-}$ to obtain an equiv-

alent linear formulation [11]:

$$\begin{cases} \min & \sum_{k,j} c_j^k \left(x_j^{k+} + x_j^{k-} \right) \\ \text{s.t.} & A\left(x^{k+} - x^{k-}\right) = r^k, \quad \forall k, \\ & \sum_{k} \left(x_j^{k+} + x_j^{k-} \right) \le b_j, \quad \forall j \in J, \\ & x_j^{k+} \ge 0, \quad \forall k \in K, \quad \forall j \in J, \\ & x_j^{k-} \ge 0, \quad \forall k \in K, \quad \forall j \in J. \end{cases}$$

For real networks such as the Paris district transmission network, the number of nodes, $\bar{I}$, can be larger than 100 and the number of links larger than 300. If one assumes that there can be a traffic requirement between any pair of nodes, the size of the problem can be larger than approximately $M \approx 5 \times 10^5$ constraints and $N \approx 3 \times 10^6$ variables, since $\bar{K} = \bar{I}(\bar{I}-1)/2$ in this case. However, the formulation can be often simplified and an equivalent smaller problem solved. In fact, when the cost per unit of flow on a given link does not depend on the commodity (i. e. $c_j^k = c_j^1$ for any k) all the commodities which have a common endpoint can be 'merged'. More precisely, consider a particular node, i, and the set, K_i, of commodities, k, such that $s_k = i$ or $t_k = i$. As the orientation of the commodities is arbitrary the commodities in K_i can be replaced with a single commodity characterized by a requirement vector, r, of components, $r_i = \sum_{k \in K_i} d_k$, $r_j = -d^k$ for all $j \in \cup_{k \in K_i} \{s_k, t_k\} - \{i\}$ and $r_j = 0$ for all $j \notin \cup_{k \in K_i} \{s_k, t_k\}$. A merging of commodities that minimizes the number of merged commodities can be found by solving a minimum node cover problem in a nondirected graph (cf. ► Network location: Covering problems) where the vertices correspond to the endpoints of the commodities and the edges correspond to the commodities. In practice a satisfactory solution to this problem can be obtained by a greedy algorithm where at each iteration the node that covers the larger number of edges not yet covered is selected. In any case, the problem reduces to a number of commodities not larger than $\bar{I}$. Assuming that $\bar{K} = \bar{I}$, one obtains $M \approx 10{,}000$ constraints and $N \approx 60{,}000$ variables. This size of problem is within the reach of modern general purpose linear programming codes. However, if extra constraints were to be included in the initial formulation the merging of the commodities might not be any longer possible. For example, this would be the case if quality of service constraints were introduced to impose that no more than p percent of any traffic requirement is routed on any link. Constraints of the form $x_j^{k+} + x_j^{k-} \le p d_k/100$ would have to be added to the above formulation. The size of this new model would be huge.

Fortunately, specialized algorithms that exploit both the block structure of the problem and the structure of each block of flow constraints can be designed to provide efficient solution methods. A primal partitioning simplex algorithm specialization is presented in [3]. Specializations based on price-directive decomposition (cf. ► Branch and price: Integer programming with column generation), resource-directive decomposition and partitioning of linear systems (simplex and dual affine scaling algorithm) are reviewed in [5]. Technical details on specializations as well as comparative experiments are presented in [4].

It is worth giving more details on one of the most attractive specialization method obtained by applying the Dantzig–Wolfe decomposition principle to the above formulation. The advantage of this specialization is threefold. First, it leads to a reformulation of the problem that appeals to network planners (as the concept of routing path is more directly part of the model) and is easily understood without any references to the Dantzig–Wolfe decomposition principle. Secondly, it can be easily implemented using modern linear programming libraries. Thirdly, and more importantly it is computationally efficient for real-life instances that are usually weakly constrained (no more than 10% edges saturated at optimality) (see [4]).

By applying the Dantzig–Wolfe decomposition principle it can be shown that the routing problem can be reformulated as follows [5]:

$$\begin{cases} \min & \sum_{k,l} [c^k]^\top v_l^k \alpha_l^k \\ \text{s.t.} & \sum_{l} \alpha_l^k = d^k, \quad k \in K, \\ & \sum_{k,l} \alpha_l^k v_l^k \le (b_1, \ldots, b_{\bar{J}})^\top, \\ & \alpha^k \ge 0, \quad k \in K. \end{cases}$$

v_l^k is the characteristic vector of a path, p_l^k, that connects s_k to t_k in the undirected network graph, i. e. the components j of v_l^k is 1 if link j belongs to p_l^k and is 0 otherwise. $[c^k]^\top v_l^k$ is the cost of p_l^k per unit of flow routed on that path (i. e. the sum of the costs c_j^k of its links) and

α_l^k is the amount of traffic routed on path p_l^k. The first set of constraints expresses that d_k units of traffic have to be routed between s_k and t_k whereas the second set of constraints expresses the limitation on the capacity available on the links to route the traffic. The formulation above is known as the *node-path formulation* of the multicommodity flow problem.

This formulation has an exponential number of columns but it is not necessary to express all the columns of this linear program. Indeed, the reduced cost of variable α_l^k is $w_l^k = [c^k + \lambda]^\intercal \nu_l^k - \pi_k$, where $-\lambda$ is the vector of simplex multipliers associated with the capacity constraints and π_k is the vector of simplex multipliers associated with the constraint corresponding to commodity k in the first group of constraints. Therefore $\min_l w_l^k$ can be obtained by computing a shortest path between s_k and t_k in the undirected network graph where the edge lengths are given by the vector $c^k + \lambda$.

In the decomposition algorithm the formulation given above is called the (*full*) *master program*. The $\overline{K}$ shortest path problems which are solved to generate columns of the master program are called the *satellite problems*. At each stage of the algorithm a *reduced master program*, i. e. a program that contains a subset of the columns of the master program, is solved. The first reduced master program is initialized with a set of columns corresponding to a basic matrix and the columns corresponding to the slack variables. If no such initial set of columns is available, artificial variables are added to the formulation and a first phase or a big-M method have to be used to drive the program to a feasible basis. After solving a reduced master program to optimality, the $\overline{K}$ satellite problems are solved to check whether the solution obtained is optimal for the full master program. If the reduced costs corresponding to the solution of the satellite problems are nonnegative the solution to the reduced master program is an optimal solution to the full master program, otherwise the master program is increased with the new columns candidate for basis entry and the simplex algorithm goes on. After the solution of a reduced master program the reduced cost of all variables in that program are nonnegative. Hence, as λ is the reduced cost vector associated with the slack variables of the coupling constraints we deduce that $\lambda \geq 0$ and that in turn the edge lengths in the satellite problems are nonnegative. The satellite problems can therefore be solved using the Dijkstra algorithm (see [9] and [13] for details of efficient implementations).

The solution of each master program can be made more efficient by using a refinement of the GUB specialization of the simplex algorithm (see [4] for details). Some variations of the algorithm may be also applied. For example, some columns generated may be discarded from the reduced master program at later stages, the generation of new columns may be carried out before optimality is reached provided that the simplex multipliers λ are nonnegative.

Finally it is worth mentioning that a new cutting plane technique based on interior point method and called the *analytic center cutting plane method* (ACCPM) was recently proposed to solve large scale convex optimization problems [8]. Its application to the dual of the above formulation gives performances that are roughly similar to those of the Dantzig–Wolfe decomposition algorithm [4] but ACCPM is much more efficient for highly nonlinear problems [7].

Further details on network programming and modeling can be found in [1,2,6,10].

See also

- ► Maximum Flow Problem
- ► Minimum Cost Flow Problem
- ► Multicommodity Flow Problems
- ► Nonconvex Network Flow Problems
- ► Piecewise Linear Network Flow Problems

References

1. Ahuja R, Magnanti T, Orlin J (1992) Network flows. Prentice-Hall, Englewood Cliffs
2. Bazaraa MS, Jarvis JJ (1977) Linear programming and network flows. Wiley, New York
3. Chardaire P, Lisser A (1996) A primal simplex specialization for solving non-oriented multicommodity flow problems. Invest Oper 5(2–3):117–152
4. Chardaire P, Lisser A (1999) Simplex and interior point specialized algorithms for solving non-oriented multicommodity flow problems. Oper Res
5. Chardaire P, Lisser A (2000) Minimum cost multicommodity flow. In: Pardalos PM, Resende M (eds) Handbook Applied Optim. Oxford Univ. Press, Oxford
6. Glover F, Klingman D, Phillips NV (1992) Network models in optimization and their applications in practice. Wiley, New York

7. Goffin J-L, Gondzio J, Sarkissian R, Vial J-P (1996) Solving non linear multicommodity flow problems by the analytic center cutting plane method. Math Program 76:131–154
8. Goffin J-L, Haurie A, Vial J-P (1992) Decomposition and nondifferentiable optimization with the projective algorithm. Manag Sci 38
9. Johnson E (1972) On shortest path and sorting. Proc. Annual Conf. ACM, Boston, p 510
10. Kennington JL, Helgason RV (1980) Algorithms for network programming. Wiley, New York
11. Minoux M (1986) Mathematical programming: Theory and algorithms. Wiley, New York
12. Minoux M, Serrault JY (1980) Subgradient optimization and large scale programming application to multicommodity network synthesis with security constraints. RAIRO 15(2):185–203
13. Williams JWJ (1964) Algorithm 232, heapsort. Comm ACM 7(6):347–348

Nonsmooth Analysis: Fréchet Subdifferentials

Alexander Y. Kruger
Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Ballarat, Australia

MSC2000: 49K27, 90C48, 58C20, 58E30

Article Outline

Keywords
Introduction
Definitions
  Fréchet Subdifferentials
  Fréchet Normal Cone
  Strict Fréchet δ-Subdifferentials
  Limiting Subdifferentials
  Fréchet ε-Subdifferentials and ε-Normals
Formulation
  Direct Calculus
  Strict Differentiability
  Variational Principles
  Sum Rules
  Extremal Principle
See also
References

Keywords

Nonsmooth analysis; Variational analysis; Subdifferential; Normal cone; Optimality conditions; Extremal principle; Multifunction; Asplund space

Introduction

Being natural generalizations of the classical Fréchet derivative and the subdifferential in the sense of convex analysis, *Fréchet subdifferentials* have been known for more than 30 years. They were probably first introduced in finite dimensions in [1] (under the name "lower semidifferentials"). Some of their properties in the infinite-dimensional setting were investigated in [18,21]. During the first decade Fréchet subdifferentials were not widely used because of rather poor (direct) calculus. They mostly served as building blocks for more sophisticated *limiting (Fréchet) subdifferentials* [19,23,30,31].

The discovery of the "fuzzy rules" in the 1980s [11,12,15,29] revitalized interest in Fréchet subdifferentials. It was shown that calculus results and optimality conditions can be formulated in terms of Fréchet and other "simple" subdifferentials computed not at the given point, but at some points arbitrarily close to it, thus incorporating "differential" properties of the function at nearby points. Such results are actually at the core of the corresponding statements for limiting subdifferentials.

Being the smallest among all "simple" subdifferentials with reasonable properties, the Fréchet subdifferentials have proved to be convenient tools for the analysis of nondifferentiable functions on *Asplund* spaces, a very important subclass of general Banach spaces. Furthermore, the main fuzzy results in terms of Fréchet subdifferentials present characterizations of Asplund spaces themselves [6,12,13,34,35,38,45].

The article contains no proofs. A more detailed survey of Fréchet subdifferentials can be found in [25].

Mostly standard notations are used throughout the article. X and Y denote normed linear spaces and X^* and Y^* denote their topological duals. $\langle \cdot, \cdot \rangle$ is a bilinear form defining a canonical paring between a space and its dual. $B_\rho(x)$ stands for a closed ball with center x and radius ρ. We write B_ρ instead of $B_\rho(0)$ and just B if $\rho = 1$ (unit ball).

Definitions

Fréchet Subdifferentials

Let $f : X \to \mathbb{R}_\infty = \mathbb{R} \cup \{+\infty\}$, $f(x) < \infty$. The set

$$\partial f(x) = \left\{ x^* \in X^* : \liminf_{u \to x} \frac{f(u) - f(x) - \langle x^*, u - x \rangle}{\|u - x\|} \geq 0 \right\} \quad (1)$$

is called the *Fréchet subdifferential* of f at x. This set is (norm) closed and convex. The next proposition shows that it generalizes the notions of the Fréchet derivative and the subdifferential in the sense of convex analysis.

Proposition 1

1. *If f is Fréchet-differentiable at x with the derivative $\nabla f(x)$ then $\partial f(x) = \{\nabla f(x)\}$.*
2. *If f is convex then*

$$\partial f(x) = \left\{ x^* \in X^* : f(u) - f(x) \geq \langle x^*, u - x \rangle, \forall u \in X \right\}.$$

Note that the Fréchet subdifferential does not change if another equivalent norm on X is used in (1).

Example 1 The set (1) can be empty. Take $f : \mathbb{R} \to \mathbb{R} : f(u) = -|u|$, $u \in \mathbb{R}$.

One can also consider the *Fréchet superdifferential*

$$\partial^+ f(x) = \left\{ x^* \in X^* : \limsup_{u \to x} \frac{f(u) - f(x) - \langle x^*, u - x \rangle}{\|u - x\|} \leq 0 \right\}. \quad (2)$$

While the set (1) consists of linear continuous functionals "supporting" f from below, the functionals from (2) "support" f from above. Unlike the classical case, the existence of two different derivative-like objects is quite natural for nonsmooth analysis: "differential" properties of a function "from below" and "from above" could be essentially different.

Subdifferentials and superdifferentials are related by the equality

$$\partial(-f)(x) = -\partial^+ f(x). \quad (3)$$

Surely, in the nondifferentiable case at least one of the sets (1) and (2) must be empty.

Proposition 2 *$\partial f(x) \neq \emptyset$ and $\partial^+ f(x) \neq \emptyset$ if and only if f is Fréchet-differentiable at x. In this case one has $\partial f(x) = \partial^+ f(x) = \{\nabla f(x)\}$.*

Example 2 Both sets (1) and (2) can be empty simultaneously. Take $f : \mathbb{R} \to \mathbb{R} : f(u) = u \sin(1/u)$ if $u \neq 0$, and $f(0) = 0$.

Example 3 The fact that the set (1) is a singleton does not imply differentiability. Take $f : \mathbb{R} \to \mathbb{R} : f(u) = \max(u \sin(1/u), 0)$ if $u \neq 0$, and $f(0) = 0$. Then f is nondifferentiable at 0, although one evidently has $\partial f(0) = \{0\}$.

Example 4 Fréchet differentiability is essential in Proposition 1 Gâteaux differentiable functions can be nonsubdifferentiable in the Fréchet sense. Take $f : \mathbb{R}^2 \to \mathbb{R} : f(u_1, u_2) = -\sqrt{|u_1|^2 + |u_2|^2}$ if $u_2 = u_1^2$, $f(u_1, u_2) = 0$ otherwise. The Gâteaux derivative of f is 0, while $\partial f(0) = \emptyset$.

Remark 1 One can define the *Gâteaux subdifferential* on the basis of the notion of the Gâteaux differentiability. For this subdifferential, the analogs of Propositions 1 and 2 and some other results hold true. Considering Gâteaux (and other types of) "simple" subdifferentials can be useful in some applications. The Gâteaux subdifferential always contains the Fréchet subdifferential.

If $\dim X < \infty$ the Fréchet subdifferential can be expressed equivalently in terms of certain generalized directional derivatives [1,16,18,39,42,43].

Fréchet Normal Cone

Now consider a set $\Omega \subset X$ and let $x \in \Omega$. Similarly to definition (1) of the Fréchet subdifferential one can define the *Fréchet normal cone*

$$N(x|\Omega) = \left\{ x^* \in X^* : \limsup_{u \overset{\Omega}{\to} x} \frac{\langle x^*, u - x \rangle}{\|u - x\|} \leq 0 \right\} \quad (4)$$

to Ω at x. Here $u \overset{\Omega}{\to} x$ means that $u \to x$ with $u \in \Omega$.

It is a norm closed and convex cone closely related to the subdifferential defined above. It is actually the Fréchet subdifferential $\partial \delta_\Omega(x)$ of the indicator function

δ_Ω of Ω ($\delta_\Omega(u) = 0$ if $u \in \Omega$ and $\delta_\Omega(u) = \infty$ otherwise).

This fact allows one to deduce some properties of normal cones from the corresponding statements about subdifferentials. Thus, it follows from Proposition 1, that the normal cone (4) generalizes the corresponding notion of convex analysis.

In finite dimensions the Fréchet normal cone coincides with the polar of the *tangent* (*contingent*, *Bouligand tangent*) cone to Ω at x. If X is reflexive it coincides with the polar of the *weak tangent* cone [1,2,9,14,43].

The relationship between Fréchet subdifferentials and normal cones is bilateral. For a function $f : X \to \mathbb{R}_\infty$ one can consider its *epigraph* $\mathrm{epi} f = \{(u, \mu) \in X \times \mathbb{R} : f(u) \le \mu\}$. If the norm on $X \times \mathbb{R}$ is compatible with that on X (that is, $\|(x, 0)\| = \|x\|$), then the following equivalent definition of the Fréchet subdifferential holds true:

$$\partial f(x) = \left\{x^* \in X^* : (x^*, -1) \in N(x, f(x)|\mathrm{epi} f)\right\} . \tag{5}$$

"Horizontal" normals to the epigraph can also be of interest. They define the *singular Fréchet subdifferential* of f at x:

$$\partial^\infty f(x) = \left\{x^* \in X^* : (x^*, 0) \in N(x, f(x)|\mathrm{epi} f)\right\} .$$

Of course, if f is *calm* at x [7,42], that is, $\|f(u)-f(x)\| \le l\|u-x\|$ for some $l > 0$ and for all u in a neighborhood of x, then the latter set is empty.

Strict Fréchet δ-Subdifferentials

As mentioned in "Introduction," Fréchet subdifferentials have poor calculus and their direct application has been rather limited. There exists a way of enriching the properties of the subdifferentials. It consists in considering differential properties of a function at nearby points.

Consider a new derivative-like object based on the Fréchet subdifferential:

$$\hat\partial_\delta f(x) = \bigcup_{\substack{u \in B_\delta(x) \\ |\mathrm{cl} f(u) - f(x)| \le \delta}} \partial(\mathrm{cl} f)(u). \tag{6}$$

It depends on some positive δ. cl f denotes here the lower semicontinuous envelope of f (its epigraph is the closure of the epigraph of f in $X \times \mathbb{R}$). Unlike (1) the set (6) can be nonconvex. It is called the *strict Fréchet δ-subdifferential* of f at x [24].

The *strict Fréchet δ-superdifferential* $\hat\partial^+_\delta f(x)$ of f at x can be defined in a similar way. The equality $\hat\partial^+_\delta f(x) = -\hat\partial_\delta(-f)(x)$ holds true. The strict subdifferentials and superdifferentials can be nonempty simultaneously and can be essentially different. The set $\hat\partial^0_\delta \varphi(x) = \hat\partial_\delta f(x) \cup \hat\partial^+_\delta f(x)$ can be useful in some situations. It is called the *strict Fréchet δ-differential* of f at x.

The *strict Fréchet δ-normal cone* to a set Ω at $x \in \Omega$ is defined similarly:

$$\hat N_\delta(x|\Omega) = \bigcup_{u \in \mathrm{cl}\Omega \cap B_\delta(x)} N(u|\mathrm{cl}\Omega) .$$

The goal of introducing strict Fréchet δ-subdifferentials is mainly notational. They are convenient for formulating "fuzzy" results, but such results can certainly be formulated in terms of ordinary Fréchet subdifferentials.

Limiting Subdifferentials

The limiting Fréchet subdifferentials are defined as limits of "simple" ones [23,30,31,34]. To simplify the definitions we assume in this subsection that $f : X \to \mathbb{R}_\infty$ is lower semicontinuous in a neighborhood of x.

The *limiting Fréchet subdifferential* of f at x is defined as

$$\bar\partial f(x) = \{x^* \in X^* : \exists \quad \text{sequences} \\ \{x_k\} \subset X, \{x^*_k\} \subset X^* \text{such that} \\ x_k \xrightarrow{f} x,\ x^*_k \xrightarrow{w^*} x^* \text{ and } x^*_k \in \partial f(x_k),\ k = 1, 2, \dots\} . \tag{7}$$

The notations $x_k \xrightarrow{f} x$ and $x^*_k \xrightarrow{w^*} x^*$ here mean, respectively, that $x_k \to x$ with $f(x_k) \to f(x)$ (*f-attentive convergence* [42]), and x^*_k converges to x^* in the *weak* * topology of X^*.

$\bar\partial f(x)$ is a weakly* sequentially closed set in X^*. In general it is nonconvex. If f is strictly differentiable at x the set (7) reduces to the derivative.

Using strict δ-subdifferentials, one can rewrite (7) in the following way:

$$\bar{\partial} f(x) = \bigcap_{\delta>0} \mathrm{cl}^* \hat{\partial}_\delta f(x) ,$$

where cl* denotes the weak* sequential closure.

Other limiting objects (the *limiting superdifferential*, the *limiting differential*, the *limiting normal cone*, the *singular limiting subdifferential*, and the *limiting coderivative*) can be defined in a similar way.

Thus, the limiting normal cone to a closed set Ω is defined by the equality

$$\bar{N}(x|\Omega) = \bigcap_{\delta>0} \mathrm{cl}^* \hat{N}_\delta(x|\Omega) .$$

It coincides with the limiting subdifferential of the indicator function of Ω. The analog of (5) is also valid:

$$\bar{\partial} f(x) = \{x^* \in X^* : (x^*, -1) \in \bar{N}(x, f(x)|\mathrm{epi} f)\} .$$

The limiting subdifferentials and normal cones have been well investigated. They possess good calculus (which is the consequence of the fuzzy calculus of Fréchet subdifferentials; see [19,23,30,31,32,34,36,42] for the properties of these objects and some examples). They have proved to be very efficient for formulating optimality conditions in nonsmooth optimization [20,22,29,30,31,32,34,36], especially in finite dimensions. When applying limiting subdifferentials in infinite dimensional spaces, one must be careful about nontriviality of the limits in the weak* topology. Additional regularity conditions are needed (*compact epi-Lipschitzness*, *sequential normal compactness*, *partial sequential normal compactness* [4,34,36], etc.)

Fréchet ε-Subdifferentials and ε-Normals

In some cases it can be convenient to use ε-extensions of the Fréchet subdifferentials and normal cones [21,23,29,44]. For instance, the *Fréchet ε-subdifferential* and the *Fréchet ε-superdifferential* of f at x are defined as

$$partial_\varepsilon f(x) = \left\{ x^* \in X^* : \liminf_{u \to x} \frac{f(u) - f(x) - \langle x^*, u - x\rangle}{\|u - x\|} \geq -\varepsilon \right\} ,$$

$$\partial_\varepsilon^+ f(x) = \left\{ x^* \in X^* : \limsup_{u \to x} \frac{f(u) - f(x) - \langle x^*, u - x\rangle}{\|u - x\|} \leq \varepsilon \right\} .$$

Unlike (1) and (2), these sets depend on the specific norm on X (when $\varepsilon > 0$).

The next two propositions extend Propositions 1 and 2 respectively.

Proposition 3 *If f is convex then* $\partial_\varepsilon f(x) = \partial f(x) + \varepsilon B^* = \{x^* \in X^* : f(u) - f(x) \geq \langle x^*, u - x\rangle - \varepsilon\|u - x\|, \ \forall u \in X\}$.

Remark 2 Note that the above ε-subdifferential differs from the corresponding notion of convex analysis, which is usually defined [41] as the set of all $x^* \in X^*$, such that $f(u) - f(x) \geq \langle x^*, u - x\rangle - \varepsilon$ for all $u \in X$.

Proposition 4 *If $x_1^* \in \partial_{\varepsilon_1} f(x)$, $x_2^* \in \partial_{\varepsilon_2}^+ f(x)$, $\varepsilon_1 \geq 0$, $\varepsilon_2 \geq 0$ then $\|x_1^* - x_2^*\| \leq \varepsilon_1 + \varepsilon_2$.*

Formulation

Direct Calculus

The propositions below present some simple calculus results for Fréchet subdifferentials. Most of them follow directly from the definitions. More advanced statements of fuzzy calculus are presented in the next subsections.

Proposition 5 *If f attains a local minimum at x then* $0 \in \partial f(x)$.

Proposition 6 $\partial(\lambda f)(x) = \lambda \partial f(x)$ *for any* $\lambda > 0$.

Proposition 7 *Let $f_1, f_2 : X \to \mathbb{R}_\infty$ be finite at x. Then*

$$\partial(f_1 + f_2)(x) \supset \partial f_1(x) + \partial f_2(x) . \tag{8}$$

The above proposition presents an example of a *Sum Rule*. Usually the sum rule is the central result of any subdifferential calculus. Unfortunately, the inclusion (8) is almost useless: it does not allow one to decompose elements of the subdifferential of the sum of functions in terms of elements of subdifferentials of the original functions. Simple examples show that inclusion (8) can be strict even in the convex case. The next proposition gives two important cases when the equality holds true in (8).

Proposition 8 *Let $f_1, f_2 : X \to \mathbb{R}_\infty$ be finite at x.*

1. *If f_1 and f_2 are convex and one of them is continuous at x then*

$$\partial(f_1 + f_2)(x) = \partial f_1(x) + \partial f_2(x).$$

2. *If f_1 is Fréchet-differentiable at x then*

$$\partial(f_1 + f_2)(x) = \nabla f_1(x) + \partial f_2(x)\ . \tag{9}$$

Part 1 of Proposition 8 is known as the Moreau–Rockafellar theorem [41]. Part 2 is a simple corollary of Proposition 7. It is an interesting example, when an inclusion implies an equality. Indeed, applying Proposition 7 to the sum of the functions $f_1 + f_2$ and $-f_2$ and making use of (3), one gets

$$\partial f_2(x) \supset \partial(f_1 + f_2)(x) - \partial^+ f_1(x)\ . \tag{10}$$

Taking into account Proposition 2, the inclusions (8) and (10) imply (9).

Proposition 8 yields a simple necessary optimality condition generalizing Proposition 5.

Proposition 9 *Let $f_1 : X \to \mathbb{R}$ be Fréchet-differentiable at x where $f_2 : X \to \mathbb{R}_\infty$ is finite. If $f_1 + f_2$ attains a local minimum at x then $-\nabla f_1(x) \subset \partial f_2(x)$.*

The next two assertions are corollaries of Propositions 7 and 9.

Proposition 10 *Let $x \in \Omega_1 \cap \Omega_2$.*
Then $N(x|\Omega_1 \cap \Omega_2) \supset N(x|\Omega_1) + N(x|\Omega_2)$.

Proposition 11 *Let f be Fréchet-differentiable at x. If f attains at x a local minimum on Ω then $-\nabla f(x) \in N(x|\Omega)$.*

For a set $\Omega \subset X$ and $u \in X$ consider the distance $d_\Omega(u) = \inf_{\omega\in\Omega} \|u - \omega\|$.

Proposition 12 ([18]) *For any $x \in \Omega$ one has $\partial d_\Omega(x) = \{x^* \in N(x|\Omega) : \|x^*\| \le 1\}$.*

Strict Differentiability

Recall that f is called *strictly differentiable* [34,42] at x (with the strict derivative $\nabla f(x)$) if

$$\lim_{\substack{u \to x,\, u' \to x \\ u \ne u'}} \frac{f(u') - f(u) - \langle \nabla f(x), u' - u\rangle}{\|u' - u\|} = 0\ .$$

In the case of a strictly differentiable function the Fréchet subdifferentials and superdifferentials at nearby points cannot differ much from the strict derivative.

Proposition 13 ([25]) *If f is strictly differentiable at x with the derivative $\nabla f(x)$ then for any $\varepsilon > 0$ there exists a $\delta > 0$ such that:*

1. *$\hat\partial^0_\delta f(x) \subset \nabla f(x) + \varepsilon B^*$.*
2. *$\nabla f(x) \in \partial_\varepsilon f(u) \cap \partial^+_\varepsilon f(u)$ for all $u \in B_\delta(x)$.*

The rest of the section is devoted to "fuzzy" results in terms of Fréchet subdifferentials and strict Fréchet δ-differentials.

Variational Principles

The *variational principles* by Ekeland [10], Borwein and Preiss [3] as well as their subsequent followers [6,8,34,40,42] are very powerful tools of modern variational analysis. They make it possible to substitute an "almost minimal" point (up to ε) by another point, arbitrarily close to the initial one, which is the local minimizer for a slightly perturbed (usually by adding a small term) function. Thus, such principles can be viewed as fuzzy results.

The next assertion is valid for an arbitrary Asplund space. It is known as the *Subdifferential Variational Principle*. Let us recall that a Banach space is called *Asplund* [6,8,34,40] if any continuous convex function on it is Fréchet-differentiable on a dense G_δ set of points. Asplund spaces form a rather broad subclass of Banach spaces. It includes, for instance, all spaces which admit Fréchet-differentiable bump functions (in particular, Fréchet smooth spaces). Reflexive spaces are examples of Fréchet smooth spaces.

Asplund spaces provide a very convenient framework for investigating "differential" properties of nonsmooth functions. Actually the Asplund property of a Banach space is not only sufficient but also a necessary condition for the fulfillment of some basic results in nonsmooth analysis involving Fréchet normals and subdifferentials (see [6,13,34,35,38] and the statements below).

Theorem 1 (Mordukhovich and Wang [38]) *Let X be Asplund, $f : X \to \mathbb{R}_\infty$ be lower semicontinuous and bounded below, $\varepsilon > 0$, $\lambda > 0$. Suppose that $f(x) < \inf f + \varepsilon$. Then there exists a $u \in B_\lambda(x)$ and an $x^* \in \partial f(u)$, such that $f(u) \le f(x)$ and $\|x^*\|_* < \varepsilon/\lambda$.*

The following theorem states that the class of Fréchet *subdifferentiability spaces* [15] coincides with Asplund spaces.

Theorem 2 *The following assertions are equivalent:*
1. *X is an Asplund space.*
2. *For any lower semicontinuous function $f : X \to \mathbb{R}_\infty$ the set $\{u \in X : \partial f(u) \neq \emptyset\}$ is dense in $\mathrm{dom} f$.*
3. *For any lower semicontinuous function $f : X \to \mathbb{R}_\infty$ there exists an $x \in \mathrm{dom} f$ such that $\partial f(x) \neq \emptyset$.*

Sum Rules

After the sum rule was first established in the limiting form in [19] (see [23,31]) the fuzzy versions were derived in [12,15] (see also [6,17,37]). Now two main versions of the fuzzy sum rule are known. For simplicity they are formulated below in terms of strict δ-subdifferentials.

Rule 1 Weak Fuzzy Sum Rule *Let $f_1, f_2 : X \to \mathbb{R}_\infty$ be finite at x and lower semicontinuous near x. Then $\partial(f_1 + f_2)(x) \subset \hat{\partial}_\delta f_1(x) + \hat{\partial}_\delta f_2(x) + U^*$ for any $\delta > 0$ and any weak* neighborhood U^* of 0 in X^*.*

Rule 2 Strong Fuzzy Sum Rule *Let $f_1 : X \to \mathbb{R}_\infty$ be finite at x and lower semicontinuous near x and $f_2 : X \to \mathbb{R}$ be Lipschitz continuous near x. Then $\partial(f_1 + f_2)(x) \subset \hat{\partial}_\delta f_1(x) + \hat{\partial}_\delta f_2(x) + \delta B^*$ for any $\delta > 0$.*

A Banach space is called a *trustworthy space* [15] (for some kind of a subdifferential) if Rule 1 is valid in it. The following theorem proved by Fabian [12] states that for the Fréchet subdifferential the class of trustworthy spaces coincides with Asplund spaces.

Theorem 3 *The following assertions are equivalent:*
1. *X is an Asplund space.*
2. *The Weak Fuzzy Sum Rule is valid in X.*
3. *The Strong Fuzzy Sum Rule is valid in X.*

The Weak Fuzzy Sum Rule yields the following representation of Fréchet normals to the intersection of closed sets.

Proposition 14 *Let Ω_1, Ω_2 be closed subsets in an Asplund space X and $x \in \Omega_1 \cap \Omega_2$. Then $N(x, \Omega_1 \cap \Omega_2) \subset \hat{N}_\delta(x, \Omega_1) + \hat{N}_\delta(x, \Omega_2) + U^*$ for any $\delta > 0$ and any weak* neighborhood U^* of 0 in X^*.*

Other fuzzy calculus results (chain rules, formulas for maximum-type functions, mean value theorems, etc.) for functions and multifunctions can be deduced from (some form of) the sum rule [5,16,24,37].

Extremal Principle

The *Extremal Principle* continues the line of variational principles discussed above and is in a sense equivalent to them as well as to the sum rules.

Let Ω_1, Ω_2 be closed subsets in X. They are called *locally extremal* [29,30] near $x \in \Omega_1 \cap \Omega_2$ if there exists a neighborhood U of x and sequences $\{a_{ik}\} \in X$, $i = 1, 2$, $k = 1, 2, \ldots$, such that $a_{ik} \to 0$ when $k \to \infty$ and $(\Omega_1 - a_{1k}) \cap (\Omega_2 - a_{2k}) \cap U = \emptyset$, $k = 1, 2, \ldots$

This means that by an arbitrarily small shift the sets can be made unintersecting in a neighborhood of x. The definition represents a rather general notion of extremality: some locally extremal system corresponds to a local solution of any optimization problem (see various examples in [22,29,31,33]).

The *Extremal Principle*, first established in [29] (see also [22,30,31]) for the case of a Fréchet smooth space (and in terms of ε-normals) and in [35] (see [34]) in the Asplund space setting, provides a dual space characterization of locally extremal systems in terms of Fréchet normals. It can be viewed as a fuzzy form of the *separation* property.

Extremal Principle *If a system of sets Ω_1, Ω_2 is locally extremal near $x \in \Omega_1 \cap \Omega_2$ then for any $\delta > 0$ there exist elements $x_1^* \in \hat{N}_\delta(x|\Omega_1)$, $x_2^* \in \hat{N}_\delta(x|\Omega_2)$ such that $\|x_1^* + x_2^*\| < \delta$, $\|x_1^*\| + \|x_2^*\| = 1$.*

The following theorem proved in [35] shows that the Extremal Principle provides an extremal characterization of Asplund spaces.

Theorem 4 *The following assertions are equivalent:*
1. *X is an Asplund space.*
2. *The Extremal Principle is valid in X.*

Due to Theorems 3 and 4 the Extremal Principle is equivalent to the Sum Rules. It is also equivalent to some other basic results of nonsmooth analysis [6,34,45].

The Extremal Principle can be viewed as a certain extension of the classical separation theorem for convex sets. It was used in [22,29,30,31,36] and in many other papers as a main tool for deducing calculus formulas and necessary optimality conditions.

The local extremality assumption in the Extremal Principle can be replaced by a weaker *stationarity* condition [26,27,28]. The resulting *Extended Extremal Principle* is formulated as a necessary and sufficient condition and is also equivalent to the asplundity of the space.

As noticed in [35], considering the extremal system provided by the pair $\{x\}, \Omega$, where x is a boundary point of a closed set Ω makes it possible to deduce from Theorem 4 the following nonconvex generalization of the well-known *Bishop–Phelps theorem* [40].

Corollary 1 *Let Ω be a closed subset in an Asplund space X and let $x \in bd\Omega$. Then for any $\delta > 0$ there exists $x^* \in \hat{N}_\delta(x|\Omega)$ such that $\|x^*\| = 1$.*

See also

► Nonsmooth Analysis: Weak Stationarity

References

1. Bazaraa MS, Goode JJ, Nashed ZZ (1974) On the cones of tangents with applications to mathematical programming. J Optim Theory Appl 13:389–426
2. Borwein JM (1978) Weak tangent cones and optimization in a Banach space. SIAM J Contr Optim 16:512–522
3. Borwein JM, Preiss D (1987) A smooth variational principle with applications to subdifferentiability and differentiability of convex functions. Trans Am Math Soc 303:517–527
4. Borwein JM, Strojwas HM (1985) Tangential approximations. Nonlinear Anal 9:1347–1366
5. Borwein JM, Zhu QJ (1999) A survey of subdifferential calculus with applications. Nonlinear Anal 38:687–773
6. Borwein JM, Zhu QJ (2005) Techniques of Variational Analysis. In: CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC, 20. Springer, New York
7. Clarke FH (1983) Optimization and Nonsmooth Analysis. Wiley, New York
8. Deville R, Godefroy G, Zizler V (1993) Smoothness and Renorming in Banach spaces. In: Pitman Monographs and Surveys in Pure and Applied Math., vol 64. Longman, Harlow UK
9. Dubovitskii AY, Milyutin AA (1965) Extremum problems in the presence of restrictions. USSR Comp Math Math Phys 5:1–80
10. Ekeland I (1974) On the variational principle. J Math Anal Appl 47:324–353
11. Fabian M (1986) Subdifferentials, local ε-supports and Asplund spaces. J London Math Soc 34:568–576
12. Fabian M (1989) Subdifferentiability and trustworthiness in the light of a new variational principle of Borwein and Preiss. Acta Univ Carolinae 30:51–56
13. Fabian M, Mordukhovich BS (1998) Nonsmooth characterizations of Asplund spaces and smooth variational principles. Set-Valued Anal 6:381–406
14. Gould FJ, Tolle JW (1975) Optimality conditions and constraint qualifications in Banach space. J Optim Theory Appl 15:667–684
15. Ioffe AD (1983) On subdifferentiability spaces. Ann NY Acad Sci 410:107–119
16. Ioffe AD (1984) Calculus of Dini subdifferentials of functions and contingent coderivatives of set-valued maps. Nonlinear Anal 8:517–539
17. Ioffe AD (1990) Proximal analysis and approximate subdifferentials. J London Math Soc 41:175–192
18. Kruger AY (1977) Subdifferentials of nonconvex functions and generalized directional derivatives. Deposited in VINITI no. 2661–77. Minsk (in Russian)
19. Kruger AY (1981) Generalized differentials of nonsmooth functions. Deposited in VINITI no. 1332–81. Minsk (in Russian)
20. Kruger AY (1981) Necessary conditions of an extremum in problems of nonsmooth optimization. Deposited in VINITI no. 1333–81. Minsk (in Russian)
21. Kruger AY (1981) ε-semidifferentials and ε-normal elements. Deposited in VINITI no. 1331–81. Minsk (in Russian)
22. Kruger AY (1985) Generalized differentials of nonsmooth functions and necessary conditions for an extremum. Siberian Math J 26:370–379
23. Kruger AY (1985) Properties of generalized differentials. Siberian Math J 26:822–832
24. Kruger AY (1996) On calculus of strict ε-semidifferentials. Dokl Akad Nauk Belarusi 40(4):34–39 (in Russian)
25. Kruger AY (2003) On Fréchet subdifferentials. J Math Sci (NY) 116(3):3325–3358 Optimization and related topics 3
26. Kruger AY (2004) Weak stationarity: eliminating the gap between necessary and sufficient conditions. Optimization 53(2):147–164
27. Kruger AY (2005) Stationarity and regularity of set systems. Pac J Optim 1(1):101–126
28. Kruger AY (2006) About regularity of collections of sets. Set-Valued Anal 14(2):187–206
29. Kruger AY, Mordukhovich BS (1980) Extremal points and the Euler equation in nonsmooth optimization. Dokl Akad Nauk BSSR 24(8):684–687 (in Russian)
30. Kruger AY, Mordukhovich BS (1980) Generalized normals and derivatives and necessary conditions for an extremum in problems of nondifferentiable programming. Deposited in VINITI, I – no. 408–80, II – no. 494–80. Minsk (in Russian)
31. Mordukhovich BS (1988) Approximation Methods in Problems of Optimization and Control. Nauka, Moscow (in Russian)
32. Mordukhovich BS (1994) Generalized differential calculus for nonsmooth and set-valued mappings. J Math Anal Appl 183(1):250–288
33. Mordukhovich BS (2001) The extremal principle and its applications to optimization and economics. In: Rubinov A,

Glover B (eds) Optimization and Related Topics, Applied Optimization, vol 47. Kluwer, Dordrecht, pp 343–369
34. Mordukhovich BS (2006) Variational Analysis and Generalized Differentiation. I Basic theory, vol 330 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Springer, Berlin
35. Mordukhovich BS, Shao Y (1996) Extremal characterizations of Asplund spaces. Proc Am Math Soc 124:197–205
36. Mordukhovich BS, Shao Y (1996) Nonsmooth sequential analysis in Asplund spaces. Trans Am Math Soc 348:1235–1280
37. Mordukhovich BS, Shao Y (1997) Fuzzy calculus for coderivatives of multifunctions. Nonlinear Anal 29:605–626
38. Mordukhovich BS, Wang B (2002) Necessary suboptimality and optimality conditions via variational principles. SIAM J Control Optim 41(2):623–640
39. Penot J-P (1978) Calcul sous-differentiel et optimisation. J Funct Anal 27:248–276
40. Phelps RR (1993) Convex Functions, Monotone Operators and Differentiability, 2nd edn. Lecture Notes in Mathematics, vol 1364. Springer, New York
41. Rockafellar RT (1970) Convex Analysis. Princeton Univ Press, Princeton
42. Rockafellar RT, Wets RJ-B (1998) Variational Analysis. Springer, New York
43. Sachs E (1978) Differentiability in optimization theory. Math Operationsforsch Statist, Ser Optim 9:497–513
44. Treiman JS (1986) Clarke's gradients and epsilon-subgradients in Banach spaces. Trans Am Math Soc 294(1):65–78
45. Zhu QJ (1998) The equivalence of several basic theorems for subdifferentials. Set-Valued Anal 81:171–185

Nonsmooth Analysis: Weak Stationarity

ALEXANDER Y. KRUGER
Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Ballarat, Australia

MSC2000: 90C46, 90C48, 58C20, 58E30

Article Outline

Keywords
Introduction
Definitions
Inf-θ-Stationarity and Inf-θ-Regularity
Inf-τ-Stationarity and Inf-τ-Regularity
Sup-Stationarity and Sup-Regularity
Dual Stationarity and Regularity
Formulation
Relations Between the "Elementary" Constants
Relations Between the "Strict" Constants
Relations Between the Primal and Dual Constants
Differentiable Functions
Convex Functions
See also
References

Keywords

Variational analysis; Subdifferential; Normal cone; Optimality; Stationarity; Regularity; Slope; Multifunction; Asplund space

Introduction

The article considers different stationarity and regularity concepts for extended real-valued functions on metric spaces.

All the properties can be characterized in terms of certain local constants. A function is said to be stationary at a point (in some sense) if the corresponding constant is zero (a critical point). Otherwise the function is said to be regular at this point (in the same sense), and the constant provides a quantitative estimate of regularity.

Traditionally the stationary behavior of a function at a point (*stationary point*) means that it is arbitrarily close to a constant near this point, that is the rate of change of the function is infinitely small compared to the increment of the variable. Of course, this is equivalent to the derivative being equal to zero (dual characterization of stationarity). The classical examples of the stationary behavior of a function, found in the textbooks, are given by the functions $y = x^2$, $y = -x^2$, and $y = x^3$. These three examples characterize the three possible types of stationary behavior in the differentiable case.

Stationarity arises naturally in optimization theory: a point of minimum or maximum is necessarily a stationary point. At the same time it is easy to see that weaker stationarity concepts are applicable to optimization problems. For instance, when dealing with minimizing a real-valued function, only its decrement must be infinitely small at a stationary point, and the function itself does not need to be arbitrarily close to a constant or even differentiable at the point. Of course, if

it is differentiable, one has the same classical stationarity concept. In the nondifferentiable case more types of the stationary behavior are possible. The examples are: $y = |x|$ and $y = \max(x, -x^2)$ (both functions are considered near the point $x = 0$). One can speak about *inf-stationarity* (the term suggested by Vladimir F. Demianov). From the point of view of maximization a concept of *sup-stationarity* can be considered in a similar way. Thus in the nondifferentiable case the stationarity splits into two "semi-stationarity" concepts.

Another pair of stationarity concepts can be of interest in optimization theory: the point itself may not be stationary (inf or sup), but in any of its neighborhood there exists another point in which the behavior of the function is arbitrarily close to stationary (a "fuzzy" condition). We will speak about *weak stationarity* (inf or sup). The exact definitions will be given below.

An example of this type of stationary behavior is given by the function $y = x\sin(1/x)$ if $x \neq 0$, and $y = 0$ if $x = 0$. One can easily see that this function is differentiable on $\mathbb{R}\backslash\{0\}$, and there exists a sequence $\{x_k\}$ such that $x_k \to 0$ and x_k is a point of local minimum, $k = 1, 2, \ldots$ Another example: $y = x + x^2\sin(1/x)$ if $x \neq 0$, and $y = 0$ if $x = 0$. This function is everywhere differentiable, $y'(0) = 1$, and there exists a sequence $\{x_k\}$ such that $x_k \to 0$ and $y'(x_k) = 0$, $k = 1, 2, \ldots$

Stationarity concepts can also be defined in terms of dual space elements (subdifferentials). The relations between primal and dual definitions provide dual characterizations of (primal space concepts of) stationarity.

This article contains no proofs. A more detailed description of the stationarity and regularity concepts for real-valued functions can be found in [11].

Mostly standard notations are used throughout this article. X denotes a metric space with distance d. $B_\rho(x)$ stands for a closed ball with center x and radius ρ.

Definitions

Inf-θ-Stationarity and Inf-θ-Regularity

Let f be a function on a metric space X with values in the extended real line $\mathbb{R}_\infty = \mathbb{R} \cup \{+\infty\}$. It is assumed to be finite at some point $x^\circ \in X$.

For $\rho > 0$ define the constant

$$\theta_\rho[f](x^\circ) = \inf_{x \in B_\rho(x^\circ)} f(x) - f(x^\circ). \tag{1}$$

Note that $\theta_\rho[f](x^\circ) \le 0$ and the equality $\theta_\rho[f](x^\circ) = 0$ for some $\rho > 0$ (for all $\rho > 0$) means that x° is a point of local (global) minimum of f.

Of course, the infimum in (1) can be limited to the set $\{x \in B_\rho(x^\circ)\colon f(x) \le f(x^\circ)\}$, or even to the set $\{x \in B_\rho(x^\circ)\colon f(x) < f(x^\circ)\}$ under the additional agreement that the infimum over the empty subset of $\mathbb{R}_-$ is 0.

The function $\rho \to \theta_\rho[f](x^\circ)$ is nonincreasing on $\mathbb{R}_+$ and $\lim_{\rho\to+0} \theta_\rho[f](x^\circ) \le 0$. The equality $\lim_{\rho\to+0} \theta_\rho[f](x^\circ) = 0$ means that f is lower semicontinuous at x°. In the latter case it can be important to know how quickly $\theta_\rho[f](x^\circ)$ approaches 0 compared to ρ.

Define two more "derivativelike" constants based on (1):

$$\theta[f](x^\circ) = \limsup_{\rho\to+0} \frac{\theta_\rho[f](x^\circ)}{\rho}, \tag{2}$$

$$\hat\theta[f](x^\circ) = \limsup_{x \xrightarrow{f} x^\circ, \rho\to+0} \frac{\theta_\rho[f](x)}{\rho}, \tag{3}$$

where $x \xrightarrow{f} x^\circ$ means that $x \to x^\circ$ with $f(x) \to f(x^\circ)$. Due to the variations of x in (3), $\hat\theta[f](x^\circ)$ gains some properties of the strict derivative.

The constants (2) and (3) are nonpositive too, and "zero cases" correspond to certain kinds of stationary behavior of f near x°. If a constant is strictly negative, this can be considered as a kind of regularity.

Definition 1 f is
(i) *inf-θ-stationary* at x° if $\theta[f](x^\circ) = 0$;
(ii) *weakly inf-θ-stationary* at x° if $\hat\theta[f](x^\circ) = 0$;
(iii) *inf-θ-regular* at x° if $\theta[f](x^\circ) < 0$;
(iv) *strongly inf-θ-regular* at x° if $\hat\theta[f](x^\circ) < 0$.

The purpose of the "inf" prefix in this definition is to emphasize that minimization problems are addressed here. Unlike the classical case, stationarity-regularity properties of nondifferentiable functions "from below" and "from above" can be essentially different.

Inf-τ-Stationarity and Inf-τ-Regularity

Another way of defining stationarity-regularity is based on using slightly modified versions of (2) and (3):

$$\tau[f](x^\circ) = \liminf_{x\to x^\circ} \frac{[f(x) - f(x^\circ)]_-}{d(x, x^\circ)}, \tag{4}$$

$$\hat{\tau}[f](x^\circ) = \limsup_{\substack{f \\ x \to x^\circ,\ \rho \to +0}} \inf_{u \in B_\rho(x) \setminus \{x\}} \frac{[f(u) - f(x)]_-}{d(u,x)} . \tag{5}$$

The notation $[\alpha]_- = \min(\alpha, 0)$ is used here. Again, only the points $x \in B_\rho(x^\circ)$ with $f(x) < f(x^\circ)$ and $u \in B_\rho(x)$ with $f(u) < f(x)$ are of interest in (4) and (5), respectively. The role of the notation is to handle the case where the set of such points is empty. Similarly to (2) and (3), these constants are nonpositive.

Remark 1 $\tau[f](x^\circ)$ coincides up to a sign with the *strong slope* $|\nabla f|(x^\circ)$ of f at x° [1] (see also [5]).

Definition 2 f is
(i) *inf-τ-stationary* at x° if $\tau[f](x^\circ) = 0$;
(ii) *weakly inf-τ-stationary* at x° if $\hat{\tau}[f](x^\circ) = 0$;
(iii) *inf-τ-regular* at x° if $\tau[f](x^\circ) < 0$;
(iv) *strongly inf-τ-regular* at x° if $\hat{\tau}[f](x^\circ) < 0$.

The relations between the constants (2), (3) and (4), (5), as well as between the corresponding stationarity and regularity concepts will be discussed in the next section.

Sup-Stationarity and Sup-Regularity

Similarly to (1)–(5) corresponding "maximization" constants can be defined. To do this one has to replace "inf," "lim inf," "lim sup," and $[\cdot]_-$ by "sup," "lim sup," "lim inf," and $[\cdot]_+$, respectively, in the corresponding definitions. The resulting constants are nonnegative. They are related to (1)–(5) by the following equalities:

$$\theta_\rho^+[f](x^\circ) = -\theta_\rho[-f](x^\circ) ,$$
$$\theta^+[f](x^\circ) = -\theta[-f](x^\circ) ,$$
$$\hat{\theta}^+[f](x^\circ) = -\hat{\theta}[-f](x^\circ) ,$$
$$\tau^+[f](x^\circ) = -\tau[-f](x^\circ) ,$$
$$\hat{\tau}^+[f](x^\circ) = -\hat{\tau}[-f](x^\circ)$$

and lead to similar *sup-stationarity* and *sup-regularity* concepts.

Of course, for a function f the set of sup-stationary (sup-regular) points is different in general from that of inf-stationary (inf-regular) points.

The "combined" concepts can also be of interest. It is natural to say that a function is stationary (in some sense) at a point if it is either inf-stationary or sup-stationary at this point. In constrast, the regularity property for a function is satisfied when this function is both inf-regular and sup-regular at the point.

Definition 3 f is
(i) *θ-stationary* at x° if $\max(\theta[f](x^\circ), \theta[-f](x^\circ)) = 0$;
(ii) *weakly θ-stationary* at x° if $\max(\hat{\theta}[f](x^\circ), \hat{\theta}[-f](x^\circ)) = 0$;
(iii) *θ-regular* at x° if $\max(\theta[f](x^\circ), \theta[-f](x^\circ)) < 0$;
(iv) *strongly θ-regular* at x° if $\max(\hat{\theta}[f](x^\circ), \hat{\theta}[-f](x^\circ)) < 0$;
(v) *τ-stationary* at x° if $\max(\tau[f](x^\circ), \tau[-f](x^\circ)) = 0$;
(vi) *weakly τ-stationary* at x° if $\max(\hat{\tau}[f](x^\circ), \hat{\tau}[-f](x^\circ)) = 0$;
(vii) *τ-regular* at x° if $\max(\tau[f](x^\circ), \tau[-f](x^\circ)) < 0$;
(viii) *strongly τ-regular* at x° if $\max(\hat{\tau}[f](x^\circ), \hat{\tau}[-f](x^\circ)) < 0$.

Strong inf-regularity can be interpreted in the following way: all points in a neighborhood of a given point have "descent sequences," and the rate of descent is uniform. In contrast to that, strong regularity is equivalent to the existence of both descent and ascent sequences with the uniformity property.

Dual Stationarity and Regularity

All definitions in the preceding subsections are primal space definitions. As in the classical analysis, dual characterizations of stationarity and regularity concepts are important. In the case of a normed linear space, such characterizations can be formulated in terms of *Fréchet subdifferentials*.

Let X be a normed linear space. Its (topological) dual is denoted X^*. $\langle \cdot, \cdot \rangle$ is the bilinear form defining the duality pairing. Recall that the Fréchet subdifferential of f at x° is defined as

$$\partial f(x^\circ) = \left\{ x^* \in X^* : \liminf_{x \to x^\circ} \frac{f(x) - f(x^\circ) - \langle x^*, x - x^\circ \rangle}{||x - x^\circ||} \geq 0 \right\} . \tag{6}$$

Definition 4 f is
(i) *inf-d-stationary* at x° if $0 \in \partial f(x^\circ)$;
(ii) *inf-d-regular* at x° if $0 \notin \partial f(x^\circ)$.

It follows immediately from the definitions that in the normed space setting inf-τ-stationarity (inf-τ-regularity) is equivalent to inf-d-stationarity (inf-d-regularity).

Somewhat more complicated constructions are needed for the characterization of weak stationarity and strong regularity.

Let us assume for simplicity that f is lower semicontinuous near x°.

In the general nonconvex setting the subdifferential mapping $\partial f(\cdot)$ fails to possess good (semi-)continuity properties. In fact, the set $\partial f(x)$ can be empty rather often. Based on (6) one can define a more robust derivativelike object:

$$\hat{\partial}_\delta f(x^\circ) = \bigcup_{\substack{x\in B_\delta(x^\circ)\\ |f(x)-f(x^\circ)|\le\delta}} \partial f(x)\,. \tag{7}$$

This object depends on a positive parameter δ and accumulates information on "differential" properties of f at nearby points, thus attaining some properties of the strict derivative. The set (7) is called the *strictδ-subdifferential* of f at x° (see [7,8,9]). In contrast to (6), set (7) can be nonconvex. However, it possesses certain subdifferential calculus.

Using (7) one more constant can be defined for characterizing stationarity/regularity properties of f:

$$\eta[f](x^\circ) = \liminf_{\delta\to 0}\{||x^*|| : x^* \in \hat{\partial}_\delta f(x^\circ)\}\,. \tag{8}$$

Unlike the constants considered in the preceding subsections, this constant is nonnegative.

Definition 5 f is
(i) *inf-η-stationary* at x° if $\eta[f](x^\circ) = 0$;
(ii) *inf-η-regular* at x° if $\eta[f](x^\circ) > 0$.

Note that the inf-η-stationary condition $\eta[f](x^\circ) = 0$ does not imply the inclusion $0 \in \hat{\partial}_\delta f(x^\circ)$.

Example 1 Take $f(x) = x$, if $x < 0$, and $f(x) = x^2$ otherwise. One has $\partial f(0) = \emptyset$, $0 \notin \hat{\partial}_\delta f(0)$ for any $\delta > 0$, while $\eta[f](0) = 0$.

Fortunately (8) happens to be closely related to (3) and (5).

Sup-d-stationarity and sup-η-stationarity as well as the corresponding regularity concepts can be defined in a similar way.

Formulation

Relations Between the "Elementary" Constants

Proposition 1 *The following assertions hold true:*
(i) $\tau[f](x^\circ) \le \theta[f](x^\circ)$;
(ii) *If* $\theta_\rho[f](x^\circ) = 0$ *for some* $\rho > 0$, *then* $\tau[f](x^\circ) = \theta[f](x^\circ) = 0$.

Proposition 1 (i) implies the relations between the corresponding stationarity and regularity concepts:
- inf-τ-stationarity $\Rightarrow$ inf-θ-stationarity;
- inf-θ-regularity $\Rightarrow$ inf-τ-regularity.

Proposition 1 (ii) means that at a point of local minimum a function is both inf-τ-stationary and inf-θ-stationary.

Inequality (i) in Proposition 1 can be strict even for functions from $\mathbb{R}$ to $\mathbb{R}$.

Example 2 Take $f(x) = -|x|$, if $|x| = 1/2^n$, $n = 1, 2, \ldots$, and $f(x) = 0$ otherwise. Obviously $\tau[f](0) = -1$. At the same time, for any $\rho \in \Xi_n = \{\rho\colon 1/2^n \le \rho < 1/2^{n-1}\}$ one has $\theta_\rho[f](0) = -1/2^n$ and

$$\sup_{\rho\in\Xi_n} \frac{\theta_\rho[f](0)}{\rho} = \frac{-1/2^n}{1/2^{n-1}} = -\frac{1}{2}\,.$$

Thus, $\theta[f](0) = -1/2$.

It is possible to modify the above example to make $\theta[f](0)$ equal zero.

Example 3 Take $f(x) = -|x|$, if $|x| = 1/n^n$, $n = 1, 2, \ldots$, and $f(x) = 0$ otherwise. One still has $\tau[f](0) = -1$ while $\theta[f](0) = 0$.

Thus, in the above example f is inf-τ-regular at 0 while being inf-θ-stationary at this point.

It is possible to modify the example further to make f continuous and even differentiable near 0 (but not strictly differentiable!) while keeping the inequality (i) in Proposition 1 strict.

Relations Between the "Strict" Constants

The relations between the elementary constants and their "strict" counterparts, as well as between the two "strict" constants, are given by the following theorem.

Theorem 1 *The following assertions hold true:*
(i) $\hat{\theta}[f](x^\circ) \ge \limsup_{x\xrightarrow{f}x^\circ} \theta[f](x)$,

(ii) $\hat{\tau}[f](x^\circ) = \limsup_{x \xrightarrow{f} x^\circ} \tau[f](x)$;

(iii) $\hat{\tau}[f](x^\circ) \le \hat{\theta}[f](x^\circ)$;

(iv) *If X is complete and f is lower semicontinuous near x°, then* $\hat{\tau}[f](x^\circ) = \hat{\theta}[f](x^\circ)$.

Parts (i) and (ii) of Theorem 1 imply the inequalities

$$\theta[f](x^\circ) \le \hat{\theta}[f](x^\circ)\,, \quad \tau[f](x^\circ) \le \hat{\tau}[f](x^\circ)\,,$$

and both of them can be strict.

Example 4 Take the function f from Example 1. Evidently, f attains a local minimum at $x_n = 1/2^n$ for any $n = 1, 2, \ldots$, and consequently, $\theta_\rho[f](x_n) = 0$ for some $\rho > 0$. It follows from Proposition 1 (ii) that $\tau[f](x_n) = \theta[f](x_n) = 0$. Consequently, $\hat{\tau}[f](0) = \hat{\theta}[f](0) = 0$. Recall that $\tau[f](0) = -1$ and $\theta[f](0) = -1/2$.

Inequalities (i) and (iii) in Theorem 1 can be strict, too.

Example 5 Define the function $f : \mathbb{R} \to \mathbb{R}$ in the following way: $f(x) = x$ if $x \le 0$, $f(x) = x - 1/n$ if $1/n < x \le 1/(n-1)$, $n = 2, 3, \ldots$, $f(x) = x - 1/2$ if $x > 1/2$. It is easy to see that $\theta[f](x) = \tau[f](x) = -1$ for any $x \in \mathbb{R}$. Then $\hat{\tau}[f](0) = -1$. On the other hand, take $x_n = 1/n + 1/n^2$, $\rho_n = 1/n$, $n = 1, 2, \ldots$ Then $f(x_n) = 1/n^2$, and consequently, $\theta_{\rho_n}[f](x_n) \ge -1/n^2$. It follows immediately that $\hat{\theta}[f](0) = 0$.

Due to part (iv) of Theorem 1, in the case of a lower semicontinuous function on a complete metric space two weak stationarity concepts as well as two strong regularity concepts coincide and the prefixes θ and τ can be omitted.

Corollary 1 *The following assertions hold true:*

(i) *Inf-θ-stationarity $\Rightarrow$ weak inf-θ-stationarity;*
strong inf-θ-regularity $\Rightarrow$ inf-θ-regularity;

(ii) *Inf-τ-stationarity $\Rightarrow$ weak inf-τ-stationarity;*
strong inf-τ-regularity $\Rightarrow$ inf-τ-regularity;

(iii) *Weak inf-τ-stationarity $\Rightarrow$ weak inf-θ-stationarity;*
strong inf-θ-regularity $\Rightarrow$ strong inf-τ-regularity;

(iv) *If X is complete and f is lower semicontinuous near x°, then*
weak inf-τ-stationarity $\Leftrightarrow$ weak inf-θ-stationarity;
strong inf-θ-regularity $\Leftrightarrow$ strong inf-τ-regularity.

The next "fuzzy" characterization of weak inf-τ-stationarity can be convenient for applications. It follows directly from definition (5).

Proposition 2 *f is weakly inf-τ-stationary at x° if and only if for any $\varepsilon > 0$ there exists an $x \in B_\varepsilon(x^\circ)$ such that $|f(x) - f(x^\circ)| \le \varepsilon$ and*

$$f(u) + \varepsilon d(u, x) \ge f(x) \quad \text{for all } u \text{ near } x\,. \tag{9}$$

Remark 2 A point x satisfying (9) is referred to in [12] (see also [6]) as a *local Ekeland point* of f (with factor ε). If all the conditions in Proposition 2 are satisfied, then x° is said to be a *stationary point* of f *with respect to minimization* [12]. Thus, stationarity with respect to minimization is equivalent to weak inf-τ-stationarity and, in the case of a lower semicontinuous function on a complete metric space, also to weak inf-θ-stationarity.

Relations Between the Primal and Dual Constants

Henceforth X is assumed to be a normed linear space. The next assertion is straightforward and has already been mentioned in the previous section.

Proposition 3

(i) *inf-τ-stationarity $\Leftrightarrow$ weak inf-d-stationarity;*

(ii) *inf-τ-regularity $\Leftrightarrow$ inf-d-regularity.*

Remark 3 Due to Propositions 1 and 3 the inclusion $0 \in \partial f(x^\circ)$ is sufficient for inf-θ-stationarity of f at x°. The opposite implication is not true in general (see Examples 2 and 3).

In what follows f is assumed to be lower semicontinuous near x°.

Theorem 2

(i) $\hat{\theta}[f](x^\circ) + \eta[f](x^\circ) \ge 0$.

(ii) *If X is Asplund, then*

$$\frac{\hat{\theta}[f](x^\circ)}{[1 + \hat{\theta}[f](x^\circ)]_+} + \eta[f](x^\circ) \le 0\,.$$

This theorem follows from [10], Theorem 2. The first part of the theorem is elementary. The proof of the second part is based on the application of the two fundamental results of variational analysis: the *Ekeland variational principle* [2] and the *fuzzy sum rule* due to Fabian [3].

Thus, in an Asplund space the constants $\hat{\theta}[f](x^\circ)$ and $\eta[f](x^\circ)$ can be zero or nonzero only simultaneously. Recall that a Banach space is called *Asplund* (see [4,13,14]) if any continuous convex function on it

is Fréchet differentiable on a dense G_δ subset. Note that in a Banach space $\hat{\theta}[f](x^\circ) = \hat{\tau}[f](x^\circ)$ due to Theorem 1.

Corollary 2

(i) *inf-η-stationarity $\Rightarrow$ weak inf-θ-stationarity;*
(ii) *strong inf-θ-regularity $\Rightarrow$ inf-η-regularity;*
(iii) *If X is Asplund, then*
weak inf-θ-stationarity $\Leftrightarrow$ weak inf-τ-stationarity $\Leftrightarrow$ inf-η-stationarity;
strong inf-θ-regularity $\Leftrightarrow$ strong inf-τ-regularity $\Leftrightarrow$ inf-η-regularity.

Differentiable Functions

The constants and corresponding stationarity/regularity concepts defined above take quite a traditional form when the function is assumed differentiable or convex. Fortunately, the number of different constants and concepts reduces significantly.

Theorem 3 *If f is Fréchet differentiable at x° with the derivative $\nabla f(x^\circ)$, then*

$$\theta[f](x^\circ) = \tau[f](x^\circ) = -\theta^+[f](x^\circ) = -\tau^+[f](x^\circ) = -||\nabla f(x^\circ)|| .$$

If, additionally, the derivative is strict, then

$$\hat{\theta}[f](x^\circ) = \hat{\tau}[f](x^\circ) = -\hat{\theta}^+[f](x^\circ) = -\hat{\tau}^+[f](x^\circ) = -||\nabla f(x^\circ)|| .$$

Recall that f is called *strictly differentiable* [13,15] at x° (with the derivative $\nabla f(x^\circ)$) if

$$\lim_{x\to x^\circ,\ u\to x^\circ} \frac{f(u) - f(x) - \langle \nabla f(x^\circ), u - x\rangle}{||u - x||} = 0.$$

This condition is stronger than the traditional Fréchet differentiability. Thus, condition $\nabla f(x^\circ) \neq 0$ does not guarantee strong regularity in the sense of Definition 1 (or Definition 2) unless f is strictly differentiable at x°.

Example 6 Take $f(x) = x + x^2 \sin(1/x)$, if $x \neq 0$ and $f(0) = 0$. This function is everywhere Fréchet differentiable and $\nabla f(0) = 1$. Thus, f is regular at zero. At the same time $\hat{\tau}[f](0) = \hat{\tau}^+[f](0) = 0$: there exists a sequence $x_k \to 0$ such that $\nabla f(x_k) \to 0$, and the assertion follows from Theorem 1, part (ii). Consequently, f is both weakly inf-stationary and weakly sup-stationary at zero.

Corollary 3 *If f is Fréchet differentiable at x° with the derivative $\nabla f(x^\circ)$, then the following conditions are equivalent:*
(i) *f is inf-θ-stationary at x°;*
(ii) *f is inf-τ-stationary at x°;*
(iii) *f is θ-stationary at x°;*
(iv) *f is τ-stationary at x°;*
(v) $\nabla f(x^\circ) = 0$.
If, additionally, the derivative is strict, then the above conditions are also equivalent to the following ones:
(vi) *f is weakly inf-θ-stationary at x°;*
(vii) *f is weakly inf-τ-stationary at x°;*
(viii) *f is weakly θ-stationary at x°;*
(xi) *f is weakly τ-stationary at x°.*

Remark 4 Stationarity and weak stationarity in the above corollary can be replaced with regularity and strong regularity, respectively, if one replaces the equality in (v) with the inequality $\nabla f(x^\circ) \neq 0$.

Convex Functions

In the convex case, as one might expect, all versions of inf-stationarity coincide and appear to be equivalent to just (local and global) minimality.

Theorem 4 *Let f be convex.*
(i) *If $\theta_\rho[f](x^\circ) < 0$ for some $\rho > 0$, then $\theta_\rho[f](x^\circ) < 0$ for all $\rho > 0$.*
(ii) *The functions $\rho \to \theta_\rho[f](x^\circ)/\rho$ and $\rho \to \theta^+_\rho[f](x^\circ)/\rho$ are nondecreasing on $\mathbb{R}_+\backslash\{0\}$.*
(iii) *The following equalities hold true:*

$$\hat{\theta}[f](x^\circ) = \hat{\tau}[f](x^\circ) = \theta[f](x^\circ) = \tau[f](x^\circ) = \inf_{\rho>0} \frac{\theta_\rho[f](x^\circ)}{\rho} = \inf_{x\neq x^\circ} \frac{[f(x) - f(x^\circ)]_-}{||x - x^\circ||},$$

$$\theta^+[f](x^\circ) = \tau^+[f](x^\circ) = \inf_{\rho>0} \frac{\theta^+_\rho[f](x^\circ)}{\rho} = \inf_{\rho>0} \sup_{||x-x^\circ||=\rho} \frac{[f(x) - f(x^\circ)]_+}{\rho}.$$

(iv) $\tau[f](x^\circ) + \tau^+[f](x^\circ) \geq 0$.
(v) $\hat{\tau}[f](x^\circ) + \hat{\tau}^+[f](x^\circ) \geq 0$.

(vi) *If* $\tau[f](x^\circ) + \tau^+[f](x^\circ) = 0$ *and* $\{x_k\} \subset X$ *is a sequence defining* $\tau[f](x^\circ)$*, that is* $x_k \to 0$ *and*

$$\tau[f](x^\circ) = \lim_{k\to\infty} \frac{f(x^\circ + x_k) - f(x^\circ)}{||x_k||}$$

then $\{-x_k\}$ *is a sequence defining* $\tau^+[f](x^\circ)$*:*

$$\tau^+[f](x^\circ) = \lim_{k\to\infty} \frac{f(x^\circ - x_k) - f(x^\circ)}{||x_k||}.$$

Corollary 4 *If* f *is convex, then the following conditions are equivalent:*

(i) f *attains a global minimum at* x°*;*
(ii) f *attains a local minimum at* x°*;*
(iii) f *is inf-*θ*-stationary at* x°*;*
(iv) f *is inf-*τ*-stationary at* x°*;*
(v) f *is* θ*-stationary at* x°*;*
(vi) f *is* τ*-stationary at* x°*;*
(vii) f *is weakly inf-*θ*-stationary at* x°*;*
(viii) f *is weakly inf-*τ*-stationary at* x°*;*
(ix) f *is weakly stationary at* x°*.*

Remark 5 The conditions $\tau[f](x^\circ) = \tau^+[f](x^\circ) = 0$ imply Fréchet differentiability of f at x° (with the derivative equal to zero). The weaker condition $\tau[f](x^\circ) + \tau^+[f](x^\circ) = 0$ in Theorem 4, (vi) implies linearity of the directional derivative of f along the direction of steepest descent (if the latter exists) with the opposite direction being automatically the direction of steepest ascent. This condition is not sufficient for differentiability of f at x° unless $X = \mathbb{R}$. Note also that the direction opposite to the direction of steepest ascent does not need to be a direction of steepest descent.

Example 7 ([1]) Take the function $f(x, y) = \max(x, y)$ on $\mathbb{R}^2$ and assume that $\mathbb{R}^2$ is equipped with the max type norm: $||x, y|| = \max(|x|, |y|)$. f is obviously not differentiable at 0. At the same time $\tau[f](0) = -1$, $\tau^+[f](0) = 1$. The vector $(-1, -1)$ defines the (unique) direction of steepest descent. The opposite vector $(1, 1)$ defines the direction of steepest ascent and f is linear along the line defined by these vectors. Note that the direction of steepest ascent is not unique. For instance, the vector $(1, 0)$ also defines the direction of steepest ascent, while the opposite vector does not define the direction of steepest descent and f is not linear along this line.

[1]The example was suggested by Alexander Rubinov (personal communication).

Remark 6 Stationarity and weak stationarity in assertions (iii)–(ix) of the above corollary can be replaced with regularity and strong regularity, respectively, if one replaces (i) and (ii) with the opposite assertions: x° is not a point of (local or global) minimum of f.

See also

► Nonsmooth Analysis: Fréchet Subdifferentials

References

1. De Giorgi E, Marino A, Tosques M (1980) Problemi di evoluzione in spazi metrici e curve di massima pendenza. Atti Accad. Nat. Lincei. Cl Sci Fiz Mat Natur 68:180–187
2. Ekeland I (1974) On the variational principle. J Math Anal Appl 47:324–353
3. Fabian M (1989) Subdifferentiability and trustworthiness in the light of a new variational principle of Borwein and Preiss. Acta Univ Carolinae 30:51–56
4. Fabian M (1997) Gâteaux Differentiability of Convex Functions and Topology. Weak Asplund Spaces. Canadian Mathematical Society Series of Monographs and Advanced Texts. Wiley, New York
5. Ioffe A D (2000) Metric regularity and subdifferential calculus. Russian Math Surveys 55:501–558
6. Klatte D, Kummer B (2002) Nonsmooth Equations in Optimization: Regularity, Calculus, Methods and Applications, vol 60 of Nonconvex Optimization and Its Applications. Kluwer, Dordrecht
7. Kruger AY (1996) On calculus of strict ε-semidifferentials. Dokl Akad Nauk Belarusi 40(4):34–39 (in Russian)
8. Kruger AY (2002) Strict (ε, δ)-semidifferentials and extremality conditions. Optimization 51:539–554
9. Kruger AY (2003) On Fréchet subdifferentials. J Math Sci (NY) 116:3:3325–3358. Optimization and related topics, 3
10. Kruger AY (2004) Weak stationarity: eliminating the gap between necessary and sufficient conditions. Optimization 53(2):147–164
11. Kruger AY (2006) Stationarity and regularity of real-valued functions. Appl Comput Math 5(1):79–93
12. Kummer B (2000) Inverse functions of pseudo regular mappings and regularity conditions. Math Program Ser B 88:313–339
13. Mordukhovich BS (2006) Variational Analysis and Generalized Differentiation. I Basic theory, vol 330 of Grundlehren der Mathematischen Wissenschaften (Fundamental Principles of Mathematical Sciences). Springer, Berlin
14. Phelps RR (1993) Convex Functions, Monotone Operators and Differentiability, 2nd edn. Lecture Notes in Mathematics, vol 1364. Springer, Berlin
15. Rockafellar RT, Wets RJ-B (1998) Variational Analysis. Springer, Berlin

Nonsmooth Optimization Approach to Clustering

ADIL BAGIROV
Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Victoria, Australia

MSC2000: 90C26, 90C56, 90C90

Article Outline

Introduction
Formulation
Methods
Modified Global k-means Algorithm
Nonsmooth Optimization Clustering Algorithm
Solving Optimization Problems
Conclusions
References

Introduction

Clustering is the *unsupervised* classification of the patterns. Cluster analysis deals with the problems of organization of a collection of patterns into clusters based on similarity. It has found many applications, including information retrieval, document extraction, image segmentation etc.

In cluster analysis we assume that we have been given a set A of a finite number of points of n-dimensional space $\mathbb{R}^n$, that is

$$A = \{a^1, \dots, a^m\}, \text{ where } a^i \in \mathbb{R}^n, \quad i = 1, \dots, m\,.$$

The subject of cluster analysis is the partition of the set A into a given number q of overlapping or disjoint subsets C_i, $i = 1, \dots, q$ with respect to predefined criteria such that

$$A = \bigcup_{i=1}^{q} C_i\,.$$

The sets $C_i, i = 1, \dots, q$ are called clusters. The clustering problem is said to be *hard clustering* if every data point belongs to one and only one cluster. Unlike hard clustering in the *fuzzy clustering* problem the clusters are allowed to overlap and instances have degrees of appearance in each cluster. In this paper we will exclusively consider the hard unconstrained clustering problem, that is we additionally assume that

$$C_i \bigcap C_k = \emptyset, \quad \forall i, k = 1, \dots, q,\ i \neq k\,.$$

and no constraints are imposed on the clusters C_i, $i = 1, \dots, q$. Thus every point $a \in A$ is contained in exactly one and only one set C_i.

Each cluster C_i can be identified by its center (or centroid). Then the clustering problem can be reduced to the following optimization problem (see [12,26]):

$$\text{minimize } \varphi(C, x) = \frac{1}{m} \sum_{i=1}^{q} \sum_{a \in C_i} \|x^i - a\|^2 \qquad (1)$$
$$\text{subject to } C \in \overline{C}, \quad x = (x^1, \dots, x^q) \in \mathbb{R}^{n \times q}$$

where $\|\cdot\|$ denotes the Euclidean norm, $C = \{C_1, \dots, C_q\}$ is a set of clusters, $\overline{C}$ is a set of all possible q-partitions of the set A, x^i is the center of the cluster C_i, $i = 1, \dots, q$:

$$x^i = \frac{1}{|C_i|} \sum_{a \in C_i} a\,,$$

and $|C_i|$ is a cardinality of the set C_i, $i = 1, \dots, q$. The problem (1) is also known as the minimum sum-of-squares clustering. The combinatorial formulation (1) of the minimum sum-of-squares clustering is not suitable for direct application of mathematical programming techniques. The problem (1) can be rewritten as the following mathematical programming problem:

$$\text{minimize } \psi(x, w) = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{q} w_{ij} \|x^j - a^i\|^2 \qquad (2)$$
$$\text{subject to } \sum_{j=1}^{q} w_{ij} = 1, \quad i = 1, \dots, m\,,$$

and

$$w_{ij} \in \{0, 1\}, \quad i = 1, \dots, m,\ j = 1, \dots, q\,.$$

Here

$$x^j = \frac{\sum_{i=1}^{m} w_{ij} a^i}{\sum_{i=1}^{m} w_{ij}}, \quad j = 1, \dots, q$$

and w_{ij} is the association weight of pattern a_i with cluster j (to be found), given by

$$w_{ij} = \begin{cases} 1 & \text{if pattern } a^i \text{ is allocated to cluster } j \\ 0 & \text{otherwise}\,. \end{cases}$$

w is an $m \times q$ matrix.

There exist different approaches to clustering including agglomerative and divisive hierarchical clustering algorithms as well as algorithms based on mathematical programming techniques. Descriptions of many of these algorithms can be found, for example, in [15,20,21,26].

Problem (2) is a global optimization problem. Therefore different algorithms of mathematical programming can be applied to solve this problem. Some review of these algorithms can be found in [16]. However, most of these algorithms are applicable for clustering on small data sets.

Different heuristics can be used for solving clustering problems on large data sets and k-means is one such algorithm. Different versions of this algorithm have been studied by many authors (see [26]). This is a fast algorithm. k-means gives good results when there are few clusters but deteriorates when there are many [16]. This algorithm achieves a local minimum of problem (1) (see [24]), however results of numerical experiments presented, for example, in [19] show that the best clustering found with k-means may be more than 50% worse than the best known one.

Much better results have been obtained with metaheuristics, such as simulated annealing, tabu search and genetic algorithms [23]. The simulated annealing approaches to clustering have been studied, for example, in [13,25,27]. Application of tabu search methods for solving clustering problem is studied in [1]. Genetic algorithms for clustering have been described in [23]. The results of numerical experiments, presented in paper [2] show that even for small problems of cluster analysis when the number of entities $m \leq 100$ and the number of clusters $q \leq 5$ these algorithms take 500–700 (sometimes several thousands) times more CPU time than the k-means algorithms. For relatively large data sets one can expect that this difference will increase. This makes metaheuristic algorithms of global optimization ineffective for solving many clustering problems.

The paper [18] develops variable neighborhood search algorithm and the paper [17] presents j-means algorithm which extends k-means by adding a jump move. The global k-means heuristic, which is an incremental approach to minimum sum-of-squares clustering problem, is developed in [22]. The incremental approach is also studied in the paper [19]. Results of numerical experiments presented show the high effectiveness of these algorithms for many clustering problems.

As mentioned above the problem (2) is the global optimization problem and the objective function in this problem is multimodal. However, global optimization techniques are highly time-consuming for solving many clustering problems. It is very important, therefore, to develop clustering algorithms that compute near global minimizers of the objective function. We propose the clustering algorithms based on nonsmooth optimization approach. The algorithms provide the capability of calculating clusters step-by-step, gradually increasing the number of data clusters until termination conditions are met, that is it allows one to calculate as many cluster as a data set contains with respect to some tolerance.

Formulation

The problems (1) and (2) can be reformulated as the following mathematical programming problem [7,8,12]

$$\begin{aligned} &\text{minimize } f(x^1,\ldots,x^q) \\ &\quad \text{subject to } x = (x^1,\ldots,x^q) \in \mathbb{R}^{n\times q}, \end{aligned} \tag{3}$$

where

$$f(x^1,\ldots,x^q) = \frac{1}{m}\sum_{i=1}^{m} \min_{j=1,\ldots,q} \|x^j - a^i\|^2 . \tag{4}$$

It is shown in [12] that problems (2) and (3) are equivalent. However, there are some differences between these two formulations:

- The number of variables in problem (2) is $(m+n)\times q$ whereas in problem (3) this number is only $n \times q$ and the number of variables does not depend on the number of instances. It should be noted that in many real-world databases the number of instances m is substantially greater than the number of attributes n.
- In the hard clustering problem (2) the coefficients w_{ij} are integer, that is the problem (2) contains both integer and continuous variables. In the nonsmooth optimization formulation variables are only continuous.
- Nonsmooth optimization formulation of the clustering problem allows one to easily consider different similarity measures.

All these circumstances can be considered as advantages of the nonsmooth optimization formulation (3).

If $q > 1$, the objective function (4) in problem (3) is nonconvex and nonsmooth. If the number q of clusters and the number n of attributes are large, we have a large-scale global optimization problem. Moreover, the form of the objective function in this problem is complex enough not to be amenable to the direct application of general purpose global optimization methods. Therefore, in order to ensure the practicality of the nonsmooth optimization approach to clustering, proper identification and use of local optimization methods is very important. Clearly, such an approach does not guarantee a globally optimal solution to problem (3). On the other hand, this approach provides a "near" global minimum of the objective function that, in turn, provides a good enough clustering description of the data set under consideration.

Note also that a meaningful choice of the number of clusters is very important for clustering analysis. It is difficult to define *a priori* how many clusters represent the set A under consideration. In order to avoid this difficulty, a step-by-step calculation of clusters is implemented in algorithms discussed in the next section.

Methods

In this section we will describe two incremental clustering algorithms. Both algorithms are based on nonsmooth optimization approach to clustering. In the first algorithm nonsmooth optimization approach is used to find starting points for k-means algorithm. This algorithm is a modification of the global k-means algorithm proposed in [22]. The second algorithm is an optimization based clustering algorithm.

Modified Global *k*-Means Algorithm

k-means algorithm and its different variations are known to be fast algorithms for clustering and they are applicable to large data sets. In this subsection we propose a new version of k-means algorithm: the modified global k-means algorithm, which in its turn is the modification of the global k-means algorithm. First we briefly describe k-means and the global k-means algorithms.

k-means algorithm proceeds as follows

Step 1. Choose a seed solution consisting of k centroids (not necessarily belonging to A).

Step 2. Allocate data points $a^i \in A$ to its closest centroid and obtain k-partition of A.

Step 3. Recompute centroids for this new partition and go to to Step 2 until no more data points change their clusters.

Nonsmooth Optimization Approach to Clustering, Algorithm 1
***k*-means algorithm**

Step 1. Compute the centroid x^1 of the set A:

$$x^1 = \frac{1}{m}\sum_{i=1}^{m} a^i$$

and set $k = 1$.

Step 2. Set $k = k + 1$ and consider the centers $x^1, x^2, \ldots, x^{k-1}$ from the previous iteration.

Step 3. Consider each point a of A as a starting point for the k-th cluster center, thus obtaining m initial solutions with k points $(x^1, x^2, \ldots, x^{k-1}, a)$; apply k-means algorithm starting from each of them; keep the best k-partition obtained and its center $(x^1, x^2, \ldots, x^{k-1}, x^k)$.

Step 4. If $k = q$ stop, otherwise go to Step 2.

Nonsmooth Optimization Approach to Clustering, Algorithm 2
The global *k*-means algorithm

The effectiveness of this algorithm highly depends on a starting point. It converges only to a local solution which can significantly differ from the global one in large data sets.

The global k-means algorithm proposed in [22] is the modification of k-means algorithm and it computes clusters successively that is in order to compute k-th cluster centroid this algorithm uses centroids of $k - 1$ clusters from the previous iteration. To compute $q \leq m$ clusters this algorithm proceeds as follows.

This version of the algorithm is not applicable for clustering on middle sized and large data sets. Two

procedures were introduced to reduce its complexity (see [22]). We mention here only one of them. Let

$$d_{k-1}^i = \min\left\{\|x^1 - a^i\|^2, \ldots, \|x^{k-1} - a^i\|^2\right\}. \quad (5)$$

For each $a^i \in A$ we compute:

$$r_i = \sum_{j=1}^{m} \min\{0, \|a^i - a^j\|^2 - d_{k-1}^j\}$$

and we take the data point $a^l \in A$ for which

$$l = \operatorname{argmin}_{i=1,\ldots,m} r_i.$$

Then k-means algorithm is applied starting from the point $(x^1, x^2, \ldots, x^{k-1}, a^l)$ to find k cluster centers.

Now we will describe a new version of the global k-means algorithm where a starting point for k-th cluster center is computed using nonsmooth optimization approach. Let us consider the problem of finding k cluster center assuming that the centers $x^1, \ldots, x^{k-1}$ for $(k-1)$-clustering problem are known. We introduce the following function:

$$\bar{f}^k(y) = \frac{1}{m}\sum_{i=1}^{m} \min\left\{d_{k-1}^i, \|y - a^i\|^2\right\} \quad (6)$$

where $y \in \mathbb{R}^n$ stands for k-th cluster center and d_{k-1}^i is defined as in (5). Consider a set

$$\overline{D} = \left\{y \in \mathbb{R}^n : \|y - a^i\|^2 \geq d_{k-1}^i\right\}.$$

This is a set where the distance from any point y to any data point is no less than the distance between this data point and its cluster center. We also consider the following set

$$D_0 = \mathbb{R}^n \setminus \overline{D} \equiv \Big\{y \in \mathbb{R}^n : \exists I \subset \{1, \ldots, m\}, I \neq \emptyset : \|y - a^i\| < d_{k-1}^i \forall i \in I\Big\}.$$

The function $\bar{f}^k$ is a constant on the set $\overline{D}$ and its value over this set is

$$\bar{f}^k(y) = d_0 \equiv \frac{1}{m}\sum_{i=1}^{m} d_{k-1}^i, \ \forall y \in \overline{D}.$$

It is clear that $x^j \in \overline{D}$ for all $j = 1, \ldots, k-1$ and $a^i \in D_0$ for all $a^i \in A$, $a^i \neq x^j$, $j = 1, \ldots, k-1$. It is also clear that $f(y) < d_0$ for all $y \in D_0$.

Step 1. For any $a^i \in D_0 \bigcap A$ calculate the set $S_2(a^i)$, the centroid c^i of this set and calculate the value $\bar{f}^k(c^i)$ of the function $\bar{f}^k$ at this point.

Step 2. Compute

$$\bar{f}_{min}^k = \min_{a^i \in D_0 \bigcap A} \bar{f}^k(c^i),$$

$$a^j = \operatorname{argmin}_{a^i \in D_0 \bigcap A} \bar{f}^k(c^i).$$

and the corresponding center c^j.

Step 3. Compute the set $S_2(c^j)$ and its centroid.

Step 4. Recompute the set $S_2(c^j)$ and its centroid until no more data points escape this set or return to this set.

Nonsmooth Optimization Approach to Clustering, Algorithm 3
An algorithm for finding the initial point

Any point $y \in D_0$ can be taken as a starting point for the k-th cluster center. Probably more preferably among them is a global minimizer of the function $\bar{f}^k$. This function is a nonconvex and nonsmooth and its minimization is difficult task. We consider a scheme for finding its local minimizer.

For any $y \in D_0$ we consider the following sets:

$$S_1(y) = \left\{a^i \in A : \|y - a^i\|^2 = d_{k-1}^i\right\},$$
$$S_2(y) = \left\{a^i \in A : \|y - a^i\|^2 < d_{k-1}^i\right\},$$
$$S_3(y) = \left\{a^i \in A : \|y - a^i\|^2 > d_{k-1}^i\right\}.$$

Since $y \in D_0$ the set $S_2(y) \neq \emptyset$. We suggest the following algorithm to find a starting point for the k-th cluster center.

Now we can describe the modified global k-means algorithm.

It is clear that $f^{k*} \geq 0$ for all $k \geq 1$ and the sequence $\{f^{k*}\}$ is decreasing, that is,

$$f^{k+1,*} \leq f^{k,*} \text{ for all } k \geq 1.$$

The latter implies that after $\bar{k} > 0$ iterations the stopping criterion in Step 4 will be satisfied.

Step 1. (Initialization). Select a tolerance $\varepsilon > 0$. Calculate the centroid $x^{1*} \in \mathbb{R}^n$ of the set A. Let f^{1*} be the corresponding value of the objective function (4). Set $k = 1$.

Step 2. (Computation of the next cluster center). Let $x^{1*}, \ldots, x^{k*}$ be the cluster centers for k clustering problem. Apply Algorithm 3 to find an initial point $y^{k+1,0} \in \mathbb{R}^n$ for the $k+1$-th cluster center.

Step 3. (Refinement of all cluster centers). Take $x^{k+1,0} = (x^{1*}, \ldots, x^{k*}, y^{k+1,0})$ as a new starting point, apply k-means algorithm to solve clustering problem for $q = k+1$. Let $x^{1*}, \ldots, x^{k+1,*}$ be a solution to this problem and $f^{k+1,*}$ be the corresponding value of the objective function (4).

Step 4. (Stopping criterion). If

$$\frac{f^{k*} - f^{k+1,*}}{f^{1*}} < \varepsilon$$

then stop, otherwise set $k = k+1$ and go to Step 2.

Nonsmooth Optimization Approach to Clustering, Algorithm 4
Modified global *k*-means algorithm

Nonsmooth Optimization Clustering Algorithm

In this subsection we propose an algorithm for clustering where nonsmooth optimization techniques are used to find a starting point for the k cluster center and to solve k-clustering problems.

It is clear that $f^{k*} \geq 0$ for all $k \geq 1$ and the sequence $\{f^{k*}\}$ is decreasing that is,

$$f^{k+1,*} \leq f^{k,*} \text{ for all } k \geq 1.$$

The latter implies that after $\bar{k} > 0$ iterations the stopping criterion in Step 4 will be satisfied.

Remark 1 One of the important questions when one tries to apply Algorithms 4 and 5 is the choice of the tolerance $\varepsilon > 0$. Large values of ε can result in the appearance of large clusters whereas small values can produce small and artificial clusters.

Remark 2 Algorithms 2, 4 and 5 are incremental clustering algorithms. Main difference between Algorithms 2 and 4 is in the way they compute starting points for the next cluster center. Algorithm 2 uses data points whereas Algorithm 4 uses local minimizers of the function $\bar{f}^k$. Algorithm 5 uses nonsmooth optimization techniques for the finding of both starting points and k-partition of a data set.

Step 1. (Initialization). Select a tolerance $\varepsilon > 0$. Calculate the centroid $x^{1*} \in \mathbb{R}^n$ of the set A. Let f^{1*} be the corresponding value of the objective function (4). Set $k = 1$.

Step 2. (Computation of the next cluster center). Select a point $y^0 \in \mathbb{R}^n$ and solve the following minimization problem:

$$\text{minimize } \bar{f}^k(y) \text{ subject to } y \in \mathbb{R}^n \tag{7}$$

where $\bar{f}^k$ is defined by (6).

Step 3. (Refinement of all cluster centers). Let $\bar{y}^{k+1,*}$ be a solution to problem (7). Take $x^{k+1,0} = (x^{1*}, \ldots, x^{k*}, \bar{y}^{k+1,*})$ as a new starting point and solve the following minimization problem:

$$\text{minimize } f^{k+1}(x) \text{ subject to}$$

$$x = (x^1, \ldots, x^{k+1}) \in \mathbb{R}^{n \times (k+1)} \tag{8}$$

where

$$f^{k+1}(a) = \frac{1}{m} \sum_{i=1}^{m} \min_{j=1,\ldots,k+1} \|x^j - a^i\|^2 .$$

Step 4. (Stopping criterion). Let $x^{k+1,*}$ be a solution to the problem (8) and $f^{k+1,*}$ be the corresponding value of the objective function. If

$$\frac{f^{k*} - f^{k+1,*}}{f^{1*}} < \varepsilon$$

then stop, otherwise set $k = k+1$ and go to Step 2.

Nonsmooth Optimization Approach to Clustering, Algorithm 5
Nonsmooth optimization clustering algorithm

Remark 3 Computational results on gene expression data sets presented in [6] demonstrate that Algorithm 4 is more efficient than Algorithm 2. However, the former requires more computational time.

Remark 4 Results of numerical experiments presented in [11] demonstrate that Algorithm 5 is efficient for solving large scale clustering problems in a reasonable CPU time. Moreover, its success to locate global solutions is higher than that for Algorithms 2 and 4. However, this algorithm requires significantly more CPU time than other algorithms.

Solving Optimization Problems

The objective functions in problems (7) and (8) are nonsmooth and nonconvex. If the number of attributes and clusters are large then the problem (8) is large scale problem. Both objective functions are non-regular and the computation of even one their subgradient may become very difficult problem (for the definition of non-regular function, see [14]). Therefore, subgradient-based methods are not always efficient for solving problems (7) and (8). We use the discrete gradient method to solve these problems [3,4,5]. This is a derivative free method.

The objective functions in problems (7) and (8) are piecewise partially separable (for the definition of piecewise partially separable functions, see [9]). The discrete gradient method was modified taking into account this special structure of the objective functions. This modified discrete gradient method is described in [10].

Conclusions

In this paper we discussed a nonsmooth optimization approach to clustering problems. Many clustering problems are large scale global optimization problems. The nonsmooth optimization approach allows one to significantly reduce the number of variables in this problem. It also can easily handle different similarity measures.

We introduced two algorithms based on the nonsmooth optimization approach. Both algorithms are incremental clustering algorithms. As these algorithms compute clusters step by step, they allow the decision maker to easily vary the number of clusters according to the criteria suggested by the nature of the decision making situation not incurring the obvious costs of the increased complexity of the solution procedure. The suggested approach utilizes a stopping criterion that prevents the appearance of small and artificial clusters. Nonsmooth optimization problems from cluster analysis have special structure which allows one to design efficient algorithms for their solution.

References

1. Al-Sultan KS (1995) A tabu search approach to the clustering problem. Pattern Recognit 28(9):1443–1451
2. Al-Sultan KS, Khan MM (1996) Computational experience on four algorithms for the hard clustering problem. Pattern Recognit Lett 17:295–308
3. Bagirov AM (1999) Minimization methods for one class of nonsmooth functions and calculation of semi-equilibrium prices. In: Eberhard A et al (eds) Progress in Optimization: Contribution from Australasia. Kluwer, Dordrecht, pp 147–175
4. Bagirov AM (2002) A method for minimization of quasidifferentiable functions. Optim Method Softw 17(1):31–60
5. Bagirov AM (2003) Continuous subdifferential approximations and their applications. J Math Sci 115(5):2567–2609
6. Bagirov AM, Mardaneh K (2006) Modified global *k*-means algorithm for clustering in gene expression data sets. In: Proceedings of the 2006 Workshop on Intelligent Systems for Bioinformatics, Hobart, Australia, 4 Dec 2006, pp 23–28
7. Bagirov AM, Rubinov AM, Yearwood J (2001) Using global optimization to improve classification for medical diagnosis and prognosis. Top Heal Inf Manag 22:65–74
8. Bagirov AM, Rubinov AM, Yearwood J (2001) Global optimization approach to classification. Optim Eng 22:65–74
9. Bagirov AM, Ugon J (2006) Piecewise partially separable functions and a derivative-free method for large-scale nonsmooth optimization. J Glob Optim 35(2):163–195
10. Bagirov AM, Ugon J (2005) An algorithm for minimizing clustering functions. Optimization 54(4–5):351–368
11. Bagirov AM, Yearwood J (2006) A new nonsmooth optimisation algorithm for minimum sum-of-squares clustering problems. Eur J Oper Res 170(2):578–596
12. Bock HH (1974) Automatische Klassifikation. Vandenhoeck & Ruprecht, Göttingen
13. Brown DE, Entail CL (1992) A practical application of simulated annealing to the clustering problem. Pattern Recognit 25:401–412
14. Clarke FH (1983) Optimization and Nonsmooth Analysis. Wiley, New York
15. Dubes R, Jain AK (1976) Clustering techniques: the user's dilemma. Pattern Recognit 8:247–260
16. Hansen P, Jaumard B (1997) Cluster analysis and mathematical programming. Math Program 79(1–3):191–215
17. Hansen P, Mladenovic N (2001) *J*-means: a new heuristic for minimum sum-of-squares clustering. Pattern Recognit 4:405–413
18. Hansen P, Mladenovic N (2001) Variable neighborhood decomposition search. J Heuristic 7:335–350
19. Hansen P, Ngai E, Cheung BK, Mladenovic N (2005) Analysis of global *k*-means, an incremental heuristic for minimum sum-of-squares clustering. J Classif 22:287–310

20. Houkins DM, Muller MW, ten Krooden JA (1982) Cluster analysis. In: Topics in Applied Multivariate Analysis. Cambridge University Press, Cambridge
21. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
22. Likas A, Vlassis M, Verbeek J (2003) The global *k*-means clustering algorithm. Pattern Recognit 36:451–461
23. Reeves CR (ed) (1993) Modern Heuristic Techniques for Combinatorial Problems. Blackwell, London
24. Selim SZ, Ismail MA (1984) *K*-means-type algorithm: generalized convergence theorem and characterization of local optimality. IEEE Trans Pattern Anal Mach Intell 6:81–87
25. Selim SZ, Al-Sultan KS (1991) A simulated annealing algorithm for the clustering. Pattern Recognit 24(10):1003–1008
26. Spath H (1980) Cluster Analysis Algorithms. Ellis Horwood Limited, Chichester
27. Sun LX, Xie YL, Song XH, Wang JH, Yu RQ (1994) Cluster analysis by simulated annealing. Comput Chem 18:103–108

Nonsmooth and Smoothing Methods for Nonlinear Complementarity Problems and Variational Inequalities

NSM

Liqun Qi, Defeng Sun
School of Math., University New South Wales, Sydney, Australia

MSC2000: 90C33, 90C30

Article Outline

Keywords
Conclusion
See also
References

Keywords

Optimization; Nonsmooth methods; Smoothing methods; Variational inequalities; Complementarity problem

Given a nonempty closed set $X \subset \mathbf{R}^n$ and a mapping $F\colon \mathbf{R}^n \to \mathbf{R}^n$, assumed to be continuously differentiable, the *variational inequalities* (abbreviated: VI) are to find an $x^* \in X$ such that

$$(x - x^*)^\top F(x^*) \geq 0\,, \quad \forall x \in X\,. \tag{1}$$

When $X = \mathbf{R}^n_+$ (the positive orthant), (1) is equivalent to the *nonlinear complementarity problem* (abbreviated: NCP): Find $x^* \in \mathbf{R}^n$ such that

$$x^* \geq 0, \quad F(x^*) \geq 0, \quad x^{*\top} F(x^*) = 0\,. \tag{2}$$

Usually, X is represented by several inequalities and equalities. By considering the Karush–Kuhn–Tucker (KKT) conditions of (1) if necessary, X is assumed to be a closed convex subset of $\mathbf{R}^n$ here. It has been proved by B.C. Eaves in [9] that solving (1) is equivalent to finding a solution of the equation

$$H(x) := x - \Pi_X[x - F(x)] = 0\,, \tag{3}$$

where Π_X is the orthogonal projection onto X. When $X = \mathbf{R}^n_+$, (3) becomes

$$H(x) = \min(x, F(x)) = 0\,, \tag{4}$$

where the operation min is taken componentwisely. This means that finding a solution of NCP is equivalent to finding a root of (4). It also has been shown by A. Fischer in [10] that finding a solution of NCP is equivalent to finding a root of another equation, namely of:

$$H_i(x) := \phi(x_i, F_i(x)) = 0, \quad i = 1, \ldots, n\,, \tag{5}$$

where ϕ is called the *Fischer–Burmeister function* (FB function), defined in [10] as

$$\phi(a, b) = \sqrt{a^2 + b^2} - (a + b), \quad a, b \in \mathbf{R}\,.$$

Due to the nonsmoothness of the orthogonal projection operator Π_X, the min function and the FB function, the function H defined either in (3), in (4) or in (5) is, in general, not smooth (i. e. continuously differentiable) no matter how smooth F is. This prevents one from using classical Newton methods to find solutions of these (nonsmooth) equations.

Suppose that $H\colon \mathbf{R}^n \to \mathbf{R}^n$ is a locally Lipschitz function (the function H defined in either (3), (4) or (5) is a locally Lipschitz function) but is not necessarily smooth. By Rademacher's theorem, H is almost everywhere differentiable. Let

$$D_H = \{x\colon\ H \text{ is differentiable at } x\}\,.$$

Then the generalized Jacobian of H at x in the sense of F.H. Clarke [6] can be defined by

$$\partial H(x) = \operatorname{conv} \partial_B H(x) ,$$

where $\partial_B H(x)$ [20] is defined by

$$\partial_B H(x) = \left\{ \lim_{\substack{x^j \to x \\ x^j \in D_H}} H'(x^j) \right\} .$$

The *nonsmooth Newton method* for solving

$$H(x) = 0, \quad x \in \mathbf{R}^n , \tag{6}$$

can be defined as follows: Having the vector $x^k \in \mathbf{R}^n$, compute x^{k+1} by

$$x^{k+1} = x^k - V_k^{-1} H(x^k) , \tag{7}$$

where $V_k \in \partial H(x^k)$. The nonsmooth Newton method (7) reduces to the classical Newton method for a system of equations if H is continuously differentiable. The classical Newton method has the favorable feature that the sequence $\{x^k\}$ generated by (7) is locally superlinearly (quadratically) convergent to a solution x^* of $H(x) = 0$ if $H'(x^*)$ is nonsingular (and H' is Lipschitz continuous) [8,18]. However, in general the iterative method (7) is not convergent for nonsmooth equations (6). See [16] for a counterexample.

In order to establish some superlinearly convergent results for the nonsmooth Newton method (7), we use the concept of *semismoothness*. Let H be directionally differentiable at x. H is said to be *semismooth* at x if

$$Vd - H'(x; d) = o(\|d\|), \quad d \to 0 ,$$

and H is called strongly semismooth at x if

$$Vd - H'(x; d) = O(\|d\|^2), \quad d \to 0 ,$$

where $V \in \partial H(x + d)$. Semismoothness was originally introduced by R. Mifflin [17] for functionals. L. Qi and J. Sun [24] extended the concept of semismoothness to vector-valued functions. See [19] for several forms of semismooth equations. Using semismoothness, they [24] presented the following convergence theorem for the generalized Newton method (7):

Theorem 1 *Suppose that $H(x^*) = 0$ and that all $V \in \partial H(x^*)$ are nonsingular. Then the generalized Newton method (7) is Q-superlinearly convergent in a neighborhood of x^* if H is semismooth at x^*, and quadratically convergent if H is strongly semismooth at $x^{\wedge} *$.*

Note that the nonsingularity of $\partial H(x^*)$ in the above theorem is somewhat restrictive in some cases. Qi [20] presented a modified version of (7), which may be stated as follows

$$x^{k+1} = x^k - V_k^{-1} H(x^k) , \tag{8}$$

where $V_k \in \partial_B H(x^k)$. The difference of this version from (7) is that V_k is chosen from $\partial_B H(x^k)$ rather than the convex hull of $\partial_B H(x^k)$. Analogous to the above theorem, Qi [20] established the following result:

Theorem 2 *Suppose that $H(x^*) = 0$ and that all $V \in \partial_B H(x^*)$ are nonsingular. Then the generalized Newton method (8) is Q-superlinearly convergent in a neighborhood of x^* if H is semismooth at x^*, and quadratically convergent at x^* if H is strongly semismooth at x^*.*

In general, neither (7) nor (8) can be globalized because θ is not necessarily continuously differentiable, where for any $x \in \mathbf{R}^n$, $\theta(x) = \|H(x)\|^2/2$. However, if θ is continuously differentiable (e. g., θ is defined via (5); [14]), the nonsmooth Newton direction is a descent direction of θ and thus globalized methods can be designed. See [7] for a line search model and [13] for a trust region model.

The feature of *smoothing methods* is to construct a smoothing approximation function G: $\mathbf{R}^n \times \mathbf{R}_{++} \to \mathbf{R}^n$ of H such that for any $\varepsilon > 0$ and $x \in \mathbf{R}^n$, $G(\varepsilon, \cdot)$ is continuously differentiable on $\mathbf{R}^n$ and satisfies

$$\|H(x) - G(\varepsilon, x)\| \to 0 \quad \text{as } \varepsilon \downarrow 0 , \tag{9}$$

and then to find a solution of $H(x) = 0$ by (inexactly) solving the following problems for a given positive sequence $\{\varepsilon^k\}$ with $\varepsilon^k \to 0$ as $k \to \infty$,

$$G(\varepsilon^k, x) = 0 . \tag{10}$$

It was suggested in [21] to use the convolution to construct smooth approximations of the nonsmooth function H. A function Φ: $\mathbf{R}^n \to \mathbf{R}_+$ is called a *kernel function* if

$$\int_{\mathbf{R}^n} \Phi(x) \, \mathrm{d}x = 1 .$$

Suppose Φ is a smooth kernel function. Define Θ: $\mathbf{R}_{++} \times \mathbf{R}^n \to \mathbf{R}_+$ by

$$\Theta(\varepsilon, x) = \varepsilon^{-n}\Phi(\varepsilon^{-1}x)\,,$$

where $(\varepsilon, x) \in \mathbf{R}_{++} \times \mathbf{R}^n$. Then a smooth approximation of the projection operator Π_X can be described by

$$P(\varepsilon, x) = \int_{\mathbf{R}^n} \Pi_X(x - y)\Theta(\varepsilon, y)\,\mathrm{d}y\,, \tag{11}$$

where $(\varepsilon, x) \in \mathbf{R}_{++} \times \mathbf{R}^n$. Suppose that

$$\kappa := \int_{\mathbf{R}^n} \|y\|\,\Phi(y)\,\mathrm{d}y < +\infty\,.$$

Then for any $x \in \mathbf{R}^n$ and $\varepsilon > 0$ one has

$$\|P(\varepsilon, x) - \Pi_X(x)\| \le \kappa\varepsilon\,.$$

In general, $P(\varepsilon, x)$ is intractable because a multidimensional integration is involved. However, it can be written explicitly if X is of special structure (for example, X is a rectangular) and Φ is chosen particularly (see [3,12]). In fact, already in 1986 S. Smale [26] gave a smooth function $\frac{(w+\sqrt{w^2+\varepsilon^2})}{2}$ to approximate $\max(0, w)$, $w \in \mathbf{R}$, and used it to study linear complementarity problems. Also see [2,15]. The paper [1] stimulates much recent study about smoothing methods for solving NCP and VI. For the convenience of discussion, for any $\varepsilon < 0$, define $P(\varepsilon, x) = P(-\varepsilon, x)$ and $P(0, x) = \Pi_X(x)$, $x \in \mathbf{R}$. Then the smooth approximation G: $\mathbf{R}^{n+1} \to \mathbf{R}^n$ of H defined in (6) can be described by

$$G(\varepsilon, x) = x - P(\varepsilon, x - F(x))\,, \tag{12}$$

where $(\varepsilon, x) \in \mathbf{R} \times \mathbf{R}^n$. Thus, for any $(\varepsilon, x) \in \mathbf{R} \times \mathbf{R}^n$,

$$\|G(\varepsilon, x) - H(x)\| \le \kappa\varepsilon\,.$$

See [21] for a general case if H is not of the form defined in (3).

Note that, for variational inequalities, if the function F is only defined on X and not well defined outside X, then the function H defined in (3) is not well defined on $\mathbf{R}^n$. In this case, one can use the normal map introduced by S.M. Robinson [25] to overcome this difficulty. It is also noted that solving (1) is equivalent to finding a solution of the equation

$$H(x) := x - \Pi_X[x - F(\Pi_X(x))] = 0\,, \tag{13}$$

where $x \in \mathbf{R}^n$. The above-defined H only requires F being defined on X instead of on $\mathbf{R}^n$ as required by the function defined in (3). Unlike Robinson's normal map, the above map does not need to work on a transformed space, it works on the original space directly. By using the definition of P, the smoothing approximation G: $\mathbf{R}^{n+1} \to \mathbf{R}^n$ of H defined in (13) can be described by

$$G(\varepsilon, x) = x - P(\varepsilon, x - F(P(\varepsilon, x)))\,, \tag{14}$$

where $(\varepsilon, x) \in \mathbf{R} \times \mathbf{R}^n$.

The first globally and superlinearly (quadratically) convergent smoothing Newton method was proposed by X. Chen, Qi and D. Sun in [4], where the authors exploited a *Jacobian consistency property* and applied this property to an infinite sequence of smoothing approximation functions to get high-order convergent methods. The smoothing function defined by (12) satisfies the Jacobian consistence property while the one defined in (14) does not satisfy this property. The method in [4] was further studied by Chen and Y. Ye in [5].

Suppose that G: $\mathbf{R}^{n+1} \to \mathbf{R}^n$ is a smoothing approximation of H. Define E: $\mathbf{R}^n \to \mathbf{R}^n$ by

$$E(z) := \begin{pmatrix} \varepsilon \\ G(\varepsilon, x) \end{pmatrix}\,, \tag{15}$$

where $z := (\varepsilon, x) \in \mathbf{R} \times \mathbf{R}^n$. Then solving $H(x) = 0$ is equivalent to finding a solution of $E(\varepsilon, x) = 0$. Note that E is continuously differentiable at any $(\varepsilon, x) \in \mathbf{R} \times \mathbf{R}^n$ with $\varepsilon \ne 0$ and is possibly nonsmooth at $(0, x) \in \mathbf{R} \times \mathbf{R}^n$. We call E a *smoothing-nonsmooth reformulation* of H. Then classical Newton methods for solving smoothing equations can be used to solve $E(z) = 0$ with one additional requirement: ε must be positive during the process of iteration. The latter can be done by solving a slightly modified Newton equation:

$$E(z) + E'(z)\Delta z = \beta\overline{z}\,, \tag{16}$$

where $\beta \in (0, \infty)$ and $\overline{z} := (\overline{\varepsilon}, 0)$ with $\overline{\varepsilon} > 0$. It is obvious that one should control ε such that it neither converges too fast (no stronger global convergence results guaranteed) nor too slow (no fast local convergence results guaranteed). A special line search model involving β and $\overline{z}$ was designed in [23] to achieve this: Choose $\overline{\varepsilon} \in \mathbf{R}_{++}$ and $\gamma \in (0, 1)$ such that $\gamma\overline{\varepsilon} < 1$. Choose constants $\delta, \sigma \in (0, 1)$. Let $\varepsilon^0 := \overline{\varepsilon}, x^0 \in \mathbf{R}^n$ be an arbitrary point. For $k = 0, 1, \ldots$, find a solution Δz^k of (16) with

$z := z^k$ and $\beta := \gamma \min\{1, e(z)\}$, where for any $y \in \mathbf{R}^{n+1}$, $e(y) = \|E(y)\|^2$. Let l_k be the smallest nonnegative integer l satisfying

$$e(z^k + \delta^l \Delta z^k) \le [1 - 2\sigma(1 - \gamma\bar{\varepsilon})\delta^l]e(z^k) .$$

Define $z^k := z^k + \delta^{l_k} \Delta z^k$. It is often verified that G is also semismooth everywhere jointly with ε and x [23]. So the semismooth theory of nonsmooth Newton methods for solving nonsmooth equations can be used to obtain superlinear (quadratic) convergence of (ε, x) for the above smoothing Newton method while the global convergence is based on the particular designed line search. See [23] for details.

Conclusion

In this paper semismooth Newton methods and smoothing Newton methods for solving NCP and VI based on nonsmooth equations have been briefly reviewed. These topics are still undergoing a very fast development. See [22] for an up-to-date review. Another nonsmooth approach for solving NCP and VI is to reformulate these problems as unconstrained optimization problems whose objective functions are once but not twice differentiable, (see [11]).

See also

- ► Composite Nonsmooth Optimization
- ► Nonconvex-Nonsmooth Calculus of Variations
- ► Solving Hemivariational Inequalities by Nonsmooth Optimization Methods

References

1. Burke J, Xu S (1999) A polynomial time interior-point path-following algorithm for LCP based on Chen–Harker–Kanzow smoothing techniques. Math Program 86:91–103
2. Chen B, Harker PT (1993) A non-interior-point continuation method for linear complementarity problems. SIAM J Matrix Anal Appl 14:1168–1190
3. Chen C, Mangasarian OL (1996) A class of smoothing functions for nonlinear and mixed complementarity problems. Comput Optim Appl 5:97–138
4. Chen X, Qi L, Sun D (1998) Global and superlinear convergence of the smoothing Newton method and its application to general box constrained variational inequalities. Math Comput 67:519–540
5. Chen X, Ye Y (1999) On homotopy-smoothing methods for variational inequalities. SIAM J Control Optim 37:589–616
6. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
7. DeLuca T, Facchinei F, Kanzow C (1996) A semismooth equation approach to the solution of nonlinear complementarity problems. Math Program 75:407–439
8. Dennis JE, Schnabel RB (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs
9. Eaves BC (1971) On the basic theorem of complementarity. Math Program 1:68–75
10. Fischer A (1992) A special Newton-type optimization method. Optim 24:269–284
11. Fukushima M (1996) Merit functions for variational inequality and complementarity problems. In: Pillo GDi, Giannessi F (eds) Nonlinear Optimization and Applications. Plenum, New York, pp 155–170
12. Gabriel SA, Moré JJ (1997) Smoothing of mixed complementarity. In: Ferris MC, Pang JS (eds) Complementarity and Variational Problems: State of the Art. SIAM, Philadelphia, pp 105–116
13. Jiang H, Fukushima M, Qi L, Sun D (1998) A trust region method for solving generalized complementarity problems. SIAM J Optim 8:140–157
14. Kanzow C (1994) An unconstrained optimization technique for large-scale linearly constrained convex minimization. Computing 53:101–117
15. Kanzow C (1996) Some noninterior continuation methods for linear complementarity problems. SIAM J Matrix Anal Appl 17:851–868
16. Kummer B (1988) Newton's method for non-differentiable functions. In: Guddat J, Bank B, Hollatz H, Kall P, Klatte D, Kummer B, Lommatzsch K, Tammer L, Vlach M, Zimmerman K (eds) Adv. Mathematical Optimization. Akademie, Berlin, pp 114–125
17. Mifflin R (1977) Semismooth and semiconvex functions in constrained optimization. SIAM J Control Optim 15:957–972
18. Ortega JM, Rheinboldt WC (1970) Iterative solution of nonlinear equations in several variables. Acad. Press, New York
19. Pang J-S, Qi L (1993) Nonsmooth equations: Motivation and algorithms. SIAM J Optim 3:443–465
20. Qi L (1993) Convergence analysis of some algorithms for solving nonsmooth equations. Math Oper Res 18:227–244
21. Qi L, Chen X (1995) A globally convergent successive approximation method for severely nonsmooth equations. SIAM J Control Optim 33:402–418
22. Qi L, Sun D (1999) A survey of some nonsmooth equations and smoothing Newton methods. In: Hill R, Eberhard A, Glover B, Ralph D (eds) Progress in Optimization. Kluwer, Dordrecht, pp 121–146
23. Qi L, Sun D, Zhou G (2000) A new look at smoothing Newton methods for nonlinear complementarity problems and box constrained variational inequalities. Math Program 87:1–35

24. Qi L, Sun J (1993) A nonsmooth version of Newton's method. Math Program 58:353–367
25. Robinson SM (1992) Normal maps induced by linear transformation. Math Oper Res 17:691–714
26. Smale S (1986) Algorithms for solving equations. In: Proc. Internat. Congress Math, pp 172–195

NP-complete Problems and Proof Methodology

SANATAN RAI, GEORGE VAIRAKTARAKIS
Department OR and Operations Management, Case Western Reserve University, Cleveland, USA

MSC2000: 90C60, 68Q25

Article Outline

Keywords
Some Known *NP*-Complete Problems
Methodology for *NP*-Completeness Proofs
Example Proofs
Conclusion
See also
References

Keywords

Computational complexity; Reducibility; Polynomial time reduction; NP-hard problem; NP-complete problem; strong NP-completeness; ordinary NP-completeness

Given a combinatorial problem, one tries to exploit its structure so as to develop a solution algorithm that guarantees identifying an optimal solution for every instance of the problem. For some problems, the inherent structure is such that one can develop an algorithm that progressively builds an optimal solution, or selects among a small number of candidate solutions. Such algorithms are quite desirable as their computer time requirements are small, and often a bounded polynomial function of the number n of parameters needed to specify the problem (e. g. $\mathcal{O}(n)$ or $\mathcal{O}(n^2)$; see [9] for details on algorithmic *complexity*). Thus, such algorithms are called *polynomial algorithms*. Problems solvable by a polynomial algorithm may be solved quickly on a computer.

Unfortunately, not all combinatorial problems possess enough structure to allow for a polynomial algorithm. Hence, when we encounter a new problem for which we cannot identify enough structure, we would like to know whether this lack of structure is due to the problem itself, or to incomplete analysis. To address this issue, one idea is to compare the structure of the problem at hand with the structure of other well known and notoriously hard problems; such problems are known in literature as *NP-complete problems*. Specifically, if we can show that our problem is 'equivalent' to an NP-complete problem, then any algorithm that solves our problem can be used to solve the hard one and vice versa. Then, we can justifiably suggest that our problem is very difficult. With this information, we can either continue focusing our efforts in finding a *polynomial time optimal algorithm* (admitting that our chances for success are low) or consider heuristics or enumeration techniques.

If we are ever able to find a polynomial time algorithm for a *NP*-complete problem, we will make one of the most important discoveries in human knowledge. This will mean that we are able to solve all hard combinatorial problems very quickly (for details about the relationship between the class of polynomially solvable problems and the class of *NP*-complete problems see [4]). If we believe that this is unlikely, then we focus our analyses on enumeration techniques. Hence, in the latter case, the equivalence between the problem at hand and the *NP*-complete problem have dictated our approach towards the problem at hand. Since 1971 when the foundations of complexity theory were developed by S.A. Cook in [2], all the papers that have appeared in the literature have taken the latter route – namely, they focus on heuristics and/or enumeration techniques. In this sense, complexity theory is a very useful tool for determining our approach towards solving difficult combinatorial problems. In this article, we present some of the fundamental techniques that have been used in the literature to prove equivalence among problems. We start in the next section by presenting a list of combinatorial optimization problems. Then, we describe some basic methodology for theorem proving in complexity theory, and conclude with a few illustrative example proofs.

Some Known *NP*-Complete Problems

In what follows we present some problems commonly used in the literature to prove equivalence between problems. All of these problems are notoriously hard, they belong to the class of *NP*-complete problems, and hence, no polynomial algorithm is known for them. In the rest of this article we present methods for proving the equivalence between selected pairs among of these problems. Our selected problems span a sample of areas in combinatorial optimization including set partition, logic, graph theory, network theory, and scheduling theory.

The following two problems are representative of problems in set partition. The problems are stated in their decision form, i. e., we only require a 'yes/no' answer to resolve them. Each *instance* is described by the input data required to define the problem, and each *question* requires a 'yes/no' answer. This presentation of combinatorial problems follows the presentation form adopted in [4] which was the first text devoted to a systematic compilation of hard combinatorial problems.

Definition 1 (*Partition*) INSTANCE: Set $A = \{a_1, \ldots, a_n\}$ of elements, and a set function $s: A \to Z^+$.

QUESTION: Does there exist a set $A' \subset A$ such that

$$\sum_{a \in A'} s(a) = \frac{1}{2} \sum_{a \in A} s(a) \,?$$

Definition 2 (3-*Partition*) INSTANCE: Set $A = \{a_1, \ldots, a_{3n}\}$ of elements, set function $s: A \to Z^+$, and threshold value B.

QUESTION: Are there subsets $A_k \subset A$, $k = 1, \ldots, n$, such that

$$\sum_{a \in A_i} s(a) = B \text{ and } |A_k| = 3 \text{ for } 1 \le i \le n \,?$$

These problems are among the most popular problems found in complexity theory. Note that in 'Partition', A is partitioned in two sets with no restriction on the number of elements per set. For this reason, 'Partition' is often referred to in the literature as 2-*partition*. In contrast, '3-partition' involves the partition of A in n sets each consisting of precisely three elements.

Definition 3 (3-*Satisfiability*) INSTANCE: A Boolean expression B in literals x_i, $i = 1, \ldots, q$,

$$B = (p_{11} \vee p_{12} \vee p_{13}) \wedge \cdots \wedge (p_{n1} \vee p_{n2} \vee p_{n3}) = \wedge_{i=1}^{n} \vee_{j=1}^{3} p_{ij}$$

where each p_{ij} is either x_k or its negation $\overline{x}_k$ for some $1 \le k \le q$.

QUESTION: Is there an assignment for the literals x_k such that B is true?

This problem is also referred to in the literature as 3-*Sat*. The related problem where every clause has an arbitrary number of literals (rather than precisely three) is known in the literature as the *satisfiability* problem, or *Sat*, and has the distinction of being the first *NP*-complete problem (see [2]). Note that, if we were able to solve 'Sat' in polynomial time, then we would be able to determine the truth value of all possible statements in propositional calculus. Effectively, we could cast every imaginable theorem in propositional form, and let a computer answer it. This would be equivalent to theorem proving using computers.

Definition 4 (*Maximum clique*) INSTANCE: Graph $G = (V, E)$ and positive integer k.

QUESTION: Does there exist a complete subgraph of G on k vertices?

The 'maximum clique' is a very important problem in graph theory with applications in diverse fields. The following problem is encountered when one wants to identify a path (with certain properties) in a given network.

Definition 5 (*Impossible pairs constrained path problem* (IPP)) INSTANCE: Directed graph $G = (V, A)$ with source s and sink t, and pairs of nodes (a_i, b_i) for $i = 1, \ldots, n$.

QUESTION: Does there exist a directed $s - t$ path containing at most one node from each pair (a_i, b_i) for $i = 1, \ldots, n$?

The following problems are found in scheduling theory where a set of jobs is to be processed in a production system so as to optimize a given objective. We use the standard 3-field notation $\alpha/\beta/\gamma$ (see [6]) where α denotes the number and type of processors, β describes the job characteristics, and γ the objective function. For example, $\alpha = 1$ indicates a single processor, and α

= Pm denotes m identical processors operated in parallel. Job characteristics include completion deadlines, start times, precedence constraints among jobs, or processing characteristics. A popular scheduling objective is the minimization of the *makespan*, $C_{\max}$ – the maximum completion time, where the maximum is taken over all jobs. Evidently, $C_{\max}$ may be the preferred objective when a manager wants to maximize the utilization of processors.

Definition 6 $\mathbf{1/r_i, d_i/C_i \le d_i}$ INSTANCE: Set $J = \{J_1, \ldots, J_n\}$ of jobs, each with a processing time p_i, a duedate d_i, and a release time r_i, $1 \le i \le n$.

QUESTION: Is there a schedule of the jobs in J such that each job starts after time r_i and completes at time $C_i \le d_i$?

In the above problem one wants a single processor schedule where every job J_i starts no earlier than time r_i and finishes no later than time d_i. Such schedule would allow on time delivery of jobs to the customers.

Definition 7 $\mathbf{Pm//C_{max}}$ INSTANCE: Set $M = \{M_1, \ldots, M_m\}$ of parallel identical processors, set $J = \{J_1, \ldots, J_n\}$ of jobs each with a processing time p_i, and threshold value B.

QUESTION: Is there a nonpreemptive assignment of the n jobs to the m processors so that at any time every machine processes at most one job, and the completion time of J_i is $C_i \le B$ for every $1 \le i \le n$?

Here we seek a schedule of the n jobs on the m processors so as to minimize the completion time of the last job. Every processor can process at most one job at a time, and every job must be processed in its entirety without being interrupted by any of the processors.

Definition 8 $\mathbf{P//prec, p_i = 1/C_{max}}$ INSTANCE: Set $J = \{J_1, \ldots, J_n\}$ of jobs with processing time $p_i = 1$ for $1 \le i \le n$, and set A of precedence constraints between jobs in J. Also, set P of parallel identical processors, and a threshold value B.

QUESTION: Is there a nonpreemptive assignment of jobs to processors so that at any time every machine processes at most one job, the job precedence constraints are satisfied, and the completion time of J_i is $C_i \le B$ for every $1 \le i \le n$?

Unlike the previous problem, the number of parallel identical processors is not specified in this problem, i. e., $\alpha = P$. Every job has unit processing time. These unit jobs must satisfy a set of precedence constraints. Among all nonpreemptive schedules that satisfy these constraints, we seek one that minimizes the completion time of the last job.

The following section presents a general methodology for *NP*-completeness proofs with the problems described above as examples.

Methodology for *NP*-Completeness Proofs

We start by presenting the four basic steps of a complexity proof. Such proofs demonstrate that a new problem Π can be transformed to a known *NP*-complete problem $P \in \mathcal{NPC}$. To indicate this reduction we use the notation $P \propto \Pi$.

1) Show that $\Pi \in \mathcal{NP}$.
2) Construct a transformation from P to Π.
3) Show that the transformation in step 2 can be effected by a polynomial time algorithm.
4) Show that there exists a solution S_P for P, if and only if there exists a solution S_Π for Π, and that the transformations S_P to S_Π and vice versa is done by a pseudopolynomial algorithm for *strong NP-complete reductions*, and by a polynomial algorithm for *ordinary NP-complete reductions*.

Step 1 requires that, given a solution S_Π of Π we can check whether S_Π provides a 'yes' or 'no' answer for Π, using a polynomial algorithm. Given an arbitrary instance I of P, step 2 requires constructing an instance I' of Π. Step 3 requires that the construction of I' from I is polynomial on the number of input data required to specify I. Finally, Step 4 requires proving that, given a solution S_P for the instance I, we can construct a solution S_Π for I' and vice versa. In almost all reductions that have appeared in literature, steps 1–3 are quite simple and usually straightforward, while step 4 often requires considerable creativity.

Step 4 refers to strong and ordinary *NP*-complete problems. In a nutshell, this is one of many classifications of *NP*-complete problems into smaller subclasses. For a detailed description of these classes see [4]. In practical terms, an ordinary *NP*-complete problem can be solved using implicit enumeration algorithms like dynamic programming. In this case, the complexity of

the algorithm is not polynomial on the *length of input data*, but it is polynomial on the *size* of these data. For instance, Partition is a *NP*-complete problem solvable by dynamic programming in $\mathcal{O}(n \sum_i s(a_i))$ time (see [8]). Evidently, this complexity is polynomial on the size $\sum_i s(a_i)$ of the data. To see that this complexity bound is not polynomial on the length of the data, consider the binary encoding scheme. In this scheme each $s(a_i)$ can be represented by a string of length $\mathcal{O}(\log s(a_i))$, and hence $s(a_1), \ldots, s(a_n)$ can be described by a string of length $\mathcal{O}(\sum_i \log s(a_i))$ which is no greater than $\mathcal{O}(n \log B)$ where $B = \sum_i s(a_i)$. We see that the time complexity $\mathcal{O}(nB)$ of the dynamic program (DP)

- is polynomial on the size B of the data.
- but not polynomial on the length of the input data for the instance I of 'partition', where length(I) = $\mathcal{O}(n \log B)$.

Notice that the complexity of this DP, $\mathcal{O}(nB)$, is not bounded by any polynomial function of $n \log B$. When the complexity of an algorithm is polynomial on the size of the data, but not the length of the input, we refer to the algorithm as a *pseudopolynomial algorithm*. A *NP*-complete problem solvable by a pseudopolynomial algorithm is called *ordinary NP-complete*. Else, the problem is *strongly NP-complete*.

As indicated by its complexity, solving 'partition' is easier than solving any problem not solvable by implicit enumeration. Such problems require explicit enumeration algorithms like branch and bound. In the list of problems given earlier, 'partition' is the only problem solvable by dynamic programming.

Given the complexity status of the known *NP*-complete problem P, we can determine whether a new problem Π is strongly or ordinary *NP*-complete, if one of the following happens:

- P is strongly *NP*-complete, and $P \propto \Pi$. Then, Π is strongly *NP*-complete.
- P is ordinary *NP*-complete, $P \propto \Pi$, and a pseudopolynomial algorithm exists for Π. Then, Π is ordinary *NP*-complete.

As indicated in the following tree, all other outcomes result to incomplete determination of the exact complexity status of problem Π.

Example Proofs

In this section we present some simple applications of the four step reduction process outlined previously.

Example 9 Partition $\propto P2//C_{\max}$. Step 1 requires us to show that $P2//C_{\max} \in \mathcal{NP}$, i. e., given a schedule S of the n jobs, we can check whether the associated makespan $C_{\max}(S) \leq B$ in polynomial time. To perform the check, we need to find the completion time of the last job processed by each of the processors. This requires no more than n additions involving the processing times of the jobs in J. Hence, $C_{\max}(S)$ can be computed in $\mathcal{O}(n)$ time, and subsequently, whether $C_{\max}(S) \leq B$ or not can be established in $\mathcal{O}(1)$ time. Hence, $P2//C_{\max} \in \mathcal{NP}$. This completes step 1.

For step 2 we must construct an instance of $P2//C_{\max}$, given an instance of 'partition'. Let I be an instance of 'partition', and $s(a_1), \ldots, s(a_n)$ be the values of the n elements $a_1, \ldots, a_n$ in I. We construct an instance I' of $P2//C_{\max}$ as follows. Let $p_i = s(a_i)$, $i \in \{1, \ldots, n\}$. The number of processors is $m = 2$ and the number of jobs is n. The threshold value B is set to $B = (1/2) \sum_i p_i$. This completes Step 2.

This construction of I' required $n + 2$ assignments, and $n + 1$ basic operations to compute B. Evidently, the total amount of effort required to construct I' is $\mathcal{O}(n)$. This concludes step 3.

In step 4, we need to show that, there exists a solution for the instance I of 'partition' if and only if there exists a solution for the instance I' of $P2//C_{\max}$. Indeed, let A_1, A_2 be a partition of A such that

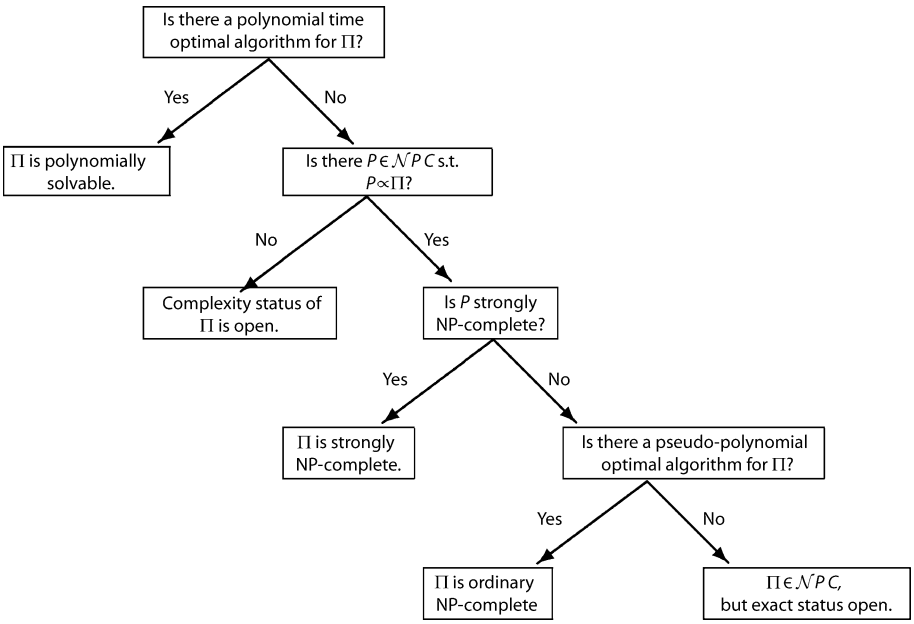
$$\sum_{a_i \in A_1} s(a_i) = \sum_{a_i \in A_2} s(a_i) = \frac{1}{2} \sum_{a_i \in A} s(a_i) .$$

From this, we can construct a solution for I' by assigning all jobs in $J^1 = \{J_i : a_i \in A_1\}$ to be processed (in any order) by processor M_1, and all jobs in $J^2 = \{J_i : a_i \in A_2\}$ to be processed (in any order) by processor M_2. Let S be the resulting schedule for $P2//C_{\max}$. By definition of A_1 and A_2,

$$\sum_{J_i \in J^1} p_i = \sum_{J_i \in J^2} p_i = \frac{1}{2} \sum_i p_i = B .$$

Clearly, the schedule S is constructed from A_1, A_2 in $\mathcal{O}(n)$ time. Similarly, given a schedule S that solves $P2//C_{\max}$ one can construct the partition A_1, A_2 in $\mathcal{O}(n)$ time as well.

Since 'partition' is an ordinary *NP*-complete problem, to completely determine the status of $P2//C_{\max}$, we will have to develop a pseudopolynomial algorithm for



NP-complete Problems and Proof Methodology, Figure 1
Establishing the complexity status of a problem Π

it. Such algorithm can in fact be developed (see [1]) which means that $P2//C_{max}$ belongs to the class of ordinary *NP*-complete problems.

Example 10 3-partition $\propto 1/r_i, d_i/C_i \leq d_i$ For Step 1, consider a given schedule S for $1/r_i, d_i/C_i \leq d_i$. By scanning the n jobs of S in sequence, we can obtain the start and completion times of the n jobs in $\mathcal{O}(n)$ time. Then, we can perform the n comparisons $r_i \leq C_i \leq d_i$ in $\mathcal{O}(n)$ time. Hence, $1/r_i, d_i/C_i \leq d_i \in \mathcal{NP}$. This completes step 1.

For Step 2, suppose we are given an instance I of 3-partition, that is, a set $A = \{a_1, \ldots, a_{3n}\}$ with associated values $s(a_1), \ldots, s(a_{3n})$, and threshold value B. For reasons that will become clear later, we make the following assumptions. Firstly, $s(a_i) < B$ for every $1 \leq i \leq 3n$, otherwise we can immediately conclude that there is no solution for the instance I of 3-partition. Secondly, we assume that the elements $a_1, \ldots, a_{3n}$ possess the property

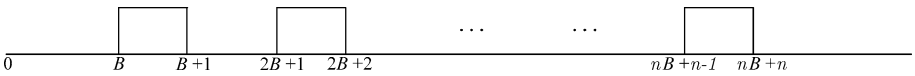
$*)$ $B/4 < s(a_i) < B/2$ for $i = 1, \ldots, 3n$.

Indeed, if this condition does not hold, we can replace the value of each element a_i by $s'(a_i) = s(a_i) + B$, and the threshold value B by $B' = 4B$. Then, it is easy to see that, $B < s'(a_i) < 2B$ (because $s(a_i) < B$), and hence $B'/4 < s'(a_i) < B'/2$ for $i = 1, \ldots, 3n$. In this case, the elements $a_1, \ldots, a_{3n}$ possess the required property with respect to the values $s'(a_i)$ and the threshold B'. Hence, without loss of generality, we assume that the elements of the instance I of 3-partition satisfy property $*$). This ensures that, if $\sum_{a_i \in A_k \subset A} s(a_i) = B$, then $|A_k| = 3$.

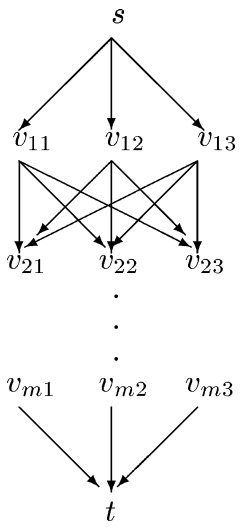
From I, we construct the instance I' of $1/r_i, d_i/C_i \leq d_i \in \mathcal{NP}$ with $4n$ jobs as follows. Set, $p_i = s(a_i)$, $r_i = 0$ and $d_i = nB + n - 1$ for $i = 1, \ldots, 3n$. Also include in I' n 'filler' jobs, $F_1, \ldots, F_n$ with processing times $p_{F_i} = 1$, start times $r_{F_i} = iB + i - 1$ and due-dates $d_{F_i} = iB + i$ for $i = 1, \ldots, n$. Note that $p_{F_i} = d_{F_i} - r_{F_i} = 1$ so the filler jobs have no slack. This completes step 2.

Clearly, The construction of I' is done in $\mathcal{O}(n)$ time. To prove step 4, let A_k, $k = 1, \ldots, n$ be a solution of I. By definition of the 3-partition problem, each set A_k is such that $|A_k| = 3$, and $\sum_{a \in A_k} s(a) = B$, $k = 1, \ldots, n$. From this, construct a schedule S for I' as follows. Let $J^k = \{J_i: a_i \in A_k\}$ for $k = 1, \ldots, n$, and schedule the jobs in J^k to be processed (in any order) starting at time $(k-1)B + k - 1$ and finishing at time $kB + k - 1$ as in Fig. 2.

Evidently, the schedule S is constructed in $\mathcal{O}(n)$ time. Alternatively, if S is a schedule that solves I', then the filler jobs should partition the $3n$ regular jobs in n distinct sets, say $J^k = \{J_i : J_i$ starts after F_{k-1} and is com-



NP-complete Problems and Proof Methodology, Figure 2
Instance of 1/r_i/C_i



NP-complete Problems and Proof Methodology, Figure 3
Instance of IPP

pleted before F_k} for $k = 1, \ldots, n$. Note that for $k = 1$, J^1 contains jobs that start after time $t = 0$ (F_0 is not defined) and complete prior to F_1. As the filler jobs have no slack, in each J_k the sum of the processing times must be precisely B units. By construction of the instance I, each J^k must contain precisely 3 jobs. Therefore, the sets $A_k = \{a_i \in A: J_i \in J^k\}$ for $k = 1, \ldots, n$ partition A into n triplets such that $\sum_{a_i \in A_k} s(a_i) = \sum_{J_i \in J^k} p_i = B$ for $k = 1, \ldots, n$, i. e., the collection $\{A_k\}_{k=1}^n$ solves I. Hence, given a schedule S that solves I', we can construct a partition $\{A_k\}_{k=1}^n$ that solves I, in $\mathcal{O}(n)$ time. This completes step 4.

This example demonstrates that we can often construct the instance I so that it satisfies certain properties that may become handy in proving step 4. As seen so far, quite often steps 1 and 3, as well as parts of step 4 are trivial, in such cases the details are often omitted. A typical example is provided next.

Example 11 3-Sat $\propto$ *IPP* Given a Boolean expression B, construct a digraph G as indicated in Fig. 3.

Specifically, the arc set of G is

$$E = \{(s, v_{1j}): \ j = 1, 2, 3\} \cup \{(v_{ij}, v_{i+1,k}): \ 1 \le i \le m-1; \ 1 \le j, k \le 3\} \cup \{(v_{mj}, t): \ j = 1, 2, 3\}.$$

Also, let the set of restricted pairs be $M = \{(v_{ij}, v_{kl}): \ p_{ij} = \overline{p}_{kl}\}$.

Let $P = sv_{1l_1} \ldots v_{ml_m}t$ be a constrained path. By making p_{il_i} true for $i = 1, \ldots, m$, we force B to be true. Since P satisfies the constraints in M, these assignments do not conflict. Therefore the existence of a path P implies the satisfiability of B and vice versa. This construction took only $\mathcal{O}(m)$ time, and the reduction is complete.

Our last example demonstrates how complexity theory can be used to derive approximation results.

Example 12 Max Clique $\propto$ *P/prec,* $p_j = 1/C_{max}$ Given $G = (V, E)$ construct a digraph D as follows. Introduce a job J_v for every $v \in V$, a job J_e for every $e \in E$, and an arc $J_v \to J_e$ whenever v is an endpoint of e. Let $l = \binom{k}{2}$ be the number of edges in a k-clique, $k' = |V| - k$ the number of nodes not in the clique and $l' = |E| - l$, the number of edges not in the clique.

Let the number of processors be $m = \max(k, l+k', l') + 1$. Introduce three sets of dummy nodes

$$X_x, \quad x = 1, \ldots, m-k,$$
$$Y_y, \quad y = 1, \ldots, m-l-k',$$
$$Z_z, \quad z = 1, \ldots, m-l'.$$

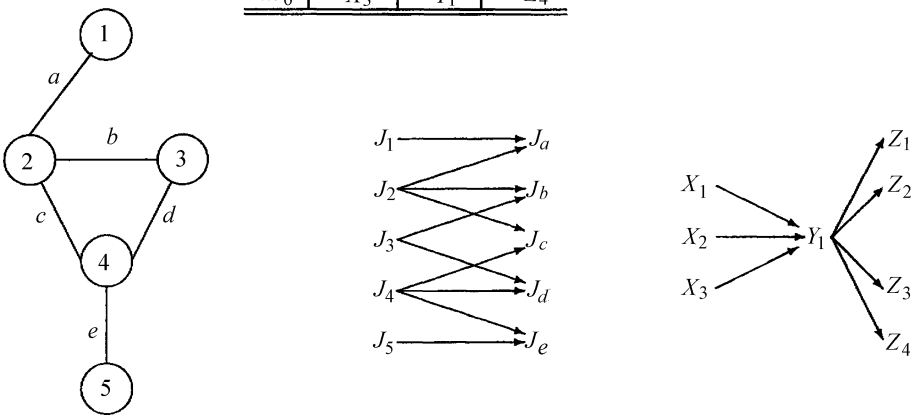
and the arcs $X_x \to Y_y \to Z_z$. Note that the total number of jobs is $3m$.

We will show that there exists a feasible schedule for the instance I of P/prec, $p_j = 1/C_{\max} = 3$ if and only if there exists a clique of size k. Or, in other words, there is a clique of size k if and only if we can complete the jobs in 3 periods.

($\Leftarrow$). Suppose a k-clique exists. Then,

- Schedule the jobs corresponding to the k clique vertices in period 1 (see Fig. 4).

	1	2	3
m_1	J_2	J_b	J_a
m_2	J_3	J_c	J_e
m_3	J_4	J_d	Z_1
m_4	X_1	J_1	Z_2
m_5	X_2	J_5	Z_3
m_6	X_3	Y_1	Z_4



NP-complete Problems and Proof Methodology, Figure 4
Instance of *P*/prec, $p = 1/C_{max}$

- Schedule the $m-k$ jobs corresponding to each X_x in the remaining processors in the first period.
- Schedule the l jobs corresponding to the clique edges in period 2, schedule the $k' = m - k$ remaining vertex jobs in period 2, and the $m - l - k'$ jobs from Yy in period 2.
- Finally, schedule the $l' = |\mathrm{E}| - l$ remaining edge jobs in period 3 and the $m - l'$ jobs from Z_z in period 3 (see Fig. 4).

This is a feasible schedule on three periods.

($\Rightarrow$). We prove this by contradiction. Suppose no k-clique exists. We will demonstrate that no 3-period schedule can exist either. Since there are m machines and $3m$ jobs, in a three period schedule all machines must be busy for all the three periods. Certainly, due to the precedence constraints $X_x \rightarrow Y_y \rightarrow Z_z$, all $m - k$ jobs must be done in period 1, all Y_y in period 2 and all the Z_z in period 3. Then, the remaining k slots in period 1 must be taken by vertex jobs. Since no k-clique exists, the edge jobs that can be scheduled in period 2 is no greater than $l - 1$. Then the jobs scheduled in period 2 are the $l - 1$ edge jobs, all the Y_y jobs and the $m - k$ remaining vertex jobs. The total number of these jobs does not exceed $m - 1$. Hence, there must be at least one machine that stays idle for a period. Hence no schedule on three periods can exist, which is a contradiction.

This completes the reduction, and since 'Max Clique' is strongly *NP*-complete, so is *P*/prec, $p_j = 1/C_{max}$.

Observe that, in Example 12, the number m of processors is not fixed. Therefore, Example 12 does not settle the complexity status for the problem in which m is specified beforehand. For example, it does not resolve the complexity status of *P*3/prec, $p_j = 1/C_{max}$. A careful examination of Example 12 can enable us to construct error bounds. Suppose we develop a heuristic H to solve *P*/prec, $p_j = 1/C_{max}$. Then, given an instance of Max Clique, we generate the associated instance of *P*/prec, $p_j = 1/C_{max}$ described in Example 12, and apply H on it. Since H cannot always solve *P*/prec, $p_j = 1/C_{max}$ optimally (unless $\mathcal{P} = \mathcal{NP}$), there will be instances where an optimal schedule on three periods exists, but our heuristic H fails to find it. Since the instance of *P*/prec, $p_j = 1/C_{max}$ produced in Example 12 uses integer processing times, whenever H fails to produce an optimal solution, its makespan C_H will be $C_H \geq 4$ rather than 3. Equivalently, the worst-case error bound for H must be $p = C_H/C^* \geq 4/3$. This observation is true for any pos-

sible heuristic for $P/\text{prec}, p_j = 1/C_{\max}$. Therefore, unless $\mathcal{P} = \mathcal{NP}$, no heuristic algorithm can exist with worst-case error bound $p < 4/3$. Therefore, research efforts for worst-case analyses for $P/\text{prec}, p_j = 1/C_{\max}$ must be focused on values of $p = 4/3$ or larger.

Conclusion

In this article we presented some basic techniques used in proving *NP*-completeness results. Following Cook's seminal paper [2], the first list of reductions for combinatorial problems was compiled in [5]. The four examples described in this article are illustrative of diverse optimization areas. They can be found in [3,4,7], and [6], respectively.

See also

- ▶ Complexity Classes in Optimization
- ▶ Complexity of Degeneracy
- ▶ Complexity of Gradients, Jacobians, and Hessians
- ▶ Complexity Theory
- ▶ Complexity Theory: Quadratic Programming
- ▶ Computational Complexity Theory
- ▶ Fractional Combinatorial Optimization
- ▶ Information-Based Complexity and Information-Based Optimization
- ▶ Kolmogorov Complexity
- ▶ Mixed Integer Nonlinear Programming
- ▶ Parallel Computing: Complexity Classes

References

1. Cheng TCE, Sin CCS (1990) A state-of-the-art review of parallel-machine scheduling research. Europ J Oper Res 47:271–292
2. Cook SA (1971) The complexity of theorem proving procedures. In: Proc. 3rd Annual ACM Symposium on Theory of Computing. ACM, New York, pp 151–158
3. Gabow HN, Maheshwari SN, Osterweil L (1976) On two problems in the generation of program test paths. IEEE Trans. Software Engin., London, pp 227–231
4. Garey MR, Johnson DS (1979) Computers and intractability. Freeman, New York
5. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations. Plenum, New York, pp 85–103
6. Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (1993) Sequencing and scheduling: Algorithms and complexity. In: Graves SC, Rinnooy Kan AHG, Zipkin P (eds) Handbook Oper. Res. and Management Sci.: Logistics of Production and Inventory. North-Holland, Amsterdam, pp 445–522
7. Lenstra JK, Rinnooy Kan AHG, Brucker P (1977) Complexity of machine scheduling problems. Ann Discret Math 1:343–362
8. Martello S, Toth P (1990) Knapsack problems: Algorithms and computer implementations. Wiley, New York
9. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: Algorithms and complexity. Prentice-Hall, Englewood Cliffs

Numerical Methods for Unary Optimization

EMILIO SPEDICATO[1], NAIYANG DENG[2], JIANZHONG ZHANG[3], SHAOLIN XI[4]

[1] Department Math., University Bergamo, Bergamo, Italy

[2] Department Basic Sci., China Agricultural University, Beijing, China

[3] Department of Mathematics, City University of Hong Kong, Kowloon Tong, Hong Kong

[4] Computer Institute, Beijing Polytechnic University, Beijing, China

MSC2000: 90C30

Article Outline

Keywords
Introduction
Goldfarb–Wang Algorithm GW
Analysis of the Replacement Criterion
Using the Trust Region Approach
An Inexact Newton Method Using Preconditioned Conjugate Gradient Techniques
See also
References

Keywords

Unary optimization; Partial-update Newton method; Goldfarb–Wang algorithm; Inexact Newton method

We give a survey of numerical methods for the *unary optimization* problem: $\min f(x) = \sum_{i=1}^{m} U_i(\alpha_i(x))$, $x \in \mathbf{R}^n$, where $U(\cdot)$ is a function of a single argument and $\alpha_i(x) = a_i^\top x$, $a_i \in \mathbf{R}^n$, $i = 1, \ldots, m$.

Introduction

Consider the unconstrained optimization problem:

$$\min f(x), \quad x \in \mathbf{R}^n . \tag{1}$$

If $f(x)$ takes the form

$$f(x) = \sum_{i=1}^{m} U_i(\alpha_i(x)), \quad x \in \mathbf{R}^n , \tag{2}$$

where $U_i(\cdot)$ is a smooth function of a single argument and $\alpha_i(x) = a_i^\top x$, $a_i \in \mathbf{R}^n$, $i = 1, \ldots, m$, then we call problem (1) a *unary optimization problem*. Usually $m \geq n$. Unary optimization problems appear in many practical fields such as linear robust regression, portfolio selection and chemical equilibrium, see, e. g. [1,2,9].

The unary optimization problem was first proposed by G.P. McCormick and A. Sofer [9]. They exploited the special structure of derivatives of the unary function and applied the resulting algorithm to the solution of the classical chemical equilibrium problem. D. Goldfarb and S. Wang [7] first proposed an implementable algorithm specially for the unary optimization problem. Using the special structure of unary functions, in order to save computation cost, they modified the regular Newton method to develop a *partial-update Newton method*. In recent years some Chinese scholars worked on this subject and published several papers. We will review the state of art in this new interesting field.

Goldfarb–Wang Algorithm GW

If the unary functions $U_i(\cdot)$, $i = 1, \ldots, m$, in (2) are all differentiable and the Hessian matrix $\nabla^2 f(x)$ is nonsingular for all $x \in \mathbf{R}^n$, then we can use Newton method to solve the problem (1), i. e., starting from the current point x^k, we compute the new iterate by

$$x^{k+1} = x^k + s^k ,$$

where the increment s^k is obtained by solving the Newton equation

$$\nabla^2 f(x^k)s = -\nabla f(x^k) . \tag{3}$$

Goldfarb and Wang noticed that the Hessian $\nabla^2 f(x)$ of the unary function has the following form:

$$\nabla^2 f(x) = \sum_{i=1}^{m} \phi_i(x) a_i a_i^\top , \tag{4}$$

where

$$\phi_i(x) = d^2 U_i \frac{\alpha_i(x)}{d\alpha_i^2} . \tag{5}$$

Let $\Phi(x) = \mathrm{diag}(\phi_1(x), \ldots, \phi_m(x))$, and $A^\top = (a_1, \ldots, a_m)$. Then we can write

$$\nabla^2 f(x) = A^\top \Phi(x) A . \tag{6}$$

Notice that only $\phi_i(x)$ on the right-hand side of (4) or (6) may vary when x changes and the rank of the matrix $\phi_i(x)\, a_i a_i^\top$ is one for any nonzero value ϕ_i. Suppose that only the jth diagonal element of $\Phi(x^k)$ and $\Phi(x^{k-1})$ differs at iteration k, i. e.,

$$A^\top \Phi(x^k) A = A^\top \Phi(x^{k-1}) A + (\phi_j(x^k) - \phi_j(x^{k-1})) a_j a_j^\top , \tag{7}$$

or equivalently,

$$\nabla^2 f(x^k) = \nabla^2 f(x^{k-1}) + (\phi_j(x^k) - \phi_{j-1}(x^k)) a_j a_j^\top . \tag{8}$$

Therefore, $\nabla^2 f(x^k)^{-1}$ can be obtained from $\nabla^2 f(x^{k-1})^{-1}$ by the well-known *Sherman–Morrison rank-one update formula*, and the next iteration can be obtained in only $O(n^2)$ arithmetic operations after evaluating ∇f and ϕ_i, $i = 1, \ldots, m$.

The essential point of the GW algorithm is, in the kth iteration, to construct an approximation B^k to the Hessian $\nabla^2 f(x^k)$. Goldfarb and Wang compute the increment s^k by solving the approximate Newton equation

$$B^k s = -\nabla f(x^k) , \tag{9}$$

where $B^k = A^\top \Phi^k A$, and $\Phi^k = \mathrm{diag}(\phi_1^k, \ldots, \phi_m^k)$. So their method can be viewed as a special type of *inexact Newton method*. The diagonal matrix Φ^k is introduced to approximate $\Phi(x^k)$, and is defined by setting, for $i = 1, \ldots, m$, $\phi_i^0 = \phi_i(x^0)$, and

$$\phi_i^{k+1} = \phi_i^k , \tag{10}$$

if $\phi_i(x^{k+1})$ is 'replaceable' by ϕ_i^k; and

$$\phi_i^{k+1} = \phi_i(x^{k+1}) \quad \text{otherwise}, \tag{11}$$

for $k \geq 0$. Choleski factorization is then used to solve (9). Notice that

$$B^{k+1} = B^k + \sum_{i \in S^k} (\phi_i^{k+1} - \phi_i^k) a_i a_i^\top . \tag{12}$$

where S^k is the set of indices i such that $\phi_i(x^{k+1})$ is not 'replaceable' by ϕ_i^k, $i = 1, \ldots, m$. We can exploit rank-one updates starting from the Choleski factorization of B^k, which needs $O(|S^k|n^2)$ operations, where $|S^k|$ is the number of elements in S^k. So it is possible to save computation cost if $|S^k|$ is not very large.

Consider the assumptions:

A1) there exists a point $x^* \in \mathbf{R}^n$ with $\nabla f(x^*) = 0$;

A2) $\nabla^2 f(x^*)$ is nonsingular;

A3) $\Phi(x)$ is continuous in a neighborhood of x^*.

Goldfarb and Wang proved the following

Theorem 1 *Let assumptions A1)–A3) hold. Then there exists $\delta > 0$ such that if $\|x^0 - x^*\| \le \delta$, the sequence x^k generated by the partial-update Newton method converges to $x*$. Moreover, the convergence is linear, i. e.,*

$$\left\|x^{k+1} - x^k\right\| \le t \left\|x^k - x^*\right\| , \quad \forall k \tag{13}$$

where $0 < t < 1$.

In order to get a higher rate of convergence, they proposed two 'replacement' criteria for the method:

- Criterion 1: For $i = 1, \ldots, m$, ϕ_i^k is replaceable by ϕ_i^{k-1} if

$$\frac{\left|\phi_i(x^k) - \phi_i^{k-1}\right|}{\left|\phi_i(x^k) - \phi_i^{k-1}\right| + \left\|\nabla f(x^k)\right\|} < \eta , \tag{14}$$

  where $0 < \eta < 1$.
- Criterion 2: For $i = 1, \ldots, m$, ϕ_i^k is replaceable by ϕ_i^{k-1} if $k \le p$ or

$$\frac{\left|\phi_i(x^k) - \phi_i^{k-1}\right|}{\max_{k-p+1 \le j \le k} \left|\phi_i(x^k) - \phi_i(x^{j-1})\right|} \le 1 , \tag{15}$$

  where p is a given positive integer.

Consider the additional assumption

A4) $\Phi(x)$ is Lipschitz continuous at x^*.

The authors proved the following

Theorem 2 *Let assumptions A1), A2) and A4) hold and let x^k be the sequence generated by GW algorithm. Then*

1) *x^k is locally quadratically convergent to x^* if Criterion 1 is used;*
2) *x^k is locally superlinearly convergent to x^* with R-order at least r_p, where r_p is the unique positive root of $t^{p+1} - t^p - 1 = 0$ if Criterion 2 is used.*

In order to obtain global convergence of the method, the authors modified the 'working approximation' Φ^k of $\Phi(x)$ to a $\widehat{\Phi}^k$, i. e., they modified the replacement criterion to ensure $A^\top \widehat{\Phi}^k A$ to be positive definite. But the modified globally convergent algorithm is only R-linearly convergent.

The authors' numerical results show the method takes less time to solve some types of problems than the modified Newton methods, but not always so.

Analysis of the Replacement Criterion

The GW algorithm opened a new field for unary optimization, but, to our knowledge, during nearly five years no more papers in this field were published. Recently (as of 1999), J.Z. Zhang, N.Y. Deng and L.H. Chen [10] discussed the efficiency of the GW algorithm. They asked whether it was more efficient than Newton's method when x^k approaches the solution x^*. They showed that, generally speaking, we can not expect the number $\|S^k\|$ to be small enough such that the computation cost to factorize B^{k+1} by rank-one updates is less than that to factorize $\nabla^2 f(x^{k+1})$ directly. The efficiency of GW algorithm heavily depends on the magnitude of the number $|S^k|$, so the frequency by which the replacement takes place is a key point. Based on this idea, the authors extended the Criterion 1 of GW algorithm to the so-called

- Criterion R_α: For $i = 1, \ldots, m$ we replace $\phi_i(x^{k+1})$ by ϕ_i^k if

$$\left|\phi_i(x^{k+1}) - \phi_i^k\right| < \sigma \left\|\nabla f(x^{k+1})\right\|^\alpha , \tag{16}$$

  where σ and α are two constants satisfying $\sigma > 0$ and $\alpha \in (0, 1]$.

It is easy to see that the GW's Criterion 1 is a special case of Criterion R_α with $\alpha = 1$ and $\sigma = \eta/(1 - \eta)$.

Consider the assumptions

A1) For $i = 1, \ldots, m$, $U_i(\cdot)$ is three times continuously differentiable;

A2) problem (1) has a solution x^* and $\nabla^2 f(x^*)$ is positive definite;

A3) the initial point x^0 is close enough to the solution x^*.

They proved two theorems:

Theorem 3 *Let assumptions A1)–A3) hold. Consider the sequence x^k generated by Algorithm R_α with $\alpha > \overline{\alpha}_{J+1}$, where J is a nonnegative integer and $\overline{\alpha}_{J+1}$ is the*

unique positive root of the equation

$$(1+\alpha)^{J+1}\alpha = 1\,. \tag{17}$$

Suppose that for a fixed i, we have
1) $d^3U_i(\alpha_i(x^*)/d\alpha_i^3 \neq 0$, and
2) there exists $\overline{K} > 0$ such that

$$\phi_i^k \neq \phi_i(x^*)\,, \forall k > \overline{K}\,. \tag{18}$$

Then
a) for any $K' > 0$, there exists an integer $k > K'$ such that

$$\phi_i^k = \phi_i(x^k)\,; \tag{19}$$

b) for any $\epsilon > 0$, there exists a $K > 0$ such that if

$$\overline{k} > K\,, \tag{20}$$

$$\phi_i^{\overline{k}} = \phi_i(x^{\overline{k}})\,, \tag{21}$$

and

$$\left| a_i^\top \frac{s^{\overline{k}}}{\left\| s^{\overline{k}} \right\|} \right| > \epsilon\,. \tag{22}$$

Then, for $x^k = x^{\overline{k}+1}, x^{\overline{k}+2}, \ldots, \phi_i(x^k)$ is replaceable by ϕ_i^{k-1} successively for at most J times.

Theorem 4 *The conditions are the same as Theorem 3. Suppose there exist M_1, $M_2 > 0$ such that when k is large enough,*

$$M_1 \left\| x^k - x^* \right\|^{1+\alpha} \leq \left\| x^{k+1} - x^* \right\| \tag{23}$$

$$\leq M_2 \left\| x^k - x^* \right\|^{1+\alpha}\,. \tag{24}$$

Then there exists $\overline{K} > 0$, such that if

$$\overline{k} > \overline{K} \tag{25}$$

and

$$\phi_i^{\overline{k}} = \phi_i(x^{\overline{k}})\,. \tag{26}$$

Then for $x^k = x^{\overline{k}+1}, x^{\overline{k}+2}, \ldots, \phi_i(x^k)$ is replaceable by ϕ_i^{k-1} successively for at least J times.

When $J = 0$, then the positive root of $(1 + \alpha)\alpha = 1$ is $\overline{\alpha}_1 = \frac{(\sqrt{5}-1)}{2}$. For GW's Criterion 1, $\alpha = 1 > \overline{\alpha}_1$, so from Theorem 3, $\phi_i(x^k)$ will never be replaceable by ϕ_i^{k-1} when k is large enough. So the GW algorithm with Criterion 1 cannot be expected to be more efficient than Newton method. The authors' numerical results also supported this conclusion.

But the authors pointed out, the idea of the GW algorithm to use the approximate Hessian B^k is promising, and efficient algorithms can hopefully be constructed.

Using the Trust Region Approach

In GW algorithm, in order to ensure global convergence, the authors modified the criteria such that the approximate Hessian B^k be positive definite. However, this prevents the algorithms from having a locally superlinear convergence, may lead to B^k being a poor approximation and may increase the computation cost. Motivated by this observation, Chen, Deng and Zhang [3] removed the requirement for B^k to be positive definite, and proposed two modified partial-update algorithms based on trust region stabilization. They also improved the replacement criteria of GW algorithm and constructed the matrix B^k in a different way such that it may be a better approximation to the Hessian.

Besides the replacement of the line search technique by the trust region strategy, the main differences between their algorithms and the GW algorithm are as follows:

a) As mentioned above, in order to get a better approximation to $\nabla^2 f(x^k)$, they abandoned the positive definiteness requirement for B^k. This change requires an efficient method to handle a trust region subproblem with indefinite matrix, so they used the indefinite dogleg method suggested in [11]. The main feature of the method is that it solves the TR subproblem approximately. It first uses a Bunch–Parlett (B-P) factorization for B^k, and then forms a dogleg curve. Instead of minimizing the objective function within the whole trust region, it makes a curvilinear search along the dogleg curve in the region.
b) They introduced a threshold value $l \approx n/6$ to control the algorithm to get the B-P factorization directly or to use a rank-one update, this motivated by the fact that the number of multiplications of the two methods is

$$c_n = \frac{1}{6}n^3 + O(n^2)$$

1 Set parameters $p > 0$, positive integers w and l. Choose the initial point $x^0 \in \mathbf{R}^n$. Set $k = 0$.

2 IF $\nabla f(x^k) = 0$, stop.

3 IF $k = 0$, go to Step 4; ELSE, define the set

$$S^k = \left\{ i : \left|\phi_i(x^k) - \phi_i^{k-1}\right| > \left|\left|\nabla f(x^k)\right|\right|^{1/p} \right\}. \tag{30}$$

IF

$$\left|S^k\right| \le w, \tag{31}$$

where $|\ S^k\ |$ is the number of elements in the set S^k, go to Step 5; otherwise go to Step 4.

4 Set

$$\phi_i^k = \phi_i(x^k), \quad i = 1, \ldots, m, \quad B^k = \nabla^2 f(x^k). \tag{32}$$

Compute the Choleski factorization $\nabla^2 f(x^k) = L_k D_k L_k^\top$ and the increment s^k by solving the Newton equation

$$\nabla^2 f(x^k) s = -\nabla f(x^k).$$

Set $x^{k+1} = x^k + s^k$. Go to Step 7.

5 Set

$$\phi_i^k = \begin{cases} \phi_i(x^k), & i \in S^k; \\ \phi_i^{k-1}, & i \notin S^k, \end{cases} \tag{33}$$

and

$$B^k = B^{k-1} + \sum_{i \in S^k} (\phi_i(x^k) - \phi_i^{k-1}) a_i a_i^\top. \tag{34}$$

Compute the Choleski factorization $B^k = L_k D_k L_k^\top$ which is obtained by successive $|\ S^k\ |$ rank-one corrections from the Choleski factorization

$$B^{k-1} = L_{k-1} D_{k-1} L_{k-1}^\top.$$

6 The increment $\tilde{s}^k$ is given by

$$\tilde{s}^k = \psi(\nabla^2 f(x^k),\ \nabla f(x^k), B^k, l), \tag{35}$$

where $\psi(\nabla^2 f(x^k),\ \nabla f(x^k), B^k, l)$ is a mapping defined by solving approximately the Newton equation

$$\nabla^2 f(x) s = -\nabla f(x) \tag{36}$$

with preconditioned conjugate gradient inner iterations, and the preconditioner is given by the inverse of B. The initial point is selected as $s = 0$. The number of inner iterations is l. The PCG method used here is a standard one, e.g. see [8, Algorithm 2.5.1]. Set $x^{k+1} = x^k + \tilde{s}^k$.

7 Set $k = k + 1$, and then go to Step 2.

◄ Numerical Methods for Unary Optimization, Algorithm 1 Deng-Wang-Zhang algorithm

and

$$c_1 = n^2 + O(n)$$

respectively, and $c_n/c_1 = n/6$. That is, if $k = 1$ or $|S^k| > l$, one performs the B-P factorization of B^k directly; otherwise, one uses the rank-one updates of the B-P factorization $|S^k|$ times to get the factorization of the matrix

$$B^k = B^{\tau_k} + \sum_{i \in S^k} (\phi_i^k - \phi_i^{\tau_k}) a_i a_i^\top$$

where τ_k is the index of the iteration at which the latest B-P factorization was performed.

c) They considered the effect of a_i on the replacement criteria. From the expression

$$\nabla^2 f(x) = \sum_{i=1}^m [\|a_i\|^2 \phi_i(x)] \left(\frac{a_i}{\|a_i\|}\right) \left(\frac{a_i}{\|a_i\|}\right)^\top \tag{27}$$

they define S^k by

$$S^k = \{ i : \frac{\|a_i\|^2 \left|\phi_i(x^k) - \phi_i^{\tau_k}\right|}{[\|a_i\|^2 \left|\phi_i(x^k) - \phi_i^{\tau_k}\right| + \min\{\gamma, \left\|\nabla f(x^k)\right\|\}]} > \eta, \quad 1 \le i \le m \}, \tag{28}$$

where $\gamma > 0$ is a constant. Then they gave two modified replacement criteria:

– Modified replacement criterion 1: For $k = 1$ and $i \in \{1, \ldots, m\}$, set $\phi_i^1 = \phi_i(x^1)$. For $k > 1$, define ϕ_i^k as follows:

1) If $|S^k| \le l$, set

$$\phi_i^k = \begin{cases} \phi_i(x^k) & \text{if } i \in S^k, \\ \phi_i^{\tau_k} & \text{if } i \notin S^k. \end{cases}$$

2) Otherwise, set

$$\phi_i^k = \phi_i(x^k).$$

2) Modified replacement criterion 2: The same as modified replacement criterion 1 except choose

an extra integer parameter $p > 0$ and if $k \le p$, S^k is the same; otherwise, define

$$S^k = \{i : \left|\phi_i(x^k) - \phi_i^{\tau_k}\right| \ge \max_{k-p+1\le j\le k} \left|\phi_i(x^k) - \phi_i(x^{j-1})\right|, \quad 0 \le i \le m\} . \tag{29}$$

Consider the assumptions

[A1] For $i = 1, \ldots, m$, $U(\cdot)$ is twice continuously differentiable;

[A2] For any real constant $\overline{f}$, the level set $\{x : f(x) \le \overline{f}\}$ is compact;

[A3] For $i = 1, \ldots, m$, the function $\phi_i(x)$ is Lipschitz continuous.

Under these assumptions the authors proved that the two algorithms are globally convergent, and if x^k converges to x^* and $\nabla^2 f(x^*)$ is positive definite, then the convergence rate of the first algorithm is quadratic, while for the second algorithm, x^k is locally superlinearly convergent to x^* with R-order at least r_p, where r_p is the unique positive root of $t^{p+1} - t^p - 1 = 0$.

The authors' numerical experiments showed that the two algorithms outperform the trust region method which uses GW's criteria 1 and 2.

An Inexact Newton Method Using Preconditioned Conjugate Gradient Techniques

More recently (1999), Deng, Wang and Zhang [6] developed Goldfarb and Wang's idea further and exploited the preconditioned conjugate gradient technique proposed in [5] and [4] to derive a new inexact Newton method. They showed that, when $n \ge 31$, the local behavior of this algorithm is superior to that of both the GW algorithms and the algorithm mentioned in the above section [3] from the efficiency point of view. Their algorithm model is given below.

Suppose problem (1) has a solution x^* and the following conditions are valid in a neighborhood of x^*:

[A1] For $i = 1, \ldots, m$, the second derivative $\phi_i(x)$ of $U_i(\cdot)$ defined by (5) is Lipschitz continuous with the constant L;

[A2] $\nabla^2 f(x^*) > 0$, i. e. the Hessian is symmetric positive definite.

The authors proved the following *local quadratic convergence theorem* about the Deng–Wang–Zhang algorithm.

Theorem 5 *Let Assumptions A1)–A2) hold. If $l \ge p$, then there are positive scalars δ and M such that if $\|x^0 - x^*\| \le \delta$, the Deng–Wang–Zhang Algorithm (Algorithm 1) is well-defined and, furthermore, the sequence $\{x^k\}$ generated by it satisfies*

$$\left\|x^{k+1} - x^*\right\| \le M \left\|x^k - x^*\right\|^2 .$$

In other words, the sequence $\{x^k\}$ converges to x^* with q-order at least 2.

The authors also studied the precisely quadratic convergence of the Deng–Wang–Zhang Algorithm, that is essential for the estimation of its computation cost. Consider the following additional assumption:

[A3] For any nonzero $h \in \mathbf{R}^n$, we have

$$\sum_{i=1}^{m} \phi_i'(x^*)(a_i^\top h)^2 a_i \ne 0 . \tag{37}$$

The authors proved

Theorem 6 *Consider the sequence $\{x^k\}$ generated by the Deng–Wang–Zhang Algorithm with $l > p$. Let Assumptions (A1)–(A3) hold. Then there is a positive scalar δ_1 and $M_1 > 0$ such that if $\|x^0 - x^*\| \le \delta_1$, then for any $k = 0, 1, \ldots$*

$$M_1 \left\|x^k - x^*\right\|^2 \le \left\|x^{k+1} - x^*\right\| \le M \left\|x^k - x^*\right\|^2 , \tag{38}$$

where M is defined in Theorem 5. In other words, the sequence $\{x^k\}$ converges to x^* with Q-order equal to 2.

In order to obtain an implementable algorithm, the authors specified the values of the parameters l, p and w in the Algorithm with the following purposes in mind:

1) retain the quadratic convergence;
2) keep the computation cost per iteration as little as possible.

Here the computation cost is only concerned with the arithmetic operations in solving the Newton equation in Step 4 and Steps 5–6. For simplicity, they only considered the number of multiplicative operations. For example, for Step 4, the arithmetic operations refer to the number Q_N of multiplications to compute the solution s^k to (36) by a direct Choleski factorization, which is

$$Q_N = Q_N(n) = \frac{n^3}{6} + \frac{3n^2}{2} - \frac{2n}{3} . \tag{39}$$

Numerical Methods for Unary Optimization, Table 1

$n =$	100	200	400	500	800	1000	2000
$w =$	0	0	0	0	0	0	0
$p =$	(2, 3)	(4, 5)	(4, 5)	(8, 9)	(8, 9)	(8, 9)	(16, 17)
$l =$	3	5	5	9	9	9	17
$\frac{\overline{Q}(n)}{Q_N(n)} \leq$	0.67	0.53	0.43	0.41	0.35	0.33	0.28

Similarly, the arithmetic operations in one conjugate gradient inner iteration in Step 6 refers to the number of multiplications, denoted by Q_{CG}, in calculating the right-hand side of (35), but with l there being replaced by 1:

$$Q_{CG} = Q_{CG}(n) = 2n^2 + 6n + 2\,. \tag{40}$$

Using the minimization of an upper bound of the average computation cost per iteration, the authors obtained the optimal parameter values of w, p and l, then established an implementable algorithm for problem (1) with $n \geq 9$ as follows.

The same as the Deng-Wang-Zhang algorithm, except that the parameters are specified as follows:

1 The parameter w: set $w = 0$.

2 The parameter p: when $19 \geq n \geq 9$, set $p \in (0, 1)$; when $30 \geq n \geq 20$, set $p \in (0, 2)$; when $n \geq 31$, set $p \in (2^{i^*}, 2^{i^*} + 1)$, where the integer i^* can be determined as follows: Let $N = [Q_N/Q_{CG}]$, divide interval $(0, N)$ by the points 2^i, $i = 1, \ldots,$ into subintervals $(0, 2], (2^1, 2^{1+1}], \ldots, (2^{j-1}, 2^j], (2^j, N]$, where $2^j + 1 \leq N \leq 2^{j+1}$, then compare the values

$$u(i) = \frac{Q_N}{1+i} + \frac{i(2^i + 1)Q_{CQ}}{1+i}$$

for $i = 1, \ldots, j$. The integer i^* is defined as the index corresponding to the smallest value u^* of $u(i)$: $u(i^*) = u^*$.

3 The parameter l: set $l = l(p)$, where $l(p)$ is defined by $p + 1 \geq l(p) > p$.

Then the authors proved the following theorem about the computation cost:

Theorem 7 *Let A1)–A3) hold. Then compared with the corresponding number $Q_N = Q_N(n)$ of Newton's method with Choleski factorization, the average arithmetic operations per iteration $\overline{Q} = \overline{Q}(n)$ of the improved Deng–Wang–Zhang algorithm has the following properties:*

1) When $30 \geq n \geq 9$, $\overline{Q}(n) \leq Q_N(n)$;

2) when $n \geq 31$, $\overline{Q}(n) < Q_N(n)$;

3) $\lim_{n\to\infty} \frac{\overline{Q}(n)}{Q_N(n)} \to 0$.

Corresponding to the theoretical result in Theorem 7, some numerical values are listed in Table 1. For example, for $n = 200$, we have $i^* = 2$, $p \in (4, 5)$, $l = 5$ and $u^* = u(2) = 0.53Q_N$, which means that the average cost per iteration of the improved Deng–Wang–Zhang algorithm is no more than 0.53 Q_N.

The authors also pointed out that the parameter selection problem in the Deng–Wang–Zhang algorithm is worth of further study from both theoretical and numerical points of view. Better performance than the one listed in Table 1 is expected due to a better parameter selection method.

See also

- ▶ Broyden Family of Methods and the BFGS Update
- ▶ Unconstrained Nonlinear Optimization: Newton–Cauchy Framework
- ▶ Unconstrained Optimization in Neural Network Training

References

1. Bandler W, Chen SH, Bienacki RM, Gao L, Madsen K, Yu H (1993) Robust circuit optimization using Huber functions. In: IEEE Trans Macrowave Theory and Techniques, pp 2279–2288

2. Byrd RH (1981/2) Algorithms for robust regression. In: Powell MJD (ed) Nonlinear Optimization. Acad. Press, New York, pp 79–84
3. Chen LH, Deng NY, Zhang JZ (1998) Modified partial-update Newton-type algorithms for unary optimization. J Optim Th Appl 97:385–406
4. Deng NY, Wang ZZ (1998) Can Newton's method be surpassed? Chinese Sci Bull 43:132–134
5. Deng NY, Wang ZZ (1998/9) Newton's method can be beaten. Quaderno DMSIA Univ. Bergamo, Bergamo
6. Deng NY, Wang ZZ, Zhang JZ (1999) An improved inexact Newton's method for unary optimization. Techn Report, Dept Math City Univ Hong Kong
7. Goldfarb D, Wang S (1993) Partial-update Newton method and unary factorable and partially separable optimization. SIAM Optim 3:382–397
8. Kelley CT (1995) Iterative methods for linear and nonlinear equations. SIAM, Philadelphia
9. McCormick GP, Sofer A (1991) Optimization with unary functions. Math Program 52:167–178
10. Zhang JZ, Deng NY, Chen LH (1999) On the replacement criteria for unary optimization. Dept Math City Univ Hong Kong, Techn Report MA-99-06
11. Zhang JZ, Xu CX (1999) A class of indefinite dogleg path methods for unconstrained optimization. SIAM J Optim 9(3):646–667

O

Oligopolistic Market Equilibrium

OME

Anna Nagurney
University Massachusetts, Amherst, USA

MSC2000: 91B06, 91B60

Article Outline

Keywords
The Classical Oligopoly Problem
A Spatial Oligopoly Model
See also
References

Keywords

Imperfect competition; Nash equilibrium; Game theory; Noncooperative behavior; Aspatial and spatial markets; Variational inequality formulations

An *oligopoly* consists of a finite number (usually few) firms involved in the production of a good. Oligopolies are a basic economic market structure, with examples ranging from large firms in automobile, computer, chemical, or mineral extraction industries to small firms with local markets. Oligopolies are examples of *imperfect competition* in that the producers or firms are sufficiently large that they affect the prices of the goods. A *monopoly*, on the other hand, consists of a single firm which has full control of the market.

Oligopoly theory dates to A. Cournot [1], who considered competition between two producers, the so-called *duopoly* problem, and is credited with being the first to study *noncooperative* behavior, in which the agents act in their own self-interest. In his study, the decisions made by the producers or firms are said to be in equilibrium if no producer can increase his income by unilateral action, given that the other producer does not alter his decision.

J.F. Nash [18,19] generalized Cournot's concept of an equilibrium for a behavioral model consisting of several agents or players, each acting in his own self-interest, which has come to be called a *noncooperative game*. Specifically, consider m players, each player i having at his disposal a strategy vector $x_i = \{x_{i1}, \ldots, x_{in}\}$ selected from a closed, convex set $K^i \subset \mathbf{R}^n$, with a utility function u_i: $K \to \mathbf{R}^1$, where $K = K^1 \times \ldots \times K^m \subset \mathbf{R}^{mn}$. The rationality postulate is that each player i selects a strategy vector $x_i \in K^i$ that maximizes his utility level $u_i(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_m)$ given the decisions $(x_j)_{j \neq i}$ of the other players. In this framework one then has:

Definition 1 (*Nash equilibrium*) A Nash equilibrium is a strategy vector $x^* = (x_1^*, \ldots, x_m^*) \in K$, such that

$$u_i(x_i^*, \widehat{x}_i^*) \geq u_i(x_i, \widehat{x}_i^*), \qquad \forall x_i \in K^i, \quad \forall i,$$

where $\widehat{x}_i^* = (x_1^*, \ldots, x_{i-1}^*, x_{i+1}^*, \ldots, x_m^*)$.

It has been shown (cf. [6,10]) that Nash equilibria satisfy variational inequalities. In the present context, under the assumptionthat each u_i is continuously differentiable on K and concave with respect to x_i, one has

Theorem 2 (*variational inequality formulation*) *Under the previous assumptions, x^* is a Nash equilibrium if and only if $x^* \in K$ is a solution of the variational inequality*

$$\langle F(x^*), x - x^* \rangle \geq 0, \qquad \forall x \in K,$$

where $F(x) \equiv (-\nabla_{x1} u_1(x), \ldots, -\nabla_{xm} u_m(x))$, *and is assumed to be a row vector, where* $\nabla_{xi} u_i(x)$ *denotes the gradient of* u_i *with respect to* x_i.

If the feasible set K is compact, then existence is guaranteed under the assumption that each utility function u_i is continuously differentiable. J.B. Rosen [22] proved existence under similar conditions. S. Karamardian [12] relaxed the assumption of compactness of K and provided a proof of existence and uniqueness of Nash equilibria under the strong monotonicity condition. As shown by D. Gabay and H. Moulin [6], the imposition of a coercivity condition on $F(x)$ will guarantee existence of a Nash equilibrium x^* even if the feasible set is no longer compact. Moreover, if $F(x)$ satisfies the strict monotonicity condition then uniqueness of x^* is guaranteed, provided that the equilibrium exists.

We begin with the presentation of a classical oligopoly model and then present a spatial oligopoly model which is related to the spatial price equilibrium problem.

The Classical Oligopoly Problem

We now describe the classical oligopoly problem (cf. [1,4]. [5,6,7,13,20,21]) in which there are m producers involved in the production of a homogeneous commodity. The quantity produced by firm i is denoted by q_i, with the production quantities grouped into a column vector $q \in \mathbf{R}^m$. Let f_i denote the cost of producing the commodity by firm i, and let ρ denote the demand price associated with the good. Assume that

$$f_i = f_i(q_i)$$

and

$$\rho = \rho\left(\sum_{i=1}^{m} q_i\right) .$$

The profit for firm i, u_i, which is the difference between the revenue and cost, can then be expressed as

$$u_i(q) = \rho\left(\sum_{i=1}^{m} q_i\right) q_i - f_i(q_i).$$

Given that the competitive mechanism is one of noncooperative behavior, one can write down immediately:

Theorem 3 (*variational inequality formulation*) *Assume that the profit function* $u_i(q)$ *for each firm* i *is concave with respect to* q_i, *and that* $u_i(q)$ *is continuously differentiable. Then* $q^* \in \boldsymbol{R}_+^m$ *is a Nash equilibrium if and only if it satisfies the variational inequality:*

$$\sum_{i=1}^{m}\left[\frac{\partial f_i(q_i^*)}{\partial q_i} - \frac{\partial \rho(\sum_{i=1}^{m} q_i^*)}{\partial q_i} q_i^* - \rho\left(\sum_{i=1}^{m} q_i^*\right)\right] \times [q_i - q_i^*] \geq 0, \qquad \forall q \in \mathbf{R}_+^m .$$

Example 4 In this oligopoly example there are three firms. The data are as follows: the producer cost functions are given by:

$$f_1(q_1) = q_1^2 + q_1 + 10 ,$$
$$f_2(q_2) = \frac{1}{2} q_2^2 + 4q_2 + 12 ,$$
$$f_3(q_3) = q_3^2 + \frac{1}{2} q_3 + 15 ,$$

and the inverse demand or price function is given by: $\rho(\sum_{i=1}^{3} q_i) = -\sum_{i=1}^{3} q_i + 5$.

The equilibrium production outputs are as follows:

$$q_1^* = \frac{23}{30} , \qquad q_2^* = 0 ,$$
$$q_3^* = \frac{14}{15} ; \qquad \sum_{i=1}^{3} q_i^* = \frac{17}{10} .$$

We now verify that the variational inequality is satisfied: $-\partial u_1(q^*)/\partial q_1$ is equal to zero, as is $-\partial u_3(q^*)/\partial q_3$, whereas $-\partial u_2(q^*)/\partial q_2 = 7/10$. Since both q_1^* and q_3^* are greater than zero, and $q_2^* = 0$, one sees that, indeed, the above variational inequality is satisfied.

Computational approaches can be found in [2,4,8,14,16,17], and the references therein.

In the special case where the production cost functions are quadratic (and separable) and the inverse demand or price function is linear, one can reformulate the Nash equilibrium conditions of the Cournot oligopoly problem as the solution to an optimization problem (see [16,23]).

A Spatial Oligopoly Model

We now describe a generalized version of the oligopoly model due to S.C. Dafermos and A. Nagurney [3] (see, also, [11]), which is *spatial* in that the firms are now

located in different regions and there are transportation costs associated with shipping the commodity between the producers and the consumers. For the relationship between this model and the perfectly competitive spatial price equilibrium problem, see [3]. For algorithms for the computation of solutions to this model, see [15] (see, also, e. g., [9,16]). For additional background, including qualitative properties and references, see [16]. In [17] dynamic oligopolistic models are presented both spatial and aspatial (see also [4] and [20]) and stability analysis results given.

Assume that there are m firms and n demand markets that are generally spatially separated. Assume that the homogeneous commodity is produced by the m firms and is consumed at the n markets. As before, let q_i denote the nonnegative commodity output produced by firm i and now let d_j denote the demand for the commodity at demand market j. Let Q_{ij} denote the nonnegative commodity shipment from supply market i to demand market j. Group the production outputs into a column vector $q \in \mathbf{R}^m_+$, the demands into a column vector $d \in \mathbf{R}^n_+$, and the commodity shipments into a column vector $Q \in \mathbf{R}^{mn}+$.

The following *conservation of flow* equations must hold:

$$q_i = \sum_{j=1}^{n} Q_{ij} , \qquad \forall i ,$$

$$d_j = \sum_{i=1}^{m} Q_{ij} , \qquad \forall j ,$$

where $Q_{ij} \geq 0$, $\forall i, j$.

As before, we associate with each firm i a production cost f_i, but allow now for the more general situation where the production cost of a firm i may depend upon the entire production pattern, that is,

$$f_i = f_i(q) .$$

Similarly, allow the demand price for the commodity at a demand market to depend, in general, upon the entire consumption pattern, that is,

$$\rho_j = \rho_j(d).$$

Let c_{ij} denote the transaction cost, which includes the transportation cost, associated with trading (shipping) the commodity between firm i and demand market j. Here we permit the transaction cost to depend, in general, upon the entire shipment pattern, that is,

$$c_{ij} = c_{ij}(Q) .$$

The profit u_i of firm i is then given by:

$$u_i = \sum_{j=1}^{n} \rho_j c_{ij} - f_i - \sum_{j=1}^{n} c_{ij} Q_{ij} ,$$

which, in view of the conservation of flow equations and the functions, one may write as

$$u = u(Q) .$$

Now consider the usual oligopolistic market mechanism, in which the m firms supply the commodity in a noncooperative fashion, each one trying to maximize his own profit. We seek to determine a nonnegative commodity distribution pattern Q for which the m firms will be in a state of equilibrium as defined below.

Definition 5 (*spatial Cournot–Nash equilibrium*) A commodity shipment distribution $Q^* \in \mathbf{R}^{mn}_+$ is said to constitute a Cournot–Nash equilibrium if for each firm i, $i = 1, \ldots, m$,

$$u_i(Q_i^*, \widehat{Q}_i^*) \geq u_i(Q_i, \widehat{Q}_i^*) , \qquad \forall Q_i \in \mathbf{R}^n_+ ,$$

where

$$Q_i \equiv \{Q_{i1}, \ldots, Q_{in}\} ,$$
$$\widehat{Q}_i^* \equiv (Q_1^*, \ldots, Q_{i-1}^*, Q_{i+1}^*, \ldots, Q_m^*) .$$

The variational inequality formulation of the Cournot–Nash equilibrium is given in the following theorem.

Theorem 6 (*variational inequality formulation*; [3]) *Assume that for each firm i the profit function $u_i(Q)$ is concave with respect to the variables $\{Q_{i1}, \ldots, Q_{in}\}$, and continuously differentiable. Then $Q^* \in \boldsymbol{R}^{mn}_+$ is a Cournot–Nash equilibrium if and only if it satisfies the variational inequality*

$$-\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{\partial u_i(Q^*)}{\partial Q_{ij}} \times (Q_{ij} - Q_{ij}^*) \geq 0 , \qquad \forall Q \in \mathbf{R}^{mn}_+ .$$

Using the expressions for the utility functions for this model and the conservation of flow equations this varia-

tional inequality may be rewritten as:

$$\sum_{i=1}^{m} \frac{\partial f_i(q^*)}{\partial q_i} \times (q_i - q_i^*) + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}(Q^*) \times (Q_{ij} - Q_{ij}^*) - \sum_{j=1}^{n} \rho_j(d^*) \times (d_j - d_j^*) - \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{l=1}^{n} \left[\frac{\partial \rho_l(d^*)}{\partial d_j} - \frac{\partial c_{il}(Q^*)}{\partial Q_{ij}} \right] \times Q_{il}^*(Q_{ij} - Q_{ij}^*) \geq 0 \,,$$

$$\forall (q, Q, d) \in K \,,$$

where $K \equiv \{(q, Q, d)$*:* $Q \geq 0$*, and the conservation of flow equations hold}.*

Note that, in the special case, where there is only a single demand market and the transaction costs are identically equal to zero, this variational inequality collapses to the variational inequality governing the *aspatial* or the classical oligopoly problem.

See also

- ► Equilibrium Networks
- ► Financial Equilibrium
- ► Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ► Spatial Price Equilibrium
- ► Traffic Network Equilibrium
- ► Walrasian Price Equilibrium

References

1. Cournot A (1897) Researches into the mathematical principles of the theory of wealth. MacMillan, New York. English edition: (1838)
2. Dafermos S (1983) An iterative scheme for variational inequalities. Math Program 26:40–47
3. Dafermos S, Nagurney A (1987) Oligopolistic and competitive behavior of spatially separated markets. Regional Sci and Urban Economics 17:245–254
4. Flam SP, Ben-Israel A (1990) A continuous approach to oligopolistic market equilibrium. Oper Res 38:1045–1051
5. Friedman J (1977) Oligopoly and the theory of games. North-Holland, Amsterdam
6. Gabay D, Moulin H (1980) On the uniqueness and stability of Nash-equilibria in noncooperative games. In: Bensoussan A, Kleindorfer P, Tapiero CS (eds) Applied Stochastic Control in Econometrics and Management Sci. North-Holland, Amsterdam, pp 271–294
7. Gabzewicz J, Vial JP (1972) Oligopoly á la Cournot in a general equilibrium analysis. J Econom Theory 14:381–400
8. Harker PT (1984) A variational inequality approach fror the determination of oligopolistic market equilibrium. Math Program 30(105–111
9. Harker PT (1986) Alternative models of spatial competition. Oper Res 34:410–425
10. Hartman P, Stampacchia G (1966) On some nonlinear elliptic differential functional equations. Acta Math 115:271–310
11. Haurie A, Marcotte P (1985) On the relationship between Nash–Cournot and Wardrop equilibria. Networks 15:295–308
12. Karamardian S (1969) Nonlinear complementarity problem with applications. J Optim Th Appl 4:87–98; Part II: 167–181
13. Manas M (1972) A linear oligopoly game. Econometrica 40:917–922
14. Murphy FH, Sherali HD, Soyster AL (1982) A mathematical programming approach for determining oligopolistic market equilibrium. Math Program 24:92–106
15. Nagurney A (1988) Algorithms for oligopolistic market equilibrium problems. Regional Sci and Urban Economics 18:425–445
16. Nagurney A (1999) Network economics: A variational inequality approach, 2nd edn. Kluwer, Dordrecht
17. Nagurney A, Zhang D (1996) Projected dynamical systems and variational inequalities with applications. Kluwer, Dordrecht
18. Nash JF (1950) Equilibrium points in n-person games. Proc Nat Acad Sci USA 36:48–49
19. Nash JF (1951) Noncooperative games. Ann of Math 54:286–298
20. Okuguchi K (1976) Expectations and stability in oligopoly models. Lecture Notes Economics and Math Systems, vol 138. Springer, Berlin
21. Okuguchi K, Szidarovsky F (1990) The theory of oligopoly with multi-product firms product firms. LectureNotes Economics and Math Systems, vol 342. Springer, Berlin
22. Rosen JB (1965) Existence and uniqueness of equilibrium points for concave n-person games. Econometrica 33:520–533
23. Spence M (1976) The implicit maximization of a function in monopolistically competitive markets. Harvard Inst Economic Res Discussion Paper, vol 461. Harvard University Press, Cambridge

Operations Research

B. D. Sivazlian
University Florida, Gainesville, USA

MSC2000: 90C27

Article Outline

Keywords
See also
References

Keywords

Operations research

Operations research originated circa 1945. The Webster's Ninth New Collegiate Dictionary (1989) defines it as: The application of scientific and especially mathematical methods to the study and analysis of problems involving complex systems (as firm management, economic planning, and the waging of war).

Case Institute of Technology (presently part of Case Western Reserve Univ.) located in Cleveland, Ohio, USA, was the first institution of higher learning in the world to establish a graduate program in Operations Research leading towards the MSc and PhD degrees. The first PhD degree in operations research in the USA was awarded to L. Friedman circa 1957, at Case. In the 1950s and early 1960s, Case was one of the leading institutions in the field of operations research. The famous operations research group at Case was well established and reputed internationally. The operations research philosophy at Case was essentially based on the premise that operations research is science. An appropriate definition of operations research is: The application of scientific methods to analyze, model, solve, and control human-created or management problems using mainly quantitative techniques [2]. Operations research provides the decision maker either with an optimum option from among a set of feasible alternative courses of action or with an optimum allocation of limited resources so as to minimize or maximize a given criterion or objective function [1]. The seven steps used in order to carry on an operations research study are:

1) Observation.
2) Analysis of the present systems. This includes:
   a) qualitative analysis, quantitative analysis and data collection;
   b) components to be incorporated such as: man, machine, material, management, money, consumer, competitors, public, government, safety, reliability, aesthetics, ethics.
3) Definition and formulation of the problem. These must involve:
   a) the decision-maker;
   b) the objectives or criteria used;
   c) the environmental or system constraints;
   d) the feasible alternative courses of action.
4) Construction of a model (synthesis/design). Four models are possible:
   a) mathematical (or symbolic or abstract) model;
   b) iconic model (e. g. CAD, graphics, drawings);
   c) analog model;
   d) simulation model.

   As an example, the construction of a mathematical model involves abstracting the operation of the system including:
   a) defining and quantifying
      i) the decision or control variables;
      ii) the parameters or uncontrollable variables;
   b) establishing functional relationships between variables and parameters;
   c) developing a mathematical function for the criterion (objective function to be minimized or maximized). One must specify:
      i) the horizon period;
      ii) the units;
      iii) the measures used such as: cost or profit (e. g. average, total, discounted), reliability, availability;
   d) setting up the constraints.
5) Derivation of a solution to the problem. This typically consists in selecting the best from among the set of feasible alternative courses of action to meet the defined objective function and satisfy the constraints through optimization techniques and tools using analytic or numerical procedures.
6) Testing and/or Validation. This involves:
   a) validation of data;
   b) validation of the model;
   c) testing the solution. The testing methodology uses:
      i) retrospective testing;
      ii) prospective testing;
      iii) simulation (accelerated testing);
      iv) others.
7) Control and implementation of the solution. The control methodology uses, for example:
   a) sensitivity analysis;

b) statistical quality control.

The implementation methodology uses, for example:

a) statistical inference;

b) sampling techniques.

Let me say that the above definition of operations research and the outlined methodology were not exactly the ones proposed in the late 1950s. However, they may be considered close enough for all practical purposes. One may consider to what extent the definition and the methodology have been altered throughout the years. In my opinion very little! In the foreword [3] Aristotle is quoted: 'And the science which knows to what end each thing must be done is the most authoritative of the sciences, and more authoritative than any ancillary science; and the end is the good of that thing, and in general the supreme good in the whole of nature'.

See also

► History of Optimization

References

1. Ackoff R (1962) Scientific method: Optimizing applied to research decisions. Wiley, New York
2. Churchman W, Ackoff R, Arnoff L (1957) Introduction to operations research. Wiley, New York
3. Kaufman A (1963) Methods and models of operations research. Prentice-Hall, Englewood Cliffs

Operations Research and Financial Markets

JOHN L.G. BOARD[1], CHARLES M.S. SUTCLIFFE[2], WILLIAM T. ZIEMBA[3]

[1] Department Accounting and Finance, The London School of Economics and Political Sci., London, UK

[2] Accounting and Finance Division, The School of Management The University Southampton, Southampton, UK

[3] Management Sci. Division Fac. Commerce and Business Admin., University British Columbia, Vancouver, Canada

MSC2000: 90C27

Article Outline

Keywords
Portfolio Theory
Pricing Derivatives
Trading Tactics
Funding Decisions
Strategic Problems
Regulatory and Legal Problems
Economic Understanding
Conclusions
References

Keywords

Finance; Mathematical programming; Simulation; Modeling; Applications; challenges for OR; Data envelopment analysis; Dynamic programming; Fractional programming; Goal programming; Linear programming; Markov chains; Monte-Carlo sampling and variance reduction; Nonlinear programming; Portfolio theory; Mean-variance; Quadratic programming; Stochastic programming

From the 1950s onwards, a strong relationship between operations research, OR, and finance has developed, resulting in a large and rapidly growing literature. Although most applications have been of OR techniques to finance, finance problems have also stimulated the development and refinement of OR techniques.

Finance problems, and especially those relating to financial markets, are particularly well suited to analysis using OR techniques. These problems are generally separable and well defined, have a clear objective (often to maximise profit or minimise risk), and have variables which are quantified in monetary terms. The relationships between the variables in finance models are usually stable and well defined, so that the resulting OR model is a good representation of the problem. As there are few concerns about human behavior ruling out the implementation of some solutions, the solutions produced by the analysis can usually be implemented. In addition, large amounts of data, both historic and real time, are readily available and can be used in OR models. Some finance problems involve very large sums of money, so that even a very small improvement in the quality of the solution is profitable to implement.

This review describes the application of OR to problems in the analysis of financial markets (e. g., the markets for debt, equity and foreign exchange markets and the corresponding derivatives markets). For a review of the application of OR to other areas of finance, such as: the management of the firm's finances: working capital management, capital investment, multinational taxation, and financial planning models (such as those developed for banks) see [3].

Portfolio Theory

A seminal application of OR techniques to finance was by H.M. Markowitz [67,68] when he specified the portfolio problem in terms of optimization over the means and variances of the assets available, and proposed the solution of this problem through quadratic programming, see [11] for a survey. In addition to specifying the portfolio problem in terms of OR techniques, Markowitz also developed solution algorithms for more general quadratic programming problems. This provides an example of the interaction between OR techniques and finance, with the former sometimes being adapted to meet the needs of the latter.

Although the most obvious application of portfolio theory is to the choice of equity portfolios, and empirical papers, e. g., [10,81] have used quadratic programming to compute efficient equity portfolios, the technique can be applied more widely (e. g., to portfolios of currencies, bonds, or commercial loans). Multiperiod portfolio problems have been specified as dynamic programming problems [35], while [75], uses a stochastic generalized network model.

OR researchers have also modified or replaced the quadratic programming approach to portfolio problems, often by explicitly specifying the relevant utility function and using stochastic linear programming with recourse to model risk in a multiperiod framework. For example, in [15] forming bond portfolios is proposed to maximize their expected value, using stochastic linear programming to allow for interest rate risk. Early references in this area are [50,59,109,110]. The scenarios included in portfolio models may be generated by Monte-Carlo simulation, prior to the use of stochastic programming to maximise expected utility, e. g., [37,97,102,103] and [105], where this approach is applied to form portfolios of mortgage backed securities.

The investment policy of a pension fund can be formulated using asset-liability management models that allow for the correlations between the values of the fund's assets and liabilities. While these problems can be formulated using quadratic programming, they have usually been solved in other ways (see [113]). For example, in [74] it is assumed that the objective was to maximise the expected value of a nonlinear utility of wealth function, and specified the problem as a nonlinear network problem, with the simulation of future pension fund liabilities. Similar asset-liability problems are also faced by insurance companies, for example [19,20,21] formulate this problem for a Japanese insurance company. See also [50,59,76]. In [52] it is pointed out that the use of Monte-Carlo simulation can bias the results by including arbitrage opportunities in the sampled scenarios. To avoid this, an arbitrage-free event tree is aggregated before its inclusion in a multistage stochastic programming model of the asset-liability problem.

Another application of quadratic programming is generalized hedging, in which the objective is usually to minimise the variance of a portfolio of a given set of assets and the chosen hedging instruments. If the hedging instruments include options, this introduces a nonlinearity into the hedging decision, and [77] devises a nonlinear programming model to hedge foreign currency exposure using a mixture of currency forward and options contracts. Similarly, quadratic programming has been used to construct index tracking portfolios, where the purpose is to select a portfolio of assets (e. g., equities or bonds) which, when combined with a matching short position in the index to be tracked, has minimum risk [69,70,86,88]. Multistage stochastic programming with recourse, in conjunction with Monte-Carlo simulation to generate the scenarios, has been used in [97] and [105] to track an index of mortgage backed securities.

A related problem is that of portfolio immunization in which the objective is to construct a portfolio of interest rate dependent securities whose value is the same as some target asset (usually another interest rate dependent asset). (There is also a literature on managing the assets and liabilities held by banks (which are taken to exclude equities), where the objective is usually to maximise the value (or expected value) of the portfolio over one (or many) time periods (net of penalty costs from constraint target violations), subject to re-

strictions of the total investment, maximum capital loss and various bank regulations.) By matching the duration of the portfolio with that of the target asset, the portfolio is immunized against small parallel shifts in the yield curve (which shows the interest rates for different maturities), [36,55,78] and [2]. These immunization studies use a risk measure which does not involve squares or cross products of the decision variables, and so linear programming, not quadratic programming, is the solution technique.

In some applications of portfolio theory, the decision variables must be integer. Quadratic integer programming is used to compute hedging strategies involving futures in [82] and [89]. A. Shapiro, 1988, used stochastic integer programming with recourse to construct bond portfolios that allow for some of the bonds being called (or redeemed early) if interest rates are low.

Some authors have argued that formulation and solving quadratic programming portfolio problems is too onerous, and proposed simplified solution techniques. W.F. Sharpe [91] proposed a single index model which can be solved by the use of special purpose quadratic programming algorithms. When each asset represents only a small proportion of the portfolio, he [92] showed that his single index model can be treated as having a linear objective function. In 1971, he suggested using a piecewise linear approximation to the quadratic objective function, enabling the application of linear programming to solve portfolio problems. Another proposal is to minimize the mean absolute deviation (MAD), which can be solved using linear programming, rather than quadratic programming (e. g., [53,54,101,106], and [100]). Another approach is to specify the problem as choosing between a range of pre-specified equity portfolios using data envelopment analysis [84]. A further approach is to reformulate the portfolio problem as a nonlinear generalized network model for which efficient solution algorithms exist [73].

Portfolio problems, with the twin objectives of maximising returns and minimising risk, can also be viewed as goal programming problems with two goals. Additional goals can be introduced, and a number of authors have solved portfolio problems using goal programming, among them [57,58], and [61]. Capital growth theory, see [40] for a survey, has been used in a number of applications for the optimal investment of repeated investments over time, see e. g. [64,65,66,111].

Pricing Derivatives

It is very important when trading in financial markets to have a good model for valuing the asset being traded, and OR techniques have made a substantial contribution in this area. Indeed, the very rapid growth of these markets is partly due to the application of OR techniques in pricing models.

P.P. Boyle [12] proposed the use of Monte-Carlo simulation as an alternative to the binomial model for pricing options for which a closed form solution is not readily available. Monte-Carlo simulation has the advantage over the binomial model that its convergence rate is independent of the number of state variables (e. g., the number of underlying asset prices and interest rates), while that of the binomial model is exponential in the number of state variables. Simulation is used to generate paths for the price of the underlying asset until maturity. The cash flows from the option for each path, weighted by their risk neutral probabilities, are discounted back to the present using the risk free rate, allowing the average present value across all the sample paths to be computed, thus yielding the current price of the option [14]. Risk neutral probabilities are inferred from prices assuming that investors have linear utility functions. Monte-Carlo simulation can also be used to compute various option price sensitivities, which include the hedge ratio. These sensitivities, or 'Greeks', are essential for many trading strategies [17].

Until recently it was thought that Monte-Carlo simulation could not be used to price American style options which can be exercised at any time before the option expires, because no closed form solutions for their price exist. This is a major problem, as the majority of options are American style. However, progress is being made in developing Monte-Carlo simulation techniques for pricing American style options [18,39]. Options have also been priced using finite difference approximations, and it has been proposed, [27] and [28], to use of linear programming to solve the finite difference approximations to the price of American style put options. In addition, American style options can be priced using dynamic programming, [29].

Provided a price history is available, a neural network can be trained to produce prices using a specified set of inputs, which can then be used for the out-of-sample pricing [49] of securities.

Mortgage backed securities (MBS) are created by the securitization of a pool of mortgages. For any specific mortgage, the borrower has the right to repay the loan early – the prepayment option, or may default on the payments of capital and interest. Thus MBS are hybrid securities, as they are variable interest rate securities with an early exercise option. Monte-Carlo simulation can be used to generate interest rate paths for future years. Forecasts of the mortgage prepayment rates then permit the computation of the cash flows from each interest rate path, and these sequences of cash flows are used to value the MBS [6,13,104]. This procedure, which can be used to identify mispriced MBS in real time, is computationally demanding and parallel (and massively parallel) and distributed processing have been used in the solution of the problem. Simulation has also been used to price collateralized mortgage obligations or CMOs [80]. Other hybrid securities, such as callable and putable bonds and convertible bonds face similar valuation problems to MBS and require similarly intensive solution methods.

There is an active secondary market in loan portfolios which may carry a significant default risk. In [26] a Markov chain analysis with 14 loan performance states is used and Monte-Carlo simulation is performed to generate the probability distribution of the present value of loan portfolios.

Trading Tactics

Besides pricing financial securities, traders are interested in finding imperfections in financial markets which can be exploited to make profits. See [51] for a survey of equity anomalies. One aspect of this is the search for weak form inefficiency (i. e. that an asset's past prices can be used as the basis of a profitable trading rule). An early attempt to find such exploitable regularities in stock prices is the use of Markov chains [30,31]. Such strategies have been found in horserace betting markets, see [42,43,44,45], and the survey book [41].

Arbitrageurs seek to exploit small price discrepancies to give riskless profits. Network models have been used to find arbitrage opportunities between sets of currencies ([22,55,73,75]). This problem can be specified as a maximal flow network, where the aim is to maximise the flow of funds out of the network, or as a shortest path network. A risk arbitrage, convergence type hedgefund trade on discrepancies between various markets for Nikkei puts is described in [94].

There has been a growing interest in using artificial intelligence based techniques (expert systems, neural networks, genetic algorithms, fuzzy logic and inductive learning) to develop trading strategies for financial markets (e. g., [38,85,96,99]). Such approaches have the advantage that they can pick up nonlinear dynamics, and require little prior specification of the relationships involved.

Funding Decisions

OR techniques have also been used to help firms determine the most appropriate method by which to raise capital from the financial markets. In [16] a chance constrained linear programming model is put forward to compute the values of the debt-equity ratio each period that maximize the value of the firm. Other studies have specified the choice between various types of funding as a linear goal programming problem [48,60].

A different approach to the debt problem is to assume that the firm has found its desired debt-equity ratio, and is purely concerned with raising the requisite debt as cheaply as possible. In this case, debt can be treated like any other input to the productive process, and inventory models used to determine the optimal 'reorder' times and quantities [9,62].

The design of callable bonds has been addressed in [23,24], using nonlinear programming, while [47] uses a simulated annealing algorithm. Firms which have issued callable debt face the bond-scheduling problem, in which they must decide when to call (repay) the existing debt and refinance it with a new issue, presumably at a lower cost. This dynamic programming problem has been modeled in [35,56] and [98].

Finally, the problem facing borrowers of choosing between alternative mortgage contracts (e. g., fixed rate, variable rate and adjustable rate mortgages) has been modeled using decision trees [46,63].

Strategic Problems

In recent years, some of the decisions facing traders and market makers in financial markets have been analysed using game theory [32,79]. Traders in stock markets seek to trade at the most attractive prices and large

trades are often broken up into a sequence of smaller trades in an effort to minimise the price impact. This can be viewed as a strategic problem, and [8] uses stochastic dynamic programming to compute an optimal trading strategy.

In [83] game theory is applied to the situation where a company has two major shareholders, and a large number of very small shareholders. This can be modeled as an oceanic game, in which the two large players behave strategically while the many small shareholders (the ocean) do not. This approach can be used to derive the highest price a large shareholder will pay in the market for corporate control.

Regulatory and Legal Problems

Financial regulators have become increasingly concerned about financial markets with their very large and rapid international financial flows. OR techniques have proved useful in regulating the capital reserves held by banks and other financial institutions to cover their risk exposure. OR techniques have also been used to ensure compliance with various legal requirements by designing appropriate strategies, and to solve other legal problems relating to financial markets.

A key regulatory issue is determining the capital required by financial institutions to underpin their activities in financial markets. An increasingly popular approach to this problem is the value at risk (VAR), which involves quantification of the lower tail of the probability distribution of outcomes from the firm's portfolio. Portfolios usually include options (or financial securities with option-like characteristics), and these have highly asymmetric payoffs. For such securities, analytical solutions to finding the probabilities in the lower tail of the payoff distribution are unreliable. Riskmetrics^{TM} uses approximations based on 'the Greeks' for options that are at or near the money (i. e. the current price of the underlying asset is close to the price at which the option can be exercised), and Monte-Carlo simulation for other options positions [72]. (A related application of Monte-Carlo simulation is stress testing, which quantifies the sensitivity of a portfolio to specified, often adverse, market scenarios.) Some securities are also subject to credit risk, which has a highly nonnormal distribution for all instruments. Therefore, Monte-Carlo simulation is relevant to modeling the credit risk of portfolios of financial instruments (e. g., loans, letters of credit, bonds, trade credit, swaps, forwards) as in CreditMetrics^{TM} [71]. Y. Zhao and Ziemba [107,108] discuss how to model downsized risk in continuous and discrete time models.

Data envelopment analysis has been used to assist in bank regulation by measuring bank efficiency, which is then used to predict bank failure [4,5].

Traders are required to put up margin when they trade options, see [87] for a linear programming model in which the problem was modeled as a transportation problem.

An extensive set of rules governs the way in which a 'to-be-announced' MBS can be structured, leading to a complexproblem in devising a feasible solution. This can be specified as a complicated integer programming problem (with the objective of maximising the originator's profit). Collateralized mortgage obligations (CMOs) also involve the securitization of a mortgage pool, but in this case the pool is structured into a series of bonds (or tranches), each with a different maturity and risks. See [25] for a complex zero-one programming model for solving this problem, with the objective of maximizing the proceeds from the issue.

In [90] a linear programming formulation is proposed to establish the maximum loss that investors could have sustained from trading in a company's shares. This figure can then be used by the company's lawyers in lawsuits claiming damages from a misleading statement by the company.

In August 1982, the Kuwait Stock Market collapsed leaving $94 billion of debt to be resolved. This led to the problem of devising a fair method for distributing the assets seized from insolvent brokers among the other brokers and private investors. This problem was solved using linear programming, which reduced the total unresolved debt to $20 billion, saving an estimated $10.34 billion in lawyer's fees [33,34,95].

Economic Understanding

OR can help in trying to understand the economic forces shaping the finance sector. Using a linear programming model of a bank, [7] employs annual data to compute movements in the shadow prices of the various constraints. They suggested that a rise in the

shadow price of the deposits constraint led to the financial innovation of negotiable CDs.

Arbitrage pricing theory (APT) seeks to identify the factors which affect asset returns. Most tests of the APT use factor analysis, and have difficulty in determining the number and definition of the factors that influence asset returns. To overcome these problems [1] suggests using a neural network, which also has the advantage that the results are distribution free.

Conclusions

Mathematical programming is the OR technique that has been most widely applied in financial markets. Most types of mathematical programming have been employed – linear, quadratic, nonlinear, integer, goal, chance constrained, stochastic, fractional, DEA and dynamic. Monte-Carlo simulation is also widely used in financial markets – mainly to value exotic options and securities with embedded options, and to estimate the VAR for various financial institutions. In some cases the use of OR techniques has influenced the way financial markets function since they permit traders to make better decisions in less time. For example, exotic options would trade with much wider bid-ask spreads, if they traded at all, in the absence of the accurate prices computed using Monte-Carlo simulation.

Other OR techniques are less used in financial markets. Arbitrage and multiperiod portfolio problems have been formulated as network models, while market efficiency has been tested using neural networks. Game theory has been applied to battles for corporate control, decision trees to analyse mortgage choice, inventory models to set the size and timing of corporate bond issues, and Markov chains to valuing loan portfolios and testing market efficiency. One important OR technique – queueing theory – has found little application in financial markets

This review has shown that OR techniques have been usefully applied to portfolio problems and the accurate pricing of complex financial instruments. They are also used by financial regulators and financial institutions in setting capital adequacy standards. It is clear from this that OR techniques play an important role in financial markets and that this role is likely to increase over time.

References

1. Ahmadi H (1993) Testability of the arbitrage pricing theory by neural networks. In: Trippi RR, Turban E (eds) Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance. Probus, Chicago, pp 421–432
2. Alexander GJ, Resnick BG (March 1985) Using linear and goal programming to immunize bond portfolios. J Banking and Finance 9(1):35–54
3. Ashford RW, Berry RH, Dyson RG (1988) Operational research & financial management. Europ J Oper Res 36(2):143–152
4. Barr RS, Seiford LM, Siems TF (December 1993) An envelopment analysis approach to measuring the managerial efficiency of banks. Ann Oper Res 45(1/4):1–19
5. Bauer PW, Berger AN, Ferrierand D.B. Humphrey GD (1998) Consistency conditions for regulatory analysis of financial institutions: A comparison of frontier efficiency methods. J Econom and Business 50(2):85–114
6. Ben-Dov Y, Hayre L, Pica V (January–February 1992) Mortgage valuation models at prudential securities. Interfaces 22(1):55–71
7. Ben-Horim M, Silber WL (November 1977) Financial innovation: A linear programming approach. J Banking and Finance 1(3):277–296
8. Bertsimas D, Lo AW (April 1998) Optimal control of execution costs. J Financial Markets 1(1):1–50
9. Bierman H (December 1966) The bond size decision. J Financial and Quantitative Anal 1(4):1–14
10. Board JLG, Sutcliffe CMS (March 1994) Estimation methods in portfolio selection and the effectiveness of short sales restrictions: UK evidence. Managem Sci 40(4):516–534
11. Board JLG, Sutcliffe CMS, Ziemba WT (2000) Portfolio theory: Mean variance. In: Gass SI, Harris CM (eds) Encycl. Oper. Res. and Management Sci. Kluwer, Dordrecht
12. Boyle PP (May 1977) Options: A Monte Carlo approach. J Financial Economics 4(3):323–338
13. Boyle PP (May-June 1989) Valuing Canadian mortgage backed securities. Financial Analysts J 45(3):55–60
14. Boyle PP, Broadie M, Glasserman P (June 1997) Monte Carlo methods for security pricing. J Econom Dynam Control 21(8–9):1267–1321
15. Bradley SP, Crane DB (October 1972) A dynamic model for bond portfolio management. Managem Sci 19(2):139–151
16. Brick IE, Mellon WG, Surkis J, Mohl M (March 1983) Optimal capital structure: A multi period programming model for use in financial planning. J Banking and Finance 7(1):45–67
17. Broadie M, Glasserman P (February 1996) Estimating security price derivatives using simulation. Managem Sci 42(2):269–285

18. Broadie M, Glasserman P (June 1997) Pricing American style securities using simulation. J Econom Dynam Control 21(8–9):1323–1352
19. Cariño DR, Kent T, Myers DH, Stacy C, Sylvanus M, Turner AL, Watanabe K, Ziemba WT (January-February 1994) The Russell–Yasuda Kasai model: An asset-liability model for a Japanese insurance company using multistage stochastic programming. Interfaces 24(1):29–49 Reprinted in: Ziemba WT, Mulvey JM (eds) Worldwide asset and liability modelling. Cambridge Univ.Press, 1998
20. Cariño DR, Ziemba WT (July-August 1998) Formulation of the Russell Yasuda Kasai financial planning model. Oper Res 46(4):433–449
21. Cariño DR, Myers D, Ziemba WT (July–August 1998) Concepts, technical issues and uses of the Russell Yasuda Kasai model. Oper Res 46(4):450–462
22. Christofides N, Hewins RD, Salkin GR (September 1979) Graph theoretic approaches to foreign exchange. J Financial and Quantitative Anal 14(3):481–500
23. Consiglio A, Zenios SA (1997) High performance computing for the computer aided design of financial products. In: Grandinetti L, Kowalik J, Vajtersic M (eds) Advances in High Performance Computing. NATO Advanced Sci Inst. Kluwer, Dordrecht, pp 273–302
24. Consiglio A, Zenios SA (June 1997) A model for designing callable bonds and its solution using tabu search. J Econom Dynam Control 21(8–9):1445–1470
25. Dahl H, Meeraus A, Zenios SA (1993) Some financial optimization models: II Financial engineering. In: Zenios SA (ed) Financial Optimization. Cambridge Univ. Press, Cambridge, pp 37–71
26. Del Angel GF, Márquez, Patiño EP (October 1998) A discrete Markov chain model for valuing loan portfolios–The case of Mexican loan sales. J Banking and Finance 22(10–11):1457–1480
27. Dempster MAH, Hutton JP (October 1996) Pricing American stock options by linear programming. Working Paper, Dept Math, Univ Essex
28. Dempster MAH, Hutton JP, Richards DG (September 1998) LP valuation of exotic American options exploiting structure. Working Paper WP 27/98, Judge Inst Management Studies, Univ Cambridge
29. Dixit AK, Pindyck RS (1994) Investment under uncertainty. Princeton Univ. Press, Princeton
30. Dryden MM (November 1968) Short-term forecasting of share prices: An information theory approach. Scottish J Political Economy 15:227–249
31. Dryden MM (March 1969) Share price movements: A Markovian approach. J Finance 24(1):49–60
32. Dutta PK, Madhavan A (March 1997) Competition and collusion in dealer markets. J Finance 52(1):245–276
33. Elimam AA, Girgis M, Kotob S (September-October 1996) The use of linear programming in disentangling the bankruptcies of Al-Manakh stock market crash. Oper Res 44(5):665–676
34. Elimam AA, Girgis M, Kotob S (January-February 1997) A solution to post crash debt entanglements in Kuwait's al-Manakh stock market. Interfaces 27(1):89–106
35. Elton EJ, Gruber MJ (May 1971) Dynamic programming applications in finance. J Finance 26(2):473–506
36. Fong HG, Vasicek O (September-October 1983) The tradeoff between return and risk in immunized portfolios. Financial Analysts J 39(5):73–78
37. Golub B, Holmer M, McKendall R, Lohlman P, Zenios SA (1995) A stochastic programming model for money management. Europ J Oper Res (85):282–296
38. Goonatilake S, Treleaven P (eds) (1995) Intelligent systems for finance and business. Wiley, New York
39. Grant D, Vora G, Weeks D (November 1997) Path dependent options: Extending the Monte Carlo simulation approach. Managem Sci 43(11):1589–1602
40. Hakansson NH, Ziemba WT (1995) Captial growth theory. In: Jarrow RA, Maksomivic V, Ziemba WT (eds) Finance. Handbooks in MS and OR, vol 9. North-Holland, Amsterdam, pp 65–86
41. Hausch DB, Lo V, Ziemba WT (eds) (1994) The efficiency of racetrack betting markets. Acad. Press, New York
42. Hausch DB, Ziemba WT (1985) Transaction costs, extent of inefficiencies, entries and multiple wagers in a racetrack betting model. Managem Sci 31(4):381–394 reprinted in: Hausch DB, Lo V, Ziemba WT (eds) The efficiency of racetrack betting markets. Acad. Press, New York, 1994
43. Hausch DB, Ziemba WT (1990) Arbitrage strategies for cross track betting on major horse races. J Business 63(1, Part 1):61–78, reprinted in: Hausch DB, Lo V, Ziemba WT (eds) The efficiency of racetrack betting markets. Acad. Press, New York, 1994
44. Hausch DB, Ziemba WT (1990) Locks at the racetrack. Interfaces 20(3):41–48, reprinted in Hausch DB, Lo V, Ziemba WT (eds) (1994) The efficiency of racetrack betting markets. Acad. Press, New York
45. Hausch DB, Ziemba WT, Rubinstein M (1981) Efficiency of the market for racetrack betting. Managem Sci 27(12):1435–1452, reprinted in: Hausch DB, Lo V, Ziemba WT (eds) The efficiency of racetrack betting markets. Acad. Press, New York, 1994
46. Heian BC, Gale JR (July-August 1988) Mortgage selection using a decision tree approach: An extension. Interfaces 18(4):72–81
47. Holmer MR, Yang D, Zenios SA (1998) Designing callable bonds using simulated annealing. In: Zopounidis C (ed) Operational Tools in the Management of Financial Risks. Kluwer, Dordrecht, pp 177–196
48. Hong HK (Winter 1981) Finance mix andcapital structure. J Business Finance and Accounting 8(4):485–491
49. Hutchinson JM, Lo AW, Poggio T (July 1994) A nonparametric approach to pricing and hedging derivative securities via learning networks. J Finance 49(3):851–889

50. Kallberg JG, White R, Ziemba WT (1982) Short run financial planning under uncertainty. Managem Sci 28(6):670–682
51. Keim DB, Ziemba WT (eds) (2000) Security market imperfections in world wide equity markets. Cambridge Univ. Press, Cambridge
52. Klaassen P (January 1998) Financial asset-pricing theory and stochastic programming models for asset/liability management: A synthesis. Managem Sci 44(1):31–48
53. Konno H, Yamazaki H (May 1991) Mean absolute deviation portfolio optimization model and its applications to Tokyo stock market. Managem Sci 37(5):519–531
54. Konno H, Yamazaki H (June 1997) An integrated stock-bond portfolio optimization model. J Econom Dynam Control 21(8–9):1427–1444
55. Kornbluth JSH, Salkin GR (1987) The management of corporate financial assets: Applications of mathematical programming models. Acad. Press, New York
56. Kraus A (December 1973) The bond refunding decision in an efficient market. J Financial and Quantitative Anal 8(5):793–806
57. Kumar PC, Philippatos GC (October 1979) Conflict resolution in investment decisions: Implementation of goal programming methodology for dual purpose funds. Decision Sci 10:562–576
58. Kumar PC, Philippatos GC, Ezzell JR (March 1978) Goal programming and the selection of portfolios by dual purpose funds. J Finance 33(1):303–310
59. Kusy MI, Ziemba WT (1986) A bank asset and liability management model. Oper Res 34(3):356–376
60. Lee SM, Eom HB (June 1989) A multi criteria approach to formulating international project financing strategies. J Oper Res Soc 40(6):519–528
61. Lee SM, Lerro AJ (December 1973) Optimizing the portfolio selection for mutual funds. J Finance 28(5):1087–1101
62. Litzenberger RH, Rutenberg DP (January 1972) Size and timing of corporate bond flotations. J Financial and Quantitative Anal 7(1):1343–1359
63. Luna RE, Reid RA (May-June 1986) Mortgage selection using a decision tree approach. Interfaces 16(3):73–81
64. MacLean LC, Ziemba WT (2000) Growth versus security tradeoffs in dynamic investment analysis. Ann Oper Res(March 2000):193–225
65. MacLean LC, Ziemba WT, Blazenko G (1992) Growth versus security in dynamic investment analysis. Managem Sci 38:1562–1585, reprinted in: Hausch DB, Lo V, Ziemba WT (eds) The efficiency of racetrack betting markets. Acad. Press, New York, 1994
66. MacLean LC, Ziemba WT, Li Y (2000) Time to wealth goals in capital accumulation and the optimal trade-off of growth versus security. UBC Mimeo
67. Markowitz H (March 1952) Portfolio selection. J Finance 7(1):77–91
68. Markowitz H (1987) Mean-variance in portfolio choice and capital markets. Blackwell, Oxford
69. Meade N, Salkin GR (October 1989) Index funds – Construction and performance measurement. J Oper Res Soc 40(10):871–879
70. Meade N, Salkin GR (July 1990) Developing and maintaining an equity index fund. J Oper Res Soc 41(7):599–607
71. Morgan JP (1997) Credit MetricsTM. Techn Report, Morgan Guarantee Trust Comp of New York
72. Morgan JP, Reuters (1996) Risk MetricsTM. Techn Report, Morgan Guarantee Trust Comp of New York
73. Mulvey JM (1987) Nonlinear network models in finance. In: Lawrence KD, Guerard JB, Reeves GR (eds) Advances in Mathematical Programming and Financial Planning, vol 1. JAI Press, Greenwich, CT, pp 253–271
74. Muley JM (May-June 1994) An asset liability investment system. Interfaces 24(3):22–33
75. Mulvey JM, Vladimirou H (November 1992) Stochastic network programming for financial planning problems. Managem Sci 38(11):1642–1664
76. Mulvey JM, Ziemba WT (1998) Asset and liability management systems for long term investors: Discussion of the issues. In: Ziemba WT, Mulvey JM (eds) Worldwide Asset and Liability Modelling. Cambridge Univ. Press, Cambridge, pp 3–38
77. Murtagh BA (1989) Optimal use of currency options. OMEGA 17(2):189–192
78. Nawalkha SK, Chambers DR (September-October 1996) An improved immunization strategy: M-absolute. Financial Analysts J 52(5):69–76
79. O'Hara M (1995) Market microstructure theory. Blackwell, Oxford
80. Paskov SH (1997) New methodologies for valuing derivatives. In: Dempster MAH, Pliska SR (eds) Mathematics of Derivative Securities. Cambridge Univ. Press, Cambridge, pp 545–582
81. Perold AF (October 1984) Large scale portfolio optimization. Managem Sci 30(10):1143–1160
82. Peterson PE, Leuthold RM (August 1987) A portfolio approach to optimal hedging for a commercial cattle feedlot. J Futures Markets 7(4):443–457
83. Powers IY (April 1987) A game theoretic model of corporate takeovers by major stockholders. Managem Sci 33(4):467–483
84. Premachandra I, Powell JG, Shi J (May 1998) Measuring the relative efficiency of fund management strategies in New Zealand using a spreadsheet-based stochastic data envelopment analysis model. OMEGA 26(2):319–331
85. Refenes AP (1995) Neural networks in the capital markets. Wiley, New York
86. Rudd A (Spring 1980) Optimal selection of passive portfolios. Financial Management 9(1):57–66
87. Rudd A, Schroeder M (December 1982) The calculation of minimum margin. Managem Sci 28(12):1368–1379
88. Seix C, Khoury AR (Spring 1986) Bond indexation: The optimal quantitative approach. J Portfolio Management 12(3):50–53

89. Shanker L (May 1993) Optimal hedging under indivisible choices. J Futures Markets 13(3):237–259
90. Sharda R (September-October1987) A simple model to estimate bounds on total market gains and losses for a particular stock. Interfaces 17(5):43–50
91. Sharpe WF (January 1963) A simplified model for portfolio analysis. Managem Sci 9(1):277–293
92. Sharpe WF (March 1967) A linear programming algorithm for mutual fund portfolio selection. Managem Sci 13(7):499–510
93. Sharpe WF (December 1971) A linear programming approximation for the general portfolio analysis problem. J Financial and Quantitative Anal 6(5):1263–1275
94. Shaw J, Thorpe EO, Ziemba WT (1995) Risk arbitrage in the Nikkei put warrant market at 1989–1990. Applied Math Finance 2:243–271
95. Taha HA (1991) Operations research analysis of a stock market problem. Comput Oper Res 18(7):597–602
96. Trippi RR, Turban E (eds) (1993) Neural networks in finance and investing: Using artificial intelligence to improve real world performance. Probus, Chicago
97. Vassiadou-Zeniou C, Zenios SA (1996) Robust optimization models for managing callable bond portfolios. Europ J Oper Res 91:264–273
98. Weingartner HM (March 1967) Optimal timing of bond refunding. Managem Sci 13(7):511–524
99. Wong K, Selvi Y (1998) Neural network applications in finance: A review and analysis of literature (1990–1996). Inform and Management 34(3):129–139
100. Worzel KJ, Vassiadou-Zeniou C, Zenios SA (March-April 1994) Integrated simulation and optimization models for tracking indices of fixed income securities. Managem Sci 42(2):223–233
101. Yawitz JB, Hempel GH, Marshall WJ (1976) A risk-return approach to the selection of optimal government bond portfolios. Financial Management 5(3, Autumn):36–45
102. Zenios SA (1991) Massively parallel computations for financial planning under uncertainty. In: Mesirov JP (ed) Very Large Scale Computation in the 21st Century. SIAM, Philadelphia, pp 273–294
103. Zenios SA (1993) A model for portfolio management with mortgage backed securities. Ann Oper Res 43(1–4):337–356
104. Zenios SA (1993) Parallel Monte Carlo simulation of mortgage backed securities. In: Zenios SA (ed) Financial Optimization. Cambridge Univ. Press, Cambridge, pp 325–343
105. Zenios SA, Holmer MR, McKendall R, Vassiadou-Zeniou C (September 1998) Dynamic models for fixed income portfolio management under uncertainty. J Econom Dynam Control 22(10):1517–1541
106. Zenios SA, Kang P (December 1993) Mean absolute deviation portfolio optimization for mortgage backed securities. Ann Oper Res 45(1–4):433–450
107. Zhao Y, Ziemba WT (forthcoming) A dynamic asset allocation model with downside risk control. J Risk
108. Zhao Y, Ziemba WT (forthcoming) A stochastic programming model using an endogenously determined worst case risk measure in a risk-return framework for dynamic asset allocation. Math Program
109. Ziemba WT (1972) Note on 'Optimal growth portfolios when returns are serially correlated'. J Financial and Quantitative Anal VII:1995–2000
110. Ziemba WT (1972) Solving non-linear programming problems with stochastic objective functions. J Financial and Quantitative Anal VII:1809–1827
111. Ziemba WT (1994) Investing in the turn of the year effect in the futures markets. Interfaces 24(3):46–61
112. Ziemba WT (1994) World wide security market regularities. Europ J Oper Res 74(2):198–229
113. Ziemba WT, Mulvey JM (1998) Worldwide asset and liability modelling. Cambridge Univ. Press, Cambridge

Operations Research Models for Supply Chain Management and Design

Joseph Geunes[1], ByungMan Chang[2]
[1] Department Industrial and Systems Engineering, University Florida, Gainesville, USA
[2] Department Industrial Engineering, Seoul National University Technol., Seoul, Korea

MSC2000: 90-02

Article Outline

Keywords
Strategic Design Models
  Distribution System Design
  Location-Routing Models
Production and Logistics Control Models
  Combined Inventory and Transportation Decisions
  Material Flows and Inventory Placement
  The Bullwhip Effect
Supply Chain Simulation Models
Summary
See also
References

Keywords

Supply chain management; Logistics management; Distribution systems

A *supply chain* is 'a connected series of activities which is concerned with planning, coordinating and controlling materials, parts, and finished goods from supplier to customer. It is concerned with two distinct flows (material and information) through the organization', [67]. *Supply chain management* (SCM) is the coordination and integration of these activities with the goal of achieving a sustainable competitive advantage. SCM therefore encompasses a wide range of strategic, financial, and operational issues.

With the emergence of supply chain management as a new discipline, the role of operations research (OR) models in effective SCM has become significant. From 1985 onwards the performance, design, and analysis of the supply chain has received increased attention. Optimizing the performance of an entire supply chain is one of the most comprehensive strategic problems facing managers today. See [63] for 23 different kinds of OR models for logistics activities in supply chain management and strong arguments that OR/MS techniques are essential for supporting the redesign of logistics processes. They suggest, however, that these various models must be coordinated effectively to properly model a multistage supply chain.

This entry considers past literature encompassing both *strategic* and *operational* issues in supply chain management and design. Our goal is to highlight many of the recent (and some not so recent, as of 2000) papers that have led to the current increased level of activity in supply chain research. Due to the volume of work done over from 1985 onwards we limit ourselves to a discussion of papers that use OR modeling techniques in both the overall design of the supply chain and the design of operations coordination and control systems within the supply chain. We do not consider work done in the growing areas of supply chain negotiation and contracts (except where such contracts may be driven by production and transportation system models) or in the value of information in the supply chain; for a detailed discussion on these and many other developments in the field of supply chain management, see [68], which provides a comprehensive view of the field of quantitative modeling applications in supply chain management. Other recent reviews of the literature in supply chain modeling include [72] and [9]. Our work complements these reviews by including the evolution of models for combined location-routing and inventory-routing decisions.

The *strategic design of a supply chain* requires managers to determine:

1) which suppliers and vendors to select for supplying raw materials;
2) the number, location, and capacity of manufacturing plants and warehouses;
3) specific transportation channels and modes for material movement between facilities;
4) raw material and end-item production amounts and control mechanisms for flows between suppliers, plants, warehouses, and customers; and
5) strategies for managing raw material, intermediate product, and finished goods inventory at each of the various locations.

Operational decisions in the supply chain, which are often constrained by choices made in the strategic design phase, include deciding short-term distribution and *logistics flows* between locations and production planning and inventory control policies at each location. Such decisions include determining the amount and timing of material flows from suppliers to plants, through plants and warehouses, and to retailers and customers. At the more detailed level, production and logistics operations planners must create detailed production sequencing and product delivery schedules.

We classify the literature we review into three broad categories:

- strategic supply chain design models;
- production and logistics coordination and control models; and
- supply chain simulation models.

The next three Sections consider models in each of these three areas. Note that within each section we have attempted to present past papers in chronological order of publication with a few exceptions to maintain continuity of presentation.

Strategic Design Models

This Section summarizes the evolution of OR models in the broad area of *supply chain design*. We first consider approaches that decide important long-term strategic factors such as location of facilities and assignment of products and/or customers to these facilities. Note that these models attempt to determine the supply chain de-

sign that provides the best combination of service and cost performance from an overall system view. That is, in these approaches, a central planner determines the best overall system configuration. We then take a look at models that combine location and transportation decisions in the subsection on *location-routing models*.

Distribution System Design

A.M. Geoffrion and G.W. Graves [36] present the first comprehensive supply chain design model for Hunt–Wesson foods. They develop a Benders decomposition-based algorithm that successfully solves a multicommodity production and distribution system design problem. This system contains several plants (with known and fixed capacities), distribution centers with limits on throughput, and retailer zones (each of which must be sourced to a single distribution center). The model finds the optimal distribution center configuration by considering fixed plus variable costs for operating warehouses as well as detailed production and transportation costs.

J.E. Hodder and M.C. Dincer [43] describe an international *plant location model* and develop a large scale nonlinear programming model. The objective function captures the difference between the expected value of profit and the variance of profit, which is scaled by a risk aversion factor. The model incorporates constraints on plant capacities, market demands, and financial investment limits. Their formulation considers the impacts of market prices, international interest rates, exchange rate fluctuations, production and transportation costs, import tariffs, and export taxes.

M.A. Cohen and H.L. Lee [22] present a single-period, multicommodity, nonlinear programming model to develop a global resource deployment policy. This paper provides a good initial effort to create a fully integrated logistics chain model by linking a set of distinct stochastic submodels. The objective function maximizes the total after-tax profit for the manufacturing facilities and distribution centers in all countries. The submodels incorporate plant production capacity limits, plant material requirements constraints, inventory balance constraints at both the plants and distribution centers, demand and supply capacity limits, and offset trade requirements. The model decides the assignment of products and subassemblies to plants, vendors to distribution centers, and distribution centers to market regions. It also determines the amounts of components, subassemblies, and final products to produce at each plant and how to ship these items between vendors, manufacturing facilities, and distribution centers.

J.H. Bookbinder and K.E. Reece [10] extend the Geoffrion–Graves [36] model to consider multicommodity *distribution system design* with *vehicle routing* and transportation fleet-sizing decisions. They combined the Geoffrion–Graves [36]*Benders decomposition* approach with the vehicle routing approach of M.L. Fisher and R. Jaikumar [33]. The master problem in their algorithm determines the number and locations of warehouses. The subproblems then determine the best set of vehicle routes (including the number and sizes of vehicles used by location), given the warehouse configuration specified by the master problem.

Cohen and Lee [23] built on their prior work by creating a deterministic model for designing a large scale distribution network. This model also incorporates offset trade requirements and estimates before and after tax profits. Although they provide a comprehensive formulation of the global supply chain design problem, no detailed solution procedure for determining the optimal configuration is provided.

Cohen and S. Moon [24] formulate a mixed integer, multicommodity model that determines the assignment of product lines and production volumes to plants. For each plant they determine inbound raw material flows and outbound finished product flows. This research provides a restricted optimization algorithm to solve production-distribution problems with piecewise linear concave production costs. The objective function consists of fixed and variable production and transportation costs. The model includes supply, capacity, assignment, demand, and raw material requirements constraints. A variant of the Benders decomposition technique is applied to solve each problem instance.

Cohen and P.R. Kleindorfer [21] present a normative model for global manufacturing operations that directs plant location and capacity decisions as well as product mix, material flow, and cash flow amounts. The model consists of several submodels (a stochastic supply chain network model, a financial flow model, a stochastic exchange rate model, and a price-demand model). The submodels link a multiperiod stochastic

master problem to a set of single-period stochastic subproblems. See [23] for a description of an implementation of this model framework.

B.C. Arntzen et al. [3] present one of the most comprehensive supply chain design models to date, which they use to redesign Digital Equipment Corporation's (DEC's) supply chain. They develop a multiperiod, multicommodity, mixed integer programming model called GSCM (Global Supply Chain Model) to optimize the configuration of DEC's global supply chain. The model accommodates multiple facilities, stages (echelons), time periods, and transportation modes. GSCM minimizes a composite function of activity days and the total cost of production, inventory, material handling, overhead, and transportation. The constraints enforce requirements on meeting customer demands, production and throughput capacity limits at each facility, and bounds on decision variables. GSCM encodes the global bill-of-materials (BOM) for each product and enforces inventory balance constraints for every product and location. Their formulation reflects offset trade and local content restrictions as well as duty payment and drawback for flows through various countries. The model decides the number and location of distribution centers, the customer-to-distribution center assignments, and product-to-plant assignments. Arntzen et al. [3] describe how DEC used GSCM to evaluate global supply chain alternatives and develop worldwide manufacturing and distribution strategies. DEC used GSCM to guide a worldwide restructuring that saved the company over $100 million.

J.D. Camm et al. [15] develop an integer programming model using an uncapacitated *facility location* formulation for Procter & Gamble's distribution network. The model minimizes the total cost of distribution center location selection and distribution center-to-customer assignments, subject to assignment constraints and a maximum number of distribution centers in operation. This model chooses the best location and scale of operation for producing items in Procter & Gamble's product line.

H. Pirkul and V. Jayaraman [58] propose a mixed integer programming model for designing a three-level distribution network. The model defines the physical flows of commodities from plants to warehouses and from warehouses to customer zones. Their method determines the locations of plants and warehouses that result in the minimum total operating plus fixed costs for the distribution network. They [58] develop a Lagrangian relaxation-based heuristic that assigns customers to open warehouses based on their demand for products, and then assigns open warehouses to open plants.

Recent approaches (as of 2000) for dynamic distribution system design, in which customer demands vary over a finite planning horizon, have been developed in [60,61] and [34]. These papers consider a set of capacitated plants with associated (uncapacitated) warehouses that must distribute products to a set of customers in each period. Each of these models requires assigning every customer to a single source, or warehouse, and hence employs heuristics for generalized assignment formulations of the problem. H.E. Romeijn and D. Romero Morales [60,61] show the asymptotic optimality of specific greedy heuristic procedures for the cyclic (repeating demand patterns) and acyclic demand cases. R. Freling et al. [34] apply a branch and price algorithm to the single sourcing problem under seasonal demands.

L.M.A. Chan and D. Simchi-Levi [17] consider a transportation-inventory-routing design problem and provide a model and algorithm for a three-level distribution system. The system contains a single vendor, multiple cross-docking warehouses, and multiple retailers with constant demand rates. We include this paper in the design section because, unlike the inventory-routing problems covered in the following section, this work develops insights regarding distribution system design strategy. The authors argue that this system corresponds closely to the Wal-Mart distribution system, which has proven immensely successful over the past decade. Their algorithm assigns each retailer to a warehouse using a bin-packing scheme, and partitions the assigned retailers into clusters using a capacitated concentrator location algorithm. They then combine retail clusters into groups that share the same reorder interval. The model minimizes asymptotic long-run average transportation plus inventory costs, and the authors show an optimal solution that forces each warehouse to receive fully loaded trucks from the vendor, but never to hold inventory. The warehouse therefore serves only as a coordinator of the frequency, time, and size of deliveries to retailers, i. e., as a cross-docking facility.

Location-Routing Models

Next we briefly summarize work done on so-called location-routing models. This research area has received vast attention starting in the 1970s and we therefore highlight a few of the more significant developments. For a detailed summary of location-routing models, see [8] and [53]. Since traditional facility location models consider only the cost of assigning customers or markets to facilities (see, for example, [25]), many researchers have shown the value of explicitly considering integrated location models that incorporate detailed vehicle routing decisions.

Perhaps the earliest work to take such an integrated view was done by I.R. Webb [73], who randomly generates depot locations and considers the impact of using straight-line distances as a substitute for route distances in a deterministic setting with vehicle capacity restrictions. The paper concludes that a significant loss of accuracy can result from using straight-line models. S.K. Jacobsen and O.B.G. Madsen [45] subsequently use sophisticated heuristic approaches to solve a two-level newspaper distribution system design problem in practice.

G. Laporte and Y. Nobert [47] and Laporte et al. [48] give exact integer programming approaches for the location-routing problem in the absence of vehicle capacity restrictions (the first paper considers a single depot, while the second extends this to multiple depots). To solve a set of randomly generated problems they employ a branch and bound algorithm with the addition of violated subtour constraints.

J. Perl and M.S. Daskin [55,56] use optimization-based heuristic approaches for solving the multiple-depot routing problem with depot throughput costs and capacities. Their heuristic initially opens all depots and solves a routing problem for each. They then use fixed cost and depot capacity considerations to determine which depots to keep open and heuristically reallocate customers to open facilities.

Laporte et al. [46] extend the analysis to consider the location of a depot that collects goods from customers and returns them to the depot. In this model customer supplies are random variables and a vehicle must return to the depot once it is filled to capacity. The problem is modeled as a two-stage stochastic program with recourse.

R. Srivastava and W.C. Benton [65] consider the impact of environmental and operational factors on solutions to location-routing problems. These factors include the customers' spatial distribution and the ratio of location cost to routing costs. The authors apply several combinations of well-known routing and location heuristics to determine the effects of specific factors on heuristic performance.

Other recent heuristics and case studies in the location-routing literature include [4,41,54,64], and [52]. The following section considers shorter-term operations decisions in supply chain management.

Production and Logistics Control Models

This Section considers models that decide more detailed operational decisions such as the timing and quantities of flows between facilities. These issues and decisions have been addressed for many years within the context of single-stage models. We focus on models that consider the impacts such decisions have on multiple entities in a supply chain. These entities may or may not belong to the same firm and may have either the same or conflicting objectives. With a few exceptions we have not summarized the voluminous literature in the area of multi-echelon inventory control. Papers in this area tend to develop cost models of multistage *inventory systems* under various cost, demand, and operational assumptions. They then seek to minimize overall production and inventory related costs. With the exception of the paper [20] noted in the following paragraph, we focus on models that consider inventory costs in conjunction with distribution and logistics related costs. For a detailed discussion of multi-echelon inventory analysis, see [39,49,68].

This Section first focuses on models that combine inventory and transportation decisions. (For in-depth analyses of general vehicle routing and inventory-routing models, see [30] and [11].) We then look at papers that consider the coordination of material flows and inventory placement in the supply chain. Finally we look at the phenomenon known as the 'bullwhip effect', (which describes the commonly observed increased demand variance at upstream stages in a supply chain) and models that have attempted to mitigate this effect.

Combined Inventory and Transportation Decisions

A.J. Clark and H. Scarf [20] provide perhaps the earliest work considering the operational interaction between two separate members of a distribution chain. They develop a periodic review inventory management model for a serial distribution system. They use a simple system consisting of a single distribution center and a single retailer to show the optimality of modified base-stock policies for the retailer's inventory control. Modified base-stock policies require a fixed base-stock level that the retailer attempts to bring inventory up to at the beginning of each period. If the retailer cannot reach its target base-stock level (because the distribution center has insufficient stock), the retailer attempts to get as close to the base-stock level as possible by exhausting distribution center stock.

In one of the initial efforts to provide an exact model that incorporates combined inventory and transportation costs with stochastic demands, A. Federgruen and P. Zipkin [31] consider a single-period problem in which a central depot must allocate its inventory among multiple retailers. Their model minimizes expected inventory holding and shortage costs plus vehicle routing costs. Each of a set of vehicles has a fixed capacity and incurs a cost equal to c_{ij} for travel between locations i and j. The model determines vehicle routes and the allocation of inventory among the retailer locations that minimizes combined inventory and routing costs.

L.B. Burns et al. [12] present the model behind a system developed in a streamlining effort for GM's Delco Electronics Division distribution network. The model examines the trade-offs between inventory and transportation costs and determines which plants should supply a variety of assembly facilities in North America. The key decision involved was between the current practice of straight-line deliveries to a single centralized warehouse versus a peddling strategy that delivers parts directly to the assembly plants. The model recommended a new peddling strategy for Delco components and resulted in a 26.9% logistics savings opportunity.

C.A. Yano and Y. Gerchak [76] consider a two-stage system where a manufacturing plant supplies an assembly plant with Just-in-time (JIT) shipments of a high-volume part. The model assumes that the retailer observes stochastic, periodic demand of a single product. They allow for emergency shipments of parts when insufficient vehicle capacity exists. They determine the order-up-to point (base-stock level) for the part, the time between successive deliveries of the part, and the number of vehicles contracted for deliveries between the supplier and assembly plant. The model minimizes the sum of assembly plant expected inventory costs, contracted shipment costs, and emergency shipment costs. R. Ernst and Pyke [29] build on the work of Yano and Gerchak [76] by including the manufacturer's (or, in their case, the warehouse's) expected inventory costs plus per unit shipping costs (as opposed to a per truck shipping cost only).

We find a more detailed treatment of the trade-off between vehicle routing and inventory costs in [2], which considers a system with a central depot and multiple geographically dispersed retailers. All stock enters the system through the depot and each retailer observes deterministic demand that occurs at a constant rate. The model minimizes inventory costs at the retailers (the depot holds no stock) plus distribution costs incurred by combining retailer deliveries into routes served by a set of vehicles with fixed capacities. The authors develop bounds on both the optimal solution and solutions from a class of heuristics they develop. They then show the asymptotic optimality of their heuristics (within a class of strategies that partitions retailers into regions, and if a vehicle visits a region it must visit all retailers in the region). For other algorithms for and extensions to the inventory-routing model see [19,27,28], and [74].

P. Chandra and Fisher [18] consider the problem of coordinating production runs with vehicle routing decisions for a single facility with multiple customers. They compared the case of separately optimizing production planning and vehicle routing decisions to a coordinated approach that attempts to minimize the total combined cost of production and routing. They found increased value in coordinating these decisions under higher production capacity, longer time horizons, and low holding and setup costs (i. e., the more flexible and less constrained the production planning situation).

M. Henig et al. [42] consider the problem of determining the optimal inventory replenishment policy structure and truck capacity when a distribution center imposes a variable cost for each unit ordered in excess of distribution center truck capacity, R. They show that the optimal periodic ordering policy under such a cost

contains two base-stock levels, S_1 and S_2 ($S_2 \geq S_1$). The policy structure requires that if initial inventory position before ordering is less than S_2, we either order up to S_1, order exactly R units, or order up to S_2, depending on the value of the initial inventory position; otherwise we do not order. The paper also specifies a method to determine the best level of truck capacity under the given cost structure.

Due to the increased reliance by manufacturers on third party less-than-truckload (LTL) carriers, Chan et al. [16] consider a multiperiod distribution model with LTL shipments. Their model assumes that a set of customers observe periodic, deterministic demand for multiple products over a finite planning horizon. The distribution network contains both suppliers and warehouses and the model allows for backlogging customer demands. The authors pose the problem as a fixed-charge minimum concave cost network flow problem over an equivalent distribution network, where the costs include LTL shipping costs plus inventory holding and shortage costs. The paper develops properties of optimal solutions and shows conditions under which the linear programming (LP) relaxation (under piecewise-linear concave costs) value equals the optimal mixed integer solution value. The heuristic solution algorithm proposed is based on effectively characterizing the cost of modifying a fractional integer variable found in the LP relaxation solution. Their results compare favorably to an original solution approach for the problem proposed in [7].

Material Flows and Inventory Placement

We next consider several papers that deal with more effectively managing material flows and inventory placement in the supply chain.

Lee and C. Billington [50] develop a stochastic model for managing material flows in Hewlett-Packard's deskjet printer supply chain. They assume that each site observes stochastic demand and employs a periodic, order-up-to inventory system and that a predetermined value is set for either a target service level or base-stock level (for each site). Their model characterizes the demand transmission process (whereby a site translates its demand into orders on its' suppliers) and the availability transfer process, which describes the availability of goods at the supplier location. They determine the review period and order quantity for each product type and location and address the trade-offs between inventory investment and service level in a multistage supply chain.

Pyke and Cohen [59] develop a stochastic model for managing material flows in an integrated three-level production-distribution system. The system contains multiple products, a single manufacturing facility, one warehouse, and a single retailer. The retailer promises its customers a minimum service level and the model minimizes total cost under constant setup and processing times at the factory. Although the transportation time from factory to retailer is constant, an option exists for the retailer to receive an expedited shipment if the finished goods stockpile at the factory cannot satisfy retailer demand. The model assumes stochastic production times and lead times between the factory and its finished goods stockpile, and approximates key inventory and service time distributions necessary for expressing expected total system cost. The outputs of the model include the economic reorder interval and replenishment batch size for each product type.

T. Altiok and R. Ranjan [1] analyze a multistage pull-type production system containing one final product, FIFO (first in, first out) processing rules, intermediate buffers, and Poisson demand. Each stage follows an (R, r) inventory policy (when inventory position falls below r, order up to R) and backorders when stock is insufficient to meet demand. They develop an iterative procedure that separately considers the flow in each two-node subsystem as a function of the policy parameters. The procedure terminates when the average throughput values of all subsystems are approximately equal. The outputs include the inventory level in each buffer and the backorder probability, P. The authors show that when P does not exceed 0.3, their approximation method provides acceptable results.

S.C. Graves and S. Willems [40] consider the placement (location) of safety stock in a multistage supply chain. They represent the supply chain as a network and assume that each stage follows a base-stock policy (a stage represents a processing function, such as procurement of raw material, component production, assembly, testing, etc.). Demand occurs only at nodes that have no successors and each stage provides a guaranteed service time for meeting downstream demand. The

model minimizes system-wide safety stock costs while meeting the guaranteed service times (which constitute the decision variables). They [40] validated their model through a successful implementation in Eastman Kodak's digital camera supply chain.

Graves et al. [38] study *requirements planning* in multistage production systems. They begin with a single-stage model that produces one (aggregated) product to stock for a finished goods stockpile. They assume that each stage forecasts demand H periods in advance and revises forecasts in each period. These forecasts drive the production plan, which is also planned H periods in advance and is revised each period. They measure three significant parameters: the smoothness of the production plan, the stability of the production plan, and the safety stock level (smoothness of the production plan differs from stability in that smoothness characterizes the variability of actual production output while stability characterizes the variability of the production schedule). The paper captures the tradeoff between production capacity and inventory requirements. The authors show how to extend the single-stage model to a multiple-stage dynamic requirements planning (DRP) model by replicating single-stage models. An application of the model to film manufacturing at Kodak resulted in a 60% decrease in inventory requirements for two items, while increasing one item's inventory by 20%, with a significant net savings reported for the business line.

The Bullwhip Effect

Several recent papers (as of 2000) have described and quantified the phenomenon known as the *bullwhip effect* in supply chains – the tendency for demand variability to increase at upstream stages in the chain. This demand variability places a burden on suppliers because of the increased safety stock and excess vehicle capacity requirements.

C.C. Holt, F. Modigliani, and J.P. Shelton [44] first showed evidence that the variation in orders can increase at upstream points in the television manufacturing chain. J.D. Sterman's [66] documentation of experience with supply chain simulation experiments shows how rational decision making can lead to the bullwhip effect in the absence of full information about partners in the chain.

Lee, V. Padmanabhan, and S. Whang [51] offer four factors to explain the existence of the bullwhip effect (demand forecast updating, order batching, price fluctuation, and shortage gaming), along with mathematical models of these phenomena. They provide insights on how coordination between channel partners can diminish the bullwhip effect. Z. Drezner et al. [26] specifically consider the impact of forecast errors on the bullwhip effect. Their model shows that even in supply chains with perfect information shared among all members, errors in demand forecasts can still create a bullwhip effect.

M.P. Baganha and Cohen [5] consider conditions under which demand stabilization (or damping) becomes economically attractive in a two-echelon distribution system containing a warehouse and multiple retailers. Their model assumes that each retailer incurs a fixed order cost, which results in the optimality of (s, S) inventory policies at each retailer (when inventory position falls below s, order up to S). Following an (s, S) policy results in autocorrelated orders from each retailer since we can expect that a retailer will not place an order in a period immediately following one in which an order was placed. Baganha and Cohen [5] therefore apply an autoregressive model to describe the distribution of retailer orders (which constitutes warehouse demand) and show situations in which retailer order damping can lead to lower system cost.

G.P. Cachon [13] studies supply chain demand variability in a model with one supplier and N retailers that face stochastic demand. Retailers follow a periodic (R, nQ) policy (order up to R using integer multiples of some base quantity, Q). The model develops exact expressions for supply chain (inventory) costs under stationary retailer demand distributions. The results show how the supplier's demand variance declines as the retailers' order intervals are lengthened or as batch size increases. The main result of the paper shows that a balanced ordering strategy (when the same number of retailers order in each period) can lead to significant savings in supply chain inventory costs.

Cachon and Zipkin [14] consider competition and cooperation in a two-stage serial supply chain. The model assumes stationary stochastic demand occurs at a single retailer who then orders from a supplier. They consider both the competitive and cooperative scenarios: in the competitive case each firm minimizes its own

cost, while a coordinated approach minimizes system costs (both firms follow a base-stock policy). The paper shows how the competitive approach reduces efficiency by leading to a Nash Equilibrium point that differs from the system optimal solution. A sequence of linear transfer payments allows the firms to achieve the system optimal solution while operating independently.

A. Balakrishnan et al. [6] consider a two-echelon supply chain with a single-distribution center, multiple retailers, and multiple products. Like Cachon and Zipkin [14], they compare expected chain costs under autonomous decisions (competitive scenario) to those under coordination mechanisms. The model differs from that of Cachon and Zipkin [14] since it considers inventory plus transportation costs, and the coordination mechanisms differ from linear transfer payments. Balakrishnan et al. [6] offer both cost- and policy-driven mechanisms that effectively use the retailer order variability as a decision variable. By tuning to the best level of order variability they show that the coordination mechanisms can lead to better system costs than under autonomous decisions by damping retailer demand variation.

Supply Chain Simulation Models

Because of the large number of decision variables and the complexity of the constraints required in developing exact cost models of large scale distribution systems, many researchers have used *simulation models* to provide valuable insight into complex supply chain dynamics. We next briefly consider models that have used simulation to derive such insights.

J. Wikner et al. [75] examine five supply chain improvement strategies using a simulation model for a three-level supply chain that includes one factory, multiple distribution centers and retailers, and carries inventory at each facility. The five strategies include

1) reducing system delays;
2) fine tuning order policy parameters;
3) removing the distribution center echelon from the system;
4) changing different echelon decision rules; and
5) improving information integration between stages.

They conclude that integrating information flow between channel partners is the most effective strategy for minimizing supply chain operating costs.

D.R. Towill et al. [70] then extend the simulation model to consider a just-in-time (JIT) delivery strategy. The JIT strategy combined with the removal of the distribution center echelon was shown to be more effective than the integration of information flow or modification of the order policy parameters. Towill and A. Del Vecchio [69] use methods from filter theory combined with a simulation model to analyze the effects of demand variability in the supply chain. They liken each stage in the chain to an electrical filter (with a response function) and analyze various supply chain responses to randomness in demand patterns. The simulated responses determine the minimum safety stock required to achieve a desired service level.

S. Tzafestas and G. Kapsiotis [71] present a mathematical programming based approach to optimize a portion of the supply chain and use simulation techniques to numerically analyze the performance of the optimized model. They consider a two-echelon system in which a manufacturer supplies multiple assembly plants. They then consider three decision scenarios. In scenario I the manufacturer minimizes its cost and its customers must accept the deliveries imposed by the manufacturer. Scenario II incorporates a central decision-maker that attempts to minimize overall system costs. Scenario III presents a decentralized decision framework where the supplier minimizes its cost subject to the demand imposed by the assembly plants under their optimal decisions. They perform the simulation under three scenarios: manufacturing facility optimization, global supply chain optimization, and decentralized optimization at each level. The numerical examples chosen by the authors do not result in significant differences in cost performance under any of the three tested scenarios.

D. Petrovic et al. [57] use fuzzy modeling to determine order-up-to levels at various stages in a supply chain. They develop a supply chain simulator that analyzes the effects of order-up-to levels on cost and the dynamic behavior of the chain in an uncertain environment. The fuzzy model handles uncertainty in both customer demand and external supply of raw materials. The model attempts to determine the stock levels and order quantities at each stage in the chain that give acceptable delivery performance at a reasonable total cost.

R. Ganeshan [35] presents a near optimal (s, Q) inventory policy (when inventory position falls below

s, order *Q* units) for a production/distribution network with multiple suppliers, a central warehouse, and multiple retailers. This model develops a system-wide cost equation (by describing the demand process at the warehouse) that includes warehouse and retailer inventory costs and uses a conjugate gradient method to find the best policy parameters. The paper verifies the costs implied by the prescribed policy parameters using a SLAM-based supply chain simulation model.

Summary

Fisher [32] notes that the performance of supply chains today is extremely poor despite advances in the areas of quick response systems, mass customization, lean manufacturing, and new technologies. As firms increase their operations throughout the world, they will have even greater need for tools that integrate operations and information among geographically dispersed locations.

The key to success in SCM requires an emphasis on integrating activities through cooperation, coordination, and information sharing throughout the entire chain. To have the greatest benefit, the supply chain must be managed as a single entity, which requires improved OR models and tools so that supply chain managers can solve problems that reflect the relationships among all supply chain activities.

Many research opportunities exist for developing global supply chain models that take into account elements necessary for providing a complete and integrated view of the system. New areas for research include the following:

- modeling the effects of a greater number of stochastic elements;
- accounting for international economic issues (including exchange rate fluctuations and risks);
- incorporating and modeling detailed BOM relationships;
- product differentiation and mass customization strategies;
- capitalizing on advances in information technologies; and
- the value of important strategic global alliances.

We have commented on many significant developments and applications of OR models. Despite these advances, Geoffrion and Powers [37] report that many of the most popular commercial software packages still use simple heuristic approaches that result in significantly suboptimal cost performance. Because of the proliferation of desktop computers and software packages with sophisticated graphical user interfaces, logistics executives have almost exclusively selected overly simplified heuristic-based software for the design and analysis of the supply chain.

Shapiro et al. [62] note that now is the perfect time for developing sophisticated OR supply chain models for personal computers and describe their success in doing so for a large consumer products company. The OR/MS profession has recently created many powerful decision tools that provide opportunities for improved SCM. It is important that these tools continue to find widespread application in industry through the development of comprehensive and user-friendly SCM systems.

See also

- ► Global Supply Chain Models
- ► Inventory Management in Supply Chains
- ► Nonconvex Network Flow Problems
- ► Piecewise Linear Network Flow Problems

References

1. Altiok T, Ranjan R (1995) Multi-stage, pull-type production/inventory systems. IIE Trans 27(190–200
2. Anily S, Federgruen A (1990) One warehouse multiple retailer systems with vehicle routing costs. Managem Sci 36(1):92–114
3. Arntzen BC, Brown GG, Harrison TP, Trafton LL (1995) Global supply chain management at Digital Equipment Corporation. Interfaces 25(1):69–93
4. Aykin T (1995) The hub location and routing problem. Europ J Oper Res 83:200–219
5. Baganha MP, Cohen MA (1998) The stabilizing effect of inventory in supply chains. Oper Res 46(3):s72–s83
6. Balakrishnan A, Geunes J, Pangburn M (1999) Impacts of variability control mechanisms on distribution channel performance. Working Paper, MS and IS Dept, Penn State Univ
7. Balakrishnan A, Graves S (1989) A composite algorithm for a concave-cost network flow problem. Networks 19:175–202
8. Balakrishnan A, Ward JE, Wong RT (1987) Integrated facility location and vehicle routing models: Recent work and future prospects. American J Math Management Sci 7:35–61

9. Beamon BM (1998) Supply chain design and analysis: Models and methods. Internat J Production Economics 55:281–294
10. Bookbinder JH, Reece KE (1988) Vehicle routing considerations in distribution system design. Europ J Oper Res 37(2):204–213
11. Bramel J, Simchi-Levi D (1997) The logic of logistics: Theory, algorithms and application for logistics management. Springer, Berlin
12. Burns LB, Hall RW, Blumenfeld DE, Daganzo CF (1987) Distribution strategies that minimize transportation and inventory costs. Oper Res 33(3):469–490
13. Cachon GP (1999) Managing supply chain demand variability with scheduled ordering policies. Managem Sci 45(6):843–856
14. Cachon GP, Zipkin P (1999) Competitive and cooperative inventory policies in a two-stage supply chain. Managem Sci 45(7):936–953
15. Camm JD, Chorman TE, Dull FA, Evans JR, Sweeney DJ, Wegryn GW (1997) Blending OR/MS judgment and GIS: Restructuring P and G's supply chain. Interfaces 27(1):128–142
16. Chan LMA, Muriel A, Simchi-Levi D (1999) Supply-chain management: Integrating inventory and transportation. Working Paper, Dept of IE and MS, Northwestern Univ
17. Chan LMA, Simchi-Levi D (1998) Probabilistic analyses and algorithms for three-level distribution systems. Managem Sci 44(11):1562–1576
18. Chandra P, Fisher ML (1994) Coordination of production and distribution planning. Europ J Oper Res 72:503–517
19. Chien TW, Balakrishnan A, Wong RT (1989) An integrated inventory allocation and vehicle routing problem. Transport Sci 32(2):67–76
20. Clark AJ, Scarf H (1960) Optimal policies for a multi-echelon inventory problem. Managem Sci 6:475–490
21. Cohen MA, Kleindorfer PR (1993) Creating value through operations: The legacy of Elwood S. Buffa. Kluwer, Dordrecht, pp 3–21
22. Cohen MA, Lee HL (1988) Strategic analysis of integrated production-distribution systems: Models and methods. Oper Res 36(2):216–228
23. Cohen MA, Lee HL (1989) Resource deployment analysis of global manufacturing and distribution networks. J Manufacturing Oper Management 2:81–104
24. Cohen MA, Moon S (1991) An integrated plant loading model with economies of scale and scope. Europ J Oper Res 50:226–279
25. Cornuejols G, Nemhauser GL, Wolsey LA (1990) The uncapacitated facility location problem. In: Mirchandani PB, Francis RL (eds) Discrete Location Theory. Wiley, New York
26. Drezner Z, Ryan JK, Simchi-Levi D (1998) Quantifying the bullwhip effect in a simple supply chain: the impact of forecasting, lead times, and information. Working Paper, Dept of IE and MS, Northwestern Univ
27. Dror M, Levy L (1986) A vehicle routing improvement algorithm comparison of a "greedy" and a matching implementation for inventory routing. Comput Oper Res 13(1):33–45
28. Dror M, Trudeau P (1986) Stochastic vehicle routing with modified savings algorithm. Europ J Oper Res 23:228–235
29. Ernst R, Pyke D (1993) Optimal base stock policies and truck capacity in a two echelon system. Naval Res Logist 40:879–903
30. Federgruen A, Simchi-Levi D (1995) Analysis of vehicle routing and inventory-routing problems. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Handbook Oper. Res. and Management Sci.: Network Routing, vol 8, 297–373
31. Federgruen A, Zipkin P (1984) A combined vehicle routing and inventory allocation problem. Oper Res 32(5):1019–1037
32. Fisher ML (1997) What is the right supply chain for your product? Harvard Business Rev 75:105–116
33. Fisher ML, Jaikumar R (1981) A generalized assignment heuristic for vehicle routing. Networks 11:109–124
34. Freling R, Romeijn HE, Romero Morales D, Wagelmans APM (1999) A branch and price algorithm for the multi-period single-sourcing problem. Working Paper 99–12, Dept of ISE, Univ Florida
35. Ganeshan R (1999) Managing supply chain inventories: A multiple retailer, one warehouse, multiple supplier model. Internat J Production Economics 59:341–354
36. Geoffrion AM, Graves GW (1974) Multicommodity distribution system design by Benders decomposition. Managem Sci 20(5):822–844
37. Geoffrion AM, Powers RF (1995) Twenty years of strategic distribution system design: An evolutionary perspective. Interfaces 25:105–128
38. Graves SC, Kletter DB, Hetzel WB (1998) A dynamic model for requirements planning with application to supply chain optimization. Oper Res 46(Sun3):S35–S49
39. Graves SC, Rinnooy Kan AHG, Zipkin PH (eds) (1993) Handbook Operations Research and Management Science: Logistics of Production and Inventory, vol 4. Elsevier, Amsterdam
40. Graves SC, Willems S (1998) Optimizing strategic safety stock placement in supply chains. Working Paper, Sloan School Management, MIT
41. Hansen PH, Hegedahl B, Hjortkjaer S, Obel B (1994) A heuristic solution to the warehouse location-routing problem. Europ J Oper Res 76:111–127
42. Henig M, Gerchak Y, Ernst R, Pyke DF (1997) An inventory model embedded in designing a supply contract. Managem Sci 43(2):184–189
43. Hodder JE, Dincer MC (1986) A multifactor model for international plant location and financing under uncertainty. Comput Oper Res 13(5):601–609

44. Holt CC, Modigliani F, Shelton JP (1968) The transmission of demand fluctuations through distribution and production systems: the TV set industry. Canad J Econom 14:718–739
45. Jacobsen SK, Madsen OBG (1980) A comparative study of heuristics for a two-level location-routing problem. Europ J Oper Res 5:378–387
46. Laporte G, Louveaux F, Mercure H (1989) Models and exact solutions for a class of stochastic location-routing problems. Europ J Oper Res 39:71–78
47. Laporte G, Nobert Y (1981) An exact algorithm for minimizing routing and operating costs in depot location. Europ J Oper Res 6:224–226
48. Laporte G, Nobert Y, Pelletier P (1983) Hamiltonian location problems. Europ J Oper Res 12:80–87
49. Schwarz LB (1981) Multi-level production/inventory control systems: Theory and practice. North-Holland, Amsterdam
50. Lee HL, Billington C (1993) Material management in decentralized supply chains. Oper Res 41(5):835–847
51. Lee HL, Padmanabhan V, Whang S (1997) Information distortion in a supply chain: The bullwhip effect. Managem Sci 43(4):546–558
52. Min H (1993) Consolidation terminal location-allocation and consolidated routing problems. J Business Logistics 17(2):235–263
53. Min H, Jayaraman V, Srivastava R (1998) Combined location-routing problems: A synthesis and future research directions. Europ J Oper Res 108:1–15
54. Nambiar JM, Gelders LF, Van Wassenhove LN (1989) Plant location and vehicle routing in the Malaysian rubber smallholder sector: A case study. Europ J Oper Res 38:14–26
55. Perl J, Daskin MS (1984) A warehouse location-routing methodology. J Business Logistics 5:92–111
56. Perl J, Daskin MS (1985) A warehouse location-routing problem. Transport Res B 19B:381–396
57. Petrovic D, Roy R, Petrovic R (1998) Modeling and simulation of a supply chain in an uncertain environment. Europ J Oper Res 109:299–309
58. Pirkul H, Jayaraman V (1998) A multi-commodity, multiplant, capacitated facility location problem: Formulation and efficient heuristic solution. Comput Oper Res 25(10):869–878
59. Pyke DF, Cohen MA (1994) Multi-product integrated production-distribution systems. Europ J Oper Res 74(1): 18–49
60. Romeijn HE, Romero Morales D (1999) Asymptotic analysis of a greedy heuristic for the multi-period single-sourcing problem: the acyclic case. Working Paper 99–13, Dept of ISE, Univ Florida
61. Romeijn HE, Romero Morales D (1999) An asymptotically optimal greedy heuristic for the multi-period single-sourcing problem: the cyclic case. Working Paper 99–11, Dept of ISE, Univ Florida
62. Shapiro JF, Singhal VM, Wagner SN (1993) Optimizing the value chain. Interfaces 23(2):102–117
63. Slats PA, Bhola B, Evers JM, Dijkhuizen G (1995) Logistic chain modeling. Europ J Oper Res 87:1–20
64. Srivastava R (1993) Alternate solution procedures for the location routing problem. OMEGA Internat J Management Sci 21(4):497–506
65. Srivastava R, Benton WC (1990) The location-routing problem: considerations in physical distribution system design. Comput Oper Res 17(5):427–435
66. Sterman JD (1989) Modeling managerial behavior: misperceptions of feedback in a dynamic decision making experiment. Managem Sci 35(3):321–339
67. Stevens GC (1989) Integrating the supply chain. Internat J Physical Distribution and Materials Management 19:3–8
68. Tayur S, Ganeshan R, Magazine, eds. M (1998) Quantitative models for supply chain management. Kluwer, Dordrecht
69. Towill DR, Del Vecchio A (1994) The application of filter theory to the study of supply chain dynamics. Production Planning and Control 5(1):82–96
70. Towill DR, Naim MM, Wikner J (1992) Industrial dynamics simulation models in the design of supply chain. Internat J Physical Distribution and Logistics Management 22(5):3–13
71. Tzafestas S, Kapsiotis G (1994) Coordinated control of manufacturing/supply chains using multi-level techniques. Comput Integrated Manufacturing Systems 7(3):206–212
72. Vidal CJ, Goetschalckx M (1997) Strategic production-distribution models: A critical review with emphasis on global supply chain models. Europ J Oper Res 98:1–18
73. Webb IR (1968) Cost functions in the location of depots for multi-delivery journeys. Oper Res Quart 19:311–328
74. Webb IR, Larson RC (1995) Period and phase of customer replenishment: A new approach to the strategic inventory/routing problem. Europ J Oper Res 85:132–148
75. Wikner J, Towill DR, Naim M (1991) Smoothing supply chain dynamics. Internat J Production Economics 22(3):231–248
76. Yano CA, Gerchak Y (1989) Transportation contracts and safety stocks for just-in-time deliveries. J Manufacturing Oper Management 2:314–330

Optimal Control of a Flexible Arm

W. Krabs
Department Math. Techn., University Darmstadt, Darmstadt, Germany

MSC2000: 93-XX

Article Outline

Keywords
Mechanical Model
The Problem of Controllability
and Minimum Norm Controllability
Solvability of the Problem of Controllability
See also
References

Keywords

Moment theory; Optimal control

Mechanical Model

Considered is the rotation of a *flexible arm* in a horizontal plane around an axis through the arm's fixed end and driven by a motor whose torque is controlled. The equations of motion of the motor are given by

$$\begin{cases} \dot{\Theta}(t) = \omega(t), \\ \dot{\omega}(t) = u(t) = \frac{\tau(t)}{J}, \end{cases} \tag{1}$$

where Θ is the angle of rotation, ω is the angular velocity, τ is the torque generated by the motor, and J is the moment of inertia of the motor and the arm. J is assumed to be constant, since the displacement of the arm due to the vibration caused by the rotation is assumed to be small.

In addition, we assume the arm to be homogeneous and of length 1. Following [6] the displacement $y = y(t, x)$ of the arm from the rotating zero line is modeled by the differential equation

$$y_{tt}(t,x) + \alpha y_{xxxx}(t,x) - \frac{\omega(t)^2}{2}\frac{\partial}{\partial x}[(1-x^2)y_x(t,x)] - \omega(t)^2 y(t,x) = -xu(t) \quad \text{for all } t \in (0,T) \text{ and } x \in (0,1), \tag{2}$$

where $T > 0$ is some given time and $\alpha = EI\rho$ with E being Young's modulus of the arm material, I being the moment of inertia of the cross-section of the arm, and ρ being the mass per unit length.

The left end of the arm is clamped, and the right end is free. This leads to the boundary conditions

$$y(t,0) = y_x(t,0) = y_{xx}(t,1) = y_{xxx}(t,1) = 0 \quad \text{for all } t \in [0,T]. \tag{3}$$

At the beginning of the motion the arm is assumed to be in rest which leads to the initial conditions

$$y(0,x) = y_t(0,x) = 0 \quad \text{for all } x \in (0,1) \tag{4}$$

and

$$\Theta(0) = \dot{\Theta}(0) = 0. \tag{5}$$

The Problem of Controllability and Minimum Norm Controllability

Let some angle $\Theta_T \in \mathbf{R}$ with $\Theta_T \neq 0$ be prescribed. Then we look for some control function $u \in L^2(0, T)$ such that the solution $y = y(t, x)$, $t \in [0, T]$, $x \in [0, 1]$ of (2), (3) and (4) satisfies the end conditions

$$y(T,x) = y_t(T,x) = 0 \quad \text{for all } x \in (0,1) \tag{6}$$

and the angle $\Theta = \Theta(t)$, $t \in [0, T]$, of rotation satisfies the end conditions

$$\Theta(T) = \Theta_T \quad \text{and} \quad \dot{\Theta}(T) = 0. \tag{7}$$

If this problem of *controllability* is solvable, then a control function $u \in L^2(0, T)$ is looked for which solves the problem of controllability and whose norm

$$\|u\|_{L^2(0,T)} = \left(\int_0^T u(t)^2 \, \mathrm{d}t\right)^{\frac{1}{2}}$$

is as small as possible.

On using (1) and (5) we get

$$\dot{\Theta}(t) = \omega(t) = \int_0^t u(s) \, \mathrm{d}s,$$
$$\Theta(t) = \int_0^t (t-s)u(s) \, \mathrm{d}s$$

for $t \in [0, T]$ so that the differential equation (2) can be rewritten in the form

$$y_{tt}(t,x) + \alpha y_{xxxx}(t,x) - \left(\int_0^T u(s) \, \mathrm{d}s\right)^2 \times \left\{\frac{1}{2}\frac{\partial}{\partial x}[(1-x^2)y_x(t,x)] + y(t,x)\right\} = -xu(t) \tag{8}$$

for $t \in (0, T)$ and $x \in (0, 1)$

and the end conditions (7) are equivalent to

$$\begin{cases} -\int_0^T tu(t)\,\mathrm{d}t = \Theta_T\,, \\ \int_0^T u(t)\,\mathrm{d}t = 0\,. \end{cases} \tag{9}$$

Solvability of the Problem of Controllability

At first we consider the special case where the cross-section areas behave like rigid bodies, i. e., they stay plane, do not change their measurements and do not rotate around their centers. Further, it is assumed that they stay orthogonal to the zero line. Then the differential equation (2) can be replaced by

$$y_{tt}(t,x) + \alpha y_{xxxx}(t,x) = -xu(t) \quad \text{for } t \in (0,T) \text{ and } x \in (0,1) \tag{10}$$

(see [1] and [2]).

In this case it can be shown that the problem of controllability is solvable for every $T > 0$ and the problem of *controllability with minimum norm* has a unique solution (see, for instance [3]). The main tool in the proof of this result is linear moment theory (see [4]).

In the general case where the displacement $y = y(t, x)$ of the arm from the rotating zero line is modeled by the differential equation (2), the problem of controllability is not exactly solvable as being shown in [5]. The main tool in the proof of this result is nonlinear moment theory. However, if one determines the unique control $u = u^1 \in L^2(0, T)$ which satisfies (9) with minimal norm such that the corresponding solution $y = y^1 = y^1(t, x)$ of (10), (3) and (4) satisfies the end conditions (6) and then determines the unique control $u = u^2 \in L^2(0, T)$ which satisfies (9) with minimum norm such that the solution $y = y^2 = y^2(t, x)$ of the differential equation

$$y^2_{tt}(t,x) + \alpha y^2_{xxxx}(t,x) = \left(\int_0^t u^1(s)\,\mathrm{d}s\right)^2 \times \left\{\frac{1}{2}\frac{\partial}{\partial x}[(1-x^2)y^1_x(t,x)] + y^1(t,x)\right\} - xu^2(t)$$

for $t \in (0, T)$, $x \in (0, 1)$ the boundary conditions (4), and the initial conditions (3) satisfies the end conditions (4), then the solution $y = y^* = y^*(t, x)$ of

$$y^*_{tt}(t,x) + \alpha y^*_{xxxx}(t,x) - \left(\int_0^t u^2(t)\,\mathrm{d}t\right)^2 \times \left\{\frac{1}{2}\frac{\partial}{\partial x}[(1-x^2)y^*_x(t,x)] + y^*(t,x)\right\} = -xu^2(t)$$

for $t \in (0, T)$, $x \in (0, 1)$, the boundary conditions (3), and the initial conditions (4) satisfies the end conditions (6) up to a very small error.

See also

- Control Vector Iteration
- Duality in Optimal Control with First Order Differential Equations
- Dynamic Programming: Continuous-time Optimal Control
- Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- Dynamic Programming: Optimal Control Applications
- Hamilton–Jacobi–Bellman Equation
- Infinite Horizon Control and Dynamic Games
- MINLP: Applications in the Interaction of Design and Control
- Multi-objective Optimization: Interaction of Design and Control
- Robust Control
- Robust Control: Schur Stability of Polytopes of Polynomials
- Semi-Infinite Programming and Control Problems
- Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- Suboptimal Control

References

1. Ballas K (1994) Steuerung eines rotierenden flexiblen Balkens mit einem Drehmoment von minimaler L2-Norm. Diplomarbeit Darmstadt
2. Delfour MC, Kern M, Passeron L, Sevenne B (1986) Modelling of a rotating flexible beam. In: Rand HE (ed) OxfordControl of Distributed Parameter Systems (4th IFAC Symp., Los Angeles). Pergamon, pp 383–387
3. Krabs W (1983) On time-optimal boundary control of vibrating beams. In: Kappel F, Kunisch K, Schappacher W (eds) Control Theory for Distributed Parameter Systems and Applications. Lecture Notes Control Inform Sci, vol 54. Springer, Berlin, pp 127–137

4. Krabs W (1992) On moment theory and controllability of one-dimensional vibrating systems and heating processes. Lecture Notes Control Inform Sci, vol 173. Springer, Berlin
5. Krabs W, Long ChiNguyen (1998) On the controllability of a robot arm. Math Meth Appl Sci 21:25–41
6. Sakawa Y, Ito R, Fujii N (1983) Optimal control of rotation of a flexible arm. In: Kappel F, Kunisch K, Schappacher W (eds) Control Theory for Distributed Parameter systems and Applications. Lecture Notes, vol 54. Springer, Berlin, pp 175–187

Optimal Design of Composite Structures

ZELDA B. ZABINSKY
Industrial Engineering, University Washington, Seattle, USA

MSC2000: 90C29, 90C26

Article Outline

Keywords
Point Design
Blended Panel
Sample Problem
COSTADE
See also
References

Keywords

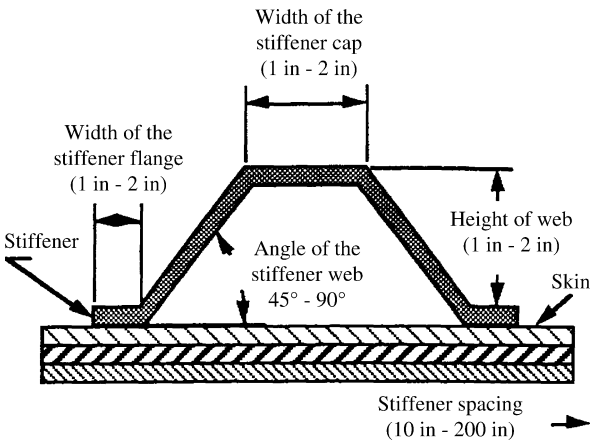
Global optimization; Random search algorithms; Simulated annealing; Improving hit and run; Hit and run methods; Mixed discrete-continuous global optimization; Design of composite structures; Structural optimization

Designing a composite stiffened panel is a complicated process, and optimization techniques are proving helpful to engineers in designing a cost-effective, practical structure. Random search algorithms (cf. also ► Random search methods), such as simulated annealing and genetic algorithms, are being used in the *design of composite structures* [3]. A benefit of these algorithms is that they can optimize functions that cannot be handled with traditional optimization techniques. In this article, a *structural design* problem for composite fuselage is described which involves solving a mixed integer global optimization problem. The objective function may be discontinuous, have many local optima, and the feasible region may be disconnected. The random search algorithm, *improving hit and run* [23], that is used to solve these composite structural optimization problems is also described in ► Global optimization: Hit and run methods. Although a *random search algorithm* can only provide a probabilistic guarantee of finding the global optimum, the benefits of having a more realistic formulation outweigh the disadvantage of not having a 100% guarantee of finding the global optimum. The near optimal solutions found using this approach have been well-received by Boeing engineers and have demonstrated significant reductions in weight and cost [15].

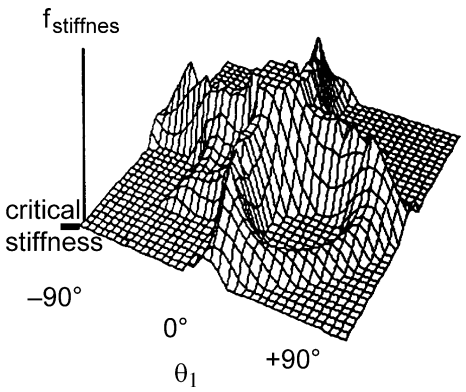
Three optimization formulations for a composite design problem will be described, each increasing in complexity and incorporating more realism. The first two formulations are point designs, where a single cross-section of a composite panel is optimized. The first formulation assumes a fixed number of plies, while the second one allows the number of plies to be a variable in the optimization. The third formulation extends a 'point' optimization to a 'blended' panel optimization, by dividing a panel into elements. This third formulation is applied to the design of a large panel with varying cross-sections. Manufacturing considerations lead to constraints in the third formulation to ensure a practical, consistent, panel. A sample problem is also presented.

Point Design

The first formulation optimizes a laminated composite structure, with skin and stiffeners. Laminated composites are composed of several thin layers, called plies, which are bonded together to form a composite laminate. A single ply consists of long reinforcing fibers (e. g., graphite fibers), embedded within a relatively weak matrix material (e. g., epoxy). Composite laminates are usually fabricated such that all fibers within an individual ply are oriented in one direction, however the angle may vary from ply to ply. The angle of the fibers in the ith ply is denoted by θ_i. The design variables for the first formulation include: the fiber an-



Optimal Design of Composite Structures, Figure 1
Design variables for a hat stiffened composite laminate



Optimal Design of Composite Structures, Figure 2
Graph of in-plane stiffness for a four ply symmetric laminate, $[\theta_1, \theta_2, \theta_2, \theta_1]$

gles for each ply in the skin and stiffeners, denoted θ_i^{skin} for $i = 1, \ldots, n^{\text{skin}}$ and $\theta_i^{\text{stiffener}}$ for $i = 1, \ldots, n^{\text{stiffener}}$, and stiffener geometry variables; height, width of cap, width of flange, angle of web, and stiffener spacing, as shown schematically in Fig. 1. In the first formulation, it is assumed that the numbers of plies in the skin and stiffeners, n^{skin} and $n^{\text{stiffener}}$, are fixed. This assumption is relaxed in the second formulation, where the number of plies will also be allowed to vary.

The first optimization formulation is stated as:

$$\begin{cases} \min & f(x) \\ \text{s.t.} & g_j(x) \geq 0 \quad \text{for } j = 1, \ldots, m, \\ & x_i^l \leq x_i \leq x_i^u \quad \text{for } i = 1, \ldots, n, \end{cases} \tag{1}$$

where the n variables consist of the $n^{\text{skin}} + n^{\text{stiffener}}$ fiber angles and five stiffener geometry variables, and are real-valued, $x \in \mathbf{R}^n$. The objective function, $f(x)$, may be cost, weight, or a combination of cost and weight, $f(x) = \delta_{\text{cost}} * f_{\text{cost}}(x) + \delta_{\text{weight}} * f_{\text{weight}}(x)$. An alternate objective may be to maximize performance, such as maximize margin of safety.

The constraints are composed of simple upper and lower bounds on the variables, $x_i^l \leq x_i \leq x_i^u$, and the more complicated structural mechanics constraints, $g_j(x) \geq 0$. The structural mechanics constraints are composed of margins of safety for strain, strength, damage tolerance and buckling analyses [16]. The inequality constraints are formed so that a feasible design has a positive margin of safety. This first formulation has been described in [1,2,4,20].

The margin-of-safety functions to be used as constraints and/or the objective function can be described as black-box functions where the functions often are only available in the form of a computer subroutine. For example, stiffness of a composite laminate can be calculated using classical lamination theory [5] or it could involve a finite element analysis [10,12].

To illustrate the global nature of these equations, a plot of the in-plane stiffness of a four ply, symmetric laminate, $[\theta_1, \theta_2, \theta_2, \theta_1]$, using classical lamination theory, is shown in Fig. 2. The greatest in-plane stiffness occurs when the fiber angles are all 0 degrees, as makes sense intuitively. See [20] for the equations used to generate the plot. The plateau in the graph represents infeasible designs, with a stiffness that is less than a prescribed critical value. If stiffness is used as an objective function, it can be seen to be nonlinear and nonconvex. If stiffness is used in the constraints to allow only those design above a threshold of critical stiffness, the feasible region itself is nonconvex and even has holes in it. This indicates what a difficult problem even attaining feasibility can be.

It is also possible to define a hierarchy of objectives. For example, we might minimize cost and when there is a tie on cost, we minimize weight, and when there is a tie on weight, then maximize margin of safety. This hierarchy is natural to the formulation because weight is not directly affected by the fiber angles, but varying the fiber angles can increase margin of safety while maintaining a low cost and weight. A two phase approach was used, where phase 1 maximized the min-

imum margin of safety until a positive value was obtained, and then phase 2 maintained feasibility while optimizing the hierarchy of cost, weight and margin of safety. A contrasting approach using a penalty method with a variable penalty factor was compared numerically in [10,11]. Numerically the penalty method was better than the hierarchical approach, although the hierarchical approach was more intuitive to the engineers.

An improvement to the first formulation involves relaxing the requirement to specify the numbers of plies in the skin and stiffener. Since the composite laminate is manufactured by laying down individual plies, it is realistic to treat the number of plies as an integer variable. Other optimization techniques have treated thickness of a ply as a continuous variable, and then rounded to the appropriate number of plies [3,17]. This is not as accurate as treating the number of plies as integer variables directly. The second formulation is very similar to the first, with additional binary variables to indicate whether a ply exists in the laminate; t_i^{skin} for $i = 1, \ldots, n^{\text{skin}}$ and $t_i^{\text{stiffener}}$ for $i = 1, \ldots, n^{\text{stiffener}}$, where $t_i = 1$ if ply i exists and takes on fiber angle θ_i, and $t_i = 0$ if ply i is dropped from the laminate. The upper bounds on the number of plies needed in the skin and stiffener, n^{skin} and $n^{\text{stiffener}}$ are now easier to provide and not a critical aspect of the inputs.

The complete second optimization formulation, including the binary variables is summarized as:

$$\begin{cases} \min & f(x,t) \\ \text{s.t.} & g_j(x,t) \geq 0 \quad \text{for } j = 1, \ldots, m, \\ & x_i^l \leq x_i \leq x_i^u \quad \text{for } i = 1, \ldots, n, \\ & t_i \in \{0,1\} \quad \text{for } i = 1, \ldots, n', \end{cases} \tag{2}$$

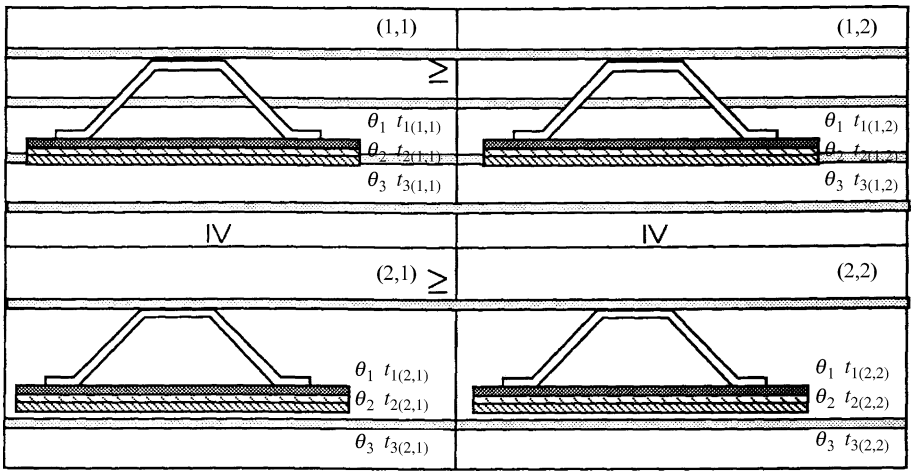
where n is the total number of continuous variables, including fiber angles and geometry variables, and n' is the largest number of plies, $n' = n^{\text{skin}} + n^{\text{stiffener}}$.

The random search algorithm had to be modified in order to solve the second formulation. See ► Global optimization: Hit and run methods, or [10,11,13] for details on the algorithm for *mixed discrete-continuous global optimization*. Once this capability was available, it was possible to create other discrete variables. The most interesting are the fiber angles, which may take on continuous values between +90 and −90 degrees, but for practical purposes are often restricted to discrete values, such as 0, ± 45, or 90 degrees. This flexibility in the formulation provides the ability to investigate manufacturing considerations. For example, running the optimization problem allowing the fiber angles to be continuous values may not be practical for manufacturing, but can provide a lower bound on weight. Then the problem can be run again to investigate the additional weight associated with using a discrete set of fiber angles. In this way, tradeoffs can be carefully evaluated early in the design process.

Blended Panel

The third formulation expands the optimization problem beyond considering a single point to include an entire panel. In the point optimization, it is assumed that the loads are constant and uniformly distributed over the section, and the design is also constant over the section. To be more realistic, the loads over a large panel such as a crown panel in a fuselage, are not evenly distributed. We could use the second formulation for the entire panel using the heaviest loads, but then we would essentially overdesign parts of the panel where the loads are much lighter. Therefore, in the third formulation, the panel is divided up into elements, where the loads are assumed constant for each element, but the loads may vary between different elements [16,19]. The elements cannot be point-optimized individually because the result may be impractical for manufacturing. For example, the stiffener spacing could increase and decrease in adjacent elements, or a 45^o ply could change to a -30^o in adjacent elements. This would not be considered manufacturable, since it is assumed that once a ply is in place, the orientation of the fiber angle cannot change, although the ply may be dropped off. The third formulation must ensure that the elements can be produced into a consistent panel, or what we call a 'blended' panel. By including binary variables for each ply in each element, 'blending rules' are established to reflect these manufacturing considerations.

Figure 3 illustrates the main blending rule for a panel divided into four elements. Consider any ply as it passes through the four elements. The fiber orientation of ply i is θ_i, and there are also four binary integer variables associated with the ply. As described in



Optimal Design of Composite Structures, Figure 3
Design variables for a blended panel

the second formulation, if the binary variable $t_{i(j,k)} = 1$ then ply i in element (j, k) exists with fiber angle θ_i, and if $t_{i(j,k)} = 0$ then ply i has been dropped in element (j, k). By adding constraints on the binary variables, we can force plies to be dropped in such a way that the panel is manufacturable. We assume that the heaviest load on the panel exists in the upper left corner, and we want to be able to drop a ply, but once it is dropped we do not allow it to be added back into the panel. Thus the main blending rule is nicknamed the 'less-than-or-equal-to rule', and includes the following constraints:

$$t_{i(j,k)} \geq t_{i(j,k+1)},$$
$$t_{i(j,k)} \geq t_{i(j+1,k)}$$

for all plies i, and for all rows j and columns k of the panel.

For the panel illustrated in Fig. 3, the above constraints would be:

$$t_{i(1,1)} \geq t_{i(1,2)},$$
$$t_{i(2,1)} \geq t_{i(2,2)}$$

and

$$t_{i(1,1)} \geq t_{i(2,1)},$$
$$t_{i(1,2)} \geq t_{i(2,2)}.$$

In the example, if ply i exists in element (1, 1) but is dropped in element (1, 2), then it must also be dropped in element (2, 2). Ply i may exist or be dropped in element (2, 1) and still satisfy the blending rule. This blending rule has been very useful in structuring the optimization over a panel where the loads are allowed to vary, and a realistic design would make use of ply drops.

To summarize the third formulation, the second formulation is expanded to span several elements, and the blending rule is introduced as additional constraints:

$$\begin{cases} \min & f(x,t) \\ \text{s.t.} & g_j(x,t) \geq 0, \quad j = 1,\ldots,m, \\ & x_i^l \leq x_i \leq x_i^u, \quad i = 1,\ldots,n', \\ & t_{i(j,k)} \geq t_{i(j,k+1)}, \quad i = 1,\ldots,n, \\ & t_{i(j,k)} \geq t_{i(j+1,k)}, \quad i = 1,\ldots,n, \\ & t_{i(j,k)} \in \{0,1\}, \quad i = 1,\ldots,n, \end{cases} \tag{3}$$

for all defined elements (j, k).

Another aspect in blending of a panel has to do with the variation allowed in stiffener geometry across elements. For example, stiffener spacing cannot vary in the axial direction, but is allowed to vary across elements in the hoop direction. In the third formulation, we include a design variable for stiffener spacing, x_j^{stsp} that is only subscripted by row of the elements, which corresponds to the hoop direction. For example, in the illustration in Fig. 3 the stiffeners are one unit apart in the first row where the loads are heaviest, and two units apart in the second row which has lighter loads. No additional constraints are needed to incorporate this re-

striction, since the definition of the variable has ensured that stiffener spacing is constant in the axial direction. The other geometry variables are also allowed to vary across rows in the hoop direction, but remain constant in the axial direction. For example, as the stiffeners get spaced farther apart, they may also become shorter. In the sample problem presented here, it was assumed that the height of a stiffener remained constant for its entire length.

The main difference between the second and third formulation is the difference between a point design and a 'blended' panel design. The panel design is of greater use to the engineers, but there is a penalty in computational complexity. The addition of elements for a *blended panel* greatly increases the number of design variables for the optimization problem. Suppose the panel is divided into $P * Q$ elements. Then there will be $(n^{\text{skin}} + n^{\text{stiffener}}) * P * Q$ binary integer variables. There will be $(n^{\text{skin}} + n^{\text{stiffener}})$ continuous variables for the fiber angles, and $5 * P$ continuous geometry variables. An alternative formulation has been developed to reduce the number of variables by using integer variables that capture the number of elements, or distance, for which a ply is runs. Details of this formulation and numerical results are presented in [6,14,22].

We next present a sample problem and discuss the differences between a point design and a 'blended' panel design.

Sample Problem

The sample problem presented here is taken from [18], and is intended to demonstrate the difference between the point formulation and the blended panel formulation. It uses two sets of loading conditions for each element as shown in Table 1., and uses the material properties associated with AS4–3501 graphite/epoxy, as in [1]. There are $2 * 3$ elements with a maximum number of 16 plies (symmetric) in the skin and stiffeners. For this sample problem there are 96 binary integer variables, and 26 continuous variables, and the computer algorithm used 25,000 function evaluations.

For the sample problem we first did a point by point optimization for each element independently to obtain a lower bound on weight, and to observe the trends of the designs. The point by point optimization results for this sample problem are presented in Table 2. Notice

Optimal Design of Composite Structures, Table 1
Loading conditions for the sample problem, where N_x, N_y, and N_{xy} are in lb and M_x, M_y, M_{xy} are in lb-in

N_x	3000	3000	2500	2500	2500	2500
N_y	2000	2000	1500	1500	1000	1000
N_{xy}	1500	−1500	1500	−1500	1000	−1000
M_x	2000	2000	2000	2000	2000	2000
M_y	0	0	0	0	0	0
M_{xy}	0	0	0	0	0	0
N_x	2500	2500	2000	2000	2000	2000
N_y	2000	2000	1500	1500	1000	1000
N_{xy}	1000	−1000	1000	−1000	500	−500
M_x	2000	2000	2000	2000	2000	2000
M_y	0	0	0	0	0	0
M_{xy}	0	0	0	0	0	0

that fiber angles vary in a way that would be impractical for manufacturing. For example, the fiber angles vary drastically between adjacent elements, such as between element (1, 1) and element (1, 2). Also notice that stiffener spacing also varies in an impractical fashion. In the first row, stiffener spacing changes from 21 inches, to 22 inches, and then up to 30 inches. This is not a manufacturable design. Table 3 depicts the optimal design of the blended panel using the third formulation. Although the overall weight increased, the optimal design is now considered a practical design. Notice that the first ply in the first element of 35^o stays in the entire panel, while the second ply in the first element of 26^o gets dropped immediately. This type of tailored ply dropoffs is manufacturable, and makes use of the ability to tailor composite materials. Also, the stiffener spacing is fixed in the axial direction at 20.6 inches in the first row, as desired, and has a slight change to 20.8 inches in the second row. This also satisfies the blending rule.

The formulations presented thus far have been for a stiffened composite panel, but the point and the blended panel formulations have also been extended to a sandwich composite panel, as depicted in Fig. 4. A sandwich panel consists of an inner core, with plies on the outside. Typically the depth of the entire panel is constant, but the core increases to compensate for plies that are dropped. The sandwich formulation was used to design a fuselage keel panel in [7].

Optimal Design of Composite Structures, Table 2
Point by point optimization of the sample problem

Skin	12 plies $[49/52/-42/10/-45/-72/]_s$	10 plies $[36/45/-39/-82/-39]_s$	8 plies $[25/50-25/-51]_s$
Stiffener	8 plies $[1/-47/1/53]_s$	10 plies $[28/17/-23/-20/-80]_s$	10 plies $[5/-8/5/-53/53]_s$
Spacing	21 inches	22 inches	30 inches
Weight	634 E − 5 lb/in^2	577 E − 5 lb/in^2	450 E − 5 lb/in^2
Skin	10 plies $[-85/55/34/-34/-44]_s$	8 plies $[-36/77/-38/35]_s$	6 plies $[1/57/-48]_s$
Stiffener	8 plies $[41/-3/-58]_s$	12 plies $[-10/14/42/-7/-59/-59]_s$	10 plies $[-33/69/12/-17/14]_s$
Spacing	21 inches	24 inches	24 inches
Weight	548 E − 5 lb/in^2	506 E − 5 lb/in^2	396 E − 5 lb/in^2

- The overall weight of this nonblended panel is 3, 111 E − 5 lb/in^2.
- The stiffener geometry variables were always at their upper and lower bounds; Height: 2 inches; Width of flange: 1 inch; Width of cap: 2 inches, and Angle of web: 90 degrees.

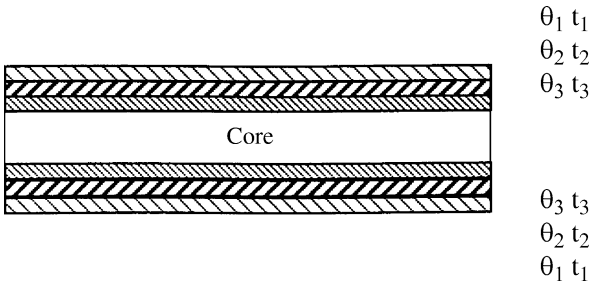
Optimal Design of Composite Structures, Table 3
Blended panel optimal design for the sample problem

Skin	12 plies $[35/26/-35/41/-42/-89/]_s$	10 plies $[35/-35/41/-42/-89]_s$	8 plies $[35/-35/41/-42/-89]_s$
Stiffener	12 plies $[-3/33/-53/-2/53/-88]_s$	10 plies $[-3/33/-53/-2/53]_s$	8 plies $[-3/-53/-2/53]_s$
Spacing	20.6 inches	20.6 inches	20.6 inches
Weight	702 E − 5 lb/in^2	585 E − 5 lb/in^2	552 E − 5 lb/in^2
Skin	10 plies $[35/-35/41/-42/-89]_s$	10 plies $[35/-35/41/-42/-89]_s$	8 plies $[35/-35/41/-89]_s$
Stiffener	12 plies $[-3/33/-53/-2/53/-88]_s$	10 plies $[-3/33/-53/-2/53]_s$	8 plies $[-3/-53/-2/53]_s$
Spacing	20.8 inches	20.8 inches	20.8 inches
Weight	616 E − 5 lb/in^2	583 E − 5 lb/in^2	466 E − 5 lb/in^2

- The overall weight of this nonblended panel is 3.504 E − 5 lb/in^2.
- The stiffener geometry variables were always at their upper and lower bounds: Height: 2 inches; Width of flange: 1 inch; Width of cap: 2 inches, and Angle of web: 90 degrees.

COSTADE

Through the collective efforts of Boeing, NASA, the University of Washington and others, a preliminary design software package called *COSTADE* (Cost/Composite Optimization Software for Transport Aircraft Design Evaluation) has been developed [7,8,9]. The three optimization formulations described here are



Optimal Design of Composite Structures, Figure 4
Design variables for a sandwich composite panel

available in the software. The optimization software has been applied to the design of aircraft composite panels, including a crown panel [15], a keel panel [7], a window belt [9], and most recently to a full fuselage barrel [10]. Research continues in defining more general blending rules [6], and more accurately reflecting the manufacturing considerations. The modified hit and run algorithm has been robust enough to solve the mixed integer-continuous global optimization problem with a hierarchy of objective functions efficiently. The design optimization has proved to be an effective aid in the design process of composite structures.

See also

- ► Bilevel Programming: Applications in Engineering
- ► Design Optimization in Computational Fluid Dynamics
- ► Interval Analysis: Application to Chemical Engineering Design Problems
- ► Multidisciplinary Design Optimization
- ► Multilevel Methods for Optimal Design
- ► Optimal Design in Nonlinear Optics
- ► Structural Optimization: History

References

1. Graesser DL, Zabinsky ZB, Tuttle ME, Kim GI: (1991) Designing laminated composites using random search techniques. Composite Structures 18:311–325
2. Graesser DL, Zabinsky ZB, Tuttle ME, Kim GI (1993) Optimal design of composite structures. Composite Structures 24:273–281
3. Haftka RT, Gürdal Z (1992) Elements of structural optimization. Kluwer, Dordrecht
4. Heglund DA, Zabinsky ZB, Tuttle ME, Graesser DL, Kim GI (1992) Optimization in preliminary structural design. In: Proc. 15th Annual IIE Aerospace and Defense Conference
5. Jones RM (1975) Mechanics of composite materials. Scripta Book Company, Washington, DC
6. Kristinsdottir BP (1997) Analysis and development of random search algorithms. PhD Thesis, University Washington
7. Mabson GE, Flynn BW, Ilcewicz LB, Graesser DL (1994) The use of COSTADE in developing composite commercial aircraft fuselage structures. In: Proc. 35th AIAA/ASME/ASCE/AHS/ASC Structure, Structural Dynamics, and Materials Conference, Hilton Head
8. Mabson GE, Fredrikson HG, Graesser DL, Metschan SL, Proctor MR, Stogin DC, Tervo DK, Zabinsky ZB, Tuttle ME, Gutowski TG (1995) Cost optimization software for transport aircraft design evaluation. In: Proc. Sixth NASA/DOD/ARPA Advanced Composite Techn. Conference
9. Metschan SL, Graesser DL, Mabson GE, Proctor MR, Tervo DK, Ilcewicz LB (1994) Manufacturing data for Costade analysis of composite fuselage panels. In: Proc. 5th NASA ACT Conference
10. Neogi S (1997) Design of large composite structures using global optimization and finite element analysis. PhD Diss, Univ Washington
11. Neogi S, Zabinsky ZB, Tuttle ME (1994) Optimal design of composites using mixed discrete and continuous variables. In: Proc. ASME Winter Annual Meeting, Symp. Processing, Design and Performance of Composite Materials, MD 52, pp 91–107
12. Neogi S, Zabinsky ZB, Tuttle ME, Kristinsdottir BP (1995) Load redistribution issues in optimal design of composites airplane structures. In: Proc. ICCE/2 Conference, New Orleans
13. Romeijn HE, Zabinsky ZB, Graesser DL, Neogi S (1999) Simulated annealing for mixed integer/continuous global optimization. J Optim Th Appl 101(2):403–427
14. Seifer JD (1995) Incorporating realistic engineering considerations into the optimal design of composite structures. Master's Thesis, Univ Washington
15. Swanson GD, Ilcewicz LB, Walker T, Graesser DL, Tuttle ME, Zabinsky ZB (1991) Local design optimization for transport fuselage crown panels. In: Proc. 9th DoD/NASA/FAA Conf. Fibrous Composites in Structural Design, Lake Tahoe
16. Tuttle ME, Zabinsky ZB (1994) Methodologies for optimal design of composite fuselage crown panels. In: Proc. 35th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Hilton Head
17. Vanderplaats GN, Weisshaar TA (1989) Optimum design of composite structures. Internat J Numer Methods in Eng 27:437–448
18. Zabinsky ZB (1994) Global optimization for composite structural design. In: Proc. 35th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Material Conference, Hilton Head

19. Zabinsky ZB (1998) Stochastic methods for practical global optimization. J Global Optim 13:433–444
20. Zabinsky ZB, Graesser DL, Tuttle ME, Kim GI (1992) Global optimization of composite laminate using Improving Hit and Run. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton University Press, Princeton, pp 343–365
21. Zabinsky ZB, Neogi S, Tuttle ME (1997) Practical global optimization for engineering design. In: 16th Internat. Symp. Mathematical Programming, Lausanne
22. Zabinsky ZB, Seifer JD, Tuttle ME, Kristinsdottir BP, Neogi N (1995) Optimal design of composite panels with varying loads. In: Proc. ICCE/2 Conference, New Orleans
23. Zabinsky ZB, Smith RL, McDonald JF, Romeijn HE, Kaufman DE (1993) Improving hit and run for global optimization. J Global Optim 3:171–192

Optimal Design in Nonlinear Optics

Gang Bao[1], Guangye Li[2]
[1] Department Math., Michigan State University, East Lansing, USA
[2] Silicon Graphics, Inc., Houston, USA

MSC2000: 34A55, 90C30, 78A60

Article Outline

Keywords
Model Formulation
Collocation Scheme
Domain Decomposition
Optimization
Numerical Experiment
Discussion
See also
References

Keywords

Optimal design; Parameter identification; ODE two-point boundary value problem; Nonlinear optics

Over the past two decades, significant technology advances have been made in *nonlinear optics* due to rapid developments of laser technology and nonlinear optical materials. Applications of nonlinear optics are everywhere, for example, lasers, spectroscopy, optical switching, parametric amplifiers and oscillators, optical computing, and communications. A remarkable application is to generate powerful coherent radiation at a frequency that is twice that of available lasers, so-called second harmonic generation (SHG). However, nonlinear optical effects are generally very weak. In order to obtain useful nonlinear optical effects, several methods may be employed. First, extremely high intensity laser beams or materials with very high nonlinearities could be used. Unfortunately, limited by the availability of technology and high costs of such lasers/materials, this method is often impractical. Another method is to increase the effective nonlinearity of the medium by using composite materials. Such a method is currently an active research topic in material sciences. The third method is a structure assisted method. The idea is to enhance the nonlinear interaction between the material and the light by using gratings, waveguide, or other diffractive structures. The advantage is that the method is very practical and can make good use of available lasers and materials [10].

This work is devoted to *optimal design* or *parameter identification* problems that arise in modeling of nonlinear optical thin films [2]. We shall restrict our attention to second order nonlinear effects, which are the simplest and representative to other nonlinear effects. The following problems are of particular interest: From the measured transmittance and reflectance at both frequencies, what kind of information might be retrieved about the medium? Given nonlinear films and coating materials in a multi-layered form, maximize the transmittance of the second harmonic field, i. e., maximize the nonlinear optical effects. Note that in the simplest setting, the problem may be formulated as a *two-point boundary value problem* for first order nonlinear system of ordinary differential equations (ODEs). One goal of this research is to identify physical properties of the medium by probing the medium with light or other energy sources. Another goal is to design new materials with desirable properties, i. e., solving an optimal design problem.

Throughout we shall view the optimal design problem as a parameter identification problem. Because of important applications in diverse areas of science and engineering, parameter identification problems for nonlinear systems of ODEs have been studied extensively. The advent of computers and parallel machines has greatly accelerated activity in this area and has

driven the need for efficient computational methods and algorithms. For closely related parameter identification for nonlinear ODE initial value problems, three general approaches have been developed. We refer to [5] for a detailed discussion. Among them, the black box approach is the simplest one which essentially separates the numerical solution of differential equations from the optimization process. The ODE solver is treated as a 'black box' with a minimum communication with the optimization procedure. However, there is a severe drawback of the approach. Due to the nonlinearity of the problem, the model often does not have a solution for all parameters. Without the necessary communication, the optimization algorithm may ask for a solution to the ODE with parameter values for which the ODE solver fails. The second approach, the all-together approach, first discretizes the ODE system and then formulates the identification problem as an optimization problem with equality constraints resulting from the ODE discretization. The third approach often referred to as the domain decomposition or in-between approach divides the interval into a number of subintervals, and uses a black box procedure to solve the problem on each subinterval. In order to solve the ODE system over the entire interval, matching conditions are introduced to patch together solutions over subintervals. Recently, J.E. Dennis, G. Li, and K.A. Williamson [5] have developed two families of algorithms, the in-between and altogether approaches, for solving ODE inverse problems. Their model problem is a nonlinear initial value problem. Efficient algorithms are introduced by a nonlinear programming formulation of the problem coupled with an orthogonal collocation scheme for solving the model ODE. The general idea is to tailor the algorithm to best fit the level of interaction between the optimization algorithm and the ODE solution technique.

In this article, we propose a domain decomposition approach for solving the parameter identification problem of nonlinear ODE two-point boundary value problems. We introduce an in-between approach for solving the parameter identification problems for nonlinear ODE two point boundary value problems, which has the following distinct features:

1) The problem is formulated as a constrained optimization problem as opposed to the standard data fitting least squares problem. The set of variables is decomposed into two parts: a set of explicit variables and a set of implicit variables, which is shown to be efficient through a complexity analysis. The continuity and boundary conditions are treated as explicit constraints for the optimization problem. Hence, the parameters and the function values at the subdivision points are EXPLICIT variables only need to be satisfied and determined at the solution or the final step. At each iteration step, the differential equation (the IMPLICIT variables) is solved over each subinterval independently. Furthermore, the extremely large number of linear systems for computing the Jacobian, the gradient of the constraints, and the Hessian of the Lagrangian, which account for more than 95% of the total computation, are all solved completely independently over each of the subintervals.
2) The optimization approach adopted from [4] and [5] is a much refined version of the successive quadratic programming (SQP) together with a globalization strategy. The basic idea of the optimization approach together with a discussion on sparsity structures may be found in [5]. We point out that the approach is very complicated which involves computations of Jacobians and Hessians with respect to the implicit variables. Since our goal in this article is to present the main ideas of the in-between approach, the tedious technical details of the optimization techniques will be left out.

We follow the general idea of Dennis, Li, and Williamson [5]. However, the situation here is more complicated for the following reasons:

1) Unlike in [5], we do not assume availability of the data in the interior of the interval. This feature is essential in many practical applications, for example, in nondestructive testing. On the other hand, we use a set of samples that correspond to a set of experiments. The samples are taken, only on the boundary, by varying the sources.
2) Because the boundary value varies from sample to sample, the size of the system of differential equations is a multiple of the samples. Unlike in the initial value problem case where the unknowns are independent of the number of data points, the number of unknowns in our case is the number of sources (or experiments) multiplied by the number of unknowns introduced in the domain decomposition approach. As a result, the number of nonlinear equa-

tions and linear systems needed to be solved for implicit differentiations and computing the Hessian of the Lagrangian is a multiple of the square of the number of sources. Consequently, even with a moderate number of experiments, the computational and spatial cost may be so high that even main frame computers may not be able to solve it unless special efforts are taken to take advantage of the sparsity structures of the gradient of the constraints and the Hessian of the Lagrangian to reduce the number and the size of the resulting systems of linear and nonlinear equations.

Our approach has the flavor of the multiple or parallel shooting method for solving ODE two point boundary value problems [9]. We point out that a straightforward modification of the approach gives rise to a variant of the multiple shooting method. However, our emphasis here is not on solving the direct problem but rather on solving the parameter identification problem or the inverse problem.

Model Formulation

We make the following assumptions: the fields are transverse; the medium is stratified; and the surface is flat. Since the medium is stratified, the fields vary only in one direction. The transversality assumption allows us to reduce the Maxwell system to a system of Helmholtz equations.

Let us specify the geometry of the model. Assume that a slab of stratified nonlinear material (say, composed of many layers of different nonlinear media) is placed between two linear homogeneous materials, say in the domain $\Omega = (0, l)$. Suppose that the whole space is filled with material in such a way that the indexes of refraction $q_1(x)$ and $q_2(x)$ at frequencies ω_1 and $\omega_2 = 2\omega_1$, respectively, satisfy

$$q_j(x) = \begin{cases} q_{j1}, & x \geq l\,, \\ q_{j0}, & x \in \Omega\,, \\ q_{j2}, & x \leq 0\,, \end{cases}$$

for $j = 1, 2$, where q_{j1} and q_{j2} are fixed constants, and q_{j0} may be some piecewise constant.

Assume that a plane wave with electric field $(0, E_I e^{iq_{11}x}, 0)$ is incident on Ω from the above. Using the jump conditions, we can derive the following two-point boundary value problem

$$\begin{aligned} &\left(\frac{d^2}{dx^2} + q_1^2\right) E_1 = \chi_1 \overline{E}_1 E_2 \quad \text{in } \Omega\,, \\ &\left(\frac{d^2}{dx^2} + q_2^2\right) E_2 = \chi_2 E_1^2 \quad \text{in } \Omega \\ &E_1'(l) + iq_{11}E_1(l) = 2iq_{11}e^{iq_{11}l}E_I\,, \\ &E_2'(l) + iq_{21}E_2(l) = 0\,, \\ &E_1'(0) - iq_{12}E_1(0) = 0\,, \\ &E_2'(0) - iq_{22}E_2(0) = 0\,, \end{aligned}$$

where $E_1 = E_1(x, \omega_1)$, $E_2 = E(x, \omega_2)$, and χ_1, χ_2 characterize the nonlinearity of the medium at frequencies ω_1, ω_2, respectively. The most striking feature of second harmonic generation is that new frequency components (at ω_2) are present.

By introducing new variables, the system may be simplified as a first order system of ODEs. From now on, we shall consider the following two point boundary value problem for a general system of nonlinear ODEs

$$\begin{aligned} y' &= F(x, y, p, k)\,, \\ Ay(l) + By(0) &= g(k)\,, \end{aligned}$$

where $x \in (0, l)$ is the independent variable, $y = y(x) \in \mathbf{R}^{ny}$ is the solution, $p \in \mathbf{R}^{np}$ denotes the parameters, $k \in \mathbf{R}^{nd}$ represents the source terms, A and B are matrices possibly depending on the parameters p. Here ny is the number of dependent variables, np is the number of parameters, and nd is the number of samples or sources.

The direct problem is to determine solutions $y = y(x, p, k)$, given the parameters p and source terms $g(k)$. Mathematically, the problem is well understood. It is well known that due to the nonlinearities, existence of solutions is not obvious. Roughly speaking, it depends on the regularity of F. Further, even a solution does exist, because of the boundary conditions, it may not be unique. Throughout the article, we assume that the two-point boundary value problem has a unique solution.

The parameter identification problem or inverse problem is to determine the parameters p from the additional boundary data. We consider the data set: $\text{ydata}(0)_s$, $\text{ydata}(l)_s$, $s = 1, \ldots,$ nd, where once again nd

is the number of samples by varying k. Define

$$\Phi(p) = \frac{1}{2}\sum_{s=1}^{\mathrm{nd}} \|y(0;p;k_s) - \mathrm{ydata}(0)_s\|^2 + \frac{1}{2}\sum_{s=1}^{\mathrm{nd}} \|y(l;p;k_s) - \mathrm{ydata}(l)_s\|^2 \ ,$$

where $\|\cdot\|$ is the vector norm defined by $\|u\|^2 = u^{\mathsf{T}} \cdot u$.

We are interested in identifying the parameters in the least squares sense, i. e., to find a p^* which best fits:

$$\begin{cases} \min & \Phi(p) \\ \text{s.t.} & y' = F(x, y; p, k_s)\,; \\ \text{b.c.} & Ay(l;k_s) + By(0;k_s) = g(k_s)\,, \\ & \quad s = 1, \ldots, \mathrm{nd}\,. \end{cases}$$

Collocation Scheme

We next describe a collocation scheme for solving the nonlinear system of ODEs. Here, we follow the general procedure given in [5]. Let us begin by dividing the interval (0, l) into ns subintervals: $[x_i, x_{i+1})$, for $i = 1, \ldots,$ ns, and $x_1 = 0$, $x_{\mathrm{ns}} = \mathrm{l}$. On the ith subinterval, the function $y(x)$ may be approximated by a polynomial: $y(x) \approx \sum_j = 0^{\mathrm{nc}} z_{ij}\ \Psi_{ij}(x)$, where nc is the number of collocation points in one subinterval, $\{\Psi_{ij}(x)\}$ is a basis of Lagrange polynomials of degree nc at the points t_{ij}, z_{ij} are the collocation coefficients to be determined, and t_{ij} is the jth collocation point on the ith subinterval.

The collocation method requires the above piecewise polynomial approximation to satisfy the ODE at the collocation points on each of the subintervals. For the sth sample, $s = 1, \ldots,$ nd, solving for $\{z^s_{ij}\}$ leads to an approximation to y at the collocation points. This step leads to the following *collocation conditions*:

$$h^s_1(p,z) = 0\,,$$
$$\text{where for}\, s = 1, \ldots, \mathrm{nd},\ j = 1, \ldots, \mathrm{nc}, i = 1, \ldots, \mathrm{ns}\,,$$
$$h^s_1(p,z) = \sum_{k=0}^{\mathrm{nc}} \frac{d\Psi_{ik}(t_{ij})}{dx}\, z^s_{ik} - F(t_{ij}, z^s_{ij}; p)\,.$$

In order to approximate the solution $y(x)$ over the interval (0, l), we need to patch together approximations over subintervals. This can be done by enforcing *continuity conditions* at all end points of the subintervals except the two end points $x_1 = 0$, $x_{\mathrm{ns}} = \mathrm{l}$. The continuity conditions are natural by assuming that the solution is continuous:

$$h^s_2(p,z) = 0, \text{ where for} s = 1, \ldots, \mathrm{nd},\ i = 2, \ldots, \mathrm{ns}\,,$$
$$h^s_2(p,z) = z^s_{i0} - \sum_{k=0}^{\mathrm{nc}} z^s_{i-1,k}\Psi_{i-1,k}(t_{i0})\,.$$

In addition, we want to enforce the *boundary value conditions*:

$$h^s_3(p,z) = 0, \text{ where for}\, s = 1, \ldots, \mathrm{nd}\,,$$
$$h^s_3(p,z) = Az^s_{00} + B\sum_{k=0}^{\mathrm{nc}} z^s_{\mathrm{ns},k}\Psi_{\mathrm{ns},k}(l)\,.$$

Set for $q = 1, 2, 3,$

$$h_q = (h^1_q, \ldots, h^{\mathrm{nd}}_q)\,.$$

We then have by combining the above conditions, that

$$h(p,z) = 0, \quad h = (h_1, h_2, h_3)^{\top}\,. \tag{1}$$

Domain Decomposition

It follows that the parameter identification problem may be formulated as a nonlinear programming problem:

$$\begin{cases} \min\limits_{p,z} & \Phi(p,z) \\ \text{s.t.} & h(p,z) = 0\,. \end{cases}$$

If both the parameters and the collocation coefficients are treated as independent variables, i. e., the variables are treated all together, then the approach is called all together or all-at-once. In this approach, the nonlinear system of equations $h(p, z) = 0$ gives rise to a set of explicit constraints. Thus, p, z only need to satisfy the constraints at the solution. Readily, one can verify that the dimension of the problem or number of unknowns in this case is np+(ns+ nc× ns)×ny×nd. There are two potential drawbacks for this approach. First, by treating all of the variables as independent variables, the size of the resulting nonlinear programming problem can be very large. Sophisticated optimization techniques are impractical for large size problems. Also, the approach does not support parallel structures. It is difficult to make it efficient in a parallel environment.

In order to exploit parallelism and reduce the size of the nonlinear programming problem, we propose a domain decomposition or in-between approach which

follows in principle [5]. The basic idea of the domain decomposition approach is to identify a set of implicit constraints, i. e., constraints that are satisfied at every iteration. More specifically, for each sample, this approach allows us to treat only the model parameters p and a subset of collocation coefficients corresponding to the ends of subintervals z^s_{i0}, $i = 0, \ldots,$ ns$-$ 1, as the independent variables of the nonlinear programming problem. Thus the dimension of the problem is reduced to np + ns × ny × nd.

We decompose the set of collocation coefficients into a set of explicit variables

$$z_{\mathcal{E}} \equiv \left\{z^s_{10}, \ldots, z^s_{\mathrm{ns},0}\right\} ,$$

and a set of implicit variables

$$z_{\mathcal{I}} \equiv \left\{z^s_{11}; \ldots; z^s_{1,\mathrm{nc}}; z^s_{21}; \ldots; z^s_{\mathrm{ns},\mathrm{nc}}\right\} ,$$

where again $z^s_{ij}, j \neq 0$, are determined by the collocation conditions.

Given p and z^s_{i0}, we then solve the nonlinear system $h_1(p, z^s_{ij}) = 0$ for the implicit variables, independently on each subinterval. The dimension of the system is ny × nc × nd on a subinterval. The special structure of the problem allows us to break the nonlinear system into nd independent systems, where the dimension of each system is ny×nc. Note that the continuity conditions and the boundary conditions are constraints of the nonlinear programming problem. They are satisfied only at the final solution (p^*, z^*) of the optimization problem.

Therefore, the problem becomes:

$$\begin{cases} \min\limits_{p, z_{\mathcal{E}}} & \Phi(p, z_{\mathcal{E}}, z_{\mathcal{I}}(p, z_{\mathcal{E}})) \\ \text{s.t.} & h_2(z_{\mathcal{E}}, z_{\mathcal{I}}(p, z_{\mathcal{E}})) = 0, \\ & h_3(z_{\mathcal{E}}, z_{\mathcal{I}}(p, z_{\mathcal{E}})) = 0, \end{cases}$$

where $z_{\mathcal{I}}(p, z_{\mathcal{E}})$ solves $h_1(z_{\mathcal{I}}; p, z_{\mathcal{E}}) = 0$.

Remark 1 A simple calculation indicates that the dimension of the new nonlinear programming problem is np + ny×nd and the number of constraints is ns×ny×nd.

We now briefly discuss some implementation issues for the domain decomposition approach. The number of nonlinear systems on one subinterval is (np + ny×nd + 1)× nd. Implicit differentiation should be used to compute the first order partial derivatives of the implicit variables with respect to the explicit variables. The total number of linear systems to be solved for the implicit differentiation on one subinterval is (np + ny×nd + 1)× nd × (ny + np). This number is usually quite large. However, the linear systems are independent not only on each of the subintervals but also for each of the samples. Furthermore, for one sample on one interval, the coefficient matrices of the linear systems are all the same. Therefore, only one LU factorization is necessary. The rest are hundreds of independent triangular solvers. The second order derivatives are computed by using the finite-difference technique. The main advantage of this approach is that the resulting nonlinear systems are independent on each of the subintervals. Thus, there is no communication between the subintervals – an ideal feature for parallel computation.

Optimization

We have formulated the problem as a nonlinear programming problem by using the domain decomposition approach. We next describe a general method developed originally in [5] for solving this type of optimization problems. The optimization algorithm is based on the successive quadratic programming (SQP) with a trust region globalization. The idea is to adopt different techniques based on how close the current approximate is from the solution. If it is 'close' to the solution, we choose a step to be the solution of the quadratic program:

$$\begin{cases} \text{minimize} & \text{A Quadratic Model} \\ \text{subject to} & \text{Linearized Constraints.} \end{cases}$$

Otherwise, if it is 'far' from the solution, we choose the step to be the solution to a trust region subproblem.

The algorithm for the QP is robust. It forms the reduced Hessian and determines whether a solution exists. Note that if the reduced Hessian is not positive definite, then the QP may have infinite number of solutions or no solution at all. If a solution does exist, the algorithm will find it. The algorithm will calculate a descent direction when the QP does not have a solution. The trust region globalization technique of [4] is employed to deal with the possible lack of positive definiteness of the reduced Hessian. Thus the algorithm handles degeneracies in the linearized constraints by using eigen decomposition.

Numerical Experiment

Our test problem is based on a simplified model of the system of nonlinear Maxwell's equations that arises in modeling second harmonic generation of nonlinear optical thin films. Under some assumptions [2]. Let $Y = (y_1, y_2, y_3, y_4)^\mathsf{T}$. The model problem takes the following form:

$$Y' = (y_2, -p_1 y_1 + p_3 y_1 y_3, y_4, p_4 y_1^2 - p_2 y_3)\,,$$

with the boundary condition

$$AY(l) + BY(0) = g\,,$$

where

$$A = \begin{pmatrix} a_{11} & 1 & 0 & 0 \\ 0 & 0 & a_{21} & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\,,$$

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ a_{12} & 1 & 0 & 0 \\ 0 & 0 & a_{22} & 1 \end{pmatrix}\,,$$

and $g = (g_k, 0, 0, 0)^\mathsf{T}$.

In this over-simplified case, the inverse problem is to determine the parameters $p = (p_1, p_2, p_3, p_4)$ from the measured $Y(0)$ and $Y(\mathrm{l})$ for a given source term g_k. Since p which characterizes the physical properties of the medium is independent of g_k, it is natural to expect better reconstruction of p by performing a set of experiments or using a number of g_k. Here the constant g_k represents the intensity (power) of the incident light, which may vary with k in a given range.

In our experiment, the known constants were chosen in the following way: $a_{11} = 5$, $a_{21} = 4$, $a_{12} = -2$ and $a_{22} = -3$. We used five pairs of sample boundary data at the end points. These data were generated by using the fixed parameters $p_1 = 0.005$, $p_2 = 0.003$, $p_3 = 0.0015$ and $p_4 = 0.025$. We used five samples g_k, $k = 1, \ldots, 5$, where $g_1 = 2$, $g_2 = 5$, $g_3 = -2$, $g_4 = -3$ and $g_5 = -5$. We chose the number of collocation points nc = 4. It is obvious that in this example, np = 4 and ny = 4. From the data, we then tried to recover the parameters p. The experiment was done on a Cray J916 which is a 16 CPU shared memory computer. The results are shown in the following three tables where ns is the number of sub-intervals, nd is the number of samples, n_x is the number of variables of the NLP problem, n_h is the number of constraints, nit is the number of iterations for the algorithm to converge, error is the l_2-norm of the relative error between the estimated parameter and the exact parameter, ncpu is the number of CPUs used for solving the problem.

Optimal Design in Nonlinear Optics, Table 1
Accuracy improvement with increasing number of samples (ns = 20, ncpu = 4)

nd	n_x	n_h	nit	error	CPU time
1	84	80	8	1.36e-2	6.16
2	164	160	5	1.12e-6	12.17
3	244	240	5	8.16e-7	26.43
4	324	320	5	6.29e-7	47.76
5	404	400	5	6.12e-7	69.04

Optimal Design in Nonlinear Optics, Table 2
Convergence effect on different number of sub-intervals (nd = 4, ncpu = 4)

ns	n_x	n_h	nit	error	CPU time
8	100	96	12	1.28e-5	19.18
16	260	256	5	8.81e-6	31.97
32	516	512	5	3.65e-7	93.45

Table 1. shows the accuracy improvement when the number of samples is increased from one to five. One may observe that after some point, additional data make very little difference in terms of reconstruction of the parameters. This is largely due to the fact that the accuracy for solving nonlinear systems of equations has already been reached. In Table 2., we demonstrate the effect of increasing the number of sub-intervals, i. e., refining the grids. Finally, we show in Table 3. the total CPU time and the speed up with different number of CPUs.

Discussion

We present a general approach on parameter identification for nonlinear ODE two-point boundary value problems that arise in optimal design of nonlinear optics. The data for our parameter identification problems are only given at the boundary points. Consequently, the problem size is a multiple of the number of

Optimal Design in Nonlinear Optics, Table 3
Speed-up on different number of CPUs (n_x = 644, n_h = 640, nd = 5, ns = 32)

ncpu	Total CPU time	Speed-up
1	118.7	1
2	117.0	2.03
4	118.4	4.01
6	122.1	5.83
8	124.3	7.64
10	127.4	9.32
12	131.6	10.82
14	136.1	12.21
16	140.4	13.53

boundary data pairs. Our approach is based on ideas of nonlinear programming and domain decompositions. It generalizes [5] to a more general setting. Our preliminary numerical results indicate that the methods not only efficient on parallel machines but also effective on sequential machines. We also develop a technique to reduce the size of linear and nonlinear systems resulting from the approach.

A new research topic is to use the general approach developed in this article and [5] to solve inverse scattering and diffraction (PDE) problems. A crucial step is to develop a fast and efficient domain decomposition solver for the direct problem. Similar ideas have recently been used by Dennis and R.M. Lewis [6] for solving an inverse conductivity problem. The interested reader is referred to [1] [3] [8] for other results on related optimal design problems.

See also

- ► Bilevel Programming: Applications in Engineering
- ► Design Optimization in Computational Fluid Dynamics
- ► Interval Analysis: Application to Chemical Engineering Design Problems
- ► Multidisciplinary Design Optimization
- ► Multilevel Methods for Optimal Design
- ► Optimal Design of Composite Structures
- ► Structural Optimization: History

References

1. Achdou Y, Pironneau O (1991) Optimization of a photocell. Optimal Control Appl Meth 12:221–246
2. Bao G, Dobson D (1994) Second harmonic generation in nonlinear optical films. J Math Phys 35(4):1622–1633
3. Bao G, Dobson D, Cox JA (1995) Mathematical studies in the rigorous grating theory. J Optim Soc Amer A 12(5):1029–1042
4. Celis MR, Dennis JE Jr, Tapia RA (1985) A trust region strategy for nonlinear equality constrained optimization. Numerical Optim. In: Boggs PT, Byrd RH, Schnabel RB (eds) SIAM, Philadelphia, pp 71–82
5. Dennis JE, Li G, Williamson KA (1993) Optimization algorithms for parameter identification (preprint)
6. Dennis JE Jr, Lewis RM (1994) A comparison of nonlinear programming approaches to an elliptic inverse problem and a new domain decomposition approach. CRPC Techn Report 94468, Rice Univ
7. Dennis JE Jr, Schnabel RB (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs
8. Dobson D (1993) Optimal design of periodic antireflective structures for the Helmholtz equation. Europ J Appl Math 4:321–340
9. Keller HB (1992) Numerical methods for two-point boundary value problems. Dover, Mineola
10. Reinisch R, Neviére M (1983) Electromagnetic theory of diffraction in nonlinear optics and surface-enhanced nonlinear optical effects. Phys Rev B 28(4):1870–1885

Optimality Criteria for Multiphase Chemical Equilibrium

WILLIAM R. SMITH[1], RONALD W. MISSEN[2]
[1] University Guelph, Guelph, Canada
[2] University Toronto, Toronto, Canada

MSC2000: 49K99, 65K05, 80A10

Article Outline

Keywords
Problem Formulation
The Special Case of Phase Equilibrium (PEP)
Kuhn–Tucker (KT) Conditions
Alternative Form of KT Conditions
Global Optimality Conditions (Reaction Tangent-Plane Criterion, RTPC)
Chemical Potential Models
Methods of Obtaining
Sensitivity Analysis
Stoichiometric Restrictions
Equilibrium Constraints

Numerical Implementation
See also
References

Keywords

Optimization; Global optimization; Thermodynamics; Multiphase chemical equilibrium; Reaction tangent-plane criterion

The multiphase, multireaction *chemical equilibrium problem* (the CEP) is a *nonlinear optimization problem* in chemical thermodynamics that is of interest in many fields. Some examples of application areas include biochemistry [23], chemical engineering [32], chemistry [2], geochemistry [3], metallurgy [21], and plasma science [10]. The CEP has a number of special features arising from its unique structure [9,28]. This makes its numerical solution especially challenging, leading to difficulties associated with multiple local minima and numerical scaling. General references to the CEP include [8,27], and [33]. A recent review of aspects of the CEP has been given in [24].

The CEP may be expressed in two general ways. The one more commonly encountered is formulated as a *global optimization problem*, when the objective function is specified at a 'macroscopic level' in terms of its analytical dependence on the underlying composition variables and an appropriate set of parameters. The other is formulated at a 'molecular level', in terms of an underlying intermolecular force model, and the optimization problem solved by means of a Monte-Carlo simulation method based in statistical mechanics [30]. In this article, we consider only the former viewpoint, and aspects of its formulation and solution.

The formulation and numerical solution of the CEP require, first, an assumption about which chemical substances are to be considered; and, second, about their distribution over possible phases. The latter may take two forms. One form of the CEP assumes that all substances are represented in all possible phases, and is referred to as the *universally accessible form* (UAF). The other form prohibits at least one substance from being present in all phases, and is referred to as the *restricted accessibility form* (RAF). Distinguishing these two forms has important consequences, as discussed below.

A special case of the CEP is the *phase equilibrium problem* (PEP), which involves a set of chemical substances which can distribute themselves among two or more phases, in the absence of chemical reaction. An example of UAF-PEP is the system *n*-butanol + water + *n*-butyl acetate [5]. This type of problem is important in the chemical processing industry, and there is much current interest in the development of efficient algorithms for its numerical solution. Elementary examples of RAF-PEP are 'simple eutectic' systems and 'steam distillation', as usually modeled.

UAF and RAF forms also arise in the general CEP in which chemical reactions occur, complicating their formulation and numerical solution. Examples of RAF-CEP include multiphase problems involving condensed phases, and multiphase problems involving ionic species accessible to only a single phase.

Problem Formulation

Defining a CEP requires specifying two thermodynamic variables from the set $\{P, V, T, U, H, S, G, A\}$. $\{T, P\}$ is a common choice and is primarily considered here; this choice implies that the objective function to be optimized is the *Gibbs function*, G. Other choices can be formulated in terms of this choice (see the section 'Sensitivity Analysis' below).

One of the most general problem formulations has been given in [25], which we summarize here. We assume the following are given:

1) a substance formula matrix, $\mathbf{A}_S \in E^M \times S$, where M is the number of elements and S is the number of substances, each of which is described by a formula vector $\mathbf{a}_i$ with entries a_{ji}, which is a column of $\mathbf{A}_S$ (electric charge is considered to be an element; in the unusual situation in which ionic species are accessible to more than one phase, a 'charge row' must be included in $\mathbf{A}_S$ for each phase); we assume here that $(\mathbf{A}_S) < S$ and is usually given by M (as assumed herein);
2) an elemental-abundance vector, $\mathbf{b} \in E^M$, with entries $b_j \geq 0$ and $\mathbf{b} \neq \mathbf{0}$;
3) a set of π *chemical potential* models or *phase classes*, $\boldsymbol{\mu}^\beta\ (T, P, \mathbf{x}^\beta, \boldsymbol{\alpha}^\beta): \beta = 1, \ldots, \pi$, where:
   - $\boldsymbol{\mu}^\beta : E^{2+\widetilde{I^\beta}+\widetilde{\alpha^\beta}} E^{\widetilde{I^\beta}}$
   - $\mathbf{x}^\beta \in E^{\widetilde{I}\beta}$ is the composition (e. g., mole-fraction) vector for $\boldsymbol{\mu}^\beta$;

- $\boldsymbol{\alpha}^{\beta}$ is the vector of parameters for $\boldsymbol{\mu}^{\beta}$;
- I^{β} is the substance index set for phase class β, containing the set of subscripts of substances that are accessible to the phase class. For a UAF, $I^{\beta} = \{1, \ldots, S\}$ for all β. For an RAF, at least one substance subscript is absent from at least one $\widetilde{I}^{\beta}$ denotes the number of substances deemed to be accessible to any phase consistent with $\boldsymbol{\mu}^{\beta}$;
- $\widetilde{\alpha}^{\beta}$ is the number of chemical potential model parameters given by $\boldsymbol{\alpha}^{\beta}$;
- each phase class satisfies the Gibbs–Duhem equation, as well as the limiting law $\lim_{x_i^{\beta} \to 0} \mu_i^{\beta} = -\infty$.

To elaborate on the term 'phase class' and to relate it to the term 'phase', we note that:

3.1) every substance in the system is represented in at least one phase class; i. e., $\cup_{\beta} = 1^{\pi} I^{\beta} = \{1, \ldots, S\}$; we emphasize that all substances need not be represented in every phase class.

3.2) there may be more than one phase accessible to a given phase class β; for each such phase k arising from a particular chemical potential model $\boldsymbol{\mu}^{\beta}$, we have $\boldsymbol{\mu}^{\beta,k} \equiv \boldsymbol{\mu}^{\beta}(T, P, \mathbf{x}^{\beta,k}, \boldsymbol{\alpha}^{\beta})$, where $\mathbf{x}^{\beta,k} \in E^{\tilde{I}\beta}$ is the composition vector for the phase.

3.3) although the number of phase classes, π, is specified a priori, the number of phases, π^{β}, 'accessible' to the phase class $\boldsymbol{\mu}^{\beta}$, may not be known a priori; furthermore, π and π^{β} are distinct from the total number of phases present at equilibrium in nonzero amounts, Π, which is also not known a priori, but is a result of the equilibrium computation.

3.4) The term *species* refers to a substance in a specific phase. Since π^{β} is generally unknown, construction of a *species index set* is not always possible.

The general statement of the chemical equilibrium problem at specified T, P and $\mathbf{b}$ is given by (omitting the dependence of $\boldsymbol{\mu}^{\beta}$ on $T, P\, \boldsymbol{\alpha}^{\beta}$):

$$\min_{\widetilde{n}^{\beta,k}, \mathbf{x}^{\beta,k}} G = \sum_{\beta=1}^{\pi} \sum_{k=1}^{\pi^{\beta}} \widetilde{n}^{\beta,k} \sum_{i \in I^{\beta}} x_i^{\beta,k} \mu_i^{\beta}(\mathbf{x}^{\beta,k}) \,, \quad (1)$$

subject to

$$\widetilde{n}^{\beta,k} \geq 0; \quad \beta = 1, \ldots, \pi; \; k = 1, \ldots, \pi^{\beta} \,, \quad (2)$$

$$x_i^{\beta,k} \geq 0; i \in I^{\beta}; \; \beta = 1, \ldots, \pi; \; k = 1, \ldots, \pi^{\beta} \,, \quad (3)$$

$$\sum_{\beta=1}^{\pi} \sum_{k=1}^{\pi^{\beta}} \widetilde{n}^{\beta,k} \sum_{i \in I^{\beta}} a_{ji} x_i^{\beta,k} - b_j = 0; \quad j = 1, \ldots, M \,, \quad (4)$$

$$\sum_{i \in I^{\beta}} x_i^{\beta,k} - 1 = 0; \quad \beta = 1, \ldots, \pi; \quad k = 1, \ldots, \pi^{\beta} \,, \quad (5)$$

where $\tilde{n}^{\beta,k}$ is the total number of moles in phase β, k. In all cases we assume that at least one feasible solution exists.

The Special Case of Phase Equilibrium (PEP)

In the PEP case, the problem statement is given by (1)–(3) and (5), with the element-balance equations (4) replaced by the substance balance:

$$\sum_{\beta=1}^{\pi} \sum_{k=1}^{\pi^{\beta}} \widetilde{n}^{\beta,k} x_i^{\beta,k} - q_i = 0; \quad i = 1, \ldots, S \,, \quad (6)$$

where q_i (≥ 0) is the (constant) total number of moles of substance i in the system.

Kuhn–Tucker (KT) Conditions

We call a solution $\{\tilde{n}^{\beta,k}, \mathbf{x}^{\beta,k}, \boldsymbol{\lambda}, \boldsymbol{\theta}\}$ ($\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$ are Lagrange multipliers) of the following equations a *Kuhn–Tucker point* (KT point):

$$\widetilde{n}^{\beta,k} \left(\mu_i^{\beta}\left(\mathbf{x}^{\beta,k}\right) - \sum_{j=1}^{M} \lambda_j a_{ji} \right) = 0 \,; \quad i \in I^{\beta}; \; \beta = 1, \ldots, \pi; \; k = 1, \ldots, \pi^{\beta} \,, \quad (7)$$

$$\theta^{\beta,k} = \sum_{i \in I^{\beta}} x_i^{\beta,k} \left(\mu_i^{\beta}\left(\mathbf{x}^{\beta,k}\right) - \sum_{j=1}^{M} \lambda_j a_{ji} \right) \geq 0 \,; \quad \beta = 1, \ldots, \pi; \; k = 1, \ldots, \pi^{\beta} \,, \quad (8)$$

$$\theta^{\beta,k} \widetilde{n}^{\beta,k} = 0 \,; \beta = 1, \ldots, \pi; \; k = 1, \ldots, \pi^{\beta} \,, \quad (9)$$

$$\sum_{\beta=1}^{\pi} \sum_{k=1}^{\pi^{\beta}} \widetilde{n}^{\beta,k} \sum_{i \in I^{\beta}} a_{ji} x_i^{\beta,k} - b_j = 0; \quad j = 1, \ldots, M \,. \quad (10)$$

Alternative Form of KT Conditions

Kuhn–Tucker conditions The KT conditions given above are referred to as the *nonstoichiometric form* of the equilibrium conditions [27, pp. 46–48]. These conditions may be expressed in an alternative form, the *stoichiometric form* [27, pp. 45–46]. These arise from expressing the element abundances in terms of *component species* ([27, Chap. 2], [29]). Equations (7), (8), and (10) are replaced, respectively, by

$$\tilde{n}^{\beta,k}\left(\mu_i^{\beta}\left(\mathbf{x}^{\beta,k}\right)-\sum_{j=1}^{M}\mu_j\nu_{ij}\right)=0\,;$$
$$i\in I^{\beta};\quad \beta=1,\ldots,\pi;\quad k=1,\ldots,\pi^{\beta}\,, \tag{11}$$

$$\theta^{\beta,k}=\sum_{i\in I^{\beta}}x_i^{\beta,k}\left(\mu_i^{\beta}\left(\mathbf{x}^{\beta,k}\right)-\sum_{j=1}^{M}\mu_j\nu_{ij}\right)\geq 0\,;$$
$$\beta=1,\ldots,\pi;\quad k=1,\ldots,\pi^{\beta}\,, \tag{12}$$

$$\sum_{\beta=1}^{\pi}\sum_{k=1}^{\pi^{\beta}}\tilde{n}^{\beta,k}\sum_{i\in I^{\beta}}\nu_{ij}x_i^{\beta,k}-q_j=0;\quad j=1,\ldots,M\,, \tag{13}$$

where ν_{ij} is a stoichiometric coefficient for species i with respect to component species j, and q_j is the total amount of component j.

Global Optimality Conditions (Reaction Tangent-Plane Criterion, RTPC)

A necessary and sufficient condition for global optimality of a KT point is that the objective function is nowhere smaller than its value at that point. This may be expressed as:

$$\Delta G^{\dagger}\equiv\sum_{\beta=1}^{\pi}\sum_{k=1}^{\pi^{\beta}}\tilde{n}^{\beta,k}\sum_{i\in I^{\beta}}x_i^{\beta,k}\times\left(\mu_i^{\beta,k}(\mathbf{x}^{\beta,k})-\sum_{j=1}^{M}\lambda_j^{\dagger}a_{ji}\right)\geq 0 \tag{14}$$

over all $\{\tilde{n}^{\beta,k}, \mathbf{x}^{\beta,k}\}$ satisfying 2–(5), where $\boldsymbol{\lambda}^{\dagger}$ is the Lagrange multiplier at the KT point.

The above criterion takes different special forms for a UAF and for an RAF. If formation of a potential new phase is feasible in terms of the element-balance equations (which is always the case for a UAF, but not necessarily for an RAF), then we may consider the phase class individually, and obtain a simpler set of conditions involving only the mole fraction variables for the phase class from the inner summation in relation (14):

$$\sum_{i\in I^{\beta}}x_i^{\beta}\left(\mu_i^{\beta}(\mathbf{x})-\sum_{j=1}^{M}\lambda_j^{\dagger}a_{ji}\right)\geq 0 \tag{15}$$

over all $\{\tilde{n}^{\beta,k}, \mathbf{x}^{\beta,k}\}$ satisfying equations (2) to (6). This is called the *reaction tangent-plane criterion* (RTPC) [9,25].

For a UAF, all I^{β} contain the complete set of substances, which means, from equation (7), that each summation involving the Lagrange multipliers ${\lambda_j}^{\dagger}$ in the above gives the chemical potential ${\mu_i}^{\dagger}$ at the KT point. Criterion (15) then becomes

$$\sum_{i=1}^{S}x_i\left(\mu_i^{\beta}(\mathbf{x})-\mu_i^{\dagger}\right)\geq 0\,. \tag{16}$$

This is the *tangent-plane criterion* (TPC) for a PEP [4] (see also [15,18]).

For an RAF, criterion (15) is not equivalent to criterion (16). A simple example arises in the system containing {CO(g), CO_2(g), O_2(g), C(s)}. There are two phase classes, with index sets I^1 = {1, 2, 3} and I^2 = {4}. Consider a KT point at which only the gas phase is present, and denote the Lagrange multipliers as ${\lambda_C}^{\dagger}$, ${\lambda_O}^{\dagger}$. To test for the presence of the solid phase, criterion (15) is

$$\mu_4-\lambda_C^{\dagger}\geq 0\,. \tag{17}$$

If this criterion is satisfied, C(s) is not present at equilibrium; otherwise C(s) is present in nonzero amount at equilibrium.

Similar, but more involved, RAF problems often arise in metallurgical applications [12]. More complex RAF problems, in which a test must be made for the simultaneous presence of multiple phases, have been considered in [7].

Chemical Potential Models

The *chemical potential* of substance i in phase class β, ${\mu_i}^{\beta}$, may be expressed in terms of activity, a_i, or fugac-

ity f_i, or activity coefficient, γ_i (for ease of notation we omit the superscript β):

$$\mu_i(T,P,\mathbf{x}) = \mu_i^*(T,P^*,\mathbf{x}^*) + RT\ln a_i(T,P,\mathbf{x}), \quad (18)$$

$$\mu_i(T,P,\mathbf{x}) = \mu_i^*(T,P^*,\mathbf{x}^*) + RT\ln\left(\frac{f_i(T,P,\mathbf{x})}{f_i^*(T,P^*,\mathbf{x}^*)}\right), \quad (19)$$

$$\mu_i(T,P,\mathbf{x}) = \mu_i^*(T,P^*,\mathbf{x}^*) + RT\ln\gamma_i(T,P,\mathbf{x})x_i\ , \quad (20)$$

where R is the universal gas constant, 8.3145J mol^{-} 1 K^{-1} (see also [27, pp. 48–57, 62–73]).

Each form consists of two parts on the right side, $\mu_i^*(T, P^*, \mathbf{x}^*)$ and a logarithmic term (which has its origins in the use of the ideal-gas equation of state). $\mu_i^*(T, P^*, \mathbf{x}^*)$ is the chemical potential in a *standard state* at $(T, P^*, \mathbf{x}^*)$, as a reference state. The superscript * denotes the requirement to specify the standard-state conditions P^* and $\mathbf{x}^*$. Methods of obtaining $\mu_i^*(T, P^*, \mathbf{x}^*)$ are discussed in the next section.

The logarithmic term is obtained from an equation of state (EOS), or a specific model (e. g., a particular γ model), or a free-energy model (e. g., [17]). A simple phase-class model is the ideal-solution model

$$\mu_i(T,P,\mathbf{x}) = \mu_i^*(T,P^*,\mathbf{x}^*) + RT\ln x_i\ . \quad (21)$$

In this case, the objective function G is convex. More generally, G may be nonconvex, leading to multiple local minima.

Methods of Obtaining

μ_i^*. Analysis of (11), together with (18)–(20), reveals that the equilibrium composition is determined not by the set of individual values of $\mu_i^*(T, P^*, \mathbf{x}^*)$, but by the linear combinations $\{\Delta\, G_j^*(T)\colon j = 1, \ldots, R\}$, where R = N− M is the maximum number of linearly independent stoichiometric vectors (or chemical equations), $\boldsymbol{\nu}_j$, with elements ν_{ij} (R as defined here is not to be confused with the universal gas constant R), and N is the number of chemical species [27, p. 17]. For a single-phase system, $N = S$; for a multiphase system, $N > S$. $\Delta\, G_j^*$ is defined by

$$\Delta G_j^* \equiv -RT\ln K_j^* \equiv \sum_{i=1}^{S}\nu_{ij}\mu_i^* \quad (22)$$

and μ_i^* is the standard chemical potential of substance i in a particular (specified) phase.

In order to give rise to the same set of equilibrium solutions, any choice of $\{\mu_i^*\}$ in (7) and (8) must yield an invariant value of $\Delta\, G_j^*(T)$ for any possible chemical equation j involving the system substances. This means that [27, p. 73] for any two choices $\mu_i^*(1)$ and $\mu_i^*(2)$,

$$\sum_{i=1}^{S}\nu_{ij}(\mu_i^*(2) - \mu_i^*(1)) = 0\ . \quad (23)$$

Equation (22) may be viewed as an undetermined set of R linearly independent equations in the N μ_i^* values, in terms of a specified $\{\Delta\, G_j^*\}$. A set of μ_i^* may then be constructed by assigning arbitrary values (zero is the most convenient) to M substances with linearly independent formula vectors [6,27, pp. 214–217]. This approach *must* be followed if values of μ_i^* are required in cases for which data are only available in the form of $\Delta\, G_j^*$ (as, for example, in many situations involving biochemical and/or ionic systems).

Values of μ_i^* obtained for a particular system using equation (22) cannot normally be used for other chemical systems. A *universal* set of μ_i^* values that can be used for any chemical system requires assigning $\{\mu_j^*\}$ to a set of species containing all possible chemical elements and whose formula vectors are linearly independent, and then assigning all μ_i^* values relative to these as a datum. The minimum number of such species is equal to the number of chemical elements; thus, the simplest and most common choice is the set of elements themselves.

Even using the elements as a reference chemical state, various routes to obtaining a universal $\{\mu_i^*\}$ are possible, and one must take great care when combining data from different sources.

For a given standard-state pressure, the temperature dependence of the thermochemical properties h (enthalpy), s (entropy), and μ is given by

$$h_i^*(T) = h_i^*(T_r) + \int_{T_r}^{T} C_{Pi}(T)\,\mathrm{d}T\ ,$$

$$s_i^*(T) = s_i^*(T_r) + \int_{T_r}^{T}\frac{C_{Pi}(T)}{T}\,\mathrm{d}T\ , \quad (24)$$

$$\mu_i^*(T) = h_i^*(T) - Ts_i^*(T)\ ,$$

$$s_i^*(T) = s_i^*(T_r) + \int_{T_r}^{T}\frac{C_{Pi}(T)}{T}\,\mathrm{d}T\ , \quad (25)$$

$$\mu_i^*(T) = h_i^*(T) - Ts_i^*(T)\,, \tag{26}$$

where T_r is an arbitrary reference temperature, and C_P is molar heat capacity at constant pressure. An alternative route to $\mu_i^*(T)$ is equation (24) together with integration of

$$\frac{\partial[\frac{\mu_i^*(T)}{T}]}{\partial T} = -\frac{h_i^*(T)}{T^2}. \tag{27}$$

Other routes to $\mu_i^*(T)$ are described in [27, pp. 65–73].

The above equations require a choice of T_r, values of $h_i^*(T_r)$, one of $\{s_i^*(T_r), \mu_i^*(T_r)\}$, and values of $C_{Pi}(T)$. Common choices of T_r are 0K and 298.15K. The former is a convenient choice when the quantities are determined from statistical mechanical considerations, and the latter is convenient from an ambient temperature consideration.

Given any set of data, $\{\mu_i^*(1)(T)\}$, another set may be formed via

$$\mu_i^{*(2)}(T) = \mu_i^{*(1)}(T) - \sum_{j=1}^{M} a_{ji} c_j(T)\,, \tag{28}$$

where $\{c_j(T)\}$ is a set of arbitrary constants for the elements; such a set ensures that (23) is satisfied. The choice

$$c_j(T) = \mu_j^{*(1)}(T) \tag{29}$$

for each element in some specified (standard) state at T gives $\mu_j^*(2)(T) \equiv 0$ for each element. $\{\mu_i^*(2)(T)\}$ is then called the *set of formation values* of μ_i^*. This choice is often used, but is not computationally convenient, since its calculation at an arbitrary T requires an agreed-upon set of choices of the element standard states at each T, as well as information for the elements concerning $C_P(T)$ and possible phase transition values of enthalpy.

Some sources of μ_i^* and other thermochemical data are listed at the web site [34].

Sensitivity Analysis

In many cases, it is desired to calculate the rate of change of the optimal solution with respect to one or more than one member of the set of underlying parameters, $\{T, P, \mu_i^*, b_j\}$. The set of *sensitivity parameters* of interest are the first order derivatives $\partial n_i / \partial p_j$, and the second order derivatives $\partial^2 n_i / \partial p_j \partial p_k$, where p_j denotes a parameter. These quantities can be used in the calculation of:

1) the solution of problems with specified thermodynamic variables other than (T, P);
2) the effect of inaccurate μ_i^* data on the equilibrium composition;
3) *thermodynamic properties* of the equilibrium reacting mixture (e. g., H^{eq}, C_P^{eq}, C_V^{eq}, …);
4) the effect of changes in the specified overall (initial) composition on the equilibrium composition (e. g., buffer capacity of aqueous systems).

These topics are discussed in ([26]; [27, Chap. 8]), and an application is described in [19].

Stoichiometric Restrictions

Stoichiometric restrictions for a CEP arise when not all solutions of the element-balance equations, (4) and (5), are allowed. Stoichiometric restrictions typically arise from kinetics. The CEP can be solved by increasing the 'effective number' of component species, removing the restrictions [27, pp. 27–36, 218–219]). A simple illustration is provided by the permanganate-peroxide reaction [16].

Equilibrium Constraints

Equilibrium constraints for a CEP result from the a priori specification of some function of the equilibrium composition. Examples include fixed pH problems and solubility calculations in aqueous chemistry. Additional examples are explored in [6,20], and [1].

Numerical Implementation

Phase-class models giving rise to a convex G are important in many areas, including rocket propellant evaluation, aqueous speciation, gas-phase chemical processing, and metallurgical operations. Solution of such problems and implementation of the RTPC when necessary (in the case of pure condensed phases or of ideal-solution phases) is relatively straightforward in such cases ([27, pp. 58–59, 204–212], [28]).

In the case of nonconvex G, most workers have focused on numerical algorithms for the UAF, and in particular the PEP. See [31] for an implementation of a *homotopy continuation method* for the PEP. Global optimization algorithms have been applied in [11,13], and [22]. See [14] for the use of an algorithm based

on interval analysis. Numerical implementation of the RTPC has not yet been fully developed.

Some computer software packages to solve certain types of chemical reaction equilibrium problems are listed at the web site [34].

See also

- ► Global Optimization: Application to Phase Equilibrium Problems
- ► Global Optimization in Phase and Chemical Reaction Equilibrium

References

1. Alberty RA (1990) Equilibrium distributions of isomer groups in homologous series of hydrocarbons at a specified partial pressure of molecular hydrogen. J Chem Phys 93:5979–5982
2. Alderight L, Gans P, Ienco A, Peters D, Sabatini A, Vacca A (1999) Hyperquad simulation and speciation (HySS). A utility program for the investigation of equilibria involving soluble and partially soluble species. Coord Chem Rev 184:311–318
3. Anderson GM, Crerar DA (1993) Thermodynamics in geochemistry: The equilibrium model. Oxford University Press, Oxford
4. Baker LE, Pierce AC, Luks KD (1982) Gibbs energy analysis of phase equilibria. Soc Petrol Eng 22:731–742
5. Block U, Hegner B (1976) Development and application of a simulation model for three-phase distillation. AIChE J 22:582
6. Cheluget EL, Missen RW, Smith WR (1987) Computer calculation of ionic equilibria using species- or reaction-related thermodynamic data. J Phys Chem 91:2428–2432
7. Harvie CE, Greenberg JP, Weare JH (1987) A chemical equilibrium algorithm for highly non-ideal multiphase systems: Free energy minimization. Geochim Cosmochim Acta 51:1045–1057
8. Holub R, Vonka P (1976) The equilibrium of gaseous systems. Reidel, London
9. Jiang Y, Chapman GR, Smith WR (1996) On the geometry of chemical reaction and phase equilibria. Fluid Phase Equilib 118:77–102
10. Kovitya P (1985) Physical properties of high-temperature plasmas of hydrogen and copper in the temperature range 5000–60000 K. IEEE Trans Plasma Sci PS-13:587–594
11. Lee YP, Rangaiah GP, Luus R (1999) Phase and chemical equilibrium calculations by direct search optimization. Comput Chem Eng 23:1183–1191
12. Madeley WD, Toguri JM (1973) Computing chemical equilibrium compositions in multiphase systems. Industr Eng Chem Fundam 12:261–262
13. McDonald CM, Floudas CA (1997) GLOPEQ: A new computational tool for the phase and chemical equilibrium problem. Comput Chem Eng 21:1–23
14. McKinnon KIM, Millar C, Mongeau M (1996) Global optimization for the chemical and phase equilibrium problems using interval analysis. State of the Art in Global Optimization. In: Floudas CA, Pardalos PM (eds) Kluwer, Dordrecht, pp 365–381
15. Michelsen ML (1982) The isothermal flash problem. Part I: Stability. Fluid Phase Equilib 9:1–19
16. Missen RW, Smith WR (1990) The permanganate-peroxide reaction: Illustration of a stoichiometric restriction. J Chem Educ 67:876–877
17. Nezbeda I, Kolafa J, Smith WR (1997) Global phase diagrams of binary mixtures: Systematic basis for describing types of phase equilibrium phenomena. J Chem Soc Faraday Trans 93:3073–3080
18. Nghiem LX, Li YK, Heidemann RA (1985) Application of the tangent plane criterion to saturation pressure and temperature computations. Fluid Phase Equilib 21:39–60
19. Norval GW, Phillips MJ, Missen RW, Smith WR (1989) Applications of equilibrium sensitivity analysis to aromatization processes. Industr Eng Chem Res 28:1884–1887
20. Norval GW, Phillips MJ, Missen RW, Smith WR (1991) Constrained chemical equilibrium and incompletely speciied elemental abundance data. Canad J Chem Eng 69:1184–1192
21. Rao YK (1985) Stoichiometry and thermodynamics of metallurgical processes. Cambridge University Press, Cambridge
22. Reynolds D, Mulholland AJ, Gomatam J (1997) Calculation of chemical and phase equilibria via simulated annealing. J Math Chem 22:25–37
23. Royer CA, Smith WR, Beecham JM (1990) Analysis of binding in macromolecular complexes: A generalized approach. Analyt Biochem 191:287–294
24. Seider WD, Widagdo S (1996) Multiphase equilibria of reactive systems. Fluid Phase Equilib 123:283–303
25. Smith JV, Missen RW, Smith WR (1993) General optimality criteria for multiphase multireaction chemical equilibrium. AIChE J 39:707–710
26. Smith WR (1969) The effects of changes in problem parameters on chemical equilibrium calculations. Canad J Chem Eng 47:95–97
27. Smith WR, Missen RW (1982) Chemical reaction equilibrium analysis: Theory and algorithms. Wiley/Interscience, New York. reprinted with corrections: Krieger, 1991
28. Smith WR, Missen RW (1988) Strategies for solving the chemical equilibrium problem and an efficient microcomputer-based algorithm. Canad J Chem Eng 66:591–598
29. Smith WR, Sikaneta I, Missen RW (1998) Chemical reaction stoichiometry (CRS)
30. Smith WR, Triska B (1994) The reaction ensemble method for the computer simulation of chemical and phase equi-

libria. I. Theory and basic examples. J Chem Phys 100:3019–3027
31. Sun AC, Seider WD (1995) Homotopy-continuation method for stability analysis in the global minimization of the Gibbs free energy. Fluid Phase Equilib 103:213–249
32. Ung S, Doherty MF (1997) Theory of phase equilibria in multireaction systems. Comput Chem Eng 21:1–23
33. Van Zeggeren F, Storey SH (1970) The computation of chemical equilibria. Cambridge University Press, Cambridge
34. http://www/uic.edu/~mansoori/_Thermodynamic.Data.and.Property_html. (1999)

Optimal Planning of Offshore Oilfield Infrastructure

IGNACIO E. GROSSMANN, SUSARA VAN DEN HEEVER
Department of Chemical Engineering,
Carnegie Mellon University, Pittsburgh, USA

Article Outline

Keywords and Phrases
Introduction
Problem Statement
Optimization Model
  Objective Function
  Constraints Valid for the Whole Infrastructure
  Disjunction for each PP, WP and Well
  Logical Constraints
Solution Method
Nomenclature
  Sets and Indices
  Continuous Variables
  Boolean Variables
  Parameters
  Superscripts
References

Keywords and Phrases

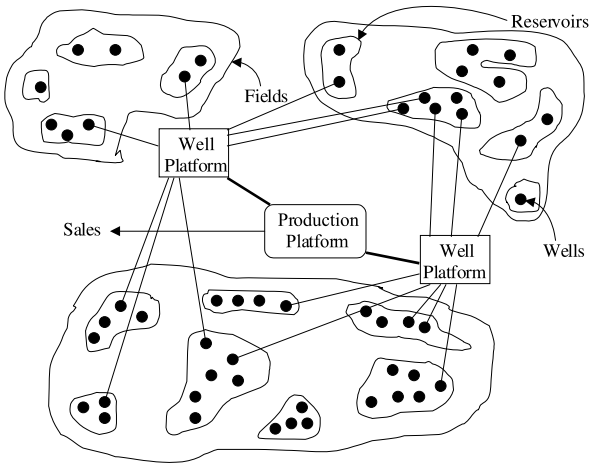
Generalized disjunctive programming; Mixed-integer programming; Offshore oilfield infrastructure; Multiperiod optimization

Introduction

Offshore oilfield infrastructure planning is a challenging problem encompassing both complex physical constraints and intricate economical specifications. An offshore oilfield infrastructure consists of Production Platforms (PP), Well Platforms (WP), wells and connecting pipelines (see Fig. 1), and is constructed for the purpose of producing oil and/or gas from one or more oilfields. Each oilfield (F) consists of a number of reservoirs (R), while each reservoir in turn contains a number of potential locations for wells (W) to be drilled.

Offshore oilfield facilities are often in operation over several decades and it is therefore important to take future conditions into consideration when designing an initial infrastructure. This can be incorporated by dividing the operating horizon into a number of time periods and allowing planning decisions in each period, while design decisions are made for the horizon as a whole. The corresponding optimization model is then run periodically with updated information on the oilfields in order to reoptimize the planning decisions of the offshore oilfield facilities.

Design decisions involve the capacities of the PPs and WPs, as well as decisions regarding *which* PPs, WPs and wells to install over the whole operating horizon. Planning decisions involve the production profiles in each period, as well as decisions regarding *when* to install PPs, WPs, and wells included in the design. Decision variables can also be grouped into discrete variables, for example those representing the installation of PPs, WPs and wells in each period, and continuous



Optimal Planning of Offshore Oilfield Infrastructure, Figure 1
Configuration of fields, well platforms and production platforms

variables, for example those representing the production profiles and pressures in each period.

Iyer and Grossmann [7] proposed a multi-period MILP model that optimizes the planning and scheduling of investment and operation decisions that include the selection of reservoirs to develop, selection of well sites and the production rates from wells at each time period. The model incorporates the nonlinear reservoir through piecewise linear approximations. Van den Heever and Grossmann [14] proposed a multi-period MINLP model for oil field infrastructure planning. As opposed to Iyer and Grossmann [7], a non-linear reservoir model was incorporated directly into the formulation. Meister, Clark and Shah [9] proposed a model for selecting the optimal information-gathering process during the exploration phase, and simultaneously optimizing the operating policies. Jonsbraten [8] presented an MILP model for optimal development of an oil field under oil price uncertainty. The author uses progressive hedging algorithm that is very similar to Lagrangean decomposition to solve the problem. Ahmed, Gorman and Bagajewicz [1] discuss the financial risk management in the planning and scheduling of offshore oil infrastructure. They introduce uncertainty, risk management and budgeting constraints to the model by Iyer and Grossmann [7] employing a sampling average algorithm to overcome the numerical difficulties and compare the results with optimum results found using upper bound risk curves. Goel and Grossmann [4,5] extended this research to gas field development planning under uncertainty. The major uncertainties were in the size and initial deliverability of the fields. The authors assumed that the uncertainty in the size and initial deliverability of the fields resolve immediately when a well platform is built on the field. Linear reservoir models were used, which provide a reasonable approximation for gas fields. The authors proposed a multistage stochastic programming model and a solution algorithm based on the problem structure. Ulstein, Nygreen and Sagli [13] presented a model for tactical planning of Norwegian petroleum production. The model maximizes the net income before taxes from the production and sale of petroleum products. Different cases with demand variations, varying quality constraints and system breakdowns are considered. The model is solved for different scenarios and solutions are compared with the base case scenario. The benefit of the model is to identify feasible ways to satisfy the demand for varying network configurations.

We describe in this chapter the deterministic model proposed by Van den Heever and Grossmann [14] which incorporates nonlinearities directly into the optimization model. Specifically, these are the reservoir pressures, gas to oil ratio, and cumulative gas produced expressed as nonlinear functions of the cumulative oil produced.

Problem Statement

The design and planning of an offshore oilfield infrastructure (refer to Fig. 1) is considered over a planning horizon of Y years, divided into T time periods (e. g. quarterly time periods). An oilfield layout consists of a number of fields, each containing one or more reservoirs and each reservoir contains one or more wellsites. After the decision has been made to produce oil from a given well site, it is drilled from a WP using drilling rigs. A network of pipelines connects the wells to the WPs and the WPs to the PPs. For our purposes, we assume that the location/allocation problem has been solved, i. e. the possible locations of the PPs and WPs, as well as the assignment of wells to WPs and WPs to PPs, are fixed. In the model, one can easily relax the assumption of fixed allocation of each well to a WP and consider that each well may be allocated to two of more WPs. However, this will clearly increase the computational effort significantly. A more practical option might be to consider allocating each well to up to two WPs. In this case, the two different allocations are treated as two choices of identical wells of which only one can be selected.

The design decisions we consider are valid for the whole planning horizon and are:

1 whether or not to include each PP, WP, and well in the infrastructure over the planning horizon
2 the capacities of the PPs and WPs

The planning decisions are made in each time period and are:

1 whether or not to install each PP and WP
2 whether or not to drill each well
3 the production profile of each well

These decisions are made under the assumption that operating conditions are constant during a given time period.

Optimization Model

The following is a complete Generalized Disjunctive Programming [6] model of the offshore oilfield infrastructure planning problem. Please refer to the list of nomenclature at the end of this chapter.

Objective Function

The objective function is to maximize the Net Present Value (NPV) which includes sales revenues, investment costs and depreciation.

$$\max \Psi = \sum_{t=1}^{T} \left\{ Rev_t - \sum_{p \in PP} \left[CI_t^p + \sum_{\pi \in WP(p)} \left\{ CI_t^{\pi,p} + \sum_{w \in W_{WP}(\pi)} CI_t^{w,\pi,p} \right\} \right] \right\} \tag{1}$$

The cost of exploration and appraisal drilling is not included in the objective, seeing that this model will be used after the results from exploration and appraisal have become available. Costs related to taxes, royalties and tariffs are not included in this model. The treatment of complex economic objectives is described in [15]. Costs that we have not included are the costs of decommissioning and well maintenance.

Constraints Valid for the Whole Infrastructure

In (2) the sales revenue in each time period is calculated from the total oil produced, which is in turn calculated in (3) as the sum of the oil produced from all production platforms. (4) calculates the amount of oil flow from each reservoir in each time period to be the sum of all oil flowrates from wells associated with that reservoir times the duration of the time period. The cumulative flow of oil from each reservoir is calculated in (5). Note that (5) is one of the linking constraints that links the time periods together and thus prevents a solution procedure where each period is solved individually. The cumulative flow of oil is used in (6) to calculate the reservoir pressure through the exponential function which is obtained by fitting a nonlinear curve to the linear interpolation data used by Iyer et al. [7].

$$Rev_t = c_{1t} x_t^{\text{total}} \tag{2}$$

$$\sum_{p \in PP} x_t^p = x_t^{\text{total}} \tag{3}$$

$$l_t^{r,f} = \Delta t \sum_{(w,\pi,p) \in W_{F,R}(f,r)} x_t^{w,\pi,p} \quad \forall r \in R(f), f \in F \tag{4}$$

$$xc_\theta^{r,f} = \sum_{t=1}^{\theta-1} l_t^{r,f} \quad \forall r \in R(f), f \in F \tag{5}$$

$$v_t^{r,f} = \gamma_{p1}^{r,f} \exp(\gamma_{p2}^{r,f} xc_t^{r,f}) \quad \forall r \in R(f), f \in F \text{ for } t = 1 \ldots T \tag{6}$$

Disjunction for each PP, WP and Well

We exploit the hierarchical structure of the oilfield to formulate a disjunctive model. The production platforms are at the highest level of the hierarchy, and the disjunction includes all constraints valid for that PP, as well as the disjunction for the next hierarchical level, i. e. for all WPs associated with that PP. In turn, the disjunction for each WP, which is located within the disjunction of a PP, contains all constraints valid for that WP, as well as the disjunctions for all wells associated with that WP. We present the disjunctions here with numbers indicating the constraints present, and follow with the explanation of the individual constraints contained in each disjunction (Fig. 2)

The outer disjunction is valid for each PP in each time period and can be interpreted as follows: If production platform p has been installed during or before period t (discrete expression $\vee_{\theta=1}^{t} z_\theta^p = \text{true}$), then all constraints in the largest bracket are applied:

$$\begin{bmatrix} z_t^p = 1 \\ CI_t^p = c_{2t}^p + c_{3t}^p e_t^p \quad (7) \\ e_t^p \le U \quad (8) \end{bmatrix} \vee \begin{bmatrix} z_t^p = 0 \\ CI_t^p, e_t^p = 0 \end{bmatrix}$$

$$x_t^p \le d_t^p \tag{9}$$

$$d_t^p = d_{t-1}^p + e_t^p \tag{10}$$

$$\sum_{\pi \in WP_{(p)}} x_t^{\pi,p} = x_t^p \tag{11}$$

$$\sum_{\pi \in WP_{(p)}} g_t^{\pi,p} = g_t^p \tag{12}$$

$$
\left[\begin{array}{c}
\bigvee_{\theta=1}^{t} z_\theta^p \\
\left[\begin{array}{c} z_t^p \\ (7),(8) \end{array}\right] \vee \left[\begin{array}{c} \neg z_t^p \\ CI_t^p, e_t^p = 0 \end{array}\right] \\
(9),(10),(11),(12) \\
\left[\begin{array}{c}
\bigvee_{\theta=1}^{t} z_\theta^{\pi,p} \\
\left[\begin{array}{c} z_t^{\pi,p} \\ (13),(14) \end{array}\right] \vee \left[\begin{array}{c} \neg z_t^{\pi,p} \\ CI_t^{\pi,p}, e_t^{\pi,p} = 0 \end{array}\right] \\
(15),(16),(17),(18),(19) \\
\left[\begin{array}{c}
\bigvee_{\theta=1}^{t} z_\theta^{w,\pi,p} \\
\left[\begin{array}{c} z_t^{w,\pi,p} \\ (20) \end{array}\right] \vee \left[\begin{array}{c} \neg z_t^{w,\pi,p} \\ CI_t^{w,\pi,p} = 0 \end{array}\right] \\
(21),(22),(23),(24),(25), \\
(26),(27),(28),(29)
\end{array}\right] \vee \left[\begin{array}{c}
\neg \bigvee_{\theta=1}^{t} z_\theta^{w,\pi,p} \\
x_t^{w,\pi,p} = 0 \\
g_t^{w,\pi,p} = 0 \\
xc_t^{w,\pi,p} = 0 \\
gc_t^{w,\pi,p} = 0
\end{array}\right] \\
\forall w \in W_{WP}(\pi)
\end{array}\right] \vee \left[\begin{array}{c}
\neg \bigvee_{\theta=1}^{t} z_\theta^{\pi,p} \\
x_t^{\pi,p} = 0 \\
g_t^{\pi,p} = 0
\end{array}\right] \\
\forall \pi \in WP(p)
\end{array}\right] \vee \left[\begin{array}{c}
\neg \bigvee_{\theta=1}^{t} z_\theta^p \\
x_t^p = 0 \\
g_t^p = 0
\end{array}\right] \\
\forall p \in PP, t \in T
$$

Optimal Planning of Offshore Oilfield Infrastructure, Figure 2

First, the smaller nested disjunction is used to calculate the discounted investment cost (including depreciation) of the production platform in each time period. This cost is calculated if production platform p is installed in period t ($z_t^p = True$), otherwise it is set to zero. (7) relates the cost as a function of the expansion capacity, which is set to zero if the production platform is not installed ($z_t^p = False$), while (8) sets an upper bound on the expansion. (9) determines the design capacity to be the maximum flow among all time periods, and this is modeled linearly by defining the expansion variable which can take a non-zero value in only one time period. (11) and (12) are mass balances calculating the oil/gas flow from the PP as the sum of the flow from all WPs associated with that PP. If the production platform has not been installed yet (discrete expression $\vee_{\theta=1}^{t} z_\theta^p$ = false), the oil/gas flows, as well as investment cost, are set to zero.

The middle disjunction is valid for all well platforms associated with production platform p and is only applied if the discrete expression $\vee_{\theta=1}^{t} z_\theta^p$ is true. This disjunction states that if well platform π has been installed before or during period t (discrete expression $\vee_{\theta=1}^{t} z_\theta^{\pi,p}$ = true), then the constraints present in that disjunction are applied:

$$
\left[\begin{array}{cc}
z_t^{\pi,p} & \\
CI_t^{\pi,p} = c_{2t}^{\pi,p} + c_{3t}^{\pi,p} e_t^{\pi,p} & (13) \\
e_t^{\pi,p} \le U & (14)
\end{array}\right] \vee \left[\begin{array}{c} \neg z_t^{\pi,p} \\ CI_t^{\pi,p}, e_t^{\pi,p} = 0 \end{array}\right]
$$

$$
x_t^{\pi,p} \le d_t^{\pi,p} \tag{15}
$$

$$
d_t^{\pi,p} = d_{t-1}^{\pi,p} + e_t^{\pi,p} \tag{16}
$$

$$
\sum_{\pi \in W_{WP}(\pi)} x_t^{w,\pi,p} = x_t^{\pi,p} \tag{17}
$$

$$
\sum_{\pi \in W_{WP}(\pi)} g_t^{w,\pi,p} = g_t^{\pi,p} \tag{18}
$$

$$
v_t^p = v_t^{\pi,p} - \alpha x_t^{\pi,p} - \beta g_t^{\pi,p} - \delta_t^{\pi,p} \tag{19}
$$

Again, the smaller nested disjunction is used to calculate the discounted investment cost (including depreciation) of the well platform in each time period. This

cost is calculated if well platform π is installed in period t ($z_t^{\pi,p} = True$), otherwise it is set to zero. (13) relates the cost as a function of the expansion capacity, which is set to zero if the well platform is not installed ($z_t^{\pi,p} = False$), while (14) sets and upper bound on the expansion. (15) and (16) determine the design capacity as described in the case of the production platform. (17) and (18) are mass balances calculating the oil/gas flow from the WP as the sum of the flow from all wells associated with that WP. (19) relates the pressure at the WP to the pressure at the PP it is associated with. The pressure at the PP is the pressure at the WP minus the pressure drop in the corresponding pipeline, which is given by the remaining terms in (19). If the production platform has not been installed yet (discrete expression $\vee_{\theta=1}^{t} z_\theta^{\pi,p}$ = false), the oil/gas flows, as well as investment cost, are set to zero.

The innermost disjunction is valid for each well w associated with well platform π, and is only included if well platform π has already been installed (discrete expression $\vee_{\theta=1}^{t} z_\theta^{\pi,p}$ = true). If well w has been drilled during or before period t (discrete expression $\vee_{\theta=1}^{t} z_\theta^{w,\pi,p}$ = true), then the following constraints are applied:

$$\begin{bmatrix} z_t^{w,\pi,p} \\ CI_t^{w,\pi,p} = c_{2t}^{w,\pi,p} \quad (20) \end{bmatrix} \vee \begin{bmatrix} \neg z_t^{w,\pi,p} \\ CI_t^{w,\pi,p} = 0 \end{bmatrix}$$

$$v_t^{\pi,p} = v_t^{w,\pi,p} - \alpha x_t^{w,\pi,p} - \beta g_t^{w,\pi,p} - \delta_t^{w,\pi,p} \tag{21}$$

$$x_t^{w,\pi,p} = \rho^{w,\pi,p}(v_t^{r,f} - v_t^{w,\pi,p}) \tag{22}$$

$$g_t^{w,\pi,p} \le x_t^{w,\pi,p} GOR_{\max} \tag{23}$$

$$x_t^{w,\pi,p} \le \rho^{w,\pi,p} P_{\max} \tag{24}$$

$$xc_t^{w,\pi,p} = \sum_{\theta=1}^{t-1} x_t^{w,\pi,p} \Delta t \tag{25}$$

$$gc_t^{w,\pi,p} = \sum_{\theta=1}^{t-1} x_t^{w,\pi,p} \Delta t \tag{26}$$

$$x_t^{w,\pi,p} \ge x_{t+1}^{w,\pi,p} \tag{27}$$

$$gc_t^{w,\pi,p} = \gamma_{g1}^{r,f} + \gamma_{g2}^{r,f} xc_t^{w,\pi,p} + \gamma_{g3}^{r,f} (xc_t^{w,\pi,p})^2 \quad \forall (w,\pi,p) \in W_{F,R}(f,r) \tag{28}$$

$$GOR_t^{w,\pi,p} = \gamma_{gor1}^{r,f} + \gamma_{gor2}^{r,f} xc_t^{w,\pi,p} + \gamma_{gor3}^{r,f} (xc_t^{w,\pi,p})^2 \quad \forall (w,\pi,p) \in W_{F,R}(f,r) \tag{29}$$

The smaller nested disjunction is used to calculate the discounted investment cost (including depreciation) of the well in each time period. This cost is calculated in (20) if well w is drilled in period t ($z_t^{w,\pi,p}$ = True), otherwise it is set to zero. (21) relates the pressure at the well to the pressure at the WP it is associated with. The pressure at the WP is the pressure at the well minus the pressure drop in the corresponding pipeline, which is given by the remaining terms. (22) states that the oil flowrate equals the productivity index times the pressure differential between reservoir and well bore. (23) restricts the gas flowrate to be the oil flow times the GOR, while (24) restricts the maximum oil flow to equal the productivity index times the maximum allowable pressure drop. The productivity index and the reservoir pressure determine the oil production rate from a well in a given time period. The well is usually capped when the GOR (gas to oil ratio) exceeds a certain threshold limit or when the pressure of the reservoir is lower than a minimum pressure. (25) and (26) calculate the cumulative flow to be the sum of flows over all periods up to the current one. Note that (25) and (26) are linking constraints that link the time periods together and prevent a solution procedure where every time period is solved separately. (27) denotes a specification by the oil company which restricts the flow profile to be non-increasing. While this company does not require a lower bound on the quantity of oil flowing through a pipeline, one could consider adding such a lower bound in the form of a threshold constraint where the flow is either zero or above some minimum, in order to address concerns that the pipeline may seize up if the flow rate were to drop below a certain level. The linear interpolation to calculate cumulative gas and GOR as functions of cumulative oil, are replaced by the nonlinear constraints (28) and (29). These quadratic equations are obtained from a curve fit of the linear interpolation data from Iyer et al. [7]. If the well has not been drilled yet (discrete expression $\vee_{\theta=1}^{t} z_\theta^{w,\pi,p}$ = false), the oil/gas flows, cu-

mulative flows, as well as investment cost, are set to zero.

Logical Constraints

These represent logical relationships between the discrete decisions. (30)–(32) specify that each well, WP and PP can be drilled/installed in only one period. (33) states that if a WP has not been installed by t, then any well w associated with that WP cannot be drilled in t. Likewise, (34) states that if a PP has not been installed by period t, then any WP associated with that PP cannot be installed in t. The restriction that only M_w wells can be drilled in any given time period, is given by (35).

$$\bigvee_{t=1}^{T} z_\theta^{w,\pi,p} \quad \forall w \in W_{WP}(\pi), \pi \in WP(p), p \in PP \tag{30}$$

$$\bigvee_{t=1}^{T} z_\theta^{\pi,p} \quad \forall \pi \in WP(p), p \in PP \tag{31}$$

$$\bigvee_{t=1}^{T} z_\theta^{p} \quad \forall p \in PP \tag{32}$$

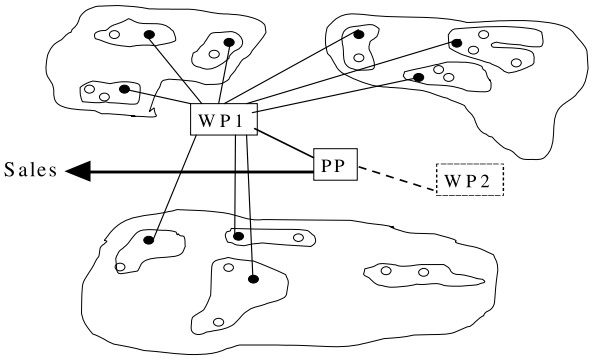
$$\neg \bigvee_{\theta=1}^{t} z_\theta^{\pi,p} \Rightarrow \neg z_t^{w,\pi,p} \quad \forall w \in W_{WP}(\pi), \pi \in WP(p), p \in PP \tag{33}$$

$$\neg \bigvee_{\theta=1}^{t} z_\theta^{p} \Rightarrow \neg z_t^{\pi,p} \forall \pi \in WP(p), p \in PP \tag{34}$$

$$\bigvee_{(w,\pi,p)} z_t^{w,\pi,p} \leq M_w \tag{35}$$

Solution Method

Van den Heever and Grossmann [14] proposed an iterative aggregation/disaggregation algorithm, where the time periods are aggregated in the design problem, and subsequently disaggregated when the planning problem is solved for the fixed infrastructure obtained from the design problem. Both of these subproblems are described with the logic-based outer approximation method [6,12]. The solution from the planning problem is used to update the aggregation scheme after each iteration. This is done through a dynamic programming subproblem which determines how time periods should be aggregated to yield an aggregate problem that resembles the disaggregate problem as close as possible. Convex envelopes are used to deal with non-convexities arising from non-linearities.



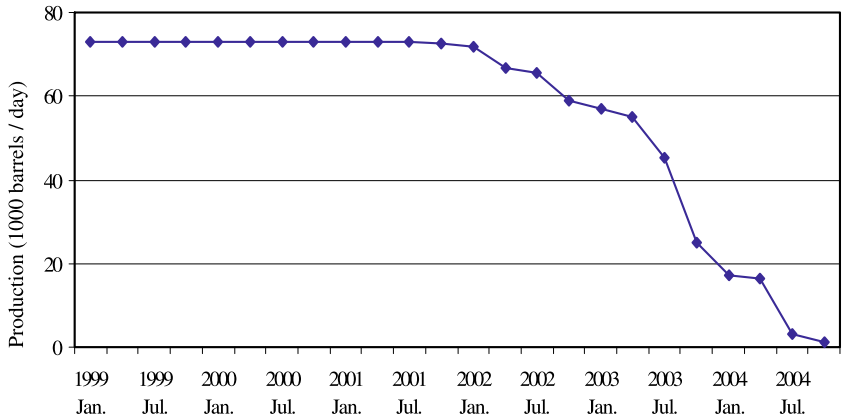
Optimal Planning of Offshore Oilfield Infrastructure, Figure 3
The final configuration

Optimal Planning of Offshore Oilfield Infrastructure, Table 1
The optimal investment plan

Item		Period invested
PP		Jan. 1999
WP1		Jan. 1999
Reservoir	Well	
2	4	Jan. 1999
3	1	Jan. 1999
5	3	Jan. 1999
4	2	Apr. 1999
7	1	Jul. 1999
6	2	Oct. 1999
1	2	Jan. 2000
9	2	Jan. 2000
10	1	Jan. 2000

Figures 3 and 4, together with Table 1 show the optimal solution obtained for the largest problem instance of 25 wells in Fig. 1 for the planning horizon of 24 periods. The final configuration is shown in Fig. 2. Note that only 9 of the potential 25 wells are drilled over the 24 periods. Of these, 3 are drilled in the first period, 1 in the second, 1 in the third, 1 in the fourth, and 3 in the fifth period as shown in Table 1.

Figure 4 shows the production profile for the whole infrastructure over the 24 time periods encompassing the six years from January 1999 up to December 2004. The net present value obtained for the profit is \$67.99 million. This final solution is found in less than 25 min by the proposed algorithm, whereas a traditional solution approach such as the OA algorithm needs more than 5 h to find the solution. Due to the short solution



Optimal Planning of Offshore Oilfield Infrastructure, Figure 4
Production profile over six years

time, the model can quickly be updated and resolved periodically as more information about the future becomes available. Also, different instances of the same problem can be solved in a relatively short time to determine the effect of different reservoir simulations on the outcome.

Nomenclature

Sets and Indices

PP	set of production platforms
p	production platform $p \in PP$
$WP(p)$	set of well platforms associated with platform p
π	well platform $\pi \in WP(p)$
F	set of fields
f	field $f \in F$
$R(f)$	set of reservoirs associated with field f
r	reservoir $r \in R(f)$
$W_{WP}(\pi)$	set of wells associated with well platform π
$W_R(r)$	set of wells associated with reservoir r
$W_{WP,R}(r, \pi)$	set of wells associated with reservoir r and well platform π
w	well $w \in W_{(.)}(.)$
t	time periods
T	aggregated time periods
T	disaggregate time periods
TA	aggregate time periods
t	disaggregate time period t ∈ T
τ	aggregate time period $\tau \in$ TA

To denote a specific well w that is associated with a specific well platform π, which is in turn associated with a specific production platform p, we use the index combination (w, π, p). Similarly, the index (π, p) applies to a specific well platform π associated with a specific production platform p. We omit superscripts in the variable definition for the sake of simplicity.

Continuous Variables

x_t	oil flow rate in period t
xc_t	cumulative oil flow up to period t
g_t	gas flow rate (volumetric) in period t
gc_t	cumulative gas flow up to period t
l_t	oil flow (mass) in period t
ϕ_t	gas-to-oil ratio (GOR) in period t
v_t	pressure in period t
δ_t	pressure drop at choke in period t
d_t	design variable in period t
e_t	design expansion variable in period t
Rev_t	sales revenue in period t
CI_t	investment cost in period t (including depreciation)
s_τ	state at the end of aggregate period τ, i. e. number of disaggregate periods available for assignment at the end of aggregate period τ
m_τ	length of aggregate period τ

Boolean Variables

z_t = true	if facility (well, WP or PP) is drilled/installed in period t

Parameters

ρ	productivity index of well
$P_{\max}$	maximum pressure drop from well bore to well head
$GOR_{\max}$	maximum GOR
m_τ	number of periods in aggregate time period τ
T_a	number of aggregate time periods, $\tau = 1..T_a$
M_w	maximum number of wells drilled in a time period
Δt	length of time period t
$\boldsymbol{U}$	upper bound parameter (defined by the respective constraint)
α	pressure drop coefficient for oil flow rate
β	pressure drop coefficient for GOR
c_{1t}	discounted revenue price coefficient for oil sales
c_{2t}	discounted fixed cost coefficient for capital investment
c_{3t}	discounted variables cost coefficient for capital investment
γ_{p1}	first coefficient for pressure vs. cumulative oil
γ_{p2}	second coefficient for pressure vs. cumulative oil
γ_{g1}	first coefficient for cumulative gas vs. cumulative oil
γ_{g2}	second coefficient for cumulative gas vs. cumulative oil
γ_{g3}	third coefficient for cumulative gas vs. cumulative oil
γ_{gor1}	first coefficient for GOR vs. cumulative oil
γ_{gor2}	second coefficient for GOR vs. cumulative oil
γ_{gor3}	third coefficient for GOR vs. cumulative oil
$\mathbf{f}_{inv,t}$	discounting factor for investment in period t
$\mathbf{f}_{dpr,t}$	discounting factor for depreciation in period t
$\mathbf{f}_{rev,t}$	discounting factor for revenue in period t
$\mathbf{I}_t$	investment costs in period t
$\mathbf{R}_t$	revenue in period t

Superscripts

(w, π, p)	variables associated with well $w \in W$, with well platform π and production platform p
(π, p)	variables associated with well platform π and production platform p
$(\boldsymbol{p})$	variables associated with production platform p
$(\boldsymbol{r})$	variables associated with reservoir r

References

1. Ahmed A, Gorman P, Bagajewicz MJ (2004) Financial risk management in offshore oil infrastructure planning and scheduling. Ind Eng Chem Res 43:3063-3072
2. Carvalho M, Pinto JM, (2006) A bilevel decomposition technique for the optimal planning of offshore platforms. Brazilian J Chem Eng 23:67-82
3. Eeg OS, Herring T (1997) Combining Linear Programming and Reservoir Simulation to Optimize Asset Value. SPE 37446 presented at the SPE Production Operations Symposium, Oklahoma City, 9–11 March 1997
4. Goel V, Grossmann IE (2004) A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. Comput Chem Eng 28(8):1409–1429
5. Goel V, Grossmann IE, El-Bakry AS, Mulkay EL (2006) A novel branch and bound algorithm for optimal development of gas fields under uncertainty in reserves. Comput Chem Eng 30:1076-1092
6. Grossmann IE (2002) Review of non-linear mixed integer and disjunctive programming techniques. Optim Eng 3:227–252
7. Iyer RR, Grossmann IE, Vasantharajan S, Cullick AS (1998) Optimal planning and scheduling offshore oil field infrastructure investment and operations. Ind Eng Chem Res 37:1380-1397
8. Jonsbraten TW (1998) Optimization models for petroleum field exploitation. PhD thesis, Norwegian School of Economics and Business Administration, Bergen
9. Meister B, Clark JMC, Shah N (1996) Optimisation of oilfield exploitation under uncertainty. Comput Chem Eng 20:S1251–S1256
10. Nygreen B, Christiansen M, Haugen K, Bjørkvoll T, Kristiansen Ø (1998) Modeling Norwegian petroleum production and transportation. Ann Oper Res 82:251
11. Sullivan JA (1982) Computer Model for Planning the Development of an Offshore Oil Field. J Petroleum Technol 34:1555
12. Turkay M, Grossmann IE (1996) Disjunctive Programming Techniques for the Optimization of Process Systems with Discontinuous Investment Costs – Multiple Size Regions. Ind Eng Chem Res 35:2611
13. Ulstein NL, Nygreen B, Sagli JR (2007) Tactical planning of offshore petroleum production. Eur J Oper Res 176:550–564
14. Van den Heever SA, Grossmann IE (2000) An iterative aggregation/disaggregation approach for the solution of a mixed integer nonlinear oilfield infrastructure planning model. Ind Eng Chem Res 39:1955–1971
15. Van den Heever SA, Grossmann IE (2001) A Lagrangean Decomposition Heuristic for the Design and Planning of Offshore Hydrocarbon Field Infrastructures with Complex Economic Objectives. Ind Eng Chem Res 40: 2857

Optimal Sensor Scheduling

GREGORY BARNETTE, MESUT YAVUZ
Research Engineering and Education Facility (REEF), University of Florida, Florida, USA

Article Outline

Background
Methods
Linear Plant Control Model
Bounded Open-Looped Covariance Propagation Model
Branch-and-Bound Search Method
Multi-site Problem Overview
Conclusions
References

Background

Sensor networks are a collection of devices designed to detect and/or measure characteristics of targets within an environment or traits of an environment itself. These measurements are then used to estimate states within the system as modeled by the designer. The design goal is to provide an optimal estimate of these states against predetermined measures of goodness. The combining of all available measurements into optimal estimates of a dynamic system is a well-studied problem and is widely implemented in variants of Kalman filters throughout many current applications (see the Analytical Science Corporation, [10]).

However; in many applications, practical limitations preclude the collection of data by all sensors at any given instance in time. In fact, systems are often limited to collecting information from a single sensor at each instance. [1,7]. For example, radar or sonar systems may be precluded from simultaneously transmitting pulses due to inter-pulse interference, or data transmission rates may exceed system bandwidth capabilities. Under these conditions, the question is no longer how to combine the measurements to obtain an optimal state estimate, but instead is in what order should the sensors be visited in order to obtain an optimal estimate. The determination of this visitation order is called the sensor scheduling problem. A variant is the single sensor, multiple-site problem, where a sensor is moved between discrete locations while maintaining an estimate of a dynamic physical attribute at each site.

Numerous optimization techniques have been used to address the sensor scheduling problem. In general, sensor scheduling has been shown to be a combinatorial optimization problem, but with certain simplifying assumptions, several successful approaches employing dynamic, linear modeling techniques have been developed. These techniques have the benefit of the extensive development of filter theory and are often incorporated directly into existing filter designs or immediate extensions to their application. In this paper, selected techniques are presented in chronological order of their development. Each is presented with a brief overview of its theoretical development followed by observations on its significant implications and application.

Methods

Linear Plant Control Model

In an early and oft cited publication, Meier et al. [8] posed the sensor selection problem as an adaptive plant control problem, in which a sequence of plant control vectors, u_k^{P} and measurement control vectors, u_k^{M} are sought to optimize plant performance. The measurement control vector selects which sensor (or sensors, in the more general case), if any, that the measurement subsystem would collect, creating a measurement matrix, $Z_k = \{z_0, z_1, \ldots, z_k\}$, where z_k is a n-dimensional column vector. Initial development is for the general discrete case. Since, in the general case, the problem is of undetermined dimensionality and unbounded in many cases of practical interest, further development is made under the assumptions that the plant and measurement subsystem are *linear* in nature and that both the system disturbance and measurement noise are *Gaussian* processes. It is also assumed that the cost function is quadratic in nature. The examination of this special case allows more definitive results to be obtained. The resulting dynamic linear system is modeled as follows.

$$\begin{aligned}
\overline{x_{k+1}} &= F_k \overline{x_k} + G_k u_k^{\mathrm{P}} + \overline{w_k} \\
\overline{z_k} &= H_k\left(u_k^{\mathrm{M}}\right)\overline{x_k} + \overline{v_k} \\
J &= E\left\{\sum_{k=0}^{N}\left[x_k^{\mathrm{T}} Q_k x_k + u_k^{\mathrm{P}^{\mathrm{T}}} R_k u_k^{\mathrm{P}} + l_k^{\mathrm{M}}\left(u_k^{\mathrm{M}}\right)\right]\right. \\
&\quad \left. + x_{N+1}^{\mathrm{T}} P_{N+1} x_{N+1}\right\}
\end{aligned}$$

where x_k is the state of the system at time k, F_k is the state transition matrix, G_k is the input matrix, w_k is zero-mean, Gaussian system noise with a covariance of Q_k, z_k is the measurement vector, H_K is the observation matrix, and v_k is zero-mean, Gaussian measurement noise with covariance R_k, which is uncorrelated with Q_k. The system state and its associated error covariance matrix are optimally propagated in accordance with the following equations.

$$u_k^{\mathrm{P}} = -K_k \hat{x}_{k\big|k}$$

where $\hat{x}_{k\big|k}$ is the optimal state estimate and

$$\begin{aligned} K_k &= \left[G_k^{\mathrm{T}} P_{k+1} G_k + R_k\right]^{-1} G_k^{\mathrm{T}} P_{k+1} F_k \\ P_k &= Q_k + F_k^{\mathrm{T}} P_{k+1} F_k + P_{k+1}^* \\ P_{k+1}^* &= F_k^{\mathrm{T}} P_{k+1} G_k \left(G_k^{\mathrm{T}} P_{k+1} G_k + R_k\right)^{-1} G_k^{\mathrm{T}} P_{k+1} F_k\,, \\ & \qquad k = 0, 1, \ldots, N\,; \end{aligned}$$

where a superscript* indicates a variable that has been updated with the current measurement and lack thereof indicates the value as propagated through the model with prior to the measurement update.

Noting that the gain matrix, K_k, and the covariance matrices, P_k and P_k^*, are independent of R_k and H_k indicates that they are also independent of the measurement control vector, u_k^{M}. This leads to a major conclusion, that the measurement control vector, u_k^{M}, can be solved separately from the plant control vector, u_k^{P}, and can in fact be determined *a priori*. This allows the computation of the measurement control matrix to be performed through the following equivalent nonlinear, deterministic control problem:

$$\text{Minimize } J^* = \sum_{k=0}^{N} \left\{ tr\left[P_{k+1}^* \hat{P}_{k|k}\right] + l_k^{\mathrm{M}}\left(u_k^{\mathrm{M}}\right)\right\}.$$

Once in this form the problem can be solved employing dynamic programming or gradient method techniques, with examples of each presented by Meier, et al. In the presence of possible local minima, the dynamic programming approach is preferred to ensure a global minimum obtained. The inclusion of a cost for taking the measurement as part of the cost function allows for the dropping of unnecessarily expensive measurements. All following techniques focus on the minimization of the error covariance, P, and assume that exactly one measurement will always be made each iteration.

Bounded Open-Looped Covariance Propagation Model

A technique for a weighted random sampling of sensors is proposed by Gupta, et al. and developed in a series of publications [4,5,6]. Here a discrete linear system is assumed and the error covariance matrix is allowed to propagate open looped (that is without measurement updates) for a predetermined number of iterations, k. Dropping the plant portion of the model used in the above model, we have the following equations.

$$\begin{aligned} \overline{x_k} &= F_{k-1}\overline{x_{k-1}} + B\overline{w_{k-1}} \\ \overline{z_k} &= H_k \overline{x_k} + \overline{v_k} \\ P_{k+1} &= F_k P_k^* F_k^{\mathrm{T}} + Q_k \end{aligned}$$

where x_k is the state of the system at time k, F_k is the state transition matrix, w_k is zero-mean, Gaussian system noise with a covariance of Q_k, B is the noise input matrix, z_k is the measurement vector, H_k is the observation matrix, and v_k is zero-mean, Gaussian measurement noise with covariance R_k, which is uncorrelated with Q_k. (Note that the lack of the $G_k\, u_k^{\mathrm{P}}$ term in this model reflects the pure sensor approach taken here vice the plant control approach taken by Meier, et al.). The error covariance matrix propagates in accordance with the following equations.

$$\begin{aligned} f_H(P) &= FPF^{\mathrm{T}} + BQB^{\mathrm{T}} \\ &\quad - FPH^{\mathrm{T}}\left(HPH^{\mathrm{T}} + R\right)^{-1} HPF^{\mathrm{T}} \\ f_H^k(P) &= f_H(f_H(f_H \ldots f_H(P))); \\ &\quad f_H \text{ is applied } k \text{ times}\,. \end{aligned}$$

While a general solution to this equation appears intractable, the concept is developed further by proving the existence of both upper and lower bounds under certain conditions. Requirements for the existence of these bounds include that P is positive semi-definite, R is positive definite, and the selection of sensor i for the jth measurement independent of all other selections. The upper bound is

$$\begin{aligned} X &= BQB^{\mathrm{T}} + AXA^{\mathrm{T}} \\ &\quad - \sum_{i=0}^{N} q_i A\left[XC_i^{\mathrm{T}}\left(R_i + C_i X C_i^{\mathrm{T}}\right) C_i X\right] A^{\mathrm{T}} \end{aligned}$$

and lower bound is

$$q_i \left|\lambda_{\max}\left(\bar{A}_j\right)\right|^2 \leq 1\,.$$

The final step in the development of this technique is to show that sensors chosen according to a Markov chain fall within these bounds, which in terms of the Markov chain parameters are

$$\begin{aligned} X_{k+1} &= \sum_{j=1}^{N} \pi_k^j X_{k+1}^j, \\ \pi_k^j X_{k+1}^j &= \sum_{i=1}^{N} f_{C_j}\left(X_k^i\right) q_{ij} \pi_k^i \\ Y_k &= \sum_{i=1}^{k} q_{jj}^{i-1}\left(\pi_{k+1-i}^j - q_{jj}\pi_{k-i}^j\right) f_{C_j}^i \\ &\quad \cdot \left(BQB^{\mathrm{T}}\right) + q_{jj}^{k-1}\pi_0^j f_{C_0}^k(P_0)\,. \end{aligned}$$

The bounds provided by this technique are loose and a designer has the opportunity to fine tune the system's performance by adjusting q_i. Selection of this parameter is known to also create a thresholding effect where use of a particular sensor is disallowed. Once implemented, this algorithm has a computational burden that is orders of magnitude less than a tree-search algorithm and finds a near-optimal solution in the steady state.

Branch-and-Bound Search Method

The next technique, developed by Feng, et al. [3], is a branch-and-bound method, also built on a discrete time, time-varying model. The sensor schedule is represented by vector u, where element $u_k = i$ means sensor i is in use at time $k \in \{1, 2, \ldots, N\}$ and U is the set of all possible u vectors. Since U is a finite set, at least one vector u will provide an optimal solution. To avoid an exhaustive search, a lower bound is developed for the propagation of the error covariance matrix and the algorithm performs a tree search for the optimal solution, pruning branches based on this lower bound. While the system model is similar to the previous model, the cost function is modified as following to reflect use of the u vector.

$$J(u) = \sum_{k=0}^{N-1} \mathrm{Tr}\left\{\Sigma(k)P_u(k)\right\}dt + c \cdot \mathrm{Tr}\left\{P_u(N)\right\}$$

where Σ is an $n \times n$ positive definite matrix-valued function with

$$\begin{aligned} \left|\Sigma_{ij}(k)\right| \leq L, k \in \{1, 2, \ldots, N\}\,, \\ i, j = 1, 2, \ldots, n\,, \end{aligned}$$

where L and c are positive constants, and

$$P_u(k) = E\left\{(x(k) - \hat{x}(k))(x(k) - \hat{x}(k))^{\mathrm{T}} \middle| \mathcal{F}_u\right\}$$

is the error covariance matrix and $\mathcal{F}_u$ is the smallest σ-algebra for which z is measurable for $u \in U$.

For two symmetric matrices P_1 and P_2 of similar dimension, the notation $P_1 \geq P_2$ was used to indicate $P_1 - P_2$ is a positive semi-definite matrix. Feng, et al. [3] proceeded to prove the lower bound on the covariance matrix exists by showing if

$$\begin{aligned} &P_{u_1}^*(k) \leq P_{u_2}^*(k) \text{ for } k \in \{1, 2, \ldots, N\}\,, \\ &\text{then } J(u_1) \leq J(u_2)\,, \end{aligned}$$

and if

$$2P_{ii} \geq \sum_{j=1}^{n} \left|P_{ij}\right|, \quad \forall i = 1, 2, \ldots, n\,,$$

then P, a $n \times n$ positive matrix, is positive semi-definite. Hence, choosing

$$\Psi(k) = \max_{i=1,2,\ldots,N} \Psi_i(k)\,,$$

where $\Psi_i(k)$ is a diagonal matrix chosen such that

$$H_i^{\mathrm{T}}(k)(D_i(k)D_i^{\mathrm{T}}(k))^{-1}C_i(k) \leq \Psi_i(k)$$

then the lower bound is given by

$$\begin{aligned} &LB(u(1), u(2), \ldots, u(i)) \\ &= \sum_{k=0}^{N-1} \mathrm{Tr}\left\{\Sigma(k)P_u^*(k)\right\}dt + c \cdot \mathrm{Tr}\left\{P_u^*(N)\right\}\,, \end{aligned}$$

subject to the system propagation equations,

$$\begin{aligned} x_k(k) &= F_{k-1}(k)\, x_{k-1} + B(k)\, w_{k-1} \\ z_k &= H_k(k)\, x_k + D(k)\, v_k \end{aligned}$$

where $k \in \{0, 1, \ldots, T-1\}$ and Q_k and R_k are uncorrelated covariance matrices for w_k and v_k, respectively.

Branching is now implemented by noting, if $\Upsilon_i(k) \leq \Upsilon_j(k)$, then for any sequences,

$\{u(1), u(2), \ldots, u(k-1), i, u(k+1), \ldots, u(T)\}$,

and $\{u(1), u(2), \ldots, u(k-1), j, u(k+1), \ldots, u(T)\}$, the solution for the covariance matrix will also be less than or equal to the solution for the second sequence, where $H_i^{\mathrm{T}}(k)(D_i(k)D_i^{\mathrm{T}}(k))^{-1}C_i(k) = \Upsilon_i(k)$.

Feng, et al. [3] present the associated algorithm and provide an illustrative example. An exhaustive search of the solution space for the example would have taken 10^{12} checks, while the branch and bound approach found the solution after branching 6842 times and examining 435 solutions.

Multi-site Problem Overview

In the multi-site single-sensor version of the sensor scheduling problem, the objective is for a single sensor to maintain an estimate of a dynamic physical attribute (e. g., position) of multiple targets. Tiwari, et al. [11] present a feasibility criterion for a single sensor to maintain a bounded estimate of an attribute at n locations. Yerrick, et al. [13] demonstrates by simulation the feasibility criterion presented in Tiwari, et al. [11] and develops a heuristic to find a good sensor motion model given the dynamics of the system under observation. Yerrick, et al. [14] provide an optimal sensor coverage solution for two sensor motion models given a model of the observed system's dynamics. The first model they study is based on i.i.d. transition probabilities among the sites. That is, the next site to be visited by the sensor is chosen according to a stationary probability distribution that is independent of the previously visited site. Their second model is more sophisticated as the moves are chosen according to a transition probability matrix.

All these mentioned works investigate probabilistic strategies for the motion of the single sensor among the sites. A deterministic approach overcomes one disadvantage of probabilistic motion: with any random motion strategy, there is nonzero probability that a particular site will not be visited at all in any finite time horizon. Yavuz and Jeffcoat [12] build an optimization model and show that it is NP-Hard. The authors also develop valid lower and upper bounds for the objective function of their model. This paper also exploits the relationships between the sensor scheduling problem and periodic scheduling problems. In particular, pinwheel scheduling and just-in-time manufacturing scheduling literatures are exploited and useful results are incorporated to sensor scheduling. The authors propose two constructive heuristic procedures to solve the sensor scheduling problem and evaluate their performance through a computational study.

Conclusions

The three approaches presented demonstrate the power of using a dynamic linear model to address the sensor scheduling problem. These approaches also integrate easily into existing linear filter algorithms and systems. As with linear filtering schemes, small system non-linearities are masked by the system and measurement noises. Should system non-linearities, prevent achieving the required level of performance needed, a filter-type of approach may still be achievable with nonlinear filters and potential approaches to the sensor scheduling problem using nonlinear filters are presented by Oshman [9] and Baras, et al. [2].

The recent contributions to multi-site single-sensor scheduling explore both probabilistic and deterministic solution approaches. The formulation of the problem is relatively new and in fact a wide range of different versions of the problem remain to be studied. Such versions should consider multiple sensors or relax one or more of the underlying assumptions; e. g., statistically independent sites, uniform transition times, no cost of movement or measurement, or time-invariant models of system dynamics and measurement.

References

1. Andersland M, Teneketzis D (1987) Decentralized sensor scheduling. In: Proceedings of the American control conference, Minneapolis, June 1987. pp 1676–1681
2. Baras J, Bensoussan A (1988) Sensor scheduling problems. In: Proceedings of the 27th conference on decision and control, Austin, Dec 1988. pp 2342–2345
3. Feng Z, Teo K, Zhao Y (2005) Branch and bound method for sensor scheduling in discrete time. J Ind Manag Optim 1(4):499–512
4. Gupta V, Chung T, Hassibi B, Murray R (2004) Sensor scheduling algorithms requiring limited time. In: Proceedings of IEEE conference on acoustics, speech, and signal processing, May 2004, Montreal, pp 825–828

5. Gupta V, Chung T, Hassibi B, Murray R (2005) On a stochastic algorithm for sensor scheduling. In: Proceedings of the 16th international federation of automatic control (IFAC) world congress, Prague, July 2005. Elsevier
6. Gupta V, Chung T, Hassibi B, Murray R (2006) On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage. Automatica 42:251–260
7. McIntyre G, Hintz K (1998) Sensor measurement scheduling: an enhanced dynamic, preemptive algorithm. Optim Eng 37(2):517–523
8. Meier L, Peschon J, Dressler R (1967) Optimal control of measurement subsystems. IEEE Trans Autom Contr AC-12(5):528–536
9. Oshman Y (1994) Optimal sensor selection strategy for discrete time state estimators. IEEE Trans Aerospace Electron Sys 30(2):307–314
10. The Analytical Sciences Corporation Staff (1974) Applied optimal estimation. MIT Press, Cambridge, MA
11. Tiwari A, Jun M, Jeffcoat D, Murray R (2005) The dynamic sensor coverage problem. In: Proceedings of the 16th international federation of automatic control (IFAC) world congress, Prague, Czech Republic, July 2005, to be published by Elsevier June 2006
12. Yavuz M, Jeffcoat D (2007) An analysis and solution of the sensor scheduling problem. In: Proceedings of the 7th cooperative control and optimization conference, Gainesville, FL, February 2007, to be published by Springer 2001
13. Yerrick N, Tiwari A, Jeffcoat D (2006) An investigation of a dynamic sensor motion strategy. In: Proceedings of the 6th cooperative control and optimization conference, Gainesville, FL, February 2006, to be published by World Scientific
14. Yerrick N, Yavuz M, Jeffcoat D (2007) Two sensor motion models for the dynamic sensor coverage problem. Mil Oper Res J 12(2):55–64

Optimal Solvent Design Approaches

Claire S. Adjiman
Department of Chemical Engineering,
Imperial College London, South Kensington Campus,
London, UK

MSC2000: 65K99

Article Outline

Introduction/Background
Formulation
General Problem Formulation
Representation of the Solvent
Methods/Applications
Design for Property Targets
Design for Process Targets
Solution Techniques
Conclusions
References

Introduction/Background

Solvents find widespread use throughout the chemical industry, from separations in bulk chemical processing to reactions, transport and separations in the fine chemicals and pharmaceutical industries. Solvents are useful for dissolving solid materials in order to enhance reaction rates, to facilitate the transport of solid materials, to provide a heat sink during highly exothermic reactions and to allow difficult separations to take place (liquid–liquid separations, absorption or extractive distillation). The choice of solvent has an impact on the economic performance of a given process, and on the quality of the products manufactured. For instance, different solvents can lead to different crystal structures being formed during crystallization; the regeneration of a solvent leaving a separation unit can be more or less energy-intensive, as in the case of chemical absorption versus physical absorption, leading to different operating costs; the amount of solvent required to process a certain amount of material may vary significantly, leading to different equipment sizes and hence capital investment.

The identification of an optimal solvent is a difficult task for several reasons. First of all, solvent design is a very complex problem in which many trade-offs must be considered. In absorption, for instance, a solvent which has a high capacity for the compound it must remove may be comparatively expensive to regenerate, and may be lost with the separated compound in larger quantities during regeneration. In reactions, a solvent which leads to a high product yield may also favor side reactions and thus result in the loss of valuable reactants. In addition, the optimal solvent is closely linked to the choice of operating conditions for a process. Thus, the decoupling of the choice of solvent and of operating conditions can lead to suboptimal solvent designs. Early methods for the computer-aided selection of solvents were based on enumeration or "generate-and-test" techniques [3,7]. However, because of the complexity of the problem, optimization

provides a natural framework for solvent design. In its most complete form, the solvent design problem is in fact an integrated plant-wide process and solvent design problem. The plant-wide aspect is required as a solvent which is optimal for a reactor may cause problems in separations (e. g., difficult recovery, poor behavior in crystallization) and vice versa. The design variables to be considered include equipment sizes and operating conditions (continuous variables), as well as the molecular structure of the solvent (integer or binary variables). The general problem structure is therefore a mixed-integer program. Usually, this problem is nonlinear in nature and, depending on the type of process model used, it may involve differential and algebraic equations.

From a practical point of view, it is not yet feasible to tackle the integrated plant-wide design problem for processes including both reaction and separation. The first publications on the use of optimization in solvent design focused on the optimization of physical property metrics such as solvent capacity and heat of vaporization [25,29]. There has since been steady progress on extensions of the problem formulation, with more direct measures of performance being used. The design of solvents for gas–liquid and liquid–liquid separations is a well-studied problem and some of the techniques developed are now used in industry. Much remains to be done on solvent design for reactions and solid–liquid separations (crystallization). For these unit operations, solvents are often chosen on the basis of intuition, similarity with other known systems and experimentation. The choices being made are thus often suboptimal. One major hurdle remains problem formulation because meaningful relations between solvent structure and solvent properties do not exist, or existing relations are very complex (e. g., Monte Carlo simulations or quantum-mechanical calculations).

In this article, the state-of-the-art in problem formulation is reviewed, and methods used to solve the problem are briefly discussed. There has been extensive and successful work on solvent selection based on "generate-and-test" approaches [3,15]. This approach is particularly well suited to the case where all decisions are integer and can therefore be enumerated but is not considered here because it does not usually rely on the formulation and solution of an optimization problem, but rather on matching certain property constraints [26]. Perspectives for further developments of optimization-based solvent design are considered at the end of this article.

Formulation

General Problem Formulation

The general formulation of the optimal solvent design problem is given by

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x},\mathbf{y}) \\ \text{subject to} \quad & h_{sp}(\mathbf{x},\mathbf{y}) = 0\,, \\ & g_{sp}(\mathbf{x},\mathbf{y}) \le 0\,, \\ & h_{cf}(\mathbf{x},\mathbf{y}) = 0\,, \\ & g_{cf}(\mathbf{x},\mathbf{y}) \le 0\,, \\ & h_{p}(\mathbf{x},\mathbf{y}) = 0\,, \\ & g_{p}(\mathbf{x},\mathbf{y}) \le 0\,, \\ & \mathbf{x} \in \mathbb{R}^{m}\,, \\ & \mathbf{y} \in \{0,1\}^{q}\,, \end{aligned} \tag{1}$$

where f is the objective function; h_{sp} is a set of structure–property equality constraints, which relate solvent molecular structure to physical properties; h_{cf} is a set of chemical feasibility and complexity equality constraints, which ensure that only meaningful solvent molecules are designed; h_{p} is a set of equality constraints describing the process, such as mass balances, energy balances and sizing constraints; g_{sp} is a set of structure–property inequality constraints; g_{cf} is a set of chemical feasibility, complexity and solvent property inequality constraints, such as limits on the solvent boiling point; g_{p} is a set of process performance constraints; $\mathbf{x}$ is an m-dimensional vector of continuous variables denoting operating conditions, physical properties or process variables; and $\mathbf{y}$ is a q-dimensional vector of binary variables describing the solvent structure.

Representation of the Solvent

Atom Groups The solvent is usually represented as a set of atom groups which are commonly used in group contribution methods [34], such as CH_3, CH_2, OH and COOH. Any given molecule can thus be represented by a vector of integer variables, $\mathbf{n}$, where the ith element corresponds to the number of groups of type i. Thus, n-butanol is represented by $n_{CH_3} = 1$, $n_{CH_2} = 3$ and

$n_{OH} = 1$. In order to allow the solution of the problem by mixed-integer programming algorithms such as the outer approximation [12,42], the integer variables are mapped onto a set of binary variables with the following constraints:

$$\sum_{k=0}^{K} 2^k y_{i,k} - n_i = 0, \qquad \forall i \in G\,, \tag{2}$$

where G denotes the set of groups, each $y_{i,k}$ denotes a binary variable and K is such that $\sum_{k=0}^{K} 2^k$ is equal to the maximum number of groups of any given type allowed in the optimal solvent. This allows the variable vector **n** to be treated as a vector of continuous variables. This approach has been used extensively in the optimal solvent design literature.

Atom Groups and Connectivity Information An alternative approach, which allows a more detailed representation of the molecule and the use of alternative structure–property relations such as second-order group contribution techniques [1,28] or connectivity indices [20], is based on a graph-theoretic representation of the compound via an adjacency matrix or similar tools. This has been proposed in the context of computer-aided molecular design by a number of authors [4,9,10,35].

Chemical Feasibility Constraints In order to ensure that the solvent designed is indeed physically meaningful, a number of constraints must be imposed to limit the combinations of binary variables. On the basis of typical boiling point constraints on solvents, it is first noted that solvents are usually acyclic, monocyclic aromatic or, more rarely, bicyclic aromatic molecules. This can be represented by three binary variables. $y_a = 1$ for an acyclic molecule, $y_b = 1$ for a bicyclic solvent molecule and $y_m = 1$ for a monocyclic molecule [29]:

$$y_a + y_b + y_m = 1\,. \tag{3}$$

Furthermore, a continuous variable, m, is defined to represent represent the type of molecule. For a monocyclic molecule, $m = 0$, for an acyclic molecule, $m = 1$ and for a bicyclic molecule, $m = -1$. This is expressed in terms of the binary variables as

$$m - (y_a - y_b) = 0\,. \tag{4}$$

The octet rule [29] ensures that the solvent molecule designed is structurally feasible and that there are no remaining free attachments in the molecule. It is based on the valency (v_i) of different structural groups:

$$\sum_{i \in G} (2 - v_i) n_i - 2m = 0\,. \tag{5}$$

To prevent any group being bonded to itself, the formation of a double bond or the formation of more than one molecule, the following constraint can be included [29]:

$$n_j(v_j - 1) + 2\,m - \sum_{i \in G} n_i \leq 0 \qquad \forall j \in G\,. \tag{6}$$

The following constraint ensures that the molecule contains at least two groups:

$$2 - \sum_{i \in G} n_i \leq 0\,. \tag{7}$$

In an aromatic molecule, the number of aromatic groups must be 6 if the molecule is monocyclic or 10 if it is bicyclic:

$$\sum_{i \in G_A} n_i - 6y_m - 10y_b = 0\,, \tag{8}$$

where G_A is the set of aromatic groups.

In a bicyclic molecule the number of aromatic carbon groups (n_{aC}) must be greater than or equal to 2:

$$2y_b - n_{aC} \leq 0\,. \tag{9}$$

Chemical complexity constraints can be imposed to describe the presence of side chains on cyclic molecules [13]. Finally, limits can be placed on the total number of groups in the molecule, on the number of specific functional groups and on particular combinations of groups. This reduces the solvent design space, making the problem more tractable and it also accounts for the fact that group contribution methods typically work well for medium-sized compounds with a few functional groups.

Design of Solvent Mixtures Several authors have expanded the size of the solvent design space by considering solvent mixtures, in which the solvents in the mixture and its composition are determined as part of the problem solution [8,38]. The presence of continuous variables to describe composition makes the use of optimization techniques, as opposed to enumeration techniques, especially suitable.

Methods/Applications

Having introduced some general concepts on the formulation of the optimal solvent design problem, we now turn our attention to the specific problems which have been tackled to date, and to the solution methods which have been employed. The applications are classified in terms of the type of objective function used. Most applications are in the area of fluid–fluid separations, but there have been a few publications on reactions and on solid–fluid separations.

Design for Property Targets

Properties from Group Contribution and Correlations Initial work in the area of solvent design was focused on identifying solvents that maximize property performance measures. In a separation process, solvent capacity and selectivity could be taken as metrics for the effectiveness of the separation and solvent loss as a metric for the difficulty of the solvent regeneration. This approach was adopted in [29], where the following function was considered as an objective:

$$\text{Capacity} \times \text{selectivity} - 100 \times \text{solvent loss}. \qquad (10)$$

This early work on optimal solvent design was applied to multicomponent gas absorption and liquid–liquid extraction using a mixed-integer nonlinear programming (MINLP) formulation, thus extending an earlier approach based on a nonlinear programming (NLP) formulation [25]. The required physical properties were calculated using group contribution techniques and similar correlations. In particular, phase equilibria was predicted using the UNIFAC approach [21]. This activity coefficient model has found many applications in solvent design thanks to its versatility.

The use of group contribution techniques and correlations allows many important selection criteria to be taken into account. Issues such as toxicity and environmental impact as deduced from octanol–water partition coefficients can be incorporated in the formulation as constraints [2]. Correlations provide a route to extending the scope of solvent design formulations. The design of solvents for crystallization can be considered by incorporating effects such as the influence of solvent choice on crystal shape [18]. Solvent choice for an extractive fermentation process can be considered by incorporating issues such as biocompatibility, inertness and ability to cause phase splitting [43]. This captures the effect of the solvent on yield in terms of its ability to promote product extraction from the mixture. The impact of solvent on reaction rate constants can be taken into account via a combination of group contribution techniques and the solvatochromic equation of [11], as shown in [13]. In all these cases, the MINLP formulation provides a single integrated framework to consider the trade-offs between the different solvent properties.

Properties from More Detailed Models Group contribution techniques are usually most accurate for medium-sized molecules which contain few functional groups [34]. While this is often appropriate for the solvent being designed, the solutes considered are often more complex and this may result in a poor representation of the solvent–solute interactions. This has repercussions on the quality of the phase equilibria predictions and can have a significant impact on the design. To address this issue, recent work has been aimed at incorporating more realistic representations of the solute and of the solvent-solute systems.

In [22,36] it was shown how continuum solvation models, in which the solvent is modeled as a continuum but the solute is modeled at the quantum-mechanical level, can be used in solvent design. Lehmann and Maranas [22] used the model of [24] to predict activity coefficients for solvent–solute pairs. This information was used to optimize the capacity and selectivity of the solvent, while taking environmental considerations into account. The use of such models poses additional difficulties in the context of optimization-based solvent design, because the evaluation of the quantum-mechanical quantities requires the solution of an optimization problem (energy minimization). This naturally leads to a bilevel formulation with a very expensive inner problem. Lehmann and Maranas [22] successfully applied their approach to a small case study, using a genetic algorithm to solve the outer solvent design problem. However, they concluded that the presence of adjustable parameters in the solvation model limited the applicability of the method because these parameters are hard to determine reliably. Sheldon et al. [36] used a different solvation model [23] and focused on minimizing the free energy of solvation. This particular problem can be formulated as a single-level problem and the authors applied this formula-

tion to a set of case studies, using the outer-approximation algorithm [12,42]. A solvent design space of over 3000 molecules was considered in each case study and the algorithm systematically converged to the optimal solution after evaluating less than 12% of the candidate molecules. Despite the high computational cost of quantum-mechanical calculations, the best solvents were identified in reasonable CPU time. However, work remains to be done to incorporate such detailed property models into more realistic solvent design problems.

Another possibility to improve the quality of property prediction is to resort to group contribution equations of state which provide a description of gas and liquid phases, in contrast to activity coefficient models which are limited to the liquid phase. This allows the representation of nonideal vapor phases, which is especially relevant for high-pressure processes. Tamouza et al. [40] and Thi et al. [41] have proposed a group contribution version of the statistical associating fluid theory (SAFT) equation of state which takes into account molecular shape and hydrogen bonding and can thus be used to model complex fluid mixtures reliably. Recently, Keskes et al. [19] have shown that a solvent for a high-pressure absorption process for CO_2 removal from natural gas can be designed based on a group contribution version of the SAFT equation of state. A limited solvent design space was used, but this opens the way for further uses of equations of state in solvent design.

Design for Process Targets

Since the mid-1990s, the formulation of the solvent design problem has become increasingly sophisticated as researchers have attempted to capture the complex trade-offs that are necessary to design an optimal process-solvent system. Two main classes of techniques have been developed: sequential approaches, in which some features of the solvent are preselected before turning to the process optimization problem, and integrated solvent and process design approaches.

Sequential Approaches In sequential approaches, a first step consists in identifying candidate solvents, or features of "good" solvents on the basis of property targets. In a second step, an optimal process is designed for the solvents selected in the first step. Such an approach was proposed by Pistikopoulos et al. [33,39] based on a multiobjective framework which accounts for economic and environmental considerations. A list of solvents is first generated on the basis of environmental and property-based criteria. The solvents chosen are then verified in different process flowsheets/designs.

In a series of papers, Papadopoulos and Linke [30,31,32] proposed an approach in which candidate solvents are first designed on the basis of property targets. A clustering algorithm is then used to extract information from the set of high-performance candidates. This information is then incorporated into a multiobjective process design problem. Both optimization problems (initial solvent design and process design) are solved using a stochastic optimization algorithm. This approach has been applied to several separation process design problems.

Integrated Solvent Design and Process Operation The simultaneous optimization of the operation of a dynamic separation process and of the solvent was considered in [14]. In this case, the process equations h_p and g_p in formulation (1) are given by a set of differential and algebraic equations, resulting in a mixed-integer dynamic optimization (MIDO) problem. The authors applied the MIDO algorithm of [5,6] and a decomposition approach based on the work of [33], in which feasibility of key property constraints is tested before solving the computationally intensive MIDO primal problem. The system considered in this work was a batch extractive distillation process, with the objective to debottleneck existing equipment.

Integrated Process and Solvent Design

Single-Objective Framework Hostrup et al. [16] proposed an MINLP formulation of the integrated process synthesis and solvent design problem. They used a series of analysis steps to explore the properties of candidate compounds and the properties of mixtures including candidate compounds. This allowed them to eliminate some candidate compounds on the basis of heuristics or on the basis of specific process constraints. Depending on the number of solvent candidates remaining after this study, a set of NLP process design problems or an MINLP is solved to identify the optimal process/solvent combination.

Multiobjective Framework Buxton et al. [8] extended the approach of [33]. Their approach is based on formulating a multiobjective problem in which economic and environmental criteria are taken into account. An approach to life-cycle analysis has been included in the formulation [17]. The large and highly nonlinear mixed-integer problem which results from this formulation is solved using a decomposition-based approach.

Solution Techniques

Several solution techniques have been brought to bear on the range of problem formulations developed so far. Many researchers have focused on formulation rather than solution technique, and have applied local optimization algorithms such as the outer-approximation [12,42] and MIDO [5,6] algorithms. Others have used decomposition-based approaches [33] or sequential approaches [30] to reduce the complexity of the problem and avoid unnecessary evaluations of large sets of equations.

The solvent design problem is inherently nonconvex: whenever phase equilibria are considered, as is always the case for separation design, highly nonlinear constraints are introduced in the problem formulation. To address this issue, there have been a few attempts at deploying and adapting global optimization techniques. Sinha et al. [37] have developed a specialized branch-and-bound algorithm. They later developed an approach based on interval analysis to tackle the same problem [38]. Wang and Achenie [44] proposed a hybrid stochastic/deterministic approach based on a combination of simulated annealing and local MINLP (outer approximation). Marcoulaki and Kokossis [27] demonstrated the use of a simulated annealing algorithm to solve solvent design problems in separation process design (liquid–liquid extraction, extractive distillation, gas absorption). Finally, genetic algorithms have been used, for instance, by Lehmann and Maranas [22].

Conclusions

The solvent design problem can naturally be framed as a mixed-integer optimization problem in which trade-offs between different properties can be considered through property targets or, more realistically, by measuring their effect on process metrics such as environmental impact or economic performance. The main difficulties in the formulation and solution arise from the complexity of the performance–structure relationships, which can be highly nonlinear, as in the case of phase-equilibrium models, or very expensive to evaluate, as in the case of detailed process models or quantum-mechanical calculations. Although much progress has been made in formulating problems for fluid–fluid separations, much remains to be done for reactive processes and solid–fluid separations. In particular, reliable property–structure relations must be developed and incorporated into the formulation of the problem. This may require new solution techniques to be developed, especially when the property–structure model requires an optimization problem to be solved. Furthermore, the nonconvexity of the problem must be addressed more generally so that reliable solutions can be guaranteed.

References

1. Abildskov J, Constantinou J, Gani R (1996) Towards the development of a second-order approximation in activity coefficient models based on group contributions. Fluid Phase Equilibria 118:1–12
2. Achenie LEK, Sinha M (2003) The design of blanket wash solvents with environmental considerations. Adv Env Res 8(2):213–227
3. Achenie LEK, Gani R, Venkatasubramanian V (eds) (2002) Computer-Aided Molecular Design: Theory and Practice. Elsevier Science, Amsterdam
4. Apostolakou A, Adjiman CS (2002) Chapter VI – Optimization methods in CAMD II. In: Achenie LEK, Gani R, Venkatasubramanian V (eds) Computer-Aided Molecular Design: Theory and Practice. Amsterdam, Elsevier Science, pp 63–94
5. Bansal V, Perkins JD, Pistikopoulos EN, Ross R, van Schijndel JMG (2000) Simultaneous design and control optimization under uncertainty. Comp Chem Eng 24:261–266
6. Bansal V, Sakizlis V, Ross R, Perkins JD, Pistikopoulos EN (2002) New algorithms for mixed-integer dynamic optimization. Comp Chem Eng 27:647–668
7. Brignole EA, Bottini S, Gani R (1986) A Strategy for the Design and Selection of Solvents for Separation Processes. Fluid Phase Equilibria 29:125–132
8. Buxton A, Livingston AG, Pistikopoulos EN (1999) Optimal design of solvent blends for environmental impact minimization. AIChE J 45:817–843
9. Camarda KV, Maranas CD (1999) Optimization in polymer design using connectivity indices. Ind Eng Chem Res 38:1884–1892

10. Churi N, Achenie LEK (1996) Novel mathematical programming model for computer aided molecular design. Ind Eng Chem Res 35:3788–3794
11. Doherty RM, Abraham MH, Harris JM, Taft RW, Kamlet MJ (1986) Linear solvation energy relationships 39. A double-difference method for estimating electrophilic solvent assistance effects in solvolysis reactions. J Organic Chem 51:4872–4875
12. Fletcher R, Leyffer S (1994) Solving mixed-integer nonlinear programs by outer approximation. Math Program 66(3):327–349
13. Folić M, Adjiman CS, Pistikopoulos EN (2007) Design of solvents for optimal reaction rate constants. AIChE J 53(5):1240–1256
14. Giovanoglou A, Barlatier J, Adjiman CS, Pistikopoulos EN, Cordiner JL (2003) Optimal solvent design for batch separation based on economic performance. AIChE J 49(12):3095–3109
15. Harper PM, Gani R (2000) A multi-step and multi-level approach for computer aided molecular design. Comp Chem Eng 24:677–683
16. Hostrup M, Harper PM, Gani R (1999) Design of environmentally benign processes: integration of solvent design and separation process synthesis. Comp Chem Eng 23(10):1395–1414
17. Hugo A, Ciumei C, Buxton A, Pistikopoulos EN (2004) Environmental impact minimization through material substitution: a multi-objective optimization approach. Green Chem 6(8):407–417
18. Karunanithi AT, Achenie LEK, Gani R (2006) A computer-aided molecular design framework for crystallization solvent design. Chem Eng Sci 61(4):1247–1260
19. Keskes E, Adjiman CS, Galindo A, Jackson G (2006) Computer-Aided Chemical Engineering, vol 21A. Elsevier Science, Amsterdam, pp 743–748
20. Kier LB, Hall LH (1976) Molecular connectivity in chemistry and drug research. Academic Press, New York
21. Larsen BL, Rasmussen P, Fredenslund A (1987) A modified UNIFAC group contribution model for prediction of phase equilibria and heats of mixing. Ind Eng Chem Res 26:2274–2286
22. Lehmann A, Maranas CD (2004) Molecular design using quantum chemical calculations for property estimation. Ind Eng Chem Res 43(13):3419–3432
23. Li J, Hawkins D, Cramer CJ, Truhlar DG (1998) Universal reaction field model based on ab initio Hartree-Fock theory. Chem Phys Lett 288:293
24. Lin ST, Sandler S (1999) Infinite dilution activity coefficients from ab initio solvation calculations. AIChE J 45(12):2606–2618
25. Macchietto S, Odele O, Omatsone O (1990) Design of optimal solvents for liquid-liquid extraction and gas absorption processes. Trans IChemE 68:429–433
26. Maranas CD (1996) Optimal Computer Aided Molecular Design: A Polymer Design Case Study. Ind Eng Chem Res 35:3788–3794
27. Marcoulaki EC Kokossis AC (2000) On the development of novel chemicals using a systematic optimisation approach. Part II. Solvent design. Chem Eng Sci 55(13):2547–2561
28. Marrero J, Gani R (2001) Group contribution based estimation of pure component properties. Fluid Phase Equilibria 183–184:183–208
29. Odele O, Macchietto S (1993) Computer aided molecular design: A novel method for optimal solvent selection. Fluid Phase Equilibria 82:47–54
30. Papadopoulos AI, Linke P (2005) A unified framework for integrated process and molecular design. Chem Eng Res Design 83(A6):674–678
31. Papadopoulos AI, Linke P (2006) Efficient integration of optimal solvent and process design using molecular clustering. Chem Eng Sci 61(19):6316–6336
32. Papadopoulos AI, Linke P (2006) Multiobjective molecular design for integrated process-solvent systems synthesis. AIChE J 52(3):1057–1070
33. Pistikopoulos EN, Stefanis SK (1998) Optimal solvent design for environmental impact minimization. Comp Chem Eng 22(6):717–733
34. Poling BE, Prausnitz JM, O'Connell J (2000) Properties of Gases and Liquids, 5th edn. McGraw-Hill, New York
35. Raman S, Maranas CD (1998) Optimization in product design with properties correlated with topological indices. Comp Chem Eng 22:747–763
36. Sheldon TJ, Folić M, Adjiman CS (2006) Solvent design using a quantum mechanical continuum solvation model. Ind Eng Chem Res 45(3):1128–1140
37. Sinha M, Achenie LEK, Ostrovsky GM (1999) Environmentally benign solvent design by global optimization. Comp Chem Eng 23(10):1381–1394
38. Sinha M, Achenie LEK, Gani R (2003) Blanket wash solvent blend design using interval analysis. Ind Eng Chem Res 42(3):516–527
39. Stefanis SK, Buxton A, Livingston AG, Pistikopoulos EN (1996) A Methodology for Environmental Impact Minimization: Solvent design and reaction path synthesis issues. Comp Chem Eng 20B:1419–1424
40. Tamouza S, Passarello JP, Tobaly P, dHemptinne JC (2004) Group contribution method with SAFT EOS applied to vapor liquid equilibria of various hydrocarbon series. Fluid Phase Equilibria 222:67–76
41. Thi TXN, Tamouza S, Tobaly P, Passarello JP, dHemptinne JC (2005) Application of group contribution SAFT equation of state (GC-SAFT) to model phase behaviour of light and heavy esters. Fluid Phase Equilibria 238(2):254–261
42. Viswanathan J, Grossmann I (1990) A combined penalty function and outer approximation method for MINLP optimization. Comp Chem Eng 14:769–782

43. Wang YP, Achenie LEK (2002) Computer aided solvent design for extractive fermentation. Fluid Phase Equilibria 20(1):1–18
44. Wang YP, Achenie LEK (2002) A hybrid global optimization approach for solvent design. Comp Chem Eng 26(10):1415–1425

Optimal Triangulations

FRANZ AURENHAMMER[1], YINFENG XU[2]
[1] Institute for Theoretical Computer Science, Graz University of Technology, Graz, Austria
[2] Xian Jiaotong University, Xian, China

MSC2000: 68Q20

Article Outline

Keywords and Phrases
Introduction
Delaunay Triangulations
Minimum Weight Triangulations
Locally Defined Subgraphs of MWT
Globally Defined Subgraphs of MWT
Some Related and Open Problems
References

Keywords and Phrases

Delaunay triangulation; Greedy triangulation; Minimum weight triangulation; β-skeleton; LMT-skeleton; Computational complexity

Introduction

A *triangulation* of a given set S of n points in the Euclidean plane is a maximal set of non-crossing straight line segments (called *edges*) which have both endpoints in S. As an equivalent definition, a triangulation of S is a partition of the convex hull of S into triangular faces whose vertex set is exactly S. Triangulations are a versatile means for partitioning and/or connecting geometric objects. They are used in many areas of engineering and scientific applications such as finite element methods, approximation theory, numerical computation, computer-aided geometric design, computational geometry, etc. Many of their applications are surveyed in [8,11,20,61].

A triangulation of S can be viewed as a planar graph whose vertex set is S and whose edge set is a subset of $S \times S$. The Eulerian relation for planar graphs implies that the number $e(S)$ of edges, and the number $t(S)$ of triangles, do not depend on the way of triangulating S. In particular

$$e(S) = 3n - 3 - h$$
$$t(S) = 2n - 2 - h$$

where h denotes the number of edges bounding the convex hull of S. The number of different triangulations of S is, however, an exponential function of n. More precisely, the number of triangulations *every* set of n points (in general position) must have is $\Omega(2.63^n)$ [57]. A general upper bound is $O(43^n)$ [68], and point sets can be constructed that exhibit $\Theta(\sqrt{72}^{\,n})$ triangulations [6]. All these bounds are very recent, and various prior bounds have been given; see the citations listed in [6,57,68].

The problem of automatically generating *optimal triangulations* for a point set S has been a subject of research since decades. Enumerating all possible triangulations and selecting an optimal one (exhaustive search) is too time-consuming even for small n. In fact, constructing optimal triangulations in polynomial time is a challenging task. This becomes more apparent as greedy methods, such as deleting candidate triangles or edges from worst to best, are doomed to fail by the NP-completeness of the following problem; see Lloyd [56]: Given a point set S and some set E of edges on S, decide whether E contains a triangulation of S.

Results on optimizing *combinatorial properties* of triangulations, such as their degree (Jansen [39]) or connectivity (Dey et al. [21] and Dillencourt [25]) are rare. Most optimization criteria for which efficient algorithms are known concern *geometric* properties of the edges and triangles.

Delaunay Triangulations

The most commonly constructed and maybe the most famous triangulation for a point set S is the *Delaunay triangulation*, $DT(S)$. See [8,27,34,45] for extensive treatments and surveys. $DT(S)$ contains – for each triple of points in S – the corresponding triangle provided its circumcircle is empty of points in S. Sib-

son [70] proved that $DT(S)$ can be constructed from any given triangulation T of S by applying any sequence of *good edge flips*. These are exchanges of diagonals in one of T's convex quadrilaterals Q such that – after the flip – the two new triangles are locally Delaunay, i. e., have circumcircles empty of vertices of Q.

Various global optimality properties of $DT(S)$ can be proved by observing that every good flip gives a *local improvement* of the respective optimality measure. Lawson [47] observed that *equiangularity* of a triangulation, which is the sorted list of its angles, increases lexicographically in this way. $DT(S)$ thus maximizes the minimum angle. *Coarseness* of a triangulation is measured by the largest circumcircle that arises for its triangles. D'Azevedo and Simpson [19] showed that $DT(S)$ minimizes coarseness in this sense, and also if smallest enclosing circles are taken rather than circumcircles. The latter property – unlike others – generalizes to higher-dimensional Delaunay triangulations; see Rajan [64]. Similarly, *fatness* may be defined as the sum of triangle inradii. Lambert [46] pointed out that $DT(S)$ maximizes fatness or, equivalently, the mean inradius. Triangular surfaces obtained from lifting $DT(S)$ to 3-space (for any given heights at triangle vertices) minimize *roughness*, which is the integral of the squared gradient; see Rippa [66]. It is also known that a variant of $DT(S)$ minimizes the minimum angle; see Eppstein [31].

The Delaunay triangulation is a special instance of *regular triangulations*, which are obtained by projecting the lower boundary part of a convex polytope in 3-space; see e. g. Edelsbrunner and Shah [28]. Regular triangulations are, thus, obviously optimal in the sense that they allow for a convex lifting surface. Some more optimality properties of $DT(S)$ can be derived from this fact; see Musin [60]. Delaunay and regular triangulations can be constrained to live in nonconvex polygons (rather than in the convex hull of the underlying point set S), see [48] and [1], respectively. Equiangularity and, with it, various other optimality properties carry over to constrained Delaunay triangulations [48].

$DT(S)$ is the geometric dual of the famous *Voronoi diagram* of a point set S and can be computed in $O(n \log n)$ time and $O(n)$ space by various different approaches; see e. g. [8,27,34]. Su and Drysdale [71] gave a thorough experimental comparison of available Delaunay triangulation algorithms.

On the negative side, $DT(S)$ fails to fulfill optimization criteria similar to those mentioned above, such as minimizing the maximum angle, or minimizing the longest edge. Edelsbrunner et al. [29,30] gave $O(n^2 \log n)$ time and $O(n^2)$ time algorithms, respectively, for computing triangulations optimal in these respects. The former algorithm is based on an *edge insertion paradigm* which is shown in Bern et al. [10] to lead – in polynomial time – to triangulations with maxmin triangle height, minmax triangle eccentricity, and minmax gradient surface, respectively.

Minimum Weight Triangulations

Most longstanding open was another optimal triangulation problem, the *minimum weight triangulation*. Here the criterion is *weight*, which is defined as the sum of all edge lengths. The complexity of computing a minimum weight triangulation, $MWT(S)$, for arbitrary planar point sets S was open since 1975 when it was mentioned in Shamos and Hoey [67]. Minimum weight triangulation is included in Garey and Johnson's [35] list of problems neither known to be NP-hard, nor known to be solvable in polynomial time. Very recently, its complexity status has been resolved; Mulzer and Rote [59] proved that minimum weight triangulation is NP-hard.

Earlier attempts to prove the minimum weight triangulation problem NP-hard have resulted in some related NP-completeness results; they are listed in [38]. Several *heuristic algorithms* have been proposed to solve this problem; see Lingas [53], Plaisted and Hong [63], and Heath and Pemmaraju [38]. None of these is known to produce a constant approximation in weight, although progress in this respect has been made later, see below.

It is well known that the Delaunay triangulation $DT(S)$ may exceed $MWT(S)$ in weight by a factor of $\Theta(n)$; see Kirkpatrick [42]. Another popular triangulation, the *greedy triangulation*, $GT(S)$, also may fail to approximate $MWT(S)$ well. $GT(S)$ is obtained by a greedy algorithm intended to yield small weight: Edges are inserted in increasing length order unless previously inserted edges are crossed and until the triangulation is completed. Several fast implementations have been proposed, e. g. by Dickerson et al. [22] who also give a brief history. Levcopoulos [49] showed

a lower bound of $\Omega(\sqrt{n})$ on the approximation factor. A matching upper bound has been given in Levcopoulos and Krznaric [50]. The same paper introduces an interesting modification of $GT(S)$ that achieves a *constant* weight approximation for $MWT(S)$ in polynomial time, though with a very large constant. Very recently, Remy and Steger [65] developed an algorithm that finds an $(1+\varepsilon)$-approximation of $MWT(S)$ in quasi-polynomial time, $n^{O(\log^8 n)}$.

For uniformly distributed point sets S, both triangulations $GT(S)$ and $DT(S)$ yield satisfactory approximations for $MWT(S)$; see e. g. Lingas [54]. In fact, $GT(S)$ seems to perform slightly better, as reported in Dickerson et al. [23]. $GT(S)$ can be constructed in $O(n)$ expected time in this case, by an algorithm in Drysdale et al. [26] or by a modification of the algorithm in Levcopoulos and Lingas [52].

The weight of a triangulation may decrease when additional points (so-called *Steiner points*) are admitted. Eppstein [32] showed that the weight of a *minimum weight Steiner triangulation* for S may be $\Omega(n)$ times smaller than the weight of $MWT(S)$. The same paper gives efficient triangulation algorithms that approximate the weight of the former within a constant factor, thus improving over previous results in Bern et al. [12]. No polynomial-time algorithms are known for the exact minimum weight Steiner triangulation problem.

Dynamic programming is a powerful tool to deal with discrete optimization problems which are decomposable in a certain sense. It leads to polynomial-time solutions for some restricted instances of the minimum weight triangulation problem. For example, if S is the vertex set of a convex polygon then $MWT(S)$ can be computed in $O(n^3)$ time and $O(n^2)$ space. The basic observation used is that – once some triangle of the triangulation has been fixed – the problem splits into subproblems (subpolygons) whose solutions can be found recursively, thereby avoiding recomputation of common subproblems. The triangulation method, first proposed by Gilbert [36] and Klincsek [44], does not really exploit convexity. It works as well for nonconvex polygons, and in fact for any interior face of a planar straight line graph. It is worth mentioning that, in the convex polygon case, $MWT(S)$ is approximated by $GT(S)$ up to a constant factor; see Levcopoulos and Lingas [51]. Anagnostou and Corneil [7] consider the case where S gives rise to a small number k of convex layers (nested convex hulls). Their dynamic programming approach works in time $O(n^{3k+1})$ and thus is polynomial for constant k. Meijer and Rappaport [58] later improved the bound to $O(n^k)$ when S is restricted to lie on k non-intersecting line segments.

The minimum weight triangulation problem can also be formulated as a *linear programming* problem. To this end, a variable x_i is assigned to each of the $\binom{n}{2}$ edges e_i defined by S. The objective is to minimize

$$\sum_{e_i} x_i |e_i|$$

subject to the constraints

$$\begin{aligned} 0 \le x_i &\le 1 \\ x_i + x_j &\le 1 \text{ for } e_i \cap e_j \neq \emptyset \\ x_i + \sum_{e_j \cap e_i \neq \emptyset} x_j &\ge 1 \,. \end{aligned}$$

The last two constraints express the property that a triangulation is a maximal set of non-crossing edges. An integer solution of this linear program yields a minimum weight triangulation: For each edge e_i, inclusion into, or exclusion from $MWT(S)$ is indicated by $x_i = 1$ or $x_i = 0$, respectively. An optimal solution need not be integer, however, and insisting on an integer solution results in an *integer programming* problem whose general version is known to be NP-complete [35]. Using a modified version of this approach, Ono et al. [62] were able to compute $MWT(S)$ for up to a hundred points.

The afore-mentioned polynomial-time results for triangulating polygonal domains (rather than point sets) give motivation for the following *subgraph approach* to compute $MWT(S)$, proposed e. g. in Xu [73] and Cheng et al. [14]: Find a (suitable) subgraph G of $MWT(S)$. If G contains k connected components, try all possibilities to add $k-1$ edges to make it a connected graph C. Complete each of these graphs C to a triangulation by optimally triangulating its faces, and select a triangulation with minimum weight, which gives $MWT(S)$. This approach, which basically is exhaustive search, can be implemented to run in $O(n^{k+2})$ time. The problem, of course, is to find candidates for G with k small.

The subgraph approach should be distinguished from the heuristic approaches in [38,53,63] we mentioned before, which also first fix some graph G for S

(for example, the minimum spanning tree) and then triangulate out its faces. The difference is that G will *not* be a subgraph of $MWT(S)$ in general, and thus does not lead to an exact solution.

Locally Defined Subgraphs of MWT

Many efforts have been put into the study of subgraphs of $MWT(S)$. Gilbert [36] pointed out that the *shortest edge* defined by S always belongs to $MWT(S)$. Another simple observation is that *unavoidable edges*, which are edges not being crossed by any other edge in $S \times S$, have to appear in any triangulation of S and thus are in $MWT(S)$. For example, all edges of the convex hull of S are unavoidable. The number of unavoidable edges does not exceed $2n - 2$, see Xu [74], and usually is very small.

Only in recent years, several more substantial subgraphs of $MWT(S)$ have been identified. One of them arises from a class of *empty neighborhood graphs* introduced by Kirkpatrick and Radke [43]. Let $p, q \in S$ and $\beta \geq 1$. The edge $\overline{pq}$ is included in the *β-skeleton*, $\beta(S)$, if the two discs of diameter $\beta|pq|$ and passing through both p and q are empty of points in S. It is not hard to see that $\beta(S)$ always is a subgraph of the Delaunay triangulation $DT(S)$. In fact, $\beta(S)$ can be constructed from $DT(S)$ in $O(n)$ time; see Lingas [55] or Jaromczyk et al. [40].

Interestingly, $\beta(S)$ is a subgraph of $MWT(S)$ for β large enough. The original bound $\beta \geq \sqrt{2}$ in Keil [41] has been improved later in Cheng and Xu [17] to $\beta > 1.1768$, which is close to the largest value $2/\sqrt{3}$ for which a counterexample is available [41]. Only for point sets S in convex position it is known that $\beta > 2/\sqrt{3}$ always implies inclusion of $\beta(S)$ in $MWT(S)$; see Hainz et al. [37]. Whereas $\beta(S)$ is a connected graph for $\beta = 1$ (known as the *Gabriel graph* of S), it may be highly disconnected for larger β-values. Cheng et al. [14] prove that – for uniformly distributed point sets S – the number of components is expected to be $\Theta(n)$.

Yang et al. [76] formulated and proved a different inclusion region: If the union of the two disks with radius $|pq|$ and centered at p and q, respectively, is empty of points in S, then $\overline{pq}$ is an edge of $MWT(S)$. That is, points p and q are mutual nearest neighbors. The skeleton generated in this way and the β-skeleton do not contain each other for $\beta > 2/\sqrt{3}$, but for smaller β-values the β-skeleton contains the former as a subgraph. Note that both graphs are defined via a symmetric and local condition. A sufficient asymmetric condition can be found in Wang et al. [72]. We refer to Eppstein [33] for a survey paper on geometric graphs.

A distinct attempt to find a sufficient local condition defines an edge e as a *light edge* [2] if there is no edge in $S \times S$ crossing e and shorter than e. Let $L(S)$ denote the graph formed by the light edges for S. $L(S)$ obviously contains all unavoidable edges, and in fact contains both Cheng and Xu's 1.1768-skeleton and the skeleton in Yang et al. [76] By construction, $L(S)$ is a subgraph of the greedy triangulation $GT(S)$, as light edges cannot be blocked by any edge previously inserted by the greedy strategy. On the other hand, $L(S)$ is no subgraph of $MWT(S)$ in general. However, if $L(S)$ happens to be a full triangulation of S, then $L(S) = MWT(S) = GT(S)$. This allows for a fast computation of $MWT(S)$ for a certain class of point sets S, using greedy triangulation algorithms.

These results are observed in Aichholzer et al. [2] as a consequence of the following matching theorem for planar triangulations, proved in Aichholzer et al. [4] and in Cheng and Xu [16]: For any two triangulations T_1 and T_2 of a fixed point set S, there is a perfect matching between the edge set of T_1 and the edge set of T_2 such that matched edges either cross or are identical. A similar matching exists for the triangle sets of T_1 and T_2, where matched triangles either overlap or are identical. The paper [2] further gives several polynomially-time computable *lower weight bounds* for $MWT(S)$, the simplest one being the weight of $L(S)$. Another bound can be stated as

$$\min_{g \in X(E)} \sum_{e \in E} |g(e)|$$

where E is any set of non-crossing edges, and $X(E)$ is the set of all matchings $g : E \to S \times S$ with the property $g(e) = e$ or $g(e)$ crosses e.

The concept of light edges gives rise to a partition of the greedy triangulation $GT(S)$ into levels $L_1, \ldots, L_k$: Let $L_1 = L(S)$ be the set of edges that are light with respect to $S \times S$, let L_2 be the set of edges that are light with respect to $(S \times S) \setminus C_1$, where C_1 is the set of edges crossing L_1, and so on. Upper bounds on the ratio in weight between $GT(S)$ and $MWT(S)$ that depend on

the number k of levels are observed in Levcopoulos and Krznaric [50] and Aichholzer et al. [5]. In particular, $GT(S)$ is a constant-factor approximation of $MWT(S)$ provided $k = O(1)$.

Globally Defined Subgraphs of MWT

We have seen that several subgraphs of $MWT(S)$ can be found from local conditions, namely, via emptiness of particular inclusion regions. The subgraphs below are defined in a global way, via intersection of triangulations.

Call a triangulation T of S *locally minimal* if every 4-sided and point-empty polygon drawn by T is optimally triangulated. That is, every convex quadrilateral contains the shorter one among its two diagonals. It is easy to see that $GT(S)$ is locally minimal and $DT(S)$, in general, is not. Let $LMT(S)$ denote the intersection of all locally minimal triangulations for S. Then $LMT(S)$ is a subgraph of $MWT(S)$, as this triangulation of course is locally minimal, too.

Whereas it is not known how to compute $LMT(S)$ in polynomial time, a surprisingly large subgraph of $LMT(S)$, the so-called *LMT-skeleton* can be identified by a simple and efficient method, proposed in Belleville et al. [9] and in Dickerson and Montague [24]: Consider some edge set $E \subset S \times S$. An edge $e \in E$ is called *redundant in* E if there is no convex quadrilateral formed by E that has e as its shorter diagonal. Edge e is called *unavoidable in* E if no other edge in E crosses e. The LMT-skeleton algorithms puts $E = S \times S$ and proceeds in several rounds. Each round first identifies all edges redundant in E and eliminates them from the set, and then includes all edges that are unavoidable in the reduced set E into the LMT-skeleton. The algorithm stops when no more edge in E can be classified as either redundant or unavoidable. The number of rounds (but not the produced LMT-skeleton) depends on the ordering in which the edges are examined. In particular, there always exists an ordering such that one round suffices; see Hainz et al. [37].

The fact that the LMT-skeleton for S, and thus $LMT(S)$, tend to be connected even for large point sets comes at a surprise. From the practical point of view, the LMT-skeleton almost always nearly triangulates S. On the other hand, a 19-point counterexample to connectedness exists [9], and for uniformly distributed points, the expected number of components is $\Theta(n)$; see Bose et al. [13]. The constant of proportionality is extremely small, however. It is interesting to note that the LMT-skeleton, and the graph of light edges $L(S)$, exhibit a similar behavior of connectedness, but do not contain each other in general. We mention further that the improved LMT-algorithm in [37], that tends to yield some additional edges of $LMT(S)$, indeed constructs $LMT(S)$ provided the original LMT-skeleton for S is connected.

The LMT-skeleton clearly can be constructed in polynomial time, and several variants have been considered in order to gain efficiency [9,15,24,37]. A powerful tool is pre-exclusion of edges before starting the LMT-algorithm, using the *exclusion region* in Das and Joseph [18]: For an edge e, consider the two triangular regions with base e and base angles $\pi/8$. If both regions contain points in S then e cannot be part of $MWT(S)$. If S is drawn from a uniform distribution, reduction to an expected linear number of candidate edges for $MWT(S)$ is achieved [22], and near-linear expected-time implementations of the LMT-algorithm exist [37,69]. The LMT-skeleton based subgraph approach enables the computation of a minimum weight triangulation for some tenthousand points in reasonable time.

Some Related and Open Problems

Let us briefly state a few open problems related to optimal triangulations.

A fast algorithm for computing the minimum weight triangulation of a simple polygon would have many applications and thus is of practical interest. Even for convex polygons, no algorithm using less than $\Theta(n^3)$ time and $\Theta(n^2)$ space is known [44]. No progress has been made since 1980 on this problem.

Minimality in weight may be relaxed to *k-optimality*, meaning that all k-sided and point-empty polygons in a triangulation are optimally triangulated. This is a generalization of local minimality which constitutes the instance $k = 4$. Whereas it is easy to compute 4-optimal triangulations (the greedy triangulation is one), no results are known on how to compute a 5-optimal triangulation in polynomial time. An algorithm based on the edge insertion paradigm is proposed in [75].

The maximal number of triangulations of a set of n points is a quantity still not well understood. The gap between the best known upper bound, $O(43^n)$ [68], and lower bound, $\Omega(\sqrt{72}^n)$ [6], is large. The common belief is that the latter function is closer to the truth.

In the last ten years, a relaxation of triangulations, so-called *pseudo-triangulations*, have been become popular, especially in computational geometry. In addition to triangles, pseudo-triangles (polygons with exactly three convex internal angles) are used as faces. Unfortunately, not much is known about optimality properties of pseudo-triangulations. Some basic properties of minimum weight pseudo-triangulations are given in the paper [3], which also shows that the greedy pseudo-triangulation always has to be a triangulation.

References

1. Aichholzer O, Aurenhammer F, Brass P, Krasser H (2003) Pseudotriangulations from surfaces and a novel type of edge flip. SIAM J Comput 32:1621–1653
2. Aichholzer O, Aurenhammer F, Cheng SW, Katoh N, Rote G, Taschwer M, Xu YF (1996) Triangulations intersect nicely. Discret Comput Geom 16:339–359
3. Aichholzer O, Aurenhammer F, Hackl T, Speckmann B (2007) On (pointed) minimun weight pseudo-triangulations. In: Proc 19th Canadian Conf Computational Geom 3:209–212
4. Aichholzer O, Aurenhammer F, Cheng S-W, Katoh N, Rote G, Taschwer M, Xu Y-F (1996) Triangulations intersect nicely. Discret Comput Geometr 16:339–359
5. Aichholzer O, Aurenhammer F, Rote G, Xu YF (1999) Constant-level greedy triangulations approximate the MWT well. J Comb Optim 2:361–369
6. Aichholzer O, Hackl T, Huemer C, Hurtado F, Krasser H, Vogtenhuber B (2007) On the number of plane geometric graphs. Graphs Combin 23:467–479
7. Anagnostou E, Corneil D (1993) Polynomial-time instances of the minimum weight triangulation problem. Comput Geom: Theory Appl 3:247–259
8. Aurenhammer F (1991) Voronoi diagrams – a survey of a fundamental geometric data structure. ACM Comput Surv 23:345–405
9. Belleville P, Keil M, McAllister M, Snoeyink J (1996) On computing edges that are in all minimum-weight triangulations. In: Proc. 12th Ann. ACM Symp. on Computational Geometry, pp V7–V8
10. Bern M, Edelsbrunner H, Eppstein D, Mitchell S, Tan TS (1993) Edge insertion for optimal triangulations. Discret Comput Geom 10:47–65
11. Bern M, Eppstein D (1995) Mesh generation and optimal triangulation. In: Du DZ, Hwang FK (eds) Computing in Euclidean Geometry, Lecture Notes Series in Computing 4. World Scientific, Singapore, pp 47–123
12. Bern M, Eppstein D, Gilbert J (1990) Provably good mesh generation. In: Proc. 31st IEEE Symp. on Foundations of Computer Science, pp 231–241
13. Bose P, Devroye L, Evans W (1996) Diamonds are not a minimum weight triangulation's best friend. Int J Comput Geometr 12(6):445–453
14. Cheng SW, Golin MJ, Tsang JCF (1995) Expected-case analysis of β-skeletons with applications to the construction of minimum-weight triangulations. In: Proc. 7th Canadian Conf. on Computational Geometry, pp 279–283
15. Cheng SW, Katoh N, Sugai M (1996) A study of the LMT-skeleton. In: Proc. Int. Symp. on Algorithms and Computation (ISAAC). Lecture Notes in Computer Science, vol 1178. Springer, pp 256–265
16. Cheng SW, Xu YF (1995) Constrained independence system and triangulations of planar point sets. In: Computing and Combinatorics, Proc. 1st Ann. Int. Conf. COCOON, Lecture Notes in Computer Science, vol 959. Springer, pp 41–50
17. Cheng SW, Xu YF (1996) Approaching the largest β-skeleton within a minimum-weight triangulation. In: Proc. 12th Ann. ACM Symp. on Computational Geometry, pp 196–203
18. Das G, Joseph G (1989) Which triangulations approximate the complete graph? Optimal Algorithms, Lecture Notes in Computer Science, vol 401. Springer, pp 168–192
19. D'Azevedo EF, Simpson RB (1989) On optimal interpolation triangle incidences. SIAM J Sci Statist Comput 10:1063–1075
20. De Floriani L (1987) Surface representation based on triangular grids. Visual Comput 3:27–50
21. Dey TK, Dillencourt MB, Ghosh SK (1994) Triangulating with high connectivity. In: Proc. 6th Canadian Conf. on Computational Geometry, pp 339–343
22. Dickerson MT, Drysdale RLS, McElfresh SA, Welzl E (1997) Fast Greedy Triangulation Algorithms. Comput Geom: Theory Appl 8:67–86
23. Dickerson MT, McElfresh SA, Montague MH (1995) New algorithms and empirical findings on minimum weight triangulation heuristics. In: Proc. 11th Ann. ACM Symp. on Computational Geometry, pp 238–247
24. Dickerson MT, Montague MH (1996) A (usually?) connected subgraph of the minimum weight triangulation. In: Proc. 12th Ann. ACM Symp. on Computational Geometry, pp 204–213
25. Dillencourt MB (1990) Toughness and Delaunay triangulations. Discret Comput Geom 5:575–601
26. Drysdale RLS, Rote G, Aichholzer O (1995) A simple linear time greedy triangulation algorithm for uniformly distributed points. Graz Univ of Technology, Graz, Austria
27. Edelsbrunner H (1987) Algorithms in Combin Geometry, EATCS Monographs on Theoretical Computer Science 10. Springer, Heidelberg

28. Edelsbrunner H, Shah NR (1996) Incremental topological flipping works for regular triangulations. Algorithmica 15:223–241
29. Edelsbrunner H, Tan TS (1993) A quadratic time algorithm for the minmax length triangulation. SIAM J Comput 22:527–551
30. Edelsbrunner H, Tan TS, Waupotitsch R (1992) An $O(N^2 \log N)$ time algorithm for the minmax angle triangulation. SIAM J Sci Statist Comput 13:994–1008
31. Eppstein D (1992) The farthest point Delaunay triangulation minimizes angles. Comput Geom: Theory Appl 1:143–148
32. Eppstein D (1994) Approximating the minimum weight steiner triangulation. Discret Comput Geom 11:163–194
33. Eppstein D (1999) Spanning trees and spanners. In: Sack JR, Urrutia J (eds) Handbook of Computational Geometry. Elsevier, pp 425–461
34. Fortune S (1995) Voronoi Diagrams and Delaunay Triangulations. In: Du DZ, Hwang FK (eds) Computing in Euclidean Geometry, Lecture Notes Series in Computing 4. World Scientific, Singapore, pp 225–265
35. Garey M, Johnson D (1979) Computers and Intractability. A Guide to the Theory of NP-completeness. W.H. Freeman
36. Gilbert PD (1979) New results in planar triangulation. M.S. thesis, Coordinated Science Laboratory, University of Illinois, Urbana
37. Hainz R, Aichholzer O, Aurenhammer F (1999) New results on MWT subgraphs. Inf Process Lett 69:215–219
38. Heath LS, Pemmaraju SV (1994) New results for the minimum weight triangulation problem. Algorithmica 12:533–552
39. Jansen K (1993) One strike against the min-max degree triangulation problem. Comput Geom: Theory Appl 3:107–120
40. Jaromczyk JW, Kowaluk M, Yao F (2007) An optimal algorithm for constructing β-skeletons in the L_p-metric. SIAM J Comput, to appear
41. Keil M (1994) Computing a subgraph of the minimum weight triangulation. Comput Geom: Theory Appl 4:13–26
42. Kirkpatrick DG (1980) A note on Delaunay and optimal triangulations. Inf Proc Lett 10:127–128
43. Kirkpatrick DG, Radke JD (1985) A framework for computational morphology In: Toussaint GT (ed) Computational Geometry, Elsevier, Amsterdam pp 217–248
44. Klincsek GT (1980) Minimal triangulations of polygonal domains. Annals Discret Math 9:127–128
45. Lambert T (1994) Empty-shape Triangulation Algorithms. Dept. of Computer Science, Univ. of Manitoba, Winnipeg, MB
46. Lambert T (1994) The Delaunay triangulation maximizes the mean inradius. In: Proc. 6th Canadian Conf. on Computational Geometry, pp 201–206
47. Lawson CL (1977) Software for C^1 surface interpolation. In: Rice JR (ed) Mathematical Software III. Academic Press, New York, pp 161–194
48. Lee DT, Lin AK (1986) Generalized Dekaunay triangulation for planar graphs. Discret Comput Geom 1:201–217
49. Levcopoulos C (1987) An $\Omega(\sqrt{(}n))$ lower bound for nonoptimality of the greedy triangulation. Inf Process Lett 25:247–251
50. Levcopoulos C, Krznaric D (1998) Quasi-greedy triangulations approximating the minimum weight triangulation. J Algorithms 27:303–338
51. Levcopoulos C, Lingas A (1987) On approximation behavior of the greedy triangulation for convex polygons. Algorithmica 2:175–193
52. Levcopoulos C, Lingas A (1992) Fast algorithms for greedy triangulation. BIT 32:280–296
53. Lingas A (1987) A new heuristic for the minimum weight triangulation. SIAM J Algebr Discret Methods 8:646–658
54. Lingas A (1986) The greedy and Delaunay triangulations are not bad in the average case. Inf Process Lett 22:25–31
55. Lingas A (1994) A linear-time construction of the relative neighborhood graph from the Delaunay triangulation. Comput Geom: Theory Appl 4:199–208
56. Lloyd EL (1977) On triangulations of a set of points in the plane. In: Proc. 18th IEEE Symp. on Foundations of Computer Science, pp 228–240
57. McCabe P, Seidel R (2004) New lower bounds for the number of straight-edge triangulations of a planar point set. In: Proc. 20nd European Workshop on Computational Geometry, pp 175–176
58. Meijer H, Rappaport D (1992) Computing the minimum weight triangulation for a set of linearly ordered points. Inf Process Lett 42:35–38
59. Mulzer W, Rote G (2006) Minimum weight triangulation is NP-hard. In: Proc. 22nd Ann. ACM Symp. on Computational Geometry, pp 1–10
60. Musin OR (1997) Properties of the Delaunay triangulation. In: Proc. 13th Ann. ACM Symp. on Computational Geometry, pp 424–426
61. Okabe A, Boots B, Sugihara K, Chiu SN (2000) Spatial Tesselations: Concepts and Applications of Voronoi Diagrams (2nd edn). Wiley
62. Ono T, Kyoda Y, Masade T, Hayase K, Shibuja T, Nakade M, Inabe M, Imai H, Imai K, Avis D (1996) A package for triangulations. In: Proc. 12th. Ann. ACM Symp. on Computational Geometry, pp V17–V18
63. Plaisted DA, Hong J (1987) A heuristic triangulation algorithm. J Algorithms 8:405–437
64. Rajan VT (1994) Optimality of the Delaunay triangulation in R^d. Discret Comput Geom 12:189–202
65. Remy J, Steger A (2006) A quasi-polynomial time approximation scheme for minimum weight triangulation. In: Proc. 38th ACM Symp. on Theory of Computing, pp 316–325

66. Rippa S (1990) Minimal roughness property of the Delaunay triangulation. Comput Aided Geom Des 7:489–497
67. Shamos MI, Hoey D (1975) Closest point problems. In: Proc. 16th IEEE Symp. on Foundations of Computer Science, pp 151–162
68. Sharir M, Welzl E (2006) Random triangulations of planar point sets. In: Proc. 22nd Ann. ACM Symposium on Computational Geometry, pp 273–281
69. Snoeyink J (1999) personal communication
70. Sibson R (1978) Locally equiangular triangulations. Comput J 21:243–245
71. Su P, Drysdale RLS (1995) A comparison of sequential Delaunay triangulation algorithms. Comput Geometr Theory Appl 7:361–385
72. Wang CA, Chin F, Xu YF (1997) A new subgraph of minimum weight triangulations. J Combin Optim 1:115–127
73. Xu YF (1992) Minimum weight triangulation problem of a planar point set. Institute of Applied Mathematics, Academia Sinica, Beijing
74. Xu YF (1996) On stable line segments in all triangulations. Appl Math JCU 11B:235–238
75. Xu YF, Dai W (2005) An algorithm for generating a 5-optimal triangulation. In: Proc. China-Japan Joint Conference on Discrete Geometry, Combinatorics, and Graph Theory, Lecture Notes in Computer Science. Springer
76. Yang BT, Xu YF, You ZY (1994) A chain decomposition algorithm for the proof of a property on minimum weight triangulations. In: Proc. 5th Int. Symp. on Algorithms and Computation (ISAAC), Lecture Notes in Computer Science, vol 834. Springer, pp 423–427

Optimization in Ad Hoc Networks

Xiaoxia Huang[1], Yuguang Fang[1],
Panos M. Pardalos[2]
[1] Department of Electrical & Computer Engineering, University of Florida, Gainesville, USA
[2] Department of Industrial & Systems Engineering, University of Florida, Gainesville, FL, USA

MSC2000: 68M12, 90B18, 90C11, 90C30

Article Outline

Introduction
Applications
  Multiconstrained Quality-of-Service Multipath Routing
  Maximum Flow Problem in Wireless Ad Hoc Networks with Directional Antennas
  Joint Mobility Control and Power-Efficient Routing in Sparse Wireless Sensor Networks
References

Introduction

Optimization techniques have found wide applications in solving various problems in ad hoc networks. Many problems in topology control, routing, maximum flow and resource allocation [23,29,33] can be formulated as optimization problems. Compared with wired networks, there are many challenges exclusive to ad hoc networks. Owing to the interdependencies across multiple layers, many factors have to be considered in a unified framework. The interdependencies complicate both problem formulation and algorithm design. For instance, interference among transmissions is a critical constraint to throughput of wireless networks; the channel is shared by all nodes in the network. Power control determines the interference range of each node. Therefore, to achieve the maximum flow in a static wireless network, power control should also be considered.

In our research, we have applied optimization-based approaches to routing, maximum flow and cross layer design problems in ad hoc networks and WSNs. The optimization problems formulated are usually nondeterministic polynomial-time (NP) hard, so the computation complexity is not affordable for resource-constrained nodes. We have devised distributed or approximation algorithms to solve the problems efficiently. In the rest of this chapter, we present some of the successful approaches that we have developed.

Applications

Multiconstrained Quality-of-Service Multipath Routing

Although small in size, sensor nodes are built with sensing, processing and computing capabilities. They report the information collected to the sink for further processing. Depending on different applications, the packets generated show diverse attributes. Different traffic has different requirements regarding packet delivery, so quality-of-service (QoS) routing is an important issue in WSNs. We have investigated both reliability and delay constraints in QoS routing. Here, reliability is defined as the packet delivery ratio. Prone to link changes and failures, sensor networks are unreliable. The empirical result from Berkeley [28] shows that the average packet loss ratio increases 5–10% per link in sensor networks. Multiconstrained routing is faced with

time complexity and/or space complexity. For wireless networks, complete and accurate state information is not available owing to the time-varying traffic and link quality. Only soft-QoS provisioning is attainable in notoriously unpredictable wireless communications. Here soft QoS is defined as guaranteeing the QoS requirements with probability. It approximates hard QoS when the probability approaches 1. It is known that finding a path subject to two or more additive constraints is NP-complete [19]. Therefore, solving the problem in a heuristic and approximate way is the only reasonable approach for resource-limited sensor nodes. Soft QoS follows naturally from the inherent random link characteristics of ad hoc networks and WSNs. Owing to the inherent difficulty of end-to-end QoS and the limited functionality of sensor nodes, some approximate methods have to be applied to deal with the computational complexity. We first formulate the end-to-end soft-QoS problem as a stochastic program. Then we propose a distributed routing algorithm based on the linear program, which is a deterministic approximation of the original end-to-end QoS routing. Our proposed routing algorithm is hop-based, so it is scalable and convenient to implement. As another favorable feature, it circumvents the formidable computational complexity of the multiconstrained path problem.

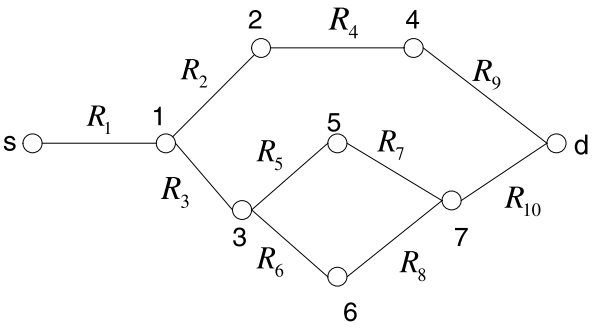
For wired networks, many papers have proposed exact or heuristic algorithms targeted at multiconstrained path or multiconstrained optimal path problems [13,18,19,26,27,42,43]. However, WSNs differ from wired networks in the limited energy, memory and computation capabilities of nodes, and link characteristics. Multipath routing has been applied to address QoS in ad hoc networks [38], but we formulate the problem in a more rigorous way and consider multiple QoS constraints.

For a given path p, the end-to-end reliability can be computed as

$$\prod_{(i,j)\in p} r_{ij}\,, \tag{1}$$

where r_{ij} is the reliability of link (i,j) on path p. If there is no single feasible path satisfying the reliability requirement, multipath routing can be used to improve the reliability. Carefully choosing a subset of existing paths, one can transfer the packet on all those paths. Although an individual path cannot achieve the performance goal, multiple paths may meet it aggregately as shown in Fig. 1. In Fig. 1, the source node assigns reliability R_1 to its next hop node 1. Neither link l_{12} nor l_{13} could satisfy this reliability requirement alone, so node 1 distributes reliability requirement R_2 to link l_{12} and R_3 to link l_{13}, so that $R_2 + R_3 \geq R_1$. The same process is performed at each intermediate node. Finally at sink node d, the three paths, $s \to 1 \to 2 \to 4 \to d$, $s \to 1 \to 3 \to 5 \to 7 \to d$ and $s \to 1 \to 3 \to 6 \to 7 \to d$, can achieve the desired reliability additively. The assembly efficiency of multiple paths is a great boon to unreliable sensor networks. Obviously, there exist many feasible combinations. To save the energy cost, the set with the minimum number of paths is chosen as the forwarding set. We argue that sending a packet on more paths induces a higher energy cost, because more data packets have to be transmitted. Using more paths introduces more contention, which degrades energy efficiency. Although some paths in the set may have more hops, it is still more energy efficient to confine packets to a few paths.

First of all, the question of how to quantify the reliability achieved by a subset of paths needs to be addressed. Then how to choose the energy-efficient path set subject to the delay constraint is our main focus. Denote d as the sink, which is assumed to be stationary. Let $P(s,d)$ denote the path set of P possible paths from a source node s to d. Each path p_j in $P(s,d)$, $j = 1, 2, \ldots, P$, is associated with delay d_j and reliability r_j. The aggregate reliability of multiple paths is approximated as the sum of the reliability of those paths. We formulate the problem as follows: $\forall p \in P(s,d)$, at



Optimization in Ad Hoc Networks, Figure 1
Reliability distribution between a source–destination pair

Optimization in Ad Hoc Networks, Table 1
Notation

Notation	Meaning
(i,j)	Link from node i to node j
$N(i)$	Neighbor set of node i
h_i	Hop count from current node i to the sink
r_{ij}	Reliability of link l_{ij}
α	Soft-QoS probability for delay
β	Soft-QoS probability for reliability
L_i^{d}	Hop requirement for delay at node i
L_i^{r}	Hop requirement for reliability at node i
D_i	Actual delay of the packet arriving at node i
R_i	Reliability requirement assigned to the path through node
i	
$\mathbf{d}_{ij}$	Delay of link l_{ij}, described as a random variable
$\mathbf{r}_{ij}$	Reliability of link l_{ij}, described as a random variable
x_j	Decision variable of whether link (i,j) is used
d_{ij}	Mean of $\mathbf{d}_{ij}$
r_{ij}	Mean of $\mathbf{r}_{ij}$
Δ_{ij}^{d}	Standard deviation of $\mathbf{d}_{ij}$
Δ_{ij}^{r}	Standard deviation of $\mathbf{r}_{ij}$

source node s,

$$\begin{aligned} &\text{Minimize } \sum_{j=1}^{P} x_j \\ &\text{subject to } x_j d_j \le D\,, \\ &\qquad \mathbf{r} = 1 - \prod_{j=1}^{P} 1 - x_j r_j \ge R\,, \\ &\qquad x_j = 0 \text{ or } 1, \text{ for all } j = 1, 2, \ldots, P\,, \end{aligned} \tag{2}$$

where D and R are denoted as the delay and reliability QoS requirements respectively, and the x_j are the decision variables for whether path j is chosen or not. This defines a 0–1 integer programming problem. For clarity, the notation used in this chapter is explained in Table 1.

The problem definition requires exact information about path quality, which is almost impossible to obtain in WSNs; hence, only soft-QoS provisioning is achievable. We can formulate the constraints of the defined problem in a probabilistic way:

$$\begin{aligned} &\text{Minimize } \sum_{j=1}^{P} x_j \\ &\text{subject to } P(x_j \mathbf{d}_j \le D) \ge \alpha, \text{ for } D > 0 \\ &\qquad P(\mathbf{r} \ge R) \ge \beta \\ &\qquad x_j = 0 \text{ or } 1, \quad \forall j \in N(i)\,, \end{aligned} \tag{3}$$

Constraint (3) can be further simplified as

$$P\left(\sum_{j=1}^{P} \log(1 - x_j r_j) \le \log(1 - R)\right) \ge \beta\,.$$

This formulation is a nonlinear program, which could have more than one solution. Solving this nonlinear program at each node once a packet has been received is not practical. So an approximate method, which could significantly simplify the computation of the original problem, while providing comparably fine results, may be more practical.

Though the end-to-end QoS problem formulation provides the exact optimal routing solution, it is subject to many inextricable challenges. First, wireless links are susceptible to fading, interference and traffic variation; therefore, it is almost impossible to obtain the exact instantaneous link state information. So path information, which is accumulated along all links on it, is even more unpredictable. Second, keeping path metrics consistent at all nodes is an even more formidable problem. Since it takes some time for updates to propagate across the network, some nodes refresh their path information with the new updates received, while other nodes are still using obsolete information for routing decisions. A packet going through nodes with asynchronous path information may miss the QoS requirement. Third, storage of voluminous end-to-end path information is dreadfully memory demanding. Possible paths between two nodes may be numerous given densely deployed nodes, whereas a sensor node is equipped with very limited memory. Furthermore, manipulation of end-to-end information is computationally burdensome for sensor nodes. The delay-constrained path problem is known to be NP-hard. The complexity is beyond the computation and energy tolerance of sensors.

The preceding reasons shed light on link-based QoS routing. Per hop information is convenient to acquire

and maintain at a low overhead cost. The acquired neighbor information is enough to make routing decisions, which saves a large amount of computation. Thus, sensor nodes are free of intricate computation. For those superior features of per hop routing, we propose approximating path quality on the basis of link quality. In wireless networks, delay and reliability tend to fluctuate with time. To model this phenomenon, we assume that the link delay and reliability are random processes $\mathbf{d}_{ij}(t)$ and $\mathbf{r}_{ij}(t)$. The time index, t, is omitted for simplicity in the following discussion. We assume that links are independent in terms of delay and reliability. Our goal is to develop a method so that both delay and reliability are ensured with high probability. We only employ the first and second moments of delay and reliability in our derivation. Now the new approximate problem to be addressed based on local information is formulated as

$$\begin{aligned} &\text{Minimize} \sum_{j\in N(i)} x_j \\ &\text{subject to } P(x_j \mathbf{d}_{ij} \le L_i^{\mathrm{d}}) \ge \alpha, \text{ for } L_i^{\mathrm{d}} > 0\,, \\ &\qquad P\Big(\sum_{j\in N(i)} x_j \mathbf{r}_{ij} \ge L_i^{\mathrm{r}}\Big) \ge \beta\,, \\ &\qquad x_j = 0 \text{ or } 1, \quad \forall j \in N(i)\,, \end{aligned} \tag{4}$$

where the $x_j's$ are the decision variables, and $\mathbf{d}_{ij}$ and $\mathbf{r}_{ij}$ are the delay and reliability of link l_{ij} at the routing decision instant, respectively. This is a probabilistic integer program. In the original problem definition, the nonlinear program is to be solved only at the source based on end-to-end information. In contrast, the approximate problem is to be resolved at all intermediate nodes since the approximate problem is based on hop information. We further reduce the computation complexity of the approximation constraints [10]. The final problem formulation is

Formulation: At each node i,

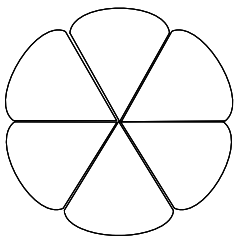
$$\begin{aligned} &\text{minimize} \sum_{j\in N(i)} x_j \\ &\text{s.t. } x_j \left(\frac{\alpha}{1-\alpha}\left(\Delta_{ij}^{\mathrm{d}}\right)^2 + 2L_i^{\mathrm{d}} d_{ij} - d_{ij}^2\right) \le L_i^{\mathrm{d}^2}\,, \\ &\quad \text{when } L_i^{\mathrm{d}} - d_{ij} > 0 \\ &\quad \sum_{j\in N(i)} x_j (r_{ij} + Q^{-1}(\beta)\Delta_{ij}^{\mathrm{r}}) \ge \sqrt[h_i]{R_i}\,, \\ &\quad x_j = 0 \text{ or } 1, \quad \forall\, j \in N(i)\,. \end{aligned}$$

The new optimization problem is a deterministic estimate of the problem formulated in (4). The problem size is proportional to the number of neighbors, which is usually small, so it can be conveniently solved by existing algorithms.

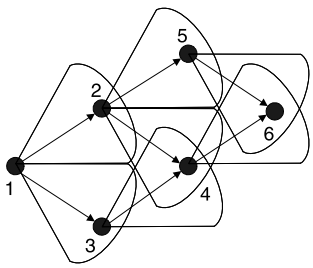
Maximum Flow Problem in Wireless Ad Hoc Networks with Directional Antennas

Owing to the hostile wireless channel, and interference within and among flows, how to achieve the maximum throughput in multihop wireless ad hoc networks has been of great interest over the past few decades. Especially for resource-constrained ad hoc networks, how to improve the system capacity is even more important. With switched-beam technology, the directional antenna is shown to be an appealing option for wireless ad hoc networks. By concentrating RF energy in the intended transmission direction, the spatial transmission region shrinks proportionally to the beam width of a sector. A directional antenna is able to reduce interference and energy consumption, and improve the spatial reuse ratio; thus, it can significantly boost the channel capacity. So the problem of interest is as follows: Given a network topology and existing traffic load, how can we achieve the maximum flow between a given source–destination pair through optimal path selection?

Owing to the different interference patterns induced by directional antennas in ad hoc networks, constraints for the maximum flow are novel and distinct. The maximum flow problem to be addressed here is different from the classical maximum flow problem in network flow theory [1,4,5,6,7,16,20,36,39]. In wired networks, there is no interference among transmissions. Any link can be active at any instant without interference from other links. However, the broadcasting nature of the wireless medium makes the shared wireless channel bottleneck for network flow. To avoid collision, links in a close neighborhood may not be active simultaneously. Furthermore, the interference condition of wireless networks with directional antennas is different from those with omnidirectional antennas. The asymptotic throughput bounds under certain assumptions regarding network topology and node configuration have been derived in many papers [9,14,17,30,40]. Without assumptions regarding the network topology or homo-



Optimization in Ad Hoc Networks, Figure 2
The directional antenna model



Optimization in Ad Hoc Networks, Figure 3
Node graph $G = (V, E)$

geneity of link capacity, we attempt to solve the problem in a generalized setting. For the first time, the interference-constrained maximum flow problem in ad hoc networks with directional antenna is formulated as an optimization problem [12]. This problem is inherently a joint multipath routing and optimal scheduling problem.

According to the beam pattern (beam radius, beam width, beam orientation), we have omnidirectional antennas, single-beam directional antennas (e. g., single-beam switched-beam antennas) and multibeam directional antennas (e. g., multibeam switched-beam antennas or sectorized beam antennas). The beam radius is the distance that a transmission reaches. The beam width is determined by the angle of a sector. For a six-beam directional antenna, the angle of a beam is $\pi/3$. The direction that a beam is targeting is defined as the beam orientation. For directional antennas, both directional transmission and directional reception are enabled. To be clear, for single-beam directional antennas, we assume only one directional transmitting beam or one directional receiving beam can be active at a time; for multibeam directional antennas, multiple directional transmission beams or multiple directional receiving beams can be active at a time. However, a beam can only be either transmitting or receiving at any instant. An illustration of a switched-beam antenna with six beams is shown in Fig. 2. Assume that the antenna is directed to discrete directions, with fixed beam radius and beam width. There is a link between nodes i and j if the distance from node j to node i is shorter than the beam radius.

An illustration of a node graph comprising nodes with directional antennas is shown in Fig. 3, though a realistic node graph is always more complex. Node 1 and node 6 are considered the source node and the destination node, respectively.

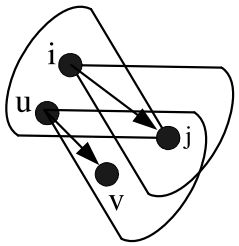
The problem to be addressed here is given network $G(V,E)$ and existing flows, find the maximum flow supported by the network between pair s–d. Before the complete problem formulation is presented, we define the constants and variables used:.

- $x_{i,j}$ indicates the flow over link (i,j).
- f is the flow from source node s to source node d.
- $b_{i,j}(i,l)$ indicates whether link (i,j) is in the l beam of node i.
- B is the total number of beams at each node.
- E is the set of edges.
- V is the set of nodes.
- θ_i^j is the beam of node i that node j resides in.

Now the maximum flow problem can be formulated as the following optimization problem.

$$\begin{aligned}
&\text{Maximize } f \\
&\text{subject to } \sum_{\{j:(i,j)\in E\}} x_{i,j} - \sum_{\{j:(j,i)\in E\}} x_{j,i} \\
&\qquad = \begin{cases} f & i = s\,, \\ 0 & i = V - \{s,d\}\,, \\ -f & i = d\,; \end{cases} \qquad (5) \\
&\qquad \sum_{(k,l)\in A_{i,j}} x_{k,l} \le u_{i,j}\,, \quad \forall (i,j) \in E\,; \\
&\qquad x_{i,j} \ge 0\,, \quad \forall (i,j) \in E\,,
\end{aligned}$$

where $u_{i,j}$ is the normalized remaining capacity or bandwidth ($0 \le u_{i,j} \le 1$) for link (i,j). The second constraint specifies the contention for the resource of each link. This is a traditional maximum flow problem with an added interference constraint.



Optimization in Ad Hoc Networks, Figure 4
Interference caused by (u,v) to (i,j)

In wireless networks, a transmission collision occurs when a receiver is in the communication range of two transmitters, because the receiver receives both time-overlapping signals and cannot decode them correctly. We assume that an antenna both transmits and receives directionally, but that it cannot transmit and receive simultaneously. With a directional antenna, two links interfere with each other if a receiver is in the transmitting beams of both transmitters, shown in Fig. 4. To guarantee a successful reception at node j, any node in the receiving beam of node j cannot transmit towards node j before current transmission finishes.

The protocol model: In the protocol model, the transmission from node i to node j is successful if (1) node j is in the transmission range of node i, $d_{ij} \le R$, where R is the transmission range; (2) any node u that is in the receiving beam of node j from node i is not transmitting in the beam covering node j (when interference range equals transmission range). This means that node j must be outside the transmission beam of node u.

Since the interference region is a beam, the information about the beam to which a link belongs is essential for routing and scheduling. Assume there are B fixed beams for each antenna. Now we can recapitulate condition (2) of the protocol model in the following way:

(2′) When (i,j) is active, for any node u in node j's receiving beam towards node i, the beam θ_u^j should keep silent. Denote $b(i,l)$ as the lth beam of node i, where $l = 1, \ldots, B$.

A single-beam directional antenna can only target one beam at a time. So the channel utilization is shared by all links in all beams. For a single-beam directional antenna, we can formulate the maximum flow problem as the following mixed-integer program (MIP).

Problem formulation:

$$
\begin{aligned}
&\text{Maximize } f \\
&\text{subject to } \sum_{\{j:\,(i,j)\in E\}} x_{i,j} - \sum_{\{j:\,(j,i)\in E\}} x_{j,i} \\
&\quad = \begin{cases} f & i = s\,, \\ 0 & i = V - \{s,d\}\,, \\ -f & i = d\,; \end{cases} \\
&\quad \underbrace{\sum_{u\in b(i,l)} \sum_{(u,v)\in E} x_{u,v} b_{u,v}(u,\theta_u^i)}_{\text{interfering links in the } l\text{th beam}} \quad + \\
&\quad \underbrace{\sum_{(k,i)\in E} x_{k,i} b_{k,i}(i,l)}_{\text{incoming flows}} \le 1\,, \quad \forall l, i, \\
&\quad \sum_{l=1}^{B} \left(\sum_{(k,i)\in E} x_{k,i} b_{k,i}(i,l) \right. \\
&\qquad \left. + \sum_{(i,j)\in E} x_{i,j} b_{i,j}(i,l) \right) \le 1\,, \quad \forall i \in V\,, \\
&\quad b_{i,j}(i,l) = \begin{cases} 1, & \text{if } (i,j) \in b(i,l)\,, \\ 0, & \text{otherwise} \end{cases} \\
&\quad x_{i,j} \ge 0\,, \quad \forall (i,j) \in E\,.
\end{aligned}
\tag{6}
$$

The first constraint describes the in-flow and the out-flow at each node. The second constraint indicates the flow interference around i as specified by condition (2′) in the protocol model. The first term represents the sum of flows causing interference to node i in beam l. When those flows are active, node i must restrain from receiving. The second term stands for the total incoming flows to node i in beam l. The sum of these two terms should be less than the normalized beam capacity of 1. By beam capacity, we mean the channel capacity or bandwidth, which is a constant. The third constraint describes the time-sharing constraint. The second and third constraints aggregately describe the interfering flows at a node. The contention region includes flows over all arcs in the one-hop area of a node.

Denote M as the number of links in the network, N as the number of nodes in the network. The number of variables and constraints in this MIP are M and

$O(N + M)$, respectively, where $O(x)$ indicates the variable on the order of x.

For a single pair of source and destination nodes in wireless networks with multibeam directional antennas, the problem formulation in (5) can be expanded more specifically as follows: *Problem formulation*:

$$\begin{aligned}
&\text{Maximize } f \\
&\text{subject to } \sum_{\{j:\,(i,j)\in E\}} x_{i,j} - \sum_{\{j:\,(j,i)\in E\}} x_{j,i} \\
&\quad = \begin{cases} f & i = s\,, \\ 0 & i = V - \{s,d\}\,, \\ -f & i = d\,; \end{cases} \\
&\quad \sum_{u\in b(i,l)} \sum_{(u,v)\in E} x_{u,v} b_{u,v}(u,\theta_u^i) \\
&\qquad + \sum_{(k,i)\in E} x_{k,i} b_{k,i}(i,l) \le 1\,, \quad \forall l, i \\
&\quad \sum_{(k,i)\in E} x_{k,i} b_{k,j}(i,l) \\
&\qquad + \sum_{(i,j)\in E} x_{i,j} b_{i,j}(i,m) \le 1\,, \\
&\quad \forall 1 \le l, m \le B,\ \forall i \in V\,, \\
&\quad b_{i,j}(i,l) = \begin{cases} 1, & \text{if } l = \theta_i^j\,, \\ 0, & \text{otherwise} \end{cases} \\
&\quad x_{i,j} \ge 0\,, \quad \forall (i,j) \in E\,.
\end{aligned} \tag{7}$$

The first two constraints are the same as those in (6). The third constraint guarantees that the flow is feasible because the in-flow and the out-flow share the capacity at the node. This constraint also implies that the in-flow from any beam should not be greater than 1. The number of variables and constraints in this MIP are M and $O(N + M)$, respectively.

Joint Mobility Control and Power-Efficient Routing in Sparse Wireless Sensor Networks

Many WSNs have been deployed for environmental monitoring. Such networks are typically sparse and consist of nodes of various capabilities. In a sparse network, a target node may be far away from other nodes and data stations. It has to transmit at high power to reach another node when it reports information collected to the sink. A transmission over a long hop consumes considerable energy, and thus undermines the purpose of long-term monitoring. For certain applications demanding an end-to-end path, communication over long distance is too expensive for energy-constrained sensor nodes.

Current routing protocols for intermittently connected networks are designed for collecting delay-tolerant data in the network [24,34,35,37]. They are incapable of supporting certain applications owing to the large delay and the lack of a path between the source and the destination. In order to support end-to-end data delivery in the sparse network, a novel routing algorithm is proposed to establish energy-efficient paths. Our idea is to utilize the mobility of controllable robots to realize energy-efficient on-demand routing in a sparse WSN. Adding another dimension, mobility, to our design, our work is different from traditional joint power control and routing [2,3,8,15,21,22,25,31,41]. Through optimal placement of mobile robots along with optimal power assignment and path selection, energy efficiency can be significantly improved.

We assume that all nodes have the same receiver sensitivity, or receiving power threshold, P_R. If the received power is greater than the threshold, the reception is successful. Otherwise, the node cannot decode the packet correctly. Since nodes transmit at the same data rate, the transmission time of a packet is a constant. So minimizing energy consumption is equivalent to minimizing power because the constant transmission time does not impact the result. Denote $P_r{}^{ij}$ and $P_t{}^{ij}$ as the receiving power and the transmission power over the directed link (i,j), respectively. Assume the Euclidean distance between nodes i and j is d^{ij}. The propagation model [32] used in our paper is

$$P_r^{ij} = \frac{P_t^{ij} h(G_t, G_r, h_t, h_r, L, \lambda)}{d_{ij}^{\gamma}} = \frac{\alpha P_t^{ij}}{d_{ij}^{\gamma}}\,, \tag{8}$$

where G_t and G_r are the gain factors of the transmitter's and the receiver's antenna and h_t and h_r are the antenna heights of the transmitter and the receiver, respectively. L is the system loss factor not related to propagation. λ is the wavelength. α is determined by G_t, G_r, h_t, h_r, L and λ, which is a constant in our paper. γ indicates the path loss exponent, $2 \le \gamma \le 6$. To correctly receive the data, the transmission power over link (i,j) must satisfy

the following condition

$$P_t^{ij} = \frac{P_r^{ij} d_{ij}^{\gamma}}{\alpha} \geq \frac{P_R d_{ij}^{\gamma}}{\alpha} . \tag{9}$$

Denote (x_i,y_i) as the coordinates of node i. Substituting d_{ij} with coordinates into (9) gives

$$P_t^{ij} \geq \frac{P_R((x_i - x_j)^2 + (y_i - y_j)^2)^{\gamma/2}}{\alpha} . \tag{10}$$

Assume the power consumed in packet reception is P_r, which is the same for all nodes. The problem of concern is how to actively place robots so that the total power consumption for data delivery is minimized. Therefore, determining the best positions of the robots, say (x_r,y_r), is one of the objectives. Besides, choosing the path and the corresponding transmission powers of all intermediate nodes on the path also determines the total power expenditure in communications. The energy consumed in mechanical movement is not considered, because the robot is constantly roaming in its territory to collect data messages.

Single Robot Assume there are N sensor nodes and one robot in the field. A single path is to be established from the source node to the data station. Therefore, each node on the path just transmits to a single next-hop node. The transmission power over a link is determined by the hop distance. Wireless links are of poor quality, so the link error rate plays an important role in energy consumption. We use expected transmission count (ETX) to quantify the retransmissions caused by link error. ETX is the expected number of transmissions for a successful reception. Assume the expected transmission count over link (i,j) is ETX_{ij}, then the expected power consumption for transmitting a packet with a link retransmission mechanism is $P_t{}^{ij}\mathrm{ETX}_{ij}$. An indicator variable A_{ij} is used to identify if link (i,j) is active. Then the transmission power of node i, $P_t{}^{i}$, is

$$P_t^i = \sum_{j=1, j\neq i}^{N+1} A_{ij}\mathrm{ETX}_{ij} P_t^{ij} .$$

When only one robot is deployed, we name it as the $N+1$th node. Now the problem of finding the minimum power path from the source sensor node s to the data station d can be formulated as an optimization problem

$$\text{Minimize} \sum_{i=1}^{N+1} \left(\sum_{j=1, j\neq i}^{N+1} A_{ij}\mathrm{ETX}_{ij} P_t^{ij} + \sum_{k=1, k\neq i}^{N+1} A_{ki} P_r^i \right) \tag{11}$$

$$\text{subject to } A_{ij} P_t^{ij} = A_{ij} \frac{P_R((x_i - x_j)^2 + (y_i - y_j)^2)^{\gamma/2}}{\alpha}, \tag{12}$$

$$\sum_{j=1, j\neq i}^{N+1} A_{ij} \leq 1 , \quad \forall i \tag{13}$$

$$\sum_{k=1, k\neq i}^{N+1} A_{ki} - \sum_{j=1, j\neq i}^{N+1} A_{ij} = \begin{cases} -1, & i = s \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

$$A_{ij} = 0 \text{ or } 1 . \tag{15}$$

In this optimization problem, the A_{ij} determine the path between the source and the destination. Since the locations of the sensor nodes are known and fixed by the sink, their coordinates are constant. We only need to determine the coordinates of the robot, which are x_{N+1} and y_{N+1}. Clearly, this is a three-dimensional problem. The objective in (11) is to minimize the total power consumption of all nodes on the end-to-end path. The same as for (10), to reach node j from node i, the transmission power has to satisfy (12). Since only a single path is chosen to carry the traffic, at most one incoming link can be active for every node, as shown in (13). Constraint (14) requires that only one outgoing link is selected for the node on the chosen path except the destination. As an indicator of whether a link is selected, A_{ij} is set to 1 if the corresponding link is selected, otherwise it is 0.

As the power consumption for reception is rather fixed, we could simplify P_r^i to P_r. According to (14), the objective can be rewritten as the following expression:

$$\min \sum_{i=1}^{N+1} \left(\sum_{j=1, j\neq i}^{N+1} A_{ij} \left(P_t^{ij}\mathrm{ETX}_{ij} + P_r \right) \right) .$$

As shown in [11], ETX is a constant for all links in our scenario. Then the problem of minimum energy

routing with the consideration of link retransmission is

$$\begin{aligned}
\text{Minimize } f(A,P) &= \sum_{i=1}^{N+1}\left(\sum_{j=1,j\neq i}^{N+1} A_{ij}(P_t^{ij}ETX + P_r)\right)\\
\text{s. t.} \quad A_{ij}P_t^{ij} &= A_{ij}\frac{P_R((x_i-x_j)^2+(y_i-y_j)^2)^{\gamma/2}}{\alpha}, \quad \forall i\\
\sum_{j=1,j\neq i}^{N} A_{ij} &\le 1, \quad \forall i\\
\sum_{k=1,k\neq i}^{N} A_{ki} - \sum_{j=1,j\neq i}^{N} A_{ij} &= \begin{cases} -1, & i=s\\ 0, & \text{otherwise}\end{cases}\\
A_{ij} &= 0 \text{ or } 1.
\end{aligned} \tag{16}$$

Define $\text{EPC}_{ij} = P_t^{ij}\text{ETX} + P_r$ as the *expected power consumption* over link (*i*,*j*). The decision variables in (16) include variables with integer and real values. The optimization problem is a nonlinear MIP, which is usually hard to solve. So we will decompose the formulated problem into subproblems, then solve them sequentially. The strategy of decomposition is as follows. First, we assume routing is fixed for any $\widetilde{A} \in \Psi$, where Ψ denotes the solution space specified by the routing constraints (13)–(15). Given the restriction of the original problem, we could obtain the optimal solution with the coordinates of the mobile robot represented as a function of A. The physical meaning of this subproblem is that given a schedule, what is the optimal location of the robot which yields the minimum-energy path? The expected energy consumption associated with each link is the link weight. The energy consumption is the measurement of the path quality. By varying the placement of the robot over a limited number of locations in each iteration, one can discover the optimal route through the shortest-path problem. Among all the shortest paths for each specific location of the robot, the one with the optimal quality is selected. This algorithm can solve the problem in polynomial time.

In the derivation, it is interesting to observe that the optimal coordinates of the robot are the average of the coordinates of its upstream and downstream nodes; therefore, on the minimum-energy path, the robot is always situated in the middle of two sensor nodes. With this observation, the solution space of the potential optimal position of the robot is reduced to the possible links between sensor nodes, say $N(N+1)/2$. This result helps us develop an algorithm to solve the minimum power routing problem efficiently. Given the possible locations of the mobile robot, the optimization problem (16) is

$$\begin{aligned}
\text{Minimize } f(A) &= \sum_{i=1}^{N+1}\left(\sum_{j=1,j\neq i}^{N+1} A_{ij}\text{EPC}_{ij}\right)\\
\text{subject to} \quad \sum_{j=1,j\neq i}^{N} A_{ij} &\le 1, \quad \forall i\\
\sum_{k=1,k\neq i}^{N} A_{ki} - \sum_{j=1,j\neq i}^{N} A_{ij} &= \begin{cases} -1, & i=s\\ 0, & \text{otherwise}\end{cases}\\
A_{ij} &= 0 \text{ or } 1.
\end{aligned} \tag{17}$$

Taking EPC_{ij} as the weight of the associated link, the above problem is actually a shortest-path problem. It can be conveniently solved by a shortest-path algorithm, such as Dijkstra's algorithm.

Multiple Robots When multiple robots are deployed in the field, the problem becomes more complicated. Suppose there are R robots; we number them as the $N+1, N+2, \ldots, N+R$ nodes, respectively. Now the problem can be formulated as follows:

$$\begin{aligned}
\text{Minimize } f(A,P) &= \sum_{i=1}^{N+R}\left(\sum_{j=1,j\neq i}^{N+R} A_{ij}(P_t^{ij}\text{ETX} + P_r)\right)\\
\text{s. t. } A_{ij}P_t^{ij} &= A_{ij}\frac{P_R\left((x_i-x_j)^2+(y_i-y_j)^2\right)^{\gamma/2}}{\alpha}, \quad \forall i\\
\sum_{j=1,j\neq i}^{N+R} A_{ij} &\le 1, \quad \forall i\\
\sum_{k=1,k\neq i}^{N+R} A_{ki} - \sum_{j=1,j\neq i}^{N+R} A_{ij} &= \begin{cases} -1, & i=s\\ 0, & \text{otherwise}\end{cases}\\
A_{ij} &= 0 \text{ or } 1.
\end{aligned}$$

In this problem, we need to determine the best positions of all robots, which may correlate with each other. In the single-robot case, the optimal location of

the robot solely depends on the locations of the sensor nodes, which are known; therefore, it is much more difficult to compute the optimal locations of multiple robots than it is to compute the optimal location of a single robot.

Following a similar technique in the single-robot case, we decompose the optimization problem into a subproblem and a master problem. The subproblem computes the best positions for the robots, while the master problem determines the optimal path. Denote $G = (V, E, W)$ as the graph of the sensor network. V is the set of vertices with $N + 1$ nodes (including data station t). E contains all the edges between each pair of vertices. W specifies the weight which is EPC-associated with each edge.

Theorem 1 *Denote $\hat{G}(\hat{V}, \hat{E}, \hat{W})$ as the resulting graph with the optimal placement of robots in the sensor network, which minimizes the energy consumption to deliver a packet from node s to node t over all possible paths. Given a path from node s to node t in graph $\hat{G}$, the robots must be situated on the edges in G(V,E,W). More specifically, each robot is located at the spot that equally divides the distance along the edge in G.*

Please refer to [11] for the detailed proof. Although Theorem 1 reveals the potential locations of robots given a path, it is still hard to solve the minimum-energy problem in the multiple-robots case owing to the large solution space. It remains to select the edges in G where robots reside and determine how many robots. Denote Q_{ij} as the number of robots on edge (i,j). The problem can be formulated as follows:

Minimize

$$\sum_{i=1}^{N}\left(\sum_{j=1,j\neq i}^{N} A_{ij}\left(\frac{P_t^{ij}\mathrm{ETX}}{(Q_{ij}+1)^{\gamma-1}} + (Q_{ij}+1)P_r\right)\right)$$

subject to $A_{ij}P_t^{ij} = A_{ij}\dfrac{P_R d_{ij}^{\gamma}}{\alpha}, \quad \forall i$

$$\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} Q_{ij} \leq R$$

$$\sum_{j=1,j\neq i}^{N} A_{ij} \leq 1, \quad \forall i$$

$$\sum_{k=1,k\neq i}^{N} A_{ki} - \sum_{j=1,j\neq i}^{N} A_{ij} = \begin{cases} -1, & i = s \\ 0, & \text{otherwise} \end{cases}$$

$A_{ij} = 0$ or 1.

The formulation is a nonconvex integer program, which is NP-hard. In [11], an approximation algorithm is proposed to solve the problem efficiently.

References

1. Armstrong RD, Chen W, Goldfarb D, Jin Z (1998) Strongly polynomial dual simplex methods for the maximum flow problem. Math Programm 1:14–33
2. Bhatia R, Kodialam M (2004) On power efficient communication over multi-hop wireless networks: joint routing, scheduling and power control. IEEE INFOCOM, HongKong, pp 1457–1466
3. Cruz RL, Santhanam AV (2003) Optimal routing, link scheduling and power control in multi-hop wireless networks. IEEE INFOCOM, San Francisco, pp 702–711
4. Fernández-Baca D, Martel CU (1989): On the efficiency of maximum-flow algorithms on networks with small integer capacities. Algorithmica 1:173–189
5. Ghosh S, Gupta A, Pemmaraju SV (1997) A self-stabilizing algorithm for the maximum flow problem. Distribut Comput 4:167–180
6. Goldberg AV, Grigoriadis MD, Tarjan RE (1991) Use of dynamic trees in a network simplex algorithm for the maximum flow problem. Math Programm 1–3:277–290
7. Goldfarb D, Hao J (1990) A primal simplex algorithm that solves the maximum flow problem in at mostnm pivots and O(n 2 m) time. Math Programm 1–3:353–365
8. Gomez J, Campbell AT (2004) A case for variable-range transmission power control in wireless multihop networks. IEEE INFOCOM, HongKong, pp 1425–1436
9. Gupta R, Kumar PR (2000) The capacity of wireless networks. IEEE Trans Inform Theory 2:388–404
10. Huang X, Fang Y Multiconstrained QoS multipath routing in wireless sensor networks. ACM Wireless Networks, available online on Jan. 2007
11. Huang X, Li P, Fang Y (2008) Joint mobility control and power efficient routing in sparse wireless sensor networks. submitted to INFOCOM
12. Huang X, Wang J, Fang Y (2007) Achieving maximum flow in interference-aware wireless sensor networks with smart antennas. Elsevier Ad Hoc Networks 6:885–896
13. Jaffe JM (1984) Algorithms for finding paths with multiple constraints. Networks 14:95–116
14. Jain K, Padhye J, Padmanabhan VN, Qiu L (2003) Impact of interference on multi-hop wireless network performance. Proc. of ACM MOBICOM (2003), San Diego, pp 66–80
15. Jung E-S, Vaidya NH (2005) Power aware routing using power control in ad hoc networks. Mobile Comput Commun Rev 3:7–18
16. Kim D, Pardalos PM (2000) A dynamic domain contraction algorithm for nonconvex piecewise linear network flow problems. J Global Optim 1–4:225–234
17. Kodialam M, Nandagopal T (2003) Characterizing achievable rates in multi-hop wireless networks: the joint routing

and scheduling problem. Proc. of ACM MOBICOM (2003), San Diego, CA, pp 42–54
18. Korkmaz T, Krunz M (2003) Bandwidth-delay constrained path selection under inaccurate state information. IEEE/ACM Trans Networking 3:384–398
19. Kuipers FA, Mieghem PV (2003) On the complexity of QoS routing. Comput Commun 4:376–387
20. Kumar S, Gupta P (2003) An incremental algorithm for the maximum flow problem. J Math Modell Algorithms 1:1–16
21. Li Q, Aslam J, Rus D (2001) Online power-aware routing in wireless ad-hoc networks. Proc. of Annual Int'l Conf. On Mobile Computing And Networking, Rome, pp 97–107
22. Li Y, Ephremides A (2005) Joint scheduling, power control, and routing algorithm for ad-hoc wireless networks. Proc. of Hawaii Int'l Conf. on System Science, Big Island
23. Lin X, Shroff NB, Srikant R (2006) A tutorial on cross-layer optimization in wireless networks. IEEE J Selected Areas Commun 8:1452–1463
24. Liu W, Zhang Y, Lu K, Fang Y (2006) Energy conservation through resource-aware movement in heterogeneous mobile ad hoc networks. J Comb Optim 1:7–20
25. Lu G, Krishnamachari B (2005) Energy efficient joint scheduling and power control for wireless sensor networks. Proc. of IEEE Conf. on Sensor and Ad Hoc Commu. and Networks, Santa Clara, pp 362–373
26. Mieghem PV, Kuipers FA (2004) Concepts of exact QoS routing algorithms. IEEE/ACM Trans Networking 5:851–864
27. Neve HD, Mieghem PV (1998): A multiple quality of service routing algorithm for PNNI. Proc. ATM Workshop, Fairfax, pp 324–328
28. Ota N, Hooks D, Wright P, Ausiander D, Peffer T (2003) Poster abstract: wireless sensor networks characterization-application to demand response energy pricing. Proc. ACM Int'l Conf. on Embedded Networked Sensor Systems, Los Angeles, pp 334–335
29. Palomar DP, Chiang M (2006) A tutorial on decomposition methods for network utility maximization. IEEE J Selected Areas Commun 8:1439–1451
30. Peraki C, Servetto SD (2003) On the maximum stable throughput problem in random networks with directional antennas. Proc. of the 4th ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003), Annapolis, pp 76–87
31. Ramanathan R, Hain RR (2000) Topology control of multihop wireless networks using transmit power adjustment. IEEE INFOCOM, pp 404–413
32. Rappaport TS (1996) Wireless communications principles and practice. Prentice Hall, Upper Saddle River
33. Resende MGC, Pardalos PM (2006) Handbook of optimization in telecommunications. Springer, Berlin
34. Shah RC, Roy S, Jain S, Brunette W (2003) Data mules: modeling a three-tier architecture for sparse sensor networks. Proc. of IEEE Int'l Workshop on Sensor Network Protocols and Applications, Anchorage, pp 30–41
35. Somasundara AA, Kanasai A, Jea DD, Estrin D, Srivastava MB (2006) Controllably mobile infrastructure for low energy embedded network. IEEE Trans Mobile Comput 6:958–973
36. Tardos E, Wayne KD (1998) Simple generalized maximum flow algorithms. Lecture Notes in Computer Science, pp 310–324
37. Tariq MMB, Ammar M, Zegura E (2006) Message ferry route design for sparse ad hoc networks with mobile nodes. Proc. of ACM int'l Symposium on Mobile Ad Hoc Networking and Computing, Florence, pp 37–48
38. Tsirigos A, Hass ZJ (2004) Analysis of multipath routing, part 2: mitigation of the effects of frequently changing network topologies. IEEE Trans Wireless Commun 2:500–511
39. Tuncel L (1994) On the complexity of preflow-push algorithms for maximum-flow problems. Algorithmica 4:353–359
40. Yi S, Pei Y, Kalyanaraman S (2003) On the capacity improvement of ad hoc wireless networks using directional antennas. Proc. of the 4th ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003), Annapolis, pp 108–116
41. Xing G, Lu C, Zhang Y, Huang Q, Pless R (2005) Minimum power configuration in wireless sensor networks. Proc. of ACM int'l Symposium on Mobile Ad Hoc Networking and Computing, Urbana-Champaign, pp 390–401
42. Yuan X (2002) Heuristic algorithms for multiconstrained quality-of-service routing. IEEE/ACM Trans Networking 2:244–256
43. Znati TF, Melhem R (2004): Node delay assignment strategies to support end-to-end requirement in heterogeneous networks. IEEE/ACM Trans Networking 5:879–892

Optimization Based Framework for Radiation Therapy

GINO J. LIM
Department of Industrial Engineering,
University of Houston, Houston, USA

MSC2000: 68W01, 90-00, 90C90, 92-08, 92C50

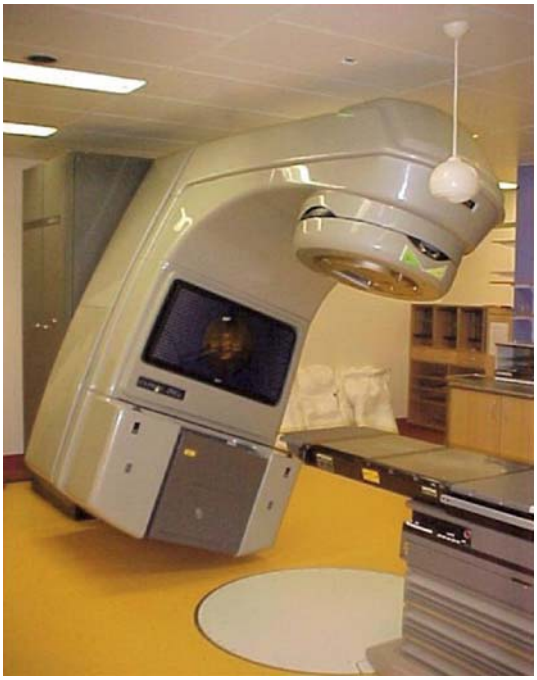
Article Outline

Introduction
Applications and Methods
The Radiation Treatment Planning Procedure
Use of Optimization Techniques
Gamma Knife Radiosurgery
Three-Dimensional Conformal Radiation Therapy
Intensity Modulated Radiation Therapy
References

Introduction

Cancer is the second leading cause of death in the United States [2]. Treatment options are determined by the type and the stage of the cancer and include surgery, *radiation therapy*, chemotherapy, etc. Physicians often use a combination of those treatments to obtain the best results. Our application is based on radiation therapy. Thanks to the continuous development of new treatment machines and technologies, it is now possible to have much greater control over the treatment delivery than was possible in the past. Researchers in optimization community have made significant contributions in improving the quality of such treatment plans for cancer patients [5,6,12,13,24,26,27,29,36,40,42,46,47]. The common objective of radiotherapy planning is to achieve tumor control by planning a significant total dose of radiation to the cancerous region to sterilize the tumor without damaging the surrounding healthy tissues. One of the major difficulties in treatment planning is due to the presence of organs-at-risk (OARs). An OAR is a healthy organ located close to the target. The dose of radiation must be severely constrained to avoid reaching an OAR because an overdose in the OAR may lead to medical complications. OAR is also termed "sensitive structure" or "critical structure" in the literature. There are several survey articles that cover the essential elements of the radiation treatment planning problem, see [17,31,37,40].

Our aim in this chapter is to describe optimization techniques to improve the delivery of radiation for cancer patients. Two types of radiation therapy are the most common and include *teletherapy* (or external beam therapy) and *brachytherapy*. Radiation is delivered from outside the body and directed at the patient's tumor location using special radiation delivery machines in *teletherapy*, (see Fig. 1). Different devices produce different types of radiation and they include Cobalt-60 machines (such as Gamma Knife radiosurgery), linear accelerators (such as intensity modulated radiation therapy), neutron beam machines, orthovoltage x-ray machines, and proton beam machines. In *brachytherapy*, radioactive substances are placed within the tumor region in the form of wires, seeds, or rods. Types of brachytherapy are categorized depending on how the radioactive sources are placed inside the body such as interstitial brachytherapy, intracavitary brachytherapy, intraluminal radiation therapy, and radioactively tagged molecules given intravenously. There are two types of radiation treatment planning: forward planning and inverse planning. In forward planning, treatment plans are typically generated by a trial and error approach. Therefore this process can be very tedious and time-consuming, and does not necessarily produce "high-quality" treatment plans. On the other hand, there has been a significant move toward inverse treatment planning. Such a move is due to significant advances in modern technologies such as imaging technologies and computer control to aid the delivery of radiation. The inverse treatment planning procedure allows modeling highly complex treatment planning problems from brachytherapy to external beam therapy. Inverse planning is also called computer based treatment planning.



Optimization Based Framework for Radiation Therapy, Figure 1
An external beam therapy machine

In inverse treatment planning, an objective function is defined to measure the goodness (quality) of a treatment plan. Two types of objective functions are often used: dose-based models and biological (radiobiological) models. The biological model argues that optimiza-

tion should be based on the biological effects resulting from the underlying radiation dose distributions. The treatment objective is usually to maximize the tumor control probability (TCP) while keeping the normal tissue complication probability (NTCP) within acceptable levels. In the dose based model, achieving accurate radiation dose distributions on organs of interest is the main concern. The treatment objective is to minimize the deviation between the projected dose that the patient will receive and the prescribed dosage that the physician provides. This is the main approach we will describe in this chapter. The biological aspect is implicitly given in the physician's prescription.

Applications and Methods

The Radiation Treatment Planning Procedure

When a patient comes in for a treatment, the doctor will choose what type of radiation beam to use for the treatment. The choice of radiation will depend on the type of the cancer the patient has and how far into the body the radiation should penetrate to reach the tumor volume.

The next step is to identify the three-dimensional shapes of organs of interest in the patient's body. The location and the volume of organs are obtained by using three-dimensional imaging techniques such as computer tomography (CT) or magnetic resonance imaging (MRI). Based on three-dimensional images, a physician specifies the tumor region as gross tumor volume (GTV), clinical target volume (CTV), planning target volume (PTV), and OARs. GTV represents the volume of the known tumor. CTV represents the volume of the suspected microscopic spread. PTV is the marginal volume necessary to account for setup variations and organ and patient motion, i. e. PTV = GTV + marginal volume. Typically, PTV is used in designing treatment plans and we call PTV a *target* in this chapter. Organ geometries are the key input data for designing a treatment plan.

A radiation physicist and a dosimetrist meet to decide what kind of radiation delivery machine to use and the number of treatments for the patient. Optimization algorithms are crucial to determine how much and where to deliver radiation in the patient's body. For most types of cancer, radiation therapy is administered 5 days each week for 5 to 8 weeks. Using small radiation doses daily rather than a few large doses helps protect normal body tissues in the treatment area. Resting over the weekend will allow some time for normal cells to recover from the radiation damage.

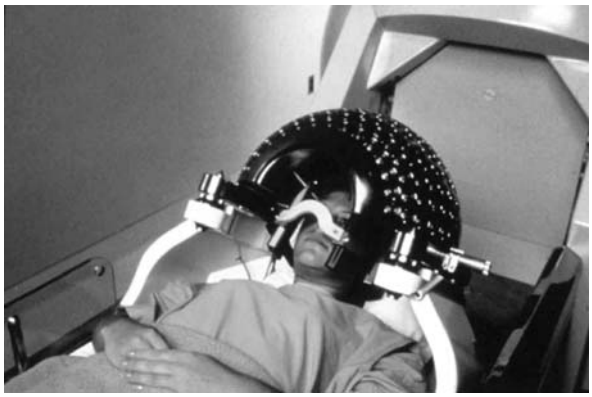
In optimization, the three-dimensional volume is represented by a grid of voxels. There are several inputs required in optimization approaches in radiation treatment planning. The first input describes the machine that delivers radiation. The second and troublesome input is the dose distribution of a particular treatment problem. A dose distribution consists of dose contribution to each voxel of the region of interest from a radiation source. It can be expressed as a functional form or a set of data. However, a difficulty of using such distributions are either the functional form is highly nonlinear [13] or the amount of data that specifies the dose distribution is too large [27]. This problem needs to be overcome in a desirable automated treatment planning tool. The third common input is the set of organ geometries that are of interest to the physician. Further common inputs are the desired dose levels for each organ of interest. These are typically provided by physicians. Other types of inputs can also be specified depending on the treatment planning problems. However, a desirable treatment planning system should be able to generate high quality treatment plans with minimum additional inputs and human guidance.

Use of Optimization Techniques

Two major goals in treatment planning optimization are speed and quality. Solution quality of a treatment plan can be measured by *uniformity, conformity*, and *avoidance* [12,27,29]. Fast solution determination in a simple manner is another essential part of a clinically useful treatment planning procedure. Acceptable dose levels of these requirements are established by various professional and advisory groups.

It is important for a treatment plan to have uniform dose distributions on the target so that *cold* and *hot spots* can be minimized. A *cold spot* is a portion of an organ that receives below its required radiation dose level. On the other hand, a *hot spot* is a portion of an organ that receives more than the desired dose level. The uniformity requirement ensures that radiation delivered to tumor volume will have minimum number of *hot spots* and *cold spots* on the target. This require-

ment can be enforced using lower and upper bounds on the dose, or approximated using penalization. The conformity requirement is used to achieve the target dose control while minimizing the damage to OARs or healthy normal tissue. This is generally expressed as a ratio of cumulative dose on the target over total dose prescribed for the entire treatment. This ratio can be used to control conformity in optimization models. As we mentioned earlier, a great difficulty of producing radiation treatment plans is the proximity of the target to the OARs. An avoidance requirement can be used to limit the dose delivered to OARs. Finally, simplicity requirements state that a treatment plan should be as simple as possible. Simple treatment plans typically reduce the treatment time as well as implementation error. In this chapter, we introduce a few optimization models and solution techniques that are practically useful for radiation treatment modalities: Gamma Knife radiosurgery, conventional three-dimensional conformal therapy (3DCRT) [27], intensity modulated radiation therapy (IMRT) [4,18]. Many treatment planning models are developed for proton therapy [48] and tomotherapy [11,22]. But they are beyond the scope of this chapter.



Optimization Based Framework for Radiation Therapy, Figure 2
Radiation delivery: a collimator is positioned on patient's head

Gamma Knife Radiosurgery

Problem The Gamma Knife is a highly specialized treatment unit that provides an advanced stereotactic approach to the treatment of tumor and vascular malformations within the head [14]. The Gamma Knife delivers a single, high dose of radiation emanating from 201 Cobalt-60 unit sources. All 201 beams simultaneously intersect at the same location in space to form an approximately spherical region that is typically termed a shot of radiation.

Gamma Knife Radiosurgery begins by finding the location and the size of the tumor. After administering local anesthesia, a stereotactic head frame is fixed to the patient's head using adjustable posts and fixation screws. This frame establishes a coordinate frame within which the target location is known precisely and serves to immobilize the patient's head within an attached focussing helmet during the treatment. An MRI or CT scan is used to determine the position of the treatment volume in relation to the coordinates determined by the head frame. Once the location and the volume of the tumor are identified, the neurosurgeon, the radiation oncologist, and the physicist work together in order to develop the patient's treatment plan. Multiple shots are often used in a treatment using a Gamma Knife due to the irregularity and size of tumor shapes and the fact that the focussing helmets are only available in four sizes (4, 8, 14 and 18 mm).

The plan aims to deliver a high dose of radiation to the intracranial target volume with minimum damage to the surrounding normal tissue. The treatment goals can vary from one neurosurgeon to the next. But the following requirements are typical for a treatment plan, although the level of treatment and importance of each may vary.

1. A complete 50% isodose line coverage of the target volume. This means that the complete tumor volume must receive at least 50% of the maximum dosage delivered to the target. This can be thought of as a "uniformity" requirement.
2. Minimize the nontarget volume that is covered by a shot or the series of delivered shots. This requirement is clear and can be thought of as a "conformity" requirement.
3. Limit the amount of dosage that is delivered to organs at risk that are close to the target. Such requirements can be thought of as an "avoidance" requirement.

There are standard rules established by the Radiation Therapy Oncology Group (RTOG) that recom-

mend the acceptable uniformity and conformity requirements. In addition to these requirements, it is also preferable to use a small number of shots to limit the treatment times and thus increase the number of patients that can be treated.

Optimization Model Formulation Most commonly known optimization models include mixed integer programming (MIP) model and mixed integer nonlinear programming (MINLP) model. MIP models guarantee the global optimality, but they are not practically useful due to the long computation time. We discuss an MINLP model that has shown to be practically useful [12]. A variant of this approach has been successfully implemented for planning treatments [39].

Suppose that the number of radiation shots for the treatment is given a priori. Adding the goal of minimizing this number is typically straightforward in the optimization model. However, solving such models can be extremely difficult.

Decision Variables: Consider a grid $\mathcal{G}$ of voxels. Let $\mathcal{T}$ denote the subset of voxels that are within the target and $\mathcal{N}$ represents the subset of voxels that are not in the target. Let $D_{i,j,k}$ denote the amount of radiation dose that a voxel (i, j, k) receives. In general, there are three types of decision variables.

1. *A set of discrete coordinates* (x_s, y_s, z_s). These are the target locations for the (ellipsoidal) shots.
2. *A discrete set of collimator sizes w*: currently four different sizes of focussing helmets are available (4 mm, 8 mm, 14 mm, 18 mm).
3. *Radiation exposure time* $t_{s,w}$: the amount of radiation to be delivered for each shot centered at location (x_s, y_s, z_s).

Constraints:

1. *Uniformity – Isodose line coverage:* A treatment plan is normally considered acceptable if a θ% *isodose* curve encompasses the tumor region. For example, 50% isodose curve is a curve that encompasses all voxels that receive at least 50% of the maximum radiation dose that is delivered to any voxels in the target volume.

$$\theta \le D_{i,j,k} \le 1, \quad (i,j,k) \in \mathcal{T} \tag{1}$$

2. *Choosing shot sizes:* The location of the shot center is chosen by a continuous optimization process. Choosing the particular shot width at each shot location is a discrete optimization problem that is treated by approximating the step function

$$H(t) = \begin{cases} 1 & \text{if } t \ge 0 \\ 0 & \text{if } t = 0 \end{cases}$$

by a nonlinear function,

$$H(t) \approx H_\alpha(t) := \frac{2\arctan(\alpha t)}{\pi}\,.$$

For increasing values of α, H_α becomes a closer approximation to the step function H. This process is typically called smoothing.

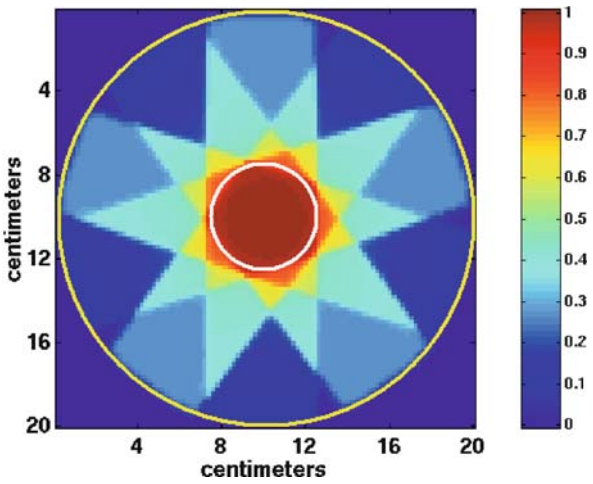
An Optimization Model:

$$\begin{aligned} \min \quad & \sum_{(i,j,k)\in\mathcal{N}} D_{i,j,k} \\ \text{s.t.} \quad & D_{i,j,k} = \sum_{(s,w)\in S\times W} t_{s,w} D_{w(S,i,j,k)} \\ & \theta \le D_{i,j,k} \le 1, \quad \forall (i,j,k) \in \mathcal{T} \\ & n = \sum_{(s,w)\in\{1,\ldots,n\}\times W} H_\alpha(t_{s,w}) \\ & t_{s,w} \ge 0. \end{aligned} \tag{2}$$

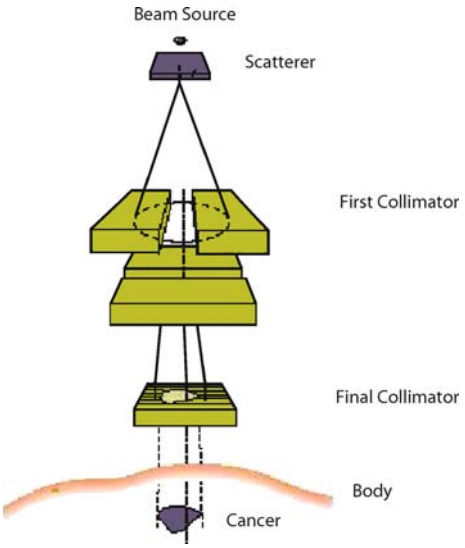
Solution Techniques The most critical problem for solving the optimization model (2) is the large number of voxels that are needed when dealing with large irregular tumors (both within and outside of the target). This makes the optimization problem computationally intractable. One approach to overcoming this problem is to remove a large number of the non-target voxels from the model. While this improves the computation time, this typically weakens the conformity of the dose to the target. Ferris et al. [12] propose a sequential solution approach to speed up the time while maintaining conformality. First, a *coarse grid* problem is solved as a nonlinear programming (NLP) model using reduced data points. Then the *finer grid* NLP problem with full data points is solved using the starting point that was obtained by the coarse grid model in the previous stage. Typically, the solution from this finer grid model is very close to a good local optimum for the MINLP. Using this solution, the full MINLP model is finally solved to determine the values of the three decision variables for this problem.

Three-Dimensional Conformal Radiation Therapy

Problem We learned from Section "Gamma Knife Radiosurgery" that the Gamma knife is specifically designed for treating diseases in the human brain. Three-dimensional conformal radiation therapy (3DCRT) adds much greater flexibility to radiation treatments that can treat various cancer patients in the brain, breast, prostate, etc. One of the main strategies for minimizing morbidity in 3DCRT is to reduce the dose delivered to normal tissues that are spatially well separated from the tumor. This can be done by using multiple beams from different angles. A single radiation beam leads to a higher dose delivered to the tissues in front of the tumor than to the tumor itself. In consequence, if one were to give a dose sufficient to control the tumor with a reasonably high probability, the dose to the upstream tissues would likely lead to unacceptable morbidity. A single beam would only be used for very superficial tumors, where there is little upstream normal tissue to damage. For deeper tumors, one uses multiple cross-firing beams delivered within minutes of one another: All encompass the tumor, but successive beams are directed toward the patient from different directions to traverse different tissues outside the target volume. The delivery of cross-firing beams is greatly facilitated by mounting the radiation-producing equipment on a gantry: multileaf-collimator (MLC).



Optimization Based Framework for Radiation Therapy, Figure 3
Effect of multiple beams: a hot spot is formed in the middle by five beams

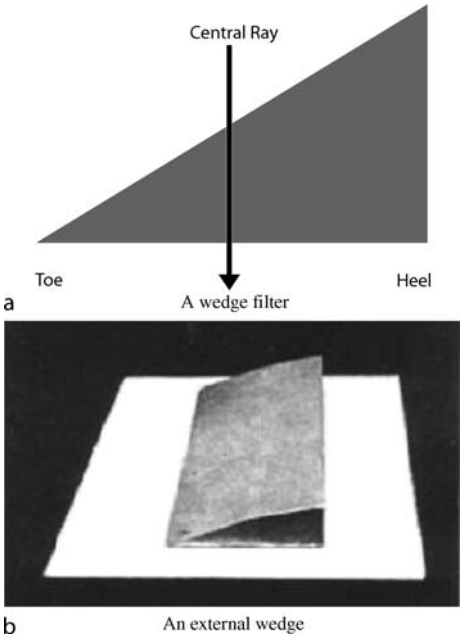


Optimization Based Framework for Radiation Therapy, Figure 4
A beam's-eye-view is a 2D shape of a tumor viewed by the beam source at a fixed angle

Several directed beams noticeably change the distribution of dose, as is illustrated in Fig. 3. As a result, dose outside the target volume can often be quite tolerable even when dose levels within the target volume are high enough to provide a substantial probability of tumor control.

The leaves of the multileaf collimator are computer controlled and can be moved to the appropriate positions to create the desired beam shape. From each beam angle, three-dimensional anatomical information is used to shape the beam of radiation to match the shape of the tumor. Given a gantry angle, the view of the tumor that the beam source can see through the multileaf collimator is called the *beam's-eye-view* of the target (see Fig. 4); [15]. This beam's-eye-view (BEV) approach ensures adequate irradiation of the tumor while reducing the dose to normal tissue.

Wedge Filters: A wedge (also called a "wedge filter") is a tapered metallic block with a thick side (the *heel*) and a thin edge (the *toe*); (see Fig. 5). This metallic wedge varies the intensity of the radiation in a linear fashion from one side of the radiation field to the other. When the wedge is placed in front of the aperture, less radiation is transmitted through the heel of the wedge than through the toe. Figure 5b shows an ex-



Optimization Based Framework for Radiation Therapy, Figure 5
Wedges

ternal 45° wedge, so named because it produces isodose lines that are oriented at approximately 45°. The quality of the dose distribution can be improved by incorporating a wedge filter into one or more of the treatment beams. Wedge filters are particularly useful in compensating for a curved patient surface, which is common in breast cancer treatments.

Two different wedge systems are used in clinical practice. In the first system, four different wedges with angles 15°, 30°, 45°, and 60° are available, and the therapist is responsible for selecting one of these wedges and inserting it with the correct orientation. In the second system, a single 60° wedge (the *universal wedge*) is permanently located on a motorized mount located within the head of the treatment unit. This wedge can be rotated to the desired orientation or removed altogether, as required by the treatment plan.

Optimization Model Formulation Suppose that the data to the optimization models are given. Let $\mathcal{D}_{(i,j,k),A}$ be the dose contribution to voxel (i, j, k) from a beam of weight 1 from angle A, S be a collection of voxels on the sensitive structure(s), and $\mathcal{N}$ be a collection of voxels on the normal tissue. When wedges are allowed in the optimization, the data will be provided as $\mathcal{D}_{(i,j,k),A,F}$ that represents the dose contribution to voxel (i, j, k) from a beam of weight 1 from angle A, using wedge orientation F.

Beam Weight Optimization: The classical optimization problem in conformal radiation therapy is to choose the weights (or intensity levels) to be delivered from a given set of angles. Suppose w_A represent the beam weight delivered from angle A, $D_{(i,j,k)}$ for the total dose deposited to voxel (i, j, k) and λ represent the relative weighting factors in the objective function. Given a set $\Omega = \mathcal{T} \cup S \cup \mathcal{N}$, a general optimization model that determines optimal radiation intensity is

$$\begin{aligned} \min_{w} \quad & \lambda_t f(\mathcal{D}_{\mathcal{T}}) + \lambda_s f(\mathcal{D}_S) + \lambda_n f(\mathcal{D}_{\mathcal{N}}) \\ \text{s.t.} \quad & D_{\Omega} = \sum_{A \in \mathcal{A}} \mathcal{D}_{\Omega,A} w_A, \\ & l \le D_{\mathcal{T}} \le u, \\ & 0 \le w_A, \ \forall A \in \mathcal{A}. \end{aligned} \tag{3}$$

Hard upper and lower bound constraints are imposed on the target dose so that, in the worst case, the resulting solution will satisfy the minimum requirement for a treatment plan. Objective function $f(\mathcal{D})$ can be defined based on the planner's preference, but a general function can be written as

$$f(\mathcal{D}.) = \|\mathcal{D}(\cdot) - \theta\|_p, p \in \{1, 2, \infty\}.$$

Note that θ is the desired dose level for an organ of interest. These problems can be cast as a quadratic programming (QP) problem ($p = 2$), minimizing the Euclidean distance between the dose delivered to each voxel and the prescribed dose [6,35,40,41]. Furthermore, linear programming (LP) has also been extensively used to improve conventional treatment planning techniques [3,24,32,37,40]. The strength of LP is its ability to control *hot* and *cold* spots or integral dose on the organs using constraints, and the presence of many state-of-the-art LP solvers. The LP model replaces the Euclidean norm objective function of a QP with a polyhedral one, for which standard reformulations (see [27,30]) result in linear programming problems. While these techniques still suffer from large amounts of data in $\mathcal{D}_{(i,j,k),A}$, they are typically solved in acceptable time frames. These models tend to find

optimal solutions more quickly than the corresponding QP formulations.

Another technique to convert the quadratic (or more generally convex) problem to a linear program is via a piecewise-linear approximation of the objective (see [36]). For a quadratic function, a uniform spacing for the breakpoints guarantees small approximation errors from the piecewise linear interpolant [23]. Since the piecewise linear interpolant is convex, standard techniques can be used to reformulate this as a linear program [16,23].

Equivalent Uniform Dose (EUD): Recently, some of the medical physics literature has been advocating the use of other forms of objective function in place of the ones outlined above. A popular alternative to those given above is that of generalized equivalent uniform dose (EUD). This is defined on a per structure basis as

$$EUD_a(D,\Omega) := \left(\frac{1}{\text{card}\,(\Omega)}\sum_{(i,j,k)\in\Omega} D^a_{(i,j,k)}\right)^{\frac{1}{a}}.$$

Note that EUD is a scaled version of the a-norm of the dose to the particular structure, and hence is known to be a convex function for any $a \geq 1$ and concave for $a \leq 1$ [7]. Thus the problem

$$\begin{aligned}
\max_{w} \quad & EUD_a(D,\mathcal{T}) \\
\text{s.t.} \quad & D_\Omega = \sum_{A\in\mathcal{A}} \mathcal{D}_{\Omega,A} w_A, \quad \Omega = \mathcal{T}\cup S\cup\mathcal{N}\,, \\
& EUD_b(D,S) \leq \phi\,, \\
& EUD_c(D,\mathcal{N}) \leq u\,, \\
& 0 \leq w_A, \qquad \forall A\in\mathcal{A}\,.
\end{aligned}$$

is a convex optimization problem provided $a \leq 1$ and $b, c \geq 1$. As such, nonlinear programming algorithms will find global solutions to these problems.

Beam Angle Selection and Wedge Orientation Optimization: Optimization also lends itself to solving the more complex problem of selecting which angles and wedge orientations to use as well as their intensities. Mixed integer programming (MIP) is a straightforward technique for these type of problems. We describe an optimization model that simultaneously optimizes beam angles, wedge orientations, and beam intensities. Wedges are placed in front of the collimator to produce a gradient over the dose distribution and can be effective for reducing dose to organs at risk. This can be done by adding an extra dimension F to the variable w_A:

$$\begin{aligned}
\min_{w,\psi} \quad & \lambda_t f(\mathcal{D}_\mathcal{T}) + \lambda_s f(\mathcal{D}_S) + \lambda_n f(\mathcal{D}_\mathcal{N}) \\
\text{s.t.} \quad & D_\Omega = \sum_{A\in\mathcal{A}} \mathcal{D}_{\Omega,A,F} w_{A,F}\,, \\
& w_{A,F} \leq M\cdot\psi_A\,, \\
& l \leq D_\mathcal{T} \leq u\,, \\
& \sum_{a\in\mathcal{A}} \psi_a \leq K\,, \\
& \psi_A \in \{0,1\},\ \forall A\in\mathcal{A}\,.
\end{aligned} \tag{4}$$

The variable ψ_A is used to determine whether or not to use an angle A for delivery. The choice of M plays a critical role in the speed of the optimization; further advice on its choice is given in [27]. Note that the data for this problem is considerably larger, increasing by a factor related to the number of wedge orientations allowed.

Solution Techniques Simulated Annealing (SA) has been well adopted in the medical community [33,44]. But the weakness of SA in the optimization point is its inability to verify the optimality. On the other hand, it is possible to find a global optimal solution for (4). However, due to its slow convergence, using the MIP model has not been very useful for designing a treatment plan in the hospital. Recently, Lim et al. [27] proposed an iterative solution approach that solves the MIP problem fast (within 20 min in two clinical case examples). It is termed *A Three-Phase Approach*.

Three-Phase Approach is a multiphase technique that "ramps up" to the solution of the full problem via a sequence of models. Essentially, the models are solved in increasing order of difficulty, with the solution of one model providing a good starting point for the next. The models differ from each other in the selection of voxels included in the formulation, and in the number of beam angles allowed.

If the most promising beam angles can be identified in advance, the full problem can be solved with a small number of discrete variables. One simple approach for removing unpromising beam angles is to remove from consideration those that pass directly through any OAR [38]. A more elaborate approach [34] introduces a score function for each candidate angle, based on

the ability of that angle to deliver a high dose to the PTV without exceeding the prescribed dose tolerance to OAR or to normal tissue located along its path. Only beam angles with the best scores are included in the model. We now describe the *Three-Phase Approach:*

1. *Phase 1: Selection of Promising Beam Angles:* The aim in this phase is to construct a subset of beam angles $\mathcal{A}_1$ that are likely to appear in the final solution of (4). We solve a collection of r MIPs, where each MIP is constructed from a reduced set of voxels consisting of the voxels in the PTV, a randomly sampled 10% of the OAR voxels (S), and the voxels in $\mathcal{R}_\rho(\mathcal{T})$; that is,

$$\Omega_1 = \{\mathcal{T} \cup S \cup \mathcal{R}_\rho(\mathcal{T})\}.$$

   We define $\mathcal{A}_1$ as the set of all angles $A \in \mathcal{A}$ for which $w_A > 0$ for at least one of these r sampled problems.
2. *Phase 2: Treatment Beam Angle Determination:* In the next phase, we select K or fewer treatment beam angles from $\mathcal{A}_1$. We solve (4) using $\mathcal{A}_1$ and a reduced set of voxels defined as follows:

$$\Omega_2 = \{\mathcal{T} \cup S \cup \mathcal{R}_\rho(\mathcal{T}) \cup \mathcal{N}_1\}.$$

   Note that $|\mathcal{A}_1|$ is typically greater than or equal to K, so the binary variables play a nontrivial role in this phase.
3. *Phase 3: Final Approximation:* In the final phase, we fix the K beam angles (by fixing $\psi_{\mathcal{A}_1} = 1$ for the angles selected in Phase 2 and $\psi_A = 0$ otherwise) and solve the resulting simplified optimization problem over the complete set of voxels. This final approximation typically takes much less time to solve than the full-scale model, because of both the smaller amount of data (due to fewer beam angles) and the absence of binary variables.

Although there is no guarantee that this technique will produce the same solution as the original full-scale model (4), Lim et al. [27] have found that the quality of its approximate solution is close to optimal based on several numerical experiments.

Intensity Modulated Radiation Therapy

Introduction A sophisticated form of treatment planning approach known as intensity modulated radiation therapy (IMRT) allows a number of differently shaped beams with different uniform radiation intensities to be delivered from each direction, which allows a high degree of flexibility in delivering radiation dose distribution from each beam angle [4,18]. In IMRT treatment planning, two-dimensional (2D) beams are divided into several hundred or thousand pencil beams to generate very precise dose distribution on the treatment volume.

Decision Variables: First, one needs to decide how many beam angles need to be coordinated for the treatment (*beam angle optimization*). For each beam angle, radiation is delivered using a multi-leaf collimator (MLC). In practice, an MLC is designed so that one leaf can only move one direction with a discrete distance. Therefore, we divide an MLC as an $M \times N$ grid of pixels. M is for the number of leaves in an MLC (note that this number can vary from one manufacturer to another), and N is for the number of discrete units that a leaf can move. Second, radiation intensity maps (fluence maps) for such beam angles need to be optimized to conform the three dimensional radiation dose requirement to control the tumor (*fluence map optimization*). For a fixed beam angle, the fluence map contains real numbers in a set of two-dimensional discrete coordinates that are associated with the MLC. Since no machine can deliver such a non-uniform real intensity map, the intensity maps are first approximated as multiples of a physically deliverable minimum discrete unit (this number can be a fraction). For example, an approximated intensity map for a 3×4 MLC may look as follows (we assume that the minimum discrete value allowed is 0.5 in this case):

$$\begin{aligned} \mathcal{W} &= \begin{pmatrix} 0 & 0.5 & 2.0 & 1.5 \\ 0.5 & 2.5 & 3.5 & 2.0 \\ 0 & 1.0 & 2.0 & 1.5 \end{pmatrix} \\ &= 0.5 \times \begin{pmatrix} 0 & 1 & 4 & 3 \\ 1 & 5 & 7 & 4 \\ 0 & 2 & 4 & 3 \end{pmatrix}. \end{aligned} \tag{5}$$

Third, since we cannot deliver non-uniform radiation (see (5)) to the treatment volume with one open beam shape, an intensity map is decomposed into several unique shape matrices such that each matrix can contain zeros and uniform value. This is called *beam segmentation problem.*

Optimization Model Formulations

Problem 1; Beam Angle and Fluence Map Optimization Since the optimal set of beam angles are intimately related to the optimal fluence map for each angle, these two problems must be dealt with together in the problem formulation. Let a denote an angle, $a \in \mathcal{A}$, l denote the leaf index of the collimator, $l = 1, 2, \ldots, m$, and p represent the position of the leaf, $p = 1, 2, \ldots, n$. Then, formulating an optimization model for optimizing both beam angles and the fluence maps is a simple extension to (4) that we discussed for the conventional 3DCRT and the model is given as

$$\begin{aligned} \min_{w,\psi} \quad & \lambda_t f(\mathcal{D}_\mathcal{T}) + \lambda_s f(\mathcal{D}_\mathcal{S}) + \lambda_n f(\mathcal{D}_\mathcal{N}) \\ \text{s.t.} \quad & D_\Omega = \sum_{a\in\mathcal{A}} \mathcal{D}_{\Omega,a,l,p} w_{a,l,p} \,, \\ & w_{a,l,p} \le M \cdot \psi_a \,, \\ & l \le D_\mathcal{T} \le u \,, \\ & \sum_{a\in\mathcal{A}} \psi_a \le K \,, \\ & \psi_a \in \{0,1\}, \ \forall a \in \mathcal{A} \,. \end{aligned} \tag{6}$$

See more details of this formulation and others in [26,28].

Solution Methods Solving this problem using any classical optimization techniques will take too long for any clinicians to use for their daily treatment planning. Lim et al. [28] proposes a fast MIP solution approach and an LP-based iterative method that exploits score functions. However, due to the computational difficulties with large data in solving the optimization problem, heuristic methods are often used in practice [25].

Problem 2; Beam Segmentation Optimization Consider a matrix

$$\mathcal{W} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ \vdots & & \vdots & \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{pmatrix},$$

where $w_{i,j} \in \mathbb{Z}$, for $i = 1, \ldots, m$, $j = 1, \ldots, n$. Our objective is to decompose the matrix $\mathcal{W}$ into K binary matrices S^k such that

$$\mathcal{W} = \sum_{k=1}^{K} \mu_k \cdot S^k \,, \tag{7}$$

where, $S^k = \left[s^k_{i,j}\right]$, $s^k_{i,j} \in \{0,1\}$, $i \in \{1,2,\ldots,m\}$, $j \in \{1,2,\ldots,n\}$, $\mu_k \in \mathbb{Z}$, $k \in \{1,2,\ldots,K\}$. Solving (7) is quite easy in general. However, this problem becomes extremely difficult to solve when we impose the following two objectives and a physical constraint.

Objectives:

1. Minimize the value of K.
2. Minimize $T :=$ the sum of the matrix multipliers, i. e., $T = \sum_{k=1}^{K} \mu_k$.

Consecutive One Constraint: For each row of a binary matrix S_k, if there are more than one non-zero elements (1's), their sequence must be consecutive, i. e. zeros are not allowed to break the non-zero sequence. For example,

0 1 1 1 0

is a feasible sequence. But,

0 1 0 1 0

is not allowed because the sequence of ones is not continuous.

Note that there can be many more constraints to this problem depending on the machine that is used for radiation delivery. Some of the common constraints are *overlap elimination constraint, interleaf collision constraint,* and *tongue-and-groove constraint.* Details about these more elaborate constraints can be found in [1,19,20,43].

Solution Methods This is a combinatorial optimization problem that is proven to be *strongly NP-hard* [9]. Optimization formulations have been proposed including integer nonlinear program (INLP) and integer programming (IP) [28]. IP models are easier to solve than INLP models. IP models with relatively small numbers of rows and columns can be solved within a reasonable amount of time using a branch-and-bound method [45]. However, as the matrix size increases (say, larger than 10) and the maximum value of the matrix $\mathcal{W}$ increases, finding global solutions for the IP models can take too long for treatment planners to use. Therefore, both researchers and planners use various heuristics. Engel [10] proposed a heuristic that generates optimal T, but K is still not optimal. Other approaches and extension to this problem can be found in [21]. A genetic algorithm has also been used by other researchers [8].

References

1. Alfredo R, Siochi C (1999) Minimizing static intensity modulation delivery time using an intensity solid paradigm. Int J Radiat Oncol Biol Med 43(3):671–680
2. American Cancer Society (2008) Cancer facts and figures 2008. www.cancer.org (4/3/2008)
3. Bahr GK, Kereiakes JG, Horwitz H, Finney R, Galvin J, Goode K (1968) The method of linear programming applied to radiation treatment planning. Radiology 91:686–693
4. Bortfeld T, Boyer AL, Schlegel W, Kahler DL, Waldron TJ (1994) Realization and verification of three-dimensional conformal radiotherapy with modulated fields. Int J Radiat Oncol Biol Phys 30(4):899–908
5. Bortfeld T, Schlegel W (1993) Optimization of beam orientations in radiation therapy: some theoretical considerations. Phys Med Biol 38(2):291–304
6. Chen Y, Michalski D, Houser C, Galvin JM (2002) A deterministic iterative least-squares algorithm for beam weight optimization in conformal radiotherapy. Phys Med Biol 47:1647–1658
7. Choi B, Deasy JO (2002) The generalized equivalent uniform dose function as a basis for intensity-modulated treatment planning. Phys Med Biol 47:3579–3589
8. Cotrutz C, Xing L (2003) Segment-based dose optimization using a genetic algorithm. Phys Med Biol 48:2987–2998
9. Ehrgott M, Baatar D, Hamacher HW, Woeginer GJ (2005) Decomposition of integer matrices and multileaf collimator sequencing. Discret Appl Math 152:6–34
10. Engel K (2005) A new algorithm for optimal multileaf collimator field segmentation. Discret Appl Math 152:35–51
11. Fang G, Geiser B, Mackie TR (1997) Software system for UW/GE tomotherapy prototype. In: Leavitt DD, Starkshall G (eds) Proceedings of the 12th International Conference on the Use of Computers in Radiation Therapy, Salt Lake City, Medical Physics Publishing, St. Louis, pp 332–334
12. Ferris MC, Lim J-H, Shepard DM (2003) Optimization approaches for treatment planning on a Gamma Knife. SIAM J Optim 13:921–937
13. Ferris MC, Lim J-H, Shepard DM (2003) Radiosurgery treatment planning via nonlinear programming. Ann Oper Res 119:247–260
14. Ganz JC (1997) Gamma Knife Surgery. Springer, Wien
15. Goitein M, Abrams M, Rowell S, Pollari H, Wiles J (1983) Multi-dimensional treatment planning: II. beam's eye-view, back projection, and projection through ct sections. Int J Radiat Oncol Biol Phys 9:789–797
16. Ho JK (1985) Relationships among linear formulations of spearable convex piecewise linear programs. Math Program Study 24:126–140
17. Holder A (2004) Radiotherapy treatment design and linear programming. In: Brandeau ML, Saintfort F, Pierskalla WP (eds) Operations Research And Health Care: A Handbook of Methods and Applications. Kluwer, Boston, pp 741–774
18. Intensity Modulated Radiation Therapy Collaborative Working Group (2001) Intensity-modulated radiotherapy: Current status and issues of interest. Int J Radiat Oncol Biol Phys 51(4):880–914
19. Jordan TJ, Williams PC (1994) The design and performance characteristics of a multileaf collimator. Phys Med Biol 39:231–251
20. Kalinowski T (2005) Realization of intensity modulated radiation fields using multileaf collimators. Electron Notes Discret Math 21:319–320
21. Kalinowski T (2005) A duality based algorithm for multileaf collimator field segmentation with interleaf collision constraint. Discret Appl Math 152:52–88
22. Kapatoes JM, Olivera GH, Balog JP, Keller H, Reckwerdt PJ, Mackie TR (2001) On the accuracy and effectiveness of dose reconstruction for tomotherapy. Phys Med Biol 46:943–966
23. Kontogiorgis S (2000) Practical piecewise-linear approximation for monotropic optimization. INFORMS J Comput 12(4):324–340
24. Langer M, Leong J (1987) Optimization of beam weights under dose-volume restriction. Int J Radiat Oncol Biol Phys 13:1255–1260
25. Langer M, Morrill S, Brown R, Lee O, Lane R (1996) A comparison of mixed integer programming and fast simulated annealing for optimized beam weights in radiation therapy. Med Phys 23:957–964 (1996)
26. Lee EK, Fox T, Crocker I (2006) Simultaneous beam geometry and intensity map optimization in intensity-modulated radiation therapy. Int J Radiat Oncol Biol Phys 64(1):301–320
27. Lim GJ, Ferris MC, Wright SJ, Shepard DM, Earl MA (2007) An optimization framework for conformal radiation treatment planning. INFORMS J Comput 19(3):366–380
28. Lim GJ, Choi J, Mohan R (2008) Iterative solution methods for beam angle and fluence map optimization in Intensity Modulated Radiation Therapy Planning. OR Spectrum 30(2):289–309
29. Lim J-H (2002) Optimization in Radiation Treatment Planning. PhD thesis, University of Wisconsin
30. Lim J-H, Ferris MC, Shepard DM (2004) Optimization tools for radiation treatment planning in matlab. In: Brandeau ML, Saintfort F, Pierskalla WP (eds) Operations Research And Health Care: A Handbook of Methods and Applications. Kluwer, Boston, pp 775–806
31. Lodwick W, McCourt S, Newman F, Humphries S (1999) Optimization methods for radiation therapy plans. In: Borgers C, Natterer F (eds) Computational Radiology and Imaging: Therapy and Diagnosis, IMA Series in Applied Mathematics. Springer, Berlin
32. Morrill S, Lane R, Wong J, Rosen II (1991) Dose-volume considerations with linear programming. Med Phys 6(18):1201–1210
33. Morrill SM, Lam KS, Lane RG, Langer M, Rosen II (1995) Very fast simulated annealing in radiation therapy treatment

plan optimization. Int J Radiat Oncol Biol Phys 31:179–188
34. Pugachev A, Xing L (2002) Incorporating prior knowledge into beam orientation optimization in IMRT. Int J Radiat Oncol Biol Phys 54:1565–1574
35. Redpath AT, Vickery BL, Wright DH (1976) A new technique for radiotherapy planning using quadratic programming. Phys Med Biol 21:781–791
36. Romeijn HE, Ahuja RK, Dempsey JF, Kumar A (2006) A new linear programming approach to radiation therapy treatment planning problems. Oper Res 54(2):201–216
37. Rosen II, Lane R, Morrill S, Belli J (1990) Treatment plan optimization using linear programming. Med Phys 18(2):141–152
38. Rowbottom CG, Khoo VS, Webb S (2001) Simultaneous optimization of beam orientations and beam weights in conformal radiotherapy. Med Phys 28(8):1696–1702
39. Shepard DM, Chin LS, DiBiase SJ, Naqvi SA, Lim J, Ferris MC (2003) Clinical implementation of an automated planning system for Gamma Knife radiosurgery. Int J Radiat Oncol Biol Phys 56:1488–1494
40. Shepard DM, Ferris MC, Olivera G, Mackie TR (1999) Optimizing the delivery of radiation to cancer patients. SIAM Rev 41:721–744
41. Starkschall G (1984) A constrained least-squares optimization method for external beam radiation therapy treatment planning. Med Phys 11:659–665
42. Tervo J, Kolmonen P (2000) A model for the control of a multileaf collimator in radiation therapy treatment planning. Inverse Problems 16:1875–1895
43. Kung J Que W, Dai J (2004) Tongue-and-groove effect in intensity modulated radiotherapy with static multileaf collimator fields. Phys Med Biol 49:399–405
44. Webb S (1989) Optimisation of conformal radiotherapy dose distributions by simulated annealing. Phys Med Biol 34(10):1349–1370
45. Wolsey LA (1998) Integer Programming. Wiley, New York
46. Wu X, Zhu Y (2001) A global optimization method for three-dimensional conformal radiotherapy treatment planning. Phys Med Biol 46:109–119
47. Xiao Y, Censor Y, Michalski D, Galvin JM (2003) The least-intensity feasible solution for aperture-based inverse planning in radiation therapy. Ann Oper Res 119:183–203
48. Yoda K, Saito Y, Sakamoto H (1997) Dose optimization of proton and heavy ion therapy using generalized sampled pattern matching. Phys Med Biol 42:2411–2420

Optimization-Based Visualization

ANTANAS ŽILINSKAS, JULIUS ŽILINSKAS
Institute of Mathematics and Informatics, Vilnius, Lithuania

MSC2000: 91C15, 65K05, 90C30, 90C27, 90C57

Article Outline

Keywords and Phrases
Introduction
Formulation
Methods/Applications
See also
References

Keywords and Phrases

Global optimization; Unidimensional scaling; Multidimensional scaling

Introduction

Very often scientific data are defined by multidimensional vectors of numerical values. To enable exploratory data analysis involving heuristic abilities of human expert's visualization of data is highly desirable: a picture is worth a thousand words. There are different approaches to visualization [8]. We consider one of the most popular approaches known as multidimensional scaling (MDS) [2,9,14,27,31,39,42]; it will be shown below that an essential part of the technique is optimization of a function possessing many optimization-adverse properties. By means of MDS a set of multidimensional vectors can be represented as a set of points in a low-dimensional space and exposed in this way to a human expert for heuristic analysis. Even more general sets of objects can be visualized: it is sufficient to know pairwise similarity/dissimilarity between the objects. Application areas of MDS vary from psychometrics [41] and market analysis [15,36] to mobile communications [22] and pharmacology [45].

Formulation

A set of n *objects* is considered whose pairwise *dissimilarities* are given by an $(n \times n)$ matrix (δ_{ij}), $i, j = 1, \ldots, n$. It is supposed that dissimilarities are nonnegative: $\delta_{ij} \geq 0$, symmetric: $\delta_{ij} = \delta_{ji}$, and $\delta_{ii} = 0$. Frequently the considered objects are vectors, and dissimilarities are defined by a metric in the corresponding vector space. Sometimes (a reciprocal to dissimilarity) the proximity relation between objects is defined; this

case can be considered similarly to the case of dissimilarity relation.

An image of a set of objects is sought as a set of points $\mathbf{x}_i \in \mathbf{R}^m$, $i = 1, \ldots, n$ in a (low-dimensional metric) *embedding* space with pairwise distances between the image points fitting the corresponding dissimilarities; normally *Minkowski distances* $d_p(\mathbf{x}_i, \mathbf{x}_j)$ between the points $\mathbf{x}_i$ and $\mathbf{x}_j$ are used where

$$d_p(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{k=1}^{m} \left| x_{ik} - x_{jk} \right|^p \right)^{1/p} .$$

The formula defines Euclidean distances when $p = 2$ and city-block distances when $p = 1$.

The problem of constructing images of the considered objects is reduced to the minimization of an accuracy-of-fit criterion, e. g., of a least-squares *stress* function,

$$\sigma_r(\mathbf{X}) = \sum_{i<j} w_{ij} \left(d\left(\mathbf{x}_i, \mathbf{x}_j\right) - \delta_{ij} \right)^2 ,$$

where *weights* w_{ij} are nonnegative: $w_{ij} \geq 0$. *Normalized stress* defined by

$$\sigma_n(\mathbf{X}) = \frac{\sum_{i<j} w_{ij} \left(d\left(\mathbf{x}_i, \mathbf{x}_j\right) - \delta_{ij} \right)^2}{\sum_{i<j} w_{ij} \delta_{ij}^2}$$

shows the proportion of unexplained sum of squares and can be used for comparison of results of different problems. Stress is not everywhere differentiable. The function normally has many local minima. It is invariant with respect to translation, rotation, and mirroring. The minimization problem of stress is high dimensional: the number of variables is $N = n \times m$. Therefore minimization of the stress function is a difficult global optimization problem.

Many global optimization methods for minimization of stress include auxiliary local minimization algorithms. Differentiability of an objective function at a minimizer is an important factor for a proper choice of a local minimization algorithm. The well-known result on differentiability of stress with Euclidean distances at a local minimizer [12] is generalized for Minkowski distances in [21]: *positiveness of distances* holds at a local minimizer – image points in the embedding space do not coincide. In the case of Minkowski distances $p > 1$ or $m = 1$, this means that stress is differentiable at a local minimizer.

The result on differentiability of stress with Minkowski distances at a local minimizer [21] does not include the case of city-block distances ($p = 1$). It was shown in [44] that positiveness of distances at a local minimizer does not imply differentiability of stress with city-block distances. Examples of images at minimizers show that values of coordinates of image points in the embedding space $m > 1$ may be equal and therefore stress may be nondifferentiable at minimizer in the case of city-block distances.

Everywhere differentiable *S-stress* is defined by

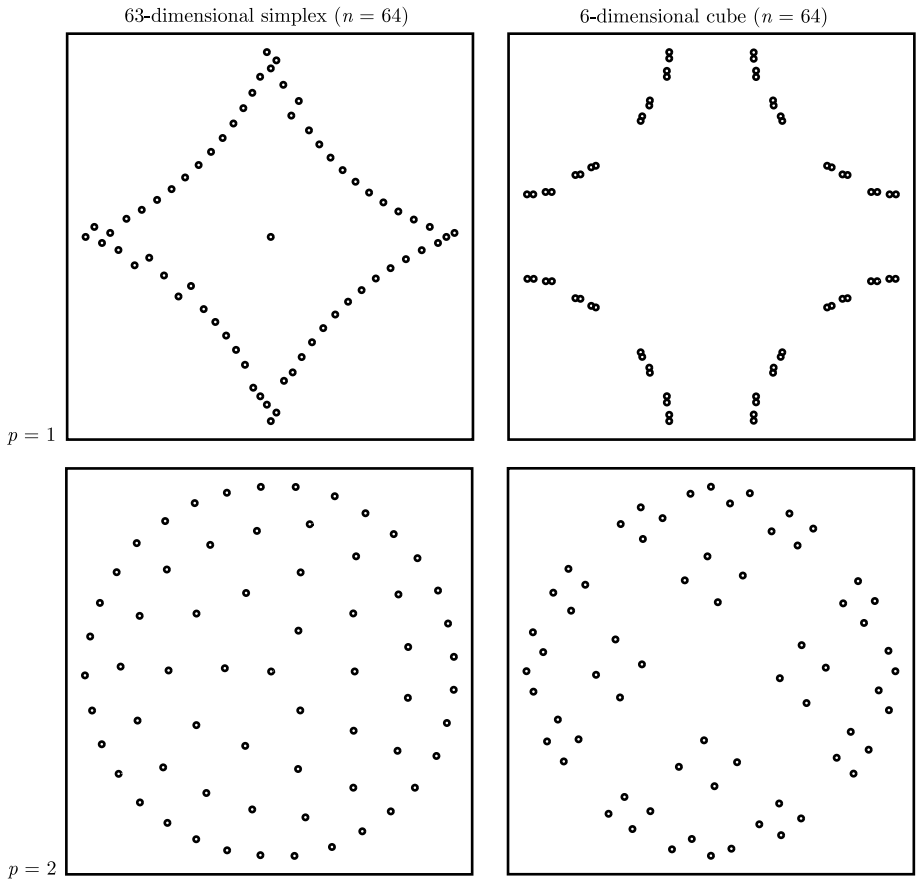
$$S_S(\mathbf{X}) = \sum_{i<j} w_{ij} \left(d^2\left(\mathbf{x}_i, \mathbf{x}_j\right) - \delta_{ij}^2 \right)^2 .$$

Sometimes instead of the least-squares stress, a *least absolute deviation* (L_1-norm) function is used:

$$S_{L1}(\mathbf{X}) = \sum_{i<j} w_{ij} \left| d\left(\mathbf{x}_i, \mathbf{x}_j\right) - \delta_{ij} \right| .$$

Examples of two-dimensional images produced using minimization of stress with city-block and Euclidean distances are shown in Fig. 1. Vertices of multidimensional geometrical figures are considered as objects to be visualized. The dissimilarity between vertices is measured by the distance in the original vector space. Although it is not possible to imagine geometrical figures in the space of dimensionality larger than 3, properties of well-understood geometrical figures are known. Multidimensional simplices and cubes are special on the symmetric location of vertices, and this feature is expected in the images. Besides this common feature, the central location of the "zero" vertex is characteristic of a multidimensional simplex. The other vertices of a simplex are equally distant from the "zero" vertex. The vertices of a multidimensional cube compose clusters of 2^k vertices corresponding to edges and faces. The vertices of a cube are equally distant from the center.

The images of a 63-dimensional simplex are shown in the left column of Fig. 1, and the images of a 6-dimensional cube are shown in the right column. Both geometrical figures have $n = 64$ vertices. The images produced by city-block MDS are shown in the upper row, and the images produced by Euclidean MDS are shown in the lower row. The images of vertices are shown by circles.



Optimization-Based Visualization, Figure 1
Images of vertices of multidimensional simplices and cubes

The image of the "zero" vertex of the simplex is located at the center of the structures. The images of the other vertices tend to form a square with a vertical diagonal when city-block distances are used and circles when Euclidean distances are used. A square with a vertical diagonal in city-block metric is equivalent to a circle in Euclidean metric – the points on such a figure are equally distant to the center. Therefore the images of the vertices are similarly distant to the "zero" vertex when city-block distances are used. The images of vertices on different circles are differently distant to the "zero" vertex when Euclidean distances are used.

The images of the vertices of the multidimensional cube tend to form a square with a vertical diagonal too when city-block distances are used; therefore they are visualized similarly distant to the center of the image. In the case of Euclidean metric, the images of the vertices of the cube tend to fill a circle; however, there is no uniformity in the location of the images of the vertices. The images of the vertices form clusters representing lower-dimensional cubes in the cases of both metrics.

Methods/Applications

MDS is a generalization of unidimensional scaling (UDS) ($m = 1$) [33] to the multidimensional case ($m > 1$).

Minimization of the stress function with equal weights for $m = 1$ can be changed to a combinatorial maximization problem [10]:

$$\max_{\boldsymbol{\psi} \in \boldsymbol{\Psi}} \sum_{i=1}^{n} \left(\sum_{j>i} \delta_{\boldsymbol{\psi}(i)\boldsymbol{\psi}(j)} - \sum_{j<i} \delta_{\boldsymbol{\psi}(i)\boldsymbol{\psi}(j)} \right)^2,$$

where $\boldsymbol{\Psi}$ is the set of all possible permutations of $1, \ldots, n$. The optimal permutation $\boldsymbol{\psi}^*$ found using maximization defines the optimal sequence of objects.

Then the coordinate values of image points are found by

$$
\begin{aligned}
x_{\psi^*(1)} &= 0, \\
x_{\psi^*(i+1)} &= x_{\psi^*(i)} + \frac{1}{n}\left(\sum_{j>i} \delta_{\psi^*(i)\psi^*(j)}\right. \\
&\quad - \sum_{j<i} \delta_{\psi^*(i)\psi^*(j)} - \sum_{j>i+1} \delta_{\psi^*(i+1)\psi^*(j)} \\
&\quad \left. + \sum_{j<i+1} \delta_{\psi^*(i+1)\psi^*(j)}\right), \\
&\quad i = 1, \ldots, n-1 .
\end{aligned}
$$

The number of local minima for the problem of UDS was estimated in [37]. There a "smoothing technique" approach was presented to locate the globally optimal solution.

A branch-and-bound method for obtaining the guaranteed globally optimal solution to the problems of UDS was presented in [7]. An interchange test and new bounding procedures were used to improve computational performance.

Guaranteed solution of larger problems is not possible; therefore heuristic approaches are used. A simulated annealing (SA) approach for the problem of UDS via maximization of the Defays criterion was presented in [5]. This algorithm includes efficient storage and computation methods to facilitate rapid evaluation of trial solutions.

Quadratic assignment methods to generate initial permutations for UDS were developed in [6]. Methods include locally optimal pairwise interchange, SA, and hybrid. It was shown that substantial improvements of UDS can be achieved using starting permutations obtained via solution to a quadratic assignment problem.

A heuristic algorithm based on SA for provision of good starting solutions for combinatorial algorithms is proposed in [3]. The heuristic starts with the partition of equally spaced discrete points. A SA algorithm is used to search the lattice defined by these points with the objective of minimizing least-squares or least absolute deviation loss function.

An algorithm implementing SA for UDS in a different way is presented in [35]. A strategy is based on a weighted alternating process: permutations and pointwise translations are used to locate the optimal configuration.

A recursive dynamic programming strategy for some problems including UDS is discussed in [23]. Four different optimization strategies for UDS have been compared in [24]: dynamic programming, iterative quadratic assignment heuristic, smoothing technique [37], and nonlinear programming reformulation [28]. The results show that the first two strategies are better than the other two and should lead to optimal solutions if some random starts are used.

A mixed-integer programming formulation for the least absolute deviation UDS is developed in [40]. Integer linear programming models for UDS were discussed in [4]. In the case of least absolute deviation UDS, the objective function is piecewise linear.

The special geometry of squared error loss function for UDS is employed in [38]. The developed algorithm is linear in the number of parameters, as the global minimum for every coordinate is conditioned on every other coordinate being held fixed.

One of the most popular algorithms for MDS is SMACOF [11]. The algorithm is based on a majorization approach [17] that replaces iteratively the original objective function by an auxiliary majorization function, which is much simpler to optimize. The convergence properties of MDS algorithms are studied in [13]. It was proved that the majorization method is globally convergent. In almost all cases the convergence is linear, with a convergence rate close to unity. The majorization algorithm has been extended to deal with Minkowski distances with $1 \leq p \leq 2$, and an algorithm that is partially based on majorization for p outside this range is suggested in [21].

A tunneling method for global minimization was introduced and adjusted for MDS with general Minkowski distances in [18]. The tunneling method alternates a local search step, in which a local minimum is sought, with a tunneling step, in which a different configuration is sought with the same value of stress as the previous local minimum. In this manner successively better local minima are obtained and the last one is often the global minimum.

A method for MDS based on combining a local search algorithm with an evolutionary strategy of generating new initial points was proposed in [32]. Its efficiency is investigated by numerical experiments. The

testing results in [22,30] proved that the hybrid algorithm combining an evolutionary global search with an efficient local descent is the most reliable, though the most time-consuming method for MDS with Euclidean distances. The advantages of genetic algorithms in MDS with nonstandard stress criteria were discussed in [16].

The concept of sequential estimation in MDS was introduced in [34]. The sequential estimation method refers to continually updating estimates of a configuration as new observations are added. Locally optimal design of the experiment was constructed.

A globalized Newton method for stress and S-stress is developed in [25]. A deterministic annealing algorithm for the S-stress is presented in [26] and experimentally compared with gradient-descent methods.

General methods for MDS can be applied in the case of city-block distances if they do not rely on the differentiability of the objective function at a minimizer. However, there are some methods developed especially for city-block MDS.

A survey of city-block MDS is presented in [1]. Topics include theoretical issues, algorithmic developments and their implications for seemingly straightforward analyses, isometries with other distances, and links to graph-theoretic models.

A distance smoothing approach for city-block MDS was proposed in [19]. The technique allows avoiding of local minima in optimization. The technique was extended to any Minkowski distance, and a majorization algorithm with a monotone nonincreasing series of stress values was suggested in [20].

A heuristic algorithm based on SA for two-dimensional city-block scaling was proposed in [3]. The heuristic starts with the partitioning of each coordinate axis into equally spaced discrete points. A SA algorithm is used to search the lattice defined by these points with the objective of minimizing least-squares or least absolute deviation loss function. The object permutations for each dimension of the solution obtained by the SA algorithm are used to find a locally optimal set of coordinates by quadratic programming.

A two-stage approach for city-block MDS was proposed in [29]. The least-squares regression is used to obtain a local minimum of stress function in the first stage. SA is used in the second stage of the method.

A bilevel method for city-block MDS was proposed in [44]. The method employs a piecewise quadratic structure of stress with city-block distances reformulating the global optimization problem as a two-level optimization problem:

$$\min_{\mathbf{P}} S(\mathbf{P}) ,$$
$$\text{s.t. } S(\mathbf{P}) = \min_{\mathbf{X} \in A(\mathbf{P})} S(\mathbf{X}) ,$$

where the upper-level combinatorial problem is defined over the set of all possible permutations of $1, \ldots, n$ for each coordinate of the embedding space and the lower-level problem is a quadratic programming problem with a positively defined quadratic objective function and linear constraints setting the sequences of values of coordinates defined by m permutations in $\mathbf{P}$. The lower-level problems are solved using a quadratic programming algorithm. The upper-level combinatorial problem can be solved by guaranteed methods for small n and using evolutionary search for larger problems.

Interaction between optimization and visualization means not only application of optimization methods to implement visualization algorithms but also application of visualization methods to analyze properties of optimization problems. For example, optimal design of the chemical engineering process considered in [43] includes a minimization problem in nine-dimensional space where the feasible region is defined by interval constraints, and a nonexplicit (black box) indicator-type constraint. Properties of the feasible region are of interest while choosing the optimization algorithm and while making a final decision about process parameters. The properties of interest cannot be proven analytically, but heuristic analysis of the image of the set of points consisting of vertices of the nine-dimensional cube and 300 randomly generated points of the feasible region is helpful to guess the form and dimensionality of the region and its location in the hypercube.

See also

- ► Continuous Global Optimization: Models, Algorithms and Software
- ► Dynamic Programming in Clustering
- ► Evolutionary Algorithms in Combinatorial Optimization
- ► Integer Programming
- ► Integer Programming: Branch and Bound Methods

► Nonlinear Least Squares: Newton-type Methods
► Simulated Annealing

References

1. Arabie P (1991) Was Euclid an unnecessarily sophisticated psychologist? Psychometrika 56:567–587
2. Borg I, Groenen PJF (2005) Modern Multidimensional Scaling: Theory and Applications, 2nd edn. Springer, New York
3. Brusco MJ (2001) A simulated annealing heuristic for unidimensional and multidimensional (city-block) scaling of symmetric proximity matrices. J Classif 18:3–33
4. Brusco MJ (2002) Integer programming methods for seriation and unidimensional scaling of proximity matrices: a review and some extensions. J Classif 19:45–67
5. Brusco MJ (2006) On the performance of simulated annealing for large-scale L_2 unidimensional scaling. J Classif 23:255–268
6. Brusco MJ, Stahl S (2000) Using quadratic assignment methods to generate initial permutations for least-squares unidimensional scaling of symmetric proximity matrices. J Classif 17:197–223
7. Brusco MJ, Stahl S (2005) Optimal least-squares unidimensional scaling: improved branch-and-bound procedures and comparison to dynamic programming. Psychometrika 70:253–270
8. Chen C (2003) Mapping Scientific Frontiers: The Quest for Knowledge Visualization. Springer, London
9. Cox TF, Cox MAA (2001) Multidimensional Scaling, 2nd edn. Chapman & Hall/CRC, Boca Raton
10. Defays D (1978) A short note on a method of seriation. Br J Math Statist Psychol 31:49–53
11. de Leeuw J (1977) Applications of convex analysis to multidimensional scaling. In: Barra JR, Brodeau F, Romier G, van Cutsem B (eds) Recent Developments in Statistics. North-Holland, Amsterdam, pp 133–145
12. de Leeuw J (1984) Differentiability of Kruskal's stress at a local minimum. Psychometrika 49:111–113
13. de Leeuw J (1988) Convergence of the majorization method for multidimensional scaling. J Classif 5:163–180
14. de Leeuw J, Heiser W (1982) Theory of multidimensional scaling. Handb Stat 2:285–316
15. DeSarbo WS, Kim Y, Wedel M, Fong DKH (1998) A Bayesian approach to the spatial representation of market structure from consumer choice data. Eur J Oper Res 111:285–305
16. Everett JE (2001) Algorithms for multidimensional scaling. In: Chambers LD (ed) The Practical Handbook of Genetic Algorithms: Applications, 2nd edn. Chapman & Hall/CRC, Boca Raton, pp 203–233
17. Groenen PJF (1993) The Majorization Approach to Multidimensional Scaling: Some Problems and Extensions. DSWO Press, Leiden
18. Groenen PJF, Heiser WJ (1996) The tunneling method for global optimization in multidimensional scaling. Psychometrika 61:529–550
19. Groenen PJF, Heiser WJ, Meulman JJ (1998) City-block scaling: smoothing strategies for avoiding local minima. In: Balderjahn I, Mathar R, Schader M (eds) Classification, Data Analysis, and Data Highways. Springer, Berlin, pp 46–53
20. Groenen PJF, Heiser WJ, Meulman JJ (1999) Global optimization in least-squares multidimensional scaling by distance smoothing. J Classif 16:225–254
21. Groenen PJF, Mathar R, Heiser WJ (1995) The majorization approach to multidimensional scaling for Minkowski distances. J Classif 12:3–19
22. Groenen P, Mathar R, Trejos J (2000) Global optimization methods for multidimensional scaling applied to mobile communication. In: Gaul W, Opitz O, Schander M (eds) Data Analysis: Scientific Modeling and Practical Applications. Springer, Berlin, pp 459–475
23. Hubert LJ, Golledge RG (1981) Matrix reorganization and dynamic programming: applications to paired comparisons and unidimensional seriation. Psychometrika 46:429–441
24. Hubert LJ, Arabie P, Meulman JJ (2002) Linear unidimensional scaling in the L_2-norm: basic optimization methods using MATLAB. J Classif 19:303–328
25. Kearsley AJ, Tapia RA, Trosset MW (1998) The solution of the metric STRESS and SSTRESS problems in multidimensional scaling using Newton's method. Comput Statist 13:369–396
26. Klock H, Buhmann JM (2000) Data visualization by multidimensional scaling: a deterministic annealing approach. Pattern Recognit 33:651–669
27. Kruskal JB, Wish M (1978) Multidimensional Scaling. Bell Laboratories, Murray Hill
28. Lau K, Leung PL, Tse K (1998) A nonlinear programming approach to metric unidimensional scaling. J Classif 15: 3–14
29. Leung PL, Lau K (2004) Estimating the city-block two-dimensional scaling model with simulated annealing. Eur J Oper Res 158:518–524
30. Mathar R (1997) A hybrid global optimization algorithm for multidimensional scaling. In: Klar R, Opitz O (eds) Classification and Knowledge Organization. Springer, Berlin, pp 63–71
31. Mathar R (1997) Multidimensionale Skalierung, Mathematische Grundlagen und Algorithmische Konzepte. Teubner, Leipzig
32. Mathar R, Žilinskas A (1993) On global optimization in two-dimensional scaling. Acta Applicandae Mathematicae 33:109–118
33. McIver JP, Carmines EG (1981) Unidimensional Scaling, Sage Publications, Newbury Park
34. Miyano H, Inukai Y (1982) Sequential estimation in multidimensional scaling. Psychometrika 47:321–336
35. Murillo A, Vera JF, Heiser WJ (2005) A permutation-translation simulated annealing algorithm for L_1 and L_2 unidimensional scaling. J Classif 22:119–138

36. Nelson TR, Rabianski J (1988) Consumer preferences in housing market analysis: an application of multidimensional scaling techniques. Real Estate Econ 16:138–159
37. Pliner V (1996) Metric unidimensional scaling and global optimization. J Classif 13:3–18
38. Poole KT (1990) Least squares metric, unidimensional scaling of multivariate linear models. Psychometrika 55:123–149
39. Schiffman SS, Reynolds ML, Young FW (1981) Introduction to Multidimensional Scaling: Theory, Methods, and Applications. Academic Press, London
40. Simantiraki E (1996) Unidimensional scaling: a linear programming approach minimizing absolute deviations. J Classif 13:19–25
41. Takane Y (2006) Applications of multidimensional scaling in psychometrics. Handb Stat 26:359–400
42. Torgerson WS (1958) Theory and Methods of Scaling. Wiley, New York
43. Žilinskas A, Fraga ES, Mackutė A (2006) Data analysis and visualisation for robust multi-criteria process optimisation. Comput Chem Eng 30:1061–1071
44. Žilinskas A, Žilinskas J (2007) Two level minimization in multidimensional scaling. J Global Optim 38:581–596
45. Žilinskas J (2006) Multidimensional scaling in protein and pharmacological sciences. In: Bogle IDL, Žilinskas J (eds) Computer Aided Methods in Optimal Design and Operations. World Scientific, Singapore, pp 139–148

Optimization in Boolean Classification Problems

Evangelos Triantaphyllou,
Jennifer Austin-Rodriguez
Department Industrial and Manufacturing Systems Engineering, Louisiana State University,
Baton Rouge, USA

MSC2000: 90C09, 90C10

Article Outline

Keywords
Background Information
Optimization Approaches
Concluding Remarks
See also
References

Keywords

Medical diagnosis; Inductive inference problem; Boolean classification problem; Learning algorithm; Conjunctive normal form; CNF; Disjunctive normal form; DNF; Satisfiability problem; SAT; Minimum number of clauses; One clause at a time approach; OCAT; GRASP approach; Randomized heuristics; Missing information; Unclassifiable examples

There are many situations in which it is necessary or desirable to classify objects into two or more mutually exclusive sets or classes.*Medical diagnosis*, for example, has been the focus of numerous research efforts over the past several years. Given a set of attributes, or relevant characteristics which describe a patient, the problem then is the extraction and identification of various biological and/or historical attributes in order to determine the correct diagnosis or classification.

To illustrate the magnitude of the problem, consider the case of *breast cancer diagnosis*. Based on a patient's medical history and on the results of *mammography screening* (the most effective diagnostic tool available to health care professionals), doctors attempt to classify *breast tumors* as being suspicious for malignancy or benign. Unfortunately, of all breast tumors which are suspected to be malignant, over 70% are later found to be benign through an expensive and emotionally trying surgical procedure called a biopsy [5]. In addition, almost 50% of those patients who actually have breast cancer are classified as benign by their physicians, so that many malignancies go unrecognized [27].

The decision maker, in this example the medical doctor, must infer from existing information the characteristics or combinations of characteristics which are indicative of a benign or malignant tumor in order to correctly classify new cases. In their most basic form, the characteristics used to describe each patient are represented by one or more binary attributes. That is, each object (patient) may be represented by a Boolean vector in which an attribute value is either 1 (true) or 0 (false). Often, the problem is compounded by the fact that complete information is not available. Continuing with the breast cancer example, suppose that the information related to all pertinent characteristics is not available due to the patient's inability to undergo certain tests because of excessive cost, the possibility of indeterminate test results, lack of knowledge

related to family histories, etc. Thus, in addition to the binary data indicating the presence or absence of a given characteristic, the attribute value or level of some characteristics may be unknown. The doctor is then faced with the problem of assessing a limited set of characteristics to determine whether a biopsy is warranted. He/She must decide if the available characteristics and/or combinations of characteristics provide sufficient information for an accurate classification of the tumor.

This problem, referred to as the *inductive inference problem* or *Boolean classification problem*, is illustrative of a vast number of similar situations throughout business, industry and medicine. Technological advances have created a 'data explosion', providing decision makers with ever increasing amounts of information. Unfortunately, this information is usually not exploited in an optimal way, and at times, not at all. Clearly, the classification problem becomes more complex as the amount of information related to the object increases. Individuals, or groups of individuals find themselves incapable of consistently and reliably handling, manipulating and analyzing the available information. As a result, the creation of computer systems capable of learning the concepts underlying the data and subsequently classifying new examples accurately and efficiently has become a practical necessity.

Background Information

As informally presented above, solving the Boolean classification problem generally involves the development of a system that learns from feature-based examples. That is, each example is described by a set of Boolean attributes. The binary vector [0 1 1 1], for instance, describes an example in which the first attribute (or characteristic) is false, and the remaining attributes are true. Each example also carries a classification: positive or negative. The goal of a *learning algorithm* is to infer from these examples a Boolean function (logical system) that is capable of accurately predicting the class of new examples. Generally, the inferred system is expressed as a Boolean function in *conjunctive normal form* (CNF) or *disjunctive normal form* (DNF).

The general form of a CNF and DNF Boolean function is defined as (1) and (2), respectively. That is:

$$\bigwedge_{j=1}^{k}\left(\bigvee_{i\in\rho_j} a_i\right) \tag{1}$$

and

$$\bigvee_{j=1}^{k}\left(\bigwedge_{i\in\rho_j} a_i\right), \tag{2}$$

where a_i is either A_i or $\overline{A}_i$. That is, a CNF expression is a conjunction of disjunctions, while a DNF expression is a disjunction of conjunctions. Any Boolean function can be transformed into CNF or DNF format [15].

To clarify the concepts presented thus far, suppose that the following sets of positive and negative examples are somehow known:

$$E^+ = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

and

$$E^- = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

The goal of a learning algorithm is to infer a Boolean function which correctly classifies all the examples. One such function (in CNF) is as follows:

$$(A_2 \vee A_4) \wedge (\overline{A}_2 \vee \overline{A}_3) \wedge (A_1 \vee A_3 \vee \overline{A}_4).$$

These three clauses, when are taken together, accept the previous four positive examples and reject the six negative examples.

Traditionally, there are two main methods with the goal of creating these intelligent systems: *decision trees* and *neural networks*. These tools, which have evolved over a forty-year period, represent a large portion of the literature on learning algorithms which propose to solve the Boolean classification problem. When applied

to classification problems in which the goal of the system is to learn from feature-based examples, decision trees have been one of the most popular methodologies for the extraction of knowledge. Because of the natural interpretation of knowledge, symbolic decision trees can be easily translated into a set of rules suitable for use in *rule-based systems*. The size and form of the decision tree is significantly affected by the ordering of the attributes, and often the resulting tree is nonoptimal or it may be overspecialized. A complex tree is not only more difficult to validate, but as J.R. Quinlan [16] demonstrated, a simpler tree is more likely to capture structures inherent in the data.

Neural networks comprise the other extreme in artificial intelligence approaches. These systems consist of a set of programs based on the structure of biological neural systems. Knowledge is represented in the form of a series of interconnected neurons, the structure of their interconnections, and the strength of their interconnections. To the user the process is a 'black box'. Though these systems have demonstrated the ability to provide accurate classifications in many applications, the examples are classified without explanations or justifications. In an attempt to overcome this deficiency, hybrid systems which combine neural networks and rule-based systems have been developed [4]. While these hybrid systems are efficient and effective in terms of both time and storage requirements, unfortunately, an exponential number of rules may be derived [17]. This renders attempts at justification of the process virtually useless due to the complexity of the explanation. The problem is that the logical rules are not derived within a complete logical framework.

Optimization Approaches

Recognizing the need to minimize the resulting system, the problem of inferring a Boolean system from positive and negative examples was formulated as a satisfiability problem (SAT) and a method for inferring a minimal DNF system was proposed [8]. The SAT problem is next translated into an integer programming (IP) problem that is then solved by using an *interior point method* developed in [9]. The method makes use of a parameter, say k, which preassumes the number of disjunctions in the DNF system to be derived. The IP problem, if feasible is solved and k is successively lowered until infeasibility is encountered and then it is concluded that there exists no system of size k or smaller which accepts all positive examples and rejects all negative examples. Many solution methods exist for solving the SAT problem (see, for instance, [6,7,8] and [26]). Unfortunately, trying to determine a minimum size Boolean function may be computationally very difficult since it is much harder to prove that a given SAT problem is infeasible than to prove it is feasible. Thus, while the SAT approach can be used with success on small data sets and a *minimal number of DNF clauses* thus to be derived, when dealing with real world data, minimizing the size of the system may be neither feasible nor desirable due to the vast amounts of time and storage required by such algorithms.

In [25] a logical (Boolean) function approach to the classification problem has been introduced with the *one clause at a time* (OCAT) approach. Like the SAT approach, the OCAT approach formulates the *Boolean classification problem* as a series of integer programming problems. The OCAT algorithm is sequential and greedy in nature. The first iteration takes as input the E^+ and E^- sets, and generates a single clause which accepts all positive examples and rejects as many negative examples as possible. This is the greedy aspect of the method. In the next iteration, it performs the same task using the original E^+ set and a revised E^- set which includes only those negative examples not rejected by the preceding CNF clause. The iterations continue until a set of clauses is constructed which rejects all the negative examples and, of course, each clause accepts all the positive examples. This algorithm is as follows:

	$i = 0;\ C = \emptyset;$
DO	WHILE ($E^- \neq \emptyset$)
1	$i \leftarrow i + 1;$
2	Find a clause c_i which accepts all members of E^+ while it reflects as many members of E^- as possible;
3	Let $E^-(c_i)$ be the set of members of E^- which are reflected by c_i;
4	Let $C \leftarrow C \cup c_i$;
5	Let $E^- \leftarrow E^- - E^-(c_i)$;
REPEAT;	

The one clause at a time (OCAT) approach (the CNF case)

The core of the method lies in step 2, the application of a branch and bound algorithm. Through the development of new search strategies and fathoming tests, E. Triantaphyllou [19] improved the performance of the branch and bound step. Still, like all branch and bound algorithms, it suffers from exponential time complexity. However, computational experiments indicate that the OCAT approach, when combined with the branch and bound algorithm, is a very efficient method for inferring logical clauses from sets of positive and negative examples. In fact, in over half of the test cases, this approach generated a minimum number of clauses. In addition, when compared to the SAT approach of [8], OCAT and the branch and bound was found to be considerably faster while performing at the same level of predictive accuracy [19]. Thus, while the OCAT approach may not always derive an absolute minimal system, computationally it is much less expensive that the SAT approaches and therefore more applicable to real world applications.

Continued research in this area resulted in the development of two *randomized heuristics* [2]. It should be stated here that this approach is similar, in principal, to the GRASP (greedy random adaptive search procedure) approach presented in [3]. The first heuristic (RA1) was developed to overcome the exponential time complexity of the OCAT's branch and bound algorithm. That is, RA1 derives a Boolean system from positive and negative examples in polynomial (quadratic) time. The primary difference between the branch and bound algorithm and the RA1 heuristic is that in each iteration, the branch and bound attempts to reject as many negative examples as possible; while RA1 attempts only to reject many negative examples. Again, the increased speed resulted in generally larger systems. When comparing the two algorithms, A.S. Deshpande and Triantaphyllou [2] found that the branch and bound used in the original OCAT approach produced in general, fewer conjunctions and required higher CPU times than the RA1 heuristic. Additionally, it was concluded that a conjunction of the RA1 heuristic and the branch and bound method performs much better in terms of both computational time/memory requirements of the process and the size of the derived system than either approach used alone.

Faced with real-world problems, in which there is often *incomplete information* related to both the attribute values and the classifications, the goal to optimize the system becomes more desirable and necessary. Each of the methods discussed thus far have considered only positive and negative examples with complete data. That is, there is no *missing information* in the data set. Often, the complete examples represent only a portion of the available data, since in general data bases are plagued by missing information. In [1] a logical method for deriving a Boolean function from positive and negative examples was introduced in which some of the attribute values may be unknown. Since the missing information did not inhibit the classification process, these attributes are treated as 'don 't care' values by the algorithm. Using a network flow algorithm, the method has been shown to efficiently derive a Boolean function with a very high predictive accuracy. The fact that the algorithm is capable of effectively handling missing information makes it more applicable to real data bases. Note, however, that the method makes no attempts at minimizing the size of the derived function.

Deshpande and Triantaphyllou [2] extended the RA1 heuristic for complete positive and negative examples, to include the use of incomplete data through the development of a second randomized heuristic, termed RA2. This method, allows not only for the inclusions of missing information in the attribute values, but it also makes use of examples which are *unclassifiable* due to the presence of missing information. That is, for some examples, the correct classification cannot be determined due to the lack of sufficient information. The objective of the second heuristic, similar to the first one, is to interactively derive a small-sized Boolean function from these three mutually exclusive sets: positive, negative, and unclassifiable examples. The algorithm consists of two phases. In each iteration of the first phase, the objective of the algorithm is to reject many negative examples while accepting all positive examples, and rejecting no unclassifiable examples. Once all negative examples have been rejected by the current set of clauses, phase II then assures that none of the unclassifiable examples are accepted by the system. When compared to the RA1 the accuracy obtained with the inclusion of unclassifiable data was always higher than the corresponding accuracy obtained without the inclusion of the unclassifiable data. This method has satisfactorily addressed the issues of efficiency and system size. Furthermore, it demonstrated that the process of extracting

knowledge from examples can be expedited by exploiting the patterns contained in missing information and unclassifiable examples.

A related issue in this area is the development of approaches for partitioning large scale problems in optimal or semi-optimal ways [24]. That was done by using a graph-theoretic approach. The same approach also allows to establish lower limits on the minimum number of clauses derivable from two given collections of input examples. Also, an approach for *guided learning* of a target Boolean function is proposed in [22]. In [10,21], and [18] the above problem was studied when the property of *monotonicity* can be established in the input data. Finally, in [12,13] and [11] some methods were presented for dealing with *fuzziness* and uncertainty.

Concluding Remarks

Clearly, minimizing the size of the inferred system is an attractive goal. A complex system is difficult to validate, difficult to apply, and difficult to understand. On the other hand, a method which seeks to minimize the size of the system creates an inefficient process which is both computationally difficult and limits the method's applicability due to the vast time and storage requirements. In light of the success of the RA2 heuristic, the authors of this article are currently conducting research aimed at the development of an 'optimal' logical method which has the ability to handle missing information not only in the attribute values, but in the classification of the examples as well.

The new method works in conjunction with the OCAT approach. Through the application of a modified B &B algorithm, CNF clauses are interactively generated such that the set of clauses, when taken together, accepts all positive examples, rejects all negative examples and neither accepts nor rejects any unclassifiable example. We consider in this effort three optimization goals: efficiency of the process, accuracy of the derived function and the number of clauses which comprise the derived function. Thus'optimal' in this sense implies the derivation of a small (hopefully minimum) and accurate Boolean system through the efficient exploitation of information contained in unclassifiable examples and, of course, the positive and negative examples.

In our current research efforts, optimization becomes even more vital. By allowing missing information and unclassifiable examples, the amount of available data increases and necessitates the use of a learning algorithm which does not require excessive amounts of time and/or memory requirements. In addition, the goal of a logical approach is to derive a system capable of accurately classifying new examples and providing justification for the decision. A logical system derived from incomplete data may encounter an example which cannot be classified due to insufficient information. This system must be capable of explaining why the example is unclassifiable. That is, it has the additional responsibility of assisting the decision maker in identifying the minimal amount of additional information required for classification of the example. In a minimal system, this information is more readily accessible.

See also

- ► Alternative Set Theory
- ► Boolean and Fuzzy Relations
- ► Checklist Paradigm Semantics for Fuzzy Logics
- ► Finite Complete Systems of Many-Valued Logic Algebras
- ► Inference of Monotone Boolean Functions
- ► Linear Programming Models for Classification
- ► Mixed Integer Classification Problems
- ► Optimization in Classifying Text Documents
- ► Statistical Classification: Optimization Approaches

References

1. Boros E, Hammer PL, Hooker J (1993) Predicting cause-effect relationships from incomplete discrete observations. RUTCOR Report RRR 9-93, Rutgers Univ
2. Deshpande AS, Triantaphyllou E (1998) A greedy randomized adaptive search procedure (GRASP) for inferring logical clauses from examples in polynomial time and some extensions. Math Comput Modelling 27(1):75–99
3. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Global Optim 6:109–133
4. Fu LM (1993) Knowledge-based connectionism for revising domain theories. IEEE Trans Syst, Man Cybern 23(1):173–182
5. Hall FM, Storella JM, Silverstone DZ, Wyshak G (1988) Non palpable breast lesions: Recommendations for biopsy based on suspicion of carcinoma at mammography. Radiology 157:353–358
6. Hooker JN (1988) Generalized resolution and cutting planes. Ann Oper Res 12(1–4):217–239
7. Hooker JN (1988) A quantitative approach to logical inference. Decision Support Systems 4:45–69

8. Kamath AP, Karmarkar NK, Ramakrishnan K, Resende M (1994) An interior point approach to Boolean vector function synthesis. In: IKE Proc. 3rd Midwest Symp. Circuits and Systems, vol 1, pp 185–189
9. Karmarkar NK, Resende M, Ramakrishnan K (1992) An interior point algorithm to solve computationally difficult set covering problems. Math Program 52(3):597–618
10. Kovalerchuk B, Triantaphyllou E, Deshpande AS, Vityaev E (1996) Interactive learning of monotone Boolean functions. Inform Sci 94(1–4):87–118
11. Kovalerchuk B, Triantaphyllou E, Ruiz JF (June 6–9 1996) Monotonicity and logical analysis of data: A mechanism for evaluation of mammographic and clinical data. In: Proc. 13-th Symp. Computer Applications in Radiology (SCAR), pp 191–196
12. Kovalerchuk B, Triantaphyllou E, Ruiz JF, Clayton J (1996) Fuzzy logic in digital mammography: Analysis of lobulation. In: Proc. FUZZ-IEEE'96 Internat. Confererence, New Orleans, 8–11 September 1996, vol 3, pp 1726–1731
13. Kovalerchuk B, Triantaphyllou E, Ruiz JF, Clayton J (1997) Fuzzy logic in computer-aided breast cancer diagnosis: Analysis of lobulation. Artif Intell Med 11:75–85
14. Kovalerchuk B, Triantaphyllou E, Ruiz JF, Torvik VI, Vityaev E (2000) The reliability issue of computer-aided breast cancer diagnosis. Comput Biomed Res 33(4):296–313
15. Peysakh J (1987) A fast algorithm to convert Boolean expressions into CNF. Techn Report IBM Computer Sci RC 12913(57971)
16. Quinlan JR (1983) Learning efficient classification procedures and their application to chess endgames. In: Michalski RS (ed) Machine Learning: An Artificial Intelligence Approach. Tioga Publ., Wellsboro
17. Shavlik JW (1994) Combining symbolic and neural learning. Machine Learning 14:321–331
18. Torvik IV, Triantaphyllou E (2000) Inferring a monotone Boolean function by asking a small number of membership questions (submitted)
19. Triantaphyllou E (1994) Inference of a minimum size Boolean function from examples by using a new efficient branch-and-bound approach. J Global Optim 5(1):69–94
20. Triantaphyllou E, Kovalerchuk B, Deshpande AS (1996) Some recent developments in logical analysis. In: Barr R, Helgason R, Kennington J (eds) Interfaces in Computer Sci. and Operations Research. Kluwer, Dordrecht, pp 215–236
21. Triantaphyllou E, Lu J (1998) The knowledge acquisition problem in monotone Boolean systems. In: Kent A, Williams JG (eds) Encycl. Computer Sci. and Techn. M Dekker, New York, pp 89–106
22. Triantaphyllou E, Soyster AL (1995) An approach to guided learning of Boolean functions. Math Comput Modelling, 23(3):69–86
23. Triantaphyllou E, Soyster AL (1995) A relationship Between CNF and DNF systems derivable from examples. ORSA J Comput 7(3):283–285
24. Triantaphyllou E, Soyster AL (1996) On the minimum number of logical clauses which can be inferred from examples. Comput Oper Res 23(8):783–799
25. Triantaphyllou E, Soyster AL, Kumara S (1994) Generating logical expressions from positive and negative examples via a branch-and-bound approach. Comput Oper Res 21(2):185–197
26. Trick M, Johnson D (1995) Second DIMACS challenge on cliques, coloring, and satisfiability. DIMACS Amer Math Soc, Providence
27. Van Dijck J, Verbeek A, Hendricks J, Holland R (1993) The current detectability of breast cancer in a mammographic screening program. Cancer 72(6):1933–1938

Optimization in Classifying Text Documents

SALVADOR NIETO SANCHEZ,
EVANGELOS TRIANTAPHYLLOU
Department Industrial and Manufacturing Systems Engineering, Louisiana State University, Baton Rouge, USA

MSC2000: 90C09, 90C10

Article Outline

Keywords
Overview of Automatic Classification of Documents
Examples of Optimization in Document Classification
Optimization in the Principle of Least Effort
Optimization in the Vector Space Model
Optimization in the Classification of Text Documents
Conclusions and Future Research
See also
References

Keywords

Document classification; Computational linguistics; Indexing terms; Context descriptors; Text classification; Keywords; Document surrogate; Principle of least effort; PLE; Vector space model; VSM; One clause at a time algorithm; OCAT; Indexing vocabulary; Optimal indexing vocabulary; Semantic analysis methodologies; Word patterns; Conjunctive normal form; CNF; Disjunctive normal form; DNF

From the 1950s onwards, the search for computerized tools and mathematical models that can speed up the *classification of large collections of documents* has been the focus of many research efforts. These efforts have been centered in developing tools that can speed up the classification of documents according to some underlying context. A current example of this situation is the *Internet*. In this worldwide conglomerate of databases, one can easily see the speed at which documents on the topic, say, 'basketball' are retrieved from among the millions of documents produced daily on the Internet. Document classification is also of paramount importance in many information retrieval applications, such as news routing [7], classification/declassification of official documents [15] e-mail filtering [27], and context derivation of electronic meetings [3].

From the 1950s onwards, various fields of the human knowledge have produced several solutions for the document classification problem (see, for example, [21,23], and [2]). Some examples of these fields are mathematical optimization, computational linguistics, expert systems, neural networks, and genetic algorithms. These methodologies have been severely limited to some degree by the huge amounts of information, both textual and graphical, generated by today's information driven society. On the other hand, this 'technological' limitation has been the boost for the development of more efficient and effective classification procedures [15].

The purpose of this article is to exhibit some contributions of discrete optimization during the process of *automatic document classification*. This paper illustrates these contributions by presenting three cases (application areas) in which optimization is used in the classification process. The first case deals with a generic procedure for the selection of a set of *indexing terms* (*keywords* or *context descriptors*). The second case deals with the selection of an optimal set of indexing terms to minimize the overlapping of keywords used in different documents. The last case deals with the classification of text documents from mutually exclusive classes. These three cases are only a tiny sample of a vast collection of related instances in the field of information retrieval systems; see [1,11], and [25] for additional literature.

This article is organized as follows. The next section presents an overview of the document classification process. The subsequent section illustrates the three application areas in which optimization has contributed in the solution of the classification problem. Finally, a summary section is given.

Overview of Automatic Classification of Documents

The automatic classification of text documents consists of grouping documents of similar context into meaningful groups in order to facilitate their storage and retrieval [22].*Text classification* can be viewed as a four-step process. In the first step, a representative sample of documents from various classes is presented to a computerized system, and a list of the co-occurring words with their frequencies is secured (see, for example, [22] and [4]).

In the second step the frequency of the words is analyzed, and only the most 'meaningful' words are extracted as indexing terms (keywords or context descriptors) [14]. The 'meaningful' words or keywords are the words with moderate co-occurring frequencies. H.P. Luhn [13], G. Salton [21], D. Cleveland and A.D. Cleveland [4], and Ch. Fox [6] suggest the elimination of the 'common' and 'rare' words (i. e., frequent and infrequent words, respectively) as indexing terms because they convey little lexical meaning. Some examples of common words are 'a', 'an', 'and', and 'the' [6]; 'rare' words are dependent on the document's subject [13].

In the third step the context of unclassified documents is determined by affixing them with the keywords that occur in their text. According to [4], 'the assignment of these keywords to a document is correct because authors usually repeat words that conform with the document's subject.' Finally, the documents which were indexed with similar keywords are grouped together [22].

The set of keywords attached to each document during the third step is often referred to as a *document surrogate* or just a *document* [4]. A surrogate is a convenient way to represent and to computationally process the context of real documents. For instance, the surrogate of seven words { 'document classification', 'document indexing', 'optimization', 'vector space model', 'logical analysis approach', 'OCAT algorithm', and 'machine learning' } is a condensed and convenient way to represent the context of this article which contains thousands of words, symbols, and numbers.

$$\begin{array}{c} \\ D_1 \\ \cdot \\ \cdot \\ \cdot \\ D_N \end{array} \begin{array}{c} \begin{array}{ccc} T_1 & \cdots & T_t \end{array} \\ \begin{bmatrix} w_{11} & \cdots & w_{1t} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ w_{N1} & \cdots & w_{Nt} \end{bmatrix} \end{array}$$

Optimization in Classifying Text Documents, Figure 1
A collection of *N* surrogates

Often, a surrogate is further simplified by defining it as a binary vector. (For nonbinary surrogates, see [22].) In this case, when a surrogate's element $w_{ij} = 1$ (or 0), it indicates the presence (or absence) of keyword T_i ($i = 1, \ldots, t$) in document D_j ($j = 1, \ldots, N$). For example, the surrogate $D_k = [\ 0\ 1\ 1\ 1\ 1\ 0]$ of six binary elements indicates the presence of keywords T_2, T_3, T_4, and T_5 and the absence of keywords T_1 and T_6 in D_k. Figure 1 shows a popular way to summarize a collection of N documents (surrogates) which are defined on t [22]:

In the last step, documents sharing similar keywords are grouped together. This classification follows from the pairwise comparison of the surrogates in Fig. 1 [22]. More on this is described in the following section.

Examples of Optimization in Document Classification

Optimization techniques have been used in various areas of text classification with various levels of success. Their utilization has been limited mainly because the size of the document classification problem is so large that even with the current computerized technologies, it would take them very long time to produce an optimal solution. Despite these technological limitations, the contributions of optimization in this field can be seen, for example, in the selection of keywords, automatic classification of documents, automatic retrieval, etc. Some applications of these techniques are presented next.

The first example illustrates the principle of least effort [29]. This principle is used for the derivation of an indexing vocabulary based solely on the frequency of the co-occurring words in a collection of documents. The second example illustrates the application of the vector space model [23] for the derivation of an optimal indexing vocabulary to minimize the overlapping of keywords used by different documents. The third example illustrates the utilization of a machine learning and operations research algorithm called the one clause at a time (OCAT) algorithm [28] for the classification of documents which belong to mutually exclusive classes.

Optimization in the Principle of Least Effort

The *principle of least effort* (PLE) can be viewed as one of the first optimization attempts in the area of document classification. It was introduced by H.P. Zipf [29]. Although the PLE does not have a strict mathematical formulation, the problem it solves can be stated as follows. Given a collection of documents, the question is how to derive the 'best' set of indexing terms that will be used to identify the subject of documents in the collection. The set of the best indexing terms (or keywords) is often referred to as *indexing vocabulary* (see, for example, [4]). Hence, the goal of the PLE is to derive an optimal indexing vocabulary with the most *meaningful words* occurring in these documents.

Under the PLE an indexing vocabulary is derived as follows. At first all the co-occurring words and their frequencies are extracted from the collection of documents. Then, these words are ranked in descending order according to their frequencies. Finally, the words with frequencies in between some preestablished upper and lower frequency limits are selected as the indexing vocabulary. The frequency boundaries of the meaningful words are determined by a trial-and-error approach [13]. Other words with co-occurring frequencies above or below the preestablished limits are known as 'common' and 'rare' words (the frequent and infrequent words, respectively) and usually are discarded for indexing purposes because they convey little lexical meaning (see, for example, [13] and [16]).

It is interesting to notice here that although the PLE does minimize the number of keywords, its unwise utilization may jeopardize the quality of the indexing vocabulary. This can be illustrated by considering the word 'a'. The word 'a' is one of the most common words in the English language (other such words are 'an', 'and', and 'the'; see, for example, [6]). Thus, if the collection of documents is about nutrition, then 'a' may represent the name of the vitamin 'a' or 'A', and its

elimination clearly would jeopardize the quality of the indexing vocabulary.

Optimization in the Vector Space Model

One of the most successful models in information retrieval systems is the *vector space model* (VSM). It was introduced in the mid 1970s in [23]. The VSM solves problems of the following nature: Given are samples of documents. Then, the question here is how to derive an optimal indexing vocabulary such that keywords used in one document are minimally used in other documents. In [23], the VSM was also extended to determine a vocabulary that minimizes the overlapping of keywords used in different classes. That is, keywords used in documents belonging to one class are minimally used in other classes.

The VSM derives this *optimal vocabulary* as follows. At first, a sample of t words is taken from all the words co-occurring in a collection of N documents. This sample of words is used as a candidate indexing vocabulary. Then, all documents in the collection are indexed (their subject is defined) by using words from this candidate vocabulary. Document surrogates are formed in this step. Next, the VSM computes the *similarity of all the surrogates* in the collection according to

$$F = \sum_{i=1}^{N} \text{sim}(D_i, D_j) \quad \text{for} \quad j = 2, \ldots, N \quad \text{and} \quad i \neq j\,. \tag{1}$$

Where $\text{sim}(D_i, D_j)$ measures the similarity between documents D_i and D_j. Usually, $\text{sim}(D_i, D_j)$ is replaced by a function that relates any two vectors, such as the functions illustrated in Table 1. This procedure is repeated by using various candidate vocabularies. Finally, the candidate vocabulary that minimizes the expression in (1) is selected as the optimal indexing vocabulary [23]. The following example illustrates an application of the VSM with *binary surrogates*. The cosine coefficient is used to solve (1).

Example 1 Let the words $T_1, \ldots, T_7$ be the set of all the words which were found in documents D_1 and D_2. (In real practice, this set may contain hundreds or even thousands of words.) Next, let D_1 and D_2 be indexed with only four of these words. Hence, the question here is: what is the 'optimal' indexing vocabulary of four words that make document D_1 to be indexed with keywords that are minimally used by D_2?

It is not difficult to realize that the number of candidate vocabularies of four words that can be formed out of seven words is equal to $\binom{7}{4} = 35$. Table 2 shows the similarities between D_1 and D_2 for only three vocabularies. The first column of this table shows these vocabularies. For example, words T_1, T_2, T_4, and T_7 correspond to the first vocabulary. The second and third column show the binary surrogates of documents D_1 and D_2. For instance, the surrogate D_1 = [1 0 1 1] indicates the absence of word T_2 and the presence of words T_1, T_4, and T_7 in D_1. Similarly, the surrogate for D_2 = [0 0 1 1] indicates the absence of T_1 and T_2 and the presence of words T_4 and T_7 in D_2. Finally, the fourth column shows the CC similarity values, or $\text{sim}(D_1, D_2)$), between D_1 and D_2 for the three vocabularies.

The CC similarity values in Table 2 indicate that when words T_3, T_4, T_5, T_6 are used as indexing terms, the similarity of documents D_1 and D_2 is minimal. Furthermore, the similarity value $\text{sim}(D_1, D_2) = 0.00$ indicates that both documents are completely different because their surrogates do not contain common words. Therefore, according to the VSM the optimal indexing vocabulary corresponds to terms T_3, T_4, T_5, and T_6. Thus, the other words T_1, T_2, and T_7 can be discarded as indexing terms.

The solution presented in this table seems to be a trivial one. However, it can be easily shown that for realistic indexing problems such a solution can be quite an elaborate one. Suppose, for example, that the number of words extracted from D_1 and D_2 is not seven, but fifty (which still is a small number for realistic situations). Furthermore, if this time one is interested in finding candidate vocabularies of ten words, rather than combinations of four words, then the number of vocabularies that can be constructed is $\binom{50}{10} = 10.27$ billions. That is, in addition to finding the minimization of (2), the VSM must also use an efficient search strategy to quickly eliminate many non optimal vocabularies. By the same token, it can be easily shown that if vocabularies of all sizes are considered, then the number of such vocabularies is determined by $2^t - 1$ (where t is the number of extracted words), which even for moderate values of t this expression is a too large number.

As a result of these humongous searching spaces, researchers have been compelled to design fast heuris-

Optimization in Classifying Text Documents, Table 1
Measures of vector similarity (taken from [22, Chap. 10]

similarity measure sim(X, Y)	Evaluation for binary term vectors	evaluation for weighted term vectors
inner product (IP)	$\mid X \cap Y \mid$	$\sum_{i=1}^{t} x_i \cdot y_i$
dice coefficient (DC)	$2\frac{\mid X \cap Y \mid}{\mid X \mid + \mid Y \mid}$	$2\frac{\sum_{i=1}^{t} x_i \cdot y_i}{\sum_{i=1}^{t} x_i^2 + \sum_{i=1}^{t} y_i^2}$
cosine coefficient (CC)	$\frac{\mid X \cap Y \mid}{\mid X \mid^{1/2} \cdot \mid Y \mid^{1/2}}$	$\frac{\sum_{i=1}^{t} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{t} x_i^2 \cdot \sum_{i=1}^{t} y_i^2}}$
Jaccard coefficient (JC)	$\frac{\mid X \cap Y \mid}{\mid X \mid + \mid Y \mid - \mid X \cap Y \mid}$	$\frac{\sum_{i=1}^{t} x_i \cdot y_i}{\sum_{i=1}^{t} x_i^2 + \sum_{i=1}^{t} y_i^2 - \sum_{i=1}^{t} x_i y_i}$

$X = (x_1, \ldots, x_i)$
$\mid X \mid$= number of terms in $\mid X \mid$.
$\mid X \cap Y \mid$= number of terms appearing jointly in X and Y.

Optimization in Classifying Text Documents, Table 2
Similarity sim(D_1, D_2) for three candidate vocabularies of four words

	Surrogates		
Vocabularies	D_1	D_2	sim(D_1, D_2)
T_1, T_2, T_4, T_7	[1011]	[0011]	0.50*
T_2, T_4, T_5, T_6	[0100]	[0110]	0.25
T_3, T_4, T_5, T_6	[0100]	[0011]	0.00

* : sim$(D_1, D_2) = 2/(4^{1/2} \times 4^{1/2}) = 0.50$.

tic search strategies at the expense of optimality. Furthermore, because the size of the indexing problem can be very large, suboptimal heuristic solutions have been preferred over optimal but slow ones [2].

Optimization in the Classification of Text Documents

The process of document classification consists of grouping documents according to their underlying subject. Some examples of familiar document subjects are history, geography, music, engineering, etc. Usually, this underlying subject is determined by the set of indexing terms which was attached to each document. That is, documents about History share indexing terms whose content describes past events. Similarly, the indexing terms of documents about geography share content which describes different geographic places on earth. Hence, the problem that this process of document classification attempts to solve is described as follows: Given samples of preclassified documents (i. e., their surrogates), the question is how to use the information contained in their surrogates such that new unclassified documents can be grouped into the appropriate classes.

There are many methodologies for solving this document classification problem. Some examples of such methodologies are: the vector space model for document classification [22]; fuzzy set theories [12] and [17]; semantic analysis methodologies [10] and [19]; and some others which use artificial intelligence approaches, [3], and [2]. To some extent all these methodologies use optimization in order to maximize (or minimize) some performance measure, which usually is the similarity between indexing terms. In what follows, we present only one methodology which is based on artificial intelligence and operations research approaches. This methodology is the one clause at a time (OCAT) algorithm [28] for the classification of examples (e. g., documents) in mutually exclusive classes. The OCAT algorithm uses optimization methodologies for constructing classification clauses (e. g., *word patterns*) of minimal (or near minimal) size. Figure 2 shows this algorithm.

Input:	E^+ and E^-
Output:	Logical rules in CNF (or DNF) form.
	$i = 1; C = \emptyset;$
DO	WHILE ($E^- \neq \emptyset$)
1:	$i \leftarrow i + 1;$ / * i indicates the ith iteration /
2:	find a clause c_i which accepts all members of E^+ while it rejects as many members of E^- as possible;
3:	let $E^-(c_i)$ be the set of members of E^- which are rejected by c_i;
4:	let $C \leftarrow C + c_i$;
5	let $E^- \leftarrow E^- - E^-(c_i)$;
	END;

Optimization in Classifying Text Documents, Figure 2
The one clause at a time (OCAT) algorithm

$$E^+ = \begin{matrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{matrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

and

$$E^- = \begin{matrix} e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \\ e_{10} \end{matrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Optimization in Classifying Text Documents, Figure 3
Two illustrative sets of positive and negative examples

The OCAT algorithm is also a machine learning algorithm. It uses logical analysis and branch and bound approaches to extract knowledge (sets of rules) from sets of preclassified examples. It takes as input data samples of examples from (usually two) mutually exclusive classes and extracts knowledge that is represented in a compact form of key data patterns which can be used to classify new unclassified examples into these two classes.

The two mutually exclusive classes are referred to as the sets of positive and negative examples (denoted by E^+ and E^-, respectively). Furthermore, the collections of examples in both classes are defined over the same set of parameters (also called atoms, characteristics, or factors) which are assumed binary valued. Figure 3 illustrates a set of four positive examples: e_1, e_2, e_3, e_4 and a set of six negative examples: e_5, e_6, e_7, e_8, e_9, e_{10}. All ten examples are defined on the four atoms A_1, A_2, A_3, and A_4. For instance, example e_1 = [0 1 0 0] indicates the presence of atom A_2 and the absence of atoms A_1, A_3, and A_4 in e_1. On the other hand, example e_5 = [1 0 1 0] indicates that atoms A_1 and A_3 are present and that atoms A_2 and A_4 are absent.

When the OCAT algorithm is used to solve the document classification problem, E^+ and E^- (i. e., the sets with the positive and negative examples, respectively) correspond to the sets of documents which belong to two mutually exclusive classes. That is, documents in the positive class are the ones that belong in only *one* of the two classes, while the documents in the other class are the negative examples. Hence, Fig. 3 may represent a set of ten documents (surrogates) which were indexed by using four keywords.

The OCAT algorithm is a greedy algorithm which determines a set of compact clauses either in the *conjunctive normal form* or *disjunctive normal form* (CNF or DNF, respectively, as defined below). For example, CNF clauses are determined as follows. In the first iteration it determines a clause that accepts all the examples in E^+ while it rejects as many examples in E^- as possible. In the second iteration it performs the same operation using the original E^+ set but this time the current E^- set contains only the negative examples that have not been rejected by any of the previous clauses. The iterations continue until the set of constructed clauses reject all the negative examples. Hence, when these CNF or DNF clauses are taken together, they accept all the positive examples while they reject all the negative examples.

The conjunctive normal form and disjunctive normal form (see, for example, [24]) are defined as in expressions (2) and (3), respectively.

$$\bigwedge_{j=1}^{k} \left(\bigvee_{i \in \rho_j} a_i \right), \tag{2}$$

$$\bigvee_{j=1}^{k} \left(\bigwedge_{i \in \rho_j} a_i \right). \tag{3}$$

Where a_i can be either A_i or $\overline{A}_i$. Thus, a CNF expression (also called a *logical clause*) over a vector $v \in \{0, 1\}^t$ is a conjunction of disjunctions defined on the terms A_i $(i = 1, \ldots, t)$. Similarly, a DNF expression is a disjunction of conjunctions on the same terms A_i.

Let n be the number of atoms and M_1 the number of positive examples. It can be easily shown that the maximum number of clauses that can be formed using n atoms and M_1 examples is equal to M_1 [28]. To form these clauses consider the first example $e_1 = [0\ 1\ 0\ 0]$. It can be observed that in order to accept this positive example at least one of the four atoms A_1, A_2, A_3, A_4 must be specified as follows: (A_1 = FALSE; i. e., $\overline{A}_1$ = TRUE), (A_2 = TRUE), (A_3 = FALSE; i. e., $\overline{A}_3$ = TRUE), and (A_4 = FALSE; i. e., $\overline{A}_4$ = FALSE). Hence, any valid CNF clause must include $\overline{A}_1$, or A_2, or $\overline{A}_3$, or $\overline{A}_4$. Similarly, the second positive example $e_1 = [1\ 1\ 0\ 0]$ indicates that any valid CNF clause must include $A\overline{A}_1$, or $A\overline{A}_2$, or $\overline{A}_3$, or $\overline{A}_4$. In this manner, all valid CNF clauses must include at least one atom as specified from each of the following sets: $\{\overline{A}_1, A\overline{A}_2, \overline{A}_3, \overline{A}_4\}$, $\{A\overline{A}_1, A\overline{A}_2, \overline{A}_3, \overline{A}_4\}$, $\{\overline{A}_1, \overline{A}_2, A_3, A\overline{A}_4\}$, and $\{A_1, \overline{A}_2, \overline{A}_3, A_4\}$. Relation (4) shows a CNF system which was derived by using the OCAT algorithm on the examples in Fig. 3:

$$(A_2 \vee A_4) \wedge (\overline{A}_2 \vee \overline{A}_3) \wedge (A_1 \vee A_3 \vee \overline{A}_4)\,. \qquad (4)$$

Example 2 An application of the OCAT algorithm can be illustrated by using a new example, say, $e_{11} = [0\ 0\ 1\ 0]$. When e_{11} is 'tested' by the above CNF expression, then it can be seen that e_{11} is classified as a negative example. This is as follows. The clause $A_2 \vee A_4$ evaluates to 0 because e_{11} does not contain neither the second nor the fourth atoms. On the other hand, both clauses $\overline{A}_1 \vee \overline{A}_3$ and $A_1 \vee A_3 \wedge \overline{A}_4$ evaluate to 1. However, when the three clauses are taken together, expression (4) evaluates to 0, thus indicating that e_{11} is a negative example.

Conclusions and Future Research

This article illustrated some contributions of optimization for solving the document classification problem. These contributions were illustrated by presenting three cases (application areas) in which optimization has been used. The first case dealt with the principle of least effort (PLE) which is used for the selection of an indexing vocabulary based solely in the frequency of the co-occurring words. The second case dealt with the vector space model (VSM) for the selection of an indexing vocabulary that minimizes the overlapping of words used in various documents (or in various document classes). The third case illustrated the one clause at a time (OCAT) algorithm for the classification of documents into mutually exclusive classes.

A common characteristic of these three cases is the huge amounts of information that need to be processed before optimal solutions can be found. Therefore, the optimization techniques presented in these examples have been used extensively only on document classification problems of small size. The main reason for this limitation is that even with the current computerized technologies, these techniques would take unacceptable processing times to find optimal solutions for larger classification problems. As a consequence, scientific research efforts have focused their attention in developing effective and efficient heuristics for solving problems of more realistic size.

See also

- ▶ Alternative Set Theory
- ▶ Boolean and Fuzzy Relations
- ▶ Checklist Paradigm Semantics for Fuzzy Logics
- ▶ Finite Complete Systems of Many-valued Logic Algebras
- ▶ Inference of Monotone Boolean Functions
- ▶ Linear Programming Models for Classification
- ▶ Mixed Integer Classification Problems
- ▶ Optimization in Boolean Classification Problems
- ▶ Statistical Classification: Optimization Approaches

References

1. Brophy P, Buckland MK, Hindle A (1976) Reader in operations research for libraries. Inform Handling Services, Englewood
2. Chen H (1996) A machine learning approach to document retrieval: An overview and experiment. Working Paper, Univ Arizona, College of Business Administration
3. Chen H, Hsu R, Orwing R, Hoopes L, Numamaker JF (1994) Automatic concept classification of text from electronic meetings. Comm ACM 30(10):55–73
4. Cleveland D, Cleveland AD (1983) Introduction to indexing and abstracting. Libraries Unlimited, Littleton
5. Croft WB (1993) Knowledge-based and statistical approaches to text retrieval. IEEE Expert 8(2):8–12
6. Fox Ch (1990) A stop list for general text. Special Interest Group on Information Retrieval 24(1–2):19–35

7. Jacobs PS (1993) Using statistical methods to improve knowledge-based news categorization. IEEE Expert 8(2):13–23
8. Jacobs PS, Rau L (1990) SCISOR: Extracting information from on-line news. Comm ACM 33(11):88–97
9. Kim J-T, Moldovan DI (1995) Acquisition of linguistic patterns for knowledge-based information extraction. IEEE Trans Knowledge and Data Eng 7(5):713–724
10. Korfhage RR, Olsen KA (1994) The role of visualization in document analysis. In: Third Annual Symposium on Document Analysis and Information Retrieval. UNLA/ISRE, Las Vegas, pp 199–207
11. Kraft DH, Boyce RB (1991) Operations research for the libraries and information agencies. Techniques for the evaluation of management decision alternatives. Acad. Press, New York
12. Lee JH, Kim MH, Lee YJ (1992) Enhancing the fuzzy set model for high quality document ranking. Microprocessing and Microprogramming 35:337–334
13. Luhn HP (1958) The Automatic creation of literature abstracts. IBM J Res Developm 2:159–165
14. Meadow ChT (1992) Text information retrieval systems. Acad. Press, New York
15. Meridian D (1996) Declassification productivity initiative study report. Techn Report, DynCorp Comp
16. Merriam–Webster (1993) Collegiate dictionary, 10th edn. Merriam-Webster, Springfield, MA
17. Molinary A, Passi G (1996) A fuzzy representation of HTML documents for information retrieval systems. In: Proc. Fifth IEEE Internat. Conf. Fuzzy Systems, pp 197–112
18. Nash SG, Sofer A (1996) Linear and nonlinear programming. McGraw-Hill, New York
19. Rau LF, Jacob PS (1991) Creating segmented databases from free text for text retrieval. In: Proc. Fourteenth Annual Internat. ACM/SIGIR Conf. Research and Development in Information Retrieval. ACM, New York, pp 337–346
20. Riggs FW (1991) Delphic language: A problem for authors and indexers. Library Sci 28(1):18–30
21. Salton G (1968) Automatic information organization and retrieval. McGraw-Hill, New York
22. Salton G (1989) Automatic text processing. The transformation, analysis, and retrieval of information by computer. Addison-Wesley, München
23. Salton G, Wong A, Yang CS (1975) A vector space model for automatic indexing. Comm ACM 18(11):613–620
24. Schneeweiss W (1989) Boolean functions with engineering applications and computer programs. Springer, Berlin
25. Swanson DR, Bookstein A (1971) Operations research: Implications for libraries. University Chicago Press, Chicago
26. Taha HA (1997) Operations research: An introduction, 6th edn. MacMillan, New York
27. Takkinen J (1996) An adaptive approach to text categorization and understanding - A preliminary study. Working Paper, Dept Computer and Information Sci, Univ Linkoping
28. Triantaphyllou E (1994) Inference of a minimum size Boolean function from examples by using a new efficient branch-and-bound approach. J Global Optim 5:64–94
29. Zipf HP (1949) Human behavior and the principle of least effort. Addison-Wesley, München

Optimization and Decision Support Systems

Freerk A. Lootsma
Fac. Math. Informatics, Delft University Technol., Delft, The Netherlands

MSC2000: 90B50

Article Outline

Keywords
References

Keywords

Linear programming; Mixed integer programming; Integer programming; Nonlinear programming; Multi-objective optimization; Cutting-stock problem; Load dispatching; Energy modeling; Group decision making

The engine is an essential part of a motor car but it is useless if it is not embedded in a car body with chairs, windshields, tyres, brakes, etc. Similarly, an optimization algorithm is useless if it is not embedded in an appropriate model and if it is not linked to the outside world via a system that helps the decision makers to use the data and the model.

Optimization systems may be found at all levels of decision making: at the level of operational planning with a short-term horizon (hours or days), such as the scheduling of the production of corrugated cardboard, glass, iron, and steel, the operational control of a water-management system, and the *load dispatching* in electricity generation; at the level of tactical planning with a medium-term horizon (months), such as the capacity planning of assembly-line production; and at the level of strategic planning with a long-term horizon (years), such as the choice of a strategy for the national energy

supply or the design of the infrastructure of a water-management system. In what follows we shall describe some of these systems in more detail.

In the voluminous literature on *linear programming* applications one constantly comes across references to the *cutting-stock problem* or the trimloss problem in the corrugated-cardboard industry, where long strips or rectangular plates of material are cut into smaller rectangles. The objective is to set the circular and the guillotine cutters in such a way that the cardboard production machine supplies the required number of rectangular plates for each client with a minimum of waste. Real-life conditions on the factory floor impose additional requirements, such as the reduction of the set-up costs due to the resetting of the cutters, which cannot be formulated in pure linear programming terms. They can be allowed for in a *mixed integer programming* model but this tends to become intractable for existing numerical methods because the number of zero-one variables may be large. A more successful approach is the generation of a number of promising cutting patterns followed by the selection of a small number of patterns for the actual production of corrugated cardboard. This is a pure *integer programming* formulation of the trimloss problem. An additional complication is that the format of the rectangular plates is not necessarily hard. Under certain conditions the planners may deviate from the ordered dimensions, and also from the required number of plates. This greatly alleviates their task to reduce the losses of time and material as much as possible. An interesting feature of the systems for the planning of cardboard production is that they do not only reduce the losses, but they also contribute to a smooth processing of orders, planning lists, deliveries, and invoices. Thus, these systems do not only tend to control the actual production but they also support the administrative processes in the factory.

Electricity generation is subject to the iron law that demand has to be satisfied, as and when it occurs. Under these constraints electricity companies usually pursue the economic objective of minimizing the fuel costs. The calculation of the cheapest production schedule is divided into several parts, each with its own time horizon. One of these parts is the static dispatch problem, the allocation of the electricity demand predicted for the next fifteen or sixty minutes to the available power units. Another part is the unit-commitment problem, the daily selection of the units for actual production. In its mathematical form the *dispatch problem* can be written as the minimization of a nonlinear fuel-cost function subject to a linear demand constraint whereas upper and lower bounds are imposed on the variables. In general, the fuel-cost function can be simplified and written as a convex, separable, quadratic function of the loads assigned to the respective power units. The dispatch problem is accordingly a quadratic continuous knapsack problem which, in its primal form, can efficiently be solved via gradient methods for *nonlinear optimization*. A dramatic acceleration occurs when the dispatch problem is considered in its dual form. In each step, after the unconstrained minimization of the associated Lagrangian function, one can immediately find a number of variables which are at their optimal values so that they can be removed from the problem formulation. This rapidly reduces the size of the dispatch problem [1]. More strongly, it implies that the dispatch problem can efficiently and reliably be solved, even if there are hundreds of variables. This problem size is normal in the electricity generation for a country of 15 or 20 million inhabitants.

After the oil crisis of the early 1970s many countries stimulated the development of so-called energy models which could be used to analyze long-term strategies for the national energy supply via linear programming. Such a model has three parts at a high aggregation level: winning and/or import of primary energy carriers like crude oil, natural gas, and coal; conversion (refineries and power plants) into secondary energy carriers like petrol and electricity; and consumption in various sectors of the national economy like heavy industry, light industry, transportation, and households. Because there is a flow of materials from winning/import to consumption, the linear programming model is largely a network flow model. In the first and the second part there are certain limits to be set by the decision makers: upper limits on import and winning, and upper limits on the nuclear capacity for electricity generation, for instance. In the third part the decision makers can introduce the scenario-dependent projections of the future energy demand. In practice, since the supply of oil and natural gas is in the hands of large multi-national companies, these models are mainly concerned with strategies for the national elec-

tricity supply. For a prespecified year, using as input the projection of the electricity demand in various sectors of the economy, the projected prices of primary energy carriers, the estimates of the investment costs of new energy technologies, and the estimated supply restrictions, capacity limitations, and conversion efficiencies, the linear programming model yields an optimal mix of secondary energy carriers and technologies. An obvious extension of an *energy model* is the incorporation of multiple objective functions representing the policy objectives:

a) to minimize the environmental damage;
b) to maximize the
   - economic benefits;
   - safety of the energy system;
   - strategic independence of the country.

The analysis of the model usually shows that the exploitation of nuclear energy is strongly in conflict with the exploitation of fossil fuels. Nuclear energy is felt to reduce the safety of the energy system, but it also reduces the environmental damage of CO_2 and NO_x emissions, and it enhances the strategic independence by the reduction of crude-oil imports.

In the 1980s energy modeling strongly contributed to the popularity of scenario analysis. It became fashionable to explore the future and to judge various long-term strategies for the national electricity generation within the framework of a number of scenarios. The increasing electricity demand could be covered by an increased nuclear capacity, by the construction of new and more powerful fossil-fuel plants, and/or by international cooperation and diversification. Scenarios were designed as follows. First, the factors were identified which would affect the economic, political, and social developments until a predetermined horizon and which were beyond the decision maker's control. Because these factors are mostly interconnected, the hypothetical future developments could be bundled in a small number of coherent scenarios. Strategic planners designed a trend-following business-as-usual scenario where the current developments were smoothly continued, an optimistic scenario where the developments proceeded in an upward direction, and a pessimistic scenario with a less upward or even downward direction of the future developments. Each strategy had certain desirable and/or undesirable consequences within the context of the respective scenarios. These consequences could be evaluated via the analysis of linear programming energy models, at least to some extent. In choosing a strategy, the decision makers would have to weigh the consequences and to take into account the scenario probabilities. This complicated process was usually prepared in a network of policy committees and advisory councils who were expected to suggest a preferred strategy on the basis of a national energy outlook [2] with a horizon of ten or twenty years.

The decision support systems just described have been used for decades. The systems for operational planning support the short-term decisions of the planners who work in situations where the rules and the data are fairly well-known. Nevertheless, even in short-term planning the objectives and the constraints are not necessarily hard. The planners in a corrugated-cardboard factory, for instance, have to find a compromise between the objective to minimize the losses of material and the objective to smooth the administrative processes on the factory floor. The demand constraints may also be relaxed. The planners may deviate from the required dimensions, and they may ask the clients, either to order a larger quantity of rectangular plates now, or to postpone the delivery to the next planning period, because the order can nicely be combined with the order of another regular client.

In the systems for strategic planning, not only the rules and the data are imprecise, but also the users are difficult to identify. Strategic planning is largely a distributed decision-making process wherein many actors are involved: members of policy committees and advisory councils, managers of energy-producing companies, representatives of trade unions, members of pressure groups, the press, etc. They all have contradictory views and conflicting objectives in the energy sector, and they have widely varying power positions. In the eighties their views have been analyzed via multicriteria decision analysis and *multi-objective optimization* [3]. The study clearly identified the critical issues in the energy debate: safety and environmental protection. The proposed compromise solution to the long-term energy supply problem contained a significant contribution of nuclear energy so that it was ignored after the Chernobyl disaster.

In the late 1980s the study had to be taken up again because the choice of primary energy carriers cannot

be postponed indefinitely, certainly not when obsolete power plants have to be redesigned and replaced within the next planning period [5]. There were several options: new nuclear power plants, or larger and more sophisticated fossil-fuel plants using oil, coal, or natural gas. The objectives of the electricity companies were clear. The two small nuclear power plants in the country covered some 7% of the national demand, and the companies were convinced that the nuclear option should be increased. However, any proposal to start the design of a new nuclear plant would be rejected by the Ministry of Economic Affairs: it would politically be unfeasible. Hence, in order to avoid a negative decision that would be irreversible for a period of ten or twenty years, the electricity companies proposed to cover the peak demand in the next planning period via special contracts with foreign companies. This strategy would enable them to delay the design of nuclear power plants beyond the planning horizon of ten years. The compromise appeared to be acceptable, at least in the late eighties, and even today the consequences do not constitute an issue for the political parties and the environmental groups. The domestic nuclear capacity will be closed down within a few years, but by the electricity imports from neighboring countries the national economy will remain dependent on nuclear power, even more than in the 1980s.

We have briefly summarized the course of events in order to sketch the fortunes of a decision support system in a field full of controversies. The system cannot dictate a solution, but it can easily identify the conflicting views of the actors and the relative importance of their arguments. It makes the criteria operational because it shows the effects of the weights assigned to them.

Particular attention must be given to the interface with the decision makers. Many decisions are made in groups: in boards, councils, committees, where the members have contradictory views, opinions, aspirations, and power positions. Single-objective optimization models therefore tend to be inadequate. A realistic model has multiple objective functions, and usually these functions have different weights for the decision makers. In fact, multi-objective optimization has two subfields: the identification of the nondominated solutions, and the selection of a nondominated solution where the objective functions are felt to be in a proper balance. The first-named subfield can be studied in the splendid isolation of mathematical research. The second subfield, however, straddles the boundary between mathematics and other disciplines because human subjectivity is an integral part of the selection process. One cannot just leave it to an optimization expert to formulate a model and to calculate an acceptable nondominated solution. At various stages the expert has to interrupt the computational process in order to fathom the preferences of the decision makers. Certain parameters (weights, targets, desired levels) are adjusted on the basis of new preference information, whereafter the computations proceed in a somewhat modified direction. It is not always clear, however, how the parameters should be adjusted in order to guarantee a rapid convergence towards an acceptable compromise. This is crucial because decision makers cannot spend much time on a particular decision problem. It is an illusion to think that many interruptions would be possible to elicit preference information. One or two sessions of the decision-making body, communication via the mail and the telephone in the time intervals between interruptions, and that 's all.

In the 1980s, experts could work with questionnaires to fathom the preferential feelings in a group. The decision makers leisurely answered the questions in their office or at home, and returned the responses via the mail. In the next session of the group the calculated results were available for discussion [3]. Today, however, information technology provides sophisticated facilities for *group decision making*. Group Decision Rooms with networked PCs and a public screen for electronic brainstorming and weighted voting are commercially available. The sessions in a GDR have more impact than the questionnaires. First, the technology of the GDR eliminates the advantages of certain discussion techniques. The group members with strong verbal skills who usually dominate a meeting lose their grip on the silent majority as soon as the buttons are to be pressed. Second, the anonymous brainstorming and voting procedures promote an egalitarian attitude in the group (this may be a stumbling block in authoritarian cultures where decisions are deferred to the boss). GDR sessions create a certain commitment among the participants, possibly due to the intense communication, so that the decisions cannot easily be reverted thereafter. In summary, one may observe

a power shift in a GDR which strongly affects the choice of a strategy [4].

References

1. Bosch PPJ van den, Lootsma FA (1987) Scheduling of power generation via large-scale nonlinear optimization. J Optim Th Appl 55:313–326
2. Energy Study Centre (1987) National energy outlook. Report ESC-42
3. Kok M (1986) Conflict analysis via multiple objective programming. Doctoral Diss, Fac Math Inform Delft, Univ Techn
4. Laplante A (Oct. 1993) Nineties style brainstorming. Techn Suppl Forbes Magazine 25:44–61
5. Lootsma FA, Boonekamp PGM, Cooke RM, Oostvoorn Fvan (1990) Choice of a long-term strategy for the national electricity supply via scenario analysis and multi-criteria analysis. Europ J Oper Res 48:189–203

Optimization with Equilibrium Constraints: A Piecewise SQP Approach

PSQP

Daniel Ralph
University Melbourne, Melbourne, Australia

MSC2000: 90C30, 90C33

Article Outline

Keywords
Local Decomposition of the Feasible Set
The Algorithm
Convergence of PSQP
Affine Equilibrium Constraints
See also
References

Keywords

MPEC; Bilevel programming; Piecewise sequential quadratic programming method; Disjunctive programming; Parametric variational inequalities; Parametric complementarity problems; KKT conditions

The *piecewise sequential quadratic programming* (PSQP) method is a numerical method for solving certain mathematical programs with equilibrium constraints (MPEC), based on the classical sequential quadratic programming (SQP) method for nonlinear programming (NLP) problems [2,12]. This description draws on both [9] and [6], which extend the original proposal for PSQP [13] that was restricted to the case of MPEC with linear complementarity constraints. See [7] for a brief account of an application of PSQP to a problem in civil engineering. Its performance on randomly generated quadratic programs with affine equilibrium constraints is documented in [6] and also in [9,10].

PSQP can be applied directly to any MPEC whose lower-level problem is a mixed complementarity problem, and indirectly to any MPEC where the lower-level problem belongs to the class of variational inequalities (cf. also ▶ Variational inequalities) (VI) that can be written via its Karush–Kuhn–Tucker conditions (KKT conditions).

The formulation of the KKT-constrained MP is

$$\begin{cases} \min & f(x,y) \\ \text{s.t.} & G(x,y) \le 0, \\ & F(x,y) + \sum_{i=1}^{\ell} \lambda_i \nabla_y g_i(x,y) = 0, \\ & \lambda \ge 0,\ g(x,y) \le 0,\ \lambda^\top g(x,y) = 0, \end{cases} \tag{1}$$

where f $f\mathbf{R}^{n+m} \to \mathbf{R}$, G: $\mathbf{R}^{n+m} \to \mathbf{R}^s$ and $F\mathbf{R}^{n+m} \to \mathbf{R}^m$, are twice continuously differentiable; g: $\mathbf{R}^{n+m}$ $\mathbf{R}^\ell$ is thrice continuously differentiable; and $\nabla_y g_i(x, y)$ is the gradient with respect to y of g_i at (x, y). This is a special case of MPEC with mixed complementarity constraints. It includes the subclass of MPEC whose equilibrium constraints are nonlinear complementarity problems by taking $g(x, y) = -y$. Note that to ease notation, equality constraints have been omitted in the constraints both at the upper level ($G(x, y) \le 0$) and the lower level ($g(x, y) \le 0$); these can be included without any difficulties.

A selection of MPEC applications including Stackelberg games, network design, and design of mechanical structures, is given in the monograph [9]. Also note that in many cases the equilibrium constraints arise as

the first order conditions of a (usually convex) optimization problem, in which case an MPEC problem is closely related, if not equivalent to a bilevel program (cf. also ► Bilevel programming: Introduction, history and overview). The volume [1] is a good source of applications of hierarchical optimization; see also the volume containing [10].

For comparison, and also later reference in the section on mathematical programs with affine equilibrium constraints, consider the general MPEC with upper-level constraints $G(x, y) \leq 0$ and the equilibrium constraint that, given x, y solves the parametric variational inequality VI($F(x, \cdot)$, $C(x)$), i. e.

$$y \in C(x), \quad \forall y' \in C(x), \quad F(x, y)^\top (y' - y) \geq 0, \quad (2)$$

where $C(x) = \{y : g(x, y) \leq 0\}$. Under some conditions on the vector function g, any pair (x, y) satisfying these constraints is associated with a KKT multiplier λ such that (x, y, λ) is feasible for (1). One such condition is that g is affine, i. e. linear plus a constant; another is that each component function g_i is convex and that the *Slater constraint qualification* holds: There exists $\widehat{y}$ such that $g_i(x, \widehat{y}) < 0$ for all i. Conversely, under convexity of g, each feasible point (x, y, λ) of (1) satisfies the general MPEC constraints.

Local Decomposition of the Feasible Set

Let $\mathcal{F}^{\mathrm{KKT}}$ denote the set of feasible points $(x, y, \lambda) \in \mathbf{R}^{n+m+\ell}$ of (1). Let $\overline{w} = (\overline{x}, \overline{y}, \overline{\lambda}) \in \mathcal{F}^{\mathrm{KKT}}$. For any other feasible point (x, y, λ), the conditions

$$\lambda \geq 0, \quad g(x, y) \leq 0, \quad \lambda^\top g(x, y) = 0\,,$$

imply *complementarity*: For each i, either $\lambda_i = 0$ or $g_i(x, y) = 0$. It follows for each $(x, y, \lambda) \in \mathcal{F}^{\mathrm{KKT}}$ near $((\overline{x}, \overline{y}, \overline{\lambda}))$, that (x, y, λ) is also feasible for some nonlinear program of the form

$$\begin{cases} \min & f(x, y) \\ \text{s.t.} & G(x, y) \leq 0, \\ & L(x, y, \lambda) = 0, \\ & \forall i \in \mathcal{J}_1 \begin{cases} \lambda_i = 0, \\ g_i(x, y) \leq 0, \end{cases} \\ & \forall i \in \mathcal{J}_2 \begin{cases} \lambda_i \geq 0, \\ g_i(x, y) = 0, \end{cases} \end{cases} \quad (3)$$

where $L(x, y, \lambda) = F(x, y) + \sum_i \lambda_i \nabla_y g_i(x, y)$ and the index sets $\mathcal{J}_1$, $\mathcal{J}_2$ partition $\{1, \ldots, \ell\}$ such that $\mathcal{J}_1 \supseteq \{i : g_i(\overline{z}) < 0\}$ and $\mathcal{J}_2 \supseteq \{i : \lambda_i > 0\}$.

For each such pair $(\mathcal{J}_1, \mathcal{J}_2)$, the feasible set of (3) is called a *branch of the feasible set* $\mathcal{F}^{\mathrm{KKT}}$ at $\overline{w}$. Decomposition is given by an easy but critical observation: The union of the branches (3) of $\mathcal{F}^{\mathrm{KKT}}$ at a feasible point $\overline{w}$ is a neighborhood of $\overline{w}$ in $\mathcal{F}^{\mathrm{KKT}}$.

Each branch can be called a 'local piece' of $\mathcal{F}^{\mathrm{KKT}}$. The PSQP method takes advantage of this piecewise, or disjunctive, or decomposition approach to $\mathcal{F}^{\mathrm{KKT}}$, hence is one of the class of *disjunctive programming* methods.

A decomposition scheme at infeasible points is required, to allow for disjunctive methods that work with infeasible iterates. This decomposition is based on another easy observation that for a point $(\overline{z}, \overline{\lambda}) \in \mathcal{F}^{\mathrm{KKT}}$, where $\overline{z} = (\overline{x}, \overline{y})$:

$$\{i : g_i(\overline{z}) < 0\} = \left\{i : g_i(\overline{z}) + \overline{\lambda}_i < 0\right\}\,,$$
$$\left\{i : \overline{\lambda}_i > 0\right\} = \left\{i : g_i(\overline{z}) + \overline{\lambda}_i > 0\right\}\,.$$

Let $w = (x, y, \lambda) \in \mathbf{R}^{n+m+\ell}$ with $\lambda \geq 0$, and write $z = (x, y)$. Let $\mathcal{A}(w)$ denote the family of index set pairs $(\mathcal{J}_1, \mathcal{J}_2)$ that partition $\{1, \ldots, \ell\}$ and satisfy

$$\begin{cases} \mathcal{J}_1 \supseteq \{i : g_i(z) + \lambda_i < 0\}\,, \\ \mathcal{J}_2 \supseteq \{i : g_i(z) + \lambda_i > 0\}\,. \end{cases} \quad (4)$$

The Algorithm

Consider multiplier vectors, called *MPEC multipliers*, $\chi \in \mathbf{R}^s$ and $\pi \in \mathbf{R}^m$ and $\eta \in \mathbf{R}^\ell$ corresponding to the constraints $G(z) \leq 0$, $L(z, \lambda) = 0$ and the block of constraints $g_i(z) \leq (=)\ 0$, respectively; and define the *MPEC Lagrangian* as

$$\mathcal{L}^{\mathrm{MPEC}}(z, \lambda, \zeta, \pi, \eta) = f(z) + \zeta^\top G(z) - \pi^\top L(z, \lambda) + \eta^\top g(z)\,.$$

Multipliers corresponding to the constraints $\lambda_i \geq 0$ (or $= 0$) are omitted here and in the sequel, since these turn out not to play a role in the PSQP method as a result of being linear.

PSQP can now be presented. At iteration k, given $w^k = (z^k, \lambda^k)$ and a triple of multipliers $v^k = (\chi^k, \pi^k, \eta^k)$, pick an arbitrary member $(\mathcal{J}_1, \mathcal{J}_2)$ of $\mathcal{A}(w^k)$. Corresponding to this pair $(\mathcal{J}_1, \mathcal{J}_2)$, form the quadratic program associated with the NLP (3) at (z^k, λ^k); the variables of this quadratic program are given by the vector $dw = (dz, d\lambda)$, with $dz = (dx, dy)$:

$$\begin{cases} \min & \nabla f(z^k)^\top dz \\ & \quad + \frac{1}{2} dw^\top \nabla^2_{ww} \mathcal{L}^{\mathrm{MPEC}}(w^k, v^k) dw \\ \text{s.t.} & G(z^k) + \nabla G(z^k)\, dz \le 0, \\ & L(w^k) + \nabla L(w^k)\, dw = 0, \\ & \forall i \in \mathcal{J}_1 \begin{cases} (\lambda^k + d\lambda)_i = 0, \\ g_i(z^k) + \nabla g_i(z^k)^\top dz \le 0, \end{cases} \\ & \forall i \in \mathcal{J}_2 \begin{cases} (\lambda^k + d\lambda)_i \ge 0, \\ g_i(z^k) + \nabla g_i(z^k)^\top dz = 0. \end{cases} \end{cases} \tag{5}$$

Here gradients of real-valued functions are denoted by ∇, e. g. $\nabla f(z^k)$; whereas the derivative $\nabla G(z^k)$ is the $s\times(n+m)$ Jacobian matrix of G at z^k; and $\nabla^2_{ww}\mathcal{L}^{\mathrm{MPEC}}(w^k, v^k)$ is the Hessian or second-derivative matrix, with respect to w, of the MPEC Lagrangian at (w^k, v^k). A KKT tuple of this quadratic program will be used to define the next iterate.

The PSQP method for (1) is summarized below.

Recall, for use in Step 1, the definition of a stationarity: A point z is *stationary* for the general nonlinear program

$$\begin{cases} \min & f(z) \\ \text{s.t.} & h(z) = 0, \\ & g(z) \ge 0, \end{cases} \tag{6}$$

for smooth functions f, g, h, if there are vectors λ_h and λ_g, with dimensions matching $h(z)$ and $g(z)$, respectively, such that the KKT conditions hold: $\nabla f(z) + \nabla h(z)^\intercal \lambda_h + \nabla g(z)^\intercal \lambda_g = 0$, $h(z) = 0$, $g(z) \le 0$, $\lambda \ge 0$ and $\lambda^\intercal g(z) = 0$. See [6] for details on using approximate stationarity in Step 1, namely checking whether the KKT conditions approximately hold, instead of stationarity.

0. Let $w^0 = (z^0, \lambda^0) \in \mathbf{R}^{n+m} \times \mathbf{R}^\ell_+$, $v^0 = (\zeta^0, \pi^0, \eta^0) \in \mathbf{R}^{s+m+\ell}$, $\mathcal{A}^0 = \mathcal{A}(w^0)$, and $k = 0$.
1. (Direction finding.)
   Choose a pair $(\mathcal{J}_1^k, \mathcal{J}_2^k) \in \mathcal{A}^k$.
   IF the QP (5) is found to be the infeasible or unbounded below, THEN go to Step 3.
   ELSE find a stationary point dw of (5), with multipliers $v = (\zeta, \pi, \eta)$ as above.
   IF $dw = 0$ (hence w^k is stationary for (3)) THEN go to 3.
2. (Serious step.)
   Let $w^{k+1} = w^k + dw$, $v^{k+1} = v$, and $\mathcal{A}^{k+1} = \mathcal{A}(w^{k+1})$. Let $k = k + 1$ and go to Step 1.
3. (Null step.)
   Let $w^{k+1} = w^k$, $v^{k+1} = v^k$, $\mathcal{A}^k = \mathcal{A}^k \backslash \{(\mathcal{J}_1^k, \mathcal{J}_2^k)\}$, and $k = k + 1$.
4. (Stopping rule.)
   IF the stopping condition is satisfied THEN STOP ELSE go to 1.

Algorithm PSQP

At each iteration PSQP selects a nonlinear program, indexed by $(\mathcal{J}_1, \mathcal{J}_2)$, and updates (w^k, v^k) to (w^{k+1}, v^{k+1}) by applying one step of the SQP method to this subproblem. The idea of selecting one of possibly several subproblems at each iteration is based on the *Kojima–Shindo method* [8] for solving piecewise smooth equations $\Phi(x) = 0$, where the mapping $\Phi \mathbf{R}^n\}\mathbf{R}^n$ is the continuous selection of a finite family of smooth functions $\{\Phi^i\}$ each of which maps $\mathbf{R}^n$ to itself. Given $x^k \in \mathbf{R}^n$, an arbitrary index $i = i(k)$ is chosen with $\Phi(x^k) = \Phi^i(x^k)$, and then x^k is updated to x^{k+1} by applying a single Newton iteration to Φ^i at x^k.

Superlinear or quadratic convergence of PSQP and of the method of [8] can be shown under appropriate conditions.

Moreover, PSQP can be viewed as a localized version of previous disjunctive approaches to finding global solutions of MPEC rather than local solution or stationary points. For instance, papers on finding global optima are given in [11], and [5]. In connection with the latter on a branch and bound approach, note that the concept of the relaxed problem, like the problem (7), can be used to generate bounds for approximating the global optimal value of an MPEC. Both [5] and [11] address problems which can be cast as MPEC with

affine equilibrium constraints; these will be discussed in a later section.

To discuss stopping conditions for Step 4, observe that if all branches of $\mathcal{F}^{\mathrm{KKT}}$ at the current point w^k are such that w^k is stationary for each of the associated nonlinear programs (3), then w^k is stationary for the MPEC. This follows because the union of branches of $\mathcal{F}^{\mathrm{KKT}}$ at w^k form a neighborhood of w^k in $\mathcal{F}^{\mathrm{KKT}}$.

A) (Stopping condition A) $\mathcal{A}^k = \emptyset$.
Stopping condition A requires an exhaustive check of stationarity of w^k for each NLP (3) where $(\mathcal{J}_1, \mathcal{J}_2) \in \mathcal{A}(w^k)$. Thus, if w^k is a local minimizer, then verifying stationarity at w^k amounts to checking stationarity of the current point for 2^β nonlinear programs where β is the cardinality of the set $\mathcal{I}_0(w^k) = \{i\colon g_i(z^k) = 0 = \lambda_i^k\}$.
To provide a more efficient stopping rule, introduce the *relaxed nonlinear program* at w^k, in which the complementarity condition $\lambda_i\, g_i(x, y) = 0$ is relaxed for indices i in $\mathcal{I}_0(w^k)$. Let $\mathcal{I}_+(w^k) = \{i\colon g_i(z^k) < \lambda_i^k\}$, $\mathcal{I}_-(w^k) = \{i : g_i(z^k) > \lambda_i^k\}$. The relaxed NLP is:

$$\begin{cases} \min & f(x,y) \\ \text{s.t.} & G(x,y) \le 0, \\ & L(x,y,\lambda) = 0, \\ & g_i(x,y) \le 0 = \lambda_i, \quad \forall i \in \mathcal{I}_+(w^k), \\ & g_i(x,y) \le 0 \le \lambda_i, \quad \forall i \in \mathcal{I}_0(w^k), \\ & g_i(x,y) = 0 \le \lambda_i, \quad \forall i \in \mathcal{I}_-(w^k). \end{cases} \tag{7}$$

Our interest in the relaxed NLP stems from the obvious fact that if $w^k \in \mathcal{F}^{\mathrm{KKT}}$, then the feasible set of (7) contains every branch of $\mathcal{F}^{\mathrm{KKT}}$ at the current iterate w^k. Hence if the current iterate is stationary for (7), then it must also be stationary for the MPEC (1).

B) (Stopping condition B.) Either $\mathcal{A}^k = \emptyset$ or w^k is a stationary point of (7) with corresponding multipliers v^k (corresponding to G, L and g).

Stopping condition B, first used in [10], is an often effective heuristic to reduce the number of branches examined (null steps executed) in order to either identify a stationary point of the MPEC, or find a branch where progress can be made. It is guaranteed to verify or disprove stationarity by checking just one branch [10], provided that the active constraints at w^k of the relaxed nonlinear program (7) have linearly independent gradients. Also, see computational examples in [6,10], where the algorithm executes considerably fewer null steps when using stopping condition B instead of the exhaustive stopping condition A, even when the relaxed NLP has linearly dependent active gradients.

Convergence of PSQP

Superlinear local convergence of the PSQP algorithm can be shown if, as well as a second order condition, there is a uniqueness condition on the optimal multipliers associated with each nonlinear program (3) for relevant pairs of index sets $(\mathcal{J}_1, \mathcal{J}_2)$. The proof of convergence [9] relies on the convergence analysis of [2] and [12], which applies to stationary points of nonlinear programs such that the corresponding KKT multipliers are unique and the second order sufficient optimality condition holds. For superlinear convergence of PSQP it is also needed that the optimal multipliers be independent of the pairs $(\mathcal{J}_1, \mathcal{J}_2)$, so that the current value of the multipliers is (locally) a reasonable estimate for the NLP subproblem corresponding to any branch.

Recall the *second order sufficient condition* for the general nonlinear program (6); this condition will be needed below for the NLP associated with each branch of $\mathcal{F}^{\mathrm{KKT}}$ at a solution point of the KKT-constrained MP. Let $\overline{z}$ be a stationary point of the NLP (6). For convenience assume (as below) that the associated KKT multipliers λ_g, λ_h are unique. Let H be the Hessian matrix, with respect to z, at $= \overline{z}$ of the Lagrangian mapping $f(z) + \lambda_h^\top h(z) + \lambda_g^\top g(z)$; and $\mathcal{C}$ be the *critical cone* of (6) at $\overline{z}$, namely the set of vectors d in z-space such that $\langle \nabla f(\overline{z}), d\rangle = 0$, $\nabla h(\overline{z})d = 0$, and $\nabla g_I(\overline{z})d \le 0$, where $\nabla g(\overline{z})_I$ is the submatrix of the Jacobian of g at $\overline{z}$ consisting of rows i such that $i(\overline{z}) = 0$. The second order sufficient condition says that H is strictly copositive on $\mathcal{C}$, i. e. $d^\intercal\, Hd > 0$ for $0 \ne d \in \mathcal{C}$.

The main convergence result for PSQP is:

Theorem 1 *In the context of the KKT-constrained MPEC (1), let the functions f and F be twice continuously differentiable, and g be thrice continuously differentiable. Let $\overline{w} = (\overline{x}, \overline{y}, \overline{\lambda})$ be a point in $\mathcal{F}^{KKT}$ and $\overline{v} = (\overline{\zeta}, \overline{\pi}, \overline{\eta}) \in \mathbf{R}^{s+m+\ell}$ be such that, for each $(\mathcal{J}_1, \mathcal{J}_2)$ in $\mathcal{A}(\overline{w})$,*

1) *$\overline{w}$ is stationary for (3);*
2) *$\overline{v}$ is the unique KKT multiplier for (3) associated with $\overline{w}$; and*

3) the second order sufficient condition for (3) holds at $\overline{w}$.

Then $\overline{w}$ *is a strict local minimizer of the MPEC (1). Moreover, for each* (w^0, v^0) *sufficiently close to* $(\overline{w}, \overline{v})$, *the PSQP algorithm is well defined and produces a sequence* $\{(w^k, v^k)\}$ *that converges Q-superlinearly to* $(\overline{w}, \overline{v})$. *Finally, convergence is Q-quadratic if, in addition,* $\nabla^2 f$, $\nabla^2 F$ *and* $\nabla^3 g$ *are Lipschitz continuous near* $(\overline{x}, \overline{y})$.

If $(\overline{x}, \overline{y}, \overline{\lambda}) \in \mathcal{F}^{\mathrm{KKT}}$ is stationary for the relaxed NLP (7), it is clear that the linear independence constraint qualification for (7) at this point – linear independence of gradients of the constraints that are active at $(\overline{x}, \overline{y}, \overline{\lambda})$ – implies the uniqueness condition on the multipliers $\overline{v} = (\overline{\zeta}, \overline{\pi}, \overline{\eta})$ required in the above result.

The above convergence result is local in the sense that a starting point near to an eventual solution point is needed. To improve the performance of the method particularly from starting points that may be far from being optimal, it is typical in the context of nonlinear programming to use a line search or trust region strategy [3]. See [6] for a line search approach to MPEC where the objective function f is smooth and convex, the upper constraints are affine, and the lower or equilibrium constraints form an affine variational inequality. (PSQP also has special properties for this case as seen shortly.)

In this case, assuming $w^k = (z^k, \lambda^k)$ is feasible and $dw = (dz, d\lambda)$ is the direction computed by solving the QP (5), the update in Step 2 of the PSQP method is modified to choose a step size $t > 0$ such that $f(z^k + t\, dz) < f(z^k)$, amongst other conditions; and then set $w^{k+1} = w^k + t\, dw$. Attempts to globalize PSQP for problems with nonlinear constraint functions, by adding penalty terms to the objective to replace these constraints, and then using line search or trust region strategies, are the subject of research. See [14,15], for example.

Quasi-Newton variants on PSQP, in which the Hessian matrix $\nabla^2_{ww}\mathcal{L}^{\mathrm{MPEC}}(w^k, v^k)$ is replaced by an easily updated approximation, for example, are also the subject of research. See [3] for motivation from nonlinear programming.

Affine Equilibrium Constraints

An important special case of the PSQP method is its application to *mathematical programs with affine equilibrium constraints* (MPAEC), that is, problems whose equilibrium constraints are in the form of affine variational inequalities (AVI). MPEC with equilibrium constraints which can be expressed as in linear complementarity problems (cf. also ► Linear complementarity problem), such as [5,11,13], fall into this subclass.

Specifically, we consider the MPAEC formulated with KKT constraints, for which the upper-level constraint function G, and the functions F and g in the lower-level VI are affine (linear plus a constant):

$$\begin{cases} \min & f(x, y) \\ \text{s.t.} & Ax + By + a \le 0, \\ & Px + Qy + q + E^\top \lambda = 0, \\ & Dx + Ey + b \le 0, \quad \lambda \ge 0, \\ & \lambda^\top (Dx + Ey + b) = 0, \end{cases} \tag{8}$$

for some $A \in \mathbf{R}^{s\times n}$, $B \in \mathbf{R}^{s\times m}$, $a \in \mathbf{R}^s$; $P \in \mathbf{R}^{m\times n}$, $Q \in \mathbf{R}^{m\times m}$, $q \in \mathbf{R}^m$; and $D \in \mathbf{R}^{\ell\times n}$, $E \in \mathbf{R}^{\ell\times m}$, $b \in \mathbf{R}^\ell$. This is the problem (1) with

$$\begin{aligned} G(x, y) &= Ax + By + a\,, \\ F(x, y) &= Px + Qy + q\,, \\ g(x, y) &= Dx + Ey + b\,. \end{aligned} \tag{9}$$

Note that a point (x, y, λ) is feasible for (8) if and only if $G(x, y) \le 0$ and y solves the affine variational inequality VI($F(x, \cdot)$ C(x)) described in (2), where $C(x) = \{y : g(x, y) \le 0$ is a polyhedral convex set. Thus the KKT-constrained MP (8) is equivalent to the MPAEC problem in its standard form:

$$\begin{cases} \min & f(x, y) \\ \text{s.t.} & Ax + By + a \le 0\,, \\ & y \text{ solves the AVI (2)}\,. \end{cases} \tag{10}$$

Some differences between the special PSQP method for MPAEC and the general method should be noted: First, in the QP subproblem (5) solved in Step 1, the objective function simplifies to

$$\nabla f(z^k)^\top\, dz + \frac{1}{2}\, dz^\top \nabla^2 f(z^k)\, dz\,.$$

Second, therefore, the vector v^k of MPEC multipliers is not needed except perhaps if stopping rule B is employed. Third, the specialized algorithm requires a feasible starting point and maintains feasibility of iterates. A heuristic 'phase-1' procedure [6] for finding a feasible

starting point is to find a local minimizer or stationary point of the indefinite quadratic program:

$$\begin{cases} \min & \lambda^\top F(x,y) \\ \text{s.t.} & G(x,y) \le 0, \\ & F(x,y) + E^\top \lambda = 0, \\ & \lambda \ge 0, \\ & g(x,y) \le 0. \end{cases}$$

See [4] for conditions under which all stationary points of the phase-1 problem are indeed feasible for the MPAEC.

Similar to SQP applied to nonlinear programs with linear constraints, the PSQP algorithm specialized to MPEC with affine KKT constraints exhibits locally superlinear convergence if the Hessian of the objective function $\nabla^2 f(\overline{x}, \overline{y})$ satisfies a second order sufficient condition, regardless of whether either the KKT multiplier vector λ or the MPEC multiplier vectors χ, π, η are unique. To make this precise, suppose $\overline{z} = (\overline{x}, \overline{y})$ is feasible for the MPAEC (10), and let $\Lambda(\overline{z})$ be the set of all KKT multipliers λ such that $(\overline{z}, \lambda)$ is feasible for (8), i. e. $(\overline{z}, \lambda) \in \mathcal{F}^{\mathrm{KKT}}$. Now for each $\lambda \in \Lambda(\overline{z})$ and $(\mathcal{J}_1, \mathcal{J}_2) \in \mathcal{A}(\overline{z}, \lambda)$, let $\mathcal{C}^{\mathrm{KKT}}(\mathcal{J}_1, \mathcal{J}_2)$ be the critical cone of the linearly constrained NLP (3) at $(\overline{z}, \lambda)$; and define the associated z-critical cone $\mathcal{C}_z\{^{\mathrm{KKT}}\}(\mathcal{J}_1, \mathcal{J}_2)$ as the set of vectors dz such that $(dz, d\lambda) \in \mathcal{C}^{\mathrm{KKT}}(\mathcal{J}_1, \mathcal{J}_2)$ for some $d\lambda \in \mathbf{R}^m$. The second order sufficient condition for $(\overline{z}, \lambda)$ to be a local minimizer of (8) for each $\lambda \in \Lambda(\overline{z})$ is that the Hessian matrix $\nabla^2 f(\overline{z})$ be copositive on each of these z-critical cones.

The second order conditions act in the following way:

Theorem 2 *Consider the MPAEC problems (8) and (10) where the function f is twice continuously differentiable, and the functions G, F and g are given by (9). Let $\overline{z}$ be a feasible point of (10) and $\Lambda(\overline{z})$ be the (nonempty) set $\{\lambda \in \mathbf{R}^m : (\overline{z}, \lambda) \in \mathcal{F}^{KKT}\}$. Suppose for each $\lambda \in \Lambda(\overline{z})$ and $(\mathcal{J}_1, \mathcal{J}_2)$ in $\mathcal{A}(\overline{z}, \lambda)$ that $(\overline{z}, \lambda)$ is stationary for the NLP (3); and that $\nabla^2 f(\overline{z})$ is strictly copositive on the z-critical cone $\mathcal{C}_z^{KKT}(\mathcal{J}_1, \mathcal{J}_2)$ defined above. Then $\{\overline{z}\} \times \Lambda(\overline{z})$ consists of local minimizers of the KKT-constrained MP (8); in fact, $\overline{z}$ is a strict local minimizer of the MPAEC (10).*

The local convergence properties of PSQP applied to MPAEC are now presented.

Theorem 3 *Let f, G, F, g, $\overline{z}$, and $\Lambda(\overline{z})$ be as above; and the associated first- and second order conditions hold. Then for each (z^0, λ^0) near $\overline{z} \times \Lambda(\overline{z})$, the PSQP algorithm specialized to MPAEC is well defined and produces a sequence $\{(z^k, \lambda^k)\}$ such that $\{z^k\}$ converges Q-superlinearly to $\overline{z}$. Convergence of $\{z^k\}$ is Q-quadratic if, in addition, $\nabla^2 f$ is Lipschitz near $\overline{z}$.*

See also

- ▶ Feasible Sequential Quadratic Programming
- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Successive Quadratic Programming
- ▶ Successive Quadratic Programming: Applications in Distillation Systems
- ▶ Successive Quadratic Programming: Applications in the Process Industry
- ▶ Successive Quadratic Programming: Decomposition Methods
- ▶ Successive Quadratic Programming: Full Space Methods
- ▶ Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Anandalingam G, Friesz T (1992) Hierarchical optimization. Ann Oper Res 34
2. Bonnans JF (1994) Local analysis of Newton-type methods for variational inequalities and nonlinear programming. Appl Math Optim 29:161–186
3. Fletcher R (1987) Practical methods of optimization. Wiley, New York
4. Fukushima M, Pang JS (1998) Some feasibility issues in mathematical programs with equilibrium constraints. SIAM J Optim 8:673–681
5. Hansen P, Jaumard B, Savard G (1992) New branch-and-bound rules for linear bilevel programming. SIAM J Sci Statist Comput 13:1194–1217
6. Jiang H, Ralph D (to appear) QPECgen, a MATLAB generator for mathematical programs with quadratic objectives and affine variational inequality constraints. Comput Optim Appl (to appear)
7. Jiang H, Ralph D, Tin–Loi F (1997) Identification of yield limits as a mathematical program with equilibrium constraints. In: Grzebieta RH, Al–Mahaidi R, Wilson JL (eds) The Mechanics of Structures and Materials: Proc. 15th Aus-

tralasian Conf. the Mechanics of Structures and Materials. A.A. Balkema, Leiden, pp 399–404
8. Kojima M, Shindo S (1986) Extensions of Newton and quasi-Newton methods to systems of PC1 equations. J Oper Res Soc Japan 29:352–374
9. Luo Z-Q, Pang JS, Ralph D (1997) Mathematical programs with equilibrium constraints. Cambridge University Press, Cambridge
10. Luo Z-Q, Pang JS, Ralph D (1998) Piecewise sequential quadratic programming for mathematical programs with nonlinear complementarity constraints. In: Migdalas A, Pardalos PM, Värbrand P (eds) Multilevel Optimization: Algorithms and Applications. Kluwer, Dordrecht, pp 209–229
11. Maier G, Giannessi F, Nappi A (1982) Indirect identification of yield limits by mathematical programming. Eng Structures 4:86–98
12. Pang JS (1993) Convergence of splitting and Newton methods for complementarity problems: An application of some sensitivity results. Math Program 58:149–160
13. Ralph D (1996) Sequential quadratic programming for mathematical programs with linear complementarity constraints. In: May RL, Easton AK(eds) Proc. seventh Conf. Computational Techniques and Applications (CTAC95). Sci. Press, Marrickville, pp 663–668
14. Scholtes S, Stöhr M (1999) Exact penalization of mathematical programs with equilibrium constraints. SIAM J Control Optim 37:617–652
15. Stöhr M (1999) Nonsmooth trust region methods and their applications to mathematical programs with equilibrium constraints. PhD Thesis, University Karlsruhe, Shaker, Aachen, 2000

Optimization in Leveled Graphs

Petra Mutzel
Institute Computergraphik und Algorithmen Techn., University Wien, Wien, Austria

MSC2000: 90C35

Article Outline

Keywords
Multiple Sequence Alignment
Leveled Crossing Minimization
Level Planarization
See also
References

Keywords

Leveled graphs; Integer programming; Crossing minimization; Planarization; Multiple sequence alignment

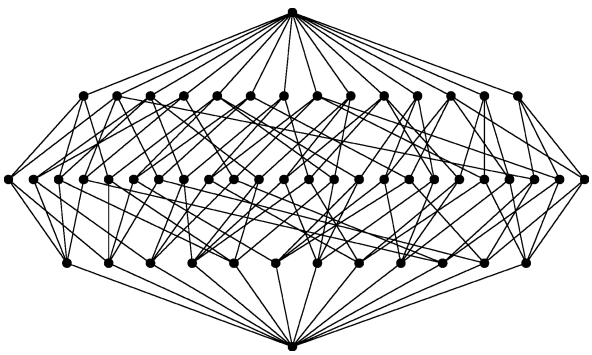
A *k-leveled graph* or a *k-level hierarchy* is defined as a graph $G = (V, E) = (V_1, \ldots, V_k, E)$ with vertex sets $V_1, \ldots, V_k$, $V = V_1 \cup \cdots \cup V_k$, $V_i \cap V_j = \emptyset$ for $i \neq j$, and an edge set E connecting vertices in levels V_i and V_j with $i \neq j$ $(1 \leq i, j \leq k)$. V_i is called the *ith level*. In a geometric representation of a k-leveled graph, the vertices in each level V_i are drawn on a horizontal line L_i with y-coordinate $k- i$, and the edges are drawn strictly monotone, i. e., an edge $(v_i, v_j) \in E$, $v_i \in V_i$, $v_j \in V_j$, $i < j$, is drawn with decreasing y-coordinates. Essentially, a k-leveled graph is a k-partite graph that is drawn in a special way.

A *proper k-leveled graph* is a k-leveled graph $G = (V_1, \ldots, V_k, E)$ in which any edge in E connects vertices in two consecutive levels V_i and V_{i+1} for $i \in \{1, \ldots, k-1\}$. Figure 1 shows a proper leveled graph on $k = 4$ levels. This graph represents the face lattice of the *cuboctahedron* [4].

Optimization problems in leveled graphs arise in applications in *computational biology* and in *automatic graph drawing*.

Multiple Sequence Alignment

In computational biology the vertices in each level V_i represent letters of a sequence S_i over a finite alphabet Σ. The optimization problem which arises is the *multiple sequence alignment* problem. Here, the k sequences $S_1, \ldots, S_k$ should be aligned so that the cost of the align-



Optimization in Leveled Graphs, Figure 1

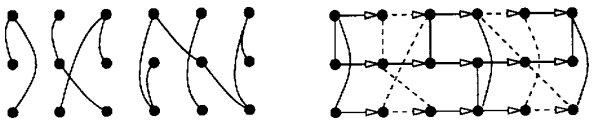
ment is maximized. An alignment can be interpreted as an array with k rows, one row for each S_i. Two letters of distinct sequences are said to be aligned if they are placed in the same column. There are many ways to measure the quality of an alignment, leading to different problem formulations. One of them is the *maximum weight trace* formulation introduced in [14]. Here, the letters of the sequences $S_i = (s_{i1}, \ldots, s_{in_i})$ are viewed as vertices in level i in a k-leveled graph $G = (V_1, \ldots, V_k, E)$. Every edge $e \in E$ has a nonnegative weight representing the gain of aligning the endpoints of the edge. We say that an alignment $\widehat{S}$ *realizes* an edge if it places the endpoints into the same column of the alignment array.

The set of edges realized by an alignment $\widehat{S}$ is called the *trace* of $\widehat{S}$, and the weight of an alignment $\widehat{S}$ is the sum of the weights of all edges in the trace of $\widehat{S}$. The goal is to compute an alignment $\widehat{S}$ of maximum weight.

The maximum weight trace problem is *NP*-hard, and can be solved in polynomial time for fixed k. A dynamic programming approach gives an algorithm with time complexity $O(k^2 2^k N)$ and space complexity $O(N)$, where $N = \prod_i n_i$, which is feasible only for very small problem instances. J. Kececioglu [15] presented a branch and bound algorithm whose implementation could optimally align six sequences of length 250 in a few minutes.

K. Reinert et al. [19] presented a first formulation as an *integer linear program*. It is based on a graph theoretical characterization of traces given in [19]. For this, the alignment graph $G = (V_1, \ldots, V_k, E)$ is extended to a mixed graph $G' = (V_1, \ldots, V_k, E, H)$ by adding a set of directed edges $H = \{(v_{ij}, v_{ij+1}): 1 \le i \le k, 1 \le j < n_i\}$. This graph is called the *extended alignment graph*. The weight function is extended to $E \cup H$ by assigning weight 0 to all the edges in H. A cycle in G' is called a *mixed cycle* if it contains at least one arc of H. In [19] it has been shown that an edge set in $T \subseteq E$ is a trace in $G = (V, E)$ if and only if there is no mixed cycle in $G' = (V, T, H)$. This characterization can be strengthened to substitute 'mixed cycle' by 'critical mixed cycle' which is essentially a simple mixed cycle visiting each sequence at most once.

Figure 2 shows an alignment graph and its extended alignment graph. Two critical mixed cycles are shown by the dotted lines. A formulation in the form of an integer linear program is now straightforward. Let $G' = (V, E, H)$ be an extended alignment graph. For every edge $e \in E$ we introduce a variable $x_e \in \{0, 1\}$ indicating whether e is in the trace or not. Let w_e denote the alignment weight of edge $e \in E$. The formulation is as follows:



Optimization in Leveled Graphs, Figure 2

$$\begin{cases} \max & \sum_{e \in E} w_e x_e \\ \text{s.t.} & \sum_{e \in C \cap E} x_e \le |E \cap C| - 1, \\ & \forall \text{ critical mixed cycles } C \text{ in } G' \\ & x_e \in \{0, 1\}, \quad \forall e \in E. \end{cases}$$

This integer linear program can be solved via a *branch and cut* approach based on *polyhedral combinatorics*. Reinert et al. define the *trace polytope* as the convex hull of the characteristic vectors of all possible traces of $G = (V, E)$. Their investigation leads to some classes of tight inequalities which can be used in a branch and cut algorithm. An implementation of a branch and cut algorithm based on ABACUS [13] could optimally align 15 sequences (arising from prion proteins) of length 230 within 2.5 hours of computation time on a SparcStation Ultra 1/170.

For two sequences, Reinert et al. have given a complete description of the trace polytope. The set $(E, \Theta) = (E, \{T: T \subseteq E \text{ is a trace in } G\})$ is an independence system, since $\emptyset \in \Theta$ and for any $T_1 \in \Theta$ and $T_2 \subseteq T_1$, also $T_2 \in \Theta$. The minimal dependent subsets or *circuits* of (E, Θ) have size two. Hence, the set (E, Θ) is a 2-regular independence system. A set $F \subseteq E$ is a *clique* in a 2-regular system (E, Θ) if $\|F\| \ge 2$ and all $\binom{\|F\|}{2}$ 2-subsets of F are circuits of (E, Θ).

For two sequences, the trace polytope is completely described by the trivial inequalities $0 \le x_e \le 1$ and by the clique inequalities $\sum_{e \in C} x_e \le 1$, where C is a maximal clique in the independence system (E, Θ). Since the separation problem for the clique inequalities can be solved in polynomial time, this provides an algorithm to solve the maximum weight trace problem for two sequences in polynomial time.

An efficient combinatorial algorithm for solving the maximum weight trace problem for two sequences has been given in [14]. Kececioglu has shown how to transform the problem into the heaviest increasing subsequence problem. This can be solved in time $O(n \log n)$, where n is the length of the sequence [9].

One can show that the maximum weight trace problem is equivalent to the 2-level planarization problem with two fixed levels arising in automatic graph drawing (see the section 'Level Planarization'). For further information on multiple sequence alignment or on combinatorial optimization problems in computational biology, see, e. g., [23].

Leveled Crossing Minimization

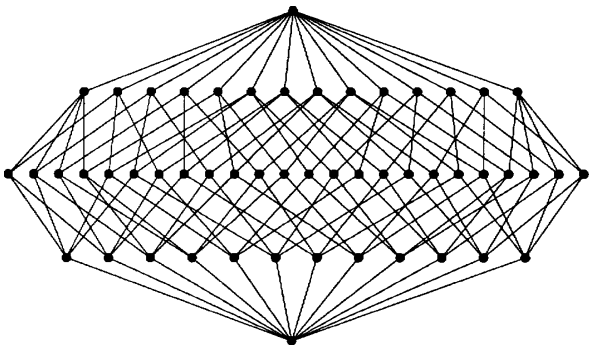
In *automatic graph drawing* the task is to compute a layout of a given graph that is easy to read and understand. Applications include flow diagrams, organization charts, entity-relationship diagrams, or PERT-diagrams.

A common method for drawing directed graphs is, as a first step, to partition the vertices into a set of k levels so that no edges have their endpoints in the same level and most of the edges point downwards. This can easily be done using a depth-first-search or a topological sorting algorithm when the graph is acyclic. The crucial step is the second one. Given a k-leveled graph, the vertices within the levels need to be permuted so that the resulting drawing is easy to read [22]. A widely used criterion for a 'nice' drawing is a minimum number of crossings.

The *k-level crossing minimization* problem arises: Given a k-leveled graph $G = (V_1, \ldots, V_k, E)$, permute the vertices within the levels so that the resulting geometric representation of G contains a minimum number of crossings.

The k-level crossing minimization problem is *NP*-hard, even when the numbers of levels is two [5]. The problem can be solved in polynomial time for 2-leveled permutation graphs [21] and for 2-leveled trees [20].

An integer linear programming formulation has been given in [10]. This is the first approach attacking the k-level crossing minimization problem via polyhedral combinatorics. Recently (1999), P. Healy and A. Kuusik [8] have shown that the original formulation can be tightened using additional inequalities derived in the so-called vertex-exchange graph. They are able to solve instances with up to 100 vertices, 100 edges, and 8 levels to provable optimality.



Optimization in Leveled Graphs, Figure 3

Figure 3 shows a k-level crossing minimum drawing of the face lattice graph shown in the first figure of this article. It turns out to be a symmetrical drawing.

F. Shahrokhi et al. [20] have suggested algorithms approximating the 2-level minimum crossing number by a factor of $O(n \log \log n)$ and $O(\log^2 n)$, respectively, for a certain class Γ of graphs. The approximation algorithms are based on the relationship between the 2-level crossing minimization problem and the linear arrangement problem. The linear arrangement problem for a graph $G = (V, E)$ is to find a bijection $f: V \to \{1, \ldots, \|V\|\}$ of minimum length. The length of the bijection f is given by $L_f = \sum_{(v,w) \in E} \| f(v) - f(w) \|$.

Shahrokhi et al. have shown that for a certain graph class Γ the order of magnitude for the optimal number of crossings is bounded from below, and above, respectively, by the minimum degree times the optimal arrangement value, and by the arboricity times the optimal arrangement value. The *arboricity* a_G of a graph $G = (V_G, E_G)$ is defined as

$$\max_H \left\lceil \frac{|E_H|}{|V_H| - 1} \right\rceil ,$$

where the maximum is taken over all subgraphs H of G with $\| V_H \| \geq 2$. Equivalent to a_G is the minimum number of edge disjoint acyclic subgraphs needed to cover G. The graph class Γ includes, e. g., connected 2-level graphs G of degree at most a constant k with $\|E\| \geq (1 + \gamma) \|V\|$, where $\gamma > 0$ is fixed. It also includes regular graphs, degree bounded graphs, and genus bounded graphs, which are not too sparse.

In practice, the crossing minimization problem for k-leveled graphs is reduced to a series of 2-level crossing minimization problems in the following way: In a preprocessing step, we add artificial vertices to the levels L_i for all the edges traversing $L_i (i = 2, \ldots, k-1)$. For $i = 1, \ldots, k-1$, we solve the 2-level crossing minimization problem for the two adjacent levels L_i and L_{i+1} with L_i fixed, repermuting the vertices on level L_{i+1}. Then we go backward, fixing level L_i and repermuting the vertices on level L_{i-1} for $i = k, k-1, \ldots, 2$. The heuristic consists of repeating these two loops until no more improvement is obtained.

Unfortunately, the 2-level crossing minimization problem in which the permutation of the vertices in one level is fixed is also *NP*-hard [3]. Therefore, a lot of effort went into the design of efficient heuristics (see, e. g., [1,22]). P. Eades and N. Wormald [3] have shown that the drawings constructed using the median heuristic lead to a number of crossings that is within a factor of three times the optimal crossing number.

M. Jünger and P. Mutzel [12] transformed the 2-level crossing minimization problem with a fixed level to a *linear ordering problem*:

Any solution is obviously completely specified by the fixed permutation π_1 of V_1 and a permutation π_2 of V_2. For $k = 1, 2$ let $y_{ij}^k = 1$ if $\pi_k(i) < \pi_k(j)$ and 0 otherwise. Thus π_k $(k = 1, 2)$ is uniquely characterized by the vector $y^k \in \{0, 1\}^{\binom{n_k}{2}}$. Let $n_1 = \| V_1 \|$, $n_2 = \| V_2 \|$, $m = \|E\|$, and let $N(v) = \{ w \in V : (v, w) \in E \}$ denote the set of neighbors of $v \in V = V_1 \cup V_2$ in G. The number of edge crossings between the levels V_1 and V_2 is given by

$$C(\pi_2) = \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} \sum_{k \in N(i)} \sum_{l \in N(j)} y_{kl}^1 y_{ji}^2 + y_{lk}^1 y_{ij}^2 .$$

Let

$$c_{ij} = \sum_{k \in N(i)} \sum_{l \in N(j)} y_{lk}^1$$

denote the number of crossings among the edges adjacent to i and j if $\pi_2(i) < \pi_2(j)$. Then

$$\begin{aligned} C(\pi_2) &= \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} c_{ij} y_{ij}^2 + c_{ji}(1 - y_{ij}^2) \\ &= \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} (c_{ij} - c_{ji}) y_{ij}^2 + \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} c_{ji} . \end{aligned}$$

For $n = n_2$, $y_{ij} = y_{ij}^2$ and $a_{ij} = c_{ij} - c_{ji}$ we solve the *linear ordering problem*

$$\text{(LO)} \begin{cases} \min & \displaystyle\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a_{ij} y_{ij} \\ & 0 \le y_{ij} + y_{jk} - y_{ik} \le 1, \\ & \quad 1 \le i < j < k \le n, \\ & y_{ij} \in \{0, 1\}, \\ & \quad 1 \le i < j \le n. \end{cases}$$

If z is the optimum value of the linear ordering problem, then

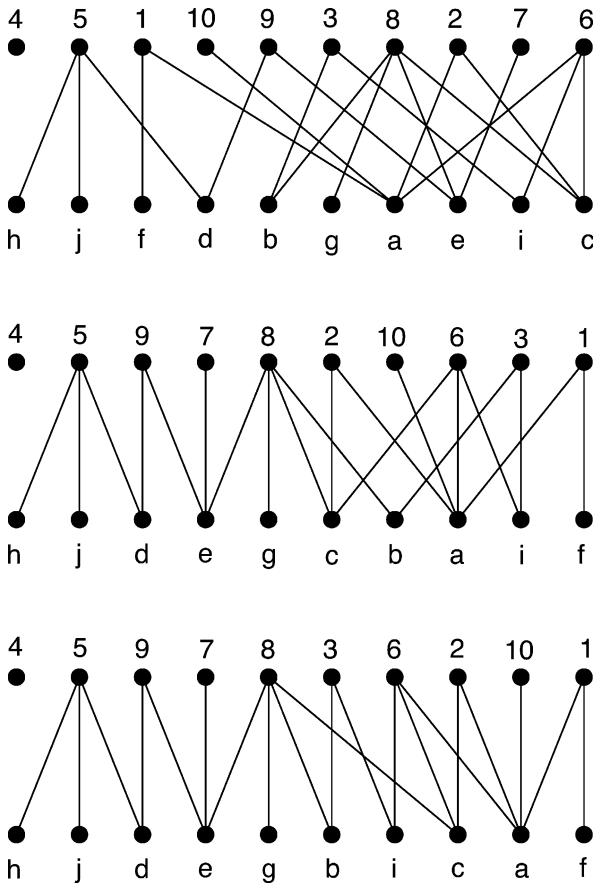
$$z + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ji}$$

is the minimum number of crossings in the corresponding 2-level drawing. The constraints of (LO) guarantee that the solutions indeed precisely correspond to all permutations π_2 of V_2.

Jünger and Mutzel [12] have shown that the resulting linear ordering problem can be solved efficiently via a branch and cut algorithm for instances containing up to 250 vertices per level. Extensive computational experiments show that the running time of their exact branch and cut algorithm is comparable to that of the widely used heuristics for instances with up to 60 vertices per level.

They combined this branch and cut algorithm with a branch and bound algorithm in order to achieve a practically efficient exact algorithm for the 2-level crossing minimization problem for instances with up to 15 vertices on the smaller level. Computational experiments have shown that the results achieved via heuristic methods are far from the optimum solution (see [12]).

In Fig. 4 an example of a 2-level graph with 20 edges is shown. The first two drawings have been computed via heuristics (LR-heuristic and barycenter heuristic, respectively) and have 30 and 10 crossings, respectively. The third one is the optimum solution with only 4 crossings. This is only one example showing that it is worth searching for better algorithms for leveled graph drawing.



Optimization in Leveled Graphs, Figure 4

Level Planarization

Several authors have suggested minimizing the number of edges 'producing' crossings instead of minimizing the number of crossings in hierarchical drawings.

Figure 5 shows two drawings of the same graph. Figure 5a) has been generated after first removing the edges that 'produce' crossings, and then drawing the remaining graph without crossings before finally inserting the four edges again. Although the drawing shown in Fig. 5a) has 34 crossings, that is 41% more crossings than the crossing minimum drawing shown in Fig. 5b), the reader will not recognize this fact.

This example motivates careful consideration of the *k-level planarization problem*. A k-level graph G is *level planar* if a k-level representation of G exists in which no two edges cross except at common endpoints. Level planarity can be tested in linear time [11]. Given a k-level graph $G = (V_1, \ldots, V_k, E)$ with weights $w_e > 0$ on the edges, the *k-level planarization problem* is to extract a level planar subgraph $G' = (V_1, \ldots, V_k, F)$, $F \subseteq E$, of maximum weight, i. e., the sum $\sum_{e \in F} w_e$ is maximum. The k-level planarization problem is *NP*-hard, even for $k = 2$ levels.

So far (1999), no exact algorithm for the k-level planarization problem on $k \geq 3$ levels is known. For $k = 2$, Mutzel [17] has presented a branch and cut algorithm based on polyhedral studies of the associated polytope. The integer programming formulation for the 2-level planarization problem is based on the following characterization of 2-level planar graphs (first presented in [7]).

A 2-level graph is 2-level planar if and only if it contains no cycle and no double claw.

Figure 6 shows a double claw (Fig. 6a) and a cycle (Fig. 6b). Although both graphs are planar bipartite graphs, they are not 2-level planar (see Fig. 6c) for the cycle). For double claw free graphs, the 2-level planarization problem is equivalent to the maximum forest subgraph problem that can be solved via a simple greedy algorithm. Shahrokhi et al. [20] have given a linear time algorithm for solving the 2-level planarization problem for 2-level acyclic graphs.

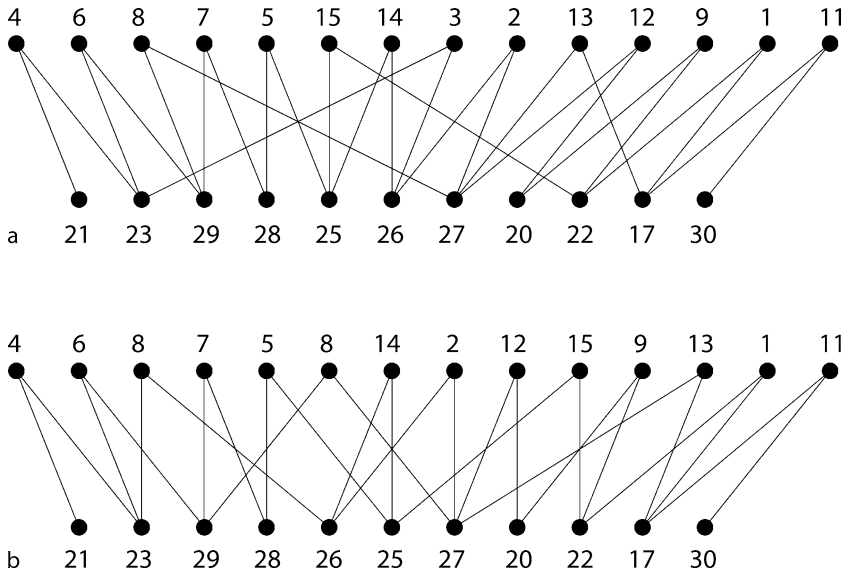
In order to get an integer linear programming formulation for the 2-level planarization problem, we introduce variables for all edges $e \in E$ of the given 2-level graph $G = (V_1, V_2, E)$. For any set $P \subseteq E$ of edges we define a characteristic vector $\chi^P \in \mathbf{R}^{\|E\|}$ with the ith component $\chi^P(e_i)$ getting value 1 if $e_i \in P$, and 0 otherwise. Any 0 –1 vector $x^\intercal = (x_{e_1}, \ldots, x_{e_{\|E\|}})$, that is the characteristic vector of a 2-level planar graph satisfies the following inequalities:

$$\sum_{e \in C} x_e \leq |C| - 1, \quad \forall \text{ cycles } C \subseteq E,$$

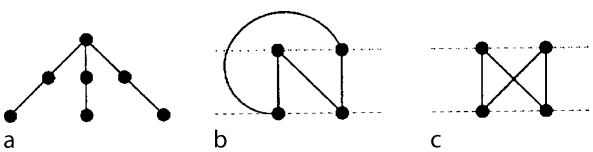
$$\sum_{e \in T} x_e \leq |T| - 1, \quad \forall \text{ double claws } T \subseteq E,$$

$$x_e \in \{0, 1\}, \quad \forall e \in E,$$

and vice versa. Hence, solving the integer linear system $\max \sum_{e \in E} w_e x_e$ with the constraints described above will give us the solution of the maximum 2-level planar subgraph problem for a given graph $G = (V_1, V_2, E)$ with weights w_e on the edges $e \in E$.



Optimization in Leveled Graphs, Figure 5



Optimization in Leveled Graphs, Figure 6

The computational experiments presented in [17] show that although most of the instances (from 20 to 100 vertices) could not be solved to optimality, solutions could be obtained which are provably within 5% of the optimum value.

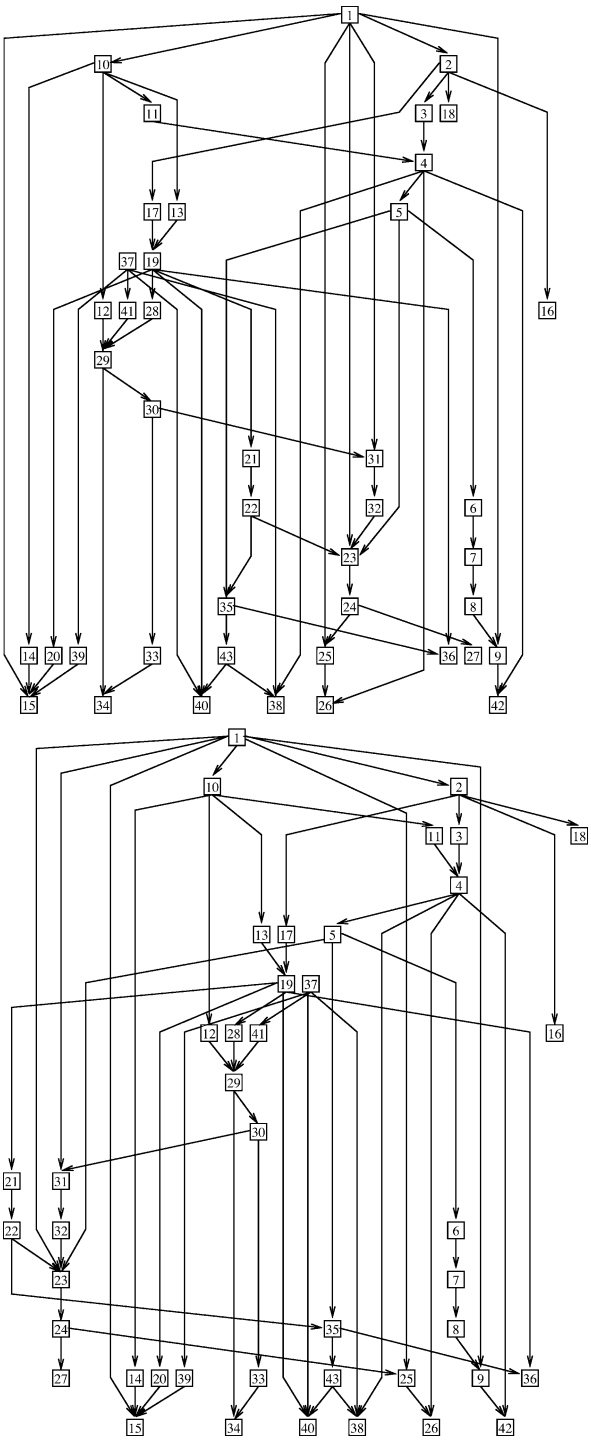
As in the 2-level crossing minimization problem, the case in which the permutation in one of the two levels is fixed has also been investigated [18]. Unfortunately, this version of the problem is also *NP*-hard [2].

While the fixed version of the crossing minimization problem turned out to be 'easier' to attack in practice than the free version, unfortunately, this is not true for the one level fixed version of the 2-level planarization problem. The integer programming formulation suggested in [18] contains variables $y_{ij} \in \{0, 1\}$ for $i < j$, $i = 1, \ldots, n_2 - 1$, $j = i + 1, \ldots, n_2$ coding the permutation π_2 of the vertices in level V_2 and variables $x_e \in \{0, 1\}$ for $e \in E$ coding the edges contained in the subgraph. In the following integer linear programming formulation for the one level fixed 2-level planarization problem let (p, i) and (q, j) be edges in E:

$$
\begin{cases}
\max & \sum_{e \in E} w_e x_e \\
& 0 \le y_{ij} + y_{jk} - y_{ik} \le 1, \\
& \quad 1 \le i < j < k \le |V_2|, \\
& y_{ij} \in \{0, 1\}, \\
& \quad 1 \le i < j < k \le |V_2|, \\
& y_{ij} + x_{(p,i)} + x_{(q,j)} \le 2, \\
& \quad i < j, \quad \pi_1(q) < \pi_1(p), \\
& -y_{ij} + x_{(p,i)} + x_{(q,j)} \le 1, \\
& \quad i < j, \quad \pi_1(p) < \pi_1(q), \\
& x_e \in \{0, 1\}, \quad \forall e \in E.
\end{cases}
$$

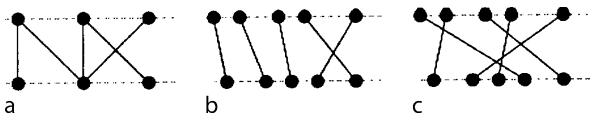
The first two constraints require the variable vector y to respect a linear ordering π_2. The last three constraints are responsible for introducing no crossings with respect to the ordering π_2 coded by y.

Mutzel and R. Weiskircher have shown that all the tight inequalities of the linear ordering polytope (see, e. g., [6]) are still tight inequalities for the polytope investigated here. In particular, all the knowledge about the linear ordering polytope can be used in order to get a practically efficient algorithm. Hence, the problem considered here is tightly connected with the 2-level crossing minimization problem in which the permutation of one level is fixed.



Optimization in Leveled Graphs, Figure 7

Figure 7 shows two drawings computed via iteratively solving the one level fixed 2-level crossing minimization problem and the 2-level planarization problem with one fixed level. In both drawings, the subproblems have been computed optimally.



Optimization in Leveled Graphs, Figure 8

A comparison of both drawings shows that it is worthwhile to also consider planarization and not only crossing minimization in hierarchical graph drawing. The ultimate goal is to solve these problems not levelwise but in one step. This will improve hierarchical drawings tremendously (see [12]). The authors in [18] conjecture that it will be easier to solve practical instances of the k-level planarization problem than of the k-level crossing minimization problem.

The question arises whether the 2-level planarization problem can be solved efficiently when the permutations of the vertices in both levels are fixed. This problem, however, is equivalent to the maximum weight trace problem for two sequences (see the section 'Multiple Sequence Alignment'). The transformation in both directions is indicated in Fig. 8.

If the graph shown in Fig. 8a) is an instance of a fixed 2-level planarization problem, it is equivalent to solving the maximum weight trace problem on the instance shown in Fig. 8b). In the case that Fig. 8a) shows an instance of the maximum weight trace problem, it is equivalent to solving the fixed 2-level planarization problem in the graph shown in Fig. 8c).

It follows that the fixed 2-level planarization problem can be solved in polynomial time. Moreover, we know the complete description of the associated polytope (see also [18]). This information is very helpful for solving the 2-level planarization problem in which the permutation of one level is fixed.

Besides the application in automatic graph drawing, the 2-level planarization problem comes up in computational biology. In *DNA mapping*, small fragments of DNA have to be ordered according to the given overlapping data and some additional information. M.S. Waterman and J.R. Griggs [24] have suggested combining the information derived by a digest mapping experiment with the information on the overlap between the DNA fragments. If the overlapping data is correct, the

maps can be represented as a 2-level planar graph. But, in practice, the overlapping data may contain errors. Hence, they suggested solving the 2-level planarization problem (see also [23]). Furthermore, the 2-level planarization problem arises in global routing for row-based VLSI layout (see [16]).

See also

- ► Graph Planarization
- ► Integer Programming

References

1. Eades P, Kelly D (1986) Heuristics for reducing crossings in 2-layered networks. Ars Combin 21A:89–98
2. Eades P, Whitesides S (1994) Drawing graphs in two layers. Theoret Comput Sci 131:361–374
3. Eades P, Wormald NC (1994) Edge crossings in drawings of bipartite graphs. Algorithmica 10:379–403
4. Fukuda K (1996) Personal Communication
5. Garey MR, Johnson DS (1983) Crossing number is NP-complete. SIAM J Alg Discrete Meth 4:312–316
6. Grötschel M, Jünger M, Reinelt G (1985) Facets of the linear ordering polytope. Math Program 33:43–60
7. Harary F, Schwenk A (1972) A new crossing number for bipartite graphs. Utilitas Math 1:203–209
8. Healy P, Kuusik A (1999) The vertex-exchange graph: A new concept for multi-level crossing minimisation. In: Kratochvil J (ed) Graph Drawing (Proc. GD'99). Lecture Notes Computer Sci. Springer, Berlin, pp 205–216
9. Jacobson G, Vo K-P (1992) Heaviest increasing/common subsequence problems. In: Apostolico A, Crochemore M, Galil Z, Manber U (eds) Combinatorial Pattern Matching, Third Ann. Symp. (CMP, 1992). Lecture Notes Computer Sci. Springer, Berlin, pp 52–66
10. Jünger M, Lee E, Mutzel P, Odenthal T (1997) A polyhedral approach to the multi-layer crossing number problem. In: DiBattista G (ed) Graph Drawing (Proc. GD'97). Lecture Notes Computer Sci. Springer, Berlin, pp 13–24
11. Jünger M, Leipert S, Mutzel P (1998) Level planarity testing in linear time. In: Whitesides S (ed) Graph Drawing (Proc. GD'98). Lecture Notes Computer Sci. Springer, Berlin
12. Jünger M, Mutzel P (1996) 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. J Graph Algorithms Appl (JGAA) 1(1):1–25, http://www.cs.brown.edu/publications/jgaa/
13. Jünger M, Thienel S (2000) The ABACUS system for branch and cut ad price algorithms in integer programming and combinatorial optimization. Softw Pract Exper 30:1325–1352
14. Kececioglu JD (1991) Exact and approximation algorithms for DNA sequence reconstruction. PhD Thesis, University Arizona
15. Kececioglu JD (1993) The maximum weight trace problem in multiple sequence alignment. In: Proc. 4-th Symp. Combinatorial Pattern Matching. Lecture Notes Computer Sci, vol 684, pp 106–119
16. Lengauer T (1990) Combinatorial algorithms for integrated circuit layout. Wiley, New York
17. Mutzel P (1997) An alternative method to crossing minimization on hierarchical graphs. In: North S (ed) Graph Drawing (Proc. GD'96). Lecture Notes Computer Sci. Springer, Berlin, pp 318–333
18. Mutzel P, Weiskircher R (1998) Two-layer planarization in graph drawing. In: Chwa K-Y, Ibara O (eds) Algorithms and Computation; Ninth Internat. Symp. (ISAAC'98). Lecture Notes Computer Sci. Springer, Berlin, pp 69–78
19. Reinert K, Lenhof HP, Mutzel P, Mehlhorn K, Kececioglu J (1997) A branch-and-cut algorithm for multiple sequence alignment. In: Proc. First Annual Internat. Conf. Computational Molecular Biology (RECOMB'97), pp 241–250
20. Shahrokhi F, Sýkora O, Székely LA, Vrťo I (1997) On bipartite crossings, largest biplanar subgraphs, and the linear arrangement problem. In: Workshop Algorithms and Data Structures (WADS'97). of In: Lecture Notes Computer Sci, vol 1272. Springer, Berlin, pp 55–68
21. Spinrad J, Brandstädt A, Stewart L (1987) Bipartite permutation graphs. Discrete Appl Math 19:279–292
22. Sugiyama K, Tagawa S, Toda M (1981) Methods for visual understanding of hierarchical systems. IEEE Trans Syst, Man Cybern SMC-11(2):109–125
23. Vingron M, Lenhof H-P, Mutzel P (1997) Computational molecular biology. In: Maffioli F, Dell'Amico M, Martello S (eds) Annotated Bibliographies in Combinatorial Optimization. Wiley, New York
24. Waterman MS, Griggs JR (1986) Interval graphs and maps of DNA. Bull Math Biology 48(2):189–195

Optimization in Medical Imaging

STEPHEN C. BILLUPS
Department Math., University Colorado Denver, Denver, USA

MSC2000: 90C90

Article Outline

Keywords
Low-Level Feature Detection
Spatial Segmentation
  Deformable Models
  Optimization Techniques for Minimizing the Energy Function

Feature Segmentation
Clustering
Supervised Classification
Summary
See also
References

Keywords

Optimization; Medical imaging; Segmentation

By allowing physicians to peer into the human body, medical imaging technologies provide vital information for medical diagnosis, treatment and research. Beyond the mere visualization of anatomical structures, these imaging technologies are now being used in such roles as surgical planning, cancer diagnosis and prognosis, intra-operative navigation, radiotherapy treatment planning, and the tracking of disease progression. A key component in the effectiveness of medical imaging technologies is the development of sophisticated computer algorithms, which extract and analyze useful information. Such algorithms enable reliable and repeatable quantitative data to be extracted in order to support accurate medical diagnosis and treatment as well as clinical research. The development of such image analysis algorithms remains a rich area of research, involving numerous applications of optimization.

Medical images are generated by a variety of technologies. Among the most widely used are X-ray computed tomography (CT), emission computed tomography, magnetic resonance imaging (MRI), biomagnetic source imaging, ultrasound, and digital subtraction angiography. (See [1] for a tutorial of these medical imaging technologies). Images from these machines are typically stored as two-, three-, or four-dimensional arrays of data elements, corresponding to two or three spatial coordinates, and possibly a temporal coordinate. These data elements are generally referred to as *pixels*, but are also called *voxels* in three-dimensions and *hypervoxels* in four dimensions. The values of these elements can be scalars, such as tissue density, or vectors, such as relaxation time pairs in magnetic resonance imaging.

A central issue in extracting information from medical images is the problem of segmenting the image, either by identifying boundaries of critical structures in the image (*spatial segmentation*), or by classifying pixels according to some set of features, such as texture or gray levels (*feature segmentation*). In the first case, the image is segmented spatially into a number of regions which correspond to anatomical objects, such as organs. These regions typically form connected sets with well-behaved boundaries. In the second case, spatial information plays little or no role. For example, microcalcifications, which may be an early sign of breast cancer, may be scattered throughout otherwise healthy tissue. In detecting such microcalcifications, the pixels of an image would be classified either as healthy tissue or as microcalcifications. In this case, the pixel classes are not connected spatially, and can have very complex boundaries.

In either case, segmenting medical images requires the integration of low-level information about the image along with a priori high-level information about what the image represents. Extracting low-level information involves detecting features such as edges or textures. Once these features are detected, they can then be combined with high-level information either in the form of an expert system or some mathematical model of the knowledge. This integration task is made more difficult by the fact that medical images tend to suffer from sampling artifacts, spatial aliasing and noise, which can cause boundaries of structures to be indistinct and disconnected.

Low-Level Feature Detection

Numerous algorithms have been developed for identifying low-level features of images. The most common are edge-detection methods and texture transforms. Edge detection methods work by applying a digital filter, which emphasizes edges in the image. As an example, vertical edges can be detected by convolving the image with the *Sobel edge filter*

1	0	−1
2	0	−2
1	0	−1

After applying this filter, the resulting image will have values close to zero, except at vertical edges, which will have values which are relatively large in magnitude. By changing the elements of the filter, edges with different orientations can be detected.

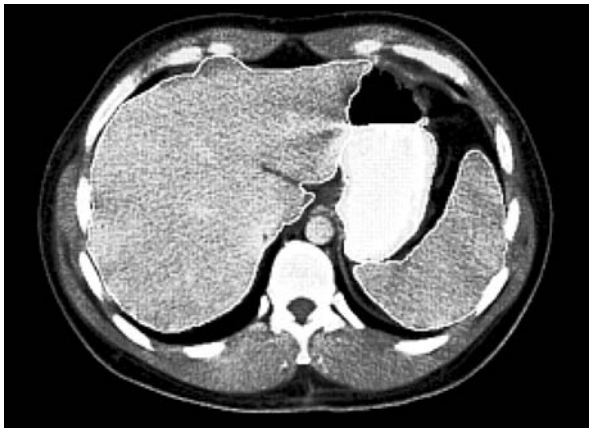
Another important low-level feature is texture. Generally speaking, this is a measure of image coarseness, smoothness and regularity. Numerous texture transforms have been proposed, including statistical, spectral and structural measurements. These transforms quantify various attributes of texture at every pixel in the image. Thus, after applying the texture transforms, each image pixel is represented by a vector of numbers which represent the texture.

Edge detection methods and texture transforms are described in detail in most recent image processing textbooks. (see, for example, [7]).

Spatial Segmentation

Once low-level features of the image have been extracted, they can then be combined with high-level information to segment the image. In the case of spatial segmentation, this generally involves determining boundaries between various regions of interest. Figure 1 provides an example of a CT scan of a human abdomen which has been partially segmented spatially. In this picture, boundaries of critical organs have been highlighted in white.

More than a thousand different algorithms have been proposed for automatically segmenting images. Among the techniques used in these algorithms are thresholding, region splitting, region merging, clustering, multiscale analysis, surface fitting, rule-based expert systems, relaxation, and deformable models.



Optimization in Medical Imaging, Figure 1
Partially segmented CT image of human abdomen. Organ boundaries are shown in white

A common theme underlying many of these techniques is the minimization of an *energy function* which in some way measures the quality of the segmentation.

Deformable Models

A spatial segmentation technique which has attracted much recent attention, particularly in the medical field, is the use of *deformable models*. Deformable models have the capability of combining low-level information derived from the image data with high-level knowledge about the characteristics of anatomical structures. An extensive survey of deformable models in medical image processing is given in [6]. Deformable models in the form of *snakes* were first introduced by M. Kass, A. Witkin, and D. Terzopoulos [5] to segment contour objects in 2D images. Deformable models are curves or surfaces, often defined using splines, which are controlled by an *energy function*. The energy function has two major components: the *external energy*, which measures how well the spline matches image features, such as edges, and the *internal energy* which measures nonaffine deformations from some model curve. By minimizing the sum of these two energy functions, a reasonable balance is achieved between matching image features and determining boundaries which are consistent with the a priori knowledge.

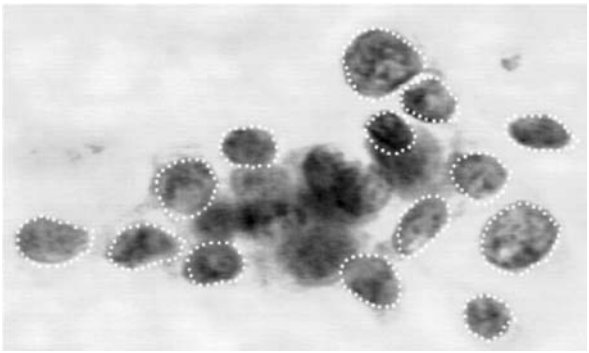
In two dimensions, the internal energy of a curve $c(s)$ can be determined by the formula

$$E = \int_s E_{\text{elasticity}} c(s) + E_{\text{bending}} c(s)\, \mathrm{d}s\,,$$

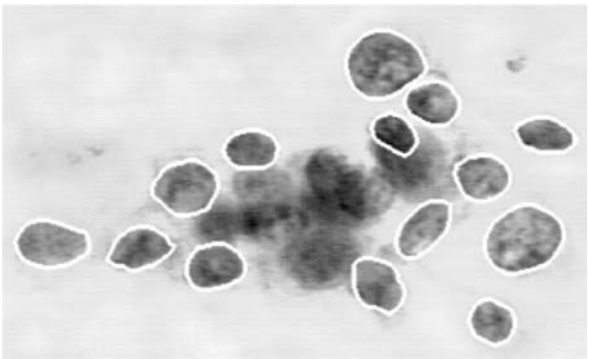
where s is the arclength of the curve, $E_{\text{elasticity}}$ represents the energy due to the elasticity of the spline and E_{bending} represents the energy due to bending. Minimizing this internal energy results in a curve which is as close as possible to the original shape of the model curve.

The external energy of a curve is determined by how well the curve matches image features. In particular, the external energy is minimized when the curve is aligned with edges or when textures on either side of the curve are different.

Figure 2 and Fig. 3 give an example of the use of snakes to segment an image representing a breast tissue sample [9]. In Figure 2 an initial estimate of the cell boundaries is entered by a technician using a mouse



Optimization in Medical Imaging, Figure 2
Initialization of snake contours for cell boundaries (Copied with permission from [9])



Optimization in Medical Imaging, Figure 3
Cell contours minimizing energy (Copied with permission from [9])

to define approximate contours of cell boundaries. After these snakes are initialized, a greedy algorithm is used to achieve a local minimum of the energy function. If the energy function value at a particular snake point can be lowered by moving the point to an adjacent pixel, then the point is moved. The process is repeated for each point until all points settle into a local minimum of the energy function. The result is shown in Fig. 3.

Early implementations of deformable models possessed only generic a priori knowledge (for example, smoothness criteria) of what region boundaries should look like. More sophisticated tools, called *deformable templates* [8] have been studied which incorporate more specific a priori information with respect to expected shapes and their spatial relationships.

Optimization Techniques for Minimizing the Energy Function

The energy function governing the deformable models or templates is, in general, a nonconvex function. Worse, in some cases, this energy function may actually be discontinuous. Consequently, local optimization techniques are usually inadequate unless the initialization of the deformable models is very accurate. This generally requires human intervention to provide an acceptable initialization.

Alternatively, global optimization techniques have been considered for minimizing the energy function. Among the more frequently used algorithms in this arena are simulated annealing (cf. ► Simulated annealing) and genetic algorithms (cf. ► Genetic algorithms).

Feature Segmentation

In applications where regions of interest are not expected to be spatially connected, it is more natural to segment according to features. To accomplish this, a number of features are evaluated at each pixel of the image. These features might include, for example, gray levels or texture transforms. The collection of measurements for a pixel forms a vector in n-dimensional *feature space*, where n is the number of features evaluated. It is then possible to classify pixels according to where they are located in feature space.

Clustering

One approach to classifying points in feature space is called *clustering*. Here, the m points representing the image are assigned to a predetermined number k of clusters. The problem can be stated more explicitly as that of determining k centers in $\mathbf{R}^n$ such that the sum of the distances of each point to the nearest center is minimized. This amounts to solving the following minimization problem:

$$\min_{C_1,\ldots,C_k} \sum_{i=1}^{m} \min_{j=1}^{k} \|x_i - C_j\| \ ,$$

where C_j, $j = 1, \ldots, k$, represent the centers and x_i, $i = 1, \ldots, m$, are the feature vectors for the m points. When the norm is the 2-norm, this problem is solved by the k-mean algorithm [4]. Whereas when the norm is the 1-

norm, the problem reduces to a bilinear program which is solved by the k-median algorithm [3].

Clustering algorithms are *unsupervised classification* schemes in that no a priori knowledge is used to determine the classification. However, the choice of the number of clusters k can play a significant role. If k is too large, it may be difficult to attach meaning to the clusters. Whereas, if k is too small, critical information may be missed; for example, microcalcifications may be clustered together with healthy tissue.

Supervised Classification

Another approach to classifying pixels is to partition feature space into a number of regions according to some a priori knowledge. This knowledge can be provided by a training set of data taken from a large collection of images. Each element of the training data includes the vector of features, along with the known classification of that vector. These data elements can be partitioned into a collection of subsets, with each subset corresponding to a unique classification.

With this training data, a discriminant function can be constructed that distinguishes between the different classifications. If the conical hulls of the subsets corresponding to each classification are disjoint, then the subsets can be discriminated by a piecewise-linear function, which can be calculated by solving a single linear program (cf. also ► Linear programming) [2]. Otherwise, a decision tree can be constructed by solving a finite sequence of linear programs.

Once the discriminant function has been determined, new images can be segmented by evaluating the features of each pixel, and then applying the discriminant function to classify the pixel.

Summary

These applications give a sense of the applicability of optimization techniques to medical image processing. Global optimization methods, such as simulated annealing and genetic algorithms, are used to minimize energy functions corresponding to deformable models; bilinear programs arise in clustering approaches to image segmentation; and linear programs arise in the training of image classification schemes.

See also

- ► Entropy Optimization: Shannon Measure of Entropy and Its Properties
- ► Maximum Entropy Principle: Image Reconstruction

References

1. Acharya R, Wasserman R, Stevens J, Hinojosa C (1995) Biomedical imaging modalities: A tutorial. Computerized Medical Imaging and Graphics 19:3–25
2. Bennett KP, Mangasarian OL (1993) Multicategory discrimination via linear programming. Optim Methods Softw 3:27–39
3. Bradley PS, Mangasarian OL, Street WN (1997) Clustering via concave minimization. Adv Neural Inform Process Systems 9:368–374
4. Jain AK, Dubes RC (1998) Algorithms for clustering data. Prentice-Hall, Englewood Cliffs
5. Kass M, Witkin A, Terzopoulos D (1987) Snakes-active contour models. Internat J Comput Vision 1:259–268
6. McInerney T, Terzopoulos D (1996) Deformable models in medical image analysis: A survey. Medical Image Anal 1:91–108
7. Pitas I (1993) Digital image processing algorithms. Prentice-Hall, Englewood Cliffs
8. Rueckert D, Burger P (1997) Geometrically deformable templates for shape-based segmentation and tracking in cardiac MR images. In: Pelillo M, Hancock ER (eds) Energy Minimization Methods in Computer Vision and Pattern Recognition. Lecture Notes Computer Sci. Springer, Berlin, pp 83–98
9. Street WN, Wolberg WH, Mangasarian OL (1993) Nuclear feature extraction for breast tumor diagnosis. In: Biomedical Image Processing and Biomedical Visualization. SPIE-The Internat. Soc. for Optical Engineering, Bellingham, pp 861–870

Optimization in Operation of Electric and Energy Power Systems

PETER G. HARHAMMER
Department Energy Economics,
Techn. University Vienna, Vienna, Austria

MSC2000: 90C35, 90C30, 90C10

Article Outline

Keywords
Problem Definition
Modeling
  Thermal Plant
  Storage Plant

Hydro-Reservoir
Run-of River Plant
Energy Purchase Contract
Power Balance
Objective Function
Optimization
Computer Implementation: Optimization in Practice
Model Formulation
Model Generator Formulation (EasyModeler/6000)
Numerical Example
Input Data
Numerical Results
Summary and Future Trends
Note
References

Keywords

Electric power system; Load dispatcher; Operation planning; Load curve; Thermal plant; Run-of-river plant; Storage plant; Energy purchase contract; Co-generation plant

From the 1960s onwards the computer-assisted solution of the problem to distribute the load of an *electric power system* among all generation units in operation (economic dispatch) attracted the attention of *load dispatchers* (persons, responsible for economic and secure operation of a power system) in utilities and engineering scientists [12]. Hand in hand with the development of computer hardware to high internal speed, large main storages and low costs, different optimization methods have been applied to more and more complex operation planning models of power systems to better reflect their reality [3,4,10,13]. Today, the operation planning problem is of high economic importance to the electric power industry in presence of deregulation measures, privatization, and an upcoming competitive marketing environment. This dramatic move to a market-based industry will significantly change power system operations. New pricing mechanisms (e. g., spot-market, auctions) will come in place and require a fast solution of new optimization problems to economically plan the operation of future power systems under new marketing conditions.

Therefore, this paper outlines the basic principles of how to model and optimize a power system to economically plan its operation.

Problem Definition

The objectives of the *operation planning* problems of electric power systems are dependent on their planning period. Short-term planning problems cover a day, a weekend or a week and are designed to solve the scheduling of generation units and contracts in the most economic way possible. Long-term planning problems cover a planning period of a season or a year and more and must allocate the resources available (e. g., fuels, storage water, contracted energy amounts) to smaller time periods. Medium-term problems for more than a week or a month are often to be solved in presence of small hydro-reservoirs or amounts of fuels to be disposed in short intervals. These three planning problems with different objectives are not independent from each other. They form an hierarchical model system (long-term, medium-term, short-term) to be run continuously to plan the operation economically from a day to a year.

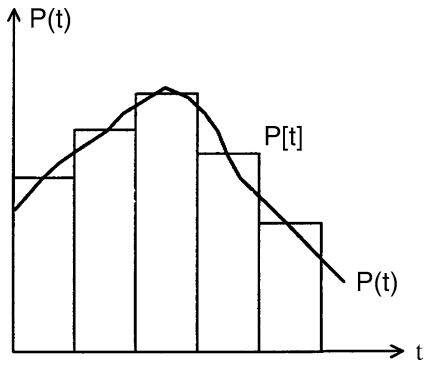
Electrical energy is not storable in large amounts. Therefore, the load dispatcher of a power system must balance its load $P_l(t)$ and the generation $P_g(t)$ including energy purchase contracts $P_c(t)$ for each point in time, expressed by equation (1).

$$P_g(t) + P_c(t) = P_l(t)\,; \tag{1}$$

$P_l(t)$ is defined by the *load curve* (total power system load over the time of day) and is given as input data representing constant load values for short time-intervals (e. g., 1 hour), called *time-steps* in modeling terms. This simplification replaces the continuous load curve by a stepwise representation. The transformation into a stepwise load curve Fig. 1, is necessary to be able to apply mathematical programming optimization methods.

The power system load $P_l[t]$ (time-discretized representation of load curve with $[t]$) is covered partially by an energy purchase contract and by generators located in different power plants.

There are two main types of power plants: *thermal plants* (including nuclear plants) and *hydro plants*, to be distinguished by their energy input (primary energy) to produce electricity (secondary energy). Thermal plants use coal, oil and gas whereas hydro plants need water of rivers or stored in reservoirs as input. Moreover, thermal generation units (a generator in a thermal



Optimization in Operation of Electric and Energy Power Systems, Figure 1
Load curve representation

plant) are characterized by other operational conditions than hydro ones. These different operational conditions must be taken into account to correctly model the different types of generation units.

Thermal generation units have, in a first approach, unlimited access to their fuels (coal, oil, gas) to be payed for. The operational costs of thermal units are defined by the cost curve for fuels between a minimal and a maximal electric power output. Moreover, their start-up or shut-down procedure lasts a number of hours and also causes procedure costs. Finally, their electric power output can be changed with a small time gradient only.

There are also two types of hydro plants to be distinguished. Firstly, the *run-of-river plants* that are located on a river. They generate electricity out of the (no-cost) water transported by the river. Therefore, their electric power output is constant over longer time periods (e. g., a day) and changes only when the amount of the water freight of the river changes. There is no degree of freedom for the optimization but a run-of-river plant must be taken into account to compute correct generation schedules, accepted by load dispatchers.

Secondly, *storage plants* which can only use a small amount of their reservoir-stored water during the planning period. This condition makes the model dynamic in time and requires special attention when modeling a storage plant. Fast changes of their electric power output are possible within their generation limits. This is necessary to follow large differences of the power system load of consecutive time-steps.

Sometimes, storage plants have pumping-units too which transport water from a lower reservoir to a higher one at times when the energy costs needed to pump the water are low (e. g. during the night). When the energy costs are high, the pumped water is released from the upper reservoir to generate electricity by running into the lower one. This pumping possibility makes the operation of a power system more flexible but also more complex. Therefore, pumping units are not considered here to not overburden the reader.

The last important power system element that contributes to cover the power system load is the *energy purchase contract* with an outside utility. Such a contract is defined by the costs for the delivered energy and is limited by a maximum power and often by a certain amount of energy allowed to be purchased during a given time period (e. g., day).

Summarizing the afore described power system elements, equation (1) can be enhanced as follows:

$$P_g[t] = P_{th}[t] + P_{rr}[t] + P_s[t] \tag{2}$$

$$P_{th}[t] + P_{rr}[t] + P_s[t] + P_c[t] = P_l[t]\,, \tag{3}$$

where

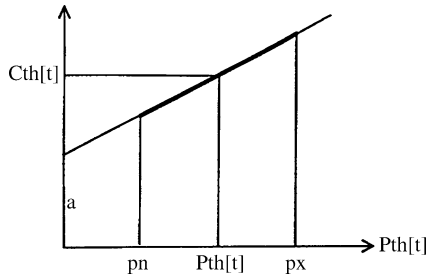
- $P_{th}[t]$: generation of thermal plants at time-step t;
- $P_{rr}[t]$: generation of run-of-river plants at time-step t;
- $P_s[t]$: generation of storage plants at time-step t.

Equation (3) describes the power balance of a so-called hydro-thermal system in its most general form (pumping not included).

The short-term scheduling problem is chosen as a sample of the operation planning problem for the complete planning cycle (day to year). The load dispatcher must solve this operation planning problem on a regular (daily) basis in order to achieve his objective to operate the power system and its elements in the most economic way possible.

Modeling

In the following, all power plants discussed before, the energy purchase contract, the power balance, and the objective to minimize operational costs are modeled in MIP-terms (mixed integer programming). Simple modeling approaches with linear relationships are chosen for the power system elements involved to support the understanding of the reader for the short-term



Optimization in Operation of Electric and Energy Power Systems, Figure 2
Cost curve of a thermal unit

scheduling problem of the operation planning process in power systems.

Before starting with the model descriptions of the power system elements involved in covering the power system load, it is necessary to summarize the time parameters used for modeling.

The planning period is the time horizon (e. g., day, weekend, week, month, year) covered by the optimization problem. Each time horizon is split up into so-called time-steps with equal length (e. g., 1 hour, 0.5 hour) during which the power system load is approached as a constant value (see Fig. 1). This simplification makes the application of mathematical programming methods possible since the right-hand side of the matrix is transformed into a constant value.

In general, there are always more than one unit in a power plant. Here, only one generation unit is assumed. Therefore, the term unit is used synonymously with plant in the following.

Thermal Plant

A linear relationship of the cost $C_{\text{th}}[t]$ to electric generation $P_{\text{th}}[t]$ is chosen for the sake of simplicity, not altering the modeling approach of the thermal unit itself.

- Fuel costs:

$$C_{\text{th}}[t] = a * Y[t] + b * P_{\text{th}}[t] \,. \tag{4}$$

- Limits on electric generation:

$$p_n * Y[t] \le P_{\text{th}}[t] \le p_{\text{x}} * Y[t] \,. \tag{5}$$

- Start-up indicator:

$$U[t] \ge Y[t] - Y[t-1] \,, \tag{6}$$

$$U[t] \in (0,1), \quad Y[t] \in (0,1) \,. \tag{7}$$

- Start-up costs:

$$C_{\text{s}}[t] = c_{\text{sth}} * U[t] \,. \tag{8}$$

- Data (thermal unit)
  - a: fixed costs;
  - b: incremental costs;
  - p_{n}: minimum electric power output;
  - p_{x}: maximum electric power output;
  - c_{sth}: start-up costs.
- Variables (thermal unit)
  - $C_{\text{th}}[t]$: fuel costs at time-step t;
  - $C_{\text{s}}[t]$: start-up costs at time-step t;
  - $P_{\text{th}}[t]$: electric generation at time-step t;
  - $Y[t]$: on/off indicator at time-step t;
  - $U[t]$: start-up indicator at time-step t.

Storage Plant

In a first modeling approach, the water discharge curve is assumed to be linear between zero and a maximum discharge value q_{x}. Moreover, a constant head $H = H_{\text{s}}(t)$ is also permissible:

$$P_{\text{s}}(t) = k * Q_{\text{s}}(t) * H_{\text{s}}(t) \,. \tag{9}$$

Therefore the equation (9) for the electric generation $P_{\text{s}}(t)$ of a storage plant with a water discharge $Q_{\text{s}}(t)$ and a water height $H_{\text{s}}(t)$ can be transformed into (12) by applying equation (11):

$$P_{\text{s}}(t) = k * Q_{\text{s}}(t) * H_{\text{s}}(t) \,, \tag{10}$$

$$H_{\text{s}}(t) = H = \text{const} \,, \tag{11}$$

$$P_{\text{s}}(t) = k_{H_{\text{s}}} * Q_{\text{s}}(t) \,; \tag{12}$$

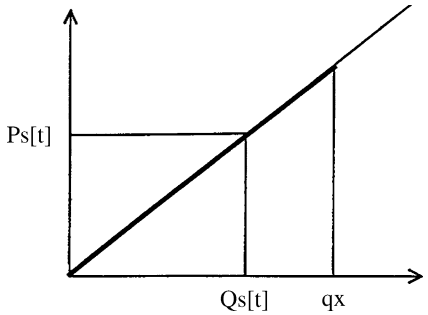
- Electric generation

$$P_{\text{s}}[t] = k_{H_{\text{s}}} * Q_{\text{s}}[t] \,. \tag{13}$$

- Limits on water discharge

$$0 \le Q_{\text{s}}[t] \le q_{\text{x}} \,. \tag{14}$$

- Data (hydro unit; storage plant)
  - $k_{H_{\text{s}}}$: gradient of the linear water discharge curve;
  - q_{x}: maximum possible water discharge.



Optimization in Operation of Electric and Energy Power Systems, Figure 3
Water discharge curve

- Variables (hydro unit; storage plant)
  - $P_s[t]$: electric generation at time-step t;
  - $Q_s[t]$: water discharge (cubicmeter per second) for electric generation at time step t.

Hydro-Reservoir

Each storage plant is connected to at least one hydro-reservoir. The content of the hydro-reservoir changes form time-step to time-step and can be computed from the so-called *storage equation* (15) expressing the water volume $V[t]$ in the reservoir at the end of each time-step t:

- Water volume

$$V[t] = V[t-1] - 3600 * Q_s[t]\,\mathrm{d}t + 3600 * Q_{\mathrm{in}}[t] * \mathrm{d}t\,. \tag{15}$$

- Volume limits

$$v_{\mathrm{n}} \le V[t] \le v_{\mathrm{x}}\,. \tag{16}$$

- Storage condition

$$A_T = V[0] - V[T]\,. \tag{17}$$

The equation (15) defines the amount of water allowed for generation of the storage plant during the planning period. In short-term planning (e. g., daily optimization) it is often required that the water volume at the end of the planning horizon $V[T]$ equals the starting volume $V[0]$.

- Data (hydro-reservoir)
  - $V[0]$: volume in the reservoir at the beginning of the planning period;
  - A_T: water volume available for generation to the storage plant during the planning period;
  - v_{n}: minimum volume allowed in the reservoir;
  - v_{x}: maximum volume allowed in the reservoir;
  - $Q_{\mathrm{in}}[t]$: natural inflow into the reservoir at time-step t;
  - dt: time-steplength in hours.
- Variables (hydro-reservoir)
  - $P_s[t]$: electric generation at time-step t;
  - $Q_s[t]$: water discharge (cubicmeter per second) for electric generation at time-step t.

Run-of River Plant

The electric generation of a run-of-river plant is directly dependent form the water freight of the river. In general, the water freight of a river remains constant during a short-term planning period of a day or even a weekend. Therefore, the electric generation of a run-of-river plant remains constant during the planning period and has no degree of freedom in mathematical terms concerning its (constant) generation. Although of no mathematical importance to the optimization, run-of-river plants must be considered in the model because of practice-related reasons. A load dispatcher needs the economic dispatch problem to be solved in engineering terms, taking into account all power system elements, including run-of-river plants, contributing to cover the power system load.

- Data (hydro unit; run-of-river plant)
  - $P_{\mathrm{rr}}[t]$: hydro-unit (run-of-river plant) – electric generation.

Energy Purchase Contract

Although there is a wide variety of different contract conditions, a very simple contract is modeled here to demonstrate the basic principles.

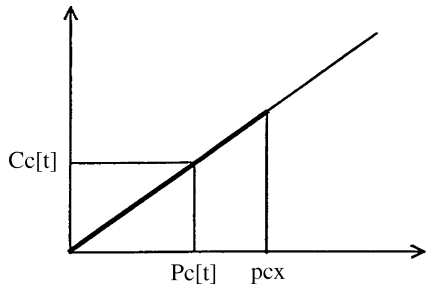
Constant incremental costs are assumed for the purchased energy between $P_c = 0$ and $P_c = p_{cx}$, the maximum power allowed to be purchased from the selling utility during the planning period.

- Contract costs

$$C_c[t] = T_c * P_c[t] * \mathrm{d}t\,. \tag{18}$$

- Contracted electric power limits

$$0 \le P_c[t] \le p_{cx}\,. \tag{19}$$



Optimization in Operation of Electric and Energy Power Systems, Figure 4
Contract cost curve

In addition to the above (basic) model of a contract, there is often a maximum amount of contracted energy allowed to be purchased from the selling utility during the planning period as follows:

$$\sum_{t=1}^{T} P_c[t] * \mathrm{d}t \leq a_{cx} \tag{20}$$

- Data (energy purchase contract)
  - T_c: tariff for the purchased energy;
  - dt: time-steplength (already defined);
  - p_{cx}: maximum power allowed to be purchased from the selling utility;
  - a_{cx}: maximum energy allowed to be purchased from the selling utility during the planning period.
- Variables (energy purchase contract)
  - $C_c[t]$: contract costs at time-step t;
  - $P_c[t]$: electric power of purchased energy at time-step t.

Power Balance

The power balance equation (3) is the most important part of an operation planning optimization model and is therefore repeated here again:

$$P_{th}[t] + P_{rr}[t] + P_s[t] + P_c[t] = P_l[t]\,. \tag{21}$$

Both the variables $P_{th}[t]$, $P_s[t]$ and $P_c[t]$ and the input data $P_{rr}[t]$ and $P_l[t]$ were already defined when the different power system elements were modeled.

Objective Function

One of the main goals of utilities is to operate their power system as economic as possible. Therefore, the operational costs during the planning period must be minimized to meet the overall target of a load dispatcher. This objective formulates as follows:

$$Z = \sum_{t=1}^{T} (C_{th}[t] + C_c[t])\ \mathrm{d}t + \sum_{t=1}^{T} C_s[t] \rightarrow \min\,. \tag{22}$$

The variables $C_{th}[t]$, $C_c[t]$ and $C_s[t]$ were already defined when describing the model of the thermal plant and the model of the contract.

Optimization

Since the model of the power system was formulated in MIP-terms (mixed integer programming) an optimization code with integrated MIP-techniques (linear programming followed by branch and bound method) must be applied. Each professional optimization code offers MIP methods to the user in different modularity; sometimes as package with a control language (e. g., MPSX/370 and MIP/370 [6]) or as a subroutine library (e. g., OSL [9]).

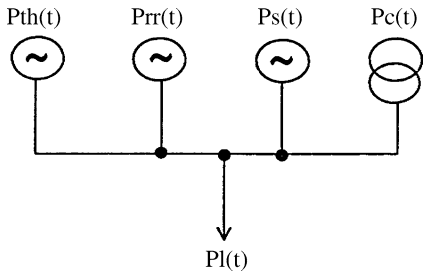
Moreover, the solution time and the accuracy of results are two main aspects that must be taken into account when planning the operation of an electric power system by applying optimization techniques. Therefore, a careful selection of the solution procedure is necessary to cope with both the accuracy of results required and a practice-related solution time that are primarily conflicting goals. But today, the solution of each operation planning problem of an electric power system as an MIP-optimization-model is possible in the solution time required and with the accuracy necessary, even for highly nonlinear and combinatorial problems (e. g., unit commitment: scheduling of a large number of thermal units taking into account time-dependent start-up costs) [2,14]. Moreover, multiprocessor hardware (e. g. RS/6000-SP) and related optimization codes (e. g., OSLp [8]) are further elements to speed up the solution time for time critical applications, sometimes arising in the operational planning environment of very large scale electric power systems. It must be emphasized in this context that three elements must be taken into account to successfully meet challenging solution-time targets. An appropriate modeling approach [14], an optimization procedure with acceleration techniques implemented and a problem dependent hardware with a professional optimization code.

Computer Implementation: Optimization in Practice

The computer implementation is an important step when developing an optimization model for the load dispatch practice. The problem definition, the modeling approach, and the optimization techniques are relatively easy to define. However, it is much more difficult to transform the optimization model, written on a paper, into a computer executable form. This bottleneck can be overcome by modeling tools, so-called model generators (e. g., *EasyModeler/6000* [7]), allowing to formulate the optimization model in algebraic terms and at the same time offering ease-of-use support for data input and debugging. Moreover, a close program interaction between the model generation tool and an optimization code is a must for a fast and successful model development and model usage in the utility practice of planning the operation of its power system. Therefore, tools with algebraic model formulations are widely accepted by engineers.

In the sequel, the power balance equation (21) is applied as a demonstration example concerning the ease-of-use of a model generator (EasyModeler/6000) appropriate for application by utility engineers and load dispatchers.

Model Formulation



Optimization in Operation of Electric and Energy Power Systems, Figure 5
Power balance

$$P_{\mathrm{th}}[t] + P_{\mathrm{rr}}[t] + P_{\mathrm{s}}[t] + P_{\mathrm{c}}[t] = P_{\mathrm{l}}[t]\,. \tag{23}$$

Model Generator Formulation (EasyModeler/6000)

- DATA
  - ...
  - $P_{\mathrm{l}}[t]$ (power system load);
  - $P_{\mathrm{rr}}[t]$ (generation; run-of-river plant → constant power output);
  - ...
- ENDDATA
- VARIABLE
  - ...
  - $P_{\mathrm{th}}[t]$ (generation; thermal plant);
  - $P_{\mathrm{s}}[t]$ (generation; storage plant);
  - $P_{\mathrm{c}}[t]$ (energy purchase contract; power);
  - ...
- ENDVARIABLE
- CONSTRAINTS
  - ...
  - Load (FORALL t)

$$P_{\mathrm{th}}[t] + P_{\mathrm{s}}[t] + P_{\mathrm{c}}[t] = P_{\mathrm{l}}[t] - P_{\mathrm{rr}}[t];$$

  - ...
- ENDCONSTRAINT

Comparing the power balance equation (21) with the formulation of the model generator ($P_{\mathrm{rr}}[t]$ as a constant value for each t appears on the right-hand side), no relevant differences occur in their algebraic representation. The same is valid for the modeling approaches of all other power system elements. Therefore, the application of a model generator to real-life problems can be learned very easily, even by self-education. Moreover, it supports a very fast development of optimization models occurring in the operational practice of utilities.

Numerical Example

In order to support the reader's understanding of the given problem, the numerical example is based on the (simplified) model of a power system, described before, to economically plan its operation. The result of the optimization run in Table 1 offers the schedules of the power system elements in operation and the associated costs as well as the value of the objective function.

Input Data

$t = 1, \ldots, 7$, $dt = 1$.

- Thermal plant
  - p_{n}: 100MW;
  - p_{x}:950MW;
  - a: 0MW;

Optimization in Operation of Electric and Energy Power Systems, Table 1
Schedule for all power system elements (7 time steps)

t	$P_o[t]$	$P_{th}[t]$	$P_{rr}[t]$	$P_{Io}[t]$	$P_o[t]$
1	200	110	90	—	—
2	300	210	90	—	—
3	600	410	90	—	100
4	800	615.4	90	—	94.6
5	1000	798.5	90	11.5	100
6	700	510	90	—	100
7	200	100	90	4.6	5.4

  - b: 26.3158MU/MW (MU denoting money unit);
  - c_{sth}: 1000MU.
- Run-of-river plant
  - $P_{\text{rr}}[t]$: 200MW.
- Storage plant
  - k_{H_s}: 2.304MW second/m^3;
  - q_{x}: 34.72m^3/second.
- Hydro-reservoir
  - $v[o]$: 41.23 $*$ 10$^3 m^3$;
  - v_{n}: 0m^3;
  - v_{x}: 150 $*$ 10$^3 m^3$;
  - A_T: 0m^3;
  - q_{in}: 1m^3/second.
- Energy purchase contract
  - T_{c}: 25MU/MWh;
  - p_{cx}: 100MW;
  - a_{cx}: 400MWh.

Numerical Results

- Objective function: 82 470.3MU.

Summary and Future Trends

Today, operational planning tools are state-of-the-art in utilities to support load dispatchers in meeting their objective to economically operate their power systems [15]. These planning tools are based on computer-assisted optimization models. From the 1960s onwards, these models have been closely linked to the development of computer hardware, modeling tools and optimization codes. With decreasing price/performance of computers, power system models became larger and larger, taking into account more details of the power system elements involved and so coming closer to their technical and operational reality. It must be emphasized in this context that MIP modeling and optimization methods have turned out to be the best and most effective strategy as an overall compromise of the conflicting targets: accuracy of results and solution time. Multiprocessor hardware and an associated optimization code are also available for time critical applications with combinatorial and highly nonlinear models.

In the future, utilities (sometimes privatized) will have to operate their power systems in a widely deregulated environment of a competitive market. This expected move to a market driven industry will significantly change electric utility operations [11]. New pricing mechanisms of energy trading (e. g. spot-marketing, auctions) will require the formulation of new optimization problems [1,5]. They will have to be solved much faster than up to now in order to support effectively and efficiently the decision making process in the business environment to come for utilities in the future. This new situation of utilities is a great but fascinating challenge for future research concerning the new operation planning targets of power systems.

Although this contribution referred to electric power systems only, the same modeling and optimization principles can be applied for other line-based energy systems (e. g., gas, district heating), even for coupled ones with *co-generation plants* (plants, that produce electricity and steam for industrial or public use in district heating systems).

Note

In memoriam Felix Schlaepfer, my friend, who contributed relevantly to the economic dispatch problem.

References

1. Delson JK, Shalidehpours M (1992) Linear programming applications to power system economics, planning and operations. IEEE Trans Power Systems 3:1155–1163
2. Floudas ACH (1995) Nonlinear and mixed-integer optimization. Oxford University Press, Oxford
3. Harhammer PG (1974) Economic dispatch by mixed-integer programmming. Diss, TU-Vienna (In German)
4. Harhammer PG (1976) Economic operation of electric power systems. In: Proc IX Internat Symp Math Program

5. Harhammer PG (1996) Advanced operation planning: Second generation tools (energy transaction management system). In: Power System Control and Management. IEE Conf. Publ., vol 421, pp 7–11
6. IBM (1988) MPSX/370 user guide: Version 2
7. IBM (1994) AIX EasyModeler/6000, user guide, release 2.0
8. IBM (1994) IBM parallel optimization subroutine library: User guide, release 1
9. IBM (1996) Optimization subroutine library: Guide and reference, release 3
10. IEEE (1976) Application of optimization methods in power system engineering. IEEE Tutorial Course Text (1976):44–68
11. IEEE Operating Planning Working Group (1992) Current issues in operational planning. IEEE Trans Power Systems 3:1197–1210
12. Kirchmayer LK (1958) Economic operation of power systems. Wiley, New York
13. Merlin A, Martin P (1969) Méthode de répartition journalière d'un ensemble de moyens de production thermique hydraulique. EDF Bull Direction d'Etude et Rech (Ser B Reseau électrique, matériel électrique) 4:13–38
14. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
15. Steinbauer E, Schadler A (1992) Modeling in hydro-thermal optimization. In: Proc. SVOR/ASRO-Tutorial on Optimization in Planning and Operation of Electric Power Systems, pp 125–139

Optimization Problems in Unit-Disk Graphs

B. Balasundaram, S. Butenko
Department of Industrial & Systems Engineering, Texas A&M University College Station, Texas, USA

MSC2000: 05C85, 05C69, 05C15, 05C62, 90C27, 90C59

Article Outline

Keywords and Phrases
Introduction
Definitions
Applications
Models
  Cliques
  Independent Sets
  Vertex Cover
  Domination
  Coloring and Clique Partitioning
  Related Results
Conclusions
References

Keywords and Phrases

Unit-disk graphs; Cliques; Independent sets; Dominating sets; Coloring; Approximation algorithms; PTAS

Introduction

Unit-Disk Graphs (UDGs) are intersection graphs of equal diameter (or unit diameter w.l.o.g.) circles in the Euclidean plane. In the *geometric (or disk) representation*, each circle is specified by the coordinates of its center. Three *equivalent* graph models can be defined with vertices representing the circles [18]. In the *intersection graph model*, two vertices are adjacent if the corresponding circles intersect (tangent circles are also said to intersect). In the *containment graph model*, two vertices are adjacent when one circle contains the center of the other. In the *proximity graph model*, an edge exists between two vertices if the Euclidean distance between the centers of corresponding circles is within a specified bound. Recognizing UDGs is NP-hard [10] and hence no polynomial time algorithm is known for deriving the geometric representation from the graph model. From an algorithmic perspective this places an textitasis on whether or not the geometric representation is needed as input. UDGs are not necessarily perfect or planar [18] as several other geometric intersection graph classes are and thus motivate the need for dedicated theoretical study.

The remainder of this article is organized as follows. We introduce the necessary definitions and notations for the various optimization problems on graphs considered in this article in Sect. "Definitions". A brief survey of applications modeled using UDGs and the role of optimization problems discussed in this chapter are presented in Sect. "Applications". A survey of algorithms and their key ideas for cliques, independent sets, vertex covers, domination, graph coloring and clique paritioning are presented in Sect. "Models". We conclude with a summary in Sect. "Conclusions".

Definitions

We consider simple, undirected graphs on n vertices and m edges denoted by $G = (V, E)$. For

a vertex $v \in V$, $N(v)$ is its neighborhood and $N[v] = N(v) \cup \{v\}$ is its closed neighborhood. We denote the complementary graph by $\bar{G} = (V, \bar{E})$ and the subgraph induced by $S \subseteq V$ by $G[S]$. We denote by $\delta(G)$ and $\Delta(G)$ the minimum and maximum vertex degrees in G respectively. Denote by $d(u, v)$ the shortest distance between $u, v \in V$, then the *k-neighborhood* of v is defined as $N_k(v) = \{u \in V : d(u, v) \leq k\}$. We also use the notation $G - v$ and $G - I$ to refer to the graph obtained from G by deleting vertex v (and incident edges) and by deleting a subset of vertices I (and incident edges), respectively. That is $G - v = G[V \setminus \{v\}]$ and $G - I = G[V \setminus I]$.

A *clique* is a subset of pairwise adjacent vertices in G. The *maximum clique problem* is to find a largest cardinality clique in G and the *clique number* $\omega(G)$ is the size of a maximum clique. An *independent set* (or *stable set*) is a subset of mutually non-adjacent vertices and the maximum independent set problem is to find an independent set of maximum cardinality. The *independence number* (or *stability number*) of a graph G is denoted by $\alpha(G)$ and it is the size of a maximum independent set. A *maximal* clique (independent set) is one that is not a proper subset of another clique (independent set). Cliques and independent sets are complementary to each other in the sense that $C \subseteq V$ is a clique in G if and only if C is an independent set in $\bar{G}$. For arbitrary graphs, the maximum clique and independent set problems are equivalent and possess similar complexity and approximation results. Algorithms and heuristics for one can be adapted via complement for the other. However, this equivalence is not naturally extended to geometric graphs that do not preserve upon complementing, their geometric property. For instance planar independent set is NP-complete [30] while clique is trivial. Results on UDGs will be discussed in Sect. "Cliques", "Independent Sets". A *vertex cover* S is a subset of vertices such that every edge in G has at least one end point in S. We denote by $\beta(G)$, the size of a minimum vertex cover. Clearly if $I \subseteq V$ is independent, then $V \setminus I$ is a vertex cover of G.

A *proper coloring* of a graph is one in which every vertex is colored (assigned a natural number) such that no two vertices of the same color are adjacent. A graph is said to be *k-colorable* if it admits a proper coloring with k colors. Vertices of the same color are referred to as a *color class* and they induce an independent set. The *chromatic number* of the graph, denoted by $\chi(G)$ is the minimum number of colors required to properly color G. Note that for any graph G, $\omega(G) \leq \chi(G)$, as different colors are required to color the vertices of a clique. The famous theorem by Brooks on graph coloring [11] states that $\chi(G) \leq \Delta(G)$ if G is neither a complete graph nor an odd cycle. A related problem is the *minimum clique partitioning* problem which is to partition the given graph G into a minimum number of cliques, $\bar{\chi}(G)$. Note that this is exactly the graph coloring problem on $\bar{G}$ and $\bar{\chi}(G) = \chi(\bar{G})$.

A *dominating set* D is a subset of vertices such that every vertex in the graph is either in this set or has a neighbor in this set. A *minimal* dominating set contains no proper subset which is also dominating. The minimum cardinality of a dominating set is called the *domination number*, denoted by $\gamma(G)$. Note that every *maximal* independent set is also a *minimal* dominating set. If a dominating set D is independent, it is called an *independent dominating set*. A a dominating set D is called a *connected dominating set* if $G[D]$ is connected. The independent and connected domination numbers (obviously defined) are denoted by $\gamma_i(G)$ and $\gamma_c(G)$. Naturally, G is assumed to be connected when we consider connected domination.

An approximation algorithm with approximation ratio $\rho > 1$ for an optimization problem Π, outputs for every instance x of Π with an optimal value $opt(x)$, a solution of value $sol(x)$ in time polynomial in size of x, such that $sol(x) \leq \rho \times opt(x)$ if Π is a minimization problem or $sol(x) \geq opt(x)/\rho$ if Π is a maximization problem.

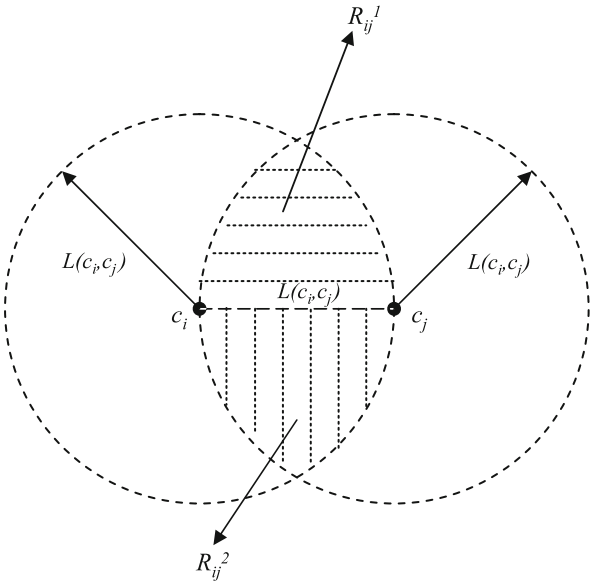
An optimization problem Π admits a *fully polynomial time approximation scheme* (FPTAS) if there is an approximation algorithm with approximation ratio $1 + \epsilon$ for any $\epsilon > 0$ that runs in time polynomial in size of the input *and* $1/\epsilon$. Π is said to admit a *polynomial time approximation scheme* (PTAS) if it has a polynomial time approximation algorithm with approximation ratio $1 + \epsilon$ for each *fixed* $\epsilon > 0$. A problem that is NP-hard in the strong sense [30], does not admit a FPTAS unless P = NP.

Applications

A major application area for UDG models is in wireless communication. Here the underlying *connectivity*

graph of the wireless nodes with equal and omnidirectional transmission-reception range can be modeled as a UDG [5,33]. Various optimization problems studied on UDGs are solved to facilitate effective operation of such networks.

For instance, a maximum independent set corresponds to a largest set of wireless nodes that can broadcast simultaneously without interference [56]. Alternately in location logistics, $\alpha(G)$ is also the maximum number of facilities that can be located in n potential locations if proximity between any two facilities is undesirable [45,61]. Clique and clique partitioning are popular approaches for clustering wireless networks [40]. Maximal cliques are also used to model and avoid link interference in ad-hoc networks [32]. A dominating set in UDGs modeling wireless network function as a small set of nodes that can send an emergency communication to the entire graph [18]. Domination and connected domination are also used to cluster wireless networks. The vertices in a dominating set D are designated as *cluster-heads* and $N[v]$ for each $v \in D$ forms a cluster. Inside a cluster formed in this fashion, 2-hop communication is possible between any pair of nodes via the cluster-head. In mobile wireless networks, the nodes are weighted appropriately to find a weighted dominating that can yield cluster-heads that are less mobile [8,16]. Alternately, if a virtual backbone is desirable among the cluster-heads, a connected dominating set is used in clustering [19,20]. Clustering is an important problem in wireless networks as it helps routing and improves efficiency and throughput [57]. Graph coloring problems are used to solve channel assignment problems in wireless networks such as frequency assignment, code or time slot assignment depending on the protocols used [33]. The idea is that the chromatic number of the connectivity graph is the smallest set of frequency bands (time-slots or codes) required to communicate without interference. The UDG recognition problem also has applications in determining molecular conformations [34].



Optimization Problems in Unit-Disk Graphs, Figure 1
The region R_{ij} is shaded

Models

Cliques

An $O(n^{4.5})$ time algorithm for finding a maximum clique in a UDG $G = (V, E)$ given the disk representation is presented in [18]. We briefly describe the ideas presented in [18]. Consider the set of disks $V = \{1, \ldots, n\}$ with centers at $c_i, \forall i \in V$. For a pair $i, j \in V$, $(i, j) \in E$ if and only if the Euclidean distance $L(c_i, c_j) \leq 1$. Denote by R_{ij} the region of intersection of two disks of *radius* $L(c_i, c_j)$ centered at c_i and c_j (see Fig. 1). Let $H_{ij} \subseteq V$ denote the disks with centers in the region R_{ij}. Consider a maximum clique C and let $i, j \in C$ be the farthest pair (in terms of Euclidean distance) of vertices in C, then $C \subseteq H_{ij}$. If such a farthest pair i, j in some maximum clique C is known, then we only need to find a maximum clique in $G[H_{ij}]$ to find a maximum clique in G. Since such an i, j pair is unknown, we can enumerate over all $(i, j) \in E$ to derive a polynomial time algorithm, if we can solve the maximum clique problem in polynomial time on every $G[H_{ij}]$. This is facilitated by the following observation made in [18]. Consider the region R_{ij} with $L(c_i, c_j) \leq 1$. The line joining c_i and c_j bisects R_{ij} into R_{ij}^1 and R_{ij}^2. The disk centers located in each half form a clique and hence the complement $\overline{G[H_{ij}]}$ is a bipartite graph. Since maximum independent set problem in bipartite graphs can be solved in $O(n^{2.5})$, we can find the maximum clique in $G[H_{ij}]$ in the same time which results in the claimed polynomial runtime. After the polynomial solvability was established in [18], the running time has been improved to $O(n^{3.5} \log n)$ in [9].

A relevant notion of *robust algorithms* for restricted graph classes such as UDGs was introduced recently in [55]. A robust algorithm for solving a problem on UDGs would accept only the graph G in standard format (adjacency list or matrix) as an input and solve the problem if it is indeed a UDG, or report that G is not a UDG. A polynomial time robust algorithm is presented in [55] for finding a maximum clique in UDGs (without the geometric representation) which returns a maximum clique or reports that G is not a UDG. The existence of a *polynomial* time *robust* algorithm for the maximum clique problem on UDGs is a surprising result given the NP-hardness of UDG recognition. A key idea is an ordering $L = e_1, e_2, \ldots, e_m$ of edges of G (input in standard format) referred to as a *cobipartite neighborhood edge elimination ordering* (CNEEO). Denote by $G_L(i)$ the subgraph of G with edge set $\{e_i, e_{i+1}, \ldots, e_m\}$. Define for each edge $e_i = (u, v)$ the set $N_L(i)$ to be the set of vertices adjacent to both u and v in $G_L(i)$. The authors define an edge ordering L to be CNEEO if for each e_i, $N_L(i)$ induces a cobipartite (complement of bipartite) graph in G. The authors then prove that given G and a CNEEO L, a maximum clique can be found in polynomial time and describe a greedy algorithm for determining a CNEEO L if it exists or certifying that G has none in polynomial time. Finally, the authors show that every UDG admits a CNEEO there by completing the robust polynomial time algorithm for maximum clique problem on UDGs (in fact for the larger class of graphs that admit a CNEEO).

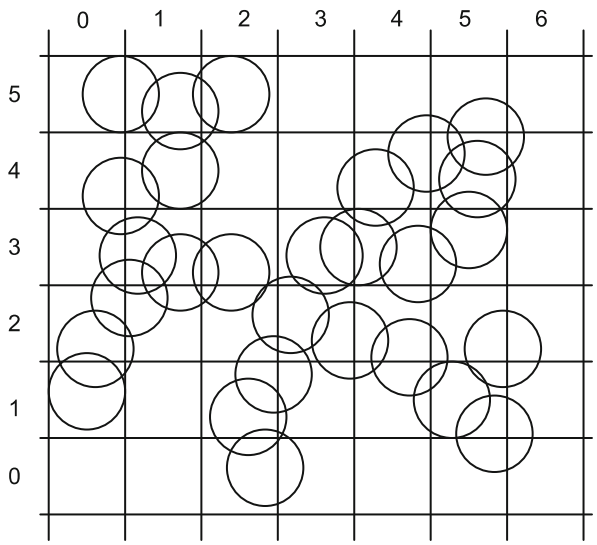
Independent Sets

Contrary to maximum clique, the maximum independent set (MIS) problem on UDGs is known to be NP-hard, even when the disk representation is given [18]. However, simple constant factor approximation algorithms and PTASs have been developed for this problem. Note that the strong NP-hardness of the MIS problem precludes the possibility of a FPTAS unless P = NP [30].

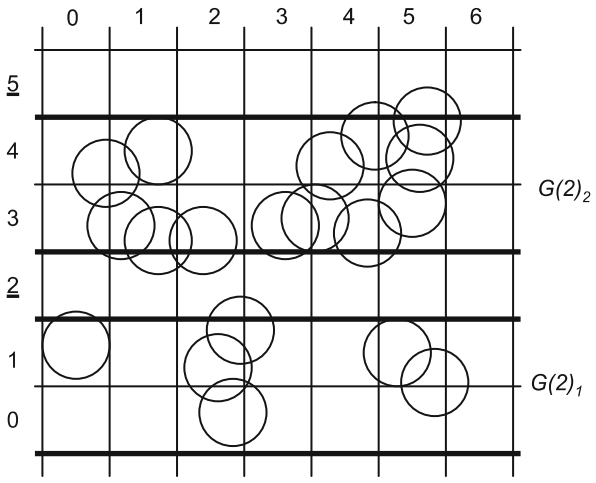
Given a graph G that does not contain a $(p+1)$-*claw* as an induced subgraph, an $O(n \log n + m)$ algorithm is presented in [37] to find an independent set of size at least $\alpha(G)/p$. A p-claw is a graph on $p+1$ vertices $V_p = \{u_0, u_1, \ldots, u_p\}$ such that u_0 is adjacent to all other vertices and $V_p \setminus \{u_0\}$ is an independent set. The algorithm proceeds by adding a vertex $v \in V$ to I followed by the removal of v and its neighbors *i. e.,* $N[v]$ from the graph. This step is repeated until V is empty, and the resulting independent set I is maximal. Let I^* denote a MIS in G. Suppose for the sake of argument that we sequentially removed vertices of $N[v]$ from I^* for each v removed from I. In any step, if v removed from I is also in I^*, the number of vertices in I^* deleted in that step is exactly one. If $v \in I \setminus I^*$, the number of vertices removed from I^* is at most p since a MIS in $N(v)$ has at most p vertices. Since I is maximal, I^* will be empty when I is empty and $\alpha(G) = |I^*| \leq |I \cap I^*| + p \times |I \setminus I^*| \leq p|I|$. By geometry, UDGs do not contain a 6-claw [45] and the above algorithm is a 5-approximation for the MIS problem on UDGs.

A simple 3-approximation algorithm is presented in [45] that, given a UDG G constructs an independent set of size at least $\alpha(G)/3$. This algorithm is based on the observation that every UDG has some vertex v such that $\alpha(G[N(v)]) \leq 3$. In particular, this is true for the vertex corresponding to the disk with minimum x-coordinate. Since every induced subgraph of a UDG is also a UDG, we can apply the same algorithm stated before from [37] and the observation will continue to hold in each step. But the vertex v added to I in each step is one with a MIS of size at most 3 in $N(v)$ yielding the desired approximation ratio. Given the disk representation, such a vertex v can be found easily in each step and without the disk representation such a vertex can certainly be found in polynomial time ($O(n^5)$).

The shifting strategy for geometric graphs introduced in [38], analogous to techniques for planar graphs introduced in [4] is the key ingredient in the PTASs developed for the MIS problem in [39,47]. The approaches are similar and we follow the presentation in [39]. Let $G = (V, E)$ be the UDG and the center of each disk in V is specified. If we seek an independent set IS$[G]$ such that $|\text{IS}[G]| \geq (1-\epsilon)\alpha(G)$, then choose parameter k to be the smallest integer for which $(k/(k+1))^2 \geq 1-\epsilon$. Grid the region containing G with unit squares by dividing into horizontal and vertical strips of unit width. Assume that the intervals on the axes corresponding to each strip are left closed and right open. This is necessary to deal with disks with centers on the boundary of two strips. For some $0 \leq i \leq k$, delete all the disks with centers in ev-



Optimization Problems in Unit-Disk Graphs, Figure 2
An example UDG *G* with a unit grid applied



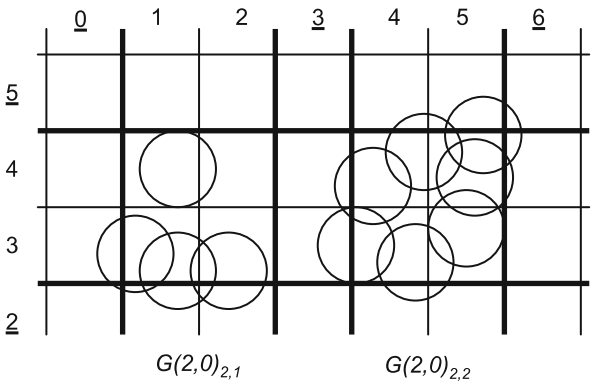
Optimization Problems in Unit-Disk Graphs, Figure 3
Graph *G*(*i*) with $i = k = 2$. Disks centered in the underlined horizontal strips (2,5) have been deleted leaving 2 horizontal super-strips corresponding to disjoint graphs $G(2)_1$ and $G(2)_2$

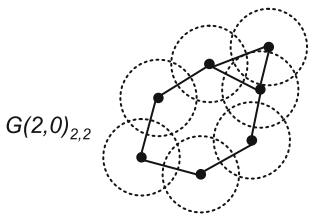
ery horizontal strip congruent to $i \mod (k+1)$ and denote the resulting UDG by $G(i)$. This leaves r disjoint horizontal "super"-strips of width k containing disjoint UDGs $G(i)_1, G(i)_2, \ldots, G(i)_r$ such that

$$G(i) = \bigcup_{1 \le j \le r} G(i)_j \ .$$

See Fig. 2 and Fig. 3. Varying i varies the deleted horizontal strips and hence permits us to *shift* the horizontal super-strips vertically over the graph G.

Now for some $1 \le j \le r$, select the *jth* horizontal super-strip. For some $0 \le l \le k$, delete from the super-strip $G(i)_j$, all disks with centers in every vertical strip congruent to $l \mod (k+1)$ and denote the resulting UDG by $G(i,l)_j$. Parameter l can similarly be seen as the horizontal shift parameter for the vertical super-strips. This partitions $G(i,l)_j$ into s_j UDGs $G(i,l)_{j,1}, \ldots, G(i,l)_{j,s_j}$ each contained in a square block of side k. See Fig. 4. Thus any MIS of $G(i,l)_{j,t}$ with $1 \le j \le r$ and $1 \le t \le s_j$ is of size at most $O(k^2)$ and can be found in time $n^{O(k^2)}$ by enumeration. The independent set returned is the union of independent sets from disjoint UDGs, but the one corresponding to the best block partition which depends on the shift parameters i, l. This can be expressed as follows. For a fixed vertical shift i and horizontal super-strip j, and





Optimization Problems in Unit-Disk Graphs, Figure 4
Graph $G(2, 0)_2$ with $l = 0$ (above). Disks centered in the underlined vertical strips (0, 3, 6) have been deleted leaving two 2×2 square blocks. Graph $G(2, 0)_{2,2}$ inside a 2×2 square block (below)

for some choice of l,

$$\mathrm{IS}[G(i,l)_j] := \bigcup_{1 \le t \le s_j} \mathrm{MIS}[G(i,l)_{j,t}] \ .$$

The best independent set in super-strip j is then obtained by horizontally shifting the grid, *i. e.,* varying l, thus

$$\mathrm{IS}[G(i)_j] := \max_{0 \le l \le k} \mathrm{IS}[G(i,l)_j] \ .$$

For each i, having found the best independent set in each super-strip,

$$\mathrm{IS}[G(i)] := \bigcup_{1 \le j \le r} \mathrm{IS}[G(i)_j] \ .$$

By varying the horizontal shift parameter i, we can select the best independent set for the graph as

$$\mathrm{IS}[G] := \max_{0 \le i \le k} \mathrm{IS}[G(i)] \ .$$

It is shown in [39] that

$$|\mathrm{IS}[G]| \ge (k/(k+1))^2 \alpha(G)$$

and the algorithm has a running time of $n^{O(k^2)}$. By the choice of k, this implies a PTAS for the MIS problem on UDGs. Suggestions for improving running time and solution quality are presented in [39,47]. After a vertical shift i, this involves solving the MIS problem optimally on each horizontal super-strip j using dynamic programming (DP) instead of approximating by horizontal shifting. This results in $|\mathrm{IS}[G]| \ge (k/(k+1))\alpha(G)$ and a total running time of $n^{O(k)}$.

In either version, the shifting strategy helps us to divide-and-conquer by breaking down the graph into pieces on which optimal resolution is possible (by enumeration or DP) in polynomial time, but at the same time bound the error created by the division process as division is flexible. We refer to [39] for details on the performance guarantees mentioned.

Clearly, the disk representation is required for the above PTAS to work. The open problem of whether a *robust* PTAS exists for the problem in the sense described in Sect. "Cliques" was settled positively in [51]. Given a graph $G = (V, E)$ (in standard format) and a desired error $\epsilon > 0$ we seek an independent set of size at least $\alpha(G)/\rho$ where $\rho = 1 + \epsilon$. The algorithm starts with some arbitrary vertex v and finds a MIS I_k in $G[N_k(v)]$ for $k = 0, 1, 2, \ldots$ sequentially until the condition $|I_{k+1}| > \rho |I_k|$ is violated. Let r denote the smallest $k \ge 0$ for which $|I_{r+1}| \le \rho |I_r|$. The authors show that there exists a constant (dependent on ρ) $c(\rho)$ such that $r \le c(\rho)$ and each MIS I_k can be found in polynomial time. By the choice of r, we know that $\alpha(G[N_{r+1}(v)]) \le \rho |I_r|$, *i. e.,* I_r is a ρ-approximate MIS for $G[N_{r+1}(v)]$. Suppose we have a ρ-approximate MIS I' for $G' = G[V \setminus N_{r+1}(v)]$, then clearly $I = I' \cup I_r$ is independent since $I' \subset V \setminus N_{r+1}(v)$ and $I_r \subset N_r(v)$. Furthermore, $\alpha(G) \le \alpha(G[N_{r+1}(v)]) + \alpha(G') \le \rho |I|$, *i. e.,* I is a ρ-approximate MIS of G. This fact combined with the fact that every vertex induced subgraph of a UDG is also a UDG, we have an inductive argument leading to the required PTAS. Robustness of the above algorithm is due to the following observations. The performance guarantee does not require G to be a UDG, and the algorithm always returns a $(1+\epsilon)$-approximate solution. However, geometry of UDGs is required to establish polynomially bounded running times for finding MIS in k-neighborhoods and, the existence of a constant $c(\rho)$. The proof of these claims also shows that if there exists an independent set $I_r > (2r+1)^2$ (bound assumes unit-radius disk representation, which is equivalent to other representations discussed before) then G is not a UDG. Since this certificate can be obtained in polynomial time, this PTAS is robust.

Vertex Cover

The minimum vertex cover problem on UDGs is also NP-hard as shown in [18]. Given a UDG $G = (V, E)$, a polynomial time heuristic that does not require a disk representation to find a vertex cover of size at most $1.5\beta(G)$ is presented in [45]. This algorithm requires results from [37,50]. The first result is the well-known Nemhauser-Trotter (NT) decomposition [50] which states given an arbitrary graph $G = (V, E)$ there exist disjoint vertex subsets P and Q such that (1) there exists a minimum vertex cover containing P; (2) if D is a vertex cover for $G[Q]$ then $D \cup P$ is a vertex cover for G; (3) any minimum vertex cover of $G[Q]$ contains at least $|Q|/2$ vertices. The second result from [37] states that following a NT decomposition, if $G[Q]$ can be colored using k colors, then $P \cup (Q \setminus S)$ is a vertex cover of size at most $2(1 - 1/k)\beta(G)$, where S is the largest color class in $G[Q]$.

The authors of [45] show that triangle-free UDGs can be colored using 4 colors. Given a UDG G, the heuristic first deletes vertices that form a triangle in

G (call it V'). With $G := G - V'$, NT decomposition is then applied to the resulting triangle-free UDG G to identify sets P and Q. $G[Q]$ is then colored using 4 colors and the set $S \subseteq Q$ corresponding to the largest color class is identified. The heuristic then returns $V' \cup P \cup Q \setminus S$ as the approximate vertex cover. The approximation ratio follows by applying the *local-ratio principle* [6,7,45] as follows. In V' we pick 3 vertices for each triangle, and we have to pick at least 2. And for the triangle-free UDG, which is 4-colorable, the result from [37] applies. The running time of the heuristic is dominated by the time to obtain NT decomposition which can be accomplished in polynomial time.

A PTAS has been developed in [39] for minimum vertex cover that uses an approach similar to the PTAS for the MIS problem described in Sect. "Independent Sets" from the same article. For $0 \le i < k$, instead of deleting a horizontal strip congruent to $i \mod (k+1)$, this approach uses super-strips of width $k+1$ overlapping at horizontal strips congruent to $i \mod k$. Then solving the MIS problem exactly using DP on each super-strip $G(i)_j$ also yields a minimum vertex cover. For a fixed i, the union over $0 \le j \le r$ of minimum vertex covers of each $G(i)_j$ is a valid vertex cover for G and the smallest vertex cover found over all i has size at most $((k+1)/k)\beta(G)$. Details are available in [39].

Domination

Minimum dominating set (MDS) problem, minimum independent dominating set (MIDS) problem and minimum connected dominating set (MCDS) problem are known to be NP-hard for UDGs [18]. In fact, they are NP-hard even when restricted to a subclass of UDGs called *grid graphs* on which MIS is polynomial time solvable [18]. The observation that a maximal independent set is also a minimal dominating set is used frequently in approximating dominating sets. It has been proven in [45] that any maximal independent set in a UDG G is no larger than five times its domination number, *i. e.,* $\alpha(G) \le 5\gamma(G) \le 5\gamma_i(G)$. This follows from the observation that if D is a maximal independent set in a UDG G, then any vertex in a MDS (or a MIDS) can dominate at most 5 vertices in D. Any maximal independent set is hence a 5-approximate solution for the minimum dominating set (MDS) problem and the minimum independent dominating set (MIDS) problem. A 10-approximate algorithm for MCDS problem is also presented in [45]. This bound has been improved to 8 in several papers [2,12,14,60] which present distributed implementations that are applicable in a practical setting in wireless networks. These heuristics construct a maximal independent set (which is dominating) and then connect it using a tree approach to obtain a CDS. This approach is based on the result from [2] that for a UDG G,

$$\alpha(G) \le 4\gamma_c(G) + 1 \,. \tag{1}$$

The maximal independent set I that is constructed (in polynomial time [60]) also has the property that for any $I' \subset I$, I' and $I \setminus I'$ are exactly distance two away from each other *i. e.,* there exist $u_1 \in I'$ and $u_2 \in I \setminus I'$ with $d(u_1, u_2) = 2$. The maximal independent set is connected using a spanning tree approach in [2,12,60] and using a *Steiner tree* in [14]. The bound (1) was improved recently in [62] to

$$\alpha(G) \le 3.8\gamma_c(G) + 1.2 \,. \tag{2}$$

This tighter bound shows that the 8-approximate algorithms such as the ones from [14,60] are in fact 7.8-approximate. It is also observed in [62] that if

$$\alpha(G) \le a\gamma_c(G) + b \,,$$

then $2.5 \le a \le 3.8$, which suggested that further improvement of their result was possible. Recently in [28], it has been shown that

$$\alpha(G) \le 3.453\gamma_c(G) + 8.291 \tag{3}$$

for UDGs and a distributed algorithm is presented that finds a CDS of size at most $6.91\gamma_c(G) + 16.58$.

Using the bound (2), a 6.8-approximate algorithm for the MCDS problem has also been proposed recently in [48] that connects a maximal independent set using a Steiner tree approach. Given a set of vertices designated as *terminals*, a tree connecting the terminals such that every leaf is a terminal is called a *Steiner tree*. The non-terminal nodes are called *Steiner nodes*. In principle, we could find a Steiner tree with minimum number of Steiner nodes (ST-MSN) with a maximal independent set I as terminals and the Steiner nodes S_I^* union

I yields a CDS for the UDG G. Instead of solving ST-MSN optimally, the authors approximate this problem. This is sufficient if we make the following observation.

From a MCDS D, we can obtain a solution for ST-MSN problem with terminals I as follows. We can find a spanning tree in $G[D]$; add an edge between each vertex in $I \setminus D$ and some vertex in D; and remove any leaf which is not a terminal in the resulting tree. The Steiner nodes in this solution are contained in D. Hence in the optimal solution to the ST-MSN problem, the number of Steiner nodes is at most $\gamma_c(G)$.

In the approach taken in [48], first a maximal independent set I is found such that every subset of I and its complement are exactly distance two apart. The authors develop and use a 3-approximate algorithm for the ST-MSN problem on UDGs given terminals I (see [48] for details). Denote the Steiner nodes in the 3-approximate solution for the ST-MSN problem by S_I, then its size is at most $3\gamma_c(G)$. Thus, the CDS $S_I \cup I$ has size at most $6.8\gamma_c(G) + 1.2$. This appears to be the approach with best performance guarantee available presently.

The weighted version of the MDS and MCDS problems, where the vertices of the UDG G are weighted and the objective is to find a dominating or a connected dominating set of minimum weight (sum of the weights of the selected vertices) have only been studied recently and the first constant factor approximation algorithms have been developed in [3]. A factor 72 approximation for MWDS and a factor 89 approximation for MWCDS are available and these problems appear to be more complicated than their unweighted counterparts.

A PTAS for the MDS problem given the disk representation was developed in [39], along similar lines as the schemes proposed for maximum independent set and minimum vertex cover problems. MCDS problem also has a PTAS developed in [17] for UDGs when the UDG is presented in its disk representation. In this work, an approximation algorithm running in time $n^{O((s \log s)^2)}$ is presented that constructs a CDS of size no larger than $(1 + 1/s)\gamma_c(G)$. The algorithm uses a grid based divide-and-conquer approach in combination with the shifting strategy. A robust PTAS for the MDS problem on UDGs was proposed recently in [52].

We briefly describe the robust PTAS for MDS on UDGs from [52]. Given a graph $G = (V, E)$, the authors define a *2-separated* collection of subsets S, as $S = \{S_1, \ldots, S_k\}$ with $S_i \subset V, i = 1, \ldots, k$ satisfying

$$\forall i \neq j, d(s, t) > 2, \forall s \in S_i, \forall t \in S_j \, .$$

If $D(S)$ denotes a MDS of $G[S]$, the authors show that for a 2-separated collection S in G,

$$\gamma(G) = |D(V)| \geq \sum_{i=1}^{k} |D(S_i)| \, .$$

Furthermore, if we have subsets T_i such that $S_i \subset T_i, i = 1, \ldots, k$ and a bound $\rho \geq 1$ such that

$$|D(T_i)| \leq \rho |D(S_i)|, \forall i = 1, \ldots, k, \tag{4}$$

and

$$D' = \bigcup_{i=1,\ldots,k} D(T_i) \text{ dominates } G, \tag{5}$$

then D' is a ρ-approximate MDS of G. This is true since,

$$|D'| \leq \sum_{i=1}^{k} |D(T_i)| \leq \rho \sum_{i=1}^{k} |D(S_i)|$$
$$\leq \rho |D(V)| = \rho \gamma(G) \, .$$

Given a UDG $G = (V, E)$ and an $\epsilon > 0$, the algorithm in [52] constructs in polynomial time (for fixed ε), a 2-separation S_i and the supersets T_i with $\rho = 1 + \epsilon$ satisfying the required properties (4), (5). This is accomplished as follows. First, we start with an arbitrary vertex $v_1 \in V_1 := V$ and compute a MDS $D(N_k(v_1))$ of $N_k(v_1)$ for $k = 0, 1, 2, \ldots$ until the condition

$$|D(N_{k+2}(v_1))| > \rho |D(N_k(v_1))|$$

is violated. Denote by r_1 the smallest k for which the above condition is violated, *i. e.*, $|D(N_{r_1+2}(v_1))| \leq \rho |D(N_{r_1}(v_1))|$. Then we iterate this procedure for the graph induced by $V_{i+1} := V_i \setminus N_{r_i+2}(v_i)$ until $V_{i+1} = \emptyset$. Note that in the subsequent iterations, the k-neighborhood is defined with respect to the current graph $G[V_{i+1}]$. Suppose this procedure terminates after K iterations, let $T_i = N_{r_i+2}(v_i)$ and let $S_i = N_{r_i}(v_i)$ for $i = 1, \ldots, K$. The authors then show that $S_1, \ldots, S_K$ is a 2-separated collection and $\bigcup_{i=1}^{K} D(T_i)$ dominates G. The termination condition for each iteration, $|D(T_i))| \leq \rho |D(S_i)|$, guarantees the required approximation ratio.

It is noted in [52] that G needs not be a UDG to derive the approximation ratio, however it is necessary to show polynomial time solvability. The running time guarantee is provided by the following results from [52]. Firstly, the number of iterations $K \leq n$ and in each iteration i, the MDS in each k-neighborhood $N_k(v_i)$ can be found in polynomial time since its size is shown to be bounded by a polynomial function of k, and finally the number of k-neighborhoods considered is also bounded since $r_i \leq c(\rho)$ where $c(\rho)$ is a constant that depends only on the desired approximation factor ρ. Finally, this PTAS can be made robust by utilizing the same certification approach to show the graph is not a UDG used for the MIS problem developed by the same authors, described in Sect. "Independent Sets".

Coloring and Clique Partitioning

The graph coloring problem on UDGs is known to be NP-hard. In [18], 3-colorability of UDGs is shown to be NP-complete and hence it follows that no approximation algorithm can achieve a ratio within $4/3$, unless P = NP. In fact k-colorability of UDGs is NP-complete for any fixed $k \geq 3$ [31]. A simple 3-approximation algorithm for the problem was presented in [45] based on results from [37,58]. Let

$$p(G) = \max_{H \subseteq G} \delta(H) \ ,$$

the largest p such that G contains a subgraph H of minimum degree p. Then

$$\chi(G) \leq p(G) + 1 [58]$$

and $p(G)$ can be found in $O(m + n)$ steps [37,58] as follows. Let $p := 0$ and let v be a vertex of minimum degree in G. Repeating the steps, $p := \max\{p, \delta(G)\}$ followed by $G := G - v$ until no vertices remain in G, finds $p(G)$. If we denote by v_i, the vertex removed in step i, each v_i then has at most $p(G)$ neighbors in $v_{i+1}, \ldots, v_n$. Processing the vertices in the order $v_n, \ldots, v_1$, and coloring each vertex with the smallest color not yet assigned to any of its neighbors already colored, guarantees a coloring of G with at most $p(G) + 1$ colors. If G is a UDG, then it is proven in [45] that

$$\frac{p(G)}{3} + 1 \leq \chi(G) \ .$$

Using similar approaches, it has also been shown in [54] that a UDG G can be colored using no more than $3\omega(G) - 2$ colors. A 3-approximate algorithm for coloring UDGs using network flow and matching techniques is also available from [31].

Clique partitioning is NP-complete even when restricted to *coin graphs* (UDGs where all overlaps are tangential) [15]. A polynomial time 3-approximate algorithm for this problem that uses the disk representation is available from [15]. The algorithm proceeds by first partitioning the plane into horizontal strips of width $\sqrt{3}$. A disk belongs to strip i if its center lies on the strip. A disk with its center on the boundary is assigned to the strip on top. Let G_i denote the UDG induced by disks in strip i and $V(G_i)$ are vertex disjoint. Solve the minimum clique partitioning problem exactly on each G_i and let Z_i denote the collection of cliques. The authors observe that this can be accomplished in polynomial time by coloring the complement since each G_i is a *cocomparability graph* [9,15]. The clique partition returned by the algorithm is $Z := \bigcup_i Z_i$ and it can be shown that $|Z| \leq 3\bar{\chi}(G)$ as follows. Let Z^* denote a minimum clique partition of UDG G and let Z_i^* be the restriction of Z^* to G_i obtained by excluding the vertices not in $V(G_i)$ from the cliques in Z^*. Z_i^* is a valid clique partition of G_i and hence $|Z_i^*| \geq |Z_i|$. If C is a clique in Z^*, the authors observe that based on geometric arguments, the centers of disks in C must lie inside three consecutive strips. Hence, each C in Z^* is a union of at most 3 disjoint cliques from Z_{j-1}^*, Z_j^*, and Z_{j+1}^* for some j. Hence we have,

$$|Z| = \sum_i |Z_i| \leq \sum_i |Z_i^*| \leq 3|Z^*| \ .$$

The running time of the approximation algorithm is dominated by the exact solution step on each strip resulting in $O(n + \bar{m})$ where $\bar{m}$ denotes the number of edges in $\bar{G}$.

Related Results

A survey of on-line and off-line approximation algorithms for independent set and coloring problems on UDGs and general *disk graphs* (intersection graphs of disks of arbitrary radii) can be found in [23]. A short survey of results for cliques, independent sets and coloring of disk graphs is also available in [27]. A survey

of complexity results on recognizing several variants of UDGs can be found in [36]. PTAS for *maximum weighted independent set* and *minimum weighted vertex cover* problems on intersection models of disks are available in [24].

A notion of *thickness* of UDGs is introduced and *fixed parameter tractability* of maximum independent set, minimum vertex cover and minimum (connected) dominating set problems (with thickness as parameter) is established in [59]. A parameterized algorithm running in $n^{O(\sqrt{k})}$ for finding an independent set of size k on *bounded ratio disk graphs* (the ratio of maximum diameter to minimum diameter is bounded by a constant) are presented in [1].

Several variants of the classical vertex coloring problem have been considered on UDGs, primarily motivated by different frequency assignment problems that arise in wireless networks. Apart from natural generalizations of UDGs such as general disk graphs, and bounded ratio disk graphs mentioned before, other generalizations of UDGs such as *Quasi* UDGs [42], *bisectored* UDGs [53] and *double disk graphs* [44] have also been developed motivated by wireless applications. Coloring problems have been studied in the context of these generalizations.

Algorithms for *distance constrained labeling*, which is a generalization of the well-known vertex coloring problem, for disk graphs are presented in [26]. Another variant of coloring called the *multicoloring* problem on UDGs is considered in [49]. The notion of *conflict-free coloring* is introduced and studied in the context of disk graphs in [25]. A k-improper coloring of a graph is one in which each color class induces a subgraph of maximum degree k. Note that 0-improper coloring is a proper coloring by the standard definition. For fixed k, the k-improper coloring problem has been shown to be NP-complete in [35]. Coloring and other problems on *bisectored unit disk graphs*, which generalize UDGs to allow for the phenomenon of *cell sectorization* in wireless communication are studied in [53]. Another generalization of UDGs motivated by frequency assignment problems in wireless networks are *double disk graphs*. Here, two concentric disks of arbitrary radii are associated with each vertex, and two vertices are adjacent if the inner disk of one intersects the outer disk of the other. For instance, one could think of the inner disk as the receiver range and the outer disk as the transmission range. Coloring problems on these graphs are studied and constant factor approximation algorithms are developed in [22,44]. Hierarchical models of UDGs formed by a sequence of labeled UDGs is considered in [46]. PTAS for the maximum independent set, minimum dominating set, minimum clique cover, and minimum vertex coloring problems for UDGs specified hierarchically are developed in [46].

The notion of *well-separated pair decomposition* [13] with applications in geometric proximity problems is studied in the context of UDGs and algorithms for the same are developed in [29]. In [41], the hardness of approximately embedding UDGs is considered. Given a UDG $G = (V, E)$, let $L(c_u, c_v)$ denote the Euclidean distance between centers c_u, c_v of discs $u, v \in V$ in an embedding $emb(G)$. The authors define the quality of an embedding $emb(G)$ as

$$q(emb(G)) = \frac{\max\limits_{(u,v)\in E} L(c_u, c_v)}{\min\limits_{(u,v)\notin E} L(c_u, c_v)}.$$

Note that for any proper unit disk embedding $emb(G)$, the numerator of $q(emb(G))$ is at most 1 and the denominator is more than 1. The authors of [41] then show that finding an embedding $emb(G)$ for a UDG G such that $q(emb(G)) \leq \sqrt{3/2} - \epsilon$ where $\epsilon \to 0$ as $n \to \infty$ is NP-hard.

A data structure referred to as *extended doubly connected edge list* is developed in [43] for representing UDGs which facilitate faster implementation of routing algorithms in mobile wireless networks.

Max-cut and *max-bisection* problems in UDGs are shown to be NP-hard in [21].

Conclusions

In this chapter, we have surveyed results from literature on classical combinatorial optimization problems such as the maximum clique, maximum independent set, minimum vertex cover, minimum (connected) domination, graph coloring and minimum clique partitioning on unit-disk graphs. Brief descriptions of the approaches taken to solve these problems and the key ideas involved have been explained. Several recent results from literature have also been presented. A summary of important results surveyed can be found in Table 1.

Optimization Problems in Unit-Disk Graphs, Table 1
A summary of results surveyed in this chapter

Problem	Complexity	Constant factor	PTAS	Robust algo.
Clique	In P [18]	N/A	N/A	Poly-time [55]
Independent Set	NPC [18]	3 [45] ‡	[39] †	PTAS [51]
Vertex cover	NPC [18]	1.5 [45] ‡	[39] †	
Domination	NPC [18]	5 [45] ‡	[39] †	PTAS [52]
Connected Domination	NPC [18]	6.8 [48] ‡	[17] †	
Coloring	NPC [18,31]	3 [45] ‡		
Clique Partitioning	NPC [15]	3 [15] ‡		

† Algorithm requires disk representation. ‡ Algorithm does not use a disk representation, but graph must be a UDG to ensure running time and/or performance guarantees.

References

1. Alber J, Fiala J (2004) Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. J Algorithm 52(2):134–151
2. Alzoubi KM, Wan PJ, Frieder O (2002) Distributed heuristics for connected dominating sets in wireless ad hoc networks. J Commun Netw 4:22–29
3. Ambühl C, Erlebach T, Mihal'ák M, Nunkesser M (2006) Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In: Diaz J, Jansen K, Rolim JDP, Zwick U (eds) Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques vol 4110 of Lecture Notes in Computer Science. Springer. pp 3–14
4. Baker BS (1994) Approximation algorithms for NP-complete problems on planar graphs. J ACM 41(1):153–180
5. Balasundaram B, Butenko S (2006) Graph domination, coloring and cliques in telecommunications. In: Resende MGC, Pardalos PM (eds) Handbook of Optimization in Telecommunications. Spinger, New York, pp 865–890
6. Bar-Yehuda R, Even S (1982) On approximating a vertex cover for planar graphs. In: STOC '82: Proceeding of the Fourteenth Annual ACM Symposium on Theory of Computing, pp 303–309
7. Bar-Yehuda R, Even S (1985) A local-ratio theorem for approximating the weighted vertex cover problem. Ann Discret Math 25:27–46
8. Basagni S (1999) Distributed clustering for ad hoc networks. In Proceedings of the (1999) International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '99) p 310
9. Breu H (1996) Algorithmic Aspects of Constrained Unit Disk Graphs, PhD thesis. University of British Columbia
10. Breu H, Kirkpatrick DG (1998) Unit disk graph recognition is NP-hard. Comput Geometr Theory Appl 9(1–2):3–24
11. Brooks RL (1941) On coloring the nodes of a network. Proc Cambridge Philos Soc 37:194–197
12. Butenko S, Cheng X, Du DZ, Pardalos P (2003) On the construction of virtual backbone for ad hoc wireless networks. In Butenko S, Murphey R, Pardalos PM (eds), Cooperative Control: Models, Applications and Algorithms. Kluwer Academic, pp 43–54
13. Callahan PB, Kosaraju SR (1995) Algorithms for dynamic closest pair and n-body potential fields. In: Proceedings of the sixth annual ACM-SIAM symposium on discrete algorithms (SODA '95). Society for Industrial and Applied Mathematics, Philadelphia, pp 263–272
14. Cardei M, Cheng X, Cheng X, Du DZ (2002) Connected domination in multihop ad hoc wireless networks. In: Caulfield HJ, Chen SH, Cheng HD, Duro RJ, Honavar V, Kerre EE, Lu M, Romay MG, Shih TK, Wang DVPP, Yang Y (eds) Proceedings of the 6th Joint Conference on Information Science JCIS/Association for Intelligent Machinery, Inc, pp 251–255
15. Cerioli MR, Faria L, Ferreira TO, Protti F (2004) On minimum clique partition and maximum independent set on unit disk graphs and penny graphs: complexity and approximation. In: Latin-American Conference on Combinatorics, Graphs and Applications, vol 18 of Electronic Notes in Discrete Mathematics. Elsevier, Amsterdam, pp 73–79 (electronic)
16. Chatterjee M, Das S, Turgut D (2002) WCA: A weighted clustering algorithm for mobile ad hoc networks. J Cluster Computing, Special Issue on Mobile Ad hoc Netw 5:193–204
17. Cheng X, Huang X, Li D, Wu W, Du DZ (2003) A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. Netw 42(4): 202–208
18. Clark BN, Colbourn CJ, Johnson DS (1990) Unit disk graphs. Discret Math 86:165–177
19. Das B, Bharghavan V (1997) Routing in ad-hoc networks using minimum connected dominating sets. In: Proceedings of IEEE International Conference on Communications, pp 376–380
20. Das B, Sivakumar R, Bharghavan V (1997) Routing in ad-hoc networks using a virtual backbone. In: Proceedings of the International Conference on Computers and Communication Networks (IC3N), pp 1–20
21. Diaz J, Kaminski M (2007) Max-cut and max-bisection are np-hard on unit disk graphs. Theoret Comput Sci 377(1–3):271–276
22. Du H, Jia X, Li D, Wu W (2004) Coloring of double disk graphs. J Global Optim 28(1):115–119
23. Erlebach T, Fiala J (2006) Independence and coloring problems on intersection graphs of disks. In: Bampis E, Jansen K, Kenyon C (eds) Efficient Approximation and Online Al-

gorithms, vol 3484 of Lecture Notes in Computer Science. Springer, pp 135–155
24. Erlebach T, Jansen K, Seidel E (2001) Polynomial-time approximation schemes for geometric graphs. In: SODA '01: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, pp 671–679
25. Even G, Lotker Z, Ron D, Smorodinsky S (2004) Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. SIAM J Comput 33(1):94–136
26. Fiala J, Fishkin AV, Fomin F (2004) On distance constrained labeling of disk graphs. Theoret Comp Sci 326(1–3):261–292
27. Fishkin AV (2004) Disk graphs: A short survey. In: Jansen K, Solis-Oba R (eds) Approximation and Online Algorithms, vol 2909 of Lecture Notes in Computer Science. Springer, Berlin, pp 260–264
28. Funke S, Kesselman A, Meyer U, Segal M (2006) A simple improved distributed algorithm for minimum CDS in unit disk graphs. ACM Trans Sensor Netw 2(3):444–453
29. Gao J, Zhang L (2005) Well-separated pair decomposition for the unit-disk graph metric and its applications. SIAM J Comput 35(1):151–169
30. Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-completeness. WH Freeman, New York
31. Gräf A, Stumpf M, Weißenfels G (1998) On coloring unit disk graphs. Algorithmica 20(3):277–293
32. Gupta R, Walrand J (2004) Approximating maximal cliques in ad-hoc networks. In: Proceedings of the 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '04), pp 365–369
33. Hale WK (1980) Frequency assignment: Theory and applications. Proc IEEE 68(12):1497–1514
34. Havel TF (1982) The Combinatorial Distance Geometry Approach to the Calculation of Molecular Conformation. PhD thesis, University of California, Berkeley
35. Havet F, Kang RJ, Sereni J-S (2005) Improper colouring of unit disk graphs. Electron Notes Discret Math 22:123–128
36. Hliněný P, Kratochvíl J (2001) Representing graphs by disks and balls (a survey of recognition-complexity results). Discret Math 229(1–3):101–124
37. Hochbaum DS (1983) Efficient bounds for the stable set, vertex cover and set packing problems. Discret Appl Math 6(3):243–254
38. Hochbaum DS, Maass W (1985) Approximation schemes for covering and packing problems in image processing and vlsi. J ACM 32(1):130–136
39. Hunt HB 3rd, Marathe MV, Radhakrishnan V, Ravi SS, Rosenkrantz DJ, Stearns RE (1998) Nc-approximation schemes for np- and pspace-hard problems for geometric graphs. J Algorithms 26(2):238–274
40. Krishna P, Chatterjee M, Vaidya NH, Pradhan DK (1995) A cluster-based approach for routing in ad-hoc networks. In: Proceedings of the USENIX Symposium on Location Independent and Mobile Computing, pp 1–8
41. Kuhn F, Moscibroda T, Wattenhofer R (2004) Unit disk graph approximation. In: Proceedings of the 2004 joint workshop on foundations of mobile computing (DIALM-POMC '04) ACM Press, New York, pp 17–23
42. Kuhn F, Zollinger A (2003) Ad-hoc networks beyond unit disk graphs. In: Proceedings of the 2003 joint workshop on Foundations of mobile computing (DIALM-POMC '03). ACM Press, New York, pp 69–78
43. Li J, Liang X, Selveraj H, Muthukumar V, Gewali LP (2005) A novel data structure for unit disk graphs. J Comb Math Comb Comput 54:145–156
44. Malesińska E, Piskorz S, Weißenfels G (1998) On the chromatic number of disk graphs. Networks 32(1):13–22
45. Marathe MV, Breu H, Hunt HB 3rd, Ravi S, Rosenkrantz DJ (1995) Simple heuristics for unit disk graphs. Networks 25:59–68
46. Marathe MV, Radhakrishnan V, Hunt HB 3rd, Ravi SS (1997) Hierarchically specified unit disk graphs. Theor Comp Sci 174(1–2):23–65
47. Matsui T (2000) Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs. In: Akiyama J, Kano M, Urabe M (eds) Discrete and Computational Geometry, vol 1763 of Lecture Notes in Computer Science. Springer, pp 194–200
48. Min M, Du H, Jia X, Huang CX, Huang SC-H, Wu W (2006) Improving construction for connected dominating set with steiner tree in wireless sensor networks. J Global Optim 35(1):444–453
49. Miyamoto Y, Matsui T (2005) Multicoloring unit disk graphs on triangular lattice points. In: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '05). Society for Industrial and Applied Mathematics, Philadelphia, pp 895–896
50. Nemhauser GL, Trotter LE (1975) Vertex packing: structural properties and algorithms. Math Program 8:232–248
51. Nieberg T, Hurink J, Kern W (2004) A robust PTAS for maximum weight independent sets in unit disk graphs. In: Hromkovic J, Nagl M, Westfechtel B (eds) Graph-Theoretic Concepts in Computer Science, vol 3353 of Lecture Notes in Computer Science. Springer, pp 214–221
52. Nieberg T, Hurink JL (2006) A PTAS for the minimum dominating set problem in unit disk graphs. In: Erlebach T, Persiano G (eds) Approximation and Online Algorithms, vol 3879 of Lecture Notes in Computer Science. Springer, pp 296–306
53. Nolan J (2004) Bisectored unit disk graphs. Networks 43(3): 141–152
54. Peeters R (1991) On coloring *j*-unit sphere graphs. Technical report, Department of Economics, Tilburg University
55. Raghavan V, Spinrad J (2003) Robust algorithms for restricted domains. J Algorithm 48(1):160–172

56. Ramaswami R, Parhi KK (1989) Distributed scheduling of broadcasts in a radio network. In: Proceedings of the Eighth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '89) vol 2, pp 497–504
57. Stojmenovic I (ed) (2002) Handbook of Wireless Networks and Mobile Computing. Wiley InterScience
58. Szekeres G, Wilf HS (1968) An inequality for the chromatic number of a graph. J Comb Theory 4:1–3
59. van Leeuwen EJ (2005) Approximation algorithms for unit disk graphs. In: Kratsch D (ed) Graph-theoretic concepts in computer science, vol 3787 of Lecture Notes in Computer Science. Springer, Berlin, pp 351–361
60. Wan P-J, Alzoubi KM, Frieder O (2004) Distributed construction of connected dominating set in wireless ad hoc networks. Mobile Netw Appl 9(2):141–149
61. Wang DW, Kuo Y-S (1988) A study on two geometric location problems. Informat Proc Let 28(6):281–286
62. Wu W, Du H, Jia X, Li Y, Huang SC-H (2006) Minimum connected dominating sets and maximal independent sets in unit disk graphs. Theoret Comp Sci 352(1):1–7

Optimization Software

OS

Almerico Murli[1,2]
[1] University Naples Federico II, Napoli, Italy
[2] Center for Research on Parallel Computing and Supercomputers of the CNR (CPS-CNR), Napoli, Italy

MSC2000: 90C30, 90C26, 90C10

Article Outline

Keywords
See also
References

Keywords

Mathematical software; Optimization; High performance computing

The origin of the term *mathematical software* can be traced back to J.R. Rice, who organized a symposium on this topic in 1969 [20,21]. Today, the term 'mathematical software' refers to accurate, efficient and reliable software for the solution of mathematical problems that arise in scientific and engineering applications. Different applications are described by similar mathematical models, leading to common computational kernels. Mathematical software provides solutions to these kernels, and supplies building blocks for the development of application software. Therefore, the availability of mathematical software simplifies the solution of application problems by relieving users from having to deal with details related to basic algorithms and their implementations, while exploiting the experience and know-how of mathematical software developers that is needed to produce reliable, accurate and robust modules. Mathematical software is therefore the result of the collaboration of experts in different fields of scientific computing, as it is confirmed by the existence of organized mathematical software repositories and cross-indexed catalogs [1,10,18].

The production of mathematical software is a complex process, ranging from the development of algorithms, to their implementation in specific environments, to the development of user-friendly interfaces, and to intensive testing and quality assurance of the final product. Moreover, this activity is largely influenced by the evolution of computer architectures.

To solve real world problems, hardware must be 'dressed' with a suitable suite of software products. This software can be grouped into three main layers that we refer to as *low-level*, *medium-level* and *high-level*. These terms indicate how close the software is to the hardware (with low-level referring to the closest layer), and, at the same time, how close the software is to the real-world problem (with high-level software referring to the layer closest to the application).

We divide mathematical software into the following main categories [23]:

a) *individual routines*, sometimes gathered into *collections*,
b) *packages of basic routines*,
c) *packages for specific mathematical areas*,
d) *general-purpose libraries*,
e) *problem solving environments* (PSE).

Note that problem solving environments have been included among the above categories although they do not consist only of mathematical software, in order to give an idea of the trends in the development of scientific software and of the role of mathematical software in such contexts.

In the field of optimization, a lot of sequential software is available within each one of the above categories. The continuing evolution of most optimization algorithms and software makes hard both the task to give an exhaustive overview of the current (1999) state-of-the-art and the process to rapidly individuate a software among the existing ones which is able to take advantage of the special nature of the particular problem to be solved.

A first step along such direction should be to formulate the optimization problem as one of the standard *optimization paradigms*. This usually leads to a taxonomy, according to which most optimization problems can be classified. One of the most complete classifications can be found in [17]. It provides an up-to-date (1999) online optimization tree guide to the different subfields of optimization and includes an overview of the major algorithms in each area, with pointers to software packages where appropriate. The first two branches of such optimization tree provide pathways through continuous and discrete problems. Following these pathways we are able to meet the most relevant classes of optimization problems, ranging from nonlinear equations and nonlinear least squares for the unconstrained continuous optimization, to linear programming and general nonlinearly constrained problems for the constrained continuous optimization.

The choice of an algorithm, and related software, to solve an optimization problem has to be made taking into account either some intrinsic properties of the problem, such as:

- type of the objective function and constraints,
- size (number of variables and constraints),
- sparsity degree,

or the main factors that determine the computational cost of the algorithm, such as:

- evaluation of objective functions, constraints, and/or derivatives,
- number of evaluations of objective functions, constraints, and/or derivatives,
- number of variables or constraints,
- number of iterations (optimization algorithms are essentially iterative).

We point out that a large number of available optimization software belongs to category c), that is it consists of packages which are specifically aimed to optimization problems and that can be divided into the following two groups:

- *single-class*, that is packages which address a specific problem class;
- *multiple-class*, that is packages that cover more than one class of problems.

Among the software collections belonging to the first group, we mention WHIZARD [25] for linear programming which uses primal, dual and network simplex algorithms, and L-BFGS-B [5] for bound-constrained optimization problems, which uses a limited memory BFGS algorithm and it is suitable for solving large problems.

Among the packages belonging to the multiple-class we mention MINPACK-1 [14] which is intended to solve systems of nonlinear equations and nonlinear least squares problems, and it is based on the trust region concept.

One of the most recently developed packages of the multiple-class group is LANCELOT [6,12], which provides solvers for unconstrained optimization problems, systems of nonlinear equations, bound-constrained optimization problems, and general nonlinearly constrained optimization problems.

We observe that in many cases software packages which can be used for more than one problem class would sacrifice some efficiency by handling some special nature of the main target problem as the general one.

We also mention three general numerical software libraries, NAG, Harwell and IMSL (category d)), which contain optimization capabilities.

In the 1990s, much research effort has been addressed to develop modeling languages and *optimization systems* (category e), that is PSEs). The basic idea of a modeling programming language is to provide user with common notation and familiar concepts to formulate optimization models and examine solutions, while computer manages communication with an appropriate solver. One of the most used modeling languages is AMPL [9], which offers an interactive command environment for setting up and solving linear and nonlinear optimization problems, in continuous or discrete variables.

An optimization system is a complete system formed by modeling languages to formulate the optimization problem, as well as by easy-to-use user interfaces to solve it and other utilities, like report writing,

model management and execution control. An example of currently available optimization systems is the NEOS Server [17] which allows to solve optimization problems automatically over the Internet, with minimal input from the user and with state-of-the-art optimization software without downloading and linking code. Furthermore, the NEOS Server provides derivatives and sparsity patterns determined automatically with ADIFOR, which is a tool for the automatic differentiation of Fortran programs [3].

Finally, we observe that, as pointed out in [7,26], computational kernels that frequently appear in numerical optimization are those of dense and sparse linear algebra (matrix factorization, orthogonalization, preconditioning, etc.). In most optimization software computational kernels are solved by using efficient routines from standard packages designed for basic linear algebra problems (BLAS, LAPACK).

While the quality of mathematical software can be considered satisfactory for 'traditional' computing environments, the widespread and effective use of *high performance computing* (HPC) resources, required for the solution of the so-called *grand challenges*, is still inhibited by the lack of software suitable for such advanced computational environments [23].

Design and implementation of mathematical software for high performance computing environments has to take into account a number of issues in addition to the ones faced for sequential and vector systems. The new features to be dealt with include a variety of processor and memory system architectures, the lack of standard language features for specifying parallel operations or data distribution, and nondeterminism in execution.

The number of available *parallel optimization* packages is small if compared with the large number of sequential packages. Among them, we mention BTN (*block truncated Newton*) [16], for unconstrained minimization in shared and distributed memory computing environments and PDS, a collection of routines for solving unconstrained nonlinear optimization problems using direct search methods, which has been developed for distributed memory architectures [24]. Moreover, versions of GENOS (*generalized network optimization system*), for solving unconstrained optimization problems with network and generalized network constraints, are available for vector and parallel machines [8]. Routines for unconstrained nonlinear problems based on Newton-type methods, are included in the *NAG parallel library*. Other efforts to produce optimization software for HPC environments are gathered into a few projects currently under development. A well know project is MINPACK-2, aimed mainly at developing a version of MINPACK-1 suitable for advanced architectures [2].

Two basic strategies for introducing parallelism into optimization algorithms, and more generally into numerical algorithms, can be identified:

- parallelizing the computational kernels;
- parallelizing the methods.

One of the already mentioned computational kernel in optimization algorithms is the *evaluation of objective functions and/or derivatives*, which can dominate the overall execution time. If the evaluations are computationally intensive, one can exploit parallelism in each of them, and the exploitation depends on the type of the objective function. For example, in the case of partially separable functions, i. e. functions expressed as the sum of element functions on which small subsets of variables have disjoint effects, one can exploit parallelism by having different processors compute different element functions concurrently, as in [11].

With the strategy of 'parallelizing the method', parallelism is introduced at a level higher than computational kernels, often leading to new optimization methods. For example, in the context of quasi-Newton methods, parallelism has been introduced using line-searches that evaluate the objective function at multiple points concurrently. The basic idea consists in choosing several points along a search direction, evaluating the objective function in each of them concurrently, and using the point with the lowest function value at the next iterate.

A further example can be found in the area of global optimization. The idea is to partition the feasible region into subregions, where each processor searches for a local minimum; the global minimum is obtained by comparing the local results (see for example [4,19]).

Finally we observe that an effective implementation of the parallelization strategies described requires the use of *dynamic load balancing techniques*, to ensure as much as possible that the workload is uniformly distributed among the processors and, hence, to minimize processor idle time.

See also

- Continuous Global Optimization: Models, Algorithms and Software
- Large Scale Unconstrained Optimization
- Modeling Languages in Optimization: A New Paradigm

References

1. ACM Trans Math Softw http://www.netlib.org/toms
2. Averick BM, Moré JJ: User's guide for the MINPACK-2 test problem collection. Argonne Nat Lab Math Comput Sci Div 157, http://www.mcs.anl.gov/summaries/minpack93/
3. Bishof C, Carle A, Hovland P, Khademi P, Mauer A (1993) ADIFOR 2.0 user's guide (revision D). ANL/MCS-TM-92
4. Byrd RH, Dert CL, Rinnooy Kan AHG, Schnabel RG (1990) Concurrent stochastic methods for global optimization. Math Program 46:1–29
5. Byrd RH, Lu P, Nocedal J (1997) L-BFGS-B: Fortran subroutines for large scale bound constrained optimization. ACM Trans Math Softw 23:550–560
6. Conn AR, Gould NMI, Toint PHL (1992) LANCELOT: A Fortran package for large scale nonlinear optimization. Ser Comput Math, vol 17. Springer, Berlin
7. Conn AR, Gould NIM, Toint PHL (1994) Large-scale nonlinear constrained optimization: A survey. In: Spedicato E (ed) Algorithms for Continuous Optimization: the State of Art. Ser C: Math and Physical Sci., vol 434. Kluwer, Dordrecht, pp 287–332
8. Dembo JM, Zenios SA (1987) GENOS 1.0 user's guide: A generalized network optimization system. Report 87-13-03, Dept Decision Sci Wharton School, Univ Pennsylvania
9. Fourer R, Gay DM, Kernighan BW (1993) AMPL: A modeling language for mathematical programming. Duxbury Press, Pacific Grove
10. GAMS: Guide to the available mathematical software. Nat Inst Standards and Techn (NIST) http://gams.nist.gov
11. Grienwank A, Toint PHL (1983) Numerical experiments with partially separable optimization problems. In: Dohl A, Eckmann B (eds) Numerical Analysis: Lecture Notes in Mathematics, vol 1066. Springer, Berlin
12. LANCELOT. Dept Comput and Inform, Council Central Lab Res Councils http://www.dci.clrc.ac.uk/Activity/LANCELOT
13. MIPIII (1994) User's manual. Ketron Management Sci.
14. Moré JJ, Sorensen DC, Hillstom KE, Garbow BS (1984) The MINPACK project. In: Cornell WJ (eds) Source and Development of Mathematical Software. Prentice-Hall, Englewood Cliffs
15. Moré JJ, Wright SJ (1993) Optimization software guide. SIAM, Philadelphia
16. Nash SG, Sofer A (1992) BTN: Software for parallel unconstrained optimization. ACM Trans Math Softw 18:414–448
17. NEOS Optimization software: NEOS guide. Optim Techn Center Argonne Nat Lab and Northwestern Univ, http://www-c.mcs.anl.gov/home/otc/
18. Netlib Repository. Univ Tennessee, Knoxville and Oak Ridge Nat Lab, http://www.netlib.org
19. Pardalos PM, Philips A, Rosen JB (1992) Topics in parallel computing in mathematical programming. Sci. Press, Marrickville
20. Rice JR (1969) Announcement and call for papers, mathematical software. SIGNUM Lett 4(3)
21. Rice JR (1971) Mathematical software. Acad. Press, New York
22. Rice JR, Boisvert RF (1996) From scientific software libraries to problem software environments. IEEE Comput Sci Eng Fall
23. Serafino DDI, Maddalena L, Messina P, Murli A (1998) Some perspectives on high performance mathematical software. In: De Leone R, Murli A, Pardalos PM, Toraldo G (eds) High Performance Algorithms and Software in Nonlinear Optimization. Kluwer, Dordrecht
24. Torczon V: PDS: Direct search methods for unconstrained optimization on either sequential or parallel machines http://www-c.mcs.anl.gov/home/otc/
25. Whiz C (1994) Linear programming optimizer. Ketion Management Sci
26. Wright MH (1993) Some linear algebra issues in large-scale optimization. In: Proc NATO ASI Conf Linear Algebra for Large-Scale and Real Time Applications. Kluwer, Dordrecht

Optimization Strategies for Dynamic Systems

ARTURO CERVANTES, L. T. BIEGLER
Chemical Engineering Department Carnegie, Mellon University, Pittsburgh, USA

MSC2000: 93-XX, 65L99

Article Outline

Keywords
Variational Methods
Partial Discretization
  Dynamic Programming
  Sequential Methods
  Sensitivity-Based Gradients
  Adjoint-Based Gradients
Full Discretization
  Multiple Shooting
  Collocation
  NLP Techniques
  Full Space SQP Approaches

Emerging Areas
Addressing Bottlenecks in NLP Solvers
Discrete Decisions In Dynamic Optimization
Multistage Applications
See also
References

Keywords

Dynamic optimization; NLP; Adjoint; Sensitivity; Collocation; SQP

Interest in *dynamic simulation* and *optimization* of chemical processes has increased significantly during the 1980s– 1990s. Common problems include control and scheduling of batch processes; startup, upset, shutdown and transient analysis; safety studies and the evaluation of control schemes. Chemical processes are modeled dynamically using differential-algebraic equations (DAEs). The DAE formulation consists of differential equations that describe the dynamic behavior of the system, such as mass and energy balances, and algebraic equations that ensure physical and thermodynamic relations.

The general *dynamic optimization* problem can be stated as follows:

$$\min_{z(t),y(t),u(t),t_f,p} \varphi(z(t_f), y(t_f), u(t_f), t_f, p) \tag{1}$$

s.t. DAE model:

$$\frac{\mathrm{d}z(t)}{\mathrm{d}t} = F\left(z(t), y(t), u(t), t, p\right) , \tag{2}$$

$$G\left(z(t), y(t), u(t), t, p\right) = 0 , \tag{3}$$

initial conditions:

$$z(0) = z^0 , \tag{4}$$

point conditions:

$$G_s\left(z(t_s), y(t_s), u(t_s), t_s, p)\right) = 0 , \tag{5}$$

bounds:

$$\begin{cases} z^L \le z(t) \le & z^U, \\ y^L \le y(t) \le & y^U, \\ u^L \le u(t) \le & u^U, \\ p^L \le p \le & p^U, \\ t_f^L \le t_f \le & t_f^U, \end{cases} \tag{6}$$

where

φ	is a scalar objective function,
F	are differential equation constraints,
G	are algebraic equation constraints,
G_s	are additional *point conditions* at times t_s,
z	are differential state profile vectors,
z^0	are the initial values of z,
y	are algebraic state profile vectors,
u	are control state profile vectors,
p	is a time-independent parameter vector.

We assume, without loss of generality, that the index of the DAE system is one and that the objective function is in linear *Mayer form*. Otherwise, it is easy to reformulate most problems to this form. Dynamic optimization problems can be solved either by the variational approach or by applying some level of discretization that converts the original continuous time problem into a discrete problem. The first approaches are focused on obtaining a solution to the classical necessary conditions for optimality. These approaches are also known as *indirect methods*.

The methods that discretize the original continuous time formulation can be divided into two categories, according to the level of discretization. Here we distinguish between the methods that discretize only the control profiles (partial discretization) and those that discretize the state and control profiles (full discretization). Basically, the partially discretized problem can be solved either by dynamic programming or by applying a nonlinear programming (NLP) strategy (*direct-sequential*). A basic characteristic of these methods is that at every iteration a feasible solution of the DAE system, for given control values, is obtained by integration. The main advantage of these approaches is that they generate smaller discrete problems than full discretization methods.

The methods that fully discretize the continuous time problem also apply NLP strategies to solve the discrete system and are known as direct-simultaneous methods. These methods can use different NLP and discretization techniques but the basic characteristic is that they solve the DAE system only once, at the optimum. In addition, they have better stability properties than partial discretization methods, especially in the presence of exponentially increasing modes. On the other

hand, the discrete problem is larger and may require special solution techniques.

With this classification we take into account the degree of discretization used by the different methods. Some authors might prefer to classify the methods according to the solution strategy as indirect methods, methods based on *dynamic programming* and direct methods [41].

This article is organized as follows. In the next section we present the description of the *variational methods*. Following this we describe methods that partially discretize the dynamic optimization problem, and we then discuss full discretization methods as well. Finally, we conclude with a brief description of some of the emerging areas related to dynamic optimization such as *interior point methods* and the solution of *mixed integer dynamic optimization* problems and *multistage problems*.

Variational Methods

These methods are based on the solution of the first order necessary conditions for optimality that are obtained from *Pontryagin's maximum principle* [37]. For the problem (1)–(4), the optimality conditions are formulated as a set of differential-algebraic equations

$$\frac{\mathrm{d}z}{\mathrm{d}t} = \frac{\partial H}{\partial \lambda} = F(z(t), y(t), u(t), p)\,, \quad z(0) = z_0\,, \tag{7}$$

$$\frac{d\lambda}{\mathrm{d}t} = -\frac{\partial H}{\partial z}, \quad \lambda(t_f) = \frac{\partial \varphi}{\partial z} + \frac{\partial G_f}{\partial z_f} v_f\,, \tag{8}$$

$$\begin{aligned} \frac{\partial H}{\partial y} &= \frac{\partial F}{\partial y}\lambda + \frac{\partial G}{\partial y}\mu = 0\,, \\ \frac{\partial H}{\partial u} &= 0\,, \\ \int_0^{t_f} \frac{\partial H}{\partial p}\,\mathrm{d}t &= 0\,, \\ G\left(z(t), y(t), u(t), t, p\right) &= 0\,, \end{aligned} \tag{9}$$

where the *Hamiltonian*, H, is a scalar function of the form

$$H(t) = \lambda(t)^{\top} F(t) + \mu(t)^{\top} G(t) \tag{10}$$

and λ, μ are vectors of the *adjoint variables* and v_f is the multiplier associated with the final time constraint, $G_f\,(z(t_f), y(t_f), u(t_f), t_f, p)) = 0$.

The main problem in obtaining a solution to these equations are the boundary conditions. Normally the state variables are assigned initial conditions and the adjoint variables are assigned final conditions. This procedure leads to a *two-point boundary value problem* (TPBVP) that can be solved with different approaches: single shooting, invariant embedding, multiple shooting, *collocation* on finite elements and finite differences.

In the single shooting methods the missing initial conditions values are guessed. Then, an initial value solver integrates the DAE forward and a Newton iteration is applied to adjust the guessed initial conditions so that the final conditions are equal to the given values. The main disadvantage of this method is that in many cases the problem cannot be solved for a given set of guessed initial conditions, due to nonlinearities and instabilities of the DAE system.

Invariant embedding [45] is a procedure for converting the TPBVP to a initial value problem (IVP). It is based on assuming the structure of the solution, and results in solution procedures analogous to the Riccati matrix differential equation. The main disadvantage here is the high dimensionality of the resulting problem.

Multiple shooting methods follow the same idea as single shooting, but now the integration horizon is divided into smaller subintervals. In this way, state variable values are not only guessed at initial time, but also at several points in between. Then the system equations are decomposed by either solving a collocation system for each region or using a direct integrator along the nominal trajectory on each subinterval. The Newton iteration is also needed to enforce the continuity between subintervals. The discretization methods (or global methods) are known as the most stable. The solution to the TPBVP is obtained simultaneously for the whole horizon, so the initial conditions do not need to be guessed.

For the multiple shooting and discretization methods, special decomposition strategies are usually used to decompose the structured linear algebraic system that is obtained at every iteration of the solution procedure. Efficient factorizations schemes, based on structured Gaussian elimination [27,30] and structured orthogonal factorization [50] can be used in order to minimize the computational effort. Although these methods work well for problems without bounds, handling inequality

constraints is difficult, unless a priori information about the active constraints is known.

Partial Discretization

Dynamic Programming

The use of *iterative dynamic programming* (IDP) for the solution of dynamic optimization problems has been limited largely because of the high dimensionality usually associated with it. This problem is often avoided by allowing a very coarse grid, which in some cases can be accurate enough [13]. Although the IDP algorithm is slower than most gradient-based algorithms, it can be useful to cross-check results of relatively small problems ($n < 100$). This is especially true when the global optimum is unknown, as the probability of obtaining the global optimum is usually high once the grid is not poorly chosen [20]. For these techniques the time horizon is divided into P time stages, each of length L. Then, the control variables are usually represented as piecewise constant or piecewise linear functions in each interval. The piecewise linear functions in each interval (t_k, t_{k+1}), usually takes the form

$$u(t) = u_i + \left(\frac{u_{i+1} - u_i}{L}\right)(t - t_i) \ ,$$

where u_i and u_{i+1} are the values of u at t_i and t_{i+1}, respectively.

The dynamic optimization problem is to find u_i, $i = 0, \ldots, P-1$, that minimize a given objective function. The basic search algorithm is the following [33]:

1) Divide the time interval $[0, t_f]$ into P time stages, each of length L.
2) Choose the number of allowable values M for u.
3) Choose an initial profile for each u_i, initial region size r_i, and the contraction factor γ.
4) By using the initial control policy, integrate the system from $t = 0$ to t_f to generate the state trajectory and store the values of the states at the beginning of each time stage, so that the states at $(i-1)$ corresponds to the value of the states at the beginning of stage i.
5) Starting at stage P, integrate the system from $t_f - L$ to t_f using as initial value the states at $P-1$ from step 4 once with each of the allowable values for the control vector. Choose the control u_{P-1} that gives the minimum value for the objective function, and store the value.
6) Step back to stage $P-1$, corresponding to time $t_f - 2L$. For each allowable value of u_{P-2} integrate the system by using as initial value the states at $P-2$ chosen from step 4 and the given control policy (constant or linear). Continue integration until $t = t_f$ using for the last stage the value u_{P-1} from step 5. Compare the M values of the objective function and choose the u_{P-2} that gives the smallest value.
7) Continue the procedure until stage 1, corresponding to the initial time $t = 0$.
8) Reduce the region for allowable control, $r^{k+1} = \gamma\, r^k$, where k is the iteration index. Use the control policy from step 7 as the midpoint for the allowable values for the control u at each stage.
9) Increment the iteration index and go to step 5. Continue the procedure for a specified number of iterations and examine the results.

This algorithm works well when the dynamic optimization problem does not include bounds on state variables. In order to include them, a penalty term has to be added into the objective function to penalize the constraint violation. This can be done by adding a state variable for each inequality that measures the constraint violation over time [35] or by computing the constraint violation at given points in time [20].

Sequential Methods

In the sequential methods, only the control variables are discretized. This is why these techniques are also known as *control parametrization* methods. Given the initial conditions and a given set of control parameters, the DAE system is solved with a differential algebraic equation solver at each iteration. This produces the value of the objective function, which is used by a nonlinear programming solver to find the optimal parameters in the control parametrization. The sequential method is of the feasible path type, that is, in every iteration the DAE system is solved. This procedure is very robust when the system contains only stable modes. If this is not the case, finding a feasible solution for a given set of control parameters can be difficult.

The time horizon is divided into P time stages and at each stage the control variables are represented with a piecewise constant, a piecewise linear or a polynomial approximation [22,48]. Also, a common practice is to use a set of Lagrange polynomials. So, in each stage i,

the control variables can be written as:

$$u_i(t) = \sum_{q=1}^{\text{ncol}} \psi_q \left(\frac{t - t_{i-1}}{L_i} \right) u_{i,q} \,, \tag{11}$$

where $u_{i,q}$ represent the values of the control variables, in stage i at collocation point q. Here, ψ_q is a Lagrange polynomial of order ncol satisfying,

$$\psi_q(\rho_r) = \delta_{q,r} \,.$$

The gradients of the objective function with respect to the control parameters can be calculated with the sensitivity equations of the DAE system or by integration of the adjoint equations.

Sensitivity-Based Gradients

The *sensitivity equations* are found by differentiating the DAE system after the control has been discretized with the parameter set θ [21,45], and for each element, θ_j, we write:

$$\begin{aligned}
&\frac{\partial \left(\frac{\mathrm{d}z}{\mathrm{d}t}\right)}{\partial \theta_j} = \frac{d\left(\frac{\partial z}{\partial \theta_j}\right)}{\mathrm{d}t} \\
&= \left(\frac{\partial F}{\partial z}\right)^{\top} \left(\frac{\partial z}{\partial \theta_j}\right) + \left(\frac{\partial F}{\partial y}\right)^{\top} \left(\frac{\partial y}{\partial \theta_j}\right) + \left(\frac{\partial F}{\partial \theta_j}\right) \,, \\
&\left(\frac{\partial G}{\partial z}\right)^{\top} \left(\frac{\partial z}{\partial \theta_j}\right) + \left(\frac{\partial G}{\partial y}\right)^{\top} \left(\frac{\partial y}{\partial \theta_j}\right) + \left(\frac{\partial G}{\partial \theta_j}\right) = 0 .
\end{aligned}$$

The solution of the sensitivity equations is simplified because the Jacobians in the sensitivity equations ($\partial F/\partial z$, $\partial G/\partial z$, $\partial F/\partial y$, $\partial G/\partial y$) are equal to the DAE system Jacobians calculated at each step (or given number of steps) of the integration. The computational effort is reduced to one matrix multiplication per parameter per Jacobian evaluation. Once the sensitivities of the states with respect to the parameters are known, the gradient of the objective function and the constrains c, can be calculated as follows [21,45]:

$$\begin{aligned}
\frac{d\varphi}{d\theta} &= \left(\frac{\partial \varphi}{\partial \theta}\right) + \left(\frac{\partial z}{\partial \theta}\right)\left(\frac{\partial \varphi}{\partial z}\right) + \left(\frac{\partial y}{\partial \theta}\right)\left(\frac{\partial \varphi}{\partial y}\right) \,, \\
\frac{dc}{d\theta} &= \left(\frac{\partial c}{\partial \theta}\right) + \left(\frac{\partial z}{\partial \theta}\right)\left(\frac{\partial c}{\partial z}\right) + \left(\frac{\partial y}{\partial \theta}\right)\left(\frac{\partial c}{\partial y}\right) \,.
\end{aligned}$$

Although there have been a lot of advances in solving sensitivity equation more efficiently [23], the computational effort of solving them is still an expensive part of the optimization algorithms. The cost of solving these equations is strongly dependent on the number of input variables. Current directions for handling the computational load include exploitation of faster computer hardware and parallel computer programming architectures, as well as more efficient solution strategies that avoid repeated factorization of Jacobians [5].

The methods that are based in this approach cannot treat the bounds on state variables directly, because the state variables are not included in the nonlinear programming problem. Special methods have been developed to address this problem. Most of the techniques for dealing with inequality path constraints rely on defining a measure of the constraint violation over the entire horizon, and then penalizing it in the objective function, or forcing it directly to zero through an end-point constraint [49].

An alternative is to transform the inequality constraints into equalities by adding a square slack variable [26]. The slack variables can then be treated as control variables and the proper bounds can be imposed in the NLP. The problem of this approach is that it can generate high-index problems, and special index reduction techniques have to be applied at the same time.

In [49], inequality path constraints are handled through a hybrid approach which is the result of the combined application of the discretization of these constraints at a finite number of points, and forcing an integral measure of their violation to zero. Each inequality path constraint requires the introduction of an additional ordinary differential equation, and it is converted to $2P$ (P = number of stages) point constraints and one end-point constraint.

The use of initial value solvers that can handle directly the path constraints has also been studied [22]. The main idea is to use an algorithm for constrained dynamic simulation so that any admissible combination of the control parameters produces an initial value problem that is feasible with respect to the path constraints. The algorithm proceeds by detecting activation and deactivation of the constraints during the solution, and solving the resulting high-index DAE system and their related sensitivities.

Adjoint-Based Gradients

The gradients can also be calculated through *adjoint methods* [14,25,40] at a cost independent of the number of input variables. The DAE adjoint equations are determined from the Hamiltonian function (10). The adjoint profiles λ and μ form a semi-explicit index one DAE that can be solved easily. Once the adjoint system (8)–(9) is solved, the gradients are obtained from

$$\delta\varphi = \int_0^{t_f} \left(\frac{\partial F}{\partial u}\lambda + \frac{\partial G}{\partial u}\mu \right) \delta u \, \mathrm{d}t .$$

If the control profile is discretized into piecewise constants u_i. The gradient with respect to u_i can be expressed as

$$\delta\varphi = \int_0^{t_1} \left(\frac{\partial F}{\partial u}\lambda + \frac{\partial G}{\partial u}\mu \right) \mathrm{d}t \, du_1 + \cdots + \int_{t_{P-1}}^{t_P} \left(\frac{\partial F}{\partial u}\lambda + \frac{\partial G}{\partial u}\mu \right) \mathrm{d}t \, du_P$$

and

$$\frac{d\varphi}{du_i} = \int_{t_{i-1}}^{t_i} \left(\frac{\partial F}{\partial u}\lambda + \frac{\partial G}{\partial u}\mu \right) \mathrm{d}t \, .$$

Adjoint methods are not difficult to automate, but they require the storage of the state profiles for the subsequent adjoint calculation. Also, Jacobians for system and adjoint equation integration can be evaluated at different times, in general, sparse LU factors of Jacobians from system equation integration are not used while solving adjoint equations. The use of implicit Runge–Kutta methods that transform the DAE system into discrete-time implicit equations [38] can solve this problem.

As in the sensitivity based methods, in the adjoint-based methods, the bounds of the states variables can not be treated directly. Usually, when state constraints are imposed a separate adjoint system is developed for each constraint. However, if path constraints are handled individually, we face a daunting task because of the number of adjoint systems that must be developed. Special techniques that reduce the number of adjoints variables have been developed to overcome this problem [38]. Other techniques approximate the constraint satisfaction (constraint aggregation methods) by introducing an *exact penalty function* [10,40] or a *Kreis-selmeier–Steinhauser function* [10] into the problem.

Full Discretization

Full discretization methods explicitly discretize all the variables of the DAE system and generate a large scale nonlinear programming problem that is usually solved with a *successive quadratic programming* (SQP) algorithm. These kinds of methods follow a simultaneous approach (or infeasible path approach); that is, the DAE system is not solved at every iteration. It is only solved at the optimum point. Because of the size of the problem, special decomposition strategies are used to solve the NLP efficiently. Despite this characteristic, the simultaneous approach has advantages for problems with state variable (or path) constraints and for systems where instabilities occur for a range of inputs. In addition, the simultaneous approach can avoid intermediate solutions that may not exist, be difficult to obtain, or require excessive computational effort.

The are two main different approaches to discretize the state variables explicitly, multiple shooting [12] and collocation on finite elements [19]. We briefly describe both of them in the following sections.

Multiple Shooting

In these methods the control variables are approximated by suitable parametrizations using only a finite set of control parameters. Usually a piecewise constant or piecewise linear representation is used. On each stage $i = 0, \ldots, P-1$ a time transformation is used [29]

$$\theta_I(\tau, v) = t_i + \tau h_i \, , \quad t_i = t_0 + \sum_{k=0}^{i-1} h_i , \quad \tau \in [0, 1] \, ,$$

with $v = (t_0, d_0, d_1, \ldots, d_{P-1})$, with a dimensionless discretization grid

$$0 = \tau_{i,0} < \tau_{i,1} < \cdots < \tau_{i,m_i} = 1$$

such that $\theta_I(\tau_{i,0}, v) = t_i$ and $\theta_i(\tau_{i,m_i}, v) = t_i + 1$. A piecewise approximation u_i of the control u_i is then defined by

$$\widehat{u}_i(\tau) = \varphi_{i,j}(\tau, q_{ij})$$

using local control parameters q_{ij}. The functions $\varphi_{i,j}$ are given basic functions, typically vectors of polynomials. If a piecewise constant approximation is chosen, this function takes the form $\varphi_{i,j}(\tau, q_{ij}) = q_{ij}$. For a piecewise

linear approximation the functions are expressed as

$$\varphi_{i,j}(\tau, q_{ij}) = q_{ij}^1 + \frac{\tau - \tau_{ij}}{\tau_{i,j+1} - \tau_{ij}} \left(q_{ij}^2 - q_{ij}^1\right) ,$$

$$q_{ij} = \begin{pmatrix} q_{ij}^1 \\ q_{ij}^2 \end{pmatrix}$$

by linear interpolation between the values q_{ij}^1 and q_{ij}^2 at the endpoints of the stage. With this representation, a continuous approximation can be obtained by imposing continuity equations between the stages.

After this, the DAE system is explicitly discretized on each stage $i = 0, \ldots, P-1$ at the points τ_{ij} of the discretization grid using multiple shooting [11,43]. At each grid point the values of the state variables $s_{ij} = (s_{ij}^z, s_{ij}^y)$ are chosen as additional unknowns. In this way a set of relaxed decoupled initial value problems (IVP) is obtained:

$$\begin{aligned} \frac{\mathrm{d}z_i}{d\tau} &= f_i\left(z_i(\tau), y_i(\tau), \varphi_{i,j}(\tau, q_{ij}), p, \theta_i(\tau, \nu)\right) h_i , \\ 0 &= g_i\left(z_i(\tau), y_i(\tau), \varphi_{i,j}(\tau, q_{ij}), p, \theta_i(\tau, \nu)\right) \\ &\quad - g_i\left(s_{ij}^z, s_{ij}^y, \varphi_{i,j}(\tau_{ij}, q_{ij}), p, \theta_i(\tau_{ij}, \nu)\right) \end{aligned} \tag{12}$$

with initial conditions

$$z_i(\tau_{ij}) = s_{ij}^z, \quad y_i(\tau_{ij}) = s_{ij}^y .$$

By including into the NLP the continuity conditions for the differential variables and the consistency conditions

$$0 = g_i\left(s_{ij}^z, s_{ij}^y, \varphi_{i,j}(\tau_{ij}, q_{ij}), p, \theta_i(\tau_{ij}, \nu)\right)$$

as equality constraints, the final solution satisfies the DAE system. With this approach, the inequality constraints for states and controls can be imposed directly at the grid points. For piecewise constant or linear controls this approximation is adequate, but path constraints for the states may not be satisfied between grid points. This problem can be avoided by applying special techniques to enforce feasibility, like the ones used in the sequential methods.

The resulting NLP is solved using an SQP-type method that requires at each iteration the calculation of the objective function gradient and the constraint Jacobians. For almost all the different functions explicit formulas are available, and the corresponding derivatives can easily be calculated. The only exception is $z_i(\tau_{i,j+1})$, which is computed by numerical integration of the relaxed decoupled IVP, hence the sensitivities with respect to initial values and parameters must be determined. This task is performed with the same techniques used in sequential methods. The only difference is that they are applied at every stage, and this allows a parallel implementation of this kind of algorithm. The SQP-type methods used for the solution of the NLP are very similar to the algorithms used after the full discretization using collocation. For this reason, we consider the collocation methods next.

Collocation

The continuous time problem is converted into an NLP by approximating the profiles as a family of polynomials on finite elements. Different polynomial representations are used in the literature. In [16,46] a monomial basis representation [4] for the differential profiles is used. This representation is recommended because of smaller condition number and smaller rounding errors:

$$z(t) = z_{i-1} + h_i \sum_{q=1}^{\mathrm{ncol}} \Omega_q\left(\frac{t - t_{i-1}}{h_i}\right) \left.\frac{\mathrm{d}z}{\mathrm{d}t}\right|_{i,q} , \tag{13}$$

where z_{i-1} is the value of the differential variable at the beginning of element i, h_i is the length of element i, $\mathrm{d}z/\mathrm{d}t_{i,q}$ is the value of its first derivative in element i at the collocation point q, and Ω_q is a polynomial of order ncol, satisfying

$$\begin{aligned} &\Omega_q(0) = 0, \quad \text{for } q = 1, \ldots, \mathrm{ncol} , \\ &\frac{d}{\mathrm{d}t}\Omega_q(\rho_r) = \delta_{q,r} \quad \text{for } q = 1, \ldots, \mathrm{ncol} , \end{aligned}$$

where ρ_r is the collocation point within each element. One disadvantage of the representation (13) is that state path constraints can only be enforced directly at the mesh points dividing each element. However, we can solve this problem by adding bounded algebraic variables to the problem formulation. The control and algebraic profiles are approximated using Lagrange polynomials of the form

$$y(t) = \sum_{q=1}^{\mathrm{ncol}} \psi_q\left(\frac{t - t_{i-1}}{h_i}\right) y_{i,q} , \tag{14}$$

$$u(t) = \sum_{q=1}^{\mathrm{ncol}} \psi_q\left(\frac{t - t_{i-1}}{L_i}\right) u_{i,q} , \tag{15}$$

where $y_{i,q}$ and $u_{i,q}$ represent the values of the algebraic and control variables, respectively, in element i at collocation point q.

Here, ψ_q is a Lagrange polynomial of order ncol satisfying,

$$\psi_q(\rho_r) = \delta_{q,r} .$$

Other authors also prefer to use low order Lagrange polynomials [7,31] for the differential variables.

$$z(t) = \sum_{q=1}^{\text{ncol}} \psi_q \left(\frac{t - t_{i-1}}{h_i} \right) z_{i,q} . \tag{16}$$

NLP Techniques

The large scale NLP problems that arise from the full discretization of the DAE system are usually solved using successive quadratic programming (SQP) methods. These methods can be classified into full space and reduced space approaches.

Full Space SQP Approaches

Full space methods take advantage of the DAE optimization problem structure and the sparsity of the model. They are very efficient for problems with many degrees of freedom [7,8] as the optimality conditions can be easily stored and factored. Two important disadvantages of these methods are that second derivatives of the objective function and constraints are usually required, and special precautions are necessary to ensure descent properties. In [1], a full space algorithm which exploits the almost block diagonal structure of the DAE optimization problem was developed. This approach decouples the optimality conditions for each block of the *quadratic programming* (QP) subproblem using an affine transform. This way, the first order conditions in the state and control variables can be solved recursively, making the effort of solving it increase linearly with the number of blocks. Also in [7,8] a full space method is presented. In this work, the sparsity and the block diagonal structure are also exploited, but the degrees of freedom of the problems solved are relatively large compared to the number of variables. reduced space SQP In process engineering problems, the degrees of freedom are relatively few, as the number of state variables is much larger than the number of control variables. In these cases, a reduced space SQP approach (rSQP) can be very efficient. With this approach, either projected Hessian matrices or their quasi-Newton approximations may be used, avoiding the necessity of second derivatives. An efficient algorithm can be constructed by decoupling the search direction into its components in range and null spaces and solving a smaller QP subproblem at every iteration. When using collocation, this decomposition allows to exploit the structure of the collocation matrix [32] decreasing the computational effort of these methods.

A partially reduced strategy using multiple shooting was developed in [42] and more recently in [29]. In this strategy, the structured NLP is projected onto the reduced space of differential variables plus control parameters, utilizing the natural decomposition of the discretized states into differential and algebraic variables. This algorithm is particularly efficient for problems with relatively large number of algebraic constraints. In addition to these methods, specialized decomposition procedures that take advantage of the structure of the Hessian were explored in [44].

Emerging Areas

In this final section we briefly summarize areas of research that emerge (as of 2000) for dynamic optimization. These extend the methods presented so far to larger and more challenging applications and can be classified as improvements to nonlinear programming solvers, extensions to include discrete decisions and the treatment of multistage dynamic systems.

Addressing Bottlenecks in NLP Solvers

Several features in the above NLP strategies lead to performance bottlenecks for dynamic optimization. As the problem size increases, the selection of the correct active set can be expensive, especially for tightly constrained NLPs. It has been noted that the normally efficient active set strategies in [7,8] can become time consuming as many state and control profiles become constrained by their bounds. To overcome this problem, *interior point* and *barrier methods* allow us to deal with many active constraints in an efficient manner. This advantage is rooted in improved complexity properties of interior point methods. Whereas active set strategies have an exponential worst-case complexity, inte-

rior point methods typically have a complexity proportional to a low power of the number of variable bounds. In actual practice the resulting performance for active set strategies exhibits a polynomial increase in the number of active set iterations, whereas the number of interior point iterations is independent of problem size.

The barrier (or interior point) method can be motivated by representing the NLP (resulting from a discretization of (1)–(6)):

$$\begin{cases} \min & f(x) \\ \text{s.t.} & c(x) = 0, \\ & x \geq 0 \end{cases}$$

as

$$\begin{cases} \min & f(x) - \mu \sum_i \ln(x_i) \\ \text{s.t.} & c(x) = 0 \end{cases}$$

and solving the equality constrained problem for a decreasing sequence of positive μ. This transformation can be applied directly to the NLP or at the QP level for the subproblem derived from an SQP algorithm. In the latter case, an interior point method can be derived that follows a central path for decreasing μ. Very efficient implementations (e. g., predictor corrector algorithms [34] have been developed for this purpose and these have desirable convergence rates. However, despite these properties, interior point QP solvers, embedded within SQP, are not competitive with active set strategies unless the number of active constraints is large. Here active set solvers take advantage of warm starts from previous QP solutions, while so far interior point solvers still require a fixed computational cost (typically about ten linear factorizations and solutions of the KKT system) regardless of the number of active constraints. To reduce this fixed cost to only one KKT factorization per NLP integration, it becomes advantageous to develop the barrier method at the NLP level. Recently efficient barrier NLP solvers have been developed, including the LOQO solver in [47] and the NITRO solver in [15]. For these methods there are still some limitations on convergence properties, due to nonconvex NLPs.

NLP methods based on interior point concepts allow us to exploit directly all of the features mentioned above for dynamic systems. Examples that demonstrate the performance of these approaches include the solution of linear *model predictive control* (MPC) problems [39] and nonlinear MPC problems [1] using interior point QP solvers and the solution of large optimal control problems using barrier NLP solvers [17].

Discrete Decisions In Dynamic Optimization

Along with the DAE models described in (2)–(3), it becomes important to consider the modeling of discrete events in many dynamic simulation and optimization problems. In chemical processes, examples of this phenomena include phase changes in vapor-liquid equilibrium systems, changes in modes in the operation of safety and relief valves, vessels running dry or overflowing, discrete decisions made by control systems and explosions due to accidents. These actions can be reversible or irreversible with the state profiles and should be modeled with appropriate logical constraints. An interesting presentation on modeling discrete events can be found in [6]. The simulation of these events is often triggered by an appropriate discontinuity function which monitors a change in the condition and leads to a change in the state equations. These changes can be reformulated either by using complementarity conditions (with positive continuous variables x and y alternately set to zero) [24] or as binary decision variables [6]. These additional variables can then be embedded within optimization problems. Here complementarity conditions can be reformulated through smoothing [18] to yield an NLP while the incorporation of integer variables leads to mixed integer optimization problems.

For the latter case, several studies have considered the solution of mixed integer dynamic optimization (MIDO) problems. In particular, [3] developed a complete discretization of the state and control variables to form a mixed integer nonlinear program. On the other hand, [2] apply a sequential strategy and discretize only the control profile. In this case, careful attention is paid to the calculation of sensitivity information across discrete decisions that are triggered in time.

Multistage Applications

The ability to solve large dynamic optimization problems and to model discrete decisions allows the inte-

gration of multiple dynamic systems for design and analysis. Here different dynamic stages of operation can be considered with individual models for each dynamic stage. Multistage applications in process engineering include startups and transients in dynamic systems with different modes of operation, design and operation of periodic processes with different models (e. g., adsorption, regeneration, pressurization, in a dynamic cycle, [36]), synthesis of chemical reactor networks [28], changes in physical phenomena due to discrete changes (as seen above) and multiproduct and multiperiod batch plants where scheduling and dynamics need to be combined and different sequences and dynamic operations need to be optimized.

For these applications each stage is described by separate state variables and models as in equations (2)–(3). These stages include an overall objective function with parameters linking among stages and control profiles that are manipulated within each stage. Moreover, multistage models need to incorporate transitions between dynamic stages. These can include logical conditions and transitions to multiple models for different operation. Moreover, the DAE models for each stage require consistent initializations across profile discontinuities, triggered by discrete decisions.

The solution of multistage optimization problems has been considered in a number of recent studies. See [9] for the simultaneous design, operation and scheduling of a multiproduct batch plant by solving a large NLP. More recently (as of 2000), multistage problems have been considered as mixed integer problems using sequential strategies [2] as well as simultaneous strategies [3,28]. These applications only represent the initial stages of dynamic systems modeling, in order to deal with an integrated analysis and optimization of large scale process models. With the development of more efficient decomposition and solution strategies for dynamic optimization, much more challenging and diverse multistage applications will continue to be considered.

See also

- ► Dynamic Programming: Continuous-Time Optimal Control
- ► Dynamic Programming: Infinite Horizon Problems, Overview
- ► Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ► Dynamic Programming: Optimal Control Applications
- ► Dynamic Programming: Stochastic Shortest Path Problems
- ► Hamilton–Jacobi–Bellman Equation
- ► Infinite Horizon Control and Dynamic Games
- ► Quasidifferentiable Optimization: Stability of Dynamic Systems

References

1. Albuquerque J, Gopal V, Staus G, Biegler LT, Ydstie BE (1997) Interior point SQP strategies for large-scale structured process optimization problems. Comput Chem Eng 23:283
2. Allgor R, Barton P (1999) Mixed integer dynamic optimization I: Problem formulation. Comput Chem Eng 23(4–5):457
3. Avraam M, Shah N, Pantelides C (1998) Modeling and optimization of general hybrid systems in continuous time domain. Comput Chem Eng 22:S221
4. Bader G, Ascher UM (1987) A new basis implementation for mixed order boundary value ODE solver. SIAM J Sci Comput 8:483–500
5. Barton PI, Allgor RJ, Feehery WF, Galan S (1998) Dynamic optimization in a discontinuous world. Industr Eng Chem Res 37:966–981
6. Barton P, Park T (1997) Analysis and control of combined discrete/continuous systems: Progress and challenges in the chemical process industries. AIChE Symp Ser 93(316):102
7. Betts JT, Frank PD (1994) A sparse nonlinear optimization algorithm. J Optim Th Appl 82:543
8. Betts JT, Huffman WP (1992) Application of sparse nonlinear programming to trajectory optimization. J Guidance, Dynamic Control 15:198
9. Bhatia T, Biegler LT (1996) Dynamic optimization in the design and scheduling of multiproduct batch plants. I–EC Res 35(7):2234
10. Bloss KF, Biegler LT, Schiesser WE (1999) Dynamic process optimization through adjoint formulations and constraint aggregation. Industr Eng Chem Res 38:421–432
11. Bock HG, Eich E, Scloeder JP (1988) Numerical solution of constrained least squares boundary value problems in differential-algebraic equations. In: Strehmel K (ed) Numerical Treatment of Differential Equations
12. Bock HG, Plitt KJ (1984) A multiple shooting algorithm for direct solution of optimal control problems. 9th IFAC World Congress

13. Bojko B, Luus R (1992) Use of random admissible values for control in iterative dynamic programming. Industr Eng Chem Res 31:1308–1314
14. Bryson AE, Ho YC (1969) Applied optimal control: Optimization, estimation, and control. Ginn and Company
15. Byrd RH, Hribar ME, Nocedal J (1997) An interior point algorithm for large scale nonlinear programming. Techn Report, Optim Techn Center, Northwestern Univ
16. Cervantes A, Biegler LT (1998) Large-scale DAE optimization using simultaneous nonlinear programming formulations. AIChE J 44:1038
17. Cervantes AM, Waechter A, Tutuncu R, Biegler LT (2000) A reduced space interior point strategy for optimization of differential algebraic systems. Comput Chem Eng 24:39–51
18. Chen C, Mangasarian OL (1996) A class of smoothing functions for nonlinear and mixed complementarity problems. Comput Optim Appl 5(2):97
19. Cuthrell JE, Biegler LT (1987) On the optimization of differential-algebraic process systems. AIChE J 33:1257–1270
20. Dadebo SA, McAuley KB (1995) Dynamic optimization of constrained chemical engineering problems using dynamic programming. Comput Chem Eng 19(5):513–525
21. Dovi VG, Reverberi AP (1993) Optimal solutions of processes described by systems of differential algebraic equations. Chem Eng Sci 48:2609–2614
22. Feehery WF, Barton PI (1998) Dynamic optimization with state variable path constraints. Comput Chem Eng 22:1241–1256
23. Feehery WF, Tolsma JE, Barton PI (1997) Efficient sensitivity analysis of large-scale differential-algebraic equations. Appl Numer Math (1997):41–54
24. Gopal V, Biegler LT (1997) Nonsmooth dynamic simulation with linear programming based methods. Comput Chem Eng 21(7):675
25. Hasdorff L (1976) Gradient optimization and nonlinear control. Wiley, New York
26. Jacobson DH, Lele MM (1969) A transformation technique for optimal control problems with state variable inequality constraint. IEEE Trans Autom Control:AC-14 457–464
27. Keller HB (1974) Accurate difference methods for two-point boundary value problems. SIAM J Numer Anal 11:305–320
28. Lakshmanan A, Biegler LT (1995) Synthesis of optimal chemical reactor networks. I–EC Res 35(4):1344
29. Leineweber DB (1999) Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models. PhD Thesis, University Heidelberg
30. Lentini M, Pereyra V (1977) An adaptive finite difference solver for nonlinear two-point boundary value problems with mild boundary layers. SIAM J Numer Anal 14:91–111
31. Logsdon JS, Biegler LT (1989) Accurate solution of differential-algebraic optimization problems. Industr Eng Chem Res 28:1628–1639
32. Logsdon JS, Biegler LT (1992) Decomposition strategies for large-scale dynamic optimization problems. Chem Eng Sci 47:851–864
33. Luus R (1993) Piecewise linear continuous optimal control by iterative dynamic programming. Industr Eng Chem Res 32:859–865
34. Mehrotra S (1993) Quadratic convergence in a primal-dual method. Math Oper Res 15:741
35. Mekarapiruk W, Luus R (1997) Optimal control of inequality state constrained systems. Industr Eng Chem Res 36:1686–1694
36. Nilchan S, Pantelides C (1998) On the optimization of periodic adsorption processes. Adsorption 4:113
37. Pontryagin LS, Boltyanskii VG, Gamkrelidge R, Mishchenko E (1962) The mathematical theory of optimal processes. Wiley, New York
38. Pytlak R (1998) Runge–Kutta based procedure for the optimal control of differential-algebraic equations. J Optim Th Appl 97:675–705
39. Rao CV, Rawlings JB, Wright S (1998) On the application of interior point methods to model predictive control. J Optim Th Appl 99:723
40. Sargent RWH, Sullivan GR (1979) Development of feed changeover policies for refinery distillation units. Industr Eng Chem Process Des Developm 18:113
41. van Schijndel J, Pistikopoulos EN (2000) Towards the integration of process design, process control and process operability – Current status and future trends, FOCAPD 99. In: AIChE Symp Ser, vol 96, 99
42. Schulz VH (1997) Solving discretized optimization problems by partially reduced SQP methods. Comput Vis Sci 1
43. Schulz VH, Bock HG, Steinbach MC (1998) Exploiting invariants in the numerical solution of multipoint boundary value problems for DAE. SIAM J Sci Comput 19:440–467
44. Steinbach MC (1995) Fast recursive SQP methods for large-scale optimal control problems. PhD Thesis, University Heidelberg
45. Storen S, Hertzberg T (1995) The sequential linear quadratic programming algorithm for solving dynamic optimization problems – A review. Comput Chem Eng 19s:S495–S500
46. Tanartkit P, Biegler LT (1995) Stable decomposition for dynamic optimization. Industr Eng Chem Res 34:1253–1266
47. Vanderbei RJ, Shanno DF (1997) An interior point algorithm for nonconvex nonlinear programming. Techn Report CEOR SOR-97-21, Princeton Univ
48. Vassiliadis VS (1993) Computational solution of dynamic optimization problems with general differential-algebraic constraints. PhD Thesis, University London
49. Vassiliadis VS, Sargent RWH, Pantelides CC (1994) Solution of a class of multistage dynamic optimization problems 2. Problems with path constraints. Industr Eng Chem Res 33:2123–2133

50. Wright SJ (1992) Stable parallel algorithm for two-point boundary value problems. SIAM J Sci Statist Comput 13:742–764

Optimization Techniques for Phase Retrieval Based on Single-Crystal X-Ray Diffraction Data

ALEXANDER BARTON SMITH,
NIKOLAOS V. SAHINIDIS
Department of Chemical Engineering,
Carnegie Mellon University, Pittsburgh, USA

Article Outline

Introduction
Patterson Methods
MAD
Direct Methods
The Minimal Principle
Maximum Determinant
Entropy Maximization
Conclusion
References

Introduction

The 3D structure of molecules is important for a multitude of reasons, prominent in contemporary studies: structure-property relations, reaction kinetics and dynamics, and drug design. Determining these structures is not straight forward, however, and a variety of techniques have been developed, including X-ray diffraction, NMR spectroscopy, and electron spectroscopy. Despite the many techniques available, at least in the area of protein structures, more than 85% of structures present in the PDB have been solved using X-ray diffraction methods [10].

In a single-crystal X-ray diffraction experiment, X-rays are focused on a molecular structure which has been crystallized. The incident rays are then diffracted and their intensity is sampled using a detector. The resulting pattern is recorded and analyzed, yielding data which primarily includes a reciprocal space metric (h, k, l), termed Miller index, followed by a diffraction intensity at each individual coordinate set. The coordinates (h, k, l) describe an infinite set of parallel planes through a given crystal using the primitive reciprocal lattice vectors as a basis.

An ideal crystal can be described as a periodic arrangement of atoms repeated infinitely in space. This allows for a characterization of 3-D crystal structure in terms of a Fourier series known as the density function:

$$\rho(\mathbf{x}) = \frac{1}{V}\sum_m F_{\mathbf{H}_m} \exp(-2\pi\imath\,\mathbf{H}_m \cdot \mathbf{x}), \tag{1}$$

where V is the unit cell volume, m is an index for the set of reflections, $\mathbf{H}$ is a (h, k, l) Miller index, and $F_{\mathbf{H}}$ is a structure factor defined as:

$$F_{\mathbf{H}} = \sum_j f_j \exp(2\pi\imath\,\mathbf{H} \cdot \mathbf{x}_j), \tag{2}$$

where j is an index for the set of atoms in the structure, f_j is an atomic scattering factor for atom j, and $\mathbf{x}_j$ is the position of atom j. A structure factor can also be written in terms of an amplitude and phase:

$$F_{\mathbf{H}} = |F_{\mathbf{H}}| \exp(\imath\phi_{\mathbf{H}}). \tag{3}$$

Hence, full characterization of a crystal structure requires both amplitude and phase data for a large number of reflections, $\mathbf{H}$. In a traditional X-ray diffraction experiment, the diffraction intensity measured, $I_{\mathbf{H}}$, is directly proportional to the structure factor amplitude, $|F_{\mathbf{H}}|$. Phase data, however, is not directly available, yet vital for reconstructing the density function of a crystal. It is therefore in the difficult task of phase retrieval, for which this article is concerned.

First, a discussion of the Patterson and MAD techniques is presented. Traditionally, these techniques do not directly rely on optimization. Thus, the treatment of these methods is brief, focusing on some interesting uses of optimization in their context. Then, two prominent direct methods are discussed, both of which rely on the solution of difficult nonconvex optimization formulations. Minimal principle methods are addressed first, with particular emphasis on the solution of centrosymmetric structures. Then, maximum entropy methods are presented, with a focus on the maximum determinant method.

Patterson Methods

Derivation of an electron density map requires amplitude and phase information for a large number of reciprocal lattice vectors, $\mathbf{H}$. As mentioned previously in the

introduction, phase information is not directly measurable in a traditional X-ray diffraction experiment, a serious obstacle in calculation of a density map. In 1934, with this severe limitation in mind, it was Arthur Lindo Patterson, who first began to look at a Fourier transform of diffraction intensities instead of structure factors. Such a transform does not require any knowledge of phase information. Naturally, this image is not identical to the original density function. However, certain structural information is still contained in the resultant map. Hence, techniques based on a Fourier transform of intensity, still used in some contemporary structural solution methods, form the global class of Patterson methods.

The Patterson function is simply the convolution of the density function with its inverse:

$$P(\mathbf{u}) = \rho(\mathbf{x}) \star \rho(-\mathbf{x}). \tag{4}$$

The discrete form of the Patterson function is written as:

$$P(\mathbf{u}) = \frac{1}{V}\sum_{m} |F_{\mathbf{H}_m}|^2 \exp(-2\pi\imath\, \mathbf{H}_m \cdot \mathbf{u}). \tag{5}$$

A Patterson map can therefore be constructed from a sufficient set of diffraction intensities measured for a set of reciprocal lattice vectors, $\mathbf{H}$, in a single-crystal X-ray diffraction experiment.

The physical meaning of a Patterson map in relation to the original structure is easiest to interpret when atomic contributions to the density function are considered point-like. Hence, for the time, assume electron density is large at the exact position $\mathbf{x}_j$ of each atom in the crystal and 0 at non-atomic site coordinates. Then, it is clear from (4) that the Patterson function at $\mathbf{u}$ will only be substantial when it is a difference of two atomic position vectors, namely $\mathbf{u} = \mathbf{x}_j - \mathbf{x}_i$. Thus, for every possible combination of interatomic differences we observe a Patterson peak, with intensity proportionally to the product of the atomic scattering factors of the participant atoms. Consequently, a N-atom structure, without consideration of overlap, yields a Patterson map with $N(N-1)$ peaks and a highly pronounced central peak. In other words, a Patterson map is the superposition of many copies of the original electron density map.

Solution by Patterson methods requires a technique for deconvolution of the density function from the Patterson map. In traditional techniques [2], the Patterson function, $P(\mathbf{u})$, is displaced by two vectors, $\mathbf{x}_i$ and $\mathbf{x}_k$. Then, the two shifted Patterson maps are superimposed,

$$P_{ik}(\mathbf{u}) = P(\mathbf{u} - \mathbf{x}_i) + P(\mathbf{u} - \mathbf{x}_k) \tag{6}$$

Ideally, the difference of the two vectors chosen to derive $P_{ik}(\mathbf{u})$ represents an interatomic difference vector. In such a case, the structural image in $P_{ik}(\mathbf{u})$ is enhanced. This procedure is repeated until sufficient information is available to construct the original density function. A Patterson map is particularly useful in the case of heavy atom structures since the atomic scattering factor of an atom is proportional to the atomic number of its composite element. Hence, when a heavy atom is present at position $\mathbf{x}_k$, all peaks at $\mathbf{u} = \mathbf{x}_k - \mathbf{x}_j$ are pronounced in the map. Naturally, from such a Patterson map, it is much easier to deconvolute the original density function.

A function for the selection of appropriate displacement vectors, used for the solution of Patterson maps, is presented in [11] and [9]. The generalized symmetry minimum function, SMF, can be written as:

$$SMF(\mathbf{u}) = \sum_{i} \min_{s} P(R_s\mathbf{x}_i + v_s - \mathbf{u}), \tag{7}$$

where i is the set of sampled points, $\mathbf{x}_i$, in the Patterson map, R represents a symmetry rotation operator, v a translational operator, and s the set of symmetry relations present. This form of the SMF is applicable to the full space of the Patterson map. The positions of atoms are chosen from peaks in $SMF(\mathbf{u})$. SMF is still used in contemporary algorithms for displacement vector selection, and has yielded structural solutions for molecules containing up to 6,000 non-hydrogen atoms in the asymmetric unit [3]. The success, however, is still largely restricted to structures containing heavy atoms.

In terms of optimization, solution by Patterson methods is classically posed in terms of a search over a set of interatomic distance vectors calculated from a Patterson map. In [8] this was done for the solution of myoglobin, based on the 'minimum average' function:

$$\max\ Z = \sum_{m\in M} P_m \Big/ \sum_{m\in M} W_m \tag{8}$$

$$s.t.\ P_m = P(\mathbf{w}_m), \forall m \in M \tag{9}$$

$$W_m = W(\mathbf{w}_m), \forall m \in M \tag{10}$$

$$\mathbf{w} \in \Omega, \tag{11}$$

where M is a subset of all available ratios having the lowest values of P_m/W_m, P_m and W_m are the values of the Patterson and test functions, and $\mathbf{w}$ is a vector chosen from Ω, the set of all possible orientations in the Patterson map. Traditionally, this problem is solved in a stochastic manner: a large set of trials is constructed with randomly generated orientations for $\mathbf{w}$. The merit function is then calculated for each trial, and the best solution is selected subject to further solution filtering.

MAD

Multiple anomalous diffraction, MAD, directly solves the phase problem by means vector analysis of diffraction amplitudes sampled for at least three wavelengths. The list of available elements, which produce anomalous differences is limited by the set of experimentally available wavelengths. With the exception of sulfur, none of the elements commonly found in biological macromolecules yield sufficient anomalous scattering. Hence, some sort of chemical substitution is often required. The calculation of phases is straight forward and simply requires knowledge of the position of the anomalous scatterers in the unit cell.

Since most elements which anomalously scatter are heavier, Patterson methods are often sufficient to locate the substituted atoms. Determining the candidates for substitution and calculation of the heavy atom structure is, however, at times difficult when a large number of substitution sites exist. In [13], a figure of merit is proposed for automated selection of the correct trial heavy atom structures. This figure of merit is composed of four individual tests: Patterson function, difference Fourier, phasing figure of merit, and solvent location.

Direct Methods

Direct methods refers to a class of techniques which rely on probability theory to determine phases in reciprocal space using data from a single-crystal X-ray diffraction experiment. These methods, revolutionary at the time they were introduced, pushed the limit of solvable structures substantially. Unfortunately, the accuracy of direct methods fall as the size of the asymmetric unit increases. In addition, these methods typically rely on near atomic data resolution. Finally, since all direct methods operate primarily in reciprocal space, a Fourier transform of the phases derived is required to construct the electron density function. From this density function, atomic positions are then selected.

Most direct methods rely on the use of origin-independent combinations of phases, most prominent of which are the triplet phase relations:

$$\Phi = \phi_{\mathbf{H}} + \phi_{\mathbf{K}} + \phi_{-\mathbf{H}-\mathbf{K}}. \tag{12}$$

In the sequel, minimal principle methods for phasing are first discussed. These techniques rely on solution of a NLP, with triplets defining the constraint set. Then, focus is placed on the class of maximum entropy methods for phasing, with particular emphasis on maximum determinant methods.

The Minimal Principle

The minimal principle methods are certainly some of the most well-known phasing techniques. Like all direct methods, they represent a phasing technique, which is applied in the reciprocal space of a crystal. Specifically, a phase solution is determined from the minimal principle by minimizing a merit function in the framework of a nonconvex NLP. The optimization program, first proposed by [5], is as follows:

$$\min R_{\min} = \sum_t A_t \left[\cos(\Phi_t) - \omega_t \right]^2 \Big/ \left(\sum_t A_t \right) \tag{13}$$

$$s.t.\ \phi_{\mathbf{H}_t} + \phi_{\mathbf{K}_t} + \phi_{-\mathbf{H}_t-\mathbf{K}_t} = \Phi_t\ , \quad \forall t \in T, \tag{14}$$

$$\phi_{\mathbf{H}_m} \in [0, 2\pi]\ , \quad \forall m \in M, \tag{15}$$

where M denotes the total number of reflections from an X-ray diffraction experiment after all the symmetry equivalent reflections have been removed, T denotes the set of triplet phase invariants, $A_t = 2N^{-1/2}|E_{\mathbf{H}}||E_{\mathbf{K}}||E_{-\mathbf{H}-\mathbf{K}}|$, $\omega_t = I_1(A_t)/I_0(A_t)$, $|E|$ is a normalized structure factor amplitude, N is the number of atoms in the unit cell, I_n is a modified Bessel function of order n, and $\mathbf{H}$ and $\mathbf{K}$ denote Miller indices.

In all practical cases, it has been demonstrated [7] that a set of phases which minimizes $R_{\min}$ and satisfies atomicity constraints represent the true phase solution for a structure.

Most contemporary phasing algorithms, which make use of the minimal principle, rely on a combination of local search and stochastic optimization techniques. One commonly used direct methods package for phasing is known as SnB [16].

For centrosymmetric structures, an integer programming model of the minimal principle has been derived by [12]:

$$\min\ R_{\min} = \sum_t A_t\left[4\beta_t + 1 - 2\omega_t + \omega_t^2\right] \Big/ \left(\sum_t A_t\right) \quad (16)$$

$$s.t.\ \phi_{\mathbf{H}_t} + \phi_{\mathbf{K}_t} + \phi_{-\mathbf{H}_t-\mathbf{K}_t} = 2\alpha_t + \beta_t\,, \quad \forall t \in T, \quad (17)$$

$$\phi_{\mathbf{H}_m} + \phi_{\mathbf{H}_m\mathbf{R}_s} = 1\,, \quad \forall m \in M, \forall s \in S_m\,, \quad (18)$$

$$\phi_{\mathbf{H}_m} = \phi_{\mathbf{H}_m\mathbf{R}_u}\,, \quad \forall m \in M, \forall u \in U_m\,, \quad (19)$$

$$\alpha_t, \beta_t \in \{0,1\}\,, \quad \forall t \in T\,, \quad (20)$$

$$\phi_{\mathbf{H}_m} \in \{0,1\}\,, \quad \forall m \in M\,, \quad (21)$$

where M, as before, denotes the total number of reflections from an X-ray diffraction experiment after all the symmetry equivalent reflections have been removed, S_m is the set of shifted phases related to $\mathbf{H}_m$ by rotational symmetry $\mathbf{R}_s$, and U_m is the set of unshifted phases related to $\mathbf{H}_m$ by rotational symmetry $\mathbf{R}_u$. Solution of this model is nontrivial. In addition, it has been shown [17] that a global solution to the minimal principle model can in fact represent a false minimum, which does not correspond to a true phase solution. False minima result when a true structural solution contains triplet invariants, which sum to an odd multiple of π, termed 'odd triplets'. Such odd triplets are typically absent from strong A-value triplet sets. This has motivated the formulation of a modified integer minimal principle, which solves over a subset, T' of the full triplet set T. The global minimum of this model can easily be obtained by simply finding a phase solution which satisfies the constraint set when all invariants are set to zero:

$$\phi_{\mathbf{H}_t} + \phi_{\mathbf{K}_t} + \phi_{-\mathbf{H}_t-\mathbf{K}_t} = 0\,, \quad \forall t \in T' \quad (22)$$

$$\phi_{\mathbf{H}_m} + \phi_{\mathbf{H}_m\mathbf{R}_s} = 1\,, \quad \forall m \in M, \forall s \in S_m \quad (23)$$

$$\phi_{\mathbf{H}_m} = \phi_{\mathbf{H}_m\mathbf{R}_u}\,, \quad \forall m \in M, \forall u \in U_m \quad (24)$$

$$\phi \in \{0,1\}. \quad (25)$$

This model can be solved in polynomial time, and has been shown to greatly enhance computational efficiency for a variety of structures when compared to a standard crystallography package [12].

Maximum Determinant

One of the first papers, which would lay a foundation for maximum entropy methods for phasing was published in 1950. By constructing the Hermitian forms of the structure factor function in terms of electron density and noting the positivity of electron density, [6] showed that a system of determinants containing certain $F_{\mathbf{H}}$'s must be non-negative. This yielded the concept of a Karle–Hauptman matrix, abbreviated as KH matrix.

Later, [14] proposed a method by which KH matrices could be utilized for phasing of crystal structures. First, consider the Sayre equation:

$$\langle E_{\mathbf{K}} E_{\mathbf{H}-\mathbf{K}} \rangle^{\mathbf{K}} = \theta E_{\mathbf{H}}, \quad (26)$$

where E is a normalized structure factor at a particular Miller index, θ is a normalization constant, and the brackets around the left-hand side indicate an average over $\mathbf{K}$. If substitutions are made, $\mathbf{H} = \mathbf{H}_i - \mathbf{H}_j$ and $\mathbf{K} = \mathbf{L} + \mathbf{H}_i$, in terms of the rows $i \in r$ and columns $j \in c$ of a KH matrix, $\mathbf{L}$ a random vector, (26) reduces to:

$$\langle E_i E_j \rangle^{\star} = \langle E_{\mathbf{L}+\mathbf{H}_i} E_{-\mathbf{L}-\mathbf{H}_j} \rangle^{\mathbf{L}} = \theta E_{\mathbf{H}_i-\mathbf{H}_j} \quad i, \forall j \in c. \quad (27)$$

The values of $E_{\mathbf{H}_i-\mathbf{H}_j}$, in the corresponding KH matrix, represent correlation coefficients, and consequently form a covariance matrix. If the following assumptions are then taken: a large number of atoms in the unit cell, the positions of which are mutually independent, then it is possible to derive a conditional joint probability law of the basic form:

$$p(E_1, \ldots, E_m) = C \exp\left(N \frac{\delta_{m+1}}{D_m}\right), \quad (28)$$

where D_m is the determinant of a KH matrix of order m, denoted by $\mathbf{A}_m$, and δ_{m+1} is the determinant of $\mathbf{A}_m$

with one additional row and column appended. In this situation, the structure factors which compose the $\mathbf{A}_m$ matrix are known, the phases of the appended structure factors are varied. Ultimately, (28), implies that the most probable value for the phases of the structure factors added to $\mathbf{A}_m$ will maximize the determinant δ_{m+1}. A generalized rule was also suggested by Tsoucaris [14]: given a KH matrix which contains structure factors with unknown phases, the most probably set of phases will maximize the determinant of the KH matrix. In terms of an optimization framework:

$$\max\ Z = \det A \tag{29}$$

$$s.t.\ A \succ 0 \tag{30}$$

$$A = A(\phi) \tag{31}$$

$$\phi_{\mathbf{H}_m} \in \{0, 2\pi\}, \quad \forall m \in M. \tag{32}$$

CRUNCH is a well known crystallography package [4] in which the phasing is done primarily based on solution of the maximum determinant formulation. The determinant is maximized using a local search technique in combination with an expression of the derivative of the determinant of $\mathbf{A}$ in terms of element i, j:

$$\frac{\delta \det \mathbf{A}}{\delta \alpha_{ij}} = 2|a_{ij}||b_{ij}| \sin(\beta_{ij} - \alpha_{ij}) \det \mathbf{A}, \tag{33}$$

where α and β are phases of elements a_{ij} and b_{ij} respectively and b_{ij} is related to a_{ij} through the inverse of $\mathbf{A}$. In addition, programs such as CRUNCH will often maximize the determinant for a large number of small matrices until enough phase information is available to compose a full density function. Each of these matrices will typically contain a small amount of overlap to fix origin specification. It has also been shown by [15], that the eigenvalues of KH matrices can be used to assess phase set quality and for phase refinement.

Entropy Maximization

The more general technique of entropy maximization is discussed in detail by [1]. Essentially, the optimization model described involves both entropy and physical considerations. In general, the solution is done by maximizing entropy defined as:

$$S(q) = -\int_V q(\mathbf{x}) \log [\, q(\mathbf{x}) / m(\mathbf{x}) \,] \, \mathbf{d}^3\mathbf{x}, \tag{34}$$

where q is the probability density of atoms and m is a 'prior prejudice', typically taken as a uniform distribution. It then remains to write q in terms of the normalized structure factors U of the crystallography problem. In the space group P1, this is achieved as:

$$q(x) = \frac{1}{V} \sum_M U_{\mathbf{H}_m} \exp(-2\pi\imath \mathbf{H}_m \cdot \mathbf{x}) \tag{35}$$

$$U_{\mathbf{H}} = \int_V q(\mathbf{x}) \exp(2\pi\imath \mathbf{H} \cdot \mathbf{x}) \mathbf{d}^3\mathbf{x}. \tag{36}$$

Solution of the maximum entropy formulation will typically allow phasing of structures beyond the size limitations of other direct methods.

Conclusion

Phase information is not directly measurable from a traditional X-ray diffraction experiment. Numerous techniques have been developed for phasing of crystal structures, most well-known are the Patterson, direct, and MAD methods. Typically Patterson methods rely on the presence of one or more heavy atoms in the structure. MAD methods require the presence of a measurable anomalous difference. Direct methods phase in reciprocal space and are typically subject to size and resolution limits. In the field of direct methods, much work with regard to optimization has been done. Still, despite the age of the crystallography field, many problems are still open in the area of robust and reliable determination of crystal structures.

References

1. Bricogne G (1984) Maximum entropy and the foundations of direct methods. Acta Cryst A40:410–445
2. Buerger MJ (1951) A new approach to crystal-structure analysis. Acta Cryst 4:531–544
3. Burla MC, Caliandro R, Carrozzini B, Cascarano GL, de Caro L, Giacovazzo C, Polidori G, Siliqi D (2006) Use of Patterson-based methods automatically to determine the structures of heavy-atom-containing proteins with up to 6000 non-hydrogen atoms in the asymmetric unit. J Appl Cryst 39:728–734
4. de Gelder R, de Graaff RAG, Schenk H (1993) Automatic determination of crystal structures using Karle–Hauptman matrices. Acta Cryst A49:287–293
5. Debaerdemaeker T, Woolfson MM (1983) On the application of phase relationships to complex structures. XXII Techniques for random phase refinement. Acta Cryst A39:193–196

6. Karle J, Hauptman H (1950) The phases and magnitudes of the structure factors. Acta Cryst 3:181–187
7. Miller R, DeTitta GT, Jones R, Langs DA, Weeks CM, Hauptman H (1993) On the application of the minimal principle to solve unknown structures. Science 259:1430–1433
8. Nordman CE (1972) An application of vector space search methods to the Patterson function of myoglobin. Acta Cryst A28:134–143
9. Pavelcik F (1988) Patterson-oriented automatic structure determination: Getting a good start. Acta Cryst A44:724–729
10. PDB Protein Data Bank (2006) http://www.pdb.org/pdb/home/home.do. Current as of November 2006
11. Simpson PG, Dobrott RD, Lipscomb WN (1965) The symmetry minimum function: High order image seeking functions in X-ray crystallography. Acta Cryst 18:169–179
12. Smith AB, Xu H, Sahinidis NV (2007) An integer minimal principle and triplet sieve method for phasing centrosymmetric structures. Acta Cryst A 63:164–171
13. Terwilliger TC, Berendzen J (1999) Automated MAD and MIR structure solution. Acta Cryst D55:849–861
14. Tsoucaris G (1970) A new method for phase determination. The 'maximum determinant rule'. Acta Cryst A26:492–499
15. van der Plas JL, de Graaff RAG, Schenk H (1998) Karle–Hauptman matrices and eigenvalues: A practical approach. Acta Cryst A54:267–272
16. Weeks CM, Miller R (1999) The design and implementation of SnB version 2.0. J Appl Cryst 32:120–124
17. Xu H, Weeks CM, Deacon AM, Miller R, Hauptman HA (2000) Ill-conditioned Shake-and-Bake: The trap of the false minimum. Acta Cryst A56:112–118

Optimization in Water Resources

LAUREANO F. ESCUDERO
Centro de Investig.-Operat.,
University M. Hernández, Elche, Spain

MSC2000: 90C30, 90C35

Article Outline

Keywords
Problem Description
Stochastic Approach
Nonanticipative Water Resources Policies
Conclusions
See also
References

Keywords

Hydrological exogenous inflow; Water demand; Catchment management; Surface and groundwater resources; Environmental targets; Stochasticity; Scenario analysis; Nonanticipativity water resources policies; Full recourse; Splitting variables; Benders decomposition approach; augmented Lagrangian decomposition approach

Problem Description

The complexity of water-related problems is escalating as the uses of water and the (environmental and others) objectives to fulfill continue to expand. Most of the easier structural solutions for greater water resources utilization have already been implemented and new projects, including interbasin transfers, find some opposition in the Society. In these circumstances the need for a rational water resources planning is becoming stronger than ever as a result of the impact of the changes in the general climatic conditions and the increasing demand of water resources using.

To ensure the successful catchment management of complex water resource systems (interaction of reservoirs and channels in the surface of rivers as well as aquifers and other groundwater resources), it is essential that the most reliable models and supporting tools can be used. In the field of *conjunctive use of water resource systems*, the reality is complex and, so, the models for planning are large (in terms of the number of decision variables) and stochastic (there are parameters such as the hydrological exogenous inflow and demand for different uses whose values cannot be controlled by the decision maker and are uncertain). The property of uncertainty makes the water resources planning difficult to tackle, but yet the solution is critical for a proper utilization of the (scarce) water resources.

The *multiperiod optimization modeling framework* should aim to confer the ability to solve vital problems to water resources planning agencies. The problem consists of *water resources planning under uncertainty on hydrological exogenous inflow and demand* for a set of inter-related (and transboundary) basin systems along a given planning horizon. It should have a direct bearing on the assignment of water resources to the requirements of the different uses, by operating significant demand savings and *minimizing the degradation in qual-*

ity of both water environment and natural environment associated with its use. The main elements of the problem are the water resource sources (such as the *surface and groundwater systems*), the water demand centers to satisfy current and potential future needs (for hydropower generation, irrigation, industrial, domestic, recreative and ecological purposes among others) and the infrastructure of reservoirs and *water transportation systems* including artificial, natural and would-to-be basin and inter-basin channels.

A water resource system is included by a surface subsystem and a groundwater subsystem, both interconnected. The system can be viewed as a physical network whose nodes and arcs are as follows:

1) *Nodes with water storage capacity* (i. e., reservoirs including lakes), where evaporation and losses by infiltration to groundwater should be considered. This type of nodes can have associated hydropower generation units that make use of water but it does not reduced it practically.
2) *Physical junction nodes*. They are points in the river where the waterflow has some modification such as river confluences, hydrological inflows, diversions, etc.
3) *Demand nodes*. Other demand uses are irrigation, urban, industrial, recreation and for ecological purposes among others. They can be represented by consumptive and (partially) nonconsumptive water demand nodes.
4) *Return nodes*. They are nodes (points in the river) where water is (partially) returned from some demand nodes.
5) *On-the-river hydropower nodes*. They are nodes without water storage capacity, so, they can only make use of the waterflow for satisfying hydropower generation needs, but without regulating it, nor reducing it either.
6) *Surface water pumping facilities*. These nodes allow water pumping to upstream reservoirs.
7) *Natural stream arcs*. Different types of arcs can be modeled as network arcs, such as natural channels (i. e., river reaches in multireservoir systems), canals, ditches, interbasin transfers, etc.
8) *Aquifers*. They are nodes from the groundwater system with water storage capacity. Conjunctive use of surface water and groundwater is of great importance in many basins, given the scarcity of water resources and the competition between conflicting uses.
9) *Controlled recharge facilities*. These nodes allow direct injection of surface water into the aquifers.
10) *Groundwater pumping facilities*. These nodes allow direct pumping of groundwater to the natural stream arcs.

The main purpose of optimization in the field consists of determining the water resources availability and demand balancing for each period of the planning horizon under study. In case of no balancing feasible solution, the approach should provide a water resources planning to minimize the weighted unbalancing deviation. The water flows through interbasin transfer channels along a given time planning horizon. Technically, the problem is converted to a *time replicated network*. Novel modeling schemes for *multistage linking constraints* to force upper bounds on the water demand cumulated deficit for given consecutive time periods should be considered, see [8]. This type of constraints force water management policies to avoid disastrous consequences of *drought out events*. For the same purpose a constraint type can be modeled to preserve 'earmarked' reserve stored water in (directly and nondirectly) upstream reservoirs along the river to satisfy potential future needs in selected demand centers at given time periods.

The decision maker should decide about the water volume to be stored at each reservoir and controlled aquifer and, then, the water volume to be released, such that physical structural constraints are satisfied and water utilization policies are prioritized and optimized. These policies are related to environmental objectives, hydroelectrical production, irrigation, urban and industrial demands and other uses, reservoirs' water levels, aquifers' artificial recharge and pumping policies, reserve stored water at reservoirs to satisfy potential future needs at selected demand centers during drought out events, etc. So, the objective function to minimize is the expected value of a composite function included by the penalization of the deficits on the satisfaction target levels, the weights of water flow through natural stream and surface pumping arcs and the weights of water pumping from aquifers along the planning horizon.

As an important byproduct the system should determine the risk of significant water deficiencies and to

mitigate the consequences of extreme events as drought outs and floods. The assessment of the impact on water balancing due to water channelling infrastructure modifications, as well as modifications on water resource using policies for some demand types are some other useful results. See some approaches in [1,2,7,9] and [12] - [13] among others.

Other results from using optimization schemes for water resources planning are as follows.

- Assessing environment protection by enforcing lower bounds on the flow through natural stream arcs and water quality.
- Assessing the degree of systems' reliability.
- Determining the risk of significant water deficiencies even over very extended areas.
- Qualifying the water demand according to the requirements of the different uses, and assessing the impact on each other use.
- Advancing and quantifying the potential repercussions on the Environment and the Economy of certain water utilization policies at given time periods under a variety of potential scenarios.
- Determining the structural works and management changes that should be performed to mitigate disastrous consequences of drought out and flood events.
- Assessing the *rational use of groundwater* by helping to decide when and how much aquifer pumping should be performed (and when and how much aquifer artificial recharge should be commanded) to preserve their structural constitution, given the scenario tree that is foreseen (see below), and the ranking and weighting of the demand uses to consider.
- Assessing the need and timing of inter-basin transfers by considering the potential scenarios to occur and the demand uses in the different river basin areas. Similar impact for transboundary rivers.

Stochastic Approach

The optimization problem described above can be expressed in the following model structuring,

$$\begin{cases} \min\limits_{v} & c^{\top} v \\ \text{s.t.} & Av = p \\ & v \geq 0, \end{cases} \tag{1}$$

where c is the vector of the objective function coefficients, A is the $m \times n$ constraint matrix, p is the right-hand side m-vector and v is the n-vector of the decision variables to optimise. It must be extended in order to deal properly with uncertainty in the values of some parameters, say, c and p in this case, hydrological exogenous inflow and demand in various uses. The class of optimization problems with uncertainty in the parameters is among the most intractable class in numerical computation.

In any case one needs to consider two additional features. In the first place, one must model the availability of hydrological information over time, and state what sort of water resource using decisions can be made at each of the various stages. Secondly, to compute an optimal water resources solution in the stochastic area any proposed solution should also be compared with other candidate solutions as it is done in the deterministic field. But, in the stochastic setting, the criteria by which this comparison can be performed are much less clear. Thus, one needs an approach to model the uncertainty in the problem data. The traditional approach is to make probabilistic distribution assumptions, estimate the parameters from historical data and, then, develop an stochastic model to take the uncertainty into account. Such an approach may not be appropriate if only limited information is available. In many such cases one may employ a technique so-called *scenario analysis*, where the uncertainty is modeled via a set of scenarios [6].

Let S denote the set of scenarios to consider, and w^s the likelihood that the decision maker assigns to scenario s for $s \in S$. So, in contrast to traditional mathematical programming approaches, state-of-the-art schemes model the uncertainty by using scenarios to characterize the uncertain parameters. A scenario tree is generated and, through the use of *full recourse* techniques, an implementable solution is obtained for the first time stage by considering all scenarios but without subordinating to any of them; additionally, a coordinated solution for each scenario group at the other time stages should also be provided. While this approach is used, the so-called deterministic equivalent model (DEM) has a huge number of constraints and variables. So, very often the problem structure (network-like and others) is lost as a consequence of the need to impose additional conditions on the value of the variables to ensure the coherence of the water resource using decisions taken at different time stages.

The minimization of the expected value of the composite function included by the penalization of the deficit on the satisfaction target levels can be expressed

$$\begin{cases} \min\limits_{v} & \sum\limits_{s\in S} w^s c^{s\top} v \\ \text{s.t.} & Av = p^s, \quad \forall s \in S, \\ & v \geq 0. \end{cases} \tag{2}$$

Note that (2) gives an implementable policy based on the so-called *simple recourse* scheme. (See that the whole vector of decision variables is anticipated at stage 1).

Nonanticipative Water Resources Policies

Model (2) does anticipate decisions in v that for multistage environments may not be needed at stage $r = 1$. Very frequently the decisions for stage $r = 1$ are the decisions to be made since at stage $r = 2$ one may realize that some of the data has been changed, some scenarios vanish, etc. In this case, the model will be usually reoptimized in a rolling planning horizon mode. When only spot decisions (i. e., decisions for the first stage) are to be made, the information about future uncertainty is taken into account for a better spot decision making. This type of scheme is termed *full recourse*.

Let R denote the set of stages and v_r^s the vector of the variables related to stage r under scenario s for $r \in R$ and $s \in S$, and v^s is the set of vectors $v_r^s \; \forall r \in R$. The so-called *nonanticipative principle* is stated as follows, see [15]: If two different scenarios, say, s and s' are identical up to stage r on the basis of the information available about them up to that stage, then the values of the v-variables must be identical up to stage r. This principle guarantees that the solution obtained from the model is not dependent at stage r on the information that is not yet available. To illustrate this concept, consider a so-called *scenario tree* where each node represents a point in time where a decision on water resource using can be made. Once a decision is being made several contingencies can happen, and information related to these contingencies is available at the beginning of the next stage. This information structure is visualized as a tree, where each root-to-leave path represents one specific scenario and corresponds to one realization of the uncertain parameters.

In order to introduce the implications of this principle, see [8], in water resources planning optimization, let us define a set of scenario groups, say, G_r for each stage r, such that all scenarios having the same realizations of the uncertainty up to stage r belong to the same scenario group, say, g for $g \in G_r$. Let $S_{g,r}$ denote the set of scenarios that belong to group g at stage r for $S_{g,r} \subseteq S$. Let a node in the scenario tree be represented by the pair, say, (k, r) for $k \in G_r$, $r \in R$, such that the scenario tree is defined by the set of nodes $\cup_k \in G_r$, $r \in R(k, r)$ and the set of directed arcs E, where $(k, \ell) \in E$ if and only if S_ℓ, $r+1 \subseteq S_{k,r}$ for $k \in G_r$ and $\ell \in G_{r+1}$. Let $G_r^k \equiv \{\ell \in G_{r+1}\}(k, \ell) \in E$. Finally, let N denote the set of solutions that satisfy the so-called *nonanticipativity constraints*. That is,

$$v \in N \equiv \left\{ v^s : \begin{array}{c} v_r^s = v_r^{s'}, \\ \forall s, s' \in S_{g,r}, \\ g \in G_r, r \in R \end{array} \right\}. \tag{3}$$

So, the DEM of the so-called *full recourse* version of model (1) can be expressed

$$\begin{cases} \min\limits_{v} & \sum\limits_{s\in S} w^s c^{s\top} v^s \\ \text{s.t.} & Av^s = p^s, \quad \forall s \in S, \\ & v \in N, \\ & v^s \geq 0, \quad \forall s \in S. \end{cases} \tag{4}$$

Model (4) has a nice structure that we may exploit. Two approaches can be used to represent the nonanticipativity constraints (3). One approach is based on a *compact representation*, where (3) is used to eliminate variables in (4) as well as for reducing model size, so that there is a single variable for each element at each scenario group of each stage, but any special structure of the constraints in (1) is destroyed. In this case let the variables vector $v = (x, y, z)$ have the following structure: $x_{g,r}$, vector of variables with nonzero coefficients in the constraints related to stage r alone for $g \in G_r$, $r \in R$; $y_{g,r}$, vector of variables with nonzero elements in the constraints related to the stages r and $r+1$ (for the water resources planning problem this type of variables represent the stored water in the reservoirs and aquifers at the last time period of stage r); and $z_{g,r}$, vector of variables with nonzero elements in the constraints related to stage r as well as in the constraints related to sets of stages to be defined below.

Introduce the following additional notation. U is the set of z-related constraint blocks through time stages, so-called *multistage linking constraints*, R_u is the set of time stages related to constraint block u for $\in U$, $\underline{r}_u$ and $\overline{r}_u$ are the smallest and largest elements from R_u, respectively, and $N_{g,u}$ is the set of nodes in the directed path through the set of time stages (i. e., set R_u) whose ending node is node $(g, \overline{r}_u)$ and the unique origin node is, say, $(i, \underline{r}_u)$. So, the pair (k, r) index for variable $z_{k,r}$ is such that $(k, r) \in N_{g,u}$ for ending node $(g, \overline{r}_u)$ and constraint block u. This type of constraint block can be represented as follows:

$$Z_{g,u} \quad \begin{cases} \sum_{(k,\tau)\in N_{g,u}} D_{u,\tau} z_{k,\tau} = d_{g,u} \\ \forall g \in G_{\overline{r}_u}, \qquad\qquad u \in U, \end{cases} \tag{5}$$

where $D_{u,\tau}$ is the matrix for constraint block u related to the z-variables from stage τ, and $d_{g,u}$ is the right-hand side of constraint block u for scenario group g from stage $\overline{r}_u, u \in U$. (See that constraint block u has $|G_{\overline{r}_u}|$ versions.) For the water resources planning this type of constraints prevent that the cumulated water demand deficit in given nodes through consecutive time stages can violate given upper bounds under given scenario groups.

The compact representation of model (4) can be expressed as follows:

$$\begin{cases} \min\limits_{x,y} & \sum\limits_{r\in R}\sum\limits_{g\in G_r} w_{g,r} \\ & \quad \cdot(a_{g,r}^\top x_{g,r} + b_{g,r}^\top y_{g,r} + c_{g,r}^\top z_{g,r}) \\ \text{s.t.} & 0 \le x_{g,r}, y_{g,r}, z_{g,r} \in X_{g,r} \\ & \forall g \in G_r, \quad r \in R, \\ & z_{g,r} \in Z_{g,u}, \quad \forall g \in G_{\overline{r}_u}, \quad u \in U, \end{cases} \tag{6}$$

such that

$$X_{g,r} \quad \begin{cases} A_r x_{g,r} + B'_r y_{\ell,r-1} + B_r y_{g,r} \\ \quad + C_{g,r} z_{g,r} = p_{g,r} \\ \forall g \in G_r, \quad r \in R, \end{cases} \tag{7}$$

where $w_{g,r}$ gives the weight for scenario group g at stage r, such that $a_{g,r}$, $b_{g,r}$ and $c_{g,r}$ are the x-, y- and z-variables related objective function coefficients respectively, for the pair (g, r), A_r, B_r and C_r are the appropriate constraint matrices and $p_{g,r}$ is the right-hand side, all with the conformable dimensions, and ℓ: $\{g \in G^{\ell}_{r-1}$ for $g \in G_r, r \in R\}$, and

$$w_{g,r} = \sum_{s\in S_{g,r}} w^s \,. \tag{8}$$

One of the main inconveniences of the compact representation (5)–(7) is the inherent difficulty for its decomposition in smaller models. Given the large scale instances of the model, easy decomposition is a key for success. It can be obtained from the so-called *splitting variable representation*. It requires to produce *sibles* of the y- and z-variables.

For this purpose let $N^{g,r}$ denote the set of pairs (k, τ) such that $k \in G_\tau$, $\tau \in R\tau \ge r$ and $\exists u \in \frac{U}{\tau} = \overline{r}_u$ for $(g, r) \in N_{k,u}$. That is, (g, r) and (k, τ) for $g \in G_r$ and $k \in G_\tau$ are any two nodes in the scenario tree for $r, \tau \in R$, such that there is a constraint block u for $u \in U$ where $\tau \equiv \overline{r}_u$ and there is a path from some node, say, $(i, \underline{r}_u)$ to node (k, τ) through node (g, r).

In order to introduce the new representation, let us rename the y- and z-variables such that $y_{g,r}$ and $z_{g,r}$ will be replaced by $y^0_{g,r}$ and $z^{g,r}_{g,r}$, respectively, and add the new variables $y_\ell, r-1^g$, where $\ell : g \in G^{\ell}_{r-1}$, and z^k, $\tau_{g,r}$, $\forall (k, \tau) \in N^{g,r}$, $g \in G_r$, $r \in R$. So, the splitting variable representation is as follows:

$$\begin{cases} \min\limits_{x,y,z} & \sum\limits_{r\in R}\sum\limits_{g\in G_r} w_{g,r} \\ & \quad \cdot(a_{g,r}^\top x_{g,r} + b_{g,r}^\top y^0_{g,r} + c_{g,r}^\top z^{g,r}_{g,r}) \\ \text{s.t.} & X_{g,r}, \\ & Z_{g,u}, \\ & Y^{\ell}_{g,r}, \\ & Z^{k,\tau}_{g,r}, \\ & x, y, z \ge 0, \end{cases} \tag{9}$$

where

$$X_{g,r} : \left\{ \begin{array}{c} A_r x_{g,r} + B'_r y^g_{\ell,r-1} + B_r y^0_{g,r} \\ + C_r z^{g,r}_{g,r} = p_{g,r}, \\ \forall g \in G_r, r \in R, \\ \ell : g \in G^{\ell}_{r-1} \end{array} \right\}, \tag{10}$$

$$Z_{g,u} : \left\{ \begin{array}{c} \sum_{(k,\tau)\in N_{g,u}} D_{u,\tau} z^{g,\overline{r}_u}_{k,\tau} = d_{g,u}, \\ \forall g \in G_{\overline{r}_u}, \quad u \in U, \end{array} \right\}, \tag{11}$$

$$Y^{\ell}_{g,r} : \left\{ \begin{array}{c} y^{\ell}_{g,r} - y^{\ell+1}_{g,r} = 0, \\ \forall \ell \in \{0\} \cup G^g_r, \; g \in G_r, \; r \in R \end{array} \right\}, \tag{12}$$

$$Z_{g,r}^{k,\tau} : \left\{ \begin{array}{c} z_{g,r}^{g,r} - z_{g,r}^{k,\tau} = 0, \\ \forall (k,\tau) \in N^{g,r},\ g \in G_r,\ r \in R \end{array} \right\} . \quad (13)$$

Remark 1 The two constraint blocks (12) and (13) in (9) are the expressions for the nonanticipativity constraints (3).

Different types of decomposition approaches can be used for solving model (8); namely, augmented Lagrangian and Benders [3] decomposition schemes, both being very amenable for using parallel computing approaches, see [4,5,10,11,14,16], and [17] among others.

Conclusions

Full recourse based mathematical programming schemes have been used as the kernels of decision support systems for water resource planning under water exogenous inflow and demand uncertainty, where the uncertainty is treated via scenario analysis. This methodology results in a huge deterministic equivalent model (with hundreds of thousands of constraints and variables), where care should be taken to preserve the constraint structure of the original problem. Two very useful constraint types are considered for the demand centers, namely, upper bounds on the deficit of reserve stored water in (directly and nondirectly) upstream reservoirs to satisfy potential future needs, and upper bounds on the consecutive time periods cumulated water demand deficit (so-called *multistage linking constraints*).

A splitting variable scheme for the reservoirs and controlled aquifers stored water modeling representation can be used, as well as decomposition frameworks based on Benders and augmented Lagrangian approaches. On the other hand, given the separability of the subproblems attached to the nodes of the scenario tree as well as the reduced overhead required, one can be motivated to develop parallel computing versions of the decomposition approaches on a distributed environment. It will significantly help to solve large scale water resource planning problems under uncertainty in water exogenous inflow and demand parameters.

See also

► Global Optimization in the Analysis and Management of Environmental Systems

References

1. Andreu J, Capilla J (1993) Optimization and simulation models applied to the Segura water resources system. In: Marco JB, Harboe R, Salas JD (eds) Stochastic Hydrology and its use in Water Resources Systems Simulation and Optimization. Kluwer, Dordrecht, pp 425–438
2. Andreu J, Capilla J, Sanchs E (1996) AQUATOOL, a generalized decision support system for water-resources planning and operational management. J Hydrology 177:269–291
3. Benders JF (1962) Partitioning procedures for solving mixed variables programming problems. Numerische Math 4:238–252
4. Birge JR (1985) Decomposition and partitioning methods for multistage stochastic linear programs. Oper Res 33:1089–1107
5. Dempster MAH, Thompson RT (1996) Parallelization of EVPI based importance sampling procedures for solving multistage stochastic linear programmes on MIMD architectures. Parallel Optimization Colloquium. Lab. PRiSM, University Versailles, Versailles
6. Escudero LF (1994) Robust decision making as a decision making aid under uncertainty. Decision Theory and Decision Analysis. In: Ríos S (ed) Kluwer, Dordrecht, pp 127–138
7. Escudero LF Pastor JI (ed) (1996) Modelización de la optimización de la gestión hídrica. Sistemas de Informacín y Control en la Gestión del Agua. IBERDROLA Institute Tecnol., pp 317–338
8. Escudero LF (1998) On using multistage linking constraints for stochastic optimization. Rev Acad Ciencias 92:371–376
9. Escudero LF (2000) WARSYP, a robust modelling approach for water resources system planning under uncertainty. Ann Oper Res 95:313–339
10. Escudero LF, de la Fuente JL, García C, Prieto FJ (1996) Hydropower generation management under uncertainty via scenario analysis and parallel computation. IEEE Trans Power Systems 11:683–689
11. Gassmann MI (1990) MSLiP: A computer code for the multistage linear programming problem. Math Program 47:407–423
12. Labadie JM, Brazil LE, Corbu I, Johnson LE (eds) (1989) Computerized decision support systems for water managers. Amer. Soc. Civil Engineers, Reston
13. Marco JB, Harboe R, Salas JD (eds) (1993) Stochastic hydrology and its use in water resources systems simulation and optimization. Kluwer, Dordrecht
14. Mulvey JM, Ruszczyński A (1992) A diagonal quadratic approximation method for large-scale linear programs. Oper Res Lett 12:205–221
15. Rockafellar RT, Wets RJ-B (1991) Scenario and policy aggregation in optimization under uncertainty. Math Oper Res 16:119–147

16. Ruszczyński A (1993) Parallel decomposition of multistage stochastic programs. Math Program 58:201–208
17. Van Slyke R, Wets RJ-B (1969) L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J Appl Math 17:638–663

Optimizing Facility Location with Euclidean and Rectilinear Distances

JACK BRIMBERG[1], GEORGE O. WESOLOWSKY[2]
[1] Royal Military College of Canada, Kingston, Canada
[2] McMaster University, Hamilton, Canada

MSC2000: 90B80, 90B85

Article Outline

Abstract
Keywords
Introduction
The Model
Euclidean Closest Distances
Rectilinear Closest Distances
See also
References

Abstract

Finding the median (or minisum) point in continuous space relative to a set of weighted demand points (customers, markets, existing facilities) is a classical problem in location theory. We examine an extension of the problem where the existing facilities are modeled as fixed areas, and the median point is determined relative to the *closest* distances to these areas. Two popular distance metrics are considered: the Euclidean and rectangular norms. These demonstrate the differences between the classes of round (or smooth) norms and block norms. Mathematical properties of the new models are used to adapt existing solution methods to solve them.

Keywords

Euclidean norm; Rectangular norm; Closest distance; Median

Introduction

In continuous location theory, facilities to be optimally located are generally represented by points, and the customers or markets that they serve are also geometrical points in space. The objective is to find the optimal site of one or more facilities with respect to a specified performance measure such as the sum of transportation costs. This is one of the oldest formal optimization problems in mathematics and has a long and interesting history ([15] Section 1.3, [7,9,21]). Many variants of the problem exist. A very basic version of the location problem is to minimize:

$$W(x) = \sum_{i=1}^{n} w_i K(x - a_i)\,, \tag{1}$$

where $x = (x_1, x_2)$ is the unknown facility location in $\Re^2$, w_i is a positive weight representing transportation cost per unit distance for customer i, and $K(x-a_i)$ is a norm measuring distance from the facility location x to the fixed location $a_i = (a_{i1}, a_{i2})$ of demand point i. The most common distance measure is Euclidean or straight-line distance and in this case, the most common solution procedure is some form of nonlinear optimization such as gradient descent.

A simple yet elegant solution procedure for the minisum problem (1) with Euclidean (ℓ_2) distances was proposed by Weiszfeld in 1937 [20]. Setting the first-order partial derivatives of $W(x)$ to zero, we obtain the following system of equations for a stationary point:

$$\sum_{i=1}^{n} \frac{w_i(x_t - a_{it})}{\ell_2(x - a_i)} = 0\,, \quad t = 1, 2\,. \tag{2}$$

Since the objective function is convex, the above equations are both necessary and sufficient for any differentiable point to be a global solution of (1). However, since the system of equations cannot be solved explicitly, Weiszfeld developed the following one-point iterative scheme by isolating the coordinates on one side:

$$x_t^{q+1} = \frac{\sum\limits_i w_i a_{it}/\ell_2(x^q - a_i)}{\sum\limits_i w_i/\ell_2(x^q - a_i)}\,, \quad t = 1, 2, \tag{3}$$

where $q = 0, 1, 2, \ldots$, denotes the iteration number.

The Weiszfeld procedure is equivalent to a gradient descent with predetermined step size. In a semi-

nal paper by Kuhn [14], global convergence of the procedure was proven provided that no iterate coincides with a fixed point a_i , where the iteration functions are undefined. The local convergence rate is always linear when the optimal solution occurs at a differentiable point; however, when the optimal solution coincides with a fixed point, the convergence rate may be sublinear or quadratic under certain conditions [13]. The Weiszfeld procedure has been generalized to other distance functions, and convergence properties have been studied e. g. see [1,2] for ℓ_p distances. We note that global convergence is guaranteed for the ℓ_p norm if the parameter $p \in [1, 2]$, that is, for the rectangular norm ($p = 1$), the Euclidean norm ($p = 2$), and all ℓ_p norms in between.

The ℓ_p norm, with $1 < p < \infty$, belongs to the class of round norms that are characterized by unit circles with no 'flat spots'. Block norms, on the other hand, have unit circles that are polygons in $\Re^2$ (or polyhedrons in higher dimensional space). With this property, the objective function in (1) becomes convex piecewise linear. Block norms play a useful role in location models (e. g., see [17,19]), as the search in continuous space may now be confined to a finite set of points, and linear programming or related techniques may be used. Similarly, many other types of problems become much easier to solve when absolute values occur in a certain form. Examples of block norms include the rectangular norm ($p = 1$), Tchebycheff norm ($p = \infty$), and a linear combination of the two that has been used to approximate the ℓ_p norm [18].

When distances are rectangular, that is, when:

$$K(x - a_i) = (|x_1 - a_{i1}| + | x_2 - a_{i2}|) , \qquad (4)$$

minimizing W(x) is accomplished by simply finding weighted median locations separately along the x_1 and x_2 axes (e. g., see [15] Chapter 2, or [9]). For example, if the a_{i1}'s are ordered from smallest to largest, with their weights attached, then the a_{i1} associated with the median of the weights would determine the optimal solution for x_1. The separability property of rectangular distances allows for the solution of more complex problems. For example, the demand points may be replaced by demand 'areas', and the distance becomes the expected distance between the facility and each demand for a given density function of space distributed demand [6,22].

The Model

We now consider the problem of locating a new facility denoted by point $x \in \Re^2$ to service a set of n specified demand regions (or market areas) denoted by $A_i \subset \Re^2$, $i = 1, \ldots, n$. The A_i are assumed to be fixed areas in the plane that are bounded and closed, with known demands again specified by weighting constants, $w_i > 0$, $i = 1, \ldots, n$. The objective is to find the point x that minimizes the weighted sum of distances to the n demand regions.

What makes our problem different from the well-studied minisum problem discussed above is that the travel distance separating the facility from a demand region A_i is now defined as the distance measured by norm K from x to the closest point in A_i. This may be interpreted as flow from the facility entering the given market at the closest entry point. Internal distribution costs within the market area are assumed to be unimportant or 'someone else's concern' (see [4,5] for further discussion).

The idea of closest distances is well known in set theory (e. g. [11]). Area demands have been used many times in location problems, but along with the assumption that travel distances within areas are relevant: e. g. see [6,10,22]. If some form of aggregation is used, then the expected travel distance from the facility to some 'mean' point in the interior of the demand area determines the transportation cost.

Denote the closest point in A_i by $a_i(x)$. Since this point is determined by the intersection of the smallest possible circle of norm K centered at x with A_i, it follows that if A_i is a convex region, and K a round norm, then, $a_i(x)$ is always uniquely defined. The travel distance now becomes:

$$d(x, A_i) = \min_{y \in A_i} K(x - y) = K(x - a_i(x)) . \qquad (5)$$

The single facility minisum problem with closest distance function then takes the form:

$$\min W(x) = \sum_{i=1}^{n} w_i d(x, A_i) = \sum_{i=1}^{n} w_i K(x - a_i(x)) . \qquad (6)$$

Property 1 (Brimberg and Wesolowsky, [4]) If A_i is a convex region, then $d(x, A_i)$ is a convex function of x for any norm K.

It follows that if all the A_i are given as convex regions, then W is a sum of convex functions that is itself convex. In this case a general descent algorithm may be used to obtain the optimal solution. However, more specialized methods may be devised based on properties of the particular model.

Euclidean Closest Distances

We now specify the norm K to be the Euclidean norm, that is,

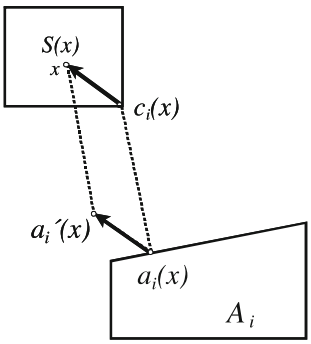
$$d(x, A_i) = \ell_2(x - a_i(x)) = [(x_1 - a_{i1}(x))^2 + (x_2 - a_{i2}(x))^2]^{1/2}, \quad (7)$$
$$\forall x = (x_1, x_2),\ a_i(x) = (a_{i1}(x), a_{i2}(x)) \in \Re^2 .$$

Let $int(A_i)$ and B_i denote, respectively, the interior and boundary of A_i, $i = 1, \ldots, n$. Let $\overline{a_i(x)}$ represent the "fixed point" $a_i(x)$ which is assumed to be unique. When $x \in int(A_i)$, it is clear that $\partial(d(x, A_i))/\partial x_j$ exists, and is equal to 0 for all j. However, since the functional form of $W(x)$ changes crossing from the interior to the exterior of A_i, it also follows that when $x \in B_i$, $\partial(d(x, A_i))/\partial x_j$ is undefined for at least one direction j. The following result [5] provides an interesting relation when x is a point outside A_i.

Property 2 Consider any $x \notin A_i$. Then the partial derivative $\partial(d(x, A_i))/\partial x_j$ is defined $\forall j$; its value at x is the same when $a_i(x)$ is replaced by the fixed point $\overline{a_i(x)}$.

It follows that if x is external to all the A_i, the gradient vector of $W(x)$ may be calculated by replacing all the demand regions by the associated fixed points $\overline{a_i(x)}$. Meanwhile if x is inside a region, we simply delete that region from the calculation. Thus the problem may be converted, at least locally, to the standard form given in (1). The basic idea behind the algorithm given in [5] may now be summarized as follows:

Given an initial location x^0, we determine the closest point $a_i(x^0)$ for each demand region A_i not containing x^0. These $a_i(x^0)$are then treated as fixed points replacing the respective areas A_i. This allows us now to make use of the well-known Weiszfeld procedure introduced above. One Weiszfeld iteration produces a new location x^1 with lower objective function value relative to the set $\{a_i(x^0); i = 1, \ldots, n\}$, although reduction of the step size may be required if x^0 falls within a demand region. Using the new location x^1, we then recalculate the closest points to obtain $\{a_i(x^1); i = 1, \ldots, n\}$, and a further improvement in the objective function. The whole process is repeated to provide a sequence of descent moves. The algorithm will converge to the optimal solution if all the A_i are convex regions; otherwise, since the objective function is no longer convex, we are only guaranteed a local minimum.



Optimizing Facility Location with Euclidean and Rectilinear Distances, Figure 1
Closest distances with an area facility

Suppose now that instead of being a point, the new facility is represented by an area of fixed dimensions and orientation, denoted by $S(x)$, where x is a specified 'center' point. The closest distance between the facility and demand region A_i is then defined as:

$$d(x, A_i) = \min_{z \in S(x), y \in A_i} \{\ell_2(z - y)\} = \ell_2(c_i(x) - a_i(x)), \quad (8)$$

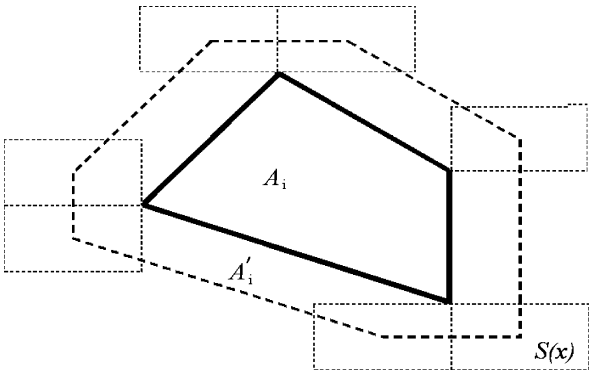
where $c_i(x)$ and $a_i(x)$ are the closest points in $S(x)$ and A_i, respectively. As illustrated in Fig. 1, we may replace $c_i(x)$ by x and $a_i(x)$ by $a'_i(x) = a_i(x) + (x - c_i(x))$.

Using this insight, the problem may be converted to the original form by replacing each A_i by an enlarged area A'_i [5]. This is illustrated in Fig. 2, where $S(x)$ is a rectangle, and A_i a polygon. Note that A'_i is also a polygon but with a larger number of sides.

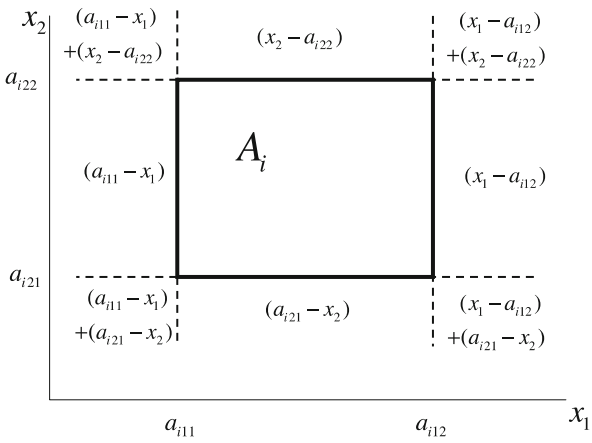
Rectilinear Closest Distances

The norm K is now specified by the rectangular norm; hence,

$$d(x, A_i) = \ell_1(x - a_i(x)) = |x_1 - a_{i1}(x)| + |x_2 - a_{i2}(x)|. \quad (9)$$



Optimizing Facility Location with Euclidean and Rectilinear Distances, Figure 2
Enlarged Area A_i'



Optimizing Facility Location with Euclidean and Rectilinear Distances, Figure 3
Distances from area A_i

The problem now becomes:

$$\min W(x) = \sum_{i=1}^{n} w_i(|x_1 - a_{i1}(x)| + |x_2 - a_{i2}(x)|). \quad (10)$$

If the area A_i is a rectangle, then the distance $d(x, A_i)$ has a very convenient form. Referring to the notation and closest distances given in Fig. 3, it follows that

$$d(x, A_i) = \frac{1}{2} \sum_{j=1}^{2} \sum_{k=1}^{2} |x_j - a_{ijk}| - \tau_i, \quad (11)$$

where

$$\tau_i = \frac{a_{i12} - a_{i11}}{2} + \frac{a_{i22} - a_{i21}}{2} \quad (12)$$

from which we can conclude that this problem is equivalent to solving:

$$\min W(x) = \sum_{i=1}^{n} \sum_{j=1}^{2} \sum_{k=1}^{2} w_i |x_j - a_{ijk}| \quad (13)$$

This problem is now separable, and is very easily solvable by ordering coordinates and taking medians, as discussed previously (see [15], Chapter 2). The physical interpretation of this problem is also interesting. If each rectangle A_i with weight or demand w_i is replaced by two fixed points with the same weights $\boldsymbol{w}_i$ at diagonally opposing vertices, then the original problem becomes an equivalent problem with point demands.

In common with many location models using rectilinear distances, $W(x)$ remains piecewise-linear along any straight line in the solution space when closest rectangular distances are in use. Specifically, $W(x)$ is linear in segments of the plane marked off by lines drawn through the vertices of polygonal demand areas. This is particularly important in location problems, such as the maximin location problem, where the objective function is not convex. It means that the problem can be decomposed into a number of problems with linear objective functions [3,4,8,10,12].

Another characteristic of this problem in common with other rectilinear location problems (and block norms in general) is that it can be solved with linear programming; that is, optimizing $W(x)$ can be expressed as a set of linear programming formulations.

The rectangular distance function has discontinuities in its derivatives, so that associated location problems provide difficulties for gradient descent based procedures. One way around this is to use hyperbolic approximations for the terms represented by absolute differences [16,23].

See also

- ▶ Combinatorial Optimization Algorithms in Resource Allocation Problems
- ▶ Competitive Facility Location
- ▶ Facility Location with Externalities
- ▶ Facility Location Problems with Spatial Interaction
- ▶ Facility Location with Staircase Costs
- ▶ Global Optimization in Weber's Problem with Attraction and Repulsion
- ▶ MINLP: Application in Facility Location-allocation

- ▶ Multifacility and Restricted Location Problems
- ▶ Network Location: Covering Problems
- ▶ Production-Distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Stochastic Transportation and Location Problems
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Brimberg J, Love RF (1992) Local convergence in a generalized Fermat-Weber problem. Ann Oper Res 40:33–66
2. Brimberg J, Love RF (1993) Global convergence of a generalized iterative procedure for the minisum location problem with ℓ_p distances. Oper Res 41:1153–1163
3. Brimberg J, Mehrez A (1994) Multi-facility location using a maximin criterion and rectangular distances. Locat Sci 2:11–19
4. Brimberg J, Wesolowsky GO (2000) Note: Facility location with closest rectangular distances. Nav Res Logist 47:77–84
5. Brimberg J, Wesolowsky GO (2002) Minisum location with closest Euclidean distances. Ann Oper Res 111:151–165
6. Buchanan DJ, Wesolowsky GO (1993) Locating a noxious facility with respect to several polygonal regions using asymmetric distances. IIE Trans 25(1):77–88
7. Drezner Z (ed) (1995) Facility location: A survey of applications and methods. Springer, New York
8. Drezner Z, Wesolowsky GO (1983) The location of an obnoxious facility with rectangular distances. J Reg Sci 23:241–248
9. Francis RL, McGinnis LF Jr, White JA (1992) Facility layout and location: An analytical approach, 2nd edn. Internat Ser Industr and Systems Engin. Prentice-Hall, Englewood Cliffs
10. Hamacher HW, Nickel S (1995) Restricted planar location problems and applications. Nav Res Logist 42:967–992
11. Hausdorff F (1914) Grundzuge der Mengenlehre. reprinted (1949) by the Chelsea Publishing Company, New York
12. Juel H, Love RF (1976) An efficient computational procedure for solving multi-facility rectilinear facilities location problems. Oper Res Quart 26:697–703
13. Katz IN (1974) Local convergence in Fermat's problem. Math Program 6:89–104
14. Kuhn HW (1973) A note on Fermat's problem. Math Program 4:98–107
15. Love RF, Morris JG, Wesolowsky GO (1988) Facilities location: Models and methods. North Holland, New York
16. Morris JG (1981) Convergence of the Weiszfeld algorithm for Weber problems using a generalized distance function. Oper Res 29:37–48
17. Thisse JF, Ward JE, Wendell RE (1984) Some properties of location problems with block and round norms. Oper Res 32:1309–1327
18. Ward JE, Wendell RE (1980) A new norm for measuring distance which yields linear location problems. Oper Res 28:836–844
19. Ward JE, Wendell RE (1985) Using block norms for location modelling. Oper Res 33:1074–1090
20. Weiszfeld E (1937) Sur le point lequel la somme des distances de n points donnes est minimum. Tohoku Math J 43:355–386
21. Wesolowsky GO (1993) The Weber problem: Its history and perspectives. Locat Sci 1(1):5–23
22. Wesolowsky GO, Love RF (1971) Location of facilities with rectangular distances among point and area destinations. Nav Res Logist Quart 18(1):83–90
23. Wesolowsky GO, Love RF (1972) A nonlinear approximation method for solving a generalized rectangular distance Weber problem. Manag Sci 18:656–663

Order Complementarity

GEORGE ISAC
Royal Military College of Canada, Kingston, Canada

MSC2000: 90C33

Article Outline

Keywords
Preliminaries
Order Complementarity Problems
Order Complementarity Problem as Mathematical Model
  Order Complementarity Problem as Mathematical Model
  Global Reproduction of an Economic System Working with Several Technologies
  Discrete Dynamic Complementarity Problem
Solution Methods
See also
References

Keywords

Ordered vector spaces; Vector lattice; Order complementarity

The introduction of the order complementarity problem in complementarity theory can be justified by the following two reasons:

1) In the study of some particular classical complementarity problems the essential fact is not the orthogonality in the sense of an inner-product, but important is the lattice orthogonality. It is very useful, in some circumstances, to represent the classical complementarity problem as an order complementarity problem.
2) In several practical problems the complementarity condition appears with more than one operator. We have this situation, for example in Economics, in Lubrication Theory, in Stochastic Optimal Control Theory, etc. [16]. The study of order complementarity problems is a new chapter in Complementarity Theory. This chapter is now in developing.

Preliminaries

Denote by $E(\tau)$ (respectively, by $(E, \|\cdot\|)$ and $(E, \langle\cdot\rangle, \cdot)$) a locally convex space (respectively, a Banach space and a Hilbert space). Suppose that E is ordered by a *pointed convex cone* $\mathbf{K} \subset E$, i. e., $\mathbf{K}$ is a subset of E satisfying the following properties:

1) $\mathbf{K} + \mathbf{K} \subseteq \mathbf{K}$;
2) $\lambda\, \mathbf{K} \subseteq \mathbf{K}$ for all $\lambda \in \mathbf{R}_+$; and
3) $\mathbf{K} \cap (-\mathbf{K}) = \{0\}$.

Denote by $\leq$ the ordering defined by $\mathbf{K}$, that is, $x \leq y$ if and only if $y - x \in \mathbf{K}$. Assume that the ordered vector space $(E, \mathbf{K})$ is a *vector lattice*, i. e., for every pair $(x, y) \in E \times E$, the supremum $\vee (x, y)$ and the infimum $\wedge (x, y)$ with respect to the ordering $\leq$ exist in E. In this case, for every $x, y, z \in E$ we have:

1) $\vee (x, y) + z = \vee (x + z, y + z)$;
2) $\wedge (x, y) + z = \wedge (x + z, y + z)$;
3) $\vee (x \vee y, z) = \vee [\vee (x, y), \vee (y, z)] = \vee (x, y, z)$.

We can show that $\vee (x, y) = - \wedge (-x, -y)$ for all $x, y \in E$.

Let $E(\tau)$ be a locally convex space and $\mathbf{K} \subset E$ a closed convex cone.

We say that $\mathbf{K}$ is *regular* (respectively, *completely regular*) if all monotone increasing and order bounded (resp. topological bounded) sequences of elements of $\mathbf{K}$ are τ-convergent. Let $D \subset E$ be a subset. An operator $T: D \to E$ is said to be *isotone* (respectively, *antitone*) if $x_1 \leq x_2$ $(x_1, x_2 \in D)$ implies $T(x_1) \leq T(x_2)$ (respectively, $T(x_2) \leq T(x_1)$). If S is a subset of D, we denote by $\mu(S)$ the *measure of noncompactness* of S i. e., $\mu(S) = \inf\{r > 0$: S can be recovered by a finite family of subsets of E whose diameter $< r\}$.

We say that T is a *k-set-contraction* $(k \geq 0)$ if it is continuous, bounded and $\mu(T(S)) \leq k\mu(S)$ for any bounded set $S \subset D$. A k-set-contraction is called a *strict-set-contraction* if $k < 1$. T is called *condensing* if it is continuous, bounded and $\mu(T(S)) < \mu(S)$ for any bounded set $S \subset D$ with $\mu(S) > 0$. If $\mathbf{K} \subset E$ is a pointed convex cone we say that a bilinear form $\langle\cdot, \cdot\rangle$ on E is $\mathbf{K}$-*local* if and only if $\langle x, y\rangle = 0$, whenever $x, y \in \mathbf{K}$ and $\wedge (x, y) = 0$.

Order Complementarity Problems

The order complementarity problems represent a relatively new chapter in complementarity theory. The order complementarity problems are necessary since, in many situations some classical complementarity problems must be represented as a lattice orthogonality problem. Furthermore in some practical problems, we must use the complementarity condition simultaneously with respect to several operators.

Let $(E, \mathbf{K})$ be a vector lattice with respect to the order defined by the pointed convex cone $\mathbf{K}$. Let D be a nonempty subset of E. In particular, the set D can be the cone $\mathbf{K}$. Given m linear or nonlinear functions $f_1, \ldots, f_m: E \to E$, the *order complementarity problem* associated with the family of functions and with the set D is:

$$\mathrm{OCP}(\{f_i\}_{i=1}^n; D) \begin{cases} \text{find} & x_0 \in D \\ \text{s.t.} & \wedge(f_1(x_0), \ldots, f_m(x_0)) \\ & \quad = 0. \end{cases}$$

In [16] this problem is named the *implicit general order complementarity problem*. We have several interesting particular cases:

1) If $m = 2$, $D = E$, $f_1 = \mathrm{Id}$ (the identity mapping) and $f_2(x) = Tx + q$, where $T: E \to E$ is a linear mapping and q an element in E, we have the *linear order complementarity problem* denoted by $\mathrm{LOCP}(T, q)$. This problem was studied systematically for the first time in 1989 in [2] where several interesting new classes of linear operators were introduced. We find, for example, the operators of classes (H^+), (S), (Z), (K), (P) and (A).

2) If m is arbitrary and the functions f_i ($i = 1, \ldots, m$) are affine mappings we have the *generalized linear order complementarity problem*. Several results about this problem are obtained in [9,16,25].
3) If $m = 2$, $D = \mathbf{K}$ and f_1, f_2 are nonlinear mappings we have the *nonlinear order complementarity problem* studied for the first time in 1986, in [12].
4) If $m = 3$, $D = E$, $f_1 = \mathrm{Id}$ and f_2, f_3 are nonlinear we have the order complementarity problem introduced in lubrication theory in 1986 in [22].
5) If m is arbitrary, $D = \mathbf{K}$, $f_1 = \mathrm{Id}$ and $f_2, \ldots, f_m$ are nonlinear but having the form $f_i(x) = x - T_i(x)$ ($i = 1, \ldots, m$), with every T_i a nonlinear mapping, we have the *generalized order complementarity problem* studied systematically in [17], and for set-valued mappings in [18].

Order Complementarity Problem as Mathematical Model

The order complementarity problems can be considered also as mathematical models for many practical problems. We indicate in this section some of such models.

Mixed Lubrication Problem

Consider the mixed lubrication in the context of a journal bearing with elastic support. The problem is to study the contact pressure X. In this case, $E = H^1(\Omega)$ (defined over $L^2(\Omega)$) and the cone is $\mathbf{K} = \{\mu \in H^1(\Omega)$: $\mu \geq 0$ a.e. on $\Omega\}$. We have two operators, T_1 (X) and $T_2(X)$, where T_1 is generally an integral operator and T_2 is the Reynolds' partial differential operator. For the definition of these operators, the reader is referred to [16,17,22]. In this case, there are three distinct functions which cause the decomposition of the spatial area into three disjoint regions: the innermost region (solid-to-solid contact), the elasto-hydrodynamic lubrication region (solid-to-fluid contact), and the cavity region (in which the pressure returns to the ambient value). The complementarity formulation is based on the observation that the contact pressure X satisfies the following equations specified for every region:

1) $X \geq 0$, $T_1(X) = 0$, $T_2(X) \geq 0$ (solid-to-solid contact);
2) $X = 0$, $T_1(x) \geq 0$, $T_2(X) \geq 0$ (cavity point);
3) $X \geq 0$, $T_1(X) \geq 0$, $T_2(X) = 0$ (lubrication point).

The problem to know the contact pressure X is equivalent to the solvability of the problem OCP(Id, T_1, T_2;$\mathbf{K}$). Initially, this problem was defined in [22] and until now it is not solved.

Global Reproduction of an Economic System Working with Several Technologies

Consider a nonlinear economic system which is a generalization of the classical linear input-output system defined by Leontief. Suppose that the system has n production sectors and every sector works with m technologies to produce one type of output. The number of technologies is the same for every sector. Every sector is constrained to use the production of the others. Let x_j be the level in units of the gross activity performed in the sector j. Suppose that to produce x_j units in the sector j, $f_{ij}^k(x_j)$ units from the technology k of sector i are needed as inputs. We make the following assumptions: for all i, j, k:

1) all $f_{ij}^k(x_j)$ are continuous;
2) $f_{ij}^k(0) = 0$;
3) $0 \leq u_j \leq v_j$ implies $f_{ij}^k(u_j) \leq f_{ij}^k(v_j)$.

The balances between total activities and final demands for the technology k are given by $x_i = \sum_{j=1}^n f_{ij}^k(x_j) + y_j$, $i = 1, \ldots, n$, where y_i is the final demand for the sector i. Denote this system by $S(\{f_{ij}^k\})$. This is the classical Leontief nonlinear input-output system studied by several authors (see the references cited in [17]). We replace condition 3) by the following more realistic condition:

4) there exists a continuous mapping $\Phi: \mathbf{R}^n \to \mathbf{R}^n$ such that:
   i) $\mathrm{Id} + \Phi$ is invertible and $(\mathrm{Id} + \Phi)^{-1}$ is isotone (with respect to the ordering defined by $\mathbf{R}_+^n$);
   ii) $\Phi(x) + \sum_{j=1}^n f_j^k(x_j)$ is isotone for every k, where $x = (x_1, \ldots, x_n)^\intercal$ and $f_j^k(x_j) = [f_{ij}^k(x_j)]_{i=1}^n$.

If $S(\{f_{ij}^k(x_j)\})$ satisfies 1), 2) and 4) we say that it is a *tolerant system* and in this case Φ is a *tolerance*. We define $F^k(x) = x - \sum_{j=1}^n f_j^k(x_j)$ and for any $y^0 > 0$,

$$S_{y^0} = \left\{ x \in \mathbf{R}_+^n : \begin{array}{c} F^1(x) - y^0 \geq 0 \\ \vdots \\ F^m(x) - y^0 \geq 0 \end{array} \right\}.$$

For this model, the problem is to show that given $y^0 > 0$ with S_{y^0} nonempty, the problem OCP($T_1, \ldots, T_m, \mathbf{R}_+^n$)

has a solution $x^0 > 0$ which is the least element of S_{y^0}, where

$$T_1(x) = F^1(x) - y^0, \ldots, T_m(x) = F^m(x) - y^0 .$$

In this case we say that the *production* x^0 *is realizing* y^0 *with a minimal social cost*. This model was studied in [17]. In this paper it is shown that a tolerant economic system, which is locally nonlinear and with the functions f_{ij}^k not necessarily isotone, has a behavior similar to a classical Leontief model.

Discrete Dynamic Complementarity Problem

One of the recent discoveries in complementarity theory is the *dynamic complementarity problem*. It seems that this problem was defined in [10] and it is a unifying framework for fluid and diffusion approximation of stochastic flow networks. Now we consider the *discrete dynamic complementarity problem* (DDCP). Let $(\mathbf{R}^m, \langle \cdot, \cdot \rangle)$ be the Euclidean space ordered by $\mathbf{R}^m_+$. Let $x = \{x(0), \ldots, x(n), \ldots\}$ be a sequence of vectors in $\mathbf{R}^m$. Assume that $x(0) \geq 0$ and $\mathcal{R}$ is a real $(m \times m)$-matrix. The problem (DDCP) is the following: given the sequence x and the matrix $\mathcal{R}$ find the sequence $y = \{y(0), \ldots, y(n), \ldots\}$ such that for all $n \in \mathbf{N}$:

i) $z(n) = x(n) + \mathcal{R}y(n) \geq 0$;
ii) $y(0) = 0$, $\Delta\, y(n) = y(n) - y(n-1) \geq 0$;
iii) $\langle z(n), \Delta\, y(n) \rangle = 0$.

We assume by convention that $y(-1) = 0$ and $\mathbf{N} = \{0, 1, 2, \ldots\}$. Consider the vector space $S = \{x\colon x\colon \mathbf{N} \to \mathbf{R}^m$, ordered by the convex cone

$$\mathbf{K} = \{x \in S\colon\ x(n) \geq 0 \text{ for all } n \in \mathbf{N}\}$$

and endowed with the Fréchet locally convex topology defined by the family of seminorms $\{p_r\}_{r \in \mathbf{N}}$, where $p_r = \sum_{n=0}^{r} \| x(n) \|$. The space S is a vector lattice and $\mathbf{K}$ is normal. We define the following operators from S into S:

$$T_1(y) = \{x(0) + \mathcal{R}y(0), \ldots, x(n) + \mathcal{R}y(n), \ldots\} ,$$
$$T_2(y) = \{0, y(1) - y(0), \ldots, y(n) - y(n-1), \ldots\} ,$$

We put $A = \{y \in S\colon y(0) = 0\}$. The solvability of problem DDCP is equivalent to the solvability of the problem OCP(T_1, T_2;A). The study of dynamic complementarity problem is an interesting new research domain in Complementarity Theory. The reader can find other examples of order complementarity problems in [14,16], where it is shown that the generalized linear complementarity problem in Cottle and Dantzig's sense or the Bellman routing problem can be reformulated as on order complementarity problem.

Solution Methods

Let $E(\tau)$ be a locally convex space ordered by a closed pointed convex cone $\mathbf{K} \subset E$ suppose that E is a vector lattice. Given the operators $T_1, \ldots, T_m$ E { E and the set $D \subset E$, we consider the problem OCP($\{T_i\}_{i=1}^m$, D). Because the fact that the operator '$\wedge$' is used in the definition of this problem, many classical methods applicable to the solvability of nonlinear equations or to the solvability of fixed point problems are not applicable. For example the operator '$\wedge$' can distroy the compactness or the differentiability of operators T_i ($i = 1, \ldots, m$). However, some fixed point methods or some topological methods are applicable. In this sense several existence theorems and iterative methods for solvability of the problem OCP($\{T_i\}_{i=1}^m$, D) are presented in the papers [12,13,14,15,16,17,18]. Several existence results can be obtained using the fixed point theory and the following result.

Theorem 1 *If Φ_0: $E \to E$ is an arbitrary function such that $(Id + \Phi_0)^{-1}$ exists, then $x_* \in D$ is a solution of the problem OCP($\{T_i\}_{i=1}^m$, D) if and only if x_* is a fixed point of the mapping $\mathcal{H}(x)$ defined by*

$$\mathcal{H}(x) = (\mathrm{Id} + \Phi_0)^{-1} \times (\vee\{(\mathrm{Id} + \Phi_0 - T_1)(x), \ldots, (\mathrm{Id} + \Phi_0 - T_m)(x)\})$$

for every $x \in E$.

The importance of this Theorem is the fact that for practical problems we can choose the mapping Φ_0 such that $\mathcal{H}(x)$ has some good properties. We denote $H(x) = \wedge\,(x - T_1(x), \ldots, x - T_m(x))$, for all $x \in E$.

Definition 2 We say that H is Φ_0-*isotone* on D if there exists a mapping Φ: $E \to E$ such that $H + \Phi$ is isotone on D, $(\mathrm{Id} + \Phi)$ is invertible and $(\mathrm{Id} + \Phi)^{-1}$ is isotone.

We recall that D is order convex if $u, v \in D$ and $u \leq w \leq v$ imply that $w \in D$.

Theorem 3 *Let $(E(\tau), \mathbf{K})$ be a locally convex vector lattice and $D \subset E$ an ordered convex set. If the following assumptions are satisfied:*

1) $\mathbf{K}$ is a regular cone;
2) H is Φ-isotone;
3) $(Id + \Phi)^{-1}(H + \Phi)$ is continuous;
4) there exist $x_0, y_0 \in D$ such that $x_0 \leq y_0$, $x_0 \leq H(x_0)$ and $H(y_0) \leq y_0$,
then the problem has a minimal and a maximal solutions, both computable by iterative methods.

Proof The proof is in [16].

Suppose that $\mathrm{Id} - T_i = R_i + S_i$, where R_i is isotone and S_i antitone. In this case we can associate to the mapping H the mapping

$$\widehat{H}(x, y) = \wedge\{R_1(x) + S_1(y), \ldots, R_m(x) + S_m(y)\}$$

for all $x, y \in E$. We obtain that H is a *heterotonic operator* in Opoitsev's sense (see [23]). We say that (x_*, y_*) is a *coupled fixed point* for H if $\widehat{H}(x_*, y_*) = x_*$ and $\widehat{H}(y_*, x_*) = y_*$. We have the following result:

Theorem 4 *Let $(E, \|\cdot\|)$ be a uniformly convex Banach space ordered by a regular pointed closed convex cone $\mathbf{K} \subset E$. If one of the following assumptions are satisfied:*
1) $\widehat{H}$ is nonexpansive;
2) $\widehat{H}$ is condensing with respect to a measure of noncompactness;
3) $\widehat{H}$ is continuous and $\dim E < +\infty$,
then for very conical interval $[x_0, y_0]$ strongly invariant for H (i. e., $x_0 \leq \widehat{H}(x_0, y_0)$ and $\widehat{H}(y_0, x_0) \leq y_0$), there exists a coupled fixed point (x_*, y_*) for H and a solution $\widehat{x}$ of the problem OCP($\{T_i\}_{i=1}^m$, $\mathbf{K}$) such that $x_* \leq \widehat{x} \leq y_*$. Moreover $x_* = \lim_{k\to\infty} x_k$ and $y_* = \lim_{k\to\infty} y_k$, where $x_k = \widehat{H}(x_{k-1}, y_{k-1})$ and $y_k = \widehat{H}(y_{k-1}, x_{k-1})$.

Proof The proof of this theorem is in [16].

Several interesting existence results based on a special topological index, defined on cones, are presented in [15]. This topological index was defined in [23]. The problem OCP($\{T_i\}_{i=1}^m$, D) can be also studied by the topological degree. Several results in this sense have been obtained especially for the Linear Order Complementarity Problem [8,9,25]. The results obtained for the problem OCP($\{T_i\}_{i=1}^m$, D) can be applied to the problem NCP(f, $\mathbf{K}$) when E is a Hilbert space and the inner product is $\mathbf{K}$-local, since in this case the problem NCP(f, $\mathbf{K}$) is equivalent to the problem OCP(Id, f, $\mathbf{K}$).

See also

- Convex-Simplex Algorithm
- Equivalence Between Nonlinear Complementarity Problem and Fixed Point Problem
- Generalized Nonlinear Complementarity Problem
- Integer Linear Complementary Problem
- LCP: Pardalos–Rosen Mixed Integer Formulation
- Lemke Method
- Linear Complementarity Problem
- Linear Programming
- Parametric Linear Programming: Cost Simplex Algorithm
- Principal Pivoting Methods for Linear Complementarity Problems
- Sequential Simplex Method
- Topological Methods in Complementarity Theory

References

1. Bellman R (1958) On a routing problem. Quart Appl Math 16:87–90
2. Borwein JM, Dempster MAH (1989) The linear order complementarity problem. Math Oper Res 14:534–558
3. Carbone A, Isac G (1998) The generalized order complementarity problem: Applications to economics and an existence result. Nonlinear Stud 5(2):129–151
4. Cottle RW, Dantzig GB (1970) A generalization of the linear complementarity problem. J Combin Th 8:79–90
5. Ebiefung AA (1991) The generalized linear complementarity problem and its applications. PhD Thesis, Clemson University
6. Ebiefung AA (1995) Nonlinear mappings associated with the generalized linear complementarity problem. Math Program 69:225–268
7. Ebiefung AA, Kostreva M (1993) The generalized Leontief input-output model and its application to the choice of new technology. Ann Oper Res 44:161–172
8. Gowda MS (1993) Applications of degree theory to linear complementarity problems. Math Oper Res 18:868–879
9. Gowda MS, Sznajder R (1994) The generalized order linear complementarity problem. SIAM J Matrix Anal Appl 15(3):779–795
10. Harrison JM, Reiman MI (1981) Reflected Brownian motion on an orthant. Ann of Probab 9:302–308
11. Hyers DH, Isac G, Rassias TM (1997) Topics in nonlinear analysis and applications. World Sci., Singapore
12. Isac G (1986) Complementarity problem and coincidence equations on convex cones. Boll Unione Mat Ital Ser B 6:925–943
13. Isac G (1992) Complementarity problems. Lecture Notes Math, vol 1528. Springer, Berlin

14. Isac G (1992) Iterative methods for the general order complementarity problem. In: Singh SP (ed) Approximation Theory, Spline Functions and Applications. Kluwer, Dordrecht, pp 365–380
15. Isac G (1996) The fold complementarity problem and the order complementarity problem. Topol Meth Nonlinear Anal 8:343–358
16. Isac G, Goeleven D (1993) The implicit general order complementarity problem: models and iterative methods. Ann Oper Res 44:63–92
17. Isac G, Kostreva M (1991) The generalized order complementarity problem. J Optim Th Appl 71(3):517–534
18. Isac G, Kostreva M (1991) Kneser's theorem and the multivalued generalized order complementarity. Appl Math Lett 4(6):81–85
19. Isac G, Kostreva M (1996) The implicit generalized order complementarity problem and Leontief's input-output model. Appl Math 24(2):113–125
20. Mandelbaum A (1990) The dynamic complementarity problem. Working Paper, Stanford Business School
21. Oh KP (1984) The numerical solution of dynamically loaded elasto-hydro-dynamic contact as a nonlinear complementarity problem. J Tribology 106:88–95
22. Oh KP (1986) The formulation of the mixed lubrication problem as a generalized nonlinear complementarity problem. J Tribology 108:598604
23. Opoitsev VI (1979) a generalization of the theory of monotone and concave operators. Trans Moscow, Math Soc 2:243–279
24. Szand BP (1989) The generalized complementarity problem. PhD Thesis, Rensselaer Polytech. Institute, Troy, New York
25. Sznajder R (1994) Degree theoretic analysis of the vertical and horizontal linear complementarity problem. PhD Thesis, University Maryland

Oriented Matroids

PAOLA FESTA
Dip. Mat. e Inform., University Salerno, Baronissi, Italy

MSC2000: 90C09, 90C10

Article Outline

Keywords
Historical Review
Axiom Systems for Oriented Matroids.
  Circuits and Circuit Axioms.
  Minors
  Duality.
  Chirotopes and Basis Orientations
  Vectors and Covectors
General Topics
  Directed Graphs
  Real Vectors Spaces
  Vector Configurations
  Point Configurations
  Hyperplane Arrangements.
  Topological Representation Theorem
Conclusions
See also
References

Keywords

Combinatorial optimization; Greedy technique; Graph optimization

Matroids have been defined in 1935 as generalization of graphs and matrices.

Starting from the 1950s they have had increasing interest and the theoretical results obtained have been used for solving several difficult problems in various fields such as civil, electrical, and mechanical engineering, computer science, and mathematics. *Oriented matroids* are a special class of matroids. They can be viewed as a combinatorial abstraction of real hyperplanes arrangements, of point configurations over the reals, of convex polytopes, or of directed graphs. Scope of this article is to introduce the reader to the theory of oriented matroids, providing an extensive discussion of the axiom systems for them and illustrating the different aspects that characterize these objects.

Historical Review

In 1935 H. Whitney in [35] studied the *linear dependence* and its important application in mathematics. A number of equivalent axiomatic systems for matroids is contained in his pioneering paper, that is considered the first scientific work about matroid theory.

In the 1950s and 1960s, starting from the Whitney's ideas, W. Tutte in [22,23,24,25,26,27,28,29,30] built a considerable body of theory about the structural properties of matroids, which became popular in the 1960s, when J. Edmonds in [7,8,9,10,11,12,13] introduced the matroid theory in combinatorial optimization. From 1965 on, a growing number of researchers

became interested in matroids. The origins of oriented matroids go back to these years due to R.G. Bland, J. Folkman, M. Las Vergnas, and Lawrence. Among the first results, see [17] and [15], which contains the fundamental topological representation theorem for oriented matroids. To the same years further notions of oriented matroids go back due to Bland, who was motivated by linear programming duality theory ([3,4]), and, independently, to Las Vergnas, who was motivated by graph theory ([31,32,33,34]). In 1978 appeared [5].

In the literature there are several other early papers, among them [16,19], and [21].

[2] contains a comprehensive and tutorial treatment of oriented matroids, whose current research updates are provided in [36].

Axiom Systems for Oriented Matroids.

Researchers from various mathematical areas arrived at four basic, equivalent, axiom systems:

1) circuit axioms;
2) orthogonality axioms;
3) chirotopes, or basis orientations;
4) vector axioms.

In order to understand these axiom systems, some notions and definition are needed.

Definition 1 A *signed set* X is a set $\underline{X}$ together with a partition (X^+, X^-) of $\underline{X}$. X^+ is the positive set; X^- is the negative set. X is positive (negative), if $X^+ = \emptyset$ ($X^- = \emptyset$). $\underline{X} = X^+ \cup X^-$ is the support of X.

Definition 2 Let X and Y be two signed sets, X is a restriction of Y if and only if $X^+ \subseteq Y^+$ and $X^- \subseteq Y^-$.

Definition 3 Let F be an unsigned set and X be a signed set. Then, the restriction of X to F, denoted $X|_F$, is a signed set $Y = X \cup F$, such that $Y^+ = X^+ \cup F$ and $Y^- = X^- \cup F$.

Definition 4 A signed set X can be defined also throughout a mapping $\mathrm{sg}_X: X \to \{-1, 1\}$, such that $X^+ = \{x: \mathrm{sg}_X(x) = 1\}$ and $X^- = \{x: \mathrm{sg}_X(x) = -1\}$. sg_X is called the *signature* of X.

In the following, $X \setminus Y$ denotes the restriction of X to $\underline{X} \setminus \underline{Y}$.

Definition 5 Let X and Y be two signed sets, their *composition* $X \circ Y$ is a signed set such that $(X \circ Y)^+ = X^+ \cup (Y^+ \setminus X^-)$ and $(X \circ Y)^- = X^- \cup (Y^- \setminus X^+)$.

Note that $\circ$ is associative, while $X \circ Y = Y \circ X$ if and only if the restrictions of X and Y to their intersection are equal.

Definition 6 The *opposite of a signed set* X, denoted $-X$, is the signed set such that $(-X)^+ = X^-$ and $(-X)^- = X^+$.

Definition 7 Let X and Y be two signed sets. X and Y are called *orthogonal signed sets*, denoted by $X \perp Y$, if either $X \cap Y = \emptyset$, or the restrictions of X and Y to their intersection are neither equal nor opposite, i. e. there must exist $x, y \in X \cap Y$ such that $X(x)Y(x) = -X(y)Y(y)$.

Definition 8 Let E be any set. A *signed subset* of E is a signed set whose support is contained in E.

A signed subset of E can be identified with an element of $\{-1, 0, 1\}^E$, which is usually abbreviated by $\{-, 0, +\}^E$. If $E = \{1, \ldots, n\}$ and $X \in \{-, 0, +\}^E$, X is a *sign vector* having n entries $-$, 0, or $+$.

Circuits and Circuit Axioms.

In this Section we provide the definition of an oriented matroid in terms of its signed circuits.

Definition 9 (Circuit axioms) A collection $\mathcal{C}$ of signed subsets of a set E is the set of *signed circuits* of an oriented matroid $\mathcal{M}$ on E if and only if $\mathcal{C}$ satisfies the following axioms:

1) $\emptyset \notin \mathcal{C}$;
2) symmetry: $\mathcal{C} = -\mathcal{C}$;
3) incomparability: $\forall X, Y \in \mathcal{C}$, if $\underline{X} \subseteq \underline{Y}$, then $X = Y$ or $X = -Y$;
4) weak elimination: $\forall X, Y \in \mathcal{C}, X \neq -Y$, and $x \in X^+ \cap Y^-$ there exists a $Z \in \mathcal{C}$ such that $Z^+ \subseteq (X^+ \cup Y^+) \setminus \{x\}$ and $Z^- \subseteq (X^- \cup Y^-) \setminus \{x\}$.

Note that the Axioms 1), 3), and 4) of Definition 9 are the circuit axioms of an ordinary matroid.

Corollary 10 *The circuit supports* $\underline{\mathcal{C}} = \{\underline{X}: X \in \mathcal{C}\}$ *in an oriented matroid* $\mathcal{M}$ *are the circuits of a matroid* $\underline{\mathcal{M}}$, *called underlying matroid of* $\mathcal{M}$.

Definition 11 C is called a *circuit orientation* of $\underline{\mathcal{M}}$, which has the same rank of $\mathcal{M}$.

An ordinary matroid M is *orientable* if it has a circuit orientation.

Deciding the orientability of a matroid is a difficult problem. Bland and Las Vergnas [5] proved that a binary matroid is orientable if and only if it is regular.

Definition 12 Let $\mathcal{M}$ be an oriented matroid on a set E and let $x \in E$. Then x is a loop of $\mathcal{M}$ if $(\{x\}, \emptyset) \in \mathcal{C}$.

If $x \notin X$ for every $X \in \mathcal{C}$, x is called *coloop* of $\mathcal{M}$.

Bland and Las Vergnas [5] and, independently, Folkman and Lawrence [15] obtained the following result.

Theorem 13 *Let $\mathcal{C}$ be a collection of signed subsets of a set E satisfying 1), 2), and 3) of Definition 9. Then 4) of Definition 9 is equivalent to*

4') strong elimination: $\forall\ X, Y \in \mathcal{C}$, $x \in X^+ \cap Y^-$, and $f \in (X^+ \setminus Y^-) \cup (X^- \setminus Y^+)$, there is a $Z \in \mathcal{C}$ such that $f \in Z$, $Z^+ \subseteq (X^+ \cup Y^+) \setminus \{x\}$, and $Z^- \subseteq (X^- \cup Y^-) \setminus \{x\}$.

Minors

As for the ordinary case also for an oriented matroid $\mathcal{M}$ on a set E, it is possible to define *submatroids* or *minors* induced by a subset F of E by *deletion* and/or *contraction*.

For any $\mathcal{S} \subseteq \{-, 0, +\}^E$, let Min$\mathcal{S}$ be the collection of nonempty signed sets in $\mathcal{S}$ with inclusion-minimal support and let Max$\mathcal{S}$ be the collection of signed sets in $\mathcal{S}$ with inclusion-maximal supports. Then the following properties hold.

Proposition 14 (Deletion) *Let $\mathcal{M}$ be an oriented matroid on a set E with set of signed circuits $\mathcal{C}$, and let $F \subseteq E$. Then $C' = \{X \in \mathcal{C}: \underline{X} \subseteq F\}$ is the set of circuits of an oriented matroid on F called the submatroid of $\mathcal{M}$ induced on F and denoted by $\mathcal{M}(F)$.*

Proposition 15 (Contraction) *Let $\mathcal{M}$ be an oriented matroid on a set E with set of signed circuits $\mathcal{C}$, and let $F \subseteq E$. Then Min$\{X|_F: X \in \mathcal{C}\}$ is the set of circuits of an oriented matroid on F called the contraction of $\mathcal{M}$ to F and denoted by $\mathcal{M}/A$, where $A = E \setminus F$.*

Proposition 16 *Let $\mathcal{M}$ be an oriented matroid on a set E, and let A, B be two disjoint subsets of E. It holds*

$$(\mathcal{M} \setminus A) \setminus B = \mathcal{M} \setminus (A \cup B)\,,$$

$$\frac{(\frac{\mathcal{M}}{A})}{B} = \frac{\mathcal{M}}{(A \cup B)}\,,$$

$$\frac{(\mathcal{M} \setminus A)}{B} = (\frac{\mathcal{M}}{B}) \setminus A\,.$$

Definition 17 A circuit signature of a matroid assigns to each circuit C two opposite signed sets X and $-X$ supported by C.

Theorem 18 *Let M be a matroid on a set E, and let $\mathcal{S}$ be a circuit signature of M. Suppose that for all $x \in E$ the induced circuit signatures $\mathcal{S} \setminus \{x\}$ and $\mathcal{S}/\{x\}$ are circuit orientations of $M \setminus \{x\}$ and $M/\{x\}$, respectively. Then, one of the following condition holds:*

1) $\mathcal{S}$ is a circuit orientation of $\mathcal{M}$;
2) $|E| = 3$;
3) $|E| = 4$.

Duality.

Let $\mathcal{M}$ be an oriented matroid on a set E, B a basis of $\underline{\mathcal{M}}$, and $x \in E \setminus B$, then there is a unique circuit $c(x, B)$ of $\underline{\mathcal{M}}$. $\underline{c}(x, B)$ is contained in $B \cup \{x\}$ and supports a unique signed circuit of $\mathcal{M}$. Let $c(x, B)$ be the basic circuit of x with respect to B, the signed circuit supported by $\underline{c}(x, B)$ which has x in its positive part. On the other hand, given $x \in B$, $\underline{c}^*(x, B)$ denotes a unique cocircuit of $\underline{\mathcal{M}}$ disjoint from $B \setminus \{x\}$. $c^*(x, B)$ denotes the corresponding signed cocircuit of $\mathcal{M}$ which is positive on $\{x\}$ with respect to B. The existence and uniqueness of $c^*(x, B)$ is proved in the following Proposition.

Proposition 19 *Let $\mathcal{M}$ be an oriented matroid on a set E with set of circuits C. Then the following three properties hold.*

1) There exist a unique signature C^* of the cocircuits of $\underline{\mathcal{M}}$ such that $X \perp Y$, for all $X \in C$ and $Y \in C^*$.
2) The collection C^* is the set of signed circuits of an oriented matroid on E called the dual matroid (or orthogonal matroid) of $\mathcal{M}$, denoted by $\mathcal{M}^*$.
3) $\mathcal{M}^{**} = \mathcal{M}$.

The following Theorem, proved in [5], states the axiomatic definition of an oriented matroid in terms of its dual.

Theorem 20 (Orthogonality axioms) *Let M be a matroid, C be a circuit signature of M, and C^* be a cocircuit signature of M. Then the following properties are equivalent.*

1) C and C^* are the circuit collections of a pair of dual matroids.
2) $X \perp Y$, $\forall X \in C$, $\forall Y \in C^*$.
3) $X \perp Y$, $\forall X \in C$, $\forall Y \in C^*$, with $|\underline{X} \cap \underline{Y}| \leq 3$.

Bland and Las Vergnas have given [5] and [6] a further axiom system, called *painting axioms* by them, expressed in the following theorem.

Theorem 21 (Painting axioms) *Let C and C^* be two collections of signed subsets of a set E. Then, C is the set of circuits and C^* is the set of cocircuit of an oriented matroid on E if and only if they satisfy the conditions 1)–3) in Definition 9 and one of the two following equivalent properties.*

1) *3-painting: For all 3-partitions $E = B \cup G \cup R$ and $x \in B$ either there exists an $X \in C$ such that $x \in \underline{X} \subseteq B \cup G$ and $\underline{X} \cap B \subseteq X^+$, or there exists a $Y \in C^*$ such that $x \in \underline{Y} \subseteq B \cup R$ and $\underline{Y} \cap B \subseteq Y^+$, but not both.*
2) *4-painting: For all 4-partitions $E = B \cup W \cup G \cup R$ and $x \in B \cup W$ either there exists an $X \in C$ such that $x \in \underline{X} \subseteq B \cup W \cup G$, $\underline{X} \cap B \subseteq X^+$, and $\underline{X} \cap W \subseteq X^-$, or there exists a $Y \in C^*$ such that $x \in \underline{Y} \subseteq B \cup W \cup R$, $\underline{Y} \cap B \subseteq Y^+$, and $\underline{Y} \cap W \subseteq Y^-$, but not both.*

Corollary 22 *Each element of an oriented matroid belongs either to a positive circuit or to a positive cocircuit, but not to both.*

The previous Corollary is helpful to define a special type of oriented matroid, called *acyclic oriented matroid*.

Definition 23 An oriented matroid $\mathcal{M} = (E, C)$ is *acyclic* if it does not contain a positive circuit.

$\mathcal{M}$ is *totally acyclic* if all its elements are contained in a positive circuit.

The identification of the minors of $\mathcal{M}^*$ with the duals of the minors of $\mathcal{M}$ follows from Theorem 20, as showed in the following proposition.

Proposition 24 *Let $\mathcal{M}$ be an oriented matroid on a set E and A be a subset of E. Then,*

$$(\mathcal{M} \setminus A)^* = \mathcal{M}^*/A\,,$$
$$(\mathcal{M}/A)^* = \mathcal{M}^* \setminus A\,.$$

Chirotopes and Basis Orientations

An oriented matroid can be also defined by giving a sign to its bases. In the following are reported some definitions, needed to understand how to construct a basis orientation of a given oriented matroid, which becomes characterized in terms of signed bases, as shown in [32] and [34].

Definition 25 The basis signature of an oriented matroid is called *chirotope*.

Definition 26 Let $\mathcal{M}$ be an oriented matroid on a set E with signed circuits C. The *bases* of $\mathcal{M}$ are those maximal subsets of E that do not contain any circuit, i. e. they are the bases of $\underline{\mathcal{M}}$.

Definition 27 A *basis orientation of an oriented matroid* $\mathcal{M}$ is a mapping χ of the set of ordered bases of $\mathcal{M}$ to $\{-1, 1\}$ such that

1) χ is alternating;
2) for each two ordered bases of $\mathcal{M}$ of the form $(a, x_2, \ldots, x_r)$ and $(b, x_2, \ldots, x_r)$, $a \neq b$, it holds that

$$\chi(b, x_2, \ldots, x_r) = -C(a)C(b)\chi(a, x_2, \ldots, x_r)\,,$$

where C is one of the two opposite signed circuits of $\mathcal{M}$ in the set $(a, b, x_2, \ldots, x_r)$.

Condition 2) in Definition 27 is also known as *pivoting property*.

Las Vergnas [32] and [34] has proved that every oriented matroid $\mathcal{M}$ has exactly two basis orientations, which are opposite. He also showed that if χ is a basis orientation of $\mathcal{M}$, then $\mathcal{M}$ is uniquely determined by $\underline{\mathcal{M}}$ and χ.

The pivoting property 2) in Definition 27 can be rewritten also as follows.

Definition 28 A *basis orientation* χ of an oriented matroid $\mathcal{M}$ is such that

2*) for each two ordered bases of $\mathcal{M}$ of the form $(a, x_2, \ldots, x_r)$ and $(b, x_2, \ldots, x_r)$, $a \neq b$, it holds that

$$\chi(b, x_2, \ldots, x_r) = D(a)D(b)\chi(a, x_2, \ldots, x_r)\,,$$

where D is one of the two opposite signed cocircuits complementary to the hyperplane spanned by $\{x_2, \ldots, x_r\}$ in $\mathcal{M}$.

Conditions 2 and 2* of Definitions 27 and 28, respectively, are equivalent for a map χ, if $\mathcal{M}$ is an oriented matroid.

Definition 29 (Chirotope axioms) Let E be a finite set and let $r \geq 1$ be an integer. A *chirotope* of rank r is a mapping $\chi: E^r \to \{-1, 0, 1\}$ satisfying the following three properties:

1) χ is not identically zero;

2) χ is alternating, i. e.

$$\chi(x_{\sigma_1}, \ldots, x_{\sigma_r}) = \operatorname{sign}(\sigma)\chi(x_1, \ldots, x_r) ,$$

for each permutation σ and each $x_1, \ldots, x_r \in E$;

3) for all $x_1, \ldots, x_r, y_1, \ldots, y_r \in E$ such that for $i = 1, \ldots, r$, $\chi_1 \cdot \chi_2 \geq 0$, where

$$\chi_1 = \chi(y_i, x_2, \ldots, x_r) ,$$
$$\chi_2 = \chi(y_1, \ldots, y_{i-1}, x_1, y_{i+1}, \ldots, y_r) ,$$

it holds that

$$\chi(x_1, \ldots, x_r) \cdot \chi(y_1, \ldots, y_r) \geq 0 .$$

J. Lawrence [18] proved the following result.

Theorem 30 *Let E be a set and let $r \geq 1$ be an integer. Then, a mapping χ: E^r: {−1, 0, 1} is a basis orientation of an oriented matroid of rank r on E if and only if it is a chirotope.*

Vectors and Covectors

The concept of vectors and covectors for oriented matroids has been introduced in 1978 by Bland and Las Vergnas [5].

Definition 31 A *vector* of an oriented matroid is any composition of circuits.

A *covector* of a vector of the dual oriented matroid, i. e. any composition of cocircuits.

The set of vectors $\mathcal{V}$ of an oriented matroid $\mathcal{M}$ can be viewed as partially ordered set. The partial order '$\leq$' is given by

$$Y \leq X \quad \text{if } Y \text{ is a restriction of } X .$$

$(\mathcal{V}, \leq)$ is a pure ordered set of rank $\rho^* = \rho(\mathcal{M}^*)$. It has a unique minimal element 0 and its atoms (covering 0) are the circuits of $\mathcal{M}$. Note that all above definitions and properties of vectors can be easily dualized for covectors.

The formal characterization of the set of vectors of an oriented matroid is contained in the following theorem due to Edmonds and A. Mandel in 1982 [14].

Theorem 32 (Vectors axioms) *A collection $\mathcal{V}$ of signed subsets of a set E is the set of vectors of an oriented matroid if and only if the following properties hold:*

1) $\emptyset \in \mathcal{V}$.
2) symmetry: $\mathcal{V} = -\mathcal{V}$;
3) composition: $\forall X, Y \in \mathcal{V}$, $X \operatorname{circ} Y \in \mathcal{V}$;
4) strong vector elimination: $\forall X, Y \in \mathcal{V}$, $a \in X^+ \cap Y^-$ and $b \in (\underline{X} \setminus \underline{Y}) \cup (\underline{Y} \setminus \underline{X}) \cup (X^+ \cap Y^+) \cup (X^- \cap Y^-)$, there exists $Z \in \mathcal{V}$ such that
i) $Z^+ \subseteq (X^+ \cup Y^+) \backslash \{a\}$;
ii) $Z^- \subseteq (X^- \cup Y^-) \backslash \{a\}$;
iii) $b \in \underline{Z}$.

In Theorem 32, condition 3) can be replaced by one of the following two conditions:

3′) vector elimination: $\forall$ $X, Y \in \mathcal{V}$ and $a \in X^+ \cap Y^-$, there exists $Z \in \mathcal{V}$ such that
i) $Z^+ \subseteq (X^+ \cup Y^+) \backslash \{a\}$;
ii) $Z^- \subseteq (X^- \cup Y^-) \backslash \{a\}$;
iii) $(\underline{X} \backslash \underline{Y}) \cup (\underline{Y} \backslash \underline{X}) \cup (X^+ \cap Y^+) \cup (X^- \cap Y^-) \subseteq \underline{Z}$.

3) Y-approximation of X: $\forall X, Y \in \mathcal{V}$ with $\underline{Y} \subseteq \underline{X}$ and $X^+ \cap Y^- \neq \emptyset$, there is a proper restriction Z of X such that
i) $Z \in \mathcal{V}$;
ii) $(X^+ \setminus Y^-) \cup (X^- \setminus Y^+) \subseteq \underline{Z}$

The oriented matroid operations of deletion and contraction, and the duality concept can be formulated in terms of vectors as formalized in the following two propositions.

Proposition 33 *Let $\mathcal{M}$ be an oriented matroid on E with set of circuits C and set of vectors $\mathcal{V}$, and let $A \subseteq E$.*

- *The set of vectors of the deletion matroid $\mathcal{M} \backslash A$ is the set*

$$\mathcal{V} \setminus A = \{X \in \mathcal{V} : \ \underline{X} \cap A = \emptyset\} .$$

- *The set of vectors of the contraction matroid $\mathcal{M}/A$ is the set*

$$\mathcal{V}/A = \{X \setminus A : \ X \in \mathcal{V}\} .$$

Proposition 34 *Let $\mathcal{M}$ be an oriented matroid on E with set of circuits C and set of vectors $\mathcal{V}$. Then the set of vectors of its dual matroid $\mathcal{M}^*$ is the set $\mathcal{L} = \{Y \in \{+, -, 0\}^E : X \perp Y, \forall X \in C\} = \{Y \in \{+, -, 0\}^E : X \perp Y, \forall X \in \mathcal{V}\}$.*

General Topics

This Section is devoted to showing how the oriented matroids collocate in both pure and applied mathematics and how their four axioms systems arise from the following four topics:

- directed graphs;
- orthogonal pairs of real vector subspaces;
- point configurations and convex polytopes;
- real hyperplane arrangements.

Directed Graphs

Let $D = (V, E)$ be a digraph and let C be the set of its simple cycles. Each cycle $c \in C$ is associated with an orientation, i. e. each of them consists of some forward (positive) edges and some backward (negative) edges. Therefore, any $c \in C$ can be viewed as a signed subset of E with positive and negative parts. A signed subset of E deriving from a cycle $c \in C$ is called a *signed circuit* of D and all signed circuits of D form the collection

$$C = \left\{X = (X^+, X^-) \colon \ X \text{ a signed circuit of } D\right\} .$$

The oriented matroid $\mathcal{M} = \mathcal{M}_D$ of D, also denoted by $\mathcal{M} = \mathcal{M}(E)$, is given by the pair (E, C).

For a digraph D it is also possible to define the set of its signed cocircuits as follows. Let $V = (V_1, V_2)$ be a *minimal cut* of D, i. e. a partition of the nodes of D such that the removing of the edges connecting elements in V_1 to elements in V_2 increases by one the number of components of the underlying undirected graph. Let Y^+ be the set of edges in D from V_1 to V_2 and Y^- be the set of edges from V_2 to V_1, then the sets $Y = (Y^+, Y^-)$ are signed sets called *signed cocircuits* of D, which form the collection

$$C^* = \left\{ Y = (Y^+, Y^-) \colon \begin{array}{c} Y \text{ a signed} \\ \text{cocircuit of } D \end{array} \right\} ,$$

where C^* provides the collection of circuits of the dual oriented matroid of $\mathcal{M}_D$.

It is quite easy to show that the digraph D satisfies the properties 1)–4) of circuit axioms expressed by Definition 9 and that all properties of D are reflected in C and C^*. For example, if D does not contain any oriented cycle, then C will contain no positive circuit, i. e. the matroid corresponding to D will be an acyclic oriented matroid.

Real Vectors Spaces

In this Section the two most important ways to relate oriented matroids to real vector spaces are considered: point configurations and hyperplane arrangements.

Vector Configurations

Generally speaking, given an arbitrary field F and a finite set of vectors that spans a vector space of dimension r over F, the minimal linear dependences generate the circuits of a matroid of rank r. In order to get an oriented matroid, the field F must to be ordered. In more detail, given a finite set $E = \{v_1, \ldots, v_n\}$ of vectors that spans a vector space of dimension r over an ordered field $\{v_1, \ldots, v_n\} \subseteq \mathbf{R}^r$, then a minimal linear dependence is such that

$$\sum_{i=1}^{n} \lambda_i v_i = 0 ,$$

with $\lambda_i \in \mathbf{R}$ and the circuits of the associated oriented matroid $\mathcal{M} = (E, C)$ of a vector configuration E are the sets $X = (X^+, X^-)$ such that

$$X^+ := \{i \colon \ \lambda_i > 0\} , \quad X^- := \{i \colon \ \lambda_i < 0\}$$

for all the minimal dependences among the vectors v_i.

The bases of the matroid corresponding to a vector configuration E are the subsets of E that form vector space bases, i. e. all subsets $\{v_{i_1}, \ldots, v_{i_r}\}$ of E such that $\det(\{v_{i_1}, \ldots, v_{i_r}\}) \neq 0$.

For a vector configuration over the real field $\mathbf{R}$, let consider the signs of the determinants of ordered r-subsets of $\{v_1, \ldots, v_n\}$, then the basis orientation or chirotope of the vector configuration is defined as

$$\chi(i_1, \ldots, i_r) := \operatorname{sign} \det\{v_{i_1}, \ldots, v_{i_r}\} ,$$

where $\chi(i_1, \ldots, i_r) \in \{+, -, 0\}$. χ is an asymmetric function and satisfies the properties 1)–3) of chirotope axioms expressed in Definition 29.

Point Configurations

In the following, starting from the observation that every vector configuration in $\mathbf{R}^r \backslash \{0\}$ corresponds to an affine point configuration in an $(r-1)$-dimensional affine space, it will be showed how any point configuration in a real affine space leads to an acyclic oriented matroid. In fact, after choosing a linear form $l_0(v_i) \neq 0$ for all i it is possible to define an $(r-1)$-dimensional affine space as

$$\mathcal{A}^{r-1} := \{x \in \mathbf{R}^r \colon \ l_0(x) = 1\} ,$$

and to associate with each vector v_i the point $1/l_0(v_i)v_i \in \mathcal{A}^{r-1}$. Vectors v_i with $l_0(v_i) < 0$ correspond to points called 'points with negative weight'. If the given vector configuration does not contain any positive linear dependences, i. e.

$$\sum_i \lambda_i v_i = 0, \quad \lambda_i \geq 0 ,$$

then it is possible to choose l_0 such that $l_0(v_i) > 0$ for all i. This corresponds to the situation where the oriented matroid is acyclic, which can be always achieved by simply replacing some of the vectors v_i by their negatives. The circuits of the corresponding acyclic oriented matroid are given by the signs of the coefficients in the minimal affine dependences ($\sum_i \lambda_i v_i = 0$, $\lambda_i \geq 0$). For example, the vertices of a complex polytope describe an acyclic oriented matroid.

The sign patterns of arbitrary (possibly non minimal) affine dependences can be derived from the circuits by compositions as follows. Given two signed sets X and Y,

$$X \circ Y = \left(X^+ \cup (Y^+ \setminus X^-), X^- \cup (Y^- \setminus X^+)\right) .$$

The signed sets so obtained are called *vectors* of the oriented matroid.

Hyperplane Arrangements.

A real hyperplane arrangement $\mathcal{H} = \{H_1, \ldots, H_n\}$ is a finite set of hyperplanes through the origin in $\mathbf{R}^r$. Since every hyperplane is defined by giving a linear function $l_i(x) = \sum_{j=1}^r a_{ij}xj$, any hyperplane can be defined as $H_i = \{x \in \mathbf{R}^r : l_i(x) = 0\}$. Since l_i can be viewed as vectors in the dual space $(\mathbf{R}^r)^*$, they form a vector configuration in $\mathbf{R}^r)^*$, which determines an oriented matroid. In fact, once chosen the vectors l_i, a positive side $H_i^+ = \{x \in \mathbf{R}^r : l_i(x) \geq 0\}$ of H_i is distinguished and the oriented matroid corresponds to the arrangement of halfspaces $\{H_i^+ : 1 \leq i \leq n\}$ in $\mathbf{R}^r$.

As showed below in Theorem 36, not only every real arrangement of hyperplanes gives rise to an oriented matroid, but the inverse is also 'nearly' true.

Topological Representation Theorem

In [15] Folkman and Lawrence showed that each oriented matroid has a pseudosphere representation. This property, expressed in the so called *topological representation theorem*, is a generalization of the hyperplane arrangement model to arbitrary oriented matroids. In this Section the topological representation theorem will be treated superficially, giving only its meaning and some of the most important consequences that it implies. For more details about this fundamental result in the theory of oriented matroids, see [2].

Let E be a finite, parallel-free, spanning set of nonzero vectors in $\mathbf{R}^{r+1}$, and let $C \subseteq \{+, -, 0\}^E$ be the set of signed circuits of the corresponding oriented matroid. For each $e \in E$, let $S_e = \{x \in \mathbf{R}^{r+1} : \langle x, e\rangle = 0, \|x\| = 1\}$. The positive and negative parts of S_e are respectively $S_e^+ = \{x \in \mathbf{R}^{r+1} : \langle x, e\rangle \geq 0, \|x\| = 1\}$ and $S_e^+ = -S_e^+$. It is easy to prove that

1) S_e^+ and S_e^- are subsets of the unit r-sphere $S^r = \{x \in \mathbf{R}^{r+1} : \|x\| = 1\}$;
2) S_e is a linear $(r-1)$-sphere;
3) S_e^+ and S_e^- are the two closed hemispheres of S_e, which is the intersection of S^r and the hyperplane orthogonal to e, so that the arrangement of spheres $\mathcal{A} = (S_e)_{e \in E}$ is equivalent to an arrangement of hyperplane discussed above.

In fact, once established the arrangement of spheres $\mathcal{A}$ and distributed the signs + and − to the hemispheres, such signed arrangements of $(r-1)$-spheres in S^r identify an oriented matroid of rank $r+1$. In more details, the signed circuits C in this case are the vectors $X \in \{+, -, 0\}^E$ such that

c_1) $\cup_{e \in \underline{X}} S_e^{X_e} = S^r$, where $S_e^{X_e}$ is either S_e^+ or S_e^-;
c_2) $\underline{X} = \{e \in E : X_e \neq 0\}$ is minimal with property c_1.

A subset S of S^r is called a *pseudosphere* if there exists a homomorphism $h : S^r \to S^r$ such that $S = h(S^{r-1})$, where $S^{r-1} = \{x \in S^r : x_{r+1} = 0\}$.

Definition 35 An arrangement of pseudospheres $\mathcal{A} = (S_e)_{e \in E}$ is a finite set of pseudospheres S_e in S^r such that

- for $A \subseteq E$ each $S_A = \cap_{e \in A} S_e \neq \emptyset$, is homeomorphic to a sphere of some dimension;
- for every $e \in E$ and every nonempty intersection S_A such that $S_A \not\subseteq S_e$, the intersection $S_A \cap S_e$ is a pseudosphere in S_A with sides $S_A \cap S_e^+$ and $S_A \cap S_e^-$.

Theorem 36 (Topological representation) *Let $\mathcal{A}$ be a signed arrangement of pseudospheres, and let $C(\mathcal{A})$ be the family of the sign vectors $X \in \{+, -, 0\}^E$ that satisfy c_1 and c_2, then*

tr_1) *if* $\mathcal{A} = (S_e)_{e \in E}$ *is a signed arrangement of pseudospheres in* S^r*, then* $C(\mathcal{A})$ *is the family of circuits of an oriented matroid on E and whose rank is r + 1;*

tr_2) *if (E, C) is an oriented matroid of rank r + 1, then there exists a signed arrangement of pseudospheres* $\mathcal{A}$ *in* S^r *such that* $C = C(\mathcal{A})$*;*

tr_3) *given two signed arrangements* $\mathcal{A}$ *and* $\mathcal{A}'$*, then* $C = C(\mathcal{A}) = C = C(\mathcal{A}')$ *if and only if* $\mathcal{A}' = h(\mathcal{A})$ *for some self-homomorphism h of* S^r*.*

Corollary 37 *There is a 1-to-1 correspondence between arrangement of pseudospheres in* S^r *and oriented matroid of rank r + 1.*

Conclusions

Starting from the 1950s, matroids and oriented matroids have had increasing interest. A huge number of scientific works have been published on those subjects and a large collection of matroid theorems and theoretical results exists.

This article has introduced the combinatorial theory of oriented matroids, providing their axiomatic definitions and their basic properties.

See also

► Matroids

References

1. Bachem A, Kern W (1992) Linear programming duality: An introduction to oriented matroids. Springer, Berlin
2. Björner A, Las Vergnas M, Sturmfels B, White N, Ziegler GM (1993) Oriented matroids. Encycl Math Appl, vol 46. Cambridge University Press, Cambridge
3. Bland RG (1977) A combinatorial abstraction of linear programming. J Combin Th B 23:33–57
4. Bland RG (1977) New finite pivoting rules for the simplex method. Math Oper Res 2:103–107
5. Bland RG, Las Vergnas M (1978) Orientability of matroids. J Combin Th B 23:94–123
6. Bland RG, Las Vergnas M (1979) Minty colorings and orientations of matroids. Ann New York Acad Sci 319:86–92
7. Edmonds J (1965) Lehman's switching game and a theorem of Tutte and Nash–Williams. J Res Nat Bureau Standards (B) 69:73–77
8. Edmonds J (1965) Maximum matching and a polyhedron with {0, 1} vertices. J Res Nat Bureau Standards (B) 69:125–130
9. Edmonds J (1965) Minimum partition of a matroid into independent subsets. J Res Nat Bureau Standards (B) 69:67–72
10. Edmonds J (1965) Paths, trees, and flowers. Canad J Math 17:449–467
11. Edmonds J (1967) Optimum branchings. J Res Nat Bureau Standards (B) 71:233–240
12. Edmonds J (1967) Systems of distinct representatives and linear algebra. J Res Nat Bureau Standards (B) 71:241–245
13. Edmonds J (1970) Submodular functions, matroids, and certain polyhedra. In: Guy R, Hanani H, Sauer N, Schönheim J (eds) Combinatorial Structures and Their Applications. Gordon and Breach, New York
14. Edmonds J, Mandel A (1982) Topology of oriented matroids. PhD Thesis of A. Mandel, University Waterloo
15. Folkman J, Lawrence J (1978) Oriented matroids. J Combin Th B 25:199–236
16. Fulkerson DR (1968) Networks, frames, blocking systems. In: Dantzig GB, Veinott AF (eds) Mathematics of the Decision Systems, Part I. Lect Appl Math., vol 2. Amer. Math. Soc., Providence, pp 303–334
17. Lawrence J (1975) Oriented matroids. PhD Thesis, University Washington, Seattle
18. Lawrence J (1982) Oriented matroids and multiply ordered sets. Linear Alg & Its Appl 48:1–12
19. Minty GJ (1966) On the axiomatic foundations of the theories of directed linear graphs, electrical networks, and network-programming. J Math Mechanics 15:485–520
20. Oxley JG (1992) Matroid theory. Oxford University Press, Oxford
21. Rockafellar RT (1969) The elementary vectors of a subspace of R^n. In: Combinatorial Mathematics and Its Applications (Proc. Chapel Hill Conf.). University North Carolina Press, Chapel Hill, pp 104–127
22. Tutte WT (1958) A homotopy theorem for matroids I–II. Trans Amer Math Soc 88:144–160; 161–174
23. Tutte WT (1959) Matroids and graphs. Trans Amer Math Soc 90:527–552
24. Tutte WT (1960) An algorithm for determining wheter a given binary matroid is graphic. Proc Amer Math Soc 11:905–917
25. Tutte WT (1961) On the problem of decomposing a graph into n connected factors. J London Math Soc 36:221–230
26. Tutte WT (1964) From matrices to graphs. Canad J Math 16:108–127
27. Tutte WT (1965) Lectures on matroids. J Res Nat Bureau Standards (B) 69:1–47
28. Tutte WT (1966) Connectivity in graphs. University Toronto Press, Toronto
29. Tutte WT (1966) Connectivity in matroids. Canad J Math 18:1301–1324
30. Tutte WT (1971) Introduction to the theory of matroids. Amer. Elsevier, New York
31. Las Vergnas M (1975) Coordinatizable strong maps of matroids (preprint)

32. Las Vergnas M (1975) Matroides orientables. CR Acad Sci Paris Ser A 280:61–64
33. Las Vergnas M (1977) Acyclic and totally cyclic orientations of combinatorial geometries. Discret Math 20:51–61
34. Las Vergnas M (1978) Bases in oriented matroids. J Combin Th B 25:283–289
35. Whitney H (1935) On the abstract properties of linear dependence. Amer J Math 57:509–533
36. Ziegler GM (1998) Oriented matroids today. Electronic J Combin 3

Orthogonal Triangularization

UDAYA BHASKAR VEMULAPATI
School of Computer Sci., University Central Florida, Orlando, USA

MSC2000: 65F25, 15A23, 65F05, 65F20, 65F22

Article Outline

Keywords
Orthogonal Factorization
Rank Revealing Factorizations
Complete Orthogonal Factorization
See also
References

Keywords

QR factorization; Rank revealing factorization; Complete orthogonal factorization

Triangularization is a process of reducing a square (rectangular) matrix into upper-triangular (upper-trapezoidal) form by applying a series of *elementary transformations*. There are basically two types of elementary transformations: orthogonal and nonorthogonal. Examples of nonorthogonal transformations include classical Gaussian transforms, and Gauss–Jordan transforms.

Orthogonal Factorization

Here the matrix $A \in \mathbf{R}^{m\times n}$ is reduced to an upper-trapezoidal form using a sequence of elementary orthogonal transformations (such as Householder or Givens transformations; see ► QR factorization for more details).

Rank Revealing Factorizations

Orthogonal factorizations can also be used effectively for computing the *numerical rank* [2] of a matrix. The general idea is to identify the independent columns of the matrix and permute them to the left-hand side; i. e. find a permutation matrix Π such that

$$Q^\top A\Pi = \begin{array}{c} \\ r \\ m-r \end{array} \begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \end{array}, \tag{1}$$

where R_{11} is nonsingular, upper triangular and $\| R_{22} \| < \epsilon$. It is said that the ϵ-rank of A is r. The following *QR factorization with column-pivoting* computes factorization in (1). Given $A \in \mathbf{R}^{m\times n}$ with $m \geq n$, the following algorithm computes $r = \text{rank}(A)$ by permuting columns while computing the QR factorization (see [2] for details).

```
for j = 1 : n
    c(j) = A(1 : m, j)^⊤ A(1 : m, j);
end;
r = 0;
find   k s.t. c(k) = max(c(1 : n));
       t = c(k);
while  t > ϵ
       r = r + 1;
       exchange columns i and k;
       exchange c(k) and c(i);
       v = house(A(r : m, r));
       Apply to rest of the columns;
       for i = r + 1 : n
           c(i) = c(i) − A(r, i)^2;
       end;
       if r < n
           find k such that
           c(k) = max(c(r + 1 : n));
           t = c(k);
       else t = 0;
       end;
```

Complete Orthogonal Factorization

Sometimes, it is desirable to reduce the matrix $A \in \mathbf{R}^{m\times n}$ to the following form

$$Q^\top AZ = \begin{pmatrix} T_{11} & 0 \\ 0 & 0 \end{pmatrix},$$

where $T_{11} \in \mathbf{R}^{r\times r}$ is nonsingular and upper triangular. Such a factorization is rank revealing and the rank of A will be r. Such factorizations are very useful in rank-deficient optimization problems (such as rank-deficient least squares) [1].

The first step towards complete orthogonal factorization is to apply QR factorization with column pivoting. Thus,

$$Q^\top A\Pi = \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} .$$

Then apply a series of orthogonal transformations on the right-hand side so that R_{12} is zeroed. While R_{12} is being zeroed, values in R_{11} will change, and let us call the modified R_{11} as T_{11}. The only trick here is make sure that while R_{12} is being zeroed, zeros that are already in R_{11} are not disturbed.

Let us consider an example of a matrix $A \in \mathbf{R}^{5\times 6}$ whose rank is three. After the QR with column pivoting, let

$$Q^\top A\Pi = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ 0 & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ 0 & 0 & a_{33} & a_{34} & a_{35} & a_{36} \\ 0 & 0 & 0 & a_{44} & a_{45} & a_{46} \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} \end{pmatrix} ,$$

where

$$\left\| \begin{pmatrix} a_{44} & a_{45} & a_{46} \\ a_{54} & a_{55} & a_{56} \end{pmatrix} \right\|_2 < \epsilon$$

and Q is orthogonal and Π is some permutation matrix. The following sequence of Givens transformations,

$$Q^\top A\Pi G^3_{36} G^3_{35} G^3_{34} G^2_{26} G^2_{25} G^2_{24} G^1_{16} G^1_{15} G^1_{14}$$

will zero R_{12} without disturbing the zeros that are already present in R_{11}.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- Cholesky Factorization
- Interval Linear Systems
- Large Scale Trust Region Problems
- Large Scale Unconstrained Optimization
- Linear Programming
- Overdetermined Systems of Linear Equations
- QR Factorization
- Solving Large Scale and Sparse Semidefinite Programs
- Symmetric Systems of Linear Equations

References

1. Gill PE, Murray W, Wright MH (1991) Numerical linear algebra and optimization, vol 1. Addison-Wesley, Reading
2. Golub GH, Van Loan CF (1997) Matrix computations. Johns Hopkins University Press, Baltimore

Outranking Methods

Denis Bouyssou
LAMSADE University Paris Dauphine, Paris, France

MSC2000: 90-XX

Article Outline

Keywords
Basic Ideas
ELECTRE I
Extensions
Practical Considerations
Theoretical Appraisal
Practical Applications
  Environment
  Finance
  Health
  Location
  Transportation
  Miscellaneous
See also
References

Keywords

MCDM; ELECTRE; Outranking; Concordance; Discordance; Preference modeling

A classical problem in the field of *multiple criteria decision making* (MCDM) is to build a preference relation on a set of multi-attributed alternatives on the basis of preferences expressed on each attribute and 'inter-attribute' information such as weights. Based on this

preference relation (or, more generally, on various relations obtained following a robustness analysis) a recommendation is elaborated (e. g. exhibiting a subset likely to contain the 'best' alternatives).

A common way [20] to do so is to attach a number $v(x)$ to each alternative $x \in X$ and to declare that x is 'at least as good as' y if and only if $v(x) \geq v(y)$. The number $v(x)$ depends on the evaluations $x_1, \ldots, x_n$ of x on the n attributes and we have $v(x) = V(x_1, \ldots, x_n)$. The most common form for V is an additive value function in which $V(x_1, \ldots, x_n) = \sum_{i=1}^{n} k_i v_i(x_i)$; in that case the task of the analyst reduces down to assessing the partial value functions v_i and the scaling constants k_i. The preference relation that is built using this *value function approach* is a *weak order*, i. e. a complete and transitive binary relation. Using such information it is not difficult, in general, to elaborate a recommendation. The definition of the aggregation function V may not always be simple however. Making all alternatives comparable in a 'nice transitive way' requires much information and, in particular, a detailed analysis of trade-offs between attributes.

Outranking methods (OMs) were first developed in France in the late 1960s following difficulties experienced with the value function approach in dealing with practical problems. They are closely associated with the name of B. Roy, who developed the well-known family of ELECTRE methods. A large part of the literature on OMs was written in French which has been prejudicial to their international diffusion; good accounts in English are [18,32,38,39,46,53,56,57], while detailed references in French include [24,31,41,47,48].

Basic Ideas

As in the value function approach, OMs build a preference relation, usually called an *outranking relation*, among alternatives evaluated on several attributes. Roy defines an outranking relation as a binary relation S on the set X of alternatives such that xSy if, given what is known about the preferences of the decision-maker, the quality of the evaluations of the alternatives and the nature of the problem, there are enough arguments to decide that x is at least as good as y, while there is no essential reason to refute that statement.

In most OMs the outranking relation is built through a series of *pairwise comparisons* of the alternatives (this implies that these methods deal with *finite sets of the alternatives*; their underlying principles may however be adapted in order to deal with infinite sets [19]). Although pairwise comparisons can be done in many ways, the *concordance-discordance* principle is prevalent in most OMs (exceptions include [2,50]). It consists in declaring that an alternative x is at least as good as an alternative y (xSy) if:

- a *majority* of the attributes supports this assertion (*concordance condition*); and
- the opposition of the other attributes (the *minority*) is not 'too strong' (*nondiscordance condition*).

This principle is at variance with the ones underlying the value function approach. It rests on a 'voting' analogy and may be used without having recourse to a subtle analysis of trade-offs between attributes. It mainly uses *ordinal* considerations and has a strong *noncompensatory* flavor [3,11]. The application of this principle gives rise, in general, to binary relations which are neither complete (i. e. it is possible that Not(xSy) and Not(ySx)) nor transitive (i. e. we may have xSy, ySz and Not(xSz)). Exploiting an outranking relation in order to arrive at a recommendation is therefore not an easy task and calls for the application of specific techniques [41,53].

We briefly describe below ELECTRE I [35], which is the oldest and simplest OM before coming to some extensions and comments.

ELECTRE I

Consider a finite set of alternatives X evaluated on a family $N = \{1, \ldots, n\}$ of attributes. A first step in the comparison of two alternatives $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ is to know how they compare on each attribute. ELECTRE I uses a traditional preference model for this purpose: a weak order (i. e. a complete and transitive binary relation) S_i is supposed to be defined on each $i \in N$, $x_i S_i y_i$ meaning that x is judged at least as good as y on attribute i. Dealing with a finite set, it is not restrictive (see [15]) to assume the existence of a real valued function g_i such that

$$x_i S_i y_i \;\Leftrightarrow\; g_i(x_i) \geq g_i(y_i)\,.$$

Quite often in practice, numbers are used to evaluate the alternatives on the various attributes and the relations S_i stem from the comparison of these numbers [4].

In ELECTRE I, the analysis of the proposition xSy rests on the partition of the set N of attributes into a *concordant coalition* $C(xSy) = \{j \in N: g_j(x_j) \geq g_j(y_j)\}$ and a *discordant coalition* $D(xSy) = \{j \in N: g_j(x_j) < g_j(y_j)\}$. The proposition xSy will be accepted if the concordant coalition $C(xSy)$ is 'sufficiently important' (concordance condition) and if on any of the attributes in $D(xSy)$ the 'difference of preference' in favor of y is not considered to be 'large' (condition of nondiscordance). In order to implement the concordance condition, a positive *weight* k_j is assigned to each attribute $j \in N$ and the importance of a coalition is supposed to be represented by the sum of the weights of the attributes belonging to that coalition. Thus the concordance index $c(x, y) = \sum_{j \in C(xSy)} k_j / \sum_{j \in N} k_j$ represents the relative importance of the coalition $C(xSy)$ in the set N of all attributes; we have $c(x, y) \in [0, 1]$. Whether or not $C(xSy)$ is 'sufficiently important' is then judged comparing $c(x, y)$ to a *concordance level* $s \in [1/2, 1]$. It is worth noting that the partition of N into $C(xSy)$ and $D(xSy)$ and the computation of the concordance index $c(x, y)$ rest on purely ordinal comparisons: altering the functions g_j without altering the binary relations S_j will not change the values of the concordance index. Suppose that $c(x, y) \geq s$. Concluding that xSy) would give no power to the attributes in $D(xSy)$). If on any of these attributes the, positive, preference difference between y and x is 'large' there are good reasons to reject the proposition xSy). The definition of 'large' preference differences is done in ELECTRE I via the definition of nonnegative *veto thresholds* v_j (which may vary with g_j) on each attribute; a preference difference is declared 'large' as soon as $g_j(y_j) - g_j(x_j) > v_j$. It should be noticed that the implementation of the nondiscordance principle through the definition of veto thresholds v_j linked to a particular functions g_j is a matter of commodity only; what is in fact looked for is a subset of the asymmetric part of S_j corresponding to 'large' preference differences, which may be done independently of any numerical representation. In summary, we have in ELECTRE I:

$$\begin{gathered} xSx \\ \Updownarrow \\ c(x, y) \geq s \\ \text{and} \\ g_j(y_j) - g_j(x_j) < v_j, \ \forall j \in D(xSy) \, . \end{gathered}$$

When $s = 1$ (which amounts to requiring unanimity of the attributes in order to accept outranking) or $v_j = 0$ for all j (implying that all positive preference differences are 'large'), the outranking relation S is nothing but the so-called *dominance relation* Δ defined by

$$x \Delta y \ \Leftrightarrow \ [x_j S_j y_j \text{ for all } j \in N] \, .$$

It is not difficult to see that it is always true that $\Delta \subseteq S$. An outranking relation may be usefully seen as an enrichment of the dominance relation Δ in which unanimity of the attributes is not required and not all positive preference differences are considered 'large'; decreasing the value of s and/or increasing the values of the v_j results in a richer but somewhat riskier outranking relation. Although, the dominance relation Δ is clearly reflexive and transitive (but not complete), simple examples, inspired by *Condorcet's paradox* [49], show that, in general, S is neither complete nor transitive when $s < 1$ and $v_j > 0$.

It is important to note that in ELECTRE I, the weights k_j cannot be interpreted as substitution rates or *trade-offs*; they are thus fundamentally different from the scaling constants that are used in the *value function approach*. In line with the voting analogy underlying the concordance-discordance principle, it is useful to interpret k_j as the 'number of votes' given to attribute j (this number of votes being independent of the choice of the function g_j), the concordance threshold s specifying a level of 'qualified majority'.

ELECTRE I was originally designed to lead to 'choice-type' results. Since S may not be complete or transitive, the set $\{x \in X: xSy \text{ for all } y \in X\}$ of *maximal alternatives* (in X given S) can be empty. In order to overcome this difficulty, ELECTRE I determines the minimal (with respect to inclusion) set of alternatives not outranking each other such that all the alternatives outside of this set are outranked by at least one alternative from this set. Technically, this leads to the determination of the *kernel of the graph* (X, S) after the detection and elimination by reduction of possible circuits (a well-known result in graph theory proves the existence and unicity of kernels in graphs without circuits).

Extensions

Besides ELECTRE I, many other OMs have been proposed in the literature [13,14,23,33,36,40,42,45,58].

They mainly differ on:

- the type of result that is looked for (e. g. one may wish to use S to rank order alternatives or to sort them into pre-defined categories);
- the way the outranking relation is built. It is indeed possible to implement the concordance-discordance principle in various ways (e. g. allowing for synergy effects in $D(xSy)$). Moreover, varying the values of s and of the thresholds v_j lead to several different relations S; such variations are incorporated in methods which use several nested outranking relations (as in [40,42]) or a *fuzzy outranking relation* in which a 'credibility' is attached to each arc in the graph (X, S) as in ([13,14,36,58]) (on the notion of fuzzy outranking relation see ([17,28]);
- the way alternatives are compared on each attribute. In ELECTRE I it is postulated that alternatives can be compared on each attribute according to a weak order. This traditional preference model may be inappropriate considering the inevitable elements of imprecision, uncertainty and inaccurate determination entering the evaluations of the alternatives. Indifference on each attribute may not be fully transitive; moreover there may exist cases in which the transition from indifference to strict preference is not without ambiguity giving rise to models involving 'weak preference' relations (such models involve *indifference* and/or *preference thresholds* [34,37,51,52].

The following table summarizes the main characteristics of the existing ELECTRE methods and might help in choosing an appropriate OM. See [41] and [56] for a complete description of these methods and of many others in a similar vein, in particular the TACTIC method [54] and the family of PROMETHEE methods [13,14].

Outranking Methods, Table 1
Main characteristics of existing ELECTRE methods; adapted from [41]

Electre methods	Preference model on each attribute	Use of weights	No. of outrank. rel. used	Type of result
I [35]	trad.	yes	1	Choice
IS [45]	nontrad.	yes	1	Choice
II [40]	trad.	yes	2	Ranking (partial)
III [36]	nontrad.	yes	1 (fuzzy)	Ranking (partial)
IV [42]	nontrad.	no	up to 5	Ranking (partial)
Tri [41] [58]	nontrad.	yes	1	Assignm. into predef. categ.

Practical Considerations

We give here some indications on how to give a value to the parameters used in ELECTRE I: weights k_j, veto thresholds v_j and the concordance threshold s (they may be transposed to all ELECTRE methods; for a more detailed account see [24,26,41,48] and for an alternative approach [21]). Before doing so, it is important to note that the underlying philosophy of OMs is not to describe as accurately as possible the preferences of a decision-maker. This decision-maker is often a remote abstract entity (the state, the region, the firm); when this is not the case he/she is frequently not very accessible and his/her preferences may be only very partially structured. Searching for the 'true' values of k_j, v_j or s makes little sense under these conditions. The *concordance-discordance* principle is best seen as a useful and easily understandable convention to help structuring preferences. The 'assessment' of the parameters of the method should therefore aim at transforming what appears to be the stable basic judgements of the actors to be helped into numerical values. Needless to say that, under these conditions, the elaboration of a recommendation should be preceded by a thorough robustness analysis.

In order to give a numerical value to the weights k_j it is useful to envisage imaginary but realistic alternatives combining plausible evaluations on the various attributes. Consider two such alternatives x and y such that $g_j(x_j) > g_j(y_j)$ for all $j \in J \subset N$ and $g_i(y_i) > g_i(x_i)$ for all $i \notin J$. If the differences between the evaluations of x and y have been chosen in such a way as to avoid 'large' preference differences and if it may be agreed that x is at least as good as y while y is not at least as good as x, we

can then infer that $\sum_{j \in J} k_j \geq s$ and $\sum_{i \notin J} k_i < s$, supposing without loss of generality that $\sum_{j \in N} k_j = 1$. Combining several questions of this type gives rise to a polyhedron of plausible values for k_j and s to be explored during the robustness analysis, see [43,44]. It should be noted here that the precise numerical values of k_j and s are irrelevant in ELECTRE I as long as they imply a similar partition of subsets of attributes into 'winning' coalitions (for which the sum of the weights exceed the concordance threshold) and 'losing' ones.

Consider now two imaginary alternatives x and y such that $g_j(x_j) > g_j(y_j)$ for all $j \in N \backslash \{i\}$ and choose $g_i(y_i)$ to be one of the best evaluations on attribute i and $g_i(x_i)$ to be one of the worst. We have $D(xSy) = \{i\}$. If it can be accepted that xSy, then it is clear that no veto power should be conferred to attribute i, which amount to setting v_i to an arbitrarily large number. If not, attribute i has a veto power; in order to give a value to v_i one can then increase $g_i(x_i)$ and/or decrease $g_i(y_i)$ till xSy is accepted. A slightly larger value than the difference $g_i(y_i) - g_i(x_i)$ leading to the acceptance of xSy gives a plausible value for v_i (note that before choosing a constant value of v_i it should be checked that the maximum difference $g_i(y_i) - g_i(x_i)$ on attribute i compatible with xSy does not vary along the scale of g_i; when this is the case variable thresholds can be easily used).

Theoretical Appraisal

OMs have often been criticized for their lack of axiomatic foundations; ELECTRE I was proposed on a more or less ad hoc basis and subsequent methods aimed at extending it. The situation has changed dramatically in recent years giving rise to a variety of studies investigating the foundations of these methods. In particular, it is worth mentioning that:

- the links between concordance-discordance principle leading to possibly intransitive and incomplete outranking relations and classical aggregation problems in social choice theory (exemplified by Arrow's impossibility theorem; see [49]) has been studied in depth [1,5,27];
- outranking methods may be axiomatised in more or less the same way as the various instances of the value function approach (see [3,10,11,30,54]), the axioms emphasizing the 'ordinal' and 'noncompensatory' features of the methods;
- the structural properties of outranking relations have been studied in depth [7], this problem having strong links with the classical problems of the construction of voting paradoxes [25] and the binary choice probabilities problem [16];
- various ways of exploiting outranking relations have been carefully analyzed and/or axiomatized see [6,8,9,12,22,29,55].

This literature on the foundations of OMs while still being in its early stages has already greatly contributed to a better understanding of these methods and their underlying hypotheses.

Practical Applications

OMs have been applied in real-world studies since their creation. It is impossible to give here a complete list of applications and references. We only mention a few significant applications in various fields (detailed bibliographical indications may be found in [41]).

Environment

Forestry management (Canada), Nuclear waste management (Belgium), Pollution prevention and control (France), Solid waste management (Finland, Greece), Water resource management (France, Hungary, USA);

Finance

Allocation of grants (Belgium), Analysis of the international diversification of portfolios (Canada), Equitable burden sharing in international institutions (Belgium), Investment planning (France), Portfolio management (Canada);

Health

Computer-aided diagnosis (France), Epidemiology (France), Identification of bacteria (Belgium), Management of hospitals (Canada);

Location

Airports (Canada, the Netherlands), High voltage electric lines (France, Canada), Schools (France), Thermal power plants (Algeria);

Transportation

Choice of a highway route (France), Planning the renovation of metro stations (France), Selection of suburban metro extensions projects (France);

Miscellaneous

Analysis of tenders (France, Portugal), Choice between forecasting models (Belgium), Choice of a marketing strategy (France), Inventory management (France), Production planning in a job-shop (Canada), Promotion of navy officers (Portugal), Regional planning (The Netherlands).

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Arrow KJ, Raynaud H (1986) Social choice and multicriterion decision making. MIT, Cambridge
2. Bana e Costa CA (1990) An additive value function technique with fuzzy outranking relation for dealing with poor intercriteria preference information. In: Bana e Costa CA (ed) Readings in Multiple Criteria Decision Aid. Springer, Berlin, pp 351–382
3. Bouyssou D (1986) Some remarks on the notion of compensation in MCDM. Europ J Oper Res 26:150–160
4. Bouyssou D (1990) Building criteria: A prerequisite for MCDA. In: Bana e Costa CA (ed) Readings in multiple criteria decision aid. Springer, Berlin, pp 58–80
5. Bouyssou D (1992) On some properties of outranking relations based on a concordance-discordance principle. In: Duckenstein L, Goicoechea A and Zionts S (eds) Multiple criteria decision making. Springer, Berlin, pp 93–106
6. Bouyssou D (1992) Ranking methods based on valued preference relations: A characterization of the net flow method. Europ J Oper Res 60:61–67
7. Bouyssou D (1996) Outranking relations: do they have special properties? J Multi-Criteria Decision Anal 5:9–111
8. Bouyssou D, Perny P (1992) Ranking method for valued preference relations: A characterization of a method based on leaving and entering flows. Europ J Oper Res 61:186–194
9. Bouyssou D, Pirlot M (1997) Choosing and ranking on the basis of fuzzy preference relations with the "Min in Favor". In: Fandel G, Gal T (eds) Multiple criteria decision making, Proc. 12th Internat. Conf., Hagen. Springer, Berlin, pp 115–127
10. Bouyssou D, Pirlot M, Vincke Ph (1997) A general model of preference aggregation. In: Karwan MH, Spronk J, Wallenius J (eds) Essays in decision making. Springer, Berlin, pp 120–134
11. Bouyssou D, Vansnick J-C (1986) Noncompensatory and generalized noncompensatory preferences structures. Theory and Decision 21:251–266
12. Bouyssou D, Vincke Ph (1997) Ranking alternatives on the basis of preference relations: A progress report with special emphasis on outranking relations. J Multi-Criteria Decision Anal 6:77–85
13. Brans JP, Mareschal B, Vincke Ph (1984) PROMETHEE: A new family of outranking methods in multicriteria analysis. In: Brans JP (ed) OR'84. North-Holland, Amsterdam, pp 408–421
14. Brans JP, Vincke Ph (1985) A preference ranking organization method. Managem Sci 31:647–656
15. Fishburn PC (1970) Utility theory for decision making. Wiley, New York
16. Fishburn PC (1992) Induced binary probabilities and the linear ordering polytope: A status report. Math Social Sci 23:67–80
17. Fodor J, Roubens M (1994) Fuzzy preference modelling and multiple criteria decision support. Kluwer, Dordrecht
18. Goicoechea A, Hansen DR, Duckstein L (1982) Multiobjective decision analysis with engineering and business applications. Wiley, New York
19. Jaszkiewicz A, Slowinski R (1995) The light beam search: outranking based interactive procedure for multiple-objective mathematical programming. In: Pardalos PM,

Siskos Y, Zopounidis C (eds) Advances in multicriteria analysis. Kluwer, Dordrecht, pp 129–146
20. Keeney RL, Raiffa H (1976) Decisions with multiple objectives: preferences and value tradeoffs. Wiley, New York
21. Kiss LN, Martel J-M, Nadeau R (1994) ELECCALC: An interactive software for modelling the decision-maker's preferences. Decision Support Systems 12:311–326
22. Marchant Th (1996) Valued relations aggregation with the Borda method. J Multi-Criteria Decision Anal 5:127–132
23. Matarazzo B (1990) A pairwise comparison approach: the MAPPAC and PRAGMA methods. In: Bana e Costa CA (ed) Readings in multiple criteria decision aid. Springer, Berlin, pp 253–273
24. Maystre LY, Pictet J, Simos J (1994) Méthodes multicritères ELECTRE. Press. Polytechniques Univ. Romandes, Lausanne
25. McGarvey DC (1953) A theorem on the construction of voting paradoxes. Econometrica 21:608–610
26. Mousseau V (1995) Eliciting information concerning the relative importance of criteria. In: Pardalos PM Siskos Y, Zopounidis C (eds) Advances in multicriteria analysis. Kluwer, Dordrecht, pp 17–43
27. Perny P (1992) Sur le non respect de l'axiome d'indépendance dans les méthodes de type Electre. Cahiers CERO 34:211–232
28. Perny P, Roy B (1992) The use of fuzzy outranking relations in preference modelling. Fuzzy Sets and Systems 49:33–53
29. Pirlot M (1995) A characterization of "min" as a procedure for exploiting valued preference relations and related results. J Multi-Criteria Decision Anal 4:37–56
30. Pirlot M (1997) A common framework for describing some outranking methods. J Multi-Criteria Decision Anal 6:86–92
31. Pomerol J-Ch, Barba-Romero S (1993) Choix multicritère dans l'entreprise. Hermès
32. Pomerol J-Ch, Barba–Romero S (2000) Multicriterion decision in management: Principles and practice. Kluwer, Dordrecht
33. Roubens M (1982) Preference relations on actions and criteria in multicriteria decision making. Europ J Oper Res 10:51–55
34. Roubens M, Vincke Ph (1985) Preference modelling. Springer, Berlin
35. Roy B (1968) Classement et choix en présence de points de vue multiples (la méthode ELECTRE). RIRO 2:57–75
36. Roy B (1978) ELECTRE III: un algorithme de classement fondé sur une représentation floue des préférences en présence de critères multiples. Cahiers CERO 20:3–24
37. Roy B (1985) Méthodologie multicritère d'aide à la décision. Economica, Paris
38. Roy B (1991) The outranking approach and the foundations of ELECTRE methods. Theory and Decision 31:49–73
39. Roy B (1996) Multicriteria methodology for decision aiding. Kluwer, Dordrecht
40. Roy B, Bertier P (1973) La méthode ELECTRE II: une application au media-planning. In: Ross M (ed) OR'72. North-Holland, Amsterdam, pp 291–302
41. Roy B, Bouyssou D (1993) Aide multicritère à la décision: méethodes et cas. Economica
42. Roy B, Hugonnard J-Ch (1982) Ranking of suburban line extension projects on the Paris metro system by a multicriteria method. Transport Res 16A:301–312
43. Roy B, Mousseau V (1996) A theoretical framework for analysing the notion of relative importance of criteria. J Multi-Criteria Decision Anal 5:145–159
44. Roy B, Present M, Silhol D (1986) A programming method for determining which Paris metro station should be renovated. Europ J Oper Res 24:318–334
45. Roy B, Skalka J-M (1984) ELECTRE IS: Aspects méthodologiques et guide d'utilisation. Doc du Lamsade Univ Paris-Dauphine 30
46. Roy B, Vanderpooten D (1996) The European school MCDA: emergence, basic features and current works. J Multi-Criteria Decision Anal 5:22–37
47. Schäarlig A (1985) Décider sur plusieurs critères: panorama de l'aide à la dècision multicritére. Press. Polytechniques Univ. Romandes, Lausanne
48. Schärlig A (1996) Pratiquer ELECTRE et PROMETHEE. Press. Polytechniques Univ. Romandes, Lausanne
49. Sen AK (1986) Social choice theory. In: Arrow KJ, Intriligator MD (eds) Handbook Math. Economics, vol III. Elsevier, Amsterdam, pp 1073–1181
50. Siskos J (1982) A way to deal with fuzzy preferences in multi-criteria decision problems. Europ J Oper Res 10:314–324
51. Tsoukiàs A, Vincke Ph (1992) A survey on nonconventional preference modelling. Ricerca Oper 61:5–49
52. Tsoukiàs A, Vincke Ph (1997) Extended preference structures in MCDA. In: Climaco J (ed) Multicriteria Analysis. Springer, Berlin, pp 37–50
53. Vanderpooten D (1990) The construction of prescriptions in outranking methods. In: Bana e Costa CA (ed) Readings in Multiple Criteria Decision Aid. Springer, Berlin, pp 184–215
54. Vansnick J-C (1986) On the problem of weights in multiple criteria decision-making (the noncompensatory approach). Europ J Oper Res 24:288–294
55. Vincke Ph (1992) Exploitation of a crisp relation in a ranking problem. Theory and Decision 32:221–240
56. Vincke Ph (1992) Multicriteria decision-aid. Wiley, New York
57. Vincke Ph (1999) Outranking approach. In: Gal T, Stewart TJ, Hanne Th (eds) Multicriteria decision making: Advances in MCDM models, algorithms, theory and applications. Kluwer, Dordrecht, pp 11.1–11.30
58. Yu W (1992) Aide multicritèere à la déecision dans le cadre de la probléematique du tri: concepts, méthodes et applications. Doctoral Thesis, Univ Paris-Dauphine

Overdetermined Systems of Linear Equations

OSLE

MUSTAFA Ç. PINAR
Bilkent University, Ankara, Turkey

MSC2000: 65K05, 65D10

Article Outline

Keywords
See also
References

Keywords

Parameter estimation; Optimization; Approximation

Consider a set of experimental observations represented by a vector b $\in \mathbf{R}^n$. The goal is to estimate a set of parameters $x \in \mathbf{R}^m$ with the help of a matrix of independent input conditions represented by $A \in \mathbf{R}^{n\times m}$. In other words, one wishes to express b in terms of A. However, one may have a larger number of experimental observations than parameters to be estimated, i. e., it may be the case that $n >> m$. The problem described above is a typical estimation problem which gives rise to an overdetermined system of linear equations:

$$Ax = b\,. \tag{1}$$

In general one cannot expect to obtain a vector x which satisfies (1) even if A has m linearly independent columns. This feature of the problem leads to the search for a vector x which makes Ax as close as possible to b. The closeness is measured in some suitable norm which is usually either the 2-norm, or the 1-norm, or the ∞-norm. The most common is the 2-norm which yields the well-known linear *least squares problem*:

$$\min_x \|Ax - b\|_2 = \sqrt{(Ax-b)^\top (Ax-b)}\,. \tag{2}$$

The linear least squares approach is usually preferred because it leads to a simpler problem. More precisely, it admits a closed-form solution which can be obtained by solving the linear system of equations:

$$A^\top Ax = 2A^\top b\,. \tag{3}$$

Since $A^\top A$ is a symmetric positive (semi)definite matrix (it is positive definite when A has m linearly independent columns, in which case the solution is unique) it can be decomposed in the form of $LDL^\top$ (or, Choleski factorization) where L is unit lower triangular, and D is diagonal. The factored form can then be used to solve (3) which has always a solution. However, this method is only reliable when A is a well-conditioned matrix. A more numerically stable way to solve (3) is to use an orthogonal factorization (e. g., QR) combined with a pivoting strategy. A detailed treatment of the linear least squares problem can be found in [8].

In some instances, the set of observations includes gross inaccuracies or wild points. In such cases, it may be preferable to use the 1-norm which leads to the following estimation problem

$$\min_x \|Ax - b\|_1 = \sum_{i=1}^{n} |(Ax-b)_i|\,, \tag{4}$$

where $(Ax-b)_i$ is used to represent the ith component of $Ax - b$. The function in (4) is not differentiable at those points where $(Ax-b)_i = 0$ for some $i \in \{1, \ldots, n\}$. The problem is commonly referred to as the ℓ_1 *estimation problem*. The parameter values obtained from the minimization problem (4) will not be as adversely affected by the presence of wild points as the estimates obtained using (3). On the other hand, in contrast to the linear least squares problem (4) is a combinatorial optimization problem because it can be shown that a minimizing point x has the property that some of the components of the *residual vector* $Ax - b$ are equal to zero, some are positive and some are negative (this property is what makes this approach immune to wild points). Hence, if one had access to the information as to which components are zero, positive, and negative, respectively, one could find a minimizing point x by solving the following linear program:

$$\begin{cases} \min\limits_x & \sum\limits_{i\in\mathcal{A}^c} s_i^*(Ax-b)_i \\ \text{s.t.} & (Ax-b)_i = 0, \quad \forall i \in \mathcal{A}, \end{cases}$$

where $\mathcal{A}$ is the set of indices corresponding to zero components of $Ax - b$, $\mathcal{A}^c$ is its complement with respect to $\{1, \ldots, n\}$, and s_i^* is the *sign function* which assumes the value + 1 for positive residuals, and − 1 for negative residuals, respectively.

Unfortunately, one has a priori no idea about s_i^* and $\mathcal{A}$. An alternative way to pose (4) leads to the following problem:

$$\begin{cases} \min\limits_{\substack{c\geq 0\\ d\geq 0\\ u\geq 0\\ v\geq 0}} & \sum\limits_{i=1}^{n}(u_i+v_i) \\ \text{s.t.} & u-v+A(c-d)=b, \end{cases}$$

with $(Ax-b)_i = u_i + v_i$ and $x_j = c_j - d_j$. The equivalence of (4) and the above linear program is discussed in [17]. The most successful attempts at solving (4) were based on the above reformulation and its dual problem. Notably, I. Barrodale and F.D.K. Roberts [4] specialized the simplex algorithm of linear programming to the above formulation by taking advantage of the complementarity between the u_j and v_j variables in the pivoting process. R.D. Armstrong, E.L. Frome and D.S. Kung [1] developed a revised simplex algorithm for the linear programming formulation of the problem. A different algorithm which aims at minimizing the nondifferentiable 1-norm function (4) was given in [6].

A more recent idea for solving the ℓ_1 estimation problem was given in [12]. This idea is quite different from those mentioned above in that it replaces the original function with a once continuously differentiable function, and leads to the following problem:

$$\min_x \sum_{i=1}^{n} \rho((Ax-b)_i)\,, \tag{5}$$

where

$$\rho(t) = \begin{cases} \frac{t^2}{2\gamma} & \text{if } |t|\leq \gamma, \\ |t| - \frac{\gamma}{2} & \text{if } |t| > \gamma, \end{cases} \tag{6}$$

with t being a knock-off variable, and γ a positive scalar. This function is known as *Huber's M-estimator* function ([9]) in the statistics literature as it was introduced by P.J. Huber as a *robust estimator* in the face of *inaccuracies in the observations*. K. Madsen and H.B. Nielsen observed that they can obtain a solution (4) by repeatedly solving (5) for decreasing values γ tending to zero. They were also able to avoid the potentially ill-conditioning effects of driving γ to zero.

As far as obtaining a set of parameters 'immune' to grossly inaccurate observations, one has the option to use the 1-norm or (4), or the Huber problem (5). It is interesting that Huber's problem was used as a subproblem to solve (4). The relationship between problems (4) and (5) were further explored in [13] and [11].

Another popular choice for the solution of overdetermined systems of linear equations is to compute a solution to minimize the ∞-norm of the residual vector. This approach yields the problem

$$\min_x \max |(Ax-b)_i|\ . \tag{7}$$

The problem is commonly known as the *Chebyshev problem*. Here, one faces again a problem of a combinatorial nature as it can be proved that a solution to (7) has certain residual values equal to the maximum in absolute value, and others smaller than this value in modulus, respectively. This partition of the residuals at a minimizing point is obviously unknown. Hence, one must resort to some algorithm to compute a solution to (7) much the same way as in the case of (4). Here, again there exist approaches based on minimizing the nondifferentiable function in (7) (nondifferentiable at points where at least two residuals attain the maximum value in modulus). The most notable of such methods are that of Bartels–Golub [7], Bartels–Conn–Charalambous [5]. There exist also methods based on the linear programming formulation which is given as follows in [17]:

$$\begin{cases} \min\limits_{x,z} & z \\ \text{s.t.} & -z \leq (Ax-b)_i \leq z, \quad \forall\, i=1,\ldots,n. \end{cases}$$

Some of the approaches based on linear programming favored the above primal formulation for use in a penalty function algorithm [10,15]. Some others used the dual formulation:

$$\begin{cases} \min\limits_{\substack{v\geq 0\\ w\geq 0}} & (v-w)^\top b \\ \text{s.t.} & A^\top(v-w)=0 \\ & e^\top(v+w)=1, \end{cases}$$

where e represents a vector of all ones. Among these approaches, the most successful is the simplex adaptation of [2].

A survey of the use of the 2-norm, 1-norm and ∞-norm criteria in linear regression in statistics is given in [14], but contains only developments until 1981.

Some of the algorithms mentioned above are available as software packages. In particular, the 1-norm algorithms of Barrodale–Roberts and of Bartels–Conn–Sinclair are available in the NAG (Numerical Algorithms Group) software library. The 1-norm and Huber algorithms of Madsen–Nielsen are available from the authors. The Chebyshev algorithm of Barrodale–Phillips is available in the NAG library, and also in the ACM collection [3]. The Chebyshev algorithm of Pĭnar–Elhedhli is available from the authors. A copy of the Bartels–Golub algorithm for the Chebyshev problem can be obtained from [16].

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- Cholesky Factorization
- Interval Linear Systems
- Large Scale Trust Region Problems
- Large Scale Unconstrained Optimization
- Linear Programming
- Nonlinear Least Squares: Trust Region Methods
- Orthogonal Triangularization
- QR Factorization
- Solving Large Scale and Sparse Semidefinite Programs
- Symmetric Systems of Linear Equations

References

1. Armstrong RD, Frome EL, Kung DS (1979) A revised simplex algorithm for the absolute deviation curve fitting problem. Comm Statist–Simula Computa 8:175–190
2. Barrodale I, Phillips C (1974) An improved algorithm for discrete Chebyshev linear approximation. In: Hartnell BL, Williams HC (eds) Proc. Fourth Manitoba Conf. Numer. Math. Utilitas Math. Pub., pp 177–190
3. Barrodale I, Phillips C (1975) Algorithm 495: solution of an overdetermined system of linear equations in the Chebyshev norm. ACM Trans Math Softw 1:264–270
4. Barrodale I, Roberts FDK (1973) An improved algorithm for discrete linear $\ell 1$ approximation. SIAM J Numer Anal 10:839–848
5. Bartels RH, Conn AR, Charalambous C (1978) On Cline's direct method for solving overdetermined linear systems in the $\ell\infty$ sense. SIAM J Numer Anal 15:255–270
6. Bartels RH, Conn AR, Sinclair JW (1978) Minimisation techniques for piecewise differentiable functions: the $\ell 1$ solution to an overdetermined linear system. SIAM J Numer Anal 15:224–241
7. Bartels RH, Golub GH (1968) Stable numerical methods for obtaining the Chebyshev solution to an overdetermined system of equations. Comm ACM 11:401–406
8. Björk A (1996) Numerical methods for least squares problems. SIAM, Philadelphia
9. Huber PJ (1981) Robust statistics. Wiley, New York
10. Joe B, Bartels RH (1983) An exact penalty method for constrained, discrete, linear $\ell\infty$ data fitting. SIAM J Sci Comput 4:69–84
11. Li W, Swetits J (1998) Linear $\ell 1$ estimator and Huber M-estimator. SIAM J Optim 8
12. Madsen K, Nielsen HB (1993) A finite smoothing algorithm for linear $\ell 1$ estimation. SIAM J Optim 3:223–235
13. Madsen K, Nielsen HB, Pinar MČ (1994) New characterizations of $\ell 1$ solutions to overdetermined systems of linear equations. Oper Res Lett 16:159–166
14. Narula SC (1982) Optimization techniques in linear regression: A review. Optimization in Stat. In: TIMS/Studies Management Sci, vol 19, pp 11–29
15. Pinar MČ, Elhedhli S (1998) A penalty continuation method for the $\ell\infty$ solution of overdetermined linear systems. BIT 38:127–150
16. Schryer N (1998) via G.H. Golub, Email: golub@sccm.stanford.edu
17. Watson GA (1980) Approximation theory and numerical methods. Wiley, New York

P

Packet Annealing

DAVID SHALLOWAY
Cornell University, Ithaca, USA

MSC2000: 92B05

Article Outline

Keywords and Phrases
See also
References

Keywords and Phrases

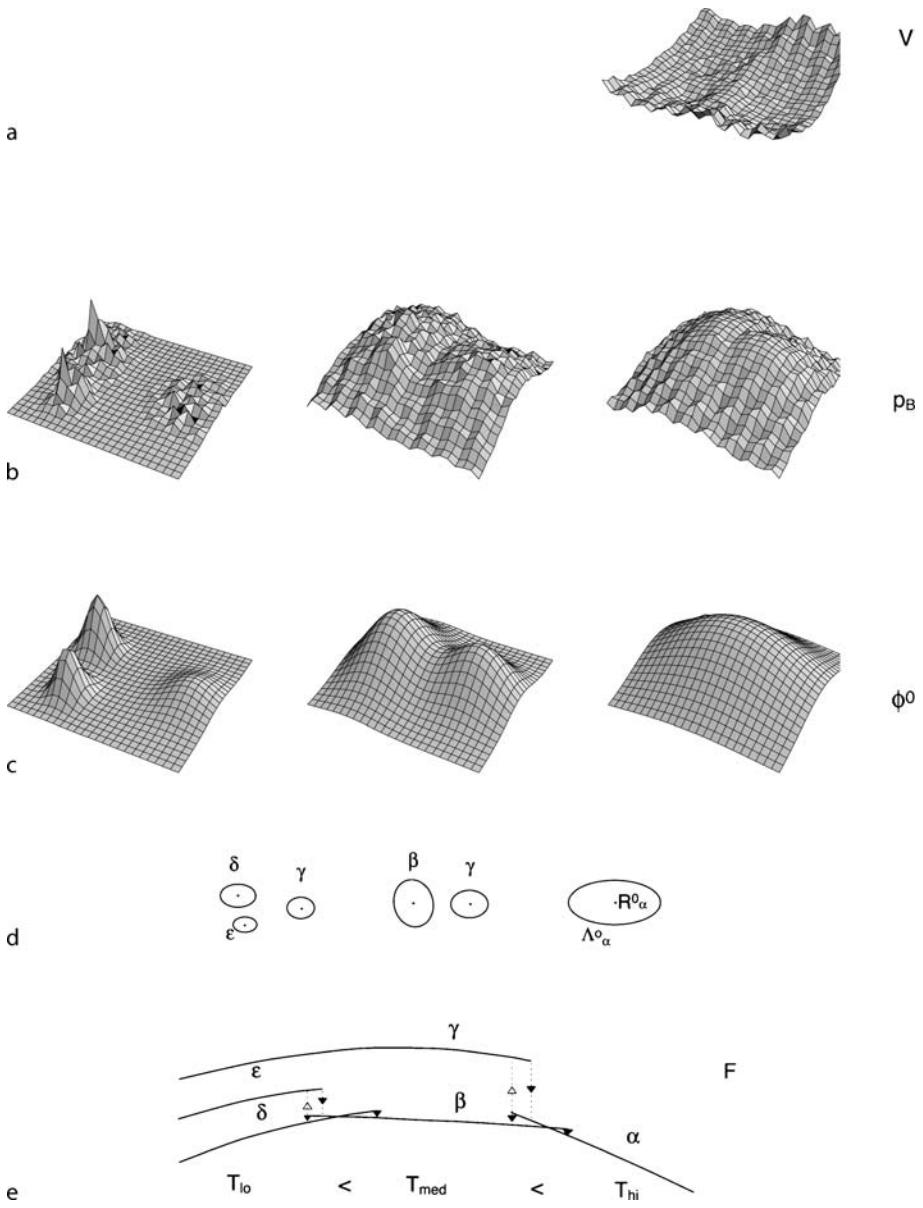
Optimization; Annealing; Hierarchical; Smoothing; Macrostate; Statistical mechanics; Trajectory diagrams

The packet annealing method [1,7,9] was motivated and developed specifically for thermodynamic global minimization problems such as those encountered in protein structure prediction [3], but it may be applicable to other global minimization problems as well. It uses the intrinsic variable-scale coarse-grained hierarchical structure [11] of the potential energy (objective function) landscape to guide a deterministic search to the global minimum. The method is similar to simulated annealing [4] in that it assumes that each point in the search space, parameterized by multidimensional vector R, corresponds to a conformation of a physical system whose motion is governed by the potential energy $V(R)$. According to statistical mechanics, at temperature T, the conformational probability distribution is the Gibbs–Boltzmann distribution [5],

$$p_B(T;R) = \frac{\exp\left[-\beta V(R)\right]}{\int \exp\left[-\beta V(R)\right] \mathrm{d}R} ,$$

where $\beta \equiv 1/k_B T$ is the inverse temperature (k_B is Boltzmann's constant which relates the energy and temperature scales). As $T \to 0$, all probability is concentrated in the vicinity of the global minimum, R_g, of V. Simulated annealing attempts to find R_g by following $p_B(\beta;R)$ as the system is cooled using the Metropolis [6] or other (e. g., molecular dynamics) search procedures to simultaneously search the entire space. In contrast, during cooling, packet annealing recursively subdivides conformation space into a sequence of compact macrostate regions which are separated from each other by potential energy barriers that are large compared to the current T. By this means, the space is subdivided into a growing number of smaller and smaller macrostates which are searched in parallel. The hierarchical relationships between the macrostates are represented in tree-like macrostate trajectory diagrams which describe the thermodynamic properties of the macrostates as functions of temperature. This allows computational effort to be focussed on the most promising subregions of conformation space. A key feature is that both the characteristic energetic and spatial scales of each macrostate are computed during the search process so that each macrostate can be searched using appropriately coarse-grained energetic and spatial variables. The nature of the linkage between these scales, the scaling properties of the system, determines the difficulty of finding the minimum.

For illustration, consider the problem of finding the global minimum of the two-dimensional potential shown in Fig. 1a. At a high temperature where $k_B T$ exceeds the internal energy barriers (right panel, Fig. 1b), the probability distribution is spread fairly smoothly over a compact region and can be coarsely approximated by a single Gaussian characteristic packet ϕ_α^0 (right panel, Fig. 1c) which is characterized by its



Packet Annealing, Figure 1
Packet annealing analysis

centroid (vector R^0_α) and root-mean-square size (tensor Λ^0_α) (Fig. 1d). As T is reduced to the point where the central energy barrier becomes larger than k_B T, the distribution bifurcates into two lobes β and γ which can be approximated by two child packets (center panels, Fig. 1b, Fig. 1c, Fig. 1d). As temperature is further decreased, β bifurcates into two children δ and ε (left panels). By this temperature it is evident that the peak within the ϕ^0_δ packet corresponds to the global minimum of V. While this could be found by a random search (as in simulated annealing), it is clear that it would be more efficient to use the hierarchically coarse-grained structure manifested by the characteristic packet analysis to direct the search.

- A model two-dimensional potential, $V(r_1, r_2)$.

- The corresponding Gibbs–Boltzmann probability distribution p_B at three temperatures $T_{hi} > T_{med} > T_{lo}$.
- Superposition of the Gaussian packets that are solutions of the characteristic packet equations at the three temperatures (a large number of characteristic packets, corresponding to the very small-scale fluctuations of V, will appear at lower temperatures).
- The characteristic packets are characterized by the positions of their center-of-masses (R^0) and by their root-mean-square fluctuation tensors (Λ^0), represented here by ellipses.
- Free-energy vs. temperature trajectory diagram for this temperature range. Solid lines represent metastable macrostate trajectories and dotted lines represent transitions. The discontinuities in the trajectories correspond to branch points at which packets bifurcate (from [1])

The characteristic packets are computed by solving a coupled set of self-consistent equations intentifying Gaussian distributions which are metastable in the stochastic physical system having potential V(R) [10]. This is equivalent to finding which locally minimize, the Hellinger distance, between p_B and [2]. The characteristic packets only coarsely approximate p_B; their main role is to determine the boundaries of the macrostate regions and thus dissect conformation space. Thermodynamic properties such as entropy and free energy can then be computed for each macrostate region and compactly represented in trajectory diagrams (Fig. 1e). All trajectories are tracked in this simple example, so finding the global minimum is guaranteed. This is not the case in more complicated problems where the number of branches exceeds computational capacity, and it is necessary to prune the trajectory diagram and pursue only a limited subset of branches. These are selected by a branch selection algorithm which uses the macrostate thermodynamic properties to predict those which are most likely to contain the global minimum. While the global minimum in Fig. 1 could be found by following only the trajectory having the lowest free-energy at each bifurcation, in general, multiple trajectories will have to be followed. The efficiency of the method will largely be determined by the ability of the branch selection algorithm to minimize the number of needed search trajectories. This is problem-dependent and requires the existence of underlying regularities within the class of potentials being studied. It has been suggested that such regularities do exist for protein potential functions [1]; this is an active area of research.

A key feature is that reduced accuracy is sufficient until the final (low) temperature is reached—it is only necessary that each trajectory remain within the catchment region of its macrostate. Within each macrostate region V(R) can be replaced with an approximation which has been spatially smoothed on a scale commensurate with the size of the macrostate. The use of smoothing is somewhat analogous to that occurring in the diffusion equation [8] and Gaussian density annealing [12] methods [11].

See also

- ▶ Bayesian Global Optimization
- ▶ Genetic Algorithms
- ▶ Genetic Algorithms for Protein Structure Prediction
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Lennard–Jones and Morse Clusters
- ▶ Global Optimization in Protein Folding
- ▶ Molecular Structure Determination: Convex Global Underestimation
- ▶ Monte-Carlo Simulated Annealing in Protein Folding
- ▶ Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach
- ▶ Phase Problem in X-ray Crystallography: Shake and Bake Approach
- ▶ Protein Folding: Generalized-Ensemble Algorithms
- ▶ Random Search Methods
- ▶ Simulated Annealing
- ▶ Simulated Annealing Methods in Protein Folding
- ▶ Stochastic Global Optimization: Stopping Rules
- ▶ Stochastic Global Optimization: Two-Phase Methods

References

1. Church BW, Orešič M, Shalloway D (1996) Tracking metastable states to free-energy global minima. In: Pardalos PM, Shalloway D, Xue G (eds) Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding: DIMACS Workshop. DIMACS 23, Amer Math Soc, pp 41–64

2. Church BW, Ulitsky A, Shalloway D (1999) Macrostate dissection of thermodynamic monte carlo integrals. Adv Chem Phys 105:273–310
3. Church BW, Shalloway D (2001) Top-down free-energy minimization on protein potential energy landscapes. Proc Natl Acad Sci USA 98:6098–6103
4. Kirkpatrick S Jr, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
5. Landau LD, Lifshitz EM (1969) Statistical physics. Addison-Wesley, Reading
6. Metropolis N, Metropolis AW, Rosenbluth MN, Teller AH, Teller E (1953) Equations of state calculations by fast computing machines. J Chem Phys 21:1087–1092
7. Orešič M, Shalloway D (1994) Hierarchical characterization of energy landscapes using Gaussian packet states. J Chem Phys 101:9844–9857
8. Piela L, Kostrowicki J, Scheraga HA (1989) The multiple minima problem in the conformational analysis of molecules. Deformation of the potential energy hypersurface by the diffusion equation method. J Phys Chem 93(8):3339–3348
9. Shalloway D (1992) Application of the renormalization group to deterministic global minimization of molecular conformation energy functions. J Glob Optim 2:281–311
10. Shalloway D (1996) Macrostates of classical stocastic systems. J Chem Phys 105(22):9986–10007
11. Shalloway D (1997) Variable-scale coarse-graining in macromolecular global optimization. In: Biegler LT, Coleman TF, Conn AR, Santosa FN (eds) Large Scale Optimization with Applications to Inverse Problems, Optimal Control and Design, and Molecular and Structural Optimization. Springer, New York, pp 135–161
12. Straub JE (1996) Optimization techniques with applications to proteins. In: Elber R (ed) New Developments in Theoretical Studies of Proteins. World Sci, Singapore, pp 137–196

Parallel Computing: Complexity Classes

MARTIN FÜRER
Pennsylvania State University,
University Park, USA

MSC2000: 68Q05, 68Q15, 03D15

Article Outline

Keywords
Unrestricted Parallelism
Restricted Amount of Hardware, Circuits
Restricted Amount of Hardware, PRAMs
BSP and LogP
See also
References

Keywords

Parallel machines; Complexity classes; PRAM; NC; Bulk synchronous parallel model; BSP model

Unrestricted Parallelism

For simplicity, complexity theory usually focuses on decision problems. A *complexity class* consists of all decision problems solvable with given resource bounds.

The *depth of a Boolean circuit* is the most obvious measure of parallel time, because all gates can work in parallel. Problems solvable by uniform families of circuits of depth $O(\log n)$ have long been recognized as an important complexity class. Here n always denotes the length of an input, and it is assumed that the family contains one circuit for each n.

Complexity theory for parallel computations really started with the observation of the strong correspondence between parallel time and Turing machine space complexity discovered by A. Borodin [1]. In terms of circuit complexity, this means that not only is the depth of a Boolean circuit equal to the parallel time, but for every narrow circuit there is an equivalent shallow circuit. The close relationship between space in sequential machines and time in parallel machines is known as the *parallel computation thesis* [8].

This relationship between space and time complexity classes has been strengthened by the introduction of *alternating Turing machines* [2] as a computing model with unbounded parallelism. These machines have universal states in addition to the existential states present in nondeterministic machines. For universal configurations, all successor configurations must lead to acceptance, while for existential configurations at least one successor must lead to acceptance.

For time functions $T(n)$ and space functions $S(n)$, alternating Turing machines define the complexity classes ATIME($T(n)$) and ASPACE($S(n)$). With ALOGSPACE = ASPACE($\log n$) and APTIME = $\cup_c$ ATIME(n^c), the correspondence between sequential and parallel complexity classes is ALOGSPACE = P (polynomial time) and APTIME = *PSPACE* (polyno-

mial space). Similar relations hold at higher levels of the time and space hierarchy.

Restricted Amount of Hardware, Circuits

More practical complexity classes (for parallel computing) simultaneously restrict the parallel time and the amount of hardware used. S.A. Cook [3] has proposed to consider complexity classes obtained by simultaneously bounding the time and space of Turing machines. The most important of these classes has later been named SC (Steve's class). SC^k consists of all decision problems that can be solved simultaneously in polynomial time and $O(\log^k n)$ space. Finally, SC is defined as $SC = \cup_k SC^k$.

Similarly, N. Pippenger has studied the complexity classes (later called NC^k) consisting of all problems that can be solved by uniform families of Boolean circuits such that the size of the nth circuit is polynomial in n and its depth is $O(\log^k n)$ (see [9]). Again, NC is defined as $NC = \cup_k NC^k$. The classes NC and NC^k are much more widely used than the classes SC and SC^k, because NC directly measures the parallel time. Informally, it consists of all decision problems that can be solved very fast (in *polylogarithmic time*) on a parallel machine with only a moderate (polynomial) amount of hardware. It is an open problem (as of 2000) whether NC = SC [1].

If Boolean circuits are allowed to have 'AND' and 'OR' gates with an arbitrary number of inputs, then the restriction to polynomial size and depth $O(\log^k n)$ defines the class AC^k. It is easy to see that $NC^k \subseteq AC^k \subseteq NC^{k+1}$.

By allowing randomized computations, NC extends to RNC. Formally, these circuits may contain gates (without inputs) producing independent random outputs 0 and 1 (representing false or true) with equal probability. Such a randomized circuit solves a problem, if it rejects every negative instance, while accepting every positive instance with probability at least 1/2. Deciding whether a graph has a perfect matching is in RNC, but is not known to be in NC.

For good reasons, practical parallel computing has initially focused on utilizing the obvious parallelism present in many scientific computations due to the presence of matrices, vectors or simple loops. At the same time, theoretical research has tried to classify problems according to their efficient solvability by parallel algorithms, the central question being whether a problem is in NC.

Restricted Amount of Hardware, PRAMs

To show membership in NC, and for designing any parallel algorithm while abstracting from all communication issues, the PRAM model of parallel computing has been defined. A *PRAM* [6,13] consists of many cooperating processors, each being a random access machines (*RAM*, [4]). Each processor can do local computations consisting of additions, subtractions, shifts, conditional and unconditional jumps, as well as indirect addressing. Arbitrary long shifts and multiplications of large numbers are not allowed in one step, as these operations would make a single processor impractically powerful.

The processors in a PRAM are synchronized, and communication between processors is accomplished by a global memory. The intention is not to suggest that synchronized operation and global memory are easy to realize, but that it simplifies programming of a parallel machine. Simulation of global memory (by local memory and communication) can be handled by a compiler or even directly implemented in hardware ([10]). Several flavors of PRAM have been defined.

- In an *EREW PRAM*, different processors are not allowed to access the same memory location simultaneously.
- In a *CREW PRAM*, only writing to memory is so restricted.
- In a *CRCW PRAM*, simultaneous reading and writing is allowed.

There are three kinds of CRCW PRAMs with different ways to handle concurrent writing.

- In the COMMON model, all processors writing to the same location simultaneously, are required to write the same data.
- In the ARBITRARY model, an arbitrary processor succeeds (i. e., it writes last).
- In the PRIORITY model, the lowest numbered processor succeeds with writing.

Interesting complexity classes are obtained by restricting computations to be *uniform*. Technically, it is assumed that a *logspace Turing machine* produces the programs for all processors. Then the complexity

classes EREW^k, CREW^k and CRCW^k are defined as the classes of problems solved by a uniform PRAM of the given type in time $O(\log^k n)$ with polynomially many processors. (For CRCW^k the most powerful type PRIORITY is assumed.) It is known that $NC^k \subseteq EREW^k \subseteq CREW^k \subseteq CRCW^k = AC^k \subseteq NC^{k+1}$ (see [9]).

Besides minimizing the parallel time, a key issue in parallel computing is to minimize the *work*, which is the product of the parallel time with the number of processors. An algorithm is said to be *optimal* if the ratio between the work and the optimal sequential time is bounded by a constant, and it is said to be *efficient* if that ratio is bounded by a polylogarithmic factor.

While the PRAM model abstracts from *communication* issues, another branch of research has focused exactly on communication for various interesting arrangements of processors, like arrays, trees and hypercubes [11].

Theoretically, it has been shown that PRAM algorithms can be implemented by some fixed networks of communicating processors with very little loss of speed [7,14]. Yet, it has been felt that the constant factors in the speed loss could be too big to warrant the restriction of the algorithm designer to simply program a PRAM and let the compiler handle all communication.

BSP and LogP

Several parallel machine models have been proposed with two goals in mind.

- Programs (like PRAM programs) are portable to various types of physical parallel machines.
- The programmer (unlike a PRAM programmer) has some control over the communication between processors in order to obtain high efficiency.

These two somewhat contradicting goals can be achieved by letting the programmer choose the source and target, but not the path and detailed timing of each message. The two most influential such models are the *BSP model* (*bulk synchronous parallel model*) of L.G. Valiant [15,16] and the *LogP model* [5].

The BSP model is called a *bridging model*, because it is intended to bridge the gap between hardware and software for parallel computing, as the von Neumann model did for sequential computing. Thanks to compilers, the programmer does not have to know too many details of the actual machine.

The BSP model performs a sequence of supersteps, each consisting of three phases, local computation, global communication and a barrier synchronization. The latter is a global check that all components have finished a superstep. The BSP model is characterized by two parameters, L and g. The time unit is the duration of a local operation. The periodicity parameter L measures the length of a superstep, while g measures the length of a global operation.

The LogP model has basically the same goals as the BSP model, but intends to give the programmer slightly more flexibility to address important performance issues without having to deal with unnecessary details. The LogP model is characterized by four parameters, the latency L, the overhead per message o, the time gap between messages g (for each processor) and the ratio P between the number of processors and the number of memory modules. This allows each parallel machine to be characterized by only a few parameters. The algorithm designer can prescribe different methods for different ranges of the parameters. Then such an algorithm can compile efficiently on various parallel machines.

The BSP and LogP models, as well as the *QSM* (*queueing shared-memory model*), are actually quite closely related as indicated by various simulation results (e. g., [12]).

See also

- ▶ Asynchronous Distributed Optimization Algorithms
- ▶ Automatic Differentiation: Parallel Computation
- ▶ Complexity Classes in Optimization
- ▶ Complexity of Degeneracy
- ▶ Complexity of Gradients, Jacobians, and Hessians
- ▶ Complexity Theory
- ▶ Complexity Theory: Quadratic Programming
- ▶ Computational Complexity Theory
- ▶ Fractional Combinatorial Optimization
- ▶ Heuristic Search
- ▶ Information-Based Complexity and Information-Based Optimization
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Kolmogorov Complexity

- ▶ Load Balancing for Parallel Optimization Techniques
- ▶ Mixed Integer Nonlinear Programming
- ▶ Parallel Computing: Models
- ▶ Parallel Heuristic Search
- ▶ Stochastic Network Problems: Massively Parallel Solution

References

1. Borodin A (1977) On relating time and space to size and depth. SIAM J Comput 6(4):733–744
2. Chandra AK, Kozen DC, Stockmeyer LJ (1981) Alternation. J ACM 28(1):114–133
3. Cook SA (1981) Towards a complexity theory of synchronous parallel computation. L'Enseign Math 27:99–124
4. Cook SA, Reckhow RA (1973) Time-bounded random access machines. J Comput Syst Sci 7:354–375
5. Culler DE, Karp RM, Patterson D, Sahay A, Santos EE, Schauser KE, Subramonian R, von Eicken T (1996) LogP, a practical model of parallel computation. Comm ACM 39(11):78–85
6. Fortune S, Wyllie J (1978) Parallelism in random access machines. In: Conf. Record Tenth Annual ACM Symp. Theory of Computing, pp 114–118
7. Galil Z, Paul WJ (1983) An efficient general-purpose parallel computer. J ACM 30(2):360–387
8. Goldschlager LM (1982) A universal interconnection pattern for parallel computers. J ACM 29(4):1073–1086
9. Karp RM, Ramachandran V (1990) Parallel algorithms for shared-memory machines. Handbook Theoret. Computer Sci. A: Algorithms and Complexity. In: van Leeuwen J (eds) MIT, Elsevier, pp 869–941
10. Keller J, Paul WJ, Scheerer D (1994) Realization of PRAMs: Processor design. 8th Internat. Workshop on Distributed Algorithms, WDAG. In: Lecture Notes Computer Sci, vol 857. Springer, Berlin, pp 17–27
11. Leighton FT (1992) Introduction to parallel algorithms and architectures: Arrays – Trees – Hypercubes. Morgan Kaufmann, San Mateo
12. Ramachandran V, Grayson B, Dahlin M (1999) Emulations between QSM, BSP, and LogP: A framework for general-purpose parallel algorithm design. In: 10th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp S957–958
13. Stockmeyer L, Vishkin U (1984) Simulation of parallel random access machines by circuits. SIAM J Comput 13(2):409–422
14. Valiant LG (1982) A scheme for fast parallel communication. SIAM J Comput 11(2):350–361
15. Valiant LG (1990) A bridging model for parallel computation. Comm ACM 33(8):103–111
16. Valiant LG (1990) General purpose parallel architectures. In: van Leeuwen J (ed) Algorithms and Complexity. Handbook Theoret. Computer Sci., vol A. MIT, Elsevier, pp 943–971

Parallel Computing: Models

Afonso Ferreira
CNRS-I3S-INRIA, INRIA Sophia Antipolis, Sophia–Antipolis, France

MSC2000: 65K05, 65Y05

Article Outline

Keywords
Models
Atomic Models
- PRAM
- Atomic GRAM
- The 2-dimensional grid
- The Hypercube

Bulk Models
- Bulk GRAM
- BSP
- CGM
- LogP

See also
References

Keywords

Models for parallel computing; Parallel algorithm; Algorithm design; BSP; CGM; PRAM; Grid; Hypercube; Bulk synchronous parallel computer; Coarse grained multicomputer; Parallel random access machine; PC clusters; Shared memory parallel machines; Distributed memory parallel machines

The rapid growth and large availability of high speed networking have brought *high performance computing systems* (HPCS) to the reach of many people wishing to process very large data and difficult problems as fast as possible. Such systems evolve at an incredible pace and different machines with new architectures, programming models and paradigms, and computation granularity are proposed every year. For instance, the use of personal computers (PC) clusters interconnected by high performance local networks with raw throughput close to 1Gb/s and latency smaller than

10 μs yielded, in the late 1990s, parallel systems whose computing power was close to or even better than the one of super-computers of the middle of the 1980s, for a hundredth to a tenth of their nominal price. Their local networks were either realized with off-the-shelf ware (e. g. Myrinet and Fast Ethernet), or application-driven devices, in which case additional functionalities were built-in, mainly at the memory access level. Such clusters supported the Linux operational system, among others, and offered a system level virtualization for user-friendly programming environments like multi-threading, communication libraries, automatic load-balance, I/O systems, etc.

When using an HPCS to solve computationally intensive problems, the first aspect to be understood is the level of concurrency existing in the problem, i. e., which tasks can be executed simultaneously and which cannot. For instance, there are cases where a problem is not adapted at all to the parallel setting and only very small benefit from parallelism can be obtained. Therefore, the choice of a model for *parallel computing* is of primeval importance.

Models

Parallel computing models can be roughly divided into those which implicitly hide or assume the value of its parameters and those which explicitly set these values, as follows. The machine *size* can be defined by a parameter p or be (implicitly) connected to the problem input size n. The *topology* of the communication network is either explicit (ring, grid, hypercube, etc.) and the processors communicate only with neighbors, or hidden, i. e., the processors can communicate with any other processor through a high speed interconnection medium. The *communication costs* may depend on the size of the messages sent and on the distance they travel, or be fixed by message or even by set of messages sent. Finally, although *parallel computers* are asynchronous, and some models take this into consideration, most of the existing models suppose a synchronized *mode*, with some kind of barrier of synchronization – even light ones, based on *rendez-vous* communications. In the following, we briefly describe the most important models for parallel computing and the way they deal with the parameters above. For more details, we refer to [11,12,14,22,23,24,29].

Atomic Models

In these models, a parallel machine consists of a large set of atomic processors communicating by the exchange of atomic messages at a constant cost per message. The number of processors is usually taken as a function of n, the input size of the problem to be solved.

PRAM

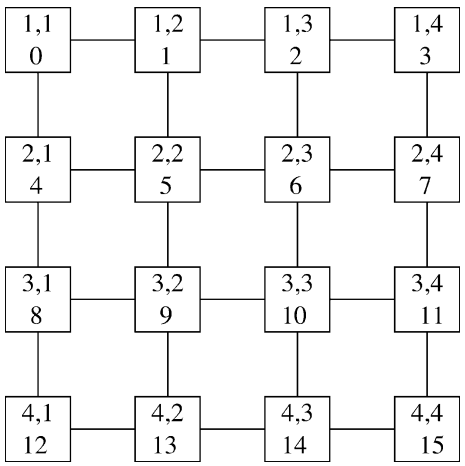
The shared memory model known as *parallel random access machine* (PRAM) [19,22,23] is the best known *model for parallel algorithm design*, because of its high abstraction level. In this model, a parallel machine consists of a large set of atomic processors. They communicate through a shared memory, where any position can be accessed in constant time. In order to design an algorithm we just have to describe a sequence of synchronous parallel operations executed by the processors on the shared memory, without worrying about scheduling the communications between the processors. This model is perfectly adapted to determine the level of parallelism inherent to a problem, since all parameters are implicit or hidden [10,21]. Unfortunately, however, only small shared-memory parallel computers have been built so far, because of technological constraints regarding the concurrent access to the memory in constant time, when the number of processors is large.

Atomic GRAM

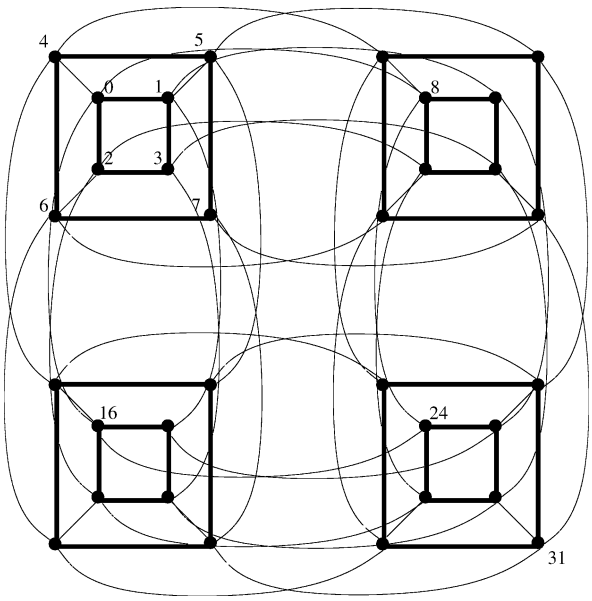
One way to solve the problem raised by the fully connected shared memory was to consider distributed memory machines and make explicit the topology through which the atomic processors communicate. Among the most used *GRAMs* (where G stands for the graph defining the topology of the communication network), we find grids and hypercubes [11,24], whose definitions follow.

The 2-dimensional grid

The *2-dimensional grid* (called grid in the remainder) of size N is composed of N processors $PE_{i,j}$, $1 \leq i, j \leq \sqrt{N}$, such that processor $PE_{i,j}$ is linked to processors $PE_{i-1,j}$, $PE_{i+1,j}$, $PE_{i,j-1}$, $PE_{i,j+1}$, for $2 \leq i, j \leq \sqrt{N} - 1$. The grid has degree 4, diameter $O(\sqrt{N})$ (a longest



Parallel Computing: Models, Figure 1
A 4 × 4 grid



Parallel Computing: Models, Figure 2
A Hypercube *H*(5) with 32 nodes and diameter 5. We can see its decomposition in four *H*(3), in bold

path existing between processors $PE_{1,1}$ and $PE_{\sqrt{N},\sqrt{N}}$, for instance), and bisection width $\sqrt{N}$ (which can be obtained by deleting all links between processors $PE_{i,\frac{\sqrt{N}}{2}}$ and $PE_{i,\frac{\sqrt{N}}{2}+1}$, for $1 \leq i \leq \sqrt{N}$).

Grids can be generalized in two ways. The first one is to consider d-dimensional grids which are defined on N processors $PE_{i_1,\ldots,i_d}$, $1 \leq i_k \leq N^{1\backslash d}$, with links between $PE_{i_1,\ldots,i_k,\ldots,i_d}$ and $PE_{i_1,\ldots,i_k\pm1,\ldots,i_d}$, for $1 \leq k \leq d$. The second classic generalization is to add links in the 2-dimensional grid between processors $PE_{1,j}$ and $PE_{\sqrt{N},j}$ and between $PE_{i,1}$ and $PE_{i,\sqrt{N}}$ for $1 \leq i, j \leq \sqrt{N}$. The obtained structure is called a *2-dimensional torus*. As 2-dimensional grids, 2-dimensional tori can be generalized to *d-dimensional tori*.

Processors of a grid are usually numbered in a row major order by using a unique index. $PE_{i,j}$ is then denoted by $PE_{(i-1)\sqrt{N}+j-1}$, that is, processors are linearly numbered from left to right and from top to bottom, with indices in the range $0, \ldots, N-1$.

The Hypercube

An interconnection network with the topology of a *d-dimensional hypercube*, denoted $H(d)$, is composed of $N = 2^d$ processors, labeled from 0 to $N-1$, and $dN\backslash 2$ communication links. Let $(i)_2$ be the binary string representing i and i_k denote the k-th digit, from right to left, in $(i)_2$. Then, the neighbors of PE_i are all PE_j such that $(i)_2$ and $(j)_2$ differ in exactly one bit position, say k, $0 \leq k < d$, implying that its degree is d. In this case, we say that PE_i and PE_j are neighbors along dimension k. It is not difficult to see that the maximum distance in a hypercube is given by those pairs of processors whose binary string differ in all d positions, implying that its diameter is $d = \log N$. Finally, the bisection width of a hypercube is $N/2$.

Figure 2 shows that processors are the vertices of a hypercube of dimension d, each connected to d neighbors. Notice further that, for instance, PE_0 and PE_4 are neighbors along dimension 2 in any $d > 2$-dimensional hypercube.

A hypercube $H(d)$ can be decomposed in d different ways into two copies of $H(d-1)$, with $N/2$ edges connecting them. In order to find one such decomposition, it suffices to fix any one bit position in the processors' addresses, say position k, $0 \leq k < d$. Then, the two copies of $H(d-1)$ are composed of the vertices $i_{d-1}\cdots i_{k+1}\ 0\ i_{k-1}\cdots i_0$ and $i_{d-1}\cdots i_{k+1}\ 1\ i_{k-1}\cdots i_0$, respectively. It is interesting to notice that one can use these decompositions in order to implement divide and conquer algorithms in hypercubes.

Let us further remark that the hypercube is vertex and edge symmetric, making it easy to use. Very informally, we could say that, as far as the neighbors are con-

cerned, the hypercube looks the same from every node. Note that this is also true for the torus, but false for the grid.

Bulk Models

Parallel algorithm designers who dealt with real problems to solve, soon discovered that they needed not only to determine a problem's inherent parallelism, but also to write algorithms that could be efficiently implemented. Hence, atomic models were not of help, since parallel computers cannot expand endlessly in order to reflect the size of the problem at hand, and distributed memory bulk models appeared.

In opposition to the atomic models, the number of processors, p, becomes a parameter and each processor is supposed to be able to hold n/p data. A *parallel algorithm* is then a sequence of *supersteps*, composed of local computations followed by a communication round, where messages are exchanged among the processors. Notice that such an algorithm, where the number of supersteps is small and independent of n, will be efficient in any HPCS, provided that the communication procedures are implemented in an efficient manner, what is usually the case.

Bulk GRAM

One difference between bulk and atomic GRAMs is the number of processors, that here is set to an independent value p. In each communication phase, processors can then send a message of varying length to their neighbors. The cost of these communications is also modeled, according to a value proportional to the initialization of the communication channel plus the time spent by the message to arrive at the destination [5,11,20].

BSP

The BSP model (for *bulk synchronous parallel*) [28], uses slackness in the number of processors and memory mapping via hash functions to hide communication latency and provide for the efficient execution of atomic PRAM algorithms on existing hardware.

An input of size n is distributed evenly across a p-processor parallel computer. In a single superstep each processor may send h and receive h' messages (called an (h, h')-relation) and then perform an internal computation on its internal memory cells using the messages it has just received. To avoid any conflicts that might be caused by asynchronies in the network (whose topology is left undefined) the messages sent out in a round t by some processor cannot depend upon any messages that the processor receives in round t (but, of course, they may depend upon messages received in round $t - 1$).

Communication costs depend on two parameters, namely the latency L and the throughput g. The cost of an (h, h')-relation performed by a processor is then $L + \max(h, h')\, g$, and the total cost of a communication round is $\max_i = 1, \ldots, pL + \max(h_i, h_i')\, g$ where h_i (respectively, h_i') represents the amount of data sent (respectively, received) by processor i. Precise models of parallel computers can be obtained by assigning realistic values to L and g. More detailed BSP models and algorithms can be found in [2,3,27,29].

CGM

The *coarse grained multicomputer* model, or CGM(n, p) for short, was introduced in [7]. The CGM(n, p) is a BSP model consisting of a set of p processors with $O(n/p)$ local memory each, in which each superstep has $h = h' = O(n/p)$.

The originality of CGM stems from its cost model. The algorithm designer will no longer try to minimize the overall amount of data exchanged, as in the BSP, but rather design algorithms with a small number of supersteps, independent of the input size n. As a consequence, these algorithms will be very efficient in practice. Ideally, algorithms should run for a constant number of supersteps, as it is the case for sorting [27] and many problems in image processing [17,18], computational geometry [13,15], and optimization [8,16]. On the other hand, graph problems seem to be somewhat more complex, some of them requiring $O(\log p)$ communication rounds to be solved [4].

LogP

This model uses the BSP model as a starting point and focuses on the technological trend from fine-grained parallel machines towards coarse-grained systems, advocating portable parallel algorithm design [6]. It characterizes a *distributed memory parallel computer* by four parameters (whence its name), as follows.

- L: the latency of a small message being communicated from user-space to user-space.

- *o*: the overhead incurred in a communication. It is defined as the time interval during which a processor cannot do anything else because it is communicating.
- *g*: the gap between two consecutive message transmissions (or receptions) by one processor. It is defined as the inverse of the bandwidth of the communication processing element.
- *P*: the number of processors.

Memory resources are considered finite. Hence, only L/g messages can be at the same time in the network. The cost to communicate an elementary packet between two processors is $L+ 2o$. If a reception acknowledgment is required, this cost becomes $2L+ 2o$. Designing algorithms in LogP may become an elaborate task, because of the several parameters involved, and the asynchronous character of the models [1,9,26]. It is also very interesting to notice that work on LogP algorithms are very similar to those on bulk GRAM algorithms, done some 10 years earlier [5,25].

See also

- ► Asynchronous Distributed Optimization Algorithms
- ► Automatic Differentiation: Parallel Computation
- ► Heuristic Search
- ► Interval Analysis: Parallel Methods for Global Optimization
- ► Load Balancing for Parallel Optimization Techniques
- ► Parallel Computing: Complexity Classes
- ► Parallel Heuristic Search
- ► Stochastic Network Problems: Massively Parallel Solution

References

1. Alexandrov A, Ionescu M, Schauser KE, Scheiman C In: Proc. 7th ACM Symp. Parallel Algorithms and Architectures. LogGP: Incorporating long messages into the LogP model: One step closer towards a realistic model for parallel computation
2. Baumker A, Dittrich W, Meyer auf der Heide F (1995) Truly efficient parallel algorithms: c-optimal multisearch for an extension of the BSP model. In: Proc. Europ. Symp. Algorithms, pp 17–30
3. Baumker A, Meyer auf der Heide F (1997) Communications in efficient parallel searching. In: Bilardi G, Ferreira A, Luling R, Rolim J (eds) 4th Internat. Symp. Solving Irregularly Structured Problems in Parallel, IRREGULAR'97. Lecture Notes Computer Sci., vol 1253. Springer, Berlin, pp 255–272
4. Caceres E, Dehne F, Ferreira A, Flocchini P, Rieping I, Roncato A, Santoro N, Song S (1997) Efficient parallel graph algorithms for coarse grained multicomputers and BSP. In: Degano P, Gorrieri R, Marchetti-Spaccamela A (eds) Proc ICALP'97. Lecture Notes Computer Sci., vol 1256. Springer, Berlin, pp 390–400
5. Cosnard M, Trystram D (1995) Parallel algorithms and architectures. Internat. Thompson Computer Press, Boston, MA
6. Culler DE, Karp RM, Patterson DA, Sahay A, Shacuser KE, Santos E, Subramonian R, von Eicken T (1993) LogP: Towards a realistic model of parallel computation. In: Proc. 4th ACM SIGPLAN Symp. on Princ. and Practice of Parallel Programming, pp 1–12
7. Dehne F, Fabri A, Rau-Chaplin A (1993) Scalable parallel geometric algorithms for coarse grained multicomputers. In: Proc. 9th ACM Symp. on Computational Geom., pp 298–307
8. Diallo M, Ferreira A, Rau-Chaplin A, Ueda S (1999) Scalable 2d convex hull and triangulation algorithms for coarse-grained multicomputers. J Parallel Distributed Comput 56(1):47–70
9. Dusseau AC, Culler DE, Schauser KE, Martin R (1996) Fast parallel sorting under LogP: Experience with the CM5. IEEE Trans Parallel and Distributed Systems
10. Eppstein D, Galil Z (1988) Parallel algorithmic techniques for combinatorial computation. Ann Rev Comput Sci 3:233–283
11. Ferreira A (1996) Parallel and communications algorithms for hypercube multiprocessors. Handbook Parallel and Distributed Computing. In: Zomaya A (ed) McGraw-Hill, New York, pp 568–589
12. Ferreira A, Fraigniaud P (Dec. 1997) Impact du modèle sur l'algorithmique. C.R. Ecole dètè CNRS sur la conception et mise en œ uvre d'applications parallèles irrègulières de grande taille. pp 383–402
13. Ferreira A, Kenyon C, Rau-Chaplin A, éda S (1997) d-Dimensional range search on multicomputers. In: Proc. 11th IEEE Internat. Parallel Processing Symp. IEEE Computer Soc Press, New York, pp 616–620
14. Ferreira A, Morvan M (1996) Models for parallel algorithm design: An introduction. In: Migdalas A, Pardalos PM, Storoy S (eds) Parallel Computing in Optimization. Kluwer, Dordrecht, pp 1–26
15. Ferreira A, Rau-Chaplin A, Ubéda S (1995) Scalable 2d convex hull and triangulation algorithms for coarse-grained multicomputers. In: Proc. 7th IEEE Symp. Parallel and Distributed Processing – SPDP, October 1995. IEEE Computer Soc Press, New York, pp 561–569
16. Ferreira A, Robson JM (1996) Fast and scalable parallel algorithms for knapsack and similar problems. J Parallel Distributed Comput 39(1):1–13

17. Ferreira A, Ubéda S (1994) Ultra-fast parallel contour tracking, with applications to thinning. Pattern Recognition 27(7):867–878
18. Ferreira A, Ubéda S (1995) Parallel complexity of the medial axis transform. In: Proc. IEEE Internat. Conf. Image Processing – ICIP, 1995. IEEE, New York, pp 105–107
19. Fortune S, Wyllie J (1978) Parallelism in random access machines. In: Proc. 10th ACM Symp. Theory of Computing, pp 114–118
20. Fraigniaud P, Lazard E (1994) Methods and problems of communication in usual networks. Discrete Appl Math 53:79–133
21. Gibbons A, Rytter W (1988) Efficient parallel algorithms. Cambridge University Press, Cambridge
22. Ja'Ja' J (1992) An introduction to parallel algorithms. Addison-Wesley, Reading
23. Karp R, Ramachandran V (1990) Parallel algorithms for shared-memory machines. In: van Leeuwen J (ed) Handbook Theoret. Computer Sci. MIT, Elsevier, pp 869–942
24. Leighton FT (1991) Introduction to parallel algorithms: Arrays, trees, hypercubes. Morgan Kaufmann, San Mateo
25. Robert Y (1990) The impact of vector and parallel architectures on Gaussian elimination algorithms. Manchester University Press, Manchester
26. Santos EE (1995) Solving triangular linear systems in parallel using substitution. In: Proc. 7th IEEE Symp. Parallel and Distributed Processing. IEEE Computer Soc Press, New York, pp 553–560
27. Shi H, Schaeffer J (1992) Parallel sorting by regular sampling. J Parallel Distributed Comput 14:361–372
28. Valiant L (1990) A bridging model for parallel computation. Comm ACM 38(8):103–111
29. Valiant L (1990) General purpose parallel architectures. In: van Leeuwen J (ed) Handbook Theoret. Computer Sci. MIT and Elsevier, pp 943–972

Parallel Heuristic Search

Alexander Reinefeld
ZIB Berlin, Berlin, Germany

MSC2000: 68W10, 68W15, 68R05, 68T20

Article Outline

Optimization – Parallel Search Algorithms – Heuristics
Task Partitioning
Parallel Window Search
Tree Partitioning
Stack Splitting
Search-Frontier Splitting
Work Distribution
Table Driven Search
See also
References

Optimization – Parallel Search Algorithms – Heuristics

Many applications in the field of artificial intelligence and operations research rely on heuristic search (cf. Heuristic search) as their primary solution method. Because these applications often spawn very large decision trees, the design of efficient parallel algorithms is of prime importance. Depending on the level of parallelism (fine-grained or coarse-grained) and the degree of parallelism (moderately or massively parallel), the techniques used in parallel heuristic search can be categorized into one of three classes:

- task partitioning
- parallel window search
- tree partitioning

Task Partitioning

The method of task partitioning (or operator partitioning) provides a fine-grained parallelism at the lowest level, the operator level. It speeds up the processing of individual nodes by performing repetitive steps like successor generation, node evaluation, and bookkeeping in parallel.

Task partitioning is especially popular in the field of computer chess, where special purpose hardware was built to assist in the move generation and board evaluation. The world computer chess champions Deep Thought [3] and Deep Blue, for example, employ special-purpose coprocessors for generating all moves of a position and for evaluating all 64 fields of a board in parallel. In the early years, hardware implementations were only feasible for simple operators that could be easily implemented in silicon. With the advent of programmable hardware accelerator chips like FPGAs (field programmable gate arrays), more complex parts of the tree search could be implemented in hardware, which dramatically improved the performance of computer chess programs, most notably that of the world champion Shredder.

Parallel Window Search

In parallel window search, all processors examine the entire tree, but each with another search bound. This method was originally developed to speed up game playing programs. As first suggested by Baudet [1], the total range of values in a game tree is subdivided into p nonoverlapping alpha-beta windows (where p is the number of processors), so that approximately one third is covered. The advantage is, that the processor having the true minimax value in its window will find it faster by virtue of starting with a narrow search window instead of using the full-width window. Even the unsuccessful processors are productive: They determine whether the minimax value lies below or above their assigned search interval. If the true minimax value does not lie in one of the initial p windows, the processors are re-scheduled to cover some of the remaining intervals. The iterative re-scheduling process is continued (like in binary search) until the final solution is found by one of the processors.

In some favorable cases, parallel window search may even cause superlinear speedup $s(p) > p$ when the minimax value is found early in the search. Moreover, the communication and synchronization overheads are quite low, allowing efficient execution on loosely coupled parallel systems. On the negative side, however, is the limited scalability of parallel window search. Even with an infinite number of processors, a maximal speedup of five or six can be attained in chess trees, because in the best case all $w^{\lceil d/2 \rceil} + w^{\lfloor d/2 \rfloor} - 1$ nodes of the minimal tree (with width w and depth d) must be examined by the 'successful' processor returning the minimax value. For this reason, parallel window search is suitable for small systems with up to three processors only.

Parallel window search can also be applied to single-agent searches like iterative-deepening A$*$ (IDA$*$) [7]. Here, different processors are used to search the entire tree up to different thresholds (windows), hoping that one of them would find a solution. If not, a global administration scheme determines the next larger threshold, and the node expansion starts over again. Note, that this scheme works only in applications where the increments in the threshold are known a priori. In the 15-puzzle, for example, the first processor's threshold would be set to the heuristic estimate $h(n)$ of the initial position n, the next processor gets the threshold $h(n) + 2$, and so on.

As in adversary game tree search, the maximum scalability is limited to the maximum number of iterations. Because the iterations with consecutive thresholds are not explored in sequential order, the first solution may not be optimal. Optimality can, however, be guaranteed by completing all shallower iterations than that of the best solution found so far. The better the node ordering, the faster the solution speed. In the extreme case, superlinear speedup can be achieved when the solution is found early in some 'left' part of the tree.

Because of its nonoptimality and its limited scalability, parallel window search is mainly used for quickly determining a (possibly suboptimal) solution which is then improved by other means [7].

Tree Partitioning

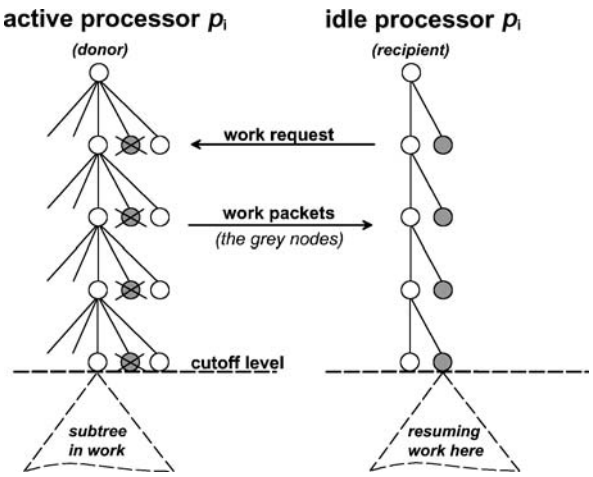
In tree partitioning, the total search space is split into smaller parts (subtrees) for simultaneous exploration by different processors. Once the subtrees have been distributed among the processors, only little communication is necessary for broadcasting improved bound values and for termination detection.

Compared to the other two techniques, tree partitioning is the only parallelization strategy that allows (in principle) to employ an unlimited number of processors. This is especially true for the use of tree partitioning in parallel depth-first search (DFS). Here, applications have been successfully tested on massively parallel systems with more than a thousand processors.

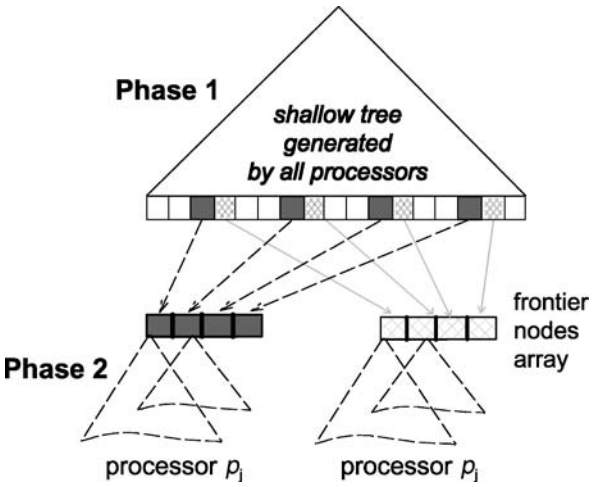
Because DFS trees tend to be highly irregular in practice, i. e. they exhibit varying branching degrees and search depths, static tree partitioning methods are insufficient to keep all processors busy during the whole computation. Two dynamic work partitioning methods have been proposed: stack splitting and search-frontier splitting.

Stack Splitting

In the stack splitting scheme [5], the local work is partitioned by splitting the donor's search stack into two, one of which is given to the requester. Care must be taken to select a suitable amount of work for shipment. On the one hand, the transferred work must be large enough to vindicate the communication costs, and on



Parallel Heuristic Search, Figure 1
Stack splitting



Parallel Heuristic Search, Figure 2
Search-frontier splitting

the other hand, not too much work should be sent to avoid thrashing effects. Several strategies for selecting nodes from a donor's stack have been proposed:

- Removing nodes from the tree levels near the root give coarse-grained work packets, which are likely to be splitted a second time during the search.
- Removing nodes near a cut-off level deep in the tree give fine-grained work packets, making it sometimes necessary to send more requests for obtaining enough work.
- Removing a 'vertical slice of nodes', one from each level, may be useful in trees with large branching factors and irregular search depths.

Stack-splitting works for simple DFS as well as for iterative DFS. The iterative-deepening search PIDA∗ (Parallel IDA∗, [4]) starts a new iteration with an increased cost-bound when all processors have finished their current iteration without success. The end of an iteration is determined by a barrier synchronization, e. g., the distributed termination detection algorithm of Dijkstra [2].

Search-Frontier Splitting

Rather than subdividing a processor's stack, search-frontier splitting initially generates a suitable number of 'work packets'. A work packet is a node from a 'search-frontier' in the tree, containing nodes n with the same cost value f(n). Search-frontier splitting has two phases:

- In an initialization phase, the nodes of a search-frontier are distributedly generated by a cost-bounded BFS or an iterative DFS. The bound is incrementally increased until at least p nodes with the same cost f(p) are generated and stored in the local memories. Each frontier forms the root of a subtree that represents an indivisible piece of work used in the second phase.
- In the main asynchronous search phase, each processor expands its own frontier nodes in DFS or DFBB fashion. When a processor gets idle, it sends a request for a work packet (unprocessed frontier node) to another processor.

The initialization phase is only short. A suitable amount of frontier nodes is generated to provide a fine work granularity. As before, the work packets must neither be too small (to pay off for the communication costs) nor too large (to avoid thrashing effects).

In practice, little load balancing is required, because the expansion of the local frontier nodes keep the processors busy for most of the time. Search-frontier splitting is applicable to DFS and iterative DFS. The iterative-deepening variant AIDA∗ (asynchronous IDA∗ [8]) starts a new iteration on the previously used frontier nodes. This has the effect of a self-improving load balancing scheme, because all subtrees tend to grow at approximately the same rate when searching to the next larger cost-bound. In practice, the communication overhead decreases with increasing search time.

Work Distribution

The work distribution is either initiated by the sender (donor) or by the receiver (recipient). In the sender initiated work distribution, the generation of subtasks is independent from idle processors. It is based on the rationale that the local work-load of any two processors in the system should not differ by a factor $\geq \delta$ [6]. Work packets are delivered to weakly loaded processors either on demand (when they become idle) or when imbalances occur. In conjunction with a node priority scheme, sender initiated work distribution may help avoiding speedup anomalies, but at the cost of a reduced execution speed.

The receiver initiated work distribution scheme, also known as work stealing, is more popular. Work requests are only sent by processors that became idle. When a donor has work to share, it returns a work packet, otherwise it notifies the requester accordingly. More sophisticated variants start issuing work requests as soon as there are fewer than δ work packets left on their stack, thereby reducing communication latency by overlapping communication and computation.

But which processor should best be addressed to obtain a work packet? The answer depends on the topology and characteristic of the interconnection network of the parallel system. Four receiver initiated work distribution methods have been extensively tested and analysed:

- ARR: In the asynchronous round robin strategy, each processor maintains a local variable target pointing to the next donor. Whenever a processor runs out of work, it sends a work request to the target processor and increments target (modulo p) thereafter.
- GRR: The global round robin method works similar to ARR, but with a global target variable instead. Whenever a processor runs out of work, it looks up the global target variable, increments it, and sends a work request to the assigned donor. Hence, four messages are sent for a single work request. Memory contention can be reduced by introducing a hierarchy of distributed target variables.
- RP: In random polling, idle processors send work requests to randomly chosen processors. Each donor is selected with the same probability.
- PF: In packet forwarding, unsuccessful work requests are not returned to the sender, but forwarded to the next neighbor. On ring topologies, work requests are forwarded until a processor responds with a work packet, or the message makes a full round through the ring, thereby indicating that no work is available.

Which of the work distribution schemes to choose depends on the interconnection topology of the target platform. On systems with a small communication diameter (e. g. hypercube), RP and ARR give best speedups, while PF performs better on systems with a large communication diameter (e. g. ring, torus) [4,8].

Table Driven Search

In many domains, application-specific heuristics and search enhancements introduce interdependencies between the generated states, making efficient parallelization a challenging task. Much of this information is stored in memory tables (e. g. transposition tables or refutation lists), which are essentially large caches in which the generated nodes and some book-keeping information are stored. Before expanding a new node, a table lookup is performed to check whether information on that node is available. This is especially useful when a node can have multiple predecessors, i. e. when the search space is a graph rather than a tree.

In parallel implementations the transposition table is partitioned among the processors. Each time a processor extends a new node, it first does a remote lookup by sending a message to the processor responsible for that portion of the table and waiting for the result. This results in a large time-overhead – even with asynchronous message passing. As an efficient alternative, transposition-driven work scheduling (TDS) [9] was introduced, which migrates the node to be expanded to the processor that may contain the corresponding transposition table entry. From that moment on, the receiving processor is responsible for the table lookup and, depending on the result, for extending the subtree of that node to the given search depth. TDS was introduced for parallel single-agent search (like IDA*) but can also be applied to multi-agent search.

See also

- ▶ Asynchronous Distributed Optimization Algorithms
- ▶ Automatic Differentiation: Parallel Computation
- ▶ Heuristic Search
- ▶ Load Balancing for Parallel Optimization Techniques
- ▶ Parallel Computing: Complexity Classes
- ▶ Parallel Computing: Models
- ▶ Stochastic Network Problems: Massively Parallel Solution

References

1. Baudet G (1978) The design and analysis of algorithms for asynchronous multiprocessors. PhD Thesis, Dept Comput Sci, Carnegie-Mellon Univ, Pittsburgh
2. Dijkstra E, Feijen WHJ, van Gasteren AJM (1983) Derivation of a termination detection algorithm for distributed computation. Inform Process Lett 16:217–219
3. Hsu FH (1990) Large scale parallelization of Alpha-Beta search: An algorithmic and architectural study with computer chess. PhD Dissertation, Carnegie-Mellon Univ, Pittsburgh
4. Kumar V, Grama A, Gupta A, Karypis G (1994) Introduction to parallel computing. Benjamin Cummings, Harlow
5. Kumar V, Rao V (1990) Scalable parallel formulations of depth-first search. In: Kumar V, Gopalakrishnan PS, Kanal L (eds) Parallel Algorithms for Machine Intelligence and Vision. Springer, Berlin, pp 1–41
6. Lüling R, Monien B (1992) Load balancing for distributed branch & bound algorithms. Int Par Process Symp, IPPS 92. IEEE Comput Soc Press, pp 543–549
7. Powley C, Korf RE (1991) Single-agent parallel window search. IEEE Trans Pattern Anal Machine Intell PAMI 13(5):466–477
8. Reinefeld A (1996) Parallel search in discrete optimization problems. J Simul Pract Theory 4:169–188
9. Romein JW, Bal HE, Schaeffer J, Plaat A (2002) A performance analysis of transposition-table-driven work scheduling in distributed search. IEEE Trans Parallel Distrib Syst 13(5):447–459

Parametric Global Optimization: Sensitivity

HUBERTUS TH. JONGEN[1], JAN-J. RÜCKMANN JONGEN[2], OLIVER STEIN[1]

[1] Department Math., Aachen University Technol., Aachen, Germany

[2] Department Math., Ilmenau University Technol., Ilmenau, Germany

MSC2000: 90C31, 90C34, 90C34

Article Outline

Keywords
The Reduction Ansatz
Global (Structural) Stability
  Topological Stability of $M[h, g]$
  Strong Stability of Stationary Points
  Global (Structural) Stability of SIP(f, h, g)
Generic Transitions
See also
References

Keywords

Structural stability; reduction Ansatz; Mangasarian–Fromovitz constraint qualification; Parametric programming; Genericity; Singularity; Trap-door point

For finite-dimensional optimization problems with infinitely many inequality constraints ('semi-infinite' optimization), the topological concept of global stability and its algebraic characterization are introduced in the following. Global stability refers to perturbations of the defining functions and their derivatives up to second order. Then, transitions between stable problems via one-parametric families are considered. Necessarily, certain unstable situations will be met and they will be discussed by means of the underlying singularities.

The considered optimization problems are of the following type:

$$(\mathrm{SIP}) \begin{cases} \min & f(x) \\ \text{s.t.} & x \in M[h, g], \end{cases}$$

where

$$M[h,g] = \left\{ x \in \mathbf{R}^n : \begin{array}{c} h_i(x) = 0,\ i \in A, \\ g(x,y) \geq 0 \\ \text{for all } y \in Y \end{array} \right\},$$

$$Y = \left\{ y \in \mathbf{R}^r : \begin{array}{c} u_i(y) = 0,\ i \in I, \\ v_j(y) \geq 0,\ j \in J \end{array} \right\},$$

$h = (h_i, i \in A)$, $u = (u_i, i \in I)$, $v = (v_j, j \in J)$, $|A| < n$, $|I| < r$, $|J| < \infty$. The index set Y is assumed to be *compact*. All defining functions are assumed to be of class C^k ($k \geq 2$), i. e., $f \in C^k(\mathbf{R}^n, \mathbf{R})$, $g \in C^k(\mathbf{R}^n \times \mathbf{R}^r, \mathbf{R})$, etc.

The Reduction Ansatz

Under certain assumptions it is possible to reduce (SIP) *locally* to an optimization problem with a *finite* number of inequality constraints. In fact, put $Y_0(x) = \{y \in Y: g(x, y) = 0\}$ and let $\overline{x} \in M[h, g]$. Then, all points of $Y_0(\overline{x})$ are *global* minima for $g(\overline{x}, \cdot)|_Y$. Consequently, if $Y_0(\overline{x}) = \{\overline{y}_1, \ldots, \overline{y}_p\}$ are 'nondegenerate' respectively 'strongly stable' (cf. [16]), then the implicit function theorem guarantees the existence of local C^1- respectively Lipschitz continuous, mappings $y_j(x)$, where $y_j(x)$ is a *local* minimum for $g(x, \cdot)|_Y$, $j = 1, \ldots,$ p. But then, in a neighborhood of $\overline{x}$, we can describe the feasible set $M[\, h, g]$ by means of the equalities $h_i = 0$, $i \in A$, and the *finite* number of C^2-, respectively $C^{1,1}$-, inequality constraints $\varphi_j(x) \geq 0$, $j = 1, \ldots, p$, where $\varphi_j(x) = g(x, y_j(x))$ (local *marginal function*) (cf. [4]).

Global (Structural) Stability

In this section, global (structural) stability of (SIP) will be introduced and characterized by using the concept of topological stability of the feasible set $M[\, h, g]$ and the concept of strong stability of stationary points for (SIP). Throughout this section assume that for all $y \in Y$ the gradients $Du_i(\overline{y})$, $i \in I$, $Dv_j(\overline{y})$, $j \in J_0(\overline{y})$ are linearly independent, where $J_0(\overline{y}) = \{j \in J:\ v_j(\overline{y}) = 0\}$.

Topological Stability of $M[\, h, g]$

The set $M[\, h, g]$ is called *topologically stable* with respect to the strong C^2-topology (briefly: C^2_s-stable) if there exists a C^2_s-neighborhood $\mathbf{U}$ of (h, g) in C^2 such that $M[\, h, g]$ is homeomorphic with $M[\widetilde{h}, \widetilde{g}]$ for every $(\widetilde{h}, \widetilde{g}) \in \mathbf{U}$. Here, a C^2_s-neighborhood of (h, g) is generated by perturbations of (h, g) and their derivatives up to second order which are controlled by a positive continuous function $\varepsilon(\cdot)$: $\mathbf{R}^n \to \mathbf{R}$ (for details, cf. [6]).

The topological stability of $M[\, h, g]$ is closely related with the *Mangasarian–Fromovitz constraint qualification* (MFCQ). The (MFCQ) is said to hold at $\overline{x} \in M[h, g]$ if the (row) vectors $Dh_i(\overline{x})$, $i \in A$ are linearly independent and if there exists a vector $\xi \in \mathbf{R}^n$ satisfying $Dh_i(\overline{x}) \cdot \xi = 0$, $i \in A$ and $D_x g(\overline{x}, \overline{y}) \cdot \xi > 0$ for all $\overline{y} \in Y_0(\overline{x})$. In [12] it is shown that topological stability of $M[\, h, g]$ can be characterized by an equivalent algebraic condition: If $M[\, h, g]$ is compact, then $M[\, h, g]$ is C^2_s-stable if and only if (MFCQ) holds at all $\overline{x} \in M[h, g]$. This equivalence was proved first in [2] for finite optimization problems (i. e. for those with finitely many constraints) and, then, it was generalized for semi-infinite problems in [12]. A generalization to noncompact feasible sets $M[\, h, g]$ under a stronger constraint qualification is presented in [9].

Strong Stability of Stationary Points

The strong stability of a stationary point for a finite problem and its equivalent algebraic characterization have been introduced in [16].

A point $\overline{x} \in M[h, g]$ is called a *stationary point* for (SIP) if there exist $\overline{y}_1, \ldots, \overline{y}_p \in Y_0(\overline{x})$, and reals $\overline{\beta}_i$, $i \in A$, $\overline{\gamma}_j \geq 0$, $j = 1, \ldots, p$ such that

$$Df(\overline{x}) = \sum_{i \in A} \overline{\beta}_i Dh_i(\overline{x}) + \sum_{j=1}^{p} \overline{\gamma}_j D_x g(\overline{x}, \overline{y}_j)\,.$$

A stationary point can be a local minimum or a saddle point for (SIP). Let SIP(f, h, g) denote the semi-infinite problem that is generated by the function vector (f, h, g). The *strong stability of a stationary point* $x(f, h, g)$ for SIP(f, h, g) is a local property; it means the existence and local uniqueness of a stationary point $x(\widetilde{f}, \widetilde{h}, \widetilde{g})$ for SIP$(\widetilde{f}, \widetilde{h}, \widetilde{g})$ where all local sufficiently small perturbations of (f, h, g) and their derivatives up to second order are considered and where $x(\widetilde{f}, \widetilde{h}, \widetilde{g})$ depends continuously on the perturbed function vector $(\widetilde{f}, \widetilde{h}, \widetilde{g})$. Here, the function vector (f, h, g) is considered as a parameter, and under certain assumptions equivalent algebraic conditions for strong stability can be obtained by using the implicit function theorem in the corresponding function space. This was done first in [16] for finite problems, a generalization to semi-infinite problems is given in [17] where the following three cases are distinguished for the considered point $\overline{x} = x(f, h, g)$.

1) The set $Y_0(\overline{x})$ is finite with $Y_0(\overline{x}) = \{\overline{y}_1, \ldots, \overline{y}_p\}$, and all points $\overline{y}_1, \ldots, \overline{y}_p$ are strongly stable local minima for $g(\overline{x}, \cdot)|_Y$. Furthermore, the *linear independence constraint qualification* (LICQ) holds at $\overline{x} \in M[h, g]$, i. e. the vectors $Dh_i(\overline{x})$, $i \in A$, $D_x g(\overline{x}, \overline{y}_j)$, $j = 1, \ldots, p$ are linearly independent.

2) $Y_0(\overline{x}) = \{\overline{y}_1, \ldots, \overline{y}_p\}$, and all points $\overline{y}_1, \ldots, \overline{y}_p$ are strongly stable local minima for $g(\overline{x}, \cdot)|_Y$. Furthermore, (MFCQ) holds at $\overline{x} \in M[h, g$ but not (LICQ).
3) Not all points from $Y_0(\overline{x})$ are strongly stable local minima for $g(\overline{x}, \cdot)|_Y$.

In Case 1 and Case 2 the reduction Ansatz is fulfilled at $\overline{x} \in M[h, g]$ can be described locally by means of finitely many $C^{1,1}$-constraints. Therefore, the Karush–Kuhn–Tucker system (written as a system of equations as in [16,17]) is Lipschitz continuous and has a generalized Jacobian. Then, an equivalent algebraic characterization for the strong stability of $\overline{x} \in M[h, g]$ in Case 1 and Case 2 is a condition on a certain subset of this generalized Jacobian (for details, cf. [17]): In Case 1 one obtains a condition on coherent orientation; here, all elements (matrices) of this subset of the generalized Jacobian restricted to the corresponding tangent space have a nonvanishing determinant with a common sign. In Case 2, all elements of this subset restricted to the corresponding tangent space are positive definite. In particular, in Case 2 a strongly stable stationary point $\overline{x} \in M[h, g]$ has to be a local minimum for SIP(f, h, g).

In Case 3 the active index set $Y_0(\overline{x})$ might contain infinitely many points! The equivalent algebraic characterization for strong stability in that case is also a positive definiteness condition but a rather technical one (for details, see [17]). In Case 3 a strongly stable stationary point $\overline{x} \in M[h, g]$ has to be a local minimum for SIP(f, h, g) as well.

Global (Structural) Stability of SIP(f, h, g)

Two problems SIP(f, h, g) and SIP($\widetilde{f}, \widetilde{h}, \widetilde{g}$) are called *equivalent* if there exist continuous mappings $\varphi \mathbf{R} \times \mathbf{R}^n$: $\mathbf{R}^n$, $\psi \mathbf{R} \to \mathbf{R}$ with the following properties:

1) The mapping $\varphi(t, \cdot)$: $\mathbf{R}^n \to \mathbf{R}^n$ is a homeomorphism for each $t \in \mathbf{R}$.
2) The mapping ψ is a homeomorphism and monotonically increasing.
3) For all $t \in \mathbf{R}$: $\varphi_t[\mathbf{L}^t(f, h, g)] = \mathbf{L}^{\psi(t)}(\widetilde{f}, \widetilde{h}, \widetilde{g})$, where $\varphi_t := \varphi(t, \cdot)$ and $\mathbf{L}^t(f, h, g) = \{x \in M[h, g]: f(x) \le t\}$.

The semi-infinite problem SIP(f, h, g) is called *structurally stable* if there exists a C_s^2-neighborhood $\mathbf{V}$ of the defining triple (f, h, g) such that SIP($\widetilde{f}, \widetilde{h}, \widetilde{g}$) and SIP($f$, h, g) are equivalent for all (f, h, g) $\in \mathbf{V}$.

The structural stability of SIP(f, h, g) can be characterized by means of the introduced topological stability of $M[h, g]$ as well as the strong stability of stationary points as follows. If $M[h, g]$ is compact, then SIP(f, h, g) is structurally stable if and only if (MFCQ) holds at all $x \in M[h, g]$, every stationary point for SIP(f, h, g) is strongly stable and different stationary points have different f-values. This result was proved first in [1,4] for finite problems and, then, it was (partially) generalized to semi-infinite problems in [8,20].

Generic Transitions

In this section one-parametric families $(\mathrm{SIP})_t$ of semi-infinite optimization problems are considered:

$$(\mathrm{SIP})_t \begin{cases} \min\limits_x & f(x, t), \\ \text{s.t.} & x \in M(t), \end{cases}$$

where all defining functions f, h, g, u, v depend on one additional variable t (the parameter), i. e., one considers $f(x, t)$, $h_i(x, t)$, $i \in I$, $g(x, t, y)$, etc. Note that, in particular, the index set $Y(t)$ now also depends on the parameter t. This is in contrast to one-parametric *finite* optimization problems (i. e., $Y(t)$ finite), where the index set of inequality constraints is not assumed to be parameter-dependent (i. e., $Y(t)$ constant; cf., e. g., [5,7,13,15]). It is assumed (in the sequel briefly referred to by A_{CUSC}) that each set $Y(t) \subset \mathbf{R}^r$, $t \in \mathbf{R}$, is compact, and that the set-valued mapping $t \to Y(t)$ is upper semicontinuous at each $t \in \mathbf{R}$.

A point $\overline{x} \in M(\overline{t})$ is called a *generalized critical point* (shortly, g.c.point) for $(\mathrm{SIP})_{\overline{t}}$ if the family of vectors $D_x f(\overline{x}, \overline{t})$, $D_x h_i(\overline{x}, \overline{t})$, $i \in I$, $D_x g(\overline{x}, \overline{t}, y)$, $y \in Y_0(\overline{x}, \overline{t}))$ is linearly dependent. Here, $D_x f$ stands for the row vector of first partial derivatives, and $Y_0(\overline{x}, \overline{t})$ denotes the parameter-dependent set of active inequality constraints, i. e., $Y_0(x, t) = \{y \in Y(t): g(x, t, y) = 0\}$. The *generalized critical point set* is defined to be the set $\Sigma = \{(x, t) \in \mathbf{R}^n \times \mathbf{R}: x$ is a g.c.point for $(\mathrm{SIP})_t\}$. By the well-known first order necessary optimality condition of F. John, for every local minimizer $\overline{x}$ of $(\mathrm{SIP})_{\overline{t}}$ one has $\overline{z} = (\overline{x}, \overline{t}) \in \Sigma$.

The main idea for the investigation of parameter-dependent local minimizers is to study the larger set Σ which contains also local maximizers and several kinds of saddle points. For a fixed generic problem, these different classes of g.c.points can easily be distinguished by algebraic conditions which use the so-called linear

and quadratic (co-)indices. In particular, the *linear (co-)index* refers to the number of (positive) negative Lagrange multipliers corresponding to active inequality constraints; the *quadratic (co-)index* counts the number of (positive) negative eigenvalues of the restriction of the Hessian of a Lagrangian to the tangent space. For a one-parametric problem, these numbers may only change at *singularities* in Σ. The nonsingular points in Σ are termed *nondegenerate critical points* (for details, cf. [13]). In the finite case, i. e., when $Y(t)$ is a constant and finite set, a nondegenerate critical point is a local minimizer if and only if its linear as well as its quadratic index vanish. Moreover, at a non-degenerate critical point $\bar{z}$, the set Σ can locally be described by the implicit function theorem which yields a regular parametrization of Σ by t. Since the quadruple of linear and quadratic (co-)indices remains constant in a neighborhood of $\bar{z}$, a nondegenerate local minimizer remains stable under small pertubations of the parameter t.

In [5,7], H.Th. Jongen, P. Jonker, and F. Twilt showed for one-parametric finite optimization problems that, apart from non-degenerate critical points (points of type 1) generically there occur exactly four different types of singularities in Σ (points of type 2, 3, 4, and 5). At each singularity the change of the index quadruple is completely described by certain characteristic numbers which can be computed from the problem data at the singularity. Results about the topological structure of Σ around the singular points yield the fundamentals for the design of numerical path following methods (for details, cf. [3]).

In *one-parametric semi-infinite optimization*, generically three additional singularities come into play. Let all defining functions of $(\mathrm{SIP})_t$ be three times continuously differentiable and define the set CUSC to be the subset of $C^3(\mathbf{R}^r \times \mathbf{R}, \mathbf{R})^{|I|+|J|}$ consisting of all functions (u, v) which define index sets $Y(t)$ such that A_{CUSC} holds. Then there exists a C^3_s-open dense subset $\mathbf{F}$ of $C^3(\mathbf{R}^n \times \mathbf{R}, \mathbf{R})^{|A|+1} \times C^3(\mathbf{R}^n \times \mathbf{R} \times \mathbf{R}^r, \mathbf{R}) \times \mathrm{CUSC}$, such that for all $(f, h, g, u, v) \in \mathbf{F}$ each point of the corresponding g.c.point set Σ is one of eight types. The latter result is given in [10] and proved in detail in [18].

In the remainder of the present section, this type classification is motivated and the local structure of Σ at the typically semi-infinite singularities is discussed. Since in the generic case the number of active indices at a point $(\bar{x}, \bar{t}) \in \Sigma$ cannot exceed $n+1$ (cf. [10]), one deals with finitely many solutions of the so-called *lower level problem* $\min g(x, t, \cdot)|_{Y(t)}$, as well as with the *upper level problem* $f(\cdot, \mathrm{t})|_{M(t)}$. The idea of the type classification in the semi-infinite case is that in exactly one of these finitely many *finite* optimization problems a singularity of codimension one occurs.

First, assume that all active indices are nondegenerate global minimizers of the lower level problem. Then a local reduction of the semi-infinite to a finite one-parametric optimization problem can be performed, as described in the previous section about the Reduction Ansatz. Here, the assumption A_{CUSC} is needed. In this reduced problem, all five types from one-parametric finite programming can occur. In particular, $(\bar{x}, \bar{t})$ is a nondegenerate critical point of $(\mathrm{SIP})_{\bar{t}}$ if it is of type 1 for the reduced problem. Moreover, this shows that the four singularities from the finite case persist in the semi-infinite setting.

The typically semi-infinite singularities arise if exactly one of the active indices in $Y_0(\bar{x}, \bar{t}) = \{\bar{y}_1, \ldots, \bar{y}_p\}$ is degenerate. Let this be the index $\bar{y}_p$. In the generic case, the degeneracy can only be due to situations such as at points of type 2, 3, 4, and 5. As points of type 3 are never local minimizers (cf. [10]), one is left with three possibilities which give rise to the typically semi-infinite singularities of type 6, 7, and 8. The present survey will not handle all details of the type definitions and of the local structures of Σ. For locally simplified problems these definitions are given in [10], whereas the definitions in terms of the original problem data are given in [18]. In the following a one-dimensional manifold Γ in $\mathbf{R}^n \times \mathbf{R}$ is said to exhibit a *turning point* at $\bar{z} = (\bar{x}, \bar{t}) \in \Gamma$ if the function $\Phi(x, t) \equiv t$, restricted to Γ, possesses a local extremum at $\bar{z}$. If, additionally, Γ is locally a C^2-manifold and the extremum of Φ is nondegenerate, then $\bar{z}$ is called a *quadratic turning point*.

A g.c.point $\bar{z}$ of type 6 can roughly be characterized by the fact that the gradients of constraints which are active at $\bar{y}_p$ in the lower level problem are linearly independent, and exactly one of the multipliers corresponding to active inequality constraints vanishes. Now consider two auxiliary problems $(\mathrm{SIP})^e_t$ and $(\mathrm{SIP})^d_t$ where this inequality constraint is treated as an equality constraint, respectively deleted as a constraint. It can be shown that $\bar{z}$ is a non-degenerate critical point of both $(\mathrm{SIP})^e_t$ and $(\mathrm{SIP})^d_t$, which gives rise to two solu-

tion curves Ψ^e and Ψ^d in $\mathbf{R}^n \times \mathbf{R}$, both being regularly parametrized by t. Exactly one branch Ψ^e_+ and Ψ^d_+ of each curve belongs to Σ, so that at $\bar{z}$ the set Σ is locally a one-dimensional manifold which is piecewise of differentiability class C^2. In the case $|A| + p = n$, the curves Ψ^e and Ψ^d meet tangentially, and Σ does not exhibit a turning point at $\bar{z}$, whereas in the case $|A| + p < n$ the curves Ψ^e and Ψ^d meet under a nonvanishing angle. Moreover, it can be shown that the linear index and co-index remain constant when passing a point of type 6 along Σ, whereas the quadratic index and co-index change if and only if Σ exhibits a turning point at $\bar{z}$. In the latter case, the quadratic index changes by one, and characteristic numbers determine the direction of change. In particular, a local minimizer can only be lost at a point of type 6, if Σ exhibits a turning point there. In this case, a feasible direction of quadratic descent (i. e., a *jump direction*) can be given (cf. [3,11,18]).

At a g.c.point $\bar{z}$ of type 7, the number of active constraints at $\bar{y}_p$ in the lower level problem does not exceed r, and their gradients are linearly dependent. It turns out that necessarily a component of the index set $Y(\bar{t})$ vanishes under pertubations of the parameter (cf. also [8]). As a consequence, the feasible set mapping $t \to M(t)$ is not upper semicontinuous at $\bar{t}$, and a branch of Σ emanates, respectively ends at $\bar{z}$. More precisely, Σ coincides locally with one branch of a one-dimensional C^2-manifold which exhibits a quadratic turning point at $\bar{z}$. A feasible direction of linear descent can be given if Σ consists locally of local minimizers (cf. [11,18]).

At a g.c.point $\bar{z}$ of type 8, the number of active constraints at $\bar{y}_p$ in the lower level problem equals $r + 1$. If the Mangasarian–Fromovitz constraint qualification holds at $\bar{y}_p$, one calls $\bar{z}$ to be of type 8*a*, else of type 8*b*. First consider points of type 8*a*. Similarly to the situation at points of type 6, there are two auxiliary problems which give rise to two C^2-curves Ψ^1, Ψ^2 of nondegenerate critical points, both being regularly parametrized by t, where exactly one branch of each curve belongs to Σ. Since Ψ^1 and Ψ^2 meet in $\bar{z}$ under a nonvanishing angle, Σ is locally a one-dimensional manifold which is piecewise of differentiability class C^2. It can be shown that Σ does not exhibit a turning point at $\bar{z}$ and that the index quadruple remains constant when passing $\bar{z}$ along Σ. Now let $\bar{z}$ be a g.c.point of type 8*b*. Similarly to the situation at points of type 7, a component of the index set $Y(\bar{t})$ vanishes under pertubations of the parameter, and a branch of Σ emanates, respectively ends at $\bar{z}$. More precisely, Σ coincides locally with one branch of a one-dimensional C^2-manifold which can be regularly parametrized by t. Again a feasible direction of linear descent can be given if Σ consists locally of local minimizers (cf. [11,18]).

Finally, there is a remarkable phenomenon concerning the global structure of Σ. In contrast to the case of one-parametric finite programming where the singular points form the (relative) boundary of the set of non-degenerate critical points (cf. [5,7]), in the semi-infinite case there appears an additional type of boundary point which does not belong to Σ. This so-called *trap-door point* occurs when a new component of $Y(t)$ is born at a parameter value $\bar{t}$ (recall that at points of type 7 and type 8*b* such a component vanishes). The occurrence of trap-door points as well as of points of type 7 and 8*b* can be avoided if the mapping $t \to Y(t)$ is not only assumed to be upper but also lower semicontinuous. In this case, the singular points form the (relative) boundary of the set of non-degenerate critical points. For details, cf. [18,19].

See also

- ▶ Bounds and Solution Vector Estimates for Parametric NLPs
- ▶ Multiparametric Linear Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Nonlocal Sensitivity Analysis with Automatic Differentiation
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Parametric Optimization: Embeddings, Path Following and Singularities
- ▶ Selfdual Parametric Method for Linear Programs
- ▶ Sensitivity Analysis of Complementarity Problems
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Sensitivity and Stability in NLP
- ▶ Sensitivity and Stability in NLP: Approximation
- ▶ Sensitivity and Stability in NLP: Continuity and Differential Stability

References

1. Guddat J, Jongen HTh (1987) Structural stability and nonlinear optimization. Optim 18:617–631
2. Guddat J, Jongen HTh, Rückmann J-J (1986) On stability and stationary points in nonlinear optimization. J Austral Math Soc B28:36–56
3. Guddat J, Guerra Vasquez F, Jongen HTh (1990) Parametric optimization: singularities, pathfollowing and jumps. Wiley and Teubner, Stuttgart
4. Hettich R, Jongen HTh (1978) Semi-infinite programming: conditions of optimality and applications. In: Stoer J (ed) Optimization Techniques, Part 2. Lecture Notes Control Inform Sci. Springer, Berlin, pp 1–11
5. Jongen HTh, Jonker P, Twilt F (1986) Critical sets in parametric optimization. Math Program 34:333–353
6. Jongen HTh, Jonker P, Twilt F (1986) Nonlinear optimization in Rn, II. transversality, flows, parametric aspects. P. Lang, Frankfurt am Main
7. Jongen HTh, Jonker P, Twilt F (1986) One-parameter families of optimization problems: equality constraints. J Optim Th Appl 48:141–161
8. Jongen HTh, Rückmann J-J (1998) On stability and deformation in semi-infinite optimization. In: Reemtsen R, Rückmann J-J (eds) Semi—Infinite Programming. Kluwer, Dordrecht, pp 29–66
9. Jongen HTh, Rückmann J-J, Weber G-W (1994) One-parametric semi-infinite optimization: on the stability of the feasible set. SIAM J Optim 4(3):637–648
10. Jongen HTh, Stein O (1997) On generic one-parametric semi-infinite optimization. SIAM J Optim 7(4):1103–1137
11. Jongen HTh, Stein O (1997) Parametric semi-infinite programming: Jumps in the set of local minimizers. In: Guddat J, Jongen HTh, Nožička F, Still G, Twilt F (eds) Parametric Optimization and Related Topics IV. P. Lang, Frankfurt am Main, pp 161–175
12. Jongen HTh, Twilt F, Weber G-W (1992) Semi-infinite optimization: structure and stability of the feasible set. J Optim Th Appl 72:529–552
13. Jongen HTh, Weber G-W (1990) On parametric nonlinear programming. Ann Oper Res 27:253–284
14. Jongen HTh, Weber G-W (1991) Nonlinear optimization: characterization of structural stability. J Global Optim 1:47–64
15. Jongen HTh, Weber G-W (1992) Nonconvex optimization and its structural frontiers. In: Krabs W, Zowe J (eds) Modern Methods of Optimization. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 151–203
16. Kojima M (1980) Strongly stable stationary solutions in nonlinear programs. In: Robinson SM (ed) Analysis and Computation of Fixed Points. Acad. Press, New York, pp 93–138
17. Rückmann J-J (1999) On existence and uniqueness of stationary points in semi-infinite optimization. Math Program 86:387–415
18. Stein O (1997) On parametric semi-infinite optimization. PhD Thesis. Shaker, Aachen
19. Stein O (1997) Trap-doors in the solution set of semi-infinite optimization problems. In: Gritzmann P, Horst R, Sachs E, Tichatschke R (eds) Recent Advances in Optimization. Springer, Berlin, pp 348–355
20. Weber G-W (1992) Charakterisierung struktureller Stabilität in der nichtlinearen Optimierung. PhD Thesis, Department Math, RWTH Aachen, Aachen

Parametric Linear Programming: Cost Simplex Algorithm

VIVEK DUA, EFSTRATIOS N. PISTIKOPOULOS
Imperial College, London, UK

MSC2000: 90C05, 90C31

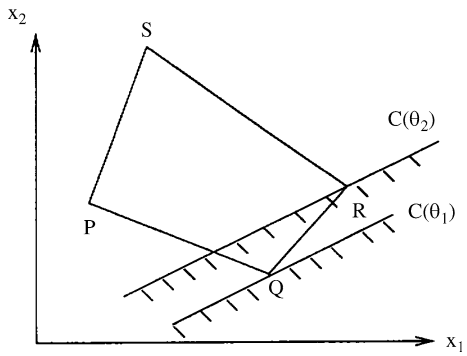
Article Outline

Keywords
Parametric Linear Programming
Objective Function Parametrization
See also
References

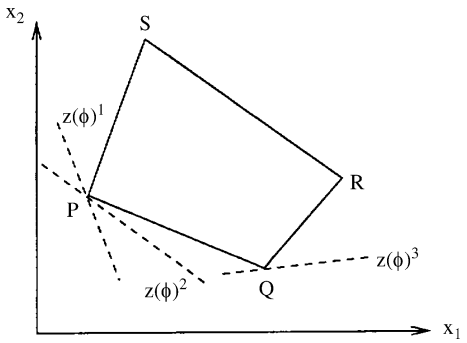
Keywords

Parametric linear programming; Critical region; Sensitivity analysis with respect to changes in cost coefficients

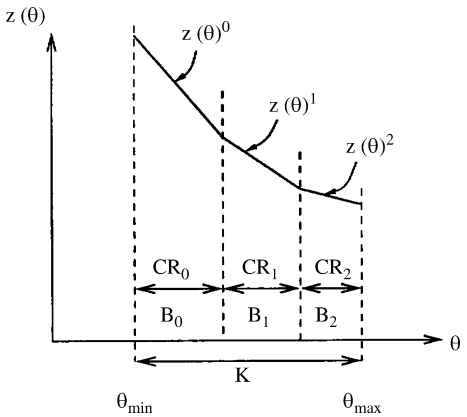
In this article, we describe solution approaches for two types of linear problems involving uncertain parameters. The first problem that is discussed, involves a single uncertain parameter present on the right-hand side of the constraints, and the second problem involves uncertainty in coefficients of the objective function. The significance of solving these problems is that, while the first problem takes into account the case when a parameter associated with the model equations, such as demand and supply is uncertain, the second problem incorporates uncertainty in the coefficients of variables which define the objective function, such as cost of raw materials and selling price of products. Mathematically, the former problem depicts the value of objective function as the feasible region shrinks (or enlarges), and



Parametric Linear Programming: Cost Simplex Algorithm, Figure 1
Right-hand side parametrization



Parametric Linear Programming: Cost Simplex Algorithm, Figure 2
Objective function parametrization



Parametric Linear Programming: Cost Simplex Algorithm, Figure 3
Optimal value function (right-hand side case)

the latter characterizes the 'fixed' feasible region with a number of regions corresponding to different sets of active and inactive constraints.

For the first problem, in Fig. 1, *PQRS* represents an initial feasible region corresponding to the value θ_1 of the uncertain parameter, and *C* represents the constraint involving θ. As θ_1 changes from θ_1 to θ_2, the feasible region and the set of active constraints changes. For a given range of θ, the aim is then to identify intervals of θ within which the optimal value function function, $z(\theta)$, preserves its optimality. For the second problem, in Fig. 2, where $z(\phi)$ represents the objective function and ϕ represents the vector of uncertain parameters in the objective function coefficients, although the feasible region *PQRS* remains constant, the slope of objective function changes with change in ϕ. The aim in this case is to identify regions rather than the intervals of uncertain parameter space. Next we will describe solution approaches for both the problems.

Parametric Linear Programming

Consider the following parametric linear programming problem:

$$\begin{cases} z(\theta) & = \min\limits_{x} c^{\top} x \\ \text{s.t.} & Ax = b + f\theta, \\ & x \geq 0, \\ & \theta_{\min} \leq \theta \leq \theta_{\max}, \\ & x \in \mathbf{R}^n, \end{cases} \tag{1}$$

where x is a vector of continuous variables; A is a constant matrix, and c, b and f are constant vectors of appropriate dimensions; θ is a scalar uncertain parameter. The solution of (1) is approached by incorporating the uncertain parameter, θ, in the simplex tableau. The solution procedure consists of two phases. In the first phase, an optimal solution for any θ in $[\theta_{\min}, \theta_{\max}]$ is obtained. Let B_1 denote the corresponding optimal basis, and $x_B^1(\theta) = B^{-}1(b + f\,\theta)$, the vector of corresponding basic variables. The *critical region*, CR_1 (Fig. 1), associated with this basis is given by the condition of *primal feasibility*, $x_B^1(\theta) \geq 0$, together with the condition that $\theta_{\min} \leq \theta \leq \theta_{\max}$. The optimal value function, $z(\theta)^1$, is given by $z(\theta)^1 = c_B^{\top} x_B^1(\theta)$. This completes the first phase. In the second phase, the *neighboring bases* (B_0 or B_2) are identified by one dual step. The procedure is

repeated until the complete range of θ has been characterized by critical regions and corresponding optimal value functions. While solving problems of the form given in (1) is usually computationally expensive, it is better than exhaustively obtaining the optimal solutions for all the values of θ that lie within $\theta_{\min}$ and $\theta_{\max}$. However, in the worst case the computational requirements for solving (1) are not bounded by a polynomial in the size of the problem [10]. Some other references on the subject are [1,2,3,11] and [12].

Objective Function Parametrization

Consider the following linear programming problem with uncertain objective function coefficients:

$$\begin{cases} z(\phi) & = \min\limits_{x} c^{\top}(\phi)x \\ \text{s.t.} & Ax = b, \\ & x \geq 0, \\ & x \in \mathbf{R}^n, \quad \phi \in \mathbf{R}^s, \end{cases} \tag{2}$$

where $c(\phi) = c + H\,\phi$, such that c is a constant vector and H is a constant matrix; b is a constant vector; ϕ is a vector of uncertain parameters, such that for each $\phi \in K$, $\phi \in \mathbf{R}^s$, (2) has a finite optimal solution, and has no optimal solution for $\phi \in \mathbf{R}^s - K$. Further, consider the following restriction on $\phi \in \Psi$, $\Psi = \{\phi\colon G\phi \leq g\}$, where G is a constant matrix and g is a constant vector. By defining new variables v and z, (2) can be rewritten as:

$$\begin{cases} z(\phi) & = \min\limits_{x} \phi^{\top} v + z \\ \text{s.t.} & Ax = b, \\ & v = H^{\top}x, \\ & z = c^{\top}x, \\ & x \geq 0, \\ & x \in \mathbf{R}^n, \quad \phi \in \mathbf{R}^s. \end{cases} \tag{3}$$

In the simplex based solution algorithm [4], associated with each optimal basis is a region of ϕ which can be uniquely defined for that basis. The basic idea of the solution approach is then to identify all such bases so that Ψ is completely characterized by a number of regions, and the corresponding value of the objective function. This can be achieved as follows.

Let B denote an optimal basis and x_B denote the corresponding optimal solution; also let ρ denote the corresponding index, also let the basic variables in H be denoted by H_B, then we can write:

$$\begin{aligned} {}^{\rho}H^{\top} &= H_B^{\top}Y - H^{\top}\,, \\ {}^{\rho}z^{\top} &= c_B^{\top}Y - c^{\top}\,, \\ h_{N+1} &= H_B^{\top}x_B\,, \\ z^{(\rho)} &= c_B^{\top}x_B\,, \end{aligned} \tag{4}$$

where

$$Y = B^{-1}A\,, \quad x_B = B^{-1}b\,.$$

The set of equations in (3) can now be rewritten as follows:

$$\begin{aligned} Ax &= b\,, \\ v + {}^{\rho}H^{\top}x &= h_{N+1}\,, \\ z + {}^{\rho}z^{\top}x &= z^{(\rho)}\,, \end{aligned}$$

where ${}^{\rho}z = c_B^{\top}Y$, and using appropriate transformation, the objective function can be written as:

$$z(\phi) = -({}^{\rho}z^{\top} + \phi^{\top}{}^{\rho}H^{\top})x + \phi^{\top}h_{N+1} + z^{(\rho)}\,. \tag{5}$$

By denoting $c_B(\phi) = c_B + H_B\,\phi$, and substituting in (4) the following is obtained:

$${}^{\rho}z(\phi) = {}^{\rho}z + {}^{\rho}H\phi\,. \tag{6}$$

As described later, this is an important relation for characterizing the region of ϕ associated with the optimal basis B. In order to obtain parametric solution and the corresponding regions of ϕ, we first need the following:

- The optimal value function, $z_{\min}(\phi)$, is concave, linear and continuous over K.
- Two bases are said to be *neighboring bases* if and only if

  i) there exists $\phi^* \in K$ such that both bases are optimal bases for $\phi^* \in K$, and
  ii) it is possible to pass from one bases to the another by one primal simplex step.

- The critical region where the basis B remains optimal is given by the following condition of *dual feasibility*: ${}^{\rho}z(\phi) \geq 0$, or $-{}^{\rho}H\,\phi \leq {}^{\rho}z$, together with the restriction on ϕ given by $G\,\phi \leq g$.
- item Two critical regions are said to be neighbors (or *neighboring critical regions*) if their correspond-

ing optimal bases are neighbors (or neighboring bases).

The solution algorithm then relies on finding an initial optimal solution and the corresponding critical region, and then identifying all the critical regions by passing from one region to its neighbor by a primal simplex step. The procedure is continued until the complete space of ϕ, along with the corresponding optimal value function, has been characterized. Also see [5,6,7,8,9,13,14] and [3] for further details.

See also

- Bounds and Solution Vector Estimates for Parametric NLPs
- Convex-simplex Algorithm
- Global Optimization in Multiplicative Programming
- Lemke Method
- Linear Complementarity Problem
- Linear Programming
- Multiparametric Linear Programming
- Multiparametric Mixed Integer Linear Programming
- Multiplicative Programming
- Nondifferentiable Optimization: Parametric Programming
- Parametric Global Optimization: Sensitivity
- Parametric Mixed Integer Nonlinear Optimization
- Parametric Optimization: Embeddings, Path Following and Singularities
- Selfdual Parametric Method for Linear Programs
- Sequential Simplex Method

References

1. Akgül M (1984) A note on shadow prices in linear programming. J Oper Res Soc 35:425–431
2. Aucamp DC, Steinberg DI (1982) A note on shadow prices in linear programming. J Oper Res Soc 33:557–565
3. Gal T (1995) Postoptimal analyses, parametric programming, and related topics. de Gruyter, Berlin
4. Gal T, Nedoma J (1972) Multiparametric linear programming. Managem Sci 18:406–422
5. Gass SI (1975) Linear programming – Methods and applications, 4th edn. McGraw-Hill, New York
6. Gass SI, Saaty TL (1955) The parametric objective function. II. Oper Res 3:395–401
7. Greenberg HJ (1993) How to analyze the results of linear programs - Part 2: Price Interpretation. Interfaces 23(5):97–114
8. Kelley JE (1959) Parametric programming and the primal-dual algorithm. Oper Res 7:327–334
9. McKeown PG, Minch RA (1982) Multiplicative interval variation of objective function coefficients in linear programming. Managem Sci 28:1462–1470
10. Murty KG (1980) Computational complexity of parametric linear programming. Math Program 19:213–219
11. Murty KG (1983) Linear programming. Wiley, New York
12. Roos C, Terlaky T, Vial J-Ph (1997) Theory and algorithms for linear optimization, an interior point approach. Wiley, New York
13. Simons E (1962) A note on parametric linear programming. Managem Sci 8:355–358
14. Ward JE, Wendell RE (1990) Approaches to sensitivity analysis in linear programming. Ann Oper Res 27:3–38

Parametric Mixed Integer Nonlinear Optimization

VIVEK DUA, EFSTRATIOS N. PISTIKOPOULOS
Imperial College, London, UK

MSC2000: 90C31, 90C11

Article Outline

Keywords
See also
References

Keywords

Parametric upper and lower bounds; Decomposition algorithms

In this article we will present some developments in the subject of parametric programming for problems involving 0–1 integer variables and nonlinearities in the model. For pure quadratic integer problems, [13] proposed an algorithm by extending some of the concepts described in [10] for the case of linear models. The problem they considered can be stated as follows:

$$\begin{cases} z(\theta) & = \min\limits_{y} p(y) \\ \text{s.t.} & Ay \geq b + r\theta, \\ & 0 \leq \theta \leq 1, \\ & y \in \{0,1\}^m, \end{cases} \tag{1}$$

where θ is a scalar uncertain parameter; $p(y) \equiv y^{\mathsf{T}}Qy$; r is a nonnegative n-vector, such that only $r_1 \geq 0$; Q is an $(m \times m)$ matrix; A is an $(n \times m)$ matrix. The solution procedure is based upon the observation that as θ is increased from 0 to 1, it results in shrinking of the feasible region (because $r \geq 0$) of (1) and hence only a solution worse than (greater than or equal to) the solution at 0 can be obtained, i. e., $z(\theta)$ is a nondecreasing function of θ (see Fig. 1). Another interesting feature of the formulation in (1) is that since θ is bounded between lower and upper bounds, there is a finite number of (integer) solutions that will lie between these bounds. Further, because the optimal solution of (1) at a fixed value of θ corresponds to lattice points of a polyhedron, it remain optimal for a range of θ until another lattice point becomes optimal (this results in a finite number of intervals of θ corresponding to the finite number of solutions that lie in [0, 1]; Fig. 1). [13] proposed the following procedure for identifying these (*critical*) *intervals* and corresponding optimal solutions:

0 Set $k = 1$ and let $\theta = 0$.

1 For (1) find the optimum solution y_k and the corresponding $p(y_k)$. Also find θ_k using:

$$\theta_k = \frac{a_1 y_k - b_1}{r_1},$$

where a_1 denotes the first row of A.
IF no such y_k exists, THEN go to 3,
ELSE go to 2.

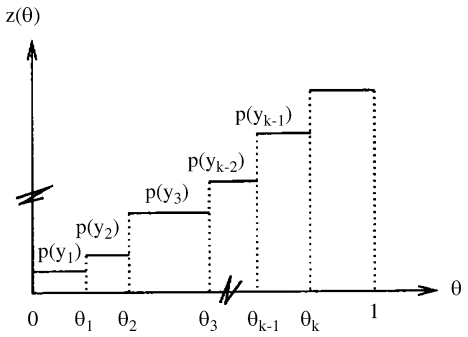
2 Set $k = k + 1$ and let $\theta = \theta_{k-1} + \delta$, where

$$\theta = \frac{a_1 y_{k-1} - b_1 + g}{r_1} = \theta_{k-1} + \delta,$$

where $\delta = g/r_1$ and g is the greatest common divisor of the elements of a_1.
IF $\theta \leq 1$, THEN go to 1, else go to 3.

3 Stop, all critical solutions have been found.

[2] extended their previous work on dynamic programming to present a solution procedure for pure integer nonlinear programming problems. [18] extended the branch and bound technique of [17] for linear programs to nonlinear programs. They also discussed the extension of their results to the mixed integer case. We next consider the following parametric mixed integer nonlinear programming problem:



Parametric Mixed Integer Nonlinear Optimization, Figure 1
***z*(*θ*) is a discontinuous nondecreasing step function**

$$\begin{cases} z(\theta) & = \min\limits_{y,x} c^{\top} y + f(x) \\ \text{s.t.} & h(x) = 0, \\ & By + g(x) \leq b + r\theta, \\ & \theta_{\min} \leq \theta \leq \theta_{\max}, \\ & x \in X, \quad y \in \{0,1\}^m, \end{cases} \tag{2}$$

where f and g are continuously differentiable and convex on the n-dimensional compact polyhedral convex set $X = \{x: x \in \mathbf{R}^n, A_1 x \leq a_1$; h is affine with respect to x; θ is a scalar uncertain parameter; c, b and r are constant vectors and B is a constant matrix.

The solution of (2) can be approached by using

i) outer-approximation (OA) [3]; or,
ii) generalized Benders decomposition (GBD) principles [9].

See [7] for details and some applications of these algorithms. The basic idea in both the approaches is to decompose the problem into a *primal* and a *master subproblem*, and the basic difference between the two approaches is in the formulation of the master subproblem. The solution of primal subproblem, which is obtained for fixed y, represents a parametric upper bound, whereas, the solution of master subproblem, which is obtained for a relaxation of the initial problem, represents a parametric lower bound. The solution algorithm is based upon iterating between these two subproblems (upper and lower bounds).

The primal subproblem is formulated by fixing $y = \overline{y}$ in (2) to obtain a parametric NLP (pNLP) of the following form:

$$\begin{cases} z(\overline{y},\theta) & = \min\limits_{x} c^\top \overline{y} + f(x) \\ \text{s.t.} & h(x) = 0, \\ & B\overline{y} + g(x) \le b + r\theta, \\ & \theta_i \le \theta \le \theta_{i+1}, \\ & x \in X. \end{cases} \tag{3}$$

Under appropriate continuity and convexity properties ([4,5,6,11]), the solution of (3) is approximated by creating linear parametric upper and lower bounds at the extreme points of the interval $[\theta_i, \theta_{i+1}]$. A *parametric upper bound* is given by:

$$\widehat{z}(\overline{y},\theta) = \alpha z^*(\overline{y},\theta_i) + (1-\alpha) z^*(\overline{y},\theta_{i+1}), \quad \alpha \in (0,1) \tag{4}$$

and a *parametric lower bound* is given by:

$$\begin{aligned} &\overline{z}(\overline{y},\theta) = \max\{\mathrm{LB}_i, \mathrm{LB}_{i+1}\}, \\ &\mathrm{LB}_i = z^*(\overline{y},\theta_i) + \nabla_\theta z^*(\overline{y},\theta_i)(\theta-\theta_i), \\ &\mathrm{LB}_{i+1} = z^*(\overline{y},\theta_{i+1}) + \nabla_\theta z^*(\overline{y},\theta_{i+1})(\theta-\theta_{i+1}), \end{aligned} \tag{5}$$

where $z^*(\overline{y},\theta_i)$ and $z^*(\overline{y},\theta_{i+1})$ are the solutions of (3) at θ_i and θ_{i+1}, respectively; $\nabla_\theta z^*$ is evaluated through the Lagrange multipliers. The parametric upper and lower bounds are tightened at the intersection point (in θ) of LB_i and LB_{i+1} breaking the interval of θ into two smaller intervals, within which another set of upper and lower bounds are obtained. This approximation is continued until the difference between upper and lower bounds is within ϵ. If an infeasibility is found at an extreme point, the following problem is solved to obtain the feasible interval:

$$\begin{cases} \min\limits_{x,\theta} \theta \quad (\text{or } \max\limits_{x,\theta} \theta) \\ \text{s.t.} \quad h(x) = 0, \\ \quad B\overline{y} + g(x) - r\theta \le b, \\ \quad x \in X, \\ \quad \theta_i \le \theta \le \theta_{i+1}. \end{cases} \tag{6}$$

The final solution of a primal subproblem is given by parametric upper bounds obtained in their corresponding intervals of θ.

The master subproblem can then be constructed by using either OA or GBD principles. Using OA principles, master subproblem is formulated as follows ([1,15,16]):

$$\begin{cases} z'(\theta) & = \min\limits_{y,x,\mu} c^\top y + \mu \\ \text{s.t.} & T(\lambda)[h(x^*) + \nabla h(x^*)(x-x^*)] \le 0, \\ & By + g(x^*) + \nabla g(x^*)(x-x^*) \\ & \quad \le b_0 + r\theta, \\ & f(x^*) + \nabla f(x^*)(x-x^*) - \mu \le 0, \\ & \sum\limits_{j\in J} y_j - \sum\limits_{j\in L} y_j \le |J| - 1, \\ & \theta_i \le \theta \le \theta_{i+1}, \\ & x \in X, \quad y \in \{0,1\}^m, \end{cases} \tag{7}$$

where μ is a scalar variable; $T(\lambda)$ is a diagonal matrix with elements

$$t_{p,p} = \begin{cases} -1, & \lambda_p < 0, \\ 0, & \lambda_p = 0, \\ 1, & \lambda_p > 0, \end{cases}$$

where λ_p is the Lagrange multiplier of equation h_p, $p = 1,\ldots,P$, [12]; x^* are the solutions of (3) obtained at the extreme points of intervals of θ while solving the primal subproblem; $J = \{j: y = 1\}$ and $L = \{j: y = 0\}$, and $|J|$ is the cardinality of J.

Alternatively, using GBD principles, the master subproblem is formulated as follows [14]:

$$\begin{cases} z'(\theta) & = \min\limits_{y,\mu} \mu \\ \text{s.t.} & \mu \ge c^\top y + f(x^*) + \lambda^\top [h(x^*)] \\ & \quad + \tau^\top [By + g(x^*) - b - r\theta], \\ & \lambda_{\mathrm{inf}}^\top [h(x^*)] \\ & \quad + \tau_{\mathrm{inf}}^\top [By + g(x^*) - b - r\theta] \le 0, \\ & \theta_i \le \theta \le \theta_{i+1}, \\ & y \in \{0,1\}^m, \end{cases} \tag{8}$$

where λ and τ are Lagrange multipliers obtained at extreme points of intervals of θ in the primal subproblem; and the subscript inf corresponds to the Lagrange multipliers in the infeasible interval of θ, where following

relaxed problem is solved:

$$\begin{cases} \min\limits_{x,s_1,s_2,\phi} & (s_1+s_2+\phi) \\ \text{s.t.} & h(x)+s_1-s_2=0, \\ & By+g(x)-b-r\theta_{\text{inf}} \le \phi, \\ & s_1,s_2,\phi \ge 0, \\ & x \in X, \end{cases} \tag{9}$$

where s_1, s_2 and ϕ are positive slack variables and the subscript inf corresponds to infeasible θ points.

Note that (7) and (8) are parametric mixed integer linear programming (pMILP) problems, which can be solved using the following algorithm proposed by [15]. The solution is approached by fixing θ at the lower value of the interval, in the master subproblem. This results in a deterministic MILP, which is then solved to obtain an integer solution given by y'. Fixing $y = y'$ in the master subproblem results in a parametric LP (pLP) problem. The solution of this pLP, obtained by using an algorithm described by [8], is given by a linear parametric profile, $\widehat{z}'(\theta)$, and represents an upper bound on the solution of master subproblem. Another MILP is formulated, to obtain a *breakpoint* θ_{bp}, where some other integer solution is lower than the upper bound, $\widehat{z}'(\theta)$, as follows:

$$\begin{cases} \theta_{\text{bp}} & = \min\limits_{x,\theta} \theta \\ \text{s.t.} & T(\lambda)[h(x^*)+\nabla h(x^*)(x-x^*)] \le 0, \\ & By+g(x^*)+\nabla g(x^*)(x-x^*) \\ & \quad \le b_0+r\theta, \\ & f(x^*)+\nabla f(x^*)(x-x^*)-\mu \le 0, \\ & \sum\limits_{j\in J} y_j - \sum\limits_{j\in L} y_j \le |J|-1, \\ & \mu \le \widehat{z}'(\theta), \\ & \theta_i \le \theta \le \theta_{i+1}, \\ & x \in X, \\ & y \in \{0,1\}^m, \quad y \ne y'. \end{cases} \tag{10}$$

For the interval $[\theta_i, \theta_{\text{bp}}]$ the solution is given by $z'(\theta) = \widehat{z}(\theta)'$, and for the rest of the interval, $[\theta_{\text{bp}}, \theta_{i+1}]$ the procedure is repeated until (10) is infeasible. The final solution is given by a set of parametric profiles valid in their corresponding intervals.

0	(initialization) Define an interval of θ; tolerance, ϵ; an upper bound $\widehat{z}^*(\theta) = \infty$; initial $y = \overline{y}$.
1	(primal problem) For each interval with a new integer structure, $\overline{y}$:
1a	Solve problem (3) to obtain $\widehat{z}(\overline{y}, \theta)$ such that tolerance, ϵ, is satisfied.
1b	If $\widehat{z}(\overline{y}, 0) \le \widehat{z}^*(\theta)$ update the best upper bound function, and the corresponding integer solutions, y^*.
2	(master problem) For each interval, solve either (7) or (8) to obtain a lower bound $z'(\theta)$ and the corresponding break points. Define the new set of intervals.
3	(convergence) If for some interval $\widehat{z}^*(\theta) \le z'(\theta)$, or the master problem is feasible, the solution is given by the current solution, otherwise return to primal problem with new integer solution.

This solution of the master subproblem represents a parametric lower bound. If in an interval the lower bound, $z'(\theta)$, exceeds the upper bound, $\check{z}(\overline{y}, \theta)$, or the master subproblem is infeasible the algorithm stops with the current solution in those intervals as the final solution, otherwise, the new vector y along with the corresponding interval obtained from the solution of the master subproblem is returned back to the primal subproblem.

The main steps of the algorithm are summarized above.

These algorithms have been tested on a number of applications including process synthesis and planning, heat exchanger network synthesis, multi-objective optimization and simultaneous product and process design ([1,14,15]).

See also

- Bounds and Solution Vector Estimates for Parametric NLPs
- Branch and Price: Integer Programming with Column Generation
- Decomposition Techniques for MILP: Lagrangian Relaxation
- Integer Linear Complementary Problem
- Integer Programming
- Integer Programming: Algebraic Methods

- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Linear Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Parametric Global Optimization: Sensitivity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Parametric Optimization: Embeddings, Path Following and Singularities
- ▶ Selfdual Parametric Method for Linear Programs
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem

References

1. Acevedo J, Pistikopoulos EN (1996) A parametric MINLP algorithm for process synthesis problems under uncertainty. Industr Eng Chem Res 35(1):147–58
2. Cooper MW (1981) Postoptimality analysis in nonlinear integer programming: The right-hand side case. Naval Res Logist Quart 28:301–307
3. Durán MA, Grossmann IE (1986) An outer-aproximation algorithm for a class of MINLP. Math Program 36:307–339
4. Fiacco AV (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Acad. Press, New York
5. Fiacco AV, Kyparisis J (1986) Convexity and concavity properties of the optimal value function in parametric nonlinear programming. J Optim Th Appl 48:95–126
6. Fiacco AV, Kyparisis J (1988) Computable bounds on parametric solutions of convex problems. Math Program 40:213–221
7. Floudas CA (1995) Nonlinear and mixed-integer optimization. Oxford University Press, Oxford
8. Gal T (1995) Postoptimal analyses, parametric programming, and related topics. de Gruyter, Berlin
9. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10(4):237–260
10. Geoffrion AM, Nauss R (1977) Parametric and postoptimality analysis in integer linear programming. Managem Sci 23(5):453–466
11. Hogan WW (1973) The continuity of the perturbation function of a convex program. Oper Res 21:351–352
12. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization for process flowsheets. Industr Eng Chem Res 26(9):1869–1880
13. McBride RD, Yorkmark JS (1980) Finding all the solutions for a class of parametric quadratic integer programming problems. Managem Sci 26:784–795
14. Papalexandri KP, Dimkou TI (1998) A parametric mixed integer optimization algorithm for multi-objective engineering problems involving discrete decisions. Industr Eng Chem Res 37(5):1866–1882
15. Pertsinidis A (1992) On the parametric optimization of mathematical programs with binary variables and its application in the chemical engineering process synthesis. PhD Thesis, Carnegie-Mellon University
16. Pertsinidis A, Grossmann IE, McRae GJ (1998) Parametric optimization of MILP programs and a framework for the parametric optimization of MINLPs. Comput Chem Eng 22:S205
17. Schrage L, Wolsey L (1985) Sensitivity analysis for {branch-and-bound} integer programming. Oper Res 33:1008–1023
18. Skorin-Kapov J, Granot F (1987) Nonlinear integer programming: Sensitivity analysis for branch-and-bound. Oper Res Lett 6(6):269–274

Parametric Optimization: Embeddings, Path Following and Singularities

JÜRGEN GUDDAT[1], FRANCISCO GUERRA[2], DIETER NOWACK[1]

[1] Department Math., Humboldt University, Berlin, Germany

[2] Department Fisica y Mat., University Americas, Cholula, Mexico

MSC2000: 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34, 65K05, 65K10

Article Outline

Keywords
Introduction

Generic Singularities, the Approach via Piecewise Differentiability and Topological Stability
Concluding Remarks
See also
References

Keywords

Parametric optimization; Generic singularities; Path following; Homotopy method; Continuation method; Embedding

Introduction

The principle of *embedding* for nonlinear equations has been known for more than 40 years (cf. e. g. [9, Chapt. 11], and the historical remarks there, [10]). We consider the nonlinear system of equations

$$F(x) = 0 \tag{1}$$

defined by F: $\mathbf{R}^n \to \mathbf{R}^n$ (e. g. $F \in C^2(\mathbf{R}^n, \mathbf{R}^n)$) and equation (1) is embedded by

$$H(x, t) = 0\,, \quad t \in [0, 1]\,,$$

with the following properties:

$$H(x^0, 0) = 0\,, \quad H(x, 1) = F(x)\,,$$

where H: $\mathbf{R}^n \times \mathbf{R} \to \mathbf{R}^n$, $x^0 \in \mathbf{R}^n$ arbitrarily chosen and fixed.

The following so-called standard embedding is one example:

$$H(x, t) := F(x) + (t - 1)F(x^0)\,.$$

Moreover, there are many practical examples, in particular from mechanics and electrotechnics, which, a priori, depend on one real parameter t. Then the function H is given explicitly. In order to find a solution for (1), we have to find a discretization of the interval $[0, 1]$:

$$0 = t_0 < \cdots < t_i < t_{i+1} < \cdots < t_N = 1 \tag{2}$$

and corresponding $(x(t_i), t_i)$ with $H(x(t_i), t_i) = 0$, $i = 1, \ldots, N$, using the starting point x^0. The main tools for finding such a discretization are *path following* methods (also called *homotopy methods* or *continuation methods*). The general principle will be explained under the following assumptions:

E1) $H \in C^2(\mathbf{R}^n \times \mathbf{R}, \mathbf{R})$.

E2) There exists a function $x \in C^1([\underline{t}, \bar{t}], \mathbb{R}^n)$ such that $H(x(t), t) = 0$.

If we denote by $r(t)$ the radius of convergence, for instance for the Newton method solving $H(x, t) = 0$ in a neighborhood of $x(t)$, then we have (cf. [3])

Theorem 1 *Assume E1) and E2). Then there exists a real number $r > 0$ such that $r() \geq r$ for all $t \in [\underline{t}, \bar{t}]$.*

Path following methods are realized by so-called predictor-corrector schemes. The main idea of such a scheme is the following: $t_i \to t_{i+1}$ beginning at some t with known $x(t)$ is the predictor, and the Newton method is the corrector. More precisely, using the known point $(\widetilde{x}^i, t_i)$ as an approximation of $(x(t_i), t_i)$, we compute an approximation $(\widetilde{x}^{i+1}, t_{i+1})$ of $(x(t_{i+}), t_{i+1})$ by a finite number of Newton steps, where t_{i+1} has to be chosen in such a way that $\widetilde{x}^i$ lies in the convergence region of the problem

$$H(x, t_{i+1}) = 0$$

if $|t_{i+1} - t_i|$ is chosen sufficiently small.

For example, consider $H(x, t) = 3/x + t - 4.5$. The point $(\widetilde{x}^i, t_i) = (0.7499, 0.5)$ is an approximation of $(x(t_i), t_i)$ and for $t_{i+1} = 0.6$, $\widetilde{x}^i$ lies in the convergence region of $H(x, 0.6) = 0$, starting at $\widetilde{x}^{i+1} = 0.7692$.

Let $\bar{t}$ be the endpoint of the scheme. In case of $[\underline{t}, \bar{t}] = [0, 1]$, we obtain, in a finite number of steps, a point lying in $r(1)$ and having at least superlinear convergence.

Concerning parameter-dependent equations we refer to [1] and the very good and informative bibliography therein.

Now we consider a nonlinear optimization problem:

$$\text{(P)}\ \min \left\{ f(x) \colon \begin{array}{l} h_i(x) = 0,\ i \in I, \\ g_j(x) \leq 0,\ j \in J \end{array} \right\},$$

(where $I := \{1, \ldots, m\}$, $J = \{1, \ldots, s\}$ and assume that f, h_i, $g_j \in C^k(\mathbf{R}^m, \mathbf{R})$, $i \in I$, $j \in J$, and $k \geq 1$ will be specified later) and the corresponding Karush–Kuhn–

Tucker system (shortly KKT-system)

$$\begin{cases} Df(x) + \sum_{i\in I} \lambda_i Dh_i(x) \\ \quad + \sum_{j\in J} \mu_j Dg_j(x) = 0, \\ h_i(x,t) = 0, & i \in I, \\ g_j(x) \le 0, & j \in J, \\ \mu_j \ge 0, & j \in J, \\ \mu_j \cdot g_j(x) = 0, & j \in J \end{cases} \tag{3}$$

(here $Df(x) := (\partial f/\partial x_1, \ldots, \partial f/\partial x_n)$ is a row vector).

In the literature we can distinguish two general approaches to realize the idea explained above for nonlinear optimization problems.

- First approach: Transform the KKT system into a system of equations or into a generalized equation.
- Second approach: (P) will be embedded by a one-parametric optimization problem.

In the first approach, both transformations are well known. The KKT system (3) is a mixture of equations and inequalities. The reformulation to a system of equations is quite simple. For $y \in \mathbf{R}$ we define

$$y^+ := \max\{y, 0\}\,, \quad y^- := \min\{y, 0\}\,.$$

We can formulate the KKT system as an equivalent system of equations:

$$\mathcal{H}(x,y) := \begin{bmatrix} D(f(x) + \sum_{i\in I} y_i Dh_i(x) \\ \quad + \sum_{j\in J} y_j^+ Dg_j(x), \\ h_i(x), \quad i \in I, \\ y_j^- - g_j(x), \quad j \in J \end{bmatrix} = 0 \tag{4}$$

(cf. e. g. [3, Ref. 134 and 135]). $\mathcal{H}(x, y)$ is called a *Kojima function*. We refer to the interesting article [7], and the references therein. Here, $f, h_i, g_j \in C^{1,1}(\mathbf{R}^n, \mathbf{R})$, $i \in I$, $j \in J$ (i. e., there exist derivatives which are locally Lipschitz). Two embeddings (parametrizations) of $\mathcal{H}(x, y, t) = 0$ are investigated. In particular, this permits new interpretations of the related solution methods (penalty and a new *barrier* method) and allows for estimates of the solutions by using implicit function theorems.

There are also possibilities to reformulate the KKT system into a k-times ($k \ge 1$) continuously differentiable nonlinear parameter-dependent system of equations (cf. e. g. [3, Ref. 62] and the references there). Then we can use standard methods for parameter-dependent equations (cf. the literature mentioned above). Furthermore, we can also consider the KKT system as a generalized equation (cf. [3, Ref. 193] and the references there) and use the corresponding path following methods (cf. [3, Ref. 177]).

Before we come to the second approach, we introduce a general *one- parametric nonlinear optimization* problem

$$\begin{aligned} &\min\{f(x,t)\colon\ x \in M(t)\}\,, \\ &t \in \mathbf{R}, \quad \text{resp. } t \in [0,1]\,, \\ &M(t) = \left\{ x \in \mathbf{R}^n\colon \begin{array}{l} h_i(x,t) = 0,\ i \in I\,, \\ g_j(x,t) \le 0,\ j \in J \end{array} \right\}\,, \\ &f, h_i, g_j \in C^k(\mathbf{R}^n \times \mathbf{R}, \mathbf{R})\,, \\ &i \in I, \quad j \in J, \quad k \ge 2\,, \\ &I := \{1, \ldots, m\}, \quad J := \{1, \ldots, s\}\,. \end{aligned}$$

Furthermore, we introduce the following notations:

$$\Sigma_{\text{gc}} := \left\{ (x,t) \in \mathbf{R}^n \times \mathbf{R}\colon \begin{array}{c} x \text{ a generalized} \\ \text{critical point of} \\ P(t) \end{array} \right\}\,,$$

$$\Sigma_{\text{stat}} := \left\{ (x,t) \in \mathbf{R}^n \times \mathbf{R}\colon \begin{array}{c} x \text{ a stationary} \\ \text{point of } P(t) \end{array} \right\}\,,$$

$$\Sigma_{\text{loc}} := \left\{ (x,t) \in \mathbf{R}^n \times \mathbf{R}\colon \begin{array}{c} x \text{ a local} \\ \text{minimizer of} \\ P(t) \end{array} \right\}\,,$$

(a generalized critical (g.c.) point is defined in [3] for instance).

We adapt the concept of embedding to the problem (P) using the problem $P(t)$ with at least the following properties:

A1) A local minimizer x^0 for $P(0)$ is known;

A2) $P(1)$ is equivalent to (P).

The conditions A1) and A2) are fulfilled if we choose e. g. $P(t)$ defined by

$$\begin{aligned} f(x,t) &:= tf(x) + (1-t)\left\|x - x^0\right\|^2\,, \\ h_i(x,t) &:= h_i(x) + (t-1)h_i(x^0), \quad i \in I\,, \\ g_j(x,t) &:= g_j(x) + (t-1)\left|g_j(x^0)\right|, \quad j \in J\,, \end{aligned}$$

where $x^0 \in \mathbf{R}^n$ is arbitrarily fixed.

Here too, we have to find a discretization (2) of the interval [0, 1] and corresponding points $(x(t_i), t_i) \in \Sigma_{\text{loc}}$ (Σ_{stat} or Σ_{gc}), $i = 1, \ldots, N$. We can also use a predictor-corrector scheme applied to the full problem $P(t)$ in some interval $[\underline{t}, \bar{t}] \subseteq [0, 1]$ (a modification

of the above Theorem holds, cf. [3]) or to a finite number of reduced problems with equality constraints only:

$$P^{J_0}(t): \quad \min \left\{ f(x,t): \begin{array}{l} h_i(x,t)=0,\ i\in I, \\ g_j(x,t)=0,\ j\in J_0, \end{array} \right\}, \quad t\in[t_s,t_{s+1}],$$

where J_0 is the index set of active constraints, which is constant in a certain interval $[t_s, t_{s+1}] \subseteq [0,1]$. We call this procedure an active index set strategy. In order to understand how difficult it is to find such a discretization (2), we give a very brief summary on two classes of functions (f, H, G) ($H = (h_1, \ldots, h_m)^\intercal$, $G = (g_1, \ldots, g_s)^\intercal$), namely, the Jongen–Jonker–Twilt class $\mathcal{F}$ (cf. [3] and the original articles [3, Ref. 115 and 118]) and the Kojima–Hirabayashi class (cf. [3, Ref. 135]).

Generic Singularities, the Approach via Piecewise Differentiability and Topological Stability

First, we introduce two well-known constraint qualifications.

The *linear independence constraint qualification* (LICQ) is satisfied at $\overline{x} \in M(\overline{t})$ if the vectors $D_x h_i(\overline{x},\overline{t})$, $i \in I$, $D_x g_j(\overline{x},\overline{t})$, $j \in J_0(\overline{x},\overline{t})$, are linearly independent ($J_0(x,t) := \{j \in J : g_j(x,t) = 0\}$).

The *Mangasarian–Fromovitz constraint qualification* (MFCQ) is satisfied at $\overline{x} \in M(\overline{t})$ if:

MF1) $D_x h_i(\overline{x},\overline{t})$, $i \in I$, are linearly independent

MF2) there exists a vector $\xi \in \mathbf{R}^n$ with

$$D_x h_i(\overline{x},\overline{t})\xi = 0, \quad i\in I,$$
$$D_x g_j(\overline{x},\overline{t})\xi < 0, \quad j\in J_0(\overline{x},\overline{t}).$$

We consider the Kojima function $\mathcal{H}(x, y, t)$ corresponding to $P(t)$ and $\mathcal{H}(x, y, t)$ is piecewise continuously differentiable (shortly PC^1) (see [4]). The classical definition of a regular value of a continuously differentiable function is generalized for PC^1 functions. Furthermore, it is shown that, if 0 is a regular value of $\mathcal{H}$, then the set $\mathcal{H}^{-1}(0)$ is PC^1 manifold (cf. [3, Ref. 135].

Next, we cite our short characterization of the class $\mathcal{F}$ introduced by H.Th. Jongen, P. Jonker and F. Twilt.

The space $C^3(\mathbf{R}^n \times \mathbf{R}, \mathbf{R})$ will be endowed with the strong (or Whitney-) C^3_s-topology, the C^3_s-topology of the product of a finite number of copies of $C^3(\mathbf{R}^n \times \mathbf{R}, \mathbf{R})$ being the induced product topology. A typical C^3_s base-neighborhood $\mathcal{N}$ of the zero function in $C^3(\mathbf{R}^n \times \mathbf{R}, \mathbf{R})$ is induced by means of a continuous positive function $\varepsilon\colon \mathbf{R}^n \times \mathbf{R} \to \mathbf{R}$ as follows:

$$\mathcal{N}_\varepsilon = \{\phi \in C^3(\mathbf{R}^n\times\mathbf{R}): \|\phi(z)\| + \sum \left\|\frac{\partial\phi}{\partial z_i}(z)\right\| + \sum \left\|\frac{\partial^2\phi}{\partial z_i \partial z_j}(z)\right\| + \sum \left\|\frac{\partial^3\phi}{\partial z_i\partial z_j\partial z_k}(z)\right\| < \varepsilon(z) \text{ for all } z\in\mathbf{R}^{n+1}\},$$

where $z = (x, t) \in \mathbf{R}^n \times \mathbf{R}$. A typical C^3_s base-neighborhood of $f \in C^3(\mathbf{R}^n \times \mathbf{R}, \mathbf{R})$ will be the set $f + \mathcal{N}_\varepsilon$.

The local structure of Σ_{gc} is completely described if (f, H, G) belongs to a C^3_s-open and dense subset $\mathcal{F}$ of $C^3(\mathbf{R}^n \times \mathbf{R}, \mathbf{R})^{1+m+s}$.

If $(f, H, G) \in \mathcal{F}$ then Σ_{gc} can be divided into 5 types:

- *Type 1*: A point $\overline{z} = (\overline{x}, \overline{t}) \in \Sigma_{\text{gc}}$ is of Type 1 if the following conditions are satisfied

$$\left(D_x f + \sum_{i\in I} \lambda_i D_x h_i + \sum_{j\in J_0(\overline{z})} \mu_j D_x g_j \right)\Big|_{z=\overline{z}} = 0, \tag{5}$$

$$\text{the LICQ is satisfied at } \overline{x} \in M(\overline{t}), \tag{6}$$

$$\mu_j \neq 0, \quad j \in J_0(\overline{z}), \tag{7}$$

$$D^2_x L(\overline{x})\big|_{T(\overline{z})} \quad \text{is nonsingular.} \tag{8}$$

Here, $D^2_x L$ is the Hessian of the Lagrangian

$$L = f + \sum_{i\in I} \lambda_i h_i + \sum_{j\in J_0(z)} \mu_j g_j,$$

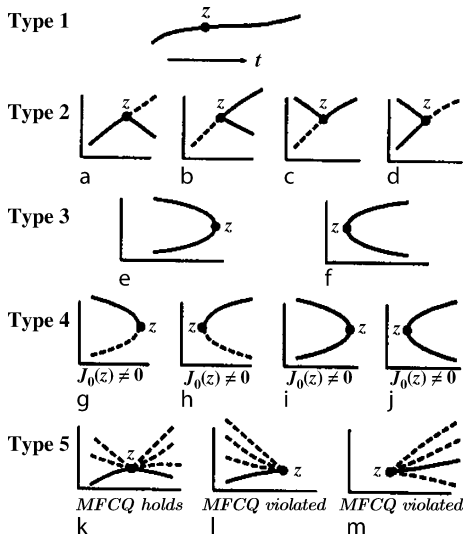
and the uniquely determined numbers $\lambda_1 \mu_j$ are taken from (5).

Furthermore,

$$T(z) = \left\{ \xi \in \mathbf{R}^n : \begin{array}{l} D_x h_i(z)\xi = 0,\ i\in I, \\ D_x g_j(z)\xi = 0,\ j\in J_0(z) \end{array} \right\}$$

is the tangent space at z. $D^2_x L(z)|_{T(z)}$ represents $V^\intercal D^2_x L V$, where V is a matrix whose columns form a basis of $T(z)$.

A point of Type 1 is called a *nondegenerate critical point*. The set Σ_{gc} is the closure of the set of all



Parametric Optimization: Embeddings, Path Following and Singularities, Figure 1

points of Type 1, the points of the Types 2–5 constitute a discrete subset of Σ_{gc}. The points of the Types 2–5 represent three basic degeneracies:

- *Type 2*: violation of (7).
- *Type 3*: violation of (8).
- *Type 4*: violation of (6) and

$$|I| + |J_0(\bar{z})| - 1 < n \,.$$

- *Type 5*: violation of (6) and

$$|I| + |J_0(\bar{z})| = n + 1 \,.$$

For each of these five types Fig. 1 illustrates the local structure of Σ_{gc}. Let Σ_{gc}^{ν}, $\nu \in \{1, \ldots, 5\}$ be the set of g.c. points of Type ν. Figure 2 illustrates the local structure of $\mathcal{F}$ in Σ_{loc} and Σ_{stat}.The class $\mathcal{F}$ is defined by

$$\mathcal{F} = \left\{(f, H, G) \in C^3(\mathbf{R}^n \times \mathbf{R}, \mathbf{R})^{1+m+s} : \Sigma_{gc} \subset \cup_{\nu=1}^{5} \Sigma_{gc}^{\nu}\right\} .$$

The following theorem (cf. [4, Ref. 15]) provides a special perturbation of (f, H, G) with additional parameters that can be chosen arbitrarily small such that the perturbed function vector belongs to the class $\mathcal{F}$.

Theorem 2 *Let $(f, H, G) \in C^3(\boldsymbol{R}^n \times \boldsymbol{R}, \boldsymbol{R}^{1+m+s})$. Then, for almost all $(b, A, c, D, e, F) \in \boldsymbol{R}^n \times \boldsymbol{R}^{n(n+1)/2} \times \boldsymbol{R}^m \times \boldsymbol{R}^{mn} \times \boldsymbol{R}^s \times \boldsymbol{R}^{sn}$, we have*

$$(f(x,t) + b^\top x + x^\top Ax, H(x,t) + c + Dx \,, \; G(x,t) + e + Fx) \in \mathcal{F} \,.$$

Here 'almost all' means: each measurable subset of

$$\{(b, A, c, D, e, F) \colon (f(x,t) + b^\top x + x^\top Ax, \; H(x,t) + c + Dx, G(x,t) + e + Fx) \notin \mathcal{F}\}$$

has Lebesgue-measure zero.

Remark 3 There is also a perturbation theorem for the Kojima–Hirabayashi class with a linear perturbation in the objective and scalar perturbations in the constraints (cf. [3, Ref. 135])

Remark 4 (cf. [3]) Considering Σ_{stat} we note that the condition $(f, H, G) \in \mathcal{F}$ implies that zero is a regular value of the Kojima-mapping $\mathcal{H}$.

Definition 5 Let $K \subseteq \mathbf{R} \cup \{\pm\infty\}$.

i) The problem $P(t)$ is called *regular in the sense of Jongen–Jonker–Twilt* (briefly: *JJT-regular*) (with respect to K) if

$$(f, H, G) \in \mathcal{F} \left((\mathbf{R}^n \times K) \cap \Sigma_{gc} \subset \bigcup_{\nu=1}^{5} \Sigma_{gc}^{\nu} \right) .$$

ii) The problem $P(t)$ is called *regular in the sense of Kojima–Hirabayashi* (briefly: *KH-regular*) (with respect to K) if $0 \in \mathbf{R}^{n+m+s}$ is a regular value of $\mathcal{H}$ $(\mathcal{H}|_{R^n \times R^m \times R^s \times K})$.

Now, we present some facts about path following methods (for more details see [3]). For this, we assume A1), A2) and that $(f, H, G) \in \mathcal{F}$. The algorithm PATH III (cf. [3, Chap. 4]) computes a numerical description of a compact connected component in Σ_{gc} and Σ_{stat}, respectively.

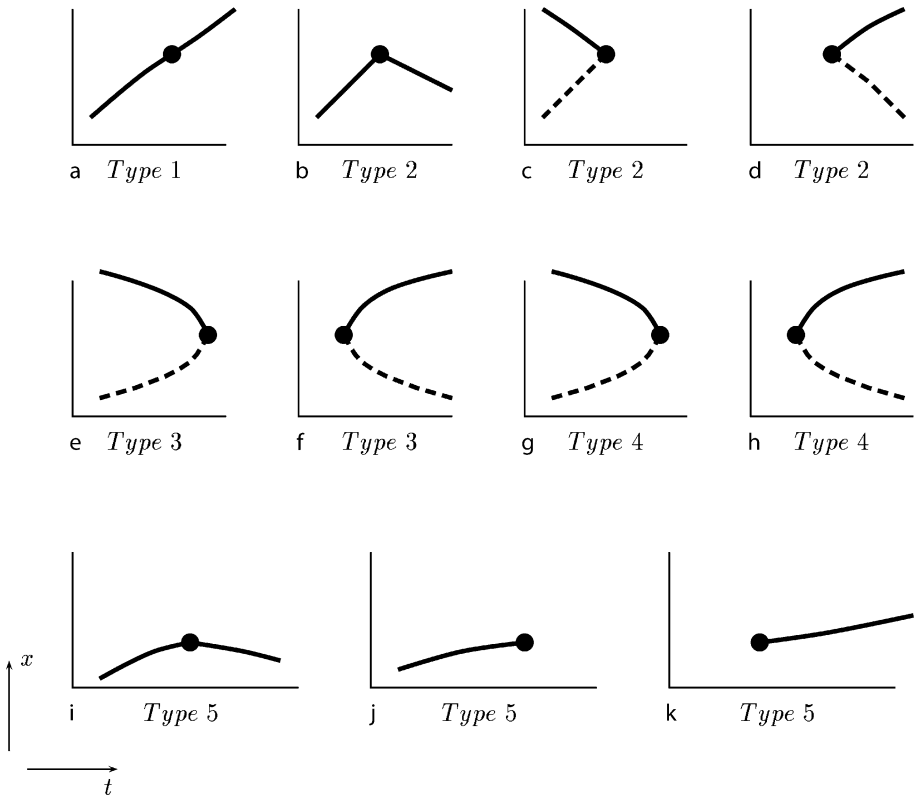
In the last part of this section we present two theorems that are essential for the analysis of considered embeddings in the literature and for modifying embeddings that are successful under certain assumptions.

Theorem 6 ([3, Ref. 71]) *We assume*

C1) $M(t)$ is nonempty and there exists a compact set C with $M(t) \subseteq C$ for all $t \in [0, 1]$.

C2) $P(t)$ is KH-regular with respect to $[0, 1]$.

C3) There exists a $t_1 > 0$ and a continuous function x: $[0, t_1) \to \boldsymbol{R}^n$ such that $x(t)$ is the unique stationary point for $P(t)$ for $t \in [0, t_1)$.



Parametric Optimization: Embeddings, Path Following and Singularities, Figure 2
The full curve stands for the curve of local minimizers and the dotted curve represents a curve of stationary points

C4) MFCQ is satisfied for all $x \in M(t)$ for all $t \in [0, 1]$. Then there exists a PC^1-path in Σ_{stat} that connects $(x^0, 0)$ with some point $(x^*, 1)$.

Applying our first Remark we obtain

Corollary 7 *We assume C1), C3), C4) and D2) P(t) is JJT-regular with respect to [0, 1]. Then there exists a PC^2-path $K(x^0, 0)$ in Σ_{stat} connecting $(x^0, 0)$ with some point $(x^*, 1)$. Furthermore, if $(x, t) \in K(x^0, 0)$, then (x, t) belongs to $\cup_{\nu \in \{1, 2, 3, 5\}} \Sigma^{\nu}_{gc}$.*

Finally we present a consequence of a general topological stability result given in [3, Ref. 87]:

Theorem 8 *We assume C1) and C4). Then $M(t_1)$ is homeomorphic with $M(t_2)$ for all $t_1, t_2 \in [0, 1]$.*

Remark 9 We see that the MFCQ plays an important role in both theorems above. Unless the MFCQ is satisfied, we do not attain a point $(\widehat{x}, 1) \in \Sigma_{\text{stat}}$ by path following methods only, g.c. points of the Types 4 and 5 may appear and the path ends in Σ_{stat}.

From this point of view we refer to the analysis of possible jumps from one connected component to another in Σ_{stat} and Σ_{gc}, respectively (cf. [3, Chap. 5]). In the worst case we have to find all connected components in Σ_{gc}. Since we do not have jumps in all cases, this problem has not been solved yet. This is not surprising because, in case we surely find a discretization (1) and a corresponding point $(x^i, t_i) \in \Sigma_{\text{gc}}$, $i = 1, \ldots, N$, the problem of global optimization is solved (cf. e. g. [3, Sect. 6.3]).

Concluding Remarks

i) We have restricted ourselves to *parametric optimization* problems in $\mathbf{R}^n$ so far. We refer to semi-infinite problems (cf. e. g. [6, pp. 161–176], and the references in that article).
ii) The theory and methods described in the section above are used for an analysis and appropriate modifications of the standard embedding (cf. [6, pp. 59–

84]), the penalty, exact penalty, and Lagrange multiplier embeddings (cf. [4] and [4, Refs. 2, 3, 6]), and for constructing new methods [2]. Summarizing, we have now more clearness concerning the problem of what kind of difficulties (mainly singularities) may appear and how to overcome them in some cases (see also [5]).

iii) We refer to further applications: One *parametric optimization* problems arising for instance in practical problems, multi-objective optimization, stochastic optimization etc. (cf. [3, Chap. 1]). Of course, on the one hand, the applications described there are not complete and, on the other hand, there is a stormy development in several fields.

iv) Finally, we would like to refer to so-called interior point methods, which are path following methods, too (cf. e. g. [8]).

See also

- ▶ Bounds and Solution Vector Estimates for Parametric NLPs
- ▶ Globally Convergent Homotopy Methods
- ▶ Multiparametric Linear Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Parametric Global Optimization: Sensitivity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Selfdual Parametric Method for Linear Programs
- ▶ Topology of Global Optimization

References

1. Allgower EL, Georg K (1990) Numerical continuation methods. An introduction. Springer, Berlin
2. Gomez W (1997) On generic quadratic penalty embeddings for nonlinear optimization problems. Preprint 97-18, Inst Math, Humboldt Univ (submitted)
3. Guddat J, Guerra F, Th. Jongen H (1990) Parametric optimization: singularities, pathfollowing and jumps. Teubner and Wiley, New York
4. Guddat J, Guerra F, Nowack D (1997) On the role of the Mangasarian–Fromovitz constraint qualification for penalty-, exact penalty-, and Lagrange multiplier methods. In: Fiacco AV (ed) Mathematical Programming with Data Perturbations. M. Dekker, New York, pp 159–183
5. Guddat J, Guerra F, Nowack D (2000) A parametric approach for solution methods in nonlinear optimization
6. Guddat J, Th. Jongen H, Nožička F, Still G, Twilt F (eds) (1997) Parametric optimization and related topics. Approx and Optim. P. Lang, Frankfurt am Main
7. Kummer B (1997) Parametrizations of Kojima's system and relations to penalty and barrier functions. Math Program 76:579–592
8. Nesterov Yu (1997) Interior point methods: An old and new approach to nonlinear programming. Math Program B 79:1–31; 285–299
9. Schwetlick H (1979) Numerische Lösung nichtlinearer Gleichungen. VEB Deutsch. Verlag Wissenschaft., Berlin
10. Wacker HJ (ed) (1978) Continuation methods. Acad. Press, New York

Peptide Identification via Mixed-Integer Optimization

Peter A. DiMaggio Jr.,
Christodoulos A. Floudas
Department of Chemical Engineering,
Princeton University, Princeton, USA

Article Outline

Keywords
Introduction
Formulation
  Mathematical Model for Peptide Identification
  Algorithmic Framework
Cases
  Quadrupole Time-of-Flight, QTOF, Spectra
Conclusions
References

Keywords

Mixed-integer linear optimization; Peptide identification; Tandem mass spectrometry

Introduction

Peptide and protein identification is of fundamental importance in the study of proteomics. Tandem mass spectrometry (MS/MS) coupled with high performance liquid chromatography (HPLC) has emerged as a powerful protocol for high-throughput and high sensitivity peptide and protein identification experiments. In

recognition of the extensive amount of sequence information embedded in a single mass spectrum, tandem MS has served as an impetus for the recent development of numerous computational approaches formulated to sequence peptides robustly and efficiently with particular emphasis on the integration of these algorithms into a high throughput computational framework. The two most frequently reported computational approaches in the literature are (a) de novo and (b) database search methods, both of which can utilize deterministic, probabilistic and/or stochastic solution techniques.

The majority of peptide identification methods used in practice are database search methods [2,11,19,21,30,31,32,36,37] due to their accuracy and ease of implementation. A variety of techniques for peptide identification via databases are currently available. One approach, as implemented in the SEQUEST algorithm [11,36,37], uses cross-correlation to mathematically determine the similarity between a theoretical tandem mass spectrum predicted from a peptide sequence in the database and the experimental tandem mass spectrum under investigation. The more frequently used technique, known as probability-based matching, utilizes a probabilistic model to determine whether an ion peak match between the experimental and theoretical tandem mass spectrum is actual or random [2,19,30,32]. Various models have been formulated for this purpose, ranging from a likelihood ratio hypothesis test [2,19] to the null hypothesis that peptide matches are random [32]. The major limitation of database methods is they are ineffective if the database in which the search is conducted does not contain the corresponding protein responsible for generating the tandem mass spectrum.

De novo methods are advantageous over database techniques since they can be used to find novel proteins, amino acid mutations and study the proteome before the genome. A prominent methodology for the de novo peptide identification problem is the spectrum graph approach [3,4,5,6,8,12,18,22,25,34,35]. In the majority of spectrum graph representations, the peaks in the tandem mass spectrum as translated as nodes on a directed graph, where the nodes are connected by edges if the mass difference between them is equal to the weight of an amino acid. The nodes or edges of the spectrum graph are typically assigned scores based on empirically-derived weights. Various alternative techniques to the spectrum graph approach have also been developed. For example, the de novo algorithm PEAKS [26] generates 10,000 potential sequences using a dynamic programming algorithm and then in a subsequent step reevaluates the predicted sequences using a stricter confidence scorer. Another technique addresses the peptide identification problem via stochastic optimization using genetic algorithms to solve multi-objective models and can empirically test for independence between scoring functions [20,27,28]. The algorithm NovoHMM [13] uses a hidden Markov model to solve the peptide identification problem, where the observable random variables are the observed mass peaks and the hidden variables correspond to the unknown peptide sequence. Despite the vast potential of de novo methods, they can be computationally demanding and may exhibit variable prediction accuracies.

In this work, we present a novel mixed-integer linear optimization (MILP) approach to efficiently address the de novo peptide identification problem so as to form a basis for a high-throughput computational framework for peptide identification [9,10]. This framework is denoted as **PILOT**, which stands for **P**eptide identification via Mixed-**I**nteger **L**inear **O**ptimization and **T**andem mass spectrometry.

Formulation

This section provides a thorough description of the mathematical model and algorithmic framework for the de novo identification of peptides of spectra resulting from tandem mass spectrometry.

Mathematical Model for Peptide Identification

The essential components of the mixed-integer linear programming problem formulation are the parameters, sets, binary variables, constraint equations and objective function [10].

Parameters Each tandem mass spectrum of a peptide contains the mass of the parent peptide, its charge state, and a list of mass-to-charge ratios and their corresponding intensities of the fragment ions of the peptide. It is important to note that the mass of the parent peptide and the masses of the ion peaks in the tandem mass

spectra are subject to a certain degree of experimental error [8]. The parameters are:

m_P	mass of parent peptide
mass(ion peak i)	mass of ion peak i
λ_i	intensity of ion peak i

Set Definitions The first set to consider is the mass difference between all of the peaks in the tandem mass spectrum, which we denote by the matrix M defined in Eq. (1).

$$M = \{M_{i,j} = mass(\text{ion peak } j) - mass(\text{ion peak } i) : mass(\text{ion peak } j) > mass(\text{ion peak } i)\} \quad (1)$$

The index i corresponds to the rows and the index j corresponds to the columns of the matrix $M_{i,j}$. The construction of the peptide sequence should be restricted to using only those peaks whose mass difference is equal to the weight of an amino acid. The indices corresponding to these peak pairs are stored in the matrix S, defined in Eq. (2).

$$S = \{S_{i,j} = (i, j) : M_{i,j} = \text{mass of an amino acid}\} \quad (2)$$

Thus, the mass difference between peak i and peak j is equal to the weight of some amino acid for every $(i, j) \in S_{i,j}$. The subsequent problem formulation will only be considered over this set, $S_{i,j}$.

The physical relationships between fragment ions can be used to construct additional sets. The sequencing of a candidate peptide must be done using ions from the *same* ion series (i. e., **b**, **y**, etc.). While it is not known a priori of what ion type a given mass peak is, there do exist important relationships among the different ions. As the charged parent peptide undergoes collision-induced dissociation (CID), it primarily fragments into two ion pairs: either **a** and **x**, **b** and **y**, or **c** and **z**, where all three pairs are what are known as *complementary ions* by definition. These ion pairs are easily identified a priori since the sum of two complementary ions is equal to the weight of the parent peptide, m_P, as determined experimentally. The indices of peak pairs which satisfy this relationship are stored in the set C, defined in Eq. (3).

$$C = \{C_{i,j} = (i, j) : mass(\text{ion peak } i) + mass(\text{ion peak } j) = m_P + 2;\ i \neq j\} \quad (3)$$

This set will be useful for eliminating certain ions in the sequencing calculations. However, one should note that further fragmentation of these ions is possible and frequently observed, which places limitations on how many complementary ions are actually detected in a tandem mass spectrum.

Different types of ion series begin and end at different m/z values in the tandem mass spectrum. For instance, a candidate peptide derived using the **y**-ion series must begin at the weight of water (19 Daltons) and terminate at the weight of the parent peptide ($m_P + 1$), whereas deriving the same sequence using the **b**-ion series, the appropriate bounds become zero mass (1 Dalton) and the weight of the parent peptide subtracted by the weight of water ($m_P - 17$), in respective order. To model this mathematically, two new sets are created which correspond to the boundary conditions at the "head" of the peptide and the "tail" of the peptide. Note that the sets presented below consider *only* the possibility for **b**- or **y**-ions in the candidate sequence.

$$BC_i^{\text{head}} = \{1, 19\} \text{ Daltons} \quad (4)$$

$$BC_j^{\text{tail}} = \{m_P - 17,\ m_P + 1\} \text{ Daltons} \quad (5)$$

Binary Variables Binary {0-1} variables are utilized in the problem formulation to model which peaks (p_i) and paths connecting peaks ($w_{i,j}$) are used in the construction of the candidate sequence. The use of binary variables also allows us to invoke logical inference when formulating the model constraints.

$$p_i = \begin{cases} 1, \text{ if peak } (i) \text{ is selected} \\ 0, \text{ otherwise} \end{cases}$$

$$w_{i,j} = \begin{cases} 1, \text{ if peaks } (i) \text{ and } (j) \text{ are connected by a path } (i.e., p_i = p_j = 1) \\ 0, \text{ otherwise} \end{cases}$$

Constraints Several constraints derived from ion properties and graph theory are formulated in terms of the binary variables via logical inference. The first constraint exploits the fact that the candidate peptide

must be sequenced using ions of the same type and that complementary ions are of different type by definition. Thus, if peak i is used to construct a candidate sequence and the peak pair (i, j) belong to the complementary ion set defined in Eq. (3), then peak j should be eliminated from consideration in the sequencing calculations. This is modeled mathematically by Eq. (6).

$$p_i + p_j \leq 1 \quad \forall\, (i,j) \in C_{i,j} \tag{6}$$

An obvious but important constraint to impose on the candidate sequence is that the summation of the weights of its amino acids is equal to the mass of the parent peptide (m_P). It is well-known that the experimentally measured parent peptide mass is subject to a certain degree of experimental error [8], which is dependent on the resolution of the mass spectrometer used. Thus, exact conservation of mass cannot be achieved but must be relaxed by some tolerance of error, as shown in Eq. (7).

$$\sum_{(i,j)\in S_{i,j}} M_{i,j} \cdot w_{i,j} \leq (m_P - 18) + tolerance \tag{7}$$

$$\sum_{(i,j)\in S_{i,j}} M_{i,j} \cdot w_{i,j} \geq (m_P - 18) - tolerance \tag{8}$$

The algorithm typically uses a tolerance of error of ± 2 Daltons above and below the parent peptide mass. It is also possible to formulate the *tolerance* term as a variable and then incorporate it into the model such that its value is minimized.

The sequencing of the candidate peptide is best envisioned as connecting peaks in the tandem mass spectrum with paths that correspond to weights of amino acids. To ensure that the paths selected are continuous and non-degenerate, we use the *flow conservation law* from graph theory which has been used extensively in process synthesis problems [1,15,16,17,24,29,33], as shown in Eq. (9).

$$\sum_{j\in S_{j,i}} w_{j,i} - \sum_{k\in S_{i,k}} w_{i,k} = 0 \qquad \forall i,\ i \notin BC_i^{\text{head}},\ i \notin BC_i^{\text{tail}} \tag{9}$$

The above constraints ensure that the number of input paths entering a peak is equal to the number of output paths leaving that peak.

To anchor the beginning and end of the candidate sequence, the peaks denoted as the boundary conditions for the sequencing are activated, as shown in Eqs. (10) and (11).

$$\sum_{i\in BC_i^{\text{head}}} \sum_{j\in S_{i,j}} w_{i,j} = 1 \tag{10}$$

$$\sum_{j\in BC_j^{\text{tail}}} \sum_{i\in S_{i,j}} w_{i,j} = 1 \tag{11}$$

One should note that the existence for certain boundary condition elements (contained in the sets BC^{head} and BC^{tail}) are checked for by the preprocessing algorithm (described in a subsequent section) and can be adjusted if information is missing from the tandem mass spectrum. Furthermore, these constraints enforce the non-degeneracy of paths since *only one* path can initiate and terminate the sequence, respectively.

The final set of constraints constitutes the mathematical relationship between the binary variables representing the peaks, p_i, and the paths connecting the peaks, $w_{i,j}$:

$$\sum_{j\in S_{\{i,j\}}} w_{i,j} = p_i \qquad \forall i \in BC_i^{\text{head}} \tag{12}$$

$$\sum_{j\in S_{i,j}} w_{j,i} = p_i \qquad \forall i \notin BC_i^{\text{head}} \tag{13}$$

These constraints ensure that if there exists a path entering and leaving a peak k (i. e., $w_{i,k} = 1$ and $w_{k,j} = 1$), then peak k will be activated in the construction of the candidate sequence (i. e., $p_k = 1$). These constraints also allow the option for removing peaks (and the paths connected to these peaks) from the sequencing calculations by simply deactivating the binary variables that represents those peaks (p_i). For instance, this is useful for eliminating the precursor ion and multiply-charged ions from consideration.

Objective Function The **b**- and **y**-ion peaks are on average the most abundant in intensity throughout the entire m/z range [23]. Based on this observation, the objective function is postulated as an explicit function of the peak intensities in attempt to maximize the number of **b**- and **y**-ions used in constructing the candidate

sequence.

$$\underset{p_k,\, w_{i,j}}{MAX} \sum_{(i,j)\in S_{i,j}} \lambda_j \cdot w_{i,j} \tag{14}$$

Equations (1–14) comprise the entire mathematical model for the de novo peptide sequencing problem using tandem mass spectrometry. The entire problem formulation is summarized in problem (P).

$$\begin{aligned}
&\underset{p_k,\, w_{i,j}}{MAX} \sum_{(i,j)\in S_{i,j}} \lambda_j \cdot w_{i,j} \\
&s.t. \sum_{(i,j)\in S_{i,j}} M_{i,j} \cdot w_{i,j} \le m_P + tolerance \\
&\sum_{(i,j)\in S_{i,j}} M_{i,j} \cdot w_{i,j} \ge m_P - tolerance \\
&p_i + p_j \le 1 \quad \forall\,(i,j) \in C_{i,j} \\
&\sum_{j\in S_{i,j}} w_{i,j} = p_i \quad \forall i \in BC_i^{\text{head}} \\
&\sum_{j\in S_{i,j}} w_{j,i} = p_i \quad \forall i \notin BC_i^{\text{head}} \\
&\sum_{i\in BC_i^{\text{head}}} \sum_{j\in S_{i,j}} w_{i,j} = 1 \qquad (P) \\
&\sum_{j\in BC_j^{\text{tail}}} \sum_{i\in S_{i,j}} w_{i,j} = 1 \\
&\sum_{j\in S_{j,i}} w_{j,i} - \sum_{k\in S_{i,k}} w_{i,k} = 0 \\
&\forall i,\ i \notin BC_i^{\text{head}},\ i \notin BC_i^{\text{tail}} \\
&w_{i,j},\ p_k = \{0,1\} \quad \forall (i,j),(k)
\end{aligned}$$

The resulting problem (P) is a mixed-integer linear programming (MILP) problem and can be solved to optimality using existing methods (e. g., CPLEX [7]). To generate a rank-ordered list of candidate sequences, integer cuts are used to exclude previous solutions from be revisited. That is, for every solution, an integer cut is incorporated into the model using the following general form[14]:

$$\sum_{(i,j)\in B} w_{i,j} - \sum_{(i,j)\in NB} w_{i,j} \le |B| - 1 \tag{15}$$

where $B = \{(i,j) : w_{i,j} = 1\}$

$NB = \{(i,j) : w_{i,j} = 0\}$

$|B|$ is the cardinality of B

Peptide Identification via Mixed-Integer Optimization, Table 1
Ions identified by the preprocessing algorithm

Ion Type	Relation to the $\mathbf{b}_2$-ion
$\mathbf{y}_{n-2}$	$m_P + 2 - \mathbf{b}_2$
$\mathbf{a}_2$	$\mathbf{b}_2$ - 28
$\mathbf{b}_1$	AA_1 or AA_2 where $AA_1 + AA_2 + 1 = \mathbf{b}_2$
$\mathbf{y}_{n-1}$	$m_P + 2 - AA_1$ or $m_P + 2 - AA_2$ where $AA_1 + AA_2 + 1 = \mathbf{b}_2$

Algorithmic Framework

The overall algorithm PILOT is comprised of: (1) a preprocessing algorithm used to identify certain peaks and to validate boundary conditions, (2) a two-stage mixed-integer linear optimization framework to address missing ion peaks due to residue-dependent fragmentation characteristics, and (3) a post-processing technique for selecting the most probable sequence by cross-correlating the theoretical tandem mass spectra of the candidate sequences with the experimental tandem mass spectrum [9].

Preprocessing of Spectral Data Before formulating the MILP problem, the raw tandem mass spectrum is analyzed using a preprocessing algorithm to elucidate key spectral features that can be exploited in the sequencing calculations. In particular, certain ion peaks are sought to confirm the proposed boundary conditions previously mentioned. First, the raw spectrum is examined for the existence of the typically abundant in intensity $\mathbf{b}_2$-ion [23], whose validity can be confirmed by its complementary $\mathbf{y}_{n-2}$-ion. If the corresponding $\mathbf{y}_{n-2}$-ion also exists in the spectrum, then the two possible $\mathbf{y}_{n-1}$-ions are back-calculated using the mass of the parent peptide (m_P) and the weights of the amino acids which constitute the $\mathbf{b}_2$-ion (see Table 1), and the spectrum is once again searched to confirm these ions.

Every $\mathbf{b}_2$-ion amino acid pair found by the algorithm is then ranked based on the intensities of their supporting ions (i. e., $y_{\{n-2\}}$, $y_{\{n-1\}}$, $a_{\{2\}}$ and their isotopic offsets and neutral losses of water and ammonia). The highest scoring $\mathbf{b}_2$-ion amino acid pair is then selected to represent N-terminal boundary conditions for the sequencing calculations. The preprocessing algorithm is also used to identify the C-terminal amino

acid based on known ion peaks. For example, if the peptide is a result of tryptic digestion, then its C-terminal amino acid must be lysine (indicated by an ion peak at 147 Da) or arginine (indicated by an ion peak at 175 Da). Multiply-charged ions and immonium ions are also searched for in the experimental tandem mass spectrum to guide the interpretation process.

Missing Ion Peaks: Two-Stage Algorithmic Approach In the mixed-integer linear optimization model (see problem P), the algorithm attempts to derive several candidate sequences using only *single* amino acid weights to connect mass peaks. However, if the spectrum is missing certain mass peaks then it might not be possible to fully construct the correct sequence in this fashion. To accommodate this issue, the sequencing problem is split into two stages: stage one derives the candidate peptides using only single amino weights and stage two allows for the possibility of using *two* amino acid weights. That is to say, in the second stage the additional option is available to connect two peaks via the weight of any two combined amino acids. In this stage the emphasis is again placed on primarily using the weights of single amino acids to construct the candidate sequences by penalizing the use of combined amino acid weights. This is accomplished in the objective function by multiplying the path variable corresponding to multiple residues by a penalty weighting fraction which is less than one and decreases with increasing mass error. As a result, the driving force for the algorithm is the single residue weights, while the double residue weights are utilized only to bridge the gap between disjoint single residue segments of the candidate sequence.

Postprocessing: Scoring Candidate Sequences A subsequent scoring of the candidate peptide sequences is necessary in order to: (a) assign amino acids to regions of the peptide not considered in the sequencing calculations due to boundary condition adjustments, (b) resolve doublet and triplet amino acid combinations due to missing peaks, and (c) validate the "**b**" or "**y**" ions used to construct the candidate sequence by looking for other supporting ions in the raw tandem mass spectrum. For cases (a) and (b), the weight in the candidate sequence is replaced by permutations of amino acids consistent with this mass. This results in a super set of candidate sequences whose theoretical tandem mass spectra can be predicted and compared to the experimental tandem mass spectrum for validity.

Since only the **y**- or **b**- ion series were used in constructing the peptide sequences, it would be beneficial to utilize various other types of ions when scoring these candidate peptides in order to exploit as much information from the tandem mass spectrum as possible. In particular, the assignment of a peak as a **b**- or **y**-ion can be confirmed by the existence of supporting isotopes, neutral losses of small molecules (i. e.,-H_2O, -NH_3, -H_2O-NH_3, -H_2O-H_2O) and multiply-charged ions. To introduce dependencies among the ion series, a reward/penalty system was created. For instance, a match between a predicted **y**-ion and a peak in the experimental spectrum is more probable if the corresponding **y**-ions isotopes and offsets are also found in the experimental spectrum [18]. Thus, the score from a match between **b**- or **y**-ion is assigned a reward proportional to the number of its corresponding isotopes and offsets that also match with the experimental spectrum. Conversely, the existence of isotopic offsets and and neutral loss ions *without* a corresponding **y**- or **b**-ion are penalized in the score. These conventions address the likelihood that the peaks used in the construction of the candidate sequence are actually of the **b**- or **y**-ion series.

Cases

The proposed framework is for the de novo identification of doubly-charged tryptic peptides that were ionized via electrospray ionization. A comparative study is presented with several existing de novo peptide identification methods to demonstrate the predictive capabilities of the proposed framework, PILOT. The de novo peptide identifications algorithms Lutefisk, LutefiskXP, PepNovo, PEAKS, EigenMS, NovoHMM, were selected on the basis of availability and reported performance. Tandem MS for quadrupole time-of-flight mass spectrometers were analyzed in the comparative study.

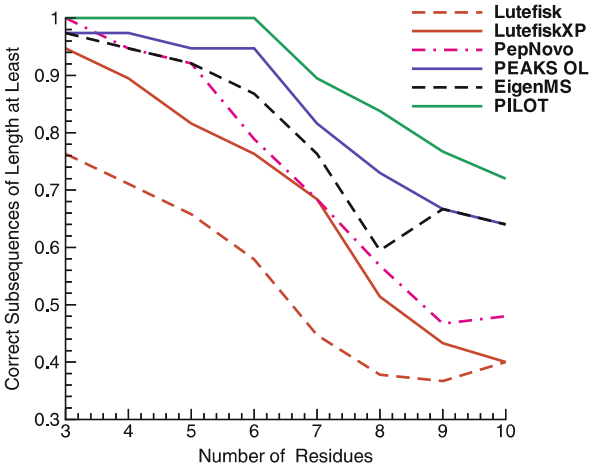
Quadrupole Time-of-Flight, QTOF, Spectra

To test the method's performance on quadrupole time-of-flight tandem mass spectra, we selected an existing data set that is publicly available [26]. These spectra were collected with Q-TOF2 and Q-TOF-Global

mass spectrometers for a control mixture of four known proteins: alcohol dehydrogenase (yeast), myoglobin (horse), albumin (bovine, BSA), and cytochrome C (horse). A total of 38 doubly-charged tryptic peptides were examined using Lutefisk, LutefiskXP, PepNovo, PEAKS, EigenMS and PILOT [9]. The top-ranked sequence reported from each of these methods is used in the comparison.

The results of the comparison can be summarized using various measures. First, consider the overall peptide identification accuracy of the de novo methods, which is reported in Table 2. In terms of correct identifications, PILOT is superior to the other de novo methods with an identification rate of about 66%, followed by PEAKS and EigenMS, both at about 53%. A common limitation of de novo methods is the inability to assign the correct N-terminal amino acid pair or resolve isobaric residues (i. e., Q or GA, W or SV, etc.). To accommodate this limitation in the comparison, we also reported the percentage of predictions for which there are only one, two, or three *incorrect* amino acid assignments in the entire sequence. In Table 2, it is seen that allowing for up to three *incorrect* amino acids increases the identification rate for all methods on the order of 30%, which supports the hypothesis that these limitations are inherent in all the de novo methods. The last entry in Table 2 reports the number of correctly assigned residues normalized by the total number of actual residues (which is 418 for the 38 doubly-charged peptides considered). PILOT outperforms the other de novo methods with a residue accuracy of 91%.

Another alternative metric of performance is the percentage of correct continuous subsequences of a given number of amino acids [13,18]. These trends are summarized in Fig. 1 for all of the de novo methods. This alternative metric reveals that although LutefiskXP has a lower overall identification rate than Lutefisk, it exhibits a much better accuracy over subsequences of varying length (as shown in Fig. 1). Note that some of the trends in Fig. 1 exhibit an increase in accuracy for correct subsequences greater than 8 consecutive amino acids in length. This is because these counts are normalized by the total number of peptides that are of *at least* the length specified, which decreases from 37 for subsequences of length 8 to 30 for subsequences of length 9 to 25 for subsequences of length 10. For each of the 38 QTOF peptides, PILOT predicts at least 6 consecutive amino acids correctly for every peptide and performs consistently better than the other de novo methods over the entire range of subsequences considered.



Peptide Identification via Mixed-Integer Optimization, Figure 1
Comparison of correct subsequences of varying length for quadrupole time-of-flight predictions

Conclusions

A novel mixed-integer linear optimization (MILP) framework, PILOT, was proposed for the de novo identification of peptides using tandem mass spectrometry data. PILOT is the first reported mixed-integer linear optimization formulation for the peptide identification problem which can introduce integer cuts to generate a rigorous rank-ordered list of candidate sequences, introduce complementary ions directly into the sequencing calculations and allow for the error tolerance to be introduced as a variable term. For a given experimental MS/MS spectrum, PILOT generates a rank-ordered list of potential candidate sequences and a cross-correlation technique is employed to select the most probable peptide by assessing the degree of similarity between the theoretical tandem mass spectra of the predicted sequences and the experimental tandem mass spectrum. A comparative study using tandem mass spectra from quadrupole time-of-flight was presented to benchmark the performance of the proposed framework with several existing de novo methods.

Peptide Identification via Mixed-Integer Optimization, Table 2
Identification rates for QTOF spectra

	Lutefisk	LutefiskXP	PepNovo	PEAKS Online	EigenMS	PILOT
Correct Identifications	10 (0.263)	9 (0.237)	16 (0.421)	21 (0.553)	20 (0.526)	25 (0.658)
with in 1 Residue	11 (0.290)	10 (0.263)	17 (0.447)	22 (0.579)	21 (0.553)	25 (0.658)
with in 2 Residue	23 (0.605)	22 (0.579)	25 (0.658)	29 (0.763)	29 (0.763)	33 (0.868)
with in 3 Residue	23 (0.605)	25 (0.658)	27 (0.711)	32 (0.842)	30 (0.790)	35 (0.921)
Total Correct Residues	245 (0.586)	294 (0.703)	337 (0.806)	366 (0.876)	353 (0.845)	381 (0.912)

References

1. Aggarwal A, Floudas CA (1990) Synthesis of general separation sequences – nonsharp separations. Comp Chem Eng 14(6):631–653
2. Bafna V, Edwards N (2001) SCOPE: a probabilistic model for scoring tandem mass spectra against a peptide database. Bioinformatics 17:S13–S21
3. Bern M, Goldberg D (2006) De novo analysis of peptide tandem mass spectra by spectral graph partitioning. J Comp Biol 13(2):364–378
4. Cannon WR, Jarman KD (2003) Improved peptide sequencing using isotope information inherent in tandem mass spectra. Rapid Commun Mass Spectrom 17:1793–1801
5. Chen T, Bingwen L (2003) A suboptimal algorithm for de novo peptide sequencing via tandem mass spectrometry. J Comp Biol 10(1):1–12
6. Chen T, Kao MY, Tepel M, Rush J, Church GM (2001) A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. J Comp Biol 10(3):325–337
7. CPLEX (2005) ILOG CPLEX 9.0 User's Manual. ILOG S.A., Gentilly
8. Dancik V, Addona TA, Clauser KR, Vath JE, Pevzner PA (1999) De novo peptide sequencing via tandem mass spectrometry. J Comp Biol 6(3):327–342
9. DiMaggio PA, Floudas CA (2007) De novo peptide identification via tandem mass spectrometry and integer linear optimization. Anal Chem 79:1433–1446
10. DiMaggio PA, Floudas CA (2007) A mixed-integer optimization framework for de novo peptide identification. AIChE J 53(1):160–173
11. Eng JK, McCormack AL, Yates JR (1994) An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. J Am Soc Mass Spectrom 5:976–989
12. Fernandez de Cossio J, Gonzalez J, Satomi Y, Shima T, Okumura N, Besada V, Betancourt L, Padron G, Shimonishi Y, Takao T (2000) Automated interpretation of low-energy collision-induced dissociation spectra by seqms, a software aid for de novo sequencing by tandem mass spectrometry. Electrophoresis 21:1694–1699
13. Fischer B, Roth V, Roos F, Grossmann J, Baginsky S, Widmayer P, Gruissem W, Buhmann JM (2005) NovoHMM: A hidden Markov model for de novo peptide sequencing. Anal Chem 77:7265–7273
14. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization. Oxford University Press, New York
15. Floudas CA, Anastasiadis SH (1988) Synthesis of general distillation sequences with several multicomponent feeds and products. Chem Eng Sci 43(9):2407–2419
16. Floudas CA, Ciric AR (1989) A retrofit approach of heat exchanger networks. Comp Chem Eng 13(6):703–715
17. Floudas CA, Grossmann IE (1987) Synthesis of flexible heat exchanger networks with uncertain flowrates and temperatures. Comp Chem Eng 11(4):319–336
18. Frank A, Pevzner P (2005) PepNovo: De novo peptide sequencing via probabilistic network modeling. Anal Chem 77(4):964–973
19. Havilio M, Haddad Y, Smilansky Z (2003) Intensity-based statistical scorer for tandem mass spectrometry. Anal Chem 75(3):435–444
20. Heredia-Langner A, Cannon WR, Jarman KD, Jarman KH (2004) Sequence optimization as an alternative to de novo analysis of tandem mass spectrometry data. Bioinformatics 20(14):2296–2304
21. Hernandez P, Gras R, Frey J, Appel RD (2003) Popitam: Towards new heuristic strategies to improve protein identification from tandem mass spectrometry data. Proteomics 3:870–878
22. Jarman KD, Cannon WR, Jarman KH, Heredia-Langner A (2003) A model of random sequences for de novo peptide sequencing. In: 3rd IEEE Int Symp BioInf BioEng. IEEE Computer Society, Los Alamitos, pp 206–213
23. Kinter M, Sherman NE (2000) Protein Sequencing and Identification using Tandem Mass Spectrometry. Wiley, New York
24. Kokossis AC, Floudas CA (1994) Optimization of complex reactor networks-II: nonisothermal operation. Chem Eng Sci 49(7):1037–1051
25. Lubeck O, Sewell C, Gu S, Chen X, Cai DM (2002) New computational approaches for de novo peptide sequencing from ms/ms experiments. Proc IEEE 90(12):1868–1874
26. Ma B, Zhang KZ, Hendrie C, Liang C, Li M, Doherty-Kirby A, Lajoie G (2003) PEAKS: powerful software for peptide de

novo sequencing by tandem mass spectrometry. Rapid Commun Mass Spectrom 17:2337–2342
27. Malard JM, Heredia-Langner A, Baxter DJ, Jarman KH, Cannon WR (2004) Constrained de novo peptide identification via multi-objective optimization. In: HiCOMB Proc 2004. Santa Fe
28. Malard JM, Heredia-Langner A, Cannon WR, Mooney R, Baxter DJ (2005) Peptide identification via constrained multi-objective optimization: Pareto-based genetic algorithms. Concurrency Computat: Pract Exper 17:1–18
29. Paules GE IV, Floudas CA (1989) APROS: Algorithmic development methodology for discrete-continuous optimization problems. Oper Res J 37(6):902–915
30. Perkins DN, Pappin DJC, Creasy DM, Cottrell JS (1999) Probability-based protein identification by searching sequence databases using mass spectrometry data. Electrophoresis 20:3551–3567
31. Pevzner PA, Mulyukov Z, Dancik V, Tang CL (2001) Efficiency of database search for identification of mutated and modified proteins via mass spectrometry. Genome Res 11:290–299
32. Sadygov RG, Yates JR (2003) A hypergeometric probability model for protein identification and validation using tandem mass spectral data and protein sequence databases. Anal Chem 75:3792–3798
33. Schrijver A (1986) Theory of Linear and Integer Programming. Wiley, West Sussex
34. Taylor JA, Johnson RS (1997) Sequence database searches via de novo peptide sequencing by tandem mass spectrometry. Rapid Commun Mass Spectrom 11:1067–1075
35. Taylor JA, Johnson RS (2001) Implementation and uses of automated de novo peptide sequencing by tandem mass spectrometry. Anal Chem 73:2594–2604
36. Yates JR, Eng JK, McCormack AL (1995) Mining genomes: correlating tandem mass spectra of modified and unmodified peptides to sequences in nucleotide databases. Anal Chem 67:3202–3210
37. Yates JR, Eng JK, McCormack AL, Schieltz D (1995) Method to correlate tandem mass spectra of modified peptides to amino acid sequences in the protein database. Anal Chem 67:1426–1436

Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization

Neculai Andrei[1,2]
[1] Research Institute for Informatics, Center for Advanced Modeling and Optimization, Bucharest, Romania
[2] Academy of Romanian Scientists, Bucharest, Romania

MSC2000: 49M07, 49M10, 90C06, 65K05

Article Outline

Keywords and Phrases
Introduction
Classical Conjugate-Gradient Algorithms
Hybrid Conjugate-Gradient Methods
Scaled Conjugate-Gradient Algorithms
SCALCG Algorithm
Modified Conjugate-Gradient Algorithms
Parametric Conjugate-Gradient Algorithms
Performance Profiles
Conclusion and Discussion
References

Keywords and Phrases

Unconstrained optimization; Conjugate gradient; Hybrid conjugate gradient; Scaled conjugate gradient; Conjugacy condition; Numerical comparisons; Dolan-Moré profile.

Introduction

Conjugate-gradient methods represent an important class of unconstrained optimization algorithms with strong local and global convergence properties and modest memory requirements. An excellent survey of the development of different versions of nonlinear conjugate-gradient methods, with special attention to global convergence properties, is presented by Hager and Zhang [22]. This family of algorithms includes a lot of variants, well known in the literature, with important convergence properties and numerical efficiency. The purpose of this chapter is to present these algorithms as well as their performances to solve a large variety of large-scale unconstrained optimization problems.

For solving the nonlinear unconstrained optimization problem

$$\min \{f(x) : x \in R^n\} \ , \tag{1}$$

where $f : R^n \to R$ is a continuously differentiable function bounded from below, starting from an initial guess

$x_0 \in R^n$ a nonlinear conjugate-gradient method generates a sequence $\{x_k\}$ as

$$x_{k+1} = x_k + \alpha_k d_k \ , \tag{2}$$

where $\alpha_k > 0$ is obtained by line search, and the directions d_k are generated as

$$d_{k+1} = -g_{k+1} + \beta_k s_k, \quad d_0 = -g_0 \ . \tag{3}$$

In (3) β_k is known as the conjugate-gradient parameter, $s_k = x_{k+1} - x_k$ and $g_k = \nabla f(x_k)$. Consider $\|.\|$ the Euclidean norm and define $y_k = g_{k+1} - g_k$. The line search in the conjugate-gradient algorithms is often based on the standard Wolfe conditions:

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \rho \alpha_k g_k^T d_k \ , \tag{4}$$

$$g_{k+1}^T d_k \geq \sigma g_k^T d_k \ , \tag{5}$$

where d_k is a descent direction and $0 < \rho \leq \sigma < 1$. For some conjugate-gradient algorithms, stronger versions of the Wolfe conditions are needed to ensure convergence and to enhance stability. According to the formula for β_k computation, the conjugate-gradient algorithms can be classified as classical, hybrid, scaled, modified and parametric. In the following we shall present these algorithms and insist on their numerical Dolan and Moré 's performances profiles for solving large-scale unconstrained optimization problems.

The history of conjugate-gradient method begins with the seminal paper of Hestenes and Stiefel [23], who presented an algorithm for solving symmetric, positive-definite linear algebraic systems. In 1964 Fletcher and Reeves [18] extended the domain of application of conjugate-gradient method to nonlinear problems, thus starting the nonlinear conjugate-gradient research direction. The main advantages of the conjugate-gradient method are its low memory requirements and its convergence speed. A large variety of nonlinear conjugate-gradient algorithms are known. For each of them convergence results have been proved in mild conditions which refer to the Lipschitz and boundedness assumptions. To prove the global convergence of nonlinear conjugate-gradient methods, often the Zoutendijk condition is used combined with analysis showing that the sufficient descent condition $g_k^T d_k \leq -c \|g_k\|^2$ holds, and that there exists a constant δ such that $\|d_k\|^2 \leq \delta k$. Often, the convergence analysis of conjugate-gradient algorithms, for general nonlinear functions, follows insights developed by Gilbert and Nocedal [19]. The idea is to bound the change $u_{k+1} - u_k$ in the normalized direction $u_k = d_k / \|d_k\|$, which is used to conclude, by contradiction, that the gradients cannot be bounded away from zero.

Classical Conjugate-Gradient Algorithms

These algorithms are defined by (2) and (3), where the parameter β_k is computed as in Table 1. Observe that these algorithms can be classified as algorithms with $\|g_{k+1}\|^2$ in the numerator of β_k and algorithms with $g_{k+1}^T y_k$ in the numerator of parameter β_k.

The FR, CD and DY methods (see the tables for the definitions of the acronyms used for the algorithms throughout the text), with $\|g_{k+1}\|^2$ in the numerator of β_k, have strong convergence theory, but all

Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Table 1
Classical conjugate-gradient algorithms

No.	Formula	Author(s)
1.	$\beta_k^{HS} = \frac{y_k^T g_{k+1}}{y_k^T s_k}$	Hestenes and Stiefel [23] (HS). The first conjugate-gradient algorithm for linear algebraic systems
2.	$\beta_k^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$	Fletcher and Reeves [18] (FR). The first conjugate-gradient algorithm for nonlinear functions
3.	$\beta_k^{PRP} = \frac{y_k^T g_{k+1}}{g_k^T g_k}$	Polak-Ribiere [31] and Polyak [32] (PRP)
4.	$\beta_k^{PRP+} = \max\left\{0, \frac{y_k^T g_{k+1}}{g_k^T g_k}\right\}$	Polak-Ribiere and Polyak + (PRP+) suggested by Powell [33]
5.	$\beta_k^{CD} = -\frac{g_{k+1}^T g_{k+1}}{g_k^T d_k}$	Conjugate descent (CD) introduced by Fletcher [17]
6.	$\beta_k^{LS} = -\frac{y_k^T g_{k+1}}{g_k^T d_k}$	Liu and Storey [26] (LS)
7.	$\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$	Dai and Yuan [13] (DY)

these methods are susceptible to jamming. They begin to take small steps without making any significant progress to the minimum. On the other hand, the HS, PRP and LS methods, with $g_{k+1}^T y_k$ in the numerator of parameterβ_k, have a built-in restart feature that addresses the jamming phenomenon. When the step s_k is small, the factor $y_k = g_{k+1} - g_k$ in the numerator of β_k tends to zero. Therefore, β_k becomes small and the new direction d_{k+1} in (3) is essentially the steepest descent direction $-g_{k+1}$. With other words, the HS, PRP and LS methods automatically adjust β_k to avoid jamming, and their performances are better than the performance of methods with $\|g_{k+1}\|^2$ in the numerator of β_k.

Hybrid Conjugate-Gradient Methods

These algorithms have been devised to exploit the attractive features of the classical conjugate-gradient algorithms. They are defined by (2) and (3), where the parameter β_k is as in Table 2. There are two classes of hybrid algorithms. The first class of the hybrid algorithms combines in a *projective* manner the algorithms having $\|g_{k+1}\|^2$ in the numerator of β_k with the algorithms having $g_{k+1}^T y_k$ in the numerator of β_k. The second class of hybrid algorithms, more recently established, considers *convex combinations* of algorithms with $\|g_{k+1}\|^2$ in the numerator of β_k and the algorithms having $g_{k+1}^T y_k$ in the numerator of β_k. In general, the performances of hybrid conjugate-gradient algorithms are better than the performances of classical conjugate-gradient algorithms.

Scaled Conjugate-Gradient Algorithms

The algorithms in this class generate a sequence x_k of approximations to the minimum x^* of f, in which

$$x_{k+1} = x_k + \alpha_k d_k \ , \tag{6}$$

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k \ , \tag{7}$$

where θ_{k+1} is a parameter. The iterative process is initialized with an initial point x_0 and $d_0 = -g_0$. Observe that if $\theta_{k+1} = 1$, then we get the classical conjugate-gradient algorithms according to the value of the scalar parameter β_k. On the other hand, if $\beta_k = 0$, then we get another class of algorithms according to the selection of the parameter θ_{k+1}. Considering $\beta_k = 0$, there are two possibilities for θ_{k+1}: a positive scalar or a positive-definite matrix. If $\theta_{k+1} = 1$, then we have the steepest-descent algorithm. If $\theta_{k+1} = \nabla^2 f(x_{k+1})^{-1}$, or an approximation of it, then we get the Newton or the quasi-Newton algorithms, respectively. Therefore, we see that in the general case, when $\theta_{k+1} \neq 0$ is selected in a quasi-Newton manner, and $\beta_k \neq 0$, (7) represents a combination between the quasi-Newton and the conjugate-gradient methods. However, if θ_{k+1} is a matrix containing some useful information about the inverse Hessian of function f, we are better off using $d_{k+1} = -\theta_{k+1} g_{k+1}$ since the addition of the term $\beta_k s_k$ in (7) may prevent the direction d_k from being a descent direction unless the line search is sufficiently accurate. Therefore, in the following we shall consider θ_{k+1} as a positive scalar which contains some useful information on the inverse Hessian of function f.

To determine β_k consider the following procedure [1,2,3,4]. As we know, the Newton direction for solving (1) is given by $d_{k+1} = -\nabla^2 f(x_{k+1})^{-1} g_{k+1}$. Therefore, from the equality

$$-\nabla^2 f(x_{k+1})^{-1} g_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k \ ,$$

we get

$$\beta_k = \frac{s_k^T \nabla^2 f(x_{k+1}) \theta_{k+1} g_{k+1} - s_k^T g_{k+1}}{s_k^T \nabla^2 f(x_{k+1}) s_k} \ . \tag{8}$$

Using the Taylor development, after some algebra, we get

$$\beta_k = \frac{(\theta_{k+1} y_k - s_k)^T g_{k+1}}{y_k^T s_k} \ , \tag{9}$$

where $y_k = g_{k+1} - g_k$. Birgin and Martínez [10], who first introduced scaled conjugate-gradient algorithms, arrived at the same formula for β_k, but using a geometric interpretation of quadratic function minimization. The parameter β_k in (7) can be defined, as in Table 3, where the scaling parameter θ_k is computed as

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k} \ . \tag{10}$$

Another scaled conjugate-gradient algorithm was presented by Andrei [1,2,3,4]. This is a scaled memoryless Broyden–Fletcher–Goldfarb–Shanno (BFGS) preconditioned conjugate-gradient algorithm. The basic idea is to combine the scaled memoryless BFGS method and the preconditioning technique in the frame of the conjugate-gradient method. The preconditioner, which

Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Table 2
Hybrid conjugate-gradient algorithms

Nr.	Formula	Author(s)
1.	$\beta_k^{hDY} = max\{c\beta_k^{DY}, min\{\beta_k^{HS}, \beta_k^{DY}\}\}$	Hybrid Dai–Yuan [15] (hDY)
2.	$\beta_k^{hDYz} = max\{0, min\{\beta_k^{HS}, \beta_k^{DY}\}\}$	Hybrid Dai–Yuan zero [15] (hDYz)
3.	$\beta_k^{GN} = max\{-\beta_k^{FR}, min\{\beta_k^{PRP}, \beta_k^{FR}\}\}$	Gilbert and Nocedal [19] (GN)
4.	$\beta_k^{HuS} = max\{0, min\{\beta_k^{PRP}, \beta_k^{FR}\}\}$	Hu and Storey [24] (HuS)
5.	$\beta_k^{TaS} = \begin{cases} \beta_k^{PRP} & 0 \le \beta_k^{PRP} \le \beta_k^{FR}, \\ \beta_k^{FR} & \text{otherwise} \end{cases}$	Touati-Ahmed and Storey [37] (TaS)
6.	$\beta_k^{LS-CD} = max\{0, min\{\beta_k^{LS}, \beta_k^{CD}\}\}$	Hybrid Liu–Storey, conjugate descent (LS-CD)
7.	$\beta_k^{CCOMB} = (1-\theta_k)\frac{g_{k+1}^T y_k}{g_k^T g_k} + \theta_k \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$, $\theta_k = \frac{(y_k^T g_{k+1})(y_k^T s_k) - (y_k^T g_{k+1})(g_k^T g_k)}{(y_k^T g_{k+1})(y_k^T s_k) - \|g_{k+1}\|^2 \|g_k\|^2}$. If $\theta_k \le 0$, then set $\theta_k = 0$, i. e., $\beta_k^{CCOMB} = \beta_k^{PRP}$; if $\theta_k \ge 1$, then take $\theta_k = 1$, i. e., $\beta_k^{CCOMB} = \beta_k^{DY}$	Andrei [7]. Convex combination of PRP and DY where θ_k is obtained by a conjugacy condition (CCOMB)
8.	$\beta_k^{NDOMB} = (1-\theta_k)\frac{g_{k+1}^T y_k}{g_k^T g_k} + \theta_k \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$, $\theta_k = \frac{(y_k^T g_{k+1} - s_k^T g_{k+1})\|g_k\|^2 - (g_{k+1}^T y_k)(y_k^T s_k)}{\|g_{k+1}\|^2 \|g_k\|^2 - (g_{k+1}^T y_k)(y_k^T s_k)}$. If $\theta_k \le 0$, then set $\theta_k = 0$, i. e., $\beta_k^{NDOMB} = \beta_k^{PRP}$; if $\theta_k \ge 1$, then take $\theta_k = 1$, i. e., $\beta_k^{NDOMB} = \beta_k^{DY}$	Andrei [7]. Convex combination of PRP and DY where θ_k is obtained using the Newton direction (NDOMB)
9.	$\beta_k^{NDHSDY} = (1-\theta_k)\frac{g_{k+1}^T y_k}{y_k^T s_k} + \theta_k \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$, $\theta_k = -\frac{s_k^T g_{k+1}}{g_k^T g_{k+1}}$. If $\theta_k \le 0$, then set $\theta_k = 0$, i. e., $\beta_k^{NDHSDY} = \beta_k^{HS}$; if $\theta_k \ge 1$, then take $\theta_k = 1$, i. e., $\beta_k^{NDHSDY} = \beta_k^{DY}$	Andrei [8]. Convex combination of HS and DY, where θ_k is obtained using the Newton direction (NDHSDY)

Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Table 3
Scaled conjugate-gradient algorithms

No.	Formula	Author(s)
1.	$\beta_k^{BM} = \frac{g_{k+1}^T(\theta_k y_k - s_k)}{y_k^T s_k}$	Scaled Perry. Suggested by Birgin and Martínez [10] and Andrei [1,2,3,4] (BM)
2.	$\beta_k^{BM+} = max\left\{0, \frac{\theta_k g_{k+1}^T y_k}{y_k^T s_k}\right\} - \frac{g_{k+1}^T s_k}{y_k^T s_k}$	Scaled Perry+. Suggested by Birgin and Martínez [10] (BM+)
3.	$\beta_k^{sPRP} = \frac{\theta_k g_{k+1}^T y_k}{\alpha_k \theta_{k-1} g_k^T g_k}$	Scaled Polak-Ribière–Polyak. Suggested by Birgin and Martínez [10] and Andrei [1,2,3,4] (sPRP)
4.	$\beta_k^{sFR} = \frac{\theta_k g_{k+1}^T g_{k+1}}{\alpha_k \theta_{k-1} g_k^T g_k}$	Scaled Fletcher–Reeves. Suggested by Birgin and Martínez [10] and Andrei [1,2,3,4] (sFR)
5.	$\beta_k^{sHS} = \theta_{k+1} \frac{g_{k+1}^T y_k}{y_k^T s_k}$	Scaled Hestenes–Stiefel (sHS) [1]

Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Table 4
Modified conjugate-gradient algorithms

No.	Formula	Author(s)
1.	$d_{k+1} = -g_{k+1} + \bar{\beta}_k^N d_k, \quad d_0 = -g_0,$ $\bar{\beta}_k^N = max\{\beta_k^N, \eta_k\},$ $\eta_k = \frac{-1}{\|d_k\| min\{\eta, \|g_k\|\}}, \quad \eta = 0.01$ $\beta_k^N = \frac{1}{y_k^T d_k}\left(y_k - 2\frac{\|y_k\|^2}{y_k^T d_k} d_k\right)^T g_{k+1}$	Introduced by Hager and Zhang [20,21] (CG_DESCENT). This is a modification of the HS method
2.	$\beta_k^{ACGA} = \frac{1}{y_k^T s_k}\left(y_k - \frac{g_{k+1}^T y_k}{y_k^T s_k} s_k\right)^T g_{k+1}$	Suggested by Andrei [9] (ACGA)
3.	$\beta_k^{ACGA+} = \max\left\{0, \frac{y_k^T g_{k+1}}{y_k^T s_k}\right\}\left(1 - \frac{s_k^T g_{k+1}}{y_k^T s_k}\right)$	Suggested by Andrei [9] (ACGA+)
4.	$d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k^{CGSD} d_k, \quad d_0 = -g_0,$ $\beta_k^{CGSD} = \frac{1}{y_k^T s_k}\left(g_{k+1} - \frac{g_{k+1}^T y_k}{y_k^T s_k} s_k\right)^T g_{k+1}$ $\theta_{k+1} = \frac{\|g_{k+1}\|^2}{y_k^T g_{k+1}}$	Introduced by Andrei [5] (CGSD) as a modification of the DY method
5.	$\beta_k^{APRP} = \frac{1}{y_k^T s_k}\left(y_k - \frac{\|y_k\|^2}{\|g_k\|^2} s_k\right)^T g_{k+1}$	Suggested by Andrei [9] (APRP). This is a modification of the PRP method

is also a scaled memoryless BFGS matrix, is reset when the Powell restart criterion holds. The parameter scaling the gradient is selected as the spectral gradient (10).

SCALCG Algorithm

Step 1. Initialization. Select $x_0 \in R^n$, and the parameters $0 < \rho \leq \sigma < 1$. Compute $f(x_0)$ and $g_0 = \nabla f(x_0)$. Set $d_0 = -g_0$ and $\alpha_0 = 1/\|g_0\|$. Set $k = 0$.

Step 2. Line search. Compute α_k satisfying the Wolfe conditions (4) and (5). Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 3. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$.

Step 4. Scaling factor computation. Compute θ_k using (10).

Step 5. Restart direction. Compute the (restart) direction d_k as

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \theta_{k+1}\left(\frac{g_{k+1}^T s_k}{y_k^T s_k}\right) y_k - \left[\left(1 + \theta_{k+1}\frac{y_k^T y_k}{y_k^T s_k}\right)\frac{g_{k+1}^T s_k}{y_k^T s_k} - \theta_{k+1}\frac{g_{k+1}^T y_k}{y_k^T s_k}\right] s_k .$$

Step 6. Line search. Compute the initial guess $\alpha_k = \alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. Using this initialization compute α_k satisfying the Wolfe conditions. Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 7. Store: $\theta = \theta_k$, $s = s_k$ and $y = y_k$.

Step 8. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$.

Step 9. Restart. If the Powell restart criterion $\left|g_{k+1}^T g_k\right| \geq 0.2\|g_{k+1}\|^2$ is satisfied, then go to step 4 (a restart step), otherwise continue with step 10 (a standard step).

Step 10. Standard direction. Compute the direction d_k as

$$d_{k+1} = -v + \frac{(g_{k+1}^T s_k) w + (g_{k+1}^T w) s_k}{y_k^T s_k} - \left(1 + \frac{y_k^T w}{y_k^T s_k}\right)\frac{g_{k+1}^T s_k}{y_k^T s_k} s_k ,$$

where v and w are computed as

$$v = \theta g_{k+1} - \theta\left(\frac{g_{k+1}^T s}{y^T s}\right) y + \left[\left(1 + \theta\frac{y^T y}{y^T s}\right)\frac{g_{k+1}^T s}{y^T s} - \theta\frac{g_{k+1}^T y}{y^T s}\right] s ,$$

Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Table 5
Parametric conjugate-gradient algorithms

No.	Formula	Author(s)
1.	$\beta_k^{DL} = \frac{g_{k+1}^T(y_k - ts_k)}{y_k^T s_k}$, $t > 0$ is a constant	Dai and Liao [12] (DL)
2.	$\beta_k^{DL+} = max\left\{0, \frac{y_k^T g_{k+1}}{y_k^T s_k}\right\} - t\frac{s_k^T g_{k+1}}{y_k^T s_k}$, $t > 0$ is a constant	Dai and Liao + [12] (DL+)
3.	$\beta_k^{YT} = \frac{g_{k+1}^T(z_k - ts_k)}{d_k^T z_k}$, where $z_k = y_k + \frac{\delta\xi_k}{s_k^T u_k}u_k$, $\xi_k = 6(f_k - f_{k+1}) + 3(g_k + g_{k+1})^T s_k$, $\delta \geq 0$ is a constant and $u_k \in R^n$ satisfies $s_k^T u_k \neq 0$; for example $u_k = d_k$	Suggested by Yabe and Takano [38] (YT) based on a modified secant condition given by Zhang et al. [39]
4.	$\beta_k^{YT+} = \max\left\{0, \frac{g_{k+1}^T z_k}{d_k^T z_k}\right\} - t\frac{g_{k+1}^T s_k}{d_k^T z_k}$	Suggested by Yabe and Takano plus [38] (YT+)
5.	$\beta_k = \frac{\|g_{k+1}\|^2}{\lambda_k\|g_k\|^2 + (1-\lambda_k)d_k^T y_k}$, $\lambda_k \in [0, 1]$. The FR algorithm corresponds to $\lambda_k = 1$. The DY algorithm corresponds to $\lambda_k = 0$	Suggested by Dai and Yuan [14]
6.	$\beta_k = \frac{\mu_k\|g_{k+1}\|^2 + (1-\mu_k)g_{k+1}^T y_k}{\lambda_k\|g_k\|^2 + (1-\lambda_k)d_k^T y_k}$, $\lambda_k, \mu_k \in [0, 1]$	Suggested by Nazareth [28]. This two-parameter family includes the methods FR, DY, PRP and HS in extreme cases
7.	$\beta_k = \frac{\mu_k\|g_{k+1}\|^2 + (1-\mu_k)g_{k+1}^T y_k}{(1-\lambda_k-\omega_k)\|g_k\|^2 + \lambda_k d_k^T y_k - \omega_k d_k^T g_k}$, $\lambda_k, \mu_k \in [0, 1]$ and $\omega_k \in [0, 1-\lambda_k]$	Suggested by Dai and Yuan [14] This three-parameter family includes the six classical conjugate-gradient algorithms, as well as the previous one-parameter and two-parameter families

and

$$w = \theta y_k - \theta\left(\frac{y_k^T s}{y^T s}\right) y + \left[\left(1 + \theta\frac{y^T y}{y^T s}\right)\frac{y_k^T s}{y^T s} - \theta\frac{y_k^T y}{y^T s}\right] s,$$

with saved values θ, s and y.

Step 11. Line search. Compute the initial guess $\alpha_k = \alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. Using this initialization compute α_k satisfying the Wolfe conditions. Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 12. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$ and go to step 9.

To a great extent, the SCALCG algorithm [1,2,3,4] is very close to the Perry–Shanno computational scheme [30,34,35]. SCALCG is a scaled memoryless BFGS preconditioned algorithm where the scaling factor is the inverse of a scalar approximation of the Hessian. If the Powell restart criterion $\left|g_{k+1}^T g_k\right| \geq 0.2\|g_{k+1}\|^2$ is used, for general functions f bounded from below with bounded second partial derivatives and bounded level set, using the same arguments considered by Shanno in [35], one can prove that either the iterates converge to a point x^* satisfying $\|g(x^*)\| = 0$ or the iterates cycle.

Modified Conjugate-Gradient Algorithms

We know a large variety of modified conjugate-gradient algorithms. All of them are designed to improve the

performances of the classical computational schemes using the idea of preconditioning or the modification of classical schemes in order to satisfy the sufficient-descent condition. The algorithms in this class are characterized by (2) and (3), where the parameter β_k is computed as in Table 4.

Maximization in the formula for the $\bar{\beta}_k^N$ computation scheme by Hager and Zhang plays the role of the truncation operation like in the PRP+ scheme, for example. Hager and Zhang obtained this algorithm by deleting a term from the search direction for the memoryless quasi-Newton scheme of Perry [30] and Shanno [34]. Hager and Zhang [20] proved the global convergence with inexact line search, showing that for any line search and any function, the sufficient-descent condition $g_k^T d_k \leq -(7/8)\,\|g_k\|^2$ is satisfied and the jamming is avoided essentially owing to the $y_k^T g_{k+1}$ term in the formula for β_k^N.

The ACGA and ACGA+ computational schemes are a modification of the DY conjugate-gradient algorithm, designed to satisfy the sufficient-descent condition. Andrei [9] proved that for uniformly convex functions under a strong Wolfe condition the ACGA is globally convergent. The CGSD algorithm is also a modification of the Dai and Yuan conjugate-gradient algorithm. Andrei [9] proved the global convergence of CGSD for general nonlinear functions under the Wolfe conditions.

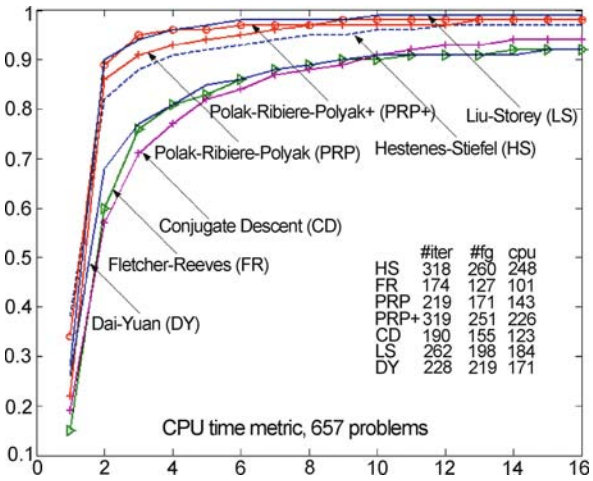
One of the best conjugate-gradient algorithms in this class is CONMIN by Shanno [34] and Shanno and Phua [36]. Using the Hestenes and Stiefel formula for updating β_k, Perry [30] suggested a formula for computing the search direction d_{k+1} which satisfies a system of linear equations, similar but not identical to the quasi-Newton equation. Shanno [34] reconsidered the method of Perry and interpreted it as a memoryless BFGS updating formula. In this algorithm g_{k+1} is modified by a positive-definite matrix which best estimates the inverse Hessian, without any additional storage requirements. For convex functions, under inexact line search, Shanno [35] proved the global convergence of CONMIN.

Parametric Conjugate-Gradient Algorithms

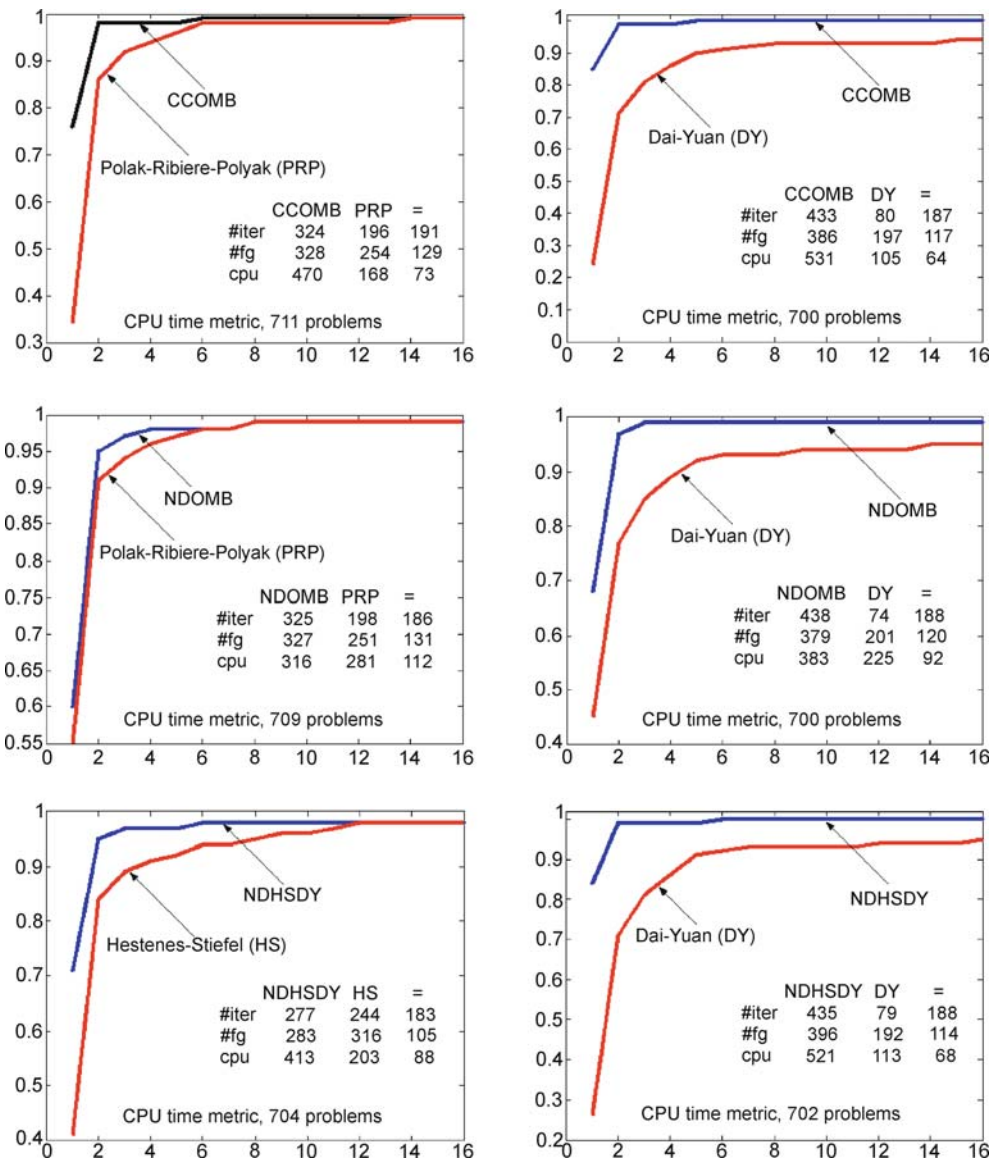
The parametric conjugate-gradient algorithms were introduced in the same way as the quasi-Newton methods were combined to get the Broyden or the Huang families. These algorithms are defined by (2) and (3), where the parameter β_k is as in Table 5.

Performance Profiles

In this section we present the computational performance of a Fortran implementation of conjugate-gradient algorithms on a set of 750 unconstrained optimization test problems. The test problems are the unconstrained problems in the CUTE [11] library, along with other large-scale optimization problems presented in [6]. We selected 75 large-scale unconstrained optimization problems in extended or generalized form. For each function we have considered ten numerical experiments with the number of variables $n = 1000, 2000, \ldots, 10000$. CG_DESCENT was authored by Hager and Zhang [20,21], CONMIN by Shanno and Phua [36]. The CG_DESCENT code contains the variant CG_DESCENT(w) implementing the Wolfe line search and the variant CG_DESCENT(aw) implementing an approximate Wolfe line search. The Wolfe conditions implemented in CG_DESCENT(w) can compute a solution with an accuracy on the order of the square root of machine epsilon. In contrast, the approximate Wolfe line search implemented in CG_DESCENT(aw) can compute a solution with



Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 1
Performance profiles of the HS, FR, PRP, PRP+, CD, LS and DY methods. See the tables for the definitions of the acronyms used for the algorithms referred to in all the figures



Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 2
Performance profiles of some hybrid conjugate-gradient algorithms (continued on next page)

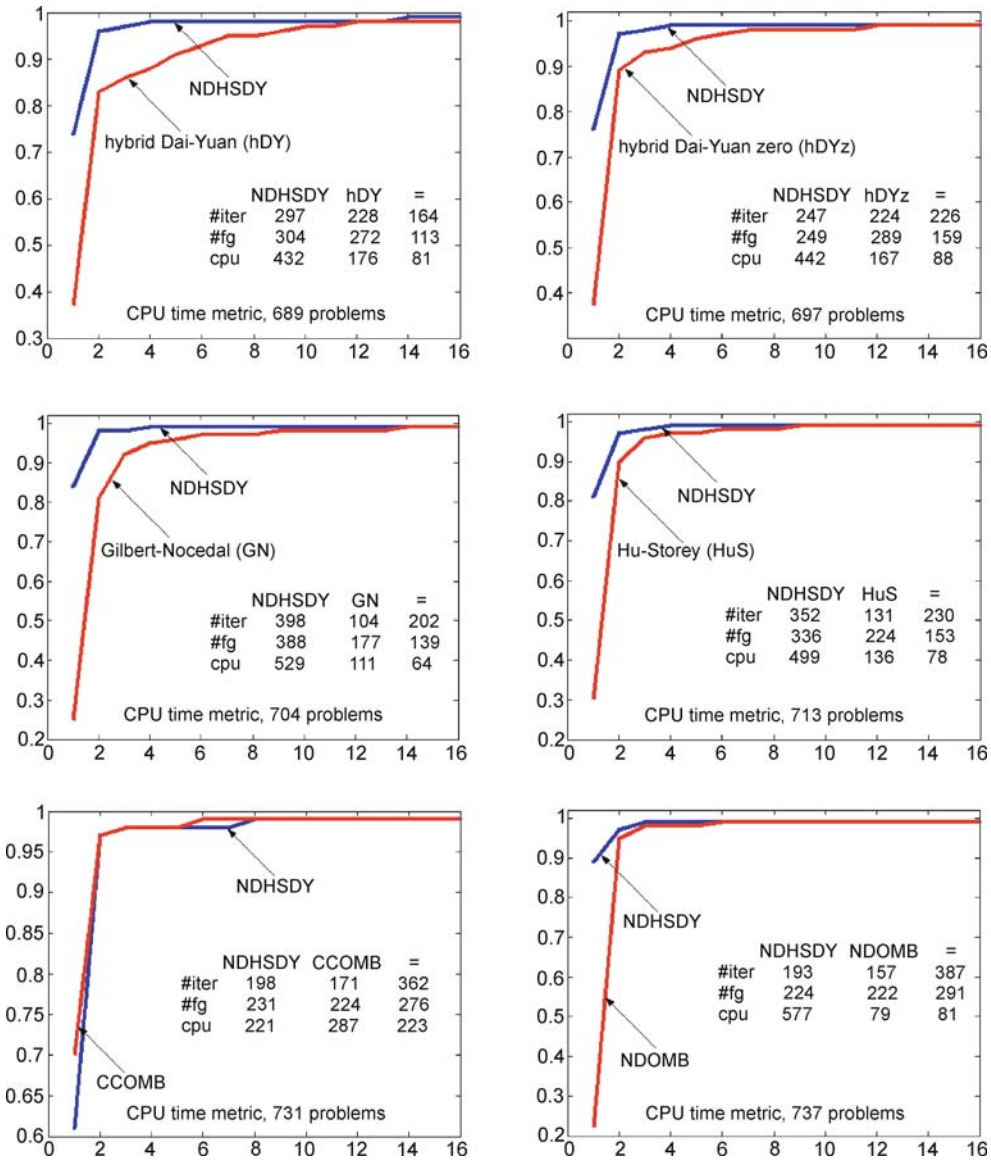
an accuracy on the order of machine epsilon. The rest of the algorithms considered in this study are authored by Andrei. All codes were written in double-precision Fortran and compiled with f77 (default compiler settings) on an Intel Pentium 4, 1.8 GHz workstation.

All algorithms implement the Wolfe line search conditions with $\rho = 0.0001$ and$\sigma = 0.9$, and the same stopping criterion $\|g_k\|_\infty \leq 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector.

The comparisons of the algorithms are given in the following context. Let f_i^{ALG1} and f_i^{ALG2} be the optimal value found by ALG1 and ALG2, for problem $i = 1, \ldots, 750$, respectively. We say that, in the particular problem i, the performance of ALG1 was better than the performance of ALG2 if

$$\left| f_i^{\text{ALG1}} - f_i^{\text{ALG2}} \right| < 10^{-3}$$

and the number of iterations, or the number of function-gradient evaluations, or the CPU time of ALG1

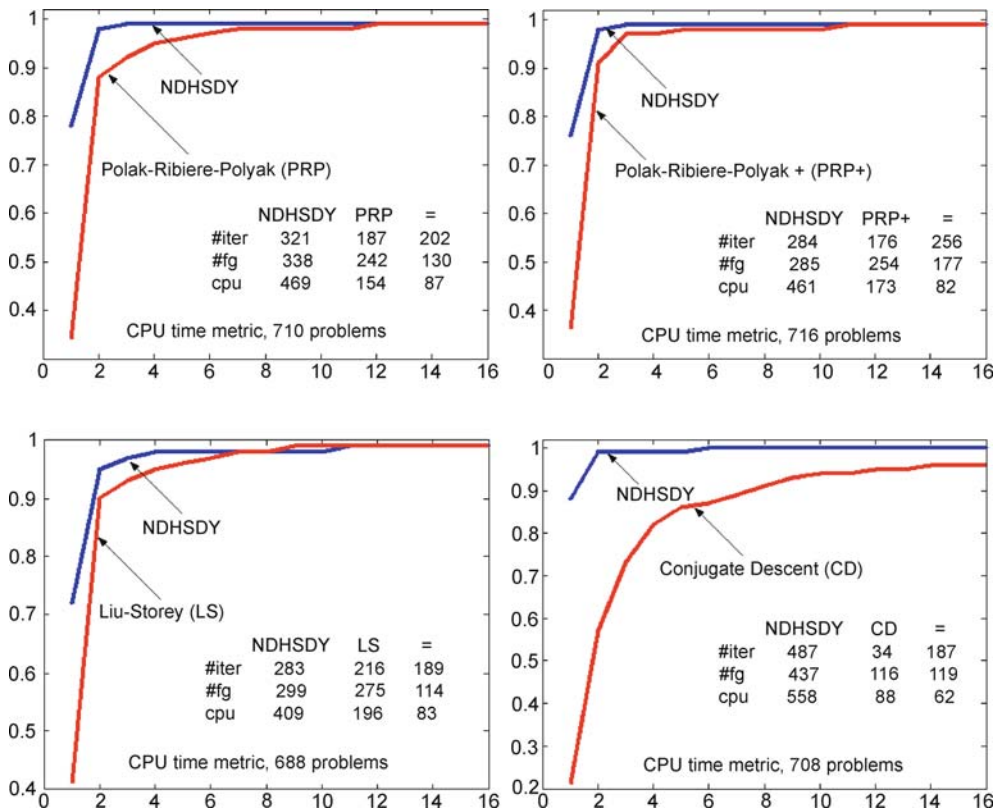


Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 2 (continued)

was less than the number of iterations, or the number of function-gradient evaluations, or the CPU time corresponding to ALG2, respectively.

The performances of these algorithms were evaluated using the profiles of Dolan and Moré [16] corresponding to this set of 750 test problems we extracted from the CUTE collection [11] and from [6]. For each algorithm, we plot the fraction of problems for which the algorithm is within a factor of the best CPU time. The left side of the figures gives the percentage of the test problems, out of 750, for which an algorithm is more performant; the right side gives the percentage of the test problems that were successfully solved by each of the algorithms. Mainly, the right side represents a measure of an algorithm's robustness.

In the first set of numerical experiments we compare the classical conjugate-gradient algorithms. Figure 1 shows the CPU time performance profiles of these algorithms.



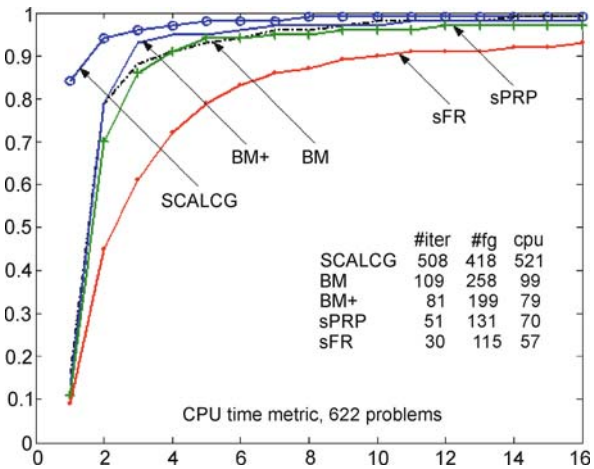
Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 3
Performance profiles of the NDHSDY algorithm compared with those of some classical conjugate-gradient algorithms

From Fig. 1 we see that the FR, CD and DY methods, although they have strong convergence properties, may not perform well in practice owing to jamming. In contrast, although the HS, PRP and LS methods in general may not converge, they often perform better than the FR, CD and DY methods.

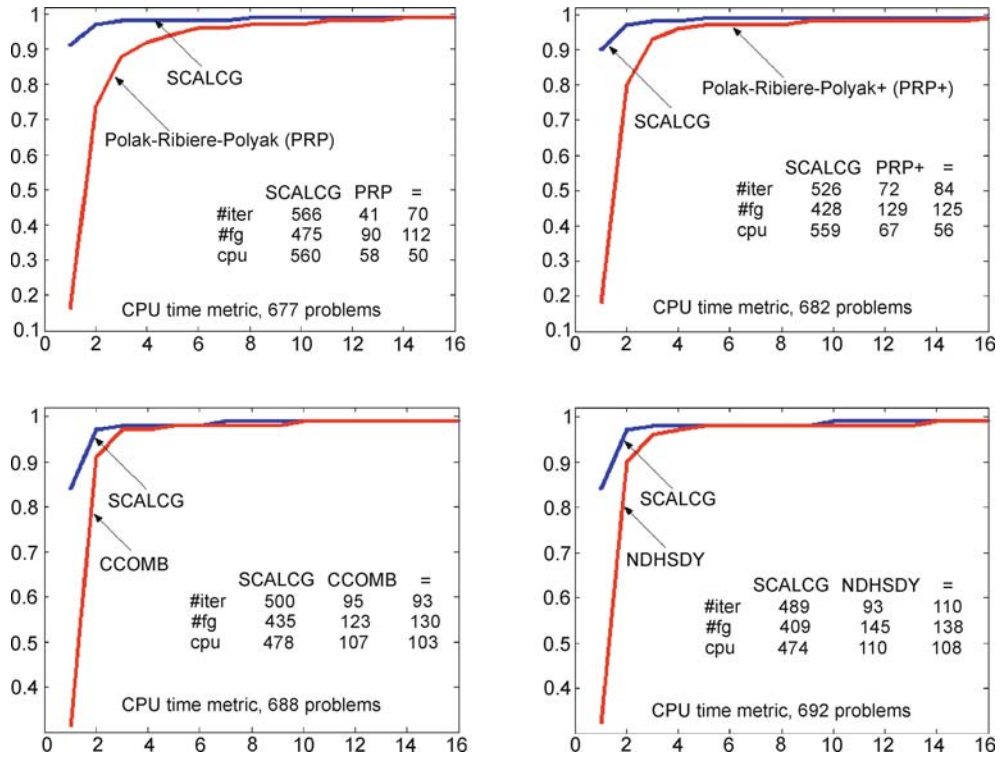
Figure 2 presents the performance profiles of some hybrid conjugate-gradient algorithms.

Figure 3 presents the performance profiles of the NDHSDY algorithm compared with those of the classical conjugate-gradient algorithms: PRP, PRP+, LS and CD. It seems that the best algorithm is the hybrid algorithm NDHSDY given by a convex combination of HS and DY, where the parameter in the convex combination is obtained using the Newton direction.

In the next set of numerical experiments we compare the scaled conjugate-gradient algorithms. Figure 4 shows the performance profiles of SCALCG, BM, BM+, sPRP and sFR. We see that the SCALCG algorithm is top performer among the scaled conjugate-gradient algorithms.



Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 4
Performance profiles of scaled conjugate-gradient algorithms



Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 5
Performance profiles of the SCALCG algorithm compared with those of the PRP, PRP+, CCOMB and NDHSDY algorithms

Figure 5 shows the performance profile of the SCALCG algorithm compared with those of the classical conjugate-gradient algorithms PRP and PRP+, as well as the hybrid algorithms CCOMB and NDHSDY.

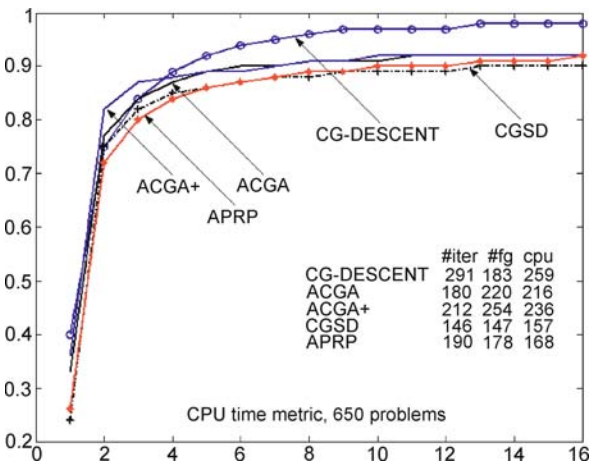
In the following we compare the modified conjugate-gradient algorithms CG_DESCENT(w), ACGA, ACGA+, CGSD and APRP. Figure 6 presents the performance profiles of these algorithms.

Figure 7 presents the performance profiles of the CG_DESCENT(w) algorithm compared with those of the PRP, PRP+, NDHSDY and SCALCG algorithms.

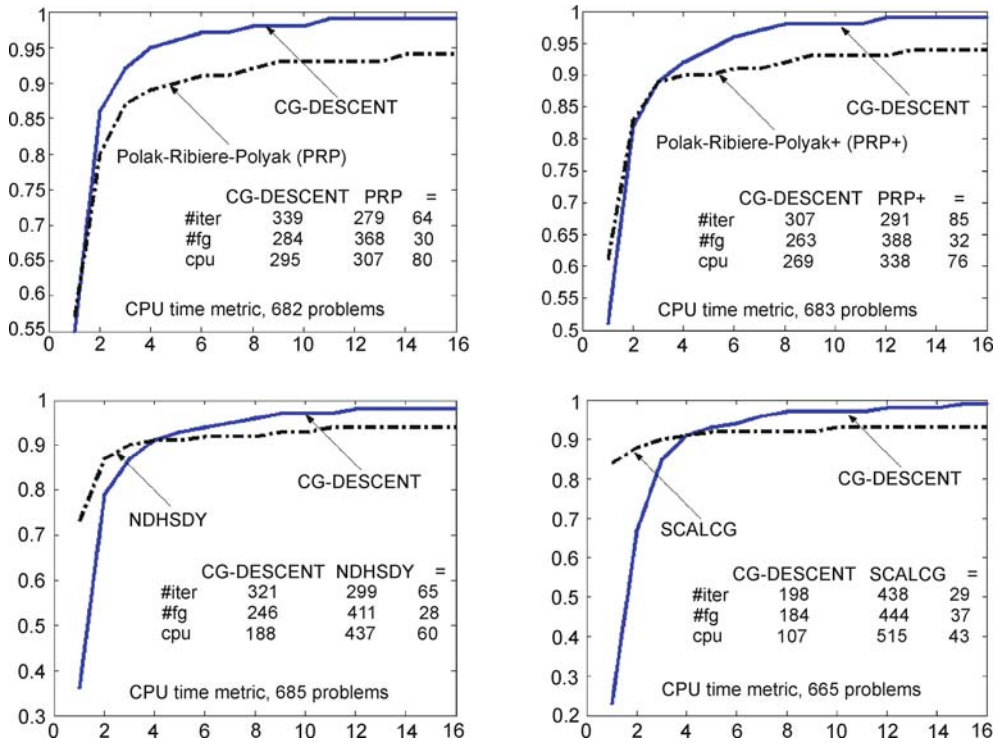
Now, comparing CONMIN with some other modified conjugate-gradient algorithms, ACGA, ACGA+, CGSD and APRP, we obtained the performance profiles as in Fig. 8.

We see that CONMIN is the top performer. Figure 9 presents the performances profiles of the CONMIN algorithm compared with those of the PRP, NDHSDY, SCALCG and CG_DESCENT algorithms.

Finally, let us consider the parametric conjugate-gradient algorithms DL(t=1) and DL+(t=1). Figure 10 shows the performance profiles of the DL and DL+ algorithms compared with those of the PRP, SCALCG and CONMIN algorithms.



Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 6
Performance profiles of the CG_DESCENT, ACGA, ACGA+, CGSD and APRP algorithms

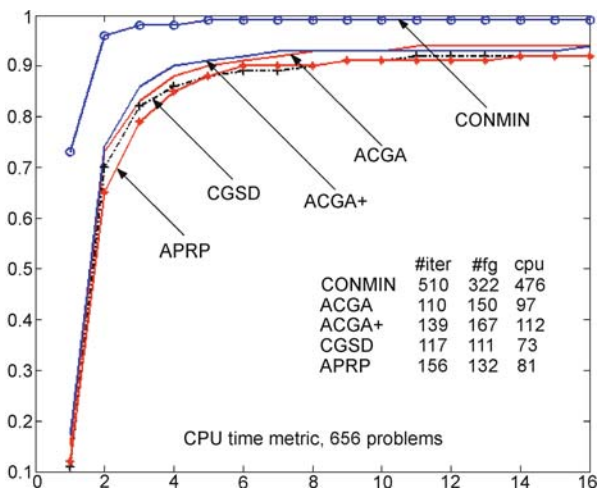


Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 7
Performance profiles of the CG_DESCENT(w) algorithm compared with those of the PRP, PRP+, NDHSDY and SCALCG algorithms

Conclusion and Discussion

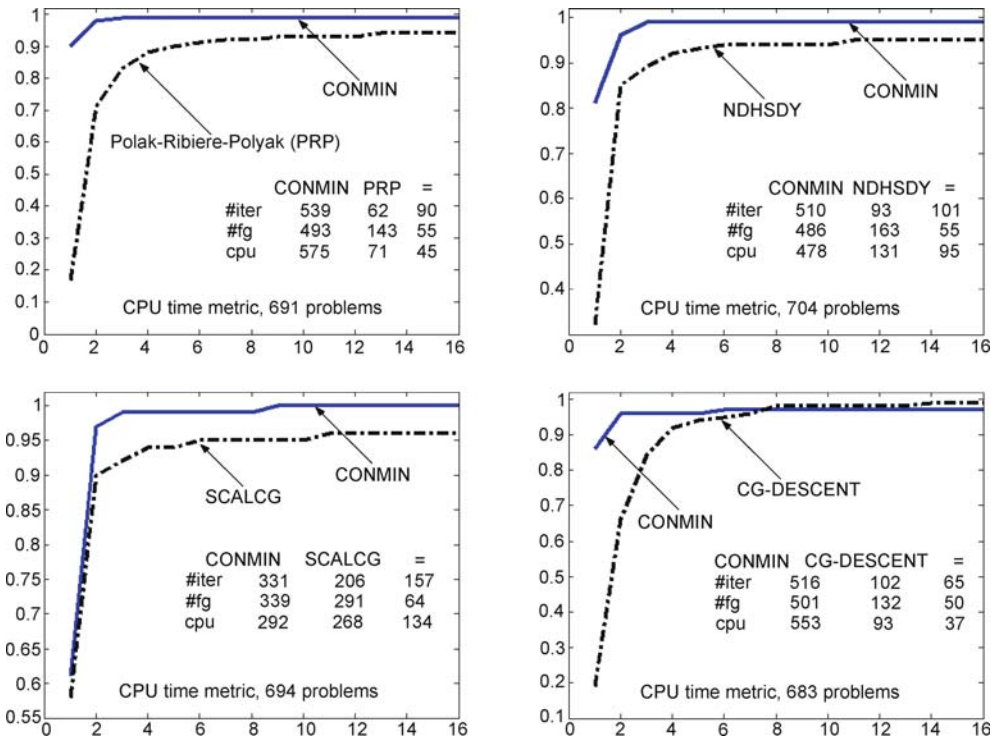
Conjugate-gradient algorithms are one of the most elegant and probably the simplest algorithms for computational nonlinear optimization. Their theory is well established [22] and they have proved to be surprisingly effective in solving real practical applications. The computational study presented here, which include 29 conjugate-gradient algorithms, shows that the most effective are CONMIN, CG_DESCENT and SCALCG. Close to these algorithms is NDHSDY, a convex combination of HS and DY conjugate-gradient algorithms in which the parameter is computed using the Newton direction. Concerning the robustness, CG_DESCENT is in first place.

This computational study involved a large variety of nonlinear test functions. However, to draw conclusions about the effectiveness of these algorithms, the test functions must be organized on some classes with well-established characteristics, and one must see which conjugate-gradient algorithm is more successful. This remains to be explored.



Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 8
Performance profiles of the CONMIN, ACGA, ACGA+, CGSD and APRP algorithms

It is worth seeing a comparison between the most successful conjugate-gradient algorithms and the quasi-



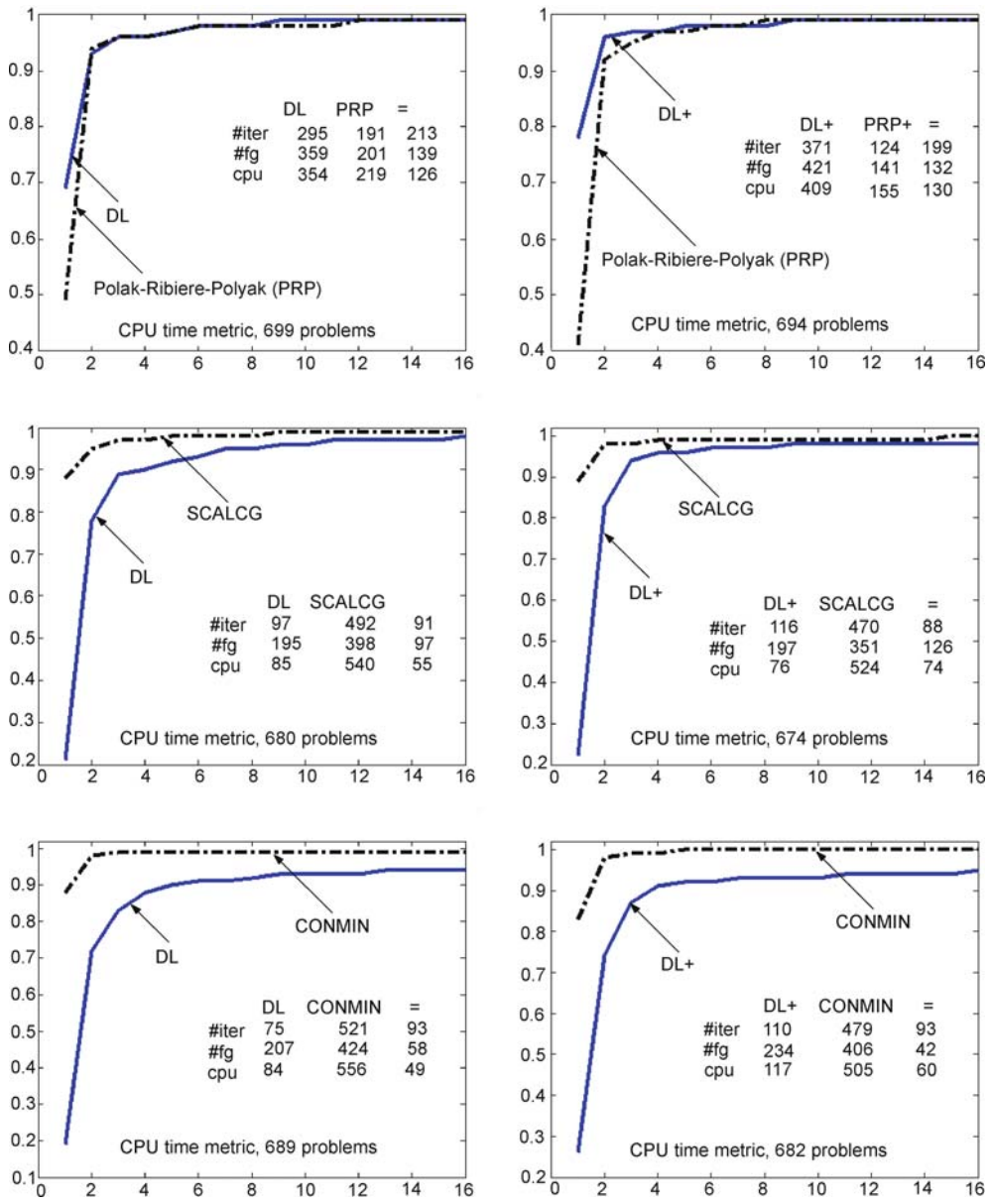
Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 9
Performance profiles of the CONMIN algorithm compared with those of the PRP, NDHSDY, SCALCG and CG_DESCENT algorithms

Newton limited BFGS (LBFGS) algorithm of Nocedal [29]. Figure 11 shows the performance profiles of the CONMIN, SCALCG, CG_DESCENT and NDHSDY algorithms compared with those of the LBFGS (m = 3) algorithm, an implementation given by Liu and Nocedal [25] using the line search of Moré and Thuente [27].

From Fig. 11 we see that LBFGS (m = 3) is way more successful than any conjugate-gradient algorithm. Closest to LBFGS is CONMIN.

Even though conjugate-gradient methods are relevant nonlinear optimization methods, there are some open problems which deserve additional research:

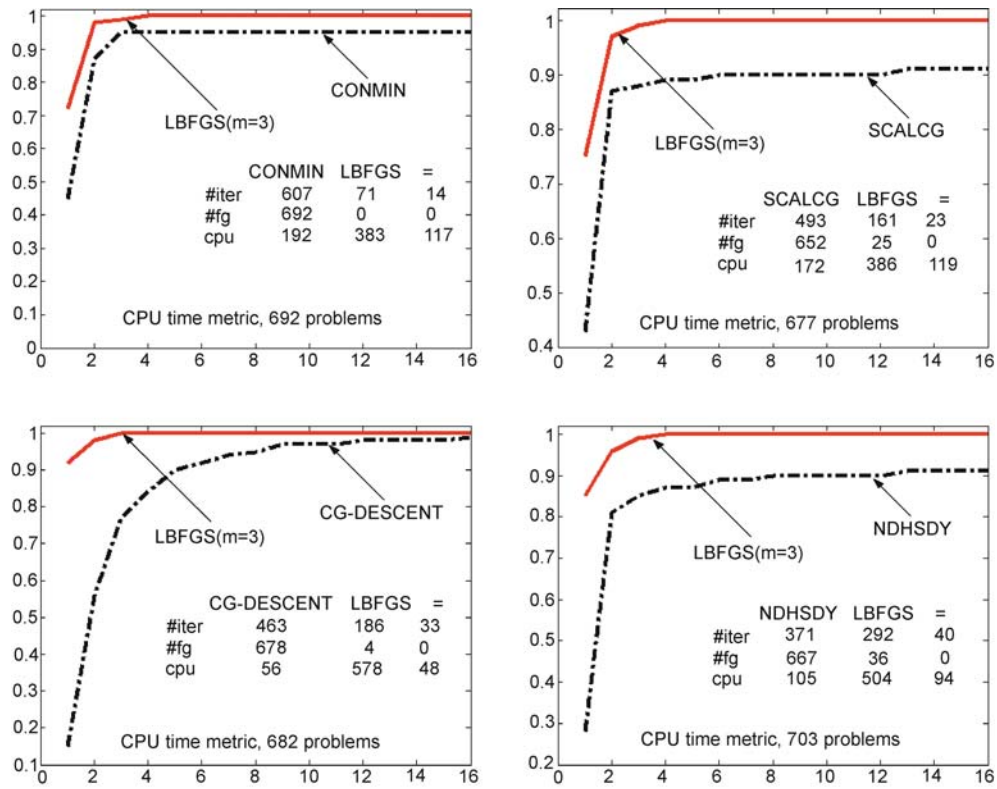
1. In contrast to the quasi-Newton methods for which the step length for the vast majority of iterations is equal to 1, the step length in conjugate gradient methods differs from 1, being larger or smaller up by to two order of magnitude depending on how the problem is scaled. In conjugate-gradient methods the size of α_k varies in a very unpredictabe way.
2. Another open problem is the preconditioning of conjugate-gradient algorithms. The scaled conjugate-gradient algorithms by Birgin and Martínez [10] and Andrei [1,2,3,4] introduce a scaling of g_{k+1} in the direction d_{k+1} computation. However, if the definition of θ_{k+1} in (7) does contain enough information about the inverse Hessian of the minimizing function, then it is better to use the search direction $d_{k+1} = -\theta_{k+1} g_{k+1}$, since the addition of the term $\beta_k s_k$ in (7) may prevent d_{k+1} from being a descent direction unless the line search is sufficiently accurate. In scaled conjugate-gradient algorithms there is a very delicate balance between $-\theta_{k+1} g_{k+1}$ and $\beta_k s_k$, which brings to attention the preconditioning question.
3. Another open problem with conjugate-gradient methods is that the structure of the minimizing problem is not taken into account to design more efficient computational schemes. This is in sharp contrast to quasi-Newton or truncated Newton methods.



Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 10
Performance profiles of the DL(t=1) and DL+(t=1) algorithms compared with those of the PRP, SCALCG and CONMIN algorithms

References

1. Andrei N (2007) Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization. Optim Methods Softw 22:561–571
2. Andrei N (2007) A scaled BFGS preconditioned conjugate gradient algorithm for unconstrained optimization. Appl Math Lett 20:645–650
3. Andrei N, Scaled conjugate gradient algorithms for unconstrained optimization. Comput Optim Appl 38:401–416
4. Andrei N (2007) A scaled nonlinear conjugate gradient algorithm for unconstrained optimization. http://www.informaworld.com/smpp/content%7Econtent=a779773265%7Edb=all%7Eorder=pubdate: First published on: 21 June 2007
5. Andrei N (2008) A Dai-Yuan conjugate gradient algorithm with sufficient descent and conjugacy conditions for unconstrained optimization. Appl Math Lett 21:165–171



Performance Profiles of Conjugate-Gradient Algorithms for Unconstrained Optimization, Figure 11
Performance profiles of the LBFGS(m = 3) algorithm compared with those of the CONMIN, SCALCG, CG_DESCENT and NDHSDY conjugate-gradient algorithms

6. Andrei N, Test functions for unconstrained optimization. http://www.ici.ro/camo/neculai/SCALCG/evalfg.for
7. Andrei N (2007) New hybrid conjugate gradient algorithms for unconstrained optimization. ICI Thechnical Report, September 26
8. Andrei N (2007) Another hybrid conjugate gradient algorithm for unconstrained optimization. ICI Technical Report, July 18
9. Andrei N (2007) Another nonlinear conjugate gradient algorithm for unconstrained optimization. ICI Thechnical Report, May 16
10. Birgin E, Martínez M (2001) A spectral conjugate gradient method for unconstrained optimization. Appl Math Optim 43:117–128
11. Bongartz I, Conn AR, Gould NIM, Toint PL (1995) CUTE: constrained and unconstrained testing environments. ACM Trans Math Softw 21:123–160
12. Dai YH, Liao LZ (2001), New conjugacy conditions and related nonlinear conjugate gradient methods. Appl Math Optim 43:87–101
13. Dai YH, Yuan Y (1999) A nonlinear conjugate gradient method with a strong global convergence property. SIAM J Optim 10:177–182
14. Dai YH, Yuan Y (2001) A three-parameter family of hybrid conjugate gradient method. Math Comput 70:1155–1167
15. Dai YH, Yuan Y (2003) A class of globally convergent conjugate gradient methods. Sci China Ser A 46:251–261
16. Dolan ED, Moré JJ (2002) Benchmarcking optimization software with performance profiles. Math Program 91:201–213
17. Fletcher R (1987) Practical Methods of Optimization, 2nd edn. Wiley, Chichester
18. Fletcher R, Reeves C (1964) Function minimization by conjugate gradients. Comput J 7:149–154
19. Gilbert JC, Nocedal J (1992) Global convergence properties of conjugate gradient methods. SIAM J Optim 2:21–42
20. Hager WW, Zhang H (2005) A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J Optim 16:170–192
21. Hager WW, Zhang H (2006) Algorithm 851: CG_DESCENT, A conjugate gradient method with guaranteed descent. ACM Trans Math Softw 32:113–137
22. Hager WW, Zhang H (2006), A survey of nonlinear conjugate gradient methods. Pac J Optim 2:35–58

23. Hestenes MR, Stiefel E (1952) Methods of conjugate gradients for solving linear systems. J Res Nat Bur Stand Sec B 48:409–436
24. Hu YF, Storey C (1991) Global convergence result for conjugate gradient methods. J Optim Theory Appl 71:399–405
25. Liu D, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Math Program B 45:503–528
26. Liu DC, Storey (1991) Efficient generalized conjugate gradient algorithms. Part 1: Theory. J Optim Theory Appl 69:129–137
27. Moré JJ, Thuente DJ (1994) Line search algorithms with guaranteed sufficient decrease. ACM Trans Math Softw 20:286–307
28. Nazareth JL (1999) Conjugate gradient methods. In: Floudas C, Pardalos P (eds) Encyclopedia of Optimization. Kluwer, Boston
29. Nocedal J (1980) Updating quasi-Newton matrices with limited starage. Math Comput 35:773–782
30. Perry A (1978) A modified conjugate gradient algorithm. Oper Res 26:1073–1078
31. Polak E, Ribière G (1969) Note sur la convergence de directions conjugées. Rev Francaise Informat Recherche Operationelle 3e Année 16:35–43
32. Polyak BT (1969) The conjugate gradient method in extreme problems. URSS Comp Math Math Phys 9:94–112
33. Powell MJD (1984) Nonconvex minimization calculations and the conjugate gradient method. Numerical Analysis (Dundee, (1983)), Lecture Notes in Mathematics, vol 1066. Springer, Berlin, pp 122–141
34. Shanno DF (1978) Conjugate gradient methods with inexact searches. Math Oper Res 3:244–256
35. Shanno DF (1978) On the convergence of a new conjugate gradient algorithm. SIAM J Numer Anal 15:1247–1257
36. Shanno DF, Phua KH (1976) Algorithm 500, Minimization of unconstrained multivariate functions. ACM Trans Math Softw 2:87–94
37. Touati-Ahmed D, Storey C (1990) Efficient hybrid conjugate gradient techniques. J Optim Theory Appl 64:379–397
38. Yabe H, Takano M (2004) Global convergence properties of nonlinear conjugate gradient methods with modified secant conditions. Comput Optim Appl (COAP) 28:203–225
39. Zhang JZ, Deng NY, Chen LH (1999) New quasi-Newton equation and related methods for unconstrained optimization. J Optim Theory Appl 102:147–167

Phase Problem in X-ray Crystallography: Shake and Bake Approach

HERBERT A. HAUPTMAN
Hauptman–Woodward Medical Research Institute Inc., Buffalo, USA

MSC2000: 90C26

Article Outline

Keywords
Introduction
Identities Among the Phases
Structure Invariants
Probabilistic Background
For Fixed H and K, the Conditional Probability Distribution of the Triplet Φ_{HK}
Minimal Principle
Computer Program 'Shake and Bake'
Applications
A Notable Application: Determination by Shake and Bake of the Previously Known Crystal Structure of Toxin II From the Scorpion
See also
References

Keywords

Global optimization; Phase problem; Normalized structure factors; Structure invariants; Minimal function; Minimal principle; Shake and bake algorithm

The *phase problem* of *X*-ray crystallography is formulated as one in constrained global minimization. The problem of reaching this minimum is solved by the *Shake and Bake algorithm* which effectively by-passes the myriad of (unconstrained) local minima. The function whose constrained global minimum is sought is known as the *minimal function*.

Introduction

When a beam of monochromatic *X*-rays strikes a crystal, the crystal scatters the incident beam in different directions and with different intensities determined by the crystal structure, that is the arrangement of the atoms in the unit cell of the crystal. From the intensities of the scattered *X*-rays, the directions of which are labeled by the so-called reciprocal lattice vectors H, a set of numbers $|E_{\mathbf{H}}|$ may be derived, one corresponding to each scattered *X*-ray. The phases $\phi_{\mathbf{H}}$ of the scattered *X*-rays are lost in the scattering (diffraction) experiment, that is to say cannot be measured. However, the elucidation of the crystal structure requires a knowledge of

the complex numbers

$$E_{\mathbf{H}} = |E_{\mathbf{H}}| \exp(i\phi_{\mathbf{H}}) , \tag{1}$$

the so-called *normalized structure factors*, of which only the magnitudes $|E_{\mathbf{H}}|$ can be determined from experiment. Thus the phase $\phi_{\mathbf{H}}$, unobtainable from the diffraction experiment, must be assigned to each $|E_{\mathbf{H}}|$. The problem of determining the phases $\phi_{\mathbf{H}}$ when only the magnitudes $|E_{\mathbf{H}}|$ are available, is called the 'phase problem'. A major advance in the early 1950s was the recognition that the lost phase information was to be found in the measurable intensities of the diffraction pattern. In fact, owing to the known atomicity of crystal structures and the redundancy of observed magnitudes $|E_{\mathbf{H}}|$, the phase problem is not only solvable in principle but is a greatly overdetermined one.

The phase problem is here formulated as one in constrained global minimization. The ability to impose constraints, in the form of identities among the phases which must, of necessity, be satisfied, even if only incompletely and approximately, enables one to avoid the countless local minima of the minimal function and to arrive at the unique constrained global minimum. The shake and bake algorithm, which implements the minimal principle formulated here, provides a routine and completely automatic solution of the phase problem when diffraction intensities to atomic resolution (1.2Å or better) are available.

Identities Among the Phases

The relationship between the crystal structure and the diffraction pattern, in the case that the structure consists of N identical atoms in the unit cell (the only case to be considered here), is given by the pair of equations

$$E_{\mathbf{H}} = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} \exp(2\pi i \mathbf{H} \cdot r_j) , \tag{2}$$

$$\langle E_{\mathbf{H}} \exp(-2\pi i \mathbf{H} \cdot r) \rangle_{\mathbf{H}} = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } r = r_j, \\ 0 & \text{if } r \neq r_j, \end{cases} \tag{3}$$

where $|E_{\mathbf{H}}|$ is obtained directly from the intensity of the X-ray scattered in the direction labeled by the reciprocal lattice vector H, whose three components are integers, r is an arbitrary three-dimensional vector, and r_j is the position vector of the atom labeled j. Thus the positions of the maxima of the triple Fourier series (3), a function of the position vector r, yield the crystal structure directly. However, in order to calculate (3) both the phases $\phi_{\mathbf{H}}$ and magnitudes $|E_{\mathbf{H}}|$ (in (1)) are needed. Since only the magnitudes $|E_{\mathbf{H}}|$ are obtainable from the measured intensities in the diffraction experiment while the phases $\phi_{\mathbf{H}}$ are lost, (3) makes clear why the phase problem has historically been regarded as the central problem of X-ray crystallography.

The system of equations (2) implies the existence of relationships among the normalized structure factors $E_{\mathbf{H}}$ since the (relatively few) unknown atomic position vectors r_j may, at least in principle, be eliminated. In this way one obtains a system of equations among the normalized structure factors $E_{\mathbf{H}}$ alone, that is, among the phases $\phi_{\mathbf{H}}$ and magnitudes $|E_{\mathbf{H}}|$:

$$F(E) = G(\phi; |E|) = 0 \tag{4}$$

which depends on N but is independent of the atomic position vectors r_j. For a specified crystal structure the magnitudes $|E|$ are determined (by (2)). Thus the system of equations (4) leads directly to a system of identities among the phases ϕ alone:

$$G(\phi; |E|) \rightarrow H(\phi || E|) = 0 \tag{5}$$

dependent now on the presumed known magnitudes $|E|$, which must of necessity be satisfied. The direct methods are those which exploit these relationships to go directly from known magnitudes $|E_{\mathbf{H}}|$ to the desired phases $\phi_{\mathbf{H}}$

Structure Invariants

Equation (3) implies that the normalized structure factors $E_{\mathbf{H}}$, that is to say magnitudes $|E_{\mathbf{H}}|$ and phases $\phi_{\mathbf{H}}$, determine the crystal structure. Equation (2) however does not imply that, conversely, the crystal structure determines unique values for the normalized structure factors $E_{\mathbf{H}}$ since the atomic position vectors r_j depend not only on the crystal structure but on the choice of origin as well. As it turns out, however, the magnitudes $|E_{\mathbf{H}}|$ are in fact uniquely determined by the crystal structure independently of the choice of origin, but the values of the individual phases depend on the choice of origin as well as on the crystal structure. Nevertheless there exist special linear combinations of the phases, the so-called *structure invariants*, whose values

are uniquely determined by the crystal structure and are independent of the choice of origin. The most important class of structure invariants, the so-called triplets, are the linear combinations of three phases

$$\phi_{\mathbf{HK}} = \phi_{\mathbf{H}} + \phi_{\mathbf{K}} + \phi_{-\mathbf{H}-\mathbf{K}} \tag{6}$$

where **H** and **K** are arbitrary reciprocal lattice vectors.

Probabilistic Background

It is assumed that the atomic position vectors r_j are random variables which are uniformly and independently distributed. Under this assumption the normalized structure factors $E_{\mathbf{H}}$, as functions ((2)) of the primitive random variables r_j, are themselves random variables. Since the magnitudes $|E_{\mathbf{H}}|$ are known from the diffraction experiment, (1) and (2) imply that the phases $\phi_{\mathbf{H}}$ also are random variables. Hence it makes sense to ask for the conditional probability distribution of the special linear combinations of three phases, (6), the triplets $\phi_{\mathbf{HK}}$, assuming that the values of an appropriate set of observed magnitudes $|E|$ have been given.

For Fixed H and K, the Conditional Probability Distribution of the Triplet $\Phi_{\mathbf{HK}}$

Under the assumptions of the previous section, the conditional probability distribution of the triplet $\phi_{\mathbf{HK}}$, (6), given the three magnitudes $|E_{\mathbf{H}}|$, $|E_{\mathbf{K}}|$, $|E_{\mathbf{H+K}}|$ is

$$\mathsf{P}(\frac{\Phi}{A_{\mathbf{HK}}}) = \frac{1}{2\pi I_0(A_{\mathbf{HK}})} \exp(A_{\mathbf{HK}} \cos \Phi)\,, \tag{7}$$

where Φ represents the triplet $\phi_{\mathbf{HK}}$, the parameter $A_{\mathbf{HK}}$ is defined by

$$A_{\mathbf{HK}} = \frac{2}{\sqrt{N}} |E_{\mathbf{H}} E_{\mathbf{K}} E_{\mathbf{H+K}}| > 0 \tag{8}$$

and I_0 is the modified Bessel function. Since $A_{\mathbf{HK}} > 0$, the distribution (7) implies that the mode of $\phi_{\mathbf{H}}$ is zero and the conditional expectation (or average) of $\cos\phi_{\mathbf{H}}$, given $A_{\mathbf{HK}}$, is

$$\mathsf{E}(\cos \phi_{\mathbf{HK}}) = \frac{I_1(A_{\mathbf{HK}})}{I_0(A_{\mathbf{HK}})} > 0\,, \tag{9}$$

where I_1 is the modified Bessel function. It is also readily confirmed that the larger the value of $A_{\mathbf{HK}}$ the smaller is the conditional variance of $\cos\phi_{\mathbf{HK}}$, given $A_{\mathbf{HK}}$. It is to be stressed that the conditional expected value of the cosine, (9), is always positive since $A_{\mathbf{HK}} > 0$.

Minimal Principle

It is assumed that a crystal structure consisting of N identical atoms in the unit cell is fixed, but unknown, that the magnitudes $|E|$ of the normalized structure factors E are known, and that a sufficiently large base of phases, corresponding to the largest magnitudes $|E|$, is specified. In view of (9), one is led to construct the so-called minimal function:

$$R(\phi) = \frac{1}{\sum_{\mathbf{H},\mathbf{K}} A_{\mathbf{HK}}} \times \sum_{\mathbf{H},\mathbf{K}} A_{\mathbf{HK}} \left\{ \cos \phi_{\mathbf{HK}} - \frac{I_1(A_{\mathbf{HK}})}{I_0(A_{\mathbf{HK}})} \right\}^2 \tag{10}$$

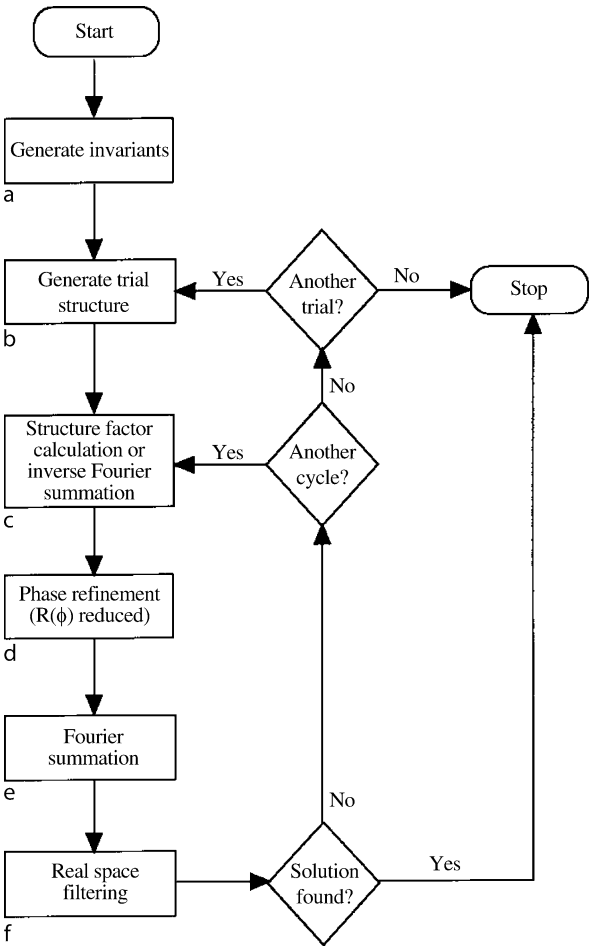
which, because of (6), is seen to be a function of the phases ϕ, dependent on known magnitudes $|E|$. Again in view of (9), one is led to conjecture that the global minimum of $R(\phi)$ constrained to satisfy the identities (5) yields the correct values of all the phases (the *minimal principle*). It is to be emphasized that the unconstrained global minimum of $R(\phi)$ does not give us the answer we seek, nor do any of the (innumerable) local minima.

Computer Program 'Shake and Bake'

The six-part shake and bake phase determination procedure, shown by the flow diagram in Fig. 1, combines minimal-function phase refinement and real-space filtering. It is an iterative process that is repeated until a solution is achieved or a designated number of cycles have been performed. With reference to Fig. 1, the major steps of the algorithm are described next.

A) Generate invariants. Normalized structure-factor magnitudes ($|E|$'s) are generated by standard scaling methods and the triplet invariants that involve the largest corresponding $|E|$'s are generated. Parameter choices that must be made at this stage include the numbers of phases and triplets to be used. The total number of invariants is ordinarily chosen to be at least 100 times the number of atoms whose positions are to be determined.

B) Generate trial structure. A trial structure or model is generated that is comprised of a number of randomly positioned atoms equal to the number of atoms in the unit cell. The starting coordinate sets



Phase Problem in X-ray Crystallography: Shake and Bake Approach, Figure 1
Flow chart for Shake and Bake, the minimal-function phase refinement and real-space filtering procedure

are subject to the restrictions that no two atoms are closer than a specified distance (normally 1.2Å) and that no atom is within bonding distance of more than four other atoms.

C) Structure-factor calculation. A normalized structure-factor calculation based on the trial coordinates is used to compute initial values for all the desired phases simultaneously. In subsequent cycles, peaks selected from the most recent Fourier series are used as atoms to generate new phase values.

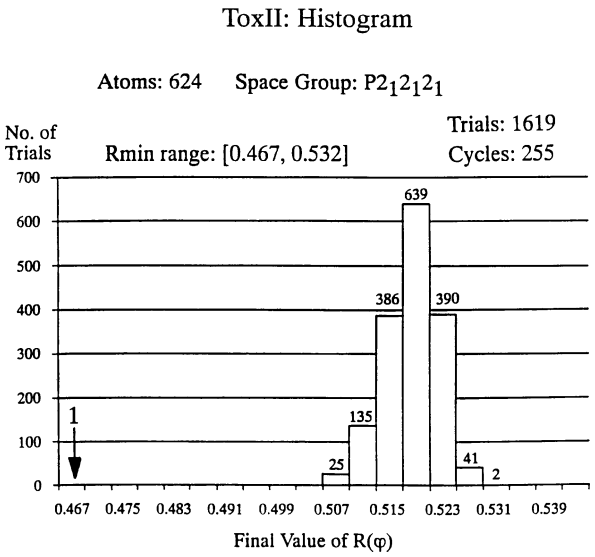
D) Phase refinement. The values of the phases are perturbed by a parameter-shift method in which $R(\phi)$, which measures the mean-square difference between estimated and calculated structure invariants, is reduced in value. $R(\phi)$ is initially computed on the basis of the set of phase values obtained from the structure-factor calculation in step C. The phase set is ordered in decreasing magnitude of the associated $|E|$'s. The value of the first phase is incremented by a preset amount and $R(\phi)$ is recalculated. If the new calculated value of $R(\phi)$ is lower than the previous one, the value of the first phase is incremented again by the preset amount. This is continued until $R(\phi)$ no longer decreases or until a predetermined number of increments has been applied to the first phase. A completely analogous course is taken if, on the initial incrementation, $R(\phi)$ increases, except that the value of the first phase is decremented until $R(\phi)$ no longer decreases or until the predetermined number of decrements has been applied. The remaining phase values are varied in sequence as just described. Note that, when the ith phase value is varied, the new values determined for the previous $i-1$ phases are used immediately in the calculation of $R(\phi)$. This process, when convergent, often, but not always, yields the constrained global minimum of $R(\phi)$. The stepsize and number of steps are variables whose values must be chosen.

E) Fourier summation. Fourier summation is used to transform phase information into an electron-density map (refer to (3)). The grid size must be specified.

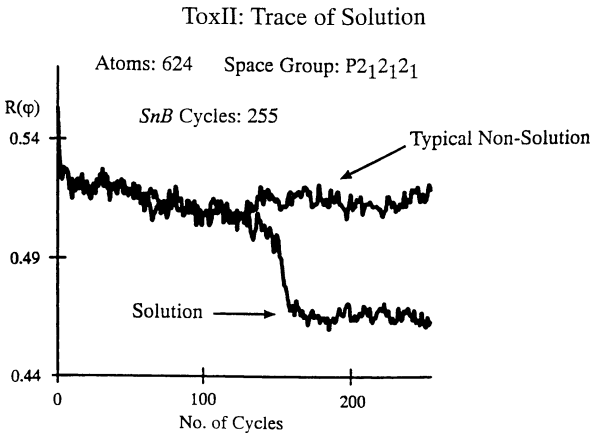
F) Real-space filtering (identities among phases imposed). Image enhancement has been accomplished by a discrete electron-density modification consisting of the selection of a specified number of the largest peaks on the Fourier map for use in the next structure-factor calculation. The simple choice, in each cycle, of a number of the largest peaks corresponding to the number of expected atoms has given satisfactory results. No minimum-interpeak-distance criterion is applied at this stage.

Applications

Shake and Bake has been tested successfully, with no failure, using experimentally measured atomic resolution data for some 30 structures, many of which had been difficult to solve with existing techniques or had defeated all previous attempts. These structures range in size from 25 to 1000 independent nonhydrogen



Phase Problem in X-ray Crystallography: Shake and Bake Approach, Figure 2
Final values of the minimal function $R(\phi)$ after 255 cycles for 1619 shake and bake trials for the 624 atom Tox II structure clearly showing the separation between the single solution and the 1618 nonsolutions



Phase Problem in X-ray Crystallography: Shake and Bake Approach, Figure 3
The course of $R(\phi)$ for Tox II, as a function of cycle number, for the solution trial and for a typical nonsolution trial

atoms. Although a number of these structures had been previously known, this fact was not used in these applications. In all cases those trials which led to solutions were readily identified by the behavior of the minimal function $R(\phi)$ (See Fig. 2; Fig. 3).

A Notable Application: Determination by Shake and Bake of the Previously Known Crystal Structure of Toxin II From the Scorpion

Androctorius australis Hector. This structure, consisting of 64 amino acid residues (512 protein atoms, the heaviest comprising four disulfide bridges) and 112 water molecules (a total of 624 atoms), crystallizes in the space group $P2_12_12_1$ and diffracts to a resolution of 0.96Å. A total of 50,000 triplets having the largest values of $A_{\mathbf{HK}}$ were generated from the 5, 000 phases with the largest values of $|E|$ and used in the definition of the minimal function $R(\phi)$. A total of 1619 Shake and Bake trials were run, each for 255 cycles. The final value of $R(\phi)$ for the trial which led to the solution was 0.467, the value of the constrained global minimum of $R(\phi)$. The range of final values of $R(\phi)$ for the remaining 1618 trials was [0.507, 0.532] (see histogram, Fig. 2), clearly nonsolutions.

Figure 3 shows the course of the minimal function $R(\phi)$, as a function of cycle number, for the trial which led to the solution and for a typical nonsolution trial. Both trials show almost identical behavior for some 130 cycles when $R(\phi)$ for the solution trial drops precipitously from a value of about 0.50 to 0.467 and remains at about that level for all remaining cycles. For the nonsolution trial however, $R(\phi)$ oscillates between 0.51 and 0.52 for all remaining cycles.

See also

- ► Adaptive Simulated Annealing and its Application to Protein Folding
- ► Genetic Algorithms
- ► Global Optimization in Lennard–Jones and Morse Clusters
- ► Global Optimization in Protein Folding
- ► Molecular Structure Determination: Convex Global Underestimation
- ► Monte-Carlo Simulated Annealing in Protein Folding
- ► Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach
- ► Packet Annealing
- ► Protein Folding: Generalized-Ensemble Algorithms
- ► Simulated Annealing
- ► Simulated Annealing Methods in Protein Folding

References

1. Detitta G, Weeks C, Thuman P, Miller R, Hauptman H (1994) Structure solution by minimal function phase refinement and fourier filtering: Theoretical basis. Acta Crystallogr A50:203–210
2. Hauptman H (1991) A minimal principle in the phase problem. Proc. Internat. School Crystallographic Computing, Oxford University Press, Oxford, pp 324–332
3. Weeks CM, Detitta GT, Hauptman HA, Thuman P, Miller R (1994) Structure solution by minimal function phase refinement and Fourier filtering II. Implementation and applications. Acta Crystallogr A50:210–220

Piecewise Linear Network Flow Problems

PLNFP

DUKWON KIM
University Florida, Gainesville, USA

MSC2000: 90B10

Article Outline

Keywords
Convex PLNFP
FCNFP
Concave PLNFP
Indefinite PLNFP
Applications
  Transportation Problems
  Location Problems
Concluding Remarks
See also
References

Keywords

Piecewise linear arc cost; Fixed charge; Minimum cost network flow; Global optimization

In this article, the minimum cost network flow problems [38] are considered with piecewise linear arc costs, so called *piecewise linear minimum cost network flow problem* (*PLNFP*). As a special subclass of minimum cost network flow problems, general piecewise linear network problems can be classified according to the type of piecewise cost functions. Using general network flow constraints, PLNFP can be stated as follows:

Given a directed graph $G = (N, A)$ consisting of a set N of m nodes and a set A of n arcs, then solve [PLNFP]:

$$\text{minimize } f(x) = \sum_{(i,j)\in A} f_{ij}(x_{ij})$$

subject to

$$\sum_{(k,i)\in A} x_{ki} - \sum_{(i,k)\in A} x_{ik} = b_i\,, \quad \forall\, i \in N\,, \tag{1}$$

$$0 \le x_{ij} \le u_{ij}\,, \quad \forall (i,j) \in A\,, \tag{2}$$

where f is separable and each f_{ij} is piecewise linear. For instance, the arc cost $f_{ij}(x_{ij})$ can be defined as follows:

$$f_{ij}(x_{ij}) = \begin{cases} c_{ij}^1 x_{ij} + s_{ij}^1, & 0 \le x_{ij} < \lambda_{ij}^1, \\ c_{ij}^2 x_{ij} + s_{ij}^2, & \lambda_{ij}^1 \le x_{ij} < \lambda_{ij}^2, \\ \vdots & \vdots \\ c_{ij}^{r_{ij}} x_{ij} + s_{ij}^{rij}, & \lambda_{ij}^{r_{ij}-1} \le x_{ij} \le \lambda_{ij}^{r_{ij}}, \end{cases}$$

where λ_{ij}^k for $k = 1$ to $r_{ij} - 1$ are breakpoints in the given interval $[0, u_{ij}]$. The constraints in (1) are called the *conservation of flow equations*. The constraints in (2) are called *capacity constraints on the arc flows*. The problem is *uncapacitated* if $u_{ij} = \infty$, $\forall (i,j) \in A$.

Three specific classes of PLNFP are identified based on the arc costs, f_{ij} as follows:

1) Convex PLNFP.
2) Concave PLNFP.
3) Indefinite PLNFP.

In some cases, indefinite PLNFP is called discontinuous PLNFP, since it usually results from a set of discontinuities in the arc cost functions. Since the *fixed charge network flow problem* (FCNFP) has a very close relation to the PLNFP, it is important to understand the special structure of FCNFP to solve PLNFP. Due to the global optimality property of *concave minimization* [19,33], global solutions can be obtained at extreme points of feasible regions for the cases of FCNFP and concave PLNFP. A recent survey on minimum concave-cost network flow problems can be found in [16].

Convex PLNFP

Firstly, let us consider a convex PLNFP. Suppose constraints are all linear, and the cost function to be mini-

mized is separable and piecewise linear. Then the proportionality assumption is violated for the cost function. However, if the piecewise linear function is convex, the problem can still be modeled as an LP [29].

Let $f_{ij}(x_{ij})$ denote the contribution of x_{ij} to the separable objective function. Suppose that there are $r_{ij} - 1$ breakpoints at which $f_{ij}(x_{ij})$ changes slope, such that

$$0 = \lambda_{ij}^0 < \lambda_{ij}^1 < \cdots < \lambda_{ij}^{r_{ij}-1} < \lambda_{ij}^{r_{ij}} = u_{ij} .$$

Let the slope in the subinterval $\lambda_{ij}^{k-1} \le x_{ij} \le \lambda_{ij}^k$ be c_{ij}^k for $k = 1$ to r_{ij}, and let y_{ij}^k be the portion of x_{ij} lying in the kth subinterval, λ_{ij}^{k-1} to λ_{ij}^k, (i. e., y_{ij}^k is the length of the overlap of the interval 0 to x_{ij} with the subinterval λ_{ij}^{k-1} to λ_{ij}^k), $k = 1$ to r_{ij}. When defined in this manner, the new variables $y_{ij}^1, \ldots, y_{ij}^{r_{ij}}$ partition x_{ij} such that

$$x_{ij} = y_{ij}^1 + \cdots + y_{ij}^{r_{ij}} . \tag{3}$$

These variables are subject to the constraints:

$$\begin{cases} y_{ij}^1 \le \lambda_{ij}^1 \\ 0 \le y_{ij}^2 \le \lambda_{ij}^2 - \lambda_{ij}^1 \\ \quad \vdots \\ 0 \le y_{ij}^{r_{ij}-1} \le \lambda_{ij}^{r_{ij}-1} - \lambda_{ij}^{r_{ij}-2} \\ 0 \le y_{ij}^{r_{ij}} \le \quad \lambda_{ij}^{r_{ij}} - \lambda_{ij}^{r_{ij}-1} \end{cases} \tag{4}$$

and:

for every t, if $y_{ij}^t > 0$, then each of y_{ij}^k is equal to its upper bound

$$\lambda_{ij}^k - \lambda_{ij}^{k-1} , \quad \forall k < t . \tag{5}$$

Defining the new variables as shown above, it is clear that $f_{ij}(x_{ij})$ is equal to $c_{ij}^1 y_{ij}^1 + \cdots + c_{ij}^{r_{ij}} y_{ij}^{r_{ij}}$. If the original separable piecewise linear objective function to be minimized is continuous and convex so that it holds the following *increasing* conditions

$$c_{ij}^1 < \cdots < c_{ij}^{r_{ij}} , \tag{6}$$

constraints on the new variables of the type (5) can be ignored in the transformed model. Since convex PLNFP is reformulated as an LP using the above technique and has specially structured network constraints, it can be solved efficiently in polynomial time. If f_{ij} is not continuous the slopes do not satisfy the condition (6), then the constraints (5) must be specifically included in the model. Since these constraints are not linear, the transformed model is no longer an LP.

FCNFP

Due to the similarity of its structure with the piecewise linear case, the fixed charge network flow problem (FCNFP) has close relations to PLNFP. It is very important and useful to investigate some features of FCNFP even if FCNFP does not belong to the class of PLNFP.

The FCNFP is a special case of minimum concave cost network flow problems (MCNFP) [19], whose arc cost function has a discontinuity at the origin. The arc cost function $f_{ij}(x_{ij})$ of the FCNFP has a form

$$f_{ij}(x_{ij}) = \begin{cases} 0 & \text{if} \quad x_{ij} = 0 \\ s_{ij} + c_{ij} x_{ij} & \text{if} \quad x_{ij} > 0, \end{cases} \tag{7}$$

where $s_{ij} \ge 0$ is a fixed cost for arc $(i, j) \in A$.

In many practical problems, the cost of an activity is the sum of a fixed cost and a cost proportional to the level of the activity. FCNFP is obtained by imposing a fixed cost of $s_{ij} \ge 0$ if there is positive flow on arc (i, j) and a variable cost c_{ij}. Due to the discontinuity of f_{ij}, the problem can be transformed to a 0–1 mixed integer programming problem by introducing n binary variables, indicating whether the corresponding activity is being carried out or not. Assuming $s_{ij} > 0$, f_{ij} can be replaced by

$$f_{ij} = c_{ij} x_{ij} + s_{ij} y_{ij}$$

with

$$x_{ij} \ge 0 \quad \text{and} \quad y_{ij} = \begin{cases} 0 & \text{if} \quad x_{ij} = 0 \\ 1 & \text{if} \quad x_{ij} > 0. \end{cases} \tag{8}$$

The above condition (8) can be incorporated into the capacity constraints to yield

$$0 \le x_{ij} \le u_{ij} y_{ij}, \quad y_{ij} \in \{0, 1\}.$$

Hence we obtain the following formulation of the fixed charge network flow problem $\text{FCNFP}_{\text{MIP}}$:

$$\min \sum_{(i,j)\in A} (c_{ij} x_{ij} + s_{ij} y_{ij})$$

subject to

$$Mx = b \tag{9}$$

$$0 \le x_{ij} \le u_{ij} y_{ij}\,, \quad (i,j) \in A\,, \tag{10}$$

$$y_{ij} \in \{0,1\}\,, \quad (i,j) \in A\,, \tag{11}$$

where M is an $m \times n$ node-arc incidence matrix and b is an m-dimensional column vector. This $\text{FCNFP}_{\text{MIP}}$ can be solved by using any type of classical branch and bound algorithms that use LP relaxations [30]. These LP relaxations can be solved efficiently by existing linear network algorithms exploiting the special structure of their feasible domain [18].

As we can see later, many concave and indefinite PLNFPs can be reduced to a FCNFP model by introducing new variables and modifying problem structures. It is noticed that FCNFP models reduced from original PLNFPs can be also transformed to a 0–1 mixed integer programming problem. Consequently, the size of the resulting model grows very fast even if the original PLNFP is just of medium size. This stimulates the reason that many researchers have developed new efficient schemes to improve their exact methods (especially the *branch and bound method*). Indeed, the computational effort and memory requirement to solve large scale FCNFP models have been gradually reduced in various application areas. Yet, since there is a limitation for improving exact solution methods to solve the problem in practical sense, developing a effective approximate method is still in need.

Concave PLNFP

As different from the convex case, concave PLNFPs are more difficult to solve since we cannot use the same technique in the convex case to reduce the problem into an LP. However, a concave PLNFP can be transformed to a fixed charge network problem in an extended network. The size of the extended network depends on the number of linear pieces in each arc cost function. An *arc separation procedure* (ASP) is required to solve the problem in this way and ASP can be valid due to the concavity of arc cost functions.

Let us consider an arc $(i, j) \in A$ and its arc cost f_{ij}, and suppose f_{ij} has r_{ij} linear pieces as defined previously. Then arc (i, j) can be separated into r_{ij} arcs between nodes i and j for $(i, j) \in A$. Each separated arc $(i, j)^k$ for $k = 1$ to r_{ij} has a fixed charge cost function f_{ij}^k (see Fig. 1) defined by

$$f_{ij}^k(x_{ij}) = \begin{cases} 0 & \text{if} \quad x_{ij} = 0 \\ s_{ij}^k + c_{ij}^k x_{ij} & \text{if} \quad x_{ij} > 0. \end{cases}$$

This extended network is denoted by $G_e(N, A_e)$ where the number of arcs $|A_e|$ is given by

$$n_e = |A_e| = \sum_{(i,j)\in A} r_{ij}\,.$$

After the ASP modification shown in Fig. 2, the original concave piecewise linear objective function can be expressed as a sum of fixed charge arc cost functions as follows:

$$\begin{aligned} f(x) &= \sum_{(i,j)\in A} f_{ij}(x_{ij}) \\ &= \sum_{(i,j)\in A} \sum_{k=1}^{r_{ij}} f_{ij}^k(x_{ij}^k) \\ &= \sum_{(i,j)\in A} \sum_{k=1}^{r_{ij}} (c_{ij}^k x_{ij}^k + s_{ij}^k)\,. \end{aligned} \tag{12}$$

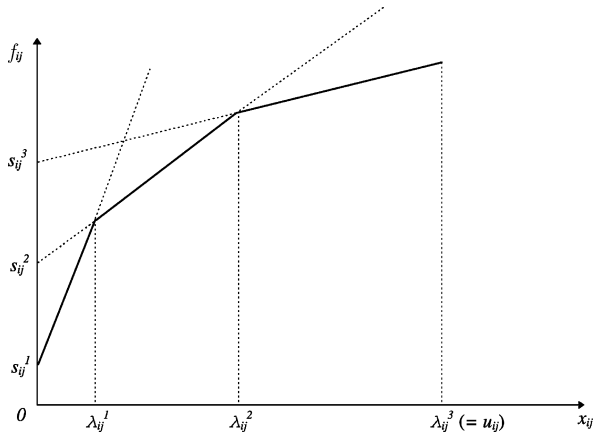
It is easy to see that the equality in (12) can not be true in general cases without a set of constraints to restrict a domain for each separated arc cost function. However, due to the following property from the concavity of arc cost functions:

$$c_{ij}^1 > \cdots > c_{ij}^{r_{ij}} > 0\,, \tag{13}$$

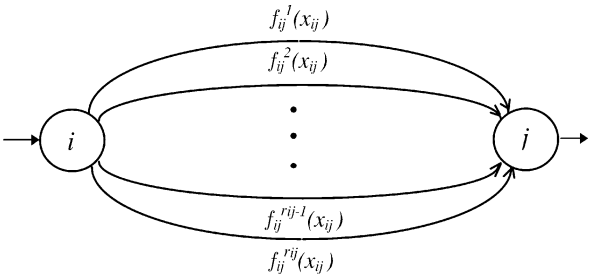
the equality holds for true in this case. More precisely, $f_{ij}(x_{ij})$ is equal to at most one of arc cost among all separated arc costs between node i and j at the optimality of minimization problems. This argument can be generalized as the following property.

Proposition 1 *Given an extended network described above, if a positive flow x_{ij}^* is optimal for a minimum concave PLNFP and if $\lambda_{ij}^{q-1} < x_{ij}^* \le \lambda_{ij}^q$ for $1 \le q \le r_{ij}$, then it takes only one arc, $(i, j)^q$ (i. e. qth arc) among all separated arcs between node i and j, $(i, j)^k$ for $k = 1, \ldots, r_{ij}$.*

Based on this, the original concave PLNFP can be reduced to a FCNFP with the objective function given



Piecewise Linear Network Flow Problems, Figure 1
Example of a concave piecewise linear arc cost function



Piecewise Linear Network Flow Problems, Figure 2
Arc separation procedure

in (12) and some extended network constraints corresponding to the constraints in (1). The resulting formulation of a concave PLNFP is shown as follows:

$$\min f(x) = \sum_{(i,j)\in A} \sum_{k=1}^{r_{ij}} f_{ij}^k(x_{ij}^k)$$

subject to

$$\sum_{(l,i)\in A} \sum_{k=1}^{r_{ij}} x_{li}^k - \sum_{(i,l)\in A} \sum_{k=1}^{r_{ij}} x_{il}^k = b_i, \quad \forall i \in N,$$

$$\sum_{k=1}^{r_{ij}} x_{ij}^k = x_{ij}, \forall (i,j) \in A,$$

$$x_{ij}^k \geq 0, \forall (i,j) \in A, \quad k = 1, \ldots, r_{ij},$$

$$0 \leq x_{ij} \leq u_{ij}, \forall (i,j) \in A. \quad (14)$$

In the formulation, it is clear that any other set of constraints is not necessary to specify lower and upper bounds for separated arcs $(i, j)^k \in A_e$, $\forall i, j \in N$, due to the above proposition.

As a result, the solution of concave PLNFP can be found by solving the fixed charge network problem formulated as above. Thus, developing an efficient algorithm to solve a FCNFP (*exactly or approximately*) is a key to solve PLNFPs in this approach.

Indefinite PLNFP

Lastly, we consider a PLNFP with indefinite arc cost functions, which is the most difficult case in this class. The major difficulty to find exact solutions for indefinite PLNFPs is originated from the structure of their arc cost function. Obviously such cost functions are neither convex nor concave, and possibly have a finite set of discontinuities. However, due to the nature of real world applications of the model, we focus on two certain types of arc cost function in indefinite PLNFP models, so called *staircase arc cost function* [6,17,26] and *sawtooth arc cost function* [26] arc cost functions, respectively.

Both arc cost functions have a very similar structure in overall shape, however, they have a different aspect at breakpoints. It can be described in mathematical form as follows:

- 'Staircase' arc cost function (see Fig. 3):

$$f_{ij}^{k-1}(\lambda_{ij}^{k-1}) < f_{ij}^k(\lambda_{ij}^{k-1} + \varepsilon),$$

for any $\varepsilon > 0$ and $k = 2, \ldots, r_{ij}$.

- 'Sawtooth' arc cost function (see Fig. 4):

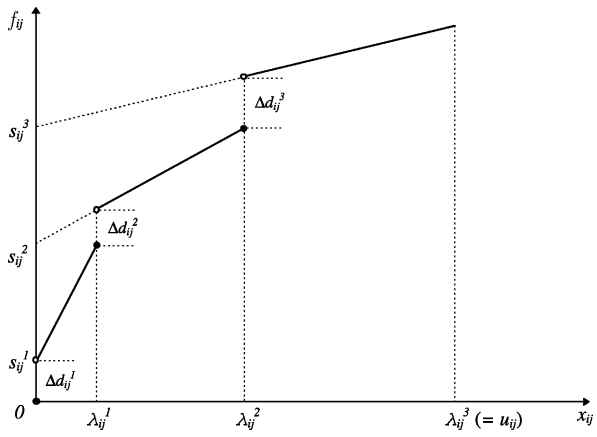
$$f_{ij}^{k-1}(\lambda_{ij}^{k-1} - \varepsilon) > f_{ij}^k(\lambda_{ij}^{k-1}),$$

for any $\varepsilon > 0$ and $k = 2, \ldots, r_{ij}$.

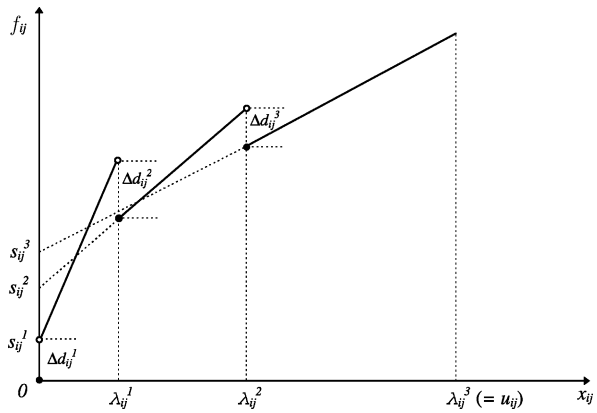
Moreover, it is assumed that the property of slopes shown in (13) is still valid since it is a very general phenomenon in real applications. Note that extreme point solutions are not guaranteed in this cases since objective functions are no longer concave.

Now, we introduce an equivalent $(\text{FCNFP})_{\text{MIP}}$ formulation of the problem with some additional parameters and binary variables. Let us define the size of interval between adjacent breakpoints as

$$\Delta\lambda_{ij}^k = \lambda_{ij}^k - \lambda_{ij}^{k-1}, \forall (i,j) \in A, \quad k = 1, \ldots, r_{ij}, \quad (15)$$



Piecewise Linear Network Flow Problems, Figure 3
Staircase arc cost function



Piecewise Linear Network Flow Problems, Figure 4
Sawtooth arc cost function

and define the gap of function values at each breakpoint in arc cost functions as

$$\begin{aligned}\Delta d_{ij}^k &= (c_{ij}^k \lambda_{ij}^{k-1} + s_{ij}^k) - (c_{ij}^{k-1}\lambda_{ij}^{k-1} + s_{ij}^{k-1}) \\ &= s_{ij}^k - s_{ij}^{k-1} + (c_{ij}^k - c_{ij}^{k-1})\lambda_{ij}^{k-1}, \ \forall (i,j) \in A, \ k\,,\end{aligned} \tag{16}$$

where $s_{ij}^0 = 0$ and $c_{ij}^0 = 0$ (also clearly $\Delta\, d_{ij}^1 = s_{ij}^1$). We now let x_{ij}^k be the part of x_{ij} that lies within level k (i. e. kth subinterval), in the following sense,

$$x_{ij}^k = \begin{cases} 0 & \text{if } x_{ij} \le \lambda_{ij}^{k-1}, \\ x_{ij} - \lambda_{ij}^{k-1} & \text{if } \lambda_{ij}^{k-1} < x_{ij} \le \lambda_{ij}^k, \\ \Delta\lambda_{ij}^k & \text{if } x_{ij} \ge \lambda_{ij}^k, \end{cases} \tag{17}$$

and we obtain the following equation for substitution into $(\text{FCNFP})_{\text{MIP}}$ model. We then introduce new binary variables defined by

$$y_{ij}^k = \begin{cases} 1 & \text{if } \lambda_{ij}^{k-1} < x_{ij} \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

Using (15)–(18), the indefinite PLNFP under consideration can be formulated as an MIP version of FCNFP as follows:

$$\min \sum_{(i,j)\in A} \sum_{k=1}^{r_{ij}} (c_{ij}^k x_{ij}^k + \Delta d_{ij}^k y_{ij}^k)$$

subject to constraints in (1) and

$$x_{ij} = \sum_{k=1}^{r_{ij}} x_{ij}^k \quad \forall (i,j)\,, \tag{19}$$

$$x_{ij}^k \le \Delta\lambda_{ij}^k y_{ij}^k \quad \forall (i,j),\ k = 1,\ldots,r_{ij}\,, \tag{20}$$

$$x_{ij}^{k-1} \ge \Delta\lambda_{ij}^{k-1} y_{ij}^k \quad \forall (i,j),\ k = 2,\ldots,r_{ij}\,, \tag{21}$$

$$x_{ij}^k \ge 0 \quad \forall (i,j),\ k = 1,\ldots,r_{ij}\,, \tag{22}$$

$$y_{ij}^k \in \{0,1\} \quad \forall (i,j),\ k = 1,\ldots,r_{ij}\,. \tag{23}$$

It is noticed that combining one constraint from (20) and one from (21) yields $\Delta\lambda_{ij}^{k-1} y_{ij}^k \le x_{ij}^{k-1} \le \Delta\lambda_{ij}^{k-1} y_{ij}^{k-1}$, which implies $y_{ij}^k \le y_{ij}^{k-1}$, $\forall (i,j)$, $k > 1$.

There is another approach to formulate the problem as a concave minimum cost network flow problem (MCNFP) model. In [26], B.W. Lamar described an equivalent formulation of MCNFP with general nonlinear arc costs (including the problem considered in this section) as a concave MCNFP on an extended network. The equivalence between the problems is based on converting each arc with an arbitrary cost function in the original problem into an arc with a concave piecewise linear cost function in series with a set of parallel arcs, each with a linear arc cost function (cf. [26] for details). Thus, the resulting problem is a concave MCNFP, which is different from the FCNFP formulation model shown above.

Applications

Piecewise linear network models have a number of applications in various areas such as transportation problems, location problems, distribution problems, communication network design problems, and economic lot-sizing problems. Due to the structure of objective functions (especially in concave and indefinite PLNFP models), many real-world situations listed above can be modeled as PLNFP. There are major applications in the following two fields which have been studied extensively by researchers.

Transportation Problems

The first major field is transportation-related problems with concave cost functions including fixed charge cost functions. The concave cost functions in this field are usually assumed to be piecewise linear in many cases. A number of algorithms developed with different schemes and their computational results have been reported. These can be found in a limited list of references [2,3,4,7,11,15,20,21,24,25,31,34,35].

The category of exact solution approach contains diverse techniques based on extreme point ranking, branch and bound, and dynamic programming methods. Since the problems can be formulated as MIP models, branch and bound approach with various branching schemes has been a major interest in the literature.

K.G. Murty [28] introduced an extreme point ranking method for solving fixed charge problems. P. McKeown [27] extended Murty's method to avoid some degeneracy of the problem. Recently, Pardalos [32] discussed a range of enumerative techniques based on vertex enumeration or extreme point ranking.

P. Rech and L.G. Barton [34] investigated a nonconvex transportation algorithm using branch and bound approach to problems with piecewise linear cost functions. These functions are approximated by a convex envelope and solved using out-of-kilter method. C.T. Bornstein and R. Rust [6] specialized this approach to the problem with staircase cost function, using successive linearizations of the objective function. P.T. Thach [37] proposed a method for decomposing the problem with a staircase structure into a sequence of much smaller subproblems.

Lamar [25] developed a branch and bound approach for cases of capacitated MCNFP in which the costs consist of piecewise linear segments. The problem is formulated an MIP, with the branching variables determining which linear cost region an arc flow falls into.

Recently, D. Kim and P.M. Pardalos [23] developed a heuristic procedure for solving general FCNFP without formulating it as an MIP. The procedure is consist of solving a series of LPs to update slopes and searching extreme points of the convex feasible region with the updated slopes. This approach provides a potential possibility of parallel implementation with different initial solutions to improve the quality of solutions. Some heuristic approaches can be found in [8,11,21,24].

Location Problems

Another major application area is to solve location problems. Since the problem in this field is to locate facilities and determine the size of facilities to minimize total cost [13], it naturally involves fixed costs and/or piecewise linear costs. Since solution methods in this field have used network formulation in many cases, they are quite similar to those for solving FCNFP or PLNFP [1,10,12,22,36]. However, exploiting their certain problem structures, there are some Lagrangian approaches [5,14] to the problems.

Recently, K. Holmberg [17] proposed a decomposition and linearization approach for solving the facility location problem with staircase costs. A comparison of heuristic and relaxation approaches in this field can be found in [9].

Concluding Remarks

In this article, three categories of PLNFP are identified and formulated in general formats. Some properties of problems in each category are investigated to show the insight of problems including FCNFP. The concave PLNFP is formulated as a FCNFP in MIP structure exploiting the concavity of arc cost functions. Moreover, the indefinite (nonconvex) PLNFP is also transformed to a FCNFP with a reduced feasible region. This implies that the extreme point solution of the transformed FCNFP may not be an extreme point solution of the original indefinite PLNFP.

A major advantage of the formulations introduced in the paper is that solutions can be found by solving fixed charge problems instead of solving difficult nonconvex optimization problems. As we can see in the

transformation to FCNFP, the size of FCNFP is usually quite large because of new binary variables introduced in the model. Thus, developing an efficient algorithm for large scale FCNFP can provide a key to solve concave and indefinite PLNFP in practice.

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Global Supply Chain Models
- ▶ Inventory Management in Supply Chains
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Nonoriented Multicommodity Flow Problems
- ▶ Operations Research Models for Supply Chain Management and Design
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Akinc U, Khumawala BM (1977) An efficient branch and bound algorithm for the capacitated warehouse location problem. Managem Sci 23:585–594
2. Balakrishnan A, Graves SC (1989) A composite algorithm for a concave-cost network flow problem. Networks 19:175–202
3. Balinski ML (1961) Fixed-cost transportation problems. Naval Res Logist Quart 8:41–54
4. Barr RS, Glover F, Klingman D (1981) A new optimization method for large scale fixed charge transportation problems. Oper Res 29:448–463
5. Beasly JE (1993) Lagrangean heuristics for location problem. Europ J Oper Res 65:383–399
6. Bornstein CT, Rust R (1988) Minimizing a sum of staircase functions under linear constraints. Optim 19:181–190
7. Cabot AV, Erenguc SS (1984) Some branch-and-bound procedures for fixed-cost transportation problems. Naval Res Logist Quart 31:145–154
8. Cooper L, Drebes C (1967) An approximate solution method for the fixed charge problem. Naval Res Logist Quart 14:101–113
9. Cornuejols G, Sridharan R, Thizy JM (1991) A comparison of heuristics and relaxations for the capacitated plant location problem. Europ J Oper Res 50:280–297
10. Davis PS, Ray TL (1969) A branch-and-bound algorithm for the capacitated facilities location problem. Naval Res Logist Quart 16:331–344
11. Diaby M (1991) Successive linear approximation procedure for generalized fixed-charge transportation problem. J Oper Res Soc 42:991–1001
12. Efroymson MA, Ray TL (1966) A branch-and-bound algorithm for plant location. Oper Res 14:361–368
13. Francis RL, McGinnis LF, White JA (1992) Facility layout and location: An analytical approach. second Prentice-Hall, Englewood Cliffs
14. Geoffrion A, McBride R (1978) Lagrangean relaxation applied to capacitated facility location problems. AIIE Trans 10:40–47
15. Gray P (1971) Exact solution of the fixed-charge transportation problem. Oper Res 19:1529–1538
16. Guisewite GM, Pardalos PM (1990) Minimum concave-cost network flow problems: Applications, complexity, and algorithms. Ann Oper Res 25:75–100
17. Holmberg K (1994) Solving the staircase cost facility location problem with decomposition and piecewise linearization. Europ J Oper Res 75:41–61
18. Horst R, Pardalos PM (eds) (1995) Handbook of Global Optimization. Kluwer, Dordrecht
19. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
20. Kennington J, Unger E (1976) A new branch-and-bound algorithm for the fixed-charge transportation problem. Managem Sci 22:1116–1126
21. Khang DB, Fujiwara O (1991) Approximate solutions of capacitated fixed-charge minimum cost network flow problems. Networks 21:689–704
22. Khumawala BM (1972) An efficient branch-and-bound algorithm for the warehouse location problem. Managem Sci 18:B718–B731
23. Kim D, Pardalos PM (1999) A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. Oper Res Lett 24:195–203
24. Kuhn H, Baumol W (1962) An approximative algorithm for the fixed charge transportation problem. Naval Res Logist Quart 9:1–15
25. Lamar BW (1993) An improved branch and bound algorithm for minimum concave cost network flow problems. J Global Optim 3:261–287
26. Lamar BW (1993) A method for solving network flow problems with general nonlinear arc costs. In: Du D-Z, Pardalos

PM (eds) Network Optimization Problems. World Sci., Singapore
27. McKeown P (1975) A vertex ranking procedure for solving the linear fixed-charge problem. Oper Res 23:1183–1191
28. Murty KG (1968) Solving the fixed charge problem by ranking the extreme points. Oper Res 16:268–279
29. Murty KG (1976) Linear and combinatorial programming. Wiley, New York
30. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
31. Palekar US, Karwan MH, Zionts S (1990) A branch-and-bound method for the fixed charge transportation problem. Managem Sci 36:1092–1105
32. Pardalos PM (1988) Enumerative techniques for solving some nonconvex global optimization problems. OR Spectrum 10:29–35
33. Pardalos PM, Rosen JB (1987) Constrained global optimization: Algorithms and applications. Lecture Notes Computer Sci, vol 268. Springer, Berlin
34. Rech P, Barton LG (1970) A non-convex transportation algorithm. In: Beale EML (ed) Applications of Mathematical Programming Techniques. English University Press, London
35. Sa G (1968) Concave programming in transportation networks. PhD Thesis, Sloan School Management, MIT
36. Soland RM (1974) Optimal facility location with concave costs. Oper Res 22:373–382
37. Thach PT (1990) A decomposition method for the min concave cost flow problem with a staircase structure. Japan J Appl Math 7:103–120
38. Zangwill WI (1968) Minimum concave-cost flows in certain network. Managem Sci 14:429–450

Pivoting Algorithms for Linear Programming Generating Two Paths

KONSTANTINOS PAPARRIZOS, NIKOLAOS SAMARAS, KONSTANTINOS TSIPLIDIS
Department Applied Informatics,
University Macedonia, Thessaloniki, Greece

MSC2000: 90C05, 90C33

Article Outline

Keywords
Introduction
Algorithm Description
Algorithm Justification
Computational Improvements
See also
References

Keywords

Simplex algorithm; Interior point algorithms; Exterior point method

We present a new class of *simplex type algorithms* for solving general linear problems. The distinguished characteristic of the new algorithms is their capability of generating two paths to the optimal solution. One path is of simplex type while the other is not. The basic solutions of the simplex path are not feasible. The nonsimplex path consists of feasible segments the endpoints of which lie on the boundary of the feasible region. Preliminary computational results indicate that the new algorithms are substantially faster than the classical simplex algorithm.

Introduction

The classical simplex algorithm [6] had been the most efficient method for solving practical linear problems until the middle of 1980s. Then N.K. Karmarkar [9] developed the first *interior point algorithm*. Subsequent research led to the development of efficient interior point algorithms which outperform the simplex algorithm on large linear problems. Despite this fact the development of pivoting algorithms that clearly outperform the classical simplex method remained of great interest. It seems that a new class of pivoting algorithms developed recently is more efficient than the simplex method.

The new algorithms differ radically from the classical simplex algorithm. The basic solutions generated are not feasible. In that sense the algorithms are *exterior point methods*. However, the algorithms generate a second path, which consists of feasible points. In fact, it consists of line segments the end point of which lie on the boundary of the feasible region. As a result, the movement between adjacent basic feasible solutions is avoided. The geometry reveals that the new algorithms are faster than the well known simplex method, a fact that is verified by the available preliminary computational results, see [2,5,8]. However, more computational results are needed to draw more safe conclusions.

The first exterior point simplex type algorithm developed by K. Paparrizos [11] for the assignment problem. Other exterior point algorithms for network flow

problems can be found in [1,10,15]. Paparrizos [12] generalized his exterior point method to the general linear problem by developing a dual in nature algorithm. Independently, a similar primal algorithm solving a sequence of linear subproblems is developed in [3]. The algorithms in [12] and [3] generate two paths. One path is feasible, the other is not. However, both paths are of simplex type. In [13] the algorithms in [12] and [3] are generalized, so that the feasible path is not of simplex type. The main algorithm presented in this paper is very similar to the algorithm described in [13]. The algorithm in [5] also generates two paths of simplex type. One path is feasible to the primal problem while the other is feasible to the dual problem. A generalization to this algorithms can be found in [14].

Algorithm Description

We will describe the algorithm using the linear problem in the standard form

$$(\mathrm{P}_1)\quad \begin{cases} \min & c^\top x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{cases}$$

where $c, x \in \mathbf{R}^n$, $b \in \mathbf{R}^m$, $A \in \mathbf{R}^{m \times n}$ and T denotes transposition. Without loss of generality we assume that A is of *full row rank*, i. e., the row rank of A is m (with $m < n$).

A submatrix of A consisting of m independent columns is called *basic matrix*. The jth column of a matrix or tableau A is denoted by a_j and the ith row by A_i. A column in a basic matrix B is called *basic*. The columns not in B are called *nonbasic*. The submatrix of A consisting of all the nonbasic variables is called a *nonbasic matrix* and it is denoted by N. Also, with B and N we denote the sets $B = \{j : a_j \text{ is basic}\}$, $N = \{j : a_j \text{ is nonbasic}\}$. The basic and nonbasic components of a vector x (respectively, c) are denoted by x_B, x_N (respectively, c_B, c_N). Setting $x_N = 0$ in the equality constraints of (P1) we have $Bx_B = b$. From the last equation we have

$$x_B = B^{-1}b$$

The solution $x_B = B^{-1}b$, $x_N = 0$ is called *basic*. A solution that satisfies all the constraints of (P1) is called *feasible*. Clearly, a basic solution is feasible if and only if $x_B \geq 0$.

Given a basic matrix B we can use the equality constraints of (P1) to express the basic components x_B as a function of the nonbasic components x_N:

$$x_B = B^{-1}b - B^{-1}Nx_N\ .$$

Substituting in the objective function of (P1) we have

$$\begin{aligned} z = c^\top x &= c_B^\top x_B + c_N^\top x_N \\ &= c_B^\top (B^{-1}b - B^{-1}Nx_N) + c_N^\top x_N \\ &= c_B^\top B^{-1}b + (c_N^\top - c_B^\top B^{-1}N)x_N\ . \end{aligned}$$

We set

$$z_j = c_j - c_B^\top B^{-1}a_j$$

and

$$H = -B^{-1}N\ .$$

The current basis B is *optimal* if $z_j \geq 0$, $j \in N$.

Now, we are ready to describe an exterior point algorithm. Let B be a nonoptimal basic matrix not necessarily feasible to (P1), i. e., $x_B = B^{-1}b \ngeq 0$. Set $Q = \{j\colon z_j < 0, j \in N\}$ and $R = N \sim Q = \{j\colon z_j \geq 0, j \in N\}$. If $Q = \emptyset$, then B is an optimal basis and x_B, x_N is an optimal solution to (P1). In this case the algorithms stops. Otherwise, a leaving variable $x_{B_r} = x_k$ is determined as follows.

First an improving direction d ($c^\intercal d < 0$) such that $d_R = 0$ and $d_Q > 0$ is constructed. The direction d is constructed in such a way so that the ray $\{\overline{x}\colon\ \overline{x} = x + td, t > 0\}$ must intersect the feasible region of problem (P1). In the case $x_B \geq 0$, d is very easily computed. Just set $d_R = 0$, $d_Q > 0$ and

$$d_B = \sum_{j \in Q} h_j d_j\ , \tag{1}$$

where $h_j = -B^{-1}a_j$. The leaving variable is determined by the following minimum ratio test,

$$\frac{x_{B_r}}{-d_{B_r}} = \min\left\{\frac{x_{B_i}}{-d_{B_i}}\colon\ d_{B_i} < 0\right\}\ . \tag{2}$$

Observe that the above minimum ratio test is precisely that used by the primal simplex algorithm. However, keep in mind that our algorithm is an exterior point method and, hence, the basic solutions are not feasible in general. Similar to the simplex method, the exterior

point algorithm stops in the case $d_B \geq 0$. In this case, it can be easily shown that problem (P1) is unbounded.

After the choice of the leaving variable $x_{B_r} = x_k$, we proceed to the choice of the entering variable x_l. It is worth mentioning that the exterior point algorithm, although of primal nature, chooses the entering and leaving variables in the reverse order of that of the primal simplex algorithm. First the leaving variable is chosen and then the entering one. In that sense, the exterior point algorithm resembles to the dual simplex method. The entering variable is chosen as follows. It is set

$$\frac{-z_p}{-h_{rp}} = \min\left\{\frac{-z_j}{-h_{rj}}:\ h_{rj} < 0, j \in Q\right\} \tag{3}$$

and

$$\frac{-z_s}{-h_{rs}} = \min\left\{\frac{-z_j}{-h_{rj}}:\ h_{rj} > 0, j \in R\right\}\ . \tag{4}$$

Then it is set $l = p$, if

$$\frac{-z_p}{-h_{rp}} \geq \frac{-z_s}{-h_{rs}}\ , \tag{5}$$

and $l = s$, otherwise. From the way the entering variable is chosen, it is easily seen that priotity is given to the variables in Q.

Let $\widehat{y}$ be the feasible point from which the ray $\{\overline{x}:\ x + td, t \geq 0\}$ exits the feasible region. Let $\widehat{B}$ be the new basis, $\widehat{B} = (B \sim \{k\}) \cup \{l\}$. Also, let $\widehat{x}$ be the new basic solution. The new direction is $\widehat{d} = \widehat{y} - \widehat{x}$ and a new iteration can be initiated.

Formally, the algorithm can be described as follows.

[0] (*Initialization*) Start with a feasible basis B. Set $N = \{1, \ldots, n\} \sim B$, $Q = \{j \in N : z_j < 0$, $R = N \sim Q$. Construct the improving direction d, where $d_R = 0$, $d_Q = 1$ and $d_B = \sum_{j \in Q} h_j$. Set also $d_0 = \sum_{j \in Q} z_j$.

[1] (*Test of optimality*) If $Q = \emptyset$, STOP. The current basic solution is optimal.

[2] (*Choice of the leaving variable*) If $d_B \geq 0$, STOP. Problem (P1) is unbounded. Otherwise, choose the leaving variable $x_{B_r} = x_k$ from relation (2).

[3] (*Choice of the entering variable*) Choose the entering variable x_l using the following two relations. Set

$$\frac{-z_p}{-h_{rp}} = \min\left\{\frac{-z_j}{-h_{rj}}:\ h_{rj} < 0, j \in Q\right\}$$

and

$$\frac{-z_s}{-h_{rs}} = \min\left\{\frac{-z_j}{-h_{rj}}:\ h_{rj} > 0, j \in R\right\}$$

If

$$\frac{-z_p}{-h_{rp}} \geq \frac{-z_s}{-h_{rs}}\ ,$$

then set $l = p$. Otherwise, i. e., if

$$\frac{-z_p}{-h_{rp}} < \frac{-z_s}{-h_{rs}}\ ,$$

set $l = p$.

[4] (*Pivot operation*) Set $B \leftarrow (B \sim \{k\}) \cup \{l\}$. If $l \in Q$, then $Q \leftarrow Q \sim \{l\}$ and $R \leftarrow R \cup \{k\}$. If $l \in R$ then $Q \leftarrow Q$ and $R \leftarrow (R \sim \{l\}) \cup \{k\}$. Let y be the boundary point from which d exits the feasible region and $\overline{x}$ the new basic solution. Set $d \leftarrow y - \overline{x}$, $x \leftarrow \overline{x}$ compute $d_0 = \sum_{j \in Q} z_j$ and go to Step 1.

Example 1 We further illustrate the algorithm by applying it to the following linear problem.

$$\begin{cases} \min & z = -2x_1 - 3x_2 + x_3 + 4x_4 \\ \text{s.t.} & x_1 + 4x_2 + 2x_3 + x_4 + x_5 = 8 \\ & x_1 + 3x_2 - 4x_3 - x_4 + x_6 = 8 \\ & x_1 + 3x_2 + 2x_3 - x_4 + x_7 = 10 \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0. \end{cases}$$

In Step 0 we set $B = [5, 6, 7]$. It is easily seen that $Q = [1, 2]$ and $R = [3, 4]$. Also, $(x_B)^\intercal = (x_5, x_6, x_7) = (8, 8, 10)$. We set $d_3 = d_4 = 0$ and $d_1 = d_2 = 1$. Then, we have $(d_B)^\intercal = (d_5, d_6, d_7) = (-1-4, -1-3, -1-3) = (-5, -4, -4)$. Finally, we have $z_1 = -2$, $z_2 = -3$, $z_4 = 1$, $z_5 = 4$, and $z_6 = z_7 = z_8 = 0$.

The algorithm does not stop in Step 1 because $Q \neq \emptyset$. As $d_B \not\geq 0$ we can choose the leaving variable. It is

$$\min\left\{\frac{x_5}{-d_5}, \frac{x_6}{-d_6}, \frac{x_7}{-d_7}\right\}$$
$$= \min\left\{\frac{8}{5}, \frac{8}{4}, \frac{10}{4}\right\} = \frac{8}{5} = \frac{x_5}{-d_5}\ .$$

Hence, $r = 1$ and $k = 5$ ($B[1] = 5$). Variable x_k is leaving.

For the choice of the entering variable we first compute

$$\min\left\{\frac{-z_1}{-h_{11}}, \frac{-z_2}{h_{12}}\right\} = \min\left\{\frac{2}{1}, \frac{3}{4}\right\} = \frac{-z_2}{h_{12}}\ .$$

There is no $j \in R$ such that $h_{1j} > 0$. Hence, $l = 2 \in Q$. The new sets B, Q and R are $B = [2, 6, 7]$, $Q = [1]$ and

$R = [3, 4, 5]$. The point y from which the improving direction exits the feasible region is given by $y = x + \bar{t}d$, where $\bar{t} = \frac{x_5}{(-d_5)} = \frac{8}{5}$. Hence, $y^\top = (0, 0, 0, 0, 8, 8, 10) + 8/5(1, 1, 0, 0, -5, -4, -4) = (8/5, 8/5, 0, 0, 0, 8/5, 18/5)$. It is easily verified that the new basic solution is $\overline{x}^\top = (0, 2, 0, 0, 0, 2, 4)$. The new direction is

$$\overline{d}^\top = (y - \overline{x})^\top = \left(\frac{8}{5}, \frac{-2}{5}, 0, 0, 0, \frac{-2}{5}, \frac{-2}{5}\right) .$$

Observe that $c^\top \overline{d} < 0$, $\overline{d}_Q > 0$ and $\overline{d}_R = 0$.

The algorithm is initialized with a basic feasible solution. As a result a two phase or a *big-M* method very similar to that of the primal simplex algorithm can be used to solve general linear problems.

Algorithm Justification

In this section we show that the algorithm correctly solves linear problems. We also show that the algorithm is finite under the usual non degeneracy assumptions.

Lemma 2 *If the algorithm stops at step 1, then the last basic solution is optimal.*

Proof The proof is by induction on the number of iterations. Assume that $z_j \geq 0$ for $j \in R$. It is easily seen that this induction hypothesis is satisfied by the initial basis. Let $\overline{B}$ be the next basis, $\overline{R}$ the updated set R and $\overline{z}_j$ the corresponding reduced costs. Then we have

$$\overline{z}_j = z_j - \frac{z_l}{h_{rl}} h_{rj} \geq 0, \quad j \neq k,\ j \in \overline{R} . \tag{6}$$

Because of the choice of the entering variable x_l we have $z_l/h_{rl} \geq 0$. If $h_{rj} \leq 0$, then $\overline{z}_j \geq 0$. If $h_{rj} > 0$, (6) is equivalent to

$$\frac{z_j}{h_{rj}} \geq \frac{z_l}{h_{rl}} ,$$

which holds because of the choice of the entering variable. If $j = k$, then $\overline{z}_k = \frac{z_l}{h_{rl}} \geq 0$. Hence, $\overline{z}_j \geq 0$ for $j \in \bar{R}$.

Denote now by $\overline{B}$ the basis just before the last basis B. From the stopping condition $Q = \emptyset$, we conclude that $\overline{Q} = \{l\}$. A simple induction on the number of iterations shows that $\overline{d}_{\overline{R}} = 0$ and $\overline{d}_l > 0$. Let $\overline{\overline{d}}$ be the direction corresponding to the increase of the entering variable x_l, $l \in \overline{Q}$. Then $\overline{d} = \lambda \overline{\overline{d}}$, where $\lambda > 0$. Let y be the boundary point from which $\overline{d}$ exits the feasible region. It is easily concluded that $x = y \geq 0$. Hence, the last basic solution x is feasible. By the well known Theorem of duality we conclude that x is optimal.

Lemma 3 *If the algorithm stops at Step 2, problem (P1) is unbounded.*

Proof From $d = y - x$ we conclude that $Ad = 0$. From $d_R = 0$ we have

$$c^\top d = \sum_{j \in Q} z_j d_j$$

From relation (6) and a simple induction we conclude that $z_j < 0$ for $j \in Q$. As $d_j \geq 0$ we have $c^\top d < 0$. This concludes the proof.

We show finiteness of the algorithm under the assumption that the dual problem of (P1) is nondegenerate, i. e., $z_j \neq 0$, for $j \in N$. We also assume that the initial basic feasible solution x is nondegenerate.

Theorem 4 *The algorithm solves problem (P1) correctly after a finite number of iterations.*

Proof The correctness of the algorithm is an immediate consequence of Lemmas 2 and 3. Let B be the current basis and $\overline{B}$ the previous one. Let $\overline{d}$ correspond to $\overline{B}$ and $\overline{y}$ be the boundary point from which $\overline{d}$ exits the feasible region. Then $d = \overline{y} - x$, where x is the basic solution corresponding to B. Also $\overline{x} = x + td \geq 0$ for $1 \leq t \leq x_{b_r}/(-d_{B_r})$. Indeed, for $t = 1$ we have $\overline{x} = \overline{y} \geq 0$. We conclude that $x_{B_r} > 0$. From the nondegeneracy assumption we have $z_l \neq 0$. Hence, the objective function decreases strictly from iteration to iteration. The decrease is $(-z_l/h_{rl})\, x_{B_r} < 0$. This concludes the proof.

The algorithm cycles on degenerate problems, see [12]. The primal-dual pivoting rule [5] also cycles, see [8]. However, refinements of these algorithms employing the least index rule [4] or the lexicographic rule are cycling free, see [3,12] and [8].

Computational Improvements

The algorithm performs two minimum ratio tests; one for choosing the entering and one for choosing the leaving variable. This fact may mislead someone to con-

clude that no modifications of the algorithm are possible. This is not true. Consider the current basis B. From the previous iteration, a boundary point y, which is used to construct the current direction d, is available. In [17] several heuristics for improving the point y and, hence, reducing the number of iterations, have been developed.

The algorithm is capable of choosing the entering variable x_l so that $l \in R$. Consequently, a minimum ratio test involving all the nonbasic variables is necessary. We can reduce the work per iteration by choosing the entering variable among those belonging to Q. Then, a variant consisting of stages is constructed. The minimum ratio test of this variant is restricted to the set Q. As a result the condition $z_j \geq 0$, $j \in R$ is not satisfied. However, the objective function improvement per iteration is larger.

As long as the basic solutions are exterior the boundary points y must satisfy $y_R = 0$. This restriction does not permit computation of good boundary point. One way to cure this computational deficiency is to force the basic solutions to be feasible to the dual problem of (P1). This way a new primal dual algorithm is constructed. This algorithm can be seen as a generalization of the primal dual pivoting rule discussed in [5].

The pivoting algorithms that generate two paths seem to be more efficient than the classical simplex algorithm. Preliminary computational results reported in [1,7] and [5] support this belief. It is worth mentioning that the algorithm is up to 10 times faster than the simplex algorithm employing the maximum coefficient rule on some specially structured linear problems of rather small size ($n \leq 1000$), see [2]. In all computational results we are aware of the computational superiority of two path pivoting algorithms over the simplex method increases as the size of the problems increases. This fact is very encouraging for the new algorithms. Certainly, more computational results, particularly on benchmark problems, are required to draw more safe conclusions. A computational study of this type is now under way in [16].

See also

- ► Criss-cross Pivoting Rules
- ► Least-index Anticycling Rules
- ► Lexicographic Pivoting Rules
- ► Linear Programming
- ► Principal Pivoting Methods for Linear Complementarity Problems
- ► Probabilistic Analysis of Simplex Algorithms
- ► Simplicial Pivoting Algorithms for Integer Programming

References

1. Achatz H, Kleinschmidt P, Paparrizos K (1991) A dual forest algorithm for the assignment problem. DIMACS 4:1–10
2. Alexouda G, Paparrizos K (1997) A comparative computational study with an exterior point simplex algorithm. Presented at 4th Balkan Conf
3. Anstreicher K, Terlaky T (1994) A monotonic build-up simplex algorithm for linear programming. Oper Res 42:556–561
4. Bland RG (1977) New finite pivoting rules for the simplex method. Math Oper Res 2:103–107
5. Chen H, Pardalos PM, Saunders M (1994) The simplex algorithm with a new primal and dual pivot rule. Oper Res Lett 16:121–127
6. Dantzig GB (1967) Linear programming and extensions. Princeton University Press, Princeton
7. Dosios K, Paparrizos K (1994) A new exterior point algorithm for linear programming problems. YJOR 4:137–148
8. Dosios K, Paparrizos K (1997) Resolution of the problem of degeneracy in a primal and dual simplex algorithm. Oper Res Lett 20:45–50
9. Karmarkar NK (1984) A polynomial-time algorithm for linear programming. Combinatorica 4:373–395
10. Paparrizos K (1988) A non-dual signature method for the assignment problem and a generalization of the dual simplex method for the transportation problem. RAIRO Oper Res 22:269–289
11. Paparrizos K (1991) An infeasible (exterior point) simplex algorithm for assignment problems. Math Program 51:45–54
12. Paparrizos K (1993) An exterior point simplex algorithm for general linear problems. Ann Oper Res 32:497–508
13. Paparrizos K (1996) Exterior point simplex algorithm: Simple and short proof of correctness. In: Proc SYMOPIS'96, pp 13–18
14. Paparrizos K (1996) A new primal and dual pivoting rule for the simplex algorithm. In: Proc SYMOPIS'96, pp 448–453
15. Paparrizos K (1996) A non improving simplex algorithm for transportation problems. RAIRO Oper Res 30:1–15
16. Samaras N (1998) Computational improvements and efficient implementation of two path pivoting algorithms. PhD Diss (in progress)
17. Tsiplidis K (1998) Efficient variants of an exterior point pivoting algorithm. PhD Diss (in progress)

Planning in the Process Industry

JOSEF KALLRATH[1]
[1] GVC/S (Scientific Computing) – B009, BASF AG, Ludwigshafen, Germany
[2] Astronomy Department, University of Florida, Gainesville, USA

Article Outline

Introduction
A Definition of Planning
Special Planning Features in the Process Industry
Some Comments on Planning and Scheduling in the Process Industry
Model Features in Planning Problems
Planning Under Uncertainty
Multi-Criteria Planning Problems
Solution Approaches
Conclusions
References

Introduction

Since there has been tremendous progress in planning in the process industry during the last 25 years, it might be worthwhile to give an overview of the current state-of-the-art of planning problems in the process industry. This is the purpose of the current contribution, which has the following structure. We start with some conceptional thoughts and some comments on special features of planning in the process industry. What is said in this article applies to the chemical but also to the pharmaceutical as well as to the food industry. The reader will find an orientation on production planning, strategic and design planning, planning under uncertainty and multiobjective planning. In the Sect. "Model Features in Planning Problems" the focus is on planning features one would expect in a process industry planning model. The Sect. "Planning Under Uncertainty" and "Multi-Criteria Planning Problems" address planning under uncertainty and multi–criteria planning.

A Definition of Planning

A definition of the term "planning" leads to a group of related terms such as "strategic planning," "design planning," "master planning," "operative planning," and "production planning." Planning needs also be distinguished from scheduling.

A starting point could be Pochet and Wolsey (p. 3 in [21]) in their definition of production planning: *Production planning is defined as the planning of the acquisition of the resources and raw materials, as well as the planning of the production activities, required to transform raw materials into finished products meeting customer demand in the most efficient or economical way.* Note that this does not say anything about the length of the time horizon. Their definition of supply chain planning is similar to that of production planning, but extends its scope by considering and integrating procurement and distribution decisions. They distinguish supply chain design problems which cover a longer time horizon and include additional decisions such as the selection of suppliers, the location of production facilities, and the design of the distribution system.

In this article we use planning for any type of strategic, design, or operative planning. We always assume that we are dealing with multisite production networks. Operative planning includes production planning within multisite production networks and scheduling of individuals sites. While in production planning the focus is rather on optimizing the trade-off between economic objectives such as cost minimization or maximization of contribution and the less tangible objective of customer satisfaction, in scheduling due dates, makespan, or machine utilization becomes more relevant. So, instead of the term "production planning" we use the term "operative planning" (or just, "planning"), with the target of supporting decisions which have an operative impact on a time scale of several months, maybe up to a year. Planning involves the determination of operational plans that support different short- or mid-term objectives for the current business and for a given multisite topology. Planning covers a horizon from a few months to 12 months, and can be extended to cover years (when it comes to strategic or design planning) and time-discrete models are used. If the time horizon becomes smaller we are in the realm of scheduling where time-continuous models become more efficient. When we extend the time horizon we are dealing with strategic planning or design planning covering 1 year up to 20 years. Design planning includes those parts of the Pochet–Wolsey definition given earlier that allow beyond the topology also for the design of

production units, or the capacity of warehouses. Strategic planning is more concerned with product and customer portfolio optimization but also with the acquisition of whole production sites. Kallrath [16] elaborates more on the concept of operative planning and the design planning problem and promotes the idea of combining both in one single model.

Special Planning Features in the Process Industry

In the process industry *continuous* and *batch production systems* can be distinguished. There exists also semibatch production, which combines features from both. Plants producing only a limited number of products each in relatively high volume typically use special purpose equipment allowing a continuous flow of materials in *long campaigns*, i. e., there is a continuous stream of input and output products with no clearly defined start or end time. Alternatively, small quantities of a large number of products are preferably produced using multipurpose equipment which is operated in *batch mode*. Batch production is characterized by *well-defined start-ups*, e. g., filling in some products and follow-up steps defined by specific tasks for heating, mixing, and reaction, and a *clearly defined end* for extracting the finished product. Batch production involves an integer number of batches, where a batch is the smallest quantity to be produced; the batch size may also vary between a lower and an upper bound. Several batches of the same product following each other immediately establish a *campaign*. Production may be subject to certain constraints, e. g., campaigns are built up by a discrete number of batches, or a minimal campaign length (or minimal production quantity) has to be observed. Within a fixed planning horizon, a certain product can be produced in several campaigns; this implies that campaigns have to be modeled as individual entities. One might argue that details of the batch production could be rather found in a scheduling model than in scheduling. However, the model provided in Kallrath and Maindl (Chap. 8 in [17]) is a clear example where batch and campaign features have been incorporated into a time-discrete planning model enhancing it by some continuous time aspects. This problem was solved first by Kallrath [10]. An elegant and numerically more efficient formulation to add time continuity to discrete-time models was developed more recently by Sürie [24]. However, it seems that this formulation still needs to be extended to support multistage production.

Chemical products produced using different production equipment could lead to different performance when used. Therefore, customers might require that a product always is produced using one particular machine, or at least it is always produced using the same machine. This feature is called *origin tracing* and is treated in Kallrath [15]. Certain performance chemicals or goods in the food industry have a limited shelf-life and are subject to an expiration date, or can only be used after a certain aging time. To trace those time stamps requires that individual storage means are considered, e. g., containers or drums, which carry the time stamp or the remaining shelf-life. A model formulation is provided in Kallrath [15].

Another special feature in the refinery or petrochemical industry or process industry in general is the *pooling* problem (see, for instance, [1], or Chap. 11 in [18]). This is an almost classic problem in nonlinear optimization. It is also known as the *fuel mixture problem* in the refinery industry but it also occurs in blending problems in the food industry. The pooling problem refers to the intrinsic nonlinear problem of forcing the same (unknown) fractional composition of multicomponent streams emerging from a pool, e. g., a tank or a splitter in a mass-flow network. Structurally, this problem contains indefinite bilinear terms (products of variables) appearing in equality constraints, e. g., mass balances. The pooling problem occurs in all multicomponent network flow problems in which the conservation of both mass flow and composition is required and both the flow and composition quantities are variable.

Nonlinear programming (NLP) models have been used by the refining, chemical, and other process industries for many years. These nonlinear problems are nonconvex and either approximated by linear ones which can be solved by linear programming (LP) or approximated by a sequence of linear models. This sequential LP technique is well established in the refinery industry but suffers from the drawback of yielding only locally optimum solutions. Although many users may identify obviously suboptimal solutions from experience, there is no validation of nonobvious suboptimal

solutions, as this would require truly globally optimal solutions. Recent advances in optimization algorithms have yielded experimental academic codes which do find globally optimal solutions to large scale pooling NLP models [20]. Nonconvex nonlinear models are not restricted to the oil refining and petrochemical sector, but arise in logistics, network design, energy, environment, and waste management as well as in finance and their solution requires global optimization.

Some Comments on Planning and Scheduling in the Process Industry

Planning and scheduling is part of company-wide logistics and supply chain management. Planning and scheduling are often treated as separate approaches to avoid mathematical complexity. Depending on the level of detail required, the borderlines between planning and scheduling are diffuse. There could be strong overlaps between scheduling and planning in production, distribution, or supply chain management and strategic planning. The main structural elements of planning and scheduling in the process industry are:

- Multipurpose (multi-product, multi-mode) reactors,
- Sequence-dependent set-up times and cleaning cost,
- Combined divergent, convergent, and cyclic material flows,
- Non-preemptive processes (no interruption), buffer times,
- Multistage, batch, and campaign production using shared intermediates,
- Multicomponent flow and nonlinear blending,
- Finite intermediate storage, dedicated, and variable tanks.

Structurally, in scheduling these features often lead to allocation and sequencing problems, knapsack structures, or to the pooling problem. Although the horizon of scheduling problems is usually only days to a few weeks, time-discrete models lead to too many binary variables. Thus, time-continuous formulations are preferable; see Janak et al. [8] or the reviews by Floudas and Lin [4] or Floudas [2]. The largest scheduling problem using a continuous-time approach was solved by Janak et al. [6,7]. It includes over 80 pieces of equipment, considers the processing recipes of hundreds of different products and leads to a linear mixed integer programming (MILP) problem with up to 463,025 constraints, 55,531 variables, among them 8,981 binary variables, and 1,472,365 non-zeroes.

In production or supply chain planning, we usually consider material flow and balance equations connecting sources and sinks of a supply network avoiding some of the complicating details of scheduling. Time-indexed models using a relative coarse discretization of time, e. g., a year, quarters, months, or weeks, are usually accurate enough. LP, MILP and nonlinear mixed integer programming (MINLP) technologies are often appropriate and successful for problems with a clear quantitative objective function as outlined in the section "Model Features in Planning Problems" or quantitative multicriteria objectives. A typical size planning problem with four sites, 800 different products, 1,500 different combinations of product and production plant, 10,000 different combinations of customer, product, package and month is reported in Sect. 5.1 in Kallrath [15]. This problem leads to over 200,000 variables, 380,000 nonzero elements, 400 integer variables, and 900 semi-continuous variables. The number of discrete variables usually can reach a few thousand.

Model Features in Planning Problems

In the literature and in available software packages we usually find *discrete-time models* supporting multi-period analysis, i. e., nearly all the data may vary over time and allow one to evaluate scenarios that involve time-dependent aspects such as seasonal demand patterns, new product introductions, and shutdown of production facilities for maintenance periods. These models include the following main structural objects which are represented by the corresponding indices of the model:

- **Locations** can be production or storage sites, hosting plants and tanks, or demand points hosting tanks.
- **Facilities** are typically production, wrapping, or inventory units that are characterized by their functional properties. Especially, in the process industry we find multistage production systems involving units with general product-mode relationships. Their functional properties are attributes such as capacity, throughput rates, product recipes, yields, minimum production utilization rates, fixed and

variable costs, or storage limitations. Facilities can be existing or potential (for design studies). Production facilities may be subject to batch and campaign constraints across periods.

- **Demand points** may represent customers, regional warehouse locations, or distributors who specify the quantity of a product they request. A demand point can be also seen as a sink of the planning model, i. e., a point where a product leaves the system and is not traced further. Demand may be subject to certain constraints, e. g., satisfying a minimum quantity of demand, observing origins of production, or supplying a customer always from the same origin.
- **Inventories** may be physically fixed entities such as tanks or warehouses but also moveable entities (e. g., drums, containers, boxes). They can be defined as:
  1. Dedicated to a single product from one production source,
  2. Dedicated to a specific product,
  3. Free to accept any product from any source or origin.

  We may encounter tank farms, and especially *multipurpose storage entities*, i. e., variable and multiproduct tanks.
- **Products** may be classified as raw materials, intermediates, finished, and salable products. A product may have several of these attributes, and it can be purchased from suppliers, produced, or sold. Products are produced according to the capabilities at the facilities and the recipes assigned; they may establish a product group, e. g., additives. Product requirements are based on market demand, which is characterized by volume, selling price, package type, time, origin, and location or by other products in which they are used as intermediate products.
- **Suppliers** or **vendors** may provide products for purchase under different offering schemes. This includes the ability to link the product supply to locations and describe contractual pricing mechanisms or availability. The solver may choose the optimal supplier.

Regarding the overall business and strategic objectives the model needs to incorporate data describing the:

- Costs, i. e., certain fixed costs, variable costs (production, transportation, inventory, external product purchase, energy, resources and utilities), and other costs,
- Commercial aspects: financial aspects such as depreciation plans, discount rates, investment plans, foreign currency exchange rates, duties and tariffs, as well as site-dependent taxes,

Maximize operating cash flow and *maximize net present value* (NPV) objective functions are used to determine the financial and operating impacts of mergers, acquisitions, consolidation initiatives, and capital spending programs affecting business. In detail this may include:

1. Maximize the net profit (free design reactors; open and close facilities),
2. Maximize the contribution margin for a fixed system of production units,
3. Maximize the contribution margin while satisfying a minimum percentage of demand,
4. Minimize the cost while satisfying full demand (allow external purchase of products),
5. Maximize total sales neglecting cost,
6. Maximize total production for a fixed system of production reactors,
7. Maximize total production of products for which demand exists,
8. Minimize energy consumption or the usage of other utilities,
9. Minimize the deviation of the usage of resources from their average usage,
10. Multicriteria objectives, e. g., maximize contribution margin and minimize total volume of transport.

Objective functions 2–10 support different short- or mid-term objectives for the current business. By using different objective functions, one can create operational plans that support strategies such as market penetration, top-line growth, or maximization of cash flow to support other business initiatives.

If, besides this broad structure, the focus is on a more detailed representation of physical entities, we find that planning models and their constraints may involve the following features (in alphabetic order):

- **Batch production** (see Kallrath [10]): The quantity of a specific product being produced in a campaign possibly over several periods must be an integer multiple of some pre-defined batch size.
- **Buy, build, close, or sell specific production assets** (see Kallrath [12]): This feature is used for closing, or selling acquisition, consolidation and capacity planning to determine the NPV and operational

impacts of adding or removing specific assets or groups of assets to the network.

- **Campaign production** (see Kallrath [10]): This allows one to impose a lower and/or an upper bound on a contiguous production run (campaign) possibly across periods; this feature is also known under the name *minimal runs*.
- **Delay cost**: Penalty costs apply if customer orders are delivered after the requested delivery date.
- **Minimum production requirements**: Minimum utilization rates modeled as semicontinuous variables have to be observed for specific production units and/or entire production locations for each production time period.
- **Multilocations**: These can be production sites, storage sites, and demand points.
- **Multipurpose production units** (see Kallrath and Wilson [18] or Chap. 8 in Kallrath and Maindl [17]): If a unit is fixed to a certain mode, several products are produced (with different mode-dependent daily production rates), and vice versa, a product can be produced in different modes. Daily production can be less than the capacity rates. A detailed mode-changing production scheme may be used to describe the cost and time required for sequence-dependent mode changes.
- **Multistage production** (see Chap. 8 in Kallrath and Maindl [17]): Free and fixed recipe structures can be used for the production of multiple intermediate products before the production of the final product with convergent and divergent product flows. The recipes may depend on the mode of the multipurpose production unit.
- **Multitime periods** (see Timpe and Kallrath [25]): Nonequidistant time period scales are possible for commercial and production needs. For instance, demand may be forecast weekly for the first quarter of the year and then quarterly for the remainder of the year.
- **Nonlinear pricing** for the purchase of products [12] or utilities (energy, water, etc.) or nonlinear cost for inventory or transportation may lead to convex and concave structures in order to model volume and price discount schemes for the products or services purchased, while, in addition, contract start-up and cancellation fees may lead to additional binary variables.
- **Order lost cost**: Penalty costs are incurred if products are not delivered as requested and promised.
- **Packaging** machines are optimized to increase machine throughput and ensure that priority is given to the most profitable products.
- **Product swaps**: With the objective of saving transportation and other costs companies often arrange joint supply agreements called *swaps*. For example, company 1 based in Europe as well in the USA has a production shortage of product A in the USA and thus purchases a defined quantity of product A in the USA from company 2. Company 2 (also located in the USA and Europe) has a customer in Europe requesting product A and thus purchases a defined quantity of product A from company 1 in Europe. Both companies get product A where they need it and avoid the cost of shipping the product. Without this type of supply agreement company 1 would have to ship product A from its European plant to the USA, and company 2 would have to ship product A from its US manufacturing plant to Europe.
- **Production origin tracing** (see Kallrath [15]): It is possible to define fixed, free, or unique origins for specific demands. For example, a customer may require that his demand is satisfied only from a specific plant in the network, or it may not be supplied from a set of plants, or the customer only requests that he is supplied from one unique plant during the whole planning horizon.
- **Shelf-life** (see Kallrath [15]): Product aging time can be traced. This allows for the application of constraints such as maximum shelf-life, disposal costs for time-expired products, and the setting of selling prices as a function of product life.
- **Transportation** and **logistics** (see Kallrath [13]): Transportation quantities are appropriately modeled by the use of semicontinuous variables. This allows minimum and maximum shipment quantities to be defined for each source location, destination location, product, and transport means combination. The logistics involves the costs, lead times, and constraints (minimum shipment quantities) associated with moving intermediate and finished products between facilities and demand points. The means of transport may be chosen by the optimizer and nonlinear cost functions have to be considered as well.

This list covers many features but may be enhanced depending on the planning problem at hand. A model supporting these features caters to the overall business and strategic objectives. The model incorporates data describing the variable costs for production, transportation, inventory, external product purchase, energy, resources, utilities, and further commercial aspects: financial aspects such as depreciation plans, discount rates, investment plans, foreign currency exchange rates, duties and tariffs, as well as site-dependent taxes. Maximize operating cash flow and maximize NPV objective functions can also be used to determine the financial and operating impacts of mergers, acquisitions, consolidation initiatives, and capital spending programs affecting business. One would expect that a planning model supports various objective functions, among them net profit (free design reactors; open and close facilities), contribution margin, cost, sales, total production, or multicriteria objectives, e. g., maximize contribution margin and minimize total transportation volume.

A possible extension which could relatively easily be connected to such a model is customer or product portfolio features as described in Kallrath [15].

Planning Under Uncertainty

In many instances, the data are not in a deterministic form and this naturally leads to optimization under uncertainty, that is, optimization problems in which at least some of the input data are subject to errors or uncertainties, or in which even some constraints hold only with some probability or are just soft. Those uncertainties can arise from many reasons:

1. Physical or technical parameters which are only known to a certain degree of accuracy. Usually, for such input parameters safe intervals can be specified.
2. Process uncertainties, e. g., stochastic fluctuations in a feed stream to a reactor, processing times.
3. Demand and price uncertainties occur in many situations: supply chain planning, investment planning, or strategic design optimization problems involving uncertain demand and price over a long planning horizon of 10–20 years.

For planning, the third point is most relevant. It is difficult to predict demand and prices, especially in strategic or design planning problems where the time horizon covers several years. Scenario-based optimization in the sense of stochastic optimization leads to large number of variables. A decision taker might be more inclined to hedge against certain risks than to find the most probable scenario. Therefore, the robust optimization framework developed by Lin et al. [9,19] seems to be more appropriate. It provides (1) an explicit trade-off between the effect of uncertainties on the objective function of choice, (2) the unified treatment of uncertainties in product demands, processing times, processing rates, prices of products, and prices of raw materials, and (3) the alternative deterministic equivalent models for a variety of types of representations of uncertainties through bounded, symmetric, normal, difference of normal, binomial, discrete, and Poisson probability distributions.

Multi-Criteria Planning Problems

In planning we may encounter the situation that there are conflicting objectives. Maximizing the contribution margin and minimizing the amount of stocked material might conflict. The novice might think if the storage costs are appropriately included in the objective function both objectives would go along with each other very well. However, some promising sales could be lost because not enough material had been stocked. Thus, the goal to minimize the amount of stock is different from maximizing the contribution margin. At least in this example it might be possible to measure both goals in the same unit of measure, in this case a monetary unit. The more general situation is that we are facing conflicting goals which cannot even be measured on a common scale.

Multiobjective optimization, also called multicriteria optimization or vector minimization problems, allows one to involve several objective functions. A simple approach to solve such problems is to express all objectives in terms of a common measure of goodness leading to the problem how to compare different objectives on a common scale. Basically, one can distinguish two cases. Either the search is for *Pareto optimal* solutions, or the problem has to be *solved for every objective function separately*.

When minimizing several objective functions simultaneously the concept of Pareto optimal solutions turns out to be useful. A solution is said to be Pareto op-

timal if no other solution exists that is at least as good according to every objective, and is strictly better according to at least one objective. When searching for Pareto optimal solutions, the task might be to *find one, find all*, or *cover the extremal set*.

A special solution approach to multiple objective problems is to require that all the objectives should come close to some target, measured each in its own scale. The targets we set for the objectives are called *goals*. Our overall objective can then be regarded as to minimize the overall deviation of our goals from their target levels. The solutions derived are Pareto optimal.

Goal programming can be considered as an extension of standard optimization problems in which targets are specified for a set of constraints. There are two basic approaches for goal programming: the preemptive (*lexicographic*) approach and the *Archimedian* approach. In the Archimedian approach weights or penalties are applied for not achieving targets. A linear combination of the violated targets weighted by some penalty factor is added, or establishes the objective function. We consider only the first approach.

In preemptive goal programming, goals are ordered according to importance and priorities. Especially, if there is a ranking between incommensurate objectives available, this method might be useful. The goal at priority level i is considered to be infinitely more important than the goal at the next lower level, $i + 1$. But they are relaxed by a certain absolute or relative amount when optimizing for the level $i + 1$. In a reactor design problem we might have the following ranking: reactor size ($i = 1$), safety issues ($i = 2$), and production output rate ($i = 3$).

Here we provide an illustrative example for *preemptive* (lexicographic) goal programming with two variables x and y subject to the constraint $42x + 13y \leq 100$ as well as the trivial bounds $x \geq 0$ and $y \geq 0$. We are given

Name	Criterion	Type	A/P	Δ
Goal 1 (OBJ1):	$5x + 2y - 20$	Max	P	10
Goal 2 (OBJ3):	$-3x + 15y - 48$	Min	A	4
Goal 3 (OBJ2):	$1.5x + 21y - 3.8$	Max	P	20

where the attribute A or P indicates whether we have to interpret Δ as an absolute value or percentage-wise. The multi-criteria LP or MILP problem is converted to a sequence of LP or MILP problems. The basic idea is to work down the list of goals according to the priority list given. Thus, we start by maximizing the LP with respect to the first goal. This gives us the objective function value z_1^*. Using this value z_1^* enables us to convert goal 1 into the constraint

$$5x + 2y - 20 \geq Z_1 = z_1^* - \frac{10}{100} z_1^* \,. \tag{1}$$

Note how we have constructed the target Z_1 for this goal (P indicates that we work percentage-wise). In the example we have three goals with the optimization sense {max, min, max}. Two times we apply a percentage-wise relaxation, one time an absolute one. Solving the original problem with the additional inequality (1) we get:

$$\begin{aligned} & z_1^* = -4.615385 \\ & \Rightarrow \quad 5x + 2y - 20 \geq -4.615385 - 0.1 \times (-4.615385) \end{aligned} \tag{2}$$

Now we minimize with respect to goal 2 adding (2) as an additional constraint. We obtain

$$\begin{aligned} & z_2^* = 51.133603 \\ & \Rightarrow \quad -3x + 15y - 48 \geq 51.133603 + 4 \end{aligned} \tag{3}$$

Similarly as for the first goal, we now have to convert the second goal into a constraint (3) (here we allow a deviation of 4) and maximize according to goal 3. Finally, we get $z_3^* = 141.943995$ and the solution $x = 0.238062$ and $y = 6.923186$. To be complete, we could also convert the third goal into a constraint, giving

$$\begin{aligned} 1.5x + 21y - 3.8 &\geq 141.943995 - 0.2 \cdot 141.943995 \\ &= 113.555196 \,. \end{aligned}$$

Note that lexicographic goal programming based on objective functions provides a useful technique to tackle multicriteria optimization problems. The great advantage is that the absolute or percentage-wise deviations used in lexicographic goal programming based on objectives are easy to interpret. However, we have to keep in mind that the sequence of the goals influences the solution strongly. Therefore, the absolute or percentage deviations have to be chosen with care. It is very important that the optimization problem can be solved to exact optimality or at least closely to optimality because

otherwise the interpretation of the permissible deviation from targets becomes difficult if not impossible.

Goal programming offers an alternative approach but should not be regarded as without defects. The specific goal levels selected greatly determine the answer. Therefore, care is needed when selecting the targets. It is also important in which units the targets are measured. Detailed treatment of goal programming appears in such books as those by Ignizio [5] and Romero [22], who introduce many variations on the basic idea, as well as in that by Schniederjans [23].

Solution Approaches

Most of the planning problems in the process industry lead to MILP or MINLP models and contain the following building blocks: *tracing the states of plants*, *modeling production*, *balance equations for material flows*, *transportation terms*, *consumption of utilities*, *cost terms*, and *special model features*. Mode changes, start-up and cancellation features, and nonlinear cost structures require many binary variables. Minimum utilization rates and transportation often require semicontinuous variables. Special features such as batch and campaign constraints across periods require special constraints to implement the concept of contiguity [10,24]. The model, however, remains linear in all variables. Only if the pooling problem occurs, e. g., in the refinery industry or the food industry, we are really facing a MINLP problem. For a review on algorithms used in LP, MILP, NLP, and MINLP the reader is referred to [11]. State-of-the art global solution techniques to nonconvex nonlinear problems were reviewed by Floudas et al. [3].

It is very convenient and saves a lot of maintenance work if the planning model is implemented in an algebraic modeling language. In modeling languages one stores the *knowledge* about a model. A model coded in a modeling language *defines the problem*; it usually does not specify how to solve it. Unlike procedural languages such as Fortran or C, modeling languages are *declarative languages* containing the problem in a declarative form by specifying the properties of the problem. *Algebraic modeling languages* [14] are a special subclass of declarative languages, and most of them are designed for specifying *optimization problems*, i. e., the model can be written in a form which is close to the mathematical notation. Usually they are capable of describing problems of the form

$$\text{Minimize } f(x) \tag{4}$$

$$\text{subject to } g(x) = 0 \tag{5}$$

$$h(x) \geq 0\,, \tag{6}$$

where $\mathbf{x}$ denotes a subset of $X = \mathbb{R}^m \times \mathbb{Z}^n$.

The problem is flattened, i. e., all variables and constraints become essentially one-dimensional, and the model is written in an *index-based* formulation, using *algebraic expressions* in a way which is close to the mathematical notation. Typically, the problem is declared using *sets*, *indices*, *parameters*, and *variables*.

In a modeling language, model and model data are kept separately. There is a clear cut between the model structure and the data. Thus, many different instances of the same model class with varying data can be solved. Many systems provide an open database connectivity (ODBC) interface for automatic database access and an interface to the most widely used spreadsheet systems. This relieves the user from the laborious duty of searching for the relevant data every time the model is used. A second advantage of this concept is that during the development phase of the model (in the cycle) the approach can be tested on *toy problems* with small artificial data sets, and later the model can be applied without change for large scale industry-relevant instances with real data.

In an algebraic modeling language, the formulation of the model is *independent of solver formats*. Different *solvers* can be connected to the modeling language, and the translation of models and data to the solver format is done automatically. This has several advantages. The formerly tedious and error-prone translation steps are done by the computer, and after thorough testing of the interface errors are very unlikely. There is a clean cut between the problem definition and the solution approach, i. e., between the modeling and the numerical, algorithmic part. In addition, for hard problems *different* solvers can be tried, making it more likely that a solution algorithm is found which produces a useful result.

Modern algebraic modeling languages such as `AIMMS`, `GAMS`, `LINGO`, `MPL`, `Mosel`, or `OPL` studio are well suitable to implement such models (see Kallrath [14] to get a flavor of all of them), use state-of-

the art commercial solvers, e. g., XPressMP (by Dash Optimization, http://www.dashoptimization.com) or CPLEX (by ILOG, http://www.ilog.com), and allow one to solve even huge MILP problems with several hundred thousand variables and constraints quite efficiently. In the case of MINLP, the solution efficiency depends strongly on the individual problem and the model formulation. However, as stressed in [11] for both problem types, MILP and MINLP, it is recommended that the full mathematical structure of a problem is exploited, that appropriate reformulations of models are made, and that problem-specific valid inequalities or cuts are used. Software packages may also differ with respect to the ability of *presolving techniques*, *default strategies* for the branch-and-bound algorithm, *cut generation* within the branch-and-cut algorithm, and last but not least *diagnosing and tracing infeasibilities*, which is an important issue in practice.

There is great progress in solving planning problems more efficiently by constructing efficient valid inequalities for certain substructures of planning problems. The well-written books by Wolsey [26] and Pochet and Wolsey [21] contain many examples. These inequalities may a priori be added to a model, and in the extreme case they would describe the complete convex hull. As an example we consider the mixed-integer inequality

$$x \le C\lambda\,, \quad 0 \le x \le X\,; \quad x \in \mathbb{R}_0^+\,, \quad \lambda \in \mathbb{N}\,, \quad (7)$$

which has the valid inequality

$$x \le X - G(K-\lambda)\,, \quad \text{where } K := \left\lceil \frac{X}{C} \right\rceil \quad \text{and}$$
$$G := X - C\,(K-1)\ . \quad (8)$$

This valid inequality (8) is the more useful, the more K and X/C deviate. A special case arising often is the situation $\lambda \in \{0,1\}$. Another example, taken from ([26], p. 129) is

$$A_1\alpha_1 + A_2\alpha_2 \le B + x\,; \quad x \in \mathbb{R}_0^+\,, \quad \alpha_1, \alpha_2 \in \mathbb{N}\,, \quad (9)$$

which for $B \notin \mathbb{N}$ leads to the valid inequality

$$\lfloor A_1 \rfloor \alpha_1 + \left(\lfloor A_2 \rfloor \alpha_2 + \frac{f_2 - f}{1-f} \right) \le \lfloor B \rfloor + \frac{x}{1-f}\,, \quad (10)$$

where the following abbreviations are used:

$$f := B - \lfloor B \rfloor\ ,\ f_1 := A_1 - \lfloor A_1 \rfloor\ ,\ f_2 := A_2 - \lfloor A_2 \rfloor\,. \quad (11)$$

The dynamic counterpart of valid inequalities added a priori to a model leads to cutting plane algorithms which avoid adding a large number of inequalities a priori to the model (note this can be equivalent to finding the complete convex hull). Instead, only those useful in the vicinity of the optimal solution are added dynamically.

With use of these techniques, for some BASF planning problems including up to 100,000 constraints and up to 150,000 variables with several thousand binary variables, good solutions with integrality gaps below 2% have been achieved within 30 min on standard Pentium machines [11].

Conclusions

Planning is strongly based on mathematical optimization exploiting large MILP problems. Strategic, design, and operative planning models including several hundred thousand variables and constraints can be solved efficiently using commercial algebraic modeling languages and attached MILP solvers. These models are connected to company-wide databases.

References

1. Fieldhouse M (1993) The Pooling Problem. In: Ciriani T, Leachman RC (eds) Optimization in Industry: Mathematical Programming and Modeling Techniques in Practice. Wiley, Chichester, pp 223–230
2. Floudas CA (2007) Short-term and Medium-term Scheduling. In: Papageorgiou LG, Georgiadis MC (eds) Supply Chain Optimization. Wiley-VCH, Wiesbaden (in print)
3. Floudas CA, Akrotiriankis IG, Caratzoulas S, Meyer CA, Kallrath J (2004) Global Optimization in the 21st Century: Advances and Challenges for Problems with Nonlinear Dynamics. In: Barbossa-Povoa A, Motos A (eds) European Symposium on Computer-Aided Process Engineering (ESCAPE) 14, Elsevier, Dordrecht, pp 23–51
4. Floudas CA, Lin X (2004) Continuous-Time versus Discrete-Time Approaches for Scheduling of Chemical Processes: A Review. Comp Chem Eng 28(11):2109–2129
5. Ignizio JP (1976) Goal Programming and Extensions. Heath, Lexington, Massachusetts
6. Janak SL, Floudas CA, Kallrath J, Vormbrock N (2006) Production Scheduling of a Large-Scale Industrial Batch Plant: I. Short-Term and Medium-Term Scheduling. Ind Eng Chem Res 45:8234–8252
7. Janak SL, Floudas CA, Kallrath J, Vormbrock N (2006) Production Scheduling of a Large-Scale Industrial Batch Plant: II. Reactive Scheduling. Ind Eng Chem Res 45:8253–8269

8. Janak SL, Lin X, Floudas CA (2004) Enhanced Continuous-Time Unit-Specific Event-Based Formulation for Short-Term Scheduling of Multipurpose Batch Processes: Resource Constraints and Mixed Storage Policies. Ind Chem Eng Res 43:2516–2533
9. Janak SL, Lin X, Floudas CA (2007) A New Robust Optimization Approach for Scheduling under Uncertainty - II. Uncertainty with Known Probability Distribution. Comp Chem Eng 31:171–195
10. Kallrath J (1999) The Concept of Contiguity in Models Based on Time-Indexed Formulations. In: Keil F, Mackens W, Voss H, Werther J (eds) Scientific Computing in Chemical Engineering II. Springer, Berlin, pp 330–337
11. Kallrath J (2000) Mixed Integer Optimization in the Chemical Process Industry: Experience, Potential and Future Perspectives. Chem Eng Res Des 78(6):809–822
12. Kallrath J (2002) Combined Strategic and Operational Planning – An MILP Success Story in Chemical Industry. OR Spectrum 24(3):315–341
13. Kallrath J (2002) Planning and Scheduling in the Process Industry. OR Spectrum 24(3):219–250
14. Kallrath J (2004) Modeling Languages in Mathematical Optimization. Kluwer, Norwell, MA
15. Kallrath J (2005) Solving Planning and Design Problems in the Process Industry Using Mixed Integer and Global Optimization. Annals Oper Res 140:339–373
16. Kallrath J (2007) Multi-Site Design and Planning Problems in the Process Industry. In: Papageorgiou LG, Georgiadis MC (eds) Supply Chain Optimization. Wiley-VCH, Wiesbaden (in print)
17. Kallrath J, Maindl TI (2006) Real Optimization with SAP-APO. Springer, Heidelberg
18. Kallrath J, Wilson JM (1997) Business Optimisation Using Mathematical Programming. Macmillan, Houndmills, Basingstoke
19. Lin X, Janak SL, Floudas CA (2004) A New Robust Optimization Approach for Scheduling under Uncertainty – I. Bounded Uncertainty. Comp Chem Eng 28:1069–1085
20. Meyer C, Floudas CA (2006) Global Optimization of a Combinatorially Complex Generalized Pooling Problem. AIChE J 52:1027–1037
21. Pochet Y, Wolsey LA (2006) Production Planning by Mixed Integer Programming. Springer, New York
22. Romero C (1991) Handbook of Critical Issues in Goal Programming. Pergamon Press, Oxford
23. Schniederjans MJ (1995) Goal Programming: Methodology and Applications. Kluwer, Boston
24. Suerie C (2005) Time Continuity in Discrete Time Models – New Approaches for Production Planning in Process Industries, vol 552 of Lecture Notes in Economics and Mathematical Systems. Springer, Heidelberg
25. Timpe C, Kallrath J (2000) Optimal Planning in Large Multi-Site Production Networks. Eur J Oper Res 126(2):422–435
26. Wolsey LA (1998) Integer Programming. Wiley, New York

Plant Layout Problems and Optimization

Lazaros G. Papageorgiou
Centre for Process Systems Engineering,
Department of Chemical Engineering,
UCL (University College London), London, UK

Article Outline

Introduction
Approaches to Process Plant Layout
A Continuous-Domain Mathematical Model
Integration Aspects
Use of Land
Safety
Location Information
Design and Operation
Layout Organization into Production Sections
Pipe Routing
Other Applications
Allocation
Data Classification
Concluding Remarks
References

Introduction

Increased competition leads contractors and chemical companies to look for potential savings at every stage of the design process. Process plant layout is an important part of the design or retrofit of chemical plants, and involves decisions concerning the spatial allocation of equipment items and the required connections between them [21]. Equipment items are allocated to one floor (single-floor case) or many floors (multifloor case) considering a number of cost and management or engineering drivers such as:

- Connectivity cost, which involves the cost of piping and other required connections between equipment items. In addition, other related network operating costs such as pumping may also be taken into account.
- Construction cost, which leads to the design of compact plants. The trade-off between the cost of occu-

pied area (land) and height (multifloor plants) must also be considered here.
- Retrofit, i. e. fitting new equipment items within an existing plant layout.
- Operation, which involves scheduling issues (e. g. pipeless plants).
- Safety, which introduces constraints with respect to the minimum allowable distance between equipment items.

Trade-offs between connectivity, pumping, construction, financial risk and installation of potential protection devices are necessary. In order to resolve these trade-offs optimally, computer-aided methods are needed to support engineers in the identification of optimal layouts subject to multiple design criteria.

Traditionally, process plant layout decisions are either ignored or do not receive appropriate attention during the design or retrofit of chemical plants. Furthermore, safety aspects are usually not considered systematically within process layout, design and operation frameworks, thus resulting in inefficient and unsafe plants. It is critical that systematic and efficient computer-aided methods are developed to support engineers in the rapid generation of alternative, safe, chemical process plant layouts.

The process plant layout problem shares many similarities with the facility layout problem, which has been the center of interest for industrial engineers for several years (for comprehensive reviews, see [14,18,22,36]). Here, we focus on the chemical process plant layout, which has attracted attention within the research community because of the particular production, environmental and safety considerations of the process industries. Such considerations affect both the objective functions and the constraints of any optimization model used.

In this chapter, a brief review of chemical plant layout research is given for the process plant layout problem and this is followed by a detailed description of an optimization-based framework. Illustrative research is presented by integrating traditional plant layout frameworks with sustainability aspects. Special attention is given to aspects associated with land use, safety and plant location. Issues related to design/operation, production organization and pipe routing are also discussed. Finally, other applications based on plant layout principles are presented.

Approaches to Process Plant Layout

The initial approaches to the process plant layout problem were based on heuristics [2,35]. Although heuristic approaches may be efficient from the computational point of view, they do not offer any guarantee for the optimality of the solution obtained. Graph-theoretic approaches have been applied to the problem of organizing equipment items into sections for single-floor plants [15]. A method to aid decisions concerning the assignment of equipment items to floors in multifloor arrangements was proposed [16], albeit with no consideration of the detailed layout within each floor. Application of stochastic optimization techniques to the process plant layout problem was demonstrated [6,7] by developing software tools capable of tackling larger problems.

Recently, a number of mathematical programming approaches have emerged based either on land space discretization [11,12,34] or continuous-domain representation [3,9,13,23,24,30]. Most of these models accommodate various important issues of the layout problem, such as rectangular equipment footprints, rectilinear distances, equipment orientation, restrictions on available space, layout organization into production sections and some safety aspects (e. g. minimum distances between equipment items).

The process plant layout problem is well known as a hard computational problem. Most literature approaches, which are based on mathematical programming and use branch-and-bound solution procedures, can usually tackle flowsheets with up to 12 equipment items. It has been demonstrated that the solution performance can greatly be improved with additional simple constraints to eliminate symmetrical layout solutions and/or tighten mathematical formulation [8,9,33,37]. Furthermore, iterative optimization-based procedures have recently been developed using construction and/or improvement solution phases, which are suitable for larger flowsheets [17,28,40].

Next, a representative mathematical programming framework is described, based on a continuous-domain representation.

A Continuous-Domain Mathematical Model

A mathematical programming model for the optimal single-floor process plant layout problem in a two-

Plant Layout Problems and Optimization, Table 1
Notation

Indices	
i,j	Equipment items
Parameters	
α_i,β_i	Dimensions of item i
C_{ij}	Connection cost between items i and j
X^{max}	Maximum x coordinate
Y^{max}	Maximum y coordinate
Decision variables	
O_i	1 if length of item i (parallel to the x axis) is equal to α_i; 0 otherwise
$E1_{ij}$, $E2_{ij}$	Nonoverlapping binary variables
l_i	Length of item i
d_i	Depth of item i
X_i,Y_i	Coordinates of the geometrical center of item i
R_{ij}	Relative distance in x coordinates between items i and j, if i is to the right of j
L_{ij}	Relative distance in x coordinates between items i and j, if i is to the left of j
A_{ij}	Relative distance in y coordinates between items i and j, if i is above j
B_{ij}	Relative distance in y coordinates between items i and j, if i is below j
D_{ij}	Total rectilinear distance between items i and j

dimensional continuous space is described here as proposed previously by Papageorgiou and Rotstein [24]. In the formulation presented here, rectangular shapes are assumed for equipment items following current industrial practice. Rectilinear distances between the equipment items are used for a more realistic estimate of piping costs as either aerial or underground corridors are usually used for the piping and instrumentation network. Equipment items, which are allowed to rotate 90°, are assumed to be connected through their geometrical centers.

Overall, the single-floor process plant layout problem can be stated as follows:

Given

- A set of N pieces of equipment and their dimensions
- Connectivity network and associated costs
- Space and equipment allocation limitations
- Minimum safety distances, if any, between equipment items
- Production sections

Determine

- The allocation of each equipment item (i. e. coordinates and orientation)

So as to optimize a suitable criterion (here, minimize equipment connection cost).

The indices, parameters and decision variables (binary and continuous) associated with the mathematical model are listed in Table 1.

Next, the main mathematical constraints of the single-floor process plant layout model are described.

Equipment Orientation Constraints The values of length, l_i, and depth, d_i, of equipment item i depend on its orientation in the space and can be determined by

$$l_i = \alpha_i O_i + \beta_i (1 - O_i) \quad \forall i \,, \tag{1}$$

$$d_i = \alpha_i + \beta_i - l_i \quad \forall i \,. \tag{2}$$

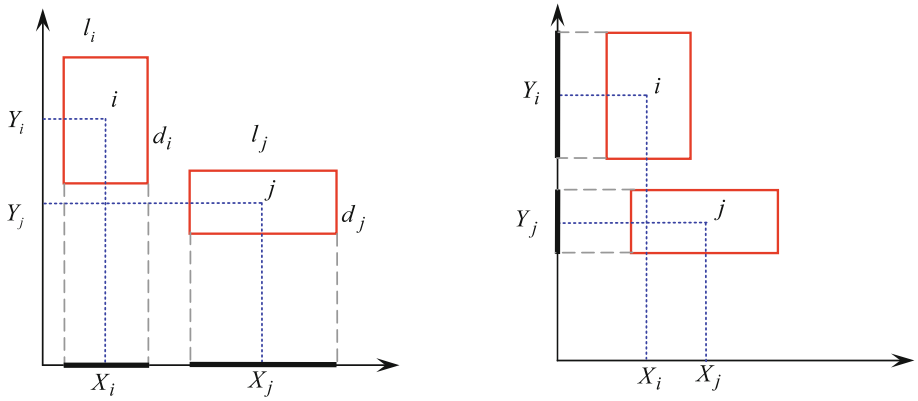
Distance Constraints The relative distances between items i and j are given by the following equalities:

$$R_{ij} - L_{ij} = X_i - X_j \quad \forall (i,j) \colon C_{ij} > 0 \,, \tag{3}$$

$$A_{ij} - B_{ij} = Y_i - Y_j \quad \forall (i,j) \colon C_{ij} > 0 \,, \tag{4}$$

$$D_{ij} = R_{ij} + L_{ij} + A_{ij} + B_{ij} \quad \forall (i,j) : C_{ij} > 0 \,. \tag{5}$$

It should be noted that the above constraints (3–5) are written only for those equipment pairs whose relative distance occurs in the objective function (i. e. terms



Plant Layout Problems and Optimization, Figure 1
Equipment nonoverlapping

with nonzero connection costs, $C_{ij} > 0$), which is minimized. As a consequence, it is guaranteed that *at most* one variable of each pair (R_{ij},L_{ij}) and (A_{ij},B_{ij}) will be nonzero at the optimal solution of each linear programming problem during the branch-and-bound solution procedure.

Nonoverlapping Constraints In order to avoid the overlapping of the two equipment items i and j occupying the same physical location, appropriate mathematical constraints should be included in the model to prohibit overlapping of the projections of each equipment footprint either in the x or y in the dimension as clearly shown in Fig. 1

Nonoverlapping is guaranteed if *at least* one of the following conditions is active:

$$X_i - X_j \geq \frac{(l_i + l_j)}{2} \quad \forall i = 1, \ldots, N-1,\ j = i+1, \ldots, N, \tag{6}$$

$$X_j - X_i \geq \frac{(l_i + l_j)}{2} \quad \forall i = 1, \ldots, N-1,\ j = i+1, \ldots, N, \tag{7}$$

$$Y_i - Y_j \geq \frac{(d_i + d_j)}{2} \quad \forall i = 1, \ldots, N-1,\ j = i+1, \ldots, N, \tag{8}$$

$$Y_j - Y_i \geq \frac{(d_i + d_j)}{2} \quad \forall i = 1, \ldots, N-1,\ j = i+1, \ldots, N. \tag{9}$$

These nonoverlapping disjunctive conditions can mathematically be modelled by including appropriate "big M" constraints and introducing two additional sets of binary variables; $E1_{ij}$ and $E2_{ij}$. Each pair of values (0 or 1) for these variables determines which constraint from (6) to (9) is active. For every pair (i,j) such that $i < j$, we have:

If constraint (6) is active, then $E1_{ij} = 0, E2_{ij} = 0$.
If constraint (7) is active, then $E1_{ij} = 1, E2_{ij} = 0$.
If constraint (8) is active, then $E1_{ij} = 0, E2_{ij} = 1$.
If constraint (9) is active, then $E1_{ij} = 1, E2_{ij} = 1$.

In summary, the nonoverlapping constraints included in the mathematical model are:

$$X_i - X_j + M(E1_{ij} + E2_{ij}) \geq \frac{(l_i + l_j)}{2} \quad \forall i = 1, \ldots, N-1,\ j = i+1, \ldots, N \tag{10}$$

$$X_j - X_i + M(1 - E1_{ij} + E2_{ij}) \geq \frac{(l_i + l_j)}{2} \quad \forall i = 1, \ldots, N-1,\ j = i+1, \ldots, N \tag{11}$$

$$Y_i - Y_j + M(1 + E1_{ij} - E2_{ij}) \geq \frac{(d_i + d_j)}{2} \quad \forall i = 1, \ldots, N-1,\ j = i+1, \ldots, N \tag{12}$$

$$Y_j - Y_i + M(2 - E1_{ij} - E2_{ij}) \geq \frac{(d_i + d_j)}{2} \quad \forall i = 1, \ldots, N-1,\ j = i+1, \ldots, N, \tag{13}$$

where M is a suitable upper bound on the distance between two equipment items. Note that the above constraints can easily be modified to include minimum/maximum distances between specific items

for safety and/or operational reasons (see, for example, [24]). More systematic frameworks to account for safety aspects are described in Sect. "Other Applications".

Additional Layout Design Constraints Lower-bound constraints on the coordinates of the geometrical center of each equipment item should be included in order to avoid intersection of items with the origin of axes, achieved by

$$X_i \geq \frac{l_i}{2} \quad \forall i\,, \tag{14}$$

$$Y_i \geq \frac{d_i}{2} \quad \forall i\,. \tag{15}$$

Similarly, upper-bound constraints can be used to force all equipment items to be allocated within a rectangular shape of land area defined by the corners (0,0) and ($X^{\max}$,$Y^{\max}$):

$$X_i + \frac{l_i}{2} \leq X^{\max} \quad \forall i\,, \tag{16}$$

$$Y_i + \frac{d_i}{2} \leq Y^{\max} \quad \forall i\,. \tag{17}$$

Objective Function The objective function considered in this model is the minimization of the total connection cost:

$$\text{Min} \sum_i \sum_{j \neq i} C_{ij} D_{ij}\,. \tag{18}$$

Other cost terms can be considered, such as land area, construction/building, vertical/horizontal pumping, protection devices and accident property damage.

All continuous variables in the formulation are defined as nonnegative. Overall, the single-floor process plant layout problem has been formulated as a mixed-integer linear programming (MILP) model (constraints 1–5, 10–18). This model has also been extended for multifloor process plant layout problems [26], combined with safety aspects [25] and integrated with design/operation of pipeless batch plants [29].

Integration Aspects

In this section, additional important aspects are discussed with respect to their integration into traditional plant layout frameworks.

Use of Land

The cost of land is an important component that should be included in the mathematical model of the process pant layout problem. Land cost has been implicitly taken into account [30] where the surface area has been assumed to be proportional to the land occupied by equipment items and not the space around items. A stochastic optimization technique [7] includes explicitly the cost of land in the objective function. Both previous efforts [7,30] refer to single-floor layouts problems.

For the multifloor plant layout case, the trade-off between the cost of occupied area (land) and that of construction (multiple floors) should be considered and resolved in an optimal manner. Mathematical programming models that have crucially addressed such issues have been developed by Georgiadis et al. [12] and Patsiatzis and Papageorgiou [26].

Sole consideration of land cost (and possibly together with construction, connection and pumping costs) may result in quite a compact layout and consequently a conservative layout solution from a safety point of view. Therefore, sufficient integration of safety aspects into existing frameworks is necessary. Such representative research efforts are described next.

Safety

Safety aspects should be considered during the early stages of the design process by using appropriate quantitative indices. Such aspects are either ignored or considered through rather simplistic terms (e. g. minimum distances [3,24]). The safety of a chemical plant can further be improved by either putting the plant equipment items far apart from each other or by installing protection devices at additional capital cost, which would reduce the area of exposure and consequently reduce the propagation of a potential accident. Various trade-offs should be considered simultaneously within the process plant layout, such as:

- Piping, pumping and land costs that increase as the distance between equipment items increases;
- Financial risk component cost, which decreases as equipment items are put far apart or by installing extra protection devices; and
- Cost of protection devices, which eliminate accident escalation.

Along the above directions, a mathematical programming model-based approach was introduced by Penteado and Ciric [30], which resulted in mixed-integer nonlinear programming (MINLP) models determining simultaneously the process plant layout, the number and the type of protection devices and the financial risk associated with accidents and their propagation to neighboring units, assuming circular equipment footprints. An alternative MINLP model has also been proposed [27] by adopting rectangular shapes and rectilinear distances between units. Both these studies were based on a risk representation related to accidents propagating from a source to a target unit utilizing the equivalent TNT method [19]. An evolutionary procedure was proposed by Fuchino et al. [10] for the arrangement of equipment models in the early design phase by considering simultaneously the position and the orientation of equipment modules, as both affect the propagation of an accident.

Two widely used indices for risk assessment and safety evaluation are the Mond Index [20] and the Dow Fire and Explosion Index [1]. The use of the Mond Index was illustrated by Castell et al. [7] by adding a penalty term in the objective function, which minimizes any violations of Mond safety distances. User intervention is crucial in this work to ensure the feasibility of all separation distance requirements. Finally, the use of the Dow Fire and Explosion Index was demonstrated by Patsiatzis et al. [25] by proposing an MILP approach to safe process plant layout. This index determines the realistic maximum loss occurring under the most adverse operating conditions and is applicable to processes where flammable, combustible or reactive material is stored or processed. It is based on historic loss data, the energy potential of the processed materials in the chemical plants and the current application of loss-prevention practices.

Location Information

Integration of location-specific information with the decision-making procedure for the process plant layout problem was successfully demonstrated by Ozyurt and Realff [23]. Such information is related to existing infrastructure, geographic aspects, climate conditions, elevation and soil characteristics. The location-specific information system is first used to derive conclusions about the plant environment, and these are then translated into additional mathematical constraints on the process plant layout problem by restricting some equipment locations.

Design and Operation

The process plant layout decisions are traditionally considered after the plant design stage has been completed. However, it is obvious that strong interactions exist between plant layout and plant design and therefore significant benefits could be gained by developing systematic, simultaneous approaches. Such an approach was proposed by Barbosa-Povoa et al. [4], where an MILP model was shown to be particularly suitable for cases with equipment of rectangular and irregular shapes as well as flexible equipment input/output point locations. The latter model has been integrated with operating aspects and applied to multipurpose batch plants [5].

It should be noted that layout considerations are of particular significance for pipeless plants as they determine the vessel transfer times, which can then affect the schedule of the plant. Simultaneous optimization-based approaches have also been developed for pipeless batch pants, which offer enhanced production flexibility and piping-free cleaning requirements, by considering plant design, layout and operation aspects [29,31].

Layout Organization into Production Sections

The organization of the plant layout problem into well-defined production sections is often required for various reasons, such as safety, efficient material handling and workforce management. The boundaries of these sections are drawn by walls or corridors, which facilitate the movement of materials and/or operators.

The organization of the plant layout into sections can formally be incorporated into the process plant layout mathematical model presented in Sect. "A Continuous-Domain Mathematical Model". The basic assumption is that the equipment items are partitioned into subsets by using either rule-based/intuitive techniques or algorithmic approaches (see, for example, [15]). Each subset of equipment items constitutes *a section*, and then each section can then be represented by a well-defined rectangular box. New constraints must be introduced to guarantee that no overlapping occurs

(1) among different sections and (2) among the footprints *within* each section [24]. The aim of the optimization is then to determine the location of both production sections and equipment items within each section so that the total layout cost is minimized.

Pipe Routing

The design of the piping system in a chemical plant constitutes an important part of deriving an efficient plant layout (apart from pipeless plants) and should be considered as early as possible in the design process. A number of factors should be taken into account, such as cost, safety, operability and structural/mechanical aspects. A pipe-routing method was proposed by Schmidt-Traub et al. [32] using grid and vector router algorithms. Optimization-based and graph theory methods have also been described by Guirardello and Swaney [13], which could include capacity constraints, pipes with branches and stress analysis.

Other Applications

This section demonstrates the application of basic plant layout principles to two interesting problems: allocation and data classification.

Allocation

Allocation problems, which include plant layout problems, with different numbers of dimensions have been research topics of great activity for many years. Most allocation problems are large-scale combinatorial optimization problems, occurring in several different industrial applications. A general purpose MILP model for allocation problems in any given number of dimensions has been presented [38]. This mathematical formulation utilizes a type of nonoverlapping constraints similar to those presented in Sect. "A Continuous-Domain Mathematical Model". The proposed model deals with the allocation of items in an N-dimensional space. Several problems, previously presented in the literature, have been solved using the proposed model, such as one-dimensional scheduling problems, two-dimensional cutting problems, as well as plant layout problems and three-dimensional packing problems. Problems defined in four dimensions are also presented and solved using the model described above.

Data Classification

Beyond chemical plant design, basic plant layout principles can also be applied to data classification problems as illustrated here. Data classification, a fundamental problem in data mining and machine learning, deals with the identification of patterns and the assignment of new samples into known groups. A rigorous mixed-integer optimization model for multiclass data classification problems has been proposed [39] using a hyperbox representation. The optimal location and dimension of each box is determined by minimizing the total number of misclassifications. Special constraints are introduced to avoid overlapping of boxes that belong to different classes. These constraints simply represent an extension of those described in Sect. "A Continuous–Domain Mathematical Model" to cover more than two dimensions.

Concluding Remarks

In this chapter, a comprehensive review of the chemical plant layout research has been presented. Systematic, optimization-based frameworks for plant layout have the potential to offer contractors and chemical companies significant savings and increase business competitiveness. The future will show additional benefits if issues related to dynamic plant layout, uncertainty and more efficient solution procedures for larger problems are resolved.

References

1. American Institute of Chemical Engineers (1994) Dow's Fire and Explosion Index hazard Classification Guide. New York
2. Amorese L, Cena V, Mustacchi C (1977) Heuristic for Compact Location of Process Components. Chem Eng Sci 32(2):119–124
3. Barbosa-Povoa AP, Mateus R, Novais AQ (2001) Optimal two-dimensional layout of industrial facilities. Int J Prod Res 39(12):2567–2593
4. Barbosa-Povoa AP, Mateus R, Novais AQ (2002) Optimal design and layout of industrial facilities: A simultaneous approach. Indust Eng Chem Res 41(15):3601–3609
5. Barbosa-Povoa AP, Mateus R, Novais AQ (2002) Optimal design and layout of industrial facilities: An application to multipurpose batch plants. Indust Eng Chem Res 41(15):3610–3620

6. Burdorf A, Kampczyk B, Lederhose M, Schmidt-Traub H (2004) CAPD – computer-aided plant design. Comput Chem Eng 28(1–2):73–81
7. Castell CML, Lakshmanan R, Skilling JM, Banares-Alcantara R (1998) Optimisation of process plant layout using genetic algorithms. Comput Chem Eng 22:S993-S996
8. Castillo I, Westerlund T (2005) An epsilon-accurate model for optimal unequal-area block layout design. Comput Oper Res 32(3):429–447
9. Castillo I, Westerlund J, Emet S, Westerlund T (2005) Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods. Comput Chem Eng 30(1):54–69
10. Fuchino T, Itoh T, Muraki M (1997) Arrangement of process equipment modules with consideration of plant safety. J Chem Eng Jap 30(5):896–901
11. Georgiadis MC, Rotstein GE, Macchietto S (1997) Optimal layout design in multipurpose batch plants. Indust Eng Chem Res 36(11):4852–4863
12. Georgiadis MC, Schilling G, Rotstein GE, Macchietto S (1999) A general mathematical programming approach for process plant layout. Comput Chem Eng 23(7):823–840
13. Guirardello R, Swaney RE (2005) Optimization of process plant layout with pipe routing. Comput Chem Eng 30(1):99–114
14. Heragu SS (2006) Facilities Design. iUniverse, Lincoln
15. Jayakumar S, Reklaitis GV (1994) Chemical-Plant Layout Via Graph Partitioning. 1. Single Level. Comput Chem Eng 18(5):441–458
16. Jayakumar S, Reklaitis GV (1996) Chemical plant layout via graph partitioning. 2. Multiple levels. Comput Chem Eng 20(5):563–578
17. Jernstrom P, Westerlund J, Papageorgiou LG (2004) An Item-based Decomposition Approach to Process Plant Layout Problems. In: Hamza MH (ed) 4th IASTED International conference on modelling, simulation, and optimization. ACTA Press, Calgary, pp 37–42
18. Kusiak A, Heragu SS (1987) The Facility Layout Problem. Eur J Oper Res 29(3):229–251
19. Lees FP (1996) Loss prevention in the process industries. Butterworth-Heinmann, London
20. Lewis DJ (1980) The Mond fire, explosion and toxicity index applied to plant layout and spacing. in AIChE Loss Prevention Symposium.
21. Mecklenburgh JC (1985) Process plant layout. Institution of Chemical Engineers, London
22. Meller RD, Narayanan V, Vance PH (1998) Optimal facility layout design. Oper Res Lett 23(3–5):117–127
23. Ozyurt DB, Realff MF (1999) Geographic and process information for chemical plant layout problems. AIChE J 45(10):2161–2174
24. Papageorgiou LG, Rotstein GE (1998) Continuous-domain mathematical models for optimal process plant layout. Indust Eng Chem Res 37(9):3631–3639
25. Patsiatzis DI, Knight G, Papageorgiou LG (2004) An MILP approach to safe process plant layout. Chem Eng Res Design 82(A5):579–586
26. Patsiatzis DI, Papageorgiou LG (2002) Optimal multi-floor process plant layout. Comput Chem Eng 26(4–5):575–583
27. Patsiatzis DI, Papageorgiou LG (2002) Safe Process Plant Layout using Mathematical Programming. In: Grievink J, von Schijndel J (eds) European symposium on computer aided process engineering-12. Elsevier Science BV, Amsterdam, p 295–300
28. Patsiatzis DI, Papageorgiou LG (2003) Efficient solution approaches for the multifloor process plant layout problem. Indust Eng Chem Res 42(4):811–824
29. Patsiatzis DI, Xu G, Papageorgiou LG (2005) Layout aspects of pipeless batch plants. Indust Eng Chem Res 44(15):5672–5679
30. Penteado FD, Ciric AR (1996) An MINLP approach for safe process plant layout. Indust Eng Chem Res 35(4):1354–1361
31. Realff MJ, Shah N, Pantelides CC (1996) Simultaneous design, layout and scheduling of pipeless batch plants. Comput Chem Eng 20(6–7):869–883
32. Schmidt-Traub H, Koster M, Holtkotter T, Nipper N (1998) Conceptual plant layout. Comput Chem Eng 22:S499–S504
33. Sherali HD, Fraticelli BMP, Meller RD (2003) Enhanced model formulations for optimal facility layout. Oper Res 51(4):629–644
34. Suzuki A, Fuchino T, Muraki M (1991) Equipment Arrangement for Batch Plants in Multifloor Buildings with Integer Programming. J Chem Eng Jap 24(6):737–742
35. Suzuki A, Fuchino T, Muraki M, Hayakawa T (1991) An Evolutionary Method of Arranging the Plot Plan for Process Plant Layout. J Chem Eng Jap 24(2):226–231
36. Topkins JA, White JA, Bozer YA, Frazelle EH, Tanchoco JMA, Trevino J (1996) Facilities Planning, 2nd edn. Wiley, New York
37. Westerlund J, Papageorgiou LG (2004) Improved Performance in Process Plant Layout Problems using Symmetry-breaking Constraints. In: Floudas CA, Agrawal R (ed) Foundations of Computer-Aided Process Design. OmniPress, Madison, pp 485–488
38. Westerlund J, Papageorgiou LG, Westerlund T (2007) A MILP Model for N-dimensional Allocation. Comput Chem Eng 31(12):1702–1714
39. Xu G, Papageorgiou LG (2006) An MILP Model for Multiclass Data Classification. In: Bogle IDL, Zilinskas J (eds) Computer Aided Methods in Optimal Design and Operation, World Scientific's book series on Computers and Operations Research. World Scientific Publishing, Singapore, pp 15–20
40. Xu G, Papageorgiou LG (2007) A construction-based approach to process plant layout using mixed-integer optimization. Indust Eng Chem Res 46(1):351–358

Pontryagin Maximum Principle

VICTOR KOROTKICH
Central Queensland University, Mackay, Australia

MSC2000: 49J15, 49K15, 93C10

Article Outline

Keywords
See also
References

Keywords

Phase space; Optimal control; Hamiltonian system; Optimality conditions

It is considered that the behavior of an object at any instant of time is characterized by n real numbers $x^1, \ldots, x^n$. The vector space X of the vector variable $x = (x^1, \ldots, x^n)$ is the phase space of the object under consideration. The motion of the object consists of the fact that the variables $x^1, \ldots, x^n$ change with time. It is assumed that the object's motion can be controlled, i. e., that the object is equipped with certain controllers on whose position the motion of the object depends. The positions of the controllers are characterized by points $u = (u^1, \ldots, u^r)$ of a certain control region U, which may be any set in some r-dimensional Euclidean space. In applications, the case where U is a closed region in the space is especially important.

In the statement of the problem it is assumed that the object's law of motion can be written in the form of a system of differential equations

$$\frac{dx^i}{dt} = f_i(x, u), \quad i = 1, \ldots, n\,, \tag{1}$$

or in vector from

$$\frac{dx}{dt} = f(x, u)\,, \tag{2}$$

where the functions f_i are defined for $x \in X$ and $u \in U$. They are supposed to be continuous in the variables $x^1, \ldots, x^n, u$ and continuously differentiable with respect to $x^1, \ldots, x^n$.

If the control law is given, i. e., a certain admissible control $u = u(t)$ is chosen, (2) takes the form

$$\frac{dx}{dt} = f(x, u(t))\,, \tag{3}$$

from which, for any initial condition $x(t_0) = x_0$, the motion of the object $x = x(t)$ is uniquely determined, i. e., the solution of (3) is defined for a certain time interval. The solution $x(t)$ is called the solution of (2) corresponding to the control $u(t)$ for the initial condition $x(t_0) = x_0$. The solution may not be defined on the entire interval $t_0 \le t \le t_1$ on which $u(t)$ is given as it may run off to infinity.

It is said that the admissible control $u(t)$, $t_0 \le t \le t_1$ transfers the phase point from the position x_0 to the position x_1 if the corresponding solution $x(t)$ of (2), satisfying the initial condition $x(t_0) = x_0$, is defined for all t, $t_0 \le t \le t_1$, and passes through the point x_1 at the time t_1.

It is supposed that an additional function $f_0(x, u)$, which is defined and continuous together with its partial derivatives $\partial f_0 / \partial x^i$, $i = 1, \ldots, n$, on all of $X \times U$, is given. The fundamental problem of finding the optimal controls is formulated as follows [1].

In the phase space X two points x_0 and x_1 are given. Among all the admissible controls $u = u(t)$ which transfer the phase point from the position x_0 to the position x_1 (if such controls exist), find one for which the functional

$$J = \int_{t_0}^{t_1} f_0(x(t), u(t))\, dt \tag{4}$$

takes on the least possible value. Here $x(t)$ is the solution of (2) with initial condition $x(t_0) = x_0$ corresponding to the control $u(t)$ and t_1 is the time at which this solution passes through x_1. The control $u(t)$ which yields the solution of the problem is called an *optimal control* corresponding to a transition from x_0 to x_1. The corresponding trajectory $x(t)$ is called an *optimal trajectory*.

An important special case of the fundamental problem is the one where $f_0(x, u) \equiv 1$. In this case, the functional (4) takes the form

$$J = t_1 - t_0$$

and the optimality of the control $u(t)$ signifies minimality of the transition time from x_0 to x_1. The problem of

finding the optimal controls and trajectories in this case is called the time-optimal problem.

In order to formulate the necessary optimality condition it is convenient to adjoin a new coordinate x^0 to the phase coordinates $x^1, \ldots, x^n$, which vary according to (1). Let x^0 vary according to the law

$$\frac{dx^0}{dt} = f_0(x^1, \ldots, x^n, u(t))$$

and consider the system of differential equations

$$\frac{dx^i}{dt} = f_i(x, u), \quad i = 0, \ldots, n, \tag{5}$$

whose right-hand sides do not depend on x^0. Introducing the vector

$$\overline{x} = (x^0, \ldots, x^n)$$

in the $(n+1)$-dimensional vector space $\overline{X}$, system (5) may be rewritten in vector form

$$\frac{d\overline{x}}{dt} = \overline{f}(x, u),$$

where $\overline{f}(x, u)$ is the vector in $\overline{X}$ with coordinates $f_0(x, u), \ldots, f_n(x, u)$. Note that $\overline{f}(x, u)$ does not depend on the coordinate x^0 of the vector $\overline{x}$.

To formulate the theorem, which yields the solution of the fundamental problem, in addition to the fundamental system (5) another system of equations

$$\frac{d\psi_i}{dt} = -\sum_{j=0}^{n} \frac{\partial f_j(x, u)}{\partial x^i} \psi_j, \quad i = 0, \ldots, n, \tag{6}$$

in the auxiliary variables $\psi_0, \ldots, \psi_n$ is considered.

If we choose an admissible control $u(t)$, $t_0 \le t \le t_1$, and have the corresponding phase trajectory $x(t)$ of system (5) with initial condition $x(t_0) = x_0$, system (6) takes the form

$$\frac{d\psi_i}{dt} = -\sum_{j=0}^{n} \frac{\partial f_j(x(t), u(t))}{\partial x^i} \psi_j, \tag{7}$$

where $i = 0, \ldots, n$.

This system is linear and homogeneous. Therefore, for any initial condition it admits the unique solution

$$\overline{\psi} = (\psi_0, \ldots, \psi_n)$$

for the ψ_i, which is defined on the entire interval $t_0 \le t \le t_1$. Just as the solution $\overline{x}(t)$ of (5), the solution of (7) consists of continuous functions $\psi_i(t)$ which have everywhere, except at a finite number of points, i. e., at the points of discontinuity of $u(t)$, continuous derivatives with respect to t. For any initial condition each solution of (7) is called the solution of (6) corresponding to the chosen control $u(t)$ and phase trajectory $\overline{x}(t)$.

Systems (5) and (6) are combined into one entry. To do so the function

$$H(\overline{\psi}, x, u) = (\overline{\psi}, \overline{f}(x, u)) = \sum_{j=0}^{n} \psi_j f_j(x, u)$$

of the variables

$$x^1, \ldots, x^n, \psi_0, \ldots, \psi_n, u^1, \ldots, u^r$$

is considered.

The systems (5) and (6) may be rewritten with the aid of the function H in the form of the Hamiltonian system

$$\frac{dx^i}{dt} = \frac{\partial H}{\partial \psi^i}, \quad i = 0, \ldots, n, \tag{8}$$

$$\frac{d\psi_i}{dt} = -\frac{\partial H}{\partial x^i}, \quad i = 0, \ldots, n. \tag{9}$$

Thus, taking an arbitrary admissible, i. e., piecewise continuous, control $u(t)$, $t_0 \le t \le t_1$, and the initial condition $\overline{x}(t_0) = \overline{x}_0$, the corresponding trajectory $\overline{x}(t) = (x^0(t), \ldots, x^n(t))$ can be found. After that the solution of (9),

$$\overline{\psi}(t) = (\psi_0(t), \ldots, \psi_n(t)),$$

corresponding to the functions $u(t)$ and $\overline{x}(t)$ can be found. It should be emphasized that the vector functions $\overline{x}(t)$ and $\overline{\psi}(t)$ are continuous and have everywhere, except at a finite number of points, continuous derivatives with respect to t.

For fixed values of $\overline{\psi}$ and x the function H becomes a function of the parameter $u \in U$. The least upper bound of the values of this function is denoted by

$$M(\overline{\psi}, x) = \sup_{u \in U} H(\overline{\psi}, x, u).$$

If the continuous function H achieves its upper bound on U, then $M(\overline{\psi}, x)$ is the maximum of the values of H for fixed $\overline{\psi}$ and x.

A necessary condition for optimality is given by Theorem 1, which is called the *maximum principle* [1].

Theorem 1 *Let $u(t)$, $t_0 \le t \le t_1$, be an admissible control such that the corresponding trajectory $\overline{x}(t)$ which begins at the point $\overline{x}_0$ at the time t_0 passes at some time t_1 through a point on a line Π. In order that $u(t)$ and $\overline{x}(t)$ be optimal it is necessary that there exist a nonzero continuous vector function $\overline{\psi}(t) = (\psi_0(t), \dots, \psi_n(t))$ corresponding to $u(t)$ and $\overline{x}(t)$ such that*

1) for every t, $t_0 \le t \le t_1$, the function $H(\overline{\psi}(t), x(t), u)$ of the variable $u \in U$ attains its maximum at the point $u = u(t)$,

$$H(\overline{\psi}(t), x(t), u(t)) = M(\overline{\psi}(t), x(t)) ; \tag{10}$$

2) at the terminal time t_1 the relations

$$\psi_0(t_1) \le 0, \quad M(\overline{\psi}(t_1), x(t_1)) = 0 \tag{11}$$

are satisfied. Furthermore, it turns out that if $\overline{\psi}(t), \overline{x}(t)$ and $u(t)$ satisfy system (8)–(9) and condition 1), the time functions $\psi_0(t)$ and $M(\overline{\psi}(t_1), x(t_1))$ are constant. Thus, (11) may be verified at any time t, $t_0 \le t \le t_1$, and not just at t_1.

From Theorem 1 an analogous necessary condition for time optimality can be derived. To do this, it is necessary to set $f_0(x, u)$ of Theorem 1 equal to 1. The function H then takes the form

$$H = \psi_0 + \sum_{j=1}^{n} \psi_j f_j(x, u) .$$

Introducing the n-dimensional vector $\psi = (\psi_1, \dots, \psi_n)$ and the function

$$H'(\psi, x, u) = \sum_{j=1}^{n} \psi_j f_j(x, u) ,$$

equations (1) and (6) can be rewritten in the form of the Hamiltonian system

$$\frac{\mathrm{d}x^i}{\mathrm{d}t} = \frac{\partial H'}{\partial \psi_i}, \quad i = 1, \dots, n , \tag{12}$$

$$\frac{d\psi_i}{\mathrm{d}t} = -\frac{\partial H'}{\partial x^i}, \quad i = 1, \dots, n . \tag{13}$$

For fixed values of ψ and x, H' is a function of u. The upper bound of the values of this function is denoted by

$$M'(\psi, x) = \sup_{u \in U} H'(\psi, x, u) .$$

Because

$$H'(\psi, x, u) = H(\overline{\psi}, x, u) - \psi_0 ,$$

then

$$M'(\psi, x) = M(\overline{\psi}, x) - \psi_0 ,$$

and therefore (10) and (11) now take the form

$$\begin{aligned} H'(\psi(t), x(t), u(t)) &= M'(\psi(t), x(t)) \\ &= -\psi_0 \ge 0 . \end{aligned}$$

Thus, the following theorem is valid [1].

Theorem 2 *Let $u(t)$, $t_0 \le t \le t_1$, be an admissible control which transfers the phase point from x_0 to x_1 and let $x(t)$ be the corresponding trajectory, so that $x(t_0) = x_0$, $x(t_1) = x_1$. In order that $u(t)$ and $x(t)$ be time-optimal it is necessary that there exist a nonzero, continuous vector function $\psi(t) = (\psi_1(t), \dots, \psi_n(t))$ corresponding to $u(t)$ and $x(t)$ such that*

1) for all t, $t_0 \le t \le t_1$, the function $H'(\psi(t), x(t), u)$ of the variable $u \in U$ attains its maximum at the point $u = u(t)$,

$$H'(\psi(t), x(t), u(t)) = M'(\psi(t), x(t)) ;$$

2) at the terminal time t_1 the relation

$$M'(\psi(t_1), x(t_1)) \ge 0 \tag{14}$$

is satisfied. Furthermore, it turns out that if $\psi(t)$, $x(t)$ and $u(t)$ satisfy system (12)–(13) and condition 1), the time function $M'(\psi(t), x(t))$ is constant. Thus, (14) may be verified at any time t, $t_0 \le t \le t_1$, and not just at t_1.

From among all the trajectories which start at $\overline{x}_0$ and end on some point of Π (as well as the corresponding controls), Theorem 1 allows us to single out those separate, isolated trajectories and controls which satisfy all the formulated conditions. In fact, there are $2n + 3$ relations (8)–(10) for $2n + 3$ variables x^j, ψ_j and u. Furthermore, since (10) is not differential and the number of differential equations equals $2n + 2$, the solutions of the system (8)–(10) in general depend on $2n + 2$ parameters. However, one of these parameters is redundant as the functions $\psi_j(t)$ are defined only up to a common multiple. In addition, one of the parameters is determined by the condition that

$$\max_{u \in U} H(\overline{\psi}(t_0), x(t_0), u)$$

vanishes.

Thus, there are 2n parameters, on which the whole manifold of solutions of (8)–(10) depends. It follows that 2n parameters must be chosen in such a way that the trajectory $\overline{x}(t)$ passes through $\bar{x}_0$ at the given time $t = t_0$ and through a point of Π for some $t > t_0$. The number $t_1 - t_0$ is also a parameter, so that there are altogether $2n + 1$ essential parameters. The condition that the trajectory must pass through the point $\overline{x}_0$ and the line Π gives rise to $2n + 1$ relations. Hence, one can expect that there exist only separate, isolated trajectories joining the point $\overline{x}_0$ with the line Π, which satisfy the conditions of Theorem 1. Only these separate, isolated trajectories can turn out to be optimal.

See also

- ► Dynamic Programming: Continuous-Time Optimal Control
- ► Hamilton–Jacobi–Bellman Equation
- ► High-Order Maximum Principle for Abnormal Extremals

References

1. Pontryagin L, Boltyanskii V, Gamkrelidze R, Mishchenko E (1962) The mathematical theory of optimal processes. Interscience, New York

Portfolio Selection: Markowitz Mean-variance Model

JOHN L.G. BOARD[1], CHARLES M.S. SUTCLIFFE[2], WILLIAM T. ZIEMBA[3]

[1] Department Accounting and Finance, The London School of Economics and Political Sci., London, UK

[2] Accounting and Finance Division, The School of Management, The University Southampton, Southampton, UK

[3] Management Sci. Division Fac. Commerce and Business Admin., University British Columbia, Vancouver, Canada

MSC2000: 91B28

Article Outline

Keywords
Portfolio Selection Problem
The Use of Mean And Variance
Solution of Portfolio Selection Model
Short Selling
Estimation Problems
Summary
References

Keywords

Linear programming; Nonlinear programming; Quadratic programming; Estimation risk; Stein estimators; Short selling; Finance

Portfolio Selection Problem

The heart of the portfolio problem is the selection of an optimal set of investment assets by rational economic agents. Elements of portfolio problems were discussed in the 1930's and 1940's by J.R. Hicks, [19], J. Marschak [46], D.H. Leavens [37], J.B. Williams [62], and others; see [45] for a survey of these early contributions. However, the first formal specification of such a selection model was by H.M. Markowitz [40,42] who defined a mean-variance model for calculating optimal portfolios; see also A.D. Roy [51], whose first safety model is very close to the mean value model. Following [55,58,59] and [50], this portfolio selection model may be stated as:

$$\begin{cases} \min & x'Vx \\ \text{s.t.} & x'r = r_p \\ & x'e = 1, \end{cases} \tag{1}$$

where x is a column vector of investment proportions in each of the risky assets, V is a positive semidefinite variance-covariance matrix of asset returns, r is a column vector of expected asset returns, r_p is the investor's target rate of return and e is a column unit vector. An explicit solution for the problem can be found using the procedures described in [47,66], or [50].

Restrictions on short selling can be modeled by augmenting (1) by the constraints:

$$x \geq 0\,, \tag{2}$$

where 0 is a column vector of zeros. The problem now becomes a classic example of quadratic mathematical

programming; indeed, the development of the portfolio problem coincided with early developments in nonlinear programming. Markowitz's [41] critical line algorithm is a systematic solution procedure. Formal investigations of the properties of both formulations, and variants, appear in [22,57], and the references above.

The Use of Mean And Variance

The economic justification for this model is based on the von Neumann–Morgenstern expected utility results, discussed in this context by Markowitz [42]. The model can also be viewed in terms of consumer choice theory together with the characteristics model developed by K. Lancaster [36]. His argument is that goods purchased by consumers seldom yield a single, well defined service; instead, each good may be viewed as a collection of attributes each of which gives the consumer some benefit (or dis-benefit). Thus preference is defined over those characteristics embodied in a good rather than over the good itself. The analysis focusses attention on the attributes of assets rather than on the assets per se. This requires the assumption that utility depends only on the characteristics. With k characteristics, C_k, we need

$$U = f(W) = g(C_1, \ldots, C_k) ,$$

where U and W represent utility and wealth. Modeling too few characteristics will yield apparently false empirical results. Clearly, the benefits of this approach increase as the number of assets rises relative to the number of characteristics. The objects of choice are the characteristics $C_1, \ldots, C_k$. In portfolio theory, these are taken to be payoff (return) and risk.

At Markowitz's suggestion, when dealing with choice among risky assets, payoff is measured as the expected return of the distribution of returns, and risk by the standard deviation of returns. Apart from minor exceptions, see [66], this pair of characteristics form a complete description of assets which is consistent with expected utility theory in only two cases: assets have normal distributions, or investors have quadratic utility of wealth functions. The adequacy of these assumptions has been investigated by a number of authors (e. g., [5,16,60]). Although returns have been found to be nonnormal and the quadratic utility has a number of objectionable features (not least diminishing marginal utility of wealth for high wealth), several authors demonstrate approximation results which are sufficient for mean variance analysis ([39,49,52]).

A number of authors, including Markowitz [42], consider alternatives to the variance and suggest the use of the semivariance. This suggestion has been extended into workable portfolio selection rules. E. Fama [14] and S. Tsiang [61] have argued the usefulness of the semi-interquartile range as a measure of risk. A. Kraus and R.F. Litzenberger [35] and others have examined the effect of preferences defined in terms of the third moment which allows investor choice in terms of skewness. J.G. Kallberg and W.T. Ziemba [31,32] show that risk aversion preferences are sufficient to determine optimal portfolio choice if assets have normally distributed returns whatever the form of the assumed, concave, utility function.

Solution of Portfolio Selection Model

In the absence of short sales restrictions, (1) can be rewritten as

$$\min L = \frac{1}{2} x' V x - \lambda_1 (x' r - r_p) - \lambda_2 (x' e - 1) . \quad (3)$$

The first order conditions are

$$V x = \lambda_1 r + \lambda_2 e ,$$

which shows that, for any efficient x, there is a linear relation between expected returns r and their covariances, Vx.

Solving for x:

$$x = \lambda_1 V^{-1} r + \lambda_2 V^{-1} e = V^{-1} [r\ e] A^{-1} [r_p\ 1]' , \quad (4)$$

where

$$A = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} r' V^{-1} r & r' V^{-1} e \\ r' V^{-1} e & e' V^{-1} e \end{bmatrix} .$$

Substituting (4) into the definition of portfolio variance, $x' V x$, yields

$$V_p = [r_p 1] A^{-1} [r_p 1]' ,$$
$$S_p = \left[\frac{c r_p^2 - 2 b r_p + a}{ac - b^2} \right]^{\frac{1}{2}} , \quad (5)$$

where V_p and S_p represent portfolio variance and standard deviation, respectively. This defines the efficient

set, which is a hyperbola in mean/standard-deviation space (or a parabola in mean/variance space). The minimum risk is at $S_{\min} = c^{-½}$ and $r_{\min} = b/c$ (both strictly positive). Rational risk averse investors will hold portfolios lying on this boundary with $r \geq r_{\min}$.

Each efficient portfolio, p, has an orthogonal portfolio z (i. e., such that $\mathrm{Cov}(r_p, r_z) = 0$) with return

$$r_z = \frac{a - br_p}{b - cr_p} .$$

Using this, the efficient set degenerates into the straight line tangent to the hyperbola at p which has intercept r_z:

$$r = r_z + \lambda s , \tag{6}$$

where r and s represent vectors of the expected return and risks of efficient portfolios, and $\lambda = (r_p - r_z)/S_p$ can be interpreted as the additional expected return per unit of risk. This is known as the *Sharpe ratio* [54,56]. Equation (6) shows a two-fund separation theorem, that linear combinations of only two portfolios are sufficient to describe the entire efficient set.

Under the additional assumptions of *homogeneous beliefs* (so that all investors perceive the same parameters) and equilibrium, (6) becomes the *capital market line*. The *security market line* (i. e. the relationship between expected returns and systematic risk or beta), which is the outcome of the capital asset pricing model (CAPM), can be derived by premultiplying (4) by V and simplifying using the definitions of V_p and r_z:

$$r = r_z e + (r_p - r_z)\beta , \quad \text{where } \beta = \frac{Vx}{V_p} . \tag{7}$$

If it exists, the risk-free rate of interest may be substituted for r_z (definitionally, the risk-free return will be uncorrelated with the return on all risky assets). Equation (7) then becomes the original CAPM in which expected return is calculated as the risk-free rate plus a risk premium (measured in terms of an asset's covariance with the market portfolio). The CAPM forms one of the cornerstones of modern finance theory and is not addressed here. Discussion of the CAPM can be found in [22] and [17], while systematic fundamental and seasonal violations of the theory are presented in [63] and [34].

Short Selling

The assumption that assets may be sold short (i. e., $x_i < 0$) is justified when the model is used to derive analytical results for the portfolio problem. Also, when considering equilibrium (e. g., the CAPM), none of the short selling constraints should be binding (because in aggregate, short selling must net out to zero). However, significant short selling restrictions do face investors in most real markets. These restrictions may be in the form of absolute prohibition, the extra cost of deposits to back short selling or self imposed controls designed to limit potential losses. For example, the NYSE imposes the 'uptick rule' under which short sales are allowed only if the price of the immediately preceding trade was higher than, or equal to, the trade preceding it (as short selling is profitable when the market is falling), this rule substantially limits its attractiveness.

The set of quadratic programming problems to find the efficient frontier when short sales are ruled out can be formulated as either minimising the portfolio risk for a specified sequence of portfolio returns (r_p) by repeatedly solving equations (1) and (2), or maximising the weighted sum of portfolio risk and return for a chosen range of risk-return trade-off parameters (μ) by repeatedly solving (8). This latter approach has the advantages of locating only points on the efficient frontier and, for evenly spaced increments in μ, locating more points on the efficient frontier where its curvature is greatest.

$$\begin{cases} \max & \alpha = x'Vx - \mu(x'r - r_p) \\ \text{s.t.} & x \geq 0 \\ & x'e = 1. \end{cases} \tag{8}$$

When short sales are permitted, a position (long or short) is taken in every asset, while when short selling is ruled out, the solution involves long positions in only about 10% of the available assets. When short selling is permitted, about half the assets are required to be sold short, often in large amounts, and sometimes in amounts exceeding the initial value of the investment portfolio. Indeed, this is the main activity of 'short seller' funds.

In contrast, most models based on portfolio theory, in particular the CAPM, ignore short selling constraints [43,44]. This change is consistent with the development of equilibrium models for which institutional restric-

tions are inappropriate (and if imposed would not be binding). However, when short selling is permitted, the number of asset return observations is required to exceed the number of assets, while complementary slackness means that this condition need not be met when short selling is ruled out. Computational procedures to solve mean-variance models with various types of constraints, and the optimal combination of safe and risky assets for various utility functions are discussed in [65].

Estimation Problems

The model (1) requires estimates of r and V for the period during which the portfolio is to be held. This estimation problem has been given relatively little attention, and many authors, both practitioners and academics, have used historical values as if they were precise estimates of future values. However, S.D. Hodges and R.A. Brealey [21], among others, demonstrate the benefits obtained from even slight improvements on historical data.

Estimation risk can be allowed for either by using different methods to forecast asset returns, variances and covariances, which are then used in place of the historical values in the portfolio model, or by using the historical values in a modified portfolio selection technique [2]. Since the portfolio selection model of Markowitz takes these estimates as parametric, there is no theoretical guidance on the estimation method and a variety of methods have been proposed to provide the estimates. The *Sharpe single index market model* [53] has been widely applied to forecast the covariance matrix. Originally proposed to reduce the computation required by the full model, it assumes a linear relation between stock returns and some measure of the market, $r = \alpha + \beta' m + \varepsilon$ (for market index m and residuals ε). This uses historical estimates of the means and variances; however, the implied covariance matrix is $V_1 = \nu_m \beta\beta' + V$, where ν_m is the variance of the index, β is a column vector of slope coefficients from regressing each asset on the market index and V is a diagonal matrix of the variances of the residuals from each of these regressions. A number of studies have found that models based on the single index model outperform those based on the full historical method, see e. g. [4].

The *overall mean method*, first proposed in [12], is based on the finding that, although historical estimates of means are satisfactory, data are typically not stable enough to allow accurate estimation of the $N(N-1)/2$ covariance terms. The crudest solution is to assume that the correlations between all pairs of assets expected in the next period are equal to the mean of all the historic correlations. An estimate of V can then be derived from this. E.J. Elton, M.J. Gruber and T.J. Urich [13] compared the overall mean method of forecasting the covariance matrix with forecasts made using historical values, and four alternative versions of the single index model. They concluded that the overall mean model was clearly superior. A simplified procedure for estimating the overall mean correlation appears in [1].

In recent years, statisticians have shown increasing interest in *Bayesian methods* [20] and particularly *James–Stein estimators* ([10,11,30,48]). The intuition behind this approach is that returns which are far from the norm have a higher chance of containing measurement error than those close to it. Thus, estimates of returns, based on individual share data, are cross-sectionally 'shrunk' towards a global estimate of expected returns which is based on all the data. Although these estimators have unusual properties, they are generally expected to perform well in large samples.

P. Jorion [27,28] examined the performance of Bayes–Stein estimation using both simulated and small real data sets and concluded that the Bayes–Stein approach outperformed the use of historical estimates of returns and the covariance matrix. However, he found [29] that the index model outperformed Stein and historical models. J.L.G. Board and C.M.S. Sutcliffe [4] applied these and other methods to large data sets. They found that, in contrast to earlier studies, the relative performance of Bayes–Stein was mixed. While it produced reasonable estimates of the mean returns vector, there were superior methods (e. g., use of the overall mean) for estimating the covariance matrix when short sales were permitted. They also found that, when short sales were prohibited, actual portfolio performance was clearly improved, although there was little to choose between the various estimation methods.

An alternative approach is to try to control for errors in the parameter estimates by imposing additional constraints on (1). Clearly, ex-ante the solution to such a model cannot dominate (1), however,

ex-post, dominance might emerge (i. e., what seems, in advance, to be an inferior portfolio might actually perform better than others). The argument is that adding constraints to (1) to impose lower bounds (i. e., prohibiting short sales) and/or upper bounds (forcing diversification) can be used as an ad hoc method of avoiding the worst effects of estimation risk. Of course, extreme, but possibly desirable, corner solutions will also be excluded by this technique. K.J. Cohen and J.A. Pogue [8] imposed upper bounds of 2.5% on any asset. Board and Sutcliffe [3] studied the effects of placing upper bounds on the investment proportions, which may be interpreted as a response to estimation risk. Using historical forecasts of returns and the covariance matrix, and with short sales excluded, they found that forcing diversification leads to improved actual performance over the unconstrained model. C.R. Hensel and A.L. Turner [18] have also studied adjusting the inputs and outputs to improve portfolio performance.

V.R. Chopra and Ziemba [7], following the work of Kallberg and Ziemba [33], showed that errors in the mean values have a much greater effect than errors in the variances, which are in turn more important than errors in the covariances. Their simulations show errors of the order of 20 to 2 to 1. This quantifies the earlier findings and stresses the importance of having good estimates of the asset means. Chopra [6] shows that mean, variance and covariance errors affect turnover in the same way.

Another approach is to use fundamental analysis to provide external information to modify the estimates [21]. Clearly, among the simplest external data to add are the seasonal (e. g., turn of the year, and month and weekend) effects which have been found in most stock markets around the world. Incorporation of these into the parameter estimates can substantially improve the performance of the model. Ziemba [63] demonstrated the benefits of factor models to estimate the mean returns.

Summary

We have considered only the single period mean-variance portfolio theory model. Although recent developments have focussed on extending the model to multiple periods, most of these models which assume frictionless capital markets require the solution of a sequence of instantaneous mean-variance models in which the existence of transactions costs adds enormously to the complexity of the problem. Surveys covering dynamic portfolio theory appear in [9,22,66], and [23]; see also [64].

References

1. Aneja YP, Chandra R, Gunay E (1989) A portfolio approach to estimating the average correlation coefficient for the constant correlation model. J Finance 44:1435–1438
2. Bawa VS, Brown SJ, Klein RW (1979) Estimation risk and optimal portfolio choice. North-Holland, Amsterdam
3. Board JLG, Sutcliffe CMS (1988) Forced diversification. Quart Rev Economics and Business 28(3):43–52
4. Board JLG, Sutcliffe CMS (1994) Estimation methods in portfolio selection and the effectiveness of short sales restrictions: UK evidence. Managem Sci 40:516–534
5. Borch K (1969) A note on uncertainty and indifference curves. Review of Economic Studies 36:1–4
6. Chopra VR (1993) Improving optimization. J Investing Fall:51–59
7. Chopra VR, Ziemba WT (1993) The effect of errors in means, variances and covariances on optimal portfolio choice. J Portfolio Management 19(2):6–13
8. Cohen KJ, Pogue JA (1967) An empirical evaluation of alternative portfolio selection models. J Business 40:166–193
9. Constantinides G, Malliaris G (1995) Portfolio theory. In: Jarrow R, Maksimovic V, Ziemba WT (eds) Handbook Finance. North-Holland, Amsterdam
10. Efron B, Morris C (1975) Data analysis using Stein's estimator and its generalizations. J Amer Statist Assoc 70:311–319
11. Efron B, Morris C (1977) Stein's paradox in statistics. Scientif Amer 236(5):119–127
12. Elton EJ, Gruber MJ (1973) Estimating the dependence structure of share prices: Implications for portfolio selection. J Finance 28:1203–1232
13. Elton EJ, Gruber MJ, Urich TJ (1978) Are betas best? J Finance 33:1375–1384
14. Fama E (1971) Risk, return and equilibrium. J Political Economy 79:30–55
15. Fama E (1976) Foundations of finance. Blackwell, Oxford
16. Feldstein M (1969) Mean variance analysis in the theory of liquidity preference and portfolio selection. Review of Economic Studies 36:5–12
17. Ferson W (1995) Theory and testing of asset pricing models. In: Jarrow R, Maksimovic V, Ziemba W (eds) Handbook Finance. North-Holland, Amsterdam
18. Hensel CR, Turner AL (1998) Making superior asset allocation decisions: A practitioner's guide. In: Mulvey JM, Ziemba WT (eds) Worldwide Asset and Liability Modeling. Cambridge University Press, Cambridge, pp 62–83

19. Hicks JR (1935) A suggestion for simplifying the theory of money. Economica (February, 1935):1–19
20. Hodges SD (1976) Problems in the application of portfolio selection. OMEGA Internat J Management Sci 4:699–709
21. Hodges SD, Brealey RA (1973) Portfolio selection in a dynamic and uncertain world. Financial Analysts J 29(March, 1973):50–65
22. Huang CF, Litzenberger RH (1988) Foundations for financial economics. North-Holland, Amsterdam
23. Ingersoll J (1987) Theory of financial decision making. Rowman & Littlefield, Lanham
24. Jarrow R Maksimovic V, Ziemba WT (eds) (1995) Finance. North-Holland, Amsterdam
25. Jobson JD, Korkie B (1981) Putting Markowitz theory to work. J Portfolio Management Summer (1981):70–74
26. Jobson JD, Korkie B, Ratti V (1979) Improved estimation for Markowitz portfolios using James–Stein type estimators. Proc. Business Economics and Statistics Section. Amer. Statist. Assoc., Alexandria, VA, pp 279–284
27. Jorion P (1985) Internat. portfolio diversification with estimation error. J Business 58:259–278
28. Jorion P (1986) Bayes–Stein estimation for portfolio analysis. J Financial and Quantitative Anal 21:279–292
29. Jorion P (1991) Bayesian and CAPM estimators of the means: Implications for portfolio selection. J Banking and Finance 15:717–727
30. Judge GG, Bock ME (1978) The statistical implications of pre-test and Stein-rule estimators in econometrics. North-Holland, Amsterdam
31. Kallberg JG, Ziemba WT (1979) On the robustness of the Arrow–Pratt risk aversion measure. Economics Lett 2:21–26
32. Kallberg JG, Ziemba WT (1983) Comparison of alternative utility functions in portfolio selection. Managem Sci 29:1257–1276
33. Kallberg JG, Ziemba WT (1984) Mis-specification in portfolio selection problems. In: Bamberg G, Spremann K (eds) Risk and Capital. Lecture Notes Economics and Math Systems. Springer, Berlin
34. Keim DB, Ziemba WT (eds) (2000) Security market imperfections in worldwide equity markets. Cambridge University Press, Cambridge
35. Kraus A, Litzenberger RF (1976) Skewness preference and the valuation of risk assets. J Finance 31:1085–1100
36. Lancaster K (1971) Consumer demand: A new approach. Columbia Univ Press, New York
37. Leavens DH (1945) Diversification of investments. 80 (Trusts and Estates 469–473
38. Levy H (1969) A utility function depending on the first three moments. J Finance 24:715–719
39. Levy H, Markowitz HM (1979) Approximating executed utility by a function of mean and variance. Amer Economic Rev 69:308–317
40. Markowitz HM (1952) Portfolio selection. J Finance March 7(1):77–91
41. Markowitz HM (1956) The optimization of a quadratic function subject to linear constraints. Naval Res Logistics Quart 3:111–133
42. Markowitz HM (1959) Portfolio selection: Efficient diversification of investments. Yale University Press, New Haven
43. Markowitz HM (1983) Nonnegative or not nonnegative: A question about CAPMs. J Finance 38:283–295
44. Markowitz HM (1987) Mean-variance in portfolio choice and capital markets. Blackwell, Oxford
45. Markowitz HM (1999) The early history of portfolio theory: 1600–1960. Financial Analysists J (July/August 1999):5–16
46. Marschak J (1938) Money and the theory of assets. Econometrica 6:311–325
47. Merton RC (1972) An analytic derivation of the efficient portfolio frontier. J Financial and Quantitative Anal 7:1851–1872
48. Morris C (1983) Parametric empirical Bayes inference: Theory and applications. J Amer Statist Assoc 78:47–55
49. Ohlson J (1975) Asymptotic validity of quadratic utility as the trading interval approaches zero. In: Ziemba WT, Vickson RG (eds) Stochastic Optimization Models in Finance
50. Roll R (1972) A critique of the asset pricing theory's tests. J Financial Economics 4:129–176
51. Roy AD (1952) Safety first and the holding of assets. Econometrica 20(3):431–449
52. Samuelson P (1970) The fundamental approximation theorem of portfolio analysis in terms of means variances and higher moments. Rev Economic Stud 37:537–542
53. Sharpe WF (1963) A simplified model for portfolio analysis. Managem Sci 9:277–293
54. Sharpe WF (1966) Mutual fund performance. J Business 39:119–138
55. Sharpe WF (1970) Portfolio theory and capital markets. McGraw-Hill, New York
56. Sharpe WF (1994) The Sharpe ratio. J Portfolio Management Fall (1994):49–58
57. Szegö GP (1980) Portfolio theory, with application to bank assess management. Acad. Press, New York
58. Tobin J (1958) Liquidity preference as behaviour towards risk. Review of Economic Studies 26:65–86
59. Tobin J (1965) The theory of portfolio selection. In: Hahn FH, Brechling FPR (eds) The Theory of Interest Rates. Internat. Economic Assoc. and MacMillan, London, pp 3–51
60. Tsiang S (1972) The rationale of the mean standard deviation analysis, skewness preference and the demand for money. Amer Economic Rev 62:354–371
61. Tsiang S (1973) Risk, return and portfolio analysis: Comment. J Political Economy 81:748–751
62. Williams JB (1938) The theory of investment value. Harvard University Press, Cambridge
63. Ziemba WT (1994) World wide security market regularities. Europ J Oper Res 74(2):198–229
64. Ziemba WT, Mulvey JM (eds) (1998) Worldwide asset and liability modelling. Cambridge University Press, Cambridge

65. Ziemba WT, Parkan C, Brooks-Hill FJ (1974) Calculation of investment portfolios with risk free borrowing and lending. Managem Sci 21:209–222
66. Ziemba WT, Vickson RG (eds) (1975) Stochastic optimization models in finance. Acad. Press, New York

Portfolio Selection and Multicriteria Analysis

CONSTANTIN ZOPOUNIDIS[1], CHRISTIAN HURSON[2]
[1] Department Production Engineering and Management, Financial Engineering Lab., Techn. University Crete University Campus, Chania, Greece
[2] IUT de Rouen Technique de Commercialisation, University Rouen, CREGO, Mont Saint Aignan, France

MSC2000: 91B28, 90C26

Article Outline

Keywords
MCDM and Portfolio Selection
  Portfolio Selection: A Multicriteria Problem
  Review of Existing Study
Methodological Framework
  Basic Components
  Portfolio Data and Criteria
  Multicriteria Analysis for the Selection of Attractive Stocks
  MINORA System
  The ELECTRE TRI Outranking MCDM Method
  Multicriteria Analysis for the Determination of Attractive Stocks' Proportions in a Portfolio: The ADELAIS System
  Ranking of Stocks Using the MINORA System
  Sorting of Stocks Using the ELECTRE TRI Method
  Determination of the Attractive Stocks' Proportions in a Portfolio Using the ADELAIS System
Concluding Remarks
See also
References

Keywords

Finance; Portfolio management; Multi-objective linear programming; Outranking relation; Additive utility functions

Portfolio selection concerns the problem of finding the most attractive stocks and the determination of their proportions in a portfolio, which is essentially a matter of arbitration between the risk and the return. In 1952 H.M. Markowitz [17] proposing the mean-variance model gives the start of a theory that had known since a great development. According to this approach, an investor follows two conflicting objectives that are the maximization of the expected return and the minimization of the risk measured by the variance of the return. On the same basis (essentially measuring the risk by the variance or another measure of the variation of the return) several other models were developed. A complete overview of these models can be found in [5], including the single-index models, the multi-index models, the average correlation models, the mixed models, the utility model and models using different criteria such as the geometric mean return, stochastic dominance, safety first and skewness.

However, an analysis of risk nature in *portfolio management* shows that the latter comes from various origins and then its nature is multidimensional. The traditional theoretical approach mentioned above does not take into account this multidimensional measure of risk. Also, individual goals and investor's preferences cannot be incorporated in these models. *Multiple criteria decision making* (MCDM) provides the methodological basis to resolve the inherent multicriteria nature of portfolio selection problem. Additionally, it builds realistic models by taking into account, apart of the two basic criteria of return and risk (mean-variance model), a number of important criteria, such as marketability, price earning ratio (PER), growth of the dividends, and others. Furthermore, MCDM, have the advantage of taking into account the preferences of any particular investor. Recently, several authors have developed a new approach, using MCDM for portfolio management.

In this article we develop some arguments for the use of MCDM in portfolio management and we present a brief review of the existing articles concerning this new approach (Section 2). Then, we propose a multicriteria methodological framework in two stages (Section 3). In the first stage the ELECTRE TRI [28] method and the MINORA ([24,25]) systems are used to select attractive stocks. In the second stage, the ADELAIS [23] system is used to construct a portfolio of the attractive stocks chosen in the first stage. This methodological framework is illustrated on a case study. Finally,

we summarize the used methodological framework and underline its advantages then give some concluding remarks.

MCDM and Portfolio Selection

Portfolio Selection: A Multicriteria Problem

To manage efficiently portfolio selection, it is necessary to take into account all the factors that influence the financial markets. Then, portfolio management is a multicriteria problem. Effectively, multifactor models and APT (arbitrage pricing theory) point out the existence of several influence factors for the determination of the stock prices. Furthermore, fundamental analysis models, commonly used in practice, underlines that stock prices are also dependent on the firm health and its capacity to pay dividends. The latter problem itself is a multicriteria problem because, in order to solve it, we must appreciate the profitability of the firm, its debt level (in the short and long terms) and quality of management. Finally, in practice, an investor has a personal attitude and particular objectives.

By reducing the risk to its probabilistic dimension, the classical approach cannot take into account its multidimensional nature and risk measures that do not have a concrete and practical economical meaning. In addition, this approach imposes a norm to the investor's behavior that can be restrictive. Also, it cannot take into account the personal attitude and preferences of a real investor confronted with a given risk in a particular situation. However, experience has proved that the classical approach is useful, for instance concerning the diversification principle and the use of the beta as measure of risk. Thus, the use of the classical approach seems to be necessary but not sufficient, to manage portfolio selection efficiently. Some additional criteria must be added to the classical risk-return criteria. In practice, these additional criteria can be found in fundamental analysis or constructed following the personal goals of the investor.

Note that the application of the above principles is difficult because of the complexity of multicriteria problems on the one hand and the use of criteria from different origins and of conflicting nature on the other hand. Furthermore MCDM will facilitate and favor the analysis of compromise between the criteria. It equally permits to manage the heterogeneity of criteria scale and the fuzzy and imprecise nature of the evaluation that it will contribute to clarify. (Here the words fuzzy and imprecise refer to: a) the delicacy of an investor's judgement (the human nature and the lack of information), that will not always allow to discriminate between two close situations; and b) the use of a representation model, which is a simplification of reality that expresses itself in an error term.)

Linking the multicriteria evaluation of an asset portfolio and the research of a satisfactory solution to the investor's preferences, the MCDM methods allow to take into account the investors' specific objectives. Furthermore, these methods do not impose any normative scheme to the comportment of the investors. The use of MCDM methods allows to synthesize in a single procedure the theoretical and practical aspects of portfolio management, then it allows a non normative use of theory.

The originality of MCDM provides the possibility to obtain a gain of time and/or to increase the number of stocks considered by the practitioner by systematizing the decision process. Moreover, in a market close to efficiency, as are all the big markets, it is the good and fast use of all available information that ensures informational efficiency of capital markets and will permit the investor to compete.

Review of Existing Study

In this section we present a brief review of articles concerning MCDM and portfolio management. An analysis and a more complete presentation of most of these papers can be found in [7] or [12]. T.L. Saaty, P. Rogers and R. Pell [22] proposed in 1980 to construct a portfolio using the analytic hierarchy process (AHP) methodology. They consider that stocks must be compared following the investor objectives and the criteria that influence the prices. In addition, the latter criteria are themselves influenced by some global economic factors. In AHP, factors, criteria and stocks are successively weighed following their relative importance. Finally, the weight of each stock gives its proportion in the portfolio.

S.M. Lee and L. Chesser [16] present a goal programming (GP) model to construct a portfolio. The used objectives are the research of a minimum return, minimization of risk (using β), various diversification

objectives and some specific objectives to the investors. In GP, the investor can set his desires in preference order in a simple and natural way.

A. Rios-Garcia and S. Rios-Insua [20] construct a portfolio using multi-attribute utility theory (MAUT) and *multi-objective linear programming* (MOLP). Particularly, the authors propose the use of Bayesian econometrics to update the investor multi-attribute utility function, and examine the problem posed by the nonlinear criteria in MOLP.

Y. Evrard and R. Zisswiller [6] use MAUT to perform a valuation of some stocks. The aim is to show how it is possible to construct models that link the attributes of the stocks (return, risk, PER and earnings per share) and the investors' preferences.

H. Nakayama, T. Takeguchi and M. Sano [19] propose a graphics interactive methodology to construct a portfolio using the expected return, the variance and the evolution of the return in the past.

J.M. Martel, N.T. Khoury and M. Bergeron [18] perform a portfolio selection using the outranking methods ELECTRE I and ELECTRE II. From the stocks included in two portfolios selected by a real investor, they generate 50 portfolios. These portfolios are compared using ELECTRE I and II and follow four criteria: the return, the logarithmic variance, the PER, and the liquidity. The aim is to determine which portfolio fits well to the decision criteria.

G. Colson and C. De Bruyn [1] propose a system that performs a stock valuation and allows the construction of a portfolio. The heart of this system is the confrontation of the following objectives:

1) Obtain a minimum level of gain.
2) Maintain the risk under a predetermined level.
3) Obtain a minimum level of gain as dividends and interests.
4) Insure a sufficient level of diversification, firm control or liquidity.

This system is subdivided between two subsystems, the single decision model (SDM) and the simultaneous management model (SMM). The SDM ranks the stocks using various statistics criteria and information coming from correspondents. The SMM is a GP model as in [16].

A. Szala [26] performs stock evaluation in collaboration with a French investment company. For the financial analyst of the company, that examines a small number of stocks using numerous criteria, Szala uses the outranking method ELECTRE III to obtain a ranking of the stocks. For traders or portfolios managers, it is not realistic to use many criteria because they generally manage a great number of stocks. Then, concerning traders and portfolio managers, Szala decided to amalgamate the financial criteria in a synthesis criterion obtained by using the PREFCALC system (PREFCALC is an interactive system of the same nature as MINORA that will be presented in the following section of this paper).

Khoury, Martel and Bergeron [14] use the outranking methods ELECTRE IS and ELECTRE III to select international index portfolios. They generate 19 index portfolios from the indexes of 16 countries. The criteria that have been used are: return, standard deviation, transaction costs, country risk, available cover for foreign currencies and exchange risk.

The purpose of Colson and M. Zeleny ([2,29]) is to construct an efficient frontier in concordance with the principles of stochastic dominance. To achieve their aim, they propose to use a three-dimensional vector, the 'prospect ranking vector' (PRV), as a measure of risk. The first component of the PRV is the probability not to achieve a minimal return, the second component is the expected return and the third component is the probability to exceed a maximum return. We find an interest updating the PRV in order to perform a portfolio selection rather than construct an efficient frontier [9]. To achieve this result, it is necessary to modify the PRV in order to obtain more complete measures of risk. Then, the first component is divided into two components: the first is destined to protect the investor against strong losses and the second is used to take into account the other possible losses (not so strong but significant). Concerning the third component, the probability to exceed a maximum level of return is changed in the probability to significantly exceed the expected return. Then, this new version of the PRV can be associated to other criteria to perform a multicriteria portfolio selection. In [9] portfolio selection is managed by using the MINORA system that will be presented in the following section.

C. Zopounidis, D.K. Despotis and I. Kamaratou [31] propose the use of the ADELAIS system to construct a portfolio using some diversification constraints, some constraints representing the investor's personal pref-

erences and the following criteria: return, dividends, β, earnings per share growth, marketability, PER. The ADELAIS system is also used in our methodology that will be presented later.

M. Tamiz, R. Hasham and D.F. Jones [27] propose to use GP for portfolio evaluation and selection. The proposed method consists of two stages: the first stage estimates the sensitivity of the stocks to specific factors using GP and regression analysis. They use twelve factors: UK, US and German interest rates, US and German inflation rates, Dow Jones index, Nikkei average index, Hang Sang index, oil price, gold price, house price and sterling index. The sensitivities obtained by the GP model and those of the regression analysis are compared. In the second stage, a portfolio is selected by using a GP model based on the decision maker's scenarios and preferences.

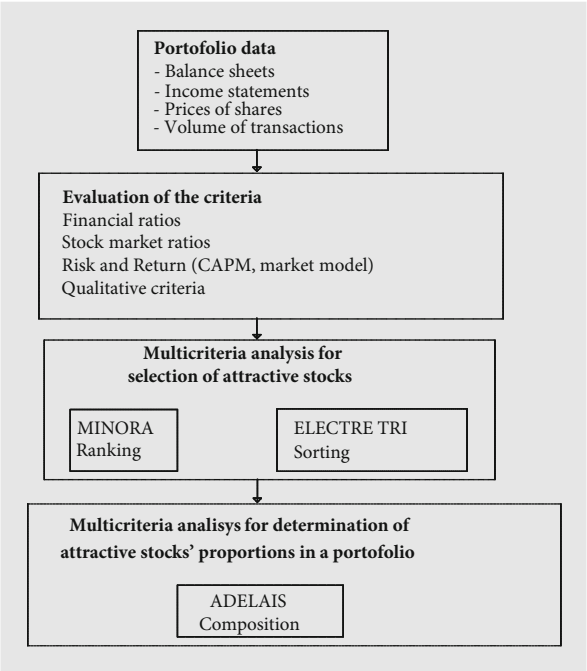
C. Dominiak [4] presents a procedure for security selection that uses a multicriteria discrete analysis method based on the idea of reference solution. The criteria he used are divided into three groups:

1) Valuation measures that contain Price book value ratio and PER.
2) Fundamental variables that contain profit margin ratio and changes in quarterly net profits.
3) Technical indicators that contain rate of change, relative strength index and price appreciation during the last 3 months.

Ch. Hurson and N. Ricci [8] propose to combine arbitrage pricing theory (APT) and MCDM to model the portfolio management process. First, APT is used to construct some efficient portfolios, to estimate their expected return and to identify influence factors and risk origins. Then, two multicriteria decision methods: the ELECTRE TRI outranking method and the MINORA interactive system are used to select attractive portfolios, using APT factors as selection criteria. This methodology is illustrated by an application to the Greek market.

Methodological Framework

In this section we will present our methodology and illustrate it on a case study. For a more detailed presentation and applications on Greek and Belgium market, see [7,10,11,12,32]. A first application of MINORA to portfolio management can also be found in [30].



Portfolio Selection and Multicriteria Analysis, Figure 1
Basic components of the methodological framework

Basic Components

The basic components of the proposed methodological framework are presented in Fig. 1.

Portfolio Data and Criteria

For the analysis and the selection of stocks, it is necessary to possess the following basic information for at least three consecutive years: balance sheets, income statements and stock market information, i. e. prices of shares and volume of transactions. This consecutive financial information allows the investor or portfolio manager to evaluate the evolution of the stock's situation, and to form financial and stock market ratios relevant to portfolio management.

On the basis of the portfolio data one can calculate financial and stock market ratios (return on equity, return on investment, current ratio, interest charges/sales, earnings per share, price earnings ratio, dividend per share, etc.), the pair risk/return from the theoretical portfolio models capital assets pricing model (CAPM) and market model. Apart from the quantitative criteria, the investor or portfolio manager

needs to have more general information so that his evaluation would be as objective and complete as possible. Such information about a stock may be its quality of management, image, security of information, position on the market, etc. This qualitative information is sometimes more important than the financial, because, for example, if the firm does not have good managers, its financial results (sales, net income, return) will not be satisfactory.

Financial and stock market ratios have become an accepted evaluation technique of financial analysts and portfolio managers. They provide a quantitative view of every element that concerns the internal operation of a firm as well as its relations with the outer world, and permit fast processing of a large volume of financial data. The financial ratios have already been used in many fields of financial management. C.F. Lee [15] has grouped every financial ratio that has been used in the forecasting of firm failure, bond rating, market return and mergers. The risk and the return issue of theoretical portfolio models (CAPM and market model) are the basic criteria in portfolio selection. Qualitative criteria are modeled according to the preferences of each user (portfolio manager) with the aid of an ordinal scale (3 better than 2 and 2 better than 1).

Multicriteria Analysis for the Selection of Attractive Stocks

In this step the two MCDM methods MINORA and ELECTRE TRI are used to help the portfolio manager in the selection of a set of attractive stocks taking into consideration his preferences and experiences. Here, only a short description of the two methods is given.

MINORA System

MINORA is an interactive multicriteria decision making system that ranks a set of alternatives from the best ones to the worst ones using several criteria. In this purpose the MINORA system uses the UTA ranking algorithm [13]. From the ranking by the decision maker of a subset of well-known alternatives, UTA uses *ordinal regression* to estimate a set of separable *additive utility functions* of the following form:

$$u(g) = u_1(g_1) + \cdots + u_k(g_k)\,,$$

where $g = (g_1, \ldots, g_k)$ is the performance vector of an alternative and $u_i(g_i)$ is the marginal utility function of criteria i, normalized between 0 and 1. The ordinal regression is performed using linear programming (for more details see [3]). In MINORA the interaction takes the form of an analysis of inconsistencies between the ranking established by the decision maker and the ranking obtained from the utility function estimated by UTA. Two measures of these inconsistencies are used in MINORA:

1) The F indicator which is the sum of the deviations of the ordinal regression curve (global utility versus decision maker's ranking), e. g. the sum of estimation errors.
2) The Kendall's t that gives a measure, from -1 to 1, of the correlation between the decision maker ranking and the ranking resulting from the utility function.

At optimality, when the two rankings are similar, the F indicator is equal to 0 and the Kendall's t to 1. The interaction is organized around four questions presented to the decision maker:

1) Is he ready to modify his ranking?
2) Does he wish to modify the relative importance of a criterion, its scale or the marginal utilities (trade-off analysis)?
3) Does he wish to modify the family of criteria used: to add, cancel, modify, divide or join some criteria?
4) Does he wish to modify the whole formulation of the problem?

These questions send back to the corresponding stages of MINORA and the method stops when an acceptable compromise is determined. Then the result (a utility function) is extrapolated to the whole set of alternative to give a ranking of them.

The MINORA system presents two main advantages:

- It furnishes a ranking of stocks that is a natural preoccupation frequently used by portfolio managers.
- The form of the interactivity, all the originality of the MINORA system can be found in the inconsistencies analysis in an interactive way. It allows to help the decision maker to construct his own model in a non normative way and organizes, in an unique procedure, all the activity of decision making, from the model formulation to the final result (a ranking of the alternatives from the best to the worst in the

case of MINORA). At the same time the decision maker is constantly integrated to the resolution processes and can control its evolution at any moment.

Finally, it should be noted that the MINORA system has been used successfully to solve numerous management problems.

The ELECTRE TRI Outranking MCDM Method

ELECTRE TRI is an outranking method specially conceived for *sorting problems*. In our methodology the ELECTRE TRI method is used to sort the stocks in three categories: attractive stocks, uncertain stocks (to be studied further) and nonattractive stocks. ELECTRE TRI deals only with ordered categories (complete order). The categories are defined by some reference alternatives or reference profiles (one down profile and one up profile) which are themselves defined by their values on the criteria. Next we can define the categories C_i, $i = 1, \ldots, c$, where C_1 is the worst category and C_c the best one. We can also define the profiles r_i, $i = 1, \ldots, c-1$, where r_1 and r_{c-1} are the lower and the upper profile respectively. Then, the profile r_i is the theoretical limit between two categories C_i and C_{i+1} and r_i represents a fictitious stock which is strictly better than r_{i-1} on each criterion. In ELECTRE TRI, the information asked from the decision maker about his preferences takes the form, for each criterion and each profile, of a relative weight and an indifference, preference and veto thresholds. To sort the stocks, ELECTRE TRI compares each of them to the profiles using the concepts of indifference, preference and veto thresholds in order to construct a concordance index, a discordance index and finally a valued outranking relation as in ELECTRE III method (cf. [21]). This valued outranking relation $s_s(a, b) \in [0, 1]$ measures the strength of the relation 'a outranks b' (a is at least as good as b). This valued outranking relation is transformed in a 'net' outranking relation in the following way: $s_s(a, b) \geq \lambda \Leftrightarrow aSb$, where S represents the net outranking relation, a and b two stocks and λ a 'cut level' ($0.5 \leq \lambda \leq 1$) above which the relation a outranks b is considered as valid. Then the preference P, the indifference I and the incomparability R are defined as follows:

- $aIb \Leftrightarrow aSb$ and bSa,
- $aPb \Leftrightarrow aSb$ and no bSa,
- $aRb \Leftrightarrow$ no aSb and no bSa.

In ELECTRE TRI there are two non total compensation sorting procedures (the pessimistic one and the optimistic one) to assign each alternative into one of the categories defined in advance. In the sorting procedure, stock a is compared at first to the worst profile r_1 and in the case of aPr_1, a is compared to the second profile r_2, etc., until one of the following situations appears:

i) aPr_i or aIr_{i+1} and $r_{i+1}\, Pa$;

ii) aPr_i and $r_{i+1}\, Ra, \ldots, r_{i+m}\, Ra, r_{i+m+1}\, Pa$.

In situation i) both the procedures assign stock a to category $i + 1$. In situation ii), the pessimistic procedure classifies stock a to category $i + 1$, while the optimistic procedure classifies stock a to category $i + m + 1$. When the value of λ gradually decreases, the pessimistic procedure becomes less compulsive and the optimistic procedure less permissive. Evidently the optimistic procedure tends to classify the stocks to the higher possible category, in contrast to the pessimistic procedure that tends to classify the stocks to the lower possible category. In general, the pessimistic procedure is applied when a policy of prudence is necessary or when the available means are very constraining. The optimistic procedure is applied to problems where the decision maker desires to favor the alternatives that present some particular interest or some exceptional qualities. In portfolio management the optimistic procedure will be well adapted to an optimistic investor with a speculative investment policy, while a prudent investor, following a passive investment policy, will prefer the pessimistic procedure.

ELECTRE TRI manages incomparability in such a way that it will point out the alternatives that have particularities in their evaluations. In cases where some alternatives belong to different categories in both procedures, the conclusion is that they are incomparable with one or more reference profiles (as the number of categories between the two assignments is increasing, the 'particularities' of the alternatives are becoming more important). This is because these alternatives have good values for some criteria and, simultaneously, bad values for other criteria; moreover, these particular alternatives must be examined with attention. In this way the notion of incomparability included in ELECTRE TRI brings an important information to the decision maker and for this reason the best way to employ ELECTRE TRI is to use the two assignments procedures and to

compare the results. The advantages of ELECTRE TRI are the following:

- ELECTRE TRI by sorting the stocks is well adapted to the purpose of portfolio management (acceptable stocks, stocks to be studied further and unacceptable stocks).
- ELECTRE TRI, as all the methods of the ELECTRE family, accepts intransitivity and incomparability. In ELECTRE TRI this is done in such a way that the method will point out the alternatives that have particularities in their evaluation.
- The ELECTRE family uses techniques that are easily understandable by the decision maker.

The methods from the ELECTRE family are very popular and they have been used with success in a great number of studies [21].

Multicriteria Analysis for the Determination of Attractive Stocks' Proportions in a Portfolio: The ADELAIS System

MINORA and ELECTRE TRI give the possibility to the portfolio manager to select the most attractive stocks he desires to include in his final portfolio. Then, the ADELAIS system helps him to determine the proportions (percentage) of capital invested in the above attractive stocks. ADELAIS is an interactive computer system developed to support the search for a satisfactory solution in MOLP problems of the general form:

$$\begin{cases} \max & \{f_1(\mathbf{x}), \ldots, f_n(\mathbf{x})\} \\ \text{s.t.} & \mathbf{x} \in A, \end{cases}$$

where $\mathbf{x} = (x_1, \ldots, x_m)$ is the vector of decision variables, $f_1(\mathbf{x}), \ldots, f_n(\mathbf{x})$ are linear functions of the decision variables and A is the set of the feasible solutions defined by linear constraints $A \subseteq \mathbf{R}^m$. The system provides extensive data management capabilities and concerning the solution process it provides a 'two level' interaction: interactive assessment of the portfolio manager's utility function and interactive modification of the satisfaction levels. The system operates in two phases: a preliminary phase that is performed once and an iterative phase. In the preliminary phase, ADELAIS provides the upper and lower bounds for each objective function, the pay-off matrix, the starting efficient solution in a minimax sense and its rate of closeness to the ideal values (upper bounds). At each iteration of the iterative phase a new efficient solution is presented the portfolio manager with the upper and lower bounds for each objective, the achievement percentages with respect to the upper bounds, the satisfaction levels (i. e., the revised lower bounds) established in previous iterations. Then, the system asks the portfolio manager to indicate the objectives he desires to improve and if he intends to decrease some other objectives in compensation. The portfolio manager's answers, combined with relative answers of previous iterations, form the basis for the establishment of new satisfaction levels. The new satisfaction levels limit the feasible set but the system provides the portfolio manager with the possibility to relax them if desired by analyzing the local trade-offs among the objectives. If the portfolio manager does not intend to decrease some objectives the system stops on the current efficient solution (best compromise), otherwise a reference set of decision alternatives (i. e. a set of vectors that might be assumed by the objectives) is constructed and the UTA method is used to estimate an additive utility function as in the MINORA system. Once a satisfactory utility function is assessed, this is maximized over the set A in order to obtain a new efficient solution taking into account the portfolio manager's preferences.

The advantages of the ADELAIS system for the portfolio manager, are the following:

- It allows determining the proportions of the attractive stocks in the portfolio that is not possible with ranking or sorting methods.
- The interactive analysis of the inconsistencies as in the MINORA system, which helps the portfolio manager to understand the portfolio selection problem.
- The interactive revision of the satisfaction levels orientates in a way easy to understand the research of the best compromise that is well adapted to the portfolio manager's preferences.

Example 1 We will illustrated our methodology by a case study coming from [10]. The sample considered in the study consists of 40 stocks coming from the financial and commercial sectors of the Athens Stock Exchange (ASE). These stocks are evaluated using the 7 following criteria:

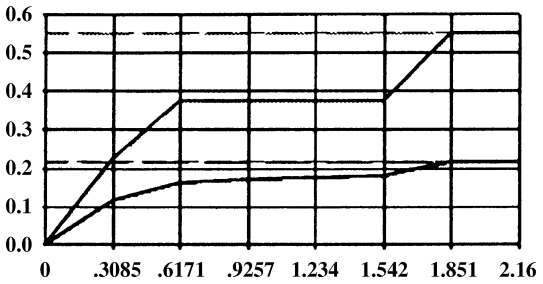
- g_1: Return that must be maximized.
- g_2: Marketability (percentage of shares traded) that must be maximized.
- g_3: Beta risk that must be minimized.
- g_4: Price earnings ratio (price of the share/earnings per share). Since there are negative values of earnings (losses), we choose to maximize the inverse of this criterion
- g_5: Growth of dividend per share that must be maximized.
- g_6: Acid test (current assets minus inventories/current liabilities) that must be maximized.
- g_7: Return on equity (i. e. net income/stockholder equity) that must be maximized.

All the stocks of the sample have been evaluated on the five stock market criteria $g_1, \ldots, g_5$, while, in order to obtain most significant results, the criterion g_6 is used for the stocks of the commercial sector (CS1 to CS20) and the criterion g_7 for the stocks of the financial sector (FS1 to FS20).

Table 1 presents the multicriteria evaluation for the stocks of the commercial sector. Qualitative criteria are not included because they are dependent to the portfolio manager's personal information that was not available in this application.

Portfolio Selection and Multicriteria Analysis, Table 1
Multicriteria evaluation of stocks for the commercial sector

	g_1	g_2	g_3	g_4	g_5	g_6
CS1	0.82	0.45	0.26	−4.7	−100	0.45
CS2	0.41	0.63	0.03	2.28	−20	2.04
CS3	0.57	0.2	0.1	6.08	−33.3	1.08
CS4	0.24	0.02	0.08	2.41	−53.5	0.62
CS5	0	0.46	0.62	5.04	−76.5	3.02
CS6	0.93	0.02	0.14	2.82	6.38	0.72
CS7	0.01	0.69	0.77	7.55	−40	3.23
CS8	0.86	0.86	0.86	4.28	3.71	0.57
CS9	2.16	0.6	0.12	2.11	56.3	0.51
CS10	1.24	0.12	0.62	11.65	12.5	1.17
CS11	0.8	0.58	0.62	13.7	34.6	1.54
CS12	1.23	0.37	0.64	8.97	45.9	0.96
CS13	0.24	0.28	0.73	−1.75	0	0.72
CS14	0.26	0.65	0.58	4.88	7.14	0.9
CS15	1.1	0.76	0.54	0.29	0	0.73
CS16	1.79	0.55	0.73	5.88	−100	2.69
CS17	1.02	1.06	0.82	5.5	6.38	0.73
CS18	1.32	1.12	0.94	12.06	−61	2.69
CS19	1.36	0.04	1.02	1.79	110	2.31
CS20	0.57	0.17	0.23	−11.5	0	0.52



Portfolio Selection and Multicriteria Analysis, Figure 2
Marginal utility curves of the criterion return

Ranking of Stocks Using the MINORA System

The reference set and its ranking for the commercial sector is the following: CS16; CS18; CS12, CS19; CS9, CS3; CS7, CS8; CS1; CS20; for this purpose the portfolio manager can use a graphic help. The MINORA system provides two basic results: the criteria's graphics (i. e. marginal utilities, Fig. 2 and the ordinal regression curve (ranking versus global utility, Fig. 3). An important graphic help is at the disposal of the portfolio manager in order to interpret these results. In Fig. 2 there are three utility curves for the return criterion (min, middle and max). The middle one corresponds to the above presented model of additive utility and, also, gives the relative weight for the criterion. The two others show the entire range of the possible marginal utility functions and gives an idea of the sensitivity of the criteria. Figure 3 shows the ordinal regression curve (stocks' ranking versus global utility).

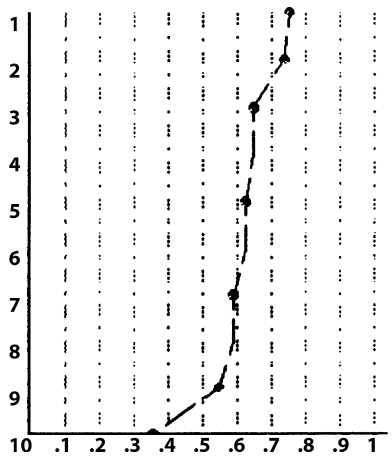
Table 2 presents the pre-ordering of the portfolio manager (D.R.), the stock names (Actions), the global utility of every stock (G.U.) and the ordering from MINORA (M.R.).

Sorting of Stocks Using the ELECTRE TRI Method

The objective is to sort the stocks in the following three categories: attractive stocks (C3), uncertain stocks to be studied further (C2), non attractive stocks (C1). The reference profiles and the thresholds related to the concordance and discordance indices are fixed by the port-

Portfolio Selection and Multicriteria Analysis, Table 2
Global utility for the sample of 40 Greek stocks (extrapolation phase); (∗) Stocks not included in the reference set

Actions	G.U.	D.R.	M.R.	Actions	G.U.	D.R.	M.R.
CS16	0.761	1	1	FS5	0.679	∗	1
CS18	0.745	2	2	FS10	0.665	1	2
CS11	0.684	∗	3	FS20	0.570	2	3
CS12	0.656	3	4	FS9	0.570	2	3
CS19	0.656	3	4	FS6	0.555	∗	5
CS2	0.643	∗	6	FS2	0.554	∗	6
CS17	0.635	∗	7	FS18	0.549	4	7
CS3	0.632	5	8	FS3	0.549	4	7
CS9	0.632	5	8	FS17	0.534	∗	9
CS10	0.622	∗	10	FS19	0.532	∗	10
CS7	0.596	7	11	FS4	0.511	6	11
CS8	0.596	7	11	FS15	0.504	∗	12
CS15	0.582	∗	13	FS14	0.499	7	13
CS5	0.559	∗	14	FS7	0.490	8	14
CS1	0.554	9	15	FS8	0.490	8	14
CS14	0.554	∗	16	FS16	0.484	∗	16
CS13	0.495	∗	17	FS12	0.476	∗	17
CS6	0.471	∗	18	FS1	0.433	∗	18
CS4	0.388	∗	19	FS11	0.395	10	19
CS20	0.362	10	20	FS13	0.210	∗	20



Portfolio Selection and Multicriteria Analysis, Figure 3
Portfolio manager's ordinal regression curve

Portfolio Selection and Multicriteria Analysis, Table 3
Reference profiles and thresholds

Parameters	g_1	g_2	g_3	g_4	g_5	g_6
High ref. profile	1	0.35	0.6	6	10	1.1
Low ref. profile	0.5	0.7	0.25	2.5	0	0.7
Indiff. threshold	0.05	0.05	0	0.1	8.72	0.05
Pref. threshold	0.25	0.2	0.2	0.5	10	0.25
Veto threshold	2	1	1	10	180	2.75

folio manager according to his experience and preferences. Table 3 presents this preferential information. Table 4 presents the sorting results of the ELECTRE TRI method and recapitulates the MINORA results for comparison. The stocks that belong to the best category (C3) in both optimistic and pessimistic sorting are proposed without hesitation to the portfolio manager for selection. The stocks that belong to the worst category (C1) in both optimistic and pessimistic sorting are not proposed to the portfolio manager. When the stocks belong to the uncertain category (C2) for both optimistic and pessimistic sorting, this means that these have moderate values on all criteria and, consequently,

Portfolio Selection and Multicriteria Analysis, Table 4
Sorting of stocks

	Optimistic	Pessimistic
C1	CS4, CS13, CS20, FS1, FS7, FS11, FS12, FS13	CS1, CS4, CS13, CS19, CS20, FS1, FS7, FS9, FS11, FS12, FS13, FS14
C2	CS1, CS2, CS3, CS5, CS6, CS8, CS14, CS15, CS17, FS8, FS9, FS14, FS19	CS2, CS3, CS5, CS6, CS7, CS8, CS9, CS12, CS14, CS15, CS17, FS3, FS8, FS10, FS16, FS19
C3	CS7, CS9, CS10, CS11, CS12, CS16, CS18, CS19, FS2, FS3, FS4, FS5, FS6, FS10, FS15, FS16, FS17, FS18, FS20	CS10, CS11, CS16, CS18, FS2, FS4, FS5, FS6, FS15, FS17, FS18, FS20

they must be studied further. In the cases where some stocks belong to different categories in both optimistic and pessimistic sorting, this means that they are incomparable with one or two reference profiles. This is due to the fact that these stocks have good values for some criteria and, simultaneously, bad values for other criteria. Thus, the notion of incomparability underlies the particularities of these stocks that must be examined further and brings an important information to the portfolio manager. Comparing the ranking results of the MINORA system with the sorting results of the ELECTRE TRI method, one can remark, generally, that there is an agreement, that is, the stocks which are well ranked (i. e. top of the ranking) by MINORA are in the best category C3 by ELECTRE TRI, and vice versa. This agreement between these two methods asserts the interest of the methodology and allows the portfolio manager to be confident with their results.

Determination of the Attractive Stocks' Proportions in a Portfolio Using the ADELAIS System

According to the results obtained above, the portfolio manager can choose a subset of the best stocks from each sector and then, using the ADELAIS system, determine the proportions invested in each selected stock. The chosen set is the following: FS10, FS5, FS20, FS6, FS2 from the financial sector and CS16, CS18, CS12, CS11, CS9 from the commercial sector.

The decision variables, X_i, are the percentages of capital invested in each stock. For the needs of the study we note: X_1 = FS10, X_2 = FS5, X_3 = FS20, X_4 = FS6, X_5 = FS2, X_6 = CS16, X_7 = CS18, X_8 = CS12, X_9 = CS11, X_{10} = CS9.

Since, the return, the marketability, the beta, the price earnings ratio and the growth dividend per share of the constructed portfolio are directly related to the percentages of capital invested in each stock, only these five criteria are used in the application of the ADELAIS system. On the other hand, the acid test ratio and the return on equity can only be used as evaluation criteria of the financial soundness of each stock, and they do not characterize the constructed portfolio. Consequently, the objective functions are:

- Maximize the return (g_1): max $R_1X_1 + \cdots + R_{10}X_{10}$.
- Maximize the marketability (g_2): max $M_1X_1 + \cdots + M_{10}X_{10}$.
- Minimize the beta (g_3): min $B_1X_1 + \cdots + B_{10}X_{10}$.
- Maximize the price earnings ratio (g_4): max $PER_1X_1 + \cdots + PER_{10}X_{10}$.
- Maximize the growth of dividends per share (g_5): max $GD_1X_1 + \cdots + GD_{10}X_{10}$.

Here, R_i, M_i, B_i, PER_i, and GD_i are the values of the corresponding objective for the stock i. The constraints are:

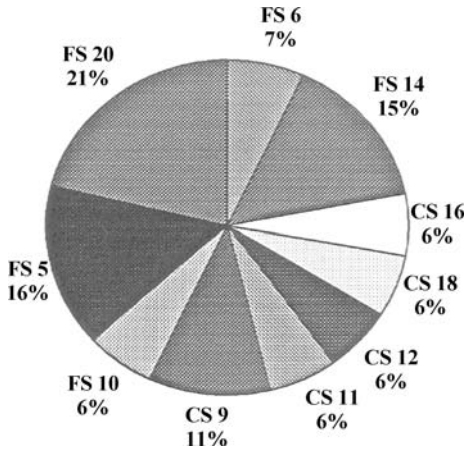
- $X_1 + \cdots + X_{10} = 1$, all the available capital must be invested.
- $0.05 < X_1 < 0.2, \ldots, 0.05 < X_{10} < 0.2$ upper and lower limits of the amount to be invested in each stock (% of capital).

At the beginning (preliminary phase) ADELAIS proceeded to the estimation of an initial efficient portfolio of stocks (compromise) and presents it as in Table 5. A pay-off table for the objectives was also given to the portfolio manager.

With respect to the initial solution, the portfolio manager being satisfied by the attained values of the marketability and the growth of dividends per share asked for an improvement of the other objectives. On the basis of this information the system generated a set of portfolios, and asked

Portfolio Selection and Multicriteria Analysis, Table 5
Initial efficient solution; R_c is the rate of closeness to the ideal (upper bound)

Criteria Initial solution	g_1	g_2	g_3	g_4	g_5
Upper bound	1.61	0.64	0.672	13.98	49.92
Compromise	1.44	0.58	0.745	11.9	47.5
Satisfaction Level	12.05	0.34	0.933	9.08	−6.33
Lower bound	12.05	0.34	0.933	9.08	−6.33
R_c	58.2%	81.2%	60.5%	57.5%	95.7%



Portfolio Selection and Multicriteria Analysis, Figure 4
Portion of capital invested in the portfolio

Portfolio Selection and Multicriteria Analysis, Table 6
Best compromise solution

Solution	g_1	g_2	g_3	g_4	g_5
Attained values	1.47	0.52	0.73	11.9	47.22
R_c	65%	61.9%	65.8%	57.5%	95.2%

the portfolio manager to rank them. On the basis of these data, ADELAIS assesses an additive utility model and use it to determine a new reference solution. Finally, after three iterations ADELAIS determines the best compromise solution that could not be improved. Table 6 presents the values of the objectives and the rate of closeness to the ideal point, the corresponding portfolio appears in Fig. 4.

Concluding Remarks

In this article a review of articles is presented concerning MCDM and portfolio management and a methodological framework for portfolio selection. MCDM is a new supportive tool for portfolio selection. The use of multicriteria decision making methods allows to take into consideration the investors personal preferences and all the relevant criteria, whatever their origins, for portfolio selection. In our methodology, the conjoined use of the MINORA system and the ELECTRE TRI method have shown some advantages as the complementarity and the similarity of the obtained results that state the portfolio manager's confidence in the constitution of his portfolio; also, they satisfy one of the portfolio managers' preoccupation which is the ranking and the sorting of stocks in portfolio selection. Secondly, ELECTRE TRI with the notion of incomparability brings an important information to the portfolio manager, especially when the evaluation of stocks appears difficult. Thirdly, ADELAIS and MINORA systems provide a considerable aid to the portfolio manager to construct his own model of portfolio selection in an interactive way. This portfolio selection model is without any normative consideration proposed by the classical portfolio theory. Finally, the topic of portfolio management also covers the portfolio diversification by categories of securities (stocks, bonds, options, international securities, etc.), in order to compensate the risks. The construction of a portfolio for each of these securities and the diversification problem also are multicriteria. Thus, it will be interesting to look at the contribution that MCDM can bring to these problems. The final aim is to regroup these problems in order to formalize and improve all the processes of portfolio management.

See also

- Bi-objective Assignment Problem
- Competitive Ratio for Portfolio Management
- Decision Support Systems with Multiple Criteria
- Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- Financial Applications of Multicriteria Analysis
- Financial Optimization
- Fuzzy Multi-objective Linear Programming
- Multicriteria Sorting Methods
- Multi-objective Combinatorial Optimization
- Multi-objective Integer Linear Programming
- Multi-objective Optimization and Decision Support Systems
- Multi-objective Optimization: Interaction of Design and Control
- Multi-objective Optimization: Interactive Methods for Preference Value Functions
- Multi-objective Optimization: Lagrange Duality
- Multi-objective Optimization: Pareto Optimal Solutions, Properties
- Multiple Objective Programming Support
- Outranking Methods
- Preference Disaggregation
- Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- Preference Modeling
- Robust Optimization
- Semi-infinite Programming and Applications in Finance

References

1. Colson G, De Bruyn C (1989) An integrated multiobjective portfolio management system. Math Comput Modelling 12(10–11):1359–1381
2. Colson G, Zeleny M (1979) Uncertain prospects ranking and portfolio analysis under the condition of partial information. Math Systems in Economics, vol 44. Anton Hain Verlag, Bodenheim
3. Despotis DK, Yannacopoulos D, Zopounidis C (1990) A review of the UTA multicriteria method and some improvements. Found Computing and Decision Sci 15(2):63–76
4. Dominiak C (1997) Portfolio selection using the idea of reference solution. In: Fandel G, Gal T (eds) Multiple Criteria Decision Making, Proc. Twelfth Internat. Conf. Lecture Notes Economics and Math Systems. Hagen, pp 593–602
5. Elton EJ, Gruber MJ (1987) Modern portfolio theory and investment analysis, 3rd edn. Wiley, New York
6. Evrard Y, Zisswiller R (1983) The setting of market investor preferences: An exploratory study based on multiattribute models. Working Paper, Centre d'Enseign Sup des Affaires
7. Hurson Ch (1995) La gestion de portefeuilles boursiers et l'aide multicritère à la décision. PhD Thesis, GREQAM, Univ. d'Aix-Marseille II
8. Hurson Ch, Ricci N (1998) Multicriteria decision making and portfolio management with arbitrage pricing theory. In: Zopounidis C (ed) Operational Tools in The Management of Financial Risks. Kluwer, Dordrecht, pp 31–55
9. Hurson Ch, Zopounidis C (1993) Return, risk measures and multicriteria decision support for portfolio selection. In: Papathanassiou B, Paparrizos K (eds) Proc. 2nd Balkan Conf. Oper. Res., Thessaloniki, pp 343–357
10. Hurson Ch, Zopounidis C (1995) On the use of multicriteria decision aid methods for portfolio selection. J Euro-Asian Management 1/2:69–94
11. Hurson Ch, Zopounidis C (1996) Méthodologie multicritère pour l'évaluation et la gestion de portefeuilles d'actions. Banque et March 28:11–23
12. Hurson Ch, Zopounidis C (1997) Gestion de portefeuilles et analyse multicritère. Economica. Collection Gestion Poche
13. Jacquet-Lagrèze E, Siskos J (1982) Assessing a set of additive utility functions for multicriteria decision making: The UTA method. Europ J Oper Res 10:151–164
14. Khoury N, Martel JM, Veilleux M (1993) Méthode multicritère de sélection de portefeuilles indiciels internationaux. L'Actualité Economique 69(1)
15. Lee CF (1985) Financial analysis and planning: Theory and applications. Addison-Wesley, Reading
16. Lee SM, Chesser L (1980) Goal programming for portfolio selection. J Portfolio Management Spring 1980:22–26
17. Markowitz H (1952) Portfolio selection. J Finance 7(1):77–91
18. Martel JM, Khoury NT, Bergeron M (1988) An application of a multicriteria approach to portfolio comparisons. J Oper Res Soc 39(7):617–628
19. Nakayama H, Takeguchi T, Sano M (1983) Interactive graphics for portfolio selection. In: Hansen P (ed) Essays and Surveys on Multiple Criteria Decision Making. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 280–289
20. Rios-Garcia S, Rios-Insua S (1983) The portfolio problem with multiattributes and multiple criteria. In: Hansen P (ed) Essays and Surveys on Multiple Criteria Decision Making. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 317–325
21. Roy B, Bouyssou D (1993) Aide multicrit'ere à la décision: Méthodes et cas. Economica, Heidelberg
22. Saaty T, Rogers P, Pell R (1980) Portfolio selection through hierarchies. J Portfolio Management (Spring, 1980):16–21

23. Siskos J, Despotis DK (1989) A DSS oriented method for multiobjective linear programming problems. Decision Support Systems 5:47–55
24. Siskos Y, Spiridakos A, Yannacopoulos D (1993) MINORA: A multicriteria decision aiding system for discrete alternatives. J Inform Sci and Techn 2(2):136–149
25. Spyridakos A, Yannacopoulos D (1995) A visual approach to the procedures of multicriteria intelligence decision aiding systems. In: Janssen J Skiadas CH, Zopounidis C (eds) Advances in Stochastic Modelling and Data Analysis. Kluwer, Dordrecht, pp 339–354
26. Szala A (1990) L'Aide à la décision en gestion de portefeuille. Dipl Sup Rech Appliq, Univ Paris Dauphine
27. Tamiz M, Hasham R, Jones DF (1997) A comparison between goal programming and regression analysis for portfolio selection. In: Fandel G, Gal T (eds) Multiple Criteria Decision Making, Proc. 12th Internat. Conf. Lecture Notes Economics and Math Systems. Hagen, pp 422–432
28. Yu W (1992) ELECTRE TRI - Aspects méthodologiques et manuel d'utilisation. Lamsade Lab, Univ Paris Dauphine 74
29. Zeleny M (1977) Multidimensional measure of risk: The prospect ranking vector. In: Zionts S (ed) Multiple Criteria Problem Solving. Springer, Berlin, pp 529–548
30. Zopounidis C (1993) On the use of the MINORA decision aiding system to portfolio selection and management. J Inform Sci and Techn 2(2):150–156
31. Zopounidis C, Despotis DK, Kamaratou I (1998) Portfolio selection using the ADELAIS multiobjective linear programming system. Comput Economics 11(3):189–204
32. Zopounidis C, Godefroid M, Hurson Ch (1995) Designing a multicriteria decision support system for portfolio selection and management. In: Janssen J Skiadas CH, Zopounidis C (eds) Advances in Stochastic Modelling and Data Analysis. Kluwer, Dordrecht, pp 261–292

Potential Reduction Methods for Linear Programming

YINYU YE
Department Management Sci., University Iowa, Iowa City, USA

MSC2000: 90C05, 37A35

Article Outline

Keywords
Primal Potential Reduction Algorithm
Primal-Dual Potential Reduction Algorithm
See also
References

Keywords

Linear programming; Primal-dual; Potential function; Potential reduction

Consider the *linear programming* (LP) problem in the standard form:

$$\begin{cases} \min & c^\top x \\ \text{s.t.} & Ax = b, \\ & x \geq 0, \end{cases} \tag{1}$$

where the given matrices $A \in \mathbf{R}^{m\times n}$, $c \in \mathbf{R}^n$, $b \in \mathbf{R}^m$, and the unknown vector $x \in \mathbf{R}^n$. The dual of (LP), (DLP), can be written as:

$$\begin{cases} \max & b^\top y \\ \text{s.t.} & A^\top y + s = c, \\ & s \geq 0, \end{cases} \tag{2}$$

where the unknown vector $y \in \mathbf{R}^m$. Let $\mathcal{F}$ be the *feasible set* of (x, y, s), and let the *interior* of $\mathcal{F}$ be denoted by int $\mathcal{F}$, i. e., this is the set of feasible (x, y, s) such that $x > 0$ and $s > 0$.

Potential reduction algorithms for linear programming are generally equipped with various *potential functions* that are solely used to measure the solution's progress. There is no restriction on either path following or stepsize during the iterative process; the greater the reduction of the potential function, the faster the convergence of the algorithm. These potential functions represent the *logarithmic volumes* of certain *coordinate-aligned ellipsoids* containing the optimal solution set. There are three types of potential functions (see the references). For $(x, y, z) \in$ int $\mathcal{F}$, the first is the *primal-dual potential function*

$$\psi(x, s) = \rho \log(x^\top s) - \sum_{j=1}^{n} \log(x_j s_j) ,$$

where $\rho \geq n$. This function is also called *Tanabe–Todd–Ye potential function*. Note that for $\rho = n$

$$\psi(x, s) = n \log(x^\top s) - \sum_{j=1}^{n} \log(x_j s_j) \geq n \log n .$$

Thus, for $\rho > n$, ψ (x, s) approaching $-\infty$ implies $x^\top s$ converging to 0.

Let $z = b^{\mathsf{T}} y$ and $\overline{z} = c^{\top} x$ for some strictly feasible x and (y, s), respectively. Then, consider

$$\phi_p(x, \underline{z}) = \rho \log(c^{\top} x - \underline{z}) - \sum_{j=1}^{n} \log(x_j)$$

and

$$\phi_d(y, s, \underline{z}) = \rho \log(\overline{z} - b^{\top} y) - \sum_{j=1}^{n} \log(s_j) .$$

ϕ_p is called *Karmarkar's potential function* or the *primal potential function*, and ϕ_d is called the *dual potential function*. The three functions are closely related:

$$\psi(x, s) = \phi_p(x, \underline{z}) - \sum_{j=1}^{n} \log s_j ,$$

and

$$\psi(x, s) = \phi_d(y, s, \overline{z}) - \sum_{j=1}^{n} \log x_j .$$

Thus, the reduction in $\phi_p(x, z)$ or $\phi_d(y, s, \overline{z})$, when (y, s) or x is fixed, implies the same reduction in $\psi(x, s)$.

Depending on which potential being used as the primary reduction, there are also three types of potential reduction algorithms. Here, we describe the primal and primal-dual algorithms.

Primal Potential Reduction Algorithm

Consider a pair of $(x^k, y^k, s^k) \in \text{int}\, \mathcal{F}$. Fix $z^k = b^{\mathsf{T}} y^k$, then the *gradient vector* of the primal potential function with respect to x^k is

$$\begin{aligned} \nabla \phi_p(x^k, z^k) &= \frac{\rho}{(s^k)^{\top} x^k} c - (X^k)^{-1} \mathrm{e} \\ &= \frac{\rho}{c^{\top} x^k - z^k} c - (X^k)^{-1} \mathrm{e} . \end{aligned}$$

We directly solve the *ball-constrained linear problem* for direction d_x:

$$\begin{cases} \min & \nabla \phi_p(x^k, z^k)^{\top} d_x \\ \text{s.t.} & A d_x = 0, \\ & \left\| (X^k)^{-1} d_x \right\| \le \alpha. \end{cases}$$

Let the minimizer be d_x. Then

$$d_x = -\alpha \frac{X^k p^k}{\left\| p^k \right\|} ,$$

where

$$\begin{aligned} p^k &= p(z^k) \\ &:= \left(I - X^k A^{\top} (A (X^k)^2 A^{\top})^{-1} A X^k \right) \\ &\quad \times X^k \nabla \phi_p(x^k, z^k) . \end{aligned}$$

Update

$$x^{k+1} = x^k + d_x = x^k - \alpha \frac{X^k p^k}{\left\| p^k \right\|} , \tag{3}$$

and we have

$$\phi_p(x^{k+1}, z^k) - \phi_p(x^k, z^k) \le -\alpha \left\| p^k \right\| + \frac{\alpha^2}{2(1-\alpha)} .$$

Thus, as long as $\| p^k \| \ge \eta > 0$, we may choose an appropriate α such that

$$\phi_p(x^{k+1}, z^k) - \phi_p(x^k, z^k) \le -\delta$$

for some positive constant δ. By the relation between $\psi_\rho(x, s)$ and $\phi_p(x, z)$, the primal-dual potential function is also reduced. That is,

$$\psi_\rho(x^{k+1}, s^k) - \psi_\rho(x^k, s^k) \le -\delta .$$

However, even if $\| p^k \|$ is small, we will show that the *primal-dual potential function* can be reduced by a constant δ by increasing z^k and updating (y^k, s^k).

We focus on the expression of p^k, which can be rewritten as

$$\begin{aligned} p^k &= \left(I - X^k A^{\top} (A (X^k)^2 A^{\top})^{-1} A X^k \right) \\ &\quad \times \left(\frac{\rho}{c^{\top} x^k - z^k} X^k c - \mathrm{e} \right) \\ &= \frac{\rho}{c^{\top} x^k - z^k} X^k s(z^k) - \mathrm{e} , \end{aligned} \tag{4}$$

where

$$s(z^k) = c - A^{\top} y(z^k) \tag{5}$$

and

$$\begin{aligned} y(z^k) &= y_2 - \frac{c^{\top} x^k - z^k}{\rho} y_1 , \\ y_1 &= (A (X^k)^2 A^{\top})^{-1} b , \\ y_2 &= (A (X^k)^2 A^{\top})^{-1} A (X^k)^2 c . \end{aligned} \tag{6}$$

One can show that, when $\| p(z^k) \|$ is small, then $(x^k, y(z^k), s(z^k))$ is in the neighborhood of the *central path* and $b^\top y(z^k) > z^k$. Thus, we can increase z^k to $b^\top y(z^k)$. Moreover, $\phi(x^k, s(z^k))$ is reduced from $\phi(x^k, s^k)$ by a constant. Overall, we have the following potential reduction theorem to evaluate the progress.

Theorem 1 *Given $(x^k, y^k, s^k) \in int\, \mathcal{F}$. Let $\rho = n + \sqrt{n}$, $z^k = b^\top y^k$, x^{k+1} be given by (5), and $y^{k+1} = y(z^k)$ in (6) and $s^{k+1} = s(z^k)$ in (5). Then, either*

$$\psi(x^{k+1}, s^k) \leq \psi(x^k, s^k) - \delta$$

or

$$\psi(x^k, s^{k+1}) \leq \psi(x^k, s^k) - \delta\,,$$

where $\delta > 1/20$.

This theorem establishes an important fact: the *primal-dual potential function* can be reduced by a constant no matter where x^k and y^k are. In practice, one can perform the line search to minimize the primal-dual potential function. This results in the following primal-dual potential reduction algorithm.

Given	a central path point $(x^0, y^0, s^0) \in \text{int}\mathcal{F}$.
Let	$z^0 = b^\top y^0$.
Set	$k := 0$.
WHILE	$(s^k)^\top x^k \geq \epsilon$ DO
1	Compute y_1 and y_2 from (6).
2	IF there exists z such that $s(z) > 0$, THEN compute
	$\bar{z} = \arg \min_z \psi_p(x^k, s(z))$,
	FI
	IF $\psi_p(x^k, s(\bar{z})) < \psi_p(x^k, s^k)$, THEN $y^{k+1} = y(\bar{z})$, $s^{k+1} = s(\bar{z})$ and $z^{k+1} = b^\top y^{k+1}$; ELSE $y^{k+1} = y^k$, $s^{k+1} = s^k$ and $z^{k+1} = z^k$. FI
3	Let $x^{k+1} = x^k - \overline{\alpha} X^k p(z^{k+1})$ with $\overline{\alpha} = \arg \min_{\alpha \geq 0} \psi_p(x^k - \alpha X^k p(z^{k+1}), s^{k+1})$.
4	Set $k := k + 1$ and return to Step 1.

Primal algorithm

The performance of the algorithm results from the following corollary.

Corollary 2 *Let $\rho = n + O(\sqrt{n})$. Then, the primal algorithm terminates in at most $O(\frac{\sqrt{n}\log((x^0)^\top s^0)}{\epsilon})$ iterations with*

$$c^\top x^k - b^\top y^k \leq \epsilon\,.$$

Primal-Dual Potential Reduction Algorithm

Another technique for solving linear programs is the symmetric primal-dual algorithm. Once we have a pair $(x, y, s) \in \text{int}\,\mathcal{F}$ with $\mu = x^\top s/n$, we can generate a new iterate x^+ and (y^+, s^+) by solving for d_x, d_y and d_s from the system of linear equations:

$$\begin{aligned} S d_x + X d_s &= \gamma\mu\, \mathrm{e} - Xs\,, \\ A d_x &= 0\,, \\ -A^\top d_y - d_s &= 0\,. \end{aligned} \tag{7}$$

Let $d := (d_x, d_y, d_s)$. To show the dependence of d on the current pair (x, s) and the parameter γ, we write $d = d(x, s, \gamma)$. Note that $d_x^\top d_s = -d_x^\top A^\top d_y = 0$ here.

The system (7) is the *Newton step* starting from (x, s) which helps to find the point on the central path with duality gap $\gamma\, n\mu$. If $\gamma = 0$, it steps toward the optimal solution characterized by the optimality conditions of (LP) and (DLP); if $\gamma = 1$, it steps toward the central path point of μ; if $0 < \gamma < 1$, it steps toward a central path point with a smaller complementarity gap. In the algorithm presented here, we choose $\gamma = n/\rho < 1$. Each iterate reduces the primal-dual potential function by at least a constant δ, as does the previous potential reduction algorithm.

Let the direction $d = (d_x, d_y, d_s)$ be generated by equation (7) with $\gamma = n/\rho$, and let

$$\theta = \frac{\alpha\sqrt{\min(Xs)}}{\left\| (XS)^{-\frac{1}{2}} \left(\frac{x^\top s}{\rho}\mathrm{e} - Xs \right) \right\|}\,, \tag{8}$$

where α is a positive constant less than 1. Let

$$\begin{aligned} x^+ &= x + \theta d_x\,, \\ y^+ &= y + \theta d_y\,, \\ s^+ &= s + \theta d_s\,. \end{aligned}$$

Then, we should have $(x^+, y^+, s^+) \in \text{int}\,\mathcal{F}$ and

$$\psi_\rho(x^+, s^+) - \psi_\rho(x, s) \le -\alpha\sqrt{\min(Xs)}\left\|(XS)^{-\frac{1}{2}}\left(\mathrm{e} - \frac{\rho}{x^\top s}Xs\right)\right\| + \frac{\alpha^2}{2(1-\alpha)}.$$

Let $v \in \mathbf{R}^n$ be a positive vector and $\rho \ge \sqrt{n}$. Then, we can prove

$$\sqrt{\min(v)}\left\|V^{-\frac{1}{2}}\left(\mathrm{e} - \frac{\rho}{e^\top v}v\right)\right\| \ge \sqrt{\frac{3}{4}}.$$

Combining these we have

$$\psi_\rho(x^+, s^+) - \psi_\rho(x, s) \le -\alpha\sqrt{\frac{3}{4}} + \frac{\alpha^2}{2(1-\alpha)} = -\delta$$

for a constant δ. This result will provide a competitive theoretical iteration bound, but a faster algorithm may be again implemented by conducting a line search along the direction d to achieve the greatest reduction in the primal-dual potential function. This leads to the following algorithm.

Given	$(x^0, y^0, s^0) \in \text{int}\,\mathcal{F}$.
Set	$p \ge \sqrt{n}$ and $k := 0$.
WHILE	$(s^k)^\top x^k \ge \epsilon$ DO
1	Set $(x, s) = (x^k, s^k)$ and $\gamma = n/p$; compute (d_x, d_y, d_s) from (7).
2	Let $x^{k+1} = x^k + \overline{\alpha} d_x$, $y^{k+1} = y^k + \overline{\alpha} d_y$, and $s^{k+1} = s^k + \overline{\alpha} d_s$, where $\overline{\alpha} = \arg\min_{\alpha \ge 0} \psi_\rho\,(x^k + \alpha d_x, s^k + \alpha d_s)$.
3	Set $k := k + 1$ and return to Step 1.

Primal-dual algorithm

Theorem 3 *Let $\rho = n + O(\sqrt{n})$. Then, the primal-dual algorithm terminates in at most $O\left(\sqrt{n}\log(\frac{(x^0)^\top s^0}{\epsilon})\right)$ iterations with*

$$c^\top x^k - b^\top y^k \le \epsilon.$$

Potential reduction methods have been successively extended to solving nonlinear *conic programs*, see, e. g., [13].

See also

- Entropy Optimization: Interior Point Methods
- Homogeneous Selfdual Methods for Linear Programming
- Interior Point Methods for Semidefinite Programming
- Linear Programming: Interior Point Methods
- Linear Programming: Karmarkar Projective Algorithm
- Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Anstreicher KM (1986) A monotonic projective algorithm for fractional linear programming. Algorithmica 1(4):483–498
2. Anstreicher KM, Bosch RA (1992) Long steps in a O(n3L) algorithm for linear programming. Math Program 54:251–265
3. Freund RM (1991) Polynomial-time algorithms for linear programming based only on primal scaling and projected gradients of a potential function. Math Program 51:203–222
4. Gay DM (1987) A variant of Karmarkar's linear programming algorithm for problems in standard form. Math Program 37:81–90 Errata: (1988) Math. Program. 40:11
5. de Ghellinck G, Vial JP (1986) A polynomial Newton method for linear programming. Algorithmica 14):425–453
6. Goldfarb D, Xiao D (1991) A primal projective interior point method for linear programming. Math Program 51:17–43
7. Gonzaga CC (1990) Polynomial affine algorithms for linear programming. Math Program 49:7–21
8. den Hertog D (1992) Interior point approach to linear, quadratic and convex programming, algorithms and complexity. PhD Thesis, Fac. Math. Inform., TU Delft
9. Karmarkar NK (1984) A new polynomial-time algorithm for linear programming. Combinatorica 4:373–395
10. Kojima M, Megiddo N, Ye Y (1992) An interior point potential reduction algorithm for the linear complementarity problem. Math Program 54:267–279
11. Mizuno S (1991) O(npL) iteration O(n3L) potential reduction algorithms for linear programming. Linear Alg & Its Appl 152:155–168
12. Monteiro RDC (1992) On the continuous trajectories for a potential reduction algorithm for linear programming. Math Oper Res 17:225–253
13. Nesterov EYu, Nemirovskii AS (1991) Acceleration and parallelization of the path-following interior point method for

a linearly constrained convex quadratic problem. SIAM J Optim 1(4):548–564
14. Roos C, Terlaky T, Vial J-Ph (1997) Theory and algorithms for linear optimization: An interior point approach. Wiley, New York
15. Sturm JF, Zhang S (1996) New complexity results for the Iri–Imai method. Ann Oper Res 62:539–564
16. Tanabe K (1987) Complementarity-enforced centered Newton method for mathematical programming. In: Tone K (ed) New Methods for Linear Programming. Institute Statist. Math. Japan, Tokyo, pp 118–144
17. Todd MJ (1996) Potential-reduction methods in mathematical programming. Math Program 76:3–45
18. Todd MJ, Burrell BP (1986) An extension of Karmarkar's algorithm for linear programming using dual variables. Algorithmica 1(4):409–424
19. Todd MJ, Ye Y (1990) A centered projective algorithm for linear programming. Math Oper Res 15:508–529
20. Tsuchiya T (1991) Global convergence of the affine scaling methods for degenerate linear programming problems. Math Program 52:377–404
21. Tunçel L (1994) Constant potential primal-dual algorithms: A framework. Math Program 66:145–159
22. Tutuncu R (1995) An infeasible-interior-point potential-reduction algorithm for linear programming. PhD Thesis, School Oper. Res. and Industr. Engin., Cornell University
23. Vandenberghe L, Boyd S (1996) Semidefinite programming. SIAM Rev 38(1):49–95
24. Ye Y (1991) An O(n3L) potential reduction algorithm for linear programming. Math Program 50:239–258
25. Ye Y (1997) Interior point algorithms: Theory and analysis. Wiley/Interscience, New York

Powell Method

PM

VASSILIOS S. VASSILIADIS, RAÚL CONEJEROS
Chemical Engineering Department,
University Cambridge, Cambridge, UK

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Gradient-free minimization; Powell method; Line search methods; Pattern search

The Powell method in its basic form can be viewed as a *gradient-free minimization algorithm*. It requires repeated line search minimizations, which may be carried out using *univariate gradient free*, or *gradient based procedures*. It was introduced by M.J.D. Powell [1]. The procedure is described in the algorithm steps below.

The minimization problem considered is:

$$\min_{\mathbf{x}} f(\mathbf{x}) .$$

1. Initialization

Select an accuracy $\epsilon > 0$, and a starting point $\mathbf{x}^{(0)}$. Set the initial search directions $\mathbf{s}^{(i)}$ to be the unit vectors along each coordinate axis, for $i = 1, \ldots, n$. Set the main iteration counter to $k = 0$, and the cycle counter $i = 1$. Initialize $\mathbf{z}^{(1)} = \mathbf{x}^{(0)}$. Set counter $j = 0$ for the case where step 2.2a is used.

2. Directional univariate minimization

2.1 Determine a univariate minimizer λ_i^* for the problem $f(\mathbf{z}^{(i)} + \lambda_i \mathbf{s}^{(i)})$.
Set $\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} + \lambda_i^* \mathbf{s}^{(i)}$, and increment $i \leftarrow i + 1$.
Repeat step 2.1 until $i = n + 1$.
Check for termination (i.e. use the criterion in step 3)
Go to step 2.2a if the original version is desired, or
go to step 2.2b if the variant avoiding linearly dependent directions is chosen

2.2a New direction selection (pattern search directions)
(Original version of the method)
Set $j \leftarrow j + 1$.
IF $j \leq n$
THEN replace $\mathbf{s}^{(j)}$ by $\mathbf{z}^{(n+1)} - \mathbf{x}^{(k)}$.
ELSE reset search direction set to the coordinate directions.
END IF
Set $\mathbf{x}^{(k+1)} = \mathbf{z}^{(n+1)}$;
initialize $\mathbf{z}^{(1)} = \mathbf{x}^{(k+1)}$;
increment counter $k \leftarrow k + 1$;
set $i = 1$, go to step 2.1.

2.2b New direction selection
(Modified variant to avoid linearly dependent directions)
At the present kth main iteration calculate $\Delta = \max_{1,\ldots,n}\{f(\mathbf{z}^{(i)}) - f(\mathbf{z}^{(i+1)})\}$ and the index m for which this occurs. The corresponding direction for which this occurs is designated as $\mathbf{s}^{(m)}$.
Calculate $f_1 = f(\mathbf{x}^{(k)})$, $f_2 = f(\mathbf{z}^{(n+1)})$, $f_3 = f(2\mathbf{z}^{(n+1)} - \mathbf{x}^{(k)})$.
IF either $f_3 \leq f_1$ and/or $(f - 1 - 2f_2 + f_3)(f_1 - f_2 - \Delta)^2 \geq 0.5\Delta(f_1 - f_2)^2$
THEN
use the old set of directions $[\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(n)}]$;
set $\mathbf{x}^{(k+1)} = \mathbf{z}^{(n+1)}$ or $\mathbf{x}^{(k+1)} = 2\mathbf{z}^{(n+1)} - \mathbf{x}^{(k)}$, whichever yields the lowest value of $f(x)$;
initialize $\mathbf{z}^{(1)} = \mathbf{x}^{(k+1)}$;
increment counter $k \leftarrow k + 1$;
set $i = 1$, go to step 2.1.
ELSE define the search direction (pattern)
$\mathbf{s} = \mathbf{z}^{(n+1)} - \mathbf{x}^{(k)}$;
find the value λ^* minimizing the function $f(\mathbf{z}^{(n+1)} + \lambda\mathbf{s})$.
define the new set of directions $[\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(m-1)}, \mathbf{s}^{(m+1)}, \ldots, \mathbf{s}^{(n)}, \mathbf{s}]$;
set $\mathbf{x}^{(k+1)} = \mathbf{z}^{(n+1)} + \lambda^*\mathbf{s}$;
initialize $\mathbf{z}^{(1)} = \mathbf{x}^{(k+1)}$;
increment counter $k \leftarrow k + 1$;
set $i = 1$, go to step 2.1.
END IF

3. Termination check

A satisfactory termination criterion is generally to stop whenever at any stage of the algorithm the change in the variables is less than the required accuracy, that is when $\| \mathbf{z}^{(n+1)} - \mathbf{x}^{(k)} \| \leq \epsilon$.

In terms of the termination criterion, Powell [1] gives a more elaborate termination check procedure. This is defined by the following steps:

- Specify a set of accuracies, for each variable independently, $\epsilon_1, \ldots, \epsilon_n$, each of which is greater than zero.
- Apply the standard Powell's method until a complete cycle of n directional minimizations causes a change of less than 1/10th of the desired accuracy in the variables, individually. Call the resulting point $\mathbf{y}^{(A)}$.
- Increase every variable by 10 times the corresponding specified accuracy.
- Apply the standard Powell's method from this point until a complete cycle of n directional minimizations causes a change of less than 1/10th of the desired accuracy in the variables, individually. Call the resulting point $\mathbf{y}^{(B)}$.
- Define a search direction $\mathbf{s}^{(AB)} = \mathbf{y}^{(A)} - \mathbf{y}^{(B)}$. Calculate λ^* minimizing $f(\mathbf{y}^{(A)} + \lambda\,\mathbf{s}^{(AB)})$. Set $\mathbf{y}^{(C)} = \mathbf{y}^{(A)} + \lambda\,\mathbf{s}^{(AB)}$.
- Assume that the process has converged if the components (individually) of the vectors $(\mathbf{y}^{(A)} - \mathbf{y}^{(C)})$ and $(\mathbf{y}^{(B)} - \mathbf{y}^{(C)})$ are less than 1/10th of the set accuracies. If this does not hold, proceed to the next step.
- Replace the search direction $\mathbf{s}^{(1)}$ by $(\mathbf{y}^{(A)} - \mathbf{y}^{(C)})$ and restart the procedure from step 2 onwards (present termination control procedure).

The termination procedure proposed above by Powell is expected to be more reliable, but it is more expensive since the entire minimization problem has to be resolved at least twice, until the tight convergence criteria are satisfied.

The method has a quadratic termination property, minimizing quadratic functions in predetermined number of operations, requiring $n^2 + O(n)$ line searches. The directions generated also by the method can be shown to be conjugate (e. g. [2]).

To remedy the case where directions will tend gradually to become linearly dependent (in the original version, step 2.2a) a modification, originally proposed by Powell, is also given in step 2.2b.

See also

- ► Cyclic Coordinate Method
- ► Rosenbrock Method
- ► Sequential Simplex Method

References

1. Powell MJD (1964) An efficient method for finding the minimum of a function of several variables without calculating derivatives. Comput J 7:155–162
2. Rao SS (1984) Optimization theory and applications, 2nd edn. Wiley Eastern, New Delhi

Practical Augmented Lagrangian Methods

ERNESTO G. BIRGIN[1], J. M. MARTÍNEZ[2]
[1] Department of Computer Science, University of São Paulo, São Paulo, Brazil
[2] Department of Applied Mathematics, University of Campinas, Campinas, Brazil

MSC2000: 90C30, 49M30, 49M37, 65K05

Article Outline

Keywords
Introduction
Cases
Augmented Lagrangians and Global Optimization
Augmented Lagrangian Algorithm with Arbitrary Lower-Level Constraints
Augmented Lagrangian Algorithm with Lower-Level Box Constraints
Alternative Augmented Lagrangians
References

Keywords

Augmented Lagrangian; Nonlinear programming; Algorithms; Global convergence; Constraint qualifications; Deterministic global optimization

Introduction

We are concerned with Nonlinear Programming problems defined in the following way:

$$\begin{aligned} &\text{Minimize} && f(x) \\ &\text{subject to} && h(x) = 0 \\ &&& g(x) \leq 0 \\ &&& x \in \Omega\,, \end{aligned} \tag{1}$$

where $h : \mathbb{R}^n \to \mathbb{R}^m$, $g : \mathbb{R}^n \to \mathbb{R}^p$, $f : \mathbb{R}^n \to \mathbb{R}$ are continuous and $\Omega \subset \mathbb{R}^n$ is a closed set. From now on $\|\cdot\|$ represents the Euclidean norm and v_+ means $\max\{0, v\}$. The set $\mathbb{R}_+$ will be the set of nonnegative real numbers.

The Powell–Hestenes–Rockafellar (PHR) Augmented Lagrangian [42,54,56] is given by:

$$L_\rho(x, \lambda, \mu) = f(x) + \frac{\rho}{2}\left\{ \sum_{i=1}^{m} \left[h_i(x) + \frac{\lambda_i}{\rho} \right]^2 + \sum_{i=1}^{p} \left[\left(g_i(x) + \frac{\mu_i}{\rho} \right)_+ \right]^2 \right\}, \tag{2}$$

for all $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^m$, $\mu \in \mathbb{R}^p_+$.

PHR-based Augmented Lagrangian methods for solving (1) are based on the iterative (approximate) minimization of L_ρ with respect to $x \in \Omega$, followed by the updating of the penalty parameter ρ and the Lagrange multipliers approximations λ and μ. The most popular practical Augmented Lagrangian method gave rise to the LANCELOT package [24,25,26]. LANCELOT does not use inequality constraints $g(x) \leq 0$ in its problem formulation. When an inequality constraint $g_i(x) \leq 0$ appears in a practical problem, it is replaced by $g_i(x) + s_i = 0, s_i \geq 0$. The convergence of the LANCELOT algorithm to KKT points was proved in [24] using regularity assumptions. Under weaker assumptions that involve the Constant Positive Linear Dependence (CPLD) constraint qualification [4,55], KKT-convergence was proved in [2] for a variation of the LANCELOT method. In the original LANCELOT method Ω was a box. A generalization where Ω is a polytope may be found in [23].

The motivation of (2) comes from the classical External Penalty method [27,34,36]. In this method one minimizes the function given by

$$\Phi_\rho(x) = f(x) + \frac{\rho}{2}\left[\sum_{i=1}^{m} h_i(x)^2 + \sum_{i=1}^{p} [g_i(x)_+]^2 \right], \tag{3}$$

for successive values of ρ that tend to infinity. If, after minimizing (3) for a given ρ, a satisfactory feasibility is not achieved, the External Penalty philosophy leads to increase the value of ρ. If ρ is very large, the problem of minimizing Φ_ρ may become very difficult for ordinary minimization solvers.

The Augmented Lagrangian philosophy is different. Assume that the result of minimizing Φ_ρ, for a given ρ is not satisfactory, in terms of feasibility. Then, instead of increasing ρ (or, perhaps, besides increasing ρ) one modifies the origin with respect to which infeasibility

is penalized. For example, suppose that, after the minimization of Φ_ρ we obtain x such that $h_i(x) = c \neq 0$. A common sense conjecture would be that this "disappointing" result was obtained because we punished the deviation of $h_i(x)$ with respect to 0, whereas the correct strategy would be to punish the deviation with respect to $-c$. This leads to the Shifted Penalty idea, in which, instead of Φ_ρ, one uses the Shifted Penalty function:

$$\Phi_\rho(x, c, d) = f(x) + \frac{\rho}{2}\Bigg[\sum_{i=1}^{m}(h_i(x) + c_i)^2 + \sum_{i=1}^{p}[(g_i(x) + d_i)_+\Bigg]^2. \quad (4)$$

Writing $c_i = \lambda_i/\rho$ and $d_i = \mu_i/\rho$ we observe that the Shifted Penalty strategy coincides with the Augmented Lagrangian one. The naive modification of the shifts c_i, d_i sketched above gives rise to the best known (first-order) updating formula for the Lagrange multipliers in the Augmented Lagrangian method. It is interesting to observe that this intuitive reasoning is independent of the smoothness of f, h and g. In this article we give preference to matrix-free updating procedures, which excludes the consideration of higher order estimates [28,35].

In [3] a new PHR-like algorithm was introduced that does not use slack variables to complete inequality constraints and admits general constraints in the lower-level set Ω. In the box-constraint case, subproblems are solved using a technique introduced in [13], which improves the GENCAN algorithm [12]. CPLD-based convergence and penalty-parameter boundedness were also proved in [3] under suitable conditions on the problem.

In addition to its intrinsic adaptability to the case in which arbitrary constraints are included in Ω, the following positive characteristics of the Augmented Lagrangian approach for solving (1) must be mentioned:

1. Augmented Lagrangian methods proceed by sequential resolution of simple problems. Progress in the analysis and implementation of simple-problem optimization procedures produces an almost immediate positive effect on the effectiveness of Augmented Lagrangian algorithms. Box-constrained optimization is a dynamic area of practical optimization from which we can expect Augmented Lagrangian improvements.
2. Global minimization of the subproblems implies convergence to global minimizers of the Augmented Lagrangian method [11]. There is a large field for research on global optimization methods for box-constraint optimization. When the global box-constraint optimization problem is satisfactorily solved in practice, the effect on the associated Augmented Lagrangian method for the Nonlinear Programming problem is immediate.
3. Most box-constrained optimization methods are guaranteed to find stationary points. In practice, good methods do more than that. The line-search procedures of [12], for example, include extrapolation steps that are not necessary at all from the point of view of KKT convergence. However, they enhance the probability of convergence to global minimizers. As a consequence, the probability of convergence to Nonlinear Programming global minimizers of a practical Augmented Lagrangian method is enhanced too.
4. The global convergence theory of Augmented Lagrangian methods [11] does not need differentiability of the functions that define the Nonlinear Programming problem. In practice, this indicates that the Augmented Lagrangian approach may be successful in situations were smoothness is dubious.
5. The Augmented Lagrangian approach can be adapted to the situation in which analytic derivatives are not computed. See [47] for a derivative-free version of LANCELOT.
6. In many practical problems the Hessian of the Lagrangian is structurally dense (in the sense that any entry may be different from zero at different points) but generally sparse (given a specific point in the domain, the particular Lagrangian Hessian is a sparse matrix). As an example of this situation, consider the following formulation [18,19] of the problem of fitting circles of radii r within a circle of radius R without overlapping:

$$\text{Min} \sum_{i<j} \max\left\{0, 4r^2 - \|p_i - p_j\|_2^2\right\}^2$$
$$\text{subject to } \|p_i\|_2^2 \leq (R - r)^2.$$

The Hessian of the objective function is structurally dense but sparse at any point such that points p_i are "well distributed" within the big circle, since

only pairs of overlapping small circles appear in the expression of the objective function. Newtonian methods usually have difficulties with this situation, both in terms of memory and computer time since the sparsity pattern of the matrix changes from iteration to iteration. This difficulty is almost irrelevant for the Augmented Lagrangian approach if one uses a low-memory box-constraint solver.

7. Independently of the Lagrangian Hessian density, the structure of the KKT system may be very poor for sparse factorizations. This is a serious difficulty for Newton-based methods but not for suitable implementations of the Augmented Lagrangian PHR algorithm.
8. If the Nonlinear Programming problem has many inequality constraints the usual slack-variable approach of Interior-Point methods (also used in [2,24]) may be inconvenient. There are several approaches to reduce the effect of the presence of many slacks, but they may not be as effective as not using slacks at all.
9. Huge problems have obvious inconvenients in terms of storage requirements. The Augmented Lagrangian approach provides a radical remedy: problem data may be computed "on the flight", used when required in the subproblem, and not stored at all. This is not possible if one uses matricial approaches, independently of the sparsity strategy adopted.
10. If, at the solution of the problem, some strong constraint qualification fails to hold, the performance of Newton-like algorithms could be severely affected. The Augmented Lagrangian is not as sensitive to this type of inconvenient.

The amount of research dedicated to Augmented Lagrangian methods decreased in the 21th century. Modern methods, based on interior-point techniques, sequential quadratic programming, trust regions, restoration, nonmonotone strategies and advanced sparse linear algebra procedures attracted much more attention.

A theoretical reason, and its practical consequence, seems to be behind this switch of interest. Roughly speaking, under suitable assumptions, Interior-Point Newtonian techniques converge quadratically (or, at least, superlinearly) whereas practical Augmented Lagrangian generally converge only linearly. Therefore, if both methods converge to the same point, and the precision required is strict enough, an Interior-Point Newtonian method will require less computer time than an Augmented Lagrangian method, independently of the work per iteration. Several attempts have been made to alleviate both the slow-convergence behavior as the ill-conditioning of the subproblems [14,21,32,33,39,49]. Behind these attempts is the fact that the optimality conditions of the Augmented Lagrangian (and Penalty) subproblems may be decomposed in such a way that, for ρ large, resemble the KKT conditions of the original problem. This fact may be exploited in several ways and makes it possible that good implementations of the Augmented Lagrangian method be quite competitive with Interior-Point Newtonian techniques, even when high precision is the main requirement at the solution.

The general form of the Augmented Lagrangian method based on the PHR formula considered in this article is the following.

Algorithm 1

Let $\lambda_{\min} < \lambda_{\max}$, $\mu_{\max} > 0$, $\gamma > 1$, $0 < \tau < 1$. Let $\{\varepsilon_k\}$ be a sequence of nonnegative numbers such that $\lim_{k\to\infty} \varepsilon_k = 0$. Let $\lambda_i^1 \in [\lambda_{\min}, \lambda_{\max}]$, $i = 1, \ldots, m$, $\mu_i^1 \in [0, \mu_{\max}]$, $i = 1, \ldots, p$, and $\rho_1 > 0$. Initialize $k \leftarrow 1$.

Step 1. *Solving the subproblem.*
Compute $x^k \in \mathbb{R}^n$ an approximate solution of

$$\text{Minimize } L_{\rho_k}(x, \lambda^k, \mu^k) \text{ subject to } x \in \Omega\,. \quad (5)$$

Step 2. Define

$$V_i^k = \max\left\{g_i(x^k), -\frac{\mu_i^k}{\rho_k}\right\}, i = 1, \ldots, p\,.$$

If $k = 1$ or

$$\max\left\{\|h(x^k)\|_\infty, \|V^k\|_\infty\right\} \leq \tau \max\left\{\|h(x^{k-1})\|_\infty, \|V^{k-1}\|_\infty\right\}\,, \quad (6)$$

define $\rho_{k+1} = \rho_k$. Otherwise, define $\rho_{k+1} = \gamma\rho_k$.

Step 3. Compute $\lambda_i^{k+1} \in [\lambda_{\min}, \lambda_{\max}]$, $i = 1, \ldots, m$ and $\mu_i^{k+1} \in [0, \mu_{\max}]$, $i = 1, \ldots, p$. Set $k \leftarrow k + 1$ and go to Step 1.

In the practical implementation of Algorithm 1, we will compute $\lambda_i^{k+1} = \min\{\max\{\lambda_{\min}, \lambda_i^k + \rho_k h_i(x^k)\}, \lambda_{\max}\}$ and $\mu_i^{k+1} = \min\{\max\{0, \mu_i^k + \rho_k g_i(x^k)\}, \mu_{\max}\}$. These are safeguarded first-order estimations of the Lagrange multipliers. The safeguards defined by $\lambda_{\min}, \lambda_{\max}$ and $\mu_{\max}$ are necessary to prove the global convergence results. Some authors prefer to define Augmented Lagrangian algorithms without safeguards for the Lagrange multipliers [9,26]. However, boundedness of the multiplier estimates are necessary to prove the main convergence results and, if this boundedness is not algorithmically forced, it may be guaranteed only by means of strong problem assumptions.

Different Augmented Lagrangian algorithms will differ only in Step 1. In each case we will need a precise definition for the approximate solution of (5).

Cases

Augmented Lagrangians and Global Optimization

In this section we will only assume continuity of the functions f, h and g. Throughout the section we will assume that a global minimizer of (1) exists. Several versions of the Augmented Lagrangian method generate sequences that converge to global minimizers, provided that global minimizers of the subproblems are available. This property is inherited from the analogous property of the External Penalty method. A practical consequence of this property is the fact that Augmented Lagrangian methods tend to find feasible points with lower objective function values than other nonlinear programming solvers, when clever agressive algorithms are used for solving the subproblems.

Here we will assume we are able to find an approximate global minimizer defined by the tolerance ε_k. At each iteration, x^k will belong to $\Omega \cap P_k$, where P_k is an auxiliary set to which a global minimizer of (1) necessarily belongs. For example, P_k may be a set that contains the feasible region of (1). The presence of the constraints defined by P_k helps in the global resolution of the subproblems. Obviously, in the absence of algorithmic advantages, P_k may be defined as being $\mathbb{R}^n$. Algorithm 2 will be Algorithm 1, where Step 1 is defined as follows.

Step 1. Let $P_k \subset \mathbb{R}^n$ be a closed set such that a global minimizer z (the same for all k) belongs to P_k. Find an ε_k-global minimizer x^k of the problem Min $L_{\rho_k}(x, \lambda^k, \mu^k)$ subject to $x \in \Omega \cap P_k$. That is $x^k \in \Omega \cap P_k$ is such that:

$$L_{\rho_k}(x^k, \lambda^k, \mu^k) \leq L_{\rho_k}(x, \lambda^k, \mu^k) + \varepsilon_k\,, \qquad (7)$$

for all $x \in \Omega \cap P_k$. The ε_k-global minimum can be obtained using a deterministic global optimization approach, such as the αBB method [37].

In most deterministic global optimization methods for solving (7) the point x^{k-1} is not used as "initial approximation" as most local optimization solvers do. In fact, the concept of "initial point" has no meaning at all in this case. The information used by the Outer iteration k is the set of approximate Lagrange multipliers computed after iteration $k-1$, and nothing else.

Theorems 2 [11] is the main convergence result related to Algorithm 2. Limit points of sequences generated by this algorithm are feasible global minimizers.

Theorem 2 *Assume that the sequence $\{x^k\}$ is well defined and admits a limit point x^*. Then, x^* is a global minimizer of (1). If, instead of $\varepsilon_k \to 0$ we assume only that $\varepsilon_k \to \varepsilon \geq 0$, x^* will be feasible and $f(x^*) \leq f(x) + \varepsilon$ for all feasible x.*

The problem of finding $x^k \in \Omega \cap P_k$ satisfying (7) consists of finding an ε_k-global solution of the problem:

$$\text{Minimize } L_{\rho_k}(x, \lambda^k, \mu^k) \text{ subject to } x \in \Omega \cap P_k\,. \quad (8)$$

When Ω and P_k are defined by linear equality and/or inequality constraints and f, h, g admit continuous second derivatives, this problem has been solved in [11] using the αBB algorithm [37].

The practical results presented in [11] corroborate the theory and give hints on the effectivity of the Augmented Lagrangian method for global optimization.

Augmented Lagrangian Algorithm with Arbitrary Lower-Level Constraints

In [2,3] safeguarded Augmented Lagrangian methods were defined that converge to KKT points under the CPLD constraint qualification and exhibit good properties in terms of penalty parameter boundedness. ALGENCAN, which is publicly available in the TANGO Project web page http://www.ime.usp.br/~egbirgin/tango/, is the application of the main algorithm in [3] to problem (1).

In this section we will assume that f, h, g admit continuous first (and, sometimes, second) derivatives. Observe that the function L_ρ, defined in (2) has continuous first derivatives with respect to x, but second derivatives are not defined at the points such that $g_i(x) + \mu_i/\rho = 0$.

In Algorithm 3 we will assume that the set Ω is defined by

$$\Omega = \{x \in \mathbb{R}^n \mid \underline{h}(x) = 0, \underline{g}(x) \leq 0\}, \tag{9}$$

where $\underline{h} : \mathbb{R}^n \to \mathbb{R}^{\underline{m}}, \underline{g} : \mathbb{R}^n \to \mathbb{R}^{\underline{p}}$ are as smooth as necessary. The constraints defined by Ω are called *lower-level* constraints. Algorithm 3 is identical to Algorithm 1 except at Step 1. The subproblem resolution at Algorithm 3 is as given below.

Step 1. Compute (if possible) $x^k \in \mathbb{R}^n$ such that there exist $v^k \in \mathbb{R}^{\underline{m}}, u^k \in \mathbb{R}^{\underline{p}}$ satisfying

$$\|\nabla L_{\rho_k}(x^k, \lambda^k, \mu^k) + \sum_{i=1}^{\underline{m}} v_i^k \nabla \underline{h}_i(x^k) + \sum_{i=1}^{\underline{p}} u_i^k \nabla \underline{g}_i(x^k)\| \leq \varepsilon_k, \tag{10}$$

$$u_i^k \geq 0, \underline{g}_i(x^k) \leq \varepsilon_k \text{ for all } i = 1, \ldots, \underline{p}, \tag{11}$$

$$\underline{g}_i(x^k) < -\varepsilon_k \Rightarrow u_i^k = 0 \text{ for all } i = 1, \ldots, \underline{p}, \tag{12}$$

$$\|\underline{h}(x^k)\| \leq \varepsilon_k. \tag{13}$$

The conditions (10)–(13) are approximate KKT conditions for the minimization of L_{ρ_k} subject to the lower level constraints. If $\Omega = \mathbb{R}^n$ these conditions reduce to $\|\nabla L_{\rho_k}(x^k, \lambda^k, \mu^k)\| \leq \varepsilon_k$.

The CPLD (Constant Positive Linear Dependence) condition defined by Qi and Wei [55] is a crucial tool in the convergence theory of Algorithm 3. In [4] it has been proved that CPLD is a constraint qualification and its relation with other constraint qualifications have been reported.

A First-Order Constraint Qualification is a property of feasible points of a Nonlinear Programming problem such that, when verified at a local minimizer, implies that the local minimizer is a KKT point. The Linear-Independence Constraint Qualification (LICQ), also called *regularity*, says that the gradients of the active constraints at the feasible point x are linearly independent.

Assume that the feasible set of a nonlinear programming problem is given by $\overline{h}(x) = 0, \overline{g}(x) \leq 0$, where $\overline{h} : \mathbb{R}^n \to \mathbb{R}^{\overline{m}}$ and $\overline{g} : \mathbb{R}^n \to \mathbb{R}^{\overline{p}}$. Let $I(x) \subset \{1, \ldots, \overline{p}\}$ be the set of indices of the active inequality constraints at the feasible point x. Let $I_1 \subset \{1, \ldots, \overline{m}\}$, $I_2 \subset I(x)$. The subset of gradients of active constraints that correspond to the indices $I_1 \cup I_2$ is said to be *positively linearly dependent* if there exist multipliers λ, μ such that

$$\sum_{i \in I_1} \lambda_i \nabla \overline{h}_i(x) + \sum_{i \in I_2} \mu_i \nabla \overline{g}_i(x) = 0, \tag{14}$$

with $\mu_i \geq 0$ for all $i \in I_2$ and $\sum_{i \in I_1} |\lambda_i| + \sum_{i \in I_2} \mu_i > 0$. Otherwise, we say that these gradients are *positively linearly independent*.

The Mangasarian–Fromovitz Constraint Qualification MFCQ says that, at the feasible point x, the gradients of the active constraints are positively linearly independent [48,57].

The CPLD Constraint Qualification says that, if a subset of gradients of active constraints is positively linearly dependent at the feasible point x (i. e. (14) holds), then there exists $\delta > 0$ such that the vectors

$$\{\nabla \overline{h}_i(y)\}_{i \in I_1}, \{\nabla \overline{g}_j(y)\}_{j \in I_2}$$

are linearly dependent for all $y \in \mathbb{R}^n$ such that $\|y - x\| \leq \delta$.

The main convergence theorems related to Algorithm 3 were proved in [3]. Theorem 3 says that, if a limit point satisfies the CPLD condition with respect to the lower-level constraints, then this point is stationary with respect to a natural infeasibility measure. In other words, this theorem says that, if the limit point is not feasible, then (very likely) it is a local minimizer of the upper-level infeasibility, subject to lower-level feasibility.

Theorem 3 *Let $\{x^k\}$ be a sequence generated by Algorithm 3. Let x^* be a limit point of $\{x^k\}$. Then, if the sequence of penalty parameters $\{\rho_k\}$ is bounded, the limit point x^* is feasible. Otherwise, at least one of the following possibilities hold:*

(i) x^* is a KKT point of the problem

$$\text{Minimize } \frac{1}{2}\left[\sum_{i=1}^{m} h_i(x)^2 + \sum_{i=1}^{p}[g_i(x)_+]^2\right] \quad (15)$$
$$\text{subject to } x \in \Omega.$$

(ii) x^* does not satisfy the CPLD constraint qualification associated with Ω.

From the point of view of optimality, we are interested in the status of feasible limit points. Theorem 4 says that, under the CPLD constraint qualification, feasible limit points are stationary (KKT) points of the original problem. Since CPLD is strictly weaker than the Mangasarian–Fromovitz (MF) constraint qualification, it turns out that Theorem 4 is stronger than results where KKT conditions are proved under MF or regularity assumptions.

Theorem 4 *Let $\{x^k\}$ be a sequence generated by Algorithm 3. Assume that x^* is a feasible limit point of (1)–(9) that satisfies the CPLD constraint qualification related to set of all the constraints. Then, x^* is a KKT point of the problem (1)–(9).*

Theorems 3 and 4 are interesting and useful but they do not explain why it is better to use the Augmented Lagrangian instead of a pure penalty method. In fact, if we define $\lambda^k = 0, \mu^k = 0$ for all k these two theorems remain valid and we are in presence of a variation of the External Penalty method. The use of Lagrange multipliers estimates is justified in Theorem 5, which is also proved in [3]. Theorem 5 says that, under appropriate conditions, the sequence of penalty parameters $\{\rho_k\}$ do not tend to infinity. In practice, this means that the minimization subproblems tend to remain well-conditioned and that minimization algorithms for solving the subproblems will not face difficulties associated to very large values of ρ_k.

Theorem 5 *Assume that:*

1. *The sequence $\{x^k\}$ is generated by the application of Algorithm 3 to problem (1)–(9) and $\lim_{k\to\infty} x^k = x^*$.*
2. *In Algorithm 3 we use the updating rules $\lambda_i^{k+1} = \max\{\lambda_{\min}, \min\{\lambda_i^k + \rho_k h_i(x^k), \lambda_{\max}\}\}$ and $\mu_i^{k+1} = \max\{0, \min\{\mu^k + \rho_k g_i(x^k), \mu_{\max}\}\}$.*
3. *The point x^* is feasible ($h(x^*) = 0$, $\underline{h}(x^*) = 0$, $g(x^*) \le 0$ and $\underline{g}(x^*) \le 0$).*
4. *The gradients of the active constraints at x^* are linearly independent. The associated (unique) Lagrange multipliers are $\lambda^*, \mu^*, u^*, v^*$.*
5. *The functions $f, h, g, \underline{h}$ and $\underline{g}$ admit continuous second derivatives in a neighborhood of x^*.*
6. *Define the tangent subspace T as the set of all $z \in \mathbb{R}^n$ such that $\nabla h(x^*)^T z = \nabla \underline{h}(x^*)^T z = 0$, $\nabla g_i(x^*)^T z = 0$ for all i such that $g_i(x^*) = 0$ and $\nabla \underline{g}_i(x^*)^T z = 0$ for all i such that $\underline{g}_i(x^*) = 0$. Then, for all $z \in T, z \neq 0$,*

$$z^T\Big[\nabla^2 f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla^2 h_i(x^*) + \sum_{i=1}^{p} \mu_i^* \nabla^2 g_i(x^*) + \sum_{i=1}^{\underline{m}} v_i^* \nabla^2 \underline{h}_i(x^*) + \sum_{i=1}^{\underline{p}} u_i^* \nabla^2 g_i(x^*)\Big] z > 0\,.$$

7. *For all $i = 1, \ldots, m$, $j = 1, \ldots, p$, $\lambda_i^* \in (\lambda_{\min}, \lambda_{\max})$, $\mu_j^* \in [0, \mu_{\max})$.*
8. *For all i such that $g_i(x^*) = 0$, we have $\mu_i^* > 0$. (Strict complementarity in the upper level.)*
9. *There exists a sequence $\eta_k \to 0$ such that $\varepsilon_k \le \eta_k \max\{\|h(x^k)\|, \|V^k\|\}$ for all $k = 0, 1, 2 \ldots$*

Then, the sequence of penalty parameters $\{\rho_k\}$ is bounded.

Observe that strict complementarity is imposed only to the constraints in the upper-level set. In the lower-level set it is admissible that $\underline{g}_i(x^*) = u_i^* = 0$. Observe, too, that the assumption on the reduced positive definiteness on the Hessian of the Lagrangian is weaker than the usual second-order sufficiency assumption [36], since the subspace T is orthogonal to the gradients of *all* active constraints, and no exception is made with respect to active constraints with null multiplier u_i^*. In fact, this is not a second-order sufficiency assumption for local minimizers. It holds for the problem of minimizing $x_1 x_2$ subject to $x_2 - x_1 \le 0$ at $(0, 0)$ although $(0, 0)$ is not a local minimizer of this problem.

The last hypothesis of Theorem 5 imposes that the precision in which subproblems are solved should tend to zero faster than the measure of infeasibility-noncomplementarity. Some authors [30,31,40,41], in slightly different contexts, also used convergence toler-

ances that depend on the degree of infeasibility of the current inner iterate.

The Augmented Lagrangian method is the only efficient nonlinear programming algorithm that can take obvious advantage of the existence of case-oriented optimization solvers for problems whose constraints are a subset of the original problem constraints. The partition of the constraints in "easy" and "complicate" is very common in engineering applications. In the Augmented Lagrangian framework, easy constraints go to the lower level and complicate constraints contribute to the Augmented Lagrangian function. The most common situation is the one in which lower level constraints are linear. Location problems of this type are described in [3]. Problems with more than 3×10^6 variables and 14×10^6 constraints are solved in this way, using moderate computer time. The codes are free for download through the TANGO Project web page http://www.ime.usp.br/~egbirgin/tango/. The key for the efficiency of the Augmented Lagrangian method in these problems is the use of the Spectral Projected Gradient method [15,16,17] for solving the subproblems.

Augmented Lagrangian Algorithm with Lower-Level Box Constraints

In most applications, the definition of the lower level set Ω in (1) is:

$$\Omega = \{x \in \mathbb{R}^n \mid a \leq x \leq b\}, \tag{16}$$

where $a, b \in \mathbb{R}^n$, $a \leq b$. In other words, Ω is an n-dimensional box. By the continuity of the Augmented Lagrangian function and the compactness of Ω, this definition guarantees that a global minimizer of the subproblem exists. Many times one adds bound constraints in the lower level of a nonlinear programming problem in order to guarantee solubility of the subproblems and boundedness of the sequence $\{x^k\}$.

Obviously, the constraints (16) may be written in the form (9) and, so, Algorithm 3 may be applied and Theorems 3, 4 and 5 hold. However, many specific algorithms for box-constrained optimization exist that use stronger convergence criteria than the one given in (10)–(13). Namely, in box-constrained minimization one usually declares convergence when

$$x^k \in \Omega \text{ and } \|P_\Omega(x^k - \nabla L_{\rho_k}(x^k, \lambda^k, \mu^k)\|_k \leq \varepsilon_k, \tag{17}$$

where P_Ω denotes de Euclidean projection on Ω. The condition (17) implies (10)–(13). This leads us to define Algorithm 4 as Algorithm 1 where Step 1 is defined by (17). Theorems 3, 4 and 5 obviously apply to Algorithm 4. It must be observed, however, that, since all the points of Ω satisfy CPLD, Theorem 3 guarantees that the limit points of the generated sequence $\{x^k\}$ are KKT points of $\sum_{i=1}^m h_i(x)^2 + \sum_{i=1}^p [g_i(x)_+]^2$ subject to $x \in \Omega$.

Algorithm 4, with the subproblems solved by the box-constraint solver GENCAN [12], with the modifications introduced in [13], is called ALGENCAN. The code that implements ALGENCAN is free for download in the TANGO Project web page http://www.ime.usp.br/~egbirgin/tango/. It is written in Fortran 77 (double precision) and interfaces with AMPL, CUTEr, C/C++, Python and R (language and environment for statistical computing) are available.

The default version of ALGENCAN uses $\tau = 0.5$, $\gamma = 10$, $\lambda_{\min} = -10^{20}$, $\mu_{\max} = \lambda_{\max} = 10^{20}$, $\varepsilon_k = 10^{-4}$ for all k, $\lambda_1 = 0$, $\mu_1 = 0$ and $\rho_1 = \max\{10^{-6}, \min\{10, (2|f(x^0)|)/(\|h(x^0)\|^2 + \|g(x^0)_+\|^2\}\}$. The default convergence criterion is $\max\{\|h(x^k)\|_\infty, \|V^k\|_\infty\} \leq 10^{-4}$. The condition $\|V^k\|_\infty \leq 10^{-4}$ guarantees that, for all $i = 1, \ldots, p$, $g_i(x^k) \leq 10^{-4}$ and that $(\mu_i^k + \rho_k g_i(x^k))_+ = 0$ whenever $g_i(x^k) < -10^{-4}$. This means that, approximately, feasibility and complementarity hold at the final point. Dual feasibility with tolerance 10^{-4} is guaranteed by (17) and the choice of ε_k.

The celebrated package LANCELOT also solves the basic Nonlinear Programming problem with box constraints, but each inequality constraint is completed with a slack variable to become an equality constraint plus a lower-level bound.

A comparison between the default versions of ALGENCAN and LANCELOT B using all the (1023) problems of the CUTEr collection was reported in [3]. The executions were stopped when the precision 10^{-4} was achieved or when the allowed CPU time (5 min on an 1.8 GHz AMD Opteron 244 processor, 2 Gb of RAM memory and Linux operating system) was exhausted. Codes are in Fortran 77 and the compiler option "–O" was adopted.

Given a fixed problem, for each method $M \in$ {LANCELOT, ALGENCAN}, x_{final}^M was defined in [3] as the final point obtained by M when solving the given problem.

Practical Augmented Lagrangian Methods, Table 1
Efficiency means number of times that method M obtained the best r^M. Robustness means the number of times in which $r^M < \infty$

	ALGENCAN	LANCELOT B
Efficiency	692	544
Robustness	809	763

The point x^M_{final} is considered to be *feasible* if

$$\max\left\{\left\|h\left(x^M_{final}\right)\right\|_\infty, \left\|g\left(x^M_{final}\right)_+\right\|_\infty\right\} \le 10^{-4}.$$

Define

$$f_{best} = \min_M \left\{f\left(x^M_{final}\right) \mid x^M_{final} \text{ is feasible}\right\}.$$

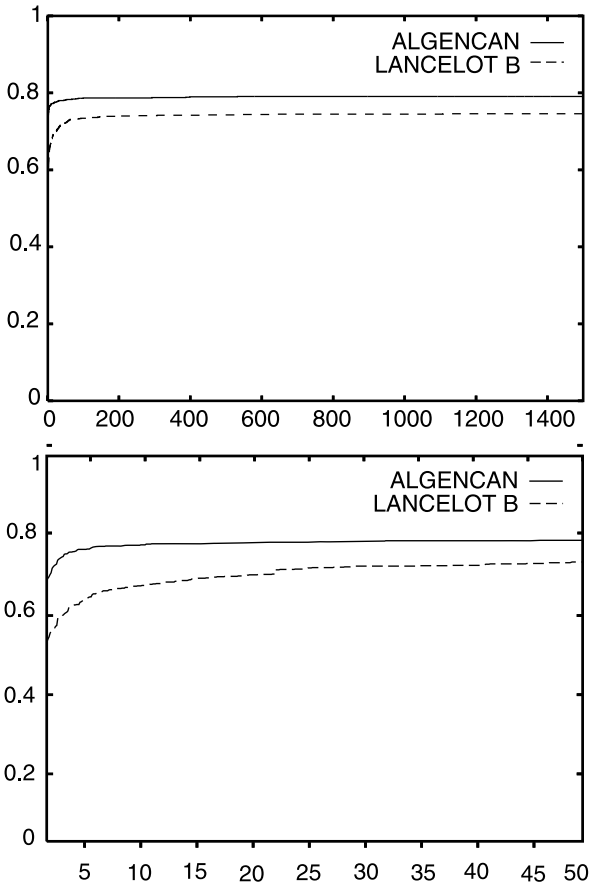
It is said that the method M *found a solution* of the problem if x^M_{final} is feasible and

$$f\left(x^M_{final}\right) \le f_{best} + 10^{-3}|f_{best}| + 10^{-6}$$
$$\text{or} \quad \max\left\{f_{best}, f\left(x^M_{final}\right)\right\} \le -10^{20}.$$

Finally, let t^M be the computer CPU time that method M used to arrive to x^M_{final}. Define

$$r^M = \begin{cases} t^M, \text{ if method M found a solution,} \\ \infty, \text{ otherwise.} \end{cases}$$

The quantity r was used as performance measurement in [3]. The results of the comparison are reported in the form of performance profiles [29] and a small numerical table. See Fig. 1 and Table 1.[1]



Practical Augmented Lagrangian Methods, Figure 1
Performance profiles of ALGENCAN and LANCELOT B in the problems of the CUTER collection. Note that there is a CPU time limit of 5 min for each pair method/problem. The second graphic is a zoom of the left-hand side of the first one

Alternative Augmented Lagrangians

The main drawback of the PHR formula (2) is that the objective function of the subproblems is not twice continuously differentiable. This is the main motivation for the introduction of many alternative Augmented Lagrangian methods. See, for example, [1,5,6,7,8,22,38,43,45,46,50,51,52,53,58]. Most of them have interesting interpretations as proximal point methods for solving the dual problem, when the original nonlinear programming problem is convex [44]. In [10] a comparison of many different Augmented Lagrangian formulae within an algorithmic framework similar to the one of Algorithm 1 has been performed using the CUTE collection [20]. In general, the PHR formula seems to be more efficient than the alternative ones for the resolution of the selected problems.

[1]When LANCELOT B solves a feasibility problem (problem with constant objective function), it minimizes the squared infeasibility instead of addressing the original problem. As a result, it sometimes finishes without satisfying the user required stopping criteria (feasibility and optimality tolerances on the the original problem). In 35 feasibility problems, LANCELOT B stopped declaring convergence but the user-required feasibility tolerance was not satisfied at the final iterate. 16 of the 35 problems seem to be problems in which LANCELOT B converged to a stationary point of the infeasibility (large objective function value of the reformulated problem). In the remaining 19 problems (less than 2% of the total number of problems), LANCELOT B seems to stop prematurely. This easy-to-solve inconvenient may slightly deteriorate the robustness of LANCELOT B reported here.

References

1. Allran RR, Johnsen SEJ (1970) An algorithm for solving nonlinear programming problems subject to nonlinear inequality constraints. Comput J 13:171–177
2. Andreani R, Birgin EG, Martínez JM, Schuverdt ML (2008) Augmented Lagrangian methods under the Constant Positive Linear Dependence constraint qualification. Math Program 111:5–32
3. Andreani R, Birgin EG, Martínez JM, Schuverdt ML (2007) On Augmented Lagrangian methods with general lower-level constraints. SIAM J Optim 18:1286–1309
4. Andreani R, Martínez JM, Schuverdt ML (2005) On the relation between the Constant Positive Linear Dependence condition and quasinormality constraint qualification. J Optim Theory Appl 125:473–485
5. Auslender A, Teboulle M, Ben-Tiba S (1999) Interior proximal and multiplier methods based on second order homogeneous kernels. Math Oper Res 24:645–668
6. Ben-Tal A, Yuzefovich I, Zibulevsky M (1992) Penalty/barrier multiplier methods for minimax and constrained smooth convex programs. Research Report 9/92, Optimization Laboratory, Faculty of Industrial Engineering Management, Technion, Haifa
7. Ben-Tal A, Zibulevsky M (1997) Penalty/barrier multiplier methods for convex programming problems. SIAM J Optim 7:347–366
8. Bertsekas DP (1982) Constrained optimization and Lagrange multiplier methods. Academic Press, New York
9. Bertsekas DP (1999) Nonlinear Programming, 2nd ed, Athena Scientific, Belmont
10. Birgin EG, Castillo R, Martínez JM (2005) Numerical comparison of Augmented Lagrangian algorithms for nonconvex problems. Comput Optim Appl 31:31–56
11. Birgin EG, Floudas CA, Martínez JM (2006) Global minimization using an Augmented Lagrangian method with variable lower-level constraints, available in Optimization Online, E-Print ID: 2006-12-1544, http://www.optimization-online.org/DB_HTML/2006/12/1544.html
12. Birgin EG, Martínez JM (2002) Large-scale active-set box-constrained optimization method with spectral projected gradients. Comput Optim Appl 23:101–125
13. Birgin EG, Martínez JM (2008) Structured minimal-memory inexact quasi-Newton method and secant preconditioners for Augmented Lagrangian Optimization. Comput Optim Appl 39:1–16
14. Birgin EG, Martínez JM, Improving ultimate convergence of an Augmented Lagrangian method. Optim Methods Softw, DOI: 10.1080/10556780701577730
15. Birgin EG, Martínez JM, Raydan M (2000) Nonmonotone spectral projected gradient methods on convex sets. SIAM J Optim 10:1196–1211
16. Birgin EG, Martínez JM, Raydan M (2001) Algorithm 813: SPG – Software for convex-constrained optimization. ACM Trans Math Softw 27:340–349
17. Birgin EG, Martínez JM, Raydan M (2003) Inexact Spectral Projected Gradient methods on convex sets. IMA J Numer Anal 23:539–559
18. Birgin EG, Martínez JM, Ronconi DP (2005) Optimizing the Packing of Cylinders into a Rectangular Container: A Nonlinear Approach. Eur J Oper Res 160:19–33
19. Birgin EG, Sobral FNC (2008) Minimizing the object dimensions in circle and sphere packing problems. Comput Oper Res 35:2357–2375
20. Bongartz I, Conn AR, Gould NIM, Toint PL (1995) CUTE: constrained and unconstrained testing environment. ACM Trans Math Softw 21:123–160
21. Broyden CG, Attia NF (1988) Penalty functions, Newton's method, and quadratic programming. J Optim Theory Appl 58:377–385
22. Castillo RA (1998) Métodos de Lagrangiano Aumentado usando penalidades generalizadas para programação não linear, Tese de Doutorado, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro
23. Conn AR, Gould NIM, Sartenaer A, Toint L, Ph (1996) Convergence properties of an Augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints. SIAM J Optim 6:674–703
24. Conn AR, Gould NIM, Toint PL (1991) A globally convergent Augmented Lagrangian algorithm for optimization with general constraints and simple bounds. SIAM J Numer Anal 28:545–572
25. Conn AR, Gould NIM, Toint PL (1992) LANCELOT: A Fortran package for large scale nonlinear optimization. Springer, Berlin
26. Conn AR, Gould NIM, Toint PL (2000) Trust Region Methods, MPS/SIAM Series on Optimization. SIAM, Philadelphia
27. Courant R (1943) Variational methods for the solution of problems of equilibrium and vibration. Bull Am Math Soc 49:1–23
28. Di Pillo G, Grippo L (1979) A new class of Augmented Lagrangians in nonlinear programming. SIAM J Control Optim 17:618–628
29. Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. Math Program 91:201–213
30. Dostál Z, Friedlander A, Santos SA (1999) Augmented Lagrangians with adaptive precision control for quadratic programming with equality constraints. Comput Optim Appl 14:37–53
31. Dostál Z, Friedlander A, Santos SA (2002) Augmented Lagrangians with adaptive precision control for quadratic programming with simple bounds and equality constraints. SIAM J Optim 13:1120–1140
32. Dussault J-P (1995) Numerical stability and efficiency of penalty algorithms. SIAM J Numer Anal 32:296–317
33. Ferreira-Mendonça L, Lopes VLR, Martínez JM, Quasi-Newton acceleration for equality constrained minimization. Comput Optim Appl, to appear

34. Fiacco AV, McCormick GP (1968) Nonlinear Programming. John Wiley, New York
35. Fletcher R (1975) An ideal penalty function for constrained optimization. J Inst Math Appl 15:319–342
36. Fletcher R (1987) Practical methods of Optimization. John Wiley
37. Floudas CA (2000) Deterministc global optimization: theory, methods and application. Kluwer, Dordrecht
38. Gonzaga CC, Castillo RA (2003) A nonlinear programming algorithm based on non-coercive penalty functions. Math Program 96:87–101
39. Gould NIM (1989) On the convergence of a sequential penalty function method for constrained minimization. SIAM J Numer Anal 26:107–128
40. Hager WW (1987) Dual techniques for constrained optimization. J Optim Theory Appl 55:33–72
41. Hager WW (1993) Analysis and implementation of a dual algorithm for constrained optimization. J Optim Theory Appl 79:427–462
42. Hestenes MR (1969) Multiplier and gradient methods. J Optim Theory Appl 4:303–320
43. Humes C, Silva PS (2000) Strict convex regularizations, proximal points and Augmented Lagrangians. RAIRO Oper Res 34:283–303
44. Iusem AN (1999) Augmented Lagrangian methods and proximal point methods for convex optimization. Investig Oper 8:11–50
45. Kort BW, Bertsekas DP (1973) Multiplier methods for convex programming. In: Proceedings of the IEEE Decision and Control Conference, San Diego, CA, pp 260–264
46. Kort BW, Bertsekas DP (1976) Combined primal–dual and penalty methods for convex programming. SIAM J Control Optim 14:268–294
47. Lewis RM, Torczon V (2002) A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. SIAM J Optim 12:1075–1089
48. Mangasarian OL, Fromovitz S (1967) The Fritz-John necessary optimality conditions in presence of equality and inequality constraints. J Math Anal Appl 17:37–47
49. Martínez JM, Santos LT (1998) Some new theoretical results on recursive quadratic programming algorithms. J Optim Theory Appl 97:435–454
50. Matioli LC, Uma nova metodologia para construção de funções de penalização para algoritmos de Lagrangiano Aumentado, Tese de Doutorado, Universidade Federal de Santa Catarina, Florianópolis
51. Murphy FH (1974) A class of exponential penalty functions. SIAM J Control 12:679–687
52. Nakayama H, Samaya H, Sawaragi Y (1975) A generalized Lagrangian function and multiplier method. J Optim Theory Appl 17:211–227
53. Polyak RA (2001) Log-sigmoid multiplier method in constrained optimization. Ann Oper Res 101:427–460
54. Powell MJD (1969) A method for nonlinear constraints in minimization problems. In: Fletcher R (ed) Optimization, Academic Press, New York, pp 283–298
55. Qi L, Wei Z (2000) On the constant positive linear dependence condition and its application to SQP methods. SIAM J Optim 10:963–981
56. Rockafellar RT (1974) Augmented Lagrange multiplier functions and duality in nonconvex programming. SIAM J Control 12:268–285
57. Rockafellar RT (1993) Lagrange multipliers and optimality. SIAM Rev 35:183–238
58. Tseng P, Bertsekas D (1993) On the convergence of the exponential multiplier method for convex programming. Math Program 17:670–690

Predictive Method for Interhelical Contacts in Alpha-Helical Proteins

S. R. McAllister, Christodoulos A. Floudas
Department of Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C11

Article Outline

Keywords
Introduction
Methods
  Dataset Selection
  Probability Generation and Probability Sets
  Interhelical Contact Prediction Model
Applications
References

Keywords

Protein folding; Alpha-helical proteins; Mixed-integer linear programming

Introduction

The protein folding question is one of the most challenging problems in computational biology. Although the structures of approximately 40,000 proteins have been determined via experimental techniques and cataloged in the Protein Data Bank (PDB) [3], there are thousands more to be discovered. Modeling proteins with computational techniques is especially critical for

proteins that do not easily crystallize or cannot be address by NMR techniques.

Protein structure prediction requires searching through a vast conformational space for the native structure. This challenge can be met through the application of powerful algorithms, such as the αBB global optimization method [1,2,11]. Many ab initio protein structure prediction methods rely on databases and statistical methods to predict short peptide fragments which are subsequently assembled using scoring functions [8,9,23,37,38,39,40,41,44,45]. Other successful approaches apply detailed physics-based force fields and search for the minimum free energy of a protein [5,6,16,17,18,19,20,21,24,25,26,27,28,29,30,31,34,36,42]. For a detailed summary of protein structure prediction methods, the reader is directed to two recent reviews [12,13].

The prediction of residue contacts within α-helical bundles is critical to the prediction of the overall tertiary structure of these proteins. Predicted interhelical distance restraints can be used to significantly reduce the conformational search space in the protein folding problem. Both modeling [22] and experiments [35] have shown that the residues that define the hydrophobic core are most crucial for folding, limiting the conformational search space. The preference of nonpolar atoms for nonaqueous environments is called the hydrophobic effect [4]. This occurrence is due to the inability of nonpolar molecules to participate in hydrogen bonding in an aqueous environment. The hydrophobic effect is a major stabilization factor for proteins, nucleic acids, and membranes. Because of the dominance of such hydrophobic interactions in protein folding, this paper focuses upon predicting specific hydrophobic residue pair contacts in protein structure. The important contributions of electrostatic, van der Waals, hydrogen bonded, torsional, and solvation energetic terms are included using secondary and tertiary structure prediction methods (e. g., the ASTRO-FOLD approach [16,17,18]).

Methods

There are three main aspects of the overall approach to the helical topology prediction problem. First a dataset of helical proteins was assembled. This dataset is then used to develop hydrophobic residue-based interhelical contact probabilities. Two optimization models are then presented to minimize these contact probabilities subject to constraints that enforce physically realistic topologies. A summary of the important details of the approach are presented here. A more detailed description of the method is available elsewhere [32].

Dataset Selection

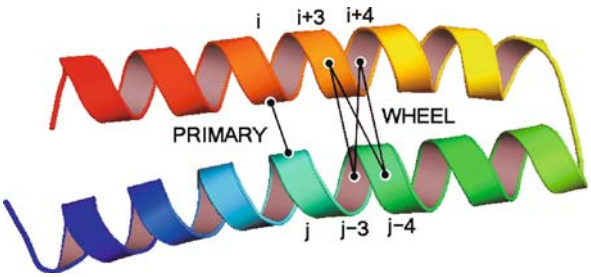
A database PDB set of 318 helical protein structures was compiled to generate probabilities for specific hydrophobic-to-hydrophobic PRIMARY residue contacts and associated hydrophobic-to-hydrophobic WHEEL residue contacts between helices of the same helical protein (this terminology is explained below). They were taken from the following sources: 20 from Table 2 of Zhang et al. [43]; 7 from Table 1 of Huang et al. [15]; 62 from the CATH database [33]; and 229 from the PDB Select 25 Database [14].

Probability Generation and Probability Sets

The probabilities will be established for both a contact at position i (denoted as a PRIMARY contact) and any associated contacts in the helical wheel position (denoted as WHEEL contacts). For the purpose of calculating the PRIMARY probabilities, two helices of a given protein in the database PDB set were considered to interact if they had a contact between a pair of residues with a PRIMARY distance between 4.0 Å and 10.0 Å. Unless otherwise specified, distances in this paper refer to C^{α}-C^{α} distances.

For a parallel helix-to-helix interaction, WHEEL contacts include the following residue combinations: $(i+3)$ to $(j+3)$; $(i+3)$ to $(j+4)$; $(i+4)$ to $(j+3)$; $(i+4)$ to $(j+4)$; $(i-3)$ to $(j-3)$; $(i-3)$ to $(j-4)$; $(i-4)$ to $(j-3)$; and $(i-4)$ to $(j-4)$. For an antiparallel helix-to-helix interaction, the following are the possible WHEEL contacts: $(i+3)$ to $(j-3)$, $(i+3)$ to $(j-4)$; $(i+4)$ to $(j-3)$; $(i+4)$ to $(j-4)$; $(i-3)$ to $(j+3)$; $(i-3)$ to $(j+4)$; $(i-4)$ to $(j+3)$; and $(i-4)$ to $(j+4)$. Figure 1 illustrates the PRIMARY and WHEEL contacts for an antiparallel helical interaction. Two residues k and l that were WHEEL residues of i and j, respectively, were considered to interact if the WHEEL distance between them was greater than or equal to 4.0 and less than 12.0 Å.

Formulating the problem as a set of PRIMARY and WHEEL contacts provides a significant advantage over



Predictive Method for Interhelical Contacts in Alpha-Helical Proteins, Figure 1
Two interacting α-helices in the test set protein 1rop (PDB). The helices here interact in an antiparallel manner, hydrophobic residues *i* and *j* form a PRIMARY contact, and the residues (*i*+3), (*i*+4) can each interact with (*j* — 3), (*j* — 4) to form WHEEL contacts if both residues of a given pair are hydrophobic. This figure was created with PyMol [7]

other methods. Instead of making assumptions about the form of the helix, such as representing the helix as a simple cylinder, the proposed method is able to address irregular helices. By selecting a set of interhelical point contacts within a specified distance range, the approach can handle the most difficult cases, including those containing helices that bend or kink.

Two helices may interact in one of three possible ways. They may be parallel or antiparallel to one another. The third type of possible interaction will be labeled unclassified. These are cases for which neither the parallel nor the antiparallel label applies. For the model prediction section of this paper, only parallel and antiparallel helical interactions were predicted.

The occurrence frequencies of each of the 36 possible hydrophobic pairs were determined by counting the number of occurrences of hydrophobic-to-hydrophobic minimum interhelical distances within the 4.0 to 10.0 Å distance range. The frequency for each pair was then split into two groups, parallel and antiparallel, based on the relative direction of the interacting helices. The total number of hydrophobic-to-hydrophobic minimum distance contacts was identified as the sum of these 36 occurrence frequencies. The PRIMARY probabilities are then established as the occurrence frequency of a specific hydrophobic pair for a specific directionality divided by the total number of hydrophobic-to-hydrophobic contacts.

Conditional WHEEL probabilities were generated for interacting helices in the database PDB set. Given that two α-helices have an interaction with a corresponding hydrophobic-to-hydrophobic minimum interaction distance within 4.0 to 10.0 Å the conditional probability that the residues on the same side of the helical wheel form any hydrophobic-to-hydrophobic contact within 4.0 to 12.0 Å was determined. These probabilities were calculated by considering the number of hydrophobic-to-hydrophobic WHEEL contacts and the total number of possible WHEEL contacts for every specific helix to helix interaction individually, calculating the probability for each interhelical residue contact by averaging over the total number of such contacts after the entire database PDB set has been considered.

Interhelical Contact Prediction Model

Given the locations of helices in a protein's primary sequence, the next step in the proposed method is to predict the interhelical PRIMARY and WHEEL residue contacts. This is done in order to impose distance constraints upon such contacts in the tertiary structure prediction section of the framework.

To accomplish this task, two mixed-integer linear programming (MILP) optimization problems were formulated. The first MILP problem, denoted as the Level 1 Model, identifies a set of the most probable interhelical PRIMARY contacts. The PRIMARY contacts selected by the Level 1 Model are enhanced by the presence of the most probable WHEEL contacts predicted in the Level 2 Model. This second model also provides a method to distinguish between equally likely results of the Level 1 model.

Level 1 Model: PRIMARY Interhelical Contacts In the Level 1 problem, the binary variables y^a_{mn} and y^p_{mn} are activated if the helices m and n of the same protein interact in an antiparallel or a parallel fashion, respectively. In addition, the binary variables w^{mn}_{ij} are defined as active when the hydrophobic residue pair (i, j) forms a PRIMARY contact, where i is in helix m and j is in helix n.

Objective Function: The objective function of Level 1, Eq. (1), corresponds to maximizing the sum of the most probable hydrophobic contacts (i, j) for the given primary sequence by considering the product of the binary variable w^{mn}_{ij}, representing the existence of a residue-residue contact, the binary variables y^a_{mn} and

y^p_{mn}, representing the existence of a helix-helix contact, and the probability of a parallel or antiparallel contact, $p^p_{ij;mn}$ or $p^a_{ij;mn}$, respectively.

$$\max \sum_m \sum_n y^a_{mn} \cdot \sum_i \sum_j w^{mn}_{ij} \cdot p^a_{ij;mn} + \sum_m \sum_n y^p_{mn} \cdot \sum_i \sum_j w^{mn}_{ij} \cdot p^p_{ij;mn} \quad (1)$$

$$y^a_{mn}, y^p_{mn}, w^{mn}_{ij} = \{0, 1\} \quad (2)$$

This objective function is nonlinear due to the products of binary variables that result. The objective function can be reformulated as a linear objective function by introducing a second pair of variables, as described elsewhere [10].

Residue Contact Rules: Every hydrophobic amino acid of helix m, i, can have at most one PRIMARY contact with another hydrophobic amino acid of helix n, j, and this is given by Eq. (3). The two terms in this equation are necessary due to the model formulation; it is assumed that the second index is always larger than the first index for any potentially active variables w_{ij} in order to reduce the number of binary variables.

$$\sum_{j;j>i} w_{ij} + \sum_{j;j<i} w_{ij} \leq 1 \quad (3)$$

Equation (4) prevents any pairs of contacts (i, j) and (i', j') from both being specified if either the number of residues between i and i' or j and j' is less than five or the number of residues between i and i' is different than the number of residues between j and j' by more than two residues. This requires that (i', j') cannot be a WHEEL contact to the PRIMARY contact (i, j) and also limits the size of kinks in the protein backbone that result from a differing separation between (i and i') and (j and j').

$$w^{mn}_{ij} + w^{mn}_{i'j'} \leq 1 \quad \forall(i, i', j, j'): \left|\text{diff}(i, i') - \text{diff}(j, j')\right| > 2 \text{ or either } \left|\text{diff}(i, i')\right| < 5 \text{ or } \left|\text{diff}(j, j')\right| < 5 \quad (4)$$

In this equation, diff(i, i') refers to the difference in sequence numbering between i and i'.

Equation (5) states that if a PRIMARY contact (i, j) occurs, then none of the WHEEL residues for i can also be part of a PRIMARY contact themselves.

$$w^{mn}_{kl} + w^{mn}_{ij} \leq 1 \quad \forall(i, j, k, l): i, k \in m \text{ and } k \text{ is in a WHEEL position of } i \quad (5)$$

Helix Contact Rules: For every helix m, Eq. (6) establishes the maximum number of PRIMARY contacts that can be specified involving m. The parameter $counth(m)$ is established based upon the number of hydrophobic residues within a helix that are not WHEEL residues to each other.

$$\sum_n \left(y^a_{mn} + y^p_{mn}\right) \leq counth(m) \quad \forall m \quad (6)$$

Equations (7)–(8) require a minimum number of loop residues between two α-helices to yield helical interactions of a given orientation.

$$y^p_{mn} = 0 \quad \forall(m, n): \text{loop between } (m, n) \text{ has less than 6 AA} \quad (7)$$

$$y^a_{mn} = 0 \quad \forall(m, n): \text{loop between } (m, n) \text{ has less than 1 AA} \quad (8)$$

Equation (9) states that two helices m and n can either interact in a parallel or antiparallel fashion, if they interact at all, but in only one manner.

$$y^a_{mn} + y^p_{mn} \leq 1 \quad \forall(m, n) \quad (9)$$

Equations (10)–(13) restrict the topology of helical interactions based upon transitive rules.

$$y^a_{mn} + y^a_{np} + y^a_{mp} \leq 2 \quad \forall(m, n, p): m \neq n \neq p, \; nhel \geq 3 \quad (10)$$

$$y^p_{mn} + y^a_{np} + y^p_{mp} \leq 2 \quad \forall(m, n, p): m \neq n \neq p, \; nhel \geq 3 \quad (11)$$

$$y^a_{mn} + y^p_{np} + y^p_{mp} \leq 2 \quad \forall(m, n, p): m \neq n \neq p, \; nhel \geq 3 \quad (12)$$

$$y^p_{mn} + y^p_{np} + y^a_{mp} \leq 2 \quad \forall(m, n, p): m \neq n \neq p, \; nhel \geq 3 \quad (13)$$

Relating Residue Contacts to Helix Contacts: A direct link between the w_{ij} PRIMARY contact variables and the y^a_{mn} and y^p_{mn} helical interaction variables is provided by Eqs. (14)–(15).

$$w^{mn}_{ij} \le y^a_{mn} + y^p_{mn} \tag{14}$$

$$y^a_{mn} + y^p_{mn} - \sum_i \sum_j w^{mn}_{ij} \le 0 \tag{15}$$

Equations (16)–(17) require the PRIMARY contact predictions, w^{mn}_{ij}, to be consistent with the variables representing the interaction directions, y^a_{mn} and y^p_{mn}.

$$w^{mn}_{ij} + w^{mn}_{i'j'} + y^a_{mn} \le 2 \quad \forall (i,j,i',j'): \; i' > i, j' > j \text{ and } |i'-i| < |j'-j| + 3 \text{ or } |i'-i| > |j'-j| - 3 \tag{16}$$

$$w^{mn}_{ij} + w^{mn}_{i'j'} + y^p_{mn} \le 2 \quad \forall (i,j,i',j'): \; i' > i, j > j' \text{ and } |i'-i| < |j'-j| + 3 \text{ or } |i'-i| > |j'-j| - 3 \tag{17}$$

Equations (18) and (19) specify that a PRIMARY contact pair (i, j) may be predicted only if it results in an overlap between the two helices of at least two-thirds of the length of the shorter helix.

$$w^{mn}_{ij} + y^a_{mn} \le 1 \quad \text{if } m,n \text{ overlap } < 2/3 \text{ of shorter helix} \tag{18}$$

$$w^{mn}_{ij} + y^p_{mn} \le 1 \quad \text{if } m,n \text{ overlap } < 2/3 \text{ of shorter helix} \tag{19}$$

Additional Features: Obtaining a rank-ordered list of the most likely sets of helical contacts is more desirable than a single solution. Equation (20) introduces the idea of integer cut constraints into the model. After each successive solve of the above model, the previous solution can be excluded from the feasible solution space using this equation. Here A is the set of active variables, I is the set of inactive variables and card(A) is the cardinality of set A.

$$\sum_{(m,n),(i,j)\in A} \left(y^a_{mn} + y^p_{mn} + w^{mn}_{ij}\right) - \sum_{(m,n),(i,j)\in I} \left(y^a_{mn} + y^p_{mn} + w^{mn}_{ij}\right) \le \text{card}(A) - 1 \tag{20}$$

Equation (21) indicates an upper limit on the number of PRIMARY contacts that two interacting helices m and n can have specified. This upper contact limit is shown as $max_contact$ in Eq. (21).

$$\sum_i \sum_j w^{mn}_{ij} \le max_contact \cdot \left(y^a_{mn} + y^p_{mn}\right) \; \forall (m,n) \tag{21}$$

Equation (22) eliminates a number of helical interactions from the Level 1 solutions equal to the value of the parameter *subtract*. The *subtract* parameter effectively loosens the helical packing, which may be desired in predicting only the most essential and hopefully smallest distance contacts.

$$\sum_m \sum_n \left(y^a_{mn} + y^p_{mn}\right) \le \left(\sum_m counth(m)/2\right) - subtract \tag{22}$$

Equations (1)–(22) represent the Level 1 mathematical model which is a mixed-integer linear optimization problem (MILP).

Level 2 Model: WHEEL Interhelical Contacts The Level 2 MILP problem serves as a check on the ordering of the solutions found in Level 1.

Objective Function: The Level 2 objective function, Eq. (23), maximizes the most probable hydrophobic (k, l) WHEEL contacts based on probabilities calculated using the database PDB set. Although this was done with the (i, j) pairs and parallel or antiparallel orientations already fixed from Level 1, the model could be altered to allow for only fixing the helical orientations after Level 1, for example. The parameters $p^a_{kl;ij;mn}$ and $p^p_{kl;ij;mn}$ give the probabilities that any hydrophobic (k, l) pair will occur on the same side of the helical

wheel as a specific PRIMARY pair (i, j) for antiparallel or parallel orientations, respectively. The Level 2 objective function must be based upon not only the $p^a_{kl;ij;mn}$ and $p^p_{kl;ij;mn}$ probabilities, but also upon the probabilities $p^a_{ij;kl;mn}$ and $p^p_{ij;kl;mn}$ as shown in Eq. (23). These latter probabilities treat the (k, l) WHEEL contacts as PRIMARY contacts themselves, and the values reflect the relative weights of the different (i, j) WHEEL contacts to these (k, l) PRIMARY contacts. This approximation allows for distinguishing between the possible (k, l) hydrophobic contacts that may be specified when there is a choice of more than one pair.

$$\max \sum_{m,n} \sum_{i,j} \sum_{k,l} w^{mn}_{kl;ij} \cdot \left[p^a_{kl;ij;mn} + p^a_{ij;kl;mn} \right] \cdot y^a_{mn} \cdot w^{mn}_{ij} + \sum_{m,n} \sum_{i,j} \sum_{k,l} w^{mn}_{kl;ij} \cdot \left[p^p_{kl;ij;mn} + p^p_{ij;kl;mn} \right] \cdot y^p_{mn} \cdot w^{mn}_{ij} \quad (23)$$

$$y^a_{mn}, y^p_{mn}, w^{mn}_{ij}, w^{mn}_{kl;ij}, y^a_{ijmn}, y^p_{ijmn} = \{0, 1\} \quad (24)$$

Like the objective function in the Level 1 model, Eq. (23) is also nonlinear due to the products of binary variables that result. It can be linearized in a similar fashion [10].

Wheel Residue Contact Rules: Equation (25) states that a maximum of one WHEEL contact is allowed to be specified per primary contact.

$$\sum_{k} \sum_{l} w^{mn}_{kl;ij} \leq w^{mn}_{ij} \quad \forall (m, n, i, j) : y^a_{mn} + y^p_{mn} = 1 \quad (25)$$

Applications

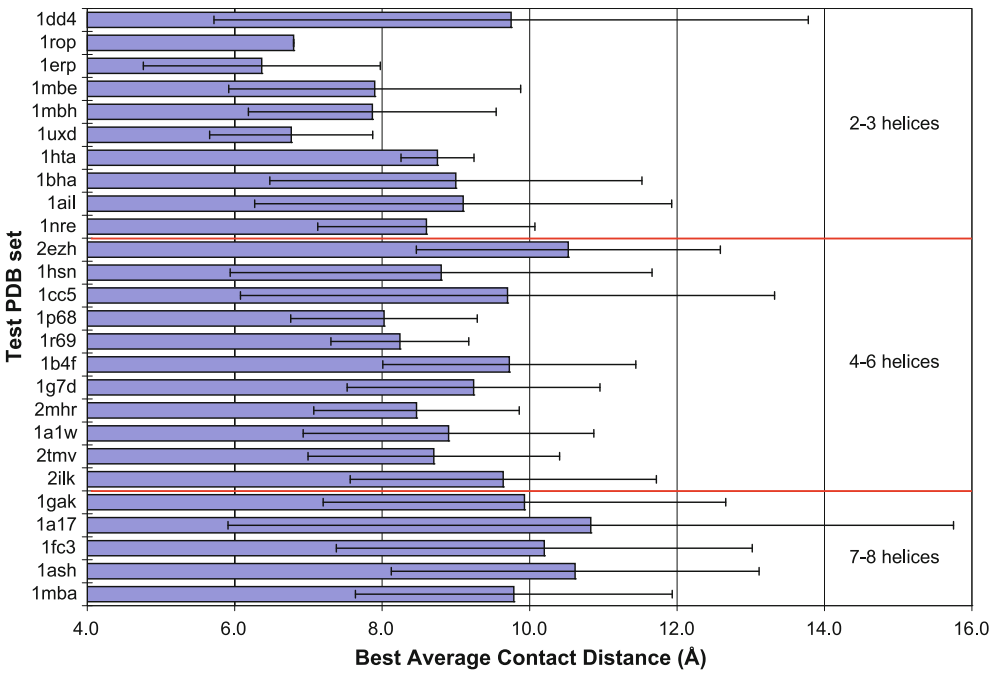
The Level 1 and Level 2 MILP optimization problems were applied to 26 target proteins with known structures in the PDB, termed the test PDB set. For each of these proteins, only the primary amino acid sequence and the experimentally-determined locations of the helices were presented to the model. The model predicted the interhelical hydrophobic residue contacts between such helices using the PRIMARY and WHEEL probabilities developed from globular helical proteins.

A predicted set of interhelical contacts (a solution) was evaluated by computing the average of the distances of these contacts from the experimentally-determined structure. The proposed framework was used to generate 20 solutions for each protein and each parameter value by applying integer cuts. The solutions in this list are ranked by objective value, from best to worst. The best contact distance average value is defined as the lowest contact distance average value identified for a specific protein. An upper limit of 14 Å was identified as a goal for the average distance corresponding to a contact prediction, since such a distance constraint would significantly improve the structure refinement.

Figure 2 displays the lowest combined PRIMARY and WHEEL contact distance averages for every test protein. For the parameters given, these are the best solutions: the experimentally-determined distance averages corresponding to contacts predicted by the model are lower for those solutions than for all other solutions in each protein system. Figure 2 demonstrates that the predictive results of the model are highly encouraging. A general goal of 5.0 to 14.0 Å for the actual distance range of contacts predicted by the model was set, since such a distance range would significantly improve the structure refinement. This goal was attained and surpassed for the entire set of the test proteins.

The error bars of one standard deviation for the distance averages indicate that 1fc3, 1ash, 1cc5, and 2ezh may fall close to this limit and that 1a17 is beyond this target. The averages for a large number of the target proteins fall far below 14.0 Å, suggesting that lower distance restraints such as 12.0 Å or even 10.0 Å and below, may be appropriate in some or even most cases.

The successful predictions of this model using both experimentally-determined helix locations as well as predicted helical regions support the thesis that hydrophobic-to-hydrophobic interactions are key to the folding of the native structures of α-helical globular proteins. Despite the variety of structural motifs present in the test PDB set, the hydrophobic-to-hydrophobic interactions based model was able to identify low distance interhelical PRIMARY and WHEEL contacts for each of the 26 proteins analyzed. This observation is also supported by the success of the *subtract* parameter in identifying the most essential contacts and further reinforces the utility of the probability values developed for the model.



Predictive Method for Interhelical Contacts in Alpha-Helical Proteins, Figure 2
Lowest contact distance averages for identified solutions to the target proteins as explained in the text. Error bars for one standard deviation of the contact distances are given

References

1. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, alphaBB, for general twice-differentiable constrained NLPs - ii. implementation and computational results. Comp Chem Eng 22:1159–1179
2. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, alphaBB, for general twice-differentiable constrained NLPs - i. theoretical advances. Comp Chem Eng 22:1137–1158
3. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) The protein data bank. Nucleic Acids Res 2000 28:235–242
4. Creighton TE (1993) Proteins: Structures and Molecular Properties. W.H. Freeman and Company: New York
5. Czaplewski C, Liwo A, Pillardy J, Oldziej S, Scheraga HA (2004) Improved conformational space annealing method to treat beta-structure with the UNRES force-field and to enhance scalability of parallel implementation. Polymer 45:677–686
6. Czaplewski C, Oldziej S, Liwo A, Scheraga HA (2004) Prediction of the structures of proteins with the UNRES force field, including dynamic formation and breaking of disulfide bonds. Protein Eng Des Sel 17:29–36
7. DeLano WL (2002) The PyMol Molecular Graphics System. San Carlos, CA USA: DeLano Scientific http://www.pymol.org
8. Eyrich VA, Standley DM, Felts AK, Friesner RA (1999) Protein tertiary structure prediction using a branch and bound algorithm. Prot Struct Funct Bioinf 35:41–57
9. Eyrich VA, Standley DM, Friesner RA (1999) Prediction of protein tertiary structure to low resolution: Performace for a large and structurally diverse test set. J Mol Biol 288:725–742
10. Floudas CA (1995) Nonlinear and Mixed-integer Optimization: Fundamentals and Applications. Oxford University Press, New York
11. Floudas CA (2000) Deterministic Global Optimization : Theory, Methods and Applications. Nonconvex Optimization and its Applications. Kluwer, Boston
12. Floudas CA (2005) Research challenges, opportunities and synergism in systems engineering and computational biology. AIChE J 51:1872–1884
13. Floudas CA, Fung HK, McAllister SR, Mönnigmann M, Rajgaria R (2006) Advances in protein structure prediction and de novo protein design: A review. Chem Eng Sci 61:966–988
14. Hobohm U, Sander C (1994) Enlarged representative set of protein structures. Prot Sci 3:522–524
15. Huang ES, Samudrala R, Ponder JW (1999) Ab initio fold prediction of small helical proteins using distance geometry and knowledge-based scoring functions. J Mol Biol 290:267–281

16. Klepeis JL, Floudas CA (2002) Ab initio prediction of helical segments in polypeptides. J Comput Chem 23:245–266
17. Klepeis JL, Floudas CA (2003) Ab initio tertiary structure prediction of proteins. J Glob Optim 25:113–140
18. Klepeis JL, Floudas CA (2003) ASTRO-FOLD: A combinatorial and global optimization framework for ab initio prediction of three-dimensional structures of proteins from the amino acid sequence. Biophys J 85:2119–2146
19. Klepeis JL, Pieja MT, Floudas CA (2003) A new class of hybrid global optimization algorithms for peptide structure prediction: Integrated hybrids. Comp Phys Commun 151:121–140
20. Klepeis JL, Pieja MT, Floudas CA (2003) Hybrid global optimization algorithms for protein structure prediction : Alternating hybrids. Biophys J 84:869–882
21. Klepeis JL, Wei YN, Hecht MH, Floudas CA (2005) Ab initio prediction of the three-dimensional structure of a de novo designed protein: A double-blind case study. Prot Struct Funct Bioinf 58:560–570
22. Lau KF, Dill KA (1990) Theory for protein mutability and biogenesis. Proc Natl Acad Sci USA 87:638–642
23. Lee J, Kim SY, Joo K, Kim I, Lee J (2004) Prediction of protein tertiary strucure using PROFESY, a novel method based on fragment assembly and conformational space annealing. Prot Struct Funct Bioinf 56:704–714
24. Lee J, Pillardy J, Czaplewski C, Arnautova Y, Ripoll DR, Liwo A, Gibson KD, Wawak RJ, Scheraga HA (2000) Efficient parallel algorithms in global optimization of potential energy functions for peptides, proteins and crystals. Comp Phys Commun 128:399–411
25. Lee J, Scheraga HA (1999) Conformational space annealing by parallel computations: Extensive conformational search of met-enkephalin and the 20-residue membrane-bound portion of melittin. Int J Quantum Chem 75:255–265
26. Lee J, Scheraga HA, Rackovsky S (1997) New optimization method for conformational energy calculations on polypeptides : Conformational space annealing. J Comput Chem 18:1222–1232
27. Lee J, Scheraga HA, Rackovsky S (1998) Conformational analysis of the 20-residue membrane-bound portion of melittin by conformational space annealing. Biopolymers 46:103–115
28. Liwo A, Arlukowicz P, Czaplewski C, Oldziej S, Pillardy J, Scheraga HA (2002) A method for optimizing potential-energy functions by hieracrchical design of the potential-energy landscape: Application to the UNRES force field. Proc Natl Acad Sci USA 99:1937–1942
29. Liwo A, Czaplewski C (2001) Cumulant-based expressions for the multibody terms for the correlation between local and electrostatic interactions in the united-residue force field. J Chem Phys 115:2323–2347
30. Liwo A, Oldziej S, Pincus MR, Wawak RJ, Rackovsky S, Scheraga HA (1997) A united-residue force field for off-lattice protein structure simulations. I. Functional forms and parameters of long-range side-chain interaction potentials from protein crystal data. J Comput Chem 18:849–873
31. Liwo A, Pincus MR, Wawak RJ, Rackovsky S, Oldziej S, Scheraga HA (1997) A united-residue force field for off-lattice protein structure simulations. II. Parameterization of short-range interactions and determination of weights of energy terms by z-score optimization. J Comput Chem 18:874–887
32. McAllister SR, Mickus BE, Klepeis JL, Floudas CA (2006) A novel approach for alpha-helical topology prediction in globular proteins: Generation of interhelical restraints. Prot Struct Funct Bioinf 65:930–952
33. Orengo CA, Michie AD, Jones S, Jones DT, Swindells MB, Thornton JM (1997) CATH - a hierarchic classification of protein domain structures. Structure 5:1093–1108
34. Pillardy J, Czaplewski C, Liwo A, Wedemeyer WJ, Lee J, Ripoll DR, Arlukowicz P, Oldziej S, Arnautova EA, Scheraga HA (2001) Development of physics-based energy functions that predict medium resolution structure for proteins of α, β and α/β structural classes. J Phys Chem B 105:7299–7311
35. Reidhaar-Olson JF, Sauer RT (1988) Combinatorial cassette mutagenesis as a probe of the informational content of protein sequences. Science 241:53–57
36. Ripoll D, Liwo A, Scheraga HA (1998) New developments of the electrostatically driven monte carlo method: Tests on the membrane-bound portion of melittin. Biopolymers 46:117–126
37. Rohl CA, Strauss CEM, Chivian D, Baker D (2004) Modeling structurally variable regions in homologous proteins with Rosetta. Prot Struct Funct Bioinf 55:656–677
38. Simons KT, Kooperberg C, Huang C, Baker D (1997) Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. J Mol Biol 268:209–225
39. Simons KT, Ruczinki I, Kooperberg C, Fox BA, Bystroff C, Baker D (1999) Improved recognition of native-like structures using a combination of sequence-dependent and sequence-independent features of proteins. Prot Struct Funct Bioinf 34:82–95
40. Skolnick J, Kolinski A, Kihara D, Betancourt M, Rotkiewicz P, Boniecki M (2001) Ab initio protein structure prediction via a combination of threading, lattice folding, clustering and structure refinement. Prot Struct Funct Bioinf 5(Suppl):149–156
41. Skolnick J, Zhang Y, Arakaki AK, Kolinski A, Boniecki M, Szilágyi A, Kihara D (2003) TOUCHSTONE: A unified approach to protein structure prediction. Prot Struct Funct Bioinf 53:469–479
42. Srinivasan R, Rose GD (2002) Ab initio prediction of protein structure using LINUS. Prot Struct Funct Bioinf 47:489–495
43. Zhang C, Hou JT, Kim SH (2002) Fold prediction of helical proteins using torsion angle dynamics and predicted restraints. Proc Natl Acad Sci USA 99:3581–3585

44. Zhang Y, Skolnick J (2004) Tertiary structure predictions on a comprehensive benchmark of medium to large size proteins. Biophys J 87:2647–2655
45. Zhang Y, Skolnick J (2004) Automated structure prediction of weakly homologous proteins on a genomic scale. Proc Natl Acad Sci USA 101:7594–7599

Preference Disaggregation

YANNIS SISKOS
Decision Support Systems Lab.,
Techn. University Crete, Chania, Greece

MSC2000: 90C29, 91A99

Article Outline

Keywords
Definitions and Notations
Outline of the Article
History
The Disaggregation Paradigm in MCDA
  Global Preference As Datum
The 'Famous' UTA Method
  Objective of the Method
  The Additive Value Model
  The UTASTAR Algorithm.
Variants and Meta-UTA Techniques
Other Disaggregation Methods
Disaggregation Under Uncertainty
Disaggregation in Multi-objective Optimization
Interactive Disaggregation Systems
Applications and Conclusions
  Inference of More Sophisticated Aggregation Models by Disaggregation
  Evaluation of the Aggregation-Disaggregation Relationship
  Experimental Evaluation of Disaggregation Procedures
  Implementation of Disaggregation Methods into DSSs and Group DSSs
See also
References

Keywords

Multicriteria analysis; Preference disaggregation; Linear programming; Decision support system

In decision making involving to multiple criteria, the basic problem stated by analysts and decision makers concerns the way that the final decision should be made. In many cases, however, this problem is posed in the opposite way: assuming that the decision is given, how is it possible to find the rational basis through which the decision was made? Or equivalently, how is it possible to assess the decision maker's preference model leading to the exact same decision as the actual one or at least the most 'similar' decision? The philosophy of preference disaggregation in multicriteria analysis is to assess/infer preference models from given preferential structures and to address decision-aiding activities through operational models within the aforementioned framework.

Definitions and Notations

Under the term *multicriteria analysis* two basic approaches have been developed involving:

a) a set of methods or models enabling the aggregation of multiple evaluation criteria to choose one or more actions from a set A;
b) an activity of decision-aid to a well-defined decision maker (individual, organization, etc.).

In both cases the set A of potential actions or decisions is analysed in terms of multiple criteria in order to model all the possible impacts, consequences or attributes related to the set A (for instance, see [11,27,47,49,69]).

B. Roy [47] outlines a general modeling methodology of decision making problems, which includes four modeling steps beginning with the definition of the object of the decision and ending with the activity of decision aid, as follows:

- Level 1: Object of the decision, including the definition of the set of potential actions A and the determination of a problematic on A (see below).
- Level 2: Modeling a consistent family of criteria assuming that these criteria are nondecreasing value functions, exhaustive and nonredundant.
- Level 3: Development of a global preference model, to aggregate the marginal preferences on the criteria.
- Level 4: Decision-aid or decision support, based on the results of level 3 and the problematic in level 1.

In level 1, Roy [47] distinguishes four referential problematics, each of which does not necessarily preclude the others. These problematics can be employed sepa-

rately or in a complementary way in all phases of the decision making process. The four problematics are the following:

- Problematic α: Choosing one action from A (choice).
- Problematic β: Sorting the actions in well defined categories which are given in a preference order (sorting).
- Problematic γ: Ranking the actions from the best one to the worst one (ranking).
- Problematic δ: Describing the actions in terms of their performances on the criteria (description).

In level 2, the modeling process must conclude on a consistent family of criteria $\{g_1, \ldots, g_n\}$. Each criterion is a nondecreasing real valued function defined on A, as follows:

$$g_i : A \to [g_{i*}, g_i^*] \subset \frac{\mathbf{R}}{a} \to g(a) \in \mathbf{R}\,, \qquad (1)$$

where:

- $[g_{i*}, g_i^*]$ is the criterion evaluation scale;
- g_{i*} is the worst level of the ith criterion;
- g_i^* is the best level of the ith criterion;
- $g_i(a)$ is the evaluation or performance of action A on the ith criterion;
- $g(a)$ is the vector of performances of action A on the n criteria.

From the above definitions the following preferential situations can be determined:

$$g_i(a) > g_i b) \Leftrightarrow a > b \quad (a \text{ is preferred to } b),$$
$$g_i(a) = g_i(b) \Leftrightarrow a \sim b \quad (a \text{ is indifferent to } b).$$

In multicriteria analysis four types of criteria are used with the following properties:

- *Measurable criterion*: The criterion enables the preferential comparison of intervals of the evaluation scale. It can be distinguished in the following subtypes [69]:
  - true criterion (without any threshold);
  - semicriterion (with indifference threshold);
  - pseudocriterion (with indifference and preference thresholds).
- *Ordinal criterion*: The criterion defines only an order on A; thus the evaluation scale is discrete (qualitative criterion);
- *Probabilistic criterion*: It covers the case of uncertainty in the actions' performances modeled by probability distributions (see the Section 'Disaggregation Under Uncertainty' below);
- *Fuzzy criterion*: The actions' performances are intervals of the criterion's evaluation scale.

The modern theoretical steams in the field of multiple criteria decision-aid (MCDA) can be distinguished into four groups:

1) Multi-objective optimization (see the Section 'Disaggregation in Multi-objective Optimization');
2) Value-focused approaches ([29,30]);
3) Outranking methods ([1,49])
4) Disaggregation methods.

Roy and D. Bouyssou [48] point out the major conceptual and methodological differences distinguishing value-focused approaches from outranking methods using the classical example regarding the location of a thermo-nuclear electrical production plant. On the conceptual level, the authors characterize the value-focused approach as descriptive of the decision maker's preferences, while outranking methods are characterized as a constructive way of building these preferences. On the methodological level, the value-focused approach proposes a value or utility function to model the decision maker's global preference (functional systems), whereas outranking methods propose outranking relations (relational systems).

Outline of the Article

The development of disaggregation methods has actually begun in 1978, with the presentation of the UTA method in the 'Cahiers du LAMSADE' series. We will summarise all the progress made in this scientific field from that moment.

The subsequent sections include: the background of the UTA method through the use of goal programming techniques in regression analysis; some thoughts about the usefulness of the disaggregation; a brief presentation of the UTA algorithm; variants of UTA; other disaggregation methods; methods for decision making under uncertainty; multi-objective optimization. Finally, the last two sections show the implementation of disaggregation methods through decision support systems and potential real-world applications.

History

The history of the disaggregation principle in multidimensional/multicriteria analyses begins with the use of goal programming techniques, a special form of linear programming structure, in assessing/inferring preference/aggregation models or in developing linear or nonlinear multidimensional regression analyses [52].

A. Charnes, W.W. Cooper and O. Ferguson [9] proposed a lineal model of optimal estimation of executive compensation by analysing or disaggregating pairwise comparisons and given measures (salaries); the model was estimated so that it could be as consistent as possible with the data from the goal programming point of view.

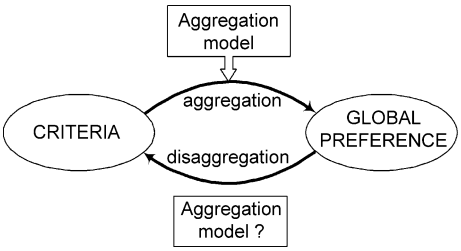
O.J. Karst [28] minimized the sum of absolute deviations via goal programming in linear regression with one variable, while H.M. Wagner [70] generalises Karst's model in the multiple regression case. Later, J.E. Kelley [31] proposed a similar model to minimize Tchebycheff's criterion in linear regression.

Later, V. Srinivasan and A.D. Shoker [65] outlined the ORDREG ordinal regression model to assess a linear value function by disaggregating pairwise judgements. N. Freed and G. Glover [17] proposed goal programming models to infer the weights of linear value functions in the frame of discriminant analysis (problematic β).

The research on handling ordinal criteria has begun with the studies [71] and [25]. Both research teams faced the same problem: to infer additive value functions by disaggregating a ranking of reference alternatives. F.W. Young, J. de Leeuw and Y. Takane [71] proposed alternating least squares techniques, without ensuring, however, that the additive value function is optimally consistent with the given ranking. In the case of the UTA method proposed by E. Jacquet-Lagrèze and J. Siskos [25] optimality is ensured through linear programming techniques.

The Disaggregation Paradigm in MCDA

In the traditional aggregation paradigm, the criteria aggregation model is known a priori, while the global preference is unknown. On the contrary, the philosophy of disaggregation involves the inference of preference models from given global preferences (Fig. 1).



Preference Disaggregation, Figure 1
The aggregation and disaggregation paradigms in MCDA

Global Preference As Datum

The clarification of the decision maker's global preference necessitates the use of a reference set of actions A_R. Usually, this set could be:

1) a set of past decision alternatives (A_R: past actions);
2) a subset of decision actions, especially when A is large ($A_R \subset A$);
3) a set of fictitious actions, consisting of performances on the criteria which can be easily judged by the decision maker to perform global comparisons A_R: fictitious actions).

In each of the above cases the decision maker is asked to externalise and/or confirm his/her global preferences on the set A_R taking into consideration the performances of the reference actions on all criteria. Usually, the form of the global preference follows the following typology:

- Measurable judgements on A_R;
- Ranking (weak order relation) on A_R (problematic γ);
- Pairwise relation;
- Sorting of reference actions (problematic β).

The 'Famous' UTA Method

Objective of the Method

The UTA method proposed by Jacquet-Lagrèze and Siskos [26] aims at inferring one or more additive value functions from a given ranking on the reference set A_R. The method uses special linear programming techniques to assess these functions so that the ranking(s) obtained through these functions on A_R is (are) as consistent as possible with the given one.

The Additive Value Model

The criteria aggregation model in UTA is assumed to be an additive value function of the following form [30]:

$$u(\underline{g}) = \sum_{i=1}^{n} p_i u_i(g_i) \tag{2}$$

subject to normalization constraints:

$$\begin{aligned} &u_i(g_{i*}) = 0\,, \quad u_i(g_i^*) = 1\,, \\ &\quad \forall i = 1, \ldots, n\,, \\ &\sum_{i=1}^{n} p_i = 1\,, \end{aligned} \tag{3}$$

where u_i, $i = 1, \ldots, n$, are nondecreasing real valued functions, named marginal value or utility functions, which are normalized between 0 and 1, and p_i is the weight of u_i.

Both the marginal and the global value functions have the monotonicity property of the true criterion. For instance, in the case of the global value function the following properties hold:

$$\begin{aligned} &u\left[\underline{g}(a)\right] > u\left[\underline{g}(b)\right] \Leftrightarrow a \succ b \quad \text{(preference)}, \\ &u\left[\underline{g}(a)\right] = u\left[\underline{g}(b)\right] \Leftrightarrow a \sim b \quad \text{(indifference)}. \end{aligned}$$

The UTA method infers an unweighted form of the additive value function, equivalent to the form defined from relations (2)–(3), as follows:

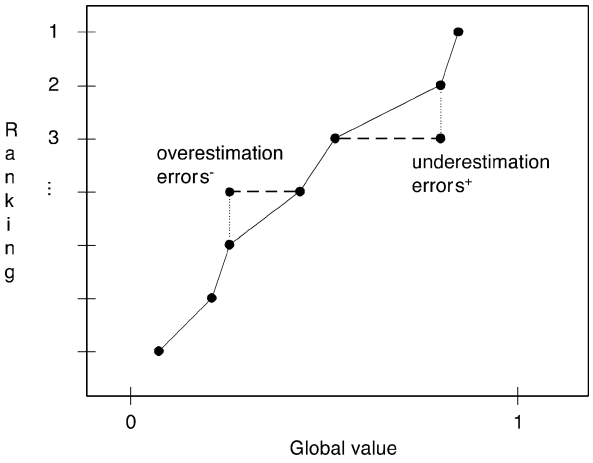
$$u(\underline{g}) = \sum_{i=1}^{n} u_i(g_i) \tag{4}$$

subject to normalization constraints:

$$u_i(g_{i*}) = 0\,, \quad \forall i = 1, \ldots, n\,, \tag{5}$$

$$\sum_{i=1}^{n} u_i(g_i^*) = 1\,. \tag{6}$$

Of course, the existence of such a preference model assumes the preferential independence of the criteria for the decision maker [30], although this assumption does not pose significant problems in a posteriori analyses such as disaggregation analysis.



Preference Disaggregation, Figure 2
Ordinal regression curve (ranking versus global value)

The UTASTAR Algorithm.

In order to assess every marginal value function, the evaluation scales of each criterion (especially in the case of measurable criteria) is discretised in a limited set of points:

$$G_i = \left\{g_{i*} = g_i^1, g_i^2, \ldots, g_i^{a_i} = g_i^*\right\}\,. \tag{7}$$

On the other hand, the set of reference actions $A_R = \{a_1, \ldots, a_k\}$ is 'rearranged' in such a way that a_1 is the head of the ranking and a_k its tail. Since the ranking has the form of a weak order, for each pair of consecutive actions (a_j, a_{j+1}) one of the two following relations holds:

$$\begin{aligned} &a_j \succ a_{j+1} \quad \text{(preference)}, \\ &a_j \sim a_{j+1} \quad \text{(indifference)}. \end{aligned}$$

In the original version of UTA [26], for each packed action $a \in A_R$ a single error $\sigma(a)$ is introduced to be minimised. Later, Y. Siskos and D. Yannacopoulos [60] introduced two errors leading to better results (Fig. 2). This variant of UTA, is now called UTASTAR.

The main computational procedure employed in UTASTAR employs linear programming techniques to find additive value functions which are as consistent as possible with the ranking on A_R:

1) Express the global value of reference actions $u[\underline{g}(a_j)]$, $j = 1, \ldots, k$, first in terms of marginal val-

ues $u_i(g_i)$, then in terms of variables:

$$w_{il} = u_i(g_i^{l+1}) - u_i(g_i^l) \geq 0, \quad i = 1, \ldots, n; \quad l = 1, \ldots, \alpha_i - 1,$$

by means of the relations

$$u_i(g_i^l) = \sum_{t=1}^{l-1} w_{it}, \quad u_i(g_i^1) = 0, \qquad \forall i \text{ and } l > 1.$$

2) Introduce two error functions σ^+ and σ^- on A_R by writing for each pair of consecutive actions in the ranking, the analytic expressions:

$$\Delta(a_j, a_{j+1}) = u[\underline{g}(a_j)] - \sigma^+(a_j) + \sigma^-(a_j) - u[\underline{g}(a_{j+1})] + \sigma^+(a_{j+1}) - \sigma^-(a_{j+1}). \quad (8)$$

3) Solve the linear program

$$\min z = \sum_{j=1}^{k} [\sigma^+(a_j) + \sigma^-(a_j)]$$

subject to the set of constraints:

$$\begin{cases} \Delta(a_j, a_{j+1}) \geq \delta & \text{if } a_j \succ a_{j+1}, \\ \Delta(a_j, a_{j+1}) = 0 & \text{if } a_j \sim a_{j+1}, \end{cases} \quad \forall j = 1, \ldots, k-1, \quad (9)$$

$$\sum_{i=1}^{n} \sum_{l=1}^{a_i - 1} w_{il} = 1, \quad w_{il} \geq 0, \quad i = 1, \ldots, n; \quad l = 1, \ldots, \alpha_i - 1, \quad (10)$$

$$\sigma^+(a_j) \geq 0, \quad \sigma^-(a_j) \geq 0, \; j = 1, \ldots, k, \quad (11)$$

σ being a small positive number.

4) Test the existence of multiple or near optimal solutions of the linear program in step 3 (stability analysis); in case of nonuniqueness, find the mean additive value function of those (near) optimal solutions which maximise the objective functions $p_i = u_i(g_i^*) = \sum_l w_{il}$ for all $i = 1, \ldots, n$ on the polyhedron (9)–(11) bounded by the new constraint:

$$\sum_{j=1}^{k} [\sigma^+(a_j) + \sigma^-(a_j)] \leq z^* + \varepsilon, \quad (12)$$

z^* being the optimal value of the linear program in step 3 and ε a very small positive number.

Variants and Meta-UTA Techniques

After the development of the UTA method several variants have been developed incorporating different forms of global preference or different forms of optimality criteria used in the linear programming formulation. The main variants include:

- Inferring u from pairwise comparisons [26].
- Maximising Kendall's τ, a consistency measure between the two rankings, via a mixed linear programming formulation [26].
- Inferring u from assignment examples in the case of problematic β ([14,22,26,78]).
- Optimising lexicographic criteria without discretization of criteria scales G_i [41].
- Inferring u in the presence of nonmonotonic preferences on the criteria evaluation scales [13].

Other techniques, named meta-UTA, aimed at the improvement of the value function with respect to near optimality analysis or to its exploitation for decision support.

D.K. Despotis, Yannacopoulos and C. Zopounidis [12] propose to minimise the error's dispersion (Tchebycheff criterion) within the UTA's step 4. In the case where UTA gives a sum of error equal to zero ($z^* = 0$), M. Beuthe and G. Scanella [5] propose the meta-UTA techniques UTAMP1 maximising δ (minimum difference between the global value of two consecutive reference actions) in near optimality analysis, and UTAMP2 maximising δ plus the minimum of marginal value step w_{il} of UTA's step 1.

Beuthe and Scanella [7] propose similar techniques for the case of $z^* > 0$ and provide some comparative analysis results for different UTA variants.

Finally, Siskos [51] suggests the construction of fuzzy outranking relations based on multiple value functions u provided by UTA's near optimality analysis.

Other Disaggregation Methods

The disaggregation logic has been employed almost in all aggregation models in multicriteria analysis. Of course, in some cases, it is not easy to infer aggregation models or procedures from their output.

A first attempt to infer ELECTRE III from a given ranking was made in [45] without satisfactory results. L.N. Kiss et al. [33] developed the ELECCALC system

that estimates indirectly the parameters of the ELECTRE II method from the decision maker's responses to questions of the system regarding his/her global preferences. V. Mousseau and R. Slowinski [40] and Mousseau et al. [39] present linear programming formulations to infer the parameters of ELECTRE TRI, except the veto thresholds, from assignment example in the case of problematic β.

P. Spiliopoulos [63] and Siskos and N.F. Matsatsinis [55] employ the UTA method iteratively as a model to analyse consumers' behavior, through the MARKEX system providing decision support during new product development [37].

The disaggregation of measurable judgements in order to infer additive value functions for prediction purposes is proposed in [32]. Siskos et al. [58] developed an ordinal regression formulation to measure customer satisfaction by disaggregating multiple satisfaction judgements. The method was implemented in the MUSA system and it was applied in several real-world studies (for instance, see [38]).

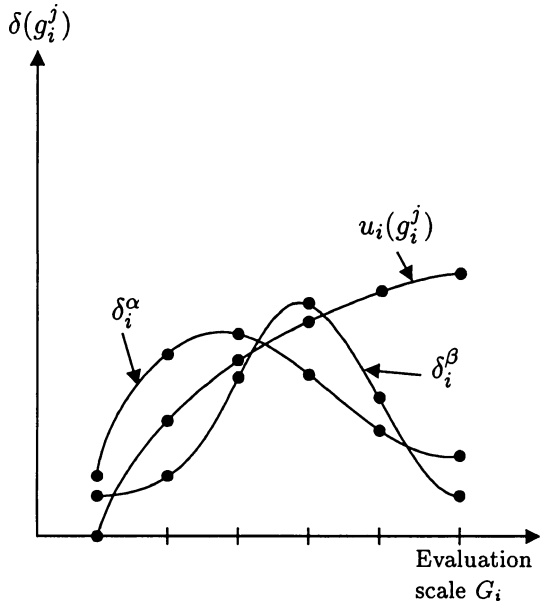
UTA was used in several works for conflict resolution in multi-actor decision situations ([8,24,36]).

Additive value functions are usually assessed in two phases: in the first phase the marginal value functions are assessed under the preferential independence conditions and in the second phase their weights are assessed by disaggregating a ranking of a small number of reference actions [50]. Two-phase disaggregation methods were implemented through the MACBETH system ([2,3]) and the MIIDAS system ([59,64]).

The general scheme of the disaggregation philosophy is also employed in other approaches, including rough sets ([16,43,62,72]), machine learning [44] and neural networks ([35,66]). All these approaches are used to infer some form of decision model (a set of decision rules or a network) from given decision results involving assignment examples, ordinal or measurable judgements.

Disaggregation Under Uncertainty

Within the framework of multicriteria decision aid under uncertainty, Siskos [52] developed a specific version of UTA (Stochastic UTA), in which the aggregation model to infer from a reference ranking is an additive utility function of the form:



Preference Disaggregation, Figure 3
Distributional evaluation and marginal value function

$$u(\underline{\delta}^a) = \sum_{i=1}^{n} \sum_{j=1}^{a_i} \delta_i^a(g_i^j) u_i(g_i^j) \tag{13}$$

subject to normalization constraints (5)–(6), with the following additional notation (see also Fig. 3):

- δ_i^a is the distributional evaluation of action A on the ith criterion;
- $\delta_i^a(g_i^j)$ is the probability that the performance of action A on the ith criterion is g_i^j;
- $u_i(g_i^j)$ is the marginal value of the performance g_i^j;
- $\underline{\delta}^a$ is the vector of distributional evaluations of action A;
- $u(\underline{\delta}^a)$ is the global utility of action A.

Of course, the additive utility function (13) has the same properties as the value function:

$$u(\underline{\delta}^a) > u(\underline{\delta}^b) \Leftrightarrow a \succ b \text{ (preference)},$$
$$u(\underline{\delta}^a) = u(\underline{\delta}^b) \Leftrightarrow a \sim b \text{ (indifference)}.$$

Similarly to the case of the UTA described above, the stochastic UTA method disaggregates a ranking of reference actions [53]. The algorithmic procedure could be expressed in the following way:

1) Express the global expected utilities of reference actions $u(\underline{\delta}^{a_j})$ in terms of variables $w_{il} = u_i\,(g_i^{l+1}) - u_i(g_i^l) \geq 0$.
2) Introduce two error functions σ^+ and σ^-:

$$\Delta(a_j, a_{j+1}) = u(\underline{\delta}^{a_j}) - \sigma^+(a_j) + \sigma^-(a_j) \tag{14}$$

$$-u(\underline{\delta}^{a_{j+1}}) + \sigma^+(a_{j+1}) - \sigma^-(a_{j+1}). \tag{15}$$

3) Solve the linear program:

$$\min z = \sum_{j=1}^{k} [\sigma^+(a_j) + \sigma^-(a_j)]$$

subject to the set of constraints:

$$\begin{cases} \Delta(a_j, a_{j+1}) \geq \delta & \text{if } a_j \succ a_{j+1}, \\ \Delta(a_j, a_{j+1}) = 0 & \text{if } a_j \sim a_{j+1}, \end{cases} \quad \forall j = 1, \ldots, k-1\,, \tag{16}$$

$$\begin{gathered} \sum_i \sum_l w_{il} = 1\,, \\ w_{il} \geq 0\,, \\ i = 1, \ldots, n; \quad l = 1, \ldots, \alpha_i - 1\,, \\ \sigma^+(a_j) \geq 0, \quad \sigma^-(a_j) \geq 0\,, \\ j = 1, 2, \ldots, k\,, \end{gathered} \tag{17}$$

δ being a small positive number.
4) Test the existence of multiple or near optimal solutions.

Of course, the ideas employed in all variants of the UTA method are also applicable in the same way in the case of the stochastic UTA.

Disaggregation in Multi-objective Optimization

The disaggregation approach is also applicable in the specific field of multi-objective optimization, mainly in the field of linear programming with multiple objective functions. For instance, in the classical methods of [18] and [73] the weights of the linear combinations of the objectives are inferred locally from trade-offs or pairwise judgements given by the decision maker at each iteration of the methods.

T.J. Stewart [67] proposed a procedure of pruning the decision alternatives using the UTA method, while Jacquet-Lagrèze, R. Meziani and Slowinski [23] developed a disaggregation method, similar to UTA, to assess a whole value function of multiple objectives for linear programming systems.

Siskos and Despotis [54] proposed an interactive method named ADELAIS that uses UTA iteratively, in order to optimise an additive value function within the feasible region defined on the basis of the satisfaction levels determined during each iteration. Finally, A. Tangian and M.J. Gruber [68] propose a different form of disaggregation techniques for the assessment of quadratic multi-objective functions.

Interactive Disaggregation Systems

Except for the normative features that a disaggregation approach provides, it also constitutes a basis for the interaction between analysts and decision makers. The issues involved during this interactive dialog and negotiation include:

- the consistency between the assessed preference model and the a priori preferences of the decision maker;
- the assessed values (values, weights, utilities, etc.); and
- the overall evaluation of potential actions (extrapolation output).

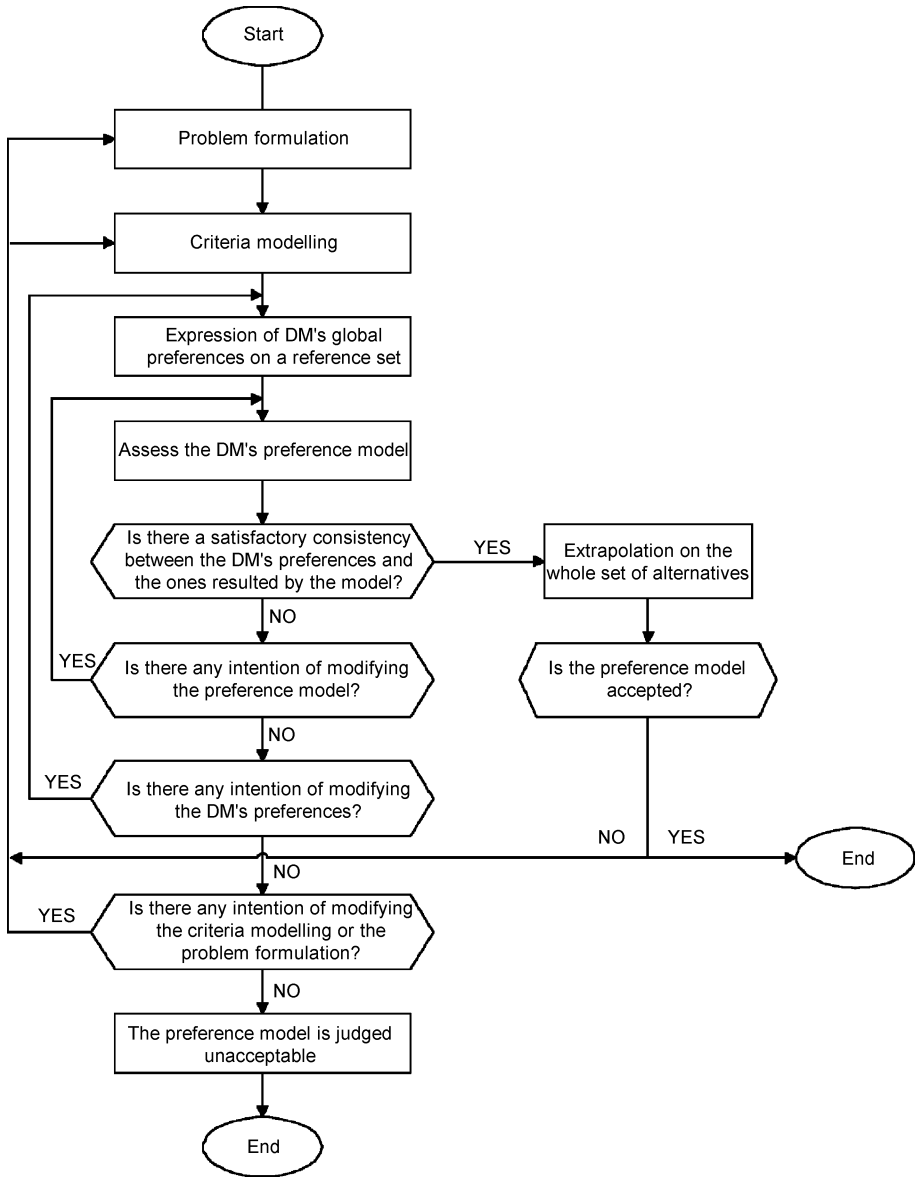
A general interaction scheme is given in Fig. 4.

Several decision support software have been developed on the basis of disaggregation methods, most of them being UTA based. They include: PREFCALC [21], MINORA-MIIDAS [59], ADELAIS [54], MARKEX [55], UTA+ [34], FINEVA [81], FINCLAS [75], PREFDIS [77], MUSTARD [6].

Applications and Conclusions

From their first appearance in 1978 onwards, preference disaggregation methods have been applied in several real-world decision making problems from the fields of financial management, marketing, environmental management, as well as human resources management. The following list reports some of these applications (list not exhaustive):

- financial management
  - venture capital [56];



Preference Disaggregation, Figure 4
Simplified decision support process based on disaggregation analysis

  - portfolio selection and management ([20,79]);
  - business failure prediction ([74,76]);
  - business financing ([61,75,81]);
  - country risk assessment ([10,42,80]);
- marketing
  - marketing of new products ([4,55,63]);
  - sales strategy problems ([46,57]);
  - customer satisfaction ([38,58]);
- environmental management ([15,19,53]);
- industrial project evaluation [22];
- job evaluation [64].

The above applications have provided insight on the applicability of preference disaggregation analysis in addressing real-world decision problems and its efficiency. The future research developments on this field required to explore further the potentials of the preference disaggregation philosophy within the context of multicriteria decision aid, include:

Inference of More Sophisticated Aggregation Models by Disaggregation

Currently most preference disaggregation methods lead to the assessment of additive value functions. However, in many cases such additive models fail to comprise the decision maker's preference in a satisfactory way, either because their underlying assumptions do not hold (preferential independence), or because they do not consider the existing interactions between the criteria. In that respect it is worth examining the development of alternative aggregation models, for instance in the form of multiplicative value functions or even outranking relations.

Evaluation of the Aggregation-Disaggregation Relationship

Both aggregation and disaggregation procedures share the same objective: to aggregate all criteria into a global preference model that will support decision making. Of course, as this paper has demonstrated, there are significant differences in the process employed in both approaches to accomplish this objective. However, it would be interesting to explore the relationship of aggregation and disaggregation procedures in terms of similarities and/or dissimilarities regarding the evaluation results obtained by both approaches. This will enable the identification of the reasons and the conditions under which aggregation and disaggregation procedures will lead to different or the same results.

Experimental Evaluation of Disaggregation Procedures

Real-world applications can be used to illustrate the decision support provided by preference disaggregation approaches in practice. However, a thorough investigation of their performance and ability to capture the decision maker's preferences requires the conduct of experimental studies. Through such studies it is possible to examine how different data conditions and preferential structures affect the efficiency of preference disaggregation approaches and the aggregation models used.

Implementation of Disaggregation Methods into DSSs and Group DSSs

Preference disaggregation procedures operate on an interactive and iterative way. The decision maker interacts with the procedure to achieve a consistent representation of his/her preferences in the aggregation model through an iterative trial and error process. The DSSs technology is well adapted to both these features, enabling the decision maker to take full advantages that preference disaggregation approaches provide, in real-time. Furthermore, this framework could be extended bearing in mind the fact that many crucial decisions are taken by a group of decision makers working in a negotiating or cooperative environment to conclude to a consensus decision. Group DSSs provide the means required in supporting such decision making situations.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Bana e Costa CA (1990) Readings in multiple criteria decision aid. Springer, Berlin

2. Bana e Costa CA, Nunes Da Silva F, Vansnick J-C (2000) Conflict dissolution in the public sector: A case-study. Europ J Oper Res
3. Bana e Costa CA, Vansnick (1997) Applications of the MACBETH approach in the framework of an additive aggregation model. J Multi-Criteria Decision Anal 6(2):107–114
4. Baourakis G, Matsatsinis NF, Siskos Y (1996) Agricultural product development using multidimensional and multicriteria analyses: The case of wine. Europ J Oper Res 94:321–334
5. Beuthe M, Scannella G (1996) Applications comparées des méthodes d'analyse multicritère UTA. RAIRO Oper Res 30(3):293–315
6. Beuthe M, Scannella G (1999) MUSTARD user's guide. GTM Fac Univ Catholiques de Mons (FUCaM)
7. Beuthe M, Scannella G (2000) Comparative analysis of UTA multicriteria methods. Europ J Oper Res
8. Bui TX (1987) Co-oP: A group decision support system for cooperative multiple criteria group decision making. Lecture Notes Computer Sci., vol 290. Springer, Berlin
9. Charnes A, Cooper WW, Ferguson RO (1955) Optimal estimation of executive compensation by linear programming. Managem Sci 1(2):138–151
10. Cosset JC, Siskos Y, Zopounidis C (1992) Evaluating country risk: A decision support approach. Global Finance J 3(1):79–95
11. De Montgolfier J, Bertier P (1978) Approche multicritère des problèmes de décision. Collection AFCET Ed. Hommes et Techniques
12. Despotis DK, Yannacopoulos D, Zopounidis C (1990) A review of the UTA multicriteria method and some improvements. Found Computing and Decision Sci 15(2):63–76
13. Despotis DK, Zopounidis C (1993) Building additive utilities in the presence of non-monotonic preference. In: Pardalos PM, Siskos Y, Zopounidis C (eds) Advances in Multicriteria Analysis. Kluwer, Dordrecht, pp 101–114
14. Devaud JM, Groussaud G, Jacquet-Lagrèze E (1980) UTADIS: Une méthode de construction de fonctions d'utilité additives rendant compte de jugements globaux. Europ. Working Group on Multicriteria Decision Aid, Paris
15. Diakoulaki D, Zopounidis C, Mavrotas G, Doumpos M (1999) The use of a preference disaggregation method in energy analysis and policy making. Energy-The Internat J 24(2):157–166
16. Dimitras AI, Slowinski R, Susmaga R, Zopounidis C (1999) Business failure prediction using rough sets. Europ J Oper Res 114:263–280
17. Freed N, Glover G (1981) Simple but powerful goal programming models for discriminant problems. Europ J Oper Res 7:44–60
18. Geoffrion AM, Dyer JS, Feinberg A (1972) An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. Managem Sci 19:357–368
19. Hatzinakos I, Yannacopoulos D, Faltsetas C, Ziourkas C (1991) Application of the MINORA decision support system to the evaluation of landslide favourability in Greece. Europ J Oper Res 50:60–75
20. Hurson Ch, Zopounidis C (1997) Gestion de portefeuille et analyse multicritère. Economica, Paris
21. Jacquet-Lagrèze E (1990) Interactive assessment of preferences using holistic judgments: The PREFCALC system. In: Bana e Costa CA (ed) Readings in Multiple Criteria Decision Aid. Springer, Berlin, pp 225–250
22. Jacquet-Lagrèze E (1995) An application of the UTA discriminant model for the evaluation of RandD projects. In: Pardalos PM, Siskos Y, Zopounidis C (eds) Advances in Multicriteria Analysis. Kluwer, Dordrecht, pp 203–211
23. Jacquet-Lagrèze E, Meziani R, Slowinski R (1987) MOLP with an interactive assessment of a piecewise linear utility function. Europ J Oper Res 31:350–357
24. Jacquet-Lagrèze E, Shakun MF (1984) Decision support systems for semistructured buying decisions. Europ J Oper Res 16:48–56
25. Jacquet-Lagrèze E, Siskos J (1978) Une méthode de construction de fonctions d'utilité, additives explicatives d'une préférence globale. Cahier du LAMSADE, vol 16. Univ. Paris-Dauphine, Paris
26. Jacquet-Lagrèze E, Siskos J (1982) Assessing a set of additive utility functions for multicriteria decision making: The UTA method. Europ J Oper Res 10:151–164
27. Jacquet-Lagrèze E, Siskos J (1983) Méthodes de décision multicritére. Ed. Hommes et Techniques
28. Karst OJ (1958) Linear curve fitting using least deviations. J Amer Statist Assoc 53:118–132
29. Keeney RL (1992) Value-focused thinking: A path to creative decisionmaking. Harvard University Press, Cambridge
30. Keeney RL, Raiffa H (1976) Decisions with multiple objectives: Preferences and value tradeoffs. Wiley, New York
31. Kelley JE (1958) An application of linear programming to curve fitting. J Industr Appl Math 6:15–22
32. Kettani O, Oral M, Siskos Y (1998) A multiple criteria analysis model for real estate evaluation. J Global Optim 12(2):197–214
33. Kiss LN, Martel JM, Nadeau R (1994) ELECCALC-An interactive software for modelling the decision maker's preferences. Decision Support Systems 12:311–326
34. Kostkowski M, Slowinski R (1996) UTA+ application (v. 1.20) – User's manual. Document du LAMSADE, vol 95. Univ. Paris-Dauphine, Paris
35. Malakooti B, Zhou YQ (1994) Feedforward artificial neural networks for solving discrete multiple criteria decision making problems. Managem Sci 40(11):1542–1561
36. Matsatsinis NF, Samaras A (2000) MCDA and preference disaggregation in group decision support systems. Europ J Oper Res
37. Matsatsinis NF, Siskos Y (2000) Intelligent support systems for marketing decision. Kluwer, Dordrecht

38. Mihelis G, Grigoroudis E, Siskos Y, Politis Y, Malandrakis Y (2000) Customer satisfaction measurement in the private bank sector. Europ J Oper Res
39. Mousseau V, Figueira J, Naux J-Ph (2000) Using assignment examples to infer weights for Electre Tri method: Some experimental results. Europ J Oper Res
40. Mousseau V, Slowinski R (1998) Inferring an ELECTRE-TRI model from assignment examples. J Global Optim 12(2):157–174
41. Oral M, Kettani O (1989) Modelling the process of multiattribute choice. J Oper Res Soc 40(3):281–291
42. Oral M, Kettani O, Cosset JC, Daouas M (1992) An estimation model for country risk rating. Internat J Forecasting 8:583–593
43. Pawlak Z (1982) Rough sets. Internat J Inform Comput Sci 11:341–356
44. Quinlan JR (1986) Induction of decision trees. Machine Learning 1:81–106
45. Richard JL (1981) Procédure multicritère d'aide á la decision en matière d'audit de stratègie: Cas des moyennes et petites industries. Thèse de 3e cycle, Univ Paris-Dauphine
46. Richard JL (1983) Aide á la décision stratégique en P.M.E. In: Jacquet-Lagrèze E, Siskos Y (eds) Méthode de Décision Multicritère. Ed. Hommes et Techniques, pp 119–142
47. Roy B (1985) Méthodologie multicritère d'aide á la décision. Economica, Paris
48. Roy B, Bouyssou D (1986) Comparison of decision-aid models applied to a nuclear power plant citing example. Europ J Oper Res 25:200–215
49. Roy B, Bouyssou D (1993) Aide multicritère á la décision: Méthodes et cas. Economica, Paris
50. Siskos J (1980) Comment modéliser les préférences au moyen de fonctions d'utilité additives. RAIRO Rech Opérat 14:53–82
51. Siskos J (1982) A way to deal with fuzzy preferences in multicriteria decision problems. Europ J Oper Res 10:314–324
52. Siskos J (1983) Analyse de systèmes de décision multicritère en univers aléatoire. Found Control Eng 8:193–212
53. Siskos J, Assimakopoulos N (1989) Multicriteria highway planning: A case study. Math Comput Modelling 12(10–11):1401–1410
54. Siskos J, Despotis DK (1989) A DSS oriented method for multiobjective linear programming problems. Decision Support Systems 5:47–55
55. Siskos J, Matsatsinis NF (1993) A DSS for market analysis and new product design. J Decision Systems 2(1):35–60
56. Siskos J, Zopounidis C (1987) The evaluation criteria of the venture capital investment activity: An interactive assessment. Europ J Oper Res 31:304–313
57. Siskos Y (1986) Evaluating a system of furniture retail outlets using an interactive ordinal regression method. Europ J Oper Res 23:179–193
58. Siskos Y, Grigoroudis E, Zopounidis C, Saurais O (1998) Measuring customer satisfaction using a survey based preference disaggregation model. J Global Optim 12(2):175–195
59. Siskos Y, Spiridakos A, Yannacopoulos D (1999) Using artificial intelligence and visual techniques into preference disaggregation analysis: The MIIDAS system. Europ J Oper Res 113:236–246
60. Siskos Y, Yannacopoulos D (1985) UTASTAR: An ordinal regression method for building additive value functions. Invest Operacion 5(1):39–53
61. Siskos Y, Zopounidis C, Pouliezos A (1994) An integrated DSS for financing firms by an industrial development bank in Greece. Decision Support Systems 12:151–168
62. Slowinski R (1995) Rough set approach to decision analysis. AI Expert Magazine 10(3):18–25
63. Spiliopoulos P (1987) Analyse et simulation du marché pour le lancement d'um nouveau produit - Réalisation d'un SIAD. Thèse de 3e cycle, Univ Paris-Dauphine
64. Spyridakos A, Siskos Y, Yannakopoulos D, Skouris A (2000) Multicriteria job evaluation for large organisations. Europ J Oper Res
65. Srinivasan V, Shocker AD (1973) Linear programming techniques for multidimensional analysis of preferences. Psychometrika 38(3):337–396
66. Stam A, Sun M, Haines M (1996) Artificial neural network representations for hierarchical preference structures. Comput Oper Res 23(12):1191–1201
67. Stewart TJ (1987) Pruning of decision alternatives in multiple criteria decision making, based on the UTA method for estimating utilities. Europ J Oper Res 28:79–88
68. Tangian A (2000) Constructing a monotonic quadratic objective function in n variables from a few 2-dimensional indifferences. Europ J Oper Res
69. Vincke Ph (1992) Multicriteria decision aid. Wiley, New York
70. Wagner HM (1959) Linear programming techniques for regression analysis. J Amer Statist Assoc 54:206–212
71. Young FW, De Leeuw J, Takane Y (1976) Regression with qualitative and quantitative variables: An alternating least squares method with optimal scaling features. Phychometrica 41:505–529
72. Zaras K (2000) Rough approximation of a preference relation by a multi-attribute stochastic dominance for deterministic and stochastic evaluation problems. Europ J Oper Res
73. Zionts A, Wallenius J (1976) An interactive programming method for solving the multiple criteria problem. Managem Sci 22:652–663
74. Zopounidis C (1987) A multicriteria decision-making methodology for the evaluation of the risk of failure and an application. Found Control Eng 12(1):45–67
75. Zopounidis C, Doumpos M (1998) Developing a multicriteria decision support system for financial classification problems: The FINCLAS system. Optim Methods Softw 8:277–304

76. Zopounidis C, Doumpos M (1999) Business failure prediction using UTADIS multicriteria analysis. J Oper Res Soc 50(11):1138–1148
77. Zopounidis C, Doumpos M (2000) PREFDIS: A multicriteria decision support system for sorting decision problems. Comput Oper Res 27(7–8):779–797
78. Zopounidis C, Doumpos M (2000) A preference disaggregation decision support system for financial classification problems. Europ J Oper Res
79. Zopounidis C, Doumpos M, Zanakis SH (1999) Stock evaluation using a preference disaggregation methodology. Decision Sci 30(2):313–336
80. Zopounidis C, Hurson Ch, Doumpos M (2000) Risquepays: Evaluation des aspects economiques, sociaux Et politiques. Economica, Paris
81. Zopounidis C, Matsatsinis NF, Doumpos M (1996) Developing a multicriteria knowledge-based decision support system for the assessment of corporate performance and viability: The FINEVA system. Fuzzy Economic Rev 1(2):35–53

Preference Disaggregation Approach: Basic Features, Examples from Financial Decision Making

CONSTANTIN ZOPOUNIDIS
Department Production Engineering and Management
Financial Engineering Lab., Techn. University Crete
University Campus, Chania, Greece

MSC2000: 90C29

Article Outline

Keywords
Preference Disaggregation Analysis
Preference Disaggregation Analysis in Ranking Problems
Preference Disaggregation Analysis in Sorting Problems
Preference Disaggregation Analysis in Choice Problems
Conclusions and Future Research
See also
References

Keywords

Multicriteria analysis; Preference disaggregation; Linear programming; Financial decision making

Multicriteria decision aid (MCDA) has emerged during the last three decades as a promising scientific field in operations research and management science, and today it has consolidated its position with a wide set of methods and tools confronting in a realistic and flexible way decision problems where multiple conflicting criteria should be considered. The tools provided by MCDA are not just some mathematical models, which aggregate several criteria, points of view or attributes, but furthermore they are decision support oriented. Actually, 'support' is a key issue in MCDA, implying that the models are not developed through a straightforward sequential process where the decision maker's role is passive. Instead an iterative process is employed to analyze the preferences of the decision maker and represent them as consistently as possible in an appropriate decision model. Throughout this process the interaction of the decision maker with the analyst is essential, providing significant preferential information to the analyst. This decision support nature of MCDA is the basic feature that distinguishes it from the classical models that the optimization approach employs.

Within the methods and tools provided by MCDA, several approaches and theoretical disciplines can be defined, although their distinction and the existing boundaries among them are often difficult to determine. In fact several authors provided different categorizations of the MCDA approaches ([19,21,34,36]). Following the proposal of P.M. Pardalos et al. [15] and C. Zopounidis [40], one can identify four major streams in MCDA:

1) *multi-objective programming* ([30,32,35]),
2) *multi-attribute utility theory* [11],
3) *outranking relations* ([18,20]),
4) *preference disaggregation analysis* [10].

The differences among these approaches can be identified in terms of the types of problems that they address, in terms of the preference models that they develop, as well as in term of the process that is employed to develop these models. As far as the type of problem is concerned, multi-objective programming addresses decision problems where there is not a finite set of alternative solutions, but instead the possible alternatives are determined implicitly through a set of constraints imposed by the nature of the problem. On the other hand, multi-attribute utility theory, outranking relations, and preference disaggregation analysis are used

to study decision problems involving the evaluation of a well-defined finite set of alternatives.

Concerning the form of the model that is employed, in multi-objective programming the decision problem is formulated as a mathematical programming model, with more than one objective functions representing the objectives of the decision maker. In both multi-attribute utility theory and preference disaggregation analysis the model is a utility function either additive or multiplicative. Finally, outranking relations are based on pairwise judgments between the alternatives of the form 'alternative *a* is at least as good as alternative *b*'.

Although there are significant differences between the four MCDA approaches regarding the types of problems that they address and the model formulation, the most significant differences involve the information that they require from the decision maker, and the procedure that is used to elicit this information. In multi-objective programming the process employed to develop the model and to obtain the best compromise solution is an interactive one. Once the decision problem has been consistently formulated as a mathematical programming problem, the efficient (nondominated) set of solutions is determined. The decision maker is then asked to provide some preferential information usually in terms of some reference points, indicating the way that the efficient set should be investigated. In multi-attribute utility theory, a direct interrogation process is employed to elicit information from the decision maker concerning the trade-offs among the conflicting criteria, attributes or points of view. These trade-offs are then used to construct the global preference model in the form of a utility function that the decision maker implicitly uses to make decisions. The modeling of the decision makers preferences in outranking relations is achieved similarly to the process used in multi-attribute utility theory (direct interrogation of the decision maker), although the type of information required differ (the decision maker must determine the weights of the evaluation criteria, as well as preference, indifference and veto thresholds). Compared to the other three approaches, preference disaggregation requires the minimal amount of information from the decision maker. The weights, trade-offs, reference points, or any other preferential information does not have to be determined a priori by the decision maker. Instead, the decision maker, based on his/her past experience is asked to provide some characteristic examples of his decision making policy through the evaluation of a 'reference' set of alternatives. Then using ordinal regression techniques, the utility function that has been implicitly used by the decision maker is estimated.

This indirect estimation of the decision makers' preferences in preference disaggregation analysis is a quite appealing characteristic compared to the direct interrogation procedures employed in multi-objective programming, multi-attribute utility theory and outranking relations.

The preference disaggregation analysis found an appropriate field of applications in the domain of *financial decision making*. Most financial decision making problems such as corporate failure prediction, credit granting, portfolio selection and management, venture capital investment, country risk assessment, etc., have a repetitive character [37], while the decisions have to be taken in real time. These two characteristics of financial decision making problems are in accordance with the general methodological framework of preference disaggregation analysis. The repetitive character of financial decision problems enables the decision analyst to exploit the experience and the past decisions of the decision maker (financial/credit analyst, portfolio manager, etc.) in order to develop the appropriate decision model. Once this model is validated, it can be used to support real time financial decision making. Zopounidis [40] provides a comprehensive discussion of the applications of preference disaggregation analysis and multicriteria analysis in general in the study of financial decision making problems, while the books ([38,39]) and [7] illustrate the application of preference disaggregation analysis in venture capital investments, business failure prediction, and portfolio selection and management.

The rest of this article will focus with more details on the basic concepts, principles and techniques used in preference disaggregation analysis. More specifically, Section 2 describes the foundations of preference disaggregation analysis, while Sections 3, 4 and 5 illustrate how the preference disaggregation analysis can be applied in the study of ranking, sorting and choice problems respectively using simple illustrative examples from the field of credit granting. Finally, Section 6

concludes the article and discusses some possible future research direction in this field.

Preference Disaggregation Analysis

The preference disaggregation approach refers to the analysis (disaggregation) of the global preferences of the decision maker to deduce the relative importance of the evaluation criteria, using ordinal regression techniques based mainly on *linear programming* formulations.

The preference disaggregation analysis is based on the simple finding that generally, in real world situations, decision makers are either unable or unwilling to provide in a direct way specific information regarding their preferences including weights or trade-offs. Even if this is possible, the procedure that will be employed to elicit such information from the decision maker is time consuming which may prevent its practical applications in real decision problems where decisions have to be taken in real time.

On the contrary, instead of describing the procedure that leads to the final decision, it would be easier for the decision maker to provide the analyst with the actual decision he/she would take considering the specific characteristics and conditions of the problem at hand.

For instance, when a committee of professors investigates the profiles of candidates for a graduate course, in order to rank them from the most appropriate to the less appropriate ones, the past evaluations that the committee has made can be used. These evaluations can either have the form of a ranking or they may express the intense of preference between two candidates (how many times an alternative is preferred compared to another alternative) [12]. Furthermore, it is even possible to consider more detailed information that the decision maker can provide, for instance the ranking of the alternatives on each evaluation criterion combined with the ranking of the criteria according to their significance [1].

The purpose of gathering such information from the decision maker is to have some representative examples of decisions taken by the decision maker. These examples reflect the decision policy and the preferences that the decision maker has implicitly used in making the decision. Consequently, through the analysis of such decision instances, the analyst can derive useful information concerning the global preference system of the decision maker.

Decision makers when making decisions evaluate each alternative over a set of factors, criteria, attributes or points of view that affect the overall evaluation of the alternatives. Then, these partial evaluations are aggregated to derive the final decision. Following the same approach, the aim of the preference disaggregation analysis is to disaggregate the overall decision into the partial evaluations on each one of the evaluation criteria. The disaggregation should be performed in such a way so that the aggregation of partial evaluations will lead to the overall evaluation that the decision maker provided. If this is not possible then the deviations that occur should be minimized.

In such a disaggregation process it is clear that the form that the partial evaluations will have, as well as the selection of the model which will be used to aggregate the partial evaluations are two key issues. The first preference disaggregation approaches employed a simple weighted sum model of the form:

$$\sum w_j d_{ij} ,$$

where w_j denotes the weight of each criterion and d_{ij} denotes the distance of the evaluation of an alternative i on criterion j from the ideal point for this criterion ([6,16,31]). However, such a model is just an oversimplification of real world situations, which suffers from two major drawbacks.

- Firstly, it is obvious that this formulation implies both the overall value (score) of an alternative as well as the partial value of the alternative on an evaluation criterion, are linear functions (the weights of the evaluation criteria are independent on the criteria's values).
- Furthermore, this model is not appropriate for considering criteria which are measured through a qualitative scale. This would require the transformation of this qualitative scale into a numerical one, which can not be unique and it may not be in accordance with the preferential information that a qualitative scale provides.

To overcome such limitations, utility functions may be used. *Utility functions* are nonlinear increasing or decreasing functions of the criteria's values, indicating the value of the alternatives in the global preference system

of the decision maker. The most common forms of utility functions used in practice include the additive form and the multiplicative form. An additive utility function can be expressed as:

$$U(\overline{g}) = \sum_j u_j(g_j) \,,$$

whereas a multiplicative utility function can be expressed as:

$$U(\overline{g}) = \frac{\prod_j (k u_j(g_j) + 1) - 1}{k} \,,$$

where $U(\overline{g})$ denotes the global utility of an alternative described by the vector of criteria $\overline{g}$, $u_j(g_j)$ is the partial or marginal utility of an alternative on criterion g_j, and k is a scaling factor. R.L. Keeney and H. Raiffa in their book [11] provide a comprehensive discussion of utility theory and the underlying assumptions of the several types of utility functions.

The aim of preference disaggregation analysis is to estimate the marginal utilities of each evaluation criterion so that their aggregation using either an additive or a multiplicative utility function results in an evaluation of the alternatives which is consistent with the decision maker's preferences and judgement policy. This estimation is achieved through mathematical programming techniques with the objective being the optimization of a measure of consistency. Multiplicative utility functions can generally be more appropriate for modeling decision makers' preferences in real world decisions taking into account possible interactions among the decision makers' preference on several criteria [14]. However, their estimation results in nonlinear programs that are computationally intensive and difficult to solve. Consequently, in practice additive utility functions are commonly used instead of multiplicative ones, since they provide a simple but also powerful approach for modeling decision makers' preference in multiple criteria decision making problems.

A well known preference disaggregation method which incorporates additive utility functions to model decision maker's preferences is the UTA method (UTilités Additives) proposed in [10]. The subsequent three subsections illustrate the UTA method as it has been proposed in [10], as well as some of its variants which have been proposed for the study of ranking, sorting and choice decision problems. More specifically the UTASTAR method [28] for ranking problems, the UTADIS method ([3,9,42]) for *sorting problems*, as well as two methodologies proposed in [22] and [33] for choice problems are presented.

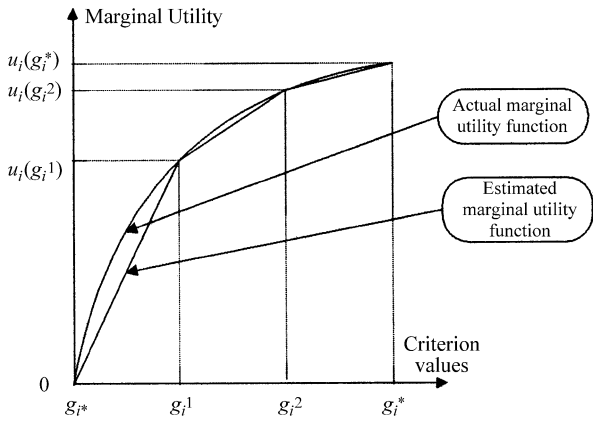
Preference Disaggregation Analysis in Ranking Problems

The UTA method performs an ordinal regression based on the preference disaggregation approach of MCDA. Following the general methodological framework of preference disaggregation analysis, the decision maker is asked to provide a ranking (pre-ordering) of a reference set of alternatives which will be used to construct the additive utility model. This reference set may consist of examples of past decisions, or of a subset of the alternatives under consideration for which the decision maker can express a global evaluation.

Given this pre-ordering, the aim of the UTA method is to estimate a set of additive utility functions which are as consistent as possible with the decision maker's preferences (pre-ordering). The marginal utilities are piecewise linear (Fig. 1). The range of values of each criterion is divided into a_{i-1} equal intervals. The number of these subintervals can be specified by the decision maker, or it can be determined by the analyst so that there is at least one alternative falling into each subinterval. The estimation of the marginal utilities is achieved through the following linear programming formulation:

$$\begin{cases} \min & F = \sum_{a \in A} \sigma(a) \\ \text{s.t.} & U(a) - U(b) + \sigma(a) - \sigma(b) \geq \delta \\ & \quad \text{if } aPb, \ \forall a, b \in A, \\ & U(a) - U(b) + \sigma(a) - \sigma(b) = 0 \\ & \quad \text{if } aIb, \ \forall a, b \in A, \\ & u_i(g_i^{j+1}) - u_i(g_i^j) \geq 0, \\ & \sum_i u_i(g_{i*}) = 1, \quad u_i(g_{i*}) = 0, \quad \forall i, \end{cases}$$

where P and I represent the preference and the indifference relations, respectively, A is the set of reference alternatives used to develop the additive utility model, $U(a) = \sum_{i=1}^{n} u_i[g_i(a)]$ is the global utility of an alternative $a \in A$, $\sigma(a)$ is an error function ($\sigma(a) \geq 0$), and δ is a threshold used to ensure the strict preference of an alternative a over an alternative b ($\delta \geq 0$).



Preference Disaggregation Approach: Basic Features, Examples from Financial Decision Making, Figure 1
Piecewise linear form of marginal utilities

Solving the above linear program a utility function is estimated which minimizes the deviations $\sigma(a)$. However, it has been observed that in many cases this utility function is not the most consistent one with the pre-ordering provided by the decision maker. On the contrary, utility functions that correspond to sub-optimal solutions of the above linear program often provide a more consistent representation of the decision maker's preferences. Furthermore, the above linear program has often multiple optimal solutions (degeneracy). Therefore, in order to examine these two essential issues, in a second stage the UTA method proceeds to a post-optimality analysis, trying to find some characteristic sub-optimal or multiple optimal solutions. A heuristic post-optimality procedure that has been proposed by E. Jacquet-Lagrèze and Y. Siskos [10] finds the solutions which correspond to extreme weights of the evaluation criteria. This is achieved by incorporating an additional constraint of the form: $F \leq F^* + k(F^*)$, where F^* is the optimal value of the objective function obtained by solving the above linear program and $k(F^*)$ is a small portion of it. The new linear program that is obtained is solved with the objective being the minimization or the maximization of the weight of each criterion:

$$\begin{cases} \min \text{ (or } \max) & \displaystyle\sum_{i=1}^{n} \rho_i u_i(g_{i*}) \\ \text{with} \quad \rho_i = 0 \text{ or } 1, & \forall i. \end{cases}$$

Siskos and D. Yannacopoulos [28] proposed the UTASTAR method, an improved variant of the UTA method. The differences between the two methods can be identified in the following two aspects:

1) Instead of using the marginal utilities $u_i(g_i^j)$ as decision variables, the difference w_{ij} between the marginal utilities of two successive values of a criterion is used: $w_{ij} = u_i(g_i^{j+1}) - u_i(g_i^j) \geq 0$. In this way the constraints $u_i(g_i^{j+1}) - u_i(g_i^j) \geq 0$ are transformed in nonnegativity constraints $w_{ij} \geq 0$.
2) Two error functions are used instead of one. The two error functions denoted as $\sigma^+(a)$ and $\sigma^-(a)$ represent the overestimation or underestimation error that may occur in the decision maker's pre-ordering. The overestimation error involves alternatives that have been ranked by the decision maker higher than their rank according to the additive utility model, while the underestimation error involves alternatives which have been ranked by the decision maker lower than their rank according to the additive utility model.

Considering these two differences the new linear program that is solved is the following:

$$\begin{cases} \min & \displaystyle\sum_{a \in A} [\sigma^+(a) - \sigma^-(a)] \\ \text{s.t.} & U(a) - U(b) + \sigma^+(a) - \sigma^-(a) \\ & \quad -\sigma^+(b) + \sigma^-(b) \geq \delta \\ & \quad \text{if } aPb, \\ & U(a) - U(b) + \sigma^+(a) - \sigma^-(a) \\ & \quad -\sigma^+(b) + \sigma^-(b) = 0 \\ & \quad \text{if } aIb, \\ & \displaystyle\sum_i \sum_j w_{ij} = 1, \\ & w_{ij}, \sigma^+(a), \sigma^-(a) \geq 0, \ \forall a \in A, \forall i, j, \\ & u_i(g_i^k) = \displaystyle\sum_{k=1}^{a_i - 1} w_{ik}, \quad \forall i, j, k. \end{cases}$$

The UTA and the UTASTAR methods have been applied successfully in a variety of decision problems, including environmental decisions [24], marketing decisions and sales strategy problems ([13,17,23]), customer satisfaction [25], venture capital investments [29], country risk assessment [2], evaluation of bankruptcy risk [37], as well as research and development decisions [5].

Preference Disaggregation Approach: Basic Features, Examples from Financial Decision Making, Table 1
Evaluations of the firms

	EBIT/TA	NU/NW	TL/TA	(CA-I)/CL
F_1	10%	24%	33%	2.97
F_2	5%	9%	78%	1.11
F_3	13%	18%	80%	0.84
F_4	15%	−8%	72%	0.85
F_5	8%	−30%	93%	0.60

In the subsequent subsection, a simple example is used to illustrate how the UTASTAR method can be applied in the study of ranking problems.

Example 1 Consider a decision problem concerning credit granting. Five firms (F_1, F_2, F_3, F_4, F_5) are seeking financing by a bank. The credit managers of the bank evaluate the firms along four financial ratios:

1) earnings before interest and taxes/total assets (EBIT/TA);
2) net income/net worth (NI/NW);
3) total liabilities/total assets (TL/TA); and
4) (current assets-inventories)/current liabilities [(CA-I)/CL].

For the ratios EBIT/TA, NI/NW and (CA-I)/CL the preferences of the credit managers are increasing functions of their values. Hence, the higher the values of these ratios the more creditworthy a firm is. On the contrary, for the ratio TL/TA the preference of the credit managers is a decreasing function of its value, since high values of this ratio mean that the firm is highly indebted. Table 1 illustrates the evaluations of the firms along these four criteria.

According to the credit policy of the bank, the following preferential structure (pre-ordering) is defined:

$$F_1 \; P \; F_2 \; P \; F_3 \; P \; F_4 \; P \; F_5 \; .$$

The first step of the UTASTAR method consists of making explicit the utilities of the alternatives (firms). The range of values of each criterion (financial ratio) is divided into a number of subintervals, so that there is at least one firm belonging to each interval. Following this rule the following scales are retained:

$$[g_{1*}, g_1^*] = [5\%, 7.5\%, 10\%, 12.5\%, 15\%] \; ,$$
$$[g_{2*}, g_2^*] = [-30\%, -16.5\%, -3\%, 10.5\%, 24\%] \; ,$$
$$[g_{3*}, g_3^*] = [93\%, 73\%, 53\%, 33\%] \; ,$$
$$[g_{4*}, g_4^*] = [0.6, 1.785, 2.97] \; .$$

Using linear interpolation the global utilities of the firms can be written as follows:

$$\begin{aligned}
U(F_1) &= u_1(10\%) + u_2(24\%) + u_3(33\%) + u_4(2.97) \; , \\
U(F_2) &= u_1(5\%) + 0.11u_2(-3\%) + 0.89u_2(10.5\%) \\
&\quad + 0.25u_3(93\%) + 0.75u_3(73\%) \\
&\quad + 0.57u_4(0.6) + 0.43u_4(1.785) \; , \\
U(F_3) &= 0.8u_1(12.5\%) + 0.2u_1(15\%) \\
&\quad + 0.44u_2(10.5\%) + 0.56u_2(24\%) \\
&\quad + 0.35u_3(93\%) + 0.65u_3(73\%) \\
&\quad + 0.80u_4(0.6) + 0.20u_4(1.785) \; , \\
U(F_4) &= u_1(15\%) + 0.37u_2(-16.5\%) \\
&\quad + 0.63u_2(-3\%) \\
&\quad + 0.95u_3(73\%) + 0.05u_3(53\%) \\
&\quad + 0.79u_4(0.6) + 0.21u_4(1.785) \; , \\
U(F_5) &= 0.8u_1(7.5\%) + 0.2u_1(10\%) \\
&\quad + u_2(-30\%) + u_3(93\%) + u_4(0.6) \; .
\end{aligned}$$

Using the transformation $u_i(g_i^k) = \sum_{k=1}^{a_i-1} w_{ik}$, the global utilities can be expressed as:

$$\begin{aligned}
U(F_1) &= w_{11} + w_{12} + w_{21} + w_{22} + w_{23} \\
&\quad + w_{24} + w_{31} + w_{32} + w_{33} + w_{41} + w_{42} \; , \qquad (1)
\end{aligned}$$

$$U(F_2) = w_{21} + w_{22} + 0.89w_{23} + 0.75w_{31} + 0.43w_{41} \; , \qquad (2)$$

$$\begin{aligned}
U(F_3) &= w_{11} + w_{12} + w_{13} + 0.20w_{14} \\
&\quad + w_{21} + w_{22} + w_{23} + 0.56w_{24} \\
&\quad + 0.65w_{31} + 0.20w_{41} \; , \qquad (3)
\end{aligned}$$

$$\begin{aligned}
U(F_4) &= w_{11} + w_{12} + w_{13} + w_{14} + w_{21} \\
&\quad + 0.63w_{22} + w_{31} + 0.05w_{32} + 0.21w_{41} \; , \\
U(F_5) &= w_{11} + 0.2w_{12} \; . \qquad (4)
\end{aligned}$$

Taking into account the pre-ordering that was defined, and the global utilities of the firms the following linear program is formulated ($\delta = 0.05$):

$$\min\{\sigma^+(F_1) + \sigma^-(F_1) + \sigma^+(F_2) + \sigma^-(F_2) + \sigma^+(F_3) + \sigma^-(F_3) + \sigma^+(F_4) + \sigma^-(F_4) + \sigma^+(F_5) + \sigma^-(F_5)\}$$

s.t.

$$w_{11} + w_{12} + 0.11w_{23} + w_{24} + 0.25w_{31} + w_{32} + w_{33} + 0.57w_{41} + w_{42} + \sigma^+(F_1) - \sigma^-(F_1) - \sigma^+(F_2) + \sigma^-(F_2) \geq 0.05\,, \tag{5}$$

$$-w_{11} - w_{12} - w_{13} - 0.2w_{14} - 0.11w_{23} - 0.56w_{24} + 0.1w_{31} + 0.23w_{41} + \sigma^+(F_2) - \sigma^-(F_2) - \sigma^+(F_3) + \sigma^-(F_3) \geq 0.05\,, \tag{6}$$

$$-0.8w_{14} + 0.37w_{22} + w_{23} + 0.56w_{24} - 0.35w_{31} - 0.05w_{32} - 0.01w_{41} + \sigma^+(F_3) - \sigma^-(F_3) - \sigma^+(F_4) + \sigma^-(F_4) \geq 0.05\,, \tag{7}$$

$$0.8w_{12} + w_{13} + w_{14} + w_{21} + 0.63w_{22} + w_{31} + 0.05w_{32} + 0.21w_{41} + \sigma^+(F_4) - \sigma^-(F_4) - \sigma^+(F_5) + \sigma^-(F_5) \geq 0.05\,, \tag{8}$$

$$w_{11} + w_{12} + w_{13} + w_{14} + w_{21} + w_{22} + w_{23} + w_{24} + w_{31} + w_{32} + w_{33} + w_{41} + w_{42} = 1\,. \tag{9}$$

Constraint (5) involves the pairwise comparison of F_1 with F_2, constraint (6) involves the pairwise comparison of F_2 with F_3, constraint (7) involves the pairwise comparison of F_3 with F_4, constraint (8) involves the pairwise comparison F_4 with F_5, while constraint (9) is used to normalize the global utilities between 0 and 1. The optimal solution to this linear problem is as follows. (The solver of Microsoft Excel has been used to solve all the presented linear programs. Since most of the solutions presented are not unique, different solutions may be obtained through other linear programming packages depending upon their particularities and vagaries.)

$$w_{13} = 0.0152,\ w_{14} = 0.0357\,,$$
$$w_{21} = 0.0758,\ w_{22} = 0.0786\,,$$
$$w_{23} = 0.0959,\ w_{24} = 0.011\,,$$
$$w_{31} = 0.1301,\ w_{32} = 0.0754,\ w_{33} = 0.0758\,,$$
$$w_{41} = 0.3306,\ w_{42} = 0.0758\,.$$

This solution is fully consistent with the predefined pre-ordering. More specifically, the global utilities obtained through this solution are:

$$U(F_1) = 0.9491,\ U(F_2) = 0.4795\,,$$
$$U(F_3) = 0.4295,\ U(F_4) = 0.3795\,,$$
$$U(F_5) = 0\,.$$

In a second stage, through the post-optimality analysis, the existence of multiple optimal solutions is investigated. In order to find the most characteristic solutions the weight of each criterion is maximized. This is achieved by solving four new linear programs with the objectives being the maximization of $w_{11} + w_{12} + w_{13} + w_{14}$, $w_{21} + w_{22} + w_{23} + w_{24}$, $w_{31} + w_{32} + w_{33}$, and $w_{41} + w_{42}$, respectively. For instance, the linear program to maximize the weight of the first criterion (EBIT/TA) would be the following:

$$\max\{w_{11} + w_{12} + w_{13} + w_{14}\}$$

s.t.

$$w_{11} + w_{12} + 0.11w_{23} + w_{24} + 0.25w_{31} + w_{32} + w_{33} + 0.57w_{41} + w_{42} \geq 0.05\,,$$
$$-w_{11} - w_{12} - w_{13} - 0.2w_{14} - 0.11w_{23} - 0.56w_{24} + 0.1w_{31} + 0.23w_{41} \geq 0.05\,,$$
$$-0.8w_{14} + 0.37w_{22} + w_{23} + 0.56w_{24} - 0.35w_{31} - 0.05w_{32} - 0.01w_{41} \geq 0.05\,,$$
$$0.8w_{12} + w_{13} + w_{14} + w_{21} + 0.63w_{22} + w_{31} + 0.05w_{32} + 0.21w_{41} \geq 0.05\,,$$
$$w_{11} + w_{12} + w_{13} + w_{14} + w_{21} + w_{22} + w_{23} + w_{24} + w_{31} + w_{32} + w_{33} + w_{41} + w_{42} = 1\,.$$

Solving this linear program the obtained solution is: $w_{14} = 0.2296$, $w_{23} = 0.2390$, $w_{41} = 0.5314$. Similarly solving the other three linear programs the following solutions are obtained:

For $\max\{w_{21} + w_{22} + w_{23} + w_{24}\}$:

$$w_{21} = 0.4297,\ w_{22} = 0.3529,\ w_{41} = 0.2174\ .$$

For $\max\{w_{31} + w_{32} + w_{33}\}$:

$$w_{23} = 0.0524,\ w_{33} = 0.7051,\ w_{41} = 0.2425\ .$$

For $\max\{w_{41} + w_{42}\}$:

$$w_{23} = 0.0524,\ w_{41} = 0.2425,\ w_{42} = 0.7051\ .$$

A simple way to select a unique utility function is to consider the mean of all the solutions obtained through the post-optimality stage:

$$w_{14} = 0.0574,\ w_{21} = 0.1074\ ,$$
$$w_{22} = 0.0882,\ w_{23} = 0.0860\ ,$$
$$w_{33} = 0.1763,\ w_{41} = 0.3084,\ w_{42} = 0.1763\ .$$

Consequently, the marginal utilities of the financial ratios are:

$$u_1(5\%) = u_1(7.5\%) = u_1(10\%) = u_1(12.5\%) = 0\ ,$$
$$u_1(15\%) = 0.0574\ ,$$
$$u_2(-30\%) = 0,\ u_2(-16.5\%) = 0.1074\ ,$$
$$u_2(-3\%) = 0.1957\ ,$$
$$u_2(10.5\%) = 0.2816\ u_2(24\%) = 0.2816\ ,$$
$$u_3(93\%) = u_3(73\%) = u_3(53\%) = 0\ ,$$
$$u_3(33\%) = 0.1763\ .$$
$$u_4(0.6) = 0,\ u_4(1.785) = 0.3084\ ,$$
$$u_4(2.97) = 0.4847\ .$$

According to this solution the global utilities of the firms are:

$$U(F_1) = 0.9426,\ U(F_2) = 0.4048\ ,$$
$$U(F_3) = 0.3548,\ U(F_4) = 0.2852,\ U(F_5) = 0\ .$$

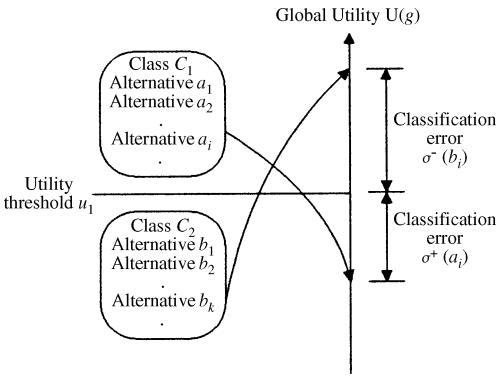
It is obvious that the ranking of the firms according to their global utilities is fully consistent with the predefined pre-ordering. Furthermore, the bank can use the obtained additive utility function to evaluate any new firm.

Preference Disaggregation Analysis in Sorting Problems

The preference disaggregation analysis except for the study of ranking problems can also be used to study *sorting problems*. In this case, the primary objective is not to rank a set of alternatives from the best ones to the worst ones, but instead the aim is to sort the alternatives into two or more predefined ordered homogeneous classes denoted as $C_1, \ldots, C_q$ (C_1 is the class of the best alternatives, and C_q is the class of the worst alternatives). The sorting of the alternatives can be accomplished in several ways. The most simple and common one is based on the comparison of the score of each alternative with some thresholds $(u_1, \ldots, u_{q-1})$ which distinguish the classes. Following the methodological framework of the UTA method the score of each alternative is its global utility $U(a)$. The sorting of an alternative a is accomplished through the following comparisons:

$$\begin{aligned}
U(a) \geq u_1 \quad &\Rightarrow \quad a \in C_1\ ,\\
u_2 \leq U(a) < u_1 \quad &\Rightarrow \quad a \in C_2\ ,\\
&\ldots\\
u_k \leq U(a) < u_{k-1} \quad &\Rightarrow \quad a \in C_k\ ,\\
&\ldots\\
U(a) < u_{q-1} \quad &\Rightarrow \quad a \in C_q\ .
\end{aligned}$$

J.M. Devaud et al. [3] (see also [9] and [42]) proposed the UTADIS method (UTilités Additives DIScriminantes) a variant of the UTA method which is especially conceived for the study of sorting problems. The objective of the method is to estimate a global utility model (additive utility function) and the utility thresholds in order to minimize the classification error. The classification error is measured through two error



Preference Disaggregation Approach: Basic Features, Examples from Financial Decision Making, Figure 2
Classification errors

functions denoted as $\sigma^+(a)$ and $\sigma^-(a)$, representing the deviations of a misclassified alternative from the utility threshold. Figure 2 illustrates these two types of errors for the two-group (class) sorting problem.

In the UTADIS method the additive utility model and the utility thresholds are estimated through the following linear program (using the transformation $w_{ij} = u_i(g_i^{j+1}) - u_i(g_i^j)$ proposed in [28]):

$$
\begin{cases}
\min \quad F = \displaystyle\sum_{a \in C_1} \sigma^+(a) + \cdots \\
\qquad\quad + \displaystyle\sum_{a \in C_k} [\sigma^+(a) + \sigma^-(a)] + \cdots \\
\qquad\quad + \displaystyle\sum_{a \in C_q} \sigma^+(a) \\
\text{s.t.} \quad U(a) - u_1 + \sigma^+(a) \geq 0, \ \forall a \in C_1, \\
\qquad U(a) - u_{k-1} - \sigma^-(a) \leq -\delta, \ \forall a \in C_k, \\
\qquad U(a) - u_k + \sigma^+(a) \geq 0, \ \forall a \in C_k, \\
\qquad U(a) - u_{q-1} - \sigma^-(a) \leq -\delta, \ \forall a \in C_q, \\
\qquad \displaystyle\sum_{i=1}^{m} \sum_{j=1}^{a_i - 1} w_{ij} = 1, \\
\qquad u_{k-1} - u_k \geq s, \ k = 2, \ldots, q-1, \\
\qquad w_{ij} \geq 0, \ \sigma^+(a) \geq 0, \ \sigma^-(a) \geq 0.
\end{cases}
$$

The first constraint implies that the global utility of an alternative $a \in C_1$ should be greater or equal to the utility threshold u_1. If this is not possible, then an amount of utility equal to $\sigma^+(a)$ should be added to the global utility of this alternative, indicating that the alternative is classified to a lower class than the one it actually belongs (cf. Fig. 2). The second set of constraints is used for alternatives that are classified by the decision maker in an intermediate class C_k. In such cases, the global utility of the alternative should be strictly lower than the utility threshold u_{k-1} (a positive small real number δ is used to ensure the strict inequality) and greater or equal to the utility threshold u_k. If either of these two conditions is not satisfied then the corresponding amount of utility $\sigma^+(a)$ or $\sigma^-(a)$ should be added (subtracted) to the global utility of the alternative. Similarly, the third constraint is used for alternatives which belong to the worst class C_q. The global utility of these alternatives should be strictly lower than the utility threshold u_{q-1}; otherwise an amount of utility equal to $\sigma^-(a)$ should be subtracted from the global utility of the alternatives, indicating that these alternatives are classified by the model to a higher (better) class than the one they actually belong (cf. Fig. 2). The fourth constraint is used as a normalization constraint, so that the global utilities and the utility thresholds are normalized between 0 and 1. Finally, the fifth constraint is used to ensure that $u_1 > \cdots > u_{q-1}$ (a positive real number $s > \delta > 0$ is used to ensure the strict inequality between the utility thresholds). The post-optimality analysis stage is performed similarly to the UTA method.

In [4] and [41] three variants of the UTADIS method were proposed to improve the classification accuracy of the obtained additive utility models. The first variant (UTADIS I) except for the classification errors also incorporates the distances of the correctly classified alternatives from the utility thresholds which have to be maximized. The second variant (UTADIS II) is based on a mixed integer programming formulation minimizing the number of misclassifications instead of their magnitude, while the third variant (UTADIS III) combines UTADIS I and II, and its aim is to minimize the number of misclassifications and maximize the distances of the correctly classified alternatives from the utility thresholds.

Except for the classification of the alternatives in the predefined classes, the global utilities of the alternatives that are estimated through the family of the UTADIS methods can be used to rank the alternatives belonging in each class. Hence, the decision maker is provided with an additional information that can be used to identify the best and the worst alternatives within each class. The applications of the UTADIS method include sales strategy problems [3], the evaluation of research and development projects [9], and several financial decision problems such as credit risk and bankruptcy risk evaluation, country risk assessment, credit card evaluation, evaluation of bank branches' efficiency, and portfolio selection and management ([4,41,42]). In the subsequent subsection, a simple example is used to illustrate how the UTADIS method can be applied in the study of sorting problems.

Example 2 Let us consider again the credit granting problem which has been used previously to illustrate the application of the UTASTAR method. In this case we assume that the credit managers of the bank are not interested in ranking the firms from the most creditworthy to the less creditworthy ones, but instead they

are interested in sorting them in two classes: the creditworthy firms which could be financed by the bank (class C_1), and the untrustworthy firms which should not be financed (class C_2). Suppose according to the credit policy of the bank, that the firms F_1, F_2 and F_3 can be considered as creditworthy firms, whereas the firms F_4 and F_5 can be considered as risky (untrustworthy) firms. The global utilities of the firms have the same form as in the case of the UTASTAR method (1)–(4). The new linear program that is used to estimate the additive utility model is the following:

$$\min \sigma^+(F_1) + \sigma^+(F_2) + \sigma^+(F_3) + \sigma^-(F_4) + \sigma^-(F_5)$$

s.t.

$$\begin{aligned} &w_{11} + w_{12} + w_{21} + w_{22} + w_{23} + w_{24} \\ &+ w_{31} + w_{32} + w_{33} \\ &\quad + w_{41} + w_{42} - u_1 + \sigma^+(F_1) \geq 0\,, \end{aligned} \tag{10}$$

$$\begin{aligned} &w_{21} + w_{22} + 0.89w_{23} + 0.75w_{31} \\ &\quad + 0.43w_{41} - u_1 + \sigma^+(F_2) \geq 0\,, \end{aligned} \tag{11}$$

$$\begin{aligned} &w_{11} + w_{12} + w_{13} + 0.20w_{14} \\ &+ w_{21} + w_{22} + w_{23} + 0.56w_{24} \\ &+ 0.65w_{31} + 0.20w_{41} - u_1 + \sigma^+(F_3) \geq 0\,, \end{aligned} \tag{12}$$

$$\begin{aligned} &w_{11} + w_{12} + w_{13} + w_{14} \\ &+ w_{21} + 0.63w_{22} + w_{31} + 0.05w_{32} \\ &\quad + 0.21w_{41} - u_1 - \sigma^-(F_4) \leq 0.001\,, \end{aligned} \tag{13}$$

$$w_{11} + 0.2w_{12} - u_1 - \sigma^-(F_5) \leq 0.001\,, \tag{14}$$

$$\begin{aligned} &w_{11} + w_{12} + w_{13} + w_{14} \\ &+ w_{21} + w_{22} + w_{23} + w_{24} \\ &\quad + w_{31} + w_{32} + w_{33} + w_{41} + w_{42} = 1\,. \end{aligned} \tag{15}$$

Constraints (10)–(12) involve the firms F_1, F_2 and F_3 which belong in class C_1 and consequently their global utility should be greater or equal to the utility threshold u_1. On the contrary, constraints (13)–(14) involve the firms F_4 and F_5 which belong in class C_2 and consequently their global utilities should be strictly lower than the utility threshold u_1 (a small positive number $\delta = 0.001$ is used to ensure the strict inequality). Constraint (15) is used similarly to the UTASTAR method for normalization purposes. Solving the above linear program the following solution is obtained:

$$w_{22} = 0.1779,\ w_{23} = 0.8018,\ w_{31} = 0.0203\,.$$

The estimated utility threshold is $u_1 = 0.9067$. According to this solution the global utilities of the firms are:

$$\begin{aligned} &U(F_1) = 1,\ U(F_2) = 0.9067\,, \\ &U(F_3) = 0.9929,\ U(F_4) = 0.1324,\ U(F_5) = 0\,. \end{aligned}$$

Obviously, all the firms are correctly classified into their original class, however the obtained solution is not unique. Consequently, the method proceeds to the post-optimality analysis stage to identify the most characteristic multiple optimal solutions. This is achieved similarly to the UTASTAR method. However in this case, five new linear programs are solved. The first four correspond to the maximization of the criteria's weights (as in the UTASTAR method), while the fifth corresponds to the maximization of the utility threshold u_1. For example, the linear programming formulation for maximizing the weight of the ratio EBIT/TA is the following:

$$\min\{w_{11} + w_{12} + w_{13} + w_{14}\}$$

s.t.

$$\begin{aligned} &w_{11} + w_{12} + w_{21} + w_{22} + w_{23} + w_{24} \\ &+ w_{31} + w_{32} + w_{33} \\ &+ w_{41} + w_{42} - u_1 \geq 0\,, \\ &w_{21} + w_{22} + 0.89w_{23} \\ &+ 0.75w_{31} + 0.43w_{41} - u_1 \geq 0\,, \\ &w_{11} + w_{12} + w_{13} + 0.20w_{14} \\ &+ w_{21} + w_{22} + w_{23} + 0.56w_{24} \\ &+ 0.65w_{31} + 0.20w_{41} - u_1 \geq 0\,, \\ &w_{11} + w_{12} + w_{13} + w_{14} + w_{21} + 0.63w_{22} \\ &+ w_{31} + 0.05w_{32} + 0.21w_{41} - u_1 \leq -0.001\,, \\ &w_{11} + 0.2w_{12} - u_1 \leq -0.001\,, \\ &w_{11} + w_{12} + w_{13} + w_{14} \\ &+ w_{21} + w_{22} + w_{23} + w_{24} \\ &+ w_{31} + w_{32} + w_{33} + w_{41} + w_{42} = 1\,. \end{aligned}$$

The solution to this linear program is:

$$w_{14} = 0.4704,\ w_{23} = 0.5296\ ,$$

and the utility threshold is $u_1 = 0.4714$.

Similarly solving the other four linear programs the following solutions are obtained.

For $\max\{w_{21} + w_{22} + w_{23} + w_{24}\}$:

$$w_{21} = 0.2772,\ w_{22} = 0.1619\ ,$$
$$w_{23} = 0.2837,\ w_{24} = 0.2772\ ,$$
$$u_1 = 0.3802\ .$$

For $\max\{w_{31} + w_{32} + w_{33}\}$:

$$w_{23} = 0.0011,\ w_{33} = 0.9989,\ u_1 = 0.001\ .$$

For $\max\{w_{41} + w_{42}\}$:

$$w_{23} = 0.0010,\ w_{41} = 0.0005\ ,$$
$$w_{42} = 0.9985,\ u_1 = 0.0011\ .$$

For $\max\{u_1\}$:

$$w_{21} = 0.9861,\ w_{22} = 0.0139\ ,$$
$$u_1 = 1\ .$$

Similarly to the UTASTAR method, the mean of these solutions is considered as a unique solution:

$$w_{14} = 0.0941,\ w_{21} = 0.2527\ ,$$
$$w_{22} = 0.0352,\ w_{23} = 0.1631\ ,$$
$$w_{24} = 0.0554,\ w_{33} = 0.1998\ ,$$
$$w_{41} = 0.0001,\ w_{42} = 0.1997\ .$$

The utility threshold is $u_1 = 0.3707$. Consequently, the marginal utilities of the financial ratios are:

$$u_1(5\%) = u_1(7.5\%) = u_1(10\%) = u_1(12.5\%) = 0\ ,$$
$$u_1(15\%) = 0.0941\ .$$
$$u_2(-30\%) = 0,\ u_2(-16.5\%) = 0.2527\ ,$$
$$u_2(-3\%) = 0.2878,\ u_2(10.5\%) = 0.4509\ ,$$
$$u_2(24\%) = 0.5064\ .$$
$$u_3(93\%) = u_3(73\%) = u_3(53\%) = 0\ ,$$
$$u_2(33\%) = 0.1998\ .$$
$$u_4(0.6) = 0,\ u_4(1.785) = 0.0001\ ,$$
$$u_4(2.97) = 0.1998\ .$$

According to this solution the global utilities of the firms are:

$$U(F_1) = 0.9059,\ U(F_2) = 0.4330\ ,$$
$$U(F_3) = 0.5008,\ U(F_4) = 0.3689,\ U(F_5) = 0\ .$$

It is obvious that the sorting of the firms according to their global utilities is fully consistent with the predefined classification. The utility function that has been constructed through the above procedure is the following:

$$U(g) = 0.0941u_1(g_1) + 0.5064u_2(g_2) + 0.1998u_3(g_3) + 0.1998u_4(g_4)\ .$$

Preference Disaggregation Analysis in Choice Problems

Except for the study of ranking and sorting problems the preference disaggregation analysis is also applicable in choice problems. In this case the decision maker is concerned with the selection of the most appropriate alternative. T.J. Stewart [33] demonstrated how the preference disaggregation analysis could be applied in such types of problems. The methodology that he proposes combines the UTA method with an interactive pruning of the alternatives until the best alternative is selected. The proposed approach proceeds in five steps:

1) Initially, a subset of the alternatives under consideration is selected.
2) Following the methodology of the UTA method, the decision maker is asked to provide a ranking (preordering) of the subset of the alternatives that was selected in step 1.
3) According to the ranking defined in the previous step and using the UTA method the corresponding utility function is estimated. In the same step, for each alternative a not belonging to the reference set used to construct the additive utility model, a linear program is solved in order to investigate the existence of a utility function for which $U(a) > U(o)$ (where o is the best alternative among the alternatives belonging to the reference set) while retaining the correct (consistent) ranking of the other alternatives of the reference set. If such a utility function does not exist, then a can be eliminated from further consideration.

4) If such a utility function exists then, in step 4, the utilities of all the alternatives (based on the utility function developed on the reference set of alternatives) are presented to the decision maker along with all the alternatives not belonging to the reference set that can be considered as best alternatives (for which there is a utility function that makes them the best alternatives).
5) If the decision maker is satisfied with the best solution then the process stops, otherwise in step 5 the decision maker selects the alternative not belonging to the reference set which outperforms the current best alternative among the reference set. This new alternative is included in the initial reference set and the decision maker is asked to provide a new ranking of the alternatives in the new reference set. The methodology proceeds from step 3, until all the alternatives not belonging into the reference set have been considered.

Another way or applying the preference disaggregation approach in choice problems has been proposed in [22]. This approach is based on constructing a fuzzy outranking relation [20] using the set of utilities estimated through the post-optimality analysis performed through the UTA method. The relation suggested is based on the calculation of the percentage of utilities for which an alternative is better than another. In this way the degree of credibility of the affirmation 'alternative a is at least as good as alternative b' is estimated. The construction of this outranking relation enables the decision maker to identify the best alternative(s) which are not outranked by any other alternative(s), while furthermore, the incomparabilities which may occur among the alternatives due to their dissimilar characteristics are identified.

Conclusions and Future Research

In this article the main features, disciplines and characteristics of the preference disaggregation approach of MCDA were presented. Furthermore, the applicability of the preference disaggregation analysis in the study of several types of problems (ranking, sorting, and choice) was demonstrated through the description of the UTA method and several of its variants.

Preference disaggregation analysis using minimal information from the decision maker constitutes a promising and powerful approach for modeling the decision makers' preference and developing a global preference model through an interactive and iterative procedure.

Multicriteria decision support systems (MCDSSs) implementing this MCDA approach can be very helpful tools both for decision analysts as well as for decision makers in making decisions in real world situations where the time is limited while the cost of taking a decision is a very significant factor. Researchers in this field have already explored the potentials of MCDSSs incorporating preference disaggregation procedures. Some representative examples are the PREFCALC system [8], the MINORA system [26], the MIIDAS system [27], the PREFDIS system [44], the FINCLAS system [43] and the FINEVA system [45] for corporate assessment, as well as the MARKEX system [13] for marketing decisions. Recently (1998), researchers have also investigated the contribution of artificial intelligence techniques in the framework of an integrated decision support system (intelligent multicriteria decision support systems), combining the preference disaggregation approach as well as other MCDA approaches with the capabilities provided by expert systems and neural networks. The appealing features of such an integration, constitute a significant field of further research both in preference disaggregation analysis as well as in MCDA in general.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality

- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Modeling

References

1. Cook WD, Kress M (1991) A multiple criteria decision model with ordinal preference data. Europ J Oper Res 54:191–198
2. Cosset J, Siskos Y, Zopounidis C (1992) Evaluating country risk: A decision support approach. Global Finance J 3(1):79–95
3. Devaud JM, Groussaud G, Jacquet-Lagrèze E (1980) UTADIS: Une méthode de construction de fonctions d'utilité additives rendant compte de jugements globaux. Europ. Working Group on Multicriteria Decision Aid, Paris
4. Doumpos M, Zopounidis C (1997) The use of the preference disaggregation analysis in the assessment of financial risks. In: Zopounidis C, Garcia Vázquez JM (eds) Managing in Uncertainty. Proc. VI Internat. Conf. AEDEM, AEDEM Ed., pp 87–98
5. Hatzinakos I, Yannacopoulos D, Faltsetas C, Ziourkas C (1991) Application of the MINORA decision support system to the evaluation of landslide favourability in Greece. Europ J Oper Res 50:60–75
6. Horsky D, Rao MR (1984) Estimation of attribute weights from preference comparisons. Managem Sci 30(7):801–822
7. Hurson Ch, Zopounidis C (1997) Gestion de portefeuille et analyse multicritère. Economica, Paris
8. Jacquet-Lagrèze E (1990) Interactive assessment of preferences using holistic judgments: The PREFCALC system. In: Bana e Costa CA (ed) Readings in Multiple Criteria Decision Making. Springer, Berlin, pp 335–350
9. Jacquet-Lagrèze E (1995) An application of the UTA discriminant model for the evaluation of R&D projects. In: Pardalos PM, Siskos Y, Zopounidis C (eds) Advances in Multicriteria Analysis. Kluwer, Dordrecht, pp 203–211
10. Jacquet-Lagrèze E, Siskos Y (1982) Assessing a set of additive utility functions for multicriteria decision making, the UTA method. Europ J Oper Res 10:151–164
11. Keeney RL, Raiffa H (1976) Decisions with multiple objectives: Preferences and value tradeoffs. Wiley, New York
12. Lam KF, Choo E-U (1995) Goal programming in preference decomposition. J Oper Res Soc 46:205–213
13. Matsatsinis NF, Siskos Y (1999) MARKEX: An intelligent decision support system for product development decisions. Europ J Oper Res 113:336–354
14. Oral M, Kettani O (1989) Modelling the process of multiattribute choice. J Oper Res Soc 40(3):281–291
15. Pardalos PM, Siskos Y, Zopounidis C (1995) Advances in multicriteria analysis. Kluwer, Dordrecht
16. Pekelman D, Sen SK (1974) Mathematical programming models for the determination of attribute weights. Managem Sci 20(8):1217–1229
17. Richard JL (1983) Aide à la décision stratégique en P.M.E. In: Jacquet-Lagrèze E, Siskos Y (eds) Méthode de Décision Multicritère. Ed. Hommes et Techniques, pp 119–142
18. Roy B (1968) Classement et choix en presence de points de vue multiples: La methode ELECTRE. RIRO 8:57–75
19. Roy B (1985) Méthodologie multicritère d'aide à la décision. Economica, Paris
20. Roy B (1991) The outranking approach and the foundations of ELECTRE methods. Theory and Decision 31:49–73
21. Schärlig A (1985) Decider sur plusieurs criteres, collection diriger l'entreprise. Press. Polytechniques Univ. Romandes
22. Siskos J (1982) A way to deal with fuzzy preferences in multi-criteria decision problems. Europ J Oper Res 10:314–324
23. Siskos J (1986) Evaluating a system of furniture retail outlets using an interactive ordinal regression method. Europ J Oper Res 23:179–193
24. Siskos J, Assimakopoulos N (1989) Multicriteria highway planning: A case study. Math Comput Modelling 12(10)-11:1401–1410
25. Siskos Y, Grigoroudis E, Zopounidis C, Saurais O (1998) Measuring customer satisfaction using a collective preference disaggregation model. J Global Optim 12(2):175–195
26. Siskos Y, Spiridakos A, Yannacopoulos D (1993) MINORA: A multicriteria decision aiding system for discrete alternatives. J Inform Sci and Techn 2:136–149
27. Siskos Y, Spiridakos A, Yannacopoulos D (1999) Using artificial intelligence and visual techniques into preference disaggregation analysis: The MIIDAS system. Europ J Oper Res 113:281–299
28. Siskos Y, Yannacopoulos D (1985) UTASTAR: An ordinal regression method for building additive value functions. Invest Operacion 5(1):39–53
29. Siskos Y, Zopounidis C (1987) The evaluation criteria of the venture capital investment activity: An interactive assessment. Europ J Oper Res 31:304–313
30. Spronk J (1981) Interactive multiple goal programming application to financial planning. Martinus Nijhoff, Boston
31. Srinivasan V, Shocker AD (1973) Linear programming techniques for multidimensional analysis of preferences. Psychometrika 38(3):337–396
32. Steuer RE (1986) Multiple criteria optimization: Theory, computation and application. Wiley, New York
33. Stewart TJ (1987) Pruning of decision alternatives in multiple criteria decision making, based on the UTA method for estimating utilities. Europ J Oper Res 28:79–88
34. Vincke Ph (1992) Multicriteria decision-aid. Wiley, New York
35. Zeleny M (1982) Multiple criteria decision making. McGraw-Hill, New York

36. Zionts S (1992) The state of multiple criteria decision making: Past, present, and future. In: Goicoechea A Duckstein L, Zionts S (eds) Multiple Criteria Decision Making. Springer, Berlin, pp 33–43
37. Zopounidis C (1987) A multicriteria decision-making methodology for the evaluation of the risk of failure and an application. Found Control Eng 12(1):45–67
38. Zopounidis C (1990) La gestion du capital-risque. Economica, Paris
39. Zopounidis C (1995) Evaluation du risque de défaillance de l'entreprise: Méthodes et cas d'application. Economica, Paris
40. Zopounidis C (1999) Multicriteria decision aid in financial management. Europ J Oper Res 119:404–415
41. Zopounidis C, Doumpos M (1997) A multicriteria decision aid methodology for the assessment of country risk. In: Zopounidis C, Garcia Vázquez JM (eds) Managing in Uncertainty. Proc. VI Internat. Conf. AEDEM. AEDEM Ed., pp 223–236
42. Zopounidis C, Doumpos M (1997) Preference disaggregation methodology in segmentation problems: The case of financial distress. In: Zopounidis C (ed) New Operational Approaches for Financial Modelling. Physica Verlag, Heidelberg, pp 417–439
43. Zopounidis C, Doumpos M (1998) Developing a multicriteria decision support system for financial classification problems: The FINCLAS system. Optim Methods Softw 8:277–304
44. Zopounidis C, Doumpos M (2000) PREFDIS: A multicriteria decision support system for sorting decision problems. Comput Oper Res 27:779–797
45. Zopounidis C, Matsatsinis NF, Doumpos M (1996) Developing a multicriteria knowledge-based decision support system for the assessment of corporate performance and viability: The FINEVA system. Fuzzy Economic Rev 1/2:35–53

Preference Modeling

PH. VINCKE
Institute Statist. et Recherche Opération. Service de Math., Gestion University Libre de Bruxelles, Brussels, Belgium

MSC2000: 90C29

Article Outline

Keywords
Basic Preference Relations
Traditional Preference Model
Extensions of the Traditional Model
Valued Preferences
Preferences on Structured Sets
Other Topics Connected to Preference Modeling
See also
References

Keywords

Preference; Ordered set; Valued relation; Utility; Measurement

Preference modeling is an inevitable step in a lot of fields, that include economy, psychology, sociology, operational research, decision support systems.

This step is sometimes implicit, as in operational research and economy, where the preferences of the decision-maker are often represented by a function to be optimized; sometimes it is studied in detail and based on experiments or inquiries, as in psychophysics or multicriteria analysis. Of course, people coming from different disciplines will, in general, have different points of view on preference modeling. It is useful to distinguish the *normative perspective* (attitude which consists in defining how to take a decision in order to satisfy predefined norms or properties; here the main questions are: how to be rational? how must be the preferences in order to reach that goal?), the *descriptive perspective* (attitude which consists in modeling, as realistically as possible, the behavior of a decision-maker; how do people take decisions?) and the *prescriptive* one (attitude which consists in helping a decision-maker to be as coherent as possible with his own goals and preferences; how to help somebody to decide?).

However, people working on preferences, even they come from different fields and have different perspectives, use more or less the same tools. The purpose of this article is to give a brief overview of these tools.

Basic Preference Relations

Let A be the set of elements (decisions, candidates for a job, commodities, consumption levels, locations, …) to be compared or evaluated. It is often assumed that, comparing two elements a and b, the decision-maker can have three attitudes: *preference* for one element (a decision-maker prefers a decision a over a decision b (eventually according to a particular point of view) if, in a situation where he has to select one of them, he

chooses a), *indifference* between a and b (a decision-maker is indifferent between two decision a and b (eventually according to a particular point of view) if both decisions are equally acceptable for him), and *incomparability* between them (two decisions a and b are incomparable (at a given moment of a decision process) if the decision-maker is not able or refuse to express a preference or an indifference, due to a lack of data or contradictory information). We will denote:

$$\begin{cases} aPb & \text{if } a \text{ is preferred to } b, \\ bPa & \text{if } b \text{ is preferred to } a, \\ aIb & \text{in case of indifference}, \\ aJb & \text{in case of incomparability}. \end{cases}$$

The basic preference relations P, I and J are the sets of couples (a, b) such that, respectively, aPb, aIb and aJb. These relations are used in most of the works devoted to preference modeling. Moreover, it is traditionally accepted that these relations are mutually exclusive and verify the following basic properties: $\forall a, b \in A$,

$$\begin{cases} aPb \Rightarrow b \not P a & (P \text{ is asymmetric}), \\ aIa & (I \text{ is reflexive}), \\ aIb \Rightarrow bIa & (I \text{ is symmetric}), \\ a \not J a & (J \text{ is irreflexive}), \\ aJb \Rightarrow bJa & (J \text{ is symmetric}). \end{cases}$$

Defining the 'preference-indifference relation' S by aSb if and only if aPb or aIb, it is not difficult to see that all the above situations can be characterized using only relation S:

$$\begin{cases} aPb & \text{if and only if} \quad aSb \text{ and } b \not S a, \\ aIb & \text{if and only if} \quad aSb \text{ and } bSa, \\ aJb & \text{if and only if} \quad a \not S b \text{ and } b \not S a. \end{cases}$$

Traditional Preference Model

A usual attitude is to associate numbers (valuations) to the elements of A and to declare that a is preferred to b if the 'value' of a is greater (or smaller if the 'value' is a cost for example) than the value of b and that there is indifference if the values are equal. This is the traditional approach in operational research, decision theory, economy or finance, where decision problems are considered as optimization ones. This attitude leads to a preference model with the following strong assumptions: $\forall a, b, c \in A$,

$$\begin{cases} aPb \text{ and } bPc \Rightarrow aPc & (P \text{ transitive}), \\ aIb \text{ and } bIc \Rightarrow aIc & (I \text{ transitive}), \\ aPb \text{ and } bIc \Rightarrow aPc, \\ aIb \text{ and } bPc \Rightarrow aPc, \end{cases}$$

and J is empty (there is no incomparability).

In terms of relation S, this means that, $\forall a, b, c \in A$,

$$\begin{cases} aSb \text{ and } bSc \Rightarrow aSc & (S \text{ transitive}), \\ a \not S b \Rightarrow bSa & (S \text{ complete}), \end{cases}$$

characterizing what is usually called a *weak order* (i. e., a transitive and complete relation). In this case, indifference I is an *equivalence* (i. e., a reflexive, symmetric and transitive relation) and the set of equivalence classes is totally ordered by relation P.

When A is finite or enumerable, the weak order structure is sufficient to represent preferences by numbers as explained here above; in the nonenumerable case, some topological conditions have to be added (see [7]).

Extensions of the Traditional Model

The transitivity of indifference is incompatible with the existence of a 'sensitivity threshold' under which the decision-maker does not feel any difference between two elements or refuses to accept a strict preference for one of the elements. Moreover, the frontier between indifference and preference is not always clear, leading to new situations (and new relations) called 'hesitation between indifference and preference' or 'weak, strong, very strong, ... preferences'. Finally it is also necessary to extend the traditional model by taking into account incomparability situations (J is not always empty). All these considerations have led to new models called *semi-orders*, *interval orders*, *partial orders*, *pseudo-orders*, *suborders*, *embedded families of preferences*, ..., which were extensively studied in the literature. The interested reader can consult [5,6,8,11,14,20,22,26].

Valued Preferences

Modelers sometimes want to quantify preferences to express either an intensity of preference, either a num-

ber (or a percentage) of votes in favor of an element over another, either a probability of preference, or a 'degree' (without any specific property). In this case, a number is associated to each couple (a, b), leading to so-called valued, fuzzy, probabilistic relations. The study of the properties of these relations, of their representations by numerical functions and of their use in decision models, is illustrated, for instance, by [9] or [12].

Preferences on Structured Sets

The set A may have a special structure: the elements of A are sometimes points in a topological space (as in economy), vectors of evaluations on several dimensions (as in multicriteria analysis), probability distributions on a set of consequences (as in decision-making under risk), sets of possible consequences depending on the states of the nature (as in decision-making under uncertainty). All these situations have led to a lot of models, concepts or methodologies that include value theory, utility theory, multi-attribute utility theory, multiple criteria decision analysis, subjective expected utility theory and conjoint measurement theory.

Some references in this abundant literature are [7,10,15,16,18,19,23,25,28,29].

Other Topics Connected to Preference Modeling

We briefly mention in this section some topics which cannot be ignored by people interested in preference modeling:

- the 'art' of collecting preference information from subjects (see for example [30]),
- the statistical analysis of preferences (see [4] or [13]),
- the geometrical representation of preferences (see for example [2]),
- the problem of meaningfulness ([21]),
- the evolution of preferences over time (see [17]),
- the philosophical aspects of preference modeling ([3,31])
- the social choice theory ([24]),
- the preference aggregation problem ([1,27].

See also

- ► Bi-objective Assignment Problem
- ► Decision Support Systems with Multiple Criteria
- ► Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ► Financial Applications of Multicriteria Analysis
- ► Fuzzy Multi-objective Linear Programming
- ► Multicriteria Sorting Methods
- ► Multi-objective Combinatorial Optimization
- ► Multi-objective Integer Linear Programming
- ► Multi-objective Optimization and Decision Support Systems
- ► Multi-objective Optimization: Interaction of Design and Control
- ► Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ► Multi-objective Optimization: Lagrange Duality
- ► Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ► Multiple Objective Programming Support
- ► Outranking Methods
- ► Portfolio Selection and Multicriteria Analysis
- ► Preference Disaggregation
- ► Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making

References

1. Arrow K (1963) Social choice and individual values, 2nd edn. Wiley, New York
2. Batteau P, Jacquet-Lagrèze E, Monjardet B (1981) Analyse et agrégation des préeférences dans les sciences sociales, économiques et de gestion. Economica, Paris
3. Broome J (1991) Weighting goods. Blackwell, Oxford
4. Coombs CH (1964) A theory of data. Wiley, New York
5. Doignon JP (1987) Threshold representation of multiple semiorders. SIAM J Alg Discrete Meth 8:77–84
6. Doignon JP, Monjardet B, Roubens M, VinckePh (1986) Biorder families, valued relations and preference modelling. J Math Psych 30(4):435–480
7. Fishburn PC (1970) Utility theory for decision-making. Wiley, New York
8. Fishburn PC (1970) Utility theory with inexact preferences and degrees of preference. Synthese 21:204–222
9. Fishburn PC (1973) Binary choice probabilities: on the varieties of stochastic transitivity. J Math Psych 10:327–352
10. Fishburn PC (1982) The foundations of expected utility. Reidel, London
11. Fishburn PC (1985) Interval orders and interval graphs. Wiley, New York
12. Fodor J, Roubens M (1994) Fuzzy preference modelling and multicriteria decision support. Kluwer, Dordrecht

13. Green PE, Tull DS, Albaum G (1988) Research for marketing decisions. Prentice-Hall, Englewood Cliffs
14. Kacprzyk J, Roubens M (1988) Non-conventional preference relations in decision making. Lecutre Notes Economics and Math Systems, vol 301. Springer Berlin
15. Keeney R, Raiffa H (1976) Decisions with multiple objectives; preferences and value trade-offs. Wiley, New York
16. Krantz DM, Luce RD, Suppes P, Tversky A (1971) Foundations of measurement I. Acad. Press, New York
17. Kreps DM (1979) A representation theorem for 'preference for flexibility'. Econometrica 47:565–577
18. Luce RD, Krantz DM, Suppes P, Tversky A (1990) Foundations of measurement III. Acad. Press, New York
19. Neumann JVon, Morgenstern O (1967) Theory of games and economic behavior. Princeton Univ. Press, Princeton
20. Pirlot M, Vincke Ph (1997) Semiorders: properties, representations, applications. Kluwer, Dordrecht
21. Roberts FS (1979) Measurement theory, with applications to decision making, utility and the social sciences. Encycl Math Appl, vol 7. Addison-Wesley, Reading
22. Roubens M, Vincke Ph (1985) Preference modelling. Lecutre Notes Economics and Math Systems, vol 250. Springer, Berlin
23. Roy B (1985) Méthodologie multicritère d'aide á la décision. Economica, Paris. English translation: Multicriteria Methodology for Decision Aiding, Kluwer, 1996
24. Sen AK (1970) Collective choice and social welfare. Holden Day, San Francisco
25. Suppes P, Krantz DM, Luce RD, Tversky A (1989) Foundations of measurement II. Acad. Press, New York
26. Tsoukiás A, VinckePh (1992) A survey on nonconventional preference modelling. Ricerca Oper (1992):5–49
27. Vincke Ph (1982) Aggregation of preferences: A review. Europ J Oper Res 9:17–22
28. VinckePh (1992) Multicriteria decision-aid. Wiley, New York
29. Wakker PP (1989) Additive representations of preferences: A new foundation of decision analysis. Kluwer, Dordrecht
30. Winterfeld Dvon, Edwards W (1986) Decision analysis and behavior research. Cambridge Univ. Press, Cambridge
31. von Wright GH (1963) The logic of preference. Edinburgh Univ. Press, Edinburgh

Preprocessing in Stochastic Programming

Stein W. Wallace
Department Indust. Econ. and Techn. Management, øshaugen Norwegian University Sci. and Techn., Trondheim, Norway

MSC2000: 90C15, 90C35, 90B10, 90B15

Article Outline

Keywords
See also
References

Keywords

Stochastic programming; Preprocessing; Feasibility; Gale–Hoffman inequalities

A two-stage stochastic linear program with random right-hand side (and with discrete random variables) is normally written as follows

$$\begin{cases} \min\limits_{x\geq 0} & cx + \mathcal{Q}(x) \\ \text{s.t.} & Ax = b \end{cases}$$

where

$$\mathcal{Q}(x) = \sum_j p_j Q(x, \xi_j)\,,$$

and

$$Q(x,\xi) = \min_{y\geq 0}\{qy\colon\ Wy = h(\xi) - T(\xi)x\}\ .$$

Here, p_j is the probability that the random vector $\widetilde{\xi}$ takes on its jth possible value ξ_j, and W is an $(m \times \mathrm{n})$ matrix. This way of formulating a two-stage stochastic program is partly motivated by solution procedures ([1,8]), partly by the time structure of the problem. For this article, the latter is the most important. The interpretation of the problem is that first (now) we make a decision x, then we observe a value of $\widetilde{\xi}$, and finally we make a *recourse decision* y based on our earlier decision x and the observed value of $\widetilde{\xi}$.

Many aspects of preprocessing (pre-analysis) are in no way particular to multistage stochastic problems. For example, the issue of removing redundant rows and columns, as well as that of consistency, are well discussed for deterministic (one-stage) problems, see for example the work on preprocessing in linear programming by H.J. Greenberg [5]. Details referring to stochastic programs are found in [10], and in [7, Chapt. 5; 6]. The one major point to make for stochastic programs is that since, for example, the matrix W shows up

in the formulation as many times as there are possible values of $\widetilde{\xi}$, it may be worthwhile to spend a lot of effort just to simplify the matrix a little. This may not be true for deterministic LPs, as it may cost more to simplify the matrix than to solve the problem to optimality without simplifications. Hence, although *technically* the issues are the same, the *motivation* for simplifications may be rather different.

An $x_0 \geq 0$ satisfying $Ax_0 = b$ may produce a *recourse problem* $Q(x_0, \xi_j)$ which is infeasible for some j. In other words, the requirement that $Q(x, \xi)$ be feasible for all ξ produces *implied constraints* on the *first-stage decisions* x. The cone pos W is a cone containing all vectors which are such that if they are put as right hand sides in the second-stage problem, they produce a feasible problem. Formally,

$$\text{pos } W = \{t\colon\ t = Wy,\ y \geq 0\}\,.$$

If pos W = $\mathbf{R}^m$ we say that we have *complete recourse*. That is easy to test for [10]. If $h(\xi) - \mathrm{T}(\xi)\ \mathrm{x} \in$ pos W for all possible ξ and all $x \geq 0$ satisfying $Ax = b$, we have *relatively complete recourse*. This is hard to test, but possible to generate. The rest of this article will discuss the generation of relatively complete recourse. The clue is that if we find all implied constrains, and add them to Ax = b, the expanded set of constraints will imply relatively complete recourse by construction. These implied constraints are useful in two directions, a) when solving the stochastic program we can use methods requiring relatively complete recourse, and b) the implied constraints will show the modeler things he has assumed, but never written down explicitly. Hence, the constraints are useful for both error detection and increased understanding. A small numerical example can be found in [7, Sect. 6.4].

To find an implied constraint for a given x_0 is equivalent to finding a *feasibility cut* in Benders decomposition (cf. also ► Generalized Benders decomposition). However, the purpose of preprocessing is not to find just one, but to find all implied constraints (feasibility cuts). More formally, we look for a *polar matrix* W^* (with a minimal number of columns), which is such that

$$y \in \text{pos } W \ \Leftrightarrow\ y^{\mathrm{T}} W^* \leq 0\,.$$

```
PROCEDURE support(W, W*);
BEGIN
    frame(W);
    done:=false;
    FOR i = 1 TO n DO
        IF NOT done THEN BEGIN
            α := W_i^T W*;
            I+ := {k : α(k) > 0};
            I− := {k : α(k) < 0};
            I= := {k : α(k) = 0};
            done:= (I− ∪ I= = ∅);
            IF done THEN W* := 0;
            IF I+ ≠ ∅ AND NOT done THEN
            BEGIN
                IF I− = ∅ THEN W* := W*_{I=}
                ELSE BEGIN
                    FOR ALL k ∈ I+, j ∈ I− DO
                    C_kj := W*_k − (α(k)/α(l)) W*_j;
                    W* := W*_{I=} ∪ W*_{I−} ∪_kj C_kj;
                    frame(W*);
                END; (*ELSE*)
            END; (*IF*)
    END; (*FOR*)
END support;
```

Pseudocode for finding the polar matrix *W

Finding W^* is a problem of exponential complexity, equivalent to extreme point enumeration. An algorithm tailored to the problem is found in [10], and presented above. For reasonably large LPs one cannot, in general, expect to be able to find all implied constraints. However, if there are few implied constraints (and that is in any case the more interesting situation) the algorithm will find them quickly. Hence, this procedure may in some situations yield a lot of insight into a problem, in other cases it leads nowhere. Only by checking can that be found out. A pseudocode for the algorithm is given below. Two calls are made to a procedure frame(W). This procedure finds a *frame* of pos W, i. e., it removes all columns from W that are nonnegative linear combinations of other columns. The variable done} is boolean. The recourse matrix W is input to the procedure, and the polar matrix W^* is output. An illustrated example of the use of this procedure is found in [7, Sect. 5.2].

For networks there are some more specific results. The starting point is the well known result by D. Gale [2] and A.J. Hoffman [6]. Let a network be given by a set of arcs $\mathcal{A} = \{1, \ldots, n\}$ and a set of nodes $\mathcal{N} = \{1, \ldots, m\}$. An arc k may be described by its start node i and end node j by $k \sim (i, j)$. We let $\beta(i)$ be the external flow in node i (with a positive number meaning supply), and let $\gamma(k)$ be the capacity of arc k. By $Q^+ = [Y, \mathcal{N}\setminus Y]^+$ we understand the set of arcs starting in a node in Y and ending in $\mathcal{N}\setminus Y$. We define Q^- similarly, and define $Q = Q^+ \cup Q^-$ as a *cut*. Finally, we let $G(Y)$ be the graph consisting of the nodes in Y, and the arcs connecting nodes in Y. The Gale–Hoffman (GH) result says that a capacitated network flow problem is feasible if and only if for every cut $Q = [Y, \mathcal{N}\setminus Y]$, the net supply in Y is less than or equal to the capacity of Q^+, i. e. if the GH-inequality

$$\sum_{i \in Y} \beta(i) \leq \sum_{k \in Q^+} \gamma(k) \tag{1}$$

is satisfied. Following [11] this has been strengthened to show that a GH-inequality is needed if and only if $G(Y)$ and $G(\mathcal{N}\setminus Y)$ are both connected. If either of them is disconnected, the GH-inequality is not needed as it is implied by other GH-inequalities. This is an interesting result as it shows that a linear dependency argument can be obtained by looking only at one inequality at a time.

An algorithm for finding all necessary GH-inequalities can be found in [12], and one for updating the inequalities when new arcs or nodes are introduced is given in [3]. The latter can be seen as a network interpretation of procedure support. These results are shown for uncapacitated networks in [4,9]. They can be found directly or deduced from the capacitated case by observing that (1) will always be satisfied as long as $Q^+ \neq \emptyset$. Hence, we keep only those GH-inequalities for which $Q^+ = \emptyset$. The result that $G(Y)$ and $G(\mathcal{N}\setminus Y)$ both must be connected still applies.

If the recourse problem is a network flow problem, the W-matrix is the node-arc incidence matrix (with a row removed), and both external flows and arc capacities are functions of the first stage decisions x and the random variables $\widetilde{\xi}$. If β and γ in (1) are replaced by the appropriate expressions in x and ξ, and the results added to $Ax = b$, we obtain relatively complete recourse, which was our goal. Since this must be true for all ξ_j for a given x, we can normally perform a worst-case analysis with respect to ξ_j, and add this result to the first case constraints.

If a capacitated network flow problem is formulated as an LP, with the upper bounds written explicitly as constraints, and procedure support is applied, the columns of W^* will correspond to an index vector for β and γ (with negative signs on the index vector for γ) in (1).

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution

- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Math 4:238–252
2. Gale D (1957) A theorem of flows in networks. Pacific J Math 7:1073–1082
3. Ghannadan S, Wallace SW (1996) Feasibility in capacitated networks: The effect of individual arcs and nodes. Ann Of MATH Artificial Intelligence 17:145–153
4. Ghannadan S, Wallace SW (1996) Feasibility in uncapacitated networks: The effect of individual arcs and nodes. Ann Oper Res 64:197–210
5. Greenberg HJ (1996) Consistency, redundancy and implied equalities in linear systems. Ann of MATH Artificial Intelligence} 17:37–83
6. Hoffman AJ (1960) Some recent applications of the theory of a multivariate random variable. In: Proc. Symp. Appl. Math., vol 10, pp 113–128
7. Kall P, Wallace SW (1994) Stochastic programming. Wiley, New York
8. Slyke Rvan, Wets RJ-B (1969) L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J Appl Math 17:638–663
9. Wallace SW, Wets RJ-B (1989) Preprocessing in stochastic programming: The case of uncapacitated networks. ORSA J Comput 1:252–270
10. Wallace SW, Wets RJ-B (1992) Preprocessing in stochastic programming: The case of linear programs. ORSA J Comput 4:45–59
11. Wallace SW, Wets RJ-B (1993) The facets of the polyhedral set determined by the Gale-Hoffman inequalities. Math Program 62(1):215–222
12. Wallace SW, Wets RJ-B (1995) Preprocessing in stochastic programming: The case of capacitated networks. ORSA J Comput 7(1):44–62

Price of Robustness for Linear Optimization Problems

Melvyn Sim
1 Business Link, NUS Business School Office, Singapore, Singapore

Article Outline

Introduction
Formulation
- Affine Data Dependency
- Worst Case Uncertainty Set
- Uncertainty Set with a Budget

Applications
Conclusions
References

Introduction

The classical paradigm in mathematical programming is to develop a model that assumes that the input data is precisely known and equal to some nominal values. This approach, however, does not take into account the influence of data uncertainties on the quality and feasibility of the model. It is therefore conceivable that as the data takes values different than the nominal ones several constraints may be violated, and the optimal solution found using the nominal data may be no longer optimal or even feasible. In a numerical case study on linear optimization problems from the Net Lib library, Ben-Tal and Nemirovski [1] concluded that in in real-world applications of linear optimization problems, one cannot ignore the possibility that a small uncertainty in the data can make the usual optimal solution completely meaningless from a practical viewpoint. This observation raises the natural question of designing solution approaches that are immune to data uncertainty, that is they are "robust". Modern robust linear optimization models are proposed by Ben-Tal and Nemirovski [1], Bertsimas and Sim [2] and Chen, Sim and Sun [3]. While the proposals of Ben-Tal and Nemirovski [1] and Chen, Sim and Sun [3] lead to second order cone optimization problems (SOCP), the proposal of Bertsimas and Sim [2] preserves the linearity of the model. We focus on the results of Bertsimas and Sim [2].

Formulation

We consider the following nominal linear optimization problem,

$$\begin{array}{ll} \text{maximize} & \boldsymbol{c}'\boldsymbol{x} \\ \text{subject to} & \tilde{\boldsymbol{a}}_i'\boldsymbol{x} \le \tilde{b}_i \quad i = 1, \ldots, m \end{array} \tag{1}$$

in which the data at every constraint, $(\tilde{\boldsymbol{a}}_i, \tilde{b}_i)$, $i = 1, \ldots, m$ are potentially uncertain at the time when the decision $\boldsymbol{x} \in \Re^n$ needs to be made. For simplicity of our exposition, we assume without loss of generality that the data at the objective $\boldsymbol{c}$ is not subject to uncertainty, since we can use the objective maximize t and add the constraint $t - \tilde{\boldsymbol{c}}'\boldsymbol{x} \le 0$ to the model.

Affine Data Dependency

We focus on constraint-wise uncertainties. Clearly, uncertainties appearing at various parts of the data in any ith constraint $(\tilde{a}_i, \tilde{b}_i)$ may be correlated. To capture such correlation, we assume that the uncertain data at the ith constraint is affinely dependent on some primitive uncertainty vector, $\tilde{z}^i \in \Re^{N_i}$ as follows

$$\begin{aligned} \tilde{\boldsymbol{a}}_i &= \boldsymbol{a}_i(\tilde{z}^i) = \boldsymbol{a}_i^0 + \sum_{j=1}^{N_i} \boldsymbol{a}_i^j \tilde{z}_j^i \,, \\ \tilde{b}_i &= b_i(\tilde{\boldsymbol{z}}^i) = b_i^0 + \sum_{j=1}^{N_i} b_i^j \tilde{z}_j^i \,, \end{aligned}$$

where $(\boldsymbol{a}_i^j, b_i^j) \in \Re^{n+1}$, $j = 1, \ldots, N_i$ are non zeros vectors. Note that we can always define a bijection mapping from a vector space of $\tilde{z}^i \in \Re^{N_i}$ at the ith constraint to the corresponding data space of $(\tilde{a}_i, \tilde{b}_i)$. Therefore, under the affine data dependency, it is always possible to map all the data uncertainties affecting the ith constraint to the primitive uncertainty vector, $\tilde{z}^i$. The affine data mapping can easily represent linear relations among data entries. As an illustration,

$$\begin{pmatrix} a_1 \\ a_2 \\ b \end{pmatrix} (z_1, z_2) = \begin{pmatrix} 100 \\ 200 \\ 300 \end{pmatrix} + \begin{pmatrix} 2 \\ -5 \\ 0 \end{pmatrix} z_1 + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} z_2,$$

the data is a vector in $\Re^3$ and has two primitive uncertainties. The first and second elements are related such that when a_1 increases by two units, a_2 decreases by five units. The values of b can change independently of a_1 or a_2 and it is controlled by another primitive uncertainty, z_2. For the case when all data entries in $(\tilde{a}_i, \tilde{b}_i)$ are independently distributed, we would have $N_i = n + 1$ and that the vector $(\boldsymbol{a}_i^j, b_i^j)$, $j = 1, \ldots, n + 1$ is a vector taking a non-zero value only at the jth row. We assume that the distributions of the primitive uncertainty vector, $\tilde{z}^i$, $i = 1, \ldots, m$ are mildly characterized as follows:

Model of Data Uncertainty U We assume that $\tilde{z}_j^i$, $j = 1, \ldots, N_i$ are independently (but not necessarily identically) distributed random variables with zero means and support in $[-1, 1]$.

Under the model of data uncertainty U, the nominal value of $\tilde{z}^i$ is a zero vector. Hence, it follows naturally that $(\boldsymbol{a}_i^0, b_i^0)$ is the nominal value of the uncertain data $(\boldsymbol{a}_i(\tilde{z}^i), b_i(\tilde{z}^i))$. Similarly, $(\boldsymbol{a}_i^j, b_i^j)$, $j = 1, \ldots, N_i$ is the direction of data perturbation under the influence of the primitive uncertainty $\tilde{z}_j^i$.

In robust optimization, we represent data uncertainty using uncertainty sets instead of probability distributions. At each constraint, we allow the primitive uncertainty vector, $\tilde{z}^i$ to vary within an uncertainty set, $\mathcal{U}_i$ without having to violate the ith constraint. We call the following problem the *robust counterpart* of Problem (1)

$$\begin{array}{ll} \text{maximize} & \boldsymbol{c}'\boldsymbol{x} \\ \text{subject to} & \boldsymbol{a}_i(\boldsymbol{z}^i)'\boldsymbol{x} \le b_i(\boldsymbol{z}^i) \\ & \forall \boldsymbol{z}^i \in \mathcal{U}_i \, i = 1, \ldots, m \,. \end{array} \tag{2}$$

Indeed, given any solution to the robust counterpart, we are able to guarantee deterministically that the solution will remain feasible if the primitive uncertainty vector, $\tilde{z}$ lies within the uncertainty set $\mathcal{U}_i$. Under the model of data uncertainty U, we are able to design uncertainty sets that guarantee feasibility in constraints with high probability without being overly conservative.

Although the robust counterpart (2) has possibly exponential or even infinite number of constraints, it is nevertheless a convex optimization problem with respect to its decision variable, $\boldsymbol{x}$. In fact, the computational complexity of the robust counterpart depends on the nature of the uncertainty set, $\mathcal{U}_i$. Here, we focus on polyhedral uncertainty sets in which the robust counterparts are computationally tractable in the form of

linear optimization problems that are moderately larger in size as compared to the nominal problems.

Using linear programming duality, we show how to convert the robust counterpart (2) into a concise linear optimization problem. We focus on the following polyhedral uncertainty set,

$$\mathcal{P}_i \doteq \{\boldsymbol{z} \in \Re^{N_i} : \boldsymbol{S}_i\boldsymbol{z} + \boldsymbol{T}_i\boldsymbol{u} \leq \boldsymbol{r}_i \text{ for some } \quad \boldsymbol{u}\}.$$

Theorem 1 *The robust counterpart*

$$\boldsymbol{a}_i(\boldsymbol{z}^i)'\boldsymbol{x} \leq b_i(\boldsymbol{z}^i) \qquad \forall \boldsymbol{z}^i \in \mathcal{P}_i, \tag{3}$$

is equivalent to

$$\begin{cases} \exists \boldsymbol{p}: \\ \boldsymbol{r}_i'\boldsymbol{p} \leq b_i^0 - \boldsymbol{a}_i^{0\prime}\boldsymbol{x} \\ \boldsymbol{S}_i'\boldsymbol{p} = \begin{pmatrix} \boldsymbol{a}_i^{1\prime}\boldsymbol{x} - b_i^1 \\ \vdots \\ \boldsymbol{a}_i^{N_i\prime}\boldsymbol{x} - b_i^{N_i} \end{pmatrix} \\ \boldsymbol{T}_i'\boldsymbol{p} = \boldsymbol{0} \\ \boldsymbol{p} \geq \boldsymbol{0}. \end{cases} \tag{4}$$

Proof 1 For notational convenience, we ignore the constraint index i. Under the affine data dependency, we can represent the constraint (3) as

$$\sum_{j=1}^{N}(\boldsymbol{a}^{j\prime}\boldsymbol{x} - b^j)z_j \leq b^0 - \boldsymbol{a}^{0\prime}\boldsymbol{x} \qquad \forall \boldsymbol{z} \in \mathcal{P}$$

or equivalently as

$$\max_{\boldsymbol{z}\in\mathcal{P}} \left\{ \sum_{j=1}^{N}(\boldsymbol{a}^{j\prime}\boldsymbol{x} - b^j)z_j \right\} \leq b^0 - \boldsymbol{a}^{0\prime}\boldsymbol{x}. \tag{5}$$

By standard linear programming duality, the objective of the following problem

$$\begin{array}{ll} \text{maximize} & \boldsymbol{d}'\boldsymbol{z} \\ \text{subject to} & \boldsymbol{S}\boldsymbol{z} + \boldsymbol{T}\boldsymbol{u} \leq \boldsymbol{r} \end{array}$$

is the same as

$$\begin{array}{lll} \text{minimize} & \boldsymbol{r}'\boldsymbol{p} & \\ \text{subject to} & \boldsymbol{S}'\boldsymbol{p} & = \boldsymbol{d} \\ & \boldsymbol{T}'\boldsymbol{p} & = \boldsymbol{0} \\ & \boldsymbol{p} & \geq \boldsymbol{0}. \end{array}$$

Hence, the constraint (5) is equivalent to

$$\begin{array}{lll} \text{minimize} & \boldsymbol{r}'\boldsymbol{p} & \leq b^0 - \boldsymbol{a}^{0\prime}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{S}'\boldsymbol{p} = \begin{pmatrix} \boldsymbol{a}^{1\prime}\boldsymbol{x} - b^1 \\ \vdots \\ \boldsymbol{a}^{N\prime}\boldsymbol{x} - b^N \end{pmatrix} & \\ & \boldsymbol{T}'\boldsymbol{p} = \boldsymbol{0} & \\ & \boldsymbol{p} \geq \boldsymbol{0}. & \end{array} \tag{6}$$

Finally, to ensure the feasibility of $\boldsymbol{x}$, we only need to find a vector $\boldsymbol{p}$ feasible in the left hand side optimization problem of constraint (6) such that $\boldsymbol{r}'\boldsymbol{p} \leq b^0 - \boldsymbol{a}^{0\prime}\boldsymbol{x}$. If such vector, $\boldsymbol{p}$ exists, the corresponding minimizer, $\boldsymbol{p}^*$ should also satisfy $\boldsymbol{r}'\boldsymbol{p}^* \leq b^0 - \boldsymbol{a}^{0\prime}\boldsymbol{x}$. □

Hence, the robust counterpart (2) in which $\mathcal{U}_i = \mathcal{P}_i$, $i = 1, \ldots, m$ is equivalent to

$$\begin{array}{lll} \text{maximize} & \boldsymbol{c}'\boldsymbol{x} & \\ \text{subject to} & \boldsymbol{r}_i{}'\boldsymbol{p}_i \leq b_i^0 - \boldsymbol{a}_i^{0\prime}\boldsymbol{x} & i = 1, \ldots, m \\ & \boldsymbol{S}_i'\boldsymbol{p}_i = \begin{pmatrix} \boldsymbol{a}_i^{1\prime}\boldsymbol{x} - b_i^1 \\ \vdots \\ \boldsymbol{a}_i^{N_i\prime}\boldsymbol{x} - b_i^{N_i} \end{pmatrix} & i = 1, \ldots, m \\ & \boldsymbol{T}_i'\boldsymbol{p}_i = \boldsymbol{0} & i = 1, \ldots, m \\ & \boldsymbol{p}_i \geq \boldsymbol{0} & i = 1, \ldots, m\,. \end{array} \tag{7}$$

Worst Case Uncertainty Set

Under the model of data uncertainty U, the primitive uncertainty vector $\tilde{z}^i$ at the ith constraint has support in $[-1, 1]^{N_i}$. Hence, we define the following worst case uncertainty set

$$\mathcal{W}_i \doteq \{\boldsymbol{z} \in \Re^{N_i} : -\boldsymbol{1} \leq \boldsymbol{z} \leq \boldsymbol{1}\}$$

for the primitive uncertainty corresponding to the i constraint. Such uncertainty set is first considered by Soyster [4].

Theorem 2 *The robust counterpart*

$$\boldsymbol{a}_i(\boldsymbol{z}^i)'\boldsymbol{x} \leq b_i(\boldsymbol{z}^i) \qquad \forall \boldsymbol{z}^i \in \mathcal{W}_i\,,$$

is equivalent to

$$\begin{cases} \exists \boldsymbol{p}, \boldsymbol{q}: \\ \mathbf{1}'(\boldsymbol{p}+\boldsymbol{q}) \le b_i^0 - \boldsymbol{a}_i^{0\prime}\boldsymbol{x} \\ \boldsymbol{p}-\boldsymbol{q} = \begin{pmatrix} \boldsymbol{a}_i^{1\prime}\boldsymbol{x} - b_i^1 \\ \vdots \\ \boldsymbol{a}_i^{N_i\prime}\boldsymbol{x} - b_i^{N_i} \end{pmatrix} \\ \boldsymbol{p}, \boldsymbol{q} \ge \mathbf{0}\,. \end{cases} \tag{8}$$

The robust counterpart (2) in which $\mathcal{U}_i = \mathcal{W}_i$, $i = 1, \ldots, m$ is equivalent to

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{c}'\boldsymbol{x} \\ \text{subject to} \quad & \mathbf{1}'(\boldsymbol{p}_i + \boldsymbol{q}_i) \le b_i^0 - \boldsymbol{a}_i^{0\prime}\boldsymbol{x} && i = 1, \ldots, m \\ & \boldsymbol{p}_i - \boldsymbol{q}_i = \begin{pmatrix} \boldsymbol{a}_i^{1\prime}\boldsymbol{x} - b_i^1 \\ \vdots \\ \boldsymbol{a}_i^{N_i\prime}\boldsymbol{x} - b_i^{N_i} \end{pmatrix} && i = 1, \ldots, m \\ & \boldsymbol{p}_i, \boldsymbol{q}_i \ge \mathbf{0} && i = 1, \ldots, m\,. \end{aligned} \tag{9}$$

Uncertainty Set with a Budget

Under the model of data uncertainty U, the worst case uncertainty set is can be over conservative. Speaking intuitively, when N_i is large, it may be unlikely for the primitive uncertainty vector $\tilde{\boldsymbol{z}}^i$ to deviate unanimously towards the violation of the constraint. Bertsimas and Sim [2] consider an uncertainty set that is able to withstand parameter uncertainty without excessively affecting the objective function as follows:

$$\mathcal{B}_i(\Gamma_i) = \left\{ \boldsymbol{z} \in \Re^{N_i} : -\mathbf{1} \le \boldsymbol{z} \le \mathbf{1}, \sum_{j=1}^{N_i} |z_j| \le \Gamma_i \right\}.$$

The goal is to be protected against all cases that up to Γ_i of the primitive uncertainties $\tilde{z}_j^i$, $j = 1, \ldots, N_i$ are allowed to to change deviate maximally in $[-1, 1]$. In other words, Bertsimas and Sim stipulate that nature will be restricted in its behavior, in that only a subset of the primitive uncertainties will change in order to adversely affect the solution. They propose an approach, that has the property that if nature behaves like this, then the robust solution will be feasible deterministically.

The parameter Γ_i, commonly known as the *budget of uncertainty* controls the size of the uncertainty set, $\mathcal{B}_i(\Gamma_i)$ such that $\mathcal{B}_i(0) = \{\mathbf{0}\}$ and $\mathcal{B}_i(N_i) = \mathcal{W}_i$. Note that the uncertainty set is a polytope as follows

$$\mathcal{B}_i(\Gamma_i) = \left\{ \boldsymbol{z} \in \Re^{N_i} : \boldsymbol{u} \le \mathbf{1}, \mathbf{1}'\boldsymbol{u} \le \Gamma_i, -\boldsymbol{u} \le \boldsymbol{z} \le \boldsymbol{u}, \text{ for some } \boldsymbol{u} \right\}.$$

Theorem 3 *The robust counterpart*

$$\boldsymbol{a}_i(\boldsymbol{z}^i)'\boldsymbol{x} \le b_i(\boldsymbol{z}^i) \qquad \forall \boldsymbol{z}^i \in \mathcal{B}_i(\Gamma_i),$$

is equivalent to

$$\begin{cases} \exists t, \boldsymbol{s}, \boldsymbol{p}, \boldsymbol{q}: \\ \Gamma_i t + \mathbf{1}'\boldsymbol{s} \le b_i^0 - \boldsymbol{a}_i^{0\prime}\boldsymbol{x} \\ \boldsymbol{s} + \mathbf{1}t = \boldsymbol{p} + \boldsymbol{q} \\ \boldsymbol{p}-\boldsymbol{q} = \begin{pmatrix} \boldsymbol{a}_i^{1\prime}\boldsymbol{x} - b_i^1 \\ \vdots \\ \boldsymbol{a}_i^{N_i\prime}\boldsymbol{x} - b_i^{N_i} \end{pmatrix} \\ \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{s} \ge \mathbf{0}, t \ge 0. \end{cases} \tag{10}$$

The robust counterpart (2) in which $\mathcal{U}_i = \mathcal{B}_i(\Gamma_i)$, $i = 1, \ldots, m$ is equivalent to

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{c}'\boldsymbol{x} \\ \text{subject to} \quad & \Gamma_i t_i + \mathbf{1}'\boldsymbol{s}_i \le b_i^0 - \boldsymbol{a}_i^{0\prime}\boldsymbol{x} && i = 1, \ldots, m \\ & \boldsymbol{s}_i + \mathbf{1}t_i = \boldsymbol{p}_i + \boldsymbol{q}_i && i = 1, \ldots, m \\ & \boldsymbol{p}_i - \boldsymbol{q}_i = \begin{pmatrix} \boldsymbol{a}_i^{1\prime}\boldsymbol{x} - b_i^1 \\ \vdots \\ \boldsymbol{a}_i^{N_i\prime}\boldsymbol{x} - b_i^{N_i} \end{pmatrix} && i = 1, \ldots, m \\ & \boldsymbol{p}_i, \boldsymbol{q}_i \boldsymbol{s}_i \ge \mathbf{0}, t_i \ge 0 && i = 1, \ldots, m\,. \end{aligned} \tag{11}$$

If more than Γ_i of the primitive uncertainties $\tilde{z}_j^i$, $j = 1, \ldots, N_i$ change, the robust solution will be feasible **with very high probability**.

Theorem 4 *Under the model of data uncertainty U, if $\boldsymbol{x}$ is feasible in the robust counterpart (10) then*

$$\Pr\left(\boldsymbol{a}_i(\tilde{\boldsymbol{z}}^i)'\boldsymbol{x} > b_i(\tilde{\boldsymbol{z}}^i)\right) \le \exp\left(-\frac{\Gamma_i^2}{2N_i}\right). \tag{12}$$

Note the original bound proposed in Bertsimas and Sim [2] require symmetrically bounded distribution. However, this condition is relaxed by Chen, Sim and Sun [3]. Nevertheless, tighter bounds can be achieved if the distributions are also symmetrically distributed.

Theorem 5

1. *Suppose $\tilde{z}_j^i$, $j = 1, \ldots, N_i$ are independent symmetrically distributed random variables in $[-1, 1]$, then if $\boldsymbol{x}$ is feasible in the robust counterpart (10), we have*

$$\Pr\left(\boldsymbol{a}_i(\tilde{\boldsymbol{z}}^i)'\boldsymbol{x} > b_i(\tilde{\boldsymbol{z}}^i)\right) \leq B(n, \Gamma_i), \tag{13}$$

where

$$B(n, \Gamma_i) == \frac{1}{2^{N_i}}\left\{(1-\mu)\binom{N_i}{\lfloor \nu \rfloor} + \sum_{l=\lfloor \nu \rfloor+1}^{N_i}\binom{N_i}{l}\right\}, \tag{14}$$

where $\nu = \frac{\Gamma_i + N_i}{2}$ and $\mu = \nu - \lfloor \nu \rfloor$.

2. *For $\Gamma_i = \theta\sqrt{N_i}$,*

$$\lim_{N_i \to \infty} B(N_i, \Gamma_i) = 1 - \Phi(\theta), \tag{15}$$

where

$$\Phi(\theta) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\theta} \exp\left(-\frac{y^2}{2}\right) \mathrm{d}y$$

is the cumulative distribution function of a standard normal.

We next compare the bounds: (12) (Bound 1), (13) (Bound 2), and the approximate bound (15). Table 1 illustrates the choice of Γ_i as a function of N_i so that the probability that a constraint is violated is less than 1%, where we used Bounds 1 and 2 and the approximate bound to evaluate the probability. It is clear that using Bounds 2 or the approximate bound gives essentially identical values of Γ_i, while using Bound 1 leads to unnecessarily higher values of Γ_i when the primitive uncertainties are symmetrically distributed. For $N_i = 200$, we need to use $\Gamma = 33.9$, i. e., only 17% of the number of uncertain data, to guarantee violation probability of less than 1%. For constraints with fewer number of uncertain data such as $N_i = 5$, it is necessary to ensure full protection, which is equivalent to the worst case or the Soyster's method. Clearly, for constraints with large number of uncertain data, the proposed approach is capable of delivering less conservative solutions compared to the Soyster's method.

Price of Robustness for Linear Optimization Problems, Table 1
Choice of Γ_i as a function of N_i so that the probability of constraint violation is less than 1%.

N_i	Γ_i from Bound 1	Γ_i from Bounds 2	Γ_i from Approx.
5	5	5	5
10	9.6	8.2	8.4
100	30.3	24.3	24.3
200	42.9	33.9	33.9
2000	135.7	105	105

Applications

Most of the practical optimization problems are in the form of a mixed integer programming (MIP) model, i. e., some of the variables in the vector $\boldsymbol{x}$ take integer values. Fortunately, the robust model (11) in the case in which the nominal problem is a MIP is still a MIP formulation, and thus can be solved in the same way that the nominal problem can be solved. Moreover, both the deterministic guarantee as well as the probabilistic guarantee (Theorem 2) is still valid. As a result, the robust approach applies for addressing data uncertainty for MIPs. In our computational studies, we apply the robust formulation to a zero-one knapsack problems that are subject to data uncertainty. We examine whether this approach is computationally tractable, and whether it succeeds in reducing the price of robustness.

The zero-one knapsack problem is the following discrete optimization problem:

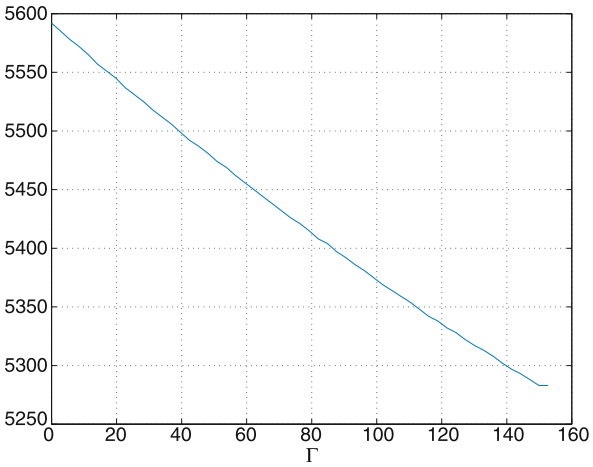
$$\begin{array}{ll} \text{maximize} & \boldsymbol{c}'\boldsymbol{x} \\ \text{subject to} & \boldsymbol{w}'\boldsymbol{x} \leq b \\ & \boldsymbol{x} \in \{0, 1\}^n \end{array}$$

Although the knapsack problem is *NP-hard*, for problems of moderate size, it is often solved to optimality using state-of-the-art MIP solvers. For this experiment, we use CPLEX 6.0 to solve to optimality a random knapsack problem of size, $n = 200$.

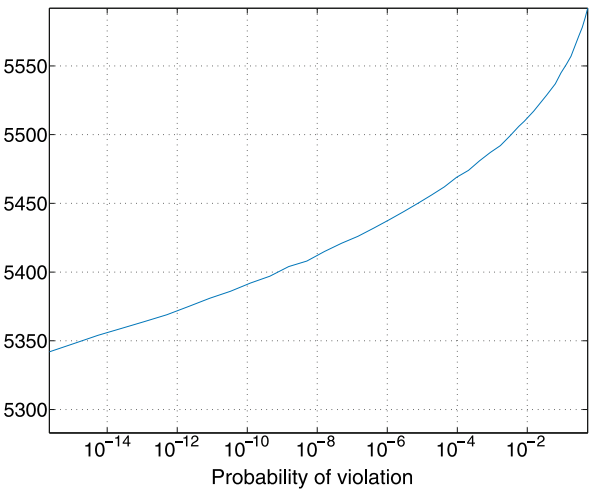
Regarding the uncertainty model for data, we assume the weights $\tilde{w}_i$ are uncertain, independently distributed and follow symmetric distributions in $[\bar{w}_i - \delta_i, \bar{w}_i + \delta_i]$. Hence, under the affine data dependency, we have

$$w_i(\tilde{\boldsymbol{z}}) = \bar{w}_i + \delta_i \tilde{z}_i \qquad i = 1, \ldots, n$$

in which $\tilde{z}_i$, $i = 1, \ldots, n$ are independently distributed and follow symmetric distributions in $[-1, 1]$. An application of this problem is to maximize the total value



Price of Robustness for Linear Optimization Problems, Figure 1
Optimal value of the robust knapsack formulation as a function of Γ



Price of Robustness for Linear Optimization Problems, Figure 2
Optimal value of the robust knapsack formulation as a function of the probability bound of constraint violation given in (13)

of goods to be loaded on a cargo that has strict weight restrictions. The weight of the individual item is assumed to be uncertain, independent of other weights and follows a symmetric distribution. In our robust model, we want to maximize the total value of the goods but allowing a maximum of 1% chance of constraint violation. The robust model is as follows:

$$\begin{aligned} &\text{maximize} && \boldsymbol{c}'\boldsymbol{x} \\ &\text{subject to} && \bar{\boldsymbol{w}}'\boldsymbol{x} + \sum_{i=1}^{n} \delta_i x_i \tilde{z}_i \le b \quad \forall \boldsymbol{z} \in \mathcal{B}(\Gamma) \\ & && \boldsymbol{x} \in \{0,1\}^n \,. \end{aligned}$$

For the random knapsack example, we set the capacity limit, b to 4000, the nominal weight, w_i being randomly chosen from the set $\{20, 21, \ldots, 29\}$ and the cost c_i randomly chosen from the set $\{16, 17, \ldots, 77\}$. We set the weight uncertainty δ_i to equal 10% of the nominal weight. The time to solve the robust discrete problems to optimality using CPLEX 6.0 on a Pentium II 400 PC ranges from 0.05 to 50 s.

Figure 1 illustrates the effect of the protection level on the objective function value. In the absence of protection to the capacity constraint, the optimal value is 5592. However, with maximum protection, that is admitting the Soyster's method, the optimal value is reduced by 5.5% to 5283. In Fig. 2, we plot the optimal value with respect to the approximate probability bound of constraint violation. In Table 2, we present a sample of the objective function value and the probability bound of constraint violation.

Price of Robustness for Linear Optimization Problems, Table 2
Results of Robust Knapsack Solutions

Γ	Probability Bound	Optimal Value	Reduction
2.8	4.49×10^{-1}	5585	0.13%
14.1	1.76×10^{-1}	5557	0.63%
25.5	4.19×10^{-2}	5531	1.09%
36.8	5.71×10^{-3}	5506	1.54%
48.1	4.35×10^{-4}	5481	1.98%
59.4	1.82×10^{-5}	5456	2.43%
70.7	4.13×10^{-7}	5432	2.86%
82.0	5.04×10^{-9}	5408	3.29%
93.3	3.30×10^{-11}	5386	3.68%
104.7	1.16×10^{-13}	5364	4.08%
116.0	2.22×10^{-16}	5342	4.47%

It is interesting to note that the optimal value is marginally affected when we increase the protection level. For instance, to have a probability guarantee of at most 0.57% chance of constraint violation, we only reduce the objective by 1.54%.

Conclusions

The robust methodology provides solutions that ensure deterministic and probabilistic guarantees that constraints will be satisfied as data change. Moreover, the protection level determines probability bounds of constraint violation, which do not depend on the solution of the robust model. As the robust model remains a linear optimization problem, the method naturally applies to discrete optimization problems. It is also possible to build uncertainty sets that are mapped from asymmetrical distributions. The interested reader may refer to Chen, Sim and Sun [3].

References

1. Ben-Tal A, Nemirovski A (2000) Robust solutions of Linear Programming problems contaminated with uncertain data. Math Program 88:411–424
2. Bertsimas D, Sim M (2004) Price of robustness. Oper Res 52:35–53
3. Chen X, Sim M, Sun P (2007) A robust optimization perspective on stochastic programming. Oper Res 55(6):1058–1071
4. Soyster AL (1973) Convex programming with set-inclusive constraints and applications to inexact linear programming. Oper Res 21:1154–1157

Principal Pivoting Methods for Linear Complementarity Problems

Tamás Terlaky
Department Comput. & Software,
McMaster University, Hamilton, Canada

MSC2000: 90C33, 90C20, 65K05

Article Outline

Keywords
Synonyms
LCP
Principal Pivoting
  Principal Pivot Algebra
  Invariant Matrix Classes
Simple Principal Pivoting Methods
General Principal Pivoting Methods
  Criss-Cross Methods
  Lemke's Algorithm
  Symmetric PPM
  Complexity
See also
References

Keywords

Principal pivoting; Block pivot; LCP; Pivot rules

Synonyms

PCP-LCP

LCP

Linear complementarity problems in the general form are given as follows (cf. also ► Linear complementarity problem):

$$\begin{aligned} -Mu + v &= q\,, \\ u, v &\geq 0\,, \\ u_i v_i &= 0\,, \quad i = 1, \ldots, n\,, \end{aligned}$$

where M is a given $(n \times n)$ matrix, $q \in \mathbf{R}^n$. The nonnegative variables u_i and v_i, $i = 1, \ldots, n$, should be nonnegative and complementary. The solvability of LCPs depend on the special properties of the coefficient matrix M. For illustration we will give some well solvable classes in the sequel.

LCPs are natural generalizations of linear programming (linear optimization) and quadratic programming. In the first case

$$M = \begin{pmatrix} 0 & A \\ -A^\top & 0 \end{pmatrix} \quad \text{and} \quad q = \begin{pmatrix} -b \\ c \end{pmatrix}\,,$$

where the matrix A and the vectors b and c are the problem data of the linear optimization problem

$$\min \left\{ c^\top x \colon\ Ax \geq b,\ x \geq 0 \right\}\,.$$

This way a block diagonal *skew-symmetric matrix* M is obtained. When the LCP is obtained from the convex quadratic optimization problem

$$\min \left\{ c^\top x + \frac{1}{2} x^\top Q x \colon\ Ax \geq b,\ x \geq 0 \right\}\,,$$

where Q is a *positive semidefinite symmetric matrix*, then

$$M = \begin{pmatrix} 0 & A \\ -A^\top & Q \end{pmatrix} \text{ and } q = \begin{pmatrix} -b \\ c \end{pmatrix}\,.$$

Here M is a positive semidefinite *bisymmetric matrix*. Bisymmetry means that the matrix has a block diagonal structure, and it is the sum of a symmetric block diagonal positive semidefinite, and a skew-symmetric block diagonal matrix.

Some other classes of solvable LCPs occur when M is

- a *P-matrix*,
- a *sufficient matrix* or, equivalently, a P_**-matrix* [21].

LCPs are solvable by using *pivot methods* or interior point methods (cf. also ► Linear programming: Interior point methods). Here the basics of some of the most popular principal pivot methods are discussed.

Principal Pivoting

When one solves and LCP by using pivoting, then the equality constraints $-Mu+v = q$ always hold. The coefficient matrix of the vector v, the unit matrix might serve as a natural initial basis. Then the basic solution $u = 0$ and $v = q$ is complementary as well. A basis is called *complementary* when exactly one of the complementary variables (u_i, v_i), for all $i = 1, \ldots, n$, is a basis variable.

In a pivot algorithm, because we are working with basic solutions, the equality constraints always hold. We strive for nonnegativity while some variables leave the basis and others enter. Because the initial basis is complementary, it is a natural idea to preserve the complementarity property. This imposes that, if a set of variables leaves the basis, then precisely the variables in the complementary set must enter the basis. Such a step is called *principal pivoting*.

Principal pivoting was introduced by A.W. Tucker [19,20]. The theory of principal pivoting, or *complementary pivot theory*, was extensively studied [4,5,6,7,13,14,16].

Principal Pivot Algebra

When we write the basis (or simplex) tableau of the LCP down, then by leaving out the basis part we get the tableau:

		u
v	q	M

Let $(\gamma, \overline{\gamma})$ be a partition of the index set $\{1, \ldots, n\}$ and let us assume that the *principal submatrix* $M_{\gamma\gamma}$ is nonsingular. Then the current representation of the LCP can be written as

$$v_\gamma = q_\gamma + M_{\gamma\gamma}u_\gamma + M_{\gamma\overline{\gamma}}u_{\overline{\gamma}} ,$$
$$v_{\overline{\gamma}} = q_{\overline{\gamma}} + M_{\overline{\gamma}\gamma}u_\gamma + M_{\overline{\gamma}\overline{\gamma}}u_{\overline{\gamma}} .$$

This in the usual tableau form can be written as A principal pivot, when the variables v_γ leave the basis and their complementary pair u_γ enters the basis can be explained both in the equation and tableau form as follows. Using the assumption that $M_{\gamma\gamma}$ is nonsingular, we may write

$$u_\gamma = q'_\gamma + M^{-1}_{\gamma\gamma}v_\gamma - M^{-1}_{\gamma\gamma}M_{\gamma\overline{\gamma}}u_{\overline{\gamma}} ,$$
$$v_{\overline{\gamma}} = q'_{\overline{\gamma}} + M_{\overline{\gamma}\gamma}M^{-1}_{\gamma\gamma}v_\gamma + M'_{\overline{\gamma}\overline{\gamma}}u_{\overline{\gamma}}$$

where

$$M'_{\overline{\gamma}\overline{\gamma}} = M_{\overline{\gamma}\overline{\gamma}} - M_{\overline{\gamma}\gamma}M^{-1}_{\gamma\gamma}M_{\gamma\overline{\gamma}} ,$$
$$q'_\gamma = -M^{-1}_{\gamma\gamma}q_\gamma ,$$
$$q'_{\overline{\gamma}} = q_{\overline{\gamma}} - M_{\overline{\gamma}\gamma}M^{-1}_{\gamma\gamma}q_\gamma .$$

This representation corresponds to the tableau

Further, the matrix

$$M' = \begin{pmatrix} M^{-1}_{\gamma\gamma} & -M^{-1}_{\gamma\gamma}M_{\gamma\overline{\gamma}} \\ M_{\overline{\gamma}\gamma}M^{-1}_{\gamma\gamma} & M_{\overline{\gamma}\overline{\gamma}} - M_{\overline{\gamma}\gamma}M^{-1}_{\gamma\gamma}M_{\gamma\overline{\gamma}} \end{pmatrix}$$

is called a *principal pivotal transform* of the matrix M. If we define

$$u' = \begin{pmatrix} v_\gamma \\ u_{\overline{\gamma}} \end{pmatrix} , \quad v' = \begin{pmatrix} u_\gamma \\ v_{\overline{\gamma}} \end{pmatrix} , \quad q' = \begin{pmatrix} q'_\gamma \\ q'_{\overline{\gamma}} \end{pmatrix} ,$$

then the LCP can equivalently be reformulated as

$$-M'u' + v' = q' ,$$
$$u', v' \geq 0 ,$$
$$u'_i v'_i = 0 , \quad i = 1, \ldots, n .$$

This way, without loss of generality, we may assume that the current principal pivot transform of the LCP contains v' as the vector of basis variables; $u' = 0$, $v' = q'$ as the current complementary solution; and, M' is the coefficient matrix of the nonbasic variables.

When only one variable leaves the basis, i. e. $\gamma = \{k\}$ for some index, while its complementary pair enters, then this operation is called a *simple principal pivot*, or *diagonal pivot*, because in this case $M_{\{k\}\{k\}} = m_{kk}$, the kth diagonal element of the matrix M. When two variables are coming in and at the same time their pairs leave the basis, then this 2×2 principal pivot is called an *exchange pivot* or *double pivot*.

Invariant Matrix Classes

Let a class of matrices K be given. Then one say that the matrix class is *invariant under principal pivoting*, if for all $M \in K$ and for all principal pivot transforms M' of M the relation $M' \in K$ holds.

It is known, that several matrix classes enjoy the invariance property. The most important invariant matrix classes are the following:

- the class of *P-matrices*;
- the class of P_0*-matrices*;
- the class of *positive definite matrices*;
- the class of *positive semidefinite matrices*;
- the class of *bisymmetric positive semidefinite matrices*;
- the class of *column sufficient matrices*;
- the class of *row sufficient matrices*;
- the class of *sufficient matrices*;
- the class of *Q-matrices*.

Detailed discussion on matrix classes, their characterization, invariance and other properties can be found in [7,16].

Simple Principal Pivoting Methods

Simple principal pivoting methods are using only simple principal pivots, i. e. principal pivots of order one. The best known variant of this method is due to Y. Bard [1] and G. Zoutendijk [22].

Let us assume that the matrix M is a P-matrix. Then all of its principal submatrices, in particular all of its 1×1 principal submatrices are nonsingular.

Due to the P-matrix property, the steps of the algorithm are executable, however there is no guarantee that the method in this form is finite. The finiteness of a variant of this most simple principal pivoting algorithm was proved by K.G. Murty [15]. *Murty's least-index refinement* is as follows:

- Let an ordering of the pairs of the complementary variables be fixed.
- In Step 1, choose the least indexed infeasible basic variable, i. e. let $k = \min\{i\colon v_i' < 0, i = 1, \ldots, n\}$.

Finally, remark that a finite variant of the Zoutendijk–Bard principal pivot rule can also be developed by using an appropriate lexicographic pivot selection rule (cf. also ► Lexicographic pivoting rules).

Principal Pivoting Methods for Linear Complementarity Problems, Table 1

		u_γ	$u_{\bar\gamma}$
v_γ	q_γ	$M_{\gamma\gamma}$	$M_{\gamma\bar\gamma}$
$v_{\bar\gamma}$	$q_{\bar\gamma}$	$M_{\bar\gamma\gamma}$	$M_{\bar\gamma\bar\gamma}$

Principal Pivoting Methods for Linear Complementarity Problems, Table 2

		v_γ	$u_{\bar\gamma}$
u_γ	q'_γ	$M_{\gamma\gamma}^{-1}$	$-M_{\gamma\gamma}^{-1}M_{\gamma\bar\gamma}$
$v_{\bar\gamma}$	$q'_{\bar\gamma}$	$M_{\bar\gamma\gamma}M_{\gamma\gamma}^{-1}$	$M_{\bar\gamma\bar\gamma} - M_{\bar\gamma\gamma}M_{\gamma\gamma}^{-1}M_{\gamma\bar\gamma}$

0	(Initialization) Let the unit matrix, the coefficient matrix of the v variables be the initial complementary basis.
1	(Leaving variable selection) IF no infeasible variable exist, THEN stop; the LCP is solved. Choose an infeasible basis variable, say v'_k.
2	(Principal pivot) Do a simple principal pivot on $m'_{kk} \neq 0$. The chosen infeasible variable leaves the basis while its complementary pair enters. Go to Step 1.

Zoutendijk–Bard principal pivot rule

General Principal Pivoting Methods

As mentioned earlier, general principal pivoting methods use not only simple principal pivots of order one, but also use larger pivot blocks. Frequently, these blocks are not determined at once, the principal blocks are build-up step-by step.

Three variants of more general principal pivoting methods will be described. The (complementary) basis tableaus will be considered in the form as given in Table 1 and Table 2.

Criss-Cross Methods

The first one is the *least-index criss-cross method* [3,9,10,12], which uses first and second order principal

0	(Initialization) Let the unit matrix, the coefficient matrix of the v variables be the initial complementary basis. Let an ordering of the pairs of the complementary variables be fixed.
1	(Leaving variable selection) IF no infeasible variable exist, THEN stop; the LCP is solved. Choose the least indexed infeasible basis variable, say v'_k, where $k = \min\{i : v'_i < 0, i = 1, \ldots, n\}$.
2	(Entering variable selection) IF the pivot (v'_k, u'_k) is possible, i.e. $m'_{kk} \neq 0$, THEN go to Step 3a. IF $m'_{kk} = 0$, THEN look for positive coordinates in row k of M'. IF all coordinates in row k are nonpositive, THEN stop; the LCP is infeasible. Choose the least indexed positive coordinate, say with index r, in row k: $r = \min\{i : m'_{ki} > 0, i = 1, \ldots, n\}$. Go to Step 3b.
3a	(Diagonal pivot) Do a simple principal pivot on m'_{kk}. Then chosen infeasible variable v'_k leaves the basis while its complementary pair u'_k enters. Go to Step 1.
3b	(Exchange pivot) Do a 2×2 principal pivot on the principal submatrix $\begin{pmatrix} m'_{kk} & m'_{kr} \\ m'_{rk} & m'_{rr} \end{pmatrix}$. The variables v'_k and v'_r leave the basis while their complementary pairs u'_k and u'_r enter. Go to Step 1.

Least-index criss-cross pivot rule for sufficient LCPs

pivots. During the execution of the algorithm all the basis solutions are complementary.

Let us assume that the matrix M is a sufficient matrix. As it was mentioned above, the class of sufficient matrices is closed under principal pivot transforms.

Due to the assumption that the matrix M is sufficient and the class of sufficient matrices is closed under principal pivoting, the steps of the algorithm are executable. It is also known that this least index criss-cross method solves sufficient LCPs in a finite number of steps [3,9,10,11,12].

Observe, that if the matrix M is a P-matrix, then only simple principal pivots are executed. Thus, the least index criss-cross rule reduces to the simple principal pivoting algorithm with Murty's least index refinement.

Lemke's Algorithm

The second algorithm is the complementary pivoting algorithm of C.E. Lemke [13,14]. This algorithm can be interpreted on different ways, here only the simplest version is presented.

If $q \geq 0$, then the vector $u = 0$, $v = q$ solves the LCP. If $q \not\geq 0$; then let $d = (-1, \ldots, -1)^\intercal \in \mathbf{R}^n$ and let consider the augmented LCP:

$$\begin{aligned} -Mu + v + d\lambda &= q\,, \\ u, v, \lambda &\geq 0\,, \\ u_i v_i &= 0\,, \quad i = 1, \ldots, n\,. \end{aligned}$$

The solution $u = 0$, $v = q - d\lambda$, where $\lambda = \min\{q_i : 1 \leq i \leq n\}$ has the following properties:

- $-Mu + v + d\lambda = q$ holds;
- all coordinates are nonnegative;
- it is complementary, i. e. $u_i v_i = 0$, $\forall i$;
- there is an index k with both u_k and v_k are nonbasic, and thus zero, further $\lambda = q_k$.

Those solutions of the augmented LCP, which satisfy these properties, will be referred to as *almost complementary solutions*. Lemke's complementary pivot algorithm traverses through a sequence of almost complementary solutions. At the last step the variable λ reaches the value zero, and, this way a complementary solution of the original LCP is obtained.

Lemke's algorithm builds-up a big principal pivot block via a series of pivots which produce almost complementary basic solutions. It does not make explicit use of any special property of the coefficient matrix, however the conclusion, at Step 2, that the LCP is infeasible might not always be valid. For large classes of LCPs, e. g. if M is a *copositive matrix*, the infeasibility conclusion is true. For detailed discussions see e. g. [7].

The algorithm in this form terminates in a finite number of steps only for instances when all almost complementary basis of the augmented LCP are nonde-

0	(Initialization) Form the augmented LCP as discussed above. The initial almost complementary feasible basis is given by $v_1, \ldots, v_{k-1}, \lambda, v_{k+1}, \ldots, v_n$. (The actual basis variables, and the variable that just have left the basis will again be denoted by v'_*, while their complementary nonbasic pairs by u'_*.)
1	(Entering variable selection) The entering variable is u'_k, the complementary pair of the variable which just have left the basis.
2	(Leaving variable selection) IF all coordinates in the column of u'_k are nonnegative, THEN stop; the LCP is infeasible. Make a primal simplex ratio test in the column of u'_k. IF the pivot (λ, u'_k) comes out of the ratio test, THEN go to Step 3a. IF for some r the pivot (v'_r, u'_k) comes out of the ratio test, THEN go to Step 3b.
3a	(Solution) Do a pivot on (λ, u'_k), stop. The resulting feasible basis is complementary; the LCP is solved.
3b	(Pivot) Do a pivot on (v'_r, u'_k) and let $k := r$. Go to step 1.

Lemke's algorithm

generate. Degeneracy resolution is possible on the basis of *lexicographic pivot selection* [7,18] as well as by using least index resolution [3]. A particularly interesting result is the lexicographic Lemke rule of M.J. Todd [18]. To date (2000), this is the only simplex method for oriented matroid programming [2] and, it is a particularly interesting simplex algorithm for linear optimization [17].

0	(Initialization) Let the unit matrix, the coefficient matrix of the v variables be the initial complementary basis; thus the initial solution is $u = 0$, $v = q$. Let $\lambda < \min\{q_i : 1 \le i \le n\}$.
1	(The distinguished variable) IF no infeasible variable exist, THEN stop; the LCP is solved. Choose an infeasible variable as the distinguished variable which has, let say, index k. We have two cases:
1i	The distinguished variable is $u'_k = \lambda < 0$.
1ii	The distinguished variable is $v'_k < 0$. In both cases we use u'_k as the *driving variable*.
2	(Blocking variable selection) Increase the value of u'_k. Let ϑ_k be the largest possible value of u'_k satisfying the following two conditions:
2a	IF v'_k is the distinguished variable THEN v'_k stays nonpositive;
2b	for all basic variables $v'_i \ge \lambda$ holds. IF $\vartheta_k = \infty$, THEN stop; the LCP is infeasible. IF $\vartheta_k = 0$, THEN no pivot is needed; let $u'_k := 0$, u'_i for all $i \ne k$ remain unchanged and let $v' := q + Mu'$. Go to Step 1. ELSE, let r be the index of the variable which blocks the increase of u'_k.
3a	(Diagonal pivot if $m'_{rr} > 0$) IF $m'_{rr} > 0$ THEN do a simple principal pivot on m'_{kk} and make v'_r nonbasic at its blocking lower bound value. IF $r = k$, THEN go to Step 1. IF $r \ne k$, THEN go to Step 2.
3b	(Exchange pivot if $m'_{rr} = 0$) IF $m'_{rr} = 0$ THEN do a 2×2 principal pivot on the principal submatrix $\begin{pmatrix} m'_{kk} & m'_{kr} \\ m'_{rk} & m'_{rr} \end{pmatrix}$. The variables v'_k and v'_r leave the basis while their complementary pairs u'_k and u'_r enter. Let $k := r$ and go to Step 2.

Symmetric principal pivot algorithm

Symmetric PPM

Finally, a general form of the symmetric principal pivoting method is presented. In the major cycles the PPM drives an infeasible variable, called the *distinguished variable* to zero. In this process an artificial lower bound $\lambda < 0$ is used. The nonbasic variables u_i' are either zero,

or equal to $\lambda < 0$. This way, the 'complementary' basic solution $v' = q + Mu'$, in fact, will not be complementary. Let M be a *row sufficient matrix*.

Although the symmetric principal pivoting algorithm is not finite for nondegenerate problems, it also can be turned into a finite one by using least index resolution [3] or lexicographic pivot selections [7].

Several variants of the principal pivoting method – symmetric, parametric, asymmetric etc. – were developed in the last decades. Good surveys of this extensive theory can be found in [7,16].

Complexity

Finally, some notes are due on worst-case behavior. As it was mentioned, and was proved in the cited literature, the presented algorithms are finite when they are furnished with either least-index resolution or lexicographic selection rules. However, in the worst case they may require exponentially many pivots to solve the LCP. The best known exponential example is due to Murty. The data is given as follows: $q_i = -1$, $\forall i$, and

$$m_{ij} = \begin{cases} 0 & \text{if } i > j\,, \\ 1 & \text{if } i = j\,, \\ 2 & \text{if } i < j\,. \end{cases}$$

Clearly the matrix M is an upper triangular P-matrix. Further, the vector $u^\intercal = (0, \ldots, 0, 1)$ and $v^\intercal = (1, \ldots, 1, 0)$ is the unique solution of this LCP. It can be shown that, e. g., Murty's simple principal pivoting method and the least-index criss-cross method needs 2^{n-1} steps to solve this LCP. On the other hand, although it is not proved in general, the results of the paper [8] indicate, that analogously to simplex methods, principal pivoting methods need a polynomial number of iteration in average.

See also

- ▶ Convex-Simplex Algorithm
- ▶ Criss-Cross Pivoting Rules
- ▶ Equivalence Between Nonlinear Complementarity Problem and Fixed Point Problem
- ▶ Generalized Nonlinear Complementarity Problem
- ▶ Integer Linear Complementary Problem
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Least-index Anticycling Rules
- ▶ Lemke Method
- ▶ Lexicographic Pivoting Rules
- ▶ Linear Complementarity Problem
- ▶ Linear Programming
- ▶ Order Complementarity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Pivoting Algorithms for Linear Programming Generating Two Paths
- ▶ Probabilistic Analysis of Simplex Algorithms
- ▶ Sequential Simplex Method
- ▶ Topological Methods in Complementarity Theory

References

1. Bard Y (1972) An ecectic approach to nonlinear programming. In: Anderson RS, Jennings LS, Ryan DM (eds) Optimization. Univ. Queensland Press, Brisbane, pp 116–128
2. Bjorner A, Vergnas M Las, Sturmfels B, White N, Ziegler G (1993) Oriented matroids. Cambridge Univ. Press, Cambridge
3. Chang YY (1979) Least index resolution of degeneracy in linear complementarity problems. Techn Report 79-14, Dept Oper Res, Stanford Univ
4. Cottle RW (1990) The principal pivoting method revisited. Math Program 48:369–385
5. Cottle RW, Dantzig GB (1968) Complementary pivot theory. In: Dantzig GB, Veinott AF (eds) Mathematics of the Decision Sci. Part I. Lect Appl Math, vol 11. Amer. Math. Soc., Providence, RI, pp 115–136
6. Cottle RW, Dantzig GB (1968) Complementary pivot theory of mathematical programming. Linear Alg & Its Appl 1:103–125
7. Cottle R, Pang JS, Stone RE (1992) The linear complementarity problem. Acad. Press, New York
8. Fukuda K, Namiki M (1994) On extremal behaviors of Murty's least index method. Math Program 64:365–370
9. Fukuda K, Terlaky T (1992) Linear complementarity and oriented matroids. J Oper Res Soc Japan 35:45–61
10. Hertog D Den, Roos C, Terlaky T (1993) The linear complementarity problem, sufficient matrices and the criss-cross method. Linear Alg & Its Appl 187:1–14
11. Klafszky E, Terlaky T (1989) Some generalizations of the criss-cross method for the linear complementarity problem of oriented matroids. Combinatorica 9:189–198
12. Klafszky E, Terlaky T (1992) Some generalizations of the criss-cross method for quadratic programming. Math Oper Statist Ser Optim 24:127–139
13. Lemke CE (1965) Bimatrix equilibrium points and mathematical programming. Managem Sci 11:681–689
14. Lemke CE (1968) On complementary pivot theory. In: Dantzig GB, Veinott AF (eds) Mathematics of the Decision

Sci, Part I. Lect Appl Math, vol 11. Amer Math. Soc, Providence, pp 95–114
15. Murty KG (1974) A note on a Bard type scheme for solving the complementarity problem. Opsearch 11(2–3):123–130
16. Murty KG (1988) Linear complementarity, linear and nonlinear programming. Heldermann, Berlin
17. Terlaky T, Zhang S (1993) Pivot rules for linear programming: A survey on recent theoretical developments. Ann Oper Res 46:203–233
18. Todd MJ (1984) Complementarity in oriented matroids. SIAM J Alg Discrete Meth 5:467–485
19. Tucker AW (1960) A combinatorial equivalence of matrices. In: Bellmann R, Hall M (eds) Proc. Symp. Applied Math., vol 10. Amer Math. Soc, Providence, pp 129–140
20. Tucker AW (1963) Principal pivotal transforms of square matrices. SIAM Rev 5:305
21. Valiaho H (1996) P $*$ -matrices are just sufficient. Linear Alg & Its Appl 239:103–108
22. Zoutendijk G (1960) Methods of feasible directions. Elsevier, Amsterdam

Probabilistic Analysis of Simplex Algorithms

PA of SA

Karl Heinz Borgwardt
University Augsburg, Augsburg, Germany

MSC2000: 90C05, 68Q25, 60D05, 52A22

Article Outline

Keywords
Numerical Experiments and Comparison of Variants
Results Under the Sign-Invariance Model
Results Under the Rotation-Symmetry Model
See also
References

Keywords

Linear programming; Average case complexity of algorithms; Stochastic geometry

Solving linear programming problems (cf. also ► Linear programming) is of enormous relevance in real world applications, which contain a lot of data and of unknown variables. Hence, the computational efficiency of solution methods is a crucial criterion for their applicability.

Today, we have a competition between the simplex method (invented around 1947 by G.B. Dantzig) and interior point methods (starting with Karmarkar's algorithm in 1984; cf. also ► Sequential quadratic programming: Interior point methods for distributed optimal control problems).

This article concentrates on simplex methods and on an investigation of their arithmetical effort, measured in terms of the *average number of pivot steps*.

Throughout the paper we discuss the following type of linear programming problems:

$$\begin{cases} \text{maximize} & v^\top x \\ \text{subject to} & a_1^\top x \le b^1, \dots, a_m^\top x \le b^m \\ \text{where} & v, a_1, \dots, a_m \in \mathbf{R}^n, b \in \mathbf{R}^m \\ \text{and} & m \ge n. \end{cases} \tag{1}$$

For abbreviation we use

$$A := \begin{pmatrix} a_1^\top \\ \vdots \\ a_m^\top \end{pmatrix} \in \mathbf{R}^{m\times n} \quad \text{and} \quad b = \begin{pmatrix} b^1 \\ \vdots \\ b^m \end{pmatrix}.$$

The matrix A collects the m gradient vectors to the restrictions as row vectors, and the vector b gives the m capacities. $X := \{x : Ax \le b, x \in \mathbf{R}^n\}$ is the feasible region, respectively the feasible polyhedron, to the problem (1), which can also be written in the form

$$\text{maximize } v^\top x \quad \text{subject to } Ax \le b\,. \tag{2}$$

Other types of programs as

$$\text{maximize } v^\top x \quad \text{subject to } Ax \le b, x \ge 0\,, \tag{3}$$

$$\text{maximize } v^\top x \quad \text{subject to } Ax = b, x \ge 0 \tag{4}$$

and hybrids or variations of such forms can easily be translated into (1). But for form (1) our discussion on the influence of distributions, dimensions and variants can be made much better in geometrical, verbal terms. All the stated results hold – after adaption – for the other forms, too.

If X has vertices and if there are optimal solutions to (1), then there is a vertex in the optimal set. In each vertex of X at least n restrictions of (1) will be active or tight. And in each edge of X at least $n-1$ restrictions are active. Every nonoptimal vertex is incident to an

edge improving the objective. And if an optimal vertex exists, every iterative construction of a connected path over such improving edges leads to the optimal vertex after a finite number of steps. These facts are exploited in the design of the simplex method, which works in two phases:

I Find a vertex $x_0 \in X$;
IF there is no vertex,
THEN STOP.

II Construct a sequence of vertices $x_0, \ldots, x_s \in X$, such that for $i = 0, \ldots, s-1$ the vertices x_i and x_{i+1} are adjacent and $v^T x_i < v^T x_{i+1}$.
IF x_s is optimal OR
IF at x_s the nonexistence of an optimal solution becomes obvious,
THEN STOP at x_s.

Phase I works in a similar manner to Phase II. Since Phase II admits a better geometrical explanation, and is simpler to analyze, we concentrate (for the beginning) on Phase II.

Note that our definition of Phase II still gives the freedom, how we determine the successor vertex (if more then one are possible). A rule for that decision will fix a *'variant'* of the simplex algorithm. The complexity of Phase II (the so-called 'simplex algorithm') is mainly determined by the number s. Less difficult to analyze is the effort to perform a single pivot step, which costs at most $O(mn)$ arithmetic operations for updating an $(m \times n)$-tableau under all reasonable variants.

In this article we are interested in the average case behavior of the random number s, when our problems (1) follow a given distribution. Since nobody knows the 'real world distribution', we have to introduce and to use a self-made *stochastic model* about the appearance of special instances of (1).

Based on that model, we will evaluate the stochastic behavior of s. It is clear that this will massively depend on

- the variant under use;
- the stochastic model, respectively distribution, chosen.

A *probabilistic analysis* of the behavior of an algorithm consists of three essential steps:

- a study of the way the algorithm is working on given, deterministic problem-instances including a characterization of the desired figures (e. g. s) for that instance;
- a consensus about an underlying stochastic model on the distribution of occurring problem-instances;
- a cumulation over all possible instances, weighted with their occurrence probability, leading to stochastic information on the *random behavior*.

So, we study the procedure of a deterministic algorithm, which is employed to solve random problem-instances.

This stands in contrast to the situation with *randomized algorithms*, where random parameters decide how the algorithm shall proceed in solving a given, deterministic problem.

Throughout the paper we shall rely on a *nondegeneracy assumption*: All submatrices of (A, b) and of $(A)^\intercal$, v are of full (i. e., maximal) rank.

This is compatible with our models either by direct conditioning or by the fact that in such a probabilistic model the set of degenerate problems is a nullset.

In this paper, we shall briefly report on experiments and their (limited) information-value. After that we come to two different stochastic models which admitted a successful probabilistic analysis. The first is the sign-invariance model, whose analysis reached its summit in the middle of the 1980s. And the second is the rotation-symmetry model, whose evaluation had started even earlier. But the refinement of that approach is still going on.

Numerical Experiments and Comparison of Variants

The first idea to learn more about the *average case behavior* of s is to carry out controlled numerical (Monte-Carlo) experiments. For that purpose, one has to fix several dimension-pairs (m, n), to use a stochastic model for generating the data, and to solve the created problem-instances by application of a given variant.

These experiments can be employed for a variety of purposes, as for forecasting the number of pivot steps s (on the basis of (m, n) for a fixed variant and model) or for comparing different variants or for recognizing the different influences of stochastic models.

All of that had been done and tried in the past. Studying the huge number of reports on such experiments leads to a very confusing and frustrating im-

pression. Since stochastic models, employed variants, dimensions and problem-types vary excessively, the results and methods can hardly be compared. In particular, it is not possible to summarize the outcome briefly, since all the test parameters would have to be mentioned exactly. So, we refer to the very informative survey by R. Shamir in [19], who comes to the following overall conclusion:

> The smaller dimension n (respectively, the dimension of the polyhedra) enters the mean value function of s in a slightly superlinear way and the larger dimension m (respectively, the number of inequality restrictions, including sign-constraints) has only a significantly sublinear influence.

Easier to understand and to interpret are experiments, when they are done parallel to a theoretical study, because then both results, the empirical and the theoretical one, can be checked whether they justify and confirm each other. This has been achieved in experiments for the so-called *rotation-symmetry model* (RSM):

- Let $b = \vec{1}$ and let $a_1 \ldots, a_m, v$ (and an auxiliary vector u) be distributed independently, identically and symmetrically under rotations on $\mathbf{R}^n \setminus \{0\}$.

Note that only $b > 0$ is essential. Choosing $b = \vec{1}$ (i. e. the vector of ones) means a simplifying standardization only.

The experiments could more or less confirm the theoretical results on $\mathsf{E}_{m,n}(s)$ (the expected number of pivot steps required for (m, n) problems). These theoretical results will be presented later. Rather informative was the comparison of the behavior of different variants and of the influence of different stochastic distributions. In [11] we tested seven variants belonging to three categories (A, B, C), whose geometric description can be given as follows.

Note that in each vertex a decision has to be made, which one of the (exactly) n tight restrictions should be de-activated. This means that a choice among the subset of the improving edges (originating from that vertex) is made. The current basis is the set of the n gradients $(a)_i$) corresponding to the active restrictions at the current vertex.

- *Category A*: Variants exploiting information on the shape of X and on the objective $v^\intercal x$.
  - (*rule of steepest ascent*) Choose that incident improving edge with smallest angle to the gradient of the objective function.
  - (*rule of greatest improvement*) Take that edge which leads to the maximal improvement of $v^\intercal x$ in the next step.
- *Category B*: Variants exploiting information on the objective only.
  - (*Dantzig's rule*) Since a vertex x is optimal if and only if v is in the cone of the gradients a_i of the current basis, we can calculate in each step the representation of v by that basis of $\mathbf{R}^n$. So every basis-gradient is associated with its v-representation coordinate. Since optimality requires a completely nonnegative representation, Dantzig's rule suggests to take the edge that de-activates the restriction whose gradient has the most negative coordinate.
  - (*shadow-vertex algorithm* or *parametric rule*) This is the variant for which theoretical studies worked very well. The results will be presented in the following sections. Therefore, we explain it in detail. This variant leads from a vertex optimizing an alternative objective $u^\intercal x$ to the optimal solution for $v^\intercal x$ (or an unbounded edge), by providing all optimal vertices to the family of objectives $(\lambda\, v + u)^\intercal x$ with $\lambda \in [0, \infty)$.
    For λ starting at 0 and increasing, the sequence of optimal vertices gives just the parametric simplex path, which had also been constructed in the early parametric variant of S.I. Gass and T.L. Saaty. They had introduced the parametric concept for another type of problems and without the geometric shadow-vertex interpretation: If we project X on $\mathrm{Span}(u, v)$, then our variant constructs a path starting in the $u^\intercal x$-optimum visiting only *shadow-vertices*. These are vertices which keep their vertex-property even after the projection. The projection-image of the constructed path is again a path along the boundary of the two-dimensional image of X. The choice of the right edge is organized by calculation of the basis representations of v and u (as in Dantzig's rule) and by minimizing the quotient of corresponding coordinates.
- *Category C*: Rules evaluating combinatorial principles only.

- (*rule of random choice*) Select randomly one of the improving edges.
- (*rule of justice*) De-activate the tight restriction that had been active most often.
- (*rule of Bland*) De-activate the tight restriction that has the least original index.

These variants were compared under three different rotation-symmetric distributions for the vectors a_i:

- *uniform distribution* on ω_n (the unit sphere of $\mathbf{R}^n$);
- *uniform distribution* on Ω_n (the full unit ball of $\mathbf{R}^n$);
- *Gaussian distribution* on $\mathbf{R}^n$.

In general, it turned out that results for Gaussian distribution were better (smaller) than for unit ball and the latter were better than the unit sphere results. This effect is simply caused by different 'redundancy-rates'. A restriction is *redundant*, if its existence or nonexistence has no impact on the shape of X. Here, the ith restriction is redundant if and only if a_i belongs to $\mathrm{Conv}(0, a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_m)$. This will never happen when all points come from ω_n and rather seldom when all points come from Ω_n, but very often when the points are Gaussian distributed. And, it is obvious, that under normal circumstances a problem becomes easier, if more restrictions are redundant, respectively if the redundancy rate is high.

The quality of the different variants can be ordered consistently. The best performance shows the rule of steepest ascent. It is slightly better than the rule of greatest improvement. These two variants show a very good performance in particular when the current vertex is still far away from the optimal one.

A bit worse are the variant of Dantzig and the shadow-vertex algorithm. The reason may be that they do not exploit information on the polyhedron itself (which may make the edge-choice more ineffective, but saves computation time in the single pivot step).

Considerably worse is the performance of the combinatorial variants. The best among these is random choice, followed by rule of justice, and finally comes Bland's rule.

The overall impression is that the differences between Category A and Category B are not dramatic, but that the differences between Category B and Category C are striking.

We have also tested the standard deviation of s and the more meaningful quotient between standard deviation and mean value (for the number of pivot steps). This quotient was less, but close to 1, when m was in the order of n. But the quotient became quite small for $m \gg n$. We understand this as a hint that in the RSM for $m \to \infty$ and fixed n (the 'asymptotic case') the shape of X, the number of facets as well as their size, and the length of edges will stabilize more and more.

However, all these experiments and their outcome are not at all satisfactory for a final judgement. One reason is that the computation time for a sufficient number of repetitions of the experiments is irresponsibly high. Hence we cannot advance to reasonably high dimensions. Also complexity theory investigations cannot be settled by limited experiments. A third argument concerns potential regression analysis attempts based on the data of the results. It is almost impossible to modelize the qualitative structure of $\mathsf{E}_{m,n}(s)$ as a function of m and n with parameters to be specified by the regression, as long as we do not understand (theoretically) the interaction between m, n and the stochastic model. Many such attempts failed as the model structures did always fit only in a bounded range of m and n.

Much more meaningful is the outcome of theoretical (arbitrary dimension) considerations. In the following, we present two successful approaches.

Results Under the Sign-Invariance Model

The first investigation under this kind of model was done by S. Smale [20]. He analyzed problems of type (3):

$$\text{maximize } v^\top x \quad \text{subject to } Ax \le b, x \ge 0$$

and treated this problem as a special case of the linear complementarity problem (cf. ▶ Linear complementarity problem)

$$\begin{cases} \text{Find} & w, z \in \mathbf{R}^{m+n} \\ \text{such that} & \text{for given } q \in \mathbf{R}^{m+n}, \\ & M \in \mathbf{R}^{(m+n)\times(m+n)}, \\ & w - Mz = q,\ w^\top z = 0, \\ & w \ge 0, z \ge 0. \end{cases} \tag{5}$$

When

$$M = \begin{pmatrix} 0 & -A \\ A^\top & 0 \end{pmatrix} \quad \text{and} \quad q = \begin{pmatrix} b \\ -v \end{pmatrix},$$

then a solution of (5) yields a solution of (3) and its dual.

As solution procedure, Smale employs the *Lemke algorithm*. One starts with a solution for $\vec{1}$ replacing q. Then one moves on $[\vec{1}, q]$ forward and performs pivot steps whenever one of the coordinates of the vector w reaches the value 0, in order to keep all components nonnegative.

The analysis amounted to the question, how many cones of a special type will be intersected by a line segment. This is a typical question for a parametric algorithm. The *expected number of pivot steps* $E_{m,n}(s^L)$ was analyzed under the following stochastic model:

1) (A, b, v) is distributed absolutely continuous.
2) The columns of (A, b) and v are independent.
3) The measure of (A, b) is invariant under coordinate permutations in columns of (A, b).

Smale proved for problems distributed under that model:

Theorem 1 ([20])

$$E_{m,n}(s^L) \leq C(n)\,(1 + \ln(m+1))^{n(n+1)}\ .$$

This shows polynomiality in m (for fixed n), but not in n. $C(n)$ is an (exponential) function of n.

Smale's studies gave a motivation for the analysis of the so-called *sign-invariance model* (SIM). It is extremely simple and only relies on a finite number of reflections and symmetries.

Let A, b and v define a nondegenerate dataset for problem (3). Let the occurrence of $\begin{pmatrix} A & b \\ v^\top & 0 \end{pmatrix}$ and of $\begin{pmatrix} S_1 A S_2 & S_1 b \\ v^\top S_2 & 0 \end{pmatrix}$ be equiprobable for every sign matrix $S_1 \in \mathbf{R}^{m\times m}$ and $S_2 \in \mathbf{R}^{n\times n}$. (A *sign matrix* is a diagonal matrix with +1 or −1 in the diagonal entries). To explain the impact of that model, it suffices to consider a somehow relaxed version of sign-invariance, the so-called *flipping model*, where we consider only the sign matrix S_1 and deal with problem instances of form (1):

$$\begin{cases} \max & v^\top x \\ \text{s.t.} & a_1^\top x \leq\geq b^1, \ldots, a_m^\top x \leq\geq b^m. \end{cases} \tag{6}$$

Here, $\leq\geq$ indicates that one of the relations $\leq$ or $\geq$ shall be valid in the formulation of the instance. We get $\leq$ if $s_{ii} = 1$ and $\geq$ if $s_{ii} = -1$ in (6), respectively in S_1. Since all sign matrices S_1 shall be equiprobable, this means that we independently determine the m directions of the relations, each one with probability ½ for $\leq$ and with probability ½ for $\geq$.

By the way, we generate exactly 2^m problem-instances out of one data set. The idea of averaging is to solve all 2^m instances, to sum up the required pivot steps and to divide by the number of instances.

This set of problem instances can be solved (as far as Phase II is concerned) by application of the shadow-vertex algorithm explained in the section above. There we realize a simplex-path over all (temporarily) optimal vertices when we traverse the set of objectives $(u + \lambda v)^\top x$ for $\lambda \geq 0$.

If we add the corresponding set of (optimal) vertices for negative values of λ, then the total set will be called the set of *co-optimal vertices*.

With s for the number of pivot steps, S_{coop} for the number of co-optimal vertices and S for the number of shadow-vertices, the following relation is obvious: $s \leq S_{\text{coop}} \leq S$.

Using simple combinatorial enumeration arguments, I. Adler and M. Haimovich [13] showed

Theorem 2 ([13]) *For type (1) under SIM:*

$$E_{m,n}(S_{coop} | \textit{a co-optimal path exists}) \leq n\frac{m-n+2}{m+1} \leq n\ .$$

So far, the analysis considers only the procedure of moving from a $u^\top x$-optimum to a $v^\top x$-optimum. But this does not fit exactly into a probabilistic analysis of a complete solution method (as Smale's method), because the $u^\top x$-optimum is not given beforehand and calculating it would be as troublesome as calculating the $v^\top x$-optimum.

In 1983–1984, the combination of this result with a design of a complete algorithm was done in three papers by M.J. Todd [21], Adler and N. Megiddo [3], Adler, R.M. Karp, Shamir [2]. They all came to the same result for $\mathsf{E}_{m,n}(s^t)$, the expected number of pivot steps required to solve the LP completely (including Phases I and II).

Theorem 3 ([2,3,21]) *For problems of type (1), respectively of type (3), distributed under SIM, the expected number of pivot steps for the complete solution by a lexicographic version of Lemke's algorithm (s^t) is*

$$E_{m,n}(s^t) \leq 2(n+1)^2\ ,$$

(respectively, $\leq 2\ \min(m^2, n^2)$).

In the first two papers, analyzing type-(3) problems, the proof was based – as in Smale's analysis – on the evaluation of a probability that a typical cone is intersected by a line. But this time, this is the line

$$[d, \vec{1}]$$

with $0 < d = (\delta, \delta^2, \ldots, \delta^{m+n})^\mathsf{T}$ and δ as small as desired.

Closer to our geometrical interpretation and easier to explain is the third approach in [2].

To explain the solution process of a type-(1) problem, we use

$$X^{(n+k)} := \left\{x \in \mathbf{R}^n : \; a_1^\top x \le b^1, \ldots, a_{n+k}^\top x \le b^{n+k}\right\}$$

for $0 \le k \le m - n$ and $X^{(m)} = X$.

The following *complete* algorithm works directly in the space $\mathbf{R}^n$ and is called the *lexicographic variant of the constraint-by-constraint method*:

Initialization
Determine the unique vertex $\overline{x} \in X^{(n)}$ and choose u as $u = \delta^1 a_1 + \ldots + \delta^n a_n$ with $\delta > 0$ sufficiently small.
Stage k $(1 \le k \le m - n)$
START at $\overline{x}$, the maximal vertex for $u^\top x$ on $X^{(n+k-1)}$.
IF $\overline{x} \in X^{(n+k)}$
THEN go to stage $k + 1$.
ELSE
use the shadow-vertex algorithm to improve the value of $a_{n+k}^\top x$ (note that so far $a_{n+k}^\top \overline{x} > b^{n+k}$),
START at $\overline{x}$ and minimize $a_{n+k}^\top$ on $X^{(n+k-1)}$.
STOP as soon as $a_{n+k}^\top x \le b^{n+k}$.
On the last traversed edge find a point $\widetilde{x}$ with $a_{n+k}^\top \widetilde{x} = b^{n+k}$.
ENTER stage $k + 1$ with $\widetilde{x}$; replace $\overline{x}$.
This is possible, because we have moved on a co-optimal path, hence $\widetilde{x}$ maximizes $u^\top x$ on $X^{(n+k)}$.
STOP if it is impossible to achieve $a_{n+k}^\top x \le b^{n+k}$, because then the original problem is infeasible.
Stage $m - n + 1$
START at $\overline{x}$, which maximizes $u^\top x$ on $X^{(m)} = X$.
Apply the shadow-vertex algorithm to find the optimal point for $v^\top x$ or discover that $v^\top x$ is unbounded on X.

In principle, this amounts to solving $(m - n + 1)$ problems, for which the average number of steps is less n each (Theorem 2). But now, due to the lexicographical choice of u, it can be exploited that (when we enter stage $k + 1$) most of the work to optimize the current objective has already been done in earlier stages. Thus, the effort of stage $k + 1$ becomes much smaller than n.

Finally, the order of the total average number of steps is $O(n^2)$ instead of $O(mn)$.

With slight additional conditions on the distributions of the A entries, Adler and Megiddo [3] could establish also a lower bound of type $C \cdot n^2$.

They argued that for $m \le 2n$, since the share of feasible problems is at least $n^{-\frac{1}{2}}$, the conditional expected number of pivot steps for solving LP's of that model, under the condition that the problem instance is feasible, is $O(n^{2.5})$.

As for every probabilistic model, one should ask about the direct impact of the model on the results.

An important feature of SIM is the fact that many instances will be infeasible, precisely

$$\frac{\text{number of feasible instances}}{\text{number of generated problems}} = \frac{\binom{m}{0} + \cdots + \binom{m}{n}}{2^m} \to 0$$

as $m \to \infty$ while n is fixed.

Only conditioning on feasible problem instances avoids averaging over a lot of easy problems. But even if we do so, we meet a remarkably small expected number of vertices:

$$\mathsf{E}_{m,n}(\text{vertices per feasible instance}) = \frac{2^n \binom{m}{n}}{\binom{m}{0} + \cdots + \binom{m}{n}},$$

which is less than 2^n and converges to that value for $m \to \infty$, n fixed.

Now it is not astonishing, that for a large class of variants the average number of pivot steps for the complete solution will be bounded from above by a function of n only (cf. [1]).

But the most important cause for simplification of problem instances with $m \gg n$ is the *average redundancy rate* (the share of the restrictions without impact on X). This expected number (conditioned on feasible

problems) is

$$\frac{\binom{m-1}{n}}{\binom{m}{0}+\cdots+\binom{m}{n}} \to 1$$

as $m \to \infty$ while n is fixed.

Simultaneously, the absolute number of nonredundant constraints (conditioned on feasibility) tends to $2n$ for $m \to \infty$ and fixed n.

So, for $m \gg n$, even for the feasible instances, the *average nonredundancy rate* will be very small. This will of course make these problems easy. And it says that the sign-invariance model gives reasonable and meaningful information only for $m = O(n)$.

SIM relies on symmetries and reflections only. The combinatorial methods for evaluation make it unlikely that a calculation of higher moments of the s-distribution can easily be done. Besides that, the model is somehow inflexible. For every set of data, the reflection procedure leads to exactly the same cumulated statistical characteristics in the total set of the 2^m instances. There is no way to choose a desired redundancy share or a size of the expected number of vertices and to parametrize certain figures in order to study their impact.

Apparently (in particular for $m \gg n$), the small upper bounds in Theorem 2 and Theorem 3 do rather *reflect the special properties of the model* than confirm the efficiency of the simplex method, which had been pointed out in [1].

Results Under the Rotation-Symmetry Model

The theoretical analysis based on the rotation-symmetry model RMS above started in 1977 [4] by work of K.H. Borgwardt and is still (1998) developing. The most important among his results, a *polynomial upper bound* for the *expected number of shadow-vertices*, was derived in 1996–1997 [10] and it had predecessors with slightly cruder bounds in 1987 [6] and 1982 [5].

Theorem 4 ([10]) *For every rotation-symmetry distribution as in RMS and for every pair (m, n) with $m \geq n$, the expected number S of shadow-vertices (and of pivot steps s in Phase II) satisfies*

$$4E_{m,n}(s) \approx E_{m,n}(S) \leq \text{const} \cdot m^{\frac{1}{(n-1)}} \cdot n^2 .$$

This result and its predecessors have been derived by translating the question about S into the dual space of the vectors a_i. Candidates for being a vertex are only the $\binom{m}{n}$ basic solutions x_Δ solving a system of n equations

$$a_{\Delta^1}{}^\top x = 1, \ldots, a_{\Delta^n}{}^\top x = 1$$

with

$$\Delta = \{\Delta^1, \ldots, \Delta^n\} \subset \{1, \ldots, m\} .$$

x_Δ is actually a vertex if all other restrictions are satisfied, i. e. $a_i^\top x_\Delta \leq 1$ for all $i \notin \Delta$.

It becomes a shadow-vertex if the projection on $\text{Span}(u, v)$ preserves its vertex property.

Now there is a one-to-one correspondence

$$x_\Delta \leftrightarrow \Delta = \{\Delta^1, \ldots, \Delta^n\} \leftrightarrow \text{Conv}(a_{\Delta^1}, \ldots, a_{\Delta^n}) .$$

Besides $X = \{x : Ax \leq b\}$ we consider its *polar polyhedron* $Y = \text{Conv}(0, a_1, \ldots, a_m)$.

The following equivalences enable us to derive the average number of shadow-vertices directly from the input data:

Lemma 5

1) *x_Δ is a vertex of X if and only if $\text{Conv}(a_{\Delta^1}, \ldots, a_{\Delta^n})$ is a facet of Y.*
2) *A vertex x_Δ is a shadow-vertex of X if and only if*

$$\text{Conv}(a_{\Delta^1}, \ldots, a_{\Delta^n}) \cap \text{Span}(u, v) \neq \emptyset.$$

The addition theorem for expectation values and the symmetry of index choices yield

$$\mathsf{E}_{m,n}(S) = \binom{m}{n} \mathsf{P}\left(\begin{array}{c} \text{Conv}(a_1, \ldots, a_n) \text{ is a facet} \\ \text{intersected by } \text{Span}(u, v) \end{array} \right) .$$

Here, one integrates over all possible configurations of $a_1, \ldots, a_m, u, v$ and weights with regard to the underlying distribution. The resulting multiple integral is very hard to evaluate. For the case of moderate dimensions (m, n arbitrary), we could only compare our integral with known results about a closely related integral. Much more efficient are the tools for evaluating the so-called *asymptotic case* ($m \to \infty$, n fixed), because there the integrals behave like Laplace integrals and can conveniently be evaluated. So it was much easier to derive *asymptotic results* for specific RSM-distributions.

In the sequel, we write $\mathsf{E}_{m,n}(S) \sim f(m,n)$ as $m \to \infty$, n fixed, when we mean that

$$C_1 \le \liminf_{\substack{m\to\infty \\ n \text{ fixed}}} \frac{\mathsf{E}_{m,n}(S)}{f(m,n)} \le \limsup_{\substack{m\to\infty \\ n \text{ fixed}}} \frac{\mathsf{E}_{m,n}(S)}{f(m,n)} \le C_2$$

for certain constants $C_1, C_2 > 0$. Besides that, we speak of an *RSM-distribution on Ω_n with algebraically decreasing tail* if

$$\mathsf{P}(\|x\| \ge r) \sim (1-r)^\gamma$$

for $r \to 1$ for a $\gamma > 0$.

Theorem 6 ([4,6,14,16,18]) *For fixed n and $m \to \infty$, the following distributions lead to the following behavior of $E_{m,n}(S)$:*

- *Gaussian distribution on $\boldsymbol{R}^n$:*

$$E_{m,n}(S) \sim \sqrt{\ln mn}^{\frac{3}{2}} .$$

- *Uniform distribution on Ω_n:*

$$E_{m,n}(S) \sim m^{\frac{1}{(n+1)}} n^2 .$$

- *Uniform distribution on ω_n:*

$$E_{m,n}(S) \sim m^{\frac{1}{(n-1)}} n^2 .$$

- *There are RSM-distributions such that*

$$E_{m,n}(S) \sim C(n) .$$

- *For RSM-distributions on Ω_n with algebraically decreasing tail:*

$$E_{m,n}(S) \sim m^{\frac{1}{(n-1+2\gamma)}} n^2 .$$

These results should be compared with corresponding results on the average number of vertices of X, respectively facets of Y, denoted by $\mathsf{E}_{m,n}(V)$ in our model.

Theorem 7 ([6,8,16]) *For fixed n and $m \to \infty$, the following distributions lead to the following behavior of $E_{m,n}(V)$:*

- *Gaussian distribution on $\boldsymbol{R}^n$:*

$$E_{m,n}(V) \sim [\ln m]^{\frac{(n-1)}{2}} 2^n \cdot \pi^{\frac{(n-1)}{2}} \cdot \frac{1}{\sqrt{n}} .$$

- *Uniform distribution on Ω_n:*

$$E_{m,n}(V) \sim m^{\frac{(n-1)}{(n+1)}} 2^{\frac{n}{2}} \times \pi^{\frac{n}{2}} n^{-\frac{1}{4}} (n+1)^{\frac{(n-1)}{2}} .$$

- *Uniform distribution on ω_n:*

$$E_{m,n}(V) \sim m^{\frac{(n-1)}{(n-1)}} 2^{\frac{n}{2}} \times \pi^{\frac{n}{2}} n^{-\frac{5}{4}} (n-1)^{\frac{(n-1)}{2}} .$$

- *For distributions on Ω_n with algebraically decreasing tail:*

$$E_{m,n}(V) \sim m^{\frac{(n-1)}{(n-1+2\gamma)}} 2^{\frac{n}{2}} \pi^{\frac{n}{2}} n^{-\frac{3}{4}} \times (n-1+2\gamma)^{\frac{(n-1)}{2}} \left(\frac{n}{2}\right)^{-\frac{(n-1)}{2(n-1+2\gamma)}} .$$

Obviously, the simplex method is able to select a rather short path through the huge set of vertices. Hereby it visits (on the average and approximately) only the nth root of the total number of available vertices.

Another very important point is the *variance of the number of shadow-vertices*, respectively of the number of required pivot steps. Due to the technical difficulties mentioned above, so far (1998) only the asymptotic case has been analyzed. K. Küfer [17] showed

Theorem 8 ([17]) *For distributions with algebraically decreasing tail on Ω_n, the quotient of variance and square of expected value behaves asymptotically as follows:*

$$\frac{\mathrm{Var}_{m,n}(s)}{E^2_{m,n}(s)} \sim \frac{1}{n} ,$$
$$\frac{\mathrm{Var}_{m,n}(S)}{E^2_{m,n}(S)} \sim m^{\frac{-1}{(n-1+2\gamma)}} .$$

Here, s is the number of pivot steps of the shadow-vertex algorithm (Phase II) and S is the number of shadow-vertices.

So far, we have dealt only with a fictive Phase II algorithm, starting at an optimal vertex for an auxiliary objective. But this vertex is impracticable to find. Now let us talk about a safe Phase I.

A special feature of our problems is the feasibility of the origin, which makes (in contrast to the sign-invariance model) every instance feasible. Based on that information, we can employ a method (cf. [5] and[6]), which applies the shadow-vertex algorithm $n-1$ times, and each time the dimension of the problem is increased. In each of these stages all the stochastic requirements of our model are satisfied.

Here we introduce

$$X^{(k)} = \left\{x\colon\ Ax \le \vec{1}, x^{k+1} = \cdots = x^n = 0\right\}\ ,$$

and formulate the *dimension-by-dimension algorithm*:

Initialization(Stage k=1)
Starting from the origin, find a vertex of $X^{(1)}$ maximizing $v^\top x = v^1 \cdot x^1$.
IF this maximal vertex does not exist,
THEN STOP.
Stage k, $2 \le k \le n$,
Use the optimal point $(\overline{x}^1, \ldots, \overline{x}^{k-1}, 0, \ldots, 0)^\top$ for $v^\top x$ on $X^{(k-1)}$, which is located on an edge of $X^{(k)}$.
1. Find an adjacent vertex in $X^{(k)}$ to that edge.
2. Apply the shadow-vertex algorithm using $e_k^\top x$ and $v^\top x$ as pair of objectives for maximizing $v^\top x$ on $X^{(k)}$.
   IF $v^\top x$ turns out to be unbounded on $X^{(k)}$, STOP.
3. IF $k < n$, set $k = k+1$ and enter the next stage.
   IF $k = n$, PRINT the optimal vertex for X.

One can derive an upper bound for this cumulation of $n-1$ applications of the shadow-vertex algorithm by summing up all the expected numbers of shadow-vertices. But this would significantly overestimate the actual number of pivot steps in this algorithm, since we would ignore that the original distribution comes from $\mathbf{R}^n$ and that only projection distributions (from $\mathbf{R}^n$ to $\mathbf{R}^k$) can determine the behavior in stage k. Since the set of projection distributions is only a small subset of the RSM-distributions in dimension k, the corresponding bound for the expected number of steps in stage k is much better. Consequently, we obtain the following result, which also holds for problems of type (3), including sign-constraints:

Theorem 9 ([9,10]) *For every pair (m, n) with $m \ge n$ and every RSM-distribution on $\mathbf{R}^n$, the expected total number of pivot steps for the dimension-by-dimension algorithm satisfies*

$$E_{m,n}(s^t) \le m^{\frac{1}{(n-1)}} \cdot n^3 \cdot C\ ,$$

as well for problems of type (1) as for problems of type (3).

But, as observed in the analysis of the constraint-by-constraint method (cf. Theorem 3), it is plausible that most work of optimizing in stage $k+1$ has already been prepared in prior stages, such that the actual number of steps in stage $k+1$ is much smaller. This was precisely clarified by G. Höfner [14] for the asymptotic case:

Theorem 10 ([14]) *For every RSM-distribution the expected total number of pivot steps in the dimension-by-dimension algorithm satisfies*

$$E_{m,n}(s^t) \sim m^{\frac{1}{(n-1)}} \cdot n^{\frac{5}{2}}$$

when $m \to \infty$ and n is fixed for problems of type (1) and (3).

It must be clear that this algorithm is crude and lengthy and has been introduced only for meeting the conditions of RSM and for making the probabilistic analysis possible.

In order to confirm the 'folklore' observation, that Phase I can be done with an effort not exceeding that of Phase II, Höfner analyzed another complete algorithm. But unfortunately, this method is assured to work only in the asymptotic case.

1) Solve the problem

$$\max \vec{1}^\top x \quad \text{subject to } Ax \le \vec{1}, x \ge 0$$

by use of the shadow-vertex algorithm starting at the vertex 0. The optimal vertex $\overline{x}$ will (in the asymptotic case) with extremely high probability be a vertex of $X = \left\{x\colon\ Ax \le \vec{1}\right\}$.

2) Start the shadow-vertex algorithm at $\overline{x}$, forget about the sign constraints and optimize $v^\top x$ on X.

It can be shown that both applications of the shadow-vertex algorithm require (on the average) at most $m^{1(n-1)} \cdot n^2 \cdot \text{const}$ pivot steps.

So, this is an algorithm with a Phase I effort not exceeding that of Phase II.

So far, the plausible and natural very good behavior of Phase I can only be guaranteed in the asymptotic case. In the moderate cases, the situation is similar to that of the sign-invariance results, where the constraint-by-constraint method needs a factor n more pivot steps than Phase II does.

As we have discussed the advantages and drawbacks of SIM, we now consider similar questions for RSM. Seemingly it is a tremendous advantage of RSM that it

generates only (the hard) feasible problems. But simultaneously it turns out to be a drawback that the given Phase I algorithms are designed in a way such that they exploit this fact and are dependent on the guarantee of '0 being feasible'.

One way to overcome that drawback lies in the following idea. Remember that we want to solve all problem instances of the type

$$\begin{cases} \max & v^\top x \\ \text{s.t.} & a_1^\top x \le b^1, \ldots, a_m^\top x \le b^m \end{cases}$$

with arbitrary values of b^i (not necessarily positive).

For integrating all these problems in our analysis, we use a 'homogenization method'. We introduce the notation $P_n := \{x : Ax \le b\}$ and reformulate our restrictions as

- $a_i^\top x \le b^i$ corresponds to $a_i^\top x \le 1 - \widetilde{b^i}$ when $b^i = 1 - \widetilde{b^i}$.

So we can demand that

$$a_1^\top x + \widetilde{b^1} \cdot 1 \le 1, \ldots, a_m^\top x + \widetilde{b^m} \cdot 1 \le 1$$

and define a polyhedron in $\mathbf{R}^{n+1}$ by

$$\left(a_i^1, \ldots, a_i^n, \widetilde{b^i}\right) \begin{pmatrix} x^1 \\ \vdots \\ x^{n+1} \end{pmatrix} \le 1 ,$$

which means that $a_i^\top \overline{x} + \widetilde{b^i} x^{n+1} \le 1 (\overline{x} \in \mathbf{R}^n)$ for $i = 1, \ldots, m$.

This system defines a new polyhedron $P_{n+1} \subset \mathbf{R}^{n+1}$. The set of feasible points with $x^{n+1} = 0$ is a one-to-one copy of $\left\{x : \; Ax \le \vec{1}\right\}$. The set of feasible points with $x^{n+1} = 1\}$ corresponds one-to-one to the set of points in P_n.

It is now clear that in level $\{x^{n+1} = 0\}$ the problem satisfies all RSM-requirements. So we can solve the optimization problem for $v^\top x$ on that artificial polyhedron. But then we can use one further stage $(n + 1)$ of the dimension-by-dimension algorithm to reach level $\{x^{n+1} = 1\}$ (by maximizing $x^{n+1} = e_{n+1}^\top x$ on P_{n+1}). If we use the shadow-vertex algorithm starting at the level $\{x^{n+1} = 0\}$-optimum, then we walk on a *co-optimal path* all the time. And there will be two possible outcomes:

1) $\max\{x^{n+1} : x \in P_{n+1}\} < 1$.
   Then level $\{x^{n+1} = 1\}$ has no feasible points and P_n is proven to be empty, respectively infeasible.
2) $\max\{x^{n+1} : x \in P_{n+1}\} \ge 1$.
   Then the shadow-vertex path in stage $n + 1$ will traverse the desired level. We calculate the intersection point, drop the last coordinate 1 and have the optimal point for $\max \mu^\top x$ subject to $Ax \le b$. This results from the co-optimality of our path.

Now the following probabilistic result is obvious:

Theorem 11 ([7]) *If* $\left(\begin{matrix} a_1 \\ \widetilde{b^1} \end{matrix}\right), \ldots, \left(\begin{matrix} a_m \\ \widetilde{b^m} \end{matrix}\right)$ *are distributed on* $\mathbf{R}^{n+1}$ *according to the RSM, then general problems of type (1) can be solved for every (m, n) with an expected total number of pivot steps as*

$$E_{m,n}(s^t) \le m^{\frac{1}{n}} (n+1)^3 \cdot C .$$

But this condition has a quite artificial flavor, because the RSM-distribution of the augmented vectors may lead to dependencies between the gradients a_i and the capacities b^i.

We know of one special distribution where both wishes (RSM-distribution and independency) can be combined, namely the Gaussian distribution on $\mathbf{R}^{n+1}$. This is the only RSM-distribution where the components of the generated vectors are independent. We obtain:

Theorem 12 ([7]) *If the vectors*

$$\left(\begin{matrix} a_1 \\ \widetilde{b^i} \end{matrix}\right), \ldots, \left(\begin{matrix} a_m \\ \widetilde{b^m} \end{matrix}\right)$$

are independent and Gaussian distributed, then

$$E_{m,n}(s^t) \le m^{\frac{1}{n}} (n+1)^3 \cdot \text{const}.$$

For more general independent distributions of the right sides (the capacities), as for uniform distribution, we could not derive satisfactory bounds so far (1998). However, this seems to be caused by technical difficulties only. The special results in Theorems 11 and 12 indicate that general problems with arbitrary independent capacity distribution may be solvable on the average with the same effort.

We conclude our report with a look on general variants. In [15] and [12] P. Huhn proved a *lower bound* on the average number of pivot steps, which is valid for all variants. Assume that Phase I has provided us with

a vertex x_0 of X, and that the objective $v^\intercal x$ had no impact on Phase I. Then we start at x_0 with Phase II and try to reach the optimal vertex x_{opt}. To bridge the distance, every variant has to use edges of the polyhedron X. Now *stochastic geometry* can provide information on the distribution of the length of these edges. If one can show that there are extremely few 'long' edges, then a large number of 'small' edges has to be used for our walk. This has been done in [15] and [12] and it gave a guarantee that no variant can (on the average) do its job with less than a certain (computable) number of steps. Quantitatively, this reads as follows. We present only the result for a special distribution, the uniform distribution on ω_n (corresponding results have been derived for a large class of distributions).

Theorem 13 [15] *In a typical RSM problem with uniform distribution on ω_n, every variant of the simplex algorithm will, on the average, require a certain number $E^{av}_{m,n}(s)$ of pivot steps, and*

$$E^{av}_{m,n}(s) \geq C \cdot m^{\frac{1}{(n-1)}} \cdot n^0 \quad \text{with } C > 0.$$

Despite the fact that here the n-order is n^0 (compare with n^2 for the shadow-vertex algorithm), this shows that no variant can perform substantially better. This means that there is no algorithm (variant) running essentially faster than the shadow-vertex algorithm, which can exploit the increasing number of options with n, and which avoids the typical order $m^{1/(n-1)}$ in the RSM.

Thus, a posteriori, the results on the shadow-vertex algorithm have proved to be quite representative. It is not the very best variant, but not much worse than the very best.

Note that the lower bound in Theorem 13 is meaningful only when $m \gg n$, because only then it becomes significantly greater than 1, although the inequality is valid for all (m, n). This is different from the results about the variance (Theorem 8) and the speedup for Phase I (Theorem 10), where it is uncertain, whether these results will be valid in moderate dimensions, too. (Perhaps not the technical difficulties are to blame.) It may as well be possible that these results essentially rely on a regularization effect of the polyhedra for large number of points, as we know it from the approximation of a ball from inside by the convex hull of a huge number of random points.

To clarify these questions, remains an important challenge for future research.

See also

- ▶ Criss-Cross Pivoting Rules
- ▶ Least-Index Anticycling Rules
- ▶ Lexicographic Pivoting Rules
- ▶ Linear Programming
- ▶ Pivoting Algorithms for Linear Programming Generating Two Paths
- ▶ Principal Pivoting Methods for Linear Complementarity Problems

References

1. Adler I, Karp R, Shamir R (1986) A family of simplex-variants solving an m × d linear program in expected number of pivot steps depending on d only. Math Oper Res 11:570–590
2. Adler I, Karp R, Shamir R (1987) A simplex-variant solving an m × d linear program in O(min(m^2, d^2)) expected number of pivot steps. J Complexity 3:372–387
3. Adler I, Megiddo N (1985) A simplex-algorithm where the average number of steps is bounded between two quadratic functions of the smaller dimension. J ACM 32:871–895
4. Borgwardt KH (1977) Untersuchungen zur Asymptotik der mittleren Schrittzahl von Simplex-Verfahren in der Linearen Optimierung. Diss, Univ. Kaiserslautern, Kaiserslautern
5. Borgwardt KH (1982) The average number of pivot steps required by the simplex-method is polynomial. Z Oper Res 26:157–177
6. Borgwardt KH (1987) The simplex method, a probabilistic analysis. Springer, Berlin
7. Borgwardt KH (1990) Probabilistic analysis of the simplex-method. Contemp Math 114:21–34
8. Borgwardt KH (1997) Average complexity of a gift-wrapping algorithm for determining the convex hull of randomly given points. Discrete Comput Geom 17:79–109
9. Borgwardt KH (1997) Ideas leading to a better bound on the average number of pivot steps for solving an LP. Oper Res Proc, vol 1997. Springer, Berlin, pp 1–12
10. Borgwardt KH (1999) A sharp upper bound for the expected number of shadow-vertices in LP-polyhedra under orthogonal projection on two-dimensional planes. Math Oper Res 22(4):925–984
11. Borgwardt KH, Damm R, Donig R, Joas G (1993) Empirical studies on the average efficiency of simplex-variants under rotation-symmetry. ORSA J Comput 5:249–260
12. Borgwardt KH, Huhn P (1999) A lower bound on the average number of pivot-steps for solving linear programs

valid for all variants of the simplex-algorithm. Math Meth Oper Res 49:175–210
13. Haimovich M (1983) The simplex-algorithm is very good! – On the expected number of pivot steps and related properties of random linear programs. Report, Columbia Univ
14. Höfner G (1995) Lineare Optimierung mit dem Schatteneckenalgorithmus – Untersuchungen zum mittleren Rechenaufwand und Entartungsverhalten. Diss., Univ. Augsburg, Augsburg
15. Huhn P (1997) Schranken für die durchschnittliche Laufzeit von Simplex-Verfahren und Innere-Punkt-Verfahren. Diss., Univ. Augsburg, Augsburg
16. Küfer K (1992) Asymptotische Varianzanalysen in der stochastischen Polyedertheorie. Diss., Univ. Kaiserslautern, Kaiserslautern
17. Küfer K (1995) On the variance of the number of pivot steps required by the Simplex-Algorithm. Z Oper Res 42:1–24
18. Küfer K (1996) An improved asymptotic analysis of the number of pivot steps required by the Simplex-Algorithm. Z Oper Res 44:147–170
19. Shamir R (1987) The efficiency of the simplex-method. Managem Sci 33:301–334
20. Smale S (1983) On the average speed of the simplex-method of linear programming. Math Program 27:241–262
21. Todd MJ (1986) Polynomial expected behavior of a pivoting algorithm for linear complimentarity and linear programming problems. Math Program 35:173–192

Probabilistic Constrained Linear Programming: Duality Theory

Éva Komáromi
Budapest University Economic Sci.,
Budapest, Hungary

MSC2000: 90C05, 90C15

Article Outline

Keywords
See also
References

Keywords

Probabilistic constrained linear programming; Duality; Linear programming; Optimization

When formulating stochastic programming problems one usually starts from an underlying deterministic problem which, in case of *probabilistic constrained linear programming* (PCLP), is the *linear programming* (LP) problem. In game theoretical formulation, the LP primal-dual pair of problems is the following:

$$\begin{cases} \sup\limits_{y\in Y'} \{yAx\} \to \min, \\ x \in X' = \{x \in \mathbf{R}^n \colon\ Ax \geq b,\ x \geq 0\}, \end{cases}$$

and

$$\begin{cases} \inf\limits_{x\in X'} \{yAx\} \to \max, \\ y \in Y' = \{y \in \mathbf{R}^m \colon\ yA \leq c,\ y \geq 0\}, \end{cases}$$

where A is a given matrix of dimension $m \times n$, b and c are given vectors of dimension m and n, and x and y are decision vectors of dimension n and m, respectively. In order to extend the *LP duality* concept [2] we reformulate the feasibility sets of the two problems by introducing probability constraints as follows:

$$\begin{cases} \sup\limits_{y\in Y} \{yAx\} \to \min, \\ x \in X = \{x \in \mathbf{R}^n \colon\ \mathrm{P}(Ax \geq \beta) \geq p,\ x \geq 0\}, \end{cases} \tag{1}$$

and

$$\begin{cases} \inf\limits_{x\in X} \{yAx\} \to \max, \\ y \in Y = \{y \in \mathbf{R}^m \colon\ \mathrm{P}(yA \leq \tau) \geq q,\ y \geq 0\}. \end{cases} \tag{2}$$

Here β and $-\tau$ are random vectors of dimension m and n with given joint continuous probability distribution functions F and G, p and q are reliability levels in $(0, 1)$; A, x, y are defined as above.

The pair of problems (1) and (2) was introduced in [3]. Each of them is a generalization of problems with probabilistic constraints and linear objective, introduced by A. Charnes and W.W. Cooper [1], by B.L. Miller and H.M. Wagner [6], and by A. Prékopa [7]. The formulation of a dual problem can be of help in elaborating a solution method, in analysing sensitivity of a solution, in assessing the closeness of the objective function value at a solution at hand from the optimal value.

Because of the presence of the bilinear function yAx, the objective functions of both (1) and (2) directly depend on the feasibility set of the other problem. It is desirable, therefore, to find objective functions for Problems (1) and (2) such that either of them could be interpreted in itself. The idea is similar to that applied

in LP where cx and yb can replace the corresponding objective functions. The argumentation leading to such a pair of self-contained problems follows.

Define the following sets:

$$B = \{b \in \mathbf{R}^m:\ F(b) \geq p, b \in \operatorname{supp} F\}\ ,$$
$$X(b) = \{x \in \mathbf{R}^n:\ Ax \geq b,\ x \geq 0\}\ ,$$
for given $b \in \mathbf{R}^m$,
$$C = \{c \in \mathbf{R}^n:\ G(-c) \geq q,\ -c \in \operatorname{supp} G\}\ ,$$
$$Y(c) = \{y \in \mathbf{R}^m:\ yA \leq c,\ y \geq 0\}\ ,$$
for given $c \in \mathbf{R}^n$.

The sets $X(b)$ and $Y(c)$ are *convex polyhedral sets*. The sets B and C are closed because F and G are assumed to be continuous, and their interiors are not empty because $p, q < 1$, B (respectively, C) is convex if F (respectively, G) is quasiconcave. The set B is bounded if the *support set* supp F (the smallest closed subset of $\mathbf{R}^m$ whose probability measure generated by F is 1) is bounded. Similarly, the set C is bounded if supp G is bounded. Assume, in the course of the transformation below, that B and C are bounded. Obviously,

$$X = \{x \in \mathbf{R}^n:\ x \in X(b),\ b \in B\}\ ,$$
$$Y = \{y \in \mathbf{R}^m:\ y \in Y(c),\ c \in C\}\ ,$$

so (1) and (2) can be rewritten in the following equivalent form:

$$\min_{b\in B}\left\{\min_{x\in X(b)}\left\{\sup_{c\in C}\left\{\sup_{y\in Y(c)} yAx\right\}\right\}\right\}\ , \tag{3}$$

and

$$\max_{c\in C}\left\{\max_{y\in Y(c)}\left\{\inf_{b\in B}\left\{\inf_{x\in X(b)} yAx\right\}\right\}\right\}\ . \tag{4}$$

Observe that

$$\sup_{y\in Y(c)} yAx = \inf_{x'\in X(Ax)}\{cx'\}\ ,$$

for any given $c \in \mathbf{R}^n$ and $x \in \mathbf{R}^n$, with $\inf cx' = -\infty$ if $Y(c) = \emptyset$, by the LP duality theorem. It means that

$$\sup_{c\in C}\left\{\sup_{y\in Y(c)} yAx\right\} = \sup_{c\in C}\left\{\inf_{x'\in X(Ax)}\{cx'\}\right\}\ ,$$

and both sides are defined to be $-\infty$ if $Y(c) = \emptyset$ for all $c \in C$. The function cx' is convex in c, concave in x', and continuous on the product set $C \otimes X(Ax)$ provided C is convex. Assume that C is convex. Then the *saddle value* of cx' exists with respect to minimizing over $X(Ax)$ and maximizing over C [9]:

$$\sup_{c\in C}\left\{\inf_{x'\in X(Ax)} cx'\right\} = \inf_{x'\in X(Ax)}\left\{\sup_{c\in C} cx'\right\}\ .$$

In fact, cx' has a *saddle point* if $\{c \in C : Y(c) \neq \emptyset\}$ satisfies the *Slater condition*: its interior is nonempty. The reason is that $\inf_{x' \in X(Ax)} cx'$ as a function of c is closed, convex hence continuous over its domain $\{c : Y(c) \neq \emptyset\}$ [10]. Therefore, it attains its maximum over the compact set $\{c \in C : Y(c) \neq \emptyset\}$, say at c°. Then, by the LP duality theorem, $\inf_{x' \in X(Ax)} c^\circ x'$ is attained, say at x°. It implies that, in the presence of the Slater condition, (c°, x°) is a saddle point, with respect to minimizing over $X(Ax)$ and maximizing over C [5]. This fact is needed to ensure that not all the optimal solutions of (1) are lost during the transformation.

Relax the minimization of $\sup_C \sup_{Y(c)} yAx$ over $X(b)$ in (3) and use infimum instead. Observe that $X(b) = \{x' \geq 0 : \exists x \in X(b)$ such that $Ax' \geq Ax\}$ for any fixed $b \in \mathbf{R}^m$, so that

$$\inf_{x\in X(b)}\left\{\inf_{x'\in X(Ax)}\left\{\sup_{c\in C} cx'\right\}\right\} = \inf_{x\in X(b)}\left\{\sup_{c\in C} cx\right\}\ ,$$

where both sides are defined to be $+\infty$ if $X(b) = \emptyset$. Then restore the minimization over $X(b)$ and obtain the following problem which corresponds to (1):

$$\begin{cases}\sup\limits_{c\in C} cx \to \min, \\ x \in X = \left\{x:\ \begin{array}{c} Ax \geq b,\ x \geq 0, \\ F(b) \geq p,\ b \in \operatorname{supp} F \end{array}\right\}\ .\end{cases} \tag{5}$$

By assuming that B is convex, a similar argumentation leads to the following problem which corresponds to (2):

$$\begin{cases}\inf\limits_{b\in B} yb \to \max, \\ y \in Y = \left\{y:\ \begin{array}{c} yA \leq c,\ y \geq 0, \\ G(-c) \geq q,\ -c \in \operatorname{supp} G \end{array}\right\}\ .\end{cases} \tag{6}$$

Next, we summarize the relation between (1) and (5), and (2) and (6), respectively.

If the probability distribution function G is quasiconcave, supp$G$ is bounded, (b°, x°) is optimal for (5),

then x° is optimal for (1). If the probability distribution function G is quasiconcave, supp G is bounded, the set $\{c \in C : Y(c) \neq \emptyset\}$ fulfills the Slater condition, x° is optimal for (1), then (b°, x°) is optimal for (5), where $b^\circ = Ax^\circ$.

If the probability distribution function F is quasiconcave, supp$F$ is bounded, (c°, y°) is optimal for (5), then y° is optimal for (2). If the probability distribution function F is quasiconcave, supp F is bounded, $\{b \in B : X(b) \neq \emptyset\}$ fulfills the Slater condition, y° is optimal for (2), then (c°, y°) is optimal for (6), where $c^\circ = y^\circ A$.

It remains to state the duality theorem, the proof of which can be found in [4].

Theorem 1 *(Duality theorem) Suppose that the probability distribution functions F and G are quasiconcave, that F is a strictly increasing function of its components, that supp G is bounded, and that the Slater condition int $\{b \in B : X(b) \neq \emptyset\} \neq \emptyset$ holds. If (5) is unbounded in value, then (6) is inconsistent. Otherwise, (6) is consistent, their values are equal, and that value is attained in (6).*

Suppose that the probability distribution functions F and G are quasiconcave, that G is a strictly increasing function of its components, that suppF is bounded, and that the Slater condition int $\{c \in C : Y(c) \neq \emptyset\} \neq \emptyset$ holds. If (6) is unbounded in value, then (5) is inconsistent. Otherwise, (5) is consistent, their values are equal, and that value is attained in (5).

Suppose that the probability distribution functions F and G are quasiconcave, that each of them is a strictly increasing function of its components, that supp F and supp G are bounded, and that the following regularity conditions hold: int $\{b \in B : X(b) \neq \emptyset\} \neq \emptyset$ and int $c \in C : Y(c) \neq \emptyset\} \neq \emptyset$. Then (1) has an optimal solution x°, and (2) has an optimal solution y° such that (x°, y°) is a saddle point of yAx with respect to minimizing over X and maximizing over Y.

The rich class of *quasiconcave probability distribution* functions includes, among others, the multidimensional normal, exponential, uniform, gamma distributions [8]. In practical problems, the support set of the probability distribution in question is usually bounded. Although the approximating theoretical distribution often has an unbounded support, reasonable truncation can be applied.

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-Stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems

► Two-Stage Stochastic Programming: Quasigradient Method
► Two-Stage Stochastic Programs with Recourse

References

1. Charnes A, Cooper WW (1959) Chance-constrained programming. Managem Sci 6:73–89
2. Dantzig GB (1963) Linear programming and extensions. Princeton Univ. Press, Princeton
3. Komáromi É (1986) Duality in probabilistic constrained programming. In: Prékopa A (ed) Proc. 12th IFIP Conf. System Modelling and Optimization. Springer, Berlin, pp 423–429
4. Komáromi É (1992) Probabilistic constraints in primal and dual linear programs: Duality results. J Optim Th Appl 75:587–602
5. Mangasarian OL (1969) Nonlinear programming. McGraw-Hill, New York
6. Miller BL, Wagner HM (1965) Chance-constrained programming with joint constraints. Oper Res 13:930–945
7. Prékopa A (1970) On probabilistic constrained programming. In: Kuhn H (ed) Proc. Princeton Symp. Math. Program., Princeton, pp 113–138
8. Prékopa A (1971) Logarithmic concave measures with application to stochastic programming. Acta Sci Math 32:301–316
9. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton
10. Williams AC (1963) Marginal values in linear programming. SIAM J Appl Math 11:82–94

Probabilistic Constrained Problems: Convexity Theory

András Prékopa
RUTCOR, Rutgers Center for Operations Research, Piscataway, USA

MSC2000: 90C15

Article Outline

Keywords
See also
References

Keywords

Probabilistic constraint; Logconcave probability density function; Logconcave function; Multivariate normal distribution

The general form of a probabilistic constraint is the following:

$$\mathsf{P}\left(g_1(\mathbf{x},\boldsymbol{\xi}) \geq 0,\ \ldots\ , g_r(\mathbf{x},\boldsymbol{\xi}) \geq 0\right) \geq p\ ,$$

where $g_r(\mathbf{x},\mathbf{y})$, $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{y} \in \mathbf{R}^q$, $i = 1, \ldots, r$, are some functions, $\boldsymbol{\xi}$ is a q-variate random vector and p is a fixed probability. In the simplest case $g_r(\mathbf{x},\mathbf{y}) = T_i\mathbf{x} - y_i$, where T_i is the ith row of a matrix T, $i = 1, \ldots, r$. In this case the above constraint take the form: $\mathsf{P}(T\mathbf{x} \geq \boldsymbol{\xi}) \geq p$.

The most important theorem in connection with the probabilistic constraint states that if $g_1, \ldots, g_r$ are concave, or quasiconcave functions in $\mathbf{R}^{n+q}$ and $\boldsymbol{\xi}$ has a continuous probability distribution with logconcave probability distribution function, then $\mathsf{P}\left(g_i(\mathbf{x},\boldsymbol{\xi}) \geq 0,\ i = 1, \ldots, r\right)$ is a logconcave function in $\mathbf{R}^n$ ([4,5]).

There are, however, important cases where the functions $g_i(\mathbf{x},\mathbf{y})$ are not concave or quasiconcave. S. Kataoka [2] and C. van de Panne and W. Popp [3] considered the probabilistic constraint of the form:

$$\mathsf{P}(\xi_1 x_1 + \cdots + \xi_n x_n \geq b) \geq p\ ,$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)^\mathsf{T}$ has a multivariate normal distribution and b is constant. The practical problems were investment and animal feed problems, respectively. If $\boldsymbol{\mu} = \mathsf{E}(\boldsymbol{\xi})$, $C = \mathsf{E}\left[(\boldsymbol{\xi} - \boldsymbol{\mu})(\boldsymbol{\xi} - \boldsymbol{\mu})^\top\right]$, then the above probabilistic constraint is shown to be equivalent to

$$\boldsymbol{\mu}^\top\mathbf{x} + \Phi^{-1}(1-p)\sqrt{\mathbf{x}^\top C\mathbf{x}} \geq b\ ,$$

where Φ is the univariate standard normal probability distribution function. If $p \geq 1/2$, then $\Phi^{-1}(1-p) \leq 0$ and the set of $\mathbf{x}$ vectors satisfying the probabilistic constraint is convex.

Generalizations of this result have been given in [1,4]. We look at the joint probabilistic constraint:

$$\mathsf{P}(\xi_{i1} x_1 + \cdots + \xi_{in} x_n \geq b_i,\ i = 1, \ldots, r) \geq p\ ,$$

where we assume that the altogether rm random variables ξ_{ij} have a joint normal distribution. The statement is that if the covariance matrices of the rows $(\xi_{i1}, \ldots, \xi_{in})$, $i = 1, \ldots, r$, are constant multiples of a fixed covariance matrix C_1, or the covariance matrices of the columns $(\xi_{1j}, \ldots, \xi_{rj})^\mathsf{T}$, $j = 1, \ldots, n$, are constant multiples of a fixed covariance matrix C_2, then the set of $\mathbf{x}$

vectors satisfying the probabilistic constraint is convex, provided that $p \geq 1/2$. For further convexity statements see [5].

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-Stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-Stage Stochastic Programming: Quasigradient Method
- ▶ Two-Stage Stochastic Programs with Recourse

References

1. Burkauskas A (1986) On the convexity problem of probabilistic constrained stochastic programming problems. Alkalmazott Mat Lapok (Applied Math Papers) 12:77–90.(In Hungarian)
2. Kataoka S (1963) A stochastic programming model. Econometrica 31:181–196
3. Panne Cvan de, Popp W (1963) Minimum cost cattle feed under probabilistic constraint. Managem Sci 9:405–430
4. Prékopa A (1974) Programming under probabilistic constraints with a random technology matrix. Math Operationsforsch Statist Ser Optim 5:109–116
5. Prékopa A (1995) Stochastic programming. Kluwer, Dordrecht

Production-Distribution System Design Problem

ABDULLAH DASCI[1], VEDAT VERTER[2]
[1] School of Administrative Studies, York University, Toronto, Canada
[2] Desautels Faculty of Management, McGill University, Montreal, Canada

MSC2000: 90B06

Article Outline

Introduction
Prevailing Models
PDSDP in Practice
Concluding Remarks
Acknowledgement
References

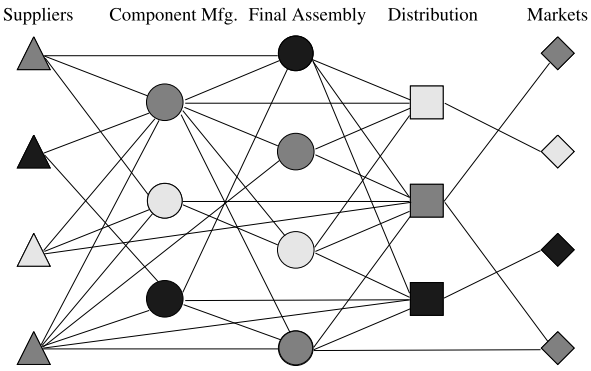
Introduction

Management of a firm's production-distribution system involves tactical and operational decisions along with strategic ones. The policies that enable a firm to

meet its strategic goals comprise a collection of structural and infrastructural decisions, which typically involve long-term commitments with regards to the configuration and coordination of its supply chain. These structural decisions, a.k.a., facility design decisions include location, capacity, product range, and production or operating technologies at various facilities. The usual objective is minimizing costs or maximizing gross profits while leaving other manufacturing tasks (e. g., goals set for quality, flexibility, etc.) for prior or subsequent analysis. Mathematical models of this nature are generally called as "production-distribution system design problems" (PDSDP).

Traditionally, location, capacity, technology, and product range decisions have been dealt with separately. Facility location models, in general, ignore the geographical differences in capacity and technology acquisition/operation costs. Whereas, capacity expansion models mostly deal with the temporal aspects of market demand and do not incorporate location decisions with regards to the establishment of new plants. The interactions between facility design decisions, however, can be significant and PDSDP research takes its motivation from these interactions. The governments of many countries provide subsidies to support economic activities of specific sectors or regions with high rate of unemployment. In taking advantage of the capital and/or employment subsidies, preferential tax rates, and free trade zones provided by governments, firms, especially multinationals need to take interdependencies between their location, capacity and technology decisions into account. These decisions could further be complicated due to varying scale and scope economies inherent in different technologies.

A production-distribution network provides an effective representation of the manufacturing and logistics activities of a firm and assists researchers with a framework to study various systems. In a typical network, nodes represent suppliers, manufacturing facilities, distribution centers, warehouses and customers of the firm. The arcs on the network delineate the flow of items between nodes. An example network of five echelons is depicted in Fig. 1. There are mainly two important feature that define the difficulty of PSDSP: first, is the number of echelons in the system and second, is the number of different types of configurational, most notably locational, decisions that need to be made.



Production-Distribution System Design Problem, Figure 1
A production-distribution network

Our objective is to provide an overview of the prevailing methodology for designing production-distribution systems. The remainder of this article is organized as follows: In the next section we give an overview and taxonomy of the prevailing PDSDP models. Subsequently, an overview of PDSDP in practice is given. The paper concludes with our remarks.

Prevailing Models

Location decisions have been attracting researchers since the end of nineteenth century. But a rigorous methodology for production-distribution systems did not come out until sixties during which two competing approaches have appeared. For a warehouse location problem, Shycon and Maffei [71] propose a simulation model claiming that, a proper model should be descriptive, whereas Kuehn and Hamburger [46] favor prescriptive models and propose a mixed integer linear program (MILP) and a heuristic solution procedure. After these two pioneering works, MILP formulations received considerably more attention in both practice and theory.

It is evident that PDSDP refers to a family of problems. The purpose here is to provide an overview of production-distribution system design literature, rather than an exhaustive review. The uncapacitated facility location problem (UFLP) is the simplest type of PDSDP, with a single-commodity, a network of two echelons (i. e., facilities and customers) of which only a single echelon of nodes (i. e. facilities) is to be located. The following is the most popular formulation known

in the literature:

$$\min \sum_{i\in I} f_i y_i + \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{i\in I} x_{ij} = 1, \quad j \in J, \tag{2}$$

$$x_{ij} \le y_i, \quad i \in I,\ j \in J, \tag{3}$$

$$x_{ij} \ge 0, \quad i \in I,\ j \in J, \tag{4}$$

$$y_i \in \{0,1\}, \quad i \in I \tag{5}$$

x_{ij} : proportion of customer j's demand satisfied by facility i,
y_i : 1 if facility i is opened, 0 otherwise,
c_{ij} : the total production and distribution costs for supplying all of the customer j's demand from facility i,
f_i : fixed cost of opening facility i,
I, J : the set of candidate facility sites and customers.

The objective is to minimize the sum of the fixed setup costs of opening facilities and the variable costs of serving the customers. Constraints (1) guarantee that each customer's demand is satisfied, and (2) ensure that only open plants can make shipments.

Beginning with Kuehn and Hamburger, most research has been directed towards devising more efficient solution procedures to UFLP. Efroymson and Ray [19] propose an LP based branch-and-bound algorithm. However, in order to solve the arising LPs more efficiently and to minimize the memory requirements, they devised a more compact but weaker formulation of UFLP by replacing constraints (2) with the following equivalent set of constraints:

$$\sum_{j\in J} x_{ij} \le n_i y_i, \quad i \in I, \tag{6}$$

where n_i is the number of customers that facility i can serve. Later, Spielberg [75] proposed another branch-and-bound (implicit enumeration) algorithm using the same weaker formulation. The largest stride towards an efficient solution procedure came in late seventies. Bilde and Krarup [8] and Erlenkotter [22], independent from each other, took advantage of the tighter formulation and proposed one of the most remarkable solution procedures. Instead of solving LPs at nodes to optimally, they devised quick procedures to obtain good solutions to the dual of LPs. Using these "good" dual solutions, they generated integer feasible primal solutions using complementary slackness conditions and a heuristic. Erlenkotter reports that most of the time the optimal solution is found after single pass. If there is a gap left, it is eliminated by branch-and-bound. After these works, this type of procedure (called dual-based branch-and-bound) has been repeatedly applied to other location problems.

The normative work on UFLP has been extended in several ways so as to analyze more realistic production-distribution systems. One of the earliest attempts was to incorporate the limited availability of land and other production factors at the alternative sites. The formulation of the capacitated version is quite similar to UFLP, where capacity constraints are appended to the formulation. Using identical formulations, Akinc and Khumawala [2] and Nauss [55] propose linear programming and Lagrangean relaxation based branch-and-bound solution techniques, respectively. On the other hand, Geoffrion and McBride [28] solve a generalization of the capacitated model in which there are lower as well as upper bounds and arbitrary constraints over structural variables (y_i).

Multicommodity models have also been widely studied in the literature(e. g. Warszawski [88], Neebe and Khumawala [56], and Karkazis and Boffey [41]). The problem is an immediate generalization of UFLP where there are multiple products. Even though they offered different formulations, all of them assumed that only one product can be assigned to a facility. This assumption is later relaxed in Klincewicz and Luss [43]. Quite a few researchers have focused on increasing the number of echelons to be located. This line of research constitutes a major stride towards the development of analytical models that are capable of representing PDSDPs of the size managers typically encounter in real life. Kaufman, Eede, and Hansen [42] and Tcha and Lee [76] are among the earlier works. The former deals with location of plants and warehouses to minimize that total cost of respective fixed costs and production and distribution costs. Tcha and Lee however deals with arbitrary number of echelons. The commonality between both models is that they ignore any cost that might arise from the interaction be-

tween plants and warehouses. Later, Gao and Robinson [26] propose an efficient dual-based branch-and-bound algorithm and Barros and Labbe [6] present a profit maximization version of the same problem. They proposed various heuristics and an exact Lagrangian relaxation-based branch-and-bound algorithm.

Thus far, works on UFLP and some of its immediate generalizations are reviewed. Arguably one of the most influential model in PDSDP literature is proposed and solved by Geoffrion and Graves [27]. Their model consists of three echelons; plants, warehouses, and customers. The objective is to minimize the total costs of production, transportation, warehousing and total fixed costs of opening the warehouses. The capacity restrictions over DCs can be both from above or below, which enables modeling piecewise linear concave DC throughput costs. Furthermore, each customer is to be supplied from one DC and there may be certain restrictions on the configurational decisions. They proposed a solution procedure based on Benders' decomposition.

The most important contribution of Geoffrion and Graves [27] is the paradigm change triggered by this paper. In multi-echelon models, the usual practice was to represent flow between neighbor echelons by different sets of variables and impose flow conservation at the nodes. Geoffrion and Graves [27] represent the flows on the network by a single set of variables from nodes in the first echelon to the nodes in the last (i. e. using more indices). With this modeling technique the number of variables grow considerably with the size but the formulation becomes tighter, which could be useful in devising more efficient algorithms.

Later, Moon [54] extended the model and solution procedure considering economies of scale in DC throughput costs. In this case the DC throughput costs are represented by general concave cost functions. Pirkul and Jayaraman [59] propose a similar model that differs in the following aspects:

i) The opening decisions are not only for warehouses but also for plants,

ii) The capacity limitations are only from above, and

iii) There are upper bounds on the number of plants and warehouses that can be opened. They devise a Lagrangean relaxation based heuristic solution procedure. In a subsequent paper [60], they also include supplier selection. Elhedli and Goffin [20] present one of the most recent advances in this area.

In designing an production-distribution system, it is crucial to optimize the configurational decisions simultaneously because a sequential approach is bound to produce suboptimal results especially when the interactions between these decisions and scope and scale economies are present. One of the earliest attempts to develop a model that consider scale economies is due to Soland [73]. His model is a simple extension of UFLP where fixed facility costs are replaced by a concave function of the size of the facility. Later, Holmberg [37] and Holmberg and Ling [38] propose a capacitated facility location problem where capacity acquisition cost is an arbitrary piecewise linear function. Verter and Dincer [85] propose alternative model where capacity costs are assumed to be general concave functions of total acquisition. More recently, Dasci and Verter [17] and Verter and Dasci [84] extend these models to a multi-product environment and selection of various dedicated and flexible technologies that display different forms of scale and scope economies. A number of authors propose models that integrate inventory control and logistics decisions into a PDSDP framework. The inventory related costs also display scale economies and therefore, necessitate a concave or nonlinear cost modeling, as in the aforementioned works. Shen [69] present a unifying work on this issue. Sneyder et al. [77] and Sourirajan et al. [74] present more recent location models that consider logistics related costs.

Since the mid-eighties, the world witnessed the emergence of global manufacturing firms, which diversify their operations to different countries. Globalization provides the firm with many advantages, such as access to cheap labor, raw material and other production factors, presence at regional markets, and access to locally available technological resources and know-how. The arising supply chain structures, however, are usually more challenging from the perspective of the manager. The strategic management of international production-distribution networks are further complicated due to the price and exchange rate uncertainties. In a series of papers, Hodder and Jucker [35,36] and Hodder and Dincer [34] propose scenario based approaches to model these uncertainties. Their models maximize expected profit less a constant portion of the

variability of the profit, which is an appropriate way to represent a risk-averse decision-maker. Later, Gutierrez and Kouvelis [31] also used a scenario-based approach to find robust solutions under all possible scenario realizations. More recently, Canel and Khumawala [11] and Kouvelis et al. [45] proposed models that include subsidies and tariffs. The prevailing studies in this field suggest that the scenario-based approach is a popular way of modeling the various types of uncertainties that characterize the international environment. The number of possible scenarios however quickly proliferate as the problem size increases, making not only the evaluation of the scenarios but also the generation of them a formidable task.

As we have seen, there are families of PDSDPs involving capacity limitation on sites, multiple products, multiple echelons, or time phased system design models. Almost all of these models are formulated as mixed integer programs and are solved usually by branch and bound. In general, Lagrangean relaxation, linear programming (LP) relaxation, and dual-based procedures (see for example, Erlenkotter [22] are the most common lower bounding techniques. While dual-based procedures are usually shown to be computationally more efficient than the former two, it is very much dependent on the special structures of the problems. On the other hand, Lagrangean and LP relaxation techniques can essentially be used for any formulation. Starting with Geoffrion and Graves [27], Bender's decomposition, which is able to handle arbitrary side constraints on structural variables, has been used few times. Nevertheless, it is among the less popular solution methods like dynamic programming. From the perspective of heuristic solution techniques, Lagrangean relaxation based and pairwise improvement type procedures are the most commonly used techniques.

While the match between a formulation and a solution procedure is certainly the defining factor in the efficiency of a solution method, tighter formulations usually lead to more efficient solutions procedures. For example in UFLP, while constraints (2) and (6) are interchangeable, the former provides a tighter formulation, which eventually led to very efficient dual-based solution procedures. Similarly, as pioneered by Geoffrion and Graves [27], defining the flows on a network by a single set of variables from nodes in the first echelon to the nodes in the last. While this formulation causes the number of decision variables to propagate, it leads to tighter formulation.

While we try to give an overview of the prevailing PDSDP models and solution methodologies, space prohibits us from mentioning tens if not hundreds other works. Therefore, we have devised a classification of the prevailing analytical models along differentiating PDSDP features in Table 1. This table gives a quick snapshot of PDSDP literature in the last 40 years. This classification is intended to reveal the strengths and weakness of the existing methodology as well as the major trends in the literature. Analytical approaches have been such a focus of research that even the table is far from exhaustive. Therefore, we refer the interested reader to the following review and critique papers: Aikens [1], Verter and Dincer [86], Vidal and Goetschalckx [87], Goetschalckx et al. [30], Klose and Drexl [44], Meixell and Gargeya [51]), Snyder [72], Sahin and Sural [64], and Shen [70].

PDSDP in Practice

In this section, we provide an overview PDSDP application in practice. In their seminal paper, Geoffrion and Graves [27], have not only provided one of the most influential models on PDSDP, but also reported arguably the first industrial implementation. They assisted Hunt-Wesson Foods, Inc in re-locating their distribution centers (DCs). The firm had been producing several hundred commodities at 14 sites and distributing through a dozen DCs. They recommended five changes in the configuration of DCs as well as the establishment of a new DC.

After this work PDSDP implementations have started to appear in the literature. Arguably, the most comprehensive implementation is presented by Arntzen et al. [4]. This paper reports on the development and use of Global Supply Chain Management (GSCM), a large scale optimization model that contributed to the re-organization of the Digital Equipment Corporation (DEC) in the late 80s and early 90s. Although its development had started as a small project, GSCM has become an essential tool to examine all aspects of supply chain management in DEC. GSCM is arguably the most comprehensive system that considers production, inventory, material handling, taxes, fa-

Production-Distribution System Design Problem, Table 1
A taxonomy of analytical approaches for production-distribution system design problems

	Objective function (C/P/M)[1]	Number of echelon in PDS	Number of structural decisions	Number of commodities (S/M)[2]	Capacity limitation on sites (+/—)	Demand (D/S)[3]	Number time periods (S/M)[2]	Side Constraints (+/—)	Capacity/technology acquisition (+/—)	Nonlinear optimization (+/—)
Kuehn and Hamburger [46]	C	3	1	M	+	D	S	—	—	—
Efroymson and Ray [19]	C	2	1	S	—	D	S	—	—	—
Spielberg [75]	C	2	1	S	—	D	S	—	—	—
Elson [21]	C	3	1	M	+	D	M	+	+	—
Warszawski [88]	C	2	1	M	—	D	S	—	—	—
Geoffrion and Graves [27]	C	3	1	M	+	D	S	+	—	+
Wesolowski and Truscott [89]	C	2	1	S	—	D	M	—	—	—
Balachandran and Jain [5]	C	2	1	S	+	S	S	—	—	—
Akinc and Khumawala [2]	C	2	1	S	+	D	S	—	—	—
LeBlanc [47]	C	2	1	S	+	S	S	—	—	—
Kaufman et al. [42]	C	3	2	S	—	D	S	—	—	—
Erlenkotter [22]	C	2	1	S	—	D	S	—	—	—
Geofrrion and McBride [28]	C	2	1	S	+	D	S	—	—	+
Nauss [55]	C	2	1	S	+	D	S	—	—	—
Karkazis and Boffey [41]	C	2	1	M	—	D	S	—	—	—
Neebe and Khumawala [56]	C	2	1	M	—	D	S	—	—	—
Van Roy and Gelders [83]	C	3	1	S	+	D	S	+	—	+
Van Roy and Erlenkotter [82]	C	2	1	S	—	D	M	—	—	—
Tcha and Lee [76]	C	M	M	S	—	D	S	—	—	—
Hodder and Jucker [36]	P	2	1	S	—	D	S	—	—	—
Hodder and Jucker [35]	P	2	1	S	—	D	S	—	—	—
Hodder and Dincer [34]	P	2	1	S	+	D	S	+	—	—
Klincewicz and Luss [43]	C	2	1	M	—	D	S	—	—	—
Moon [54]	C	3	1	M	+	D	S	+	+	+
Gao and Robinson [26]	C	3	2	S	—	D	S	—	—	—
Holmberg [35]	C	2	1	S	—	D	S	+	+	+
Robinson et al. [63]	C	3	2	S	—	D	S	—	—	—
Barros and Labbe [6]	P	3	2	S	—	D	S	—	—	—
Gutierrez and Kouvelis [31]	C	2	1	S	—	D	S	—	—	—
Verter and Dincer [86]	C	2	1	S	+/—	D	S	—	+	+
Pirkul and Jayaraman [59]	C	3	2	M	+	D	S	+	—	—
Holmberg and Ling [38]	C	2	1	S	+	D	S	—	+	+
Pirkul and Jayaraman [60]	C	4	2	M	+	D	S	+	—	—
Lim and Kim [49]	C	2	2	S	—	D	M	+	+	—
Dogan and Goetschalckx [18]	C	4	1	M	+	D	M	+	—	—
Melachrinoudis and Min [52]	M	2	1	S	+	D	M	+	—	—
Hinojosa et al. [33]	C	3	2	M	+	D	M	—	—	—
Dasci and Verter [17]	C	2	2	M	—	D	S	—	+	+
Tsiakis et al. [78]	C	4	2	M	+	D/S	S	+	+	+

Production-Distribution System Design Problem, Table 1 (continued)

	Objective function (C/P/M)[1]	Number of echelon in PDS	Number of structural decisions	Number of commodities (S/M)[2]	Capacity limitation on sites (+/−)	Demand (D/S)[3]	Number time periods (S/M)[2]	Side Constraints (+/−)	Capacity/technology acquisition (+/−)	Nonlinear optimization (+/−)
Jayaraman and Pirkul [40]	C	3	2	M	+	D	S	+	−	−
Canel and Khumawala [11]	C	2	1	S	−	D	M	−	−	−
Canel et al. [12]	C	2	2	M	−	D	M	+	−	−
Verter and Dasci [84]	C	2	2	M	−	D	S	−	+	+
Jang et al. [39]	C	3	2	M	+	D	S	+	−	−
Kouvelis et al. [36]	P	3	2	M	−	D	M	+	+	−
Paquet et al. [57]	C	M	2	M	−	D	S	+	+	+
Rantala [62]	C	4	2	M	+	D	S	+	−	+
Shen [52]	C	2	1	M	−	S	S	−	−	+
Elhedhli and Goffin [14]	C	3	1	M	−	D	S	+	−	−
Park [58]	P	2	1	M	+	D	S	+	−	−
Melo et al. [53]	C	M	M	M	−	D	M	+	+	−
Santoso et al. [66]	C	M	M	S	+	D	S	+	−	−
Amiri [3]	C	3	2	S	+	D	S	−	+	−
Cordeau et al. [14]	C	4	3	M	+	D	S	+	+	−
Sneyder et al. [60]	C	2	1	S	−	S	S	−	−	+
Sourirajan et al. [57]	C	2	1	S	+	S	S	+	−	+

cility fixed charges, production line fixed charges, transportation costs, duty costs, duty drawbacks and duty avoidance. In an effort to establish the optimal supply chain structure, GSCM reduced the number of plants from 31 to 12, which enabled the major customer regions (America, Europe, and Pacific Rim) to become self-contained. Estimated savings as of 1995 were $1.2 billion. GSCM was also used to determine the optimal network structure for distribution of spare parts and collection of defective items. Repair center facility locations and capacities were determined. Total savings of this project were estimated to be $200 million.

A set of published industrial applications are summarized in Table 2. While such works appear at a steady pace, the literature on the applications of PDSDP is quite sparse. There seem to be a few possible explanations: First, (re)-configuration of a firm's supply chain is a long-term process, which requires a genuine commitment by top management. In many cases, the possibility of success in adopting an analytical approach for (re)-designing the supply chain is severely diminished due to the lack of support from top management. Second, we are aware of many firms who choose not to report their experiences in production-distribution system design simply due to the strategic nature of these decisions. Finally, the level of many industrial projects might not justify publication in academic journals.

Most of the models developed in these projects are different from each other and the past models appeared in the literature. This might be taken as a negative sign on the applicability of generic models. However, there are several commercial software packages that have built-in functions and off-the-shelf generalized programs. The companies developing these software packages report hundreds of firms in their client roster.

One such software package is called, Strategic Logistics Integrative Modeling System (SLIM). It is an

Production-Distribution System Design Problem, Table 2
A Sample of Published Industrial Applications

Published Paper	Company	Industry	Region	Facilities	Estimated Savings
Geoffrion and Graves [27]	Hunt-Wesson	Foods	USA	14 plants, 12 DCs[2]	$1–5 million/year 9–25% of dist. cost
Van Roy and Gelders [83]	N.V. ESSO	LPG	Belgium	1 plant, 7 depots	N/A
Breitman and Lucas [10]	GM	Auto	Multinational		$1 billion total
Cohen and Lee [13]	Apple	Computer	Multinational		N/A
Martin et al. [50]	Libbey-Owens-Ford	Glass	USA	4 plants	$2 million/year
Robinson et al. [63]	Dow Consumer Products	Home- and Food-care	USA	13 CDCs[3], 23 RDCs[4]	$1.5 million/year
Pooley [61]	Ault Foods Limited	Dairy	Canada	4 plants, 12 depots	$3 million/year
Arntzen et al. [4]	DEC	Computer	Multinational	33 plants	$1.5 billion total
Feigin et al. [23]	IBM Europe	Computer	Europe	1 plant, 13 DCs	$40 million/year
Sankaran and Raghavan [65]	S. Shakti LPG Limited	LPG	India	2 ports, 5 plants	$1 million/yearN/A
Koksalan and Sural [64]	Efes Group	Brewery	Turkey	6 plants	
Sery et al. [67]	BASF Corporation	Chemical	North America	67 plants, 86 DCs	6% annual
Wouda et al. [90]	Nutricia Dairy and Drinks	Foods	Hungary	9 plants, 17 DCs	N/A
Tyagi et al. [79]	GE Plastics	Chemical	Multinational	29 plants	N/A
LeBlanc et al. [48]	Nu-kote	Imaging	Multinational	5 plants, 4 warehouses	$1 million/year
Ulstein et al. [80]	Elkem	Metallurgy	Multinational	9 plants	N/A
Fleischmann et al. [24]	BMW	Auto	Multinational	19 plants	N/A

DC: Distribution Center, CDC: Central Distribution Center, RDC: Regional Distribution Center

advanced decision support system that is used to evaluate strategic production and distribution planning problems. SLIM is developed by Prof. B. Shapiro of MIT and his associates, who have collaborated with over 100 companies in the US, Canada, Europe, and Australia [68]. Another software package is called Strategic Analysis of Integrated Logistics System (SAILS), which was developed by A. Geoffrion and his colleagues at the Insight Inc. In this package, extensive data management functions and optimization tools are integrated. Geoffrion and Powers [29] report that there have been 50 or more studies done by SAILS or one of its earlier versions. The projects result with 5–15% reduction in total transportation costs.

Generic models have the advantage of having a shorter development time and lower costs whereas custom-made models have the advantage of ability to incorporate various firm specific practices. Either generic or custom-made, above studies have shown that there are considerable savings possible from (re)designing production-distribution systems through an analytical model.

Concluding Remarks

At the time, the first edition of the encyclopedia has appeared, we stressed that the development of analytical models that could effectively represent supply chains of the sizes that are typically encountered in real life was crucial. Recent advances in PDSDP research and computing enabled researchers to analyze more complex systems and achieve this objective. A quick glance at Table 1 would show that the advances made in the last 10 years are far greater than the advances made earlier. The existing methodology on PDSDP however still needs to be improved. We see two general directions the methodology should be improved: integration of strategic and tactical decisions and including international features in PDSDPs.

It is well known that structural decisions would restrict how the firms make their tactical decisions subsequently. Therefore, integration of this decisions in a PDSDP framework should help firms improve their profits. Although few works have recently appeared in the literature that integrates these decisions [74,77], the literature still has gaps.

While issues such as globalization, outsourcing, international plant location have been around quite some time, the literature is still scant on the issue. Among the 56 works that we have classified in Table 1, only six papers consider international features such as subsidies, tariffs, and exchange rate uncertainty explicitly. There has also been a skepticism as to the impact of certain aspects globalization on firms' configuration decisions. For example, according to one survey, conducted among the plant managers of 73 large multinational firms, subsidies and free trade zones are ranked among the least important factors. However, a number of studies, mostly conducted in the US National Bureau of Economic Research, report that both international and domestic firms have been quite responsive to subsidies, free trade zones, taxes, and labor costs in deciding their plant locations. For example, Head, Ries, and Swenson [32], in their investigation of location patterns of Japanese manufacturing establishments, conclude that these firms have been quite responsive to trade zones. Similarly, Coughlin and Segev [15], in their study of foreign investment in the US, concludes that higher average labor density (as a surrogate measure of average wage rate) and higher taxes in a state are found to deter foreign investment. Finally, both Fuest and Huber [25] and Bergman, Fuss, and Regev [7] provide anecdotal and empirical evidence about how subsidies alter the location choices of domestic firms in the Eastern Germany and Israel respectively.

We can conclude that the interactions between facility design decisions are especially important for international companies. They face not only a multitude of location and technology choices but also a complex array of cost structures due to regional differences as well as governments' tax, subsidy, and free trade policies. These decisions are further complicated by manufacturing strategy options and scale and scope economies. Therefore, it is important that analytical approaches for PDSDP incorporate demand and exchange rate uncertainties as well as the other distinguishing features of the international environment.

Acknowledgement

This research was partially funded by the Natural Sciences and Engineering Research Council of Canada.

References

1. Aikens CH (1985) Facility location models for distribution planning. Eur J Oper Res 22:263–279
2. Akinc U, Khumawala BM (1977) An efficient branch and bound algorithm for the capacitated warehouse location problem. Manag Sci 23:585–594
3. Amiri A (2006) Designing a distribution network in a supply chain system: Formulation and efficient solution procedure. Eur J Oper Res 171:567–576
4. Arntzen BC, Brown GG, Harrison TP, Trafton LL (1995) Global supply chain management at Digital Equipment Corporation. Interfaces 25:69–93
5. Balachandran V, Jain S (1976) Optimal facility location under random demand with general cost structure. Nav Res Logist Q 23:421–436
6. Barros AI, Labbe M (1994) A general model for uncapacitated facility and depot location problem. Locat Sci 2:173–191
7. Bergman A, Fuss M, Regev H (1998) The effects of capital subsidization on Israeli industry. Working Paper FUSS-98-01, University of Toronto, ECIPA, Downloadable at URL: http://www.nber.org/papers/w6788.pdf
8. Bilde O, Krarup J (1977) Sharp lower bound and efficient algorithms for the simple plant location problem. Ann Discret Math 1:79–97
9. Billington CA, Davis TC (1992) Manufacturing strategy analysis: Models and practice. Int OMEGA J Manag Sci 20:587–595
10. Breitman RL, Lucas JM (1987) Planets: A modeling system for business planning. Interfaces 17:94–106
11. Canel C, Khumawala BM (2001) International facilities location: a heuristic procedure for the dynamic uncapacitated problem. Int J Prod Res 17:3975–4000
12. Canel C, Khumawala BM, Law J, Lo A (2001) An algorithm for the capacitated, multi-commodity multi-period facility location problem. Comput Oper Res 28:411–427
13. Cohen MA, Lee HL (1989) Resource deployment analysis of global and manufacturing and distribution networks. J Manuf Oper Manag 2:81–104
14. Cordeau JF, Pasin F, Solomon MM (2006) An integrated model for logistics network design. Annals Oper Res 144:59–82
15. Coughlin CC, Segev E (2000) Location determinants of new foreign-owned manufacturing plants. J Reg Sci 20:232–251
16. Crainic T, Delorme J (1993) Dual-ascent procedures for multicommodity location-allocation problems with balancing requirements. Trans Sci 27:90–101
17. Dasci A, Verter V (2001) The plant location and technology acquisition problem. Interfaces 33:963–973
18. Dogan K Goetschalckx M (1999) A primal decomposition method for the integrated design of multi-period production-distribution systems. IIE Trans 31:1027–1036

19. Efroymson MA, Ray TL (1966) A branch and bound algorithm for plant location. Oper Res 14:361–368
20. Elhedhli S, Goffin J-L (2005) Efficient production-distribution system design. Manag Sci 51:1151–1164
21. Elson DG (1972) Site location via mixed-integer programming. Oper Res Q 23:31–43
22. Erlenkotter D (1978) A dual based procedure for uncapacitated facility location. Oper Res 26:31–43
23. Feigin G, An C, Connors D, Crawford I (1996) Shape up, ship out. OR/MS Today, April:24–30
24. Fleischmann B, Ferber S, Henrich P (2006) Strategic planning of BMWs global production network. Interfaces 36:194–208
25. Fuest C, Huber B (1998) Why do countries subsidize investment and not employment? Working Paper 6685, NBER, Downloadable at URL: http://www.nber.org/papers/w6685.pdf
26. Gao L, Robinson EP (1992) A dual based optimization procedure for the two-echelon uncapacitated facility location problem. Nav Res Logist 39:191–212
27. Geoffrion AM, Graves GW (1974) Multicommodity distribution system design by bender's decomposition. Manag Sci 20:822–844
28. Geoffrion AM, McBride R (1978) Lagrangean relaxation applied to capacitated facility location problem. AIIE Trans 10:40–47
29. Geoffrion AM, Powers RF (1995) Twenty years of strategic distribution system design: An evolutionary perspective. Interfaces 25:105–127
30. Goetschalckx M, Vidal CJ, Dogan K (2002) Modeling and design of global logistics systems: A review of integrated strategic and tactical models and design algorithms. Eur J Oper Res 143:1–18
31. Gutierrez GJ, Kouvelis P (1995) A robustness approach to international sourcing. Ann Oper Res 59:165–193
32. Head CK, Ries JD, Swenson DL (1994) The attraction of foreign manufacturing investments: Investment promotion and agglomeration economies. Working Paper 4878, NBER, Downloadable at URL: http://www.nber.org/papers/w4878.pdf
33. Hinojosa Y, Puerto J, Fernandez RF (2000) A multiperiod two-echelon multicommodity capacitated plant location problem. Eur J Oper Res 123:291
34. Hodder JE, Dincer MC (1986) A multifactor model for international plant location and financing under uncertainty. Comput Oper Res 13:601–609
35. Hodder JE, Jucker JV (1985) A simple plant location model for quantity-setting firms subjected to price uncertainty. Eur J Oper Res 21:39–46
36. Hodder JE, Jucker JV (1985) International plant location under price and exchange rate uncertainty. Eng Costs Prod Econ 9:225–229
37. Holmberg K (1994) Solving the staircase cost facility location problem with decomposition and piecewise linearization. Eur J Oper Res 75:41–61
38. Holmberg K, Ling J (1997) A lagrangean heuristic for the facility location problem with staircase costs. Eur J Oper Res 75:41–61
39. Jang YC, Jang SY, Chang BM, Park J (2002) A combined model of network design and production/distribution planning for a supply network. Comput Indust Eng 43:263–281
40. Jayaraman V, Pirkul H (2001) Planning and coordination of production and distribution facilities for multiple commodities. Eur J Oper Res 133:394–408
41. Karkazis J, Boffey TB (1981) The multi-commodity facilities location problem. J Oper Res Soc 32:803–814
42. Kaufman L, Eede MV, Hansen P (1977) A plant and warehouse location problem. Oper Res Q 28:547–554
43. Klincewicz JG, Luss H (1987) A dual based algorithm for multiproduct uncapacitated facility location. Trans Sci 21:198–206
44. Klose A, Drexl A (2005) Facility location models for distribution system design. Eur J Oper Res 162:4–29
45. Kouvelis P, Rosenblatt MJ, Munson CL (2004) A mathematical programming model for global plant location problems: Analysis and insights. IIE Trans 36:127–144
46. Kuehn AA, Hamburger MJ (1963) A heuristic program for warehouse location problem. Manag Sci 9:643–666
47. LeBlanc LJ (1977) A heuristic approach for large scale discrete stochastic transportation-location problem. Comput Math Appl 3:87–94
48. LeBlanc LJ et al (2004) Nu-kotes spreadsheet linear-programming models for optimizing transportation. Interfaces 34:139–146
49. Lim S, Kim Y (1998) Capacity Planning for Phased Implementation of Flexible Manufacturing Systems under Budget Restrictions. Eur J Oper Res 104:175–186
50. Martin CH, Dent DC, Eckhart JC (1993) Integrating production, distribution, and inventory planning at Libbey-Owens-Ford. Interfaces 23(3):68–78
51. Meixell MJ, Gargeya VB (2005) Global supply chain design: A literature review and critique. Trans Res Part E 41:531–550
52. Melachrinoudis E, Min H (2000) the dynamic relocation and phase-out of a hybrid, two-echelon plant/warehousing facility: Amultiple objective approach. Eur J Oper Res 123:1–15
53. Melo MT, Nickel S, da Gama FS (2005) Dynamic multi-commodity capacitated facility location: a mathematical modeling framework for strategic supply chain planning. Comput Oper Res 33:181–208
54. Moon S (1989) Application of generalized benders decomposition to a non-linear distribution system design problem. Nav Res Logist 36:283–295
55. Nauss RM (1978) An improved algorithm for the capacitated facility location problem. J Oper Res Soc 29:1195–1201
56. Neebe AW, Khumawala BM (1981) An improved algorithm for the multi-commodity location problem. J Oper Res Soc 32:143–169

57. Paquet M, Martel A, Desaulniers G (2004) Including technology selection decisions in manufacturing network design models. Int J Comput Integrat Manufact 17:117–125
58. Park YB (2005) An integrated approach for production and distribution planning in supply chain management. Int J Prod Res 43:1205–1224
59. Pirkul H, Jayaraman V (1996) Production, transportation, and distribution planning in a multi-commodity tri-echelon system. Trans Sci 30:291–302
60. Pirkul H, Jayaraman V (1997) Locating multi-echelon production and distribution facilities for multi-commodities. Working Paper, School of Business, University of Texas at Dallas
61. Pooley J (1994) Integrated production and distribution planning at ault foods. Interfaces 24:113–121
62. Rantala J (2004) Optimizing the Supply Chain Strategy of a Multi-Unit Finnish Nursery Company. Silva Fennica 38:203–215
63. Robinson EP Jr, Gao LL, Muggenborg SD (1993) Designing an integrated distribution system at Dow Brands, Inc. Interfaces 23:107–117
64. Sahin G, Sural H (2007) A review of hierarchical facility location models. Comput Oper Res 34:2310–2331
65. Sankaran JK, Raghavan NRS (1997) Locating and sizing plants for bottling propane in South India. Interfaces 27(6):1–15
66. Santoso T, Ahmed S, Goetschalckx M, Shapiro A (2005) A stochastic programming approach for supply chain network design under uncertainty. Eur J Oper Res 167:96–115
67. Sery S, Presti V, Shobrys DE (2001) http://www.atypon-link.com/INF/doi/abs/10.1287/inte.31.3.55.9635Optimization Models for Restructuring BASF North America's Distribution System Interfaces 31(May–June): 55–65
68. Shapiro JF, Singhal VM, Wagner SN (1993) Optimizing value chain. Interfaces 23:102–117
69. Shen Z-JM (2005) A multi-commodity supply chain design problem. IIE Transactions 37:753–762
70. Shen ZJM (2007) Integrated supply chain design models: A survey and future research directions. J Ind Manag Optim 3:1–27
71. Shycon HN, Maffei RB (1960) Simulation-tool for better distribution. Harvard Business Review, Nov–Dec:65–75
72. Snyder LV (2006) Facility location under uncertainty: a review. IIE Trans 38:537–554
73. Soland RM (1974) Optimal facility location with concave costs. Oper Res 22:373–382
74. Sourirajan K, Ozsen L, Uzsoy R (2007) A single-product network design model with lead time and safety stock considerations. IIE Trans 39:411–424
75. Spielberg K (1969) Algorithms for the simple plant location problem with some side constraints. Oper Res 17:85–111
76. Tcha D, Lee B (1984) A branch and bound algorithm for the multi-level uncapacitated facility location problem. Eur J Oper Res 18:35–43
77. Teo C-P Snyder LV, Daskin MS (2007) The stochastic location model with risk pooling. Eur J Oper Res 179:1221–1238
78. Tsiakis P, Shah N, Pantelides CC (2001) Design of Multi-echelon Supply Chain Networks under Demand Uncertainty. Ind Eng Chem Res 40:3585–3604
79. Tyagi R, Kalish P, Akbay K, Munshaw G (2004) GE plastics optimizes the two-echelon global fulfillment network at its high performance polymers division. Interfaces 34:359–366
80. Ulstein et al (2006) Elkem uses optimization in redesigning its supply chain. Interfaces 36:314–325
81. Van Roy TJ (1989) Multi-level production and distribution planning with transportation fleet optimization. Manag Sci 35:1443–1453
82. Van Roy TJ, Erlenkotter D (1982) Dual-based procedure for dynamic facility location. Manag Sci 28:1091–1105
83. Van Roy TJ, Gelders LF (1981) Solving distribution problem with side constraints. Eur J Oper Res 6:61–66
84. Verter V, Dasci A (2002) The plant location and flexible technology acquisition problem. Eur J Oper Res 136:366–382
85. Verter V, Dincer MC (1995) Facility location and capacity acquisition: An integrated approach. Nav Res Log 42:1141–1160
86. Verter V, Dincer MC (1995) Global manufacturing strategy. In: Drezner Z (ed) Facility Location: A Survey of Applications and Methods. Springer, New York
87. Vidal CJ, Goetschalckx M (1997) Strategic production-distribution models: A critical review with emphasis on global supply chain models. Eur J Oper Res 98:1–18
88. Warszawski A (1973) Multi-dimensional location problems. Oper Res Q 24:165–179
89. Wesolowsky GO, Truscott WG (1975) The multi-period location-allocation problem with relocation of facilities. Manag Sci 22:57–65
90. Wouda FHE, van Beek P, Jack GA, van der Vorst J, Tacke H (2002) An application of mixed-integer linear programming models on the redesign of the supply network of Nutricia Dairy and Drinks Group in Hungary. OR Spectr 24:449–465

Protein Folding: Generalized-Ensemble Algorithms

ULRICH H. E. HANSMANN
Department of Physics, Michigan Technological University, Houghton, USA

MSC2000: 92C05, 92C40, 92-08

Article Outline

Abstract
Keywords
Acknowledgment
References

Abstract

We briefly review the development of generalized-ensemble optimization techniques and their application since publication of "Protein Folding: Generalized-Ensemble Algorithms" was published in the first edition of this book.

Keywords

Energy landscape sampling; Model hopping; Parallel tempering

"Protein Folding: Generalized-Ensemble Algorithms" was submitted to the editors in February 1999 as a contribution for the "Encyclopedia of Optimization." While the article has remained useful as a short and concise introduction, the remarkable success over the last 8 years in the forming and application of generalized-ensemble techniques to optimization problems warrants some comments. New generalized-ensemble algorithms have been developed, and the simulation of small proteins (of order ≈ 50 residues) has become feasible.

One example for the recent algorithmic developments is energy landscape paving (ELP) [5]. Like all successful stochastic optimization techniques, it aims at avoiding entrapment in local minima and to continue the search for further solutions. For this purpose, one performs in ELP low-temperature Monte Carlo simulations with an effective energy:

$$w(\tilde{E}) = e^{-\tilde{E}/k_B T} \quad \text{with} \quad \tilde{E} = E + f(H(q,t)) \,. \quad (1)$$

Here, T is a (low) temperature and $f(H(q,t))$ a function of the histogram $H(q,t)$ in a prechosen "order parameter" q. The weight of a local minimum state decreases with the time the system stays in that state, i. e., ELP deforms the energy landscape locally till the local minimum is no longer favored, and the system will explore higher energies. It will then either fall in a new local minimum or walk through this high-energy region till the corresponding histogram entries all have similar frequencies and the system again has a bias toward low energies. Note that for $f(H(q,t)) = f(H(q))$ ELP reduces to the older *generalized-ensemble* methods. We have evaluated the efficiency of ELP in simulations of the 20-residue *trp*-cage protein whose structure we could "predict" within a root-mean-square deviation (rmsd) of 1 Å [8].

Note also that ELP allows even the possibility of zero-temperature simulations [8]. For $T \to 0$ only moves with $\Delta\tilde{E} \le 0$ will be accepted. If we choose $\tilde{E} = E + cH(E,t)$, we find as acceptance criterion:

$$\Delta E + c\Delta H(q,t) \le 0 \leftrightarrow c\Delta H(q,t) \le -\Delta E \,, \quad (2)$$

where E is the physical energy. Hence, within ELP the system can overcome even at $T = 0$ any energy barrier. The waiting time for such a move is proportional to the height of the barrier that needs to be crossed. Factor c sets now only the time scale, and in this sense the $T = 0$ form of ELP is parameter free.

Today, the most popular generalized-ensemble technique in protein science is parallel tempering (also known as replica exchange) [1,3]. In its most common form, one considers N *noninteracting* copies of the molecule, each at a different temperature T_i. In addition to standard Monte Carlo [3] or molecular dynamics moves [3,9] that affect only one copy, parallel tempering introduces a new *global* update [1]: the exchange of conformations between two copies i and $j = i + 1$ with probability

$$w(\mathbf{C}^{\text{old}} \to \mathbf{C}^{\text{new}}) = \min(1, \exp(-\beta_i E(C_j) - \beta_j E(C_i) + \beta_i E(C_i) + \beta_j E(C_j))) \,. \quad (3)$$

This exchange of conformations leads to a faster convergence than is observed in regular low-temperature canonical simulations. Note that parallel tempering does not require Boltzmann weights but can be combined easily with other generalized-ensemble techniques [3].

A variant of parallel tempering is "model hopping" [6], where sampling of low-energy configurations is enhanced by performing a random walk through an ensemble of systems with slightly altered energy functions. In that way, information is exchanged between varying stages of coarse graining or different local environments. We assume that the energy function can be divided into two terms: $E = E_A + aE_B$. As in parallel

tempering, "model hopping" considers N noninteracting copies of the molecule, but adjacent copies are now exchanged with probability

$$w(\mathbf{C}^{\text{old}} \to \mathbf{C}^{\text{new}}) = \min(1, \exp\{-\beta[E_A(C_j) + a_i E_B(C_j) + E_A(C_i) + a_j E_B(C_i) - E_A(C_i) - a_i E_B(C_i) - E_A(C_j) - a_j E_B(C_j)]\}) . \quad (4)$$

Here, $\Delta a = a_j - a_i$ and $\Delta E_B = E_B(C_j) - E_B(C_i)$. Configurations perform a random walk on a ladder of models with $a_1 = 1 > a_2 > a_3 > \ldots > a_N$ that differ by the relative contributions of E_B to the total energy E of the molecule. For instance, barriers in the energy landscape of proteins often arise from van der Waals repulsion between atoms that come too close. Hence, we have considered an implementation of "model hopping" with successively smaller contributions from the van der Waals energy. While the "physical" system is on one side of the ladder (at $a_1 = 1$), the (nonphysical) model on the other end of the ladder (at $a_N \ll 1$) may allow atoms to share the same position in space. As the protein "tunnels" through energy barriers, sampling of low-energy configurations is enhanced in the "physical" model (at $a_1 = 1$). With this realization of "model hopping" we could "predict" the structure of a 46-residue protein A in an all-atom simulation within a root mean square deviation (rmsd) of 3.2 Å [6]

Model hopping also allows guiding a simulation by information obtained from homologous structures [2]. Usually, such spatial constraints introduce an additional roughness into the energy landscape and therefore often lead to extremely slow convergence of the simulation. This problem is circumvented in our approach through a random walk in an ensemble of replicas that differ by the strength with which the constraints are coupled to the system. We have demonstrated the usefulness of this approach on some examples of the CASP6 competition [2].

Generalized ensemble, as discussed in "Protein Folding: Generalized-Ensemble Algorithms" and in this addendum, now allows the *in silico* study of small proteins (built out of $\approx$ 50 amino acids) using all-atom models. Examples include the 34-residue human parathyroid hormone fragment PTH(1-34) [4], the 28-residue FSD, the 36-residue villin headpiece subdomain [5,7,10], and fragment B of protein A (46 residues)[6]. Current work aims at extending the applicability of these methods to all-atom simulations of proteins built out of 50 to 100 residues.

Acknowledgment

This work was supported in part by the National Institutes of Health (USA) Grant GM62838 and National Science Foundation (USA) Grant CHE-0313618.

References

1. Geyer CJ, Thompson A (1995) Annealing Markov chain Monte Carlo with applications to ancestral inference. J Am Stat Ass 90:909; Hukushima K, Nemoto K (1996) Exchange Monte Carlo method and application to spin glass simulations. J Phys Soc Jpn 65:1604
2. Gront D, Kolinski A, Hansmann UHE (2005) Protein structure prediction by tempering spatial constraints. J Compaid Mol Des 19:603
3. Hansmann UHE (1987) Parallel tempering algorithm for conformational studies of biological molecules. Chem Phys Lett 281:140
4. Hansmann UHE (2004) Generalized-ensemble simulations of the human parathyroid hormone fragment. J Chem Phys 120:417
5. Hansmann UHE, Wille L (2002) Global optimization by energy landscape paving. Phys Rev Lett 88:068105
6. Kwak W, Hansmann UHE (2005) Efficient sampling of protein structures by model hopping. Phys Rev Lett 95:138102
7. Lin CY, Hu CK, Hansmann UHE (2003) Parallel tempering simulations. Proteins 52:436
8. Schug A, Wenzel W, Hansmann UHE (2005) Energy landscape paving simulations of the trp-cage protein. J Chem Phys 122:194711
9. Sugita Y, Okamoto Y (1999) Replica-exchange molecular dynamics method for protein folding. Chem Phys Lett 314:141
10. Trebst S, Troyer M, Hansmann UHE (2006) Optimized parallel tempering simulations of proteins. J Chem Phys 124:174903

Protein Loop Structure Prediction Methods

Martin Mönnigmann[1], Christodoulos A. Floudas[2]

[1] Institute for Heat and Fuel Technology, Technische Universität Braunschweig, Braunschweig, Germany

[2] Department of Chemical Engineering, Princeton University, Princeton, USA

MSC2000: 92C05, 92C40

Article Outline

Introduction
Method and Applications
Method
Applications
References

Introduction

One of the most demanding problems in computational biology is the so-called ab initio protein structure prediction problem. In ab initio prediction the objective is to predict protein structure solely from physically based force fields that describe the interactions between the amino acids and interactions between amino acids and the protein's environment. The ab initio protein structure prediction problem can be cast as an optimization problem in which the minimum free energy of the molecule is sought, because this configuration corresponds to the most stable structure of the protein.

In this contribution the problem of protein loop structure prediction is addressed. By the term loops we refer to any amino acid subsequence within a protein that is not of the geometrically regular type of an α-helix or a β-strand. The importance of loops for the overall three-dimensional structure and function of proteins has been pointed out before (see, e. g., Fiser et al. [5]). Even though loops are short amino acid subsequences, they are of major importance to the overall structure. Loops are often exposed to the surface and contribute to active and binding sites. Without loops, many proteins could not fold into compact structures.

Unfortunately, loop structure is considerably more difficult to predict than the geometrically regular β-strands and α-helices, since loops possess greater structural flexibility than strands and helices, and since they have relatively few contacts with the remainder of the structure.

Methods for loop structure prediction have been investigated for at least two decades [2]. Recent progress in loop structure prediction has been achieved with approaches that combine dihedral angle sampling, steric clash detection, clustering, and scoring or energy function evaluation to build up ensembles of loop conformations.

In the so-called loop *reconstruction* problem, the anchor geometry of the protein into which the loop must fit is assumed to be known. Here we address the problem of loop structure prediction when no information on the anchor geometry is available. More precisely, we assume the secondary structure of the stem residues is assumed to be known, but the geometry of the protein into which the loop must fit is considered to be unknown in our methodology. This loop structure prediction with flexible stems must be considered more difficult than the loop reconstruction problem.

Ultimately, the optimization based method for loop structure prediction summarized here is going to be embedded in an existing ab initio protein structure prediction method [10,11,12,13,14,15,16,17,18,19].

Method and Applications

This section first introduces a new methodology for loop structure prediction for loops with flexible stems. Subsequently, results are summarized that have been obtained for a large test set of 3215 loops of known structure.

Method

The loop prediction method proceeds along the following steps: (i) generating conformers by high-resolution dihedral angle sampling, (ii) structure optimization with a physically based energy function, (iii) iterative clustering of ensembles to discard conformers that are likely to have a large rmsd with respect to the native loop structure, and (iv) strategies for selecting optimal loops from the ensemble that remains after step (iii). These steps are briefly described. For details we refer to [20].

The description of the geometry of a loop consists of two elements. The geometry of the backbone is described in terms of the dihedral angles, ϕ and ψ. Correspondingly, the side chains of each amino acids are described by side chain dihedral angles. The number of the side chain dihedral angles depends on the type of amino acid. Step (i) generates a large number of candidate backbone conformations by sampling $n-1$ dihedral angle pairs (ϕ_i, ψ_i), $i = 1, \ldots, n-1$ from appropriate probability distributions for a loop that consists of n amino acids. Conformers are generated with probability functions in a discretized (ϕ, ψ)-space [20]. In

this approach the (ϕ, ψ) space is divided into 72^2 angular bins of size $5° \times 5°$. Functions for the probability to find a (ϕ, ψ) pair in any of the $5° \times 5°$ bins are derived for each of the 20 amino acids in each of three types of secondary structure, resulting in 60 probability functions. The three types considered are α-helical, β-strand, and loop. More precisely, helical and strand-type amino acids are defined by the DSSP classification codes H, E [9]. To qualify as a loop, an amino acid sequence must not be at either terminal of the protein and must be located between strands or helices. Probability functions are derived by counting occurrences of (ϕ, ψ)-pairs for each amino acid in each bin for a reference set of known protein structures. The reference set is chosen to be all proteins with experimental resolution of 2.2 Å or better in the PdbSelect25 set (http://www.cmbi.kun.nl/gv/dssp/)[9]. This results in a set of 939 reference proteins, and an overrepresentation of angular bins due to considering structures that are too similar is avoided. Similar dihedral angle bin sets have successfully been used before [3,4]. Compact and computationally efficient representations of the dihedral angle bin sets and their sampling for conformer generation exist. We refer to [20] for details.

Having generated a backbone geometry by dihedral angle sampling, side chain angles are optimized for each amino acid by looping over the amino acids and identifying the lowest energy conformation that can be achieved with any combination of angles from a well-established library of side chain conformations (the Dunbrack rotamer library [1]). While determining the optimal side chain configuration for an amino acid, the backbone and side chain angles of the remaining amino acids are fixed. The side chain optimization iterates over the sequence of amino acids that constitute the protein and terminates when no further improvement can be achieved. Energies are calculated with a physically based force field, the ECEPP/3 force field [21]. Covalent bond lengths and bond angles are fixed at their equilibrium values, so that the conformation is a function of the torsional angles only. The energy comprises electrostatic, nonbonded, hydrogen-bonded, and torsional contributions. After backbone generation by dihedral angle sampling, and after side chain optimization, the entire structure is subjected to an energy minimization in which all degrees of freedom, both all backbone angles, and all side chain angles, are optimized simultaneously. For this purpose a sequential quadratic programming algorithm [6] is used.

Using the approach described so far, a sufficiently large number $n_c^{(0)}$ of geometrical conformations is generated and the resulting ensemble is subjected to clustering. In all results reported here, the ensemble size was chosen to be $n_c^{(0)} = 2000$. Before introducing the clustering algorithm we need the following definitions. The cluster size N_i is defined by

$$N_i = \#\{j \mid r(i,j) \leq r_{\text{thresh}}\}. \tag{1}$$

In Eq. (1) $r(i,j)$ denotes the root mean square deviation (rmsd) between conformer i and j, and the pound symbol is the cardinality operator. For details on the rmsd calculation we refer to [20]. Note that the number of conformers n_c must be chosen with care, as the complexity of the pairwise rmsd calculation is $O(n_c^2)$. In the algorithm the upper index k is introduced to count the number of iterations. The symbols $n_c^{(k)}$, $N_{i,\max}^{(k)}$ and $r_{\max}^{(k)}$ denote the number of conformers in the ensemble in iteration k, the number of conformers N_i as defined in Eq. (1) in the largest cluster found in iteration k

$$N_{i,\max}^{(k)} = \max_{i=1,\ldots,n_c^{(k)}} N_i \tag{2}$$

and the largest pairwise rmsd found in the ensemble in iteration k

$$r_{\max}^{(k)} = \max_{i,j=1,\ldots,n_c^{(k)}} r(i,j). \tag{3}$$

The relative rmsd threshold $r_{\text{thresh,rel}} \in (0,1)$ and the relative critical cluster size $N_{\text{crit,rel}} \in (0,1)$ are used to express r_{thresh} and N_{crit} in terms of the largest pairwise rmsd $r_{\max}^{(k)}$ and the largest cluster $N_{i,\max}^{(k)}$ in each iteration,

$$r_{\text{thresh}}^{(k)} = r_{\text{thresh,rel}} r_{\max}^{(k)} \tag{4}$$

$$N_{\text{crit}}^{(k)} = N_{\text{crit,rel}} N_{i,\max}^{(k)}. \tag{5}$$

The algorithm can now be stated as follows [20].

1. *Initialization*
   Reset the iteration counter, $k = 0$. Choose the ensemble size $n_c^{(0)}$ and generate the ensemble. Choose relative thresholds $r_{\text{thresh,rel}}$ and $N_{\text{crit,rel}}$. Sample values are $r_{\text{thresh,rel}} = 0.5$ and $N_{\text{crit,rel}} = 0.5$.

2. *Determination of rmsd threshold* $r_{\text{thresh}}^{(k)}$
   Calculate pairwise rmsds $r(i,j)$ for $i = 1,\ldots,n_c^{(k)}$, $j = 1,\ldots,n_c^{(k)}$.
   Determine $r_{\text{max}}^{(k)}$ from Eq. (3) and set $r_{\text{thresh}}^{(k)}$ according to Eq. (4).
3. *Determination of clusters size threshold* $N_{\text{crit}}^{(k)}$
   Calculate cluster sizes $N_i^{(k)}$ for all conformers $i = 1,\ldots,n_c^{(k)}$ according to Eq. (1), where $r_{\text{thresh}} = r_{\text{thresh}}^{(k)}$ in Eq. (1).
   Calculate $N_{i,\text{max}}^{(k)}$ according to Eq. (2). Set $N_{\text{crit}}^{(k)}$ according to Eq. (5).
4. *Termination criterion*
   If $N_i \geq N_{\text{crit}}^{(k)}$ for all $i = 1,\ldots,n_c^{(k)}$, stop.
5. *Discarding far-from-native conformations*
   Increment the iteration counter k.
   Discard conformations i for which $N_i < N_{\text{crit}}^{(k)}$. Consider the remaining conformations to constitute the new ensemble. Set $n_c^{(k)}$ to the number of conformers in the new ensemble, re-enumerate the conformers in the new ensemble with numbers $i = 1,\ldots,n_c^{(k)}$, and go to step 2.

We stress that the clustering algorithm is based on the idea of discarding structures that are likely not to be close to the native structure. In contrast, existing clustering algorithms are based on selecting structures that are close to native [3,4,7,8,22].

For test cases with known three-dimensional structure, the quality of the sampling approach can be assessed by identifying the lowest rmsd to the known structure in an ensemble. This check reveals that sampling is not restricting the overall prediction strategy [20].

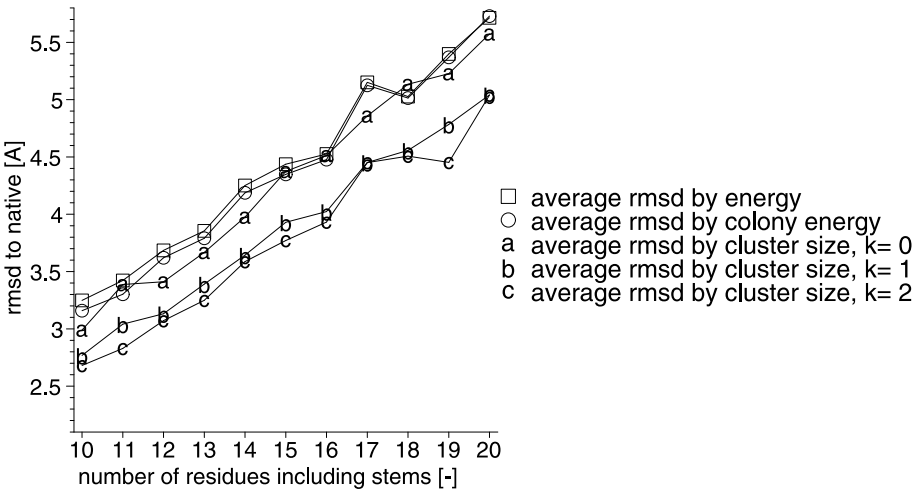
Ultimately, a valid strategy is needed for selecting one or a few conformers from an ensemble that are likely to have a small rmsd with respect to the native structure when the native structure is not known. The potential energy has been reported not to be a useful criterion [8,22]. Among other things, this can be accounted to the fact that the potential energy does not take entropic contributions into account. As a remedy, the so-called colony energy has been suggested as an approximate approach to taking entropic contributions into account [22]. Alternatively one can disregard potential energy completely and identify conformers that are likely to be close to native based on the size of clusters they form within the ensemble. We tested the potential energy, colony energy, cluster size, and a hybrid approach on a large set of 3215 loops with known structure.

Applications

In this section we summarize the results that have been obtained for a large test set of loops with known structure. Using the PdbSelect25 we identified a test set of 716 loops of total length 10 (4 loop residues and 6 stem residues), 656 loops of length 11 (5 loop residues and 6 stem residues), 387 loops of length 12 (6 loop residues and 6 stem residues), 366 loops of length 13 (7 loop residues and 6 stem residues), 283 loops of length 14 (8 loop residues and 6 stem residues), 223 loops of length 15 (9 loop residues and 6 stem residues), 176 loops of length 16 (10 loop residues and 6 stem residues), 138 loops of length 17 (11 loop residues and 6 stem residues), 115 loops of length 18 (12 loop residues and 6 stem residues), 75 loops of length 19 (13 loop residues and 6 stem residues), and 80 loops of length 20 (14 loop residues and 6 stem residues). To our knowledge, this is the largest dataset for which loop structure prediction results have been reported to date. We refer to [20] for details.

The results are summarized in Fig. 1. The results labeled *average rmsds by energy* are obtained by identifying the lowest ECEPP/3 energy conformer in each ensemble, recording its rmsd to native, and subsequently averaging over the rmsds for all ensembles of loops of a given length. The average rmsds by colony energy results when using the colony energy [22] instead of the ECEPP/3 energy. The remaining data shown in Fig. 1 is obtained with the clustering algorithm. For all of the loops reported here, the clustering algorithm terminated after $k = 2$ steps.

Figure 1 shows that the conformer that generates the largest cluster after any clustering step $k = 0, 1, 2$ is on average a better prediction for the native structure than both the lowest energy and lowest colony energy conformer. From this we infer that the cluster size is a better criterion for the identification of conformers than energy or colony energy. Furthermore, the largest cluster conformer found after step $k + 1$ of the clustering algorithm is as good as, or better than, the largest cluster conformer found in step k. In this sense,



Protein Loop Structure Prediction Methods, Figure 1
Average of rmsds as a function of total loop length for the loops from the PdbSelect25 proteins. Lines are added as a guide to the eye. Results are reproduced from [20]

repeatedly applying the clustering algorithm improves the largest cluster conformer on average.

In order to assess the computational demand created by the suggested approach, we measured the cpu times needed by our approach for 20 randomly selected loops of each sequence length. The resulting average times are 1.5 h, 5 h, and 11.25 h for a single loop with 4 loop and 6 stem residues, 10 loop and 6 stem residues, 14 stem and 6 loop residues, respectively, when the ensemble size is chosen to be 2000 conformers. These times were obtained on a single Intel Xeon 3 GHz machine running RedHat Linux 9.0.

The ensemble generation step needs a negligible fraction of the total time. The second step, energy minimization, is the computationally most demanding step. The times needed for pairwise rmsd calculations contribute between 3.5 min and 7 min to the total times given above. The time needed for the application of the clustering algorithm detailed in Sect. "Method" is independent of the loop length and contributes about 4 min per loop to the total times given above. The time-consuming step of energy minimization is amenable to parallelization, since the loops in an ensemble can be treated separately.

References

1. Bower MJ, Cohen FE, Dunbrack RL (1997) Prediction of protein side chain rotamers from a backbone dependent rotamer library: a new homology modeling tool. J Mol Biol 267:1268–1282
2. Bruccoleri RE, Karplus M (1987) Prediction of the folding of short polypeptide segments by uniform conformational sampling. Biopolym 26:137–168
3. de Bakker PIW, DePristo MA, Burke DF, Blundell TL (2003) Ab initio construction of polypeptide fragments: Accuracy of loop decoy discrimination by an all-atom statistical potential and the AMBER force field with the generalized Born solvation model. Proteins: Struct Funct Bioinform 51:21–40
4. DePristo MA, de Bakker PIW, Lovell SC, Blundell TL (2003) Ab initio construction of polypeptide fragments: Efficient generation of accurate, representative ensembles. Proteins: Struct Funct Bioinform 51:41–55
5. Fiser A, Do RKG, Sali A (2000) Modeling of loops in protein structures. Protein Sci 9:1753–1773
6. Gill PE, Murray W, Saunders MA, Wright MH (1986) NPSOL 4.0 User's Guide. Systems Optimization Laboratory, Dept. of Operations Research, Stanford University, CA
7. Hartigan JA, Wong MA (1979) Algorithm AS 136: A K-means clustering algorithm. Appl Stat 28:100–108
8. Jacobson MP, Pincus DL, Rappa CS, Day TJF, Honig B, Shaw DE, Friesner RA (2004) A hierarchical approach to all-atom protein loop prediction. Proteins: Struct Funct Bioinform 55:351–367
9. Kabsch W, Sander C (1983) Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. Biopolym 22:2577–2637
10. Klepeis JL, Floudas CA (1999) Free energy calculations for peptides via deterministic global optimization. J Chem Phys 110:7491–7512

11. Klepeis JL, Floudas CA (2002) Ab initio prediction of helical segments in polypeptides. J Comput Chem 23:245–266
12. Klepeis JL, Floudas CA (2003) Ab initio tertiary structure prediction of proteins. J Glob Optim 25:113–140
13. Klepeis JL, Floudas CA (2003) ASTRO-FOLD: A combinatorial and global optimization framework for ab initio prediction of three-dimensional structures of proteins from the amino acid sequence. Biophys J 85:2119–2146
14. Klepeis J, Floudas C (2003) Prediction of beta-sheet topology and disulfide bridges in polypeptides. J Comput Chem 24(2):191–208
15. Klepeis JL, Floudas CA (2005) Analysis and prediction of loop segments in protein structure. Comput Chem Eng 29(3):423–436
16. Klepeis JL, Floudas CA, Morikis D, Lambris JD (1999) Predicting peptide structures using nmr data and deterministic global optimization. J Comput Chem 20:1354–1370
17. Klepeis JL, Pieja MT, Floudas CA (2003) A new class of hybrid global optimization algorithms for peptide structure prediction: Integrated hybrids. Comput Phys Commun 151:121–140
18. Klepeis JL, Pieja MT, Floudas CA (2003) Hybrid global optimization algorithms for protein structure prediction: Alternating hybrids. Biophys J 84:869–882
19. Klepeis J, Wei Y, Hecht M, Floudas C (2005) Ab initio prediction of the 3-dimensional structure of a de novo designed protein: A double blind case study. Proteins: Struct Funct Bioinform 58:560–570
20. Mönnigmann M, Floudas CA (2005) Protein loop structure prediction with flexible stem geometries. Proteins: Struct Funct Bioinform 61(4):748–762
21. Némethy G, Gibson KD, Palmer KA, Yoon CN, Paterlini G, Zagari A, Rumsey S, Scheraga HA (1992) Energy parameters in polypeptides. 10. Improved geometrical parameters and nonbonded interactions for use in the ECEPP/3 algorithm, with application to proline-containing peptides. J Phys Chem 96:6472–6484
22. Xiang Z, Soto C, Honig B (2002) Evaluating conformational free energies: the colony energy and its application to the problem of loop prediction. Proceedings of the National Academy of Sciences of the United States of America, vol 99. pp 7432–7437

Pseudomonotone Maps: Properties and Applications

NICOLAS HADJISAVVAS
Department of Product and System Design Engineering, University of the Aegean, Hermoupolis, Greece

MSC2000: 49J40; 49J53; 47H05; 47H04; 26B25

Article Outline

Introduction/Background
Definitions
Formulation
  Maximal Pseudomonotonicity
  A Generalization of Paramonotone Maps
  Pseudoaffine Maps
  Pseudomonotone vs. Monotone Maps
Methods/Applications
Conclusions
References

Introduction/Background

Pseudomonotone maps were introduced by Karamardian as a generalization of monotone maps [22]. Other generalizations include various kinds of quasimonotone maps (quasimonotone, strictly quasimonotone, semistrictly quasimonotone ...). These generalizations, the relations between them, as well as the relation to generalized convexity are discussed in other articles of the Encyclopedia (see ► generalized monotone single valued maps and ► generalized monotone multivalued maps and in [16].

It should be noted that the same term a "pseudomonotone map" has been introduced by Brezis to denote a totally different class of maps [4]. The main difference between the two classes is that pseudomonotone maps in the sense of Brézis are defined through a kind of continuity property, whereas Karamardian used only the order relation of real numbers in his definition. For this reason some authors, starting by Gwinner [12], use the term "topologically pseudomonotone" for pseudomonotone maps in the Brezis sense. Although it is possible to give a definition that includes both kinds of pseudomonotonicity [11], we will use the term "pseudomonotone" only in the sense defined by Karamardian.

Pseudomonotone maps have the advantage that they lead to generalizations of existence theorems for the Stampachia variational inequality problem (VIP), without imposing additional assumptions, and with practically the same proof as for monotone maps [16]. However, this quasi-identical treatement of the VIP is not extended to other topics, and the properties of the two classes of maps are often quite dissimilar. For instance, while the sum of two monotone maps is mono-

tone, this is false for pseudomonotone maps. A vast theory has been developed for monotone maps, based on the concept of maximal monotonicity. By contrast, until recently it was believed that maximality plays no rôle for pseudomonotone maps. Consequently, some algorithms for finding the solution of VIP with maximal monotone maps have no extension to the pseudomonotone case.

This article will present some recent developments that can be considered as a first step towards filling the lacunae in the theory of pseudomonotone maps. In particular, maximal pseudomonotonicity will be discussed. The main tool is the definition of an equivalence relation in the set of all pseudomonotone maps. Also, a generalization of paramonotone maps and their use in cutting plane algorithms will be described. Finally, recent results on pseudoaffine maps and on the relation to monotone maps will be presented.

Definitions

Let X be a real Banach space and X^* be its dual. Given $x, y \in X$, $[x, y]$ denotes the line segment $\{(1-t)x+ty : t \in [0,1]\}$. For $K \subseteq X^*$, $\mathbb{R}_{++}K$ will be the set $\bigcup_{t>0} tK$. A multivalued map $T : X \to 2^{X^*}$ is a map whose values are subsets of X^*, possibly empty. The *domain* $D(T)$ of T is the set $\{x \in X : T(x) \neq \emptyset\}$, its graph the set $\mathrm{gr}(T) = \{(x, x^*) \in X \times X^* : x^* \in T(x)\}$ and its set of zeros is the set $Z_T = \{x \in X : 0 \in T(x)\}$. The map T is called upper sign-continuous [13] if for all $x \in D(T)$ and $v \in X$, the following implication holds:

$$\left(\forall t \in (0,1), \inf_{x^* \in T(x+tv)} \langle x^*, v\rangle \geq 0\right) \Rightarrow \sup_{x^* \in T(x)} \langle x^*, v\rangle \geq 0.$$

If T is upper hemicontinuous (i. e., its restriction on line segments is upper semicontinuous with respect to the weak* topology in X^*), then it is upper sign-continuous.

The map T is called monotone if for every (x, x^*), $(y, y^*) \in \mathrm{gr}(T)$, $\langle y^* - x^*, y - x\rangle \geq 0$; it is called maximal monotone if its graph is not strictly contained in the graph of any other monotone map. Also, T is called D-maximal monotone if its graph is not strictly contained in the graph of any other monotone map with the same domain.

The map T is called pseudomonotone if for every (x, x^*), $(y, y^*) \in \mathrm{gr}(T)$, the following implication holds.

$$\langle x^*, y - x\rangle \geq 0 \Rightarrow \langle y^*, y - x\rangle \geq 0.$$

Obviously, every monotone map is pseudomonotone.

Given a locally Lipschitz function $f : X \to \mathbb{R} \cup \{+\infty\}$, we denote by $\partial^0 f$ its Clarke subdifferential [7]. The locally Lipschitz function f is called pseudoconvex if for every $x \in \mathrm{dom}(f)$ and $x^* \in \partial^0 f(x)$ the following implication holds:

$$\langle x^*, y - x\rangle \geq 0 \Rightarrow f(y) \geq f(x).$$

It is known that a locally Lipschitz function f is pseudoconvex if and only if $\partial^0 f$ is a pseudomonotone map [23].

Formulation

Maximal Pseudomonotonicity

In order to introduce maximal pseudomonotone maps, one first defines an equivalence relation in the set of pseudomonotone maps. Two pseudomonotone maps T_1 and T_2 are called equivalent if they have the same domain, the same set of zeros, and for each x which is not a common zero, the elements of $T_1(x)$ are positive multiples of the elements of $T_2(x)$ and vice versa. In other words,

(a) $D(T_1) = D(T_2)$
(b) $Z_{T_1} = Z_{T_2}$,
(c) for every $x \in X \backslash Z_{T_1}$, $\mathbb{R}_{++}T_1(x) = \mathbb{R}_{++}T_2(x)$.

In this case we write $T_1 \sim T_2$. This is an equivalence relation. Another aspect of this equivalence is provided by the Stampacchia variational inequality. Given a map T and a convex subset K of X, we denote by $S(T, K)$ the set of all $x \in K$ which are solutions of the VIP:

$$\forall y \in K, \exists x^* \in T(x) : \langle x^*, y - x\rangle \geq 0.$$

The following result holds [14].

Proposition 1 *Let T_1, T_2 be pseudomonotone maps. If $T_1 \sim T_2$, then $S(T_1, K) = S(T_2, K)$ for every convex set $K \subseteq X$. Conversely, if $S(T_1, K) = S(T_2, K)$ for every line segment K and T_1, T_2 have weak*-compact convex values, then $T_1 \sim T_2$.*

Since all equivalent maps provide the same solutions to VIP, one can choose any element of the equivalence class to study or even find the solutions.

Given a pseudomonotone map T, its equivalence class has a maximum with respect to graph inclusion. This is simply the map $\hat{T}$ defined by $\hat{T}(x) = \cup_{S\sim T} S(x)$ for all $x \in D(T)$. It can be shown [13] that $\hat{T}$ is also given by the formula

$$\hat{T}(x) = \begin{cases} \emptyset, & \text{if } x \notin D(T) \\ \mathbb{R}_{++}T(x), & \text{if } x \in D(T)\backslash Z_T \\ N_{L_{T,x}}, & \text{if } x \in Z_T \end{cases}$$

where $N_{L_{T,x}}$ is the normal cone at x to the set $L_{T,x} = \{y \in X : \exists y^* \in T(y), \langle y^*, y - x\rangle \leq 0\}$.

A pseudomonotone map $\hat{T}$ is called D-maximal pseudomonotone, if the graph of $\hat{T}$ is not properly contained in the graph of any other map with the same domain. When the domain of T is convex, there is an equivalent, more appealing definition for D-maximal pseudomonotonicity [14]:

Proposition 2 *Let T be pseudomonotone and such that $D(T)$ is convex. Then T is D-maximal pseudomonotone if, and only if, every pseudomonotone extension of T with the same domain is equivalent to T.*

Some properties of the set of zeros of T are provided by the following proposition [14].

Proposition 3 *Let T be D-maximal pseudomonotone. Then Z_T is weakly closed in $D(T)$. If in addition $D(T)$ is convex, then Z_T is also convex, and $z \in Z_T$ is equivalent to*

$$\forall (y, y^*) \in \mathrm{gr}(T),\ \langle y^*, y - z\rangle \geq 0.$$

The following proposition provides a simple criterion for showing the D-maximal pseudomonotonicity of a map [13].

Proposition 4 *Assume that T is pseudomonotone, upper-sign continuous, with weak*-compact, convex values and open domain $D(T)$. Then T is D-maximal pseudomonotone.*

A simple consequence of the above proposition is:

Corollary 5 *The Clarke subdifferential $\partial^0 f$ of a locally Lipschitz, pseudoconvex function $f : X \to \mathbb{R} \cup \{+\infty\}$ is a D-maximal pseudomonotone map.*

As was explained before, from the point of view of VIP one can use any element of the equivalence class. This is fortunate, since in many cases instead of showing that a D-maximal pseudomonotone map has a "nice" property (as is the case with maximal monotone maps), one shows that an equivalent map has this property. For instance, one has:

Proposition 6 *If T is D-maximal pseudomonotone, then $\hat{T}(x)$ is convex for every $x \in D(T)$. If in particular the assumptions of Proposition 4 are satisfied, then $\hat{T}(x) \cup \{0\}$ is weak*-closed.*

Here is a case where one can find an equivalent map with a better continuity property [13]:

Proposition 7 *Let $T : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ be a pseudomonotone map, upper sign-continuous, with compact convex values. If $D(T)$ is open and convex, then there exists an equivalent upper semicontinuous map T_1 with compact convex values.*

For instance let T be a single-valued pseudomonotone map defined on an open convex subset of $\mathbb{R}^n$. If T is hemicontinuous (i. e., continuous along line segments) then the above proposition guarantees that there exists an equivalent map which is continuous. Likewise, one can show that under some fairly general assumptions, T is equivalent to a map which is generically single-valued (i. e., is single valued except on a set of the first category). See Corollary 3.10 in [13].

A Generalization of Paramonotone Maps

A monotone multivalued map T is called paramonotone if for every (x, x^*), $(y, y^*) \in \mathrm{gr}(T)$, $\langle y^* - x^*, y - x\rangle = 0$ implies that $x^* \in T(y)$ and $y^* \in T(x)$. It can be shown that the subdifferential of a proper lsc convex function is paramonotone [5]. Other examples of paramonotone maps are given in [21]. Paramonotone maps have been extensively used in algorithms for the solution of VIP [5,6,24]. The main reason is that these maps have the following "cutting plane property":

$$\left.\begin{array}{c} x \in S(T, K) \\ y \in K \\ \langle y^*, x - y\rangle \geq 0 \\ \text{for some } y^* \in T(y) \end{array}\right\} \Rightarrow y \in S(T, K). \qquad (1)$$

Assume that a map has property (1). If at the nth iteration of an algorithm one finds a point y_n that is not a solution of VIP, then all solutions of VIP belong to the intersection of K with the halfspace $\{x \in X : \langle y_n^*, x - y_n\rangle < 0\}$ where y_n^* is an arbitrary element of $T(y_n)$.

Let $K \subseteq X$ be nonempty, closed and convex. A single valued pseudomonotone map $T : K \to X^*$ is called pseudomonotone$_*$ if for all $x, y \in K$,

$$\langle T(x), y - x\rangle = \langle T(y), y - x\rangle = 0$$

implies that $T(x) = kT(y)$, for some $k > 0$ [8]. Note that single-valued pseudomonotone$_*$ maps are a generalization of single-valued paramonotone maps. To extend this generalization to the multivalued case, one needs the tools presented in the previous subsection.

Definition 8 [17] A map $T : X \to 2^{X^*}$ is pseudomonotone$_*$ on K if it is pseudomonotone and for every $x, y \in K$ and $x^* \in T(x)$, $y^* \in T(y)$, $\langle x^*, y - x\rangle = \langle y^*, y - x\rangle = 0$ imply $x^* \in \hat{T}(y)$ and $y^* \in \hat{T}(x)$.

It is easy to see that every paramonotone map is pseudomonotone$_*$. Other classes of pseudomonotone$_*$ maps is provided by the following propositions [17].

Proposition 9 *The Clarke subdifferential $\partial^0 f$ of a locally Lipschitz pseudoconvex function f is pseudomonotone$_*$.*

Proposition 10 *If the map T is pseudomonotone$_*$, then any map equivalent to T is pseudomonotone$_*$.*

Proposition 9 is a particular case of a more general situation. A map T is called cyclically pseudomonotone [9,10] if for every $(x_i, x_i^*) \in \mathrm{gr}(T)$, $i = 1, 2, \ldots, n$, the following implication holds:

$$\langle x_i^*, x_{i+1} - x_i\rangle \geq 0,\ \forall i = 1, 2, \ldots, n-1$$
$$\Rightarrow \langle x_n^*, x_1 - x_n\rangle \leq 0.$$

Proposition 11 *If T is D-maximal pseudomonotone and cyclically pseudomonotone with convex domain, then it is pseudomonotone$_*$.*

Since the Clarke subdifferential of a locally Lipscitz pseudoconvex function is D-maximal pseudomonotone and cyclically pseudomonotone, we see that the previous proposition implies Proposition 9.

We saw that paramonotone maps have the cutting plane property (1). The same is true for pseudomonotone$_*$ maps; what is more interesting is that these maps are characterized in some sense by the cutting plane property:

Proposition 12 *Let T be pseudomonotone on the convex set K. If T is pseudomonotone$_*$, then property (1) holds on every subset of K. Conversely, if property (1) holds on every convex, compact subset of K and T has convex, weak*-compact values, then T is pseudomonotone$_*$ on the interior of K.*

If T is single-valued, the assumption of pseudomonotonicity becomes redundant:

Proposition 13 *Let $T : K \to X^*$ be hemicontinuous. If T has property (1) on each convex compact subset of K, then T is pseudomonotone on K and pseudomonotone$_*$ on its interior.*

In Sect. "Methods/Applications" we will show how to apply pseudomonotone$_*$ maps for the solution of variational inequalities.

Pseudoaffine Maps

Given a convex subset K of $\mathbb{R}^n$, a single-valued map $T : K \to \mathbb{R}^n$ is called pseudoaffine (or PPM, as in [3]) if both T and $-T$ are pseudomonotone. These maps were studied in [3] in connection with VIP. It is easy to see that a differentiable function $f : K \to \mathbb{R}$ is pseudolinear (i. e., both f and $-f$ are pseudoconvex) if and only if ∇f is pseudoaffine. It is not hard to show that pseudolinear functions defined on the whole space $\mathbb{R}^n$ have a very particular form [2,25]:

Proposition 14 *A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is pseudolinear if and only if there exist a vector $u \in \mathbb{R}^n$ and a one-variable differentiable function h whose derivative is always positive or identical to zero, such that $f(x) = h(\langle u, x\rangle)$.*

If $T = \nabla f$ in this case, then $T(x) = h'(u, x)$, i. e., T is equal to a positive multiple of a constant vector. For general pseudoaffine maps (i. e., those that are not necessarily equal to a gradient) that are defined on the whole space, the following elegant characterization has been shown:

Proposition 15 *A map $T : \mathbb{R}^n \to \mathbb{R}^n$ is pseudoaffine if and only if there exists a positive function $g : \mathbb{R}^n \to \mathbb{R}$, a skew-symmetric linear map A and a vector u such that*

$$\forall x \in \mathbb{R}^n,\ T(x) = g(x)(Ax + u).$$

The proof of the above result needs some "global" arguments provided by algebraic topology and by projective geometry [2].

Pseudomonotone vs. Monotone Maps

One of the basic differences between the class of monotone maps and the class of pseudomonotone maps has to do with their stability with respect to some operations. For instance, the class of monotone maps is stable with respect to addition (i. e., the sum of two monotone maps is monotone), while this is not the case for pseudomonotone maps. By contrast, the product of a pseudomonotone map with a positive function produces a pseudomonotone map while this is not the case for monotone maps.

In particular, it was noted in [1] that a map $T : X \to 2^{X^*}$ is monotone if and only if for every $x^* \in X^*$ the map $T + x^*$ is pseudomonotone. More recently He [18] and Isac and Motreanu [20] obtained another result in this direction. Assume that X is a Hilbert space (in [18] one considered $X = \mathbb{R}^n$) and $K \subseteq X$ is a convex set with nonempty interior. Let further $T : K \to X^*$ be a continuous single-valued map which is Gâteaux differentiable in the interior of K. Then T is monotone if and only if $T + x^*$ is pseudomonotone for all x^* in a straight line of X^*. The differentiability assumption is essential in the argument of both papers [18,20] because the proof is based on a first-order characterization of generalized monotonicity.

In a recent paper [15] it was shown that the differentiability assumption is redundant, and one can also weaken considerably the assumption that the interior of K is nonempty. Given $x^* \in X^*$ and a set $K \subseteq X$, one says that x^* is perpendicular to K if the value of x^* is constant on K, i. e., $\langle x^*, y - x\rangle = 0$ for all $x, y \in K$. The following proposition holds.

Proposition 16 *Let $K \subseteq X$ be nonempty and convex and $T : K \to 2^{X^*}$ be a map with nonempty values. Assume that there exists a straight line $S = \{x_0^* + tx^* : t \in \mathbb{R}\}$ in X^* such that x^* is not perpendicular to K, and for all $z^* \in S$, $T + z^*$ is pseudomonotone. Then T is monotone.*

In case K has nonempty interior or, more generally, nonempty quasi-interior, the assumption "x^* is not perpendicular to K" is automatically fulfilled. It should also be noted that the results of this subsection are also true if we replace "pseudomonotone" by "quasimonotone" (see the article ► generalized monotone multivalued maps in this Encyclopedia for the definition).

Methods/Applications

Many of the algorithms used to find a solution of a variational inequality with a paramonotone map, can be also used in the more general case of a pseudomonotone$_*$ map. We illustrate this by an example of a perturbed auxiliary problem method. Let K be a closed convex subset of a Hilbert space H, $T : K \to 2^H$ a map with nonempty values. Choose a Gâteaux differentiable strongly convex function $M : H \to \mathbb{R}$ with a weakly continuous derivative (we can take for instance $M(x) = \|x\|^2/2$). Construct a sequence $\{x_k\}_{k\in\mathbb{N}}$ by the following algorithm.

(i) Choose an arbitrary $x_0 \in K$.
(ii) Having chosen x_k, find $x_{k+1} \in K$ and $x_{k+1}^* \in T(x_{k+1})$ such that

$$\forall y \in K,\ \langle \mu_k x_{k+1}^* + M'(x_{k+1}) - M'(x_k), y - x_{k+1}\rangle \geq 0$$

where $\{\mu_k\}_{k\in\mathbb{N}}$ is a sequence of positive constants bounded from below. Note that finding x_{k+1} amounts to solving VIP for the perturbed map $T_{k+1}(\cdot) = \mu_{k+1}T(\cdot) + M'(\cdot) - M'(x_k)$. This problem can be much easier than the original one, since for instance if T is weakly monotone and μ_{k+1} is small, then T_{k+1} is strongly monotone.

Assume that VIP has a solution and that the sequence $\{x_k\}_{k\in\mathbb{N}}$ is well-defined. Then it can be shown that if T is pseudomonotone$_*$ and satisfies a fairly general continuity condition, then the sequence $\{x_k\}_{k\in\mathbb{N}}$ converges weakly to a solution of VIP for T. Details can be found in [17].

Conclusions

The theory of pseudomonotone maps is far from been developed to a satisfactory level. By contrast, the theory of monotone maps has reached a high level of maturity [19]. It is hoped that some of the recent advances presented here, and in particular the ideas on maximality of pseudomonotone maps, will provide a firm background for the study of pseudomonotone maps. This is illustrated by the ease and naturalness with

which notions like paramonotonicity can be generalized to pseudomonotone maps, a task that seemed almost impossible before the introduction of maximal pseudomonotone maps. In addition, the new notion of a pseudomonotone$_*$ map seems to be ideally fit the cutting plane property (see Proposition 12) and this adds some confidence that the definition of maximality is on the right way.

References

1. Aussel D, Corvellec JN, Lassonde M (1994) Subdifferential characterization of quasiconvexity and convexity. J Convex Anal 1:195–201
2. Bianchi M, Hadjisavvas N, Schaible S (2003) On Pseudomonotone Maps T for which $-T$ is also pseudomonotone. J Convex Anal 10:149–168
3. Bianchi M, Schaible S (2000) An extension of pseudolinear functions and variational inequality problems. J Optim Theory Appl 104:59–71
4. Brezis H (1968) Equations et inéquations nonlinéaires dans les espaces vectoriels en dualité. Ann Inst Fourier 18:115–175
5. Bruck RE Jr (1976) An iterative solution of a variational inequality for certain monotone operators in Hilbert space. Bull Amer Math Soc 81:890–892; Corrigendum, Bull Amer Math Soc 82 (1976) 353
6. Burachik R, Iusem A (1998) A generalized proximal point algorithm for the variational inequality problem in Hilbert space. SIAM J Optim 8:197–216
7. Clarke FH (1983) Optimization and nonsmooth Analysis. Wiley Interscience, New York
8. Crouzeix JP, Marcotte P, Zhu D (2000) Conditions ensuring the applicability of cutting-plane methods for solving variational inequalities. Math Program 88:521–539
9. Daniilidis A, Hadjisavvas N (1999) On the subdifferentials of pseudoconvex and quasiconvex functions and cyclic monotonicity. J Math Anal Appl 237:30–42
10. Daniilidis A, Hadjisavvas N (2000) On generalized cyclically monotone operators and proper quasimonotonicity. Optimization 47:123–135
11. Domokos A, Kolumban J (2000) Comparison of two different types of pseudomonotone mappings, In: Popoviciu E (ed) Seminaire de la théorie de la meilleure approximation, convexité et optimisation. Editura SRIMA, Cluj-Napoca, pp 95–103
12. Gwinner J (1981) On fixed points and variational inequalities—a circular tour. Nonlinear Anal 5:565–583
13. Hadjisavvas N (2003) Continuity and maximality properties of pseudomonotone operators. J Convex Anal 10:459–469
14. Hadjisavvas N (2003) Maximal pseudomonotone operators, In: Crespi GP, Guerraggio A, Miglierina E, Rocca M (eds) Recent advances in Optimization. Datanova Editrice, Milano, pp 87–100
15. Hadjisavvas N (2006) Translations of quasimonotone maps and monotonicity. Appl Math Lett 19:913–915
16. Hadjisavvas N, Komlosi S, Schaible S (2005) Handbook of generalized convexity and generalized monotonicity. Springer, New York
17. Hadjisavvas N, Schaible S (2006) On a generalization of paramonotone maps and its application to solving the Stampacchia variational inequality. Optim 55:593–604
18. He Y (2004) A relationship between pseudomonotone and monotone mappings. Appl Math Lett 17:459–461
19. Hu S, Papageorgiou NS (1997) Handbook of Multivalued Analysis, vols I, II. Kluwer, Dordrecht
20. Isac G, Motreanu D (2004) Pseudomonotonicity and quasimonotonicity by translations versus monotonicity in Hilbert spaces. Austral J Math Anal Appl 1:1–8
21. Iusem AN (1998) On some properties of paramonotone operators. J Convex Anal 5:269–278
22. Karamardian S (1976) Complementarity over cones with monotone and pseudomonotone maps. J Optim Theory Appl 18:445–454
23. Penot J-P, Quang PH (1997) Generalized convexity of functions and generalized monotonicity of set-valued maps. J Optim Theory Appl 92:343–356
24. Solodov MD, Svaiter BF (2000) An inexact hybrid generalized proximal point algorithm and some new results on the theory of Bregman functions. Math Oper Res 25:214–230
25. Thompson WA, Parke DW (1973) Some properties of generalized concave functions. Oper Res 21:305–313

Q

QBB Global Optimization Method

YUSHAN ZHU
Department of Chemical Engineering,
Tsinghua University, Beijing, China

MSC2000: 49M37, 65K10, 90C26, 90C30

Article Outline

Keywords and Phrases
Introduction
Formulation
Simplicial Partition
Quadratic Underestimation Function
for General Non-convex Structures
QBB Underestimators for Special Structures
Properties of the QBB Underestimator
Function Decomposition
Algorithmic Procedure of QBB
Conclusion
References

Keywords and Phrases

Global optimization; Branch-and-bound; Simplicial partition; Quadratic underestimation function

Introduction

Most problems of process design, process control, process operations, and molecule design are determined by the optimal solutions; however, those problems are mainly characterized by the existence of multiple minima and maxima, as well as first-, second-, and higher-order saddle points. During the last decade we have experienced a rapid development of new methods for deterministic global optimization as well as the application of available global optimization algorithms in important engineering fields [1,2,9,10,11,12,13,14]. Recently, in order to locate the global solutions to the nonconvex phase stability analysis problems [3,4,5], a quadratic underestimation function based branch-and-bound algorithm, i. e., QBB, was developed for twice-differentiable nonlinear programs (NLPs) in terms of the simplicial partition of the constrained region [6,7].

Formulation

The nonconvex optimization problem considered in this section can be formulated as

$$(\mathrm{P}) \left| \begin{array}{ll} \min\limits_{\mathbf{x}} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \le 0 \quad i = 1, \ldots, m, \\ & \mathbf{x} \in \mathbf{S}^0 \subset \Re^n, \end{array} \right.$$

where f and $\mathbf{g}_i$ belong to C^2, the set of twice-differentiable functions, and $\mathbf{S}^0$ is a simplex defined by

$$\mathbf{S}^0 = \left\{ \mathbf{x} \in \Re^n : \mathbf{x} = \sum_{i=1}^{n+1} \lambda_i \mathbf{V}^i,\ \lambda_i \ge 0,\ \sum_{i=1}^{n+1} \lambda_i = 1 \right\},$$

where $\mathbf{V}^i \in \mathbf{V} \subset \Re^n$, $i = 1, 2, \ldots, n+1$ are the $n+1$ vertices of the simplex $\mathbf{S}^0$, and $\mathbf{V}$ is the set of its vertices. Let D_g be a subset of $\Re^n$ defined by

$$D_g = \{ \mathbf{x} \in \Re^n : \mathbf{g}_i(\mathbf{x}) \le 0,\ i = 1, 2, \ldots, m \}.$$

In general, the set D_g is nonconvex and even disconnected. We assume throughout this section that problem (P) has an optimal solution, unless otherwise stated. For any nonconvex optimization problem, i. e., (P), the QBB algorithm proposed in this section belongs

to a branch-and-bound scheme. During each iteration of this framework, a branching step and a bounding step must be finished simultaneously.

Simplicial Partition

For the branching procedure, the simplex $\mathbf{S}^0$ will be divided into refined subregions by using simplicial partition. For such kind of branching, it is a simple matter to check that for every $i \in I$, where I is the vertex set of $\mathbf{S}^0$, the points $\mathbf{V}^1, \ldots, \mathbf{V}^{i\text{-}1}, \mathbf{U}, \mathbf{V}^{i+1}, \ldots, \mathbf{V}^{n+1}$ are vertices of a simplex $\mathbf{S}_i \subset \mathbf{S}$, $\mathbf{S}$ is the current simplex, and that

$$(\text{int } \mathbf{S}_i) \cap \left(\text{int } \mathbf{S}_j\right) = \phi \quad \forall j \neq i\,; \quad \bigcup_{i \in I} \mathbf{S}_i = \mathbf{S}\,.$$

Then, the simplexes $\mathbf{S}_i$, $i \in I$, form a subdivision of the simplex $\mathbf{S}$ via $\mathbf{U}$. Each $\mathbf{S}_i$ will be referred to as a subsimplex of $\mathbf{S}$. An important special case is the bisection where $\mathbf{U}$ is a point of the longest edge of the simplex $\mathbf{S}$, for example, $\mathbf{U} \in [\mathbf{V}^m, \mathbf{V}^n]$, i. e.

$$\|\mathbf{V}^m - \mathbf{V}^n\| = \max_{\substack{i<j \\ i,\, j=1,\ldots n+1}} \left\{\left\|\mathbf{V}^i - \mathbf{V}^j\right\|\right\}\,,$$

where $\|\cdot\|$ denotes any given norm in $\Re^n$, and $\mathbf{U} = a\mathbf{V}^m + (1-a)\mathbf{V}^n$ with $0 < a \leq 1/2$. Adjiman et al. [9] proved that this simplicial bisection is exhaustive since $\delta(\mathbf{S}_k) \to 0$ as $k \to +\infty$.

Quadratic Underestimation Function for General Non-convex Structures

In the bounding step of a branch-and-bound algorithm, a lower bound is always obtained by constructing a valid convex underestimation problem for the original one appearing in the problem (P), and solving the relaxed convex NLP to global optimality. For the current simplex given by

$$\mathbf{S} = \left\{ x \in \Re^n : x = \sum_{i=1}^{n+1} \lambda_i \mathbf{V}^i,\ \lambda_i \geq 0,\ \sum_{i=1}^{n+1} \lambda_i = 1 \right\}\,, \tag{1}$$

where $\mathbf{V}^i \in \mathbf{V} \subset \Re^n$, $i = 1, 2, \ldots, n+1$ are the $n+1$ vertices of the current simplex $\mathbf{S}$, and $\mathbf{V}$ is the set of these vertices. Then, we intend to compute a lower bound $\mu(S)$ of the objective function f on $\mathbf{S} \cap D_g$. In other words, we compute a lower bound for the optimal value of the problem

$$(P(S)) \left| \begin{array}{ll} \min\limits_{\mathbf{x}} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq 0 \quad i = 1, \ldots, m\,, \\ \mathbf{x} \in \mathbf{S} \subset \Re^n\,. & \end{array} \right.$$

As mentioned above, f and g_i are generic nonconvex functions belonging to C^2, then the main idea for computing a lower bound $\mu(S)$ is to construct from problem (P(S)) a convex problem by replacing all those nonconvex functions with their respective convex underestimation functions, then solving the resulting relaxed convex problem. In order to achieve this, we see the following definition:

Definition 1 Given any nonconvex function $f(\mathbf{x})\colon S \to \Re$, $\mathbf{x} \in S \subseteq \Re^n$ belonging to C^2, the following quadratic function is defined by

$$F(\mathbf{x}) = \sum_{i=1}^{n} a_i \mathbf{x}_i^2 + \sum_{i=1}^{n} b_i \mathbf{x}_i + c\,, \tag{2}$$

where $\mathbf{x} \in \mathbf{S} \subseteq \Re^n$ and $F(\mathbf{x}) = f(\mathbf{x})$ holds at all vertices of $\mathbf{S}$. The a_i's are nonnegative scalars and are large enough such that $F(\mathbf{x}) \leq f(\mathbf{x})$, $\forall \mathbf{x} \in \mathbf{S}$.

It is trivial to see that $F(\mathbf{x})$ is convex since all quadratic coefficients, i. e., a_i's , are nonnegative. Theorem 2.2.1 in [7] ensures that $F(\mathbf{x})$ defined by Definition 1 is a convex underestimator of $f(\mathbf{x})$ if the difference function between them, i. e., $D(\mathbf{x}) = F(\mathbf{x}) - f(\mathbf{x})$, is a convex function. It is well known that $D(\mathbf{x})$ is convex if and only if its Hessian matrix $H_D(\mathbf{x})$ is positive semidefinite in the current simplex. A useful convexity condition is derived by noting that $H_D(\mathbf{x})$ is related directly to the Hessian matrix $H_f(\mathbf{x})$ of $f(\mathbf{x})$, $\mathbf{x} \in \mathbf{S}$ by the following equation:

$$\mathbf{H}_D(\mathbf{x}) = 2\Delta - \mathbf{H}_f(\mathbf{x})\,,$$

where Δ is a diagonal matrix whose diagonal elements are a_i's defined in Definition 1. Analogous to the "*diagonal shift matrix*" defined in [9], Δ here is referred to as the *diagonal underestimation matrix*, since these parameters guarantee that $F(\mathbf{x})$ defined by Eq. 2 is a rigorous underestimator of the generic nonconvex function $f(\mathbf{x})$. $D(x)$, as defined, is convex if and only if $2\Delta - H_f(\mathbf{x}) = 2\text{diag}(a_i) - H_f(\mathbf{x})$ is positive semidefinite for all $\mathbf{x} \in \mathbf{S}$.

In order to simplify the parameter calculation, the underestimator $F(\mathbf{x})$ is reformulated by using a single nonnegative a value, as follows:

$$F(\mathbf{x}) = a\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} b_i x_i + c\,. \tag{3}$$

Then, all diagonal elements of the diagonal underestimation matrix Δ are therefore equal to the uniform quadratic coefficient a defined by Eq. 3. Some interval arithmetic approaches are provided in [5,7] to estimate the quadratic coefficients with theoretical guarantee in the current simplex.

After the quadratic coefficients have been identified, the linear and constant coefficients of $F(\mathbf{x})$ defined by Eqs. 2 or 3, i. e., b_i's and c, can be given by the quadratic coefficients a_i's and the current simplex. In view of Definition 1, we know $F(\mathbf{x}) = f(\mathbf{x})$ holds at all vertices of $\mathbf{S}$, so the following linear equation group can be obtained as

$$\mathbf{V}^{k^T}\Delta\mathbf{V}^k + \mathbf{b}^T\mathbf{V}^k + c = f\left(\mathbf{V}^k\right) \quad k = 1, \ldots, n+1,$$

where $\Delta \in \Re^{n\times n}$ is the diagonal underestimation matrix whose diagonal elements are the quadratic term coefficients, a_i's defined in Eqs. 2 or 3. $\mathbf{b} \in \Re^n$ is the linear coefficient vector whose elements are b_i's defined in Eqs. 2 or 3, and c is a scalar:

$$\mathbf{b}^T\mathbf{V}^k + c = f\left(\mathbf{V}^k\right) - \mathbf{V}^{k^T}\Delta\mathbf{V} \quad k = 1, \ldots, n+1\,.$$

The vector $\mathbf{b} \in \Re^n$ is augmented as $(\mathbf{b},\, c) \in \Re^{n+1}$, in order to include the scalar c. In the same way, the matrix $\mathbf{V} \in \Re^{(n+1)\times n}$ is augmented as $(\mathbf{V},\, \mathbf{1}) \in \Re^{(n+1)\times(n+1)}$, where $\mathbf{1}$ is a column unity matrix of $\Re^n$. $(\mathbf{V},\, \mathbf{1}) \in \Re^{(n+1)\times(n+1)}$ is a regular square matrix since $\mathbf{V} \in \Re^{(n+1)\times n}$ is the coordinate matrix of the simplex which is linearly independent. Then we have

$$(\mathbf{b},\, c)^T = (\mathbf{V},\, \mathbf{1})^{-1}\left[f(\mathbf{V}) - \mathbf{V}^T\Delta\mathbf{V}\right]\,,$$

where $[f(\mathbf{V}) - \mathbf{V}^T\Delta\mathbf{V}] \in \Re^{n+1}$ is a column vector for the $n+1$ vertices of the current simplex. By virtue of this equation, it is obvious that the linear and constant coefficients defined by Eqs. 2 or 3 are determined uniquely by the quadratic coefficients and the current simplex.

By replacing all the nonconvex functions in problem (P(S)) with their corresponding quadratic function based convex underestimators described by Eq. 3, we have the following relaxed convex programming problem (QP(S)):

$$(QP(S)) \left| \begin{array}{ll} \min\limits_{\mathbf{x}} & F(\mathbf{x}) \\ \text{subject to} & G_i(\mathbf{x}) \le 0 \quad i = 1, \ldots, m\,, \\ & \mathbf{x} \in \mathbf{S} \subset \Re^n\,, \end{array} \right.$$

where

$$F(\mathbf{x}) = \sum_{i=1}^{n} a_i^f x_i^2 + \sum_{i=1}^{n} b_i^f x_i + c^f\,,$$

$$\mathbf{G}_j(\mathbf{x}) = \sum_{i=1}^{n} a_i^{g_j} x_i^2 + \sum_{i=1}^{n} b_i^{g_j} x_i + c^{g_j}$$

$$j = 1, 2, \ldots, m\,.$$

Let $D_{\mathbf{G}}$ be a subset of $\Re^n$ defined by

$$D_{\mathbf{G}} = \{x \in \Re^n : \mathbf{G}_i(\mathbf{x}) \le 0\,,\ i = 1, 2, \ldots, m\}\,.$$

Obviously, the set $D_{\mathbf{G}}$ is convex and compact. It should be noted that only additional $m+1$ quadratic parameters, i. e., a^f and a^{g_i} for $i = 1, 2, \ldots, m$, are introduced during the above transformation process if the uniform underestimation function is used, since all other linear and constant coefficients can be calculated by those quadratic parameters and the current simplex.

QBB Underestimators for Special Structures

For the concave function structure, denoted by $f^{CA}(\mathbf{x})$, whose eigenvalues are all nonpositive, i. e., $\lambda_{i,\,\mathbf{x}\in\mathbf{S}}(\mathbf{x}) \le 0$. Then, the quadratic coefficient of its underestimator defined by Eq. 2 is zero, so the valid lower bound of the concave function structure over the current simplex is a linear function. In fact, the valid bound constructed by Eq. 2 is equivalent to the convex envelope of the concave function over a simplex [7]. Let $\mathbf{S}$ be a simplex generated by the vertices $\mathbf{V}^1, \mathbf{V}^2, \ldots, \mathbf{V}^{n+1}$, i. e., $\mathbf{S} = \{\mathbf{x} \in \Re^n : \mathbf{x} = \sum_{i=1}^{n+1}\lambda_i\mathbf{V}^i,\ \lambda_i \ge 0,\ \sum_{i=1}^{n+1}\lambda_i = 1\}$, and let $f^{CA}(\mathbf{x})$ be a concave function defined on $\mathbf{S}$. Then the convex envelope of $f^{CA}(\mathbf{x})$ over $\mathbf{S}$ is the affine function $L^{CA}(\mathbf{x}) = \mathbf{b}^T\mathbf{x} + c$ that is uniquely determined by the system of linear equations $f^{CA}(\mathbf{V}^i) = \mathbf{b}^T\mathbf{V}^i + c$ for $i = 1, \ldots, n+1$.

For the general quadratic function presented by

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{q}^T\mathbf{x}$$

(note $\mathbf{H}_f(\mathbf{x}) = \mathbf{Q}$), we have the *diagonal underestimation matrix*, Δ, constructed on the basis of interval arithmetic [7], as

$$a = \max_i \left\{0,\ \frac{1}{2}\lambda_i^{\mathbf{Q}}\right\}$$

for the uniform case, and for the nonuniform case, we get

$$a_i = \max\left\{0,\ \frac{1}{2}\left(\mathbf{Q}_{ii} + \sum_{j\neq i}\left|\mathbf{Q}_{ij}\right|\right)\right\}\ .$$

Then, we have the quadratic underestimation function as

$$F(\mathbf{x}) = \mathbf{x}^T\Delta\mathbf{x} + \mathbf{b}^T\mathbf{x} + c\ ,$$

where the linear and constant coefficients, i. e., $(\mathbf{b}, c)$, can be determined uniquely by the quadratic coefficients calculated above and the current simplex.

Properties of the QBB Underestimator

For construction of the QBB underestimator, only quadratic coefficients need to be calculated since the linear and constant ones defined by Eqs. 2 or 3 can be determined uniquely by the quadratic coefficients and the current simplex. Another important property of the QBB algorithm is that the quadratic function based underestimator is always convex throughout the problem space. A potential benefit of this property is that it allows the convex solver applied to get the solution to the underestimator to have a feasible or an infeasible convergence path. Geometrically speaking, the QBB uses a convex quadratic function to approximate the convex part of a general nonconvex function directly, which can bypass the concave parts and avoid the overestimation for them.

Function Decomposition

It should be noted that the relaxed convex programming problem (QP(S)) can contain not only the quadratic underestimation functions for the generic nonconvex terms, but also the convex function terms which are not necessarily transformed into the quadratic underestimators. Then, the final underestimation strategy of the relaxed problem (QP(S)) can be slightly decomposed into the following convex programming formulation, as

$$\text{(QP(S)}')\quad \left|\begin{array}{ll} \min\limits_{\mathbf{x}} & F'(\mathbf{x}) \\ \text{subject to} & G_i'(\mathbf{x}) \le 0 \quad i = 1, \ldots, m\ , \\ & \mathbf{x} \in \mathbf{S} \subset \Re^n\ , \end{array}\right.$$

where

$$\begin{aligned} F'(\mathbf{x}) &= f^L(\mathbf{x}) + f^C(\mathbf{x}) + L_f^{CA}(\mathbf{x}) + F^{NC}(\mathbf{x})\ , \\ G_i'(\mathbf{x}) &= g_i^L(\mathbf{x}) + g_i^C(\mathbf{x}) + L_{g_i}^{CA}(\mathbf{x}) + G_i^{NC}(\mathbf{x}) \\ & \quad i = 1, 2, \ldots, m\ , \end{aligned}$$

and $f^L(\mathbf{x})$, $f^C(\mathbf{x}), L_f{}^{CA}(\mathbf{x}), g_i{}^L(\mathbf{x}), g_i{}^C(\mathbf{x})$, and $L_{g_i}^{CA}(\mathbf{x})$ represent the linear terms, convex terms, and the linear underestimation functions for the concave terms in the objective function and the constraints, respectively. $F^{NC}(\mathbf{x})$ and $G_i^{NC}(\mathbf{x})$ represent the quadratic convex underestimation functions for the generic nonconvex terms. Compared with the relaxed problem (QP(S)), the relaxed problem (QP(S)′) contains not only quadratic function terms, but also the generic convex terms of the original problem.

Algorithmic Procedure of QBB

At the start of this section, problem (P) is formulated over an initial simplex $\mathbf{S}^0$ which can be easily obtained by using an outer approximation approach. Now, we are in a position to present the proposed algorithm for solving problem (P) by using the basic operations described in previous sections.

Step 1 – Initialization. A convergence tolerance, ε_c, and a feasibility tolerance, ε_f, are selected and the iteration counter k is set to be zero. The global lower and upper bounds μ_0 and γ_0 of the global minimum of problem (P) are initialized and an initial current point $\mathbf{x}^{k,c}$ is randomly selected.

Step 2 – Local solution of problem (P) and update of upper bound. The nonconvex and nonlinear optimization problem (P) is solved locally within the current simplex $\mathbf{S}$. If the solution f_{local}^k of problem (P) is ε_f-feasible, the upper bound γ_k is updated as $\gamma_k = \min(\gamma_k, f_{\text{local}}^k)$.

Step 3 – Partitioning of the simplex. The current simplex, $\mathbf{S}^k$, is partitioned into the following two sim-

plexes ($r = 1, 2$):

$$\mathbf{S}^{k,1} = \left(\mathbf{V}^{k,0}, \ldots, \mathbf{V}^{k,m}, \ldots, \frac{\mathbf{V}^{k,m} + \mathbf{V}^{k,l}}{2}, \mathbf{V}^{k,n}\right),$$
$$\mathbf{S}^{k,2} = \left(\mathbf{V}^{k,0}, \ldots, \frac{\mathbf{V}^{k,m} + \mathbf{V}^{k,l}}{2}, \ldots, \mathbf{V}^{k,l}, \mathbf{V}^{k,n}\right),$$

where, k,m and k,l correspond to the vertices with the longest edge in the current simplex, i. e., $(k,\ m)$, $(k,\ l) = \arg\max_{i<j} \left\{\left\|\mathbf{V}^{k,j} - \mathbf{V}^{k,i}\right\|\right\}$.

Step 4 – Update of $a^r_{k,f}$, $b^r_{k,f}$, $c^r_{k,f}$ and a^r_{k,g_i}, b^r_{k,g_i}, c^r_{k,g_i} inside both subsimplexes $r = 1,\ 2$. The nonnegative parameters $a^r_{k,f}$ and a^r_{k,g_i} of the general nonconvex terms in the objective function and the constraints are updated inside both simplexes $r = 1,\ 2$ according to the methods presented in former sections, and the corresponding linear and constant coefficients, i. e., $b^r_{k,f}$, $c^r_{k,f}$ and b^r_{k,g_i}, c^r_{k,g_i}, are renewed accordingly.

Step 5 – Solutions inside both subsimplexes $r = 1,\ 2$. The convex programming problem (QP(S)′) is solved inside both subsimplexes ($r = 1,\ 2$) by using some nonlinear programming solver. If a solution $F^{k,r}_{\text{sol}}$ is feasible and less than the current upper bound, γ_k, then it is stored along with the solution point $\mathbf{x}^{k,r}_{\text{sol}}$.

Step 6 – Update iteration counter k and lower bound μ_k. The iteration counter increases by 1,

$$k \leftarrow k + 1\ ,$$

and the lower bound μ_k is updated to the minimum solution over the stored ones from the previous iterations. Furthermore, the selected solution is erased from the stored set:

$$\mu_k = F^{k',r'}_{\text{sol}}\ ,$$

where, $F^{k',r'}_{\text{sol}} = \min_{r,I}\{F^{I,r}_{\text{sol}},\ r = 1,\ 2,\ I = 1, \ldots, k-1\}$. If the set I is empty, set $\mu_k = \gamma_k$ and go to step 8.

Step 7 – Update the current point $\mathbf{x}^{k,c}$ and the current simplex $\mathbf{S}^{k}$. The current point is selected to be the solution point of the previously found minimum solution in step 6,

$$\mathbf{x}^{k,c} = \mathbf{x}^{I',r'}_{\text{sol}}\ ,$$

and the current simplex becomes the subsimplex containing the previously found solution,

$$\mathbf{S}^{k} = \left(\mathbf{V}^{k',0}, \ldots, \mathbf{V}^{k',m}, \ldots, \frac{\mathbf{V}^{k',m} + \mathbf{V}^{k',l}}{2}, \ldots, \mathbf{V}^{k',n}\right), \quad \text{if } r' = 1\ ,$$
$$\mathbf{S}^{k} = \left(\mathbf{V}^{k',0}, \ldots, \frac{\mathbf{V}^{k',m} + \mathbf{V}^{k',l}}{2}, \ldots, \mathbf{V}^{k',l}, \ldots, \mathbf{V}^{k',n}\right), \quad \text{otherwise.}$$

Step 8 – Check for convergence. If $(\gamma_k - \mu_k) > \varepsilon_c$, then return to step 2. Otherwise, ε_c-convergence has been reached. The global minimum solution and the solution point are given as

$$f^* \leftarrow f^{c,k''}\ ,$$
$$\mathbf{x}^* \leftarrow \mathbf{x}^{c,k''}\ ,$$

where, $k'' = \arg_I \left\{f^{c,I} = \gamma_k\right\},\ I = 1, \ldots, k$.

Conclusion

The QBB algorithm is guaranteed to identify the global optimum solution of problems belonging to the broad class of twice-differentiable NLPs. For any such problem, the ability to generate progressively tighter convex lower bounding problems in a branch-and-bound framework guarantees the convergence of this algorithm to within ε of the global optimum solution under the exhaustive simplicial partition of the initial simplex. Different methods [7] have been developed for the construction of the convex valid underestimators for special function structures and the generic nonconvex function structures, where the maximal eigenvalue analysis of the interval Hessian matrix provides the rigorous guarantee for the QBB algorithm to converge to the global solution.

References

1. Grossmann IE (ed) (1996) Global Optimization in Engineering Design, Kluwer book series in nonconvex optimization and its applications. Kluwer, Dordrecht
2. Floudas CA (2000) Deterministic Global Optimization: Theory, Methods and Applications. Kluwer, Dordrecht
3. Zhu Y (1999) Calculation of Phase Equilibria Based on Global Stability Analysis, Ph.D Dissertation. Institute of Process Engineering, Chinese Academy of Sciences, Beijing

4. Zhu Y, Xu Z (1999) Calculation of liquid-liquid equilibrium based on the global stability analysis for ternary mixtures by using a novel branch and bound algorithm: Application to UNIQUAC equation. Ind Eng Chem Res 38:3549–3556
5. Zhu Y, Inoue K (2001) Calculation of chemical and phase equilibrium based on stability analysis by QBB algorithm: Application to NRTL equation. Chem Eng Sci 56:6915–6931
6. Zhu Y, Kuno T (2003) Global optimization of nonconvex MINLP by a hybrid branch-and-bound and revised general Benders decomposition method. Ind Eng Chem Res 42:528–539
7. Zhu Y, Kuno T (2005) A global optimization method, QBB, for twice-differentiable nonconvex optimization problem. J Global Optim 33:435–464
8. Horst R, Tuy H (1990) Global optimization: deterministic approaches. Springer, Berlin
9. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, aBB, for general twice-differentiable constrained NLPs - I. Theoretical advances. Comput Chem Eng 22:1137–1158
10. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, aBB, for general twice-differentiable constrained NLPs - II. Implementation and computational results. Comput Chem Eng 22:1159–1179
11. Adjiman CS, Androulakis IP, Floudas CA (2000) Global optimization of mixed integer nonlinear problems. AIChE J 46:1769–1797
12. Akrotirianakis IG, Floudas CA (2004) A new class of improved convex underestimators for twice continuously differentiable constrained NLPs. J Global Optim 30:367–390
13. Akrotirianakis IG, Floudas CA (2004) Computational experience with a new class of convex underestimators: Box constrained NLP problems. J Global Optim 29:249–264
14. Meyer CA, Floudas CA (2005) Convex underestimation of twice continuously differentiable functions by piecewise quadratic perturbation: Spline aBB underestimators. J Global Optim 32:221–258

QR Factorization

Udaya Bhaskar Vemulapati
School of Computer Sci., University Central Florida, Orlando, USA

MSC2000: 65F25, 15A23, 65F05, 65F20, 65F22

Article Outline

Keywords
Properties of Orthogonal Transforms
Householder Transformations
QR Factorization Using Householder Transformations
Givens Rotations
Fast Givens Transformations
See also
References

Keywords

Orthogonal factorization; QR decomposition; Complete orthogonal factorization

QR factorization is a process of reducing a square (rectangular) matrix into upper triangular (upper trapezoidal) form by applying a series of *elementary orthogonal transformations*.

Properties of Orthogonal Transforms

Orthogonal transforms are where the transformation matrices are orthogonal. Orthogonal matrices are square matrices where each column is a unit vector and each column is mutually orthogonal to every other column. This implies that $Q \in \mathbf{R}^{n\times n}$ is orthogonal if and only if $Q^{\mathsf{T}} Q = QQ^{\mathsf{T}} = I$ (i. e. the transpose of an orthogonal matrix is its inverse). Orthogonal transformations are invariant under the 2-norm; i. e. $\| Qx \|_2 = \| x \|_2$. More details can be found in [8]. There are two popular orthogonal transformations: Householder and Givens.

Householder Transformations

These are named after A.S. Householder, who popularized their use in matrix computations. However, the properties of these matrices have been known for quite some time. For any nonzero $v \in \mathbf{R}^n$, a matrix H of the form

$$H = I - 2\frac{vv^{\top}}{v^{\top}v}$$

is called *Householder transformation*. It is easy to verify that H is symmetric, and orthogonal (which also means that it is its own inverse). Identity matrices are not Householder matrices. Geometrically, Householder matrices merely rotate a given vector in n-dimensions (without stretching or shrinking). Given any two vectors x and y such that $\| x \|_2 = \| y \|_2$, there exists

a Householder transformation H such that $Hx = y$ (it is easy to verify that $v = y - x$ satisfies this equation). Note that v completely characterizes H (in the sense that even though H is an $n \times n$ matrix, v is enough to reconstruct H, and to apply H). Also, scaling v by a scalar factor α will not change the transformation H.

QR Factorization Using Householder Transformations

Since Householder transformations rotate vectors in n-dimensions, they can be used to introduce zeroes selectively. Specifically, given any vector $x \neq 0 \in \mathbf{R}^n$, one can construct a Householder matrix H such that Hx is a multiple of e_1 (the first column of the identity matrix), i. e. make everything except the first row of Hx zero. Geometrically, this amounts rotating the vector such that it is parallel to the principal axis. It is easy to see that such H has the form $H = \mathrm{I} - 2vv^{\mathsf{T}}\ v^{\mathsf{T}}\ v$ where $v = x \pm \alpha\ e_1$ and $\alpha = \| \mathrm{x} \|_2$. In order to avoid subtracting close numbers (while dealing with floating point arithmetic), v is often chosen as $v = x + \mathrm{sign}(\xi_1)\ \alpha\ e_1$, where ξ_1 is the first element of x.

The following function House will compute the vector v, given x, that characterizes H so that $H = I - 2vv^{\mathsf{T}}/v^{\mathsf{T}}v$ and that $Hx = -\alpha\ e_1$. Also, v is scaled such that $v(1) = 1$, as the scaling does not affect H (using a notation similar to MATLAB [5]).

To apply H to a vector y, note that

$$Hy = \left(I - 2\frac{vv^{\top}}{v^{\top}v}\right) y = y - 2\frac{v(v^{\top}y)}{v^{\top}v} = y - 2\frac{v^{\top}y}{v^{\top}v}v,$$

and hence, one can compute Hy without explicitly computing H. The same idea can be extended to applying H to a set of columns $C \in \mathbf{R}^{n \times k}$. Let us call that function row. House(v, C).

```
function: v = House(x)
    n = length(x);
    v(1) = x(1) + sign(x(1)) * norm(x, 2);
    v(2 : n) = x(2 : n)/v(1);
    v(1) = 1;
end;
```

Suppose $H_1 = \mathrm{House}(x)$ with x taken as the first column of a matrix $A \in \mathbf{R}^{m \times n}$. Then H_1A will have zeros on the first column below the first row. Then one can find $H_2' = \mathrm{House}(A(2 : m, 2))$ such that everything below the second row of the second column is zeroed. Effectively, applying H_2' to the lower $(m-1) \times (n-1)$ matrix is the same as applying

$$H_2 = \begin{array}{c} \\ 1 \\ m-1 \end{array} \begin{array}{c} \begin{array}{cc} 1 & m-1 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & H_2' \end{pmatrix} \end{array}$$

to A. Note that H_2 does not affect the first row and column of H_1A. If this process is continued by applying a sequence of Householder transformations to A, it is reduced to an upper-trapezoidal matrix R; i. e.

$$H_{n-1}H_{n-2}\cdots H_1A = R\,. \tag{1}$$

Since each H_i is orthogonal, the product $Q^{\mathsf{T}} = H_{n-1}\ H_{n-2}\cdots\ H_1$ is also orthogonal. Then rearranging the equation,

$$A = QR\,, \tag{2}$$

where Q is orthogonal and R is upper-trapezoidal. This form of factorization is called *QR factorization* (or *orthogonal factorization*). The following algorithm computes the QR factorization of a matrix A.

```
function: QR(A, m, n)
    for i = 1 : min(m, n),
        v = House(A(i : m, i));
        A(i + 1 : m, i) = v(2 : m − i + 1);
        A(i+1 : m, i+1 : n) = row.House(v, A(+1 : m,
            i + 1 : n));
    end;
end;
```

In the above algorithm, the essential components of the Householder vectors are stored right where the zeros are going to be introduced. Here are the different parts of a matrix A after the algorithm is applied (superscripts indicate how many times an entry has been modified):

$$\begin{pmatrix} a_{11}^1 & a_{12}^1 & a_{13}^1 & \cdots & a_{1n}^1 \\ v_{21} & a_{22}^2 & a_{23}^2 & \cdots & a_{2n}^2 \\ v_{31} & v_{32} & a_{33}^3 & \cdots & a_{3n}^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & v_{m3} & \cdots & v_{mn} \end{pmatrix}.$$

v_{ij} is the ith component of vector v produced by the above algorithm during the jth iteration. Note that the matrix Q is available in in the lower triangular portion of the matrix in a factored form.

Givens Rotations

These rotations are named after W. Givens; they are also referred to as *Jacobi iterations*. C.G. Jacobi devised a symmetric eigenvalue algorithm based on these transformations in 1846. Consider the following 2×2 matrix of the form

$$G(\theta) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

applied to a vector $x \in \mathbf{R}^2$. It is easy to see that $G^\mathsf{T}x$ is a mere rotation of x by an angle of θ in counterclockwise direction. Such transformations are called rotations and as such are orthogonal. A straightforward extension that applies to an n-vector is given by matrices of the following form

$$G(i,j,\theta) = \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & i & & j & & \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \end{array}$$

with $c^2 + s^2 = 1$. Here $G^\mathsf{T}(i,j,\theta)\,x$ is a rotation of $x \in \mathbf{R}^n$ by an angle θ in counterclockwise direction in the (i,j)-plane. It is easy to verify that $G^\mathsf{T}(i,j,\theta)$ only modifies the rows i, j of the vector that is applied to and the remaining entries are unaffected; i. e.

$$G^\top(i,j,\theta)x = \begin{cases} cx_i - sx_j & i\text{th component}, \\ sx_i + cx_j & j\text{th component}, \\ \text{unchanged} & \text{otherwise}. \end{cases}$$

Given any vector $x \in \mathbf{R}^n$, $G^\mathsf{T}(i,j,\theta)$ can be constructed such that only the rows i, j are affected and that x_j is zeroed. Solving the following equations

$$sx_i + cx_j = 0 \quad \text{and} \quad c^2 + s^2 = 1$$

will yield

$$c = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \quad s = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}}. \tag{3}$$

Let $G_{ij}^{(k)}$ denote the application of a Givens rotation that uses rows i and j and zeros A_{jk} entry. The first column below the first row can be zeroed using a sequence of Givens rotations such as

$$Q_1 = G_{1m}^{(1)} \cdots G_{12}^{(1)}.$$

Similarly, the second column can be zeroed below the diagonal by

$$Q_2 = G_{2m}^{(2)} \cdots G_{23}^{(2)}.$$

Repeating this process for each column, A is reduced to upper-trapezoidal form, as in

$$Q_n \cdots Q_1 A = R.$$

The beauty about using Givens rotations is that there are various ways of applying these rotations and yet getting the same final QR factorization. In fact, this fact can be exploited in parallel processing very effectively. Detailed parallel QR factorization algorithms can be found in [6,7] and [3].

Fast Givens Transformations

Fast Givens Transformations involve half the number of multiplications compared to Givens rotations and they can be used to zero without an explicit square root computation. They are also referred to as *square-root-free Givens transformations*. Details can be found in [1,2,4].

Finally, it can be shown that if A has full rank, then it has a unique QR factorization if we make the diagonal elements of R positive [8].

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- Cholesky Factorization
- Interval Linear Systems
- Large Scale Trust Region Problems
- Large Scale Unconstrained Optimization
- Linear Programming
- Orthogonal Triangularization

► Overdetermined Systems of Linear Equations
► Solving Large Scale and Sparse Semidefinite Programs
► Symmetric Systems of Linear Equations

References

1. Anda AA, Park H (1994) Fast plane rotations with dynamic scaling. SIAM J Matrix Anal Appl 15:162–174
2. Gentleman WM (1973) Least squares computations by Givens rotations without square roots. J Inst Math Appl 12:329–336
3. Golub GH, Van Loan CF (1997) Matrix computations. third Johns Hopkins Univ. Press, Baltimore
4. Hammarling S (1974) A note on modifications to the Givens plane rotation. J Inst Math Appl 13:215–218
5. Hanselman D, Littlefield B (1997) The student edition of MATLAB 5 user's guide. MathWorks, Orchard Hill Place
6. Pothen A, Jha S, Vemulapati U (1987) Orthogonal factorization on the hypercube. In: Heath MT (ed) Hypercube Multiprocessors 1987. SIAM, Philadelphia, pp 587–596
7. Sameh AH, Kuck DJ (1978) On stable parallel linear system solvers. J ACM 25(1):81–91
8. Stewart GW (1973) Introduction to matrix computations. Acad. Press, New York

Quadratic Assignment Problem

QAP

LEONIDAS PITSOULIS[1], PANOS M. PARDALOS[2]
[1] Princeton University, Princeton, USA
[2] Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 90C08, 90C11, 90C27, 90C57, 90C59

Article Outline

Keywords
Formulations
Linearizations
  Lawler's Linearization
  Kaufman–Broeckx Linearization
  Frieze–Yadegar Linearization
  Adams–Johnson Linearization
Complexity Issues
  Computational Complexity
  PLS-Complexity
  Asymptotic Behavior
  Polynomially Solvable Cases
Lower Bounds
  Gilmore–Lawler Type Lower Bounds
  Variance Reduction Lower Bounds
  Eigenvalue Based Lower Bounds
  Bounds Based on Semidefinite Relaxations
Exact Solution Methods
  Branch and Bound
  Traditional Cutting Plane Methods
  Polyhedral Cutting Planes
Heuristics
  Construction Methods
  Limited Enumeration Methods
  Improvement Methods
  Tabu Search
  Simulated Annealing
  Genetic Algorithms
  Greedy Randomized Adaptive Search Procedure
  Ant Systems
Related Problems
  Biquadratic Assignment Problem
  Multidimensional Assignment Problems
  Bottleneck QAP
  Other Problems Which Can Be Formulated As QAPs
QAP Problem Generators
Surveys and Books
See also
References

Keywords

Combinatorial optimization; Quadratic assignment problem; Nonlinear assignment problems; Facility location

The quadratic assignment problem (QAP) is a combinatorial optimization problem, that although there is a substantial amount of research devoted to it, it is still, up to this date, not well solvable in the sense that no exact algorithm can solve problems of size $n > 20$ in reasonable computational time. The QAP can be viewed as a natural extension of the linear assignment problem (LAP; cf. also ► Assignment and matching). Let $\mathcal{S}_n$ denote the set of all permutations $\phi{:}N \to N$, where $N = \{1, \ldots, n\} \in \mathbf{Z}^+$. Given a cost matrix $C = (c_{ij}) \in \mathbf{R}^{n \times n}$ we can formulate the LAP using permutations as:

$$\min_{\phi \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{\phi(i)\phi(j)} = \min_{\phi \in S_n} \sum_{i=1}^{n} c_{i\phi(i)} \,. \tag{1}$$

The general formulation of the QAP as introduced by E.L. Lawler in [88] is obtained by increasing the dimension of the cost array C:

$$\min_{\phi\in S_n} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} c_{\phi(i)\phi(j)\phi(k)\phi(l)} = \min_{\phi\in S_n} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij\phi(i)\phi(j)} \,. \quad (2)$$

Formulation (2) will be referred to as the general QAP, while an instance will be denoted by QAP(C). The most widely used formulation of the QAP, and its first appearance in the literature, is that of T.C. Koopmans and M.J. Beckmann [85] which is a special case of (2). Used as a mathematical model for the location of a set of indivisible economical activities, the formulation of Koopmans and Beckmann involves three $n \times n$ input matrices with real elements $F = (f_{ij})$, $D = (d_{kl})$ and $B = (b_{ik})$, where f_{ij} is the flow between the facility i and facility j, d_{kl} is the distance between the location k and location l, and b_{ik} is the cost of placing facility i at location k. The objective is to assign each facility to a location such that the total cost is minimized. The *Koopmans–Beckmann* QAP formulation is given as follows:

$$\min_{\phi\in S_n} \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}d_{\phi(i)\phi(j)} + \sum_{i=1}^{n} b_{i\phi(i)} \,. \quad (3)$$

In the context of facility location (cf. also ► Facilities layout problems) the matrices F and D are symmetric with zeros in the diagonal, and all the matrices are nonnegative. An instance of a QAP with input matrices F, D and B will be denoted by QAP(F, D, B), while we will denote an instance by QAP(F, D), if there is no linear term (i. e., $B = 0$). It can be seen that (3) is a special case of (2) by setting $c_{ijkl} = f_{ij}d_{kl}$ for all i, j, k, l with $i \neq j$ or $k \neq l$ and $c_{ikik} = f_{ii}d_{kk} + b_{ik}$, otherwise. In terms of computational complexity (cf. also ► Complexity theory; ► Computational complexity theory), S. Sahni and T. Gonzalez [129] have shown that the QAP is *NP*-hard and that even finding an approximate solution within some constant factor from the optimal solution cannot be done in polynomial time unless $P = NP$.

Formulations

The QAP can be formulated as the following 0-1 integer programming problem with quadratic objective function (hence the name 'quadratic assignment problem'):

$$\begin{cases} \min & \sum\limits_{\substack{i,j=1\\ i\neq k\, j\neq l}}^{n}\sum\limits_{k,l=1}^{n} c_{ijkl}x_{ik}x_{jl} + \sum\limits_{i,j=1}^{n} c_{ijij}x_{ij} \\ \text{s.t.} & \sum\limits_{i=1}^{n} x_{ij} = 1, \quad j = 1,\ldots,n, \\ & \sum\limits_{j=1}^{n} x_{ij} = 1, \quad i = 1,\ldots,n, \\ & x_{ij} \in \{0,1\}, \quad i,j = 1,\ldots,n. \end{cases} \quad (4)$$

The above formulation is a direct consequence of formulation (2), where the constraints imposed by the permutations are expressed algebraically. A QAP in Koopmans–Beckmann form can be formulated in a more compact way using the inner product between two matrices:

$$\begin{cases} \min & \langle F, XDX^{\top}\rangle + \langle B, X\rangle \\ \text{s.t.} & X \in \mathbf{X}_n, \end{cases} \quad (5)$$

where $\mathbf{X}_n$ is the set of all *permutation matrices* $X = (x_{ij})$ such that their elements satisfy the constraints in (4). In the objective function of (4), let the coefficients c_{ijkl} be the entries of an $n^2 \times n^2$ matrix S, such that c_{ijkl} is on row $(i-1)\,n + k$ and column $(j-1)n + l$. Now let $Q := S - \alpha I$, where I is the $(n^2 \times n^2)$ unit matrix and α is greater than the row norm $\|S\|_\infty$ of matrix S. The subtraction of a constant from the entries on the main diagonal of S does not change the optimal solutions of the corresponding QAP, it simply adds a constant to the objective function. Hence we can consider a QAP with coefficient array Q instead of S. Let $x = (x_{11}, \ldots, x_{1n}, x_{21}, \ldots, x_{nn})^{\mathsf{T}} = (x_1, \ldots, x_{nn})^{\mathsf{T}}$. Then we can rewrite the objective function of the QAP with array of coefficients Q as a quadratic form $x^{\mathsf{T}}\,Qx$, where it can be shown that Q is symmetric and negative definite. Therefore we have a quadratic concave minimization problem (cf. also ► Concave programming) and can formulate the

QAP as:

$$\begin{cases} \min & x^\top Q x \\ \text{s.t.} & \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \\ & \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n, \\ & x_{ij} \geq 0, \quad i, j = 1, \ldots, n, \end{cases} \tag{6}$$

The above formulation was introduced in [14], and was used to derive cutting plane procedures (cf. also ► Integer programming: Cutting plane algorithms). By adding the term αI to the matrix Q instead of subtracting it, we could always assume that the objective function of the QAP is convex. This leads to the formulation of the QAP as a quadratic convex minimization problem. The QAP can also be formulated using the trace of a matrix as:

$$\begin{cases} \min & \operatorname{tr}(FXD^\top + B)X^\top \\ \text{s.t.} & X \in \mathbf{X}_n. \end{cases} \tag{7}$$

The trace formulation of the QAP first appeared in [47], and was used in [51] to introduce eigenvalue lower bounding techniques for symmetric QAPs.

Let $\operatorname{vec}(X) \in \mathbf{R}^{n^2}$ be the vector formed by the columns of a permutation matrix X. The QAP can be formulated using the Kronecker product as

$$\begin{cases} \min & \operatorname{vec}(X)^\top (F \otimes D) \operatorname{vec}(X) \\ & \quad + \operatorname{vec}(B)^\top \operatorname{vec}(X) \\ \text{s.t.} & X \in \mathbf{X}_n. \end{cases} \tag{8}$$

Using the Kronecker product, Lawler [88] provided an alternative formulation for the QAP as an $n^2 \times n^2$ LAP. An $n^2 \times n^2$ matrix C is constructed from the n^4 costs c_{ijkl}, such that the $(ijkl)$th element corresponds to the $((i-1)\,n + k, (j-1)\,n + \mathrm{l})$th element of C. The QAP then is equivalent to an LAP of dimension n^2 with C as the cost matrix, and with the additional constraint that the $n^2 \times n^2$ permutation matrix which defines a feasible solution, must be the Kronecker product of two permutation matrices of dimension n. In other words the QAP is equivalent to

$$\begin{cases} \min & \langle C, Y \rangle \\ \text{s.t.} & Y = X \otimes X, \\ & X \in \mathbf{X}_n. \end{cases} \tag{9}$$

The resulting LAP however cannot be solved efficiently (i. e., in $O(n^6)$ time) because Y, although it is an $n^2 \times n^2$ permutation matrix, is constrained to have a special structure.

Linearizations

Linearization is a technique which involves the elimination of the nonlinear term in a given objective function, in order to make it linear, through the introduction of new variables and new linear (binary) constraints. The objective is to transform a 0–1 nonlinear integer program into a provably equivalent 0–1 linear integer program, such that existing methods for linear integer programs will provide a relaxed problem where lower bounds may be computed. Though there are several ways to linearize a given nonlinear integer program, it is desirable to have a linearization that will introduce the least amount of new variables and constraints. Moreover, the 'tightness' of the relaxation of the resulting linear integer program is very important.

The first attempt to devise solution techniques for solving the QAP had to do with the elimination of the quadratic term in the objective function of (4), in order to transform the problem into a 0–1 linear program. Four such linearizations of the QAP will be presented in this section. The first is due to Lawler [88], which is the first linearization suggested for the QAP, and the second by Kaufman and F. Broeckx [80] which is the smallest with regard to the number of new variables and constraints introduced. The third is a more recent (1983) one that is due to A.M. Frieze and J. Yadegar [56], which unifies most of the previous linearizations for the QAP, and is closely related to the fourth linearization presented in this section due to W.P. Adams and T.A. Johnson [2].

Lawler's Linearization

Lawler [88] replaces the quadratic terms $x_{ij}x_{kl}$ in the objective function of (4), with n^4 variables

$$y_{ijkl} := x_{ij}x_{kl}\,, \quad i, j, k, l = 1, \ldots, n\,,$$

which results in a 0–1 linear program of $n^4 + n^2$ binary variables and $n^4 + 2n^2 + 1$ constraints. More specifically, it is proved in [88] that the QAP is equivalent to the

following 0–1 linear program

$$\begin{cases} \min & \sum_{i,j=1}^{n}\sum_{k,l=1}^{n} c_{ijkl}\, y_{ijkl} \\ \text{s.t.} & (x_{ij}) \in \mathbf{X}_n, \\ & \sum_{i,j=1}^{n}\sum_{k,l=1}^{n} y_{ijkl} = n^2, \\ & x_{ij} + x_{kl} - 2y_{ijkl} \geq 0, \\ & y_{ijkl} \in \{0,1\}, \\ & \text{for } i,j,k,l = 1,\ldots,n. \end{cases}$$

Kaufman–Broeckx Linearization

Rearranging terms in the objective function (4) we obtain

$$\sum_{i,j=1}^{n} x_{ij} \sum_{k,l=1}^{n} c_{ijkl} x_{kl}\,.$$

Kaufman and Broeckx [80] defined n^2 new real variables

$$w_{ij} := x_{ij} \sum_{k,l=1}^{n} c_{ijkl} x_{kl}\,, \quad i,j = 1,\ldots,n\,,$$

resulting in an equivalent linear objective function

$$\sum_{i,j=1}^{n} w_{ij}\,.$$

Introducing n^2 constants $a_{ij} := \sum_{k,l=1}^{n} c_{ijkl}$ for $i, j = 1, \ldots, n$, the QAP becomes equivalent to the following mixed 0–1 linear program:

$$\begin{cases} \min & \sum_{i,j=1}^{n} w_{ij} \\ \text{s.t.} & (x_{ij}) \in \mathbf{X}_n, \\ & a_{ij}x_{ij} + \sum_{k,l=1}^{n} c_{ijkl} x_{kl} - w_{ij} \leq a_{ij}, \\ & w_{ij} \geq 0, \\ & i,j = 1,\ldots,n. \end{cases}$$

The above formulation employs n^2 new real variables, n^2 binary variables and $n^2 + 2n$ constraints. The elements c_{ijkl} are all assumed to be nonnegative, which is a valid assumption since the addition of a constant to each element will not affect the optimal solution. The proof of equivalence of the QAP to the above linear integer program can be found in [80].

Frieze–Yadegar Linearization

In [56] the products of the binary variables are replaced by continuous variables (i. e. $y_{ijkl} := x_{ij}x_{kl}$), and the QAP(C) is proved to be equivalent to the following mixed 0–1 linear program:

$$\begin{cases} \min & \sum_{i,j=1}^{n}\sum_{k,l=1}^{n} c_{ijkl}\, y_{ijkl} \\ \text{s.t.} & (x_{ij}) \in \mathbf{X}_n, \\ & \sum_{i=1}^{n} y_{ijkl} = x_{kl}, \quad \forall j,k,l, \\ & \sum_{j=1}^{n} y_{ijkl} = x_{kl}, \quad \forall i,k,l, \\ & \sum_{k=1}^{n} y_{ijkl} = x_{ij}, \quad \forall i,j,l, \\ & \sum_{l=1}^{n} y_{ijkl} = x_{ij}, \quad \forall i,j,k, \\ & y_{ijij} = x_{ij}, \quad \forall i,j, \\ & 0 \geq y_{ijkl} \leq 1, \quad \forall i,j,k,l, \end{cases} \tag{10}$$

where $i, j, k, l = 1, \ldots, n$. The above program has n^4 new real variables, n^2 binary variables, and $n^4 + 4n^3 + n^2 + 2n$ constraints. Note that the constraint $y_{ijij} = x_{ij}$ is redundant since it follows from the definition of the y_{ijkl} variables. Frieze and Yadegar considered a Lagrangian relaxation of the above 0–1 linear program, and established a relationship between the lower bounds derived by the solution of the relaxation, and the lower bounds derived from decomposition techniques applied to the Gilmore–Lawler bound for the QAP.

Adams–Johnson Linearization

Adams and Johnson presented in [2] a new 0–1 linear integer formulation for the QAP, which resembles the one of Frieze and Yadegar described previously. It is based on the general linearization technique for general 0–1 polynomial programs introduced by Adams and

H.D. Sherali [3,4]. The QAP(C) is proved to be equivalent to the following mixed 0–1 linear program:

$$\begin{cases} \min & \sum_{i,j=1}^{n} \sum_{k,l=1}^{n} c_{ijkl}\, y_{ijkl} \\ \text{s.t.} & (x_{ij}) \in \mathbf{X}_n, \\ & \sum_{i=1}^{n} y_{ijkl} = x_{kl}, \quad \forall j,k,l, \\ & \sum_{j=1}^{n} y_{ijkl} = x_{kl}, \quad \forall i,k,l, \\ & y_{ijkl} = y_{klij}, \quad \forall i,j,k,l, \\ & y_{ijkl} \geq 0, \quad \forall i,j,k,l, \end{cases} \tag{11}$$

where $i, j, k, l = 1, \ldots, n$, and each y_{ijkl} represents the product $x_{ij}x_{kl}$. The above formulation contains n^2 binary variables x_{ij}, n^4 continuous variables y_{ijkl}, and $n^4 + 2n^3 + 2n$ constraints excluding the nonnegativity constraints on the continuous variables. Although as noted in [3] a significant smaller formulation in terms of both the number of variables and constraints could be obtained, the structure of the relaxation of the above formulation is favorable for solving it. As noted in [2], the constraint set of the above relaxation describes a solution matrix Y which is the Kronecker product of two permutation matrices (i. e. $Y = X \otimes X$ where $X \in \mathbf{X}_n$), showing clearly the equivalence of the above formulation with the QAP as formulated in (9). The theoretical strength of the above linearization of the QAP lies on the fact that, as shown in [2] and [73], the constraints of the relaxations derived from all previous linearizations, can be expressed as a linear combination of the constraints of the continuous relaxation of the above linearization. Moreover, many of the previously published lower-bounding techniques, can be explained based on the dual-space of this relaxation.

Complexity Issues

The first two parts of this section bring evidence to the fact that the QAP is a 'very hard' problem from the theoretical point of view. Not only the QAP cannot be solved to optimality efficiently but it even cannot be approximated efficiently within some constant approximation ratio. Furthermore, finding local optima is not a trivial task even for simply structured neighborhoods like the 2-opt neighborhood. The asymptotic behavior of the QAP and polynomially solvable special cases of the QAP are mentioned in the last two parts of this section.

Computational Complexity

Two early results obtained by Sahni and Gonzalez [129] in 1976 settled the complexity of solving and approximating the QAP. It was shown that the QAP is *NP*-hard and that even finding an ϵ-approximate solution for the QAP is a hard problem, in the sense that the existence of a polynomial ϵ-approximation algorithm implies $P = NP$.

Theorem 1 [129] *The quadratic assignment problem is strongly NP-hard. For an arbitrary $\epsilon > 0$, the existence of a polynomial time ϵ-approximation algorithm for the QAP implies $P = NP$.*

The proof is done by a reduction from the Hamiltonian cycle problem [58].

M. Queyranne [121] derives an even stronger result which further confirms the widely spread belief on the inherent difficulty of the QAP in comparison with other difficult combinatorial optimization problems. It is well known and very easy to see that the traveling salesman problem (TSP) is a special case of the QAP. The TSP on n cities can be formulated as a QAP(F, D) where F is the distance matrix of the TSP instance and D is the adjacency matrix of a Hamiltonian cycle on n vertices. In the case that the distance matrix is symmetric and satisfies the triangle inequality, the TSP is approximable in polynomial time within 3/2 as shown in [37]. Queyranne [121] showed that, unless $P = NP$, QAP(A, B) is not approximable in polynomial time within some finite approximation ratio, even if A is the distance matrix of some set of points on a line and B is a symmetric block diagonal matrix.

A more recent result of S. Arora, Frieze and H. Kaplan [6] answers partially one of the open questions stated by Queyranne [121]. What happens if matrix A is the distance matrix of n points which are regularly spaced on a line, that is, points with abscissae given by $x_p = p$, $p = 1, \ldots, n$? This special case of the QAP is termed *linear arrangement problem* and is a well studied *NP*-hard problem. In the linear arrangement problem the matrix B is not restricted to have the block diagonal structure mentioned above, but is simply a symmetric 0–1 matrix. Arora, Frieze and Kaplan [6] give

a polynomial time approximation scheme (PTAS) for the linear arrangement problem in the case that the 0–1 matrix B is dense, that is, the number of '1' entries in B is in $\Omega(n^2)$, where n is the size of the problem. They show that for each $\epsilon > 0$ there exists an ϵ-approximation algorithm for the dense linear arrangement problem with time complexity depending polynomially on n and exponentially on $1/\epsilon$, hence polynomial for each fixed $\epsilon > 0$.

PLS-Complexity

Assume that an optimization problem P is given by specifying a ground set $\mathcal{E}$, a set $\mathcal{F} \subseteq 2^{\mathcal{E}}$ of feasible solutions and an objective function $f : \mathcal{F} \to \mathbf{R}$. A globally optimal solution $S^* \in \mathcal{F}$ of the problem P is defined as:

$$f(S^*) := \min_{S \in \mathcal{F}} f(S) .$$

For any given $S \in \mathcal{F}$ denote the *neighborhood* of S by $\mathcal{N}(S) \subset \mathcal{F}$. The neighborhood of S consists of other feasible solutions which are topologically 'close' to S. A *locally optimal solution* or a *local minimum* $\overline{S} \in \mathcal{F}$ of the problem P, given the neighborhood $\mathcal{N}$ is defined as:

$$f(\overline{S}) = \min_{S \in \mathcal{N}(\overline{S})} f(S) .$$

Recently (as of 1999) it has been shown that even finding a locally optimal solution for the QAP can be prohibitively hard, that is, even local search is hard in the case of the QAP. Consider the following question 'How easy it is to find a locally optimal solution for the QAP?' Since local optimality is defined through a specific neighborhood structure, the answer depends on the involved neighborhood structure. If the neighborhood $\mathcal{N}$ is replaced by new neighborhood $\mathcal{N}'$, one would generally expect changes in the local optimality status of a solution. The theoretical basis for facing this kind of problems was introduced by D.S. Johnson, C.H. Papadimitriou and Yannakakis [72]. They define the so-called *polynomial time local search problems*, shortly *PLS problems*. A pair $(P, \mathcal{N})$, where P is a (combinatorial) optimization problem and $\mathcal{N}$ is an associated well defined neighborhood structure, defines a local search problem in which the objective is to find a locally optimal solution of P with respect to the neighborhood structure $\mathcal{N}$. Without going into technical details a problem in the PLS class is a local search problem for which local optimality can be checked in polynomial time. In analogy with decision problems, there exist complete problems in the class of PLS problems. The PLS-complete problems, are – in the usual complexity sense – the most difficult among the PLS problems.

K.A. Murthy, Pardalos and Y. Li [103] introduce a neighborhood structure for the QAP which is similar to the neighborhood structure proposed by B.W. Kernighan and S. Lin [81] for the graph partitioning problem. For this reason we will call it a *K-L type neighborhood structure for the QAP*. Murthy, Pardalos and Li [103] show that the corresponding local search problem is PLS-complete. Consider a permutation $\phi_0 \in \mathcal{S}_n$. A swap of ϕ_0 is a permutation $\phi \in \mathcal{S}_n$ obtained from ϕ_0 by applying a transposition (i, j) to it, $\phi = \phi_0 \circ (i, j)$. A *transposition* (i, j) is defined as a permutation which maps i to j, j to i, and k to k for all $k \notin \{i, j\}$. A *greedy swap* of permutation ϕ_0 is a swap ϕ which maximizes $Z(F, D, \phi_0) - Z(F, D, \phi)$ over all swaps ϕ of ϕ_0, where $Z(F, D, \phi)$ is the objective function value of QAP(F, D) with permutation ϕ (see formulation (3)). Let $\phi_0, \ldots, \phi_l$ be a sequence of permutations in $\mathcal{S}_n$, each of them being a greedy swap of the preceding one. Such a sequence is called *monotone* if for all $k = 0, \ldots, l$, in the pair (ϕ_k, ϕ_{k+1}) where $\phi_k = \phi_{k-1} \circ (i_k, j_k)$ and $\phi_{k+1} = \phi_k \circ (i_{k+1}, j_{k+1})$, we have $\{i_k, j_k\} \cap \{i_{k+1}, j_{k+1}\} = \emptyset$. The *neighborhood* of ϕ_0 consists of all permutations which occur in the (unique) maximal monotone sequence of greedy swaps starting with permutation ϕ_0. Let us denote this neighborhood structure for the QAP by $\mathcal{N}_{K-L}$. It is not difficult to see that, given a QAP(F, D) of size n and a permutation $\phi \in \mathcal{S}_n$, the cardinality of $\mathcal{N}_{K-L}(\phi)$ does not exceed $\lfloor n/2 \rfloor + 1$. It is easily seen that the local search problem (QAP, $\mathcal{N}_{K-L}$) is a PLS problem. This result can be found in [112], where the authors show that the graph partitioning problem with the neighborhood structure defined in [81] is PLS-reducible to (QAP, $\mathcal{N}_{K-L}$).

Theorem 1 [112] *The local search problem (QAP, $\mathcal{N}_{K-L}$), where $\mathcal{N}_{K-L}$ is the Kernighan–Lin type neighborhood structure for the QAP, is PLS-complete.*

The PLS-completeness of (QAP, $\mathcal{N}_{K-L}$) implies that, in the worst case, a general local search algorithm as described above involving the Kernighan–Lin type neighborhood finds a local minimum only after a time which is exponential on the problem size. Numerical results,

however, show that such local search algorithms perform quite well when applied to QAP test instances, as reported in [103].

Another simple and frequently used neighborhood structure is the so-called *pair-exchange* (or *2-opt*) neighborhood $\mathcal{N}_2$. The pair-exchange neighborhood of a permutation $\phi_0 \in \mathcal{S}_n$ consists of all permutations $\phi \in \mathcal{S}_n$ obtained from ϕ_0 by applying some transposition (i, j) to it. Specifically,

$$\mathcal{N}_2(\phi) := \{\phi \circ (i, j):\ i, j = 1, \ldots, n,\ i \neq j\} \ .$$

It can also be shown that (QAP, $\mathcal{N}_2$) is PLS-complete. A.A. Schräffer and Yannakakis [130] have proven that the graph partitioning problem with a neighborhood structure analogous to $\mathcal{N}_2$ is PLS-complete. A similar PLS-reduction as in [112] implies that the local search problem (QAP, $\mathcal{N}_2$) is PLS-complete.

Finally, let us mention that no local criteria are known for deciding how good a locally optimal solution is as compared to a global one. From the complexity point of view, deciding whether a given local optimum is a globally optimal solution to a given instance of the QAP is a hard problem, see [108].

Asymptotic Behavior

Under certain probabilistic conditions on the coefficient matrices of the QAP, the ratio between its 'best' and 'worst' values of the objective function approaches 1, as the size of the problem approaches infinity. R.E. Burkard and U. Fincke [29] identify a common combinatorial property of a number of problems which, under certain probabilistic conditions on the problem data, behave as described above.

In an early work Burkard and Fincke [28] investigate the relative difference between the worst and the best values of the objective function for the Koopmans–Beckmann QAP. They first consider the case where the coefficient matrix D is the matrix of pairwise distances of points chosen independently and uniformly from the unit square in the plane. Then the general case where entries of the flow and distance matrices F and D are independent random variables taken from a uniform distribution on [0, 1] is considered. In both cases it is shown that the relative difference mentioned above approaches 0 with probability tending to 1 as the size of the problem tends to infinity.

Later Burkard and Fincke [29] consider the ratio between the objective function values corresponding to an optimal (or best) and a worst solution of a generic combinatorial optimization problem. They find that for each $\epsilon > 0$, the ratio between the best and the worst values of the objective function lies on $(1 - \epsilon, 1 + \epsilon)$, with probability tending to 1, as the size of the problem approaches infinity. Under additional combinatorial conditions, W. Szpankowski [132] strengthens this result and improves the range of the convergence to almost surely. In the almost sure convergence the probability that the above mentioned ratio tends to 1 is equal to 1. The asymptotic behavior of the QAP can be stated in the following theorem:

Theorem 3 *Consider a sequence of problems QAP($A^{(n)}$, $B^{(n)}$) for $n \in \mathbf{N}$, with $n \times n$ coefficient matrices $A^{(n)} = (a_{ij}^{(n)})$ and $B^{(n)} = (b_{ij}^{(n)})$. Assume that $a_{ij}^{(n)}$ and $b_{ij}^{(n)}$, $n \in \mathbf{N}$, $1 \leq i, j \leq n$, are independently distributed random variables on [0, M], where M is a positive constant. Moreover, assume that the entries $a_{ij}^{(n)}$, $n \in \mathbf{N}$, $1 \leq i, j \leq n$, have the same distribution, and the entries $b_{ij}^{(n)}$, $n \in \mathbf{N}$, $1 \leq i, j \leq n$, have also the same distribution, which does not necessarily coincide with that of $a_{ij}^{(n)}$. Furthermore, assume that these variables have finite expected values, variances and third moments. Let $\phi_{opt}^{(n)}$ and $\phi_{wor}^{(n)}$ denote an optimal and a worst solution of QAP($A^{(n)}$, $B^{(n)}$), respectively, that is,*

$$Z\left(A^{(n)}, B^{(n)}, \phi_{opt}^{(n)}\right) = \min_{\phi \in S_n} Z\left(A^{(n)}, B^{(n)}, \phi\right) \ ,$$

and

$$Z\left(A^{(n)}, B^{(n)}, \phi_{wor}^{(n)}\right) = \max_{\phi \in S_n} Z\left(A^{(n)}, B^{(n)}, \phi\right) \ .$$

Then the following equality holds almost surely:

$$\lim_{n \to \infty} \frac{Z\left(A^{(n)}, B^{(n)}, \phi_{opt}^{(n)}\right)}{Z\left(A^{(n)}, B^{(n)}, \phi_{wor}^{(n)}\right)} = 1 \ .$$

The above result suggests that the value of the objective function of the problem QAP($A^{(n)}$, $B^{(n)}$) (corresponding to an arbitrary feasible solution) gets somehow close to its expected value $n^2 \mathrm{E}(A)\mathrm{E}(B)$, as the size of the problem increases, where $\mathrm{E}(A)$ and $\mathrm{E}(B)$ are the expected values of $a_{ij}^{(n)}$ and $b_{ij}^{(n)}$, $n \in \mathbf{N}$, $1 \leq i, j \leq n$, respectively. J.C.B. Frenk, M. van Houweninge, and A.G. Rinnooy

Kan [54] and W.T. Rhee [126,127] provide different analytical evaluations for this 'getting close', by imposing different probabilistic conditions on the data.

Results on the asymptotic behavior of the QAP have been exploited by M.E. Dyer, Frieze, and C.J.H. McDiarmid [46] to analyze the performance of branch and bound algorithms for QAPs with randomly generated coefficients as described above. They have shown that for such QAPs the optimal value of the continuous relaxation of Frieze–Yadegar linearization as stated in (10), is in $O(n)$ with probability tending to 1 as the size n of the QAP tends to infinity. Hence the gap between the optimal value of this continuous relaxation and the optimal value of the QAP grows like $O(n)$ with probability tending to 1 as n tends to infinity.

Polynomially Solvable Cases

Since the QAP is *NP*-hard, restricted versions which can be solved in polynomial time are an interesting aspect of the problem. A basic question arising with respect to polynomially solvable versions is the identification of those versions and the investigation of the border line between hard and easy versions of the problem. There are two ways to approach this topic: first, find structural conditions to be imposed on the coefficient matrices of the QAP so as to obtain polynomially solvable versions, and secondly, investigate other combinatorial optimization or graph-theoretical problems which can be formulated as QAPs, and embed the polynomially solvable versions of the former into special cases of the later. These two approaches yield two groups of restricted QAPs which are briefly reviewed in this section. For a detailed information on this topic, see [35].

Most of the restricted versions of the QAP with specially structured matrices involve Monge matrices or other matrices having analogous properties. A matrix $A = (a_{ij})$ is a *Monge matrix* if and only if the following *Monge inequalities* are fulfilled for each 4-tuples of indices $i, j, k, l, i < k, j < l$:

$$a_{ij} + a_{kl} \leq a_{il} + a_{kj} .$$

A matrix $A = (a_{ij})$ is an *anti-Monge matrix* if and only if the following *anti-Monge inequalities* are fulfilled for each 4-tuples of indices $i, j, k, l, i < k, j < l$:

$$a_{ij} + a_{kl} \geq a_{il} + a_{kj} .$$

A simple example of Monge and anti-Monge matrices are the *sum matrices*; the entries of a sum matrix matrix $A = (a_{ij})$ are given as $a_{ij} = \alpha_i + \beta_j$, where (α_i) and (β_j) are the generating row and column vector, respectively. A *product matrix A* is defined in an analogous way: its entries are given as $a_{ij} = \alpha_i \beta_j$, where (α_i), (β_j) are the generating vectors. If the row generating vector (α_i) and the column generating vectors (β_i) are sorted nondecreasingly, then the product matrix $(\alpha_i \beta_j)$ is an anti-Monge matrix.

In contrast with the traveling salesman problem, it turns out that the QAP with both coefficient matrices being Monge or anti-Monge is *NP*-hard, whereas the complexity of a QAP with one coefficient matrix being Monge and the other one being anti-Monge is still open, see [23] and [35]. However, some polynomially solvable special cases can be obtained by imposing additional conditions on the coefficient matrices. These special cases involve very simple matrices like product matrices or so-called *chess-board matrices*. A matrix $A = (a_{ij})$ is a chess-board matrix if its entries are given by $a_{ij} = (-1)^{i+j}$. These QAPs can either be formulated as equivalent LAPs, or they are *constant permutation QAPs* (see [23,35]), that is, their optimal solution can be given before hand, without knowing the entries of their coefficient matrices. A few other versions of the QAP involving Monge and anti-Monge matrices with additional structural properties can be solved by dynamic programming. Other restricted versions of the QAP involve matrices with a specific diagonal structure such as *circulant* and *Toeplitz matrices*. An $n \times n$ matrix $A = (a_{ij})$ is called a Toeplitz matrix if there exist numbers $c_{-n+1}, \ldots, c_{-1}, c_0, c_1, \ldots, c_{n-1}$ such that $a_{ij} = c_{j-i}$, for all i, j. A matrix A is called a circulant matrix if it is a Toeplitz matrix and the generating numbers c_i fulfill the conditions $c_i = c_{n-i}$, for $0 \leq i \leq n-1$. In other words, a Toeplitz matrix has constant entries along lines parallel to the diagonal, whereas a circulant is given by its first row and the entries of the $i-$ th row resembles the first row shifted by $i-1$ places to the right.

In general versions of the QAP with one anti-Monge (Monge) matrix and one Toeplitz (circulant) matrix, remain *NP*-hard unless additional conditions, such as monotonicity, are imposed on the coefficient matrices. A well studied problem is the so called anti-

Monge–Toeplitz QAP where the rows and columns of the anti-Monge matrix are nondecreasing, investigated in [26]. It has been shown that this problem is *NP*-hard and contains as a special case the so-called turbine balancing problem (TBP) introduced in [99] and formulated as a QAP in [87]. In the TBP we are given a number of blades to be welded in regular spacing around the cylinder of the turbine. Due to inaccuracies in the manufacturing process the weights of the blades differ slightly and consequently the gravity center of the system does not lie on the rotation axis of the cylinder, leading to instabilities. In an effort to make the system as stable as possible, it is desirable to locate the blades so as to minimize the distance between the center of gravity and the rotation axis. The mathematical formulation of this problem leads to an *NP*-hard anti-Monge–Toeplitz QAP. (For more details and for a proof of *NP*-hardness see [26].) It is probably interesting that the maximization version of this problem is polynomially solvable. Further polynomially solvable special cases of the anti-Monge–Toeplitz QAP arise if additional constraints such as benevolence or k-benevolence are imposed on the Toeplitz matrix. These conditions are expressed in terms of properties of the generating function of these matrices, see [26]. The polynomially solvable QAPs with one anti-Monge (Monge) matrix and the other one Toeplitz (circulant) matrix described above, are all constant permutation QAPs. The techniques used to prove this fact and to identify the optimal permutation is called reduction to extremal rays. This technique exploits two facts: first, the involved matrix classes form cones, and secondly, the objective function of the QAP is linear with respect to each of the coefficient matrices. These two facts allow us to restrict the investigations to instances of the QAP with 0–1 coefficient matrices which are extremal rays of the above mentioned cones. Such instances can then be handled by elementary means (exchange arguments, bounding techniques) more easily that the general given QAP. The identification of polynomially solvable special cases of the QAP which are not constant permutation QAPs and can be solved algorithmically remains a challenging open question.

Another class of matrices similar to the Monge matrices are the *Kalmanson matrices*. A matrix $A = (a_{ij})$ is a Kalmanson matrix if it is symmetric and its elements satisfy the following inequalities for all indices i, j, k, l, $i < j < k < l$:

$$a_{ij} + a_{kl} \leq a_{ik} + a_{jl}, \quad a_{il} + a_{jk} \leq a_{ik} + a_{jl} .$$

For more information on Monge, anti-Monge and Kalmanson matrices, and their properties, see [32]. The Koopmans–Beckmann QAP with one coefficient matrix being a Kalmanson matrix and the other one a Toeplitz matrix, has been investigated in [44]. The computational complexity of this problem is an open question, but analogously as in the case of the anti-Monge–Toeplitz QAP, polynomially solvable versions of the problem are obtained by imposing additional constraints to the Toeplitz matrix.

Further polynomially solvable cases arise as QAP formulations of other problems, like the linear arrangement problem, minimum feedback arc set problem, packing problems in graphs and subgraph isomorphism, see [23,35]. Polynomially solvable versions of these problems lead to polynomially solvable cases of the QAP. The coefficient matrices of these QAPs are the (weighted) adjacency matrices of the underlying graphs, and the special structure of these matrices is imposed by properties of these graphs. The methods used to solve these QAPs range from graph theoretical algorithms (in the case of the linear arrangement problem and the feedback arc set problem), to dynamic programming (in the case of subgraph isomorphism).

Lower Bounds

Lower bounding techniques are primarily used with implicit enumeration algorithms, such as branch and bound, to perform a limited search of the feasible region of a minimization problem, until an optimal solution is found. A more limited use of lower bounding techniques, is also for evaluating the performance of heuristic algorithms, by providing a relative measure of proximity of the suboptimal solution to the optimum. In comparing lower bounding techniques, the following criteria should be taken into consideration:

- complexity of computing the lower bound;
- tightness of the lower bound (i. e. closest to the optimum solution);
- efficiency in computing lower bounds for subsets of the primal feasible set.

Since there is no clear ranking of the performance of the lower bounds that will be discussed below, all of the above criteria should be kept in mind while reading the following paragraphs. Considering the asymptotic behavior of the QAP, it should be fair to assume that the tightness of the lower bound probably dominates all of the above criteria. That is, if there is a large number of feasible solutions close to the optimum, then a lower bound which is not tight enough, will fail to eliminate a large number of subproblems in the branching process.

Gilmore–Lawler Type Lower Bounds

Based on the formulation of the general QAP as a LAP of dimension n^2 stated in formulation (9), Lawler [88] derived lower bounds for the QAP, by constructing a solution matrix Y in the process of solving a series of LAPs. If the resulting matrix Y is a permutation matrix, then the objective function value is optimal, otherwise it is bounded from below by $\langle C, Y\rangle$. Specifically, consider an instance of QAP(C), where the matrix C is partitioned into n^2 minors, $C^{(i,j)} = (c_{ijkl})_{n\times n}$ for $i, j = 1, \ldots, n$. Each minor $C^{(i,j)}$ essentially contains the costs associated with the assignment $x_{ij} = 1$. Partition the solution matrix Y also into n^2 minors, $Y^{(i,j)} = (y_{ijkl})_{n\times n}$ for $i, j = 1, \ldots, n$, whose actual values are to be determined in the process. Solve the n^2 LAPs associated with each minor $C^{(i,j)}$,

$$\begin{cases} l_{ij} = \min \sum_{k=1}^{n}\sum_{l=1}^{n} c_{ijkl}\, y_{ijkl} \\ \text{s.t.} \quad \sum_{k=1}^{n} y_{ijkl} = 1, \quad l = 1, \ldots, n, \\ \qquad \sum_{l=1}^{n} y_{ijkl} = 1, \quad k = 1, \ldots, n, \\ \qquad y_{ijij} = 1, \\ \qquad y_{ijkl} \in \{0, 1\}, \quad i, j = 1, \ldots, n. \end{cases} \tag{12}$$

Observe that the last constraint essentially reduces the problem into an LAP of dimension $(n-1)$, obtained by deleting the ith row and jth column of the matrix $C^{(i,j)}$. Denote the solution matrix for each of the above LAPs by $Y^{(i,j)}$. Using the values l_{ij} from above, solve the LAP to obtain the *Gilmore–Lawler lower bound* for general QAPs:

$$\begin{cases} \text{GLB}(C) = \min & \sum_{i=1}^{n}\sum_{j=1}^{n} l_{ij}x_{ij} \\ \text{s.t.} & (x_{ij}) \in \mathbf{X}_n, \end{cases} \tag{13}$$

and denote its solution matrix by $X^* = (x^*_{ij})$. If

$$\frac{1}{n}\sum_{ij} x^*_{ij} Y^{(ij)} \in \mathbf{X}_n \,,$$

then $Y^* = (x^*_{ij}\, Y^{(ij)})_{n^2\times n^2}$ is a Kronecker product of two permutation matrices of dimension n, and then it is also an optimal solution. Otherwise the optimal solution to the QAP is bounded below by $\text{GLB}(C) = \langle C, Y^*\rangle$. Considering that each LAP can be solved in $O(n^3)$ time, the above lower bound for the general QAP of dimension n can be computed in $O(n^5)$ time.

For the more special Koopmans–Beckmann formulation of the QAP (cf. formulation (3)) where the quadratic costs are derived by the pairwise product of two matrices F and D, the structure of the problem can be used to reduce the computational effort. Before we proceed, let us make some definitions. For vectors $a, b \in \mathbf{R}^n$, define the following extremal variations of the usual inner product between vectors, by imposing an ordering in the elements of the vectors:

$$\langle a, b\rangle^- := \sum_{i=1}^{n} a_i b_i \,, \tag{14}$$

where $a_i \geq a_{i+1}$, $b_i \leq b_{i+1}$, $\forall i$, and

$$\langle a, b\rangle^+ := \sum_{i=1}^{n} a_i b_i \,, \tag{15}$$

where $a_i \geq a_{i+1}$, $b_i \geq b_{i+1}$, $\forall i$. The following is a well known result:

Proposition 4 [69] *For $a, b \in \mathbf{R}^n$ the following inequalities hold for any $\phi \in \mathcal{S}_n$:*

$$\langle a, b\rangle^- \leq \sum_{i=1}^{n} a_i b_{\phi(i)} \leq \langle a, b\rangle^+ .$$

Consider an instance QAP(F, D, B), and recall that this can be transformed into an instance of QAP(C) by assigning the values

$$c_{ijkl} = \begin{cases} f_{ik}d_{jl}, & \text{for } i \neq k,\ j \neq l, \\ f_{ii}d_{jj} + b_{ij}, & \text{for } i = k,\ j = l. \end{cases}$$

Each minor $C^{(i,j)}$ in the partitioned matrix C, is now $C^{(i,j)} = f_{(i\cdot)}\, d_{(j\cdot)}^{\top}$, where $f_{(i\cdot)}$ and $d_{(j\cdot)}$ is the ith and jth row of matrix F and D respectively. Therefore, using the result of Proposition 4, instead of solving n^2 LAPs we can easily compute the values l_{ij} as

$$l_{ij} = f_{ii}d_{jj} + b_{ij} + \left\langle \widehat{f}_{(i\cdot)}, \widehat{d}_{(j\cdot)} \right\rangle^{-} , \tag{16}$$

where the vectors $\widehat{f}_{(i\cdot)}, \widehat{d}_{(j\cdot)} \in \mathbf{R}^{n-1}$ are obtained by removing the ith and jth element of the vectors $f_{(i\cdot)}$ and $d_{(j\cdot)}$ respectively. Finally by solving the resulting LAP as in (12), we obtain the Gilmore–Lawler lower bound for the Koopmans–Beckmann QAPs, denoted by GLB(F, D), in $O(n^3)$ time. Its name is due to the fact that Lawler [88] and P.C. Gilmore [60] independently derived this lower bound, while the first author considered the case for general QAPs also. The simplicity of the Gilmore–Lawler lower bound makes it one of the most efficient to compute, although it deteriorates fast as n increases. The quality of the lower bound can be improved if the contribution of the quadratic term in the objective function is made to be smaller than that of the linear term. Consider the formulation of the general QAP where the linear and the quadratic terms are separated for clarity. By the above discussion the lower bound will be the solution to the LAP

$$\begin{cases} \min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n}(l_{ij} + c_{ijij})x_{ij} \\ \text{s.t.} & (x_{ij}) \in \mathbf{X}_n . \end{cases}$$

We want to decompose the cost coefficients in the quadratic term of (4) and transfer some of their value into the linear term such that $c_{ijij} \gg l_{ij}$, which will result in a tighter lower bound since the LAP can be solved exactly. This procedure known as *reduction* was introduced in [41], and it has been investigated by many researchers (see [18,47,56,128]). The general idea is to decompose each quadratic cost coefficient into several terms, which in turn will end up being linear cost coefficients and will be moved in the linear term of the objective function. Consider the following general decomposition for each quadratic cost coefficient in the objective function in (4):

D1) $c_{ijkl} = \overline{c}_{ijkl} + e_{ijk} + g_{ijl} + h_{ikl} + t_{jkl}, i \neq k, j \neq l.$

Here $e, g, h, t \in \mathbf{R}^{n^3}$. Substituting the above and separating terms, the objective function in (4) becomes

$$\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{\substack{j=1\\ j\neq l}}^{n}\sum_{k,l=1}^{n}\overline{c}_{ijkl}x_{ij}x_{kl} + \sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{\substack{j=1\\ j\neq l}}^{n}\sum_{k,l=1}^{n}(e_{ijk} + g_{ijl} + h_{ikl} + t_{jkl})x_{ij}x_{kl} + \sum_{i,j=1}^{n} c_{ijij}x_{ij} .$$

Consider now the term associated with the e_{ijk}:

$$\sum_{i,j=1}^{n}\sum_{\substack{k=1\\ k\neq i}}^{n}\sum_{\substack{l=1\\ l\neq j}}^{n} e_{ijk}x_{ij}x_{kl} = \sum_{i,j=1}^{n} x_{ij}\left[\sum_{\substack{k=1\\ k\neq i}}^{n} e_{ijk}\left(\sum_{\substack{l=1\\ l\neq j}}^{n} x_{kl}\right)\right] .$$

We can add the term

$$\sum_{\substack{k=1\\ k\neq i}}^{n} e_{ijk}$$

to the (i, j)th element of the LAP that composes the linear term of the objective function, since $x_{ij} = 1 \Rightarrow x_{kj} = 0 \Rightarrow \sum_{l\neq j} x_{kl} = 1, \forall k$. Using similar arguments for the vectors g, h and t, their costs become linear and the objective function with decomposed quadratic costs become

$$\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{\substack{j=1\\ j\neq l}}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\overline{c}_{ijkl}x_{ij}x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n}\widehat{c}_{ij}x_{ij} ,$$

where

$$\widehat{c}_{ij} = c_{ijij} + \sum_{\substack{k=1\\ k\neq i}}^{n} e_{ijk} + \sum_{\substack{l=1\\ l\neq j}}^{n} g_{ijl} + \sum_{\substack{k=1\\ k\neq i}}^{n} h_{kij} + \sum_{\substack{l=1\\ l\neq j}}^{n} t_{lij} .$$

Therefore we can apply the Gilmore–Lawler bound in the quadratic term of the decomposed objective function, whereas we can get an exact solution to the LAP that composes the linear term, and the sum of these two values will constitute a lower bound for the QAP. In the case of the Koopmans–Beckmann formulation of the QAP were we have two matrices F and D, the general decomposition scheme is:

D2)

$$\begin{cases} f_{ij} = \overline{f}_{ij} + \lambda_i + \mu_j, & i \neq j, \\ d_{kl} = \overline{d}_{kl} + \nu_k + p_l, & k \neq l. \end{cases}$$

Here $\lambda, \mu, \nu, p \in \mathbf{R}^n$. Substituting to the product $f_{ij}d_{kl}$ it is easily seen that D2) reduces to the general decomposition D1) with vectors $e, g, h, t \in \mathbf{R}^{n^3}$. Frieze and Yadegar [56] showed that the inclusion of the vectors h and t in D1) does not affect the value of the lower bound, and therefore are redundant (similarly the vectors μ and p for the Koopmans–Beckmann QAP are redundant in D2)). The same authors in [56] derived lower bounds for the QAP based on a Lagrangian relaxation (cf. also ► Integer programming: Lagrangian relaxation). Specifically, consider the Lagrangian relaxation of the 0–1 linear programming formulation of the QAP (see (10)), where the second and third constraints are included in the objective function, using as Lagrangian multipliers the elements of the vectors e and g. The Lagrangian function is thus defined as

$$\mathcal{L}(e,g) := \begin{cases} \min & \sum_{ijkl} c_{ijkl} y_{ijkl} \\ & \quad + \sum_{jkl} e_{jkl} \left(x_{kl} - \sum_i y_{ijkl} \right) \\ & \quad + \sum_{ikl} g_{ikl} \left(x_{kl} - \sum_j y_{ijkl} \right) \\ & \quad = \sum_{ijkl} (c_{ijkl} - e_{jkl} - g_{ikl}) y_{ijkl} \\ & \quad + \sum_{ij} \left(\sum_k e_{kij} + \sum_l g_{lij} \right) x_{ij} \\ \text{s.t.} & \text{first constraint in (10),} \\ & \text{fourth constraint in (10)} \\ & \ldots \\ & \text{last constraint in (10).} \end{cases}$$

The authors prove in [56] that for any choice of e and g, the solution to the above minimization problem will equal the value of the Gilmore–Lawler lower bound as applied to the QAP, with decomposed quadratic cost coefficients, as dictated by using the vectors e and g only in D1). Therefore, $\max_{e,g} \mathcal{L}(e,g)$ constitutes an upper bound on the lower bounds for the QAP, obtained by using the Gilmore–Lawler bound with decomposed quadratic cost coefficients. Using subgradient algorithms (cf. also ► Nondifferentiable optimization: Subgradient optimization methods) the authors derive near optimal solutions for $\max_{e,g} \mathcal{L}(e,g)$ resulting in two lower bounds, denoted by $FY1$ and $FY2$, corresponding to the two different solution approaches proposed. As suggested by the experimental results in [56], these bounds seem to be sharper than previously reported Gilmore–Lawler based lower bounds using reduction techniques. Almost all of the other approaches for obtaining lower bounds for the QAP with reduction techniques, are special cases of the general decomposition scheme D2) (see [18,47,128]).

Variance Reduction Lower Bounds

The variance reduction lower bounds were introduced in [93]. Consider an instance of the Koopmans–Beckmann formulation of the QAP, with input matrices $F = (f_{ij}), D = (d_{ij}) \in \mathbf{R}^{n \times n}$. Now partition both matrices into a sum of two matrices, $F = F_1 + F_2$ and $D = D_1 + D_2$, were $F_1 = (f_{ij}^{(1)})$, $F_2 = (f_{ij}^{(2)})$ and $D_1 = (d_{ij}^{(1)})$, $D_2 = (d_{ij}^{(2)})$. Construct an $n \times n$ matrix $L = (l_{ij})$, by solving the following n^2 LAPs:

$$l_{ij} = \min_{\substack{\phi \in S_n \\ \phi(i) = j}} \sum_{k=1}^{n} f_{ik}^{(1)} d_{j\phi(k)}^{(1)} + f_{ki}^{(1)} d_{\phi(k)j} + f_{ki} d_{\phi(k)j}^{(2)} - f_{ki}^{(2)} d_{\phi(k)j}^{(2)} \quad (17)$$

It is proved in [93] that the solution of the LAP with cost matrix L as constructed above, constitutes a lower bound for QAP(F, D). The problem of concern now, is to find a way to partition the matrices F and D such that the resulting lower bound is maximized. Observe that when $F_1 = F$ and $D_1 = D$ (i. e. no partitioning), we essentially derive the GLB(F, D).

Let $M \in \mathbf{R}^{m \times n}$, and denote its rows and columns respectively as $m_{(i\cdot)}$, and $m_{(\cdot,j)}$, $i, j = 1, \ldots, n$. Think of M

as a data set of mn elements m_{ij}, and define an average $\gamma(M)$ and a variance $V(M)$ as,

$$\gamma(M) := \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} m_{ij}\,,$$

$$V(M) := \sum_{i=1}^{m}\sum_{j=1}^{n}(\gamma(M) - m_{ij})^2\,.$$

Also define the total variance,

$$T(M,\lambda) := \lambda\sum_{i=1}^{m} V(m_{(i\cdot)}) + (1-\lambda)V(M)\,,$$

where $\lambda \in [0,1]$. The term $V(m_{(i\cdot)})$ stands for the variance of $m_{(i\cdot)}$, treated as a $1 \times n$ matrix. The authors observed that, as the variances of the matrices F and D decrease, the GLB(F, D) increases, while it is optimal if the variances of the rows of the matrices are zero. The partition scheme considered is of the form, $F_1 = F + \Delta_F$, $F_2 = -\Delta_F$, and $D_1 = D + \Delta_D$, $D_2 = -\Delta_D$. Considering only the matrix F, the problem is to find a matrix Δ_F, such that the variances of F_1 and F_2 and the sum of the variances of the rows for each F_1 and F_2 are minimized. We will only describe how Δ_F is obtained since Δ_D is obtained in the same way. The problem of minimizing the variances can be stated mathematically as

$$\min\, \theta T(F+\Delta_F,\lambda) + (1-\theta)T(-\Delta_F^{\top},\lambda)\,, \qquad (18)$$

where $\Delta_F \in \mathbf{R}^{n\times n}$ and $\theta \in [0,1]$ is a parameter. Two approximate solutions were proposed in [93], corresponding to the two reduction schemes

R1) $\delta_{ij} = \theta(f_{nn} - f_{ij}) + \delta_{nn}$,

R2) $\delta_{ij} = \theta(\gamma(f_{(\cdot,n)}) - \gamma(f_{(\cdot,j)})$.

Here $i, j = 1, \ldots, n$, $\Delta_F = (\delta_{ij})$, and with δ_{nn} being free to take any value (it was given a value of zero in the experiments conducted in [93]). In obtaining R2), the problem of minimizing the variances such that the matrix Δ_F is constrained to have constant columns, is considered. The matrix Δ_D is constructed in the same way. Based on the two reductions schemes above, the resulting lower bounds from the solution of (17) are denoted by $LB1(\theta)$, and $LB2(\theta)$. The above procedure for computing Δ_F, Δ_D has $O(n^2)$ computational complexity.

After the partitioning of the matrices F and D, the solution to the LAP with cost matrix $L = (l_{ij})$, were l_{ij} are defined in (17), will yield $LB1(\theta)$ or $LB2(\theta)$ according to what reduction scheme used. If $LB2(\theta)$ is used, the fact that the matrices F_2 and D_2 have constant columns can be exploited to compute the l_{ij}, $i, j = 1, \ldots, n$, efficiently as

$$l_{ij} = \left\langle \widehat{f}^{(1)}_{(i\cdot)}, \widehat{d}^{(1)}_{(j\cdot)} \right\rangle^{-} + f^{(2)}_{1i} \sum_{\substack{k=1\\k\neq j}}^{n} d_{kj} + d^{(2)}_{1j} \sum_{\substack{k=1\\k\neq i}}^{n} f_{kj} - (n-1) f^{(2)}_{1i} d^{(2)}_{1j} + f_{ii} d_{jj}\,,$$

where $\widehat{f}^{(1)}_{(i\cdot)}, \widehat{d}^{(1)}_{(j\cdot)} \in \mathbf{R}^{n-1}$ are the ith and jth row of F_1 and D_1 respectively, with the ith and jth elements removed from each, and $\langle\cdot,\cdot\rangle^{-}$ is defined in (14). In the case that $LB1(\theta)$ is used, the direct approach would be to solve the n^2 LAPs defined in (17), which will require $O(n^5)$ computational effort. A different approach is to calculate lower bounds for the values l_{ij}, $i, j = 1, \ldots, n$, as follows:

$$l_{ij} = \left\langle \widehat{f}^{(1)}_{(i\cdot)}, \widehat{d}^{(1)}_{(j\cdot)} \right\rangle^{-} + \left\langle \widehat{f}^{(2)}_{(\cdot,i)}, \widehat{d}_{(\cdot,j)} \right\rangle^{-} + \left\langle \widehat{f}_{(\cdot,i)}, \widehat{d}^{(2)}_{(\cdot,j)} \right\rangle^{-} + \left\langle \widehat{f}^{(2)}_{(\cdot,i)}, \widehat{d}^{(2)}_{(\cdot,j)} \right\rangle^{+}\,,$$

where each vector in the above extremal inner products, is of dimension n − 1, and corresponds to the ith row or column of the indicated matrix, upon removal of the ith element. Similarly as before the extremal inner products $\langle\cdot,\cdot\rangle^{-}$ and $\langle\cdot,\cdot\rangle^{+}$ are defined in (14) and (15). Using the above approach would require $O(n^3)$ time to compute lower bounds for the l_{ij}, $i, j = 1, \ldots, n$, thus the total computational complexity of the variance reduction lower bounds is $O(n^3)$. It is worth noting that there is also a closed form solution to problem (18) given in [71] which is

$$\delta_{ij} = \theta\lambda\frac{1-\theta}{1-\theta\lambda}\gamma(f_{(i\cdot)}) + \frac{\theta(1-\lambda)+\theta\lambda^2(1-\theta)-\theta^2\lambda^2(1-\theta)}{(1-\theta\lambda)(1-\lambda+\theta\lambda)}\gamma(F) - \frac{\theta\lambda(1-\theta)}{1-\lambda+\theta\lambda}\gamma(f_{(\cdot j)}) - \theta f_{ij}\,,$$

for $i, j = 1, \ldots, n$. However it was reported in [93] that using the above closed form in the computation of the lower bounds, poses implementation obstacles.

The experimental results conducted in [93], also suggest that the settings of $\theta = 0.5$ for $LB1(\theta)$, and $\theta = 1$ for $LB2(\theta)$ as best choices. Finally, these lower bounds perform well on QAPs with input matrices that have high variances, but their performance reduces to that of the Gilmore–Lawler bounds when the variance of the matrices is small.

Eigenvalue Based Lower Bounds

These bounds were introduced in [50,51], and are applied to the Koopmans–Beckmann formulation of the QAP. This approach utilizes known results on permutation matrices and eigenvalues, and exploits the special structure of QAP(F, D). Upon the introduction of the method in [50,51], many improvements and generalizations have appeared (see for example [65,66,67,68,123,124]). There is a resemblance with the Gilmore–Lawler based lower bounds, in the sense that, based upon a general lower bound, reduction techniques are applied to the quadratic terms of the objective function in order to improve its quality. The reduction techniques that applied to eigenvalue based lower bounds however, yield a significant improvement, which is not really the case with the Gilmore–Lawler bounds under certain reductions.

Considering the trace formulation of the QAP in (7), with F and D being real symmetric matrices, therefore with all their eigenvalues being real, the following result can be stated for the quadratic term [51]:

Theorem 5 [51] *Let $F, D \in \boldsymbol{R}^{n\times n}$ be symmetric matrices, and denote by $\lambda = (\lambda_1, \ldots, \lambda_n)^\intercal$ and $x_1, \ldots, x_n$ the eigenvalues and eigenvectors of F, and by $\mu = (\mu_1, \ldots, \mu_n)^\intercal$ and $y_1, \ldots, y_n$ the the eigenvalues and eigenvectors of D. Then the following two relations are true for all $X \in \boldsymbol{X}_n$,*

i) $tr\, FXDX^\intercal = \sum_{i=1}^n \sum_{j=1}^n \lambda_i \mu_j \langle x_i, Xy_j\rangle^2 = \lambda^\intercal S(X)\mu$. Here $S(X) = (\langle x_i, Xy_j\rangle^2)$ is a doubly stochastic matrix,

ii) $\langle \lambda, \mu\rangle^- \le tr\, FXDX^\intercal \le \langle \lambda, \mu\rangle^+$.

Using Theorem 5ii) a lower bound for QAP(F, D) based on the eigenvalues of F and D is then

$$EVB = \langle \lambda, \mu\rangle^- + \min_{X\in\mathbf{X}_n} \operatorname{tr} BX^\top ,$$

where the second term is an ordinary LAP that can be solved exactly. Observe that in Theorem 5, the smaller the interval $[\langle \lambda, \mu\rangle^-, \langle \lambda, \mu\rangle^+]$ is, the closest $\langle \lambda, \mu\rangle^-$ is to tr $FXDX^\intercal$. A possible way of making the interval smaller, is to decompose the matrices F and D such that some of their value will be transfered in the linear term, and the eigenvalues of the resulting matrices that compose the quadratic term, are as uniform in value as possible. Define the *spread* of the matrix F as

$$\text{spread}(F) := \max\left\{\left|\lambda_i - \lambda_j\right| :\ i, j = 1, \ldots, n\right\} .$$

Based on the above discussion, we want to minimize the spreads of the matrices that compose the quadratic term. There is no simple closed form for expressing spread(F) in terms of f_{ij}, however we can minimize instead a formula for the upper bound given in [98]:

$$\text{spread}(F) \le m(F) = \left[2\sum_{i=1}^n \sum_{j=1}^n f_{ij}^2 - \frac{2}{n}(\operatorname{tr} F)^2\right]^{\frac{1}{2}} . \tag{19}$$

The decomposition scheme that the authors use in [51], is the following:

$$f_{ij} = \overline{f}_{ij} + e_i + e_j + r_{ij} , \tag{20}$$

$$d_{kl} = \overline{d}_{kl} + g_k + g_l + s_{kl} , \tag{21}$$

where $r_{ij} = s_{ij} = 0$, for $i \ne j$.

Consider the decomposition for matrix F and let $\bar{F} = (\overline{f}_{ij})$. Minimizing the function $f(e, r) = m(\overline{F})$ obtained by substituting the values of $\overline{f}_{ij}$ in (19), the following values are obtained [51]:

$$z = \frac{1}{2(n-1)}\left(\sum_{i=1}^n \sum_{j=1}^n f_{ij} - \operatorname{tr} F\right) , \tag{22}$$

$$e_i = \frac{1}{n-2}\left(\sum_{j=1}^n f_{ij} - f_{ii} - z\right) , \tag{23}$$

$$r_{ii} = f_{ii} - 2e_i , \tag{24}$$

for $i = 1, \ldots, n$. Analogously we obtain the values for g and s for the decomposition of D. Replacing F and D in (7) we obtain

$$\operatorname{tr}(FXD + B)X^\top = \operatorname{tr}(\overline{F}X\overline{D} + \overline{B})X^\top ,$$

where $\overline{b}_{ij} = b_{ij} + f_{ii}d_{jj} + 2e_i \sum_{\substack{k=1 \\ k\neq j}}^{n} d_{jk}$, and matrices $\overline{F}$ and $\overline{D}$ have respective eigenvalues $\overline{\lambda} = (\overline{\lambda}_1, \ldots, \overline{\lambda}_n)$ and $\overline{\mu} = (\overline{\mu}_1, \ldots, \overline{\mu}_n)$. The corresponding eigenvalue lower bound is then

$$EVB1 = \left\langle \overline{\lambda}, \overline{\mu} \right\rangle^{-} + \min_{X \in \mathbf{X}_n} \operatorname{tr} \overline{B} X^{\top} .$$

If we restrict ourselves only to pure quadratic ($f_{ii} = d_{ii} = 0, \forall i, B = 0$) symmetric QAPs, the matrix $\overline{B}$ in the above decomposition becomes $\overline{B} = cw^{\top}$, where $c = 2(e_1, \ldots, e_n)^{\top}$ and $w = (\sum_j d_{1j}, \ldots, \sum_j d_{nj})^{\intercal}$. Therefore $\min_{X \in \mathbf{X}_n} \operatorname{tr} \overline{B} X^{\top} = \langle c, w \rangle^{-}$, and

$$EVB1 = \left\langle \overline{\lambda}, \overline{\mu} \right\rangle^{-} + \langle c, w \rangle^{-} \leq \min_{X \in \mathbf{X}_n} \operatorname{tr}(\overline{F} X \overline{D} + \overline{B}) X^{\top} .$$

We can however obtain further improvement as suggested by F. Rendl [123], who examined the linear term $\langle c, w \rangle^{-}$, and proposed a method where $EVB1$ is iteratively improved, until some specified number of iterations is reached, or we have satisfied an optimality condition. More specifically, let $S_k := \{X_1, \ldots, X_k\} \subseteq \mathbf{X}_n$, and

$$L(X_i) := \min \{ \langle c, X_i w \rangle : \ X_i \in \mathbf{X}_n \setminus S_{i-1} \} ,$$

so for any integer $k \geq 1$, $L(X_1) \leq L(X_2) \leq \cdots \leq L(X_k)$. In other words the set S_k contains the k first solutions (permutation matrices) of the problem $\min_{X \in X_n} \langle c, X_i w \rangle$, where the first solution $X_1 \ni L(X_1) = \langle c, w \rangle^{-}$. Let

$$\mathrm{QAP}(\overline{F}, \overline{D}, X_i) = \operatorname{tr}(\overline{F} X_i \overline{D} + \overline{B}) X_i^{\top} ,$$

and also define the following

$$Z(k) := \min \left\{ \mathrm{QAP}(\overline{F}, \overline{D}, X_i) : \ i = 1, \ldots, k \right\} .$$

The following inequalities [123] result:

$$Z(1) \geq \cdots \geq Z(k) \geq \left\langle \overline{\lambda}, \overline{\mu} \right\rangle^{-} + L(X_k) \geq \cdots \geq \left\langle \overline{\lambda}, \overline{\mu} \right\rangle^{-} + L(X_1) ,$$

where if $Z(i) = \left\langle \overline{\lambda}, \overline{\mu} \right\rangle^{-} + L(X_i)$ for some i, then X_i is the optimal solution to the problem. So essentially, we try to close or reduce the gap between the optimal solution of the QAP and the lower bound $EVB1$, by increasing the value of the linear term $\langle c, w \rangle^{-}$ in the bound in k steps, where k is specified as a parameter. The generation of the set S_k or ranking as it is called, is a special case of the problem of ranking the k first solutions of an assignment problem with cost matrix $(c_i w_j)$ where, as shown in [104], has time complexity $O(kn^3)$. Rendl [123] presents an $O(n \log n + (n + \log k)k)$ for this special case. There are two issues regarding the effectiveness of the above ranking procedure, in improving the lower bound, addressed in [123]. First, observe that if the vectors c and w have $m \leq n$ equal elements, then there are at least $m!$ permutation matrices $\{X_i\}$ such that the values $\langle c, X_i w \rangle$ are equal. This in turn, implies that there will be none or small improvement in the lower bound while generating S_k for quite some number of iterations. As dictated by the decomposition in (22), (23), c and w will have equal elements if the row sums of F and D are equal. One condition then for applying the ranking procedure, is that most of the row sums of F and D are not equal. Secondly, Rendl [123] also defines a ratio called the *degree of linearity* based on the ranges of the quadratic and linear terms that compose the lower bound

$$L := \frac{\left\langle \overline{\lambda}, \overline{\mu} \right\rangle^{+} - \left\langle \overline{\lambda}, \overline{\mu} \right\rangle^{-}}{\langle c, w \rangle^{+} - \langle c, w \rangle^{-}} .$$

The influence of the linear term on the lower bound then is inversely proportional to the value of L. A small value of L suggests that the ranking procedure would be beneficial for the improvement of $EVB1$ for symmetric, pure quadratic QAPs. For large values of L, we can expect that the quadratic term dominates the linear term in the objective function, and [51] suggest the following improvement on $EVB1$. Considering Theorem 5i) as applied to the reduced matrices $\overline{F}$ and $\overline{D}$, denote the elements of the matrix $S(X)$ by $s_{ij} = \langle x_i, X y_j \rangle^2$. We can apply the bounds $l_{ij} \leq s_{ij} \leq u_{ij}$ where

$$u_{ij} = \max\{(\langle x_i, y_j \rangle^{-})^2, (\langle x_i, y_j \rangle^{+})^2\},$$

$$l_{ij} = \begin{cases} 0 & \text{if } \langle x_i, y_j \rangle^{-}, \langle x_i, y_j \rangle^{+} \text{ differ in sign,} \\ \min\{(\langle x_i, y_j \rangle^{-})^2, (\langle x_i, y_j \rangle^{+})^2\} & \text{otherwise.} \end{cases}$$

Recalling the fact that the s_{ij} are the elements of a doubly stochastic matrix, we can then form the *capacitated*

transportation problem

$$\text{CTP}^* \begin{cases} \min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} \overline{\lambda}_i \overline{\mu}_j s_{ij} \\ \text{s.t.} & \displaystyle\sum_{i=1}^{n} s_{ij} = 1, \quad j = 1, \ldots, n, \\ & \displaystyle\sum_{j=1}^{n} s_{ij} = 1, \quad i = 1, \ldots, n, \\ & l_{ij} \le s_{ij} \le u_{ij}. \end{cases}$$

The new lower bound then would be

$$EVB2 = \text{CTP}^* + \langle c, w \rangle^- \ .$$

A more generalized approach to eigenvalue based lower bounding techniques, was employed in [66], that led to new lower bounds. Consider the following sets of $n \times n$ matrices, where $I \in \mathbf{R}^{n\times n}$ is the identity matrix, and $u := (1, \ldots, 1)^\mathsf{T} \in \mathbf{R}^n$ is the vector of ones,

$$\begin{aligned} \mathcal{O} &:= \left\{X\colon\ X^\top X = I\right\} \ , \\ \mathcal{E} &:= \left\{X\colon\ Xu = X^\top u = u\right\} \ , \\ \mathcal{N} &:= \{X\colon\ X \ge 0\} \ . \end{aligned}$$

It is a well known result that $\mathbf{X}_n = \mathcal{O} \cap \mathcal{E} \cap \mathcal{N}$, while the set of doubly stochastic matrices $\Omega = \mathcal{E} \cap \mathcal{N}$. Moreover, by Birkhoff's theorem [15] we know that Ω is a convex polyhedron with a vertex set $\mathbf{X}_n$, that is, $\Omega = \text{conv}\{X : X \in \mathbf{X}_n\}$. Considering the above characterization of $\mathbf{X}_n$, we can see that any solution to a relaxation of the QAP obtained from excluding one or two of the matrix sets $\mathcal{O}$, $\mathcal{E}$ and $\mathcal{N}$, will yield a lower bound. Naturally the relaxation, and therefore the lower bound, will be tighter if only one of the matrix sets is excluded. In relation to Theorem 5, Rendl and Wolkowicz [124] showed that

$$\begin{aligned} \min_{X\in\mathcal{O}} \operatorname{tr} FXDX^\top &= \operatorname{tr} F\Lambda_F\Lambda_D^\top D\Lambda_D\Lambda_F^\top = \langle \lambda, \mu \rangle^- \ , \\ \max_{X\in\mathcal{O}} \operatorname{tr} FXDX^\top &= \operatorname{tr} F\Lambda_F\Lambda_D^\top D\Lambda_D\Lambda_F^\top = \langle \lambda, \mu \rangle^+ \ , \end{aligned}$$

where Λ_F, Λ_D are the matrices with columns the eigenvectors of F and D respectively, in the order specified by the minimal (maximal) inner product of the eigenvalues. In other words, the lower bound on the quadratic part of the QAP as obtained from the *EVB*, is derived by relaxing the feasible set to that of orthogonal matrices. In [124] a new lower bound is derived, similar to *EVB*2 but using a different approach to decompose the matrices F and D. More specifically, denote the decomposition scheme in (20) and (21) by the vector $d := (e^\mathsf{T}, g^\mathsf{T}, r^\mathsf{T}, s^\mathsf{T}) \in \mathbf{R}^{4n}$, where $r = (r_{11}, \ldots, r_{nn})^\mathsf{T}$ and $s = (s_{11}, \ldots, s_{nn})^\mathsf{T}$, and consider *EVB*1 as a function of d. Maximizing this function with respect to d will result in a lower bound with the best possible decomposition that involves both the linear and quadratic terms. This leads to a nonlinear, nonsmooth, nonconcave function which is hard to solve, and a steepest ascent algorithm is proposed for maximizing it in [124]. The new bound, denoted *EVB*3, produces some of the best lower bounds for the QAP, with the expense however of high computational requirements.

All of the above discussed lower bounds, relax the set of permutation matrices to $\mathcal{O}$. A tighter relaxation was proposed in [67], where the set of permutation matrices was relaxed to $\mathcal{O} \cap \mathcal{E}$, by incorporating $\mathcal{E}$ in the objective function, by exploiting the fact that the vector of ones u is both a left and right eigenvector with eigenvalue 1, for any $X \in \mathbf{X}_n$. More specifically define

$$P := [u/\|u\| \vdots V] \ ,$$

where $V^\mathsf{T}\ u = 0$, $V^\mathsf{T}\ V = I_{n-1}$. therefore V is an orthonormal basis for $\{u\}^\perp$, while $Q := VV^\mathsf{T}$ is the orthogonal projection on $\{u\}^\perp$. The following characterization of the permutation matrices is given in [67]

Lemma 6 [67] *Let $X \in \boldsymbol{R}^{n\times n}$ and $Y \in \boldsymbol{R}^{n-1\times n-1}$. If*

$$X = P \begin{bmatrix} 1 & 0 \\ 0 & Y \end{bmatrix} P^\top \ , \tag{25}$$

then

$$\begin{aligned} & X \in \mathcal{E} \ , \\ & X \in \mathcal{N} \Leftrightarrow VYV^\top \ge -\frac{uu^\top}{\|u\|^2} \ , \\ & X \in \mathcal{O} \Leftrightarrow Y \in \mathcal{O}_{n-1} \ . \end{aligned}$$

Conversely if $X \in \mathcal{E}$ then there exists a Y such that (25) holds.

Note that the above characterization of the permutation matrices, preserves the orthogonality and the trace structure of the problem. Substituting $X = -uu^\mathsf{T}/\|u\|^2 + VYV^\mathsf{T}$ as suggested by (25), in the trace formulation of the QAP in (7), we have an equivalent projected problem (PQAP) of dimension $n-1$ on the variable

matrix Y. The new lower bound *IVB* is obtained by relaxing Y to O_{n-1}, therefore deriving a lower bound for the quadratic part of PQAP, while the linear part can be solved exactly as an LAP. Decompositions for improving the *IVB* are also considered in [67], where it is shown that the quadratic term in the projected problem is unaffected by e and g in the decomposition scheme in (20), (21). Obtaining a lower bound by considering both the quadratic and linear term is also considered in [78].

The symmetry assumption on the QAP is required by any of the eigenvalue based lower bounding techniques described above. Hadley, Rendl and Wolkowicz [68] show that any real QAP can be transformed into an equivalent QAP where the matrices F and D are Hermitian, which allows the application of eigenvalue based bounds.

Bounds Based on Semidefinite Relaxations

Recently (as of 1999), semidefinite programming (SDP) relaxations for the QAP were considered [76,77,137]. The SDP relaxations considered in these papers are solved by interior point methods or cutting plane methods (cf. also ► Linear programming: Interior point methods; ► Extended cutting plane algorithm), and the obtained solutions are valid lower bounds for the QAP. In terms of quality the bounds obtained in this way are competitive with the best existing lower bounds for the QAP. For many test instances from QAPLIB [31] such as some instances of Hadley [26], Roucairol [128], Nugent et al. [105], and Taillard [133], they are the best existing bounds. However, due to prohibitively high computation time requirements, the use of such approaches as basic bounding procedures within branch and bound algorithms is up to now not feasible. See [77,78] for a detailed description of SDP approaches to the QAP and illustrate the idea by describing just one semidefinite programming relaxation for the QAP.

The set of $n \times n$ permutation matrices $\mathbf{X}_n$ is the intersection of the set of $n \times n$ 0–1 matrices, denoted by $\mathcal{Z}_n$, and the set $\mathcal{E}_n$ of $n \times n$ matrices with row and column sums equal to 1. Moreover, $\mathbf{X}_n$ is also the intersection of $\mathcal{Z}_n$ with the set of $n \times n$ orthogonal matrices, denoted by $\mathcal{O}_n$. Hence

$$\mathbf{X}_n = \mathcal{Z}_n \cap \mathcal{E}_n = \mathcal{Z}_n \cap \mathcal{O}_n \ .$$

Recall that

$$\mathcal{O}_n = \left\{X \in \mathbf{R}^{n\times n}\colon\ XX^\top = X^\top X = I\right\}$$

and

$$\mathcal{E}_n = \left\{X \in \mathbf{R}^{n\times n}\colon\ Xu = X^\top u = u\right\}\ ,$$

where I is the $n \times n$ identity matrix and u is the n-dimensional vector of all ones. Then, the trace formulation of the QAP (7) with the additional linear term

$$-2\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}x_{ij},$$

can be represented equivalently as follows:

$$\mathrm{QAP}_E \begin{cases} \min & \mathrm{tr}(FXDX^\top - 2BX^\top) \\ \text{s.t.} & XX^\top = X^\top X = I, \\ & Xu = X^\top u = u, \\ & x_{ij}^2 - x_{ij} = 0. \end{cases}$$

In order to obtain a semidefinite relaxation for the QAP from the formulation $\mathrm{QAP}_{\mathcal{E}}$ above, we introduce first an n^2-dimensional vector $\mathrm{vec}(X)$. $\mathrm{vec}(X)$ is obtained as a columnwise ordering of the entries of matrix X. Then the vector $\mathrm{vec}(X)$ is lifted into the space of $(n^2+1)\times(n^2+1)$ matrices by introducing a matrix Y_X,

$$Y_X = \begin{pmatrix} x_0 & \mathrm{vec}(X)^\top \\ \mathrm{vec}(X) & \mathrm{vec}(X)\,\mathrm{vec}(X)^\top \end{pmatrix}.$$

Thus, Y_X has some entry x_0 in the left-upper corner followed by the vector $\mathrm{vec}(X)$ in its first row (column). The remaining terms are those of the matrix $\mathrm{vec}(X)\,\mathrm{vec}(X)^\intercal$ sitting on the right lower $n^2 \times n^2$ block of Y_X. Secondly, the coefficients of the problem are collected in an $(n^2+1)\times(n^2+1)$ matrix K given as

$$K = \begin{pmatrix} 0 & -\mathrm{vec}(B)^\top \\ \mathrm{vec}(B) & D \otimes F \end{pmatrix},$$

where the operator vec is defined as above and $D \otimes F$ is the Kronecker product of D and F.

It is easy to see that with these notations the objective function of $\mathrm{QAP}_{\mathcal{E}}$ equals $\mathrm{tr}(KY_X)$. By setting $y_{00} := x_0 = 1$ as done in [77], one obtains two additional

constraints to be fulfilled by the matrix Y_X: Y_X is positive semidefinite and matrix Y_X is a rank-one matrix. Whereas the semidefiniteness and the equality $y_{00} = 1$ can be immediately included in an SDP relaxation, the rank-one condition is hard to handle and is discarded in an SDP relaxation. In order to assure that the rank-one positive semidefinite matrix Y_X is obtained by an $n \times n$ permutation matrix as described above, other constraints should be imposed to Y_X. Such conditions can be formulated as valid constraints of an SDP formulation for the QAP by means of some new operators, acting on matrices or vectors as introduced below. Given a matrix $A \in \mathbf{R}^{n\times n}$, the operator $\mathrm{diag}(A) \in \mathbf{R}^n$ produces a vector containing the diagonal entries of matrix A in their natural order, that is, from top-left to bottom-right. The adjoint operator Diag acts on a vector $V \in \mathbf{R}^n$ and produces a matrix $\mathrm{Diag}(V) \in \mathbf{R}^{n\times n}$ with off-diagonal entries equal to 0 and the components of V on the main diagonal. For some matrix $Y \in \mathbf{R}^{(n^2+1)\times(n^2+1)}$, operator $\mathrm{arrow}(Y) \in \mathbf{R}^{(n^2+1)}$, is defined as $\mathrm{arrow}(Y) := \mathrm{diag}(Y) - (0, Y_{0,1:n^2})$, where $(0, Y_{(0,1:n^2)})$ is an $n^2 + 1$-dimensional vector with first entry equal to 0 and other entries coinciding with the entries of Y lying on the 0th row and in columns between 1 and n^2, in their natural order. The adjoint operator Arrow acts on an $n^2 + 1$-dimensional vector W and produces an $(n^2 + 1) \times (n^2 + 1)$ matrix $\mathrm{Arrow}(W)$

$$\mathrm{Arrow}(W) = \begin{pmatrix} w_0 & \frac{1}{2}W_{1:n^2}^\top \\ \frac{1}{2}W_{(1:n^2)} & \mathrm{Diag}(W_{1:n^2}) \end{pmatrix},$$

where $W_{(1:n^2)}$ is the n^2-dimensional vector obtained from W by removing its first entry w_0. Furthermore, we are going to consider an $(n^2 + 1) \times (n^2 + 1)$ matrix Y as composed of its first row $Y_{(0\cdot)}$, of its first column $Y_{(\cdot,0)}$, and of n^2 submatrices of size $n \times n$ each, which are arranged in an $n \times n$ array of $n \times n$ matrices and produce its remaining $n^2 \times n^2$ block (this is similar to the structure of a Kronecker product of two $n \times n$ matrices. The entry $y_{\alpha\beta}$, $1 \le \alpha, \beta \le n^2$, will be also denoted by $y_{(ij)(kl)}$, with $1 \le i, j, k, l \le n$, where $\alpha = (i-1)\,n + j$ and $\beta = (k-1)\,n + l$. Hence, $y_{(ij)(kl)}$ is the element with coordinates (j, l) within the $n \times n$ block with coordinates (i, k).

With these formal conventions let us define the so-called *block-0-diagonal* and *off-0-diagonal* operators, acting on an $(n^2 + 1) \times (n^2 + 1)$ matrix Y, and denoted by b^0 diag and o^0 diag, respectively. $b^0\,\mathrm{diag}(Y)$ and $o^0\,\mathrm{diag}(Y)$ are $n \times n$ matrices given as follows:

$$b^0\,\mathrm{diag}(Y) = \sum_{k=1}^{n} Y_{(k\cdot)(k\cdot)}\,,$$

$$o^0\,\mathrm{diag}(Y) = \sum_{k=1}^{n} Y_{(\cdot,k),(\cdot,k)}\,,$$

where, for $1 \le k \le n$, $Y_{(k\cdot)(k\cdot)}$ is the kth $n \times n$ matrix on the diagonal of the $n \times n$ array of matrices, defined as described above. Analogously, $Y_{(\cdot,k),(\cdot,k)}$ is an $n \times n$ matrix consisting of the diagonal elements sitting on the position (k, k) of the $n \times n$ matrices (n^2 matrices altogether) which form the $n^2 \times n^2$ lower right block of matrix Y. The corresponding adjoint operators B^0 Diag and O^0 Diag act on an $n \times n$ matrix S and produce $(n^2 + 1) \times (n^2 + 1)$ matrices as follows:

$$B^0\,\mathrm{Diag} = \begin{pmatrix} 0 & 0 \\ 0 & I \otimes S \end{pmatrix},$$

$$O^0\,\mathrm{Diag} = \begin{pmatrix} 0 & 0 \\ 0 & S \otimes I \end{pmatrix}.$$

Finally, let us denote by e_0 the $n^2 + 1$-dimensional unit vector with first component equal to 1 and all other components equal to 0, and let R be the $(n^2 + 1) \times (n^2 + 1)$ matrix given by

$$R = \begin{pmatrix} n & -u^\top \otimes u^\top \\ -u \otimes u & I \otimes E \end{pmatrix} + \begin{pmatrix} n & -u^\top \otimes u^\top \\ -u \otimes u & E \otimes I \end{pmatrix},$$

where E is the $n \times n$ matrix of all ones. With these notations, a semidefinite relaxation for $\mathrm{QAP}_{\mathcal{I}}$ is given as follows

$$\mathrm{QAP}_{\mathrm{R0}} \begin{cases} \min & \mathrm{tr}(KY) \\ \text{s.t.} & b^0\,\mathrm{diag}(Y) = I, \\ & o^0\,\mathrm{diag}(Y) = I, \\ & \mathrm{arrow}(Y) = e_0, \\ & \mathrm{tr}(RY) = 0, \\ & Y \succeq 0, \end{cases}$$

where $\preccurlyeq$ is the so-called *Löwner partial order*, that is, $A \preccurlyeq B$ if and only if $B - A \succcurlyeq 0$, that is $B - A$ is positive semidefinite. In [77] it was shown that an equivalent formulation for the considered QAP is obtained

from $\mathrm{QAP}_{\mathrm{R0}}$ by imposing one additional condition on the matrix Y, namely, the rank-one condition.

Exact Solution Methods

Several exact solution approaches for solving the QAP will be presented in this section. Specifically the exact algorithms that have been used for the QAP are dynamic programming, cutting plane algorithms, and branch and bound which appears to be the most successful one.

Branch and Bound

Branch and bound algorithms appear to be the most efficient exact algorithms for solving the QAP. For the QAP there are three types of branch and bound algorithms, namely:

- *Single assignment algorithms* ([60,88]).
- *Pair assignment algorithms* ([59,86,105]).
- *Relative positioning algorithm* ([97]).

All of the above algorithms work by iterative constructing an optimal permutation starting from an empty permutation. The single assignment algorithms seem to be the most efficient and the pair assignment algorithms do not have favorable computational results.

We will now describe a recent branch and bound algorithm for the QAP, that was proposed in [111]. In the description that follows we will consider the Koopmans–Beckmann formulation of the QAP. First let us define the necessary notation used in describing the branch and bound algorithm. A partial permutation for the set of integers $\mathcal{S}_n = \{1, \ldots, n\}$ is denoted by

$$\phi_k := \begin{pmatrix} 1 & 2 & \cdots & k \\ \phi_k(1) & \phi_k(2) & \cdots & \phi_k(k) \end{pmatrix}$$

where $k \leq n$. From now we will write $\phi_k = (\phi_k(1), \phi_k(2), \ldots, \phi_k(k))$ for short. An assignment of a facility i to a location j will be denoted by $i \to j$, while if i must never be assigned to j we will write $i \nrightarrow j$. Note that ϕ_k is essentially a partial assignment of facilities to locations. If we want to add an extra assignment to some ϕ_k, say $k+1 \to j$, we will write $\phi_{k+1} = \phi_k \cup k+1 \to j$, thereby $\phi_{k+1}(i) = \phi_k(i)$ for $i = 1, \ldots, k$, and $\phi_{k+1}(k+1) = j$. Given some ϕ_k let its range be $Q_k := \{\phi_k(i) : i = 1, \ldots, k\}$, and define the sets of nonpermissible assignments to be $E_{k+1} := \{j \in \mathcal{S}_n \setminus Q_k : k+1 \nrightarrow j\}$. Given an instance QAP$(F, D, B)$, a pair of ϕ_k and E_{k+1} completely defines a subproblem P_i as

$$\mathrm{P}_i \begin{cases} \min\limits_{\phi \in S_n} & \sum\limits_{i}^{n}\sum\limits_{j}^{n} f_{ij} d_{\phi(i)\phi(j)} + \sum\limits_{i}^{n} b_{i\phi(i)} \\ \text{s.t.} & \phi(i) = \phi_k(i), \quad i = 1, \ldots, k, \\ & \phi(k+1) \notin E_{k+1}. \end{cases}$$

The original problem P_0 is obtained for an empty partial permutation ϕ_0 and $E_1 = \emptyset$. For each P_i a lower bound $g(\mathrm{P}_i)$ can be computed, using any of the lower bounds described previously, and let the optimal solution to P_i be denoted by $f(\mathrm{P}_i)$. In the branch and bound algorithm, a forest of n binary trees is constructed, where each node of the tree corresponds to a partial subproblem P_i. The *branching* process is as follows. Given a node P_i (i. e. a subproblem) defined by some ϕ_k and E_{k+1}, two descendant nodes are created, the left child P_i^l and the right child P_i^r. For P_i^l we set $\phi_{k+1}^l = \phi_k \cup k+1 \to j$ for some $j \notin E_k$ and $E_{k+2}^l = \emptyset$, while for P_i^r we set $\phi_k^r = \phi_k$ and $E_{k+1}^r = E_{k+1} \cup j$. A node which has ϕ_k with $k = n-1$ cannot decomposed further, and it is called a terminal node. Immediately we can identify the following properties

- $g(\mathrm{P}_i) \leq f(\mathrm{P}_i)$ for any node P_i,
- $g(\mathrm{P}_i) = f(\mathrm{P}_i)$ if P_i is a terminal node,
- $g(\mathrm{P}_j) \geq g(\mathrm{P}_i)$ if P_j has descended from P_i.

A node defined by some ϕ_k and E_{k+1} will have two terminal nodes as children if $k = n-2$. Moreover, for any node $|E_{k+1}| + k \leq n-1$, while if equality holds then there is only one $j \notin E_{k+1}$ and only one left child is generated with $\phi_{k+1} = \phi_k \cup k+1 \to j$ and $E_{k+2} = \emptyset$.

The branch and bound algorithm in [111] starts by computing an upper bound solution to the original subproblem P_0 by means of a heuristic (cf. also ▶ Heuristic search). Let the corresponding permutation be $\phi = (\phi(1), \phi(2), \ldots, \phi(\mathrm{n}))$. Note that during the process of the algorithm the upper bound is continuously updated whenever a better feasible solution is found. Then n nodes are created, where for each P_i for $i = 1, \ldots, n$, we set $\phi_1 = \phi_0 \cup 1 \to \phi(i)$. $E_2 = \emptyset$, and $g(\mathrm{P}\}_i) = 0$. Then the following steps are performed at each iteration

1) Selection: Here we choose which node to examine next, and we choose the node with the maximum $g(\mathrm{P}_i)$.
2) Branching: Given the chosen node P_i from step 1, we create two new nodes P_i^l and P_i^r, based on

the branching scheme described previously. We set $g(P_i^r) = g(P_i)$ and we compute $g(P_i^l)$.

3) Elimination: If $g(P_i^l)$ is less than or equal to the current upper bound, then the node P_i^l is pruned, that is, marked not to be considered in step 1) in the future.
4) Termination: The algorithm stops if, and only if, there are no more nodes to be considered in step 1).

The authors in [111] applied the above described branch and bound algorithm for the QAP in conjunction with the variance reduction lower bounds described previously.

Traditional Cutting Plane Methods

Traditional cutting plane algorithms for the QAP have been developed by a different authors, [7,8,9,13,14], and [80]. These algorithms make use of mixed integer linear programming (MILP) formulations for the QAP which are suitable for Benders decomposition. In the vein of Benders, the MILP formulation is decomposed into a master problem and a subproblem, called also slave problem, where the master problem contains the original assignment variables and constraints. For a fixed assignment the slave problem is usually a linear program and hence, solvable in polynomial time. The master problem is a linear program formulated in terms of the original assignment variables and of the dual variables of the slave problem, and is solvable in polynomial time for fixed values of those dual variables. The algorithms work typically as follows. First, a heuristic is applied to generate a starting assignment. Then the slave problem is solved for fixed values of the assignment variables implied by that assignment, and optimal values of the primal and dual variables are computed. If the dual solution of the slave problem satisfies all constraints of the master problem, we have an optimal solution for the original MILP formulation of the QAP. Otherwise, at least one of the constraints of the master problem is violated. In this case, the master problem is solved with fixed values for the dual variables of the slave problem and the obtained solution is given as input to the slave problem. The procedure is then repeated until the solution of the slave problem fulfills all constraints of the master problem.

Clearly any solution of the master problem obtained by fixing the dual variables of the slave problem to some feasible values, is a lower bound for the considered QAP. On the other side, the objective function value of the QAP corresponding to any feasible setting of the assignment variables is an upper bound. The algorithm terminates when the lower and the upper bounds coincide. Generally, the time needed for the upper and the lower bounds to converge to a common value is too large, and hence these methods may solve to optimality only very small QAPs. However, heuristics derived from cutting plane approaches produce good suboptimal solutions in early stages of the search, see for example, [21] and [14].

Polyhedral Cutting Planes

Similarly to traditional cutting plane methods also polyhedral cutting planes or branch and cut algorithms (cf. also ► Integer programming: Branch and cut algorithms) make use of an LP or MILP relaxation of the combinatorial optimization problem to be solved, in our case the QAP. Additionally, polyhedral cutting plane methods make use of a class of (nontrivial) valid or facet defining inequalities known to be fulfilled by all feasible solutions of the original problem. If the solution of the relaxation is feasible for the original problem, we are done. Otherwise, some of the above mentioned valid inequalities are probably violated. In this case a 'cut' is performed, that is, one or more of the violated inequalities are added to the LP or MILP relaxation of our problem. The latter is resolved and the whole process is repeated. In the case that none of the valid inequalities is violated, but some integrality constraint is violated, the algorithm performs a branching step by fixing (feasible) integer values for the corresponding variable. The branching steps produce the search tree like in branch and bound algorithms. Each node of this tree is processed as described above by performing cuts and then by branching it, if necessary. Clearly, related elements of branch and bound algorithms like upper bounds, selection and branching rules play a role in branch and cut algorithms. Hence, such an approach combines elements of cutting plane and branch and bound methods. The main advantage of polyhedral cutting plane algorithms with respect to traditional cutting planes relies on the use of cuts which are valid for the whole polytope of the feasible solutions, and possibly facet defining. Traditional cutting planes

instead rely frequently on cuts which are not valid for the whole polytope of the feasible solutions. In this case the whole computation has to be done from scratch for different variable fixings. This requires additional running time and additional amounts of memory. Another and not less important drawback of traditional cutting plane algorithms is due to the 'weakness' of the cuts they involve. In contrast with cuts produced by facet defining inequalities, the weak cuts cannot avoid the slow convergence.

Polyhedral cutting plane methods for the QAP are not yet backed by a strong theory. However, some efforts to design branch and cut algorithms for the QAP have been made in [106] and [75]. M.W. Padberg and M.P. Rijal [106] have tested their algorithm on sparse QAP instances. The numerical results are encouraging, although the developed software is of preliminary nature, as claimed by the authors. V. Kaibel [75] has used branch and cut to compute lower bounds for QAP instances. His results are promising especially in the case where box inequalities are involved.

Heuristics

There is a large amount of research directed toward heuristic algorithms for solving the QAP. This is partially due to the fact that, although substantial improvements have been done in the development of exact algorithms for the QAP, problems of dimension $n > 20$ are still not practical to solve because of very high computer time requirements. The following types of heuristic algorithmic approaches have been applied towards the QAP:

- construction methods (CM);
- limited enumeration methods (LEM);
- improvement methods (IM);
- tabu search (TS);
- simulated annealing (SA);
- genetic algorithms (GA);
- greedy randomized adaptive search procedures (GRASP);
- ant systems (AS).

Construction Methods

Construction methods were introduced in [60]. They are iterative approaches which usually start with an empty permutation, and iteratively complete a partial permutation into a solution of the QAP by assigning some facility which has not been assigned yet to some free location.

```
PROCEDURE construction(φ0, Γ)
    φ = {};
    DO i = 1, . . . , n − 1 →
        IF (i, j) ∉ Γ →
            j = heur(i);
            update(φi, (i, j));
            Γ = Γ ∪ (i, j);
        FI;
        φ = φi;
    OD;
    RETURN(φ)
END construction;
```

Pseudocode for construction method

A generic construction method is presented in pseudocode under the name PROCEDURE construction (ϕ_0, Γ). Here ϕ_0, ϕ_1, …, ϕ_{n-1} are partial permutations, and heur(i) is some heuristic procedure that assigns facility i to some location j, and returns j. Γ is the set of already assigned pairs of facilities to locations. The procedure update constructs a permutation ϕ_i by adding the assignment (i, j) to ϕ_{i-1}. The heuristic heur(i) employed by update could be any heuristic which chooses a location j for facility i, $(i, j) \notin \Gamma$, in a greedy fashion or by applying local search. One of the oldest heuristics used in practice, the CRAFT heuristic, developed in [17], is a construction method. Another construction method which yields good results has been proposed in [100].

Limited Enumeration Methods

It has been observed that often enumeration methods (e. g. branch and bound algorithms) find good solutions in early stages of the search, and then employ a lot of time to marginally improve that solution or prove its optimality. Based on this observation, limited enumeration methods impose a limit on the enumeration process, which can be either a maximum number of iterations or time limit, to produce a heuristic solution. Another strategy which serves the same goal is to manipulate the lower bound. This can be done by increasing the lower bound if no improvement in the solution

is achieved during a large number of iterations, and would yield deeper cuts in the search tree to speed up the process. Clearly, such an approach may cut off the optimal solution and hence should be used carefully, possibly in conjunction with certain heuristics that perform elaborate searches in the feasible space.

Improvement Methods

These methods are otherwise called *local search algorithms*. For a comprehensive discussion of theoretical and practical aspects of local search in combinatorial optimization, see [1].

Basic ingredients of improvement methods are the neighborhood and the order in which the neighborhood is searched. A frequently used neighborhood for the QAP is the *k-exchange neighborhood* which we will define as follows. Let the difference between two permutations ϕ and ψ be $\delta(\phi, \psi) := \{i : \phi(i) \neq \psi(i)\}$, and define the distance between the two permutations to be $d(\phi, \psi) := |\delta(\phi, \psi)|$. The k-exchange neighborhood $N_k(\phi)$ for a permutation $\phi \in \mathcal{S}_n$ is

$$N_k(\phi) := \{\psi : \ d(\phi, \psi) \leq k, \ 2 \leq k \leq n\} \ .$$

The size of the neighborhood used in the k-exchange local search is $\binom{n}{k} = n!/k!(n-k)!$. For the QAP the most frequently used values for k are 2 and 3, with $N_2(\phi)$ producing better empirical results.

Another important ingredient of improvement methods is the order in which the neighborhood is scanned. This order can be either fixed previously or chosen at random. Given a neighborhood structure and a scanning order, a rule for the update of the current solution (from the current iteration to the subsequent one) should be chosen. The following update rules are frequently used:

- first improvement;
- best improvement;
- Heider's rule [70].

In the case of first improvement the current solution is updated as soon as the first improving neighbor solution is found. Best improvement scans the whole neighborhood and chooses the best improving neighbor solution (if such a solution exists at all). Heider's rule starts by scanning the neighborhood of the initial solution in a prespecified cyclic order. The current solution is updated as soon as an improving neighbor solution is found. The scanning of the neighborhood of the new solution starts there where the scanning of the previous one was interrupted (in the prespecified cyclic order).

Tabu Search

Tabu search was introduced in [62,63] as a technique to overcome local optimality. See [61] for a comprehensive introduction to tabu search algorithms.

Different implementations of tabu search have been proposed for the QAP, for example, a tabu search with fixed tabu list ([131]), the robust tabu search ([133]), where the size of the tabu list is randomly chosen between a maximum and a minimum value, and the reactive tabu search ([12]) which involves a mechanism for adopting the size of the tabu list. Reactive tabu search aims at improving the robustness of the algorithm. The algorithm notices when a cycle occurs, and increases the tabu list size according to the length of the detected cycle. The numerical results show that generally the reactive tabu search outperforms other tabu search algorithms for the QAP (see [12]). More recently, also parallel implementations of tabu search have been proposed, see for example, [36]. Tabu search algorithms allow a natural parallel implementation by dividing the burden of the search in the neighborhood among several processors.

Simulated Annealing

Simulated annealing exploits the analogy between combinatorial optimization problems and problems from statistical mechanics. S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi [82] and V. Černý [135] were among the first authors who recognized this analogy, and showed how the Metropolis algorithm (see [96]) used to simulate the behavior of a physical many-particle system can be applied as a heuristic for the traveling salesman problem.

Burkard and Rendl [33] showed that a simulated cooling process yields a general heuristic which can be applied to any combinatorial optimization problem, as soon as a neighborhood structure has been introduced in the set of its feasible solutions. In particular, they applied simulated annealing to the QAP. Other simulated annealing (SA) algorithms for the QAP have been proposed by different authors, see for example, [136] and [40]. All these algorithms employ the 2-exchange neighborhood. They differ on the way the cooling process or the thermal equilibrium is implemented. The

numerical experiments show that the performance of SA algorithms strongly depends on the values of the control parameters, and especially on the choice of the cooling schedule.

Genetic Algorithms

The so-called genetic algorithms (GA) are a nature inspired approach for combinatorial optimization problems. The basic idea is to adapt the evolutionary mechanisms acting in the selection process in nature to combinatorial optimization problems. The first genetic algorithm for optimization problems was proposed by Holland [53] in 1975. For a good coverage of theoretical and practical issues on genetic algorithms, see [43] and [64].

A number of authors have proposed genetic algorithms for the QAP. Standard algorithms, like the one developed in [134], have difficulties to generate the best known solutions even for QAPs of small or moderate size. Hybrid approaches, such as combinations of GA techniques with tabu search as the one developed in [52] seem to be more promising. More recently another hybrid algorithm, the so-called greedy genetic algorithm proposed in [5], produced very good results on large scale QAPs from QAPLIB [31].

Greedy Randomized Adaptive Search Procedure

The greedy randomized adaptive search procedure (GRASP) was introduced in [48] and has been applied successfully to different hard combinatorial optimization problems [49,83,84,125] and among them to the QAP [94,109,110] and the BiQAP [95]. See [48] for a survey and tutorial on GRASP, and to [117] for a comprehensive presentation of the implementation of GRASP to the QAP and related problems.

GRASP is a combination of greedy elements with random search elements in a two phase heuristic. It consists of a construction phase and a local improvement phase. In the construction phase good solutions from the available feasible space are constructed, whereas in the local improvement phase the neighborhood of the solution constructed in the first phase is searched for possible improvements. A pseudocode of GRASP is shown below. The input parameters are the size RCLsize of the restricted candidate list (RCL), a maximum number of iterations, and a random seed. RCL contains the candidates upon which the sampling related to the construction of a solution in the first phase will be performed.

```
PROCEDURE
        GRASP(RCLSize,MaxIter,RandomSeed)
    InputInstance();
    DO k = 1, . . . , MaxIter →
        ConstructSolution(RCLSize,RandomSeed);
        LocalSearch(BestSolutionFound);
        UpdateSolution(BestSolutionFound);
    OD;
    RETURN BestSolutionFound
END GRASP;
```

Pseudocode for generic GRASP

Ant Systems

Ant systems (AS) is a recently developed heuristic for combinatorial optimization problems which tries to imitate the behavior of an ant colony in search for food. AS was originally introduced in [45] and [38] and has already produced good results for well known problems like the traveling salesman problem (TSP) and the QAP [39,57]. Numerical results in [39,57] show that ant systems are competitive heuristics especially for real life instances of the QAP with a few very good solutions clustered together. For randomly generated instances which have many good solutions distributed somehow uniformly in the search space, AS are outperformed by other heuristics, that is, genetic algorithms or tabu search approaches.

Related Problems

Generalizations of the QAP appeared almost as soon as the problem itself. Specifically, Lawler [88] addressed the issue of extending to cubic, quartic, and *N-adic assignments problems* in general, in the same fashion as the LAP was extended to QAP in formulation (1). For the cubic assignment problem for example, we have n^6 cost coefficients c_{ijklmp} where $i, j, k, l, m, p = 1, \ldots, n$, and the problem is then defined to be

$$\begin{cases} \min & \sum_{i,j=1}^{n} \sum_{k,l=1}^{n} \sum_{m,p=1}^{n} c_{ijklmp} x_{ij} x_{kl} x_{mp} \\ \text{s.t} & (x_{ij}) \in \mathbf{X}_n. \end{cases}$$

As it is noted [88], we can construct an $n^3 \times n^3$ matrix S containing the cost coefficients, such that the cubic

assignment problem is equivalent to the LAP

$$\begin{cases} \min & \langle S, Y \rangle \\ \text{s.t.} & Y = X \otimes X \otimes X, \\ & X \in \mathbf{X}_n. \end{cases}$$

In an analogous way the LAP can be extended to any N-adic assignment problem, by considering the solution matrix Y to be the Kronecker Nth power of a permutation matrix in $\mathbf{X}_n$. In this section several generalizations and related problems of the QAP are presented, for which real applications have been found that initiated an interest in analyzing them and proposing solution techniques.

Biquadratic Assignment Problem

A generalization of the QAP is the biquadratic assignment problem (BiQAP), which is essentially a quartic assignment problem with cost coefficients formed by the products of two four-dimensional arrays. More specifically, consider two $n^4 \times n^4$ arrays $F = (f_{ijkl})$ and $D = (d_{mpst})$. The BiQAP can then be stated as:

$$\begin{cases} \min & \displaystyle\sum_{i,j,k,l=1}^{n} \sum_{m,p,s,t=1}^{n} f_{ijkl} d_{mpst} x_{im} x_{jp} x_{ks} x_{lt} \\ \text{s.t.} & \displaystyle\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \\ & \displaystyle\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n, \\ & x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n. \end{cases}$$

The major application of the BiQAP arises in very large scale integrated (VLSI) circuit design. A detailed description of the mathematical modeling of the VLSI problem to a BiQAP can be found in [24]. Deterministic improvement methods and variants of simulated annealing and tabu search have been developed for the BiQAP in [22]. Computational experiments on test problems of size up to $n = 32$, with known optimal solutions (a test problem generator is presented in [24]), suggest that one version of simulated annealing is best among those tested. The GRASP heuristic has also been applied to the BiQAP in [95], and produced the optimal solution for all the test problems generated in [24].

Multidimensional Assignment Problems

A close relative to the class of M-adic assignment problems is that of the *multidimensional assignment problems* (MAPs), often referred to as *multi-index assignment problems*, that also arise as natural extensions from the LAP. The general formulation of the MAP is

$$\begin{cases} \min & \displaystyle\sum_{i_1=1}^{M_1} \cdots \sum_{i_N=1}^{M_N} c_{i_1 \cdots i_N} x_{i_1 \cdots i_N} \\ \text{s.t.} & \displaystyle\sum_{i_2=1}^{M_2} \cdots \sum_{i_N=1}^{M_N} x_{i_1 \cdots i_N} = 1, \\ & \quad \text{for } i_1 = 1, \ldots, M_1, \\ & \displaystyle\sum_{i_1=1}^{M_1} \cdots \sum_{i_{k-1}=1}^{M_{k-1}} \sum_{i_{k+1}=1}^{M_{k+1}} \cdots \sum_{i_N=1}^{M_N} x_{i_1 \cdots i_N} = 1, \\ & \quad \text{for } i_k = 1, \ldots, M_k, \ k = 2, \ldots, N-1, \\ & \displaystyle\sum_{i_1=1}^{M_1} \cdots \sum_{i_{N-1}=1}^{M_{N-1}} x_{i_1 \cdots i_N} = 1, \\ & \quad \text{for } i_N = 1, \ldots, M_N, \\ & x_{i_1 \cdots i_N} \in \{0, 1\} \\ & \quad \text{for all } i_1 \cdots i_N, \end{cases}$$

with n^N cost coefficients $c_{i_1 \cdots i_N}$. A feasible solution to the above problem will be an N-dimensional permutation array. Multidimensional assignment problems in their general form have found many applications as a means of solving the data association problem. More specifically, the central problem in any multitarget tracking and multisensor surveillance is the data association problem of partitioning the observations into tracks and false alarms. General classes of these problems can be formulated as multidimensional assignment problems. For a detailed description on the application of MAPs for multiple target tracking applications, as well as for solution approaches, see [101,102,118].

Various applications are also contributed to special cases of the MAP. Specifically, the five-dimensional assignment problem has been successfully used for tracking elementary particles. By solving a five-dimensional assignment problem, physicists reconstruct tracks generated by charged elementary particles produced by the large electron-positron collider (LEP) at CERN institute [119]. The 3-index assignment problem is also a special case of the MAP.

Bottleneck QAP

In the *bottleneck quadratic assignment problem* (BQAP) of size n we are given an $n \times n$ flow matrix F and an $n \times n$ distance matrix D, and wish to find a permutation $\phi \in S_n$ which minimizes the objective function

$$\max \left\{ f_{ij} d_{\phi(i)\phi(j)} : \ 1 \le i, j \le n \right\}.$$

A more general BQAP analogous to the QAP in (2) is obtained if the coefficients of the problem are of the form c_{ijkl}, $1 \le i, j, k, l \le n$:

$$\min_{\phi \in S_n} \max_{1 \le i,j \le n} c_{ij\phi(i)\phi(j)}.$$

Besides the application in backboard wiring mentioned above, the BQAP has many other applications. Basically, all QAP applications give raise to applications of the BQAP because it often makes sense to minimize the largest cost instead of the overall cost incurred by some decision. A well studied problem in graph theory which can be modeled as a BQAP is the *bandwidth problem*. In the bandwidth problem we are given an undirected graph $G = (V, E)$ with vertex set V and edge set E, and seek a labeling of the vertices of G by the numbers 1, ..., n, where $|V| = n$, such that the minimum absolute value of differences of labels of vertices which are connected by an edge is minimized. In other words, we seek a labeling of vertices such that the maximum distance of 1-entries of the resulting adjacency matrix from the diagonal is minimized, that is, the bandwidth of the adjacency matrix is minimized. It is easy to see that this problem can be modeled as a special BQAP with flow matrix equal to the adjacency matrix of G for some arbitrary labeling of vertices, and distance matrix $D = (|i - j|)$.

The BQAP is *NP*-hard since it contains the bottleneck TSP as a special case. Some enumeration algorithms to solve BQAP to optimality have been proposed in [19]. These algorithms employ a Gilmore–Lawler-like bound for the BQAP which involves in turn the solution of bottleneck linear assignment problems. The algorithm for the general BQAP involves also a threshold procedure useful to reduce to 0 as many coefficients as possible. Burkard and Fincke [27] investigated the asymptotic behavior of the BQAP and proved results analogous to those obtained for the QAP: If the coefficients are independent random variables taken from a uniform distribution on [0, 1], then the relative difference between the worst and the best value of the objective function approaches 0 with probability tending to 0 as the size of the problem approaches infinity.

The BQAP and the QAP are special cases of a more general quadratic assignment problem which can be called the *algebraic QAP* (in analogy to the algebraic linear assignment problem (LAP) introduced in [30]). If $(H, *, \prec)$ is a totally ordered commutative semigroup with composition $*$ and order relation $\prec$, the algebraic QAP with cost coefficients $c_{ijkl} \in H$ is formulated as

$$\min_{\phi \in S_n} c_{11\phi(1)\phi(1)} * \ldots * c_{1n\phi(1)\phi(n)} * \ldots * c_{nn\phi(n)\phi(n)}.$$

The study of the bottleneck QAP and more generally the algebraic QAP was the starting point for the investigation of a number of algebraic combinatorial optimization problem with coefficients taken from linearly ordered semimodules, that is, linear assignment and transportation problems, flow problems, and other. See [34] for a detailed discussion on this topic.

Other Problems Which Can Be Formulated As QAPs

There are a number of other well known combinatorial optimization problems which can be formulated as QAPs with specific coefficient matrices. Of course, since QAP is not a well tractable problem, it does not make sense to use algorithms developed for the QAP to solve these other problems. All known solution methods for the QAP are far inferior compared to any of the specialized algorithms developed for solving these problems. However, the relationship between the QAP and these problems might be of benefit for a better understanding of the QAP and its inherent complexity.

Two well studied *NP*-hard combinatorial optimization problems which are special cases of the QAP, are the *graph partitioning problem* (GPP) and the *maximum clique problem* (MCP). In GPP we are given an (edge) weighted graph $G = (V, E)$ with n vertices and a number k which divides n. We want to partition the set V into k sets of equal cardinality such that the total weight of the edges cut by the partition is minimized. This problem can be formulated as a QAP with distance matrix D equal to the weighted adjacency matrix of G, and flow matrix F obtained by multiplying with -1 the adjacency matrix of the union of k disjoint complete

subgraphs with n/k vertices each. For more informations on graph partitioning problems, see [90]. In the maximum clique problem we are again given a graph $G = (V, E)$ with n vertices and wish to find the maximum $k \leq n$ such that there exists a subset $V_1 \subseteq V$ with $|V_1| = k$, which induces a clique in G, that is, all vertices of V_1 are connected by edges of G. In this case consider a QAP with distance matrix D equal to the adjacency matrix of G and flow matrix F given as adjacency matrix of a graph consisting of a clique of size k and $n - k$ isolated vertices, multiplied by -1. A clique of size k in G exists only if the optimal value of the corresponding QAP is $-k^2$. For a review on the maximum clique problem, see [114].

The traveling salesman problem (TSP) is another well known combinatorial optimization problem which is *NP*-hard, and much research has been devoted to finding efficient algorithms that will provide near-optimal solutions. In the TSP we are given a set of cities and the distances between them, and our task is to find the optimal tour that will visit each city once and will minimize the total distance traveled. In formulating the TSP as a QAP the distance matrix D is the corresponding distance matrix of the TSP, and the flow matrix F is the adjacency matrix of a complete cycle of length n. Without loss of generality the distance matrix D is considered to be symmetric. A complete cycle or tour is then defined by a permutation ϕ. The traveling salesman problem (TSP) is a notorious *NP*-hard combinatorial optimization problem. Among the abounding literature on the TSP, [89] is a comprehensive reference.

In the *linear arrangement problem* we are given a graph $G = (V, E)$ and wish to place its vertices at the points $1, \ldots, n$ on the line so as to minimize the sum of pairwise distances between vertices of G which are joined by some edge. If we consider the more general version of weighted graphs than we obtain the *backboard wiring problem*. This is an *NP*-hard problem as mentioned in [58]. It can be formulated as a QAP with distance matrix the (weighted) adjacency matrix of the given graph, and flow matrix $F = (f_{ij})$ given by $f_{ij} = |i - j|$, for all i, j. In the *minimum weight feedback arc set problem* (FASP) a weighted digraph $G = (V, E)$ with vertex set V and arc set E is given. The goal is to remove a set of arcs from E with minimum overall weight, such that all directed cycles, so-called *dicycles*, in G are destroyed and an acyclic directed subgraph remains. Clearly, the minimum weight feedback arc set problem is equivalent to the problem of finding an acyclic subgraph of G with maximum weight. The unweighted version of the FASP, that is a FASP where the edge weights of the underlying digraph equal 0 or 1, is called the *acyclic subdigraph problem* and is treated extensively in [74]. An interesting application of the FASP is the so-called triangulation of input-output tables which arises along with input-output analysis in economics used to forecast the development of industries, see [91]. For details and a concrete description of the application of triangulation results in economics, see [41] and [122]. Since the vertices of an acyclic subdigraph can be labeled topologically, that is, such that in each arc the label of its head is larger than that of its tail, the FASP can be formulated as a QAP. The distance matrix of the QAP is the weighted adjacency matrix of G and the flow matrix $F = (f_{ij})$ is a lower triangular matrix, that is, $f_{ij} = -1$ if $i \leq j$ and $f_{ij} = 0$, otherwise. The FASP is well known to be *NP*-hard (see [58,79]).

Another well known *NP*-hard problem which can be formulated as a QAP is the *graph packing problem* (cf. [16]). The graph packing problem can be formulated as a QAP with distance matrix equal to the adjacency matrix of G_2 and flow matrix equal to the adjacency matrix of G_1. A packing of G_2 into G_1 exists if and only if the optimal value of this QAP is equal to 0. In the positive case the optimal solution of the QAP determines a packing.

QAP Problem Generators

Since the QAP is a very hard problem from a practical point of view, often heuristics are the only reasonable approach to solve it, and so far there exists no performance guarantees for any of the algorithms developed for the QAP. One possibility to evaluate the performance of heuristics and to compare different heuristics is given by QAP instances with known optimal solution. Heuristics are applied to these instances and the heuristic solution is compared to the optimal one known before hand. The instances with known optimal solution should ideally have two properties: first, they should be representative in terms of their hardness, and secondly, they should not be especially easy for any of the heuristics.

Two generators of QAP instances with known optimal solution have been proposed so far: *Palubeckis' generator* [107] and the *Li–Pardalos generator* [92].

The first method for generating QAP instances with a known optimal solution was proposed by G.S. Palubeckis [107] in 1988. The input of the Palubeckis' algorithm consists of the size n of the instance to be generated, the optimal solution (permutation) ϕ of the output instance, two control parameters w and z, where $z < w$, and the distance matrix A of an $r \times s$ grid with $rs = n$. A contains rectilinear distances also called *Manhattan distances*, that is, the distance a_{ij} between two given knots i, j lying in rows r_i, r_j and in columns c_i, c_j, respectively, is given by $a_{ij} = |r_i - r_j| + |c_i - c_j|$. The output of the algorithm is a second matrix B such that ϕ is an optimal solution of QAP(A, B). The idea is to start with a matrix B such that QAP(A, B) is a trivial instance with optimal solution ϕ. Then B is transformed such that QAP(A, B) is not any more trivial, but ϕ continues to be its optimal solution.

Palubeckis starts with a constant matrix $B = (b_{ij})$ with $b_{ij} = w$. QAP(A, B) is a trivial problem because all permutations yield the same value of the objective function and thus, are optimal solutions. Hence, also the identity permutation id is an optimal solution of QAP(A, B). Then matrix B is iteratively transformed so that it is not a constant matrix any more and the identity permutation remains an optimal solution of QAP(A, B). In the last iteration the algorithm constructs an instance QAP(A', B) with optimal solution ϕ with the help of QAP(A, B) with optimal solution the identity permutation id, by setting $A' = (a^{\phi(i)\phi(j)})$. The optimal value of QAP(A', B) equals $w \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}$. D. Cyganski, R.F. Vaz and V.G. Virball [42] have observed that the QAP instances generated by Palubeckis' generator are 'easy' in the sense that their optimal value can be computed in polynomial time by solving a linear program.

Another generator of QAP instances with known solution has been proposed by Li and Pardalos [92]. As Palubeckis' generator, Li and Pardalos starts with a trivial instance QAP(A, B) with the identity permutation id as optimal solution and iteratively transforms A and B so that the resulting QAP instance still has the optimal solution id but is not trivial any more. The transformations are such that for all i, j, i', j', $a_{ij} \geq a_{i'j'}$ is equivalent to $b_{ij} \leq b'_{ij}$ at the end of each iteration.

If the coefficient matrices are considered as weighted adjacency matrices of graphs, each iteration transforms entries corresponding to some specific subgraph equipped with signs on the edges and hence called sign-subgraphs. The application of the Li–Pardalos algorithm with different sign-subgraphs yields different QAP generators. A number of generators involving different sign-subgraphs, for example, subgraphs consisting of a single edge, signed triangles and signed spanning trees have been tested. It is perhaps interesting and surprising that QAP instances generated by involving more complex sign-subgraphs are generally 'easier' than those generated by involving subgraphs consisting of single edges. Here a QAP instance is considered to be 'easy', if most heuristics applied to it find a solution near to the optimal one in a relatively short time. Nothing is known about the complexity of QAP instances generated by the Li–Pardalos generator, since the arguments used to analyze Palubeckis' generator do not apply in this case.

Surveys and Books

In this concluding section a list of survey articles and books which cover all the aspects of the QAP in depth is given.

One of the early survey articles is [51] where the eigenvalue based lower bounds for the QAP are introduced. The survey papers [20,112] and [25] cover every aspect of the QAP. Specifically, the article [25] is the most recent one, and the most comprehensive. A collection of articles with theoretical and algorithmic contributions for the QAP can be found in the book [113]. The book [35] has a comprehensive introduction on the QAP, and focuses on special cases of the QAP which can be solved in polynomial time. Finally the book [106] focuses on polyhedral aspects of the QAP.

See also

- ► Assignment and Matching
- ► Assignment Methods in Clustering
- ► Bi-objective Assignment Problem
- ► Communication Network Assignment Problem
- ► Complexity Theory: Quadratic Programming
- ► Feedback Set Problems
- ► Frequency Assignment Problem
- ► Generalized Assignment Problem

- ▶ Graph Coloring
- ▶ Graph Planarization
- ▶ Greedy Randomized Adaptive Search Procedures
- ▶ Linear Ordering Problem
- ▶ Maximum Partition Matching
- ▶ Quadratic Fractional Programming: Dinkelbach Method
- ▶ Quadratic Knapsack
- ▶ Quadratic Programming with Bound Constraints
- ▶ Quadratic Programming Over an Ellipsoid
- ▶ Quadratic Semi-assignment Problem
- ▶ Standard Quadratic Optimization Problems: Algorithms
- ▶ Standard Quadratic Optimization Problems: Applications
- ▶ Standard Quadratic Optimization Problems: Theory

References

1. Aarts E, Lenstra JK (eds) (1997) Local search in combinatorial optimization. Wiley, New York
2. Adams WP, Johnson TA (1994) Improved linear programming-based lower bounds for the quadratic assignment problem. In: Pardalos PM, Wolkowicz H (eds) Quadratic Assignment and Related Problems. DIMACS. Amer. Math. Soc., Providence, pp 43–75
3. Adams WP, Sherali HD (1986) A tight linearization and an algorithm for zero-one quadratic programming problems. Managem Sci 32(10):1274–1290
4. Adams WP, Sherali HD (1990) Linearization strategies for a class of zero-one mixed integer programming problems. Oper Res 38(2):217–226
5. Ahuja RK, Orlin JB, Tivari A (1995) A greedy genetic algorithm for the quadratic assignment problem. Techn Report 3826-95, Sloan School Management
6. Arora S, Frieze A, Kaplan H (1996) A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. In: Proc. 37-th Annual IEEE Symp. Foundations of Computer Sci. (FOCS). IEEE, New York, pp 21–30
7. Balas E, Mazzola JB (1980) Quadratic 0-1 programming by a new linearization. In: Proc TIMS/ORSA, May 1980
8. Balas E, Mazzola JB (1984) Nonlinear programming: I. Linearization techniques. Math Program 30:1–21
9. Balas E, Mazzola JB (1984) Nonlinear programming: II. Dominance relations and algorithms. Math Program 30:22–45
10. Balas E, Qi L (1993) Linear-time separation algorithms for the three-index assignment polytope. Discrete Appl Math (1993):1–12
11. Balas E, Saltzman MJ (1989) Facets of the three-index assignment polytope. Discrete Appl Math (1989):201–229
12. Battiti R, Tecchiolli G (1994) The reactive tabu search. ORSA J Comput (1994):126–140
13. Bazaraa MS, Sherali HD (1980) Bender's partitioning scheme applied to a new formulation of the quadratic assignment problem. Naval Res Logist Quart 27:29–41
14. Bazaraa MS, Sherali HD (1982) On the use of exact and heuristic cutting plane methods for the quadratic assignment problem. J Oper Res Soc 33:991–1003
15. Birkoff G (1946) Tres observaciones sobre el algebra lineal. Univ Nac Tucuman Rev (A):147–151
16. Bollobás B (1978) Extremal graph theory. Acad. Press, New York
17. Buffa ES, Armour GC, Vollmann TE (1962) Allocating facilities with CRAFT. Harvard Business Rev 42:136–158
18. Burkard RE (1973) Die Störungsmethode zur Lösung quadratischer Zuordnungsprobleme. Oper Res Verfahren 16:84–108
19. Burkard RE (1974) Quadratische Bottleneckprobleme. Oper Res Verfahren 18:26–41
20. Burkard RE (1991) Locations with spatial interactions: the quadratic assignment problem. In: Mirchandani PB, Francis RL (eds) Discrete Location Theory. Wiley, New York
21. Burkard RE, Bönniger T (1983) A heuristic for quadratic Boolean programs with applications to quadratic assignment problems. Europ J Oper Res 13:374–386
22. Burkard RE, Çela E (1995) Heuristics for biquadratic assignment problems and their computational comparison. Europ J Oper Res 83:283–300
23. Burkard RE, Çela E, Demidenko VM, Metelski NN, Woeginger GJ (1997) Perspectives of easy and hard cases of the quadratic assignment problems. Techn Report SFB 104, Techn Univ Graz
24. Burkard RE, Çela E, Klinz B (1994) On the biquadratic assignment problem. In: Pardalos PM, Wolkowicz H (eds) Quadratic assignment and related problems. DIMACS. Amer. Math. Soc., Providence, pp 117–146
25. Burkard RE, Çela E, Pardalos PM, Pitsoulis LS (1998) The quadratic assignment problem. In: Du D-Z and Pardalos PM (eds) Handbook Combinatorial Optim., vol 3. Kluwer, Dordrecht, pp 241–337
26. Burkard RE, Çela E, Rote G, Woeginger GJ (1995) The quadratic assignment problem with an Anti-Monge and a Toeplitz matrix: Easy and hard cases. Techn Report SFB 34, Techn Univ Graz
27. Burkard RE, Fincke U (1982) On random quadratic bottleneck assignment problems. Math Program 23:227–232
28. Burkard RE, Fincke U (1983) The asymptotic probabilistic behaviour of quadratic sum assignment problems. Z Oper Res 27:73–81
29. Burkard RE, Fincke U (1985) Probabilistic asymptotic properties of some combinatorial optimization problems. Discrete Appl Math 12:21–29
30. Burkard RE, Hahn W, Zimmermann U (1977) An algebraic approach to assignment problems. Math Program 12:318–327

31. Burkard RE, Karisch S, Rendl F (1997) QAPLIB – Aquadratic assignment problem library. J Global Optim 10:391–403
32. Burkard RE, Klinz B, Rudolf R (1996) Perspectives of Monge properties in optimization. Discrete Appl Math 70:95–161
33. Burkard RE, Rendl F (1984) A thermodynamically motivated simulation procedure for combinatorial optimization problems. Europ J Oper Res 17:169–174
34. Burkard RE, Zimmermann U (1982) Combinatorial optimization in linearly ordered semimodules: A survey. Modern Applied Mathematics. North-Holland, Amsterdam, pp 392–436
35. Çela E (1998) The quadratic assignment problem: Theory and algorithms. Kluwer, Dordrecht
36. Chakrapani J, Skorin-Kapov J (1993) Massively parallel tabu search for the quadratic assignment problem. Ann Oper Res 41:327–342
37. Christofides N (1976) Worst case analysis of a new heuristic for the traveling salesman problem. Grad School Industr Admin Carnegie-Mellon Univ 338
38. Colorni A, Dorigo M, Maniezzo V (1996) The ant system: optimization by a colony of cooperating agents. IEEE Trans Syst, Man Cybern Part B 26:29–41
39. Colorni A, Maniezzo V (1998) The ant system applied to the quadratic assignment problem. IEEE Trans Knowledge and Data Enginto
40. Connolly DT (1990) An improved annealing scheme for the QAP. Europ J Oper Res 46:93–100
41. Conrad K (1971) Das quadratische Zuweisungsproblem und zwei seiner Spezialfalle. Mohr-Siebeck, Tübingen
42. Cyganski D, Vaz RF, Virball VG (1994) Quadratic assignment problems with the Palubeckis' algorithm are degenerate. IEEE Trans Circuits and Systems I 41:481–484
43. Davis L (1987) Genetic algorithms and simulated annealing. Pitman, Boston
44. Deneko VG, Woeginger GJ (1996) A solvable case of the quadratic assignment problem. Techn Report SFB 88,Inst Math, Techn Univ Graz
45. Dorigo M (1992) Optimization, learning, and natural algorithms. PhD Thesis, Dip. Elettronica e Informazione Politecn. Milano. (In Italian)
46. Dyer ME, Frieze AM, McDiarmid CJH (1986) On linear programs with random costs. Math Program 35:3–16
47. Edwards CS (1980) A branch and bound algorithm for the Koopmans–Beckman quadratic assignment problem. Math Program Stud 13:35–52
48. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Global Optim 6:109–133
49. Feo TA, Resende MGC, Smith SH (1994) A greedy randomized adaptive search procedure for maximum independent set. Oper Res 42:860–878
50. Finke G, Burkard RE, Rendl F (1984) Eigenvalue approach to quadratic assignment problems. In: 5th Symp. Oper. Res., 1984
51. Finke G, Burkard RE, Rendl F (1987) Quadratic assignment problems. Ann Discret Math 31:61–82
52. Fleurent C, Ferland J (1994) Genetic hybrids for the quadratic assignment problem. In: Pardalos PM, Wolkowicz H (eds) Quadratic assignment and related problems. DIMACS. Amer. Math. Soc., Providence pp 43–75
53. Floudas CA, Pardalos PM (1990) A collection of test problems for constrained global optimization algorithms. Lecture Notes Computer Sci, vol 455. Springer, Berlin
54. Frenk JCB, van Houweninge M and, Rinnooy Kan AHG (1985) Asymptotic properties of assignment problems. Math Oper Res 10:100–116
55. Frieze AM (1974) A bilinear programming formulation of the 3-dimensional assignment problem. Math Program 7:376–379
56. Frieze AM, Yadegar J (1983) On the quadratic assignment problem. Discrete Appl Math 5:89–98
57. Gambardella LM, Taillard ED, Dorigo M (1997) Ant colonies for the QAP. Techn Report IDSIA-4-97, Ist dalle Molle Di Studi sull'Intelligenza Artificiale Lugano
58. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, New York
59. Gavett JW, Plyter NV (1966) The optimal assignment of facilities to locations by branch and bound. Oper Res 14:210–232
60. Gilmore PC (1962) Optimal and suboptimal algorithms for the quadratic assignment problem. SIAM J Appl Math 10:305–313
61. (1993) Tabu search. Ann Oper Res 41
62. Glover F (1989) Tabu search – Part I. ORSA J Comput 1:190–206
63. Glover F (1990) Tabu search – Part II. ORSA J Comput 2:4–32
64. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading
65. Hadley SW (1989) Continuous optimization approaches for the quadratic assignment problem. PhD Thesis, Univ. Waterloo
66. Hadley SW, Rendl F, Wolkowicz H (1990) Bounds for the quadratic assignment problem using continuous optimization techniques. Integer Programming and Combinatorial Optimization. Univ. Waterloo Press, Waterloo, pp 237–248
67. Hadley SW, Rendl F, Wolkowicz H (1992) A new lower bound via projection for the quadratic assignment problem. Math Oper Res 17(3):727–739
68. Hadley SW, Rendl F, Wolkowicz H (1992) Nonsymmetric quadratic assignment problems and the Hoffman–Wielandt Inequality. Linear Alg & its Appl 58:109–124
69. Hardy GG, Littlewood JE, Polya G (1952) Inequalities. Cambridge Univ. Press, Cambridge
70. Heider CH (1972) A computationally simplified pair exchange algorithm for the quadratic assignment problem. Paper 101, Center Naval Anal, Arlington

71. Jansen B (1993) A note on lower bounds for the QAP. Techn Report (dec), Delft Univ Techn Math and Computer Sci
72. Johnson DS, Papadimitriou CH, Yannakakis M (1988) How easy is local search? J Comput Syst Sci 37:79–100
73. Johnson TA (1992) New linear programming-based solution procedures for the quadratic assignment problem. PhD Thesis, Clemson Univ.
74. Jünger M (1985) Polyhedral combinatorics and the acyclic subdigraph problem. Heldermann, Berlin
75. Kaibel V (1997) Polyhedral combinatorics of the quadratic assignment problem. PhD Thesis, Univ. Köln
76. Karisch SE (1995) Nonlinear approaches for quadratic assignment and graph partition problems. PhD Thesis, Techn. Univ. Graz
77. Karisch SE, Rendl F, Wolkowicz H, Zhao Q (1998) Semidefinite programming relaxations for the quadratic assignment problem. J Combin Optim 2(1):71–109
78. Karisch SE, Rendl F, Wolkowicz H (1994) Trust regions and the quadratic assignment problem. In: Pardalos PM, Wolkowicz H (eds) Quadratic assignment and related problems. DIMACS. Amer. Math. Soc., Providence, pp 199–220
79. Karp R (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Proc. Complexity of Computer Computations. Plenum, New York, pp 85–104
80. Kaufman L, Broeckx F (1978) An algorithm for the quadratic assignment problem using Benders' decomposition. Europ J Oper Res 2:204–211
81. Kernighan B, Lin S (1972) An efficient heuristic procedure for partitioning graphs. Bell Systems J 49:291–307
82. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
83. Klincewicz JG (1992) Avoiding local optima in the p-hub location problem using tabu search and GRASP. Ann Oper Res 40:283–302
84. Klincewicz JG, Rajan A (1992) Using GRASPto solve the component grouping problem. Techn Report, AT&T Bell Lab
85. Koopmans TC, Beckmann MJ (1957) Assignment problems and the location of economic activities. Econometrica 25:53–76
86. Land AM (1963) A problem of assignment with interrelated costs. Oper Res Quart 14:185–198
87. Laporte G, Mercure H (1988) Balancing hydraulic turbine runners: A quadratic assignment problem. Europ J Oper Res 35:378–382
88. Lawler EL (1963) The quadratic assignment problem. Managem Sci 9:586–599
89. Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (1985) The traveling salesman problem: A guided tour of combinatorial optimization. Wiley, New York
90. Lengauer T (1990) Combinatorial algorithms for intergrated circuit layout. Wiley, New York
91. Leontief W (1966) Input–output economics. Oxford Univ. Press, Oxford
92. Li Y, Pardalos PM (1992) Generating quadratic assignment test problems with known optimal permutations. Comput Optim Appl 1(2):163–184
93. Li Y, Pardalos PM, Ramakrishnan KG, Resende MGC (1994) Lower bounds for the quadratic assignment problem. Ann Oper Res 50:387–410
94. Li Y, Pardalos PM, Resende MGC (1994) A greedy randomized adaptive search procedure for the quadratic assignment problem. In: Pardalos PM, Wolkowicz H (eds) Quadratic assignment and related problems. DIMACS 16. Amer. Math. Soc., Providence, pp 237–261
95. Mavridou T, Pardalos PM, Pitsoulis LS, Resende MGC (1998) A GRASPfor the biquadratic assignment problem. Europ J Oper Res 105(3):613–621
96. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equations of state calculations by fast computing machines. J Chem Phys 21:1087–1092
97. Mirchandani PB, Obata T (1979) Locational decisions with interactions between facilities: the quadratic assignment problem a review. Working Paper May Ps-79-1, Rensselaer Polytechnic Inst, Troy, New York
98. Mirsky L (1956) The spread of a matrix. Mathematika 3:127–130
99. Mosevich J (1986) Balancing hydraulic turbine runners – a discrete combinatorial optimization problem. Europ J Oper Res 26:202–204
100. Muller-Merbach H (1970) Optimale Reihenlorgen. Springer, Berlin
101. Murphey RA, Pardalos PM, Pitsoulis L (1997) A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. In: Pardalos PM, Du D-Z (eds) Network Design: Connectivity and Facility Location. DIMACS 40. Amer. Math. Soc., Providence, pp 277–302
102. Murphey RA, Pardalos PM, Pitsoulis L (1998) A parallel GRASP for the data association multidimensional assignment problem. Parallel Processing of Discrete Problems. In: IMA vol Math Appl, vol 106. Springer, Berlin, pp 159–180
103. Murthy KA, Pardalos PM, Li Y (1992) A local search algorithm for the quadratic assignment problem. Informatica 3(4):524–538
104. Murty KG (1968) An algorithm for ranking all the assignments in order of increasing cost. Oper Res 16:682–287
105. Nugent CE, Vollmann TE, Ruml J (1969) An experimental comparison of techniques for the assignment of facilities to locations. J Oper Res 16:150–173
106. Padberg MW, Rijal MP (1996) Location, scheduling, design and integer programming. Kluwer, Dordrecht
107. Palubetskis GS (1988) Generation of quadratic assignment test problems with known optimal solutions. Zh Vychisl Mat Mat Fiz 28(11):1740–1743. (In Russian.)

108. Papadimitriou CH, Wolfe D (1985) The complexity of facets resolved. In: Proc Foundations Computer Sci, pp 74–78
109. Pardalos PM, Pitsoulis LS, Resende MGC (1995) A parallel GRASP implementation for solving the quadratic assignment problem. In: Ferreira A and Rolim JDP (eds) Parallel Algorithms for Irregular Problems: State of the Art. Kluwer, Dordrecht, pp 115–133
110. Pardalos PM, Pitsoulis LS, Resende MGC (1997) Algorithm 769: FORTRAN subroutines for approximate solution of sparse quadratic assignment problems. ACM Trans Math Softw 23:196–208
111. Pardalos PM, Ramakrishnan KG, Resende MGC, Li Y (1997) Implementation of a variable reduction based lower bound in a branch and bound algorithm for the quadratic assignment problem. SIAM J Optim 7(1):280–294
112. Pardalos PM, Rendl F, Wolkowicz H (1994) The quadratic assignment problem: A survey and recent developments. In: Pardalos PM, Wolkowicz H (eds) Quadratic assignment and related problems. DIMACS vol 16. Amer. Math. Soc., Providence, pp 1–42
113. Pardalos PM, Wolkowicz H (1994) Quadratic assignment and related problems. DIMACS, vol 16. Amer. Math. Soc., Providence
114. Pardalos PM, Xue Jue (1994) The maximum clique problem. J Global Optim 4:301–328
115. Pierskalla WP (1967) The tri-substitution method for the three-multidimensional assignment problem. CORS J 5:71–81
116. Pierskalla WP (1968) The multidimensional assignment problem. Oper Res 16:422–431
117. Pitsoulis LS (1998) Algorithms for nonlinear assignment problems. PhD Thesis, Dept. Industr. Systems Engin., Univ. Florida
118. Poore AB, Rijavec N (1994) Partitioning multiple data sets: multidimensional assignments and Lagrangian relaxation. In: Pardalos PM, Wolkowicz H (eds) Quadratic assignment and related problems. DIMACS vol 16. Amer. Math. Soc., Providence, pp 317–342
119. Pusztaszeri J, Rensing PE, Liebling TM (1995) Tracking elementary particles near their primary vertex: A combinatorial approach. J Global Optim 16:422–431
120. Qi L, Balas E, Gwan G (1994) A new facet class and a polyhedral method for the three-index assignment problem. In: Du D-Z (ed) Advances in Optimization. Kluwer, Dordrecht, pp 256–274
121. Queyranne M (1986) Performance ratio of heuristics for triangle inequality quadratic assignment problems. Oper Res Lett 4:231–234
122. Reinelt G (1985) The linear ordering problem: Algorithms and applications Res and Exposition in Math, vol 8. Heldermann, Berlin
123. Rendl F (1985) Ranking scalar products to improve bounds for the quadratic assignment problem. Europ J Oper Res 20:363–372
124. Rendl F, Wolkowicz H (1992) Applications of parametric programming and eigenvalue maximization to the quadratic assignment problem. Math Program 53:63–78
125. Resende MGC, Pitsoulis LS, Pardalos PM (1997) Approximate solution of weighted MAX-SAT problems using GRASP. In: Pardalos PM Resende MGC, Du DZ (eds) The Satisfiability Problem. DIMACS vol 35. Amer. Math. Soc., Providence, pp 393–405
126. Rhee WT (1989) A note on asymptotic properties of the quadratic assigment problem. Oper Res Lett 4:197–200
127. Rhee WT (1991) Stochastic analysis of the quadratic assignment problem. Math Oper Res 2:223–239
128. Roucairol C (1979) A reduction method for quadratic assignment problems. Oper Res Verfahren 32:183–187
129. Sahni S, Gonzalez T (1976) P-complete approximation problems. J ACM 23:555–565
130. Schäffer AA, Yannakakis M (1991) Simple local search problems that are hard to solve. SIAM J Comput 20:56–87
131. Skorin-Kapov J (1990) Tabu search applied to the quadratic assignment problem. ORSA J Comput 2(1):33–45
132. Szpankowski W (1995) Combinatorial optimization problems for which almost every algorithm is asymptotically optimal! Optim 33:359–367
133. Taillard E (1991) Robust tabu search for the quadratic assignment problem. Parallel Comput 17:443–455
134. Tate DM, Smith AE (1995) A genetic approach to the quadratic assignment problem. Comput Oper Res 22:73–83
135. Černý V (1985) Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. J Optim Th Appl 45:41–51
136. Wilhelm MR, Ward TL (1987) Solving quadratic assignment problems by simulated annealing. IEEE Trans 19(1):107–119
137. Zhao Q (1996) Semidefinite programming for assignment and partitioning problems. PhD Thesis, Univ. Waterloo

Quadratic Fractional Programming: Dinkelbach Method

ANDREW T. PHILLIPS
Computer Sci. Department,
University Wisconsin–Eau Claire, Eau Claire, USA

MSC2000: 90C32

Article Outline

Keywords
Problem Formulation and Properties

Bounds and Convergence Rates
Special Cases
See also
References

Keywords

Global optimization; Fractional programming; Dinkelbach method

Given two continuous functions $f : \mathbf{R}^n \to \mathbf{R}$ and $g\mathbf{R}^n \to \mathbf{R}$ defined on a polyhedral set $S \subseteq \mathbf{R}^n$ such that $g(x) > 0$ for all $x \in S$, the *fractional programming problem* is to find some point x^* which satisfies

$$\frac{f(x^*)}{g(x*)} = \max_{x \in S} \frac{f(x)}{g(x)} \,. \tag{1}$$

Applications and algorithms for fractional programs have been treated in considerable detail since the early work of J.R. Isbell and W.H. Marlow [5]. Included among the many applications are portfolio selection, stock cutting, game theory, and numerous decision problems in management science. See [3] for work known to up to 1971 and [1,4,12,13] for the most recent surveys.

If $f(x)$ is concave and nonnegative and $g(x)$ and S are convex (and S is bounded), then (1) is called a concave-convex fractional program. It was shown in [10] that such problems can be solved by a single concave problem using a simple variable transformation. This provides an efficient approach for solving a limited class of fractional programming problems. Unfortunately, even in some of the simplest cases (for example when $f(x)$ and $g(x)$ are quadratic) a new constraint, which may be nonlinear, must be added (to the transformed feasible region), and the transformed problem becomes very difficult to solve. In addition, if the problem is not concave-convex initially, then the transformation does not even necessarily yield a concave problem. In fact, in the most general case, Eq (1) may have many local maxima which are different from the optimal one, and hence determining the global maximum is very difficult (i. e., *NP*-hard).

A different and more recent method is to replace the nonlinear functions by suitable linear underestimators and then obtain the global optimum by a vertex ranking procedure. This method, due to P.M. Pardalos [6], is applicable only when $f(x)$ is a convex quadratic function and $g(x)$ is linear (hence the ratio is quasiconvex).

Another well-known approach, and one of the oldest, is to consider the global optimization problem

$$\max_{x \in S} f(x) - \lambda g(x) \,, \tag{2}$$

where $\lambda \in \mathbf{R}$ is a constant. This 'parametric' approach, which was first proposed by W. Dinkelbach [2], generates a sequence of values λ_i that converges to the global optimum function value [11]. This method has since then been applied to many specific types of fractional programs including the concave-convex type, but very little work has been done to solve fractional programs where the ratio of two concave, two convex, or the ratio of a convex and a concave function is to be maximized. In addition, this method does not provide a sequence of improving upper bounds, and hence even though the sequence λ_i may be converging to the global optimum function value, no bound on the error is available at any iteration.

The method discussed here improves Dinkelbach's algorithm by providing a means for obtaining a sequence of improving upper bounds which, along with the corresponding sequence of improving lower bounds, will provide a bound on the error at each iteration of the solution procedure. In addition, both the sequence of lower bounds and the sequence of upper bounds converge to the global optimum function value at a 'superlinear' rate. This algorithm is also appropriate for the class of quadratic fractional programs (i. e., one or both of $f(x)$ and $g(x)$ are quadratic) where the ratio may involve concave, convex, or even indefinite terms. It combines Dinkelbach's approach with a method guaranteed to solve linearly constrained quadratic programming problems regardless of the definiteness of the quadratic from [8].

Two algorithms which are similar to the one presented here are given in [4,11]. *Schaible's method* [11] first computes a sequence of improving upper and lower bounds using an efficient section method. *Dinkelbach's algorithm* is then started as soon as the section method achieves a set of bounds that differ by some pre-specified tolerance. The algorithm presented here differs from Schaible's method in that the upper and lower bounds are continuously improving throughout the procedure. Nevertheless, in both algorithms the se-

quence of upper and lower bounds converges superlinearly.

Likewise, [4] presents a variety of related algorithms which also provide upper and lower bounds. These algorithms combine Dinkelbach's approach with various search techniques (e. g., Newton, binary, modified binary). The result is a set of related algorithms with convergence rates that vary depending on the search technique employed. T. Ibaraki [4] also provides a collection of computational results for the fractional knapsack problem and quadratic fractional programs.

Problem Formulation and Properties

The fundamental result which relates the global optimization problem (2) to the general fractional programming problem (1) is as follows: x^* solves the fractional programming problem (1) if and only if x^* solves the global optimization problem (2) with constant $\lambda^* = f(x^*)/g(x^*)$.

Dinkelbach's original iterative algorithm is based on this theorem and can be described as follows:

1	Select some $x^{(0)} \in S$. Set $\lambda^{(0)} = f(x^{(0)})/g(x^{(0)})$ and $k = 0$.
2	Solve the constrained global optimization problem (2) to get the optimal solution point $x^{(k+1)}$.
3	IF $f(x^{(k+1)}) - \lambda^{(k)} g(x^{(k+1)}) = 0$, THEN set $x^* = x^{k+1}$, $\lambda^* = \lambda^{(k)}$, STOP.
4	IF $f(x^{(k+1)}) - \lambda^{(k)} g(x^{(k+1)}) > 0$, THEN set $\lambda^{(k+1)} = f(x^{(k+1)})/g(x^{(k+1)})$ and $k = k + 1$. Go to Step 2.

Dinkelbach(S, f, g)

The efficiency of this algorithm depends on the number of times the constrained global optimization problem must be solved, and on the time spent solving it during each iteration. Also note that a test of the form $f(x^{(k+1)}) - \lambda^{(k)} g(x^{(k+1)}) < 0$ is not necessary since, for any fixed k,

$$\begin{aligned} f(x^{(k+1)}) - \lambda^{(k)} g(x^{(k+1)}) &= \max_{x \in S} f(x) - \lambda^{(k)} g(x) \\ &\geq f(x^{(k)}) - \lambda^{(k)} g(x^{(k)}) \\ &= 0 \,. \end{aligned}$$

Now consider the function $M(\lambda)$ defined as

$$M(\lambda) = \max_{x \in S} f(x) - \lambda g(x) \,. \tag{3}$$

The function $M(\lambda)$ has two interesting properties that are important in guaranteeing convergence of upper and lower bounds to λ^* and in determining the rate of this convergence. The first of these properties is that for any lower bound λ of λ^*, $M(\lambda)$ is positive, and for any upper bound λ of λ^*, $M(\lambda)$ is negative. Secondly, the function $M(\lambda)$ is convex. That is,

1) $M(\lambda) > 0$ for all $\lambda < \lambda^*$, and $M(\lambda) < 0$ for all $\lambda > \lambda^*$; and
2) $M(\lambda)$ is convex.

The sequence of iterates $\lambda^{(0)}, \lambda^{(1)}, \ldots$ generated by the algorithm Dinkelbach(S, f, g) is strictly monotone increasing, and satisfy $M(\lambda^{(i)}) > 0$ for $i = 0, 1, \ldots$ [2]. Hence, by the properties of $M(\lambda)$ listed above, they provide a strictly monotone increasing sequence of lower bounds for λ^*.

Bounds and Convergence Rates

The sequence of lower bounds $\lambda^{(i)}$ converges superlinearly to $\lambda^* \equiv f(x^*) : g(x^*) \rightarrow$ where x^* is any optimal solution for (1) as shown in [7]. However, as it now stands, the algorithm Dinkelbach(S, f, g) does not provide upper bounds on the global optimum function value λ^*. One way to obtain an initial upper bound is to solve the following two problems:

$$\max_{x \in S} f(x) \tag{3a}$$

to get the optimal solution $f(x')$, and

$$\min_{x \in S} g(x) \tag{4}$$

to get the optimal solution $g(x'')$. Then an initial upper bound is clearly given by $\gamma^{(-1)} \equiv f(x')/g(x'')$. In fact, according to the properties of M (part 1), any $\gamma \in \mathbf{R}$ satisfying $M(\gamma) < 0$ would also be an upper bound. Hence, if we define

$$\gamma^{(n)} \equiv \gamma^{(n-1)} - M(\gamma^{(n-1)}) \cdot \left(\frac{\gamma^{(n-1)} - \lambda^{(n)}}{M(\gamma^{(n-1)}) - M(\lambda^{(n)})} \right)$$

where $\lambda^{(n)}$ is the most recent lower bound of λ^* and $\gamma^{(n-1)}$ is the most recent upper bound of λ^*, then the new upper bound is given by $\gamma^{(n)}$. As Fig. 1 illustrates,

$\gamma^{(n)}$ is just the root of the line segment joining the points $(\lambda^{(n)}, M(\lambda^{(n)}))$ and $(\gamma^{(n-1)}, M(\gamma^{(n-1)}))$.

This leads to an important modification of the algorithm Dinkelbach(S, f, g):

1 Select some $x^{(0)} \in S$.
Set $\lambda^{(0)} = f(x^{(0)})/g(x^{(0)})$.
2 Solve the constrained global optimization problems (4) and (5) to get the optimal function values $f(x')$ and $g(x')$, respectively.
Set $\gamma^{(-1)} = f(x')/g(x")$ and $k = 0$.
IF $\gamma^{(-1)} - \lambda^{(0)} \leq \delta$,
THEN set $\lambda^* = \lambda^{(0)}$ and $x^* = x^{(0)}$;
STOP.
3 Solve the constrained global optimization problem

$$M(\lambda^{(k)}) = \max_{x \in S} f(x) - \lambda^{(k)} g(x) \tag{6}$$

to get the optimal solution point $x^{(k+1)}$.
4 IF $M(\lambda^{(k)}) = 0$,
THEN set $x^* = x^{(k+1)}$ and $\lambda^* = \lambda^{(k)}$;
STOP.
5 Solve the constrained global optimization problem

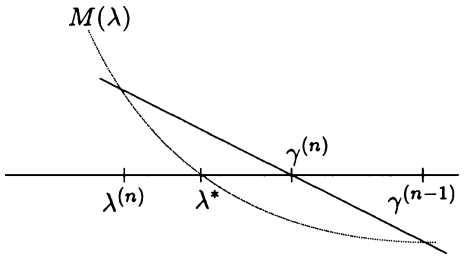
$$M(\gamma^{(k-1)}) = \max_{x \in S} f(x) - \gamma^{(k-1)} g(x) \tag{7}$$

to get the optimal solution point $y^{(k)}$.
6 IF $M(\gamma^{(k-1)}) = 0$,
THEN set $x^* = y^{(k)}$ and $\lambda^* = \gamma^{(k-1)}$;
STOP.
7 Set

$$\gamma^{(k)} = \gamma^{(k-1)} - M(\gamma^{(k-1)}) \cdot \left(\frac{\gamma^{(k-1)} - \lambda^{(k)}}{M(\gamma^{(k-1)}) - M(\lambda^{(k)})} \right).$$

8 IF $\gamma^{(k)} - \lambda^{(k)} \leq \delta$,
THEN set $\lambda^* = \lambda^{(k)}$ and $x^* = x^{(k+1)}$;
STOP.
9 Set $\lambda^{(k+1)} = f(x^{(k+1)})/g(x^{(k+1)})$ and $k = k + 1$.
Go to Step 3.

Fract(S, f, g, δ)



Quadratic Fractional Programming: Dinkelbach Method, Figure 1

Note that the parameter $\delta \geq 0$ is a user supplied stopping tolerance. The following assertion from [7] shows that the sequence iterates $\gamma^{(-1)}, \gamma^{(0)}, \gamma^{(1)}, \ldots$ is, in fact, a sequence of upper bounds on λ^*, and that the sequence is strictly monotonically decreasing:

$$\lambda^* \leq \gamma^{(i+1)} \leq \gamma^{(i)} \quad \text{for } i = -1, 0, 1, \ldots .$$

In fact, the sequence of upper bounds $\gamma^{(i)}$ also converges to λ^*, and this convergence is superlinear as well [7].

Special Cases

If the feasible set S is polyhedral and the functions $f(x)$ and $g(x)$ are either linear or quadratic, then the algorithm solves a sequence of linear or quadratic programs, respectively. In particular, if $f(x) = c^\intercal x$ and $g(x) = d^\intercal x$ then the algorithm solves the sequence of linear programs

$$\max_{x \in S} (c - \lambda^{(k)} d)^\top x \tag{8}$$

If both $f(x)$ and $g(x)$ are quadratic, i. e., $f(x) = (\frac{1}{2})\, x^\intercal Q x + c^\intercal x$ and $g(x) = (\frac{1}{2})\, x^\intercal P x + d^\intercal x$, then the algorithm solves the sequence of quadratic programs

$$\max_{x \in S} \frac{1}{2} x^\top (Q - \lambda^{(k)} P) x + (c - \lambda^{(k)} d)^\top x \,. \tag{9}$$

Notice that the matrix $(Q - \lambda^{(k)} P)$ may be indefinite, in which case the algorithm is required to find the global maximum of a linearly constrained indefinite quadratic function. Even though this is an *NP*-hard problem (e. g., when $(Q - \lambda^{(k)} P)$ is positive definite), the method developed by A.T. Phillips and J.B. Rosen [8] is guaranteed to find an ϵ-approximate global maximum (i. e., the relative error is no larger than ϵ) for any specified $\epsilon > 0$.

Furthermore, if $f(x)$ and $g(x)$ are such that $f(x) - \lambda g(x)$ is only 'partially separable', then the method developed in [9] can be used to find an ϵ-approximate global maximum for any $\epsilon > 0$. Specifically, the method in [9] is guaranteed to find solutions to the sequence of subproblems (6) and (7) if x can be partitioned into two components $x = (w, z)$ such that $f(x) - \kappa\, g(x)$ (where the constant $\kappa = \lambda^{(k)}$ or $\gamma^{(k-1)}$) can be written in the form $\phi(w) + \psi(z)$ where $\phi(w)$ is a separable convex function of w and $\psi(z)$ is a concave (but not necessarily separable) function of z. The applicability of these methods to the solution of these subproblems greatly extends the class of fractional programming problems that can be solved in practice.

See also

- ▶ Bilevel Fractional Programming
- ▶ Complexity Theory: Quadratic Programming
- ▶ Fractional Combinatorial Optimization
- ▶ Fractional Programming
- ▶ Quadratic Assignment Problem
- ▶ Quadratic Knapsack
- ▶ Quadratic Programming with Bound Constraints
- ▶ Quadratic Programming Over an Ellipsoid
- ▶ Standard Quadratic Optimization Problems: Algorithms
- ▶ Standard Quadratic Optimization Problems: Applications
- ▶ Standard Quadratic Optimization Problems: Theory

References

1. Avriel M, Diewart WE, Shaible S, Zang I (1988) Generalized concavity. Plenum, New York
2. Dinkelbach W (1967) On nonlinear fractional programming. Managem Sci 13(7):492–498
3. Grunspan M (1971) Fractional programming: A survey. Techn Report FL 50, Dept Industr Systems Engin, Univ Florida, Gainesville
4. Ibaraki T (1983) Parametric approaches to fractional programs. Math Program 26:345–362
5. Isbell JR, Marlow WH (1956) Attrition games. Naval Res Logist Quart 3:71–93
6. Pardalos PM (1986) An algorithm for a class of nonlinear fractional problems using ranking of the vertices. BIT 26:392–395
7. Pardalos PM, Phillips AT (1991) Global optimization of fractional programs. J Global Optim 1:173–182
8. Phillips AT, Rosen JB (1990) Guaranteed ϵ-approximate solution for indefinite quadratic global minimization. Naval Res Logist 37:499–514
9. Phillips AT, Rosen JB (1990) A parallel algorithm for partially separable non-convex global minimization. Ann Oper Res 25:101–118
10. Schaible S (1976) Fractional programming I, duality. Managem Sci 22(8):858–867
11. Schaible S (1976) Fractional programming II, on Dinkelbach's algorithm. Managem Sci 22(8):868–873
12. Schaible S (1981) Fractional programming: Applications and algorithms. Europ J Oper Res 12:325–338
13. Stancu-Minasian IM (1997) Fractional programming. Kluwer, Dordrecht

Quadratic Integer Programming: Complexity and Equivalent Forms

W. Art Chaovalitwongse[1], Ioannis P. Androulakis[2], Panos M. Pardalos[3]

[1] Department of Industrial and Systems Engineering, Rutgers University, Piscataway, USA

[2] Department of Biomedical Engineering, Rutgers University, Piscataway, USA

[3] Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA

MSC2000: 65K05, 90C11, 90C20

Article Outline

Keywords and Phrases
Introduction
Complexity of Quadratic Programming
Equivalence Between Discrete and Continuous Problems
Integer Programming Problems and Complementarity Problems
Integer Programming Problems and Quadratic Integer Programming Problems
Various Equivalent Forms of Quadratic Zero-One Programming Problems
Complexity of Quadratic Zero-One Programming Problems
k-clique Problem
Quadratic Zero-One Programming and Mixed Integer Programming
Quadratic Zero-One Programming and Mixed Integer Programming
A New Linearization Approach
References

Keywords and Phrases

Quadratic zero-one programming; Indefinite quadratic programming; Complexity; Optimality conditions

Introduction

In this paper we consider a quadratic programming (QP) problem of the following form:

$$\begin{aligned} \min \quad & f(x) = \frac{1}{2}x^{\mathrm{T}}Qx + c^{\mathrm{T}}x \\ \text{s.t.} \quad & x \in D \end{aligned} \tag{1}$$

where D is a polyhedron in $\mathbb{R}^n$, $c \in \mathbb{R}^n$. Without any loss of generality, we can assume that Q is a real symmetric $(n \times n)$-matrix. If this is not the case, then the matrix Q can be converted to symmetric form by replacing Q by $(Q + Q^{\mathrm{T}})/2$, which does not change the value of the objective function $f(x)$. Note that if Q is positive semidefinite, then Problem (1) is considered to be a convex minimization problem. When Q is negative semidefinite, Problem (1) is considered to be a concave minimization problem. When Q has at least one positive and one negative eigenvalue (i. e., Q is indefinite), Problem (1) is considered to be an indefinite quadratic programming problem. We know that in the case of convex minimization problem, every Kuhn-Tucker point is a local minimum, which is also a global minimum. In this case, there are a number of classical optimization methods that can obtain the globally optimal solutions of quadratic convex programming problems. These methods can be found in many places in the literature. In the case of concave minimization over polytopes, it is well known that if the problem has an optimal solution, then an optimal solution is attained at a vertex of D. On the other hand, the global minimum is not necessarily attained at a vertex of D for infinite quadratic programming problems. In this case, from second order optimality conditions, the global minimum is attained at the boundary of the feasible domain. In this research, without loss of generality, we are interested in developing solution techniques to solve general (convex, concave and indefinite) quadratic programming problems.

Complexity of Quadratic Programming

In this section we discuss the complexity of quadratic programming problems. The complexity analysis can give an idea of the possibility of developing efficient algorithms for solving the problem. In [10], the QP was shown to be $\mathcal{NP}$-hard in the case of a negative definite matrix Q. The QP was also proven to be $\mathcal{NP}$-hard by reduction to the satisfiability problem [11], and reduction to the knapsack feasibility problem [5]. Moreover, it has also been shown that checking local optimality for the QP itself is an $\mathcal{NP}$-hard problem [11]. In addition, checking for strict convexity (checking local optimality as part of the second order necessary conditions) in the QP was proven to be $\mathcal{NP}$-hard [8]. In fact, finding a local minimum and proving local optimality of such a solution to the QP may take exponential time. This is true even in the case of a small number of concave variables. For instance, although the matrix Q is of rank one with exactly one negative eigenvalue, the QP is still $\mathcal{NP}$-hard [9]. However, a large number of negative eigenvalues does not necessarily make the problem harder to solve. For example, consider the following problem:

$$\begin{aligned} \min \quad & \frac{1}{2}x^{\mathrm{T}}Qx + c^{\mathrm{T}}x \\ \text{s.t.} \quad & x \geq 0\,. \end{aligned}$$

If the matrix Q has $(n-1)$ negative eigenvalues, then there must be at least $(n-1)$ active constraints at the optimal solution [3]. Correspondingly, it is sufficient to solve $(n-1)$ different problems, in each case setting $(n-1)$ of the constraints to equalities, to find the optimal solution. In general, if the matrix Q has $(n-k)$ negative eigenvalues, then we are required to solve $\frac{n!}{k!(n-k)!}$ independent problems. In addition, the total computational time required to solve this problem is proportional to $\frac{k^3c^k n!}{k!(n-k)!}$. Thus, if k is an constant and independent of n, then the computational time is bounded by a polynomial in n. On the other hand, if k grows with n, then the computational time can grow exponentially with n [3].

Equivalence Between Discrete and Continuous Problems

Before we show the equivalence between discrete and continuous programs, it is important to discuss an equivalence property between two extremum problems [2]. Therefore, we refer to the following theorem (see [2] for a proof).

Theorem 1 *Let $\bar{Z}$ and $\bar{X}$ be compact sets in $\mathbb{R}^n$, R be a closed set in $\mathbb{R}^n$, and let the following hypotheses hold.*

H_1) *$f\colon \mathbb{R}^n \to \mathbb{R}$ is a bounded function on $\bar{X}$, and there exists an open set $A \subset \bar{Z}$ and real number α, $L > 0$ such that, for any x, $y \in S$, f satisfies the following Hölder condition: $|f(x) - f(y)| \le L\|x - y\|^{\alpha}$.*

H_2) *It is impossible to find $\varphi : \mathbb{R}^n \to \mathbb{R}$ such that*

(i) φ is continuous on $\bar{X}$,

(ii) $\varphi(x) = 0$, $x \in \bar{Z}$; $\varphi(x) > 0$, $x \in \bar{X} - \bar{Z}$,

(iii) $\forall z \in \bar{Z}$, there exists a neighborhood $S(z)$ and a real $\bar{\varepsilon} > 0$ such that, for any $x \in S(z) \cap (\bar{X} - \bar{Z})$, $\varphi(x) \ge \bar{\varepsilon}\|x - z\|^{\alpha}$.

Then a real μ_0 exists such that for any real $\mu \ge \mu_0$, $\min f(x)$, $x \in \bar{Z} \cap R$ is equivalent to $\min[f(x) + \mu\varphi(x)]$, $x \in \bar{X} \cap R$.

Now we can show an equivalence between discrete and continuous programs from the following theorem [2].

Theorem 2 *Let $e^T = (1, 1, \ldots, 1)$, $\bar{Z} = \mathbb{B}^n$, $\bar{X} = \{x \in \mathbb{R}^n;\ 0 \le x \le e\}$, $R = \{x \in \mathbb{R}^n;\ g(x) \ge 0\}$. Consider the problem*

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & g(x) \ge 0, \quad x \in B^n , \end{aligned} \tag{2}$$

and the problem

$$\begin{aligned} \min \quad & [f(x) + \mu x^T(e - x)] \\ s.t. \quad & g(x) \ge 0, \quad 0 \le x \le e . \end{aligned} \tag{3}$$

Then we suppose that f verifies assumption H_1 from Theorem 1 with $\alpha = 1$; that is, it is bounded on $\bar{X}$ and Lipschitz continuous on an open set $A \supseteq \bar{Z}$. Subsequently, there exists some $\mu_0 \in \mathbb{R}$ such that $\forall \mu < \mu_0$ Problems (2) and (3) are equivalent.

Integer Programming Problems and Complementarity Problems

The connections between integer programs and complementarity problems can be exhibited by applying KKT conditions. The results can be generalized in the quadratic programming case [4].

Theorem 3 *Let us first assume*

3a) $f\colon \mathbb{R}^n \to \mathbb{R}$, $g\colon \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable functions.

3b) $g(x)$ satisfies a constraint qualification condition at x^0 to ensure that KKT conditions are validated.

Then the nonlinear programming problem

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & g(x) \ge 0, \quad x \ge 0 , \end{aligned} \tag{4}$$

has an optimal solution x^0 if there exist $u^0 \in \mathbb{R}^n$, y^0, $v^0 \in \mathbb{R}^{\nu}$ such that (x^0, y^0, u^0, v^0) is an optimal solution to the following problem:

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & f'(x) - y^T g'(x) - u = 0, \\ & g(x) - v = 0, \\ & y^T v = 0 \\ & x^T u = 0 \\ & x, y, u, v \ge 0 . \end{aligned} \tag{5}$$

Proof 1 *Necessity.* If x^0 is an optimal solution to Problem (4), from KKT conditions we obtain (y^0, u^0) such that

$$\begin{aligned} f'(x^0) - y^{0^{\mathrm{T}}} g(x^0) - u^0 &= 0, \\ g(x^0) &\ge 0, \\ x^{0^{\mathrm{T}}} u^0 &= 0, \\ x^0,\ y^0,\ u^0 &\ge 0 . \end{aligned}$$

Let $v^0 = g(x^0)$, then (x^0, y^0, u^0, v^0) is an optimal solution to Problem (5).

Sufficiency. The proof is trivial. □

We now generalize the results of Theorem 3 to the quadratic programming case. Consider the following problem

$$\begin{aligned} \min \quad & \frac{1}{2} x^{\mathrm{T}} Q x + c^{\mathrm{T}} x \\ \text{s.t.} \quad & Ax \ge b, \\ & x \in B^n , \end{aligned} \tag{6}$$

where Q is a symmetric matrix. Using Theorem 2, Problem (6) is equivalent to

$$\begin{aligned} \min \quad & \left[\frac{1}{2} x^{\mathrm{T}}(Q - 2\mu I)x + (c^{\mathrm{T}} + \mu e^{\mathrm{T}})x\right] \\ \text{s.t.} \quad & Ax \ge b, \\ & x \le e, \\ & x \ge 0 . \end{aligned} \tag{7}$$

Applying Theorem 3 to Problem (7), we then obtain

$$\min \quad \left[\frac{1}{2}x^{\mathrm{T}}(Q-2\mu I)x+(c^{\mathrm{T}}+\mu e^{\mathrm{T}})x\right] \tag{8}$$

$$\text{s.t.} \quad c+Qx+\mu(e-2x)-y^{\mathrm{T}}A+t=u\,, \tag{9}$$

$$b-Ax=v\,, \tag{10}$$

$$e-x=w, \tag{11}$$

$$x^{\mathrm{T}}u=0\,, \tag{12}$$

$$y^{\mathrm{T}}v=0\,, \tag{13}$$

$$t^{\mathrm{T}}w=0\,, \tag{14}$$

$$x,y,t,u,v,w\geq 0\,. \tag{15}$$

Arrange the terms in (9), we then have $Qx-2\mu x=-(c+\mu e)+y^{\mathrm{T}}A-t+u$. Consequently, (8) becomes $\min[\frac{1}{2}(c^{\mathrm{T}}+\mu e^{\mathrm{T}})x+\frac{1}{2}(b^{\mathrm{T}}y-e^{\mathrm{T}}t)$. From (12), (13), and (14), we have

$$\begin{aligned} x^{\mathrm{T}}u &= 0,\\ 0 &= y^{\mathrm{T}}v=y^{\mathrm{T}}b-y^{\mathrm{T}}Ax,\\ 0 &= t^{\mathrm{T}}w=t^{\mathrm{T}}e-t^{\mathrm{T}}x\,; \end{aligned}$$

therefore, $y^{\mathrm{T}}b=y^{\mathrm{T}}Ax$ and $t^{\mathrm{T}}e=t^{\mathrm{T}}x$. Taken all together, Problem (6) is equivalent to the following problem.

$$\begin{aligned} \min \quad & \hat{c}^{\mathrm{T}}\hat{x}\\ \text{s.t.} \quad & \hat{A}\hat{x}+\hat{u}=\hat{b},\\ & \hat{x}\hat{u}=0,\\ & \hat{x},\quad \hat{u}\geq 0\,, \end{aligned}$$

where

$$\begin{aligned} \hat{x}^{\mathrm{T}} &= (x^{\mathrm{T}},y^{\mathrm{T}},t^{\mathrm{T}}),\\ \hat{u}^{\mathrm{T}} &= (u^{\mathrm{T}},v^{\mathrm{T}},w^{\mathrm{T}}),\\ \hat{A} &= \begin{pmatrix} -Q+2\mu I & A^{\mathrm{T}} & -I\\ A & 0 & 0\\ I & 0 & 0 \end{pmatrix},\\ \hat{c}^{\mathrm{T}} &= \frac{1}{2}(c^{\mathrm{T}}+\mu e^{\mathrm{T}}+e^{\mathrm{T}},b^{\mathrm{T}},e^{\mathrm{T}}),\\ \hat{b}^{\mathrm{T}} &= (c^{\mathrm{T}},b^{\mathrm{T}},e^{\mathrm{T}})\,. \end{aligned}$$

Note that there are no restrictive assumptions made on Q, this transformation is applicable to the convex case as well as the nonconvex case.

Integer Programming Problems and Quadratic Integer Programming Problems

Integer programming is used to model a variety of important practical problems in operations research, engineering, and computer science. Consider the following linear zero-one programming problem:

$$\begin{aligned} \min \quad & c^{\mathrm{T}}x\\ \text{s.t.} \quad & Ax\leq b,\quad x_i\in\{0,1\},\quad (i=1,\ldots,n) \end{aligned}$$

where A is a real $(m\times n)$-matrix, $c\in\mathbb{R}^n$ and $b\in\mathbb{R}^m$. Let $e^{\mathrm{T}}=(1,\ldots,1)\in\mathbb{R}^n$ denote the vector whose components are all equal to 1. Then the zero-one integer linear programming problem is equivalent to the following concave minimization problem:

$$\begin{aligned} \min \quad & f(x)=c^{\mathrm{T}}x+\mu x^{\mathrm{T}}(e-x)\\ \text{s.t.} \quad & Ax\leq b,\ 0\leq x\leq e \end{aligned}$$

where μ is a sufficiently large positive integer. We know that the function $f(x)$ is concave because $-x^{\mathrm{T}}x$ is concave.

The equivalence of the two problems is based on the facts that a concave function attains its minimum at a vertex and that $x^{\mathrm{T}}(x-e)=0,\ 0\leq x\leq e$, implies $x_i=0$ or 1 for $i=1,\ldots,n$. We note that a vertex of the feasible domain is not necessarily a vertex of the unit hypercube $0\leq x\leq e$, but the global minimum is attained only when $x^{\mathrm{T}}(e-x)=0$, provided that μ is a sufficiently large number.

These transformation techniques can be applied to reduce quadratic zero-one problems to equivalent concave minimization problems. For instance, consider a quadratic zero-one problem of the following form:

$$\begin{aligned} \min \quad & f(x)=c^{\mathrm{T}}x+x^{\mathrm{T}}Qx\\ \text{s.t.} \quad & x\in\{0,1\} \end{aligned}$$

where Q is a real symmetric $(n\times n)$ matrix. Given any real number μ, let $\bar{Q}=Q+\mu I$ where I is the $(n\times n)$ unit matrix, and $\bar{c}=c-\mu e$. Because of $\bar{f}(x)=f(x)$, the above quadratic zero-one problem is equivalent to the problem:

$$\begin{aligned} \min f(x) \quad &= \bar{c}^{\mathrm{T}}x+x^{\mathrm{T}}\bar{Q}x\\ \text{s.t.} \quad & x_i\in\{0,1\},\quad (i=1,\ldots,n) \end{aligned}$$

In this case, if we choose μ such that $\bar{Q} = Q + \mu I$ becomes a negative semidefinite matrix (e. g., $\mu = -\lambda$, where λ is the largest eigenvalue of Q), then the objective function $\bar{f}(x)$ becomes concave and the constraints can be replaced by $0 \le x \le e$. Thus, this problem is equivalent to the minimization of a quadratic concave function over the unit hypercube [4].

Various Equivalent Forms of Quadratic Zero-One Programming Problems

The problem considered here is a quadratic zero-one program, which has the form

$$\begin{aligned} \min f(x) \quad &= x^{\mathrm{T}} Q x, \\ \text{s.t.} \quad &x_i \in \{0,1\}, \quad i = 1, \ldots, n, \end{aligned} \tag{16}$$

where Q is an $n \times n$ matrix [6,7]. Throughout this section the following notation will be used.

- $\{0,1\}^n$: set of n dimensional 0–1 vectors.
- $R^{n\times n}$: set of $n \times n$ dimensional real matrices.
- R^n: set of n dimensional real vectors.

In order to formalize the notion of equivalence we need some definitions.

Definition 1 The problem P is "polynomially reducible" to problem P_0 if given an instance $I(P)$ of problem P, an instance $I(P_0)$ of problem P_0 can be obtained in polynomial time such that solving $I(P)$ will solve $I(P_0)$.

Definition 2 Two problems P_1 and P_2 are called "equivalent" if P_1 is "polynomially reducible" to P_2 and P_2 is "polynomially reducible" to P_1.

Consider the following three problems:

$$\begin{aligned} P\colon \min f(x) = \quad & x^{\mathrm{T}} Q x, \quad x \in \{0,1\}^n, \\ & Q \in R^{n\times n}, \\ P_1\colon \min f(x) = \quad & x^{\mathrm{T}} Q x + c^{\mathrm{T}} x, \quad x \in \{0,1\}^n, \\ & Q \in R^{n\times n}, c \in R^n, \\ P_2\colon \min f(x) = \quad & x^{\mathrm{T}} Q x, \quad x \in \{0,1\}^n, \\ & Q \in R^{n\times n}, \\ & \sum_{i=1}^{n} x_i = k \text{ for some } k \\ \text{s.t.} \quad & 0 \le k \le n, \\ & \text{where } x = (x_1, x_2, \ldots, x_n). \end{aligned}$$

Next we show that problems P, P_1, and P_2 are all "equivalent". Then, formulation P_2 will be used in the rest of the sections.

Lemma 1 *P is "polynomially reducible" to P_1.*

Proof 2 It is very easy to see that P is a special case of P_1. □

Lemma 2 *P_1 is "polynomially reducible" to P.*

Proof 3 Problem P_1 is defined as follows: $\min f(x) = x^{\mathrm{T}} Q x + c^{\mathrm{T}} x, x \in \{0,1\}^n, Q \in R^{n\times n}, c \in R^n$. If $Q = (q_{ij})$ then let $B = (b_{ij})$ where

$$b_{ij} = \begin{cases} q_{ij} & \text{if } i \ne j \\ q_{ij} + c_i & \text{if } i = j. \end{cases}$$

Since $x_i^2 = x_i$ (because $x_i \in \{0,1\}$), we have $g(x) = x^{\mathrm{T}} B x = x^{\mathrm{T}} Q x + c^{\mathrm{T}} x$. So the following problem is equivalent to problem $P_1 : \min g(x) = x^{\mathrm{T}} B x, x \in \{0,1\}^n, B \in R^{n\times n}$. □

Using Lemma 1 and Lemma 2, it is evident that P and P_1 are "equivalent".

Lemma 3 *P_2 is "polynomially reducible" to P.*

Proof 4 Problem P_2 is as follows: $\min f(x) = x^{\mathrm{T}} Q x, \quad x \in \{0,1\}^n, \quad Q \in R^{n\times n}, \sum_{i=1}^{n} x_i = k$ for some k s.t. $0 \le k \le n$. If $Q = (q_{ij})$ then let $M = 2[\sum_{j=1}^{n} \sum_{i=1}^{n} |q_{ij}|] + 1$. Now, define the following problem P: $\min g(x) = x^{\mathrm{T}} Q x + M(\sum_{i=1}^{n} x_i - k)^2$ s.t. $x \in \{0,1\}^n, Q \in R^{n\times n}$. Let $x_b = (x_1^b, \ldots, x_n^b)$ and $x_0 = (x_1^0, \ldots, x_n^0)$ such that $\sum_{i=1}^{n} x_i^b \ne k$ and $\sum_{i=1}^{n} x_i^0 = k$, then $g(x_0) \le \frac{M-1}{2}$ as $\sum_{i=1}^{n} x_i^0 = k$, $g(x_b) \ge \frac{-(M-1)}{2} + M$ or $g(x_b) \ge \frac{M+1}{2}$ as $|\sum_{i=1}^{n} x_i^b - k| \ge 1$. Therefore, $g(x_0) < g(x_b)$ if $\sum_{i=1}^{n} x_i^b \ne k$ and $\sum_{i=1}^{n} x_i^0 = k$. Hence, if $\min g(x) = g(x_0)$ where $x_0 = (x_1^0, \ldots, x_n^0)$ then $\sum_{i=1}^{n} x_i^0 = k$. So $\min f(x) = \min g(x)$. From the above discussion, it can be easily seen that P_2 is "polynomially reducible" to P. □

The proof of Lemma 3 also illustrates how equality (knapsack) constraints in a quadratic zero-one program can be eliminated.

Lemma 4 *P is "polynomially reducible" to P_2.*

Proof 5 Let problem P be defined as follows: $\min f(x) = x^{\mathrm{T}} Q x, \quad x \in \{0,1\}^n, \quad Q \in R^{n\times n}$. Define a series of $(n+1)$ problems: $P_2(0), P_2(1), P_2(2), \cdots,$

$P_2(n)$, where $P_2(j)$ is the following problem $\min f(x) = x^{\mathrm{T}}Qx, \quad x \in \{0,1\}^n, \quad Q \in R^{n\times n}, \sum_{i=1}^{n} x_i = j$. Let the minimum of the problem $P_2(j)$ be y_j, then the minimum of problem P is easily seen to be the min $\{y_0, y_1, \ldots, y_n\}$. □

Lemma 3 and Lemma 4 imply that P and P_2 are "equivalent". Since "equivalent" is a transitive relative, P, P_1, P_2 are all "equivalent".

Complexity of Quadratic Zero-One Programming Problems

Quadratic zero-one programming is a difficult problem. We next will show that the quadratic knapsack zero-one problem in (P_2) is a NP hard problem by proving that it is equivalent to the k-clique problem. A k-clique is a complete graph with k vertices.

k-clique Problem

Given a graph G=(V, E) (V is the set of vertices and E is the set of edges), does the graph G have a k-clique as one of its subgraphs?

k-clique problem is known to be NP-complete. We will show that the k-clique problem is "polynomially reducible" to problem P_2 defined in the previous subsection.

Theorem 4 *The k-clique problem is "polynomially reducible" to P_2.*

Proof 6 Problem P_2 was defined as $\min f(x) = x^{\mathrm{T}}Qx$, s.t. $x_i \in \{0,1\}, i = 1, \cdots, n, \sum_{i=1}^{n} x_i = m$ for some $0 \le m \le n$. Given the graph $G = (V, E)$, define $Q = (q_{ij})$ such that

$$q_{ij} = \begin{cases} 0 & \text{if } (v_i, v_j) \in E \\ -1 & \text{if } (v_i, v_j) \notin E\,, \end{cases}$$

where $n = |V|, m = k$ (we are trying to find a k-clique). The meaning attached to the vector $\mathbf{x} \in \{0,1\}^n$ in problem P_2 is as follows

$$x_i = \begin{cases} 1 & \text{means that } v_i \text{ is in the clique,} \\ 0 & \text{means that } v_i \text{ is not in the clique}\,. \end{cases}$$

We can easily prove that the graph G has a k-clique if and only if $\min f(x) = -k(k-1)$. So the k-clique problem is "polynomially reducible" to P_2. □

Problem P_2 is "equivalent" to P, so problem P is also NP-hard. Therefore, as the dimension of the problem increases, the necessary CPU time to solve the problem increases exponentially.

Quadratic Zero-One Programming and Mixed Integer Programming

In this section, we consider a quadratic zero-one programming problem in the following form:

$$\begin{aligned} \min f(x) \quad & = x^{\mathrm{T}}Qx, \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = k, \quad x \in \{0,1\}^n\,. \end{aligned} \tag{17}$$

Let Q be $n \times n$ matrix, whose each element $q_{i,j} \ge 0$. Define $x = (x_1, \ldots, x_n)$, where each x_i represents binary decision variables. We will show that the problem in (17) can be linearized as the following mixed integer programming problems. The first linearization technique is trivial and can be found elsewhere. Recently, more efficient linearization technique was introduced in [1]. In addition, the linearization technique for more general case (where $q_{i,j} \in real$) and multi-quadratic programming was also proposed in [1].

Conventional Linearization Approach

For each product $x_i x_j$ in the objective function of the problem (17) we introduce a new continuous variable, $x_{ij} = x_i x_j (i \ne j)$. Note that $x_{ii} = x_i^2 = x_i$ for $x_i \in \{0,1\}$. The equivalent mixed integer programming problem (MIP) is given by:

$$\begin{aligned} \min \quad & \sum_i \sum_j q_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = k, \\ & x_{ij} \le x_i, && \text{for } i, j = 1, \ldots, n (i \ne j) \\ & x_{ij} \le x_j, && \text{for } i, j = 1, \ldots, n (i \ne j) \\ & x_i + x_j - 1 \le x_{ij}, && \text{for } i, j = 1, \ldots, n (i \ne j) \\ & 0 \le x_{ij} \le 1, && \text{for } i, j = 1, \ldots, n (i \ne j) \end{aligned} \tag{18}$$

where $x_i \in \{0,1\}$, $i, j = 1, \ldots, n$.

The main disadvantage of this approach is that the number of additional variables we need to introduce is

$O(n^2)$, and the number of new constraints is also $O(n^2)$. The number of 0–1 variables remains the same.

A New Linearization Approach

Consider the following mixed integer programming problem:

$$\begin{aligned} \min_{x,y,s} g(s) \quad &= \sum_{i=1}^{n} s_i = e^{\mathrm{T}} s \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = k, \\ & Qx - y - s = 0, \\ & y \le \mu(e - x), \\ & x_i \in \{0, 1\}, \quad \text{for } i = 1, \ldots, n \\ & y_i,\ s_i \ge 0, \quad \text{for } i = 1, \ldots, n\,. \end{aligned} \tag{19}$$

where Q is an $n \times n$ matrix, whose each element $q_{i,j} \ge 0$.

In [1], the mixed integer 0–1 programming problem in (19) was proved equivalent to the quadratic zero-one programming in (17). The main advantage of this approach is that we only need to introduce $O(n)$ additional variables and $O(n)$ new constraints, where the number of 0–1 variables remains the same. This linearization technique proved more robust and more efficiently solving quadratic zero-one and multi-quadratic zero-one programming problems [1].

References

1. Chaovalitwongse WA, Pardalos PM, Prokoyev OA (2004) Reduction of multi-quadratic 0–1 programming problems to linear mixed 0–1 programming problems. Oper Res Lett 32(6):517–522
2. Giannessi F, Niccolucci F (1976) Connections between nonlinear and integer programming problems. Istituto Nazionale di Alta Matematica, Symposia Mathematica 19:161–176
3. Hager WW, Pardalos PM, Roussos IM, Sahinoglou HD (1991) Active constraints, indefinite quadratic programming, and test problems. J Optim Theor Appl 68(3):499–511
4. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
5. Pardalos PM (1991) Global optimization algorithms for linearly constrained indefinite quadratic problems. Comput Math Appl 21:87–97
6. Pardalos PM, Rodgers G (1989) Parallel branch and bound algorithms for unconstrained quadratic zero-one programming. In: Sharda R et al (eds) Impact of recent computer advances on operations research. Elsevier, pp 131–143
7. Pardalos PM, Rodgers G (1990) Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Comput 45:131–144
8. Pardalos PM, Schnitger G (1988) Checking local optimality in constrained quadratic programming is $\mathcal{NP}$-hard. Oper Res Lett 7(1):33–35
9. Pardalos PM, Vavasis S (1991) Quadratic programming with one negative eigenvalue is $\mathcal{NP}$-hard. J Global Optim 1:15–23
10. Sahni S (1974) Computationally related problems. SIAM J Comput 3:262–279
11. Vavasis S (1991) Nonlinear optimization: complexity issues. Oxford Univeristy Press, Oxford

Quadratic Knapsack

YASUTOSHI YAJIMA
Tokyo Institute Technol., Tokyo, Japan

MSC2000: 90C20, 90C60

Article Outline

Keywords
See also
References

Keywords

Quadratic programming; Knapsack constraint

The quadratic knapsack problem is one of the simplest quadratic programming problems as defined below (cf. also ► Quadratic programming with bound constraints):

$$(\mathrm{P}) \begin{cases} \min & f(x) = \frac{1}{2} x^{\mathrm{T}} Q x + c^{\mathrm{T}} x \\ \text{s.t.} & \sum_{i=1}^{n} a_i x_i = M, \\ & 0 \le x_i \le 1, \quad i = 1, \ldots, n, \end{cases}$$

where $x \in \mathbf{R}^n$ is a variable vector, $Q \in \mathbf{R}^{n \times n}$, $c \in \mathbf{R}^n$ and M is a scalar.

The problems are mainly classified by the nature of matrix Q. When the matrix Q is *positive semidefinite*, i. e., the objective function $f(x)$ is convex, problem (P) can be solved in polynomial time by the *ellipsoid algorithm* [8], and several kinds of interior point algorithms (e. g. [5,7,11], which solve general convex quadratic problems including (P) as a special case). Also, P.M. Pardalos, Y. Ye and C.G. Han [15] show a potential reduction algorithm for the special case of (P) defined below:

$$\begin{cases} \min & \frac{1}{2}x^\top Qx \\ \text{s.t.} & \sum_{i=1}^{n} x_i = 1, \\ & x_i \geq 0, \quad i = 1, \ldots, n. \end{cases}$$

In particular, when (P) has a diagonal matrix Q with positive elements, an $O(n)$ algorithm has been proposed by P. Brucker [3]. The algorithm generates the corresponding KKT condition using binary search. Pardalos and N. Kovoor [13] also propose an $O(n)$ randomized method.

The convex case is important because of its frequent appearance as a subproblem in many application areas. Among those are general convex quadratic programming [9], multicommodity network flow problems [1], resource management [2], and portfolio selection problems [10].

The problem becomes extremely difficult if $f(x)$ is not convex. S. Sahni [16] shows that the problems with the negative diagonal matrix Q are *NP*-hard (cf. also ► Computational complexity theory; ► Complexity theory), which implies that the general indefinite case is also *NP*-hard.

Let $a_1, \ldots, a_n$ and b be positive integers, and let us consider the *subset sum problem*, which finds a feasible solution of the set defined below:

$$\left\{ x : \sum_{i=1}^{n} a_i x_i = b, \quad x_i \in \{0, 1\}, \ i = 1, \ldots, n \right\}.$$

The feasibility is determined by the the following concave quadratic knapsack problem:

$$\begin{cases} \min & \sum_{i=1}^{n} x_i(1 - x_i) \\ \text{s.t.} & \sum_{i=1}^{n} a_i x_i = M, \ \ 0 \leq x_i \leq 1, \ \ i = 1, \ldots, n. \end{cases}$$

The subset sum problem is feasible if and only if the global optimum value of the corresponding quadratic knapsack problem is zero.

As we see in the above, the indefinite case arises in several combinatorial optimization problems. For example, given a graph $G(V, E)$ where $V = \{1, \ldots, n\}$ is a set of vertices and $E \subseteq V^2$ is a set of edges, find the *maximum clique* of G. This problem can be formulated in the following way:

$$\begin{cases} \min & \sum_{(i,j)\in E} -x_i x_j \\ \text{s.t.} & \sum_{i=1}^{n} x_i = 1, \\ & x_i \geq 0, \quad i = 1, \ldots, n. \end{cases}$$

If G has a maximum clique of size k, then the global maximum is $(1/k - 1)/2$. We can also formulate the *maximum independent set problem* and the *node covering problem* in a similar fashion.

One can also formulate any quadratic minimization problem over a convex hull by the quadratic knapsack problem. Consider the problem of the form:

$$\begin{cases} \min & q(z) = z^\top Mz + r^\top z \\ \text{s.t.} & z \in P, \end{cases} \tag{1}$$

where $z, r \in \mathbf{R}^m$, $M \in \mathbf{R}^{m\times m}$ and $P \subseteq \mathbf{R}^m$ is the polytope described as the convex hull of a given set of points $\{v_1, \ldots, v_n\}$. It can be verified easily that the above general quadratic problem has the following equivalent formulation

$$\begin{cases} \min & f(x) = x^\top (V^\top MV)x + r^\top Vx \\ \text{s.t.} & \sum_{i=1}^{n} x_i = 1, \\ & x_i \geq 0, \quad i = 1, \ldots, n, \end{cases} \tag{2}$$

where $V = [v_1, \ldots, v_n]$. Let z^* and x^* be optimum solutions of (1) and (2), respectively. Then we have

$$q(z^*) = f(x^*),$$

and moreover $z^* = Vx^*$.

There exist only a few algorithms for obtaining a global optimum solution for the case of the general indefinite Q. See [15] for a partitioning approach as well as an interior point method, while [4] surveys algorithms for general nonconvex quadratic problems.

The case when the objective function is separable has also been well investigated by several authors. Some practical algorithms to obtain an exact solution are reported in [6,14]. S.A. Vavasis [18] shows an $O(n(\log n)^2)$ algorithm for finding a local minimum of the problem, while K.G. Murty and S.N. Kabadi [12] show that verifying a local minimum for an indefinite quadratic problem with general constraints is *NP*-hard. Also, [17] gives an ϵ-approximation algorithm which is weakly polynomial in the problem size if the number of negative diagonal elements is fixed.

See also

- ▶ αBB Algorithm
- ▶ Complexity Theory: Quadratic Programming
- ▶ D.C. Programming
- ▶ Integer Programming
- ▶ Multidimensional Knapsack Problems
- ▶ Quadratic Assignment Problem
- ▶ Quadratic Fractional Programming: Dinkelbach Method
- ▶ Quadratic Programming with Bound Constraints
- ▶ Quadratic Programming Over an Ellipsoid
- ▶ Reverse Convex Optimization
- ▶ Standard Quadratic Optimization Problems: Algorithms
- ▶ Standard Quadratic Optimization Problems: Applications
- ▶ Standard Quadratic Optimization Problems: Theory

References

1. Ali A, Helgason R, Kennington J, Lall H (1980) Computational comparison among three multicommodity network flow algorithms. Oper Res 28(4):995–1000
2. Bitran GR, Hax AC (1981) Disaggregation and resource allocation using convex knapsack problems with bounded variables. Managem Sci 27(4):431–441
3. Brucker P (1984) An O(n) algorithm for quadratic knapsack problems. Oper Res Lett 3(3):163–166
4. Floudas CA, Visweswaran V (1995) Quadratic optimization. In: Handbook Global Optim. Nonconvex Optim. Appl., vol 2. Kluwer, Dordrecht, pp 217–269
5. Goldfarb D, Liu S (1990/1) An O(n3L) primal interior point algorithm for convex quadratic programming. Math Program A 49(3):325–340
6. Horst R, Tuy H (1993) Global optimization: Deterministic approaches, 2nd edn. Springer, Berlin
7. Kojima M, Mizuno S, Yoshise A (1991) An O ($\sqrt{n}$ L) iteration potential reduction algorithm for linear complementarity problems. Math Program A 50(3):331–342
8. Kozlov MK, Tarasov SP, Khachiyan LG (1979) Polynomial solvability of convex quadratic programming. R-Dokl 248(5):1049–1051
9. Lin YY, Pang J-S (1987) Iterative methods for large convex quadratic programs: A survey. SIAM J Control Optim 25(2):383–411
10. Markowitz HM (1952) Portfolio selection. Finance 7:77–91
11. Monteiro RDC, Adler I, Resende MGC (1990) A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension. Math Oper Res 15(2):191–214
12. Murty KG, Kabadi SN (1987) Some NP-complete problems in quadratic and nonlinear programming. Math Program 39(2):117–129
13. Pardalos PM, Kovoor N (1990) An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. Math Program A 46(3):321–328
14. Pardalos PM, Rosen JB (1987) Constrained global optimization: Algorithms and applications. Lecture Notes Computer Sci, vol 268. Springer, Berlin
15. Pardalos PM, Ye Y, Han Chi-Geun (1991) Algorithms for the solution of quadratic knapsack problems. Linear Alg & Its Appl 152:69–91
16. Sahni S (1974) Computationally related problems. SIAM J Comput 3:262–279
17. Vavasis SA (1992) Approximation algorithms for indefinite quadratic programming. Math Program B 57(2):279–311
18. Vavasis SA (1992) Local minima for indefinite quadratic knapsack problems. Math Program A 54(2):127–153

Quadratic Programming with Bound Constraints

Pasquale L. De Angelis[1], Gerardo Toraldo[2]

[1] Naval Institute, Naples, Italy

[2] University Naples 'Federico II' and CPS, Naples, Italy

MSC2000: 90C20, 65K05

Article Outline

Keywords
Synonyms
Problem Statement
Optimality Conditions
  Local Optimality Conditions
  Global Optimality Conditions

Algorithms for Local Minimization
Algorithms for Global Minimization
See also
References

Keywords

Quadratic programming; Bounds constraints; Global optimization; Indefinite quadratic problems; Optimality conditions

Synonyms

QPwBC

Problem Statement

The *bound constrained quadratic problem* has the following form:

$$\min_{x\in\Omega} f(x) = \min_{x\in\Omega} \frac{1}{2}x^\top Qx + c^\top x\,, \qquad \Omega = \{x \in \mathbf{R}^n\colon\ l \le x \le u\}\,, \tag{1}$$

where $Q = (q_{ij}) \in \mathbf{R}^{n\times n}$ is an indefinite symmetric matrix and $x, c, l, u \in \mathbf{R}^n$. Here (as always in the sequel), all inequalities involving vectors are interpreted componentwise, and $\nabla f(x) = Qx + c$ is the gradient of f. The region Ω is assumed to be nonempty (i. e. $l_i \le u_i$ for each $i \in \{1, \ldots, n\}$) and may be unbounded (i. e. $l_i = -\infty$ and/or $u_i = +\infty$ for some $i \in \{1, \ldots, n\}$). The function $f(x)$ is assumed to be bounded below on Ω. For each $x \in \Omega$, the *active set* $A(x)$ is defined as:

$$A(x) = \{i\colon\ x_i \in \{l_i, u_i\}\}\,.$$

Problems of the form (1) naturally arise in a number of different applications. Moreover, QPwBC is a basic subroutine for many nonlinear programming codes, and the monotone *linear complementarity* problem can be written in the above form. For the convex case (i. e. Q positive semidefinite), which is known to be polynomially solvable [16], many efficient algorithms exist [4,5,7,9,10,12,18,36]. However, not many algorithms exist for the efficient solution of the general nonconvex problem [8,17,22,24,25,26].

From the complexity point of view, problem (1) is *NP*-hard [32], and even checking local optimality for a feasible point is *NP*-hard [20,27]. The complexity of finding a stationary point for (1) is an open question (in the concave case this problem is PLS-complete [15]). Algorithms to construct approximate solutions [33] in polynomial time exist.

Optimality Conditions

For problem (1) the classical local optimality conditions can be stated in a very special form. Moreover, there exist interesting results about global optimality which lead to efficient numerical procedures.

Local Optimality Conditions

Proposition 1 *If $x^* \in \Omega$ is a local minimum for problem (1) then:*

A) if $q_{ii} \ge 0$, then
- *i) $[\nabla f(x^*)]_i = 0$; or*
- *ii) $[\nabla f(x^*)]_i > 0$ and $x_i^* = l_i$; or*
- *iii) $[\nabla f(x^*)]_i < 0$ and $x_i^* = u_i$.*

B) if $q_{ii} < 0$, then
- *i) $[\nabla f(x^*)]_i > 0$ and $x_i^* = l_i$; or*
- *ii) $[\nabla f(x^*)]_i < 0$ and $x_i^* = u_i$.*

Proposition 1 specializes the classical *KKT stationarity conditions*, which only involve first order information, to problem (1) by taking into the account the sign of the second order pure derivatives. If x^* is *nondegenerate*, i. e.

$$(x_i^* - l_i)(x_i^* - u_i) + \left|\left[\nabla f(x^*)\right]_i\right| \ne 0$$

for each $i \in A(x^*)$, then the conditions A)–B) are sufficient for local minimization.

The following proposition states a relationship between the number of negative eigenvalues of the matrix Q and the cardinality of the active set at a stationarity point x^*.

Proposition 2 *If the matrix Q has k negative eigenvalues counting multiplicities, then at least k constraints are active at a local solution x^* of problem (1).*

Because of Proposition 2, if f is concave, the problem is bounded if and only if all upper and lower bounds are finite, and the solution can be found by checking all the vertices of Ω. Therefore the concave QPwBC problem is equivalent to a *quadratic zero-one problem* [1,22].

Global Optimality Conditions

Global optimality conditions for problem (1) can be stated in terms of copositivity [14] of the Hessian matrix.

Definition 3 An $n \times n$ matrix Q is *copositive* with respect to a polyhedral cone $\Gamma \subset \mathbf{R}^n$ (denoted by Γ-copositive) if and only if

$$v^\top Q v \geq 0 \text{ for all } v \in \Gamma \setminus \{0\}$$

(for *strict copositivity*, $\geq$ has to be replaced by $>$).

Definition 4 Given $\overline{x} \in \Omega$, the *tangent cone* $\Gamma(\overline{x})$ of Ω in $\overline{x}$ is defined as

$$\Gamma(\overline{x}) = \{v \in \mathbf{R}^n : \ \overline{x} + \alpha v \in \Omega \text{ for some } \alpha > 0\}.$$

Definition 5 Given $\overline{x} \in \Omega$ and $v \in \mathbf{R}^n$, we define $\lambda(\overline{x}, v)$ as follows:

$$\lambda(\overline{x}, v) = \max\{\lambda \geq 0 : \ \overline{x} + \lambda v \in \Omega\}.$$

Let us consider the following decomposition for the cone $\Gamma(x)$:

$$\Gamma(x) = \left(\bigcup_{i=1}^{n} \Gamma_i^+(x)\right) \cup \left(\bigcup_{i=1}^{n} \Gamma_i^-(x)\right),$$

where

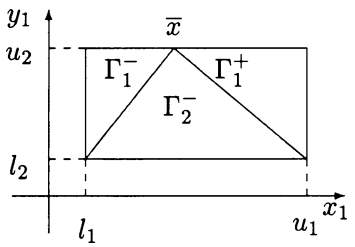
$$\Gamma_i^+(x) = \{v \in \Gamma : \ [x + \lambda(x, v) \cdot v]_i = u_i\},$$
$$\Gamma_i^-(x) = \{v \in \Gamma : \ [x + \lambda(x, v) \cdot v]_i = l_i\},$$
$$i = 1, \ldots, n,$$

i. e. if $v \in \Gamma_i^+(x) \setminus \{0\}$ (or $v \in \Gamma_i^-(x) \setminus \{0\}$), then $v_i \neq 0$ and the maximum stepsize along v moving from x saturates the ith upper (lower) constraint (see Fig. 1).

Proposition 6 *A KKT point $\overline{x}$ yields a global minimum if and only if $\overline{x}$ is stationary point and the Q_i^+ (or Q_i^-) are Γ_i^+-copositive (respectively, Γ_i^--copositive), where*

$$Q_i^+ = \left((u_i - \overline{x}_i)Q + 2\nabla f(\overline{x}) e_i^\top\right),$$
$$Q_i^- = \left((\overline{x}_i - l_i)Q - 2\nabla f(\overline{x}) e_i^\top\right).$$

Finally, the following Proposition [21] gives a sufficient condition for a KKT point to be a global minimum, in terms of convexity of some augmented function $L(x)$.



Quadratic Programming with Bound Constraints, Figure 1
Partitioning of the set $\Gamma(\overline{x})$ for the two-dimensional case

Proposition 7 *Let $\overline{x}$ be a KKT point for problem (1). Let l_i and u_i be finite for each $i \in \{1, \ldots, n\}$. Let*

$$D = \operatorname{diag}\left(\frac{|[\nabla f(\overline{x})]_1|}{u_1 - l_1}, \ldots, \frac{|[\nabla f(\overline{x})]_n|}{u_n - l_n}\right).$$

If $L(x) = f(x) + (x - \overline{x})^\top D(x - \overline{x})$ is convex in Ω, then $\overline{x}$ is a global solution of (1). Moreover, if $L(x)$ is strictly convex in Ω, then this solution is unique.

This kind of result can be a useful tool for branch and bound algorithms for global optimization. Moreover, Proposition 7 allows one to construct test problems in *quadratic programming* with known global minimum.

More results on the *global optimization* criteria for (1) exist in the literature (see, for example, [21] and references therein).

Algorithms for Local Minimization

Most of the algorithms to locally solve problem (1) can be classified in the so-called *active set strategies*, which reduce the solution of the problem to a sequence of auxiliary unconstrained subproblems on affine subspaces of $\mathbf{R}^n$ (faces). They generate a sequence of feasible points $x^{(k)}$, each $x^{(k)}$ associated with a working set $W^{(k)} \subseteq A(x^{(k)})$. The *active set algorithms* can be described according to the very general framework in Table 1.

These methods differentiate on the way they solve the subproblems $P(x^{(k)}, W^{(k)})$ and on the definition of a new face. One of the first of such algorithms, due to B.T. Polyak [29], uses a conjugate gradient algorithm to solve $P(x^{(k)}, W^{(k)})$. Since then, many modifications have been proposed for the solution of the auxiliary problem. In particular, the approximate solution of such problems is suitable to deal with large scale problems. With

Quadratic Programming with Bound Constraints, Table 1
Active set algorithm for QPwBC

Initialization:
take a first point $x^{(0)} \in \Omega$;
$W^{(0)} = A(x^{(0)})$; $k = 0$;
REPEAT
  Solve the quadratic unconstrained problem: $P(x^{(k)}, W^{(k)}) = \min f(x^{(k)} + v)$,
  $v_i = 0, \forall i \in W^{(k)}$;
  IF $P(x^{(k)}, W^{(k)})$ is unbounded below THEN
    ACT1) choose $x^{(k+1)} \in \Omega$:
    $A(x^{(k+1)}) = A(x^{(k)})$ and $f(x^{(k+1)}) < f(x^{(k)})$;
    choose $W^{(k+1)} \supset W^{(k)}$;
  ELSE
    ACT2) $\alpha^{(k)} = \max\{\alpha \in [0, 1] : x^{(k)} + \alpha v^{(k)} \in \Omega\}$;
    $x^{(k+1)} = x^{(k)} + \alpha^{(k)} v^{(k)}$;
    choose $W^{(k+1)}$
    such that $A(x^{(k+1)}) \supseteq W^{(k+1)} \neq W^{(k)}$;
  ENDIF
  $k = k + 1$;
UNTIL(stop condition holds)

regard to the definition of a new working set, in ACT2) a projected gradient step can be taken in order to add more than a new variable to the new working set [18]. Arguments of combinatorial nature show that, in non-degeneracy assumptions, an active set strategy terminates in a finite number of steps at a stationary point, provided that the exact minimization of the subproblems is performed (at least once every j steps, for some prefixed j). In case of degeneracy, the finite termination still holds for some active set algorithms. Specialized versions of active set strategies have been successfully proposed for solving large sparse problems [4,5,19].

On a completely different approach are based the algorithms that belong to the family of the interior point methods (cf. also ► Linear programming: Interior point methods); after Karmarkar's polynomial algorithm for linear programming, many interior point algorithms have been developed for the convex linear complementary problem (and therefore for the convex QPwBC). They include the primal-dual potential reduction algorithm and the path following algorithms [34]. For more detail see ► Linear complementarity problem. Finally, *penalty techniques* have been successfully proposed for the convex QPwBC [6].

Algorithms for Global Minimization

The global optimality conditions expressed in Proposition 6, suggest a very simple algorithmic framework for solving (1), whose main ingredient is the procedure COPOS(Q, Γ, d). Such a procedure [2], given an $n \times n$ matrix Q and a polyhedral cone Γ, detects either the Γ-copositivity of Q or a direction $d \in \Gamma$ such that $d^\top Qd < 0$. In the sequel all Q_i matrices and the cones Γ_i are relative to the stationary point $\overline{x}$.

In the algorithm in Table 2, COPOS is used to escape from local solution which are not global.

In [3] the basic algorithm escape has been improved using pseudoconvexity and a preprocessing procedure. However, because of complexity reasons (the problem of exactly checking copositivity is itself *NP* complete!) algorithms based on copositivity are suitable only for very small size problems.

A different approach [23], originally proposed for concave quadratic problems [30], uses a *separable formulation* based on the eigenstructure of the quadratic form. Using the linear variable transformation $x = Py$, where P is an orthogonal matrix whose columns are the eigenvectors of Q, the original problem is transformed into the separable form

$$\min_{y \in M} \phi_1(y) + \phi_2(y) ,$$

Quadratic Programming with Bound Constraints, Table 2
Global QPwBC algorithm

Initialization:
take a first stationary point $\overline{x}$;
$i = 1$;
REPEAT
  IF $\Gamma_i^+ \neq \{0\}$ THEN call COPOS(Q_i^+, Γ_i^+, d);
  IF $\Gamma_i^- \neq \{0\}$ THEN call COPOS(Q_i^-, Γ_i^-, d);
  IF a direction d is found such that $d^\top Q^+ d < 0$ or $d^\top Q^- d < 0$;
  THEN
    $x^* = \overline{x} + \lambda_{max}(d)d$;
    use x^* as starting point for a procedure that generates a new stationary point $\overline{x}$;
    $i = 0$;
  ENDIF
  $i = i + 1$;
UNTIL($i = n + 1$).

where M is a rectangle of minimum volume that contains $\widehat{\Omega} = \{y \in \mathbf{R}^n : l \leq Py \leq u\}$. The functions $\Phi_1(y)$ and $\Phi_2(y)$ are, respectively, the concave part and the convex part of the objective function.

The function $\Phi_2(y)$ can be underestimated by using a piecewise linear approximation and this gives a convex problem which approximates the original problem and for which an error bound can be given, depending both on the size of the negative eigenvalues of Q and on the size of the range of allowed displacements along the respective eigenvectors. This technique can be incorporated within a branch and bound framework. A way to improve the approximation is to make a partitioning of the domain along the eigendirections, based on the error estimate, and bounding techniques can be devised. An efficient parallel implementation is described in [28].

The *reformulation-linearization/convexification techniques* [31] are based on a suitable *linearized reformulation* of the problem (1). The goal of RLT is to try to approximate the convex envelope of the objective function over the feasible region in deriving tighter and tighter lower bounding linear programs.

Based on the combinatorial nature of the problem some *branch and bound enumerative techniques* have been proposed [13], that can be very expensive from a computational point of view, and therefore only suitable for small size problems or problems whose sparsity allows only a low number of subproblems to be explored.

More attracting from the computational point of view are algorithms based on *interior point methods*, whose main drawback is unfortunately that no guarantee exist about the convergence to the global solution of problem (1) [11].

See also

- ▶ Complexity Theory: Quadratic Programming
- ▶ D.C. Programming
- ▶ Quadratic Assignment Problem
- ▶ Quadratic Fractional Programming: Dinkelbach Method
- ▶ Quadratic Knapsack
- ▶ Quadratic Programming Over an Ellipsoid
- ▶ Reverse Convex Optimization
- ▶ Standard Quadratic Optimization Problems: Algorithms
- ▶ Standard Quadratic Optimization Problems: Applications
- ▶ Standard Quadratic Optimization Problems: Theory

References

1. Benson HH (1995) Concave minimization: Theory, applications and algorithms. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 43–142
2. Bomze IM, Danninger G (1993) A global optimization algorithm for concave quadratic problems. SIAM J Optim 3:826–842
3. Bomze IM, Danninger G (1994) A finite algorithm for solving general quadratic problems. J Global Optim 4:1–16
4. Coleman TF, Hulbert LA (1989) A direct active set algorithm for large sparse quadratic programs with simple bounds. Math Program 45:373–406
5. Coleman TF, Hulbert LA (1993) A globally and superlinearly convergent algorithm for convex quadratic programs with simple bounds. SIAM J Optim 3:298–321
6. Facchinei F, Lucidi S (1992) A class of penalty functions for optimization problems with bound constraints. Optim 26:239, 259
7. Fletcher R, Jackson MP (1974) Minimization of a quadratic function of many variables subject only to lower and upper bounds. J Inst Math Appl 14:159–174
8. Floudas CA, Visweswaran V (1995) Quadratic optimization. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 217–270
9. Frank M, Wolfe P (1956) An algorithm for quadratic programming. Naval Res Logist Quart 3:95–110
10. Gill PE, Murray W, Saunders MA, Wright MH (1991) Inertia-controlling methods for general quadratic programming. SIAM Rev 33(1):1–36
11. Han CG, Pardalos PM, Ye Y (1992) On the solution of indefinite quadratic problems using an interior point algorithm. Informatica 3(4):474–496
12. Han CG, Pardalos PM, Ye Y (1990) Computational aspects of an interior point algorithm for quadratic programming problems with box constraints. SIAM, pp 92–112
13. Hansen P, Jaumard B, Ruiz M, Xiong J (1993) Global minimization of indefinite quadratic functions subject to box constraints. Naval Res Logist 40:373–392
14. Hiriart-Urruty JB (1995) Conditions for global optimality. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 1–26
15. Johnson DS, Papadimitriou CH, Yannakakis M (1988) How easy is local search? J Comput Syst Sci 37:79–100
16. Kozlov MK, Tarasov SP, Khachian LG (1979) Polynomial solvability of convex quadratic programming. S-Dokl 20:1108–1111
17. Manas M (1968) An algorithm for a nonconvex programming problem. Econ Math Obzor Acad Nacl Ceskoslov 4:202–212

18. Moré JJ, Toraldo G (1989) Algorithms for bound constrained quadratic programming problems. Numer Math 55:377–400
19. Moré JJ, Toraldo G (1991) On the solution of large quadratic programming problems with bound constraints. SIAM J Optim 1:93–113
20. Murty KG, Kabadi SN (1987) Some NP-complete problems in quadratic and nonlinear programming. Math Program 39:117–129
21. Neumaier A (1996) Second-order sufficient optimality conditions for local and global nonlinear programming. J Global Optim 9:141–151
22. Pardalos PM (1990) Polynomial time algorithms for some classes of constrained non-convex quadratic problems. Optim 21:843–853
23. Pardalos PM (1991) Global optimisation algorithms for linearly constrained indefinite quadratic problems. Comput Math Appl 21:87–97
24. Pardalos PM, Glick JH, Rosen JB (1987) Global minimization of indefinite quadratic Problems. Computing 39:281–291
25. Pardalos PM, Rosen JB (1986) Methods for global concave minimization: A bibliographic survey. SIAM Rev 28:367–379
26. Pardalos PM, Rosen JB (1987) Constrained global optimization: Algorithms and applications. Lecture Notes Computer Sci, vol 268. Springer, Berlin
27. Pardalos PM, Schnitger G (1988) Checking local optimality in constrained quadratic programming is NP-hard. Oper Res Lett 7(1):33–35
28. Phillips AT, Rosen JB (1990) A parallel algorithm for partially separable non-convex global minimization: Linear constraints. Ann Oper Res 25:101–118
29. Poljak BT (1969) The conjugate gradient method in extremal problems. USSR Comput Math Math Phys 9:94–112
30. Rosen JB (1983) Global minimization of a linearly constrained concave function by partition of feasible domain. Math Oper Res 8:215–230
31. Sherali HD, Tuncbilek CH (1995) A reformulation-convexification approach for solving nonconvex quadratic programming problems. J Global Optim 7:1–31
32. Vavasis SA (1991) Nonlinear optimization: Complexity issues. Oxford Univ. Press, Oxford
33. Vavasis SA (1993) Polynomial time weak approximation algorithms for quadratic programming. In: Pardalos PM (ed) Complexity in Numerical Optimization. World Sci., Singapore
34. Wright S (1993) A path following infeasible-interior point algorithm for linear complementarity problems. Optim Methods Softw 2:79–106
35. Yang EK, Tolle JW (1991) A class of methods for solving large convex programs subject to box constraints. Math Program 51:223–228
36. Ye Y, Tse E (1989) An extension of Karmarkar's projective algorithm for convex quadratic programming. Math Program 44:157–179

Quadratic Programming over an Ellipsoid

Yinyu Ye
Department Management Sci., University Iowa, Iowa City, USA

MSC2000: 90C20, 90C25

Article Outline

Keywords
See also
References

Keywords

Quadratic programming; Bisection method; Newton method; Trust region model; Complexity

Quadratic programming (QP) plays an important role in optimization theory. In one sense it is a continuous optimization and a fundamental subroutine for general nonlinear programming, but it is also considered one of the most challenging combinatorial optimization problems.

One of QP problems is to minimize a *quadratic function* over an *ellipsoid constraint*. Since any ellipsoid can be transformed to a *ball* by an affine transformation, without of loss generality, we consider the following ball-constrained QP problem BQP (r):

$$\begin{cases} \min & \frac{1}{2}x^\top Qx + c^\top x \\ \text{s.t.} & x \in \mathcal{B}(r) = \{x \in \mathbf{R}^n : \ \|x\| \le r\}, \end{cases} \tag{1}$$

where $Q \in \mathbf{R}^{n\times n}$, $c \in \mathbf{R}^n$, and superscript $^\top$ denotes the transpose operation. Here, $\|\cdot\|$ denotes L_2 norm and $r > 0$ is the radius of the ball. A main recent result is that this problem is an 'easy' problem, even when the objective function is nonconvex.

We begin with a brief history of this problem. There is a class of nonlinear programming algorithms called model *trust region methods*. In these algorithms, a quadratic function is used as an approximate model of the true objective function around the current iterate. Then the main step is to minimize the model function. In general, however, the model is expected to be accurate or trusted only in a neighborhood of the current

iterate. Accordingly, the quadratic model is minimized in a L_2-norm neighborhood, which is a ball, around the current iterate. Recently (1996), it was demonstrated [5] that a class of combinatorial optimization problems can be solved by solving a sequence of ball-constrained QP problem.

The model-trust region method is due to K. Levenberg [7] and D.W. Marquardt [8]. These authors considered only the case where Q is positive definite. J.J. Moré [10] proposed an algorithm with a convergence proof for this case. D.M. Gay [4] and D.C. Sorenson [15] proposed algorithms for the general case, see also [2]. These algorithms work very well in practice, but no theoretical complexity result was established for this problem then.

It is well known [4,15] that the solution x of problem BQP (r) satisfies the following necessary and sufficient conditions:

$$\begin{aligned}(Q+\mu I)x &= -c\,,\\ \mu &\geq \max\{0,-\underline{\lambda}\}\,,\\ \|x\| &= r\,,\end{aligned} \tag{2}$$

where $\underline{\lambda}$ denotes the least eigenvalue of matrix Q. Since Q is not positive semidefinite, we must have $\underline{\lambda} < 0$.

Let μ^* and x^* satisfy conditions (2). It has been shown that μ^* is unique and

$$\mu^* \leq |\underline{\lambda}| + \frac{\|c\|}{r}\,. \tag{3}$$

It is also known that

$$|\underline{\lambda}| \leq n \max\{|q_{ij}|\}\,,$$

where q_{ij} is the (i,j)th component of matrix Q. Thus, we have

$$0 \leq \mu^* \leq \mu^0 := n \max\{|q_{ij}|\} + \frac{\|c\|}{r}\,, \tag{4}$$

where μ^0 is a computable upper bound. It is further proved that ([19])

$$\frac{1}{2}r^2\,|\underline{\lambda}| \leq \frac{1}{2}r^2\mu^* \leq q(0)-q(x^*) \leq \frac{1}{2}r^2\,|\underline{\lambda}|+r\,\|c\|\,. \tag{5}$$

This inequality can be used to develop an *approximation algorithm* for general quadratic optimization, see [3].

We now analyze the complexity of solving BQP (r). A simple *bisection method* was proposed in [18] and in [19]. For any given μ, denote solutions of the top linear equations by x_μ in conditions (2), i. e.,

$$x_\mu := -(Q+\mu I)^{-1}c\,, \quad \forall \mu > |\underline{\lambda}|\,. \tag{6}$$

For any given μ we can check to see if $\mu \geq |\lambda|$ by checking the positive definiteness of matrix $Q+\mu\, I$, which can be solved as a $LDL^\intercal$ decomposition. These facts lead to a bisection method to search for the root of $\| x_\mu \| = r$ over the interval $\mu \in [|\underline{\lambda}|, \mu^0] \subset [0, \mu^0]$. Obviously, for a given $\epsilon'' \in (0, 1)$, a μ such that, say $0 \leq \mu - \mu^* \leq \epsilon'\, \mu^*/8$, can be obtained in $O(\log(\mu^0/\mu^*) + \log(1/\epsilon'))$ bisection steps, and the cost of each step is $O(n^3)$ arithmetic operations (for performing $LDL^\intercal$ decomposition).

The remaining question is what ϵ' would be sufficient to generate an ϵ-minimizer of $q(x)$ over the ball $\mathcal{B}(r)$, that is, an x satisfying

$$\frac{q(x)-q(x^*)}{q(0)-q(x^*)} \leq \epsilon\,.$$

Let μ denote the right endpoint of the interval generated by the bisection search. Then, $\mu \geq \mu^*$. If $\mu = \mu^*$, then we get an exact solution $x^* = x_{\mu^*}$. Thus, we assume $\mu > \mu^* \geq \underline{\lambda}$. By the positive semidefiniteness of $Q+\mu^*\, I$, we have

$$\|x_\mu\| < \|x^*\| = r\,.$$

We consider two cases.

Case I. In the first case we assume

$$\left(1-\frac{\epsilon}{8\sqrt{n}}\right)\mu^* \geq |\underline{\lambda}|$$

or

$$\mu^* \geq |\underline{\lambda}| + \frac{\epsilon}{8\sqrt{n}}\mu^*\,.$$

Using the relation (6) and simplifying, we obtain

$$\begin{aligned}\|x^*\|^2 - \|x_\mu\|^2 &= (x^*)^\top\\ &\quad\times (I-(Q+\mu^* I)(Q+\mu I)^{-2}(Q+\mu^* I))x^*\\ &= (x^*)^\top(2(\mu-\mu^*)(Q+\mu I)^{-1}\\ &\quad -(\mu-\mu^*)^2(Q+\mu I)^{-2})x^*\,.\end{aligned}$$

Next we bound the above expression by using the smallest *eigenvalue* $\underline{\lambda}$ of Q. This gives

$$\begin{aligned}
&\|x^*\|^2 - \|x_\mu\|^2 \le \left(\frac{2(\mu-\mu^*)}{(\mu-|\underline{\lambda}|)} - \frac{(\mu-\mu^*)^2}{((\mu-|\underline{\lambda}|))^2}\right)\|x^*\|^2 \\
&= \left(\frac{2(\mu-\mu^*)}{(\mu-\mu^*)+(\mu^*-|\underline{\lambda}|)}\right. \\
&\quad \left. - \frac{(\mu-\mu^*)^2}{((\mu-\mu^*)+(\mu^*-|\underline{\lambda}|))^2}\right)\|x^*\|^2 \\
&= \frac{(\mu-\mu^*)^2 + 2(\mu-\mu^*)(\mu^*-|\underline{\lambda}|)}{((\mu-\mu^*)+(\mu^*-|\underline{\lambda}|))^2} r^2 \\
&= \left(1 - \frac{(\mu^*-|\underline{\lambda}|)^2}{((\mu-\mu^*)+(\mu^*-|\underline{\lambda}|))^2}\right) r^2 \\
&\le \left(1 - \frac{(\frac{\epsilon\mu^*}{8\sqrt{n}})^2}{((\mu-\mu^*)+\frac{\epsilon\mu^*}{8\sqrt{n}})^2}\right) r^2 ,
\end{aligned}$$

where in the last step we used the assumption

$$\mu^* \ge |\underline{\lambda}| + \frac{\epsilon}{8\sqrt{n}}\mu^* .$$

Therefore, if we have $\mu - \mu^* \le \epsilon'\,\mu^*/8$, then

$$\|x^*\|^2 - \|x_\mu\|^2 \le \frac{\left(\frac{2\sqrt{n}\epsilon'}{\epsilon}\right) + \left(\frac{\sqrt{n}\epsilon'}{\epsilon}\right)^2}{\left(1 + \left(\frac{\sqrt{n}\epsilon'}{\epsilon}\right)\right)^2} r^2 \le \frac{2\sqrt{n}\epsilon'}{\epsilon} r^2 . \tag{7}$$

On the other hand, note that

$$\begin{aligned}
&q(x_\mu) - q(x^*) \\
&\quad = \frac{1}{2}x_\mu^\top Q x_\mu + c^\top x_\mu - \frac{1}{2}(x^*)^\top Q x^* - c^\top x^* \\
&\quad = \frac{1}{2}(Qx_\mu + c)^\top (x_\mu - x^*) \\
&\qquad + \frac{1}{2}(Qx^* + c)^\top (x_\mu - x^*) \\
&\quad = -\frac{1}{2}\mu x_\mu^\top (x_\mu - x^*) - \frac{1}{2}\mu^*(x^*)^\top (x_\mu - x^*) \\
&\quad = -\frac{1}{2}(\mu-\mu^*)x_\mu^\top (x_\mu - x^*) \\
&\qquad - \frac{1}{2}\mu^*(\|x_\mu\|^2 - \|x^*\|^2) .
\end{aligned} \tag{8}$$

Now we use the bound (7), the assumption $\mu - \mu^* \le \epsilon'\mu^*/8$ and the fact $\| x_\mu \| \le \| x^* \| = r$ to obtain:

$$\begin{aligned}
q(x_\mu) - q(x^*) &\le \frac{\mu^* r^2 \epsilon'}{8} + r^2\mu^*\frac{\sqrt{n}\epsilon'}{\epsilon} \\
&= \left(\frac{\epsilon'}{4} + \frac{2\sqrt{n}\epsilon'}{\epsilon}\right)\frac{\mu^* r^2}{2} \\
&\le \left(\frac{\epsilon'}{4} + \frac{2\sqrt{n}\epsilon'}{\epsilon}\right)(q(0) - q(x^*)) ,
\end{aligned}$$

where the last step is due to (5). Thus, if we select

$$\epsilon' \le \frac{\epsilon^2}{2\sqrt{n} + \frac{1}{4}} ,$$

then x_μ is feasible for BQP(r) and

$$q(x_\mu) - q(x^*) \le \epsilon(q(0) - q(x^*)) ,$$

i. e., x_μ is an ϵ-*minimizer* to x^*.

Case II. In this case, we have

$$\left(1 - \frac{\epsilon}{8\sqrt{n}}\right)\mu^* < |\underline{\lambda}|$$

or

$$\mu^* < |\underline{\lambda}| + \mu^*\frac{\epsilon}{8\sqrt{n}} .$$

Again, if we have $\mu - \mu^* < \epsilon'\,\mu^*/8$, then $\mu - |\underline{\lambda}| < \frac{\epsilon'\mu^*}{8} + \frac{\mu^*\epsilon}{8\sqrt{n}}$. However, unlike Case I, we find that $\| x_\mu \|$ is not sufficiently close to r. When we observe this fact, we do the following computation, essentially due to S.A. Vavasis and R. Zippel [18], to enhance x_μ.

Let $\underline{q}$, $\left\|\underline{q}\right\| = 1$, be an *eigenvector* associated with the eigenvalue $\underline{\lambda}$. Then, one of the unit vectors e_j, $j = 1, \ldots, n-m$, must have $\left|e_j^\top \underline{q}\right| \ge \frac{1}{\sqrt{n}}$. (In fact, we can use any unit vector q to replace e_j as long as $q^\top \underline{q} \ge \frac{1}{\sqrt{n}}$. A randomly generated q will do it with high probability.) Now we solve for y from

$$(Q + \mu I)y = e_j$$

and let

$$x = x_\mu + \alpha y ,$$

where α is chosen such that $\|x\| = r$. Note we have

$$(Q + \mu I)x = -c + \alpha e_j ,$$

and in the computation of x_μ and y, matrix $Q + \mu I$ needs to be factorized only once.

It is easy to show that

$$\|y\| \geq \frac{1}{\sqrt{n}(\mu - |\underline{\lambda}|)}$$

and

$$|\alpha| \leq 2r(\mu - |\underline{\lambda}|)\sqrt{n} \leq 2r\left(\frac{\epsilon'\mu^*}{8} + \frac{\epsilon\mu^*}{8\sqrt{n}}\right)\sqrt{n}\,.$$

Then, we have from (8)

$$\begin{aligned}
&q(x) - q(x^*)\\
&= \frac{1}{2}(Qx + c)^\top (x - x^*)\\
&\quad + \frac{1}{2}(Qx^* + c)^\top (x - x^*)\\
&= \frac{1}{2}(Qx + c - \alpha e_j)^\top (x - x^*)\\
&\quad + \frac{1}{2}\alpha e_j^\top (x - x^*) - \frac{1}{2}\mu^*(x^*)^\top (x - x^*)\\
&= -\frac{1}{2}\mu x^\top (x - x^*)\\
&\quad + \frac{1}{2}\alpha e_j^\top (x - x^*) - \frac{1}{2}\mu^*(x^*)^\top (x - x^*)\\
&= -\frac{1}{2}(\mu x + \mu^* x^*)^\top (x - x^*)\\
&\quad + \frac{1}{2}\alpha e_j^\top (x - x^*)\\
&= -\frac{1}{2}(\mu - \mu^*)x^\top (x - x^*) + \frac{1}{2}\alpha e_j^\top (x - x^*)\,,
\end{aligned}$$

where the last step follows from $\|x\| = \| x^* \| = r$. Now we use $\mu - \mu^* < \epsilon'\mu^*/8$ and the preceding upper bound on α to estimate the right-hand side:

$$\begin{aligned}
q(x) - q(x^*) &\leq \frac{r^2\mu^*\epsilon'}{8} + 2\left(\frac{\epsilon'\mu^*}{8} + \frac{\epsilon\mu^*}{8\sqrt{n}}\right) r^2\sqrt{n}\\
&= \left(\frac{\epsilon'}{4} + \frac{\sqrt{n}\epsilon'}{2} + \frac{\epsilon}{2}\right)\frac{\mu^* r^2}{2}\\
&\leq \left(\frac{\epsilon'}{4} + \frac{\sqrt{n}\epsilon'}{2} + \frac{\epsilon}{2}\right)(q(0) - q(x^*))\,,
\end{aligned}$$

where the last step is due to (5). Thus, if we choose

$$\epsilon' \leq \frac{\epsilon}{\sqrt{n} + \frac{1}{2}}\,,$$

then x is feasible for BQP(r) and

$$q(x) - q(x^*) \leq \epsilon(q(0) - q(x^*)) \leq \epsilon(\overline{z} - \underline{z})\,,$$

i. e., x is an *ϵ-minimizer* of $q(x)$ over $\mathcal{B}(r)$.

Hence, the bisection method will terminate with an ϵ-minimizer of BQP(r) in at most

$$O\left(\log\left(\frac{\mu^0}{\mu^*}\right) + \log\left(\frac{1}{\epsilon}\right) + \log n\right)$$

steps, or in a total of $O(n^3(\log(\mu^0/\mu^*) + \log(1/\epsilon) + \log n))$ arithmetic operations.

Theorem *The total running time of the bisection algorithm for generating an ϵ-minimal solution to the ball-constrained QP is bounded by $O(n^3(\log(\mu^0/\mu^*) + \log(1/\epsilon) + \log n))$ arithmetic operations.*

Recently, F. Rendl and H. Wolkowicz [14] showed that BQP(r) can be reformulated as a positive semidefinite problem, which is a convex nonlinear problem. There are polynomial *interior point algorithms* (see [11]) to compute an d_x' such that $q'(d_x') - q(d_x'(\mu^k)) \leq \epsilon'$ in $O(n^3 \log(M^k/\epsilon'))$ arithmetic operations. This will also establish an

$$O\left(\left(\frac{n^6}{\epsilon}\log\frac{1}{\epsilon} + n^4\log n\right)\left(\log\frac{1}{\epsilon} + \log n\right)\right)$$

arithmetic operation bound for the algorithm.

The polynomial complexity in Theorem 1 can be further improved. In particular, see [20] for a mixed bisection and *Newton method* for solving BQP(r) and for an arithmetic operation bound $O(n^3 \log(\log(\mu^0/\mu^*) + \log(1/\epsilon')))$ to yield a μ such that $0 \leq \mu - \mu^* \leq \epsilon'$. The brief idea of the method is to first find an approximate $\underline{\mu}$ to the absolute value of the least eigenvalue $|\underline{\lambda}|$ and an approximate eigenvector q to the true $\underline{q}$, such that $0 \leq \underline{\mu} - \underline{\lambda} \leq \epsilon'$ and $q^\top q^k \geq 1 - \epsilon'$. This approximation can be done in $O(n^3 \log(\log(1/\epsilon')))$ arithmetic operations. Then, we will use q to replace e_j in Case II (i. e., $\| x_{\underline{\mu}} < r$) to enhance $x(\underline{\mu})$ and generate a desired approximation. Otherwise, we know $\mu^* > \underline{\mu}$ and, using the mixed method in [20], we will generate a $\mu \in (\underline{\mu}, \mu^0)$ such that $|\mu - \mu^*| \leq \epsilon'\,\mu^*/8$ in $O(n^3 \log(\log(\mu^0/\mu^*) + \log(1/\epsilon')))$ arithmetic operations.

Finally, let Q and c have integer data. Consider the decision problem: Is there an $x \in \mathbf{R}^n$ satisfying $\|x\| \leq 1$, and $q(x) < 0$? Under the *Turing machine* computational model, this problem can be answered in polynomial time (see [18]).

See also

- Complexity Theory: Quadratic Programming
- Quadratic Assignment Problem
- Quadratic Fractional Programming: Dinkelbach Method
- Quadratic Knapsack
- Quadratic Programming with Bound Constraints
- Standard Quadratic Optimization Problems: Algorithms
- Standard Quadratic Optimization Problems: Applications
- Standard Quadratic Optimization Problems: Theory
- Volume Computation for Polytopes: Strategies and Performances

References

1. Celis MR, Dennis JE, Tapia RA (1984) A trust region strategy for nonlinear equality constrained optimization. In: Boggs PT Byrd R, Schnable R (ed) Numerical Optimization. SIAM, Philadelphia, pp 71–82
2. Dennis JE Jr, Schnable RE (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs
3. Fu M, Luo Z-Q, Ye Y (1996) Approximation algorithms for quadratic programming. J Combin Optim (to appear)
4. Gay DM (1981) Computing optimal locally constrained steps. SIAM J Sci Statist Comput 2:186–197
5. Gibbons LE, Hearn DW, Pardalos PM (1996) A continuous based heuristic for the maximum clique problem. In: DIMACS. Amer. Math. Soc., Providence, pp 103–124
6. Kamath AP, Karmarkar NK, Ramakrishnan KG, Resende MGC (1992) A continuous approach to inductive inference. Math Program 57:215–238
7. Levenberg K (1963) A method for the solution of certain non-linear problems in least squares. Quart Appl Math 2:164–168
8. Marquardt DW (1963) An algorithm for least-squares estimation of nonlinear parameters. J SIAM 11:431–441
9. Martinez JM (1994) Local minimizers of quadratic functions on Euclidean balls and spheres. SIAM J Optim 4:159–176
10. Moré JJ (1977) The Levenberg-Marquardt algorithm: Implementation and theory. In: Watson GA (ed) Numerical Analysis. Springer, Berlin
11. Nesterov YuE, Nemirovskii AS (1993) Interior point polynomial methods in convex programming: Theory and algorithms. SIAM, Philadelphia
12. Pardalos PM, Rosen JB (1987) Constrained global optimization: Algorithms and applications. Lecture Notes Computer Sci, vol 268. Springer, Berlin
13. Powell MJD, Yuan Y (1991) A trust region algorithm for equality constrained optimization. Math Program 49:189–211
14. Rendl F, Wolkowicz H (1994) A semidefinite framework for trust region subproblems with applications to large scale minimization. CORR Report 94-32, Dept Combinatorics and Optim, Univ Waterloo, Ontario
15. Sorenson DC (1982) Newton's method with a model trust region modification. SIAM J Numer Anal 19:409–426
16. Vavasis SA (1991) Nonlinear optimization: Complexity issues. Oxford Sci. Publ., Oxford
17. Vavasis SA (1993) Polynomial time weak approximation algorithms for quadratic programming. In: Pardalos PM (ed) Complexity in Numerical Optimization. World Sci., Singapore
18. Vavasis SA, Zippel R (1990) Proving polynomial time for sphere-constrained quadratic programming. Techn Report 90–1182, Dept Computer Sci, Cornell Univ, Ithaca, NY
19. Ye Y (1992) On affine scaling algorithms for nonconvex quadratic programming. Math Program 56:285–300
20. Ye Y (1994) Combining binary search and Newton's method to compute real roots for a class of real functions. J Complexity 10:271–280
21. Yuan Y (1990) On a subproblem of trust region algorithms for constrained optimization. Math Program 47:53–63

Quadratic Semi-assignment Problem

QSAP

LEONIDAS PITSOULIS
Princeton University, Princeton, USA

MSC2000: 90C27, 90C11, 90C08

Article Outline

Keywords
See also
References

Keywords

Optimization

Consider that we have n 'objects' and m 'locations', $n > m$, and we want to assign all objects to locations with at least one object to each location, so as to minimize the overall distance covered by the flow of materials moving between different objects. Given a flow matrix $F =$

(f_{ij}) and a distance matrix $D = (d_{ij})$, we can formulate the *quadratic semi-assignment problem* as follows:

$$\begin{cases} \min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m-1}\sum_{l=k+1}^{m} f_{ij}d_{kl}x_{ki}x_{lj} \\ & \quad + \displaystyle\sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij}x_{ij} \\ \text{s.t.} & \displaystyle\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,\ldots,m, \\ & x_{ij} \in \{0,1\}, \\ & \qquad i = 1,\ldots,m, \quad j = 1,\ldots,n. \end{cases}$$

Comparing the above formulation with that of the quadratic assignment problem (cf. ► Quadratic assignment problem), we can see that the QSAP is a relaxed version of the QAP, where instead of assignment constraints we have semi-assignment constraints. SQAP unifies some interesting combinatorial optimization problems like clustering and m-coloring. In a *clustering problem* we are given n objects and a dissimilarity matrix $F = (f_{ij})$. The goal is to find a partition of these objects into m classes so as to minimize the sum of dissimilarities of objects belonging to the same class. Obviously this problem is a QSAP with coefficient matrices F and D, where D is an $m \times m$ identity matrix. In the *m-coloring problem* we are given a graph with n vertices and want to check whether its vertices can be colored by m different colors such that each two vertices which are joined by an edge receive different colors. This problem can be modeled as a SQAP with F equal to the adjacency matrix of the given graph and D the $m \times m$ identity matrix. The m-coloring has an answer 'yes' if and only if the above SQAP has optimal value equal to 0. Practical applications of the SQAP include distributed computing [5] and scheduling [1].

SQAP was originally introduced by D.E. Greenberg [2]. As pointed out in [3], this problem is *NP*-hard. I.Z. Milis and V.F. Magirou [5] propose a Lagrangian relaxation algorithm for this problem, and show that similarly as for for the QAP, it is very hard to provide optimal solutions even for SQAPs of small size. Lower bounds for the SQAP have been provided in [4], and polynomially solvable special cases have been discussed in [3].

See also

- ► Feedback Set Problems
- ► Generalized Assignment Problem
- ► Graph Coloring
- ► Graph Planarization
- ► Greedy Randomized Adaptive Search Procedures
- ► Quadratic Assignment Problem

References

1. Chretienne P (1989) A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints. Europ J Oper Res 43:225–230
2. Greenberg DE (1969) A quadratic assignment problem without column constraints. Naval Res Logist Quart 16:417–422
3. Malucelli F (1993) Quadratic assignment problems: solution methods and applications. PhD Thesis, Dip. Informatica, Univ. Pisa
4. Malucelli F, Pretolani D (1993) Lower bounds for the quadratic semi-assignment problem. Techn Report 955, Centre Rech Transports, Univ Montréal, Canada
5. Milis IZ, Magirou VF (1995) A Lagrangean relaxation algorithm for sparse quadratic assignment problems. Oper Res Lett 17:69–76
6. Stone HS (1977) Multiprocessor scheduling with the aid of network flow algorithms. IEEE Trans Softw Eng 4:85–93

Quasidifferentiable Optimization

GEORGIOS E. STAVROULAKIS
Carolo Wilhelmina Techn. University, Braunschweig, Germany

MSC2000: 49J52, 26B25, 90C99, 26E25

Article Outline

Keywords
One-Dimensional Nonsmooth Functions
  One-Sided Differentials
  Quasidifferential
  Necessary and Sufficient Optimality Conditions
Finite-Dimensional Nonsmooth Functions
  Subdifferentiable Functions
  Superdifferentiable Functions
  Quasidifferentiable Functions
Further Related Topics

See also
References

Keywords

Nonsmooth optimization; Quasidifferentiability; Nonsmooth analysis; Nonconvex optimization; Mathematical programming

Smoothness, or the existence of the classical derivative information for a function plays a significant role in the theory and the tools used today for modeling, approximation, optimization and for their applications. Nevertheless, nature seems to be more rich than the assumptions done within current mathematical or physical theories. Nonsmoothness arises in a very large number of applications. The arising phenomena, including complex dynamics, pattern formation and chaos, are appealing for both theoretical investigations and practical applications. Most of them have not yet been studied. Abandoning smoothness assumptions one arrives at the area of *nonsmooth analysis*.

Within nonsmooth approximation the classical notion of the derivative is replaced by some set-valued generalized derivative. This is required for the construction of qualitative and quantitative first order approximations of a function with points of nondifferentiability (kinks) or, respectively of a set with corners. In fact, the linearization (i. e., the linear or affine approximation) of a function at a given point which is based on the familiar Taylor expansion formula is based on the assumption that the derivative of the function (or its gradient) exists at the considered point.

Historically, for convex nondifferentiable functions, a suitable set-valued extension of the derivative has been provided by the *subdifferential* of *convex analysis*, in the sense of J.-J. Moreau and R.T. Rockafellar [8,12]. For the general case of nonconvex, nondifferentiable functions, a direct extension of the convex analysis subdifferential has been provided by the *generalized subdifferential* in the sense of F.H. Clarke and Rockafellar [1,2,13]. This notion has been used in a variety of applications, although it does not possess the above mentioned first order approximation property. One should note that a large number of notions have been proposed for the approximation of nonconvex and nonsmooth functions (or sets) or of the solution of affiliated optimization problems. A complete list would go beyond the limits of this short article. This activity demonstrates the large practical interest of this area.

The *quasidifferential* in the sense of V.F. Demyanov and A.M. Rubinov is an appropriate tool for the construction of first order approximations of functions and sets and, subsequently, for the solution of nonsmooth and nonconvex optimization problems. By treating separately convex and concave contributions of the function the quasidifferential introduces an ordered pair of convex sets. Intuitively speaking, the convex analysis subdifferential is present, for the convex contribution, while the superdifferential takes into account the concave parts (which, in turn, can also be studied by means of convex analysis arguments, since a concave function becomes convex if one changes its sign). The links of the quasidifferential with other notions of nonsmooth analysis have been discussed in ► Quasidifferentiable optimization: Dini derivatives, Clarke derivatives and [4]). More important is that certain calculus rules have been developed for the calculation of the quasidifferential of sums, differences, products, quotients and, more general, of every function that can be constructed by using finite number times the minimum and maximum operators over a finite number of classical, smooth constituent functions (see ► Quasidifferentiable optimization: Calculus of quasidifferentials and [7]). Finally, based on the notion of the quasidifferential, certain new variational formulations can be constructed which generalize the notion of *variational inequalities* of convex analysis. These variational formulations have the form of sets of variational inequalities, are valid for the general nonsmooth and nonconvex case (see also ► Quasidifferentiable optimization: Variational formulations [6,11]) and give a computationally advantageous form to the *hemivariational inequalities* in the sense of P.D. Panagiotopoulos (see, among others, ► Nonconvex energy functions: Hemivariational inequalities; ► Hemivariational inequalities: Applications in mechanics and [6,9,10,11]).

Here, the definition of the quasidifferential for one-dimensional and finite-dimensional functions is given and hints for its extension into functionals are discussed. Finally, some information on the related, and more convenient for the numerical applications notion of the codifferential and on the construction of optimization algorithms is provided.

One-Dimensional Nonsmooth Functions

Let f be a real-valued finite function defined on the real line $\mathbf{R}$. The most powerful and widely used tool to study the properties of f is the notion of derivative. Function f is called *differentiable* at $x \in \mathbf{R}$ if there exists its derivative $f'(x)$ at x, which is defined by

$$f'(x) = \lim_{\alpha \to 0} \frac{1}{\alpha} \left[f(x+\alpha) - f(x) \right] . \tag{1}$$

If this limit exists for every point of some open set $S \in \mathbf{R}$, the function f is called differentiable on S.

Among the variety of applications of the derivative one recalls here the first order approximation (linearization) of f in the neighborhood of a point x:

$$f(x+\Delta) = f(x) + f'(x)\Delta + o_x(\Delta) \tag{2}$$

with

$$\frac{o_x(\Delta)}{\Delta} \to 0 \quad \text{as } \Delta \to 0 . \tag{3}$$

Moreover, x^* is a minimum of the function f if

$$f'(x^*) = 0 . \tag{4}$$

Relation (4) defines a stationary point of f, since it also holds true for a maximum and for a saddle point of f. As is usual, higher order derivatives are checked in order to specify the nature of the stationary point.

One-Sided Differentials

Assume now that the limit (1) does not exist, but at the same time the following directional derivatives exist: the right-hand side derivative $f'_+(x)$ and the left-hand side derivative $f'_-(x)$ of f at x. The right-hand side derivative is defined by:

$$f'_+(x) = \lim_{\alpha \downarrow 0} \frac{1}{\alpha} \left[f(x+\alpha) - f(x) \right] . \tag{5}$$

Analogously, the left-hand side derivative is defined by the limit:

$$f'_-(x) = \lim_{\alpha \uparrow 0} \frac{1}{\alpha} \left[f(x+\alpha) - f(x) \right] . \tag{6}$$

Here $\alpha \downarrow 0$ means that $\alpha \to 0$, by taking positive values $\alpha > 0$ and $\alpha \uparrow 0$ means that $\alpha \to 0$, with negative values $\alpha < 0$.

It is clear that for a function f to be differentiable at x it is necessary and sufficient that $f'_+(x) = f'_-(x)$.

The *directional derivative* of a function f at point x and in the direction $x \in \mathbf{R}$ is defined by the limit:

$$f'(x,g) = \lim_{\alpha \downarrow 0} \frac{1}{\alpha} \left[f(x+\alpha g) - f(x) \right] , \tag{7}$$

if this limit exists.

The notion of directional derivative is a proper extension of the notion of the derivative. For example, it can be used to linearize a given function (cf. (2)) along a direction g. In this case relation (2) holds along a given direction, a different value holds for the opposite direction, etc, so that it provides the basis for a quasilinearization of the function f.

From the definition one may easily see that a necessary condition for a directionally differentiable function f to attain a minimum at point x^* is that:

$$f'(x^*, g) \geq 0 , \quad \forall g \in \mathbf{R} . \tag{8}$$

If strict inequality holds in (8) for every direction g not equal to zero, the condition becomes also sufficient for x^* to be a strict local minimum of f. On the other hand, a necessary condition for a directionally differentiable function f to attain a maximum at point x^* is that:

$$f'(x^{**}, g) \leq 0 , \quad \forall g \in \mathbf{R} , \tag{9}$$

with analogous implications for a strict local maximum.

A point x^* which satisfies relation (8) is called an inf-stationary point of f, while a point x^{**} satisfying (9) is called a sup-stationary point. It is interesting to observe that for a nonsmooth function first order optimality conditions may, in some cases, become sufficient for a minimum or a maximum.

Quasidifferential

A function $f\colon \mathbf{R} \to \mathbf{R}$ is called *quasidifferentiable* (q.d) at a point x if it is directionally differentiable at x and there exists a pair of closed intervals $\underline{\partial} f(x) = [v_1, v_2]$ and $\overline{\partial} f(x) = [w_1, w_2]$ such that

$$f'(x,g) = \max_{v \in \underline{\partial} f(x)} vg + \min_{w \in \overline{\partial} f(x)} wg , \quad \forall g \in \mathbf{R} . \tag{10}$$

The pair of intervals $Df(x) = [\underline{\partial} f(x), \overline{\partial} f(x)]$ is called a *quasidifferential* of f at x. The set $\underline{\partial} f(x)$ is called the

subdifferential and the set $\overline{\partial} f(x)$ the *superdifferential* of f at x.

It is clear that a quasidifferential is not uniquely defined. In fact, if a function f is quasidifferentiable at x and $Df(x)$ is its quasidifferential at this point, i. e., $Df(x) = [\underline{\partial} f(x), \overline{\partial} f(x)]$, then every pair of the form $[\underline{\partial} f(x) + C\overline{\partial} f(x) - C]$, where C is an interval $C = [c_1, c_2] \in \mathbf{R}$, with $c_1 \leq c_2$ is also a quasidifferential of f at x. In fact, the quasidifferential is a class of equivalent ordered pairs of convex sets.

Necessary and Sufficient Optimality Conditions

For a quasidifferentiable function the necessary and sufficient optimality conditions (see (8)–(9)) can be written as follows. Let $Df(x) = [\underline{\partial} f(x), \overline{\partial} f(x)]$ be a quasidifferential of f at x. A necessary condition for function f to attain a minimum at point x^* is that:

$$-\overline{\partial} f(x^*) \subset \underline{\partial} f(x^*) \,. \tag{11}$$

The condition

$$-\overline{\partial} f(x^*) \subset \operatorname{int} \underline{\partial} f(x^*) \tag{12}$$

is sufficient for x^* to be a strict local minimum of f. Analogously, a necessary condition for a maximum of f at x^{**} is that:

$$-\underline{\partial} f(x^{**}) \subset \overline{\partial} f(x^{**}) \,, \tag{13}$$

with an analogous result for a sufficient condition for a strict local maximum:

$$-\underline{\partial} f(x^{**}) \subset \operatorname{int} \overline{\partial} f(x^{**}) \,. \tag{14}$$

Finite-Dimensional Nonsmooth Functions

Subdifferentiable Functions

Let a function f defined on an open set $X \subset \mathbf{R}^n$ be directionally differentiable at a point $x \in X$. The function f is *subdifferentiable* at x if its directional derivative is a superlinear function, i. e. there exists a convex compact set U such that

$$f'(x, g) = \max_{h \in U} (h, g), \forall g \in \mathbf{R}^n \,. \tag{15}$$

Superdifferentiable Functions

A function is *superdifferentiable* at x if its directional derivative can be written by means of a convex compact set V as

$$f'(x, g) = f'_x(g) = \min_{h \in V} (h, g), \quad \forall g \in \mathbf{R}^n \,. \tag{16}$$

Quasidifferentiable Functions

A directionally differentiable function f defined on an open set $X \subset \mathbf{R}^n$ is called *quasidifferentiable* at a point $x \in X$, if there exists an ordered pair of convex compact sets $[U, V]$ in $R^n \times \mathbf{R}^n$ which produces the directional derivative of the function by:

$$f'(x, g) = f'_x(g) = \max_{h \in U} (h, g) + \min_{h \in V} (h, g), \quad \forall g \in \mathbf{R}^n \,. \tag{17}$$

Clearly, the first term on the right of (17) is a sublinear function while the second term is a superlinear function. Thus, the directional derivative of a quasidifferentiable function belongs to the space L of functions which can be written as the sum of a sublinear function and a superlinear function. Moreover with an element $[U, V]$ of the space of compact sets it is associated the class of equivalent ordered pairs of compact convex sets.

Thus, the class of equivalent ordered pairs of convex compact sets $[U, V]$ of (17) (the quasidifferential $Df(x)$ of f at x) fully describes the first order derivative of the directionally differentiable function f and gives rise to the quasilinearization (17) and, subsequently, to a qualitative and quantitative first order approximation of f in the sense of (2).

As an example, let us mention that for a differentiable function f either $Df = [\nabla f, \{0\}]$ or $Df = [\{0\}, \nabla f]$ can be used as the quasidifferential of f. For a convex, nondifferentiable function f, $Df = [\ \partial \mathrm{f}, \{0\}]$, where ∂f denotes the classical subdifferential of convex analysis [12] can be used. Analogously, for a concave function f, one may uses $Df = [\{0\}, \partial f]$, where ∂f denotes the superdifferential of the concave function f. A *difference convex function* (d.c. function) is a function f which can be expressed as the difference of two appropriately determined convex constituents, i. e., $f(x) = f_1(x) - f_2(x)$, $\forall x \in X$, where $f_1(x)$ and $f_2(x)$ are convex functions. In this case one constructs a quasidifferential simply by

$Df(x) = [\partial f_1(x), \partial f_2(x)]$, where the convex subdifferentials of the functions $f_1(x)$ and $f_2(x)$ are used.

Further Related Topics

Extension of the theory of quasidifferentiability to infinite-dimensional function spaces has not been studied till now (1999) in details. First hints can be found in [3,6].

The notion of the quasidifferential has been extended by Demyanov to the notion of the *codifferential*, which has certain advantages for numerical applications (see ► Quasidifferentiable optimization: Codifferentiable functions and [4]). Several applications of the quasidifferentiability concept and related references are given in ► Quasidifferentiable optimization: Applications and in [5].

See also

- ► Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ► Hemivariational Inequalities: Applications in Mechanics
- ► Hemivariational Inequalities: Eigenvalue Problems
- ► Hemivariational Inequalities: Static Problems
- ► Nonconvex Energy Functions: Hemivariational Inequalities
- ► Nonconvex-Nonsmooth Calculus of Variations
- ► Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ► Quasidifferentiable Optimization: Algorithms for QD Functions
- ► Quasidifferentiable Optimization: Applications
- ► Quasidifferentiable Optimization: Applications to Thermoelasticity
- ► Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ► Quasidifferentiable Optimization: Codifferentiable Functions
- ► Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ► Quasidifferentiable Optimization: Exact Penalty Methods
- ► Quasidifferentiable Optimization: Optimality Conditions
- ► Quasidifferentiable Optimization: Stability of Dynamic Systems
- ► Quasidifferentiable Optimization: Variational Formulations
- ► Quasivariational Inequalities
- ► Sensitivity Analysis of Variational Inequality Problems
- ► Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ► Variational Inequalities
- ► Variational Inequalities: F. E. Approach
- ► Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ► Variational Inequalities: Projected Dynamical System
- ► Variational Principles

References

1. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley/Interscience, New York. reprinted as: Classics Appl. Math., vol 5. SIAM, Philadelphia, PA, 1990
2. Clarke FH, Ledyaev YuS, Stern RJ, Wolenski PR (1998) Nonsmooth analysis and control theory. Springer, Berlin
3. Demyanov VF, Rubinov AM (1985) Quasidifferentiable calculus. Optim. Software, New York
4. Demyanov VF, Rubinov AM (1995) Introduction to constructive nonsmooth analysis. P. Lang, Frankfurt am Main
5. Demyanov VF, Rubinov AM (eds) (2000) Quasidifferentiability and related topics. Kluwer, Dordrecht
6. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
7. Demyanov VF, Vasiliev LN (1985) Nondifferentiable optimization. Optim. Software, New York
8. Moreau JJ (1968) La notion de sur-potentiel et les liaisons unilaté en élastostatique. CR Acad Sci Paris 267A:954–957
9. Panagiotopoulos PD (1988) Nonconvex superpotentials and hemivariational inequalities. Quasidiferentiability in mechanics. In: Moreau JJ, Panagiotopoulos PD (eds) Nonsmooth Mechanics and Applications. CISM Lect. Springer, Berlin, pp 83–176
10. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Berlin
11. Panagiotopoulos PD, Stavroulakis GE (1992) New types of variational principles based on the notion of quasidifferentiability. Acta Mechanics 94:171–194
12. Rockafellar RT (1970) Convex analysis. Math Ser, vol 28. Princeton Univ. Press, Princeton
13. Rockafellar RT (1981) The theory of subgradients and its applications to problems of optimization: Convex and nonconvex functions. Heldermann, Berlin

Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions

GEORGIOS E. STAVROULAKIS
Carolo Wilhelmina Techn. University, Braunschweig, Germany

MSC2000: 49J52, 65K99

Article Outline

Keywords
Hypodifferentiable Optimization
Comments
See also
References

Keywords

Nonsmooth optimization; Codifferentiability

Quasidifferentiability and codifferentiability extend the notion of the subdifferential of convex analysis for a quite general class of nonconvex and nonsmooth functions. If for a directionally differentiable function f: $\mathbf{R}^n \to \mathbf{R}$ there exists an ordered pair of convex compact sets $[U, V]$ in $\mathbf{R}^n \times \mathbf{R}^n$ which produces the directional derivative of f at x in the direction g by the expression:

$$f'(x,g) = \max_{h\in U}(h,g) + \min_{h\in V}(h,g)\,, \tag{1}$$

this function is called *quasidifferentiable* in the sense of V.F. Demyanov and A.M. Rubinov.

If moreover the quasidifferential of the above function is of the form $[U, 0]$ (where 0 is considered as an element of the space $\mathbf{R}^n$), then function f is called *subdifferentiable*.

More details on this notion, the calculus rules for computing quasidifferentials, its connection to other notions of nonsmooth analysis and it applications can be found in ► Quasidifferentiable optimization; ► Quasidifferentiable optimization: Calculus of quasidifferentials; ► Quasidifferentiable optimization: Dini derivatives, Clarke derivatives; ► Quasidifferentiable optimization: Applications; as well as in [1,2,3].

The quasidifferential, as well as the subdifferential of *convex analysis*, are set-valued quantities which include discontinuities at the points of nondifferentiability. In numerical algorithms this may cause problems. A notion that takes into account neighboring information would be more appropriate. This led Demyanov to extend the notion of the quasidifferential and to define the notion of the codifferential.

Let X be an open subset of $\mathbf{R}^n$ and let a function f be defined and finite for every $x \in X$. A function f is called *codifferentiable* at x if there exist convex, compact sets $\underline{d}f(x) \subset \mathbb{R}^{n+1}$ and $\overline{d}f(x) \subset \mathbb{R}^{n+1}$ such that the function admits a first order approximation in a neighborhood of x of the form

$$f(x+\Delta) = f(x) + \max_{[\alpha,v]\in \underline{d}f(x)} [\alpha + (v,\Delta)] + \min_{[b,w]\in\overline{d}f(x)} [b + (w,\Delta)] + o_x(\Delta)\,, \tag{2}$$

where $o_x(\alpha\Delta)/\alpha \to 0$, as $\alpha \downarrow 0$, $\forall \Delta \in \mathbf{R}^n$. The ordered pair of convex, compact sets $Df(x) = [\underline{d}f(x), \overline{d}f(x)]$ is called a *codifferential* of f at x, where $\underline{d}f(x)$ is a *hypodifferential* and $\overline{d}f(x)$ is a *hyperdifferential*.

If there exists a codifferential of the form $Df(x) = [\overline{d}f(x), 0]$, where 0 is considered as an element of space $\mathbf{R}^{n+1}$, the function f is called *hypodifferentiable*.

One recalls that classical convex nondifferentiable functions are subdifferentiable (resp. hypodifferentiable) in the above outlined framework, since one may use the classical convex analysis subdifferential in the above definitions for the construction of the subdifferential (resp. the hypodifferential) at a given point.

More details about codifferentiability (including extensions to higher order codifferentials) can be found in ► Quasidifferentiable optimization: Codifferentiable functions.

Hypodifferentiable Optimization

Efficient nonsmooth optimization algorithms can be constructed for hypodifferentiable functions. In fact, the technique of replacing a nondifferentiable optimization problem by an enlarged, classical, inequality constrained optimization problem has been successfully used for convex or for composite optimization problems [4,13]. For hypodifferentiable functions a direction of descent at each given point can be determined and used in an iterative optimization procedure.

Let us consider a conceptual iterative steepest descent optimization algorithm and the form it takes for nondifferentiable (hypodifferentiable) functions. First, recall that a nondifferentiable function does not possess derivatives in the classical sense. One uses set-valued approximations of the derivative (cf., the subdifferential or the hypodifferential) at the points of the nondifferentiability instead.

Accordingly optimality conditions (which will also provide the stopping rules for an optimization algorithm) and the calculation of the steepest descent direction must appropriately be modified.

The first order necessary condition for a hypodifferentiable function f to attain a minimum at point x_0 reads:

$$0 \in \underline{d} f(x_0)\,. \tag{3}$$

Points x_0 for which relation (3) is satisfied are called *inf-stationary points*. Note that the previous relation hold in the space $\mathbf{R}^{n+1}$.

If at a given point x_k, at the kth iteration of an iterative optimization scheme, relation (3) is not satisfied, then one may always find the point $\overline{z}$ with minimum norm in the closed convex set $\underline{d} f(x_k)$, such that:

$$\overline{z}^*(x_k) = (\eta^*(x_k), z^*(x_k)) = \arg \min_{\overline{z} \in \underline{d} f(x_k)} \|\overline{z}\|\,. \tag{4}$$

Since (3) is not satisfied, one has $\left\|\overline{z}^*(x_k)\right\| > 0$. The direction

$$g_k(x_k) = -\frac{z^*, (x_k)}{\|z^*(x_k)\|} \tag{5}$$

can be used as a descent direction within an optimization algorithm.

In the conceptual manner used in this note, the next step of the iterative algorithm will have the form:

$$x_{k+1} = x_k + \alpha_k x_k\,,$$

where steplength α_k will be determined from the solution of the one-dimensional optimization problem (along the direction g_k):

$$\alpha_k = \arg \min_{\alpha \geq 0} \{f(x_k + \alpha g_k)\}\,.$$

For more general quasidifferentiable and codifferentiable functions one may construct appropriate solution algorithms, see ▶ Quasidifferentiable optimization: Algorithms for QD functions and in the original literature (see, e. g., [1]).

Comments

Nondifferentiable optimization procedures have attracted the attention of several researchers and practitioners in the last decade. The lost of information which is connected with smoothing approaches is, for several applications, critical for the quality of the results. Beyond the quasidifferentiable optimization literature, previously mentioned in this note, general methods and theory for descent type methods for nonsmooth functions can be found in [7,12]. In this respect, the bundle concept has been found useful (see, among others, [6,8,9,11]). An application of this method for the solution of hemivariational inequality problems arising in mechanics can be found in [10] and [5].

Closing one would like to mention again the additional requirements of nonsmooth optimization with respect to classical, smooth one. First, stopping criteria must take into account the set-valued nature of the nonsmooth optimality conditions. Otherwise cycling in an iterative scheme or premature exit at a noncritical point may occur. This is the more critical point. Moreover, the line search must take into account the nondifferentiability of the involved function. This requirement is, usually, easily taken into account (for instance, by means of a derivative-free technique).

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-Nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions

- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Demyanov VF, Rubinov AM (1995) Introduction to constructive nonsmooth analysis. P. Lang, Frankfurt am Main
2. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
3. Demyanov VF, Vasiliev LN (1985) Nondifferentiable optimization. Optim. Software, New York
4. Fletcher R (1990) Practical methods of optimization. Wiley, New York
5. Haslinger J, Miettinen M, Panagiotopoulos PD (1999) Finite element method for hemivariational inequalities. Kluwer, Dordrecht
6. Hiriart-Urruty J-B, Lemaréchal C (1993) Convex analysis and minimization algorithms. Springer, Berlin
7. Kiwiel KC (1985) Methods of descent for nondifferentiable optimization. Springer, Berlin
8. Lemarechal C, Mifflin R (eds) (1978) Bundle methods in nonsmooth optimization. Pergamon, Oxford
9. Mäkelä MM, Neittaanmäki P (1992) Nonsmooth optimization. World Sci., Singapore
10. Miettinen M, Mäkelä MM, Haslinger J (1995) On numerical solution of hemivariational inequalities by nonsmooth optimization methods. J Global Optim 8(4):401–425
11. Schramm H, Zowe J (1992) A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. SIAM J Optim 2:121–152
12. Shor NZ (1985) Minimization methods for nondifferentiable functions. Springer, Berlin
13. Womersley RS, Fletcher R (1986) An algorithm for composite nonsmooth optimization problems. J Optim Th Appl 48:493–523

Quasidifferentiable Optimization: Algorithms for QD Functions

VLADIMIR F. DEMYANOV
St. Petersburg State University, St. Petersburg, Russia

MSC2000: 90Cxx, 65Kxx

Article Outline

Keywords
Codifferentiable (c.d.) Functions
Method of Codifferential Descent (MCD)
Method of Hypodifferential Descent (MHD)
Difference of Convex (d.c.) Functions
Difference of Max-Type (d.m.) Functions
Twice Codifferentiable Functions
Quasidifferentiable Programming Problems
See also
References

Keywords

Quasidifferentiable function; Codifferentiable function; Method of codifferential descent; Method of hypodifferential descent; D.c. function

Codifferentiable (c.d.) Functions

$f\colon \mathbf{R}^n \to \mathbf{R}$ is called *quasidifferentiable* at $x \in \mathbf{R}^n$ if it is directionally differentiable (in the sense of Dini or Hadamard) and there exists a pair $\mathcal{D}f(x) = \left[\underline{\partial} f(x), \overline{\partial} f(x)\right]$ of compact convex sets of $\mathbf{R}^n$ such that

$$f'(x,g) = \max_{v \in \underline{\partial} f(x)} (v,g) + \min_{w \in \overline{\partial} f(x)} (w,g)\,. \tag{1}$$

Here $f'(x,g)$ is either the Dini or Hadamard derivative of f at x in a direction $g \in \mathbf{R}^n$. (See ▶ Quasidifferentiable optimization: Optimality conditions).

In the sequel we discuss only the problem of minimizing the function f. In ► Quasidifferentiable optimization: Optimality conditions, necessary conditions for a minimum of f were formulated in terms of quasidifferentials (q.d.) and a formula for computing steepest descent directions was derived. However, it is difficult to apply steepest descent directions for constructing numerical methods for minimizing the function f since the quasidifferential mapping $\mathcal{D}f$ is, in general, discontinuous in the Hausdorff metric. This is why we need some other tool to overcome the discontinuity of $\mathcal{D}f$.

A function $f\colon \mathbf{R}^n \to \mathbf{R}$ is called *Dini codifferentiable* (D.c.d.) at $x \in \mathbf{R}^n$ if there exists a pair $Df(x) = [\underline{d}f(x), \overline{d}f(x)]$ of compact convex sets of $\mathbf{R}^{n+1}$ such that

$$f(x+\Delta) = f(x) + \max_{[a,v]\in \underline{d}f(x)} \big[a + (v,\Delta)\big] + \min_{[b,w]\in \overline{d}f(x)} \big[b + (w,\Delta)\big] + o_x(\Delta)\,. \tag{2}$$

$$\frac{o_x(\alpha\Delta)}{\alpha} \to 0\,, \quad \forall \Delta \in \mathbf{R}^n\,. \tag{3}$$

Here $a, b \in \mathbf{R}$; $v, w \in \mathbf{R}^n$. If in (2)

$$\frac{o_x(\Delta)}{\|\Delta\|} \overset{\|\Delta\|\to 0}{\to} 0\,, \quad \forall \Delta \in \mathbf{R}^n\,, \tag{4}$$

then f is called *Hadamard codifferentiable* (H.c.d) at x.

Without loss of generality it may be assumed that

$$\max_{[a,v]\in \underline{d}f(x)} a = \min_{[b,w]\in \overline{d}f(x)} b = 0\,. \tag{5}$$

If it causes no misunderstanding, we shall use the term codifferentiable (c.d.) for both Dini and Hadamard codifferentiable functions.

The pair $Df(x) = [\underline{d}f(x), \overline{d}f(x)]$ is called a *codifferential* of f at x, $\underline{d}f(x)$ is a *hypodifferential*, $\overline{d}f(x)$ is a *hyperdifferential*. A codifferential (like quasidifferential) is not uniquely defined. If there exists a codifferential of the form $Df(x) = [\underline{d}f(x), \{0_{n+1}\}]$, the function f is called *hypodifferentiable* at x. If there exists a codifferential of the form $Df(x) = [\{0_{n+1}\}, \overline{d}f(x)]$, the function f is called *hyperdifferentiable* at x.

It is easy to see that the class of Dini (Hadamard) codifferentiable functions coincides with the class of Dini (Hadamard) quasidifferentiable functions.

For example, if $Df(x) = [\underline{d}f(x), \overline{d}f(x)]$ is a codifferential of f at x such that (5) holds, then the pair $\mathcal{D}f(x) = [\underline{\partial}f(x), \overline{\partial}f(x)]$, where

$$\underline{\partial}f(x) = \{v \in \mathbf{R}^n\colon\ [0,v] \in \underline{d}f(x)\}\,,$$
$$\overline{\partial}f(x) = \left\{w \in \mathbf{R}^n\colon\ [0,w] \in \overline{d}f(x)\right\}\,,$$

is a quasidifferential of f at x.

A function f is called *continuously codifferentiable* at x if it is codifferentiable in some neighborhood of x and there exists a codifferential mapping Df which is Hausdorff continuous at x.

Remark 1 Of course, it is possible to introduce the notion of continuously quasidifferentiable function; however, if f is continuously q.d. at x then it is just differentiable at x.

For a fixed Δ the functions (see (1) and (2))

$$\Phi_{1x}(\Delta) = f(x) + \max_{v\in\underline{\partial}f(x)} (v,\Delta) + \min_{w\in\overline{\partial}f(x)} (w,\Delta)$$

and

$$\Phi_{2x}(\Delta) = f(x) + \max_{[a,v]\in \underline{d}f(x)} \big[a + (v,\Delta)\big] + \min_{[b,w]\in \overline{d}f(x)} \big[b + (w,\Delta)\big]$$

are both first order approximations of f in a neighborhood of x. The function $F_1(\Delta) = \Phi_{1x}(\Delta) - f(x)$ is positively homogeneous (of degree one) in Δ while the function $F_2(\Delta) = \Phi_{2x}(\Delta) - f(x)$ is, in general, not positively homogeneous. The loss of homogeneity is the price to be paid for the continuity (if any) of the codifferential mapping.

Note again that the 'value' of the mapping $\mathcal{D}f$ at any x is a pair of convex compact sets in $\mathbf{R}^{n+1}$.

If turns out that most of the known functions are continuously codifferentiable (see [3,4]). For example, all smooth, convex, concave and concavo-convex functions are continuously codifferentiable. The class of c.d. functions enjoys a very rich calculus similar to that for q.d. functions (see ► Quasidifferentiable optimization: Optimality conditions) which is a generalization of the classical differential calculus. The class of c.d. functions was introduced in [3].

First we discuss the problem of minimizing a c.d. function on the entire space (in the absence of constraints).

For a c.d. function the following necessary condition holds:

Proposition 2 *For a point $x^* \in \boldsymbol{R}^n$ to be a minimizer of f on $\boldsymbol{R}^n$ it is necessary that*

$$0_{n+1} \in \{\underline{d}f(x^*) + [0, w]\}\,, \quad \forall [0, w] \in \overline{d}f(x^*) \tag{6}$$

(it is assumed that condition (5) holds).

A point x^* satisfying (6) is called *inf-stationary*. Let x be not inf-stationary. Then there exists $\overline{w} = [0, w] \in \overline{d}f(x)$ such that

$$0_{n+1} \notin \{\underline{d}f(x^*) + \overline{w}\} = L_{\overline{w}}(x)\,. \tag{7}$$

Find

$$\min_{\overline{z} \in L_{\overline{w}}(x)} \|\overline{z}\| = \|\overline{z}_{\overline{w}}(x)\|\,.$$

(7) implies that

$$\overline{z}_{\overline{w}}(x) = \left[\eta_{\overline{w}}(x), z_{\overline{w}}(x)\right] \neq 0_{n+1}$$
$$(\eta_{\overline{w}}(x) \in \mathbf{R}\,, \quad z_{\overline{w}}(x) \in \mathbf{R}^n)\,.$$

It is also possible to show that $z_{\overline{w}}(x) \neq 0_n$ and that for the direction

$$g_{\overline{w}}(x) = -\frac{z_{\overline{w}}(x)}{\|z_{\overline{w}}(x)\|}$$

the inequality $f'(x, g_{\overline{w}}(x)) \leq -\|z_{\overline{w}}(x)\|$ holds.

Method of Codifferential Descent (MCD)

Let a function f be defined, Lipschitz and continuously codifferentiable on $\mathbf{R}^n$. Fix any $\mu > 0$. Choose an arbitrary $x_0 \in \mathbf{R}^n$. Let x_k have already been found. If condition (6) holds at x_k, then x_k is inf-stationary and the process terminates. Otherwise, for every $\overline{w} \in \overline{d}_\mu f(x_k)$ where

$$\overline{d}_\mu(x) = \left\{\overline{w} \in \overline{d}f(x):\ \overline{w} = (\omega, w),\ 0 \leq \omega \leq \mu\right\} \tag{8}$$

we find

$$\min_{\overline{z} \in L_{\overline{w}}(x_k)} \|\overline{z}\| = \|\overline{z}_{k\overline{w}}\|$$

with

$$\overline{z}_{k\overline{w}} = [\eta_{k\overline{w}}, z_{k\overline{w}}]\,, \quad L_{\overline{w}}(x_k) = [\underline{d}f(x_k) + \overline{w}]\,.$$

Now, for every $\overline{w} \in \overline{d}_\mu f(x_k)$ we find

$$\min_{\alpha \geq 0} f(x_k - \alpha z_{k\overline{w}}) = f(x_k - \alpha_{k\overline{w}} z_{k\overline{w}}) \tag{9}$$

and then

$$\min_{\overline{w} \in \overline{d}_\mu f(x_k)} f(x_k - \alpha_{k\overline{w}} z_{k\overline{w}}) = f(x_k - \alpha_{k\overline{w}_k} z_{k\overline{w}_k})\,.$$

Put $x_{k+1} = x_k - \alpha_{k\overline{w}_k} z_{k\overline{w}_k}$. Continuing in the same manner we construct a sequence $\{x_k\}$ such that $f(x_{k+1}) < f(x_k)$.

Proposition 3 *(See [4, Thm. V.5.1].) Let the set $\{x \in \boldsymbol{R}^n: f(x) \leq f(x_0)\}$ be bounded, x^* be a limit point of the sequence $\{x_k\}$ and let relation (4) hold uniformly in x from some neighborhood of x^* and in Δ from $S = \{\Delta \in \boldsymbol{R}^n: \|\Delta\| = 1\}$.*

Then the point x^* is an inf-stationary point of f (i. e. condition (6) holds).

Remark 4 The above described MCD is a conceptual method (according to the terminology of E. Polak). It should be adjusted to a specific class of functions. The MCD is a generalization of the classical steepest descent method.

For example, if for every $x \in \mathbf{R}^n$ the set $\overline{d}f(x)$ is the convex hull of a finite number of points then in (8) one can take only points $\overline{w} = (w, \omega)$ which are 'vertices' of $\overline{d}f$ such that $0 \leq \omega \leq \mu$.

In this case at each step it is required to solve only a finite number of one-dimensional optimization problems (9).

Method of Hypodifferential Descent (MHD)

Let f be defined, Lipschitz and continuously hypodifferentiable on $\mathbf{R}^n$, i. e. there exists a codifferential mapping of the form $Df(x) = [\underline{d}f(x), \{0_{n+1}\}]$ which is Hausdorff continuous. Then the necessary condition for a minimum (6) takes the form

$$0_{n+1} \in \underline{d}f(x^*)\,. \tag{10}$$

If $x \in \mathbf{R}^n$ is not an inf-stationary point (i. e., (10) does not hold) then let us find

$$\min_{\overline{z} \in \underline{d}f(x)} \|\overline{z}\| = \|(\eta(x), z(x))\| = \rho(x) = \|\overline{z}(x)\|\,.$$

Since $\rho(x) > 0$ then $\overline{z}(x) \neq 0_{n+1}$. It is possible to show that $z(x) \neq 0_n$. The direction $g(x) = -z(x)/\|z(x)\|$

is a descent direction (not necessarily the steepest descent direction). The vector function $g(x)$ is continuous at any point which is not inf-stationary.

Take any $x_0 \in \mathbf{R}^n$. Let x_k have already been constructed. Let us find $\rho(x_k) = \|\overline{z}(x_k)\|$. If $\rho(x_k) = 0$ then x_k is inf-stationary and the process terminates. Otherwise, take $g_k = -z(x_k)/\parallel z(x_k) \parallel$ and find

$$\min_{\alpha \geq 0} f(x_k + \alpha g_k) = f(x_k + \alpha_k g_k) .$$

Put $x_{k+1} = x_k + \alpha_k x_k$. Continuing analogously we construct a sequence $\{x_k\}$ such that $f(x_{k+1}) < f(x_k)$.

Proposition 5 *Let x^* be a limit point of the sequence $\{x_k\}$ and the hypotheses of Proposition 3 hold. Then $0_{n+1} \in \mathrm{d}f(x^*)$ i. e. x^* is an inf-stationary point of f.*

Difference of Convex (d.c.) Functions

Let $f(x) = f_1(x) - f_2(x)$ where $f_1, f_2 : \mathbf{R}^n \to \mathbf{R}$ are convex. A d.c. function is quasidifferentiable with the quasidifferential $\mathcal{D}f(x) = [\underline{\partial} f(x), \overline{\partial} f(x)]$ where $\underline{\partial} f(x) = \partial f_1(x)$, $\overline{\partial} f(x) = -\partial f_2(x)$, $\partial f_1(x)$ and $\partial f_2(x)$ are *subdifferentials* (in the sense of convex analysis) of the functions f_1 and f_2 respectively:

$$\partial f_i(x) = \left\{ v \in \mathbf{R}^n : \begin{array}{c} f_i(z) - f_i(x) \geq (v, z - x), \\ \forall z \in \mathbf{R}^n \end{array} \right\} .$$

The sets ∂f_i are convex and compact. The necessary condition for a minimum (6) takes the form

$$\partial f_2(x^*) \subset \partial f_1(x^*) . \tag{11}$$

If f_2 is a polyhedral function (i. e. $f_2(x) = \max_{i \in I}\{a_i + (v_i, x)\}$ where $a_i \in \mathbf{R}$, $v_i \in \mathbf{R}^n$, $I = 1, \ldots, N$) then condition (11) is sufficient for the point x^* to be a local minimizer of f.

Since the mappings ∂f_1 and ∂f_2 are discontinuous then $\mathcal{D}f$ is also discontinuous.

If F is a convex function, $\varepsilon \geq 0$ then the set

$$\partial_\varepsilon F(x) = \left\{ v \in \mathbf{R}^n : \begin{array}{c} F(z) - F(x) \\ \geq (v, z - x) - \varepsilon, \\ \forall z \in \mathbf{R}^n \end{array} \right\}$$

is called the *ε-subdifferential* of F at x.

We shall use the following properties of a convex function (see, e. g., [7]):

1) $\partial_\varepsilon F(x)$ is a closed compact set.

2) The mapping $\partial_\varepsilon F$ is Hausdorff continuous jointly in ε and x on $(0, \infty) \times \mathbf{R}^n$.

3)

$$\max_{v \in \partial_\varepsilon F(x)} (v, g) = \inf_{\alpha \geq 0} \frac{1}{\alpha} \left[F(x + \alpha g) - F(x) + \varepsilon \right] := F'_\varepsilon(x, g) .$$

In [6] the following necessary and sufficient condition for a global minimum is stated:

For a point x^* to be a global minimizer of a d.c. function $f(x) = f_1(x) - f_2(x)$ it is necessary and sufficient that

$$\partial_\varepsilon f_2(x^*) \subset \partial_\varepsilon f_1(x^*) , \quad \forall \varepsilon \geq 0 . \tag{12}$$

Note that if $\varepsilon_1 > \varepsilon_2$ and

$$f'_{\varepsilon_1 \varepsilon_2}(x, g) := f'_{1\varepsilon_1}(x, g) - f'_{2\varepsilon_2}(x, g) \leq 0 \tag{13}$$

then

$$\inf_{\alpha \geq 0} f(x + \alpha g) \leq f(x) + \varepsilon_2 - \varepsilon_1 . \tag{14}$$

Let us construct the following method for finding an inf-stationary point of f (i. e. a point satisfying (11)).

Fix ε_0, $\mu_0 = \varepsilon_0/2$. Take an arbitrary $x_{00} \in \mathbf{R}^n$. Assume that the set

$$C = \{x \in \mathbf{R}^n : \ f(x) \leq f(x_{00})\}$$

is bounded (then it is closed since f is continuous). If

$$B_{00} := \partial_{\mu_0} f_2(x_{00}) \subset A_{00} := \partial_{\varepsilon_0} f_1(x_{00})$$

then we put $x_0 = x_{00}$. If

$$\partial_{\mu_0} f_2(x_{00}) \not\subset \partial_{\varepsilon_0} f_1(x_{00})$$

then let us find

$$\max_{w \in B_{00}} \min_{v \in A_{00}} \|v - w\| = \|v_{00} - w_{00}\| = \rho_{00}$$

and put $g_{00} = (w_{00} - v_{00})/\parallel w_{00} - v_{00} \parallel$. Since $\rho_{00} > 0$ then

$$f'_{\varepsilon_0 \mu_0}(x_{00}, g_{00}) < 0$$

and, by (13) and (14), we conclude that

$$\inf_{\alpha\geq 0} f(x_{00}+\alpha g_{00}) = f(x_{00}+\alpha_{00}g_{00}) \leq f(x_{00}) - \frac{\varepsilon_0}{2}\,. \tag{15}$$

Now take $x_{01} = x_{00} + \alpha_{00}g_{00}$ and check the condition

$$B_{01} := \partial_{\mu_0} f_2(x_{01}) \subset A_{01} := \partial_{\varepsilon_0} f_1(x_{01})\,.$$

Continuing in the same manner, in a finite number of steps we shall find a point x_{0s_0} such that

$$B_{0s_0} := \partial_{\mu_0} f_2(x_{0s_0}) \subset A_{0s_0} := \partial_{\varepsilon_0} f_1(x_{0s_0}) \tag{16}$$

(it is due to (15) and the boundedness of C).

Put $x_0 = x_{0s_0}$. By (16)

$$B_0 := \partial_{\mu_0} f_2(x_0) \subset A_0 := \partial_{\varepsilon_0} f_1(x_0)\,.$$

Let x_k be constructed such that

$$\partial_{\mu_i} f_2(x_k) \subset \partial_{\varepsilon_i} f_1(x_k)\,,\quad \forall i \in 0,\ldots,k\,, \tag{17}$$

where $\mu_i = \mu_0/2^i$, $\varepsilon = \varepsilon_0/2^i$.

Put $x_{k+1,0} = x_k$. If

$$\partial_{\mu_{k+1}} f_2(x_{k+1,0}) \subset \partial_{\varepsilon_{k+1}} f_1(x_{k+1,0}) \tag{18}$$

then we take $x_{k+1} = x_{k+1,0}$. If (18) does not hold, we continue as above and in a finite number of steps a point $x_{k+1,s_{k+1}}$ will be found such that

$$\partial_{\mu_{k+1}} f_2(x_{k+1,s_{k+1}}) \subset \partial_{\varepsilon_{k+1}} f_1(x_{k+1,s_{k+1}})$$

and we put $x_{k+1} = x_{k+1,s_{k+1}}$.

As a result we construct a bounded sequence $\{x_k\}$ satisfying (17).

Proposition 6 *Any limit point of the sequence $\{x_k\}$ is an inf-stationary point of f.*

Difference of Max-Type (d.m.) Functions

Let

$$f(x) = f_1(x) - f_2(x)$$

where $f_1, f_2\colon \mathbf{R}^n \to \mathbf{R}$ are max-type functions:

$$f_1(x) = \max_{y\in G_1} \varphi_1(x,y)\,,$$

$$f_2(x) = \max_{y\in G_2} \varphi_2(x,y)$$

where φ_1 and φ_2 are continuous on $\mathbf{R}^n \times G_1$ and $\mathbf{R}^n \times G_2$, respectively, and there exist derivatives ${\varphi_1}_x'$ and ${\varphi_2}_x'$ which are continuous. The function f (called a *d.m. function*) is quasidifferentiable. It is also continuously codifferentiable:

$$f(x+\Delta) = f(x) + \max_{[a,v]\in \underline{d}f(x)} \big[a + (v,\Delta)\big] + \min_{[b,w]\in \overline{d}f(x)} \big[b + (w,\Delta)\big] + o_x(\Delta)\,,$$

where

$$\underline{d}f(x) = \operatorname{co}\left\{ [a,v]\colon \begin{array}{c} a = \varphi_1(x,y) - f_1(x), \\ v = \varphi_{1x}'(x,y), \\ y \in G_1 \end{array} \right\} \subset \mathbf{R}\times\mathbf{R}^n,$$

$$\overline{d}f(x) = \operatorname{co}\left\{ [b,w]\colon \begin{array}{c} b = f_2(x) - \varphi_2(x,y), \\ w = -\varphi_{2x}'(x,y), \\ y \in G_2 \end{array} \right\} \subset \mathbf{R}\times\mathbf{R}^n\,.$$

Here $a, b \in \mathbf{R}$; $v, w \in \mathbf{R}^n$.

Now it is possible to employ the MCD for finding inf-stationary points.

Twice Codifferentiable Functions

A function $f\colon \mathbf{R}^n \to \mathbf{R}$ is called *twice codifferentiable* at $x \in \mathbf{R}^n$ if there exist convex compact sets $d^2 f(x)$ and $\overline{d}^2 f(x) \subset \mathbf{R}\times\mathbf{R}^n\times\mathbf{R}^{n\times n}$ such that

$$f(x+\Delta) = f(x) + \max_{[a,v,A]\in \underline{d}^2 f(x)} \left[a + (v,\Delta) + \frac{1}{2}(A\Delta,\Delta)\right] + \min_{[b,w,B]\in \overline{d}^2 f(x)} \left[b + (w,\Delta) + \frac{1}{2}(B\Delta,\Delta)\right] + o(\Delta^2)$$

where

$$\frac{o((\alpha\Delta)^2)}{\alpha^2} \overset{\alpha\to 0}{\to} 0\,,\quad \forall \Delta \in \mathbf{R}^n\,.$$

Here $\mathbf{R}^{n\times n}$ is the space of real $(n\times n)$-matrices.

The pair of sets $D^2 f(x) = [\underline{d}^2 f(x), \overline{d}^2 f(x)]$ is called a *second order codifferential* of f at x. If f is twice c.d. in some neighborhood of x and the mapping $D^2 f$ is Hausdorff continuous at x, then the function is called *twice continuously codifferentiable* at x.

The class of twice c.d. functions is quite rich and enjoys a well-developed calculus (see [4]).

Let f be twice continuously c.d. on $\mathbf{R}^n$. Then the following second order Newton-type method can be employed to find inf-stationary points of f.

Take any $x_0 \in \mathbf{R}^n$. Let x_k have already been constructed. Put

$$F_k(\Delta) = \max_{[a,v,A]\in \underline{d}^2 f(x_k)} \left[a + (v,\Delta) + \frac{1}{2}(A\Delta,\Delta)\right] + \min_{[b,w,B]\in \overline{d}^2 f(x_k)} \left[b + (w,\Delta) + \frac{1}{2}(B\Delta,\Delta)\right],$$

find

$$\min_{\Delta\in\mathbf{R}^n} F_k(\Delta) = F_k(\Delta_k)$$

and take $x_{k+1} = x_k + \Delta_k$.

The sequence $\{x_k\}$ thus constructed converges (under some additional assumptions) to an inf-stationary point of f (see [1]).

Quasidifferentiable Programming Problems

Let functions f and h_i: $\mathbf{R}^n \to \mathbf{R}$ $(i \in I = 1, \ldots, N)$ be quasidifferentiable on $\mathbf{R}^n$ and let

$$\Omega = \{x \in \mathbf{R}^n : \ h_i(x) \le 0, \ \forall i \in I\} .$$

Assume that $\Omega \neq \emptyset$.

It is required to find

$$(\mathrm{P}) \min_{x\in\Omega} f(x) = f^* .$$

The set Ω is called *quasidifferentiable*, problem (P) is a quasidifferentiable (q.d.) programming problem. Necessary conditions for a minimum of f on Ω are stated in ► Quasidifferentiable optimization: Optimality conditions. If all the functions f and f_i' are, in addition, continuously codifferentiable then it is possible to extend the MCD to problem (P) (see [4]).

Another approach to problem (P) is based on the penalization technique.

We say that problem (P) is *calm* if

$$\limsup_{\varepsilon\downarrow 0} \frac{f^* - f_\varepsilon}{\varepsilon} \le B < \infty \tag{19}$$

where

$$f_\varepsilon = \inf_{x\in\Omega_\varepsilon} f(x) ,$$

$$\Omega_\varepsilon = \{x \in \mathbf{R}^n : \ h_i(x) \le \varepsilon, \ \forall i \in I\} ,$$

$$\varepsilon > 0 .$$

Proposition 7 *If the calmness condition (19) holds then there exists $A^* < \infty$ such that, for any $A > A^*$, the set of minimizers of the function f on Ω coincides with the set of minimizers of the function*

$$F(x, A) = f(x) + A\sum_{i\in I} h_i^+(x) \tag{20}$$

on $\mathbf{R}^n$. Here $h_i^+(x) = max\{0, h_i(x)\}$.

Remark 8 Thus, the constrained optimization problem (P) is reduced to the unconstrained one. Since the function $F(x, A)$ is again quasidifferentiable, one can use methods for unconstrained optimization. Another condition (different from (19)) under which Proposition 7 is valid was stated in [2].

Remark 9 Problem (P) is called a *d.c. programming problem* if all the functions f and h_i' $(i \in I)$ are d.c., i. e. $f = f_1 - f_2$, $h_i = h_{1i} - h_{2i}$ where f_1, f_2, h_{1i}, h_{2i} are convex. If the calmness condition (19) holds then, by Proposition 7, problem (P) is reduced to that of minimizing the function $F(x, A)$ (see (20)) (if A is sufficiently large). We have

$$h_i^+(x) = \max\{0, h_i(x)\} = \max\{0, h_{1i}(x) - h_{2i}(x)\} = \overline{h}_{1i}(x) - \overline{h}_{2i}(x) ,$$

where

$$\overline{h}_{1i}(x) = \max\{h_{1i}(x), h_{2i}(x)\} ,$$
$$\overline{h}_{2i}(x) = h_{1i}(x) + h_{2i}(x) .$$

The functions $\overline{h}_{1i}$ and $\overline{h}_{2i}$ are convex, therefore h_i^+ is d.c. and, hence, the function $F(x, A)$ is also d.c. and one may use the method described above for d.c. functions.

See also

- ► Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ► Hemivariational Inequalities: Applications in Mechanics
- ► Hemivariational Inequalities: Eigenvalue Problems
- ► Hemivariational Inequalities: Static Problems
- ► Nonconvex Energy Functions: Hemivariational Inequalities
- ► Nonconvex-Nonsmooth Calculus of Variations

- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Demyanov VF (1995) Fixed point theorem in nonsmooth analysis and its applications. Numer Funct Anal Optim 16(1–2):53–110
2. Demyanov VF, Di Pillo G, Facchinei F (1998) Exact penalization via Dini and Hadamard conditional derivatives. Optim Methods Softw 8:19–36
3. Demyanov VF, Rubinov AM (1980) On quasidifferentiable functionals. Dokl USSR Acad Sci (2) 250(1):21–25
4. Demyanov VF, Rubinov AM (1995) Constructive nonsmooth analysis. P. Lang, Frankfurt am Main
5. Demyanov VF, Vasiliev LV (1986) Nondifferentiable optimization. Springer and Optim. Software, Berlin
6. Hiriart–Urruty J-B (1989) From convex optimization to nonconvex optimization. In: Clarke FH, Demyanov VF, Giannessi F (eds) Nonsmooth Optimization and Related Topics. Plenum, New York, pp 219–239
7. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton

Quasidifferentiable Optimization: Applications

GEORGIOS E. STAVROULAKIS
Carolo Wilhelmina Techn. University, Braunschweig, Germany

MSC2000: 74A55, 74M10, 74M15, 65K99, 90C26, 49J35

Article Outline

Keywords
Nonsmooth Modeling
Nonsmooth and Nonconvex Optimization
Multilevel and Marginal Function Optimization
Applications in nonsmooth mechanics
See also
References

Keywords

Nonsmooth optimization; Quasidifferentiability; Nonsmooth modeling

Quasidifferentiability and the notion of the quasidifferential extend the subdifferential of convex analysis for a quite general class of nonconvex and nonsmooth, but directionally differentiable functions. By using an ordered pair of convex sets, the quasidifferential copes in a nice way with both nonsmoothness and nonconvexity issues. Since its introduction by V.F. Demyanov, a number of quasidifferential optimization problems have been studied. Moreover calculus rules have been developed and applications, among others in mechanics and engineering [5] have been considered. In addition, the related, more appropriate for numerical purposes notion of the codifferential has been introduced.

Let us consider a classical optimization algorithm, the (anti)gradient optimization, and how it is modified for quasidifferentiable functions. First, recall that a nondifferentiable function does not has derivatives in

the classical sense. One should use set-valued approximations at the points of the nondifferentiability instead. This replacement introduces the following two problems in a gradient optimization algorithm. First, one has to calculate an appropriate direction of descent, which should be followed at a given iteration step. Moreover, the optimality conditions for a nondifferentiable function have a form dictated by the set-valued approximation of the derivative (i. e., one should check, at a given point, if zero is element of a set, or if some relation between sets is satisfied). Several applications of the quasidifferentiability concept are briefly reviewed in this short article.

Nonsmooth Modeling

Let us recall first the notion of quasidifferentiability in the sense of Demyanov. A function f which is defined on an open set $X \subset \mathbf{R}^n$ and which is directionally differentiable at a point $x \in X$ is called *quasidifferentiable* if there exists an ordered pair of convex compact sets $[\underline{\partial} f(x), \overline{\partial} f(x)]$ in $\mathbf{R}^n \times \mathbf{R}^n$ which produces the directional derivative of the function by the following formula

$$f'(x,g) = \max_{w \in \underline{\partial} f(x)} \langle w, g \rangle + \min_{v \in \overline{\partial} f(x)} \langle v, g \rangle \ , \qquad (1)$$

for all directions $g \in \mathbf{R}^n$. More details are given in ► Quasidifferentiable optimization.

Relation (1) gives rise to a qualitative and quantitative nonsmooth approximation (quasilinearization) of a nonsmooth and nonconvex function f at point x. Theoretical results on nonsmooth modeling can be found, among others, in [8,9,14].

The notion of the quasidifferential gives rise to nonsmooth models, with applications in mechanics [5,11]. In particular, interesting nonconvex variational formulations can be written, as it is discussed in more detail in ► Quasidifferentiable optimization: Variational formulations. They extend the *variational inequalities*, which are valid for the convex, nondifferentiable case, and constitute a parallel development to the *hemivariational inequalities* in the sense of P.D. Panagiotopoulos (see also ► Nonconvex energy functions: Hemivariational inequalities; ► Hemivariational inequalities: Applications in mechanics as well as [12]). Furthermore, quasidifferential and codifferential optimization techniques can be used for the construction of numerical algorithms for problems of nonsmooth computational mechanics [5].

Nonsmooth and Nonconvex Optimization

The notion of the quasidifferential allows one to calculate one steepest descent direction of a quasidifferentiable function $f(x)$ at a given point x_0. Assume that at point x_0 one has the subdifferential $\underline{\partial} f(x_0)$ and the superdifferential $\overline{\partial} f(x_0)$. Then, a steepest descent direction h can be calculated by:

$$h = \frac{w_1^* + w_2^*}{\|w_1^* + w_2^*\|} \ , \qquad (2)$$

for $w_1^* \in \underline{\partial} f(x_0)$, $w_2^* \in \overline{\partial} f(x_0)$, such that

$$\|w_1^* + w_2^*\| = \max_{w_1 \in \underline{\partial} f(x_0)} \left\{ \min_{w_2 \in \overline{\partial} f(x_0)} \|w_1 + w_2\| \right\} \ .$$

Moreover, there exists necessary (and in some cases sufficient) set-valued optimality conditions for quasidifferentiable optimization problems (see ► Quasidifferentiable optimization). Thus one has whatever is needed for the construction of a numerical algorithm. Calculus rules exist for the construction of the quasidifferential (see ► Quasidifferentiable optimization: Calculus of quasidifferentials), if this is not obvious from the definition of the optimization problem. Stopping rules for an optimization algorithm can also be extracted. In fact, if the optimality criteria are satisfied, then (at least local) minimum point has been calculated. Otherwise, one can calculate a steepest descent direction by (2) and proceed with a (steepest descent like) numerical optimization scheme. In this respect the affiliated notion of the codifferentiability has certain advantages for the numerical implementation. More details can be found in ► Quasidifferentiable optimization: Codifferentiable functions and in [3,6].

It is worth noting to observe here that formula (2) may admit multiple solutions. This should be expected since one deals with nonconvex (global) optimization problems. This is actually one of the advantages of the quasidifferentiability concept since, theoretically, if one follows all possible directions of descent which may arise along an iterative algorithm one should be able to calculate multiple solutions (i. e., local minima).

More information on smooth (convex and nonconvex) optimization and appropriate algorithms devel-

oped for these problems can be found, among others, in [2,7]. Note also the multi-objective programming approach for the solution of systems of quasidifferentiable equations which has been developed in [13].

Multilevel and Marginal Function Optimization

Interesting results on the application of the quasidifferentiability concept for the sensitivity analysis and algorithms for multilevel optimization problems have been presented in [1,10].

Applications in nonsmooth mechanics

Quasidifferential modeling and optimization have been used for nonsmooth mechanics applications. As it is already mentioned these results can be found in [5,11]. A number of recent (2000) applications of quasidifferentiability can be found in [4].

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-Nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Dempe S, Pallaschke D (1997) Quasidifferentiability of optimal solutions in parametric nonlinear optimization. Optim 40(1):1–24
2. Demyanov VF (ed) (1985) Nondifferentiable optimization: motivations and applications. Springer, Berlin
3. Demyanov VF, Rubinov AM (1995) Introduction to constructive nonsmooth analysis. P. Lang, Frankfurt am Main
4. Demyanov VF, Rubinov AM (eds) (2000) Quasidifferentiability and related topics. Kluwer, Dordrecht
5. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
6. Demyanov VF, Vasiliev LN (1985) Nondifferentiable optimization. Optim. Software, New York
7. Kiwiel KC (1985) Methods of descent for nondifferentiable optimization. Springer, Berlin
8. Kuntz L (1996) Topological aspects of nonsmooth optimization. Habilitation Thesis, Univ Karlsruhe
9. Kuntz L, Scholtes S (1994) Structural analysis of nonsmooth mappings, inverse functions and metric projections. J Math Anal Appl 188:346–386
10. Luderer B (1988) Primale Dekomposition quasidifferenzierbarer Optimierungsaufgaben. Diss, Fak Math Naturwissenschaft, Techn Univ Karl-Marx-Stadt
11. Panagiotopoulos PD (1988) Nonconvex superpotentials and hemivariational inequalities. Quasidiferentiability in mechanics. In: Moreau JJ, Panagiotopoulos PD (eds) Nonsmooth Mechanics and Applications. CISM Lecture Notes. Springer, Berlin, pp 211;176
12. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Berlin

13. Pielczyk A (1991) Numerical methods for solving systems of quasidifferentiable equations. PhD Thesis, Univ. Karlsruhe, published in: Math. Systems in Economics, vol 124. Anton Hain Verlag, Frankfurt am Main
14. Scholtes S (1994) Piecewise differentiable mappings: theory and applications. Habilitation Thesis, Univ Karlsruhe

Quasidifferentiable Optimization: Applications to Thermoelasticity

GEORGIOS E. STAVROULAKIS[1,2]
[1] Carolo Wilhelmina Techn. University, Braunschweig, Germany
[2] Technical University of Crete, Chania, Greece

MSC2000: 74B99, 74D99, 74G99, 74H99, 47S40, 35R70

Article Outline

Keywords
Classical Thermoelastic Model
Quasidifferential Thermal Boundary Conditions
Variational Formulation
Quasidifferential Elastic Boundary Conditions
See also
References

Keywords

Thermoelasticity; Variational inequalities; Nonsmooth mechanics; Quasidifferentiability

Certain *semipermeability* or *temperature control* problems in *thermoelasticity*, which may be combined with analogous mechanical unilateral contact effects, can be formulated and studied in a unified framework by nonsmooth modeling techniques [7]. The theory of quasidifferentiable optimization, in the sense of V.F. Demyanov and A.M. Rubinov, provides a general framework for the treatment of both convex and nonconvex, nonsmooth modeling problems [1,2,3,4]. Coupled thermal and kinematical nonconvex unilateral effects will be modeled in the sequel by using the quasidifferentiable optimization approach. Analogous formulations which have been based on the notion of *hemivariational inequalities* have been proposed and studied for semipermeability and thermal problems by P.D. Panagiotopoulos et al. [6,7]. An extension to thermoviscoelasticity has recently been published in [5].

This short article is mainly based on the results presented in [4,7], where more details can be found.

Classical Thermoelastic Model

Let us consider a thermoelastic medium in the Euclidean space $\mathbf{R}^3$. A point is denoted by x and its coordinates with respect to a fixed Cartesian coordinate system $0x_1x_2x_3$ by x_i, $i = 1, 2, 3$. The time variable t takes values in the interval $[0, T] \subset \mathbf{R}$. Moreover, let $u = u(x, t)$ be the displacement of the material point x at time t with reference to the natural state of the body, which is characterized by zero stresses and a constant absolute temperature $\theta_0 > 0$. The density at point x of the natural state is denoted by $\rho = \rho(x)$ and the open, bounded, connected subset of $\mathbf{R}^3$ occupied by the body is denoted by Ω. As usual, the boundary Γ of Ω is assumed to be regular.

The behavior of a linear thermoelastic body is governed by the following constitutive equations for the stress tensor $\sigma = \{\sigma_{ij}\}$, $i = 1, 2, 3$, and the specific entropy deviation $\eta - \eta_0$ (where η_0 is the specific entropy of the natural state)

$$\begin{aligned}\sigma_{ij} &= t_{ij} - m_{ij}(\theta - \theta_0)\\ &= C_{ijhk}\varepsilon_{hk} - m_{ij}(\theta - \theta_0),\end{aligned} \tag{1}$$

$$\eta - \eta_0 = \frac{1}{\theta_0} c_D(\theta - \theta_0) + \frac{1}{\rho} m_{ij}\varepsilon_{ij}. \tag{2}$$

Here $\theta = \theta(x, t)$ is the absolute temperature, and $\varepsilon = \{\varepsilon_{ij}\}$ the strain tensor, which is related to the displacements by the small deformation elasticity relation

$$\varepsilon_{ij}(u) = \frac{1}{2}(u_{i,j} + u_{j,i}).$$

Here $C = \{C_{ijhk}\}$, $i, j, h, k = 1, 2, 3$, is the elasticity tensor, which satisfies the well-known symmetry and ellipticity conditions, $m = \{m_{ij}\}$ is the symmetry tensor of thermal expansion, and $c_D = c_D(x) > 0$ is the specific heat at zero strain of the body. $C(x)$, $m(x)$ and $c_D(x)$ are referred to the natural state of the body. The equations of motion read:

$$\rho u_i'' = \sigma_{ij,j} + f_i, \tag{3}$$

and the law of conservation of energy has the form

$$\rho\theta_0\eta' = -q_{i,i} + Q. \tag{4}$$

Here $f = \{f_i\}$, $f_i = f_i(x, t)$, is the volume force vector, $q = \{q_i\}$, $q_i = q_i(x, t)$, is the heat flux vector and $Q = Q(x, t)$ is the radiant heating per unit volume. *Fourier's law* of *heat conduction* reads:

$$q_i = -k_{ij}\theta_{,j}. \tag{5}$$

The symmetric tensor of thermal conductivity $k = \{k_{ij}\}$, $k_{ij} = k_{ij}(x)$, refers to the natural state of the body and satisfies the condition

$$k_{ij}a_i a_j \geq c a_i a_i, \quad \forall a = \{a_i\} \in \mathbb{R}^3,$$

where c is a positive constant. These relations lead to the following system of differential equations, which describe the *linear thermoelastic behavior of a generally nonhomogeneous and nonisotropic body*:

$$\rho u_i'' = f_i + (C_{ijhk}\varepsilon_{hk})_{,j} - \left(m_{ij}(\theta - \theta_0)\right)_{,j}, \quad \text{in } \Omega \times (0, T), \tag{6}$$

$$\rho c_D \theta' - (k_{ij}\theta_{,j})_{,i} + m_{ij}\theta_0 \varepsilon_{ij}' = Q, \quad \text{in } \Omega \times (0, T). \tag{7}$$

In the sequel the following initial conditions at $t = 0$ are assumed:

$$u_i = u_{0i}(x), \quad u_i' = u_{1i}(x) \quad \text{in } \Omega, \tag{8}$$

and

$$\theta = \overline{\theta}(x) \quad \text{in } \Omega. \tag{9}$$

Let the following bilinear forms be introduced:

$$\begin{aligned} a(u, v) &= \int_\Omega C_{ijhk}\varepsilon_{ij}(u)\varepsilon_{hk}(v)\, d\Omega, \\ (u, v) &= \int_\Omega u_i v_i \, d\Omega, \\ M_1(\theta, v) &= \int_\Omega (m_{ij}\theta)_{,j} v_i \, d\Omega, \\ M_2(u, \varphi) &= \int_\Omega m_{ij} u_{i,j} \varphi \, d\Omega, \\ K(\theta, \varphi) &= \int_\Omega k_{ij}\theta_{,j}\varphi_{,i} \, d\Omega, \\ (\theta, \varphi) &= \int_\Omega \theta\varphi \, d\Omega. \end{aligned} \tag{10}$$

Quasidifferential Thermal Boundary Conditions

In order to complete the description of the previous boundary value problem one needs to specify boundary conditions for the thermal and for the elasticity problem. First, let us assume that between the boundary temperature and the heat flux the following quasidifferential (QD) superpotential relation holds:

$$\begin{aligned} &q_i n_i = -k_{ij}\theta_{,j} n_i = w_1(\theta, t) + w_2(\theta, t), \\ &\text{with } \{w_1(\theta, t), w_2(\theta, t)\} \in Dj(\theta, t), \\ &\text{on } \Gamma_1 \times (0, T), \end{aligned} \tag{11}$$

where $\Gamma_1 \subset \Gamma$ and on the remaining part of the boundary one assumes, for simplicity, that:

$$\theta = 0 \quad \text{on } \Gamma - \Gamma_1. \tag{12}$$

For the displacements, a simple boundary condition is considered:

$$u_i = 0 \quad \text{on } \Gamma \times (0, T). \tag{13}$$

Here $n = \{n_i\}$ denotes, as usual, the unit normal to Γ directed towards the exterior of Ω.

Variational Formulation

One follows here the usual way for the construction of the variational or weak formulation of the previous boundary value problem (see also ▶ quasidifferentiable optimization: variational formulations; ▶ hemivariational inequalities: applications in mechanics). Let the virtual variations $v - u'$ and $\varphi - \theta$ are sufficiently smooth. Then, by multiplying (6) and (7) by $v - u'$ and $\varphi - \theta$ respectively, integrating over Ω, and using the Green—Gauss theorem, one obtains the variational equalities

$$\begin{aligned} &(\rho u'', v - u') + a(u, v - u') + M_1(\theta - \theta_0, v - u') \\ &= (f, v - u') + \int_\Gamma t_{ij} n_j (v_i - u_i')\, d\Gamma \\ &\text{in } \Omega \times (0, T) \end{aligned} \tag{14}$$

and

$$\begin{aligned} &(\rho c_D \theta', \varphi - \theta) + K(\theta, \varphi - \theta) + M_2(\theta_0 u', \varphi - \theta) \\ &= (Q, \varphi - \theta) + \int_\Gamma k_{ij}\theta_{,j} n_i (\varphi - \theta)\, d\Gamma \\ &\text{in } \Omega \times (0, T). \end{aligned} \tag{15}$$

Now let us assume that C_{ijhk}, k_{ij}, m_{ij}, $\rho > 0$ and $c_D > 0$ are elements of $L^\infty(\Omega)$, and that $f(t) \in [L^2(\Omega)]^3$ and $Q(t) \in L^2(\Omega)$. Moreover, the spaces $[\overset{0}{H}{}^1(\Omega)]^3$ for v, u′ and $H^1(\Omega)$ for φ, θ are introduced.

Let us recall that a QD boundary condition (for instance, the relation (11)), gives rise, due to the definition of the quasidifferential, to a min-max relation. This relation, is used for the formulation of nonconvex variational problems, as it is discussed in more details in ► quasidifferentiable optimization: variational formulations.

Thus, the variational equalities (14) and (15) are combined with the boundary conditions (11)–(13), and lead to the following variational problem: find functions u: $[0, T] \to [\overset{0}{H}{}^1(\Omega)]^3$ and θ: $[0, T] \to \Phi = H^1(\Omega)$: $\theta = 0$ on $\Gamma - \Gamma_1\}$, with $u'(t) \in [\overset{0}{H}{}^1(\Omega)]^3$, $u''(t) \in [L^2(\Omega)]^3$, $\theta'(t) \in L^2(\Omega)$, which satisfy the initial conditions and the variational expression

$$
\begin{aligned}
&(\rho u'', v - u') + a(u, v - u') + M_1(\theta - \theta_0, v - u') \\
&= (f, v - u'), \quad \forall v \in [\overset{0}{H}{}^1(\Omega)]^3,
\end{aligned}
\tag{16}
$$

and

$$
\begin{aligned}
&(\rho c_D \theta', \varphi - \theta) + K(\theta, \varphi - \theta) \\
&+ M_2(\theta_0 u', \varphi - \theta) \\
&+ \max\{\langle w_1^*, \phi - \theta\rangle : \ w_1^* \in \underline{\partial} J(\theta, t)\} \\
&+ \min\left\{\langle w_2^*, \phi - \theta\rangle : \ w_2^* \in \overline{\partial} J(\theta, t)\right\} \\
&= (Q, \varphi - \theta), \quad \forall \varphi \in \Phi.
\end{aligned}
\tag{17}
$$

Quasidifferential Elastic Boundary Conditions

Assume now simple thermal boundary conditions, i. e.,

$$
\theta = \theta_0 \quad \text{on } \Gamma \times (0, T). \tag{18}
$$

For the elasticity problem let a nonmonotone, possibly multivalued quasidifferential (QD) boundary law holds on a part Γ_S of the boundary Γ:

$$
\begin{aligned}
&-S = \{-S_i\} = \{-\sigma_{ij} n_i\} \\
&= S_1(u', x, t) + S_2(u', x, t), \\
&\{S_1(u', x, t), S_2(u', x, t)\} \in D\psi(u', x, t) \\
&\text{on } \Gamma_S \times (0, T).
\end{aligned}
\tag{19}
$$

On the remaining part of the boundary one assumes simply that:

$$
u_i = U_i \quad \text{on } \Gamma_U \times (0, T). \tag{20}
$$

Here $\Gamma = \overline{\Gamma}_U \cup \overline{\Gamma}_S$, where Γ_U and Γ_S are nonempty, disjoint, open sets, $U_i = U_i(x, t)$ is a prescribed displacement vector on Γ_U (which should be compatible with the initial conditions (8)–(9)).

In an analogous way one proceeds with the boundary value problem which is defined by the relations (6)–(9) and (18)–(19). Let v, $u' \in [H^1(\Omega)]^3$ be such that $v = u' = U'(t)$ on Γ_U and φ, $\theta \in H^1(\Omega)$ with $\varphi = \theta = \theta_0$ on Γ. In this case one gets the variational problem: find u: $[0, T] \to [H^1(\Omega)]^3$ with $u' = U$ on Γ_U and $\theta \in H^1(\Omega)$ with $\theta = \theta_0$ on Γ with $u'(t) \in [H^1(\Omega)]^3$, $u''(t) \in [L^2(\Omega)]^3$, $\theta'(t) \in L^2(\Omega)$, which satisfy the initial conditions and the variational expression

$$
\begin{aligned}
&(\rho u'', v - u') + a(u, v - u') \\
&+ M_1(\theta - \theta_0, v - u') \\
&+ \max\left\{\left\langle S_1^*, v - u'\right\rangle : \ S_1^* \in \underline{\partial} \Psi(u', t)\right\} \\
&+ \min\left\{\left\langle S_2^*, v - u'\right\rangle : \ S_2^* \in \partial \Psi(u', t)\right\} \\
&= (f, v - u'), \\
&\forall v \in [H^1(\Omega)]^3 \quad \text{with } v = U'(t) \text{ on } \Gamma_U
\end{aligned}
\tag{21}
$$

and

$$
\begin{aligned}
&(\rho c_D \theta', \varphi - \theta) + K(\theta, \varphi - \theta) \\
&+ M_2(\theta_0 u', \varphi - \theta) \\
&= (Q, \varphi - \theta), \\
&\forall \varphi \in H^1(\Omega) \quad \text{with } \varphi = \theta_0 \text{ on } \Gamma.
\end{aligned}
\tag{22}
$$

More general thermoelastic problems may be considered by considering QD laws for both the elasticity and the thermal part of the problem, or even mixed laws.

See also

► Generalized monotonicity: Applications to variational inequalities and equilibrium problems
► Hemivariational inequalities: Applications in mechanics
► Hemivariational inequalities: Eigenvalue problems
► Hemivariational inequalities: Static problems
► Nonconvex energy functions: hemivariational inequalities

- ▶ Nonconvex-nonsmooth calculus of variations
- ▶ Quasidifferentiable optimization
- ▶ Quasidifferentiable optimization: Algorithms for hypodifferentiable functions
- ▶ Quasidifferentiable optimization: Algorithms for QD functions
- ▶ Quasidifferentiable optimization: Applications
- ▶ Quasidifferentiable optimization: Calculus of quasidifferentials
- ▶ Quasidifferentiable optimization: Codifferentiable functions
- ▶ Quasidifferentiable optimization: Dini derivatives, clarke derivatives
- ▶ Quasidifferentiable optimization: Exact penalty methods
- ▶ Quasidifferentiable optimization: Optimality conditions
- ▶ Quasidifferentiable optimization: Stability of dynamic systems
- ▶ Quasidifferentiable optimization: Variational formulations
- ▶ Quasivariational inequalities
- ▶ Sensitivity analysis of variational inequality problems
- ▶ Solving hemivariational inequalities by nonsmooth optimization methods
- ▶ Variational inequalities
- ▶ Variational inequalities: F. E. approach
- ▶ Variational inequalities: Geometric interpretation, existence and uniqueness
- ▶ Variational inequalities: Projected dynamical system
- ▶ Variational principles

References

1. Demyanov VF, Rubinov AM (1983) On quasidifferentiable mappings. Math Operationsforsch Statist Ser Optim 14:3–21
2. Demyanov VF, Rubinov AM (1983) On quasidifferentiable mappings. Math Operationsforsch Statist Ser Optim 14:3–21
3. Demyanov VF, Rubinov AM (1995) Introduction to constructive nonsmooth analysis. P. Lang, Bern
4. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
5. Foutsitzi G, Stavroulakis GE (2004) A variational inequality approach to thermoviscoelasticity with monotone unilateral boundaries of kinematical and thermal type. Nonl Anal Theory Methods Appl 57:743–771
6. Haslinger J, Baniotopoulos CC, Panagiotopoulos PD (1993) A boundary multivalued integral "equation" approach to the semipermeability problem. Appl Mat 38:39–60
7. Panagiotopoulos PD (1985) Inequality problems in mechanics and applications. Convex and nonconvex energy functions. Birkhäuser, Basel

Quasidifferentiable Optimization: Calculus of Quasidifferentials

GEORGIOS E. STAVROULAKIS
Carolo Wilhelmina Techn. University, Braunschweig, Germany

MSC2000: 49J52, 65K99, 90C90

Article Outline

Keywords
One-Dimensional Case
Finite-Dimensional Case
See also
References

Keywords

Nonsmooth analysis; Nonsmooth modeling; Quasidifferentiability; Nonsmooth optimization

A function f defined on an open set $X \subset \mathbf{R}^n$ and directionally differentiable at a point $x \in X$ is called *quasidifferentiable* (in the sense of V.F. Demyanov) if there exists an ordered pair of convex compact sets $[U, V]$ in $\mathbf{R}^n \times \mathbf{R}^n$ which produces the directional derivative of the function by the following formula

$$f'(x, g) = \max_{h \in U}(h, g) + \min_{h \in V}(h, g)\,, \tag{1}$$

for all directions $g \in \mathbf{R}^n$.

Quasidifferentiability is a genuine generalization of the classical differentiability concept which is valid for smooth differentiable functions, and of the convex analysis subdifferential, which, in turn, is a set-valued differential valid for convex, nondifferentiable functions. An ordered pair of convex sets is used for the approximation of the directional derivative in (1). More

details can be found in the companion article ► Quasidifferentiable optimization, the links with other notions of nonsmooth analysis are discussed in ► Quasidifferentiable optimization: Dini derivatives, Clarke derivatives and applications are briefly presented in ► Quasidifferentiable optimization: Applications. One may also consult the original publications [2,3,4,5,6].

Using classical differential calculus and on the assumption of smooth (differentiable) functions the derivative of sums, of differences or of composite functions etc may easily be calculated. To this end one uses calculus rules and the derivatives of the involved functions (cf., the chain rule of differentiation). For quasidifferentiable functions there exist appropriate calculus rules [2]. The situation is more complicated here, since one manipulates ordered pairs of convex sets. Furthermore, calculus rules have been developed for composite functions which can be produced from a finite number of smooth constituents and from the application of a finite number of minimum or maximum operators. Moreover, as the quasidifferential of a given function is not uniquely determined (it is actually a class of equivalent ordered pairs of convex sets) one may wishes to simplify the results of such a calculus operation.

It is clear that, since quasidifferentials have found a number of applications, among others in optimization, in mechanics, in control theory and in economy, the need for refining the quasidifferential calculus and for incorporating it into automatic computational procedures (e.g, in computer algebra systems, in analogy to classical systems [1]) is obvious. For the latter task, which at the present remains open for future research efforts, use of results developed within the theory of interval analysis may be advantageous.

Calculus rules for one-dimensional functions (defined on $\mathbf{R}$) and for functions defined on $\mathbf{R}^n$ are given without proofs here. See [2,3] for more details.

One-Dimensional Case

A function in $\mathbf{R}^1$ and sets which are intervals of the real line $\mathbf{R}^1$ are considered first. Let D_1 and D_2 be two pairs of closed intervals: $D_1 = [A_1, B_1]$, $D_2 = [A_2, B_2]$, where $A_1 = [v_{11}, v_{12}]$, $B_1 = [w_{11}, w_{12}]$, $A_2 = [v_{21}, v_{22}]$, $B_2 = [w_{21}, w_{22}]$, with $v_{i1} \le v_{i2}$, $w_{i1} \le w_{i2}$, $\forall$i $\in \{1, 2\}$. Addition of intervals is defined as follows: $D = D_1 + D_2 = [A_1 + B_1, A_2 + B_2] = [A, B]$, where $A = [v_{11} + v_{21}, v_{12} + v_{22}]$ and $B = [w_{11} + w_{21}, w_{12} + w_{22}]$. Moreover, for $D = [A, B]$, $A = [v_1, v_2]$, $B = [w_1, w_2]$, $v_1 \le v_2$, $w_1 \le w_2$, multiplication by a scalar quantity λ is defined by:

$$\lambda D = \begin{cases} \lambda[A, B], & \lambda \ge 0, \\ \lambda[B, A], & \lambda < 0, \end{cases}$$

where, on the right-hand side one has $\lambda\,[A, B] = [[\lambda\, v_1, \lambda\, v_2], [\lambda\, w_1, \lambda\, w_2]]$, etc.

Based on these results concerning calculus of closed intervals one derives calculus rules for quasidifferentials in the one-dimensional case.

Let f_1 be a directionally differentiable function at a point x and let $Df_1(x) = [\underline{\partial} f_1(x), \overline{\partial} f_1(x)]$ be its quasidifferential at a point $x \in \mathbf{R}^1$: $\underline{\partial} f_1(x) = [v_{11}, v_{12}]$, $\overline{\partial} f_1(x) = [w_{11}, w_{12}]$, $v_{11} \le v_{12}$, $w_{11} \le w_{12}$. Then the function $f = \lambda\, f_1$ is also directionally differentiable at x and admits a quasidifferential of the form $Df(x) = [\underline{\partial} f(x), \overline{\partial} f(x)]$, where

$$\underline{\partial} f(x) = \begin{cases} [\lambda v_{11}, \lambda v_{12}], & \lambda \ge 0, \\ [\lambda w_{12}, \lambda w_{11}], & \lambda < 0, \end{cases}$$

$$\overline{\partial} f(x) = \begin{cases} [\lambda w_{11}, \lambda w_{12}], & \lambda \ge 0, \\ [\lambda v_{12}, \lambda v_{11}], & \lambda < 0. \end{cases}$$

If in addition, $f_1(x) \ne 0$ then the function $f = 1/f_1$ is also directionally differentiable at x and

$$Df(x) = -\frac{1}{f_1^2} Df_1(x) = [\underline{\partial} f(x), \overline{\partial} f(x)],$$

where

$$\underline{\partial} f(x) = \left[-\frac{1}{f_1^2} w_{12}, -\frac{1}{f_1^2} w_{11}\right],$$

$$\overline{\partial} f(x) = \left[-\frac{1}{f_1^2} v_{12}, -\frac{1}{f_1^2} v_{11}\right].$$

Let us consider two directionally differentiable functions f_1, f_2 at a point x and let $D f_1(x), D f_2(x)$ be their quasidifferentials

$$Df_1(x) = [\underline{\partial} f_1(x), \overline{\partial} f_1(x)],$$

$$Df_2(x) = [\underline{\partial} f_2(x), \overline{\partial} f_2(x)],$$

with the corresponding intervals denoted by:

$$\underline{\partial} f_1(x) = [v_{11}, v_{12}], \quad \overline{\partial} f_1(x) = [w_{11}, w_{12}],$$

$$\underline{\partial} f_2(x) = [v_{21}, v_{22}], \quad \overline{\partial} f_2(x) = [w_{21}, w_{22}],$$

$$v_{i1} \le v_{i2}, \quad w_{i1} \le w_{i2}, \qquad \forall i \in \{1, 2\}.$$

Then the function $f = f_1 + f_2$ is also directionally differentiable at x and one can take $Df(x) = [\underline{\partial} f(x), \overline{\partial} f(x)]$, where

$$\underline{\partial} f(x) = \underline{\partial} f_1(x) + \underline{\partial} f_2(x) = [v_1, v_2]$$
$$\overline{\partial} f(x) = \overline{\partial} f_1(x) + \overline{\partial} f_2(x) = [w_1, w_2] ,$$

with $v_1 = v_{11} + v_{21}$, $v_2 = v_{12} + v_{22}$, $w_1 = w_{11} + w_{21}$ and $w_2 = w_{12} + w_{22}$.

Analogously one proceeds with the product of two functions $f = f_1 f_2$, where

$$Df(x) = f_1(x)Df_2(x) + f_2(x)Df_1(x) = [\underline{\partial} f(x), \overline{\partial} f(x)] .$$

Furthermore, let $\varphi_i(x)$ $(i \in I = \{1, \ldots, N\})$ be directionally differentiable functions at a point x. The functions

$$f_1(x) = \max_{i \in I} \varphi_i(x) , \quad f_2(x) = \min_{i \in I} \varphi_i(x)$$

are also quasidifferentiable.

Finally, let $f(z_1, \ldots, z_m)$ be a smooth function and let $y_1, \ldots, y_m$ be quasidifferentiable functions at a point x_0. Then the function $F(x) = f(y_1(x), \ldots, y_m(x))$ is also quasidifferentiable at x_0.

One concludes that the family of quasidifferentiable functions is a linear space, closed with respect to all smooth operations, as well as the operations of taking the pointwise maximum and minimum over a finite number of functions.

Finite-Dimensional Case

In this case one needs calculus rules for pairs of convex sets of $\mathbf{R}^n$ (see, e. g. [4,6]).

Let the functions f, f_1, f_2 be quasidifferentiable at x and λ be a real number. Then the sum, the product, the function λf and the function $1/f(x)$ (or every point x such that $f(x) \neq 0$) are also quasidifferentiable and an element of their quasidifferential can be calculated as follows:

$$D(f_1 + f_2)(x) = Df_1(x) + Df_2(x) ,$$
$$D(f_1 \cdot f_2)(x) = f_1(x)Df_2(x) + f_2(x)Df_1(x) ,$$
$$D(\lambda f)(x) = \lambda Df(x) ,$$
$$D\left(\frac{1}{f}\right)(x) = -\frac{1}{f^2(x)} Df(x) .$$

Let moreover the functions $f_1, \ldots, f_m$ be defined on an open set $X \subset \mathbf{R}^n$ and be quasidifferentiable at $x \in X$. Then, the functions

$$\phi_1(x) = \max_{i \in \{1,\ldots,m\}} f_i(x), \phi_2(x) = \min_{i \in \{1,\ldots,m\}} f_i(x)$$

are quasidifferentiable at x as well. The following relations hold:

$$D\phi_j(x) = \left[\underline{\partial} \phi_j(x), \overline{\partial} \phi_j(x)\right] , \quad j = 1, 2 ,$$

with

$$\underline{\partial} \phi_1(x) = \operatorname{co} \bigcup_{k \in R(x)} \left(\underline{\partial} f_k(x) - \sum_{\substack{i \in R(x), \\ i \neq k}} \overline{\partial} f_i(x) \right) ,$$
$$\overline{\partial} \phi_1(x) = \sum_{k \in R(x)} \overline{\partial} f_k(x), \underline{\partial} \phi_2(x) = \sum_{k \in q(x)} \underline{\partial} f_k(x),$$
$$\overline{\partial} \phi_2(x) = \operatorname{co} \bigcup_{k \in Q(x)} \left(\overline{\partial} f_k(x) - \sum_{\substack{i \in Q(x), \\ i \neq k}} \underline{\partial} f_i(x) \right) .$$

Here, $[\underline{\partial} f_k(x), \overline{\partial} f_k(x)]$ is a quasidifferential of f_k at x and the following activity sets have been used:

$$R(x) = \{i \in I: \ f_i(x) = \phi_1(x)\} ,$$
$$Q(x) = \{i \in I: \ f_i(x) = \phi_2(x)\} ,$$

where $I = \{1, \ldots, n\}$.

Finally, consider the case of a composite function. Let a mapping $H(x) = (h_1(x), \ldots, h_m(x))$ be defined such that $H(x)$: $X \to Y$, where X is an open set in $\mathbf{R}^n$ and Y is an open set in $\mathbf{R}^m$ and every function h_i is quasidifferentiable at $x_0 \in X$. Let us assume that a function f is defined on Y and is Hadamard differentiable and quasidifferentiable at $y_0 = H(x_0)$. Then the composite function

$$\phi(x) = f(H(x))$$

is quasidifferentiable at x_0 and its quasidifferential $D\phi(x_0) = [\underline{\partial}\phi(x_0), \overline{\partial}\phi(x_0)]$ is expressed by the formulas:

$$\underline{\partial}\phi(x_0) = \left\{ p: \begin{array}{c} p = \sum_{i=1}^{m} (\nu^{(i)}(\lambda_i + \mu_i) - \underline{\nu}^{(i)}\lambda_i - \overline{\nu}^{(i)}\mu_i), \\ \nu = (\nu^{(1)}, \ldots, \nu^{(m)}) \in \underline{\partial} f(y_0), \\ \lambda_i \in \underline{\partial} h_i(x_0),\ \mu_i \in \overline{\partial} h_i(x_0) \end{array} \right\}$$

$$\overline{\partial}\phi(x_0) = \left\{ l: \begin{array}{c} l = \sum_{i=1}^{m} (\nu^{(i)}(\lambda_i + \mu_i) + \underline{\nu}^{(i)}\lambda_i + \overline{\nu}^{(i)}\mu_i), \\ \nu = (\nu^{(1)}, \ldots, \nu^{(m)}) \in \overline{\partial} f(y_0), \\ \lambda_i \in \underline{\partial} h_i(x_0),\ \mu_i \in \overline{\partial} h_i(x_0) \end{array} \right\},$$

where $\underline{\nu}$ and $\overline{\nu}$ are arbitrary vectors such that

$$\underline{\nu} \le \nu \le \overline{\nu}, \quad \forall \nu \in \underline{\partial} f(y_0) \cup \left(-\overline{\partial} f(y_0)\right).$$

Concrete examples and the derivation of the above rules can be found in the above given literature. One should only mention that if some of the involved sets (i. e., the subdifferential or the superdifferential) happens to be polyhedral, then certain of the previous rules can be simplified significantly (see, e. g., [7]). The latter case appears, among others, in the applications of quasidifferential calculus within a finite element method environment for applications in mechanics (see also ▶ Quasidifferentiable optimization: Variational formulations; ▶ Quasidifferentiable optimization: Applications to thermoelasticity, and [5]).

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-Nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Beltzer AI (1990) Variational and finite element methods: A symbolic computation approach. Springer, Berlin
2. Demyanov VF, Dixon LCW (1986) Quasidifferential calculus. Math Program Stud, vol 29. North-Holland, Amsterdam
3. Demyanov VF, Rubinov AM (1985) Quasidifferentiable calculus. Optim. Software, New York
4. Demyanov VF, Rubinov AM (1995) Introduction to constructive nonsmooth analysis. P. Lang, Frankfurt am Main
5. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
6. Demyanov VF, Vasiliev LN (1985) Nondifferentiable optimization. Optim. Software, New York
7. Stavroulakis GE, Polyakova LN (1996) Difference convex optimization techniques in nonsmooth computational mechanics. Optim Methods Softw 7(1):55–87

Quasidifferentiable Optimization: Codifferentiable Functions

GEORGIOS E. STAVROULAKIS
Carolo Wilhelmina Techn. University, Braunschweig, Germany

MSC2000: 65K99, 70-08, 49J52, 90C25

Article Outline

Keywords
Codifferentiable Functions
Twice Codifferentiable Functions
Applications
See also
References

Keywords

Nonsmooth optimization; Quasidifferentiability; Codifferentiability; Nonconvex optimization

If for a directionally differentiable function there exists an ordered pair of convex compact sets $[U, V]$ in $\mathbf{R}^n \times \mathbf{R}^n$ which produces the directional derivative of f at x in the direction g by the expression:

$$f'(x,g) = \max_{h\in U}(h,g) + \min_{h\in V}(h,g)\,; \tag{1}$$

this function is called *quasidifferentiable* in the sense of V.F. Demyanov and A.M. Rubinov. This notion covers a large number of structured nonconvex and nonsmooth functions, which can be used for the solution of nonconvex and global optimization problems. For instance, the class of difference convex functions is included.

More details on this notion, the calculus rules for computing quasidifferentials, its connection to other notions of nonsmooth analysis and it applications can be found in ► Quasidifferentiable optimization; ► Quasidifferentiable optimization: Calculus of quasidifferentials; ► Quasidifferentiable optimization: Dini derivatives, Clarke derivatives; ► Quasidifferentiable optimization: Applications, as well as in [1,2,3].

The quasidifferential, as well as the subdifferential of *convex analysis*, are set-valued quantities which include discontinuities at the points of nondifferentiability. In numerical algorithms this may cause problems. A notion that takes into account neighboring information would be more appropriate. This led Demyanov to extend the notion of the quasidifferential by introducing the *codifferential*. Accordingly, the notions of *subdifferential* and *superdifferential* are extended to the notions of *hypodifferential* and *hyperdifferential*. One should mention that all quasidifferentiable functions are codifferentiable as well. Moreover, calculus rules exists, in analogy to the quasidifferential calculus rules.

These notions, which are useful for the construction of numerical algorithms in nonsmooth optimization [1] and nonsmooth computational mechanics [2] are introduced in this short paper. More details are given in the cited literature and in the previously mentioned lemmas.

Codifferentiable Functions

Let X be an open subset of $\mathbf{R}^n$ and let a function f be defined and finite for every $x \in X$. A function f is called *codifferentiable* at x if there exist convex compact sets $\underline{d}f(x) \subset \mathbf{R}^{n+1}$ and $\overline{d}f(x) \subset \mathbf{R}^{n+1}$ such that the function admits a first order approximation in a neighborhood of x of the form

$$f(x+\Delta) = f(x) + \max_{[\alpha,v]\in \underline{d}f(x)} [\alpha + (v,\Delta)] + \min_{[b,w]\in\overline{d}f(x)} [b + (w,\Delta)] + o_x(\Delta)\,, \tag{2}$$

where $o_x(\alpha\ \Delta)/\alpha \to 0$ as $\alpha \downarrow 0$, $\forall \Delta \in \mathbf{R}^n$. The ordered pair of convex compact sets $Df(x) = [\underline{d}f(x), \overline{d}f(x)]$ is called a *codifferential* of f at x, where $df(x)$ is a *hypodifferential* and $\overline{d}f(x)$ is a *hyperdifferential*.

If moreover there exists a codifferential Df which is Hausdorff continuous in a neighborhood of x, the function f is called *continuously codifferentiable* at x.

If there exists a codifferential of the form $Df(x) = [df(\mathbf{x}), \{0\}]$, the function f is called *hypodifferentiable*, while if there exists a codifferential of the form $Df(x) = [\{0\}, \overline{d}f(x)]$ the function is called *hyperdifferentiable*.

Note here that for a continuously codifferentiable function the first order approximation which is based on (2) is a continuous function in both x and Δ (recall that the analogous approximation based on the quasidifferential is a continuous function of only Δ).

Twice Codifferentiable Functions

Twice codifferentiable functions present a suitable tool for constructing higher order approximations of nondifferentiable functions. They extend the notion of second order derivatives of classical smooth analysis.

Let a function f be defined on an open set $X \subset \mathbf{R}^n$ and let it be finite there. The function f is *twice codifferentiable* at $x \in X$ if there exist convex compact sets $d^2f(x)$, and $\overline{d}^2 f(x)$, both subsets of $\mathbf{R} \times \mathbf{R}^n \times \mathbf{R}^{n+n}$ such that

$$f(x+\Delta) = f(x) + \max_{[\alpha,v,A]\in \underline{d}^2 f(x)} \left[\alpha + (v,\Delta) + \frac{1}{2}(A\Delta,\Delta)\right] + \min_{[b,w,B]\in \overline{d}^2 f(x)} \left[b + (w,\Delta) + \frac{1}{2}(B\Delta,\Delta)\right] + o(\Delta^2),$$

with $o((\alpha\ \Delta)^2)/\alpha^2 \to 0$ as $\alpha \downarrow 0$ and $\forall \Delta \in \mathbf{R}^n$. Here $\mathbf{R}^{n+n}$ is the space of real $(n \times n)$-matrices.

The ordered pair of convex sets $D^2f(x) = [\underline{d}^2 f(x), \overline{d}^2 f(x)]$ is called a *second order codifferential* of f at x, the set $d^2f(x)$ is a *second order hypodifferential* and the set $d^2f(x)$ is a *second order hyperdifferential* of f at x. Moreover, if f is twice codifferentiable in some neighborhood of a point x and the mapping $D^2 f$ is Hausdorff continuous at x, then the function f is called *twice continuously codifferentiable* at x.

Analogously to the quasidifferentiable and codifferentiable functions, twice hypodifferentiable functions and twice hyperdifferentiable functions may be defined. Calculus rules do also exist for twice codifferentiable functions (see [1, p. 216]).

For example, let f be convex and finite on a convex set $X \subset \mathbf{R}^n$, $x \in X$, and let X_0 be an arbitrary closed convex and bounded subset of X with $x \in \operatorname{int} X_0$. In this case one may consider the second order codifferential $D^2f(x) = [d^2f(x), 0]>$, with

$$\underline{d}^2 f(x) = \operatorname{co}\left\{ [\alpha, v, A] : \begin{array}{c} \alpha = f(z) - f(x) \\ +(v(z), x - z), \\ v(z) \in \underline{\partial} f(z), \\ A = 0 \in \mathbf{R}^{n+n}, \\ z \in X_0 \end{array} \right\}.$$

Here $v(z) \in \partial f(z)$ is an arbitrary element of the set valued mapping, which is kept fixed for every $z \in X_0$ and $\partial f(z)$ is equal to the classical convex analysis subdifferential.

Moreover, for a twice continuously differentiable function f it is well-known that

$$f(x+\Delta) = f(x) + (f'(x), \Delta) + \frac{1}{2}(f''(x)\Delta, \Delta) + o(\Delta^2),$$

where $f''(x)$ is the matrix of second order derivatives (Hessian) of f at x. The function f is twice continuously codifferentiable and one may consider (among other choices) one of the following second order codifferentials of f:

$$\underline{d}^2 f(x) = \{[0, f'(x), f''(x)]\},$$
$$\overline{d}^2 f(x) = \{0, 0, 0\},$$

or

$$\underline{d}^2 f(x) = \{0, 0, 0\},$$
$$\overline{d}^2 f(x) = \{[0, f'(x), f'(x)]\}.$$

Applications

Efficient nonsmooth optimization algorithms can be constructed based on the notion of the codifferential, or, for hypodifferentiable functions, on the notion of the hypodifferential. In fact, the technique of replacing a nondifferentiable optimization problem by an enlarged, classical, inequality constrained optimization problem has been successfully used for convex or for composite optimization problems [4,15]. For hypodifferentiable functions a direction of descent at each given point can be determined and used in an iterative optimization procedure. For general, codifferentiable functions, several directions of descent can be determined. This can be expected, given that one deals with nonconvex, global optimization problems. Some details in this direction are given in ► Quasidifferentiable optimization: Applications and in the original publications [3,11].

Furthermore, twice (or higher order) quasidifferentials and codifferentials provide set-valued approximations of the higher order derivatives of a function. For numerical optimization tasks this information may lead to more efficient algorithms, in analogy to the use of Hessian matrices in classical, smooth optimization. Other attempts for generalized second order derivatives can be found, for convex functions in [5] and

for nonconvex functions in [10,12]. For more information in the area of nonsmooth optimization see, e. g., [6,7,8,9,13].

Another area of interest for practical applications will be the use of this information for the construction of necessary and sufficient (local)optimality conditions. Applications of these results include stability and sensitivity analysis for quasidifferentiable and codifferentiable optimization problems. In mechanics, this information can be used for the study of the stability of structures governed by quasidifferentiable superpotentials (cf. e. g., [14] and ► Quasidifferentiable optimization: Stability of dynamic systems). Applications in economics will be of interest as well. Much work remains to be done in this area, which is open for further investigations (as of 1999).

See also

- ► Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ► Hemivariational Inequalities: Applications in Mechanics
- ► Hemivariational Inequalities: Eigenvalue Problems
- ► Hemivariational Inequalities: Static Problems
- ► Nonconvex Energy Functions: Hemivariational Inequalities
- ► Nonconvex-nonsmooth Calculus of Variations
- ► Quasidifferentiable Optimization
- ► Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ► Quasidifferentiable Optimization: Algorithms for QD Functions
- ► Quasidifferentiable Optimization: Applications
- ► Quasidifferentiable Optimization: Applications to Thermoelasticity
- ► Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ► Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ► Quasidifferentiable Optimization: Exact Penalty Methods
- ► Quasidifferentiable Optimization: Optimality Conditions
- ► Quasidifferentiable Optimization: Stability of Dynamic Systems
- ► Quasidifferentiable Optimization: Variational Formulations
- ► Quasivariational Inequalities
- ► Sensitivity Analysis of Variational Inequality Problems
- ► Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ► Variational Inequalities
- ► Variational Inequalities: F. E. Approach
- ► Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ► Variational Inequalities: Projected Dynamical System
- ► Variational Principles

References

1. Demyanov VF, Rubinov AM (1995) Introduction to constructive nonsmooth analysis. P. Lang, Frankfurt am Main
2. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
3. Demyanov VF, Vasiliev LN (1985) Nondifferentiable optimization. Optim. Software, New York
4. Fletcher R (1990) Practical methods of optimization. Wiley, New York
5. Hiriart-Urruty J-B (1986) A new set-valued second-order derivative for convex functions. In: Fermat-Days 85: Mathematics for Optimization. Math Stud, vol 129. North-Holland, Amsterdam, pp 157–182
6. Hiriart-Urruty J-B, Lemaréchal C (1993) Convex analysis and minimization algorithms. Springer, Berlin
7. Kiwiel KC (1985) Methods of descent for nondifferentiable optimization. Springer, Berlin
8. Lemarechal C, Mifflin R (eds) (1978) Bundle methods in nonsmooth optimization. Pergamon, Oxford
9. Mäkelä MM, Neittaanmäki P (1992) Nonsmooth optimization. World Sci., Singapore
10. Pallaschke D, Recht P, Urbanski R (1987) On extensions of the second-order derivative. Bull Acad Polon Sci Ser Func Anal (1987):751–763
11. Polyakova LN (1986) On minimizing the sum of a convex function and a concave function. Math Program Stud 29:69–73
12. Recht P (1993) Generalized derivatives: An approach to a new gradient in non-smooth optimization. PhD Diss., Univ. Karlsruhe. Published in: Serie Math. Systems in Economics, vol 136. Anton Hain Verlag, Meisenheim
13. Schramm H, Zowe J (1992) A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, con-

vergence analysis, numerical results. SIAM J Optim 2:121–152
14. Stavroulakis GE, Goeleven D, Panagiotopoulos PD (1995) Stability of elastic bodies with nonmonotone multivalued boundary conditions of the quasidifferentiable type. J Elasticity 41(2):137–149
15. Womersley RS, Fletcher R (1986) An algorithm for composite nonsmooth optimization problems. J Optim Th Appl 48:493–523

Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives

GEORGIOS E. STAVROULAKIS
Carolo Wilhelmina Techn. University, Braunschweig, Germany

MSC2000: 49J52, 26E25, 52A27, 90C99

Article Outline

Keywords
Dini Derivatives
Clarke Derivatives
Quasidifferential and Clarke Subdifferential
Differences of Convex Sets
Estimation Results
See also
References

Keywords

Nonsmooth analysis; Clarke generalized gradient; Quasidifferentiability; Dini derivative

The notion of the *quasidifferential* in the sense of V.F. Demyanov and A.M. Rubinov [5] constitutes a set-valued extension of the classical differential, which is appropriate for nonsmooth and generally nonconvex but directionally differentiable functions. This class of functions covers a large number of applications in nonsmooth analysis and, among others, includes the popular in *global optimization* class of difference convex functions. The quasidifferential approximates the directional derivative of a function by using an ordered pair of convex sets, the subdifferential and the superdifferential. Definitions are given in ► Quasidifferentiable optimization. Information on the corresponding calculus can be found in ► Quasidifferentiable optimization: Calculus of quasidifferentials.

Here, the relation between quasidifferentials and more classical notions in nonsmooth analysis is briefly addressed. In particular, the Dini directional derivatives and the F.H. Clarke [2,3] derivatives are considered. Other notions of nonsmooth analysis may be found, among others, in the recent publications [1,3,11].

Dini Derivatives

The *Dini upper derivative* of a function $f\colon \mathbf{R}^n \to \overline{\mathbf{R}}$ at a point $x \in \operatorname{dom} f$ in a direction $g \in \mathbf{R}^n$ is defined by:

$$f_{\mathrm{D}}^{\uparrow} = \limsup_{\alpha \downarrow 0} \frac{1}{\alpha}[f(x, \alpha g) - f(x)] \,. \tag{1}$$

Note that the upper limit in (1) is not necessarily finite. Analogously, the *Dini lower derivative* of f at x is defined by the relation

$$f_{\mathrm{D}}^{\downarrow} = \liminf_{\alpha \downarrow 0} \frac{1}{\alpha}[f(x, \alpha g) - f(x)] \,.$$

Recall that if the limit

$$f'(x, g) = \lim_{\alpha \downarrow 0} \frac{1}{\alpha}[f(x + \alpha g) - f(x)]$$

exists it is called the *derivative* of a function f at a point x in a direction g, or the *Dini derivative* and it is denoted by ${f_D}'(x, g)$.

Since the Dini derivative (resp. the Dini upper or lower derivative) is just the one-sided (resp. the one-sided upper or lower) derivative of an ordinary real-valued function, one can uses the methods developed to study functions of one variable. Thus, for instance, calculus rules for directional derivatives can be constructed.

A function f defined on an open set Ω is called *Dini uniformly directionally differentiable* at a point $x \in \Omega$ if it is directionally differentiable at x and there exists a real number α_0 such that

$$\begin{aligned} &\frac{1}{\alpha}[f(x + \alpha g) - f(x) - \alpha f'(x, g)] < \epsilon \,, \\ &\forall \alpha \in (0, \alpha_0) \,, \quad \forall g \in S \,, \end{aligned} \tag{2}$$

where $S = \{g \,|g| = 1$ is the unit sphere. By setting $\alpha\, g = v$ in (2) one gets:

$$\begin{aligned} &\left|f(x + v) - f(x) - f'_x(v)\right| < \epsilon \,\|v\| \,, \\ &\forall v \text{ such that } \|v\| \le \alpha_0 \,. \end{aligned}$$

Thus the uniform directional differentiability means that $\frac{1}{\|v\|}\left|f(x+v)-f(x)-f'_x(v)\right|$ tends to zero, as $\|v\| \to 0$.

More details on Dini derivatives and their use in optimization can be found in [9].

Clarke Derivatives

Let us consider the upper and lower Dini derivatives of a function f for a fixed direction g, i. e. the functions $x \to f_D^{\uparrow}(x,g)$ and $x \to f_D^{\downarrow}(x,g)$. Let us also consider the upper (resp. the lower) regularizations of these functions:

$$\overline{f}_{\mathrm{D}}^{\uparrow}(x,g) = \max\left\{ f_{\mathrm{D}}^{\uparrow}(x,g), \limsup_{x' \to x} f_{\mathrm{D}}^{\uparrow}(x,g)\right\},$$

respectively

$$\overline{f}_{\mathrm{D}}^{\downarrow}(x,g) = \min\left\{ f_{\mathrm{D}}^{\downarrow}(x,g), \liminf_{x' \to x} f_{\mathrm{D}}^{\downarrow}(x,g)\right\}.$$

For a Lipschitz function f, the upper and lower Dini derivatives are bounded in some neighborhood of x, hence both previous limits are finite.

The Clarke upper and lower derivatives are defined as upper and lower regularizations of the Dini upper and lower derivatives, i. e.

$$f_{\mathrm{CL}}^{\uparrow}(x,g) = \overline{f}_{\mathrm{D}}^{\uparrow}(x,g),$$
$$f_{\mathrm{CL}}^{\downarrow}(x,g) = \overline{f}_{\mathrm{D}}^{\downarrow}(x,g).$$

For the initial, equivalent definition of these quantities, see [2,6 p. 69]. Here the approach of [8] has been followed. For every fixed direction g, the function $x \to f_{CL}^{\uparrow}(x,g)$ is upper semicontinuous and the function $x \to f_{CL}^{\downarrow}(x,g)$ is lower semicontinuous.

It is appropriate to recall here some properties of the Clarke derivatives. For every fixed point x, the function $g \to f_{CL}^{\uparrow}(x,g)$ is sublinear and the function $g \to f_{CL}^{\downarrow}(x,g)$ is superlinear, thus the subdifferential $\partial f_{CL}^{\uparrow}(x,g)$ and the superdifferential $\overline{\partial} f_{\mathrm{CL}}^{\downarrow}(x,g)$ can be determined, such that

$$f_{\mathrm{CL}}^{\uparrow}(x,g) = \max_{l \in \underline{\partial} f_{\mathrm{CL}}^{\uparrow}(x,g)} (l,g),$$
$$f_{\mathrm{CL}}^{\downarrow}(x,g) = \min_{w \in \overline{\partial} f_{\mathrm{CL}}^{\downarrow}(x,g)} (w,g).$$

Moreover, the following relations hold

$$f_{\mathrm{CL}}^{\uparrow}(x,-g) = (-f)_{\mathrm{CL}}^{\uparrow}(x,-g),$$
$$f_{\mathrm{CL}}^{\uparrow}(x,g) = -f_{\mathrm{CL}}^{\uparrow}(x,-g).$$

From the above properties it results that

$$\max_{l \in \underline{\partial} f_{\mathrm{CL}}^{\uparrow}(x,g)} (l,g) = \max_{w \in \overline{\partial} f_{\mathrm{CL}}^{\downarrow}(x,g)} (w,g);$$

thus the two compact convex sets coincide. The Clarke subdifferential is thus defined as

$$\partial_{\mathrm{CL}} f(x) = \underline{\partial} f_{\mathrm{CL}}^{\uparrow}(x,g) = \overline{\partial} f_{\mathrm{CL}}^{\downarrow}(x,g).$$

The mapping $x \to \partial_{\mathrm{CL}} f(x)$ is upper semicontinuous. An element of the Clarke subdifferential is called a *generalized gradient* of f at x.

Concerning the relation between the directional derivative of the function (if it is directionally differentiable) and the Clarke upper and lower derivatives one has, in general,

$$f_{\mathrm{CL}}^{\downarrow}(x,g) \le f'(x,g) \le f_{\mathrm{CL}}^{\uparrow}(x,g). \tag{3}$$

Thus Clarke upper and lower derivatives are a sublinear majorant and a superlinear minorant of $f'(x,g)$ respectively. Only in the case of an u.s.c. (resp. l.s.c.) directional derivative $f'(x,g)$ the second (resp. the first) inequality in (3) holds as an equality. The latter property is considered to be the major drawback of the Clarke subdifferential in nonsmooth analysis applications, because it does not always gives rise to an approximation of the directional derivative at the points of nondifferentiability.

For further reference we recall here the necessary optimality conditions for a locally Lipschitz function f at a point x:

$$0 \in \partial_{\mathrm{CL}} f(x).$$

Note also that since approximations of sets and functions are linked, the notion of Clarke subdifferential gives rise to a notion for the generalized tangent cone (and respectively a generalized normal cone). The reader is referred to [2,3] [6, p. 83], [10] for more details.

Quasidifferential and Clarke Subdifferential

Before giving some information on the links between the quasidifferential and the Clarke subdifferential, some elements on the several definitions of the difference of convex compact sets are in order.

Differences of Convex Sets

Before stating the definition, some introductory material must be given. The max-face of a compact set U generated by $x \in \mathbf{R}^n$ is defined by

$$G_x(U) = \left\{ h \in U \colon\ (h, x) = \max_{g \in U} (x, g) \right\} .$$

Recall that the max-face set coincides with the subdifferential of the *support function* of U, i. e. $G_x(U) = \underline{\partial}\, p_U(x)$. Recall also that for a convex function defined on $\mathbf{R}^n$ the set of points of $T \subset \mathbf{R}^n$ where max-face is a singleton is of measure zero is a set of full measure (with respect to $\mathbf{R}^n$, i. e. $\mathbf{R}^n \setminus T$ is a set of measure zero).

The difference of two sets U and V, $U \dot{-} V$ is defined on the set of full measure T where both $G_x(U)$ and $G_x(V)$ are singletons by:

$$U \dot{-} V = \operatorname{clco} \{ \nabla p_U(x) - \nabla p_V(x) \colon\ x \in T \} ,$$

where ∇g denotes the gradient of function g. One may observe here that if $U = V + W$ then $U \dot{-} V = W$.

An equivalent definition of $U \dot{-} V$ is given by

$$U \dot{-} V = \operatorname{clco} \left(\bigcup_{x \in T_{U,V}} [G_x(U) - G_x(V)] \right) , \qquad (4)$$

where the dependence of T on both U and V is explicitly indicated.

An extension of (4) leads to the quasidifference operation $\ddot{-}$, defined by

$$U \ddot{-} V = \operatorname{clco} \left(\bigcup_{x \neq 0} [G_x(U) - G_x(V)] \right) . \qquad (5)$$

Unfortunately $\ddot{-}$ is not invariant with respect to the equivalence relation $\approx$. Nevertheless an estimate of the form $U \ddot{-} V \supset U \dot{-} V$, for every sets U and V always holds and in some cases there exist conditions under which the inclusion holds as an equality (see e. g. [6, p. 117]).

Estimation Results

The Clarke subdifferential can also be generated, in some cases, by the set operators difference $\dot{-}$ and quasidifference $\ddot{-}$ applied on the subdifferential $\overline{\partial} f(x)$ and the superdifferential $\underline{\partial} f(x)$ of a quasidifferentiable function f(x).

Under appropriate assumptions on f, and for appropriate choice of the elements of the subdifferential and the subdifferential of f at x, an estimate of the following form can be extracted:

$$A \subset \partial_{\mathrm{CL}} f(x) \subset B.$$

with set $A = \underline{\partial} f(x) \dot{-} (-\overline{\partial} f(x))$ and set $B = \underline{\partial} f(x) \ddot{-} (-\overline{\partial} f(x))$, as it is discussed in [6, pp. 143–155]. A different approach to the study of the relationship between the Clarke subdifferential and the quasidifferential is followed in [7] (see also [6, pp. 156–159]). More details in this direction can also be found in [4].

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-Nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems

- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Aubin P, Frankowska H (1991) Set–valued analysis. Birkhäuser, Basel
2. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley/Interscience, New York. Reprinted as: Classics Appl. Math., vol 5. SIAM, 1990
3. Clarke FH, Ledyaev SYu, Stern RJ, Wolenski PR (1998) Nonsmooth analysis and control theory. Springer, Berlin
4. Demyanov VF, Jeyakumar V (1997) Hunting for a smaller convex subdifferential. J Global Optim 13(10):305–326
5. Demyanov VF, Rubinov AM (1980) On quasidifferentiable functionals. Soviet Math Dokl 21:14–17
6. Demyanov VF, Rubinov AM (1995) Introduction to constructive nonsmooth analysis. P. Lang, Frankfurt am Main
7. Demyanov VF, Sutti C (1995) Representation of the Clarke subdifferential for a regular quasidifferentiable function. J Optim Th Appl 87(3):553–562
8. Demyanov VF, Vasiliev LN (1985) Nondifferentiable optimization. Optim. Software, New York
9. Ioffe AD (1984) Calculus of Dini subdifferentials of functions and contingent coderivatives of set-valued maps. Nonlinear Anal Th Methods Appl 8(5):517–539
10. Panagiotopoulos PD (1985) Inequality problems in mechanics and applications. Convex and nonconvex energy functions. Birkhäuser, Basel
11. Rockafellar RT, WetsRJ-B (1998) Variational analysis. Springer, Berlin

Quasidifferentiable Optimization: Exact Penalty Methods

LYUDMILA N. POLYAKOVA
St. Petersburg State University, St. Petersburg, Russia

MSC2000: 90Cxx

Article Outline

Keywords
Regularity Condition 1
Regularity Condition 2
Regularity Condition 3
Regularity Condition 4
See also
References

Keywords

Quasidifferential functions; Quasidifferential; Exact penalty function; Exact penalty parameter; Regularity

Penalty function methods are used for solving many constrained optimization problems of the form: Find

$$\inf_{x\in X} f(x) = f^* \,, \tag{1}$$

where f is a locally Lipschitz quasidifferentiable function defined on $\mathbf{R}^n$, $\mathcal{D}f(x) = [\overline{\partial} f(x), \underline{\partial} f(x)]$ is its quasidifferential at a point $x \in \mathbf{R}^n$, $X \subset \mathbf{R}^n$ is a closed set.

It is always possible to define X in the form (see [2])

$$X = \{x \in \mathbf{R}^n : \ \varphi(x) = 0\} \,, \tag{2}$$

where φ is also a locally Lipschitz quasidifferentiable function defined on $\mathbf{R}^n$, the pair of sets $\mathcal{D}\varphi(x) = [\overline{\partial}\varphi(x), \underline{\partial}\varphi(x)]$ is a quasidifferential of φ at $x \in \mathbf{R}^n$ and

$$\varphi(x) > 0 \,, \quad \forall x \notin X \,.$$

Thus the set X is the set of global minimum points of the function φ on $\mathbf{R}^n$. Hence, it is closed. We shall assume that the set $X \subset \mathbf{R}^n$ is not empty and bounded.

As the function φ is quasidifferentiable then the following expansion holds:

$$\varphi(x + \alpha g) = \varphi(x) + \alpha\varphi'(x, g) + o(\alpha, x, g) \,,$$

where

$$\frac{o(\alpha, x, g)}{\alpha} \overset{\alpha\downarrow 0}{\to} 0 \,.$$

We shall assume that in this expansion at each point $x \in \mathbf{R}^n$ the convergence to 0 is uniform with respect to $g \in \mathbf{R}^n$, $\|g\| = 1$.

The idea of penalty function methods consists in reducing the problem (1) to a problem of the unconstrained optimization. Among the different approaches existing for such reduction we shall consider the method of exact penalty functions.

For solving the problem (1) a penalty function

$$F(c,x) = f(x) + c\varphi(x)$$

is introduced, where c is a nonnegative number, and then the problem

$$\inf_{x\in\mathbf{R}^n} F(c,x) \tag{3}$$

is considered.

We assume that $\inf_x \in \mathbf{R}^n F(c,x)$ is attained for every $c \geq 0$. In practice it would be useful to find conditions which guarantee that there exists an exact penalty parameter $c^* \geq 0$ such that the set

$$\left\{ x \in \mathbf{R}^n : \; x = \arg\min_{x\in\mathbf{R}^n} F(c^*,x) \right\}$$

coincides with the set

$$\left\{ x \in \mathbf{R}^n : \; x = \arg\min_{x\in X} f(x) \right\} .$$

At first such a problem was investigated in [5,10] for the problem of convex programming. Now there are many works in this field of mathematics. (See, for example, [1,4,6,8,9]).

The implementation of exact penalty function methods first of all depends on the properties of the function φ. Therefore various conditions are imposed on φ to make it possible to solve our problem. We shall consider some of them.

Regularity Condition 1

(See [3].)

We say that a *regularity condition* is satisfied for the function φ if for any boundary point $x^* \in \mathrm{bd}X$ there exist positive real numbers $\varepsilon(x^*)$ and $\beta(x^*)$, such that

$$\frac{o(\alpha,x,g)}{\alpha} > -\varphi'(x,g) + \beta(x^*)$$
$$= -\left[\max_{v\in\underline{\partial}\varphi(x)} \langle v,g\rangle + \min_{w\in\overline{\partial}\varphi(x)} \langle w,g\rangle \right] + \beta(x^*) ,$$
$$\forall x \in \mathcal{A}(X) \cap S_{\varepsilon(x^*)}(x^*) ,$$
$$\forall \alpha \in (0, \varepsilon(x^*)] ,$$
$$\forall g \in N(X,x) : \; \|g\| = 1 ,$$

where bd X is the set of boundary points of X,

$$S_{\varepsilon(x^*)}(x^*) = \{ x \in \mathbf{R}^n : \; \|x - x^*\| \leq \varepsilon(x^*) \} ,$$

$$\mathcal{A}(X) = \left\{ x \in \mathrm{bd}\, X : \begin{array}{c} \exists z \notin X : \\ x \text{ is a projection of } z \end{array} \right\} ,$$

$N(X,x)$ is the *normal cone* to the set X at the point $x \in X$:

$$N(X,x) = \left\{ g \in \mathbf{R}^n : \begin{array}{c} \langle g_1, g\rangle \leq 0, \\ \forall g_1 \in \Gamma(X,x) \end{array} \right\}$$

and

$$\Gamma(X,x) = \left\{ g \in \mathbf{R}^n : \begin{array}{c} \exists g_k \in \mathbf{R}^n, \alpha_k \geq 0, \\ g_k \to g, \; \alpha_k \downarrow 0, \\ x + \alpha_k g_k \in X \end{array} \right\} .$$

The regularity condition 1 is a condition about the behavior of the function φ only at the boundary points of the set X.

If for the function φ the regularity condition 1 holds, then there exists an exact penalty parameter c^*.

Since in practice the exact penalty parameter is a priori unknown, a sequence of real values c_k is constructed, satisfying the conditions

$$0 = c_0 < \cdots < c_k < \cdots ,$$
$$\lim_{k\to+\infty} c_k = +\infty .$$

Let us find

$$x(\cdot_k) = \arg\min_{x\in\mathbf{R}^n} F(\cdot_k, x) .$$

As a result, a decreasing sequence of real values $\{\varphi(x(c_k))\}$ is constructed. There exists an integer $K > 0$ such that $x(c_k) \in X$, $\forall k > K$. Thus, for $k > K$, the points $x(c_k)$ will be global minimum points of a function $F(c_k, x)$ on $\mathbf{R}^n$, i. e. will be solutions of problem (1). The value of the penalty parameter c^* is directly proportional to the Lipschitz constant of the function f on the set

$$\mathcal{L}(x^{**}) = \{ x \in \mathbf{R}^n : \; \varphi(x) \leq \varphi(x^{**}) \} ,$$

where

$$x^{**} = \min_{x\in\mathbf{R}^n} f(x) ,$$

and inversely proportional to the number $\beta(x^*)$, where the point x^* is a limit point of the sequence $\{x(c_k)\}$. In this method the regularity condition 1 is used only in a neighborhood of the point x^*.

Note that the function φ is essentially nondifferentiable at the boundary points of the set X.

For differentiable functions at the boundary points of the set X, the regularity condition 1 does not hold. For this reason, if the function φ at the boundary points of the set X is superdifferentiable, then it cannot be used for constructing a sequence of exact penalty functions $F(c, x)$.

An example of a function, which can be used for constructing a family of exact penalty functions (even not requiring the set X to be bounded) is the Euclidean distance function. For it as an exact penalty parameter it is possible to take the Lipschitz constant of the function f on the set $\mathcal{L}(x^{**})$. However this function is not suitable for practical use due to computational problems.

We shall notice, that the regularity condition 1 is not constructive. Sometimes instead of it one uses another regularity condition.

Regularity Condition 2

(See [3].)

We shall assume that there exists a real number $\beta > 0$, such that the following inequality holds

$$\inf_{x\in\mathcal{A}(X)} \min_{\substack{\|g\|=1,\\ g\in N(X,x)}} \varphi'(x,g) = \inf_{x\in\mathcal{A}(X)} \min_{\substack{\|g\|=1,\\ g\in N(X,x)}} \left[\max_{v\in\underline{\partial}\varphi(x)} \langle v, g\rangle + \min_{w\in\overline{\partial}\varphi(x)} \langle w, g\rangle\right] \geq \beta\,. \tag{4}$$

If the set X is bounded and does not consist of isolated points, and the regularity condition 2 is fulfilled for the family of penalty functions $\{F(c_k, \mathrm{x})\}$ then there exists an exact penalty parameter.

Under some assumptions on the set X it is possible to get an analytical representation of the normal cone for this set at each boundary point and then, having calculated the constant β, it is possible to evaluate the exact penalty parameter c^*.

We shall assume, that for the function φ at each point $x \in \mathrm{bd}\, X$ the regularity condition 2 holds. Then the representation of the normal cone $N(X, x)$ to the given set X at the point $x \in \mathrm{bd}\, X$,

$$N(X,x) = \bigcap_{w\in\overline{\partial}\varphi(x)} \mathrm{cl\,cone}(\underline{\partial}\varphi(x) + w)\,,$$

holds. Here, $\mathrm{cl}\, A$ is the closure of A, $\mathrm{cone}\, A$ is the conical hull of A.

For example, if the function φ is subdifferentiable at each boundary point of the set X, then

$$N(X,x) = \mathrm{cl\,cone}(\underline{\partial}\varphi(x))\,,$$

and (4) can be rewritten as

$$\inf_{x\in\mathcal{A}(X)} \min_{\substack{\|g\|=1,\\ g\in\mathrm{cl\,cone}(\underline{\partial}\varphi(x))}} \max_{v\in\underline{\partial}\varphi(x)} \langle v, g\rangle \geq \beta > 0\,. \tag{5}$$

Example 1 Let

$$f_0(x) = \frac{1}{2}\langle A_0 x, x\rangle + \langle b_0, x\rangle\,, \quad x, b_0 \in \mathbf{R}^n\,,$$
$$f_i(x) = \frac{1}{2}\langle A_i x, x\rangle + \langle b_i, x\rangle + c_i\,,$$
$$x, b_i \in \mathbf{R}^n, \quad c_i \in \mathbf{R}^1, \quad i \in I = 1, \ldots, p\,,$$

where f_i, $i \in (0, \ldots, p)$ are strongly convex functions. All the matrices A_i, $i = 0, \ldots, p$, are positive definite.

Consider the problem

$$\min_{x\in X} f_0(x)\,, \tag{6}$$

where

$$X = \{x \in \mathbf{R}^n :\ f_i(x) \leq 0,\ i \in I\}\,.$$

Let

$$\varphi(x) = \max\{0, \varphi_1(x)\}\,,$$
$$\varphi_1(x) = \max_{i\in I} f_i(x)\,,$$
$$x \in \mathbf{R}^n\,.$$

Then

$$X = \{x \in \mathbf{R}^n :\ \varphi(x) = 0\}\,.$$

Let

$$\overline{x} = \arg\min_{x\in\mathbf{R}^n} \varphi_1(x)\,.$$

We shall assume that $\varphi_1(\overline{x}) < 0$.

Problem (6) is a convex programming problem. For this problem the regularity condition 1 is valid.

Let d be the radius of the maximal ball centered at the point $\overline{x}$ and inscribed into the set X. Then the number $c^* = L/2\ md$ is an exact penalty parameter for the

problem (6). In this equality L is a Lipschitz constant of the function f_0 on the set

$$X_1 = \{x \in \mathbf{R}^n : \varphi(x) \le \varphi(x^{**})\} ,$$
$$x^{**} = \arg\min_{x \in \mathbf{R}^n} f_0(x) ,$$

i. e. the number

$$L = \max_{x \in X_1} \|A_0 x + b_0\|$$

can be taken as a Lipschitz constant of f on X_1, m is a strong convexity constant of the function φ_1 $m = \min_{i \in I} \overline{m}_i$, where $\overline{m}_i$ is constant of strong convexity of the function f_i, $i \in I$).

Example 2 Consider the optimization problem

$$\min_{x \in X} f(x) , \tag{7}$$

where

$$f(x) = f_1(x) - f_2(x) ,$$
$$X = \{x \in \mathbf{R}^n : \varphi_1(x) - \varphi_2(x) \le 0\} ,$$

$f_1, f_2, \varphi_1, \varphi_2$ are convex functions defined on $\mathbf{R}^n$.

Let the set X be bounded. It can be defined in the form

$$X = \{x \in \mathbf{R}^n : \varphi(x) = 0\} ,$$

where $\varphi(x) = \max\{0, \varphi_1(x) - \varphi_2(x)\}$.

The function φ can be represented as the difference of convex (d.c.) functions

$$\varphi(x) = \max\{\varphi_1(x), \varphi_2(x)\} - \varphi_2(x) .$$

We shall consider only points $x \in X$ where $\varphi_1(x) = \varphi_2(x)$. Then the pair of convex sets

$$\mathcal{D}\varphi(x) = \left[\operatorname{co}\{\partial\varphi_1(x), \partial\varphi_2(x)\}, -\partial\varphi_2(x)\right]$$

is a quasidifferential of the function φ at a point x, where $\partial\varphi_i(x)$, $i = 1, 2$, is the subdifferential of the convex function φ_i at x in the sense of convex analysis. Here co A is the convex hull of A.

If the regularity condition 2 is valid for the function φ, i. e. there exists a real value $\beta > 0$ such that

$$\inf_{x \in \mathcal{A}(X)} \min_{\substack{\|g\| = 1 \\ g \in N(X, x)}} \left\{ \max_{v \in \operatorname{co}\{\partial\varphi_1(x), \partial\varphi_2(x)\}} \langle v, g\rangle - \max_{w \in \partial\varphi_2(x)} \langle w, g\rangle \right\} \ge \beta ,$$

then there exists an exact penalty parameter c^* for the sequence of penalty functions

$$F(c_k, x) = f(x) + c_k\varphi(x) = \left(f_1(x) + c_k \max\{\varphi_1(x), \varphi_2(x)\}\right) - \left(f_2(x) + c_k\varphi_2(x)\right) .$$

Let the set X be defined as

$$X = \{x \in \mathbf{R}^n : \varphi_1(x) - \varphi_2(x) = 0\} ;$$

then it can be rewritten as

$$X = \{x \in \mathbf{R}^n : \varphi(x) = 0\} ,$$

where $\varphi(x) = \max\{0, |;\varphi_1(x) - \varphi_2(x)|\}$.

In this case the function φ can be represented as the difference of convex functions, namely

$$\varphi(x) = \max\{2\varphi_1(x), 2\varphi_2(x), \varphi_1(x) + \varphi_2(x)\} - (\varphi_1(x) + \varphi_2(x)) .$$

If the regularity condition 2 is valid for the function φ, i. e. there exists a real value $\beta > 0$ such that

$$\inf_{x \in \mathcal{A}(X)} \min_{\substack{\|g\| = 1 \\ g \in N(X, x)}} \left\{ \max_{v \in \operatorname{co}\{2\partial\varphi_1(x), 2\partial\varphi_2(x), \partial\varphi_1(x) + \varphi_2(x)\}} \langle v, g\rangle - \max_{w \in \partial[\varphi_1(x) + \varphi_2(x)]} \langle w, g\rangle \right\} \ge \beta,$$

then there exists an exact penalty parameter c^* for the sequence of penalty functions

$$F(c_k, x) = f(x) + c_k\varphi(x) = f_1(x) + c_k \max\{2\varphi_1(x), 2\varphi_2(x), \varphi_1(x) + \varphi_2(x)\} - \left(f_2(x) + c_k\varphi_1(x) + c_k\varphi_2(x)\right) .$$

Thus the solution of the problem (7) can be obtained as the result of unconstrained optimization of d.c.functions.

V.F. Demyanov [2] considers the following condition for constructing a family of exact penalty functions. Put

$$d(x) = \min_{\|g\| = 1} \varphi'(x, g) = \min_{\|g\| = 1} \left[\max_{v \in \underline{\partial}\varphi(x)} \langle v, g\rangle + \min_{w \in \overline{\partial}\varphi(x)} \langle w, g\rangle \right] ,$$
$$\Psi_X(x) = \limsup_{x' \to x, x' \notin X} d(x') .$$

Regularity Condition 3

If for some $\varepsilon > 0$ the set

$$X_\varepsilon = \{x \in \mathbf{R}^n : \varphi(x) \le \varepsilon\}$$

is bounded and

$$\Psi_X(x) < 0\,, \quad \forall x \in \mathrm{bd}\, X\,,$$

then for the family of penalty functions $F(c_k, x)$ there exists an exact penalty parameter $c^* < \infty$.

To use the regularity condition 3 it is necessary to know the behavior of the function φ in the neighborhood of the set X.

Sometimes the following regularity condition is used (see [7]).

Regularity Condition 4

(Condition of *ρ-regularity*). We say that the problem (1) with the set X is *ρ-regular* if there exists a positive number β such that the inequality

$$\varphi(x) \ge \beta \rho_X(x)\,, \quad \forall x \in \mathbf{R}^n \backslash X\,,$$

holds, where ρ_X is the Euclidean distance function.

It is not difficult to observe that the regularity condition 4 is not constructive. In [7] the existence of an exact penalty parameter for a family of penalty functions is proved for problems of nonlinear programming if the condition of ρ-regularity is satisfied.

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-Nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Charalambous C (1978) A lower bound for the controlling parameters of the exact penalty functions. Math Program 15:278–290
2. Demyanov VF, Rubinov AM (1995) Constructive nonsmooth analysis. P. Lang, Frankfurt am Main
3. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
4. Di Pillo G, Facchinei F (1992) Regularity conditions and exact penalty functions in Lipschitz programming problems. Nonsmooth optimization metods and applications (Amsterdam, 1992). In: Giannessi F (ed). Gordon and Breach, New York, pp 107–120
5. Eremin II (1967) Method of 'penalties' in convex programming. Dokl USSR Acad Sci 173(4):748–751
6. Evtushenko Y, Zhadan V (1990) Exact auxiliary functions. Informatica 1(1):40–55
7. Fedorov VV (1979) Numerical methods of a maximin. Nauka, Moscow

8. Han S, Mangasarian O (1979) Exact penalty functions in nonlinear programming. Math Program 17:251–269
9. Pietrzykowski T (1969) An exact potential method for constrained maxima. SIAM J Numer Anal 16:299–304
10. Zangwill W (1967) Non-linear programming via penalty functions. Managem Sci 13(5):344–358

Quasidifferentiable Optimization: Optimality Conditions

VLADIMIR F. DEMYANOV
St. Petersburg State University, St. Petersburg, Russia

MSC2000: 90Cxx

Article Outline

Keywords
Directionally Differentiable Functions
Quasidifferentiable Functions
  Examples of q.d. Functions
Calculus of Quasidifferentials
Necessary and Sufficient Conditions for an Unconstrained Optimum
Directions of Steepest Descent and Ascent
Necessary and Sufficient Conditions for a Constrained Optimum
See also
References

Keywords

Quasidifferential; Quasidifferential calculus; Necessary and sufficient conditions; Directional derivative

Directionally Differentiable Functions

Let f be a real-valued function defined on an open set $X \subset \mathbf{R}^n$, $x \in X$. The function f is called *Dini differentiable* at the point x in a direction $g \in \mathbf{R}^n$ if there exists the finite limit

$$f'_{\mathrm{D}}(x,g) = \lim_{\alpha \downarrow 0} \frac{1}{\alpha}[f(x+\alpha g) - f(x)]\,. \tag{1}$$

Here $\alpha \downarrow 0$ means that $\alpha \to 0, \alpha > 0$. The quantity $f'_{\mathrm{D}}(x, g)$ is called the *Dini derivative* of f at x in a direction g.

The function f is called *Hadamard differentiable* at the point x in a direction $g \in \mathbf{R}^n$ if there exists the finite limit

$$f'_{\mathrm{H}}(x,g) = \lim_{[\alpha, g'] \to [+0, g]} \frac{1}{\alpha}[f(x+\alpha g') - f(x)]\,. \tag{2}$$

Clearly, if f is Hadamard differentiable at x in a direction g then it is Dini differentiable as well and

$$f'_{\mathrm{H}}(x,g) = f'_{\mathrm{D}}(x,g)\,. \tag{3}$$

If the limit in (1) exists and is finite for every $g \in \mathbf{R}^n$ then the function f is called *Dini directionally differentiable* (D-d.d) at x. The quantity $f'_{\mathrm{D}}(x, g)$ is called the *Hadamard derivative* of f at x in a direction g.

If the limit in (2) exists and is finite for every $g \in \mathbf{R}^n$ then the function f is called the *Hadamard directionally differentiable* (H-d.d) at x. Of course, every H-d.d. function at x is D-d.d., the converse is not necessarily true.

The directional (and generalized directional) derivatives may be used to describe optimality conditions (see ▶ Dini and Hadamard derivatives in optimization). However, using properties of special classes of functions one can expect to get more 'constructive' conditions. One of such classes is the family of quasidifferentiable functions.

Quasidifferentiable Functions

Let f be a real-valued function defined on an open set $X \subset \mathbf{R}^n$, $x \in X$. The function f is called *Dini* (*Hadamard*) *quasidifferentiable* (q.d) at x if it is Dini (Hadamard) directionally differentiable at x and if its directional derivative $f'_{\mathrm{D}}(x, g)$ ($f'_{\mathrm{H}}(x, g)$) can be represented in the form

$$f'_{\mathrm{D}}(x,g) = \max_{v \in \underline{\partial} f_{\mathrm{D}}(x)} (v,g) + \min_{w \in \overline{\partial} f_{\mathrm{D}}(x)} (w,g),$$

$$\left(f'_{\mathrm{H}}(x,g) = \max_{v \in \underline{\partial} f_{\mathrm{H}}(x)} (v,g) + \min_{w \in \overline{\partial} f_{\mathrm{H}}(x)} (w,g) \right),$$

where the sets $\underline{\partial} f_{\mathrm{D}}(x), \overline{\partial} f_{\mathrm{D}}(x), \underline{\partial} f_{\mathrm{H}}(x), \overline{\partial} f_{\mathrm{H}}(x)$ are convex compact sets of $\mathbf{R}^n$. The pair

$$\mathrm{D} f_{\mathrm{D}}(x) = [\underline{\partial} f_{\mathrm{D}}(x), \overline{\partial} f_{\mathrm{D}}(x)]\,,$$

$$\left(\mathrm{D} f_{\mathrm{H}}(x) = [\underline{\partial} f_{\mathrm{H}}(x), \overline{\partial} f_{\mathrm{H}}(x)] \right)$$

is called a *Dini* (*Hadamard*) *quasidifferential* of f at x. Most of the results stated below are valid for both Dini

and Hadamard q.d. functions, therefore we shall use the notation $Df(x) = [\underline{\partial} f(x), \overline{\partial} f(x)$ for both $D_D f(x)$ and $D_H f(x)$ and the pair $Df(x)$ will just be called a *quasidifferential* of f at x. Analogously, the notation $f'(x, g)$ is used for both $f'_D(x, g)$ and $f'_H(x, g)$.

The directional derivative $f'(x, g)$ is positively homogeneous (in g) of degree one:

$$f'(x, \lambda g) = \lambda f'(x, g)\,, \quad \forall \lambda > 0\,. \tag{4}$$

Note that Hadamard quasidifferentiability implies Dini quasidifferentiability, the converse not necessarily being true.

Thus for a quasidifferentiable (q.d) function

$$f'(x, g) = \max_{v \in \underline{\partial} f(x)} (v, g) + \min_{w \in \overline{\partial} f(x)} (w, g)\,, \quad \forall g \in \mathbf{R}^n. \tag{5}$$

The set $\underline{\partial} f(x)$ is called a *subdifferential* of f at x, and the set $\overline{\partial} f(x)$ is called a *superdifferential* of f at x. Note that a quasidifferential is not uniquely defined: If a pair $D = [A, B]$ is a quasidifferential of f at x then, e. g., for any convex compact set $C \subset \mathbf{R}^n$ the pair $D_1 = [A + C, B - C]$ is a quasidifferential of f at x (since, by (5), both these pairs produce the same function $f'(x, g)$). The equivalence class of pairs of convex compact sets $Df(x) = [\underline{\partial} f(x), \overline{\partial} f(x)$ producing the function $f'(x, g)$ by formula (5) is called the *quasidifferential* of f at x (we shall use the same notation $Df(x)$ for the quasidifferential of f at x).

If there exists a quasidifferential $Df(x)$ of the form $Df(x) = [\underline{\partial} f(x), \{0_n\}]$ then f is called *subdifferentiable* at x. If there exists a quasidifferential $Df(x)$ of the form $Df(x) = [\{0_n\}, \overline{\partial} f(x)]$ then f is called *superdifferentiable* at x. Here $0_n = (0, \ldots, 0) \in \mathbf{R}^n$.

Examples of q.d. Functions

1) If f is a smooth function on X then

$$f'(x, g) = (f'(x), g)\,, \tag{6}$$

where $f'(x)$ is the gradient of f at x. It is clear that

$$f'(x, g) = \max_{v \in \underline{\partial} f(x)} (v, g) + \min_{w \in \overline{\partial} f(x)} (w, g)\,, \tag{7}$$

with

$$\underline{\partial} f(x) = \{f'(x)\}\,, \quad \overline{\partial} f(x) = \{0_n\}\,.$$

Hence, f is Hadamard quasidifferentiable and even subdifferentiable. Since in (7) one can also take

$$\underline{\partial} f(x) = \{0_n\}\,, \quad \overline{\partial} f(x) = \{f'(x)\}\,,$$

then f is superdifferentiable as well.

2) If f is a convex function on a convex open set $X \subset \mathbf{R}^n$ then (as it is known from convex analysis) f is H-d.d. on X and

$$f'(x, g) = \max_{v \in \partial f(x)} (v, g)\,,$$

where $\partial f(x)$ is the subdifferential of f (in the sense of convex analysis):

$$\partial f(x) = \left\{ v \in \mathbf{R}^n : \begin{array}{c} f(z) - f(x) \geq (v, z - x) \\ \forall z \in X \end{array} \right\}.$$

Therefore f is Hadamard quasidifferentiable and one can take the pair $Df(x) = [\,\partial f(x), \{0_n\}]$ as its quasidifferential. Thus, f is even subdifferentiable.

3) If f is concave on a convex set X then f is H-d.d. and

$$f'(x, g) = \min_{w \in \overline{\partial} f(x)} (w, g),$$

where

$$\overline{\partial} f(x) = \left\{ w \in \mathbf{R}^n : \begin{array}{c} f(z) - f(x) \leq (w, z - x) \\ \forall z \in X \end{array} \right\}.$$

Hence, f is Hadamard quasidifferentiable and one can take the pair

$$Df(x) = [\{0_n\}, \quad \overline{\partial} f(x)]$$

as its quasidifferential. Thus, f is even superdifferentiable.

Calculus of Quasidifferentials

The family of q.d. functions enjoys a well-developed calculus: First let us define the operation of addition of two pairs of compact convex sets and the operation of multiplication of a pair by a real number.

If $D_1 = [A_1, B_1]$, $D_2 = [A_2, B_2]$ are pairs of convex compact sets in $\mathbf{R}^n$ then

$$D_1 + D_2 = [A, B]$$

with

$$A = A_1 + A_2\,, \quad B = B_1 + B_2\,.$$

If D = [A, B] where A and B are convex compact sets, $\lambda \in \mathbf{R}$ then

$$\lambda D = \begin{cases} [\lambda A, \lambda B], & \lambda \geq 0, \\ [\lambda B, \lambda A], & \lambda < 0. \end{cases}$$

Let $X \subset \mathbf{R}^n$ be an open set.

Proposition 1 ([1, Chap. III])

1) If functions $f_1, \ldots, f_N$ are quasidifferentiable at a point $x \in X$, and $\lambda_1, \ldots, \lambda_N$ are real numbers then the function

$$f = \sum_{i=1}^{N} \lambda_i f_i$$

is also quasidifferentiable at x with a quasidifferential $Df(x) = [\underline{\partial} f(x), \overline{\partial} f(x)]$ where

$$Df(x) = \sum_{i=1}^{N} \lambda_i Df_i(x), \tag{8}$$

$Df_i(x)$ being a quasidifferential of f_i at x.

2) If f_1 and f_2 are quasidifferentiable at a point $x \in X$ then the function $f = f_1 \cdot f_2$ is also q.d. at x and

$$Df(x) = f_1(x)Df_2(x) + f_2(x)Df_1(x). \tag{9}$$

3) If f_1 and f_2 are quasidifferentiable at a point $x \in X$ and $f_2(x) \neq 0$ then the function $f = \{f_1 f_2\}$ is also q.d. at x and

$$Df(x) = \frac{1}{f_2^2(x)} \left[f_2(x)Df_1(x) - f_1(x)Df_2(x) \right]. \tag{10}$$

4) Let functions $f_1, \ldots, f_N$ be quasidifferentiable at a point $x \in X$. Construct the functions

$$\varphi_1(x) = \max_{i \in 1,\ldots,N} f_i(x),$$
$$\varphi_2(x) = \min_{i \in 1,\ldots,N} f_i(x).$$

Then the functions φ_1 and φ_2 are q.d. at x and

$$D\varphi_1(x) = [\underline{\partial}\varphi_1(x), \overline{\partial}\varphi_1(x)],$$
$$D\varphi_2(x) = [\underline{\partial}\varphi_2(x), \overline{\partial}\varphi_2(x)], \tag{11}$$

where

$$\underline{\partial}\varphi_1(x) = \operatorname{co}\left\{ \underline{\partial} f_k(x) - \sum_{\substack{i \in R(x) \\ i \neq k}} \overline{\partial} f_i(x) \colon\ k \in R(x) \right\},$$

$$\overline{\partial}\varphi_1(x) = \sum_{k \in R(x)} \overline{\partial} f_k,$$

$$\underline{\partial}\varphi_2(x) = \sum_{k \in Q(x)} \underline{\partial} f_k,$$

$$\overline{\partial}\varphi_2(x) = \operatorname{co}\left\{ \overline{\partial} f_k(x) - \sum_{\substack{i \in Q(x) \\ i \neq k}} \underline{\partial} f_i(x) \colon\ k \in Q(x) \right\}.$$

Here $[\underline{\partial} f_k(x), \overline{\partial} f_k(x)]$ is a quasidifferential of the function f_k at the point x,

$$R(x) = \{i \in 1, \ldots, N \colon\ f_i(x) = \varphi_1(x)\},$$
$$Q(x) = \{i \in 1, \ldots, N \colon\ f_i(x) = \varphi_2(x)\}.$$

The following composition theorem holds.

Proposition 2 [1, Chap. III]) *Let X be an open set in $\mathbf{R}^n$, Y be an open set in $\mathbf{R}^m$ and let a mapping $H(x) = (h_1(x), \ldots, h_m(x))$ be defined on X, take its values in Y and its coordinate functions h_i be quasidifferentiable at a point $x_0 \in X$. Assume also that a function f is defined on Y and is Hadamard quasidifferentiable at the point $y_0 = H(x_0)$. Then the function*

$$\varphi(x) = f(H(x))$$

is quasidifferentiable at the point x_0.

The corresponding formula for the quasidifferential of φ at x_0 is presented in [Thm. III.2.3]

Remark 3 Thus, the family of quasidifferentiable functions is a linear space closed with respect to all 'smooth' operations and, what is most important, the operations of taking the pointwise maximum and minimum. Formulas (8)–(10) are just generalizations of the rules of classical differential calculus. Most problems and results of classical differential calculus may be formulated for nonsmooth functions in terms of quasidifferentials (see, e. g., [1,3]). For example, a mean value theorem is valid [5].

Necessary and Sufficient Conditions for an Unconstrained Optimum

The following results hold due to the properties of directionally differentiable functions.

Let $X \subset \mathbf{R}^n$ be an open set, f be a real-valued function defined and directionally differentiable on X.

Proposition 4 *For a point $x* \in X$ to be a local or global minimizer of f on X it is necessary that*

$$f'(x^*, g) \geq 0\,, \quad \forall g \in \mathbf{R}^n\,. \tag{12}$$

If f is Hadamard d.d. at $x*$ and

$$f'_H(x^*, g) > 0\,, \quad \forall g \in \mathbf{R}^n\,,\ g \neq 0_n\,, \tag{13}$$

then $x*$ is a strict local minimizer of f.

For a point $x** \in X$ to be a local or global maximizer of f on X it is necessary that

$$f'(x^{**}, g) \leq 0\,, \quad \forall g \in \mathbf{R}^n\,. \tag{14}$$

If f is Hadamard d.d. at $x**$ and

$$f'_H(x^{**}, g) < 0\,, \quad \forall g \in \mathbf{R}^n\,,\ g \neq 0_n\,, \tag{15}$$

then $x**$ is a strict local maximizer of f.

These conditions may be restated in terms of quasidifferentials. Let f be quasidifferentiable on an open set $X \subset \mathbf{R}^n$.

Proposition 5 (see [1,3,5]) *For a point $x* \in X$ to be a local or global minimizer of f on X it is necessary that*

$$-\overline{\partial} f(x^*) \subset \underline{\partial} f(x^*)\,. \tag{16}$$

If f is Hadamard quasidifferentiable at $x*$ and

$$-\overline{\partial} f(x^*) \subset \operatorname{int} \underline{\partial} f(x^*), \tag{17}$$

then $x*$ is a strict local minimizer of f.

For a point $x^{**} \in X$ to be a local or global maximizer of f on X it is necessary that

$$-\underline{\partial} f(x^{**}) \subset \overline{\partial} f(x^{**})\,. \tag{18}$$

If f is Hadamard quasidifferentiable at x^{**} and

$$-\underline{\partial} f(x^{**}) \subset \operatorname{int} \overline{\partial} f(x^{**})\,, \tag{19}$$

then x^{**} is a strict local maximizer of f.

Remark 6 The quasidifferential represents a generalization of the notion of gradient to the nonsmooth case and therefore conditions (16)–(19) are first order optimality conditions.

In the smooth case one can take $\mathrm{D}f(x) = [\underline{\partial} f(x), \overline{\partial} f(x)]$ where $\underline{\partial} f(x) = \{f'(x)\}$, $\overline{\partial} f(x) = \{0_n\}$; therefore condition (16) is equivalent to

$$f'(x^*) = 0_n\,, \tag{20}$$

condition (18) is equivalent to

$$f'(x^{**}) = 0_n\,, \tag{21}$$

and, since both sets $\underline{\partial} f$ and $\overline{\partial} f$ are singletons, the conditions (17) and (19) are impossible. Thus, conditions (17) and (19) are essentially 'nonsmooth'.

A point $x^* \in X$ satisfying (16) is called an inf-stationary point, a point $x^{**} \in X$ satisfying (18) is called a sup-stationary point of f. In the smooth case the necessary condition for a minimum (20) is the same as the necessary condition for a maximum (21).

Directions of Steepest Descent and Ascent

Let $x \in X$ be not an inf-stationary point of f (i. e. condition (16) is not satisfied). Take $w \in \overline{\partial} f(x)$ and find

$$\min_{v \in \underline{\partial} f(x)} \|v + w\| = \|v(w) + w\| = \rho_1(w).$$

Since $\underline{\partial} f(x)$ is a convex compact set, the point $v(w)$ is unique. Find now

$$\max_{w \in \overline{\partial} f(x)} \rho_1(w) = \rho_1(w(x)).$$

The point $w(x)$ is not necessarily unique. As x is not an inf-stationary point, then $\rho_1(w(x)) > 0$. The direction

$$g_1(x) = -\frac{v(w(x)) + w(x)}{\|v(w(x)) + w(x)\|} = -\frac{v(w(x)) + w(x)}{\rho_1(w(x))} \tag{22}$$

is a *steepest descent direction* of the function f at the point x, i. e.

$$f'(x, g_1(x))) = \min_{\|g\|=1} f'(x, g)\,.$$

Here $\|\cdot\|$ is the Euclidean norm. The quantity $f'(x, g_1(x)) = -\rho_1(w(x))$ is the *rate of steepest descent* of f

at x. It may happen that the set of steepest descent directions is not a singleton (and it need not be convex too). Recall that in the smooth case the steepest descent direction is always unique (if x is not a stationary point).

Similarly, if $x \in X$ is not a sup-stationary point of f (i. e. condition (18) does not hold) then let us take $v \in \partial f(x)$ and find

$$\min_{w \in \overline{\partial} f(x)} \|v + w\| = \|v + w(v)\| = \rho_2(v)$$

and

$$\max_{v \in \underline{\partial} f(x)} \rho_2(v) = \rho_2(v(x)) \,.$$

The direction

$$g_2(x) = \frac{v(x) + w(v(x))}{\|v(x) + w(v(x))\|} = \frac{v(x) + w(v(x))}{\rho_2(v(x))} \tag{23}$$

is a *steepest ascent direction* of the function f at x, i. e.

$$f'(x, g_2(x))) = \max_{\|g\|=1} f'(x, g) \,.$$

The quantity $f'(x, g_2(x)) = \rho_2(v(x))$ is the *rate of steepest ascent* of f at x. As above it may happen that there exist many steepest ascent directions.

Remark 7 Thus, the necessary conditions (16) and (18) are 'constructive': in the case where one of these conditions is violated we are able to find steepest descent or ascent directions.

The condition for a minimum (16) can be rewritten in the equivalent form

$$0_n \in \bigcap_{w \in \overline{\partial} f(x^*)} [\underline{\partial} f(x^*) + w] := L_1(x^*) \,, \tag{24}$$

and the condition for a maximum (18) can also be represented in the equivalent form

$$0_n \in \bigcap_{v \in \underline{\partial} f(x^{**})} [\overline{\partial} f(x^{**}) + v] := L_2(x^{**}) \,. \tag{25}$$

However, if, for example, (24) is violated at a point x, we are unable to recover steepest descent directions, it may even happen that the set $L_1(x)$ is empty (see [1, Sects. V.2 and V.3]).

Therefore, condition (24) is not 'constructive': if a point x is not inf-stationary then condition (24) supplies no information about the behavior of the function in a neighborhood of x and we are unable to get a 'better' point (e. g., to decrease the value of the function). The same is true for the condition for a maximum (25). Nevertheless conditions (25) and (25) may be useful for some other purposes.

Example 8 Let $x = (x^{(1)}, x^{(2)}) \in \mathbf{R}^2$, $x_0 = (0, 0)$, $f(x) = |x^{(1)}| - |x^{(2)}|$. We have $f(x) = f_1(x) - f_2(x)$, where $f_1(x) = \max\{f_3(x), f_4(x)\}$, $f_2(x) = \{f_5(x), f_6(x)\}$, $f_3(x) = x^{(1)}$, $f_4(x) = -x^{(1)}$, $f_5(x) = x^{(2)}$, $f_6(x) = -x^{(2)}$. The functions f_3–f_6 are smooth therefore (see (7))

$$\begin{aligned}
\mathrm{D}f_3(x) &= [\underline{\partial} f_3(x), \overline{\partial} f_3(x)] \,, \\
\text{with } \underline{\partial} f_3(x) &= \{(1, 0)\}, \ \overline{\partial} f_3(x) = \{(0, 0)\} \,, \\
\mathrm{D}f_4(x) &= [\underline{\partial} f_4(x), \overline{\partial} f_4(x)] \,, \\
\text{with } \underline{\partial} f_4(x) &= \{(-1, 0)\}, \ \overline{\partial} f_4(x) = \{(0, 0)\} \,, \\
\mathrm{D}f_5(x) &= [\underline{\partial} f_5(x), \overline{\partial} f_5(x)] \,, \\
\text{with } \underline{\partial} f_5(x) &= \{(0, 1)\}, \ \overline{\partial} f_5(x) = \{(0, 0)\} \,, \\
\mathrm{D}f_6(x) &= [\underline{\partial} f_6(x), \overline{\partial} f_6(x)] \,, \\
\text{with } \underline{\partial} f_6(x) &= \{(0, -1)\}, \ \overline{\partial} f_6(x) = \{(0, 0)\} \,,
\end{aligned}$$

Applying (11) one gets $\mathrm{D}f_1(x_0) = [\underline{\partial} f_1(x_0), \overline{\partial} f_1(x_0)]$, where

$$\begin{aligned}
\underline{\partial} f_1(x_0) &= eratornameco\{\underline{\partial} f_3(x_0) - \overline{\partial} f_4(x_0) \\
\underline{\partial} f_4(x_0) - \overline{\partial} f_3(x_0)\} &= \mathrm{co}\{(1, 0), (-1, 0)\} \,, \\
\overline{\partial} f_1(x_0) &= \{(0, 0)\} \,, \\
\mathrm{D}f_2(x_0) &= [\underline{\partial} f_2(x_0), \overline{\partial} f_2(x_0)] \,,
\end{aligned}$$

where

$$\begin{aligned}
\underline{\partial} f_2(x_0) &= \mathrm{co}\{\underline{\partial} f_5(x_0) - \overline{\partial} f_6(x_0), \underline{\partial} f_6(x_0) - \overline{\partial} f_5(x_0)\} \\
&= \mathrm{co}\{(0, 1), (0, -1)\} \,, \\
\overline{\partial} f_2(x_0) &= \{(0, 0)\} \,.
\end{aligned}$$

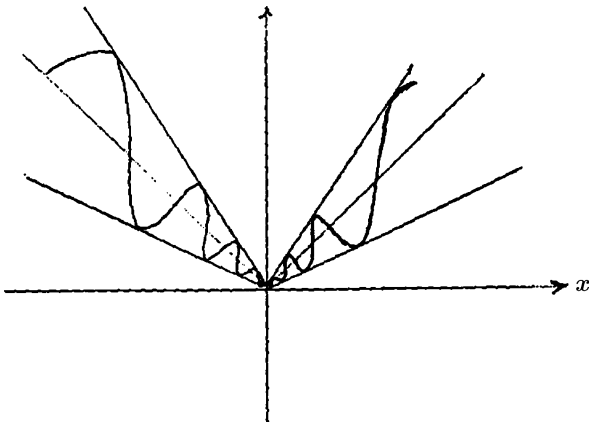
Finally, formula (8) yields

$$\mathrm{D}f(x_0) = [\underline{\partial} f(x_0), \overline{\partial} f(x_0)] \,,$$

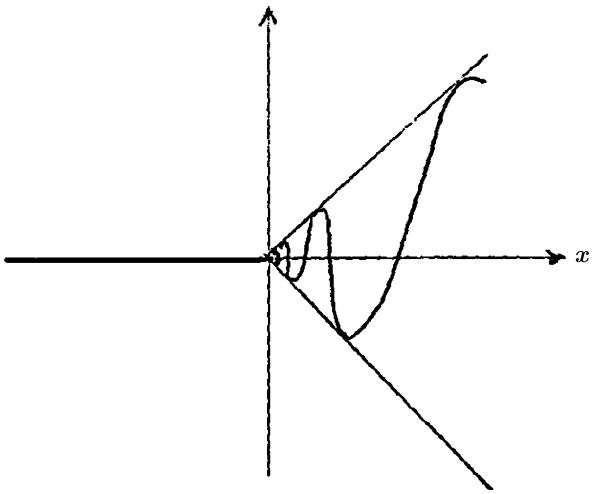
where

$$\begin{aligned}
\underline{\partial} f(x_0) &= \mathrm{co}\{(1, 0), (-1, 0)\} \,, \\
\overline{\partial} f(x_0) &= \mathrm{co}\{(0, 1), (0, -1)\} \,.
\end{aligned}$$

Since (see Fig. 1) conditions (16) and (18) are not satisfied, the point x_0 is neither inf-stationary nor sup-stationary.



Quasidifferentiable Optimization: Optimality Conditions, Figure 1



Quasidifferentiable Optimization: Optimality Conditions, Figure 2

Applying (22) and (23) we conclude that there exist two directions of steepest descent: $g_1 = (0, 1)$, $g_1' = (0, -1)$ and two directions of steepest ascent: $g_2 = (1, 0)$, $g_2' = (-1, 0)$.

It is also clear that the sets (see (24), (25))

$$L_1(x_0) = \bigcap_{w \in \overline{\partial} f(x_0)} [\underline{\partial} f(x_0) + w]$$

and

$$L_2(x_0) = \bigcap_{v \in \underline{\partial} f(x_0)} [\overline{\partial} f(x_0) + v]$$

are both empty.

Remark 9 If a function f is directionally differentiable but not quasidifferentiable, and if its directional derivative $f'(x, g)$ is continuous as a function of direction (this is the case, e. g., if f is directionally differentiable and Lipschitz) then (by the Stone–Weierstrass theorem) its directional derivative may be approximated (to within any given accuracy) by the difference of two positively homogeneous convex functions, i. e.

$$f'(x, g) \approx \max_{v \in A}(v, g) + \min_{w \in B}(w, g)\,, \tag{26}$$

where A and B are convex compact sets in $\mathbf{R}^n$. Relation (26) shows that f' can be studied by means of quasidifferential calculus (e. g., one is able to find an approximation of a steepest descent direction etc.). Corresponding results can be found in [1,4].

Remark 10 In many cases of practical importance the quasidifferential of a function f is a pair of sets each of them being the convex hull of a finite number of points or/and balls. If this happens it is easy to store and operate with the quasidifferential, to check necessary conditions, to find directions of descent or ascent, to construct numerical methods.

Necessary and Sufficient Conditions for a Constrained Optimum

Let a function f be defined and finite on some open set $X \subset \mathbf{R}^n$ and let $\Omega \subset X$. Consider the problem of finding a minimum or a maximum of f on Ω. For the definiteness in the sequel we shall consider only the problem of minimizing f on Ω since the problem of maximizing f is the problem of minimizing the function $f_1 = -f$.

Let $x \in \Omega$. The set

$$\Gamma(x, \Omega) = \left\{ g \in \mathbf{R}^n : \begin{array}{c} \exists\{[\alpha_k, g_k]\}: \\ [\alpha_k, g_k] \to [+0, g] \\ x + \alpha_k g_k \in \Omega \\ \forall k \end{array} \right\} \tag{27}$$

is called the *Bouligand cone* to the set Ω at the point x (or the *cone of feasible directions*). It is nonempty and closed. If $x \in \operatorname{int} \Omega$ then $\Gamma(x, \Omega) = \mathbf{R}^n$.

Proposition 11 *Let f be Hadamard directionally differentiable at a point $x^* \in \Omega$. For the point x^* to be a local or global minimizer of f on Ω it is necessary that*

$$f'_H(x^*, g) \geq 0\,, \quad \forall g \in \Gamma(x^*, \Omega)\,. \tag{28}$$

If

$$f'_H(x^*, g) > 0\,, \quad \forall g \in \Gamma(x^*, \Omega),\ g \neq 0_n\,, \tag{29}$$

then x^* is a strict local minimizer of f on Ω, i. e. there exists $\delta > 0$ such that

$$f(x) > f(x^*)\,,$$
$$\forall x \in \Omega,\ \|x - x^*\| < \delta,\ x \neq x^*\,.$$

The set $\Omega \subset \mathbf{R}^n$ is called *quasidifferentiable* if it can be represented in the form

$$\Omega = \{x \in \mathbf{R}^n :\ h(x) \leq 0\}\,, \tag{30}$$

where h is a quasidifferentiable function.

Take $x \in \Omega$ and consider the cones

$$\gamma_1(x) = \{g \in \mathbf{R}^n :\ h'(x, g) < 0\}\,,$$
$$\Gamma_1(x) = \{g \in \mathbf{R}^n :\ h'(x, g) \leq 0\}\,.$$

Let $h(x) = 0$. We say that the 'regularity condition is satisfied at x' if

$$\operatorname{cl}\gamma_1(x) = \Gamma_1(x)\,. \tag{31}$$

If $h(x) = 0$ (i. e., by (30), $x \in \Omega$) and the regularity condition (31) holds then

$$\Gamma(x, \Omega) = \Gamma_1(x)\,. \tag{32}$$

Now we are able to express condition (28) and (29) in terms of quasidifferentials of the functions f and h.

If $h(x) < 0$ then $x \in \operatorname{int}\Omega$, $\Gamma(x, \Omega) = \mathbf{R}^n$ and, by Proposition 5, conditions (16) and (17) hold. Therefore let us consider the case $h(x) = 0$.

Proposition 12 *Let functions f and h be Hadamard quasidifferentiable at a point $x^* \in \Omega$ and $h(x^*) = 0$. Assume also that the regularity condition (31) is satisfied at x^*. For the point x^* to be a local or global minimizer of f on Ω it is necessary that*

$$\left(\underline{\partial} f(x^*) + w\right) \bigcap \left[-\operatorname{cl}(\operatorname{cone}(\underline{\partial} h(x^*) + w'))\right] \neq \emptyset \tag{33}$$

for all $w \in \overline{\partial} f(x^*)$, $w' \in \overline{\partial} h(x^*)$.

Condition (33) is equivalent to the condition

$$-\overline{\partial} f(x^*) \subset L(x^*)\,, \tag{34}$$

where

$$L(x) = \bigcap_{w \in \overline{\partial} h(x)} \left[\underline{\partial} f(x) + \operatorname{cl}(\operatorname{cone}(\underline{\partial} h(x) + w))\right]\,.$$

The set $L(x)$ is nonempty and convex.

If $h(x^*) = 0$ and

$$-\overline{\partial} f(x^*) \subset \operatorname{int} L(x^*)\,, \tag{35}$$

then x^* is a strict local minimizer of f on Ω.

A point $x^* \in \Omega$ is called an inf-*stationary point* of f on Ω if condition (28) holds.

Let $x \in \Omega$, $h(x) = 0$. Assume that x is not an inf-stationary point and find

$$\min_{z \in [\underline{\partial} f(x) + w]} \|z + z'\| = \|z(w, w') + z'(w, w')\|$$
$$= \|v(w, w') + w + v'(w, w') + w'\|$$
$$= \|q(w, w')\| = d(w, w')$$

and

$$\rho(x) = \max_{\substack{w \in \overline{\partial} f(x) \\ w' \in \overline{\partial} h(x)}} d(w, w') \tag{36}$$
$$= d(w_0, w'_0) = \|q(w_0, w'_0)\|\,.$$

Since x is not inf-stationary then $\rho(x) > 0$.

Proposition 13 *If $h(x) = 0$ and the regularity condition (31) holds then the direction*

$$g_0 = -\frac{q(w_0, w'_0)}{\rho(x)} \tag{37}$$

is a steepest descent direction of f on Ω at x and $g_0 \in \Gamma(x, \Omega)$,

$$f'(x, g_0) = \min_{\|g\| = 1, g \in \Gamma(x)} f'(x, g)$$
$$= -\|q(w_0, w'_0)\| = -\rho(x)$$

i. e. $-\rho(x)$ is the rate of steepest descent.

Remark 14 If there exist several pairs $[w_0, w'_0](w_0 \in \overline{\partial} f(x), w'_0 \in \overline{\partial} h(x))$ satisfying (36), then (by (37)) there are several steepest descent directions.

Remark 15 Condition (33) is also equivalent to

$$0_n \in \bigcap_{\substack{w \in \overline{\partial} f(x') \\ w' \in \overline{\partial} h(x^*)}} \left[\underline{\partial} f(x^*) + w + \operatorname{cl}(\operatorname{cone}(\underline{\partial} h(x^*) + w'))\right]$$
$$= L'(x^*)\,. \tag{38}$$

However, condition (38) is not 'constructive' since the set $L'(x)$ may happen to be empty if x is not a stationary point (we consider the case $h(x) = 0$).

Proposition 16 *Let $x^* \in \Omega$ and $h(x^*) = 0$. Assume that the functions f and h are quasidifferentiable at x^*. For the point x^* to be a local or global minimizer of f on Ω it is necessary that*

$$L_1(x^*) \subset L_2(x^*)\,, \tag{39}$$

where

$$L_1(x) = -[\overline{\partial} f(x) + \overline{\partial} h(x)]\,,$$
$$L_2(x) = \mathrm{co}\{\underline{\partial} f(x) - \overline{\partial} h(x)\,, \underline{\partial} h(x) - \overline{\partial} f(x)\}.$$

If, in addition, f and h are Hadamard q.d. at x^*, $h(x^*) = 0$ and $L_1(x^*) \subset \mathrm{int}\, L_2(x^*)$ then x^* is a strict local minimizer of f on Ω.

Proposition 17 *Let $h(x^*) = 0$, f and h be Hadamard q.d. at x^*. If the regularity condition (31) holds at x^* then condition (39) is equivalent to condition (28).*

Let $x \in \Omega$, $h(x) = 0$. Assume that (39) does not hold. Find

$$d(x) = \max_{v \in L_1(x)} \rho(v) = \rho(v(x)),$$

where

$$\rho(v) = \min_{w \in L_2(x)} \|v - w\| = \|v - w(v)\|\,.$$

Since (39) is not satisfied then $\rho(v(x)) > 0$.

Proposition 18 *The direction*

$$g_0' = \frac{v(x) - w(v(x))}{\rho(v(x))} \tag{40}$$

is a descent direction of f on Ω at x.

Remark 19 While the steepest descent direction g_0 (see (37)) may be not admissible, the direction $g_0{}'$ (see (40)) is always admissible, i. e. for sufficiently small $\alpha > 0$ we have $x + \alpha\, g_0{}' \in \Omega$.

Recent results and the state-of-the-art in quasidifferential calculus can be found in [2].

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Demyanov VF, Rubinov AM (1995) Constructive nonsmooth analysis. P. Lang, Frankfurt am Main

2. Demyanov VF, Rubinov AM (eds) (2000) Quasidifferentiablity and related topics. Kluwer, Dordrecht
3. Demyanov VF, Vasiliev LV (1986) Nondifferentiable optimization. Springer and Optim. Software, Berlin
4. Gorokhovik VV (1986) ε-quasidifferentiability of real-valued functions and optimality conditions in extremal problems. Math Program Stud 29:203–218
5. Xia ZQ (1987) On mean-value theorems in quasidifferential calculus. J Math Res Exp Dalian, China, DIT 4:681–684

Quasidifferentiable Optimization: Stability of Dynamic Systems

GEORGIOS E. STAVROULAKIS
Carolo Wilhelmina Techn. University, Braunschweig, Germany

MSC2000: 74G60, 74H99, 49J52, 49Q10, 74K99, 74Pxx, 90C90

Article Outline

Keywords
Smooth Potentials and Stability in Mechanics
Incremental Algorithm
Nonsmooth Superpotentials
Nonsmooth Local Approximations
Stability Results. Discussion
See also
References

Keywords

Quasidifferentiability; Nonsmooth mechanics; Elastic stability

Problems in mechanics whose governing relations can be obtained from a generally nondifferentiable and nonconvex, but quasidifferentiable (in the sense of V.F. Demyanov and A.M. Rubinov) potential function are considered. They consider a fairly general form for the modeling and the study of nonsmooth problems in mechanics [4] and they cover certain classes of variational and hemivariational inequality problems of mechanics [14,15]. The notion of hemivariational inequalities has been introduced and thoroughly studied in mechanics by P.D. Panagiotopoulos (see also ► Nonconvex energy functions: Hemivariational inequalities; ► Hemivariational inequalities: Applications in mechanics). Moreover, there exists extensive theoretical support for the use of quasidifferentiable calculus and optimization techniques, see, e. g., [3,4]. For methods and heuristic algorithms of nonconvex optimization in computational mechanics, see [12]. In this short note some techniques for treating stability problems for nonsmooth structures are outlined. This way results for classical, smooth structures (e. g., [1,10,11]) can be extended to cover nonsmooth ones (cf., also [8,9]). This work and the preliminary results outlined here are based on [18,19].

All previously mentioned potentials are piecewise-differentiable and may be described, in general, as continuous selections of differentiable functions. In turn, the structural analysis problem results from minimality or in general critical point conditions of the potential (see examples in [2,6,7,14,15]).

Results from stability analysis of parametric optimization problems for nondifferentiable functions are used for the study of a stepwise holonomic, incremental structural analysis problem. In particular the systematic first and second order linearizations proposed in respectively, and the arising normal forms are adopted for the potential energy function.

The techniques outlined here may be useful both for the analysis of the stability of structures which involve nonmonotone and possibly multivalued nonlinearities (in a holonomic or a stepwise holonomic setting) and for the design of incremental-iterative algorithms for structural analysis purposes.

Smooth Potentials and Stability in Mechanics

Let a discretized elastostatic analysis problem be formulated as a potential energy minimization problem:

$$\min_{u \in U_{\text{ad}}} \left\{ \overline{\Pi}(u, \lambda) = \Pi(e(u)) - p(u, \lambda) \right\} . \tag{1}$$

Here u is the n-vector of displacement degrees of freedom, e is the m-vector of discrete element deformations, $\Pi\ (e) \in \mathbf{R}$ is the internal energy density, $p\ (u, \lambda) \in \mathbf{R}$ is the external loading potential, parametrized by a loading scalar $\lambda \in \mathbf{R}^1$ and U ad $\subseteq \mathbf{R}^n$ is the space of admissible displacements. Displacement u and deformation e vectors are connected by the geometric compatibility operator $A(u)$: $\mathbf{R}^n \to \mathbf{R}^m$ such that $e\ (u) = A\ (u)$ holds.

On the assumption that $\overline{\Pi}(u,\lambda)$ is smooth, the equilibrium configurations for the structural system are critical points of this potential, i. e. for fixed $\lambda = \overline{\lambda}$ one has:

$$\Sigma_{\text{crit}} = \left\{ u \in \mathbf{R}^n \colon \ \nabla_u \overline{\Pi}(u,\overline{\lambda}) = 0 \right\} . \tag{2}$$

Moreover, inspection of the second order derivatives (the Hessian matrix of $\overline{\Pi}(u,\overline{\lambda})$) gives us stability information [1]. If u is a nondegenerate critical point, i. e. $u \in \Sigma_{\text{crit}}$ and $\nabla_u^2 \overline{\Pi}(u,\overline{\lambda})$ is regular, then a positive or negative definite Hessian $\nabla_u^2 \overline{\Pi}(u,\overline{\lambda})$ indicates that the point u is a local minimum or maximum, respectively. Only local minima correspond to stable equilibrium configurations. If $\nabla_u^2 \overline{\Pi}(u,\overline{\lambda})$ is singular in $u \in \Sigma_{\text{crit}}$, then higher order derivatives of $\overline{\Pi}(u,\overline{\lambda})$ must be examined for stability [1].

If u_0 is either a noncritical or a nondegenerate critical point of $\overline{\Pi}(u,\overline{\lambda})$, which is assumed here to be at least a C^2-function, i. e. two times continuously differentiable, then $\overline{\Pi}(u,\overline{\lambda})$ is C^1-equivalent to its second order approximation around u_0, i. e.

$$\begin{aligned}(\overline{\Pi} \circ \Phi)(u) = {} & \overline{\Pi}(u_0,\overline{\lambda}) + \nabla_u \overline{\Pi}(u_0,\overline{\lambda})^{\top}(u-u_0) \\ & + \frac{1}{2}(u-u_0)^{\top} \nabla_u^2 \overline{\Pi}(u_0,\overline{\lambda})(u-u_0) ,\end{aligned} \tag{3}$$

where Φ is a C^1-coordinate transformation (diffeomorphism). In the vicinity of a nondegenerate critical point the behavior of $\overline{\Pi}(u,\overline{\lambda})$ is characterized by the number of negative eigenvalues of $\nabla_u^2 \overline{\Pi}(u,\overline{\lambda})$ (the *quadratic index*).

In the coordinates $\Delta u = u - u_0$ and the notation $\Delta\overline{\Pi}(\Delta u,\overline{\lambda}) = \overline{\Pi}(u,\overline{\lambda}) - \overline{\Pi}(u_0,\overline{\lambda})$ we can determine (cf. [5, p. 21]) a local C^1-coordinate transformation Φ: $U \to V$, where U, V are neighborhoods of the origin, such that:

$$\begin{aligned}\Delta\overline{\Pi} \circ \Phi^{-1}(y) = {} & -y_1^2 - \cdots - y_k^2 + y_{k+1}^2 + \cdots + y_n^2 , \\ & \forall y \in V .\end{aligned} \tag{4}$$

Qualitative stability results for fixed load $\lambda = \overline{\lambda}$ are recognized in the normal form (4).

Incremental Algorithm

Incremental-iterative solution algorithms are based on appropriate approximations of (1). Let us consider the one-parametric load incrementation on the following case of (1) (cf. [10]):

$$\min_{u \in \mathbf{R}^n} \{ \overline{\Pi}(u,\lambda) = \Pi(u) - \lambda p^{\top} u \} . \tag{5}$$

For equilibrium we have

$$\nabla_u \overline{\Pi}(u,\lambda) = 0 \quad \Rightarrow \quad \nabla_u \Pi(u) - \lambda p = 0 \tag{6}$$

For the examination of the stability of a solution we study the following relation in terms of $\lambda - \lambda_0 = \Delta\lambda$, (defining Δu as a function of $\Delta\lambda$, if $\nabla_u^2 \Pi(u_0)$ is regular)

$$\nabla_u^2 \Pi(u_0)\Delta u + \Delta\lambda p = 0 , \tag{7}$$

which connects the incremental displacement Δu for a change of loading equal to $\Delta\lambda p$. Relation (7) can be produced by subtraction of the Taylor expansions of the equilibrium equation (6) in $(u_0 + \Delta u, \lambda_0 + \Delta\lambda)$ and (u_0, λ_0), respectively, and by using the approximation (up to higher order terms)

$$\nabla_u \Pi(u_0 + \Delta u) = \nabla_u \Pi(u_0) + \nabla_u^2 \Pi(u_0)\Delta u . \tag{8}$$

Consider the coordinate transformation: $\Delta u = \Phi^{-1}(y) = \phi_i y_i = F y$ where ϕ_i are the eigenvectors of $\nabla^2 \Pi(u_0)$ and the summation convention over repeated indices is used. Then equation (7) is written in the new coordinate system as:

$$\begin{aligned}& \nabla^2 \Pi(u_0) F y - p\Delta\lambda = 0 \\ & \quad \Rightarrow \quad F^{\top} \nabla^2 \Pi(u_0) F y - F^{\top} p \Delta\lambda = 0 \\ & \quad \Rightarrow \quad \left[\omega_i y_i\right] - F^{\top} p \Delta\lambda = 0 .\end{aligned} \tag{9}$$

Here ω_i are the eigenvalues of the local tangential stiffness matrix $K(u_0) = \nabla^2 \Pi(u_0)$, which act as stability coefficients for the linearized equation of equilibrium (7) [10,11].

Nonsmooth Superpotentials

Let us assume problem (1) with a nonsmooth potential energy function. For simplicity let only the internal energy function $\Pi(u)$ be nonsmooth and $U_{\text{ad}} = \mathbf{R}^n$ in (1).

Let V denote an open subset of $\mathbf{R}^n$. We call a function f: $V \to \mathbf{R}$ a *continuous selection* of the C^r-functions g_i: $V \to \mathbf{R}$, $1 \le i \le k$, (briefly, $f \in \text{CS}\{g_1, \ldots, g_k\}$), if f is continuous and $\forall u \in V\ \exists i \in \{1, \ldots, k\}$: $f(u) = g_i(u)$. Let

$\Pi(u)$ in (1) be a piecewise differentiable PC^r) function of appropriate order $r > 1$, defined on an open set $U \subseteq \mathbf{R}^n$. This means that (cf. [7]) at every point $u_0 \in U$ there exists an open neighborhood $V \subseteq U$ and a finite collection of C^r-functions $\{\Pi_1, \ldots, \Pi_k\}$ defined on V such that $\Pi|_V \in \mathrm{CS}\{\Pi_1, \ldots, \Pi_k\}$.

Let $I(u)$ be the *active index set* set$\{i\colon \Pi(u) = \Pi_k(u)\}$ One considers a smooth external loading potential $p(u, \lambda)$, which depends on the one-dimensional loading parameter λ (cf. (1)).

The assumption of a PC^r-potential energy function is very general and covers a large number of nonsmooth mechanics applications (see, also, [13, Chap. 8]). More detailed analysis of the requirements which are necessary in order for a PC^r-function to be the potential of a certain structural analysis problem must be investigated on a case-by-case basis.

Any PC^r-potential is locally Lipschitz continuous and Bouligand differentiable with the B-derivative at a point $u_0 \in \mathbf{R}^n$ in the direction $d \in \mathbf{R}^n$ being a continuous selection of the functions $\nabla\Pi_i(u_0)^\top d, i \in \widehat{I}(u_0)$. Here $\widehat{I}(u_0)$ denotes the *essentially active index set* $\widehat{I}(u_0) = \{i \in I(u_0)\colon\ u_0 \in \mathrm{cl}(\mathrm{int}(\{u \in U\colon \Pi(u) = \Pi_i(u)\}))\}$, with cl (resp. int) abbreviating the closure (resp. the interior) of a set. For completeness, recall that *Clarke's generalized subdifferential* is given by [7] $\partial_{\mathrm{Cl}}\Pi(u_0) = \mathrm{conv}\left\{\nabla\Pi_i(u_0)\colon\ i \in \widehat{I}(u_0)\right\}$ where conv stands for the convex hull.

Nonsmooth Local Approximations

For the needs of the applications in mechanics the first and the second order differentiation, or the appropriate analogous nonsmooth notions, and suitable local nonsmooth approximations which generalize the (second order) Taylor expansion of a smooth function are used. A local coordinate transformation will provide us with a simple formulation of the energy minimization problem, cf. (4), which, in turn, will be used for the extraction of stability information analogous to (9).

Following [6], a critical point u_0 of a PC^2-potential function $\overline{\Pi}(u)$ is called a *nondegenerate critical point* if $\overline{\Pi}$ is locally representable as a continuous selection of functions $\overline{\Pi}_1, \ldots, \overline{\Pi}_k$ such that the following properties are true:

ND1) the vectors $\nabla\overline{\Pi}_j(u_0), j \in I(u_O)\backslash\{i\}$ are linearly independent $\forall \mathrm{i} \in I(u_0)$,

ND2) the restricted Hessian of the Lagrangian, the matrix $\nabla^2 L(u_0)\,|_{V(u_0)}$, is invertible.

Here $V(u\}_0)$ denotes the space

$$\left\{ y \in \mathbf{R}^n\colon \begin{array}{c} [\nabla\overline{\Pi}_i(u_0) - \nabla\overline{\Pi}_j(u_0)]^\top y = 0, \\ i, j \in I(u_0) \end{array} \right\}.$$

For the Lagrangian

$$L(u) = \sum_{i \in I(u_0)} \lambda_i \overline{\Pi}_i(u)$$

holds

$$\begin{aligned} &\sum_{i \in I(u_0)} \lambda_i \nabla\overline{\Pi}_i(u_0) = 0\,, \\ &\sum_{i \in I(u_0)} \lambda_i = 1\,, \quad \lambda_i \geq 0\,. \end{aligned} \tag{10}$$

The qualitative behavior of the potential energy function, the link to the stability of the described mechanical system, can be shown if one considers the normal form (cf. (4))). In this context, the following result of [6] is of importance.

Let $\overline{\Pi} \in \mathrm{CS}(\overline{\Pi}_i, i \in I)$ and let $u_0 \in \mathbf{R}^n$ be a nondegenerate critical point for $\overline{\Pi}$ with quadratic index equal to q. Suppose moreover that $|I_0(u_0)| = k + 1$. Then at u_0, the potential is topologically equivalent to $g(y)$, where

$$\begin{aligned} g(y_1, \ldots, y_n) = \overline{\Pi}(u_0) + \mathrm{CS}\left(y_1, \ldots, y_k, -\sum_{i=1}^{k} y_i\right) \\ - \sum_{j=k+1}^{k+q} y_j^2 + \sum_{r=k+q+1}^{n} y_j^2\,. \end{aligned} \tag{11}$$

One observes that the second term in the right-hand side of (11) is sufficiently rich to describe locally every type of nonsmooth, finite-dimensional functions.

Furthermore, following [7] one notes that a PC^2-function can always be transformed into the min-max normal form:

$$\overline{\Pi}(u) = \max_{1 \leq i \leq k} \min_{j \in M_i} \overline{\Pi}_j(u)\,, \tag{12}$$

where (12) is considered as a local representation of the potential in a neighborhood of u_0, $M_i \subseteq \{1, \ldots, \mathrm{m}\}$ and the functions $\overline{\Pi}_j\colon\ U \to \mathbf{R}$, $j \in \{1, \ldots, \mathrm{m}\}$, are C^2-functions.

In this case a consistent nonsmooth second order approximation of the PC^2-potential, expressed by the normal form (12), is given by:

$$\max_{1\le i\le k}\min_{j\in M_i}\left\{\begin{array}{c}\overline{\Pi}_j(u_0)+\nabla\overline{\Pi}_j(u_0)^\top(u-u_0)\\ +\frac{1}{2}(u-u_0)^\top\nabla^2\overline{\Pi}_j(u_0)(u-u_0)\end{array}\right\}. \tag{13}$$

Note here that the previously denoted min-max form is not defined in an unique way.

Stability Results. Discussion

For a structural analysis system with a structured nonsmooth PC^2-potential with (11) and for a nondegenerate critical point $u_0 \in \mathbf{R}^n$, the local approximation (11) is available. Let us assume a potential of the external loading equal to $\lambda\, p^\intercal\, u$, as in (5) and let for the present the load parameter λ be fixed to a given value. From (11) the following complete subdivision of the coordinate space $\mathbf{R}^n$ arises:

$$\mathbf{R}^n = \mathbf{R}^k \oplus \mathbf{R}^q \oplus \mathbf{R}^{n-k-q} = \mathbf{R}^{\mathrm{non}} \oplus \mathbf{R}^{\mathrm{un}} \oplus \mathbf{R}^{\mathrm{st}}\,, \tag{14}$$

where $\mathbf{R}^{\mathrm{non}}$ stands for the essentially nondifferentiable subspace, $\mathbf{R}^{\mathrm{un}}$ for the unstable subspace and $\mathbf{R}^{\mathrm{st}}$ for the stable subspace. Let, moreover, the local coordinate transformation that leads to (11) be traced for the components of the vector $\lambda\ p$ (cf. (6)–(9)). Let the components of the last vector in the three subspaces of ((14)) be $\widetilde{p}^{\mathrm{non}}$, $\widetilde{p}^{\mathrm{un}}$ and $\widetilde{p}^{\mathrm{st}}$, respectively.

Further one considers the type of the CS in the linear term of the right-hand side in (11) in comparison with the three above defined components of the loading vector. This information can be used for stability analysis. Smooth and nonsmooth contributions should be treated separately. For the nonsmooth part, for example, if one has a max-type function and $q = 0$, then only stable local minima of the potential energy function arise.

The above outlined scheme can be followed for the derivation of stability considerations for a structure at a given point and for a given loading level (λ is constant). For the examination of the stability question along a given loading path (one-parametric change of λ) one should take into account that the local representation (11) may change as λ changes. The results are qualitative of the same nature, but, for practical applications, a combinatorial problem arises, which concerns the way of possible changes of the subdivision (14) as loading changes. Further work in this direction will generalize the computational mechanics techniques for the tracing of post-buckling equilibria in nonsmooth mechanics' applications. Theoretical support will be provided by the theory of parametric optimization (cf., e. g., [5] and the applications in contact mechanics [16,17]).

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-Nonsmooth Calculus of Variations
- ▶ Optimization Strategies for Dynamic Systems
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities

- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Bazant ZP, Cedolin L (1991) Stability of structures. Elastic, inelastic, fracture and damage theories. Oxford Univ. Press, Oxford
2. Curnier A, He Q-C, Zysset Ph (1995) Conewise linear elastic materials. J Elasticity 37:1–38
3. Demyanov VF, Rubinov AM (1985) Quasidifferentiable calculus. Optim. Software, New York
4. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
5. Guddat J, Guerra Vasquez F, Jongen ThH (1990) Parametric optimization: Singularities, path following and jumps. Teubner and Wiley, New York
6. Jongen ThH, Pallaschke D (1988) On linearization and continuous selection of functions. Optim 19(3):343–353
7. Kuntz L, Scholtes S (1995) Qualitative aspects of the local approximation of a piecewise differentiable function. Nonlinear Anal Th Methods Appl 25(2):197–215
8. Kurutz M (1993) Stability of structures with nonsmooth nonconvex energy functionals. The one dimensional case. Europ J Mechanics A Solids 12(3):347–385
9. Rohde A, Stavroulakis GE (1997) Genericity analysis for path-following methods. Application in unilateral contact elastostatics. Z Angew Math Mechanics (ZAMM) 77(10):777–790
10. Kurutz M (1994) Equilibrium paths of polygonally elastic structures. Mechanics of Structures and Machines 22(2):181–210
11. Li L-Y (1991) The criteria of identifying the type of critical points. Arch Appl Mechanics 61:231–235
12. Li L-Y (1994) Determination of stability in nonlinear analysis of structures. Arch Appl Mechanics 64:119–126
13. Mistakidis ES, Stavroulakis GE (1997) Nonconvex optimization in mechanics. Algorithms, heuristics and engineering applications by the F.E.M. Kluwer, Dordrecht
14. Pallaschke D, Rolewicz S (1997) Foundations of mathematical optimization. Convex analysis without linearity. Kluwer, Dordrecht
15. Panagiotopoulos PD (1985) Inequality problems in mechanics and applications. Convex and nonconvex energy functions. Birkhäuser, Basel
16. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Berlin
17. Rohde A, Stavroulakis GE (1995) Path–following energy optimization in unilateral contact problems. J Global Optim 6:347–365
18. Stavroulakis GE, Rohde A (1996) Stability of structures with quasidifferentiable energy functions. In: Sotiropoulos D, Beskos DE (eds) 2nd Greec Conf. Computational Mechanics, Chania, June 1996. GRACAM, pp 406–413
19. Stavroulakis GE, Rohde A (1999) Normal forms and stability in nonsmooth potential elastostatics. Mechanics Res Comm 26(2):185–190

Quasidifferentiable Optimization: Variational Formulations

GEORGIOS E. STAVROULAKIS
Carolo Wilhelmina Techn. University, Braunschweig, Germany

MSC2000: 74G99, 74H99, 74Pxx, 49J40, 49M05, 49S05

Article Outline

Keywords
Variational Formulation of Subdifferential Laws
Variational Formulation of Quasidifferential Laws
Example: an Elastostatic Problem Involving QD-Superpotentials
Variational Equality
Convex Variational Inequality
QD Laws and Systems of Variational Inequalities
See also
References

Keywords

Variational problems; Variational inequalities; Nonsmooth optimization; Nonsmooth mechanics; Quasidifferentiability

In science and, especially in engineering, the variational or weak formulation of a given boundary value problem has certain advantages. Instead of writing pointwise relations (for example, partial differential equations) which hold for each point of the considered system, one multiplies the governing relation with an arbitrary virtual variation, integrates over the entire area and requires that the latter integral be equal to zero. This is a weak or a variational formulation of the problem. Since the considered virtual variation is arbitrary

one gets back, on the assumption of sufficient regularity, the initial pointwise relations.

Variational formulations provide the basis for the development of numerical approximation methods (for example, by the *finite element method*). One of the advantages is that by performing partial integration one transfers differentiability requirements from the actual variables of the problem to the virtual ones, which, in turn, results in less demanding requirements on the complexity of the required finite element basis approximation functions. The literature on variational problems is very large, so that every selection of references would be incomplete. In this sense, let us mention here the publications [1,3,7,8,14].

In the language of smooth optimization, instead of considering the first order optimality condition that the derivative of a function at a given point is equal to zero, one proceeds as follows. The latter equation is multiplied by a virtual change of the variables along an arbitrary direction. Then, one considers the equivalent relation that the directional derivative of the function is equal to zero for all directions emanating from the assumed point.

In mechanics the arising quantities have a physical meaning (for instance, they correspond to the virtual work of a system). For historical reasons one speaks about *variational principles*. Moreover, on adequate smoothness assumptions one writes variational equalities. Finally, for engineering applications, and depending on the nature of the studied problem, one has to solve, after numerical discretization, systems of linear or nonlinear equations.

In connection with convex, nondifferentiable potentials or for convex problems with inequality constraints it is intuitively conceivable that not all virtual variations are allowed for. The theory of variational inequalities has been developed for the study of this class of problems. It is connected with the subdifferential of convex analysis and it is appropriate for the study of monotone operators [5,9]. In simple cases, or after appropriate reformulations one gets linear or nonlinear complementarity problems (see, e. g. [6] for a recent review).

For general nonconvex and nonsmooth problems a nonconvex extension of the notion of the variational inequality is required. For potential operators and by using the generalized subdifferential in the sense of F.H. Clarke, this class of variational problems have been developed and studied by P.D. Panagiotopoulos, who called them *hemivariational inequalities*. See ► Nonconvex energy functions: Hemivariational inequalities; ► Hemivariational inequalities: Applications in mechanics or [9,11] for more details.

The notion of quasidifferentiability, in the sense of V.F. Demyanov and A.M. Rubinov, provides an elegant way for the formulation and study of nonconvex variational inequality problems. By taking advantage of the ability of the quasidifferentials to provide a qualitative and quantitative nonsmooth approximation of a nonsmooth function one arrives at a very powerful variational description of the problem. This link has been studied for several applications in mechanics in [4,10,12]. One should mention that the author's understanding of this theory and their first attempts have been based on previous theoretical results of C.A. Stuart and J.F. Toland [13] and G. Auchmuty [2] concerning difference convex energy functions. Of course, the class of *difference convex functions* is included in the class of quasidifferentiable functions, so that the here presented approach is sufficiently general.

Variational Formulation of Subdifferential Laws

Let us assume a monotone possibly multivalued (i. e., with complete vertical branches) relation (a law) between the quantities u and $-f$. To be more precise, one may think about a nonlinear boundary law which connects boundary reactions $-f$ with boundary displacements u in mechanics. Let a convex l.s.c. and proper function Φ exists, the convex superpotential in the sense of J.-J. Moreau, and that the previously mentioned law is written in the subdifferential form:

$$-f \in \partial\Phi(u)\,. \tag{1}$$

Here ∂ denotes the subdifferential of convex analysis. Function $\Phi(u)$ can be considered as the potential energy corresponding to the mechanical law (1).

By definition, (1) is equivalent to the following variational inequality:

$$\Phi(u^*) - \Phi(u) \geq -\langle f, u^* - u\rangle\,, \quad \forall u^* \in \mathbf{R}\,. \tag{2}$$

For example, if Φ is the indicator I_K of a convex closed interval K of $\mathbf{R}$, then one has

$$-f \in \partial I_K(u)\,. \tag{3}$$

This is a unilateral constraint as one easily recognizes by considering the equivalent variational inequality (for $u \in K$):

$$\langle f, u^* - u\rangle \geq 0\,, \quad \forall u^* \in K\,. \tag{4}$$

Indeed, if $u_\star - u$ is an admissible variation of u (in the sense that it satisfies (4)), then the same does not hold for the variation $u - u_\star$. Of course in the one-dimensional case $\langle\cdot,\cdot\rangle$ is a simple multiplication. For multidimensional problems it will be an inner product.

Analogous subdifferential relations can be written for multidimensional laws (for example, for constitutive laws in elastoplasticity [9]).

Variational Formulation of Quasidifferential Laws

Let us assume now a nonmonotone possibly multivalued relation. By means of a real-valued, *quasidifferentiable superpotential* energy function Φ, one may expresses this relation in the form:

$$-f = w_1 + w_2\,, \tag{5}$$

with $\{w_1, w_2\} \in D\Phi(u) = [\underline{\partial}\Phi(u), \overline{\partial}\Phi(u)]$.

By definition, (5) is equivalent to the relation:

$$\begin{aligned}\langle -f, u^* - u\rangle = &\max_{w_1^* \in \underline{\partial}\Phi(u)} \langle w_1^*, u^* - u\rangle \\ &+ \min_{w_2^* \in \overline{\partial}\Phi(u)} \langle w_2^*, u^* - u\rangle\,, \\ &\forall u^* \in U\,,\end{aligned} \tag{6}$$

for $u \in U$, or with the *system of variational inequalities*

$$\begin{aligned}&\langle -f, u^* - u\rangle - \langle w_2^*, u^* - u\rangle \leq \max_{w_1^* \in \underline{\partial}\Phi(u)} \langle w_1^*, u^* - u\rangle\,, \\ &\forall u^* \in U\,, \quad \forall w_2^* \in \overline{\partial}\Phi(u)\,,\end{aligned} \tag{7}$$

and

$$\begin{aligned}&\langle -f, u^* - u\rangle - \langle w_1^*, u^* - u\rangle \geq \min_{w_2^* \in \overline{\partial}\Phi(u)} \langle w_2^*, u^* - u\rangle\,, \\ &\forall u^* \in U\,, \quad \forall w_1^* \in \underline{\partial}\Phi(u)\,.\end{aligned} \tag{8}$$

Space U is in general a subspace of $\mathbf{R}^n$ and depends on the considered application.

Analogously one treats multidimensional relations (for example, boundary adhesive layers) or constitutive laws (e. g., materials with softening effects). A number of concrete examples have been given in [4, Chap. 3].

Example: an Elastostatic Problem Involving QD-Superpotentials

Let $\Omega \subset \mathbf{R}^3$ be an open bounded subset occupied by a deformable body in its undeformed state. On the assumption of small deformations one writes the virtual work relation

$$\begin{aligned}&\int_\Omega \sigma_{ij}(u)\varepsilon_{ij}(v-u)\,d\Omega \\ &\quad = \int_\Omega f_i(v_i - u_i)\,d\Omega + \int_\Gamma \sigma_{ij} n_j (v_i - u_i) d\Gamma\,, \\ &\forall v \in V\,,\end{aligned} \tag{9}$$

for $u \in V$. Here V denotes the function space of the displacements which will be defined further. As it has been outlined previously, for the derivation of (9) one multiplies the equilibrium equation:

$$\sigma_{ij,j} + f_i = 0\,, \tag{10}$$

where f_i is the volume force vector, by a virtual displacement $v_i - u_i$ and then we have integrated over Ω. On the assumption of appropriately regular functions, one applies the Green–Gauss theorem by taking into account the strain-displacement relation

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}). \tag{11}$$

Let us assume further that the body is linearly elastic, i. e. that

$$\sigma_{ij} = C_{ijhk}\varepsilon_{hk}\,, \tag{12}$$

where $C = \{C_{ijhk}\}$, $i, j, h, k = 1, 2, 3$, is the elasticity tensor which obeys to the well-known symmetry and ellipticity conditions. The energy bilinear form of linear elasticity is further denoted by $\alpha\,(u, v) = \int_\Omega C_{ijhk}\varepsilon_{ij}(u)\varepsilon_{hk}(v)d\Omega$.

Variational Equality

For example, let us assume first that on the boundary Γ of the structure the classical boundary conditions $S_N = 0$ and $u_{T_i} = 0$, $i = 1, 2, 3$, hold. Then one gets the classical

variational equality: Find $u \in V_0 = \{v\colon v \in V, v_{T_i} = 0$ on $\Gamma\}$ such that

$$\alpha(u, v) = \int_\Omega f_i v_i \, d\Omega \,, \quad \forall v \in V_0 \,.$$

Convex Variational Inequality

Furthermore, let us assume now that on Γ the general monotone multivalued subdifferential boundary condition (1) holds. Using (2) and (9) one obtains the following variational inequality: find $u \in V$ with $\Phi(u) < \infty$, such that

$$\alpha(u, v-u) + \int_\Gamma (\Phi(v) - \Phi(u)) \, d\Gamma \geq \int_\Omega f_i(v_i - u_i) \, d\Omega \,,$$
$$\forall v \in V : \; j(v) < \infty \,.$$

QD Laws and Systems of Variational Inequalities

Let us assume that on Γ the nonmonotone, possibly multivalued boundary condition (5) holds, where Φ is a quasidifferentiable functional. It has the form

$$-S = w_1 + w_2 \,,$$

with $\{w_1, w_2\} \in Dj(u) = \{\underline{\partial}\Phi(u), \overline{\partial}\Phi(u)\}$.

Then one has, by definition, the relation (6), where $\Phi'(u, v) = \langle -S, v\rangle$. Finally, by an analogous way, one has the variational problem: find $u \in V$, $w_1, w_2 \in W$ such as to satisfy the relation

$$\begin{aligned} &\alpha(u, v-u) - \int_\Omega f_i(v_i - u_i) \, d\Omega \\ &+ \max_{\substack{w_1^*(x) \in \underline{\partial}\Phi(u(x)) \\ \text{a.e. on } \Gamma}} \langle w_1^*, v-u\rangle \\ &+ \min_{\substack{w_2^*(x) \in \overline{\partial}\Phi(u(x)) \\ \text{a.e. on } \Gamma}} \langle w_2^*, v-u\rangle = 0 \,, \\ &\forall v \in V \,. \end{aligned} \tag{13}$$

The function spaces V and W depend on the studied application. For instance, for three-dimensional elastostatics the following choice has been proposed in [4]: $V = [H^1(\Omega)]^3$, $W = [L^2(\Gamma)]^3$. A more general formulation, also proposed in the previously given original publication, would be to assume that $w_1, w_2 \in [H^{-1/2}(\Gamma)]^3$.

Then in the left-hand side of (13) on should replace $w_1(x) \in \overline{\partial}\Phi(u(x))$ a.e. on Γ by $w_1 \in \overline{\partial}F(u)$ and $w_2(x) \in \underline{\partial}\,\Phi(u(x))$ a.e. on Γ by $w_2 \in \underline{\partial}F(u)$, where one assumes that

$$F(u) = \begin{cases} \displaystyle\int_\Gamma \Phi(u(x)) \, d\Gamma & \text{if } \Phi(\cdot) \in L^2(\Gamma), \\ \infty & \text{otherwise.} \end{cases}$$

Then instead of (13) one has the following problem: find $u \in [H^1(\Omega)]^3$, $w_1, w_2 \in [H^{-1/2}(\Gamma)]^3$ such that

$$\begin{aligned} &\alpha(u, v-u) - \int_\Omega f_i(v_i - u_i) \, d\Omega \\ &+ \max_{w_1^*} \{\langle w_1^* + w_2^*, v-u\rangle : \; w_1^* \in \underline{\partial}F(u)\} \\ &+ \min_{w_2^*} \{\langle w_1^* + w_2^*, v-u\rangle : \; w_1^* \in \underline{\partial}F(u)\} = 0 \,, \\ &\forall v \in [H^1(\Omega)]^3 \,. \end{aligned}$$

One should mention that the related questions concerning the extension of *QD*-superpotentials to function spaces remain still open.

Moreover we can write the min-max form which reads: find $u \in [H^1(\Omega)]^3$ such as to satisfy the relation:

$$\begin{aligned} &\alpha(u, v-u) - \int_\Omega f_i(v_i - u_i) \, d\Omega \\ &+ \min_{w_2^* \in \underline{\partial}F(u)} \max_{w_1^* \in \overline{\partial}F(u)} \{\langle w_1^* + w_2^*, v-u\rangle = 0 \,, \\ &\forall v \in [H^1(\Omega)]^3 \,. \end{aligned}$$

If in particular the superpotential F can be expressed as the difference of two convex functions, i. e. if $F = \Phi_1 - \Phi_2$, with Φ_1 and Φ_2 convex, then one has

$$\underline{\partial}F = \partial\Phi_1 \,, \quad \overline{\partial}F = -\partial\Phi_2 \,,$$

where $\partial$is the subdifferential of the convex analysis. In this case the following system of variational inequalities results, as it can easily be shown by using the definition of the subdifferential: find $u \in [H^1(\Omega)]^3$, such as to satisfy

$$\begin{aligned} &\alpha(u, v-u) - \int_\Omega f_i(v_i - u_i) \, d\Omega \\ &\quad - \langle w_2^*, v-u\rangle + \Phi_1(v) - \Phi_1(u) \geq 0 \,, \\ &\quad \forall v \in [H^1(\Omega)]^3 \end{aligned}$$

for all $w_2^\star \in [H^{-1/2}(\Gamma)]^3$ such that

$$\langle w_2^*, v-u\rangle \leq \Phi_2(v) - \Phi_2(u), \qquad \forall v \in [H^1(\Omega)]^3 \,.$$

Further information on variational formulations in elastostatics can be found in ► Hemivariational inequalities: Applications in mechanics.

See also

- ► Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ► Hemivariational Inequalities: Eigenvalue Problems
- ► Hemivariational Inequalities: Static Problems
- ► Nonconvex Energy Functions: Hemivariational Inequalities
- ► Nonconvex-Nonsmooth Calculus of Variations
- ► Quasidifferentiable Optimization
- ► Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ► Quasidifferentiable Optimization: Algorithms for QD Functions
- ► Quasidifferentiable Optimization: Applications
- ► Quasidifferentiable Optimization: Applications to Thermoelasticity
- ► Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ► Quasidifferentiable Optimization: Codifferentiable Functions
- ► Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ► Quasidifferentiable Optimization: Exact Penalty Methods
- ► Quasidifferentiable Optimization: Optimality Conditions
- ► Quasidifferentiable Optimization: Stability of Dynamic Systems
- ► Quasivariational Inequalities
- ► Sensitivity Analysis of Variational Inequality Problems
- ► Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ► Variational Inequalities
- ► Variational Inequalities: F. E. Approach
- ► Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ► Variational Inequalities: Projected Dynamical System
- ► Variational Principles

References

1. Argyris JH, Kelsey S (1960) Energy theorems and structural analysis. Butterworths, London
2. Auchmuty G (1989) Duality algorithms for nonconvex variational principles. Numer Funct Anal Optim 10:211–264
3. Dautray R, Lions J-L (1998) Functional and variational methods. Math Anal and Numer Methods for Sci and Tech, vol 2. Springer, Berlin
4. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
5. Ekeland I, Temam R (1976) Convex analysis and variational problems. North-Holland, Amsterdam
6. Ferris MC, Pang JS (1997) Engineering and economic applications of complementarity problems. SIAM Rev 39(4):669–713
7. Lippmann H (1960) Extremum and variational principles in mechanics. CISM Courses and Lectures, vol 54. Springer, Berlin
8. Oden JT, Reddy JN (1982) Variational methods in theoretical mechanics. Springer, Berlin
9. Panagiotopoulos PD (1985) Inequality problems in mechanics and applications. Convex and nonconvex energy functions. Birkhäuser, Basel
10. Panagiotopoulos PD (1988) Nonconvex superpotentials and hemivariational inequalities. Quasidiferentiability in mechanics. In: Moreau JJ, Panagiotopoulos PD (eds) Nonsmooth Mechanics and Applications. CISM Lecture Notes. Springer, Berlin, pp 83–176
11. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Berlin
12. Panagiotopoulos PD, Stavroulakis GE (1992) New types of variational principles based on the notion of quasidifferentiability. Acta Mechanics 94:171–194
13. Stuart CA, Toland JF (1980) A variational method for boundary value problems with discontinuous nonlinearities. J London Math Soc (2) 21:319–328
14. Washizu K (1968) Variational methods in elasticity and plasticity. Pergamon, Oxford

Quasivariational Inequalities

GEORGIOS E. STAVROULAKIS
Carolo Wilhelmina Techn. University,
Braunschweig, Germany

MSC2000: 49J40, 70-08, 49Q10, 74K99, 74Pxx

Article Outline

Keywords
Variational Inequalities
Implicit Variational Inequalities and Quasivariational Inequalities
Mechanical Example:
Coupled Unilateral Contact Problem with Friction
Signorini–Coulomb Unilateral Frictional Contact
Numerical Algorithms: Applications
See also
References

Keywords

Variational inequalities; Nonsmooth mechanics; Implicit variational problems

Variational or weak formulations of boundary value problems in science and, in particular, in engineering are integral, energetic expressions of all involved quantities (involving differential equations and boundary conditions). Usually, under differentiability (smoothness) assumptions of the involved variables and equalities throughout the considered model one gets variational equality problems. The strong formulation of the initial problem (i. e., constitutive relations, boundary conditions, etc) can be reconstructed if one considers certain values for the (otherwise arbitrary) variations in the weak form, i. e., in the variational equality. Variational formulations provide the basis for modern computational mechanics techniques (e. g., the *finite element method*) and for this reason they have been extensively studied in the affiliated literature (see, among others, [22]). In terms of optimization they can be considered as stationary point statements for the total differential of an appropriately constructed (convex or nonconvex) potential energy function, provided that the studied problem admits a potential. Namely, the weak formulation expresses the fact that the variation of a function for every small variation of the involved independent variables is equal to zero, which, due to the arbitrariness of the variations, is equivalent to the more classical requirement that the first derivative of the function vanishes at a critical point.

Due to inequality-type constraints or due to lack of differentiability in the involved functions one is sometimes obliged to consider one-sided (unilateral) variations of the problem's variables. A systematic way of doing so is provided by the theory of *variational inequalities* [13]. They are related to monotone operators, to convex, nondifferentiable optimization problems and to complementarity problems. Variational inequalities have been applied for the study of problems in engineering [17,20], economics, transportation planning and flow in networks (see also [6,8,10]).

Extensions for nonconvex variational inequalities, which are based on the generalized gradient approach in the sense of F.H. Clarke, have been proposed and studied by P.D. Panagiotopoulos who named them *hemivariational inequalities*. Details are given in ► Nonconvex energy functions: Hemivariational inequalities and ► Hemivariational inequalities: Applications in mechanics. Parallel developments which are based on the notion of the quasidifferentiability in the sense of V.F. Demyanov and A.M. Rubinov are described in ► Quasidifferentiable optimization: Variational formulations.

Furthermore, there exist problems where the admissible space (for the variables and their variations) or the involved potentials depend on the solution of the problem. This class of implicit variational inequality problems are called quasivariational inequalities. They have been used for the modeling of stochastic impulsive control problems, in free boundary problems, in mechanics and in economy. The interested reader may find more information in the references [1,2,7,9,15]. Here a short outline of quasivariational inequality problems is given. A model application arising in unilateral contact problems with Coulomb friction in engineering mechanics demonstrates the discussed ideas. This approach is based on early theoretical and numerical studied of [19] (see also numerical applications in [11,12]) and, among others, have recently been tested for several convex and nonconvex problems of mechanics in [14].

Variational Inequalities

Let us first consider abstract variational formulations of a boundary value problem which is defined in a subset Ω of $\mathbf{R}^n$, $n = 1, \ldots, n$, with boundary Γ. Let V be a real Hilbert space and V' be its dual space. Let $a(\cdot, \cdot)$: $V \times V \to \mathbf{R}$ be a symmetric, continuous and coercive bilinear form and $(l, \cdot)$ be a continuous linear form on V. An

abstract variational problem reads: find $u \in V$ such that

$$a(u, v-u) = (l, v-u)\,, \quad \forall v \in V\,. \tag{1}$$

Let moreover K be a closed convex subset of V and assume that a solution of the boundary value problem within the set K is sought. It can be shown that this solution is characterized by the following abstract variational inequality (of the G. Fichera type, see [20, p. 188]):

$$\begin{cases} \text{Find} & u \in K \subset V \\ \text{s.t.} & a(u, v-u) \geq (l, v-u), \\ & \forall v \in K. \end{cases} \tag{2}$$

For a convex, l.s.c. proper functional Φ on V one may define the more general (nonlinear) variational inequality ([14]):

$$\begin{cases} \text{Find} & u \in V \\ \text{s.t.} & \alpha(u, v) + \Phi(v) - \Phi(u) \geq \\ & \quad \geq (l, v-u), \\ & \forall v \in V. \end{cases} \tag{3}$$

It is obvious that (2) is a special case of (3), with $\Phi = I_K$, where the *indicator function* of the set K is defined by $I_K(v) = 0$ if $v \in K$, $+\infty$ otherwise.

Let moreover j: $\mathbf{R} \to \mathbf{R}$ denotes a locally Lipschitz function and let $j^0(u, v{-}u)$ denotes the generalized gradient of the nonconvex and nonsmooth function j. By definition, one has the following connection with the generalized gradient, in the sense of Clarke:

$$j^0(u, v) = \{\max \langle w, v\rangle : \ w \in \partial_{\mathrm{CL}} j(u)\}\ . \tag{4}$$

A hemivariational inequality problem reads:

$$\begin{cases} \text{find} & u \in V \\ \text{s.t.} & a(u, v-u) + \displaystyle\int_\Omega j^0(u, v-u)\, d\Omega \\ & \quad \geq (l, v-u), \\ & \forall v \in V. \end{cases} \tag{5}$$

Implicit Variational Inequalities and Quasivariational Inequalities

If one assumes for instance that the linear form $(l, \cdot)$ or the set K in the previous relations depend on the solution u, one gets various types of implicit variational inequalities or quasivariational inequalities.

Let the set K be a variable of the solution u. Then from (2) one gets the quasivariational inequality:

$$\begin{cases} \text{find} & u \in K(u) \subset V \\ \text{s.t.} & \alpha(u, v) \geq (l, v-u), \\ & \forall v \in K(u). \end{cases}$$

Along the same lines one formulates from (3) the implicit variational inequality:

$$\begin{cases} \text{find} & u \in V \\ \text{s.t.} & \alpha(u, v) + \Phi(u; v) - \Phi(u; u) \\ & \quad \geq (l, v-u), \\ & \forall v \in V. \end{cases}$$

Here the first argument in $\Phi(\cdot, \cdot)$ is tackled as a parameter. A concrete application of this method will be demonstrated by the mechanical problem in the next section.

Finally, in analogy to the previous extensions, for a continuous mapping $h(u)$ the following quasihemivariational inequality (which may also be characterized as implicit hemivariational inequality) problem can be written (see [16, p. 128]):

$$\begin{cases} \text{find} & u \in V \\ \text{s.t.} & a(u, v-u) + h(u) j^0(u, v-u) d \\ & \quad \geq l(v-u), \\ & \forall v \in V. \end{cases} \tag{6}$$

Mechanical Example: Coupled Unilateral Contact Problem with Friction

Let $\Omega \in \mathbf{R}^3$ be an open bounded subset occupied by a deformable body in its undeformed state. On the assumption of small deformations one writes the virtual work relation (for $u \in V$)

$$\begin{aligned} \int_\Omega \sigma_{ij}\varepsilon_{ij}(v-u)\, d\Omega = &\int_\Omega f_i(v_i - u_i)\, d\Omega \\ &+ \int_\Gamma S_N(v_N - u_N)\, d\Gamma \\ &+ \int_\Gamma S_{T_i}(v_{T_i} - u_{T_i})\, d\Gamma\ , \\ \forall v \in V\,. & \end{aligned} \tag{7}$$

Here V denotes the function space of the displacements, which in general is an appropriate subset of $H^1(\Omega)$ and f_i, S_N, $S_{T_i} \in L_2(\Gamma)$. Recall here that the abstract bilinear form $\alpha(\cdot, \cdot)$ reads in this case of linear elasticity

$$\alpha(u, v) = \int_\Omega C_{ijhk}\varepsilon_{ij}(u)\varepsilon_{hk}(v)\, d\Omega\,. \tag{8}$$

Moreover the underlying elastostatic equilibrium equation boundary value problem has the form:

$$\sigma_{ij,j} + f_i = 0\,, \tag{9}$$

where the f_i is the volume force vector. One recalls here the strain-displacement relation (small deformation theory):

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i})\,. \tag{10}$$

Let a linearly elastic body be assumed, i. e., the constitutive material relation reads:

$$\sigma_{ij} = C_{ijhk}\varepsilon_{hk}\,,$$

where $C = \{C_{ijhk}\}$, $i, j, h, k = 1, 2, 3$, is the elasticity tensor which satisfies the well-known symmetry and ellipticity properties.

Recall here that on the assumption that classical support conditions hold on Γ (i. e., say $u_N = 0$ and $u_{T_i} = 0$, $i = 1, 2, 3$) one gets the following variational equality:

$$\begin{cases} \text{Find} \quad u \in V_0 = \left\{ v \in V\colon \begin{array}{r} v_N = 0, \\ v_{T_i} = 0 \\ \text{on } \Gamma \end{array} \right\} \\ \text{s.t.} \quad \alpha(u, v) = \displaystyle\int_\Omega f_i v_i\, d\Omega, \\ \quad\ \forall v \in V_0. \end{cases} \tag{11}$$

Signorini–Coulomb Unilateral Frictional Contact

Let us assume the pointwise unilateral contact relations (known as *Signorini condition*, for the frictionless unilateral contact case):

$$\begin{aligned} &-S_N \geq 0\,, \quad u_N - g \leq 0\,, \\ &-S_N(u_N - g) = 0 \quad \text{on } \Gamma\,. \end{aligned} \tag{12}$$

Here, the inequalities on the boundary tractions correspond to the mechanical restriction that no tensile tractions are permitted. Moreover, the normal boundary displacements should not be greater that a given initial distance g, because no penetration is allowed. Finally, the complementarity relation expresses the physical fact that either contact is realized or a separation takes place.

A simplified static version of the Coulomb's friction law connects the tangential (frictional) forces S_{T_i} with the normal (contact) forces S_N by the relation

$$\gamma = \mu\,|S_N| - |S_T| \geq 0\,. \tag{13}$$

Here $|*|$ denotes the norm in $\mathbf{R}^3$ and μ is the friction coefficient. The friction mechanism is considered to work in the following way: If $|S_T| < \mu|S_N|$ (i. e. $\gamma > 0$) the slipping value γ must be equal to zero and if $|S_T| = \mu|S_N|$ (i. e. $\gamma = 0$) then we have slipping in the opposite direction of S_T. Explicitly we have:

$$\begin{cases} \text{if } \gamma > 0 \quad \text{then } y_T = 0, \\ \text{if } \gamma = 0 \quad \text{then there exists } \sigma > 0 \\ \qquad\qquad \text{s.t. } y_{Ti} = -\sigma S_{Ti}, \end{cases} \tag{14}$$

where $i = 1, 2, 3$ refers to the components of vector S_T with respect to a reference Cartesian coordinate system.

Contact law (12) can be written in the superpotential form:

$$-S_N \in \partial \mathbf{I}_{U_{\text{ad}}}(u_N) = \partial \Phi_N(u) = \mathcal{N}_{U_{\text{ad}}}(u_N)\,. \tag{15}$$

Here the set of admissible displacements is introduced:

$$U_{\text{ad}} = \{u \in V\colon\ u_N - g \leq 0\} \tag{16}$$

and the notions of the convex analysis subdifferential and of the normal cone to a set have been used. The corresponding variational inequality reads

$$-S_N(u_N)(v_N - u_N) \leq 0\,, \quad \forall v_N \in U_{\text{ad}}\,. \tag{17}$$

For the friction law one writes, analogously:

$$-S_T \in \partial_{u_N}(\mu\,|S_N|\,|u_N|) = \partial \Phi_T(S_N; u_T)\,, \tag{18}$$

where the involved potential is nondifferentiable (due to the absolute value nonlinearity of $|u_N|$) and depends on the normal contact traction S_N, thus, implicitly, on the solution of the problem u, i. e. one considers the parametrized potential $\Phi_T(u;u_T) = \Phi_T(S_N;u_T)$

$= \mu|S_N||u_N|$. The corresponding variational inequality reads

$$-S_T(u_T)(v_T - u_T) \le \Phi_T(u; v_T) - \Phi_T(u; u_T) , \quad \forall v_T \in U_{\text{ad}} . \tag{19}$$

Combining relations (7), (9) and (15) one gets the implicit variational inequality: find $u \in U_{ad}$ such that

$$\int_\Omega \sigma_{ij}\varepsilon_{ij}(v - u)\, d\Omega + \Phi_T(u; v_T) - \Phi_T(u; u_T) \ge \int_\Omega f_i(v_i - u_i)\, d\Omega , \quad \forall v \in U_{\text{ad}} . \tag{20}$$

A dual problem in terms of stresses provides us a corresponding quasivariational inequality problem. For simplicity, a two-dimensional problem is considered further. Moreover, the following set of admissible boundary tractions for the Signorini–Coulomb unilateral contact problem is assumed:

$$S_{\text{ad}} = \{(S_N, S_T):\ g_j(S_N, S_T) \le 0,\ j = 1, 2\} ,$$

where the constraint functions have the form:

$$g_1(S_N, S_T) = \mu S_N - S_T ,$$
$$g_1(S_N, S_T) = \mu S_N + S_T .$$

Moreover one needs the set of admissible stresses (which include the boundary tractions): $\Sigma(\sigma) = \{\sigma: \sigma_{ij,j} + f_i = 0\} \cap S_{ad}$. It may be shown that in this case the previous problem is expressed in the form of the quasivariational inequality:

$$\begin{cases} \text{find} & \sigma \in \Sigma(\sigma) \\ \text{s.t.} & \displaystyle\int_\Omega \varepsilon_{ij}(\tau_{ij} - \sigma_{ij})\, d\Omega \ge 0, \\ & \forall \tau \in \Sigma(\sigma) \end{cases}$$

Numerical Algorithms: Applications

Theoretical results and numerical algorithms can be found in several books dealing with variational inequalities, convex analysis and their applications. For the numerical solution, usually one solves, iteratively, a number of variational inequality problems. The resulting series approximates the solution of the initial quasivariational inequality. Further information and references, mainly connected with the mechanical problems used as model applications in this paper and their generalizations, can be found in [3,5,14,18,21]. Finally iterative solution methods can be based on multilevel optimization techniques, as it is discussed in ▶ Multilevel optimization in mechanics.

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-Nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness

- ► Variational Inequalities: Projected Dynamical System
- ► Variational Principles

References

1. Baiocchi C, Capelo A (1984) Variational and quasivariational inequalities. Applications to free boundary problems. Wiley, New York
2. Bensoussan A, Lions J-L (1984) Impulse control and quasivariational inequalities. Gauthier-Villars
3. Bisbos CC (1995) A competitive game algorithm with five players for unilateral contact problems including the rotational and the thermal degrees of freedom. In: Raous M, Jean M, Moreau JJ (eds) Contact Mechanics. Plenum, New York, pp 251–258
4. Brézis H (1972) Problèmes unilatéraux. J Math Pures Appl 51:1–168
5. Curnier A, He Q-C, Telega JJ (1992) Formulation of unilateral contact between two elastic bodies undergoing finite deformations. CR Acad Sc Paris Ser II-1, p 314
6. Ferris MC, Pang JS (1997) Engineering and economic applications of complementarity problems. SIAM Rev 39(4):669–713
7. Friedman A and Spruck J (eds) (1993) Variational and free boundary problems. IMA vol Math Appl. Springer, Berlin
8. Friesz TL, Bernstein DH, Strough R (1996) Dynamic systems, variational inequalities and control theoretic models for predicting time-varying urban network flows. Transport Sci 30:14–31
9. Harker PT (1991) Generalized Nash games and quasivariational inequalities. Europ J Oper Res 54:81–94
10. Harker PT, Pang JS (1990) Finite dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications. Math Program 48:161–220
11. Kalker JJ (1988) Contact mechanical algorithms. Comm Appl Numer Meth 4:25–32
12. Kalker JJ (1990) Three-dimensional elastic bodies in rolling contact. Kluwer, Dordrecht
13. Kinderlehrer D, Stampaccia G (1980) An introduction to variational inequalities and their application. Acad. Press, New York
14. Mistakidis ES, Stavroulakis GE (1998) Nonconvex Optimization in Mechanics. Algorithms, heuristics and engineering applications by the F.E.M. Kluwer, Dordrecht
15. Mosco V (1976) Implicit variational problems and quasivariational inequalities. In: Nonlinear operators and the calculus of variations. Lecture Notes Math, vol 543. Springer, Berlin, pp 83–156
16. Naniewicz Z, Panagiotopoulos PD (1995) Mathematical theory of hemivariational inequalities and applications. M. Dekker, New York
17. Oden JT, Kikuchi N (1988) Contact problems in elasticity: A study of variational inequalities and finite element methods. SIAM, Philadelphia
18. Outrata J, Kočvara M, Zowe J (1998) Nonsmooth approach to optimization problems with equilibrium constraints. Kluwer, Dordrecht
19. Panagiotopoulos PD (1975) A nonlinear programming approach to the unilateral contact and friction boundary value problem in the theory of elasticity. Ingen Archiv 44:421–432
20. Panagiotopoulos PD (1985) Inequality problems in mechanics and applications. Convex and nonconvex energy functions. Birkhäuser, Basel
21. Telega JJ (1988) Topics on unilateral contact problems of elasticity and inelasticity. In: Moreau JJ, Panagiotopoulos PD (eds) Nonsmooth Mechanics and Applications. CISM Courses and Lectures, vol 302. Springer, Berlin, pp 341–462
22. Washizu K (1968) Variational methods in elasticity and plasticity. Pergamon, Oxford

R

Railroad Crew Scheduling

Arvind Kumar[1],
Balachandran Vaidyanathan[2],
Krishna C. Jha[1], Ravindra K. Ahuja[2]
[1] Innovative Scheduling, Inc., GTEC, Gainesville, USA
[2] Department of Industrial & Systems Engineering, University of Florida, Gainesville, USA

Article Outline

Keywords
Introduction
Definitions
  Terminology
  Regulatory and Contractual Requirements
  Problem Inputs
Formulation
  Space-Time Network
  Integer Programming Formulation
Methods and Applications
  Successive Constraint Generation (SCG)
  Quadratic Cost-Perturbation (QCP) Algorithm
  Applications
Conclusions
References

Keywords

Crew scheduling; Multi-commodity flows; Network flows; Space-time network; Mixed integer programming; Railroad; Decision support; Decomposition; Relaxation technique

Introduction

Crew scheduling problems (CSPs) consist of assigning crews to trains and creating rosters for each crew, while satisfying several Federal Railway Administration (FRA) regulations and trade-union work rules. The objectives are to minimize the cost of operating trains on one hand and to improve the quality of life for crew on the other hand. This chapter gives a comprehensive description of the existing literature related to crew scheduling, an overview of CSP for the North American Railroads and describes an application of space-time network flow based multi-commodity methods [13] to solve this problem. Network flow models have found successful applications in a large number of diverse fields which include applied mathematics, computer science, engineering, management, and operations research [1].

Crew scheduling has been historically associated with airlines and mass-transit companies. Several papers on crew schedule management have appeared in the past literature; most notable among these are due to [4,6,8,15,16]. All these articles explore a set covering based approach to solve the crew scheduling problem. Crew scheduling is conventionally divided into two stages: (1) *Crew pairing*: A crew pairing is a sequence of connected segments that start and end at the same crew base and satisfy all legality constraints. The objective is to find the minimum-cost set of crew pairings such that each flight or train segment is covered. (2) *Crew rostering*: The objective here is to assign individual crew members to trips or sequences of crew pairings. This pairing and rostering approach uses a set covering formulation and is usually solved using column generation embedded in a branch-and-bound framework (also called branch-and-price [10]). This approach has gained wide acceptance and application in both the airline industry [2,3,11] and in European [5], Asian [7], and Australian railroads [9,14].

While there have been several papers devoted to the study of railroad crew scheduling problems in Europe, Asia, and Australia, North American railroad problems are yet to be addressed satisfactorily. One application of optimization methods to North American railroad crew scheduling is due to [12]. They studied crew balancing in the context of a major North American Railroad, BNSF Railway and developed a dynamic programming approach to solve this problem. The major short-coming of their approach is that they did not consider the possibility of different crew types; each governed by a different set of rules. Another drawback is that their approach could handle only a particular crew district configuration (*single-ended* crew district). While most crew districts in North America are *single-ended*, there are several which are *double-ended* or even more complex (these configurations are described in the next section). The multi-commodity network flow approach described here models all the rules considered by [12] and also handles the case where different crew pools have different sets of rules. It is also applicable to all the crew district configurations encountered in North America.

Crew pairing and rostering approaches which use column generation have been the predominantly successful methods to solve crew scheduling problems. However, this approach is not suited for North American railroads due to the following reasons:

1. The rail network of North American railroad is divided into several *crew districts*. As a train follows its route, it goes from one crew district to another, picking up and dropping off crew at crew change terminals. Almost all crew districts consist of two or three terminals. Hence, a pairing and rostering approach is needlessly complex and not required since most pairings would consist of two trains, a train from home to away and a train from away to home. In addition, rail networks typically consist of 200–300 crew districts and the emphasis is on an approach which is simple and fast, and column generation techniques which are computationally intensive are not appropriate.
2. The FRA regulations governing North American railroads are extremely complex. The most complicating of these rules is *First-In-First-Out* (FIFO) requirement. FIFO constraints require that crews should be called on duty in the order in which they become qualified for assignment at a location. The success of all approaches using column generation or branch-and-price algorithms is contingent on the ease of solving the sub-problem. It should be noted that the addition of the FIFO side constraints to the problem would spoil the special structure of the sub-problem and blow up the computational times.

Henceforth, whenever CSP is mentioned, it is mentioned in the context of the North American railroad CSP.

Definitions

This section gives an overview of the CSP and defines some of the essential terminology needed to understand the problem. It also gives an overview of some of the typical regulations which govern crew management and lists the set of inputs required to properly define and formulate the crew scheduling problem.

Terminology

Crew District: The rail network of a railroad is divided into *crew districts* that constitute a subset of terminals (nodes). Each crew district is typically a geographic corridor over which trains can travel with one crew. A typical railroad network for a major railroad in the U.S. may be divided into as many as 200 to 300 crew districts. As a train follows its route, it goes from one crew district to another, picking up and dropping off crew at crew change terminals.

Crew Pools: Within a crew district, there are several types of crews called *crew pools* or *crew types*, which may be governed by different trade-union rules and regulations. For example, a crew pool may have preference for the trains operated in a pre-specified time window. Similarly, a crew pool consisting of senior crew personnel is assigned only to pre-designated trains so that crews in that pool know their working hours ahead of time.

Home and Away Terminals: The terminals where crews from a crew pool change trains are designated either as *home terminals* or *away terminals*. The railroad does not incur any lodging cost when a crew is at its home terminal. However, the railroad has to make arrangements for crew accommodation at their away ter-

minals. A crew district with one home terminal and one away terminal is called a *single-ended crew district*. The other type of crew district is a *double-ended crew district,* in which more than one terminal is a home terminal for different crew pools. Some of the other crew district configurations are crew districts with one home terminal and several away terminals, and crew districts with several home terminals and corresponding sets of away terminals.

Crew Detention: Once a crew reaches its away terminal and rests for the prescribed hours, the crew is ready to head back to its home terminal. However, if there is no train scheduled to depart, then the crew may have to wait in a hotel. According to the trade-union rules, once a crew is at the away terminal for more than a pre-specified number of hours (generally 16 hours), the crew earns wages (called *detention costs*) without being on duty.

Crew Deadheading: *Crew deadheading* refers to the repositioning of crew between terminals. A crew normally operates a train from its home terminal to an away terminal, rests for a designated time, and then operates another train back to its home terminal. Sometimes, at the away terminal, there is no return train projected for some time, or there is a shortage of crews at another terminal. Thus, instead of waiting for train assignment at its current terminal, the crew can take a taxicab or a train (as a passenger) and deadhead to the home terminal. Similarly, the crew may also deadhead from a home terminal to an away terminal in order to rebalance and better match the train demand patterns and avoid train delays.

On-duty and Tie-up Time: Whenever a crew is assigned to a train, it performs some tasks to prepare the train for departure, and hence crews are called on-duty before train departure time. The time at which the crew has to report for duty is called the *on-duty time.* Similarly, a crew performs some tasks after the arrival of the train at its destination, and hence crews are released from duty after the train arrival. The time at which the crew is released from duty is called *tie-up time*. The duty duration before train departure is referred to as *duty-before-departure* and the duty duration after train arrival as *duty-after-arrival.* Hence, the total duty time (or *duty-period*) of a crew assigned to a train is the sum of the *duty-before-departure*, the *duty-after-arrival,* and the travel time of the train.

Duty Period: In most cases, duty-period of a crew assigned to a train is the total duration between the *on-duty time* and the *tie-up time*. In some cases when a crew rests for a very short time at an away location before getting assigned to a train, the rest time and the duration of the second train may also included in the duty period of the crew.

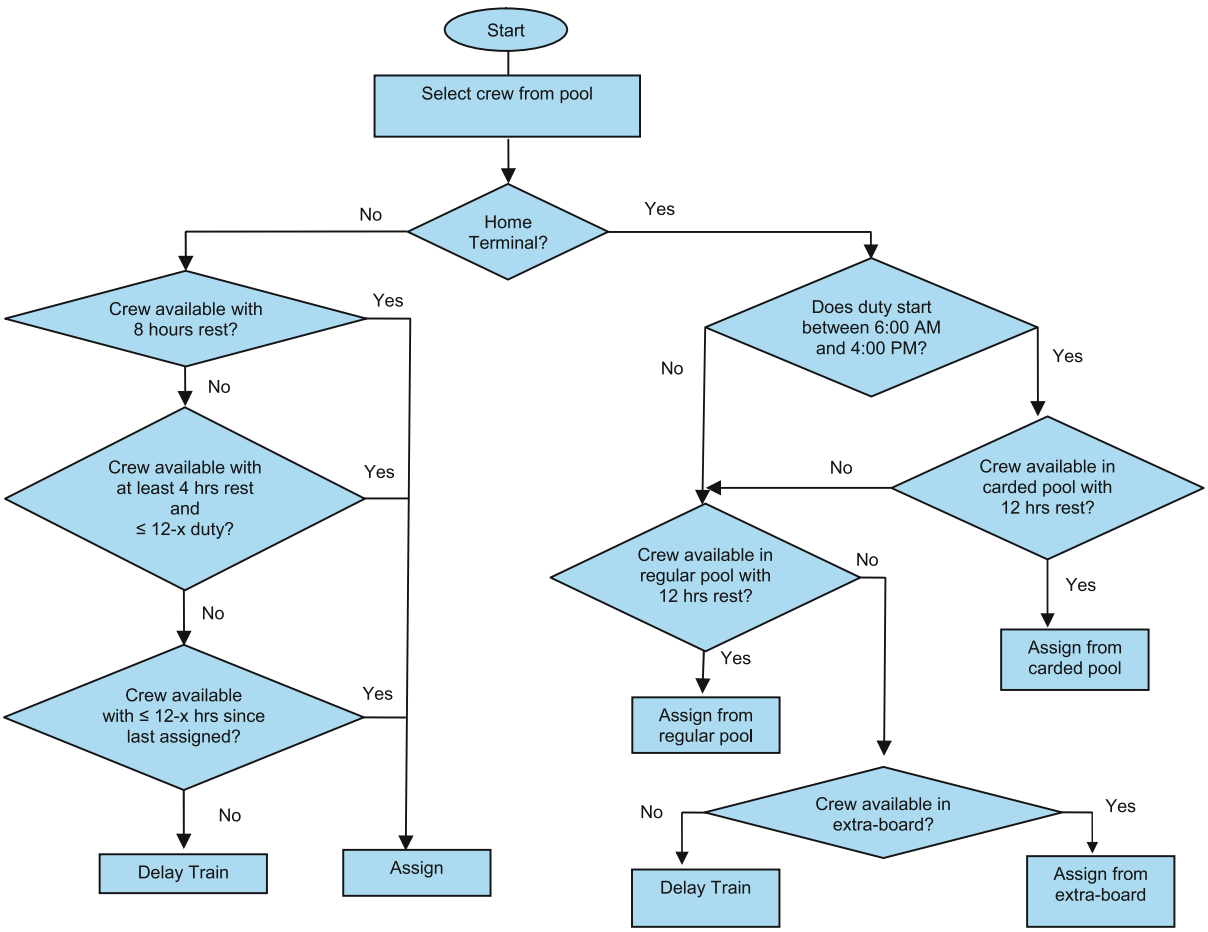
Dead Crews: By federal law, a train crew can only be on duty for a maximum of 12 consecutive hours, at which time the crew must cease all work and it becomes *dead or dog-lawed*.

Train Delays: When a train reaches a crew-change location and there is no available crew qualified to operate this train, the train must be delayed. Train delays due to crew unavailability are quite common among railroads. These delays are very expensive (some estimate $1000 per hour) and can be reduced significantly through better crew scheduling and train scheduling.

Regulatory and Contractual Requirements

Assignment of crews to trains is governed by a variety of Federal Railway Administration (FRA) regulations and trade-union rules. These regulations range from the simple to the complex. The regulations also vary from district to district and from crew pool to crew pool. Some examples of these kinds of constraints and their typical parameter values:

- Duty-period of a crew cannot exceed 12 hours. Duty-period of a crew on a train is usually calculated as the time interval between the on-duty time and tie-up time of the train.
- When a crew is released from duty at the home terminal or has been deadheaded to the home terminal, they can resume duty only after 12 hours (10 hours rest followed by 2 hours call period) if duty-period is greater than 10 hours, and after 10 hours (8 hours rest followed by 2 hours call period) if duty-period is less than or equal to 10 hours.
- Whenever a crew is released from duty at the away



Railroad Crew Scheduling, Figure 1
An example of crew assignment decision tree

terminal, they usually must go for a minimum 8 hours rest, except for a few exceptions.

- Crews belonging to certain pools must be assigned to trains in a FIFO order.
- A train can only be operated by crews belonging to pre-specified pools.
- Every train must be operated by a single crew.
- Crews are guaranteed a certain minimum pay per month regardless of whether or not they work.

Figure 1 gives an example of the kind of decision process that needs to be followed by crew planners.

Problem Inputs

The inputs for the mathematical formulation of the crew scheduling problem are:

- **Train Schedule:** The train schedule contains information about the departure time, arrival time, on-duty time, tie-up time, departure location, and arrival location for every train in each crew district it passes through.
- **Crew Pool Attributes:** This includes attributes of various crew types, namely their home locations, their away locations, minimum rest time, train preferences, etc.
- **Crew Initial Position:** This provides the position of crew at the beginning of the planning horizon. This includes information of the terminal at which a crew is released from duty, the time of release, the number of hours on duty in the previous assignment, and the crew pool the crew belongs to.
- **Train-Pool Preferences:** The train-pool prefer-

ences, if any, contain information about the set of trains that can be operated by a crew pool.

- **Away Terminal Attributes:** This consists of information about the away terminals for each crew pool. It includes the rest rules and the detention rules for each crew pool at each corresponding away terminal.
- **Deadhead Attributes:** This consists of the time taken to travel by taxi between two terminals in a crew district.
- **Cost parameters:** Cost parameters are used to set up the objective function for the crew scheduling problem. They consist of crew wage per hour, deadhead cost per hour, detention cost per hour, and train delay cost per hour.

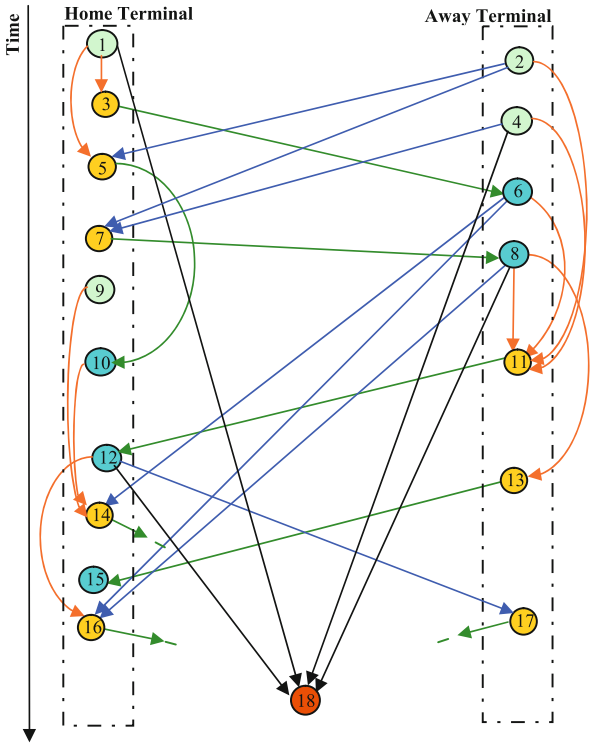
Formulation

The CSP is formulated as a multi-commodity network flow problem on a space-time network. The construction of the space-time network is described first and then the formulation of the CSP as an integer programming problem is given.

Space-Time Network

The CSP is decomposed into a separate problem for each crew district. In the space-time network, each node corresponds to a crew event and has two defining attributes: location and time. The events that are modeled while constructing the space-time network are departure of trains, arrival of trains, departure of deadheads, arrival of deadheads, supply of crew, and termination of crew duty to mark the end of the planning horizon. Figure 2 presents an example of the space-time network in a crew district. Note that for the sake of clarity, this network only represents a subset of all the arcs.

For each crew, a supply node whose time corresponds to the time at which this crew is available for assignment, and whose location corresponds to the terminal from which the crew is released for duty is created. Each supply node is assigned a supply of one unit and corresponds to a crew member. The network also has a common sink node for all crews at the end of the planning horizon. This sink has no location attribute and has the time attribute equal to the end of the planning horizon. The sink node has a demand equal to the total number of crew supplied.



Railroad Crew Scheduling, Figure 2
Space-time network for a single-ended district with a single crew type. *Node legend*: green (supply), blue (arrival), yellow (departure), red (demand) *Arc legend*: green (train), orange (rest), blue (deadhead), black (demand)

For each train (say l) passing through a crew district, a *departure node* (say l') is created at the first departing station of the train in the crew district and an arrival node (say l'') at the last arriving station of the train in the crew district. Each arrival or departure node has two attributes: place and time. For example, *place* (l') = *departure-station* (l) and *time* (l') = *on-duty-time* (l); and similarly, *place* (l'') = *arrival-station* (l) and *time* (l'') = *tie-up-time* (l).

Train arc (l', l'') is created for each train l connecting the departure node and arrival node of train l. *Deadhead arcs* are constructed to model the travel of crew by taxi. A deadhead arc is constructed between a train arrival or crew supply node at a location and a train departure node at another location. All the deadhead arcs which satisfy the contractual rules and regulations are created. *Rest arcs* are constructed to model resting of a crew at a location. A rest arc is constructed between a train arrival node or a crew supply node at a loca-

tion and a train departure node at the same location. Rest arcs are created in conformance to the contractual rules and regulations. Since the contractual regulations are often crew pool specific, deadhead arcs and rest arcs are created specific to a crew pool. Finally, *demand arcs* are created from all train arrival nodes and crew supply nodes to the sink node. Each arc has an associated cost equivalent to the crew wages, deadhead costs, or detention costs, depending on the type of the arc. It can be noted that all contractual requirements other than the FIFO constraint are easily handled in the network construction.

The space-time network does not model the case when qualified crews are not available for assignment to a train causing train delays. Train delays are modeled by the construction of additional arcs. To do this, rest arcs and deadhead arcs which do not honor the rest regulations are constructed and flows on these arcs are penalized to ensure that flows on these arcs occur only when qualified crews are not available for assignment. The flows on these arcs denote that the train will be delayed until crew becomes qualified for train operation. However, as the delay of a train may have propagating effect in the availability of crews in subsequent assignments, it is assumed that the crew assigned to a delayed train has sufficient slack in the rest time at the train arrival node to make it qualified for subsequent assignments.

Integer Programming Formulation

The CSP is formulated as an integer multi-commodity flow problem on the space-time network described in the previous section. Each crew pool represents a commodity. Crew enters the system at crew supply nodes, takes a sequence of connected train, rest, and deadhead arcs before finally reaching the sink.

Decision Variables

x_l^c : Flow of crew pool $c \in C$ on each train arc $l \in L$

x_d : Flow on deadhead arc $d \in D$

x_r : Flow on rest arc $r \in R$

Objective function

$$\text{Min} \sum_{l \in L} \sum_{c \in C} c_l^c x_l^c + \sum_{d \in D} c_d x_d + \sum_{r \in R} c_r x_r$$

Constraints

$$\sum_{c \in C} x_l^c = 1, \text{ for all } l \in L \tag{1}$$

$$\sum_{a \in i^+} x_a = 1, \text{ for all } i \in N_s \tag{2}$$

$$\sum_{a \in N_d^-} x_a = f \tag{3}$$

$$x_l^c = \sum_{a \in tail(l)_c^-} x_a, \text{ for all } l \in L, c \in C \tag{4}$$

$$x_l^c = \sum_{a \in head(l)_c^+} x_a, \text{ for all } l \in L, c \in C \tag{5}$$

$$\sum_{r' \in A_r} x_{r'} - M(1 - x_r) \leq 0, \text{ for all } r \in R \tag{6}$$

$$x_l^c \in \{0, 1\} \text{ and integer, for all } l \in L, c \in C \tag{7}$$

$$x_d \in \{0, 1\} \text{ and integer, for all } d \in D \tag{8}$$

$$x_r \in \{0, 1\} \text{ and integer, for all } r \in R \tag{9}$$

Constraint (1) is the train cover constraint which ensures that every train is assigned a qualified crew to operate it. Constraint (2) ensures flow balance at a crew supply node. Constraint (3) ensures the flow balance at the sink node. Constraints (4) and (5) ensure flow balance at train departure and arrival nodes respectively. Constraint (6) ensures that the crew assignment honors the FIFO constraint. Constraints (7), (8), and (9) specify that all the decision variables in the model are binary. The objective function is constructed to minimize the total cost of crew wages, deadheading, detentions and train delays. Note that the detention and delay costs are taken into account while calculating the cost of rest arcs.

Theorem 1 *There is a one to one correspondence between a feasible flow on the space-time network satisfying constraints (1)–(9) and a feasible solution to the crew scheduling problem.*

Most crew districts have two terminals, and a typical train schedule has around 500 trains running in

Railroad Crew Scheduling, Table 1
Notation

N	Set of nodes in the space time network	i_c^+	Set of outgoing arcs specific to crew pool c at node i
L	Set of train arcs in the network, indexed by l	i_c^-	Set of incoming arcs specific to crew pool c at node i
D	Set of deadhead arcs in the network, indexed by d	A_r	Set of arcs on which flow will violate FIFO constraint if there is flow on rest arc r
R	Set of rest arcs in the network, indexed by r	f	Total number of available crew
A	Set of arcs in the space-time network, indexed by a	M	A very large number
$G(N, A)$	Space-time network	c_l^c	Cost of crew wages for crew pool $c \in C$on train arc $l \in L$
N_s	Set of crew supply nodes	c_d	Cost of deadhead arc $d \in D$
N_d	Sink node	c_r	Cost of rest arc $r \in R$
C	Set of crew pools in the system, indexed by c	$tail(l)$	The node from which arc l originates
i^+	Set of outgoing arcs at node i	$head(l)$	The node at which arc l terminates
i^-	Set of incoming arcs at node i		

a couple of weeks in a crew district. Each crew district could have two to four crew types and around 50 crews. Therefore, the space-time network could have around 50 + 2 x 500 = 1,050 nodes. The number of deadhead arcs is typically around 25,000, and the number of rest arcs is around 100,000.

Since the number of rest arcs for a typical problem is of the order of 100,000, and as each rest arc has one FIFO constraint, the number of FIFO constraints in the model would be 100,000, which makes the problem very large. In addition, these constraints spoil the special structure of the problem and a direct approach to solve the CSP suffers from intractability and does not converge to a feasible solution in several hours of computational time. However, the integer programming problem with FIFO constraints relaxed (*Relaxed Problem*) can be solved to optimality within minutes. Efficient methods to solve the CSP are described in the next section.

Methods and Applications

Successive Constraint Generation (SCG)

The SCG algorithm works by iteratively pruning out crew assignments which violate the FIFO constraints from the current solution of a more relaxed problem. The SCG algorithm starts with the optimal solution of the Relaxed Problem. The algorithm scans the rest arcs in the current solution with positive flow, and for each rest arc assignment which violates FIFO constraints, it adds the corresponding FIFO constraints. The problem is then re-solved and the solution re-checked for FIFO infeasibilities. This process is repeated until all FIFO infeasibilities are pruned.

Algorithm-SCG

1. Solve the Relaxed Problem. If a feasible solution exists, then proceed to Step 2. Otherwise STOP as the instance is infeasible.
2. Examine all the rest arcs with positive flow in the solution of Step 1. Add FIFO constraints to the integer program on those rest arcs assignments which violate FIFO requirements.
3. If FIFO constraints are added in Step 2, re-optimize the modified integer program and go to Step 2. Otherwise, STOP. The current solution is optimal.

The SCG algorithm is an exact algorithm guaranteeing optimal solution to the original problem. However, in the worst case, SCG could add all the FIFO constraints to the integer program and would hence become an intractable approach. Fortunately, this seldom happens in practice. Computational results show that the number of constraints added is usually much less than the total number of rest arcs in the network.

While SCG is an exact algorithm and produces provably optimal solutions, the running time of this algorithm could be quite high. Some instances had running times in the order of minutes while others had much higher running times. While these running times are acceptable in the planning environment, they restrict the applicability of this algorithm in the real-time environment.

Quadratic Cost-Perturbation (QCP) Algorithm

The QCP algorithm penalizes FIFO violations. This method guarantees zero FIFO violations in the case where there is no priority in assigning crews to trains and serves as a heuristic method for the other case when there are priority restrictions. The basic intuition behind this approach is that the costs of arcs while solving the Relaxed Problem is perturbed in a way that it guarantees FIFO compliance.

The cost perturbation strategy is presented through the illustration shown in Fig. 3 for the case when there is only one crew pool type. In case (I), crew assignments are made in a non-FIFO manner, and in case (II), the assignments are made in a FIFO manner. Consider the case when crews are detained at the Terminal 2. Then, due to the nature of detention costs, the cost of the assignment (II) would definitely be less than or equal to the cost of assignment (I), and hence the solution to the Relaxed Problem would honor FIFO constraints. On the other hand, suppose all the rest arcs had a cost of zero; then both the assignments would have the same cost, and the Relaxed Problem would have no cost incentive to choose assignment (II) over assignment (I). Thus, a solution to the Relaxed Problem may violate the FIFO constraints. In order to provide an incentive to the Relaxed Problem to choose case (II) over case (I), the cost assignments on rest arcs are perturbed so that the solution of the Relaxed Problem has assignments of type (II) and not assignments of type (I).

The cost perturbation scheme that is used is a function of the duration of rest arcs. Suppose that the time duration between events corresponding to nodes 2 and 4, 4 and 5, and 5 and 7 are a, b, and c, respectively. Consider a cost assignment which is proportional to the square of the duration of rest arcs. The constant of proportionality is represented by k (k is set to a very small value).

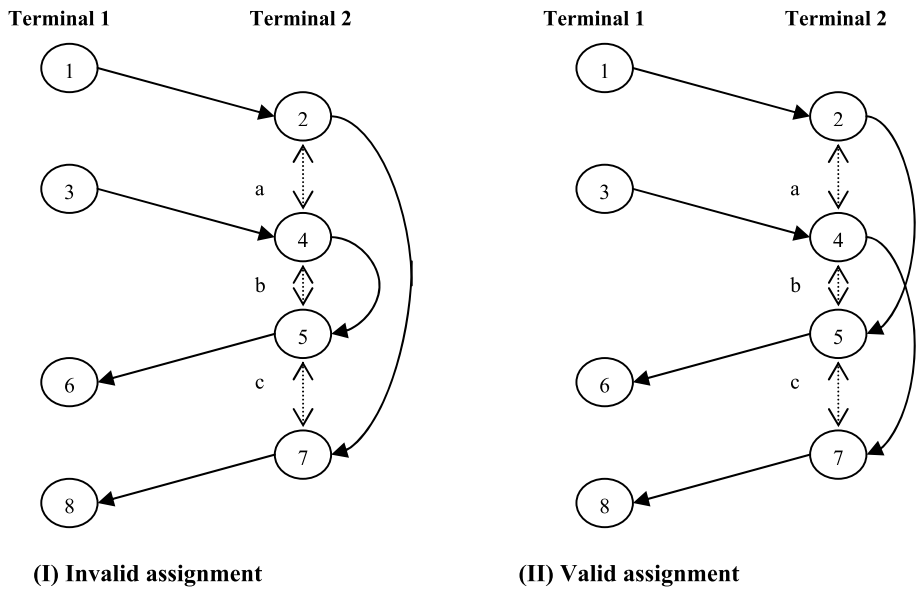
Then,

Cost of assignment (I)

$$
\begin{aligned}
&= k(\textit{duration of arc}\,(2,7))^2 \\
&\quad + k(\textit{duration of arc}\,(4,5))^2 \\
&= k(a+b+c)^2 + kb^2 \\
&= k(a^2 + 2b^2 + c^2 + 2ab + 2bc + 2ca)
\end{aligned}
$$

Cost of assignment (II)

$$
\begin{aligned}
&= k(\textit{duration of arc}\,(2,5))^2 \\
&\quad + k(\textit{duration of arc}\,(4,7))^2 \\
&= k(a+b)^2 + k(b+c)^2 \\
&= k(a^2 + 2b^2 + c^2 + 2ab + 2bc)
\end{aligned}
$$



Railroad Crew Scheduling, Figure 3
Illustrating the FIFO assignments

It can be observed that the cost of assignments in case (II) is less than that in case (I). Hence, when the rest arcs have zero costs, the QCP scheme applied to the Relaxed Problem gives FIFO compliant assignments unless there is priority in assigning crew pools to trains. These observations are stated as the following theorem.

Theorem 2 *Quadratic Cost Perturbation applied to the Relaxed Problem guarantees FIFO compliant crew assignments if there is no priority in assigning crews to the trains.*

The solution time of QCP is very short and is comparable to that of the Relaxed Problem. Note that in the case where there are priorities, this approach could be used to obtain a solution with a small number of violations and the SCG algorithm can be then used to prune out these violations. An interesting observation was that for most of the instances tested, this method produced solutions with objective function values same as those for the Relaxed Problem. This implies that FIFO constraints can be satisfied with little or no impact on the solution cost. Hence, QCP can be used to obtain excellent quality solutions using much less computational time. Due to its attractive running times and high solution quality, this method has the potential to be used in both the planning and the real-time environment.

Applications

The crew scheduling model has applications in the tactical, planning and strategic environments. Some specific examples of how the model can be used as an effective tool for decision making are provided in this section.

Tactical Crew Scheduling: Tactical scheduling refers to the decisions that need to be taken in the real-time planning environment on daily basis. The model for the CSP has several potential applications in the tactical scheduling environment. Some of the applications are: (i) Assign crews to trains, (ii) Recommend which crews to place in hotels and which crews to deadhead home, (iii) Minimize trains delayed due to shortage of crew, and (iv) Disruption management.

Crew Planning: The essence of the crew planning problem for operations or planning is to determine how many crews should be deployed in each crew pool. A planning problem is solved every month as the train pattern changes with seasonal traffic demand fluctuations. Currently, railroads solve the pool sizing problem based on historical precedent and rules-of-thumb, through negotiation with the union, and by trial and error. The network flow model can satisfy the need for a structured approach that captures all of the considerations, quantifies the various costs, and recommends the best way to define and staff crew pools. Some of the applications of the model in the planning environment are: (i) Develop and evaluate crew schedules, and (ii) Analyze robustness of the plan with respect to availability of the crew in the pool.

Crew Strategic Analysis: Strategic management involves development of policies and plans and allocating resources to implement these plans. The timeframe of strategic management extends over several months or even years. Strategic crew problems include forecasting future head-count needs and evaluating major policy changes such as negotiating changes to trade-union rules or changing the number and location of crew change points on a network. The model can be used to quickly calibrate efficient frontiers for each crew district and show what number of crews minimizes the sum of train delay costs and crew costs. Some of the applications of the network flow model in the strategic environment are: (i) Determining the number of crew districts and territory of crew districts, (ii) Effect of changing crew trade-union rules, and (iii) Forecasting crew requirement.

Conclusions

The crew scheduling problem for North American railroads is governed by several Federal Railway Administration (FRA) regulations and trade-union work rules. In addition to satisfying these regulations, the objective is to minimize the total wage costs, train delay costs, deadhead costs, and detention costs. The railroads divide their network into a number of crew districts, and each crew district has several crew pools. Each crew pool in a district could have a different set of operating rules. These factors make this a complex problem to model and solve.

The network flow formulation for the crew scheduling problem described in this article is both flexible and robust and can be easily manipulated to represent each of the possibilities encountered in real-life. The crew scheduling problem is formulated as an integer program on a space-time network. The network is constructed in such a way that all FRA regulations and trade-union work rules other than FIFO constraints are enforced during the network construction phase itself. The operational constraints are handled in the integer programming formulation. Two efficient approaches to solve the problem are described.

The crew scheduling model has applications in a wide range of settings. The broad spectrum of applications varies from the short-term problem of assigning crews to trains over the next few days to the long-term problem of forecasting crew requirements based on future demand patterns. The model gives railroad executives a method to calibrate and quantify the impact of current decisions on future operations by running several "what-if" scenarios. It has the potential to make a significant impact on the railroad's on-time performance, crew utilization, and productivity, while also improving the quality-of-life for crew and improving railroad safety.

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network Flows: Theory, Algorithms, and Applications. Prentice Hall, NJ
2. Barnhart C, Johnson EL, Nemhauser GL, Vance PH (2003) Crew scheduling. In: Hall RW (ed) Handbook of Transportation Science. Kluwer, New York, pp 493–521
3. Barnhart C, Johnson EL, Anbil R, Hatay L (1994) A column generation technique for the long-haul crew assignment problem. In: Ciriano T, Leachman R (eds) Optimization in Industry, vol II. Wiley, New York, pp 7–22
4. Bodin L, Golden B, Assad AA, Ball M (1983) Routing and scheduling of vehicles and crews: The state of the art. Comput Oper Res 10:63–211
5. Caprara A, Fischetti M, Toth P, Vigo D, Guida PL (1997) Algorithms for railway crew management. Math Program 79:124–141
6. Carraresi P, Gallo G (1984) Network models for vehicle and crew scheduling. Eur J Oper Res 16:139–151
7. Chu CK, Chan CH (1998) Crew scheduling of light rail transit in Hong Kong: From modeling to implementation. Comput Oper Res 25:887–894
8. Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995) Time-constrained routing and scheduling. In: Ball MO et al (eds) Handbooks in OR & MS, vol 8. Elsevier Science, Amsterdam, pp 35–139
9. Ernst AT, Jiang H, Krishnamoorthy M, Nott H, Sier D (2001) An integrated optimization model for train crew management. Annals Oper Res 108:211–224
10. Freling R, Lentink RM, Wagelmans APM (2004) A decision support system for crew planning in passenger transportation using a flexible branch-and-price algorithm. Annals Oper Res 127:203–222
11. Gopalakrishnan B, Johnson E (2005) Airline crew scheduling: State-of-the-art. Annals Oper Res 140:305–337
12. Gorman MF, Sarrafzadeh M (2000) An application of dynamic programming to crew balancing at Burlington Northern Santa Fe Railway. Int J Serv Tech Manage 1:174–187
13. Vaidyanathan B, Jha KC, Ahuja RK (2007) Multi-commodity network flow approach to the railroad crew-scheduling problem. IBM J Res Dev 51:325–344
14. Walker CG, Snowdon JN, Ryan DM (2005) Simultaneous disruption recovery of train timetable and crew roster in real time. Comput Oper Res 32:2077–2094
15. Wise TH (1995) Column generation and polyhedral combinatorics for airline crew scheduling. Ph.D. Dissertation, Cornell University, Ithaca
16. Wren A (ed) (1981) Computer scheduling of public transportation: Urban passenger vehicle and crew scheduling. Elsevier Science Inc., New York

Railroad Locomotive Scheduling

Arvind Kumar[1],
Balachandran Vaidyanathan[2],
Ravindra K. Ahuja[2]

[1] Innovative Scheduling, Inc., GTEC, Gainesville, USA

[2] Department of Industrial & Systems Engineering, University of Florida, Gainesville, USA

Article Outline

Keywords
Introduction
Definitions
- Problem Inputs
- Hard Constraints
- Soft Constraints
- Objective Function

Formulation
- Space-Time Network
- Problem Size and Computational Issues
- Consist Flow Formulation for the LSP

Methods and Applications
Effect of Varying Minimum Connection Time on Solution Cost
Effect of Varying Transport Volume on Key Performance Characteristics
Conclusions
References

Keywords

Locomotive scheduling; Multi-commodity flows; Network flows; Space-time network; Mixed integer programming; Railroad; Resource planning

Introduction

Everyday railroad managers need to assign hundreds of locomotives to hundreds of different trains. Locomotive scheduling entails optimally assigning a set of locomotives to each train so that the assignment satisfies a variety of business constraints and minimizes the total scheduling cost. Major US railroad companies have several billions of dollars of capital investment in locomotives. Thus, solving this problem effectively is of critical importance for railroads. This chapter gives a description of the existing literature related to locomotive scheduling, an overview of the North American railroad locomotive scheduling problem (LSP) and summary of the key features of two comprehensive models for locomotive scheduling developed by [1] and [15].

The set of locomotives assigned to a train is called *consist*. When a train arrives at its destination, its consist is either assigned to an outbound train in its entirety, or its consist goes to the pool of locomotives where new consists are formed. The former case is referred to as a *train-to-train connection* between the inbound and outbound trains, and the latter case is referred to as a *consist-busting*. The cost of assembling and disassembling consists must be controlled by developing plans that minimize *consist-busting*.

Locomotives which provide power to the train are referred to as *active* locomotives. Due to difference in power demand at different stations, locomotives are also *repositioned* from one station to another. The locomotives can be repositioned by simply *deadheading* on an existing train, or by dispatching a set of locomotives which travel independently from one station to another, also referred to as *light travel.*

LSPs are notoriously hard combinatorial optimization problems. The paper [7] presents an excellent survey of existing locomotive scheduling models and algorithms for this problem. The models described in existing literature can broadly be classified in two categories: *Single locomotive type* and *Multiple locomotive type models*. *Single locomotive scheduling models* assume that there is only one type of locomotive available for assignment. These models can be viewed as minimum cost flow problems with side constraints; some papers on single locomotive scheduling models are due to [3,4,8,10,16]. Single locomotive assignment models are appropriate for some European railroad companies but are not suited for US railroad companies since most trains are assigned multiple types of locomotives. *Multiple locomotive assignment models* have been studied by [5,6,9,12,13,14,17,18]. The most recent and comprehensive multiple locomotive assignment model is due to [1] which has been further refined by [15]. While [1] develop the initial framework to solve this problem; [15] improve the initial effort on several dimensions leading to the development of a practical locomotive scheduling approach.

Definitions

This section lists the set of inputs required to define and formulate the LSP. It also describes the constraints that govern the problem and the objective function.

Problem Inputs

Locomotive Data: A railroad typically has several different types of locomotives with different pulling and cost characteristics and different number of axles (often ranging from four to nine). The set of all the locomotive types is denoted by K and the index k represents a particular locomotive type. The following data is associated with each locomotive type $k \in K$: (i) h^k: the horsepower provided by a locomotive of type k; (ii) λ^k: the number of axles in a locomotive of type k; (iii) G^k: the weekly ownership cost for a locomotive of type k; and (iv) B^k: fleet size of locomotives of type k, that is, the number of locomotives available for assignment.

Train Data: Locomotives pull a set of trains L from their origins to their destinations. Trains have different weekly frequencies; some trains run every day, while others run less frequently. The same train running on

different days is considered as different train entities; that is, if a train runs five days a week, it is considered as five different trains. The index l is used to denote a specific train. Each train has the following associated information: (i) *dep-time*(l): the departure time for the train l; (ii) *arr-time*(l): the arrival time for train l; (iii) *dep-station*(l): the departure station for train l; (iv) *arr-station*(l): the arrival station for train l; (v) T_l: tonnage requirement of train l; (vi) β_l: horsepower per tonnage needed for train l; (vii) H_l: horsepower requirement of train l, which is defined as $H_l = \beta_l T_l$; and (viii) E_l: the penalty for using a single locomotive consist for train l.

Locomotive-Train Combinations: The following information is specified for each train-locomotive type combination: (i) c_l^k: the cost incurred in assigning an active locomotive of type k to train l; (ii) d_l^k: the cost incurred in assigning a deadheaded locomotive of type k to train l; and (iii) t_l^k: the tonnage pulling capability provided by an active locomotive of type k to train l. Also specified for each train l are three disjoint sets of locomotive types: (i) *MostPreferred*[l], the preferred classes of locomotives; (ii) *LessPreferred*[l]: the acceptable (but not preferred) classes of locomotives; and (iii) *Prohibited*[l], the prohibited classes of locomotives. When assigning locomotives to a train, locomotives from the classes listed as *MostPreferred*[l] and *LessPreferred*[l] can be assigned (a penalty is associated for using *LessPreferred*[l]).

Hard Constraints

Hard constraints are mandatory constraints which have to be satisfied for a locomotive assignment plan to be feasible.

Power Requirement for Trains: Each train must be assigned locomotives with at least the required tonnage and horsepower.

Locomotive Class to Train Type: Each train type (e. g., auto train, or merchandise train, or intermodal train) is targeted to use specific classes of locomotive types.

Geographic: Each geographic region permits the travel of only specific locomotive types. For example, it may be specified that Atlanta can only use: CW40, AC44, and AC60 locomotives.

Locomotive Balance Constraints: The number of incoming locomotives of each type into a station must be equal to the number of outgoing locomotives of that type at that station.

Active Axle Constraints: Each train must be assigned locomotives with at most 24 active axles. Exceeding 24 powered axles may overstress the couplers and cause a train separation.

Consist Size Constraints: Each train can be assigned at most 12 locomotives including both the active and deadheading locomotives. This business policy reduces risk exposure if the train were to suffer a catastrophic derailment.

Fleet Size Constraints: The number of assigned locomotives of each type is at most the number of available locomotives of that type.

Repeatability of the Schedule: The routing of locomotives should be such that the number of locomotives of each type at each station at the end of the week should be equal to the number of locomotives of each type at each station at the beginning of the next week (so that the plan is repeatable every week).

Soft Constraints

These constraints are flexible constraints and they define characteristics of a solution which are preferred but not mandatory.

Consistency in Locomotive Assignment: A train should be assigned the same consist each day that it runs. Railroads believe that crews will perform more efficiently and more safely if they operate the same equipment on a particular route and train.

Consistency in Train Connections: If locomotives traveling on a train to its destination station connect to another train originating at that station, then they should preferably make the same connection on each day that both of the trains run.

Avoid Consist Busting: Since consist busting involves the use of more resources, it is preferable to avoid consist busting.

Avoid Single Locomotive Consists: If a single locomotive is assigned to a train and this locomotive breaks down, then the train will get stranded.

Objective Function

The objective function for the LSP is to minimize the sum of: (i) Cost of ownership, maintenance, and fueling of locomotives; (ii) Cost of active and deadheaded

locomotives; (iii) Cost of light traveling locomotives; (iv) Penalty for consist-busting; (v) Penalty for inconsistency in locomotive assignment; and (vi) Penalty for using single locomotive consists.

Formulation

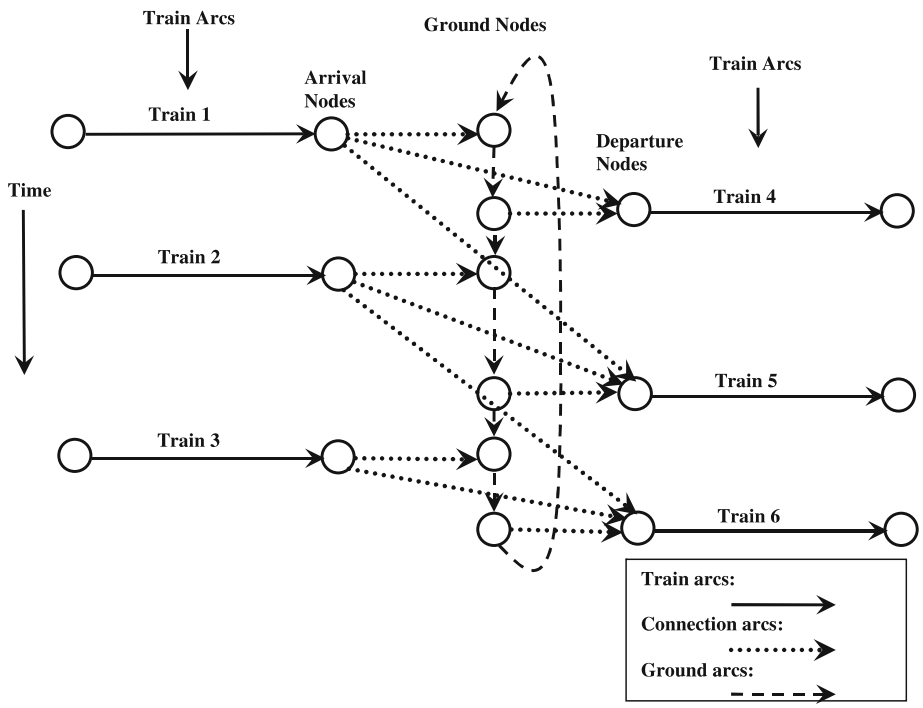
The LSP is formulated as a multi-commodity flow problem with side constraints on a network called the *weekly space-time network*. Each locomotive type defines a commodity in the network.

Space-Time Network

The weekly space-time network is denoted as $G^7 = (N^7, A^7)$, where N^7 denotes the node set and A^7 denotes the arc set. Figure 1 displays a part of the weekly space-time network at one location. The network is constructed as follows:

Nodes in the Weekly Space-Time Network: The network contains a *train arc* (i_l, j_l) for each train l. The tail node i_l of the arc denotes the event for the departure of train l at *dep-station*(l) and is called a *departure node*. The head node j_l denotes the arrival event of train l at *arr-station*(l) and is called an *arrival node*. For each arrival event, an *arrival-ground node* is created, and for each departure event, a *departure-ground* node is created. Each node in the network is associated with two attributes: time and place. *Time*(i) denotes the time attribute of node i in the weekly space-time network. The sets of departure, arrival, and ground nodes are denoted by the sets *DepNodes*, *ArrNodes*, and *GrNodes*, respectively. Let the set *AllNodes* = *DepNodes* ∪ *ArrNodes* ∪ *GrNodes*.

Arcs in the Weekly Space-Time Network: The network contains four types of arcs. The first is the set of train arcs, denoted by the set *TrArcs*, and contains an arc for every train. Each arrival node is connected to the associated arrival-ground node by a directed arc called the *arrival-ground connection arc*. Each departure-ground node is connected to the associated departure node through a directed arc called the *ground-departure connection arc*. All the ground nodes at each station are sorted in the chronological order of their time attributes and each ground node is connected to the next ground node through directed arcs called *ground arcs*. The ground arcs allow inbound locomo-



Railroad Locomotive Scheduling, Figure 1
A part of the weekly space-time network

tives to stay in an inventory pool as they wait to be connected to the outbound trains. The last ground node in the week at a station is connected to the first ground node of the week at that station through the ground arc; this ground arc models the ending inventory of locomotives for a week, which becomes the starting inventory for the following week. The possibility of an inbound train sending its entire consist to an outbound train is modeled by creating *train-train connection arcs* from train arrival nodes to train departure nodes whenever such a connection can be feasibly made. Therefore, the four kinds of arcs are: train arcs (*TrArcs*), connection arcs (*CoArcs*), and ground arcs (*GrArcs*). Let $AllArcs = TrArcs \cup CoArcs \cup GrArcs$.

The LSP can be formulated as a flow of different types of locomotives in the weekly space-time network. Locomotives flowing on train arcs are either active or deadheading; and those flowing on connection and ground arcs are idling (that is, waiting between two consecutive assignments). The following additional notation for the weekly space-time networks is used in the MIP formulations: (i) $I[i]$: the set of incoming arcs into node $i \in AllNodes$; (ii) $O[i]$: the set of arcs emanating from node $i \in AllNodes$; (iii) d_l^k: defined for every arc $l \in AllArcs$ (for a train arc l, d_l^k denotes the cost of deadheading of locomotive type k on train arc l, and for every other arc it denotes the cost of traveling for a non-active locomotive of locomotive type k on arc l); (v) CB: the set of all connection arcs from arrival nodes to ground nodes; alternatively, $CB = \{(i, j) \in AllArcs: i \in ArrNodes \text{ and } j \in GrNodes\}$; (vi) $CheckTime$: a time instant of the week when no event takes place (that is, no train arrives or departs at any station); and (vii) S: the set of arcs that cross the *CheckTime* [that is, $S = \{(i, j) \in AllArcs: time(i) < CheckTime < time(j)\}$].

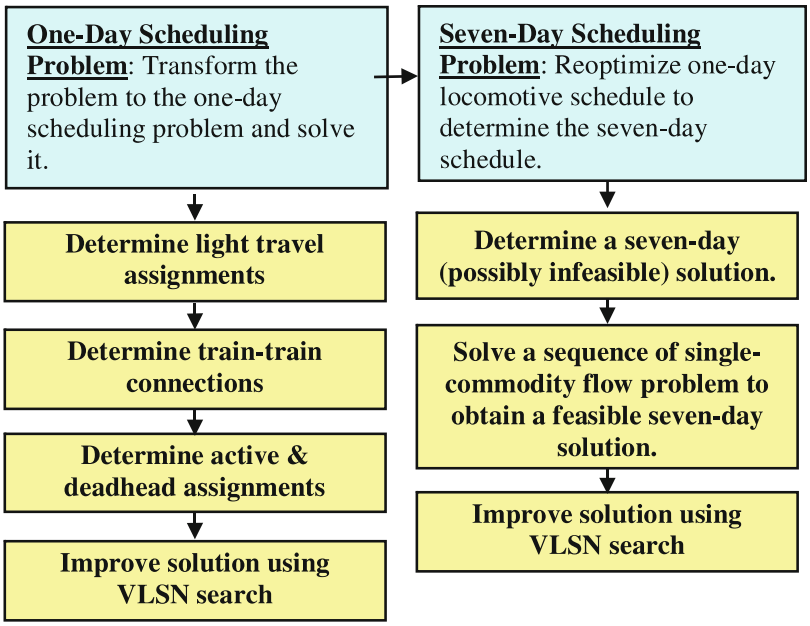
Problem Size and Computational Issues

The integer programming formulation of the LSP contains around 200,000 variables and 100,000 constraints and cannot be solved to optimality or near optimality using commercial state-of-the-art software. Even the linear programming relaxation of this problem cannot be solved in a reasonable amount of time. Additionally, the formulation given above cannot capture the consistency constraints effectively. The main contribution made in [1] was to develop a two-stage solution approach to solve this problem. In the first stage, the daily locomotive scheduling problem which is a simplified problem is solved, and in the second stage the daily locomotive schedule is modified to obtain the feasible weekly locomotive schedule.

This approximate two stage approach was motivated by the observation that in a typical problem more than 90% of the train arcs in the space-time network correspond to the trains that run 5, 6, or 7 days. Based on this observation, the daily locomotive scheduling model that is a simplification of the weekly locomotive scheduling model is created in the following manner. (i) all trains that run p days or more per week run *every* day of the week; and (ii) all trains that run fewer than p days do not run at all. (Note: This assumption results in an approximation in the sense that locomotives are provided to some trains that do not exist, and locomotives are not provided to some trains that exist.)

To transform the solution of the daily locomotive scheduling solution into a feasible solution to the weekly scheduling problem, locomotives are taken from the trains that exist in the daily problem but do not exist in the weekly problem (Type 1 trains) and assigned to the trains that do not exist in the daily problem but exist in the weekly problem (Type 2 trains). This may lead to the model using additional locomotives to meet the constraints. The solution of the daily problem can be translated into the solution of the weekly problem more effectively if the number of Type 1 trains is less than the number of Type 2 trains but still as close as possible.

Another contribution made in [1] is determination of good train-train connections and good light arcs. Railroads often specify some "candidate" train-train connections and "candidate light arcs" out of which a certain number are fixed in the final solution. In the basic formulation, each "candidate" train connection or light arc has a fixed charge variable associated with it and these fixed charge variables make the MIP very hard to solve. This issue is handled in the following manner. The space-time network that contains all the candidate train-train connection arcs and candidate light arcs is constructed. The linear programming relaxation of the LSP has no fixed charge variables, and has large costs for all the train-to-ground and ground-to-train arc flows to discourage the flow on such arcs (or,



Railroad Locomotive Scheduling, Figure 2
Overview of the multi-stage locomotive scheduling algorithm

alternatively, to discourage consist-busting), is solved. Let $\varphi(l)$ denote the total flow of locomotives (of all types) on any arc l in the daily space-time network. The candidate train-train connection arc h with the largest value of $\varphi(h)$ is selected. This arc indicates a successful potential train-train connection. Connection arc h is made the unique connection arc for the two corresponding trains and the linear programming relaxation is resolved. If this linear programming relaxation is infeasible or if it increases the cost of the new solution by an amount greater than θ, then the train-train connection is not made; otherwise, it is fixed as a good connection. This iterative procedure is repeated until either the desired number of train-train connections is reached (as specified by some parameter γ), or until the set of candidate train-train connections becomes empty. Using a similar greedy iterative approach, the set of good light moves is also determined. Figure 2 gives an overview of the overall approach in [1].

Consist Flow Formulation for the LSP

Consist busting is such an anathema to real world railroad managers that most managers stipulate that a high quality locomotive plan is one designed such that no consist busting is required. Consist busting affects crew requirements, station fluidity, locomotive productivity, and mechanical maintenance processes. Consist busting consumes between two and six additional hours per locomotive within the station, asset time that could be productively used to pull trains on the mainline. Upon reassembly, each consist must undergo extensive operational testing as well. In addition, consist-busting often results in outbound trains getting their locomotives from several inbound trains. If any of these inbound trains is delayed, the outbound train is also delayed, which potentially propagates to further delays down the line. Consequently, railroad managers seek to streamline and simplify processes in order to eliminate fragility in the operating plan. In reality, consists will be tactically busted as part of real-time operations to compensate for unplanned events.

In order to minimize consist busting, [15] extended the locomotive flow formulation described in [1]. Most features of the model are kept identical but consists are routed over the network instead of individual locomotives. In this formulation, referred to as *consist flow formulation*, each consist type (that is, a group of locomotives) is defined to be a commodity that flows on the network. Thus, the consist flow formulation differs

from the locomotive flow formulation in the sense that locomotive types are replaced by the consist types. Observe that each feasible solution of the consist flow formulation has a corresponding feasible solution to the locomotive flow formulation with the same cost, but the converse is not true. Thus, the reader might be led to believe that consist formulation is quite restrictive and may produce significantly inferior optimal solution compared to the flow formulation. However, computational results revealed that if the number and types of consists are judiciously chosen, then both formulations produce solutions with comparable quality.

The consist flow formulation is a multi-commodity integer program. The network is constructed in a similar way to that described in Sect. "Space-Time Network"

Additional Notation

C: Denotes the set of consist types available for assignment and $c \in C$represent a particular consist.

c_l^c: Cost of assigning an active consist of type $c \in C$to train l.

d_l^c: Defined for every arc $l \in AllArcs$. For a train arc $l \in TrArcs$, d_l^ccaptures the cost of deadheading a consist of type $c \in C$on arc l. For an arc $l \in CoArcs \cup GrArcs$, d_l^ccaptures the cost of idling for a consist type $c \in C$on arc l.

α^{ck}: Number of locomotives of type $k \in K$ in consist type $c \in C$.

$I[i]$: Set of arcs entering node i.

$O[i]$: Set of arcs leaving node i.

S: Set of overnight arcs or arcs that cross the Sunday midnight timeline. (This time is chosen as the time for counting the number of locomotives used in the solution.)

Decision Variables

x_l^c: Binary variable representing the number of active consists of type $c \in C$ on arc $l \in TrArcs$.

y_l^c: Integer variable representing the number of non-active consists (deadheading, light-traveling or idling) of type $c \in C$ on arc $l \in AllArcs$.

z_l: Binary variable which takes value 1 if at least one consists flows on arc $l \in LiArcs$ and 0 otherwise.

s^k: Integer variable indicating the number of unused locomotives of type $k \in K$.

Objective Function

$$\min z = \sum_{l \in TrArcs} \sum_{c \in C} c_l^c x_l^c + \sum_{l \in AllArcs} \sum_{c \in C} d_l^c y_l^c + \sum_{l \in LiArcs} F_l z_l - \sum_{k \in K} G^k s^k \quad (1)$$

Constraints

$$\sum_{c \in C} x_l^c = 1\,, \quad \text{for all} \quad l \in TrArcs\,, \quad (2)$$

$$\sum_{c \in C} \sum_{k \in K} \alpha^{ck}(x_l^c + y_l^c) \le 12, \quad \text{for all} \quad l \in TrArcs, \quad (3)$$

$$\sum_{l \in I[i]} (x_l^c + y_l^c) = \sum_{l \in O[i]} (x_l^c + y_l^c)\,, \quad \text{for all} \quad i \in AllNodes, \quad c \in C\,, \quad (4)$$

$$\sum_{k \in K} \sum_{c \in C} \alpha_{ck} y_l^c \le 12 z_l\,, \text{ for all } l \in LiArcs\,, \quad (5)$$

$$\sum_{l \in S} \sum_{c \in C} \alpha^{ck}(x_l^c + y_l^c) day(l) + s^k = B^k\,, \quad \text{for all} \quad k \in K\,, \quad (6)$$

$$x_l^c \in \{0,1\}\,, \quad \text{for all} \quad l \in TrArcs, \quad c \in C\,, \quad (7)$$

$$y_l^c \ge 0 \text{ and integer}\,, \text{for all } l \in AllArcs, c \in C\,, \quad (8)$$

$$z_l \in \{0, 1\}\,, \text{for all } l \in LiArcs\,. \quad (9)$$

$$s^k \ge 0\,, \text{ for all } k \in K \quad (10)$$

Constraint (2) ensures that every train l is assigned exactly one active consist. Constraint (3) ensures that the locomotive flow upper-bound on each train arc is satisfied. Constraint (4) ensures that flow is balanced at every node for every consist type. Constraint (5) ensures that the locomotive flow upper-bound on each light arc is satisfied. Constraint (6) ensures that the number of locomotives used for each fleet type is no more than the fleet size.

Note that in this formulation, it is not required to explicitly specify the constraints that each train gets the required tonnage, horsepower and does not exceed the 24-active axle requirement. These constraints are implicitly handled in the formulation. The active axle constraints are handled by not creating consists which have more than 24 active axles. The tonnage and horsepower

constraints are handled implicitly in the following way; if assigning consist $c \in C$as an active consist to train $l \in TrArcs$ violates the tonnage or horse power constraints, then the corresponding variable is set to zero ($x_l^c = 0$); thus disallowing the assignment of consist c to train l. The consist flow formulation has significantly less side constraints compared to the locomotive flow formulation, resulting in faster solution time. Another speed-up in the consist flow formulation comes from the fact that each active consist assignment variable, x_l^c,is a binary variable, whereas in the locomotive flow formulation, it is an integer variable and can take values 0, 1, 2, 3, … etc. The MIP optimization engine, which is typically a branch and bound algorithm, is likely to explore lesser branches for a binary integer program compared to a general integer program as there are lesser options to pursue. There are instances where the locomotive flow formulation could not give a feasible integral solution in 10 hours, but the consist flow formulation gave an optimal solution within a few minutes of computational time.

Railroads often impose complex rules on what locomotive types may be combined into ideal consists. Some locomotives do not work well together. Some railroads segregate AC powered locomotives and DC powered locomotives. These requirements are often very hard or impossible to honor in the locomotive flow formulation but are rather trivial in the consist flow formulation. Further, in the locomotive flow formulation, an outbound train often obtains its planned consist from locomotives coming from multiple trains and if any of these inbound trains is delayed, the outbound train is delayed as well. But in the consist flow formulation, all outbound trains derive their active consist only from one inbound train (but may derive their deadhead consists from other trains) thus reducing the impact of train delays. In summary, the benefits of using the consist formulation are (1) solution speed and robustness greatly improved, (2) consist busting is reduced to zero, and (3) constraints are more easily incorporated, resulting in more practical solutions.

Computational tests show that the consist flow formulation may have its optimal objective function value as much as 5% higher than that of the locomotive flow formulation. However, the solution is far superior in terms of consistency, simplicity, and robustness. Thus, it may be easier to comply with and may need overall fewer locomotives in practice (considering train delays, for example).

Methods and Applications

In [15], the authors describe methods to use the consist-based formulation to incorporate several practical requirements to generate an implementable plan. These include incremental locomotive planning, satisfying cab-signal requirements, incorporating foreign power requirements, and accounting for shop power requirements. The interested reader may refer to [15] for further details.

In this section, the result of two case studies to illustrate the uses of the model to assist decision making, are presented. The aspects of the problem observed are: (1) Effect of varying minimum connection time on solution cost, and (2) Effect of varying transport volumes on key transport performance characteristics.

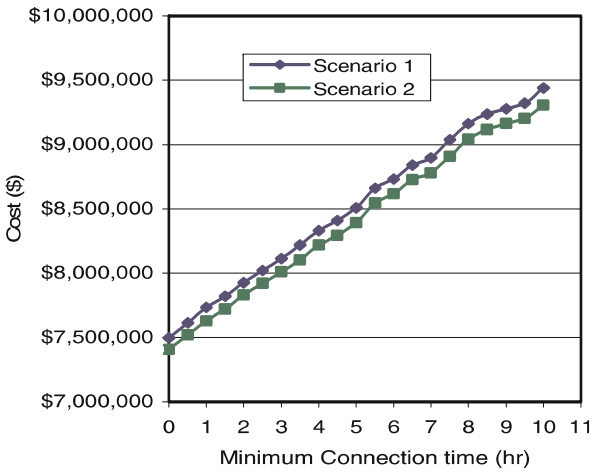
Effect of Varying Minimum Connection Time on Solution Cost

Freight trains do not run on time and often arrive later than their planned arrival time, which makes it difficult for locomotive dispatchers to adhere to the locomotive plan. One method commonly recommended to improve plan compliance is to increase the train-train minimum connection times. Although increasing the minimum connection time may improve plan adherence, it also increases locomotive costs as more locomotives will be held in inventory at terminals. This study quantifies the impact of increasing the minimum connection times.

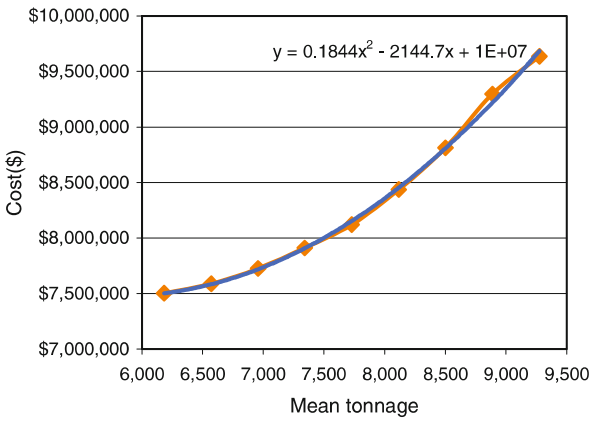
The following conclusions can be drawn from the study (see Fig. 3):

- The minimum connection time could have a significant impact on the solution cost.
- The solution cost increases linearly with the increase in connection times at the rate of around $200,000 per hour.

Depending upon the lateness of trains and the willingness of railroad planners to improve locomotive plan compliance, appropriate connection times can be used. As shown in this study, railroads can use the model to estimate the benefit of reducing locomotive connection times at terminals; this reduction may involve the investment of resources to improve efficiency of termi-



Railroad Locomotive Scheduling, Figure 3
Solution cost vs. minimum connection time



Railroad Locomotive Scheduling, Figure 4
Impact of transport volumes on solution cost

nal activities like fueling and servicing locomotives, and consist-busting overheads.

Effect of Varying Transport Volume on Key Performance Characteristics

In this study, the impact of varying transport volumes (or train tonnages) on the following key transport characteristics: number of locomotives used, solution cost, mean pulling power of a consist, and mean miles traveled per consist, is measured. The results of these tests are presented in Table 1 and Fig. 4.

The following conclusions can be drawn from this study:

- As the transport volume (mean tonnage) increases, the solution cost increases as a quadratic function as shown in Fig. 4. The nature of the variation indicates that the rate of change of cost with respect to tonnage (slope) is a linear increasing function.
- The mean pulling power of each consist and the number of locomotives used increase as expected but the surprising observation is that the average number of miles traveled by each consist remains roughly the same.

As demonstrated in this study, railroads can use the model as a generic approach for modeling the optimal number of locomotives needed or the cost as a function of the rail freight transport volume over the entire network.

Railroad Locomotive Scheduling, Table 1
Effect of varying transport volumes

% increase in tonnage	Mean tonnage of trains	Locos Used	Solution Cost ($)	Mean pulling power/consist	Mean miles/ consist
−20	6,183	1,026	7,502,802	10,099	405.05
−15	6,570	1,026	7,588,421	10,373	405.04
−10	6,956	1,042	7,726,579	10,464	404.17
−5	7,342	1,040	7,910,331	10,841	405.14
0	7,729	1,049	8,120,134	11,093	403.73
5	8,115	1,079	8,437,334	11,457	403.86
10	8,502	1,117	8,812,467	11,828	405.27
15	8,888	1,171	9,296,423	12,483	403.98
20	9,275	1,214	9,635,790	12,715	405.12

Conclusions

The LSP creates a blueprint for use by managers to assist them in making tactical, day-to-day decisions regarding locomotive assignments. This article provides an overview of some of the recent work done in this area. The locomotive planning tools presented in this article have the potential to serve as a core component in a strategic fleet sizing model. Each year, railroads develop five year outlooks and forecast the demand for freight transportation and the associated train schedules to handle that demand. Railroads also must project locomotive fleet supply, as units reach their economic limit of repair and are retired from the fleet. The gap between the future supply of locomotives and the future demand for locomotives must be closed with a combination of improved productivity and new purchases. This planning tool will go a long way towards helping railroad financial executives and strategic planners set asset goals and negotiate the purchase of new locomotives. The planning tool can also be used to perform "*what-if*" kind of analysis and allow railroad managers to take informed decisions after analyzing their global impacts.

References

1. Ahuja RK, Liu J, Orlin JB, Sharma D, Shughart LA (2005) Solving real-life locomotive scheduling problems. Trans Sci 39:503–517
2. Ahuja RK, Magnanti TL, Orlin JB (1993) Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs
3. Booler JMP (1980) The solution of a railway locomotive scheduling problem. J Oper Res Soc 31:943–948
4. Booler JMP (1995) A note on the use of Lagrangean Relaxation in railway scheduling. J Oper Res Soc 46:123–127
5. Chih KC, Hornung MA, Rothenberg MS, Kornhauser AL (1990) Implementation of a real time locomotive distribution system. In: Murthy TKS, Rivier RE, List GF, Mikolaj J (eds) Computer Applications in Railway Planning and Management. Computational Mechanics Publications, Southampton, pp 39–49
6. Cordeau JF, Soumis F, Desrosiers J (2000) A Benders decomposition approach for the locomotive and car assignment problem. Trans Sci 34(2):133–149
7. Cordeau JF, Toth P, Vigo D (1998) A survey of optimization models for train routing and scheduling. Trans Sci 32:380–404
8. Fischetti M, Toth P (1997) A package for locomotive scheduling. Technical Report DEIS-OR-97–16, University of Bologna
9. Florian M, Bushell G, Ferland J, Guerin G, Nastansky L (1976) The engine scheduling problem in a railway network. INFOR 14:121–138
10. Forbes MA, Holt JN, Watts AM (1991) Exact solution of locomotive scheduling problems. J Oper Res Soc 42:825–831
11. Nemhauser GL, Wolsey LA (1988) Integer and Combinatorial Optimization. Wiley, New York
12. Nou A, Desrosiers J, Soumis F (1997) Weekly locomotive scheduling at Swedish State Railways. Technical Report TRITA/MAT-97-OS12, Royal Institute of Technology, Stockholm
13. Ramani KV (1981) An information system for allocating coach stock on Indian Railways. Interfaces 11:44–51
14. Smith S, Sheffi Y (1988) Locomotive scheduling under uncertain demand. Trans Res Records 1251:45–53
15. Vaidyanathan B, Ahuja RK, Liu J, Shughart LA (2008) Real-life locomotive planning: new formulations and computational results. Transp Res Part B 42:147–168
16. Wright MB (1989) Applying stochastic algorithms to a locomotive scheduling problem. J Oper Res Soc 40:187–192
17. Ziarati K, Soumis F, Desrosiers J, Gelinas S, Saintonge A (1997) Locomotive assignment with heterogeneous consists at CN North America. Eur J Oper Res 97:281–292
18. Ziarati K, Soumis F, Desrosiers J, Solomon MM (1999) A branch-first, cut-second approach for locomotive assignment. Manage Sci 45:1156–1168

Random Search Methods

H. Edwin Romeijn
Department Industrial and Systems Engineering, University Florida, Gainsville, USA

MSC2000: 65K05, 90C30

Article Outline

Keywords
Conceptual Methods
(Pure) Random Search
Pure Adaptive Search
Adaptive Search
Hesitant Adaptive Search
Algorithms
Simulated Annealing
Pure Localization Search
Conclusions
See also
References

Keywords

Global optimization; Nonsmooth optimization; Unbounded optimization; Interval methods

A general *global optimization problem* is a problem of the form

$$\begin{cases} \max & f(x) \\ \text{s.t.} & x \in S, \end{cases}$$

where f is a continuous function on S, and $S \subset \mathbf{R}^d$ is a compact body, and the goal is to find a point in the set

$$S^* \equiv \{x \in S\colon\ f(x) = f^*\}$$

where

$$f^* \equiv \max_{x \in S} f(x).$$

It is well-known that, without additional assumptions, this problem is inherently unsolvable in a finite number of steps. Therefore, the global optimization problem is usually considered solved if a point is found in

$$B_\epsilon^* \equiv \{x \in S\colon\ \|x - x^*\| \le \epsilon \text{ for some } x^* \in S^*\}$$

or in the level set

$$S_\epsilon \equiv \{x \in S\colon\ f(x) \ge f^* - \epsilon\}$$

for some $\epsilon > 0$ [17].

Stochastic methods are methods that contain some stochastic elements. This means that either the outcome of the method is a random variable, or the objective function itself is considered to be a realization of a stochastic process. Therefore, the possibility of an absolute guarantee of success is sacrificed. Instead, it is usually possible to prove that, as the effort increases to infinity, an element of B_ϵ^* or S_ϵ will be found with probability one. Surveys on the topic of stochastic methods for global optimization can be found in [10,29,38] and [33].

Random search methods are those stochastic methods that rely solely on the random sampling of a sequence of points in the feasible region of the problem, according to some prespecified probability distribution, or sequence of probability distributions. These methods are applicable to, and enjoy an asymptotic performance guarantee for, a very wide class of problems. Therefore, these methods have, during the last decade, enjoyed increasing interest for its ability to handle problems whose mathematical structure is difficult (or undesirable, or even impossible) to analyze.

Conceptual Methods

The methods discussed here are of a conceptual nature, in the sense that at this point there does not exist an efficient implementation of these methods. However, the theoretical results that can be obtained for these methods are interesting in itself. Moreover, they have shown potential for inspiring (or theoretically supporting) practical algorithms for global optimization.

(Pure) Random Search

The simplest stochastic method for global optimization is *pure random search* (PRS) ([12] and [5]). This method consists of generating a sequence of independent and identically distributed uniform points in the feasible region S, while keeping track of the best point that is found. In pseudocode, the method is given below.

The sequence of points generated by this method converges to a global optimum with probability one. In particular, the probability that a point in S_ϵ is reached within the first N iterations is equal to

$$1 - (1 - \varphi(S_\epsilon))^N,$$

where φ denotes the uniform distribution on S. In other words, this method offers a probabilistic asymptotic guarantee.

```
PROCEDURE pure random search()
    InputInstance();
    Set y = −∞;
    DO
        Generate a point x from the uniform distribution over S;
        Set y = max(y, f(x));
    OD;
    RETURN(y);
END pure random search;
```

A pseudocode for PRS

In [37], F.J. Solis and R.J-B. Wets introduce a class of random search methods whose most important feature, as compared to pure random search, is that disimprove-

ments are disallowed through a generalization of the acceptance-rejection approach. Moreover, sampling is not limited to the uniform distribution, but to a (prespecified) sequence of (absolutely continuous) probability distributions on S. Asymptotic convergence with probability one can again be shown.

If the random search method is *adaptive*, i. e. if we allow the distributions to be sampled from to depend on the values of previously found solutions, then convergence cannot be assured in general.

Pure Adaptive Search

Pure adaptive search (PAS) differs from PRS, and resembles the random search methods of Solis and Wets, in that it forces improvement in each iteration. However, the improving force is much stronger in that the algorithm is truly adaptive, without using any form of acceptance-rejection. In particular, an iteration point is generated from the uniform distribution on the subset of points that are improving with respect to the previous iteration point. More formally, the method reads:

```
PROCEDURE pure adaptive search()
    InputInstance();
    Set y = −∞;
    DO
        Generate a point x from the uniform distribution over {x ∈ S : f(x) > y};
        Set y = f(x);
    OD;
    RETURN(y);
END pure adaptive search;
```

A pseudocode for PAS

This method has been introduced and analyzed in [27] for convex programming problems, and by Z.B. Zabinsky and R.L. Smith [42] for more general global optimization problems. For Lipschitz continuous problems with convex feasible regions, the expected number of iterations to achieve a solution with a given precision increases at most linearly in the dimension d of the problem. This result suggests there is hope for an efficient random search method for global optimization. In fact, several random search methods have reported empirical linearity in dimension for optimizing quadratic functions ([34,35] and [37]). PAS has been extended to the case of a finite domain in [44], with analogous performance results.

Adaptive Search

It is generally very difficult to generate a point uniformly distributed in a *level set* of the global optimization problem. A method that avoids this issue, at the cost of having to generate from other distributions than the uniform, is *adaptive search* (AS). In this method, a sequence of improving points is generated by generating points according to a sequence of distributions in the feasible region S, while using an acceptance-rejection approach to attain improvement. The sequence of generating distributions should be chosen in such a way that, as the method progresses, these distributions concentrate more and more around the global optimum of the problem. An example of a family of distributions having this property is the family of *Boltzmann distributions*. This family of distributions, parameterized by a positive parameter T, are absolutely continuous on S with densities

$$g_T(x) \propto e^{f(x)/T}.$$

Note that, as the parameter T approaches infinity, the sequence of distributions approaches the uniform distribution on S. On the other hand, it can be shown that, as the parameter T approaches zero, the sequence of distributions converges to a distribution that concentrates on the set of points where the global optimum is attained. Using this family of distributions, the adaptive search method reads:

```
PROCEDURE adaptive search()
    InputInstance();
    Set y = −∞;
    Set T = ∞;
    DO
        Generate points x from the Boltzmann distribution with parameter T
        until a point is found with f(x) > y;
        Set y = f(x);
        Decrease T;
    OD;
    RETURN(y);
END adaptive search;
```

A pseudocode for AS

This method was introduced by H.E. Romeijn and Smith [30] with the objective of studying the behavior of *simulated annealing* (see below). They generalized the key result for pure adaptive search as follows. If the value of the parameter T depends, in a monotonely decreasing way, on the current record value, then the expected number of improving points to achieve a solution with a given precision increases at most linearly in the dimension d of the problem for a wide class of global optimization problems. The parameter T can be used to limit the total number of iterations (including the nonrecord values generated in the acceptance-rejection phase). Ideally, one should choose the value of the parameter T in such a way that, during the next iteration, the probability of obtaining an improving point is at least equal to some fixed value.

Hesitant Adaptive Search

One way of implementing PAS would be to use an acceptance-rejection approach for generating points in level sets of S (assuming the problem of generating a uniformly distributed point in S itself is relatively easy). In terms of total number of iterations, this approach would be equivalent to PAS, with the following modification. At each iteration, either a new PAS iterate is generated (with some probability b) or the previous iteration point is repeated (with probability $1 - b$), where b depends on the current record value. More precisely, b is the relative measure of the level set corresponding to the current record value. *hesitant adaptive search* (HAS), introduced by D.W. Bulger and G.R. Wood [13] with the objective of studying *localization search* algorithms (see below), generalizes this way of viewing PAS by relaxing the specific choice of b mentioned above. An explicit expression for the expected number of iterations required to obtain a point with a given objective function value (or better), and even the full distribution of this random variable, can be derived (see also [40]).

Algorithms

Simulated Annealing

Simulated annealing (SA) is a random search method that avoids getting trapped in local maxima by accepting, in addition to transitions corresponding to an increase in function value, also transitions corresponding to a *decrease* in function value. The latter is done in a limited way by means of a probabilistic acceptance criterion. In the course of the maximization process, the probability of accepting deteriorations descends slowly towards zero. These 'deteriorations' make it possible to move away from local optima and explore the feasible region S in its entirety.

SA originated from an analogy with the physical annealing process of finding low energy states of a solid in a heat bath [26]. M. Pincus [28] developed an algorithm based on this analogy for solving discretizations of continuous global optimization problems. Many applications to date have been to discrete (combinatorial) optimization problems (see e. g. [1,23] and [2]).

All SA algorithms for global optimizations can be viewed as approximative versions of AS. The main problem with directly implementing AS is that, in general, it will be extremely difficult to generate points exactly from the Boltzmann distribution on S. The approximative character of SA lies in the fact that SA algorithms use a *Markov chain sampling* approach instead. This means that a Markov chain is defined on S having the property that the limiting distribution of this Markov chain is the desired Boltzmann distribution. One way of achieving this is the following. First, create a Markov chain on S that has the uniform distribution as its limiting distribution. Examples of such Markov chains are the *hit and run* generator (see [8,9,36] and [32]), and the random ball walk (see [25]). This Markov chain can then be filtered as follows to change the limiting distribution to the Boltzmann distribution. If the current iteration point is x, and the current value of the *temperature parameter* is T, then the candidate point (say z) that is generated by the Markov chain is accepted with probability

$$\min\{1, e^{(f(z)-f(x))/T}\}$$

(the *Metropolis criterion*). Otherwise, we discard the candidate point and remain at the current iteration point.

C.J.P. Bélisle [7] showed that, although successive iterations of SA may experience deteriorations in objective function value, these effects are, under mild conditions, transient if the Markov chain reaches globally, i. e., if each feasible point can be reached from any other feasible point in one iteration. In particular, if the sequence of temperature parameters T decreases to zero

```
PROCEDURE simulated annealing()
    InputInstance();
    Set y = −∞;
    Set T = ∞;
    Choose x ∈ S;
    DO
        Generate a point z according to some
        Markov chain transition distribution;
        With probability min{1, e^((f(z)−f(x))/T)},
        set x = z;
        If f(x) > y set y = f(x);
        Adjust T;
    OD;
    RETURN(y);
END simulated annealing;
```

A pseudocode for SA

in probability, SA will eventually be absorbed in arbitrarily small neighborhoods of the global maximum. The *cooling schedule* controlling the way the parameter T is decreased should be chosen in accordance with AS. In practice, this means that the temperature parameter should be proportional to (an estimation of) the value error corresponding to the current record value. If the temperature value is held constant at 0 (which means that no deteriorations are ever accepted), we obtain the *improving hit and run* algorithm (see [43]). In [24], M. Locatelli derives convergence results for simulated annealing algorithms with nonglobally reaching Markov chains.

Examples of specific SA algorithms can be found in [3,11,15,16,21,31,39] and [32]. In [22], the first polynomial time implementation of an SA algorithm is presented, albeit for the unimodal problem of minimizing a linear function over an up-monotone convex set in the positive orthant. The authors derive an SA algorithm using the rapidly mixing Markov chains developed in [18].

Finally, a class of algorithms closely related to the SA algorithms discussed above are algorithms based on the *Langevin stochastic differential equation*

$$dx(t) = \nabla f(x(t))\, dt + \epsilon(t)\, dw(t), \qquad (1)$$

where ∇f is the gradient of f and $w(t)$ is a standard d-dimensional Wiener process. If $\epsilon(t)$ is constant, then the limiting distribution of the sequence of points thus generated is precisely the Boltzmann distribution at temperature $\epsilon^2/2$. In other words, algorithms based on this result can be viewed as SA algorithms, but using continuous time instead of discrete time Markov chains. See e. g. [19] and [14] for theoretical results, and [4] for a practical implementation.

```
PROCEDURE pure localization search()
    InputInstance();
    Set y = −∞;
    Set L = S;
    DO
        Generate a point x uniformly in L;
        IF f(x) > y, set y = f(x);
        Shrink L (while remaining a superset of the
        current record level set);
    OD;
    RETURN(y);
END pure localization search;
```

A pseudocode for PLS

Pure Localization Search

The aim of *pure localization search* (PLS) is to approximate PAS. The approximation consists of the following. Instead of generating a point uniformly from a level set, a point is generated uniformly from a superset of the level set, called a *localization*. Note that PRS is an instance of PLS, where the superset of each level set is the entire feasible region S.

This class of algorithms was introduced in [6] where, in addition, an example of a particular PLS algorithm for one-dimensional Lipschitz optimization is provided.

Conclusions

The main advantage of random search methods for solving global optimization problems is that they are applicable to a very wide class of optimization problems, and nevertheless provide a convergence guarantee, albeit an asymptotic one. Therefore, random search methods have been applied most successfully to problems that are black-box and/or unstructured, so that approaches that make use of a particular mathematical structure cannot be used. Examples include industrial

design problems and the design of composite laminates [20,41]. Moreover, these methods are usually very easily implemented. As such, they are also often used as a first approach towards solving a problem, until insight into the structural properties of the problem warrant the development of new, or application of existing, special purpose optimization methods.

See also

- Adaptive Simulated Annealing and its Application to Protein Folding
- Bayesian Global Optimization
- Genetic Algorithms for Protein Structure Prediction
- Global Optimization Based on Statistical Models
- Monte-Carlo Simulated Annealing in Protein Folding
- Packet Annealing
- Simulated Annealing
- Simulated Annealing Methods in Protein Folding
- Stochastic Global Optimization: Stopping Rules
- Stochastic Global Optimization: Two-phase Methods

References

1. Aarts EHL, Korst JHM (1988) Simulated annealing and Boltzmann machines. Wiley, New York
2. Aarts EHL, van Laarhoven PJM (1989) Simulated annealing: An introduction. Statistica Neerlandica 43:31–52
3. Ali MM, Storey C (1997) Aspriation based simulated annealing algorithm. J Global Optim 11:181–191
4. Aluffi-Pentini F, Parisi V, Zirilli F (1985) Global optimization and stochastic differential equations. J Optim Th Appl 47(1):1–16
5. Anderssen RS (1972) Global optimization. In: Anderssen RS, Jennings LS, Ryan DM (eds) Optimization. University Queensland Press, Brisbane, pp 1–15
6. Baritompa WP, Baoping Z, Mladineo RH, Wood GR, Zabinsky ZB (1995) Towards pure adaptive search. J Global Optim 7:93–110
7. Bélisle CJP (1992) Convergence theorems for a class of simulated annealing algorithms on $\mathbb{R}$d. J Appl Probab 29(4):885–895
8. Bélisle CJP, Romeijn HE, Smith RL (1993) Hit-and-Run algorithms for generating multivariate distributions. Math Oper Res 18(2):255–266
9. Berbee HCP, Boender CGE, Rinnooy Kan AHG, Scheffer CL, Smith RL, Telgen J (1987) Hit-and-Run algorithms for the identification of nonredundant linear inequalities. Math Program 37:184–207
10. Betr ò B (1991) Bayesian methods in global optimization. J Global Optim 1(1)
11. Bohachevsky IO, Johnson ME, Stein ML (1986) Generalized simulated annealing for function optimization. Technometrics 28(3):209–217
12. Brooks SH (1958) A discussion of random methods for seeking maxima. Oper Res 6:244–251
13. Bulger DW, WoodGR (1998) Hesitant adaptive search for global optimisation. Math Program 81:89–102
14. Chiang T-S, Hwang C-R, Sheu S-J (1987) Diffusion for global optimization in $\mathbb{R}$n. SIAM J Control Optim 25:737–753
15. Corana A, Marchesi M, Martini C, Ridella S (1987) Minimizing multimodal functions for continuous variables with the 'simulated annealing' algorithm. ACM Trans Math Softw 13:262–280
16. Dekkers A, Aarts E (1991) Global optimization and simulated annealing. Math Program 50:367–393
17. Dixon LCW (1978) Global optima without convexity. Techn Report Numer Optim Centre, Hatfield Polytechn
18. Dyer ME, Frieze AM, Kannan R (1991) A random polynomial-time algorithm for approximating the volume of convex bodies. J ACM 38(1):1–17
19. Geman S, Hwang H-R (1986) Diffusions for global optimization. SIAM J Control Optim 24:1031–1043
20. Graesser DL, Zabinsky ZB, Tuttle ME, Kim GI (1991) Designing laminated composites using random search techniques. Composite Structures 18:311–325
21. Jones AEW, Forbes GW (1995) An adaptive simulated annealing algorithm for global optimization over continuous variables. J Global Optim 6:1–37
22. Kannan R, Mount J, Tayur S (1995) A randomized algorithm to optimize over certain convex sets. Math Oper Res 20:529–549
23. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 20:671–680
24. Locatelli M (1996) Convergence properties of simulated annealing for continuous global optimization. J Appl Probab 33:1127–1140
25. Lovász L, Simonovits M (1993) Random walks in a convex body and an improved volume algorithm. Random Struct Algorithms 4(4):359–412
26. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equations of state calculations by fast computing machines. J Chem Phys 21:1087–1092
27. Patel NR, Smith RL, Zabinsky ZB (1988) Pure adaptive search in Monte Carlo optimization. Math Program 43:317–328
28. Pincus M (1970) A Monte Carlo method for the approximate solution of certain types of constrained optimization problems. Oper Res 18:1225–1228
29. Rinnooy Kan AHG, Timmer GT (1989) Global optimization. In: Nemhauser GL, Rinnooy Kan AHG, Todd MJ (eds) Optimization. Handbook Oper Res and Management Sci. North-Holland, Amsterdam, pp 631–662

30. Romeijn HE,Smith RL (1994) Simulated annealing and adaptive search in global optimization. Probab Eng Inform Sci 8:571–590
31. Romeijn HE, Smith RL (1994) Simulated annealing for constrained global optimization. J Global Optim 5:101–126
32. Romeijn HE, Zabinsky ZB, Graesser DL, Neogi S (1999) A new reflection generator for simulated annealing in mixed integer/continuous global optimization. J Optim Th Appl 101:403–427
33. Schoen F (1991) Stochastic techniques for global optimization: A survey of recent advances. J Global Optim 1:207–228
34. Schrack G, Borowski N (1972) An experimental comparison of three random searches. In: Lootsma F (ed) Numerical Methods for Nonlinear Optimization. Acad Press, New York, pp 137–147
35. Schumer MA, Steiglitz K (1968) Adaptive step size random search. IEEE Trans Autom Control AC-13:270–276
36. Smith RL (1984) Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. Oper Res 32:1296–1308
37. Solis FJ, WetsRJB (1981) Minimization by random search techniques. Math Oper Res 6:19–30
38. Törn AA, žilinskas A (1989) Global optimization. Springer, Berlin
39. Vanderbilt D, Louie SG (1984) A Monte Carlo simulated annealing approach to optimization over continuous variables. J Comput Phys 56:259–271
40. Wood GR, Zabinsky ZB, Kristinsdottir BP (1997) Hesitant adaptive search: the distribution of the number of iterations to converge. Techn Report, Dept Math and Computing, Central Queensland Univ 97-002
41. Zabinsky ZB, Graesser DL, Tuttle ME, Kim GI (1992) Global optimization of composite laminates using improving hit-and-run. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton University Press, Princeton, pp 343–368
42. Zabinsky ZB, Smith RL (1992) Pure adaptive search in global optimization. Math Program 53(3):323–338
43. Zabinsky ZB, Smith RL, McDonald JF, Romeijn HE, Kaufman DE (1993) Improving hit-and-run for global optimization. J Global Optim 3(2):171–192
44. Zabinsky ZB, Wood GR, Steel MA, Baritompa WP (1995) Pure adaptive search for finite global optimisation. Math Program 69:443–448

Reactive Scheduling of Batch Processes

STACY L. JANAK, CHRISTODOULOS A. FLOUDAS
Princeton University,
Princeton, USA

Article Outline

Introduction
Problem Statement
Formulation
Reactive Scheduling: Unit Shutdowns
Reactive Scheduling: New or Modified Orders
Extensions of Reactive Scheduling Formulation
Cases
Nominal Production Schedule
Case 1: Reactive Scheduling with Unit Shutdown
Conclusions
References

Introduction

The purpose of reactive scheduling is to adjust a production schedule upon the occurrence of unexpected or unforeseen events. The original schedule is usually obtained a priori and then reactive scheduling is performed upon the breakdown of a piece of equipment, when new customer orders are received, or when existing orders are modified or deleted. Thus, in order to be effective, reactive scheduling systems must be able to generate updated production schedules relatively quickly. It is not desirable to do full-scale rescheduling for every unexpected event and usually heuristic approaches are developed to achieve the desired schedule modifications. Recent reviews on scheduling approaches that include reactive scheduling issues can be found in Floudas and Lin [3,4].

The proposed scheduling formulation uses the continuous-time scheduling formulation for short-term scheduling proposed by Floudas and coworkers [5,8,9] and the medium-term scheduling framework developed by Lin et al. [10] and Janak [6]. Full-scale rescheduling of each production schedule is avoided by fixing binary variables for a subset of tasks from the original production schedule. The subset of tasks to fix is determined using a detailed set of rules that reflect the production needs and can be modified for different production facilities. The fixing of tasks results in a reduced computational effort required to solve the resulting MILP problem and thus a suitable, updated production schedule can be determined in a shorter amount of CPU time.

Problem Statement

In the batch plant investigated, there are several different types of operations (or tasks) possible and the plant has many different types of units where over 80 are modeled explicitly. Also, there are hundreds of different products that can be produced through a variety of processing recipes resulting in hundreds of different tasks. Each product is made using one of the processing recipes shown in Fig. 1 or a slight variation. The recipes are represented in the form of State-Task Network (STN), in which the *state* node is denoted by a circle and the *task* node by a rectangle.

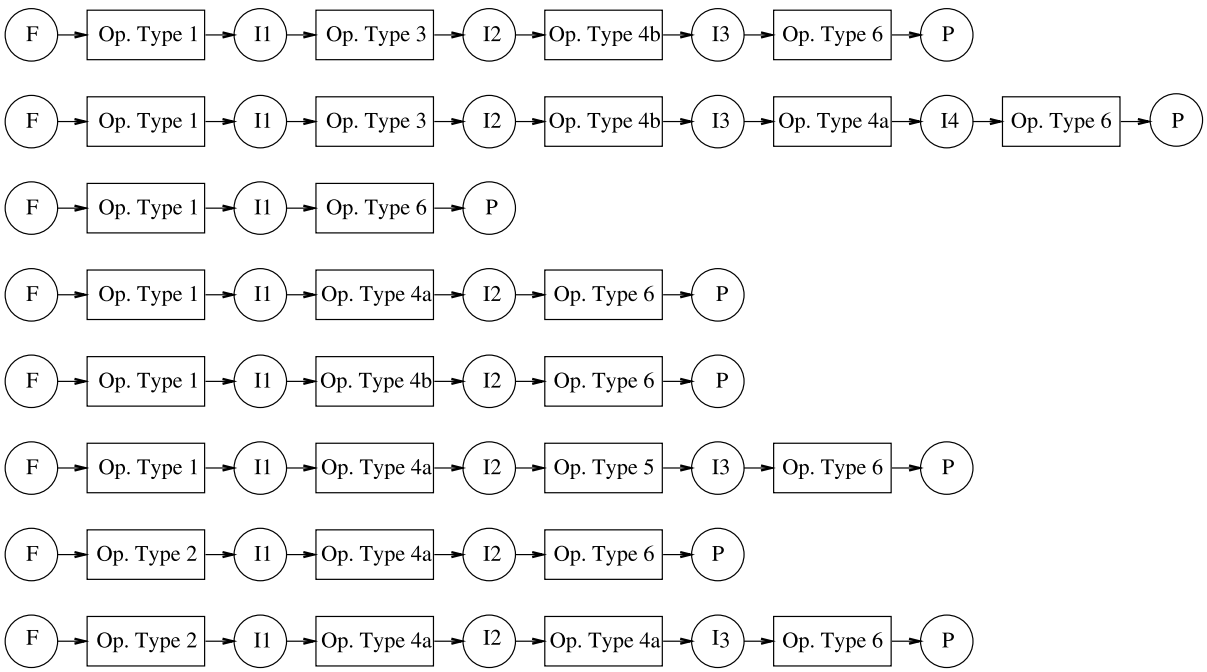
The information on which units are suitable for each product is given. All the units are utilized in a batch mode with the exception of the type 5 and 6 units, which operate in a continuous mode. The capacity limits of the type 1, 2, and 3 units vary from one product to another, while the capacity limits of the type 4a, 4b, 5, and 6 units are the same for all suitable products. The processing time or processing rate of each task in the suitable units is also specified. Additional information for the plant under investigation can be found in Janak et al. [6].

The time horizon considered for production scheduling is a few weeks or longer and customer orders are fixed throughout the time horizon with specified amounts and due dates. There is no limitation on external raw materials and we apply the zero-wait storage condition or limited intermediate storage capacity for all materials based on actual plant data. There are two different types of products produced, category 1 and 2. The solution of this medium-term scheduling problem, even for only a few days, results in a large-scale scheduling problem that can be very computationally complex. New techniques need to be developed in order to both efficiently and accurately carry out reactive scheduling.

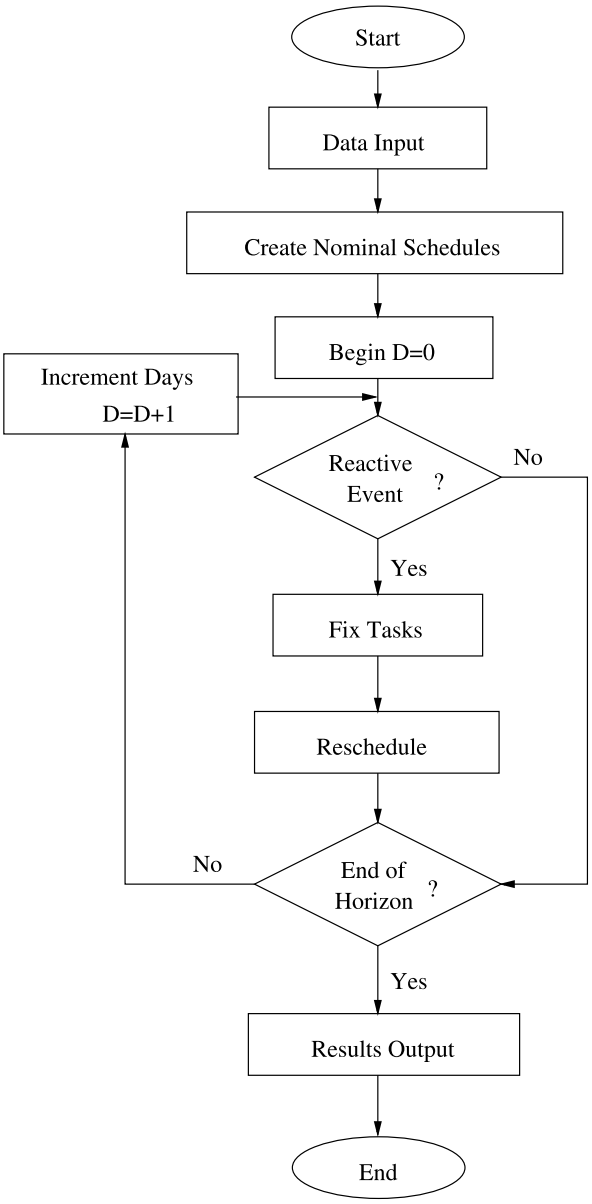
Formulation

The overall methodology for solving the reactive scheduling problem in a medium-term production scheduling framework can be summarized in the following steps. The flowchart for the overall reactive scheduling framework is shown in Fig. 2.

Step 1. Obtain a nominal production schedule for the full scheduling horizon using the medium-term



Reactive Scheduling of Batch Processes, Figure 1
State-task network (STN) representation of plant



Reactive Scheduling of Batch Processes, Figure 2
Flowchart of the reactive scheduling framework

scheduling framework proposed in Janak et al. [6] by decomposing the full scheduling horizon into smaller short-term scheduling subproblems in successive time horizons.

Step 2. Upon the realization of an unexpected event, perform the following steps.

a. Fix all the tasks in the short-term scheduling subhorizons which have already taken place using the nominal schedule.
b. For the subhorizon that is currently in production, there are two cases of unexpected events which are considered: breakdown of a unit or the addition/modification of orders. If a unit breaks down, then fix the appropriate tasks for the current subhorizon using the rules outlined in Sect. "Reactive Scheduling: Unit Shutdowns". For the case when orders are added or modified, then fix the appropriate tasks for the current subhorizon using the rules outlined in Sect. "Reactive Scheduling: New or Modified Orders". Note that a combination of both can also be performed.
c. Once the tasks have been fixed, then the overall short-term scheduling model presented in Janak et al. [6] with some modifications can be used to perform rescheduling. First, the formulation is enhanced so that the results of the level 1 decomposition model are fixed to match the results from the nominal schedules. Thus, the days in each subhorizon are fixed so that the horizons match the nominal schedules. Next, modifications need to be made to reflect changes from the unexpected events. Each of the two cases of unexpected events that can occur in a particular subhorizon requires a different modification to the model. Complete information on these modifications can be found in Sect. "Reactive Scheduling: Unit Shutdowns" and "Reactive Scheduling: New or Modified Orders"

Step 3. Once the items in Step 2 have been completed, then the horizon with the current reactive events is ready to be rescheduled. This is done in the same manner as the nominal schedule, however, a smaller time limit or integer solution limit may be employed, if desired. Next, the remaining overall time horizon is rescheduled from scratch. This is necessary since changes in the reactive time horizon can cause changes in the inventory, demand satisfaction, and unit availability which can make the nominal solution for the remaining time horizons infeasible or severely suboptimal.

Reactive Scheduling: Unit Shutdowns

For the case when a piece of equipment breaks or is shut down during a subhorizon, then, for Step 2b in the overall framework, the following tasks need to be performed. For the current subhorizon, tasks are fixed

in a particular unit at a specific event point by setting the associated binary variable to one. For the case of unit shutdown, we also fix the starting time and batch size of all tasks that are fixed. Tasks are fixed for the unit shutdown case if they start processing before the unit shutdown in all unaffected units or if they start and finish processing before the shutdown time in the unit that is shut down. For Step 2c in the overall framework, some additional information needs to be included in the model before rescheduling is performed. For the case of unit shutdown, the starting and finishing times of all tasks in the particular unit that shuts down need to be appropriately bounded so that the unit becomes unavailable during the shutdown period. Also, for the subsequent scheduling horizon, the starting time of the unit that was shut down needs to be set to either the end of the last task or to the time the unit became available again, depending on which is later. This ensures that no task will be scheduled during the blocked off time if it overlaps into the next horizon.

The mathematical formulation for reactive scheduling with unit shutdowns uses the same sets of constraints developed for short-term scheduling in Janak et al. [6] including constraints (22)–(52), (56)–(58), and the overall objective function given in constraint (66). In addition, the above additional requirements can be described mathematically as follows:

$$wv(i,j,n) = 1, \forall i \in I^{\text{fix}}, j \in J_i, n \in N \quad (1)$$

$$T^s(i,j,n) = T^s(i,j,n)^{\text{const}}, \quad \forall i \in I^{\text{fix}}, j \in J_i, n \in N \quad (2)$$

$$B(i,j,n) = B(i,j,n)^{\text{const}}, \quad \forall i \in I^{\text{fix}}, j \in J_i, n \in N \quad (3)$$

$$T^s(i,j,n), T^f(i,j,n) \le Shutdown_j^{\text{start}}, \quad \forall i \in I, j \in J^{\text{sd}}, n \in N, n \le N^{\text{int}} \quad (4)$$

$$T^s(i,j,n), T^f(i,j,n) \ge Shutdown_j^{\text{end}}, \quad \forall i \in I, j \in J^{\text{sd}}, n \in N, n > N^{\text{int}} \quad (5)$$

where I^{fix} is the set of tasks (i) to be fixed, J^{sd} is the set of units (j) which experience a shutdown, N^{int} is an intermediate event point, $T^s(i,j,n)^{\text{const}}$ is the value of the starting time of the task from the nominal scheduling run, $B(i,j,n)^{\text{const}}$ is the batch size of the task from the nominal scheduling run, $Shutdown_j^{\text{start}}$ is the beginning of the shutdown in unit (j), and $Shutdown_j^{\text{end}}$ is the end of the shutdown in unit (j).

Reactive Scheduling: New or Modified Orders

For the case when new orders are added to a scheduling horizon, then, for Step 2b in the overall framework, the following actions need to be performed. For the current subhorizon, tasks can be fixed in a particular unit at a specific event point for a variety of reasons. Some of the possible cases for fixing tasks from the nominal schedule considered in this work are as follows.

1. If the task takes place in a unit that is not suitable for any products which have new or changed orders. This is done so that production is free in any unit which may need to accommodate a new order or remove an old order.
2. If the task takes place in a unit that is heavily utilized (e. g., more than a specified percentage). This is done because well utilized units should not be rescheduled, if possible. Note that in this case, a task is only fixed in a unit that is heavily utilized if the unit does not have any tasks taking place in it which correspond to new or modified orders.
3. If the task corresponds to a product with one of the top four largest demands in the current subhorizon. This is done to help ensure that the larger demands in the horizon can be met.
4. If the task is suitable in three or less processing units. These tasks are fixed because they most likely cannot be processed anywhere else.
5. If the task occurs in a special processing unit, specific to the problem at hand. For instance, if the task occurs in a unit which has very tight production or predetermined production.
6. If the task occurs in one of the larger processing units, such as the large type 1 units, and corresponds to a product with demand that is greater than or equal to some amount. These tasks are fixed because they represent a good utilization of resources. Larger demands should be assigned to the larger processing units.

Thus, if a task satisfies one or more than one of the items above, then it is fixed in the reactive scheduling

formulation using the data from the nominal schedule. Note that a task which corresponds to a reduced or deleted order cannot be fixed, regardless of whether or not it satisfies any of the above items. For Step 2c in the overall framework, some additional information needs to be included in the model before rescheduling is performed. For the case of new orders, the demands used in the mathematical framework need to be updated to reflect any changes.

The mathematical formulation for reactive scheduling with new or modified orders also uses the same sets of constraints developed for short-term scheduling in Janak et al. [6] including constraints (22)–(52), (56)–(58), and the overall objective function given in constraint (66). In addition, the above additional requirements can be described mathematically using Eq. (1) given for reactive scheduling with unit shutdowns.

Extensions of Reactive Scheduling Formulation

As mentioned previously, it is also possible to consider a combination of reactive events. For instance, multiple units could become unavailable and multiple orders could be added or modified in a single horizon. This case can be addressed using the same methodology described in the previous section. In order to fix a task, we would need to check that the task does not correspond to a modified order and that it does not occur during a blocked time in any of the unavailable units. Then, if both of these conditions are met and if the task satisfies one or more of the above criteria, it can be fixed. Note that as more reactive events take place, most likely fewer tasks will be able be fixed in the reactive scheduling formulation, making the resulting problem more computationally complex.

Cases

In this section, an example problems is presented to demonstrate the effectiveness of the proposed approach. Additional examples can be found in Janak et al. [7]. The example considers the production scheduling of an industrial batch plant including campaign mode production. We use the nominal schedule obtained in Janak et al. [6] and we consider reactive scheduling in the event of unit shutdowns. The example is implemented with GAMS 2.50 [1] and solved using CPLEX 9.0 [2] with a 3.20 GHz Linux workstation. The dual simplex method is used with best-bound search and strong branching. The short-term scheduling horizon where reactive scheduling is performed is run with a relative optimality tolerance equal to 0.001% along with a 30 minute time limit. The subsequent short-term scheduling horizons are fully rescheduled and are run with a relative optimality tolerance equal to 0.001% along with a three hour time limit and an integer solution limit of 40.

Nominal Production Schedule

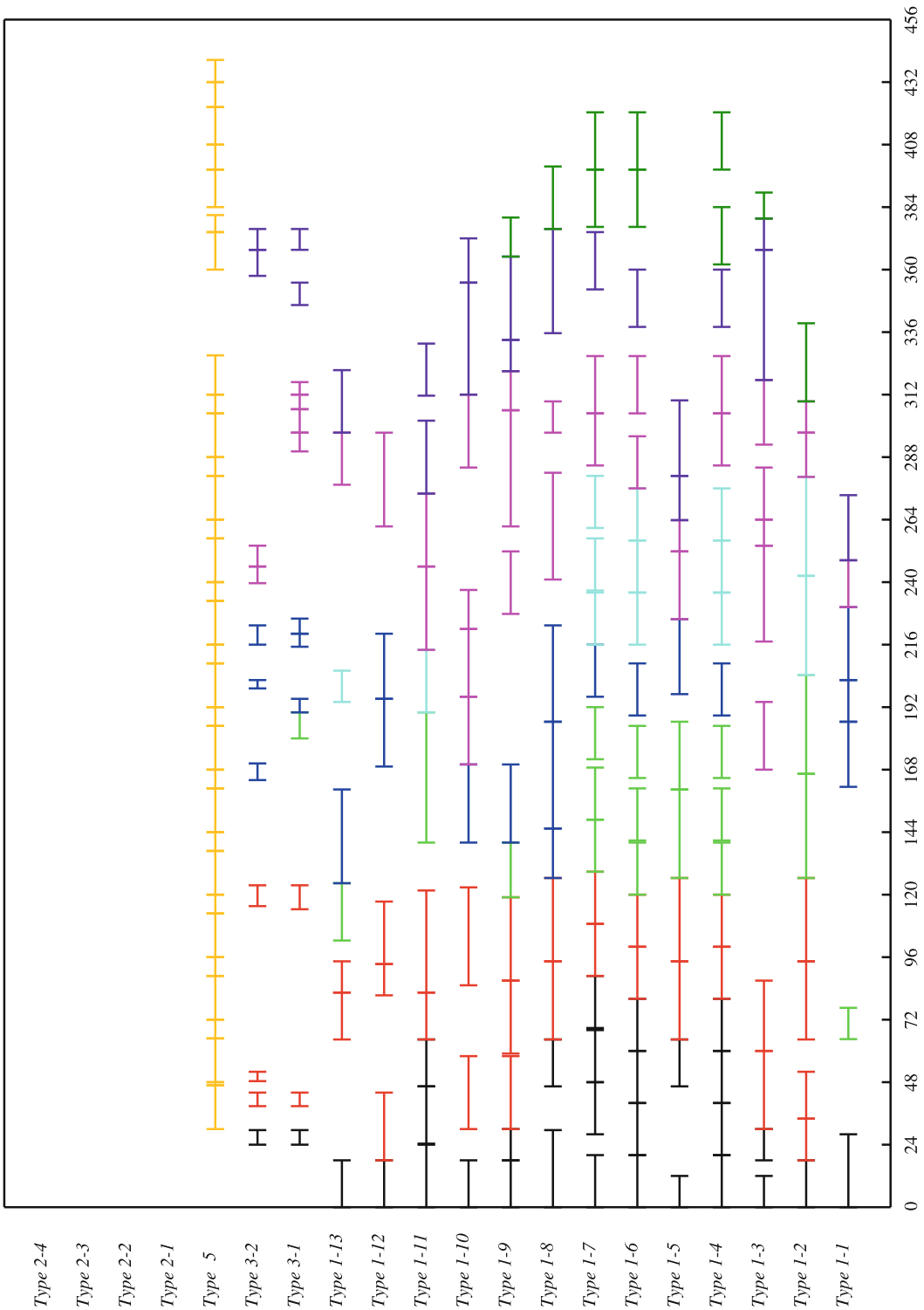
The nominal production schedule was determined in Janak et al. [6] where the total time period is 19 days, from D0 to D18. The nominal problem is solved including campaign mode production so that production in the type 5 unit is fixed to yield campaigns of predetermined length. The rolling horizon framework decomposes the time horizon into 8 individual subhorizons, each with its own products and demands. The results of the decomposition for each time horizon can be seen in Table 1.

The final production schedule for the entire time period can be seen in Fig. 3 and 4 where the processing units (operation type 1, 2, 3, and 5) are shown in the first figure and the other units (operation type 4a, 4b, and 6) are shown in the second. Each short-term scheduling horizon is represented with a different color.

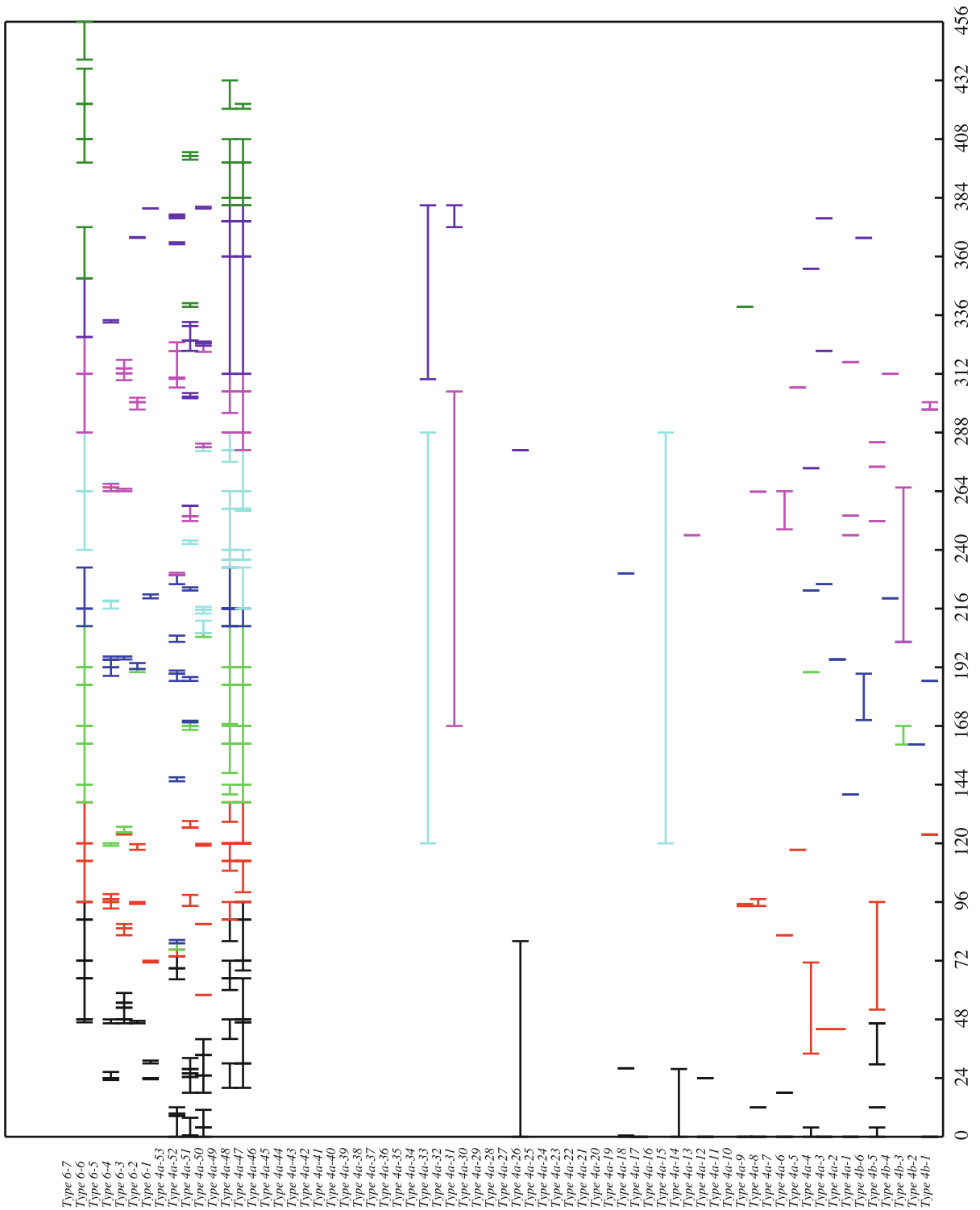
The total demand for the entire 14-day period is 2323.545 and the total production is 2719.859, where 15.00 of the demands are not met. The production schedules obtained satisfy demands for all but one product, though some due dates are relaxed, and also

Reactive Scheduling of Batch Processes, Table 1
Decomposition results for nominal problem

	Days	Event Points	Main Products	Additional Products
H1	D0–D2	9	27	1
H2	D3–D4	7	31	0
H3	D5–D6	9	51	0
H4	D7–D8	6	53	0
H5	D9–D10	10	43	0
H6	D11–D12	9	53	0
H7	D13–D14	6	56	0
H8	D15–D18	10	40	0



Reactive Scheduling of Batch Processes, Figure 3
Overall production schedule for processing units for nominal problem



Reactive Scheduling of Batch Processes, Figure 4
Overall production schedule for non-processing units for nominal problem

Reactive Scheduling of Batch Processes, Table 2
Unit utilization statistics for nominal problem

Unit	Time Used (h)	Time Left (h)	Percent Utilized
Type 1-1	152.00	304.00	33.33%
Type 1-2	327.00	129.00	71.71%
Type 1-3	270.60	185.40	59.21%
Type 1-4	368.00	88.00	80.70%
Type 1-5	264.80	191.20	58.07%
Type 1-6	368.00	88.00	80.70%
Type 1-7	368.00	88.00	80.70%
Type 1-8	323.60	132.40	70.96%
Type 1-9	311.60	144.40	68.33%
Type 1-10	268.60	187.40	58.90%
Type 1-11	303.20	152.80	66.49%
Type 1-12	167.00	289.00	36.62%
Type 1-13	162.00	294.00	35.53%
Type 5	374.52	81.48	82.13%

produce 17.06% more material than the demands require. Note that many of the processing units are not fully utilized, as shown in Table 2, indicating the potential for even more production in the given time period which may be incorporated using reactive scheduling.

Case 1: Reactive Scheduling with Unit Shutdown

In the example, we consider reactive scheduling for the first time horizon where several units are unavailable for a period of time. We will use the nominal production schedule as our base schedule where the first time horizon covers three days (i. e., D0 to D2) and utilizes nine event points. The three different unit shutdowns considered can be seen in Table 3.

The reactive scheduling framework for unit shutdowns fixes tasks that start before the latest shutdown start time in units which are unaffected and fixes tasks that start and end before the shutdown time in units which experience a shutdown. Thus, for this example, all the tasks starting before time 48 hours in all the units except Type 1-3, Type 1-5, and Type 1-11 are fixed. This means that both the binary variable ($wv(i, j, n)$), starting time ($ts(i, j, n)$), and batch size ($b(i, j, n)$) of the task need to be fixed for each task (i) in those units (j) at the event point (n) they occurred in the nominal production schedule. Thus, if we consider the nominal production schedule for the first time horizon shown in Fig. 5, then the tasks shown in Table 4 are fixed in each unit.

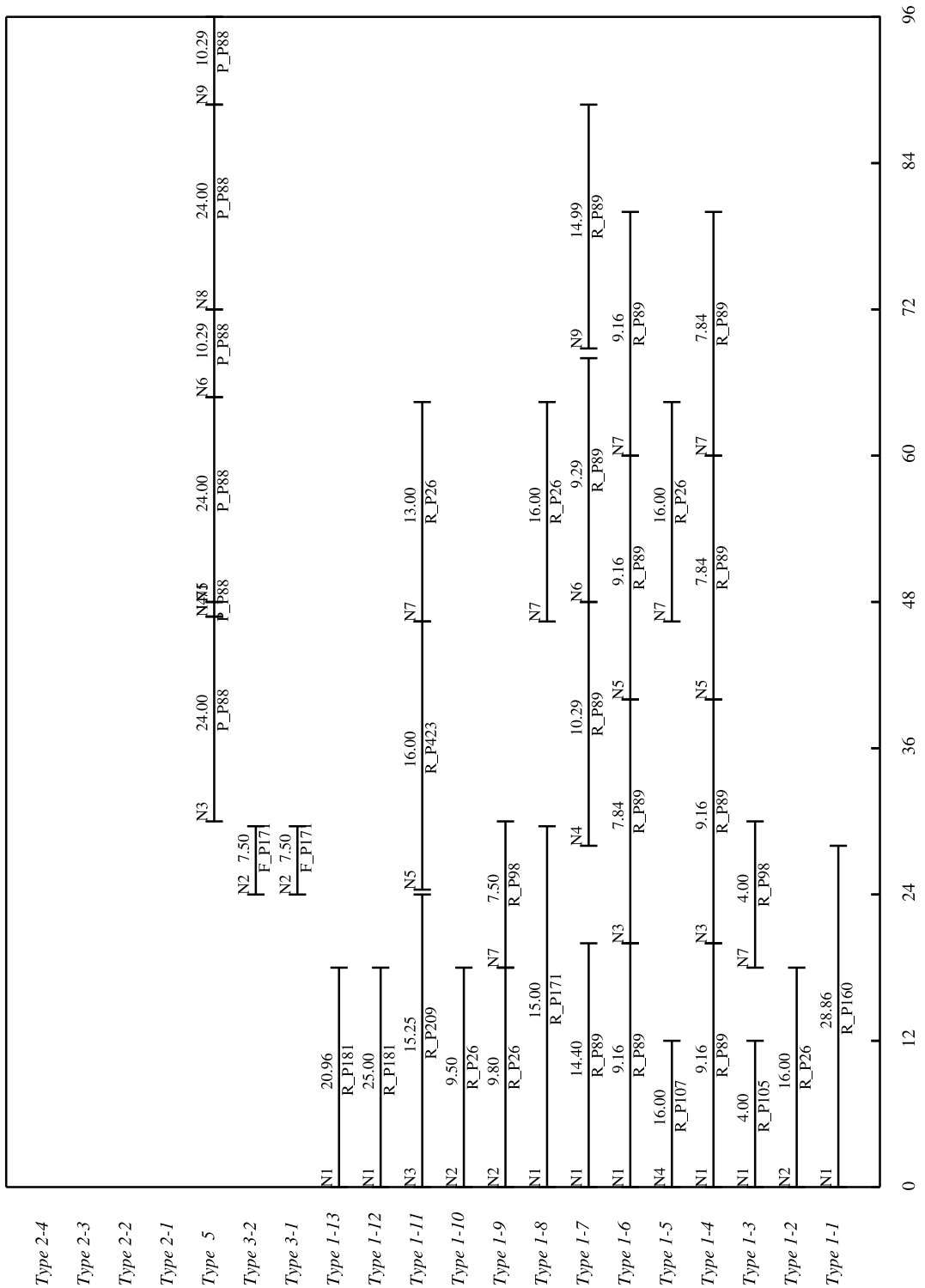
Reactive Scheduling of Batch Processes, Table 3
Unit shutdowns for case 1

Unit	Unavailable Start Time (h)	Unavailable End Time (h)
Type 1-3	24	48
Type 1-5	36	72
Type 1-11	48	72

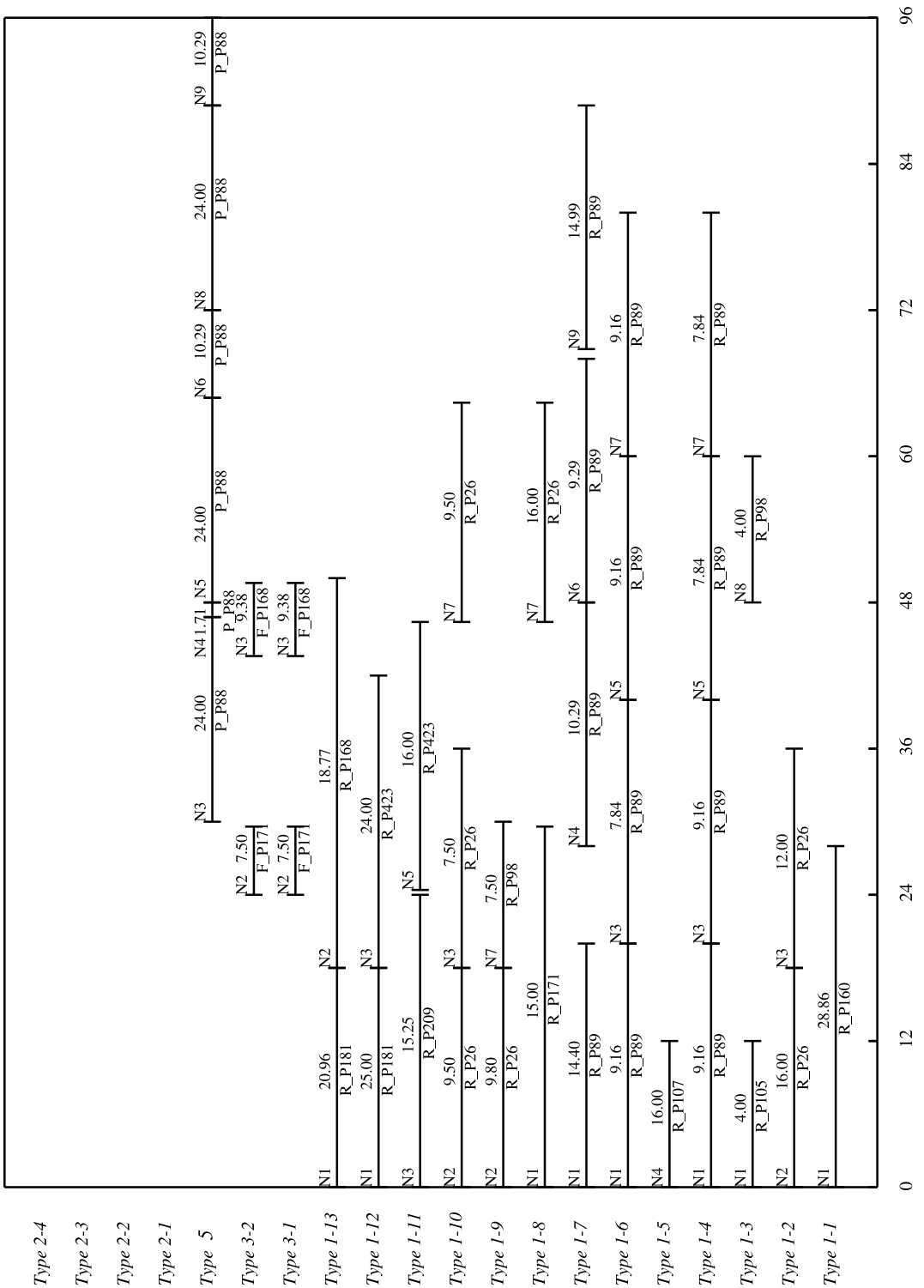
In addition, in each of the units which experience a shutdown, bounds need to be placed on the starting and finishing times for all tasks at event points which do not already have tasks fixed to them. An intermediate event point is chosen and all of the event points before and including the intermediate event point have an upper bound placed on the starting and finishing times of tasks to be less than or equal to the start of the shutdown. Similarly, all of the event points after the intermediate event point have a lower bound placed on the starting and finishing times of tasks to be greater than or equal to the end of the shutdown. Note that the intermediate event point is unit specific and is determined so that there are approximately the same number of event points before and after the shutdown while excluding event points that already have tasks assigned to them.

Reactive Scheduling of Batch Processes, Table 4
Tasks to fix in case 1

Unit	Task	Event Point
Type 1-1	R_P160	N1
Type 1-2	R_P26	N2
Type 1-3	R_P105	N1
Type 1-4	R_P89	N1, N3, N5
Type 1-5	R_P107	N4
Type 1-6	R_P89	N1, N3, N5
Type 1-7	R_P89	N1, N4, N7
Type 1-8	R_P171	N1
	R_P26	N7
Type 1-9	R_P26	N2
	R_P98	N7
Type 1-10	R_P26	N2
Type 1-11	R_P209	N3
	R_P423	N5
Type 1-12	R_P181	N1
Type 1-13	R_P181	N1



Reactive Scheduling of Batch Processes, Figure 5
Nominal production schedule for the processing units in the first horizon



Reactive Scheduling of Batch Processes, Figure 6
Reactive production schedule for the processing units in the first horizon from case 1

Reactive Scheduling of Batch Processes, Table 5
Bounds on timing in units in case 1

Unit	Fixed Event Points	Bounds
Type 1-3	N1	$ts(i,j,n), tf(i,j,n) \leq 24$ for N2, N3, N4, N5 $ts(i,j,n), tf(i,j,n) \geq 48$ for N6, N7, N8, N9
Type 1-5	N4	$ts(i,j,n), tf(i,j,n) \leq 36$ for N1, N2, N3, N5 $ts(i,j,n), tf(i,j,n) \geq 72$ for N6, N7, N8, N9
Type 1-11	N3, N5	$ts(i,j,n), tf(i,j,n) \leq 48$ for N1, N2, N4, N6 $ts(i,j,n), tf(i,j,n) \geq 72$ for N7, N8, N9

Thus, considering the nominal production schedule for the first time horizon, the bounds shown in Table 5 are placed for each of the units which experience a shutdown.

Then, fixing the above sets of tasks from Table 4 and imposing the bounds defined in Table 5, we obtain a reactive production schedule which excludes production in the affected units during the shutdown periods and maintains all of the production that has already taken place. The reactive schedule for this example can be seen in Fig. 6. Note that once a reactive schedule is obtained for the first time horizon which incorporates the unit shutdown information, the subsequent horizons must each rescheduled from scratch due to the changes in inventory, demand satisfaction, and unit availability.

Conclusions

In this chapter, we presented a reactive scheduling framework which provides an immediate response to unexpected events such as equipment breakdown or the addition or modification of orders. The reactive scheduling formulation takes into account the schedule currently in progress as well as planned productions that are not affected by the unexpected event. The proposed mathematical framework utilizes an efficient MILP mathematical framework developed for short-term scheduling problems with modifications introduced to reflect the effects of the unforeseen event. To avoid full rescheduling of the current production schedule, the formulation determines tasks which are not affected by the unforeseen event, either directly or indirectly, and can be carried out as scheduled. The resulting tasks along with additional subsets of tasks are then fixed in the MILP problem and the rest of the horizon is rescheduled. We considered two types of unexpected events: unit shutdown and the addition or modification of orders, as well as their combination. All the cases of unexpected events utilize the nominal production schedule, the original MILP formulation for short-term scheduling with modifications, and a program to determine which tasks may be fixed before rescheduling. The formulation is then able to determine an updated production schedule for the remaining time horizon in a reasonable amount of CPU time. Reactive scheduling of a large-scale industrial batch plant was performed to demonstrate the effectiveness of the proposed approach. Results indicate that the reactive scheduling framework is capable of determining updated production schedules to account for unexpected events and can also be used to improve existing production schedules.

References

1. Brooke A, Kendrick D, Meeraus A, Raman R (2003) GAMS: A User's Guide. South San Francisco
2. CPLEX (2005) ILOG CPLEX 9.0 User's Manual.
3. Floudas CA, Lin X (2004) Continuous-Time versus Discrete-Time Approaches for Scheduling of Chemical Processes: A Review. Comput Chem Eng 28:2109
4. Floudas CA, Lin X (2005) Mixed Integer Linear Programming in Process Scheduling: Modeling, Algorithms, and Applications. Ann Oper Res 139:131
5. Ierapetritou MG, Floudas CA (1998) Effective Continuous-Time Formulation for Short-Term Scheduling: 1. Multipurpose Batch Processes. Ind Eng Chem Res 37:4341
6. Janak SL, Floudas CA, Kallrath J, Vormbrock N (2006) Production Scheduling of a Large-Scale Industrial Polymer Compounding Plant: I. Short-Term and Medium-Term Scheduling. Ind Eng Chem Res 45:8234
7. Janak SL, Floudas CA, Kallrath J, Vormbrock N (2006) Production Scheduling of a Large-Scale Industrial Polymer Compounding Plant: II. Reactive Scheduling. Ind Eng Chem Res 45:8253
8. Janak SL, Lin X, Floudas CA (2004) Enhanced Continuous-Time Unit-Specific Event-Based Formulation for Short-Term Scheduling of Multipurpose Batch Processes: Resource Constraints and Mixed Storage Policies. Ind Eng Chem Res 43:2516
9. Lin X, Floudas CA (2001) Design, Synthesis and Scheduling of Multipurpose Batch Plants via an Effective Continuous-Time Formulation. Comput Chem Eng 25:665
10. Lin X, Floudas CA, Modi S, Juhasz NM (2002) Continuous-Time Optimization Approach for Medium-Range Production Scheduling of a Multiproduct Batch Plant. Ind Eng Chem Res 41:3884

Redundancy in Nonlinear Programs

RICHARD J. CARON
University Windsor, Windsor, Canada

MSC2000: 90C05, 90C20

Article Outline

Keywords
See also
References

Keywords

Redundancy; Minimal representation

A typical *nonlinear program* is to find a point in a *feasible region* that will minimize an *objective function*. The feasible region is often represented by a finite set of algebraic inequalities called *constraints*. If a constraint can be removed from the set without causing a change to the feasible region it is said to be *redundant* with respect to the remaining constraints. A constraint that is not redundant is said to be *nonredundant*, *irredundant*, or *necessary*.

A more formal definition requires some notation. Let $R \subseteq \mathbf{R}^n$ denote a nonempty feasible region. The set of constraints indexed by $I = \{1, \ldots, m\}$ is given by $G(I) = \{g_i \leq 0 : i \in I\}$, where $g_i(x)$: $\mathbf{R}^n \to \mathbf{R}$, and the region determined by $G(I)$ is $R(I) = \{x: g_i(x) \leq 0, i \in I\}$. Suppose that $R = R(I)$. We then call $G(I)$ a *representation* of R. The kth inequality constraint '$g_k(x) \leq 0$' is redundant with respect to $G(I_k)$ if $R(I_k) = R(I)$, where $I_k = I - \{k\}$. Further definitions, such as those for *relatively redundant constraints*, *weakly necessary constraints*, etc., can be found in [3,5].

Of course, there can be more than one redundant constraint. If $\widehat{I}$ is a proper subset of I, and if $R(\widehat{I}) = R(I)$, the constraint set $G(\widehat{I})$ is called a *reduction* of $G(I)$. Further, $G(\widehat{I})$ is called a *common dependency set* for the set of redundant constraints $G(I - \widehat{I})$ [8]. If all the constraints in $G(\widehat{I})$ are nonredundant, then $G(\widehat{I})$ it is called a *prime representation* of the feasible region. A related concept is that of a *minimal representation*.

For linear constraints, i. e., when $g_i(x) = a_i^\top x - b_i$, where $a_i \in \mathbf{R}^n$ and $b_i \in \mathbf{R}$, a minimal representation is one having the fewest constraints. In [11] it is proved that a representation is minimal if and only if it contains no redundant constraints and no *implicit equality constraints*. The constraint '$g_k(x) \leq 0$' is an implicit equality in $G(I)$ if $g_k(x) = 0$ for all $x \in R(I)$. In order to obtain a minimal representation the implicit equalities must first be replaced with explicit equalities, and then the redundant constraints must be removed one at a time. (The definition of redundancy is easily modified to include equality constraints.)

For the quadratic case, i. e., when $g_i(x) = a_i^\top x + (1/2)\, x^\intercal H_i\, x - b_i$, where the H_i are symmetric, positive definite matrices, a minimal representation is defined as one having the least number of quadratic constraints, and, among those with the same number of quadratic constraints, the least number of linear constraints. It was proved in [10] that a representation was minimal if and only if it contained no redundant constraints, no implicit equality constraints, and no pseudoquadratic constraints. A *pseudoquadratic constraint* is one that can be replaced by a finite number of linear constraints without causing a change to the feasible region. For example, since $\{x \in \mathbf{R}^2: x_2 = 0, x_1^2 + x_2^2 \leq 1\}$ equals $\{x \in \mathbf{R}^2: x_2 = 0, -1 \leq x_1 \leq 1\}$, the constraint '$x_1^2 + x_2^2 \leq 1$' was pseudoquadratic.

Algorithms used to detect redundant constraints can usually be classified as either deterministic methods or probabilistic methods. The *deterministic methods* are optimization based. Consider the nonlinear program $\max \{g_k(x): x \in R(I_k)\}$. If there is no solution; that is, if the program is either *unbounded* or *infeasible* ($R(I_k) = \emptyset$), then it follows from the definition of redundancy that the kth constraint is necessary. Otherwise the program has a solution x^*. If $g_k(x^*) > 0$, then again the kth constraint is necessary, and if $g_k(x^*) \leq 0$, the constraint is redundant. This method can work quite well in the linear case [6,9] as each of the constraints can be classified by solving a *linear program*. Consequently, for the linear case, the redundancy problem is polynomial. The importance of redundancy detection to the solution of large sparse linear programs is discussed in [1]; and the importance of redundancy detection for model analysis is discussed in [7]. For the nonlinear case, this approach has the drawback that it requires the solution of nonconvex programs. For example, if all the constraint functions are convex, then $\max \{g_k(x) : x \in R(I_k)\}$ is a nonconvex program.

An alternative to the deterministic, optimization based methods are the *probabilistic methods*. The first

such method, which became known as the *hyperspheres direction hit and run* method was introduced in [4]. (A description of the method can be found in [9].) This technique generates a sequence of random lines that pass through the feasible region. The lines are generated as follows. A given feasible point $x \in R$ and a direction s, sampled from a uniform distribution over the surface of the unit hypersphere, determines the line $\{x + \sigma\ s : \sigma \in \mathbf{R}\}$. The next feasible point is sampled uniformly from the feasible segment of the line. Note that in order to determine the feasible line segment the intersection points of the line with all the constraint boundaries must be calculated. Under appropriate assumptions, the constraints hit by the feasible line segment are nonredundant. This technique requires a *stopping rule* after which all constraints that have not been hit are classified, possibly with error, as redundant. The main advantages of hit and run methods are that they can very quickly identify most of the necessary constraints, and that they can be applied to a large class of nonlinearly constrained regions. The main disadvantages are the need for an initial feasible point, and the need to calculate the intersection points.

For general nonlinear programs it may well be that neither the deterministic nor the hit and run methods are applicable. For these problems, there is an algorithm [2] based on a connection between the redundancy problem and the constraints in a *set covering problem*. This method only requires that for any $x \in \mathbf{R}^n$ it can be determined if $g_i(x)$ is negative or nonnegative. In fact, the method does not even require a nonempty feasible region. This technique is probabilistic in that it samples points from $\mathbf{R}^n$. Each point that is sampled is used to generates an m-bit binary word whose kth bit is unity if and only if the kth constraint is violated at that point. Upon termination of the sampling process, the collection of all the generated binary words form the rows of a set covering matrix E having m columns, one corresponding to each constraint. Let y be a nontrivial feasible solution to the set-covering problem, i. e., $Ey \geq e$, where e is a vector of 'ones', and where y has at least one zero component. Then $G(\widehat{I})$ is a reduction of $G(I)$, where $i \in \widehat{I}$ if and only if $y_i = 1$. An objective function can be introduced to the set-covering problem so that any set covering heuristic can be used to find an approximately minimal representation. Here the definition of minimal will depend on the choice of objective. The main disadvantage to the approach is the need to generate the set covering rows. The main advantage is its general applicability.

See also

- ► Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- ► Inequality-constrained Nonlinear Optimization

References

1. Andersen ED, Andersen KD (1995) Presolving in linear programming. Math Program 71:221–245
2. Boneh A (1984) Identification of redundancy by a set-covering equivalence. In: Brans JP (ed) Operational Research'84. Elsevier, Amsterdam, pp 407–422
3. Boneh A, Boneh S, Caron RJ (1993) Constraint classification in mathematical programming. Math Program 61(1):61–74
4. Boneh A, Golan A (1979) Constraints redundancy and feasible region boundedness by a random feasible point generator (RFPG). Paper Presented at EURO III
5. Caron RJ, Boneh A, Boneh S (1997) Redundancy. In: Greenberg HJ, Gal J (eds) Recent Advances in Sensitivity Analysis and Parametric Programming. Kluwer, Dordrecht, pp 13.1–13.43
6. Caron RJ, McDonald JF, Ponic CM (1989) A degenerate extreme point strategy for the classification of linear inequalities as redundant or necessary. J Optim Th Appl 62:225–237
7. Greenberg HJ (1994) How to analyze results of linear programs – 4: Forcing substructures. Interfaces 24:121–130
8. Greenberg HJ (1996) Redundancy, implicit equalities, and infeasible sets. Ann Math Artificial Intelligence 17:37–83
9. Karwan MH, Lotfi V, Telgen J, Zionts S (eds) (1983) Redundancy in mathematical programming. Springer, Berlin
10. Obuchowska WT, Caron RJ (1995) Minimal representation of quadratically constrained convex feasible regions. Math Program 68:169–186
11. Telgen J (1982) Minimal representation of convex polyhedral sets. J Optim Th Appl 38:1–24

Reformulation-Linearization Technique for Global Optimization

HANIF D. SHERALI[1], LEO LIBERTI[2]

[1] Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, USA

[2] LIX, Ecole Polytechnique, Palaiseau, France

MSC2000: 90C26

Article Outline

Keywords
References

Keywords

Reformulation-linearization technique; Lift-and-project; Tight relaxations; Valid inequalities; Model reformulation; Convex hull; Convex envelopes; Mixed-integer 0-1 program; Polynomial programs; Nonconvex programs; Factorable programs; Reduced relaxations

Discrete and continuous nonconvex programming problems arise in a host of practical applications in the context of production planning and control, location-allocation, distribution, economics and game theory, quantum chemistry, and process and engineering design situations. Several recent advances have been made in the development of branch-and-cut type algorithms for mixed-integer linear and nonlinear programming problems, as well as polyhedral outer-approximation methods for continuous nonconvex programming problems. At the heart of these approaches is a sequence of linear (or convex) programming relaxations that drive the solution process, and the success of such algorithms is strongly tied in with the strength or tightness of these relaxations.

The Reformulation-Linearization-Technique (RLT) is a method that generates such tight linear programming relaxations for not only constructing exact solution algorithms, but also to design powerful heuristic procedures for large classes of discrete combinatorial and continuous nonconvex programming problems. Its development originated in [4,5,6], initially focusing on 0-1 and mixed 0-1 linear and polynomial programs [21,22], and later branching into the more general family of continuous, nonconvex polynomial programming problems [18,45,49]. For the family of mixed 0-1 linear (and polynomial) programs in n 0-1 variables, the RLT generates an n-level hierarchy, with the n-th level providing an explicit algebraic characterization of the convex hull of feasible solutions [21,22]. The RLT essentially consists of two steps—a reformulation step in which certain additional nonlinear valid inequalities are automatically generated, and a linearization step in which each product term is replaced by a single continuous variable. The level of the hierarchy directly corresponds to the degree of the polynomial terms produced during the reformulation stage. Hence, in the reformulation phase, given a value of the level $d \in \{1, \ldots, n\}$, the RLT constructs various polynomial factors of degree d comprised of the product of some d binary variables x_j or their complements $(1 - x_j)$. These factors are then used to multiply each of the defining constraints in the problem (including the variable bounding restrictions), to create a (nonlinear) polynomial mixed-integer zero-one programming problem. Suitable additional constraint-factor products can be used to further enhance the procedure. In general, for a variable restricted to lie in the interval $[l_j, u_j]$, the nonnegative expressions $(x_j - l_j)$ and $(u_j - x_j)$ are referred to as *bound-factors*, and for a structural inequality $\alpha x \geq \beta$, for example, the expression $(\alpha x - \beta)$ is referred to as a *constraint-factor*; implied product constraints can be generated using either bound-factors or constraint-factors. After using the relationship $x_j^2 = x_j$ for each binary variable x_j, $j \in \{1, \ldots, n\}$, which in effect accounts for the tightening of the relaxation, the linearization phase substitutes a single variable w_J (respectively, v_{Jk}), in place of each nonlinear term of the type $\prod_{j \in J} x_j$ (respectively, $y_k \prod_{j \in J} x_j$), where y represents the set of continuous variables. Hence, relaxing integrality, the nonlinear polynomial problem is linearized into a higher dimensional polyhedral set X_d defined in terms of the original variables (x, y) and the new variables (w, v). Denoting the projection of X_d onto the space of the original (x, y)-variables as X_{P_d}, it is shown that as d varies from 1 to n, we get,

$$X_{P_0} \supseteq X_{P_1} \supseteq X_{P_2} \supseteq \ldots \supseteq X_{P_n} = \text{conv}(X)\,,$$

where X_{P_0} is the ordinary linear programming relaxation, and conv(X) represents the convex hull of the original feasible region X. An extension of this development to the case of general integer/discrete variables is presented in [7,25], where the bound-factors are replaced by suitable Lagrange interpolating polynomials, and a further extension to 0-1 mixed-integer as well as general mixed-discrete semi-infinite and bounded convex programming problems is presented in [26] (see also [50]). Lovasz and Shrijver [16] and Boros et al. [9] have also independently developed various concepts related to the RLT process. This RLT

process has also been extended and enhanced in [27] through the use of more generalized constraint-factors that imply the bounding restrictions $0 \le x_j \le 1$ for $j \in \{1, \ldots, n\}$. A similar hierarchy of relaxations leading to the convex hull representation is obtained based on the use of these generalized factors in the reformulation phase, in lieu of simply the bound-factors x_j and $(1 - x_j)$, for $j \in \{1, \ldots, n\}$. In addition, this hierarchy embeds within its construction stronger logical implications than only $x_j^2 = x_j$, $\forall j \in \{1, \ldots, n\}$. For example, consider an RLT constraint that has been generated by taking the product of some nonnegative polynomial factor F with a defining constraint $\gamma^\top x \ge \delta$ to yield $[F(\gamma^\top x - \delta)]_L \ge 0$, where $[\cdot]_L$ denotes the linearization of the polynomial expression $[\cdot]$ under the RLT substitution process. Then, this constraint can be tightened by deriving a stronger valid inequality of the type $\hat{\gamma}^\top x \ge \hat{\delta}$ under the condition that $F > 0$, and then imposing the RLT constraint $[F(\hat{\gamma}^\top x - \hat{\delta})]_L \ge 0$, which is valid whenever $F = 0$ or $F > 0$. The resulting overall RLT process is shown in [27] to not only subsume the previous development, but also provide the opportunity to exploit frequently-arising special structures such as generalized/variable upper bounds, covering, partitioning, and packing constraints, as well as sparsity.

The hierarchy of higher-dimensional representations produced in this manner markedly strengthens the usual continuous relaxation, as is evidenced not only by the fact that the convex hull representation is obtained at the highest level, but that in computational studies on many classes of problems, even the first level representation helps design algorithms that significantly dominate existing procedures in the literature [4,6,20,27,30,41]. Based on a special case of the RLT process that employs the bound-factors for only a single variable at a time, Balas et al. [8] describe a *lift-and-project* cutting plane algorithm that is shown to produce encouraging results. The theoretical implications of this hierarchy are noteworthy; the resulting representations subsume and unify many published linearization methods for nonlinear 0-1 programs, and the algebraic representation available at level n promotes new methods for identifying and characterizing facets and valid linear inequalities in the original variable space, as well as for providing information that directly bridges the gap between discrete and continuous sets [3,38,40]. Indeed, since the level-n formulation characterizes the convex hull, all valid inequalities in the original variable space must be obtainable via a suitable projection; thus such a projection operation serves as an all-encompassing tool for generating valid inequalities. References [38,40] provide discussions on generating facets and tight valid inequalities for several classes of problems. Reference [3] discusses persistency issues for certain constrained and unconstrained pseudo-Boolean programming problems whereby variables that take on 0-1 values at an optimum to an RLT relaxation would persist to take on these same values at an optimum to the original problem. References [1,2,13,34,39,42], respectively discuss the use of RLT to generate improved model representations for the set partitioning, quadratic assignment, traveling salesman problems, and to 0-1 mixed-integer programs subject to various assignment constraints.

Although the Reformulation-Linearization Technique was originally designed to employ factors involving 0-1 variables in order to generate 0-1 (mixed-integer) polynomial programming problems that are subsequently re-linearized, the approach has also been extended to solve continuous, bounded variable polynomial programming problems. Problems of this type involve the optimization of a polynomial objective function subject to polynomial constraints in a set of continuous, bounded variables, and arise in numerous applications in engineering design, production, location, and distribution problems. Reference [45] prescribes an RLT process that employs suitable polynomial-factors (bound-factors based on bounding restrictions $l_j \le x_j \le u_j$, $j \in \{1, \ldots, n\}$, as well as constraint factors) to generate additional polynomial constraints through a multiplication process, which upon linearization through variable redefinitions as above, produces a linear programming relaxation. The resulting relaxation is used in concert with a suitable designed partitioning technique that attempts to reduce the error between the original nonlinear and their resulting linearized terms, in order to develop an algorithm that is proven to converge to a global optimum for this problem. Special classes of polynomial constraints based on grid factors, Lagrange interpolating polynomials, and mean value theorem constraints can be generated to further tighten these RLT relaxations [48]. In some cases (e. g., see [47]), it is benefi-

cial to retain certain simple convex constraints in the relaxation, resulting in a more general *Reformulation-Linearization/Convexification Technique*. Additionally, Sherali and Fraticelli [35] have proposed a class of semidefinite cuts based on semidefinite relaxation enhancements that can be used to significantly tighten RLT representations. While RLT essentially operates on polynomial functions having integral exponents, many engineering design applications lead to polynomial programs having general rational exponents. For such problems, a global optimization technique has been designed [18] by introducing a new level of approximation at the reformulation step, and accordingly, redesigning the partitioning scheme in order to induce the overall sequence of relaxations generated to become exact in the limit. Further extensions for solving nonlinear factorable programs for which the objective and constraint functions involve sums of products of univariate functions have also been developed [49]. Here, suitable under/over-approximating nonconvex polynomial functions are derived for the defining univariate functions in the problem, and then an appropriate partitioning scheme is devised that drives the errors from these approximations and those for the RLT process applied to the resulting polynomial program simultaneously to zero in the limit, in order to obtain a global optimum for the given factorable program. For nonconvex programs that are defined in terms of black-box functions, a new concept of a pseudo-global RLT approach has been developed by Sherali and Ganesan [36], which has been successfully applied to the design of containerships.

A special application of the RLT to mixed-integer quadratic problems subject to linear equality constraints that yields exact reformulations having fewer quadratic terms and some additional supporting RLT constraints has been developed to produce tighter convex relaxations [10,11,12,14,15]. More precisely, we multiply a subset of equality constraints $Ax = b$ by an appropriate subset of problem variables $\{x_k \mid k \in K\}$, to obtain a *reduced RLT system* $\forall k \in K(Aw_k = bx_k)$, where $w_k \equiv (x_k x_1, \ldots, x_k x_n)$ for all $k \in K$. This is equivalent to the homogeneous linear system $\forall k \in K(Az_k = 0)$ where $z_k = (w_{k1} - x_k x_1, \ldots, w_{kn} - x_k x_n)$, which may be written in a more compact way as $A'z = 0$. If we partition A' into basic and nonbasic submatrices B, N, and accordingly partition z into z_B and z_N, we have $(B|N)z = 0$, whence $Nz_N = 0$ implies that $Bz_B = 0$. We therefore conclude that enforcing the reduced RLT system and the subset of quadratic relations $w_{ki} = x_k x_i$ for (k, i) corresponding to nonbasic columns of N is enough to infer $w_{ki} = x_k x_i$ for all (k, i). In other words, by letting the RLT process ensure that $z_N = 0$, we automatically obtain as an implication of the RLT linearized constraints that the quadratic relation $z_B = 0$ will hold true as well.

For the continuous case, there exist special instances where RLT can produce convex hull or convex envelope representations [17,28]. Various classes of applications have been studied for which specialized RLT designs have been used to develop enhanced effective algorithms. This list, which is ever expanding, includes bilinear programming problems [15,28], general indefinite quadratic programming problems [12,13,47], location-allocation problems employing different distance metrics [20,29,41,46], water distribution network design problems [43,44], the solution of Hartree-Fock equations in quantum chemistry [14], the linear complementarity problem [37], and hard and fuzzy clustering problems [31,32]. References [11,19,23,24,25,33] provide expository discussions and a survey of RLT theory and applications.

References

1. Adams WP, Guignard M, Hahn PM, Hightower WL (2007) A Level-2 Reformulation-Linearization Technique Bound for the Quadratic Assignment Problem. Eur J Oper Res 180(3):983–996
2. Adams WP, Johnson TA (1994) Improved linear programming-based lower bounds for the quadratic assignment problem. In: Pardalos PM, Wolkowicz H (eds) Quadratic Assignment and Related Problems. DIMACS Series in Discret Mathematics and Theoretical Computer Science, vol 16. Providence, pp 43–75
3. Adams WP, Lassiter JB, Sherali HD (1998) Persistency in 0-1 optimization. Math Oper Res 23(2):359–389
4. Adams WP, Sherali HD (1986) A tight linearization and an algorithm for zero-one quadratic programming problems. Manage Sci 32(10):1274–1290
5. Adams WP, Sherali HD (1990) Linearization strategies for a class of zero-one mixed integer programming problems. Oper Res 38(2):217–226
6. Adams WP, Sherali HD (1993) Mixed-integer bilinear programming problems. Math Program 59(3):279–306
7. Adams WP, Sherali HD (2005) A hierarchy of relaxations leading to the convex hull representation for general discrete optimization problems. Ann Oper Res 140(1):21–47

8. Balas E, Ceria S, Cornuejols G (1993) A lift-and-project cutting plane algorithm for mixed 0-1 programs. Math Program 58(3):295–324
9. Boros E, Crama Y, Hammer PL (1990) Upper bounds for quadratic 0-1 maximization problems. Oper Res Lett 9: 73–79
10. Liberti L (2004) Reduction constraints for the global optimization of NLPs. Int Trans Oper Res 11(1):34–41
11. Liberti L (2004) Reformulation and convex relaxation techniques for global optimization. 4OR 2:255–258
12. Liberti L (2005) Linearity embedded in nonconvex programs. J Glob Optim 33(2):157–196
13. Liberti L (2007) Compact linearization of binary quadratic problems. 4OR. doi: 10.1007/s10288-006-0015-3
14. Liberti L, Lavor C, Maculan N, Chaer Nascimento MA: Reformulation in mathematical programming: an application to quantum chemistry. Discret Appl Math, submitted
15. Liberti L, Pantelides CC (2006) An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. J Glob Optim 36:161–189
16. Lovasz L, Schrijver A (1990) Cones of matrices and set functions, and 0-1 optimization. SIAM J Optim 1:166–190
17. Sherali HD (1997) Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets. Acta Math Vietnamica 22(1):245–270
18. Sherali HD (1998) Global optimization of nonconvex polynomial programming problems having rational exponents. J Glob Optim 12(3):267–283
19. Sherali HD (2002) Tight Relaxations for nonconvex optimization problems using the reformulation-linearization/convexification technique (RLT). In: Pardalos PM, Romeijn HE (eds) Heuristic Approaches. Handbook of Global Optimization, vol 2. Kluwer, Dordrecht, pp 1–63
20. Sherali HD, Adams WP (1984) A decomposition algorithm for a discrete location-allocation problem. Oper Res 32(4):878–900
21. Sherali HD, Adams WP (1990) A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J Discret Math 3(3):411–430
22. Sherali HD, Adams WP (1994) A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. Discret Appl Math 52:83–106
23. Sherali HD, Adams WP (1994) A reformulation-linearization technique (RLT) for solving discrete and continuous nonconvex programming problems. In: Gupta O (ed) XII-A Mathematics Today, special issue on Recent Advances in Mathematical Programming. pp 61–78
24. Sherali HD, Adams WP (1996) Computational advances using the reformulation-linearization technique (RLT) to solve discrete and continuous nonconvex problems. OPTIMA 49:1–6
25. Sherali HD, Adams WP (1999) A reformulation-linearization technique for solving discrete and continuous nonconvex problems. Kluwer, Dordrecht
26. Sherali HD, Adams WP (2006) Extension of a reformulation-linearization technique (RLT) to semi-infinite and convex programs under mixed 0-1 and general discrete restrictions. Manuscript, Dept. of Industrial Systems Engineering, Virginia Polytechnic Institute, Blacksburg, USA
27. Sherali HD, Adams WP, Driscoll P (1998) Exploiting special structures in constructing a hierarchy of relaxations for 0-1 mixed integer problems. Oper Res 46(3):396–405
28. Sherali HD, Alameddine A (1992) A new reformulation-linearization technique for solving bilinear programming problems. J Glob Optim 2:379–410
29. Sherali HD, Al-Loughani I, Subramanian S (2002) Global optimization procedures for the capacitated Euclidean and l_p distance multifacility location-allocation problems. Oper Res 50(3):433–448
30. Sherali HD, Brown EL (1994) A quadratic partial assignment and packing model and algorithm for the airline gate assignment problem. In: Pardalos PM, Wolkowicz H (eds) Quadratic Assignment and Related Problems. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 16. Providence, pp 343–364
31. Sherali HD, Desai J (2005) A global optimization RLT-based approach for solving the hard clustering problem. J Glob Optim 32:281–306
32. Sherali HD, Desai J (2005) A global optimization RLT-based approach for solving the fuzzy clustering problem. J Glob Optim 33:597–615
33. Sherali HD, Desai J (2005) On solving polynomial, factorable, and black-box optimization problems using the RLT methodology. In: Audet C, Hansen P, Savard G (eds) Essays and Surveys on Global Optimization. Springer, New York, pp 131–163
34. Sherali HD, Driscoll PJ (2002) On tightening the relaxation of Miller-Tucker-Zemlin formulations for asymmetric traveling salesman problems. Oper Res 50(4):656–669
35. Sherali HD, Fraticelli BMP (2002) Enhancing RLT relaxations via a new class of semidefinite cuts. J Glob Optim 22(1–4):233–261
36. Sherali HD, Ganesan V (2003) A pseudo-global optimization approach with application to the design of containerships. J Glob Optim 26(4):335–360
37. Sherali HD, Krishnamurty R, Al-Khayyal FA (1998) Enumeration approach for linear complementarity problems based on a reformulation-linearization technique. J Optim Theory Appl 99(2):481–507
38. Sherali HD, Lee Y (1995) Sequential and simultaneous liftings of minimal cover inequalities for GUB constrained knapsack polytopes. SIAM J Discret Math 8(1):133–153
39. Sherali HD, Lee Y (1996) Tighter representations for set partitioning problems via a reformulation-linearization approach. Discret Appl Math 68:153–167
40. Sherali HD, Lee Y, Adams WP (1995) A Simultaneous lifting strategy for identifying new classes of facets for the Boolean quadric polytope. Oper Res Lett 17(1):19–26
41. Sherali HD, Ramachandran S, Kim S (1994) A localization

and reformulation discrete programming approach for the rectilinear distance location-allocation problem. Discret Appl Math 49(1–3):357–378
42. Sherali HD, Sarin SC, Tsai PF (2006) A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints. Discret Optim 3:20–32
43. Sherali HD, Smith EP (1997) A global optimization approach to a water distribution network problem. J Glob Optim 11:107–132
44. Sherali HD, Subramanian S, Loganathan GV (2001) Effective relaxations and partitioning schemes for solving water distribution network design problems to global optimality. J Glob Optim 19:1–26
45. Sherali HD, Tuncbilek CH (1992) A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. J Glob Optim 2:101–112
46. Sherali HD, Tuncbilek CH (1992) A squared-Euclidean distance location-allocation problem. Nav Res Logist 39:447–469
47. Sherali HD, Tuncbilek CH (1995) A reformulation-convexification approach for solving nonconvex quadratic programming problems. J Glob Optim 7:1–31
48. Sherali HD, Tuncbilek CH (1997) New reformulation-linearization technique based relaxations for univariate and multivariate polynomial programming problems. Oper Res Lett 21(1):1–10
49. Sherali HD, Wang H (2001) Global optimization of nonconvex factorable programming problems. Math Program 89(3):459–478
50. Stubbs RA, Mehrotra S (1999) A branch-and-cut method for 0-1 mixed convex programming. Math Program 86:515–532

Regression by Special Functions: Algorithms and Complexity

VASANT A. UBHAYA
Department Computer Sci. and Operations Research, North Dakota State University, Fargo, USA

MSC2000: 90C26, 41A30, 62J02

Article Outline

Keywords
Isotonic Regression
Quasiconvex and Umbrella Regression
Convex and Concave Regression
See also
References

Keywords

Regression; Best fit; Estimate; Isotonic regression; Quasiconvex regression; Umbrella regression; Convex regression; Concave regression; Least squares distance function; Mean absolute deviation; Maximum absolute deviation; Algorithms; Computational complexity; Pool adjacent violators algorithm; Greatest convex minorant; Greatest quasiconvex minorant; Isotone functions; Monotone function

Consider the following general problem of regression. Given n measurements or observations $f = (f_1, \ldots, f_n) \in \mathbf{R}^n$ with $f_i = \mu_i + \eta_i$, where $\mu = (\mu_1, \ldots, \mu_n)$ is unknown but is in some predefined subset K of $\mathbf{R}^n$, and η_i is the noise generated by sampling, it is required to find a *best fit* or an *estimate* g of μ. The set K is dictated by the underlying system which generates f. A best fit or an optimal solution g is obtained by minimizing a suitable distance function $d(f, g)$ subject to $g \in K$. If $w = (w_1, \ldots, w_n) > 0$ is a given weight function, we use the following distance functions in our analysis:

$$d_\infty(f,g) = \max\{w_i\,|f_i - g_i| :\ 1 \le i \le n\},$$
$$d_1(f,g) = \sum\{w_i\,|f_i - g_i| :\ 1 \le i \le n\},$$
$$d_2(f,g) = \sum\left\{w_i(f_i - g_i)^2 :\ 1 \le i \le n\right\}.$$

Note that $(d_2\ (f, g))^{1/2}$ and not $d_2\ (f, g)$ is a distance function in that it satisfies the well known triangle inequality. Let the primed entries $d_\infty{}'\ (f, g)$, $d_1{}'\ (f, g)$ and $d_2{}'\ (f, g)$ denote the corresponding distances when $w_i = 1$ for all i. As will be seen later that the results for the two sets of distances are different. We call f, g, w etc. 'functions' because they may be considered as such on some underlying set $S = \{x_1, \ldots, x_n\}$, where $f_i = f(x_i)$, $g_i = g(x_i)$, $w_i = w(x_i)$ etc. The best known example of a *regression* problem is the linear regression where K consists of linear functions given by $g_i = a\,x_i + b$ with $x_1 < \cdots < x_n$, and the numbers a and b are unknowns to be determined by minimizing $d_2(f, g)$. In this article we are concerned with the following three cases: K formed by

- istone or monotone functions,
- quasiconvex and umbrella functions, and, finally
- convex and concave functions.

We develop algorithms to compute best fits and obtain their *computational complexity* . In most cases the algorithms are of *polynomial complexity* . Wherever possible we derive explicit expressions for best fits. Applications can be cited to problems in reliability engineering, economics, social sciences, *order restricted statistical inference* etc. [10,13,14,15,18,24]. For example, the failure rate of a system increases with the age of the system (isotonicity), the human mortality rate (or the number of auto accidents) first decreases and then increases with age (quasiconvexity), the efficiency of an organization first increases and then decreases with its size (quasiconcavity). Similarly, in economics assumptions of convexity/concavity are made about functions representing utility, marginal utility, productivity, supply, demand etc.

We briefly discuss the significance of different distance functions used in regression. In the classical approach to optimization and regression, the *least squares distance function* $d_2(f, g)$ was extensively used. Its differentiability properties led to certain simplification of mathematical analysis. For some time now, both $d_1(f, g)$ (*mean absolute deviation*) and $d_\infty(f, g)$ (*maximum absolute deviation*) distance functions are being used. For example, see MINMAD and MINMADAX regression in [2] and the least absolute value (LAV) or L_1-norm estimation and L_∞-norm estimation in [11]. See also [6]. If we write $d_p(f, g) = \sum \{w_i |f_i - g_i|^p : 1 \le i \le n\}$, $1 \le p < \infty$, then $(d_p(f, g))^{1/p}$ is a distance function, and $\|f\|_p = d_p(f, 0)^{1/p}$, $1 \le p < \infty$, and $\|f\|_\infty = d_\infty(f, 0)$ are, respectively, the L_p and L_∞ norms on $\mathbf{R}^n$. Both d_1 and d_∞ distances have the strong advantage that their form allows transformation of the regression problem to a linear program which facilitates computation of its solution [2,19]. The nature of the problem essentially determines the choice of the distance function. Then the algorithms developed for the solution, their computational complexities and the properties of the best fits obtained, all in turn, depend on the distance function used.

Isotonic Regression

A function $g = (g_1, \ldots, g_n) \in \mathbf{R}^n$ is called *isotonic* or *monotonic* (*nondecreasing*) if $g_1 \le \cdots \le g_n$. We let K be the set of all isotonic functions and consider the *isotonic regression problem* of minimizing $d_2(f, g)$ subject to $g \in K$. Since $d_2(f, g)$ is strictly convex in g and K is a closed convex cone, the solution of this problem is unique. (K is a cone if $\lambda h \in K$ whenever $h \in K$ and $\lambda \ge 0$.)

We describe the *pool adjacent violators algorithm* (PAV) for computing the solution. See [24] and other references given there. The form presented below appears in [5]. Let $N = \{1, \ldots, n\}$. A *partition* J of N is a decomposition of N into disjoint sets of consecutive integers such that their union is N. Each member of J is called a *block* and is denoted by B. We let $\Delta_B(c) = \sum \{w_i (f_i - c)^2 : i \in B\}$. We let $g(J)$ be any n-vector whose ith coordinate $g_i(J)$, $i \in N$, is given by $g_i(J) = \mu_B$, where B is the unique block of J containing i and μ_B is the minimizer of Δ_B. Clearly, $g_i(J)$ has identical values for all i in a block. Also μ_B equals the 'block average' $\sum \{w_i f_i : i \in B\} / \sum \{w_i : i \in B\}$. The PAV algorithm starts with the finest partition whose blocks are single integers in N and an initial infeasible solution g violating the constraint $g_1 \le \cdots \le g_n$. It successively merges blocks to reduce infeasibility and obtains a new coarser partition J and an infeasible solution $g(J)$. It terminates when $g(J)$ becomes feasible obtaining the optimal solution and the final optimal partition. Let $B = \{p, \ldots, q\}$, $1 \le p \le q \le n$, denote a block of a partition J during any iteration of the algorithm. The predecessor and successor blocks of B, denoted respectively by $B-$ and $B+$, of the same partition J, are defined as follows: If $p > 1$, then $B-$ is the block containing $p - 1$, otherwise $B- = \emptyset$. Similarly, if $q < n$, then $B+$ is the block containing $q + 1$, otherwise $B+ = \emptyset$.

It can be shown that [4,21] the algorithm is of linear time complexity ($O(n)$) in the usual unit cost RAM model [1]. We now describe an interesting implementing of the algorithm [21]. Let $W_i = \sum \{w_j : 1 \le j \le i\}$ and $F_i = \sum \{w_j f_j : 1 \le j \le i\}$ with $W_0 = F_0 = 0$. Clearly, $W_{i-1} < W_i$. By plotting the $n + 1$ points $P_i = (W_i, F_i)$, $0 \le i \le n$, in the plane we obtain the *cumulative sum diagram* (csd). Note that the slope of the line segment joining P_{i-1} and P_i is f_i. Let $Q_i = (W_i, G_i)$, be the *greatest convex minorant* (gcm) or the *lower convex hull* of the csd. This is the largest convex function which does not exceed the csd at any point. It is easy to see that $P_0 = Q_0$ and $P_n = Q_n$. Also since the gcm is convex, the slopes $(G_i - G_{i-1})/(W_i - W_{i-1})$ are nondecreasing. It has been shown that $g_i = (G_i - G_{i-1})/(W_i - W_{i-1})$, $1 \le i \le n$, give an optimal solution to the *isotonic regression* problem. Graham 's scan [12] can be easily modi-

Initialization: Set $J = \{\{i\} : i \in N\}$;
compute the minimizer μ_B of Δ_B, $B \in J$
(if $B = \{i\}$, then $\mu_B = f_i$).
Set $B = \{1\}$, $B+ = \{2\}$, $B- = \emptyset$;
WHILE $B+ \neq \emptyset$
IF $\mu_B > \mu_{B+}$ THEN
merge B and $B+$
(i.e. set $J = J\backslash\{B, B+\} \cup \{B \cup B+\}$ and
$B = B \cup B+$);
compute the new μ_B;
set $B+$ appropriately
($B-$ remains unchanged);
WHILE $B- \neq \emptyset$ and $\mu_{B-} > \mu_B$
merge B and $B-$
(i.e. set $J = J\backslash\{B, B-\} \cup \{B \cup B-\}$
and $B = B \cup B-$);
compute the new μ_B;
set $B-$ appropriately
($B+$ remains unchanged);
END WHILE
ELSE set $B = B+$;
set $B-$ and $B+$ appropriately;
END IF
END WHILE
$x_i(J) = \mu_B$, $i \in B \in J$, is an optimal solution

Regression by Special Functions: Algorithms and Complexity, Algorithm 1
Algorithm PAV (pool adjacent violators)

fied to obtain the gcm in $O(n)$ time. See e. g. [28]. Thus the overall algorithm is again $O(n)$. The treatment and the results for d_2' are identical to those of d_2.

Now consider the problem of *isotonic medium regression* which is the problem of minimizing $d_1'(f, g)$ subject to $g \in K$ [7,23]. The PAV algorithm can be applied to this problem with $\Delta_B(c) = \sum \{|f_i - c| : i \in B\}$. A minimizer μ_B of Δ_B can be easily shown to be a median of the set $\{f_i : i \in B\}$. We choose μ_B to be the central medium of the set $\{f_i : i \in B\}$ defined as follows. For an ordered (ascending) $\{c_1, \ldots, c_r\}$, it is $c_{(r+1)/2}$ if r is odd, and

$$\frac{c_{\frac{r}{2}} + c_{\frac{r}{2}+1}}{2}$$

if r is even. (If r is even, then both the lower and upper mediums, $c_{r/2}$ and $c_{r/2+1}$, or any point in between the two, minimize $\Delta_B(c)$. The nonuniqueness of the medium is addressed later.) The PAV algorithm with this choice of the central medium is of complexity $O(n^2)$, since the median of an unordered set of cardinality r can be computed in linear time $O(r)$ ([1, Algorithm 3.6]). Now consider minimizing $d_1(f, g)$. In this case, let $\Delta_B(c) = \sum \{w_i |f_i - c| : i \in B\}$, whose minimizer μ_B is a weighted median of the set $\{f_i : i \in B\}$. Again it is not unique. We use the unique weighted central medium of an ordered (ascending) set $\{c_1, \ldots, c_r\}$ with weight ω_i for c_i defined as follows. Let $\lambda = \sum \{\omega_i : 1 \leq i \leq n\}$.The central median is c_q if $\sum \{\omega_i : 1 \leq i \leq q-1\} < \lambda/2 < \sum \{\omega_i; q \leq i \leq r\}$ for some q. It is $(c_{q-1} + c_q)/2$ if $\sum \{\omega_i : 1 \leq i \leq q-1\} = \lambda/2$ (As before both the lower and upper mediums, c_{q-1} and c_q, or any point in between the two, minimize $\Delta_B(c)$). Again it can be computed in linear time by extending ([1, Algorithm 3.6]) to the weighted case. It will be seen that the ratio $\max\{w_i\} / \min\{w_i\}$ plays an important role in analysis. The PAV algorithm for this case is of complexity $O(n^2)$. Now consider the nonuniqueness of the medium. We may use the lower or upper medium for μ_B in the PAV algorithm since both minimize $\Delta_B(c)$. When we consistently use the lower (resp. upper) medium in the algorithm for d_1 or d_1', we get the minimal (resp. maximal) optimal solution to the problem. This has been established in [5] for a general case of minimizing a separable convex function subject to the monotonicity constraint. This problem includes, as special cases, the isotonic regression with distances d_1 and d_2, and other situations such as in [25,26]. Now consider the problem of *isotone optimization*, i. e., minimizing d_∞ and d_∞' [27]. Define $t_{ij} = w_i w_j/(w_i + w_j)$ and $\theta = \max\{t_{ij}(f_i - f_j) : 1 \leq i \leq j \leq n\}$. Define two isotonic functions $\underline{g}$ and $\overline{g}$ by $\underline{g}_i = \max\left\{f_j - \frac{\theta}{w_j} : 1 \leq j \leq i\right\}$, $\overline{g}_i = \min\left\{f_j + \frac{\theta}{w_j} : i \leq j \leq n\right\}$, $1 \leq i \leq n$. Then both $\underline{g}$ and $\overline{g}$ are optimal solutions to isotone optimization with distance function d_∞ and $\theta = d_\infty(f, \underline{g}) = d_\infty(f, \overline{g})$. Furthermore, a monotonic g is an optimal solution if and only if $\underline{g} \leq g \leq \overline{g}$. The computation of the solution clearly takes $O(n^2)$ time. The above results are also valid for d_∞', but in this case the results can be simplified and an $O(n)$ algorithm for the computation can be obtained (28, [Sect. 6]). Define $\underline{k}_1 = f_1$, $\underline{k}_i = \max\{\underline{k}_{i-1}, f_i\}$, $i = 2, \ldots, n$, and $\overline{k}_n = f_n$, $\overline{k}_i = \min\{\overline{k}_{i+1}, f_i\}$, $i = n-1, n-2, \ldots, 1$. Let $\theta = \frac{d'_\infty(f,\underline{k})}{2} = \frac{d'_\infty(f,\overline{k})}{2}$. Then

$\underline{g}_i = \underline{k}_i - \theta$ and $\overline{g}_i = \overline{k}_i + \theta$. Clearly, the algorithm is $O(n)$. For isotonic regression problems with integer constraints see [10] and [18].

Quasiconvex and Umbrella Regression

A function $g = (g_1, \ldots, g_n) \in \mathbf{R}^n$ is called *quasiconvex* if $g_i \leq \max\{g_p, g_q\}$ holds for all i with $p \leq i \leq q$ for all $1 \leq p \leq q \leq n$ [22]. A g is called *quasiconcave* or *umbrella* if $-g$ is quasiconvex. It can be shown that $g \in \mathbf{R}^n$ is quasiconvex if and only if there exists an index r, $1 \leq r \leq n$, such that $g_1 \geq \cdots \geq g_r \leq g_{r+1} \leq \cdots \leq g_n$. (Thus $g_r = \min\{g_i: 1 \leq i \leq n\}$.) [8,29]. If $r = 1$ (resp. n), then g is nondecreasing (resp. nonincreasing). Similar results may be stated for a quasiconcave function. In what follows we discuss regression by quasiconvex functions, the results for quasiconcave functions are symmetric. The quasiconvex problem can be transformed into $2n$ isotonic subproblems by the observation made above. Let K denote the set of all quasiconvex g, and let K_r denote all $g = (g_1, \ldots, g_n)$ with $g_1 \geq, \ldots, \geq g_r$ and $g_{r+1} \leq \cdots \leq g_n$, where $1 \leq r \leq n$. Note that K_r is the set of all quasiconvex g such that g_r or g_{r+1} equals $\min\{g_i: 1 \leq i \leq n\}$. Clearly, $K = \cup\{K_r: 1 \leq r \leq n\}$. It is easy to see that K is a closed cone which is not convex, although K_r, for each fixed r, is a closed convex cone. Hence optimal solutions obtained in this section are not necessarily unique. To solve the quasiconvex Regression problem of minimizing $d_2(f, g)$ subject to $g \in K$, we consider the following two subproblems for each r:

- $\mathrm{P1}_r$: Find g_i, $1 \leq i \leq r$, and $\Delta\ 1_r$ so that $\Delta\ 1_r = \min \sum\{w_i (f_i - g_i)^2: 1 \leq i \leq r\}$ subject to $g_1 \geq \cdots \geq g_r$; and
- $\mathrm{P2}_r$: Find g_i, $r+1 \leq i \leq n$, and $\Delta\ 2_r$ so that $\Delta\ 2_r = \min \sum\{w_i (f_i - g_i)^2: r+1 \leq i \leq n\}$ subject to $g_{r+1} \leq \cdots \leq g_n$.

We then minimize $\Delta\ 1_r + \Delta\ 2_r$ subject to $1 \leq r \leq n$. Note that both $\mathrm{P1}_r$ and $\mathrm{P2}_r$ are isotonic regression problems. They can be solved in $O(r)$ and $O(n-r)$ time respectively, using the PAV algorithm of the previous section. Thus, the quasiconvex regression problem can be solved in $O(n^2)$ time. We have shown in [31] that the complexity can be improved to $O(n)$ by using special mathematical results and suitable data structures. Our algorithm uses one forward and one backward pass on indexes $1, \ldots, n$ to obtain the unique optimal solutions of both $\mathrm{P1}_r$ and $\mathrm{P2}_r$ and the values of $\Delta_r = \Delta\ 1_r$ and $\Delta\ 2_r$ for all r. Although we use Graham 's scan and the gcm discussed in the previous section to compute the solutions of the isotonic regression subproblems $\mathrm{P1}_r$ and $\mathrm{P2}_r$, alternative schemes without using the gcm can be easily devised. Another algorithm of unknown complexity appears in a later article [9]. Now consider the *quasiconvex medium regression* problem of minimizing $d_1(f, g)$ subject to $g \in K$. We may consider problems $\mathrm{P1}_r$ and $\mathrm{P2}_r$ as above where $\sum w_i (f_i - g_i)^2$ is replaced by $\sum w_i |f_i - g_i|$. These isotonic medium regression problems can be solved in quadratic time giving an overall complexity of $O(n^3)$. Whether this complexity can be improved or not is an open issue at this time.

The strategy of transforming the quasiconvex problem into $2n$ isotonic subproblems can also be used for d_∞. Since each isotonic subproblem can be solved in $O(n^2)$ time as for isotonic regression (see above), it may seem that the complexity of the quasiconvex problem is $O(n^3)$. A little reflection will show that the computations can be organized in $O(n^2)$ time. Indeed, using the notation for isotonic regression above, the constants $t_{ij}(f_i - f_j)$, $1 \leq i \leq j \leq n$, can be computed in $O(n^2)$ time. Then the θ 's needed for the subproblems can be computed recursively in $O(n^2)$ time. The rest of the computations are $O(n)$. If $d_\infty{}'$ is used, then the complexity can be improved to $O(n)$. Let $f_m = \min\{f_i: 1 \leq i \leq n\}$, m is not unique. Define $\underline{k}_m = f_m$, $\underline{k}_i = \max\{\underline{k}_{i-1}, f_i\}$, $i = m+1, m+2, \ldots, n$, $\underline{k}_i = \max\{\underline{k}_{i+1}, f_i\}$, $i = m-1, m-2, \ldots, 1$, $\theta = \frac{d'_\infty(f,k)}{2}$ and $\underline{g}_i = \underline{k}_i - \theta$, $1 \leq i \leq n$. Then $\underline{g}$ is an optimal solution to the problem. Also, define $\overline{k}_1 = f_1$, $\overline{k}_i = \min\{\overline{k}_{i-1}, f_i\}$, $i = 2, \ldots, m-1$, $\overline{k}_n = f_n$, $\overline{k}_i = \min\{\overline{k}_{i+1}, f_i\}$, $i = n-1, n-2, \ldots, m+1$, $\overline{k}_m = f_m$, $\theta = \frac{d'_\infty(f,\overline{k})}{2}$ and $\overline{g}_i = \overline{k}_i + \theta$, $1 \leq i \leq n$. Then $\overline{k}$ is the *greatest quasiconvex minorant* of f, i. e., the largest quasiconvex function such that $\overline{k}_i \leq f_i$ for all i, and $\overline{g}$ is the maximal optimal solution to the problem. Clearly, the computations are $O(n)$ [29]. This problem on an interval is considered in [30].

Convex and Concave Regression

We consider functions f, g etc. on a set $S = \{x_1, \ldots, x_n\}$ so that $f_i = f(x_i)$, $g_i = g(x_i)$ etc. Then g is convex if $(g_i - g_{i-1})/\delta_{i-1} \leq (g_{i+1} - g_i)/\delta_i$, $2 \leq i \leq n-1$, where $\delta_i = x_{i+1} - x_i$, $1 \leq i \leq n-1$. These constraints are linear. If the points x_i are equally spaced, i. e., all δ_i have identical values, then the convexity condition be-

comes $g_{i+1} - 2\,g_i + g_{i-1} \geq 0$, $2 \leq i \leq n-1$. Let K be the set of all convex functions. Then K is a closed convex cone. We first consider the *convex regression problem* of minimizing $d_2\,(f, g)(d_2'\,(f, g))$ subject to $g \in K$. Clearly, this is a quadratic programming problem where $d_2\,(d_2')$ is a strictly convex function of g. Its solution is unique. The problem may be formulated as a linear complementarity problem and solved [3,20]. Special methods as in [16,17] and other references given there, may be applied. Some earlier work appears in [13,14]. Results on complexity analysis of algorithms for these problems, in general, are lacking. The problem with d_1, d_1' or d_∞ distance function can be formulated as a linear programming problem [2,19] and solved. Now consider d_∞'. Let $\overline{k}$ be the greatest convex minorant of f, i. e., the largest convex function such that $\overline{k}_i \leq f_i$ for all i. It may be easily computed in $O(n)$ time using Graham 's scan as for isotonic regression. Let $\theta = \frac{d'_\infty(f,\overline{k})}{2}$ and $\overline{g}_i = \overline{k}_i + \theta$, $1 \leq i \leq n$. Then $\overline{g}$ is a solution to the problem computed in $O(n)$ time. Indeed, it is the maximal optimal solution to the problem [28,32].

Since regression problems are indeed approximation problems, some of the results of [33] are applicable to our problems. In particular, note the significance of the dual cone of the cone K of isotone functions as stated in the last paragraph there.

See also

► Isotonic Regression Problems

References

1. Aho AV, Hopcroft JE, Ullman JD (1974) Design and analysis of computer algorithms. Addison-Wesley, Reading
2. Arthanari TS, Dodge Y (1981) Mathematical programming in statistics. Wiley, New York
3. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: Theory and algorithms, 2nd edn. Wiley, New York
4. Best MJ, Chakravarti N (1990) Active set algorithms for isotonic regression; a unifying framework. Math Program 47:425–439
5. Best MJ, Chakravarti N, Ubhaya VA (2000) Minimizing separable convex functions subject to simple chain constraints. SIAM J Optim 10:658–672
6. Birkes D, Dodge Y (1993) Alternative methods of regression. Wiley, New York
7. Chakravarti N (1989) Isotonic median regression; a linear programming approach. Math Oper Res 14:303–308
8. Deak E (1962) Uber konvexe und interne Functionen, sowie eine gemeinsame Verallgemeinerung von beide. Ann Univ Sci Budapest R Eotvos Sect Math 5:109–154
9. Geng Z, Shi N-Z (1990) Isotonic regression for umbrella ordering. Applied Statist 39:397–424
10. Goldstein AJ, Kruskal JB (1976) Least square fitting for monotonic functions having integer values. J Amer Statist Assoc 71:370–373
11. Gonin R, Money AH (1989) Nonlinear Lp-norm estimation. M. Dekker, New York
12. Graham RL (1972) An efficient algorithm for determining the convex hull of a finite planar set. Inform Process Lett 1:132–133
13. Hanson DL, Pledger D (1976) Consistency in concave regression. Ann Statist 4:1038–1050
14. Hildreth C (1954) Point estimates of ordinates of concave functions. J Amer Statist Assoc 49:519–619
15. Intriligator MD (1971) Mathematical optimization and economic theory. Prentice-Hall, Englewood Cliffs
16. Li W, Swetits J (1993) A Newton method for convex regression, data smoothing, and quadratic programming with bounded constraints. SIAM J Optim 3:466–488
17. Li W, Swetits J (1997) A new algorithm for solving strictly convex quadratic programs. SIAM J Optim 7:595–619
18. Liu M-H, Ubhaya VA (1997) Integer isotone optimization. SIAM J Optim 7:1152–1159
19. Murty KG (1983) Linear programming. Wiley, New York
20. Murty KG (1988) Linear complementarity, linear and nonlinear programming. Heldermann, Berlin
21. Preparata FP, Shamos MI (1985) Computational geometry. Springer, New York
22. Roberts AW, Varberg DE (1973) Convex functions. Acad. Press, New York
23. Robertson T, Wright FT (1980) Algorithms in order restricted statistical inference and the Cauchy mean value property. Ann Statist 8:645–651
24. Robertson T, Wright FT, Dykstra RL (1988) Order restricted statistical inference. Wiley, New York
25. Schwarz LB, Schrage L (1975) Optimal and system-myopic policies for multi-echelon production /inventory assembly systems. Managem Sci 21:1285–1294
26. Stromberg V (1991) An algorithm for isotonic regression with arbitrary convex distance function. Comput Statist Data Anal 11:205–219
27. Ubhaya VA (1974) Isotone optimization I-II. J Approx Theory 12:146–159; 315–331
28. Ubhaya VA (1979) An O(n) algorithm for discrete n-point convex approximation with applications to continuous case. J Math Anal Appl 72:338–354
29. Ubhaya VA (1984) O(n) algorithms for discrete n-point approximation by quasi-convex functions. Comput Math Appl 10:365–368
30. Ubhaya VA (1986) Quasi-convex optimization. J Math Anal Appl 116:439–449

31. Ubhaya VA (1987) An O(n) algorithm for least squares quasi-convex approximation. Comput Math Appl 14:583–590
32. Ubhaya VA (1990) Lipschitzian selections in best approximation by continuous functions. J Approx Theory 61:40–52
33. Ubhaya VA (2008) Best approximation by bounded or continuous functions. In: Floudas CA (ed) (2008) Encyclopedia of Optimization, 2nd edn. Springer, Berlin

Relaxation in Projection Methods

UBALDO M. GARCÍA-PALOMARES
University Simón Bolívar, Caracas, Venezuela

MSC2000: 90C30

Article Outline

Keywords
Projections
Relaxation
Fejér Property
Convergence of PPA
Discussion
See also
References

Keywords

Projection method; Relaxation

Projections

Projection methods have become a useful technique for solving the *convex feasibility problem*

$$\text{(CFP)} \quad \text{Find } x \in C,$$

where C is a closed, convex nonempty set defined in a Hilbert space H endowed with an inner product $\langle \cdot, \cdot \rangle : H^2 \to \mathbf{R}$, and the induced norm $\|x\|^2 = \langle x, x \rangle$.

Projection methods are iterative. Given $x_i \notin C$, the standard iterative scheme is

$$x_{i+1} = x_i + \omega_i (P_{S_i}(x_i) - x_i), \tag{1}$$

where $S_i \supseteq C$ is a closed convex set, $P_{S_i}(x_i)$ is the projection of x_i on the set S_i, and $\omega_i \in \mathbf{R}$ is the relaxation parameter. If $\omega_i < 1$ we call it an *underprojection*, and if $\omega_i > 1$ we call it an *overprojection*. In general, $0 < \eta \leq \omega_i \leq 2 - \eta$ is required to ensure convergence.

We attempt to present an overview of projection techniques. This section will cover the choice of the supersets $\{S_i\}_1^\infty$. Next section will study the relaxation parameter and a third section offers additional material on the subject, including pertinent references. We use a standard notation with minor peculiarities that should cause no difficulty to the reader.

Let us start by recalling that $P_S(x)$, the projection of x on a closed convex nonempty set S, is the point in S closest to x, namely,

$$P_S(x) \stackrel{\text{def}}{=} \operatorname*{argmin}_{z \in S} \|z - x\|^2 . \tag{2}$$

By the minimum principle, (2) holds if and only if $P_S(x) \in S$ and

$$[z \in S] \;\Rightarrow\; [\langle z - P_S(x), x - P_S(x) \rangle \leq 0]. \tag{3}$$

Since

$$\begin{aligned} &\langle P_S(x) - x, x - z \rangle \\ &= \langle P_S(x) - x, x - P_S(x) + P_S(x) - z \rangle , \end{aligned}$$

we immediately obtain that (3) is equivalent to

$$[z \in S] \;\Rightarrow\; \langle P_S(x) - x, x - z \rangle \leq - \|P_S(x) - x\|^2 . \tag{4}$$

In general, the projection problem (2) is difficult, but sometimes it is a straightforward computation. If S is a halfspace given by $S \stackrel{\text{def}}{=} \{z \in H: \; \langle a, z \rangle \leq \beta\}$, where $a \in H$, $\beta \in \mathbf{R}$, the projection of x onto S is given by:

$$P_S(x) := \begin{cases} x & x \in S, \\ x - \dfrac{\langle a, x \rangle - \beta}{\langle a, a \rangle} a & \text{otherwise.} \end{cases} \tag{5}$$

We can easily verify that (3) holds.

In most techniques based on (1) S_i is a halfspace, a hyperplane, or an appropriate set that renders (2) an easy problem. For instance, if C is the solution set of a linear system of equations in the Euclidean space $\mathbf{R}^n$, i. e., $C = \{z \in \mathbf{R}^n : Az = b\}$, block action methods split the system in p subsystems, i. e.,

$$[Az = b] := \begin{pmatrix} A_1 z = b_1 \\ \vdots \\ A_p z = b_p \end{pmatrix} .$$

Let $C_k = \{z \in \mathbf{R}^n : A_k z = b_k\}$, for $k = 1, \ldots, p$.
If A_k is full rank, and A_k' its transpose, then

$$P_{C_k}(x_i) = x_i - A_k'(A_k A_k')^{-1}(A_k x_i - b_k).$$

Observe that (3) and (4) hold as equalities. A standard method chooses

$$S_i = C_k, \quad k = (i \bmod p) + 1$$

Convergence will be apparent in the next section.

An important CFP is to find $x \in C$, where C is defined by convex inequalities, i. e.,

$$C = \left\{z \in H:\ f_j(z) \le b^j,\ j = 1, \ldots, m\right\}, \tag{6}$$

where $f_j(\cdot)$: $H \to \mathbf{R}$ are convex subdifferentiable functions and b^j are scalars, for $j = 1, \ldots, m$. Convexity of C can be shown by well known properties of convex functions that we state for completeness

Lemma 1 *Let $f(\cdot)$: $H \to \boldsymbol{R}$. Let $v_1, \ldots, v_p$ be vectors in H, and $\mu_1, \ldots, \mu_p$ be nonnegative scalars such that $\sum_{k=1}^p \mu_k = 1$. Then $f(\cdot)$ is convex if and only if*

$$f\left(\sum_{k=1}^p \mu_k v_k\right) \le \sum_{k=1}^p \mu_k f(v_k). \tag{7}$$

Lemma 2 *$f_1(\cdot), \ldots, f_m(\cdot)$: $H \to R$ are convex functions, and $v_1, \ldots, v_m$ are nonnegative scalars. Then $\sum_{j=1}^m v_j f_j(\cdot)$ is also convex.*

Lemma 3 *If $f(\cdot)$: $H \to \boldsymbol{R}$ is convex and subdifferentiable, with subgradient $\partial f(\cdot)$: $H \to H$, then*

$$[y, x \in H] \ \Rightarrow\ f(y) \ge f(x) + \langle \partial f(x), y - x\rangle$$

and (obviously)

$$[\beta \ge f(y)] \ \Rightarrow\ [\beta \ge f(x) + \langle \partial f(x), y - x\rangle].$$

We turn our attention to the choice of $\{S_i\}_1^\infty$ for solving $z \in C$ defined by (6). Given x_i define the halfspace

$$C_j(x_i) = \left\{z \in H:\ f_j(x_i) + \left\langle \partial f_j(x_i), z - x_i\right\rangle \le b^j\right\}. \tag{8}$$

As $[z \in C] \Rightarrow [b^j \ge f_j(z)]$ we assert by Lemma 3 that $C_j(x_i) \supseteq C$. In the next section it will be evident that convergence of (1) is ensured if $j = (i \bmod m) + 1$ and $S_i = C_j(x_i)$.

Let $f^+(\cdot) \stackrel{\text{def}}{=} \max(0, f_j(\cdot))$. Another possible choice for S_i is

$$S_i = \left\{ z \in H:\ \begin{array}{l} \sum_{j=1}^m f_j^+(x_i)(f_j(x_i) \\ + \left\langle \partial f_j(x_i), z - x_i\right\rangle \le b^j) \end{array} \right\}. \tag{9}$$

By Lemma 3, S_i contains the convex set C_i given next. It is obvious that $C_i \supseteq C$:

$$C_i = \left\{ z \in H:\ \sum_{j=1}^m f_j^+(x_i)(f_j(z) \le b^j) \right\}.$$

Relaxation

Both $\{S_i\}_1^\infty$, and $\{\omega_i\}_1^\infty$ influence the quality of convergence of projection methods significantly. This section is concerned with the relaxation parameter. We would like to choose $\{\omega_i\}_1^\infty$, such that $z \in C$ is obtained with the fewest number of iterations possible. Using the iterative scheme (1) we have for all $z \in S_i$ that

$$\begin{aligned} &\|x_{i+1} - z\|^2 = \|x_i - z + \omega_i(P_{S_i}(x_i) - x_i)\|^2 \\ &= \|x_i - z\|^2 + 2\omega_i \langle x_i - z, P_{S_i}(x_i) - x_i\rangle \\ &+ \omega_i^2 \|P_{S_i}(x_i) - x_i\|^2. \end{aligned}$$

Since $S_i \supseteq C$, we conclude by (4) that

$$\begin{aligned} &[z \in C] \ \Rightarrow\ \|x_{i+1} - z\|^2 \\ &\le \|x_i - z\|^2 - \omega_i(2 - \omega_i)\|P_{S_i}(x_i) - x_i\|^2. \end{aligned} \tag{10}$$

We may reasonably expect that $\omega_i = 1$ is the 'best' choice because $\omega(2 - \omega)$ achieves its maximum value at $\omega = 1$ and therefore x_{i+1} is the 'closest' to the set C; however, $\omega_i = 1$ for all i can be a very poor choice. Let us illustrate this fact with the following example in the Euclidean space $\mathbf{R}^2$, $z = (z^1, z^2)$:

Example 4 Let

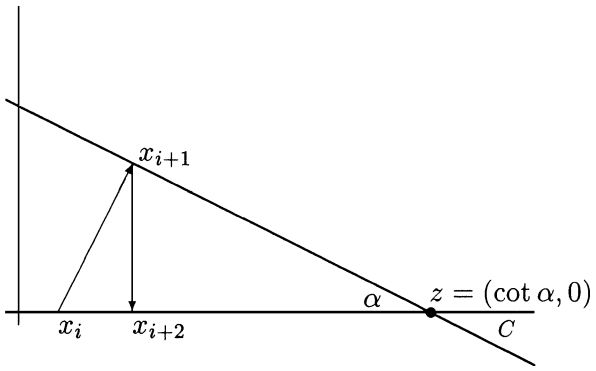
$$C = \{z^2 \le 0,\ (\sin\alpha)z^1 + (\cos\alpha)z^2 \ge \cos\alpha\},$$

$$S_i = \begin{cases} (\sin\alpha)z^1 + (\cos\alpha)z^2 \ge \cos\alpha & i \text{ even}, \\ z^2 \le 0 & i \text{ odd}. \end{cases}$$

Now the problem is: Starting at $x = 0$, estimate the number of iterations needed to generate $z \in C$.

- (Scheme 1) Assume exact projections, i. e. $\forall i$: $x_{i+1} = P_{S_i}(x_i)$.
  We deduce from Fig. 1 that for all i:

$$\|x_{i+1} - z\| = \|x_i - z\| \cos\alpha. \tag{11}$$



Relaxation in Projection Methods, Figure 1
Successive projections

Consequently:

$$\|x_i - z\| = \|x_0 - z\| \cos^i \alpha, \quad \text{for all } i, \tag{12}$$

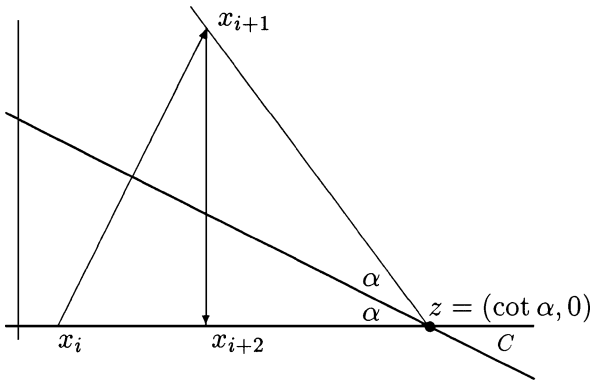
which means that the smaller α, the bigger the number of iterations. Indeed, if $\alpha = 10^o$ and x_0 is the origin we need more than 700 iterations ($i > 700$) to ensure that $\| x_i - z \| \leq 10^{-4}$.

- (Scheme 2) Assume

$$x_{i+1} = \begin{cases} x_i + 2(P_{S_i}(x_i) - x_i) & i \text{ even}, \\ P_{S_i}(x_i) & i \text{ odd}. \end{cases}$$

We deduce from Fig. 2 for i even that

$$\begin{aligned} \|x_{i+1} - z\| &= \|x_i - z\|, \\ \|x_{i+2} - z\| &= \|x_{i+1} - z\| \cos(2\alpha) \\ &= \|x_i - z\| \cos(2\alpha). \end{aligned}$$



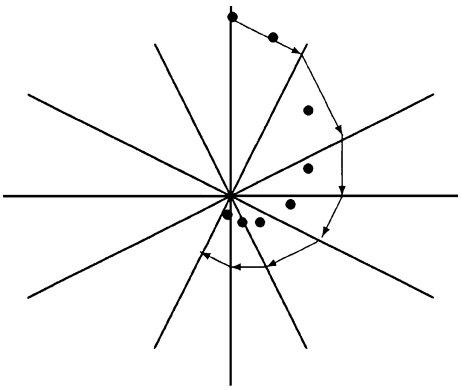
Relaxation in Projection Methods, Figure 2
Overprojections

Consequently:

$$\|x_{2i} - z\| = \|x_0 - z\| \cos^i(2\alpha). \tag{13}$$

If $\alpha = 10^o$ and x_0 is the origin then $\| x_i - z \| \leq 10^{-4}$ for all $i \geq 350$, a significant reduction in the number of iterations.

A small angle α means a small rate of convergence, and overprojection ($\omega > 1$) has the desirable practical effect of opening up this angle. Convergence of projection methods may significantly improve if an optimum value of the relaxation parameter is chosen; however this choice is often difficult. We argue that, in general, it is beneficial to overproject ($\omega > 1$), despite some rare examples require underprojection ($\omega < 1$) to achieve a better (linear) rate of convergence. Figure 3 illustrates this latter possibility. The convex set C is the unique point intersection of a bundle of straight lines passing through it.



Relaxation in Projection Methods, Figure 3
Under projection •, …, •; Projection →

Fejér Property

Most convergence analysis of projection methods are derived from the so called Fejér property, namely, the monotonicity of the sequence $\{\| x_i - z \|\}_1^\infty$. Note from (10) that

$$\begin{pmatrix} z \in C \\ 0 \leq \omega_i \leq 2 \end{pmatrix} \Rightarrow [\|x_{i+1} - z\| \leq \|x_i - z\|]. \tag{14}$$

Hence, a sufficient condition to preserve the Fejér property is that the value of the relaxation parameter ω_i belongs to the interval [0, 2]. To ensure convergence a stronger condition is usually imposed. $\{\omega_i\}_1^\infty$

must stay uniformly bounded away from 0, and 2. We will show below that this condition is not necessary. We will obtain a relaxation parameter $\omega > 2$, preserving the Fejér property, and not impairing convergence. Even more, numerical results reveal that the quality of convergence improves significantly. In order to present the key facts without obscuring our presentation, we focus our attention on parallel algorithms for solving the *convex intersection problem* (CIP), namely: Find $z \in C$, where C is a nonempty intersection of a finite collection of p convex sets, that is,

$$C = C_1 \cap \cdots \cap C_p.$$

We assume (again for the sake of clarity) that the projection problem

$$y_{ik} = \underset{x \in C_k}{\operatorname{argmin}} \|x - x_i\|^2 \tag{15}$$

is easily solved.

Problem	CIP
Data	x_0, an estimate to $z \in C$,
	η: $0 < \eta \leq 1$,
	δ: $0 < \delta \leq 1/p$,
	$i = 0$, the iteration number
REPEAT	
	FOR $k = 1, \ldots, p$ DO
	IN parallel
	Let y_{ik} be defined by (15)
	$d_{ik} = y_{ik} - x_i$
	END parallel
	Choose $\omega_{ik} : \eta \leq \omega_{ik} \leq (2 - \eta)$,
	Choose $\mu_{ik} \geq \delta$: $\sum_{k=1}^{p} \mu_{ik} = 1$,
	END FOR
	$x_{i+1} = x_i + \sum_{k=1}^{p} \mu_{ik}\omega_{ik}d_{ik}$, $i = i + 1$
UNTIL	Convergence

Parallel projection algorithm (PPA)

Above we have presented a parallel projection algorithm (PPA); we will now sketch its proof of convergence under suitable standard conditions. Then we introduce a λ factor to modify the relaxation parameters and argue that the solution of a quadratic programming problem will lead to *optimal relaxation*. Finally we present the accelerated projection algorithm (APA) that subsumes our work.

Convergence of PPA

By using (10), (7) on the convex function $f(\cdot) = \|\cdot\|^2$, and the definitions of ω_{ik} and μ_{ik} we obtain for $z \in C$ that

$$\begin{aligned}
\|x_{i+1} - z\|^2 &= \left\| x_i + \sum_{k=1}^{p} \mu_{ik}\omega_{ik}d_{ik} - z \right\|^2 \\
&= \left\| \sum_{k=1}^{p} \mu_{ik}(x_i + \omega_{ik}d_{ik} - z) \right\|^2 \\
&\leq \sum_{k=1}^{p} \mu_{ik} \|x_i + \omega_{ik}d_{ik} - z\|^2 \\
&\leq \sum_{k=1}^{p} \mu_{ik} \left[\|x_i - z\|^2 - \omega_{ik}(2 - \omega_{ik}) \|d_{ik}\|^2 \right] \\
&\leq \|x_i - z\|^2 - \eta^2 \sum_{k=1}^{p} \mu_{ik} \|d_{ik}\|^2 \\
&\leq \|x_i - z\|^2 - p\delta\eta^2 \sum_{k=1}^{p} \|d_{ik}\|^2 .
\end{aligned}$$

From the last inequality we observe that the Fejér property is maintained. Hence, $\{x_i\}_1^\infty$ is a bounded sequence. Moreover, $\{\| x_i - z \|\}_1^\infty$ decreases monotonically and is bounded below, therefore it has a limit. This forces $\{\sum_{k=1}^{p} \| d_{ik} \|^2\}_1^\infty \to 0$. Hence, $\{d_{ik}\}_1^\infty \to 0$, $k = 1, \ldots, p$, which happens to be a convenient convergence test.

So far the relaxation parameters μ_{ik} ω_{ik} belong to $(0, 2)$, as required by standard convergence analysis for projection methods. We now show the existence of a λ factor that, in general, takes the relaxation parameters out of the interval $(0, 2)$, but the Fejér property of the sequence $\{\| x_i - z \|\}_1^\infty$ and the desired convergence condition are preserved.

Let us put

$$x(\lambda) \stackrel{\text{def}}{=} x_i + \lambda \sum_{k=1}^{p} \mu_{ik}\omega_{ik}d_{ik}. \tag{16}$$

We then obtain for $\lambda \geq 0$ that

$$[z \in C] \;\Rightarrow\; \left[\|x(\lambda) - z\|^2 \leq \|x_i - z\|^2 + \varphi(\lambda)\right], \tag{17}$$

where by (4)

$$\varphi(\lambda) = -2\lambda \sum_{k=1}^{p} \mu_{ik}\omega_{ik} \|d_{ik}\|^2 + \lambda^2 \left\| \sum_{k=1}^{p} \mu_{ik}\omega_{ik}d_{ik} \right\|^2 . \tag{18}$$

The minimum of $\varphi(\lambda)$ occurs at

$$\lambda_i = \frac{\sum_{k=1}^{p} \mu_{ik}\omega_{ik} \|d_{ik}\|^2}{\left\|\sum_{k=1}^{p} \mu_{ik}\omega_{ik} d_{ik}\right\|^2} > 0 \tag{19}$$

and for any $\alpha \geq 0$ we obtain that

$$\varphi(\alpha\lambda_i) = -\alpha(2-\alpha)\frac{\left[\sum_{k=1}^{p} \mu_{ik}\omega_{ik} \|d_{ik}\|^2\right]^2}{\left\|\sum_{k=1}^{p} \mu_{ik}\omega_{ik} d_{ik}\right\|^2}. \tag{20}$$

If we choose α_i: $\eta \leq \alpha_i \leq 2-\eta$ and let $x_{i+1} \stackrel{\text{def}}{=} x(\alpha_i\lambda_i)$, we ensure by (17) and (20) that the Fejér property is preserved. Consequently $\{\varphi(\alpha_i\,\lambda_i)\}_1^\infty \to 0$ and we can assert that $\{d_{ik}\}_1^\infty \to 0$, $k = 1, \ldots, p$, as long as the sequences $\{\mu_{ik}\omega_{ik}\}_{i=1}^\infty$, $k = 1, \ldots, p$, remain uniformly positive. Note that no upper bounds and no other conditions whatsoever are imposed on the latter sequences.

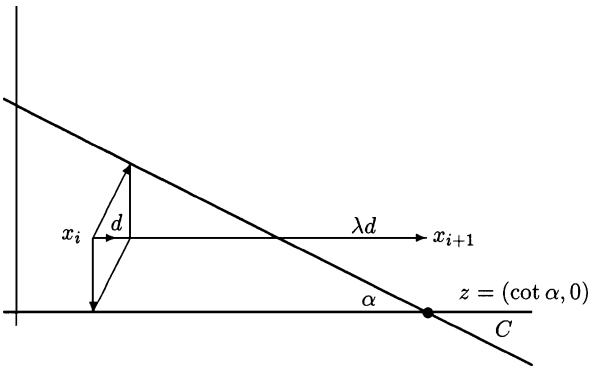
To observe that λ_i may cause an overprojection, assume that $\omega_{ik} = 1$, $k = 1, \ldots, p$, at all iterations; then $x_{i+1} := x(\alpha_i\,\lambda_i)$ becomes

$$x_{i+1} := x_i + \underbrace{\frac{\sum_{k=1}^{p} \mu_{ik} \|d_{ik}\|^2}{\left\|\sum_{k=1}^{p} \mu_{ik} d_{ik}\right\|^2}}_{\lambda_i} \alpha_i \underbrace{\sum_{k=1}^{p} \mu_{ik} d_{ik}}_{d_i}, \tag{21}$$

where $\lambda_i \geq 1$ by (7).

Figure 4 shows one iteration on Example 4 with $\alpha_i = 1$, and $\mu_{ik} = 0.5$.

The net effect of the λ factor is to pull the sequence $\{x_i\}_1^\infty$ out of a wedge of small angle. In the optimization jargon we look for the minimum of a given upper bound of the distance from $x(\lambda)$ to C along the direction $d_i := \sum_{k=1}^{p} \mu_{ik}\omega_{ik}\, d_{ik}$ starting from x_i. It is worth noticing that sometimes inequality (17) becomes an equality; for instance, when C is a linear system of equations. In this case $x(\lambda_i)$ is indeed the closest point to C along d_i.



Relaxation in Projection Methods, Figure 4
λ factor

Let us proceed one step further with our analysis. Denote the vector w in the Euclidean space $\mathbf{R}^p$ with components $w^k = \mu_k\,\omega_k$, $k = 1, \ldots, p$. We then write

$$x(w) = x_i + \sum_{k=1}^{p} w^k d_{ik} \tag{22}$$

and obtain (almost verbatim as argued for $\varphi(\lambda)$)

$$\|x(w) - z\|^2 \leq \|x_i - z\|^2 + \varphi(w),$$

$$\varphi(w) = -2\sum_{k=1}^{p} w^k \|d_{ik}\|^2 + \left\|\sum_{k=1}^{p} w^k d_{ik}\right\|^2.$$

Hence we argue that

$$w_i = \underset{w^k \geq \delta}{\operatorname{argmin}}\, \varphi(w) \tag{23}$$

should yield 'good' (over)relaxation parameters. But the parallel algorithm can degrade severely, unless we have at hand an efficient algorithm for solving the quadratic program (23). Otherwise, any acceptable value for w_i combined with the λ factor can be advantageous.

The net effect of w_i is to obtain 'locally' the best direction of search to locate some $z \in C$. Thus, w_i can be considered as the optimum relaxation parameter. In Example 4, z is obtained in one step.

Table 1 summarizes the accelerated parallel algorithm (APA) that subsumes our work. We dropped the iteration number i for convenience.

Discussion

The literature on projection methods is rather vast and rapidly growing. Reviews on projection methods and/or their applications in mathematics, physics and social sciences published in the 1990s include [1,4,5,7,10,13,21,23], and others.

Projection methods have a long history. In their early inceptions [6,20,22] no relaxation parameters were introduced, but soon it was noticed that overrelaxation (or overprojection) could speed convergence. Successive over relaxation techniques were introduced

Relaxation in Projection Methods, Table 1
Accelerated parallel algorithm

Problem	Find $z \in C = C_1 \cap \ldots \cap C_p$
Data	x, an estimate of z,
	$0 < \eta \leq 1, 0 < \delta \leq 1/p$,
	$0 < \epsilon$ for the convergence test
REPEAT	
	FOR $k = 1, \ldots, p$ DO
	IN parallel
	$y_k = \text{argmin}_{y \in C_k} \|y - x\|^2$,
	$d_k = y_k - x$
	END parallel
	Find $w^k \geq \delta$
	END FOR
	$\lambda = \frac{\sum_{k=1}^{P} w^k \|d_k\|^2}{\|\sum_{k=1}^{P} w^k d_k\|^2}$
	Choose α: $\eta \leq \alpha \leq 2 - \eta$
	Update: $x := x + \lambda\alpha \sum_{k=1}^{P} w^k d_k$
UNTIL	$\sum_{k=1}^{P} \|d_k\| < \epsilon$

in the solution of large linear system of equations. Significant work on determining and computing the optimum relaxation was performed in the 1950s ([19] gives a good list of references on the subject). R. Bramley and A. Sameh [2] found that underprojection could improve convergence, at least theoretically, in the solution of a nonsymmetric linear system of equations.

Convergence rate of projection methods is linear under mild conditions, i. e., $\| x_{i+1} - z \| \leq \gamma_i \| x_i - z \|$, where $\{\gamma_i \in \mathbf{R}\}_1^\infty$ is uniformly positive and strictly less than one. The value of γ_i depends strongly on the angle α_i between supporting hyperplanes of S_i and S_{i+1} [11,18,24]. Overprojection and most techniques used to improve the quality of convergence of projection methods merely attempt to reduce γ_i. For the convex inequality problem, see [12,14]. In [12] projection aggregation methods were developed and the choice of (9) was justified. In [14] a projection algorithm with a superlinear rate of convergence is presented, where $\{\omega_i\}_1^\infty \to 1$ is needed.

We proved that the sequence $\{x_i\}_1^\infty$ generated by the parallel algorithms (PPA and APA) satisfies the Fejér property. Strong convergence in Hilbert spaces for a countable number of sets can be proved under mild additional conditions [1,9, Thm. 2.16]. It is straightforward to show that convergence is preserved if we project on closed supersets $S_{ik} \supseteq C_k$, $k = 1, \ldots, p$. We only need

$$\left\| P_{S_{ik}}(x) - x \right\| \geq \delta \left\| P_{C_k}(x) - x \right\|$$

for some $\delta > 0$.

The λ factor has been suggested in [8,12], and [15]. Numerical results reported in the latter paper, in [9,25] and [17] are impressive. However, the actual theoretical improvement of the quality of convergence is an open question. See [16] for theoretical results when C is the intersection of affine sets (hyperplanes). In [12,15] the use of the λ factor is analysed for sequential versions of the projection method.

We suggest the solution of the quadratic problem (23) to obtain 'optimal' relaxation parameters. K.C. Kiwiel recommends the solution of the nonlinear programming problem (20) [21]. To the author's knowledge, there exists no evidence of the superiority of either approach. We do not believe that the exact solution of either problem is a good strategy, because this can degrade the performance of a parallel algorithm. Our recommendation is to obtain some acceptable initial values for the relaxation parameters, and then use the λ factor.

Throughout the paper we have assumed that $C \neq \emptyset$. The inconsistency case, i. e., $C = \emptyset$, has attracted a lot of attention lately. See [3] and references therein.

See also

- Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- Inequality-constrained Nonlinear Optimization

References

1. Bauschke HH, Borwein JM (1996) On projection algorithms for solving convex feasibility problems. SIAM Rev 38:367–426
2. Bramley R, Sameh A (1992) Row projection methods for large nonsymmetric linear systems. SIAM J Sci Statist Comput 13:168–193
3. Byrne C, Censor Y (1998) Proximity function minimization for separable, jointly convex Bregman distances, with applications. Techn Report, Dept Math Univ Haifa, Israel Feb
4. Censor Y (1990) On variable block algebraic reconstruction techniques. In: Herman G, Louis A, Natterer F (eds) Mathematical Methods in Tomography. Springer, Berlin, pp 133–140

5. Censor Y, Zenios SA (1997) Parallel optimization. Theory, algorithms and applications. Oxford University Press, Oxford
6. Cimmino G (1938) Calcolo approssimate per le soluzioni dei sistemi di equazioni lineari. La Ricerca Sci:326–333
7. Combettes PL (1993) The foundations of set theoretic estimation. In: Proc IEEE 81, pp 182–208
8. Combettes PL (1994) Inconsistent signal feasibility problems: Least square solutions in a product space. IEEE Trans Signal Processing 42:2955–2966
9. Combettes PL (1997) Convex set theoretic image recovery by extrapolated iterations of parallel subgradient projections. IEEE Trans Image Processing 6(4):1–15
10. Deutsch F (1992) The method of alternating projections. In: Singh SP (ed) Theory, Spline Functions and Applications. Kluwer, Dordrecht, pp 105–121
11. Flåm SD, Zowe J (1990) Relaxed outer projections, weighted averages and convex feasibility. Bit 30:289–300
12. García-Palomares UM (1993) Parallel projected aggregation methods for solving the convex feasibility problem. SIAM J Optim 3-4:882–900
13. García-Palomares UM (1994) Aplicación de los métodos de proyección en el problema de factibilidad convexa: Un repaso. Invest Oper 4(3):1–17
14. García-Palomares UM (1998) A superlinearly convergent projection algorithm for solving the convex inequality problem. Oper Res Lett 22:97–103
15. García-Palomares UM, González Castaño FJ (1998) Incomplete projection algorithms for solving the convex feasibility problem. Numer Algorithms 18:177–193
16. Gearhart WB, Koshy M (1989) Acceleration schemes for the method of alternating projections. J Comput Appl Math 26:239–249
17. González Castaño FJ (1998) Contribución al encaminamientóptimo en redes de circuitos virtuales con supervivencia mediante arquitecturas de memoria distribuida. Tesis Doctoral, Dept Tecn Telecomunicaciones, Univ Vigo España
18. Gubin LG, Polyak BT, Raik EV (1967) The method of projections for finding the common point of convex sets. USSR Comput Math Math Phys 7:1–24
19. Householder AS (1964) The theory of matrices in numerical analysis. Blaisdell, New York
20. Kaczmarz S (1937) Angenäherte Auflösung von Systemen Linerer Gleichungen. Bull Internat Acad Polon Sci Lett:355–357
21. Kiwiel KC (1995) Block-iterative surrogate projection methods for convex feasibility problems. Linear Alg Appl 215:225–260
22. Neumann Jvon (1950) Functional operators 2. Princeton University Press, Princeton
23. Sezan MI (1992) An overview of convex projections theory and its applications to image recovery problems. Ultramicroscopy 40:55–67
24. Smith KT, Solomon DC, Wagner SL (1976) Practical and mathematical aspects of the problem of reconstructing objects from radiographs. BullAmer Math Soc 83:1227–1270
25. Özakas H, Akgül M, Pinar M (1997) A new algorithm with long steps for the simultaneous block projections approach for the linear feasibility problem. Techn Report, IEOR Dept Industr Engin, Bilkent Univ Ankara, Turkey, 9703

Replicator Dynamics in Combinatorial Optimization

MARCELLO PELILLO
University Ca' Foscari di Venezia, Venice, Italy

MSC2000: 90C27, 90C20, 90C35, 90C59, 91A22, 37B25, 05C69, 05C60

Article Outline

Keywords
The Model and its Properties
Maximum Clique Problems
Graph Isomorphism
Subtree Isomorphism
A Geometric Problem
Multipopulation Models
Conclusions
See also
References

Keywords

Quadratic programming; Graph; Clique; Evolutionary game theory; Heuristics; Dynamical system

Replicator equations are a class of *dynamical systems* developed and studied in the context of *evolutionary game theory*, a discipline pioneered by J. Maynard Smith [36] which aims to model the evolution of animal behavior using the principles and tools of game theory. Because of their dynamical properties, they have been recently applied with significant success to a number of *combinatorial optimization* problems. It is the purpose of this article to provide a summary and an up-to-date bibliography of these applications.

The Model and its Properties

In this Section we discuss the basic intuition behind replicator equations and present a few theoretical properties that are instrumental for their application to optimization problems. For a more systematic treatment see [23,55].

Consider a large, ideally infinite population of individuals belonging to the same species which compete for a particular limited resource, such as food, territory, etc. This kind of conflict is modeled as a game, the players being pairs of randomly selected population members. In contrast to traditional application fields of game theory, such as economics or sociology [33], players here do not behave 'rationally,' but act instead according to a pre-programmed behavior pattern, or *pure strategy*. Reproduction is assumed to be asexual, which means that, apart from mutation, offspring will inherit the same genetic material, and hence behavioral phenotype, as its parent. Let $J = \{1, \ldots, n\}$ be the set of pure strategies and, for all $i \in J$, let $x_i(t)$ be the relative frequency of population members playing strategy i, at time t. The *state* of the system at time t is simply the vector $\mathbf{x}(t) = (x_1(t), \ldots, x_n(t))^\intercal$. Clearly, the states are constrained to lie in the standard simplex of the n-dimensional Euclidean space $\mathbf{R}^n$:

$$S_n = \left\{\mathbf{x} \in \mathbb{R}^n : \; x_i \geq 0, \; \forall i \in J, \; \mathbf{e}^\top \mathbf{x} = 1\right\}.$$

Here and in the sequel, the letter $\mathbf{e}$ is reserved for a vector of appropriate length, consisting of unit entries (hence $\mathbf{e}^\intercal \mathbf{x} = \sum_i x_i$).

One advantage of applying game theory to biology is that the notion of 'utility' is much simpler and clearer than in human contexts. Here, a player's utility can simply be measured in terms of Darwinian fitness or reproductive success, i. e., the player's expected number of offspring. Let $W = (w_{ij})$ be the $n \times n$ 'payoff' (or fitness) matrix. Specifically, for each pair of strategies $i, j \in J$, w_{ij} represents the payoff of an individual playing strategy i against an opponent playing strategy j. Without loss of generality, we shall assume that the payoff matrix is nonnegative, i. e., $w_{ij} \geq 0$ for all $i, j \in J$. At time t, the average payoff of strategy i is given by:

$$\pi_i(t) = \sum_{j=1}^{n} w_{ij} x_j(t), \tag{1}$$

while the mean payoff over the entire population is $\sum_{i=1}^n x_i(t)\pi_i(t)$.

In evolutionary game theory the assumption is made that the game is played over and over, generation after generation, and that the action of natural selection will result in the evolution of the fittest strategies. If successive generations blend into each other, the evolution of behavioral phenotypes can be described by the following set of differential equations [53]:

$$\dot{x}_i(t) = x_i(t)\left(\pi_i(t) - \sum_{j=1}^{n} x_j(t)\pi_j(t)\right) \tag{2}$$

for $i = 1, \ldots, n$, where a dot signifies derivative with respect to time. The basic idea behind this model is that the average rate of increase $\dot{x}_i(t)/x_i(t)$ equals the difference between the average fitness of strategy i and the mean fitness over the entire population. It is straightforward to show that the simplex S_n is invariant under equation (2) or, in other words, any trajectory starting in S_n will remain in S_n. To see this, simply note that $(d/dt)\sum_i x_i(t) = \sum_i \dot{x}_i(t) = 0$, which means that the (relative) interior of S_n (i. e., the set defined by $x_i > 0$, for all $i = 1, \ldots, n$) is invariant. The additional observation that the boundary too is invariant, completes the proof.

Similar arguments provide a rationale for the following discrete-time version of the replicator dynamics, assuming nonoverlapping generations, which can be obtained from (2) by setting $1/\Delta t = \sum_{j=1}^n x_j(t)\pi_j(t)$:

$$x_i(t + \Delta t) = \frac{x_i(t)\pi_i(t)}{\sum_{j=1}^n x_j(t)\pi_j(t)} \tag{3}$$

for $i = 1, \ldots, n$. Because of the nonnegativity of the fitness matrix W and the normalization factor, this system too makes the simplex S_n invariant as its continuous counterpart.

A point $\mathbf{x} = \mathbf{x}(t)$ is said to be a *stationary* (or equilibrium) point for the dynamical systems under consideration if $\dot{x}_i(t) = 0$ in the continuous-time case, and $x_i(t + \Delta t) = x_i(t)$ in the discrete-time case ($i = 1, \ldots, n$). Moreover, a stationary point is said to be *asymptotically stable* if any trajectory starting in its vicinity will converge to it as $t \to \infty$. It turns out that both the continuous-time and discrete-time replicator dynamics have the same set of stationary points, namely all the

points in S_n satisfying, for all $i = 1, \ldots, n$, the condition

$$x_i(t)\left(\pi_i(t) - \sum_{j=1}^{n} x_j(t)\pi_j(t)\right) = 0,$$

or, equivalently, $\pi_i(t) = \sum_{j=1}^{n} x_j(t)\pi_j(t)$ whenever $x_i > 0$.

Equations (2) and (3) arise independently in different branches of theoretical biology [23]. In population ecology, for example, the famous Lotka–Volterra equations for predator-prey systems turn out to be equivalent to the continuous-time dynamics (2), under a simple barycentric transformation and a change in velocity. In population genetics they are known as *selection equations* [17]. In this case, each x_i represents the frequency of the ith allele A_i and the payoff w_{ij} is the fitness of genotype $A_i\ A_j$. Here the fitness matrix W is always symmetric. The discrete-time dynamical equations turn out to be a special case of a general class of dynamical systems introduced in [2] and studied in [3] in the context of Markov chain theory. They also represent an instance of the so-called *relaxation labeling processes*, a class of parallel, distributed algorithms developed in computer vision to solve (continuous) constraint satisfaction problems [25,44,50]. An independent connection between relaxation labeling processes and game theory has recently been described in [37].

The following theorem states that under replicator dynamics the population's average fitness always increases, provided that the payoff matrix is symmetric (in game theory terminology, this situation is referred to as a doubly symmetric game).

Theorem 1 *Suppose that the (nonnegative) payoff matrix W is symmetric. Then, the quadratic polynomial F defined as*

$$F(\mathbf{x}) = \mathbf{x}^{\top} W\mathbf{x} \tag{4}$$

is strictly increasing along any nonconstant trajectory of both continuous-time (2) and discrete-time (3) replicator equations. In other words, for all $t \geq 0$ we have

$$\frac{d}{dt}F(\mathbf{x}(t)) > 0$$

for system (2), and $F(\boldsymbol{x}(t + \Delta\, t)) > F(\boldsymbol{x}(t))$ for system (3), unless $\boldsymbol{x}(t)$ is a stationary point. Furthermore, any such trajectory converges to a (unique) stationary point.

The previous result is known in mathematical biology as the *fundamental theorem of natural selection* [17,23,55] and, in its original form, traces back to [18]. As far as the discrete-time model is concerned, it can be regarded as a straightforward implication of the Baum–Eagon theorem [2,3] which is valid for general polynomial functions over product of simplices. F.R. Waugh and R.M. Westervelt [54] also proved a similar result for a related class of continuous- and discrete-time dynamical systems. In the discrete-time case, however, they put bounds on the eigenvalues of W in order to achieve convergence to fixed points.

The fact that all trajectories of the replicator dynamics converge to a stationary point has been proved in [32,34]. However, in general, not all stationary points are local maximizers of F on S_n. The vertices of S_n, for example, are all stationary points for (2) and (3) whatever the landscape of F. Moreover, there may exist trajectories which, starting from the interior of S_n, eventually approach a saddle point of F. However, a result proved by I. Bomze [5] asserts that all asymptotically stable stationary points of replicator dynamics correspond to (strict) local maximizers of F on S_n, and vice versa (see [10] for additional results relating the fields of optimization theory, evolutionary game theory and the qualitative behavior of dynamical systems).

Under continuous-time replicator dynamics, the trajectories approach their limits most efficiently in the sense that (2) is a gradient system if one uses the (non-Euclidean) *Shahshahani metric* [23] which, for any point $\mathbf{u} \in S_n$, is defined as

$$d_{\mathbf{u}}(\mathbf{x}, \mathbf{y}) = \sum_{\{i:\ u_i > 0\}} \frac{1}{u_i} x_i y_i.$$

This efficiency result is called *Kimura's maximum principle*.

Maximum Clique Problems

Let $G = (V, E)$ be an undirected graph, where $V = \{1, \ldots, n\}$ is the set of vertices and $E \subseteq V \times V$ is the set of edges. The *order* of G is the number of its vertices, and its *size* is the number of edges. Two vertices $i, j \in V$ are said to be *adjacent* if $(i, j) \in E$. The *adjacency matrix* of G is the $n \times n$ symmetric matrix $A_G = (a_{ij})$ defined as

follows:

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

A subset C of vertices in G is called a *clique* if all its vertices are mutually adjacent, i. e., for all $i, j \in C$, with i $\neq j$, we have $(i, j) \in E$. A clique is said to be *maximal* if it is not contained in any larger clique, and *maximum* if it is the largest clique in the graph. The *clique number*, denoted by $\omega(G)$, is defined as the cardinality of the maximum clique. The *maximum clique problem* is to find a clique whose cardinality equals the clique number.

The maximum clique problem is a well-known example of combinatorial optimization problem, not only because it was one of the first problems shown to be *NP*-complete [19], but also for its theoretical as well as practical implications. Due to the inherent computational complexity of the problem, exact algorithms are guaranteed to return a solution only in a time which increases exponentially with the number of vertices in the graph, and this makes them inapplicable even to moderately large problem instances. Moreover, a series of recent theoretical results show that the problem is in fact difficult to solve even in terms of approximation. Because of these negative results, much effort has recently been directed towards devising efficient heuristics for finding large cliques, for which no formal guarantee of performance may be provided, but are anyway of interest in practical applications. We refer to [8] for a recent survey of results concerning algorithms, complexity and applications of this problem.

In 1965, T.S. Motzkin and E.G. Straus [38] established a remarkable connection between the maximum clique problem and a certain *quadratic programming* problem. Consider the following quadratic function, sometimes called the *Lagrangian* of G:

$$f_G(\mathbf{x}) = \mathbf{x}^\top A_G \mathbf{x} \tag{5}$$

and let $\mathbf{x}^*$ be a global maximizer of f_G on S_n, n being the order of G. In [38] it is proved that the clique number of G is related to $f_G(\mathbf{x}^{*})$ by the following formula:

$$\omega(G) = \frac{1}{1 - f_G(\mathbf{x}^*)}. \tag{6}$$

Additionally, it is shown that a subset of vertices C is a maximum clique of G if and only if its *characteristic vector* $\mathbf{x}^C$, which is the vector of S_n defined as

$$x_i^C = \begin{cases} \dfrac{1}{|C|} & \text{if } i \in C, \\ 0 & \text{otherwise,} \end{cases}$$

is a global maximizer of f_G on S_n. In [21,47], the Motzkin–Straus theorem has been extended by providing a characterization of maximal cliques in terms of local maximizers of f_G on S_n.

Once that the maximum clique problem is formulated in terms of maximizing a quadratic polynomial over the standard simplex, the use of replicator dynamics naturally suggests itself [42]. In fact, consider a replicator system with payoff matrix defined as:

$$W = A_G.$$

From the fundamental theorem of natural selection, we know that the replicator dynamical systems, starting from an arbitrary initial state, will iteratively maximize the Lagrangian f_G in S_n, and will eventually converge to a local maximizer which, by virtue of the Motzkin–Straus formula provides an estimate of the clique number of G. Additionally, if the converged solution happens to be a characteristic vector of some subset of vertices of G, then we are also able to extract the vertices comprising the clique from its nonzero components. Clearly, in theory there is no formal guarantee that the converged solution will be a global maximizer of f_G. However, experimental work suggests that the basins of attraction of global maximizers are quite large, and frequently the algorithm converges to one of them.

In [42], M. Pelillo presents extensive experimental results with the previous approach over thousands of randomly generated graphs. The discrete-time dynamics (3) was used, and the system was started from the vector $(1/n, \ldots, 1/n)^\intercal$ which corresponds to the simplex barycenter. Two series of experiments were conducted. In the first one, graphs with a relatively small number of vertices were considered, i. e. with up to 500 vertices and densities ranging from 0.10 to 0.90. The solutions found by the algorithm were always very close to the optimal ones, as found by standard exact algorithms. In the second part of the study, graphs with up to 2000 vertices and about one million edges were used (in this case all graphs had density 0.50). Here to gauge the quality of the solutions found the *Matula estimate* was employed,

which accurately predicts the clique number in a random graph, when the number of vertices is sufficiently large [35]. Specifically, let

$$M(n,\delta) = 2\log_{1/\delta} n - 2\log_{1/\delta}\log_{1/\delta} + 2\log_{1/\delta}\frac{e}{2} + 1\,.$$

D.W. Matula proved that, as $n \to \infty$, the order of the maximum clique in an n-vertex δ-density random graph is either $\lfloor M(n,\delta)\rfloor$ or $\lceil M(n,\delta)\rceil$ with probability tending to 1, where $\lfloor x\rfloor$ denotes the largest integer less than or equal to x, and $\lceil x\rceil$ denotes the smallest integer greater than or equal to x. Interestingly, it was also shown that the smallest maximal clique is expected to have $M(n,\delta)/2$ vertices [4]. Experimentally, in [42] it was found that the cardinality of the cliques found by the replicator dynamical system turned out to be significantly larger than the estimated minimum, thereby contradicting what is known as the *Jerrum conjecture* [27], which states that in a large 0.5-density random graph it may be hard to find a clique whose order is even a bit larger than that of the smallest maximal clique. A similar conclusion was also drawn in [26]. Overall, the results presented in [42] were competitive with those obtained using more sophisticated neural network heuristics, both in terms of quality of solutions and speed.

One drawback associated with the original Motzkin–Straus formulation, however, relates to the existence of spurious solutions, i. e., maximizers of f_G which are not in the form of characteristic vectors. This was first observed in [40]. To illustrate, consider the path P^3, i. e. the graph with three vertices $\{1, 2, 3\}$ and two edges, one between 1 and 2, and the other between 2 and 3. Clearly, $C = \{1, 2\}$ and $D = \{2, 3\}$ are maximum cliques, and from the Motzkin–Straus theorem it follows that their characteristic vectors $\mathbf{x}^C$ and $\mathbf{x}^D$ are global maximizers of the Lagrangian of P^3 in S_3. However, it can easily be proved that all the points lying on the segment connecting $\mathbf{x}^C$ and $\mathbf{x}^D$, which is a subset of S_3 since the simplex is convex, are also global solutions of the Motzkin–Straus program. See [47] for general characterizations of such spurious solutions. In principle, spurious solutions represent a problem since, while providing information about the cardinality of the maximum clique, they do not allow us to easily extract its vertices.

The spurious solution problem has been solved in [5]. Consider the following regularized version of f_G:

$$\widehat{f}_G(\mathbf{x}) = \mathbf{x}^\top A_G \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathbf{x}, \tag{7}$$

which is obtained from (5) by substituting the adjacency matrix A_G of G with

$$\widehat{A}_G = A_G + \frac{1}{2} I_n,$$

where I_n is the $n \times n$ identity matrix. Unlike the Motzkin–Straus formulation, it can be proved that all maximizers of $\widehat{f}_G$ on S_n are strict, and are characteristic vectors of maximal/maximum cliques in the graph [5].

Theorem 2 *Let C be a subset of vertices of a graph G, and let $\boldsymbol{x}^C$ be its characteristic vector. Then, C is a maximum (maximal) clique of G if and only if $\boldsymbol{x}^C$ is a global (local) maximizer of $\widehat{f}_G$ in S_n. Moreover, all local (and hence global) maximizers of $\widehat{f}_G$ over S_n are strict.*

In an exact sense, therefore, a one-to-one correspondence exists between maximal cliques and local maximizers of $\widehat{f}_G$ in S_n on the one hand and maximum cliques and global maximizers on the other hand.

Preliminary experiments with this regularized formulation (7) on random graphs are reported in [5], and a more extensive empirical study on DIMACS benchmark graphs is presented in [10]. The emerging picture is the following. The solutions produced by the replicator models are typically very close to the ones obtained using more sophisticated continuous-based heuristics. Moreover, the original version of the Motzkin–Straus problem performs slightly better than its regularized counterpart, but the former often returns spurious solutions. This may be intuitively explained by observing that, since all local maxima of $\widehat{f}_G$ are strict, its landscape is certainly less smoothed than the one associated to the nonregularized version. This therefore enhances the tendency of local optimization procedures to get stuck into local maxima. This is the price to pay for the algorithm to return nonspurious, 'informative' solutions.

In order to study the effects of varying the starting point of clique finding replicator dynamics, Bomze and F. Rendl [12] implemented various sophisticated heuristics and compared them with the usual (less expensive) strategy of starting from the simplex barycenter. Surprisingly, they concluded that the amount of

sophistication seems to have no significant impact on the quality of the solutions obtained. Additionally, they showed that using (Runge–Kutta discretizations of) the continuous-time dynamics (2) instead of (3) does not improve efficiency. This analysis indicates that to improve the performance of replicator dynamics on the maximum clique problem one has necessarily to resort to some escape strategies. Various attempts along this direction can be found in [5,6,9,13].

The Motzkin–Straus theorem has been generalized to the weighted case in [21]. Let $G = (V, E, \mathbf{w})$ be a weighted graph, where $V = \{1, \ldots, n\}$ is the vertex set, $E \subseteq V \times V$ is the edge set and $\mathbf{w} \in \mathbf{R}^n$ is the *weight vector*, the ith component of which corresponds to the weight assigned to vertex i. It is assumed that $w_i > 0$ for all $i \in V$. Given a subset of vertices C, the weight assigned to C is defined as

$$W(C) = \sum_{i \in C} w_i.$$

A maximal weight clique C is one that is not contained in any other clique having weight larger than $W(C)$. Since we are assuming that all weights are positive, it is clear that the concepts of maximal clique and maximal weight clique coincide. A maximum weight clique is one having largest total weight, and the *weighted clique number* of G, denoted $\omega(G, \mathbf{w})$, is its weight. The *maximum weight clique problem* is to find a clique C such that $W(C) = \omega(G, \mathbf{w})$ (see [8] for a recent review). The classical (unweighted) version of the maximum clique problem arises as a special case when all vertices have the same weight. For this reason the maximum weight clique problem has at least the same computational complexity as its unweighted counterpart.

Note that the original Motzkin–Straus program for unweighted graphs can be reformulated as a minimization problem by considering the function

$$g(\mathbf{x}) = \mathbf{x}^\top (I + A_{\overline{G}})\mathbf{x},$$

where $A_{\overline{G}}$ is the adjacency matrix of the complement graph $\overline{G}$, which is the graph having the same vertex set as G and $\overline{E} = \{(i, j) \in V \times V : i \neq j \text{ and } (i, j) \notin E\}$ as its edge set. It is straightforward to see that if $\mathbf{x}^*$ is a global minimizer of g in S_n, then $\omega(G) = 1/g(\mathbf{x}^*)$. This is simply a different formulation of the Motzkin–Straus formula (6). Now, consider a weighted graph $G = (V, E, \mathbf{w})$, and let $\mathcal{M}(G, \mathbf{w})$ be the class of symmetric $n \times n$ matrices $M = (m_{ij})_{i,j \in V}$ defined as $2m_{ij} \geq m_{ii} + m_{jj}$ if $(i, j) \notin E$ and $m_{ij} = 0$ otherwise, and $m_{ii} = 1/w_i$ for all $i \in V$. Given a global solution $\mathbf{x}^*$ of the following quadratic program, which is in general indefinite,

$$\begin{cases} \min & g(\mathbf{x}) = \mathbf{x}^\top M \mathbf{x} \\ \text{s.t.} & \mathbf{x} \in S_n, \end{cases} \tag{8}$$

we have [21]:

$$\omega(G, \mathbf{w}) = \frac{1}{g(\mathbf{x}^*)}$$

for any matrix $M \in \mathcal{M}(G, \mathbf{w})$. Furthermore, denote by $\mathbf{x}^C(\mathbf{w})$ the *weighted characteristic vector* of C, which is the vector in S_n with coordinates

$$x_i^C(\mathbf{w}) = \begin{cases} \dfrac{w_i}{W(C)} & \text{if } i \in C, \\ 0 & \text{otherwise.} \end{cases}$$

It turns out that a subset C of vertices is a maximum weight clique if and only if its characteristic vector $\mathbf{x}^C(\mathbf{w})$ is a global minimizer of (8). Notice that the matrix $I + A_{\overline{G}}$ belongs to $\mathcal{M}(G, \mathbf{e})$. In other words, the original Motzkin–Straus theorem turns out to be a special case of the preceding result.

As in the unweighted case, this formulation suffers from the existence of spurious solutions, and this entails the lack of a one-to-one correspondence between the solutions of the continuous optimization problem and those of the original, discrete one. In [11] these spurious solutions are characterized and a regularized version which avoids this kind of problems is introduced (see also [7]). Specifically, let $\mathcal{N}(G, \mathbf{w})$ be the the class of $n \times n$ symmetric matrices $M = (m_{ij})_{i,j \in V}$ defined as $m_{ij} \geq m_{ii} + m_{jj}$ if $(i, j) \notin E$ and $m_{ij} = 0$ otherwise, and $m_{ii} = 1/2w_i$ for all $i \in V$. The following theorem is the weighted counterpart of Theorem 2.

Theorem 3 *Let C be a subset of vertices of a weighted graph $G = (V, E, \mathbf{w})$, and let $\mathbf{x}^C(\mathbf{w})$ be its characteristic vector. Then, for any matrix $M \in N(G, \mathbf{w})$, C is a maximum (maximal) weight clique of G if and only if $\mathbf{x}^C(\mathbf{w})$ is a global (local) solution of program (8). Moreover, all local (and hence global) solutions of (8) are strict.*

Note that $\mathcal{N}(G, \mathbf{w})$ is isomorphic to the positive orthant in $\binom{n}{2} - |E|$ dimensions. This class is a polyhedral

pointed cone with its apex given by the matrix $M(\mathbf{w}) = (m_{ij}(\mathbf{w}))_{i,j \in V}$ with entries

$$m_{ij}(\mathbf{w}) = \begin{cases} \dfrac{1}{2w_i} & \text{if } i = j, \\ \dfrac{1}{2w_i} + \dfrac{1}{2w_j} & \text{if } i \neq j,\ (i,j) \notin E, \\ 0 & \text{if } i \neq j,\ (i,j) \in E. \end{cases}$$

Observe that in the unweighted case, $M(\mathbf{e}) = \mathbf{e}\mathbf{e}^\top - \widehat{A}_G = \widehat{A}_{\overline{G}}$, the regularized adjacency matrix of the complement graph $\overline{G}$. This reflects the elementary property that an independent set of G, i. e. a subset of pairwise nonadjacent vertices, is a clique of $\overline{G}$. Hence, while the local maximizers of $\mathbf{x}^\top \widehat{A}_G \mathbf{x}$ over S_n correspond to maximal cliques of G, the local minimizers of $\mathbf{x}^\top \widehat{A}_G \mathbf{x}$ over S_n correspond to maximal independent sets.

Theorem 3 suggests using replicator equations to approximately solve the maximum weight clique problem. Indeed, note that replicator equations are maximization procedures, while ours is a minimization problem. However, it is a straightforward exercise to see that the problem of minimizing a quadratic form $\mathbf{x}^\mathsf{T}$ M $\mathbf{x}$ on S_n is equivalent to maximizing $\gamma \mathbf{e}\mathbf{e}^\mathsf{T} - M$, where γ is an arbitrary constant. Therefore, the payoff matrix for replicator dynamics to be used in this case is:

$$W = \gamma \mathbf{e}\mathbf{e}^\top - M$$

where $M = (m_{ij})$ is any matrix in $\mathcal{N}(G, \mathbf{w})$, and

$$\gamma = \max_{i,j \in V} m_{ij}.$$

Experiments with this approach on both random graphs and DIMACS benchmark graphs are reported in [11]. Weights were generated randomly in both cases. The results obtained with replicator dynamics (3) were compared with those produced by a very efficient maximum weight clique algorithm of the branch and bound variety. The algorithm performed remarkably well especially on large and dense graphs, and it was typically an order of magnitude more efficient than its competitor.

Graph Isomorphism

Given two graphs $G' = (V', E')$ and $G'' = (V'', \mathrm{E}'')$, an *isomorphism* between them is any bijection ϕ: $V' V''$ such that $(i, j) \in \mathrm{E}' \Leftrightarrow (\phi(i), \phi(j)) \in \mathrm{E}''$, for all $i, j \in$ V′. Two graphs are said to be *isomorphic* if there exists an isomorphism between them. The *graph isomorphism problem* is therefore to decide whether two graphs are isomorphic and, in the affirmative, to find an isomorphism.

The graph isomorphism problem is one of those few combinatorial optimization problems which still resist any computational complexity characterization [19,28]. Despite decades of active research, no polynomial time algorithm for it has yet been found. At the same time, while clearly belonging to *NP*, no proof has been provided that it is *NP*-complete. Indeed, there is strong evidence that this cannot be the case, for otherwise the polynomial hierarchy would collapse [14,52]. The current belief is that the problem lies strictly between the *P* and *NP*-complete classes.

The subgraph isomorphism problem is more general and in fact more difficult, being *NP*-complete [19]. Given two graphs, it is the problem of determining whether one is isomorphic to a subgraph of the other. At the highest level of generality we find the maximum common subgraph problem, which consists of finding the largest isomorphic subgraphs of two graphs. A simpler version of this problem is to find a maximal common subgraph, i. e., an isomorphism between subgraphs which is not included in any larger subgraph isomorphism.

H.G. Barrow and R.M. Burstall [1], and also D. Kozen [30], introduced the notion of an *association graph* as a useful auxiliary graph structure for solving general graph/subgraph isomorphism problems. Specifically, the association graph derived from graphs $G' = (V', E')$ and $G'' = (V'', E'')$ is the undirected graph $G = (V, E)$ where

$$V = V' \times V''$$

and

$$E = \{((i,h),(j,k)) \in V \times V \colon i \neq j, h \neq k, \\ (i,j) \in E' \Leftrightarrow (h,k) \in E''\}.$$

The following straightforward result establishes an equivalence between the graph isomorphism problem and the maximum clique problem [46].

Theorem 4 *Let $G' = (V', E')$ and $G'' = (V'', E'')$ be two graphs of order n, and let G be the corresponding association graph. Then, G' and G'' are isomorphic if and only if*

$\omega(G) = n$. In this case, any maximum clique of G induces an isomorphism between G' and G'', and vice versa. In general, maximum (maximal) cliques in G are in one-to-one correspondence with maximum (maximal) common subgraph isomorphisms between G' and G''.

By virtue of Theorem 2, it is a straightforward exercise to formulate the graph isomorphism problem in terms of a quadratic programming problem. Let G' and G'' be two arbitrary graphs of order n, and let A_G denote the adjacency matrix of the corresponding association graph G, whose order is n^2. The graph isomorphism problem is equivalent to the following program:

$$\begin{cases} \max & \widehat{f}_G(\mathbf{x}) = \mathbf{x}^\top (A + \frac{1}{2} I_{n^2})\mathbf{x} \\ \text{s.t.} & \mathbf{x} \in S_{n^2}. \end{cases} \tag{9}$$

More precisely, G' and G'' are isomorphic if and only if $\widehat{f}_G(\mathbf{x}^*) = 1 - 1/2n$. In this case, any global solution to (9) induces an isomorphism between G' and G'', and vice versa. In general, local (global) solutions to (9) are in one-to-one correspondence with maximal (maximum) common subgraph isomorphisms between G' and G''.

The previous result allows one to use replicator dynamics with payoff matrix

$$W = A_G + \frac{1}{2} I_{n2}$$

as a heuristic for graph isomorphism problems. Starting from an arbitrary initial state, the dynamical system will converge to a local solution of (9). This will correspond to a characteristic vector of a maximal clique in the association graph G which, in turn, will induce an isomorphism between two subgraphs of G' and G'' which is maximal, in the sense that there is no other isomorphism between subgraphs of G' and G'' that includes the one found.

The algorithm outlined above has been tested over hundreds of random 100-vertex graphs with expected densities ranging from 1% to 99%. Except for very sparse and very dense instances, the algorithm was always able to obtain a correct isomorphism very efficiently. In terms of quality of solutions, the result compare favorably with those obtained using more sophisticated state-of-the-art deterministic annealing heuristics which, in contrast to replicator dynamics, are explicitly designed to escape from poor local solutions. As far as computational time is concerned, replicator dynamics turned out to be significantly faster.

In [46] experiments were also done using the following exponential version of replicator equations, which arises as a model of evolution guided by imitation [22,23,24,55]:

$$\dot{x}_i(t) = x_i(t) \left(\frac{e^{\kappa \pi_i(t)}}{\sum_{j=1}^n x_j(t) e^{\kappa \pi_j(t)}} - 1 \right), \tag{10}$$

$i = 1, \ldots, n$, where κ is a positive constant. As κ tends to 0, the orbits of this dynamics approach those of the standard, 'first order' replicator model (2), slowed down by the factor κ; moreover, for large values of κ the model approximates the so-called 'best-reply' dynamics [24]. As it turns out [22], these models behave essentially in the same way as the standard replicator equations (2), the only difference being the size of the basins of attraction around stable equilibria.

A customary way of discretizing equation (10) is given by the following difference equations [15,20]:

$$x_i(t+1) = \frac{x_i(t) e^{\kappa \pi_i(t)}}{\sum_{j=1}^n x_j(t) e^{\kappa \pi_j(t)}}, \tag{11}$$

$i = 1, \ldots, n$. The extensive results reported in [46] with this dynamics show that exponential replicator dynamics may be considerably faster and even more accurate than the standard, first order model.

The approach just described is general and can clearly be extended to deal with subgraph isomorphism or relational structure matching problems [45]. Preliminary experiments, however, seem to indicate that local optima may represent a problem here, especially in matching sparse and dense graphs. In these cases escape procedures like those presented in [5,6,9,13] would be helpful.

Subtree Isomorphism

Given a graph $G = (V, E)$, a *path* is any sequence of distinct vertices $i_0 \cdots i_n$ such that for all $k = 1, \ldots, n$, $(i_{k-1}, i_k) \in E$; in this case, the *length* of the path is n. If $i_0 = i_n$ the path is called a *cycle*. A graph is said to be *connected* if any pair of vertices is joined by a path. The *distance* between two vertices i and j, denoted by $d(i, j)$, is the length of the shortest path joining them (by convention $d(i, j) = \infty$, if there is no such path). Given a subset of

vertices $C \subseteq V$, the *induced subgraph* $G[C]$ is the graph having C as its vertex set, and two vertices are adjacent in $G[C]$ if and only if they are adjacent in G.

A connected graph with no cycles is called a *tree*. A *rooted tree* is one which has a distinguished vertex, called the *root*. The *level* of a vertex i in a rooted tree, denoted by $\mathrm{lev}(i)$, is the length of the path connecting the root to i. Note that there is an obvious equivalence between rooted trees and directed trees, where the edges are assumed to be oriented. We shall therefore use the same terminology typically used for directed trees to define the relation between two adjacent vertices. In particular, if $(i, j) \in E$ and $\mathrm{lev}(j) - \mathrm{lev}(i) = +1$, we say that i is the *parent* of j and, conversely, j is a *child* of i. Trees have a number of interesting properties. One which turns out to be very useful for our characterization is that in a tree any two vertices are connected by a *unique path*.

Let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be two rooted trees. Any bijection $\phi H_1 \rightarrow H_2$, with $H_1 \subseteq V_1$ and $H_2 \subseteq V_2$, is called a *subtree isomorphism* if it preserves the adjacency and hierarchical relationships between the vertices and, in addition, the subgraphs obtained when we restrict ourselves to H_1 and H_2, i. e., $T_1[H_1]$ and $T_2[H_2]$, are trees. The former condition amounts to stating that, given $i, j \in H_1$, we have $(i, j) \in E_1$ if and only if $(\phi(i), \phi(j)) \in E_2$, and i is the parent of j if and only if $\phi(i)$ is the parent of $\phi(j)$. A subtree isomorphism is *maximal* if there is no other subtree isomorphism ϕ': $H_1' \rightarrow H_2'$ with H_1 a strict subset of H_1', and *maximum* if H_1 has largest cardinality. The maximal (maximum) subtree isomorphism problem is to find a maximal (maximum) subtree isomorphism between two rooted trees. This is a problem solvable in polynomial time [19].

Let i and j be two distinct vertices of a rooted tree T, and let $\mathrm{i} = x_0 \cdots x_n = j$ be the (unique) path joining them. The *path-string* of i and j, denoted by $\mathrm{str}(i, j)$, is the string $s_1 \cdots s_n$ on the alphabet $\{-1, +1\}$ where, for all $k = 1, \ldots, n$, $s_i = \mathrm{lev}(x_k) - \mathrm{lev}(x_{k-1})$. By convention, when $i = j$ we define $\mathrm{str}(i, j) = \varepsilon$, where ε is the null string (i. e., the string having zero length). The path-string concept has a very intuitive meaning. Because of the orientation induced by the root, only two types of elementary moves can be done from any given vertex, i. e., going down to one of the children (if one exists) or going up to the parent (if the vertex is not the root). Assigning to the first move the label +1, and to the second the label −1, the path-string of i and j is simply the string of elementary moves required to move from i to j, following the unique path joining them.

The *tree association graph* (TAG) of two rooted trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ is the graph $G = (V, E)$ where

$$V = V_1 \times V_2 \tag{12}$$

and, for any two vertices (i, h) and (j, k) in V, we have

$$((i, h), (j, k)) \in E \Leftrightarrow \mathrm{str}(i, j) = \mathrm{str}(h, k). \tag{13}$$

The following theorem establishes a one-to-one correspondence between the maximum subtree isomorphism problem and the maximum clique problem [48].

Theorem 5 *Any maximal (maximum) subtree isomorphism between two rooted trees induces a maximal (maximum) clique in the corresponding TAG, and vice versa.*

In many practical applications the trees being matched have vertices with an associated vector of symbolic and/or numeric attributes. The framework just described can naturally be extended for solving attributed tree matching problems [48].

Formally, an *attributed tree* is a triple $T = (V, E, \alpha)$, where (V, E) is the 'underlying' rooted tree and α is a function which assigns an attribute vector $\alpha(i)$ to each vertex $i \in V$. It is clear that in matching two attributed trees, the objective is to find an isomorphism which pairs vertices having 'similar' attributes. To this end, let σ be any similarity measure on the attribute space, i. e., any (symmetric) function which assigns a positive number to any pair of attribute vectors. If ϕ: $H_1 \rightarrow H_2$ is a subtree isomorphism between two attributed trees $T_1 = (V_1, E_1, \alpha_1)$ and $T_2 = (V_2, E_2, \alpha_2)$, the overall similarity between the induced subtrees $T_1[H_1]$ and $T_2[H_2]$ can be defined as follows:

$$S(\phi) = \sum_{i \in H_1} \sigma(\alpha_1(i), \alpha_2(\phi(i))).$$

The isomorphism ϕ is called a *maximal similarity subtree isomorphism* if there is no other subtree isomorphism ϕ': $H'_1 \rightarrow H'_2$ such that H_1 is a strict subset of H_1' and $S(\phi) < S(\phi')$. It is called a *maximum similarity subtree isomorphism* if $S(\phi)$ is largest among all subtree isomorphisms between T_1 and T_2.

The *weighted TAG* of two attributed trees T_1 and T_2 is the weighted graph $G = (V, E, \mathbf{w})$, where V and E are defined as in (12) and (13), and $\mathbf{w}$ is a vector which assigns a positive weight to each vertex $(i, h) \in V = V_1 \times V_2$ as follows:

$$w_{ih} = \sigma(\alpha_1(i), \alpha_2(h)).$$

The following result is the weighted counterpart of Theorem 5 [48].

Theorem 6 *Any maximal (maximum) similarity subtree isomorphism between two attributed trees induces a maximal (maximum) weight clique in the corresponding weighted TAG, and vice versa.*

Theorems 5 and 6 provide a formal justification for applying replicator dynamics to find maximal subtree isomorphisms. In [48] this approach has been applied in computer vision to the problem of matching articulated and deformed visual shapes described by 'shock' trees, an abstract representation of shape based on the singularities arising during a curve evolution process. The experiments, conducted on a number of shapes representing various object classes, yielded very good results, both in the weighted and in the unweighted case. The system typically converged towards the globally optimal solutions in only a few seconds, and compared favorably with another powerful tree matching algorithm.

A Geometric Problem

Let $G = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be a finite set of points in $\mathbf{R}^n$. The *convex hull* of G, denoted by conv(G), is defined as the smallest convex set containing G. A basic problem in computational geometry is to determine whether a given query point $\mathbf{y}$ is inside or outside conv(G) [49]. This task can easily be accomplished by a replicator dynamical system [43]. Such an algorithm can be used as a subroutine for solving more general geometric problems, such as the polygon inclusion and the convex hull problems.

Consider the $n \times m$ real matrix defined as $X = [\mathbf{x}_1 \cdots \mathbf{x}_m]$. It is well known that conv (G) can be written as

$$\text{conv}(G) = \{\mathbf{u} \in \mathbb{R}^n : \ \mathbf{u} = X\mathbf{v}, \mathbf{v} \in S_m\}.$$

Given an arbitrary point $\mathbf{y} \in \mathbf{R}^n$ the measure

$$E(\mathbf{y}, G) = \min_{\mathbf{v} \in S_m} \|X\mathbf{v} - \mathbf{y}\|_2,$$

sometimes called the *exteriority* of $\mathbf{y}$ to conv(G), is just the Euclidean distance between $\mathbf{y}$ and its closest point in conv(G). The exteriority measure can provide useful information about the ability of neural networks to generalize well [16]. Clearly, $\mathbf{y} \in \text{conv}(G)$ if and only if $E(\mathbf{y}, G) = 0$, in which case the closest point to $\mathbf{y}$ is $\mathbf{y}$ itself.

For convenience, the problem of evaluating $E(\mathbf{y}, G)$ is translated into the equivalent (but more manageable) quadratic program:

$$\begin{cases} \min & C(\mathbf{v}) = \frac{1}{2}\|X\mathbf{v} - \mathbf{y}\|_2^2 \\ \text{s.t.} & \mathbf{v} \in S_m. \end{cases} \tag{14}$$

It is a well-known fact that C is convex (strictly convex indeed if the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$ happen to be linearly independent), and this implies that all local minima of C are also global minima. Any descent procedure is therefore guaranteed to approach the global optimal solution in this case, without the risk of getting trapped into poor local minima.

It is interesting to note that a similar optimization problem, known as the problem of 'optimal stability', also arises in the context of learning in perceptron neural networks, where the goal is to derive the parameters of the network so as to ensure larger basins of attraction [31,51]. Moreover, our problem turns out to be closely related to that of determining whether a given set of prototype vectors can be stored in a neural network associative memory [29].

Note that the quadratic objective function in (14) is explicitly written as follows:

$$C(\mathbf{v}) = \frac{1}{2}\mathbf{v}^\top X^\top X\mathbf{v} - \mathbf{y}^\top X\mathbf{v} + \frac{1}{2}\mathbf{y}^\top\mathbf{y},$$

which is a nonhomogeneous quadratic polynomial. In order for replicator equations to find a solution of problem (14), we need to construct the payoff matrix as

$$W = X^\top X$$

and to replace the π function defined in (1) with:

$$\pi_i(t) = \sum_{j=1}^{n} w_{ij}x_j(t) + s_i,$$

where s_i equals the ith component of $-X^\top\mathbf{y}$. After a proper rescaling of W and the s_i's, it is readily seen that C is a Liapunov function for both continuous-time

(2) and discrete-time (3) dynamics. The algorithms will converge to a local solution of (14), say $\mathbf{v}^*$, starting from any interior point. Owing to the convexity of C, $\mathbf{v}^*$ will be also a global minimizer of C, so that the exteriority can be calculated as:

$$E(\mathbf{y}, G) = \sqrt{2C(\mathbf{v}*)}.$$

In [43], experiments with a simple toy problem demonstrate the validity of the approach.

Multipopulation Models

The single-population replicator equations discussed so far can easily be generalized to the case where interactions take place among $n \geq 2$ individuals randomly drawn from n distinct populations [23,55]. In this case, the continuous-time dynamics (2) becomes

$$\dot{x}_i^\mu(t) = x_i^\mu(t)\left(\pi_i^\mu(t) - \sum_\nu x_i^\nu(t)\pi_i^\nu(t)\right), \qquad (15)$$

and its discrete-time counterpart is

$$x_i^\mu(t+\Delta t) = \frac{x_i^\mu(t)\pi_i^\mu(t)}{\sum_\nu x_i^\nu(t)\pi_i^\nu(t)}. \qquad (16)$$

The function π can either be linear, as in (1), or can take a more general form. If there exists a polynomial F such that

$$\pi_i^\mu = \frac{\partial F}{\partial x_i^\mu},$$

then it can be proved that F strictly increases along any trajectory of both dynamics [2,3,23]. Note that these dynamics work in a product of standard simplices.

H. Mühlenbein et al. [39] used multipopulation replicator equations to approximately solve the *graph partitioning problem*, which is *NP*-complete [19]. Given a graph $G = (V, E)$ with edge weights w_{ij}, their goal was to partition the vertices of G into a predefined number of clusters in such a way as to maximize the overall intrapartition traffic

$$F = \prod_\mu K^\mu,$$

where

$$K^\mu = \sum_i \sum_j w_{ij} x_i^\mu x_j^\mu$$

is the intrapartition traffic for cluster μ. Here, x_i^μ can be interpreted as the probability that vertex i belongs to cluster μ.

By putting

$$\pi_i^\mu = \frac{2F\sum_j w_{ij}x_j^\mu}{K^\mu},$$

the replicator equations seen above will indeed converge toward a maximizer of F. However, in so doing the system typically converges towards an interior attractor, thereby giving an infeasible solution. To avoid this problem, Mühlenbein et al. [39] put a 'selection pressure' parameter S on the main diagonal of the weight matrix, and altered it during the evolution of the process. Intuitively, $S = 0$ has no influence on the system. Negative values of S prevent the vertices to decide for a partition, whereas positive values force the vertices to take a decision. The proposed algorithm starts with a negative value of S, and makes the discrete-time dynamics (16) evolve. After convergence, if an infeasible solution has been found, S is increased and the algorithm is started again. The entire procedure is iterated until convergence to a feasible solution. A similar, but more principled, strategy for the maximum clique problem can be found in [9]. The results presented in [39] on a particular problem instance are fairly encouraging. However, more experiments on larger and diverse graphs are needed to fully assess the potential of the approach.

Multipopulation replicator models have also been used in [39,41] to solve the traveling salesman problem, which asks for the shortest closed tour connecting a given set of cities, subject to the constraint that each city be visited only once. The results presented on small problem instances, i. e., up to 30 cities, are encouraging but it seems that the results do not scale well with the size of the problem.

Conclusions

Despite their simplicity and inherent inability to escape from local solutions, replicator dynamics have proved to be a useful heuristic for attacking a variety of combinatorial optimization problems. They are completely devoid of operational parameters, which typically require a lengthy, problem-dependent tuning phase, and are especially suited for parallel hardware implementation.

See also

- Combinatorial Matrix Analysis
- Combinatorial Optimization Algorithms in Resource Allocation Problems
- Combinatorial Optimization Games
- Evolutionary Algorithms in Combinatorial Optimization
- Fractional Combinatorial Optimization
- Graph Coloring
- Greedy Randomized Adaptive Search Procedures
- Heuristics for Maximum Clique and Independent Set
- Multi-objective Combinatorial Optimization
- Neural Networks for Combinatorial Optimization
- Neuro-dynamic Programming
- Unconstrained Optimization in Neural Network Training

References

1. Barrow HG, Burstall RM (1976) Subgraph isomorphism, matching relational structures and maximal cliques. Inform Process Lett 4(4):83–84
2. Baum LE, Eagon JA (1967) An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. Bull Amer Math Soc 73:360–363
3. Baum LE, Sell GR (1968) Growth transformations for functions on manifolds. Pacific J Math 27(2):211–227
4. Bollobás B, Erdös P (1976) Cliques in random graphs. Math Proc Cambridge Philos Soc 80:419–427
5. Bomze IM (1997) Evolution towards the maximum clique. J Global Optim 10:143–164
6. Bomze IM (1997) Global escape strategies for maximizing quadratic forms over a simplex. J Global Optim 11:325–338
7. Bomze IM (1998) On standard quadratic optimization problems. J Global Optim 13:369–387
8. Bomze IM, Budinich M, Pardalos PM, Pelillo M (1999) The maximum clique problem. In: Du D-Z, Pardalos PM (ed) Handbook Combinatorial Optim, Suppl A. Kluwer, Dordrecht, pp 1–74
9. Bomze IM, Budinich M, Pelillo M, Rossi C (2000) An new "annealed" heuristic for the maximum clique problem. Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems. In: Pardalos PM (eds) Kluwer, Dordrecht, pp 78–95
10. Bomze IM, Pelillo M, Giacomini R (1997) Evolutionary approach to the maximum clique problem: Empirical evidence on a larger scale. Developments in Global Optimization, pp 95–108
11. Bomze IM, Pelillo M, Stix V (2000) Approximating the maximum weight clique using replicator dynamics. IEEE Trans Neural Networks 11(6)
12. Bomze IM, Rendl F (1998) Replicator dynamics for evolution towards the maximum clique: Variations and experiments. High Performance Algorithms and Software in Nonlinear Optimization. In: De Leone R, Murlï A, Pardalos PM, Toraldo G (eds) Kluwer, pp 53–67
13. Bomze IM, Stix V (1999) Genetic engineering via negative fitness: Evolutionary dynamics for global optimization. Ann Oper Res 90
14. Boppana RB, Hastad J, Zachos S (1987) Does co-NP have short interactive proofs? Inform Process Lett 25:127–132
15. Cabrales A, Sobel J (1992) On the limit points of discrete selection dynamics. J Econom Theory 57:407–419
16. Courrieu P (1994) Three algorithms for estimating the domain of validity of feedforward neural networks. Neural Networks 7(1):169–174
17. Crow JF, Kimura M (1970) An introduction to population genetics theory. Harper and Row, New York
18. Fisher RA (1930) The genetical theory of natural selection. Oxford Univ Press, Oxford
19. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, New York
20. Gaunersdorfer A, Hofbauer J (1995) Fictitious play, Shapley polygons, and the replicator equation. Games Econom Behav 11:279–303
21. Gibbons LE, Hearn DW, Pardalos PM, Ramana MV (1997) Continuous characterizations of the maximum clique problem. Math Oper Res 22:754–768
22. Hofbauer J (1995) Imitation dynamics for games. Preprint Collegium Budapest
23. Hofbauer J, Sigmund K (1998) Evolutionary games and population dynamics. Cambridge Univ Press, Cambridge
24. Hofbauer J, Weibull JW (1996) Evolutionary selection against dominated strategies. J Econom Theory 71:558–573
25. Hummel RA, Zucker SW (1983) On the foundations of relaxation labeling processes. IEEE Trans Pattern Anal Machine Intell 5:267–287
26. Jagota A (1995) Approximating maximum clique with a Hopfield neural network. IEEE Trans Neural Networks 6:724–735
27. Jerrum M (1992) Large cliques elude the Metropolis process. Random Struct Algorithms 3:347–359
28. Johnson DS (1988) The NP-completeness column: An ongoing guide. J Algorithms 9:426–444
29. Kamp Y, Hasler M (1990) Recursive neural networks for associative memory. Wiley, New York
30. Kozen D (1978) A clique problem equivalent to graph isomorphism. SIGACT News 50–52
31. Krauth W, MÉzard M (1987) Learning algorithms with optimal stability in neural networks. J Phys A 20:L745–L752

32. Losert V, Akin E (1983) Dynamics of games and genes: Discrete versus continuous time. J Math Biol 17:241–251
33. Luce RD, Raiffa H (1957) Games and decisions. Wiley, New York
34. Yu. Lyubich, Maistrowskii GD, Ol'khovskii Yu G (1980) Selection-induced convergence to equilibrium in a single-locus autosomal population. Probl Inform Transmission 16:66–75
35. Matula DW (1976) The largest clique size in a random graph. Techn Report, Dept Computer Sci, Southern Methodist Univ, CS7608
36. Maynard-Smith J (1982) Evolution and the theory of games. Cambridge Univ Press, Cambridge
37. Miller DA, Zucker SW (1991) Copositive-plus Lemke algorithm solves polymatrix games. Oper Res Lett 10:285–290
38. Motzkin TS, Straus EG (1965) Maxima for graphs and a new proof of a theorem of Turán. Canad J Math 17:533–540
39. Mühlenbein H, Gorges-Schleuter M, Kräamer O (1988) Evolution algorithms in combinatorial optimization. Parallel Comput 7:65–85
40. Pardalos PM, Phillips AT (1990) A global optimization approach for solving the maximum clique problem. Internat J Comput Math 33:209–216
41. Pelillo M (1993) Relaxation labeling processes for the traveling salesman problem. Proc Internat J Conf Neural Networks, pp 2429–2432
42. Pelillo M (1995) Relaxation labeling networks for the maximum clique problem. J Artif Neural Networks 2:313–328
43. Pelillo M (1996) A relaxation algorithm for estimating the domain of validity of feedforward neural networks. Neural Proc Lett 3, no(3):113–121
44. Pelillo M (1997) The dynamics of nonlinear relaxation labeling processes. J Math Imaging Vision 7(4):309–323
45. Pelillo M (1998) A unifying framework for relational structure matching. Proc 14th Internat Conf Pattern Recognition, IEEE Computer Soc Press, pp 1316–1319
46. Pelillo M (1999) Replicator equations, maximal cliques, and graph isomorphism. Neural Computation 11(8):2023–2045
47. Pelillo M, Jagota A (1995) Feasible and infeasible maxima in a quadratic program for maximum clique. J Artif Neural Networks 2:411–420
48. Pelillo M, Siddiqi K, Zucker SW (1999) Matching hierarchical structures using association graphs. IEEE Trans Pattern Anal Machine Intell 21(11):1105–1120
49. Preparata FP, Shamos MI (1985) Computational geometry: An introduction. Springer, Berlin
50. Rosenfeld A, Hummel RA, Zucker SW (1976) Scene labeling by relaxation operations. IEEE Trans Syst, Man Cybern 6:420–433
51. Ruján P (1993) A fast method for calculating the perceptron with maximal stability. J Phys I France 3:277–290
52. Schöning U (1988) Graph isomorphism is in the low hierarchy. J Comput Syst Sci 37:312–323
53. Taylor P, Jonker L (1978) Evolutionarily stable strategies and game dynamics. Math Biosci 40:145–156
54. Waugh FR, Westervelt RM (1993) Analog neural networks with local competition. I. Dynamics and stability. Phys Rev E 47(6):4524–4536
55. Weibull JW (1995) Evolutionary game theory. MIT, Cambridge

Resource Allocation for Epidemic Control

MARGARET BRANDEAU
Stanford University, Stanford, USA

MSC2000: 49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX

Article Outline

Keywords
Epidemic Models
Analytical Results
Numerical Analyses
Practical Tools for Decision Makers
See also
References

Keywords

Optimal control; Markov process; Ordinary differential equations; Resource allocation; Epidemiology; Systems theory and control

This article describes *resource allocation* for control of epidemics of infectious diseases in humans, particularly diseases that are spread directly between individuals, such as sexually-transmitted diseases and influenza. In its most general form, the problem is to determine the optimal amount to spend over time and in different populations on programs for controlling the spread of an infectious disease. Restricted versions of the problem include that of determining the level of investment over time in a single program targeted to a single population; determination of the appropriate investment in competing interventions targeted to the same population; and determination of the appropriate allocation of resources for a single intervention targeted to different populations.

While resource allocation problems have been studied by economists and operations researchers for many years (see, for example, [4,10,26]), resource allocation

for *epidemic control* poses special challenges. Epidemics of infectious diseases are dynamic and are inherently nonlinear: while an epidemic is growing, saving one person today from getting infected could translate into scores of people being saved from infection over time. In addition, an epidemic may progress differently in different populations, and different programs for the control of an infectious disease can have very different costs and effectiveness.

Approaches to the problem of resource allocation for epidemic control include analytical derivations for simple epidemic models, numerical analysis of more sophisticated epidemic models, and heuristic approaches for decision makers. Before these approaches are described, a simple *epidemic model* is presented. Understanding this type of model is key to understanding the resource allocation problem.

Epidemic Models

One of the simplest types of epidemic models assumes transmission of an infectious disease within a closed population that is divided into three subgroups: uninfected individuals, infected individuals, and those removed from the infection-transmission process [12,13]. To specify the model, let $x(t)$ represent the number of uninfected individuals in the population at time t, $y(t)$ the number of infected individuals in the population at time t (these individuals are assumed to be infectious), and $z(t)$ the number of individuals removed from the infection-transmission process at time t. Let $\beta(t)$ be the rate of infection-transmitting contacts at time t, $u(t)$ the rate at which susceptibles are immunized at time t, and $\gamma(t)$ the rate of removal from the population at time t. The model can be written as

$$\frac{dx(t)}{dt} = -\beta(t)x(t)y(t) - u(t), \tag{1}$$

$$\frac{dy(t)}{dt} = \beta(t)x(t)y(t) - \gamma(t)y(t), \tag{2}$$

$$\frac{dz(t)}{dt} = \gamma(t)y(t) + u(t) \tag{3}$$

This model and similar models provide the foundation for much of the work that has been done on optimal resource allocation for epidemic control.

A key feature of this and other models of infectious disease is a nonlinear growth rate that is a function of the size of the uninfected group multiplied by the size of the infected group. More sophisticated models may include features such as entry into and exit from the population, further subdivision of the population by risk group and disease stage, different types of infectious contacts, variable infectivity rates, and stochastic parameters. Comprehensive expositions of mathematical epidemic models can be found in [2] and [3].

Analytical Results

A number of researchers have considered the application of *control theory* to simple epidemic models similar to that outlined above with the goal of obtaining analytical results characterizing the form of the optimal solution. The parameters of the model may be assumed to be deterministic or stochastic. Examples of controls typically considered include vaccination of susceptibles (which increases the rate $u(t)$), treatment or removal of infectious persons (which increases the rate $\gamma(t)$), and reduction in the sufficient contact rate ($\beta(t)$) between susceptibles and infectious persons. A finite or infinite time horizon may be considered. The goal is to determine the optimal control over time: for example, the optimal rate of immunization $u(t)$ for $0 \leq t \leq T$. For analytical tractability, most analyses assume that only one type of control is applied (and thus only one parameter is affected by the control).

A typical objective in the application of such control might be to minimize the cost of control (e. g., immunization cost plus the fixed cost of establishing the immunization program) plus the cost associated with the number of individuals who become infected. With the exception of the fixed cost of establishing a control program, costs are usually assumed to be linear: the cost of control is a constant multiplied by the affected parameter, and the cost of disease is a constant multiplied by the number of individuals who become infected.

Use of simple epidemic models and linear cost functions allows for characterization of the form of the optimal solution: for example, immunization of all individuals until the epidemic is reduced to a certain level (a 'bang-bang' solution with a single switching point). A survey up to 1977 of the application of control theory to infectious diseases can be found in K. Wickwire

[27]. More recent work of this type can be found in D. Greenhalgh [6], S.P. Sethi [23], and Sethi and P.W. Staats [24].

A related type of analysis considers the optimal timing of interventions for epidemic control. For example, H.L. Lee and W.P. Pierskalla [15] considered mass screening for a contagious disease with no latent period. The goal is to minimize the average number of infected individuals in the population over a fixed time horizon. The authors showed that under certain assumptions, an optimal strategy is mass screening at equal time intervals.

Another type of analysis considers allocation of resources among different population subgroups with the goal of disease eradication (reducing the disease equilibrium in each population to zero). Use of an equilibrium condition allows for analytical tractability. For example, R.M. May and R.M. Anderson [17] considered the distribution of vaccine among several heterogeneously mixing populations. The goal is to eradicate the disease with as little vaccine as possible. They characterized the optimal fraction of each population that should be immunized. As another example, J. Abounadi and L.M. Wein [1] analyzed resource allocation for control of human immunodeficiency virus (HIV) among homogeneously mixing and heterogeneously mixing population groups. Expenditure of resources on a given population group reduces that group's contact rates. The goal is to eradicate the disease while minimizing the total cost of control (which is a monotonically increasing function of the reduction in the contact rates). The authors developed analytical results characterizing the amount of resource that should be spent on each population.

More recent analytical work by A. Richter, M.L. Brandeau and G.S. Zaric [22] considers allocation of resources across nonmixing populations with the goal of minimizing the total number of new infections that occur over a fixed time horizon. Resources for transmission reduction are to be allocated among the population groups subject to a constraint on total available resources (cost of control is a monotonically increasing function of the reduction in the transmission rate) and subject to limits on attainable transmission rates in each population. The authors establish conditions under which the resource allocation problem is convex or concave, as well as other analytical results. Zaric [28] extended such analyses to more sophisticated epidemic models.

Numerical Analyses

Another approach to the problem of resource allocation for epidemic control uses numerical analysis of more sophisticated epidemic models, often tailored for specific diseases. Several notable examples are mentioned here.

Using a model of tuberculosis epidemiology, policies for the control of tuberculosis in developing countries were analyzed by C. ReVelle and colleagues in [18] and [19]. Numerical analysis was used to determine the set of interventions that minimizes the cost of control required to achieve a specified number of active cases at the end of a given time horizon. I.M. Longini, E. Ackerman and L.R. Elveback [16] used numerical analysis to determine the optimal distribution of a fixed amount of vaccine among different age groups during an Influenza A epidemic. H.W. Hethcote and J.A. Yorke [8] and Hethcote, Yorke and A. Nold [9] used numerical analysis of an epidemic model to evaluate the equilibrium epidemic state for policies aimed at controlling the spread of gonorrhea. Although the results were not compared explicitly with program cost, the authors suggested that such comparison must be made before the appropriate allocation of resources can be determined.

A framework similar to that of [22] was used by Richter, Brandeau and D.K. Owens [21] to evaluate allocation of HIV prevention resources among different programs targeted to different risk groups in the patient population of a large health care system. The authors estimated the cost-effect function associated with each intervention as applied to each population (defined as the reduction in transmission that could be achieved as a function of expenditure), and used numerical analysis to determine the optimal allocation of a fixed budget across the prevention programs and the populations.

In [5], C.M. Friedrich and Brandeau investigated the optimal level of funding for a single HIV prevention program targeted to a single population using a simple epidemic model. Via simulation, the authors developed qualitative insights into the nature of the optimal investment (e. g., spend no money, spend some of the money, or spend as much money as possible) for different types of cost-effect functions.

Another approach to the resource allocation problem uses ideas from artificial intelligence. W.Y. Tan and S. Yakowitz [25] proposed the application of a *machine-learning algorithm* for a *Markov decision process* (e. g., see T.L. Lai and Yakowitz [14]) to determine the optimal policy for control of an epidemic. The approach was illustrated using numerical analysis of a stochastic model of the HIV epidemic in a single population. The goal is to minimize the number of new infections by continuously allocating a fixed amount of resources between programs that either lower the contact rate or lower the infectivity per contact. As the learning algorithm progresses, the resource allocation progresses to an equilibrium; [25] showed that this equilibrium is the optimal allocation.

Practical Tools for Decision Makers

Existing analytical work on resource allocation for epidemic control has limited applicability due to the simple epidemic models and simple control policies that are assumed. Numerical approaches can provide applicable results, but require the development of realistic epidemic models and the collection of a significant amount of data. As an alternative, E.H. Kaplan [11] has proposed a heuristic approach for community planners who must allocate HIV prevention resources. He also suggests that decision makers subjectively construct *production functions* that estimate the number of new HIV infections that would occur in a particular population group if x incremental dollars were spent on a given HIV prevention program targeted to that group. Then the problem of allocating a fixed prevention budget among prevention programs and populations so as to minimize new HIV cases reduces to a knapsack problem.

See also

► Combinatorial Optimization Algorithms in Resource Allocation Problems

References

1. Abounadi J, Wein LM (1993) Resource allocation for AIDS. Oral Presentation ORSA/TIMS Meeting, Phoenix
2. Anderson RM, May RM (1991) Infectious diseases of humans: dynamics and control. Oxford Univ Press, Oxford
3. Bailey NTJ (1975) The mathematical theory of infectious diseases and its applications. Hafner, New York
4. Denardo EV (1982) Dynamic programming. Prentice-Hall, Englewood Cliffs
5. Friedrich CM, Brandeau ML (1998) Using simulation to find optimal funding levels for HIV prevention programs with different costs and effectiveness. In: Proc 1998 Medical Sci Simulation Conf, Soc Computer Simulation, pp 58–64
6. Greenhalgh D (1986) Control of an epidemic spreading in a heterogeneously mixing population. Math Biosci 80:23–45
7. Hethcote HW (1976) Qualitative analyses of communicable disease models. Math Biosci 28:335–356
8. Hethcote HW, Yorke JA (1984) Gonorrhea transmission dynamics and control. Lect Notes Biomath, vol 56
9. Hethcote HW, Yorke JA, Nold A (1982) Gonorrhea modeling: A comparison of control methods. Math Biosci 58:93–109
10. Hillier FS, Lieberman GJ (1993) Operations research. Holden Day, San Francisco
11. Kaplan EH (1997) Economic evaluation and HIV prevention community planning: A policy analyst's perspective. In: Holtgrave DR (ed) Handbook HIV Prevention Policy Analysis. Plenum, New York
12. Kermack WO, McKendrick AG (1927) Contributions to the mathematical theory of epidemics, Part I. Proc Royal Statist Soc (Ser A) 115:700–721
13. Kermack WO, McKendrick AG (1932) Contributions to the mathematical theory of epidemics, Part II. Proc Royal Statist Soc (Ser A) 138:55–83
14. Lai TL, Yakowitz S (1995) Machine learning and nonparametric bandit theory. IEEE Trans Autom Control 40:1199–1210
15. Lee HL, Pierskalla WP (1988) Mass screening models for contagious diseases with no latent period. Oper Res 36:917–928
16. Longini IM, Ackerman E, Elveback LR (1978) An optimization model for influenza A epidemics. Math Biosci 38:141–157
17. May RM, Anderson RM (1984) Spatial heterogeneity and the design of immunization programs. Math Biosci 72:83–111
18. ReVelle C, Feldmann F, Lynn W (1969) An optimization model of tuberculosis epidemiology. Managem Sci 16:B190–B211
19. ReVelle C, Male J (1970) A mathematical model for determining case finding and treatment activities in tuberculosis control programs. Amer Rev Resp Dis 102:403–411
20. Richter A (1996) Optimal resource allocation for epidemic control. PhD Thesis, Oper Res Dept, Stanford Univ, Stanford
21. Richter A, Brandeau ML, Owens DK (1999) An analysis of optimal resource allocation for HIV prevention in injection drug users and nonusers. Medical Decision Making 19:167–179

22. Richter A, Brandeau ML, Zaric GS (2000) Optimal resource allocation for epidemic control in multiple independent populations. Techn Report, Dept Industr Engin and Engin Management, Stanford Univ, Stanford
23. Sethi SP (1978) Optimal quarantine programmes for controlling an epidemic spread. J Oper Res Soc 29:265–268
24. Sethi SP, Staats PW (1978) Optimal control of some simple deterministic epidemic models. J Oper Res Soc 29:129–136
25. Tan WY, Yakowitz S (1996) Machine learning for Markov decision processes with application to an AIDS allocation problem. Techn Report, Systems and Industr Engin Dept, Univ Arizona
26. Varian HR (1978) Microeconomic analysis. W Norton, New York
27. Wickwire K (1977) Mathematical models for the control of pests and infectious diseases: A survey. Theoret Population Biol 11:182–238
28. Zaric GS (2000) Resource allocation for epidemic control over short time horizons. PhD Thesis, Dept Industr Engin and Engin Management, Stanford Univ, Stanford

Reverse Convex Optimization

Reverse convex programming

STEPHEN E. JACOBSEN
University California, Los Angeles, USA

MSC2000: 90C26, 90C30

Article Outline

Keywords
Definitions
Examples
Some Basic Concepts for Solution Methods
  X a Convex Polytope, *f* Linear
  X Convex, F Convex
Test Problem Construction
See also
References

Keywords

Reverse convex programming; Nonconvex programming; Global optimization

The purpose of this article is to introduce the complex reverse convex programming problem. As such, this article is not a complete survey of the research into solution methods for *reverse convex programs*. This article will provide several examples that demonstrate the importance or relevance of the problem class, and will introduce the reader to some typical solution strategies that illustrate the combinatorial nature of the problem. Much of the early work in reverse convex programming deals with finding local minima [1,2,15,17]. The discussion of this article is directed toward the *global optimization* of reverse convex programs.

Definitions

Let X and C denote two convex subsets of $\mathbf{R}^n$, where X is closed. Let $G = \mathrm{cl}(C^c)$; that is, G is the closure of the complement of a convex subset of $\mathbf{R}^n$. We call such a set a *reverse convex set*. The corresponding reverse convex feasible region is defined as

$$F = X \cap G. \tag{1}$$

Let $f\colon O \to \mathbf{R}^1\}$, where O is an open subset of $\mathbf{R}^n$ such that $O \supset F$. The general reverse convex programming problem (RCP) is defined as

$$\min\{f(x)\colon\ x \in X \cap G\}\,. \tag{2}$$

Clearly, F is generally a nonconvex set and, moreover, F is often disconnected. As a result, (2) is a member of a class of difficult optimization problems, whether or not f is a convex function.

Examples

The first two examples below are well-known difficult examples in their own right; each can be rewritten as an equivalent reverse convex optimization problem. There is no computational significance of this conversion; however, the conversion is included so the reader may better understand the general complexity of the reverse convex class of optimization problems.

Example 1 Let

$$XI = \left\{ x \in \mathbb{R}^n\colon \quad \begin{array}{c} Ax \geq b, x_j \in \{0,1\}, \\ j = 1,\ldots,n \end{array} \right\},$$

where A is $(m \times n)$-matrix and b is an m-vector. Then

$$\min\left\{c^\top x\colon\ x \in XI\right\}. \tag{3}$$

is the well-known 0–1 linear integer programming problem. Define the two sets $X = \{x \in \mathbf{R}^n\colon Ax \geq b,\ 0$

$\le x_j \le 1, j = 1, \ldots, n\}$ and $G = \{x \in \mathbf{R}^n: \sum_j x_j(1 - x_j) \le 0\}$. We see that G is a reverse convex set and that (3) is equivalent to the reverse convex optimization problem

$$\min \left\{c^\top x: \ x \in X \cap G\right\}. \tag{4}$$

Example 2 Let $X_A = \{x \in \mathbf{R}^n: Ax \ge b, x \ge 0\}$ and let f be a concave function on $\mathbf{R}^n$. The problem

$$\min \{f(x): \ x \in X_A\} \tag{5}$$

is the well-known, and difficult, concave minimization problem (f is continuous since we assume it is defined on $\mathbf{R}^n$). Introduce an additional variable, η, and the inequality constraint $\eta - f(x) \ge 0$. Let $X = \{(x, \eta) \in \mathbf{R}^{n+1}: Ax + \eta 0 \ge b, x \ge 0\}$, where 0 is a column m-vector of zeros and let $G = \{(x, \eta): \eta - f(x) \ge 0\}$, a reverse convex set. Then (5) is equivalent to the reverse convex optimization problem

$$\min_{x,\eta} \{\eta: \ (x, \eta) \in X \cap G\}. \tag{6}$$

Example 3 Let E denote the node-arc incidence matrix of a connected, directed graph (assume there are m nodes and n arcs). Let $f_{i,j}$ denote the nonnegative flow on arc (i, j) and let $k_{i,j} \ge 0$ denote the capacity of arc (i, j). Assume the capacity of arc (i, j) can be increased by an amount $x_{i,j} \ge 0$ at a corresponding cost of $c_{i,j}(x_{i,j}) \ge 0$. Also assume, for each arc (i, j), we have $c_{i,j}(x_{i,j}) < c_{i,j}(\overline{x}_{i,j})$ if $x_{i,j} < \overline{x}_{i,j}$, $c_{i,j}(x_{i,j}) \to \infty$, as $x_{i,j} \to \infty$, and $c_{i,j}$ is continuous. Let $B > 0$ denote the capacity expansion budget; it is desired to increase the flow capacity (from the source to the sink, respectively represented by the first and last rows of E) as much as possible. We assume the network at hand has the property that there are *economies of scale* present. That is, the average cost of capacity expansion, for each arc, is a decreasing function. That is, each incremental capacity cost function, $c_{i,j}$, is concave. Let v denote the value of a feasible flow vector f. Let $q = (-1, 0, \ldots, 0, 1)'$. The optimization problem may be written as

$$\begin{cases} \max\limits_{f,x,v} & v \\ \text{s.t.} & \mathsf{E}f - vq = 0 \\ & f - x \le k \\ & \sum\limits_{i,j} c_{i,j}(x_{i,j}) \le B \\ & f \ge 0, \quad x \ge 0. \end{cases} \tag{7}$$

Since each $c_{i,j}$ is concave and continuous, the set $G = \{x \ge 0: \sum_{i,j} c_{i,j}(x_{i,j}) \le B\}$ is a reverse convex set.

Example 4 Consider the following problem, an example of the linear bilevel optimization problem [5],

$$\begin{cases} \min\limits_{x,y} & c^\top x + d^\top y \\ \text{s.t.} & A_1 x + D_1 y \ge b^1, \end{cases} \tag{8}$$

where y solves

$$\begin{cases} \min\limits_{z} & f^\top z \\ \text{s.t.} & A_2 x + D_2 z \ge b^2. \end{cases}$$

The interpretation of this problem is that a 'king' decides upon a vector pair, (x^*, y^*). He imposes his choice of x^* upon his subjects; however, this benevolent king will allow his subjects to choose the vector y (since the king is optimistic and, he assumes, omniscient as well, he assumes his subjects' optimal choice of y, when faced with x^*, will be the same as his choice, y^*). Let

$$\sigma(b^2 - A_2 x) = \begin{cases} \min\limits_{z} & f^\top z \\ \text{s.t.} & A_2 x + D_2 z \ge b^2, \end{cases} \tag{9}$$

denote the so-called optimal value function for the king's subjects, as a function of the king's choice of x. The king's problem (8) is equivalent to

$$\begin{cases} \min\limits_{x,y} & c^\top x + d^\top y \\ \text{s.t.} & A_1 x + D_1 y \ge b^1 \\ & A_2 x + D_2 y \ge b^2 \\ & \sigma(b^2 - A_2 x) - f^\top y \ge 0. \end{cases} \tag{10}$$

Since σ is a convex polyhedral function of x, we see that the last constraint of (10) is a reverse convex constraint. That is, the set $G = \{(x, y): \sigma(b^2 - A_2 x) - f^\top y \ge 0\}$ is a reverse convex set. Of course this problem is quite difficult in that, unlike the previous examples, the reverse convex constraint is not explicitly known.

Example 5 Consider the reverse convex optimization problem with several reverse convex constraints:

$$\min \{f(x): \ x \in X, \ g_i(x) \ge 0, \ i = 1, \ldots, m\} \tag{11}$$

where each g_i is a convex function. By the addition of a variable, (11) can be converted to a reverse convex optimization problem with one reverse convex constraint [21]. Let

$$g(x) = \min \{g_i(x): \ i = 1, \ldots, m\} \tag{12}$$

and let

$$p(x) = \sum_i g_i(x),$$
$$q(x) = \max\left\{\sum_{j\neq i} g_j(x):\ i = 1,\ldots,m\right\}. \tag{13}$$

Then $0 \leq g(x) = p(x) - q(x)$ is equivalent to the two constraints, in (x, t),

$$\begin{cases} q(x) - t \leq 0, \\ p(x) - t \geq 0, \end{cases} \tag{14}$$

where, of course, the last constraint is a reverse convex constraint. See U. Ueing [22] for an early branch and bound approach to the global optimization of (11).

Example 6 Consider the separable optimization problem

$$\begin{cases} \min\limits_{\substack{x\in X\\ y\in Y}} & f_1(x) + f_2(y) \\ \text{s.t.} & g^1(x) + g^2(y) \geq 0, \end{cases} \tag{15}$$

where X, Y are convex sets of, perhaps, different dimensions; f_1 is convex, f_2 is concave, g^1 is a vector of concave functions, and g^2 is a vector of convex functions. Assume for each $y \in Y$, that the problem

$$(\mathrm{P}_y) \quad \begin{cases} \min\limits_{x\in X} & f_1(x) \\ \text{s.t.} & g^1(x) \geq -g^2(y) \end{cases}$$

has a saddle point. Let y^i, $i = 1, \ldots, k$, be points of Y and let (x^i, λ^i) denote a saddle point for (P_{y^i}), $i = 1, \ldots, k$. The kth relaxed master of Benders' decomposition procedure [6,8] is

$$\begin{cases} \min\limits_{\substack{y\in Y\\ \eta}} & \eta \\ \text{s.t.} & \eta - \left(f_2(y) - \lambda^{i\top} g^2(y)\right) \\ & \quad \geq f_1(x^i) - \lambda^{i\top} g^1(x^i), \\ & \quad i = 1, \ldots, k. \end{cases} \tag{16}$$

This is a problem with several reverse convex constraints. Note that a simple change of variable, $\gamma = \eta - f_2(y)$, leads to the equivalent problem

$$\begin{cases} \min\limits_{\substack{y\in Y\\ \gamma}} & \gamma + f_2(y) \\ \text{s.t.} & \gamma + \lambda^{i\top} g^2(y) \\ & \quad \geq f_1(x^i) - \lambda^{i\top} g^1(x^i), \\ & \quad i = 1, \ldots, k. \end{cases} \tag{17}$$

This problem involves the minimization of a concave function subject to several reverse convex constraints, a very difficult formulation of the relaxed master.

Some Basic Concepts for Solution Methods

Let f: $\mathbf{R}^n \to \mathbf{R}^1$ and g: $\mathbf{R}^n \to \mathbf{R}^1$ be convex functions and let $G = \{ x \in \mathbf{R}^n : g(x) \geq 0\}$, a reverse convex set. Also assume

$$\exists w \in X \cap G^c \ni f(w) < f^*, \tag{18}$$

where f^* is the optimal value of the reverse convex problem (2). Then any optimal solution $\overline{x}$ has the property that $g(\overline{x}) = 0$; it then follows that

$$0 = \max\{g(x):\ x \in X,\ f(x) \leq f(\overline{x})\} \tag{19}$$

and, therefore, $\overline{x}$ is optimal for (19). Under certain additional 'stability' [21] conditions, it is also the case that if $\overline{x}$ satisfies (19) then $\overline{x}$ is optimal for (2).

X a Convex Polytope, f Linear

In order to develop an understanding of the combinatorial nature of the problem, it is useful to consider the following. If, in addition to the above assumptions, X is a convex polytope then the feasible region (1) is also a convex polytope (see [11,12]). This, in turn, implies that there is an optimal solution for (2) on an edge of the feasible region (1) if f is a linear function (or, of course, if f is a concave function). In this case we refer to the reverse convex programming problem as *LRCP* to denote that we are dealing with *linear programs with an additional reverse convex constraint*. Assume a vertex, $\overline{x}$ of X, can be found with the property that $g(\overline{x}) \geq 0$. Then pivot via the simplex algorithm, so as to decrease $f(x) = c^\mathsf{T} x$, until a neighboring pair of vertices, u and v, is found with the property $g(u) \geq 0$ and $g(v) < 0$. Let $z \in [u, v]$ be the last point on the edge $[u, v]$ with the property $g(z) = 0$. Consider the convex polytope $X \cap \{x: c^\mathsf{T} x = c^\mathsf{T} z\}$ and generate the $n - 1$ neighbors (we assume nondegeneracy for this discussion) of z on the hyperplane $\{x: c^\mathsf{T} x = c^\mathsf{T} z\}$. If one of those neighbors, say η, is such that $g(\eta) \geq 0$ then we consider the edge, $[\eta_1, \eta_2]$ of X, that contains η and continuing pivoting, via the simplex algorithm, from that edge's endpoint which is feasible. On the other hand, if no neighbor of z is feasible, we solve (19) with $\overline{x} = z$. If $\overline{x}$ is

optimal for (19) then, under 'stability', $\overline{x}$ is optimal for (2). If $\overline{x}$ is not optimal for (19), let x' be optimal; then x' is a vertex of X (if $c^\top x' < c^\top \overline{x}$) or x' is on an edge of X (if $c^\top x' = c^\top \overline{x}$) and, of course, $g(x') > 0$. Note that we have made an implicit assumption that (19) is an easier problem to solve than the original problem (2). There is no theoretical justification for this assumption; often however, in practice, the assumption appears to hold. Also, all that is needed is an x' with the aforementioned properties; an optimal x' is not necessarily required.

See [3,4] for an early usage of the concave minimization problem (19) in the context of the application of Benders' decomposition [6,8] to an economies-of-scale network capacity expansion problem of the form of (7). Also see [11] for a full discussion of the above pivoting method.

Since an edge optimal solution exists, a branch and bound edge search strategy may be in order. The first is due to R.J. Hillestad [10]; also, see [14]. We will not discuss this approach.

There is also a large literature, employing cutting planes based on the seminal paper of H. Tuy [20], which discusses the usefulness and difficulties of cutting planes for reverse convex optimization (e. g., see [7,9,12,13,18]). This literature will not be discussed in this article.

X Convex, F Convex

Under assumption (18), and following Tuy [21], define the mapping $\pi: X \cap G \to \mathbf{R}^n$ by

$$\alpha = \min\{\delta \in [0,1]:\ g(x + \delta(w - x)) = 0\}\ ,$$
$$\pi(x) = x + \alpha(w - x)\ . \tag{20}$$

If z solves (19) but $g(z) > 0$, then $f(\pi(z)) < f(\overline{x})$. This suggests the following:

Select	$\overline{x} \in X \cap \partial G (\partial G = \{x : g(x) = 0\})$
Repeat	Let z solve (19)
	If $g(z) > 0, \overline{x} \leftarrow \pi(z)$
Until	$\{g(z) = 0\}$

Under a few technical assumptions, this method will, in the limit, produce an optimal solution. However, the major step in the algorithm involves a convex maximization problem over a convex set and, in some sense, illustrates the relationship between convex maximization (concave minimization) and reverse convex programs. Of course, the convex maximization problem is often as difficult to solve as the original reverse convex program (2) and, therefore, solving the latter by a sequence of convex maximization problems may not be effective. However, there are excellent concave minimization methods (cf. e. g., ► Concave Programming).

There are various algorithmic strategies that attempt to overcome the inherent difficulties for solving (19) directly. Let δ_x denote a support for g at the point x. Let $\overline{x} \in X \cap G$ and let z solve the convex optimization problem, with a linear objective,

$$\max\left\{\delta_{\overline{x}}^\top (x - \overline{x}):\ x \in X,\ f(x) \le f(\overline{x})\right\}. \tag{21}$$

If $\delta_{\overline{x}}(z - \overline{x}) > 0$, then $g(z) > 0$ and $f(\pi(z)) < f(\overline{x})$. To see the latter, observe

$$\begin{aligned} f(\pi(z)) &\le f(z) + \alpha(f(w) - f(z)) \\ &< f(z) + \alpha(f(w) - f^*) < f(z) \le f(\overline{x})\ , \end{aligned}$$

where the strict inequalities hold since (18) implies that $\alpha > 0$ and that z cannot be optimal for (2). This suggests the following:

Select	$\overline{x} \in X \cap G$
DO	Repeat
	Let z solve (21)
	If $\delta_{\overline{x}}^\top (z - \overline{x}) > 0, \overline{x} \leftarrow \pi(z)$
	Until $\left\{\delta_{\overline{x}}^\top (z - \overline{x}) = 0\right\}$
	$\overline{x} \leftarrow \pi(z)$
	let z solve (19)
	If $g(z) > 0, \overline{x} \leftarrow \pi(z)$
WHILE	$\{g(z) > 0\}$

Of course, the point of solving (21) is to avoid, as long as possible, addressing problem (19). Nevertheless, (19) must be dealt with eventually. Another approach is to use an outer approximation method for problem (19). For instance, see Tuy [21] for a method which generates a sequence of convex polytopes, $\{S_k\}$, with the properties $S_k \subset S_{k-1}$ and each $S_k \supset X \cap G$. Other methods, and combinations of methods, have been developed for problem (2) and the reader is referred to [13] as well as to the 'Journal of Global Optimization'.

Test Problem Construction

At this point in time, as one might imagine, there are no generally applicable and efficient methods for solving (2), the general RCP. Therefore, in order to test new procedures, it is important to have at hand a method for generating problems for which we know the optimal vector. There are several such methods; we will develop one, the first, due to Y. Sung and J.B. Rosen [19] (also, see [16]), for constructing a concave minimization problem, over a convex polytope, whose answer is known. A slight modification of that method leads to the construction of an LRCP whose answer is known. We proceed as follows.

Let $X = \{x \in \mathbf{R}^n : Ax \leq b\}$, where A is $m \times n$, $m > n$ (e. g., nonnegativity constraints are included), and it is assumed that X is bounded. Let $f(x) = c^{\intercal}x$ and let $\overline{x}$ be any edge point of X that is not a vertex of X. Then $\overline{x}$ is a vertex of $X \cap \{x : c^{\top}x \leq c^{\top}\overline{x}\}$. Without loss of generality, assume the first $n-1$ rows of (A, b) define the edge of X of which $\overline{x}$ is an element. That is, $\overline{x}$ is the unique solution of the system of equations

$$\begin{aligned} &a_i^{\top}x = b_i, \quad i = 1, \ldots, n-1, \\ &c^{\top}x = c^{\top}\bar{x} \end{aligned} \tag{22}$$

(we assume $c^{\intercal}$ is not a linear combination of the rows $a_i^{\top}$, $i = 1, \ldots, n-1$). Let $Dx = d$ denote the matrix representation of (22). Let

$$\begin{aligned} v_i &= \min\left\{a_i^{\top}x : Ax \leq b, c^{\top}x \leq c^{\top}\overline{x}\right\}, \\ &i = 1, \ldots, n-1, \\ v_n &= \min\left\{c^{\top}x : Ax \leq b\right\} \end{aligned} \tag{23}$$

and, for arbitrarily small $\varepsilon > 0$, define $v^{\intercal}(\varepsilon) = (v_1 - \varepsilon, \ldots, v_n - \varepsilon)$. Let the jth component of the vector r be defined by $r_j = (d_j + v_j(\varepsilon))/2$. Then $\overline{x}$ is the unique optimal solution for the LRCP

$$\min\left\{c^{\top}x : \begin{array}{c} Ax \leq b, \\ \|Dx - r\|^2 - \|d - v(\varepsilon)\|^2/4 \geq 0 \end{array}\right\},$$

where $\|\cdot\|$ denotes the Euclidean norm.

See also

- ▶ αBB Algorithm
- ▶ D.C. Programming
- ▶ Quadratic Knapsack
- ▶ Quadratic Programming with Bound Constraints
- ▶ Standard Quadratic Optimization Problems: Theory

References

1. Avriel M, Williams AC (1970) Complementary geometric programming. SIAM J Appl Math 19:121–141
2. Avriel M, Williams AC (1971) An extension of geometric programming with applications in engineering optimization. J Eng Math 5:187–194
3. Bansal P, Jacobsen SE (1975) An algorithm for optimizing network flow capacity expansion under economies-of-scale. J Optim Th Appl 15:565–586
4. Bansal P, Jacobsen SE (1975) Characterization of basic solutions for a class of nonconvex programs. J Optim Th Appl 15:549–564
5. Bard JF (1983) An algorithm for solving the general bilevel programming problem. Math Oper Res 8:260–272
6. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Math 4:238–252
7. Fulop J (1988) A finite cutting plane method for solving linear programs with an additional reverse convex constraint. Working Paper, Dept Oper Res, Hungarian Acad Sci, Budapest
8. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 4:237–260
9. Gurlitz TR, Jacobsen SE (1991) On the use of cuts in reverse convex programs. J Optim Th Appl 68:257–274
10. Hillestad RJ (1975) Optimization problems subject to a budget constraint with economies-of-scale. J Oper Res 6:1091–1098
11. Hillestad RJ, Jacobsen SE (1980) Linear programs with an additional reverse convex constraint. J Appl Math Optim 6:257–269
12. Hillestad RJ, Jacobsen SE (1980) Reverse convex programming. J Appl Math Optim 6:63–78
13. Horst R, Tuy H (1990) Global optimization: deterministic approaches. Springer, Berlin
14. Jacobsen SE, Moshirvaziri K (1996) Computational experience using an edge search algorithm for linear reverse convex programs. J Global Optim 9:153–167
15. Meyer R (1970) The validity of a family of optimization methods. SIAM J Control 8:41–54
16. Moshirvaziri K (1994) A generalization of the construction of test problems for nonconvex optimization. J Global Optim 5:21–34
17. Rosen JB (1966) Iterative solution of nonlinear optimal control problems. SIAM J Control 4:223–244
18. Sen S, Sherali HD (1987) Nondifferentiable reverse convex programs and facetial convexity cuts via a disjunctive characterization. Math Program 37:169–183
19. Sung Y, Rosen JB (1982) Global minimum test problem construction. Math Program 24:353–355
20. Tuy H (1964) Concave programming under linear constraints. Soviet Math 5:1437–1440
21. Tuy H (1987) Convex programs with an additional reverse convex constraint. J Optim Th Appl 52:463–485

22. Ueing U (1972) A combinatorial method to compute a global solution of certain non-convex optimization problems. In: Lootsma FA (ed) Global Optimization: Deterministic Approaches. Acad Press, New York, pp 223–230

Robust Control

Stability margin

MICHAEL G. SAFONOV
Department Electrical Engineering–Systems, University Southern California, Los Angeles, USA

MSC2000: 93D09

Article Outline

Keywords
Canonical Robust Control Problem
Linear Time-Invariant Robustness
Topological Separation, Sectors and IQCs
Restrictions on Q
Linear Matrix Inequalities
Uncertainty Structure
State-Space Robustness Analysis
History and Further Reading
See also
References

Keywords

Automation; Control engineering; Feedback; Linear matrix inequality; Semidefinite programming; Uncertain systems

Control engineers are interested in the amount of modeling uncertainty that can be tolerated in feedback control systems. *Robust control theory* concerns the evaluation and optimization of uncertainty tolerance, otherwise known as *stability margin*. The purpose of robust control theory is to enable control engineers to determine quantitatively whether or not a feedback control design is capable of maintaining satisfactory performance for all perturbations within a given class. Beyond this, they may also seek to optimize control system robustness by choosing a feedback controller that maximizes uncertainty tolerance. In either case, the problem is essentially a *nonconvex optimization* problem.

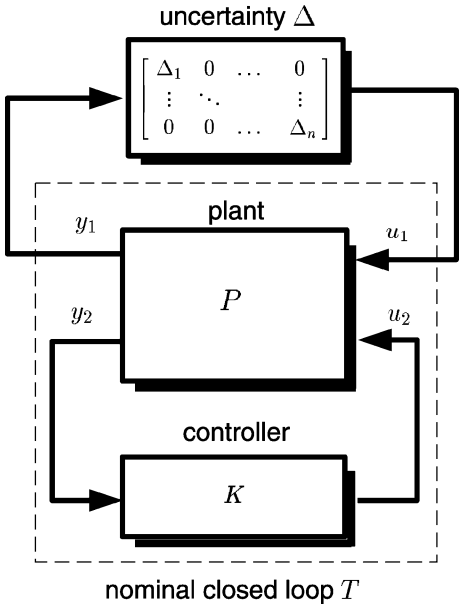
Canonical Robust Control Problem

By a suitable choice of variables, most robust control problems can be cast in the general framework of the canonical uncertain control system (S) depicted in Fig. 1 (cf. [25, p. 62]). Given the plant P(s) and a set Δ of uncertain feedback perturbations Δ, the canonical *robust control synthesis* problem is to find a controller $K(s)$ so that the closed-loop transfer function matrix remains stable for all block diagonal perturbation matrices

$$\Delta = \begin{pmatrix} \Delta_1 & 0 & \cdots & 0 \\ 0 & \Delta_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \Delta_n \end{pmatrix} \in \mathbf{\Delta}.$$

The Δ_i's are called *uncertainties* and the set Δ is called the *uncertainty set*. Via the introduction of an additional *fictitious uncertainty*, J.C. Doyle, J. Wall and G. Stein [10] observed that one may embed performance issues such a noise attenuation requirements within the robust stability framework; so, the focus on stability robustness is not overly restrictive.

When the controller $K(s)$ is given beforehand, then the resulting simplified problem is called the *robustness analysis* problem, also known as the multivariable sta-



Robust Control, Figure 1
Canonical uncertain control system (S)

bility margin analysis problem. On an abstract level, M.G. Safonov [25] showed the analysis problem to be equivalent to testing for the *topological separation* of the graphs of the operators T and Δ – this is so even in very general cases where T and Δ are nonlinear operators.

The *multivariable stability margin* $K_m(T)$ associated with a given controller K is [26,27]

$$K_m(T) \triangleq \begin{cases} \inf\limits_{\gamma \geq 0} & \gamma \\ \text{s.t.} & \text{system (S) is unstable} \\ & \Delta \in \gamma \mathbf{\Delta}. \end{cases}$$

Thus, the quantity K_m is the smallest nonnegative real number γ for which the 'scaled' set $\gamma\ \Delta$ contains a destabilizing Δ. In cases where there is no $\gamma \geq 0$ for which there is a destabilizing $\Delta \in \gamma\Delta$, one defines $K_m = \infty$. Clearly, a controller $K(s)$ solves the robust control synthesis problem if and only if the nominal closed-loop transfer function $T(s)$ satisfies

$$K_m(T) > 1.$$

It may be assumed without loss of generality that $\dim(u_1) = \dim(y_1)$.

Linear Time-Invariant Robustness

If, as is often assumed by control engineers, both Δ and T are linear time-invariant with stable rational Laplace transforms, then robustness analysis may be related to the *characteristic equation* of system (S)

$$\det(I - \Delta(s)T(s)) = 0. \tag{1}$$

The system (S) is said to be stable for a given $\Delta(s)$ if and only if (1) has no solution $\overline{\mathbb{C}}_+$ where $\overline{\mathbb{C}}_+$ denotes the closed right-half of the complex plane.

Suppose, additionally, that the set Δ is specified in the frequency-domain as simply the set of stable Δ's for which $\Delta(j\omega) \in \Delta(j\omega)$ where, for each ω, $\Delta(j\omega)$ is a given set of matrices. Then, the multivariable stability margin can be evaluated as

$$K_m = \inf_{\omega} k_m(j\omega)$$

where

$$k_m(T(j\omega)) \triangleq \begin{cases} \inf\limits_{\gamma \geq 0} & \gamma \\ \text{s.t.} & \det(I - \gamma\Delta(j\omega)T(j\omega)) = 0, \\ & \Delta(j\omega) \in \mathbf{\Delta}(j\omega). \end{cases} \tag{2}$$

Closely related to the multivariable stability margin $k_m(j\omega)$ is the *structured singular value* [9]

$$\mu(T(j\omega)) \triangleq \frac{1}{k_m(T(j\omega))}.$$

Thus, an 'optimal' solution to the robust control synthesis problem is obtained if one can solve the so-called *μ-synthesis control* problem

$$\min_{K(s)} \sup_{\omega} \mu(T(j\omega)).$$

The exact computation of $\mu(T(j\omega))$ is in general impractical, except for very simply structured uncertainty sets Δ. Indeed, the optimization problem (2) is in general *NP-hard*, meaning that it cannot be computed in polynomial time in the worst cases [6]. Still, easy to compute conservative upper-bounds on μ abound, but even for these it is usually necessary to first reformulate the robustness analysis problem.

Topological Separation, Sectors and IQCs

The robustness analysis problem (2) can be interpreted in terms of a 'topological separation' of the graphs of $\Delta(j\omega)$ and $T(j\omega)$ (e. g., [15,16,25]):

$$\text{graph}(\Delta) \triangleq \left\{ z = \begin{pmatrix} u_1 \\ y_1 \end{pmatrix} \in \text{range} \begin{pmatrix} \Delta \\ I \end{pmatrix} \right\} \tag{3}$$

$$\text{graph}(\gamma T) \triangleq \left\{ z = \begin{pmatrix} u_1 \\ y_1 \end{pmatrix} \in \text{range} \begin{pmatrix} I \\ \gamma T \end{pmatrix} \right\}. \tag{4}$$

In particular, the robustness condition $k_m(T) > 1$ is equivalent to the existence of a complex matrix Q such that the quadratic functional z^*Qz topologically separates graph(Δ) and graph(γT) in the sense that [15,25]

$$\text{Re}(z^*Qz) \geq 0 \quad \text{for all } z \in \text{graph}(\Delta), z \neq 0, \tag{5}$$

$$\text{Re}(z^*Qz) < 0 \quad \text{for all } z \in \text{graph}(\gamma T), z \neq 0. \tag{6}$$

Indeed, for $\epsilon > 0$ sufficiently small one such matrix is [16]

$$Q(j\omega) = \begin{pmatrix} \gamma T^*(j\omega) \\ -I \end{pmatrix} \begin{pmatrix} \gamma T(j\omega) & -I \end{pmatrix} - \epsilon I.$$

Safonov [25], building on the 1960s nonlinear stability work of I.W. Sandberg [32,33] and G. Zames [37], called this quadratic form of the topological separation condition the *sector stability criterion*. Later, A.

Megretski and A. Rantzer [19] have dubbed it the *integral quadratic constraint* (*IQC*) approach and linked it to the *S-procedure* nonlinear stability concept of V.A. Yakubovich.

In formulating robustness analysis as an optimization problem, the conditions (3)–(4) are usually rewritten as matrix definiteness conditions

$$\text{herm}\left(\begin{pmatrix}\Delta(j\omega)\\ I\end{pmatrix}^* Q(j\omega)\begin{pmatrix}\Delta(j\omega)\\ I\end{pmatrix}\right)\geq 0 \quad \text{for all } \Delta\in\mathbf{\Delta} \tag{7}$$

$$\text{herm}\left(\begin{pmatrix}I\\ \gamma T(j\omega)\end{pmatrix}^* Q(j\omega)\begin{pmatrix}I\\ \gamma T(j\omega)\end{pmatrix}\right)<0, \tag{8}$$

where

$$\text{herm}(X)\triangleq\frac{1}{2}(X+X^*).$$

Consequently, the problem (2) can be reformulated as the optimization $k_m(T(j\omega))\triangleq\inf_{\gamma\geq 0}\gamma$ subject to the LMI constraints (7)–(8). Unfortunately, verifying that there exists a Q satisfying (7)–(8) is not in general easier than the original problem. The problem remains inherently *NP*-hard [6].

As is apparent from (7)–(8), the problem in this form is an instance *linear matrix inequality* (*LMI*) problem, but with possibly infinitely many LMI constraints: (7) has one LMI for each element of the set Δ which in general may be infinite – for example, even the unit interval $\Delta=[0,1]\subset\mathbf{R}$ has infinitely many points.

Restrictions on Q

The class of matrices Q which may potentially solve the optimization (7)–(8) is inherently restricted. K.C. Goh and Safonov [15] established that every IQC topological separating functional is isomorphic to the *positivity* and *small gain* stability criteria of Zames [37]. More precisely, for some invertible matrix $\widetilde{F}$, the change of variables

$$\begin{pmatrix}\widetilde{u}_1\\ \widetilde{y}_1\end{pmatrix}\triangleq\widetilde{F}\begin{pmatrix}u_1\\ y_1\end{pmatrix}=\widetilde{F}z$$

causes the topological separation conditions (5)–(6) to simplify to the so-called 'small gain' form

$$\|\widetilde{y}_1\|^2-\|\widetilde{u}_1\|^2\geq 0 \text{ for all } z\in\text{graph}(\Delta),\ z\neq 0,$$
$$\|\widetilde{y}_1\|^2-\|\widetilde{u}_1\|^2<0 \text{ for all } z\in\text{graph}(\gamma T),\ z\neq 0,$$

which corresponds to $Q=\widetilde{F}^*\begin{pmatrix}-I & 0\\ 0 & I\end{pmatrix}\widetilde{F}$. Similarly, if one defines

$$\widehat{F}\triangleq\begin{pmatrix}-I & I\\ I & I\end{pmatrix}\widetilde{F},$$

then under the invertible change of variables

$$\begin{pmatrix}\widehat{u}_1\\ \widehat{y}_1\end{pmatrix}\triangleq\widehat{F}\begin{pmatrix}u_1\\ y_1\end{pmatrix}=\widehat{F}z$$

conditions (5)–(6) assume the 'positivity' form

$$\text{Re}(\widehat{y}_1^*\widehat{u})\geq 0 \quad \text{for all } z\in\text{graph}(\Delta),\ z\neq 0,$$
$$\text{Re}(\widehat{y}_1^*\widehat{u})<0 \quad \text{for all } z\in\text{graph}(\gamma T),\ z\neq 0,$$

which corresponds to $Q=\widehat{F}^*\begin{pmatrix}0 & 0\\ I & 0\end{pmatrix}\widehat{F}$. One implication of these results is that, without loss of generality, one may restrict the Q matrices to those for which $\text{rank}(Q)=\dim(Q)/2$ without introducing any conservativeness in solving the optimization. Another implication is that any Q that solves (7)–(8) must have a Hermitian part whose sign matrix has exactly as many 1's as -1's. But, despite these freedoms to further restrict Q as above, the problem of exact computation of k_m remains inherently *NP*-hard.

Linear Matrix Inequalities

Since exact computation of the multivariable stability margin is inherently *NP*-hard, one must settle for computing upper and/or lower bounds on k_m. In particular, 'conservative' lower-bounds are preferred by control engineers, since these give conservative sufficient conditions for stability of the uncertain system.

Many practical algorithms for computing a lower-bound $\underline{k}_m$ on the multivariable stability margin involve the solution of finite-dimensional linear matrix inequalities of the form

$$k_m(T(j\omega))\geq\underline{k}_m(T(j\omega))\triangleq\begin{cases}\inf\limits_{\gamma\geq 0} & \gamma\\ \text{s.t.} & \\ \text{herm} & \left(\begin{pmatrix}I\\ \gamma T(j\omega)\end{pmatrix}^* Q(M)\begin{pmatrix}I\\ \gamma T(j\omega)\end{pmatrix}\right)\\ & <0,\\ & \text{herm}(M(\theta))>0.\end{cases} \tag{9}$$

Casting the k_m lower-bounding problem in this framework typically involves assuming additional 'structural' characteristics for the uncertainty set Δ and exploiting these characteristics to identify linearly parametrized subsets of matrices $Q(\theta)$ and $M(\theta)$ for which (7) is known to hold a priori. Then, having eliminated the constraint (7), the problem assumes the form of the *semidefinite programming* problem (9), which is practical to solve. Indeed, for each fixed $\gamma \geq 0$ it is a convex *linear matrix inequality* (LMI) optimization problem. The key to success here is identify which classes of linearly parametrized matrices $Q(\theta)$ and $M(\theta)$ go with which sorts of uncertainty structure.

Uncertainty Structure

Several commonly encountered uncertainty structures (cf. [13]) are listed in the table below along with linearly parametrized $M(\theta)$ and $Q(M)$ such that

$$\begin{pmatrix}\Delta\\ I\end{pmatrix}^* Q(M)\begin{pmatrix}\Delta\\ I\end{pmatrix} \geq 0$$

$$\text{for all } \begin{cases}\Delta \in \mathbf{\Delta},\\ M(\theta) \text{ with } \operatorname{herm}(M(\theta)) > 0.\end{cases}$$

Uncertainty Δ	$M(\theta)$ and $Q(\theta)$
Small gain $\Delta \in \mathbf{C}^{m\times m}$ $\parallel \Delta \parallel \leq 1$,	$M = \theta I,\ \theta \in \mathbf{R}$, $Q = \theta\begin{pmatrix}-M & 0\\ 0 & M\end{pmatrix}$
Positive $\Delta \in \mathbf{C}^{m\times m}$ $\operatorname{herm}(\Delta) \geq 0$,	$M(\theta) = \theta I,\ \theta \in \mathbf{R}$, $Q = \begin{pmatrix}0 & 0\\ M & 0\end{pmatrix}$
Repeated small gain $\delta \in \mathbf{R}$ $\Delta = \delta I_{m\times m}, \mid \delta \mid \leq 1$,	$M(\theta) = \theta \in \mathbf{C}^{m\times m}$, $Q = \begin{pmatrix}-M & M\\ -M & M\end{pmatrix}$
Repeated positive $\delta \in \mathbf{R}$ $\Delta = \delta I_{m\times m},\ \delta \geq 0$,	$M(\theta) = \theta \in \mathbf{C}^{m\times m}$, $Q = \begin{pmatrix}0 & 0\\ M & 0\end{pmatrix}$
Repeated positive $\delta \in \mathbf{C}$ $\Delta = \delta I_{m\times m}$, $\operatorname{Re}(\delta) \geq 0$,	$M(\theta) = \theta \in \mathbf{C}^{m\times m}$, $M(\theta) = M^*(\theta)$, $Q = \begin{pmatrix}0 & 0\\ M & 0\end{pmatrix}$

When it is known that each of the individual uncertainties Δ_i satisfies such a condition for a Q_i ($i = 1, \ldots, m$) partitioned evenly as

$$Q_i \triangleq \begin{pmatrix}Q_i^{(11)} & Q_i^{(12)}\\ Q_i^{(21)} & Q_i^{(22)}\end{pmatrix},$$

then it follows that inequality (7) holds for a likewise partitioned Q having (for $j, k = 1, 2$)

$$Q^{(jk)} = \begin{pmatrix}Q_1^{(jk)}(M_1) & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & Q_n^{(jk)}(M_n)\end{pmatrix}.$$

State-Space Robustness Analysis

Insofar as stability analysis is concerned, the 'state-integrator' operator $(1/s)I$ may be regarded as a repeated positive uncertainty $\delta I, \delta \in \overline{\mathbb{C}}_+$, [34]. This leads to a rearrangement of the block diagram of the system (S) so that the state-integrator is relocated from inside the nominal plant $T(s) = C(Is - A)^{-1}B + D$ into the the Δ block as an additional *fictitious uncertainty*. Thus, Δ and T become

$$\Delta \leftarrow \begin{pmatrix}\frac{1}{s}I & 0 & \cdots & 0\\ 0 & \Delta_1 & \cdots & 0\\ \vdots & & \ddots & \vdots\\ 0 & 0 & \cdots & \Delta_n\end{pmatrix}, \quad T \leftarrow \begin{pmatrix}A & B\\ C & D\end{pmatrix},$$

whence T becomes a constant matrix.

History and Further Reading

The origins of robust control theory can be traced to 1970s research work headed by M. Athans at the MIT Electronic Systems Laboratory. Up to this time, most mathematical control theorists generally assumed that untried 1960s optimal control theories would produce superior feedback designs. In 1971, researchers Athans [1] and H.H. Rosenbrock [24] began to voice cautious doubts about the this popular view. By 1975 the worst expectations had been confirmed when early attempts to apply *linear-quadratic Gaussian* (LQG) optimal control theory to practical feedback designs produced unacceptable results, see [2,21,30]. The problem was quickly identified to be a lack of attention to the *multivariable stability margin* problem [36]. Athans

and Safonov [25,29] introduced the use of the term *robustness* to describe this problem and laid the foundation for its solution based on quadratic separating functionals. The idea of computing lower-bounds on k_m via LMI optimization is due to M.K.H. Fan and A.L. Tits [12].

The evolution robust control theory during its infancy and up to 1982 is accurately described in [30]. Bibliographies focusing on more recent developments in the theory of multivariable stability margin may be found in references [8,22]. The article [27] provides an elementary introduction to robust control theory including motivating examples and a sampling of nonoptimization techniques for estimating conservative bounds on multivariable stability margin. Books on robust control design methods include J.M. Maciejowski [18], S. Skogestad and I. Postlethwaite [35], and B.R. Barmish [4]. Linear matrix inequality (LMI) optimization methods in control theory are emphasized in books of S.P. Boyd, L. ElGhaoui, E. Feron and V. Balakrishnan [5] and of ElGhaoui and S. Niculescu [11]. References [17,17,23] describe classes of optimal H_∞ robust control synthesis problems can be reduced to LMI optimizations using embeddings based on *Parrott's theorem* and *Finsler's theorem*. Bilinear matrix inequality (BMI) formulations of the robust control synthesis problems are described in [20,31]. Commercial software packages for robust control include [3,7,14].

See also

- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Optimal Control of a Flexible Arm
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Athans M (1971) Editorial on the LQG problem. IEEE Trans Autom Control AC-16(6):528
2. Athans M, Castanon D, Dunn K-P, Greene CS, Lee WH, Sandell NR, Willsky AS (1977) The stochastic control of the F-8C aircraft using a multiple model adaptive control (MMAC) method. Part I: equilibrium flight. IEEE Trans Autom Control AC-22(5):768–7791
3. Balas GJ, Doyle JC, Glover K, Packard A, Smith R (1991) μ-Analysis and synthesis toolbox (μ-tools). MathWorks, Natick
4. Barmish BR (1994) New tools for robustness in linear systems. MacMillan, New York
5. Boyd SP, El Ghaoui L, Feron E, Balakrishnan V (1994) Linear matrix inequalities in systems and control theory. SIAM, Philadelphia
6. Braatz RP, Young PM, Doyle JC, Morari M (1994) Computational complexity of μ calculation. IEEE Trans Autom Control 39(5):1000–1002
7. Chiang RY, Safonov MG (1988) Robust control toolbox
8. Dorato P, Tempo R, Muscato G (1993) Bibliography on robust control. Automatica, 29(1):201–213
9. Doyle JC (1982) Structured uncertainty in control system design. IEEE Proc 129-D(6):242–250
10. Doyle JC, Wall J, Stein G (1982) Performance and robustness analysis for structured uncertainty. In: Proc IEEE Conf Decision and Control (December 8–10, 1982, Orlando), IEEE Computer Soc Press, New York, pp 629–636
11. El Ghaoui L, Niculescu S (eds) (1998) Recent advances in LMI theory for control. SIAM, Philadelphia
12. Fan MKH, Tits AL (1988) m-form numerical range and the computation of the structured singular value. IEEE Trans Autom Control AC-33(3):284–289
13. Fan MKH, Tits AL, Doyle JC (Jan. 1991) Robustness in the presence of mixed parametric uncertainty and unmodeled dynamics. IEEE Trans Autom Control AC-36(1):25–38
14. Gahinet P, Nemirovskii A, Laub A, Chilali M (1995) LMI control toolbox. MathWorks, Natick
15. Goh KC, Safonov MG (1995) Robustness analysis, sectors and quadratic functionals. In: Proc IEEE Conf on Decision and Control (December 13–15, 1995), IEEE Computer Soc Press, New York, pp 1988–1993
16. Iwasaki T (1997) Robust stability analysis with quadratic separator: Parametric time-varying uncertainty case. In: Proc IEEE Conf on Decision and Control 5, IEEE Computer Soc Press, New York, pp 4880–4885

17. Iwasaki T, Skelton RE (1994) All controllers for the general H∞ control problem: LMI existence conditions and state space formulas. Automatica 30(8):1307–1317
18. Maciejowski JM (1989) Multivariable feedback design. Addison-Wesley, Reading
19. Megretski A, Rantzer A (1997) System analysis via integral quadratic constraints. IEEE Trans Autom Control AC-42(6):819–830
20. Mesbahi M, Safonov MG, Papavassilopoulos GP (2000) Bilinearity and complementarity in robust control. In: El Ghaoui L, Niculescu S (eds) Recent Advances in LMI Theory for Control. SIAM, Philadelphia, 269–292
21. Recent developments in the robustness theory of multivariable systems. In: Sandell NR (ed) (1979) Proc MIT/ONR Workshop (April 25–27) at MIT, Lab Information and Decision Systems, Cambridge
22. Packard A, Doyle J (1993) The complex structured singular value. Automatica 29(1):71–109
23. Packard A, Zhou K, Pandey P, Becker G (1991) A collection of robust control problems leading to LMI's. In: Proc IEEE Conf on Decision and Control 2 (December 11–13, 1991), IEEE Computer Soc Press, New York, pp 1245–1250
24. Rosenbrock HH, McMorran PD (1971) Good, bad or optimal? IEEE Trans Autom Control AC-16(6):552–529
25. Safonov MG (1980) Stability and robustness of multivariable feedback systems. MIT, Cambridge. Based on author's PhD Thesis (1977) "Robustness and Stability Aspects of Stochastic Multivariable Feedback System Design, MIT, Cambridge
26. Safonov MG (1980) Stability margins of diagonally perturbed multivariable feedback-systems. Techn Report, Honeywell Systems and Res Center, MR12588
27. Safonov MG (1981) Stability margins of diagonally perturbed multivariable feedback systems. In: Proc IEEE Conf on Decision and Control (December 16–18, 1981), IEEE Computer Soc Press, New York
28. Safonov MG (1999) Robust control. In: Webster JG (ed) Encycl Electrical and Electronics Engin, vol 18. Wiley, New York pp 592–602
29. Safonov MG, Athans M (1976) Gain and phase margin for multiloop LQG regulators. In: Proc IEEE Conf on Decision and Control (December 1–3, 1976), IEEE Computer Soc Press, New York, pp 361–368
30. Safonov MG, Fan MKH (1997) Editorial, special issue on multivariable stability margin. Internat J Robust and Nonlinear Control 7(2):97–103
31. Safonov MG, Papavassilopoulos G (1994) The diameter of an intersection of ellipsoids and BMI robust synthesis. In: Proc IFAC Symposium on Robust Control Design. IFAC, Laxenberg, pp 313–317
32. Sandberg IW (1964) A frequency-domain condition for the stability of feedback systems containing a single time-varying nonlinear element. Bell System Techn J 43(4):1601–1608
33. Sandberg IW (1964) On the L2-boundedness of solutions of nonlinear functional equations. Bell System Techn J 43(4):1581–1599
34. Sideris A (1992) Elimination of frequency search from robustness tests. IEEE Trans Autom Control 37(10):1635–1640
35. Skogestad S, Postlethwaite I (1996) Multivariable feedback control. Wiley, New York
36. Wong PK (1975) On the interaction structure of linear multi-input feedback control systems. PhD Thesis, Electrical Engin, MIT
37. Zames G (1966) On the input–output stability of time-varying nonlinear feedback systems – Parts I and II. IEEE Trans Autom Control AC-15(2–3):228–238; 465–467

Robust Control: Schur Stability of Polytopes of Polynomials

JUERGEN ACKERMANN
DLR Oberpfaffenhofen, Wessling, Germany

MSC2000: 93D09, 93C55, 39A11

Article Outline

Keywords
Solution
Summary
See also
References

Keywords

Robust stability; Discrete-time systems

Consider the robust stability analysis of linear, time-invariant, discrete-time control systems with uncertain real parameters q_i, $i = 1, \ldots, \ell$, with given lower and upper bounds $q_i^- \le q_i \le q_i^+$. The scalars q_i form a vector $\mathbf{q} = [q_1, \ldots, q_\ell]^\mathsf{T}$ that is bounded by a hyperrectangle $Q = \{\mathbf{q}: q_i^- \le q_i \le q_i^+,\ i = 1, \ldots, \ell\}$. The characteristic polynomial is

$$P(z, \mathbf{q}) = \sum_{k=0}^{n} a_k(\mathbf{q}) z^k, \quad \mathbf{q} \in Q.$$

For simplicity we assume $a_n(\mathbf{q}) > 0$ for all $\mathbf{q} \in Q$. Suppose the coefficients $a_k(q_1, \ldots, q_\ell)$ depend (affine) linearly on the uncertain parameters, i. e.

$$a_k(\mathbf{q}) = a_{k0} + a_{k1} q_1 + \cdots + a_{k\ell} q_\ell.$$

Then $P(z, Q) = \{P(z, \mathbf{q}): \mathbf{q} \in Q\}$ is a *polytope of polynomials*. The question of robust *Schur stability* of the polytope is now: Are all roots of $P(z, Q)$ located inside the unit circle of the complex z-plane?

Solution

Define the vertex polynomials $P_i(z)$, $i = 1, \ldots, 2^\ell$, for which all q_i take on their maximum or minimum value. An obvious necessary condition for robust Schur stability is that all vertex polynomials P_i are Schur stable.

The following derivation of a necessary and sufficient condition was given in [1]. In the first step the edge theorem [2] is used. For the present problem is says, that the polytope of polynomials is Schur stable if and only if all exposed edges are Schur stable. The notion of 'exposed' edge is illustrated by the case $\ell = 2$. The admissible region is a rectangle in the (q_1, q_2)-plane. The four edges of the rectangle are exposed, the two diagonals are, however, not exposed. In the ℓ-dimensional hyperrectangle each vertex is connected to ℓ neighboring vertices by exposed edges. Counting each edge only once there are $\ell\, 2^\ell - 1\}$ edges. The edge theorem reduces the test of an ℓ-dimensional continuum to a finite number of one-dimensional stability tests for the edges.

The second step is now the Schur stability test for an edge between vertices $\mathbf{q}_B$ and $\mathbf{q}_C$ with corresponding polynomials $P_B(z)$ and $P_C(z)$. A point on the edge is described by

$$P_A(z, \alpha) = \alpha P_B(z) + (1 - \alpha) P_C(z), \alpha \in (0, 1).$$

Starting from a stable polynomial $P_A(z, 0) = P_C(z)$ there are three possibilities how $P_A(z, \alpha)$ can become unstable with increasing α:

i) a real root crosses the point $z = +1$;
ii) a real root crosses the point $z = -1$; and
iii) a complex conjugate pair of roots crosses the unit circle.

By stability of the vertices $P_B(1) > 0$, $P_C(1) > 0$ and therefore $P_A(1, \alpha) > 0$ for $\alpha \in (0, 1)$. A similar argument holds for $P_B(-1)$ and $P_C(-1)$.

Condition iii) is checked by the following test. For a polynomial

$$P(z) = \sum_{k=0}^{n} a_k z^k = a_n \prod_{i=1}^{n} (z - z_i),$$

define the $(n-1) \times (n-1)$ matrices

$$X = \begin{pmatrix} a_n & a_{n-1} & \cdots & \cdots & a_2 \\ 0 & a_n & & & \vdots \\ \vdots & & & & \vdots \\ \vdots & & & a_n & a_{n-1} \\ 0 & \cdots & \cdots & 0 & a_n \end{pmatrix},$$

$$Y = \begin{pmatrix} 0 & \cdots & \cdots & 0 & a_0 \\ \vdots & & & a_0 & a_1 \\ \vdots & & & & \vdots \\ 0 & a_0 & & & \vdots \\ a_0 & a_1 & \cdots & \cdots & a_{n-2} \end{pmatrix},$$

$$S(P) = X - Y.$$

It is shown in [4] that

$$\det S(P) = a_n^{n-1} \times \prod_{\substack{k=1 \\ i<k}}^{n} (1 - z_i z_k).$$

Notice that $S(P) = 0$ for $z_i = 1/z_k$, which is true for complex conjugate roots on the unit circle (and for pairs z_i, z_k for which $|z_i| > 1$ or $|z_k| > 1$, i. e. the stability boundary has been crossed already). The following arguments from [1] follow closely a corresponding argument in [3] for Hurwitz stability. $P_A(z, \alpha)$, $\alpha \in (0, 1)$, has no complex conjugate roots on the unit circle if and only if

$$S(P_A) = S[\alpha P_B + (1-\alpha) P_C] = \alpha S(P_B) + (1-\alpha) S(P_C)$$

is nonsingular for all $\alpha \in (0, 1)$. Equivalently,

$$\frac{S(P_A)}{\alpha} = S(P_B) - \lambda S(P_C)$$

(with $\lambda = (\alpha - 1)/\alpha$) is nonsingular for all $\lambda \in (-\infty, 0)$ and equivalently

$$\frac{S(P_A) S(P_C)^{-1}}{\alpha} = S(P_B) S(P_C)^{-1} - \lambda I$$

is nonsingular for all $\lambda \in (-\infty, 0)$. That is $S(P_B)\, S(P_C)^{-1}$ has no negative real eigenvalues.

Summary

A polytope of polynomials of degree n with ℓ interval parameters is Schur stable if and only if

i) all 2^ℓ vertices are Schur stable;
ii) for all $\ell 2^\ell - 1$ edges a testing matrix has no negative real eigenvalues. The computation of the testing matrix involves forming two $(n-1)\times(n-1)$ matrices from the coefficients of the corresponding vertex polynomials, one inversion and one multiplication.

Since the sequence of vertices is arbitrary, the total number of inversions is $\ell 2^{\ell-2}$.

See also

- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Optimal Control of a Flexible Arm
- ▶ Robust Control
- ▶ Semi-infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Ackermann J, Barmish BR (1988) Robust Schur stability of a polytope of polynomials. IEEE Trans Autom Control 33:984–986
2. Bartlett AC, Hollot CV, Huang-Lin (1988) Root locations of an entire polytope of polynomials: it suffices to check the edges. Math Control, Signals and Systems 1:61–71
3. Bialas S (1985) A necessary and sufficient condition for the stability of convex combinations of stable polynomials or matrices. Bull Polish Acad Sci 33:473–480
4. Jury EI, Pavlidis T (1963) Stability and aperiodicity constraints for systems design. IEEE Trans Circuit Theory:137–141

Robust Design of Dynamic Systems by Constructive Nonlinear Dynamics

Martin Mönnigmann[1],
Wolfgang Marquardt[2]

[1] Technische Universität Braunschweig, Braunschweig, Germany

[2] AVT – Process Systems Engineering, RWTH Aachen University, Lehrstuhl für Prozesstechnik, Aachen, Germany

MSC2000: 37N40, 90C30, 90C34

Article Outline

Introduction
Definitions
Method
  Critical Manifolds and Normal Vectors
  Algorithm
Cases
Conclusions
References

Introduction

In engineering applications, optimization problems often arise in which an optimal steady state of a dynamical system is sought. Chemical production plants, for example, can often be described by ordinary differential equations and algebraic equations, and an optimal continuous operation corresponds to an optimal steady state of the differential algebraic (DAE) model.

Because an optimal steady state is sought, the dynamical optimization problem reduces to an algebraic optimization problem. This point of view, however, neglects the stability properties of the dynamical system. While a steady state of a dynamical system is the solution of a nonlinear set of algebraic equations, these algebraic equations do not reveal anything about the stability of the resulting steady state. In fact, a steady state that is optimal with respect to a profit function may be found, but this steady state may turn out to be unstable. This problem has been addressed with the use of matrix measures before [2,14,17]. Unfortunately, matrix measures overestimate the region of instability in the process design space and therefore lead to suboptimal results.

The present contribution summarizes recent progress on a new approach to take stability boundaries into account in steady state optimization. The new methodology, which is based on ideas from nonlinear dynamics, permits to consider the stability of nonlinear dynamical systems without having to introduce approximations. The approach is based on the notion of the distance to critical manifolds in the parameter space [16]. This idea proves to be very generally applicable. As a result, the approach cannot only be used to consider stability boundaries in steady state optimization, but other boundaries that are critical for the systems' dynamics as well as feasibility boundaries.

In engineering models, some or all parameters are often *uncertain*, i. e., they are only known up to a considerable error. In a chemical production plant, for example, a precise value of some or all kinetic constants of the chemical reactions may not be known. In fact, uncertainty of this kind arises systematically in engineering models, because precise measurements of system parameters create cost themselves. Therefore, a tradeoff usually exists between measuring unknown system parameters to desirable precision on the one hand, and affordable precision on the other hand.

Definitions

In this section the problem is introduced. For a more detailed introduction we refer to [24].

A large class of dynamical systems can be modeled by differential-algebraic (DAE) systems of the form

$$\begin{aligned} \dot{x}^{\mathrm{d}}(t) &= f^{\mathrm{d}}(x^{\mathrm{d}}(t), x^{\mathrm{a}}(t), u(t), d(t), \vartheta, t),\ x^{\mathrm{d}}(0) = x_0^{\mathrm{d}}, \\ 0 &= f^{\mathrm{a}}(x^{\mathrm{d}}(t), x^{\mathrm{a}}(t), u(t), d(t), \vartheta, t), \\ y(t) &= h(x^{\mathrm{d}}(t), x^{\mathrm{a}}(t), u(t), d(t), \vartheta, t)\,, \end{aligned} \tag{1}$$

where $x = (x^{\mathrm{dT}}, x^{\mathrm{aT}})^{\mathrm{T}} \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, $d \in \mathbb{R}^{n_d}$, $\vartheta \in \mathbb{R}^{n_\vartheta}$, $y \in \mathbb{R}^{n_y}$ are state variables, inputs, disturbances, parameters, and outputs of the system. In (1), t denotes time, and $f := (f^{\mathrm{dT}}, f^{\mathrm{aT}})^{\mathrm{T}}$ and h are smooth functions which map from some subset $U \subset \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \times \mathbb{R}^{n_\vartheta} \times \mathbb{R}$ onto $\mathbb{R}^{n_x}$ and $\mathbb{R}^{n_y}$, respectively. The state variables x and the corresponding equations have been partitioned into dynamic state variables x^{d} and differential equations f^{d}, and algebraic variables x^{a} and equations f^{a}. In the sequel we assume that inputs u, and disturbances d vary only quasi-statically compared to the system dynamics. Mönnigmann and Marquardt [24] show that the system (1) can be simplified considerably under this assumption. The simplified system reads

$$\begin{aligned} \dot{x} &= f(x, \alpha, p), \quad x(0) = x_0, \\ y &= h(x, \alpha, p), \end{aligned} \tag{2}$$

where p denotes all inputs, references, and parameters that can be assumed to be known precisely, while α denotes inputs, references, disturbances, and parameters that can be modeled in terms of an average value and an uncertainty. The parameters α therefore have the form

$$\alpha_i \in [\alpha] := [\alpha_i^{(0)} - \Delta\alpha_i, \alpha_i^{(0)} + \Delta\alpha_i], \quad i = 1, \ldots, n_\alpha\,. \tag{3}$$

By a slight abuse of notation, we use the same symbols f and h in (1) and (2).

The steady state optimization problem for the uncertain dynamical system (2) with stability constraints can now be stated:

$$\begin{aligned} &\min_{x^{(0)}, \alpha^{(0)}, p^{(0)}} && \phi(x^{(0)}, \alpha^{(0)}, p^{(0)}) && (a) \\ &\text{subject to} && 0 = f(x^{(0)}, \alpha^{(0)}, p^{(0)}), && (b) \\ &&& \left.\begin{aligned} &0 = f(\tilde{x}, \tilde{\alpha}, p^{(0)}) \\ &\mathrm{Re}(\lambda) < 0 \quad \forall \lambda \in \sigma(J(\tilde{x}, \tilde{\alpha}, p^{(0)}) \end{aligned}\right\} \forall \tilde{\alpha} \in [\alpha], && (c) \\ &&& 0 \le g(\hat{x}, \hat{\alpha}, p^{(0)}) \qquad \forall \hat{\alpha} \in [\alpha], && (d) \\ &&& x \in X, \alpha \in A, p \in P\,. && (e) \end{aligned} \tag{4}$$

where ϕ is the merit function and (4b) ensures that the optimal point of operation is a steady state of the dynamical system. Equations (4c) constitute a semi-infinite constraint that guarantees all eigenvalues in the spectrum $\sigma(J)$ of the Jacobian $J(x, \alpha, p)$ of f with respect to x to be in the open left half of the complex plane. This constraint ensures stability of the nonlinear system. Since constraint (4c) enforces this condition for all α within the uncertainty region (3), the resulting optimal steady state will be robust with respect to the uncertainty in α. In (4d), g is a twice continuously differentiable function that maps into $\mathbb{R}^{n_g}$, $n_g \ge 1$. Equation (4d) is a semi-infinite constraint that ensures the inequality constraints to hold for all α. Finally, (4e) denotes bounds on x, α, and p. In a typical application

there exist box constraints on some or all of the state variables and parameters. We note that some or all of the nominal values of the uncertain parameters $\alpha^{(0)}$ may be degrees of freedom in the optimization.

Method

Mönnigmann et al. [19] introduced an algorithm for solving the semi-infinite problem (4). This algorithm is based on detecting and backing-off critical manifolds. We first introduce the notion of a critical manifold and a normal vector, and then present the algorithm for solving (4).

Critical Manifolds and Normal Vectors

In order to introduce the notion of a critical manifold, consider a single feasibility constraint

$$0 \leq g_i(x, \alpha, p) \tag{5}$$

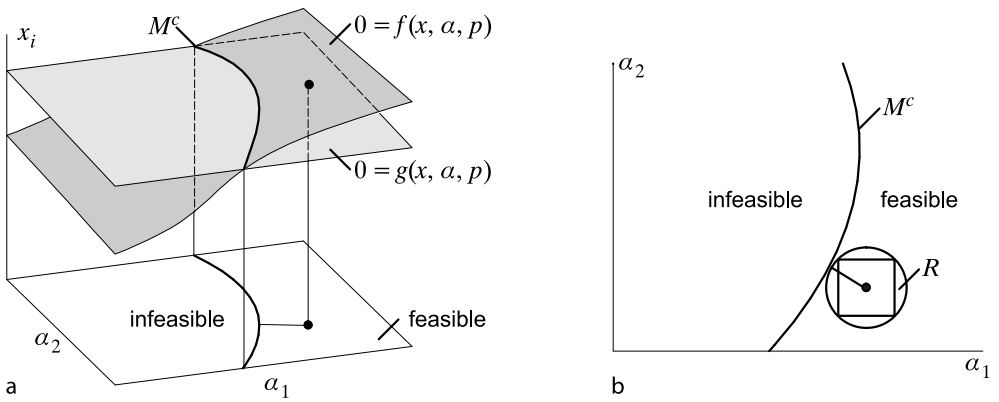
from among the feasibility constraints in (4d). In this simple case, the critical manifold, denoted by M^c, is defined by the set of points at which the constraint (5) is active,

$$M^c = \{(x, \alpha, p) | \, 0 = f(x, \alpha, p), 0 = g_i(x, \alpha, p)\} \ . \tag{6}$$

This critical manifold M_i^C separates the part of the space of uncertain parameters α in which the constraint holds from the part in which the constraint is violated, cf. Fig. 1.

In Fig. 1 the dot marks a candidate steady state of operation in the feasible regime. The shortest distance in the space of the uncertain parameters (α_1, α_2) between this candidate point of operation and the critical manifold occurs along the marked direction that is normal to the critical manifold. By imposing a minimum distance from the candidate point of operation to the critical boundary, we can ensure that the critical boundary is not crossed, regardless of the actual values of the uncertain parameters in the uncertainty region (3). This amounts to overestimating the uncertainty region by a circle and forcing the circle to at most touch the critical boundary tangentially, or to stay at a larger distance from the critical boundary, cf. Fig. 1. In the situation sketched in Fig. 1 we assume that the uncertain parameters have been scaled appropriately. This can be done, for example, by measuring them in units of their uncertainties $\Delta\alpha_i$ [18]. In the general case, the uncertainty region (3) defines a hyperrectangle and the circle shown in Fig. 1b becomes a hyperellipsoid. Regardless of the number n_α and scaling of the uncertain parameters, the distance between the candidate point of operation and the critical manifold, or the distance between the uncertainty region and the critical manifold can be measured with the aid of the normal vector. We refer to [18] for details.

The concepts of a critical manifold and of measuring the distance to a critical manifold along a normal vector have been introduced for feasibility constraints. The most important application of the normal vector approach, however, is the use of normal vectors to en-



Robust Design of Dynamic Systems by Constructive Nonlinear Dynamics, Figure 1
Critical manifold M^c and robustness region R. This figure is reproduced from [16]. We note that the overestimation introduced by the circle in (b) can be mitigated by other descriptions [9].

sure a parametric distance from stability boundaries in order to guarantee robust stability. Critical points for stability of ODE and DAE systems have been investigated in bifurcation theory. A thorough discussion of the use of normal vectors to manifolds of bifurcation points is beyond the scope of this paper and we refer the reader to textbooks in applied bifurcation theory (see, for example, [15]) and previous papers on the use of normal vectors in steady state optimization [18,21]. We give a sketch of the idea, however.

A simple result of nonlinear systems theory states that a steady state $(x^{(0)}, \alpha^{(0)}, p^{(0)})$ of a nonlinear system (2) is stable if all eigenvalues of the Jacobian $A(x, \alpha, p)$ defined by

$$A_{ij} = \frac{\partial f_i}{\partial x_j}(x, \alpha, p) \,,$$

evaluated at this steady state, lie in the left half of the complex plane. Since A is non-symmetric in general, stability may be lost due to either a real eigenvalue or a complex conjugate pair of eigenvalues in the open right half of the complex plane. In bifurcation theory, these two cases are known as saddle-node or Hopf bifurcation [15]. Under genericity conditions [15], a manifold of saddle-node (Hopf) bifurcations exists in the vicinity of a saddle-node (Hopf) bifurcation point. Saddle-node bifurcations are usually characterized by the necessary conditions

$$\begin{aligned} 0 &= f(x^\star, \alpha^\star, p^\star) && (a) \\ 0 &= A(x^\star, \alpha^\star, p^\star)v && (b) \\ 0 &= v^T v - 1 && (c) \end{aligned} \tag{7}$$

where $(x^\star, \alpha^\star, p^\star)$ is the bifurcation point. Equations (7a) ensure that the bifurcation point is a steady state. Equations (7b) are eigenequations with a real eigenvalue zero, the critical eigenvalue. Finally, (7c) is a regularization that is necessary because (7b) determines the eigenvector v up to its length only. Based on these necessary conditions the manifold

$$M^c := \left\{(x^\star, \alpha^\star, p^\star) \,|\exists v \in \mathbb{R}^n \text{ such that (7) holds}\right\} \tag{8}$$

can be stated. Similar necessary conditions can be stated for Hopf bifurcation points. Note that the determinant of A is zero if and only if a real zero eigenvalue exists. The determinant therefore can be used as a test function for checking if a steady state may be a saddle-node bifurcation (numerically more robust test functions exist, see [15], for example). Test functions of this type will be used in the algorithm to detect the crossing of critical manifolds.

Mönnigmann and Marquardt [18] presented a scheme for the derivation of systems of equations for the calculation of normal vectors. This scheme can be applied to defining equations for manifolds of the form (5) or (6). Mönnigmann and Marquardt [18] applied this scheme to a number of bifurcation point types. The saddle-node normal vector system, for example, comprises of

Equation (7) and

$$0 = \sum_{i=1}^{n_x} v_j \frac{\partial f_j}{\partial \alpha_i}(x^\star, \alpha^\star, p^\star) - r_i, \quad i = 1, \ldots, n_\alpha \,, \tag{9}$$

where $r \in \mathbb{R}^{n_\alpha}$ is the desired normal vector. This system has first been stated and used by Dobson [4]. Similar systems for the calculation of normal vector can be derived for the feasibility constraints, Hopf bifurcations, and other critical dynamical points [18]. In general, these systems have the form

$$0 = G^{(\tau)}(x, \tilde{x}^\tau, \alpha, p, r), \tag{10}$$

where $\tilde{x}$ denote auxiliary variables such as the eigenvector v in (7). The upper index τ is introduced to distinguish between the various types of critical points and manifolds such as $\tau = \texttt{saddle-node}$, $\tau = \texttt{Hopf}$, or $\tau = \texttt{feasibility}$.

Algorithm

Based on the normal vector systems (10) an algorithm for solving (4) can be stated. Before proceeding to the algorithm a few remarks are necessary. For brevity, these remarks are given in an informal fashion. For a more detailed and precise discussion we refer to [20,21].

Loosely speaking, the optimization algorithm will move the robustness region (3) (or an approximation thereof such as the circle in Fig. 1) in the parameter space while seeking a minimum for the profit function (4a). In this process, critical manifolds may be crossed. Test functions, like the determinant introduced in Sect.

"Critical Manifolds and Normal Vectors", can be used to signal the crossing of critical manifolds. Whenever a critical point is detected, this point is added to a set of known critical points denoted by $\mathcal{J}$. Later in the algorithm these points are used to initialize normal vector constraints. Critical points that have been used to initialize normal vector constraints are collected in a set $\mathcal{I}$. Points in $\mathcal{J}$ can be processed to $\mathcal{I}$ by solving an optimization problem that is not discussed here for brevity. We refer to [20] for details.

Given the index set $\mathcal{I}$, the following optimization problem is solved:

$$\begin{aligned}
&\min_{x^{(0)},\alpha^{(0)},p^{(0)}} \quad \phi(x^{(0)},\alpha^{(0)},p^{(0)}) && (a)\\
&\text{subject to} \quad 0 = f(x^{(0)},\alpha^{(0)},p^{(0)}) && (b)\\
&0 = G^{(\tau_i,i)}(x^{(i)},\tilde{x}^{(\tau_i,i)},\alpha^{(i)},p^{(0)},r^{(i)}), \quad \forall i \in \mathcal{I}, && (c)\\
&0 = \alpha^{(0)} - \alpha^{(i)} + l\frac{r^{(i)}}{||r^{(i)}||_2}, \quad \forall i \in \mathcal{I}, && (d)\\
&0 \le l^{(i)} - \sqrt{n_\alpha}, \quad \forall i \in \mathcal{I} && (e)\\
&x \in X, \quad \alpha \in A, \quad p \in P\,. && (f)
\end{aligned} \tag{11}$$

Because the crossing of critical manifolds is only detected at the nominal point of operation, critical manifolds may enter the robustness region without being detected [20]. For this reason, a rigorous numerical test for critical points in the robustness region is necessary. Interval arithmetics can be used to carry out this test [21] for problems of moderate complexity.

Finally, an adjustable parameter $\hat{l}$, $\hat{l} > \sqrt{n_\alpha}$, has to be chosen. This parameter is used to switch off normal vector constraints.

The algorithm can now be stated as follows.

1. (Initialization) Choose a steady state of (2) that is stable in the sense of (4c) and feasible with respect to the constraints (4d). Choose a value for $\hat{l}$. If critical point in the vicinity of the steady state are known a priori, put them into $\mathcal{J}$. Set $\mathcal{I} = \emptyset$.
2. (Update of $\mathcal{I}$) Process points in $\mathcal{J}$ to $\mathcal{I}$ [21]. Remove critical points from $\mathcal{I}$ for which $l^{(i)} > \hat{l}$.
3. (Optimization) Solve problem (11) using the current index set $\mathcal{I}$.
4. (Detection of critical manifolds) Analyze the path between the starting and end point of step 3. If a critical manifold has been crossed along this path, put this point into $\mathcal{J}$ and return to step 2. Otherwise proceed to step 5.
5. (Rigorous search for critical points) Check for critical points in the robustness region (3). If a critical point exists that has not been detected in step 4, put it into $\mathcal{J}$ and return to step 2. Otherwise stop.

For several details on the practical implementation we refer to [20]. For details on step 5 we refer to [21].

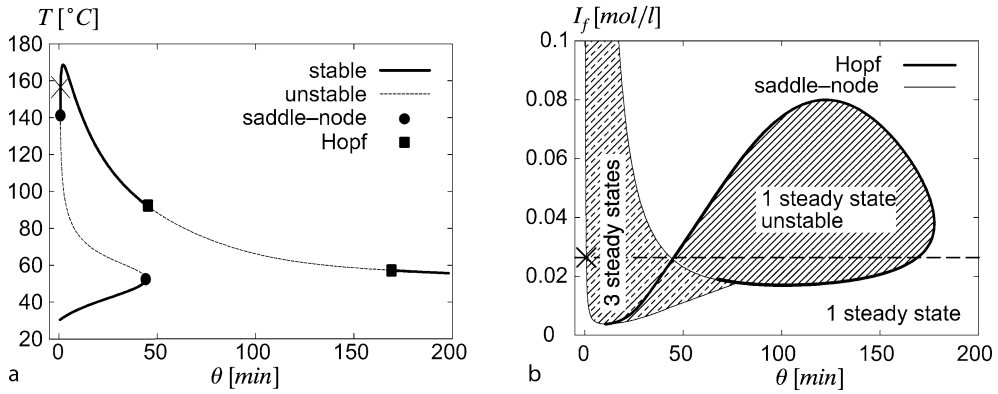
Cases

The normal vector based optimization approach has successfully been applied to different types of problems and a number of cases. The treated models range from a few to a few hundred model equations in size:

- Steady state optimization of parametrically uncertain dynamical systems with constraints for stability [1,9,21,22,24];
- Steady state optimization of parametrically uncertain systems with constraints on the location of higher codimension critical points. Normal vector constraints on the location of these points can be used to guarantee parametric robustness with respect to hysteresis, for example [7,18].
- Steady state optimization with parametrically robust pole placement. In applications of this type, the algorithm is applied to simultaneously seek an optimal plant design and to tune controller parameters [5,8,10,11,12,13,16,19,23].
- Optimization of parametrically uncertain transient processes [3,6].

Here the use of the normal vector based methodology is demonstrated with a small example. Because only two uncertain parameters exist in this example, the results can be visualized. The result for this example have first been published in [24].

The system treated here is a simple model for the solution free radical homopolymerisation of vinyl acetate. The model has been analyzed thoroughly with respect to its nonlinear dynamic behavior by numerical bifurcation analysis. We refer to [25] for details on the model and its analysis. The process is optimized with respect to an economic cost function that takes the cost of monomer, the cost of an initiator of the polymerization, and the cost of the solvent into account [24]. Two parameters are assumed to be uncertain. These are the residence time in the reactor θ and the concentration of



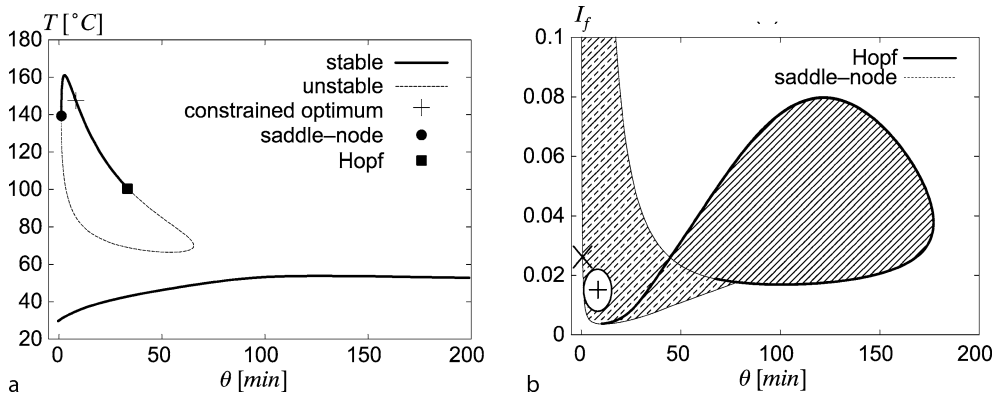
Robust Design of Dynamic Systems by Constructive Nonlinear Dynamics, Figure 2
Optimum which results in the first sweep through the algorithm with $\mathcal{J} = \emptyset, \mathcal{I} = \emptyset$. The optimum that results in step three is marked by the cross. These results were first reported in [24]

initiator I_f:

$$\begin{aligned} \Delta\theta &= \Delta\alpha_1 = 5 \text{ min}, \\ \Delta I_f &= \Delta\alpha_2 = 0.005 \text{ mol/l}. \end{aligned} \tag{12}$$

Assuming that nothing is known about the location of critical boundaries, the algorithm proceeds with $\mathcal{J} = \mathcal{I} = \emptyset$. The parameter $\hat{l}$ is set to $\hat{l} = 2$. The value of $\hat{l}$ is not important, since normal vector constraints are never switched off in step two in this particular example. The result of the optimization solved in step three is shown in Fig. 2a. Figure 2b shows the location of the stability boundaries which result from saddle-node and Hopf bifurcations as a function of θ and I_f. The particular value of I_f, at which Fig. 2a was obtained, is marked by the horizontal dashed line in Fig. 2b. While the optimal point of operation lies on a stable branch of steady states, the optimal point is obviously not robust with respect to variations in θ. The analysis in step 4 or 5 will therefore reveal that a loss of stability is likely due to the saddle-node bifurcations in the vicinity of the candidate optimal point.

In the next step a close saddle-node bifurcation is therefore added to $\mathcal{J}$, and the optimization (11) is repeated with a single normal vector constraint. The result is shown in Fig. 3. Since the robustness region marked by the ellipse contains no critical points, the algorithm terminates. An optimal point of operation that is stable despite the parametric uncertainty (12) has therefore been found.



Robust Design of Dynamic Systems by Constructive Nonlinear Dynamics, Figure 3
Optimum which results from the optimization with constraint on the minimal distance to the saddle-node bifurcation manifold. The optimum from Fig. 2 is marked for reference. These results were first reported in [24]

Conclusions

This contribution summarizes recent progress in the development of methods for the optimization of nonlinear dynamical systems with parametric uncertainties. The approach has successfully been applied to various problems in plant design and integrated plant design and controller tuning. Cases discussed here are restricted to steady state optimization. The extension to the robust optimization of transient processes is subject to current investigations [3,9].

References

1. Bischof CH, Lang B, Marquardt W, Mönnigmann M (2001) Verified determination of singularitites in chemical processes. In: Krämer W, von Gudenberg W (eds) Scientific Computing, Validated Numerics, Interval Methods. Kluwer Academic/Plenum Publishers, New York, pp 305–316
2. Blanco AM, Bandoni JA (2003) Interaction between process design and process operability of chemical processes: an eigenvalue optimization approach. Comput Chem Eng 27:1291–1301
3. Diehl M, Gerhard J, Marquardt W, Mönnigmann M (2008) Numerical solution approaches for robust nonlinear optimal control problems. Comput Chem Eng, in press
4. Dobson I (1993) Computing a closest bifurcation instability in multidimensional parameter space. J Nonlinear Sci 3:307–327
5. Gerhard J, Laiou M-C, Mönnigmann M, Marquardt W, Lakehal-Ayat M, Aneke E, Busch B (2005) Robust yaw control design with active differential and active roll control systems. In: Proceed 16th IFAC World Congress on Automatic Control, Prague, Czech Republic, July 4–8
6. Gerhard J, Marquardt W, Mönnigmann M (2008) Normal vectors on critical manifolds for robust design of transient processes in the presence of fast disturbances. SIAM J Appl Dyn Syst, in print
7. Gerhard J, Mönnigmann M, Marquardt W (2004) Robust stable nonlinear control and design of a CSTR in a large operating range. In: Shaw SL, MacGregor JF (eds) Proceed DYCOPS 7, Cambridge, MA, USA, July 5–7
8. Gerhard J, Mönnigmann M, Marquardt W (2005) Constructive nonlinear dynamics – Foundations and application to robust nonlinear control. In: Meurer T, Graichen K, Gilles ED (eds) Control and Observer Design for Nonlinear Finite- and Infinite-Dimensional Systems. Springer, Berlin, pp 165–182
9. Gerhard J, Mönnigmann M, Marquardt W (2006) Steady state optimization with guaranteed stability of a tryptophan biosynthesis model. Comput Chem Eng, in press
10. Grosch R, Briesen H, Marquardt W (2005) Simultaneous tuning of multiple control loops for an unstable crystallizer employing a robust design approach. In: 16th International Symposium on Industrial Crystallization (ISIC), Dresden, Germany, VDI-Berichte 1901, VDI-Verlag, pp 1027–1032
11. Grosch R, Mönnigmann M, Briesen H, Marquardt W (2002) Optimal design of a continuous crystallizer with guaranteed parametric robust stability. In: European Symposium on Computer Aided Process Engineering - 12, Elsevier, Amsterdam, pp 205–210
12. Grosch R, Mönnigmann M, Marquardt W (2003) Integrated robust optimal design and control of an MSMPR crystallization process. AIChE Annual Meeting, San Francisco, USA, November 16–21
13. Grosch R, Mönnigmann M, Marquardt W (2008) Integrated design and control for robust performance: Application to an MSMPR crystallizer. J Process Control 18(2):173–178
14. Kokossis AC, Floudas CA (1994) Stability in optimal-design – Synthesis of complex reactor networks. AIChE J 40(5):849–861
15. Kuznetsov YA (1999) Elements of Applied Bifurcation Theory, 2nd edn. Springer, New York
16. Marquardt W, Mönnigmann M (2005) Constructive nonlinear dynamics in process systems engineering. Comput Chem Eng 29:1265–1275
17. Mohideen MJ, Perkins JD, Pistikopoulos EN (1997) Towards an efficient numerical procedure for mixed integer optimal control. Comput Chem Eng 21:457–462
18. Mönnigmann M, Marquardt W (2002) Normal vectors on manifolds of critical points for parametric robustness of equilibrium solutions of ODE systems. J Nonlinear Sci 12:85–112
19. Mönnigmann M, Marquardt W (2005) Steady state process optimization with guaranteed robust stability and flexibility: Application to HDA reaction section. Ind Eng Chem Res 44:2737–2753
20. Mönnigmann M (2004) Constructive Nonlinear Dynamics Methods for the Design of Chemical Engineering Processes. PhD thesis, RWTH Aachen University
21. Mönnigmann M, Marquardt W, Beelitz T, Bischof CB, Lang B, Willems P (2007) A hybrid approach for efficient robust chemical process design. SIAM Rev 49:236–254
22. Mönnigmann M, Marquardt W (2000) Chemical process optimization with guaranteed robustness of nonlinear dynamics characteristics. AIChE Annual Meeting 2000, Los Angeles, November 12–17
23. Mönnigmann M, Marquardt W (2002) Parametrically robust control-integrated design of nonlinear systems. In: Proceedings of American Control Conference, vol 6, Anchorage, USA, pp 4321–4326
24. Mönnigmann M, Marquardt W (2003) Steady-state process optimization with guaranteed robust stability and feasibility. AIChE J 49(12):3110–3126
25. Teymour F, Ray WH (1992) The dynamic behaviour of continuous polymerization reactors – VI. Complex dynamics in full–scale reactors. Chem Eng Sci 47:4133–4140

Robust Global Optimization

HOANG TUY
Institute of Mathematics, VAST, Hanoi, Vietnam

MSC2000: 90C26, 90C31

Article Outline

Introduction
Difficulties of Problems with Nonconvex Constraints
$\mathcal{D}(C)$-Optimization Problems
A Robust Approach
  Successive Incumbent Transcending
  The SIT Algorithm for $\mathcal{D}(C)$ Optimization
Extensions
References

Introduction

Many deficiencies in global optimization methods for problems with nonconvex constraints require the necessity to reexamine certain concepts of approximate optimal solutions and to develop a robust approach to these problems.

Difficulties of Problems with Nonconvex Constraints

A wide class of global optimization problems have the form

$$\min\{f(x)|g_i(x) \geq 0 (i = 1, \ldots, m), x \in [a, b]\}, \quad \text{(P)}$$

where $f, g_1, \ldots, g_m$ are nonconvex continuous real-valued functions on $\mathbb{R}^n$, $a, b \in \mathbb{R}^n$, and $[a, b] := \{x \in \mathbb{R}^n | a \leq x \leq b\}$. When the feasible set

$$D := \{x \in [a, b] | g_i(x) \geq 0, i = 1, \ldots, m\}$$

is highly nonconvex, computing just one feasible solution may be almost as hard as solving the problem itself. Therefore, most current methods for solving these problems are confined to finding only an approximate optimal value rather than an approximate optimal solution. Even within these limitations, many methods are deficient in one way or another, providing approximate optimal solutions which are not guaranteed to be close enough to the true optimum, or in other cases that very unstable with respect to perturbations of the data or refinements of the tolerances.

Typically, the given problem is relaxed to

$$\min\{f(x)|g_i(x)+\varepsilon \geq 0 (i = 1, \ldots, m) x \in [a, b]\}, \quad \text{(R)}$$

where $\varepsilon > 0$ is the tolerance. Although this problem is a bit easier than (P) because it satisfies the regularity assumption, an optimal solution of it can rarely be computed exactly in finitely many iterations. So the best that one can expect to compute in finitely many iterations is an η-optimal solution of (R), i. e., a feasible solution $x^\star$ of (R) such that $f(x^*) \leq f(x) + \eta$ for every other feasible solution x. Such an $x^\star$ is sometimes referred to as an (ε, η)-*approximate optimal solution* of the original problem. Unfortunately, the inadequacy of this concept has been shown on simple examples where an (ε, η)-approximate optimal solution lies so far away from the true optimum that it can hardly be accepted as an approximation of the latter in some reasonable sense.

In view of these limitations and deficiencies, a robust approach to nonconvex global optimization problems is desirable which could provide the user with reliable good feasible solutions, stable under small perturbations of the data and easily implementable, though not necessarily the best among all possible feasible solutions.

$\mathcal{D}(C)$-Optimization Problems

The basic idea of this robust approach is to transform an optimization problem with a complicated nonconvex constraint set into a sequence of problems with nice constraint sets. Such a transformation is possible if the functions describing the problem belong to certain classes we are going to define.

For any two functions $g_1, g_2 \colon \mathbb{R}^n \to \mathbb{R}$ write $g = g_1 \vee g_2$, $h = g_1 \wedge g_2$ if $g(x) = \max\{g_1(x), g_2(x)\}$, $h(x) = \min\{g_1(x), g_2(x)\}$.

Let C be a family of real-valued functions on $\mathbb{R}^n$ such that
(i) $f_1, f_2 \in C, \alpha_1, \alpha_2 \in \mathbb{R}_+ \Rightarrow \alpha_1 f_1 + \alpha_2 f_2 \in C$;
(ii) $g_1, g_2 \in C \Rightarrow g_1 \vee g_2 \in C$.

Proposition 1 *Under assumptions i and ii the family $\mathcal{D}(C) = C - C$ is a vector lattice with respect to the two operations $\vee$ and $\wedge$.*

Also note that if $f \in \mathcal{D}(C)$ then $|f| \in \mathcal{D}(C)$ because $|f| = \max\{f, -f\}$.

An optimization problem of the form (P) where $f, g_i \in \mathcal{D}(C)$, $i = 1, \ldots, m$, is called a $\mathcal{D}(C)$-*optimization problem.*

Proposition 2 *Assume C satisfies assumptions i and ii and also that:*
(iii) Every function $x \mapsto x_i$, *with* $i \in \{1, \ldots, n\}$, *belongs to C.*

Then by simple manipulations any $\mathcal{D}(C)$*-optimization problem can be rewritten in the form (called "standard")*

$$\min\{f(x) | g(x) \geq 0, x \in [a, b]\}\,, \quad \text{where} \quad f \in C\,, \quad g \in \mathcal{D}(C)\,. \quad (1)$$

Two most important cases when C satisfies assumptions i, ii, and iii are:

1) C is the family of convex functions. Any $f \in \mathcal{D}(C)$ is then called a *dc function* and a $\mathcal{D}(C)$-optimization problem is called a *dc optimization* problem. Until recently most problems studied in global optimization could be shown to belong to this class [1].

2) C is the family of increasing functions, i. e., functions $f(x)$ such that $x' \leq x \Rightarrow f(x') \leq f(x)$. Any $f \in \mathcal{F}$ is then called a *dm function* and a $\mathcal{D}(C)$-optimization problem is called a *dm optimization*, or else a *monotonic optimization* problem. A theory of monotonic optimization has emerged in recent years that has been shown to parallel dc optimization in several respects [2,5,6].

As a result, any dc (dm, respectively) optimization problem can be written in the form (1) where f, g_1, and g_2 are convex (increasing, respectively) functions.

Since any polynomial can be viewed either as a dc or as a dm function, the set of dc functions or dm functions on a box $[a, b]$ is dense in the space $C[a, b]$ of continuous functions on $[a, b]$ with the supnorm topology. Virtually every global optimization of interest belongs to either of the basic classes described above.

A Robust Approach

In a continuous nonconvex optimization problem, an isolated optimal solution is usually very difficult to compute and very difficult to implement when it is computable. To avoid these difficulties most global optimization methods assume that the feasible set D satisfies

$$D = cl(int)D\,,$$

where cl$A$, int$A$ denote the closure and the interior, respectively, of the set A. However, this condition is generally very hard to check. Practically we often have to consider feasible sets which are not known a priori to contain isolated points or not.

Therefore, from a practical point of view it is desirable to know how to discard isolated feasible solutions without having to check for their presence.

A nonisolated feasible solution x^* of (P) is called an *essential optimal solution* if

$$f(x^*) \leq f(x) \quad \forall x \in D^*\,,$$

where D^* denotes the set of nonisolated points (accumulation points) of $D := \{x \in [a, b] | g(x) \geq 0\}$. In other words, an essential optimal solution is an optimal solution of the problem

$$\min\{f(x) | x \in D^*\}\,.$$

For a given tolerance $\varepsilon \geq 0$ a point $x \in [a, b]$ satisfying $g(x) \geq \varepsilon$ is said to be ε-*feasible*, and an ε-feasible point which is nonisolated is called an *essential* ε-*feasible solution*. In other words, an essential ε-feasible solution is a nonisolated point of the set $D_\varepsilon := \{x \in [a, b] | g(x) \geq \varepsilon\}$. For given tolerances $\varepsilon > 0, \eta > 0$, an essential ε-feasible solution x^* is called an *essential* (ε, η)-*optimal solution* if

$$f(x^*) \leq f(x) + \eta \quad \forall x \in D^*_\varepsilon\,,$$

where $D^*{}_\varepsilon$ is the set of all essential ε-feasible solutions. An essential (ε, η)-optimal solution for $\varepsilon = 0, \eta = 0$ is obviously essential optimal.

The above discussion suggests that instead of trying to find an optimal solution to (P), it would be more practical and reasonable to look for an essential (ε, η)-optimal solution.

The robust approach embodies this point of view for $\mathcal{D}(C)$-optimization, i. e., for a class of problems that includes virtually every nonconvex global optimization problem of interest.

Interchangeability Between Objective and Constraint
From now on we consider problem (P) where g_1,

$\ldots, g_m \in \mathcal{D}(C)$. Setting $g(x) = \min_{i=1,\ldots,m} g_i(x)$, we rewrite (P) as

$$\min\{f(x) | g(x) \geq 0\,, \quad x \in [a,b]\}\,, \tag{P}$$

where $f, g \in \mathcal{D}(C)$. Further, by Proposition 2, without loss of generality we can assume $f \in C$. Given $\varepsilon, \gamma \in \mathbb{R}$, let us consider the pair of optimization problems

$$\min\{f(x) | g(x) \geq \varepsilon, x \in [a,b]\}\,, \tag{P_ε}$$

$$\max\{g(x) | f(x) \leq \gamma, x \in [a,b]\}\,, \tag{Q_γ}$$

where the objective and constraint functions are interchanged. Owing to the fact $f \in C$, a key feature of problem (Q_γ) for our purpose is that its feasible set is a *convex* set (if C is the set of convex functions) or a *normal* set (if C is the set of increasing functions), so in either case problem (Q_γ) has no isolated feasible solution and computing a feasible solution to (Q_γ) can be done at cheap cost.

Proposition 3 *If max (Q_γ) $< \varepsilon$ then min (P_ε) $> \gamma$.*

On the basis of this property, the robust approach to $\mathcal{D}(C)$-optimization consists in replacing the original problem (P), possibly very difficult, by a sequence of easier, stable, problems (Q_γ), where the parameter γ can be iteratively adjusted until a stable (robust) solution to (P) is obtained.

Successive Incumbent Transcending

A key step towards finding a global optimal solution of a problem (P) is to deal with the following question of incumbent transcending.

(*, γ) *Given a real number γ, check whether problem (P) has an essential ε-feasible solution x satisfying $f(x) \leq \gamma$, and find one such solution if it exists.*

Using Proposition 3, consider a convergent branch-and-bound (BB) algorithm for solving problem (Q_γ) generating a sequence of partition sets M_k together with numbers $\alpha(M_k) \in \mathbb{R} \cup \{-\infty\}$, and points x^k, y^k such that

$$x^k \in M_k, f(x^k) \leq \gamma\,, \tag{2}$$

$$\alpha(M_k) \geq \max(7)\,, \tag{3}$$

$$\alpha(M_k) - g(x^k) \to 0 \quad (k \to +\infty)\,. \tag{4}$$

$\alpha(M)$ is an upper bound over M and (3) holds because M_k is the partition set with largest $\alpha(M)$ among all partition sets currently of interest, while (4) is the convergence condition.

From (4) we have $g(x^k) \to \max$ (Q_γ) and hence, for $\varepsilon > 0$ given, either $g(x^k) > 0$ for some k or $\alpha(M_{kk}) < \varepsilon$ for some k. In the former case, x^k is an essential ε-feasible solution of (P) satisfying $f(x^k) \leq \gamma$. In the latter case, max (Q_γ) $< \varepsilon$; hence by Proposition 3, min (P_ε) $> \gamma$, so no feasible solution x of (P) exists such that $f(x) \leq \gamma$ (hence, if $\gamma = f(\bar{x}) - \eta$ and $\bar{x}$ is an essential ε-feasible solution, then $\bar{x}$ is an η-optimal solution of (P)).

Thus, with the stopping criterium "$g(x^k) > 0$ or $\alpha(M_k) < \varepsilon$" a convergent BB procedure for solving (Q_γ) will help to transcend a given incumbent value γ, i. e., to solve the subproblem (*, γ).

For brevity a convergent BB procedure for solving (Q_γ) with stopping criterion "$\alpha(M) < \varepsilon$ or $g(x^k) > 0$" will be referred to as *procedure* (*, γ).

Using this finite procedure, one can solve problem (P) according to the following successive incumbent transcending (SIT) scheme, where γ_0 denotes an arbitrary real number larger than $\max\{f(x) | x \in [a,b]\}$.

SIT (Successive Incumbent Transcending) Scheme

Start with $\gamma = \gamma_0$.

Call procedure (*, γ). If an essential ε-feasible solution $\bar{x}$ of (P) is obtained with $f(\bar{x}) \leq \gamma$, reset $\gamma \leftarrow f(\bar{x}) - \eta$ and repeat. Otherwise, stop: $\bar{x}$ is an η-optimal solution if $\gamma = f(\bar{x}) - \eta$; problem (P) has no ε-feasible solution if $\gamma = \gamma_0$.

Since $f(D)$ is compact and $\eta > 0$ this scheme is necessarily finite.

The SIT Algorithm for $\mathcal{D}(C)$ Optimization

Incorporating procedure (*, γ) into the SIT scheme, we obtain the following SIT algorithm for (P).

Let γ_0 be any real number such that $\gamma > \max\{f(x) | x \in [a,b]\}$.

SIT algorithm for (P)

Select tolerances $\varepsilon > 0, \eta > 0$.

Step 0. Let $\mathcal{P}_1 = \{M_1\}, M_1 = [a,b], \mathcal{R}_1 = \emptyset$. Let $\gamma = \gamma_0$. Set $k = 1$.

Step 1. For each box (hyperrectangle) $M \in \mathcal{P}_k$ compute an upper bound $\alpha(M)$ for $\max\{g(x)|x \in M \cap \Omega, f(x) \leq \gamma\}$. Delete every M such that $\alpha(M) < \varepsilon$.

Step 2. Let $\mathcal{P}'_k$ be the collection of boxes resulting from $\mathcal{P}_k$ after completion of step 1. Let $\mathcal{R}'_k = \mathcal{R}_k \cup \mathcal{P}'_k$.

Step 3. If $\mathcal{R}'_k = \emptyset$ then *terminate*. If $\gamma = \gamma_0$ the problem (P) is essentially ε-infeasible; otherwise, the essential feasible solution $\bar{x}$ such that $\gamma = f(\bar{x}) - \eta$ is an essential η-optimal solution of (P).

Step 4. If $\mathcal{R}'_k \neq \emptyset$, let $M_k \in \operatorname{argmax}\{\alpha(M)|M \in \mathcal{R}'_k\}$. Divide M_k into two subboxes using the standard bisection. Let $\mathcal{P}_{k+1}$ be the collection of these two subboxes of M_k, $\mathcal{R}_{k+1} = \mathcal{R}'_k \setminus \{M_k\}$. Increment k, and return to step 1.

Proposition 4 *The SIT algorithm terminates after finitely many steps, yielding an (ε, η)-optimal solution or evidence that the problem has no essential ε-feasible solution.*

Extensions

So far all constraints have been assumed to be of inequality type: $g_i(x) \geq 0$, $i = 1, \ldots, m$. In the case when there are equality constraints such as

$$h_j(x) = 0\,, \quad j = 1, \ldots, s\,,$$

one can use linear equalities to eliminate certain variables, so without loss of generality it can always be assumed that all equality constraints are nonlinear. Since an exact solution to a nonlinear system of equations cannot be expected to be computable in finitely many steps, one should be content with replacing every given equality constraint $h_j(x) = 0$ by an approximate one:

$$-\delta \leq h_j(x) \leq \delta\,, \quad j = 1, \ldots, s\,,$$

where $\delta > 0$ is the tolerance. A mixed system with both inequality and equality constraints can thus be replaced with an approximate system involving only inequality constraints to which the above approach can be applied.

References

1. Tuy H (1978) Convex Analysis and Global Optimization. Kluwer, Dordrecht
2. Tuy H (2000) Monotonic Optimization: Problems and Solution Approaches. SIAM J Optim 11(2):464–494
3. Tuy H (2005) Robust Solution of Nonconvex Global Optimization problems. J Glob Optim 32:307–323
4. Tuy H (2005) Polynomial Optimization: A Robust Approach. Pac J Optim 1:357–374
5. Tuy H, Al-Khayyal F, Thach PT (2005) Monotonic Optimization: Branch and Cut Methods. In: Audet C, Hansen P, Savard G (eds) Essays and Surveys on Global Optimization. Springer, Berlin, pp 39–78
6. Tuy H, Hoai-Phuong NT (2007) A Robust Algorithm for Quadratic Optimization Under Quadratic Constraints. J Glob Optim 37:557–569

Robust Linear Programming with Right-Hand-Side Uncertainty, Duality and Applications

MICHEL MINOUX
University Paris, Paris, France

Article Outline

Abstract
Keywords and Phrases
Introduction
Duality and Robustness for LPs with Uncertainty on the RHS
Single-Stage Robust (LP) Decision Model
Two-Stage Robust (LP) Decision Model
An Application of the Two-Stage Robust LP Model with RHS Uncertainty: Robust PERT Scheduling
  Formulation as a Two-Stage Robust LP Model
  Differences with Bertsimas and Sim's Approach
Conclusions
References

Abstract

The various robust linear programming models investigated so far in the literature essentially appear to be based either on what is referred to as *rowwise* uncertainty models or on *columnwise* uncertainty models (these basically assume that the rows, or the columns, of the constraint matrix are subject to changes within

a well-specified *uncertainty set*). In this chapter, we discuss a special case of columnwise uncertainty, namely the subclass of robust linear programming (LP) models with uncertainty limited to the right hand side (RHS) only (this subclass does not appear to have been significantly investigated so far). In this context we introduce the concept of a 'two-stage robust LP model' as opposed to the standard case (which might be referred to as a 'single-stage robust LP model') and we address the question of whether LP duality can be used to convert a LP problem with RHS uncertainty into a robust LP problem with uncertainty on the objective function. We show how to derive both statements of (a) the dual to the robust model and (b) the robust version of the dual. The resulting expressions of the objective function to be optimized in both cases, appear to be clearly distinct. Moreover, from a complexity point of view, one appears to be efficiently solvable (it reduces to a convex optimization problem), whereas the other, as a nonconvex optimization problem, is expected to be computationally difficult in the general case. As an application of the two-stage robust LP model introduced here, we next investigate the *PERT (program evaluation and review technique) scheduling problem*, considering two possible natural ways of specifying the uncertainty set for the task durations: the case where the uncertainty set is a scaled ball with respect to the L_∞ norm; the case where the uncertainty set is a scaled Hamming ball of bounded radius (which, though leading to a quite different model, bears some resemblance to the well-known Bertsimas–Sim approach to robustness). We show that in both cases the resulting robust optimization problem can be efficiently solved in polynomial time.

Keywords and Phrases

Robust linear programming; Duality; PERT scheduling

Introduction

Various models for handling robustness objectives with respect to uncertainties on some specified coefficients in linear programming (LP) models have been proposed in the literature. We can mention Soyster [9], Ben-Tal and Nemirovski [1,2] and Bertsimas and Sim [3,4].

The various approaches proposed can roughly be divided into two distinct categories, depending on whether the underlying uncertainty model refers to possible fluctuations on the row vectors of the constraint matrix (we call this *'rowwise uncertainty'*), or on the column vectors (we call this *'columnwise uncertainty'*).

Columnwise uncertainty was first considered by Soyster [9]. In this model each column A_j of the $m \times n$ constraint matrix is either supposed to be exactly known, or is only known to belong to a given subset $K_j \subset \mathbb{R}^m$ ('uncertainty set'). The cost vector and the right hand side (RHS) are supposed to be certain. A robust solution is a solution which is feasible for all possible choices of the uncertain column vectors in their respective uncertainty sets. With this definition, assuming nonnegativity constraints on all variables of the LP, it can easily be shown that the problem of finding an optimal robust solution reduces to solving an ordinary LP with constraint matrix $A = (a_{i,j})$, where $\forall i, j$, and the coefficient $a_{i,j}$ is defined as

- $a_{i,j} = \max_{v \in K_j}\{v_i\}$ in case of a ith constraint of the form $\leq$
- $a_{i,j} = \min_{v \in K_j}\{v_i\}$ in case of a ith constraint of the form $\geq$.

Note that the above maximization (or minimization) can easily be carried out if we assume the uncertainty sets are either of finite cardinality (and not too big!) or closed convex.

As observed by many authors, a drawback of Soyster's model is that it usually leads to rather conservative solutions, in other words the price to pay for robustness in the above sense is often too high.

Contrasting with the above, rowwise uncertainty has attracted more interest and has been studied by, among others, Ben-Tal and Nemirovski [1,2], and more recently by Bertsimas and Sim [3,4].

Ben-Tal and Nemirovski start with the assumption that each row A_i of the constraint matrix belongs to a known uncertainty set consisting of an ellipsoid $E_i \subset \mathbb{R}^n$, and a solution $x \in \mathbb{R}^n, x \geq 0$ is said to be robust in this context if and only if it satisfies

$$\text{for all } i : A_i x \leq b_i, \forall A_i \in E_i.$$

Ben-Tal and Nemirovski then show that finding an optimal robust solution reduces to solving a conic

quadratic problem, which can be done in polynomial time.

A way to obviate nonlinearity, while retaining the idea of rowwise uncertainty, was proposed by Bertsimas and Sim [3,4], considering a slightly different model of uncertainty. More precisely, they assume that each uncertain coefficient $a_{i,j}$ can take values in a given interval $[a_{i,j} - \alpha_{i,j}, a_{i,j} + \alpha_{i,j}]$ and, for each row i, a positive parameter $\Gamma_i > 0$ (not larger than the total number of uncertain coefficients in row i) is considered. A solution x is then qualified as Γ-robust (in the sense of Bertsimas and Sim) if and only if for all $i = 1, \ldots, m$ this solution satisfies the ith constraint for all possible choices of the coefficients in row i such that at most Γ_i of the uncertain coefficients in the row are allowed to deviate from the nominal values $a_{i,j}$ (note that the above statement implicitly assumes the Γ_i parameters to be integers, but Bertsimas and Sim show that a slightly more general definition, allowing for nonintegral values of the Γ_i's can be handled in the same way). With this model of uncertainty, Bertsimas and Sim show that finding an optimal Γ-robust solution can be reduced to solving an ordinary linear program only moderately increased in size, thus opening the way to large-scale applications. Moreover the approach readily extends to optimization problems including integrality constraints on all or part of the variables; in that case the robust version of the problem is a mixed-integer program, but, again, the resulting robust model is only moderately increased in size compared with the original model.

In the present chapter we investigate a specific subclass of robust LP decision problems with columnwise uncertainty, namely LP problems with uncertainty on the RHS coefficients only. To handle such problems, a first natural idea would be to use duality to reformulate them as robust LPs with uncertainty on the objective. This is the subject of Sect. "Duality and Robustness for LPs with Uncertainty on the RHS".

Duality and Robustness for LPs with Uncertainty on the RHS

We first address in this section the question of whether duality can prove in any way useful to convert a columnwise uncertain linear program into a rowwise uncertain linear program, assuming of course the same uncertainty model for the columns of the given linear program and for the corresponding rows in the dual.

Intuitively, no nice (i. e. strong) duality result is to be expected when taking into account robustness constraints since, in both the primal and the dual, there is a price to pay for uncertainty; therefore, if we maximize in the primal, the robust primal optimal solution value will be (in general strictly) less than the optimal solution value of the 'nominal' primal LP, and minimizing in the dual will lead to a robust dual optimal solution value (in general) strictly larger than the same value.

Let us illustrate the phenomenon for a small typical example. Consider the following LP (a continuous knapsack problem actually) with two uncertain coefficients a_1 and a_2 in the constraint matrix

$$\begin{array}{lll} & \text{Maximize} & 4x_1 + 3x_2 \\ \text{(P)} & \text{subject to} & \\ & & a_1 x_1 + a_2 x_2 \leq 4 \\ & & x_1 \geq 0, x_2 \geq 0\,, \end{array}$$

the standard LP dual of which reads

$$\begin{array}{lll} & \text{Minimize} & 4u \\ \text{(D)} & \text{subject to} & \\ & & a_1 u \geq 4 \\ & & a_2 u \geq 3 \\ & & u \geq 0\,. \end{array}$$

Let us assume that the uncertainty set for a_1 is the real interval $[2, 3]$, the uncertainty set for a_2 is the real interval $[1, 2]$, and let us take as the definition of a robust solution in both (P) and (D) a solution which is feasible for any possible values of a_1 and a_2 in their respective uncertainty sets. Then it is easily seen that the optimal robust primal solution is $x^0 = [0, 2]$ with corresponding primal objective function value 6; and the optimal robust dual solution is $u^0 = 3$ with corresponding dual objective function value 12. This example thus clearly shows that no natural extension of the usual properties related to LP duality is to be expected in the context of robust LP.

A special case of columnwise uncertainty in LP is when uncertainty only concerns the coefficients of the RHS. Such problems frequently arise in practical applications. As a typical example, we mention the robust program evaluation and review technique (PERT) scheduling problem with uncertainty on the durations

of (some of) the tasks, assuming that a robust earliest termination date has to be determined. More precisely, we want to determine the minimum total duration of the project under any possible assignment of task durations, taken in a given uncertainty set. The two-stage robust model discussed in Sect. "Two-Stage Robust (LP) Decision Model" will appear to be relevant to such applications.

Consider the following LP

$$\begin{aligned} \max \quad & c^{\mathrm{T}} x \\ & Ax \leq b \\ & x \geq 0 \end{aligned}$$

and assume that the RHS b is not known exactly, but is only known to belong to some uncertainty set $B \subset \mathbb{R}^m$. The set B may be finite or infinite (we will introduce additional assumptions for this set when necessary).

Two distinct robustness models for LPs with uncertain RHS will be successively discussed in the following sections, namely single-stage robust decision models (Sect. "Single-Stage Robust (LP) Decision Model") and two-stage robust decision models (Sect. "Two-Stage Robust (LP) Decision Model"). For both cases it will be shown that, even in the restricted situation addressed here (uncertainty on the RHS only), one cannot use standard duality theory to convert a columnwise uncertain linear program into a rowwise uncertain linear program while preserving equivalence. Also, examples will be provided to show that the two-stage robust LP decision model is capable of producing less conservative solutions than the single-stage robust LP model.

Single-Stage Robust (LP) Decision Model

We first consider the simplest case where the values of all the decision variables x have to be fixed (taking into account uncertainty) before we get any kind of information on the actual realization of the uncertain parameters. In such a simple model (indeed a special case of Soyster's model) feasibility has to be ensured for any $b \in B$, and the problem to be solved simply reduces to

$$\begin{aligned} \max \quad & c^{\mathrm{T}} x \\ & Ax \leq \underline{b} \\ & x \geq 0\,, \end{aligned}$$

where $\forall i, \underline{b}_i = \min_{b \in B}\{b_i\}$.

The (standard) LP dual to the above problem reads

$$\begin{aligned} (\mathrm{D1}) \quad \min \quad & u^{\mathrm{T}} \underline{b} \\ & u^{\mathrm{T}} A \geq c \\ & u \geq 0\,. \end{aligned}$$

On the other hand, if we consider the dual to

$$\begin{aligned} \max \quad & c^{\mathrm{T}} x \\ & Ax \leq b \\ & x \geq 0 \end{aligned}$$

we get

$$\begin{aligned} \min \quad & u^{\mathrm{T}} b \\ & u^{\mathrm{T}} A \geq c \\ & u \geq 0\,. \end{aligned}$$

Now consider the robust version of this dual problem where the cost vector b is uncertain and can take any value in B. A simple and natural objective in this context is to find u achieving a minimum value of $\max u^{\mathrm{T}} b$ over all possible $b \in B$, thus leading to

$$\begin{aligned} (\mathrm{D2}) \quad \min_{u} \max_{b \in B} \quad & \{u^{\mathrm{T}} b\} \\ \text{subject to} \quad & u^{\mathrm{T}} A \geq c \\ & u \geq 0\,. \end{aligned}$$

It is clearly realized that (D1) and (D2) are completely different optimization problems for the same solution sets since

$$\forall u \geq 0, u^{\mathrm{T}} \underline{b} \leq \max_{b \in B}\{u^{\mathrm{T}} b\}\,,$$

with strict inequality holding in the general case.

If the set B is either finite or closed convex, it is observed that (D2) can be efficiently solved via standard convex optimization techniques since the function

$$u \rightarrow \max_{b \in B}\{u^{\mathrm{T}} b\}$$

is convex. Also, in this case, $\underline{b}$ can be efficiently computed since $\min_{b \in B}\{b_i\}$ too is a convex optimization problem; therefore problem (D1), too, can be efficiently solved.

Two-Stage Robust (LP) Decision Model

It frequently arises in applications that the process of decision making under uncertainty can be decomposed into two successive steps (two-stage decision making) or more (multistage decision making). For simplicity of presentation, we restrict ourselves here to the case of two-stage decision making. In this case, the set of decision variables x is decomposed (partitioned) into two distinct sets of variables which we denote y and z. The y variables concern the decisions to be taken in the first stage (before knowing anything about which realization of uncertainty will arise) and the z variables concern the decisions to be taken in the second stage (after realization of uncertainty).

Limiting ourselves, to make the discussion easier to follow, to the case where the objective function only depends on the decision variables of the first stage, our decision problem can thus be rewritten

$$\begin{array}{lll} \text{(I)} & \max & \gamma^{\mathrm{T}} y \\ & \text{subject to} & Fy + Gz \geq b \\ & & y \geq 0, z \geq 0\,, \end{array}$$

where γ and b are vectors and F and G are matrices of appropriate dimensions. (Observe that the reason for restricting ourselves to the case of an objective not depending on the z variables is only for the sake of simplicity in the presentation; our two-stage robust decision model would readily handle the general case of an objective depending on both the y and the z variables). Now, since the RHS b in (I) is uncertain, we have to make our robustness objectives precise. In the sequel, we consider robustness for a solution y by requiring that feasibility can be ensured for any possible RHS $b \in B$ by using the second-stage decision variables z (by analogy with the terminology used in stochastic programming, the z variables might be referred to as 'recourse' variables). So, if we define $Y = \{y / y \geq 0 \text{ and} \forall b \in B, \exists z \geq 0 : Gz \leq b - Fy\}$, we want to solve

$$\max_{y \in Y} \{\gamma^{\mathrm{T}} y\}.$$

Note that in the above, for any given robust solution y, the value taken by the z variables depends on which $b \in B$ is actually realized. This is an important feature of our model which explains why it can produce less conservative solutions than Soyster's model (see the example given in Remark 1 below).

According to Farkas' lemma, we know that, for fixed $b \in B$, a necessary and sufficient condition for the existence of $z \geq 0$ verifying $Gz \leq b - Fy$ is that

$$u^{\mathrm{T}}(b - Fy) \geq 0 \quad \forall u \text{ in the polyhedral cone:}$$
$$C = \{u / u^{\mathrm{T}} G \geq 0 \text{ and } u \geq 0\}.$$

Denoting $u^1, u^2, \ldots, u^p$, the extreme rays of the above cone, we can equivalently represent the set Y as the system of linear inequalities

$$(u^j)^{\mathrm{T}} Fy \leq (u^j)^{\mathrm{T}} b, \forall b \in B, \forall j = 1, \ldots p\,,$$

which is equivalent to

$$(u^j)^{\mathrm{T}} Fy \leq w^j = \min_{b \in B} (u^j)^{\mathrm{T}} b.$$

The robust two-stage decision problem is then reformulated as

$$\begin{array}{lll} \text{(I)}' & \max & \gamma^{\mathrm{T}} y \\ & \text{subject to} & (u^j)^{\mathrm{T}} Fy \leq w^j, \forall j = 1, \ldots, p \\ & & y \geq 0\,. \end{array}$$

Since we are interested in investigating duality in the context of robustness, let us state the (standard) dual to (I)′. So, introducing dual variables $\lambda_j, (j = 1, \ldots p)$, the dual to (I)′ can be written as

$$\begin{array}{lll} \min & \displaystyle\sum_j \lambda_j w^j = \sum_j \lambda_j \min_{b \in B} (u^j)^{\mathrm{T}} b \\ \text{subject to} & \displaystyle\sum_j \lambda_j F^{\mathrm{T}} u^j \geq \gamma \\ & \lambda_j \geq 0, \forall j = 1, \ldots, p \end{array}$$

and since: $\{u / u = \sum \lambda_j u^j, \lambda_j \geq 0\} = \{u / G^{\mathrm{T}} u \geq 0, u \geq 0\}$, this can be rewritten as

$$\begin{array}{lll} \text{(DI)}' & \min\limits_{u} \min\limits_{b \in B} & u^{\mathrm{T}} b \\ & \text{subject to} & F^{\mathrm{T}} u \geq \gamma \\ & & G^{\mathrm{T}} u \geq 0 \\ & & u \geq 0 \end{array}$$

or, equivalently if we denote W as the set of solutions to (DI)′,

$$\min_{u \in W} \min_{b \in B} \{u^{\mathrm{T}} b\}\,. \tag{1}$$

An a priori different way of using duality in our context would be to take the (standard) LP dual to (I) for fixed b, and then to carry out robustness analysis with respect to the coefficients b of the objective in the dual, allowing b to take all possible values in B . The LP dual to (I) reads

$$\begin{aligned} \text{(DI)} \qquad \min \quad & u^{\mathrm{T}} b \\ \text{subject to} \quad & F^{\mathrm{T}} u \geq \gamma \\ & G^{\mathrm{T}} u \geq 0 \\ & u \geq 0 \,. \end{aligned}$$

A natural robust version of (DI) consists in finding the dual solution u minimizing the value of $u^{\mathrm{T}} b$ produced by the worst possible b:

$$\min_{u \in W} \max_{b \in B} \{u^{\mathrm{T}} b\} \,, \tag{2}$$

which is to be contrasted with (1): indeed, it is seen that the robust version of the dual significantly differs from the dual of the robust version of the initial (primal) problem (I) because the function of u to be minimized is $\min_{b \in B}\{u^{\mathrm{T}} b\}$ in one case, and $\max_{b \in B}\{u^{\mathrm{T}} b\}$ in the other case.

It is worth observing that this structural difference between the two functions also implies a difference with respect to the practical solvability of the corresponding problems. The objective function in (2) is convex in u, making the robust version of (DI) efficiently solvable, whereas the objective in (1) is concave in u, making (DI)′ and thus (by standard LP duality) (I)′ also difficult problems in the general case.

Remark 1 As already suggested, the two-stage robust decision model proposed here is capable of producing less conservative solutions than Soyster's model. The reason for this is that if, for a given uncertainty set B, we consider Soyster's model for problem (I), the problem to be solved is

$$\max_{y \in Y_S} \{\gamma^{\mathrm{T}} y\} \,,$$

where the set Y_S is defined as $\{y / y \geq 0 \text{ and } \exists z \geq 0 : Gz \leq \underline{b} - Fy\}$ with $\underline{b}$ defined as

$$\forall i, \underline{b}_i = \min_{b \in B} \{b_i\} \,.$$

It is easily seen that $Y_S \subseteq Y = \{y / y \geq 0 \text{ and } \forall b \in B, \exists z \geq 0 : Gz \leq b - Fy\}$ and cases where strict inclusion holds (leading to an improved robust optimal solution value over the optimal value of Soyster's model) can easily be found, as illustrated by the following example.

In this example we consider three variables, $y \geq 0, z_1 \geq 0$ and $z_2 \geq 0$, and three constraints,

$$\begin{aligned} y - z_1 &\leq b_1 \\ y - z_2 &\leq b_2 \\ z_1 + z_2 &\leq b_3 \,, \end{aligned}$$

and the uncertainty set B is taken as the set containing the two vectors $(1, 0, 1)^{\mathrm{T}}$ and $(0, 1, 1)^{\mathrm{T}}$. The objective function is to maximize y. It is easily checked that the set Y_S corresponding to Soyster's model is in this case the real interval $[0, 1/2]$ leading to an optimal robust solution value 0.5. On the other hand, the set Y corresponding to our two-stage model is the real interval $[0, 1]$ leading to the (less conservative) optimal robust solution value 1. Indeed, the value $y = 1$ is feasible in our model because, in the case where $b = (1, 0, 1)^{\mathrm{T}}$ occurs, we can take $z_1 = 0$ and $z_2 = 1$, and in the case where $b = (0, 1, 0)^{\mathrm{T}}$ occurs, we can take $z_1 = 1$ and $z_2 = 0$. (Observe, as already pointed out, that the value taken by the z variables indeed depends on which $b \in B$ is actually realized.) Of course, this example does not rule out the possibility of having $Y = Y_S$ for some special instances. As will be seen in the next section, this possibility will arise in connection with the robust PERT scheduling problem, in the special case (referred to there as Case 1) where the uncertainty set on the task durations is the Cartesian product of a family of real intervals.

An Application of the Two-Stage Robust LP Model with RHS Uncertainty: Robust PERT Scheduling

In this section we specialize the general two-stage robust LP decision model investigated earlier to robust PERT scheduling, with an uncertainty set D on the durations of the tasks, supposed to be given as a (finite or infinite) list of 'scenarios'. More precisely, we want to determine an earliest termination date which can be achieved for any realization of the task durations d in a given uncertainty set D.

Formulation as a Two-Stage Robust LP Model

Consider a PERT network represented as a directed circuitless graph N in which the nodes correspond to tasks

(the tasks are numbered $i = 1, 2, \ldots n$; the set of tasks is denoted I), and there is an arc (i,j) with length (duration) d_i whenever there is a precedence constraint stating that processing of task j should not start before completion of task i. The set of arcs is denoted U. We assume that node 1 has no immediate predecessor (it thus represents the initial task) and node n has no direct successor (it thus represents the terminal task of the project). Denoting y_j $(j = 1, \ldots n)$ the starting date for each task j, and assuming first that the task durations d_i are exactly known, we want to minimize the total duration of the project while satisfying all precedence constraints; in other words

$$\begin{aligned} &\text{Maximize} \quad -y_n \\ &\text{subject to} \\ &\qquad y_1 = 0 \\ &\qquad y_i - y_j \le -d_i, \forall (i,j) \in U\,. \end{aligned}$$

Indeed, it is easy to check that in the above, $y_1 = 0$ can be replaced by $y_1 \ge 0$, or equivalently $-y_1 \le 0$; thus, the problem can be rewritten

$$\begin{aligned} &\text{Maximize} \quad -y_n \\ &\text{subject to} \\ &\qquad -y_1 \le 0 \\ &\qquad y_i - y_j \le -d_i, \forall (i,j) \in U\,. \end{aligned}$$

This model is recognized as a special case of (I), the constraint matrix $[F, G]$ being formed by the transpose of the node-arc incidence matrix of N with an additional row involving variable y_1 only (with associated coefficient -1). F is reduced in this case to a single column (the column corresponding to node n in the transpose of the incidence matrix of N). The RHS vector b is the vector with coefficients equal to the opposite of the task durations (more specifically, the RHS coefficient for the constraint corresponding to arc $(i,j) \in U$ is equal to $-d_i$). Note that we do not state explicitly the nonnegativity conditions on y, since they are implied by the precedence constraints and nonnegativity of the d_i coefficients.

Thus, the problem is cast in a form very similar to that of (I), the only difference being that the nonnegativity conditions on y and z are dropped. The consequence of this for the analysis in Sect. "Two-Stage Robust (LP) Decision Model" is just that we have to consider the polyhedral cone $C' = \{u / u^{\mathrm{T}} G = 0 \text{ and } u \ge 0\}$ instead of the polyhedral cone: $C = \{u / u^{\mathrm{T}} G \ge 0 \text{ and } u \ge 0\}$.

Owing to the special structure of the G matrix arising in the PERT scheduling problem, we have the following result.

Proposition 1 *The extreme rays of the polyhedral cone C' are in 1-1 correspondence with the characteristic vectors of the various paths between node 1 and node n in N.*

Proof: By observing that G^{T} is the node-arc incidence matrix of the graph N without the row associated with node n but with an extra column with coefficient -1 in the first row and all other coefficients 0, we realize that u satisfying $G^{\mathrm{T}} u = 0$ and $u \ge 0$ corresponds to a nonnegative flow between node 1 and node n in N with value equal to $u_{1,1}$, the component of u corresponding to the extra column with coefficient -1 in the first row and all other coefficients 0. Therefore the extreme rays of the cone C' correspond to the incidence vectors of the various paths connecting node 1 to node n in N. □

Let us denote $P = \{\pi^1, \pi^2, \ldots, \pi^K\}$ the set of all paths between 1 and n in N, $u^1, u^2, \ldots, u^K$ the corresponding characteristic vectors and the condition $(u^k)^{\mathrm{T}}(b - Fy) \ge 0$ specializes to $-\sum_{i \in \pi^k} d_i + y_n \ge 0$ (this is because, in that case

$$(u^k)^{\mathrm{T}} b = -\sum_{i \in \pi^k} d_i \text{ and} (u^k)^{\mathrm{T}} F y = -y_n).$$

So the condition for feasibility is that for each path π^k, $y_n \ge \sum_{i \in \pi^k} d_i$. In view of this, the robust PERT scheduling problem can be reformulated as

$$\begin{aligned} &\max \quad -y_n \\ &\text{subject to} \\ &\qquad y_n \ge \sum_{i \in \pi^k} d_i, \forall \pi^k \in P, \forall d \in D\,, \end{aligned}$$

where we recall that D denotes the uncertainty set for the task durations.

This problem therefore reduces to determining the path π^k maximizing, over the set P of all possible paths in N, the objective function

$$\max_{d \in D} \{\sum_{i \in \pi^k} d_i\}\,.$$

In other words we want to solve

$$\max_{\pi \in P} \max_{d \in D} \{\sum_{i \in \pi} d_i\} \text{ (RPS)}\,,$$

where RPS stands for robust PERT scheduling problem.

Now, if we want to go further into the analysis of (RPS), we have to specify how the uncertainty set D is defined. Of course there are many possible ways for this; we content ourselves below with examining two among the most natural possible definitions, and show that, for each of them, the above robust optimization problem (RPS) can be efficiently solved.

Case 1: *D is a scaled ball with respect to the L_∞ norm.*

The first easy special case is when, for each task i, the duration d_i can take any value in a given real interval $[d_i^-, d_i^+]$ with $0 \le d_i^- \le d_i^+$. In this case D is the Cartesian product $[d_1^-, d_1^+] \times [d_2^-, d_2^+] \times \cdots \times [d_n^-, d_n^+]$, which may be viewed as a scaled ball with respect to the L_∞ norm (using component-wise scaling to have all intervals of equal width). It is easily seen that an optimal robust solution for problem (RPS) can be obtained in this case by looking at a longest path (critical path) in N when each of the tasks i is assigned the longest possible duration d_i^+.

As an illustration of the above, consider the following example with $n = 7$ tasks, where the graph of precedence constraints has the following arcs: $(1, 2)$, $(1, 3)$, $(2, 3)$, $(2, 5)$, $(2, 6)$, $(3, 4)$, $(3, 7)$, $(4, 5)$, $(5, 7)$ and $(6, 7)$. Thus, task 2 cannot be started before completion of task 1, etc. Also note that the tasks are numbered according to a topological ordering of the graph, since there is no arc (i, j) with $i > j$. The associated intervals $[d_i^-, d_i^+]$ for the durations of the tasks are given in Table 1:

Robust Linear Programming with Right-Hand-Side Uncertainty, Duality and Applications, Table 1
Associated intervals $[d_i^-, d_i^+]$ for the durations of the tasks

Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
[2, 4]	[4, 8]	[3, 6]	[4, 8]	[4, 8]	[8, 16]

Task 7 is not shown in the table because it is a dummy task (of duration 0, without uncertainty) representing the end of the schedule. It is easy to see that in this example, the optimal solution to the (RPS) problem has duration 34 and corresponds to the critical path $(1, 2, 3, 4, 5, 7)$. Indeed 34 is the earliest achievable termination date if we require that the schedule remains feasible for any possible choice of the task durations in the Cartesian product $[d_1^-, d_1^+] \times [d_2^-, d_2^+] \times \cdots \times [d_6^-, d_6^+]$. This corresponds to the situation where the duration of each task i is d_i^+.

Case 2: *D is a scaled Hamming ball of bounded radius Γ.*

Here again we assume that, for each task i, the duration d_i can take any value in a given real interval $[d_i, d_i + \Delta_i]$ with $d_i \ge 0$. d_i is called the nominal value of the duration for task i, $d_i + \Delta_i$ being the possible extreme (or worst-case) value for the task duration. As is actually the case in many practical applications, it is unlikely that all tasks simultaneously take on worst-case values. To take this observation into account, we will impose an upper bound Γ on the number of task durations which are allowed to take on a worst-case value, given that all task durations which do not take on a worst-case value are assumed to have a value equal to their nominal value. More formally, if we associate with each task i a 0-1 integer variable u_i, the uncertainty set D corresponding to this definition is

$$D = \{\theta = (\theta_i)_{i=1,\ldots,n}, \theta_i = d_i + \Delta_i u_i \quad (i = 1, \ldots, n) \text{ such that } \sum_{i=1}^{n} u_i \le \Gamma, u_i \in \{0, 1\}, \forall i\}.$$

As can be seen from the above definition, in the special case where all Δ_i are equal to 1, D is recognized as the Hamming ball of radius Γ centered at $d = (d_i)$, $i = 1, \ldots n$ (in other words, $\theta - d$ can be any 0-1 vector with at most Γ components equal to 1). When the Δ_i's take on arbitrary positive values, the Hamming ball structure is still present after applying scaling to each component i with respect to the corresponding Δ_i value.

We note here that, in spite of the fact that the definition above is close in spirit to the concept of uncertainty suggested by Bertsimas and Sim [3,4], our model is fairly different from the one studied by those authors, since they restricted themselves to rowwise uncertainty, whereas in our robust PERT scheduling problem, we have uncertainty on the RHS only (a special case of columnwise uncertainty). For more detailed discussion of this issue, see Sect. "Differences with Bertsimas and Sim's Approach".

We now show that, with the above definition of the uncertainty set D, problem (RPS) can be efficiently solved via a dynamic programming recursion. To that

aim, we will consider the problem with parameter Γ as only one representative of the class of problems (RPS$[i,k]$) for i running from 1 to n (the number of tasks) and k running from 0 to Γ. More precisely, assuming that the tasks are numbered according to a topological ordering of the circuitless graph N, (RPS$[i,k]$) consists of the robust PERT scheduling subproblem corresponding to the subset of tasks 1, 2,…,i, the durations of at most k of which are allowed to take on their worst-case values. The case $k=0$ (no deviation allowed) corresponds to the usual PERT scheduling problem in terms of the nominal values for the task durations. We denote $v^*[i,k]$ the optimal objective function value for problem (RPS$[i,k]$), and for any task i, we denote Pred$[i]$ the set of tasks j such that (j,i) is an arc of N (the set of direct predecessors of node i). Bellman's optimality principle then leads to the following dynamic programming recursion:

$$\forall i \in [1,n], \forall k = 0,1,\ldots\Gamma:$$
$$v^*[i,k] = \max_{j \in Pred[i]} \max\{v^*[j,k] + d_j;\; v^*[j,k-1] + d_j + \Delta_j\}. \quad (3)$$

The optimal value of the robust PERT scheduling problem we are interested in is then $v^*[n,\Gamma]$. The rationale behind (3) can easily be explained as follows. Consider the set of all paths from 1 to $j \in$ Pred$[i]$. The duration of arc (j,i) has nominal value d_j and worst-case value $d_j + \Delta_j$. The maximum duration of a path from 1 to i through j with at most k tasks allowed to deviate from their nominal values can be obtained: either by allowing for at most k deviations on the subset of tasks $\{1,2,\ldots,j\}$ and taking the nominal duration for arc (j,i) or by allowing at most $k-1$ deviations on the subset of tasks $\{1,2,\ldots j\}$ and taking the worst-case duration for arc (j,i). Thus, the optimal value for node i via node j is the maximum value among these two alternatives, and the optimal value for node i is the maximum taken on the set of all direct predecessors of i. Obviously, solving the recursion (3) is achieved in polynomial time $O(m \times n)$, where m is the number of arcs and n the number of nodes of the PERT network; more precisely the complexity is $O(m \times \Gamma)$, where Γ, the parameter defining the uncertainty set, is at most n, the number of tasks (but is often significantly smaller than n in practical applications).

Robust Linear Programming with Right-Hand-Side Uncertainty, Duality and Applications, Table 2

	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$
$k=0$	2	6	9	13	6	17
$k=1$	4	10	13	17	10	22
$k=2$	4	12	16	21	12	26
$k=3$	4	12	18	24	12	29

Let us illustrate the above for the same seven-task example as the one considered to illustrate Case 1. We thus consider the same intervals for the task durations, the lower bound of each interval representing the nominal task duration, and the upper bound representing the worst-case duration. For $\Gamma = 3$, application of the recursion (3) leads to the $v^*[i,k]$ values shown in Table 2.

For instance the value $v^*[4,2] = 16$ corresponds to the path (1, 2, 3, 4) with three arcs of nominal durations 2, 4 and 3, respectively. If, in this path, two arcs out of three are allowed to take on their worst-case durations, the worst case (maximum) is obtained when task 2 has duration 8 and task 3 has duration 6 (task 1 keeping its nominal duration 2), the resulting length of the path being 8 + 6 + 2 = 16. Let us also illustrate how the recursion (3) works for computing, e. g., $v^*[7,2]$ and $v^*[7,3]$. We have

$$\begin{aligned} v^*[7,2] = \max\{&v^*[3,2]+3; v^*[3,1]+6; \\ &v^*[5,2]+4; v^*[5,1]+8; \\ &v^*[6,2]+8; v^*[6,1]+16\} \\ = \max\{&15,16,25,25,20,26\} = 26. \end{aligned}$$

The maximum above is obtained for $j=6$ and the corresponding optimal path is (1, 2, 6, 7).

Similarly we have

$$\begin{aligned} v^*[7,3] = \max\{&v^*[3,3]+3; v^*[3,2]+6; \\ &v^*[5,3]+4; v^*[5,2]+8; \\ &v^*[6,3]+8; v^*[6,2]+16\} \\ = \max\{&15,18,28,29,20,28\} = 29. \end{aligned}$$

The maximum above is obtained for $j=5$ and the corresponding optimal path is $(1,2,3,4,5,7)$. It is thus seen that, depending on the choice of the control parameter Γ, various optimal paths are obtained which, of course, may differ from the optimal solution to the

nonrobust PERT scheduling problem (considering only the nominal values for the task durations). Also, observe that the value $v^*[7,3] = 29$ corresponds to a less conservative robust solution than the one obtained in Case 1.

Differences with Bertsimas and Sim's Approach

We now turn to show that, in spite of the similarity in the definition of the uncertainty sets, the robust version of the PERT scheduling problem investigated here is essentially different from the model proposed by Bertsimas and Sim [3] for the robust version of the shortest-path problem. From an abstract point of view, the difference basically stems from the fact that, in our case, we are faced with a LP problem with uncertainty on the RHS, whereas Bertsimas and Sim addressed a LP problem with uncertainty on the cost coefficients. However, to further understand the source of this difference, we show below which difficulties would arise if we wanted to apply the Bertsimas–Sim approach to the robust longest (critical) path problem on a directed circuitless graph G.

Following these authors, the robust shortest s-t path problem in G with uncertainty parameter Γ (assuming $\Gamma \in \mathbb{N}$) is formulated as

$$\text{(RSP)}\quad \min_{x\in X}\left\{\sum_{(i,j)\in U} c_{i,j}x_{i,j} + \max_{S\subseteq U,|S|\le\Gamma}\sum_{(i,j)\in S}\Delta_{i,j}x_{i,j}\right\},$$

where X denotes the set of incidence vectors of all s-t paths in G, $c_{i,j}$ denotes the nominal cost of arc (i,j) and $c_{i,j} + \Delta_{i,j}$ is the worst-case cost of arc (i,j). After transformation of (RSP) using the duality theorem to convert the second term in the braces into a minimization, the problem is reformulated as a standard LP, the solution of which reduces to $m + 1$ applications of a standard shortest-path algorithm. Observe that one of the reasons for all the above to work so nicely is that the second term in the braces, as a function of x, is convex in x, since it is the pointwise maximum of a finite number of linear functions.

The above approach is still valid if, instead of looking for an optimum robust minimum cost path, we are looking for an optimum robust maximum benefit path: $c_{i,j} > 0$ being interpreted as a reward associated with the use of arc (i,j), the effect of uncertainty being to reduce the nominal reward $c_{i,j}$ by the amount $\Delta_{i,j}$. The problem would then take the form

$$\max_{x\in X}\left\{\sum_{(i,j)\in U} c_{i,j}x_{i,j} - \max_{S\subseteq U,|S|\le\Gamma}\sum_{(i,j)\in S}\Delta_{i,j}x_{i,j}\right\},$$

which is essentially analogous to the above robust minimum cost path, up to a change in the signs of the coefficients in the objective (still assuming, of course, that the graph G under consideration is circuitless). In particular, we note that the function to be maximized is concave, so we still have a convex optimization problem.

By contrast, the robust PERT scheduling problem addressed in the present paper is formulated as

$$\max_{x\in X}\left\{\sum_{(i,j)\in U} c_{i,j}x_{i,j} + \max_{S\subseteq U,|S|\le\Gamma}\sum_{(i,j)\in S}\Delta_{i,j}x_{i,j}\right\},$$

with $c_{i,j} > 0$ and $\Delta_{i,j} > 0$.

It is then readily observed that this problem consists in maximizing a convex function of x on $\{0,1\}^m$, and it is well known that this cannot be simply reduced to ordinary LP as is the case for Bertsimas and Sim's approach. Thus, robust PERT scheduling may be viewed as a typical illustration of the big differences between models featuring rowwise uncertainty and models featuring columnwise uncertainty in robust LP.

Conclusions

In this chapter, various robust LP problems have been investigated, and the question of whether LP duality can still be used to help solve such problems has been addressed and answered negatively. Among the problems considered, robust LP problems with uncertainty on the RHS only have been recognized as an interesting subclass of problems, for which the solution techniques should not confine themselves to the classical approach proposed by Soyster [10]. In this respect we have been led to propose a new class of robust LP models referred to here as 'two-stage robust decision models' which can be expected to lead to less 'conservative' optimal robust solutions than those usually obtained from Soyster's model. In order to show the practical usefulness of this two-stage model, a specialization to robust PERT scheduling was discussed, leading, under two natural ways of defining the uncertainty set with respect to

the task durations, to efficient solution methods. Also some fundamental differences between our approach to robust PERT scheduling and the one proposed by Bertsimas and Sim [3] in the context of the robust shortest-path problem were pointed out. We think that many other possible applications of this two-stage robust modeling approach deserve further investigations, for instance in dynamic inventory management, optimal resource allocation problems and telecommunication problems.

References

1. Ben-Tal A, Nemirovski A (1998) Robust Convex Optimization. Math Oper Res 23:769–805
2. Ben-Tal A, Nemirovski A (2000) Robust Solution of Linear Programming Problems Contaminated with Uncertain Data. Math Prog 88:411–424
3. Bertsimas D, Sim M (2003) Robust Discrete Optimization and Network Flows. Math Prog B 98:49–71
4. Bertsimas D, Sim M (2004) The Price of Robustness. Oper Res 52:1 35–53
5. Chinneck JW, Ramadan K (2000) Linear Programming with Interval Coefficients. J Oper Res Soc 51:2 209–220
6. Inuiguchi M, Sakawa M (1995) Minimax Regret Solution to Linear Programming Problems with an Interval Objective Function. EJOR 86:526–536
7. Kouvelis P, Yu G (1997) Robust Discrete Optimization and its Applications. Kluwer, Boston
8. Mulvey JM, Vanderbei RJ, Zenios SA (1995) Robust Optimization of Large Scale Systems. Oper Res 43:2 264–281
9. Soyster AL (1973) Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming. Oper Res 21:1154–1157
10. Soyster AL (1979) Inexact Linear Programming with Generalized Resource Sets. EJOR 3:316–321

Robust Optimization

RO

Stavros A. Zenios
University Cyprus, Nicosia, Cyprus

MSC2000: 90C90, 91B28

Article Outline

Keywords
Robust Optimization Models
Comparisons of Robust Optimization with Sensitivity Analysis and Stochastic Linear Programming
Notes
See also
References

Keywords

Optimization

The analyst who attempts to build a mathematical model for a real-world system is often faced with the problem of uncertain, noisy, incomplete or erroneous data. This is true for several application domains. In business applications noisy data are prevalent. Returns of financial instruments, demand for a firm's products, the cost of fuel, and consumption of power and other resources are examples of model data that are known with some probabilistic distribution at best. In social sciences data are often incomplete – for example, partial census surveys are carried out periodically in lieu of a complete census of the population. In the physical sciences and engineering data are usually subject to measurement errors, as in models of image restoration from remote sensing experiments. For some applications not much is lost by assuming that the value of the uncertain data is known and then developing a deterministic mathematical programming model. Worst case or mean values can be used in this respect because they provide reasonable approximations when either the level of uncertainty is low, or when the uncertain parameters have a minor impact on the system we want to model. For many applications, however, uncertainty plays a key role in the performance of the real-world system: worst-case analysis often leads to conservative and potentially expensive solutions, and solving the *mean value problem*, i. e., a problem where all random variables are replaced by their mean values, can even lead to nonsensical solutions since the mean of a random variable might not be a value that can be realized in practice.

A general approach to dealing with uncertainty is to assign to the unknown parameters a probability distribution, which should then be incorporated into an appropriate mathematical programming model. Early models for dealing with data uncertainty were in the

form of sensitivity analysis of the mean value model. Later developments formulated stochastic linear programming models whereby the data uncertainty was incorporated in the estimation of the expected value of the objective function. This chapter addresses the problem of planning under uncertainty using robust optimization models.

Robust Optimization Models

Robust optimization formulations are applicable to optimization models that have two distinct components: a *structural component* that is fixed and free of any noise in its input data; and a *control component* that is subject to noisy input data. In some cases the robust optimization model is identical to a two-stage stochastic program with recourse. But it also allows additional flexibility in dealing with noise. In order to define the model we introduce two sets of variables:

- $x \in \mathbf{R}^{n_0}$ denotes the vector of decision variables that depend only on the noise-free structural constraints. These are the *design variables* whose values are independent of realizations of the noisy parameters.
- $y \in \mathbf{R}^{n_1}$ denotes the vector of *control variables* that can be adjusted once the uncertain parameters are observed. Their optimal values depend both on the realization of uncertain parameters, and on the optimal values of the design variables.

The optimization model we are interested in has the following structure:

$$\min \langle c, x\rangle + \langle d, y\rangle \tag{1}$$

such that

$$Ax = b, \tag{2}$$

$$Bx + Cy = e, \tag{3}$$

$$x \in \mathbb{R}^{n_0}_{+}, \tag{4}$$

$$y \in \mathbb{R}^{n_1}_{+}, \tag{5}$$

where b, c, d, e are given vectors, A, B, C are given matrices and $\langle \cdot, \cdot \rangle$ is the inner product of its arguments. Equation (2) denotes the *structural constraints* that are free of noise. Equation (3) denotes the *control constraints*. The coefficients of these constraints, i. e., the elements of B, C and e are subject to noise. The cost vector d is also subject to noise, while A, b and c are not.

To define the robust optimization problem we introduce an index set $\Omega := \{1, \ldots, N\}$. With each index $s \in \Omega$ we associate the scenario set $\{d(s), B(s), C(s), e(s)\}$ of realizations of the control coefficients. Reference to an index s implies reference to the scenario set associated with this index. The probability of the sth scenario is p_s, and $\sum_{s \in \Omega} p_s = 1$. Now the following question is posed: What are the desirable characteristics of a solution to problem (1)–(5) when the coefficients of the constraints (3) take values from some given set of scenarios? The solution is considered robust with respect to optimality if it remains 'close' to optimal for any realization of the scenario index $s \in \Omega$. The problem is then termed *solution robust*. The solution is robust with respect to feasibility if it remains 'almost' feasible for any realization of s. The problem is then termed *model robust*. The concepts of 'close' and 'almost' are precisely defined later through the choice of appropriate norms.

It is unlikely that a solution to the mathematical program will remain both feasible and optimal for all realizations of s. If the system being modeled has substantial built-in redundancies, then it might be possible to find solutions that remain both feasible and optimal. Otherwise a model is needed that permits a trade-off between solution and model robustness. Robust optimization models formalize a way to measure this trade-off.

First let us introduce a set $\{y^1, \ldots, y^N\}$ of control variables for each scenario $s \in \Omega$, and another set $\{z^1, \ldots, z^N\}$ of *feasibility error vectors* that measure the infeasibility of the control constraints under each scenario.

The real-valued objective function $\xi(x, y) = \langle c, x\rangle + \langle d, y\rangle$ is a random variable taking the value $\xi_s(x, y^s) := \langle c, x\rangle + \langle d(s), y^s\rangle$ with probability p_s. Hence, there is no longer a simple single choice for an aggregate objective function. The expected value

$$\sigma(\cdot) := \sum_{s \in \Omega} p_s \xi_s(\cdot) \tag{6}$$

is precisely the objective function used in stochastic programming formulations. Another choice is to employ worst-case analysis and minimize the maximal value. The objective function is then defined by

$$\sigma(\cdot) := \max_{s \in \Omega} \xi_s(\cdot).$$

The robust optimization formulation also allows the introduction of higher moments of the distribution of $\xi(\cdot)$ in the optimization model. Indeed, the introduction of higher moments is one of the features of robust optimization that distinguishes it from the stochastic programming model. For example, we could use a nonlinear utility function that embodies a trade-off between mean value and variability in this mean value. If $\mathcal{U}(\xi_s)$ denotes the utility of ξ_s, then the function

$$\sigma(\cdot) := \sum_{s \in \Omega} p_s \mathcal{U}(\xi_s(\cdot))$$

captures the risk preference of the user. A popular choice of utility functions, typically used in portfolio management applications, is the logarithmic function $\mathcal{U}(\xi_s) = \log \xi_s$. The general robust optimization model includes a term $\sigma(x, y^1, \ldots, y^N)$ in the objective function to denote the dependence of the function value on the scenario index s. This term controls solution robustness, and can take different forms depending on the application. The examples mentioned above are some popular choices.

The robust optimization model introduces a second term in the objective function to control model robustness. This term is a feasibility penalty function, denoted by $\rho(z^1, \ldots, z^N)$, and it is used to penalize violations of the control constraints under some of the scenarios. The introduction of this penalty function also distinguishes the robust optimization model from the stochastic programming approach for dealing with noisy data. In particular, the model recognizes that it may not always be possible to arrive at a feasible solution to a problem under all scenarios. Infeasibilities will inevitably arise, and they will be dealt with outside the optimization model. The robust optimization model generates solutions that present the modeler with the fewest infeasibilities to be dealt with outside the model.

The specific choice of penalty function is problem dependent, and it also has implications for the choice of a solution algorithm. Two suitable *penalty functions* are the following:

- $\rho(z^1, \ldots, z^N) := \sum_{s \in \Omega} p_s \parallel z^s \parallel^2$. This quadratic penalty function (i. e., a weighted ℓ_2-norm) is applicable to equality control constraints where both positive and negative violations of the constraints are equally undesirable. The resulting quadratic programming problem is twice continuously differentiable, and can be solved using standard quadratic programming algorithms, although it is typically large scale.
- $\rho(z^1, \ldots, z^N) := \sum_{s \in \Omega} p_s \max(0, \max_j z_j^s)$, where z_j^s is the jth element of vector z^s. This penalty function is applicable to inequality control constraints when only positive violations are of interest (negative values of some z_j indicate slack in the inequality constraints). With this choice of penalty function, however, the resulting mathematical program is nondifferentiable.

The robust optimization model takes a multicriteria objective form. A goal programming weight parameter λ is used to derive a spectrum of answers that trade off solution for model robustness. The general formulation of the robust optimization model is stated as follows:

$$\begin{cases} \min & \sigma(x, y^1, \ldots, y^N) + \lambda \rho(z^1, \ldots, z^N) \\ \text{s.t.} & Ax = b, \\ & B(s)x + C(s)y^s + z^s = e(s), \quad \forall s \in \Omega, \\ & x \in \mathbb{R}_+^{n_0}, \\ & y^s \in \mathbb{R}_+^{n_1}, \quad \forall s \in \Omega, \\ & z^s \in \mathbb{R}_+^{m}, \quad \forall s \in \Omega. \end{cases}$$

Comparisons of Robust Optimization with Sensitivity Analysis and Stochastic Linear Programming

We compare here the alternative approaches for dealing with uncertainty. In particular, we contrast robust optimization (RO), stochastic linear programming (SLP) and sensitivity analysis (SA). We will see that RO enjoys several advantages, but it is not without its shortcomings.

Sensitivity analysis is a reactive approach to controlling uncertainty. It just measures the sensitivity of a solution – typicallly of the solution to the mean value problem – to changes in the input data. It provides no mechanism by which this sensitivity can be controlled.

Stochastic linear programming and robust optimization are constructive approaches for controlling uncertainty in the values of model parameters. In this respect they are both superior to SA. With stochastic linear programming models the decision maker is afforded the flexibility of recourse variables. These correspond to the control variables of RO and provide the mechanism with which the model recommendations

can be adjusted to account for the realizations of random data.

The SLP model, however, optimizes only the first moment of the distribution of the objective value $\xi(\cdot)$. It ignores higher moments of the distribution, and the decision maker's preferences towards risk. These aspects are particularly important for asymmetric distributions, and for risk averse decision makers. Furthermore, aiming at expected value optimization implicitly assumes an active management style whereby the control (i. e., recourse) variables are easily adjusted as scenarios unfold. Large changes in the objective values ξ_s may be observed across different scenarios, but their expected value will be minimal. The RO model minimizes higher moments as well, e. g., the variance of the distribution of $\xi(\cdot)$. Hence, it assumes a more passive management style. Since the value of ξ_s will not differ substantially among different scenarios, little or no adjustment of the control variables will be needed. In this respect RO can be viewed as a SLP whereby the recourse decisions are implicitly restricted.

This distinction between RO and SLP is important, and defines their domain of applicability. Applied to personnel planning, for example, a SLP solution will design a workforce that can be adjusted (by hirings or layoffs) to meet demand at the least expected cost. The important consideration of maintaining stability of employment can not be captured. The RO model, on the other hand, will design a workforce that will need few adjustments in order to cope with demand for all scenarios. However, the cost of the RO solution will be higher than the cost of the SLP solution.

Another important distinction of RO from SLP is the handling of the constraints. Stochastic linear programming models aim at finding the design variables x such that for each realized scenario s a control variable setting y^s is possible that satisfies the control constraints. For systems with some redundancy such a solution might always be possible. The SLP literature even allows for the notion of *relatively complete recourse*, whereby a feasible solution y^s exists for all scenarios, and for any value of x that satisfies the design constraints. What happens in cases where no feasible pair (x, y^s) is possible for every scenario? The SLP model is declared infeasible. RO explicitly accounts for this possibility. In engineering applications (e. g., image restoration) such situations inevitably arise due to measurement errors. Multiple measurements of the same quantity may be inconsistent with each other. Hence, even if the underlying physical system has a solution (in this case an image does exist!) it will not satisfy all the measurements. The RO model, through the use of error terms $\{z^s\}$ and the infeasibility penalty function $\rho(\cdot)$, will find a solution that violates the constraints by the least amount. This approach is receiving increasing attention in the operations research literature.

While RO has some distinct advantages over SA and SLP, it is not without limitations. First, RO models are *parametric programs* and we have no mechanism for specifying a priori a 'correct' choice of the parameter λ. Of course, this problem is prevalent in multicriteria programming optimization. Second, the scenarios in Ω are just one possible set of realizations of the problem data. RO does not provide a means by which the scenarios can be specified. This problem is prevalent in SLP models as well. Substantial progress has been made in recent years in integrating variance reduction methods, such as importance sampling, into stochastic linear programming, and these techniques also apply to RO.

Despite these potential shortcomings, we emphasize that working only with expected values (as in the linear programming formulations) is fundamentally limited for problems with noisy data. Even going a step further – that is, working with expected values and *hedging* against small changes in these values – is also inappropriate. In this respect, RO provides a significantly improved modeling framework.

Robust optimization integrates the methods of multi-objective programming with stochastic programming. It also extends SLP with the introduction of higher moments of the objective value, and with the notion of model robustness. Both RO and SLP provide constructive mechanisms for dealing with uncertain data.

Notes

The material in this article is drawn from [1, Chap. 13], where additional material and extensive discussion of the literature can be found.

The robust optimization model was suggested in [7], which also discussed example applications of robust optimization to the classic diet problem, and problems in finance, engineering design and capacity plan-

ning. J.K. Sengupta [10] had discussed earlier the robustness of stochastic linear programming solutions. Another approach for enforcing robustness in stochastic programs by means of restricted recourse models was introduced in [11].

The terminology of 'structural' and 'control' variables is borrowed from the flexibility analysis of manufacturing systems; see [9].

Applications of robust optimization are discussed in [5] (for environmental planning), [2] (for image reconstruction from projections), [8] (for capacity planning), [12] (for matrix balancing), [6] (for capacity expansion for power generation firms), [4] (for capacity expansion planning in manufacturing). A robust optimization approach to capacity planning for a multiproduct, multifacility production firm was suggested in [3], and applied to plan car manufacturing facilities for the General Motors Company.

See also

- ► Competitive Ratio for Portfolio Management
- ► Financial Applications of Multicriteria Analysis
- ► Financial Optimization
- ► Portfolio Selection and Multicriteria Analysis
- ► Semi-infinite Programming and Applications in Finance

References

1. Censor Y, Zenios SA (1997) Parallel optimization: Theory, algorithms, and applications. Numer Math Sci Comput. Oxford Univ Press, Oxford
2. Elfving T (1989) An algorithm for maximum entropy image reconstruction from noisy data. Math Comput Modelling 12:729–745
3. Eppen GD, Martin RK, Schrage L (1989) A scenario approach to capacity planning. Oper Res 37:517–527
4. Gutierrez GJ, Kouvelis P (1995) A robustness approach to international outsourcing. Ann Oper Res 59:165–193
5. King AJ, Rockafellar RT, Somlyódy L, Wets RJ-B (1988) Lake eutrophication management: The lake Balaton project. In: Ermoliev Yu, Wets RJ-B (eds) Numerical Techniques for Stochastic Optimization. Ser Comput Math. Springer, Berlin, pp 435–444
6. Malcolm S, Zenios SA (1994) Robust optimization for power capacity expansion planning. J Oper Res Soc 45(9):1040–1049
7. Mulvey JM, Vanderbei RJ, Zenios SA (1995) Robust optimization of large scale systems. Oper Res 43:264–281
8. Paraskevopoulos D, Karakitsos E, Rustem B (1991) Robust capacity planning under uncertainty. Managem Sci 37(7):787–800
9. Seider WD, Brengel DD, Widagdo S (1991) Nonlinear analysis in process design. AIChE J 37(1):1–38
10. Sengupta JK (1991) Robust solutions in stochastic linear programming. J Oper Res Soc 42(10):857–870
11. Vladimirou H, Zenios SA (1999) Scalable parallel computations for large-scale stochastic programming. Ann Oper Res 90:87–129
12. Zenios SA, Zenios SA (1992) Robust optimization for matrix balancing from noisy data. Report, Decision Sci Dept, Wharton School, Univ Pennsylvania, 01–02

Robust Optimization: Mixed-Integer Linear Programs

STACY L. JANAK, CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C11, 90C30, 90B35

Article Outline

Introduction
  A Motivating Example
Formulation
  Bounded Uncertainty
  Uncertainty with Known Probability Distribution
Applications
  Uncertainty in Processing Times
  Uncertainty in Product Demands
  Uncertainty in Market Prices
Cases
  Case 1: Bounded Uncertainty in the Processing Times
  Case 2: Uncertainty with a Normal Distribution in the Market Prices
Conclusions
References

Introduction

The research area of production scheduling has received considerable attention from both academia and the chemical processing industries over the past decade. Most of the work in the literature assumes that all data are deterministic - that is, they are of constant, known values. However, in reality, uncertainty is prevalent in

many scheduling problems due to lack of accurate process models and variability of process and environmental data. Thus, an emerging area of research aims at developing methods to address the problem of scheduling under uncertainty, in order to create reliable schedules which remain feasible in the presence of parameter uncertainty.

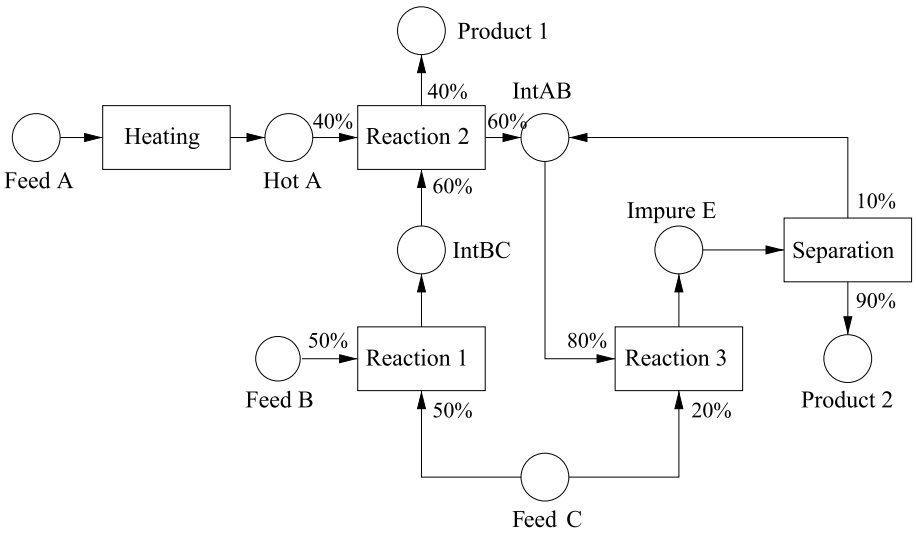
The issue of robustness in scheduling under uncertainty has received relatively little attention, in spite of its importance and the fact that there has been a substantial amount of work to address the problem of design and operation of batch plants under uncertainty. Most of the existing work has followed the scenario-based framework, in which the uncertainty is modeled through the use of a number of scenarios, using either discrete probability distributions or the discretization of continuous probability distribution functions, and the expectation of a certain performance criterion, such as the expected profit which is optimized with respect to the scheduling decision variables. The scenario-based approaches provide a straightforward way to implicitly incorporate uncertainty. However, they inevitably enlarge the size of the problem significantly as the number of scenarios increases exponentially with the number of uncertain parameters. This main drawback limits the application of these approaches to solve practical problems with a large number of uncertain parameters. An alternative approach for scheduling under uncertainty is reactive scheduling. It is carried out to adjust a schedule, which is usually obtained a priori in a deterministic manner, upon realization of the uncertain parameters or occurrence of unexpected events. Due to the "online" nature of reactive scheduling, it is required to generate updated schedules in a timely manner and often, heuristic approaches are developed for schedule modifications.

In this chapter, we propose a novel robust optimization approach to address the problem of scheduling under uncertainty. The underlying framework is based on a robust optimization methodology first introduced for Linear Programming (LP) problems by Ben-Tal and Nemirovski [1] and extended in this work for Mixed-Integer Linear Programming (MILP) problems. The approach produces "robust" solutions which are, in a sense, immune against uncertainties in both the coefficients and right-hand-side parameters of the inequality constraints. The approach can be applied to address the problem of production scheduling with uncertain processing times, market demands, and/or prices of products and raw materials. We consider uncertainty in scheduling problems which can be characterized as bounded or bounded and symmetric as well as uncertainty described by a known probability distribution function, such as a uniform or normal distribution. Additional work on the robust optimization of short-term scheduling problems can be found in Lin et al. [2] and Janak et al. [3].

A Motivating Example

Consider the following example process that was first presented by Kondili et al. [4] and has been widely studied in the literature. Two different products can be produced from three feeds according to the State-Task Network as shown in Fig. 1. Five processing tasks are considered including *Heating*, *Reaction 1*, *Reaction 2*, *Reaction 3*, and *Separation*. Four units are available and four intermediate materials are processed in-between tasks. The initial stock level for all intermediates and products is assumed to be zero. Each task has a unit-specific, variable processing time. The relevant data are shown in Table 1. The objective is to maximize the profit from the sale of products manufactured in a time horizon of 12 hours.

The continuous-time formulation proposed by Floudas and coworkers [5,6,7,8] is used to solve this simple scheduling problem. The "nominal" solution is shown in Fig. 2, which features intensive utilization of the two reactors and an objective function value (profit) of 3638.75. However, this solution can become completely infeasible when there is uncertainty in the processing times of the tasks. That is, when a task requires a longer processing time than its nominal value, it may not be able to finish processing within the time interval assigned in the nominal schedule. In this example, even a very small perturbation may make the schedule infeasible, especially for the two reactors, and can have a substantial effect on the scheduling decisions. For instance, if the processing time of each task is increased by simply 10% of its nominal value, then the nominal schedule will become infeasible and the optimal schedule with the slightly increased processing times will be significantly different from the nominal schedule, as shown in Fig. 3. In the *Heater* and the *Separator*, the number



Robust Optimization: Mixed-Integer Linear Programs, Figure 1
State-task network for the motivating example

Robust Optimization: Mixed-Integer Linear Programs, Table 1
Data for the motivating example

Units	Capacity	Suitability	Processing Time
Heater	100	heating	1.0
Reactor 1	50	reaction1,2,3	2.0,2.0,1.0
Reactor 2	80	reaction1,2,3	2.0,2.0,1.0
Separator	200	separation	2.0

States	Storage Capacity	Initial Amount	Price
Feed A	Unlimited	Unlimited	0.0
Feed B	Unlimited	Unlimited	0.0
Feed C	Unlimited	Unlimited	0.0
Hot A	100	0.0	0.0
IntAB	200	0.0	0.0
IntBC	150	0.0	0.0
ImpureE	200	0.0	0.0
Product 1	Unlimited	0.0	10.0
Product 2	Unlimited	0.0	10.0

of tasks as well as processing amounts change, while in *Reactor 1* and *Reactor 2*, even the task sequences are different. Furthermore, the profit is reduced to 3264.69.

It is clear that solving a scheduling problem at the nominal values of the uncertain data is not enough. To obtain reliable and efficient schedules, systematic and effective approaches which take into account uncertainty are required. In the rest of this chapter, we propose a new robust optimization framework to generate schedules that are reliable in the presence of uncertainty arising from various sources.

Formulation

Consider the following generic Mixed-Integer Linear Programming (MILP) problem

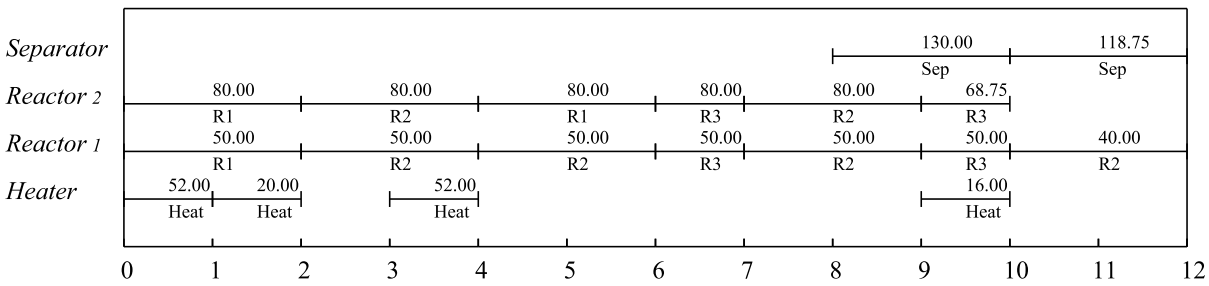
$$\begin{aligned} \underset{x,\,y}{\text{Min/Max}}\ & c^T x + d^T y \\ \text{s.t. } & Ex + Fy = e \\ & Ax + By \le p \\ & \underline{x} \le x \le \bar{x} \quad y = 0, 1\,. \end{aligned} \tag{1}$$

Assume that the uncertainty arises from both the coefficients and the right-hand-side parameters of the inequality constraints, namely, a_{lm}, b_{lk} and p_l. We are then concerned about the feasibility of the following inequality:

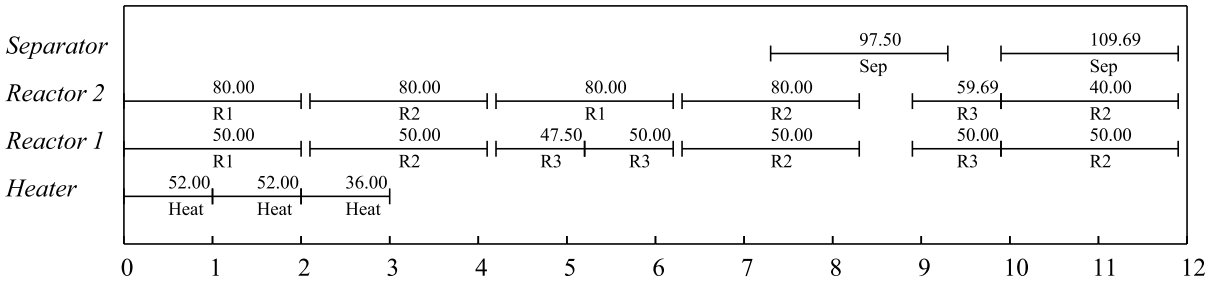
$$\sum_m \tilde{a}_{lm} x_m + \sum_k \tilde{b}_{lk} y_k \le \tilde{p}_l\,, \tag{2}$$

where a_{lm}, b_{lk}, and p_l are the nominal values of the uncertain parameters and $\tilde{a}_{lm}$, $\tilde{b}_{lk}$, and $\tilde{p}_l$ are the "true" values of the uncertain parameters.

As shown in the previous section with the motivating example on scheduling, the optimal solution of an



Robust Optimization: Mixed-Integer Linear Programs, Figure 2
Optimal solution with nominal processing times (profit = 3638.75)



Robust Optimization: Mixed-Integer Linear Programs, Figure 3
Optimal solution with processing times increased by 10% (profit = 3264.69)

MILP problem may become infeasible – that is, one or more constraints are violated substantially – if the nominal data is slightly perturbed. Our objective here is to develop a robust optimization methodology to generate "reliable" solutions to the MILP problem, which are immune against uncertainty. This robust optimization methodology was first introduced for Linear Programming (LP) problems with uncertain linear coefficients by Ben-Tal and Nemirovski [1] and is extended in this work to MILP problems under uncertainty. We consider several different types of uncertainty including: (i) bounded uncertainty, (ii) bounded and symmetric uncertainty, and (iii) uncertainty described by a known distribution such as a uniform or normal distribution.

Bounded Uncertainty

Suppose that the uncertain data range in the following intervals:

$$|\tilde{a}_{lm} - a_{lm}| \le \epsilon |a_{lm}|,\ |\tilde{b}_{lk} - b_{lk}| \le \epsilon |b_{lk}|,$$
$$|\tilde{p}_l - p_l| \le \epsilon |p_l|\,, \quad (3)$$

where $\tilde{a}_{lm}$, $\tilde{b}_{lk}$ and $\tilde{p}_l$ are the "true" values, a_{lm}, b_{lk} and p_l are the nominal values, and $\epsilon > 0$ is a given (relative) uncertainty level.

We call a solution (x,y) robust if it satisfies the following conditions: (i) (x,y) is feasible for the nominal problem, and (ii) whatever are the true values of the coefficients and right-hand-side parameters within the corresponding intervals, (x,y) must satisfy the lth inequality constraint with an error of at most $\delta \cdot \max[1, |p_l|]$, where δ is a given infeasibility tolerance.

Theorem 1 *Given an infeasibility tolerance, (δ), to generate robust solutions, the following so-called (ϵ,δ)-Interval Robust Counterpart (IRC[ϵ,δ]) of the original uncertain MILP problem can be derived.*

$$\begin{aligned}
&\underset{x,\,y,\,u}{\text{Min/Max}}\ c^T x + d^T y \\
&\quad \text{s.t. } Ex + Fy = e \\
&\qquad Ax + By \le p \\
&\qquad \sum_m a_{lm} x_m + \epsilon \sum_{m \in M_l} |a_{lm}| u_m + \sum_k b_{lk} y_k \\
&\qquad\quad + \epsilon \sum_{k \in K_l} |b_{lk}| y_k \\
&\qquad \le p_l - \epsilon |p_l| + \delta \cdot \max[1, |p_l|], \quad \forall l
\end{aligned}$$

$$
\begin{aligned}
& -u_m \le x_m \le u_m, \quad \forall m \\
& \underline{x} \le x \le \bar{x} \\
& y_k = 0, 1, \quad \forall k
\end{aligned} \tag{4}
$$

where M_l and K_l are the set of indices of the x and y variables, respectively, with uncertain coefficients in the lth inequality constraint.

Proof 1 We want to find a robust solution (x,y) which satisfies condition (i) and condition (ii), that is:

$$
\begin{aligned}
& \forall l, \quad \forall (\tilde{a}_{lm}: |\tilde{a}_{lm} - a_{lm}| \le \epsilon |a_{lm}|, \tilde{b}_{lk}: \\
& \quad |\tilde{b}_{lk} - b_{lk}| \le \epsilon |b_{lk}|, \text{ and } \tilde{p}_l: |\tilde{p}_l - p_l| \le \epsilon |p_l|): \\
& \quad \sum_{m \notin M_l} a_{lm} x_m + \sum_{m \in M_l} \tilde{a}_{lm} x_m \\
& \quad + \sum_{k \notin K_l} b_{lk} y_k + \sum_{k \in K_l} \tilde{b}_{lk} y_k \\
& \quad \le \tilde{p}_l + \delta \cdot \max[1, |p_l|] \, .
\end{aligned} \tag{5}
$$

Using the worst-case values of the uncertain parameters, or those that make the inequality constraint the most difficult to satisfy,

$$
\begin{aligned}
& \tilde{a}_{lm} x_m \le a_{lm} x_m + \epsilon |a_{lm}||x_m|, \\
& \tilde{b}_{lk} y_k \le b_{lk} y_k + \epsilon |b_{lk}| y_k, \\
& \text{and} \quad \tilde{p}_l \ge p_l - \epsilon |p_l|
\end{aligned} \tag{6}
$$

and substituting into Eq. (5) and rearranging terms, it is clear that a solution (x,y) is robust if and only if it is a feasible solution of the following optimization problem.

$$
\begin{aligned}
\underset{x,\, y}{\text{Min/Max}} \quad & c^T x + d^T y \\
\text{s.t.} \quad & Ex + Fy = e \\
& Ax + By \le p \\
& \sum_m a_{lm} x_m + \epsilon \sum_{m \in M_l} |a_{lm}||x_m| \\
& \quad + \sum_k b_{lk} y_k + \epsilon \sum_{k \in K_l} |b_{lk}| y_k \\
& \le p_l - \epsilon |p_l| + \delta \cdot \max[1, |p_l|], \quad \forall l \\
& \underline{x} \le x \le \bar{x} \\
& y_k = 0, 1, \quad \forall k \, .
\end{aligned} \tag{7}
$$

The above problem is equivalent to problem (4) where the absolute value operator is represented with a set of auxiliary variables (u_m) and a set of additional constraints. □

For each inequality constraint that involves uncertain coefficients and/or right-hand-side parameters, an additional constraint is introduced to incorporate the uncertainty and maintain the relationships among the relevant binary and continuous variables under the uncertainty level and the given infeasibility tolerance. Essentially, this constraint considers the worst-case values of the uncertain parameters which make the inequality the most difficult to maintain. At the same time, a certain degree of relaxation is introduced to allow tolerable violations of the constraint. Note that mathematical model (4) remains an MILP model. Compared to the original deterministic MILP problem, the robust counterpart has a set of auxiliary variables (u_m) and a set of additional constraints relating the variables x_m and u_m.

Uncertainty with Known Probability Distribution

Assume that for the inequality constraint l, the true values of the uncertain parameters are obtained from their nominal values by the random perturbations

$$
\begin{aligned}
\tilde{a}_{lm} &= (1 + \epsilon \xi_{lm}) a_{lm} \\
\tilde{b}_{lk} &= (1 + \epsilon \xi_{lk}) b_{lk} \\
\tilde{p}_l &= (1 + \epsilon \xi_l) p_l
\end{aligned} \tag{8}
$$

where ξ_{lm}, ξ_{lk} and ξ_l are independent random variables and $\epsilon > 0$ is a given (relative) uncertainty level.

In this situation, we call a solution (x,y) robust if it satisfies the following: (i) (x,y) is feasible for the nominal problem, and (ii) for every inequality l, the probability of violation of the uncertain inequality is at most κ, or

$$
Pr\left\{ \sum_m \tilde{a}_{lm} x_m + \sum_k \tilde{b}_{lk} y_k > \tilde{p}_l + \delta \cdot \max[1, |p_l|] \right\} \le \kappa \tag{9}
$$

where $\delta > 0$ is a given feasibility tolerance and $\kappa > 0$ is a given reliability level. Thus, κ represents the probability of violation of constraint l where $\kappa = 0\%$ indicates that there is no chance of constraint violation, yielding the most conservative solution.

If the probability distributions of the random variables ξ_{lm}, ξ_{lk} and ξ_l in the uncertain parameters are

known, it is possible to obtain a more accurate estimation of the probability measures involved. The MILP from (1) can be rewritten as an uncertain MILP as follows

$$\begin{aligned} \underset{x,\,y}{\text{Min}} \;& c^T x + d^T y \\ \text{s.t.} \;& Ex + Fy = e \\ & \underset{\xi \in Z}{\text{Max}} \left[\tilde{A}x + \tilde{B}y - \tilde{p} \le \delta \cdot \max[1, |p|] \right] \\ & \underline{x} \le x \le \bar{x} \\ & y = 0, 1 \\ & \xi \equiv [A, B, p] \in Z \end{aligned} \tag{10}$$

where the data set $\xi \equiv [A, B, p]$ varies in a given uncertainty set Z, $\tilde{A}$, $\tilde{B}$, and $\tilde{p}$ represent the "true" values of the uncertain coefficients, and $\delta \ge 0$ is an infeasibility tolerance introduced to allow a certain amount of infeasibility into the inequality constraint. The inequality can be written in expanded form as

$$\underset{\xi \in Z}{\text{Max}} \left[\sum_m \tilde{a}_{lm} x_m + \sum_k \tilde{b}_{lk} y_k - \tilde{p}_l \le \delta \cdot \max\left[1, |p_l|\right] \right] \tag{11}$$

for every constraint l where $\tilde{a}_{lm}$, $\tilde{b}_{lk}$, and $\tilde{p}_l$ are again the true values of the uncertain coefficients. Using the expressions for the true values of the uncertain coefficients given in constraint (8), the uncertain inequality in (11) can be rewritten as follows

$$\begin{aligned} \underset{\xi \in Z}{\text{Max}} \Big[& \sum_m (1 + \epsilon \xi_{lm}) a_{lm} x_m + \sum_k (1 + \epsilon \xi_{lk}) b_{lk} y_k \\ & - (1 + \epsilon \xi_l) p_l \le \delta \cdot \max\left[1, |p_l|\right] \Big] . \end{aligned} \tag{12}$$

Rearranging terms, we get

$$\begin{aligned} \underset{\xi \in Z}{\text{Max}} \Bigg[& \sum_m a_{lm} x_m + \sum_k b_{lk} y_k - p_l + \epsilon \\ & \left(\sum_{m \in M_l} \xi_{lm} a_{lm} x_m + \sum_{k \in K_l} \xi_{lk} b_{lk} y_k - \xi_l p_l \right) \\ & \le \delta \cdot \max\left[1, |p_l|\right] \Bigg] \end{aligned} \tag{13}$$

where M_l and K_l define the sets of uncertain parameters a_{lm} and b_{lk}, respectively, for constraint l. Then, a solution (x,y) to the original uncertain MILP given in Eq. (10) which satisfies this constraint is called "reliable" because it takes into account the maximum amount of uncertainty $\xi \in Z$ and allows an amount of infeasibility δ. Now, to transform the constraint into a deterministic form, we instead consider the following formulation

$$\begin{aligned} Pr \Bigg\{ & \sum_m a_{lm} x_m + \sum_k b_{lk} y_k - p_l + \epsilon \\ & \left(\sum_{m \in M_l} \xi_{lm} a_{lm} x_m + \sum_{k \in K_l} \xi_{lk} b_{lk} y_k - \xi_l p_l \right) \\ & > \delta \cdot \max\left[1, |p_l|\right] \Bigg\} \le \kappa . \end{aligned} \tag{14}$$

This constraint enforces that the probability of violation of the uncertain inequality is at most κ, where $\delta \ge 0$ is a given feasibility tolerance (i. e., amount of error allowed in the feasibility of constraint l) and $\kappa \ge 0$ is a given reliability level (i. e., the probability of violation of constraint l where $\kappa = 0$ indicates that there is no chance of constraint violation). Thus, if we know a probability distribution function for the sum of the random variables,

$$\xi = \sum_{m \in M_l} \xi_{lm} a_{lm} x_m + \sum_{k \in K_l} \xi_{lk} b_{lk} y_k - \xi_l p_l \tag{15}$$

we can use this information in the probabilistic constraint (14) to write a deterministic form for the uncertain constraint which is "almost reliable", depending on the value of κ. This is done using the definition of a probability distribution function and the following relationship

$$F_\xi(\lambda) = Pr\{\xi \le \lambda\} = 1 - Pr\{\xi > \lambda\} = 1 - \kappa \tag{16}$$

to replace the stochastic elements in constraint (13), generating a deterministic constraint that is "almost reliable" for the given uncertainty level, ϵ, infeasibility tolerance, δ, and reliability level, κ. The final form of the deterministic constraint (or robust counterpart problem) is simply determined using the inverse distribution function (quantile) of the random variable ξ

$$F_\xi^{-1}(1 - \kappa) = f(\lambda, |a_{lm}| x_m, |b_{lk}| y_k, |p_l|) . \tag{17}$$

Thus, the additional constraints in the Robust Counterpart (RC) problem can be written as

$$\sum_m a_{lm}x_m + \sum_k b_{lk}y_k + \epsilon f(\lambda, |a_{lm}|x_m, |b_{lk}|y_k, |p_l|) \le p_l + \delta \cdot \max[1, |p_l|], \quad \forall l \tag{18}$$

where ϵ is a given uncertainty level, δ is a given infeasibility tolerance, and λ is determined from κ using constraint (16) and the probability distribution function for ξ.

Uncertainty with Normal Probability Distribution Suppose that the distributions of the random variables ξ_{lm}, ξ_{lk} and ξ_l in (8) are all standardized normal distributions with zero as the mean and one as the standard deviation. Then, the distribution of ξ defined in (15) is also a normal distribution, with zero as the mean and $\sqrt{\sum_{m\in M_l} a_{lm}^2 x_m^2 + \sum_{k\in K_l} b_{lk}^2 y_k + p_l^2}$ as the standard deviation.

Theorem 2 *Given an uncertainty level (ϵ), an infeasibility tolerance (δ), and a reliability level (κ), to generate robust solutions, the following (ϵ,δ,κ)-Robust Counterpart (RC[ϵ,δ,κ]) of the original uncertain MILP problem can be derived.*

$$\begin{aligned}
\underset{x,y}{\text{Min/Max}} \quad & c^T x + d^T y \\
\text{s.t.} \quad & Ex + Fy = e \\
& Ax + By \le p \\
& \sum_m a_{lm}x_m + \sum_k b_{lk}y_k + \epsilon\lambda \\
& \quad \sqrt{\sum_{m\in M_l} a_{lm}^2 x_m^2 + \sum_{k\in K_l} b_{lk}^2 y_k + p_l^2} \\
& \quad \le p_l + \delta \cdot \max[1, |p_l|], \quad \forall l \\
& \underline{x} \le x \le \bar{x} \\
& y_k = 0, 1, \quad \forall k
\end{aligned} \tag{19}$$

where $\lambda = F_n^{-1}(1-\kappa)$ and F_n^{-1} is the inverse distribution function of a random variable with standardized normal distribution. Thus, λ and κ are related as follows

$$\kappa = 1 - F_n(\lambda)$$
$$\kappa = 1 - Pr\{\xi \le \lambda\}$$
$$\kappa = 1 - \int_{-\infty}^{\lambda} \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right) dx$$

where ξ is a random variable with standardized normal distribution.

Proof 2 Let (x,y) satisfy the following

$$\sum_m a_{lm}x_m + \sum_k b_{lk}y_k + \epsilon\lambda \sqrt{\sum_{m\in M_l} a_{lm}^2 x_m^2 + \sum_{k\in K_l} b_{lk}^2 y_k + p_l^2} \le p_l + \delta \cdot \max[1, |p_l|] \tag{20}$$

where $\lambda = F_n^{-1}(1-\kappa)$ and F_n^{-1} is the inverse distribution function of a random variable with standardized normal distribution. Then

$$\begin{aligned}
& Pr\left\{\sum_m \tilde{a}_{lm}x_m + \sum_k \tilde{b}_{lk}y_k > \tilde{p}_l + \delta \cdot \max[1, |p_l|]\right\} \\
& = Pr\Bigg\{\sum_m a_{lm}x_m + \epsilon \sum_{m\in M_l} \xi_{lm}|a_{lm}|x_m \\
& \quad + \sum_k b_{lk}y_k + \epsilon \sum_{k\in K_l} \xi_{lk}|b_{lk}|y_k \\
& \quad > p_l + \epsilon\xi_l|p_l| + \delta \cdot \max[1, |p_l|]\Bigg\} \\
& \le Pr\Bigg\{\Bigg(\sum_{m\in M_l} \xi_{lm}|a_{lm}|x_m \\
& \quad + \sum_{k\in K_l} \xi_{lk}|b_{lk}|y_k - \xi_l|p_l|\Bigg) / \\
& \quad \sqrt{\sum_{m\in M_l} a_{lm}^2 x_m^2 + \sum_{k\in K_l} b_{lk}^2 y_k + p_l^2} > \lambda\Bigg\} \\
& = 1 - Pr\Bigg\{\Bigg(\sum_{m\in M_l} \xi_{lm}|a_{lm}|x_m \\
& \quad + \sum_{k\in K_l} \xi_{lk}|b_{lk}|y_k - \xi_l|p_l|\Bigg) / \\
& \quad \sqrt{\sum_{m\in M_l} a_{lm}^2 x_m^2 + \sum_{k\in K_l} b_{lk}^2 y_k + p_l^2} \le \lambda\Bigg\} \\
& = 1 - F_n(\lambda) = 1 - (1-\kappa) = \kappa\,. \qquad \square
\end{aligned}$$

Note that $(\sum_{m\in M_l} \xi_{lm}|a_{lm}|x_m + \sum_{k\in K_l} \xi_{lk}|b_{lk}|y_k - \xi_l|p_l|)/\sqrt{\sum_{m\in M_l} a_{lm}^2 x_m^2 + \sum_{k\in K_l} b_{lk}^2 y_k + p_l^2}$ is also a ran-

dom variable with standardized normal distribution. This formulation results in a convex MINLP problem, but can still be solved efficiently using a mixed-integer nonlinear solver (e. g., DICOPT [9], MINOPT [10]).

In the discussion above, for simplicity, we have assumed that there is a single common uncertainty level (ϵ), infeasibility tolerance (δ), and reliability level (κ) in each MILP or convex MINLP problem with uncertain parameters. However, the proposed robust optimization techniques can easily be extended to account for the more general case in which the uncertainty level varies from one parameter to another and the infeasibility tolerance and reliability level are dependent on the constraint of interest. Furthermore, note that for each type of uncertainty addressed, one additional constraint is introduced for each inequality constraint with uncertain parameter(s) and auxiliary variables are added if needed. Because the transformation is carried out at the level of constraints, in principle, the various robust optimization techniques presented can be applied to a single MILP or convex MINLP problem involving different types of uncertainties. More specifically, for each inequality constraint, as long as all of its uncertain parameters are of the same type, an additional constraint that corresponds to the uncertainty type can be introduced to obtain the deterministic robust counterpart problem. It should be pointed out that the aforementioned robust optimization methodology circumvents any need for explicit or implicit discretization or sampling of the uncertain data, avoiding an undesirable increase in the size of the problem. Thus, the proposed methodology is potentially capable of handling problems with a large number of uncertain parameters.

Applications

The robust optimization methodology proposed in the previous section can be applied to address the problem of scheduling under uncertainty. In this work, we employ the continuous-time formulation presented by Floudas and coworkers [5,6,7,8], which leads to MILP models, to develop new robust scheduling approaches for the following three classes of uncertainties: (i) uncertainty in processing times/rates of tasks, (ii) uncertainty in market demands for products, and (iii) uncertainty in market prices of products and raw materials.

Uncertainty in Processing Times

The parameters of processing times/rates of tasks participate in the duration constraint and appear as linear coefficients of the binary variable (i. e., α_{ij}) and the continuous variable (i. e., β_{ij}) as follows:

$$T^f(i,j,n) - T^s(i,j,n) = \alpha_{ij} \cdot wv(i,n) + \beta_{ij} \cdot B(i,j,n)\,, \tag{21}$$

where $wv(i,n)$ is a binary variable indicating whether or not task (i) starts at event point (n), $B(i,j,n)$ is a continuous variable determining the batch-size of the task, and $T^s(i,j,n)$ and $T^f(i,j,n)$ are continuous variables representing the starting and finishing time of the task, respectively. Note that this is an equality constraint. Thus, in order to apply the robust optimization techniques proposed in the previous section for inequality constraints with uncertain parameters, the duration constraint is relaxed to an inequality constraint

$$T^f(i,j,n) - T^s(i,j,n) \geq \alpha_{ij} \cdot wv(i,n) + \beta_{ij} \cdot B(i,j,n)\,. \tag{22}$$

Consequently, the variable $T^f(i,j,n)$ represents the lower bound on the finishing time of the task, instead of the exact finishing time as determined by the original duration constraint. Using this modified duration constraint, the various robust optimization techniques can be readily applied to consider uncertainty in the parameters α_{ij} and β_{ij}.

For example, consider a task with parameters α_{ij} and β_{ij} exhibiting bounded uncertainty in the following ranges

$$\alpha_{ij}^L \leq \tilde{\alpha}_{ij} \leq \alpha_{ij}^U,\ \beta_{ij}^L \leq \tilde{\beta}_{ij} \leq \beta_{ij}^U\,. \tag{23}$$

According to Theorem 1, to obtain the deterministic robust counterpart problem, the following constraint is added to the original scheduling model

$$T^f(i,j,n) - T^s(i,j,n) \geq \alpha_{ij}^U \cdot wv(i,n) + \beta_{ij}^U \cdot B(i,j,n) - \delta. \tag{24}$$

Note that no auxiliary variables need to be introduced because the variable $B(i,j,n)$ (batch-size of the task) is non-negative by definition.

Alternatively, consider a batch task with fixed processing time represented by parameter α_{ij}. Then, the

true value of the processing time can be represented in terms of the nominal processing time as follows:

$$\tilde{\alpha}_{ij} = (1 + \epsilon\xi_{\alpha_{ij}})\alpha_{ij}\,, \tag{25}$$

where $\xi_{\alpha_{ij}}$ is a random variable with known distribution.

For the case where the uncertainty is characterized by a standardized normal distribution, then, according to Theorem 2, to obtain the deterministic robust counterpart problem, the following constraint is added to the original scheduling model:

$$T^s(i,j,n) - T^f(i,j,n) + [1+\epsilon\lambda]\alpha_{ij}\cdot wv(i,n) \le \delta\,, \tag{26}$$

where $\lambda = F^{-1}(1-\kappa)$ and F_n^{-1} is the inverse distribution function of a random variable with a standardized normal distribution.

Uncertainty in Product Demands

The product demands (i. e., dem_s) appear as the right-hand-side parameters in the demand constraints

$$STF(s) \ge dem_s, \quad \forall s \in S^p \tag{27}$$

where $STF(s)$ is a continuous variable representing the amount of state (s) accumulated at the end of the time horizon and S^p is the set of final products.

The robust optimization techniques can be directly applied to these inequality constraints with uncertain parameters. For example, in the case of bounded uncertainty,

$$dem_s^L \le \tilde{dem}_s \le dem_s^U\,, \tag{28}$$

and according to Theorem 1, the constraint to be added to the original scheduling model to derive the deterministic robust counterpart problem is as follows:

$$STF(s) \ge dem_s^U - \delta\,. \tag{29}$$

Alternatively, for case of uncertainty with a known distribution, if we consider an uncertain product demand represented by parameter dem_s, then the true value of the product demand can be represented in terms of the nominal product demand as follows:

$$\tilde{dem}_s = (1 + \epsilon\xi_s)dem_s\,, \tag{30}$$

where ξ_s is a random variable with known distribution.

For the case of normal uncertainty, according to Theorem 2, the constraint to be added to the original scheduling model to derive the deterministic robust counterpart problem is

$$STF(s) \ge dem_s(1 + \epsilon\lambda - \delta) \tag{31}$$

where $\lambda = F_n^{-1}(1-\kappa)$ and F_n^{-1} is the inverse distribution function of a random variable with standardized normal distribution.

Uncertainty in Market Prices

The market prices (i. e., price_s) participate in the objective function for the calculation of the overall profit:

$$\begin{aligned}\text{Maximize Profit} = &\sum_{s\in S^p} \text{price}_s \cdot STF(s)\\ &- \sum_{s\in S^r} \text{price}_s \cdot STO(s)\,,\end{aligned} \tag{32}$$

where S^p and S^r are the sets of final products and raw materials, respectively, and $STO(s)$ and $STF(s)$ are continuous variables representing the initial amount of state (s) at the beginning and the final amount of state (s) at the end, respectively. The objective function can be expressed in an equivalent way as follows:

$$\begin{aligned}&\text{Maximize Profit}\\ &\text{s.t.Profit} \le \sum_{s\in S^p} \text{price}_s \cdot STF(s)\\ &\qquad\qquad - \sum_{s\in S^r} \text{price}_s \cdot STO(s)\,.\end{aligned} \tag{33}$$

Now the uncertain parameters price>$_s$ appear as linear coefficients multiplying the continuous variables $STF(s)$ and $STO(s)$ in an inequality constraint and the robust optimization techniques can be readily applied.

For example, if the uncertainty is normally distributed,

$$\tilde{\text{price}}_s = (1 + \epsilon\xi_s)\text{price}_s \tag{34}$$

where ξ_s is a standardized normal random variable, then according to Theorem 2 the deterministic robust counterpart problem can be obtained by introducing the following constraint to the original scheduling

model:

$$\text{Profit} \le \sum_{s\in S^p} \text{price}_s \cdot STF(s) - \sum_{s\in S^r} \text{price}_s \cdot STO(s) - \epsilon\lambda \sqrt{\sum_{s\in S^p} \text{price}_s^2 \cdot STF(s)^2 + \sum_{s\in S^r} \text{price}_s^2 \cdot STO(s)^2} + \delta\,, \tag{35}$$

where $\lambda = F^{-1}(1-\kappa)$ and F_n^{-1} is the inverse distribution function of a random variable with standardized normal distribution.

Cases

In this section, the robust optimization formulation is applied to two example problems. Both the examples are implemented with GAMS [11] on a 3.20 GHz Linux workstation. The MILP problems are solved using CPLEX 8.1 while the MINLP problems are solved using DICOPT [9].

Case 1: Bounded Uncertainty in the Processing Times

Let us revisit the motivating example in Sect. "A Motivating Example". Assume that the uncertainty of the processing times is bounded and the (relative) uncertainty level (ϵ) is 15%, that is,

$$0.85\alpha \le \tilde{\alpha} \le 1.15\alpha \tag{36}$$

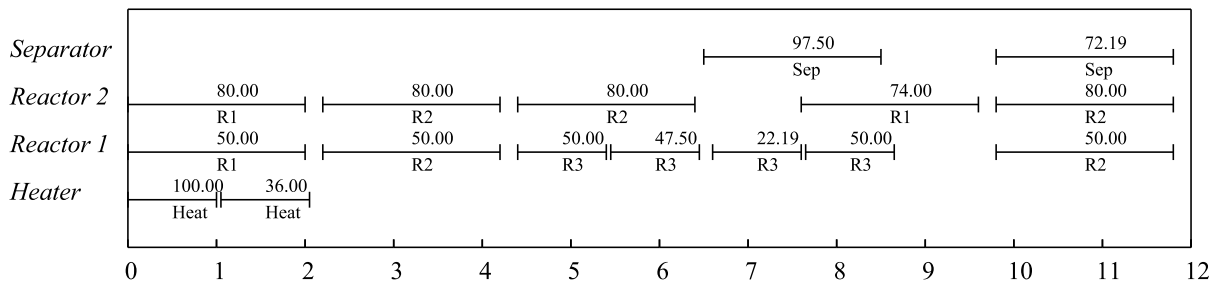
and the infeasibility tolerance level (δ) is 10%.

By solving the IRC[ϵ, δ] problem, a "robust" schedule is obtained, as shown in Fig. 4, which takes into account uncertainty in the processing times. The nominal schedule can be seen in Fig. 2 in Sect. "A Motivating Example". Compared to the nominal solution which is obtained at the nominal values of the processing times, the robust solution exhibits very different scheduling strategies, including both task-unit assignments and task timings. Even the sequences of tasks in the two reactors in Fig. 4 deviates significantly from those in the nominal solution in Fig. 2. The robust solution ensures that the robust schedule obtained is feasible with the specified uncertainty level and infeasibility tolerance. However, the resulting profit is reduced, from 3638.75 to 2887.19, which reflects the effect of uncertainty on overall production. A comparison of the model and solution statistics for the nominal and robust solutions can be found in Table 2.

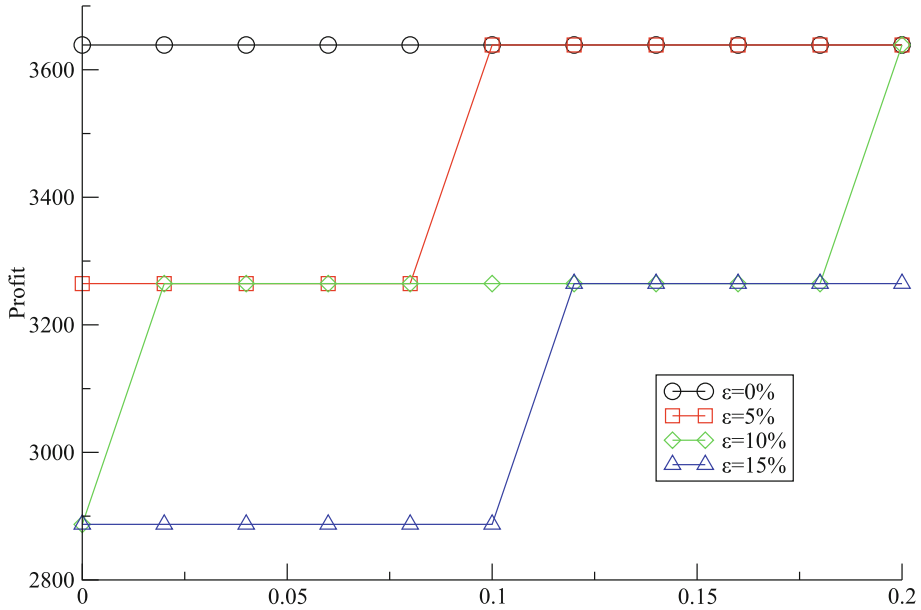
Robust Optimization: Mixed-Integer Linear Programs, Table 2
Model and solution statistics of case 1

	Nominal solution	Robust solution
Profit	3638.75	2887.19
CPU time (s)	2.68	114.47
Binary Variables	96	96
Continuous Variables	378	378
Constraints	553	713

Figure 5 summarizes the results of the IRC problem with three different levels of uncertainty. It is shown that with a given infeasibility tolerance, the maximal profit that can be achieved decreases as the uncertainty level increases, which indicates more "conservative" scheduling decisions because of the existence of uncertainty. On the other hand, at a given uncertainty level, the profit increases as the infeasibility tolerance is increased, which means more "aggressive" schedul-



Robust Optimization: Mixed-Integer Linear Programs, Figure 4
Robust solution for case 1 (ϵ = 15%, δ = 10%, profit = 2887.19)

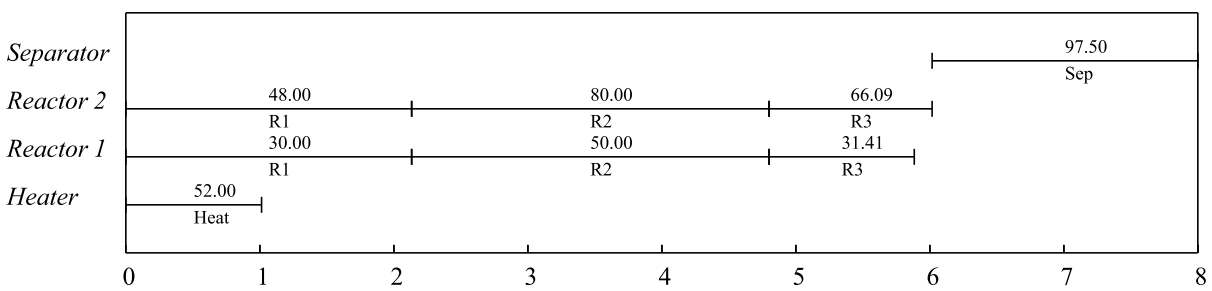


Robust Optimization: Mixed-Integer Linear Programs, Figure 5
Profit vs. infeasibility tolerance at different uncertainty levels for case 1

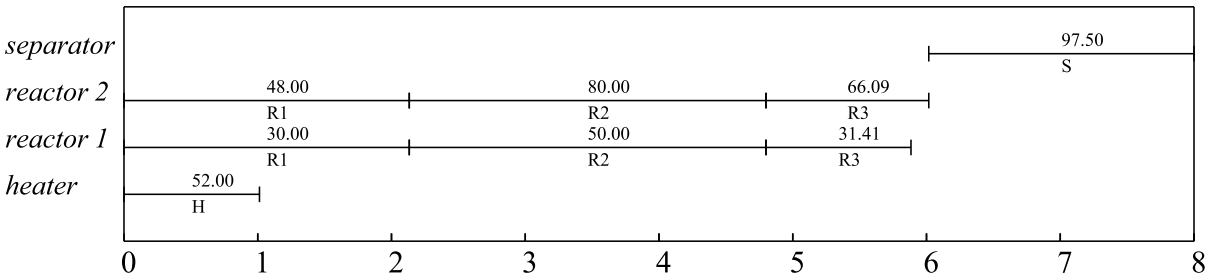
ing arrangements can be incorporated if violations of related timing constraints can be tolerated to a larger extent. These results are consistent with intuition and other approaches, however, with the robust optimization approach, the effects of uncertainty and the trade-offs between conflicting objectives are quantified rigorously and efficiently. It should be noted that at a given uncertainty level, the objective value of profit as well as the corresponding schedule change dramatically at discrete points as the infeasibility tolerance increases. This behavior is caused by special characteristics of the example problem, including the fixed time horizon and the fixed processing times of tasks.

Case 2: Uncertainty with a Normal Distribution in the Market Prices

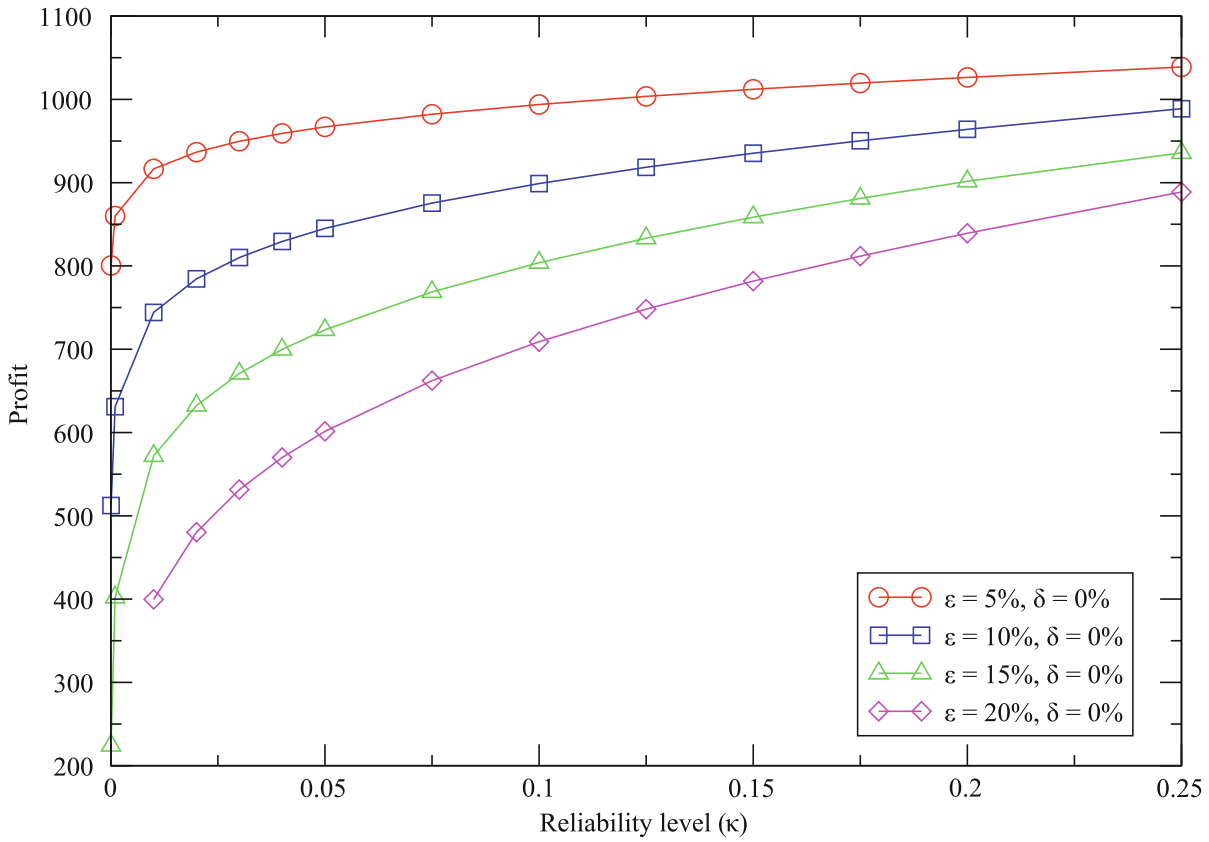
In this example, we consider uncertainty with a normal distribution in the market prices for the same process and processing time data given in Case 1. The objective function is the maximization of profit in a time horizon of 8 hours. The uncertainty level (ϵ) is 5%, the infeasibility tolerance (δ) is 5%, and the reliability level (κ) is 5%. The nominal schedule is shown in Fig. 6 with a profit of 1088.75. The robust schedule is obtained by solving the robust counterpart problem, as shown in Fig. 7, and the corresponding profit is 966.97. By



Robust Optimization: Mixed-Integer Linear Programs, Figure 6
Nominal solution for case 4 (profit = 1088.75)



Robust Optimization: Mixed-Integer Linear Programs, Figure 7
Robust solution for case 2 ($\epsilon = 5\%$, $\delta = 5\%$, $\kappa = 5\%$, profit = 966.97)



Robust Optimization: Mixed-Integer Linear Programs, Figure 8
Profit vs. reliability level at different uncertainty and infeasibility levels for case 2

executing this schedule, the profit is guaranteed to be least 966.97 with a probability of 95% in the presence of the 5% uncertainty in the prices of the products and raw materials. A comparison of the model and solution statistics for the nominal and robust solutions can be found in Table 3.

Figure 8 summarizes the results of the RC problem at several different levels of uncertainty and an infeasibility tolerance of 0% at increasing values of the reliability level. It is shown that at a given reliability level, the maximal profit that can be achieved decreases as the uncertainty level increases, which indicates more con-

Robust Optimization: Mixed-Integer Linear Programs, Table 3
Model and solution statistics for case 2

	Nominal solution	Robust solution
Profit	1088.75	966.97
CPU time (s)	0.02	0.05
Binary Variables	60	60
Continuous Variables	280	280
Constraints	334	334

servative scheduling decisions because of the existence of uncertainty. Also, at a given uncertainty level and infeasibility tolerance, the profit increases as the reliability level increases, meaning that as the probability of violation of the uncertain constraint, or κ, increases, then λ decreases and according to Eq. (35), the profit takes on a larger value.

Conclusions

In this chapter, we propose a new approach to address the scheduling under uncertainty problem based on a robust optimization methodology, which when applied to MILP problems, produces "robust" solutions that are, in a sense, immune against uncertainties in both the coefficients in the objective function, the left-hand-side parameters and the right-hand-side parameters of the inequality constraints. A unique feature of the proposed approach is that it can address many uncertain parameters. The approach can be applied to address the problem of production scheduling with uncertain processing times, market demands, and/or prices of products and raw materials. Our computational results show that this approach provides an effective way to address scheduling problems under uncertainty, producing reliable schedules and generating helpful insights on the tradeoffs between conflicting objectives.

References

1. Ben-Tal A, Nemirovski A (2000) Robust Solutions of Linear Programming Problems Contaminated with Uncertain Data. Math Program 88:411
2. Lin X, Janak SL, Floudas CA (2004) A New Robust Optimization Approach for Scheduling under Uncertainty: I. Bounded Uncertainty. Comp Chem Eng 28:1069
3. Janak SL, Lin X, Floudas CA (2007) A New Robust Optimization Approach for Scheduling under Uncertainty: I. Uncertainty with Known Probability Distribution. Comp Chem Eng 31:171
4. Kondili E, Pantelides CC, Sargent RWH (1993) A General Algorithm for Short-Term Scheduling of Batch Operations – I. MILP Formulation. Comp Chem Eng 17:211
5. Ierapetritou MG, Floudas CA (1998) Effective Continuous-Time Formulation for Short-Term Scheduling: 1. Multipurpose Batch Processes. Ind Eng Chem Res 37:4341
6. Ierapetritou MG, Floudas CA (1998) Effective Continuous-Time Formulation for Short-Term Scheduling: 2. Continuous and Semi-continuous Processes. Ind Eng Chem Res 37:4360
7. Ierapetritou MG, Hené TS, Floudas CA (1999) Effective Continuous-Time Formulation for Short-Term Scheduling: 3. Multiple Intermediate Due Dates. Ind Eng Chem Res 38:3446
8. Lin X, Floudas CA (2001) Design, Synthesis and Scheduling of Multipurpose Batch Plants via an Effective Continuous-Time Formulation. Comp Chem Eng 25:665
9. Viswanathan J, Grossmann IE (1990) DICOPT ++: A Program for Mixed Integer Nonlinear Optimization, User's Guide. Engineering Design Research Center. Carnegie Mellon University, Pittsburgh
10. Schweiger CA, Floudas CA (1997) MINOPT: A Software Package for Mixed-Integer Nonlinear Optimization, User's Guide. Computer-Aided Systems Laboratory Dept. of Chemical Engineering. Princeton University, Princeton
11. Brooke A, Kendrick D, Meeraus A (1998) GAMS: A User's Guide. GAMS Development Corporation, Washington

Rosenbrock Method

RM

VASSILIOS S. VASSILIADIS, RAÚL CONEJEROS
Chemical Engineering Department,
University Cambridge,
Cambridge, UK

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Gradient-free minimization; Rosenbrock method; Line search methods; Pattern search; Gram–Schmidt orthogonalization

The Rosenbrock method is in its basic form a *gradient-free minimization algorithm*. It was introduced by H.H. Rosenbrock [2], and avoids the use of *line searches*. It is based on *orthogonal search directions* with alternating minimizations between these and using *pattern search* at the end of each orthogonal direction search cycle. The algorithm is given below. The minimization problem considered is:

$$\min_{\mathbf{x}} f(\mathbf{x}).$$

1. Initialization

Set the search directions to be the coordinate directions, $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n)}$, where the $\mathbf{d}^{(i)}$ are vectors of zeros, except for a 1 in the ith position. Select a scalar value $\epsilon > 0$ to be used as the termination tolerance, an expansion factor $\beta_1 > 1$ and a contraction factor β_2 such that $-1 < \beta_2 < 0$. Set the initial stepsizes $\delta_i^{(0)}$, $i = 1, \ldots, n$, along each of the above defined search directions. Select an initial point $\mathbf{x}^{(0)}$ and initialize by setting $\mathbf{z}^{(1)} = \mathbf{x}^{(0)}$. Set the pattern iteration counter $k = 0$ and the direction search counter $i = 1$. Initialize the stepsizes along each direction to $\delta_i = \delta_i^{(0)}$.

2. Main Iteration Step (direction Search)

2.1 Forward search
IF $f(\mathbf{z}^{(i)} + \delta_i \mathbf{d}^{(i)}) < f(\mathbf{z}^{(i)})$
THEN
forward search step i is successful;
set $\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} + \delta_i \mathbf{d}^{(i)}$;
set $\delta_i \leftarrow \beta_1 \delta_i$.
ELSE (if $f(\mathbf{z}^{(i)} + \delta_i \mathbf{d}^{(i)}) \geq f(\mathbf{z}^{(i)})$)
forward search step is unsuccessful;
set $\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)}$;
set $\delta_i \leftarrow \beta_2 \delta_i$.
END IF;
IF $i < n$,
increment search counter $i \leftarrow i + 1$;
go to step 2.1.
ELSE (if $i = n$) go to step 2.2.
END IF;

2.2 IF $f(\mathbf{z}^{(n+1)}) < f(\mathbf{z}^{(1)})$
(at least one improvement achieved in 2.1);
set $\mathbf{z}^{(1)} = \mathbf{z}^{(n+1)}$, $i = 1$;
go to step 2.1.
ELSE (if $f(\mathbf{z}^{(n+1)}) = f(\mathbf{z}^{(1)})$)
(no improvement achieved in 2.1);
IF $f(\mathbf{z}^{(n+1)}) < f(\mathbf{x}^{(k)})$
(one improvement in iteration k);
go to step 3.
ELSE (if $f(\mathbf{z}^{(n+1)}) = f(\mathbf{x}^{(k)})$)
(no improvement in iteration k);
IF $\mid \delta_i \mid \leq \epsilon$ for all i,
THEN $\mathbf{x}^{(k)}$ is an estimate of the optimal solution;
STOP.
ELSE
set $\mathbf{z}^{(1)} = \mathbf{z}^{(n+1)}$;
set $i = 1$, go to step 2.1.
END IF;
END IF;
END IF;

3. Pattern Search and New Search Direction Set Generation

3.1 Set $\mathbf{x}^{(k+1)} = \mathbf{z}^{(n+1)}$.
IF $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| < \epsilon$,
THEN $\mathbf{x}^{(k+1)}$ is an estimate of the optimal solution; STOP.
ELSE solve the linear system
$[\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n)}] \cdot [\lambda_1, \ldots, \lambda_n]^\top = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ for λ_i;
go to step 3.2

3.2 Orthonormalization of new search directions (*Gram-Schmidt procedure*)

a set $\mathbf{a}^{(i)} = \mathbf{d}^{(i)}$ if $\lambda_i = 0$;
set $\mathbf{a}^{(i)} = \sum_{j=i}^{n} \mathbf{d}^{(i)}$ if $\lambda_i \neq 0$.

b set $\mathbf{b}^{(1)} = \mathbf{a}^{(1)}$;
set $\mathbf{b}^{(i)} = \mathbf{a}^{(i)} - \sum_{j=1}^{i-1} \left((\mathbf{a}^{(i)})^\top \widehat{\mathbf{d}}^{(j)} \right) \widehat{\mathbf{d}}^{(j)}$, $i \geq 2$,
where $\widehat{\mathbf{d}}^{(i)} = \mathbf{b}^{(i)} / \| \mathbf{b}^{(i)} \|$;
denote these new directions $\widehat{\mathbf{d}}^{(i)}$ as $\mathbf{d}^{(i)}$.

3.3 Reset the stepsizes $\delta_i = \delta_i^{(0)}$, for $i = 1, \ldots, n$.
set the initial point $\mathbf{z}^{(1)} = \mathbf{x}^{(k+1)}$;
increment the main cycle counter $k \leftarrow k + 1$;
set $i = 1$;
Go to step 1.

The procedure as described in the algorithm above can be seen to be taking discrete steps along each of the n directions. Success in each of these is followed by an expansion of the stepsize, while a failure is marked by a reduction of the stepsize, changing direction using β_2 which is negative, to be taken in the next cycle. Upon a single success of the cycle of n searches a new set of search directions is generated by the orthogonalization procedure (Gram–Schmidt). Continued failures in the search directions will result in stepsize shrinking and triggering of the termination criterion (within ϵ tolerance) eventually.

It is possible to derive a continuous minimization procedure along each of the search directions, by replacing the discrete step search with a line search procedure. Such a scheme can be found for example in [1], where it is also demonstrated that under differentiability assumptions on the objective function f it is possible to show that Rosenbrock's method converges to a stationary point of f (minimum under convexity assumptions).

See also

- ► Cyclic Coordinate Method
- ► Powell Method
- ► Sequential Simplex Method

References

1. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming, theory and algorithms. Wiley, New York
2. Rosensbrock HH (1960) An automatic method for finding the greatest or least value of a function. Comput J 3:175–184

Rosen's Method, Global Convergence, and Powell's Conjecture

FSP

Ding-Zhu Du[1], Panos M. Pardalos[2], Weili Wu[1]
[1] Department Computer Sci. and Engineering, University Minnesota, Minneapolis, USA
[2] Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 90C30

Article Outline

Keywords
Rosen's Method
Global Convergence
Powell's Conjecture
See also
References

Keywords

Nonlinear programming; Rosen gradient projection method; Global convergence

The *gradient projection* method proposed by J.B. Rosen in 1960 [82,83] is one of earliest methods in the history of mathematical programming for solving constrained optimization problems. The importance of this method in the literature also stems from the fact that many more efficient algorithms, in linear and nonlinear programming, developed later (e. g. by D. Goldfarb [42], by B.H. Murtagh and R.W.H. Sargent [63] and by N.K. Karmarkar [53]) incorporated the basic ideas propounded by Rosen.

The *global convergence* of Rosen's method was a long-standing open problem. Since Rosen's method is included in many textbooks, the convergence problem became quite well-known. In fact, almost all books (such as [3,4,57,67]) that have a chapter or a section to introduce Rosen's method recognize the problem on the global convergence of Rosen's method. Through efforts of 26 years, the proof was finally found [35,36,47].

The study on the global convergence of Rosen's method had a great impact on the development of a general theory of global convergence in nonlinear programming. In fact, many new techniques [28,29] were discovered to reach the final solution. It is desirable to solve other open problems with them.

One of big remaining open problems about global convergence in nonlinear programming is *Powell's conjecture* that the *DFP method* (*Davidon–Fletcher–Powell*) for unconstrained optimization is globally convergent. Progress has been made slowly [69,70,71].

This article will review the story about Rosen's method and survey the results in theory of global con-

vergence and the development about Powell's conjecture.

Rosen's Method

Consider *linearly constrained optimization problems* in the following form:

$$\begin{cases} \max & f(\mathbf{x}) \\ \text{s.t.} & \mathbf{a}_j^\top \mathbf{x} \geq b_j, \quad j = 1, \ldots, m, \end{cases} \tag{1}$$

where $\mathbf{x} \in \mathbf{R}^n$; $\mathbf{R}^n$ is the n-dimensional Euclidean space whose points are represented as column vectors. A point is called a *feasible point* if it satisfies the constraints. The set of all feasible points is called the *feasible region*. Without special mentioning, we always assume that the function f is continuously differentiable in a convex set containing the feasible region. The function f is always referred to be an *objective function*. For simplicity of notations, we denote $\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x})$, and especially $\mathbf{g}_k = \mathbf{g}(\mathbf{x}_k)$ and $\mathbf{g}^* = \mathbf{g}(\mathbf{x}^*)$.

A nonzero vector $\mathbf{d}$ is called a *feasible direction* at a feasible point $\mathbf{x}$ if there exists $\overline{\lambda} > 0$ such that for any $\lambda \in [0, \overline{\lambda}]$, $\mathbf{x} + \lambda \mathbf{d}$ is feasible.

A constraint is called an *active constraint* at a feasible point $\mathbf{x}$ if its equality sign holds at the point. The set of indices of active constraints is called the *active set*, denoted by $J(\mathbf{x})$, i. e.

$$J(\mathbf{x}) = \left\{ j: \ \mathbf{a}_j^\top \mathbf{x} = b_j \right\}.$$

In particular, we denote

$$J_k = J(\mathbf{x}_k) \quad \text{and} \quad J^* = J(\mathbf{x}^*).$$

For simplicity, we also denote $M = \{1, \ldots, m\}$. Note that $J \subset J'$ will stand for $J \subseteq J'$ and $J \neq J'$. For a singleton $\{h\}$, we write $J \setminus h$ and $J \cup h$ instead of $J \setminus \{h\}$ and $J \cup \{h\}$, respectively.

For the feasible region in the considered problem (1), a direction $\mathbf{d}$ is *feasible* at a feasible point $\mathbf{x}$ if and only if $\mathbf{a}_j^\top \mathbf{d} = 0$ for $j \in J(\mathbf{x})$. In fact, if $\mathbf{d}$ is a feasible direction at point $\mathbf{x}$, then there exists a number $\lambda > 0$ such that $\mathbf{x} + \lambda \mathbf{d}$ is a feasible point. Thus, $\mathbf{a}_j^\top (\mathbf{x} + \lambda \mathbf{d}) \geq b_j$ for $j \in M$. Since $\mathbf{a}_j^\top \mathbf{x} = b_j$ for $j \in J(\mathbf{x})$, we have $\lambda \mathbf{a}_j^\top \mathbf{d} \geq 0$ for $j \in J(\mathbf{x})$. Therefore, $\mathbf{a}_j^\top \mathbf{d} \geq 0$ for $j \in J(\mathbf{x})$.

Conversely, suppose $\mathbf{a}_j^\top \mathbf{d} \geq 0$ for $j \in J(\mathbf{x})$. Set

$$\overline{\lambda} = \begin{cases} \min \left\{ \dfrac{b_j - \mathbf{a}_j^\top \mathbf{x}}{\mathbf{a}_j^\top \mathbf{d}} : \ \mathbf{a}_j^\top \mathbf{d} < 0 \right\} & \text{if } \exists j: \\ & \mathbf{a}_j^\top \mathbf{d} < 0, \\ 1 & \text{otherwise.} \end{cases}$$

Clearly, $\overline{\lambda} > 0$ and for $\lambda \in [0, \overline{\lambda}]$,, $\mathbf{x} + \lambda \mathbf{d}$ is feasible.

Now, define

$$D^1(\mathbf{x}) = \left\{ \mathbf{d}: \ \mathbf{a}_j^\top \mathbf{d} = 0 \text{ for } j \in J(\mathbf{x}) \right\}.$$

Then $D^1(\mathbf{x})$ is exactly the set of all feasible direction at point $\mathbf{x}$.

A feasible point $\mathbf{x} *$ is said to be a *local maximum* of problem (1) if there exists a neighborhood of the point $\mathbf{x}^*$ such that for any feasible point $\mathbf{x}$ in the neighborhood, $f(\mathbf{x}) \leq \mathrm{f}(\mathbf{x}^*)$.

A nonzero vector $\mathbf{d}$ is called an *ascendant direction* at a feasible point $\mathbf{x}$ if $\mathbf{g}(\mathbf{x})^\intercal \mathbf{d} > 0$. Clearly, if $\mathbf{d}$ is a *feasible ascendant direction* at $\mathbf{x}$, then we can find a feasible point $\mathbf{x}'$ along the direction $\mathbf{d}$ such that $f(\mathbf{x}') > f(\mathbf{x})$. Therefore, $\mathbf{x}$ is a local maximum only if there does not exist a feasible ascendant direction at $\mathbf{x}$. That is, a feasible point $\mathbf{x}$ is a local maximum only if

$$D^1(\mathbf{x}) \cap D^2(\mathbf{x}) = \emptyset,$$

where

$$D^2(\mathbf{x}) = \left\{ \mathbf{d}: \ \mathbf{g}^\top \mathbf{d} > 0 \right\}.$$

The following Theorem states a necessary and sufficient condition for $D^1(\mathbf{x}) \cap \mathrm{D}^2(\mathbf{x}) = \emptyset$.

Theorem 1 *Let $\boldsymbol{x}$ be a feasible point. Then $D^1(\boldsymbol{x}) \cap D^2(\boldsymbol{x}) = \emptyset$ if and only if there exist u_j, $j \in J(\boldsymbol{x})$, such that*

$$\mathbf{g}(\mathbf{x}) = \sum_{j \in J(\mathbf{x})} u_j \mathbf{a}_j \tag{2}$$

and

$$u_j \leq 0 \text{ for } j \in J(\mathbf{x}). \tag{3}$$

A feasible point $\mathbf{x}$ is called a *Kuhn–Tucker point* if there exist u_j, $j \in J(\mathbf{x})$, satisfying (2) and (3). Clearly, for a linearly constrained optimization problem, every local maximum is a Kuhn–Tucker point. However, a Kuhn–Tucker point may not be a local maximum.

A feasible point $\mathbf{x}$ is *regular* if all active constraints at $\mathbf{x}$ are linearly independent, i. e. $\mathbf{a}_j$ for $j \in J(\mathbf{x})$ are linearly dependent. The problem (1) is *nondegenerate* if its constraints satisfy the following *regularity condition*: Every feasible point is regular.

Suppose the problem (1) is nondegenerate. At each feasible point $\mathbf{x}$, the subspace $D_J = \{\mathbf{d}:\mathbf{a}_j^\top \mathbf{d} = 0,\ j \in J\}$ where $J = J(\mathbf{x})$ is the tangent plane at the point. The *orthogonal projection operator* on D_J is denoted by P_J. It is not hard to find out that

$$P_J = I - A_J(A_J^\top A_J)^{-1}A_J^\top,$$

where A_J denotes the matrix consisting of column vectors $\mathbf{a}_j, j \in J$. An important property of gradient projection is as follows.

Theorem 2 *Let $0 < c \le +\infty$. For $J = J(\boldsymbol{x})$, define*

$$\mathbf{d} = \begin{cases} P_J\mathbf{g} & \text{if } \|P_J\mathbf{g}\| > c \cdot u_h, \\ P_{J\setminus h}\mathbf{g} & \text{otherwise}, \end{cases}$$

where $u_h = \max_{j\in J} u_j$. Then $\boldsymbol{d} = \boldsymbol{0}$ if and only if $\boldsymbol{x}$ is a Kuhn–Tucker point. Furthermore, if $\boldsymbol{d} \neq \boldsymbol{0}$, then $\boldsymbol{d}$ is an ascendant feasible direction.

Theorem 2 suggests the algorithm Rosen's method below.

The global convergence of Rosen's method relies on a parameter chosen at each iteration. To avoid the zigzag phenomena, Rosen chose the parameter to be a positive number with an upper bound. He also gave a convergence proof in his paper. However, it did not take very long for someone to point out that there is a serious mistake in the proof. Since then, substantial efforts have been made on the problem. Maybe, it is due to a natural expectation for seeking simplicity. Most books incorrectly state Rosen's method by setting the parameter to be zero which is exactly what Rosen tried to avoid. See [30] for a counterexample to the method described in those books. It was indicated in [35] that this counterexample also works for the case in which the parameter varies and converges to zero as the computation runs from the first iteration to the infinity. It follows that Rosen's restriction on the parameter is not sufficient for the convergence.

Initially, choose a feasible point $\mathbf{x}_1$. At each iteration $k = 1, 2, \ldots$, the algorithm carries out the following steps.

1 Choose a positive number c_k. Compute a search direction by the following formula:

$$\mathbf{d}_k = \begin{cases} P_{J_k}\mathbf{g}_k & \text{if } \| P_{J_k}\mathbf{g}_k \| > c_k u_{kh_k}, \\ P_{J_k\setminus h_k} & \text{otherwise}, \end{cases}$$

where

$$(u_{kj},\ j \in J_k)^\top = (A_{J_k}^\top A_{J_k})^{-1}A_{J_k}^\top \mathbf{g}^k$$

and

$$u_{kh_k} = \max\{u_{kj} :\ j \in J\setminus M'\}.$$

2 If $\mathbf{d}_k \neq \mathbf{0}$, then stop; $\mathbf{x}_k$ is a Kuhn-Tucker point.
If $\mathbf{d}_k \neq \mathbf{0}$, then compute

$$m = \min\{\frac{b_j - \mathbf{a}_j^\top \mathbf{x}_k}{\mathbf{a}_j^\top \mathbf{d}_k} : \mathbf{a}_j^\top \mathbf{d}_k < 0\},$$

$$\overline{\lambda}_k = \begin{cases} 1 & \text{if } \mathbf{a}_j^\top \mathbf{d}_k \ge 0 \text{ for all } j \notin J_k, \\ m & \text{otherwise}, \end{cases}$$

and find a new point $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k$, $(0 < \lambda_k \le \overline{\lambda}_k)$ by a line search procedure.

Rosen's method

The first convergent version was found by E. Polak [66]. Compared with the original one, Polak's version is too complicated. First of all, he used the ϵ-active set strategy. Secondly, the version contains a special procedure, which involves computing the gradient projection several times. Polak proved the convergence of his version under a condition, named by ϵ-hypothesis. X.-S. Zhang [96] showed that ϵ-hypothesis is equivalent to

the regularity condition. Zhang's work motivated from a Yue–Han theorem [92].

M. Yue and J. Han utilized Polak's procedure to study the reduced gradient method. The reduced gradient method was first proposed by P. Wolfe [87]. An important step in the reduced gradient method is to compute a reduced basis. There are several ways for finding a reduced basis [33,85,86,92,93]. It has been known to be quite efficient in practice. Wolfe's original version of reduced gradient method is not globally convergent. A counterexample has been given by Wolfe [90] himself. Several convergent versions [57,85,86,92,93] appeared later. Among them, Wang [85] derived his version from the *Levitin–Polyak's method* [56]. Levitin–Polyak's method is a gradient projection method different from Rosen's method. In each iteration, they chose the search direction to be the projection of gradient on the feasible region. In general, we cannot find a closed form formula for such a projection. However, theoretically, the method can be put in very a general setting [1,46,52,60].

Yue and Han [92] also obtained a new property of a linearly constrained set in the format of the standard form of linear programming. These types of properties are very useful for the application of ϵ-active set strategy. See [32] for a generalization. D.-Z. Du [22] also found a simpler convergent version for Rosen's method by deleting Polak's special procedure.

Although several convergent versions have been found, one still wanted to know what would happen when the parameter is also bounded below by a positive number. In fact, all the above convergent versions use some ideas different from Rosen's one for anti-zigzaging. Thus, it was still open whether Rosen's original idea that keeps the parameter away from zero and infinity works or not. (Allowing the parameter to approach zero as the iteration varies was considered as an oversight in [82].)

Zhang [97] made the first breakthrough in this direction; he showed that if the parameter c_k equals a positive constant at every iteration, then Rosen's method is convergent in the three-dimensional space. Soon later, Du and Zhang [35] established a more general result that if the parameter c_k is chosen to be a positive constant with a specific upper bound, then Rosen's method is convergent in the n-dimensional space. This is the first solution for the convergence problem. In the same paper, Du and Zhang also conjecture that the upper bound on the constant can be deleted, that is, as long as the parameter does not vary as the iteration varies, it can be chosen to be any positive number. This conjecture was settled through several efforts including [26,47], and [36]. Finally, [34] showed a convergence theorem with a more general rule for selecting the parameter.

The global convergence of Rosen's method also depends on the choice of line search. In [36] it is assumed that the line search is normal, including exact line search, Curry test, Goldstein test, Wolfe test, and Armijo rule [2].

Theorem 3 *Consider Rosen's method with a normal line search procedure. Let α and β be two positive numbers with $\alpha \le \beta$. Suppose in Rosen's method the parameters c_k are chosen to satisfy $\alpha \le c_k \le \beta$. Then the method either stops at a Kuhn–Tucker point or generates an infinite sequence whose cluster points are all Kuhn–Tucker points.*

If the line search is not normal, X.-D. Hu [48] showed that the global convergence of Rosen's method may fail But, with Hu's counterexample, the method in [22] still has the global convergence property. If the steplength in the line search is uniformly bounded, the choice of parameter can be further relaxed [29], which gives an improvement of Ritter's result [80].

Theorem 4 *Consider Rosen's method with a normal line search procedure of uniformly bounded steplength. Let α and β be two positive numbers and ℓ and ℓ' two positive integers. Suppose in Rosen's method the parameters c_k are chosen to satisfy*

a) $c_k \ge \alpha$ whenever $J_k = \cdots = J_{k-\ell}$; and

b) $c_k \le \beta$ whenever $J_k \ne \cdots \ne J_{k-\ell'}$ and $|J_k| = \cdots = |J_{k-\ell'}|$.

Then the method either stops at a Kuhn–Tucker point or generates an infinite sequence whose cluster points are all Kuhn–Tucker points.

Finally, we would like to mention that [45] gives a method to deal with this degenerative case. This method uses the lexicographic simplex procedure to deal with degenerate line constraints. It is worth mentioning that Bland's rule [5] can also be used here instead of the lexicographic simplex procedure.

Global Convergence

The significance of the solution for the global convergence problem of Rosen's method is not only on a single algorithm. In fact, the solution is obtained based on discovery of new techniques and new developments of global convergence theory in nonlinear programming.

To give a unified convergence theorem, W.I. Zangwill [94,95] introduced a point-to-set mapping; each iteration that finds a new point from the current point is viewed as taking an element from the image set of the point-to-set mapping. He showed an abstract convergence theorem under the closeness of the mapping.

Let X and Y be two topological spaces. Denote by $\mathcal{P}(X)$ the collection of all subsets of X. A mapping from X to $\mathcal{P}(Y)$ is usually called a *point-to-set mapping*. A point-to-set mapping $A{:}X \to \mathcal{P}(Y)$ is said to be *closed* at a point $\mathbf{x}$ if

$$\begin{cases} \mathbf{x}_k \to \mathbf{x} \\ \mathbf{y}_k \to \mathbf{y} \\ \mathbf{y}_k \in A(\mathbf{x}_k) \end{cases} \quad \to \quad \mathbf{y} \in A(\mathbf{x}). \tag{4}$$

A point-to-set mapping is *closed* if it is closed at every point in its definition domain.

Consider the following abstract algorithm.

Let Γ be a subset of a topological space X. Let A be a point-to-set mapping from $X \setminus \Gamma$ to $P(X)$.
0 Choose an initial point $\mathbf{x}_1$.
k If $\mathbf{x}_k \in \Gamma$, then stop. Otherwise, choose $\mathbf{x}_{k+1} \in A(\mathbf{x}_k)$.

Zangwill's algorithm

Theorem 5 (Zangwill's theorem) *Let f be a continuous function on a topological space X and Γ a subset of X. Let A be a closed point-to-set mapping from $X \setminus \Gamma$ to $\mathcal{P}(X)$ such that*

a) the closure of $\cup_{k=1}^{\infty} A(\boldsymbol{x}_k)$ is compact for every convergent sequence of points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ in $X \setminus \Gamma$; and

b) for every $\boldsymbol{x} \in X \setminus \Gamma$ and $\boldsymbol{y} \in A(\boldsymbol{x})$, $f(\boldsymbol{y}) > f(\boldsymbol{x})$.

Then Zangwill's algorithm either stops at a point in Γ or generates an infinite sequence whose cluster points are all in Γ.

However, for a constrained optimization problem, the closeness is lost when the line search procedure is stopped by a constraint. Du [23] proved a result to tell when an algorithm has the global convergence provable by Zangwill's theorem. G.P. McCormick [58] suggested to search along a broken line, that is, if the search is stopped by reaching a new constraint then do not stop there and, keeping the constraint active, find a new direction to continue the search. K. Ritter [78,79,80] decomposed the broken line search into several line searches and applied it to a family of feasible direction method including the gradient projection method. Since the discovery of Zangwill's theorem, many abstract convergence results based on the point-to-set mapping have been established. See [50,51,61,62] for some of them. Some necessary or sufficient conditions for global convergence, such as Bazaraa–Shetty's condition [4] and Wolfe's work [88,89,90], also appeared. However, none of them is powerful enough to show the convergence of Rosen's gradient projection method.

As the open problem on Rosen's method is resolved, a number of new techniques [28,29] for studying the global convergence of 'nonclosed' algorithms have been discovered. These techniques have now been known as *slope lemmas*.

Lemma 6 (first slope lemma) *Let $\{\boldsymbol{x}_k\}$ be a sequence of feasible points such that $f(\boldsymbol{x}_k) < f(\boldsymbol{x}_{k+1})$ for $k = 1, 2, \ldots$. Let $\boldsymbol{x}^*$ be a cluster point of the sequence such that for any subsequence $\{\boldsymbol{x}_k\}_{k \in K}$ converging to $\boldsymbol{x}^*$, $\boldsymbol{x}_{k+1} - \boldsymbol{x}_k \to \boldsymbol{0}$ as $k \to \infty$, $k \in K$. If $\{\boldsymbol{x}_k\}$ does not converge to $\boldsymbol{x}^*$, then there exists a subsequence $\{\boldsymbol{x}_k\}_{k \in K}$ such that $\boldsymbol{x}_k \to \boldsymbol{x}^*$ as $k \to \infty$, $k \in K$, and*

$$\lim_{k \to \infty, k \in K} \frac{\mathbf{g}_k^{\top}(\mathbf{x}_{k+1} - \mathbf{x}_k)}{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|} = 0. \tag{5}$$

Lemma 7 (second slope lemma) *Let $\{\boldsymbol{x}_k\}$ be a sequence of feasible points, in a linearly constrained region, convergent to $\boldsymbol{x}^*$. Suppose $f(\boldsymbol{x}_k) < f(\boldsymbol{x}_{k+1})$ for all k. Then there exists a subsequence $\{\boldsymbol{x}_k\}_{k \in K}$ such that for every J with $J = J_k$ for infinitely many k,*

$$0 \le \lim_{k \to \infty, k \in K} \frac{\mathbf{g}_k^{\top}(\mathbf{x}_{k+1} - \mathbf{x}_k)}{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|} \le \|P_J \mathbf{g}(\mathbf{x}^*)\|.$$

Lemma 8 (third slope lemma) *Let $\{\boldsymbol{x}_k\}$ be a sequence of feasible points such that for all k, $f(\boldsymbol{x}_k) <$*

$f(\boldsymbol{x}_{k+1})$. Let $\boldsymbol{x}^*$ be a cluster point of the sequence. Suppose that for any subsequence $\{\boldsymbol{x}_k\}_{k\in K}$ converging to $\boldsymbol{x}^*$, $\lim_{k\to\infty,\, k\in K}(\boldsymbol{x}_{k+1}-\boldsymbol{x}_k)=\boldsymbol{0}$. If there exists a positive number μ such that for all k,

$$\frac{\mathbf{g}_k^\top(\mathbf{x}_{k+1}-\mathbf{x}_k)}{\|\mathbf{x}_{k+1}-\mathbf{x}_k\|}\geq \mu\max\{\|P_{J_k}\mathbf{g}_k\|,\|P_{J_{k+1}}\mathbf{g}_k\|\},$$

then the following two statements are equivalent:

a) there exists a subsequence $\{\boldsymbol{x}_k\}_{k\in K}$ converging to $\boldsymbol{x}^*$ such that

$$\lim_{k\to\infty,k\in K}\frac{\mathbf{g}_k^\top(\mathbf{x}_{k+1}-\mathbf{x}_k)}{\|\mathbf{x}_{k+1}-\mathbf{x}_k\|}=0. \tag{6}$$

b) For every subsequence $\{\boldsymbol{x}_k\}_{k\in K}$ converging to $\boldsymbol{x}^*$, (6) holds.

The first and the second slope lemmas have intuitive background as follows.

When you passed by a mountain, you may have noticed that the road somehow looks like a snake in *s*-shape. Why was the road built in such a way? The answer is easy, increase the length to decrease the slope. The longer its length is, the smaller its slope is. However, did you think that as its length approaches infinity its slope approaches zero? This is what the first slope lemma states.

Consider a path going up on a mountain. The average slope is the ratio of the height increment over the length of the projection of the path onto the level plane. Clearly, the average slope of any path cannot exceed the slope of the direct path connecting the two endpoints. This is what the second slope lemma states.

The power of the first slope lemma is surprising. It can show that a large class of feasible direction methods share a convergence property that if the generated sequence of points has a cluster point but does not converge to it, then every cluster point of the sequence is a Kuhn–Tucker point. It is interesting to point out that Wolfe [90] has showed by a counterexample that Zoutendijk's method [4,98] can generate a sequence of points converging to a point which is not a Fritz John point. However, with the first slope lemma, we can still show that it generates a sequence of points which do not converge to a point, then every cluster point of the sequence is a Fritz John point.

Powell's Conjecture

Consider an unconstrained optimization problem as follows:

$$\max f(\mathbf{x}),$$

where f is continuously differentiable in $\mathbf{R}^n$.

The first *quasi-Newton method* was initially proposed by W.C. Davidon [16] in 1959, but his work was published nine years later [17]. The public attention on Davidon's work was largely due to the introduction of R. Fletcher and M.J.D. Powell [41] in 1963. This method is called the *DFP method*(Davidon–Fletcher–Powell).

DFP method

Choose an initial point $\mathbf{x}_1$ and an initial positive definite symmetric matrix H_1. Set $k=1$.

1 Compute $\mathbf{g}_k$. If $\mathbf{g}_k=\mathbf{0}$, then stop: else, go to step 2;

2 If $k>1$, then compute a positive definite symmetric matrix H_k with an updating formula

$$H_k=H_{k-1}-\frac{\boldsymbol{\sigma}_{k-1}\boldsymbol{\sigma}_{k-1}^\top}{\boldsymbol{\sigma}_{k-1}^\top\mathbf{y}_{k-1}}-\frac{H_{k-1}\mathbf{y}_{k-1}(H_{k-1}\mathbf{y}_{k-1})^\top}{\mathbf{y}_{k-1}^\top H_{k-1}\mathbf{y}_{k-1}},$$

where $\mathbf{y}_{k-1}=\mathbf{g}_k-\mathbf{g}_{k-1}$ and $\boldsymbol{\sigma}_{k-1}=\mathbf{x}_k-\mathbf{x}_{k-1}$. Compute a search direction $\mathbf{d}_k=H_k\mathbf{g}_k$. Set $\mathbf{x}_{k+1}=\mathbf{x}_k+\theta\mathbf{d}_k$ and $k:=k+1$, where θ is chosen by a line search. Go to step 1.

Since then, many variations and generalizations of quasi-Newton methods [6,7,44,49,84,91] appeared in the literature. There are three issues on the convergence of quasi-Newton methods: quadratic termination, global convergence, and convergence rate. All quasi-Newton methods have quadratic termination under certain conditions [8,9]. However, the global convergence is a difficult problem for quasi-Newton methods. Powell [69] in 1971 established the first convergence theorem that if the objective function is twice continuously differentiable and uniformly concave, then the DFP method with exact line search generates a sequence convergent to a stationary point.

L.C.W. Dixon [20,21] at the same period found that all methods in *Broyden's family* with exact line search actually generate an identical sequence. This discovery extended Powell's result to all members in Broyden's family. It is interesting to point out that although Powell in 1971 obtained a convergence theorem, he did not believe that DFP method in general is globally convergent. Thus, he conjectured in 1971 that there exists a twice continuously differentiable objective function such that DFP method generates a sequence convergent to a point which is not a stationary point. However, one year later he changed his mind. In fact, after eighteen months he worked on his own conjecture, but he did not find such a counterexample. Instead, he [70] proved that his conjecture in 1971 is false for objective functions with two variables and that the uniform concavity in his convergence theorem can be replaced by concavity and an upper bound on the objective function.

Theorem 9 *Let f be a twice continuously differentiable concave real function on $\boldsymbol{R}^n$. Suppose that the level set $\{\boldsymbol{x}: f(\boldsymbol{x}) \geq f(\boldsymbol{x}_1)\}$ is bounded. Then the DFP method with exact line search and initial point $\boldsymbol{x}_1$ either stops at the maximum or generates an infinite sequence whose function value converges to the maximum value of $f(\boldsymbol{x})$.*

Therefore, since 1972, the following is considered as the conjecture of Powell:

Conjecture 10 (Powell's conjecture) *Suppose the objective function is continuously differentiable. Then every cluster point of the sequence generated by DFP method with exact line search is a stationary point. D. Pu and W. Yu [77] made an important progress on Powell's conjecture. They showed the following.*

Theorem 11 *Suppose DFP method generates an infinite sequence $\{\boldsymbol{x}_k\}$ converging to $\boldsymbol{x}^*$. If the objective function f belongs to the class $C^{1,1}$, that is, there exists a constant L such that for every $\boldsymbol{x}$ and $\boldsymbol{y}$*

$$\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\|,$$

then

$$\liminf_{k\to\infty} \|\mathbf{g}_k\| = 0.$$

Powell [72] also showed a global convergence theorem for *BFGS method* with *inexact line search* (*Wolfe test*) for twice continuously differentiable and concave objective functions with a upper bound. Ritter [81] tried to generalize Powell's result to whole Broyden's family. However, the restriction on search step in line search restricts success of his result. R.H. Byrd, J. Nocedal, and Y. Yuan [12] successfully generalized Powell's result to all members in Broyden's family except DFP method. It is an very interesting open problem whether the DFP method with the Wolfe test has the same global convergence. With a modified Wolfe test and a suitably smooth objective function, Pu [76] showed that if a member in Broyden's family generates a convergent sequence, then the sequence converges to a stationary point.

For the convergence rate, Powell [69] showed that if the objective function is uniformly concave and its Hession matrix satisfies the Lipschitz condition, then DFP method with exact line search is superlinearly convergent. This can be extended to all members in Broyden's family by Dixon 's theorem. Similarly, Powell [72] also showed that if the objective function is uniformly concave and its Hession matrix satisfies Lipschitz condition, then BFGS method with Wolfe test is superlinearly convergent. Byrd, Nocedal, and Yuan [12] generalized this result to Broyden's family except DFP method under the condition that the Hession matrix is Hölder continuous. Pu [75] showed that with the modified Wolfe test, DFP method is one-step superlinearly convergent.

Combining Rosen's method with variable metric methods, Goldfarb [42] and Murtagh and Sargant [63] obtained two efficient algorithms. Powell [71] showed that these two algorithms are actually equivalent. The convergence of such algorithms is still an open problem. However, several convergent variations [55,59,79,80] have been established. The convergence of quasi-Newton methods for nonlinearly constrained optimization can be found in [15,65,73,74].

See also

- ► Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- ► First Order Constraint Qualifications
- ► Inequality-constrained Nonlinear Optimization
- ► Kuhn–Tucker Optimality Conditions
- ► Lagrangian Duality: Basics

- ▶ Saddle Point Theory and Optimality Conditions
- ▶ Second Order Constraint Qualifications
- ▶ Second Order Optimality Conditions for Nonlinear Optimization
- ▶ Successive Quadratic Programming: Full Space Methods

References

1. Allen E, Helgason R, Kennington J (1987) A generalization of Polyak's convergence result for subgradient optimization. Math Program 37:309–318
2. Armijo L (1966) Minimization of functions having Lipschitz continuous first partial derivatives. Pacific J Math 16:1–3
3. Avriel M (1976) Nonlinear programming: Analysis and methods. Prentice-Hall, Englewood Cliffs
4. Bazaraa MS, Shetty CM (1979) Nonlinear programming: Theory and algorithms. Wiley, New York
5. Bland RG (1977) New finite pivoting rules for the simplex method. Math Oper Res 2(2):103–107
6. Broyden CG (1965) A class of methods for solving nonlinear simultaneous equations. Math Comput 19:577–593
7. Broyden CG (1967) Quasi-Newton methods and their application to function minimization. Math Comput 21:368–381
8. Broyden CG (1970) The convergence of a class of double-rank minimization algorithms 1: general consideration. J Inst Math Appl 6:76–90
9. Broyden CG (1970) The convergence of a class of double-rank minimization algorithms 2: the new algorithms. J Inst Math Appl 6:222–231
10. Buckley A (1975) An alternate implementation of Goldfarb's minimization algorithm. Math Program 8:207–231
11. Byrd RH (1985) An example of irregular convergence in some constrained optimization methods that use the projected Hessian. Math Program 32:232–237
12. Byrd RH, Nocedal J, Yuan Y (1987) Global convergence of a class of quasi-Newton method on convex problems. SIAM J Numer Anal 24:1171–1190
13. Byrd RH, Shultz GA. A practical class of globally convergent active set strategies for linearly constrained optimization. Manuscript (unpublished)
14. Calamai PH, More JJ (1987) Projected gradient methods for linearly constrained problems. Math Program 39:93–116
15. Coleman TF, Conn AR (1984) On the local convergence of a quasi-Newton method for the nonlinear programming problem. SIAM J Numer Anal 1:755–769
16. Davidon WC (1959) Variable metric method for minimization. AEC Res Developm Report ANL-5990, no. nov
17. Davidon WC (1968) Variable metric method for minimization. Comput J 10:406–410
18. Dax A (1978) The gradient projection method for quadratic programming. Report, Inst Math, Hebrew Univ, Jerusalem
19. Dembo RS, Klincewicz JG (1985) Dealing with degeneracy in reduced gradient algorithms. Math Program 31:257–363
20. Dixon LCW (1972) Quasi-Newton algorithms generate identical points. Math Program 2:383–387
21. Dixon LCW (1972) Quasi-Newton algorithms generate identical points II, the proof of four new theorems. Math Program 3:345–358
22. Du D-Z (1983) A modification of Rosen-Polak's algorithm. Kexue Tongbao 28:301–305
23. Du D-Z (1985) Changing of point-to-set maps and point-to-set map families for continuities. Acta Math Applic Sinica 8(2):142–150, in Chinese
24. Du D-Z (1985) A family of gradient projection algorithms. Acta Math Applic Sinica, English Ser 2:1–13
25. Du D-Z (1985) A gradient projection algorithm for convex programming with nonlinear constraints. Acta Math Applic Sinica 8:7–16, in Chinese
26. Du D-Z (1987) Remarks on the convergence of Rosen's gradient projection method. Acta Math Applic Sinica, English Ser 3:270–279
27. Du D-Z (1988) Gradient projection methods in linear and nonlinear programming. Hadronic Press, Palm Harbor
28. Du D-Z (1991) Convergence theory of feasible direction methods. Sci Press, Marrickville, Australia
29. Du D-Z (1992) Rosen's method and slope lemmas. In: Pardalos PM (ed) Advances in Optimization and Parallel Computing. Elsevier, Amsterdam, pp 68–84
30. Du D-Z, Sun J, Song T-T (1980) A counterexample for Rosen's gradient projection method. Math Assortment 1:4–6, in Chinese
31. Du D-Z, Du X-F. A convergent reduced gradient algorithm without using special pivot. Math Numer Sinica, to appear
32. Du D-Z, Sun J (1983) A new gradient projection method. Math Numer Sinica 4:378–386, in Chinese
33. Du D-Z, Sun J, Song T-T (1984) Simplified finite pivoting processes in the reduced gradient algorithms. Acta Math Applic Sinica 7:142–146, in Chinese
34. Du D-Z, Wu F, Zhang X-S. On Rosen's gradient projection methods. Ann Oper Res, to appear
35. Du D-Z, Zhang X-S (1986) A convergence theorem of Rosen's gradient projection method. Math Program 36:135–144
36. Du D-Z, Zhang X-S (1989) Global convergence of Rosen's gradient projection methods. Math Program 44
37. Du D-Z, Zhang X-S (1989) Notes on a new gradient projection method. System Sci and Math Sci 2
38. Dunn JC (1981) Global and asymptotic convergence rate estimates for a class of projected gradient processes. SIAM J Control Optim 19:368–400
39. Dunn JC (1987) On the convergence of projected gradient processes to singular critical point. J Optim Th Appl 55:203–216
40. Fletcher R (1987) Practical methods of optimization, unconstrained optimization. Wiley, New York

41. Fletcher R, Powell MJD (1963) A rapidly convergent descent method for minimization. Comput J 6:163–168
42. Goldfarb D (1969) Extension of Davidon's variable metric method to maximization under linear inequality and equality constraints. SIAM J Appl Math 17:739–764
43. Goldfarb D (1970) A family of variable metric methods derived by variational means. Math Comput 24:23–26
44. Goldstein J (1970) Variations on variable-metric methods. Math Comput 24:1–22
45. Gui X-Y, Du D-Z (1984) A superlinearly convergent method to linearly constrained optimization problems under degeneracy. Acta Math Applic Sinica, English Ser 1:76–84
46. Haraux A (1977) How to differentiate the projection on a convex set in Hilbert space: Some applications to variational inequalities. J Math Soc Japan 20:615–631
47. He G-Z (1990) Proof of convergence of Rosen's gradient projection method. Acta Math Applic Sinica 1
48. Hu X-D (1989) Nonlinear programming: A unified approach and the upper bound of partial concentrator. PhD Thesis, Inst Appl Math, Chinese Acad Sci
49. Huang HY (1970) Unified approach to quadratically convergent algorithms for function minimization. J Optim Th Appl 5:405–423
50. Huard P (1975) Optimization algorithms and point-to-set maps. Math Program 8:308–331
51. Huard P (1979) Extensions of Zangwill's theorem. Math Program Stud 10:98–103
52. Kalfon P, Ribiere G, Sogno JC (1969) A method of feasible directions using projection operators. Proc IFIP congr 68: Information Processing, 1 North-Holland
53. Karmarkar N (1984) A new polynomial-time algorithm for linear programming. Combinatorica 4:373–395
54. Klee V, Minty GJ (1972) How good is the simplex algorithm? In: Shisha O (ed) Inequalities III. Acad Press, New York
55. Kwei H-Y, Gui X-Y, Wu F, Lai Y-L (1979) Extension of a variable metric algorithm to a linearly constrained optimization problem: A variation of Goldfarb's algorithm. In: Haley KB (ed) O.R. 1978. North-Holland, Amsterdam
56. Levitin ES, Polyak BT (1966) Constrained minimization methods. USSR Comput Math Math Phys 6(5):1–50
57. Luenberger DG (1973) Introduction to linear and nonlinear programming. Addison-Wesley, Reading
58. McCormick GP (1969) Anti-zigzagging by bending. Managem Sci 15:315–320
59. McCormick GP (1990) A second order method for the linearly constrained nonlinear programming problem. In: Rosen JB, Mangasarian OL, Ritter K (eds) Nonlinear Programming. Acad Press, New York
60. McCormick GP, Tapia RA (1972) The gradient projection method under mild differentiability conditions. SIAM J Control Optim 10:93–98
61. Meyer RR (1970) The validity of a family of optimization methods. SIAM J Control Optim 8:41–54
62. Meyer RR (1976) Sufficient conditions for the convergence of monotonic mathematical programming algorithms. J Comput Syst Sci 12:108–121
63. Murtagh BA, Sargent RWH (1969) A constrained minimization method with quadratic convergence. In: Fletcher R (ed) Optimization. Acad Press, New York
64. Murtagh BA, Saunders MA (1978) Large-scale linearly constrained optimization. Math Program 14:14–72
65. Nocedal J, Overton ML (1985) Projected Hessian updating algorithms for nonlinearly constrained optimization. SIAM J Numer Anal 1:821–850
66. Polak E (1969) On the convergence of optimization algorithms. Revue Franc Inform Rech Oper 3:17–34
67. Polak E (1971) Computational methods in optimization. Acad Press, New York
68. Pollak HO (1978) Some remarks on the Steiner problem. J Combin Th A 24:278–295
69. Powell MJD (1971) On the convergence of the variable metric algorithm. J Inst Math Appl 7:21–36
70. Powell MJD (1972) Some properties of the variable metric algorithm. In: Lootsma FA (ed) Numerical Methods for Nonlinear Optimization. Acad Press, New York
71. Powell MJD (1974) Unconstrained minimization and extensions for constraints. In: Hammer PL, Zoutendijk G (eds) Mathematical Programming Theory and Practice. North-Holland, Amsterdam, pp 31–79
72. Powell MJD (1976) Some global convergence properties of a variable metric algorithm for minimization without exact line searches. In: Cottle RW, Lemke CE (eds) Nonlinear Programming. Amer Math Soc, Providence
73. Powell MJD (1978) The convergence of variable metric methods for nonlinearly constrained optimization calculations. In: Mangasarian O, Meyer R, Robinson S (eds) Nonlinear Programming 3. Acad Press, New York, pp 27–63
74. Powell MJD, Yuan Y (1991) A trust region algorithm for equality unconstrained optimization. Math Program 49:189–211
75. Pu D (1992) A class of DFP algorithm without exact linear search. Asia-Pacific J Oper Res 9:207–220
76. Pu D (1997) The convergence of Broyden algorithms without convexity assumption. System Sci and Math Sci 10:289–298
77. Pu D, Yu W (1988) On convergence property of DFP algorithm. J Qüfu Normal Univ 14(3):63–69
78. Ritter K (1973) A superlinearly convergent method for minimization problems with linear inequality constraints. Math Program 4:44–71
79. Ritter K (1975) A method of conjugate directions for linearly constrained nonlinear programming problems. SIAM J Numer Anal 12:272–303
80. Ritter K (1980) Convergence and superlinear convergence of algorithms for linearly constrained minimization problems. In: Dixon LCW, Spedicado E, Szego GP (eds) Nonlinear Optimization: Theory and Algorithms II. Birkhäuser, Basel, pp 221–251

81. Ritter K (1981) Global and superlinear convergence of a class of variable metric methods. Math Program Stud 14:178–205
82. Rosen JB (1960) The gradient projection method for nonlinear programming, Part I: linear constraints. SIAM J Appl Math 8:181–217
83. Rosen JB (1961) The gradient projection method for nonlinear programming, Part II: nonlinear constraints. SIAM J Appl Math 9:514–553
84. Shanno DF (1970) Conditioning of quasi-Newton methods for function minimization. Math Comput 24:647–656
85. Wang CY (1981) Simplifications of a new pivoting rule and Levitin-Polyak gradient projection method and their convergent property. Acta Math Applic Sinica 4:37–52, in Chinese
86. Wang CY (1983) On convergence property of an improved reduced gradient method. Kexue Tongbao 28:577–582
87. Wolfe P (1963) Methods of nonlinear programming. In: Graves RL, Wolfe P (eds) Recent Advances in Mathematical Programming
88. Wolfe P (1969) Convergence conditions for ascent methods. SIAM Rev 11:226–235
89. Wolfe P (1970) Convergence theory in nonlinear programming. In: Abadie J (ed) Integer and Nonlinear Programming. North-Holland, Amsterdam
90. Wolfe P (1972) On the convergence of gradient methods under constraints. IBM J Res Developm 16:407–411
91. Wu F, Gui X (1981) A class of variable metric methods with n+1 parameters. Acta Math 24:921–930
92. Yue M, Han J (1979) A new reduced gradient method. Sci Sinica 22:1099–1113
93. Yue M, Han J (1984) A unified approach to the feasible direction methods for nonlinear programming with linear constraints. Acta Math Applic Sinica, English Ser 1:63–73
94. Zangwill WI (1969) Convergence conditions for nonlinear programming algorithms. Managem Sci 16:1–13
95. Zangwill WI (1969) Nonlinear programming: A unified approach. Prentice-Hall, Englewood Cliffs
96. Zhang X-S (1979) An improved Rosen-Polak method. Acta Math Applic Sinica 2:257–267, in Chinese
97. Zhang X-S (1985 1987) On the convergence of Rosen's gradient projection method. Acta Math Applic Sinica 8:125–128, in Chinese. Also in: English Ser 3:280–288
98. Zoutendijk G (1960) Methods of feasible direction methods. Elsevier, Amsterdam

S

Saddle Point Theory and Optimality Conditions

JØRGEN TIND
University Copenhagen, Copenhagen, Denmark

MSC2000: 90C06

Article Outline

Keywords
Saddle Points
Mathematical Programming
Convex Programming
See also
References

Keywords

Optimization; Saddle point

The notion of a saddle point is a fundamental concept in many areas of science and economics. A classical instance is the famous saddle point theorem for a zero-sum matrix game due to J. von Neumann [2]. We shall here emphasize the utility of saddle points in the context of optimization.

Saddle Points

Let us first recall some fundamental observations about saddle points and duality. Consider a function $K(x, y)$: $X \times Y \to \mathbf{R}$, where $X \subseteq \mathbf{R}^n$ and $Y \subseteq \mathbf{R}^m$. We say that $(x_0, y_0) \in X \times Y$ is a *saddle point* of K on the sets X and Y if

$$K(x, y_0) \leq K(x_0, y_0) \leq K(x_0, y) \tag{1}$$

for all $x \in X$ and $y \in Y$.

Consider next the following two programs:

$$z = \max_{x \in X} \inf_{y \in Y} K(x, y) \tag{2}$$

and

$$w = \min_{y \in Y} \sup_{x \in X} K(x, y). \tag{3}$$

Since one program is derived from the other via an interchange of the optimization directions they constitute a pair of so-called *dual programs*.

We say that $x_0 \in X$ is an *optimal solution of the program* (2) if $\inf_{y \in Y} K(x_0, y) = \max_{x \in X} \inf_{y \in Y} K(x, y))$. Similarly, $y_0 \in Y$ is an optimal solution of the program (3) if $\min_{y \in Y} \sup_{x \in X} K(x, y) = \sup_{x \in X} K(x, y_0)$. Observe that

$$\inf_{y \in Y} K(x_0, y) \leq K(x_0, y_0) \leq \sup_{x \in X} K(x, y_0) \tag{4}$$

for any $(x_0, y_0) \in X \times Y$. This implies that we always have so-called *weak duality*: $z \leq w$. We speak of *strong duality* when optimal solutions (x_0, y_0) exist for program (2) and (3), respectively, so that $z = w$. If so, the inequalities of (4) are turned into equations. In this case $\sup_{x \in X} K(x, y_0)$ is obtained by x_0, so that the sup operator is naturally replaced by the max operator. Similarly for the inf operator and we obtain that

$$\min_{y \in Y} K(x_0, y) = K(x_0, y_0) = \max_{x \in X} K(x, y_0).$$

However, this expression is equivalent to the definition of a saddle point.

This leads us to the result that (x_0, y_0) is a saddle point if and only if x_0 and y_0 are optimal solutions of the programs (2) and (3), respectively, and with equal values, i. e. $z = w$.

Mathematical Programming

We shall now study the particular situation where $K(x, y)$ is the Lagrange function of a mathematical programming problem. Let $f(x)$: $\mathbf{R}^n \to \mathbf{R}$ and $g(x)$: $\mathbf{R}^n \to \mathbf{R}^m$. Further let $K(x, y) = f(x) - yg(x) + yb$, where $b \in \mathbf{R}^m$, and $Y = \mathbf{R}^m_+$, the nonnegative orthant. In this setting program (2) becomes an ordinary mathematical programming problem stated in the following so-called *primal* form:

$$z_p = \begin{cases} \max & f(x) \\ \text{s.t.} & g(x) \le b \\ & x \in X. \end{cases} \tag{5}$$

The *dual* form of (5) is derived from program (3):

$$\min_{y \in \mathbb{R}^m_+} \sup_{x \in X} f(x) - yg(x) + yb$$

or with $u \in \mathbf{R}$ alternatively as

$$w_p = \begin{cases} \min & u + yb \\ \text{s.t.} & u + yg(x) \ge f(x), \quad \forall x \in X, \\ & u \in \mathbb{R}, \quad y \in \mathbb{R}^m_+. \end{cases} \tag{6}$$

Since the programs (5) and (6) are special cases of the dual programs (2) and (3), we get that optimal solutions x_0 and y_0 exist for (5) and (6), respectively, and with $z_p = w_p$ if and only if (x_0, y_0) is a saddle point for $K(x, y) = f(x) - yg(x) + yb$.

The condition (1) for a saddle point (x_0, y_0) looks here as follows:

$$\begin{aligned} & f(x) - y_0 g(x) + y_0 b \\ & \le f(x_0) - y_0 g(x_0) + y_0 b \\ & \le f(x_0) - yg(x_0) + yb \end{aligned} \tag{7}$$

for all $x \in X$ and $y \in \mathbf{R}^m_+$.

We shall show in this particular case that the condition for a saddle point can be restated in an alternative form as the so-called *optimality conditions*:

i) x_0 maximizes $f(x) - y_0 g(x)$ over X;
ii) $y_0(g(x_0) - b) = 0$;
iii) $y_0 \ge 0$; and
iv) $g(x_0) \le b$.

So, assume that we have a saddle point (x_0, y_0) satisfying (7). Then condition i) is implied by the left inequality of (7). If $g(x_0) \not\le b$ then by an appropriate choice of nonnegative elements y we can violate the right inequality of (7). This implies iv). Condition iii) is implied directly. Conditions iii) and iv) imply that $y_0(g(x_0) - b) \le 0$. If this inequality is strict then y_0 does not minimize over $\mathbf{R}_+$ according to the right inequality of (7). This proves ii). Conversely, by similar arguments we can show that i)–iv) imply the saddle point condition (7).

We shall next study the mathematical programming problem (5) and provide conditions leading directly to the existence of a saddle point or equivalently to the satisfaction of the optimality conditions. In this context we shall study the perturbation function of the primal program (5) by varying the right-hand side of the constraints. Let $d \in \mathbf{R}^m$. The perturbation function $\phi(d)$ is then defined as follows:

$$\phi(d) = \begin{cases} \max & f(x) \\ \text{s.t.} & g(x) \le d \\ & x \in X. \end{cases}$$

Let $D = \{d \in \mathbf{R}^m : \exists x \in X \text{ s.t. } g(x) \le d\}$. Consider next the following program:

$$\begin{cases} \min & u + yb \\ \text{s.t.} & u + yd \ge \phi(d), \quad \forall d \in D, \\ & u \in \mathbb{R}, \quad y \in \mathbb{R}^m_+. \end{cases} \tag{8}$$

We shall show that (8) is equivalent to the dual program (6). Assume (u, y) is a feasible solution of (8), i. e. it satisfies the constraints of (8). For $x \in X$ and $d = g(x)$ we then obtain that $u + yg(x) = u + yd \ge \phi(d) = \phi(g(x)) \ge f(x)$. Hence (u, y) is also feasible in (6). Conversely, assume that (u, y) is feasible in (6). If $x \in X$ and $g(x) \le d$ we immediately get that $f(x) \le \phi(d)$ and moreover that $u + yd \ge u + yg(x) \ge f(x)$. Hence $u + yd \ge \phi(d)$ implying that (u, y) is also feasible in (8). So with equal set of feasible solutions and with the same objective function the two programs are equivalent.

The dual program in the form of (8) has a nice geometric interpretation. For a given set of coefficients (u, y) the sum $u + yd$ becomes an affine function in $d \in \mathbf{R}^m$. The objective of (8) is to select the coefficients in such a way that this function always is above the perturbation function, but with the lowest possible value at the point b. (This confirms the existence of weak duality for the dual programs (5) and (6)). For an optimal solution

of the primal programming problem (5) we thus additionally have strong duality if the affine function coincides with the perturbation function at the point b. In mathematical terms the perturbation function ϕ satisfies the following condition:

$$\phi(d) \leq \phi(b) + y(d - b), \quad \forall d \in D, \tag{9}$$

and we say that the perturbation function is *superdifferentiable* at b.

Significant cases exist for this to occur. For example, if the perturbation function is finite and concave over the set D of feasible right-hand sides then it is superdifferentiable at points in the relative interior of D. Moreover, at the boundary the perturbation function is not superdifferentiable if and only if the perturbation function has a directional derivative of value $+\infty$ at the selected point. This case rarely occurs, and if this does not happen the perturbation function is said to be *stable*.

These mainly geometrical observations can be treated more rigorously applying some classical results about conjugate functions and supporting hyperplanes in convex analysis, see [3].

Convex Programming

In this last section we assume that $X = \mathbf{R}^n_+, f(x)$ is differentiable and concave and that the components of $g(x)$ are convex and differentiable. These assumptions are indeed fulfilled in many applications. In this case the perturbation function is finite and concave on $\mathbf{R}^m_+$. We also assume that the perturbation function is stable. By the observations above we then get the following *fundamental property*: If an optimal solution x_0 exists for the primal program (5), then a vector y_0 exists such that (x_0, y_0) constitutes a saddle point (or equivalently that (x_0, y_0) satisfies the optimality conditions).

Moreover by the assumptions made on convexity and differentiability the optimality conditions can be restated into the famous *Karush–Kuhn–Tucker conditions* [1]:

- $(\nabla_x f(x_0) - y_0 \nabla_x g(x_0)) \geq 0$;
- $(\nabla_x f(x_0) - y_0 \nabla_x g(x_0))x_0 = 0$;
- $y_0(g(x_0) - b) = 0$;
- $g(x_0) \leq b$ and $(x_0, y_0) \geq 0$.

Therefore and subject to the assumptions made, the Karush–Kuhn–Tucker conditions provide necessary and sufficient conditions for an optimal solution of the mathematical programming problem (5). This result has had a tremendous impact on the development of algorithms to solve mathematical programming problems. Moreover, if the perturbation function $\phi(d)$ is also differentiable at b then it is approximated by the right-hand side of (9). This happens in many applications. Thus with numerically small deviations $d - b$ the vector y_0 measures the marginal effects on the optimal value of the objective function of (5). Due to this property y_0 is often denoted as the vector of *shadow prices* and as such it plays a central role in the discussion and interpretation of results obtained by mathematical programs, in particular when modeled over problems in economics.

See also

- ▶ Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- ▶ First Order Constraint Qualifications
- ▶ Inequality-constrained Nonlinear Optimization
- ▶ Kuhn–Tucker Optimality Conditions
- ▶ Lagrangian Duality: Basics
- ▶ Rosen's Method, Global Convergence, and Powell's Conjecture
- ▶ Second Order Constraint Qualifications
- ▶ Second Order Optimality Conditions for Nonlinear Optimization

References

1. Kuhn HW, Tucker AW (1951) Nonlinear programming. In: Proc Second Berkeley Symp Math Statist Probab. Univ Calif Press, Berkeley, pp 481–490
2. Neumann J Von, Morgenstern O (1947) Theory of games and economic behaviour. Princeton Univ Press, Princeton
3. Rockafellar RT (1970) Convex analysis. Princeton Univ Press, Princeton

Second Order Constraint Qualifications

DIETHARD KLATTE
Institute OR, University Zurich, Zurich, Switzerland

MSC2000: 90C30, 49K27, 90C31, 49K40

Article Outline

Keywords
Remarks
See also
References

Keywords

Constraint qualification; Optimality conditions; Duality; Stability; Parabolic curve approach

A *second order constraint qualification* (SOCQ) is a condition which is imposed on the analytical description of the constraint set of an optimization problem and which usually involves second order approximations of the data functions. second order constraint qualifications are essential in order to establish second order optimality conditions, but they play also a certain role in the perturbation analysis for optimization problems. Roughly speaking, second order constraint qualifications establish a link between the geometry of the given set and certain kinds of second order approximations of the analytical data. Second order constraint qualifications are closely related to *first order constraint qualifications* (cf. ▶ First Order Constraint Qualifications), in particular, see that article for notions such as LICQ, MFCQ, Robinson CQ and others.

Historically, SOCQs were first introduced in studies of second order optimality conditions for smooth nonlinear programs, see, e. g., [11,12,19,20]. Given twice continuously differentiable functions g_i: $\mathbf{R}^n \to \mathbf{R}$ ($i = 0, \ldots, r$), the smooth mathematical programming problem is

$$(\mathrm{P}) \quad \begin{cases} \min & g_0(x) \\ \text{s.t.} & g_i(x) \le 0, \\ & \quad i \in I := \{1, \ldots, m\}, \\ & g_j(x) = 0, \\ & \quad j \in J := \{m+1, \ldots, r\}. \end{cases} \tag{1}$$

A (dual) second order necessary optimality condition of Fritz John type is the following one: If $\overline{x}$ is a local minimizer of (P), then for every *critical direction* d, i. e., every d satisfying

$$\begin{cases} \langle Dg_i(\overline{x}), d\rangle \le 0, & i \in I^0_{\overline{x}}, \\ \langle Dg_j(\overline{x}), d\rangle = 0, & j \in J, \end{cases}$$

where $I_{\overline{x}} := \{i \in I\colon\ g_i(\overline{x}) = 0\}$ and $I^0_{\overline{x}} := I_{\overline{x}} \cup \{0\}$, there exist multipliers $u_i \ge 0$ for $i \in I^0_{\overline{x}}$ and $u_j \in \mathbf{R}$ for $j \in J$, not all zero, such that

$$\begin{cases} u_i \langle Dg_i(\overline{x}), d\rangle = 0, & i \in I^0_{\overline{x}}, \\ \displaystyle\sum_{k \in I^0_{\overline{x}} \cup J} u_k Dg_k(\overline{x}) = 0, & \\ \displaystyle\sum_{k \in I^0_{\overline{x}} \cup J} u_k \left\langle d, D^2 g_k(\overline{x}) d\right\rangle \ge 0, & \end{cases} \tag{2}$$

see [2,3,16,18]. A SOCQ comes into play if one asks when u_0 can be chosen as 1 in order to have an objective-independent condition of Kuhn–Tucker type. Suppose d is a critical direction. Then under the above assumptions, the multiplier u_0 in (2) can be chosen as 1 if there holds (see [2]) the

- *Ben-Tal SOCQ*: $Dg_j(\overline{x})$, $j \in J$ are linearly independent, and there exists some $h \ne 0$ such that $\langle h, Dg_i(\overline{x})\rangle + \left\langle d, D^2 g_i(\overline{x}) d\right\rangle < 0$ for $i \in I(\overline{x}, d)$, and $\left\langle h, Dg_j(\overline{x})\right\rangle + \left\langle d, D^2 g_j(\overline{x}) d\right\rangle = 0$ for $j \in J$, where $i \in I(\overline{x}, d)$ if and only if $i \in I_{\overline{x}}$ and $\langle Dg_i(\overline{x}), d\rangle = 0$.

A similar SOCQ plays a role in infinite-dimensional settings [3]. If MFCQ holds at a local minimizer $\overline{x}$ of (P), then the Ben-Tal SOCQ is fulfilled at $\overline{x}$ for each critical direction d [2,3,25]. The Ben-Tal SOCQ does not guarantee that the same multiplier vector can be taken such that (2) is satisfied (with $u_0 = 1$) for *each* critical direction. However, under LICQ this so-called strong necessary condition holds, see [2]. For other SOCQs implying the strong necessary condition, see [12]. Classical textbooks like [11,19] often apply the

- *McCormick SOCQ* at $\overline{x} \in M$: Any vector d satisfying $\langle Dg_i(\overline{x}), d\rangle = 0$ for $i \in I_{\overline{x}} \cup J$ is the tangent of an arc $\alpha(\theta)$, twice differentiable, along which $g_i(\alpha(\theta)) \equiv 0$ for all $i \in I_{\overline{x}} \cup J$, where $\theta \in [0, \varepsilon]$, $\varepsilon > 0$. M denotes the constraint set of (P).

If McCormick's SOCQ, together with a first order CQ, holds, then a strong necessary condition is fulfilled in a much weaker form, namely, only for critical directions d satisfying additionally $\langle Dg_i(\overline{x}), d\rangle = 0$ for all $i \in I_{\overline{x}}$. Note that LICQ implies the McCormick SOSC [11], but the Kuhn–Tucker (first order) CQ does not imply the McCormick SOSC [11].

While strong necessary optimality conditions and the corresponding (restrictive) SOCQs rely on assumptions ensuring that active inequalities may be handled as equations, the approach which goes back to

A. Ben-Tal [2,3] is based on verification of optimality along curves that have a second order expansion, so-called *parabolic curves*. On the one hand, the concept of parabolic curves and related second order tangent sets requires only weak and 'natural' constraint qualifications, on the other hand, it leads to explicit second order optimality conditions involving second order terms of the original data.

This approach was continued in the remarkable paper [9] for problems of the type

$$(\widehat{\mathrm{P}}) \quad \begin{cases} \min & f(x) \\ \text{s.t.} & G(x) \in C \end{cases} \tag{3}$$

(with (smooth) *abstract constraints*), where G maps a Banach space X into a Banach space Y, f maps X into $\mathbf{R}$, f, G are twice continuously differentiable, and C is a nonempty closed convex subset of Y. The following discussion of the role of SOCQs in this context takes pattern from [5,6,9,21,24].

Assume $X = \mathbf{R}^n$. Define the *Lagrange function* by $L(x, u) := f(x)+\langle u, G(x)\rangle$. The *first order tangent set* $T_C(\overline{x})$ to C at $\overline{x}$ consists of all directions p satisfying $\operatorname{dist}(\overline{x} + tp, C) \sim o(t)$, the *second order tangent set* $T_C^2(\overline{x}, p)$ to C at $\overline{x}$ in direction p consists of all directions q satisfying

$$\operatorname{dist}\left(\overline{x} + tp + \frac{1}{2}t^2 q, C\right) \sim o(t^2),$$

where $\operatorname{dist}(x, C)$ denotes the Euclidean point-to-set distance from x to C, and $\varphi(t) \sim o(t^k)$ means the *Landau notation*, i. e., $\lim_{t \downarrow 0}\varphi(t)/t^k = 0$. Now the direction $h \in X$ is said to be *critical* at the feasible point $\overline{x}$ if and only if $DG(\overline{x})h \in T_C(G(\overline{x}))$ and $\langle Df(\overline{x}), h\rangle \leq 0$. Then one has [9]:

Theorem 1 *Let $\overline{x}$ be a local minimizer for f on the feasible set S defined by (3). If Robinson's CQ holds at $\overline{x}$, then for all critical directions h at $\overline{x}$,*

$$\sup_{u \in \Lambda(\overline{x})} \left\{\left\langle h, D_{xx}^2 L(\overline{x}, u)h\right\rangle - \sigma(u, Q_h)\right\} \geq 0, \tag{4}$$

where $\Lambda(\overline{x})$ is the set of Lagrange multipliers associated with $\overline{x}$ for $(\widehat{P})$, $Q_h := T_C^2(G(\overline{x}), DG(\overline{x})h)$, and $\sigma(\cdot, Q)$ is the support function of Q, $u \longmapsto \sigma(u, Q) = \sup_{q \in Q}\langle u, q\rangle$.

In order to have a narrow gap between necessary and sufficient optimality conditions, the following directional SOCQ, introduced in [4,6], is very important:

- *Directional SOCQ*: The set C is said to be *second order regular* at $y \in C$ in a direction $p \in T_C(y)$ and with respect to a linear mapping $M: X \to Y$, if for any sequence $y_n \in C$ of the form

$$y_n := y + t_n p + \frac{1}{2}t_n^2 q_n,$$

where $t \downarrow 0$, $q_n := Mw_n + a_n$, $\{a_n\}$ converging in Y, $w_n \in X$, $t_n w_n \to 0$, one has $\lim \operatorname{dist}(q_n, T_C^2(y, p)) = 0$.

If C is second order regular at y with respect to all such p, M, X then C is called second order regular at y.

For example, the polyhedral convex cone C in the setting of a nonlinear program (1) is second order regular. Further, the cones of positive and negative semidefinite matrices are second order regular (which is used for deriving optimality conditions in semidefinite programming [23]). For several other settings of C, for example in semi-infinite programming and composite optimization, the directional SOCQ has been specified in [6], where also the following crucial result can be found (known as the *equivalence theorem*):

Theorem 2 *If Robinson's CQ holds at a feasible point $\overline{x}$, and if for every critical direction h, the set C is second order regular at $G(\overline{x})$ in the direction $DG(\overline{x})h$ and with respect to $DG(\overline{x})(\cdot)$, then the inequality (4) holds strictly (i. e., with > 0) for nonzero critical directions h if and only if f satisfies, with some positive c, the quadratic growth condition $f(x) \geq f(\overline{x}) + c\,\|x - \overline{x}\|^2$ for feasible x near $\overline{x}$.*

In this sense, there is no gap between second order necessary and sufficient conditions. Similar second order optimality characterizations under some directional second order regularity were derived in [21] for so-called parabolically regular (extended-valued) functions. Note that the latter concept is weaker than that of the directional SOCQ.

If the abstract constraint (3) reduces to the nonlinear programming form (1), then the term $\sigma(u, Q_h)$ in (4), called 'shifting term' or 'σ-term', vanishes. Hence, the equivalence theorem reduces to a well-known fact for smooth nonlinear programs (see, e. g., [2,25]), since MFCQ and Robinson's CQ coincide in this situation.

In contrast to this, for more general settings, $\sigma(u, Q_h)$ is essential. Conditions which allow a concrete representation of the σ-term can be considered as SOCQs. For standard semi-infinite programs with compact in-

dex set I (i. e., C is the cone of nonnegative continuous functions on I), this has been done in [5,24]. Conditions which ensure the local reduction of a semi-infinite program to a standard nonlinear program ('reduction approach') also allow a concrete representation of $\sigma(u, Q_h)$, for standard semi-infinite programs see, e. g., [13,22], for generalized semi-infinite programs see, e. g., [14,17].

The directional SOCQ defined above (or some stronger versions, respectively), together with a first order directional CQ, have been essentially used (e. g., in [5,7,8]) in sensitivity and stability analysis of problem $(\widehat{P})$ under perturbations along certain directions: second order expansion of optimal values, first order expansion of optimal solutions, differential stability, Lipschitz stability, etc.

Remarks

In this article, the focus was on smooth problems. Though the notion SOCQ is not often used in *nonsmooth* optimization, the study of higher-order tangent sets and their influence in optimality conditions of higher order could be considered as an analogy to SOCQs. For many material in this direction see, e. g., [1,21]. A special case between smooth and nonsmooth programs are so-called $C^{1,1}$ *optimization problem* (i. e., problems of type (P), but the data are differentiable with locally Lipschitzian gradients). A SOCQ for this class of problems is a stronger variant of the Abadie CQ: the Bouligand (contingent) cone has to coincide with the linearization cone [15]. This SOSC, together with some first order CQ, is again of interest in deriving necessary second order optimality conditions, see [10,15].

See also

- ► Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- ► First Order Constraint Qualifications
- ► Inequality-constrained Nonlinear Optimization
- ► Kuhn–Tucker Optimality Conditions
- ► Lagrangian Duality: Basics
- ► Rosen's Method, Global Convergence, and Powell's Conjecture
- ► Saddle Point Theory and Optimality Conditions
- ► Second Order Optimality Conditions for Nonlinear Optimization

References

1. Aubin J-P, Frankowska H (1990) Set-valued analysis. Birkhäuser, Basel
2. Ben-Tal A (1980) Second-order and related extremality conditions in nonlinear programming. J Optim Th Appl 31:143–165
3. Ben-Tal A, Zowe J (1982) A unified theory of first and second order conditions for extremum problems in topological vector spaces. Math Program Stud 19:39–76
4. Bonnans JF, Cominetti R (1996) Perturbed optimization in Banach spaces I, II, III. SIAM J Control Optim 34:1151–1171; 1172–1189; 1555–1567
5. Bonnans JF, Cominetti R, Shapiro A (1998) Sensitivity analysis of optimization problems under abstract constraints. Math Oper Res 23:806–831
6. Bonnans JF, Cominetti R, Shapiro A (1999) Second order optimality conditions based on parabolic second order tangent sets. SIAM J Optim 9:466–492
7. Bonnans JF, Ioffe AD, Shapiro A (1992) Expansion of exact and approximate nonlinear programming. In: Oettli W, Pallaschke D (eds) Advances in Optimizations. Springer, Berlin, pp 103–117
8. Bonnans JF, Shapiro A (2000) Perturbation analysis of optimization problems. Springer, Berlin
9. Cominetti R (1990) Metric regularity, tangent sets and second-order optimality conditions. Appl Math Optim 21:265–287
10. Dien PH, Sach PH (1989) Second-order optimality conditions for the extremal problem under inclusion constraints. Appl Math Optim 20:71–80
11. Fiacco AV, McCormick GP (1968) Nonlinear programming: Sequential unconstrained minimization techniques. Wiley, New York
12. Hestenes MR (1975) Optimization theory – The finite-dimensional case. Wiley, New York
13. Hettich R, Jongen HTh (1978) Semi-infinite programming: conditions of optimality and applications. In: Stoer J (ed) Optimization Techniques 2. Springer, Berlin, pp 1–11
14. Hettich R, Still G (1995) Second-order optimality conditions for generalized semi-infinite programming problems. Optim 34:195–211
15. Hiriart-Urruty J-B, Strodiot JJ, Nguyen V Hien (1984) Generalized Hessian matrix and second order optimality conditions for problems with $C^{1,1}$-data. Appl Math Optim 11:43–56
16. Ioffe AD (1979) Necessary and sufficient conditions for a local minimum. 3: Second order conditions and augmented duality. SIAM J Control Optim 17:266–288
17. Klatte D (1992) Stability of stationary solutions in semi-infinite optimization via the reduction approach. In: Oettli W, Pallaschke D (eds) Advances in Optimization. Springer, Berlin, pp 155–170
18. Levitin ES, Miljutin AA, Osmolovskii NP (1974) On conditions for a local minimum in a problem with constraints. In:

Mitjagin BS (ed) Mathematical Economics and Functional Analysis. Nauka, Moscow, pp 139–202, in Russian
19. Luenberger DE (1973) Introduction to linear and nonlinear programming. Addison-Wesley, Reading
20. McCormick GP (1967) Second-order conditions for constrained minima. SIAM J Appl Math 15:641–652
21. Rockafellar RT, Wets RJ-B (1997) Variational analysis. Springer, Berlin
22. Shapiro A (1985) Second-order derivatives of extremal-value functions and optimality conditions for semi-infinite programs. Math Oper Res 10:207–219
23. Shapiro A (1997) First and second order analysis of nonlinear semidefinite programs. Math Program 77:301–320
24. Shapiro A (1998) First and second order optimality conditions and perturbation analysis of semi–infinite programming problems. In: Reemtsen R, Rückmann J-J (eds) Semi-Infinite Programming. Kluwer, Dordrecht, pp 103–133
25. Still G, Streng M (1996) Optimality conditions in smooth nonlinear programming. J Optim Th Appl 90:483–515

Second Order Optimality Conditions for Nonlinear Optimization

Katta G. Murty
Department IOE, University Michigan,
Ann Arbor, USA

MSC2000: 90C39, 90C26

Article Outline

Keywords
Optimality Conditions for the Unconstrained Minimization Problem (1)
Optimality Conditions for the Equality Constrained Minimization Problem (2)
Optimality Conditions for the General Constrained Optimization Problem (3)
See also
References

Keywords

Nonlinear optimization problem; Convex and nonconvex programming problems; Local minimum; Global minimum; Stationary point; Constraint qualification; Regularity condition; Mangasarian–Fromovitz constraint qualification; Necessary optimality conditions; Sufficient optimality conditions; KKT conditions; First order and second order optimality conditions

The quest for *optimality conditions* for *nonlinear optimization problems* started a long time ago, however, until calculus was invented there was little progress. Calculus has enriched and accelerated this endeavor significantly.

In the following discussion $x = (x_1, \ldots, x_n)^T$ denotes the column vector of decision variables whose optimal values need to be determined. All functions are assumed to be continuous real valued functions of x. When discussing the first (second) order optimality conditions, we make the assumption that all functions are continuously differentiable (twice continuously differentiable).

For any real valued function $f(x)$, we denote the *gradient vector* of $f(x)$ at $\overline{x}$, $(\partial f(\overline{x})/\partial x_1, \ldots, \partial f(\overline{x})/\partial x_n)$, written as a row vector, by $\nabla_x f(\overline{x})$ and the $n \times n$ *Hessian matrix* of $f(x)$ at $\overline{x}$, $(\partial^2 f(\overline{x})/\partial x_i \partial x_j)$, by $\nabla^2_{xx} f(\overline{x})$.

We discuss the nonlinear optimization problem in terms of 'minimization' of the objective function. Instead, if an objective function $F(x)$ is required to be 'maximized', this problem is equivalent to minimizing $-F(x)$ subject to the same constraints. Because of this, we state all our results in terms of minimization problems.

Nonlinear optimization problems can be classified into three broad types, which are of the following forms:

- *Unconstrained minimization*

$$\begin{cases} \min & \theta(x) \\ \text{s.t.} & x \in \mathbb{R}^n. \end{cases} \tag{1}$$

- *Equality-constrained optimization*

$$\begin{cases} \min & \theta(x) \\ \text{s.t.} & g_i(x) = 0, \quad i = 1, \ldots, m. \end{cases} \tag{2}$$

- *General constrained optimization*

$$\begin{cases} \min & \theta(x) \\ \text{s.t.} & g_i(x) \begin{cases} = 0, & i = 1, \ldots, m, \\ \geq 0, & i = m+1, \ldots, m+p. \end{cases} \end{cases} \tag{3}$$

The general problem (3) is said to be a *convex programming problem* [2,3] if the objective function to

be minimized, $\theta(x)$, is *convex*; the equality-constraint functions $g_i(x)$, $i = 1, \ldots, m$, are *affine*; and the inequality ($\geq$) constraint functions $g_i(x)$, $i = m + 1, \ldots, m + p$ are *concave*; a *nonconvex programming problem* otherwise.

In linear programming, we only talk about an *optimum solution*, but in nonlinear optimization, we need to consider different types of optimum solutions. In minimization problems the feasible solution $\overline{x}$ is said to be a:

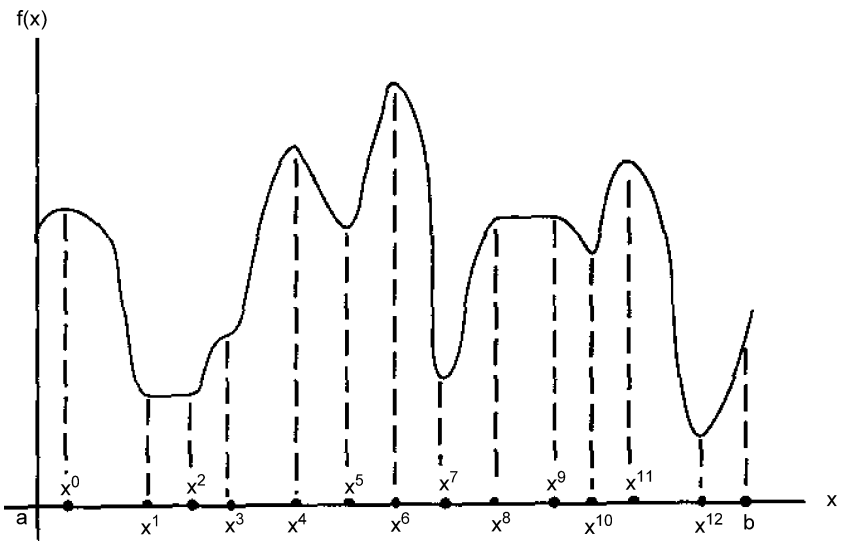
- *local* (or *relative*) *minimum* if $\theta(x) \geq \theta(\overline{x})$ for all feasible x satisfying $\|x - \overline{x}\| \leq \epsilon$ for some positive ϵ.
- *strong* (or *strict*) *local* (or *relative*) *minimum* if $\theta(x) > \theta(\overline{x})$ for all feasible $x \neq \overline{x}$ satisfying $\|x - \overline{x}\| \leq \epsilon$ for some positive ϵ.
- *global minimum* if $\theta(x) \geq \theta(\overline{x})$ for all feasible x.
- *stationary point* if it satisfies one of the necessary conditions for a local minimum.

Corresponding concepts (weak or strong local maximum, global maximum) for maximization problems are defined analogously. We illustrate these concepts in Fig. 1 for the problem of minimizing the graphed function $f(x)$ of a single variable $x \in \mathbf{R}^1$ in the interval $a \leq x \leq b$. The points a, x^5, x^7, x^{10}, x^{12} are strong local minima; x^0, x^4, x^6, x^{11}, b are strong local maxima; x^{12} is the global minimum, and x^6 is the global maximum. x^1, x^2 are weak local minima; and x^8, x^9 are weak local maxima. x^3 is a stationary point even though it is neither a local maximum nor a local minimum. In each of the intervals $x^1 \leq x \leq x^2$, and $x^8 \leq x \leq x^9$, $f(x)$ is a constant; and every point in the interior of these intervals (i. e., points x satisfying $x^1 < x < x^2$ or $x^8 < x < x^9$) is both a weak local minimum and a weak local maximum.

In linear programming all local minima are global minima, hence we only talk about an 'optimum solution' there. As seen above, in nonlinear programming models these concepts could be different.

The original intent of the problem is of course to find a global minimum. Using existing algorithms, this is possible with reasonable efficiency only for convex programming problems (which includes linear programs and convex quadratic programs as special cases). As discussed in [3,4], for general nonconvex programming problems, even finding a local minimum is hard; existing efficient algorithms can at best guarantee convergence to a stationary point on such problems.

Unfortunately, there are no known useful conditions that can efficiently characterize a global minimum for the general nonlinear programming problem. We only know some necessary, and some sufficient conditions for a local minimum for this general problem. In convex programming problems however, every local minimum is a global minimum; so for this nice class of problems we have useful necessary and sufficient conditions for a global minimum.



Second Order Optimality Conditions for Nonlinear Optimization, Figure 1

Optimality Conditions for the Unconstrained Minimization Problem (1)

These were developed first for one-dimensional minimization problems in the 17th century as Newton was developing calculus, and very soon after extended to multidimensional minimization problems. These conditions are:

- *First order necessary conditions* for $\overline{x}$ to be a local minimum: $\nabla_x \theta(\overline{x}) = 0$.
- *Second order necessary conditions* for $\overline{x}$ to be a local minimum: $\nabla_x \theta(\overline{x}) = 0$ and $\nabla^2_{xx}(\theta(\overline{x}))$ is positive semidefinite (*PSD*).
- *Second order sufficient conditions* for $\overline{x}$ to be a strict local minimum: $\nabla_x \theta(\overline{x}) = 0$, and $\nabla^2_{xx}(\theta(\overline{x}))$ is positive definite (*PD*).
- *Necessary and sufficient conditions* for $\overline{x}$ to be a global minimum if $\theta(x)$ is convex: $\nabla_x \theta(\overline{x}) = 0$.

The computational effort needed to check whether a given square matrix is PSD or PD is $O(n^3)$. Hence, given $\overline{x}$, each of the above conditions can be checked efficiently.

When $\theta(x)$ is nonconvex, there is a slight gap between the necessary and sufficient conditions for a local minimum when $\nabla_x \theta(\overline{x}) = 0$ and $\nabla^2_{xx}(\theta(\overline{x}))$ is PSD and not PD. In this case we are unable to conclude that either $\overline{x}$ is a local minimum, or that it is not. To bridge this gap, more complicated conditions involving higher order derivatives are needed, but they are impractical, particularly when n is large.

Optimality Conditions for the Equality Constrained Minimization Problem (2)

Inspired by the study of problems in mechanics, these conditions, which form the foundation of nonlinear programming theory, were developed in the 18th and 19th centuries. Major results were obtained by L. Euler and J.L. Lagrange and were first published in a book written by Lagrange in 1788.

The necessary conditions are derived under a condition on the constraints in (2) known as a *constraint qualification* (*CQ*). A well known CQ is known as the *regularity condition*.

The feasible solution $\overline{x}$ for (2) is said to satisfy the regularity condition (and hence called a *regular point* for (2)) if $\{\nabla_x g_1(\overline{x}), \ldots, \nabla_x g_m(\overline{x})\}$ is linearly independent.

The optimality conditions for a feasible solution $\overline{x}$ to be a local minimum for (2) are:

- *First order necessary conditions* for $\overline{x}$ to be a local minimum for (2): If $\overline{x}$ is a local minimum for (2), and either all the constraints are linear constraints, or $\overline{x}$ is a regular feasible solution, there exists $\overline{\pi} = (\overline{\pi}_1, \ldots, \overline{\pi}_m)$ such that

$$\nabla_x \theta(\overline{x}) = \sum_{i=1}^{m} \overline{\pi}_i \nabla_x g_i(\overline{x}).$$

  The vector $\overline{\pi}$ in the above condition is known as the *Lagrange multiplier vector*. The function $L(x, \overline{\pi}) = \theta(x) - \sum_{i=1}^{m} \overline{\pi}_i g_i(x)$ is known as the *Lagrangian function* for (2) with Lagrange multiplier vector $\overline{\pi}$. The above necessary condition can also be written as $\nabla_x L(\overline{x}, \overline{\pi}) = 0$.
- *Second order necessary conditions* for $\overline{x}$ to be a local minimum for (2): If $\overline{x}$ is a local minimum for (2), and either all the constraints are linear constraints, or $\overline{x}$ is a regular feasible solution, there exists $\overline{\pi} = (\overline{\pi}_1, \ldots, \overline{\pi}_m)$ such that

$$\nabla_x \theta(\overline{x}) = \sum_{i=1}^{m} \overline{\pi}_i \nabla_x g_i(\overline{x})$$

  and

$$y^\top \nabla^2_{xx}(L(\overline{x}, \overline{\pi}))y \geq 0 \quad \text{for all } y \in T,$$

  where $T = \{y\colon\ \nabla_x g_i(\overline{x})y = 0 \text{ for all } i = 1, \ldots, m\}$.
- *Sufficient condition* for $\overline{x}$ to be a strict local minimum for (2): If $\overline{x}$ is a feasible solution for (2), and there exists a $\overline{\pi} = (\overline{\pi}_1, \ldots, \overline{\pi}_m)$ such that

$$\nabla_x \theta(\overline{x}) = \sum_{i=1}^{m} \overline{\pi}_i \nabla_x g_i(\overline{x})$$

  and

$$y^\top \nabla^2_{xx}(L(\overline{x}, \overline{\pi}))y > 0 \quad \text{for all } 0 \neq y \in T,$$

  where T is defined above, then $\overline{x}$ is a strict local minimum for (2).

Given a regular feasible solution $\overline{x}$, the existence of a Lagrange multiplier vector $\overline{\pi}$ which satisfies either of the necessary conditions given above together with $\overline{x}$, can be checked efficiently.

Here again, in nonconvex programs, there is a small gap between the necessary and sufficient optimality

conditions for a local minimum. If the second order necessary condition holds at $\overline{x}$, but not the sufficient condition, then using these conditions we are not able to guarantee that either $\overline{x}$ is a local minimum, or that it is not.

Optimality Conditions for the General Constrained Optimization Problem (3)

Fundamental ideas about necessary optimality conditions for nonlinear optimization problems including some inequality constraints, have been investigated beginning with J.B.J. Fourier in 1798; and later by A. Cournot, M. Ostrogradsky, C.F. Gauss, J. Farkas, and G. Hamel among others. Rigorous development of these conditions in the form we know them today has been completed by W. Karush in 1939, and later in essentially the same form by H.W. Kuhn and A.W. Tucker in 1951.

Given a feasible solution $\overline{x}$ for (3), the ith constraint in (3) is said to be *active* at $\overline{x}$ if $g_i(\overline{x}) = 0$, *inactive* otherwise. Thus all equality constraints are active at every feasible solution, and an inequality constraint is active at a feasible solution if it holds as an equation there. We will denote the index set of active constraints at a feasible solution $\overline{x}$, $\{i\colon\ g_i(\overline{x}) = 0\}$ by $B(\overline{x})$.

Necessary optimality conditions are derived under a CQ. There are several CQ, some weaker than the others. The principal ones are:

- *Regularity condition*: The feasible solution $\overline{x}$ satisfies this CQ and is therefore called a *regular point* for (3) if $\{\nabla_x g_i(\overline{x})\colon\ i \in B(\overline{x})\}$ is linearly independent. This condition can be checked efficiently.
- *First order CQ*: The feasible solution $\overline{x}$ for (3) satisfies this CQ if for each

$$y \in \left\{ y\colon \begin{array}{l} \nabla_x g_i(\overline{x}) y = 0, \\ \quad i = 1, \ldots, m; \\ \nabla_x g_i(\overline{x}) y \geq 0, \\ \quad i \in \{m+1, \ldots, m+p\} \cap B(\overline{x}) \end{array} \right\},$$

  y is the tangent direction to a differentiable curve emanating from $\overline{x}$ and lying in the feasible region. This condition is hard to check.
- *Second order CQ*: The feasible solution $\overline{x}$ for (3) satisfies this CQ if for each $y \in \{y\colon\ \nabla_x g_i(\overline{x}) y = 0,\ i \in B(\overline{x})\}$, there exists a twice differentiable curve emanating from $\overline{x}$ and lying in the region $\{x\colon\ g_i(x) = 0,\ i \in B(\overline{x})\}$, for which y is the tangent direction at $\overline{x}$. This condition is hard to check.
- *Mangasarian–Fromovitz CQ*: The feasible solution $\overline{x}$ for (3) satisfies this CQ if the set

$$\{d\colon\ \nabla_x g_i(\overline{x}) d = 0,\ i = 1, \ldots, m\} \cap \left\{ d\colon \begin{array}{l} \nabla_x g_i(\overline{x}) d > 0, \\ \text{for } i \in \{m+1, \ldots, m+p\} \cap B(\overline{x}) \end{array} \right\}$$

  is nonempty, and $\{\nabla_x g_i(\overline{x})\colon\ i = 1, \ldots, m\}$ is linearly independent. This condition can be checked efficiently.

The Lagrangian function for (3) with Lagrange multiplier vector $\overline{\pi} = (\overline{\pi}_1, \ldots, \overline{\pi}_{m+p})$ is $L(x, \overline{\pi}) = \theta(x) - \sum_{i=1}^{m+p} \overline{\pi}_i g_i(x)$. The optimality conditions for a feasible solution $\overline{x}$ to be a local minimum for (3) are:

- *First order necessary conditions* for $\overline{x}$ to be a local minimum for (3): If $\overline{x}$ is a local minimum for (3), and either all the constraints are linear constraints, or $\overline{x}$ satisfies the regularity, or the 1st order, or the Mangasarian–Fromovitz CQs, there exists a Lagrange multiplier vector $\overline{\pi} = (\overline{\pi}_1, \ldots, \overline{\pi}_{m+p})$ such that

$$\begin{aligned} &\nabla_x \theta(\overline{x}) = \sum_{i=1}^{m+p} \overline{\pi}_i \nabla_x g_i(\overline{x}), \\ &\overline{\pi}_i \geq 0 \quad \text{for } i \in \{m+1, \ldots, m+p\}, \\ &\overline{\pi}_i g_i(\overline{x}) = 0 \quad \text{for } i \in \{m+1, \ldots, m+p\}. \end{aligned}$$

  In the literature, these conditions are commonly referred to as the *KKT conditions* (*Karush–Kuhn–Tucker conditions*). Given $\overline{x}$, checking for the existence of a Lagrange multiplier vector which together with $\overline{x}$ satisfies these conditions can be posed as a linear programming problem and solved efficiently.
- *Second order necessary conditions* for $\overline{x}$ to be a local minimum for (3): If $\overline{x}$ is a local minimum for (3), and either all the constraints are linear constraints, or $\overline{x}$ satisfies the regularity, or the 2nd order, or the Mangasarian–Fromovitz CQs, there exists a Lagrange multiplier vector $\overline{\pi} = (\overline{\pi}_1, \ldots, \overline{\pi}_{m+p})$ such that

$$\begin{aligned} &\nabla_x \theta(\overline{x}) = \sum_{i=1}^{m+p} \overline{\pi}_i \nabla_x g_i(\overline{x}), \\ &\overline{\pi}_i \geq 0 \quad \text{for } i \in \{m+1, \ldots, m+p\}, \\ &\overline{\pi}_i g_i(\overline{x}) = 0 \quad \text{for } i \in \{m+1, \ldots, m+p\} \end{aligned}$$

and

$$y^\top(\nabla^2_{xx}L(\overline{x},\overline{\pi}))y \geq 0 \quad \text{for all } y \in T_1,$$

where

$$T_1 = \{y\colon\ \nabla_x g_i(\overline{x})y = 0,\ i \in B(\overline{x})\}.$$

Given $\overline{x}$, the existence of a Lagrange multiplier vector which together with $\overline{x}$ satisfies these conditions can be checked efficiently using linear programming and PSD checking methods.

- *Sufficient conditions* for $\overline{x}$ to be a strict local minimum for (3): If the feasible solution $\overline{x}$ for (3) is such that there exists a Lagrange multiplier vector $\overline{\pi} = (\overline{\pi}_1, \ldots, \overline{\pi}_{m+p})$ which together with $\overline{x}$ satisfies

$$\nabla_x\theta(\overline{x}) = \sum_{i=1}^{m+p} \overline{\pi}_i \nabla_x g_i(\overline{x}),$$
$$\overline{\pi}_i \geq 0 \quad \text{for } i \in \{m+1, \ldots, m+p\},$$
$$\overline{\pi}_i g_i(\overline{x}) = 0 \quad \text{for } i \in \{m+1, \ldots, m+p\}$$

and

$$y^\top(\nabla^2_{xx}L(\overline{x},\overline{\pi}))y > 0 \quad \text{for all } y \in T_2,$$

where

$$T_2 = \{y : \nabla_x g_i(\overline{x})y = 0 \text{ for } i \in \{1, \ldots, m\} \cup (\{i\colon\ \overline{\pi}_i > 0\} \cap \{m+1, \ldots, m+p\} \cap B(\overline{x}))$$

and

$$\nabla_x g_i(\overline{x})y \geq 0 \quad \text{for all } i \text{ in the set } \{i\colon\ \overline{\pi}_i = 0\} \cap \{m+1, \ldots, m+p\} \cap B(\overline{x})\}.$$

If (3) is a nonconvex program, verifying whether the last condition among the sufficient optimality conditions holds is hard (K.G. Murty and S.N. Kabadi [4] have shown that the simple special case of this problem: checking whether $x^\top Dx \geq 0$ for all $x \geq 0$ is hard if D is not a PSD matrix).

A weaker sufficient condition for $\overline{x}$ to be a strict local minimum for (3) is obtained by replacing T_2 in the above condition by the set

$$T_3 = \{y : \nabla_x g_i(\overline{x})y = 0 \text{ for all } i \in \{1, \ldots, m\} \cup (\{i\colon\ \overline{\pi}_i > 0\} \cap \{m+1, \ldots, m+p\} \cap B(\overline{x}))\}.$$

This weaker sufficient condition can be checked efficiently.

- *Necessary and sufficient conditions* for $\overline{x}$ to be a global minimum of (3) if it is a convex program: If (3) is a convex program, then the first order necessary conditions stated above are necessary and sufficient for a given feasible solution to be a global minimum.

If (3) is a nonconvex program [2,3], the gap between the second order necessary conditions, and the sufficient conditions for a local minimum is small, but in case the problem under consideration falls in this gap, we are unable to confirm that either $\overline{x}$ is or is not a local minimum using these conditions. [1,5]

See also

- ▶ Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- ▶ First Order Constraint Qualifications
- ▶ Inequality-constrained Nonlinear Optimization
- ▶ Kuhn–Tucker Optimality Conditions
- ▶ Lagrangian Duality: Basics
- ▶ Rosen's Method, Global Convergence, and Powell's Conjecture
- ▶ Saddle Point Theory and Optimality Conditions
- ▶ Second Order Constraint Qualifications

References

1. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming theory and algorithms, 2nd edn. Wiley, New York
2. Fiacco AV, McCormick GP (1968) Nonlinear programming: sequential unconstrained minimization techniques. Wiley, New York
3. Murty KG (1988) Linear complementarity, linear and nonlinear programming. Heldermann, Berlin
4. Murty KG, Kabadi SN (1987) Some NP-complete problems in quadratic and nonlinear programming. Math Program 39:117–129
5. Prékopa A (1980) On the development of optimization theory. Amer Math Monthly 87:527–542

Selection of Maximally Informative Genes

Ioannis P. Androulakis[1], Eric Yang[2]

[1] Department of Chemical and Biochemical Engineering, Rutgers University, Piscataway, USA

[2] Department of Biomedical Engineering, Rutgers University, Piscataway, USA

Article Outline

Introduction
Formulations
A Mixed Integer Formulation for Gene Selection
A Mixed Integer Formulation for Feature Selection and Classifier Complexity Minimization
Cases
Gene Selection and Complexity Minimization
Conclusions
References

Introduction

Gene expression microarray experiments have been celebrated as a revolution in biology, attracting significant interest because they allow for the analysis of the combined effects of numerous genetic and environmental components. This global approach will allow a fundamental shift from "… piece-by-piece to global analysis and from hypothesis driven research to discovery-based formulation and subsequent testing of hypotheses …" [7,8,10,15,24,29,41]. One of the major challenges is to extract in a systematic and rigorous way the biologically relevant components from the array experiments in order to establish meaningful connections linking genetic information to cellular function. Because of the significant amount of experimental information that is generated (expression levels of thousands of genes), computer-assisted knowledge extraction is the only realistic alternative for managing such an information deluge. A number of excellent publications have focused on different aspects of gene expression experiments [1,2,3,6,15,20,25,32,35,36,38]. Novel computational approaches that exploit large warehouses of gene expression data have been identified as major enablers for realizing fully the potential of this technology [4]. A significant concern with microarray analyzes is the ability to assign a certain level of significance to smaller subsets of genes whose expression patterns could potentially indicate a more direct involvement in the biological process under study.

The identification of smaller sets of "informative genes" is a manifestation of a boarder problem in the machine learning community, namely the problem of "feature selection". Feature selection has received increased attention with the recent advances in functional genomics that resulted in the creation of high-dimensional feature sets. A number of recent publications [11,13,20,47] have devised various approaches for extracting critical, differentially expressed genes in a systematic manner. The advantage of multivariate methods is that they take into account collaborative effects of gene expression activities. Reducing the number of measured variables reduces the degrees of freedom, hence avoiding pointless over-fitting. Too few genes will not discriminate or predict; too many genes might introduce noise to the model rather than information. Therefore, the identification of informative genes is a significant component of an integrated, computer-assisted analysis of array experiments. In most cases the question of identifying differentially expressed genes is restated as a hypothesis-testing problem in which the null hypothesis of no association between expression levels and responses of interest is tested [15].

One of the reasons for performing gene selection, which will be the focus of this chapter, is in tissue classification. Samples from multiple cell types (for example different cancer types, cancerous and normal cells etc.) are comparatively analyzed using microarray gene expression measurements. The question therefore becomes how to identify which genes provide consistent signatures that distinctly characterize the different classes. The problem can be viewed as either a supervised classification problem in which the classes are already known, or as an unsupervised clustering problem in which we attempt to identify the classes contained within the data. In gene selection, the computational problem is equivalent to that of feature selection in multidimensional data sets. Identifying the minimum number of gene markers is however critical because this reduced set can provide information about the biology behind the experiment as well as define the basis for future therapeutic agents. Typical examples are discussed in [12,20,23,33].

The obvious way to simplify the complexity of the model (classifier) is to minimize the number of degrees of freedom used for building the model. Traditionally this problem is approached in a stepwise fashion as a feed-forward or backward feature selection process [30]. This approach is widely used in analyzing microarray data [20]. The problem however is that synergistic effects are not properly captured, and it is often difficult to come to a conclusion as to what the actual number of informative features is [27]. There-

fore, a search algorithm must explicitly account in the objective for the actual number of features used in the model. For linear discriminant models, concepts such as the Akaike and Bayesian Information Criteria (AIC, BIC) have been used, albeit in a stepwise fashion. In either case the objective (maximum likelihood estimation) is augmented to account for the number of features (variables) used in the model. Recently, researchers incorporated the number of features in their search objective [19]. Therefore, we treat the total number of features used in the model explicitly as one of our complexity criterion. The definition of model complexity is not an easy task. When the decision boundary is a hyper-plane, it is rather straightforward to require a minimum number of non-zero coefficients (AIC and BIC criteria discussed earlier). In nonlinear classifiers, or when the separations are defined as the intersection of multiple hyper-planes, it is not obvious how to decide which model is "simpler" as the notion of simplicity is ill defined.

In this article we will discuss two integer optimization formulations for selecting informative genes. The first formulation is based on a mixed integer linear formulation that while minimizing the classification error, attempts to simultaneously minimize the complexity of the classifier by controlling directly the number of features used. The second formulation expands the complexity quantification attempting to control both the number of features and the complexity of the classifier while maximizing its performance.

Formulations

A Mixed Integer Formulation for Gene Selection

A mixed-integer linear formulation was recently proposed [46]. Feature selection is always considered within the framework of a given analysis. This could be model development/fitting, classification, clustering etc. In other words we want to extract the minimum number of required independent variables necessary to perform a particular task. Therefore, an objective measuring the "goodness of fit" will be required. The parameters associated with the model naturally define a continuous optimization problem. The notion of selecting a subset of variables, out of a superset of possible alternatives, naturally lends itself to a combinatorial (integer) optimization problem. Therefore, depending on the model used to describe the data the problem of feature selection will end up being a mixed integer (non) linear optimization problem. Furthermore, this problem is a multi-criteria optimization since one wishes to simultaneously minimize the model error and the number of features used. Let m denote the number of observations for a two-class problem such that k and l denote the number of samples in each class (for example number of benign and cancerous cells respectively). We also denote as I_1 and I_2 the indices of the corresponding samples and $I = I_1 \cup I_2$ denotes the entire set of samples. Finally the set J denotes the set of all genes recorded in the observations and $J' \subset J$ denotes the set of genes (features) that are required to develop an accurate model. The expression data is presented in the form x_{ij}, $i = 1, \ldots, I$, $j = 1, \ldots, J$. A linear classifier is constructed as:

$$\begin{cases} \beta_0 + \sum_{j \in J} \beta_j x_{ij} < 0\,, & i \in I_1 \\ \beta_0 + \sum_{j \in J} \beta_j x_{ij} > 0\,, & i \in I_2 \end{cases}.$$

However, because the observations are not, in general, perfectly separable by a linear model a goal programming formulation can be proposed whose goal is to estimate the coefficients that minimize the deviations from the classifier model, Fig. 1.

In order to minimize the number of variables used in the classifier, thereby extracting the most relevant features for the specific linear model, binary variables need to be introduced to define whether a particular

$$\begin{aligned} &\min : \sum_{i \in I_1} d_i^1 + \sum_{i \in I_2} d_i^2 \\ &\text{s.t.} \\ &\beta_0 + \sum_{j \in J} \beta_j x_{ij} - d_i^1 + d_i^2 = -\sigma, i \in I_1 \\ &\beta_0 + \sum_{j \in J} \beta_j x_{ij} - d_i^1 + d_i^2 = \sigma, i \in I_2 \\ &\beta_j \in \mathbb{R} \\ &d_i^1, d_i^2 \in \mathbb{R}^+ \end{aligned}$$

Selection of Maximally Informative Genes, Figure 1
Optimization-based classification model

$$\min : \sum_{i \in I_1} d_i^1 + \sum_{i \in I_2} d_i^2$$
$$\text{s.t.}$$
$$\beta_0 + \sum_{j \in J} \beta_j x_{ij} - d_i^1 + d_i^2 = -\sigma, i \in I_1$$
$$\beta_0 + \sum_{j \in J} \beta_j x_{ij} - d_i^1 + d_i^2 = \sigma, i \in I_2$$
$$\sum_{j \in J} y_j \leq \epsilon$$
$$\beta_j - M y_j \leq 0$$
$$-\beta_j - M y_j \leq 0$$
$$\beta_j \in \mathbb{R}, j \in J$$
$$d_i^1, d_i^2 \in \mathbb{R}^+, i \in I$$
$$y_j \in 0, 1$$

Selection of Maximally Informative Genes, Figure 2
Mixed integer formulation of the feature selection problem

variable is used in the model or not. Therefore:

$$y_j = \begin{cases} 1\,, & j \in J' \\ 0\,, & j \notin J' \end{cases} \quad .$$

The number of "active" genes can therefore be constrained (that is introduced parametrically in the formulation in order to avoid the solution of a multi-criteria optimization problem). According to the ε-constraint method one additional constraint of the form: $\sum_{j \in J'} y_j \leq \varepsilon$ is introduced. The complete MIP formulation is shown in Fig. 2.

A Mixed Integer Formulation for Feature Selection and Classifier Complexity Minimization

An interesting idea was recently introduced [45] in the context of oblique multicategory classification trees, whereby the class assignment is modeled through the use of the concept of purity of a partition. Ideally, one wishes to construct a multivariate classifier in such a way that each "partition" is occupied by elements of a single class (orthogonal partitions). However, this formulation is faced with a number of complexities. First, it is highly non-linear, second it does not perform feature selection and finally it builds classifiers sequentially, in the sense of the one-against-all concept. However, the purity concept introduces a very intelligent way of quantifying the ability of a classifier to partition the data. Furthermore, with proper modifications to be discussed shortly, it allows the quantification of the complexity of the classifier.

We will discuss the basic elements of recently proposed approach [53] which can be effectively generalized in the context of a mixed-integer optimization. This generalization will be used to develop a general framework of oblique multi-category trees to address the question of how to build simple, yet informative, classifiers that simultaneously perform informative feature selection. We assume that we are given the ensemble of gene expression data in the form of an f-dimensional vectors belonging to k distinct classes. The question is to identify how many, and which, of these f features are critical for the construction of a simplified, yet informative, classifier. An oblique multi-category classifier is defined by the intersection of a number of planes. We term these intersections "partitions", $\pi = 2^p$ where p is the number of planes. We define complexity as the number of occupied partitions that are required to properly classify the data. $q_{n,\pi}$ is a binary variable indicating whether point "n" belongs in a partition π. The total number of points of class k in partition π is termed $\sigma_{k,\pi}$ whereas $v_{k,\pi}$ denotes the fraction of points of class k in said partition. Finally, $y_{k,\pi}$ is a binary variable which is 1 if that partition contains even a single point from class k, and 0 otherwise. The location of a point relative to a particular plane is defined according to the binary variable $z_{n,p}$ which is 1 if the point is below the plane, and 0 otherwise. This variable is a critical one since it basically defines all the auxiliary variables in the formulation. Finally s_f is a binary variable indicating whether feature "f" is used in the construction of the classifier. Give that p planes create 2^p partitions, we wish to identify the partitions, and the corresponding spatial distribution of points in the reduced space defined by s_f which will create the "purest" possible partition. We model this by analyzing the product $y_{k,\pi} v_{k',\pi}, k \neq k'$. In order to account for non-linearly classifiable problems we are basically looking for partitions that satisfy the set of constraints depicted in Fig. 3.

The modeling idea is that (i) the partition contains no point of class "k" and the maximum numbers of "k" (empty partition), (ii) the partition contains points of class "k" and contains the minimum number of points

$$y_{k,\pi}\nu_{k',\pi} \leq E \Rightarrow \begin{cases} y_{k,\pi} = 0 \wedge \nu_{k',\pi} \leq 1 \\ y_{k,\pi} = 1 \wedge \nu_{k',\pi} \leq E \\ y_{k,\pi} = 0 \wedge \nu_{k',\pi} \leq E \end{cases}$$

or

$$y_{k',\pi}\nu_{k,\pi} \leq E \Rightarrow \begin{cases} y_{k',\pi} = 0 \wedge \nu_{k,\pi} \leq 1 \\ y_{k',\pi} = 1 \wedge \nu_{k,\pi} \leq E \\ y_{k',\pi} = 0 \wedge \nu_{k,\pi} \leq E \end{cases}$$

Selection of Maximally Informative Genes, Figure 3
Modeling occupied partitions

of class "k", and (iii) if no points of class "k" are present, then it may contain an arbitrary number of points of class "k". Obviously, the point is to maximize the number of type (i) partitions while satisfying the "purity" requirements. The objective thus becomes to minimize the "slack" variable E. The novelty of our approach is that it allows the complete control of the complexity of the model by treating explicitly the number of features, and number of occupied partitions. The detailed formulation is summarized in Fig. 4.

The proposed formulation optimizes simultaneously for a number of design criteria:

- the feature selection;
- the construction of a multivariate, multi-class classifier; and
- the creation of multiple structurally alternative solutions via the introduction of integer cuts [5].

The overall framework remains linear and the solution is done parametrically for a given number of occupied partitions and number of features. This is a simple way for decoupling the objectives, however, more elaborate multi-objective schemes can, and will be, explored. We will discuss a number of computational studies to illustrate the method.

Cases

Gene Selection and Complexity Minimization

Our case study will focus on the model proposed in mainly because these results will demonstrate how to interpret computational results in their biological concept. This analysis demonstrated that there is indeed a string relationship between computational complexity, as modeled through the various optimization formulations, and biological relevance.

"Small Round Blue Cell Tumors" (SRBCT) is a descriptive category encompassing a large number of malignant tumors that tend to occur in childhood. They are united by their similar histo-pathological appearance. However, subtle clues may be present to distinguish between the tumors. For proper characterization, pathologists often employ immunohistochemistry, electron microscopy, and molecular analysis for chromosomal abnormalities. The SRBCTs of childhood include neuroblastoma (NB), rhabdomyosarcoma (RMS), non Hodgkin lymphoma (NHL), and the Ewing family of tumors (EWS). Currently no single biological or chemical test exists that can detect SRBCTs. A comprehensive study was presented in which a large number of genes were monitored. The data were reduced by SVD decomposition and the leading factors were used to train an Artificial Neural Network to build a predictive diagnostic device.

In order to analyzed this data set with the mixed integer formulation, Fig. 4 the raw data are first preprocessed used an extension of the signal-to-ratio approach introduced by [20]. The original method was extended for multiclass-class problems in order to assist in the elimination of irrelevant features. This step reduced the initial number of features to 500. Multiple cuts were generated for a multitude of features/plane combinations and we discuss representative results to illustrate the extracted information. The maximally informative model has 3 features and 2 planes (4 occupied partitions), Fig. 5. As a standard validation, it was verified that scrambling the data (systematic error) does significantly deteriorate the performance of the classifier which is a further validation that the underlying structure in the data in the result of a random process.

Through the aforementioned analysis several conserved key genes across multiple solutions were identified, all of which are integral in tumor progression. The first of these genes, caveolin-1 (CAV-1), has been documented, when its expression patterns are altered, to be a key component in the formation of a variety of tumors, such as prostate [50], bladder [37], esophageal, and mammary [26,51]. The effects of CAV-1 in tumorigenesis fall into three major categories: 1) deregulation of cell cycle control [49]; 2) metalloproteinase production [51]; and 3) induction of angiogenesis [44]. The second gene identified, neurofibromatosis 2 (NF2), has also been shown to be involved in cancer formation, in

$$\min E$$

$$s.t.$$

$$v_{k',\pi} \le E + (1 - y_{k,\pi})$$

$$or \qquad \forall k, k' > k, \pi$$

$$v_{k,\pi} \le E + (1 - y_{k',\pi})$$

$$\sum_{k,\pi} y_{k,\pi} = N_\pi$$

$$q_{n,\pi} \le \frac{\sum_p c_1(p,\pi) z_{n,p} + c_0(\pi)}{N_p}$$

$$q_{n,\pi} \ge \sum_p c_1(p,\pi) z_{n,p} + c_0(\pi) - (N_p - 1)$$

$$\sum_n \sum_\pi q_{n,\pi} = N$$

$$v_{k,\pi} = \frac{\sigma_{k,\pi}}{N_k}$$

$$\sigma_{k,\pi} = \sum_n B(n,k) q_{n,\pi}$$

$$\sum_\pi v_{k,\pi} = 1$$

$$\sum_f D(n,f) w_{f,p} + U z_{n,p} \le U + \vartheta_p - z_{n,p}\varepsilon$$

$$\sum_f D(n,f) w_{f,p} + (u - \varepsilon) z_{n,p} \ge \varepsilon + \vartheta_p$$

$$y_{k,\pi} \le N_k v_{k,\pi}$$

$$y_{k,\pi} \ge v_{k,\pi}$$

$$w_{f,p} \ge \mu s_f$$

$$w_{f,p} \le M s_f$$

$$\sum_f s_f = N_f$$

$$z_{n,p} = \begin{cases} 1 \text{ if } \sum_f D(n,f) w_{f,p} \le \vartheta_p \\ 0 \text{ if } \sum_f D(n,f) w_{f,p} \ge \vartheta_p \end{cases}$$

$$q_{n,\pi} = \begin{cases} 1 \text{ if point } n \text{ belongs to partition } \pi \\ 0 \text{ otherwise} \end{cases}$$

$\sigma_{k,\pi}$ = number of points of class k in partition π

$v_{k,\pi}$ = fraction of points of class k in partition π

$$y_{k,\pi} = \begin{cases} 1 \text{ if partition } \pi \text{ containts points of class } k \\ 0 \text{ otherwise} \end{cases}$$

$$s_f = \begin{cases} 1 \text{ if feature } f \text{ is used} \\ 0 \text{ otherwise} \end{cases}$$

$$y_{k,\pi} v_{k',\pi} \le E \Rightarrow \left\{ \begin{array}{l} y_{k,\pi} = 0 \wedge v_{k',\pi} \le 1 \\ y_{k,\pi} = 1 \wedge v_{k',\pi} \le E \\ y_{k,\pi} = 0 \wedge v_{k',\pi} \le E \end{array} \right\} \Leftrightarrow v_{k',\pi} \le E + (1 - y_{k,\pi})$$

or

$$y_{k',\pi} v_{k,\pi} \le E \Rightarrow \left\{ \begin{array}{l} y_{k',\pi} = 0 \wedge v_{k,\pi} \le 1 \\ y_{k',\pi} = 1 \wedge v_{k,\pi} \le E \\ y_{k',\pi} = 0 \wedge v_{k,\pi} \le E \end{array} \right\} \Leftrightarrow v_{k,\pi} \le E + (1 - y_{k',\pi})$$

$w_{f,p}$ and ϑ_p = plane parameters, Dw $\le$ ϑ (*variables*)

N = number of samples

N_π = desired number of occupied partitions (*parameter*)

N_p = number of planes (*parameter*)

N_f = number of features to be selected (*parameter*)

N_k = number of samples in class k (*known*)

u, U, μ, M = big-M parameters

c_0, c_1 = model parameters (*known*)

D(n, f), B(n,k) = data

p = *planes*

π = *partitions* = 2^p

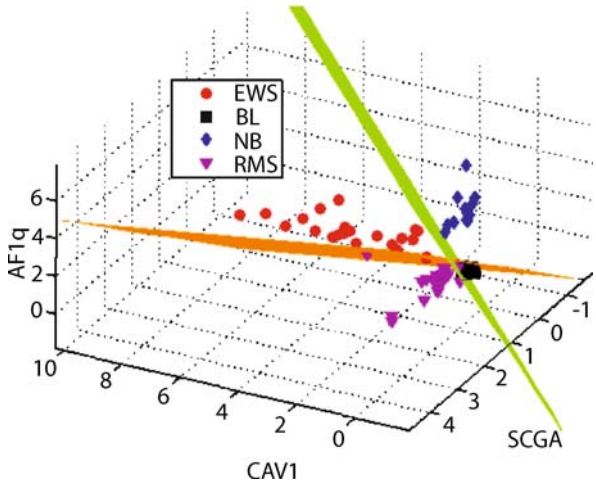
f = *features*

k = *classes*

n = *samples*

Selection of Maximally Informative Genes, Figure 4
Mixed formulation for minimizing informative genes and classifier complexity

neural based tumors such as schwannomas and meningiomas [17,31,40,52]. NF2 exerts its effect through its gene product, merlin, which is involved in the regulation of cell motility and cell proliferation. Recent studies have highlighted the involvement of merlin in tumor suppression through the inhibition of Rac signaling. In cases where the production of NF2 is diminished, RAC signaling becomes activated, and the tumor suppression capabilities of NF2 are lost. The third major gene identified is a myeloid/lymphoid or mixed-lineage leukemia marker (AF1Q). While the literature on this gene is limited, it is known that this gene is necessary for neuronal differentiation [28]. Thus it may be possible that uncontrolled regulation of this gene may lead to neuronal-based tumors. The fourth gene, sarcoglycan, alpha (SGCA), is a component of the dystrophin-



Selection of Maximally Informative Genes, Figure 5
Optimal space decomposition for the SRBCT problem

glycoprotein complex, and has been linked to the onset of mammary tumorigenesis [48]. Tumorgenesis is induced when mutations inhibit the production of SGCA. This leads to a subsequent loss of control over a variety of functions such as growth control, cell survival, cytoskeletal organization, basement membrane assembly, branching morphogenesis, polarity, and tumor suppression in epithelial cells. As mentioned previously in the case of CAV-1, when these key components of cellular function become aberrant, tumorigenesis ensues. The final gene identified through our analysis is the CD99 antigen (CD99), which has been determined to be a marker of lung carcinomas as well as mammary tumors. It is thought that CD99 might play an integral role in the aggregation of breast cancer cells, the initiating step of tumorigenesis. CD99 also assists in the invasive processes characteristic of metastatic tumors. Finally, the gene for receptor, IgG, alpha chain transporter (FCGRT, FCRN) was also selected. This gene, as well as a few others, has been detected through the use of cDNA microarrays, in studies involved in elucidating the underlying genomic profile of astrocytomas [22]. Specifically FCRN is known to mediate immune defense in response to the onset of pilocytic astrocytomas, possible keeping the astroctyoma in a benign state. In addition, FCRN is known to be expressed by dendritic cells [54] and may serve as a basal mechanism of immune function.

Conclusions

This summary presented a number of optimization-based formulation, with emphasis on mathematical programming, for addressing the problem of gene selection. Numerous issues can be raised for future research. In fact the advantage of a MP-based formalism is the tremendous flexibility it provides.

Multi-objective optimization: Interpretation of biological information needs to tackle multiple simultaneous objectives. In this short review we discussed simultaneous optimization of accuracy and size of classifier (number of features). In clustering applications the number of clusters is yet another level of complexity, hence an additional decision variable. Therefore, multicriteria trade-off curves (Pareto solutions) have to be developed for these high-dimensional mixed integer (non) linear optimization problems.

Incorporation of biological constraints: One of the advantages of using mathematical programming techniques is that constrains can be readily accounted for. Thus far microarray analyzes approaches treat the array data as raw unconstrained measurements. One of the targets of microarray analysis is to identify potential correlations among the data. However, prior biological knowledge is not taken into account mainly because most data mining methods cannot handle implicit or explicit constraints. Recently, the need to account for biological driven constraints when clustering expression profiles was demonstrated [42].

Large-scale combinatorial optimization: The development of scalable algorithms is a daunting task in optimization theory. With the recent developments in genomics we should be expecting routinely that the analysis of gene arrays composed of tens of thousands of probes (hence tens of thousands of binary variables in the MIP gene selection formulation). Recent works [14,18,39], discuss various mixed-integer reformulations to the classification problem. The recent work of Shioda et al. [43], identified opportunities for successful reformulations of various data mining tasks in the context of linear integer optimization. Busygin et al., present some more recent ideas for addressing the bi-clustering problem as a fractional 0-1 optimization problems [9]. Undoubtedly, integer optimization will play a prominent role in feature algorithmic developments as recent results demonstrate the complemen-

tarity of the different methodologies, suggesting that a unified approach may help to uncover complex genetic risk factors not currently discovered with a single method [34].

Global optimization: The development of general non-linear, non-convex separating boundaries naturally leads to requirements of solving large-scale combinatorial non-linear problems to global optimality. Recent advances in the theory and practice of deterministic global optimization are also expected to be critical enablers [16].

Analyzing almost empty spaces: The sparseness of the data set is a critical roadblock. Accurate models can be developed using convoluted optimization approaches. However, we would constantly lack appropriately populated datasets in order to achieve a reasonable balance between the thousands of independent variables (genes measured) and necessary measurements (tissue samples) for a robust identification. Information theoretic approaches accounting for complexity (Akaike and Bayesian Information Criteria) should be developed to strike a balance between the complexity and the accuracy of the model so as to avoid pointless over fitting of the sparsely populated datasets.

Uncertainty considerations: Noise and uncertainty in the data is a given. Therefore, data mining algorithms in general and mathematical programming formulations in particular have to account for the presence of noise. Issues from robustness and uncertainty propagation have to be incorporated. However, an interesting issue emerges: how do we distinguish between noise and an infrequent, albeit interesting observation? This in fact maybe a question with no answer especially if we consider the implications of sparsely populated data sets.

References

1. Alizadeh AA, Eisen MB, Davis RE, Ma C, Lossos IS, Rosenwald A, Boldrick JC, Sabet H, Tran T, Yu X et al (2000) Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature 403:503–511
2. Allander SV, Nupponen NN, Ringner M, Hostetter G, Maher GW, Goldberger N, Chen Y, Carpten J, Elkahloun AG, Meltzer PS (2001) Gastrointestinal stromal tumors with KIT mutations exhibit a remarkably homogeneous gene expression profile. Cancer Res 61:8624–8628
3. Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJ (1999) Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc Natl Acad Sci USA 96:6745–6750
4. Bassett DE Jr, Eisen MB, Boguski MS (1999) Gene expression informatics–it's all in your mine. Nat Genet 21:51–55
5. Biegler LT, Grossmann IE, Westerberg AW (1997) Systematic Methods of Chemical Process Design. Prentice Hall
6. Bittner M, Meltzer P, Chen Y, Jiang Y, Seftor E, Hendrix M, Radmacher M, Simon R, Yakhini Z, Ben-Dor A et al (2000) Molecular classification of cutaneous malignant melanoma by gene expression profiling. Nature 406:536–540
7. Bowtell DD (1999) Options available–from start to finish–for obtaining expression data by microarray. Nat Genet 21:25–32
8. Brown PO, Botstein D (1999) Exploring the new world of the genome with DNA microarrays. Nat Genet 21:33–37
9. Busygin S, Prokopyev OA, Pardalos PM (2005) Feature selectionfor consistent biclustering via fractinal 0-1 programming. J Comb Optim 10:7–21
10. Cheung VG, Morley M, Aguilar F, Massimi A, Kucherlapati R, Childs G (1999) Making and reading microarrays. Nat Genet 21:15–19
11. Chilingaryan A, Gevorgyan N, Vardanyan A, Jones D, Szabo A (2002) Multivariate approach for selecting sets of differentially expressed genes. Math Biosci 176:59–69
12. Davis CA, Gerick F, Hintermair V, Friedel CC, Fundel K, Kuffner R, Zimmer R (2006) Reliable gene signatures for microarray classification: assessment of stability and performance. Bioinformatics 22:2356–2363
13. Dettling M, Buhlmann P: Finding predictive gene groups from microarray data. J Multivariate Anal 90:106–131
14. Duarte Silva AP, Stam A (1997) A mixed integer programming algorithm for minimizing the training sample misclassification cost in two-group classification. Ann Oper Res 74:129–157
15. Dudoit S, Yang YH, Callow MJ, Speed TP (2002) Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. Statistica Sinica 12:111–139
16. Floudas CA (2000) Nonlinear and Mixed-Integer Optimization: Fundamentals and Applciations. Oxford University Press
17. Fraenzer JT, Pan H, Minimo L Jr, Smith GM, Knauer D, Hung G (2003) Overexpression of the NF2 gene inhibits schwannoma cell proliferation through promoting PDGFR degradation. Int J Oncol 23:1493–1500
18. Gallagher RJ, Lee EK, Patterson DA (1997) Constrained discriminant analysis via 0/1 mixed integer programming. Ann Oper Res 74:65–88
19. Glen JJ (2001) Classification accuracy in discriminant analysis: a mixed integer programming approach. J Oper Res Soc 52:328–339
20. Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA et

al (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286:531–537
21. Hsu CW, Lin CJ (2002) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Netw 13:415–425
22. Huang H, Hara A, Homma T, Yonekawa Y, Ohgaki H (2005) Altered expression of immune defense genes in pilocytic astrocytomas. J Neuropathol Exp Neurol 64:891–901
23. Huang HL, Lee CC, Ho SY (2007) Selecting a minimal number of relevant genes from microarray data to design accurate tissue classifiers. Biosystems 90:78–86
24. Kafatos FC (2002) A revolutionary landscape: the restructuring of biology and its convergence with medicine. J Mol Biol 319:861–867
25. Khan J, Wei JS, Ringner M, Saal LH, Ladanyi M, Westermann F, Berthold F, Schwab M, Antonescu CR, Peterson C, Meltzer PS (2001) Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nat Med 7:673–679
26. Lee H, Park DS, Razani B, Russell RG, Pestell RG, Lisanti MP (2002) Caveolin-1 mutations (P132L and null) and the pathogenesis of breast cancer: caveolin-1 (P132L) behaves in a dominant-negative manner and caveolin-1 (-/-) null mice show mammary epithelial cell hyperplasia. Am J Pathol 161:1357–1369
27. Li W, Yang Y (eds) (2002) How many genes are neede for a discriminant microarray data analysis. Kluwer, Dordrecht
28. Lin HJ, Shaffer KM, Sun Z, Jay G, He WW, Ma W (2004) AF1q, a differentially expressed gene during neuronal differentiation, transforms HEK cells into neuron-like cells. Brain Res Mol Brain Res 131:126–130
29. Lipshutz RJ, Fodor SP, Gingeras TR, Lockhart DJ (1999) High density synthetic oligonucleotide arrays. Nat Genet 21:20–24
30. Liu H, Motoda H (2000) Feature selection for knowledge discovery and data mining. Kluwer, Dordrecht
31. Lomas J, Bello MJ, Arjona D, Alonso ME, Martinez-Glez V, Lopez-Marin I, Aminoso C, de Campos JM, Isla A, Vaquero J, Rey JA (2005) Genetic and epigenetic alteration of the NF2 gene in sporadic meningiomas. Genes Chromosomes Cancer 42:314–319
32. Luo J, Duggan DJ, Chen Y, Sauvageot J, Ewing CM, Bittner ML, Trent JM, Isaacs WB (2001) Human prostate cancer and benign prostatic hyperplasia: molecular dissection by gene expression profiling. Cancer Res 61:4683–4688
33. Maglietta R, D'Addabbo A, Piepoli A, Perri F, Liuni S, Pesole G, Ancona N (2007) Selection of relevant genes in cancer diagnosis based on their prediction accuracy. Artif Intell Med 40:29–44
34. Moscato P, Berretta R, Hourani M, Mendes A, Cotta C (2005) Genes related with Alzheimer's disease: A comparison of evolutionary search, statistical and integer programming approaches. Appl Evol Comput Proc 3449:84–94
35. Perou CM, Jeffrey SS, van de Rijn M, Rees CA, Eisen MB, Ross DT, Pergamenschikov A, Williams CF, Zhu SX, Lee JC et al (1999) Distinctive gene expression patterns in human mammary epithelial cells and breast cancers. Proc Natl Acad Sci USA 96:9212–9217
36. Pollack JR, Perou CM, Alizadeh AA, Eisen MB, Pergamenschikov A, Williams CF, Jeffrey SS, Botstein D, Brown PO (1999) Genome-wide analysis of DNA copy-number changes using cDNA microarrays. Nat Genet 23:41–46
37. Rajjayabun PH, Garg S, Durkan GC, Charlton R, Robinson MC, Mellon JK (2001) Caveolin-1 Expression is Associated with high-grade Bladder Cancer. Urology 58:811–814
38. Ross DT, Scherf U, Eisen MB, Perou CM, Rees C, Spellman P, Iyer V, Jeffrey SS, Van de Rijn M, Waltham M et al (2000) Systematic variation in gene expression patterns in human cancer cell lines. Nat Genet 24:227–235
39. Rubin PA (1997) Solving mixed integer classification problems by decomposition. Ann Oper Res 74:51–64
40. Ryu CH, Kim SW, Lee KH, Lee JY, Kim H, Lee WK, Choi BH, Lim Y, Kim YH, Lee KH et al (2005) The merlin tumor suppressor interacts with Ral guanine nucleotide dissociation stimulator and inhibits its activity. Oncogene 24:5355–5364
41. Schena M, Shalon D, Davis RW, Brown PO (1995) Quantitative monitoring of gene expression patterns with a complementary DNA microarray. Science 270:467–470
42. Sese J, Kurokawa Y, Monden M, Kato K, Morishita S (2004) Constrained clusters of gene expression profiles with pathological features. Bioinformatics 20:3137–3145
43. Shioda R (2003) Integer optimization in data mining. MIT, Operations Research
44. Sonveaux P, Martinive P, DeWever J, Batova Z, Daneau G, Pelat M, Ghisdal P, Gregoire V, Dessy C, Balligand JL, Feron O (2004) Caveolin-1 expression is critical for vascular endothelial growth factor-induced ischemic hindlimb collateralization and nitric oxide-mediated angiogenesis. Circ Res 95:154–161
45. Street WN (2005) Oblique multicategory decision trees using nonlinear programming. Inf J Comput 17:25–31
46. Sun M, Xiong M (2003) A mathematical programming approach for gene selection and tissue classification. Bioinformatics 19:1243–1251
47. Szabo A, Boucher K, Carroll WL, Klebanov LB, Tsodikov AD, Yakovlev AY (2002) Variable selection and pattern recognition with gene expression data generated by the microarray technology. Math Biosci 176:71–98
48. Weir ML, Muschler J (2003) Dystroglycan: emerging roles in mammary gland function. J Mammary Gland Biol Neoplasia 8:409–419
49. Williams TM, Cheung MW, Park DS, Razani B, Cohen AW, Muller WJ, Di Vizio D, Chopra NG, Pestell RG, Lisanti MP (2003) Loss of caveolin-1 gene expression accelerates the development of dysplastic mammary lesions in tumor-prone transgenic mice. Mol Biol Cell 14:1027–1042

50. Williams TM, Hassan GS, Li J, Cohen AW, Medina F, Frank PG, Pestell RG, Di Vizio D, Loda M, Lisanti MP (2005) Caveolin-1 promotes tumor progression in an autochthonous mouse model of prostate cancer: genetic ablation of Cav-1 delays advanced prostate tumor development in tramp mice. J Biol Chem 280:25134–25145
51. Williams TM, Medina F, Badano I, Hazan RB, Hutchinson J, Muller WJ, Chopra NG, Scherer PE, Pestell RG, Lisanti MP (2004) Caveolin-1 gene disruption promotes mammary tumorigenesis and dramatically enhances lung metastasis in vivo. Role of Cav-1 in cell invasiveness and matrix metalloproteinase (MMP-2/9) secretion. J Biol Chem 279:51630–51646
52. Xiao GH, Chernoff J, Testa JR (2003) NF2: the wizardry of merlin. Genes Chromosomes Cancer 38:389–399
53. Yang E, Maguire T, Yarmush ML, Androulakis IP (2007) Informative Gene Selection and Design of Regulatory Networks using Integer Optimization. Comput Chem Eng, accepted for publication
54. Zhu X, Meng G, Dickinson BL, Li X, Mizoguchi E, Miao L, Wang Y, Robert C, Wu B, Smith PD et al (2001) MHC class I-related neonatal Fc receptor for IgG is functionally expressed in monocytes, intestinal macrophages, and dendritic cells. J Immunol 166:3266–3276

Selfdual Parametric Method for Linear Programs

Criss-cross Method

VIVEK DUA, EFSTRATIOS N. PISTIKOPOULOS
Imperial College, London, UK

MSC2000: 90C05, 90C06

Article Outline

Keywords
See also
References

Keywords

Linear programming; Primal and dual simplex algorithms

In this article we will describe the selfdual parametric method for solving linear programs (LPs) of the following form:

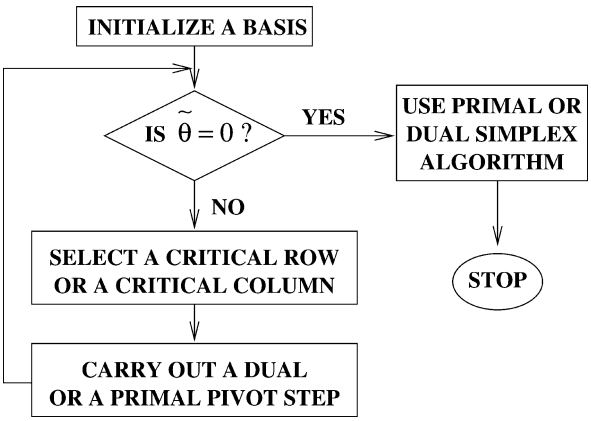
$$\begin{cases} \min\limits_{x} & c^\top x \\ \text{s.t.} & Ax = b\,, \quad x \geq 0\,, \quad x \in \mathbb{R}^n, \end{cases} \tag{1}$$

where x is a vector of continuous variables; A is a constant matrix; c and b are constant vectors and we assume that (1) does not contain redundant constraints (see [8] or [7] for a procedure for removing redundant constraints). The solution of (1) can be approached by primal or dual simplex methods, or their variants [4,8]; however, see [9] for interior point methods. Primal and dual simplex algorithms start from a primal or dual feasible solution, which is then solved to optimality by using primal or dual simplex pivot steps respectively. On the other hand, the selfdual parametric method, which is also called the criss-cross method, does not require a starting feasible solution and uses a combination of primal and dual simplex pivot steps. The criss-cross method is based upon introducing a parameter, θ, into the model, (1), and then minimizing θ by using primal or dual simplex pivot steps. If θ decreases to zero the problem is primal or dual feasible and then the optimal solution is found by using primal or dual simplex method respectively, otherwise, the problem is infeasible.

In order to apply the selfdual parametric method, the parameter θ is introduced and the problem in (1) is rewritten in the following form [4,8]:

$$\begin{cases} \min\limits_{x} & \sum\limits_{j=1}^{n} (c_j' + \theta \widehat{c}_j) x_j \\ \text{s.t.} & x_i + \sum\limits_{j=m+1}^{n} a_{ij}' x_j = b_i' + \theta \widehat{b}_i, \\ & i = 1, \ldots, m, \\ & x_j \geq 0 \quad \text{for all } j \\ & x \in \mathbb{R}^n, \end{cases} \tag{2}$$

where $(x_1, \ldots, x_m)$ is the vector of the selected basic variables; $(0, \ldots, 0, c_{m+1}', \ldots, c_n')$ and $(b_1', \ldots, b_m')$ are the updated c and b vectors after pricing out the basic columns; $\widehat{b}_i = 0$ if $b' \geq 0$, $\widehat{b}_i = 1$ otherwise; and, $\widehat{c}_i = 0$, if $c' \geq 0$, $\widehat{c}_i = 1$ otherwise. The basic idea of the selfdual parametric method is to minimize θ by using primal or dual simplex pivot steps – the smallest value of θ corresponding to an optimal tableau is called the *critical value*, $\widetilde{\theta}$. When $b_i', c_j' \geq 0$ then $\widetilde{\theta} = 0$, and the problem can be solved to an optimal solution by using primal or dual simplex method. Otherwise, $\widetilde{\theta} > 0$ and



Selfdual Parametric Method for Linear Programs, Figure 1
Criss-cross method

is given by

$$\max\left\{\left\{-\frac{b_i'}{\widetilde{b}_i}:\ \text{with } i \text{ s.t. } b_i' < 0\right\},\ \left\{-\frac{c_j'}{\widetilde{c}_j}:\ \text{with } j \text{ s.t. } c_j' < 0\right\}\right\}.$$

In such a case, i. e., when $\widetilde{\theta} > 0$, if $\widetilde{\theta} = -(b_i'/\widehat{b}_i)$, the ith row of the current tableau is defined as a *critical row* and if $\widetilde{\theta} = -(c_j'/\widehat{c}_j)$, then the jth row of the current tableau is defined as a *critical column*. The criss-cross method identifies a critical column or a critical row and a primal or dual simplex step, respectively, is carried out. Either $\widetilde{\theta}$ decreases to zero in a finite number of steps or the problem is primal or dual infeasible. The basic idea of the algorithm is outlined in Fig. 1.

Other useful references on the subject are [1,2,3,5, 10,11] and [6].

See also

- ► Bounds and Solution Vector Estimates for Parametric NLPs
- ► Multiparametric Linear Programming
- ► Multiparametric Mixed Integer Linear Programming
- ► Nondifferentiable Optimization: Parametric Programming
- ► Parametric Global Optimization: Sensitivity
- ► Parametric Linear Programming: Cost Simplex Algorithm
- ► Parametric Mixed Integer Nonlinear Optimization
- ► Parametric Optimization: Embeddings, Path Following and Singularities

References

1. Bland RG (1977) New finite pivoting rules for the simplex. Math Oper Res 2:103–107
2. Chen H-D, Pardalos PM, Saunders MA (1994) The simplex algorithm with a new primal and dual pivot rule. Oper Res Lett 16:121–127
3. Clausen J (1987) A note on Edmunds–Fukuda pivoting rule for the simplex method. Europ J Oper Res 29:378–383
4. Dantzig GB (1963) Linear programming and extensions. Princeton Univ Press, Princeton
5. Fukuda K, Matsui T (1991) On the finiteness of the criss-cross method. Europ J Oper Res 52:119–124
6. Fukuda K, Terlaky T (1997) Criss-cross methods: A fresh view on pivot algorithms. Math Program 79:369–395
7. Gal T (1995) Postoptimal analyses, parametric programming, and related topics. de Gruyter, Berlin
8. Murty KG (1983) Linear programming. Wiley, New York
9. Roos C, Terlaky T, VialJ-Ph (1997) Theory and algorithms for linear optimization, an interior point approach. Wiley, New York
10. Zionts S (1969) The criss-cross method for solving linear programming problems. Managem Sci 15:426–445
11. Zionts S (1972) Some empirical tests of the criss-cross method. Managem Sci 19:406–410

Semidefinite Programming and Determinant Maximization

LIEVEN VANDENBERGHE[1], STEPHEN BOYD[2], SHAO-PO WU[2]
[1] University California, Los Angeles, USA
[2] Stanford University, Stanford, USA

MSC2000: 90C25, 90C55, 90C25, 90C90, 15A15

Article Outline

Keywords
Ellipsoidal Approximation
Experiment Design
Estimation of Structured Covariance Matrices
Quasi-Newton Updates
Conclusion
See also
References

Keywords

Semidefinite programming; Convex optimization; Linear matrix inequality; Interior point methods; Ellipsoidal approximation; Experiment design

A wide variety of nonlinear *convex optimization* problems can be expressed in terms of *linear matrix inequalities* (LMIs), i. e., convex constraints of the form

$$x_1A_1 + \cdots + x_nA_n \preceq B$$

where $x \in \mathbf{R}^n$ is the optimization variable, $A_i = A_i^\top \in \mathbf{R}^{m\times m}$, $B = B^\intercal \in \mathbf{R}^{m\times m}$ are given matrices, and the inequality sign denotes matrix inequality (i. e., the matrix $B - \sum_i x_iA_i$ is positive semidefinite). Linear matrix inequalities arise in systems and control, combinatorial optimization, statistics, eigenvalue optimization, and several other fields.

The most widely studied problem of this type is the *semidefinite programming problem* (SDP), in which we minimize a linear function subject to an LMI constraint:

$$\begin{cases} \min & c^\top x \\ \text{s.t.} & x_1A_1 + \cdots + x_nA_n \preceq B. \end{cases}$$

Semidefinite programming has received a great deal of attention recently, motivated by the discovery of new applications and by the development of new *interior point methods*. For surveys of the theory and applications of SDP, see [1,5,14,15,16,21]. SDPs are convex optimization problems, since the linear matrix inequality is a convex constraint, and the objective function is linear.

An interesting closely related problem is the problem of maximizing the determinant of a linear matrix function, subject to LMI constraints:

$$\begin{cases} \max & \det(C_0 + x_1C_1 + \cdots + x_nC_n) \\ \text{s.t.} & C_0 + x_1C_1 + \cdots + x_nC_n \succ 0 \\ & x_1A_1 + \cdots + x_nA_n \preceq B \end{cases} \tag{1}$$

where $C_i = C_i^\top \in \mathbf{R}^{p\times p}$, $A_i = A_i^\top \in \mathbf{R}^{m\times m}$, and $B = B^\intercal \in \mathbf{R}^{m\times m}$ are given. Although the objective function is not concave, this problem can be easily transformed in a convex optimization problem. We can first note that the function $f(X) = -(\det X)^{1/p}$ is convex on the set of positive definite matrices in $\mathbf{R}^{p\times p}$, so problem (1) is equivalent to the convex minimization problem

$$\begin{cases} \min & -(\det C(x))^{\frac{1}{p}} \\ \text{s.t.} & C(x) \succ 0 \\ & x_1A_1 + \cdots + x_nA_n \preceq B, \end{cases}$$

where $C(x) = C_0 + x_1C_1 + \cdots + x_nC_n$. An alternative formulation is

$$\begin{cases} \min & \log\det C(x)^{-1} \\ \text{s.t.} & C(x) \succ 0 \\ & x_1A_1 + \cdots + x_nA_n \preceq B. \end{cases}$$

This is a convex problem, since the function $\log\det X^{-1}$ is convex on the set of positive definite matrices.

A unified form that includes both the SDP and the determinant maximization problem is

$$\begin{cases} \min & c^\top x + \log\det C(x)^{-1} \\ \text{s.t.} & C(x) \succ 0 \\ & x_1A_1 + \cdots + x_nA_n \preceq B. \end{cases} \tag{2}$$

It is clear that this problem reduces to an SDP when $C_0 = 1$ and $C_i = 0$ for $i > 0$, and to a determinant maximization problem when $c = 0$. Moreover, as we will see below, there exist important applications where both terms arise.

Reference [22] gives a detailed discussion of problem (2), including an overview of applications and a description of an efficient interior point method. In this article, we illustrate the applications of (2) with a few examples from different areas. Following [22], we will refer to (2) as a *max-det problem*.

Ellipsoidal Approximation

Our first class of examples are *ellipsoidal approximation* problems. We can distinguish two basic problems. The first is the problem of finding the *minimum-volume ellipsoid* around a given set C. The second problem is the problem of finding the *maximum-volume ellipsoid* contained in a given convex set C. Both can be formulated as convex (semi-infinite) programming problems.

To solve the first problem, it is convenient to parametrize the ellipsoid as the pre-image of a unit ball under an affine transformation, i. e., as

$$\mathcal{E} = \{v:\ \|Av + b\| \le 1\}.$$

It can be assumed without loss of generality that $A = A^\mathsf{T} \succ 0$, in which case the volume of $\mathcal{E}$ is proportional to $\det A^{-1}$. The problem of computing the minimum-volume ellipsoid containing C can be written as

$$\begin{cases} \min & \log\det A^{-1} \\ \text{s.t.} & A = A^\mathsf{T} \succ 0 \\ & \|Av + b\| \le 1, \\ & \text{for all } v \in C, \end{cases} \tag{3}$$

where the variables are A and b. Note that both the objective function and the constraints are convex in A and b.

For the second problem, where we maximize the volume of ellipsoids enclosed in a convex set C, it is more convenient to represent the ellipsoid as the *image* of the unit ball under an affine transformation, i. e., as

$$\mathcal{E} = \{By + d : \ \|y\| \le 1\}.$$

Again it can be assumed that $B = B^\mathsf{T} \succ 0$. The volume is proportional to $\det B$, so we can find the maximum volume ellipsoid inside C by solving the convex optimization problem

$$\begin{cases} \max & \log\det B \\ \text{s.t.} & B = B^\mathsf{T} \succ 0 \\ & By + d \in C, \\ & \forall y, \ \|y\| \le 1 \end{cases} \tag{4}$$

in the variables B and d.

For general C, problems (4) and (3) are semi-infinite programming problems. They reduce to finite problems in certain cases, which we now review.

We first consider the problem of finding the minimum volume ellipsoid that contains given points $x^1, \ldots, x^K$ in $\mathbf{R}^n$, i. e.,

$$C = \{x^1, \ldots, x^K\},$$

(or, equivalently, the convex hull of $\{x^1, \ldots, x^K\}$). This problem has applications in cluster analysis [4,18]), and robust statistics [19, §7]. Applying (3), we can write this problem as

$$\begin{cases} \min & \log\det A^{-1} \\ \text{s.t.} & \left\|Ax^i + b\right\| \le 1, \ i = 1, \ldots, K, \\ & A = A^\mathsf{T} \succ 0, \end{cases} \tag{5}$$

where the variables are $A = A^\mathsf{T} \in \mathbf{R}^{n\times n}$ and $b \in \mathbf{R}^n$. The norm constraints $\|Ax^i + b\| \le 1$, which are convex quadratic inequalities in the variables A and b, can be expressed as LMIs

$$\begin{pmatrix} I & Ax^i + b \\ (Ax^i + b)^\mathsf{T} & 1 \end{pmatrix} \succeq 0,$$

so (5) is a max-det problem in the variables A and b.

In a similar way, we can compute the maximum volume ellipsoid contained in a polytope described by a set of linear inequalities

$$C = \left\{x : \ a_i^\mathsf{T} x \le b_i, \ i = 1, \ldots, L\right\}.$$

To apply (4) we first work out the last constraint:

$$\begin{aligned} & By + d \in C, \quad \forall \ \|y\| \le 1 \\ & \Rightarrow a_i^\mathsf{T}(By + d) \le b_i, \quad \forall \ \|y\| \le 1 \\ & \Rightarrow \max_{\|y\|\le 1} a_i^\mathsf{T} By + a_i^\mathsf{T} d \le b_i \\ & \Rightarrow \|Ba_i\| + a_i^\mathsf{T} d \le b_i. \end{aligned}$$

This is a convex constraint in B and d, and equivalent to the LMI

$$\begin{pmatrix} (b_i - a_i^\mathsf{T} d)I & Ba_i \\ a_i^\mathsf{T} B & b_i - a_i^\mathsf{T} d \end{pmatrix} \succeq 0.$$

We can therefore formulate (4) as a max-det problem in the variables B and d:

$$\begin{cases} \min & \log\det B^{-1} \\ \text{s.t.} & B \succ 0 \\ & \begin{pmatrix} (b_i - a_i^\mathsf{T} d)I & Ba_i \\ (Ba_i)^\mathsf{T} & b_i - a_i^\mathsf{T} d \end{pmatrix} \succeq 0, \\ & i = 1, \ldots, L. \end{cases}$$

These techniques extend to several interesting cases where C is not finite or polyhedral, but is defined as a combination (the sum, union, or intersection) of ellipsoids. In particular, it is possible to compute the optimal inner approximation of the intersection or the sum of ellipsoids, and the optimal outer approximation of the union or sum of ellipsoids, by solving a max-det problem. See [5] and [8] for details.

As an example, consider the problem of finding the minimum volume ellipsoid $\mathcal{E}_0$ containing K given ellipsoids $\mathcal{E}_1, \ldots, \mathcal{E}_K$ described as

$$\mathcal{E}_i = \left\{x : \ x^\mathsf{T} A_i x + 2b_i^\mathsf{T} x + c_i \le 0\right\}.$$

The solution can be found by solving the following max-det problem in the variables $A_0 = A_0^\top$, b_0, and K scalar variables τ_i:

$$\begin{cases} \min & \log\det A_0^{-1} \\ \text{s.t.} & A_0 = A_0^\top \succ 0 \\ & \tau_1 \geq 0, \ldots, \tau_K \geq 0 \\ & \begin{pmatrix} A_0 - \tau_i A_i & b_0 - \tau_i b_i & 0 \\ (b_0 - \tau_i b_i)^\top & -1 - \tau_i c_i & b_0^\top \\ 0 & b_0 & -A_0 \end{pmatrix} \preceq 0, \\ & i = 1, \ldots, K. \end{cases}$$

(c_0 is given by $c_0 = b_0^\top A_0^{-1} b_0 - 1$.) See [5, p. 43], for details.

Experiment Design

The field of *experiment design* is another source of max-det problems. Suppose $x \in \mathbf{R}^n$ is an unknown quantity that we want to estimate from a measurement $y = Ax + w$ where $w \sim \mathcal{N}(0, \mathrm{I})$ is measurement noise, and $A \in \mathbf{R}^{N\times n}$ ($N \geq n$) has full rank. The minimum variance estimator is given by $\widehat{x} = (A^\top A)^{-1} A^\top y$, and has an error covariance $\mathsf{E}(x - \widehat{x})(x - \widehat{x})^\top = (A^\top A)^{-1}$. Suppose that the rows of the matrix A can be chosen among M possible test vectors $v^i \in \mathbf{R}^n$. The goal of experiment design is to choose the rows of A so that the error covariance is 'small'. We can write $A^\mathsf{T} A = N \sum_{i=1}^M \lambda_i v_i v_i^\top$, where λ_i is the fraction of measurements that use the test vector v^i. If $N \gg M$ we can ignore the fact that the numbers λ_i are integer multiples of $1/N$, and treat them as continuous variables.

Several different criteria can be used to measure the size of the error covariance matrix (e. g., the maximum eigenvalue, the trace, or the determinant). A design in which the determinant $\det(AA^\mathsf{T})^{-1}$ is minimized is called *D-optimal*. This problem can be expressed as a max-det problem

$$\begin{cases} \min & \log\det \left(\sum_{i=1}^M \lambda_i v^i (v^i)^\top \right)^{-1} \\ \text{s.t.} & \lambda_i \geq 0, \quad i = 1, \ldots, M, \\ & \sum_{i=1}^M \lambda_i = 1. \end{cases}$$

This formulation of D-optimal experiment design has the advantage that one can easily incorporate useful convex constraints, e. g., linear inequalities on the numbers λ_i. A few interesting possibilities are mentioned in [22]. For surveys of experiment design, see [9,12,17].

Estimation of Structured Covariance Matrices

Suppose we want to estimate the covariance matrix $\Sigma \in \mathbf{R}^{p\times p}$ of a normal distribution $\mathcal{N}(0, \Sigma)$ from M samples $y^{(1)}, \ldots, y^{(M)}$. The *maximum likelihood estimate* for Σ is the positive definite matrix that maximizes $\prod_{i=1}^M p(y^{(i)})$, where

$$p(x) = \left((2\pi)^p \det \Sigma\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} x^\top \Sigma^{-1} x\right).$$

In other words, Σ can be found by solving

$$\begin{cases} \min & \log\det \Sigma^{-1} - \dfrac{1}{M} \displaystyle\sum_{i=1}^M y^{(i)\top} \Sigma^{-1} y^{(i)} \\ \text{s.t.} & \Sigma = \Sigma^\top \succ 0, \end{cases} \tag{6}$$

which can be expressed as a max-det problem in the *inverse* $R = \Sigma^{-1}$:

$$\begin{cases} \min & \mathrm{Tr}\, SR + \log\det R^{-1} \\ \text{s.t.} & R \succ 0, \end{cases} \tag{7}$$

where

$$S = \frac{1}{M} \sum_{i=1}^M y^{(i)} y^{(i)\top}.$$

Without any additional constraints, this problem has a straightforward analytical solution: $R = S^{-1}$ (provided S is nonsingular). The formulation as a max-det problem is useful when additional constraints are added. To give a simple illustration, bounds on the variances Σ_{ii} can be expressed as LMIs in R

$$\Sigma_{ii} = e_i^\top R^{-1} e_i \leq \alpha \Rightarrow \begin{pmatrix} R & e_i \\ e_i^\top & \alpha \end{pmatrix} \succeq 0.$$

Adding constraints is also useful when the matrix S is singular (for example, because the number of samples is too small) and, as a consequence, the max-det problem (7) is unbounded below. In this case we can impose constraints (i. e., prior information) on Σ, for example lower and upper bounds on the diagonal elements of R. See also [2,3,6], [20, §6.13], and [10].

Quasi-Newton Updates

In a quasi-Newton method for unconstrained minimization of a convex function f on $\mathbf{R}^n$, the Newton step $-\nabla^2 f(x)^{-1}\nabla f(x)$ is replaced by $-H^{-1}\nabla f(x)$, where $H = H^\mathsf{T} \succ 0$ is an approximation of the Hessian matrix, based on prior information and previous gradient evaluations. At each iteration, as the algorithm moves from x to the next point x^+, a new approximation H^+ is determined, based on the current H, and on the difference between the gradients at x^+ and x. A good updating rule for H should satisfy several properties: H^+ should be close to H, it should be easy to compute (or, more precisely, the search direction $-H^{+-1}\nabla f(x^+)$ should be easy to compute), and it should incorporate the new information obtained by evaluating the gradient $\nabla f(x^+)$. This last property is usually enforced by imposing the secant condition

$$H^+(x^+ - x) = \nabla f(x^+) - \nabla f(x). \tag{8}$$

R.H. Byrd and J. Nocedal [7] have proposed to measure the difference between H and H^+ by using the *Kullback–Leibler divergence* (or *relative entropy*), given by

$$\frac{1}{2}\left(\operatorname{Tr} H^{-1}H^+ - \log\det H^{-1}H^+ - n\right)$$

(see also [11]). The Kullback–Leibler divergence is nonnegative for all positive definite H and H^+, and zero only if $H^+ = H$. The update H^+ that satisfies the secant condition and minimizes the Kullback–Leibler divergence is the solution of the following optimization problem:

$$\begin{cases} \min & \operatorname{Tr} H^{-1}H^+ - \log\det H^{-1}H^+ \\ \text{s.t.} & H^+ \succ 0 \\ & H^+(x^+ - x) = \nabla f(x^+) - \nabla f(x). \end{cases} \tag{9}$$

R. Fletcher [13] has shown that the solution is given by

$$H^+ = H - \frac{Hss^\top H}{s^\top Hs} + \frac{gg^\top}{s^\top g}, \tag{10}$$

assuming that $s^\mathsf{T} g > 0$, where $s = x^+ - x$ and $g = \nabla f(x^+) - \nabla f(x)$. Formula (10) is well known in unconstrained optimization as the *BFGS quasi-Newton update* (*Broyden–Fletcher–Goldfarb–Shanno*).

Fletcher's observation opens the possibility of adding more complicated LMI constraints to (9), and solving the resulting problem numerically. For example, we can impose a certain sparsity pattern on H^+, or we can relax the secant condition as

$$\left\| H^+(x^+ - x) - \nabla f(x^+) + \nabla f(x) \right\| \le \epsilon,$$

where ϵ is a given tolerance.

Updating H by numerically solving a convex optimization problem will obviously involve far more computation than the BFGS update. Thus, this formulation for *quasi-Newton updates* is only interesting when gradient evaluations are very expensive.

Conclusion

The max-det problem (2) is an extension of the semidefinite programming problem, and includes a wide variety of convex optimization problems as special cases. Some of the applications we discussed have been studied extensively in the literature, and in some cases analytic solutions or efficient specialized algorithms have been developed. The practical importance of the general problem is that it can be handled efficiently using recently developed interior point methods. This allows us to solve interesting variations of problems for which no analytical solution or specialized method exist, and it opens the possibility of adding useful LMI constraints.

See also

- ▶ αBB Algorithm
- ▶ Duality for Semidefinite Programming
- ▶ Eigenvalue Enclosures for Ordinary Differential Equations
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Interior Point Methods for Semidefinite Programming
- ▶ Interval Analysis: Eigenvalue Bounds of Interval Matrices
- ▶ Matrix Completion Problems
- ▶ Semidefinite Programming: Optimality Conditions and Stability
- ▶ Semidefinite Programming and Structural Optimization
- ▶ Semi-infinite Programming, Semidefinite Programming and Perfect Duality
- ▶ Solving Large Scale and Sparse Semidefinite Programs

References

1. Alizadeh F (Feb. 1995) Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM J Optim 5(1):13–51
2. Anderson TW (1969) Statistical inference for covariance matrices with linear structure. In: Krishnaiah PR (ed) Multivariate Analysis II. Proc. Second Internat. Symp. Multivariate Analysis, Wright State Univ., Dayton, Ohio, June 17-21, 1968. Acad. Press, New York, pp 55–66
3. Anderson TW (1970) Estimation of covariance matrices which are linear combinations or whose inverses are linear combinations of given matrices. In: Bose RC, ET AL (eds) Essays in Probability and Statistics. Univ. North Carolina Press, Chapel Hill, NC, pp 1–24
4. Barnes ER (1982) An algorithm for separating patterns by ellipsoids. IBM J Res Developm, 26:759–764
5. Boyd S, El Ghaoui L, Feron E, Balakrishnan V (June 1994) Linear matrix inequalities in system and control theory. Stud Appl Math, vol 15. SIAM, Philadelphia
6. Burg JP, Luenberger DG, Wenger DL (1982) Estimation of structured covariance matrices. Proc IEEE 30:963–974
7. Byrd RH, Nocedal J (1989) A tool for the analysis of quasi-Newton methods with application to unconstrained minimization. SIAM J Numer Anal 26(3):727–739
8. Chernousko FL (1994) State estimation for dynamic systems. CRC Press, Boca Raton, FL
9. Cook D, Fedorov V (1995) Constrained optimization of experimental design. Statistics, 26:129–178
10. Dembo A (1986) The relation between maximum likelihood estimation of structured covariance matrices and periodograms. IEEE Trans Acoustics, Speech and Signal Processing 34:1661–1662
11. Dennis JE, Wolkowicz H (1993) Sizing and least change secant methods. Techn. Report Dept. Combinatorics and Optimization Univ. Waterloo, CORR 90-02
12. Fedorov VV (1971) Theory of optimal experiments. Acad. Press, New York
13. Fletcher R (1991) A new variational result for quasi-Newton formulae. SIAM J Optim 1(1):18–21
14. Lewis AS, Overton ML (1996) Eigenvalue optimization. Acta Numer 5:149–190
15. Nesterov Yu, Nemirovsky A (1994) Interior-point polynomial methods in convex programming. Stud Appl Math, vol 13. SIAM, Philadelphia
16. Pardalos PM, Wolkowicz H (eds) (1998) Topics in semidefinite programming and interior-point methods. Fields Inst Commun. Amer. Math. Soc., Providence, RI
17. Pukelsheim F (1993) Optimal design of experiments. Wiley, New York
18. Rosen JB (1965) Pattern separation by convex programming. J Math Anal Appl, 10:123–134
19. Rousseeuw PJ, Leroy AM (1987) Robust regression and outlier detection. Wiley, New York
20. Scharf LL (1991) Statistical signal processing. Addison-Wesley, Reading, MA
21. Vandenberghe L, Boyd S (March 1996) Semidefinite programming. SIAM Rev 38(1):49–95
22. Vandenberghe L, Boyd S, Wu S-P (Apr. 1998) Determinant maximization with linear matrix inequality constraints. SIAM J Matrix Anal Appl 19(2):499–533

Semidefinite Programming: Optimality Conditions and Stability

ALEXANDER SHAPIRO
Georgia Institute Technol. Atlanta, Atlanta, USA

MSC2000: 90C22, 90C25, 90C31

Article Outline

Keywords
Duality
First Order Optimality Conditions
Second Order Optimality Conditions
Stability Analysis
See also
References

Keywords

Semidefinite programming; Duality; Tangent cone; Nondegeneracy; Complementarity condition; Lagrange multipliers; Stability; Sensitivity analysis

In this article we discuss optimization problems of the form

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & f(x) \\ \text{s.t.} & G(x) \preceq 0, \end{cases} \tag{1}$$

where $f(x)$ is a real valued function, $G(x)$ is a mapping from $\mathbf{R}^n$ into the space $\mathcal{S}^p$ of $p \times p$ symmetric matrices, and the notation $A \preccurlyeq 0$ (the notation $A \succcurlyeq 0$) means that A is a negative (positive) semidefinite matrix. We refer to (1) as a *semidefinite programming problem*. The above semidefinite programming problem is said to be *linear* if the objective function $f(x)$ is linear and the

mapping $G(x)$ is affine, i. e. is of the form

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & c^\top x \\ \text{s.t.} & A_0 + \sum\limits_{i=1}^n x_i A_i \preceq 0, \end{cases} \tag{2}$$

where $c \in \mathbf{R}^n$ and $A_0, \ldots, A_n \in \mathcal{S}^p$ are given matrices.

Let us observe that the constraint $G(x) \preccurlyeq 0$ can be written as $G(x) \in \mathcal{S}^p_-$, where

$$\mathcal{S}^p_- := \{Z \in \mathcal{S}^p : \ Z \preceq 0\}$$

denotes the set of $p \times p$ negative semidefinite symmetric matrices. Note that the set $\mathcal{S}^p_-$ forms a closed convex cone in the space $\mathcal{S}^p$, and that $\mathcal{S}^p$ can be viewed as a linear (vector) space of dimension $p(p+1)/2$. We equip $\mathcal{S}^p$ with the scalar (inner) product

$$A \bullet B := \operatorname{trace}(AB) = \sum_{i,j=1}^p a_{ij} b_{ij},$$

between two matrices $A = (a_{ij})$ and $B = (b_{ij})$.

With these observations at hand one can notice a certain similarity between problem (1) and *nonlinear programming* problems. In a nonlinear programming problem corresponding constraints can be written in the form $G(x) \in \mathbf{R}^m_-$, where $G(x)$ is a mapping from $\mathbf{R}^n$ into $\mathbf{R}^m$ and $\mathbf{R}^m_-$ is the negative orthant of the space $\mathbf{R}^m$. That is, in both cases the constraints can be formulated as convex cone inclusions. Let us note at this point that the *polar* (negative dual) of $\mathcal{S}^p_-$ is given by the cone $\mathcal{S}^p_+$ of positive semidefinite matrices.

Duality

With problem (1) is associated the following *Lagrangian* function

$$L(x, \Omega) := f(x) + \Omega \bullet G(x)$$

of $x \in \mathbf{R}^n$ and $\Omega \in \mathcal{S}^p$. Noting that, for a given x, the maximum of $L(x, \Omega)$ with respect to $\Omega \in \mathcal{S}^p_+$, equals $f(x)$ if $G(x) \in \mathcal{S}^p_-$ and $+\infty$ otherwise, we can write problem (1) in the form

$$\min_{x\in\mathbb{R}^n} \max_{\Omega\in\mathcal{S}^p_+} L(x, \Omega). \tag{3}$$

By interchanging the order in which the operators min and max are applied, we obtain

$$\max_{\Omega\in\mathcal{S}^p_+} \left\{ \phi(\Omega) := \min_{x\in\mathbb{R}^n} L(x, \Omega) \right\}. \tag{4}$$

We refer to the pair (1) and (4) as *primal* and *dual* problems, respectively. In particular, in case the primal problem is the *linear* semidefinite program (2), the dual problem takes the form

$$\begin{cases} \max\limits_{\Omega\in\mathcal{S}^p_+} & \Omega \bullet A_0 \\ \text{s.t.} & c_i + \Omega \bullet A_i = 0, \\ & i = 1, \ldots, n. \end{cases} \tag{5}$$

There is a long history in mathematical programming of considering dual problems. In semidefinite programming dual problems were studied in various forms, for example, in [1,9,12].

The optimal value of the primal problem is always greater than or equal to the optimal value of the dual problem. It is said that there is no *duality gap* between the primal and dual problems if their optimal values are equal. Note that in the semidefinite programming, the duality gap can happen even in the linear case. It is possible to give various regularity conditions, called constraint qualifications, ensuring the 'no duality gap' property. In the linear case, if $\bar{x}$ and $\bar{\Omega}$ are feasible points of the problems (2) and (5), respectively, then $\bar{x}$ is an optimal solution of (2), $\bar{\Omega}$ is an optimal solution of (5), and there is no duality gap between these problems if and only if the following complementarity condition

$$\bar{\Omega} \bullet G(\bar{x}) = 0 \tag{6}$$

holds, where $G(\bar{x}) = A_0 + \sum_{i=1}^n \bar{x}_i A_i$. This result can be extended to a certain class of convex semidefinite programming problems [10]. Note that since $\bar{\Omega}$ is a feasible point of (5) and hence $\bar{\Omega} \in \mathcal{S}^p_+$, and $\bar{x}$ is a feasible point of (2) and hence $G(\bar{x}) \in \mathcal{S}^p_-$, the complementarity condition (6) is equivalent to $\bar{\Omega} G(\bar{x}) = \mathbf{O}$, where $\mathbf{O}$ denotes the zero $p \times p$ matrix.

First Order Optimality Conditions

Consider now a (possibly nonlinear) semidefinite programming problem in the form (1), and suppose that $f(x)$ and $G(x)$ are continuously differentiable. Let $\bar{x}$ be a locally optimal solution of (1). With $\bar{x}$ is associated the following system of first order optimality conditions:

$$\begin{cases} \nabla_x L(\bar{x}, \Omega) = 0, \quad \Omega \succeq 0, \\ \Omega \bullet G(\bar{x}) = 0. \end{cases} \tag{7}$$

In case the semidefinite program is linear, the first and second conditions in (7) ensure that Ω is a feasible point of the dual problem (5), and the last condition corresponds to the complementarity condition (6). Therefore, in that case, if there exists a matrix $\Omega \in \mathcal{S}^p$ satisfying conditions (7), then Ω is an optimal solution of the corresponding dual problem. For general (possibly nonlinear) semidefinite programming problems, conditions (7) are obtained by linearization of $f(x)$ and $G(x)$ at the point $\overline{x}$.

It should be noted that existence of a matrix Ω satisfying the first order optimality conditions (7), is not guaranteed even in the linear case. If such a matrix exists we refer to it as a *Lagrange multipliers* matrix. It is possible to show that the set of Lagrange multipliers matrices is non empty and bounded if and only if the following constraint qualification holds: there exists a vector $h \in \mathbf{R}^n$ such that the matrix $G(\overline{x}) + dG(\overline{x})h$ is negative definite (i. e. belongs to the interior of $\mathcal{S}^p_-$), where

$$dG(\overline{x})h := \sum_{i=1}^{n} h_i \frac{\partial G(\overline{x})}{\partial x_i}$$

is the differential of G at $\overline{x}$.

The above constraint qualification can be viewed as an extension of the *Mangasarian–Fromovitz constraint qualification* [5], used in nonlinear programming. Note that in the linear case the partial-derivatives matrix $\frac{\partial G(\overline{x})}{\partial x_i}$ equals A_i, $i = 1, \ldots, n$, and hence the above constraint qualification is equivalent to the Slater condition: there exists $x^* \in \mathbf{R}^n$ such that the matrix $G(x^*)$ is negative definite. This equivalence also holds for *convex* semidefinite programming problems. For a discussion of constraint qualifications in cone constrained problems see [6,7,8,13].

It is possible to formulate the above constraint qualification in another equivalent form. Recall that for a convex closed subset K of a finite-dimensional vector space and $y \in K$, the *tangent cone* $T_K(y)$, to K at y, is formed by vectors z such that the distance from $y + tz$ to K is of order $o(t)$, $t \geq 0$. The condition

$$dG(\overline{x})\mathbb{R}^n + T_{\mathcal{S}^p_-}(G(\overline{x})) = \mathcal{S}^p \tag{8}$$

is another equivalent form of the above constraint qualification. It is possible to show that the tangent cone to the set $\mathcal{S}^p_-$, at $G(\overline{x})$, can be written in the form

$$T_{\mathcal{S}^p_-}(G(\overline{x})) = \left\{Z \in \mathcal{S}^p : \; E^\top Z E \preceq 0\right\},$$

where E is a $p \times (p - r)$ matrix of full column rank $p - r$ such that $G(\overline{x})E = \mathbf{O}$ and r is the rank of $G(\overline{x})$, e. g. [10]. (If $r = p$, i. e. $G(\overline{x})$ is negative definite and hence belongs to the interior of $\mathcal{S}^p_-$, then $T_{\mathcal{S}^p_-}(G(\overline{x}))$ coincides with the whole space $\mathcal{S}^p$.)

Consider now the set $\mathcal{W}_r$ of $p \times p$ symmetric matrices of rank r. This set forms a smooth manifold in the space $\mathcal{S}^p$, and the tangent space to $\mathcal{W}_r$ at the point $G(\overline{x})$ is given by

$$T_{\mathcal{W}_r}(G(\overline{x})) = \left\{Z \in \mathcal{S}^p : \; E^\top Z E = 0\right\}.$$

It follows from the above formulas that $T_{\mathcal{W}_r}(G(\overline{x}))$ coincides with the *lineality* space of the cone $T_{\mathcal{S}^p_-}(G(\overline{x}))$, i. e. it is the largest linear subspace of $T_{\mathcal{S}^p_-}(G(\overline{x}))$. It is said that the point $\overline{x}$ is *nondegenerate* ([2,10]) if the following condition holds

$$dG(\overline{x})\mathbb{R}^n + T_{\mathcal{W}_r}(G(\overline{x})) = \mathcal{S}^p. \tag{9}$$

The above condition (9) means that the mapping G intersects $\mathcal{W}_r$ *transversally* at the point $\overline{x}$. It immediately follows from the corresponding result in differential geometry that nondegeneracy, in a certain sense, is a *generic* property (cf. [11]). It is clear that constraint qualification (8) implies (9). Therefore we have by the above discussion that if the point $\overline{x}$ is nondegenerate, then there exists a Lagrange multipliers matrix Ω satisfying conditions (7). In fact it is not difficult to show that if $\overline{x}$ is nondegenerate, then such Lagrange multipliers matrix is *unique*.

Let us note now that it follows from the complementarity condition (6) that

$$\operatorname{rank}(\overline{\Omega}) + \operatorname{rank}(G(\overline{x})) \leq p.$$

It is said that the *strict complementarity* condition holds at $\overline{x}$ if there is a Lagrange multipliers matrix $\overline{\Omega}$ such that

$$\operatorname{rank}(\overline{\Omega}) + \operatorname{rank}(G(\overline{x})) = p.$$

It is possible to show that, in a certain sense, strict complementarity is also a generic property ([2]). Under the strict complementarity condition, nondegeneracy of $\overline{x}$ is a necessary and sufficient condition for uniqueness of the corresponding Lagrange multipliers matrix.

Second Order Optimality Conditions

In the convex (and, in particular, in the linear) case first order conditions (7) are also sufficient for a feasible point $\overline{x}$ to be an optimal solution of the problem (1).

However, even in the convex case second order optimality conditions can be an important part of the analysis, e. g. second order conditions are intimately related to stability properties of the corresponding optimal solutions. From now on we assume that $f(x)$ and $G(x)$ are twice continuously differentiable.

Let $\overline{x}$ be a *stationary* point of the problem (1), i. e. $\overline{x}$ is feasible and there exists a Lagrange multipliers matrix Ω satisfying first order optimality conditions (7). With $\overline{x}$ is associated the so-called *critical cone*

$$C(\overline{x}) := \left\{ h: \quad \begin{array}{l} h^\top \nabla f(\overline{x}) = 0, \\ dG(\overline{x})h \in T_{\mathcal{S}^p_-}(G(\overline{x})) \end{array} \right\}.$$

This critical cone represents those directions $h \in \mathbf{R}^n$ for which first order conditions (7) do not provide an information about optimality of $\overline{x}$. Let us denote by $\Lambda(\overline{x})$ the set of all Lagrange multipliers matrices Ω satisfying conditions (7), and suppose that the constraint qualification (8) holds. Recall that $\Lambda(\overline{x})$ is non empty and bounded if and only if the constraint qualification (8) is satisfied. We can write now second order conditions for $\overline{x}$ to be a locally optimal solution of the problem (1) as follows ([3,10]).

- Second order *necessary* conditions: for any critical direction $h \in C(\overline{x})$ there exists a Lagrange multipliers matrix $\overline{\Omega} \in \Lambda(\overline{x})$ such that

$$h^\top \nabla^2_{xx} L(\overline{x}, \overline{\Omega})h + h^\top H(\overline{x}, \overline{\Omega})h \geq 0. \tag{10}$$

- Second order *sufficient* conditions: for any nonzero critical direction $h \in C(\overline{x})$ there exists a Lagrange multipliers matrix $\overline{\Omega} \in \Lambda(\overline{x})$ such that

$$h^\top \nabla^2_{xx} L(\overline{x}, \overline{\Omega})h + h^\top H(\overline{x}, \overline{\Omega})h > 0. \tag{11}$$

Here $\nabla^2_{xx} L(\overline{x}, \overline{\Omega})$ stands for the Hessian matrix of second order partial derivatives and $H(\overline{x}, \overline{\Omega})$ denotes the $n \times n$ symmetric matrix whose ij-element is

$$[H(\overline{x}, \overline{\Omega})]_{ij} := -2 \operatorname{trace}\left(\overline{\Omega} G_i [G(\overline{x})]^\dagger G_j\right),$$

where $G_i := \frac{\partial G(\overline{x})}{\partial x_i}$ and $[G(\overline{x})]^\dagger$ denotes the Moore–Penrose pseudo-inverse of the matrix $G(\overline{x})$.

Note that there is no gap between the above second order necessary and sufficient conditions in the sense that the weak inequality sign in (10) is replaced by the strict inequality in (11). The additional term $h^\top H(\overline{x}, \overline{\Omega})h$, which appears in the above conditions, represents the *curvature* of the set $\mathcal{S}^p_-$. The matrix $H(\overline{x}, \overline{\Omega})$ is positive semidefinite, and therefore this 'curvature term' is always nonnegative.

In fact, the above second order sufficient conditions (11) are equivalent to the so-called *second order growth* condition: there exists a constant $\alpha > 0$ such that for all feasible x, i. e. satisfying $G(x) \in \mathcal{S}^p_-$, in a neighborhood of $\overline{x}$ the inequality

$$f(x) \geq f(\overline{x}) + \alpha \|x - \overline{x}\|^2 \tag{12}$$

holds. If the semidefinite programming problem is linear, then all elements of the Hessian matrix $\nabla^2_{xx} L(\overline{x}, \overline{\Omega})$ are zeros, and hence the first term in (10) and (11) vanishes. Nevertheless, even in the linear case the second (curvature) term can be positive, and the second order growth condition (12) can hold.

If the point $\overline{x}$ is nondegenerate, then there exists a unique Lagrange multipliers matrix $\overline{\Omega}$, and if, moreover, the strict complementarity condition holds, then the critical cone becomes a linear space and can be written as

$$C(\overline{x}) = \left\{ h: \ \sum_{i=1}^n h_i E^\top G_i E = 0 \right\}.$$

Stability Analysis

Suppose now that the considered semidefinite programming problem is subject to perturbations. That is, consider the following family of optimization problems

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} & f(x,u) \\ \text{s.t.} & G(x,u) \preceq 0, \end{cases} \tag{13}$$

depending on the parameter vector $u \in \mathbf{R}^m$ guiding perturbations of the above problem. We assume that $f(x,u)$ and $G(x,u)$ are twice continuously differentiable and that for a given value $u = u_0$ of the parameter vector, problem (13) coincides with the 'unperturbed' problem (1). For example, in the linear case we can view some of the matrices A_i as parameter vectors which are subject to perturbations.

Let us denote by $S(u)$ the set of optimal solutions of the parametric problem (13). What can be said about continuity properties of $S(u)$, as a function of the pa-

rameter vector u? Note that $S(u)$ can have more than one element and can be empty. It is possible to construct examples where the optimal-solutions set $S(u_0)$, of the unperturbed problem, is non empty while $S(u)$ is empty for arbitrary small changes in u. Of course such examples are pathological and unsuitable for numerical analysis since arbitrary small perturbations of the data may result in unsolvability of the considered problem.

It is possible to give various conditions ensuring continuity (upper semicontinuity) of $S(u)$. For example, in the convex (linear) case, if $S(u_0) = \{\overline{x}\}$ is a singleton (i. e. the unperturbed problem has the unique optimal solution $\overline{x}$) and the Slater condition holds, then any optimal solution $\widehat{x}(u) \in S(u)$, of the parametric problem (13), converges to $\overline{x}$ as $u \to u_0$. Yet even if $\widehat{x}(u)$ is continuous at $u = u_0$, the rate at which it changes can be much faster than the rate of change of u, i. e. small perturbations in u can bring large changes in the corresponding optimal solution. In that case the problem is said to be *ill-conditioned* and may be difficult for a numerical solution.

It is said that an optimal solution $\overline{x}$, of the unperturbed problem, is *stable* if for all u sufficiently close to u_0 the optimal-solutions set $S(u)$ is non empty and $\widehat{x}(u) \to \overline{x}$, as $u \to u_0$, for any $\widehat{x}(u) \in S(u)$. If, moreover, there is a positive constant κ such that

$$\left\|\widehat{x}(u) - \overline{x}\right\| \leq \kappa \left\|u - u_0\right\|, \tag{14}$$

then $\overline{x}$ is said to be *Lipschitz stable*.

It is not difficult to show that if the constraint mapping $G(x, u) = G(x)$ does not depend on u, and hence the feasible set of (13) is fixed, then the second order growth condition (12) is a sufficient condition for a stable optimal solution $\overline{x}$ to be Lipschitz stable. The general case is more subtle and, in order to ensure Lipschitz stability of $\overline{x}$, some additional conditions are required (see [4] for an extensive discussion of that type results).

Let us finally remark that if the point $\overline{x}$ is nondegenerate and the strict complementarity condition holds, then the optimal solution $\widehat{x}(u)$ is a continuously differentiable function of the parameter vector ([10]). In that case one can approximately evaluate the constant κ in (14) by using the corresponding linear approximation (first order Taylor expansion) of $\widehat{x}(u)$ at u_0. It turns out that the rate of change of $\widehat{x}(u)$ is closely related to conditioning of the matrix $\nabla^2_{xx} L(\overline{x}, \overline{\Omega}) + H(\overline{x}, \overline{\Omega})$, used in the corresponding second order optimality conditions, restricted to the critical cone $C(\overline{x})$ which is a linear space in that case.

See also

- ▶ Duality for Semidefinite Programming
- ▶ Interior Point Methods for Semidefinite Programming
- ▶ Semidefinite Programming and Determinant Maximization
- ▶ Semidefinite Programming and Structural Optimization
- ▶ Semi-infinite Programming, Semidefinite Programming and Perfect Duality
- ▶ Solving Large Scale and Sparse Semidefinite Programs

References

1. Alizadeh F (1995) Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM J Optim 5:13–51
2. Alizadeh F, Haeberly JP, Overton ML (1997) Complementarity and nondegeneracy in semidefinite programming. Math Program 77:111–128
3. Bonnans JF, Cominetti R, Shapiro A (1999) Second order optimality conditions based on parabolic second order tangent sets. SIAM J Optim 9:466–492
4. Bonnans JF, Shapiro A (1999) Optimization problems with perturbations: A guided tour. SIAM Rev 40:228–264
5. Mangasarian O, Fromovitz S (1967) The Fritz-John necessary optimality conditions in the presence of equality and inequality constraints. J Math Anal Appl 7:37–47
6. Penot JP (1982) On regularity conditions in mathematical programming. Math Program Stud 19:167–199
7. Robinson SM (1976) First order conditions for general nonlinear optimization. SIAM J Appl Math 30:597–607
8. Robinson SM (1976) Stability theorems for systems of inequalities. Part II: differentiable nonlinear systems. SIAM J Numer Anal 13:497–513
9. Shapiro A (1985) Extremal problems on the set of nonnegative definite matrices. Linear Alg & Its Appl 67:7–18
10. Shapiro A (1997) First and second order analysis of nonlinear semidefinite programs. Math Program B 77:301–320
11. Shapiro A, Fan MKH (1995) On eigenvalue optimization. SIAM J Optim 5:552–569
12. Wolkowicz H (1981) Some applications of optimization in matrix theory. Linear Alg & Its Appl 40:101–118
13. Zowe J, Kurcyusz S (1979) Regularity and stability for the mathematical programming problem in Banach spaces. Appl Math Optim 5:49–62

Semidefinite Programming and the Sensor Network Localization Problem, SNLP

CLAYTON W. COMMANDER[1], MICHELLE A. RAGLE[2], YINYU YE[3]

[1] Air Force Research Laboratory, Munitions Directorate, Dept. of Industrial and Systems Engineering, University of Florida, Gainesville, USA

[2] Center for Applied Optimization, Dept. of Industrial and Systems Engineering, University of Florida, Gainesville, USA

[3] Dept. of Management Science and Engineering, Electrical Engineering, Stanford University, Stanford, USA

Article Outline

Keywords
Introduction
Organization
Idiosyncrasies
Formulation
Methods
Review of Solution Approaches
Semidefinite Programming Model
Conclusion
See also
References

Keywords

Sensor network localization problem; Euclidean distance matrix completion problem [2,3]; Graph realization problem

Introduction

Research on ad hoc wireless sensor networks has increased greatly in recent years [24]. Sensor networks usually consist of a large number of sensors which are deployed to collect data of interest. Such networks are versatile tools which provide a low-cost method of target tracking, as well as monitoring seismic activity, temperature, sound levels, and light [9]. Information gathered by the sensors is only useful if the positions of the sensors are known. However, it is often the case that the use of a GPS system is too costly or, consumes too much power, or the network is being deployed in a location in which GPS is denied [21].

Recently, techniques have been developed which estimate the node locations based on a mixture of distance measurements and angle measurements between pairs of nodes in the network. This problem is referred to as the Sensor Network Localization Problem, (SNLP) and can be formally stated as follows: Given the true positions of some of the nodes and the pair-wise distances between some nodes, how can the positions of all of the nodes be estimated? [6,9,28].

Organization

Throughout the article, we will investigate the SNLP. In the following section, we formally define the problem statement and in Sect. "Methods", we review several solution techniques which appear throughout the literature. In Subsect. "Semidefinite Programming Model", we describe a semidefinite programming (SDP) model for the problem. We then provide a general assessment of this approach and describe some implementation details. We highlight this method specifically because of its advantages over heuristic methods. Particularly, the SDP method is known to localize any network whenever a unique solution exists, and to do so in polynomial time. We provide some concluding remarks in Sect. "Conclusion" and indicate directions of future research. Finally, a list of cross references is provided in Sect. "See also". We conclude this section with an introduction to some of the symbology that will appear most prevalently throughout the article.

Idiosyncrasies

Here we briefly introduce some of the symbols and notations we will employ throughout this paper. Define the trace of a symmetric matrix A, denoted $\mathrm{Trace}(A)$ as the sum of the diagonal entries. The standard trace inner product of two matrices A and B is given as $\langle A, B\rangle = \mathrm{Trace}(A^{\mathrm{T}}B)$. The 2-norm of a vector x is denoted as $||x||$ and is defined to be $\sqrt{\langle x, x\rangle}$. A positive semidefinite matrix A will be denoted as $A \succeq 0$. Agree to let I_d and $\mathbf{0}_d$ respectively represent the identity matrix and a vector containing all zeros, both with dimension $d \in \mathbb{Z}$. Finally, we will use *italics* for emphasis, $\mathcal{CALLIGRAPHY}$

to refer to formulations, and Small Caps for problem names. Any other locally used terms and symbols will be defined in the sections in which they appear.

Formulation

A wireless sensor network is typically made up of a number of densely distributed sensors that collect data. An instance of the SNLP consists of a set of m so-called *anchor* points whose positions are known a priori [9,12,22]. The object is to determine the location of n *sensor* points in the system based upon information obtained from the anchor sensors. Let the anchor points and the sensor points be respectively denoted as $a_1, a_2, \ldots, a_m \in R^d$ and $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$. The Euclidean distances $\bar{d}_{kj}$ between points a_k and x_j for some k,j, and d_{ij} between x_i and x_j for some $i < j$ are also given. Let $N_a = \{(k, j) : \bar{d}_{k,j} \text{ is specified}\}$ denote the sensor/sensor pairs and $N_x = \{(i, j) : i < j, d_{ij} \text{ is specified}\}$ represent the sensor/anchor. Then the sensor network localization problem as defined in [9] is to find the localization (estimated position) of $x_i, x_2, \ldots, x_n \in \mathbb{R}^d$ such that:

$$\begin{aligned} \mathcal{SNLP}: \quad & \|a_k - x_j\|^2 = \bar{d}_{kj}^2, \forall (k, j) \in N_a, \text{ and} \\ & \|x_i - x_j\|^2 = d_{ij}^2, \forall (i, j) \in N_x. \end{aligned} \tag{1}$$

From this seemingly simple formulation, many difficult questions arise. For a given instance of the SNLP, does this instance have a realization in the required dimension? If so, is the realization unique? We should note that these two seemingly related questions are quite different from a computational perspective. It has been shown that determining if an instance of the SNLP has a unique realization in $\mathbb{R}^2$ can be determined efficiently under certain assumptions [20]. On the other hand, it remains $\mathcal{NP}$-complete [17] to compute a realization on the plane, even if the instance is *known* to have a unique realization [5]. This is the main problem of interest in this article.

Methods

In this section, we review several solution methods which have been applied to the SNLP. Particularly in Subsect. "Semidefinite Programming Model", we highlight the techniques of Ye et al. [1,8,9,28] and the application of semidefinite programming methods for efficiently computing solutions to large-scale instances of the SNLP under a variety of circumstances.

Review of Solution Approaches

Several techniques have been applied to the SNLP, all having some redeeming qualities [9,16,18]. Several techniques involve the use of distance or angle measurements between the anchor points in order to compute a localization [13,15,23,25,26,27]. Another common technique used by Bulusu et al. [11] and Howard et al. [19] is to employ a grid or a set of surveyed points whose locations are known. Then, the sensor localization is attempted using the relative distances between sensors and the set of *beacon* points [9].

The so-called "DV-Hop" technique of Niculescu and Nash [23] is an efficient method in dense topologies whereby the anchor nodes flood the network with their location information. This allows other points to *triangulate* their positions based on the information of the anchor nodes. This information is then passed along to other sensor nodes who use the combined locations to triangulate their positions. However, for widely dispersed and irregular topologies, the relative errors in the node estimation tends to be fairly substantial.

A similar technique proposed by Savarese et al. [25] uses a method similar to the "DV-Hop" algorithm described above to provide a rough estimate of the location information. These estimates are then improved by applying a least-squares triangulation using these estimates as well as a new collection of estimated positions [9].

The "iterative multilateration" technique of Savvides et al. [26] is another effective method especially when the number of anchor nodes is relatively high. This method calculates via triangulation the positions of those nodes that are adjacent to at least three anchor points. Then these new localized nodes become anchor points and the process continues. In the end every node in the network has become an anchor.

In [15], the so-called "multidimensional scaling" algorithm is proposed. The heuristic begins by making an initial estimate of the node positions based solely on the connectivity and basic distance and angle information of the *non-anchor* nodes. Then using a variant of singular value decomposition [29], a map is generated

of the relative locations of the nodes. Finally, these estimates are greatly improved and an absolute global map is produced by taking into account the locations of the anchor nodes and updating the estimates accordingly.

The work of Doherty et al. [13] involves a technique in which linear bounding hyperplanes are used to model the proximity constraints on the nodes which can communication with each other as convex constraints [1,9]. However, these constraints are often too loose and provide solutions which are not helpful in terms of calculating the unique realization of the sensors.

As we see, the drawback with most sensor network localization techniques involving heuristics is that they do not always find a unique solution even when it exists, or require excessive computation time to do so [28]. A recently developed method introduced by So and Ye in [28] uses a semidefinite programming (SDP) model that guarantees the discovery of a unique solution when it exists. Furthermore, the solution can be computed in polynomial time. In the following subsection, we present the SDP model, discuss the motivation behind using this approach and analyze some properties of the model.

Semidefinite Programming Model

A semidefinite program is a convex optimization problem where the objective function is linear and the constraint is defined by a linear matrix inequality. Given a vector $c \in \mathbb{R}^m$, and $m+1$ symmetric matrices $F_0, F_1, \ldots, F_m \in \mathbb{R}^{nxn}$, a semidefinite program can be written in the form:

$$\min_{x \in \mathbb{R}^m} \{c^T x | F(x) \succeq 0\},$$

where $F(x) = F_0 + \sum_{i=1}^m x_i F_i$, and $F(x) \succeq 0$ implies that $F(x)$ is positive semidefinite [14,30]. Hence, both the objective function and the constraint are convex, and therefore semidefinite programs are closely related to linear programs, and many algorithms for solving linear programming problems have generalizations that apply to semidefinite programs as well [30].

In [28], the authors note that (1) is a non-convex optimization problem, which is difficult to solve in general. They propose a SDP relaxation by converting the nonconvex quadratic distance constraints into linear constraints as follows. Specifically, let $X = [x_1, x_2, \ldots, x_n]$ be the $d \times n$ matrix which we are are trying to determine. Let $e_{ij} \in \mathbb{R}$ be the vector where the ith position is 1, the jth position is -1, and all other entries are zeros. Then for all $(i, j) \in N_x$, we have that:

$$||x_i - x_j||^2 = e_{ij}^{\mathrm{T}} X^{\mathrm{T}} X e_{ij}\,. \tag{2}$$

Furthermore, for all $(k, j) \in N_a$ it follows that

$$||a_k - x_j||^2 = (a_k; e_j)^{\mathrm{T}} [I_d; X]^{\mathrm{T}} [I_d; X](a_k; e_j)\,, \tag{3}$$

where e_j is a vector of all zeros except for -1 at entry j, and $(a_k; e_j) \in \mathbb{R}^{d+n}$ is a vector consisting of a_k "on top of" e_j [9]. Using these definitions, we can reformulate the problem as follows.

$$\text{Find } X \in \mathbb{R}^{d \times n} \text{ and } Y \in \mathbb{R}^{n \times n} \tag{4}$$

such that

$$e_{ij}^{\mathrm{T}} Y e_{ij} = d_{ij}^2, \forall (i, j) \in N_x\,, \tag{5}$$

$$(a_k; e_j)^{\mathrm{T}} \begin{pmatrix} I_d & X \\ X^{\mathrm{T}} & Y \end{pmatrix} (a_k; e_j) = \tilde{d}_{kj}^2, \forall (k, j) \in N_a\,, \tag{6}$$

$$Y = X^{\mathrm{T}} X\,. \tag{7}$$

The intuition behind the SDP formulation is to relax constraint (7) to $Y \succeq X^{\mathrm{T}} X$; thus implying that $Y - X^{\mathrm{T}} X$ is positive semidefinite. Boyd et al. [10] among others have shown that a positive semidefinite matrix $Y - X^{\mathrm{T}} X$ can be expressed as

$$Z = \begin{pmatrix} I_d & X \\ X^{\mathrm{T}} & Y \end{pmatrix} \succeq 0. \tag{8}$$

Define $Z_{1:d,1:d}$ to be the $d \times d$ principle submatrix of Z. Then the SDP relaxation of the SNLP as given in [28], is to find $Z \in \mathbb{R}^{(d+n)\times(d+n)}$ to:

$$\mathcal{SDP}: \quad \text{maximize} \quad 0 \tag{9}$$

subject to

$$Z_{1:d,1:d} = I_d\,, \tag{10}$$

$$\langle (\mathbf{0}; e_{ij})(\mathbf{0}; e_{ij})^{\mathrm{T}}, Z \rangle = d_{ij}^2, \forall (i, j) \in N_x\,, \tag{11}$$

$$\langle (a_k; e_j)(a_k; e_j)^{\mathrm{T}}, Z \rangle = \tilde{d}_{kj}^2, \forall (k, j) \in N_a\,, \tag{12}$$

$$Z \succeq 0\,. \tag{13}$$

Notice that by definition, any feasible solution matrix Z must have at least rank d [9]. We can formulate the dual of the SDP relaxation as

$\mathcal{SDP}$-$\mathcal{D}$: minimize

$$\langle I_d, V\rangle + \sum_{(i,j)\in N_x} y_{ij} d_{ij}^2 + \sum_{(k,j)\in N_a} w_{kj} \bar{d}_{kj}^2 \tag{14}$$

subject to

$$\begin{pmatrix} V & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} + \sum_{(i,j)\in N_x} y_{ij}(\mathbf{0}; e_{ij})(\mathbf{0}; e_{ij})^{\mathrm{T}} + \sum_{(k,j)\in N_a} w_{kj}(a_k; e_j)(a_k; e_j)^{\mathrm{T}} \succeq 0\,. \tag{15}$$

Notice that the dual formulation is always feasible. In particular having $V = 0$, $y_{ij} = 0$ for all $(i, j) \in N_x$, and $w_{kj} = 0$ for all $(k, j) \in N_a$ forms a feasible solution.

In [28], So and Ye postulate and prove several results regarding the above formulations. We will highlight the key theorems and provide a basic analysis. For detailed proofs and a more in-depth study, see [28].

The first result provides a class of instances for which the SDP relaxation is exact, i. e. for instances when the matrix Z has rank d. Suppose that formulation $\mathcal{SDP}$ is feasible. This implies that the distance measurements d_{ij} and $\bar{d}_{kj}$ are exact for the positions $\bar{X} = [\bar{x}_1, \ldots \bar{x}_n]$. Then, we have the following result.

Theorem 1 *Let $\bar{Z}$ be a feasible solution for $\mathcal{SDP}$ and $\bar{U}$ be an optimal slack matrix of $\mathcal{SDP}$-$\mathcal{D}$. Then by the duality theorem for semidefinite programming [4], it follows that:*

1. $\langle \bar{Z}, \bar{U}\rangle = 0$;
2. $\mathrm{rank}(\bar{Z}) + \mathrm{rank}(\bar{U}) \le d + n$;
3. $\mathrm{rank}(\bar{Z}) \ge d$ *and* $\mathrm{rank}(\bar{U}) \le n$.

A immediate consequence of this theorem is that for optimal dual slack matrices $\bar{U}$ such that $\mathrm{rank}(\bar{U}) = n$, it follows that $\mathrm{rank}(\bar{Z}) = d$. Therefore, formulation $\mathcal{SNLP}$ is equivalent to formulation $\mathcal{SDP}$ implying that the $\mathcal{SNLP}$ formulation can be solved optimally in polynomial time [28].

The next theorem establishes the existence of a large group of efficiently localizable graphs.

Theorem 2 *Suppose that the network in question is connected. Then the following are equivalent:*

1. *Problem $\mathcal{SNLP}$ is uniquely localizable.*
2. *The max-rank solution matrix of $\mathcal{SDP}$ has rank d.*
3. *The solution matrix of $\mathcal{SDP}$, represented by (8) , satisfies $Y = X^T X$.*

This theorem has several significant implications. First, we have that as long as $\mathcal{SNLP}$ has a unique localization, then it can be computed in polynomial time by solving the corresponding semidefinite relaxation. The converse also holds. That is, if the solution matrix to the semidefinite relaxation X has rank d, then X is the unique localization for formulation $\mathcal{SNLP}$ [28]. Lastly, as we mentioned above we have the existence of a family of graphs for which the localization can be efficiently computed despite the underlying $\mathcal{NP}$-completeness of the SNLP in general.

The seminal work of So and Ye [28] which we highlighted above provides a baseline to which many extensions can be made. To begin with, the results presented above are based on the assumption that the distance measurements are exact. The work of Biswas et al. in [9] provides extensions to handle inaccurate and incomplete measurements. This greatly improves the robustness of the SDP formulation, making the model more applicable to real-world scenarios in which inaccuracies are inevitable. Furthermore, in [1] the authors provide SDP formulations of the SNLP which incorporate angle information which can be used alone or in concert with distance information to calculate sensor realizations. This method is particularly useful when the sensors can detect multiple angles [1]. We see that many extensions are possible and that the by using semidefinite programming methods, a large class of sensor network localization problems are able to be solved more efficiently and effectively than by previous heuristic techniques.

Conclusion

The focus of this article was the sensor network localization problem (SNLP), with particular attention given to a set of robust solution technique based on a semidefinite programming model. After an introduction to the problem, we highlighted several solution approaches which have been applied. Next, we presented an analyzed the SDP formulation of So and Ye [28]. The results proved for the SDP relaxation of the SNLP have provided a framework which can be extended to other

problems in distance geometry in which angle and distance information are mutual between pairs of points. Such problems include Euclidean ball packing and most recently 3-dimensional molecule conformation problems [7].

See also

- Graph Realization via Semidefinite Programming
- Semidefinite Programming and Determinant Maximization
- Semidefinite Programming: Optimality Conditions and Stability
- Semidefinite Programming and Structural Optimization
- Solving Large Scale and Sparse Semidefinite Programs

References

1. Aghajan H, Biswas P, Ye Y (2005) Semidefinite programming algorithms for sensor network localization using angle of arrival information. In: 39th Annual Asilomar Conference on Signals, Systems and Computers
2. Alfakih AY, Wolkowicz H (1998) On the embeddability of weighted graphs in Euclidean spaces, Technical Report CORR 98–12. University of Waterloo, Dept. of Combinatorics and Optimization
3. Alfakih AY, Wolkowicz H (2002) Euclidean distance matrices and the molecular conformation problem, Technical Report CORR 2002–17. University of Waterloo, Dept. of Combinatorics and Optimization
4. Alizadeh F (1995) Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM J Optim 5:13–51
5. Aspnes J, Goldenberg D, Yang R (2004) On the computational complexity of sensor network localization. In: Algorithmic Aspects of Wireless Sensor Networks: First International Workshop, ALGOSENSORS 2004, Turku, Finland, July 16, 2004. Lecture Notes in Computer Science, vol 3121. Springer, 2004, pp 32–44
6. Beutel J (1999) Geolocation in a picoradio environment. M.S. thesis. Berkeley, ETH Zurich
7. Biswas P, Liang T, Toh K, Ye Y (2005) An SDP based approach for anchor-free 3d graph realization. SIAM J Sci Comp, submitted
8. Biswas P, Liang T, Toh K, Ye Y, Wang T (2006) Semidefinite programming approaches for sensor network localization with noisy distance measurements. IEEE Trans Autom Sci Eng 3(4):360–371
9. Biswas P, Lian T, Wang T, Ye Y (2006) Semidefinite programming based algorithms sensor network localization. ACM Trans Sens Netw 2(2):188–220
10. Boyd S, El Ghaoui L, Feron E, Balakrishnan V (1994) Linear Matrix Inequalities in System and Control Theory, vol 15 of Studies in Applied Mathematics. SIAM, Philadelphia
11. Bulusu N, Heidemann J, Estrin D (2000) GPS-less low cost outdoor localization for very small devices. Technical report, Computer Science Department, University of Southern California
12. Chong C, Kumar SP (2003) Sensor networks: Evolution, opportunities, and challenges. Proc IEEE 91(8):1247–1256
13. Doherty L, Ghaoui LE, Pister SJ (2001) Convex position estimation in wireless sensor networks. IEEE Infocom 3:1655–1663
14. Fan MKH, Gong Y (1997) Eingenvalue multiplicity estimate in semidefinite programming. J Optim Theory Appl 94(1):55–72
15. Fromherz M, Ruml W, Shang Y, Zhang Y (2004) Localization from connectivity in sensor networks. IEEE Trans Parallel Distrib Syst 15(11):961–974
16. Ganesan D, Krishnamachari B, Woo A, Culler D, Estrin D, Wicker S (2002) An empirical study of epidemic algorithms in large scale multihop wireless networks. Technical report. University of California, Los Angeles
17. Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. WH Freeman and Company, San Francisco
18. Hightower J, Borriello G (2001) Location systems for upbiquitous computing. Computer 34:57–66
19. Howard A, Matarić M, Sukhatme G (2001) Relaxation on a mesh: a formalism for generalized localization. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Wailea, Hawaii, pp 1055–1060
20. Jackson B, Jordán T (2005) Connected rigidity matroids and unique realizations of graphs. J Comb Theory B 94:1–29
21. Kaplan E (1996) Understanding GPS: Principles and Applications. Artech House, Norwood
22. Langendoen K, Reijers N (2003) Distributed localization in wireless sensor networks: a quantitative comparison. Comp Netw 43:499–518
23. Niculescu D, Nath B (2001) Ad hoc positioning system (aps). IEEE GLOBECOM 1:2926–2931
24. Resende MGC, Pardalos PM (2006) Handbook of Optimization in Telecommunications. Springer, New York
25. Savarese C, Langendoen K, Rabay J (2002) Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In: USENIX Technical Annual Conference, Monterey, June 2002, pp 317–328
26. Savvides A, Han CC, Srivastava MB (2001) Dynamic fine-grained localization in ad-hoc sensor networks. In: Proceedings of the Fifth International Conference on Mobile Computing and Networking (Mobicom 2001), Rome, July, pp 166–179
27. Savvides A, Park H, Srivastava MB (2002) The bits and flops of the n-hop multilateration primitive for node localization problems. In: Proceedings of the First International Work-

shop on Wireless Networks and Applications held in conjunction with Mobicom, Atlanta, 28 Sept 2002
28. So A, Ye Y (2006) Theory of semidefinite programming for sensor network localization. Mathematical Programming, published online, DOI: 10.1007/s10107–006-0040–1
29. Strang G (1998) Introduction to Linear Algebra, 3rd edn. Wellesley-Cambridge Press, Wellesley
30. Vandenberghe L, Boyd S (1996) Semidefinite programming. SIAM Rev 38:49–95

Semidefinite Programming and Structural Optimization

AHARON BEN-TAL[1], ARKADI NEMIROVSKI[1], JOCHEM ZOWE[2]
[1] Fac. Industrial Engineering and Management, Technion: Israel Institute Technol., Technion-City, Haifa, Israel
[2] Institute Applied Math., University Nürnberg–Erlangen, Erlangen, Germany

MSC2000: 90C25, 90C27, 90C90

Article Outline

Keywords
Truss Design
Shape Design
'Standard Case' of the SD Problem
  Standard Case
  The Set of Loads
Semidefinite Reformulation of P_{ini}
Multiload Truss Design
Robust Obstacle-Free Truss Design
Multiload Shape Design
Robust Obstacle-Free Shape Design
Computational Issues
See also
References

Keywords

Optimal design; Linearly elastic systems; Semidefinite programming; Interior point methods

Structural design (SD) deals with optimal design of *linearly elastic mechanical constructions*. Mathematically, the SD problem is as follows. We are considering a mechanical construction S with finitely many degrees of freedom M, so that virtual displacements of S are specified by vectors $w \in \mathbf{R}^M$. We are also given a set $W \subset \mathbf{R}^M$ of *kinematically admissible displacements*. The energy of elastic deformation of S under a displacement w is a nonnegative quadratic form $w^T Bw/2$ of the displacement. B is a symmetric positive semidefinite matrix characterizing S; this matrix is assumed to depend linearly on the vector t of design parameters of the construction: $B = B(t)$.

The construction can be subject to an external *load*; mathematically, a load is a vector $f \in \mathbf{R}^M$. The equilibrium displacement W_f caused by the load minimizes the potential energy $w^T B(t)w/2 - f^T w$ over $w \in W$:

$$w_f \in \operatorname*{Argmax}_{w \in W} \left[f^T w - \frac{1}{2} w^T B(t) w \right]; \tag{1}$$

the corresponding optimal value

$$\mathrm{compl}(f; t) = \sup_{w \in W} \left[f^T w - \frac{1}{2} w^T B(t) w \right] \tag{2}$$

— the *compliance* of S under load f — indicates how stiff is the construction w.r.t. the load (the less is the compliance, the better).

The SD problem in its general setting is:

- Given a set $F \subset \mathbf{R}^M$ of tentative loads and a set T of admissible values of the design vector t, find $t \in T$ which minimizes the worst-case, w.r.t. loads from F,
- compliance:

$$(P_{ini}) \quad \begin{cases} \min & \mathrm{compl}_F(t) \\ & \equiv \sup_{f \in F} \mathrm{compl}(f; t) \\ \text{s.t.} & t \in T. \end{cases}$$

The outlined general setting has two particular cases of especial interest.

Truss Design

A *truss* is a construction, like an electric mast or the Eifel Tower, comprised of thin *elastic bars* linked with each other at *nodes*. In the standard truss topology design problem the nodes form a given finite set in $\mathbf{R}^d$ ($d = 2$ for planar and $d = 3$ for spatial trusses), and all pair connections of the nodes by bars are allowed. Virtual displacements of a node form a given linear subspace of $\mathbf{R}^d$, and the space $\mathbf{R}^M$ of virtual displacements

of the truss is the direct product of these subspaces over all nodes. The set W of admissible displacements is cut off $\mathbf{R}^M$ by a number of inequality constraints (typically linear) representing *obstacles* – restrictions on the displacements of the nodes coming from absolutely rigid partial supports.

The design variables of the truss are volumes t_ℓ of all tentative bars, and the corresponding matrix $B(t)$ is $B(t) = \sum_{\ell=1}^N b_\ell b_\ell^\top t_\ell$, where N is the number of tentative bars and the vectors $b_\ell \in \mathbf{R}^M$ are readily given by the geometry of the nodal set. An external load – a collection of physical forces acting at the nodes of truss – can be identified with a vector $f \in \mathbf{R}^M$, and the equilibrium displacement and compliance of truss associated with load f are given by (1), (2), respectively.

The set T of feasible design vectors always is a subset of $\mathbf{R}^N_+$ (bar volumes must be nonnegative) satisfying the *resource constraint* $\sum_{\ell=1}^N t_\ell \le v$ (upper bound on the weight of the truss); the description of T may include also some other, normally linear, constraints.

Shape Design

A *shape* is a construction comprised of material continuously distributed in a given 2D or 3D domain Ω, the mechanical properties of the material varying from point to point. Thus, a shape is a distributed mechanical system with infinitely many degrees of freedom. In order to get a computationally tractable model, a *finite element approximation* is applied. Specifically,

- the infinite-dimensional space of displacements of the actual construction (the space of vector fields on Ω) is approximated by its finite-dimensional subspace $\mathbf{R}^M$;
- Ω is partitioned into N cells C_ℓ, $\ell = 1, \dots, N$, and the mechanical properties of the material are assumed to be constant within every cell.

  With this approximation, the energy of elastic deformation of shape under displacement w is

$$E(w) = \frac{1}{2}\sum_{\ell=1}^N v_\ell^{-1}\,\mathrm{Tr}\left(t_\ell \int_{C_\ell} e_P(w) e_P^\top(w)\,dP\right), \tag{3}$$

  where Tr stands for the trace, and
- v_ℓ is the d-dimensional volume of C_ℓ;
- $e_P(w)$ is the *strain tensor* associated with displacement w at a point $P \in \Omega$; the only property of this tensor important in our context is that $e_P(w)$ is an L_∞ function of P taking values in the Euclidean space $\mathbf{R}^D$ ($D = d(d+1)/2$) linearly depending on w;
- $v_\ell^{-1} t_\ell$ is the 'rigidity tensor' of the material specifying the mechanical properties of the material in cell C_ℓ; mathematically, t_ℓ is a symmetric positive semidefinite $D \times D$ matrix.

After finite element approximation, the set of kinematically admissible displacements becomes a subset W of $\mathbf{R}^M$, an external load acting at a shape can be identified with a vector $f \in \mathbf{R}^M$, the equilibrium displacement w_f of the shape minimizes the potential energy $E(w) - f^\top w$:

$$w_f \in \operatorname*{Argmax}_{w \in W}\left[f^\top w - E(w)\right], \tag{4}$$

and the rigidity properties of the shape w.r.t. the load are measured by the compliance

$$\mathrm{compl}(t; f) = \sup_{w \in W}\left[f^\top w - E(w)\right]. \tag{5}$$

The set T of feasible design vectors is always a subset of the set $(\mathbf{S}^D_+)^N$, $\mathbf{S}^D_+$ being the cone of positive semidefinite $D \times D$ matrices (the rigidity tensors must be positive semidefinite). Typical additional restrictions defining T are the 'resource constraints' imposed on the quantities $\mathrm{Tr}(t_\ell)$ (these quantities in a sense measure densities of the material in the cells). The most important case is the one of

$$T = \left\{ t = (t_1, \dots, t_N) : \begin{array}{c} t_\ell \in \mathbf{S}^D,\ t_\ell \succeq 0, \\ \underline{\rho}_\ell \le \mathrm{Tr}(t_\ell) \le \overline{\rho}_\ell, \\ l = 1, \dots, N; \\ \sum_{\ell=1}^N \mathrm{Tr}(t_\ell) \le v \end{array} \right\}, \tag{6}$$
$$\left\{ \begin{array}{l} 0 \le \underline{\rho}_\ell \le \overline{\rho}_\ell < \infty, \quad l = 1, \dots, N; \\ \sum_{\ell=1}^N \underline{\rho}_\ell < v, \end{array} \right.$$

where $\mathbf{S}^D$ is the space of symmetric $D \times D$ matrices and the relation $A \succcurlyeq B$ for $A, B \in \mathbf{S}^D$ means that $A - B$ is symmetric positive semidefinite.

'Standard Case' of the SD Problem

The truss and the shape problems are covered by a single particular case of ($\mathrm{P_{ini}}$), the one where the 'design

variables' t_ℓ are positive semidefinite symmetric matrices of certain row dimension D, the constraints defining T are restrictions on the vectors comprised of traces of these matrices, and

$$B(t) = \sum_{\ell=1}^{N} \sum_{s=1}^{S} b_{\ell s} t_\ell b_{\ell s}^\top, \tag{7}$$

where $b_{\ell s}$ are given $M \times D$ matrices.

Indeed, the truss problem clearly fits the indicated scheme (with $D = S = 1$, $\nu_\ell = 1, \mathrm{l} = 1, \ldots, N$). In the case of the shape problem, there always exist S 'quadrature grids' $\{P_{\ell s} \in C_\ell\}_{s=1}^S$ and 'quadrature weights' $\{\gamma_{\ell s}^2\}_{s=1}^S$, $\ell = 1, \ldots, N$, such that

$$\frac{1}{\nu_\ell} \int_{C_\ell} e_P(w) e_P^\top(w)\, dP = \sum_{s=1}^{S} \gamma_{ls}^2 e_{P_{\ell s}}(w) e_{P_{\ell s}}^\top(w)$$

identically in $w \in \mathbf{R}^M$. Specifying matrices $b_{\ell s}$ according to

$$b_{\ell s}^\top w = \gamma_{\ell s} e_{P_{\ell s}}(w), \quad \forall w \in \mathbb{R}^M,$$

the energy (3) becomes $\frac{w^\top B(t) w}{2}$ with $B(t)$ given by (7), and relation (5) becomes (2).

Standard Case

The 'standard case' of the general SD problem is as follows:

S.1	The space of the design vectors is the direct product of N of the spaces $\mathbf{S}^D$ of symmetric $D \times D$ matrices, so that the design vector is $t = (t_1, \ldots, t_N) : t_\ell \in \mathbf{S}^D$, $\ell = 1, \ldots, N$.
S.2	The set T of admissible designs is given by (6).
S.3	The mapping $t \mapsto B(t)$ is given by (7), and the matrix $\sum_{\ell=1}^N \sum_{s=1}^S b_{\ell_s} b_{\ell_s}^\top$ is positive definite;
S.4	The set W of kinematically admissible displacements is polyhedral: $$W = \{w \in \mathbf{R}^M : Pw \le p\},\ p \in \mathbf{R}^q, \tag{8}$$ and the system of constraints $Pw \le p$ satisfies the Slater condition: $\exists \overline{w} : P\overline{w} < p$.

The Set of Loads

In the traditional literature on structural design (see [1,2,3,4,5,7] and references therein) F is either a singleton $\{f\}$ ('single-load case'), or, more generally, a finite set:

$$F = \{f_1, \ldots, f_k\}. \tag{9}$$

Recently, a *robust* setting of the problem was proposed and motivated [6], where F is an ellipsoid:

$$F = \left\{ f = Qu :\ u \in \mathbb{R}^k,\ u^\top u \le 1 \right\}. \tag{10}$$

Semidefinite Reformulation of $\mathrm{P_{ini}}$

It turns out that the standard case of the multiload SD problem can be naturally posed as a *semidefinite program*; this is the case for the robust SD problem as well, provided that there are no obstacles ($W = \mathbf{R}^M$). In fact, there are three semidefinite settings of the standard SD problem: the *primal*(P), the *dual*(D) and the *equivalent primal* (P^+). The relations between these forms are as follows:

- is, basically, a straightforward semidefinite reformulation of ($\mathrm{P_{ini}}$);
- (D) is obtained from the *Fenchel dual* of (P) by analytic elimination of part of variables,
- (P^+) is obtained from the Fenchel dual of (D) by analytic elimination of part of variables.

The explicit semidefinite forms of the standard truss and shape design problems are as follows.

Multiload Truss Design

Here T, W, F are given by (6) (where $D = 1$), (8) and (9), respectively.

$$(\mathrm{P}) \quad \begin{cases} \tau \to \min \\ \begin{pmatrix} 2[\tau - p^\top y_i] & y_i^\top P - f_i^\top \\ P^\top y_i - f_i & \sum_{\ell=1}^N b_\ell b_\ell^\top t_\ell \end{pmatrix} \succeq 0, \\ \quad i = 1, \ldots, k; \\ \underline{\rho}_\ell \le t_\ell \le \overline{\rho}, \\ \quad \ell = 1, \ldots, N; \\ \sum_{\ell=1}^N t_\ell \le \nu; \\ y_i \ge 0, \quad i = 1, \ldots, k, \end{cases}$$

$$\left[\tau, t_\ell \in \mathbb{R},\ y_i \in \mathbb{R}^q\right],$$

$$\text{(D)}\quad \begin{cases} -2\sum_{i=1}^{k} f_i^\top w_i + \sum_{\ell=1}^{N}[\overline{\rho}_\ell \sigma_\ell^+ - \underline{\rho}_\ell \sigma_\ell^-] \\ \quad +\nu\gamma \to \min \\ \begin{pmatrix} \alpha_1 & & & \vdots & b_\ell^\top w_1 \\ & \ddots & & \vdots & \vdots \\ & & \alpha_k & \vdots & b_\ell^\top w_k \\ \cdots & \cdots & \cdots & \vdots & \cdots \\ b_\ell^\top w_1 & \cdots & b_\ell^\top w_k & \vdots & \gamma + \sigma_\ell^+ - \sigma_\ell^- \end{pmatrix} \\ \succeq 0, \quad \ell = 1, \ldots, N; \\ \sigma_\ell^\pm \geq 0, \quad \ell = 1, \ldots, N; \\ \gamma \geq 0; \\ Pw_i \leq \alpha_i p, \quad i = 1, \ldots, k; \\ 2\sum_{i=1}^{k} \alpha_i = 1 \end{cases}$$

$$\left[\alpha_i, \sigma_\ell^\pm, \gamma \in \mathbb{R},\ w_i \in \mathbb{R}^M\right]$$

$$\text{(P}^+\text{)}\quad \begin{cases} \tau \to \min \\ \begin{pmatrix} 2[\tau - p^\top y_i] & \vdots & q_1^i & \cdots & q_N^i \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ q_1^i & \vdots & t_1 & & \\ \vdots & \vdots & & \ddots & \\ q_N^i & \vdots & & & t_N \end{pmatrix} \succeq 0, \\ \quad i = 1, \ldots, k; \\ \underline{\rho}_\ell \leq t_\ell \leq \overline{\rho}_\ell, \\ \quad \ell = 1, \ldots, N; \\ \sum_{\ell=1}^{N} t_\ell \leq \nu; \\ y_i \geq 0, \quad i = 1, \ldots, k; \\ \sum_{\ell=1}^{N} q_\ell^i b_\ell = f_i - P^\top y_i, \\ \quad i = 1, \ldots, k, \end{cases}$$

$$\left[\tau, t_\ell, q_\ell^i \in \mathbb{R},\ y_i \in \mathbb{R}^q\right].$$

The simplest truss design problem (single load, no obstacles, trivial bounds on bar volumes: $\underline{\rho}_\ell = 0, \overline{\rho}_\ell = \nu$) can be reduced to a *linear programming program*. Indeed, in this case (P$^+$)is the program

$$\begin{cases} \frac{1}{2}\sum_\ell t_\ell^{-1} q_\ell^2 \to & \min \\ & t_\ell \geq 0, \\ & \sum_\ell t_\ell \leq \nu, \\ & \sum_\ell q_\ell b_\ell = f \end{cases}$$

$$\left[t_\ell, q_\ell \in \mathbb{R}, \quad \ell = 1, \ldots, N\right].$$

Partial minimization w.r.t. t_ℓ results in the program

$$\begin{cases} \frac{1}{2\nu}\left(\sum_\ell |q_\ell|\right)^2 \to & \min \\ \text{s.t.} & \sum_\ell q_\ell b_\ell = f, \end{cases}$$

which is, essentially, an LP program.

Robust Obstacle-Free Truss Design

Here F is the ellipsoid (10), T is given by (6) with $D = 1$, and $W = \mathbf{R}^M$.

$$\text{(P)}\quad \begin{cases} \tau \to \min \\ \begin{pmatrix} 2\tau I_k & Q^\top \\ Q & \sum_{\ell=1}^{N} b_\ell b_\ell^\top t_\ell \end{pmatrix} \succeq 0; \\ \underline{\rho}_\ell \leq t_\ell \leq \overline{\rho}, \\ \quad \ell = 1, \ldots, N; \\ \sum_{\ell=1}^{N} t_\ell \leq \nu, \end{cases}$$

$$[\tau, t_\ell \in \mathbb{R}]$$

(from now on, I_k is the $k \times k$ unit matrix),

$$
\text{(D)}\quad
\begin{cases}
-2\,\mathrm{Tr}(Q^\top w) + \sum_{\ell=1}^{N} [\overline{\rho}_\ell \sigma_\ell^+ - \underline{\rho}_\ell \sigma_\ell^-] + \nu\gamma \\
\to \min \\
\begin{pmatrix} \alpha & w^\top b_\ell \\ b_\ell^\top w & \gamma + \sigma_\ell^+ - \sigma_\ell^- \end{pmatrix} \succeq 0, \\
\quad \ell = 1, \ldots, N; \\
\sigma_\ell^\pm \geq 0, \quad \ell = 1, \ldots, N; \\
\gamma \geq 0; \\
2\,\mathrm{Tr}(\alpha) = 1
\end{cases}
$$

$$
\left[\alpha \in \mathbf{S}^k, \sigma_\ell^\pm, \gamma \in \mathbb{R}, w \in \mathbb{R}^{M\times k}\right],
$$

$$
\text{(P}^+\text{)}\quad
\begin{cases}
\tau \to \min \\
\begin{pmatrix} 2\tau I_k & \vdots & q_1^\top & \cdots & q_N^\top \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ q_1 & \vdots & t_1 & & \\ \vdots & \vdots & & \ddots & \\ q_N & \vdots & & & t_N \end{pmatrix} \succeq 0; \\
\underline{\rho}_\ell \leq t_\ell \leq \overline{\rho}_\ell, \\
\quad \ell = 1, \ldots, N; \\
\sum_{\ell=1}^{N} t_\ell \leq \nu; \\
\sum_{\ell=1}^{N} b_\ell q_\ell = Q
\end{cases}
$$

$$
\left[\tau \in \mathbb{R},\ t_\ell \in \mathbf{S}^D,\ q_\ell^\top \in \mathbb{R}^k\right]
$$

Multiload Shape Design

Here T, W, F are given by (6) (where $D = 3$ for planar and $D = 6$ for spatial shapes), (8), (9), respectively.

$$
\text{(P)}\quad
\begin{cases}
\tau \to \min \\
\begin{pmatrix} 2[\tau - p^\top y_i] & y_i^\top P - f_i^\top \\ P^\top y_i - f_i & \sum_{\ell=1}^{N}\sum_{s=1}^{S} b_{\ell s} t_\ell b_{\ell s}^\top \end{pmatrix} \succeq 0, \\
\quad i = 1, \ldots, k; \\
t_\ell \succeq 0, \quad \ell = 1, \ldots, N; \\
\underline{\rho}_\ell \leq \mathrm{Tr}(t_\ell) \leq \overline{\rho}, \quad \ell = 1, \ldots, N; \\
\sum_{\ell=1}^{N} \mathrm{Tr}(t_\ell) \leq \nu; \\
y_i \geq 0, \quad i = 1, \ldots, k
\end{cases}
$$

$$
\left[\tau \in \mathbb{R},\ t_\ell \in \mathbf{S}^D,\ y_i \in \mathbb{R}^q\right]
$$

$$
\text{(D)}\quad
\begin{cases}
-2\sum_{i=1}^{k} f_i^\top w_i + \sum_{\ell=1}^{N} [\overline{\rho}_\ell \sigma_\ell^+ - \underline{\rho}_\ell \sigma_\ell^-] + \nu\gamma \to \min \\
\begin{pmatrix} \alpha_1 I_S & & \vdots & B_\ell[w_1] \\ & \ddots & \vdots & \\ & & \vdots & \\ & \alpha_k I_S & \vdots & B_\ell[w_k] \\ \cdots & \cdots & \cdots & \cdots \\ B_\ell^\top[w_1] & \cdots & B_\ell^\top[w_k] & \vdots \quad (\gamma + \sigma_\ell^+ - \sigma_\ell^-) I_D \end{pmatrix} \\
\succeq 0, \quad \ell = 1, \ldots, N; \\
\sigma_\ell^\pm \geq 0, \quad \ell = 1, \ldots, N; \\
\gamma \geq 0; \\
Pw_i \leq \alpha_i p, \quad i = 1, \ldots, k; \\
2\sum_{i=1}^{k} \alpha_i = 1
\end{cases}
$$

$$
\left[\alpha_i, \sigma_\ell^\pm, \gamma \in \mathbb{R},\ w_i \in \mathbb{R}^M\right],
$$

where $\mathrm{B}_\ell[\mathrm{w}] = \begin{pmatrix} w^\top b_{\ell 1} \\ \cdots \\ w^\top b_{\ell S} \end{pmatrix}$,

$$
\text{(P}^+\text{)}\quad
\begin{cases}
\tau \to \min \\
\begin{pmatrix} 2[\tau - p^\top y_i] & \vdots & [\mathbf{q}_1^i]^\top & \cdots & [\mathbf{q}_N^i]^\top \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{q}_1^i & \vdots & \mathbf{t}_1 & & \\ \vdots & \vdots & & \ddots & \\ \mathbf{q}_N^i & & & & \mathbf{t}_N \end{pmatrix} \succeq 0, \\
\quad i = 1, \ldots, k; \\
\left[\mathbf{q}_\ell^i = \begin{pmatrix} q_{\ell 1}^i \\ \cdots \\ q_{\ell S}^i \end{pmatrix},\ \mathbf{t}_\ell = \begin{pmatrix} t_\ell & & \\ & \ddots & \\ & & t_\ell \end{pmatrix}\right] \\
\underline{\rho}_\ell \leq \mathrm{Tr}(t_\ell) \leq \overline{\rho}_\ell, \qquad l = 1, \ldots, N; \\
\sum_{\ell=1}^{N} \mathrm{Tr}(t_\ell) \leq \nu; \\
y_i \geq 0, \quad i = 1, \ldots, k; \\
\sum_{\ell=1}^{N}\sum_{s=1}^{S} b_{\ell s} q_{\ell s}^i = f_i - P y_i^\top, \qquad i = 1, \ldots, k;
\end{cases}
$$

$$
\left[\tau \in \mathbb{R},\ t_\ell \in \mathbf{S}^D,\ q_{\ell s}^i \in \mathbb{R}^D,\ y_i \in \mathbb{R}^q\right].
$$

Robust Obstacle-Free Shape Design

Here T is given by (6) (where $D = 3$ for planar and $D = 6$ for spatial shapes), F is the ellipsoid (10) and $W = \mathbf{R}^M$.

$$
\text{(P)}\quad
\begin{cases}
\tau \to \min \\
\begin{pmatrix} 2\tau I_k & Q^\top \\ Q & \sum\limits_{\ell=1}^{N}\sum\limits_{s=1}^{S} b_{\ell s} t_\ell b_{\ell s}^\top \end{pmatrix} \succeq 0; \\
t_\ell \succeq 0, \quad \ell = 1, \ldots, N; \\
\underline{\rho}_\ell \le \operatorname{Tr}(t_\ell) \le \overline{\rho}, \quad \ell = 1, \ldots, N; \\
\sum\limits_{\ell=1}^{N} \operatorname{Tr}(t_\ell) \le v,
\end{cases}
$$

$$
\left[\tau, t_\ell \in \mathbb{R}\right],
$$

$$
\text{(D)}\quad
\begin{cases}
-2\operatorname{Tr}(Q^\top w) + \sum\limits_{\ell=1}^{N} [\overline{\rho}_\ell \sigma_\ell^+ - \underline{\rho}_\ell \sigma_\ell^-] + v\gamma \to \min \\
\begin{pmatrix} \alpha & & \vdots & w^\top b_{\ell 1} \\ & \ddots & \vdots & \vdots \\ & & \alpha & \vdots \\ & & & w^\top b_{\ell S} \\ \cdots & \cdots & \cdots & \cdots \\ b_{\ell 1}^\top w & \cdots & b_{\ell S}^\top w & \vdots \quad (\gamma + \sigma_\ell^+ - \sigma_\ell^-) I_D \end{pmatrix} \succeq 0, \\
\quad \ell = 1, \ldots, N; \\
\sigma_\ell^\pm \ge 0, \quad \ell = 1, \ldots, N; \\
\gamma \ge 0; \\
2\operatorname{Tr}(\alpha) = 1
\end{cases}
$$

$$
\left[\alpha \in \mathbf{S}^k,\ \sigma_\ell^\pm,\ \gamma \in \mathbb{R},\ w \in \mathbb{R}^{M\times k}\right],
$$

$$
\text{(P}^+\text{)}\quad
\begin{cases}
\tau \to \min \\
\begin{pmatrix} 2\tau I_k & \vdots & \mathbf{q}_1^\top & \cdots & \mathbf{q}_N^\top \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{q}_1 & \vdots & \mathbf{t}_1 & & \\ \vdots & \vdots & & \ddots & \\ \mathbf{q}_N & \vdots & & & \mathbf{t}_N \end{pmatrix} \succeq 0; \\
\left[\mathbf{q}_\ell = \begin{pmatrix} q_{\ell 1} \\ \cdots \\ q_{\ell S} \end{pmatrix},\ \mathbf{t}_\ell = \begin{pmatrix} t_\ell & & \\ & \ddots & \\ & & t_\ell \end{pmatrix} \right] \\
\underline{\rho}_\ell \le \operatorname{Tr}(t_\ell) \le \overline{\rho}_\ell, \quad \ell = 1, \ldots, N; \\
\sum\limits_{\ell=1}^{N} \operatorname{Tr}(t_\ell) \le v; \\
\sum\limits_{\ell=1}^{N}\sum\limits_{s=1}^{S} b_{\ell s} q_{\ell s} = Q
\end{cases}
$$

$$
\left[\tau \in \mathbb{R},\ t_\ell \in \mathbf{S}^D,\ q_{\ell s} \in \mathbb{R}^{D\times k}\right].
$$

Relations between the original setting of the SD problem (P_{ini}) and its semidefinite forms (P), (D), (P^+) are summarized in the following statement:

Theorem 1 *Consider standard truss or shape case of problem (P_{ini}), and assume that the set of loads F is either (9), or (10); in the latter case, assume also that there are no obstacles, i. e., $W = \boldsymbol{R}^M$. Then*

i) A pair (τ, t) can be extended to a feasible solution to problems (P) and (P^+) if and only if $t \in T$ and $\mathrm{compl}_F(t) \le \tau$; consequently, the problems (P) and (P^+) are equivalent semidefinite reformulations of the problem of interest.

ii) All three programs (P), (D), (P^+) are strictly feasible (i. e., for each problem there exists a feasible solution satisfying strict forms of all inequality constraints) and solvable. The optimal values of (P) and (P^+) are equal to each other and to the optimal value in the problem of interest; the optimal value of (D) is minus the one of (P) and (P^+).

iii) For each program, every level set of its objective (i. e., the part of the feasible set where the objective is $\le$ a constant) is bounded.

Computational Issues

Semidefinite forms of the standard SD problem are well suited for solving by modern *polynomial time interior point methods*. The 'computational bottleneck' is huge sizes of SD problems of actual interest. The limitations imposed by this bottleneck heavily depend on which one of the forms (P), (D), (P^+) is solved, and in many cases (e. g., in truss design) (D) is by far better suited for numerical processing than other forms of the SD problem. For detailed discussion of computational issues, including 'computationally cheap' techniques for recovering a (nearly) optimal design $\{t_\ell\}_{\ell=1}^N$ from (nearly) optimal solution to (D), see [5,6].

See also

- ▶ Duality for Semidefinite Programming
- ▶ Interior Point Methods for Semidefinite Programming
- ▶ Semidefinite Programming and Determinant Maximization
- ▶ Semidefinite Programming: Optimality Conditions and Stability

- ▶ Semi-infinite Programming, Semidefinite Programming and Perfect Duality
- ▶ Solving Large Scale and Sparse Semidefinite Programs
- ▶ Structural Optimization
- ▶ Structural Optimization: History
- ▶ Topology of Global Optimization
- ▶ Topology Optimization

References

1. Bendsoe MP (1995) Optimization of structural topology shape and material. Springer, Berlin
2. Bendsoe MP, Ben-Tal A, Zowe J (1994) Optimization methods for truss geometry and topology design. Structural Optim 4:141–159
3. Bendsoe MP, Guedes JM, Haber RB, Pedersen P, Taylor JE (1994) An analytical model to predict optimal material properties in the context of Optimal Structural Design. J Appl Mechanics 61:930–937
4. Ben-Tal A, Bendsoe MP (1993) A new method for optimal truss topology design. SIAM J Optim 3:322–358
5. Ben-Tal A, Nemirovski A (1994) Potential reduction polynomial time method for truss topology design. SIAM J Optim 4:596–612
6. Ben-Tal A, Nemirovski A (1997) Stable truss topology design via semidefinite programming. SIAM J Optim 7:991–1016
7. Ringertz U (1993) On finding the optimal distribution of material properties. Structural Optim 5:265–267

Semi-infinite Programming and Applications in Finance

K. O. Kortanek[1], Vladimir G. Medvedev[2]

[1] University Iowa, Iowa City, USA

[2] Byelorussian State University, Minsk, Republic Belarus

MSC2000: 90C34, 91B28

Article Outline

Keywords
Problem Statement and Early Numerical Solution Methods
The Support Problems Method Developed in Belarus
Extended Support Problems Method
A SIP Approach for Estimating Uncertainty in Dynamical Systems
The Minimax Observation Problem Under Uncertainty with Perturbations
A Prototype: Analog of the Vasicek Model with Impulse Perturbations
Estimating the Spot Rate for Bonds with Constant Maturities
A Nonarbitrage Condition for LDSU
See also
References

Keywords

Duality equality; Optimality conditions; Supports problems method; Dynamical system; Nonstochastic uncertainty; Perturbations; Term structure of interest rates; Term to maturity

Problem Statement and Early Numerical Solution Methods

Consider the linear semi-infinite programming problem LSIP, where 's.t.' means 'subject to' throughout this paper:

$$\begin{cases} \nu_{\mathrm{P}} = & \max c^{\top} x \\ \text{s.t.} & f(x,t) \le 0, \\ & t \in T, \quad d_* \le x \le d^*, \end{cases} \tag{1}$$

where

$$f(x,t) = a^{\top}(t)x - b(t), \quad t \in T = [t_*, t^*],$$
$$x, c, d_*, d^* \in \mathbb{R}^n;$$
$$a(\cdot) \in \mathbb{R}^n, \quad b(\cdot) \in \mathbb{R},$$

are differentiable functions. Its *dual semi-infinite program* DLSIP is:

$$\begin{cases} \nu_{\mathrm{D}} = & \inf \int_T b(t)\, d\alpha(t) + \sum_{r=1}^{n} v_r^+ d_r^* + v_r^- d_{*r} \\ \text{s.t.} & \int_T a_r(t)\, d\alpha(t) + v_r^+ - v_r^- = c_r, \\ & r = 1, \ldots, n, \end{cases}$$

where α is a nonnegative *Riemann–Stieltjes measure* [13], and $v_r^+, v_r \ge 0$.

Examples in transportation theory and wavelet filter design are given, respectively, in [18,19]. In [4, Thm. 4] it is proved that $\nu_{\mathrm{P}} = \nu_{\mathrm{D}}$, but the infimum need not be attained. This equation is termed the *duality equality*.

Example 1

$$\max\left\{-x_1:\quad \begin{array}{rl} -tx_1 - t^2x_2 \le 0, & \forall t\in[0,1],\\ -1\le x_i\le 1, & i=1,2\end{array}\right\}.$$

Here the common program value is 0 but it is not attained in the dual.

In the 1970s however algorithms were developed under more regularity.

Definition 2 The linear system (1) satisfies the *Slater condition* if there exists $\overline{x}$, $d_* \le \overline{x} \le d^*$, for which it is satisfied strictly with $\overline{x}$.

With the Slater condition S.-Å. Gustafson [9] supplemented the necessary complementarity conditions (analogous to LP) with the now classical *matching of derivative conditions*, see also [11, (2.3)–(2.8)]. Rather than reviewing these we focus on recent extensions (appearing in Russian) of Gustafson's well known *three phase algorithm*, see [8,10]. [12, Fig. 28-1] presents the logic flow of semi-infinite programming algorithms appearing in 1973.

The Support Problems Method Developed in Belarus

The support problems method was suggested in [6] and further developed and algorithmically implemented in [24]. It is based on the principle of eliminating the subsets of the index set (T) where the constraints are violated. The first component consists of technical procedures for forming, solving, and analyzing sequences of LP problems having a small number of constraints which does not depend on a preassigned accuracy. The second component is a finishing procedure which roughly speaking employs a Newton procedure on a system of nonlinear necessary conditions (the matching of derivatives). However, the method is the most general available of the hybrid types [14] because it uses higher order derivatives.

We assume the set of feasible solutions of LSIP is nonempty, namely,

$$X=\left\{x\in\mathbb{R}^n:\quad \begin{array}{rl} f(x,t)\le 0, & \forall t\in T,\\ d_*\le x\le d^*\end{array}\right\}.$$

Define

$$J=\{1,\dots,n\},$$
$$f^{(s)}(x,t)=\frac{\partial f(x,t)}{\partial t^s},$$
$$a^{(s)}(t)=\left(a_j^{(s)}(t)=\frac{d^s a_j(t)}{dt^s},\quad j\in J\right),$$
$$N(q)=\begin{cases}\emptyset & \text{if } q<0,\\ \{0,\dots,q\} & \text{if } q\ge 0,\end{cases}$$

and let $p=p(x,t)\in\{0,1,\dots\}$ be a positive integer satisfying

$$\begin{aligned} &f^{(s)}(x,t)=0,\quad s\in N(p-1),\\ &f^{(p)}(x,t)\ne 0.\end{aligned}\tag{2}$$

From (2) it follows that

$$p=p(x,t)\in\begin{cases}\{0,2,\dots\}, & f^{(p)}(x,t)<0\\ & \text{if } t\notin\{t^*,t_*\},\\ \{0,1,\dots\}, & f^{(p)}(x,t)<0\\ & \text{if } t=t_*,\\ \{0,1,\dots\}, & f^{(p)}(x,t)<0\\ & \text{if } p \text{ is even}\\ & \text{and } t=t^*,\\ \{0,1,\dots\}, & f^{(p)}(x,t)>0\\ & \text{if } p \text{ is odd}\\ & \text{and } t=t^*,\end{cases}\tag{3}$$

Definition 3 The integer $q=q(t)$ is called the *motionless degree* (see [20]) at the point $t\in T$ in problem (1) if

$$f^{(s)}(x,t)=0,\quad s\in N(q(t)),\quad \forall x\in X,\tag{4}$$

and there exists at least one feasible solution $\widehat{x}\in X$ such that

$$f^{(q(t)+1)}(\widehat{x},t)\begin{cases}<0 & \text{if } t\ne t^* \text{ or}\\ & t=t^* \text{ and } p \text{ is even},\\ >0 & \text{if } t=t^* \text{ and } p \text{ is odd}.\end{cases}\tag{5}$$

It follows from Definition 3 that

$$q(t)\in\begin{cases}\{-1,1,3,5,\dots,p^*(t)-1\} & \text{if } t\in\,]t_*,t^*[\,,\\ \{-1,0,1,2,\dots,p^*(t)-1\} & \text{if } t\in\{t_*;t^*\}\end{cases}$$

where $p^*(t)=\max p(x,t)$, $x\in X$.

Remark 4 If in problem (1) there exists at least one point $t \in T$ so that $q(t) \geq 0$, then the Slater condition does not hold.

Following [20], let us describe the algorithm for constructing the function $q(t)$, $t \in T$. Define

$$T(x) = \{t \in T\colon\ f(x,t) = 0\} = \{t_i\colon\ i \in I\},$$
$$I = I(x) = \{1, \ldots, k = k(x)\}.$$

Set $q_i^{(0)} = -1$, $i \in I$; $k = 0$. Denote

$$I^{*(k)} = \left\{ i \in I\colon \begin{array}{c} t_i = t^*, \\ q_i^{(k)} + 1 \text{ an odd integer} \end{array} \right\},$$
$$I^{0(k)} = I \setminus I^{*(k)};$$
$$X_i^{(k)} = \left\{ z \in \mathbb{R}^k\colon \begin{array}{c} f^{(s)}(z, t_i) = 0, \\ s \in N(q_i^{(k)}), \\ f^{(q_i^{(k)}+1)}(z, t_i) \leq 0 \\ \text{if } i \in I^{0(k)}, \\ f^{(q_i^{(k)}+1)}(z, t_i) \geq 0 \\ \text{if } i \in I^{*(k)}, \\ d_* \leq z \leq d^* \end{array} \right\},$$
$$X^{(k)} = \bigcap_{i \in I} X_i^{(k)}.$$

For each $j \in I$ let $f_j^{(k)}(z) = f^{(q_i^{(k)}+1)}(z, t_j)$, and let $x^{(j)}$ be an optimal solution of the following LP problem:

$$x^{(j)} = \begin{cases} \operatorname{argmin}\limits_{z \in X^{(k)}} f_j^{(k)}(z) & \text{if } j \in I^{0(k)}, \\ \operatorname{argmax}\limits_{z \in X^{(k)}} f_j^{(k)}(z) & \text{if } j \in I^{*(k)}. \end{cases} \tag{6}$$

Since $x \in X^{(k)}$, it follows that an optimal solution of (6) always exists. Set $I^{(k)} = \{j \in I\colon f_j^{(k)}(x^{(j)}) = 0\}$. The following cases occur.

- $I^{(k)} \neq \emptyset$. Set

$$q_j^{(k+1)} = \begin{cases} q_j^{(k)}, & j \in I \setminus I^{(k)}, \\ q_j^{(k)} + 2, & t_j \notin \{t_*, t^*\}, \\ q_j^{(k)} + 1, & t_j \in \{t_*, t^*\}; \end{cases}$$

- $I^{(k)} = \emptyset$. Set

$$q(t) = \begin{cases} -1, & t \in T \setminus T(x) \\ q_j^{(k)}, & j \in I,\ t = t_j. \end{cases}$$

The function $q(\cdot)$ so defined satisfies conditions (1)–(5), but the details must be left to [20].

Let x be the feasible solution of the problem (1). Define

$$T(x) = \{t \in T\colon\ f(x,t) = 0\} = \{t_i\colon\ i \in I\},$$
$$I = I(x) = \{1, \ldots, \overline{k} = k(x)\},$$
$$p_i = p(x, t_i), \quad q_i = q(t_i), \quad i \in I,$$
$$I_0 = \{i \in I\colon\ q_i + 1 = p_i\}, I_* = I \setminus I_0,$$
$$I^* = \{i \in I\colon\ t_i = t^*,\ q_i + 1 \text{ odd}\}, I^0 = I \setminus I^*.$$

The following result is proved in [20].

Theorem 5 *The feasible solution $x \in X$ is the optimal solution of problem (1) if and only if the vector x is the optimal solution of the following LP problem:*

$$\begin{cases} \max & c^\top z \\ s.t. & d_* \leq z \leq d^*, \end{cases}$$

and

$$f^{(s)}(z, t_j) = 0, \quad s \in N(q_j), \quad j \in I, \tag{7}$$

$$f^{(q_j+1)}(z, t_j) \begin{cases} \leq 0 & \text{if } j \in I^0 \cap I_*, \\ \geq 0 & \text{if } j \in I^* \cap I_*. \end{cases} \tag{8}$$

Extended Support Problems Method

We conclude the algorithmic part of the paper with a very general description of the extension to problems not having interior point (non-Slater problems).

1 Determine a feasible solution $\hat{x}$ for problem (1).
In this step the LSIP problem with a floating number of variables is solved by the support problems method.
2 Determine motionless points $t_j \in T(\widehat{x})$ and the corresponding motionless degrees q_j, $j \in I(\widehat{x})$.
In this step one finds all points in the interval T for which $f(\widehat{x}, t) = 0$ and solves the sequence of the *LP* problems for defining the motionless degrees at these points.
3 Determine an optimal solution of (1).
In this step the constraints (7) − (8) are added to the constraints of the original problem (1). The resulting problem is solved by the support problems method.

Extended support problems method

We turn now to very recent applications of SIP to finance, which employ semi-infinite programs having index sets, T, appearing in a partitioned form, corresponding to units of time. A basic structure for LSIP is an index set-partitioned form having the following structure:

$$\text{(PLSIP)} \quad \begin{cases} \min & c^\top x \\ \text{s.t.} & b_*(t) \le A(t)x \le b^*(t), \\ & \quad \forall t \in T^L, \\ & T^L = \bigcup_{i=1}^{L} T_i = [t_{i-1}, \quad t_i], \\ & g_* \le Gx \le g^*, \\ & d_* \le x \le d^*, \end{cases} \tag{9}$$

where c, x, d^*, d_* are n-dimensional vectors, $b_*(\cdot)$, $b^*(\cdot)$ are m-dimensional functions, $A(\cdot)$ an $m \times n$ matrix function, and G an $p \times n$ matrix, with g^*, g^* p-dimensional vectors. We shall use the T^L notation throughout for various choices of the integer L. Typically, T_i is the ith day of an observational period.

A SIP Approach for Estimating Uncertainty in Dynamical Systems

Generally, the problem of estimation occurring in nondeterministic systems has been investigated by means of many stochastic models beginning with the papers of N. Wiener [28] and R.E. Kalman [16]. In the 1970s, nonstochastic observation models ('minimax', 'guaranteed') under uncertainty were developed in [5,21,22]. During the 1980s a new approach for optimization of linear dynamical systems under uncertainty was developed by R. Gabasov and F.M. Kirillova; see [7].

Our approach [17] to modeling uncertainty is in contrast to other qualitative approaches, for example, based on stochastic differential equations. In the latter case certain mathematical assumptions are made about the underlying stochastic processes which may be difficult to verify in real situations, for example, in the financial derivatives and assets markets. We demonstrate our approach by applying the following general *minimax observation problem*, stated with unknown parameters under nonstochastic uncertainty, to differential equations models for interest rates of shortest duration, termed the *spot interest rate*. The models we develop are analogous to some of the stochastic differential equations models appearing in the literature.

The Minimax Observation Problem Under Uncertainty with Perturbations

Our main model is the well known linear dynamic system under nonstochastic uncertainty with perturbations, LDSU, over the time interval, $\mathcal{T} = [0, T]$:

$$\begin{aligned} &\dot{x} = Ax + Dw(t), \\ &x(0) = x_0 \in X_0, \\ &X_0 = \left\{ x \in \mathbb{R}^n : \begin{array}{c} d_* \le x \le d^*; \\ g_* \le Gx \le g^* \end{array} \right\}, \\ &W(t) = \left\{ w(t) \in \mathbb{R}^l : \; w_* \le w(t) \le w^* \right\}, \\ &D \in \mathbb{R}^{n \times l}, \quad w_*, w^* \in \mathbb{R}^l. \end{aligned} \tag{10}$$

The fundamental matrix F of (10) has the following properties:

$$\begin{aligned} \dot{F} &= AF, \quad F(0) = E, \\ F(t-s) &= F(t-p)F(p-s), \\ F(t+s) &= F(t)F(s), \\ F^{-1}(t) &= F(-t), \end{aligned}$$

yielding the form of a solution of (10) by the *Cauchy formula*:

$$x(t) = F(t)x_0 + \int_0^t F(t)F(-s)Du(s)\, ds. \tag{11}$$

We assume that the state $x(t)$ of the system (10) is estimated from a sensor system of the form:

$$y(t) = h^\top x(t) + z(t), \quad \forall t \in \mathcal{T}, \tag{12}$$

which is a measurement system giving inexact information about current state of system (10), where $z(t)$ is an unknown piecewise continuous measurement error function.

Let (10), (12) generate a signal $y^*(t)$, $t \in \mathcal{T}$ with some measurement error $z^*(t)$, $t \in \mathcal{T}$. With our approach we seek $\widehat{x}(\cdot)$ by solving the following minimax observation problem:

$$\min_{(x, w(\cdot)) \in X_0 \times W(\cdot)} \max_{t \in \mathcal{T}} |z^*(t)|. \tag{13}$$

We obtain an equivalent infinite linear program from (13) by substituting (11) into (12) in order to obtain an

explicit expression for $z^*(\cdot)$:

$$z^*(t) = y^*(t) - h^\top F(t)x_0 - h^\top \int_0^t F(t)F(-s)Dw(s)\,ds.$$

Hence (13) is equivalent to:

$$\begin{cases} \min & v \\ \text{s.t.} & y^*(t) \le h^\top F(t)x + \int_0^t F(t)F(-s)Dw(s)\,ds + v, \\ & h^\top F(t)x + \int_0^t F(t)F(-s)Dw(s)\,ds - v \le y^*(t), \\ & x \in X_0, \quad v \ge 0, \\ & w_* \le w(t) \le w^*, \quad \forall t \in \mathcal{T}. \end{cases} \tag{14}$$

Our main application to state estimation is the following one. Suppose (x^0, w^0, v^0) is optimal for (14). Then

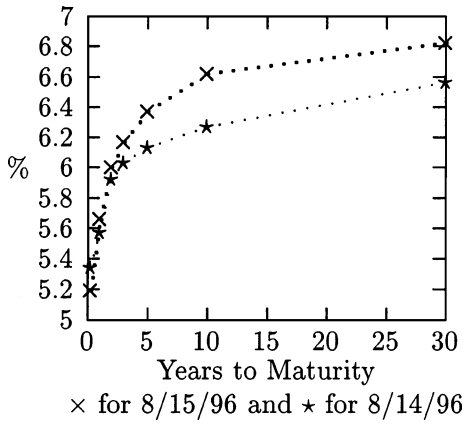
$$\widehat{x}(t) = F(t)x^0 + \int_0^t F(t)F(-s)Dw^0(s)\,ds$$

is an estimate of the state $x(t)$ of the system (10). This estimate gives the minimal possible maximum absolute value of the measurement error v^0. Special purpose algorithms for problems of this type have been developed in [24,25]. To implement this approach we must specify a class of impulse perturbations, and we illustrate one such prototype class next

A Prototype: Analog of the Vasicek Model with Impulse Perturbations

The financial markets setting shall be that of a default-free discount bond paying \$ 1 at maturity time T. At this point for convenience, we take the inception to be 0, while letting $P(t, T)$ denote the price of this bond at time t, $0 \le t \le T$. In actuality the bond may be a 91 day treasury bill issued at 10/1/97 ($= t_0$) maturing on 12/31/97 = t_0 + 91days. By definition, $P(T, T) = 1$. For $t \le T$, the *yield to maturity* $R(t, T)$ prevailing at time t is the internal rate of return at time t on a bond with maturity date T:

$$R(t, T) = -\frac{1}{T-t}\log P(t, T), \quad 0 \le t \le T. \tag{15}$$



Semi-infinite Programming and Applications in Finance, Figure 1
US treasury yield curve: Constant maturities

The *interest rate yield curve* is the plot of R against the time to maturity, see [2,15,29]. For $T \ge t$, as a function of T, $R(t, T)$ is usually called the *term structure of interest rates* at time t. Figure 1 is an illustration from recent data.

The instantaneous short rate or spot rate prevailing at time t, $r(t)$, (see, [26]) is defined as:

$$r(t) = \lim_{T \to t} R(t, T). \tag{16}$$

Hence,

$$P(t, T) = e^{-\int_t^T r(s)\,ds}. \tag{17}$$

The spot rate cannot be observed from real data, but it is the focus of various stochastic differential equations models. Our prototype LDSU model shall also be constructed around the spot rate, r. It follows from (15), (17) that

$$R(t, T) = \frac{1}{T-t}\int_t^T r(s)\,ds. \tag{18}$$

We illustrate the approach with the classical model of O. Vasicek [26], where the standard Brownian motion, Z, underlies the stochastic differential equation for the spot rate, and where α, β, and σ are parameters, see [1,3,27]:

$$dr = (\alpha + \beta r)\,dt + \sigma\,dZ. \tag{19}$$

The parameters are employed to capture shifts and volatility of the spot rate, [3]. We hypothesize that the

spot rate is governed by the following linear dynamic system with unknown parameters and nonstochastic uncertainty. Let N be a positive integer,

$$\begin{aligned} &\dot{r} = \alpha + \beta r(t) + w(t), \\ &\beta \neq 0, \quad r(t_0) = r_0, \quad t \in T^N, \end{aligned} \tag{20}$$

where T^N is defined in (9), $L = N$. Assume that a priori information about the unknown parameters, $\omega(\cdot)$, of this LDEU takes the form:

$$\begin{aligned} \omega &= (r_0, \alpha, \beta, w_i, \quad i = 1, \ldots, N), \\ \omega_* &= (r_*, \alpha_*, \beta_*, w_{*i}, \quad i = 1, \ldots, N), \\ \omega^* &= (r^*, \alpha^*, \beta^*, w_i^*, \quad i = 1, \ldots, N), \\ \Omega &= \left\{\omega \in \mathbb{R}^{N+3} \colon \omega_* \leq \omega \leq \omega^*\right\}, \\ w(t) &= w_i, \quad w_{*i} \leq w_i \leq w_i^*, \\ &\qquad t \in T_i, \quad i = 1, \ldots, N, \end{aligned} \tag{21}$$

where $w(\cdot)$, are *piecewise-constant perturbations*. Other choices of classes of perturbations appear in [17]. System (20)–(21) comprise our prototype of (10).

Remark 6 We stress the dependence of the spot rate on the parameters in (21) by writing $r(\cdot|\omega)$. However, to emphasize the status of ω as an independent variable we write $r(t|\omega) = f(t, \omega)$.

We address next how the general measurement system (12) specializes in our Vasicek-based prototype.

Estimating the Spot Rate for Bonds with Constant Maturities

We define the period of observation to be T^M, for a positive integer, M, see (9). Let τ be the current time-to-maturity (maturity term), so that with inception date t a τ-maturity bond becomes due at date $t + \tau$.

Assume that we have observed values of the treasury yield curve giving a series of yields to maturity $R_i^{(\tau)}$, $i = 1, \ldots, M$, for some given maturity term τ during M days of observation. The date t_M is the last day of the observation period, becoming the current date. Under this interpretation we build a piecewise constant form of the yield to maturity, for numerical stability reasons, and consistent with LDSU:

$$\begin{aligned} &\widehat{R}(t, t+\tau) = R_i^{(\tau)}, \\ &t \in T_i, \quad i = 1, \ldots, M. \end{aligned} \tag{22}$$

The table below illustrates some observations of US treasury yield curve rates, for successive years.

Date	3-mo		6-mo	9-mo	1-yr
01/03/94	3.16	$\leftarrow 1$	3.39	3.67	4.66
.	.		.	.	.
.	.		.	.	.
01/18/96	5.11		5.02	5.01	
01/19/96	5.1		5.06	5.02	5.03

Observations of treasury yield curves.
Legend: $\leftarrow 1 : R_i^{(\tau)} = R_{01/03/94}^{(3\text{ mo})}$

Definition 7 By the *Ω-based yield* we mean the averaged integral

$$\begin{aligned} &p(t, \omega|\tau) = \frac{1}{\tau}\int_t^{t+\tau} r(s|\omega)\, ds, \quad t \in T^M, \\ &\omega \in \Omega, \quad \text{the set of unknown parameters.} \end{aligned} \tag{23}$$

The *estimation error*, $\varepsilon(t, \omega)$ is the difference,

$$\begin{aligned} &p(t, \omega|\tau) - \widehat{R}(t, t+\tau), \\ &t \in T^M, \quad \omega \in \Omega. \end{aligned} \tag{24}$$

Note that (24) corresponds to the measurement error function in (10). We compute the estimate ω^0 of unknown parameters ω by minimizing over $\omega \in \Omega$, the maximum absolute value of the function of estimation errors $\varepsilon(t, \omega)$ on the interval T^M. This leads to the following problem:

$$\min_{\omega \in \Omega} \max_{t \in T^M} |\varepsilon(t, \omega)|. \tag{25}$$

Problem (25) may be written as the following nonlinear semi-infinite programming NSIP problem, see [14]:

$$\begin{cases} \min & \nu_\tau \\ \text{s.t.} & p(t, \omega|\tau) - \nu_\tau \leq \widehat{R}(t, t+\tau), \\ & \widehat{R}(t, t+\tau) \leq p(t, \omega|\tau) + \nu_\tau, \\ & t \in T^M, \quad \omega \in \Omega, \quad \nu_\tau \geq 0. \end{cases} \tag{26}$$

We shall call problem (26) the *τ-programmed problem of spot rate estimation*, but we must be more specific. We return to our basic model (20)–(21) and apply the Cauchy formula to find that the solution of the dif-

ferential equation (20) has the form

$$
\begin{aligned}
r(t|\omega) &= e^{\beta t} r_0 + \frac{\alpha}{\beta}(e^{\beta t} - 1) \\
&+ \sum_{j=1}^{i-1} w_j \frac{e^{\beta t}}{\beta}(e^{-\beta t_{j-1}} - e^{-\beta t_j}) \\
&+ w_i \frac{1}{\beta}(e^{\beta(t-t_{i-1})} - 1), \\
&t \in T_i, \quad i = 1, \ldots, N, \quad N = M + \tau.
\end{aligned} \tag{27}
$$

To derive the explicit form of problem (26) means that we must first specify the function

$$
\begin{aligned}
&p(t, \omega|\tau) = \frac{1}{\tau}\int_t^{\tau} r(s|\omega)\, ds, \\
&t \in T_i, \quad i = 1, \ldots, M.
\end{aligned}
$$

After a tedious series of integrations we obtain

$$
\begin{aligned}
p(t, \omega|\tau) = &\frac{e^{\beta(t+\tau)} - e^{\beta t}}{\tau\beta} r_0 \\
&+ \left(\frac{e^{\beta(t+\tau)} - e^{\beta t}}{\tau\beta^2} - \frac{1}{\beta}\right)\alpha \\
&+ \sum_{k=1}^{i+\tau} a_k^i(\beta, t|\tau) w_k,
\end{aligned}
$$

where $a_k^i(\beta, t|\tau) =$

$$
\begin{cases}
\dfrac{e^{\beta(t-t_{k-1}+\tau)} - e^{\beta(t-t_k+\tau)} + e^{\beta(t-t_k)} - e^{\beta(t-t_{k-1})}}{\tau\beta^2}, \\
\qquad k < i, \\
\dfrac{e^{\beta(t-t_{k-1}+\tau)} - e^{\beta(t-t_k+\tau)} - e^{\beta(t-t_{k-1})} + 1}{\tau\beta^2} - \dfrac{t_k - t}{\tau\beta}, \\
\qquad k = i, \\
\dfrac{e^{\beta(t-t_{k-1}+\tau)} - e^{\beta(t-t_k+\tau)}}{\tau\beta^2} - \dfrac{t_k - t_{k-1}}{\tau\beta}, \\
\qquad i < k < i + \tau, \\
\dfrac{e^{\beta(t-t_{k-1}+\tau)} - 1}{\tau\beta^2} - \dfrac{t - t_{k-1} + \tau}{\tau\beta}, \\
\qquad k = i + \tau.
\end{cases} \tag{28}
$$

From the above we obtain the following nonlinear semi-infinite programming problem

$$
\begin{cases}
\min \quad v_\tau \\
\text{s.t.} \quad \dfrac{e^{\beta(t+\tau)} - e^{\beta t}}{\tau\beta} r_0 \\
\qquad + \left(\dfrac{e^{\beta(t+\tau)} - e^{\beta t}}{\tau\beta^2} - \dfrac{1}{\beta}\right)\alpha \\
\qquad + \displaystyle\sum_{k=1}^{i+\tau} a_k^i(\beta, t|\tau) w_k - v_\tau \\
\qquad \le \widehat{R}(t, t+\tau), \\
\hat{R}(t, t+\tau) \\
\qquad \le \dfrac{e^{\beta(t+\tau)} - e^{\beta t}}{\tau\beta} r_0 \\
\qquad + \left(\dfrac{e^{\beta(t+\tau)} - e^{\beta t}}{\tau\beta^2} - \dfrac{1}{\beta}\right)\alpha \\
\qquad + \displaystyle\sum_{k=1}^{i+\tau} a_k^i(\beta, t|\tau) w_k + v_\tau,
\end{cases} \tag{29}
$$

where

$$
\begin{aligned}
&t \in T_i, \quad i = 1, \ldots, M; \\
&\alpha_* \le \alpha \le \alpha^*, \quad \beta_* \le \beta \le \beta^*, \quad r_* \le r_0 \le r^*, \\
&v \ge 0, \quad w_* \le w_i \le w^*, \quad i = 1, \ldots, N.
\end{aligned}
$$

Let $(\omega^0, v^{\tau 0})$ be the solution of problem (29), (26). Upon substituting ω^0 for ω in (27) yields a formula that we term the *τ-estimate of the spot rate*, i. e.,

$$
r_\tau(t) = f(t, \omega^0), \quad t \in T^M. \tag{30}
$$

It follows from (17) that the function

$$
\begin{aligned}
&P(t, \widehat{t}) = e^{-\int_t^{\widehat{t}} r_\tau(s)\, ds}, \\
&t \in [t_0, t_N], \quad \widehat{t} \in [t, t_N],
\end{aligned} \tag{31}
$$

will be an estimate of the price at time t of a discount bond maturing at time $\widehat{t}$.

The function $r_\tau(t) = f(t|\omega^0)$, while an estimate in the interval $[t_0, t_M]$, becomes the forecast in the future interval (the time after the current time t_M), namely, the forecast interval $[t_M, t_N]$, where $N = M + \tau$. If the function $r(t, |\omega^0)$ is defined over $t \in [t_N, t_{M^{\mathcal{P}}}]$, where $M + \tau < M^{\mathcal{P}}$, then the predicted price of a discount bond maturing at time $\widehat{t}$ is given by:

$$
\begin{aligned}
&\widetilde{P}(t, \widehat{t}) = e^{-\int_t^{\widehat{t}} r_\tau(s)\, ds}, \\
&t \in [t_N, t_{M^{\mathcal{P}}}], \quad \widehat{t} \in [t, t_{M^{\mathcal{P}}}],
\end{aligned} \tag{32}
$$

and the predicted yield is

$$\widetilde{R}(t,\widehat{t}) = \frac{1}{\widehat{t}-t}\int_t^{\widehat{t}} r_\tau(\widehat{t})\,ds, \quad t \in [t_N, t_{M^p}], \quad s \in [t, t_{M^p}] \tag{33}$$

at time t of discount bond maturing at date $\widehat{t}$.

A Nonarbitrage Condition for LDSU

Analogous to the stochastic case, we construct a portfolio with two bonds with differing maturities, T_1 and T_2, selling one unit of the T_1 maturity and buying Δ of the T_2 maturity. The value of the portfolio is (see [29, Sect. 17.5]),

$$\Pi(t) = P(t, T_1) - \Delta P(t, T_2). \tag{34}$$

Analogously, we differentiate $\Pi(t)$ with respect to time, t, recognizing that all of our parameters are independent of time, (see also Remark 6. Hence we obtain

$$\frac{d\Pi(t)}{dt} = \Pi(t) f(t, \omega). \tag{35}$$

But (35) states that the return on the portfolio equals the risk-free rate, the spot rate, [29, p. 271]. But this required condition is not our complete measure of nonarbitrage because our approach depends on actual observations and real data. Let us be more precise.

If at time T_1 \$1 is invested in the risk-free market (what the observations of actual data show) and grows to \$$M$ at time T_2, then \$$M$ must be compared with what the estimated spot rate returns over this period, where we assume $f(s, \omega) \geq 0$ for all ω, i. e.,

$$e^{\int_{T_1}^{T_2} f(s,\omega)\mathrm{d}s}.$$

If $M > e^{\int_{T_1}^{T_2} f(s,\omega)\mathrm{d}s}$, we borrow \$1 at T_1 at the spot rate and invest it in the risk-free market during the period $[T_1, T_2]$. At T_1 we make a profit of $M - e^{\int_{T_1}^{T_2} f(s,\omega)\mathrm{d}s}$. If $M < e^{\int_{T_1}^{T_2} f(s,\omega)\mathrm{d}s}$, we borrow \$1 in the risk-free market (supported by observed data) and invest it at the spot rate over the period $[T_1, T_2]$. We make a profit of $e^{\int_{T_1}^{T_2} f(s,\omega)\mathrm{d}s} - M$. This analysis motivates the following condition.

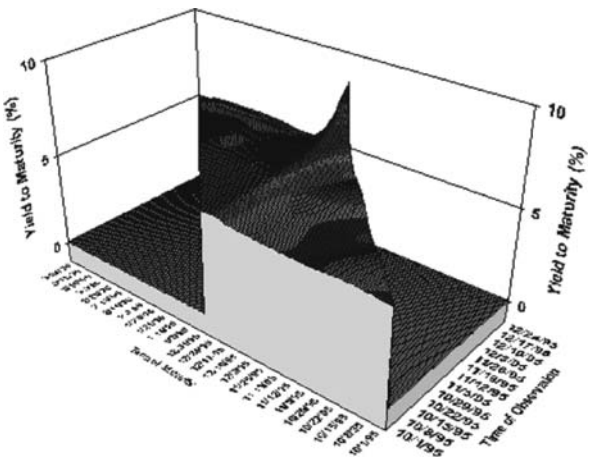
Assumption 8 Let $f(\cdot, \cdot)$ be a nonnegative function specified as in Remark 6. For any arbitrary T_1, T_2, and W satisfying $T_1 < T_2$, $W > 0$, there exists a $\omega^* \in \Omega$ such that:

$$\int_{T_1}^{T_2} f(s, \omega^*)\,ds = W. \tag{36}$$

This is a necessary condition to guarantee existence of nonarbitrage in the broader sense of using real observational data for estimation by the rules and models we have introduced.

We conclude by presenting one of the figures obtained from a solution of problem (26), (29) with piecewise-constant perturbations for maturity term τ = 91 days over the last three months of 1995. Using the notation of (29) Fig. 2 is a three-dimensional plot of $\widetilde{R}(t, t+\tau) \equiv p(t, \omega^0|\tau)$ having two boundaries both with slope 1 that have financial interpretations:

- the left-most boundary is $\widetilde{R}(t, t + 91$ days), 10/1/95 $\leq t \leq$ 12/24/95, depicting the estimated yield curve over the observation period;
- the right-most boundary is $\widetilde{R}(t, t)$, 10/1/95 $\leq t \leq$ 12/24/95, and depicts the unobservable estimated spot rate over the observation period.



Semi-infinite Programming and Applications in Finance, Figure 2
Estimate of the term structure of interest rates using the analog of the Vasicek model with piecewise-constant perturbations for observations of yield to maturity, τ = 3 mo, made during the observation period 1/8/1995 to 31/12/1995

It is interesting to note that in the stochastic case the market price of risk cannot be estimated uniquely in many cases. This result is demonstrated in [23].

See also

- Competitive Ratio for Portfolio Management
- Financial Applications of Multicriteria Analysis
- Financial Optimization
- Portfolio Selection and Multicriteria Analysis
- Robust Optimization
- Semi-infinite Programming: Approximation Methods
- Semi-infinite Programming and Control Problems
- Semi-infinite Programming: Discretization Methods
- Semi-infinite Programming: Methods for Linear Problems
- Semi-infinite Programming: Numerical Methods
- Semi-infinite Programming: Second Order Optimality Conditions
- Semi-infinite Programming, Semidefinite Programming and Perfect Duality

References

1. Back K (1996) Yield curve models: A mathematical review. In: Ledermann J, Klein R, Nelkin I (eds) Option Embedded Bonds. Irwin Publ, Toronto
2. Baxter M, Rennie A (1997) Financial calculus, an introduction to derivative pricing. Cambridge Univ Press, Cambridge
3. Chan KC, Karolyi GA, Longstaff FA, Sanders AB (1992) An empirical comparison of alternative models of the short-term interest rate. J Finance 47:1209–1227
4. Charnes A, Cooper WW, Kortanek KO (1965) On representation of semi-infinite programs which have no duality gaps. Managem Sci 12:113–121
5. Chernousko FL (1989) The estimating of state for dynamic systems. Nauka, Moscow
6. Gabasov R, Kirillova FM, Kostyukova OI (1984) Constructive methods of optimization. Part 2. Control problems. Univ Press Belarus, Minsk
7. Gabasov R, Kirillova FM, Prischepova S Optimal feedback control. no. 207 in Lecture Notes Economics and Inform. Systems. Springer, Berlin
8. Glashoff K, Gustafson S-Å (1983) Linear optimization and approximation. Appl Math Sci, vol 45. Springer, Berlin
9. Gustafson S-Å (1970) On the computational solution of a class of generalized moment problems. SIAM J Numer Anal 7:343–357
10. Gustafson S-Å (1983) A three phase algorithm for semi-infinite programs. In: Fiacco AV, Kortanek KO (eds) Semi-Infinite Programming and Applications. In: Lecture Notes Economics and Math Systems, no 215. Springer, Berlin
11. Gustafson S-Å, Kortanek KO (1973) Numerical treatment of a class of semi-infinite programming problems. Naval Res Logist Quart 20:477–504
12. Gustafson S-Å, Kortanek KO (1992) Semi-infinite programming-recent trends of development. In: Phillips FY, Rousseau JJ (eds) Systems and Management Sci by Extremal Methods Research Honoring Abraham Charnes at Age 70. Kluwer, Dordrecht, pp 463–478
13. Gustafson S-Å, Kortanek KO, Rom WO (1970) Non-Chebysevian moment problems. SIAM J Numer Anal 7:335–342
14. Hettich R, Kortanek KO (1993) Semi-infinite programming: Theory, methods, and applications. SIAM Rev 35:380–429
15. Hull JC (1997) Options, futures, and other derivative securities, 3rd edn. Prentice-Hall, Englewood Cliffs
16. Kalman RE (1960) A new approach to linear filtering and prediction problems. J Basic Engineering 821:34–45
17. Kortanek KO, Medvedev VG (1999) Models for estimating the structure of interest rates from observations of yield curves. In: Avellaneda M (ed) Quantitative Analysis in Financial Markets. World Sci, Singapore, pp 53–120
18. Kortanek KO, Moulin P (1998) Semi-infinite programming in orthogonal wavelet filter design. In: Reemtsen R and RÜckmann J-J (eds) Semi-infinite Programming. Nonconvex Optim Appl. Kluwer, Dordrecht, pp 323–357
19. Kortanek KO, Yamasaki M (1982) Semi-infinite transportation problems. J Math Anal Appl 88:555–565
20. Kostyukova OI (Sept. 1995) Investigation of the linear extremal problem with continuum constraints. Prepr Inst Math Acad Sci BSSR 26(336)
21. Krasovsky NN (1976) Theory of control with movement. Nauka, Moscow
22. Kurzansky AB (1977) Control and observation in indefiniteness conditions. Nauka, Moscow
23. Medvedev G, Cox SH (1996) The market price of risk for affine interest rate term structures, Proc 6th Internat AFIR-Colloq Aktuarielle Ansätze für Finanz-Risken AFIR 1996 (Nuremberg), vol 1. VVW Karsruhe, Karlsruhe, pp 913–924
24. Medvedev VG (1994) Optimal observations of initial state and input disturbances for dynamic systems. SAMS 14:275–288
25. Medvedev VG (1994) Positional algorithm for optimal observations of linear dynamical systems. SAMS 16:93–111
26. Vasicek O (1977) An equilibrium characterization of the term structure. J Financial Economics 5:177–188
27. Vetzal KR (1994) A survey of stochastic continuous time models of the term structure of interest rates. Insurance: Math and Economics 14:139–161
28. Wiener N (1949) The extrapolation, interpolation, and smoothing of stationary Time series. Wiley, New York
29. Wilmot P, Howson S, Dewyne J (1997) The mathematics of financial derivatives, a student introduction. Cambridge Univ Press, Cambridge

Semi-infinite Programming: Approximation Methods

Sven-Åke Gustafson
Stavanger University, Stavanger, Norway

MSC2000: 90C34

Article Outline

Keywords
Nonlinear Semi-Infinite Programs
Computationally Equivalent Semi-Infinite Programs
Approximation in the Uniform Norm
See also
References

Keywords

Nonlinear semi-infinite programs; Computational equivalent; Index set; Nonnegative interpolatory operator

The class of general semi-infinite programming problems may be looked upon as a generalization of the class of optimization problems with finitely many variables and constraints since in a semi-infinite program, either the number of variables or the number of constraints (but not both at the same time) may be infinite. In the present paper we will mainly deal with semi-infinite programs with finitely many variables. The main theoretical as well as practical difficulty is that one needs to verify that a proposed optimal solution satisfies infinitely many inequality constraints. In addition, nonlinear problems may have many local minima and one needs to verify that a calculated optimum is indeed global. We will discuss the concept of *computational equivalence*, i. e. that for a fixed computer and software one may construct an optimization problem with finitely many variables and constraints which has the same computer representation as the semi-infinite program whose solution is sought. We will deal in detail with the class of linear problems and give some numerical examples, illustrating computational equivalence. We will consider one-sided approximation and approximation in the maximum norm. For the application of global optimization to linear semi-infinite programming and an illustration on the air quality control problem, see ► Semi-infinite Programming: Methods for Linear Problems. The literature on semi-infinite programs and their applications to problems in science and engineering is extensive. For a general introduction to this field, see [8] [10] and [14]. Recent results are to be found, for instance, in [16].

Nonlinear Semi-Infinite Programs

We will study the following class of problems:

Definition 1 Let S be a fixed set, F:$\mathbf{R}^n \to \mathbf{R}$ and G:$\mathbf{R}^n \times S \to \mathbf{R}$ two fixed functions. Then the following task will be called a *semi-infinite program*:

$$\min F(y) \tag{1}$$

over all $y \in \mathbf{R}^n$ subject to the constraint

$$G(y, s) \geq 0, \quad s \in S. \tag{2}$$

Remark 2 The data of the semi-infinite program (1), (2) are hence the index set S, the real-valued functions F defined on $\mathbf{R}^n$ and the real-valued function G, defined on $\mathbf{R}^n \times S$.

Definition 3 Use the notation of Definition 1. Put

$$Y = \{y \in \mathbb{R}^n : \ G(y, s) \geq 0\}, \tag{3}$$

$$V = \inf_{y \in Y} F(y). \tag{4}$$

If Y is empty, then we define $V = -\infty$, i. e. the condition (2) is inconsistent.

We observe that (1) and (2) define a very general class of problems. If we restrict S to be finite, then we get nonlinear optimization problems with finitely many nonlinear constraints. If F and G are linear with respect to y then we get linear programs if S is finite, otherwise linear semi-infinite programs. If we require Y, the set of feasible solutions to be compact and F to be continuous on Y, then the existence of optimal solutions is guaranteed. We need to impose further assumptions in order to show that a proposed computational scheme is efficient. When a linear semi-infinite program is to be solved, a common approach is to discretize the problem, i. e. replace the index set S by a finite subset T and

then solve the linear program hereby arising numerically, which always can be done with a finite number of arithmetic operations. For nonlinear problems, the numerical solution of the discretized problem is still difficult, since it may have many local extrema and the verification that a calculated extremum is the global optimum sought is nontrivial. For this reason discretization has not been the main approach for the treatment of nonlinear semi-infinite programs. Instead one has sought binding constraints, worked with penalty methods and settled for calculation of a local optimum. See [3,4,6,7,13,14,15] and [16]. Quasirandom methods have also been used for estimating the global optimum. See, e. g., [5].

Computationally Equivalent Semi-Infinite Programs

As known, a computer may only store a finite number of symbols. Symbol-manipulating languages like Maple and Mathematica may perform relatively complicated operations exactly by means of formula manipulations but their capacity is limited to a certain extent. This is illustrated by their treatment of operations on rational numbers which may be represented exactly as pairs of integers. However, if one would try to solve a linear systems of equations with rational coefficients exactly, then one would find that the available storage capacity is exceeded already for relatively small systems. We assume from now on that we have a language like Fortran which may manipulate integers exactly, provided that their magnitude is limited by a bound which is known but depends on the computer and the software used. Arithmetic operations on real numbers may be performed with high accuracy, but not exactly and bounds for the errors may be derived. Since the storage of the computer is limited the set $\mathbf{R}$ of real numbers must be represented by a finite subset, the computer numbers. Therefore the set $\mathbf{R}$ may be split into a finite number of subsets whose elements have identical representations in the computer. Two reals, which have the same computer representations are considered computationally equivalent, since it is not possible to distinguish between them by means of computational operations. As a consequence of this the index set S in (2) must be represented by a finite subset T of computer numbers. We introduce

Definition 3 The program

$$\min F^*(y) \tag{5}$$

over all $y \in \mathbf{R}^n$ subject to the constraint

$$G^*(y, s) \geq 0, \quad s \in S, \tag{6}$$

is said to be *computational equivalence* to the program (1), (2) if there is a compact set $Y^* \subset \mathbf{R}^n$ which contains all the feasible solutions of both programs such that:

- $F^*(y)$ and $F(y)$ are computationally equivalent on Y^*;
- $G^*(y, s)$ and $G(y, s)$ are computationally equivalent on $Y^* \times S$.

Remark 5 The requirement that the set of feasible solutions should be compact may seem awkward, since a semi-infinite program may have an unbounded feasible set even if the set of optimal solutions is bounded. Therefore one considers regularized semi-infinite programs which have the constraint $\|y\| \leq M$ where, the positive number M is chosen so large that this constraint is not binding for at least one optimal solution.

The next issue is to construct an optimization problem with finitely many constraints whose set of feasible solutions is computationally equivalent to that of (1), (2). We will outline the procedure for doing this which is presented in [1,11,12]. We next introduce

Definition 6 Use the notations and definitions of (1) and (2). Let $T \in S$ be a finite set with N points:

$$T = \{t_1, \ldots, t_N\}.$$

Let further $w_1, \ldots, w_N$ be N continuous functions defined on S and having the properties:

$$\begin{cases} w_j(s) \geq 0, & s \in S,\ j = 1, \ldots, N, \\ \sum_{j=1}^{N} w_j(s) = 1, & s \in S, \\ w_j(s_i) = \delta_{ij}, & i = 1, \ldots, N,\ j = 1, \ldots, N. \end{cases} \tag{7}$$

Let f be a continuous mapping defined on S and such that $f(s)$ is either a real number or a vector in $\mathbf{R}^n$. We now define the nonnegative *interpolatory operator* based on $T \in S$ by

$$(Lf)(s) = \sum_{j=1}^{N} w_j(s) f(t_j).$$

Remark 7 We note that L is a linear operator having f as argument. Also note that $(Lf)(s) = f(s)$ for all $s \in T$.

Remark 8 In the case of linear semi-infinite programming one may prove that if we replace the functions defined on S with the outputs of the mapping L then this semi-infinite program has the same feasible set as the linear program obtained by replacing the index set S by the grid T used for defining L. See, e. g., [8] or [12].

Example 9 We consider the simple linear semi-infinite program:

$$\min y_1 + \frac{y_2}{\pi},$$

subject to the constraints

$$y_1 + y_2 s \geq \sqrt{1+s}, \quad 0 \leq s \leq 1.$$

One finds directly that the optimal y_1, y_2 are defined by the condition that the straight $y_1 + y_2 s$ should be the tangent to the curve $\sqrt{1+s}$ at the point $s = 1/\pi$. Assume that we use a computer with working relative accuracy $1.0 \cdot 10^{-8}$. We define here L as the result of equidistant linear interpolation with stepsize h. We notice that the functions $a_1(s) = 1$ and $a_2(s) = s$ are interpolated exactly and one may verify that the maximal relative interpolation error for $\sqrt{1+s}$ is given by $h^2/32$. Therefore, if we take $h = 5 \cdot 10^{-4}$, the discretized linear program becomes computationally equivalent with the original one. In this case T has 2001 elements.

Approximation in the Uniform Norm

We use the same definitions as in (1) and (2). Let f be a function which is continuous on S. We define its maximum norm:

$$\|f\| = \max_{s \in S} |f(s)| .$$

Next we consider the problem to determine

$$\min_{y \in \mathbb{R}^n} \max_{s \in S} |G(y, s)| .$$

An equivalent formulation of this task is given by the semi-infinite program:

$$\min y_0$$

subject to the constraints

$$G(y, s) + y_0 \geq 0, \quad -G(y, s) + y_0 \geq 0, \quad s \in S.$$

In this last problem, the variables are the real y_0 and the vector $y \in \mathbf{R}^n$. At each point $s \in S$ two inequality conditions are to be satisfied. To calculate the optimal value corresponding to a certain y one needs to perform a global optimization over S. We note that the problem is consistent. If we discretize the problem, replacing S by a finite subset T this may be interpreted as approximating over T.

If G can be written

$$G(y, s) = a(s)^\top y - b(s), \tag{8}$$

then we have a linear approximation problem and seek to approximate b by a linear combination of $a_1, \ldots, a_n$ which are real-valued functions on S. The corresponding discretized problem becomes a linear program which may be solved by means of the simplex algorithm. See, e. g., [8]. Due to the special structure of these problems the exchange algorithms by Remez, see [2] are applicable. If these latter algorithms are used for the original problem, one needs to perform a global optimization in each exchange step. The three-phase algorithm described in [8,9] and [10] may be adapted to the uniform approximation problem as well. The idea is to seek q points $\{s_1, \ldots, s_q\} \in S$ such that the absolute value of $G(y, s)$ achieves its maximum y_0. One needs to keep track of the sign of the deviation and whether the extreme value is achieved at a boundary point of S or in the interior. The number of local extrema as well as their approximate positions may be obtained from a discretized version of the problem in the case of (8), i. e. the linear approximation problem. In the nonlinear case they are generally found by other means. See, e. g., papers in [13] and [16]. We illustrate this with an example.

Example 10 Determine the straight line

$$y_1 + y_2 s$$

which approximates the function $\exp s$ best in the maximum norm over the real interval $[0, 1]$. Thus we seek the solution to the problem

$$\min_{y_1, y_2} \max_{0 \leq s \leq 1} |e^s - y_1 - y_2 s| .$$

It is easy to verify, e. g., from a simple graph, that the maximum deviation in absolute value occurs at three points s_1, s_2, s_3 and further, that $s_1 = 0, s_3 = 1$. Thus these

points are at the boundary of S and s_2 is in the interior. Hence we get a condition on the derivative of the pointwise deviation considered as a function of s at s_2. Let y_0 be the maximal absolute value of the deviation. Then we obtain the following nonlinear system of equations

$$\begin{cases} \exp(s_i) - y_1 - y_2 s_i & = y_0, \quad i = 1, 3, \\ -\exp(s_2) + y_1 + y_2 s_2 & = y_0, \\ \exp(s_2) - y_2 & = 0, \\ s_1 = 0, \quad s_3 & = 1. \end{cases}$$

This system may be solved by means of Newton's method. Here it was easy to guess the form of the nonlinear system giving the optimal solution. The fact that the infinitely many constraints are satisfied, namely that the absolute value of the deviation at each point is less than or equal to y_0 may be verified analytically. In a more general context, the nonlinear system giving the optimal solution may be constructed from a discretized version of the approximation problem. Still, the verification that the optimum approximation has been found may be nontrivial, even in the case of linear approximation problems. In this situation, quasirandom methods may be contemplated. Experiments with this approach will be reported elsewhere.

See also

- ▶ Semi-infinite Programming and Control Problems
- ▶ Semi-infinite Programming: Discretization Methods
- ▶ Semi-infinite Programming: Methods for Linear Problems
- ▶ Semi-infinite Programming: Numerical Methods
- ▶ Semi-infinite Programming: Second Order Optimality Conditions
- ▶ Semi-infinite Programming, Semidefinite Programming and Perfect Duality

References

1. Brans JP (eds) (1981) Operations research '81. North-Holland, Amsterdam
2. Cheney EW (1966) Introduction to approximation theory. McGraw-Hill, New York
3. Conn AR, Gould NIM (1987) An exact penalty function for semi-infinite programming. Math Program 37:19–40
4. Coope ID, Price CJ (1994) A two-parameter exact penalty function for nonlinear programming. J Optim Th Appl 83(1):49–61
5. Coope ID, Price CJ (1998) Exact penalty function methods for nonlinear semi-infinite programming. In: Reemtsen R, Rückmann JJ (eds) Semi-infinite programming. Kluwer, Dordrecht, pp 49–61
6. Coope ID, Watson GA (1985) A projected Lagrangian algorithm for semi-infinite programming. Math Program 32:337–356
7. Fiacco AC, Kortanek KO (eds) (1983) Semi-infinite programming and applications. Lecture Notes Economics and Math Systems. Springer, Berlin
8. Glashoff K, Gustafson S-Å (1983) Linear optimization and approximation. Springer, Berlin
9. Glashoff K, Gustafson S-Å (1978) Einführung in die Lineare Optimierung. Wissenschaftl Buchgesellschaft Darmstadt, Darmstadt
10. Goberna MA, López MA (1998) Linear semi-infinite optimization. Wiley, New York
11. Gustafson S-Å (1981) A general three phase algorithm for nonlinear semi-infinite programs. In: Brans JP (ed) Operational Research '81. North-Holland, Amsterdam, pp 495–508
12. Gustafson S-Å (1983) A three phase algorithm for semi-infinite programs. In: Fiacco AC, Kortanek KO (eds) Semi-Infinite Programming and Applications. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 138–157
13. Hettich R (ed) (1979) Semi-infinite programming and applications. Lecture Notes Control Inform Sci. Springer, Berlin
14. Hettich R, Zencke P (1982) Numerische Methoden der Approximation und Semiinfiniten Optimierung. Teubner, Leipzig
15. Price CJ (1992) Non-linear semi-infinite programming. PhD Thesis, Univ Canterbury, Canterbury, New Zealand
16. Reemtsen R, Rückmann JJ (eds) (1998) Semi-infinite programming. Kluwer, Dordrecht

Semi-infinite Programming and Control Problems

J. E. Rubio
School of Math., University Leeds, Leeds, England, UK

MSC2000: 49J27, 90C34, 03H10

Article Outline

Keywords
Integral Relationships
Metamorphosis
Existence and Structure
Approximations and Examples
Unbounded Controls and Nonstandard Methods

See also
References

Keywords

Control theory; Calculus of variations; Measure theory; Unbounded controls; Partial differential equations; Diffusion equation; Nonstandard analysis; Shock

We have developed over the last twenty years an approach for the study of *optimal control* and *variational problems* based on the consideration of *measure spaces*; see [11,12] and the references there. In many ways this work has been inspired by the work of L.C. Young [19] which, starting in the period between the Wars, opened up a new vista for mathematics; concepts associated with distributions and chains, for instance, are descendents of his work, and so is our contribution.

Integral Relationships

The applications of measure theory to optimization problems are based on the identification of linear functionals with a class of *Radon measures*, by *Riesz' theorem* [11]. We note that this theorem only applies if the underlying space is compact, this will cause trouble when considering unbounded sets of controls: the handling of infinities there will be done by means of *Loeb measures* in the setting of *nonstandard analysis*.

In order to use Riesz' theorem we need to rewrite the relationships associated with optimization problems in the form of linear functionals on appropriate function spaces. We show how to do this in two specific cases.

We consider first a *finite-dimensional control problem*, to be referred to as problem P1. Let x, u be vectors in Euclidean n-spaces $\mathbf{R}^n$ and $\mathbf{R}^m$ respectively, t a real variable, $J := [t_0, t_f]$ with $t_0 < t_f$, A a compact subset of $\mathbf{R}^n$, x_0, x_f points in A, a bounded, closed subset U of $\mathbf{R}^m$. Further, let $\Omega_1 := J \times A \times U$, and $g\Omega_1 \to \mathbf{R}^n$ a continuous function. The *control function* $t \in J \to u(t) \in U$ is Lebesgue-measurable, and the *trajectory function* $t \in J \to x(t) \in A$ is the (absolutely continuous) solution of

$$\dot{x}(t) = g(t, x(t), u(t)), \qquad t \in J^\circ, \tag{1}$$

the differential equation describing the system to be controlled. We assume that the class $\mathcal{F}_1$ of all *admissible* trajectory-control pairs $p := [x(\cdot), u(\cdot)]$ is nonempty, and seek to minimize the functional $I: \mathcal{F}_1 \to \mathbf{R}$

$$I(p) = \int_{t_0}^{t_f} f_{01}(t, x(t), u(t))\, dt, \tag{2}$$

for $p \in \mathcal{F}_1$. Here f_{01} is a continuous function defined on Ω_1.

We develop now some equalities that are satisfied by the admissible pairs. Let B be an open ball in $\mathbf{R}^{n+1}$ containing $J \times A$; we denote by $C'(B)$ the space of all real-valued functions on B that are uniformly continuous on B together with their first derivatives. Let $\phi \in C'(B)$; define

$$\widehat{\phi}(t, x, u) := \phi_x(t, x)u + \phi_t(t, x) \tag{3}$$

for all $(t, x, u) \in \Omega_1$. Of course, $\widehat{\phi} \in C(\Omega_1)$. If $p = [x(\cdot), u(\cdot)]$ is an admissible pair,

$$\int_J \widehat{\phi}(t, x(t), u(t))\, dt = \phi(t_f, x_f) - \phi(t_0, x_0) := \Delta\phi, \tag{4}$$

for all $\phi \in C'(B)$. There are two special cases which are of interest; in the first we put $\psi(t, x) := x_j\psi(t)$, with $1 \le j \le n$, and $\psi \in \mathcal{D}(J^\circ)$; see [11]. Then, putting $\psi_j(t, x, u) := x_j\psi'(t) + u_j\psi(t)$, for $1 \le j \le n$ and $\psi \in \mathcal{D}(J^\circ)$, the equality (4) becomes $\int_J \psi_j(t, x(t), u(t))\, dt = 0$, since the (test) functions in $\mathcal{D}(J^\circ)$ are zero at the boundary of J. The second case of interest happens when the function ϕ is chosen as a differentiable function of the time t only, $\psi(t, x, u) := \theta(t)$, $(t, x, u) \in \Omega_1$, then $\widehat{\psi}(t, x, u) = \dot{\theta}(t)$, $(t, x, u) \in \Omega_1$. We introduce a subspace of $C(\Omega_1)$, to be denoted by $C_a(\Omega_1)$, consisting of those functions which depend only on the first variable t; then the equalities (4) become: $\int_J h(t, x(t), u)\, dt = a_h$, $h \in C_a(\Omega_1)$, with a_h the Lebesgue integral of $h(\cdot, x, u)$ over J, independent of x and u. We will now choose for each of these spaces countable sets of functions whose linear combinations are dense in the corresponding spaces in the appropriate topologies. Thus we obtain countable sets of equalities.

For the space $C'(\Omega_1)$ we shall choose $\{\phi_i\}$, a set of polynomials in $(t, x_1, \ldots, x_n)$; for $\mathcal{D}(J^\circ)$, $\{\chi_j\}$, the sequence of functions of the type when the functions ψ are the sine and cosine functions

$$\sin\left(2\pi r \frac{t - t_0}{\Delta t}\right), \quad 1 - \cos\left(2\pi r \frac{t - t_0}{\Delta t}\right),$$

$r = 1, 2, \ldots, j = 1, \ldots, n$, and for $C_a(\Omega_1)$ the sequence $\{h_k\}$, a set of polynomials in t. Thus, finally, we obtain our set of integral equalities:

$$\begin{aligned}
&\int_J \widehat{\phi}_i(t, x(t), u(t))\, dt = \Delta\phi_i, \qquad i = 1, 2, \ldots,\\
&\int_J \chi_j(t, x(t), u(t))\, dt = 0, \qquad j = 1, 2, \ldots,\\
&\int_J h_k(t, x(t), u(t))\, dt = a_{h_k}, \qquad k = 1, 2, \ldots.
\end{aligned} \tag{5}$$

We will consider now an optimal control problem, to be denoted by P2, associated with a *nonlinear diffusion equation*. We follow the notation in [7] and [13]; see also [4], where we study a nonlinear wave equation. Take D, a bounded domain in $\mathbf{R}^n$ with smooth boundary ∂D, and T, a positive real number, and define: $Q_T := D\times (0, T)$; $\Gamma_T := \partial D \times (0, T)$; $D_0 := D\times \{0\}$; $D_T := D\times \{T\}$.

We also choose some functions: $\overline{Q}_T\mathbb{R}$, $k \in K$; $f\colon \mathbb{R}\times\mathbb{R}^n\times\overline{Q}_T \to \mathbb{R}$, $f \in C(\mathbb{R}\times\mathbb{R}^n\times\overline{Q}_T)$.

Consider the nonlinear diffusion equation:

$$\begin{aligned}
u_t(x, t) - \operatorname{div}(k(x, t)\nabla u(x, t))\\
= f(u(x, t), \nabla u(x, t), x, t)
\end{aligned} \tag{6}$$

for $(x, t) \in Q_T$, with the initial condition $u(x, 0) = 0$, $x \in D$, and the *boundary condition* $\nabla u\cdot n|_{\Gamma_T} = v$; here n is the outward normal, and the function $(s, t) \in \Gamma_T \to v(s, t) \in V \subset \mathbf{R}$ is the *control function*, taking values in a *bounded* control set V.

A pair (u, v) of trajectory-function u and control-function v is said to be *admissible* if:

i) The function $(x, t) \to u(x, t)$ is a classical solution of (6), that is, it is in $C^{2,1}(Q_T) \cap C(Q_T \cup \Gamma_T \cup \overline{D}_0)$ and satisfies (5).
ii) The control function is continuous in Γ_T.
iii) The terminal relationship $u(\cdot, T) = g$ is satisfied; g is a given continuous function on D_T.

The set of admissible pairs for this problem will be denoted by $\mathcal{F}_2$, and assumed to be nonempty. Since the control set V is bounded, then there are *bounded* sets $A \subset \mathbf{R}$ and $B \subset \mathbf{R}^n$ so that $u(x, t) \in A$, $\nabla u(x, t) \in B$, for all $(x, t) \in \overline{Q}_T$. Since there are many such sets A and B, we choose from those the *minimum sets*, that is, either the intersections $\cap A$ and $\cap B$ of all sets satisfying the relations above, or subsets of them. Thus, every point in our set A (respectively, B) will be a state (respectively, a gradient of a state) which can be reached by an admissible control inside the time interval $[0, T]$.

The optimization problem associated with this equation is as follows. Let f_{02}, f_1 be nonnegative, Lipschitz-continuous real-valued functions on $\mathbf{R}^{2n+2}$, $\mathbf{R}^{n+1}$ respectively. Then we wish to find a minimizing pair (u, v) of admissible trajectory u and control v for the functional

$$\begin{aligned}
J(u, v) := \int_{Q_T} f_{02}(u(x, t), \nabla u(x, t), x, t)\, dx\, dt\\
+ \int_{\Gamma_T} f_1(v(s, t), s, t)\, ds\, dt.
\end{aligned} \tag{7}$$

We transform now this problem. Let ψ be in K. Then one can show [7] that the classical solution of (6), if it exists, satisfies the integral relation

$$\begin{aligned}
\int_{Q_T} [u\psi_t - k\nabla u\nabla\psi + f\psi]\, dx\, dt\\
= -\int_{\Gamma_T} k\psi v\, ds\, dt + \int_{D_T} g\psi\, dx,
\end{aligned} \tag{8}$$

for all $\psi \in K$. To these equalities we must add the equivalent of the last set of equations in (5). If a function ξ: $A \times B \times Q_T \to \mathbf{R}$ depends only on (x, t), $\int_{Q_T}\xi dx\, dt = a_\xi$, the Lebesgue integral of ξ over Q_T. Also, if a function ς: $V \times \Gamma_T \to \mathbf{R}$ depends only on (s, t), $\int_{\Gamma_T} \varsigma\, ds\, dt = b_\varsigma$, the Lebesgue integral of ς over Γ_T. By choosing countable dense sets as before, $\{\psi_i\}$, $\{\xi_j\}$, $\{\varsigma_k\}$, we can obtain a countable set of equalities associated with problem P2:

$$\begin{aligned}
&\int_{Q_T} [u\psi_{it} - k\nabla u\nabla\psi_i + f\psi_i]\, dx\, dt\\
&= -\int_{\Gamma_T} k\psi_i v\, ds\, dt + \int_{D_T} g\psi_i\, dx,\\
&i = 1, 2, \ldots,\\
&\int_{Q_T} \xi_j\, dx\, dt = a_{\xi_j}, \qquad j = 1, 2, \ldots,\\
&\int_{\Gamma_T} \zeta_k\, ds\, dt = b_{\zeta_k}, \qquad k = 1, 2, \ldots.
\end{aligned} \tag{9}$$

Metamorphosis

We proceed to transform the control problems defined above; instead of minimizing over sets of admissible pairs trajectory-control, we find that it is possible to

minimize over a *measure space*, in the case of problem P1, and the product of two measure spaces, in that of problem P2. In general, the minimization of the functionals (2) over the set $\mathcal{F}_1$ and (7) over the set $\mathcal{F}_2$ are not possible: the infima are not attained; it is not possible, for instance, to write *necessary conditions* for these problems. We proceed then to transform them.

The advantages of the new formulations are:

i) An automatic existence theory: there always are minimizers for our measure-theoretical problems;
ii) The new problems are linear, and then one can use the whole paraphernalia of linear analysis;
iii) Also, our minimization is *global*: the value reached, say, numerically is close to what one could reasonably call the global infimum of each problem.

The price to pay for these advantages is that the final state is reached only *asymptotically*: that is, as the number of (linear) constraints associated with the measure-theoretical problem tends to infinity.

Let us consider first problem P1. An admissible pair $p := [x(\cdot), u(\cdot)]$ defines a linear, bounded, positive functional

$$p\colon\ F \to \int_J F(t, x(t), u(t))\, dt \in \mathbb{R}$$

in the space $C(\Omega_1)$ of continuous real-valued functions F, with $\Omega_1 := J \times A \times U$. By Riesz' theorem, the admissible pair p defines a *Radon measure* μ on Ω_1 so that (2) becomes

$$I(\mu) = \mu(f_{01}), \tag{10}$$

while (5) becomes

$$\begin{aligned} &\mu(\widehat{\phi}_i) = \Delta\phi_i, \qquad i = 1, 2, \ldots, \\ &\mu(\chi_j) = 0, \qquad j = 1, 2, \ldots, \\ &\mu(h_k) = a_{h_k}, \qquad k = 1, 2, \ldots, \end{aligned} \tag{11}$$

where we have written $\mu(F) := \int_{\Omega_1} F\, d\mu$. Note that we have not achieved anything new so far; the minimization of the functional (2) over (5) is exactly equivalent to that of the functional $\mu(f_{01})$ over (11). We shall consider below the extension of our problem, the minimization of (10) over the set S_1 of *all* measures μ in $\mathcal{M}^+(\Omega_1)$ satisfying (11).

In the case of Problem P2, we can proceed similarly. A solution of (6) defines a linear, bounded positive functional

$$u(\cdot,\cdot)\colon\ F \to \int_{Q_T} F(u(x,t), \nabla u(x,t), x, t)\, dx\, dt$$

in the space $C(\Omega_2)$ of continuous real-valued functions F, with $\Omega_2 := A \times B \times Q_T$. Also, a control v defines a linear, bounded, positive functional:

$$v(\cdot,\cdot)\colon\ G \to \int_{\Gamma_T} G(v(s,t), s, t)\, ds\, dt$$

in the space $C(\omega)$ of continuous functions G, $\omega := V \times \Gamma_T$.

By Riesz's theorem, an admissible pair (u, v) defines two Radon measures λ and ν, the first on Ω_2, the second on ω, so that (9) becomes:

$$\begin{aligned} &\int_{\Omega_2} F_i\, d\lambda + \int_{\omega} G_i\, d\nu = \int_{D_T} g\psi_i\, dx := \alpha_i, \\ &i = 1, 2, \ldots, \end{aligned} \tag{12}$$

where

$$\begin{aligned} &F_i(u, w, x, t) := u\psi_{it}(x, t) \\ &\quad - \kappa(x, t) w \nabla\psi_i(x, t) + f(u, w, x, t)\psi_i(x, t), \\ &G_i(v, s, t) := \psi_i(x|_{\partial D}, t) v. \end{aligned}$$

Thus, the minimization of the functional (7) over $\mathcal{F}_2$ is equivalent to the minimization of

$$I(\lambda, \nu) = \lambda(f_{02}) + \nu(f_1), \tag{13}$$

where we have written as above $\lambda(f)$ for $\int_{\Omega_2} f\, d\lambda$, and $\nu(g)$ for $\int_{\omega} f\, d\nu$, over the set of measures (λ, ν) corresponding to admissible pairs, which satisfy

$$\begin{aligned} &\lambda(F_i) + \nu(G_i) = \alpha_i, \qquad i = 1, 2, \ldots, \\ &\lambda(\xi_j) = a_{\xi_j}, \qquad j = 1, 2, \ldots, \\ &\nu(\zeta_k) = b_{\zeta_k}, \qquad k = 1, 2, \ldots. \end{aligned} \tag{14}$$

Again, we have not achieved anything new. As in the case of P1, we shall consider the extension of our problem, the minimization of (13) over the set S_2 of *all pairs* of measures (λ, ν) in $\mathcal{M}^+(\Omega_2) \times \mathcal{M}^+(\omega)$ satisfying (14).

We have developed, therefore, two *infinite-dimensional linear programming* problems, the minimization of linear forms (10) or (13) over sets S_1 and S_2, respectively, of positive Radon measures satisfying countably-infinite sets of linear equalities, (11) or (14). In the next section we consider the two main problems associated with their usefulness: do they have solutions; and how do the solutions help us solve our optimization problems.

Existence and Structure

We start by choosing finite, but variable, subsets of equalities from the sets (11) and (14); in this way we consider two *semi-infinite* linear programming problems, with finite number of equalities taking place in infinite-dimensional spaces. The first, to be denoted by $LS1 = LS1(M_1, M_2, M_3)$ consists in minimizing the linear functional (10) over the subset $S_1(M_1, M_2, M_3)$ of $\mathcal{M}^+(\Omega_1)$ defined by the equalities

$$\begin{aligned} &\mu(\widehat{\phi}_i) = \Delta\phi_i, \qquad i = 1, \ldots, M_1, \\ &\mu(\chi_j) = 0, \qquad j = 1, \ldots, M_2, \\ &\mu(h_k) = a_{h_k}, \qquad k = 1, \ldots, M_3; \end{aligned} \tag{15}$$

while the second, to be denoted by $LS2 = LS2(N_1, N_2, N_3)$ consists in minimizing the linear functional (13) over the subset $S_2(N_1, N_2, N_3)$ of $\mathcal{M}^+(\Omega_2) \times \mathcal{M}^+(\omega)$ defined by

$$\begin{aligned} &\lambda(F_i) + \nu(G_i) = \alpha_i, \qquad i = 1, \ldots, N_1, \\ &\lambda(\xi_j) = a_{\xi_j}, \qquad j = 1, \ldots, N_2, \\ &\nu(\zeta_k) = b_{\zeta_k}, \qquad k = 1, \ldots, N3. \end{aligned} \tag{16}$$

We can prove that our minimization is global [11,12].

Proposition 1

i) As $M_1, M_2, M_3 \to \infty$,

$$\begin{aligned} &\inf_{S_1(M_1,M_2,M_3)} \mu(f_{01}) \\ &\to \inf_{S_1} \mu(f_{01}) \le \inf_{\mathcal{F}_1} I. \end{aligned} \tag{17}$$

ii) As $N_1, N_2, N_3 \to \infty$,

$$\begin{aligned} &\inf_{S_2(N_1,N_2,N_3)} [\lambda(f_{02}) + \nu(f_1)] \\ &\to \inf_{S_2} [\lambda(f_{02}) + \nu(f_1)] \le \inf_{\mathcal{F}_2} J. \end{aligned} \tag{18}$$

Thus, we can approach the *global infima* by taking a large enough number of equalities. The fact that the global infima can be strictly less than the classical one is discussed in [11].

There are two aspects of these *semi-infinite linear programming* problems which are of interest to us; their characteristics such as existence and characterization of solutions, and the relationship of these solutions to the original optimization problems. We examine first the linear programs themselves. The conclusions of the following proposition follow from weak*-compactness of the sets of measures, and *Rosenbloom's theorem* [10]. We denote by $\delta(z)$ the atomic measure with support $\{z\}$.

Proposition 2

i) The linear programs LS1 and LS2 defined by (10)–(15) and (13)–(16) respectively have minimizers.

ii) The solution of the program LS1, (10)–(15), has the form

$$\mu_{opt} = \sum_{i=1}^{M} \alpha_i \delta(t_i, x_i, u_i),$$

with $\alpha_i \ge 0$, $M := M_1 + M_2 + M_3$.

iii) The solution of the program LS2, (13)–(16), has the form

$$\begin{aligned} &(\lambda_{opt}, \nu_{opt}) \\ &= \left(\sum_{i=1}^{N} \alpha_i \delta(u_i, w_i, x_i, t_i), \sum_{i=1}^{N} \beta_i \delta(v_i, s_i, t_i) \right), \end{aligned} \tag{19}$$

with $\alpha_i, \beta_i \ge 0$, $N := N_1 + N_2 + N_3$.

In part iii) of this Proposition we have identified (λ, ν) with the product $\mu \times \nu$; other possibilities exist [12].

We study now the other aspect of these solutions, how useful they are. How do we construct *suboptimal* pairs of trajectories and controls for our functionals, once we have solved the linear programming problems? We start with the problem *LS*1, (10)–(15), and shall proceed in several steps:

1) We proceed to obtain a solution μ_{opt} of the form given above in ii).
2) We obtain a weak*-approximation to this measure by a pair of piecewise constant functions (x, u). The exact procedure for this construction, to be explained in detail below, involves the use of the support points of μ_{opt} as well as the coefficients α_i.
3) We use the function u as control in (1) with $x(t_0) = x_0$ and obtain a trajectory $x_R(\cdot)$. The pair $p_R := (x_R, u)$ can be shown to have the following properties:
   a) By taking the numbers M_1, M_2, M_3 sufficiently large, one can make $I(p_R)$ as near as desired to $\inf_S \mu(f_{01})$.
   b) The final state $x_R(t_f)$ tends to x_f as $M_1, M_2, M_3 \to \infty$.
   c) The constraint $x_R(t_f) \in A$ tends to be satisfied in a similar manner.

In the case of problem *LS*2, (13)–(16), we proceed in a similar fashion.

1) Firstly we shall obtain optimal measures $(\lambda_{\text{opt}}, \nu_{\text{opt}})$ for this problem.
2) We obtain then a weak* approximation to $(\lambda_{\text{opt}}, \nu_{\text{opt}})$ by a set of two piecewise-constant functions (u, v) by means of results involving weak*-density.
3) The control function v obtained above is in $L_2(\Gamma_T)$, that is, for each $t \in (0, T)$, $v(\cdot, t) \in L_2(\partial D)$, since it is piecewise constant and the set Q_T is bounded. It can serve then as boundary function for a (necessarily weak) solution of the system (6) to be denoted by u_v. This solution will be in $H^1(Q_T)$.
4) Borrowing the term from H. Rudolph [17], we shall call the pair (u_v, v) of trajectory and control functions *asymptotically admissible* if:
   a) The control function $v \in L_2(Q_T)$, and $v(s, t) \in V$.
   b) The trajectory u_v is the weak solution of (6) corresponding to the admissible control $v \in L_2(Q_T)$.
   c) The trajectory function satisfies the appropriate constraints.
   d) The final value $u_v(\cdot, T)$ of the trajectory function tends in $L_2(D_T)$ to the prescribed function g as $N_1, N_2, N_3 \to \infty$.
5) If the numbers N_1, N_2, N_3 are sufficiently large, and the weak*-approximation in step ii) above sufficiently good, then the value at the pair (u_v, v) of the functional J, $J(u_v, v)$, is close to the $\inf_S J$, and thus is a good suboptimal pair. Note that no use is made of the trajectory u, obtained in step ii) together with the control v.

We proceed in the next section to deal with approximation and examples.

Approximations and Examples

The pattern should be clear; if one wants to obtain an estimate for a solution to problems such as P1 or P2, one should develop semi-infinite linear programming problems such as *LS*1 or *LS*2, find out the corresponding solutions (the coefficients α_i and the supports of the atomic measures as in Proposition 1) and then build suboptimal pairs. There are two main ways of estimating solutions of semi-infinite linear programming problems such as *LS*1 or *LS*2.

The first consists in using functional analytical techniques in the space of measures; see [5,17]. *Rudolph's method* has been used with great success to solve problems such as P1, in a large number of dimensions; see [18] for a most impressive application of Rudolph's method, as well as of our theory.

The second approach consists in approximating the infinite-dimensional problem by a finite-dimensional one, taking place in a Euclidean space of large dimension. We indicate first how to do this with respect to Problem P1. We note that in this problem the continuous functions h_k, $k = 1, \ldots, M_3$ have been replaced in practice by M_3 *lower semicontinuous* pulse-like functions, also to be denoted by h_k; the set J is divided into M_3 equal segments, and the function h_k equals 1 in the kth of such segments, zero elsewhere. This is explained in detail in [11], and has been done because it brings better stability to the numerical processes.

A further concept must be introduced now [11]. Note that we have in *LS*1 a *nonlinear* optimization problem, in which the unknowns are the coefficients α_i and supports (t_i, x_i, u_i), $i = 1, \ldots, M$. In order to find a linear approximation to this problem, we consider $\widehat{\omega}$, a countable dense subset of $\Omega_1 := J \times A \times U$. Taking $\Upsilon \gg M$ elements from ω, we can write (10)–(15) as follows. We write $w_\ell := (t_\ell, x_\ell, u_\ell)$, and wish to minimize

$$\sum_{\ell=1}^{\Upsilon} \alpha_\ell f_{01}(w_\ell) \tag{20}$$

on the set defined by the elements $\alpha_\ell \geq 0$, $\ell = 1, \ldots, \Upsilon$, which satisfy, further,

$$\begin{cases} \sum_{\ell=1}^{\Upsilon} \alpha_\ell \widehat{\phi}_i(t_\ell, x_\ell, u_\ell) = \Delta\phi_i, & i = 1, \ldots, M_1, \\ \sum_{\ell=1}^{\Upsilon} \alpha_\ell \chi_j(t_\ell, x_\ell, u_\ell) = 0, & j = 1, \ldots, M_2, \\ \sum_{\ell=1}^{\Upsilon} \alpha_\ell h_k(t_\ell, x_\ell, u_\ell) = a_{h_k}, & k = 1, \ldots, M_3. \end{cases} \tag{21}$$

Here, then, the supports w_ℓ are fixed, in $\widehat{\omega}$; the coefficients α_ℓ, $\ell = 1, \ldots, \Upsilon$, are the only unknowns; this is an $M \times \Upsilon$ (finite-dimensional) linear program. Of course as $\Upsilon \to \infty$ the support of the optimal measure μ_{opt} in Proposition 2 can be approximated closer and closer by that of μ^{Υ}_{opt}, the solution of (20)–(21). Note, further, that at most M of the unknown α-s are nonzero; we shall assume that the problem has essential

regularity, and that exactly M of these α-s are nonzero; see [11] for a discussion of this point. Once this *finite-dimensional linear program* is solved, suboptimal pairs can be constructed as explained above; the construction of the control function from the coefficients α_ℓ and supports u_ℓ is as follows.

1) The time set $J = [t_0, t_f]]$ has been divided into M_3 equal subdivisions J_k, $k = 1, \ldots, M_3$, each of measure $\Delta t/M_3$, with $\Delta t := t_f - t_0$.
2) There is a total of M indices ℓ associated with those values α_ℓ that are nonzero. To each of the subdivision J_k of J defined above are associated a number of these indices; if only one is so associated, then the value of α_ℓ equals $\Delta t/M_3$; if more than one are associated, then the sum of the corresponding α_ℓ-s adds up to $\Delta t/M_3$.
3) Without loss of generality, let J_k, for $1 \le j \le M_3$ be associated with two α_ℓ-s, α_{ℓ_1} and α_{ℓ_2}, as explained above, a typical situation; then we build the curve $u(\cdot)$ on J_k by making $u(t)$, $t \in J_k$, equal to u_{ℓ_1} or u_{ℓ_2} in each of the two partitions of J_k with lengths α_{ℓ_1} and α_{ℓ_2} respectively.

An example of this process will be given in the next section.

In the case of problem P2, a similar construction can be made. We have however chosen here Rudolph's method, for which an initial finite-dimensional estimate is necessary.

It is convenient to work in $\mathcal{M}^+(\Omega_2 \times \omega)$, identifying $(\lambda, \nu) \to \lambda \times \nu$; see [13] for details. The system we have chosen is

$$\begin{aligned} & u_t(x,t) - \operatorname{div}\left(\frac{x^2}{1+t^2}\nabla u(x,t)\right) \\ & = \frac{\|\nabla u(x,t)\|_E^2 + 1}{1 + x^2 + u(x,t)^2}, \\ & u_x(0,t) = 0, \qquad u_x(1,t) = v(t), \\ & t \in (0,1), \qquad x \in (0,1); \end{aligned} \tag{22}$$

that is, $k(x,t) := x^2/(1+t^2)$, $f(u,w,x,t) := \| w \|_E^2 + 1)/(1 + x^2 + u^2)$. We have taken $V = [-10, 10]$, $g(x) = 0.075$, $x \in [0, 1]$, and we wish to minimize:

$$J(u,v) = \int_{Q_T} \frac{u(x,t)^2 + \|\nabla u(x,t)\|_E^2}{1 + \sin^2(tu(t,x))}\, dt;$$

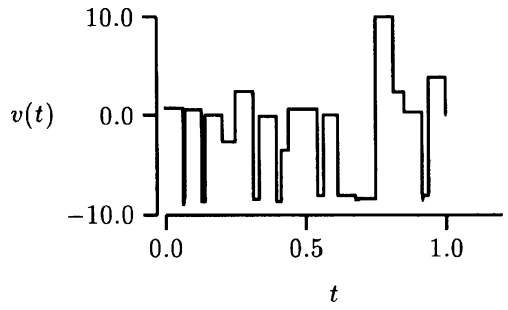
that is, $f_{02}(u,w,x,t) := (u^2 + \|w\|_E^2)/(1+\sin^2(tu))$, and $f_1 \equiv 0$. The boundary ∂D is composed of two points only, of which only one, the one at $x = 1$, plays an active role, the control being the heat flow at that point.

The functions ψ were chosen of the form $\psi(x, t) = t^p \cos(\ell\pi x) + q(t)$, or $\psi(x, t) = t^p \sin(h\pi x) + q(t)$; the functions q are test functions introduced to improve the behavior at $x = 1$ for the determination of an *initial solution*, as explained below. Ten such functions ψ were chosen, with values of $p = 1, 2$ and $\ell = 1, 2, 3$, $h = 1, 2$. The 16 functions ξ were chosen by dividing the square $[0, 1] \times [0, 1]$ into 16 equal squares, the functions ξ being the characteristic functions of the individual squares. Thus the total number of constraints m equals $m = 1 + 10 + 16 = 27$. The computational method consists of three steps:

i) The most difficult problem encountered was that of finding an initial solution from which, in part ii) of the method, one can iterate towards the minimum. This was done here by means of a finite-dimensional linear program, obtained by *discretizing* all the variables of the problem. It was necessary to find an initial solution by first solving for some of the parameters, thus the need of the functions q introduced above, because the (rather rough) discretization tended to make the problem infeasible. Then a finite-dimensional *simplex program* was run, of rather small size, with 677 variables and, of course, 27 constraints. Only the first phase (the one that produces a feasible solution) was run. It is usual in these problems to use a discretized solution as an initial one; see [5,17, Chapts. 5, 6].
ii) Then the *simplex algorithm* of Rudolph was run using the output of step i) as initial solution, and after 87 iterations it converged to a value of 0.202247; it had started with a value of 1.93919. This is a numerical estimation of the *global minimum*.
iii) Once the optimization is performed, a nearly-optimal control v can be constructed; the method is shown in detail in [13], and follows the same general rules as the previous case. The graph of the resulting control is shown in Fig. 1.

Unbounded Controls and Nonstandard Methods

We consider in this last section a finite-dimensional control problem just like P1, but in which the control set U is unbounded. We shall use the same notation for P1 as above. The problem now is that since U is un-



Semi-infinite Programming and Control Problems, Figure 1
Graph of the nearly-optimal control *v*. Note that the control set is *V* = [−10, 10]

bounded we may obtain 'impulses' as controls and thus discontinuous trajectories. We need to be able to handle infinities, a role for which nonstandard analysis is well suited. We start by extending the control space, and consider U a subset of $\overline{\mathbb{R}}^m$, the m-dimensional space generated by the extended real line $\overline{\mathbb{R}}$. Nothing much has changed, but we can put a topology on $\Omega := J \times A \times U$ which makes it compact [1]. We proceed now with our nonstandard construction; see [2,12,14]. We will work in a *nonstandard framework* given by a *superstructure* $V(W)$, $\overline{\mathbb{R}} \subset W$. The superstructure $V(^*V)$ is also an *enlargement*, and $\mathcal{N}_1$-saturated. We study integrals of the form

$$\int_J f(t, x(t), u(t))\, dt, \tag{23}$$

with $p \in \mathcal{F}_1$ and $f \in C(\Omega')$. Then,

$$\forall p \in \mathcal{F}_1 \colon \int_J f(t, x(t), u(t))\, dt \in \overline{\mathbb{R}}; \tag{24}$$

by transfer,

$$\forall p \in^* \mathcal{F}_1 \colon {}^*\int_{^*J} {}^*f(t, x(t), u(t))\, dt \in {}^*\overline{\mathbb{R}}. \tag{25}$$

Thus, the nonstandard version of P1 consists in minimizing

$${}^*I(p) := {}^*\int_{^*J} {}^*f_{01}(t, x(t), u(t))\, dt \tag{26}$$

on the class ${}^*\mathcal{F}_1$ of pairs satisfying

$${}^*\int_{^*J} {}^*f_i(t, x(t), u(t))\, dt = b_i, \quad i = 1, \ldots, M, \tag{27}$$

where the system (15) has been written in a compact form. Consider now the map suggested by (24). If $p \in \mathcal{F}_1$ is fixed, the map

$$\nu_p \colon F \to \int_J F(t, x(t), u(t))\, dt \in \overline{\mathbb{R}}, \quad F \in C(\Omega') \tag{28}$$

is linear and positive. By Riesz's theorem, there is a measure, to be denoted also by ν_p, on the *Borel sets* $\mathcal{B}$ of Ω', that represents this map; remember that Ω' is compact. Then $(^*\Omega', {}^*\mathcal{B}, {}^*_p\nu)$ is a nonstandard measure space and then (see [9]):

Lemma 3 *There is a measure space $(^*\Omega', \mathcal{A}, \mu_L^p)$ so that μ_L^p is the Loeb measure associated with ν_p; then,*

$${}^*\int_{^*J} F(t, x(t), u(t))\, dt = \mu_L^p(F) := \int_{^*\Omega'} F d\, \mu_L^p, \qquad F \in C(^*\Omega'). \tag{29}$$

The algebra $\mathcal{A}$ is an extension of the algebra ${}^*\mathcal{B}$.

Thus, one can write the optimization problem as the problem of minimizing

$$J(\mu_L^p) := \mu_L^p(f_{01}), \tag{30}$$

over the set $\mathcal{M}_M^L$ of measures of the form μ_L^p defined by

$$\mu_L^p(f_i) = b_i, \qquad i = 1, \ldots, M. \tag{31}$$

Proposition 4

i) The infima associated with the problems (26)–(27) and (30)–(31) are equal.

ii) For any positive infinitesimal $s \in {}^*\overline{\mathbb{R}}$, we can find a near-minimizer $\mu_s \in \mathcal{M}_M^L$ for the functional J in (30) in the set $\mathcal{M}_M^L$, so that

$$J(\mu_s) = \inf_{\mathcal{M}_M^L} J + s. \tag{32}$$

Let, then, s be a fixed positive infinitesimal in ${}^*\overline{\mathbb{R}}$, and μ_s the corresponding near-minimizer for J on $\mathcal{M}_M^L$. We can proceed to map back this measure to the standard world, by means of the *standard part map*, see [2].

Proposition 5 *There is a Baire measure μ_{opt} on Ω' so that:*

i) *If S is a Baire set in Ω',*

$$\mu_{opt}(S) = {}^{\circ}\mu_s(\mathrm{st}_{\Omega'}^{-1}(S)),$$

where $\mathrm{st}_{\Omega'}^{-1}(S)$ is the union of the monads of the elements of S.

ii)
$$\mu_{opt}(f_{01}) := \int_{\Omega'} f_{01}\, d\mu_{opt} \leq \inf_{\mathcal{F}_1} \int_J f_{01}(t, x(t), \dot{x}(t))\, dt. \quad (33)$$

iii) *The measure μ_{opt} is a solution of the following optimization problem. Minimize*

$$\mu(f_{01}) \quad (34)$$

over the set $\mathcal{M}_M^+(\Omega')$ of positive Baire measures on Ω' satisfying

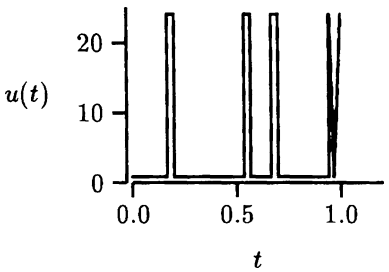
$$\mu(f_i) = b_i, \qquad i = 1, \ldots, M. \quad (35)$$

iv) *The support of μ_{opt} contains subsets of Ω' in which at least one component of the variable u is either $-\infty$ or ∞. The measure μ_{opt} is defined by a Baire measure on $J \times A \times V$, with V the set of all finite elements of U, plus atomic measures on those subsets.*

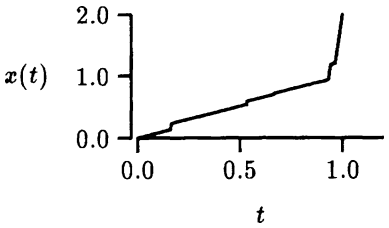
In problems of interest, in which the function f_{01} tends to infinity at infinity, and in which the infimum is finite, elements $(t, x, u) \in \Omega'$ with one or more components of value ∞ or $-\infty$ do not really occur in the support of μ_{opt}; note that expressions such as $\infty - \infty$ are not defined for the extended real line. Thus, if $f_{01}(t, x, z) = \infty$ whenever a component of u is either ∞ or $-\infty$, and that the minimum associated with the linear program is finite, such elements are not present in the support of μ_{opt}.

We are now in a strong position to solve our original problem: the optimization problem P1 with unbounded U in the standard world. We proceed as in P1 in previous sections, solving for μ_{opt}, approximating its support, building suboptimal pairs.

We have solved a simple problem, taken from [6], with $n = 1$ and $f_{01}(t, x, u) := (u^3 - 1)^{1/3}$, and $\dot{x} = u$; the other parameters are $t_0 = 0$, $t_f = 1$, $x_0 = 0$, $x_f = 2$. The numerical approximation was performed with the following parameters: $Q = 24$, $M_1 = 2$, $M_2 = 8$, $M_3 = 30$, $M = 40$, $N = 27000$. Each of the axis associated with the variables (t, x, u) was divided into 30 parts; the minimum obtained was 0.49587, which should be compared with the minimum for the problem obtained by semiclassical means in [6] of 0.413. The approximation problems for this kind of optimization problems are fierce; it is necessary to have a large value of Q as well as very fine mesh, thus very large linear programs. The graphs of the control u and the trajectory x can be seen in Fig. 2 and Fig. 3.



Semi-infinite Programming and Control Problems, Figure 2
Graph of the control *u*



Semi-infinite Programming and Control Problems, Figure 3
Graph of the trajectory *x*

We should note that, even in this simple problem, we have achieved something not easily accomplished by the more traditional methods, which would have found it extremely hard to deal with the cube of a *'delta function'*; they mostly deal in problems in which the control variable, our u, appears *linearly*.

The method employed here appears promising to deal with partial differential equations with solutions exhibiting *shocks*, such as those studied in [8]. See also [15,16].

We have also extended these methods to the *design of optimal shapes* associated with nonlinear differential equations [3].

See also

- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations

- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Optimal Control of a Flexible Arm
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming: Approximation Methods
- ▶ Semi-infinite Programming: Discretization Methods
- ▶ Semi-infinite Programming: Methods for Linear Problems
- ▶ Semi-infinite Programming: Numerical Methods
- ▶ Semi-infinite Programming: Second Order Optimality Conditions
- ▶ Semi-infinite Programming, Semidefinite Programming and Perfect Duality
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Berge C (1963) Topological spaces. Oliver and Boyd, Edinburgh
2. Cutland NJ (eds) (1988) Nonstandard analysis and its applications. Cambridge Univ Press, Cambridge
3. Fakharzadeh A (1997) Shapes, measures and elliptic equations. PhD Thesis, Univ Leeds, Leeds
4. Farahi MH, Rubio JE, Wilson DA (1996) The global control of a nonlinear wave equation. Internat J Control 65:1–15
5. Glashof K, Gustafson S (1983) Linear optimization and approximation. Springer, Berlin
6. Lawden DF (1959) Discontinuous solutions of variational problems. J Austral Math Soc 1:27–37
7. Mikhailov VP (1978) Partial differential equations. MIR, Moscow
8. Oberguggenberger M (1992) Multiplication of distributions and applications to partial differential equations. Longman, London
9. Render H (1993) Pushing down Loeb measures. Math Scand 72:61–84
10. Rosenbloom PC (1952) Quelques classes de problèmes extremaux. Bull Soc Math France 80:183–216
11. Rubio JE (1986) Control and optimization: the linear treatment of nonlinear problems. MUP and Wiley, New York
12. Rubio JE (1994) Optimization and nonstandard analysis. M Dekker, New York
13. Rubio JE (1995) The global control of nonlinear diffusion equations. SIAM J Control Optim 33:308–322
14. Rubio JE (1998) The global optimization of variational problems with discontinuous solutions. J Global Optim 12:225–237
15. Rubio JE (2000) Optimal control problems with unbounded constraint set. Optim 48:191–210
16. Rubio JE. The global control of shock waves
17. Rudolph H (1987) Simplexalgorithmus der semiinfiniten linearen Optimierung. Wiss Leuna–Merseburg Tech Hochsch 5:782–806
18. Rudolph H (1990) Global solution in optimal control via SILP. In: Sebastian H-J (ed) System modelling and optimization. Lecture Notes Control Inform Sci. Springer, Berlin, pp 394–402
19. Young LC (1969) Calculus of variations and optimal control theory. WB Saunders, Philadelphia

Semi-infinite Programming: Discretization Methods

SIP

Rembert Reemtsen
Brandenburg Technical University Cottbus, Cottbus, Germany

MSC2000: 90C34, 90C05, 90C25, 90C30

Article Outline

Keywords
Convergence of Solutions of Discretized SIP Problems
Solution of Discretized SIP Problems
See also
References

Keywords

Semi-infinite programming; Optimization algorithms; Discretization of optimization problems

Consider the following optimization problem with respect to $x \in \mathbf{R}^n$, in which $Y \subseteq \mathbf{R}^m$ is a nonempty com-

pact set and $F \in C(\mathbf{R}^n)$ and $g \in C(\mathbf{R}^n \times Y)$ are given functions:

$$P[Y] \quad \begin{cases} \min & f(x) \\ \text{s.t.} & g(x, y) \le 0, \quad y \in Y. \end{cases}$$

If Y is an infinite set, this problem includes finitely many unknowns and infinitely many inequality constraints and, therefore, is said to be a *semi-infinite programming* (briefly: *SIP*) problem. The problem is denoted as *linear* if f and $g(\cdot, y)$, $y \in Y$, are affine-linear, as *convex* if these functions are convex, and as *nonlinear* in all other cases.

Results which hold for $P[Y]$ under the given general conditions similarly apply to problems with constraints $g_j(x, y) \le 0$, $y \in Y^j$, for $j = 1, \ldots, p$, where $Y^j \subseteq \mathbf{R}^{m_j}$ is nonempty and compact and $g_j \in C(\mathbf{R}^n \times Y^j)$ (e. g. [43]). Note in this connection that an ordinary inequality constraint $\widetilde{g}_j(x) \le 0$ can, for example, be expressed as $g_j(x, y) \le 0$, $y \in \{1\}$, for $[g_j(x, y) := y\widetilde{g}_j(x)$ and that an equality constraint $h_j(x) = 0$ can be described by the two inequality constraints $\pm h_j(x) \le 0$. Inclusion of equality constraints in this way, however, is not possible when the existence of a point is required at which all inequality constraints are strictly satisfied. A comprehensive survey of numerical methods for the solution of such SIP problems can be found in [43].

In the following discussion, $\|\cdot\|$ is an arbitrary norm and $\|\cdot\|_p$, $1 \le p \le \infty$, the l^p-norm in some space $\mathbf{R}^s$. The set $\mathbf{N}_0$ equals $\mathbf{N} \cup \{0\}$, the number $|A|$ means the cardinality of a set A, and

$$\operatorname{dist}(D, Y) := \sup_{y \in Y} \inf_{z \in D} \|y - z\|_\infty$$

is the *density* of $D \subseteq Y$ in Y. Moreover, for each $D \subseteq Y$, a problem $P[D]$ is defined by

$$P[D] \quad \begin{cases} \min & f(x) \\ \text{s.t.} & g(x, y) \le 0, \quad y \in D. \end{cases}$$

The *set of feasible points* of $P[D]$ is denoted by

$$F(D) := \{x \in \mathbf{R}^n : \ g(x, y) \le 0, \quad y \in D\}$$

and its *minimal value* by

$$\mu(D) := \inf_{x \in F(D)} f(x).$$

In case $D \subseteq Y$ is a finite and Y an infinite set, problem $P[D]$ is said to be a *discretized SIP problem*.

A point $x^* \in F(D)$ is called a *global solution* of $p[D]$ if $f(x^*) = \mu$ (D) and a *local solution* if $f(x^*) \le f(x)$ is true for all $x \in F(D) \cap U(x^*)$ with some open ball $U(x^*)$ centered at x^*. Usually convergence of algorithms to a global solution of some problem $p[D]$ can be guaranteed only for linear and convex problems. Therefore, if not specified, a 'solution' of $p[D]$ for some D in the following is a point which can be obtained in practice as the limit point of some sequence generated by an algorithm. This may be a local or a global solution of the problem or, more generally, a point at which some first order optimality condition is satisfied.

One approach to the solution of a SIP problem $p[Y]$ is to (approximately) solve $p[Y_i]$ for $i = 0, 1, \ldots$, where $\{Y_i\}$ is a sequence of finite subsets ('*grids*') of Y with

$$\lim_{i \to \infty} \operatorname{dist}(Y_i, Y) = 0.$$

A procedure of the latter type is denoted as a *discretization method*. The grid sequence $\{Y_i\}$ needed for that is usually prescribed a priori where, typically, the grids are equidistant and have the property $Y_{i+1} \supseteq Y_i$, $i \in \mathbf{N}_0$. Occasionally $\{Y_i\}$ is also successively defined a posteriori ('adaptively') such that information obtained on the ith discretization level is utilized to define Y_{i+1}.

If $x^{*i} \in F(Y_i)$ is a solution of $p[Y_i]$, $i \in \mathbf{N}_0$, and accumulation points of $\{x^{*i}\}$ solve $p[Y]$, then also accumulation points of each sequence $\{\widetilde{x}^i\}$ of approximate solutions $\widetilde{x}^i$ of $p[Y_i]$ are solutions of $p[Y]$ as long as $\lim_{i \to \infty}(\widetilde{x}^i - x^{*i}) = 0$. Thus, it has to be guaranteed that accumulation points of $\{x^{*i}\}$ solve $p[Y]$ and that the algorithm used for the solution of $p[Y_i]$ generates such point $\widetilde{x}^i$ end after finitely many iterations. Both is separately discussed below.

In practice, only a finite set of discretized problems $p[Y_i]$, $i = 0, \ldots, I$, can be solved for some $I \in \mathbf{N}$. Thus a discretization method describes a way to efficiently compute a solution of a (finely) discretized SIP problem $p[Y_I]$. Such solution serves as an approximate solution of the given SIP problem where its accuracy is usually determined by the density of Y_I in Y.

A clear advantage of discretization methods over other SIP methods is that they exclusively work with finite subsets of Y. In particular, for the finite program $p[Y_i]$, feasibility of a point $x \in \mathbf{R}^n$ can be checked easily, other than for the SIP problem $p[Y]$ itself. Therefore a discretization method is especially suited for SIP problems with a solution x^* at which $g(x^*, \cdot)$ is (almost) constant on Y or on parts of Y. The latter occurs, for example, at SIP problems originating from complex Chebyshev approximation [41].

A drawback of discretization methods is that they require a huge number of function evaluations and that the numerical costs for solving the discretized problems and hence for gaining accuracy with respect to the SIP problem increase dramatically with decreasing grid densities. Therefore, in view of reasonable computing times, the maximal number of grid points and hence the accuracy obtainable by discretization methods are limited in practice. (Grids with at most 50,000 to 100,000 points for problems with less than 100 variables are typical.) Furthermore, a solution $x^{*i} \in F(Y_i)$ of $p[Y_i]$ is normally only an *outer approximation* of a solution of the SIP problem, and hence especially an approximate solution of $p[Y]$ that is not feasible for $p[Y]$. Observe that, for $Y_i \subseteq Y$, a global solution x^{*i} of $p[Y_i]$ which is feasible for $p[Y]$ also solves $p[Y]$ since

$$f(x^{*i}) = \inf_{x \in F(Y_i)} f(x) \leq \inf_{x \in F(Y)} f(x) \leq f(x^{*i}),$$

but that such equivalence of a SIP problem with a finite optimization problem is the exceptional case [43].

The solution reached by a discretization procedure, however, often suffices for practical purposes. If this is not the case, then, under certain conditions, the solution can be improved by a locally convergent *reduction based method* [43]. Such two-phase procedures currently represent the most promising methods at least for the solution of nonlinear SIP problems.

Convergence of Solutions of Discretized SIP Problems

A solution of a discretized SIP problem is not necessarily an approximate solution of the SIP problem (see [23,43] for counterexamples). Such approximation property is only given under suitable assumptions. For that let $\{Y_i\}$ be a sequence of finite subsets of Y satisfying

$$\lim_{i \to \infty} \operatorname{dist}(Y_i, Y) = 0$$

and let

$$\Lambda(x^F, D) := F(D) \cap \left\{x \in \mathbf{R}^n : \ f(x) \leq f(x^F)\right\}$$

be a *level set* with respect to $x^F \in F(Y)$ and $D \subseteq Y$.

Then, if $Y_i \subseteq Y_{i+1} \subseteq Y$ for $i \in \mathbf{N}_0$ and $\Lambda\,(x^F, Y_0)$ is bounded for some $x^F \in F(Y)$, the following can be shown (see [42] for a more general form of this result): Problem $p[Y_i]$, $i \in \mathbf{N}_0$, possesses a global solution $x^{*i} \in \mathbf{R}^n$; moreover, $\{x^{*i}\}$ has an accumulation point, each such point is a global solution of $p[Y]$, and $\{\mu(Y_i)\}$ converges monotonically increasing to $\mu(Y)$.

Especially for convex problems, a bounded set $\Lambda(x^F, Y_0)$ exists if and only if the SIP problem $p[Y]$ has a nonempty bounded set of solutions [43]. Similar results were also proved for linear problems in [10,11,17], for convex problems in [46], and for nonlinear ones in [12]. Furthermore, for linear problems, the general possibility of discretization is studied in [4,5,6]. Some of the given theorems do not require the inclusion $Y_i \subseteq Y_{i+1}$ for all $i \in \mathbf{N}_0$, but efficient use of the obtained solution of $p[Y_i]$ as a starting vector for $p[Y_{i+1}]$ normally suggests that Y_{i+1} contains Y_i or at least those points of Y_i which belong to constraints that are active this solution.

An extension of the above convergence result guarantees convergence of solutions $x^{*\,i}$ of problems $p[D_i]$, $i \in \mathbf{N}_0$, where D_0 equals Y_0, the set D_{i+1} satisfies $D_i \cup \{y^i\} \subseteq D_{i+1} \subseteq Y_{i+1}$, and Y^i is a point with $g(x^{*i}, y^i) = \max_{y \in Y_{i+1}} g(x^{*i}, y)$ [40,42,43]. A variant of this statement concerning also nonlinear problems is derived in [30, p. 464] where $x^{*\,i}$ only needs to be a certain (approximate) stationary point of $p[D_i]$. Rules which allow to drop some of the constraints in $p[D_i]$ were given in [7,26]. But, in this case, the size of $|D_i|$ is not easily controlled and the choice of $\{Y_i\}$ not obvious ($|Y_i|$ should grow slowly if e. g. $D_{i+1} := D_i \cup \{y^i\}$).

Another convergence theorem refers to nonlinear SIP problems and the practically relevant case that $p[Y_i]$ is solved by a *sequential quadratic programming* (briefly: *SQP*) type method. For that let $f \in C^3\,(\mathbf{R}^n)$ and $g \in C^{3,\,0}(\mathbf{R}^n \times Y)$, and define, for each $x \in \mathbf{R}^n$, $\rho > 0$, and compact set $D \subseteq Y$, the *exact L_∞-penalty function*

$$L_\infty(x, \rho, D) := f(x) + \rho\, \phi^+(x, D),$$

where

$$\phi^+(x, D) := \max_{y \in D} \max\{g(x, y), 0\}.$$

A *stationary*, respectively *critical*, point of $L_\infty(\cdot, \rho, D)$ that is feasible for $p[D]$ is a *Karush–Kuhn–Tucker point* (KKT point) of $p[D]$, and, conversely, a KKT point of $p[D]$ is a stationary point of $L_\infty(\cdot, \rho, D)$ if ρ is sufficiently large (see e. g. [2,9] for details in the finite case and use the definition of a KKT point of a SIP problem from e. g. [16,17], which also applies to finite problems).

Then, if, for all $i \in \mathbf{N}_0$, there exists a stationary point x^{*i} of $L_\infty(\cdot, \rho_i, Y_i)$ for some $\rho_i > 0$ and if $\rho_i \leq \rho^*$ with some ρ^*, one has [43]: Each accumulation point x^* of $\{x^{*i}\}$ is a stationary point of $L_\infty(\cdot, \rho^*, Y)$; if $\{x^{*i_j}\}$ converges to x^* and $\lim_{j \to \infty} \phi^+(x^{*i_j}, Y_{i_j}) = 0$, then x^* also is a KKT point of $p[Y]$.

For example, the SQP type methods in [9] and [24] for the solution of a discretized SIP problem $p[Y_i]$ (see below) converge to a stationary point x^{*i} of $L_\infty(\cdot, \rho_i, Y_i)$ and a KKT point of $p[Y_i]$, respectively. The existence of an accumulation point of $\{x^{*i}\}$ is especially guaranteed for the method in [9] if a certain level set for the exact L_∞-penalty function is compact [9].

Only few rate of convergence results for a sequence of solutions of discretized SIP problems are known (cf. [30,41,43]). Note also in this regard that the numerical costs for solving discretized SIP problems normally tend to infinity with decreasing grid density. For a general theory on the discretiza- tion of SIP problems, the reader is referred to [29,30].

Solution of Discretized SIP Problems

Except for small n and $|Y_i|$, it is not advisable to solve a discretized SIP problem $p[Y_i]$ by an arbitrary method for finite programming, since such methods often require the solution of subproblems which have the same number of constraints as the problem itself and hence do not use to advantage that the constraints in a SIP problem originate from a continuous function. Moreover, if a finely discretized SIP problem $p[Y_I]$ is to be solved (the objective of a discretization method), it is much more efficient to solve $p[Y_I]$ via a sequence of problems $p[Y_i]$, $i = 0, \ldots, I$, with progressively refined grids Y_i rather than to approach $p[Y_I]$ directly. (See e. g. [9] for numerical examples showing this.) For the all-over efficiency of a discretization method it is, therefore, also important that the (approximate) solution $\widetilde{x}^i$ of $p[Y_i]$ (and possibly additional information) can be exploited for the solution of $p[Y_{i+1}]$. Such point $\widetilde{x}^i$, however, is usually infeasible for $p[Y_{i+1}]$ if $Y_{i+1} \neq Y_i$. Therefore methods which start from a feasible point of $p[Y_i]$ may turn out to be too costly within such scheme.

A variety of methods for the efficient solution of a discretized SIP problem $p[Y_i]$ has been suggested. Concerning *linear SIP problems*, in particular the following algorithm has been specified and successfully applied (let $i \in \mathbf{N}_0$ be fixed and choose $k := 0$ if $i = 0$):

0 Choose $k \in \mathbf{N}_0$ and $D_k \subseteq Y_i$ with $n \leq |D_k| < \infty$.
1 Find a solution $x^k \in \mathbf{R}^n$ of $P[D_k]$.
Let $A_k := \{y \in D_k : g(x^k, y) = 0\}$.
2 Find $y^k \in Y_i$ such that $g(x^k, y^k) = \max_{y \in Y_i} g(x^k, y)$.
3 IF $g(x^k, y^k) \leq 0$, STOP!
ELSE choose $D_{k+1} \subseteq Y_i$ with $D_{k+1} \supseteq A_k \cup \{y^k\}$.
4 Set $k := k + 1$ and go to Step 1.

If the stopping criterion of the algorithm is fulfilled, the algorithm can be continued with $i := i + 1$, beginning with Step 2 and the current index k. Especially, if $\{Y_i\}$ is a sequence of grids such that $Y_i \subseteq Y_{i+1} \subseteq Y$, if $D_0 := Y_0$ and if $\Lambda(x^F, Y_0)$ is bounded for some $x^F \in F(Y)$, then each problem $p[D_k]$ has a global solution.

It can be shown that this algorithm stops after finitely many iterations with a global solution $x^k \in F(Y_i)$ of $p[Y_i]$, provided that $\mu(D_{k+1}) = \mu(D_k)$ is true only for finitely many $k \in \mathbf{N}$ in succession [40]. The latter is guaranteed, for example, if the solution of problem $p[D_k]$ is unique [40], which is almost always the case on a computer. In practice, such solution should be computed via solution of the dual problem for $p[D_k]$, since, in that way, $p[D_{k+1}]$ can be solved very efficiently and much computing time is saved (e. g. [43]). The stopping criterion in Step 3 may be exchanged for $g(x^k, y^k) \leq \varepsilon_i$ where $\{\varepsilon_i\}$ is a zero sequence of nonnegative numbers.

Discretization procedures of this type for linear SIP problems were proposed in [14,17,40] with certain sets

$$D_{k+1} \supseteq \left\{ y \in Y_i : \; g(x^k, y) \geq \vartheta_k \right\}$$

and different choices of $\vartheta_k \leq 0$ (see also [15] for an extension to quadratic problems). The cardinality $|D_{k+1}|$ of such set D_{k+1}, however, can become quite large so that, in practice, a proper subset of D_{k+1} may have to be selected by a cumbersome management. Another promising choice of D_{k+1}, which can be motivated by the success of the methods e. g. in [8,9,28], and [36], but which has not been tried yet, would be to only add some or all violating discrete local maximizers of $g(x^k, \cdot)$ on Y_i to the set of active points A_k.

An extension of the algorithm for linear problems from [40] to *convex problems* was developed in [41,42] and applied to large filter design problems in [35] (see also [36] for computing times and a comparison with another method). As a *cutting plane method*, this latter method requires the knowledge of a compact set $X \supseteq \Lambda\,(x^F, Y_0)$ for some $x^F \in F(Y)$, which is described by finitely many linear inequality constraints. Typically, in practice, X is given by box constraints $\alpha_j \leq x_j \leq \beta_j$, $j = 1, \ldots, n$, where it is known or assumed that $p[Y]$ has a global solution which satisfies such bounds. A way to construct a bounded set X numerically without use of such a priori knowledge on the solution has been found for SIP problems which correspond to linear complex Chebyshev approximation problems [41].

An approach to finite convex programs in [13] combines an *interior point logarithmic barrier method* with a cutting plane technique, which allows to add and delete constraints and hence is capable of solving finely discretized convex SIP problems. In [20] this method has been modified and incorporated into a 'dynamic' heuristic discretization procedure for the solution of SIP problems. (See [43] for a comment on the numerical results in [20].) Further ideas concerning the solution of discretized convex SIP problems can be found in [21,22,23,47], and [48].

A number of methods is oriented towards the solution of *nonlinear discretized SIP problems*. In the 1970s, some authors had developed *combined methods of feasible directions* for nonlinear finite optimization problems, which can start from an arbitrary point in $\mathbf{R}^n$ (e. g. [34]). For SIP problems, such methods have been embedded into a discretization scheme, where convergence of certain obtainable (approximate) stationary points of $p[Y_i]$ to a related stationary point, respectively KKT point, of $p[Y]$ can be proven under relatively weak assumptions [8,28,33]. In particular, in [33], the subproblems occurring at solution of the discretized problem $p[Y_i]$ contain one constraint for each member of the entire *set of ε-most active points*

$$Y_{i,\varepsilon}^{+}(x) := \left\{ y \in Y_i : \; g(x, y) \geq \phi^{+}(x, Y_i) - \varepsilon \right\},$$

for some $\varepsilon \geq 0$, while, in [8], the special structure of a discretized SIP problem is exploited and constraints are needed only for the usually much smaller set of *discrete 'left' local maximizers* in $Y_{i,\varepsilon}^{+}(x)$. The behavior of the algorithm could be improved by addition of further points to the latter set [28]. A numerical comparison of the method in [8] with other methods is found in [45].

Using first order information only, these algorithms for $p[Y_i]$ normally have at best a linear rate of convergence. Indeed, the r-linear convergence of a modification of the method for finite problems from [34] could be shown [31], where, in this method, however, all constraints have to be respected for the construction of the quadratic subproblems. The method was incorporated into an adaptive discretization procedure such that, for certain convex problems, the entire sequence of iterates has the same r-linear rate of convergence as the inner method for the finite subproblems [32]. Another variant of the method from [34], using ε-most active constraints only, can be found in [30, p.279], but the convergence rate of this method does not seem to be known.

A combined discretization procedure and exact penalty function method using first order information was developed in [30, p. 479]. Another, normally at best linearly convergent method can be found in [18,19], where the subproblems contain constraints for all elements of the *set of ε-global points*

$$Y_{i,\varepsilon}(x) := \left\{ y \in Y_i : \; g(x, y) \geq \max_{y \in Y_i} g(x, y) - \varepsilon \right\}$$

at x with respect to Y_i for a certain $\varepsilon \geq 0$. (Note that $Y_{i,\varepsilon}(x)$ equals $Y_{i,\varepsilon}^{+}(x)$ for each point $x \in \mathbf{R}^n$ outside or on the boundary of $F(Y_i)$ and that especially $Y_{i,0}(x)$

represents the set of *discrete global maximizers* of $g(x,\cdot)$ on Y_i.) Consult, furthermore, [1] for an unconventional discretization procedure for problems with convex constraints, which generates feasible points with respect to $p[Y]$.

Some authors have developed *SQP type algorithms* for the solution of nonlinear finite optimization problems with large numbers of constraints, as they are given by discretized SIP problems. The purpose of such developments is to considerably reduce the size of the arising quadratic subproblems and the total number of gradient evaluations in comparison to standard SQP methods and to simultaneously preserve the good convergence properties of these methods (e. g. [3]). Methods of this form were given in [9,24] (with code contained in the software package CFSQP), [27,44] (without convergence proofs), and, for linearly constrained problems, in [39] (with code in [24]). A special feature of the algorithm in [24] is that the iterates remain feasible with respect to $p[Y_i]$. The peculiarity of the algorithm in [9] is that the quadratic subproblem at the kth iteration only needs to include constraints for the usually small *set of discrete ϵ-global local maximizers*

$$Y_{i,\varepsilon}^{l}(x^k) := \left\{\widetilde{y} \in Y_{i,\varepsilon}(x^k):\ g(x^k,\widetilde{y}) \geq g(x^k, y), y \in U_i(\widetilde{y})\right\},$$

where $U_i(\widetilde{y})$ is a *discrete neighborhood* of $\widetilde{y}$ in Y_i, consisting of $\widetilde{y}$ and neighboring points of $\widetilde{y}$ in Y_i (see also [43]). As an example of such SQP type algorithms for the solution of a discretized SIP problem $p[Y_i]$, the algorithm from [9] is given below in a rudimentary form (again let $i \in \mathbf{N}_0$ be fixed and choose $k := 0$ if $i = 0$).

Convergence of this algorithm to a stationary point of the exact L_∞-penalty function, respectively a KKT point, of $p[Y_i]$ is guaranteed under standard assumptions. If, in addition, the Maratos effect avoiding scheme from [25] is properly incorporated into the algorithm, then, under some additional assumptions usually required in this context, it also converges r-superlinearly. The final data obtained by the algorithm (with an adequate stopping criterion) can normally be completely employed to initialize the algorithm for $p[Y_{i+1}]$, in case a sequence of discretized SIP problems is solved.

0 Select $\alpha \in (0, 1/2)$, $\beta \in (0,1)$, $\varepsilon > 0$, and $k \in \mathbf{N}_0$.
Choose $\rho_k > 0$, $x^k \in \mathbf{R}^n$, a symmetric positive definite matrix $H_k \in \mathbf{R}^{n\times n}$, and a subset $D_k \subseteq Y_i$ with $D_k \supseteq Y_{i,\varepsilon}^{l}(x^k)$.

1 Compute the unique solution $(d^k, \xi_k) \in \mathbf{R}^n \times \mathbf{R}$ of the quadratic problem

$$\begin{cases} \min & \frac{1}{2} d^\top H_k d + \nabla f(x^k)^\top d + \rho_k \xi \\ \text{s.t.} & g(x^k, y) + \nabla_x g(x^k, y)^\top d \leq \xi, \\ & y \in D_k, \\ & \xi \geq 0, \end{cases}$$

and associated Lagrange multipliers $(\lambda^k, \tilde{\lambda}_k) \in \mathbf{R}^{|D_k|} \times \mathbf{R}$.
IF $\| d^k \| = 0$, STOP|

2 IF $\xi_k = 0$ and $\| \lambda^k \|_1 \geq \rho_k$, set $\rho_k := 1.5 \| \lambda^k \|_1$.

3 Let $\ell \in \mathbf{N}_0$ be the smallest number such that $t_k := \beta^\ell$ satisfies

$$L_\infty(x^k, \rho_k, Y_i) - L_\infty(x^k + t_k d^k, \rho_k, Y_i) \geq \alpha t_k {d^k}^\top H_k d^k$$

Set $x^{k+1} := x^k + t_k d^k$ and $\rho_{k+1} := \rho_k$.

4 Compute H_{k+1} by Powell's modification of the BFGS update ([37]) with respect to the Hessian of the Lagrangian

$$L_i(x, \lambda(Y_i)) := f(x) + \sum_{y \in Y_i} \lambda(y) g(x, y)$$

of $P[Y_i]$, where $\lambda(Y_i) := (\lambda(y))_{y\in Y_i}$, and choose a set $D_{k+1} \subseteq Y_i$ with $D_{k+1} \supseteq Y_{i,\varepsilon}^{l}(x^{k+1})$.
Set $k := k + 1$ and go to Step 1.

Applied to a discretized SIP problem, the algorithm of [9] improves another one from [2, Sect. 4.2] which, instead of $Y_{i,\varepsilon}^{l}(x^k)$, employs the in this case usually much larger, upper set $Y_{i,\varepsilon}(x^k)$. But, although the choice $D_k := Y_{i,\varepsilon}^{l}(x^k)$ is suitable in the above algorithm, addition of further points to D_k is advised to improve the all-over efficiency of the method. In this regard, the selection rules from [24] and [44] have turned out to be valuable.

The SQP algorithm in [9] can be similarly used also for the solution of the *locally reduced problem* in a second phase of a two-phase approach to the solution of

$p[Y]$ and has shown to yield excellent results in this way. At earlier two-phase approaches, always two different methods had been applied to the problems in both phases.

In addition to the mentioned methods, also *stochastic discretization procedures* have been developed which are capable of providing a *quasi-optimal solution* of a finely discretized SIP problem (e. g. [49]).

See also

- ► Semi-infinite Programming: Approximation Methods
- ► Semi-infinite Programming and Control Problems
- ► Semi-infinite Programming: Methods for Linear Problems
- ► Semi-infinite Programming: Numerical Methods
- ► Semi-infinite Programming: Second Order Optimality Conditions
- ► Semi-infinite Programming, Semidefinite Programming and Perfect Duality

References

1. Ašić MD, Kovačević-Vujčić VV (1988) An interior semi-infinite programming method. J Optim Th Appl 59:353–367
2. Bertsekas DP (1996) Constrained optimization and Lagrange multiplier methods. Athena Sci, Belmont
3. Boggs PT, Tolle JW (1995) Sequential quadratic programming. Acta Numer 4:1–51
4. Goberna MA, López MA (1987) Reduction and discrete approximation in linear semi-infinite programming. Optim 18:643–658
5. Goberna MA, López MA (1988) Optimal value function in semi-infinite programming. J Optim Th Appl 59:261–279
6. Goberna MA, López MA (1998) Linear semi-infinite optimization. Wiley, New York
7. Gonzaga C, Polak E (1979) On constraint dropping schemes and optimality functions for a class of outer approximations algorithms. SIAM J Control Optim 17:477–493
8. Gonzaga C, Polak E, Trahan R (1980) An improved algorithm for optimization problems with functional inequality constraints. IEEE Trans Autom Control AC-25:49–54
9. Görner S (1997) Ein Hybridverfahren zur Lösung nichtlinearer semi-infiniter Optimierungsprobleme. PhD Thesis, Techn Univ Berlin, Berlin, Germany
10. Grigorieff R, Reemtsen R (1990) Discrete approximations of minimization problems. II Applications. Numer Funct Anal Optim 11:721–761
11. Gustafson S-A (1979) On numerical analysis in semi-infinite programming. In: Hettich R (ed) Semi-Infinite Programming. Springer, Berlin, pp 51–65
12. Gustafson S-A (1983) A three-phase algorithm for semi-infinite programs. In: Fiacco AV, Kortanek KO (eds) Semi-Infinite Programming and Applications. Springer, Berlin, pp 138–157
13. Den Hertog D, Kaliski J, Roos C, Terlaky T (1995) A logarithmic barrier cutting plane method for convex programming. Ann Oper Res 58:69–98
14. Hettich R (1986) An implementation of a discretization method for semi-infinite programming. Math Program 34:354–361
15. Hettich R, Gramlich G (1990) A note on an implementation of a method for quadratic semi-infinite programming. Math Program 46:249–254
16. Hettich R, Kortanek KO (1993) Semi-infinite programming: theory, methods, and applications. SIAM Rev 35:380–429
17. Hettich R, Zencke P (1982) Numerische Methoden der Approximation und semi-infiniten Optimierung. Teubner, Leipzig
18. Huth M (1987) Superlinear konvergente Verfahren zur Lösung semi-infiniter Optimierungsaufgaben - Eine Hybridmethode. PhD Thesis, Pädagogische Hochschule Halle, Halle, GDR
19. Huth M, Tichatschke R (1990) A hybrid method for semi-infinite programming problems. In: Rieder U et al (eds) Methods Oper Res 62. Anton Hain Verlag, Meisenheim, pp 79–90
20. Kaliski J, Haglin D, Roos C, Terlaky T (1997) Logarithmic barrier decomposition methods for semi-infinite programming. Internat Trans Oper Res 4:285–303
21. Kaplan A, Tichatschke R (1992) A regularized penalty method for solving convex semi-infinite programs. Optim 26:215–228
22. Kaplan A, Tichatschke R (1993) Regularized penalty methods for semi-infinite programming problems. In: Brosowski B, Deutsch F, Guddat J (eds) Approximation & Optimization. P Lang, Frankfurt am Main, pp 341–356
23. Kaplan A, Tichatschke R (1994) Stable methods for ill-posed variational problems. Akademie, Berlin
24. Lawrence CT, Tits AL (1998) Feasible sequential quadratic programming for finely discretized problems from SIP. In: Reemtsen R and Rückmann J-J (eds) Semi-Infinite Programming. Kluwer, Dordrecht, pp 159–193
25. Mayne DQ, Polak E (1982) A superlinearly convergent algorithm for constrained optimization problems. Math Program Stud 16:45–61
26. Mayne DQ, Polak E, Trahan R (1979) An outer approximations algorithm for computer-aided design problems. J Optim Th Appl 28:331–352
27. Mine H, Fukushima M, Tanaka Y (1984) On the use of ϵ-most-active constraints in an exact penalty function method for nonlinear optimization. IEEE Trans Autom Control AC-29:1040–1042

28. Panier ER, Tits AL (1989) A globally convergent algorithm with adaptively refined discretization for semi-infinite optimization problems arising in engineering design. IEEE Trans Autom Control 34:903–908
29. Polak E (1993) On the use of consistent approximations in the solution of semi-infinite optimization and optimal control problems. Math Program 62:385–414
30. Polak E (1997) Optimization. Algorithms and consistent approximations. Springer, Berlin
31. Polak E, He L (1991) Unified steerable phase I-phase II method of feasible directions for semi-infinite optimization. J Optim Th Appl 69:83–107
32. Polak E, He L (1992) Rate-preserving discretization strategies for semi-infinite programming and optimal control. SIAM J Control Optim 30:548–572
33. Polak E, Mayne DQ (1976) An algorithm for optimization problems with functional inequality constraints. IEEE Trans Autom Control AC-21:184–193
34. Polak E, Trahan R, Mayne DQ (1979) Combined phase I-phase II methods of feasible directions. Math Program 17:61–73
35. Potchinkov A, Reemtsen R (1994) FIR filter design in the complex domain by a semi-infinite programming technique I-II. Arch Elektronik und Übertragungstechnik 48:135–144; 200–209
36. Potchinkov A, Reemtsen R (1995) The design of FIR filters in the complex plane by convex optimization. Signal Processing 46:127–146
37. Powell MJD (1978) The convergence of variable metric methods for nonlinearly constrained optimization calculations. In: Mangasarian OL, Meyer RR, Robinson SM (eds) Nonlinear Programming 3. Acad Press, New York, pp 27–63
38. Powell MJD (1989) A tolerant algorithm for linearly constrained optimization calculations. Math Program 45:547–566
39. Powell MJD (1989) TOLMIN: A Fortran package for linearly constrained optimization calculations. Techn Report, DAMTP Univ Cambridge NA2
40. Reemtsen R (1991) Discretization methods for the solution of semi-infinite programming problems. J Optim Th Appl 71:85–103
41. Reemtsen R (1992) A cutting plane method for solving minimax problems in the complex plane. Numer Algorithms 2:409–436
42. Reemtsen R (1994) Some outer approximation methods for semi-infinite optimization problems. J Comput Appl Math 53:87–108
43. Reemtsen R, Görner S (1998) Numerical methods for semi-infinite programming: A survey. In: Reemtsen R, Rückmann J-J (eds) Semi-Infinite Programming. Kluwer, Dordrecht, pp 195–275
44. Schittkowski K (1992) Solving nonlinear programming problems with very many constraints. Optim 25:179–196
45. Tanaka Y, Fukushima M, Ibaraki T (1988) A comparative study of several semi-infinite nonlinear programming algorithms. Europ J Oper Res 36:92–100
46. Tichatschke R (1979) Stetigkeitseigenschaften und Konvergenz von Folgen diskretisierter semi-infiniter konvexer Optimierungsaufgaben. Wiss Z TH Karl-Marx-Stadt 21:577–586
47. Tichatschke R (1985) Semi-infinite programming problems. Banach Center Publ 14:543–554
48. Tichatschke R, Schwartz B (1982) Methods of feasible directions for semi-infinite programming problems I-II. Wiss Inform TH Karl-Marx-Stadt 33:1–15; 16–23
49. Volkov YV, Zavriev SK (1997) A general stochastic outer approximations method. SIAM J Control Optim 35:1387–1421

Semi-infinite Programming: Methods for Linear Problems

Sven-Åke Gustafson
Stavanger University, Stavanger, Norway

MSC2000: 90C34, 90C05

Article Outline

Keywords
Definition of Linear Semi-Infinite Programs
A Global Maximization Problem
An Air Pollution Control Problem
Concluding Remarks
See also
References

Keywords

Primal (linear) semi-infinite program; Dual semi-infinite program; Global maximization problem; Pseudorandom; Curse of dimensionality; Air pollution; Source class; generalized Gauss quadrature rule

The class of linear semi-infinite programming problems may be looked upon as a generalization of the class of linear programs. In a semi-infinite program, either the number of variables or the number of constraints (but not both at the same time) may be infinite. In the present paper we will mainly deal with semi-infinite programs of the latter type. The main theoretical as well

as practical difficulty is that one needs to verify that a proposed optimal solution satisfies infinitely many linear inequality constraints. We will describe two applications which illustrate this, namely global optimization and an air quality control problem. In a companion paper in this volume we will describe how to solve semi-infinite programs numerically by means of systematic approximation with simpler problems. The literature on linear semi-infinite programs is vast. For a general introduction to this field, see [5,6] and [7]. Recent results are to be found for instance, in [11].

Definition of Linear Semi-Infinite Programs

We now introduce, following [6]:

Definition 1 Let S be a fixed set, a: $S \to \mathbf{R}^n$, b: $S \to \mathbf{R}$ two fixed functions, $c \in \mathbf{R}^n$ a fixed vector. Then the following task will be called a *primal linear semi-infinite program*:

$$\min c^\top y, \tag{1}$$

over all $y \in \mathbf{R}^n$ subject to the constraint

$$a(s)^\top y \geq b(s), \quad s \in S. \tag{2}$$

Remark 2 The data of the semi-infinite program (1) and (2) are hence the index set S, the vector-valued function a and the real-valued function b, both defined on S as well as the fixed vector $c \in \mathbf{R}^n$.

Definition 3 Use the notation of Definition 1. Put

$$Y = \left\{ y \in \mathbb{R}^n : \; a(s)^\top y \geq b(s], \; s \in S \right\}, \tag{3}$$

$$V = \inf_{y \in Y} c^\top y. \tag{4}$$

If Y is empty, then we define $V = -\infty$, i. e. the condition (2) is inconsistent. Y is called the *set of feasible solutions* to (1), (2).

The following theorem on dual inequality holds:

Theorem 4 *Use the notations of Definition 1. Assume that there is a subset $\{s_1, \ldots, s_q\} \subset S$, such that*

$$\sum_{i=1}^{q} x_i a(s_i) = c, \qquad x_i \geq 0, \quad i = 1, \ldots, q. \tag{5}$$

Then the following inequality holds for all y satisfying (2):

$$c^\top y \geq \sum_{i=1}^{q} x_i b(s_i).$$

Proof Let (2) and (5) be satisfied. Then

$$a(s_i)^\top y \geq b(s_i), \quad i = 1, \ldots, q.$$

Multiplying by x_i and summing over i we obtain

$$\sum_{i=1}^{q} x_i a(s_i)^\top y \geq \sum_{i=1}^{q} x_i b(s_i). \tag{6}$$

The result now follows from (5).

Remark 5 The right-hand side of (6) gives a lower bound for the optimal value of the linear semi-infinite program defined by (1) and (2).

We next consider the task of finding the largest value of this lower bound:

Definition 6 The *dual semi-infinite program* is defined by: Determine a finite subset $\{s_1, \ldots, s_q\} \subset S$ and real numbers $x_1, \ldots, x_q$ such that the expression

$$\sum_{i=1}^{q} x_i b(s_i) \tag{7}$$

is maximized, subject to the constraints

$$\sum_{i=1}^{q} x_i a(s_i) = c, \qquad x_i \geq 0, \quad i = 1, \ldots, q. \tag{8}$$

Remark 7 If S is a finite set, then (1), (2) define a linear program and (7), (8) is equivalent to its dual.

The problems in Definitions 1 and 6 above admit several different geometric interpretations. Thus (8) means that the vector c shall be written as a nonnegative linear combination of the q vectors $a(s_1), \ldots, a(s_q)$. The inequality (2) means that the real-valued function b is to be approximated from the above over S by a linear combination of the n functions $a_1, \ldots, a_n$.

We note that the feasible subset Y defined by (3) is a convex subset of $\mathbf{R}^n$, if it is nonempty. For each fixed $s \in S$ the relation

$$a(s)^\top y = b(s)$$

defines a supporting hyperplane to Y. Hence program (1), (2) is the task to minimize the linear form $c^\top y$ over Y. The latter problem may be formally rewritten as the equivalent task of minimizing

$$c^\top y,$$

subject to the single constraint

$$g(y) \geq 0,$$

where

$$g(y) = \min_{s \in S} a(s)^\top y - b(s).$$

Thus the evaluation of $g(y)$ requires the solution of a global minimization problem, which may be nontrivial if S is infinite. Under fairly general assumptions the duality results of linear programming may be extended to semi-infinite programs. In particular, the programs (1), (2) and (7), (8) have optimal solutions and the same optimal value. See e. g. [5] and [6]. We state the following theorem on *complementary slackness*:

Theorem 8 *Let y be an optimal solution of (1), (2) and let q, $\{s_1, \ldots, s_q\} \in S$, $x_1, \ldots, x_q$ be an optimal solution of (7), (8). Put*

$$d(s) = a(s)^\top y - b(s), \quad s \in S.$$

If the two problems also have the same optimal value, the following relations hold:

$$d(s) \geq 0, \quad s \in S, \tag{9}$$

$$\sum_{i=1}^{q} x_i a(s_i) = c, \tag{10}$$

$$x_i d(s_i) = 0, \quad i = 1, \ldots, q, \tag{11}$$

the function d achieves its global minimum at

$$s_i, \quad i = 1, \ldots, q. \tag{12}$$

The proof is given e. g. in [6].

Remark 9 The relations (9)–(12) give necessary conditions for optimality. If the set S is finite, then (1), (2) and (7), (8) form a dual pair of linear programs and we recognize Theorem 8 in this case as the complementary slackness result in linear programming.

If S is infinite, then (10), (11) may be combined to a nonlinear system of equations with x_i, s_i, $i = 1, \ldots, q$, and y as unknown variables. Further relations may be derived from (12) and included in the system which then has equally many scalar unknowns as it has equations. This requires that q is known and one must also require that the function d has continuous partial derivatives of the first order. In addition it must be known for each s_i whether this point is at the boundary or in the interior of S. These considerations are used in the three-phase algorithm. See e. g. [5,6] or [7], whose main ideas are as follows:

1	Solve the problems (1), (2) and (7), (8) approximately by replacing S with a finite subset T to obtain a dual pair of linear programs.
2	Determine q and approximate values for the variables x_i, s_i, $i = 1, \ldots, q$, and y from the results of Phase 1. Decide for each s_i whether it is an interior or boundary point of S. Construct a nonlinear system of equations with respect to the variables sought.
3	Solve the nonlinear system of equations by a suitable numerical scheme. Check the feasibility of the calculated solution, in particular that the infinitely many constraints in (2) are satisfied. In case of failure return to Phase 1 and redo the whole process with a finer grid.

A Global Maximization Problem

We discuss now a very special instance of the problem (1), (2). The treatment of this test problem illustrates some fundamental properties of semi-infinite programs. Thus we put $n = 1$, $a_1(s) = 1$, $c_1 = 1$ and consider the task

Example 10

$$\min y, \tag{13}$$

subject to the constraint

$$y \geq b(s), \quad s \in S. \tag{14}$$

In this example y is a real number. We note that (14) may define infinitely many constraints. The dual of the problem (13), (14) may be written

$$\max b(s), \quad s \in S. \tag{15}$$

By placing various assumptions on the function b and the set S one may illustrate the different properties a dual pair of semi-infinite programs may have. Thus if S is finite, then both programs achieve their optimal values in accordance with the theory of linear programs. If b is bounded on S, then the primal program (13), (14) has the optimal solution

$$y = \sup_{s \in S} b(s). \tag{16}$$

If in addition S is compact and b continuous, then the dual (15) also achieves its optimal value and we write

$$y = \max_{s \in S} b(s). \tag{17}$$

For a discussion of the various states of semi-infinite programs, see [5,6] and [7].

Example 10 may also be used to illustrate the properties of some suggested numerical schemes for linear semi-infinite programs. We note that (17) is useful for numerical work only if there is a way to determine all local maxima to b on S. If b has continuous partial derivatives of the first order, we may try to solve by numerical methods the equation

$$\nabla b(s) = 0. \tag{18}$$

In addition one would need to study the values of b at the boundary of S.

We discuss now the three-phase algorithm as described in [3,5,6,7,9,10]. When applied to the problem (13), (14) this scheme becomes:

1	Let T_ℓ, $\ell = 1, 2, \ldots$, be a sequence of grids (subsets to S) such that each T_ℓ contains finitely many points and $\lim_{\ell \to \infty} \max_{s \in S} \min_{t \in T_\ell} \|s - t\| = 0.$ (19) Calculate an s_ℓ which solves $\max_{t \in T_\ell} b(t).$ (20)
2	Take s_ℓ as starting value.
3	To determine the local maxima of b with some numerical scheme.

Example 11 We consider the following special case of Example 10:

$$\min b(s) = \sin \frac{1}{s + 0.000001}, \quad s \in [0, 1].$$

In this case there are many local maxima and the grid would need to be very fine, if the three-phase method should locate them all. A direct analytic treatment would of course be easy, since the expression for b is simple but the example illustrates potential difficulties.

We note that if S has several dimensions, then the 'curse of dimensionality' sets in, making a systematic discretization approach inefficient, since it implies that b must be tabulated and this table must be large in order to describe b. In a practical situation there is no assurance that the optimal solution has been found, if this cannot be tested analytically. Since one has to accept the possibility that a nonoptimal value has been accepted one may consider pseudorandom generation of grids instead of the deterministic approaches described above and in the references given up to now. A major advantage of these approaches is that the computational complexity is not crucially dependent on the number of dimensions and statistical estimates for the uncertainty in the optimal value may be developed. For an introduction to global optimization methods based on random processes, see [12]. The author is working on methods for applying random schemes for the general semi-infinite program (1), (2).

An Air Pollution Control Problem

The three-phase approach, described above may be directly applied to general linear semi-infinite programs. We illustrate with the air pollution control problem as described on [6, p. 17; 184]. Here the air quality control area is represented by the compact set $A \subset \mathbf{R}^2$. The annual mean concentration of a certain inert pollutant (e. g. sulphur dioxide, SO_2) is represented by a real-valued function p, defined on A. There are many sources emitting pollution and each car, house or power plant may contribute to the pollution. The permissible level of pollution is defined by a given function v. If there is a point $s \in A$ such that the standard v is exceeded, i. e. is such that

$$p(s) > v(s), \tag{21}$$

then the pollution concentration must be reduced. After reduction, the remaining pollution must satisfy the standard at each point $s \in A$ and this fact implies that infinitely many inequality constraints must be met. We will assume that the sources of pollution may be split

into $n+1$ classes and that the sources in each class will be regulated in the same way. We further require that the superposition principle is valid, i. e. that the concentration contributions add up. Therefore we write

$$p(s) = \sum_{r=0}^{n} u_r(s), \quad s \in A, \tag{22}$$

where the concentration contribution from source-class r is given by the nonnegative function u_r. In particular, u_0 represents the concentration contribution from background sources which cannot be regulated. Next, u_1 may be the contribution from trucks, operating in A, u_2 from passenger cars, u_3 from residential heating and so on. The implementation of an abatement policy implies that the contribution from class r is reduced by the fraction E_r, hence the remaining concentration contribution from this class is

$$(1 - E_r)u_r(s), \quad 0 \le E_r \le 1.$$

The reduction vector $E \in \mathbf{R}^n$ must therefore satisfy the constraints

$$0 \le E_r \le 1, \quad r = 1, \ldots, n,$$

$$u_0(s) + \sum_{r=1}^{n} (1 - E_r)u_r(s) \le v(s), \quad s \in A.$$

The last relation may also be written

$$\sum_{r=1}^{n} E_r u_r(s) \ge p(s) - v(s), \quad s \in A,$$

which may be interpreted as the requirement that the total removed pollution should amount at least to the difference between the original pollution and the required standard $v(s)$.

Assume that the cost associated with the pollution abatement is given by the linear form $K(E)$ where

$$K(E) = \sum_{r=1}^{n} c_r E_r.$$

The task to determine E such that the air quality standard v is satisfied at each $s \in S$, while the control cost is minimized is hence the solution to the problem: Determine

$$V = \min_{E \in \mathbb{R}^n} \sum_{r=1}^{n} c_r E_r, \tag{23}$$

subject to the constraints

$$\sum_{r=1}^{n} E_r u_r(s) \ge p(s) - v(s), \quad s \in A, \tag{24}$$

$$E_r \ge 0, \quad r = 1, \ldots, n, \tag{25}$$

$$-E_r \ge -1, \quad r = 1, \ldots, n. \tag{26}$$

Thus the equations (23)–(26) define a linear semi-infinite program of the type (1) and (2). We note that if there is a point $s^* \in A$, such that $u_0(s^*) > v(s^*)$, then the background pollution is above the standard and there is no control policy achieving the desired goal. Hence the program (23)–(26) is inconsistent. Otherwise we conclude the existence of an optimal policy if the functions u_r, $r = 0, \ldots, n$ are continuous. For the purpose of numerical treatment we replace the area A by a finite grid T in (24) and solve the corresponding linear program. In so doing, we enforce the standard at the finite grid only and hence it may be violated somewhere between the gridpoints. However, convergence obtains if we consider a sequence of finer and finer grids as in (19). It is also possible to derive a nonlinear system and use the three-phase algorithm. It is of interest here that Theorem 8 implies that if an optimal control policy has been calculated, there is a subset of points where the standard is met exactly. It has been proposed to put air quality sampling stations at these points. For further details, see [6].

Concluding Remarks

There are a large number of other applications of semi-infinite programming. One such is the approximation of functions in the maximum norm. Special algorithms have been developed for classes of such applications. For the approximation in the maximum norm, the algorithms of Remez (see [2]) are applicable. But we note also here that a global maximization needs to be carried out at each step of the algorithm. An one-sided approximation problem has been studied by [1]. The solutions of the dual of this problem are associated with quadrature rules of generalized Gauss type. However, the determination of abscissaæ and weights of such rules, using only the parameters of the corresponding semi-infinite program, is notoriously ill-conditioned. For the so-called classical cases associated with the names of Legendre, Hermite and Jacobi, other analytic informa-

tion is used, for instance, the coefficients of the three-term recurrence relation formulas for corresponding orthogonal polynomials. See e. g. [4] and [8]. These methods give an alternative way of finding the solutions of certain special semi-infinite programs which may be used for testing the accuracy and efficiency of general methods for semi-infinite programs.

See also

- ▶ Semi-infinite Programming: Approximation Methods
- ▶ Semi-infinite Programming and Control Problems
- ▶ Semi-infinite Programming: Discretization Methods
- ▶ Semi-infinite Programming: Numerical Methods
- ▶ Semi-infinite Programming: Second Order Optimality Conditions
- ▶ Semi-infinite Programming, Semidefinite Programming and Perfect Duality

References

1. Bojanic R, Devore R (1966) On polynomials of best one-sided approximation. L'Enseign Math 12(2):139–164
2. Cheney EW (1966) Introduction to approximation theory. McGraw-Hill, New York
3. Fiacco AC, Kortanek KO (eds) (1983) Semi-infinite programming and applications. Lecture Notes Economics and Math Systems. Springer, Berlin
4. Gautschi W (1970) On the construction of Gaussian quadrature rules from modified moments. Math Comput 24:245–260
5. Glashoff K, Gustafson S-Å (1978) Einführung in die Lineare Optimierung. Wissenschaftl Buchgesellschaft Darmstadt, Darmstadt
6. Glashoff K, Gustafson S-Å (1983) Linear optimization and approximation. Springer, Berlin
7. Goberna MA, López MA (1998) Linear semi-infinite optimization. Wiley, New York
8. Golub GH, Welsch JH (1969) Calculation of Gauss quadrature rules. Math Comput 23:221–230
9. Gustafson S-Å (1983) A three phase algorithm for semi-infinite programs. In: Fiacco AC, Kortanek KO (eds) Semi-Infinite Programming and Applications. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 138–157
10. Gustafson S-Å, Kortanek KO (1973) Numerical treatment of a class of semi-infinite programming problems. Naval Res Logist Quart 20:477–504
11. Reemtsen R, Rückmann JJ (eds) (1998) Semi-infinite programming. Kluwer, Dordrecht
12. Törn A, Žilinskas A (1989) Global optimisation. Lecture Notes Computer Sci, vol 350. Springer, Berlin

Semi-infinite Programming: Numerical Methods

SIP

Rainer Hettich, Alexander Kaplan, Rainer Tichatschke
University Trier, Trier, Germany

MSC2000: 90C34, 90C26, 90C25, 90C34

Article Outline

Keywords
Exchange Methods
Discretization Methods
Reduction Methods
See also
References

Keywords

Semi-infinite optimization; Approximation; Numerical algorithms

The problem

$$\mathcal{P}(T) \quad \begin{cases} \min & \{F(z)\colon\ z \in K\}\,, \\ K = & \{z \in \mathbf{R}^n\colon\ g(z,t) \le 0,\ \forall t \in T\}\,, \end{cases} \tag{1}$$

is considered, where T is a compact subset of the space $\mathbf{R}^m$, F and $g(\cdot, t)$ are continuous functions from $\mathbf{R}^n$ into $\mathbf{R}$ and $g(z, \cdot)$ is continuous on T. If the functions F and $g(\cdot, t)$ are assumed to be convex, we deal with convex semi-infinite problems, and many important applications lead to linear semi-infinite problems, with linear functions F and $g(\cdot, t)$.

With some additional writing effort, the content of this article can be easily extended to the case that the set of feasible points is given as the intersection of finitely many sets

$$\{z \in \mathbf{R}^n\colon\ g_s(z,t) \le 0, \quad \forall t \in T^s\}\,, \quad s \in S,$$

where, for each $s \in S$, T^s is a compact subset of $\mathbf{R}^{m_s}$ and g_s has the same properties as g.

The notion 'semi-infinite programming problem' is due to the property that the feasible set K belongs to a finite-dimensional space whereas T is an infinite set.

The references here are not primarily intended to give a complete documentation of the original papers and sources or to reflect the historical development of this field. Except for the case that reference is given to explicit statements or techniques, it was our aim to draw the reader's attention to the basic approaches.

Concerning theory and properties of semi-infinite programming, see the monographs [1,19,24], papers [12,18] and to the proceeding volumes [6,33].

Transforming $g(z, t) \leq 0$, $\forall t \in T$, into $\max_{t \in T} g(z, t) \leq 0$, one could treat semi-infinite problems in the framework of nondifferentiable optimization. We do not adopt this approach here (see the review paper [27]).

The algorithms for semi-infinite problems generate usually a sequence of finitely constrained auxiliary optimization problems suitable for being solved by standard algorithms for finite optimization. We assume the latter to be available. We concentrate on the different ways these auxiliary problems may be generated and confine ourselves to some remarks on properties the finite optimization algorithms should favorably have. Roughly one can distinguish three classes of methods according to the way the auxiliary (finite) problems are generated: Exchange methods, including cutting plane methods (applicable only for convex problems), discretization methods and methods based on local reduction. Exchange and discretization methods are more recommendable for a 'first phase' of the solution process, whereas the reduction approach is recommendable for a final stage in order to provide a higher accuracy of the solution and a better rate of convergence.

In order to sketch briefly the respective ideas behind the exchange and discretization methods, we suppose that K is a compact set and introduce the finitely constrained auxiliary problems

$$\mathcal{P}(T_i) \quad \begin{cases} \min & \{F(z)\colon z \in K_i\}, \\ K_i = & \{z \in Z_0\colon g(z,t) \leq 0,\ \forall t \in T_i\}, \end{cases} \tag{2}$$

with Z_0 a convex compact set, $Z_0 \supset K$. Sometimes Z_0 is artificially introduced in order to provide solvability of these problems. If $K \neq \emptyset$, the approximate problem (2) has a solution for every finite set $T_i \subset T$.

Omitting detailed assumptions on the problems and methods involved, we concentrate on the main ideas of the numerical approaches mentioned above.

Exchange Methods

This notion refers to the fact that in step i the set K_{i+1} is obtained from K_i by addition of at least one new constraint and (in many algorithms) deletion of some of the constraints of K_i, i. e., an exchange of constraints takes place.

- Step i: Given $T_i \subset T$, $|T_i| < \infty$. Compute (approximately) a solution z^i of $\mathcal{P}(T_i)$ and some (or all) local maxima $t_1^i, \ldots, t_{q_i}^i$ of the subproblem

$$\mathcal{Q}(z^i) \quad \max \left\{ g(z^i, t)\colon\ t \in T \right\}.$$

Stop if $g(z^i, t_j^i) \leq 0$, $j = 1, \ldots, q_i$. Otherwise, choose T_{i+1} such that the following relations hold:

$$T_{i+1} \subset T_i \cup \{t_1^i, \ldots, t_{q_i}^i\},$$
$$\max_{t \in T_{i+1}} g(z^i, t) > 0.$$

Necessary for convergence is

$$\max_{j=1,\ldots,q_i} g(z^i, t_j^i) = \max_{t \in T} g(z^i, t), \tag{3}$$

i. e. the global solution of the nonconvex optimization problem $\mathcal{Q}(z^i)$ is required in each step (or at least in a subsequence of steps) which for $\dim T \geq 2$ may be very costly. Under (3) convergence may be proved for instance in case that constraints are never deleted from T_i (cf. [18]).

In fact, exchange methods are similar to the classical REMES-algorithm for solving linear Chebyshev approximation problems. Special algorithms mainly differ in the choice of T_{i+1}. In the literature, cf., e. g., [14,23,30,40,41], there are different algorithms which have important additional features: The ability to add several constraints $g(z, t^i) \leq 0$, where t^i are some points with $g(z^i, t^i) > 0$, (or to add only the constraint which is maximally violated at z^i) and to delete constraints in an efficient way in the case of a convex problem (1). In [23] a cutting plane algorithm for convex problems is suggested, which ensures a linear rate of convergence with respect to the values of the objective functional.

Discretization Methods

Algorithms of this type compute a solution of problem (1) by solving a sequence of problems $\mathcal{P}(T_i)$, where T_i is an h_i-grid on T, i. e., a finite subset $T_i \subset T$ such that $\sup_{t \in T} \operatorname{dist}(t, T_i) \leq h_i$. In the ith step a fixed (usually,

uniform) grid T_i is considered, generated by $h_i = \gamma_i h_{i-1}$, with $\gamma_i \in (0, 1)$ chosen a priorily or defined by the solution procedure itself. In step i only subsets $\overline{T}_i$ of T_i are used and in a number of algorithms one has ($k_i \in \mathbf{N}$):

$$\gamma_i = \frac{1}{k_{i-1}} \quad \text{with} \quad k_{i-1} \geq 2$$

implying $T_{i-1} \subset T_i \; \forall i$.

- Step i: Given h_{i-1}, the last set $\overline{T}_{i-1} \subset T_{i-1}$ and a solution z^{i-1} of problem $\mathcal{P}(\overline{T}_{i-1})$;

  i1) choose $h_i = \gamma_i h_{i-1}$ and generate T_i;
  i2) select $\overline{T}_i \subset T_i$;
  i3) compute a solution $\overline{z}$ of $\mathcal{P}(\overline{T}_i)$. If $\overline{z}$ is feasible for $\mathcal{P}(T_i)$ within a given accuracy, put $z^i := \overline{z}$ (define also γ_{i+1} if the sequence $\{\gamma_k\}$ is not a priorily chosen) and continue with Step$(i+1)$. Otherwise repeat i2) for a new choice of $\overline{T}_i$, enlarging the old one.

With regard to efficiency, it is important to use as much information as possible from former grids in solving $\mathcal{P}(\overline{T}_i)$. Substantial differences among the algorithms of this type are in the choice of $\overline{T}_i$. Preferably $\overline{T}_i$ should be chosen such that the solution of $\mathcal{P}(\overline{T}_i)$ approximately solves also $\mathcal{P}(T_i)$. Most of the suggested algorithms select $\overline{T}_i$ such that

$$\overline{T}_i \supset T_i^\alpha = \{t \in T_i : \; g(\overline{z}, t) \geq -\alpha\},$$

with $\alpha > 0$ some chosen threshold and $\overline{z}$ the foregoing inner iterate (cf. i3)).

The choice of α is crucial (cf. [15,31,32]): A large value of α leads to many constraints in $\mathcal{P}(\overline{T}_i)$ and choosing α too small one should expect a large number of steps i2), i3) for fixed T_i. Adaptive strategies for decreasing α are used in [7] and [26].

In [39] special versions of the method of feasible directions coordinate the choice of the grid and the search direction in order to obtain feasible solutions of (1) in each step. In this and some other papers information about the last $\overline{T}_{i-1}$ and z^{i-1} is used in choosing γ_i and the initial set $\overline{T}_i$ at ith step. In [28] discretization processes are carried out such that, for a certain class of convex semi-infinite programming problems, the rate of convergence of the basic optimization algorithm used for solving the discretized problems is retained for the whole solution procedure. Hereby only one step of the optimization algorithm is performed in each problem $\mathcal{P}(\overline{T}_i)$ to define $\overline{z}$ (see i3)).

Reduction Methods

These methods use the fact that under appropriate assumptions the original (infinitely many) constraints

$$g(z, t) \leq 0, \quad \forall t \in T,$$

can be replaced by finitely many constraints which locally are sufficient to describe K. In the sequel, we assume that $F \in C^2(\mathbf{R}^n)$, $g \in C^2(\mathbf{R}^n \times T)$. Let $\overline{z} \in \mathbf{R}^n$ be a given point. Let $\overline{t}_1, \ldots, \overline{t}_{q(\overline{z})}$, with $q(\overline{z}) < \infty$, be all the local solutions of

$$\mathcal{Q}(\overline{z}) \quad \max\{g(\overline{z}, t) : \; t \in T\}.$$

Obviously, $\overline{z} \in K$, if and only if

$$g(\overline{z}, \overline{t}_j) \leq 0, \quad j = 1, \ldots, q(\overline{z}).$$

Assume now that there exist a neighborhood $\overline{\mathcal{U}}$ of $\overline{z}$ and twice continuously differentiable functions

$$t_j : \; \overline{\mathcal{U}} \to T, \quad t_j(\overline{z}) = \overline{t}_j, \quad j = 1, \ldots, q(\overline{z})$$

such that for all $z \in \overline{\mathcal{U}}$ the $t_j(z)$ are all local solutions of $\mathcal{Q}(z)$. Then, with

$$G_j(z) := g(z, t_j(z)), \quad j = 1, \ldots, q(\overline{z}),$$

we have $G_j \in C^2(\overline{\mathcal{U}})$ and

$$K \cap \overline{\mathcal{U}} = \{z \in \overline{\mathcal{U}} : \; G_j(z) \leq 0, \; j = 1, \ldots, q(\overline{z})\},$$

i. e. in $\overline{\mathcal{U}}$ we may replace problem $\mathcal{P}(T)$ by the finite problem

$$\mathcal{P}_{\overline{z}}(T) \quad \min\{F(z) : \; G_j(z) \leq 0, \; j = 1, \ldots, q(\overline{z})\}.$$

Sufficient for the existence of $\overline{\mathcal{U}}$ and t_j as above is that all solutions $\overline{t}_j$ of $\mathcal{Q}(\overline{z})$ are nondegenerate (cf. [17,18] and also ► Semi-infinite Programming: Second Order Optimality Conditions). We note that this 'reducibility in $\overline{z}$' is a generic property (see [20,44]), thus it holds 'almost everywhere for almost all problems'.

Now, reduction methods generally can be described as follows:

- Step i: Given z^{i-1} (not necessarily feasible);

  i1) compute all local maxima $t_1^{i-1}, \ldots, t_{q_{i-1}}^{i-1}$ of problem $\mathcal{Q}(z^{i-1})$;

  i2) apply some steps of an algorithm for finite problems to the reduced finite-dimensional problem $\mathcal{P}_{z^{i-1}}(T)$:

  $$\min \left\{F(z):\ G_j(z) \le 0,\ j = 1, \ldots, q_{i-1}\right\},$$

  where $G_j(z) = g(z, t_j(z))$ and the functions $t_j(\cdot)$ are defined in a neighborhood of z^{i-1}. Let z^{i,k_i} be the last iterate of the algorithm mentioned;

  i3) set $z^i = z^{i,k_i}$ and continue with Step$(i+1)$.

In order to apply second order methods usually second order derivatives are required and to guarantee superlinear convergence, additionally a strong second order optimality condition has to be satisfied. Conditions concerning problem (1) which provide locally superlinear convergence of some SQP-methods applied to problem $\mathcal{P}_{z^{i-1}}(T)$ are given in [9] (see also ▶ Semi-infinite Programming: Second Order Optimality Conditions). These conditions ensure, in particular, twice continuous differentiability of the functions G_j. SQP-methods, using augmented Lagrangian functions and quasi-Newton update of their Hessians have been effective in this context (cf. [9,16,25,29,38,43]). Here, according to the results in [9], an inexact evaluation of the functions $t_j(\cdot)$, and hence, of the functions G_j, is admitted.

To give a more explicit version of the reduction approach, we specify the steps i2) and i3) from above, following [9].

Given $z^{i,0} = z^{i-1}$, $B_{i,0} = B_{i-1}$ (B_0 is, in principle, an arbitrary symmetric positive definite $n \times n$-matrix). Then, for $l = 1, \ldots, k_i$, with given $z^{i,l-1}$, $B_{i,l-1}$ do:

- compute a solution s^l and a vector of Lagrange multipliers $\lambda^{i,l}$ of the quadratic programming problem

$$\begin{cases} \min & \left\{\nabla F(z^{i,l-1})^\top s + \frac{1}{2} s^\top B_{i,l-1} s\right\} \\ \text{s.t.} & G_j(z^{i,l-1}) + \nabla G_j(z^{i,l-1})^\top s \le 0, \\ & j = 1, \ldots, q_{i-1}; \end{cases}$$

- compute a steplength α_l by a rule providing a decrease of function (4) below (see, for instance [3]);

- update

$$z^{i,l} = z^{i,l-1} + \alpha_l s^l,$$
$$B_{i,l} = B_{i,l-1} + \frac{y^l (y^l)^\top}{(y^l)^\top s} - \frac{B_{i,l-1} s^l (B_{i,l-1} s^l)^\top}{(s^l)^\top B_{i,l-1} s^l},$$

(BFGS-formula), where

$$y^l = \nabla_z \mathcal{L}^{i-1}(z^{i,l}, \lambda^{i,l}, c) - \nabla_z \mathcal{L}^{i-1}(z^{i,l-1}, \lambda^{i,l}, c),$$
$$\mathcal{L}^{i-1}(z, \lambda, c) = F(z) + \sum_{j=1}^{q_{i-1}} \lambda_j G_j(z) + \frac{c}{2} \sum_{j=1}^{q_{i-1}} (G_j(z))^2$$

is the augmented Lagrangian of problem $\mathcal{P}_{z^{i-1}}(T)$, concerning the choice of $c > 0$ see [18 Assumption 7.8];

- set $B_i = B_{i,k_i}$, $z^i = z^{i,k_i}$, $i = i + 1$.

Conditions ensuring a local q-superlinear convergence of the sequence $\{(z^{i,l}, \lambda^{i,l})\}$ to a Kuhn–Tucker point of problem (1) are given in [9].

In the first instance SQP-methods are only locally convergent. Therefore, hybrid techniques combining robust globally convergent descent algorithms with SQP-methods have been developed (cf. [8,10,11]). Globalization techniques for SQP-methods were introduced in [13] using the exact penalty function

$$F(z) + d \sum_{j=1}^{q_i} \max[0, G_j(z)] \tag{4}$$

(with sufficient large $d > 0$) to control the stepsize. In the papers [3,42] and [37] alternative penalty functions with the same goal have been introduced.

The idea to apply for semi-infinite programming a penalty technique coupled with a discretization seems to be promising. However, numerically this may be rather difficult: The condition number of the Hessians of the objective functions of the auxiliary problems depends essentially not only on the penalty parameter, but also on the common influence of almost active constraints. The number of these constraints usually increases with the number of discretization points. Therefore, the use of exact penalty functions or their smooth approximations has the advantage that it is possible

to work with a fixed penalty parameter. Exact penalty functions of integral type for semi-infinite programming have been proposed in [2]. In [4] and [5] exchange techniques are coupled with penalization of the auxiliary problems, which are finally solved by interior point methods. Interior point methods for solving smooth, convex semi-infinite programming are also suggested in [35,36] and [34]. This approach requires computations of integrals over T to obtain the components of the gradients and Hessians of some aggregated, logarithmic barrier functions. The solution is reached by following the so-called central path of the problem by a predictor-corrector method. In [21,22] proximal penalty approaches have been proposed, with a special deleting rule for the constraints based on the existence of upper bounds for the Lagrange multipliers of the regularized auxiliary problems uniform with respect to a sequence of discretizations.

See also

- ► Semi-infinite Programming: Approximation Methods
- ► Semi-infinite Programming and Control Problems
- ► Semi-infinite Programming: Discretization Methods
- ► Semi-infinite Programming: Methods for Linear Problems
- ► Semi-infinite Programming: Second Order Optimality Conditions
- ► Semi-infinite Programming, Semidefinite Programming and Perfect Duality

References

1. Blum E, Oettli W (1975) Mathematische Optimierung. Grundlagen und Verfahren. Springer, Berlin
2. Conn AR, Gould NIM (1987) An exact penalty function for semi-infinite programming. Math Program 37:19–40
3. Coope ID, Watson GA (1985) A projected Lagrangian algorithm for semi-infinite programming. Math Program 32:337–356
4. Fang S-Z, Lin C-J, Wu S-Y (1994) On solving convex quadratic semi-infinite programming problems. Optim 31:107–125
5. Fang S-Z, Tsao H-SJ (1996) An efficient computational procedure for solving entropy optimization problems with infinitely many linear constraints. J Comput Appl Math 72:127–139
6. Fiacco AV, Kortanek K (eds), (1983) Semi-infinite programming and applications. Springer, Berlin
7. Gonzaga C, Polak E, Trahan R (1980) An improved algorithm for optimization problems with functional inequality constraints. IEEE Trans Autom Control 25:49–54
8. Görner St (1997) Ein Hybridverfahren zur Lösung nichtlinearer semi-infiniter Optimierungsaufgaben. PhD Thesis, Techn Univ Berlin, Berlin
9. Gramlich G, Hettich R, Sachs E (1995) SQP-methods in semi-infinite programming. SIAM J Optim 5:641–658
10. Gustafson SA (1983) A three-phase algorithm for semi-infinite programming problems. In: Fiacco AV, Kortanek KO (eds) Semi-infinite Programming and Applications. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 138–157
11. Gustafson SA, Kortanek KO (1973) Numerical solution of a class of semi-infinite programming problems. Naval Res Logist Quart 20:477–504
12. Gustafson SA, Kortanek KO (1983) Semi-infinite programming and applications. In: Bachem A, Grötschel M, Korte B (eds) Mathematical Programming: The State of the Art. Springer, Berlin, pp 132–157
13. Han SP (1977) A globally convergent method for nonlinear programming. J Optim Th Appl 22:297–309
14. Hettich R (1983) A review of numerical methods for semi-infinite optimization. In: Fiacco AV, Kortanek KO (eds) Semi-infinite Programming and Applications. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 158–178
15. Hettich R, Gramlich G (1990) A note on an implementation of a method for quadratic semi-infinite programs. Math Program 46:249–254
16. Hettich R, Honstede W van (1979) On quadratically convergent methods for semi-infinite programming. In: Hettich R (ed) Semi-infinite Programming. Lecture Notes Control Inform Sci. Springer, Berlin, pp 97–111
17. Hettich R, Jongen HTh (1978) Semi-infinite programming: Conditions of optimality and applications. In: Lecture Notes Control Inform Sci, vol 7, pp 1–11
18. Hettich R, Kortanek KO (1993) Semi-infinite programming: Theory, methods and applications. SIAM Rev 35:380–429
19. Hettich R, Zencke P (1982) Numerische Methoden der Approximation und semi-infiniten Optimierung. Studienbücher Math. Teubner, Leipzig
20. Jongen HTh, Jonker P, Twilt F (1986) Critical sets in parametric optimization. Math Program 34:333–353
21. Kaplan A, Tichatschke R (1992) A regularized penalty method for solving convex semi-infinite programs. Optim 26:215–228
22. Kaplan A, Tichatschke R (1996) Path-following proximal approach for solving ill-posed convex semi-infinite programming problems. J Optim Th Appl 90:113–137
23. Kortanek KO, No H (1993) A central cutting plane algorithm for convex semi-infinite programs. SIAM J Optim 3:901–918
24. Krabs W (1975) Optimierung und Approximation. Studienbücher Math. Teubner, Leipzig

25. Lawrence CT, Tits AL (1997) Feasible sequential quadratic programming for finely discretized problems from SIP. In: Rückmann J, Reemtsen R (eds) Semi-Infinite Programming. Kluwer, Dordrecht, pp 159–193
26. Painer ER, Tits AL (1989) A globally convergent algorithm with adaptively refined discretization for semi-infinite optimization problems arising in engineering design. IEEE Trans Autom Control 39:903–908
27. Polak E (1987) On the mathematical fundations of nondifferentiable optimization in engineering design. SIAM Rev 29:21–89
28. Polak E, He L (1992) Rate preserving discretization strategies for semi-infinite programming and optimal control. SIAM J Control Optim 30:548–572
29. Polak E, Tits AL (1982) A recursive quadratic programming algorithm for semi-infinite programs. J Appl Math Optim 8:325–349
30. Potchinkov A, Reemtsen R (1995) The design of FIR-filters in the complex plane by convex optimization. Signal Processing 46:127–146
31. Reemtsen R (1992) A cutting plane method for solving minimax problems. Numer Algorithms 2:409–436
32. Reemtsen R (1994) Some outer approximation methods for semi-infinite optimization problems. J Comput Appl Math 53:87–108
33. Reemtsen R, Rückmann J-J (eds) (1998) Semi-infinite programming. Kluwer, Dordrecht
34. Schättler U (1996) An interior-point method for solving semi-infinite programming problems. Ann Oper Res 62:277–301
35. Sonnevend G (1990) Applications of analytic centers for the numerical solution of semi-infinite, convex programs arising in control theory. In: Lecture Notes Control Inform Sci, vol 143, pp 413–422
36. Sonnevend G (1994) A new class of a high order interior point methods for the solution of convex semi-infinite optimization problems. In: Bulirsch R, Kraft D (eds) Computational and optimal control. Birkhäuser, Basel, pp 193–211
37. Tanaka Y, Fukushima M, Hasegawa T (1987) Implementable L-infinity penalty function-method for semi-infinite optimization. Internat J Syst Sci 18:1563–1568
38. Tanaka Y, Fukushima M, Ibaraki T (1988) A globally convergent SQP-method for semi-infinite optimization. J Comput Appl Math 23:141–153
39. Tichatschke R (1985) Semi-infinite programming problems. Math Control Th, Banach Center Publ 14:543–554
40. Tichatschke R, Lohse Th (1982) Eine verallgemeinerte Schnittmethode für konvexe semi-infinite Optimierungsaufgaben. Wiss Z Techn Hochsch Karl-Marx-Stadt 24:332–338
41. Tichatschke R, Nebeling V (1988) A cutting- plane algorithm for quadratic semi-infinite programming problems. Optim 19:803–817
42. Watson GA (1981) Globally convergent methods for semi-infinite programming problems. BIT 21:362–373
43. Zhou JL, Tits AL (1996) An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. SIAM J Optim 6:461–487
44. Zwier G (1987) Structural analysis in semi-infinite programming. PhD Thesis, Twente Univ, Enschede

Semi-infinite Programming: Second Order Optimality Conditions

SIP

Rainer Hettich[1], Georg Still[2]

[1] University Trier, Trier, Germany

[2] University Twente, Enschede, The Netherlands

MSC2000: 90C31, 90C34

Article Outline

Keywords
The Parametric Problem $\Theta(\bar{z})$
The Reduced Problem
Optimality Conditions for SIP
Optimality Conditions for GSIP
See also
References

Keywords

Semi-infinite optimization; Second order necessary and sufficient optimality conditions; Reduced problem in semi-infinite programming

We are concerned with *semi-infinite optimization problems*

$$\text{SIP} \quad \begin{cases} \max & F(z) \\ \text{s.t.} & z \in Z \end{cases}$$

with feasible set

$$Z = \{z \in \mathbb{R}^n : \ g(z,t) \leq 0 \text{ for all } t \in B\},$$

and

$$B = \left\{t \in \mathbb{R}^m : \ h^j(t) \leq 0, j \in J\right\},$$

J a finite index set, $F \in C^2(\mathbf{R}^n, \mathbf{R})$, $g \in C^2(\mathbf{R}^n \times \mathbf{R}^m, \mathbf{R})$, $h^j \in C^2(\mathbf{R}^n, \mathbf{R})$, $j \in J$. We assume that B is compact.

Later on we consider *generalized semi-infinite problems* GSIP, where the index set $B = B(z)$ depends on z, i. e. $B = B(z) = \{t \in \mathbf{R}^m : h^j(z, t) \leq 0, j \in J\}$ with given $h^j \in C^2(\mathbf{R}^n \times \mathbf{R}^m, \mathbf{R})$, $j \in J$. In this case we assume that the range of the set-valued mapping B remains in a compact set $B_0 \subset \mathbf{R}^m$, i. e. $B(z) \subset B_0$ for all $z \in \mathbf{R}^n$.

In the same way as in finite optimization, second order optimality conditions for SIP are an important tool for answering a number of theoretical and practical questions. First order optimality conditions may again be given by a system of nonlinear equations, the so-called Karush–Kuhn–Tucker equations. Then, strong sufficient second order conditions imply this system to be regular and this regularity allows to answer questions of stability of solutions, dependence of solutions on problem parameters but also superlinear convergence of some classes of numerical methods (cf. [3] and the references therein). We give the conditions in a form which is suitable for applications like these. We follow the 'reduction approach' as in [3] and [4].

In the sequel, the *parametric optimization problem*

$$\Theta(z) \quad \begin{cases} \max\limits_{t} & g(z, t) \\ \text{s.t.} & t \in B \end{cases}$$

will play an important role. For $\overline{z} \in Z$ we define the *active index set*

$$E(\overline{z}) = \left\{\overline{t} \in B : \; g(\overline{z}, \overline{t}) = 0\right\}.$$

Obviously, any point $\overline{t}$ in $E(\overline{z})$ is a (global) maximum of $\Theta(\overline{z})$. Throughout we assume that $E(\overline{z})$ is nonempty. (If $E(\overline{z}) = \emptyset$, then by the assumptions above, the point $\overline{z}$ is an interior point of Z and the optimality conditions coincide with the optimality conditions in unconstrained optimization.)

The row vectors of partial derivatives of g with respect to z, t will be denoted by g_z, g_t. For a function v: $\mathbf{R}^n \to \mathbf{R}$ we denote the directional derivative of v in $\overline{z}$ in the direction $\xi \in \mathbf{R}^n$ by $Dv(\overline{z}; \xi)$,

$$Dv(\overline{z}; \xi) = \lim_{\tau \to 0+} \frac{v(\overline{z} + \tau\xi) - v(\overline{z})}{\tau}.$$

If the derivative v_z is itself directionally differentiable in the direction ξ, we put $D^2 v(\overline{z}; \xi) := Dv_z(\overline{z}; \xi)\xi$.

For brevity, we do not consider vector-valued functions g or equality constraints in the definition of Z and B.

The Parametric Problem $\Theta(\overline{z})$

In this section we prepare the 'reduction approach' for obtaining optimality conditions for SIP. We briefly outline a stability result (cf. [5]) for the parametric problem (applying to SIP and GSIP as well)

$$\Theta(z) \quad \begin{cases} \max\limits_{t} & g(z, t) \\ \text{s.t.} & t \in B(z), \end{cases}$$

$B(z) := \{t : h^j(z, t) \leq 0, j \in J\}$. Let $\overline{z}$ and a point $\overline{t}$ which is feasible for $\Theta(\overline{z})$ be given. We define the active index set of $\Theta(\overline{z})$ as

$$J(\overline{z}, \overline{t}) = \left\{j \in J : \; h^j(\overline{z}, \overline{t}) = 0\right\}.$$

We say that the *linear independency constraint qualification* (LICQ) holds if the vectors

$$h_t^j(\overline{z}, \overline{t}), \quad j \in J(\overline{z}, \overline{t}),$$

are linearly independent. The (weaker) *Mangasarian–Fromovitz constraint qualification* (MFCQ) is said to hold in $\overline{t}$ for $\Theta(\overline{z})$, if there exists a vector $\overline{\eta} \in \mathbb{R}^m$ such that

$$h_t^j(\overline{z}, \overline{t})\overline{\eta} < 0, \quad j \in J(\overline{z}, \overline{t}).$$

Let a solution $\overline{t}$ of $\Theta(\overline{z})$ be given. If at $\overline{t}$ the MFCQ is satisfied, then necessarily the Kuhn–Tucker condition is fulfilled, i. e. there exists a multiplier vector $\overline{\gamma} \in \mathbb{R}^{|J(\overline{z}, \overline{t})|}$ such that

$$L_t^{\overline{t}}(\overline{z}, \overline{t}, \overline{\gamma}) = 0, \quad \overline{\gamma} \geq 0, \tag{1}$$

where $L^{\overline{t}}$ denotes the *Lagrange function* of $\Theta(z)$ at $(\overline{z}, \overline{t})$,

$$L^{\overline{t}}(z, t, \gamma) = g(z, t) - \sum_{j \in J(\overline{z}, \overline{t})} \gamma_j h^j(z, t).$$

We say that *strict complementary slackness* (SCS) holds if the multiplier $\overline{\gamma}$ in (1) satisfies

$$\overline{\gamma}_j > 0, \quad j \in J(\overline{z}, \overline{t}).$$

Let us introduce the following second order sufficient optimality condition for $\Theta(\overline{z})$:

$A_{\mathbf{red}}$) Let a solution $\overline{t}$ of $\Theta(\overline{z})$ be given. Assume that LICQ is satisfied. (Under LICQ there is a unique multiplier $\overline{\gamma}$ such that (1) is fulfilled.) We assume

that the strong second order condition (SSOC) holds,

$$\eta^\top L^{\bar t}_{tt}(\bar z, \bar t, \bar\gamma)\eta < 0 \quad \text{for all } \eta \in T(\bar z, \bar t) \setminus \{0\},$$

where $T(\bar z, \bar t)$ is the subspace $T(\bar z, \bar t) = \{\eta \in \mathbb{R}^m : \bar\gamma_j h^j_t(\bar z, \bar t)\eta = 0,\ j \in J(\bar z, \bar t)\}$.

Theorem 1 *Let $\bar t$ be a solution of $\Theta(\bar z)$ such that A_{red}) is satisfied. Then there exist neighborhoods $U(\bar z)$ of $\bar z$ and $V(\bar t)$ of $\bar t$ and mappings $t : U(\bar z) \to V(\bar t)$, $\gamma : U(\bar z) \to \mathbb{R}^{|J(\bar z, \bar t)|}$ such that:*

- $t(\bar z) = \bar t$,
- $\gamma(\bar z) = \bar\gamma$,
- *and for all $z \in U(\bar z)$ the value $t(z)$ is the unique local solution of $\Theta(z)$ in $V(\bar t) \cap B(z)$ with corresponding multiplier $\gamma(z)$ such that LICQ and SSOC are satisfied.*

The functions t, γ are Lipschitz continuous and directionally differentiable. The value function

$$G(z) := g(z, t(z)),$$

is continuously differentiable in $U(\bar z)$ and the derivative G_z is Lipschitz continuous and directionally differentiable with:

$$\begin{aligned} G_z(z) &= g_z(z, t(z)) \\ &\quad - \sum_{j \in J(\bar z, \bar t)} \gamma_j(z) h^j_z(z, t(z)) \\ &= L^{\bar t}_z(z, t(z), \gamma(z)), \end{aligned} \tag{2}$$

$$\begin{aligned} D^2G(z;\xi) &= \xi^\top L^{\bar t}_{zz}(z, t(z), \gamma(z))\xi \\ &\quad - D^\top t(z;\xi) L^{\bar t}_{tt}(z, t(z), \gamma(z)) Dt(z;\xi) \\ &\quad - 2 \sum_{j \in J(\bar z, \bar t)} D\gamma_j(z;\xi) h^j_z(z, t(z))\xi. \end{aligned} \tag{3}$$

The directional derivatives $Dt(z;\xi)$ are given as the unique solutions of the quadratic program

$$\mathrm{P}(\xi) \quad \begin{cases} \max\limits_{\eta} & F(\xi, \eta) \\ s.t. & h^j_z(z, t(z))\xi + h^j_t(z, t(z))\eta \\ & \begin{cases} = 0 & \text{if } \gamma_j(z) > 0 \\ \le 0 & \text{if } \gamma_j(z) = 0 \end{cases} \\ & \text{for } j \in J(\bar z, \bar t), \end{cases}$$

where

$$\begin{aligned} F(\xi, \eta) := \frac{1}{2}\eta^\top L^{\bar t}_{tt}(z, t(z), \gamma(z))\eta \\ + \eta^\top L^{\bar t}_{zt}(z, t(z), \gamma(z))\xi \end{aligned}$$

The directional derivatives $D\gamma(z;\xi)$ are given by the unique multipliers $\beta(\xi)$ of $P(\xi)$, $D\gamma(z;\xi) = \beta(\xi)$.

Remark 2

a) For SIP, i. e. if the functions h^j do not depend on z, the derivatives of G and problem P(ξ) take the simpler form:

$$\begin{aligned} &G_z(z) = g_z(z, t(z)), \\ &D^2G(z;\xi) = \xi^\top g_{zz}(z, t(z))\xi \\ &\quad - D^\top t(z;\xi) L^{\bar t}_{tt}(z, t(z), \gamma(z)) Dt(z;\xi) \end{aligned}$$

and

$$\mathrm{P}(\xi) \quad \begin{cases} \max\limits_{\eta} & \frac{1}{2}\eta^\top L^{\bar t}_{tt}(z, t(z), \gamma(z))\eta \\ & \qquad + \eta^\top g_{zt}(z, t(z))\xi \\ \text{s.t.} & h^j_t(z, t(z))\eta \\ & \qquad \begin{cases} = 0 & \text{if } \gamma_j(z) > 0 \\ \le 0 & \text{if } \gamma_j(z) = 0 \end{cases} \\ & \qquad j \in J(\bar z, \bar t). \end{cases}$$

b) If in A_{red}), in addition to LICQ and SSOC, we assume SCS, then the functions t, γ are continuously differentiable and G is twice continuously differentiable. Consequently, in (3) we can write $Dt(z;\xi) = t_z(z)\xi$, $D\gamma(z;\xi) = \gamma_z(z)\xi$.

The Reduced Problem

Let $\bar z$ be feasible for SIP. By assuming that A_{red}) is satisfied for all solutions $\bar t$ of $\Theta(z)$ (i. e. all $\bar t \in E(\bar z)$), by applying Theorem 1, the problem SIP can locally be reduced to an equivalent finite optimization problem.

Theorem 3 *Let $\bar z \in Z$ be given such that A_{red}) is satisfied for all $\bar t \in E(\bar z)$. Then the set $E(\bar z)$ is finite, $E(\bar z) = \{\bar t^1, \dots, \bar t^r\}$, and there is a neighborhood $U(\bar z)$ of $\bar z$ such that with the functions*

$$G^\ell(z) := g(z, t^\ell(z)), \quad \ell = 1, \dots, r,$$

in Theorem 1 (corresponding to the solutions $\bar t^\ell$ of $\Theta(\bar z)$) the following holds: $z \in U(\bar z)$ is a local maximizer of SIP if and only if z is a local maximizer of the so-called reduced problem

$$\mathrm{SIP}_{red}(\bar z) \quad \begin{cases} \max & F(z) \\ s.t. & z \in U(\bar z) \\ & G^\ell(z) \le 0, \\ & \quad \ell = 1, \dots, r. \end{cases}$$

By Theorem 1, the functions G^ℓ are continuously differentiable in $U(\overline{z})$ and G^ℓ_z is Lipschitz continuous and directionally differentiable.

Optimality Conditions for SIP

We consider SIP and assume, that for $\overline{z} \in Z$ the regularity conditions of Theorem 3 hold. Then, in a neighborhood $U(\overline{z})$ of $\overline{z}$, the problem SIP is equivalent with the finite problem $\mathrm{SIP}_{\mathrm{red}}(\overline{z})$. Let $\mathbf{L}$ denote the (F. John type) *Lagrangian* of $\mathrm{SIP}_{\mathrm{red}}(\overline{z})$,

$$\mathbf{L}(z,\lambda) = \lambda_0 F(z) - \sum_{\ell=1}^{r} \lambda_\ell G^\ell(z),$$

with $G^\ell(z) = g(z, t^\ell(z))$, and K the *cone of critical directions* for $\mathrm{SIP}_{\mathrm{red}}(\overline{z})$ in $\overline{z}$,

$$K = \left\{ \xi \in \mathbb{R}^n : \begin{array}{l} F_z(\overline{z})\xi \geq 0, \\ G^\ell_z(\overline{z})\xi \leq 0, \end{array} \ \ell = 1, \ldots, r \right\}.$$

The following dual optimality conditions hold.

Theorem 4 *Let $\overline{z}$ be feasible for SIP such that the assumptions of Theorem 3 are satisfied. Then we have:*

a) (Necessary conditions) Suppose, $\overline{z}$ is a local maximizer of SIP. Then, for any $\xi \in K$ there exist multipliers $\overline{\lambda}_0(\xi), \ldots, \overline{\lambda}_r(\xi) \geq 0$, not all zero, such that (with the expressions for $S_{SIP}(\xi)$, $Q_{SIP}(\xi)$ below)

$$S_{SIP}(\xi) = 0 \quad \textit{and} \quad Q_{SIP}(\xi) \leq 0.$$

The multipliers $\overline{\lambda}_\ell(\xi)$ can be chosen 0 if $g_z(\overline{z}, \overline{t}^\ell)\xi < 0$, and $\overline{\lambda}_0(\xi)$ can be chosen 0 if $F_z(\overline{z})\xi > 0$.

b) (Sufficient conditions) Suppose that for any $\xi \in K \setminus \{0\}$ there exist multipliers $\overline{\lambda}_0(\xi), \ldots, \overline{\lambda}_r(\xi) \geq 0$ such that

$$S_{SIP}(\xi) = 0 \quad \textit{and} \quad Q_{SIP}(\xi) < 0.$$

Then $\overline{z}$ is a strict local maximizer of SIP.

The expressions for $S_{SIP}(\xi)$, $Q_{SIP}(\xi)$ are:

$$\begin{aligned} S_{SIP}(\xi) &= \mathbf{L}_z(\overline{z}, \overline{\lambda}(\xi)) \\ &= \overline{\lambda}_0(\xi) F_z(\overline{z}) - \sum_{\ell=1}^{r} \overline{\lambda}_\ell(\xi) g_z(\overline{z}, \overline{t}^\ell) \end{aligned} \tag{4}$$

$$\begin{aligned} Q_{SIP}(\xi) &= \xi^\top \left(\overline{\lambda}_0(\xi) F_{zz}(\overline{z}) - \sum_{\ell=1}^{r} \overline{\lambda}_\ell(\xi) g_{zz}(\overline{z}, \overline{t}^\ell) \right) \xi \\ &\quad + \sum_{\ell=1}^{r} \overline{\lambda}_\ell(\xi) D^\top t^\ell(\overline{z};\xi) L^{\overline{t}^\ell}_{tt}(\overline{z}, \overline{t}^\ell, \overline{\gamma}^\ell) D t^\ell(\overline{z};\xi). \end{aligned} \tag{5}$$

The proof of Theorem 4 follows by applying F. John type optimality conditions of finite optimization to $\mathrm{SIP}_{\mathrm{red}}(\overline{z})$ (conditions, modified to deal with functions whose derivatives are Lipschitz continuous and directionally differentiable cf. [4]). This leads to the following primal necessary conditions (under the assumptions of Theorem 4): Let $\overline{z}$ be a local solution of SIP. Then, for any $\xi \in K$ the following system has no solution $d \in \mathbf{R}^n$:

$$\begin{cases} g_z(\overline{z}, \overline{t}^\ell) d + \xi^\top g_{zz}(\overline{z}, \overline{t}^\ell)\xi \\ \quad -D^\top t(\overline{z};\xi) L^{\overline{t}^\ell}_{tt}(\overline{z}, \overline{t}^\ell, \overline{\gamma}^\ell) D t(\overline{z};\xi) < 0, \\ \quad \overline{t}^\ell \in E(\overline{z};\xi), \\ F_z(\overline{z}) d + \xi^\top F_{zz}(\overline{z})\xi > 0, \\ \quad \text{if } F_z(\overline{z})\xi = 0, \end{cases} \tag{6}$$

where $E(\overline{z},\xi) = \{\overline{t}^\ell \in E(\overline{z}) : \ g_z(\overline{z}, \overline{t}^\ell)\xi = 0\}$. An appropriate theorem of the alternative implies Theorem 4a. Correspondingly, the sufficient condition in Theorem 4b is equivalent to the nonsolvability of the system (6) with strict signs replaced by the weak signs, $E(\overline{z};\xi)$ replaced by $E(\overline{z})$ and 'if $F_z(\overline{z})\xi = 0$' omitted.

In the following remark we discuss optimality conditions under stronger as well as conditions under weaker regularity assumptions.

Remark 5 Let $\overline{z}$ be feasible for SIP.

a) If in $\mathrm{A}_{\mathrm{red}}$), in addition to LICQ and SSOC we assume SCS, then (cf. Remark 2b) the functions G^ℓ in the reduced problem $\mathrm{SIP}_{\mathrm{red}}(\overline{z})$ are twice continuously differentiable and in Theorem 4 the term $Q_{\mathrm{SIP}}(\xi)$ can be replaced by

$$\begin{aligned} \xi^\top \Bigg(\overline{\lambda}_0(\xi) F_{zz}(\overline{z}) - \sum_{\ell=1}^{r} \overline{\lambda}_\ell(\xi) g_{zz}(\overline{z}, \overline{t}^\ell) \\ + \sum_{\ell=1}^{r} \overline{\lambda}_\ell(\xi) t^\ell_z(\overline{z})^\top L^{\overline{t}^\ell}_{tt}(\overline{z}, \overline{t}^\ell, \overline{\gamma}^\ell) t^\ell_z(\overline{z}) \Bigg) \xi. \end{aligned} \tag{7}$$

Such optimality conditions have been derived in [2]. The idea of the 'reduction approach' goes back to [12] where in addition, it is assumed that LICQ holds: $g_z(\overline{z}, \overline{t}^\ell)(= G^\ell_z(\overline{z}))$, $\ell = 1, \ldots, r$, are linearly independent. Then in the terms S and Q we have $\overline{\lambda}_0 = 1$ (Kuhn–Tucker condition) and the multipliers $\overline{\lambda}_1, \ldots, \overline{\lambda}_r$ are unique (independently of ξ), i. e. the matrix between parentheses in (7) is negative (semi)definite.

b) In [3, Thm. 5.4] a sufficient optimality condition similar to Theorem 4b is proven which does not make use of a reduction assumption. It is based on the idea to replace in (5) the directional derivative $Dt^\ell(\overline{z};\xi)$ by a solution of the quadratic problem $P^\ell(\xi)$ and assumes that condition LICQ holds at all points $\overline{t} \in E(\overline{z})$.

c) In [7,8] H. Kawasaki has given necessary and sufficient optimality conditions under weaker regularity assumptions and without assuming differentiability of g with respect to t. The expressions

$$D^\top t^\ell(\overline{z};\xi) L_{tt}^{\overline{t}^\ell}(\overline{z},\overline{t}^\ell,\overline{\gamma}^\ell) Dt^\ell(\overline{z};\xi) \tag{8}$$

are replaced by terms $2\varepsilon(\overline{t}^\ell,\xi)$ which describe a second order approximation of the feasible set Z at $\overline{z}$ in the direction ξ. The necessary dual conditions (cf. [8, Thm. 3.1]) do not make regularity assumptions on $\Theta(\overline{z})$. A similar sufficient condition without a gap is obtained (apart from the strong sign in the second order condition) (cf. [8, Thm. 4.1]) under the assumption that B is convex and conditions on the behavior of certain functions such as $g(\overline{z},t)$ which, in particular, imply that the set $E(\overline{z};\xi)$ is finite. In [8] it is shown that for the special case $B = [0,1]$, under the regularity conditions of Theorem 4, the term $2\varepsilon(\overline{t}^\ell,\xi)$ coincides with (8). These terms (often called 'shift terms') reflect the 'semi-infinite structure' inasmuch as they describe the dependence of $E(z)$ on z.

d) In [1] the 'shift terms' are given in the form

$$\sigma(\lambda(\xi),\tau(\xi)) \tag{9}$$

where λ is a multiplier function (in the space of measures), $\tau(\xi)$ describes a second order approximation of the feasible set and σ denotes the support function. There is a gap between the necessary and the sufficient conditions (cf. [1, Thms. 3.1; 3.2]) inasmuch as (apart from the strong sign) in the sufficient condition the term (9) is given with a set $\tau(\xi)$ which can be strictly larger than the corresponding set in the necessary condition. However, under the following assumptions there is no gap: $g(\cdot,t)$ is twice differentiable, g_{zz} is continuous on $\mathbf{R}^n \times B$, $g(\overline{z},\cdot)$ is twice continuously differentiable, $g_{z_i}(\overline{z},\cdot)$ are continuously differentiable; the constraint qualification (CQ) is fulfilled: there exists $\overline{\xi}$ such that $g_z(\overline{z},\overline{t})\overline{\xi} < 0$, $\overline{t} \in E(\overline{z})$; $g(\overline{z},t)$ satisfies a certain second order growth condition; B and $E(\overline{z})$ are smooth (compact) manifolds.

Optimality Conditions for GSIP

We briefly outline optimality conditions for GSIP. The statements of Theorems 3 and 4 remain true for the generalized problem with the only modification that due to the dependence of B (i. e. h^j) on z the expressions for $S(\xi)$ and $Q(\xi)$ (cf. Theorem 4) contain extra terms (all terms in (2), (3)). Consequently, Theorem 4 holds for GSIP if the expressions for $S_{\mathrm{SIP}}(\xi)$ and $Q_{\mathrm{SIP}}(\xi)$ in (4) and (5) are replaced by:

$$S_{\mathrm{GSIP}}(\xi) = S_{\mathrm{SIP}}(\xi) + \sum_{\ell=1}^{r} \overline{\lambda}_\ell(\xi) \sum_{j\in J(\overline{z},\overline{t}^\ell)} \overline{\gamma}_j^\ell h_z^j(\overline{z},\overline{t}^\ell),$$

$$Q_{\mathrm{GSIP}}(\xi) = Q_{\mathrm{SIP}}(\xi) + \sum_{\ell=1}^{r} \overline{\lambda}_\ell(\xi) \times \sum_{j\in J(\overline{z},\overline{t}^\ell)} \left(\overline{\gamma}_j^\ell \xi^\top h_{zz}^j(\overline{z},\overline{t}^\ell)\xi + 2D\gamma_j^\ell(\overline{z};\xi) h_z^j(\overline{z},\overline{t}^\ell)\xi\right).$$

We give some information on other optimality conditions for GSIP.

Remark 6 Under the additional assumption SCS A_{red}), as in the SIP case in Remark 5a, the formula for $Q_{\mathrm{GSIP}}(\xi)$ simplify. Corresponding optimality conditions are implicitly contained in [10] (see also [11]).

A second order sufficient condition not based on a reduction (as in Remark 5b) is to be found in [4, Thm. 5.1]. For first order necessary conditions under weak regularity assumptions, see [6] and [9]. In the first paper no regularity assumption on $\Theta(\overline{z})$ is made, whereas the latter assumes that MFCQ is fulfilled for any $\overline{t} \in E(\overline{z})$.

See also

- ▶ Semi-infinite Programming: Approximation Methods
- ▶ Semi-infinite Programming and Control Problems
- ▶ Semi-infinite Programming: Discretization Methods
- ▶ Semi-infinite Programming: Methods for Linear Problems
- ▶ Semi-infinite Programming: Numerical Methods
- ▶ Semi-infinite Programming, Semidefinite Programming and Perfect Duality

References

1. Bonnansand F, Cominetti R, Shapiro A. Second order necessary and sufficient optimality conditions under abstract constraint, to appear
2. Hettich R, Jongen HTh (1978) Semi-infinite programming: Conditions of optimality and applications. In: Stoer J (ed) Optimization Techniques, Part 2. Lecture Notes Control Inform Sci. Springer, Berlin, pp 1–11
3. Hettich R, Kortanek K (1993) Semi-infinite programming: Theory, methods and applications. SIAM Rev 35(3):380–429
4. Hettich R, Still G (1995) Second order optimality conditions for generalized semi-infinite programming problems. Optim 34:195–211
5. Jittorntrum K (1984) Solution point differentiability without strict complementarity in nonlinear programming. In: Fiacco AV (ed) Sensitivity: Stability and Parametric Analysis. Math Program Stud, pp 127–138
6. Jongen HTh, Rückmann J-J, Stein O (1998) Generalized semi-infinite optimization: A first order optimality condition and examples. Math Program 83:145–158
7. Kawasaki H (1988) An envelope-like effect of infinitely many inequality constraints on second order necessary conditions for minimization problems. Math Program 41:73–96
8. Kawasaki H (1992) Second-Order necessary and sufficient optimality conditions for minimizing a sup-type function. Appl Math Optim 26:195–220
9. Rückmann J-J, Shapiro A (1999) On first order optimality conditions in generalized semi-infinite programming. J Optim Th Appl 101(1):677–691
10. Shapiro A (1985) Second order derivatives of extremal value functions and optimality conditions for semi-infinite programs. Math Oper Res 10:207–219
11. Still G (1999) Generalized semi-infinite programming: Theory and methods. Europ J Oper Res 119:301–313
12. Wetterling W (1970) Definitheitsbedingungen für relative Extrema bei Optimierungs- und Approximationsaufgaben. Numer Math 15:122–136

Semi-infinite Programming, Semidefinite Programming and Perfect Duality

K. O. Kortanek, Qinghong Zhang
University Iowa, Iowa City, USA

MSC2000: 90C05, 90C25, 90C30, 90C34

Article Outline

Keywords
Duality of the Linear SIP Problem
Dual Semidefinite Programs
Perfect Duality from the View
of Linear Semi-Infinite Programming
The SDP as a Conic Convex Program
See also
References

Keywords

Semi-infinite programming; Conic convex programs; Perfect duality; Descent vector; Perturbations

Duality of the Linear SIP Problem

Linear semi-infinite programming is a next level of extension of elementary finite linear programming where now finitely many variables occur in infinitely many linear inequalities. Some of the earliest papers were by A. Charnes, W.W. Cooper, and K.O. Kortanek, [3,4,5,6]. More recent surveys by S.-Å. Gustafson and R. Hettich were co-authored with Kortanek, see [8,11], and several symposium volumes on semi-infinite programming have appeared since the late 1970s, [7,10].

In this paper we begin with the primal program denoted by **Program** I. The analogy to a primal finite linear program is thus made apparent.

> **Program** I. Let T be a nonempty set and $u = (u_1, \ldots, u_{n+1})$ be real-valued functions on T and $b \in \mathbf{R}^n$. Find $v_I = \sup y^\mathsf{T} b$ from among $y \in \mathbf{R}^n$ such that $y^\mathsf{T} u(t) \leq u_{n+1}(t)$, $\forall t \in T$.

Clearly, if we choose a finite subset of $\{t_1, \ldots, t_k\}$ of T, then we obtain a finite LP approximation to Program I, namely,

> **Program** I_k. Find $v_{I_k} = \max y^\mathsf{T} b$ subject to $y \in \mathbf{R}^n$ and $y^\mathsf{T} u(t_i) \leq u_{n+1}(t_i)$, $i = \overline{1, k}$.

Remark 1 Without sufficient regularity conditions, Program I_k may be a very bad approximation to Program I, and possibly useless. Here, we also use the typical Russian notation $i = \overline{1, m}$ for $i = 1, \ldots, m$.

Constructing the dual program to a finite LP is a pleasant task, and in this case yields the following,

> **Program** II_k. Find $v_{I_k} = \min \sum_{i=1}^k u_{n+1}(t_i)\ \lambda_i$ with $\lambda_i \geq 0$, $i = \overline{1, k}$ such that $\sum_{i=1}^k u(t_i)\ \lambda_i = b$.

Now any finite LP approximation depends on a selection of a finite subset of the given *SIP index set* T, and of course there are infinitely many such choices. To attempt to recover the given infinite problem by this procedure requires that we at least allow the finite subsets of T to be freely chosen. This is formulated mathematically with *generalized finite sequences*, a particular *function space*, in the dual program. There are close connections between this construction and G.B. Dantzig 's concept of *generalized linear programming with variable coefficients*, and some of these were explored in a paper by Kortanek in the 1970s, [12].

Program *II*. Find $v_{II} = \inf_{t \in \operatorname{supp} \lambda} u_{n+1}(t)\lambda(t)$ from among $\lambda \in \mathbf{R}^{(S)}$, $\lambda(\cdot) \geq 0$, such that $\sum_{t \in \operatorname{supp} \lambda} u(t)\, \lambda\, (t) = b$, where $\lambda \in \mathbf{R}^{(S)}$ are generalized finite sequences, namely $\operatorname{supp} \lambda = \{t\colon \lambda(t) \neq 0\}$ is a finite set.

From a probabilist 's point of view Program *II* is equivalent to the following:

Program *IIB*. Find $v_{IIB} = \inf \int_T u_{n+1}\, d\alpha$ where α are nonnegative Borel measures on a Borel set $T \subset \mathbf{R}^n$ subject to $\int_T u_i\, d\alpha = b_i$, $i = \overline{1,k}$, and where u_i are Borel integrable functions, see [9].

Using analogous algebraic manipulations as in finite linear programming, one can verify that if y and λ are feasible respectively for Programs *I* and *II*, then

$$y^\top b \leq \sum_{s \in \operatorname{supp} \lambda} u_{n+1}(s)\lambda(s).$$

When both programs are feasible, it follows that

$$v_I \leq v_{II}. \tag{1}$$

Termed the *duality inequality*, (1) becomes an equality when certain regularity conditions hold. The most widespread sufficient condition is that the functions $u(\cdot)$ are continuous on a compact set S, and the *Slater condition* holds, namely there exists $\widehat{y}$ such that $\widehat{y}^\top u(t) < u_{n+1}(t), \forall t \in S$. This condition is also termed *superconsistency*, and it is defined for the dual Program *II* in a different way. As reviewed in [11, Thm. 6.11], a program is superconsistent if and only if its dual program has *bounded level sets*. Bounded level sets are necessary and sufficient to overcome the 'danger' noted below Program I_k about using any one finite linear program as an approximation to a linear semi-infinite program.

In general, without a constraint qualification a *duality gap*, $v_I < v_{II}$ can occur.

Example 2

$$v_{I1} \sup \left\{-y_1 \colon \ -y_1 t - y_2 t^2 \leq -t,\ \forall t \in [0,1]\right\}.$$

Here $v_{I1} = -1$, but $v_{II1} = +\infty$ reflecting the inconsistency of the dual program.

During the 1970s affine perturbations of the b-vector were constructed of the form $b + \epsilon\, v$, where $\epsilon > 0$, $v \in \mathbf{R}^n$ leading naturally to the following dual programs.

$$\mathrm{I}_v^\epsilon \quad \begin{cases} v_{I_v^\epsilon} & = \sup y^\top (b + \epsilon v) \\ \text{s.t.} & y^\top u(t) \leq u_{n+1}(t), \quad \forall t \in T \end{cases}$$

and

$$\mathrm{II}_v^\epsilon \quad \begin{cases} v_{II_v^\epsilon} & = \inf\limits_{t \in \operatorname{supp} \lambda} u_{n+1}(t)\lambda(t) \\ \text{from among} & \lambda \in \mathbb{R}^{(S)}, \quad \lambda(\cdot) \geq 0, \\ \text{s.t.} & \sum\limits_{t \in \operatorname{supp} \lambda} u(t)\lambda(t) = b + \epsilon v. \end{cases}$$

The idea was that $\mathbf{II}_v^\epsilon$ would suggest a Program *II** that would be in perfect duality with Program *I*. To accomplish this, certain properties were required of the vector v used in the perturbation.

Definition 3 A vector $v \in \mathbf{R}^n$ that satisfies the following property is termed a *descent vector* for Program *I*: given any $\epsilon > 0$, there exists a generalized finite sequence $\lambda^\epsilon(\cdot)$ which is feasible for Program II_v^ϵ such that in addition,

$$\sum_t u_{n+1}(t)\lambda^\epsilon(t) \leq v_I. \tag{2}$$

To see how a descent vector is used, let $\delta > 0$ but otherwise arbitrary. Let $y^{(\delta)}$ be *I*-feasible such that

$$v_I \geq b^\top y^{(\delta)} \geq v_I - \frac{\delta}{2}. \tag{3}$$

Let $\epsilon < \delta/2\, \| y^{(\delta)\intercal}\, v\, \|$, where without loss of generality $y^{(\delta)\intercal} v \neq 0$). By Definition 3 there exists $\lambda^\epsilon(\cdot)$ which is II_v^ϵ-feasible. Hence by the duality inequality,

$$\sum_t u_{n+1}(t)\lambda^\epsilon(t) \geq y^{(\delta)\top} b + \epsilon y^{(\delta)\top} v. \tag{4}$$

Combining (2)–(4) yields

$$v_I \geq \sum_t u_{n+1}(t)\lambda^{\epsilon}(t) \geq v_I - \frac{\delta}{2} - \frac{\delta}{2} = v_I - \delta\,. \tag{5}$$

The interpretation of (5) is that we can obtain a solution of II_v^{ϵ} with objective function value as close to v_I as desired. The *formal* perfect dual to Program I is the following one.

Program II^*. inf$w$ from $w \in \mathbf{R}$, $(v, v_{m+1}) \in \mathbf{R}^{m+1}$ which satisfy the following constraints: given any $\epsilon > 0$, $\exists \delta$, $0 < \delta \leq \epsilon$ such that

$$\sum_{t \in \operatorname{supp}\mu} u_i(t)\mu(t) = c_i + \delta v_i, \quad i = \overline{1,m},$$
$$\sum_{t \in \operatorname{supp}\mu} u_0(t)\mu(t) - w \leq \delta v_{m+1}$$

has a solution $\mu \in \mathbf{R}^{(B)}$.

Rather than pursue duality theory in depth, we look to newly discovered relationships between of LSIP and semidefinite programming, SDP, see [14,15,18,19].

Dual Semidefinite Programs

We begin with the following standard primal SDP program:

$$P \quad \begin{cases} \sup & c^{\top}x \\ \text{s.t.} & \sum_{i=1}^{m} x_i Q_i \preceq Q_0, \end{cases} \tag{6}$$

where Q_i, $i = \overline{0,m}$, are $\overline{n} \times \overline{n}$ symmetric matrices.

Remark 4 It is convenient to denote the value of a program (P) by $v(P)$, and the value of a particular feasible point/solution, say x, simply by $v(x)$.

$$D \quad \begin{cases} \inf & U \bullet Q_0 \\ \text{s.t.} & U \bullet Q_i = c_i, \quad i = \overline{1,m}, \\ & U \succeq 0. \end{cases} \tag{7}$$

For Programs P and D, we have the following weak duality lemma, also termed the *duality inequality*.

Lemma 5 *If U is D-feasible and x is P-feasible, then $v(x) \leq v(U)$.*

Now let $\mathcal{V}_j$, $j = \overline{1,n}$, be an orthonormal basis for the linear space $\mathbf{S}^{\overline{n}}$ (of $\overline{n} \times \overline{n}$ symmetric matrices, $n = \frac{\overline{n}}{2} \times (\overline{n}+1)$) with inner product $U \bullet W \equiv \langle U, W \rangle$ = trace UW, where $U, W \in \mathbf{S}^{\overline{n}}$, and UW denotes ordinary, dimension compatible matrix multiplication.

Let

$$Q_0 = \sum_{j=1}^{n} b_j \mathcal{V}_j \equiv b^{\top}\mathcal{V},$$
$$Q_i = \sum_{j=1}^{n} a_{ij}\mathcal{V}_j, \quad A = (a_{ij})_{m \times n},$$
$$U = \sum_{j=1}^{n} y_j \mathcal{V}_j \equiv y^{\top}\mathcal{V}.$$

Definition 6 The convex cone $\mathbf{K} \subset \mathbf{R}^n$ is defined with respect to the given orthonormal basis by:

$$z = \{z_j\} \in \mathbf{K} \quad \text{if and only if} \quad z^{\top}\mathcal{V} \succeq 0.$$

The dual cone is conveniently expressed as $\mathbf{K}^* = \{s \in \mathbf{R}^n : s^{\intercal}\mathbf{K} \subseteq \mathbf{R}_+, \mathbf{R}_+$ is the set of nonnegative reals. Here, $\mathbf{K} = \mathbf{K}^*$. We also use the terminology appearing in [16, Sec. 1.5.1] namely:

$$Q^*\colon \mathbf{S}^{\overline{n}} \to \mathbb{R}^m,$$
$$Q^*(U) = (U \bullet Q_i)_{i=\overline{1,m}},$$
$$Q^{\#}\colon \mathbf{S}^{\overline{n}} \to \mathbb{R}^{m+1},$$
$$Q^{\#}(U) = (Q_0 \bullet U, Q^*(U))^{\top}.$$

Perfect Duality from the View of Linear Semi-Infinite Programming

Program (P) is equivalent to the following LSIP:

$$\begin{cases} \sup & c^{\top}x \\ \text{s.t.} & \sum_{i=1}^{m} x_i u_i(t) \leq u_0(t), \quad \forall t \in \mathbf{B}, \end{cases} \tag{8}$$
$$\mathbf{B} \equiv \left\{t \in \mathbb{R}^{\overline{n}} : \|t\| = \|t\|_2 = 1\right\}$$
$$\text{where} \quad u_i(t) = t^{\top}Q_i t, \quad i = \overline{0,m}.$$

As reviewed in [11 Sec. 6.4], we have the following perfect dual, D_G:

$$\begin{aligned} &v(D_G) = \inf w; \\ &w \in \mathbb{R} \\ &\text{s.t. } (c, w)^\top \in \text{cl}(M_{m+1}), \\ &\text{where } M_{m+1} \\ &= \text{co}(\{(u_1(t), \ldots, u_m(t), u_0(t))^\top\}_{t \in \mathbf{B}}) \subset \mathbb{R}^{m+1}. \end{aligned} \tag{9}$$

Rather than considering arbitrary paths in M_{m+1} converging to $(c, w)^\intercal$, we need only consider movement along a *descent ray*, [13], associated with the minimization of Program D^*, which generates straight line paths, (9), namely:

$$D^* \quad \begin{cases} \inf & w; \\ & w \in \mathbb{R} \text{ and } \begin{pmatrix} v & v_{m+1} \end{pmatrix}^\top \in \mathbb{R}^{m+1} \end{cases}$$

which satisfy the condition:

$$\begin{cases} \text{given} & \epsilon > 0, \\ \exists \mu \in \mathbb{R}^{(\mathbf{B})}, & \mu \geq 0, \\ \text{s.t.} & \displaystyle\sum_{t \in \text{supp}\, \mu} u_i(t)\mu(t) = c_i + \epsilon v_i, \\ & i = \overline{1, m}, \\ & \displaystyle\sum_{t \in \text{supp}\, \mu} u_0(t)\mu(t) - w \leq \epsilon v_{m+1}. \end{cases} \tag{10}$$

We need another program using descent vectors (for minimization) that is visibly close to the SDP structure:

$$D_{DV} \quad \begin{cases} \inf & (U + W) \bullet Q_0, \\ \text{where} & U, W \in \mathbf{S}^{\overline{n}}, \quad U \succeq 0, \end{cases}$$

satisfying: given any $\kappa > 0$, $\exists \epsilon_\kappa > 0$:

$$\epsilon_\kappa > \delta > 0 \Rightarrow \exists U_\delta : \quad Q_i \bullet (U_\delta + \delta W) = c_i, \quad i = \overline{1, m} \tag{11}$$

$$Q_0 \bullet (U_\delta + \delta W) - \kappa \leq Q_0 \bullet (U + W). \tag{12}$$

Definition 7 We say that V is *feasible for* D_{DV} if there are $U, W \in \mathbf{S}^{\overline{n}}$ with $U \succcurlyeq 0$, such that $V = U + W$, and the constraints of D_{DV} are satisfied.

Our first goal is to show that Programs D^* and D_{DV} are equivalent. We need two lemmas for this task.

Lemma 8 *If $\mu \in \boldsymbol{R}^{(\boldsymbol{B})}$, $\mu \geq 0$, then $\exists$a PSD U such that*

$$\sum_t u_i(t)\mu(t) = Q_i \bullet U, \quad i = \overline{0, m}.$$

Conversely, if U is PSD, then $\exists \mu \in \boldsymbol{R}^{(\boldsymbol{B})}$, $\mu \geq 0$, such that

$$\sum_t u_i(t)\mu(t) = Q_i \bullet U, \quad i = \overline{0, m}.$$

Proof Let supp $\mu(t) = \{t_1, \ldots, t_p\}$ with $t_j^\top = (t_{j1}, \ldots, t_{j\overline{n}})$. Then

$$u_i(t_j) = t_j^\top Q_i t_j = Q_i \bullet U_j, \tag{13}$$

where $U_j = (t_{jk} t_{jl})_{k,l=\overline{1,n}}$.

Set $U = \sum_{j=1}^p \mu(t_j)\, U_j$. Since each $U_j \succcurlyeq 0$, $j = \overline{1, p}$, it follows that

U is PSD, and

$$Q_i \bullet U = \sum_{t \in \text{supp}\, \mu} u_i(t)\mu(t), \quad i = \overline{0, m}.$$

Conversely, if U is PSD, then there exists $B \in \mathbf{S}^{\overline{n}}$ such that

$$B^\top = B^{-1} \text{ with } BXB^{-1} = U, \quad X \text{ diagonal}.$$

Therefore,

$$\begin{aligned} &Q_i \bullet U = Q_i \bullet BXB^{-1} = \text{tr}(Q_i BXB^{-1}) \\ &= \text{tr}(B^{-1} Q_i BX) = B^{-1} Q_i B \bullet X. \end{aligned}$$

Let

$$X = \text{diag}\{\lambda_1, \ldots, \lambda_{\overline{n}}\}, \quad \lambda_j \geq 0, \quad j = \overline{1, \overline{n}}.$$

Then

$$Q_i \bullet U = \text{tr}(B^{-1} Q_i BX) = \sum_{j=1}^{\overline{n}} \theta_j^\top B^{-1} Q_i B \theta_j,$$

where θ_j has $\sqrt{\lambda_j}$ in its jth component, and zero elsewhere.

Let

$$t_j = \begin{cases} 0 & \text{if } \lambda_j = 0, \\ \dfrac{\theta_j^\top B^{-1}}{\left\| \theta_j^\top B^{-1} \right\|} & \text{if } \lambda_j \neq 0 \end{cases}$$

and

$$\mu(t) = \begin{cases} 0 & \text{if } t \notin \{t_1, \ldots, t_{\overline{n}}\}, \\ \left\| \theta_j^\top B^{-1} \right\|^2 & \text{if } t = t_j, \quad j \in \overline{1, \overline{n}}. \end{cases}$$

Hence, we obtain,

$$Q_i \bullet U = B^{-1} Q_i B \bullet X = \sum_{t \in \operatorname{supp} \mu} \mu(t) u_i(t).$$

For the next result we will need the following program:

$$D^{**} \begin{cases} \inf & w; \\ \text{where} & w \in \mathbb{R} \quad \text{and} \quad W \in \mathbf{S}^{\overline{n}} \quad \text{satisfy:} \end{cases}$$

given any $\epsilon > 0$, $\exists U_\epsilon \in \mathbf{S}^{\overline{n}}$, $U_\epsilon \geq 0$:

$$Q_i \bullet (U_\epsilon + \epsilon W) = c_i, \quad i = \overline{1, m}, \tag{14}$$

$$Q_0 \bullet (U_\epsilon + \epsilon W) \leq w. \tag{15}$$

Lemma 9 *Program D^* is equivalent to Program D^{**}.*

Proof Let D^{**} be consistent with feasible point $w \in \mathbf{R}$, $W \in \mathbf{S}^{\overline{n}}$. Simply use Lemma 8 and set

$$Q_i \bullet W = -v_i, \quad i = \overline{1, m},$$
$$Q_0 \bullet W = -v_{m+1}.$$

Hence, any feasible solution of D^{**} is feasible for D^*.

Suppose now $(v, v_{m+1}, w)^\intercal$ is feasible for D^*. By Lemma 8, for any $\epsilon > 0$, $\exists U_\epsilon$ such that

$$Q_i \bullet U_\epsilon = c_i + \epsilon v_i, \quad i = \overline{1, m}, \tag{16}$$

$$Q_0 \bullet U_\epsilon - w \leq \epsilon v_{m+1}. \tag{17}$$

Therefore, for any $\epsilon > 0$ the equation $AY = c + \epsilon v$ has solutions and so does $AY = -v$, (A is the matrix defined above Definition 6), so $\exists W \in \mathbf{S}^{\overline{n}}$ such that $Q^*(W) = -v$. We consider two cases.

1) $Q_0 \in$ linear span $\{Q_i\}_{i=1}^m$.
   Suppose $Q_0 = \sum_{i=1}^m a_i Q_i$. Then set $w' = \sum_{i=1}^m a_i c_i$. It follows from (17) that $w' \leq w$ and (W, w') is feasible for D^{**}.
2) Q_0 is linearly independent of $\{Q_i\}_{i=1}^m$.
   For this case it follows that there exists $Y_1, Y_2 \subset \mathbf{R}^n$ such that,

$$\begin{pmatrix} A \\ b^\intercal Y_1 \end{pmatrix} Y_1 = \begin{pmatrix} 0 \\ -v_{m+1} \end{pmatrix}$$

and

$$\begin{pmatrix} A \\ b^\intercal Y_1 \end{pmatrix} Y_2 = \begin{pmatrix} -v \\ 0 \end{pmatrix}.$$

Therefore, $\exists W_i \in \mathbf{S}^{\overline{n}}$, $i = 1, 2$, such that

$$Q^*(W_1) = 0, \qquad Q^*(W_2) = -v,$$
$$Q_0 \bullet W_1 = -v_{m+1}, \qquad Q_0 \bullet W_2 = 0.$$

Hence, $(W_1 + W_2, w)$ is feasible for D^{**}.

Since the objective functions are identical, namely w, it follows that D^* is equivalent to D^{**}.

Theorem 10 *Programs D^* and D_{DV} are equivalent.*

Proof It suffices to show that D_{DV} and D^{**} are equivalent. We begin by supposing that D_{DV} is feasible with feasible solution V. This means $\exists U \succcurlyeq 0$, $W \in \mathbf{S}^{\overline{n}}$ satisfying (11) and (12). Let $w_n = Q_0 \bullet (U + W) + 1/n$. Then (W, w_n) is feasible for D^{**} and $v(D_{DV}) \geq v(D^{**})$.

Conversely, suppose D^{**} is feasible with feasible solution, (W, w). Then $\exists U$ such that

$$Q_i \bullet (U + W) = c_i$$
$$Q_0 \bullet (U + W) \leq w, \quad i = \overline{1, m}.$$

1) $Q_0 \in$ linear span $\{Q_i\}_{i=1}^m$.
   In this case $Q_0 \bullet (U + W)$ is determined by $\{Q_i \bullet (U + W)\}_{i=1}^m$. Therefore, $U + W$ is feasible for D_{DV} and $v(D_{DV}) \leq v(D^{**})$.
2) Q_0 is linearly independent of $\{Q_i\}_{i=1}^m$.
   In this case there exists $W_0 \in \mathbf{S}^{\overline{n}}$ satisfying

$$Q_i \bullet W_0 = 0, \qquad Q_0 \bullet W_0 = 1.$$

Now, for any $\kappa > 0$, $\exists M > 0$, $M\kappa = w - Q_0 \bullet (U + W)$. Hence,

$$w = Q_0 \bullet (U + W + M\kappa W_0)$$

and

$$Q_i \bullet (U + W + M\kappa W_0) = c_i, \quad i = \overline{1, m}.$$

But for any $\delta < 1/M$ ($\equiv \epsilon$), $\exists U_\delta \succcurlyeq 0$ U_δ satisfies (14) and 15.

Hence, $Q_i \bullet (U_\delta + \delta(W + M\kappa W_0)) = c_i$, $i = \overline{1, m}$, and

$$Q_0 \bullet (U_\delta + \delta(W + M\kappa W_0))$$
$$= Q_0 \bullet (U_\delta + \delta W) + \delta M\kappa \leq w + \delta M\kappa.$$

Therefore,

$$Q_0 \bullet (U_\delta + \delta(W + M\kappa W_0)) - \delta M\kappa \leq Q_0 \bullet (U + (W + M\kappa W_0)).$$

So

$$Q_0 \bullet (U_\delta + \delta(W + M\kappa W_0)) - \kappa \leq Q_0 \bullet (U + (W + M\kappa W_0)).$$

Therefore, $(U + W + (w - Q_0 \bullet (U + W))\, W_0)$ is a feasible solution for D_{DV} and it is easy to show that $\nu(D_{DV}) = \nu(D^{**})$. Hence, in conclusion we have obtained the following equivalence:

$$D_{DV} \Leftrightarrow D^{**} \Leftrightarrow D^*.$$

Corollary 11 *Program P and Program D_{DV} are in perfect duality.*

The SDP as a Conic Convex Program

Recently, M.V. Ramana [16], Zhi-Quan Luo, Jos.F. Sturm, and Shuzhong Zhang [15], and J.F. Strum [18] developed a perfect dual of the semidefinite program *P*, (6) involving only linear and positive definiteness constraints. The authors [15] state that their regularized program coincides with Ramana's *extended Lagrange-Slater dual* and that this fact was already recognized in [17]; see also [1,2]. In [14] we investigate these connections by using results for regularized conic convex programs. We develop some relationships between all the perfect duals including the one presented earlier in this paper, II*. We refer the interested reader to all of these papers, but here conclude by showing how to restate the SDP program *P* as a conic convex program.

Earlier the convex cone $\mathbf{K}$ and the matrix A were introduced. Clearly, Program *P*, (6), is equivalent to:

$$\begin{cases} \sup & c^\top x \\ \text{s.t.} & b - A^\top x \in \mathbf{K}^* \quad (= \mathbf{K}) \end{cases} \tag{18}$$

with dual program,

$$\begin{cases} \inf & b^\top y \\ \text{s.t.} & Ay = c, \quad y \in \mathbf{K}. \end{cases} \tag{19}$$

Program *P* is equivalent to the following conic convex program:

$$\text{P}' \quad \begin{cases} \inf & (0, c)^\top (s, t) \\ \text{s.t.} & (s, t) \in \mathbb{R}^{n+m}, \\ & (s, t) - (b, 0) = (A^\top x, x), \\ & \exists x \in \mathbb{R}^m \\ \text{and} & (s, t) \in \mathbf{K}^* \times \mathbb{R}^m = \mathbf{K} \times \mathbb{R}^m. \end{cases}$$

Program *D* is equivalent to:

$$\text{D}' \quad \begin{cases} \inf & (b, 0)^\top (y, u) \\ \text{from among} & (y, u) \in \mathbb{R}^{n+m} \\ \text{s.t.} & (y, u) - (0, c) \in \ker[A\ I] \\ \text{and} & (y, u) \in \mathbf{K} \times \{0\}. \end{cases}$$

Program D' is a conic convex program which in the notation used in [15] becomes:

$$CP((0, c), (b, 0), \ker[A\ I], \mathbf{K} \times \{0\}).$$

In [15] a finite sequence of regularized programs is constructed having certain properties that lead to the construction of an SDP perfect dual.

See also

- Duality for Semidefinite Programming
- Interior Point Methods for Semidefinite Programming
- Semidefinite Programming and Determinant Maximization
- Semidefinite Programming: Optimality Conditions and Stability
- Semidefinite Programming and Structural Optimization
- Semi-infinite Programming: Approximation Methods
- Semi-infinite Programming and Control Problems
- Semi-infinite Programming: Discretization Methods
- Semi-infinite Programming: Methods for Linear Problems
- Semi-infinite Programming: Numerical Methods
- Semi-infinite Programming: Second Order Optimality Conditions
- Solving Large Scale and Sparse Semidefinite Programs

References

1. Borwein JM, Wolkowicz H (1981) Characterizations of optimality for the abstract convex program with finite dimensional range. J Austral Math Soc (Ser A) 30:390–411
2. Borwein JM, Wolkowicz H (1981) Regularizing the abstract convex program. J Math Anal Appl 83:495–530
3. Charnes A, Cooper WW, Kortanek KO (1962) Duality, Haar programs and finite sequence spaces. Proc Nat Acad Sci USA 48:782–786
4. Charnes A, Cooper WW, Kortanek KO (1962) A duality theory for convex programs with convex constraints. Bull Amer Math Soc 68:605–608
5. Charnes A, Cooper WW, Kortanek KO (1963) Duality in semi-infinite programs and some works of Haar and Caratheodory. Managem Sci 9:208–228
6. Charnes A, Cooper WW, Kortanek KO (1965) On representation of semi-infinite programs which have no duality gaps. Managem Sci 12:113–121
7. Fiacco AV, Kortanek KO (eds) (1983) Semi-infinite programming and applications. Lecture Notes Economics and Math Systems. Springer, Berlin
8. Gustafson SÅ, Kortanek KO (1983) Semi-infinite programming and applications. In: Bachem A, Grötschel M, Korte B (eds) Mathematical Programming the State of the Art Bonn 1982. Springer, Berlin, pp 132–157
9. Gustafson S-å, Kortanek KO, Rom WO (1970) Non-Chebysevian moment problems. SIAM J Numer Anal 7:335–342
10. Hettich R (ed) (1979) Semi-infinite programming. Lecture Notes Control Inform Systems. Springer, Berlin
11. Hettich R, Kortanek KO (1993) Semi-infinite programming: theory, methods, and applications. SIAM Rev 35:380–429
12. Kortanek KO (1976) Perfect duality in generalized linear programming. In: Prékopa A (ed) Proc. IX Internat. Symp. on Math. Program., North Holland and Publ. House Hungarian Acad. Sci., 43–58
13. Kortanek KO (1977) Constructing a perfect duality in infinite programming. Appl Math Optim 3:357–372
14. Kortanek KO, Zhang Qinghong (2001) Perfect duality in semi-infinite and semidefinite programming. Math Program 91 1(127–144
15. Luo Zhi-Quan, Sturm JF, Zhang Shuzhong (Apr. 1997) Duality results for conic convex programming. Techn. Report Econometric Inst. Erasmus Univ. 9719/A
16. Ramana MV (1997) An exact duality theory for semidefinite programming and its complexity implications. Math Program B 77:129–162, Semidefinite Programming, Edited by Michael Overton and Henry Wolkowitz.
17. Ramana MV, Tunçel L, Wolkowicz H (1997) Strong duality for semidefinite programming. SIAM J Optim 7:641–662
18. Sturm JF (Sept. 1997) Primal-dual interior point approach to semidefinite programming. PhD Thesis Erasmus Univ. Rotterdam, The Netherlands,), Tinbergen Inst. Res. Ser. vol 156, Thesis Publ., Amsterdam, The Netherlands, 1997.
19. Vandenberghe L, Boyd S (1996) Semidefinite programming. SIAM Rev 38:49–95

Sensitivity Analysis of Complementarity Problems

GEORGE J. KYPARISIS
Department Decision Sci. and Information Systems
College of Business Administration,
Florida Internat. University, Miami, USA

MSC2000: 90C31, 90C33

Article Outline

Keywords
See also
References

Keywords

Complementarity problem; Sensitivity analysis

Sensitivity analysis has been extensively developed for a variety of problems in mathematical programming and applied mathematics [2]. It is concerned, in general, with investigation of properties of solutions to problems with perturbed data. This type of analysis derives its importance from the fact that almost all problems in mathematical programming and applied mathematics are solved for a fixed, specified set of data. As a result, the computed solution may be considerably inaccurate or even infeasible if the data are subjected to changes.

Nonlinear complementarity problems were originally formulated in the context of mathematical programming and then shown to be a special case of variational inequality problems ([1,6]). *Linear complementarity problems* were introduced earlier as extensions of convex quadratic programming problems (and bimatrix games) [1]. Initial sensitivity analysis results were developed for linear complementarity problems by extending corresponding results from linear and quadratic programming [1]. In recent years it has been widely recognized that complementarity problems generalize many economic, game-theoretic and transportation *equilibrium problems* ([1,6,11]). These equilibrium

problems were modeled as finite-dimensional complementarity problems (and variational inequalities) and the issues of computational efficiency and sensitivity analysis became an important part of their analysis. The computation of equilibria results in predictions of the behavior of the underlying system. As a result, the sensitivity of the equilibrium solution to changes in model parameters and data needs to be analyzed by the modeler.

The (finite-dimensional) *parametric nonlinear complementarity problem* is defined as

$$\text{NCP}(\epsilon) \quad \begin{cases} \text{find} & x \\ \text{s.t.} & x \geq 0, \\ & F(x,\epsilon) \geq 0, \\ & F(x,\epsilon)^\top x = 0, \end{cases}$$

where F is a mapping from $\mathbf{R}^n \times \mathbf{R}^k$ to $\mathbf{R}^n$ and $\epsilon \in \mathbf{R}^k$ is the parameter vector. Observe that NCP(ϵ) is equivalent to the parametric variational inequality VI(ϵ) with mapping F and the fixed feasible set $X(\epsilon) = K = \{x: x \geq 0\}$. The *parametric linear complementarity problem* is defined as

$$\text{LCP}(M,q) \quad \begin{cases} \text{find} & x \\ \text{s.t.} & x \geq 0, \\ & Mx + q \geq 0, \\ & (Mx+q)^\top x = 0, \end{cases}$$

where M is an $n \times n$ matrix and $q \in \mathbf{R}^n$ (both M and q are considered parameters here).

Let S be the solution point-to-set map for NCP(ϵ) which assigns to each parameter $\epsilon \in \mathbf{R}^k$ the set of solutions of NCP(ϵ)

$$S(\epsilon) = \left\{x \geq 0:\ F(x,\epsilon) \geq 0,\ F(x,\epsilon)^\top x = 0\right\}.$$

Similarly, let the solution point-to-set map for LCP(M,q) be defined as

$$S(M,q) = \left\{x \geq 0:\ Mx + q \geq 0,\ (Mx+q)^\top x = 0\right\}.$$

Sensitivity analysis of the parametric nonlinear complementarity problem NCP(ϵ) is concerned with:

- The existence and (local) uniqueness of solutions of NCP(ϵ), that is, investigating whether the solution set $S(\epsilon)$ is nonempty and a singleton.
- Continuity properties of the solution set map S, such as Lipschitz continuity, both for multivalued and single-valued S.
- Differentiability properties of the solution set map S, such as directional or Fréchet differentiability of a single-valued map S and differentiability in a generalized sense of a multivalued map S.

The fundamental sensitivity analysis results were obtained for the NCP(ϵ) problem using the equivalent parametric *generalized equation* of the form ([12,13]):

$$\text{GE}_K(\epsilon) \quad \begin{cases} \text{find} & x \\ \text{s.t.} & 0 \in F(x,\epsilon) + N_K(x), \end{cases}$$

where $N_K(x)$ is the *normal cone* of the set $K = \{x: x \geq 0\}$ at x.

S.M. Robinson [13] showed that under the assumptions that the mapping F is continuously differentiable and that the solution x^* of the unperturbed problem NCP(ϵ^*) is a *regular solution*, the solution set $S(\epsilon)$ of NCP(ϵ) is (locally) nonempty and a singleton (that is, $S(\epsilon) = \{x(\epsilon)\}$) for all ϵ near ϵ^*. In addition, the solution $x(\epsilon)$ is Lipschitz continuous in ϵ. Under the same assumptions it was also shown later that $x(\epsilon)$ is directionally differentiable at ϵ^* [7] and, under further assumptions on K, that $x(\epsilon)$ is Fréchet differentiable at ϵ^* (see surveys in [6,8]). Sensitivity analysis results described up to now for the problem NCP(ϵ) can be directly applied to the perturbed linear complementarity problem LCP(ϵ) where we substitute $M(\epsilon)$ and $q(\epsilon)$ for the matrix M and the vector q in the formulation LCP(M,q).

Further extensions of the sensitivity analysis results for NCP(ϵ) were obtained for situations where the solution set map $S(\epsilon)$ is multivalued. These include conditions for the existence, continuity and differentiability (in a generalized sense) of the point-to-set map $S(\epsilon)$ ([4,9]).

Another approach to the study of perturbed solutions for nonlinear complementarity problems, called *stability analysis*, was proposed in [5]. In this approach, the mapping F itself is the problem parameter (instead of being dependent on a parameter ϵ) and results from degree theory are utilized to obtain conditions for existence and (generalized) continuity of the multivalued solution map S as a function of F. This approach was also applied to linear complementarity problems LCP(M,q) where (M,q) is the problem parameter and

conditions for stability are expressed in terms of properties of the matrix M [1]. Recent extensions of stability analysis for linear and nonlinear complementarity problems, using degree theory, can be found in [3,4].

Sensitivity analysis of nonlinear complementarity problems was applied to several equilibrium problems, including the general spatial price equilibrium model [14] and the *Cournot–Nash oligopolistic equilibrium model* [15]. Another class of problems to which sensitivity analysis results for complementarity problems may be applied are mathematical programs with equilibrium constraints which arise in game theory, bilevel programming, and network design problems [10]. These problems include constraints of the type $x \in S(\epsilon)$ where, in certain cases, $S(\epsilon)$ may be the solution set of a complementarity problem. Computational methods for solving these problems rely on the ability to calculate the (generalized) derivatives of the solution set map $S(\epsilon)$.

See also

- ► Nonlocal Sensitivity Analysis with Automatic Differentiation
- ► Parametric Global Optimization: Sensitivity
- ► Sensitivity Analysis of Variational Inequality Problems
- ► Sensitivity and Stability in NLP
- ► Sensitivity and Stability in NLP: Approximation
- ► Sensitivity and Stability in NLP: Continuity and Differential Stability

References

1. Cottle RW, Pang JS, Stone RE (1992) The linear complementarity problem. Acad. Press, New York
2. Fiacco AV (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Acad. Press, New York
3. Gowda MS, Pang JS (1992) On solution stability of the linear complementarity problem. Math Oper Res 16:77–83
4. Gowda MS, Pang JS (1994) Stability analysis of variational inequalities and nonlinear complementarity problems, via the mixed linear complementarity problem and degree theory. Math Oper Res 19:831–879
5. Ha CD (1987) Application of degree theory in stability of the complementarity problem. Math Oper Res 12:368–376
6. Harker PT, Pang JS (1990) Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications. Math Program 48:161–220
7. Kyparisis J (1986) Uniqueness and differentiability of solutions of parametric nonlinear complementarity problems. Math Program 36:105–113
8. Kyparisis J (1990) Sensitivity analysis for variational inequalities and nonlinear complementarity problems. Ann Oper Res 27:143–174
9. Kyparisis J (1992) Parametric variational inequalities with multivalued solution sets. Math Oper Res 17:341–364
10. Luo ZQ, Pang JS, Ralph D (1996) Mathematical programs with equilibrium constraints. Cambridge Univ. Press, Cambridge
11. Nagurney A (1993) Network economics: A variational inequality approach. Kluwer, Dordrecht
12. Robinson SM (1979) Generalized equations and their solutions, Part I: Basic theory. Math Program Stud 10:128–141
13. Robinson SM (1980) Strongly regular generalized equations. Math Oper Res 5:43–62
14. Tobin RL (1985) General spatial price equilibria: sensitivity analysis for variational inequality and nonlinear complementarity formulations. In: Harker PT (ed) Spatial Price Equilibrium: Advances in Theory, Computation and Application. Lecture Notes Economics and Math Systems, vol 249. Springer, Berlin
15. Tobin RL (1990) Sensitivity analysis for a Cournot equilibrium. Oper Res Lett 9:345–351

Sensitivity Analysis of Variational Inequality Problems

GEORGE J. KYPARISIS
Department Decision Sci. and Information Systems
College of Business Administration,
Florida Internat. University, Miami, USA

MSC2000: 90C31, 65K10

Article Outline

Keywords
See also
References

Keywords

Variational inequalities; Sensitivity analysis

Sensitivity analysis has been extensively developed for a variety of problems in mathematical programming and applied mathematics [2]. It is concerned, in general, with investigation of properties of solutions to

problems with perturbed data. This type of analysis derives its importance from the fact that almost all problems in mathematical programming and applied mathematics are solved for a fixed, specified set of data. As a result, the computed solution may be considerably inaccurate or even infeasible if the data are subjected to changes.

Variational inequality problems were originally developed to model partial differential equations arising from applications in mechanics [5]. These problems were formulated in infinite-dimensional spaces and sensitivity analysis issues were not addressed. However, it has been widely recognized in recent years that variational inequalities are direct generalizations of many economic, game-theoretic and transportation *equilibrium problems* ([4,10]). These equilibrium problems were modeled as finite-dimensional variational inequalities and the issues of computational efficiency and sensitivity analysis became an important part of their analysis. The computation of equilibria results in predictions of the behavior of the underlying system. As a result, the sensitivity of the equilibrium solution to changes in model parameters and data needs to be analyzed by the modeler.

The (finite-dimensional) general *parametric variational inequality problem* is defined as

$$\mathrm{VI}(\epsilon) \quad \begin{cases} \text{find} & x \in X(\epsilon) \\ \text{s.t.} & F(x,\epsilon)^\top (y - x) \geq 0 \\ & \text{for all } y \in X(\epsilon), \end{cases}$$

where F is a mapping from $\mathbf{R}^n \times \mathbf{R}^k$ to $\mathbf{R}^n$, X is a feasible *point-to-set map* from $\mathbf{R}^k$ to $\mathbf{R}^n$ (that is, it assigns to each point ϵ the feasible set $X(\epsilon)$ of VI(ϵ)), and $\epsilon \in \mathbf{R}^k$ is the parameter vector. Let S be the solution point-to-set map which assigns to each parameter $\epsilon \in \mathbf{R}^k$ the set of solutions

$$S(\epsilon) = \left\{ x \in X(\epsilon): \quad \begin{array}{c} F(x,\epsilon)^\top (y - x) \geq 0, \\ \forall y \in X(\epsilon) \end{array} \right\}.$$

Sensitivity analysis of the parametric variational inequality VI(ϵ) problem is concerned with:

- The existence and (local) uniqueness of solutions of VI(ϵ), that is investigating whether the solution set $S(\epsilon)$ is nonempty and a singleton.
- Continuity properties of the solution set map S, such as Lipschitz continuity, both for multivalued and single-valued S.
- Differentiability properties of the solution set map S such as directional or Fréchet differentiability of a single-valued map S and differentiability in a generalized sense of a multivalued map S.

The fundamental sensitivity analysis results were initially obtained for the special VI(ϵ) problem, where $X(\epsilon) = K$ is a fixed (closed and convex) set, using the equivalent parametric *generalized equation* of the form ([12,13]):

$$\mathrm{GE}_K(\epsilon) \quad \begin{cases} \text{find} & x \\ \text{s.t.} & 0 \in F(x,\epsilon) + N_K(x), \end{cases}$$

where $N_K(x)$ is the *normal cone* of the set K at x.

S.M. Robinson [13] showed that under the assumptions that the mapping F is continuously differentiable and that the solution x^* of the unperturbed problem VI(ϵ^*) is *regular*, the solution set $S(\epsilon)$ is (locally) nonempty and a singleton (that is $S(\epsilon) = \{x(\epsilon)\}$) for all ϵ near ϵ^*. In addition, the solution $x(\epsilon)$ is Lipschitz continuous in ϵ. Under an additional assumption that K is polyhedral, it was also shown later that $x(\epsilon)$ is directionally differentiable at ϵ^* and, under further assumptions on K, that $x(\epsilon)$ is Fréchet differentiable at ϵ^* (see the surveys in [4,7]).

Sensitivity analysis results were then extended to the more general VI(ϵ) problem, where $X(\epsilon) = \{x \in \mathbf{R}^n: g(x,\epsilon) \geq 0, h(x,\epsilon) = 0\}$ is defined using mappings g and h. For this problem, the equivalent parametric generalized equation is of the form ([7,7]):

$$\mathrm{GKKT}_0(\epsilon) \quad \begin{cases} \text{find} & (x,u,w) \\ \text{s.t} & 0 \in T(x,u,w,\epsilon) \\ & \quad + N_{\mathbb{R}^n \times \mathbb{R}^m_+ \times \mathbb{R}^p}(x,u,w), \end{cases}$$

where

$$T(x,u,w,\epsilon)^\top = [L_D(x,u,w,\epsilon)^\top, g(x,\epsilon)^\top, h(x,\epsilon)^\top]$$

and $L_D(x,u,w,\epsilon) = F(x,\epsilon) - \nabla_x g(x,\epsilon)^\intercal u + \nabla_x h(x,\epsilon)^\intercal w$. The generalized equation GKKT$_0(\epsilon)$ is equivalent to the system of *generalized Karush–Kuhn–Tucker conditions* similar to those in nonlinear programming [2].

By applying the results obtained for VI(ϵ) with fixed K to GK$KT_0(\epsilon)$, Robinson [13] showed that under the assumptions that the mapping F is once- and the mappings g and h are twice-continuously differentiable and

that the solution (x^*, u^*, w^*) of the unperturbed problem $GKKT_0(\epsilon^*)$ is regular, the solution set $S(\epsilon)$ is (locally) nonempty and a singleton (that is $S(\epsilon) = \{(x(\epsilon), u(\epsilon), w(\epsilon))\}$) for all ϵ near ϵ^*. In addition, the solution $(x(\epsilon), u(\epsilon), w(\epsilon))$ is Lipschitz continuous in ϵ. Under the same assumptions, it was also shown later that $(x(\epsilon), u(\epsilon), w(\epsilon))$ is directionally differentiable at ϵ^* [6] and, under an additional *strict complementarity slackness condition*, that $(x(\epsilon), u(\epsilon), w(\epsilon))$ is Fréchet differentiable at ϵ^* (see the surveys in [4,7]).

Further extensions of the sensitivity analysis results for $VI(\epsilon)$ were obtained for situations where the solution set map $S(\epsilon)$ is multivalued. These include conditions for the existence, continuity and differentiability (in a generalized sense) of the point-to-set map $S(\epsilon)$ ([3,8,11]).

Sensitivity analysis of variational inequalities was applied to a variety of equilibrium problems. These applications include the traffic assignment or network equilibrium model ([10,11]), the general spatial economic equilibrium models ([1,10,14]), and the *spatial competition facility location models* including *spatial price equilibrium* and *Cournot–Nash oligopolistic equilibrium* [15]. Another class of problems to which sensitivity analysis results were extensively applied are mathematical programs with equilibrium constraints which arise in game theory, bilevel programming, and network design problems [9]. These problems include constraints of the type $x \in S(\epsilon)$ where $S(\epsilon)$ is the solution set of a variational inequality $VI(\epsilon)$. Computational methods for solving these problems rely on the ability to calculate the (generalized) derivatives of the solution set map $S(\epsilon)$.

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-nonsmooth Calculus of Variations
- ▶ Nonlocal Sensitivity Analysis with Automatic Differentiation
- ▶ Parametric Global Optimization: Sensitivity
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Complementarity Problems
- ▶ Sensitivity and Stability in NLP
- ▶ Sensitivity and Stability in NLP: Approximation
- ▶ Sensitivity and Stability in NLP: Continuity and Differential Stability
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Dafermos S, Nagurney A (1984) Sensitivity analysis for the general spatial economic equilibrium problem. Oper Res 32:1069–1086
2. Fiacco AV (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Acad. Press, New York
3. Gowda MS, Pang J-S (1994) Stability analysis of variational inequalities and nonlinear complementarity problems, via

the mixed linear complementarity problem and degree theory. Math Oper Res 19:831–879
4. Harker PT, Pang JS (1990) Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications. Math Program 48:161–220
5. Kinderlehrer D, Stampacchia G (1980) An introduction to variational inequalities and their applications. Acad. Press, New York
6. Kyparisis J (1987) Sensitivity analysis framework for variational inequalities. Math Program 38:203–213
7. Kyparisis J (1990) Sensitivity analysis for variational inequalities and nonlinear complementarity problems. Ann Oper Res 27:143–174
8. Kyparisis J (1992) Parametric variational inequalities with multivalued solution sets. Math Oper Res 17:341–364
9. Luo ZQ, Pang JS, Ralph D (1996) Mathematical programs with equilibrium constraints. Cambridge Univ. Press, Cambridge
10. Nagurney A (1993) Network economics: A variational inequality approach. Kluwer, Dordrecht
11. Qiu Y, Magnanti TL (1989) Sensitivity analysis for variational inequalities defined on polyhedral sets. Math Oper Res 14:410–432
12. Robinson SM (1979) Generalized equations and their solutions, Part I: Basic theory. Math Program Stud 10:128–141
13. Robinson SM (1980) Strongly regular generalized equations. Math Oper Res 5:43–62
14. Tobin RL (1985) General spatial price equilibria: sensitivity analysis for variational inequality and nonlinear complementarity formulations. In: Harker PT (ed) Spatial Price Equilibrium: Advances in Theory, Computation and Application. Lecture Notes Economics and Math Systems, vol 249. Springer, Berlin
15. Tobin RL, Friesz TL (1986) Spatial competition facility location models: definition, formulation and solution approach. Ann Oper Res 6:49–74

Sensitivity and Stability in NLP

A. V. Fiacco
Department Operations Res.,
George Washington University,
Washington, DC, USA

MSC2000: 90C31

Article Outline

Keywords
See also
References

Keywords

Nonlinear programming; Linear programming; Optimality conditions; Sensitivity analysis

We are interested in what happens to a solution of a problem when there are changes in the problem data. We shall use the term *sensitivity analysis* to refer to the calculation and study of any collection of measurements associated with any changes, infinitesimally small or finite. The measure of change may be explicit or implicit, qualitative or quantitative. For example, a qualitative characteristic may be the continuity of a change, while a quantitative one might be a numerical bound. An explicit measure is usually a closed form expression while an implicit one is inferred by a proof of existence of a property or quantity. Stable will mean 'well-behaved' or 'well-conditioned', in a given context. Traditionally, *well-conditioned* generally meant small solution changes from small data changes, but many subjective interpretations are possible and currently in use. The concept is relative to a given property, data value or data change. Stochastic changes and effects are certainly possible and frequently encountered in applications, but our focus is deterministic.

Some familiar examples will hopefully clarify the kinds of problems and conclusions that are of interest. Suppose $Ax = b$, where A is an $n \times n$ matrix and x and b are in E^n, n-dimensional Euclidean space. If A is nonsingular, then we know that $x = A^{-1}b$. Now, if A or b changes, we can track the change in x if we know A^{-1}. We have a closed form smooth stable explicit solution $x(A; b)$ of x as a function of A and b. Of course, computationally, we may even here encounter serious difficulties, since A^{-1} or b may be difficult to evaluate. For example, suppose that A and b are functions of some parameter $\epsilon \in E^q$, which we denote by $A(\epsilon)$ and $b(\epsilon)$. If A remains nonsingular and b remains defined for $\epsilon \in T \subseteq E^q$, then $x(\epsilon) = A^{-1}(\epsilon)b(\epsilon)$ for $\epsilon \in T$. In this instance, at least we know the general form of the parametric solution, but even so a closed form expression $x(\epsilon)$ of x as a function of ϵ might be too difficult to obtain. Issues become more complicated, e. g., for what ϵ is x continuous, differentiable, rational, bounded, etc.? Can we calculate x and its derivatives or bounds, relative to ϵ, or at least make good estimates? If A is singular or b is not in the range of A, then $x(\epsilon)$ may not exist or may be mul-

tivalued. Can we characterize the solution set $S(\epsilon)$ for $\epsilon \in T$?

This example suggests that we are thinking of a solution as a function of the problem data, though the function may be extremely complicated, multivalued and only implicitly defined. Our central objective is to characterize this functional relationship. Why? Because, inevitably, data are 'soft', subject to error or change, uncertain. It is usually desirable to approximate new solutions and the effects of data changes, once a solution has been calculated, without requiring the calculation of a solution for each new data value.

A closed form solution will generally not be available, except at an aggregate level for simple models such as that noted above. There are many such models in the sciences, in effect whenever we have a formula for a quantity as a closed form expression. The reader will know many examples (e. g., the quadratic formula for the solution of $ax^2 + bx + c = 0$ where $x, a, b, c \in E^1$, $A = \pi r^2$, $F = ma$, $E = mc^2$, etc.), but perhaps may not have focused on viewing these formulas as representations of solutions of parametric problems, in effect solutions in terms of problem data. This is our orientation here, albeit as noted, we will not usually have the luxury of a closed form solution at our disposal.

We are interested in a particular problem formulation that has many varied and important applications. The model is the nonlinear programming (NLP) problem that is defined as follows: Find a vector x that solves

$$\text{(P)} \quad \begin{cases} \min\limits_{x} & f(x) \\ \text{s.t.} & g_i(x) \geq 0 \quad (i = 1, \ldots, m), \\ & h_j(x) = 0 \quad (j = 1, \ldots, p). \end{cases}$$

Since our focus is on the effect of data changes on x, we explicitly introduce a vector ϵ that represents the data and study the problem of finding a solution $x(\epsilon)$ of

$$\text{P}(\epsilon) \quad \begin{cases} \min\limits_{x} & f(x, \epsilon) \\ \text{s.t.} & g_i(x, \epsilon) \geq 0 \quad (i = 1, \ldots, m), \\ & h_j(x, \epsilon) = 0 \quad (j = 1, \ldots, p). \end{cases}$$

We assume that $x \in E^n$ (Euclidean n-space), though all the results we mention have extensions to more general spaces, and we assume $\epsilon \in T$, a subset of E^q. In this framework, when ϵ is fixed, the parametric NLP P(ϵ) becomes a standard NLP of the form (P). We are interested in a solution $x(\epsilon)$, the 'optimal value' $f[x(\epsilon), \epsilon]$ and other quantities, when ϵ changes.

Some simple examples may help to illustrate the problem. Suppose we have the linear program:

$$\begin{cases} \min\limits_{x} & x \\ \text{s.t.} & x \geq \epsilon \\ & x \in E^1, \quad \epsilon \in E^1. \end{cases}$$

Then, the solution $x(\epsilon) = \epsilon$ and the optimal value $f[x(\epsilon), \epsilon] = x(\epsilon) = \epsilon$, where ϵ can be any number. This situation is rather ideal: the solution is defined and unique for every ϵ, the solution and optimal value are infinitely differentiable, in this case linear in ϵ. Everything is completely stable.

As another example of a very well-behaved problem, now an NLP in E^2, consider:

$$\begin{cases} \min\limits_{x} & (x_1 - \epsilon_1^2)^2 + (x_2 - |\epsilon_2|)^2 \\ \text{s.t.} & x_1,\ x_2 \geq 0, \\ & x \in E^2, \quad \epsilon \in E^2. \end{cases}$$

It is easy to see that $x(\epsilon) = [x_1(\epsilon), x_2(\epsilon)] = (\epsilon_1^2, |\epsilon_2|)$ and $f[x(\epsilon), \epsilon] = 0$ for any $\epsilon \in E^2$. Here again, the solution $x(\epsilon)$ is a well-defined unique continuous function for all ϵ in E^2. The component $x_1(\epsilon)$ is infinitely differentiable, but now $x_2(\epsilon)$ is not differentiable at $\epsilon_2 = 0$. The optimal value is perfectly stable, indeed constant. Again, we have $x(\epsilon)$ and $f[x(\epsilon), \epsilon]$ in closed form.

Unfortunately, things are not always so nice, even for very simple problems. Consider the linear program

$$\text{LP}(\epsilon) \quad \begin{cases} \min\limits_{x} & \epsilon x \\ \text{s.t.} & x \geq 0, \\ & x \in E^1, \quad \epsilon \in E^1. \end{cases}$$

When $\epsilon > 0$, then $x(\epsilon) = 0$ solves this problem and $f[x(\epsilon), \epsilon] = 0$. When $\epsilon = 0$, then $f[x(\epsilon), \epsilon] = 0$ for any $x \in E^1$, hence LP(0) is solved for any $x \geq 0$, so the solution set $S(\epsilon)$ by $\epsilon = 0$ is given by $S(0) = \{x \in E^1 : x \geq 0\}$ and of course $f[x(0), 0] = 0$. When $\epsilon < 0$, things collapse, since the objective function $f(x, \epsilon)$ is unbounded below in the feasible set $R = \{x \in E^1 : x \geq 0\}$. Symbolically, we see that $f(x, \epsilon) = \epsilon x \to -\infty$ as $x \to +\infty$ when $\epsilon < 0$. Here, we say that the infimum (smallest value of f) is 'not attained', nor is the solution (minimizing value of x) attained. Summarizing, the solution is stable for ϵ

> 0, unstable for $\epsilon = 0$, since the solution set $S(0)$ can vanish in every small neighborhood of $\epsilon = 0$, and unsolvable for $\epsilon < 0$.

An equivalent example in E^2 may give more perspective on the geometry of the problem. Consider

$$\begin{cases} \min\limits_{x} & x_2 \\ \text{s.t.} & \epsilon x_1 \le x_2, \quad x_1 \ge 0, x \in E^2, \quad \epsilon \in E^1. \end{cases}$$

The reader is encouraged to sketch the feasible region for $\epsilon > 0, \epsilon = 0$ and $\epsilon < 0$. Clearly, the smallest value of x_2 for $\epsilon > 0$ or $\epsilon = 0$ is $x_2 = 0$, but for $\epsilon < 0$, the component x_2 is not bounded below. The associated values of x_1 are $x_1 = 0$ for $\epsilon > 0$ and any $x_1 \ge 0$ for $\epsilon = 0$ and x_1 unbounded above as $x_2 \to -\infty$ for $\epsilon < 0$. Collecting this information, we have the following: The solution $x(\epsilon) = (0, 0)$ for $\epsilon > 0$; $x(\epsilon)$ is any $x \in S(0) = \{x \in E^2: x_1 \ge 0, x_2 = 0\}$ for $\epsilon = 0$; and the solution is not attained for $\epsilon < 0$. The optimal value $f[x(\epsilon), \epsilon] = 0$ for $\epsilon \ge 0$ and is not bounded below for $\epsilon < 0$. As before, the problem is stable for $\epsilon > 0$, but unstable for $\epsilon = 0$ since the solution does not exist for some ϵ in any neighborhood of $\epsilon = 0$, and unsolvable for $\epsilon < 0$.

What causes such erratic behavior and what guarantees such stability, for different parameter values? Can we identify conditions that imply some sort of stability or that warn of the presence of instability? Can changes in a solution be predicted within a specified error tolerance for problem data changes, without solving the problem again for each new perturbation? A resolution and understanding of such questions is the substance of sensitivity and stability analysis. Note that in our small examples, the problem functions taken individually are continuous and differentiable and extremely well behaved. Clearly, we need to capture the effect of parameter changes on their joint collective behavior, at least in a neighborhood of a solution.

Some of the important tools and concepts that have been especially useful in obtaining conditions and results for sensitivity calculations and stability characterizations are:

i) optimality conditions,
ii) constraint qualifications,
iii) implicit function theorems,
iv) directional derivatives,
v) point-to-set maps and set and map convergence and continuity,
vi) condition number,
vii) convexity and convex analysis, and
viii) duality.

The list is subjective and far from exhaustive. Some of these constructs will be utilized in the preliminary results we mention, but we cannot provide additional coverage in this brief presentation.

We next present a few basis results that illustrate the application of some of the mentioned mathematical tools and indicate typical assumptions and conclusions. Consider the important special realization of our general problem $P(\epsilon)$ when the constraints $g(x) \ge 0$ and $h(x) = 0$ are not present, resulting in the so-called *unconstrained problem*

$$P_1(\epsilon) \quad \begin{cases} \min\limits_{x} & f(x, \epsilon) \\ \text{s.t.} & x \in E^n \quad (\epsilon \in E^q). \end{cases}$$

Assume that f is twice jointly continuously differentiable in (x, ϵ). Let $\nabla_x f(x) = [\, \partial f(x)/\partial x_1, \ldots, \partial f(x)/\partial x_n]$ denote the usual *gradient* of f at x and $\nabla_x^2 f(x) = \nabla[\nabla f(x)^T]$ (with (i, j)th element given by $\partial^2 f(x)/\partial x_i \partial x_j$) define the matrix called the *Hessian* of f at x. (Note that $\nabla_x f$ is an n-dimensional row vector and $\nabla_x^2 f$ is an $n \times n$ matrix.) The following result is well known and the assumptions will be called the *second order sufficient conditions* at (x, ϵ), denoted by SOSC(x, ϵ). Let $\epsilon = \bar{\epsilon}$, a fixed quantity.

Proposition 1 *If $\nabla_x f(\bar{x}, \bar{\epsilon}) = 0$ and $\nabla_x^2 f(\bar{x}, \bar{\epsilon})$ is positive definite, then $\bar{x}$ is a strict and locally isolated minimizer of $f(x, \bar{\epsilon})$.*

This result is of intrinsic interest and is very important and well known. What is especially intriguing is that sensitivity results follow without additional assumptions.

Proposition 2 *Assuming* SOSC$(\bar{x}, \bar{\epsilon})$ *as before, for ϵ near $\bar{\epsilon}$ there exists a unique once continuously differentiable vector function x with value $x(\epsilon)$ such that $x(\bar{\epsilon}) = \bar{x}$ and SOSC$(x(\epsilon), \epsilon)$ continues to hold and hence $x(\epsilon)$ is a locally unique (i. e., isolated) local minimizer of $f(x, \epsilon)$. Furthermore, the optimal value f^* is twice continuously differentiable, where we define $f^*(\epsilon) = f[x(\epsilon), \epsilon]$.*

This may be viewed as a stability result, essentially an existence theorem (following directly from a classical implicit function theorem) that guarantees that the assumptions made at $(\bar{x}, \bar{\epsilon})$ will persist near $(\bar{x}, \bar{\epsilon})$ at $[x(\epsilon),$

$\epsilon]$, hence also the conclusions. This suggests an intuitively appealing definition of an ideal form of stability, with respect to a given change in the data. We might define 'assumption stability' to be the persistence of initial conditions and assumptions in the sense indicated.

Interestingly, we can go even further and derive a sensitivity formula from our observations, again without any new assumptions.

Proposition 3 *Assuming as in Proposition 1 and with $x(\epsilon)$ as in Proposition 2, we obtain the following relations at $[x(\epsilon), \epsilon]$ near $(\overline{x}, \overline{\epsilon})$:*

i) Defining the optimal value $f^*(\epsilon) = f[x(\epsilon), \epsilon]$, we conclude from the chain rule for differentiation that

$$\begin{aligned}\nabla_\epsilon f^*(\epsilon) &= d/d\epsilon[f[x(\epsilon), \epsilon]] \\ &= \nabla_x f \cdot \nabla_\epsilon x + \nabla_\epsilon f = \nabla_\epsilon f(x, \epsilon)|_{x=x(\epsilon)}\end{aligned}$$

since $\nabla_x f[x(\epsilon), \epsilon] = 0$, where $F(y)|_{y=z}$ means to evaluate F at $y = z$. With this understanding, we write

$$\nabla_\epsilon f^*(\epsilon) = \nabla_\epsilon f[x(\epsilon), \epsilon].$$

ii) Differentiating the last result again by ϵ, we find that

$$\nabla^2_\epsilon f^*(\epsilon) = \nabla^2_{x\epsilon} f \nabla_\epsilon x + \nabla^2_\epsilon f|_{x=x(\epsilon)}.$$

iii) Differentiating $\nabla_x f[x(\epsilon), \epsilon] = 0$ by ϵ yields

$$\nabla^2_x f \nabla_\epsilon x + \nabla^2_{\epsilon x} f = 0$$

from which we conclude that

$$\nabla_\epsilon x(\epsilon) = -\nabla^2_x f[x(\epsilon), \epsilon]^{-1} \nabla^2_{\epsilon x} f[x(\epsilon), \epsilon].$$

Here, $\nabla^2_{\epsilon x} f = \nabla_\epsilon(\nabla_x f^\top)$.

Thus, without additional conditions other than the optimality conditions SOSC and the assumption of the presence of data in the form of a parameter vector and appropriate smoothness, we have a characterization of local optimality in Proposition 1, a parametric stability and existence theorem in Proposition 2, and sensitivity measurements in the form of parameter-derivative formulas, for both the optimal value and local minimizing point, in Proposition 3. Note that the derivatives are computable, once we have calculated a solution $x(\epsilon)$ for a given ϵ, hence in particular at $\overline{\epsilon}$ where $x(\overline{\epsilon}) = \overline{x}$. These results are extremely important and useful and they also dramatize some important principles, i. e., the persistence of assumptions as a key to stability, and the intimate connection between optimality conditions and stability and sensitivity analysis. Of course, the conclusions are local, but this is not surprising since the results are based on information at one point. As a postscript, we also note that SOSC is often invoked and generally needed for the optimal convergence and rate of convergence behavior of numerical algorithms, e. g., for the quadratic convergence rate of Newton's method. Thus, we conclude that sensitivity and stability analysis, optimality conditions and characterizations and the convergence properties of computational algorithms are extremely closely related. Their underlying theory can undoubtedly be unified at a general level.

Applications of results such as those given are numerous. Perhaps most directly, we can estimate $f^*(\epsilon)$ and $x(\epsilon)$ for small changes in ϵ by using a truncated Taylor's series as follows:

i) $f^*(\epsilon) \approx f^*(\overline{\epsilon}) + \nabla_\epsilon f^*(\overline{\epsilon})(\epsilon - \overline{\epsilon}) + \frac{(\epsilon-\overline{\epsilon})^\top \nabla^2_\epsilon f^*(\overline{\epsilon})(\epsilon-\overline{\epsilon})}{2}$ where $\nabla_\epsilon f^*$ and $\nabla^2_\epsilon f^*$ are as given above in i) and ii), and

ii) $x(\epsilon) \approx x(\overline{\epsilon}) + \nabla_\epsilon x(\overline{\epsilon})(\epsilon - \overline{\epsilon}) = \overline{x} - \nabla^2_x f^{-1}(\overline{x}, \overline{\epsilon}) \nabla^2_{\epsilon x} f(\overline{x}, \overline{\epsilon})(\epsilon - \overline{\epsilon})$.

These estimates are available when a good approximation to $\overline{x}$ has been obtained.

We conclude this brief expository article by indicating a number of other applications, extensions and references.

There is now an enormous body of literature devoted to sensitivity and stability results in mathematical programming, most of which have been published starting in the 1970s. The issues surrounding well-posedness and resulting definitions, e. g., solvability for small data perturbations, continuous solution changes for continuous data changes, and others are classical however and have been studied in mathematics and physics and other disciplines since early times. A rudimentary general theory has been known for some years. The recent burst of activity was undoubtedly stimulated by the tremendous advances in NLP in the 1960s and 1970s, largely resulting from the preceding emergence and computational demands of modern optimization applications in the sciences, engineering, economics, industry, and collectively what came to be known as *operations research*, the advent of modern computers,

and the development and success of linear programming (LP) methodology.

For LP, developed in the 1940s, there quickly followed what was called *post-optimality sensitivity analysis* that included parametric closed form expressions for solution changes with data changes and error bounds, this having been recently significantly extended. Existence theorems and closed form expressions have also been recently developed for optimal value and solution point parameter-derivatives and directional derivatives, as well as parametric bounds, for the general parametric nonlinear program (NLP) of the form $P(\epsilon)$, analogous to those given in this article for the unconstrained problem $P_1(\epsilon)$. Significant sensitivity and stability results have been obtained for many important and well known classes of problems, e. g., geometric programs, separable programs, semi-infinite and infinite programming, control theory, multi-objective optimization, integer programming and stochastic programming. The parametric perturbation theory has been significantly extended to more general models as well, e. g., variational inequalities and generalized equations. Results include many variations in optimality conditions, constraint qualifications, definitions of continuity and differentiability, generalized convexity and other mathematical notions and tools. A significant body of theory now also extends to nonsmooth (i. e., nondifferentiable) functions, and to perturbations that are more general than perturbation of a parameter, e. g., function and set perturbations.

Applications are abundant. They include solution and optimal value extrapolation for perturbed data, as already noted, model validation, scaling and regularization, decomposition algorithms, bilevel programming, optimization involving implicitly defined functions, duality relations, analysis of convergence properties of algorithms, the effect of data perturbations on algorithmic performance, approximation of sensitivity measurements from algorithmic information and much more. Computational implementations on actual practical real-world problems do exist but are still quite limited. Much remains to be done both in theory and practice, e. g., the provision of standard software for user-friendly sensitivity and stability calculations in major NLP solution computer programs, as available in LP (though user-friendliness may need more emphasis, even here). The theory is rich and every facet of optimality characterization and algorithmic definition and performance is a fertile field for the study of the effect of data perturbations.

The significant contributions to this field are too numerous to mention and thus we shall not attempt to single out key contributors. Instead, we shall cite only [1,2,3], references with which the author has been personally involved, not to presume to monopolize personal credit but because they contain numerous references and state-of-the-art surveys and scholarly papers from many established and emerging leaders in the field. These books and journals and the tutorial and research articles and hundreds of references therein will hopefully quickly and surely lead the interested reader to the central core of the published results in this vital area.

See also

- ► Ill-posed Variational Problems
- ► Nonlocal Sensitivity Analysis with Automatic Differentiation
- ► Parametric Global Optimization: Sensitivity
- ► Sensitivity Analysis of Complementarity Problems
- ► Sensitivity Analysis of Variational Inequality Problems
- ► Sensitivity and Stability in NLP: Approximation
- ► Sensitivity and Stability in NLP: Continuity and Differential Stability

References

1. Fiacco AV (ed) (1990) Optimization with data perturbations. Ann Oper Res 27 Baltzer, Basel
2. Fiacco AV (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Acad. Press, New York
3. Fiacco AV (ed) (1998) Mathematical programming with data perturbations. Lecture Notes Pure and Applied Math, vol 195. M. Dekker, New York

Sensitivity and Stability in NLP: Approximation

A. V. Fiacco
Department Operations Res.,
George Washington University, Washington, USA

MSC2000: 90C31

Article Outline

Keywords
Parameter Derivatives
Applications of Sensitivity Analysis
Computational Efficiencies and AlgorithmicApproximations Computational Efficiencies and Algorithmic Approximations
Computational Efficiencies and AlgorithmicApproximations
Computational Efficiencies and Algorithmic Approximations
Algorithmic Approximations
Bounds
Linear Equations
Solution-Point Bounds for NLP
Computable Optimal Value Bounds f^* Convex
g_i Jointly Concave, h_j Jointly Linear Affine
$P(\epsilon)$Jointly Convex
$\widetilde{f}$ and $\overline{f}$ for Subsets of T
Bounds on the Distance of a Feasible Point to a Solution Point
See also
References

Keywords

Sensitivity analysis; Quadratic programming; Bounds

In this article, we describe a number of (for the most part) computable formulas and bounds that can be used to approximate sensitivity and stability measurements for nonlinear programming (NLP) involving a parameter. We draw freely from the definitions and results presented in ► Sensitivity and stability in NLP and ► Sensitivity and stability in NLP: Continuity and differential stability. The reader is advised to read these before proceeding with the results we present here.

As noted in ► Sensitivity and stability in NLP and ► Sensitivity and stability in NLP: Continuity and differential stability, the problem of interest is to find a solution $x(\epsilon)$ of

$$\mathrm{P}(\epsilon) \quad \begin{cases} \min\limits_{x} & f(x,\epsilon) \\ \text{s.t.} & g_i(x,\epsilon) \geq 0 \quad (i=1,\ldots,m), \\ & h_j(x,\epsilon) = 0 \quad (j=1,\ldots,p), \end{cases}$$

where $x \in E^n$ and the parameter $\epsilon \in T \subseteq E^q$. We denote a local solution by $x(\epsilon)$, the local optimal value by $f^*(\epsilon) = f[x(\epsilon), \epsilon]$, solution set by $S(\epsilon)$, and feasible region by $R(\epsilon)$.

Our focus is on general finite-dimensional deterministic NLP involving a parameter. The reader may know that there is a fairly fine-tuned sensitivity analysis methodology for linear programming (LP), often referred to as *post-optimality analysis*, including a *range-analysis* concerned with maximum tolerances on changes in the objective function coefficients within which a solution and basis does not change, and formulas for right-hand side and constraint coefficient matrix changes, bounds on changes, and the like. Recent work has sharpened the classical theory to allow for more simultaneous data changes and extend the parameter change tolerance idea. Likewise, sensitivity approximation results have been developed for quadratic programs, networks, separable programs, geometric programs, and many other mathematical programs having a special structure. In the direction of more generality, sensitivity and stability approximation results have been developed for semi-infinite programming, infinite programming, control theory, discrete programming, stochastic programming, and many other general programs, extending further to related generalizations suchas variational inequalities, generalized equations, nonsmooth analysis, and more.

We consider here three types of sensitivity and stability measurements at a general level for smooth NLP involving a parameter:

i) parameter-derivatives of a local solution $x(\epsilon)$ and the local optimal value function $f^*(\epsilon) = f[x(\epsilon), \epsilon]$;
ii) algorithmic approximations of solution parameter-derivatives using a classical and well known barrier function (interior point) algorithm; and
iii) parameter-dependent solution bounds.

Parameter Derivatives

In ► Sensitivity and stability in NLP: Continuity and differential stability we defined the *Karush–Kuhn–Tucker conditions* (KKT) at a feasible point x of $\mathrm{P}(\epsilon)$ to be the existence of $u \geq 0$ and w such that (conditions KKT(x, u, w, ϵ)):

$$\begin{cases} \nabla_x L(x,u,w,\epsilon) = 0, & \\ u_i g_i(x,\epsilon) = 0 & (i=1,\ldots,m), \\ h_j(x,\epsilon) = 0 & (j=1,\ldots,p), \end{cases}$$

where

$$L(x,u,w,\epsilon) = f(x,\epsilon) - \sum_{i=1}^{m} u_i g_i(x,\epsilon) + \sum_{j=1}^{p} w_j h_j(x,\epsilon)$$

is the Lagrangian and u, w are called the *Lagrange multipliers* associated with x. Definitions of SOSC, LI and SCS were also given in ► Sensitivity and stability in NLP: Continuity and differential stability, along with the following result, which we repeat here for convenience. For simplicity, assume that the functions defining P(ϵ) are twice jointly continuously differentiable in (x, ϵ) unless otherwise specified.

Proposition 1 *Assume that KKT, SOSC, LI and SCS hold at a feasible point $\overline{x}$ of P($\overline{\epsilon}$) with associated Lagrange multipliers $(\overline{u}, \overline{w})$, where $\overline{\epsilon} \in$ interior T. Then, for ϵ in a neighborhood of $\overline{\epsilon}$ in T we have the following consequences:*

i) There exists a locally unique once continuously differentiable vector function $y = (x, u, w)$ such that the assumptions continue to hold at $y(\epsilon) = [x(\epsilon), u(\epsilon) w(\epsilon)]$, where $y(\overline{\epsilon}) = \overline{y} = (\overline{x}, \overline{u}, \overline{w})$.

ii) The point $x(\epsilon)$ is an isolated local minimizer of problem P(ϵ) with associated unique Lagrange multipliers $[u(\epsilon), w(\epsilon)]$; and

iii) The local optimal value function f^* is twice continuously differentiable in ϵ.

This Proposition follows from the fact that the (x, u, w)-Jacobian (i. e., the matrix of partial derivatives with respect to (x, u, w)) of the KKT equation system given above is nonsingular at $(\overline{x}, \overline{u}, \overline{w}, \overline{\epsilon})$. Thus, a classical implicit function theorem is applicable and the results quickly follow, once it is shown that SOSC persists near $\epsilon = \overline{\epsilon}$.

Remark 2 G.P. McCormick must be credited for initially identifying and applying the conditions, KKT, SOSC, LI and SCS, which imply the existence of the KKT y-Jacobian inverse. He initiated the use of these assumptions to obtain extrapolation results for classical barrier function methods in terms of the algorithmic parameter. See [6, Chap. 5]. He also used these to obtain results similar to those in Proposition 1 but for a problem that has additive perturbations that may be nonlinear in x and linear in ϵ, a special case of problem P(ϵ). See [6, Thm. 6]. The author showed that KKT, SOSC, LI and SCS suffice for results for the general problem P(ϵ)as given in Proposition 1, as well as for the results given in the rest of this article. These developments are pursued in detail in [2]. McCormick also stated a partial converse of Proposition 1 as follows: If KKT and a weakened form of SONC hold and the KKT y-Jacobian has an inverse at $(\overline{x}, \overline{u}, \overline{w})$ (for problem P(ϵ) with ϵ not involved, i. e., ϵ fixed), then SOSC must hold at $(\overline{x}, \overline{u}, \overline{w})$. This is [6, Cor. 7], where *SONC* () the second order necessary condition) here means the second order part of SOSC, weakened to $z^\intercal \nabla_x^2 Lz \geq 0$ for all z as in SOSC. We note that SCS and LI also follow from the assumptions of [6, Cor. 7]. We can show and here note that the following partial converse of Proposition 1 is true and equivalent to [6, Cor. 7] just cited: If $\overline{x}$ is a local minimizer of problem P(ϵ) (with ϵ fixed) and KKT and the KKT y-Jacobian inverse exist at $(\overline{x}, \overline{u}, \overline{w})$, then again all the assumptions of Proposition 1, i. e., KKT, SOSC, LI and SCS, must hold at $(\overline{x}, \overline{u}, \overline{w})$.

To obtain parameter derivatives of (x, u, w), we simply note that KKT(x, u, w, ϵ) is locally identically satisfied at $[x(\epsilon), u(\epsilon), w(\epsilon)]$ near $\epsilon = \overline{\epsilon}$, and hence these equations can be differentiated with respect to ϵ. We have the following (conditions KKT$[x(\epsilon), u(\epsilon), w(\epsilon), \epsilon]$):

$$\begin{cases} \nabla_x L[x(\epsilon), u(\epsilon), w(\epsilon), \epsilon] = 0, & \\ u_i(\epsilon) g_i[x(\epsilon), \epsilon] = 0 & (i = 1, \ldots, m), \\ h_j[x(\epsilon), \epsilon] = 0 & (j = 1, \ldots, p). \end{cases}$$

With F representing the vector function of equalities on the left, this has the form $F[y(\epsilon), \epsilon] = 0$ where $y = (x, u, w)$. Since our assumptions imply that F is differentiable in ϵ, we can use the chain rule for differentiation to conclude that $dF/d\epsilon = \nabla_y F \nabla_\epsilon y + \nabla_\epsilon F = 0$. Applying this calculation to the KKT equations we get the following:

$$M(\epsilon)\nabla_\epsilon y(\epsilon) + \widetilde{N}(\epsilon) = 0,$$

where

$$M = \begin{pmatrix} \nabla_x^2 L & -\nabla g_1^\top & \cdots & -\nabla g_m^\top & \nabla h_1^\top \cdots \nabla h_p^\top \\ u_1 \nabla_x g_1 & g_1 & & 0 & \\ \ddots & & \cdots & 0 & 0 \\ u_m \nabla_x g_m & 0 & & g_m & \\ \nabla h_1 & & & & \\ \ddots & & 0 & & 0 \\ \nabla h_p & & & & \end{pmatrix}$$

is the Jacobian of the KKT equations with respect to y and

$$\widetilde{N} = [\nabla^2_{\epsilon x} L^\top, u_1 \nabla_\epsilon g_1^\top, \ldots, u_m \nabla_\epsilon g_m^\top, \nabla_\epsilon h_1^\top, \ldots, \nabla_\epsilon h_p^\top]^\top$$

is the Jacobian of the KKT with respect to ϵ.

Since M is nonsingular, as noted, we obtain the result

$$\nabla_\epsilon y(\epsilon) = M(\epsilon)^{-1} N(\epsilon), \tag{1}$$

where we have introduced $N = -\widetilde{N}$, for convenience. Thus, (1) provides a formula for the parameter derivatives of all components of (x, u, w) for ϵ near $\overline{\epsilon}$. In particular, once we calculate the local solution $\overline{x}$ with associated Lagrange multipliers $(\overline{x})$, then we can compute

$$\nabla_\epsilon y(\overline{\epsilon}) = M(\overline{\epsilon})^{-1} N(\overline{\epsilon}), \tag{2}$$

where $y(\overline{\epsilon}) = [x(\overline{\epsilon}), u(\overline{\epsilon}), w(\overline{\epsilon})] = (\overline{x}, \overline{u}, \overline{w}) = \overline{y}$ and of course $\nabla_\epsilon y = [\nabla_\epsilon x^\intercal, \nabla_\epsilon u^\intercal, \nabla_\epsilon w^\intercal]^\intercal$. It also follows that $\nabla_\epsilon y$ is continuous near $\overline{\epsilon}$, so $\nabla_\epsilon y(\epsilon) \to \nabla_\epsilon y(\overline{\epsilon})$ as $\epsilon \to \overline{\epsilon}$.

We can also calculate parameter derivatives of first and second order of the local optimal value function $f^*(\epsilon) = f[x(\epsilon), \epsilon]$, again by repeated use of the chain rule. The results follow. Since the KKT imply that

a) $f^*(\epsilon) = L[x(\epsilon), u(\epsilon), w(\epsilon), \epsilon]$, then, since $\nabla_\epsilon f^* = \nabla_y L \nabla_\epsilon y + \nabla_\epsilon L$ and since it can be shown that $\nabla_y L \nabla_\epsilon y = 0$, it follows that
b) $\nabla_\epsilon f^*(\epsilon) = \nabla_\epsilon L(x, u, w, \epsilon)|_{(x, u, w) = [x(\epsilon), u(\epsilon), w(\epsilon)]}$, and the derivative of this gives $\nabla^2_\epsilon f^*(\epsilon)$, i. e.,
c) $\nabla^2_\epsilon f^*(\epsilon) = d/d\epsilon[\nabla_\epsilon L[x(\epsilon), u(\epsilon), w(\epsilon), \epsilon]]$, where the vertical bar in b) denotes evaluation at the specified point.

As before, these expressions apply for all ϵ near $\overline{\epsilon}$, they can be evaluated at $\overline{\epsilon}$ once $(\overline{x}, \overline{u}, \overline{w})$ is available, and they are continuous near $\overline{\epsilon}$.

We derive a formula for $\nabla^2_\epsilon f^*$ in explicit form in the next section, for the general problem P(ϵ).

Some special realizations of problem P(ϵ) should be mentioned. They are intrinsically important and also result in considerably simplified formulas.

If problem P(ϵ) is unconstrained, then the result (as stated in ► Sensitivity and stability in NLP) is that $\nabla_\epsilon f^* = \nabla_\epsilon f[x(\epsilon), \epsilon]$, with $\nabla_\epsilon x = -\nabla^2_x f^{-1} \nabla^2_{\epsilon x} f$ and $\nabla^2_\epsilon f^* = \nabla^2_{x\epsilon} f \nabla_\epsilon x + \nabla^2_\epsilon f$, which can be conveniently expressed as

$$\nabla^2_\epsilon f^* = [\nabla_\epsilon x^\top, I] \nabla^2_{(x,\epsilon)} f [\nabla_\epsilon x^\top, I]^\top,$$

where

$$\nabla^2_{(x,\epsilon)} f = \begin{pmatrix} \nabla^2_x f & \nabla^2_{\epsilon x} f \\ \nabla^2_{x\epsilon} f & \nabla^2_\epsilon f \end{pmatrix}.$$

If the constraints of problem P(ϵ) are present but independent of ϵ, then again $\nabla_\epsilon f^* = \nabla_\epsilon f[x(\epsilon), \epsilon]$ and $\nabla^2_\epsilon f^* = \nabla_x \epsilon^2 f \nabla_\epsilon x + \nabla^2_\epsilon f$, but now $\nabla_\epsilon x$ is obtained from (1) with $N = [-\nabla^2_{\epsilon x} f^\intercal, 0]^\intercal$.

An extremely important case is the so-called *right-hand side problem*, an instance of problem P(ϵ) with form

$$\mathrm{P}_1(\epsilon) \quad \begin{cases} \min\limits_x & f(x) \\ \text{s.t.} & g_i(x) \geq \epsilon_i \quad (i = 1, \ldots, m), \\ & h_j(x) = \epsilon_j + m \quad (j = 1, \ldots, p). \end{cases}$$

This was mentioned also in ► Sensitivity and stability in NLP: Continuity and differential stability, in the context of the directional derivative $D_z f^*$ of f^* which we found there to be $D_z f^*(\epsilon) = [u(\epsilon), -w(\epsilon)]^\intercal z$. Consistent with this result, we find for problem $\mathrm{P}_1(\epsilon)$, under our current assumptions, that

$$\begin{aligned} \nabla_\epsilon f^*(\epsilon) &= [u(\epsilon), -w(\epsilon)]^\top; \\ \nabla^2_\epsilon f^*(\epsilon) &= [\nabla_\epsilon u(\epsilon)^\top, -\nabla_\epsilon w(\epsilon)^\top]^\top. \end{aligned} \tag{3}$$

Thus, as noted in ► Sensitivity and stability in NLP: Continuity and differential stability, the rate of change of the optimal value with respect to a small change only in the kth constraint value is captured by the value of the associated optimal Lagrange multiplier. In applications, since the components of ϵ may generally be viewed as 'resource levels', this establishes a direct connection between a given level and its imputed value (with regard to a marginal change in the optimal value), given by the associated optimal Lagrange multiplier. Thus, as noted before in ► Sensitivity and stability in NLP: Continuity and differential stability, optimal Lagrange multipliers are often called 'shadow prices'.

The Lagrange multipliers are explicitly involved as variables in dual programs, and constitute all the dual variables for linear programming problems. A consid-

erable literature is devoted to duality and dual programs and their relationship. As noted, duality and sensitivity analysis results are closely connected. We do not pursue this here. See [2] or any standard NLP-LP textbook for an introduction.

Applications of Sensitivity Analysis

There are numerous applications of sensitivity and stability results such as those briefly presented. One of the most immediate is verification of local stability and the determination of the relative importance of each parameter at a local solution, in accordance with its relative effect on the optimal value, solution point or Lagrange multipliers. This assessment is also relevant to model validation in determining if the relative effects noted are reasonable, whether changes are warranted, whether some parameters should be rescaled or calculated with more precision or others assumed constant and locally ignored for simplicity because of their little effect, etc.

Another immediate application is the estimation of a solution or optimal value for perturbed data. This was indicated in ► Sensitivity and stability in NLP for the unconstrained problem. For the general problem P(ϵ), under the assumptions of Proposition 1, an approximation of $y(\epsilon)$ in a neighborhood of $\overline{\epsilon}$ is given by the first order terms of a Taylor's series

$$y(\epsilon) \approx y(\overline{\epsilon}) + \nabla_\epsilon y(\overline{\epsilon})(\epsilon - \overline{\epsilon}).$$

Using the results of the previous section, particularly (2), this translates into

$$(x(\epsilon), u(\epsilon), w(\epsilon)) \approx (\overline{x}, \overline{u}, \overline{w}) + M(\overline{\epsilon})^{-1} N(\overline{\epsilon})(\epsilon - \overline{\epsilon}). \quad (4)$$

Similarly, we can obtain first- or second order estimates of the optimal value function, respectively as follows, near $\epsilon = \overline{\epsilon}$:

$$f^*(\epsilon) \approx f^*(\overline{\epsilon}) + \nabla_\epsilon f^*(\overline{\epsilon})(\epsilon - \overline{\epsilon}),$$

and

$$f^*(\epsilon) \approx f^*(\overline{\epsilon}) + \nabla_\epsilon f^*(\overline{\epsilon})(\epsilon - \overline{\epsilon}) + \frac{1}{2}(\epsilon - \overline{\epsilon})^\top \nabla^2_\epsilon f^*(\overline{\epsilon})(\epsilon - \overline{\epsilon}). \quad (5)$$

These approximations provide computable estimates when a close approximation of $((\overline{x}, \overline{u}, \overline{w}))$ is available, where we can use the relationships indicated in the previous section for $\nabla_\epsilon f^*$, $\nabla^2_\epsilon f^*$, etc., under the various cases that can arise.

Numerous other applications are known and documented and the reader can undoubtedly think of several. We cannot pursue more developments here, but we do indicate some computational efficiencies and algorithmic approximation possibilities in the next section.

Computational Efficiencies and AlgorithmicApproximations Computational Efficiencies and Algorithmic Approximations

First, we show how the results of Proposition 1 can be exploited to greatly simplify the calculation of the parameter-derivatives, and also yield some new formulas. Then, we show how the solution point and optimal value parameter derivatives can be approximated by a solution algorithm as a solution is approached.

Computational Efficiencies

Above, we noted that the equations KKT[$y(\epsilon), \epsilon$], i. e.

$$\begin{cases} \nabla_x L[x(\epsilon), u(\epsilon), w(\epsilon), \epsilon] = 0, & \\ u_i(\epsilon) g_i[x(\epsilon), \epsilon] = 0 & (i = 1, \ldots, m), \\ h_j[x(\epsilon), \epsilon] = 0 & (j = 1, \ldots, p), \end{cases}$$

hold in a neighborhood of $\overline{\epsilon}$. Under the assumptions, all is changing continuously with ϵ, locally, and indeed the system is once continuously differentiable. Having calculated $(\overline{x}, \overline{u}, \overline{w})$ with $\epsilon = \overline{\epsilon}$, we have valuable additional information. We know which g_i and u_i are 0 or positive at $(\overline{y}, \overline{\epsilon})$, and we know this relative status of g_i and u_i will sustain near $\epsilon = \overline{\epsilon}$ at $y(\epsilon)$. Thus, we can delete the nonbinding-constraint equations and we can divide out the positive u_i, without losing any information, locally. We proceed to do this and, for convenience, relabel the binding constraints $\widehat{g} = (g_1, \ldots, g_r)^\top$, and associated (all positive) Lagrange multipliers $\widehat{u} = (u_1, \ldots, u_r)^\top$, so that $g_i > 0$ and $u_i = 0$ for $i = r + 1, \ldots, m$ can be ignored and suppressed in the subsequent calculation. We redefine the Lagrangian accordingly to $\widehat{L} = f - \sum_{i=1}^r u_i g_i + \sum_{j=1}^p w_j h_j$ and relabel $\nabla_\epsilon y$ as $\nabla_\epsilon y$, to indicate the reduced derivative, so that $\widehat{y} = (x, \widehat{u}, w)$.

Given all these simplifications, the KKT system becomes KKT$[\widehat{y}(\epsilon), \epsilon]$:

$$\begin{cases} \nabla_x \widehat{L}[x(\epsilon), \widehat{u}(\epsilon), w(\epsilon), \epsilon] = 0, \\ -g_i[x(\epsilon), \epsilon] = 0 & (i = 1, \ldots, r), \\ h_j[x(\epsilon), \epsilon] = 0 & (j = 1, \ldots, p), \end{cases}$$

where we have written $-g_i$ rather than $+g_i$ to obtain a symmetric (x, u, w)-Jacobian of this system and further computational advantage. Now, differentiating by ϵ yields

$$\widehat{M}(\epsilon)\nabla_\epsilon \widehat{y}(\epsilon) - \widehat{N}(\epsilon) = 0$$

and analogously as before in (1), we obtain

$$\nabla_\epsilon \widehat{y}(\epsilon) = \widehat{M}(\epsilon)^{-1}\widehat{N}(\epsilon), \tag{6}$$

where the Jacobian of KKT with respect to $(x, \widehat{u}, w)$ is

$$\widehat{M} = \begin{pmatrix} \nabla_x^2 \widehat{L} & P^\top \\ P & 0 \end{pmatrix}$$

and the negative of the Jacobian with respect to ϵ is

$$\widehat{N} = [-\nabla^2_{\epsilon x}\widehat{L}^\top, \nabla_\epsilon g_1^\top, \ldots, \nabla_\epsilon g_r^\top, \\ -\nabla_\epsilon h_1^\top, \ldots, -\nabla_\epsilon h_p^\top]^\top$$

and

$$P^\top = [-\nabla_x \widehat{g}^\top, \nabla_x h^\top].$$

Remark 3 We note that the above system KKT$[\widehat{y}(\epsilon), \epsilon]$ turns out to be $\nabla_{\underset{y}{\sim}}\widehat{L} = 0$ and we get $\widehat{M} = \nabla^2_{\underset{y}{\sim}}\widehat{L}$, all evaluated at $[\widehat{y}(\epsilon), \epsilon]$, where we recall that $\widehat{y} = (x, \widehat{u}, w)$. Also, note that $\nabla^2_{\underset{\epsilon y}{\sim}}\widehat{L} = -\widehat{N} = \nabla^2_{\underset{y\epsilon}{\sim}}\widehat{L}^\top$.

Of course, all of the formulas presented in the first Section still hold, in terms of this new reduced structure, to provide all the relevant and nontrivial formulas for all the cases mentioned there. In addition, we are able to express all of the block components A_{11}, A_{12}, A_{21}, A_{22} of $\widehat{M}^{-1}$, for all cases that can arise, in terms of the original problem derivatives, resulting in enormous computational savings. See [2, Chap. 4, Sect. 2]. Further economies are introducedin the special formulas that apply for the right-hand side perturbation problem $P_1(\epsilon)$ and other particular instances of the general problem $P(\epsilon)$. The optimal value formulas also simplify considerably and, for the general problem since $f^* = \widehat{L}$ and $\nabla_\epsilon f^* = \nabla_\epsilon \widehat{L}$, we obtain the elegant result

$$\begin{aligned} \nabla^2_\epsilon f^* &= \nabla^2_{\underset{y\epsilon}{\sim}}\widehat{L}\nabla_\epsilon \widehat{y} + \nabla^2_\epsilon \widehat{L} \\ &= -\widehat{N}^\top \nabla_\epsilon \widehat{y} + \nabla^2_\epsilon \widehat{L} = -\widehat{N}^\top \widehat{M}^{-1}\widehat{N} + \nabla^2_\epsilon \widehat{L} \end{aligned} \tag{7}$$

which can be expressed as

$$\begin{aligned} \nabla^2_\epsilon f^* &= [\nabla_\epsilon \widehat{y}^\top, I]\nabla^2_{\underset{(y,\epsilon)}{\sim}}\widehat{L}\begin{pmatrix} \nabla_\epsilon \widehat{y} \\ I \end{pmatrix} \\ &= \begin{pmatrix} \nabla_\epsilon \widehat{y}^\top, & I \end{pmatrix}\begin{pmatrix} \nabla^2_{\underset{y}{\sim}}\widehat{L} & \nabla^2_{\underset{\epsilon y}{\sim}}\widehat{L} \\ \nabla^2_{\underset{y\epsilon}{\sim}}\widehat{L} & \nabla^2_\epsilon \widehat{L} \end{pmatrix}\begin{pmatrix} \nabla_\epsilon \widehat{y} \\ I \end{pmatrix} \\ &= \begin{pmatrix} \nabla_\epsilon \widehat{y}^\top & I \end{pmatrix}\begin{pmatrix} \widehat{M} & -\widehat{N} \\ -\widehat{N}^\top & \nabla^2_\epsilon \widehat{L} \end{pmatrix}\begin{pmatrix} \nabla_\epsilon \widehat{y} \\ I \end{pmatrix}. \end{aligned}$$

This expression and the first equation in (7) are new and were not given in [2]. It turns out that $\nabla^2_\epsilon f^*$ can be reduced to the simpler form (presented in [2]),

$$\begin{aligned} \nabla^2_\epsilon f^* &= \begin{pmatrix} \nabla_\epsilon x^\top & I \end{pmatrix}\nabla^2_{(x,\epsilon)}\widehat{L}\begin{pmatrix} \nabla_\epsilon x \\ I \end{pmatrix} \\ &= \begin{pmatrix} \nabla_\epsilon x^\top & I \end{pmatrix}\begin{pmatrix} \nabla^2_x \widehat{L} & \nabla^2_{\epsilon x}\widehat{L} \\ \nabla^2_{x\epsilon}\widehat{L} & \nabla^2_\epsilon \widehat{L} \end{pmatrix}\begin{pmatrix} \nabla_\epsilon x \\ I \end{pmatrix}. \end{aligned} \tag{8}$$

Remark 4 If the given problem $P(\epsilon)$ is jointly convex in (x, ϵ), i. e., f convex, all g_i concave, all h_j linear affine in (x, ϵ) with T convex, then it is well known that f^* is convex. This fact becomes immediately evident from (8), since $P(\epsilon)$ jointly convex implies that $\widehat{L}$ is jointly convex. But this implies that $\nabla^2_{(x,\epsilon)}\widehat{L}$ is positive semidefinite, which means that $\nabla^2_\epsilon f^*$ is positive semidefinite, hence f^* must be convex. Of course, we need to assume that problem $P(\epsilon)$ has a solution for each $\epsilon \in T$ where the conclusion of Proposition 1iii) holds, to guarantee that (8) holds for each $\epsilon \in T$.

Algorithmic Approximations

An intriguing possibility is the simultaneous approximation of sensitivity and stability analysis information, as a solution is being estimated by an algorithm in progress. We illustrate the feasibility of this idea with results that have been obtained by a classical combination barrier-exterior point algorithm, forerunner of a powerful class of interior-exterior point methods that are currently among the best methods available, both for LP and NLP.

The classical *logarithmic-quadratic barrier-penalty function* for problem P(ϵ) is defined as

$$W(x,\epsilon,r) = f(x,\epsilon) - r\sum_{i=1}^{m}\log g_i(x,\epsilon) + \frac{1}{2r}\sum_{j=1}^{p} h_j^2(x,\epsilon)\,. \quad (9)$$

A solution procedure based on W for solving problem P(ϵ) for any fixed value of ϵ is briefly described as follows: Select $r_k > 0$ such that $r_k \to 0$ as $k \to +\infty$ and solve $\min_x W(x,\epsilon,r_k)$ s.t. $g_i(x,\epsilon) > 0$ (for all i) to obtain $x(\epsilon, r_k)$, for $k = 1, 2, \ldots$. Then, ideally, $x(\epsilon, r_k) \to x(\epsilon)$, a solution of P($\epsilon$), as $k \to +\infty$.

At a minimizer $x(\epsilon, r)$ of $W(x, \epsilon, r)$, we must have $\nabla_x W = 0$. Since $\nabla_x W = \nabla_x f - \sum_{i=1}^m (r/g_i)\nabla g_i + \sum_{j=1}^p (h_j/r)\nabla h_j$ and since the Lagrangianof problem P(ϵ) is $L = f - \sum_{i=1}^m u_i g_i + \sum_{j=1}^p w_j h_j$, yielding $\nabla_x L = \nabla_x f - \sum_{i=1}^m u_i \nabla_x g_i + \sum_{j=1}^p w_j \nabla h_j$, then $\nabla_x W = 0$ is equivalent to $\nabla_x L = 0$ with $r/g_i = u_i$ and $h_j/r = w_j$. Hence, combined with the results ofProposition 1, we have the following additional consequences.

Proposition 5 *Under the assumptions of Proposition 1, for any (ϵ, r) in a neighborhood of $((\overline{\epsilon}, 0), 0)$ with $r > 0$, there exists a unique once continuously differentiable vector function y such that $y(\epsilon, r) = [x(\epsilon, r), u(\epsilon, r), w(\epsilon, r)]$, where the assumptions of Proposition 1 continue to hold, and such that conditions KKT$[y(\epsilon, r), \epsilon, r]$:*

$$\begin{cases} \nabla_x L[x(\epsilon,r), u(\epsilon,r), w(\epsilon,r), \epsilon] = 0, \\ u_i(\epsilon,r) g_i[x(\epsilon,r),\epsilon] = r \quad (i = 1,\ldots,m), \\ h_j[x(\epsilon,r),\epsilon] = r w_j \quad (j = 1,\ldots,p), \end{cases}$$

with $y(\overline{\epsilon}, 0) = (\overline{x}, \overline{u}, \overline{w})$, and suchthat $x(\epsilon, r)$ is a locally unique unconstrained local minimizer of $W(x, \epsilon, r)$, with all $g_i[x(\epsilon, r), \epsilon] > 0$ and $\nabla_x^2 W[x(\epsilon, r), \epsilon, r]$ positive definite.

Note that the resulting equations in Proposition 5, satisfied at $y(\epsilon, r)$, are a simple perturbationof the KKT equations exhibited above, and we have labeled them accordingly. This is a striking example of the close relationship between optimality conditions and algorithmic theory, and as we now indicate, sensitivity analysis as well.

Just as above, we can immediately differentiate equations KKT$[y(\epsilon, r), \epsilon, r]$ withrespect to ϵ, now to obtain the perturbed formula for the parameter derivative $\nabla_\epsilon y(\epsilon)$ given in (1). The analogous reasoning applies and we obtain

$$\nabla_\epsilon y(\epsilon, r) = M(\epsilon, r)^{-1} N(\epsilon, r), \quad (10)$$

where M and $-N$ are the Jacobians of the perturbed KKT system with respect to (x, u, w) and ϵ, respectively.

The following results hold, for any ϵ or $\widehat{\epsilon}$ sufficiently close to $\overline{\epsilon}$ and for $r > 0$ and small enough:

i) $y(\widehat{\epsilon}, 0) = y(\widehat{\epsilon})$,
ii) $\nabla_\epsilon y(\widehat{\epsilon}, 0) = \nabla_\epsilon y(\widehat{\epsilon})$,
iii) $y(\epsilon, r) \to y(\widehat{\epsilon})$ as $(\epsilon, r) \to (\widehat{\epsilon}, 0)$, and
iv) $\nabla_\epsilon y(\epsilon, r) \to \nabla_\epsilon y(\widehat{\epsilon}) = M(\widehat{\epsilon})^{-1} N(\widehat{\epsilon})$ as $(\epsilon, r) \to (\widehat{\epsilon}, 0)$.

Thus, in particular, we can take $\widehat{\epsilon} = \overline{\epsilon}$ and estimate $y(\overline{\epsilon})$ and $\nabla_\epsilon y(\overline{\epsilon})$ by $y(\epsilon, r)$ and $\nabla_\epsilon y(\epsilon, r)$, for (ϵ, r) near $(\overline{\epsilon}, 0)$.

These results lead immediately to optimal value approximations since we find that, for ϵ near $\overline{\epsilon}$ and $r > 0$ small:

i) $W^*(\epsilon, r) \to f^*(\epsilon)$,
ii) $\nabla_\epsilon W^*(\epsilon, r) \to \nabla_\epsilon f^*(\epsilon)$, and
iii) $\nabla_\epsilon^2 W^*(\epsilon, r) \to \nabla_\epsilon^2 f^*(\epsilon)$ as $r \to 0$, where $W^*(\epsilon, r) = W[x(\epsilon, r), \epsilon, r]$.

We also note that since $\nabla_x W = 0$ and $\nabla_x^2 W$ is positive definite near $\nabla_x W = 0$, we have that $\nabla_x^2 W \nabla_\epsilon x + \nabla_{\epsilon x}^2 W = 0$, yielding

$$\nabla_\epsilon x(\epsilon, r) = -\nabla_x^2 W^{-1} \nabla_{\epsilon x}^2 W, \quad (11)$$

an alternative calculation for $\nabla_\epsilon x$, leading immediately to simple formulas for $\nabla_\epsilon u_i(\epsilon, r)$ and $\nabla_\epsilon w_j(\epsilon, r)$, through the relations $u_i = r/g_i$ and $w_j = h_j/r$. These results provide a basis for approximating solution sensitivity information at an unconstrained minimizer of W by exploiting algorithmic properties and calculations. It is important to note that most of the information required to calculate the parameter derivatives will already be available from the usual implementations of an algorithm based on W that is used to find a solution of problem P(ϵ).

In the next section, we present a few basic results that provide upper bounds on a solution point change and upper and lower bounds on the change in the optimal value, when the problem data parameter is perturbed. Many extensions and variations are possible and many have been reported. Our objective here is to give an idea of the kind of bound information that can be calculated, once a solution has been determined.

Bounds

Linear Equations

In ► Sensitivity and stability in NLP, our first example of an important and widely encountered mathematical model is a system of equations,

$$Ax = b, \tag{12}$$

where A is an $n \times n$ real matrix and x and b are in E^n. We briefly discussed a number of issues that might arise in characterizing a solution x for changes in the data, A and b. As noted, we may think of a solution or set of solutions as a function of A and b, viewed as parameters, exploit the nonsingularity of A to actually obtain a closed form expression for $x(A; b)$ and otherwise endeavor to track x as A and b vary, or at least establish existence and continuity and parameter-differentiability properties and the like. Another approach is the development of bounds on a solution change resulting from the data perturbations. Since the equations model is so important, and since many of the sensitivity results that we have presented here and in ► Sensitivity and stability in NLP and ► Sensitivity and stability in NLP: Continuity and differential stability and that have been presented elsewhere reduce the problem to equations (or their linearization) that follow from optimality conditions, the model is also quite relevant. Hence, we give a classical result that is both important and instructive in revealing what information is crucial in the calculation of bounds.

Let $\|\cdot\|$ denote a vector norm or its corresponding matrix norm, as relevant in the following context. Perturbations in the data A and b are denoted by δA and δb, respectively.

Proposition 6 *Suppose A^{-1} exists and x solves $Ax = b$ while $x + \delta x$ solves $(A+ \delta A)(x +\delta x) = b+ \delta b$. Suppose $\alpha = \| A^{-1}\delta A \| < 1$. Then, the corresponding change δx in the solution x satisfies the following inequality*

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{c(A)}{1-\alpha}\left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|}\right), \tag{13}$$

where $c(A) = \|A\| \|A^{-1}\|$ is the so-called condition number of A.

This result provides a bound for a generally small perturbation of A (since we require $\alpha < 1$) on the relative change in x for relative changes in the data. For $\| \delta A \|$ small enough, α will be close to 0, and the crucial factor is the condition number: if $c(A)$ is small enough, then small relative solution changes result from small relative data changes, but if $c(A)$ is too large, then large solution changes may result from small data changes. The former and latter cases correspond to what is often termed to be 'well-conditioned' or 'ill-conditioned', respectively. (Generally the larger the value of $c(A)$, the more difficult it is to solve (12) with prescribed accuracy.) This bound is of intrinsic interest and should provide a useful perspective for the reader.Some of the bounds results that follow for various classes of parametric programs may be seen to be related to some variation of (13). In this context, note that the condition number $c(A) = \lambda_{\max}/\lambda_{\min} \geq 1$, where $\lambda_{\max}$ and $\lambda_{\min}$ are the positive square root of the maximum and minimum eigenvalues of $A^{\mathsf{T}} A$, respectively, and we assume that $\lambda_{\min} > 0$. If A is a real positive definite symmetric matrix, then $\lambda_{\min}$ and $\lambda_{\max}$ are simply the minimum and maximum eigenvalues of A.

Solution-Point Bounds for NLP

We begin with a bound on the perturbed solution of a *quadratic program* (QP), i. e., a mathematical programming problem having a quadratic objective function and linear constraints:

$$\mathrm{Q}(K, k) \quad \begin{cases} \min\limits_{x \in E^n} & \frac{1}{2} x^{\top} K x - k^{\top} x \\ \text{s.t.} & Cx \leq c, \quad Dx = d. \end{cases}$$

Proposition 7 *Assume that K is a real positive definite and symmetric matrix, with minimum eigenvalue $\overline{\lambda}$. Suppose the data K and k are perturbed to $\widehat{K}$ and $\widehat{K}$ and let $\delta = \max\{\left\|\widehat{K} - K\right\|, \left\|\widehat{k} - k\right\|\}$. Assume that $\overline{x}$ solves $Q(K, k)$ and $\widehat{x}$ solves $Q(\widehat{K}, \widehat{k})$. Then, provided $\delta < \overline{\lambda}$, it follows that*

$$\left\|\widehat{x} - \overline{x}\right\| \leq \frac{\delta}{\overline{\lambda} - \delta}(1 + \|\overline{x}\|). \tag{14}$$

The following results apply to the general parametric problem $\mathrm{P}(\epsilon)$ introduced above, under various assumptions and conditions already used here and in ► Sensitivity and stability in NLP: Continuity and differential stability for continuity and derivativecharacterizations. See ► Sensitivity and stability in NLP: Continuity and differential stability for the involved definitions.

The next proposition invokes all the same assumptions as Proposition 1, except that we now assume that the functions of problem P(ϵ) have twice differentiable partial derivatives in x that are jointly continuous in (x, ϵ), rather than assuming twice continuous partial derivatives jointly in (x, ϵ) as was done for Proposition 1. With this understanding, we have the following result.

Proposition 8 *If the conditions* KKT$(\overline{x}, \overline{u}, \overline{w}, \epsilon)$, SOSC$(\overline{x}, \overline{u}, \overline{w}, \epsilon)$ *and* LI$(\overline{x}, \epsilon)$ *and* SCS$(\overline{x}, \epsilon)$ *hold for* $\overline{x} \in R(\epsilon)$ *at* $\epsilon = \overline{\epsilon} \in$ interior T, *with the differentiability assumptions weakened as explained in the preceding paragraph, then there exists a locally unique continuous vector function* $y = (x, u, w)$ *such that* $y(\overline{\epsilon}) = [x(\overline{\epsilon}), u(\overline{\epsilon}), w(\overline{\epsilon})] = (\overline{x}, \overline{u}, \overline{w}) = \overline{y}$ *and such that these assumptions continue to hold at* $y(\epsilon) = [x(\epsilon), u(\epsilon), w(\epsilon)]$ *in a neighborhood of* $\overline{\epsilon}$. *This implies that* $x(\epsilon)$ *is an isolated local minimizer with associated unique Lagrange multipliers* $[u(\epsilon), w(\epsilon)]$. *Furthermore, for any* $\lambda \in (0, 1)$, *near* $[u(\epsilon), w(\epsilon)]$, *the following bound holds:*

$$\|y - y(\epsilon)\| \leq (1 - \lambda)^{-1} \left\| M(\overline{\epsilon})^{-1} \right\| \|F(y, \epsilon)\|, \quad (15)$$

where $F = [\nabla_x L, u_1 g_1, \ldots, u_m g_m, h_1, \ldots, h_p]^\intercal$ *so that* $F(\overline{y}, \overline{\epsilon}) = 0$ *represents the Karush–Kuhn–Tucker conditions and* $M = \nabla_y F$, *all as introduced in above.*

The real-valued function ϕ is said to be *Lipschitz continuous* on a set S contained in a normed space X if there exists a quantity $\lambda > 0$ such that $|\phi(x) - \phi(y)| \leq \lambda \| x - y$ for all $x, y \in S$.

In ► Sensitivity and stability in NLP: Continuity and differential stability we gave the following result, summarized here for convenience.

Proposition 9 *If the conditions* KKT$(\overline{x}, \overline{u}, \overline{w}, \epsilon)$, SSOSC$(\overline{x}, \overline{u}, \overline{w}, \epsilon)$ *and* LI$(\overline{x}, \epsilon)$ *hold for a feasible* $\overline{x}$ *at* $\epsilon = \overline{\epsilon} \in$ interior T, *then near* $\overline{\epsilon}$ *there exists* $y = (x, u, w)$ *continuous and locally unique such that* $y(\overline{\epsilon}) = \overline{y}$ *and such that the assumptions persist at* $y(\epsilon)$; $x(\epsilon)$ *is an isolated local minimizer with associated unique Lagrange multipliers*$[u(\epsilon), w(\epsilon)]$; x, u *and* w *are locally Lipschitz and directionally differentiable; and* f^* *is once continuously differentiable and twice directionally differentiable. Therefore, in particular it follows that for* ϵ *near* $\overline{\epsilon}$, *there exists* $\alpha, \beta, \gamma > 0$ *such that*

a) $\|x(\epsilon) - \overline{x}\| \leq \alpha \|\epsilon - \overline{\epsilon}\|$;
b) $\|u(\epsilon) - \overline{u}\| \leq \beta \|\epsilon - \overline{\epsilon}\|$; *and*
c) $\|w(\epsilon) - \overline{w}\| \leq \gamma \|\epsilon - \overline{\epsilon}\|$.

Thus, we have bounds on the local solution point and associated Lagrange multipliers that strongly regulate the local rate of change of these quantities, assuring that it is uniform and stable in the sense indicated and providing a sharper result than merely knowing continuity. However, the result is theoretical per se and does not give a prescription for calculating the Lipschitz constants, α, β, and γ. Such issues are important and ongoing in obtaining *computable solution* bounds.

Many such Lipschitz continuity results have been obtained in this area, both for a local solution point and its associated Lagrange multipliers, as well as for the local and global optimal value function. To cite still another, under assumptions that we have encountered in ► Sensitivity and stability in NLP: Continuity and differential stability, Proposition 10, we have the following (in addition to the several conclusions given in ► Sensitivity and stability in NLP: Continuity and differential stability): If KKT, GSSOSC and MFCQ hold at $(\overline{x}, u, w)$ for $\epsilon = \overline{\epsilon}$, then the local solution x is again locally unique and Lipschitz, although here the Lagrange multipliers associated with x are not unique in general.

Next, we present a simple general scheme for obtaining computable optimal value bounds for finite parameter perturbations, once a solution point is available. This will be briefly presented for the classes of problems of the form P(ϵ) where the optimal value function f^* is convex on a convex parameter set T. Since a concave function is the negative of a convex function, the results also apply to f^* concave.The approach has even been extended to special classes of problems where f^* is neither convex nor concave. To avoid any pathological exceptions, we assume solution attainment and constraint regularity sufficient to imply the Karush–Kuhn–Tucker conditions at any subject local solution point, in the remainder of the article.

Computable Optimal Value Bounds f^* Convex

We shall first assume that f^* is convex and well-defined for all $\epsilon \in T$, aconvex subset of E^q. Conditions implying the convexity of f^* for problem P(ϵ) are well known. For example, in ► Sensitivity and stability in NLP: Continuity and differential stability we noted the following result:

If P(ϵ) is jointly convex in (x, ϵ), i. e., if f and the $-g_i$ are jointly convex and the h_j are linear-affine in (x, ϵ)

for all i and j and any $\epsilon \in T$ convex, then the optimal value f^* is convex on T. More general assumptions are known that imply the convexity of f^*. Suffice it to say that the class is large of problems $P(\epsilon)$ for which f^* is convex. Many results involving convexity and concavity characterizations of f^* are given in [4] and [7].

We are assuming that f^* is convex on T, a convex nonempty set. Consider any $\epsilon_1, \epsilon_2 \in T$, with $\epsilon_1 \neq \epsilon_2$ and denote by $[\epsilon_1, \epsilon_2]$ the interval from ϵ_1 to ϵ_2. We first show how to obtain bounds on f^* over $[\epsilon_1, \epsilon_2]$. Define $\epsilon(\alpha) = \alpha\epsilon_2 + (1-\alpha)\epsilon_1$, where $0 \leq \alpha \leq 1$. Note that for any $\overline{\epsilon} \in [\epsilon_1, \epsilon_2]$, there exists a unique $\overline{\alpha}$ such that $\overline{\epsilon} = \epsilon(\overline{\alpha})$. Then, we obtain a unique global upper bound $U(\alpha)$ on f^* over $[\epsilon_1, \epsilon_2]$ from the following result that follows from the definition of a convex function:

$$f^*[\epsilon(\alpha)] \leq \alpha f^*(\epsilon_2) + (1-\alpha) f^*(\epsilon_1) \equiv U(\alpha). \quad (16)$$

Next, suppose that $\nabla_\epsilon f^*(\epsilon_1)$ and $\nabla_\epsilon f^*(\epsilon_2)$ exist. Then, again from the definition of convexity it is known that at any $\widehat{\epsilon} \in T$ where $\nabla_\epsilon f^*(\widehat{\epsilon})$ exists we have

$$f^*(\epsilon) \geq f^*(\widehat{\epsilon}) + \nabla_\epsilon f^*(\widehat{\epsilon})(\epsilon - \widehat{\epsilon}), \quad \text{for any } \epsilon \in T. \quad (17)$$

For $\widehat{\epsilon} = \epsilon_1$, or $\widehat{\epsilon} = \epsilon_2$, we thus obtain two global lower bounds $L_1^*(\epsilon)$ and $L_2^*(\epsilon)$. Thus, $L^*(\epsilon) = \max\{L_1^*(\epsilon), L_2^*(\epsilon)\}$ gives a convex lower bound for any $\epsilon \in T$. We can apply this result to $\epsilon = \epsilon(\alpha)$, with $\widehat{\epsilon} = \epsilon_1$ and $\widehat{\epsilon} = \epsilon_2$ and denote the right-hand side of the inequality by $L_1(\alpha)$ and $L_2(\alpha)$, respectively. Next, let $L(\alpha) = \max\{L_1(\alpha), L_2(\alpha)\}$. Then, it follows that

$$\begin{aligned} L(\alpha) &\leq f^*[\epsilon(\alpha)] \\ &\leq U(\alpha) \quad \text{for any } \alpha \in [0,1]. \end{aligned} \quad (18)$$

This last inequality gives global upper and lower bounds onthe optimal value f^* over the interval $[\epsilon_1, \epsilon_2]$. Note that $U(\alpha)$ can be calculated when we have determined only the value of f^* at ϵ_1 and ϵ_2. The lower bound $L(\alpha)$ can be computed when we have also determined $\nabla_\epsilon f^*$ at ϵ_1 and ϵ_2. Thus, with f^* and $\nabla_\epsilon f^*$ well-defined at $\epsilon_1, \epsilon_2 \in T$, these quantities associated with only two problems, $P(\epsilon_1)$ and $P(\epsilon_2)$, respectively, are enough to provide parametric global upper and lower bounds over any interval $[\epsilon_1, \epsilon_2]$.

The bounds over $[\epsilon_1, \epsilon_2]$ can be significantly improved in general each time f^* and $\nabla_\epsilon f^*$ are determined for a new value of ϵ in the interval. For example, suppose we take $\epsilon = \epsilon_3 \in (\epsilon_1, \epsilon_2)$. Once we determine f^* and $\nabla_\epsilon f^*$ at ϵ_3, we can calculate new upper and lower bounds, as above, over $[\epsilon_1, \epsilon_3]$ and $[\epsilon_3, \epsilon_2]$. Because of the convexity of f^*, the new bounds can only improve. In fact, continuing this way with new intermediate parameter values in $[\epsilon_1, \epsilon_2]$, it is theoretically possible to calculate upper and lower bounds to any prescribed degree of accuracy, since the bounds will converge ultimately from below and above to the graph of f^* over $[\epsilon_1, \epsilon_2]$.

Note that to calculate bounds over $[\epsilon_1, \epsilon_2]$, we need the convexity of f^* only over the interval $[\epsilon_1, \epsilon_2]$, not over the entire set T, allowing for significant additional generality, e. g., allowing us to drop the assumption of convexity of T. Furthermore, the process can be extended to provide bounds over more general subsets of T, as follows. Suppose we are interested in $\epsilon_1, \ldots, \epsilon_k \in T$ and suppose we can determine f^* and $\nabla_\epsilon f^*$ at the ϵ_i. Analogously as before, define $\epsilon(\alpha) = \sum_{i=1}^k \alpha_i \epsilon_i$, with $\sum_{i=1}^k \alpha_i = 1$ and $\alpha_i \geq 0$. Then, an upper bound at any such $\epsilon(\alpha)$ is obtained from Jensen's inequality,

$$f^*[\epsilon(\alpha)] \leq \sum_{i=1}^k \alpha_i f^*(\epsilon_i) \equiv U(\alpha), \quad (19)$$

where $\alpha = (\alpha_1, \ldots, \alpha_k)$. Lower bounds $L_i^*(\epsilon)$ on f^* over the entire set T are computable at each ϵ_i using (17) with $\widehat{\epsilon} = \epsilon_i$ $(i = 1, \ldots, k)$, yielding $L^*(\epsilon) = \max\{L_1^*(\epsilon), \ldots, L_k^*(\epsilon)\}$ as a piecewise-linear convex global lower bound on f^* over T. Adapting this to $\epsilon = \epsilon(\alpha)$, we replace ϵ in the lower bound inequality (17)by $\epsilon(\alpha)$ and the $\widehat{\epsilon}$ by ϵ_i to get $L_i(\alpha)$ $(i = 1, \ldots, k)$, finally getting $L(\alpha) = \max\{L_1(\alpha), \ldots, L_k(\alpha)\}$. We thus have

$$L(\alpha) \leq f^*[\epsilon(\alpha)] \leq U(\alpha) \quad (20)$$

for any $\alpha = (\alpha_1, \ldots, \alpha_k)$ such that $\sum_{i=1}^k \alpha_i = 1$ and $\alpha_i \geq 0$ $(i = 1, \ldots, k)$.

Remark 10 We note that, unlike before where $\epsilon(\alpha) = \alpha\epsilon_2 + (1-\alpha)\epsilon_1$ and there is only one α for a given $\epsilon = \epsilon(\alpha) \in [\epsilon_1, \epsilon_2]$, there now may be a set of values of α such that $\epsilon(\alpha) = \epsilon$. Thus, for agiven ϵ such that $\epsilon = \epsilon(\alpha)$, the upper bound $U(\alpha)$ may have a range of values. In this instance, we note that the best bound $U^*(\epsilon)$ is the optimal value of: $\min_\alpha U(\alpha)$ s.t. $\sum_{i=1}^k \alpha_i \epsilon_i = \epsilon$, $\sum_{i=1}^k \alpha_i = 1$, $\alpha_i \geq 0$ $(i = 1, \ldots, k)$, a linear program with parameter ϵ. (The lower bound will have only one

value for any such ϵ, as before.) The upper bound will be single-valued for $\epsilon = \epsilon(\alpha)$, if the set of ϵ such that $\epsilon = \epsilon(\alpha)$ is a *simplex* with vertices ϵ_i in the subset of T that is spanned by the given ϵ_i. Thus, an effective way to generate upper bounds over any subset S of T is to systematically cover S with contiguous simplexes, using points ϵ_i of S. The upper and lower bounds over each simplex will be uniquely determined and loose bounds over a given simplex can be sharpened by selecting a new parameter vector $\overline{\epsilon}$ in the given simplex, subdividing into newly determined contiguous simplexes with the new vertex $\overline{\epsilon}$, and proceeding as indicated above, to obtain upper and lower bounds over each simplex.

Next, we show how to obtain a simpler and sharper upper bound, if the constraint functions g_i and h_j are as prescribed when $P(\epsilon)$ is a jointly convex program (i. e., g_i jointly concave and h_j jointly linear-affine, for all i and j).

g_i Jointly Concave, h_j Jointly Linear Affine

With g_i and h_j as indicated, it turns out that if $\widehat{x}(\epsilon_1)$ is a feasible point of $P(\epsilon_1)$ and $\widehat{x}(\epsilon_2)$ is a feasible point of $P(\epsilon_2)$, then $\widetilde{x}(\alpha) \equiv \alpha\widehat{x}(\epsilon_2) + (1-\alpha)\widehat{x}(\epsilon_1)$ is a feasible point of $P[\epsilon(\alpha)]$ for any $\alpha \in [0, 1]$, where $\epsilon(\alpha) = \alpha\epsilon_2 + (1-\alpha)\epsilon_1$. This gives us a feasible point of $P(\epsilon)$ for any $\epsilon \in [\epsilon_1, \epsilon_2]$. It also means that $f^*[\epsilon(\alpha)] \leq f[\widetilde{x}(\alpha), \epsilon(\alpha)] = \widetilde{f}(\alpha)$, i. e., the optimal value of problem $P[\epsilon(\alpha)]$ is bounded above by the objective function evaluated at the feasible point $\widetilde{x}(\alpha)$. This immediately provides an easily computable upper bound on f^* over $[\epsilon_1, \epsilon_2]$ once $\widehat{x}(\epsilon_1)$ and $\widehat{x}(\epsilon_2)$ are available. There is no requirement here that f be convex. If $\widehat{x}(\epsilon_1)$ solves $P(\epsilon_1)$ and $\widehat{x}(\epsilon_2)$ solves $P(\epsilon_2)$, then this upper bound is exact at ϵ_1 and ϵ_2. (Note that f^* will generally be nonconvex if f is not convex.)

$P(\epsilon)$ Jointly Convex

If problem $P(\epsilon)$ is jointly convex, then we not only have the g_i jointly concave and the h_j jointlylinear affine as before but now also have f jointly convex, hence f^* is convex and the upper bound has additional properties. In this case, it turns out that $\nabla^2_\epsilon f^*$ is also convex over [0, 1]. Furthermore, if $x(\epsilon_1)$ solves $P(\epsilon_1)$ and $x(\epsilon_2)$ solves $P(\epsilon_2)$, and $\overline{x}(\alpha) = \alpha x(\epsilon_2) + (1-\alpha)x(\epsilon_1)$ then it follows that $\overline{f}(\alpha) = f[\overline{x}(\alpha), \epsilon(\alpha)]$ gives aconvex upper bound on f^* over $[\epsilon_1, \epsilon_2]$ that either agrees with or is lower and hence sharper than the linear upper bound $U(\alpha)$ described earlier, i. e.,

$$f^*[\epsilon(\alpha)] \leq \overline{f}(\alpha) \leq U(\alpha) \quad \text{for all } \alpha \in [0, 1]. \tag{21}$$

As noted, $\overline{f}(\alpha)$ is immediately computable once a solution is determined for$P(\epsilon_1)$ and $P(\epsilon_2)$. Following the technique for improving $U(\alpha)$, this bound can be improved by solving $P(\epsilon)$ for $\epsilon_3 \in (\epsilon_1, \epsilon_2)$, then proceeding as above for the sub-intervals $[\epsilon_1, \epsilon_3]$ and $[\epsilon_3, \epsilon_2]$.

$\widetilde{f}$ and $\overline{f}$ for Subsets of T

The calculation of $\widetilde{f}$ and $\overline{f}$ can be extended to bound f^* above over more general subsets of T, as well, proceeding somewhat analogously to the procedure given above for f^* convex that was indicated for $U(\alpha)$. We shall illustrate this for $\overline{f}(\alpha)$, and assume that $P(\epsilon)$ is jointly convex. Given a set of values $\{\epsilon_1, \ldots, \epsilon_k\}$, where $\epsilon_i \in T$, suppose we obtain solutions $x(\epsilon_i)$ for $P(\epsilon_i)$, $i = 1, \ldots, k$. Then, as before we consider $\epsilon(\alpha) = \sum_{i=1}^k \alpha_i\epsilon_i$, with $\sum_{i=1}^k \alpha_i = 1$ and $\alpha_i \geq 0$ for all i. If we denote $\overline{x}(\alpha) = \sum_{i=1}^k \alpha_i x(\epsilon_i)$, then it follows that $\overline{x}(\alpha)$ is a feasible point of $P[\epsilon(\alpha)]$, hence again

$$f^*[\epsilon(\alpha)] \leq f[\overline{x}(\alpha), \epsilon(\alpha)] \equiv \overline{f}(\alpha) \leq U(\alpha), \quad \text{for } \alpha = (\alpha_1, \ldots, \alpha_k), \tag{22}$$

where here $U(\alpha) = \sum_{i=1}^k \alpha_i f^*(\epsilon_i)$ as in (19) and $\overline{f}$ is convex over α and we thus have an upper bound on $f^*(\epsilon)$ for any $\epsilon \in T$ such that $\epsilon = \epsilon(\alpha)$ for α as prescribed. Similar remarks to those given previously apply regarding multiple values of α corresponding to any given $\epsilon = \sum_{i=1}^k \alpha_i\epsilon_i$. Again, proceeding to cover T systematically with contiguous simplexes and calculating bounds over each simplex would eliminate this multiple-value problem and result in unique upper and lower bounds, and would appear to be a very effective and natural way to proceed.

Remark 11 As for g_i jointly concave and h_j jointly linear affine, we candrop the convexity assumption on f and we can also simply assume that we have only a *feasible point* $\widehat{x}(\epsilon_i)$ of $P(\epsilon_i)$ for $i = 1, \ldots, k$ (rather than a solution point $x(\epsilon_i)$). Then, we still obtain an upper bound since we again have $\widetilde{x}(\alpha) \in R[\epsilon(\alpha)]$ and hence we again have $\widetilde{x}(\alpha) \in R[\epsilon(\alpha)]$, where we now have $\widetilde{x}(\alpha) = \sum_{i=1}^k \alpha_i\widehat{x}(\epsilon_i)$.

We turn last to a calculation of a bound on the distance from a feasible point of P(ϵ) to a solution of P(ϵ).

Bounds on the Distance of a Feasible Point to a Solution Point

The first bound is for a given unperturbed problem.

Proposition 12 *Suppose we are given problem P(ϵ), with feasible region R(ϵ) convex in x with ϵ fixed. Assume $x(\epsilon)$ locally solves P(ϵ), assume that the feasible region is not a singleton, and assume that there exists a number $m(\epsilon) > 0$ such that $z^{\intercal}\nabla_x^2 f(x, \epsilon)z \geq m(\epsilon)\ \|z\|^2$ for any $x \in R(\epsilon)$ and any $z \in E^n$ such that z is a feasible direction of $R(\epsilon)$ from $x(\epsilon)$, i. e., $x(\epsilon)+ \beta\ z \in R(\epsilon)$ for any β and some $\overline{\beta}$ such that $0 < \beta \leq \overline{\beta}$. Then, $x(\epsilon)$ is the unique global minimizer and the following bound holds:*

$$\|x - x(\epsilon)\| \leq \left(\frac{f(x,\epsilon) - f^*(\epsilon)}{\frac{m(\epsilon)}{2}}\right)^{\frac{1}{2}} \tag{23}$$

for any $x \in R(\epsilon)$.

Remark 13 The conclusions of this Proposition hold in some neighborhood of $x(\epsilon)$ if we assume that the uniform quadratic underestimation of $z^{\intercal}\ \nabla_x^2 fz$ holds only near $x(\epsilon)$ (rather than for any $x \in R(\epsilon)$). Hence, Propositions 14 and 15 also hold if the respective distances being bounded are sufficiently small.

Using the fact that $\nabla_x f[x(\epsilon), \epsilon]z \geq 0$ must hold, the inequality (23)follows easily from a second order Taylor's series expansion of f around $x(\epsilon)$. This provides a bound on the distance from any given feasible point of problem P(ϵ) to the solution $x(\epsilon)$ of P(ϵ). This bound is of theoretical interest but may not be computable without $x(\epsilon)$ because m, z, and f^* depend on $R(\epsilon)$ and $x(\epsilon)$. As for z, if we let z beany $z \in E^n$, then the largest suitable m will generally be smaller than with z restricted, but z will be free of dependency on $x(\epsilon)$ and the results will apply if $m > 0$. Also, with z unrestricted, we note that the function f is strictly convex in $R(\epsilon)$. Nonetheless, even for z unrestricted, the largest suitable value of m is the optimal value (if positive) of a nonconvex program, $\min_{(x,\,z)} z^{\intercal}\nabla_x^2 f(x, \epsilon)z$ s.t. $\|z\| = 1$, $x \in R(\epsilon)$, which may be prohibitive except for special classes of problems. For example, if f is quadratic in x and $\nabla_x^2 f$ is positive definite, then m may be taken to be the minimum eigenvalue of $\nabla_x^2\ f$. We deal with f^* by using a computable upper bound in the sequel.

Next, we allow ϵ to change and obtain a bound on the distance from a solution $x[\epsilon(\alpha)]$ of P[$\epsilon(\alpha)$] to the computable feasible point $\widetilde{x}(\alpha)$, as defined in Remark 11, using the bound (23).

Proposition 14 *In addition to the assumptions of Proposition 12, suppose that for any $\epsilon = \epsilon(\alpha)$, the problem constraints g_i are jointly concave in (x, ϵ) and the h_j jointly affine, and also suppose that problem P(ϵ) has a feasible point $\widehat{x}(\epsilon)$ for $\epsilon = \epsilon_i \in T$ convex, where i = 1, …, k. Then it follows that $\widetilde{x}(\alpha) \in R[\epsilon(\alpha)]$ and*

$$\|\widetilde{x}(\alpha) - x[\epsilon(\alpha)]\| \leq \left(\frac{f[\widetilde{x}(\alpha), \epsilon(\alpha)] - f^*[\epsilon(\alpha)]}{\frac{m[\epsilon(\alpha)]}{2}}\right)^{\frac{1}{2}} \tag{24}$$

for any $\alpha = (\alpha_1, \ldots, \alpha_k)$ $\sum_{i=1}^k \alpha_i = 1$and $\alpha_i \geq 0$ (i = 1, …, k), where $\widetilde{x}(\alpha) = \sum_{i=1}^k \alpha_i \widehat{x}(\epsilon_i)$, $\epsilon(\alpha) = \sum_{i=1}^k \alpha_i\epsilon_i$, $x[\epsilon(\alpha)]$ solves P[$\epsilon(\alpha)$] and f^* [$\epsilon(\alpha)$] is the optimal value of P[$\epsilon(\alpha)$].

The upper bound (24) involves $f^*[\epsilon(\alpha)]$ which would generally not be available unless $x[\epsilon(\alpha)]$ were known. We want a computable upper bound notrequiring $x[\epsilon(\alpha)]$, and such is available if the optimal value f^* of P(ϵ) is convex over the set of $\epsilon = \epsilon(\alpha)$, using the results for f^* convex above.In particular, recall that f^* is convex if P(ϵ) is jointly convex. We obtain the following result that gives computable bounds, once a suitable value for m has been determined.

Proposition 15 *In addition to the assumptions of Proposition 14, assume that $x(\epsilon_i)$ solves P(ϵ_i)(i = 1, …, k) and $\overline{x}(\alpha) = \sum_{i=1}^k \alpha_i x(\epsilon_i)$, suppose that we also have f jointly convex in the set of $\epsilon = \epsilon(\alpha)$ (thus making P(ϵ) jointly convex and hence f^* convex). Then, since $f^*[\epsilon(\alpha)] \geq L(\alpha)$ from (20), we have from (24) that*

$$\|\overline{x}(\alpha) - x[\epsilon(\alpha)]\| \leq \left(\frac{f[\overline{x}(\alpha), \epsilon(\alpha)] - L(\alpha)}{\frac{m[\epsilon(\alpha)]}{2}}\right)^{\frac{1}{2}} \tag{25}$$

and therefore

$$\|\overline{x}(\alpha) - x[\epsilon(\alpha)]\| \leq \left(\frac{U(\alpha) - L(\alpha)}{\left[\frac{m[\epsilon(\alpha)]}{2}\right]}\right)^{\frac{1}{2}} \tag{26}$$

for any $\alpha = (\alpha_1, \ldots, \alpha_k)$ such that $\sum_{i=1}^k \alpha_i = 1$ and $\alpha_i \geq 0$ (i = 1, …, k), where U(α) and L(α) are given above and we have also used (22).

This provides computable bounds over any $\epsilon = \epsilon(\alpha)$, once the solutions $x(\epsilon_i)$ $(i = 1, \ldots, k)$ have been determined, provided m can be calculated or bounded below. There are results that are applicable to the calculation of such a number for important classes of problems. For a fixed (x, ϵ), the value m can be taken to be theminimum eigenvalue of $\nabla_x^2 f$. For f quadratic, this simplifies, as noted earlier. For this and other important computable cases, see [5].

Our last result gives bounds under the same conditions as Propositions 12, 14 and 15, but using a first order bounding condition on the objective function f, ratherthan the second order condition of Proposition 12. It is a simple consequence of the convexity of f.

Proposition 16 *Assume as in Proposition 15, except that we now let* $\nabla f[x(\epsilon), \epsilon]z \geq m(\epsilon) \parallel z \parallel$ *replace the condition that* $z^\intercal \nabla_x^2 f(x, \epsilon)z \geq m(\epsilon) \|z\|^2$. *Then, it follows that*

$$\|x - x(\epsilon)\| \leq \frac{f(x, \epsilon) - f^*(\epsilon)}{m(\epsilon)} \tag{27}$$

for any $x \in R(\epsilon)$. *Now, adding the other assumptions of Proposition 15, we get the bounds*

$$\begin{aligned} \|\overline{x}(\alpha) - x[\epsilon(\alpha)]\| &\leq \frac{f[\overline{x}(\alpha), \epsilon(\alpha)] - L(\alpha)}{m[\epsilon(\alpha)]} \\ &\leq \frac{U(\alpha) - L(\alpha)}{m[\epsilon(\alpha)]}. \end{aligned} \tag{28}$$

These bounds are mainly of theoretical interest because the verification of our condition involving m requires the solution $x(\epsilon)$. Our condition, $\nabla_x fz \geq m \|z\|$ is a minimal growth condition on the directional derivative of f at a solution $x(\epsilon)$ in any feasible direction. A suitable m would be the infimum at $x(\epsilon)$ of $\nabla_x fz$ for $\parallel z \parallel = 1$ and z a feasible direction from $x(\epsilon)$, providing this is positive. This leads to the more relaxed problem (where $x = x(\epsilon)$): $\min_z \nabla fz$ s.t. $\nabla g_i z \geq 0$ for all i such that $g_i = 0$ and $\nabla h_j z = 0$ for all j and $\parallel z \parallel = 1$. See the Remark below. This can be formulated as a linear program and precisely corresponds to the search direction problem required for well-known *methods of feasible directions*. Thus, we obtain another connectionbetween optimality, algorithmic and sensitivity analysis calculations.

Remark 17 With convexity, the condition $\nabla fz > 0$ at a feasible point, for any feasible direction z, is sufficient for unique global minimization at that point. This condition is frequently satisfied at a solution, e. g., it holds at the minimizer of a nondegenerate linear programming problem. We note further that $\nabla fz \geq 0$ for any such z is a sufficient condition for a global minimum of a convex problem and is a necessary condition at a local minimizer of a general (i. e., not necessarily convex) differentiable programming problem. It should also be mentioned that at $x = x(\epsilon)$, the set of unit vectors z, such that $\nabla g_i z \geq 0$ for all i such that $g_i = 0$ and $\nabla h_j z = 0$ for all j, is the same or larger than the set of unit vectors z that are feasible directions of $R(\epsilon)$ from $x(\epsilon)$. Thus, the largest acceptable number $m = \min_z \nabla fz$ at $x(\epsilon)$ will be the same or smaller for the former set of z then for the latter set. However, the former set has the advantage of having a simple algebraic formulation and computability (as noted, it yields a linear programming problem). Thus, we can strengthen the quadratic growth condition in Propositions 12, 14 and 15, and the linear growth condition in Proposition 16, to the indicated set of z. All of these results will then be valid with respect to the newly resulting bounds. Finally, we should mention another appealing fact. With the set of z so chosen, $\nabla fz > 0$ at $x(\epsilon)$ implies that $x(\epsilon)$ is a strict local minimizer for a general programming problem (i. e., without convexity).

Most of the material in this article and many references can be found in [2]. References [1] and [3] give state-of-the-art tutorials and numerous extensions in a modern setting, along with an extensive bibliography. The bounds result of Proposition 6 for the solution of a linear system of equations can be found in [8, Chap. 6, Sect. 6.4]. Other bounds results may be found in [2] and [5], and many other optimal value convexity and concavity results are given in [4] and [7].

See also

- ▶ Nonlocal Sensitivity Analysis with Automatic Differentiation
- ▶ Parametric Global Optimization: Sensitivity
- ▶ Sensitivity Analysis of Complementarity Problems
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Sensitivity and Stability in NLP
- ▶ Sensitivity and Stability in NLP: Continuity and Differential Stability

References

1. Fiacco AV (ed) (1990) Optimization with data perturbations. Ann Oper Res 27 Baltzer, Basel
2. Fiacco AV (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Acad. Press, New York
3. Fiacco AV (ed) (1998) Mathematical programming with data perturbations. Lecture Notes Pure and Applied Math. M. Dekker, New York
4. Fiacco AV, Kyparisis J (1986) Convexity and concavity properties of the optimal value function in parametric programming. J Optim Th Appl 48:95–126
5. Fiacco AV, Kyparisis J (1988) Computable bounds on parametric solutions of convex problems. Math Program 40:213–221
6. Fiacco AV, McCormick GP (1968) Nonlinear programming, sequential unconstrained minimization techniques. Wiley, New York, Unabridged corrected version: Classics Appl. Math. SIAM, 1990.
7. Kyparisis J, Fiacco AV (1987) Generalized convexity and concavity of the optimal value function in nonlinear programming. Math Program 39:285–304
8. Noble B, Daniel JW (1988) Applied linear algebra. Prentice-Hall, Englewood Cliffs, NJ

Sensitivity and Stability in NLP: Continuity and Differential Stability

A. V. Fiacco
Department Operations Res.,
George Washington University, Washington, DC, USA

MSC2000: 90C31

Article Outline

Keywords
Optimality Conditions and Constraint Qualification
See also
References

Keywords

Sensitivity analysis; Nonlinear programming; Optimality conditions; Constraint qualification

In ► Sensitivity and stability in NLP we introduced the parametric nonlinear programming (NLP) problem, gave several examples, mentioned various important applications of sensitivity and stability results for parameter changes and indicated several tools and concepts that have proved effective. We suggest that the reader peruse these preliminaries before reading this continuation of developments.

For convenience, we again state the problem of interest:

$$\mathrm{P}(\epsilon) \quad \begin{cases} \min & f(x,\epsilon) \\ \text{s.t.} & g_i(x,\epsilon) \geq 0 \quad (i=1,\ldots,m), \\ & h_j(x,\epsilon) = 0 \quad (j=1,\ldots,p), \end{cases}$$

where $x \in E^n$ and the parameter (data) $\epsilon \in T \subseteq E^q$. We present some basic continuity and differentiability results of the parameter-dependent solution point $x(\epsilon)$ and optimal value $f^*(\epsilon)$ of problem $\mathrm{P}(\epsilon)$. We focus on concrete results for specific classes of problems that are frequently encountered. More general results may be found in more detailed technical treatments elsewhere and a few publications will be noted for reference and further study. In particular, we do not include nonsmooth results or the notions of point-to-set maps, semicontinuity of maps and functions, generalized gradients, alternative definitions of continuity or differentiability, functional or set perturbations (e. g., f to $\widetilde{f}$, g to $\widetilde{g}$, h to $\widetilde{h}$, $x \in M$ to $x \in \widetilde{M}$, etc.) and other very important mathematical concepts, models or instruments that lead to significant extensions and generalizations of the basic results we present. Our purpose here is to provide a quick and hopefully unencumbered introduction and simple exposition of significant rudimentary results.

The results we present require some well-known smoothness properties, optimality conditions and constraint regularity assumptions, called *constraint qualifications*. We briefly review these in the next section.

If f and $-g_i$ are convex in x and h_j is linear-affine in x, for all i, j and any $\epsilon \in T$, the problem $\mathrm{P}(\epsilon)$ is said to be *convex* in x. If these function properties hold jointly in (x,ϵ) and T is a convex set, then $\mathrm{P}(\epsilon)$ is said to be *jointly convex*. For simplicity, continuous differentiability and continuity are assumed.

Optimality Conditions and Constraint Qualification

Associated with the problem $\mathrm{P}(\epsilon)$ are the *Lagrangian*

$$L(x,u,w,\epsilon) = f(x,\epsilon) - \sum_{i=1}^{m} u_i g_i(x,\epsilon) + \sum_{j=1}^{p} w_j h_j(x,\epsilon)$$

and the *optimal value function*

$$f^*(\epsilon) = \begin{cases} \inf_{x \in R(\epsilon)} f(x, \epsilon) & \text{if } R(\epsilon) \neq \emptyset, \\ +\infty & \text{if } R(\epsilon) = \emptyset, \end{cases}$$

where the *feasible region*

$$R(\epsilon) = \left\{ x \in E^n : \begin{array}{ll} g_i(x, \epsilon) \geq 0 & (i = 1, \ldots, m) \\ h_j(x, \epsilon) = 0 & (j = 1, \ldots, p) \end{array} \right\}.$$

If a local solution $x(\epsilon)$ is known to exist, then it is understood that the optimal value is the local optimal value $f^*(\epsilon) = f[x(\epsilon), \epsilon]$ for local results involving $x(\epsilon)$. The *solution set* $S(\epsilon)$ of $P(\epsilon)$ is defined as

$$S(\epsilon) = \{x \text{ is a local solution} : \ f(x, \epsilon) = f^*(\epsilon)\}.$$

The following concepts and definitions are needed.

a) The *directional derivative* of f^* at $\overline{\epsilon}$ in direction z is

$$D_z f^*(\overline{\epsilon}) = \lim_{\alpha \to 0^+} \frac{f^*(\overline{\epsilon} + \alpha z) - f^*(\overline{\epsilon})}{\alpha}.$$

In the next conditions, assume that x is feasible and that ϵ is fixed at some value in T. Differentiability is assumed as needed.

b) The *Karush–Kuhn–Tucker conditions* (known to be necessary under appropriate regularity conditions at a local solution x of a differentiable problem) are: there exist $u \geq 0$ and w such that (condition KKT(x, u, w, ϵ))

$$\begin{cases} \nabla_x L(x, u, w, \epsilon) = 0, & \\ u_i g_i(x, \epsilon) = 0 & (i = 1, \ldots, m), \\ h_j(x, \epsilon) = 0 & (j = 1, \ldots, p), \end{cases}$$

where $x \in R(\epsilon)$, $u = (u_1, \ldots, u_m)$ and $w = (w_1, \ldots, w_p)$ are called *Lagrange multipliers*, $u \geq 0$ means $u_i \geq 0$ ($i = 1, \ldots, m$), and ∇_x denotes the n-component gradient (row vector) with respect to x.

The set of Lagrange multipliers (u, w) that satisfy KKT(x, u, w, ϵ) is denoted by $K(x, \epsilon)$.

c) The *second order sufficient condition*, noted by SOSC(x, u, w, ϵ), is as follows: KKT(x, u, w, ϵ) holds for some (u, w) at $x \in R(\epsilon)$ and

$$z^\top \nabla_x^2 L(x, u, w, \epsilon) z > 0$$

for all $z \neq 0$, $z \in Z$, where

$$Z = \left\{ z \in E^n : \begin{array}{c} \nabla_x g_i(x, \epsilon) z \geq 0 \\ \text{for } i \text{ s.t.} \\ g_i(x, \epsilon) = 0; \\ \nabla_x g_i(x, \epsilon) z = 0 \\ \text{for } i \text{ s.t.} \\ g_i(x, \epsilon) = 0, \ u_i > 0; \\ \nabla_x h_j(x, \epsilon) z = 0 \\ (j = 1, \ldots, p) \end{array} \right\}.$$

d) The *general second order sufficient condition*, designated GSOSC(x, u, w, ϵ), requires that SOSC holds for all $(u, w) \in K(x, \epsilon)$.

e) The *strong second order sufficient condition*, denoted by SSOSC(x, u, w, ϵ), is the same as SOSC(x, u, w, ϵ) with one change: the restriction in the set Z, that $\nabla_x g_i(x, \epsilon) z \geq 0$ for i such that $g_i(x, \epsilon) = 0$, is dropped.

f) The *general strong second order sufficient condition*, designated GSSOSC(x, u, w, ϵ), requires that SSOSC holds for all $(u, w) \in K(x, \epsilon)$.

g) *strict complementary slackness* at a point x, noted by SCS(x, ϵ), is said to hold if the associated KKT multipliers are such that $u_i > 0$ for all binding g_i (i. e., all i such that $g_i(x, \epsilon) = 0$). (This term derives from the fact that the condition $u_i g_i = 0$ in the KKT equations is known as *complementary slackness*.)

There are many *constraint qualifications*. We shall give results involving the following three:

i) The first is *linear independence* at a point $x \in R(\epsilon)$ of the binding constraint gradients, designated by LI(x, ϵ); i. e., $\nabla_x g_i(x, \epsilon)$ for i such that $g_i(x, \epsilon) = 0$ and $\nabla_x h_j(x, \epsilon)$ for $j = 1, \ldots, p$ are linearly independent.

ii) The *Mangasarian–Fromovitz constraint qualification* holds at $x \in R(\epsilon)$, noted by MFCQ(x, ϵ), if: a) there exists a vector z such that $\nabla_x g_i(x, \epsilon) z > 0$ for all i such that $g_i(x, \epsilon) = 0$ and $\nabla_x h_j(x, \epsilon) z = 0$ for $j = 1, \ldots, p$; and *b*) the vectors $\nabla_x h_j(x, \epsilon)$ ($j = 1, \ldots, p$) are linearly independent.

iii) The *generalized Slater constraint qualification*, denoted by GS(ϵ), holds for the convex problem $P(\epsilon)$ if there exists a feasible point $\overline{x}$ such that $g_i(\overline{x}, \epsilon) > 0$ for all i and such that $\nabla_x h_1(\overline{x}, \epsilon), \ldots, \nabla_x h_p(\overline{x}, \epsilon)$ are linearly independent. For $P(\epsilon)$ convex, MFCQ holding at each feasible point and GS are equivalent.

The following well known results can now be stated, for the standard nonparametric problem P(ϵ), i. e., with ϵ fixed:

a) SOSC($\overline{x}, u, w, \epsilon$) implies that $\overline{x}$ is a *strict local minimizer* (i. e., the unique global minimizer over the intersection of the feasible region and some neighborhood of $\overline{x}$);
b) GSOSC($\overline{x}, u, w, \epsilon$), u, w, ϵ) and MFCQ($\overline{x}, \epsilon$) imply that $\overline{x}$ is an *isolated local minimizer* (i. e., the unique local minimizer in the intersection of the feasible region and some neighborhood of $\overline{x}$);
c) Although GSSOSC at $\overline{x}$ implies GSOSC at $\overline{x}$ and also implies SSOSC($\overline{x}, u, w, \epsilon$), which implies SOSC($\overline{x}, u, w, \epsilon$), u, w, ϵ), the condition GSSOSC alone does not imply that $\overline{x}$ is an isolated local minimizer. A constraint qualification is needed, along with each second order condition, as in b) above and as in Propositions 8, 9 and 10 below. In fact, even the much stronger conditions, KKT($\overline{x}, u, w, \epsilon$) and $z^{\intercal}\, \nabla_x^2\, L(x, u, w, \epsilon)z > 0$ for all $z \neq 0$ and any x and any $(u, w) \in K(\overline{x}, \epsilon)$, do not imply that $\overline{x}$ is an isolated local minimizer. (See the counterexample by S.M. Robinson in [2, p. 71].)
d) If x is a local minimizer and LI($\overline{x}, \epsilon$) or MFCQ($\overline{x}, \epsilon$) holds, then KKT($\overline{x}, u, w, \epsilon$) holds. If LI($\overline{x}, \epsilon$) holds, then the Lagrange multipliers (u, w) are unique. MFCQ($\overline{x}, \epsilon$) holds if and only if the set of associated Lagrange multipliers satisfying KKT($\overline{x}, u, w, \epsilon$) is nonempty, compact and convex. If LI($\overline{x}, \epsilon$) holds, then MFCQ($\overline{x}, \epsilon$) holds. Therefore, it follows from b) that SOSC($\overline{x}, u, w, \epsilon$) and LI($\overline{x}, \epsilon$) imply that $\overline{x}$ is an isolated local minimizer.

Remark 1 A stronger form of MFCQ is necessary and sufficient for a unique Lagrange multiplier, but will not be used here.

e) If problem P(ϵ) is convex, then any local solution is global and the solution set is convex, and if the KKT($\overline{x}, u, w, \epsilon$) holds, then $\overline{x}$ is a global solution. Also, if GS(ϵ) holds, then so does KKT(x, u, w, ϵ) at any local solution x.

With this brief perspective, we present several basic sensitivity and stability results that hold for problem P(ϵ). We avoid detail and focus only on certain key implications of the assumptions.

Proposition 2 *For the once differentiable problem with nonempty uniformly compact feasible region R(ϵ), for ϵ near $\overline{\epsilon}$, the optimal value function f^* is continuous at $\overline{\epsilon}$ if MFCQ holds for some $x \in S(\overline{\epsilon})$.*

Proposition 3 *The optimal value function f^* is convex on T if P(ϵ) is jointly convex in (x, ϵ) as defined. Assuming solution attainment, this further implies that f^* is continuous and directionally differentiable in the interior of T.*

Proposition 4 *If R does not vary with ϵ and f is concave in ϵ, and T is convex, then f^* is concave on T. Again, assuming the solution is attained, this means f^* is continuous and directionally differentiable in the interior of T.*

Proposition 5 *Suppose $R(\epsilon) \neq \emptyset$ and compact and does not change with ϵ, and assume f and $\nabla_\epsilon f$ are continuous in (x, ϵ). Then, at any $\epsilon \in T$, it follows that $S(\epsilon) \neq \emptyset$ and compact and the directional derivative $D_z f^*$ exists for any direction z and is given by*

$$D_z f^*(\epsilon) = \min_x \nabla_\epsilon f(x, \epsilon) z \quad s.t.\ x \in S(\epsilon). \tag{1}$$

Proposition 6 *Assume that the problem P(ϵ) is convex in x for each $\epsilon \in T$ convex and the problem functions are once continuously differentiable in (x, ϵ). Then, if $\overline{\epsilon} \in$ interior T and the set of points satisfying* KKT($x, u, w, \overline{\epsilon}$)) *is nonempty and bounded, or equivalently, if the solution set $S(\epsilon) \neq \emptyset$ and bounded (hence compact) and the Slater constraint qualification GS(ϵ) holds for P(ϵ) with $\epsilon = \overline{\epsilon} \in$ interior T, then in a neighborhood $N(\overline{\epsilon})$ of $\overline{\epsilon}$, GS(ϵ) holds, $S(\epsilon) \neq \emptyset$ (and $S(\epsilon)$ is convex) for each $\epsilon \in N(\overline{\epsilon})$ and $S(\epsilon)$ is uniformly compact in $N(\overline{\epsilon})$. Furthermore, f^* is continuous and directionally differentiable in $N(\overline{\epsilon})$ and in any direction z and*

$$D_z f^*(\epsilon) = \min_{x \in S(\epsilon)} \max_{(u,w) \in K(x,\epsilon)} \nabla_\epsilon L(x, u, w, \epsilon) z, \tag{2}$$

where $K(x, \epsilon)$ is the set of (u, w) such that KKT(x, u, w, ϵ) holds.

Remark 7 As indicated, the assumption GS is equivalent to MFCQ or to assuming $K(x, \epsilon) \neq \emptyset$ and bounded at a local solution. Dispensing with convexity but assuming LI(x, ϵ) holds for each $x \in S(\epsilon)$, rather than GS(ϵ), we have for each $K(x, \epsilon)$ a singleton set $\{(u(x), w(x)\}$ and hence

$$D_z f^*(\epsilon) = \min_{x \in S(\epsilon)} \nabla_\epsilon L(x, u(x, \epsilon), w(x, \epsilon), \epsilon) z. \tag{3}$$

Suppose we assume that the functions defining $P(\epsilon)$ are twice continuously differentiable in (x, ϵ). Then, we have the following second order results.

Proposition 8 *If* KKT$(\overline{x}, \overline{u}, \overline{w}, \epsilon)$, SOSC$(\overline{x}, \overline{u}, \overline{w}, \epsilon)$ *and* LI$(\overline{x}, \epsilon)$ *and* SCS$(\overline{x}, \epsilon)$ *hold for* $\overline{x} \in R(\epsilon)$ *at* $\epsilon = \overline{\epsilon} \in$ interior T, *then there exists a locally unique and once continuously differentiable vector function (x, u, w) such that* $[x(\overline{\epsilon}), u(\overline{\epsilon}), w(\overline{\epsilon})] = (\overline{x}, \overline{u}, \overline{w})$ *and such that these assumptions continue to hold at [x(ϵ), u(ϵ), w(ϵ)] in a neighborhood* $N(\overline{\epsilon})$ *of* $\overline{\epsilon}$. *This implies that x(ϵ) is an isolated local minimizer with associated unique Lagrange multipliers [u(ϵ), w(ϵ)]. Furthermore,* f^* *is continuous and in fact twice continuously differentiable in* $N(\overline{\epsilon})$, *where* $f^*(\epsilon) = f[x(\epsilon), \epsilon]$.

Proposition 9 *Suppose* KKT$(\overline{x}, \overline{u}, \overline{w}, \epsilon)$ *and* SSOSC$(\overline{x}, \overline{u}, \overline{w}, \epsilon)$ *and* LI$(\overline{x}, \epsilon)$ *hold for* $\overline{x} \in R(\overline{\epsilon})$ *and* $\overline{\epsilon} \in$ interior T. *Then there exists (x, u, w) continuous and locally unique in* $N(\overline{\epsilon})$ *such that* $[x(\overline{\epsilon}), u(\overline{\epsilon}), w(\overline{\epsilon})] = (\overline{x}, \overline{u}, \overline{w})$ *and such that the assumptions persist at [x(ϵ), u(ϵ), w(ϵ)] and as before, x(ϵ) is an isolated local minimizer with associated unique Lagrange multipliers [u(ϵ), w(ϵ)]. Now, it turns out that for* $\epsilon \in N(\overline{\epsilon})$, *x, u, w are Lipschitz and directionally differentiable, and* f^* *is continuous and once continuously differentiable and twice directionally differentiable.*

Proposition 10 *If we again assume* KKT$(\overline{x}, \overline{u}, \overline{w}, \epsilon)$ *and further strengthen the second order conditions to* GSSOSC$(\overline{x}, u, w, \epsilon)$ *and now assume* MFCQ$(\overline{x}, \epsilon)$, *for* $\overline{x} \in R(\epsilon)$ *and* $\epsilon = \overline{\epsilon} \in$ interior T, *it follows that there exists a locally unique vector function x for* $\epsilon \in N(\overline{\epsilon})$ *such that* $x(\overline{\epsilon}) = \overline{x}$ *and once again the assumptions continue to persist at x and its associated nonempty compact convex set of Lagrange multipliers also exists as a result of MFCQ continuing to hold in* $N(\overline{\epsilon})$. *Again, x(ϵ) is an isolated local minimizer, but now we can show only that x is continuous and* f^* *is once directionally differentiable. It follows that for* $\epsilon \in N(\overline{\epsilon})$,

$$D_z f^*(\epsilon) = \max_{(u,w)K[x(\epsilon),\epsilon]} \nabla_\epsilon L[x(\epsilon), u, w, \epsilon]z. \tag{4}$$

Propositions 6–10 demonstrate what we have called *assumption stability* (i. e., persistence of initial assumptions). A unique local solution with the given properties continues to be locally unique and continuous and sometimes even differentiable, with continuous small data changes, and satisfying characterizations follow. All of the results we have given are now very well known and have been finely tuned, most therefore invoking close to 'minimal assumptions'. Weaker conditions will generally significantly change the conclusions. For example if GSSOSC is replaced by the weaker condition GSOSC in the assumptions given in Proposition 10, then the perturbed solution $x(\epsilon)$ need no longer be locally unique, although the initial solution $\overline{x} = x(\overline{\epsilon})$ is an isolated local minimizer. Assumption stability is lost, since GSOSC does not persist.

We offer a few observations concerning the rate of change of the optimal value $f^*(\epsilon)$, with data changes. See (2). $D_z f^*$ is itself an optimal value, a contribution to which comes either from f or from the g_i and h_j, through the Lagrangian, its value generally depending on the problem solution set and the optimal set of Lagrange multipliers. Note that if the constraints are not dependent on the parameter ϵ, then (2), (3) and (4) all collapse to formula (1): $D_z f^*(\epsilon) = \min_{S(\epsilon)} \nabla_\epsilon f(x, \epsilon)z$, formula (4) reducing to (1) without the min, as expected from Proposition 5. Note further that this formula does not depend on the Lagrange multipliers, but only on the behavior of f over the solution set. On the other hand, if f does not depend on ϵ, the dependency is all on the constraints and multipliers. In particular, consider the so-called *right-hand side perturbation problem*

$$P_1(\epsilon) \quad \begin{cases} \min & f(x) \\ \text{s.t.} & g_i(x) \geq \epsilon_i \quad (i = 1, \ldots, m), \\ & h_j(x) = \epsilon_j + m \quad (j = 1, \ldots, p). \end{cases}$$

Then, applying the results given in Propositions 6 and 10, assuming for simplicity that the local solution $x(\epsilon)$ and its associated optimal Lagrange multipliers $(u(\epsilon), w(\epsilon))$ are unique, and using formulas (2), (3) or (4), we find that

$$D_z f^*(\epsilon) = [u(\epsilon), -w(\epsilon)]^\top z \tag{5}$$

Thus, the directional rate of change of f^* with respect to changes in the constraint values is captured entirely by the optimal Lagrange multipliers, e. g., at the local minimizer $x(\epsilon)$ the rate of change of $f^*(\epsilon)$ along a unit vector in the direction of ϵ_i, $1 \leq i \leq m$, is given by $u_i(\epsilon)$. This is an extremely important and well known result. In ap-

plications, it translates to the multipliers being *shadow prices* (i. e., imputed values) of resource levels.

Two simple examples may help to clarify some of these results. The interested reader should graph these problems to appreciate the geometry.

Example 11 (Assume $|\epsilon| \leq 1/2$.)

$$\mathrm{P}(\epsilon) \quad \begin{cases} \min & \epsilon x_1 + x_2 \\ \text{s.t.} & x_1 \geq 0, \quad x_2 \geq 0, \\ & x_1 + x_2 \geq 1, \quad x_1^2 + x_2^2 \leq 4. \end{cases}$$

The solution is $S(0) = \{x \in E^2 : 1 \leq x_1 \leq 2, x_2 = 0\}, f^*(0) = 0$; $S(\epsilon) = (1, 0)$ and $f^*(\epsilon) = \epsilon$ for $\epsilon > 0$; and $S(\epsilon) = (2, 0)$ with $f^*(\epsilon) = 2\epsilon$ for $\epsilon < 0$. It follows that $D_z f^*(0) = \min_{S(0)} \nabla_\epsilon f[x(0), 0]z = \min_{S(0)} x_1 z = z$ if $z > 0$ and $2z$ if $z < 0$, which agrees with the values that follow by direct calculation of the closed form solution. We also note that the optimal value is concave, agreeing with results given previously, since f is concave in ϵ and the feasible region R is fixed. (Note that the problem is convex in x, Slater's condition holds, and $S(0)$ is bounded, so Propositions 4, 5 and 6 all apply.)

The next example is trivial, but illustrative.

Example 12

$$\begin{cases} \min & x_2 \\ \text{s.t.} & x_2 - x_1^2 \geq \epsilon. \end{cases}$$

The solution is $x(\epsilon) = (0, \epsilon)$, the optimal value $f^*(\epsilon) = \epsilon$ and the optimal Lagrange multiplier is $u(\epsilon) = 1$, for any ϵ. The problem is jointly convex and Slater's condition holds. The solution is unique, LI and SCS hold and in fact all the given second order conditions hold for this problem, so all the results we have given for these will hold. In particular, we see that $D_z f^*(\epsilon) = u(\epsilon)z = z$, as expected. Note that f^* is convex and differentiable and $x(\epsilon)$ and $u(\epsilon)$ are unique and differentiable, all as predicted by the theory.

There are many variations and extensions of the small sample of results that we have presented here. We have only been able to give the reader a flavor of this interesting and important subject.

Many references exist and many would have to be cited, if we were to give proper credit to all the many researchers who have contributed, even to the handful of results presented. Rather than attempt this, we refer the reader to three references. These references contain numerous tutorials and hundreds of additional references and should provide a quicker introduction to the subject than initially attempting to study scholarly papers scattered in the literature.

See also

- Nonlocal Sensitivity Analysis with Automatic Differentiation
- Parametric Global Optimization: Sensitivity
- Sensitivity Analysis of Complementarity Problems
- Sensitivity Analysis of Variational Inequality Problems
- Sensitivity and Stability in NLP
- Sensitivity and Stability in NLP: Approximation

References

1. Fiacco AV (ed) (1990) Optimization with data perturbations. Ann Oper Res, vol 27. Baltzer, Basel
2. Fiacco AV (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Acad. Press, New York
3. Fiacco AV (ed) (1998) Mathematical programming with data perturbations. Lecture Notes Pure and Applied Math. M. Dekker, New York

Sequential Cutting Plane Algorithm

Claus Still, Tapio Westerlund
Process Design Laboratory, Åbo Akademi University, Åbo, Finland

MSC2000: 90C11, 90C26

Article Outline

Keywords and Phrases
Formulation
Methods
  Branch and Bound
  NLP Iteration
  Algorithm Pseudo-Code
Cases
Conclusions
References

Keywords and Phrases

Mixed integer nonlinear programming; Extended cutting plane; Sequential cutting plane; Branch and bound

The SCP (Sequential Cutting Plane) algorithm [6,7,8] can be used for solving both NLP (Nonlinear Programming) and MINLP (Mixed Integer Nonlinear Programming) problems efficiently. For convex problems the algorithm finds global optimal solutions. For non-convex problems, global optimality cannot be guaranteed. Nevertheless, the algorithm can also be used on non-convex problems to find good approximations of the global optimal solution.

The SCP algorithm presented here uses a branch and bound strategy to solve MINLP problems where an NLP problem is solved in each integer node of the tree. The NLP problems are not solved to optimality, rather one iteration step is taken at each integer node of the tree and linearizations of the nonlinear constraints are added as cuts to the problem. The iteration step consists of performing an NLP iteration as described in [6,8], where the algorithm solves a sequence of linear programming problems. Note that in [7], the approach was different than the one used here. In [7], an NLP iteration was performed in each node of the branch and bound tree, which can be inefficient in difficult combinatorial problems where the branch and bound tree is large.

Formulation

The SCP algorithm solves problems of the form

$$\begin{aligned} \min \quad & f(x,y), \\ \text{s.t.} \quad & g_j(x,y) \le 0, j = 1,\dots,m, x \in X,\ y \in Y\,. \end{aligned} \tag{1}$$

The set X is a bounded, box-constrained set of the form $X = \{x \in R^{n_r} | x^{LB} \le x \le x^{UB}\}$ and the set Y is a finite bounded set $Y = \{y \in Z^{n_z} | y^{LB} \le y \le y^{UB}\}$. The functions f and g are convex and continuously differentiable.

The performance of the algorithm on a difficult set of block optimization problems, see [2], is presented. The ECP (Extended Cutting Plane) algorithm [9,10] proved to be very efficient in [2] for solving these types of optimization problems. The SCP algorithm further enhances the ideas from the Extended Cutting Plane algorithm in order to solve these difficult MINLP problems even more efficiently.

Methods

The problem (1) is solved by the SCP algorithm by doing a normal branch and bound procedure on a relaxed version of (1). In each integer node of the tree, the integer variables are fixed. An NLP iteration is then performed on the NLP subproblem obtained by fixing the integer variables. If the iterate is optimal, the solution in the integer node is an upper bound on the optimal solution of the original problem (1) and can be used for dropping nodes from the tree. If not, linearizations of the nonlinear constraints are added in the current iterate as cuts to the relaxed version of (1) and the branching process is continued.

Branch and Bound

The root node $\tilde{P}^1$ of the tree is constructed by relaxing (1) such that the integer requirements and the nonlinear constraints from the problem formulation are dropped. If some of the constraints $g_j(x,y) \le 0,\ j = 1,\dots,m$, are linear, they can be kept in the relaxed problem.

A branch and bound procedure, see [4], is then performed on this linear relaxation until an integer feasible node $\tilde{P}^k$ is obtained. In this node, the integer variables y^k are fixed and an NLP iteration is performed in order to solve the NLP subproblem

$$\begin{aligned} \min \quad & f(x,y^k)\,, \\ \text{s.t.} \quad & g(x,y^k) \le 0\,, x \in X\,. \end{aligned} \tag{2}$$

Note that the problem is not solved to optimality. Instead one NLP iteration is performed in order to get a good approximation of the optimal solution to the subproblem. Let x^k be the iterate obtained after the NLP iteration on (2).

The iterate (x^k, y^k) can be used to add linearizations of the violated and active nonlinear constraints in (x^k, y^k),

$$g_j(x^k,y^k) + \nabla g_j(x^k,y^k)^T(x - x^k, y - y^k) \le 0\,, \quad j \in \{j : g_j(x^k,y^k) \ge 0\}\,,$$

to the set of cuts Ω_k in order to obtain a new set of cuts Ω_{k+1}.

If x^k is the optimal solution to (2), then $f(x^k, y^k)$ is an upper bound of the optimal solution to (1) and can be used to drop unexplored nodes from the branch and bound tree whenever the lower bounds of the nodes are greater than this upper bound.

NLP Iteration

The NLP subproblem (2) is solved by performing an NLP iteration [6,8]. In an NLP iteration, a sequence of linear programming (LP) subiterations is performed. In each subiteration (i) within the NLP iteration, an LP problem $LP(x^{(i)})$ is generated in the current iterate $(x^{(i)}, y^k)$. The LP problem $LP(x^{(i)})$ is of the form

$$\begin{aligned} \min \quad & \nabla f(x^{(i)}, y^k)^T d + Ct \,, \\ \text{s.t.} \quad & g_j(x^{(i)}, y^k) + \nabla g_j(x^{(i)}, y^k)^T d - t \le 0 \,, \\ & j = 1, \ldots, m \,, \\ & (d^{(r)})^T H^{(i)} d = 0, r = 1, \ldots, i-1, \quad i>1 \,, \\ & t \ge 0 \,, \\ & x^{(i)} + d_x \in X, d_y = \mathbf{0} \,, \end{aligned}$$

where $d = (d_x, d_y)$, and $d^{(r)}, r = 1, \ldots, i-1$, are the previously obtained search directions within the NLP iteration. Also, $H^{(i)}$ is the current estimate of the Hessian of the Lagrangian

$$L(x, \lambda) = f(x, y^k) + \sum_{j=1}^{m} \lambda_j g_j(x, y^k) \,.$$

The BFGS update formula was used in the implementation of the algorithm to approximate $H^{(i)}$.

The dual optimal solution to the LP problem $LP(x^{(i)})$ provides a Lagrange multiplier estimate, which can be used when estimating the Hessian of the Lagrangian. Consequently, the Hessian approximation can be updated in each LP subiteration.

The solution to each LP problem $LP(x^{(i)})$ provides a search direction $d^{(i)} = (d_x^{(i)}, \mathbf{0})$. A line search is performed in the obtained search direction minimizing a modified function based on the Lagrangian of (2),

$$\tilde{L}(x, \lambda) = f(x, y^k) + \sum_{j=1}^{m} \lambda_j g_j(x, y^k)^+ + \rho \sum_{j=1}^{m} (g_j(x, y^k)^+)^2 \,,$$

where $g_j(x, y^k)^+ = \max(g_j(x, y^k), 0)$ and $\rho\,(>0)$ is a penalty parameter.

The current iterate $(x^{(i)}, y^k)$ is then updated using the solution $\alpha^{(i)}$ of the line search such that $x^{(i+1)} = x^{(i)} + \alpha^{(i)} d_x^{(i)}$, where $\alpha^{(i)}$ is the step length found in the line search.

A new LP problem is then constructed in the updated iterate $(x^{(i+1)}, y^k)$. The new LP problem is constructed in a similar way as the previous LP problem with equality constraints requiring the new search direction to be a conjugate direction to the previously obtained search directions with respect to the current estimate of the Hessian of the Lagrangian.

The linear equality constraints

$$(d^{(r)})^T H^{(i)} d = 0, \quad r = 1, \ldots, i-1 \,,$$

are cutting hyperplanes ensuring that d will be a conjugate direction to the old directions $d^{(r)}$.

Initialize: Do a number of NLP iterations on the continuous relaxation of (1) to obtain the initial iterate for the problem. Construct the relaxed node $\bar{P}^1$ and insert it into the branch and bound tree as the top node and set the upper bound $U = \infty$. Add cutting planes for the initial iterate to the set of cutting planes Ω_1. Let $k = 1$.

While (there are unexamined nodes in the tree) **do**

1. Do normal branching on the branch and bound tree including the cutting planes Ω_k as linear cuts until integer solution $(\tilde{x}^k, y^k)$ found.
2. Fix y^k and perform an NLP iteration on (2). Let the iterate after the NLP iteration be x^k.
3. **If** solution x^k optimal for (2) Then update upper bound $U := \min\{U, f(x^k, y^k)\}$.
4. **If** (2) is found to be infeasible Then let x^k be the last iterate from the NLP iteration.
5. Let $\Omega_{k+1} = \Omega_k$. Add cutting planes generated in (x^k, y^k) for the active and violated constraints and add them to Ω_{k+1}. Let $k := k+1$.

End While

Sequential Cutting Plane Algorithm, Algorithm 1
Pseudo-code for the SCP algorithm

The new LP problem is then solved, a new line search performed and the iterate updated again. The procedure is repeated until the LP problem becomes infeasible, a sufficient number of steps have been taken or the current solution to the LP problem is sufficiently close to zero.

Normally, the NLP iteration would then be repeated until an optimal point is found. However, in this version of the algorithm only one NLP iteration is performed in each integer node of the branch and bound tree in order to improve the performance of the algorithm.

Convergence properties of the NLP version of the SCP algorithm have been analyzed in earlier papers, see [6] and [8].

Algorithm Pseudo-Code

The algorithm is summarized in Algorithm 1.

Cases

The performance of the algorithm is illustrated on a set of difficult block optimization problems presented in [2]. The paper concerns optimizing the arrangement of a number of departments with unequal area requirements. It is possible to formulate the problems as mixed integer nonlinear programming problems, where the constraints are department and floor area requirements as well as department locational restrictions. The optimization target is to minimize the cost associated with the projected interactions between departments. In [2], the Extended Cutting Plane method was compared to a number of commercial algorithms and proved to be superior to the other solvers.

In Table 1 the results for the solvers in [2] are summarized in terms of the number of problems solved to optimality, number of problems for which a feasible solution was obtained and number of problems for which no solution was obtained within 12 h of CPU time. More information about the solvers used can be found in [1].

Sequential Cutting Plane Algorithm, Table 1
Performance of the solvers on the block layout problem as reported in [2]

Solution	BARON	DICOPT	MINLPbb	SBB	α-ECP
Optimal	13	5	5	5	31
Feasible	16	2	27	23	1
No solution	3	25	0	4	0

The results are excellent for the α-ECP algorithm on these test problems. It only failed to solve one problem to global optimality and obtained the best integer feasible solution for the problem it could not solve to global optimality.

In Fig. 1 the results of the new SCP algorithm are compared with the α-ECP algorithm using a performance profile [3]. The results are also compared with a version where each integer node is solved to optimality by repeating the NLP iterations until an optimal solution to (2) is found. This version of the SCP algorithm is denoted SCP-NLP. Note that if you solve the integer nodes to optimality, the procedure is similar to the method described in [5]. CPLEX was used as the MILP solver and the NLP part of the algorithm was implemented in MATLAB. A maximum of 12 CPU h for each problem was used when solving the problems.

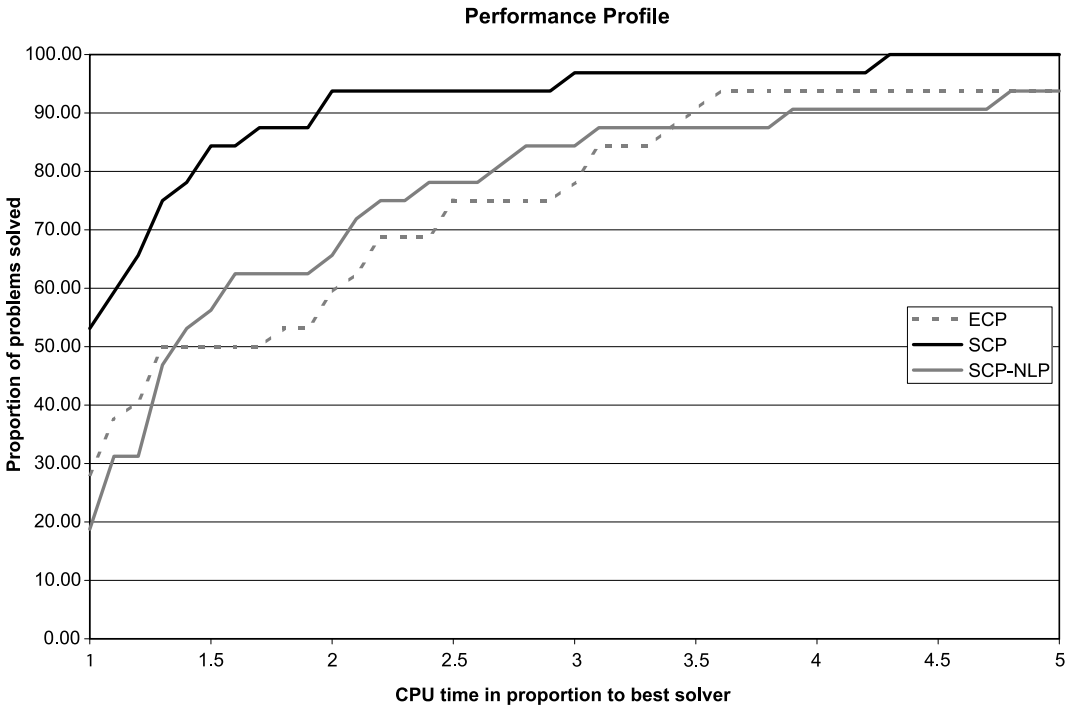
As can be seen from the results, the SCP algorithm performed very well on the test problems. In more than half of the cases it was the fastest solver. Considering that some of these problems can take several hours to solve, the performance improvement is significant.

In Fig. 2 the best-known solutions of the SCP and α-ECP solvers, when running the solvers for a maximum of 60 CPU s, are compared with the best-known solutions for the other solvers in [2], when running these solvers for a maximum of 12 CPU h.

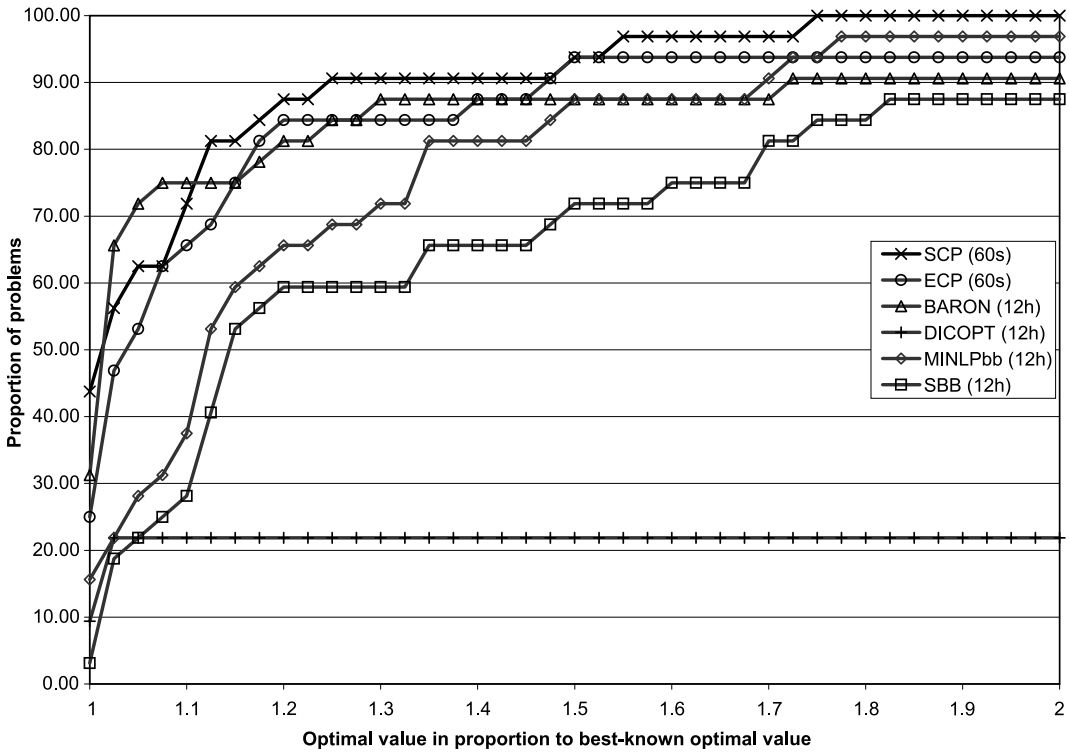
Note that both the SCP and α-ECP solvers return very good solutions after 60 CPU s in comparison with the other solvers that were run for a maximum of 12 CPU h. In fact, the SCP algorithm reported the best-known solution in almost half of the cases. Thus, the SCP algorithm can also be used for efficiently finding good solution candidates, when the problems are too difficult to solve to the global optimum.

Conclusions

The Sequential Cutting Plane algorithm is well-suited for solving mixed integer nonlinear programming problems. The numerical results above for a set of challenging block layout problems support this claim. Another advantage is that the algorithm will efficiently find a good feasible solution to a problem, even when the



Sequential Cutting Plane Algorithm, Figure 1
Performance profile comparing CPU times for solving the block layout problems



Sequential Cutting Plane Algorithm, Figure 2
Performance profile comparing the optimal values of the solvers. Observe that 60 CPU seconds is used for SCP/ECP and 12 CPU hours for the other solvers

problem is not solved to optimality. Thus, difficult combinatorial optimization problems in real-world applications could be solved by combining the SCP algorithm with some non-deterministic approach such as an evolutionary algorithm.

References

1. Bussieck MR, Pruessner A (2003) Mixed-Integer Nonlinear Programming. SIAG/OPT Newsl Views-and-News 14:19–22
2. Castillo I, Westerlund J, Emet S, Westerlund T (2005) Optimization of Block Layout Design Problems with Unequal Areas: A Comparison of MILP and MINLP Optimization Methods. Comput Chem Eng 30:54–69
3. Dolan ED, Moré JJ (2002) Benchmarking Optimization Software with Performance Profiles. Math Program 91:201–213
4. Land AH, Doig AG (1960) An Automatic Method of Solving Discrete Programming Problems. Econom 28:497–520
5. Quesada I, Grossmann IE (1992) An LP/NLP Based Branch and Bound Algorithm for Convex MINLP Optimization Problems. Comput Chem Eng 16:937–947
6. Still C, Westerlund T (2006) A Sequential Cutting Plane Algorithm for Solving Convex NLP Problems. Eur J Oper Res 173:444–464
7. Still C, Westerlund T (2006) Solving Convex MINLP Optimization Problems Using a Sequential Cutting Plane Algorithm. Comput Optim Appl 34:63–83
8. Still C, Westerlund T (2007) A Linear Programming-Based Optimization Algorithm for Solving Nonlinear Programming Problems. Eur J Oper Res (submitted)
9. Westerlund T, Pörn R (2002) Solving Pseudo-Convex Mixed Integer Optimization Problems by Cutting Plane Techniques. Optim Eng 3:253–280
10. Westerlund T, Skrifvars H, Harjunkoski I, Pörn R (1998) An Extended Cutting Plane Method for a Class of Non-Convex MINLP Problems. Comput Chem Eng 22:357–365

Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems

SQPIP

MATTHIAS HEINKENSCHLOSS
Department Computational and Applied Math., Rice University, Houston, USA

MSC2000: 49M99, 49K20, 90C55

Article Outline

Keywords
See also
References

Keywords

Optimal control problem; Partial differential equations; Sequential quadratic programming; Interior point methods

For the purpose of this article, *distributed optimal control problems* are optimization problems which are posed in function space and in which one of the constraints is a partial differential equation. An example of such a problem is

$$\begin{cases} \min & \frac{1}{2}\int_\Omega (y(x)-\widehat{y}(x))^2\,dx + \frac{\omega}{2}\int_\Gamma u^2(x)\,dx \\ \text{s.t.} & -\Delta y(x) + y^3(x) - y(x) = 0 \quad \text{in } \Omega, \\ & \frac{\partial}{\partial n} y(x) = u(x) \quad \text{in } \Gamma, \\ & u_l(x) \le u(x) \le u_u(x) \quad \text{a.e. on } \Gamma, \\ & y_l(x) \le y(x) \le y_u(x) \quad \text{a.e. in } \Omega, \end{cases}$$

where Ω is a bounded open domain in $\mathbf{R}^2$, Γ is the boundary of Ω, $\omega > 0$ is a given parameter, and the functions u_l, u_u, $\widehat{y}$, y_l, y_u are given. The functions y and u are called the *states* and *controls*, respectively. The partial differential equation including the boundary conditions is called the *state equation* or the *governing equation*.

Abstractly, a distributed optimal control problem may be written as

$$\min J(y,u), \tag{1}$$

such that

$$c(y,u) = 0, \tag{2}$$

$$u \in U_{\text{ad}}, \tag{3}$$

$$y \in Y_{\text{ad}}, \tag{4}$$

where Y, C are Hilbert spaces, U is a Banach space, $Y_{\text{ad}} \subset Y$, $U_{\text{ad}} \subset U$ are closed convex sets, and $J: Y \times U \to \mathbf{R}$, c: $Y \times U \to C$ are twice Fréchet differentiable functions.

In the previous example, $Y = H^1(\Omega)$, $C = (H^1(\Omega))_*$, $U = L^2(\Gamma)$, and $Y_{\text{ad}} = \{y \in H^1(\Omega): y_l \le y \le y_u \text{ a.e. in } \Omega\}$, $U_{\text{ad}} = \{u \in L^2(\Gamma): u_l \le u \le u_u \text{ a.e. in } \Gamma\}$. Several optimal control problems that fit into this framework are studied in [4,7,8,12]. This problem formulation also covers *optimal design problems* [5,9] and *parameter identification problems*.

After a discretization of the problem (1)–(4) using, e. g., finite elements, one often obtains a *nonlinear programming problem* of the form

$$\min J^h(y^h, u^h), \tag{5}$$

such that

$$c^h(y^h, u^h) = 0, \tag{6}$$

$$u_l^h \le u^h \le u_u^h, \tag{7}$$

$$y_l^h \le y^h \le y_u^h, \tag{8}$$

where $y^h \in \mathbf{R}^{n_y}$, $u^h \in \mathbf{R}^{n_u}$ and $J^h \mathbf{R}^{n_y+n_u} \to \mathbf{R}$, $c^h: \mathbf{R}^{n_y+n_u} \to \mathbf{R}^{n_y}$ are twice differentiable functions. The number of discretized states n_y tends to be large. Depending on the type of control, n_u can be small (boundary control) or large (distributed control).

Sequential quadratic programming interior point (SQPIP) methods have been used to solve distributed optimal control problems. See, e. g., [2,3,5,7,9,10,11]. While the various SQPIP methods differ, they share some important design features. In each iteration a subproblem is solved that only involves a linearization of the state equation (2) or (6) and a quadratic approximation of the Lagrangian $L^h(y^h, u^h, \lambda^h) = J^h(y^h, u^h) + \langle c^h(y^h, u^h), \lambda^h \rangle$. All iterates stay strictly feasible with respect to the bounds, but the nonlinear state equation is only satisfied in the limit. SQPIP methods attempt to achieve feasibility and optimality at the same time. Such all-at-once approaches are usually more efficient than solution approaches that maintain feasibility of the nonlinear state equation at every iteration. For a general introduction to SQP methods and interior point methods for nonlinear programming problems see ► Successive quadratic programming: Solution by active sets and interior point methods; ► Linear programming: Interior point methods.

This article focuses on the application of SQPIP methods to distributed optimal control problems. For such problems *affine scaling SQPIP methods* [3] and various versions of *primal-dual SQPIP methods* ([2,5,7,9,10,11]) have been used. Primal-dual interior point methods have been applied to the nonlinear problem (5)–(8) directly or as inequality constrained quadratic programming subproblem solvers. In all references above, SQPIP methods have been applied to *discretized optimal control problems* (5)–(8). While these applications have been successful, there are several open research issues. The development of SQPIP methods for large scale nonlinear programming problems is a very active research area. Additional research issues arise when SQPIP methods are applied to distributed optimal control problems. The latter research issues and achievements will be described in the following.

Distributed optimal control problems have a particular problem structure. It is derived from the underlying infinite-dimensional problem and from the division of optimization variables into states and controls. Since one is interested in the solution of the infinite-dimensional problem (1)–(4), one needs to consider sequences of discretized problems (5)–(8). While for a fixed discretization the nonlinear programming framework may be applicable, the underlying infinite-dimensional problem is important when sequences of successively refined discretizations are considered. It is important to understand the convergence behavior of the SQPIP algorithm in the infinite-dimensional setting, because this often dominates the convergence behavior of the algorithm for the discretized problem when the discretization is fine. Convergence results in [2,11] for SQPIP algorithms applied to elliptic optimal control problems show that in several cases the number of optimization iterations increases only slowly, if at all, when the discretization is refined. This indicates that there might be an underlying infinite-dimensional convergence theory. However, such a theory is not yet known for most interior point algorithms. Currently, the only analyses of interior point methods for infinite-dimensional problems are [6,13,14].

Most available SQPIP algorithms for nonlinear programming use sparse direct linear algebra to solve subproblems. This is not suitable for many distributed optimal control problems. Here, subproblems involve the solution of linearized partial differential equations,

which is best done using problem specific solvers, such as multigrid or domain decomposition techniques. How to adjust the accuracy of iterative subproblem solves within SQPIP methods to obtain an efficient, globally convergent algorithm is not yet completely understood. An analysis of the influence of inexact problem information on the convergence of SQPIP methods is also important because discretizations of distributed optimal control problems can lead to errors in the derivative information. For example, this may happen when derivative information for the discretized problem (5)–(8) is computed by discretizing the Fréchet-derivatives of the infinite-dimensional problem (1)–(4). In such cases error in derivative information often goes to zero as the discretization is refined. An understanding of the influence of inexact derivative information on the convergence of SQPIP methods is necessary for the development of efficient and robust algorithms, including the development of grid refinement strategies within the optimization.

When considering SQPIP methods for distributed optimal control problems it is useful to distinguish between problems with and without state constraints (4), (8). Problems (1)–(3) or (5)–(7) which include only control constraints are often easier to solve. For control constrained problems there exist more solution approaches, like projection methods, than for state constrained problems. If the controls u are in $U = L^p$, $p \in [1, \infty)$ or $p = \infty$, which is, e. g., the case in the example problem at the beginning of this article, then optimality conditions for the infinite-dimensional problem can be formulated in a form that is suitable for the development of interior point methods. In particular, under suitable assumptions it is possible to show that Lagrange multipliers corresponding to (3) are in L^∞. See, e. g., [13,14]. Development of optimality conditions for distributed optimal control problems (1)–(4) with state constraints that are suitable for use in optimization algorithms is an active research area. Existing results are less general than the ones for control constrained problems. Moreover, the Lagrange multipliers corresponding to the state constraints (4), which are optimization variables in most SQPIP methods, are usually only measures that can not be represented by functions in L^p. See, e. g., the discussion and references in [10]. For the discretized problem (5)–(7) it is often possible to show that the partial Jacobian $\partial/\partial y)c^h(y^h, u^h)$ is invertible for all $u_l^h \le u^h \le u_u^h$, $y^h \in \mathbf{R}^{n_y}$. In this case the Jacobian

$$\begin{pmatrix} \frac{\partial}{\partial y} c^h(y^h, u^h) & \frac{\partial}{\partial u} c^h(y^h, u^h) \\ 0 & \pm I_{A(u^h)} \end{pmatrix}$$

of the active constraints (6), (7) has full row rank, i. e., it is appropriate to assume the *linear independence constraint qualification* for the problem (5)–(7). This assumption is made in many local convergence analyses for SQPIP methods for nonlinear programming. For the problem (5)–(8) with state constraints, this assumptions is often too restrictive. For example, the linear independence constraint qualification will be violated if more that n_u+n_y control and state constraints (7), (8) are active ([1,15]). Finally, the quality of some preconditioners for the iterative solution of subproblems within SQPIP method is improved when control constraints (7) are present, but decreased if state constraints (8) are given (see, e. g., [1]).

SQPIP methods for distributed optimal control problems is a rapidly developing area. This class of optimization algorithms shows great promise, especially for problems with state constraints. The reader will find detailed discussions in the references and related information in ▶ Successive quadratic programming: Solution by active sets and interior point methods; ▶ Linear programming: Interior point methods on SQP methods and interior point methods for nonlinear programming problems.

See also

- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Entropy Optimization: Interior Point Methods
- ▶ Feasible Sequential Quadratic Programming
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Homogeneous Selfdual Methods for Linear Programming
- ▶ Infinite Horizon Control and Dynamic Games

- ▶ Interior Point Methods for Semidefinite Programming
- ▶ Linear Programming: Interior Point Methods
- ▶ Linear Programming: Karmarkar Projective Algorithm
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Optimal Control of a Flexible Arm
- ▶ Optimization with Equilibrium Constraints: A Piecewise SQP Approach
- ▶ Potential Reduction Methods for Linear Programming
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming and Control Problems
- ▶ Suboptimal Control
- ▶ Successive Quadratic Programming
- ▶ Successive Quadratic Programming: Applications in Distillation Systems
- ▶ Successive Quadratic Programming: Applications in the Process Industry
- ▶ Successive Quadratic Programming: Decomposition Methods
- ▶ Successive Quadratic Programming: Full Space Methods
- ▶ Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Battermann A, Heinkenschloss M (1998) Preconditioners for Karush–Kuhn–Tucker systems arising in the optimal control of distributed systems. In: Desch W, Kappel F, Kunisch K (eds) Optimal Control of Partial Differential Equations. Internat Ser Num Math. Birkhäuser, Basel, pp 15–32
2. Bergounioux M, Haddou M, Hintermüller M, Kunisch K (1998) A comparison of interior-point methods and a Moreau–Yosida based active set strategy for constrained optimal control problems. Techn. Report Inst. Math. Karl-Franzens-Univ. Graz
3. Dennis JE, Heinkenschloss M, Vicente LN (1998) Trust-region interior-point algorithms for a class of nonlinear programming problems. SIAM J Control Optim 36:1750–1794
4. Gunzburger MD, Hou LS (1996) Finite dimensional approximation of a class of constrained nonlinear optimal control problems. SIAM J Control Optim 34:1001–1043
5. Herskovits J, Laporte E, Le Tallec P, Santos G (1996) A quasi-Newton interior-point algorithm applied to constrained optimum design in computational fluid dynamics. Europ J Finite Elements 5:595–617
6. Ito S, Kelley CT, Sachs EW (1995) Inexact primal-dual interior-point iteration for linear programs in function spaces. Comput Optim Appl, 4:189–201
7. Leibfritz F, Sachs EW (2000) Inexact SQP interior-point methods and large-scale control problems. SIAM J Control Optim 38:272–293
8. Lions JL (1971) Optimal control of systems governed by partial differential equations. Springer, Berlin
9. Maar B, Schulz V (2000) Interior-point multigrid methods for topology optimization. Structural Optim 19:214–224
10. Mittelmann HD, Maurer H (2000) Interior-point methods for solving elliptic control problems with control and state constraints: boundary and distributed control. Comput Optim Appl 16:29–55
11. Mittelmann HD, Maurer H (2001) Interior-point methods for solving elliptic control problems with control and state constraints. Part 2: Distributed Control. Comput Optim Appl 18:141–160
12. Neittaanmäki P, Tiba D (1994) Optimal control of nonlinear parabolic systems. Theory, algorithms, and applications. M. Dekker, New York
13. Ulbrich M, Ulbrich S (2000) Superlinear convergence of affine-scaling interior-point Newton methods for infinite-dimensional nonlinear problems with pointwise bounds. SIAM J Control Optim 38:1938–1984
14. Ulbrich M, Ulbrich S, Heinkenschloss M (1999) Global convergence of affine-scaling interior-point Newton methods for infinite-dimensional nonlinear problems with pointwise bounds. SIAM J Control Optim 37:731–764
15. Vicente LN (1998) On interior-point Newton algorithms for discretized optimal control problems with state constraints. Optim Methods Softw 8:249–275

Sequential Simplex Method

SSM

VASSILIOS S. VASSILIADIS, RAÚL CONEJEROS
Chemical Engineering Department,
University Cambridge, Cambridge, UK

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Gradient-free minimization; Sequential simplex method; Expansion operations; Contraction operation; Reflection operations; Nondifferentiable objective functions

The sequential simplex method is due to the original work of W. Spendley, G.R. Hext and F.R. Himsworth [2]. It was later developed further by J.A. Nelder and R. Mead [1]. An exposition of the ideas underlying the method and its operation are as follows.

The minimization problem considered is:

$$\min_{\mathbf{x}} f(\mathbf{x}).$$

First, a *simplex* is defined as the geometric object formed by a set of $n + 1$ points in the case of n variables (dimensions). Equidistant points form a *regular simplex*. In two dimensions such an object is a triangle, in three dimensions it is a tetrahedron and so forth.

The basic idea of the *downhill simplex method* (for minimization of n-dimensional functions) rests on the ability to use the geometric object to move a vertex at a time in the direction of descending function values. To achieve this firstly one needs to define the initial layout of the vertices, and then apply general types of moves to modify the object. The basic moves are three, namely, *reflection*, *expansion*, and *contraction*. A suitable termination criterion is also needed. All these basic characteristics of the sequential simplex algorithm are outlined below.

1. Initial simplex construction

Generally, one can select any number of $n + 1$ points to form the *initial simplex*, as long as each subset of n of them is capable of spanning the n-dimensional space, in other words there are precisely n linearly independent vectors in the set. A simple procedure is to select for example an initial vertex point $\mathbf{x}^{(1)}$ and then arraying the remaining n points scaled in the coordinate directions:

$$\mathbf{x}^{(i)} = \mathbf{x}^{(1)} + \lambda_i \mathbf{e}^{(i-1)}, \quad i = 2, \ldots, n+1,$$

where λ_i is some set of constants which reflect the guess of the problem's characteristic scales in each of its variables. The vectors $\mathbf{e}^{(i)}$ are the unit vectors along each coordinate.

Having constructed the initial simplex each of the vertices must be evaluated in terms of the objective function value, leading to the corresponding values f_1, $\ldots$, f_{n+1}. Following this, we denote the corresponding index, function value and vertex vector for the lowest, highest and replacing new vertex by the triplets $\{l, f_l, \mathbf{x}^{(l)}\}$, $\{h, f_h, \mathbf{x}^{(h)}\}$, $\{r, f_r, \mathbf{x}^{(r)}\}$. It is also useful to define the *centroid of the simplex* (excluding the highest function value vertex) by:

$$\mathbf{x}^{(c)} = \frac{1}{n} \sum_{i=1, i\neq h}^{n+1} \mathbf{x}^{(i)}, \tag{1}$$

thus having a new triplet representing this centroid given by $\{c, f_c, \mathbf{x}^{(c)}\}$.

2. Reflection (main operation)

This corresponds to rejecting vertex h, with maximum function value among all other vertices, and replacing it by a reflected point r through the opposite face of the simplex. This is based on the expectation that such a new point will have a smaller value. The reflected point is constructed by:

$$\mathbf{x}^{(r)} = \mathbf{x}^{(c)} + \alpha_1(\mathbf{x}^{(c)} - \mathbf{x}^{(h)}),$$

where $\alpha_1 > 0$ is the *reflection coefficient*, determining how far the new point will be on the far side of the centroid vertex c. Clearly, the definition above results in $\mathbf{x}^{(r)}$ to lie on the line joining the vectors $\mathbf{x}^{(h)}$ and $\mathbf{x}^{(c)}$. The definition of the reflection coefficient is thus:

$$\alpha_1 = \frac{\left\|\mathbf{x}^{(r)} - \mathbf{x}^{(c)}\right\|_2}{\left\|\mathbf{x}^{(h)} - \mathbf{x}^{(c)}\right\|_2}.$$

If $f_l \leq f_r \leq f_h$, then the new vertex r is accepted, defining a new simplex. It is quite easy though for reflection-only based operation to get in a cyclic operation without improvement. This may happen if the simplex gets positioned in such a way that the reflected point gives the same function value as the original one (the highest) which it is replacing. One may be able to introduce rules, such as not allowing returns to be made to already visited locations, or by replacing the second highest value of the simplex by a new point.

If the function value is found to be $f_r < f_l$ then the expansion operation is carried out in step 3. If, on the other hand, the expansion produces a point for which f_r

$> f_l$ holds for all points in the simplex except the highest value one (point h) then the contraction operation is carried out in step 4.

3. Expansion

If during the *reflection* process above, it is found that the new point is better than all others in the previous simplex, that is $f_r < f_l$, then the reflection is successful in producing a new candidate minimum point. In such a case it is expected that the direction along the centroid defined in (1) and the replaced point h is a *descent direction*. Hence, it is desirable to move further along this direction by expanding the simplex. The expansion is defined by a new index e point:

$$\mathbf{x}^{(e)} = \mathbf{x}^{(c)} + \alpha_2(\mathbf{x}^{(r)} - \mathbf{x}^{(c)}),$$

again having as a definition of the *expansion coefficient* the ratio:

$$\alpha_2 = \frac{\left\|\mathbf{x}^{(e)} - \mathbf{x}^{(c)}\right\|_2}{\left\|\mathbf{x}^{(r)} - \mathbf{x}^{(c)}\right\|_2},$$

which indicates how much further the original replacement step is taken in the expansion phase.

If the procedure satisfies $f_e < f_l$ the point $\mathbf{x}^{(h)}$ is replaced by the point $\mathbf{x}^{(e)}$ and another reflection operation is restarted. If this does not improve the function value, that is $f_e > f_l$ holds, then the replacement is such that the point $\mathbf{x}^{(h)}$ is replaced by the point $\mathbf{x}^{(r)}$ and another reflection process is started again (going back to step 2).

4. Contraction

If the *reflection* process has produced a point for which $f_r > f_l$ holds for all points in the simplex except the highest value one (point h), then the point $\mathbf{x}^{(h)}$ is replaced by $\mathbf{x}^{(r)}$. But this will cause the new highest value point to be the one just introduced, hence it is necessary to contract the simplex by the following rule, where s indicates the new contracted point:

$$\mathbf{x}^{(s)} = \mathbf{x}^{(c)} + \alpha_3(\mathbf{x}^{(h)} - \mathbf{x}^{(c)}), \tag{2}$$

where the parameter α_3 is the *contraction coefficient*, such that $0 \le \alpha_3 \le 1$, and defines the ratio:

$$\alpha_3 = \frac{\left\|\mathbf{x}^{(s)} - \mathbf{x}^{(c)}\right\|_2}{\left\|\mathbf{x}^{(h)} - \mathbf{x}^{(c)}\right\|_2},$$

defining by how much the direction to the highest value vertex from the centroid is contracted.

If $f_r > f_h$, then the contraction operation in (2) is used, without of course replacing the original highest value point h.

If the point s is such that $f_s < \min\{f_h, f_r\}$ then point $\mathbf{x}^{(h)}$ is replaced by $\mathbf{x}^{(s)}$ and reflection operations resume (going to step 2).

If the contraction has failed, yielding $f_s \ge \min\{f_h, f_r\}$ then it is proposed to replace all points $\mathbf{x}^{(i)}$ by moving closer to the lowest value point using $(\mathbf{x}^{(i)} + \mathbf{x}^{(l)})/2$ and restarting reflection operations (going to step 2).

5. Termination criteria

Termination criteria are necessarily by the construction of the method going to have to be based on some 'average' value along all current vertices in the simplex. A suitable criterion, which may be tested each time the simplex has been modified by any of the above three operations, is the following:

$$\left(\frac{1}{n+1}\sum_{i=1}^{n+1}(f_i - f_0)^2\right)^{1/2} \le \epsilon,$$

where the value f_0 may be set to be the centroid calculated in (1), or one that includes all present points, i. e. even the highest point excluded in the summation in (1).

The above criterion is a standard deviation measure of all simplex points, which should be less or equal to some $\epsilon > 0$ specified tolerance on its value.

The various parameters $\alpha_1, \alpha_2, \alpha_3$ appearing in the various operations of the algorithm are either fixed, or can be adapted using line search type operations to find the best values that will enhance the desired effect of the operation involved (steps 2–4).

The sequential simplex method requires only function values generally, and is a simple algorithmic procedure. Generally, it is not the best derivative-free methods, requiring several function evaluations for larger problems (in terms of number of variables). However, whenever a quick and easy search phase is required for an unconstrained optimization problem, without too many variables, this method may be quite useful. Another advantage is that the method does not require the objective function $f(x)$ to be differentiable, hence it may be useful for such practical applications.

See also

- ▶ Convex-simplex Algorithm
- ▶ Cyclic Coordinate Method
- ▶ Lemke Method
- ▶ Linear Complementarity Problem
- ▶ Linear Programming
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Powell Method
- ▶ Rosenbrock Method

References

1. Nedler JA, Mead R (1965) A simplex method for function minimization. Comput J 7:308–313
2. Spendley W, Hext GR, Himsworth FR (1965) Sequential application of simplex design in optimization and evolutionary operations. Technometrics 4:441

Set Covering, Packing and Partitioning Problems

KARLA HOFFMAN[1], MANFRED PADBERG[2]
[1] Department of Systems Engineering and Operations Research, George Mason University, Fairfax, USA
[2] New York University, New York, USA

MSC2000: 90C10, 90C11, 90C27, 90C57

Article Outline

Keywords
Applicability of the Problem
Solution Approaches
Reformulation of the Linear Description of the Problem
Heuristics for the Set Partitioning and Covering Problems
Exact Solution Approaches to the Set Covering, Packing and Partitioning Problems
The Future
See also
References

Keywords

Integer programming; Combinatorial optimization; Enumeration; Polyhedral methods; Disjunctive programming; Lift-and-project

We consider the class of problems having the following structure:

$$\begin{cases} \min & cx \\ \text{s.t.} & Ax \geq e^{\top} \\ & x_j = 0, 1 \quad \text{for } j = 1, \ldots, n, \end{cases}$$

where A is a $m \times n$ matrix of zeros and ones, $e = (1, \ldots, 1)$ is a vector of m ones and c is a vector of n (arbitrary) rational components. This *pure 0–1 linear programming problem* is called the *set covering problem*. When the inequalities are replaced by equations the problem is called the *set partitioning problem*, and when all of the $\geq$ constraints are replaced by $\leq$ constraints, the problem is called the *set packing problem*.

Applicability of the Problem

Many applications arise having the packing, partitioning and covering structure. Delivery and routing problems, scheduling problems and location problems often take on a set covering structure whereby one wishes to assure that every customer is served by some location, vehicle or person. Other applications include switching theory, the testing of VLSI circuits, and line balancing. Similarly, scheduling problems whereby one wishes to satisfy as much demand as possible, without creating conflicts often requires the set packing format. Finally, when every customer must be served by exactly one server, the problem takes on the set partitioning format. Commonly cited problems having this structure include the *crew-scheduling problem*, where every flight leg of an airline must be scheduled by exactly one cockpit crew; another is the *political districting problem*, whereby regions must be divided into voting districts such that every citizen is assigned to exactly one district. See the survey [4] which contains a bibliography on applications.

Recently (as of 2000), reformulating them as either set-covering problems or set-partitioning problems having an extraordinary number of variables has

solved a variety of difficult problems. Because, for even small instances of the problem, the problem size cannot be explicitly solved, techniques known as column-generation, which began with the seminal work [16] on the *cutting-stock problem*, are employed. An overview of such transformation methods can be found in [5]. For specific implementations to the *vehicle routing problem*, see [11], for the bandwidth packing problem, see [27], for the generalized assignment problem see [29] and for alternative column-generation strategies for solving the cutting-stock problems, see [5].

J. Bramel and D. Simchi-Levi [7] have shown that the set-partitioning formulation for the vehicle routing problem with time windows is a tight formulation, i. e. the relative gap between the fractional linear programming solution and the global integer solution is small. Similar results are obtained for the bin-packing problem [9] and for machine scheduling [8].

Solution Approaches

Once the problem has been formulated as a set-covering, set-packing or set-partitioning problem, the search for an optimal (or near-optimal) solution to this *NP*-hard 0–1 linear programming problem remains. Most solution approaches start by considering the *linear programming relaxation* (LP relaxation) of the respective problem. If the matrix A is a perfect zero-one matrix, see [23], then the LP relaxation of both the set packing and the set partitioning problem have a zero-one optimal solution for all choices of the objective function. Likewise, if the matrix A is an ideal matrix, see [26], then the same holds true for the set covering and the set-partitioning problem. Problems arising in practice need not, however, have perfect or ideal matrices. Nevertheless, it has been observed in computational practice that as long as the problems to be solved are relatively small, linear programming (or linear programming coupled with branch and bound) is likely to provide integer solutions quickly. However, as the problem size increases, the nonintegrality of the linear programming solution increases dramatically as does the length and size of the branching tree. It is for these larger instances of the problem that approximation techniques, *reformulation* and exact procedures have been developed that exploit the underlying structure of the problem.

Reformulation of the Linear Description of the Problem

The natural structure of packing, covering and partitioning approaches provides opportunities to automatically remove any unnecessary rows or columns, and to remove any variables that cannot exist in any optimal solution. Checks for inconsistencies among the constraints are also performed. Reformulation procedures for the set covering problem have been well known for a long time [15] but had not been implemented into a special-purpose code for solving very large scale problems until 1993 [19].

Heuristics for the Set Partitioning and Covering Problems

Virtually every heuristic approach for solving general integer programming problems has been applied to the set-covering, packing and partitioning problems. The set covering and packing formulations naturally lend themselves to greedy starts (i. e. an approach that at every iteration myopically chooses the next best step without regard for its implications on future moves), see e. g. [14]. Interchange approaches have also been applied here; a swap of one or more columns is taken whenever such a swap improves the objective function value. Newer heuristic approaches such as genetic algorithms (cf. also ► Genetic algorithms), probabilistic search [13], simulated annealing (cf. also ► Simulated annealing methods in protein folding) [5], and neural networks (cf. also ► Neural networks for combinatorial optimization) [1] have each been tried. Unfortunately, there has not been a comparative testing across such methods to determine under what circumstances a specific method might perform best. J.E. Beasley [6] maintains an extensive test set of problem instances for these important problems.

In addition, one can embed heuristics within exact algorithms so that one iteratively tightens the upper bound at the same time that one is attempting to get a tight approximation to the lower bound for the problem. See [19] for a description of a linear-programming based heuristic for the set-partitioning problem with side constraints, and [2] for heuristics based on using Lagrangian relaxation (cf. also ► Integer programming: Lagrangian relaxation) embedded within branch and bound to solve the set covering problem.

Exact Solution Approaches to the Set Covering, Packing and Partitioning Problems

Exact approaches to solving set partitioning, covering and packing problems require algorithms that generate both good lower and upper bounds on the true minimum value of the problem instance. One can use any of the heuristics mentioned above to obtain a good upper bound to these problems. One should note, however, that the set covering and packing problems are easier problems for heuristic search because for these problems, it is, in general, easy to find feasible solutions. The set-partitioning problem may create unique concerns for some of these algorithms specifically because each row must be covered exactly once.

In general, the lower bound on the optimal solution value is obtained by solving a relaxation of the optimization problem. That is, one solves another optimization problem whose set of feasible solutions properly contains all feasible solutions of the original problem and whose objective function value is less than or equal to the true objective function value for points feasible to the original problem. Thus, we replace the 'true' problem by one with a larger feasible region that is more easily solved. There are two standard relaxations for covering, packing and partitioning problems: Lagrangian relaxation (where the feasible set is usually required to maintain 0–1 feasibility, but many if not all of the constraints are moved to the objective function with a penalty term) and the linear programming relaxation (where only the integrality constraints are relaxed and the objective function remains the original function). For the set-covering problem, in [12] a Lagrangian formulation and *subgradient optimization* is used. In [3] various Lagrangian relaxations are tested, including some that incorporated cuts within the formulation and kept a disjoint set of the original linear constraints unrelaxed. In [2], dual and primal heuristics, recursive variable fixing and subgradient optimization are embedded within a *branch and bound* tree search.

An alternative approach to solving set partitioning, packing and set covering problems is *branch and cut*. This method begins by solving the linear programming relaxation to the problem and then tightening the formulation by adding new linear inequalities to the constraint set.

Specifically, it requires finding linear inequalities that are violated by a given relaxation but are satisfied by all feasible zero-one points. The most successful *cutting plane approaches* are based on *polyhedral theory*, that is they replace the constraint set of an integer programming problem by a convexification of the feasible zero-one points and extreme rays of the problem. Some of the polyhedral cuts useful for set-partitioning problems are clique inequalities, odd cycles, and the complements of odd cycles in the intersection graph associated with the matrix A. For a complete description of how such cuts are embedded into a tree search structure that also uses heuristics, and reformulation and variable fixing techniques, see [19].

For details on polyhedral structure see [10,22,24,25] and [28]. Currently, the polyhedral description of these problems is incomplete. As our understanding of the mathematical structure of the set partitioning, packing and covering polytopes improves, and with the continuing advancement in computer technology, it is likely that many difficult and important problems will be solved by being able to solve larger and larger set partitioning problems to proven optimality.

The Future

The recent interest (as of 2000) in reformulating hard, important scheduling problems into set partitioning problems via column generation has reinvigorated research into both linear and integer programming solution techniques. The linear programming relaxations of these very large set partitioning problems yield highly degenerate problems for the primal simplex method to solve. This degeneracy resulted in the revisiting of both primal and dual steepest edge methods, see [17]. The fact that the *column generation approaches* require the solution to many set-partitioning problems requires that we better understand the structure of these problems. More research into polyhedral structure, time-staged network optimization (for the subproblem solutions), and careful attention to computer implementation details are likely to yield successes to problems (such as machine-shop scheduling) that up until now have not allowed exact solution approaches.

See also

► Branch and Price: Integer Programming with Column Generation

- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Graph Coloring
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem

References

1. Aourid M, Kaminska B (1994) Neural networks for the set covering problem: An application to the test vector compaction. IEEE Internat Conf Neural Networks: Conf Proc 7:4645–4649
2. Balas E, Carrera MC (1996) A dynamic subgradient-based branch-and-bound procedure for set covering. Oper Res 44:875–890
3. Balas E, Ho A (1980) Set covering algorithms using cutting planes, heuristics, and subgradient optimization: A computational study. Math Program 12:37–60
4. Balas E, Padberg MW (1976) Set partitioning: A survey. SIAM Rev 18:710–760
5. Barnhart C, Johnson ED, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch and price column generation for solving hugh integer programs. Oper Res 46:316–329
6. Beasley JE (1990) A Lagrangian heuristic for set-covering problems. Naval Res Logist 37:151–164
7. Bramel J, Simchi-Levi D (1998) On the effectiveness of set covering formulations for the vehicle routing problem with time windows. Techn. Report Dept. Industr. Engin. and Management Sci. Northwestern Univ.
8. Chan LMA, Muriel A, Simchi-Levi D (1998) Parallel machine scheduling, linear programming, and parameter list scheduling heuristics. Techn. Report Dept. Industr. Engin. and Management Sci. Northwestern Univ.
9. Chan LMA, Simchi-Levi D, Bramel J (1998) Worst-case analysis, linear programming and the bin-packing problem. Techn. Report Dept. Industr. Engin. and Management Sci. Northwestern Univ.
10. Cornuejols G, Sassano A (1989) On the 0-1 facets of the set covering polytope. Math Program 43:45–56
11. Desrochers M, Soumis F (1989) A column generation approach to the urban transit crew scheduling problem. Transport Sci 23:1–13
12. Etcheberry J (1977) The set covering problem: A new implicit enumeration algorithm. Oper Res 25:760–772
13. Feo A, Mauricio GC, Resende A (1989) Probabilistic heuristic for a computationally difficult set covering problem. Oper Res Lett 8:67–71
14. Fisher M, Wolsey L (1982) On the greedy heuristic for covering and packing problems. SIAM J Alg Discrete Meth 3:584–591
15. Garfinkel RS, Nemhauser GL (1972) Integer programming. Wiley, New York, pp 302–305
16. Gilmore PC, Gomory RE (1961) A linear programming approach to the cutting stock problem. Oper Res 9:849–859
17. Goldfarb D, Reid JK (1977) A practicable steepest-edge algorithm. Math Program 12:361–371
18. Hall N, Hochbaum D (1989) A fast approximation algorithm for the multicovering problem. Discrete Appl Math 15:35–40
19. Hoffman K, Padberg MW (1993) Solving airline crew scheduling problems by branch and cut. Managem Sci 39:657–682
20. Huang WC, Kao CY, Hong JT (1994) A genetic algorithm for set covering problems. In: IEEE Internat. Conf. Genetic Algorithms: Proc., pp 569–574
21. Jacobs L, Brusco M (1995) Note: A local-search heuristic for large set-covering problems. Naval Res Logist 42:1129–1140
22. Padberg MW (1973) On the facial structure of set packing polyhedra. Math Program 5:199–215
23. Padberg MW (1974) Perfect zero-one matrices. Math Program 6:180–196
24. Padberg MW (1975) On the complexity of the set packing polyhedra. Ann Discret Math 1:421–434
25. Padberg MW (1979) Covering, packing and knapsack problems. Ann Discret Math 4:265–287
26. Padberg MW (1993) Lehman's forbidden minor characterization of ideal 0-1 matrices. Discret Math 111:409–410
27. Parker M, Ryan J (1994) A column generation algorithm for bandwidth packing. Telecommunication Systems 2:185–196
28. Sassano A (1989) On the facial structure of the set covering polytope. Math Program 44:181–202
29. Savelsbergh MWP (1997) A branch-and-price algorithm for the generalized assignment problem. Oper Res 45:831–841

Set-valued Optimization

JOHANNES JAHN
University Erlangen–Nürnberg, Erlangen, Germany

MSC2000: 90C48, 90C29, 49K27

Article Outline

Keywords
See also
References

Keywords

Set-valued analysis; Vector optimization

Nowadays (2000) *set-valued optimization* means set-valued analysis and its application to optimization, and it is an extension of continuous optimization to the set-valued case. In this research area one investigates optimization problems with constraints and/or an objective function described by set-valued maps, or investigations in set-valued analysis are applied to standard optimization problems. In the last decade there has been an increasing interest in set-valued optimization (e. g., see the special issue [7]).

General optimization problems with *set-valued constraints* or a *set-valued objective function* are closely related to problems in stochastic programming, fuzzy programming and optimal control. If the values of a given function vary in a specified region, this fact could be described using a membership function in the theory of fuzzy sets or using information on the distribution of the function values. In this general setting probability distributions or membership functions are not needed because only sets are considered.

Optimal control problems with differential inclusions belong to this class of set-valued optimization problems as well. Set-valued optimization seems to have the potential to become a bridge between different areas in optimization. And it is a substantial extension of standard optimization theory. Set-valued analysis is the most important tool for such an advancement in continuous optimization. And conversely, the development of set-valued analysis receives important impulses from optimization.

Set-valued optimization problems have been investigated by many authors for instance, there are papers on optimality conditions (e. g., [3,4,5,6,9,11,14,17,19,21,22]), duality theory (e. g., [8,20,23,24]) and related topics (e. g., [10,15,16,25]). For further developments see [7].

In the following let X, Y and Z be real linear spaces, let Y and Z be partially ordered by convex cones $C_Y \subset Y$ and $C_Z \subset Z$, respectively (then $\leq_{C_Y}$ and $\leq_{C_Z}$ denote the corresponding partial orderings), let $\widehat{S}$ be a nonempty subset of X, and let $F\colon\ \widehat{S} \to 2^Y$ and $G\colon\ \widehat{S} \to 2^Z$ be set-valued maps. Throughout this article it is generally assumed that the domain of a set-valued map equals its effective domain, i. e. for every element of the domain the image is a nonempty set.

Under these assumptions one considers the *set-valued optimization problem*

$$\text{(SVOP)} \qquad \begin{cases} \min & F(x) \\ \text{s.t.} & G(x) \cap (-C_Z) \neq \emptyset, \\ & x \in \widehat{S}. \end{cases}$$

For simplicity let

$$S := \left\{ x \in \widehat{S}\colon\ G(x) \cap (-C_Z) \neq \emptyset \right\}$$

denote the feasible set of this problem which is assumed to be nonempty. If G is single-valued, the constraint in (SVOP) reduces to $G(x) \in -C_Z$ or $G(x) \leq_{C_Z} 0_Z$ generalizing equality and inequality constraints. If, in addition, F is single-valued, then the problem (SVOP) is a general vector optimization problem.

It is also possible to use a constraint of the form $G(x) \subset -C_Z$. But with a simple transformation (see [20]) this type of a constraint can be transformed to the type of the constraint in problem (SVOP). This transformation has the drawback that convexity and continuity properties of the map may be lost.

As a simple example for problem (SVOP) consider the case that the objective function of a standard optimization problem is not known explicitly. But one assumes that for every feasible x a lower bound $f(x)$ and an upper bound $g(x)$ are given. In this case one can replace the objective function by a set-valued map F with

$$F(x) := [f(x), g(x)] \quad \text{for all } x \in S.$$

In practice it turns out that set-valued optimization makes only sense for a set-valued map F whose lower

bound cannot be described by a function f (because the minimization of F is in a certain sense equivalent to the minimization of f).

Now the actual minimality notion used in set-valued optimization is presented.

Definition 1 A pair $(\overline{x}, \overline{y})$ with $\overline{x} \in S$ and $\overline{y} \in F(\overline{x})$ is called a *minimizer* of problem (SVOP), if $\overline{y}$ is a minimal element of the set $F(S) := \cup_{x \in S} F(x)$, i. e.

$$(\{\overline{y}\} - C_Y) \cap F(S) \subset \{\overline{y}\} + C_Y.$$

It is known from *vector optimization* that the so-called weak minimality notion is the appropriate concept for the formulation of necessary and sufficient optimality conditions. This fact also holds for the set-valued case.

Definition 2 If in addition $\operatorname{int}(C_Y) \neq \emptyset$, then a pair $(\overline{x}, \overline{y})$ with $\overline{x} \in S$ and $\overline{y} \in F(\overline{x})$ is called a *weak minimizer* of problem (SVOP), if $\overline{y}$ is a weakly minimal element of the set $F(S)$, i. e.

$$(\{\overline{y}\} - \operatorname{int}(C_Y)) \cap F(S) = \emptyset.$$

In order to obtain optimality conditions generalizing the known classical conditions a suitable differentiability notion is now introduced.

Definition 3 [14] In addition, let X and Y be real normed spaces, and let a pair $(\overline{x}, \overline{y})$ with $\overline{x} \in S$ and $\overline{y} \in F(\overline{x})$ be given. A single-valued map $DF(\overline{x}, \overline{y})\colon X \to Y$ whose epigraph equals the contingent cone (e. g., see [13]) to the epigraph of F at $(\overline{x}, \overline{y})$, i. e.

$$\operatorname{epi}(DF(\overline{x}, \overline{y})) = T(\operatorname{epi}(F), (\overline{x}, \overline{y}))$$

is called *contingent epiderivative* of F at $(\overline{x}, \overline{y})$.

Here the *epigraph* of F is defined as

$$\operatorname{epi}(F) := \{(x, y) \in X \times Y\colon\ x \in S,\ y \in F(x) + C_Y\}.$$

The contingent epiderivative is a possible generalization of the well-known directional derivative in the single-valued case. Under convexity assumptions the contingent epiderivative is a sublinear map. In set-valued optimization convex maps are introduced as follows.

Definition 4 The set-valued map $F\colon S \to 2^Y$ is called C_Y*-convex*, if for all $x_1, x_2 \in S$ and $\lambda \in [0, 1]$

$$\lambda F(x_1) + (1 - \lambda)F(x_2) \subset F(\lambda x_1 + (1 - \lambda)x_2) + C_Y.$$

Using the concept of the contingent epiderivative it is also possible to introduce subgradients of a set-valued map.

Definition 5 [2] Let the contingent epiderivative $DF(\overline{x}, \overline{y})$ of F at $(\overline{x}, \overline{y})$ exist with $\overline{x} \in S$ and $\overline{y} \in F(\overline{x})$. A linear map $L\colon X \to Y$ with

$$L(x) \leq_{C_Y} DF(\overline{x}, \overline{y})(x), \quad \text{for all } x \in X,$$

is called a *subgradient* of F at $(\overline{x}, \overline{y})$. The set $\partial F(\overline{x}, \overline{y})$ of all subgradients L of F at $(\overline{x}, \overline{y})$ is called *subdifferential* of F at $(\overline{x}, \overline{y})$.

Theorem 6 [2] *In addition to the assumptions, let X and Y be real normed spaces, let $S = X$, let C_Y be pointed, let Y be order complete, let F be C_Y-convex, and let the contingent epiderivative $DF(\overline{x}, \overline{y})$ of F at $(\overline{x}, \overline{y})$ exist with $\overline{x} \in S$ and $\overline{y} \in F(\overline{x})$. Then the subdifferential $\partial F(\overline{x}, \overline{y})$ is nonempty.*

Next, a complete characterization of weak minimizers in convex set-valued optimization is presented.

Theorem 7 [14] *In addition to the assumptions, let X and Y be real normed spaces, let S be a convex set, let $\operatorname{int}(C_Y) \neq \emptyset$, let F be C_Y-convex, and let the contingent epiderivative $DF(\overline{x}, \overline{y})$ exist at $\overline{x} \in S$ and $\overline{y} \in F(\overline{x})$. The pair $(\overline{x}, \overline{y})$ is a weak minimizer of problem (SVOP) if and only if*

$$DF(\overline{x}, \overline{y})(x - \overline{x}) \notin -\operatorname{int}(C_Y) \quad \textit{for all } x \in S.$$

A corresponding result can be shown for the constrained case where G describes an inequality constraint as in problem (SVOP). This optimality condition extends the *Lagrange multiplier rule* to the set-valued case.

Theorem 8 [11] *In addition to the assumptions, let X and Y be real normed spaces, let $\widehat{S}$ be a convex set, let $\operatorname{int}(C_Y) \neq \emptyset$, let F be C_Y-convex, let G be C_Z-convex, let $\overline{x} \in S$ and $\overline{y} \in F(\overline{x})$ be given, and let the contingent epiderivative of (F, G) at $(\overline{x}, (\overline{y}, \overline{z}))$ exist for an arbitrary*

$\overline{z} \in G(\overline{x}) \cap (-C_Z)$. *Moreover, let the regularity assumption*

$$\{z\colon\ (y,z) \in D(F,G)(\overline{x}(\overline{y},\overline{z}))(\mathrm{cone}(S-\{\overline{x}\}))\} + \mathrm{cone}(C_Z + \{\overline{z}\}) = Z$$

be satisfied (cone$(\cdots)$ denotes the cone generated by a set [13]). Then $(\overline{x},\overline{y})$ is a weak minimizer of the problem (SVOP) if and only if there are continuous linear functionals $t \in C_{Y^*} \setminus \{0_{Y^*}\}$ and $u \in C_{Z^*}$ with

$$t(y) + u(z) \geq 0$$

for all $(y,z) = D(F,G)(\overline{x}(\overline{y},\overline{z}))(x-\overline{x})$ with $x \in \widehat{S}$, and

$$u(\overline{z}) = 0.$$

General results on set-valued analysis can be found in [1]; while [12] and [18] present the theoretical background of vector optimization.

See also

- ► Generalized Monotone Multivalued Maps
- ► Generalized Monotone Single Valued Maps

References

1. Aubin J-P, Frankowska H (1990) Set-valued analysis. Birkhäuser, Basel
2. Baier J, Jahn J (1999) On subdifferentials of set-valued maps. J Optim Th Appl 100:233–240
3. Borwein JM (1977) Multivalued convexity and optimization: A unified approach to inequality and equality constraints. Math Program 13:183–199
4. Borwein JM (1981) A Lagrange multiplier theorem and a sandwich theorem for convex relations. Math Scand 48:189–204
5. Borwein JM (1983) Adjoint process duality. Math Oper Res 8:403–434
6. Chen GY, Jahn J (1998) Optimality conditions for set-valued optimization problems. Math Meth Oper Res 48:187–200
7. Chen GY, Jahn J (1998) Special issue on 'Set-valued optimization'. Math Meth Oper Res 48(2):151–285
8. Corley HW (1987) Existence and Lagrangian duality for maximizations of set-valued functions. J Optim Th Appl 54:489–501
9. Corley HW (1988) Optimality conditions for maximizations of set-valued functions. J Optim Th Appl 58:1–10
10. Dutta J, Vetrivel V (1999) Theorems of the alternative in set-valued optimization. Manuscript, India
11. Götz A, Jahn J (1999) The Lagrange multiplier rule in set-valued optimization. SIAM J Optim 10:331–344
12. Jahn J (1986) Mathematical vector optimization in partially ordered linear spaces. P. Lang, Frankfurt am Main
13. Jahn J (1996) Introduction to the theory of nonlinear optimization. Springer, Berlin
14. Jahn J, Rauh R (1997) Contingent epiderivatives and set-valued optimization. Math Meth Oper Res 46:193–211
15. Klose J (1992) Sensitivity analysis using the tangent derivative. Numer Funct Anal Optim 13:143–153
16. Kuroiwa D (1998) Natural criteria of set-valued optimization. Manuscript, Shimane Univ., Japan
17. Li Z (1999) A theorem of the alternative and its application to the optimization of set-valued maps. J Optim Th Appl 100:365–375
18. Luc DT (1989) Theory of vector optimization. Springer, Berlin
19. Luc DT (1991) Contingent derivatives of set-valued maps and applications to vector optimization. Math Program 50:99–111
20. Luc DT, Jahn J (1992) Axiomatic approach to duality in optimization. Numer Funct Anal Optim 13:305–326
21. Luc DT, Malivert C (1992) Invex optimization problems. Bull Austral Math Soc 46:47–66
22. Oettli W (1980) Optimality conditions for programming problems involving multivalued mappings. Manuscript, Univ. Mannheim, Germany
23. Postolică V (1986) Vectorial optimization programs with multifunctions and duality. Ann Sci Math Québec 10:85–102
24. Song W (1998) A generalization of Fenchel duality in set-valued vector optimization. Math Meth Oper Res 48:259–272
25. Zhuang D (1989) Regularity and minimality properties of set-valued structures in optimization. Diss. Dalhousie Univ., Halifax

Shape Optimization

SO

JAROSLAV HASLINGER
Charles University, Prague, Czech Republic

MSC2000: 49J20, 49J52

Article Outline

Keywords
See also
References

Keywords

Shape optimization; Optimal shape design; Shape sensitivity analysis

Shape optimization is a part of the larger field called structural optimization (cf. ► Structural optimization; ► Structural optimization: History). Structural optimization can be characterized as an applied branch of the optimal control theory in which the control variable is related to the geometry of structures. It can be divided into 3 parts:

i) *sizing optimization* (optimization of a typical size: a thickness optimization of beams, plates, etc.);
ii) *shape optimization* (optimization of a shape of structures, keeping the topology of an initial design);
iii) *topology optimization* (it makes possible to change the topology of an initial design).

An abstract formulation of a large class of optimal shape design problems reads as follows:

$$(\mathrm{P}) \quad \begin{cases} \text{Find} & \Omega^* \in \mathcal{O} \\ \text{s.t.} & I(\Omega^*, u(\Omega^*)) \le I(\Omega, u(\Omega)), \\ & \quad \forall \Omega \in \mathcal{O}. \end{cases}$$

Here $\mathcal{O}$ is a family of *admissible domains*, $u(\Omega)$ is asolution of a *state problem* $(\mathcal{P}(\Omega))$, describing the behavior of a structure, represented by a domain Ω and I is a *cost functional*. The state problem is typically given by partial differential equations (PDEs) or by variational inequalities (VIs). The mathematical analysis of (P) includes:

j) the study of existence of solutions to (P);
jj) discretization of (P) and the convergence analysis;
jjj) sensitivity analysis.

In order to guarantee the existence of solutions to (P) we make several assumptions (note that there is no uniqueness of the solution, in general). First we introduce the concept of a convergence in $\mathcal{O}$, i. e. if $\{\Omega_n\}$, $\Omega_n \in \mathcal{O}$, is a sequence, we specify the meaning of saying: 'Ω_n tends ω' to $\Omega_n \overset{\mathcal{O}}{\to} \Omega$. With any $\Omega \in \mathcal{O}$, a Hilbert space $V(\Omega)$ of functions defined in Ω will be associated (space of functions with finite energy). If $\Omega_n \overset{\mathcal{O}}{\to} \Omega$ and $y_n \in V(\Omega_n)$, $y \in V(\Omega)$, then one has to define the convergence '$y_n \to y$' (note that the domain of definition of functions varies). Finally, let u: $\Omega \to u(\Omega) \in V(\Omega)$, $\Omega \in \mathcal{O}$ be a *state relation* (PDE, VI, etc.) with $u(\Omega)$ being the solution of $(\mathcal{P}(\Omega))$ and $G = \{(\Omega, u(\Omega)) : \Omega \in \mathcal{O}\}$ be its graph. We suppose that:

1) G is *compact* in the following sense: if $\{\Omega_n\}$, $\Omega_n \in \mathcal{O}$, is an arbitrary sequence, then there exist a subsequence $\{(\Omega_{n_k}, u(\Omega_{n_k}))\}$ and an element $(\Omega, u(\Omega)) \in G$ such that

$$\Omega_{n_k} \overset{\mathcal{O}}{\to} \Omega, \quad u(\Omega_{n_k}) \to u(\Omega), \quad k \to \infty,$$

in the specified sense;
2) I is *lower semicontinuous*: if $\{\Omega_n\}$, $\{v_n\}$, where $\Omega_n \in \mathcal{O}$, $v_n \in V(\Omega_n)$ are arbitrary sequences such that $\Omega_n \overset{\mathcal{O}}{\to} \Omega$, '$v_n \to v$' with $\Omega \in \mathcal{O}$, $v \in V(\Omega)$ then

$$\liminf_{n\to\infty} I(\Omega_n, v_n) \ge I(\Omega, v).$$

It holds (see [4]):

Theorem 1 *Let 1)–2) be satisfied. Then (P) has a solution.*

Example 2 Let $\mathcal{O}$ be a family of bounded domains satisfying the *uniform cone property* (see [1]), let $(\mathcal{P}(\Omega))$ be given by the Neumann problem

$$(\mathcal{P}(\Omega)) \quad \begin{cases} -\Delta u + u(\Omega) = f & \text{in } \Omega \in \mathcal{O} \\ \frac{\partial u(\Omega)}{\partial \nu} = 0 & \text{on } \partial\Omega, \end{cases}$$

and $I(\Omega, y) = \int_\Omega |y - z_d|^2 \, dx$, where $f, z_d \in L^2_{\mathrm{loc}}(\mathbf{R}^m)$ are given functions. We set $V(\Omega) \equiv H^1(\Omega)$, i. e. $V(\Omega)$ is the standard Sobolev space of functions defined in Ω, whose derivatives up to the order one are square integrable in Ω, i. e. elements of $L^2(\Omega)$ (see [7]).The weak formulation of $(\mathcal{P}(\Omega))$ is given by:

$$\begin{cases} \text{Find} & u(\Omega) \in H^1(\Omega) \\ \text{s.t.} & \displaystyle\int_\Omega (\operatorname{grad} u(\Omega) \cdot \operatorname{grad} \phi + u\phi) \, dx \\ & \quad = \displaystyle\int_\Omega f\phi \, dx, \quad \forall \phi \in H^1(\Omega). \end{cases}$$

We define:

- $\Omega_n \overset{\mathcal{O}}{\to} \Omega$ if and only if $\chi(\Omega_n) \to \chi(\Omega)$ in $L^2(\mathbf{R}^m)$;
- '$y_n \to y$' if and only if $\chi(\Omega_n) \to \chi(\Omega)$ (weakly) in $L^2(\mathbb{R}^m)$.

Here, $\chi(\Omega)$ is the characteristic function of Ω, $\widehat{\Omega}$ is a domain containing all $\Omega \in \mathcal{O}$ and the symbol '$\sim$' stands for the *uniform extension* of functions from their domain of definition to $\widehat{\Omega}$ (see [1]). One can verify 1)–

2) for the convergences introduced above (see [1,4,8]). Thus, the corresponding SO problem has a solution.

To define an *approximation* of (P) we replace $\mathcal{O}$ by a system '$\mathcal{O}_h$', $h \to 0+$. Any $\mathcal{O}_h$ contains domains Ω_h whose shapes are described by a finite number of parameters. This number depends on the discretization parameter h (the boundary of Ω_h is piecewise linear, a spline function, etc.). In what follows we shall suppose that $\mathcal{O}_h \subset \mathcal{O}$ for any $h > 0$. With any $\Omega_h \in \mathcal{O}_h$, a finite-dimensional space $V_h(\Omega_h)$ will be associated (a finite element space, e. g.). The state problem $(\mathcal{P}(\Omega))$ will be replaced by its suitable discretization $(\mathcal{P}(\Omega_h))_h$ (by using the Ritz–Galerkin method, e. g.). Finally, the cost functional I may be approximated (I_h), as well. The *approximation* of (P) reads as follows:

$$(\mathrm{P})_h \begin{cases} \text{Find} & \Omega_h^* \in \mathcal{O}_h \\ \text{s.t.} & I_h(\Omega_h^*, u_h(\Omega_h^*)) \le I_h(\Omega_h, u_h(\Omega_h)), \\ & \forall \Omega_h \in \mathcal{O}_h. \end{cases}$$

Here, $u_h(\Omega_h)$ is a solutionof $(\mathcal{P}(\Omega_h)_h)$. Problem $(\mathrm{P})_h$ expressed in the algebraic form leads to a *nonlinear mathematical programming problem*, in general.

To establish a relation between solutions to (P) and $(\mathrm{P})_h$, when $h \to 0+$, the following assumptions are needed:

3) for any $\Omega \in \mathcal{O}$ there exists a sequence $\{\Omega_h\}$, $\Omega_h \in \mathcal{O}_h$, such that

$$\Omega_h \overset{\mathcal{O}}{\to} \Omega, \quad h \to 0+;$$

4) for every sequence $\{(\Omega_h, u_h(\Omega_h))\}$, where $\Omega_h \in \mathcal{O}_h$ and $u_h(\Omega_h)$ solves $(\mathcal{P}(\Omega_h))_h$, there exist its subsequence $\{(\Omega_{h_k}, u_{h_k}(\Omega_{h_k}))\}$ and an element $(\Omega, u(\Omega)) \in G$ such that

$$\Omega_{h_k} \overset{\mathcal{O}}{\to} \Omega,$$

$$'u_{h_k}(\Omega_{h_k}) \to u(\Omega)', k \to \infty;$$

5) if $\Omega_h \overset{\mathcal{O}}{\to} \Omega$ with $\Omega_h \in \mathcal{O}_h$, $\Omega \in \mathcal{O}$ and '$u_h(\Omega_h) \to u(\Omega)$', then

$$\lim_{h \to 0+} I_h(\Omega_h, u_h(\Omega_h)) = I(\Omega, u(\Omega)).$$

Then one can prove (see [4]):

Theorem 3 *Let 3)–5) be satisfied and let for any $h > 0$ there exists an optimal pair $(\Omega_h^*, u_h(\Omega_h^*))$. Then there exista subsequence $\{(\Omega_{h_k}^*, u_{h_k}(\Omega_{h_k^*}))'$ and an element $(\Omega^*, u(\Omega^*)) \in G$ such that*

$$6) \quad \begin{cases} \Omega_{h_k}^* \overset{\mathcal{O}}{\to} \Omega^* \\ 'u_{h_k}(\Omega_{h_k}^*) \to u(\Omega^*)', \quad k \to \infty, \end{cases}$$

and $(\Omega^*, u(\Omega^*))$ is an optimal pair for (P). Moreover any such cluster point $(\Omega^*, u(\Omega^*))$ of a sequence $\{(\Omega_h^*, u_h(\Omega_h^*))\}$ in the sense of 6) is an optimal pair for (P).

Example 4 We describe the approximation of (P) from Example 2, considered in $\mathbf{R}^2$. For any $h > 0$, the family $\mathcal{O}_h$ contains polygonal domains, being the piecewise linear approximations of $\Omega \in \mathcal{O}$ and such that

$$\partial\Omega_h = \bigcup_{i=1}^{n(h)} \overline{A_i A_{i+1}}$$

($A_{n(h)+1} \equiv A_1$) with the length $|\overline{A_i A_{i+1}}| \le h$ for any side. Let $\{\mathcal{T}(h, \Omega_h)\}$, $h \to 0+$ be a family of triangulations of $\overline{\Omega}_h$ (see [2]) satisfying:

7) any $\overline{A_i A_{i+1}}$ is the side of just one boundary triangle $T \in \mathcal{T}(h, \Omega_h)$;

8) the number of nodes in $\mathcal{T}(h, \Omega_h)$ is the same for any $\Omega_h \in \mathcal{O}_h$ (h being fixed) and the nodes have still the same neighbors;

9) the position of internal nodes of $\mathcal{T}(h, \Omega_h)$ *continuously* depends on variations of the boundary nodes A_i, $i = 1, \ldots, n(h)$;

10) the family $\{\mathcal{T}(h, \Omega_h)\}$ satisfies the *uniform angle condition*:

$$\exists \theta_0 > 0 \quad \text{such that } \theta_T \ge \theta_0$$

holds for any triangle $T \in \mathcal{T}(h, \Omega_h)$, for any $\Omega_h \in \mathcal{O}_h$ and any $h > 0$, where θ_T is the minimal interior angle of T.

With any such $\mathcal{T}(h, \Omega_h)$, the space of allpiecewise linear functions $V_h(\Omega_h)$ will be associated (see [2]). Finally, $(\mathcal{P}(\Omega))$ is replaced by:

$$(\mathcal{P}(\Omega_h))_h \begin{cases} \text{Find} & u_h(\Omega_h) \in V_h(\Omega_h) \\ \text{s.t.} & \int_{\Omega_h} (\operatorname{grad} u_h(\Omega_h) \cdot \operatorname{grad} \phi_h \\ & \quad + u_h(\Omega_h)\phi_h)\, dx \\ & = \int_{\Omega_h} f\phi_h\, dx, \\ & \forall \phi_h \in V_h(\Omega_h). \end{cases}$$

It can be shown (see [4]) that 3)–5) are satisfied, provided that 7)–10) hold. Thus (P) and $(P)_h$ are close on subsequences as follows from Theorem 3.

Shape sensitivity analysis is a specific field of SO, analyzing the differentiability of the solution of state problems with respect to shape variations. There are several concepts of the shape differential calculus: the *method of mappings* [6], the *material derivative approach* [9] and the *boundary variation technique* [8]. Higher order derivatives in SO and their application are studied in [3,5]. On the contrary, if the state problem is given by VI, then the differentiability of the mapping: $\Omega \to u(\Omega)$ is weakened due to the fact that such mapping is only *Lipschitz continuous* (see [9]). Thus the resulting minimization problem is nonsmooth, in general. Such a type of problems can be realized or by using methods of *nonsmooth optimization* or by *regularizing* the state problem(see [4]). The use of global optimization methods in SO based on function evaluations only combined with the so-called *fictitious domain methods* is studied in [4].

See also

► Topological Derivative in Shape Optimization

References

1. Chenais D (1975) On the existence of a solution in a domain identification problem. J Math Anal Appl 52:189–289
2. Ciarlet PG (1978) The finite element method for elliptic problems. North-Holland, Amsterdam
3. Guillaume P, Masmoudi M (1994) Computation of higher order derivatives in optimal shape design. Numer Math 67:231–250
4. Haslinger J, Neittaanmäaki P (1996) Finite element approximation for optimal shape, material and topology design. Wiley, New York
5. Masmoudi M, Guillaume P, Broudiscou C (1995) Application of automatic differentiation to optimal shape design. Adv Structural Optim 25:413–446
6. Murat F, Simon J (1976) Sur le controle par un domaine géométrique. Publ. Lab. Anal. Numér. Univ. Paris 6,).
7. Necas J (1967) Les méthodes directes en théorie des equations elliptiques. Academia, Prague
8. Pironneau O (1984) Optimal shape design for elliptic systems. Springer, Berlin
9. Sokolowski J, Zolesio JP (1992) Introduction to shape optimization. Springer, Berlin

Shape Reconstruction Methods for Nonconvex Feasibility Analysis

Ipsita Banerjee, Marianthi Ierapetritou
Department of Chemical and Biochemical Engineering, Rutgers University, Piscataway, USA

MSC2000: 90-08, 90C26, 90C31

Article Outline

Introduction
Definition
α-Shape Approach
Selection of α
Formulation
Feasibility Analysis Using α Shape
Sampling Technique
Cases
Process Operation Example
Conclusions
References

Introduction

Uncertainties in chemical plants appear for a variety of reasons. There are internal reasons, such as fluctuations of values of reaction constants and physical properties, and external reasons, such as quality and flow rates of feed streams. The need to account for uncertainty in various stages of plant operations has been identified as one of the most important problems in chemical plant design and operation [7,8,18].

There are two main problems associated with the consideration of uncertainty in decision making: the quantification of the feasibility and flexibility of a process design and the incorporation of uncertainty within a decision stage. The quantification of process feasibility is most commonly addressed by utilizing the feasibility function introduced by Swaney and Grossmann, which requires constraint satisfaction over a specified uncertainty space, whereas flexibility evaluation is associated with a quantitative measure of the feasible space. Halemane and Grossmann [10] proposed a feasibility measure for a given design based on the worst points for feasible operation, which can be mathematically formulated as a max-min-max optimization problem:

$$\chi(d) = \max_{\theta \in T} \min_{z} \max_{j \in J} f_j(d, z, \theta), \tag{1}$$

where T is the feasible space of θ described as $T = \{\theta|\theta^L \le \theta \le \theta^U\}$, where θ_L, θ_U are lower and upper bounds, respectively.

The general formulation for quantifying flexibility, known as the *flexibility index problem*, can be defined as the determination of maximum deviation Δ that a given design d can tolerate, such that every point θ in the uncertain parameter space ($T(\delta)$) is feasible [1]. A well-studied case is the hyperrectangle representation δ, $T(\delta) = \{\theta|\theta^N - \delta\Delta\theta^- \le \theta \le \theta^N + \delta\Delta\theta^+\}$, where $\Delta\theta^+$ and $\Delta\theta^-$ are the expected deviations of the uncertain parameters in the positive and negative directions and δ the deviation along a specified direction. Other descriptions of $T(\delta)$, such as the parametric hyperellipsoid, have also been investigated [15].

The flexibility index can be determined from the formulation proposed by Swaney and Grossmann [18] as:

$$\begin{aligned} & F = \max \delta \\ \text{Subject to } & \max_{\theta \in T(\delta)} \psi(\theta, d) \le 0 \\ & \delta \ge 0 \,. \end{aligned} \tag{2}$$

One approach to determining the flexibility index is by vertex enumeration, in which the maximum displacement is computed along each vertex direction. This scheme is based on the assumption that the critical points (θ^c) lie at the vertices of $T(\Delta^c)$, which holds only under certain convexity conditions. Other existing approaches to quantifying flexibility involves deterministic measures such as the resilience index (RI) proposed by Saboo et al. [16] and stochastic measures such as design reliability proposed by Kubic and Stein [12] and the stochastic flexibility index proposed by Pistikopoulos and Mazzuchi [14] and Straub and Grossmann [17]. Recently Ierapetritou and coworkers [6] introduced a new approach to quantifying process feasibility based on the description of the feasible region by an approximation of the convex hull. Their approach results in an accurate representation of process feasibility.

However, the convex hull approach is limited in its application to only convex and 1-D quasiconvex feasible regions, and its performance deteriorates in the presence of nonconvex constraints. This shortcoming can be overcome by utilizing surface reconstruction ideas to capture the accurate shape of the feasible region.

Definition

The main problem definition for surface reconstruction is, given a set of range points, to reconstruct a manifold that closely approximates the surface of the original model. The range data are a set of discrete points in three-dimensional space that have been sampled from the physical environment or can be obtained using laser scanners that generate data points on the surface of an object. The problem naturally arises in a variety of practical situations such as range scanning an object from multiple view points, recovery of biological shapes from two-dimensional slices, interactive surface sketching, etc. Surface reconstruction has extensive applications in the areas of automatic mesh generation and geometric modeling, molecular structure, and protein folding analysis.

The problem of feasibility analysis is analogous to that of surface reconstruction since the main effort of feasibility analysis lies in identifying and accurately estimating the boundary of the feasible region. In previous approaches this boundary is approximated by linear inequalities, either by incorporating a hyperrectangle [18] or by describing an approximation of the convex hull [6] inside the feasible space. These methods can have satisfactory performance in case of convex, connected feasible regions but will be inaccurate for cases of nonconvex or disjoint feasible regions. On the other hand, the surface reconstruction scheme can successfully describe both nonconvex and disjoint regions defining the bounding surface by piecewise linear functions. The present work proposes a feasibility analysis scheme based on surface reconstruction ideas, in particular, the α-shape methodology for surface reconstruction.

α-Shape Approach

Various approaches are described in the literature for determining the shape of a pattern class from sampled points. Many of these approaches are concerned with efficient construction of convex hulls for a set of points in a plane. Jarvis [11] was one of the first to consider the problem of computing shape as a generalization of the convex hull of a planar point set.

A mathematically rigorous definition of shape was later introduced by Edelsbrunner et al. [3] as a natural generalization of the convex hulls, which is referred to as α hull. The α hull of a point set is based on the notion of generalized discs in a plane. The family of α hulls includes the smallest enclosing circle, the set itself, and an essentially continuous set of enclosing regions in between these two extremes.

Edelsbrunner et al. [3] also define a combinatorial variant of the α hull called the α shape of a planar set, which can be viewed as the boundary of the α hull with curved edges replaced by straight ones. Conceptually, α shapes are a generalization of the convex hull of a point set $\mathbb{S}$, with α varying from 0 to ∞. The α shape of $\mathbb{S}$ is a polytope that is neither necessarily convex nor connected. For $\alpha = \infty$, the α shape is identical to the convex hull of $\mathbb{S}$. However, as α decreases, the α shape shrinks by gradually developing cavities. When α becomes small enough, the polytope disappears and reduces to the data set itself.

To provide an intuitive notion of the concept, Edelsbrunner describes the space R^3 to be filled with styrofoam and the point set $\mathbb{S}$ to be made of more solid material, such as rock. Now if a spherical eraser with radius α curves out the styrofoam at all positions where it does not enclose any of the sprinkled rocks (the point set $\mathbb{S}$), the resulting object that formed will be called an α hull. The surface of the object can be straightened by substituting straight edges for circular ones and triangles for spherical caps. The obtained object is the α shape of $\mathbb{S}$. It is a polytope in a fairly general sense: it can be concave and even disconnected; it can contain two-dimensional patches of triangles and one-dimensional strings of edges, and its components can be as small as single points. The parameter α controls the degree of details captured by the α shape.

It is possible to generalize all the concepts involved in the construction of α shape (i. e., α hulls, α complexes, Delaunay triangulation, Voronoi diagrams) to a finite set of points $\mathbb{S}$ in $\mathcal{R}^d$ for arbitrary dimension d. This generalization, combined with an extension to weighted points, is developed in Edelsbrunner [2]. However, the implementation details of the problem becomes progressively more complex with increasing dimension, and the worst-case complexity of the problem grows exponentially.

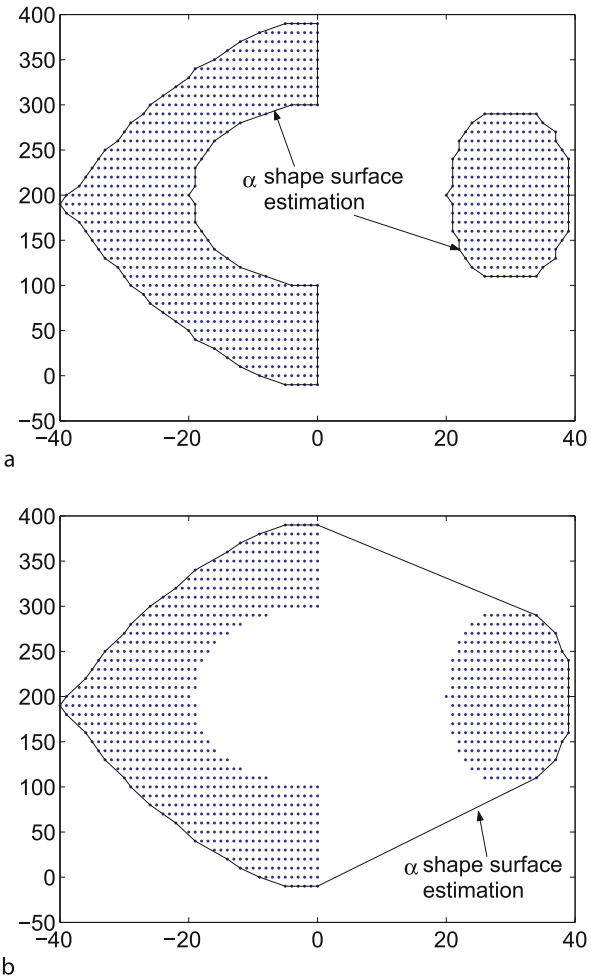
Selection of α

The computed α shape of a given set of sample points explicitly depends on the chosen value of α, which controls the level of detail of the constructed surface. Mandal et al. [13] present a systematic methodology for selecting the value of α in R^2. They visualize the problem of obtaining the shape of $\mathbb{S}$ as a set-estimation problem where an unknown set $\mathfrak{A} \in \mathfrak{B}$ is to be estimated on the basis of a finite number of points $X_1, X_2, \ldots, X_n \in \mathfrak{A}$. As n increases, $\mathbb{S}(n)$ will cover many parts of $\mathfrak{A}$, and hence the value of α for $\mathbb{S}(n)$ should depend on the sample size (n); thus α is a function of n. Additionally α should also be a function of the interpoint distance of the sampled n points of $\mathbb{S}(n)$. To account for the dependence on the interpoint distance, the authors have constructed the minimum spanning tree (MST) of the sampled data points. If l_n represents the sum of edge weights of the MST, where the edge weight is taken to be the Euclidean distance between the points, then the appropriate value of α for the construction of α shape is given by

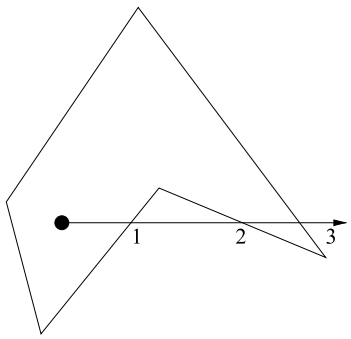
$$h_n = \sqrt{\frac{l_n}{n}}, \tag{3}$$

where n is the total number of sample points.

To illustrate the performance of α shape in capturing the shape of an object, a disjoint, nonconvex object is chosen, as illustrated in Fig. (1). The sampled points represent a 2-D object, which is the input to the α-shape construction code. The α shape identifies from the input data set points that lie on the boundary of the object. These points are joined by a line to describe the surface of the object. The above figure also illustrates the dependence of the captured shape on the chosen value of α. The α value estimated by performing the MST operation is 120, at which value the α shape was found to capture the nonconvex as well as the disjoint nature of the object. By further increasing the value of α the performance of α shape deteriorates, and at very high α the α shape forms a convex hull of the object (Fig. 1b). Hence the level of detail captured by the α shape strongly depends on the chosen value of α, and progressively decreasing the value of α will capture the shape more accurately.



Shape Reconstruction Methods for Nonconvex Feasibility Analysis, Figure 1
Performance of α shape for different α values: (a) 120 (b) 100,000



Shape Reconstruction Methods for Nonconvex Feasibility Analysis, Figure 2
Point-in-polygon test: an odd number of intersections means the point is inside; an even number of intersections means the point is outside

Formulation

Feasibility Analysis Using α Shape

The overall aim of feasibility analysis is the determination of the range of parameters over which a particular process is feasible. A formal definition of this problem is to obtain a mathematical description of the region in parameter space bounded by the process constraints. This region can be considered analogous to an object whose shape or surface can be estimated using the α-shape technique. The input to any surface-reconstruction algorithm needs to be a set of points representing the object whose surface needs to be determined. The steps involved in determining the feasible region using the surface reconstruction ideas are as follows:

- Generate sample data points to adequately represent the feasible region under consideration.
- Construct the α shape for the sampled data using the α estimate obtained from the MST of the data set.
- Join the identified boundary points to obtain a polygonal representation of the feasible region.

Having defined the surface or shape of the feasible region, the next step involves determining whether a particular point belongs to the feasible region. Since the feasible region has been approximated by a polygon, a simple way to check if a point is inside the polygon is by using one of the point-in-polygon tests [9]. One method to determine whether a point is inside a region is the *Jordan Curve Theorem*, which states that a point is inside a polygon if, for any semi-infinite ray from this point, there is an odd number of intersections of the ray with the polygon's edges (Fig. 2). Conversely, a point is outside a polygon if the ray intersects the polygon's edges an even number of times or does not intersect at all. Following this, whenever a parameter needs to be checked for feasibility in a polygon estimated feasible region, a semi-infinite ray is drawn from the point in any direction, and the number of intersections is noted, which determines whether or not the point is feasible.

Sampling Technique

The first step in the proposed approach is to have a good representation of the feasible region. Most of the com-

mon sampling techniques sample the parameter space based on the distribution of the uncertain parameter, which are considered to be uniform for the cases considered here for simplicity of presentation. Under this condition it leads to uniform sampling of the entire parameter space, irrespective of whether or not the sampled points are feasible. However, typically, the feasible region covers only a very restricted region of the entire parameter space. Hence sampling techniques covering the entire range of uncertain parameter prove to be inefficient, particularly when evaluation of the process constraints is an expensive operation. A new sampling technique is thus introduced here that takes advantage of the fact that typically a small section of the entire parameter space is feasible. The sampling problem is formulated as an optimization problem and is solved using a genetic algorithm (GA). The use of a GA as a solution procedure proves to be very efficient for this problem since the search scheme has the inherent property of concentrating around regions having good solutions, which is the feasible solution for the problem addressed here, thereby reducing expensive function evaluation.

The formulation of the sampling problem as an optimization problem is given by

$$\begin{aligned} &\max_{\theta} V_{\text{feas}} \\ \text{subject to } &(f_1)_\theta \le 0 \\ &(f_2)_\theta \le 0 \\ &\vdots \\ &(f_n)_\theta \le 0\,, \end{aligned} \tag{4}$$

where V_{feas} is the volume of the feasible region evaluated by constructing the α shape using the sampled feasible points. The optimization variables are the parameter values θ, which are sampled by the GA to optimize the objective, and $f_1, f_2 \ldots, f_n$ are the constraints of the feasibility problem evaluated at θ. However, in this formulation there is no optimal value of the variable θ that will maximize the volume, but we are interested in the entire sampled set of feasible θ values, using which the volume is evaluated by constructing an α shape over the entire set of feasible θ values. Since the objective is to maximize the volume of the feasible space, whenever a chosen value of θ satisfies the constraint functions, the volume is evaluated to update the objective function. When the value is not feasible, there is no need to reevaluate the volume since it will not change, but the fitness function is penalized by assigning it a small value. Solving this problem using a GA reduces the required number of function evaluations by minimizing the unnecessary evaluation of infeasible parameter space.

Cases

The idea of using surface reconstruction for the estimation of a feasible region is illustrated by a few case studies.

The feasible region is defined by the following sets of convex and nonconvex constraints:

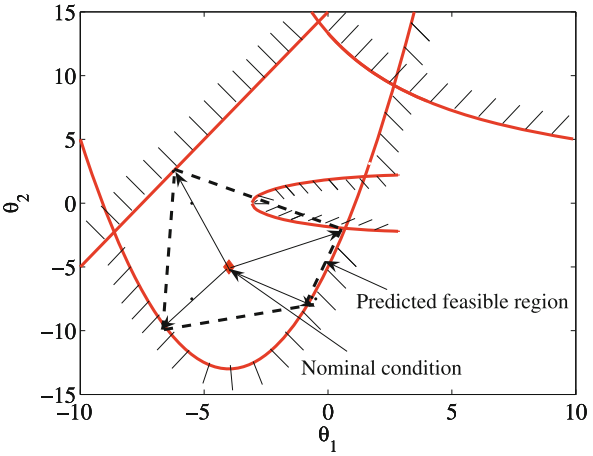
$$f_1 = \theta_2 - 2\theta_1 - 15 \le 0, \tag{5}$$

$$f_2 = \frac{\theta_1^2}{2} + 4\theta_1 - 5 - \theta_2 \le 0, \tag{6}$$

$$f_3 = \theta_2(6 + \theta_1) - 80 \le 0, \tag{7}$$

$$f_4 = 10 - \frac{(\theta_1 - 4)^2}{5} - 2\theta_2^2 \le 0. \tag{8}$$

Figure (3) illustrates the actual nature of the feasible region bounded by inequalities (5)–(8) and the convex hull approximation of the enclosed feasible region. The first step in the proposed scheme is to sample the feasi-

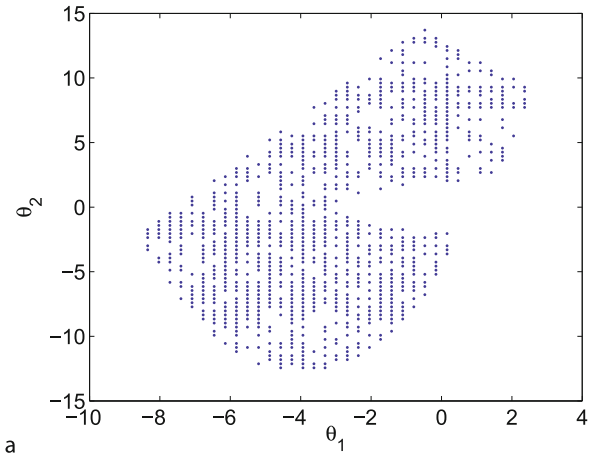


Shape Reconstruction Methods for Nonconvex Feasibility Analysis, Figure 3
Feasible region bounded by convex and nonconvex constraints and its estimation using convex hull

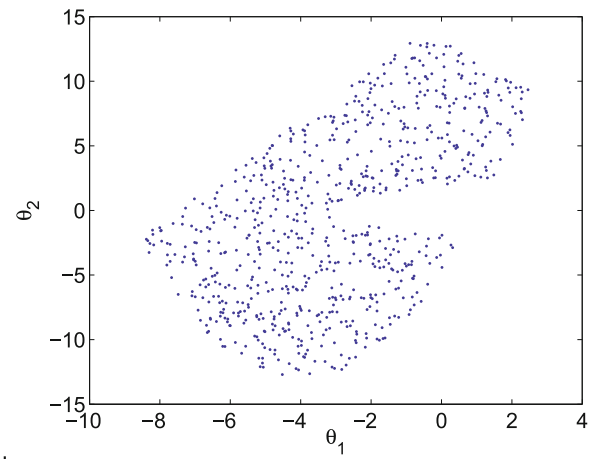
ble space efficiently. The optimization problem for sampling is given by

$$\begin{aligned} \max_{\theta_1,\theta_2} V_{\text{feas}}, \\ \theta_2 - 2\theta_1 - 15 \le 0, \\ \frac{\theta_1^2}{2} + 4\theta_1 - 5 - \theta_2 \le 0, \\ 10 - \frac{(\theta_1 - 4)^2}{5} - \frac{\theta_2^2}{0.5} \le 0, \\ \theta_2(6 + \theta_1) - 80 \le 0. \end{aligned} \tag{9}$$

Both uncertain parameters are considered to vary within the range of $(-20, 20)$. In order to solve the problem using a GA, the parameters θ_1 and θ_2 are encoded as bits, with 7 bits for each parameters, giving rise to a 14-bit chromosome. A population size of 20 is chosen for this problem following the guideline of Edwards et al. [4].



a



b

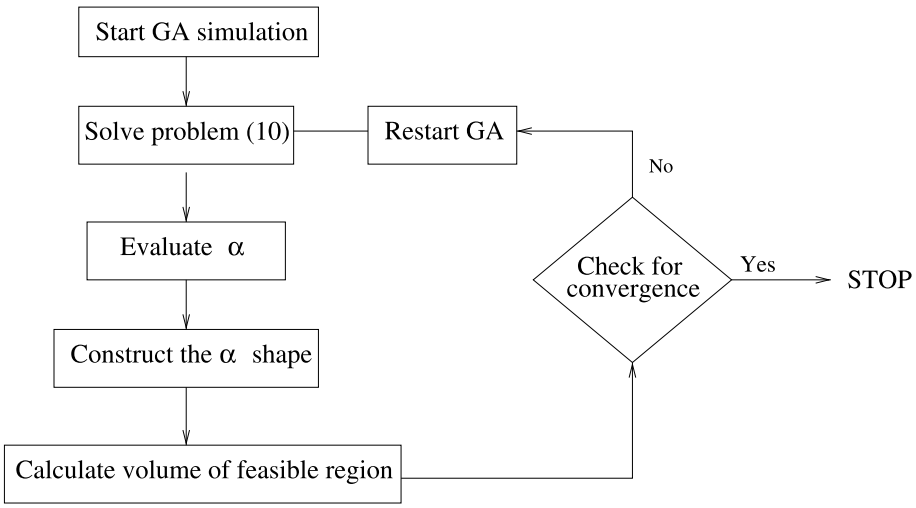
Shape Reconstruction Methods for Nonconvex Feasibility Analysis, Figure 4
Sampling of feasible space using (a) genetic algorithm and (b) random sample

The working principle of a GA is based on generating multiple numbers of good solutions. Hence evaluating the volume for each of the feasible parameter values (θ) will reduce the efficiency of the procedure because of the repetition of the solution. To avoid this, a memory of the sampled parameter value is maintained and updated. For every generated chromosome in the population of the GA simulation, the stored parameter values are searched to check for uniqueness of the new solution. If a new solution is unique, then the constraints are evaluated; else it is updated from the memory. Chromosome evolution through 2000 generations requires a total of 40,000 function calls, of which only 3064 are unique and 938 are feasible points as illustrated in Fig. 4. The same problem was solved by drawing random samples in the range $(-20, 20)$ for both uncertain parameters (Fig. 4b), where 9830 function calls were required to generate 950 feasible points. However, this procedure is particularly advantageous when the feasible region is a small portion of the entire parameter range. Otherwise its performance becomes comparable to random sampling over the entire parameter range.

In the above formulation, the volume of the feasible region, V_{feas}, is computed by generating an α shape of the sampled points. An alternative formulation for generating the sampled data set is given by

$$\begin{aligned} \max \sum \Theta_{\text{feas}}, \\ \theta_2 - 2\theta_1 - 15 \le 0, \\ \frac{\theta_1^2}{2} + 4\theta_1 - 5 - \theta_2 \le 0, \\ 10 - \frac{(\theta_1 - 4)^2}{5} - \frac{\theta_2^2}{0.5} \le 0, \\ \theta_2(6 + \theta_1) - 80 \le 0, \end{aligned} \tag{10}$$

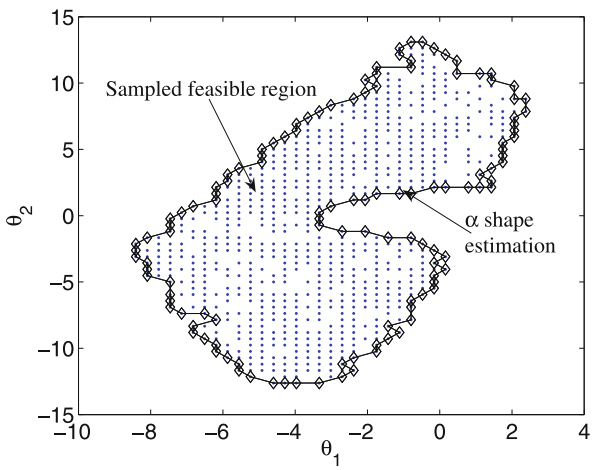
where Θ_{feas} represents a feasible sample point and the objective is to maximize the total number of sampled feasible points. This formulation is computationally less demanding since it does not require volume evaluation of the feasible region at every step. However, it suffers from the disadvantage of a lack of a convergence criterion. To overcome this problem, a hybrid of these two



Shape Reconstruction Methods for Nonconvex Feasibility Analysis, Figure 5
Modified algorithm for sampling of feasible region

formulations (9) and (10) is used, where the main algorithm is evolved according to formulation (9), and the volume is evaluated only at intermediate points to check for convergence of the simulation. The overall procedure is illustrated in Fig. 5.

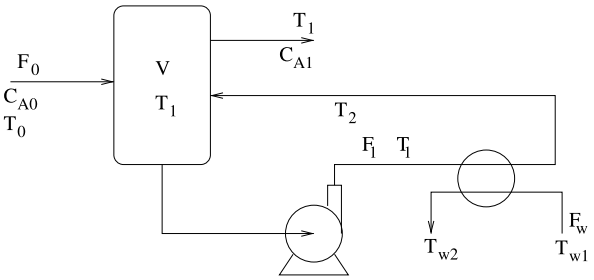
Once a good estimate of the feasible region is obtained by the sampling scheme, the surface-reconstruction algorithm is used to determine points forming the boundary of the feasible region, which are then joined by a straight line as illustrated by Fig. 6. The value of α plays a crucial role in determining the degree of detail captured by the α shape. The α value determined by the procedure outlined in Sect. "α-Shape Approach" for the 938 sampled points is 25, which was found to capture the nonconvex nature of the object with adequate accuracy, as illustrated in Fig. 6.



Shape Reconstruction Methods for Nonconvex Feasibility Analysis, Figure 6
Performance of α shape in predicting feasible space using $\alpha = 25$

Process Operation Example

This example represents the flow sheet shown in Fig. 7, consisting of a reactor and heat exchanger [5] where a first-order exothermic reaction $A \rightarrow B$ is taking place. The existing design has a reactor volume (V) of 4.6 m^3 and a heat exchanger area (A) of 12 m^2. Two uncertain parameters are considered, the feed flow rate F_0

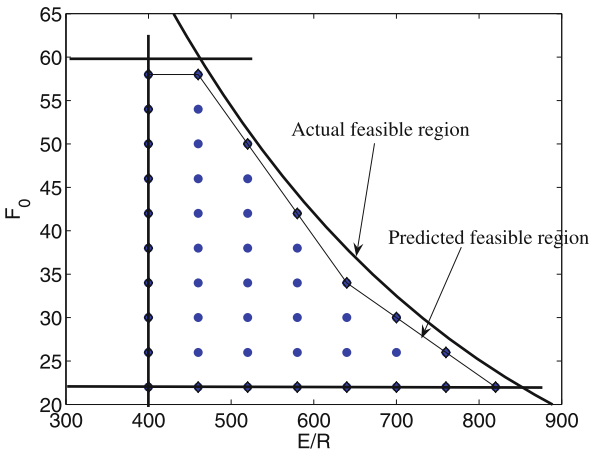


Shape Reconstruction Methods for Nonconvex Feasibility Analysis, Figure 7
Reactor-cooler example

and the activation energy E/R. A mathematical model of this process is given by:

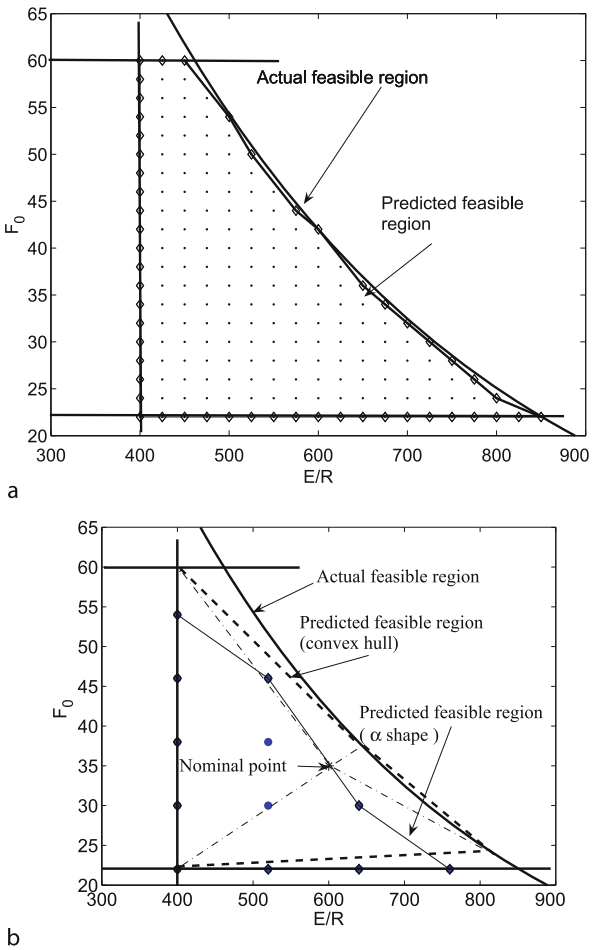
$$
\begin{aligned}
F_0(c_{A0} - c_{A1})/c_{A0} &= Vk_0 \exp(-E/RT)c_{A1} \\
(-\Delta H)F_0(c_{A0} - c_{A1})/c_{A0} &= F_0 c_p(T_1 - T_0) + Q_{HE} \\
Q_{HE} &= F_1 c_p(T_1 - T_2) \\
Q_{HE} &= F_w c_{pw}(T_{w2} - T_{w1}) \\
Q_{HE} &= AU\Delta T_{ln} \\
\Delta T_{ln} &= \frac{(T_1 - T_{w2}) - (T_2 - T_{w1})}{ln(T_1 - T_{w2})/(T_2 - T_{w1})} \\
V^d &\geq V \\
(c_{A0} - c_{A1})/c_{A0} &\geq 0.9 \\
311 \leq T_1 &\leq 389 \\
T_1 - T_2 &\geq 0.0 \\
T_{w2} - T_{w1} &\geq 0.0 \\
T_1 - T_{w2} &\geq 11.1 \\
T_2 - T_{w1} &\geq 11.1 \\
T_0 &= 333\,\text{K}, T_{w1} = 300\,\text{K}, \\
U &= 1635\,\text{kJ/(m}^2 hK) \\
c_p &= 167.4\,\text{kJ/kmol}, \\
c_{A0} &= 32.04\,\text{kmol/m}^3, \\
-\Delta H &= 23260\,\text{kJ/kmol}\,. \qquad (11)
\end{aligned}
$$

The range of uncertain parameters E/R and F_0 over which the design remains feasible is illustrated in Fig. 8.



Shape Reconstruction Methods for Nonconvex Feasibility Analysis, Figure 8
Performance of α shape in predicting the feasible space of reactor-cooler example

The aim is to have a description of the range of the parameters E/R and F_0 over which the operation remains feasible. Following the proposed approach for feasibility analysis, the feasible space is first sampled by solving the problem at different values of the parameters, and a representation of the feasible region is obtained. In the next step, these sampled points are analyzed by α shape to identify points lying on the boundary of the feasible region. These identified surface points are joined by straight lines to obtain a polygonal estimation of the feasible region. Figure 8 compares the actual feasible region with that of α-shape estimation obtained with 400 sample points, which was found to perform with great accuracy. To understand the effect of



Shape Reconstruction Methods for Nonconvex Feasibility Analysis, Figure 9
Effect of sampling density on the performance of α shape (a) 100 points (b) 25 points

sample density on the performance of the α-shape procedure, the feasible region evaluation was performed with fewer sample points of 100 and 25, as illustrated in Fig. 9. The α-shape prediction was found to underpredict the feasible region since the sampling density was inadequate in capturing the entire region. However, there was no overprediction of the nonconvex feasible region. Figure 9b also compares the performance of the convex hull with that of α shape, where it is observed that even though the sampling density was very low, α shape still captured the nonconvex nature of the feasible region. The convex hull is constructed by performing line searches toward the vertices of the uncertain space to locate points on the boundary of the feasible region. The convex hull covers a larger percentage of the feasible region compared to the α shape for the case of sparse sampling, but it overpredicts the feasible region over the nonconvex constraint. The performance of α shape is directly dependent on the information captured by the sampling of the feasible space. It is important to know, however, that in the absence of sufficient information the α shape will be a poor predictor of the feasible space, but it will not lead to erroneous results.

Conclusions

The problem of evaluating the feasible range of a process operation is addressed in this paper using surface-reconstruction ideas. The problem definition is to evaluate and quantify the uncertain parameter range over which a process retains its feasibility. In the present approach the feasible region is viewed as an object, with process constraints defining the boundary of the object. Surface-reconstruction ideas are used to define the shape of the object. The procedure starts by first sampling the feasible region to obtain a representation of the feasible space. An α shape is then constructed of the sampled points, which identifies points forming the boundary of the object. These points are joined to have a polygonal representation of the feasible region. Finally, determination of whether a point is feasible or not can be done by a point-in-polygon check. Examples are presented to illustrate the performance of the proposed scheme in nonconvex and even disjoint problems.

The application of the proposed technique in higher dimensions becomes computationally challenging. One way of dealing with this issue is by reducing the dimensionality of the problem. The ideas of principal component analysis [19] can be utilized to map the original uncertainty space to the reduced dimensional space of important eigen directions. The α-shape ideas can then be applied in the reduced space and the feasibility information mapped back to the original uncertain space. These ideas are currently being explored by the authors.

References

1. Biegler LT, Grossmann IE (2004) Retrospective on optimization. Comput Chem Eng 28:1169–1192
2. Edelsbrunner H (1992) Weighted alpha shapes Tech. Rep. UIUCDCS-R-92–1760, Dept Comp Sci, UIUC,IL
3. Edelsbrunner H, Kirkpatrick DG, Seidel R (1983) On the shape of a set of points in a plane. IEEE Trans Inform Theory IT-29:551–559
4. Edwards K, Edgar TF, Manousiouthakis VI (1998) Kinetic model reduction using Genetic Algorithm. Comput Chem Eng 22:239–246
5. Floudas CA, Gumus ZH, Ierapetritou MG (2001) Global optimization in design under uncertainty: feasibility test and flexibility index problems. Ind Eng Chem Res 40:4267–4282
6. Goyal V, Ierapetritou MG (2002) Determination of operability limits using simplicial approximation. AIChE J 48:2902–2909
7. Grossmann IE, Halemane KP (1982) A decomposition strategy for designing flexible chemical plants. AIChE J 28:686–694
8. Grossmann IE, Sargent RWH (1978) Optimum design of chemical plants with uncertain parameters. AIChE J 24:1021–1028
9. Haines E (1994) Point in polygon strategies. In: Heckbert P (ed) Graphic Gems IV. Academic Press, Boston, p 24
10. Halemane KP, Grossmann IE (1983) Optimal process design under uncertainty. AIChE J 29:425–433
11. Jarvis RA (1977) Computing the shape hull of points in the plane. Proc IEEE Comp Soc Conf Pattern Recognition and Image Processes, 231–241
12. Kubic WL, Stein FP (1988) A theory of design reliability using probability and fuzzy sets. AIChE J 34:583–601
13. Mandal DB, Murthy CA (1997) Selection of alpha for alpha-hull in $\mathcal{R}^2$. Pattern Recog 30:1759–1767
14. Pistikopoulos EN, Mazzuchi TA (1990) A novel flexibility analysis approach for processes with stochastic parameters. Comput Chem Eng 14:991–1000
15. Rooney WR, Biegler LT (1999) Incorporating joint confidence regions into design under uncertainty. Comput Chem Eng 23:1563–1575
16. Saboo AK, Morari M, Woodcock DC (1983) Design of resilient processing plants-VIII. A resilience index for heat exchanger networks. Chem Eng Sci 40:1553–1565

17. Straub DA, Grossmann IE (1993) Design optimization of stochastic flexibility. Comput Chem Eng 17:339–354
18. Swaney RE, Grossmann IE (1985) An index for operational flexibility in chemical process design I: Formulation and theory. AIChE J 31:621–630
19. Vajda S, Valko P, Turanyi T (1985) Principal component analysis of kinetic models. Int J Chem Kinet 17:55–81

Shape Selective Zeolite Separation and Catalysis: Optimization Methods

Chrysanthos E. Gounaris, James Wei, Christodoulos A. Floudas
Department of Chemical Engineering, Princeton University, Princeton, USA

MSC2000: 74A40, 90C26

Article Outline

Keywords and Phrases
Introduction
Characterization of Molecular Footprints
Definition of Strain and Calculation of Strain Index
Strain-Based Screening
References

Keywords and Phrases

Catalysis; Molecular sieves; Selectivity; Separations; Supramolecular chemistry; Zeolites

Introduction

The use of zeolites as molecular sieves, absorbents and catalysts has today been well established in a wide variety of processes. However, almost the totality of applications involves a very small number of nearly circular structures, like Linde type A, faujasite and ZSM-5. These structures are usually modified to meet the specific needs of each process. Modification techniques, such as ion exchange or coke deposition, usually result in a distribution of pore sizes and shapes, something that retards the ability of the molecular sieve to be highly selective. On the other hand, there is a great variety of natural and synthetic zeolites that has been developed, but no significant effort has been made to find potential catalysis and separation applications for them. There could very well be existing structures that are highly selective in their unmodified state, or requiring a small amount of modification, just because their windows happen to be of the proper size and shape. Gounaris et al. [7,8] developed a mathematical framework, which is based on optimization, to address exactly this issue. The framework can identify shape selective zeolite structures and provide researchers with a rigorous way to determine the best candidate portals for the process of interest.

Characterization of Molecular Footprints

For spherical molecules going through circular windows, such as the noble gases approaching a Linde type A window, the molecular shape and the rotational orientation are not important for penetration into a channel since every possible rotation results in the same projection. The Lennard-Jones length is often used as an order-of-magnitude estimation of the size of the molecule [11]. It is also the starting point for several attempts to compare these lengths with nominal zeolite window diameters so as to identify zeolite windows that are suitable for separating a set of molecules. For instance, zeolite 3A has a diameter that is between that of H_2 and O_2; thus, it would be a good candidate for their separation. Since most molecules are not spherical, and most windows are not circular, we need more accurate methods to characterize molecules.

We start from a simple model of a molecule as atoms connected by bonds, obtained by a quantum mechanics or a molecular mechanics calculation. In the hard-sphere model, each atom is represented by a sphere of van der Waals radius, and the bond lengths and angles are considered fixed – equivalent to an inflexible molecule at absolute zero temperature.

When a molecule approaches the opening of a channel, various rotational orientations of the molecule give rise to different projections on the horizontal plane. From this ensemble of projections, the ones that are most favorable for penetration are usually the smallest, which we would call “footprints.” If the molecule is a spheroid with three different axes, then the molecule should be oriented so that the longest axis is perpendicular to the plane, and the footprint is the ellipse formed by the two smaller axes. If the molecule is a rectangular parallelepiped, then the footprint is the rectangle formed by the two smaller axes. This suggests the fol-

lowing useful quantitative measures of the size of a footprint involving no more than two parameters:

(a) The footprint is the projection that can be enclosed by the smallest possible circle, characterized by its radius ρ_0.
(b) The footprint is the projection that can be enclosed by the ellipse with the smallest possible area. This footprint is characterized by its major and minor radii, which are denoted, respectively, with ρ_1 and ρ_2. Its eccentricity is defined as $e = \sqrt{1 - \rho_2/\rho_1}$.
(c) The footprint is the projection that can be enclosed by the rectangle with the smallest possible area. This footprint is characterized by its major and minor lengths, which are denoted respectively with α_1 and α_2. The aspect ratio is defined as $\mathrm{AR} = \alpha_2/\alpha_1$.
(d) The footprint is the projection that minimizes the sum of the distances of the projected atomic nuclei from a suitable center. This footprint can be characterized by a major diameter d_M, which is the largest distance between two points on the edge of the footprint, and a minor one d_m, which is the width in the direction perpendicular to the major diameter. For an exact definition and a detailed description of the calculation of these diameters, see Gounaris et al. [8].
(e) The footprint is the projection that minimizes the sum of the distances of all the projected atomic nuclei from each other. It can be quantified with the same parameters as in measure *d*.

The computations of these quantitative measures are formulated as nonlinear programming problems or as bilevel nonlinear programming problems, and are described in detail in Gounaris et al. [8]. See Gounaris et al. [7] for examples of footprints of popular molecules and for relevant illustrations.

Definition of Strain and Calculation of Strain Index

When a guest molecule approaches a host portal, there are three possible outcomes: free passage, constrained passage, and no passage. When the molecular projection of the guest can be entirely contained within the portal, there is no hindrance and the passage is free. When some of the atomic nuclei in the projection fall outside the window, and no rotation and translation can prevent this, then there is no passage. However, if all nuclei fall inside the portal but some atomic radii extend beyond this area, then there can be constrained passage in the sense that the atomic spheres have to be squeezed so as to fit in the portal.

We define the amount of distortion on a single atom as

$$\delta = \frac{r_\mathrm{s}}{r_\mathrm{o}} \tag{1}$$

where r_o is the original atomic radius and r_s is the squeezed atomic radius.

The total strain, *S*, for a guest to penetrate through a host portal is quantified as

$$S = S_\mathrm{G} + S_\mathrm{H} = \sum_i \left(\frac{1}{\delta_i^{12}} - \frac{1}{\delta_i^{6}} \right) + \sum_j \left(\frac{1}{\delta_j^{12}} - \frac{1}{\delta_j^{6}} \right) \tag{2}$$

where δ_i and δ_j are, respectively, the amounts of distortion on the *i*th guest (G) and *j*th host (H) atom. In practice, only the oxygen atoms in the zeolite window and the outer (often hydrogen) atoms of the molecule make significant contributions.

There is a strain associated with every projection of the guest molecule, but there is some optimal projection that exhibits the minimum possible strain, denoted as S^*. This optimal projection for one channel may be different from that for another channel with a different shape. For instance, a molecule in the shape of a cylinder can give a rectangular or a circular projection, depending on the requirement posed by the shape of the portal.

Since S^* values could span a wide range of orders of magnitude, the introduction of a logarithmic scale is necessary for a better representation. We define the "strain index" as

$$\mathrm{SI} = \log(1 + S^*)\,. \tag{3}$$

The strain index can serve as a measure of the total distortion required for penetration to take place. A host–guest pair exhibiting a strain index of zero would correspond to a free passage, while a strain index approaching infinity would correspond to no passage at all.

A rigorous algorithmic framework to calculate robustly the strain index of a given host-guest pair was developed by Gounaris et al. [7,8]. The optimization formulation that models the problem is described below. Let us first present the notation used.

Indices:

$i = 1, 2, \ldots, M$ Atoms in the guest molecule
$j = 1, 2, \ldots, N$ Atoms in the host molecule
$k = 1, 2, \ldots, K$ Constraints defining host interior

Decision variables:

φ, ϑ, ψ Rotation angles
xt, yt Translation of guest projection on the x-y plane
δ_i, δ_j Required distortion of guest and host atoms

Auxiliary variables:

d_{ij} Distance of the ith guest atom projection from the jth host atom
$\begin{bmatrix} x_i \\ y_i \end{bmatrix}$ Position of the ith guest atom (after rotation and projection)

Parameters:

$\begin{bmatrix} x_i^0 \\ y_i^0 \\ z_i^0 \end{bmatrix}$ Coordinates of the ith guest atom (random orientation)
$\begin{bmatrix} xh_j \\ yh_j \end{bmatrix}$ Position of jth host atom on x-y plane
a_k, b_k, c_k Parameters defining the convex hull of the host
r_i, r_j Effective atomic radius of guest and host atoms

The objective is to minimize the total strain, S, required for penetration:

$$\min_{\substack{\varphi,\theta,\psi \\ xt,yt \\ \delta_i,\delta_j}} S = \min_{\substack{\varphi,\theta,\psi \\ xt,yt \\ \delta_i,\delta_j}} \left\{ \sum_i \left(\frac{1}{\delta_i^{12}} - \frac{1}{\delta_i^6} \right) + \sum_j \left(\frac{1}{\delta_j^{12}} - \frac{1}{\delta_j^6} \right) \right\} . \tag{4}$$

For every guest atom, the position of its center should correspond to a valid rotation that resulted from the original conformation provided. According to the "$x\,y\,z$" convention for rotation matrices, the coordinates of the projected atom nuclei are given by

$$\begin{aligned} x_i &= \cos\theta \cdot \cos\varphi \cdot x_i^o + \cos\theta \cdot \sin\varphi \cdot y_i^o \\ &\quad - \sin\theta \cdot z_i^o + xt \quad \forall i \\ y_i &= (\sin\psi \cdot \sin\theta \cdot \cos\varphi - \cos\psi \cdot \sin\varphi) \cdot x_i^o \\ &\quad + (\sin\psi \cdot \sin\theta \cdot \sin\varphi + \cos\psi \cdot \cos\varphi) \cdot y_i^o \\ &\quad + \cos\theta \cdot \sin\psi \cdot z_i^o + yt \quad \forall i . \end{aligned} \tag{5}$$

The terms xt and yt allow for translation of the projection on the x–y plane, so as to obtain a better fit with respect to the host. Note that the host–guest conformations are provided independently, and there is no requirement that they use the same reference coordinate system.

For every pair of host–guest atoms, we impose the condition that their effective spheres cannot intersect with each other, therefore implying that they have to be squeezed to fit:

$$\begin{aligned} d_{ij} &\geq \delta_i \cdot r_i + \delta_j \cdot r_j \\ d_{ij}^2 &= (x_i - xh_j)^2 + (y_i - yh_j)^2 \quad \forall (i,j) . \end{aligned} \tag{6}$$

In order to avoid obtaining (otherwise valid) solutions where the guest is completely outside the portal area, we have to include also a set of constraints that outer-approximates the portal. A set of linear constraints that serves the purpose is the one that describes the convex polygon whose vertices coincide with the atom centers of the host:

$$a_k \cdot x_i + b_k \cdot y_i \leq c_k \quad \forall (i,k) . \tag{7}$$

The parameters a_k, b_k,and c_k can be easily calculated from the host atom coordinates $\begin{bmatrix} xh_j \\ yh_j \end{bmatrix}$. Note that only those atoms that participate in the host's convex outer approximation are used for this calculation; therefore, K does not necessarily have to equal N. This only happens in the case of convex portals.

Finally, the following bounds have to be applied to the decision variables of the problem:

$$-\pi < \varphi, \theta, \psi \leq +\pi \tag{8}$$

$$\begin{aligned} 0 \leq \delta_i \leq 1 \quad \forall i \\ 0 \leq \delta_j \leq 1 \quad \forall j . \end{aligned} \tag{9}$$

The bounds on the Euler angles are imposed so as not to obtain periodic solutions, while the bounds on the deltas relate to their definitions. Note that no bound is imposed on the translation variables xt and yt, which are allowed to vary freely.

The minimization of the objective function (4), subject to constraints (5)–(9), constitutes a nonlinear programming problem that involves continuous variables. The problem is nonconvex, with nonconvexities introduced both by the objective function (definition of total strain) and by the constraints (projected rotations

Shape Selective Zeolite Separation and Catalysis: Optimization Methods, Figure 1
Complete strain index database

and atom–atom distances). This problem can be rigorously addressed by deterministic global optimization methods such as αBB [1,2,5,6,10]. For computational efficiency reasons, Gounaris et al. [7,8] chose instead to employ local optimization methods with an insightful initialization scheme that was effective in avoiding convergence to nonglobal solutions.

They applied their method on a large database of zeolite portals and molecules. In particular, they considered 38 popular molecules and 123 zeolite structures (corresponding to a total of 217 different windows). For an exact list of the windows considered, see Gounaris et al. [8]. Complete reference for all these structures can be found in the *Atlas of the International Zeolite Association* [3,4]. Figure 1 shows a schematic representation of all the results and can serve as a database of strain indices. Such a database shows the relations between many molecules and zeolite rings and can be a powerful tool for the identification of portal candidates that are selective between two molecules. It can offer a systematic screening technique which has an energetic basis and does not rely exclusively on qualitative measures. Once it has been identified that a zeolite structure is a good candidate to admit selectively some molecule, experimental studies should be employed to accurately determine diffusion rates or Langmuir isotherms. These results could also be supported further by molecular dynamics or Monte Carlo simulations [9,12,13] which are tailored to study the specific sorbent/sorbate systems under consideration.

Strain-Based Screening

When a set of molecules approaches a zeolite channel window, the results can be described as a triage. When all the molecules pass, such as in the case of hydrogen,

nitrogen and oxygen approaching the relatively large opening of faujasite, there is no selectivity and no separation. When none of the molecules pass, such as the case of oleic and linoleic acid approaching the channel of SAPO-56, there is also no selectivity and no separation. When some of the molecules have strain indices different from those of others, such as in the case of ethane and ethylene approaching ERS-7, then there is selectivity and potential for catalysis or separation. A higher strain index may reduce the diffusion rate or reduce the equilibrium adsorption, instead of completely denying passage, but it would nevertheless serve the separation scheme.

When a molecule is being squeezed to fit a host channel, some activation energy is required which would lead to a decrease of the equilibrium concentration in the channel according to the Boltzmann equation:

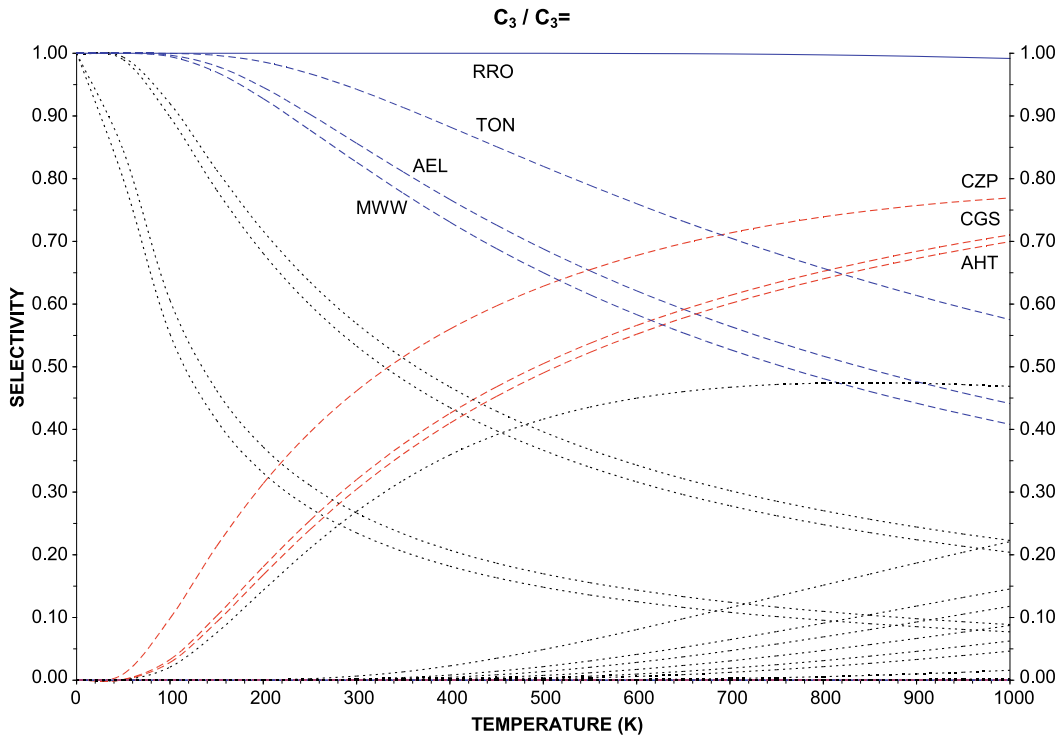
$$\frac{C}{C_0} = \exp\left(-\frac{E}{RT}\right) = \exp\left(-\frac{4\varepsilon S}{RT}\right) , \qquad (10)$$

where ε is some "hardness" coefficient. An averaged Lennard–Jones parameter may be used.

Let us define selectivity between two molecules A and B, γ_{AB}, as the difference between their reduced equilibrium concentrations:

$$\gamma_{AB} = \left|\left(\frac{C}{C_0}\right)^{A} - \left(\frac{C}{C_0}\right)^{B}\right| = \left|\exp\left(-\frac{E^{A}}{RT}\right) - \exp\left(-\frac{E^{B}}{RT}\right)\right| . \qquad (11)$$

If the distortion energies required are similar, both molecules will penetrate the same relative amount, and therefore $C^A/C^B \approx C_0^A/C_0^B$ and there is no selectivity ($\gamma_{AB} \to 0$). On the other hand, if the energies are substantially different, the penetration levels will be different and high selectivity will be achieved ($\gamma_{AB} \to 1$). In the case where $E_A \to 0$, selectivity is at a maximum at very low temperatures, but a rise in temperature can activate molecules B and selectivity will decline.



Shape Selective Zeolite Separation and Catalysis: Optimization Methods, Figure 2
Selectivity vs. temperature for the C_3/C_3= system. *RRO* RUB-41, *TON* Theta-1, *AEL* AlPO-11, *MWW* MCM-22, *CZP* chiral zincophosphate, *CGS* cobalt gallium phosphate-6, *AHT* AlPO-H2

An illustrative example of such calculations is presented in Fig. 2, where selectivity is plotted versus temperature for the system of propane/propylene. RUB-41 is identified as the most promising candidate for the separation of the two C_3 molecules. It maintains a very high selectivity along the whole range of temperatures considered. Theta-1, AlPO-11 and MCM-22 are also selective at ambient temperature, but their performance deteriorates at higher temperatures. All these structures correspond to the case where one of the two molecules (propylene) enjoys a free passage through the portal, while the second molecule (propane) has to experience some distortion. A different trend holds for chiral zincophosphate (CZP), cobalt gallium phosphate-6 and AlPO-H2, which seem to benefit from an increase in temperature. The potential of RUB-41 to be highly selective on the C_3 system can be explained by the high degree of similarity between propylene's projection and the actual shape of the portal, a similarity that resembles the analogy between a lock and a key.

References

1. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB, for general twice-differentiable constrained NLPs – I. Theoretical advances. Comput Chem Eng 22:1137–1158
2. Akrotirianakis IG, Floudas CA (2004) A new class of improved convex underestimators for twice continuously differentiable constrained NLPs. J Glob Optim 30:367–390
3. Baerlocher C, McCusker LB (2006) Database of Zeolite Structures: http://www.iza-structure.org/databases/
4. Baerlocher C, Meier WM, Olson DH (2001) Atlas of Zeolite Framework Types: 5th revised edn. Elsevier, Amsterdam
5. Floudas CA (2000) Global optimization in design and control of chemical process systems. J Process Control 10:125–134
6. Floudas CA, Akrotirianakis IG, Caratzoulas S, Meyer CA, Kallrath J (2005) Global optimization in the 21st century: Advances and challenges. Comput Chem Eng 29:1185–1202
7. Gounaris CE, Floudas CA, Wei J (2006) Rational design of shape selective catalysis and separation: I Concepts and analysis. Chem Eng Sci 61:7933–7948
8. Gounaris CE, Wei J, Floudas CA (2006) Rational design of shape selective catalysis and separation: II. Mathematical model and computational studies. Chem Eng Sci 61:7949–7962
9. June RL, Bell AT, Theodorou DN (1991) Transition-state studies of xenon and SF_6 diffusion in silicalite. J Phys Chem 95:8866–8878
10. Meyer CA, Floudas CA (2005) Convex underestimation of twice continuously differentiable functions by piecewise quadratic perturbation: Spline alpha BB underestimators. J Glob Optim 32:221–258
11. Reid RC, Prausnitz JM, Poling BE (1987) The Properties of Gases and Liquids, 4th edn. McGraw-Hill, New York
12. Smit B, Siepmann JI (1994) Computer simulations of the energetics and siting of n-alkanes in zeolites. J Phys Chem 98:8442–8452
13. Vlugt TJH, Krishna R, Smit B (1999) Molecular simulations of adsorption isotherms for linear and branched alkanes and their mixtures in silicalite. J Phys Chem B 103:1102–1118

Shor, Naum Zuselevich

Alla R. Kammerdiner
Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA

MSC2000: 01A70, 90-03

Article Outline

Biographical Summary
Biographical Details
References

Biographical Summary

Naum Zuselevich Shor (1937–2006) is recognized as one of the paramount researchers in the field of optimization. He is well known for his significant contributions to many important areas of optimization, including nonlinear and stochastic programming, computational methods for nonsmooth optimization, discrete optimization problems, matrix optimization, dual quadratic bounds in multiextremal programming problems, and numerical algorithms for solving large-scale optimization problems.

Biographical Details

A renowned Ukrainian mathematician, Shor was born on January 1, 1937. His childhood took place during the horrific years of World War II. In 1954, Shor entered the Mechanics and Mathematics Department of National Taras Shevchenko University in Kiev, Ukraine. Two years later in 1956, the young brilliant scientist Victor Mikhylovich Glushkov moved to Kiev and was

appointed director of a newly created Computing Center (formerly one of the laboratories at the Institute of Mathematics) of the National Academy of Sciences of Ukraine (NASU). Shor decided to focus his attention on differential algebra and began working on his diploma thesis under Glushkov's supervision.

In 1958, Shor graduated from Taras Shevchenko University and was invited by Glushkov to join the Computing Center, which 4 years later became the Institute of Cybernetics of NASU. There Shor became an active part of the research group guided by another talented mathematician, Vladimir Sergeevich Mikhalevich. First, Shor examined the problems of modeling and optimization of the reliability of computing devices, as well as application of noise power spectrum analysis to various problems in radiology. By 1960, the research team of Mikhalevich had evolved into the Department for Applied Problems with a focus on optimal planning and design. This transformation had completely shifted Shor's scientific interests toward the field of optimization, which emerged as a new area of mathematics in the 1940s. Working together on the optimal selection of design decisions, Shor and Mikhalevich constructed a numerical procedure for sequential analysis of variants. The proposed procedure represented a generalization of dynamic programming algorithms. It could easily be employed for solving various applied problems of optimal design and planning, including, but not limited to, gas supply systems, electrical networks, and transportation route systems. As a result, their ingenious numerical scheme received a high number of citations. Continuing his work in optimal design, Shor had also suggested a method for solving optimal design problems for lengthy objects and treelike structures. In addition, Shor was the first to apply the subgradient descent method to optimization of nonsmooth functions in 1962. Specifically, using the subgradient scheme, he devised an approach for solving large-scale dual network transportation problems by reduction to the maximization of a piecewise linear function. His approach later became well known as the generalized gradient descent method. In 1964, Shor defended his Ph.D. dissertation entitled "On the Structure of Algorithms for Numerical Solution of Problems of Optimal Planning and Design."

After earning his doctor of philosophy degree in 1964, Shor continued his work on application of his generalized gradient descent method to various mathematical programming problems, including block programming and two-stage stochastic programming. In 1967, he also cowrote a book with Mikhalevich on the computational approaches to optimal selection of design decisions. Only 1 year after the book on optimal design had been published, Yuriy Ermoliev and Shor devised a modification of Shor's subgradient method for solving two-stage stochastic programming problems. This revolutionary approach was later advanced even further by the research team led by Ermoliev and became known as the direct quasi-gradient method for optimization under uncertainty.

Another pioneering approach in optimization, which was introduced by Shor in 1970, was based on the idea of space transformation known as dilation. Almost concurrently, Shor worked on two methods involving space dilation. The first technique is the method with space dilation in the direction of the subgradient, which is used for solving systems of nonlinear equations and inequalities. The second method, also known as the r-algorithm, utilizes the operation of space dilation in the direction of the difference between two consecutive subgradients. The r-algorithm has become one of the most efficient procedures for solving complex optimization problems. Nearly a decade of his research on the subgradient and subgradient-type methods with space transformation was finally summarized by Shor in his monograph *Minimization Methods for Non-Differentiable Functions and Applications*, which was first published in 1979, and just 6 years later was translated into English.

Remarkably, the famous ellipsoid method, which was independently formulated by A.S. Nemirovsky and D.B. Yudin in 1975, and which was used by L.G. Khachiyan in 1979 to prove that linear programming problems can be solved in polynomial time, is actually a special case of Shor's method with space dilation in the direction of the difference between two consecutive subgradients.

In the early 1980s, Shor became captivated with graph theory while working on the network optimization problems. Together with his Ph.D. student G.A. Donets, he was investigating the graph coloring problems. In particular, they formulated a hypothesis on the number of solutions for coloring of a plain graph. This hypothesis on plain graph colorings is ev-

idently supported by computational experiments. In 1982, their treatise on the algebraic approach to the problems of plain graph coloring was published.

In the second half of the 1980s, Shor collaborated with S.I. Stetsenko on quadratic extremal problems. Later he investigated dual estimates in multiextremal problems and produced a paper on this subject in 1992. The same year, Shor's joint paper with O.A. Berezovski, where they described new procedures for constructing optimal inscribed and circumscribed ellipsoids, also appeared in the press. The scope of his research was continuously expanding to other complex areas of optimization to include minimization of matrix functions (1995), generalized set partitioning problems (1996), polynomial optimization problems (1998), nonsmooth optimization in stochastic programming (1999), and Lagrangian bounds for multiextremal polynomial and discrete optimization problems (2002). Furthermore, in 1998, his extensive analysis of polynomial problems using methods of nondifferential optimization was published in a monograph. The book included a comprehensive review of techniques used in nondifferential optimization as well as their application to various problems, such as the problems of discrete optimization and graph optimization, polynomial problems, and optimal Lyapunov functions. Polynomial problems were given special consideration. Specifically, Shor discovered that in order for the dual quadratic bound of a polynomial to be equal to the global minimum of such a polynomial, it is necessary and sufficient that the difference between the polynomial and its global minimum could be represented as a sum of squares of real polynomials. This result illustrates the connection between the problems of nonconvex polynomial optimization and David Hilbert's 17th problem about representation of a definite rational function as a quotient of the sum of squares, which was posed by Hilbert in 1900 and solved by Emil Artin in 1927.

Until his death on February 25, 2006, Shor kept actively working on different intricate optimization problems. His dedication to research, immense knowledge, and unstoppable intellect were manifested in his undisputable achievements in the field of mathematical programming. During his long research career, Shor won numerous awards. Among the most prestigious are both the former USSR and Ukraine State Prizes in Science and Technology (1973, 1981, 1993, and 2000). In addition, for his great contribution to computational methods for solving large-scale optimization problems, Shor was recently awarded the Glushkov Prize and the Mikhalevich Prize. In recognition of his lifelong scientific accomplishments, he was elected as an Associate Member of the National Academy of Sciences in Ukraine in 1990, and became a full Member of the Academy on December 4, 1997. During his life, Shor had over 180 research papers and nine books published and supervised Ph.D. dissertations of over 35 students who now continue his scientific work not only in Ukraine, but all over the world.

References

1. Zhurbenko N, Stetsyuk P, Lykhovyd O, Sharifov F, Berezovski O (2002) Congratulations to Naum Shor on his 65th birthday. J Global Optim 24:111-114
2. Homenko A (2002) Bad Arithmetic: Went to Greece, but a suitcase was left in Hungary… Ukrainian newspaper "Today", No.1344, December 28 (http://today.viaduk.net/Segodnya.nsf/. Accessed on 14 Jan 2007)
3. http://www.icyb.kiev.ua/. Accessed 16 Jan 2007

Shortest Path Tree Algorithms

SPT

Paola Festa
Dip. Mat. e Inform., Universitá Salerno, Baronissi, Italy

MSC2000: 90C27, 90B10

Article Outline

Keywords
Mathematical Model
A Generic Shortest Path Algorithm
Implementations of the Generic Algorithm
  Label Setting Methods
  S-HEAP
  S-DIAL
  Label Correcting Methods
  Bellman–Ford Method
  D'Esopo–Pape Method
Auction Algorithms
  Graph Reduction in Auction Algorithms
  Modified Version of the Standard Auction Algorithm
  Graph Collapsing In Auction Algorithms

Graph Collapsing Auction Algorithm
Virtual Source Concept Applied in Auction Algorithms
A New Virtual Source Algorithm
Computational Results
Conclusions
See also
References

Keywords

Combinatorial optimization; Auction technique; Network optimization; Shortest path problem

The shortest path tree problem is a classical and widely studied combinatorial problem ([1,23,24]). The scope of this article is to provide an extensive treatment of the major classical approaches. It then proceeds focusing on the auction algorithm and some of its recently developed variants. There is a discussion of the theoretical and practical performance of the treated methods comparing their effectiveness.

Mathematical Model

The shortest path problem can be posed in more than one way:

1) to find the shortest path from a single source to a single destination;
2) to find the shortest path from each of several sources to each of several destinations;
3) to find the shortest path from one single source to all destinations.

Problems of type 3) are also called *shortest path tree problems* (SPT).

Before describing the single source-all destinations shortest path problem, we report in the following the needed notation and definitions.

Let $G = (V, E, C)$ be a directed graph, where

- V is a set of nodes, numbered $1, \ldots, n$;
- $E = \{(i, j): i, j \in V\}$ is a set of m edges;
- C: $E \to \mathbf{R}$ is a function that assigns a length to any edge $(i, j) \in E$;
- a *forward path* $P = \{(i_1, i_2), \ldots, (i_{k-1}, i_k)\}$ is a set of edges, whose length is the length of its edges.

In order to assure that the shortest path problem admits a solution, it must be assumed that:

- all cycles in the graph have nonnegative length;
- the graph is strongly connected.

The last assumption can be removed by defining the distance between not connected nodes equal to $+\infty$.

Let moreover s be the label of the source node; then the *single source shortest path tree problem* (SPT) can be formulated as

$$\begin{cases} \min & \displaystyle\sum_{(i,j)\in E} c(i,j)x(i,j) \\ \text{s.t.} & \displaystyle\sum_{(i,j)\in E} x(i,j) - \sum_{(h,i)\in E} x(h,i) = b_i \\ & b_i = -1,\ i \neq s, \quad \text{and} \quad b_s = n-1 \\ & x(i,j) \in \{0,1\}, \quad \forall (i,j) \in E. \end{cases} \tag{1}$$

The dual problem (DSPT) is:

$$\begin{cases} \max & (n-1)\pi(s) - \displaystyle\sum_{j\neq s} \pi(j) \\ \text{s.t.} & \pi(i) - \pi(j) \leq c(i,j), \quad \forall (i,j) \in E, \end{cases} \tag{2}$$

where $\pi(i)$ is the dual variable associated with the node i.

A Generic Shortest Path Algorithm

Any shortest path algorithm for the single source-all destinations problem maintains and adjusts a vector $\{\pi(1), \ldots, \pi(n)\}$ of *distance labels* that can be scalars either ∞ and that satisfy

Proposition 1 *Let $\{\pi(1), \ldots, \pi(n)\}$ be scalars satisfying*

$$\pi(j) \leq \pi(i) + c(i,j), \quad \forall (i,j) \in E, \tag{3}$$

and let P be a path starting from a node i_1 and ending at a node i_k. If

$$\pi(j) = \pi(i) + c(i,j), \quad \forall (i,j) \in P. \tag{4}$$

Then P is a shortest path from i_1 to i_k.

The conditions (3) and (4) are also known as *complementary slackness conditions* (CSC) from the connection of the shortest path problem with the minimum cost flow problem. The generic shortest path algorithm starts with some vector of labels $\{\pi(1), \ldots, \pi(n)\}$ and successively selects edges (i, j) that violate (4). For each violating edge it sets

$$\pi(j) = \pi(i) + c(i,j)$$

and stops when CSC is satisfied by all edges. Intuitively, the labels $\pi(i)$ can be view as the length of some path P_i from the source to the node i. Therefore, if $\pi(j) > \pi(i) + c(i, j)$, the path obtained by extending P_i by edge (i, j) is shorter than P_j whose length is $\pi(j)$. This can be iterated to find successively better paths from the source to various destinations.

The violating edges could be arbitrarily chosen, but a more efficient way is to establish an order of selecting nodes from a set L, called *candidate list*, and checking violation of the CSC for all of their outgoing edges.

Let the node labeled 1 be the source node; then the pseudocode of a prototype shortest path algorithm is as follows:

```
Set L = {1}, π(1) = 0, π(i) = ∞ ∀i ≠ 1.
WHILE L ≠ ∅
    select from L a node i;
    FOR each outgoing edge (i, j)
        IF π(j) > π(i) + c(i, j)
            set π(j) = π(i) + c(i, j)
            add j to L if j ∉ L
        ENDIF
    ENDFOR
ENDWHILE
```

Pseudocode of a prototype shortest path algorithm

Implementations of the Generic Algorithm

In the literature there exist many implementations of the generic algorithm that differ in the criterion of selection of the next node to be removed from the set L. Traditionally, they are divided into two groups:

1) *Label setting methods*: the node i to be removed from L corresponds to the minimum label. If the input data are nonnegative, it can be shown that each node will enter L at most once and its label has permanent value the first time that node is extracted from L. At each iteration must be calculated the minimum label over L and many implementations of this approach differ in the procedure they use to obtain that minimum.
2) *Label correcting methods*: the choice of the node i requires less calculations, even if a node i can be involved more than once.

Label Setting Methods

The first label setting algorithm is due to E.W. Dijkstra in 1959 [20]. In this method the next node to be removed from L is the node i such that $i = \arg\min_{j \in L} \pi(j)$. There are different versions of this algorithm depending on the particular data structure representing the set L and used to facilitate the removal and the addition of nodes, as well as finding the node with the minimum label and this choice is crucial for good practical and theoretical performance. The most famous and efficient Dijkstra-like algorithms are S-HEAP and S-DIAL.

```
Set L = {1}, π(1) = 0, π(i) = ∞ ∀i ≠ 1.
Set L(1) = 1, L(last) = nil, p_i = nil ∀i
WHILE L ≠ ∅
    i = L(1);
    replace L(1) by L(last)
    order heap L
    FOR each outgoing edge (i, j)
        IF π(j) > π(i) + c(i, j)
            set π(j) = π(i) + c(i, j)
            set p_j = i
            IF j ∉ L
                insert j into L as L(last)
                order heap L
            ENDIF
        ENDIF
    ENDFOR
ENDWHILE
```

Pseudocode of S-HEAP

S-HEAP

The data structure chosen to represent the set L is a *binary heap*, i. e. a tree whose radix corresponds to the node having the minimum label and in which each node has label not greater than those of its children. By using this data structure, the removal of the node $L(1)$ corresponding to the minimum label, the insertion of a new node in the last position L (last) and the correction of the label of a node already inserted in L have complexity $O(\log q) \leq O(\log n)$, where q is the cardinality of L and n the number of nodes. At each iteration the radix of the heap L is removed and some labels of nodes belonging to L may decrease. Therefore, some nodes may have to be repositioned in L, while

some other nodes may enter in it for the first time in L and have to be put at the right position. Each of the just mentioned operations takes $O(\log n)$ time. The total number of insertions is n as well as that of removals. Therefore, the number of operations needed to keep ordered the heap L is $O((n + r)\log n)$, where r is the total number of repositioning operations. To get an upper bound on r, it is enough to observe that there is at most one repositioning for each edge, because each edge is involved at most once. Thus, $r \leq m$ and the total operation count needed to maintain L is $O(m \log n)$. Because this dominates the $O(m)$ operation count to examine each edge, the worst-case running time of S-HEAP is $O(m \log n)$, even if experimental results ([9]) indicate that it grows approximately like $O(m + n \log n)$, because usually r is a small multiple of n and considerably less than m.

S-DIAL

This algorithm, due to R.B. Dial in 1969 [19], assumes that all edge lengths are nonnegative. L is implemented as a direct-address table, a dynamic array having a number of elements equal to the maximum number of different label values. Since every finite label is equal to the length of some path with no cycles, the possible label values range in

$$[0, (n-1) \max_{(i,j)\in E} c(i,j)].$$

The entry i of L, also called a *bucket*, is a double-linked list containing each node whose label is i. The algorithm starts with the source node 1 in the bucket $L(0)$ and all other buckets are empty. At the first iteration the algorithm puts each node $(1, i)$ in the bracket $L(c(1, i))$ and then proceeds to examine the bucket $L(1)$. If $L(1)$ is nonempty, it repeats the process, removing from L each node with label 1 and moving other nodes to smaller numbered buckets, otherwise it proceeds checking bucket $L(2)$ etc.

Checking the emptiness of a bucket and inserting or removing a node from a bucket require $O(1)$ time; searching the minimum label node requires $O(n \max_{(i,j)\in E} c(i,j))$, while adjusting node labels and repositioning nodes between buckets require $O(m)$. Therefore, the running time of S-DIAL is pseudopolynomial and is given by $O(m + n \max_{(i,j)\in E} c(i,j))$.

```
Set L(0) = {1}, π(1) = 0, π(i) = ∞ ∀i ≠ 1.
Set z = 0, p_i = nil ∀i.
WHILE L ≠ ∅
    move on L until L(z) ≠ ∅;
    set i equal to the first element of L(z)
    remove i from L(z)
    FOR each outgoing edge (i, j)
        IF π(j) > π(i) + c(i, j)
            IF j ∈ L THEN remove j from L(π(j))
            set π(j) = π(i) + c(i, j), p_j = i
            insert j into L(π(j))
            ENDIF
        ENDIF
    ENDFOR
ENDWHILE
```

Pseudocode of S-DIAL

Label Correcting Methods

The label correcting methods require less sophisticated calculations to select the next node to be removed from L, but as counterpart they may involve a node more than once. All these methods implement the set L as a queue and differ in the particular type of queue they use and in the choice of the position in the queue L, where new node labels are inserted. In this section we will treat two among the most famous label correcting methods: the Bellman–Ford method and the D'Esopo–Pape method.

Bellman–Ford Method

The Bellman–Ford method is related to the method proposed by R. Bellman [3] and L.R. Ford [22]. It uses a FIFO strategy to maintain the queue L: a node is removed only from the top of the queue and is inserted at its bottom. This method proceeds in cycles of iterations: the first cycle consists of iterating on the source node 1, while in each subsequent cycle the nodes entered L during the previous cycle are removed from L in the same order that they were inserted.

The Bellman–Ford algorithm solves the SPT in the more general case in which the edge lengths can be negative and fails to terminate if and only if there exists a path starting from the source and containing a negative cycle. In the case where all cycles in the graph have

```
Set L = {1}, π(1) = 0, π(i) = ∞ ∀i ≠ 1.
Set p_i = nil, ∀i.
WHILE L ≠ ∅
    set i = top element of the queue L
    remove i from L
    FOR each outgoing edge (i, j)
        IF π(j) > π(i) + c(i, j)
            set π(j) = π(i) + c(i, j), p_j = i
            IF j ∉ L THEN
                insert j into L at the bottom
            ENDIF
        ENDIF
    ENDFOR
ENDWHILE
```

Pseudocode of the Bellman–Ford method

nonnegative length, the shortest distance of every node can be obtained after at most $n - 1$ iteration cycles. Since in each iteration cycle each edge is involved at most once and each iteration cycle requires $O(m)$ operations, the running time of this method is $O(nm)$.

D'Esopo–Pape Method

Like the Bellman–Ford method this method can be used to detect the presence of a negative cycle and like the Bellman–Ford method, a node is always removed from the top of the queue L, but it is inserted at its bottom if it has never been in L before, otherwise it is put at the top. The choice of this inserting strategy comes observing that the removal and the updating of the label of a node i affect the labels of a subset N_i of neighbor nodes j with $(i, j) \in E$. Therefore, by placing the node at the top of the queue, the labels of nodes belonging to N_i will be updated as quickly as possible.

Auction Algorithms

The *auction approach* was proposed by D. Bertsekas [4] (see also [5,6]) for solving the assignment problem. It was then generalized for the transportation problem, the minimum cost flow problem [10,11,12] and for the shortest path [7]. A complete survey of the auction algorithm can be found in [8, Chapt. 4].

To solve the problem SP, the standard forward auction algorithm follows a primal-dual approach and consists of tree basic operations: *path extension*, *path contraction* and *dual price increase*.

```
Let i be the terminal node of P.
IF π(i) < min_{(i,j)∈E}{c(i, j) + π(j)}
    THEN go to Step 1;
    ELSE go to Step 2.
ENDIF
Step 1 (CONTRACT PATH)
    Set π(i) = min_{(i,j)∈E}{c(i, j) + π(j)}.
    IF i ≠ s
        contract P and go to next iteration.
    ENDIF
Step 2 (EXTEND PATH)
    Extend P by node
    j_i = arg min_{(i,j)∈E}{c(i, j) + π(j)}.
```

The algorithm starts with a pair (P, π) satisfying CSC, (at start P may consist only of s and π may be zero), then it proceeds in iterations, transforming (P, π) into another pair satisfying CSC, that is at each iteration a dual feasible solution and a primal (infeasible) solution are available for which complementary slackness holds. Therefore, while the algorithm maintains complementary slackness, it either constructs a new primal solution (not necessarily feasible) or a new dual feasible solution, until a primal feasible (and hence also optimal) is obtained. In more detail, at each iteration the candidate path P is either extended by adding a new node at the end of the path or contracted by deleting from P the last inserted node, said terminal node. At any iteration, if no extensions or contractions are possible, the value of the dual variable corresponding to the terminal node of P is raised. The algorithm terminates when the candidate path P is extended by the target node, in case of single source-single destination problem, or when each node has been involved at least once, in case of single source-all destinations problem.

Even if the auction algorithm is introduced for solving the single source-single destination problem, it can be easily adapted to solve the SPT problem. In fact, in this case it terminates when each node has been visited at least once by the algorithm.

Graph Reduction in Auction Algorithms

The original version of the auction algorithm for the shortest path problem was modified by S. Pallottino and

M.G. Scutellá [27], who found out conditions under that it is possible to 'prune' the original graph. In more detail in [27], they showed that every time the standard forward auction algorithm reaches a node f, the optimality of the candidate path P enables to delete from the original graph all edges whose head is f, except the edge (k, f), if k is its predecessor in P. The set V of the nodes becomes in that way partitioned in the set of the nodes never visited by the algorithm and those visited at least once and that have only one incoming edge. The algorithm they have developed has a strongly polynomial computational time equal to $O(m^2)$, where m is the number of edges in the graph, without requiring that the cycle lengths must be positive, without assumptions on the topology of the graph and whatever are the input data.

Strengthening the graph reduction idea using upper bounds to the node shortest distances, they developed [13] an auction algorithm, whose computational time is $O(n \min\{m, n \log n\})$, since it deletes arcs more effectively. The set V of the nodes becomes now partitioned in three sets: the set of the nodes never visited by the algorithm, the set of those visited at least once (said *tree nodes set*) and that of the nodes never visited, but connected through an edge to at least one tree node (said *border nodes set*). The upper bounds u_j, $j \in V$, to the node shortest distances that they use, behave exactly as the temporary labels that Dijkstra's algorithm associates to each node of the graph (see e. g. [8,20,23,28]). In fact, such an upper bound u_i expresses exactly the shortest distance from the source to i and as soon as the *border node* i becomes *tree node*. Bertsekas, Pallottino and Scutellá use those upper bounds in order to 'prune' the original graph as much as possible. In fact the algorithm they developed in [13] deletes not only the incoming edges of the last visited node i as in [27], but also all the edges $(i, j) \in E$ if $u_i + c(i, j) \geq u_j$ or otherwise the edge $(k, j) \in E$ for which k is a *tree node* other than i.

Modified Version of the Standard Auction Algorithm

A modified auction algorithm (MA), due to [16], reaches a substantial computational time improvement over the standard algorithm. Its peculiar characteristic is that the CSC are not longer maintained verified during the algorithm iterations. It proceeds as the standard auction algorithm, but it does not require that the dual feasibility has to be maintained throughout the algorithm; this allows to raise the dual prices higher than in the standard algorithm and, consequently, the number of path contractions becomes substantially reduced. More precisely, the dual variable associated with the terminal node i of the candidate path P is raised to the second minimum value in the set

$$\{-(c(k, i) - \pi(k) + \pi(i), (c(i, p) - \pi(i) + \pi(p)) : (i, p) \in E\}$$

where k is the node that immediately comes before i in P. The correctness and convergence of MA are showed true through the following theoretical results, whose proofs are in [16].

Graph Collapsing In Auction Algorithms

All *graph collapsing auction algorithms* due to [15], are based on the following simple idea. When a node of the graph is visited for the first time by the auction algorithm, then the shortest path from the source to this node is found. Moreover, during the successive computations, any sub-paths extracted from an optimal path can be substituted by (collapsed to) a single arc of the same length. Suppose that at the end of the kth iteration of the auction algorithm the node i_l is visited for the first time. This means that i_l is the terminal node of the current candidate path $P = \{s = i_0, \ldots, i_l\}$. The CSC are satisfied and for the arcs belonging to P it holds that

$$\pi(i_j) = \pi(i_{j+1}) + c(i_j, i_{j+1}).$$

Due of this property of the candidate path P, from the point of view of the algorithm, i. e. with respect to the sequence of nodes visited for the first time, it is equivalent to consider the original graph or a graph where a subpath is replaced by a single arc, whose length is equal to the length of the replaced subpath. The exact meaning of this equivalence has been clarified in [15]. Here the description of the topological transformations has been intentionally left vague, because there are several different methods to realize these transformations leading to different algorithms which can perform more or less efficiently. In the next section one of such approaches, which seem particularly fruitful, will be described in full details. Because it performs also the graph

Let s and d be the labels of the source and the target node respectively.
For each $i \in E$ let FS(i) be its forward star and let P be the candidate path.
Let pred(i) be the node k such that $(k, i) \in E \cap P$.
Step 0: Set pred(i) = oldpred(i) = NIL, $\forall i \in V$.
$P = \{s\}$.
Choose $\pi \in \mathbf{R}^{|V|}$ such that $c_{lp} - \pi_l + \pi_p \geq 0, \forall (l, p) \in E$.
Step 1: Let i be the terminal node of the path P.
IF $i = d$, THEN stop, ELSE go to Step 2.
Step 2: Compute FS(i) := $\{(i, p) \in E\}$, k = pred(i).
IF $i = s$
$\gamma_1 = \min_{(s,p)\in FS(s)}\{c_{sp} + pi_p\}$
$p^* = \arg\min_{(s,p)\in FS(s)}\{c_{sp} + \pi_p\}$
IF pred(p^*) = NIL
remove from E each arc (l, p^*), $l \neq s$
oldpred(p^*) = s
pred(p^*) = s
IF $|$ FS(s) $|> 1$
$\pi(s) = \min_{(s,p)\in FS(s), p\neq p^*}\{c_{sp} + \pi_p\}$
ELSE $\pi(s) = \gamma_1$
$i = p^*$ and go to Step 1.
ELSE // case $i \neq s$ //
IF FS(i) = $\emptyset$ // Construction due to FS(i) = $\emptyset$ //
remove from E each arc (l, i)
i = pred(i) and go to Step 1.
ELSE // case FS(i) $\neq \emptyset$ //
k = pred(i), in = $\pi(k) - c_{ki}$,
$\gamma_1 = \min_{(i,p)\in FS(i)}\{c_{ip} + \pi_p\}$
IF in $< \gamma_1$ // Normal contraction //
$\pi(i) = \gamma_1$, $i = k$ and go to Step 1.
ELSE
$p^* = \arg\min_{(i,p)\in FS(i)}\{c_{ip} + \pi_p\}$
$\gamma_2 = \min_{(i,p)\in FS(i), p\neq p^*}\{c_{ip} + \pi_p\}$
$\pi(i) = \gamma_2$
IF in $> \gamma_2$ // Normal extension //
IF pred(p^*) = NIL
remove from E each arc (l, p^*), $l \neq i$
oldpred(p^*) = i
pred(p^*) = i
ELSE // Graph collapsing extension //
add the arc (k, p^*) to the set E
set $c_{kp^*} = c_{ki} + c_{ip^*}$
IF pred(p^*) = NIL
remove from E each arc (l, p^*)
oldpred(p^*) = i
ELSE
remove from E the arc (i, p^*)
pred(p^*) = k
$i = p^*$ and go to Step 1.

Pseudocode of the algorithm GCA2 solving the single source-single destination problem

'pruning' operation proposed in [27], it returns a solution under the same assumptions made in [27], that is without the requirement of positive cycle lengths, without assumptions on the topology of the graph and for any kind of input data.

The main idea of graph reduction seems to be similar to that introduced in [13], because the effectiveness of both belongs to the reduction of the number of iterations (nodes visiting), but the approaches are somewhat different. In fact, while their method uses several criteria to delete those edges that certainly will not belong to a shortest path before passing over them, the approach in [15] deletes edges belonging to shortest paths, replacing a chain of them with a single edge.

Graph Collapsing Auction Algorithm

In this section is described an auction algorithm (GCA2), due to [15], that applies fruitfully the graph collapsing concept to the modified version of the auction algorithm (MA). Note that, because during the computation MA does not maintain verified the CSC along the current candidate path, it is no longer straightforward to implement the substitution of a piece of the path with only one arc. In GCA2 this substitution is realized during the extension step, only when the second minimum value computed during the price updating phase of MA results from the incoming arc on the current node, because only in this case the CSC are verified as equality.

Besides applying the graph collapsing concept to the modified auction algorithm, GCA2 uses the dual prices updating idea even when a graph collapsing occurs. In fact, in step 2 of GCA2, when a graph collapsing occurs, a certain amount of computational effort is achieved updating the price of the current node through γ_2, the third minimum value, which after the collapsing becomes the second minimum value. In [15] it is shown that the computational complexity of GCA2 is not worst than that of MA, which is no worse than that of the Pallottino's algorithm [27].

Virtual Source Concept Applied in Auction Algorithms

In [14] two algorithms were proposed having complexity $O(n^2)$ and in which the computational effort is reduced fully exploiting the below described property of the auction algorithm: when it reaches for the first time a node, said current node, the shortest path from the source to current node is found. Hence, it is obvious that all the subtrees rooted at the current node can be computed applying the algorithm from a *virtual source* located on the current node itself, and that the complete shortest path tree can be assembled joining pieces of optimal subpaths so obtained.

In order to make well defined the joining operation, the algorithm associates a label to each virtual source. In this aspect it resembles Dijkstra's algorithm [20], but the order on which new nodes are explored remains equal to that of the auction algorithm. The *virtual source algorithm* actually combines the good characteristics of both the approaches.

The first version of the virtual source algorithm (VSA1) evolves as the standard auction algorithm with

Step 0: choose $\pi \in \mathbf{R}^{|V|}$ such that
$c(l, p) - \pi(l) + \pi(p) \geq 0, \forall (l, p) \in E.$
$\text{pred}(i) = \text{NIL}$ and $d(i) = +\infty$ for each $i \in V$
sort FS(s) in nondecreasing order
$w(s) = \pi(s) = \text{SELECT.MIN FS}(s))$
$w(l) = \pi(l)$ for each $l \in V, l \neq s$
$E = E \setminus \{(l, s), (g, j_s) : l, g \in V\}$
$Q = \{s\}$, $i = s$ and go to Step 1.
Step 1: IF $\mid Q \mid = \mid V \mid$ OR $w(q) = +\infty, \forall q \in Q$,
THEN stop.
IF $\pi(i) = \text{SELECT.MIN FS}(i))$ go to Step 2
ELSE go to Step 3.
Step 2: Let $j_i = \arg\min_{(i,p)\in \text{FS}(i)}\{c(i, p) + \pi(p)\}$
sort FS(j_i) in nondecreasing order
$\text{pred}(j_i) = i, d(j_i) = w(i)$
$\pi(j_i) = \text{SELECT.MIN FS}(j_i))$
$w(j_i) = w(i) + \pi(j_i)$
$E = E \setminus \{(l, j_i) : l \in V\}$
INSERT(Q, j_i) and go to Step 3.
Step 3: IF $\pi(j) = +\infty, \forall j \in \text{FS}(i)$ OR $\text{FS}(i) = \emptyset$
$w(i) = \pi(i) = +\infty$
ELSE
$\pi(i)^{\text{old}} = \pi(i)$
$\pi(i) = \text{SELECT.MIN FS}(i))$
$w(i) = w_k(i) + (\pi(i) - \pi(i)^{\text{old}})$.
UPDATE(Q, i)
$i = \text{SELECT.MIN}Q)$ and go to Step 1.

Pseudocode for VSA2

two differences. First of all the algorithm maintains a special list Q of those nodes that during the computation become virtual sources. Moreover, the contraction phase is modified: every time it becomes needed to perform a contraction on a node i, the algorithm updates the value of a parameter $w(i)$, called *weight* associated to i, which is inserted in Q if it does not belong to Q yet. The next terminal node is that corresponding to the actual minimum weight on Q. During the iterations of the algorithm the list Q is maintained sorted in nondecreasing order of weights. The crucial characteristic of VSA1 is the way in which the weights of the nodes are updated. At any iteration the weight associated to each node i expresses the shortest distance from the source to the node i itself added to the actual minimum value of the function $c + \pi$ on the set of edges outgoing from i.

Even if VSA1 is interesting from a theoretical point of view, it can be easily furthermore improved to completely eliminate not only the contraction phases, as performed by VSA1, but also extension phases. The resulting algorithm, called VSA2, performs only insertions of nodes in Q and updating of Q. Since any extension of the auction algorithms is followed by a contraction and since VSA1 creates a virtual source during a contraction phase, the authors thought to anticipate the virtual source placement on a node as soon as that node is discovered by the algorithm. A relative pseudocode is given above.

A New Virtual Source Algorithm

In the algorithm VSA2 described in the previous section both the contraction and the extension phases are completely eliminated. The computational time depends exclusively on how efficiently can be performed the dictionary operations of insertion and updating of the data structure chosen to represent Q, the set containing the virtual sources.

In order to improve the performance of VSA2, in which Q is implemented as a queue, in the new proposed algorithm the virtual sources weights are maintained in some sorted fashion, using the property that VSA2 assigns nondecreasing weights. The basic idea is similar to that realized by Dial, who chose a direct-address table as data structure. Nevertheless, given the bounds of the memory available on a typical computer, storing a direct-address table of size equal to the maximum length of a path is impractical, or even impossible, if the number of nodes and/or the maximum arc cost are large. In the new proposed algorithm, instead of using the virtual sources weights as an array index directly, the array index is computed from the weight. The data structure resulting is called hash table. Since it typically uses an array of size proportional to the number of elements actually stored, an hash table is more effective than the direct addressing technique, when, as in our case, the number of elements actually stored is small compared to the size of the direct-address table. In fact, while in a direct-address table a node having label k is directly stored in the slot k, with hashing the virtual source having weight k is stored in the slot $g(k)$, where if n is the number of nodes of the graph and $l = (n-1)\max_{(i,j)\in E} c(i,j)$

$$g\colon\ U = \{0,\ldots,l\} \to \{0,\ldots,m-1\}$$

is a function that maps U into the slots $V[0,\ldots,m-1]$. The efficiency of an hash table depends on the choice of the hash function g. An hash function is 'good' if it is such that each element is almost equally likely to hash to any of the m slots. The most popular techniques for designing a good hash function are hashing by division, hashing by multiplication and universal hashing. In practice, heuristic techniques can be also used to create hash functions that are likely to perform well. For a detailed analysis of these techniques, see [18].

The hash function allocated in the new proposed algorithm has been designed following the division method, in which for creating hash functions a key k is mapped into one of m slots by taking the remainder of k divided by m. Formally, the general hash function is the following:

$$g(k) = k \mod m.$$

In the new virtual source algorithm m has been chosen in order to process at most 10 elements for each unsuccessful search.

Even if this line of research is still being investigated, the hash function g above defined has already led to satisfactory results, as discussed in the next section dedicated to the computational results.

Computational Results

Detailed results are reported in [15,21], and [14]. Here, we briefly describe the results obtained comparing two

classes of algorithms that seem to be the most competitive: the auction class, from that GCA algorithms descend, and the class of Dijkstra-like algorithms.

In [15,21], and [14] we have considered as representative of the first class the forward modified auction (MA) [16], that in some preliminary performed tests has been selected as the faster one over the all competitors in all the instances. For the second class we have chosen the forward S-HEAP and the forward S-DIAL of [24] because they are widely known state of the art algorithms and because all the implementations of Dijkstra's forward algorithm often behaves similarly [17].

The experiments were conducted on a Digital Alpha 4100 running Digital UNIX V4.0B. All the auction programs were written in C and compiled with gcc ver. 2.7.2, while for the S-HEAP and S-DIAL algorithms we used the Fortran implementation due to [25].

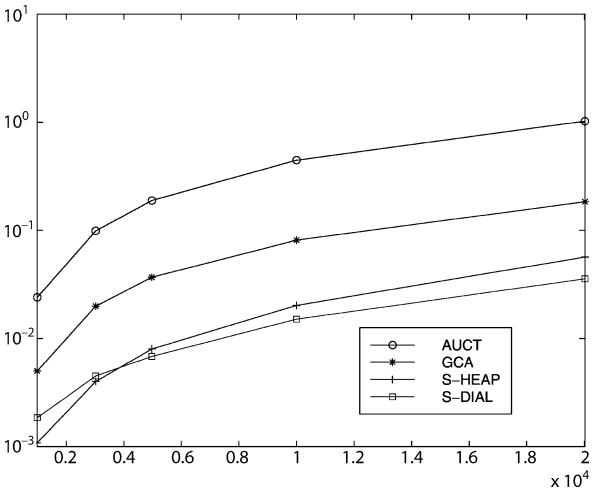
To cover most of the practically encountered instances different kinds of problems were studied; for any family of problems different sizes were considered and for any size we recorded running times employed by any algorithm averaged over ten different random generated instances.

The types of problems taken into account were: square and long grid networks, generated using the GRIDGEN code by Y. Lee and J.B. Orlin and networks generated using the NETGEN code [26] with different densities. For any of such networks we have solved the SPT problem.

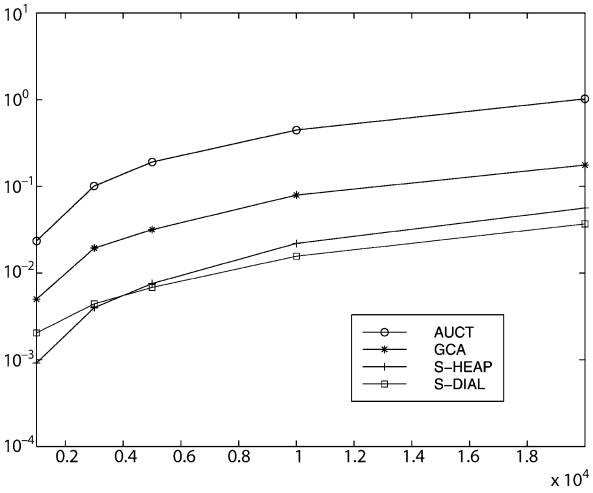
The following three groups of graphs are considered:

1) Square and long grid-like graphs, generated by using the GRIDGEN code, whose node number n varies from 1000 to 20000. The source is always placed at one end of the grids, so the diameter is large. All grids are planar and square with average degree for any node equal to 4. The length/height ratio is approximatively 30.
2) General random networks generated by using the NETGEN code [26]. Even in this case n ranges from 1000 to 20000, using as arc number m both $4n$ and $10n$.
3) Complete networks, whose nodes number n ranges from 100 to 500.

In all cases the arc lengths are randomly chosen in the range 0–10000; for any of them we solved the SPP from the node labelled 1 by the network generator to all the other nodes. In all instances GCA2 has outperformed MA.



Shortest Path Tree Algorithms, Figure 1
Mean computational time on square grids

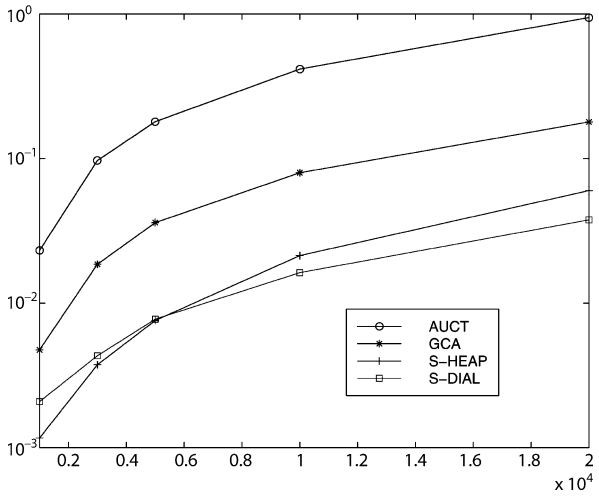


Shortest Path Tree Algorithms, Figure 2
Mean computational time on elongated grids

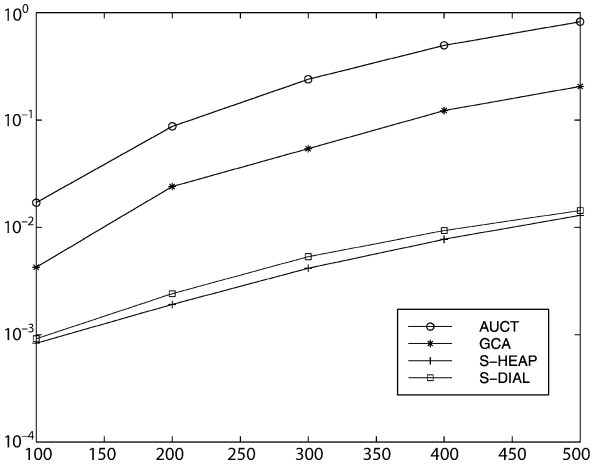
The figures show the mean computational time of the tested algorithms in seconds and in log. scale.

MA and GCA2 have analogous behavior, but in each instance GCA2 leads to a computational time saving at least of 50% over MA.

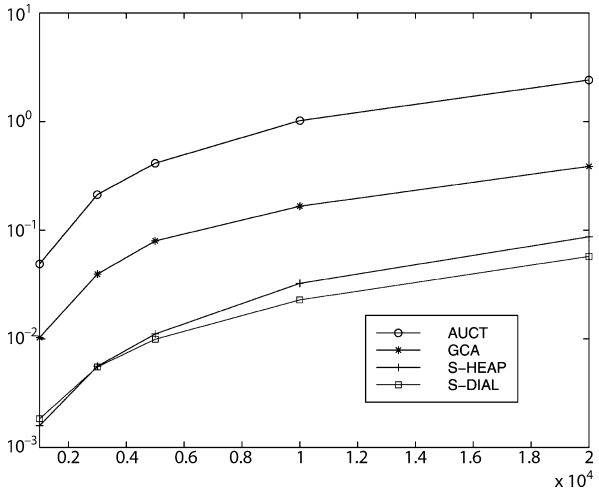
Recently (as of 1999), we have realized a new implementation of the algorithm VSA2 in which the set con-



Shortest Path Tree Algorithms, Figure 3
Mean computational time on sparse graphs



Shortest Path Tree Algorithms, Figure 5
Mean computational time on complete graphs



Shortest Path Tree Algorithms, Figure 4
Mean computational time on dense graphs

taining the virtual sources is an hash table. Even if the worst-case complexity has been not improved, the new algorithm reduces in practice the computational time of VSA2. Even though the testing phase of this new algorithm is at the moment limited to general dense random graphs, analyzing the results obtained, it seems to be competitive with the Dijkstra-like algorithms.

Conclusions

This article is a brief survey of the most popular algorithms and of some novel approaches for solving shortest path problems. The new methods proposed are variations of the auction technique. They are based on topological transformations of the graph, operated during the iteration of the algorithms. The main idea is based on the property of any auction algorithm for the shortest path that if a node is included at a certain step in the candidate path, then the shortest path to this node is found. Different realizations of such approaches are described leading to different algorithms: graph collapsing algorithms and virtual sources algorithms. The graph collapsing algorithms presented are two. The former one is the straight application of the idea to the standard auction algorithm of Bertsekas improved in [27], while the latter applies the same concept to the modified auction of [16]. Strengthening the peculiar characteristics of the standard auction method, it has been developed the family of virtual source algorithms.

For all the algorithms an upper bound on the total number of required operations is found. An extensive set of numerical test has been carried out and looking the results it is possible to conclude that the algorithm GCA1 has only a theoretical relevance. GCA2, instead, has been revealed in all cases much more efficient. It seems to be one of the better algorithms for solving the shortest path problem and extends the applicability of the auction approach. GCA2 fills the performance gap between them and the better Dijkstra-like algorithms,

while the most recent virtual source algorithm seems to completely eliminate it.

See also

- ▶ Auction Algorithms
- ▶ Bottleneck Steiner Tree Problems
- ▶ Capacitated Minimum Spanning Trees
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Maximum Flow Problem
- ▶ Minimax Game Tree Searching
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms and applications. Prentice-Hall, Englewood Cliffs, NJ
2. Becker A, Geiger D (1996) Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. Artif Intell 83:167–188
3. Bellman R (1957) Dynamic programming. Princeton Univ. Press, Princeton
4. Bertsekas D (1979) A distributed algorithm for the assignment problem. Working Paper Lab. Inform. and Decision Syst. MIT
5. Bertsekas D (1985) A distributed asynchronous relaxation algorithm for the Assignment Problem. In: 24th IEEE Conf. Decision and Control, Ft. Lauderdale, FL, pp 1703–1704
6. Bertsekas D (1988) The auction algorithm: A distributed relaxation method for the assignment problems. Ann Oper Res 14:105–123
7. Bertsekas D (1991) The auction algorithm for shortest paths. SIAM J Optim 1:425–447
8. Bertsekas D (1991) Linear networks optimization: Algorithms and codes. MIT, Cambridge, MA
9. Bertsekas DP (1998) Network optimization: Continuous and discrete models. Athena Sci., Belmont, MA
10. Bertsekas D, Castanon DA (1989) The auction algorithm for minimum cost network flow problem. Report Lab. Information and Decision Systems MIT P-1925
11. Bertsekas D, Castanon DA (1989) The auction algorithm for transportation problems. Ann Oper Res 20:67–96
12. Bertsekas D, Castanon DA (1991) A generic auction algorithm for the minimum cost network flow problem. Report Lab. For Information and Decision Systems MIT P-2084
13. Bertsekas D, Pallottino S, Scutellá MG (1995) Polynomial auction algorithms for shortest paths. Comput Optim Appl 4:99–125
14. Cerulli R, Festa P, Raiconi G (1997) An efficient auction algorithm for the shortest path problem using virtual source concept. Techn. Report D.I.A.R.M. Capocelli Univ. Salerno 7
15. Cerulli R, Festa P, Raiconi G (2000) Graph collapsing in auction algorithms. Techn. Report D.I.A.R.M. Capocelli Univ. Salerno 6/97, to appear in: Computational Optimization and Application
16. Cerulli R, Leone R De, Piacente G (1994) A modified auction algorithm for the shortest path problem. Optim Methods Softw 4:209–224
17. Chernassky BV, Goldberg AV, Radzik T (1996) Shortest path algorithms: Theory and experimental evaluation. Math Program 73:129–174
18. Cormen TH, Leiserson CE, Rivest RL (1990) Introduction to algorithms. MIT, Cambridge, MA
19. Dial RB (1969) Algorithm 360: Shortest path forest with topological ordering. Comm ACM 12:632–633
20. Dijkstra E (1959) A note on two problems in connexion with graphs. Numer Math 1:269–271
21. Festa P, Cerulli R, Raiconi G (1999) Complexity and experimental evaluation of primal-dual shortest path tree algorithms. In: Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems. Kluwer, Dordrecht
22. Ford LR Jr (1956) Network flow theory. Report The Rand Corp. P-923
23. Gallo G, Pallottino S (1986) Shortest path methods: A unified approach. Math Program Stud 26:38–64
24. Gallo G, Pallottino S (1988) Shortest path methods. Ann Oper Res 13:3–79
25. Gallo G, Pallottino S, Ruggeri C, Storchi G (1984) Metodi ed algoritmi per la determinazione di cammini minimi. Monografie di Software Mat. 29
26. Klingman D, Napier A, Stutz J (1974) NETGEN - A program for generating large scale (un) capacitated assignment, transportation, and minimum cost flow network problems. Managem Sci 20:814–822
27. Pallottino S, Scutellá MG (1991) Strongly polynomial auction algorithms for shortest path. Ricerca Oper 21:60–60
28. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: Algorithms and complexity. Prentice-Hall, Englewood Cliffs, NJ

Short-Term Scheduling of Batch Processes with Resources

STACY L. JANAK, CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering,
Princeton University,
Princeton, USA

Article Outline

Introduction
  Conceptual Model Enhancements:
  Splitting of the Tasks
Formulation
  Allocation Constraints
  Capacity Constraints
  Batch-Size Matching Constraints: Processing Tasks
  Batch-Size Matching Constraints: Utility Tasks
  Material Balances
  Duration Constraints
  Sequence Constraints: Processing Tasks
  Sequence Constraints: Storage Tasks
  Sequence Constraints: Utility-Related Tasks
  Order Satisfaction Constraints
  Bound Constraints
  Objective Function
Methods
  Number of Event Points
  Tightening Constraints
  Sequential Processes
Cases
  Case 1: Resource Constraints, Mixed Storage Policies,
    Variable Batch Sizes, and Processing Times
  Case 2: Sequential Process
    with Order-Dependent Processing Times
Conclusions
References

Introduction

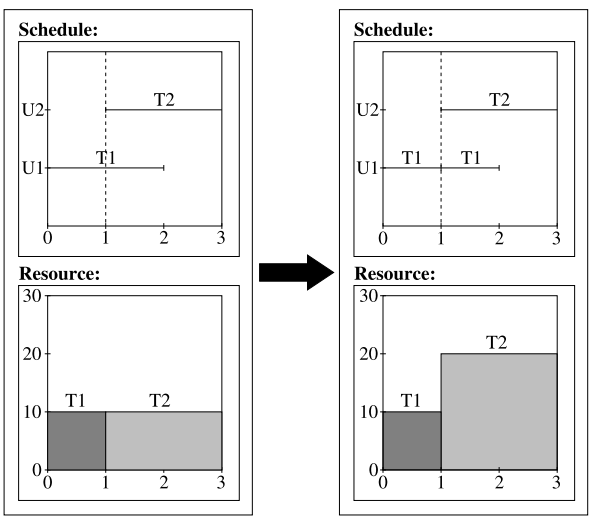
In this chapter, we propose an enhanced state-task network (STN) mixed-integer linear programming (MILP) model for the short-term scheduling of multiproduct and multipurpose batch plants with intermediate due dates. The proposed approach extends the continuous-time scheduling model which was originally developed by Floudas and coworkers [6,7,8,10]. This enhanced formulation is able to account for limited, renewable resources, various storage policies, including unlimited intermediate storage (UIS), finite intermediate storage (FIS), no intermediate storage (NIS), and zero-wait (ZW) conditions, and incorporates several additional features, including variable batch sizes and processing times, batch mixing and splitting, and sequence-dependent changeover times. The enhanced formulation still utilizes a continuous-time representation employing a necessary number of event points of unknown location corresponding to the activation of a task. However, tasks are allowed to continue over several, consecutive event points, enabling resource and storage quantities to be correctly determined at each task activation. The full mathematical model and additional computational results can be found in [9].

There are several other models in the scheduling literature which are capable of accounting for resource constraints as well as mixed storage policies in short-term scheduling problems. Maravelias and Grossmann [11] developed a global event based continuous-time MILP model which utilizes the STN approach and addresses the general problem of batch scheduling, including resource constraints, variable batch sizes and processing times, various storage policies, batch mixing and splitting, and sequence-dependent changeover times. Their model utilizes the idea of task decoupling, eliminates binaries for unit assignment and the continuous variables for start times of tasks, proposes a new class of valid tightening inequalities, and was the first general STN-based model capable of handling resource considerations. However, owing to the continuous-time representation used, which is common for all units, their formulation always requires an extra event point for the end of the last task, generating larger and more complex models than the proposed formulation. In addition, Castro et al. [2] presented a general, continuous-time MILP model for scheduling of batch processes based on the resource-task network (RTN) representation. It is uses a global event based representation of time and treats all types of resources in a unified way so that no special sequencing constraints are required. The authors claim that it is a simpler and less degenerate mathematical model than other RTN continuous-time formulations, such as the model of Schilling [13]. In later work, Castro et al. [3] extended their RTN formulation to consider continuous processing tasks, better constraints to handle ZW conditions for batch tasks, and tighter timing constraints. The authors also extended their formulation to deal with sequence-dependent cleaning tasks.

Conceptual Model Enhancements: Splitting of the Tasks

As previously mentioned, the proposed formulation allows all tasks to continue over several consecutive event points. This change was implemented so that the amount of resource utilized by a task is correctly determined in relationship to all other tasks which utilize the same resource at all instances of time. For example, consider the production schedule and associated resource utilization given in Fig. 1 for a process which has two tasks that utilize the same resource. Both tasks (T1 and T2) have a processing time of 1 and utilize ten units of the resource with each batch. The schedule on the left, which is determined with the original scheduling model, does not recognize that task T1 is active when task T2 begins. At time 2 when task T2 starts, it does not see that task T1 is active and thus determines that only ten units of the resource are currently being used instead of 20. However, if you look at the images on the right, which were generated with the proposed formulation, task T1 is split over two consecutive event points. Thus, when task T2 starts at time 1, task T1 is active at that time point and the calculated resource utilization is correct.

Note that splitting of processing tasks in our continuous-time model is also necessary to account for several other features inherent in scheduling problems in addition to limited, renewable resources. For instance, in order to model FIS for a state that can be produced or consumed by more than one task, it may be necessary to allow the tasks to continue over more than one event point. If the tasks have different processing times or do not start and/or finish at the same time, then task splitting will have to be incorporated to ensure that storage limits are maintained. Also, for STNs that employ recycle loops, the intermediate state recycled in the loop is consumed by a task that occurs earlier in the STN than the task that produces the recycled state, creating a complicated time dependence between the two tasks that must be maintained in order to avoid violating material balances. If these related tasks have different processing times, it may be necessary to allow task splitting for the task(s) with longer processing times in order to determine the best possible solutions.



Short-Term Scheduling of Batch Processes with Resources, Figure 1
Task splitting for resource utilization

As a consequence of tasks extending over multiple event points, each processing task must have two sets of binary variables and one set of continuous variables associated with it. The binary variable $ws(i, n)$ indicates that a task i starts at event point n, while the binary variable,$wf(i, n)$ indicates that a task i ends at event point n. In addition, the continuous variable $w(i, n)$ indicates that a task i is active at event point n, regardless of whether the task is starting, finishing, or just processing at that event point. In response to this change, the enhanced model has an expanded set of constraints in order to accurately keep track of the utilization of units as well as the timing and sequence of tasks that have been split. In addition, two new sets of tasks are introduced into the mathematical model. One set, (i^{st}), represents the storage of intermediate states, while the other set, (u), gives the utilization of a resource. New constraints are then introduced into the enhanced model in order to relate the timing and sequence of these new tasks with their associated processing tasks so that specified limits can be enforced.

Formulation

The proposed formulation requires the following indices, sets, parameters, and variables:

Indices:

i processing tasks;
i^{st} storage tasks;
j units;

k orders;
n event points representing the beginning of a task;
s states;
u utilities;

Sets:

I processing tasks;
I_s^{st} storage tasks for state s;
I_j tasks which can be performed in unit j;
I_k tasks which process order k;
I^{ps} tasks which are processing or storing;
I_s^{p} tasks which produce state s;
I_s^{c} tasks which consume state s;
I_u tasks which consume utility u;
J units;
J_i units which are suitable for performing task i;
K orders;
K_i orders which are processed by task i;
K_s orders which produce state s;
N event points within the time horizon;
S states;
S_f states with finite intermediate storage;
S_n states with no intermediate storage;
S_p states which are final products;
S_r states which are raw materials;
S_z states with ZW constraint;
U utilities;

Parameters:

α_{ij} constant term of processing time of task i in unit j;
β_{ij} variable term of processing time of task i in unit j;
δ_{iu} variable term of consumption of utility u by task i;
γ_{iu} constant term of consumption of utility u by task i;
ρ_{is} proportion of state s produced that is consumed by task i;
am_k amount of order k;
av_u maximum availability of utility u;
$cap_{ij}^{\max}$ maximum capacity for task i in unit j;
$cap_{ij}^{\min}$ minimum capacity for task i in unit j;
cap_s^{st} capacity of storage for state s;
dem_s demand of state s;
due_k due date of order k;
H time horizon;
$price_s$ price of state s;
ST_s^0 initial available amount of state s;
$ST_s^{\max}$ maximum amount of state s;

Continuous variables:

$B(i, j, n)$ amount of material undertaking task i in unit j at event point n;
$B^{\mathrm{s}}(i, j, n)$ amount of material starting processing at event point n;
$B^{\mathrm{f}}(i, j, n)$ amount of material finishing processing at event point n;
$BU(i, u, n)$ amount of utility u consumed by task i at event point n;
$B_{\mathrm{ut}}(u, n)$ remaining level of utility u at event point n;
$B_{\mathrm{st}}(i^{\mathrm{st}}, n)$ amount of material stored by storage task i^{st} at event point n;
$D(s, n)$ amount of state s delivered at event point n;
MS makespan;
$ST(s, n)$ amount of state s at event point n;
$STF(s)$ final amount of state s at the end of the time horizon;
$STO(s)$ initial amount of state s at the beginning of the time horizon;
$T^{\mathrm{s}}(i, j, n)$ time at which task i starts in unit j at event point n;
$T^{\mathrm{f}}(i, j, n)$ time at which task i finishes in unit j at event point n;
$T_{\mathrm{st}}^{s}(i^{\mathrm{st}}, n)$ time at which storage task i^{st} starts at event point n;
$T_{\mathrm{st}}^{\mathrm{f}}(i^{\mathrm{st}}, n)$ time at which storage task i^{st} finishes at event point n;
$T_{\mathrm{ut}}^{s}(u, n)$ starting time of a change in utility u at event point n;
$T_{\mathrm{ut}}^{\mathrm{f}}(u, n)$ finishing time of a change in utility u at event point n;
$TT^{\mathrm{s}}(j, n)$ starting time of the active task in unit j at event point n;
$TT^{\mathrm{f}}(j, n)$ finishing time of the active task in unit j at event point n;
$w(i, n)$ indicates if task i is activated at event point n;

Binary variables:

$ws(i, n)$ assigns the beginning of task i at event point n;
$wf(i, n)$ assigns the ending of task i at event point n;
$y(k, i, n)$ assigns the delivery of order k through task i at event point n.

On the basis of this notation, the mathematical model for the short-term scheduling of batch plants with mixed storage policy and resource constraints involves the following constraints.

Allocation Constraints

$$\sum_{i \in I_j} w(i,n) \le 1, \quad \forall j \in J, \quad n \in N \tag{1}$$

$$w(i,n) = \sum_{n' \le n} ws(i,n') - \sum_{n' < n} wf(i,n'), \quad \forall i \in I, n \in N \tag{2}$$

$$\sum_{n \in N} ws(i,n) = \sum_{n \in N} wf(i,n), \quad \forall i \in I \tag{3}$$

$$ws(i,n) \le 1 - \sum_{n' < n} ws(i,n') + \sum_{n' < n} wf(i,n'), \quad \forall i \in I, \quad n \in N \tag{4}$$

$$wf(i,n) \le \sum_{n' \le n} ws(i,n') - \sum_{n' < n} wf(i,n'), \quad \forall i \in I, \quad n \in N \tag{5}$$

These constraints express the requirement that for each unit j and at each event point n, only one of the tasks that can be performed in this unit (i. e., $i \in I_j$) should take place. Constraints (2) relate the continuous variable $w(i,n)$ to the binary variables $ws(i,n)$ and $wf(i,n)$ so that $w(i,n)$ will take on a value of 1 if task i is activated at event point n. Constraints (3) require that each processing task i must both start and finish during the time horizon. Constraints (4) require that processing task i cannot start at event point n if it has started at an earlier event point n' and has not finished by event point n. Constraints (5) require that processing task i cannot finish at event point n unless is has started at an earlier event point n' and has not finished by event point n.

Capacity Constraints

$$cap_{ij}^{\min} \cdot w(i,n) \le B(i,j,n) \le cap_{ij}^{\max} \cdot w(i,n), \quad \forall i \in I, \quad j \in J_i, \quad n \in N \tag{6}$$

$$B_{\mathrm{st}}(i^{\mathrm{st}}, n) \le cap_s^{\mathrm{st}}, \quad \forall i^{\mathrm{st}} \in I_s^{\mathrm{st}}, \quad n \in N \tag{7}$$

These first set of constraints express the requirement for the batch size of a processing task i at a unit j, $B(i,j,n)$, to be greater than the minimum amount of material, $cap_{ij}^{\min}$, and less than the maximum amount of material, $cap_{ij}^{\max}$, that can be processed by task i in unit j. Constraints (7) represent the maximum available storage capacity for each storage task i^{st} at each event point n.

Batch-Size Matching Constraints: Processing Tasks

$$B(i,j,n) \le B(i,j,n-1) + cap_{ij}^{\max}\left[1 - w(i,n-1) + wf(i,n-1)\right], \quad \forall i \in I, \quad j \in J_i, \quad n \in N, \ n > 1 \tag{8}$$

$$B(i,j,n) \ge B(i,j,n-1) - cap_{ij}^{\max}\left[1 - w(i,n-1) + wf(i,n-1)\right], \quad \forall i \in I, \quad j \in J_i, \quad n \in N, \ n > 1 \tag{9}$$

$$B^{\mathrm{s}}(i,j,n) \le B(i,j,n), \quad \forall i \in I, \quad j \in J_i, \quad n \in N \tag{10}$$

$$B^{\mathrm{s}}(i,j,n) \le cap_{ij}^{\max} \cdot ws(i,n), \quad \forall i \in I, \quad j \in J_i, \quad n \in N \tag{11}$$

$$B^{\mathrm{s}}(i,j,n) \ge B(i,j,n) - cap_{ij}^{\max}[1 - ws(i,n)], \quad \forall i \in I, \quad j \in J_i, \quad n \in N \tag{12}$$

$$B^{\mathrm{f}}(i,j,n) \le B(i,j,n), \quad \forall i \in I, \quad j \in J_i, \quad n \in N \tag{13}$$

$$B^{\mathrm{f}}(i,j,n) \le cap_{ij}^{\max} \cdot wf(i,n), \quad \forall i \in I, \quad j \in J_i, \quad n \in N \tag{14}$$

$$B^{\mathrm{f}}(i,j,n) \ge B(i,j,n) - cap_{ij}^{\max}[1 - wf(i,n)], \quad \forall i \in I, \quad j \in J_i, \quad n \in N \tag{15}$$

Constraints (8) and (9) represent the relationship between the batch size of task i in unit j at two consecutive event points $n-1$ and n. These constraints are required because tasks can extend over several event points and the batch sizes at these consecutive event points must be consistent. Constraints (10)–(12) relate the variables $B(i,j,n)$ and $B^{\mathrm{s}}(i,j,n)$, where $B^{\mathrm{s}}(i,j,n)$ is the amount of material that is starting to be processed at event point n. Similar to the previous set of constraints, constraints (13)–(15) relate the variables

$B(i,j,n)$ and $B^{\mathrm{f}}(i,j,n)$, where $B^{\mathrm{f}}(i,j,n)$ is amount of material that is finishing being processed at event point n.

Batch-Size Matching Constraints: Utility Tasks

$$BU(i,u,n) = \gamma_{iu} w(i,n) + \eta_{iu} B(i,j,n), \quad \forall u \in U, \quad i \in I_u, \quad j \in J_i, \quad n \in N \tag{16}$$

$$\sum_{i \in I_u} BU(i,u,n) + B_{\mathrm{ut}}(u,n) = \sum_{i \in I_u} BU(i,u,n-1) + B_{\mathrm{ut}}(u,n-1), \quad \forall u \in U, \quad n \in N, \ n > 1 \tag{17}$$

$$\sum_{i \in I_u} BU(i,u,n) + B_{\mathrm{ut}}(u,n) = av_u, \quad \forall u \in U, \quad n \in N, \ n = 1 \tag{18}$$

where γ_{iu} and η_{iu} are the constant and variable terms of the amount of utility u consumed by task i in unit j at event point n, $BU(i,u,n)$ is the amount of utility u consumed by task i at event point n, and $B_{\mathrm{ut}}(u,n)$ is the amount of utility u available at event point n. Constraints (16) represent the amount of utility required by the unit to process $B(i,j,n)$ of material while performing task i. Constraints (17) express the mass balance on the utilities, requiring that the amount of utility at event point n is equal to the amount of utility at event point $n-1$. Constraints (18) express the requirement that the amount of utility u at the first event point, including the amount available, $B_{\mathrm{ut}}(u,n)$, and the amount consumed, $\sum_{i \in I_u} BU(i,u,n)$, must be equal to the original amount of utility u available, av_u.

Material Balances

$$ST(s,n) = ST(s,n-1) - D(s,n) + \sum_{i \in I_s^{\mathrm{p}}} \rho_{is} \sum_{j \in J_i} B^{\mathrm{f}}(i,j,n-1) + \sum_{i \in I_s^{\mathrm{c}}} \rho_{is} \sum_{j \in J_i} B^{\mathrm{s}}(i,j,n) + \sum_{i^{\mathrm{st}} \in I_s^{\mathrm{st}}} B_{\mathrm{st}}(i^{\mathrm{st}},n-1) - \sum_{i^{\mathrm{st}} \in I_s^{\mathrm{st}}} B_{\mathrm{st}}(i^{\mathrm{st}},n), \quad \forall s \in S, \quad n \in N, \ n > 1 \tag{19}$$

$$ST(s,n) = STO(s) + \sum_{i \in I_s^{\mathrm{c}}} \rho_{is} \sum_{j \in J_i} B^{\mathrm{s}}(i,j,n) - \sum_{i^{\mathrm{st}} \in I_s^{\mathrm{st}}} B_{\mathrm{st}}(i^{\mathrm{st}},n), \quad \forall s \in S, \quad n \in N, \ n = 1 \tag{20}$$

$$STF(s) = ST(s,n) - D(s,n) + \sum_{i \in I_s^{\mathrm{p}}} \rho_{is} \sum_{j \in J_i} B^{\mathrm{f}}(i,j,n) + \sum_{i^{\mathrm{st}} \in I_s^{\mathrm{st}}} B_{\mathrm{st}}(i^{\mathrm{st}},n), \quad \forall s \in S, \quad n \in N, \ n = N \tag{21}$$

According to constraints (19), the amount of material of state s at event point n is equal to that at event point $n-1$ increased by any amounts produced or stored at event point $n-1$, decreased by any amounts consumed or stored at event point n, and decreased by the amount required by the market at event point n, $D(s,n)$. Constraints (20) and (21) represent the material balance on state s at the first and last event points, respectively. The amount of state (s) at the first event point is equal to the initial amount, $STO(s)$, decreased by any amounts consumed or stored at the first event point. The total amount of state s at the end of last event point, $STF(s)$, is equal to the amount at the beginning of the last event point, $ST(s,n)$, increased by any amounts produced or stored at the last event point and decreased by the amount required by the market at the last event point.

Duration Constraints

$$T^{\mathrm{f}}(i,j,n) \geq T^{\mathrm{s}}(i,j,n), \quad \forall i \in I, \quad j \in J_i, \quad n \in N \tag{22}$$

$$T^{\mathrm{f}}(i,j,n) \leq T^{\mathrm{s}}(i,j,n) + H \cdot w(i,n), \quad \forall i \in I, \quad j \in J_i, \quad n \in N \tag{23}$$

$$T^{\mathrm{s}}(i,j,n) \leq T^{\mathrm{f}}(i,j,n-1) + H[1 - w(i,n-1) + wf(i,n-1)], \quad \forall i \in I, \quad j \in J_i, \quad n \in N, \ n > 1 \tag{24}$$

$$T^{\mathrm{f}}(i,j,n') - T^{\mathrm{s}}(i,j,n) \geq \alpha_{ij} ws(i,n) + \beta_{ij} B^{\mathrm{s}}(i,j,n) - H[1 - ws(i,n)] - H[1 - wf(i,n')] - H[\sum_{n \leq n'' < n'} wf(i,n'')], \quad \forall i \in I, \quad j \in J_i, \quad n \in N, \quad n' \in N, \ n \leq n' \tag{25}$$

$$T^{\mathrm{f}}(i,j,n') - T^{\mathrm{s}}(i,j,n) \le \alpha_{ij} ws(i,n) + \beta_{ij} B^{\mathrm{s}}(i,j,n) + H[1 - ws(i,n)] + H[1 - wf(i,n')] + H[\sum_{n \le n'' < n'} wf(i,n'')],$$
$$\forall i \notin I^{\mathrm{ps}},\quad j \in J_i,\quad n \in N,\quad n' \in N,\quad n \le n'. \tag{26}$$

The first set of constraints represent the relationship between the starting and finishing times of task i in unit j at event point n. Because tasks can extend over multiple event points, the finishing time is not assigned from the start time, but must be greater or equal to the start time. Constraints (23) also represent the relationship between the starting and finishing times of task i in unit j at event point n. If task i does not take place at event point n, then along with constraints (22), the finishing time is set equal to the starting time. Constraints (24) relate the starting time of task i in unit j at event point n to the finishing time of the same task in the same unit at the previous event point, $n-1$. These constraints are relaxed unless task i is active and does not finish processing at event point $n-1$. In this case, task i must extend to the following event point, n, so that this constraint, $T^{\mathrm{s}}(i,j,n) \le T^{\mathrm{f}}(i,j,n-1)$, along with the sequencing constraints (29), $T^{\mathrm{s}}(i,j,n) \ge T^{\mathrm{f}}(i,j,n-1)$, results in the two times being equal. Constraints (25) and (26) relate the starting time of task i in unit j at event point n with its finishing time at a later event point n'.

$$T^{\mathrm{f}}_{\mathrm{st}}(i^{\mathrm{st}},n) \ge T^{\mathrm{s}}_{\mathrm{st}}(i^{\mathrm{st}},n),\quad \forall i^{\mathrm{st}} \in I^{\mathrm{st}}_s,\quad n \in N \tag{27}$$

$$T^{\mathrm{f}}_{\mathrm{ut}}(u,n) \ge T^{\mathrm{s}}_{\mathrm{ut}}(u,n),\quad \forall u \in U,\quad n \in N. \tag{28}$$

The first set of constraints relate the starting and finishing times of a storage task i^{st} so that the finishing time must always be greater than or equal to the start time. The second set of constraints relate the starting and finishing times of changes in the amount of utility u so that the finish time must always be greater than or equal to the start time.

Sequence Constraints: Processing Tasks

$$T^{\mathrm{s}}(i,j,n) \ge T^{\mathrm{f}}(i,j,n-1),\quad \forall i \in I,\quad j \in J_i,\quad n \in N,\quad n > 1 \tag{29}$$

$$T^{\mathrm{s}}(i,j,n) \ge T^{\mathrm{f}}(i',j,n-1) - H[1 - w(i',n-1)],\quad \forall j \in J,\quad i \in I_j,\quad i' \in I_j,\quad i \ne i',\quad n \in N,\quad n > 1 \tag{30}$$

$$T^{\mathrm{s}}(i,j,n) \ge T^{\mathrm{f}}(i',j',n-1) - H[1 - wf(i',n-1)],\quad \forall s \in S,\quad i \in I^{\mathrm{c}}_s,\quad i' \in I^{\mathrm{p}}_s,\quad j \in J_i,\quad j' \in J_{i'},\quad j \ne j',\quad n \in N,\quad n > 1 \tag{31}$$

$$T^{\mathrm{s}}(i,j,n) \le T^{\mathrm{f}}(i',j',n-1) + H[2 - wf(i',n-1) - ws(i,n)],\quad \forall s \in S^z, S^{\mathrm{f}}, S^n,\quad i \in I^{\mathrm{c}}_s,\quad i' \in I^{\mathrm{p}}_s,\quad j \in J_i,\quad j' \in J_{i'},\quad j \ne j',\quad n \in N,\quad n > 1. \tag{32}$$

The sequence constraints in (29) state that task i starting at event point n should start after the end of the same task performed at the same unit j which has finished at the previous event point, $n-1$. The constraints in (30) are written for tasks i and i' that are performed in the same unit j at event points n and $n-1$, respectively. If both tasks take place in the same unit, they should be at most consecutive. Constraints (31) relate tasks i and i' which are performed in different units j and j' but take place consecutively according to the production recipe. Constraints (32) are written for different tasks i and i' that take place consecutively with the "zero-wait" (ZW) condition owing to storage restrictions on the intermediate material. Combined with constraints 31, these constraints enforce that task i in unit j at event point n starts immediately after the end of task i' in unit j' at event point $n-1$ if both tasks are activated.

Sequence Constraints: Storage Tasks

$$T^{\mathrm{s}}(i,j,n) \ge T^{\mathrm{f}}_{\mathrm{st}}(i^{\mathrm{st}},n-1),\quad \forall s \in S,\quad i \in I^{\mathrm{c}}_s,\quad j \in J_i,\quad i^{\mathrm{st}} \in I^{\mathrm{st}}_s,\quad n \in N,\quad n > 1 \tag{33}$$

$$T^{\mathrm{s}}(i,j,n) \le T^{\mathrm{f}}_{\mathrm{st}}(i^{\mathrm{st}},n-1) + H[1 - ws(i,n)],\quad \forall s \in S^{\mathrm{f}},\quad i \in I^{\mathrm{c}}_s,\quad j \in J_i,\quad i^{\mathrm{st}} \in I^{\mathrm{st}}_s,\quad n \in N,\quad n > 1 \tag{34}$$

$$T^{s}_{st}(i^{st}, n) \geq T^{f}(i', j', n-1) - H[1 - wf(i', n-1)],$$
$$\forall s \in S,\quad i' \in I^{p}_{s},\quad j' \in J_{i'},\quad i^{st} \in I^{st}_{s},$$
$$n \in N,\ n > 1 \quad (35)$$

$$T^{s}_{st}(i^{st}, n) \leq T^{f}(i', j', n-1) + H[1 - wf(i', n-1)],$$
$$\forall s \in S^{f},\quad i' \in I^{p}_{s},\quad j' \in J_{i'},\quad i^{st} \in I^{st}_{s},$$
$$n \in N,\ n > 1 \quad (36)$$

$$T^{s}_{st}(i^{st}, n) = T^{f}_{st}(i^{st}, n-1),\quad \forall i^{st} \in I^{st}_{s},\quad n \in N,\ n > 1 \quad (37)$$

The first two sets of constraints relate the starting time of a processing task i at an event point n to the finishing time of a storage task i^{st} at the previous event point, $n-1$. Note that (34) is only written for states s with FIS. Thus, if task i starts at event point n and consumes a state s that requires FIS, then we have $T^{s}(i, j, n) = T^{f}_{st}(i^{st}, n-1)$ for all storage tasks i^{st} for state s. Constraints (35) and (36) relate the starting time of a storage task i^{st} at an event point n to the processing task i' at the previous event point, $n-1$. Similar to constraints (33) and (34), these constraints enforce the timing for a processing task that produces a FIS state and a storage task which stores the same FIS state. Constraints (37) relate the starting and finishing time of a storage task i^{st} at two consecutive event points. They ensure that, along with constraints (33)–(36), the timing for storage of FIS states will be enforced so that storage limitations are not violated.

Sequence Constraints: Utility-Related Tasks

$$T^{f}(i, j, n-1) \geq T^{s}_{ut}(u, n) - H[1 - w(i, n-1) + wf(i, n-1)],$$
$$\forall u \in U,\quad i \in I_{u},\quad j \in J_{i},\quad n \in N, n > 1 \quad (38)$$

$$T^{f}(i, j, n-1) \leq T^{s}_{ut}(u, n) + H[1 - w(i, n-1)],$$
$$\forall u \in U,\quad i \in I_{u},\quad j \in J_{i},\quad n \in N,\ n > 1 \quad (39)$$

$$T^{s}_{ut}(u, n) \geq T^{s}(i, j, n) - H[1 - w(i, n)],$$
$$\forall u \in U,\quad i \in I_{u},\quad j \in J_{i},\quad n \in N \quad (40)$$

$$T^{s}_{ut}(u, n) \leq T^{s}(i, j, n) + H[1 - w(i, n)],$$
$$\forall u \in U,\quad i \in I_{u},\quad j \in J_{i},\quad n \in N \quad (41)$$

$$T^{s}_{ut}(u, n) = T^{f}_{ut}(u, n-1),\quad \forall u \in U,\quad n \in N,\ n > 1 \quad (42)$$

The first two sets of constraints relate the finishing time of a processing task i which utilizes utility u at an event point $n-1$ to the starting time of the usage of utility u at the next event point. Constraints (40) and (41) relate the starting time of the usage of utility u at event point n to the processing task i which utilizes utility u at the current event point. Constraints (42) relate the starting and the finishing time of the usage of utility u at two consecutive event points. They ensure that, along with constraints (38)–(41), the timing for the changes in the utility level will be consistent and the amounts of utilities used can be monitored exactly and specified limits enforced.

Order Satisfaction Constraints

The order satisfaction constraints provided here are written for problems involving network-represented processes. Note that these constraints can easily be modified for the case of sequential process problems. This is done by relating orders to units in the same manner as orders are related to tasks below.

$$\sum_{i \in I_k} \sum_{n \in N} y(k, i, n) = 1,\quad \forall k \in K \quad (43)$$

$$wf(i, n) = \sum_{k \in K_i} y(k, i, n),\quad \forall i \in I,\quad n \in N \quad (44)$$

$$D(s, n) = \sum_{k \in K_s} \sum_{i \in I_k} am_k \cdot y(k, i, n),$$
$$\forall s \in S^{p},\quad n \in N \quad (45)$$

$$T^{f}(i, j, n) \leq due_k + H[2 - y(k, i, n) - wf(i, n)],$$
$$\forall s \in S,\quad k \in K_s,\quad i \in I_k,\quad j \in J_i,\quad n \in N \quad (46)$$

$$T^{f}(i, j, n) \geq due_k - H[2 - y(k, i, n) - wf(i, n)],$$
$$\forall s \in S,\quad k \in K_s,\quad i \in I_k,\quad j \in J_i,\quad n \in N \quad (47)$$

The first set of constraints ensure each order k is met exactly once; thus, each order is processed by only one

task i and is delivered at exactly one event point n. Constraints (44) relate the delivery of an order k through task i to the activation of task i at event point n. Constraints (45) relate the amount of state s delivered at event point n, $D(s,n)$, to the amount of that state due through order k. Thus, if state s has two orders associated with it, $k1$ and $k2$, of amounts am_{k1} and am_{k2}, respectively, and orders $k1$ and $k2$ are both delivered at event point n, then the delivery of state s at event point n is represented as $D(s,n) = am_{k1} + am_{k2}$ and the amount of the state delivered will be equal to the amount ordered. Constraints (46) and (47) relate the time that order k is due to the actual time that order k is delivered.

Bound Constraints

$$
\begin{aligned}
&T^{\mathrm{f}}(i,j,n) \le H, \quad \forall i \in I, \quad j \in J_i, \quad n \in N \\
&T^{\mathrm{s}}(i,j,n) \le H, \quad \forall i \in I, \quad j \in J_i, \quad n \in N \\
&T^{\mathrm{f}}_{\mathrm{ut}}(u,n) \le H, \quad \forall u \in U, \quad n \in N \\
&T^{\mathrm{s}}_{\mathrm{ut}}(u,n) \le H, \quad \forall u \in U, \quad n \in N \\
&T^{\mathrm{s}}_{\mathrm{ut}}(u,n) = 0, \quad \forall u \in U, \quad n \in N, \; n = 1 \\
&STO(s) = 0, \quad \forall s \notin S^r \\
&STO(s) \le ST^0_s, \quad \forall s \in S^r \\
&ST(s,n) = 0, \quad \forall s \in S^z, S^{\mathrm{f}}, S^n \quad n \in N \\
&ST(s,n) \le ST^{\max}_s, \quad \forall s \in S, \quad n \in N \\
&D(s,n) = 0, \quad \forall s \notin S^{\mathrm{p}}, \quad n \in N \\
&0 \le w(i,n) \le 1, \quad \forall i \in I, \quad n \in N
\end{aligned}
\tag{48}
$$

These constraints represent bounds on several of the continuous variables. The starting and finishing times of processing tasks and changes in utility level must all be within the time horizon. The start time for the changes in utilities at the first event point is set to zero. The initial amounts of all non-raw-material states are set to zero, the intermediate amounts of all ZW, NIS, and FIS states are set to zero, and the amounts of all nonproduct deliveries are set to zero. Also, the continuous variable representing the activation of task i in unit j at event point n, $w(i,n)$, must fall between zero and one.

Objective Function

There are several different objective functions that can be employed with a general short-term scheduling problem. Three of the most common types are reviewed below.

Maximization of sales

$$
\max \sum_{s \in S^p} price_s \cdot \left[\sum_{n \in N} D(s,n) + STF(s) \right] \tag{50}
$$

The objective function represents the maximization of the value of the final products.

Minimization of makespan

$$
Min \quad MS \tag{50}
$$

$$
s.t. \quad MS \ge T^{\mathrm{f}}(i,j,n), \quad \forall i \in I, \quad j \in J_i, \quad n \in N \tag{51}
$$

$$
STF(s) = dem_s, \quad \forall s \in S^p \,, \tag{52}
$$

where MS is the makespan. The objective function represents the minimization of the makespan of the process for a fixed demand for each state s, dem_s, contained in the set of final products, S^p.

Minimization of order earliness

$$
Min \sum_{k \in K} due_k - \left[\sum_{i \in I_k} \sum_{j \in J_i} \sum_{n \in N} T^{\mathrm{f}}(i,j,n) \cdot y(k,i,n) \right] \tag{52}
$$

The objective function represents the minimization of the total earliness of all orders where the bilinear term, which is a product of a continuous and a binary variable, can be replaced with a continuous variable and supporting constraints using a Glover transformation [4,5].

Methods

Number of Event Points

In this formulation, the number of event points is determined using the same approach as proposed in [6]. First, the problem is solved with a small number of event points to obtain a solution. Then, the number of event points is increased by one and the problem is resolved to obtain a better solution. This is repeated until an additional increase in the number of event points does not result in any improvement in the objective function.

Tightening Constraints

Following the tightening constraints suggested by Maravelias and Grossmann [11], similar constraints are introduced to tighten the relaxed solution of the proposed enhanced formulation. Specifically, constraints (54) tighten the formulation by enforcing the condition that the summation of the processing times of the tasks assigned to a specific unit j should be less than or equal to the time horizon.

$$\sum_{i\in I_j}\sum_{n\in N}\alpha_{ij}ws(i,n)+\beta_{ij}B^{s}(i,j,n)\le H,\quad \forall j\in J \quad (54)$$

Furthermore, this condition is also enforced for each unit j at each event point n as follows:

$$\sum_{i\in I_j}\sum_{n'\ge n}\alpha_{ij}ws(i,n')+\beta_{ij}B^{s}(i,j,n') \le H-TT^{s}(j,n),\quad \forall j\in J,\quad n\in N \quad (55)$$

$$\sum_{i\in I_j}\sum_{n'<n}\alpha_{ij}ws(i,n')+\beta_{ij}B^{s}(i,j,n') \le TT^{f}(j,n-1)+H[1-\sum_{i\in I_j}wf(i,n-1)], \quad \forall j\in J,\quad n\in N,\ n>1, \quad (56)$$

where $TT^{s}(j,n)$ and $TT^{f}(j,n)$ are the starting time and the finishing time, respectively, of the task active in unit j at event point n. They are defined as follows:

$$\begin{aligned} TT^{s}(j,n) &\le T^{s}(i,j,n)+H[1-ws(i,n)], \quad \forall j\in J,\quad i\in I_j,\quad n\in N \\ TT^{s}(j,n) &\ge T^{s}(i,j,n)-H[1-ws(i,n)], \quad \forall j\in J,\quad i\in I_j,\quad n\in N \end{aligned} \quad (57)$$

$$\begin{aligned} TT^{f}(j,n) &\le T^{f}(i,j,n)+H[1-wf(i,n)], \quad \forall j\in J,\quad i\in I_j,\quad n\in N \\ TT^{f}(j,n) &\ge T^{f}(i,j,n)-H[1-wf(i,n)], \quad \forall j\in J,\quad i\in I_j,\quad n\in N \end{aligned} \quad (58)$$

Thus, constraints (55) enforce the condition that the summation of the processing times of all tasks starting in unit j at event points n or greater must be less than or equal to the amount of time remaining. Likewise, constraints (56) enforce the condition that the summation of the processing times of all tasks finishing in unit j before event point n must be less than or equal to the amount of time that has passed up to the beginning of event point n. Note that constraints (56) are only active if a task finishes at the previous event point, $n-1$, otherwise, $TT^{f}(j,n)$ will not have an exact value and the constraint is relaxed.

The addition of constraints (54)–(58) leads to relaxed solutions with smaller sums of processing times, or smaller durations. This then leads to fewer activated binary variables, $ws(i,n)$ and $wf(i,n)$. Moreover, the continuous variables including the batch sizes ($B^{s}(i,j,n)$, $B^{f}(i,j,n)$, $B(i,j,n)$) and the amounts of states ($ST(s,n)$, $STF(s)$) are all bounded by the binary variables. Finally, because these continuous variables appear in the objective function, the addition of these constraints results in tighter relaxations.

Sequential Processes

Single-stage and multistage sequential processes are batch- or order-oriented and thus do not need to include tasks or states or any of the constraints involving states. The model described in the previous section can be applied to sequential processes with a few modifications. For instance, there are no defined tasks, states, batch sizes, or material amounts and all material balances and capacity constraints are unnecessary. Thus, the basic constraints (1)–(5), (22)–(26), (29)–(32), and part of (48) all apply. The order satisfaction constraints (43)–(47) need to be modified as previously detailed. If storage constraints are to be considered, then constraints (27) and (33)–(37) need to be included and if resource constraints are to be considered, then constraints (28) and (38)–(42) should also be included. Furthermore, all of the binary and continuous variables and their participating constraints should be modified similarly to the order satisfaction constraints to reflect a dependence on orders associated with units instead of tasks associated with units.

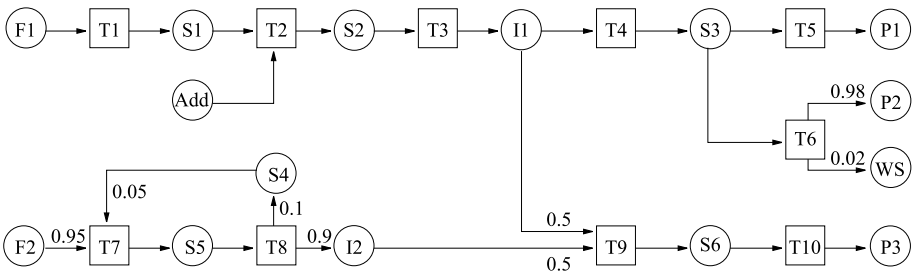
Cases

In this section, two example problems are presented to demonstrate the effectiveness of the proposed approach. Both general network-represented and sequential processes are considered. Comparisons with previously published approaches are also provided. All examples are implemented with GAMS 2.5 [1] and are

solved using CPLEX 8.1 with a 3.00 GHz Linux workstation. The default GAMS/CPLEX options are used in all runs with the exception that the CPLEX option for feasibility is activated and a relative optimality tolerance equal to 0.01% was used as the termination criterion.

Case 1: Resource Constraints, Mixed Storage Policies, Variable Batch Sizes, and Processing Times

The second case also comes from Maravelias and Grossmann [11] and involves the STN given in Fig. 2. This example includes resource constraints, mixed storage policies, and variable batch sizes, processing times, and utility requirements. The plant consists of six units involving ten processing tasks and fourteen states. Unlimited intermediate storage (UIS) is available for raw materials F1 and F2, intermediates I1 and I2, and final products P1–P3 and WS. Finite intermediate storage (FIS) is available for states S3 and S4, while no intermediate storage (NIS) is available for states S2 and S6 and a ZW policy applies for states S1 and S5. There are three different renewable utilities: cooling water (CW), low-pressure steam (LPS), and high-pressure steam (HPS). Tasks T2, T7, T9, and T10 require CW; tasks T1, T3, T5, and T8 require LPS; and tasks T4 and T6 require HPS. The maximum availabilities of CW, LPS, and HPS are 25, 40, and 20 kg/min, respectively. The corresponding data for the example can be found in Tables 1 and 2. The objective function is the maximization of sales and time horizons of 12 and 14 h are considered.



Short-Term Scheduling of Batch Processes with Resources, Figure 2
State-task network for case 1

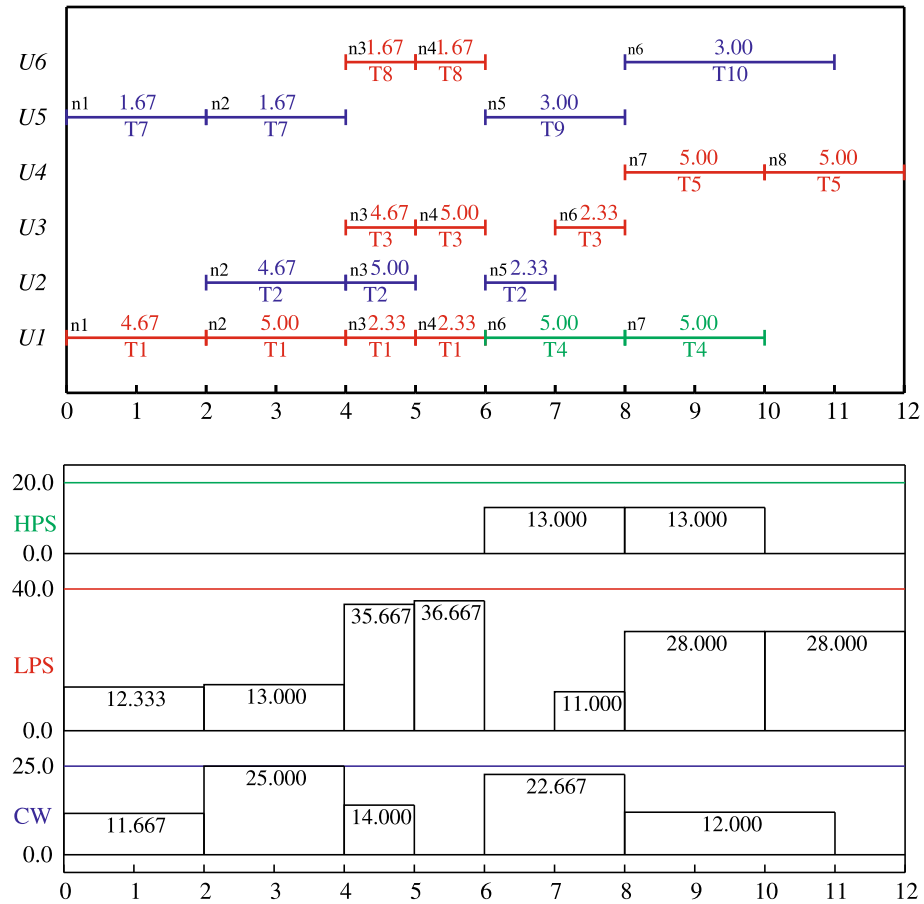
Short-Term Scheduling of Batch Processes with Resources, Table 1
Storage restrictions for case 1

	F1	F2	S1	S2	S3	S4	S5	S6	INT1	INT2	P1	P2	P3
ST_s^{max} (kg)	∞	∞	0	0	15	40	0	0	∞	∞	∞	∞	∞
ST_s^0 (kg)	100	100	0	0	0	10	0	0	0	0	0	0	0
$price_s$ ($/kg)	–	–	–	–	–	–	–	–	–	–	1	1	1

Short-Term Scheduling of Batch Processes with Resources, Table 2
Resource utilizations for case 1

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
Unit	U1	U2	U3	U1	U4	U4	U5	U6	U5	U6
cap^{max} (kg)	5	8	6	5	8	8	3	4	3	4
α (h)	2	1	1	2	2	2	4	2	2	3
Utility	LPS	CW	LPS	HPS	LPS	HPS	CW	LPS	CW	CW
γ (kg/min)	3	4	4	3	8	4	5	5	5	3
δ (kg/min/kg batch)	2	2	3	2	4	3	4	3	3	3

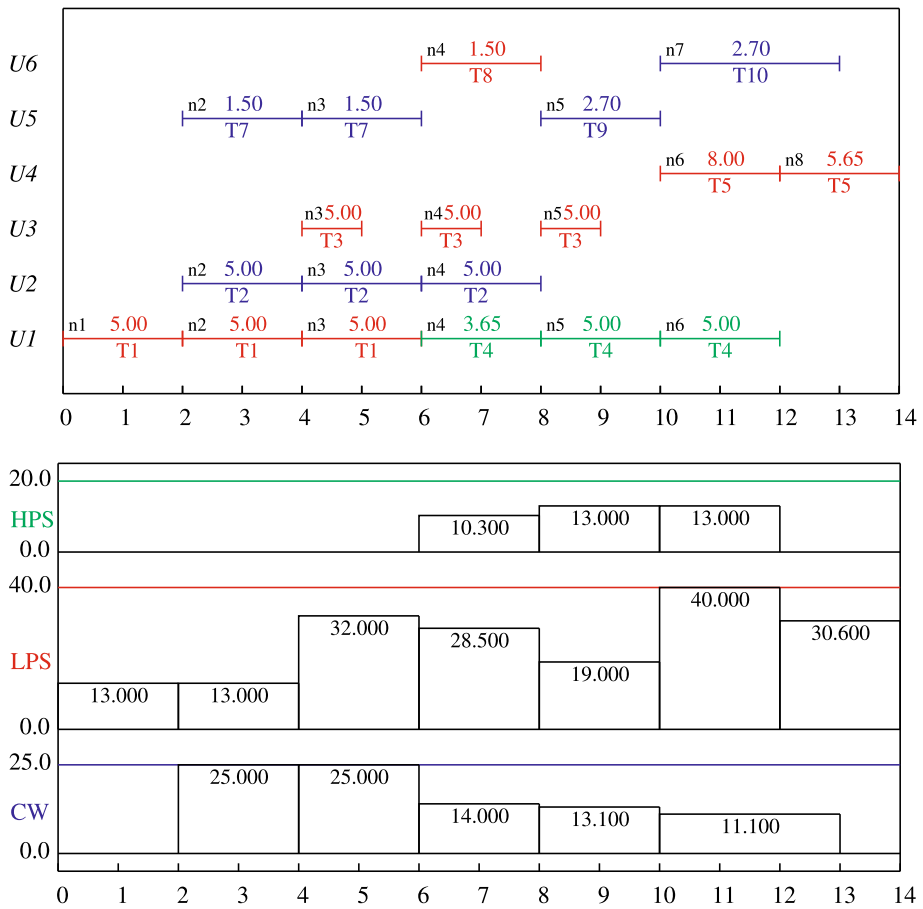
LPS low-pressure steam, *CW* cooling water, *HPS* high-pressure steam



Short-Term Scheduling of Batch Processes with Resources, Figure 3
Schedule for case 1 with a time horizon of 12 h. *HPS* high-pressure steam, *LPS* low-pressure steam, *CW* cooling water

For a time horizon of 12 h, the optimal sales are \$13,000 and eight event points are required. The production schedule and resource utilization levels can be seen in Fig. 3. The problem involves 3318 constraints, 110 binary variables, and 1077 continuous variables. Its optimal solution was found in 222 nodes and 1.71 s. For a time horizon of 14 h, the optimal sales are \$16,350 and eight event points are required. The production schedule and resource utilization levels can be seen in Fig. 4. The problem involves 3354 constraints, 109 binary variables, and 1077 continuous variables. Its optimal solution was found in 2869 nodes and 15.65 s. Note that in both cases, the limiting resource is CW, as can be seen from the resource utilization levels. In both schedules, tasks T2 and T7 occurring at the same time requires the maximum amount of CW available.

Case 1 was also solved with the model M* of Maravelias and Grossmann [11] to compare the two formulations. Although this example was solved in their original paper, we have re-solved it here in order to compare the models using the same computational tools. The model and solution statistics using both models can be seen in Table 3. For the time horizon of 12 h using nine time points, the model involved 2396 constraints, 180 binary variables, and 1408 continuous variables. The same optimal solution of \$13,000 was found in 23,235 nodes and 64.92 s. For the time horizon of 14 h using ten time points, the model involved 2663 constraints, 200 binary variables, and 1564 continuous variables. The same optimal solution of \$16,350 was found in 22,625 nodes and 112.66 s. Note that the reported number of nodes and CPU seconds for both time hori-



Short-Term Scheduling of Batch Processes with Resources, Figure 4
Schedule for case 1 with a time horizon of 14 h

zons using our application of model M* are different from those found by Maravelias and Grossmann [11]. For a time horizon of 12 h, they reported 2107 nodes, while our application of model M* took 23,235 nodes. For a time horizon of 14 h, they reported 60,070 nodes, while our application of model M* took 22,625 nodes. In addition, for both time horizons, model M* of Maravelias and Grossmann [11] takes at least one more time point and thus involves more binary and continuous variables. Also, the time horizon of 12 h took over 100 times more nodes to solve, while the time horizon of 14 h took over 10 times more nodes to solve. This indicates that when a larger number of time points are considered in a problem, the proposed model performs better computationally than the model of Maravelias and Grossmann [11], even when the objective is the maximization of sales.

Case 2: Sequential Process with Order-Dependent Processing Times

The second case is taken from Pinto and Grossmann [12] and involves a sequential process containing one stage with four parallel extruders of unequal capacity and with processing times depending on the order being processed. A total of 12 orders are due at specific times over a 30-day period. The corresponding processing rate and due date data for the example can be found in Table 4. The objective of the problem is to meet all orders while minimizing earliness, as seen in constraint (53).

The optimal processing schedule is given in Fig. 5 with an objective function value of 1.026. The problem was modeled with the formulation of Ierapetritou and Floudas [6] using only the allocation, duration, and

Short-Term Scheduling of Batch Processes with Resources, Table 3
Model and solution statistics for case 1

	Proposed formulation		Maravelias and Grossmann [11] formulation	
Horizon	12 h	14 h	12 h	14 h
Event points	8	8	9	10
Binary variables	110	110	180	200
Continuous variables	1077	1077	1408	1564
Constraints	3318	3354	2396	2663
LP relaxation	19000	19000	18423.5	22186.7
Objective	$13000	$16350	$13000	$16350
Nodes[1]	222	2869	23235 (2107)	22625 (60070)
CPU time (s)	1.71	15.65	64.92	112.66

[1]Numbers in parentheses represent values reported by Maravelias and Grossmann [11]

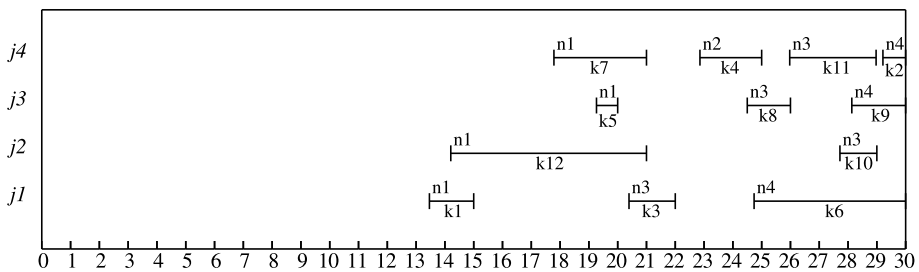
Short-Term Scheduling of Batch Processes with Resources, Table 4
Data for case 2

Order	Due date (day)	Processing time (days)			
		j1	j2	j3	j4
1	15	1.538			1.194
2	30	1.500			0.789
3	22	1.607			0.818
4	25			1.564	2.143
5	20			0.736	1.017
6	30	5.263			3.200
7	21	4.865		3.025	3.214
8	26			1.500	1.440
9	30			1.869	2.459
10	29		1.282		
11	30		3.750		3.000
12	21		6.796	7.000	5.600
Transition		0.180	0.175	0.00	0.237

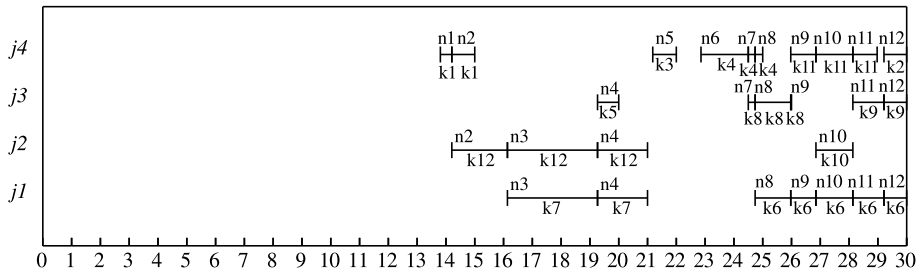
same task in the same unit and different task in the same unit sequencing constraints along with the order satisfaction constraints outlined in (43)–(47) and the objective given in constraint (53).

Suppose now that, owing to limited manpower, there is a hard constraint on the number of extruders which can operate at the same time. We will consider the case where three extruders may operate simultaneously (type 1) and the case where only two extruders may operate simultaneously (type 2). For both cases, we employ the model outlined in Sect. "Sequential Processes", again using the order satisfaction constraints and the objective function to minimize the earliness of the orders. For type 1 with three extruders, the optimal objective function value is 1.895 and the production schedule can be seen in Fig. 6. For type 2, the optimal objective function value is 7.909 and the production schedule can be seen in Fig. 7. Model and solution statistics for all three cases can be seen in Table 5.

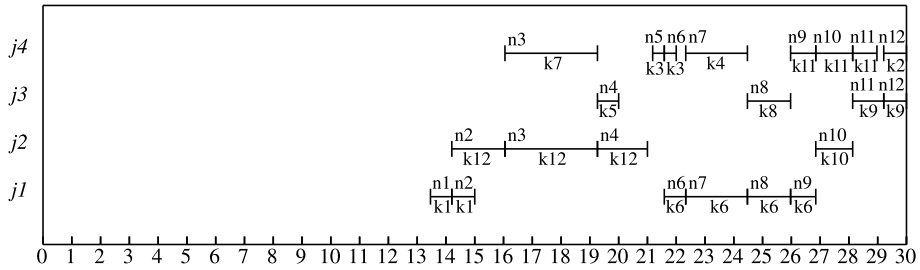
It can be seen from Table 5 that consideration of resource constraints in the form of limited manpower increases the computational complexity of the problem. Resource considerations require a more complicated model involving more variables and constraints. For instance, resource considerations require an event point for every time the resource level changes, or in this case, for each order. However, a problem without resources only requires as many event points as the maximum number of sequential tasks. For this example, the simpler problem without resource considerations only requires four event points, while the more com-



Short-Term Scheduling of Batch Processes with Resources, Figure 5
Schedule for case 2 without limited manpower



Short-Term Scheduling of Batch Processes with Resources, Figure 6
Schedule for case 2 with three extruders (type 1)



Short-Term Scheduling of Batch Processes with Resources, Figure 7
Schedule for case 2 with two extruders (type 2)

plicated problem with resource constraints requires 12 event points, resulting in many more binary variables and thus a much more complex problem.

In order to test the effectiveness of the proposed formulation when used with sequential processes, we performed a computational comparison for this example with the model of Pinto and Grossmann [12]. Although this example was solved in their original paper, the objective function used was the maximization of starting times instead of the minimization of tardiness. So, we re-solved our model using the maximization of the starting times as the objective. Pinto and Grossmann [12] reported optimal objectives of 269.10, 268.24, and 264.98 for the three cases of no resources, resources limited to three extruders, and resources limited to two extruders. Our optimal objective function values with the same objective were 269.10, 268.82, and 265.74, respectively. Thus, the proposed model found improved schedules with a better objective function value for the case where resources are limited to three extruders and the case where resources are limited to two extruders. This is not unexpected, however, owing to the fact that the model used in [12] employs the concept of time slots and all slot-based formulations restrict the time representations and hence they can result, by definition, in suboptimal solutions. Note that the model and solution statistics found in Table 5 for this problem using an objective function of minimization of order earliness are comparable to the model and solution statistics determined using an objective of maximization of start times. We do not make a com-

Short-Term Scheduling of Batch Processes with Resources, Table 5
Model and solution statistics for case 2

	4 extruders simultaneously	3 extruders simultaneously	2 extruders simultaneously
Event points	4	12	12
Binary variables	150	458	444
Continuous variables	513	2137	2137
Constraints	1389	10382	10382
LP relaxation	0	0	0
Objective	1.026	1.895	7.334
Nodes	7	1374	38621
CPU time (s)	0.07	6.53	236.87

parison with the model and solution statistics presented in [12] because the authors did not report the integrality gaps or other optimality criterion used; thus, a direct comparison would not be meaningful.

Conclusions

In this chapter, an enhanced continuous-time formulation was presented for the short-term scheduling of multipurpose batch plants with intermediate due dates. The proposed formulation incorporates several features, including various storage policies (UIS, FIS, NIS, ZW), resource constraints, variable batch sizes and processing times, batch mixing and splitting, and sequence-dependent changeover times. The key features of the proposed formulation include a continuous-time representation utilizing a necessary number of event points of unknown location corresponding to the activation of a task. Also, tasks are allowed to continue over several event points, enabling resource quantities to be correctly determined at the beginning of each resource utilization. Four examples were presented to illustrate the effectiveness of the proposed formulation. The computational results were compared with those in the literature and it was shown that the proposed formulation is significantly faster than other general resource-constrained models, especially for problems requiring many time points.

References

1. Brooke A, Kendrick D, Meeraus A (1988) GAMS: A User's Guide. South San Francisco, CA
2. Castro P, Barbosa-Póvoa APFD, Matos H (2001) An Improved RTN Continuous-Time Formulation for the Short-Term Scheduling of Multipurpose Batch Plants. Ind Eng Chem Res 40:2059
3. Castro P, Barbosa-Póvoa APFD, Matos HA, Novais AQ (2004) Simple Continuous-Time Formulation for Short-Term Scheduling of Batch and Continuous Processes. Ind Eng Chem Res 43:105
4. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization. Oxford University Press
5. Glover F (1975) Improved Linear Integer Programming Formulations of Nonlinear Integer Problems. Manage Sci 22:455
6. Ierapetritou MG, Floudas CA (1998) Effective Continuous-Time Formulation for Short-Term Scheduling: 1. Multipurpose Batch Processes. Ind Eng Chem Res 37:4341
7. Ierapetritou MG, Floudas CA (1998) Effective Continuous-Time Formulation for Short-Term Scheduling: 2. Continuous and Semi-continuous Processes. Ind Eng Chem Res 37:4360
8. Ierapetritou MG, Hené TS, Floudas CA (1999) Effective Continuous-Time Formulation for Short-Term Scheduling: 3. Multiple Intermediate Due Dates. Ind Eng Chem Res 38:3446
9. Janak SL, Lin X, Floudas CA (2004) Enhanced Continuous-Time Unit-Specic Event Based Formulation for Short-Term Scheduling of Multipurpose Batch Processes: Resource Constraints and Mixed Storage Policies. Ind Eng Chem Res 43:2516
10. Lin X, Floudas CA (2001) Design, Synthesis and Scheduling of Multipurpose Batch Plants via an Effective Continuous-Time Formulation. Comp Chem Engin 25:665
11. Maravelias CT, Grossmann IE (2003) A New General Continuous-Time State Task Network Formulation for Short-Term Scheduling of Multipurpose Batch Plants. Ind Eng Chem Res 42:3056
12. Pinto JM, Grossmann IE (1997) A Logic-Based Approach to Scheduling Problems with Resource Constraints. Comp Chem Engin 21:801
13. Schilling G (1997) Optimal Scheduling of Multipurpose Plants. Ph.D. thesis, University of London

Short-Term Scheduling of Continuous Processes

Munawar A. Shaik,
Christodoulos A. Floudas
Department of Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90B35, 65K05, 90C90, 90C11

Article Outline

Introduction
Formulation
  Unlimited Intermediate Storage
  Dedicated Finite Intermediate Storage
    with Storage Bypassing Allowed
  No Intermediate Storage
  Flexible Finite Intermediate Storage
    with Storage Bypassing Allowed
  Dedicated and Flexible Finite Intermediate Storage
    Without Storage Bypassing
Computational Case Study
Conclusions
  Nomenclature
References

Introduction

The scheduling problem of multiproduct and multipurpose continuous plants has received relatively little attention in the literature despite its practical importance in the chemical process industries that produce a variety of products using both batch and continuous production modes. Scheduling of a continuous plant typically involves handling continuous production, management of transitions, and accommodating inventory constraints, while meeting demands for final products with specified due dates. The problem deals with sequencing of products on each unit, quantitative determination of production amounts, optimal start and finish times of production, and storage tasks. One of the key differences between scheduling of batch processes and scheduling continuous processes is in handling the processing times. In batch plants, the processing times are typically fixed and known a priori, and the production amount depends on the capacity of the batch unit. In continuous plants, the processing times are a function of unit-dependent processing rates, final product demand, and storage limitations. Additionally, in continuous plants, the production amount is available continuously while it is being produced, unlike in batch plants, where the amount is available only after the end time for the batch that is being processing. Owing to these differences, the problem of scheduling of continuous plants has drawn separate attention.

Floudas and Lin [3,4] presented extensive reviews comparing various discrete-time-based and continuous-time-based formulations. During the last two decades, numerous formulations have been proposed in the literature based on continuous-time representation, owing to their established advantages over discrete-time representations. On the basis of the time representation used, the different continuous-time models proposed in the literature can be broadly classified into three distinct categories: slot-based, global event-based, and unit-specific event-based formulations for both network-represented and sequential processes. In the slot-based models, the time horizon is represented in terms of ordered blocks of unknown, variable lengths, or time slots. In addition, alternative methods have been developed which define continuous variables directly to represent the timings of tasks without the use of time slots. These methods can be classified into two different representations of time, global event-based models and unit-specific event-based models. Global event-based models use a set of events that are common across all units, and the event points are defined for either the beginning or the end (or both) of each task in each unit. On the other hand, unit-specific event-based models define events on a unit basis, allowing tasks corresponding to the same event point but in different units to take place at different times. For sequential processes, other alternative approaches based on precedence relationships have also been used. A detailed comparison of various continuous-time models for short-term scheduling of batch plants was performed recently by Shaik et al. [28]. They concluded that, owing to the heterogeneous locations of the event points used, the unit-specific event-based models always require fewer event points and exhibit favorable computational performance compared with both slot-based and global event-based models.

There are two types of demand patterns for scheduling of continuous processes that have been addressed in the literature: cyclic and short-term. In cyclic scheduling, a cyclic mode is assumed and the product demands are specified in terms of constant demand rates at the end of a specified time horizon, while short-term scheduling deals with a general problem where the product demands have different sets of due dates. Sahinidis and Grossmann [25] proposed a slot-based mixed-integer nonlinear programming (MINLP) model for cyclic scheduling of single-stage continuous parallel production lines that do not share any common resources. Pinto and Grossmann [24] modeled cyclic scheduling for sequential operation of a multistage, multiproduct continuous plant based on a slot-based continuous-time representation, leading to an MINLP model with explicit inventory breakpoints for representing intermediate storage. Munawar et al. [23] extended the slot-based cyclic scheduling formulation of Pinto and Grossmann [24] to hybrid flowshops consisting of serial and parallel configurations of processing and storage units. They proposed a modified definition of the time slot to account for feed losses during product transitions. Zhang and Sargent [30,31] proposed a global event-based MINLP model using a resource-task network (RTN) for the optimal operation of a mixed-production facility consisting of batch and continuous processes. Schilling and Pantelides [26]

also used the RTN framework for their slot-based continuous-time formulation for short-term scheduling of batch and continuous processes and proposed a special branch and bound solution that branches both on discrete and on continuous variables. Karimi and McDonald [15,19] developed two slot-based models for production planning and short-term scheduling, which differ in the pre-assignment of slots to the time periods for a single-stage multiproduct facility consisting of parallel semicontinuous processors. Floudas and coworkers [7,8,9,10,11,12,13,14,16,17,18] developed unit-specific event-based models for a variety of problems involving design, synthesis, short-term scheduling, medium-term scheduling, reactive scheduling, and scheduling under uncertainty. Ierapetritou and Floudas [7,8] proposed a novel short-term scheduling model for batch and continuous processes using a state-task-network (STN) framework deploying a unit-specific event-based continuous-time representation. They used an approximation of the storage-task timings for handling different storage requirements in their model for continuous processes [8]. Mockus and Reklaitis [21,22] proposed a global event-based MINLP model for short-term scheduling of batch and continuous processes in which both the start and the end times of tasks occur at events that are common across all units. Their formulation can handle resource constraints such as limited availability of utilities and manpower. Giannelos and Georgiadis [5,6] proposed a unit-specific event-based formulation for short-term scheduling of batch and continuous processes using a STN representation. Their model is similar to that of Ierapetritou and Floudas [7,8], but they relaxed the task durations using buffer times and implicitly eliminated the various big-M constraints of Ierapetritou and Floudas [7,8]. In their models [5,6], the authors introduced special duration and sequencing constraints to ensure feasibility of material balances. The start and end times of the tasks producing/consuming the same state were, respectively, enforced to be the same. While this feature is essential for continuous processes when bypassing of storage is allowed (the reasons are discussed later in this chapter), for all other cases, it will be very restrictive and may lead to suboptimal solutions. Giannelos and Georgiadis [6] enforced these restrictions in their model for short-term scheduling of batch plants as well, leading to suboptimal solutions as recently observed by Sundaramoorthy and Karimi [29] and Shaik et al. [28]. Mendez and Cerda [20] proposed a production-campaign-based algorithmic approach for short-term scheduling of continuous processes, leading to compact models for the case when storage bypassing is allowed, with the assumptions that only one state is produced by each task and no initial inventories exist for intermediate states. Castro et al. [1] developed a global event-based formulation for short-term scheduling of batch and continuous processes using a RTN representation, where changeovers are treated as additional batch tasks. Most of the above mentioned models [1,5,8,20] can handle different storage requirements such as unlimited, finite, flexible, dedicated, and no intermediate storage policies. However, for an industrial case study of consumer goods manufacturing involving making, storage, and packing tasks, Ierapetritou and Floudas [8] solved the cases of unlimited and finite intermediate storage and reported approximate suboptimal solutions. Mendez and Cerda [20] and Giannelos and Georgiadis [5] also reported suboptimal solutions for the case of finite intermediate storage with no maximum demand limits. Castro et al. [1,2] could not find the global optimal solution for the no-intermediate-storage case even at higher event points and large computational times. They classified the problem as intractable and used a decomposition strategy [2] to improve the computational performance.

Recently, Shaik and Floudas [27] proposed an improved model for short-term scheduling of continuous processes using the unit-specific event-based continuous-time representation. They modified and extended the formulation of Ierapetritou and Floudas [8] and presented improved sequencing constraints to rigorously address the different storage requirements. Their formulation is based on the STN representation, resulting in a mixed-integer linear programming model that accurately accounts for various storage requirements such as dedicated, finite, unlimited, and no intermediate storage policies. The formulation allows for unit-dependent variable processing rates, sequence-dependent changeovers, and with/without the option of bypassing of storage. The Shaik and Floudas model is presented in Sect. "Formulation" for the cases with and without storage bypassing. In Sect. "Computational Case Study," different variants of an industrial case

study from fast moving consumer goods manufacturing is presented to demonstrate the capability of the model.

Formulation

The Shaik and Floudas model for short-term scheduling of continuous plants is described below for several instances of different storage requirements, which include (a) unlimited intermediate storage, (b) dedicated finite intermediate storage, (c) no intermediate storage, and (d) flexible finite intermediate storage. Both the with and the without bypassing of storage requirements options are discussed. Initially, the first case, when storage bypassing is allowed, is discussed, and then, the second case, without bypassing of storage, is discussed in Sect. "Dedicated and Flexible Finite Intermediate Storage Without Storage Bypassing."

Unlimited Intermediate Storage

Initially, consider the simple case of unlimited intermediate storage . For this case, there is no need to model the storage tasks explicitly, and hence the model is described only using the continuous processing tasks i_p. There is no difference in the model due to whether bypassing of storage is allowed or not, because of the availability of unlimited intermediate storage. The mathematical model consists of the following allocation constraints, capacity constraints, material balances for raw materials and intermediates, demand, duration, and sequencing constraints:

Allocation Constraints.

$$\sum_{i \in I_j} w(i,n) \le 1 \quad \forall j \in J,\ n \in N \tag{1}$$

In each unit, only one task or no task takes place at any event as represented by constraint (1).

Capacity Constraints for Processing Tasks.

$$R_{i_p}^{\min}(T^f(i_p,n) - T^s(i_p,n)) \le b(i_p,n) \le R_{i_p}^{\max}(T^f(i_p,n) - T^s(i_p,n)),\ \forall i_p \in I_p,\ n \in N \tag{2}$$

$$b(i_p,n) = R_{i_p}(T^f(i_p,n) - T^s(i_p,n)) \quad \forall i_p \in I_p,\ n \in N \tag{3}$$

The amount of material processed by a continuous processing task is constrained by lower and upper bounds in (2), which are a function of the duration of the corresponding task i_p, which is represented by the difference between the end time and the start time of the task at event n, $(T^f(i_p,n) - T^s(i_p,n))$, and the minimum and the maximum processing rate of the task i_p. For the case of constant processing rates, the amount of production is related as described by constraint (3).

Material Balances. (A) Raw Materials.

$$ST_0(s,n) + \sum_{i_p \in I_s} \rho^c_{s i_p} b(i_p,n) = 0 \quad \forall s \in S^R,\ n \in N \tag{4}$$

In constraint (4), the amount of raw material, as and when required from the external resources, is related to the amounts consumed at the corresponding event n. On the other hand, if the entire raw material requirement is supplied at the beginning of the scheduling horizon, then constraint (4) is modified as follows:

$$ST(s,n) = ST(s,n-1) + \sum_{i_p \in I_s} \rho^c_{s i_p} b(i_p,n) \quad \forall s \in S^R,\ n \in N,\ n > 1\,, \tag{5}$$

$$ST(s,n) = ST_0(s) + \sum_{i_p \in I_s} \rho^c_{s i_p} b(i_p,n) \quad \forall s \in S^R,\ n = 1\,. \tag{6}$$

The total initial amount required from external resources, $ST_0(s)$, calculated in (6), is either partly consumed at the first event $n = 1$ or remains in the storage, which is consumed during the subsequent events as described in (5).

(B) Intermediates.

$$ST(s,n) = ST(s,n-1) + \sum_{i_p \in I_s} \rho^p_{s i_p} b(i_p,n) + \sum_{i_p \in I_s} \rho^c_{s i_p} b(i_p,n) \quad \forall s \in S^{IN},\ n \in N,\ n > 1 \tag{7}$$

$$ST(s,n) = ST_0(s) + \sum_{i_p \in I_s} \rho^p_{s i_p} b(i_p,n) + \sum_{i_p \in I_s} \rho^c_{s i_p} b(i_p,n) \quad \forall s \in S^{IN},\ n = 1 \tag{8}$$

Similarly, for the intermediate state s, the material balance is written in constraints (7) and (8). $ST(s,n)$

defines the excess amount of state s at event n. In constraint (7), the first term on the right-hand side signifies the amount of state s stored at the previous event $(n-1)$ in the storage; the second term represents the amount of state s produced by the upstream processing task at event n. This total amount is either consumed in the downstream processing task indicated by the third term or remains in the storage at event n as shown on the left-hand side. At the first event, the initial amount of available intermediates is taken into account in (8). These constraints are based on the condition that an intermediate material is allowed to go directly to the production task bypassing the storage, because only the excess amount, $ST(s,n)$, is stored. The other case, where storage bypassing is not allowed, irrespective of whether the amount produced by the upstream processing unit is in excess of the amount consumed by the downstream processing, is discussed in Sect. "Dedicated and Flexible Finite Intermediate Storage Without Storage Bypassing."

Demand Constraints.

$$D_s^{\min} \le \sum_{n\in N}\sum_{i_p\in I_s} \rho^{p}_{si_p} b(i_p,n) \le D_s^{\max} \quad \forall s \in S^{p} \quad (9)$$

The material balance for the final product state s is given in constraint (9), where the total production of state s is limited between the specified lower and upper bounds on the demands of the final product.

Duration Constraints for Processing Tasks.

$$T^f(i_p,n) - T^s(i_p,n) \le Hw(i_p,n) \quad \forall i_p \in I_p,\ n \in N \quad (10)$$

The duration constraint for processing tasks is given in (10), which states that the duration is zero if the corresponding processing task is not active, otherwise the constraint is relaxed.

Sequencing Constraints. (A) Same Task in the Same Unit.

$$T^s(i,n+1) \ge T^f(i,n) \quad \forall i \in I,\ n \in N,\ n \ne N \quad (11)$$

Constraint (11) states that the start time of a task at the next event should be greater than the finish time of the same task at the current event.

(B) Different Tasks in the Same Unit. For two different tasks taking place in the same unit, the different possible changeover requirements are (i) no setup time required, (ii) changeover time is required but is independent of the sequence in which the two consecutive tasks take place, and (iii) sequence-dependent changeovers.

For the first case of no changeover times, the two constraints for the same task in the same unit and different tasks in the same unit can be combined into one equation as shown in (12):

$$T^s(i,n+1) \ge T^f(i',n) \quad \forall j \in J,\ i \in I_j,\ i' \in I_j,\ n \in N,\ n \ne N. \quad (12)$$

For the second case of sequence-independent changeovers, the constraint for different tasks in the same unit requires modification as shown in (13), where τ_j is the changeover time in unit j:

$$T^s(i,n+1) \ge T^f(i',n) + \tau_j w(i',n) \quad \forall j \in J,\ i,i' \in I_j,\ i \ne i',\ n \in N,\ n \ne N. \quad (13)$$

For the third case of sequence-dependent changeovers, the constraint for different tasks in the same unit is generalized as follows:

$$T^s(i,n) \ge T^f(i',n') + t^{cl}_{i'i} w(i,n) - H(1-w(i',n')) - H\sum_{i''}\sum_{n'<n''<n} w(i'',n'') \quad \forall j \in J,\ i,i' \in I_j,\ i \ne i',\ n,n' \in N,\ n > n'. \quad (14)$$

Note that in constraints (12)–(14), only the last constraint (14), has big-M terms, unlike in the previous formulations of Ierapetritou and Floudas [7,8]. This feature is observed to result in better linear programming relaxed solutions which often improve the computational performance.

(C) Different Tasks in Different Units for Processing Tasks. The start and finish times of different processing tasks i'_p and i_p, which produce or consume the same state s, need to be aligned, and we have the following two alternative ways of writing these constraints, (15) and (16) or (15A) and (16A):

$$T^s(i_p,n) \ge T^s(i'_p,n) - H(1-w(i'_p,n)) \quad \forall s \in S^{IN},\ i_p \in I_s,\ i'_p \in I_s,\ i_p \ne i'_p,\ \rho^{p}_{si'_p} > 0,\ \rho^{c}_{si_p} < 0,\ n \in N, \quad (15)$$

$$T^f(i_p, n) \geq T^f(i'_p, n) - H(1 - w(i'_p, n))$$
$$\forall s \in S^{IN},\ i_p \in I_s,\ i'_p \in I_s,\ i_p \neq i'_p,\ \rho^p_{si'_p} > 0,$$
$$\rho^c_{si_p} < 0,\ n \in N\,, \quad (16)$$

$$T^s(i_p, n) \geq T^s(i'_p, n) - H(2 - w(i'_p, n)$$
$$- w(i_p, n)) \quad \forall s \in S^{IN},\ i_p \in I_s,\ i'_p \in I_s,$$
$$i_p \neq i'_p,\ \rho^p_{si'_p} > 0,\ \rho^c_{si_p} < 0,\ n \in N\,, \quad (15A)$$

$$T^f(i_p, n) \geq T^f(i'_p, n) - H(2 - w(i'_p, n)$$
$$- w(i_p, n)) \quad \forall s \in S^{IN},\ i_p \in I_s,\ i'_p \in I_s,$$
$$i_p \neq i'_p,\ \rho^p_{si'_p} > 0,\ \rho^c_{si_p} < 0,\ n \in N\,. \quad (16A)$$

In these constraints the start and end times of the downstream (consuming) processing tasks are enforced to be later than the corresponding times of the upstream (producing) processing tasks that process the same state *s*. Constraints (15) and (16) enforce the condition that the start and end times of the consuming tasks need to be always aligned to those of the corresponding producing tasks whenever the producing task is active; (15A) and (16A) state that the start and end times of the corresponding processing tasks need to be aligned conditionally, when both tasks are active.

Constraints (15) and (16) are required when we use the variable $ST(s, n)$ in the material balance constraint (7), for instance, when we do not consider storage as a separate task for the unlimited intermediate storage case (or for the dedicated finite intermediate storage case discussed later). Otherwise, if we do not have $ST(s, n)$ (for instance, for the no intermediate storage case discussed later) or if we consider storage as a separate task and use the variable $B(i_{st}, n)$ instead of $ST(s, n)$ in the material balance (for instance, for the dedicated and flexible finite intermediate storage cases discussed later), then we need constraints (15A) and (16A). The reason for this is that when we do not consider storage as a separate task and use the variable $ST(s, n)$ in the material balance constraint (7), there is an implicit assumption that when the downstream consuming task starts at event *n*, the amount stored in $ST(s, n-1)$ would be available for consumption, which may not always be valid (because of the heterogeneous locations of events used), unless the consuming tasks are always aligned to the producing tasks whenever the producing tasks are active, as described in the sequencing constraints (15) and (16). It should be noted that the fact that the model of Ierapetritou and Floudas [8] for continuous processes results in the reported approximate solutions [1,5,20] is due to lack of the second constraint, (16) or (16A), relating the end times of the producing and consuming tasks.

Extra Tightening Constraint. The following tightening constraint gives a better linear programming relaxation:

$$\sum_{n \in N} \sum_{i \in I_j} (T^f(i, n) - T^s(i, n)) \leq H - \tau_j^{min} \quad \forall j \in J\,. \quad (17)$$

It states that the sum of the durations of all tasks suitable in unit *j* is limited by the available time in the horizon ($H - \tau_j^{min}$), where τ_j^{min} is a lower bound on the total cleanup time required in unit *j*.

Dedicated Finite Intermediate Storage with Storage Bypassing Allowed

Next, consider the case where a dedicated finite intermediate storage is available for each intermediate state *s*. For this case as well, there is no need to model the storage tasks explicitly, because the storage tasks are anyway dedicated in nature. We are only interested in constraining the finite nature of the intermediate storage. Hence, the model is described here only using the continuous processing tasks i_p, and the other case when storage is considered as a separate task is discussed in Sect. "Flexible Finite Intermediate Storage with Storage Bypassing Allowed." All the above constraints for the case of unlimited storage remain the same except for the constraints for different tasks in different units for processing tasks. Here, since we do not consider storage as a separate task, we use constraints (15) and (16). Additional constraints would be required depending on whether storage bypassing is allowed or not. Initially, we consider the case when storage bypassing is allowed and then discuss the other case in Sect. "Dedicated and Flexible Finite Intermediate Storage Without Storage Bypassing".

Storage Bypassing Allowed. When an intermediate material is allowed to go directly to the production task

bypassing a finite intermediate storage or if there is no intermediate storage available, then to ensure the feasibility of the inventory capacity balance, the start and finish times of processing tasks i'_{p} and i_{p}, which produce and consume the same state s, respectively, need to be the *same* if both tasks are active at the same event point as shown in the following constraints:

$$T^s(i_{\mathrm{p}}, n) \le T^s(i'_{\mathrm{p}}, n) + H(2 - w(i'_{\mathrm{p}}, n) - w(i_{\mathrm{p}}, n)) \quad \forall s \in S^{\mathrm{FIS}},\ i_{\mathrm{p}} \in I_s,\ i'_{\mathrm{p}} \in I_s,\ i_{\mathrm{p}} \neq i'_{\mathrm{p}},\ \rho^{\mathrm{p}}_{si'_{\mathrm{p}}} > 0,\ \rho^{c}_{si_{\mathrm{p}}} < 0,\ n \in N\,, \quad (18)$$

$$T^f(i_{\mathrm{p}}, n) \le T^f(i'_{\mathrm{p}}, n) + H(2 - w(i'_{\mathrm{p}}, n) - w(i_{\mathrm{p}}, n)) \quad \forall s \in S^{\mathrm{FIS}},\ i_{\mathrm{p}} \in I_s,\ i'_{\mathrm{p}} \in I_s,\ i_{\mathrm{p}} \neq i'_{\mathrm{p}},\ \rho^{\mathrm{p}}_{si'_{\mathrm{p}}} > 0,\ \rho^{c}_{si_{\mathrm{p}}} < 0,\ n \in N\,. \quad (19)$$

So, from constraints (15), (16), (18), and (19), both the start and the end times of the producing and the consuming tasks of the same state s would be the same, if both tasks are active at event n. If either of the tasks is not active, then constraints (18) and (19) are relaxed and are trivially satisfied. This zero-wait condition is required to ensure the feasibility of the inventory capacity balance as illustrated in Shaik and Floudas [27], because of the unit-specific nature of the continuous-time representation used. The formulation of Ierapetritou and Floudas [8] did not take this into account. In the formulation of Giannelos and Georgiadis [6], this condition was enforced even for batch plants as well, which is unrealistic, and hence their formulation led to suboptimal solutions as observed by Sundaramoorthy and Karimi [29] and Shaik et al. [28].

Now, to constrain the finite nature of the intermediate storage available, the following bounds are added for the states that have the finite storage requirements:

$$ST(s, n) \le ST^{\max}_s \quad \forall s \in S^{\mathrm{FIS}},\ n \in N\,. \quad (20)$$

No Intermediate Storage

For the case when no intermediate storage is available, the excess amount of state s, $ST(s, n)$, is driven to zero at each event n or simply this variable is eliminated. Then, the material balance constraints (7) and (8) reduce to the following, meaning that the amount of state s produced at an event has to be consumed at the same event:

$$\sum_{i_{\mathrm{p}} \in I_s} \rho^{\mathrm{p}}_{si_{\mathrm{p}}} b(i_{\mathrm{p}}, n) + \sum_{i_{\mathrm{p}} \in I_s} \rho^{c}_{si_{\mathrm{p}}} b(i_{\mathrm{p}}, n) = 0 \quad \forall s \in S^{\mathrm{IN}},\ n \in N\,. \quad (21)$$

The condition of enforcing the same start and end times of the producing and consuming tasks of the same state s, described in constraints (15A), (16A), (18), and (19) is again applicable because no intermediate storage is available. Here we use (17) and (18) because in the material balance constraint (21), there is no assumption that the consuming task will receive material from the storage at the previous event, and hence there is no need to enforce the alignment unconditionally.

Flexible Finite Intermediate Storage with Storage Bypassing Allowed

This is a general case where finite intermediate storage is available and for each material state several suitable storage options exist. A material state can be stored in all or a limited number of storage units and vice versa. To handle this general case we need to introduce separate tasks for the storage activity, because the storage cannot have more than one state at any time and we need to accommodate additional constraints for relating the timing of storage tasks (i_{st}) to that of the processing tasks (i_{p}). The nature of these constraints would be different depending on whether storage bypassing is allowed or not. Initially, we consider the case when storage bypassing is allowed and then discuss the other case in Sect. "Dedicated and Flexible Finite Intermediate Storage Without Storage Bypassing." For the dedicated finite intermediate storage case, in contrast to the model discussed in Sect."Dedicated Finite Intermediate Storage with Storage Bypassing Allowed," if we alternatively chose to consider storage as a separate task, then the following same model would be applicable.

The additional constraints in the mathematical model are described below for the case of flexible finite intermediate storage when storage bypassing is allowed. The allocation constraint in (1) remains the same except that now it is written over all units (both processing and storage).

Allocation Constraints for Storage Tasks.

$$w(i_{st}, n+1) \geq w(i_{st}, n) + z(i_{st}, n) - 1 \quad \forall i_{st} \in I_{st},\ n \in N \tag{22}$$

Constraint (22) states that if a storage task is active and it stores a nonzero amount at event n, then the same storage task should be active at the next event $n+1$ as well. Additionally, this constraint also avoids unnecessary tank-to-tank transfer of material because the same storage task would be active at the next event as well.

Capacity Constraints for Storage Tasks

$$b(i_{st}, n) \leq V_{i_{st}}^{max} w(i_{st}, n) \quad \forall i_{st} \in I_{st},\ n \in N \tag{23}$$

$$b(i_{st}, n) \leq V_{i_{st}}^{max} z(i_{st}, n) \quad \forall i_{st} \in I_{st},\ n \in N \tag{24}$$

The capacity constraints for processing tasks remain the same as in (2) or (3), while for storage tasks the amount of material that can be stored is limited by the available capacity of the corresponding storage unit as shown in constraint (23). Constraint (24) is the same as (25), but uses a different binary variable $z(i_{st}, n)$ to confine only those instances when $b(i_{st}, n) \neq 0$, which is realized through the penalty term on the number of binary variables in the objective function described in (45).

Material Balances Constraint (4), for calculating the amount of raw material as and when required from the external resources, remains the same. For the other case when the entire raw material requirement is supplied at the beginning of the scheduling horizon, constraints (5) and (6) are modified as follows:

$$\sum_{i_{st} \in I_{st}} \rho_{s i_{st}}^{p} b(i_{st}, n-1) + \sum_{i \in I_s} \rho_{si}^{c} b(i, n) = 0 \quad \forall s \in S^{R}, n \in N, n > 1\,, \tag{25}$$

$$ST_0(s) + \sum_{n=1} \sum_{i \in I_s} \rho_{si}^{c} b(i, n) = 0 \quad \forall s \in S^{R}\,, \tag{26}$$

where the set I_s consists of both processing and storage tasks. The variable $ST(s, n)$ is eliminated here because separate storage tasks are defined explicitly. Similarly, the material balances in (7) and (8) for the intermediates are modified as given below:

$$\sum_{i_p \in I_s} \rho_{s i_p}^{p} b(i_p, n) + \sum_{i_{st} \in I_s} \rho_{s i_{st}}^{p} b(i_{st}, n-1) + \sum_{i \in I_s} \rho_{si}^{c} b(i, n) = 0 \quad \forall s \in S^{FIS}, n \in N, n > 1\,, \tag{27}$$

$$\sum_{i_p \in I_s} \rho_{s i_p}^{p} b(i_p, n) + ST_0(s) + \sum_{i \in I_s} \rho_{si}^{c} b(i, n) = 0 \quad \forall s \in S^{FIS}, n = 1\,, \tag{28}$$

where the set I_s consists of both processing and storage tasks. These constraints are based on the same condition that an intermediate material is allowed to go directly to the production task bypassing the storage. The demand constraint in (9) remains the same.

Duration Constraints for Storage Tasks

$$T^{f}(i_{st}, n) \geq T^{s}(i_{st}, n) \quad \forall i_{st} \in I_{st},\ n \in N \tag{29}$$

The duration constraint given in (10) remains the same for processing tasks, while for the storage tasks the finish times have to be later than the start times as shown in (29).

The sequencing constraints, (11)–(14), (15A), (16A), (18), (19) for the same task in the same unit, different tasks in the same unit, and different tasks in different units for the processing tasks remain the same. Here, because storage is considered as a separate task we use constraints (15A) and (16A). The sequencing constraints for storage tasks are defined below.

Sequencing Constraints. Different Tasks in Different Units for Storage Tasks The start time of a storage task that stores the intermediate state s should be same as the start time of the processing task that either produces or consumes state s as follows:

$$T^{s}(i_{st}, n) \geq T^{s}(i_p, n) - H(2 - w(i_{st}, n) - w(i_p, n)) \quad \forall s \in S^{IN}, i_{st} \in I_s, i_p \in I_s, n \in N\,, \tag{30}$$

$$T^{s}(i_{st}, n) \leq T^{s}(i_p, n) + H(2 - w(i_{st}, n) - w(i_p, n)) \quad \forall s \in S^{IN},\ i_{st} \in I_s,\ i_p \in I_s,\ n \in N\,. \tag{31}$$

If the corresponding processing task that either produces or consumes state s is not active ($w(i_p, n) = 0$), and there is a nonzero amount in the storage ($z(i_{st}, n) = 1$), then the start time of the storage task should be equal to the finish time of the same task at the previous event and is realized through constraints (32) and (11).

$$T^s(i_{st}, n) \le T^f(i_{st}, n-1) + H(1 - z(i_{st}, n) + w(i_p, n)) \quad \forall s \in S^{IN},\ i_{st} \in I_s,\ i_p \in I_s,\ n \in N, n > 1 \quad (32)$$

The finish time of the storage task for storing state s should be greater than the finish times of both the producing and the consuming tasks of state s as follows:

$$T^f(i_{st}, n) \ge T^f(i_p, n) - H(2 - w(i_{st}, n) - w(i_p, n)) \quad \forall s \in S^{IN},\ i_{st} \in I_s,\ i_p \in I_s,\ n \in N\ . \quad (33)$$

Additionally, if the amount stored in the storage is zero ($z(i_{st}, n) = 0$), then the finish time of the storage task should be same as the finish time of the corresponding consuming processing task and is realized from constraints (33) and (34).

$$T^f(i_{st}, n) \le T^f(i_p, n) + H(2 - w(i_{st}, n) - w(i_p, n)) + Hz(i_{st}, n) \quad \forall s \in S^{IN},\ i_{st} \in I_s,\ i_p \in I_s,\ \rho^c_{si_p} < 0,\ n \in N \quad (34)$$

However, if the amount stored in the storage is nonzero ($z(i_{st}, n) = 1$), then it should remain in the storage until the start time of the processing task at the next event. So, the finish time of the storage task should be same as the start time of the next processing task and is realized from constraints (35) and (36):

$$T^f(i_{st}, n) \ge T^s(i_p, n+1) - H(2 - w(i_{st}, n) - w(i_p, n+1)) - H(1 - z(i_{st}, n))\ , \quad \forall s \in S^{IN},\ i_{st} \in I_s,\ i_p \in I_s,\ n \in N,\ n \ne N\ , \quad (35)$$

$$T^f(i_{st}, n) \le T^s(i_p, n+1) + H(2 - w(i_{st}, n) - w(i_p, n+1)) + H(1 - z(i_{st}, n))\ , \quad \forall s \in S^{IN},\ i_{st} \in I_s,\ i_p \in I_s,\ n \in N,\ n \ne N\ . \quad (36)$$

The tightening constraint given in (17) is again applicable.

Dedicated and Flexible Finite Intermediate Storage Without Storage Bypassing

The nature of constraints is different when storage bypassing is not allowed for the cases where production must go through finite storage (dedicated or flexible) before consumption in the downstream units. This is a general case and we do not need to enforce the same start and end times for producing or consuming tasks of the same state, because the material always goes through storage.

Dedicated Finite Intermediate Storage Case Without Bypassing of Storage Here again, we have the option of considering storage as a separate task or not. In this section, we describe the model without considering storage as a separate task, and the other case when storage is considered as a separate task is discussed in the next section along with the flexible finite intermediate storage case.

When we do not consider storage as a separate task, the model comprises all the constraints, (1)–(14), (15), (16), and (17), discussed in Sect. "Unlimited Intermediate Storage." Additionally we need constraint (20) and the following material balance constraints for the intermediate states.

Material Balance for Intermediates

$$ST(s, n-1) + \sum_{i_p \in I_s} \rho^p_{si_p} b(i_p, n) \le ST^{max}_s \quad \forall s \in S^{FIS},\ n \in N,\ n > 1 \quad (37)$$

$$ST_0(s) + \sum_{i_p \in I_s} \rho^p_{si_p} b(i_p, n) \le ST^{max}_s \quad \forall s \in S^{FIS},\ n \in N,\ n = 1 \quad (38)$$

In constraints (37) and (38), the total amount received into the dedicated storage at each event is constrained to be within the maximum capacity limits.

Flexible Finite Intermediate Storage Case Without Bypassing of Storage For the dedicated finite intermediate storage case as well, in contrast to the model discussed in the previous section, if we alternatively chose to consider storage as a separate task, then the following same model would be applicable for both

the dedicated and the flexible finite storage cases. Constraints (1), (2), (4), (9)–(11), (14), (16A), (17), (22)–(24), (27)–(29), and (32)–(36), discussed in the previous sections, are required. Constraint (15A) is not required as it is implicitly enforced, because consuming tasks are aligned with storage tasks, which in turn are aligned with the producing tasks. Additionally, we need the following allocation, material balance, and sequencing constraints.

Allocation Constraints

$$\sum_{i_{st} \in \rho^c_{s i_{st}}} w(i_{st}, n) \geq w(i_p, n) \qquad \forall s \in S^{FIS},\ i_p \in \rho^p_{s i_p},\ n \in N \quad (39)$$

Constraint (39) states that whenever a producing task of state s is active then one or more of suitable storage tasks also need to be active for all intermediate states that have the finite storage requirement.

Material Balance for Intermediates Here, because the storage tasks are modeled explicitly, constraints (37) and (38) are modified as follows:

$$\sum_{i_{st} \in I_s} \rho^p_{s i_{st}} b(i_{st}, n-1) + \sum_{i_p \in I_s} \rho^p_{s i_p} b(i_p, n) \leq \sum_{i_{st} \in I_s} V^{max}_{i_{si}} w(i_{st}, n) \quad \forall s \in S^{FIS},\ n \in N,\ n > 1, \quad (40)$$

$$ST_0(s) + \sum_{i_p \in I_s} \rho^p_{s i_p} b(i_p, n) \leq \sum_{i_{st} \in I_s} V^{max}_{i_{si}} w(i_{st}, n) \quad \forall s \in S^{FIS},\ n = 1\,. \quad (41)$$

Sequencing Constraints. Different Tasks in Different Units

$$T^s(i_{st}, n) \geq T^s(i_p, n) - H(2 - w(i_{st}, n) - w(i_p, n)) \quad \forall s \in S^{IN},\ i_{st} \in I_s,\ i_p \in \rho^p_{s i_p},\ n \in N \quad (42)$$

$$T^s(i_{st}, n) \leq T^s(i_p, n) + H(2 - w(i_{st}, n) - w(i_p, n)) \quad \forall s \in S^{IN},\ i_{st} \in I_s,\ i_p \in \rho^p_{s i_p},\ n \in N \quad (43)$$

$$T^s(i_p, n) \geq T^s(i_{st}, n) - H(2 - w(i_{st}, n) - w(i_p, n)) \quad \forall s \in S^{IN},\ i_{st} \in I_s,\ i_p \in \rho^c_{s i_p},\ n \in N \quad (44)$$

Constraints (42) and (43) impose the requirement that the start time of the storage task is the same as the start time of the corresponding processing task that produces the intermediate state, if both the storage task and the producing task are active. Constraint (44) states that the start time of the processing task that consumes the intermediate state should be later than the start time of the corresponding storage task if both tasks are active.

Objective: Maximization of Profit

$$C_1 \sum_{s \in S^P} \sum_{n \in N} price_s \sum_{i \in I_s} \rho^p_{si} b(i, n) - C_2 \sum_{i \in I} \sum_{n \in N} w(i, n) - C_3 \sum_{i_{st} \in I_{st}} \sum_{n \in N} z(i_{st}, n) \quad (45)$$

The objective is maximization of profit due to sales from the production of final products, and additionally there are penalties for the total number of binary variables as shown in (45), where C_i is the corresponding cost coefficient.

Objective: Minimization of Makespan For the objective of minimization of the makespan (MS) the following constraints need to be added:

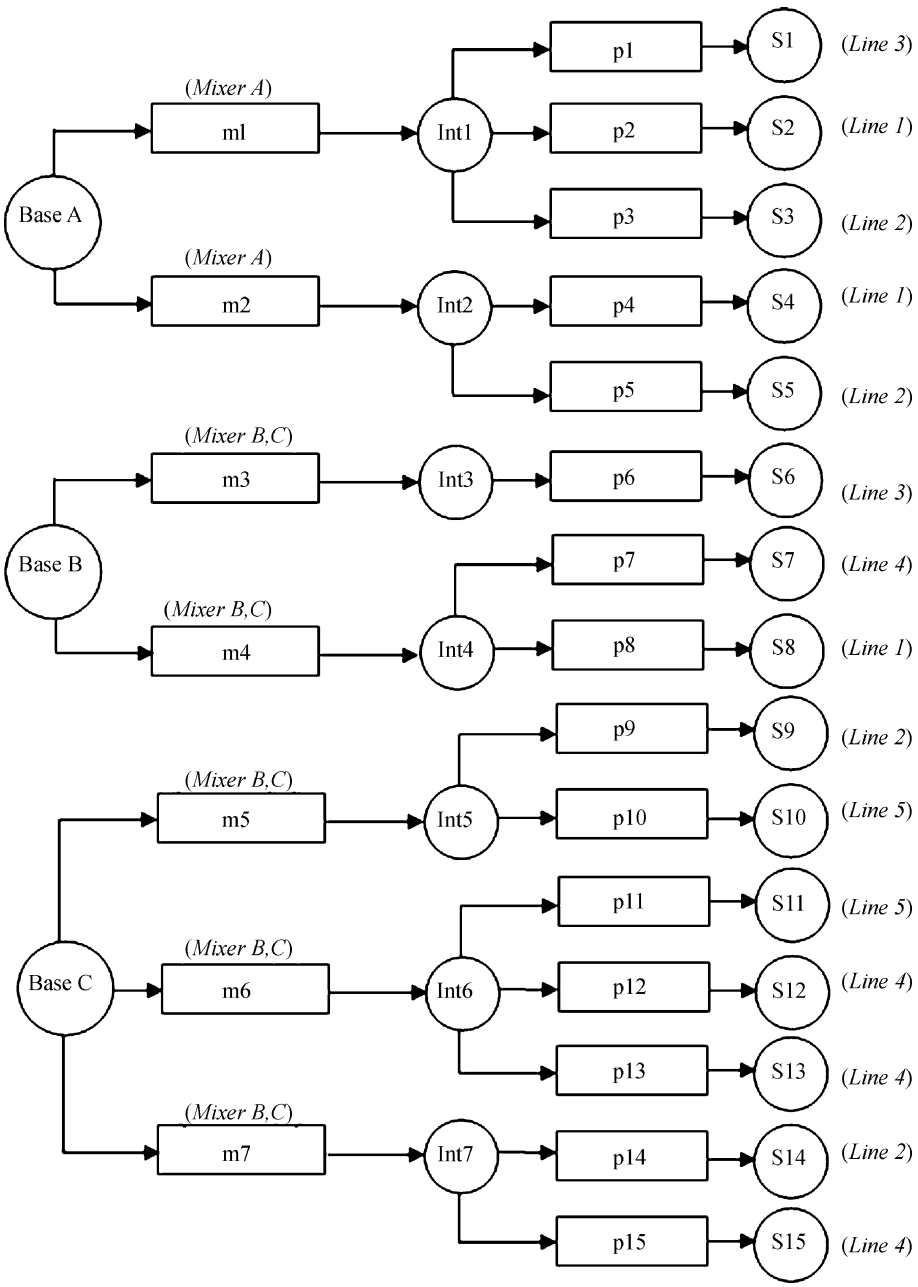
$$\text{Minimize MS and } T^f(i, n) \leq MS \quad \forall i \in I,\ n \in N\,. \quad (46)$$

The tightening constraint in (17) is modified as follows:

$$\sum_{n \in N} \sum_{i \in I_j} (T^f(i, n) - T^s(i, n)) \leq MS - \tau^{min}_j \quad \forall j \in J\,. \quad (47)$$

Computational Case Study

An industrial case study of a fast moving consumer goods manufacturing plant that has been extensively studied by several authors [1,2,5,8,20,27] is considered. The plant follows a common production sequence: mixing, storage, and packing. The mixing stage has



Short-Term Scheduling of Continuous Processes, Figure 1
State-task-network representation of the plant

three parallel mixers (mixers A, B, and C) operating in a continuous mode that produce seven intermediates (Int1–Int7) using three different base stocks (bases A, B, and C) available as required. These intermediates may be stored in three storage tanks (tanks A, B, and C) or directly packed in five continuous packing lines (lines 1–5), thus producing 15 final products (S1–S15). The STN representation of the plant is shown in Fig. 1 along with the unit suitability for each task.

The base stocks are available in unlimited initial amounts. For each task suitable on multiple units, a separate task is considered. For instance, two making tasks

Short-Term Scheduling of Continuous Processes, Table 1
Production rates

Task (i_p)	Unit (J_p)	R_i^{max} (ton/h)
m1	Mixer A	17
m2	Mixer A	17
m31	Mixer B	17
m32	Mixer C	17
m41	Mixer B	17
m42	Mixer C	12.24
m51	Mixer B	12.24
m52	Mixer C	12.24
m61	Mixer B	12.24
m62	Mixer C	12.24
m71	Mixer B	12.24
m72	Mixer C	12.24
p1	Line 3	5.5714
p2	Line 1	5.8333
p3	Line 2	2.7083
p4	Line 1	5.8333
p5	Line 2	2.7083
p6	Line 3	5.5714
p7	Line 4	2.2410
p8	Line 1	5.8333
p9	Line 2	2.7083
p10	Line 5	5.3571
p11	Line 5	5.3571
p12	Line 4	3.3333
p13	Line 4	2.2410
p14	Line 2	2.7083
p15	Line 4	3.3333

Short-Term Scheduling of Continuous Processes, Table 2
Cleanup times

Changeover ($i'_p \leftrightarrow i_p$)	Unit (J_p)	$\tau^{cl}_{i'i}$	τ_i^{min}
(p2, p4) ↔ p8	Line 1	1	1
(p3, p5) ↔ (p9, p14)	Line 2	4	4
p1 ↔ p6	Line 3	1	1
(p12, p15) ↔ (p13, p7)	Line 4	2	2

Short-Term Scheduling of Continuous Processes, Table 3
State-specific data

State(s)	$ST_0(s)$	ST_s^{max}	D_s^{min}	$Price_s$
Base A, B, C	∞			
Int1–Int7		60		
S1			220	1
S2			251	1
S3			116	1
S4			15	1
S5			7	1
S6			47	1
S7			8.5	1
S8			144	1
S9			42.5	1
S10			114.5	1
S11			53	1
S12			2.5	1
S13			16.5	1
S14			13.5	1
S15			17.5	1

(m31 and m32) are considered for producing "Int3" using mixers B and C, respectively. The 15 final products are produced using 15 packing tasks (p1–p15). The problem data such as production rates, task-unit suitability, and cleanup times used in the literature [5,27] are given in Tables 1 and 2. The state-specific data such as minimum demand specifications and prices for the final products, and storage limitations are shown in Table 3. A time horizon of 120 h is considered.

The case study with finite intermediate storage is considered. We compare the results from the literature [1,2,5,8,20], except for the model of Giannelos and Georgiadis [5], for which the comparison is based on our implementation of their model. All the computations in this work were performed using a 3.2 GHz Pentium 4 machine with 1 GB RAM using GAMS (distribution 21.7) and CPLEX 9.0.2, and the case studies were solved with an integrality gap of less than 0.3%.

Finite Intermediate Storage. Consider the case of flexible finite intermediate storage for all the intermediates. Three storage tanks (tanks A, B, and C) are available each with a maximum capacity of 60 ton, and any intermediate can be stored in any of the three tanks. Since there are seven intermediates, 21 additional storage tasks need to be considered. Unlike in the Shaik and Floudas model, the start and end times of storage tasks are not precisely calculated in the model of Giannelos and Georgiadis [5]; rather it seems that they determine and/or adjust these timings during postprocessing. In our implementation of their model, we used the following equation to calculate the start times of all active

storage tasks during postprocessing:

$$T^{s}_{it} = \tau_{it} - \theta_{\mathrm{st}}, \quad \forall s \in S^{fis},\ i \in I^{\mathrm{st}}_{s},\ t \in T\,. \qquad (48)$$

Then, in order to avoid the unnecessary activation of storage tasks in our implementation of their model, we used the following objective function, which is similar to (45) of the Shaik and Floudas model:

$$C_1 \sum_{s \in S^{P}} \mathrm{price}_{s} ST(s, t_n) - C_2 \sum_{i \in I} \sum_{t \in T} x(i,t) - C_3 \sum_{s \in S^{fis}} \sum_{t \in T} y(s,t)\,. \qquad (49)$$

The model statistics are reported in Table 4. For the objective function of maximization of profit in (45) with $C_1 = 10$ and $C_2 = C_3 = 1$, the Shaik and Floudas model requires four events, and the objective function is found to be 26910.181 in 465.61 CPU s with an integrality gap of 0.12%, which corresponds to a profit of 2695.32. The Gantt chart for the Shaik and Floudas model is depicted in Fig. 2.

This case is regarded as one of the hardest instances to solve in the literature. The Shaik and Floudas model finds the global optimum solution and requires consideration of only four events compared with the global event-based model of Castro et al. [1], which reported ten events for this case as shown in Table 4. The campaign-based algorithmic model of Mendez and Cerda [20], although it is very compact in terms of the fewest problem statistics, could not find the global optimal solution for this case (the best solution reported corresponds to a profit of 2670.28). The formulation of Giannelos and Georgiadis [5] also reported a suboptimal solution corresponding to a profit of 2689.48 using four events. On the basis of our implementation of their model, an improved objective value of 2692.06 was found within a maximum CPU time of 60000 s and an integrality gap of 0.12%. However, in the Gantt chart it was observed that the end times of some storage tasks are not precisely calculated as discussed in Shaik and Floudas [27].

Short-Term Scheduling of Continuous Processes, Table 4
Results for the case study: finite intermediate storage

	Shaik and Floudas[27]	Castro et al.[1]	Giannelos and Georgiadis[5]	Mendez and Cerda[20]
Events	4	10	4	
Binary variables	276	330	220	60
Continuous variables	580	927	712	87
Constraints	4267	1127	2113	361
Nonzeros	21130		6884	
RMILP	26946.72	2695.32	2695.32	
MILP	26910.18	2695.32	2692.06	2670.28
Profit	2695.32	2695.32	2692.06	2670.28
Integrality gap (%)	0.12	0	0.12	
CPU time (s)	465.61	163[a]	60000	399[a]
Nodes	7144	3934	3307486	

[a]CPU time for other models reported for completeness only

Conclusions

In this study, the formulation of Shaik and Floudas [27], an improved model for short-term scheduling of continuous processes based on a STN representation using a unit-specific event-based continuous-time formulation, was presented. The model of Ierapetritou and Floudas [8] for continuous processes was modified and extended to precisely account for different storage requirements, such as dedicated, flexible, unlimited, and no intermediate storage policies. The efficacy of their formulation was demonstrated for an industrial case study from a fast moving consumer goods manufacturing process reported in the literature.

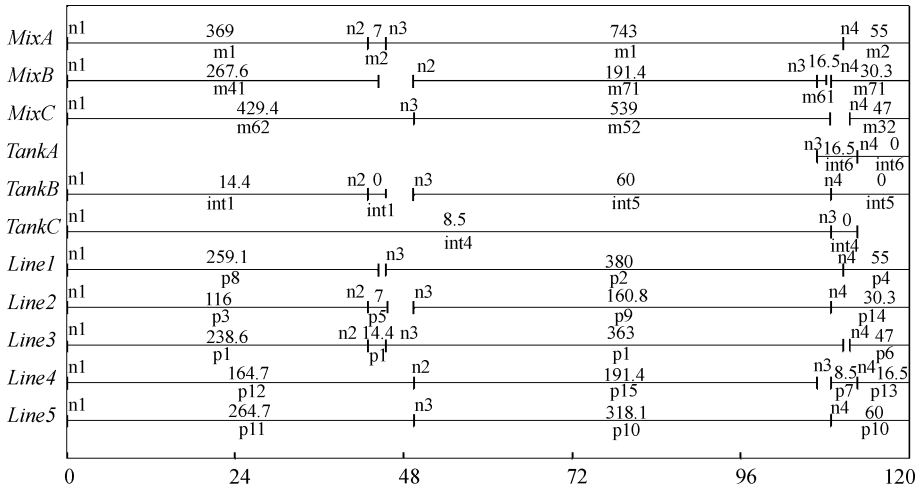
Nomenclature

Indices

i tasks
i_{p} processing tasks
i_{st} storage tasks
j units
n events indicating beginning of a task
s states

Sets

I tasks
I_{p} processing tasks
I_{st} storage tasks
I_j tasks which can be performed in unit j
I_s tasks which process state s and either produce or consume



Short-Term Scheduling of Continuous Processes, Figure 2
Gantt chart for the finite intermediate storage case

J units (both processing and storage)
J_{p} processing units
J_{st} storage units
J_i units which are suitable for performing task i
N event points within the time horizon
S states
S^{R} states that are raw materials
S^{IN} states that are intermediates
S^{FIS} intermediate states that have finite storage requirements
S^{P} states that are final products

Parameters

$R_i^{\min}$ minimum processing rate of task i, ton/h
$R_i^{\max}$ maximum processing rate of task i, ton/h
R_i processing rate of task i if it is constant, ton/h
$V_{i_{\mathrm{st}}}^{\max}$ maximum storage capacity for storage task i_{st}, ton
H time horizon, h
price_s price of state s
$D_s^{\min}$ lower bound on demand for state s, ton
$D_s^{\max}$ upper bound on demand for state s, ton
τ_j sequence independent cleanup time required in unit j, h
$t_{ii'}^{\mathrm{cl}}$ sequence dependent cleanup time required between tasks i and i', h
$\tau_j^{\min}$ minimum total cleanup time required for unit j, h
$\rho_{si}^{\mathrm{p}}, \rho_{si}^{c}$ proportion of state s produced, consumed from tasks i, respectively, $\rho_{si}^{\mathrm{p}} \geq 0,\ \rho_{si}^{c} \leq 0$

Variables

$w(i, n)$ binary variable for assignment of task i at the beginning of event n
$z(i_{\mathrm{st}}, n)$ binary variable to determine if storage task i_{st} stores a nonzero amount at event n
$b(i, n)$ amount of material undertaking task i at event n, ton
$ST_0(s, n)$ amount of state $s \in S^R$ that is required from external resources at event n, ton
$ST(s, n)$ excess amount of state s that needs to be stored at event n, ton
$T^s(i, n)$ time at which task i starts at event n, h
$T^f(i, n)$ time at which task i ends at event n, h

References

1. Castro PM, Barbosa-Povoa AP, Matos HA, Novais AQ (2004) Simple continuous-time formulation for short-term scheduling of batch and continuous processes. Ind Eng Chem Res 43:105–118
2. Castro PM, Barbosa-Povoa AP, Novais AQ (2004) A divide-and-conquer strategy for the scheduling of process plants subject to changeovers using continuous-time formulations. Ind Eng Chem Res 43:7939–7950
3. Floudas CA, Lin X (2004) Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. Comput Chem Eng 28:2109–2129

4. Floudas CA, Lin X (2005) Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. Ann Oper Res 139:131–162
5. Giannelos NF, Georgiadis MC (2002) A novel event-driven formulation for short-term scheduling of multipurpose continuous processes. Ind Eng Chem Res 41:2431–2439
6. Giannelos NF, Georgiadis MC (2002) A simple new continuous-time formulation for short-term scheduling of multipurpose batch processes. Ind Eng Chem Res 41:2178–2184
7. Ierapetritou MG, Floudas CA (1998) Effective continuous-time formulation for short-term scheduling: 1. Multipurpose batch processes. Ind Eng Chem Res 37:4341–4359
8. Ierapetritou MG, Floudas CA (1998) Effective continuous-time formulation for short-term scheduling: 2. Continuous and semi-continuous processes. Ind Eng Chem Res 37:4360–4374
9. Ierapetritou MG, Hene TS, Floudas CA (1999) Effective continuous-time formulation for short-term scheduling: 3. Multiple intermediate due dates. Ind Eng Chem Res 38:3446–3461
10. Janak SL, Floudas CA, Kallrath J, Vormbrock N (2006) Production scheduling of a large-scale industrial batch plant. I. Short-term and medium-term scheduling. Ind Eng Chem Res 45:8234–8252
11. Janak SL, Floudas CA, Kallrath J, Vormbrock N (2006) Production scheduling of a large-scale industrial batch plant. II. Reactive scheduling. Ind Eng Chem Res 45:8253–8269
12. Janak SL, Lin X, Floudas CA (2007) A new robust optimization approach for scheduling under uncertainty: II. Uncertainty with known probability distribution. Comput Chem Eng 31:171–195
13. Janak SL, Lin X, Floudas CA (2004) Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: Resource constraints and mixed storage policies. Ind Eng Chem Res 43:2516–2533
14. Janak SL, Lin X, Floudas CA (2005) Comments on "Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: resource constraints and mixed storage policies.". Ind Eng Chem Res 44:426
15. Karimi IA, McDonald CM (1997) Planning and scheduling of parallel semi-continuous processes: 2. Short-term scheduling. Ind Eng Chem Res 36:2701–2714
16. Lin X, Floudas CA (2001) Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation. Comput Chem Eng 25:665–674
17. Lin X, Janak SL, Floudas CA (2004) A new robust optimization approach for scheduling under uncertainty: I. Bounded uncertainty. Comput Chem Eng 28:1069–1085
18. Lin X, Floudas CA, Modi S, Juhasz NM (2002) Continuous-time optimization approach for medium-range production scheduling of a multiproduct batch plant. Ind Eng Chem Res 41:3884–3906
19. Mc Donald CM, Karimi IA (1997) Planning and scheduling of parallel semi-continuous processes: 1. Production planning. Ind Eng Chem Res 36:2691–2700
20. Mendez CA, Cerda J (2002) An efficient MILP continuous-time formulation for short-term scheduling of multiproduct continuous facilities. Comput Chem Eng 26:687–695
21. Mockus L, Reklaitis GV (1999) Continuous-time representation approach to batch and continuous process scheduling: 1. MINLP formulation. Ind Eng Chem Res 38:197–203
22. Mockus L, Reklaitis GV (1999) Continuous-time representation approach to batch and continuous process scheduling: 2. Computational issues. Ind Eng Chem Res 38:204–210
23. Munawar SA, Bhushan M, Gudi RD, Belliappa AM (2003) Cyclic scheduling of continuous multi-product plants in a hybrid flowshop facility. Ind Eng Chem Res 42:5861–5882
24. Pinto JM, Grossmann IE (1994) Optimal cyclic scheduling of multistage continuous multiproduct plants. Comput Chem Eng 18:797–816
25. Sahinidis NV, Grossmann IE (1991) MINLP model for cyclic multiproduct scheduling on continuous parallel lines. Comput Chem Eng 15:85–103
26. Schilling G, Pantelides CC (1996) A simple continuous-time process scheduling formulation and a novel solution algorithm. Comput Chem Eng 20:S1221–S1226
27. Shaik MA, Floudas CA (2007) Improved unit-specific event-based model continuous-time model for short-term scheduling of continuous processes: Rigorous treatment of storage requirements. Ind Eng Chem Res 46:1764–1774
28. Shaik MA, Janak SL, Floudas CA (2006) Continuous-time models for short-term scheduling of multipurpose batch plants: A comparative study. Ind Eng Chem Res 45:6190–6209
29. Sundaramoorthy A, Karimi IA (2005) A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. Chem Eng Sci 60:2679–2702
30. Zhang X, Sargent RWH (1996) The optimal operation of mixed production facilities – A general formulation and some approaches for the solution. Comput Chem Eng 20:897–904
31. Zhang X, Sargent RWH (1998) The optimal operation of mixed production facilities – Extensions and improvements. Comput Chem Eng 22:1287–1295

Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks

MUNAWAR A. SHAIK, CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90B35, 65K05, 90C90, 90C11

Article Outline

Introduction
Motivation
Formulation
Allocation Constraints
Capacity Constraints
Material Balances
Duration Constraints
Sequencing Constraints
Tightening Constraint
Utility Related Constraints
Bounds on Variables
Objective Function
Computational Case Studies
Conclusions
Nomenclature
Indices
Sets
Parameters
Binary Variables
Positive Variables
References

Introduction

The research area of batch and continuous process scheduling has received great attention from both academia and industry in the past two decades. Floudas and Lin [2,3] presented extensive reviews comparing various discrete- and continuous-time-based formulations. During the last two decades, numerous formulations have been proposed in the literature based on continuous-time representation, due to their established advantages over discrete-time representations. On the basis of the time representation used, the different continuous-time models proposed in the literature can be broadly classified into three distinct categories: slot-based, global event-based, and unit-specific event-based formulations. In the slot-based models, the time horizon is represented in terms of ordered blocks of unknown, variable lengths, or time slots. Global event-based models use a set of events that are common across all units, and the event points are defined for either the beginning or end (or both) of each task in each unit. On the other hand, unit-specific event-based models define events on a unit basis, allowing tasks corresponding to the same event point but in different units to take place at different times. For sequential processes, other alternative approaches based on precedence relationships have also been used.

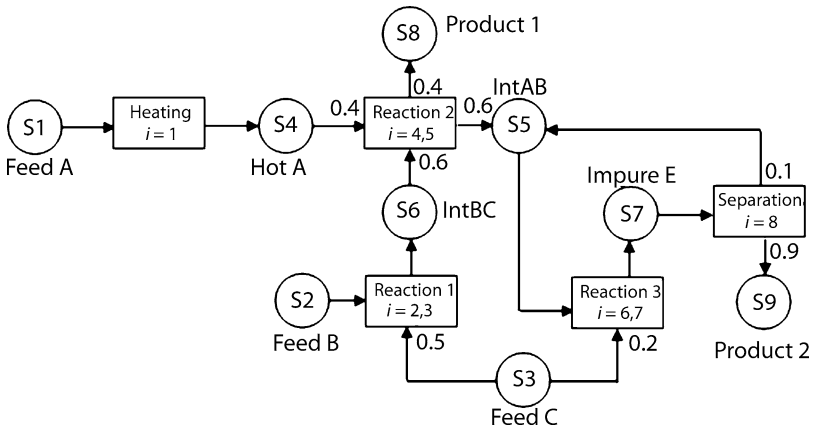
Motivation

A detailed comparison of various continuous-time models for short-term scheduling of batch plants was performed recently by Shaik et al. [8] They concluded that, due to heterogeneous locations of event points used, the unit-specific event-based models always require less event points and exhibit favorable computational performance compared to both slot-based and global event-based models. For problems that do not have resource considerations such as utility constraints, it was found [8] that the modified model of Ierapetritou and Floudas [4] as presented in Shaik et al. [8], outperforms the other models both in terms of least problem size and fast computational performance. Similarly, for problems with resource constraints the enhanced model of Janak et al. [5,6] was found [5,6,8] to perform well.

However, for an additional instance of the following example involving a recycle stream shown in Fig. 1, it is observed that the model of Ierapetritou and Floudas [4] yields a suboptimal solution as discussed below.

Example 1 Consider the second example discussed in Shaik et al. [8], in which two different products are produced through five processing stages: heating, reactions 1, 2, and 3, and separation, as shown in the STN representation of the plant flow sheet in Fig. 1. Since each of the reaction tasks can take place in two different reactors, each reaction is represented by two separate tasks. The processing time of task i on unit j is assumed to be a linear function, $\alpha_i + \beta_i B$, of its batch size, B. The relevant data [8] of the constant (α_i) and variable (β_i) coefficients for processing times of different tasks (i), the suitable units (j), and their minimum ($B_i^{\min}$) and maximum ($B_i^{\max}$) batch sizes are shown in Table 1. The initial stock level for all intermediates is assumed to be zero and unlimited storage capacity is assumed for all states. The prices of products 1 and 2 are \$10/mu.

For the objective of maximization of profit and a time horizon (H) of 10 h, this example is solved using the unit-specific event-based model of Ierapetritou and Floudas [4] (I&F), the global event-based models of Castro et al. [1](CBMN), and Maravelias and Gross-



Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Figure 1
State-task network representation for example 1

Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Table 1
Data of coefficients of processing times of tasks, limits on batch sizes of units for example 1

Task(i)		Unit(j)	α_i	β_i	B_i^{min}(mu)	B_i^{max}(mu)
Heating	($i=1$)	Heater	0.667	0.00667	–	100
Reaction1	($i=2$)	Reactor1	1.334	0.02664	–	50
	($i=3$)	Reactor2	1.334	0.01665	–	80
Reaction2	($i=4$)	Reactor1	1.334	0.02664	–	50
	($i=5$)	Reactor2	1.334	0.01665	–	80
Reaction3	($i=6$)	Reactor1	0.667	0.01332	–	50
	($i=7$)	Reactor2	0.667	0.008325	–	80
Separation	($i=8$)	Separator	1.3342	0.00666	–	200

mann [7] (M&G), and using the slot-based model of Sundaramoorthy and Karimi [10] (S&K). All the resulting mixed-integer linear programming (MILP) models are solved in GAMS distribution 21.1 using CPLEX 8.1.0 on the same computer (3 GHz Pentium 4 with 2 GB RAM) as in Shaik et al. [8] Table 2 provides a comparative study of these models in terms of the problem statistics such as the number of binary and continuous variables, number of constraints, CPU time taken to solve to the specified integrality gap, the number of nodes taken to reach the optimal solution, the objective function at the relaxed node, and so forth. It should be noted that for the S&K model, n event points are shown to represent $n-1$ slots for a valid comparison with the other global-event and unit-specific event-based models. In the CBMN model, there is an additional parameter (Δt) that defines a limit on the maximum number of events over which a task can occur.

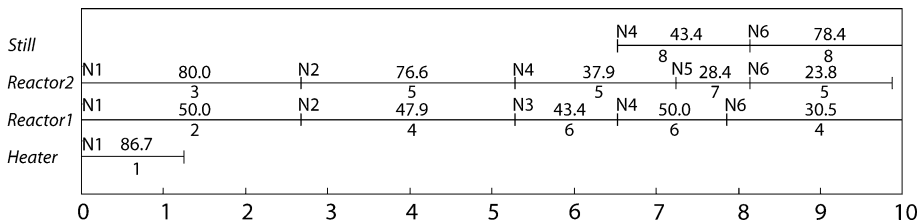
For this case, the slot-based and global event-based models require at least eight event points and are able to find the global optimal solution of 1962.7, compared to the unit-specific event-based I&F model, which gives a suboptimal solution of 1943.2 with six events and with higher events as well. When this case is solved using the enhanced unit-specific event-based model of Janak et al. [5,6], (JLF) it found the global optimal solution of 1962.7 using six events, as shown in Table 2. The Gantt chart for the JLF model is shown in Fig. 2.

The reason for this exception can be attributed to the fact that the I&F model does not allow a task to continue over several events, while the JLF model is an enhanced version of the I&F model that allows tasks to take place over multiple event points in order to accurately account for the resource considerations such as utility requirements. Although, there are no resource

Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Table 2
Model statistics and computational results for example 1 under maximization of profit

Model	Events	CPU Time (s)	Nodes	RMILP ($)	MILP ($)	Binary variables	Continuous variables	Constraints	Nonzeros
Example 1 ($H = 10$)									
S&K	8	105.5	88679	2690.6	1962.7	84	433	456	1615
M&G	8	507.64	184605	2690.6	1962.7	112	609	1402	4884
CBMN($\Delta t = 1$)	8	1.82	6449	2690.6	1860.7[a]	56	170	189	760
($\Delta t = 2$)	8	81.95	194968	3136.3	1959[a]	104	218	261	1238
($\Delta t = 3$)	8	207.43	366226	3136.3	1962.7	144	258	321	1635
I&F	6	2.16	6713	3078.4	1943.2[a]	34	140	267	859
	7	43.73	101415	3551.8	1943.2[a]	42	165	318	1046
JLF	6	322.20	138714	5284.5	1962.7	68	386	1500	5874

[a] Suboptimal solution



Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Figure 2
Gantt chart for example 1 using the JLF model

constraints in this example, it can be observed from the Gantt chart of Fig. 2 for the JLF model, that this schedule will not be feasible for the I&F model. According to the constraint for different tasks in different units of the I&F model, for state 's5', the consuming task ($i = 7$) at event 'N5' should start after the end time of the producing task ($i = 8$) at event 'N4', which is clearly not the case in the global optimal solution of Fig. 2. This constraint becomes relaxed in the JLF model because the producing task does not end at event 'N4', but it continues over the next event and ends at event 'N5' in the global optimal solution. The models of S&K, M&G, and CBMN allow tasks to take place over multiple events, and hence, are able to find the global optimal solution. Moreover, in these models the events/slots are globally aligned, and hence, they do not require the above-mentioned sequencing constraint for different tasks in different units, which is generally required for the unit-specific event-based models.

This example demonstrates that, although, there are no resource constraints, in some cases, it is a requirement for the unit-specific event-based models as well to allow tasks to take place over multiple events in order to find the global optimal solution. To understand such cases, let us examine the constraint for different tasks in different units that is used in the I&F model. It states that the consuming task at the current event should start after the end time of producing task at the previous event that processes the same state, which need not be true if there is sufficient material for the consuming task to start production, which happens to be the case in the particular instance of Fig. 2. The amount of state 's5' produced by the recycle stream ($i = 8$) at event 'N4' is not necessary for starting the consuming task ($i = 7$) at event 'N5'. So, this constraint needs to be relaxed depending on whether there is sufficient amount for the consuming task to start production or not, which is implicitly realized by allowing the tasks to take place over multiple events.

If we focus on the computational performance of the JLF model in Table 2, it has poor LP relaxation, and requires large number of constraints, nonzeros, and CPU time, compared to the other competitive models of S&K and CBMN. Because, the JLF model was originally developed to handle the more general case of problems with resource constraints, it can be observed that it does not reduce well, in terms of problem statistics, to the case of no resources. With this motivation, Shaik and Floudas [9], proposed a unified modeling approach using unit-specific event-based continuous-time representation, which (i) can handle problems with resource constraints by allowing tasks to take place over multiple events, (ii) efficiently reduces to the case of no resources, and (iii) is applicable for batch as well as continuous processes with mixed storage policies. The unified model of Shaik and Floudas [9] (S&F) for short-term scheduling of batch plants is described in the next section. In Sect. "Computational Case Studies", we consider examples of problems with and without resources and compare the performance of the S&F model with other models.

Formulation

The nomenclature used in the S&F formulation is given in the Nomenclature section. The three index binary variable $w(i,n,n')$ defines the assignment of task i that starts at event n and ends at event n' ($n' \geq n$). To exercise control on the maximum number of events over which a task is allowed to continue, a parameter, Δn, is defined such that $n \leq n' \leq n + \Delta n$, $\Delta n = 0, 1, \ldots$. So, the task i that starts at event n may end either at the same event point $n(\Delta n = 0)$, which would be similar to I&F model, or it may end at a later event $n + \Delta n$ ($\Delta n = 1, \ldots$), which would be similar to the model of Janak et al. [5,6] In the model of Castro et al. [1](CBMN) also such parameter (Δt) was defined. However, unlike in the S&F model [9], in all the slot-based and global event-based models [1,7,10], a task that starts at an event cannot end at the same event, thus by definition, the unit-specific event-based models [5,6,9] would require at least one event less compared to the slot-based and global event-based models.

The mathematical model has the following allocation, capacity, material balance, duration, sequencing, and utility related constraints.

Allocation Constraints

$$\sum_{i \in I_j} \sum_{n' \in N, n \leq n' \leq n + \Delta n} w(i,n,n') \leq 1 \quad \forall j \in J, n \in N \tag{1}$$

$$\sum_{i \in I_j} \sum_{n \in N, n \leq n' \leq n + \Delta n} w(i,n,n') \leq 1 \quad \forall j \in J, n' \in N, \Delta n > 0 \tag{2}$$

$$\sum_{i' \in I_j, i \neq i'} \sum_{n' \in N, n \leq n' \leq n + \Delta n} w(i',n',n) \leq 1 - \sum_{n' \in N, n \leq n' \leq n + \Delta n} w(i,n,n') \quad \forall i \in I_j, j \in J, n \in N, \Delta n > 0 \tag{3}$$

$$\sum_{n' \in N, n \leq n' \leq n + \Delta n} w(i,n,n') \leq 1 - \sum_{n' \in N, n' < n} \sum_{n'' \in N, n' \leq n'' \leq n' + \Delta n} \sum_{j \in J_i} \sum_{i' \in I_j} w(i',n',n'') + \sum_{n'' \in N} \sum_{n' \in N, n' < n, n'' \leq n' \leq n'' + \Delta n} \sum_{j \in J_i} \sum_{i' \in I_j} w(i',n'',n') \quad \forall i \in I, n \in N, n > 1, \Delta n > 0 \tag{4}$$

$$\sum_{n' \in N, n' \leq n \leq n' + \Delta n} w(i,n',n) \leq \sum_{n' \in N, n' \leq n} \sum_{n'' \in N, n' \leq n'' \leq n' + \Delta n} w(i,n',n'') - \sum_{n'' \in N} \sum_{n' \in N, n' < n, n'' \leq n' \leq n'' + \Delta n} w(i,n'',n') \quad \forall i \in I, n \in N, \Delta n > 0 \tag{5}$$

Constraint (1) enforces that at the most one task can start at each event and constraint (2) states that at the most one task can end at each event. Constraint (3) states that if a task starts at an event, a different task cannot end at the same event, because only the task that starts at an event can end at the same event. Constraint (4) states that a new task can start only if the total number of tasks that started earlier matches the total number of tasks ending. Constraint (5) states that a task cannot end unless it started earlier. Note that constraints (2)–(5) are applicable only if $\Delta n > 0$.

Capacity Constraints

$$B_i^{\min} w(i,n,n') \le b(i,n,n') \le B_i^{\max} w(i,n,n') \quad \forall i \in I, n, n' \in N, n \le n' \le n + \Delta n \tag{6}$$

For each task, constraint (6) enforces the minimum and maximum batch size.

Material Balances

$$ST(s,n) = ST(s,n-1) + \sum_{i \in I_s^p} \sum_{n' \in N, n' \le n-1 \le n' + \Delta n} b(i,n',n-1) + \sum_{i \in I_s^c} \sum_{n' \in N, n \le n' \le n + \Delta n} b(i,n,n') \quad \forall s \in S, n \in N, n > 1 \tag{7}$$

$$ST(s,n) = ST_0(s) + \sum_{i \in I_s^c} \sum_{n' \in N, n \le n' \le n + \Delta n} b(i,n,n') \quad \forall s \in S, n = 1 \tag{8}$$

In the material balance (7), the amount of a state at the previous event is adjusted by the amount of the state produced by the tasks that are ending at the previous event and by the amount of the state being consumed by the tasks that are starting at the current event. In (8), the initial available amount of the state is taken into account.

Duration Constraints

$$T^f(i,n) = T^s(i,n) + \alpha_i w(i,n,n) + \beta_i b(i,n,n) \quad \forall i \in I, n \in N, \Delta n = 0 \tag{9}$$

$$T^f(i,n) \ge T^s(i,n) \quad \forall i \in I, n \in N, \Delta n > 0 \tag{10}$$

$$T^f(i,n') \ge T^s(i,n) + \alpha_i w(i,n,n') + \beta_i b(i,n,n') - M(1 - w(i,n,n')) \quad \forall i \in I, n, n' \in N, n \le n' \le n + \Delta n, \Delta n > 0 \tag{11}$$

$$T^f(i,n') \le T^s(i,n) + \alpha_i w(i,n,n') + \beta_i b(i,n,n') + M(1 - w(i,n,n')) \quad \forall i \in I, n, n' \in N, n \le n' \le n + \Delta n, \Delta n > 0 \tag{12}$$

If $\Delta n = 0$, then the finish time of a task that started at the same event is calculated from (9). Otherwise, if $\Delta n > 0$, then the finish time of a task that started at an earlier event is calculated from (10)–(12). Note that, because of the usage of three-index binary and continuous variables, the duration constraints in the S&F model are simpler and have fewer big-M terms compared to the duration constraints in the model of Janak et al. [5,6], and hence it is observed to result in improved LP relaxations. Janak et al. [5,6] used several additional tightening constraints and bilinear variables to improve the LP relaxation, which are not necessary in the S&F formulation.

Sequencing Constraints

Same Task in the Same Unit

$$T^s(i,n+1) \ge T^f(i,n) \quad \forall i \in I, n \in N, n < N \tag{13}$$

$$T^s(i,n+1) \le T^f(i,n) + M\left[1 - \left(\sum_{n' \in N, n' \le n} \sum_{n'' \in N, n' \le n'' \le n' + \Delta n} w(i,n',n'') - \sum_{n'' \in N} \sum_{n' \in N, n' < n, n'' \le n' \le n'' + \Delta n} w(i,n'',n')\right)\right] + M \sum_{n' \in N, n' \le n \le n' + \Delta n} w(i,n',n) \quad \forall i \in I, n \in N, n < N, \Delta n > 0 \tag{14}$$

If $\Delta n = 0$, then the constraint for same task in the same unit is given in (13). Otherwise, if $\Delta n > 0$, then the constraint for same task in the same unit is given by (13) and (14), where the zero wait condition of (14) is additionally applied when the task is active at event n but not ending at event n.

Different Tasks in the Same Unit

$$T^s(i,n+1) \ge T^f(i',n) \quad \forall i, i' \in I_j, i \ne i', j \in J, n \in N, n < N \tag{15}$$

Different Tasks in Different Units

$$T^s(i,n+1) \geq T^f(i',n) - M\left(1 - \sum_{n'\in N, n'\leq n\leq n'+\Delta n} w(i',n',n)\right) \quad \forall s\in S, i\in I_s^c, i'\in I_s^p, i\in I_j, i'\in I_{j'}, i\neq i', j,j'\in J, j\neq j', n\in N, n<N \tag{16}$$

For different tasks that produce or consume the same state, the start time of the consuming task at the next event is enforced to be later than the finish time of the producing task at the current event, provided the producing task is finishing at the current event.

Tightening Constraint

$$\sum_{i\in I_j}\sum_{n\in N}\sum_{n'\in N, n\leq n'\leq n+\Delta n}\left(\alpha_i w(i,n,n') + \beta_i b(i,n,n')\right) \leq H \quad \forall j\in J \tag{17}$$

The sum of the durations of all tasks suitable in each unit should be within the time horizon.

Utility Related Constraints

$$\sum_{i\in I_u}\left[\gamma_{iu}\left(\sum_{n'\in N, n'\leq n}\sum_{n''\in N, n'\leq n''\leq n'+\Delta n} w(i,n',n'') - \sum_{n''\in N}\sum_{n'\in N, n'<n, n''\leq n'\leq n''+\Delta n} w(i,n'',n')\right)\right] + \sum_{i\in I_u}\left[\delta_{iu}\left(\sum_{n'\in N, n'\leq n}\sum_{n''\in N, n'\leq n''\leq n'+\Delta n} b(i,n',n'') - \sum_{n''\in N}\sum_{n'\in N, n'<n, n''\leq n'\leq n''+\Delta n} b(i,n'',n')\right)\right] \leq U_u^{\max} \quad \forall u\in U, n\in N \tag{18}$$

The consumption of each utility at an event by all suitable active tasks is limited to the maximum availability in (18).

Sequencing of Utility Related Tasks

$$T_{ut}^s(u,n+1) \geq T_{ut}^s(u,n) \quad \forall u\in U, n\in N, n<N \tag{19}$$

Constraint (19) is similar to the constraint for same task in the same unit for each utility.

$$T_{ut}^s(u,n) \geq T^s(i,n) - M\left[1 - \left(\sum_{n'\in N, n'\leq n}\sum_{n''\in N, n'\leq n''\leq n'+\Delta n} w(i,n',n'') - \sum_{n''\in N}\sum_{n'\in N, n'<n, n''\leq n'\leq n''+\Delta n} w(i,n'',n')\right)\right] \quad \forall u\in U, i\in I_u, n\in N \tag{20}$$

$$T_{ut}^s(u,n) \leq T^s(i,n) + M\left[1 - \left(\sum_{n'\in N, n'\leq n}\sum_{n''\in N, n'\leq n''\leq n'+\Delta n} w(i,n',n'') - \sum_{n''\in N}\sum_{n'\in N, n'<n, n''\leq n'\leq n''+\Delta n} w(i,n'',n')\right)\right] \quad \forall u\in U, i\in I_u, n\in N \tag{21}$$

In constraints (20) and (21), at each event, the start times of all suitable tasks that consume a utility are enforced to be equal, and are assigned to $T_{ut}^s(u,n)$, if the task is active.

$$T^f(i,n-1) \geq T_{ut}^s(u,n) - M\left[1 - \left(\sum_{n'\in N, n'\leq n-1}\sum_{n''\in N, n'\leq n''\leq n'+\Delta n} w(i,n',n'') - \sum_{n''\in N}\sum_{n'\in N, n'<n-1, n''\leq n'\leq n''+\Delta n} w(i,n'',n')\right)\right] - M\sum_{n'\in N, n'\leq n-1\leq n'+\Delta n} w(i,n',n-1) \quad \forall u\in U, i\in I_u, n\in N, n>1 \tag{22}$$

$$T^f(i,n-1) \le T^s_{\mathrm{ut}}(u,n) + M\left[1-\left(\sum_{n'\in N,n'\le n-1}\sum_{n''\in N,n'\le n''\le n'+\Delta n} w(i,n',n'') - \sum_{n''\in N}\sum_{n'\in N,n'<n-1,n''\le n'\le n''+\Delta n} w(i,n'',n')\right)\right] \quad \forall u\in U, i\in I_u, n\in N, n>1 \tag{23}$$

In constraints (22) and (23), the end times of all suitable tasks that consume a utility at previous event are enforced to be before $T^s_{\mathrm{ut}}(u,n)$, if the task is active. If any of these tasks are finishing at the previous event, then the end times are enforced to be equal to $T^s_{\mathrm{ut}}(u,n)$. Note that, unlike in the model of Janak et al. [5,6], the end time of utility consumption $T^f_{\mathrm{ut}}(u,n)$ is not part of the model here, but is accurately calculated as a parameter after solving the model.

Bounds on Variables

$$\begin{aligned} &T^s(i,n)\le H;\ T^f(i,n)\le H;\ T^s_{\mathrm{ut}}(u,n)\le H;\\ &b(i,n,n')\le B_i^{\max};\ ST(s,n)\le ST_s^{\max};\\ &ST_0(s)\le ST_s^0 \end{aligned} \tag{24}$$

$$w(i,n,n')=0, b(i,n,n')=0 \quad \forall n'<n \tag{25}$$

$$T^s_{\mathrm{ut}}(u,n)=0, \quad \forall n=1 \tag{26}$$

$$ST_0(s)=0, \forall s\notin S^R \tag{27}$$

In (24), general bounds are added to different continuous variables. The non-permissible cases of the three-index binary and continuous variables are eliminated in (25). In (26), $T^s_{\mathrm{ut}}(u,n)$ at the first event should be assigned to the reference start time of the horizon, which is assigned to zero for simplicity. In (27), the initial amounts of all states, except the raw materials, should be assigned to their appropriate values, which are assigned to zero for simplicity. Depending on the STN of actual process considered, additionally it is possible to identify tasks that cannot occur at certain events and the corresponding binary and continuous variables can be eliminated.

Objective Function

Maximization of Profit

$$Max\ Profit = \sum_{s\in S^f}\sum_{n=N}\left(ST(s,n)+\sum_{i\in I_s^p}\sum_{n'\in N,n'\le n\le n'+\Delta n} b(i,n',n)\right) \tag{28}$$

For the objective of maximization of profit, the total amount of the final products produced by the last event is considered in (28). In all the constraints involving big-M terms, the value of M can be assigned to the time horizon, H.

Minimization of Makespan (MS)

$$Min\ MS \tag{29}$$

$$ST(s,N)+\sum_{n=N}\sum_{i\in I_s^p}\sum_{n'\in N,n'\le n\le n'+\Delta n} b(i,n',n)\ge D_s \quad \forall s\in S^f \tag{30}$$

$$T^f(i,N)\le MS \quad \forall i\in I \tag{31}$$

For the objective of minimization of makespan, MS, the demand constraints for the final products are given in (30). The makespan should be the upper bound on the end time of each task at the last event. The parameter H, in the tightening constraint of (17) needs to be replaced by MS.

Computational Case Studies

Example 1 Consider the same example discussed earlier. There are no resource considerations, hence, constraints. 18–33 will disappear. For the objective of maximization of profit, Shaik et al. [8] presented the comparative study for two different time horizons ($H=8$ h and $H=12$ h) for this example. In this study, we will consider two more instances of this example ($H=10$ h and $H=16$ h) and evaluate the performance of the unified S&F model. The model statistics for the objective of maximization of profit are given in Table 3.

Similar to the CBMN model, for each instance, at each event, the S&F model is solved using increasing values of $\Delta n = 0, 1, \ldots$

Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Table 3
Model statistics and computational results for example 1 under maximization of profit

Model	Events	CPU Time (s)	Nodes	RMILP ($)	MILP ($)	Binary variables	Continuous variables	Constraints	Nonzeros
Example 1a ($H = 8$)									
S&K	5	0.07	4	1730.9	1498.6	48	235	249	859
M&G	5	0.16	26	1730.9	1498.6	64	360	826	2457
CBMN($\Delta t = 1$)	5	0.01	4	1730.9	1498.6	32	104	114	439
I&F	4	0.03	13	1812.1	1498.6	18	90	165	485
	5	0.28	883	2305.3	1498.6	26	115	216	672
S&F($\Delta n = 0$)	4	0.01	9	1730.9	1498.6	18	122	193	511
($\Delta n = 0$)	5	0.22	530	2123.3	1498.6	26	155	252	696
Example 1b ($H = 10$)									
S&K	8	105.5	88679	2690.6	1962.7	84	433	456	1615
M&G	8	507.64	184605	2690.6	1962.7	112	609	1402	4884
CBMN($\Delta t = 1$)	8	1.82	6449	2690.6	1860.7[a]	56	170	189	760
($\Delta t = 2$)	8	81.95	194968	3136.3	1959[a]	104	218	261	1238
($\Delta t = 3$)	8	207.43	366226	3136.3	1962.7	144	258	321	1635
I&F	6	2.16	6713	3078.4	1943.2[a]	34	140	267	859
	7	43.73	101415	3551.8	1943.2[a]	42	165	318	1046
S&F($\Delta n = 0$)	6	2.13	6335	2730.7	1943.2[a]	34	188	311	881
($\Delta n = 0$)	7	27.93	64076	2780.2	1943.2[a]	42	221	370	1066
($\Delta n = 1$)	6	14.40	18902	2730.7	1962.7	65	219	692	2206
Example 1c ($H = 12$)									
S&K	7	1.93	1234	3002.5	2610.1	72	367	387	1363
	8	29.63	16678	3167.8	2610.3	84	433	456	1615
	9	561.58	288574	3265.2	2646.8	96	499	525	1867
	10	10889.61	3438353	3315.8	2646.8	108	565	594	2119
	11	> 67000[b]	17270000	3343.4	2646.8[a]	120	631	663	2371
M&G	7	2.15	814	3002.5	2610.1	96	526	1210	4019
	8	58.31	17679	3167.8	2610.3	112	609	1402	4884
	9	2317.38	611206	3265.2	2646.8	128	692	1594	5805
	10	> 67000[c]	10737753	3315.8	2646.8[a]	144	775	1786	6782
	11	> 67000[d]	9060850	3343.4	2658.5	160	858	1978	7815
CBMN($\Delta t = 2$)	7	0.63	1039	3045.0	2610.1	88	188	224	1050
	8	14.39	32463	3391.0	2610.3	104	218	261	1238
	9	331.72	593182	3730.5	2646.8	120	248	298	1426
	10	4366.09	6018234	4070.0	2646.8	136	278	335	1614
	11	> 67000[f]	80602289	4409.5	2646.8[a]	152	308	372	1802
I&F	7	6.19	14962	3788.3	2658.5	42	165	318	1046
	8	105.64	211617	4297.9	2658.5	50	190	369	1233
S&F($\Delta n = 0$)	7	5.06	10960	3301.0	2658.5	42	221	370	1066
($\Delta n = 0$)	8	96.13	171071	3350.5	2658.5	50	254	429	1251

Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Table 3 (continued)

Model	Events	CPU Time (s)	Nodes	RMILP ($)	MILP ($)	Binary variables	Continuous variables	Constraints	Nonzeros
Example 1d ($H = 16$)									
S&K	10	809.58	231810	4318.8	3738.4	108	565	594	2119
	11	38788.48	10065424	4404.8	3738.4	120	631	663	2371
M&G	10	8866.64	1402457	4318.8	3738.4	144	775	1786	6782
	11	> 67000[g]	9225164	4404.8	3738.4	160	858	1978	7815
CBMN($\Delta t = 1$)	10	15.04	33503	4318.8	3658.1[a]	72	214	239	974
($\Delta t = 2$)	10	206.30	315632	4579.4	3738.4	136	278	335	1614
($\Delta t = 2$)	11	12392.47	15779526	4918.9	3738.4	152	308	372	1802
I&F	8	16.35	32106	4435.0	3738.4	50	190	369	1233
	9	586.47	1057072	5054.8	3738.4	58	215	420	1420
S&F($\Delta n = 0$)	8	13.68	25814	4291.7	3738.4	50	254	429	1251
($\Delta n = 0$)	9	340.24	487795	4439	3738.4	58	287	488	1436

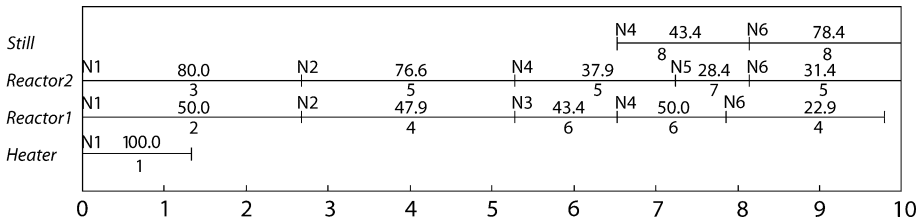
[a] Suboptimal solution; Relative Gap: 1.59%[b], 3.16%[c], 5.12%[d], 28.16%[e], 2.58%[f], 5.46%[g], 7.72%[h]

For the first instance, (example 1a), for $H = 8$ h, it can be observed that the unified model (S&F) performs equally well compared to the I&F model. The S&F model requires less number of nodes and gives better RMIP values compared to the I&F model. The S&F model requires the same number of binary variables as that of I&F model. For the second instance of this example, (example 1b), for $H = 10$ h, as already discussed in the motivation section, the I&F model fails to find the global optimal solution even at higher events. This can be confirmed by solving the S&F model for $\Delta n = 0$, which also gives the same suboptimal solution, because when $\Delta n = 0$ (similar to $\Delta t = 1$ for the CBMN model), the tasks are not allowed to take place over multiple events. For $\Delta n = 0$, the S&F model gives better RMIP values compared to I&F model. For $\Delta n = 1$, using six events the S&F model is able to find the global optimal solution in 14.4 CPU s. The CBMN model requires a value of $\Delta t = 3$ to find the global optimal solution. Compared to the JLF model from Table 2, there is almost 50% reduction in the RMIP value, in the number of constraints, and nonzeros for the S&F model. The number of binary and continuous variables are also fewer, apart from the exceptional computational performance of the S&F model. The schedule obtained by the S&F model is similar to the JLF model as shown in Fig. 3.

For the third instance (example 1c), for $H = 12$ h, it can be observed that the unified model (S&F) at $\Delta n = 0$, performs slightly faster compared to the I&F model. The S&F model requires less number of nodes and gives better RMIP values compared to the I&F model. Among the slot-based/global event-based models that require at least 11 events, only the M&G model is able to find the global optimal solution in the specified CPU time. The unit-specific event-based models require only 7 events to find the global optimal solution with exceptional computational performance. Similar conclusions hold true for the fourth instance (example 1d), for $H = 16$ h. The unified model (S&F) at $\Delta n = 0$, performs faster compared to the I&F model. The S&F model requires less number of nodes and gives better RMIP values compared to the I&F model. The slot-based/global event-based models require 10 events, while the unit-specific event-based models require only 8 events to find the global optimal solution with faster computational performance. When we consider an additional event, the S&F model again outperforms all the other models as seen in Table 3.

For the objective of minimization of makespan the computational results are given in Table 4. For models involving big-M constraints a value of $M = 50$ h is used, similar to Shaik et al. [8]. The S&F model gives better RMIP values compared to the I&F model.

Among the slot-based/global event-based models that require 10 events, the CBMN model takes a total CPU time of 106.6 s. As discussed in Shaik et al. [8] since the CBMN model gives a suboptimal solution at



Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Figure 3
Gantt chart for example 1b using the S&F model

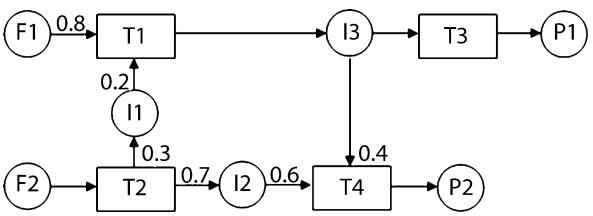
Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Table 4
Model statistics and computational results for example 1 under minimization of makespan

Model	Events	H	CPU Time (s)	Nodes	RMILP (h)	MILP (h)	Binary variables	Continuous variables	Constraints	Nonzeros
Example 1 ($D_8 = D_9 = 200$ mu)										
S&K	9	–	10.98	5378	18.685	19.789	96	556	528	1936
	10	–	519.35	142108	18.685	19.340	108	622	597	2188
M&G	9	50	66.55	15674	18.685	19.789	128	693	1598	5869
	10	50	5693.53	1066939	18.685	19.340	144	776	1790	6850
CBMN($\Delta t = 1$)	9	–	0.71	1809	18.685	19.789	64	193	216	872
($\Delta t = 1$)	10	–	50.49	134189	18.685	19.789[a]	72	215	241	979
($\Delta t = 2$)	10	–	56.11	109917	15.654	19.340	136	279	337	1623
I&F	8	50	0.78	1008	12.738	19.764	45	190	367	1211
	9	50	74.26	111907	12.477	19.340	53	215	418	1398
S&F($\Delta n = 0$)	8	50	1.42	2280	18.685	19.764	45	254	435	1244
($\Delta n = 0$)	9	50	98.60	105673	18.685	19.340	53	287	494	1429

[a] Suboptimal solution

$\Delta t = 1$, we consider the total time for both $\Delta t = 1$ and $\Delta t = 2$. Similarly, for the S&F model as well we need to add the total CPU time while comparing with other models. The unit-specific event-based models require 9 events to find the global optimal solution. Here, the I&F model solves faster compared to the unified S&F model. For problems involving no resource considerations, the S&F model was found [9] to perform either equally well or better than the I&F model.

Example 2 Now, consider an example with resource considerations and mixed storage policies. The STN for this example is shown in Fig. 4, and the corresponding data [5,7,8] is given in Table 5 and 6.



Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Figure 4
STN for example 2

Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Table 5
State related data for example 2

	F1	F2	I1	I2	I3	P1	P2
ST_s^{max}(kg)	1000	1000	200	100	500	1000	1000
ST_s^0(kg)	400	400	0	0	0	0	0
price$_s$($/kg)	0	0	0	0	0	30	40

Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Table 6
Task related data for example 2[a]

	B^{min}	B^{max}	T1		T2		T3		T4		T1		T2		T3		T4	
			α	β	α	β	α	β	α	β	γ_{iHS}	δ_{iHS}	γ_{iCW}	δ_{iCW}	γ_{iHS}	δ_{iHS}	γ_{iCW}	δ_{iCW}
R1	40	80	0.5	0.025	0.75	0.0375					6	0.25	4	0.3				
R2	25	50	0.5	0.4	0.75	0.06					4	0.25	3	0.3				
R3	40	80					0.25	0.0125	0.5	0.025					8	0.4	4	0.5

[a] B^{min}/B^{max} in kg, α in h, β in h/kg, γ in kg/min, and δ in kg/min per kg of batch

There are two types of reactors available for the process (types I and II), with two reactors of type I (R1 and R2) and one reactor of type II (R3) with four reactions suitable in them. Reactions T1 and T2 require a type I reactor, whereas reactions T3 and T4 require a type II reactor. Additionally, reactions T1 and T3 are endothermic, where the required heat is provided by steam (HS) available in limited amounts. Reactions T2 and T4 are exothermic, and the required cooling water (CW) is also available in limited amounts. Each reactor allows variable batch sizes, where the minimum batch size is half the capacity of the reactor. The processing times and the utility requirements include a fixed time and a variable term that is proportional to the batch size. The processing times are set so that the minimum batch size is processed in 60% of the time needed for the maximum batch size. For the raw materials and final products, unlimited storage is available, while for the intermediates, finite storage is available. Two different cases of this example studied in the literature [5,7,8] are considered that differ in the resource availability. In the first case (example 2a), we assume that the availability of both HS and CW is 40 kg/min, and in the second case (example 2b), it is 30 kg/min. Also, two different objective functions, maximization of profit and minimization of makespan, are considered. A comparative study of different continuous-time models for this example was already provided in Janak et al. [5,6] (JLF) and Shaik et al. [8], where additional bounding constraints were added to improve the computational performance of the JLF model. In this study, in order to investigate the effect of using the three-index binary and continuous variables, we compare the performance of the unified S&F model with the JLF model without the addition of any additional bounding constraints.

Maximization of Profit. For the objective of maximization of profit and a time horizon of 8 h, the optimal solution is \$5904.0 in the first case (example 2a) and \$5227.778 in the second case (example 2b). The computational results in terms of the model statistics and the CPU times are reported in Table 7 for the models of JLF and S&F.

Minimization of Makespan For the objective of minimization of makespan, the optimal solution is 8.5 h in the first case (example 2a) and 9.025 h in the second case (example 2b). The computational results in terms of the model statistics and the CPU times are

Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Table 7
Model statistics and computational results for example 2 under maximization of profit

Model	Events	CPU Time (s)	Nodes	RMILP (\$)	MILP (\$)	Binary variables	Continuous variables	Constraints	Nonzeros
Example 2a									
JLF	6	6.31	2838	11927.3	5904.0	48	273	1254	4483
SF($\Delta n = 1$)	6	2.03	2140	10713.8	5904.0	42	169	679	2135
Example 2b									
JLF	5	0.31	52	7147.1	5227.8	36	220	984	3198
SF($\Delta n = 1$)	5	0.07	49	6240	5227.8	30	136	540	1607

Short-Term Scheduling, Resource Constrained: Unified Modeling Frameworks, Table 8
Model statistics and computational results for example 2 under minimization of makespan

Model	Events	CPU Time (s)	Nodes	RMILP (h)	MILP (h)	Binary variables	Continuous variables	Constraints	Nonzeros
Example 2a									
JLF	7	10.95	2365	2.44	8.5	60	326	1545	5999
SF($\Delta n = 1$)	7	6.03	4612	5.08	8.5	54	202	825	2677
Example 2b									
JLF	6	1.63	166	3.00	9.025	48	273	1263	4519
SF($\Delta n = 0$)	6	0.22	103	5.08	9.025	24	151	404	1072

reported in Table 8. For constraints involving big-M terms, a common value of $M = 10$ is used.

For both the objective functions, the unified S&F model has better RMIP values and faster computational performance compared to the JLF model. Also there is a drastic reduction in the problem statistics especially the number of continuous variables, constraints and nonzeros in all the instances considered.

Conclusions

When there are resource considerations such as utility requirements, the unit-specific event-based models need to consider the formulations such as Janak et al. [5,6], that allow tasks to take place over multiple events. In this study, it is demonstrated that for short-term scheduling problem of batch plants involving no resource considerations as well, we need to allow tasks to take place over multiple events in order to ensure achieving the global optimal solution. In such cases, the model of Ierapetritou and Floudas [4] yields suboptimal solutions, while the model of Janak et al. [5,6] which was originally developed for solving problems with resource constraints, does not reduce well to the case of no resources in terms of problem statistics. Hence, a unified modeling approach is discussed based on the unit-specific event-based continuous-time formulation of Shaik and Floudas [9] where they consider three-index binary and continuous variables. Their model is applicable to both problems of with and without resources in a unified way. The Shaik and Floudas [9] model is found to perform either equally well or better than the I&F model for problems involving no resource considerations, and is found to perform better than the JLF model for problems with resource constraints.

Nomenclature

Indices

i, i'	tasks
j, j'	units
n, n', n''	events
s	states
u	utilities

Sets

I	tasks
I_j	tasks which can be performed in unit j
I_s	tasks which process state s and either produce or consume
I_s^p	tasks which produce state s
I_s^c	tasks which consume state s
I_u	tasks which consume utility u
J	units
J_i	units which are suitable for performing task i
N	event points within the time horizon
S	states
S^R	states that are raw materials
S^{IN}	states that are intermediates
S^P	states that are final products
U	utilities

Parameters

$B_i^{\min}$	minimum capacity (batch size) of task i
$B_i^{\max}$	maximum capacity (batch size) of task i
ST_s^o	initial amount of state s available
$ST_s^{\max}$	maximum amount of state s
α_i	coefficient of constant term of processing time of task i

β_i coefficient of variable term of processing time of task i

γ_{iu} coefficient of constant term of consumption of utility u by task i

δ_{iu} coefficient of variable term of consumption of utility u by task i

ρ_{is} proportion of state s produced ($\rho_{is} \geq 0$), consumed ($\rho_{is} \leq 0$) by task i

H time horizon, h

$price_s$ price of state s

D_s demand for state s,

Δn limit on the maximum number of events over which a task is allowed to continue

$U_u^{\max}$ maximum availability of utility u

M large positive number in big-M constraints

Binary Variables

$w(i, n, n')$ binary variable for assignment of task i that starts at event n and ends at event n'

Positive Variables

$b(i, n, n')$ amount of material undertaking task i that starts at event n and ends at event n'

$ST_0(s)$ initial amount of state $s \in S^R$ that is required from external resources

$ST(s, n)$ excess amount of state s that needs to be stored at event n

$T^s(i, n)$ time at which task i starts at event n

$T^f(i, n)$ time at which task i ends at event n

$T^s_{\mathbf{ut}}(u, n)$ start time at which there is a change in the consumption of utility u at event n

References

1. Castro PM, Barbosa-Povoa AP, Matos HA, Novais AQ (2004) Simple continuous-time formulation for short-term scheduling of batch and continuous processes. Ind Eng Chem Res 43:105–118
2. Floudas CA, Lin X (2004) Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. Comput Chem Eng 28:2109–2129
3. Floudas CA, Lin X (2005) Mixed integer linear programming in process scheduling: modeling, algorithms, and applications. Ann Oper Res 139:131–162
4. Ierapetritou MG, Floudas CA (1998) Effective continuous-time formulation for short-term scheduling: 1. Multipurpose batch processes. Ind Eng Chem Res 37:4341–4359
5. Janak SL, Lin X, Floudas CA (2004) Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: resource constraints and mixed storage policies. Ind Eng Chem Res 43:2516–2533
6. Janak SL, Lin X, Floudas CA (2005) Comments on "Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: resource constraints and mixed storage policies". Ind Eng Chem Res 44:426
7. Maravelias CT, Grossmann IE (2003) New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. Ind Eng Chem Res 42:3056–3074
8. Shaik MA, Janak SL, Floudas CA (2006) Continuous-time models for short-term scheduling of multipurpose batch plants: A comparative study. Ind Eng Chem Res 45:6190–6209
9. Shaik MA, Floudas CA (2007) Novel unified modeling approach for short-term scheduling. submitted for publication
10. Sundaramoorthy A, Karimi IA (2005) A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. Chem Eng Sci 60:2679–2702

Short-Term Scheduling Under Uncertainty: Sensitivity Analysis

Zhenya Jia, Marianthi Ierapetritou
Department of Chemical and Biochemical Engineering, Rutgers University, Piscataway, USA

Article Outline

Introduction
Methods
- Deterministic Scheduling Formulation
- MILP Sensitivity Analysis
- Robustness Metric
- Proposed Uncertainty Analysis Approach

Case
Conclusions
References

Indices

i = tasks
j = units
n = event points
s = states
k = scenarios

Parameters

price(s) = price of state s

$\rho^p(s, i)$, $\rho^c(s, i)$ = proportion of state s produced, consumed from task i, respectively

r(s) = market requirement for state s at the end of the time horizon

Vmin(i,j) = minimum capacity of unit j when processing task i

Vmax(i,j) = maximum capacity of unit j when processing task i

stmax(s) = maximum storage capacity of state s

Variables

wv(i,j,n) = binary variables that assign the beginning of task i in unit j at event point n

b(i,j,n) = amount of material undertaking task i in unit j at event point n

d(s,n) = amount of state s being delivered to the market at event point n

H = time horizon

st(s,n) = amount of state s at event point n

Ts(i,j,n) = starting time of task i in unit j at event point n

Tf(i,j,n) = finishing time of task i in unit j at event point n

Introduction

There has been a significant amount of work devoted to the area of short-term scheduling, which involves the determination of the order in which tasks use units and various resources and the detailed timing of the execution of all tasks so as to achieve the desired performance. The problem data are usually assumed to be deterministic in the studies. However, in real plants, parameters like raw material availability, processing times, and market requirements vary with respect to time and are often subject to unexpected deviations. Therefore, the consideration of uncertainty in the scheduling problem becomes of great importance in order to preserve plant feasibility and viability during operations.

Although there are a large number of papers that address uncertainty in process design, much less attention has been devoted to the issue of uncertainty in process planning and scheduling, mainly owing to the increased complexity of the deterministic problem. Among the work that has appeared in the literature is that of Shah and Pantelides [18] that addressed the problem of the design of multipurpose batch plants considering different schedules for different sets of production requirements using a scenario-based approach [6] and an approximate solution strategy. Pistikopoulos and Ierapetritou [14] presented a two-stage stochastic programming formulation for the problem of batch plant design and operations under uncertainty. The multiperiod planning and scheduling of multiproduct plants under demand uncertainty was addressed by Petkov and Maranas [13]. The stochastic elements in their proposed model are expressed with equivalent deterministic forms, resulting in a convex mixed-integer nonlinear programming (MINLP) problem. Schmidt and Grossmann [16] considered the optimal scheduling of new product testing tasks and reformulated the initial nonlinear, nonconvex disjunctive model as a mixed-integer linear programming (MILP) problem using different sets of simplifying assumptions that give rise to different models. The uncertainties in planning and scheduling problems are generally described through probabilistic models. During the last decade, fuzzy set theory has been applied to scheduling optimization using heuristic search techniques [8,10]. Recently, Balasubramanian and Grossmann [2] developed MILP models for flowshop scheduling and new product development processing scheduling, based on a fuzzy representation of uncertainty. Daniels and Carrillo [4] addressed the problem of β-robust scheduling in single-stage production facilities with uncertain processing times. Vin and Ierapetritou [20] proposed a multiperiod programming model to improve the schedule performance of batch plants under demand uncertainty. Acevedo and Pistikopoulos [1] addressed linear process engineering problems under uncertainty using a branch and bound algorithm, based on solution of multiparametric linear programs at each node of the tree, and the evaluation of the uncertain parameter space for which a node must be considered.

However, most of the existing approaches can handle only a certain type of uncertain parameters, mostly uncertainty in product demands, and more importantly, the additional complexity makes them infeasible for realistic applications. In this work, a novel framework is proposed for uncertainty analysis of scheduling problems based on the ideas of sensitivity analysis of the

corresponding MILP problem which is addressed using a branch and bound solution method.

Methods

Deterministic Scheduling Formulation

In this work, the mathematical model used for batch plant scheduling follows the main idea of the continuous time formulation proposed by Ierapetritou and Floudas [9]. The model involves the following constraints:

$$\text{Minimize } H \text{ or maximize } \sum_s \sum_n price(s)d(s,n), \quad (1)$$

$$\text{subject to } \sum_{i\in I_j} wv(i,j,n) \le 1, \quad (2)$$

$$\begin{aligned} st(s,n) =& st(s,n-1) - d(s,n), \\ &+ \sum_{i\in I_j} \rho^p(s,i) \sum_{j\in J_i} b(i,j,n-1) \\ &+ \sum_{i\in I_j} \rho^c(s,i) \sum_{j\in J_i} b(i,j,n), \end{aligned} \quad (3)$$

$$st(s,n) \le stmax(s), \quad (4)$$

$$\begin{aligned} &Vmin(i,j)wv(i,j,n) \\ &\quad \le b(i,j,n) \le Vmax(i,j)wv(i,j,n), \end{aligned} \quad (5)$$

$$\sum_n d(s,n) \ge r(s), \quad (6)$$

$$\begin{aligned} Tf(i,j,n) = Ts(i,j,n) &+ \alpha(i,j)wv(i,j,n) \\ &+ \beta(i,j)b(i,j,n), \end{aligned} \quad (7)$$

$$Ts(i,j,n+1) \ge Tf(i,j,n) - U(1-wv(i,j,n)), \quad (8)$$

$$Ts(i,j,n) \ge Tf(i',j,n) - U(1-wv(i',j,n)), \quad (9)$$

$$Ts(i,j,n) \ge Tf(i',j',n) - U(1-wv(i',j',n)), \quad (10)$$

$$Ts(i,j,n+1) \ge Ts(i,j,n), \quad (11)$$

$$Tf(i,j,n+1) \ge Tf(i,j,n), \quad (12)$$

$$Tf(i,j,n) \le H, \quad (13)$$

$$Ts(i,j,n) \le H, \quad (14)$$

where U denotes an upper bound of the makespan, for the cases where the objective is the minimization of the makespan. For the cases where maximization of profit is considered, $U = H$ in constraints (8)–(10). In general, the objective function is to minimize the makespan as shown in (1) or to maximize the total profit. Allocation constraint (2) states that only one of the tasks can be performed in each unit at an event point n. Constraint (3) represents the material balance for each material at each event point n being equal to that at event point $n-1$, adjusted by any amounts produced and consumed between event points $n-1$ and n, and delivered to the market at event point n. The storage and capacity limitations of the production units are expressed by constraints (4) and (5). Constraint (6) is written to satisfy the demands of the final products. Constraints (7)–(14) represent time limitations due to task duration and sequence requirements in the same or different production units.

Although there are a large number of papers that deal with uncertainty issues concerning process design and production planning, as reported in "Introduction," the issue of uncertainty is not well studied for scheduling problems, mainly owing to the high complexity of the deterministic case.

MILP Sensitivity Analysis

The formulation presented in "Deterministic Scheduling Formulation" corresponds to the MILP problem where the binary variables ($wv(i,j,n)$) denote the assignment of tasks i to units j at event point n, respectively, throughout the time horizon. Therefore, the effects of operation parameters on the plant performance can be investigated through the sensitivity analysis of the MILP model of the deterministic scheduling problem.

Although sensitivity analysis theory is well developed in linear programming, efforts are still being made in order to handle the integer programming case, mainly owing to lack of optimality criteria for the integer optimization problems. Schrage and Wolsey [17] examined the effect of a small perturbation on the right-hand side or objective function coefficients in an integer program by collecting dual information at each node of the branch and bound tree while solving the original integer program and using a recursive scheme to obtain an upper bound on the objective function. Their results were extended to nonlinear integer programming

problems by Skorin-Kapov and Granot [19]. Pertsinidis et al. [12] developed a sensitivity analysis algorithm for parametric MILP programs, which also provides the results of the MILP master problem for the MINLP case, while an algorithm that provides a sequence of improving parametric lower and upper bounds is employed for parametric nonlinear integer programming subproblems. An algebraic geometry algorithm for solving integer programming problems was presented by Bertsimas et al. [3] Their method provides a natural generalization of the Farkas lemma for integer programming and leads to a method of performing sensitivity analysis.

A method of sensitivity analysis for MILP was presented by Dawande and Hooker [5], based on the idea of inference duality. It reveals that any perturbation that satisfies a certain system of linear inequalities will reduce the optimal value no more than a prespecified amount. The inference-based sensitivity analysis consists of two parts: dual analysis that determines how much the problem can be perturbed while keeping the objective function value in a certain range, while primal analysis gives an upper bound on how much the objective function value will increase if the problem is perturbed by a certain amount. The dual solution is obtained by using inference methods to generate constraints at every node that is violated by the branching cuts. The dual solution can be viewed as a proof of optimality and can be utilized to determine under what parameter perturbations the dual solution still provides a valid proof. More specifically the main results of the inference-based sensitivity analysis are summarized below.

For the general mixed-integer problem

$$\begin{aligned} &\text{Minimize } z = cx \\ &\text{subject to } Ax \geq a, 0 \leq x \leq h, \\ &\qquad x_j \text{ integer}, j = 1, \ldots, k\,. \end{aligned} \tag{15}$$

Assuming a perturbation of all problem parameters such that

$$\begin{aligned} &\text{Minimize } z = (c + \Delta c)x \\ &\text{subject to } (A + \Delta A)x \geq a + \Delta a\,, 0 \leq x \leq h, \\ &\qquad x_j \text{ integer}, j = 1, \ldots, k\,. \end{aligned} \tag{16}$$

If there exist $s_1^p, \ldots, s_n^p$ that satisfy the following set of inequalities, the constraint $z \geq z^* - \Delta z$ remains valid: for the perturbations ΔA and Δa in the parameters involved in the left-hand side and the right-hand side of the constraints

$$\begin{aligned} &\lambda_i^p \sum_{j=1}^{n} A_{ij}\underline{u}_j^p + \sum_{j=1}^{n} s_j^p(\overline{u}_j^p - \underline{u}_j^p) - \lambda_i^p \Delta a_i \leq r_p\,, \\ &\qquad s_j^p \geq \lambda_i^p \Delta A_{ij}, \quad s_j^p \geq -q_j^p, \quad j = 1, \ldots, n\,, \\ &\qquad r_p = -\sum_{j=1}^{n} q_j^p \overline{u}_j^p + \lambda^p a - z_p + \Delta z_p\,; \end{aligned} \tag{17}$$

for a perturbation Δc of the coefficients of the objective function

$$\begin{aligned} &\sum_{j=1}^{n} \Delta c_j \underline{u}_j^p - s_j^p(\overline{u}_j^p - \underline{u}_j^p) \geq -r_p\,, \\ &s_j^p \geq -\Delta c_j, s_j^p \geq -q_j^p, \quad j = 1, \ldots, \end{aligned} \tag{18}$$

where $q_j^p = \lambda_i^p A_{ij} - \lambda_0^p c_j$; p corresponds to the leaf node where the dual variable of the objective function (λ_0^p) equals 1, whereas $\underline{u}_j^p$ and $\overline{u}_j^p$ denote the lower and the upper bound of x_j at node p, respectively; z_p is the objective value at node p; and $\Delta z_p = z^* - z_p$. Leaf nodes are the nodes at which the branch and bound procedure terminates based on standard fathoming criteria [7]. Thus, with use of constraints (17) and (18) in the scheduling problem, the range of parameters where the objective remains within certain limits can be identified and used to evaluate alternative schedules at the branch and bound tree. Moreover, the importance of different constraints and parameters is obtained and can be utilized to improve future plant operability.

Robustness Metric

In order to improve the schedule flexibility prior to its execution, it is important to measure the performance of a deterministic schedule under changing conditions due to uncertainty.

Standard deviation (SD) is one of the most commonly used metrics to evaluate the robustness of a schedule. To evaluate the SD, the deterministic model with a fixed sequence of tasks ($wv(i, j, n)$) is solved for different realizations of uncertain parameters that define the set of scenarios k which results in different makespans H_k. The SD is then defined as

$$SD_{\text{avg}} = \sqrt{\sum_k \frac{(H_k - H_{\text{avg}})^2}{(p_{\text{tot}} - 1)}}, H_{\text{avg}} = \frac{\sum_k H_k}{p_{\text{tot}}}\,, \tag{19}$$

where H_{avg} is the average makespan over all the scenarios, and (p_{tot}) denotes the total number of scenarios. A detailed discussion of different robustness metrics can be found in Samsatli et al. [15]. Vin and Ierapetritou [20] proposed a robustness metric taking into consideration the infeasible scenarios. In the case of infeasibility, the problem is solved to meet the maximum demand possible by incorporating slack variables in the demand constraints. Then the inventories of all raw materials and intermediates at the end of the schedule are used as the initial conditions in a new problem with the same schedule to satisfy the unmet demand. The makespan under infeasibility (H_{corr}) is determined as the sum of those two makespans. Their proposed robustness metric is defined as

$$SD_{corr} = \sqrt{\sum_{k} \frac{(H_{act} - H_{avg})^2}{(p_{tot} - 1)}}, \tag{20}$$

where $H_{act} = H_k$, if scenario k is feasible and $H_{act} = H_{corr}$, if scenario k is infeasible.

Proposed Uncertainty Analysis Approach

The basic idea of the proposed approach is to utilize the information obtained from the sensitivity analysis of the deterministic solution to determine (1) the importance of different parameters and constraints and (2) the range of parameters where the optimal solution remains unchanged. The main steps of the proposed approach are shown in Fig. 1. More specifically, there are two parts in the proposed analysis. In the first part, important information about the effect of different parameters is extracted following the sensitivity analysis step, whereas in the second part alternative schedules are determined and evaluated for different uncertainty ranges.

First, the deterministic scheduling is solved at the nominal values using a branch and bound solution approach, and the dual multipliers $\boldsymbol{\lambda}^p$ are collected at each leaf node p. Then the inference-based sensitivity analysis as described in the previous section is performed for all the important scheduling parameters, including demands, prices, processing times and capacities. Note that only the dual information of the nodes that correspond to nonzero dual variables is required. Using the results of this analysis, one can answer a number of very interesting questions regarding the robustness of the plant to parameter changes.

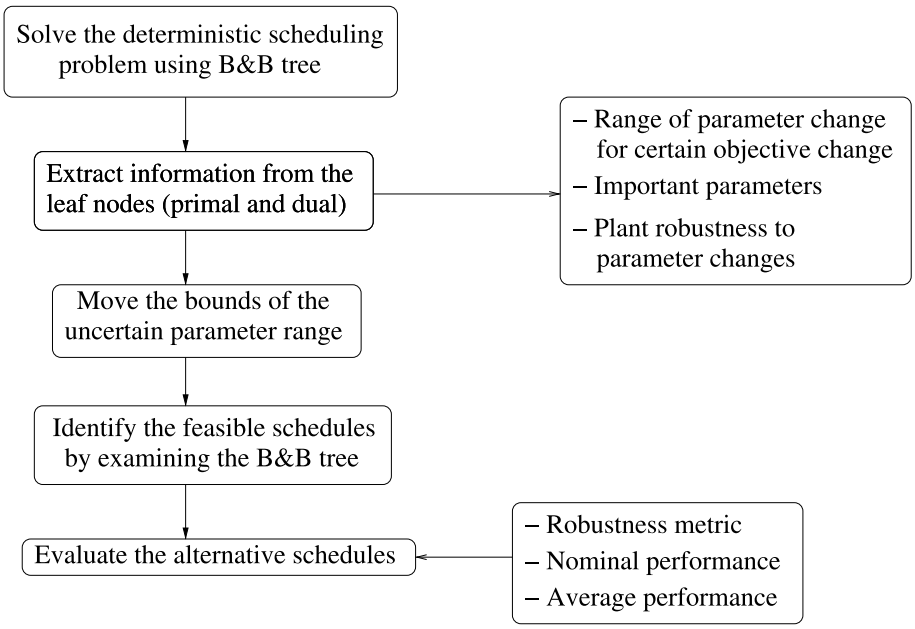
In particular:

- How does the capacity of the units affect the production objective?
- What is the range of product demand that can be covered and how much would the profit be affected by such changes?
- What is the effect of a price change on the objective value?
- What is the significance of the constraints involved in the model? Are there any redundant sets of constraints?

The first question can be answered by imposing the same perturbation on capacity constraint (5) for the different units involved in the production of specific products and determining the change in the objective function (Δz). The unit with the largest effect on the objective value is also the most critical one for the production of this product and thus a change in its capacity will result in the largest production change. Similarly, the rest of the questions can be answered by analyzing perturbations at the appropriate constraints together with the effects on the objective function. The results for two examples are given in the next section.

In the second part of the analysis, the sensitivity information is used to define the range of uncertain parameters where the schedule is optimal and to identify alternative schedules at different uncertainty ranges. The set of constraints (17) and (18) are used to determine the range of uncertain parameters for certain changes in the objective function. The branch and bound procedure is then continued on the nodes with the objective value within the predicted limits to identify new optimal solutions. The alternative schedules are evaluated using the robustness metric (SD_{corr}) as defined in "Robustness Metric," the average and the nominal schedule performance in terms of the objective function.

Since the entire analysis is based on a single branch and bound tree among a large number of possible branch and bound trees that can be used to solve the MILP, it provides conservative sensitivity ranges. Theoretically, the exact sensitivity ranges can be obtained by investigating an exponential number of branch and bound trees. However, using the above analysis, one can

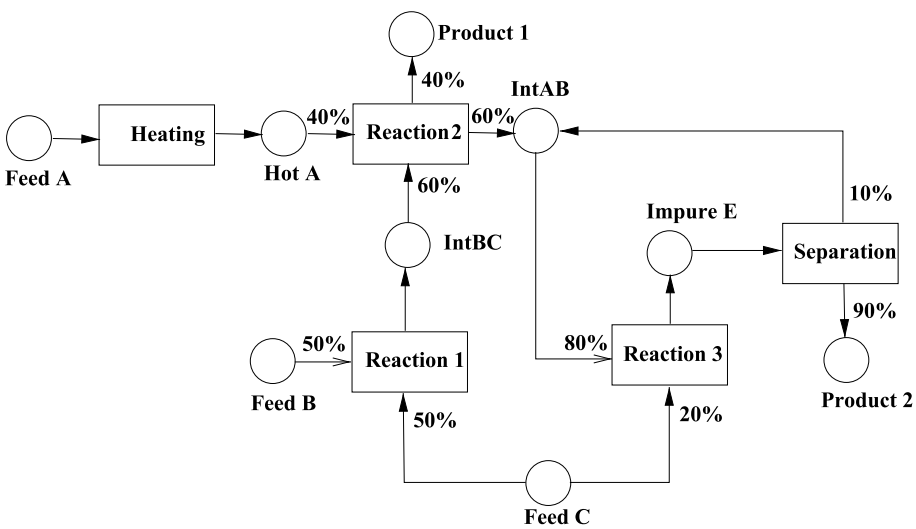


Short-Term Scheduling Under Uncertainty: Sensitivity Analysis, Figure 1
Flow chart of proposed approach. *B&B* branch and bound

extract useful information regarding the approximate range of the parameter change for a certain objective change and the robustness of the plant to parameter changes, and one can also determine the importance of different parameters as illustrated in the next section.

Case

The case study [9] considers two different products produced through five processing stages: heating, reactions 1, 2 and 3, and separation of product 2 from impure E as



Short-Term Scheduling Under Uncertainty: Sensitivity Analysis, Figure 2
State-task network representation for the case

illustrated in the state-task network (STN) representation in Fig. 2. For the first part of the analysis, the problem is solved with the objective of maximizing the profit within the time horizon of 12 h. After the sensitivity analysis has been performed, the following information is obtained. It is found that the most critical task of the production line is reaction 2. By decreasing the processing capacity of reaction 2 in reactor 1 or reactor 2 by 11 units, the profit will be reduced by 5%, whereas very small change or no change at all is observed in the objective function with a processing capacity change for reaction 1 or reaction 3 in both reactors. The objective value is also not sensitive to the change of other parameters, for example, the processing capacity of separation in the separator can drop by up to 120 units without the profit decreasing. Another important modeling issue that can be addressed is the question of constraint redundancy. Here the importance of storage constraints is investigated and it is found that these constraints are redundant since they are not active in any of the solution branch and bound nodes. More interestingly, the duration constraints are also found to be redundant, which means that the maximum processing capacities are already reached with the current processing times, so the profit cannot be improved even with zero processing times assuming a fixed number of event points. For the second part of the analysis, the demand of product 2 is considered to be the uncertain parameter varying within the range [20, 80] and the objective function is modified to minimize the makespan. A branch and bound tree is constructed at nominal point $r('p2') = 50$ and the dual information is stored at each node.

Applying the inference duality sensitivity analysis, one obtains the following expression regarding the range of demand change following a specific objective change (ΔH): $-0.0297\Delta d \leq \Delta H$, which means that if the demand is increased by Δd, the new makespan becomes at most $H_{\text{nom}} + 0.0297\Delta d$. When $r('p2')$ is increased from 50 to 80, schedule 1 becomes infeasible. Then, we solve the linear programming problem at each leaf node with the demand of 80 and check the leaf nodes with the objective value below 7.89 that is obtained using this inequality in the branch and bound tree. The new optimal solution is found to be schedule 2 and schedule 3 is one feasible solution, as illustrated in Table 2 . The schedules are then evaluated with respect to the mean and nominal makespan and the SD within the demand range [20, 80] and the val-

Short-Term Scheduling Under Uncertainty: Sensitivity Analysis, Table 1
Comparison of alternative schedules for the case

	Schedule 1	Schedule 2	Schedule 3
$H_{\text{nom}}(h)$	7.00	7.14	7.40
$H_{\text{avg}}(h)$	8.15	7.24	7.40
SD_{corr}	2.63	0.29	0.27

Short-Term Scheduling Under Uncertainty: Sensitivity Analysis, Table 2
Values of binary variables of optimal schedules

(Task, unit)	n_0	n_1	n_2	n_3
(Heating, heater)	1	0	0	0
(Reaction 1, reactor 1)	0	0	0	0
(Reaction 1, reactor 2)	1	1	0	0
(Reaction 2, reactor 1)	0	1	0	1
(Reaction 2, reactor 2)	0	0	0	0
(Reaction 3, reactor 1)	0	0	1	0
(Reaction 3, reactor 2)	0	0	1	0
(Separation, still)	0	0	0	1
(schedule 1)				

(Task, unit)	n_0	n_1	n_2	n_3
(Heating, heater)	1	1	0	0
(Reaction 1, reactor 1)	1	0	0	0
(Reaction 1, reactor 2)	1	0	0	0
(Reaction 2, reactor 1)	0	1	0	1
(Reaction 2, reactor 2)	0	1	0	0
(Reaction 3, reactor 1)	0	0	1	0
(Reaction 3, reactor 2)	0	0	1	0
(Separation, still)	0	0	0	1
(schedule 2)				

(Task, unit)	n_0	n_1	n_2	n_3
(Heating, heater)	1	0	0	0
(Reaction 1, reactor 1)	1	0	0	0
(Reaction 1, reactor 2)	1	0	0	0
(Reaction 2, reactor 1)	0	1	0	0
(Reaction 2, reactor 2)	0	1	0	0
(Reaction 3, reactor 1)	0	0	1	0
(Reaction 3, reactor 2)	0	0	1	0
(Separation, still)	0	0	0	1
(schedule 3)				

ues are shown in Table 1. Compared with schedule 2, schedule 3 has a larger mean makespan but a lower SD, which means higher robustness; therefore, depending on the decision-maker's attitude towards risk and the expected growth in demand, one can choose schedule 3 over schedule 2, whereas schedule 1 remains a valid alternative if the demand is expected to remain constant.

Note that the proposed uncertainty analysis does not substantially increase the problem complexity. That is due to the fact that the required information is already obtained from the solution of the deterministic problem.

Conclusions

An integrated framework was developed in the work reported here to handle uncertainty in short-term scheduling based on the idea of inference-based sensitivity analysis for the MILP problem and the utilization of a branch and bound solution method. The proposed method leads to the determination of the importance of different parameters and constraints on the objective function and the generation and evaluation of a set of alternative schedules given the variability of the uncertain parameters. The main advantage of the proposed method is that no substantial complexity is added compared with the solution of the deterministic case since the only additional information required is the dual information at the leaf nodes of the branch and bound tree. One illustrative example was presented to highlight the information extracted by the proposed approach and the complexity involved.

References

1. Acevedo J, Pistikopoulos EN (1997) A multiparametric programming approach for linear process engineering problems under uncertainty. Ind Eng Chem Res 36:717–728
2. Balasubramanian J, Grossmann IE (2003) Scheduling optimization under uncertainty – an alternative approach. Comput Chem Eng 27:469–490
3. Bertsimas D, Georgia P, Tayur S (2000) A new algebraic geometry algorithm for integer programming. Manag Sci 46:999–1008
4. Daniels RL, Carrillo JE (1997) β-robust scheduling for single-machine systems with uncertain processing times. IIE Trans 29:977–985
5. Dawande MW, Hooker JN (2000) Inference-based sensitivity analysis for mixed integer/linear programming. Oper Res 48:623–634
6. Dembo R (1991) Scenario Optimization. Ann Oper Res 30:63–80
7. Floudas CA (1995) Nonlinear and mixed-integer optimization. Oxford, New York
8. Fortemps P (1997) Jobshop scheduling with imprecise durations: a fuzzy approach. IEEE Trans Fuzzy Syst 5:557–569
9. Ierapetritou MG, Floudas CA (1998) Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. Ind Eng Chem Res 37:4341–4359
10. Ishibuchi H, Yamamoto N, Murata T, Tanaka H (1994) Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems. Fuzzy Sets Syst 67:81–100
11. Maravelias CT, Grossmann IE (2003) New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. Ind Eng Chem Res 42:3056–3074
12. Pertsinidis A, Grossmann IE, McRae GJ (1998) Parametric optimization of MILP programs and a framework for the parametric optimization of MINLPs. Comput Chem Eng 22:205–212
13. Petkov SB, Maranas CD (1997) Multiperiod planning and scheduling of multiproduct batch plants under demand uncertainty. Ind Eng Chem Res 36:4864–4881
14. Pistikopoulos EN, Ierapetritou MG (1995) A novel approach for optimal process design under uncertainty. Comput Chem Eng 19:1089–1110
15. Samsatli NJ, Papageorgiou LG, Shah N (1998) Robustness metrics for dynamic optimization models under parameter uncertainty. AICHE J 44:1993–2006
16. Schmidt C, Grossmann IE (1996) Optimization models for the scheduling of testing tasks in new product development. Ind Eng Chem Res 35:3498–3510
17. Schrage L, Wolsey L (1985) Sensitivity analysis for branch and bound integer programming. Oper Res 33:1008–1023
18. Shah N, Pantelides CC (1992) Design of multipurpose batch plants with uncertain production requirements. Ind Eng Chem Res 31:1325–1337
19. Skorin-Kapov J, Granot F (1987) Non-linear integer programming: sensitivity analysis for branch and bound. Oper Res Lett 6:269–274
20. Vin JP, Ierapetritou MG (2001) Robust short-term scheduling of multiproduct batch plants under demand uncertainty. Ind Eng Chem Res 40:4543–4554

Signal Processing with Higher Order Statistics

HOS

DIMITRIOS HATZINAKOS
University of Toronto, Toronto, Canada

MSC2000: 90C26, 90C90

Article Outline

Keywords
See also
References

Keywords

Higher-order statistics; nonGaussian signal processing; Nonlinear signal processing

Signal processing methodologies based on *higher-order statistics* or spectra (HOS) of order greater than two have become important signal processing tools in a variety of application areas: digital communications, system identification and spectral analysis, source separation and array processing, time delay estimation, image and speech processing and biomedical applications among others, [3,4,9]. The increased popularity of HOS in signal processing applications can be attributed to many attractive properties they possess: preservation of *nonminimum phase* information, ability to detect/identify nonlinear behavior, robustness to Gaussian and other forms of noise, etc. It is well known that when a signal is Gaussian there is no benefit in considering the HOS of this signal since all the statistical information is conveyed by its first and second order statistics (SOS). However, for nonGaussian signals the SOS do not provide a complete description and a lot of important information can be extracted from their HOS, [8,9,11].

The n-order moment and cumulant sequences of an n-order stationary random process $\{y(i)\}$, $i = 1, 2, \ldots$ are defined as, [9]:

$$M_{y,n}(\lambda_1, \ldots, \lambda_{n-1}) = \mathsf{E}\{y_1 \cdots y_n\},$$
$$C_{y,n}(\lambda_1, \ldots, \lambda_{n-1}) = \sum (-1)^{p-1}(p-1)!\,\mathsf{E}\left\{\prod_{i\in I_1} y_i\right\} \cdots \mathsf{E}\left\{\prod_{i\in I_p} y_i\right\}, \tag{1}$$

where, $y_1 = y(i)$, $y_2 = y(i+\lambda_1), \ldots, y_n = y(i+\lambda_{n-1})$, the summation covers all partitions $(I_1, \ldots, I_p)$, $p = 1, \ldots, n$, of the set $\{1, \ldots, n\}$, $\lambda_k = 0, \pm 1, \pm 2, \ldots$, and $E\{\cdot\}$ denotes statistical expectation. The second, third and fourth order cumulants of zero-mean processes are utilized often in practice and take the form

$$C_{y,2}(\lambda) = M_{y,2}(\lambda), \tag{2}$$

$$C_{y,3}(\lambda_1, \lambda_2) = M_{y,3}(\lambda_1, \lambda_2), \tag{3}$$

$$\begin{aligned} C_{y,4}(\lambda_1, \lambda_2, \lambda_3) = {} & M_{y,4}(\lambda_1, \lambda_2, \lambda_3) \\ & - M_{y,2}(\lambda_1) M_{y,2}(\lambda_2 - \lambda_3) \\ & - M_{y,2}(\lambda_2) M_{y,2}(\lambda_3 - \lambda_1) \\ & - M_{y,2}(\lambda_3) M_{y,2}(\lambda_1 - \lambda_2). \end{aligned} \tag{4}$$

The $\gamma_{y,2} = C_{y,2}(0)$ is the *variance*, the $\gamma_{y,3} = C_{y,3}(0,0)$ is the *skewness*, and the $\gamma_{y,4} = C_{y,4}(0,0,0)$ is the *kurtosis* of $\{y(i)\}$. For a complex process the definitions in (1) may include conjugation in one or more terms in the products.

The n-order *polyspectrum* (*higher-order spectrum*) of $\{y(i)\}$ is defined as the $(n-1)$-dimensional discrete Fourier transform of $C_{y,n}(\lambda_1, \ldots, \lambda_{n-1})$, that is

$$S_{y,n}(e^{j\omega_1}, \ldots, e^{j\omega_{n-1}}) = \sum_{\lambda_1, \ldots, \lambda_{n-1} = -\infty}^{\infty} C_{y,n}(\lambda_1, \ldots, \lambda_{n-1}) \prod_{l=1}^{n-1} e^{-j\omega_l \lambda_l}, \tag{5}$$

$|\omega_l| \le \pi$, $l = 1, \ldots, n-1$, $|\sum_{l=1}^{n-1} \omega_l| \le \pi$. For $n = 2, 3, 4$ we obtain the power spectrum, the bispectrum and the trispectrum, respectively.

Cumulants are utilized as measures of *'Gaussianity'* and statistical independence because they satisfy the following two important properties:

1) Given that the set of random variables $\{y(i), y(i+\lambda_1), \ldots, y(i+\lambda_{n-1})\}$ is divided into any number of mutually independent subsets, then, $C_{y,n}(\lambda_1, \ldots, \lambda_{n-1}) = 0$. Therefore, if a random process $\{y(i)\}$ is independent identically distributed, then

$$C_{y,n}(\lambda_1, \ldots, \lambda_{n-1}) = \gamma_{y,n} \delta(\lambda_1) \cdots \delta(\lambda_{n-1}),$$

where $\delta(\lambda) = 0$, $\lambda \neq 0$, is the delta function. Also, given that $\{x(i)\}$, $\{z(i)\}$, $i = 1, 2, \ldots$ are two independent processes and $y(i) = x(i) + z(i)$, then

$$C_{y,n}(\lambda_1, \ldots, \lambda_{n-1}) = C_{x,n}(\lambda_1, \ldots, \lambda_{n-1}) + C_{z,n}(\lambda_1, \ldots, \lambda_{n-1}). \tag{6}$$

2) If the set of random variables $\{y(i), y(i+\lambda_1), \ldots, y(i+\lambda_{n-1})\}$ are jointly Gaussian, then $C_{y,n}(\lambda_1, \ldots, \lambda_{n-1}) = 0$, for $n \geq 3$.

These properties do not hold for moments. For this reason cumulants are often more attractive than moments in applications of HOS, [7,9].

Often in practice we want to calculate the higher-order cumulants of the process

$$y(k) = s(k) + w(k),$$

where the $s(k)$ may be either deterministic signal or a random process and the $w(k)$ is a zero-mean stationary Gaussian noise process independent from $s(k)$. In practice, the estimation of HOS is based on time averaging. To overcome problems of nonstationarity of $y(k)$ in the case where $s(k)$ is a deterministic energy signal, it is necessary to assume multiple realizations $y_j(k)$, $j = 1, \ldots, J$, $k = 1, \ldots, K$, of sufficient length and estimate the n-order moment as follows:

$$\widehat{M}_{y,n}(\lambda_1, \ldots, \lambda_{n-1}) = G \sum_{j=1}^{J} \sum_{k=1}^{K} y_j(k) y_j(k+\lambda_1) \cdots y_j(k+\lambda_{n-1}) \tag{7}$$

for $\lambda_l = 0, \pm 1, \ldots, \pm L$, and $G = 1/J$, [9,10]. If $s(k)$ is random and locally stationary or a deterministic power signal, then (7) applies by segmenting the $y(k)$ into J possibly overlapping segments (considered as multiple realizations) with $G = 1/JK$. The sample estimate $\widehat{C}_{y,n}(\lambda_1, \ldots, \lambda_{n-1})$ is obtained by substituting moment estimates in the definitions of cumulants.

To gain insight into the utilization of HOS let us consider the following examples.

Example 1 Let $x(k) = \sum_n C_n e^{(j\omega_n k + \phi_n)}$ where, ϕ_n are independent identically distributed random variables uniformly distributed in the interval $[-\pi, \pi]$. This is a harmonic stationary process. Let $y(k)$ and $z(k)$ be the responses of a linear and a nonlinear system, respectively, both driven by $x(k)$. Then,

$$y(k) = \sum_n A_n e^{(j\omega_n k + \theta_n)},$$

$$z(k) = \sum_n B_n e^{(j\omega_n k + \psi_n)} + \sum_{m,l} B_m B_l e^{[j(\omega_m + \omega_l)k + \psi_m + \psi_l]} + \sum_{m,l,i} \cdots .$$

It can be shown that [3,9]:

$$C_{y,3}(\lambda_1, \lambda_2) = 0, \quad C_{z,3}(\lambda_1, \lambda_2) \neq 0.$$

In general, the polyspectra of a system output can be utilized in various ways in detecting as well as characterizing various types of nonlinearities in the system, e. g., detecting quadratic and cubic phase coupling in harmonic processes and identifying nonlinear *Volterra filters* driven by Gaussian processes among others, [9].

Example 2 Consider now a linear filtering problem where the linear time invariant (LTI) system with impulse response $\{f(k)\}$ is driven by a stationary random sequence $\{x(k)\}_{k=1,2,\ldots}$. Assuming that the system is stable, the following expression can be written for the n-order cumulant of the system output $\{y(k)\}$, [2,7]:

$$C_{y,n}(\lambda_1, \ldots, \lambda_{n-1}) = C_{x,n}(\lambda_1, \ldots, \lambda_{n-1}) * \left[\sum_{k=-\infty}^{\infty} f(k) f(k+\lambda_1) \cdots f(k+\lambda_{n-1}) \right], \tag{8}$$

where $*$ denotes $(n-1)$-dimensional linear convolution. In the special case where $\{x(k)\}$ is independent identically distributed, zero-mean, non–Gaussian, $\{f(k)\}$, $k = 0, \ldots, q$, is finite length and $\{w(k)\}$ is additive stationary zero-mean Gaussian noise statistically independent from $\{x(k)\}$, the following relations hold for the diagonal cumulants ($\lambda_i = \lambda$ for all i):

$$y(k) = \sum_{n=0}^{q} f(n) \cdot x(k-n) + w(k),$$

$$C_{y,2}(\lambda) = \gamma_{x,2} \cdot \sum_{k=0}^{q} f(k) f(k+\lambda) + C_{w,2}(\lambda),$$

$$C_{y,n}(\lambda, \ldots, \lambda) = \gamma_{x,n} \cdot \sum_{k=0}^{q} f(k) f^{n-1}(k+\lambda),$$

$$\lambda = -q, \ldots, 0, \ldots, q, \quad n \geq 3.$$

The $C_{y,2}(\lambda)$ is corrupted by noise. On the other hand, the $C_{y,n}(\lambda, \ldots, \lambda)$, $n \geq 3$, are noise free and proportional to the corresponding order correlation of the channel coefficients. Note that the second order cumulants (power spectrum) above do not preserve the true phase character of $f(k)$ unless the system is *minimum phase* (that is, all q zeros of its Z-transform are inside the unit circle). Thus, from $C_{y,2}(\lambda)$ alone only an equivalent

minimum phase system (within all-pass phase ambiguities) can be recovered. On the other hand, cumulants of order greater than two (polyspectra) preserve the true phase character of the system and thus are able to identify the system correctly up to a sign and possibly a constant linear phase term. This property, is the reason behind the wide utilization of HOS for blind system identification and deconvolution in seismic signal processing, dispersive communications channels, speech processing and other applications, [4,8].

To have a glance of optimization procedures that involve HOS, consider the recovery of the coefficients $f(k)$, in the linear filtering problem. The nonlinear-least squares approach proposes the minimization of the nonlinear function

$$\sum_{\lambda=-q}^{q}\left[\widehat{C}_{y,n}(\lambda,\ldots,\lambda)-\gamma_{x,n}\sum_{k=0}^{q}f(k)f^{(n-1)}(k+\lambda)\right]^2 \tag{9}$$

with respect to the unknown parameters $\{\gamma_{x,n}, f(k): k = 0,\ldots,q\}$, [2,6]. $\widehat{C}_{y,n}(\lambda,\ldots,\lambda)$ is the estimated diagonal cumulant from data samples. In practice the unknown order q must be estimated by means of model order selection criteria.

Minimization of (9) is a difficult problem which requires tedious searching programming techniques and proper initialization to avoid local equilibria. Thus, it is customary to seek solutions based on linear relations, [2,9,12]. Usually, a linear relation between the unknown parameters of the system model and the higher-order cumulants of the observed process is established. The solution is obtained by forming and solving an overdetermined linear system of equations. A variety of such algorithms have been proposed in the literature based on various system models.

Alternatively, we may consider the following deconvolution scenario, [1,3,4]:

$$\widetilde{x}(n) = u(n)*y(n) = [u(n)*f(n)]*x(n),$$

where $u(n)$ is an appropriate filter so that $\widetilde{x}(n) = A\cdot x(n-D)$ where D is a constant delay and A a constant phase term. Since the effect of linear filtering (i. e., convolution with $f(n)$) increases the Gaussianity of a random process (central limit theorem), then, inverse filtering (i. e., deconvolution with $u(n)$) must decrease the Gaussianity of the process. Based on this idea, deconvolution can rely on maximizing or minimizing an appropriate measure of Gaussianity such as the kurtosis, $\gamma_{\tilde{x},4}$, of $\widetilde{x}(n)$ with respect to the inverse filter coefficients. Actually, a variety of algorithms have been derived for deconvolution based on the constrained maximization of the objective function (for $m \neq r$, $m, r \geq 2$), [1]:

$$\frac{|\gamma_{\tilde{x},m}|^r}{|\gamma_{\tilde{x},r}|^m}.$$

To date the utilization of HOS in applications has been hampered by: i) the high computational complexity and the requirement for long data records in obtaining reliable estimates of HOS, and ii) the hard assumptions made regarding the stationarity and ergodicity of the available data. The emergence of faster digital hardware and the introduction of efficient HOS estimators will facilitate the wider utilization of HOS, [5,9]. A detailed coverage of signal processing algorithms and applications with HOS can be found in [9]. A recent extensive biography of HOS containing over 1700 entries has been compiled in [11].

See also

► Global Optimization Methods for Harmonic Retrieval

References

1. Cadzow JA (1996) Blind deconvolution via cumulant extrema. IEEE Signal Processing Magazine May:24–42
2. Giannakis GB, Mendel JM (1989) Identification of nonminimum phase systems using higher-order statistics. IEEE Trans Acoustics, Speech and Signal Processing 37:360–377
3. Hatzinakos D (1994) Higher-order spectral (H.O.S.) analysis. In: Control and Dynamic Systems, vol 65. Acad. Press, New York, pp 115–168
4. Haykin S (1994) Blind deconvolution. Prentice-Hall, Englewood Cliffs, NJ
5. Leung GCW, Hatzinakos D (1996) Implementation aspects of various higher-order statistics estimators. Franklin Inst 333(B):349–367
6. Lii KS, Rosenblatt M (1982) Deconvolution and estimation of transfer function phase and coefficients for non-Gaussian linear processes. Ann Statist 10:1195–1208
7. Mendel JM (1991) Tutorial on higher-order statistics (spectra) in signal processing and system theory: Theoretical results and some applications. Proc IEEE 79:278–305
8. Nikias CL, Mendel JM (1993) Signal processing with higher-order spectra. IEEE Signal Processing Magazine July:10–37

9. Nikias CL, Petropulu AP (1993) Higher-order spectral analysis: A non-linear signal processing framework. Prentice-Hall, Englewood Cliffs, NJ
10. Swami A (1993) Pitfalls in polyspectra. Proc IEEE ICASSP IV:97–100
11. Swami A, Giannakis GB, Zhou G (1997) Bibliography on higher-order statistics. Signal Processing 60:65–126
12. Tugnait JK (1987) Identification in linear stochastic system via second- and fourth-order cumulant matching. IEEE Trans Inform Theory 33(3):393–407

Simple Recourse Problem

SHANE DYE
University Canterbury, Christchurch, New Zealand

MSC2000: 90C06, 90C08, 90C15

Article Outline

Keywords
See also
References

Keywords

Stochastic programming; Recourse; Newsboy problem

A simple recourse problem is a stochastic linear program with recourse (cf. ► Stochastic linear programs with recourse and arbitrary multivariate distributions) for which the recourse action simply involves calculating linear penalties based on the surplus and shortfalls of scarce resources. In general all second stage parameters may be random.

The general simple recourse problem may be formulated as follows:

$$\min \left\{ cx + E_{\xi}[Q(x,\xi)] : \; Ax = b, x \geq 0 \right\}, \qquad (1)$$

where

$$Q(x,\xi) = \begin{cases} \inf & q^{+}(\xi)y^{+} + q^{-}(\xi)y^{-} \\ \text{s.t.} & y^{+} - y^{-} = h(\xi) - T(\xi)x, \\ & y^{+}, y^{-} \geq 0. \end{cases} \qquad (2)$$

Here $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^{m}$, and $c \in \mathbf{R}^{n}$ are given matrices. The uncertain parameters are $h\colon \mathbf{R}^{r} \to \mathbf{R}^{k}$, $q^{+}\colon \mathbf{R}^{r} \to \mathbf{R}^{k}$, $q^{-}\colon \mathbf{R}^{r} \to \mathbf{R}^{k}$, and $T\colon \mathbf{R}^{r} \to \mathbf{R}^{k\times n}$ where $\boldsymbol{\xi}$ is a random variable defined on the probability space (Ξ, F, μ) with $\Xi \subset \mathbf{R}^{r}$ the support of the measure μ. In the general case the probability distribution of $\boldsymbol{\xi}$ is continuous or has a very large number of realizations, which makes directly solving the deterministic equivalent very difficult.

One well-known example of a simple recourse problem is the *newsboy problem*. In this problem a newsboy must decide how many newspapers to order for sale the next day with only probabilistic information about the next day's demand. Unsold newspapers will be sold back to the supplier at a reduced rate and additional demand must be supplied to the customers but at a higher cost to the newsboy.

Three efficient methods have been developed for the case where only the right-hand side parameters are random. These methods are: the *primal method* (see ► Simple recourse problem: Primal method); the *dual method* (see ► Simple recourse problem: Dual method); and a method using the dualplex algorithm [1]. A good reference for the primal and dual methods for simple recourse problems is [2].

See also

► Combinatorial Optimization Algorithms in Resource Allocation Problems

References

1. Bryson NA, Gass SI (1994) Solving discrete stochastic linear programs with simple recourse by the dualplex algorithm. Comput Oper Res 21:11–17
2. Prékopa A (1995) Stochastic programming. Kluwer, Dordrecht

Simple Recourse Problem: Dual Method

SHANE DYE
University Canterbury, Christchurch, New Zealand

MSC2000: 90-08, 90C05, 90C06, 90C08, 90C15

Article Outline

Keywords
See also
References

Keywords

Stochastic programming; Recourse

The dual method for solving simple recourse problems (cf. ► Simple recourse problem), devised by A. Prékopa [1], is based on the dual simplex algorithm. As with the *primal method* (see ► Simple recourse problem: Primal method), this method only allows for uncertainty in the right-hand side parameters, and a finite, discrete probability distribution, while the stochastic dependence or independence of the random variables is not important.

This simple recourse problem may be formulated as follows:

$$\min \left\{ cx + \mathsf{E}_\xi [Q(x,\xi)] \colon \; Ax = b, x \geq 0 \right\}, \tag{1}$$

where

$$Q(x,\xi) = \begin{cases} \inf & q^+ y^+ + q^- y^- \\ \text{s.t.} & y^+ - y^- = \xi - Tx, \\ & y^+, y^- \geq 0. \end{cases} \tag{2}$$

Here $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$, $c \in \mathbf{R}^n$, $q^+ \in \mathbf{R}^r$, $q^- \in \mathbf{R}^r$, $q = q^+ - q^- > 0$, and $T \in \mathbf{R}^{r \times n}$ are given matrices. ξ is a random variable defined on the probability space (Ξ, F, μ) with $\Xi \subset \mathbf{R}^r$ the support of the measure μ.

Using linear programming duality, $\mathcal{E}_\xi [Q(x,\xi)]$ may be rewritten as

$$\sum_{i=1}^{r} \left(q_i^+ (\mathsf{E}_\xi [\xi_i] - T_i x) + q \int_{-\infty}^{T_i x} F_i(z)\, dz \right). \tag{3}$$

The objective function of (1) is then a piecewise-linear, convex function with breakpoints derived from the elements of Ξ.

Let $\xi_{i,1}, \ldots, \xi_{i,k_i}$ be the possible values of ξ_i in increasing order, with $p_{i,1}, \ldots, p_{i,k_i}$ the corresponding probabilities. Introduce $\xi_{i,0} < \xi_{i,1}$ and $\xi_{i,k_i+1} > \xi_{i,k_i}$ for $i = 1, \ldots, r$ with the property that $\xi_{i,0} < T_i x < \xi_{i,k_1+1}$ for all x feasible for (1) and put $p_{i,0} = p_{i,k_1+1} = 0$. The stochastic linear program (1) is reformulated as an equivalent (deterministic) linear program. The following notation is used:

$$f_{ij} = -q_i^+ + q(p_{i1}\xi_{i1} + \cdots + p_{i,j-1}\xi_{i,j-1}),$$

for $j = 1, \ldots, k_i + 1$, $i = 1, \ldots, r$, representing the function values at the breakpoints.

$$\begin{cases} \min & cx + f\lambda \\ \text{s.t.} & Ax = b, \\ & T_i x - \sum_{j=0}^{k_i+1} \xi_{ij} \lambda_{ij} = 0, \quad i = 1, \ldots, r, \\ & \sum_{j=0}^{k_i+1} \lambda_{ij} = 1, \quad i = 1, \ldots, r, \\ & x \geq 0, \quad \lambda \geq 0. \end{cases} \tag{4}$$

This linear program may be efficiently solved using the dual simplex method. All dual feasible bases have the following form. For some s, $1 \leq s \leq r$, there are $m + s$ basic x variables, $r - s$ of the i in $\{1, \ldots, r\}$ have basic variable pairs of the form $(\lambda_{i,j_i}, \lambda_{i,j_i+1})$ and the remaining s of these i have only one basic λ_{ij_i} variable.

The algorithm corresponds to finding an initial dual feasible basis then using the dual simplex method. Only columns corresponding to variables which might enter the basis need to be calculated at each step. This amounts to all nonbasic x variables and at most two λ_{ij} for each $i \in \{1, \ldots, r\}$ (the two immediately surrounding the basic λ_{ij} variables for each i).

See also

- ► Approximation of Extremum Problems with Probability Functionals
- ► Approximation of Multivariate Probability Integrals
- ► Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ► Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ► General Moment Optimization Problems
- ► Logconcave Measures, Logconvexity
- ► Logconcavity of Discrete Distributions
- ► L-shaped Method for Two-stage Stochastic Programs with Recourse
- ► Multistage Stochastic Programming: Barycentric Approximation
- ► Preprocessing in Stochastic Programming
- ► Probabilistic Constrained Linear Programming: Duality Theory
- ► Probabilistic Constrained Problems: Convexity Theory

- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Prèkopa A (1990) Dual method for the solution of one-stage stochastic programming problem with random RHS obeying a discrete probability distribution. Z Oper Res 34:441–461

Simple Recourse Problem: Primal Method

Shane Dye
University Canterbury, Christchurch, New Zealand

MSC2000: 90-08, 90C05, 90C06, 90C08, 90C15, 49M25

Article Outline

Keywords
See also
References

Keywords

Stochastic programming; Recourse; Working basis

The primal method for solving simple recourse problems (cf. also ▶ Simple recourse problem), devised by R.J-B. Wets [3], is based on the concept of a *working basis* introduced by G.B. Dantzig [1]. The primal method only allows for uncertainty in the right-hand side parameters, however stochastic dependence or independence of these parameters is not important to the method. A good reference for this algorithm is [2].

This simple recourse problem may be formulated as follows:

$$\min \left\{ cx + \mathsf{E}_\xi [Q(x,\xi)] \colon \; Ax = b, x \geq 0 \right\}, \tag{1}$$

where

$$Q(x,\xi) = \begin{cases} \inf & q^+ y^+ + q^- y^- \\ \text{s.t.} & y^+ - y^- = \xi - Tx, \\ & y^+, y^- \geq 0. \end{cases} \tag{2}$$

Here $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^m$, $c \in \mathbf{R}^n$, $q^+ \in \mathbf{R}^r$, $q^- \in \mathbf{R}^r$, $q = q^+ - q^- > 0$, and $T \in \mathbf{R}^{r\times n}$ are given matrices. ξ is a random variable defined on the probability space (Ξ, F, μ) with $\Xi \subset \mathbf{R}^r$ the support of the measure μ.

The method given here assumes ξ is a discrete random variable with finitely many realizations (i. e., Ξ is finite). For continuous distributions the method may be efficiently used on successively finer approximating distributions and error bounds calculated for each approximation.

Using linear programming duality, $\mathsf{E}_\xi [Q(x,\xi)]$ may be rewritten as

$$\sum_{i=1}^{r} \left(q_i^+ (\mathsf{E}_\xi [\xi_i] - T_i x) + q \int_{-\infty}^{T_i x} F_i(z)\, dz \right). \tag{3}$$

The objective function of (1) is then a piecewise-linear, convex function with breakpoints derived from the elements of Ξ.

Let $\xi_{i,1}, \ldots, \xi_{i,k_i}$ be the possible values of ξ_i in increasing order, with $p_{i,1}, \ldots, p_{i,k_i}$ the corresponding probabilities. Introduce $\xi_{i,0} < \xi_{i,1}$ and $\xi_{i,k_i+1} > \xi_{i,k_i}$ for $i = 1, \ldots, r$ with the property that $\xi_{i,0} < T_i x < \xi_{i,k_1+1}$ for all x feasible for (1) and put $p_{i,0} = p_{i,k_1+1} = 0$. The stochastic linear program (1) is reformulated as an

equivalent (deterministic) linear program. The following notation is used:

$$h_{ij} = \xi_{i,j} - \xi_{i,j-1},$$

for $j = 1, \ldots, k_i + 1$, $i = 1, \ldots, r$, representing the lengths of the intervals of the piecewise linear function, and

$$g_{ij} = -q_i^+ + q(p_{i0} + \cdots + p_{i,j-1}),$$

for $j = 1, \ldots, k_i + 1$, $i = 1, \ldots, r$, representing the gradients between breakpoints.

$$\begin{cases} \min & cx + gv \\ \text{s.t.} & Ax = b, \\ & T_i x - \sum_{j=1}^{k_i+1} v_{ij} = \xi_{i,0} \quad i = 1, \ldots, r, \\ & v + s = h, \\ & x \geq 0, \quad 0 \leq v, s \leq h. \end{cases} \tag{4}$$

This linear program may be solved using the simplex method only considering bases with the property that for each $i = 1, \ldots, r$ there is ℓ_i such that $v_{i1}, \ldots, v_{i\ell_i}$, $s_{i,\ell_i+1}, \ldots, s_{i,k_i+1}$ are basic, $s_{i1,\ldots}, s_{i,\ell_i-1}, v_{i,\ell_i+1}, \ldots, v_{i,k_i+1}$ are nonbasic, and $s_{i\ell_i}$ may or may not be basic. To reduce the amount of computation involved only so-called *key variables* need to be recorded as being basic. These are and basic x_j and basic $v_{i\ell}$ for which $s_{i\ell}$ is also basic. There are always $m+r$ key variables and the *working basis*, W, is given by the first $m + r$ rows of the columns corresponding to the key variables in the (full) basis.

The algorithm corresponds to finding an initial working basis, then using the (upper bounded) simplex method with only the working basis inverse stored. For this, all reduced costs may be calculated with little extra effort beyond that required for a linear program with as many variables and only $m + r$ constraints. The choice of the pivot column is constrained by the requirement of maintaining a basis with the required property. The calculation of the pivot row and the pivot step are essentially the same as for the upper bounded simplex method.

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Dantzig GB (1955) Upper bounds, secondary constraints, and block triangularity in linear programming. Econometrica 23:174–183
2. Prèkopa A (1995) Stochastic programming. Kluwer, Dordrecht
3. Wets RJ-B (1983) Solving stochastic programs with simple recourse. J Stochastics 10:219–242

Simplicial Decomposition

SIRIPHONG LAWPHONGPANICH
Naval Postgraduate School, Monterey, USA

MSC2000: 90C30

Article Outline

Keywords
See also
References

Keywords

Carathéodory theorem; Column generation; Convex combinations; Convex hull; Dantzig–Wolfe decomposition; Dimension; Extreme point; first order Taylor series expansion; Frank–Wolfe algorithm; Globally optimal; Inner linearization; Restriction; Linear program; Master problem; Node-arc incidence matrix; Nonlinear dynamic network flow problem; Nonlinear multicommodity flow problems; Nonlinear programming; Nonlinear program; Nonlinear single commodity network flow problem; Polyhedron; Polyhedral set; Positive definite matrix; Pseudoconvexity; P-simplex; Rank; regularized Frank–Wolfe algorithm; Regularized subproblem; Restricted simplicial decomposition; Shortest path problem; Simplex algorithm; Simplicial decomposition; Subproblem; Superlinear convergent rate; Traffic assignment problem; Side constraints

Simplicial decomposition (SD) can be viewed either as a generalization of the Frank–Wolfe algorithm [6] (cf. ► Frank–Wolfe algorithm) or an extension of Dantzig–Wolfe decomposition [5] to *nonlinear programs*. The term 'simplicial decomposition' is due to B. von Hohenbalken [22], but the essential idea is generally known as *column generation* and has been called *inner linearization/restriction* by A.M. Geoffrion [7].

In general, SD addresses the following problem

$$\min_{x\in S} f(x), \tag{1}$$

where $f(x)$ is *pseudoconvex*. The set S is typically a nonempty and bounded *polyhedron*, i. e., $S = \{x \in \mathbf{R}^n : Ax \le \mathrm{b}, x \ge 0\}$, A is a $m \times n$ matrix, and $b \in \mathbf{R}^m$. With S being bounded and *polyhedral*, problem (1) can be restated as

$$\begin{cases} \min & f\left(\sum_{i=1}^{n} \lambda_i Y^i\right) \\ \text{s.t.} & \sum_{i=1}^{n} \lambda_i = 1, \\ & \lambda_i \ge 0, \quad \forall i = 1, \ldots, n, \end{cases} \tag{2}$$

where n is the number of extreme points of S, each of which is represented as Y^i. In words, problem (2) finds a *convex combination of the extreme points*, Y^i, that minimizes $f(x)$. For real-world problems, the number of extreme points is generally large and it is impractical to generate all of them a priori. Instead, SD generates extreme points one at a time as follows.

Simplical decomposition technique

0 Select $x^1 \in S$ and set $k = 1$.

1 Let $Y^k = \operatorname{argmin}_{y\in S} \nabla f(x^k)^\top y$.
IF $\nabla f(x^k)^\top (Y^k - x^k) \ge 0$,
THEN stop and x^k is an optimal solution.
ELSE, go to Step 2.

2 Let

$$\lambda^k = \begin{cases} \arg\min\limits_{\lambda} & f(\lambda_0 Z^k + \sum\limits_{i\in I^k} \lambda_i Y^i) \\ \text{s.t.} & \lambda_0 + \sum\limits_{i\in I^k} \lambda_i = 1, \\ & \lambda_i \ge 0,\ \forall i \in I^k \cup \{0\}, \end{cases}$$

where $I^k \subseteq \{1, \ldots, k\}$, and $Z^k = 0$ or x^j for some $j \in \{1, \ldots, k\}$.
Set $x^{k+1} = \lambda_0^k Z^k + \sum\limits_{i\in I^k} \lambda_i^k Y^i$ and $k = k + 1$.
Return to Step 1.

Since S is polyhedral, the problem in Step 1 is a *linear program*, generally called the *subproblem*. When solved by the simplex algorithm, its solution, Y^k, is guaranteed to be an extreme point. In the event that x^k satisfies the stopping criterion, the following sequence of inequalities demonstrates that x^k must be globally optimal:

$$\forall x \in S\colon f(x) \geq f(x^k) + \nabla f(x^k)^\top (x - x^k)$$
$$\geq f(x^k) + \nabla f(x^k)^\top (Y^k - x^k) \geq f(x^k).$$

The three inequalities follow from the pseudoconvexity of $f(x)$, the fact that Y^k solves the subproblem, and the stopping criterion, respectively.

The problem in Step 2, or the *master problem*, is structurally the same as problem (2) and finds a convex combination of Z^k and extreme points in I^k that minimizes $f(x)$. This convex combination produces a new point, x^{k+1}, with a better objective value. To justify, consider the *first order Taylor series expansion* of $f(x)$, i. e.,

$$f(x^k + \lambda(Y^k - x^k))$$
$$= f(x^k) + \lambda \nabla f(x^k)^\top (Y^k - x^k)$$
$$+ \lambda \left\| Y^k - x^k \right\| \alpha(x^k; \lambda(Y^k - x^k)),$$

where $\lim_{\lambda \to 0} \alpha(x^k; \lambda(Y^k - x^k)) = 0$. When Step 2 is executed, $\nabla f(x^k)^\intercal (Y^k - x^k) < 0$ and the above expansion implies that there exists a sufficiently small $\widehat{\lambda} \in (0, 1)$ such that $f(x^k + \widehat{\lambda}(Y^k - x^k)) < f(x^k)$. When Z^k and I^k are properly defined (see below), $x^k + \widehat{\lambda}(Y^k - x^k)$ lies in the convex hull of Z^k and Y^i, for all $i \in I^k$. Since λ^k solves the master problem, the following must hold:

$$f(x^{k+1}) = f(\lambda_0^k Z^k + \sum_{i \in I^k} \lambda_i^k Y^k)$$
$$\leq f(x^k + \widehat{\lambda}(Y^k - x^k)) < f(x^k).$$

So, the objective value decreases after each iteration.

Note that Z^k is not necessarily an extreme point of S. However, Z^k and the (index) set I^k provide some flexibility and determine the number of iterations to achieve an optimal solution. For example, if $I^k = \{k\}$ and $Z^k = x^k$ in Step 2, then SD reduces to the Frank–Wolfe algorithm, which converges in the limit to an optimal solution.

When $I^0 = \emptyset$,

$$I^k = \left\{ i\colon\ i \in I^{k-1} \text{ and } \lambda_i^{k-1} > 0 \right\} \cup \{k\},$$

and $Z^k = 0$, the resulting algorithm is essentially the same as those in [11,22] and [23], and converges after a finite number of iterations. For this choice of Z^k and I^k, SD drops or discards extreme points with zero weight, $\lambda_i^{k-1} = 0$, to reduce the size of the master problem and, perhaps, to release computational resources for other uses as well. To obtain finite convergence, note that the number of possible index sets, I^k, is finite since there are only a finite number of extreme points for S. For each I^k generated by the algorithm, there is an associated minimum objective value, $f(x^{k+1})$, that is always decreasing for $k \geq 1$. This implies that the algorithm generates a sequence of distinct I^k. Since the number of possible I^k is finite, the sequence cannot be infinite, i. e., the algorithm must terminate finitely.

For the above choice of Z^k and I^k, the *Carathéodory theorem* (see, e. g., [2]) guarantees that the cardinality of I^k is at most rank(A) + 1. Thus, allocating computational resources for storing rank(A) + 1 extreme points is sufficient to ensure finite convergence. However, rank(A) + 1 is large for large scale problems and allocating such a large amount of resources may be impractical. As an alternative, D.W. Hearn, S. Lawphongpanich, and J.A. Ventura [10] proposed the following extreme point dropping scheme to restrict the cardinality of I^k to at most r, where $r \geq 1$.

1	When $k = 1$, $I^1 = \{1\}$ and $Z^1 = x^1$.
2	For $k > 1$, let β denote the cardinality of the set $\{i\colon\ i \in I^{k-1} \text{ and } \lambda_i^{k-1} > 0\}$.
2a	IF $\beta < r$, THEN set $I^k = \{i\colon\ i \in I^{k-1} \text{ and } \lambda_i^{k-1} > 0\} \cup \{k\}$ $Z^k = Z^{k-1}$.
2b	IF $\beta = r$, THEN set $I^k = \{i : i \in I^{k-1}, i \neq i^\#, \text{ and } \lambda_i^{k-1} > 0\} \cup \{k\}$ $Z^k = x^k$, where $i^\# = \text{argmin}_i\{\lambda_i^{k-1}\colon\ \lambda_i^{k-1} > 0\}$.

An extreme point dropping scheme

In Step 2, extreme points with zero weight, i. e., $\lambda_i^{k-1} = 0$, are dropped from the master problem in iteration k. When the number of remaining (or positively

weighted) extreme points is less than r, the new extreme point, Y^k, is added to the master problem (see step 2a). Otherwise, the new extreme point replaces one of remaining extreme points with the smallest weight (see step 2b) to keep the cardinality of I^k at r. The choices for Z^k in steps 2a and 2b ensure that Z^k and the extreme points in I^k always form a *p-simplex* (see [20]), a fact essential for proving finite convergence. With the above extreme point dropping scheme, the resulting SD, known as the *restricted simplicial decomposition* (RSD), converges finitely when problem (1) has a unique solution, x^*, and $r \geq \dim(\Phi) + 1$, where $\Phi = \{Y^i: \nabla f(x^*)^\intercal (Y^i - x^*) = 0, i = 1, \ldots, n\}$. (See [9].) When $r < \dim(\Phi) + 1$, RSD can be shown to converge in the limit to x^* using standard arguments in nonlinear programming (see, e. g., [9] and [15]).

In practice, a more successful application of SD is in solving *large nonlinear multicommodity flow problems* (see, e. g., [1]) of the form:

$$\begin{cases} \min\limits_{x} & f\left(\sum\limits_{c=1}^{C} x(c)\right) \\ \text{s.t.} & Ax(c) = b(c), \quad \forall c, \\ & x(c) \geq 0, \quad \forall c, \end{cases} \tag{3}$$

where A is a node-arc incidence matrix of a network with m nodes and n arcs, $b(c) \in \mathbf{R}^m$ is a supply/demand vector for each commodity c, $x(c) \in \mathbf{R}^n$ is a flow vector for commodity c, and $f(x)$ is a pseudoconvex travel cost function. In Step 1 of SD, the subproblem for problem (3) decomposes into C problems (one for each commodity c) of the following form:

$$y^*(c) = \begin{cases} \operatorname{argmin}\limits_{y} & \nabla f\left(\sum\limits_{c'=1}^{C} x^k(c')\right)^{\top} y \\ \text{s.t.} & Ay = b(c), \\ & y \geq 0. \end{cases} \tag{4}$$

Problem (4) is a *shortest path problem* and can be solved efficiently with specialized network algorithms (see, e. g., [1]). J.D. Murchland [18] first discussed SD as a method for solving problem (3) that is generally known as the *traffic assignment problem* in transportation science. D.G. Cantor and M. Gerla [4] (see also [8]) implemented SD for solving problem (3) to route messages in computer communication networks. Later, the results in [9] and [10] renewed the interest in SD by demonstrating empirically that RSD efficiently solves large traffic assignment problems. In [10], the master problem is solved by a method with at least a superlinear convergent rate, e. g., [3] and [17], and r is relatively small.

When applied to nonlinear single commodity (e. g., [10] and [17]) or *dynamic network flow problems* (e. g., [19]), SD may not be as efficient as other methods. For these problems, the dimension of Φ tends to be large and each extreme point in Φ contributes little as part of the convex combination that forms x^*.

In the literature, there are several extensions and modifications to SD, restricted or otherwise. First, it is claimed in [11] that SD also applies to problems in which S is convex, but not necessarily polyhedral. For example, $S = \{x \in \mathbf{R}^n: g_i(x) \leq 0, i = 1, \ldots, m\}$, where $g_i(x)$ is convex on $\mathbf{R}^n$. In this case, a straightforward application of SD (see [11]) would yield a subproblem with nonlinear constraints, a problem as complex as the original. Later, Ventura and Hearn [21] proposed a modification for RSD in which the subproblem is a linear program instead.

Second, when applied to problem (3), the representation of the extreme points can effect the convergence rate of SD. In particular, the extreme point, Y^k, can be represented either as $Y^k = \sum_{c=1}^{C} y^k(c)$, an aggregate form, or $(Y^k)^\intercal = (y^k(1)^\intercal, \ldots, y^k(C)^\intercal)$, a disaggregate form. The latter renders the master problem larger and more complex as shown below:

$$\begin{cases} \min & f\left(\sum\limits_{c=1}^{C}\left[\lambda_0(c) z^k(c) + \sum\limits_{i \in I^k} \lambda_i(c) y^k(c)\right]\right) \\ \text{s.t.} & \lambda_0(c) + \sum\limits_{i \in I^k} \lambda_i(c) = 1, \quad \forall c, \\ & \lambda_i(c) \geq 0, \quad \forall c \text{ and } i \in I^k \cup \{0\}. \end{cases}$$

Despite the increased in problem complexity, T. Larsson and M. Patriksson [12] demonstrated empirically that SD with disaggregate extreme points converges faster on several real-world traffic assignment problems.

Third, A. Migdalas [16] introduced an extension to the Frank–Wolfe algorithm called the *regularized Frank–Wolfe algorithm* in which the subproblem has a nonlinear term in the objective function to control the distance between Y^k and x^k. For example, one version

of the regularized subproblem is

$$Y^k = \operatorname*{argmin}_{y \in S} \left\{ \nabla f(x^k)^\top y + \frac{1}{2}(y - x^k)^\top D^k (y - x^k) \right\} ,$$

where D^k is a positive definite matrix. In [13], Larsson, Patriksson, and C. Rydergren solved the above subproblem approximately by performing several iterations of the Frank–Wolfe algorithm and showed empirically that the regularized subproblem can improve the convergence of SD.

Finally, C.H. Wu and Ventura [24] and Lawphongpanich [14] extended SD to solve problems with side constraints, i. e., $S = \{x \in \mathbf{R}^n : Ax \le b, Dx \le d, x \ge 0\}$, where A and b are as defined for problem (1), D is a $q \times n$ matrix, and $d \in \mathbf{R}^q$. Here, A may have a special structure that can be exploited computationally, and D, representing the side constraints, does not.

See also

- ► Decomposition Principle of Linear Programming
- ► Generalized Benders Decomposition
- ► MINLP: Generalized Cross Decomposition
- ► MINLP: Logic-based Methods
- ► Simplicial Decomposition Algorithms
- ► Stochastic Linear Programming: Decomposition and Cutting Planes
- ► Successive Quadratic Programming: Decomposition Methods

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs, NJ
2. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: Theory and algorithms. Wiley, New York
3. Bertsekas DP (1982) Projected Newton methods for optimization problems with simple constraints. SIAM J Control Optim 20:221–246
4. Cantor DG, Gerla M (1974) Optimal routing in a packet switched network. IEEE Trans Computers c-23:1062–1069
5. Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8:101–111
6. Frank M, Wolfe P (1956) An algorithm for quadratic programming. Naval Res Logist Quart 3:95–110
7. Geoffrion AM (1970) Elements of large-scale mathematical programming. Managem Sci 16:652–691
8. Gerla M (1973) The design of store-and-forward (S/F) networks for computer communications. PhD Thesis Dept Computer Sci Univ Calif, Los Angeles
9. Hearn DW, Lawphongpanich S, Ventura JA (1985) Finiteness in restricted simplicial decomposition. Oper Res Lett 4:125–130
10. Hearn DW, Lawphongpanich S, Ventura JA (1987) Restricted simplicial decompsition: Computation and extension. Math Program 31:99–118
11. Holloway C (1974) An extension of the Frank–Wolfe method of feasible directions. Math Program 6:14–27
12. Larsson T, Patriksson M (1992) Simplicial decomposition with disaggregate representation for the traffic assignment problem. Transport Sci 26:4–17
13. Larsson T, Patriksson M, Rydergren C (1997) Applications of simplicial decomposition with nonlinear column generations to nonlinear network flows. In: Pardalos PM, Hearn DW, Hager WH (eds) Network Optimization. Lecture Notes Economics & Math Systems. Springer, Berlin, pp 346–373
14. Lawphongpanich S (2000) Simplicial with truncated Dantzig–Wolfe decomposition for nonlinear multicommodity network flow problems with side constraints. Oper Res Lett 26:33–41
15. Lawphongpanich S, Hearn DW (1986) Restricted simplicial decomposition with applications to the traffic assignment problem. Ricerca Oper 38:97–120
16. Migdalas A (1994) A regularization of the Frank–Wolfe method and unification of certain nonlinear programming method. Math Program 65:331–346
17. Mulvey J, Zenios SA, Ahlfeld DP (1990) Simplicial decomposition for convex generalized networks. J Inform Optim Sci 11:359–387
18. Murchland JD (1970) Road network traffic distribution in equilibrium. In: Henn R, Künzi HP, Schubert H (eds) Proc. Math. Models in the Social Sci. Anton Hain Verlag, Meisenheim, pp 145–183
19. Powell WB (1993) On algorithms for nonlinear dynamic networks. In: Du D-Z, Pardalos PM (eds) Network Optimization Problems. World Sci., Singapore, pp 203–231
20. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton
21. Ventura JA, Hearn DW (1993) Restricted simplicial decomposition for convex constrained problems. Math Program 59:71–85
22. Von Hohenbalken B (1975) A finite algorithm to maximize certain pseudo-concave function on polytopes. Math Program 8:189–206
23. Von Hohenbalken B (1977) Simplicial decomposition in nonlinear programming algorithms. Math Program 13:49–68
24. Wu C-H, Ventura JA (1994) Restricted simplicial decomposition with side constraints and its application to capacitated multicommodity networks. In: IME Working Paper Dept. Industr. and Management Systems Engin., Penn. State Univ., vol 94-128

Simplicial Decomposition Algorithms

MICHAEL PATRIKSSON
Department Math., Chalmers University Technol., Göteborg, Sweden

MSC2000: 90C06, 90C25, 90C35

Article Outline

Keywords
See also
References

Keywords

Inner approximation; Representation theorem; Carathéodory's theorem; Simplex; Linearization; Pseudoconvex function; Decomposition/coordination; Column generation; Restricted master problem; Dantzig–Wolfe decomposition; Pricing-out; Column dropping; Optimal face; Frank–Wolfe; Disaggregated representation; Variational inequalities; Merit function; Nondifferentiable convex program

Simplicial decomposition (SD) is a class of methods for solving continuous problems in mathematical programming with convex feasible sets. There are two main characteristics of the methods in this class:

i) an approximation of the original problem is constructed and solved, wherein the original feasible set is replaced by a polyhedral subset thereof, that is, an *inner approximation* of it which is spanned by a finite set of feasible solutions; and
ii) this inner approximation is improved (that is, enlarged) by generating a vector (or, column) in the feasible set through the solution of another approximation of the original problem wherein the original cost function is approximated (often by a linear function).

As such, the class of SD methods may be placed within the framework of *column generation* methods. Another characteristic of an SD method however is that the sequence of solutions to the inner approximated problems tends to a solution to the original problem in such a way that the cost function (or, some merit function) strictly monotonically approaches its optimal value. Therefore, the class of SD methods also falls within the framework of iterative descent (or, ascent) algorithms for continuous mathematical programs.

We consider, for the most part, the solution of the differentiable optimization problem

$$\begin{cases} \min & f(x), \\ \text{s.t.} & x \in X, \end{cases} \tag{1}$$

where $f\colon X \to \mathbf{R}$ is *pseudoconvex* on X (that is, for any $x, y \in X$, $\nabla f(x)^{\mathsf{T}}(y - x) \geq 0$ implies $f(x) \leq f(y)$), and where $X := \{x \in \mathbf{R}^n : Ax = b; x \geq 0^n\}$ is a nonempty polyhedral set.

The derivation of the method rests on two classical results on the representation of convex sets and of points in such sets. The first result is the *representation theorem* (e. g., [2,18]), which states that:

i) the set of extreme points p^i, $i \in \mathcal{P}$, of the polyhedral set X is nonempty and finite;
ii) the set of extreme directions d^i, $i \in \mathcal{D}$, is empty if and only if X is bounded, and if X is not bounded, then it is nonempty and finite; finally, and most importantly,
iii) a vector $x \in \mathbf{R}^n$ belongs to X if and only if it can be represented as a convex combination of the extreme points plus a nonnegative linear combination of the extreme directions, that is, for some vectors λ and μ,

$$x = \sum_{i \in \mathcal{P}} \lambda_i p^i + \sum_{i \in \mathcal{D}} \mu_i d^i, \tag{2a}$$

$$\sum_{i \in \mathcal{P}} \lambda_i = 1, \tag{2b}$$

$$\lambda_i \geq 0, \qquad i \in \mathcal{P}, \tag{2c}$$

$$\mu_i \geq 0, \qquad i \in \mathcal{D}. \tag{2d}$$

Thus, in principle, the polyhedral set X can be given an inner representation in terms of extreme points and directions, and the problem (1) can be cast in the variables λ_i and μ_i instead of in x. The advantage of making this problem transformation is that the inner representation of X is much simpler than its original, outer, representation in terms of linear equalities and inequalities; disregarding the definitional constraints (2a), the set described by (2) is the Cartesian product of a *simplex* and the nonnegative orthant, an optimization over which often can be made with little more effort than for an unconstrained problem (e. g., [3,4]). Furthermore,

the inner representation may also be useful for interpreting the result of an optimization, since the extreme points and directions may have further significance; in applications to network flows, for example, the representation theorem states that a link flow $x \in X$, where the polyhedral set X describes the flow conservation and nonnegativity constraints for the network flow, can equivalently be represented by (or, decomposed into) the sum of flows on routes and in cycles (e. g., [1, Thm. 3.5]). The latter also explains the origin of the term *simplicial decomposition*: the variable transformation decomposes a feasible solution into the sum of variables that (in the bounded case) forms a simplex.

The disadvantage of the transformation is that since the number of extreme points and directions of a polyhedral set grows exponentially with its dimension, the transformation introduces an impractically large number of variables. The practical use of simplicial decomposition then hinges on the second basic result in the representation of convex sets, *Carathéodory's theorem* (e. g., [26, Thm. 17.1]). This result states that a point x in the convex hull of any subset X of $\mathbf{R}^n$ can be represented as a convex combination of at most as many elements of X as its dimension, dim$X$ (which is defined as the dimension of its affine hull), plus one. (This number is not larger than $n + 1$.) Although Carathéodory's theorem is not stated in terms of extreme points and directions, its natural application in the context of simplicial decomposition is that, in the case of a bounded polyhedral set, for example, any feasible point can be described as the convex combination of extreme points of the set, the total number of which need never exceed the dimension of the polyhedron plus one. (This result obviously refines the representation theorem.)

The classical form of the simplicial decomposition method was first described by B. von Hohenbalken [31] (see however the end of this article for some earlier references to similar algorithms) for the problem (1). The algorithm alternates between the solution of two problems. Given known subsets $\widehat{\mathcal{P}}$ and $\widehat{\mathcal{D}}$ of $\mathcal{P}$ and $\mathcal{D}$, respectively, f is minimized over the inner approximation of X which is defined when these subsets replace $\mathcal{P}$ and $\mathcal{D}$ in (2), in terms of the variables $\widehat{\lambda}_i$, $i \in \widehat{\mathcal{P}}$, and $\widehat{\mu}_i$, $i \in \widehat{\mathcal{D}}$. (We will denote this problem the *restricted master problem* (RMP); it is also sometimes referred to as the *coordination step*.) Notice that we use the notation $\widehat{\lambda}$ and $\widehat{\mu}$ to distinguish the vectors in the RMP from the (longer) vectors λ and μ in the *complete master problem* which is equivalent to (1) and is defined by the system (2). Further denoting by $\widehat{\Lambda}$ the set of vectors $(\widehat{\lambda}, \widehat{\mu})$ satisfying the restriction of the system (2b)–(2c) to the known subsets $\widehat{\mathcal{P}}$ and $\widehat{\mathcal{D}}$ and utilizing (2a) to substitute x for $(\widehat{\lambda}, \widehat{\mu})$ (we write $x = x(\widehat{\lambda}, \widehat{\mu})$), the RMP may then be formulated as

$$\begin{cases} \min & f(x(\widehat{\lambda}, \widehat{\mu})), \\ \text{s.t.} & (\widehat{\lambda}, \widehat{\mu}) \in \widehat{\Lambda}. \end{cases} \tag{3}$$

Alternately, a profitable extreme point or direction of X is generated through the solution of an approximation of (1), in which f is replaced by its first order, linear, approximation, $y \longmapsto f(x) + \nabla f(x)^{\mathsf{T}}(y - x)$, defined at the solution, x, to the RMP (3), that is, by the problem

$$\begin{cases} \min & \nabla f(x)^{\top} y, \\ \text{s.t} & y \in X; \end{cases} \tag{4}$$

this approximate problem is a linear programming problem, which in general is much easier to solve than the original one. (This is called the *column generation subproblem*, and corresponds to the *decomposition* step in some descriptions of column generation methods.) If the solution to this problem lies within the current inner approximation, then the conclusion is that the current solution, x, is optimal in (1), since, then, $\nabla f(x)^{\mathsf{T}}(y - x) \geq 0$ must hold for all $y \in X$. Otherwise, $\widehat{\mathcal{P}}$ or $\widehat{\mathcal{D}}$ is augmented by a new element, the resulting inner approximation is improved (that is, enlarged), and the solution to the new RMP has a strictly lower objective value than the previous one; the latter result follows since the strict inequality $\nabla f(x)^{\mathsf{T}} d < 0$ holds (that is, d defines a direction of descent with respect to f at x), where d denotes either the direction $d := y - x$ towards the new extreme point y or an extreme direction. The iteration is then repeated with the solution of a new column generation subproblem defined at the solution to the RMP. In the method of [31], Carathéodory's theorem is utilized in the validation of a *column dropping rule*, according to which any extreme point or direction whose weight in the expression of the solution x to the RMP is zero is removed; thanks to the finiteness of $\mathcal{P}$ and $\mathcal{D}$ and the strictly decreasing values of f, the convergence of the SD algorithm in the number of RMP is finite.

In the case of a convex function f, a finite termination criterion is automatically supplied, on the one hand by the upper bound on the optimal objective value of (1) that is defined by the solution to the RMP, and on the other hand by the lower bound that is supplied by the solution to the column generation subproblem; in fact, letting x be an arbitrary feasible solution to (1), and x^* be any optimal solution to (1), we obtain from the optimality of x^* and by the convexity of f that

$$\begin{aligned} f(x) &\geq f(x^*) \\ &\geq f(x) + \nabla f(x)^\top (x^* - x) \\ &\geq f(x) + \min_{y \in X} \{\nabla f(x)^\top (y - x)\}; \end{aligned} \tag{5}$$

note that the problem defined in (5) is precisely the column generation subproblem (4) defined at x. A finite termination criterion for the solution of the RMP can be defined by the analogous lower bound. In this case, due to the simple form of the constraints defined by (2b)–(2d), the lower bound is available directly from the value of the gradient of f with respect to $(\widehat{\lambda}, \widehat{\mu})$ at $x = x(\widehat{\lambda}, \widehat{\mu})$. Indeed, the lower bound is either (analogously to (5)) given by the current objective value plus

$$\min_{i \in \widehat{\mathcal{P}}} \left\{ \frac{\partial f(x(\widehat{\lambda}, \widehat{\mu}))}{\partial \widehat{\lambda}_i} \right\} - \nabla_{\widehat{\lambda}} f(x(\widehat{\lambda}, \widehat{\mu}))^\top \widehat{\lambda} - \nabla_{\widehat{\mu}} f(x(\widehat{\lambda}, \widehat{\mu}))^\top \widehat{\mu}$$

or it is minus infinity (if $\nabla f(x)^\top d^i < 0$ holds for some $i \in \widehat{\mathcal{D}}$).

Assume now that the function f is *strictly pseudoconvex* (that is, for any $x, y \in X$ with $x \neq y$, $\nabla f(x)^\top (y - x) \geq 0$ implies that $f(x) < f(y)$ holds), so that the optimal solution x^* is unique, and for simplicity we also assume that X is bounded. For such problems, an improvement over the original scheme was devised in [10,11]. The basis for the improvement is the observation that a particular feasible solution, such as the optimal one, can be represented as the convex combination of an often much smaller number of extreme points than $\dim X + 1$, as implied by Carathéodory's theorem; in fact, the highest number of extreme points needed to describe the optimal solution x^* is $\dim F^* + 1$, where F^* is the *optimal face* of X, that is, the face of X of the smallest dimension which contains x^*. (In the present context, this set may be described by

$$F^* = \left\{ y \in X : \ \nabla f(x^*)^\top (y - x^*) = 0 \right\},$$

a set which is spanned by the extreme points of X that solve the linear approximation (4) to (1) defined at the optimal solution.) Based on this observation, they devise a modification of the original scheme, in which the number of extreme points retained is kept below a positive integer, r; when this number of extreme points has been reached, any new extreme point generated replaces the column in $\widehat{\mathcal{P}}$ that received the least weight in the solution to the RMP. In order to ensure the convergence of the algorithm, the optimal solution x to the RMP must also be retained as an individual column (however not counted among the r columns). They show that the modified algorithm is finitely convergent in the number of RMP, provided that $r \geq \dim F^* + 1$. Referred to as restricted simplicial decomposition (RSD), the scheme is shown below, for the case of a bounded set X.

PROCEDURE RSD(r)
(Init): $x^0 \in X$, $p^x := x^0$, $\widehat{\mathcal{P}} = \emptyset$, $t := 0$.
(Sub): Solve (4) defined at $x^t \Rightarrow p^{i_t}$, $i_t \in \mathcal{P}$.
(Augment): $i_t \in \widehat{\mathcal{P}} \Rightarrow x^t$ is optimal.
| $\widehat{\mathcal{P}}$ |$= r \Rightarrow$ replace an element of $\widehat{\mathcal{P}}$ by i_t.
| $\widehat{\mathcal{P}}$ |$< r \Rightarrow \widehat{\mathcal{P}} := \widehat{\mathcal{P}} \cup \{i_t\}$.
(Master): x^{t+1} minimizes f over the convex hull of p^x and p^i, $i \in \widehat{\mathcal{P}}$.
(Update): Let $p^x := x^{t+1}$, $t := t + 1$.
Go to the Subproblem.
END

The value of r is crucial to the performance of the algorithm. If $r \geq \dim F^* + 1$, then since the number of RMP is finite, the local rate of convergence is governed by the local convergence rate of the method chosen for the solution of the RMP; thus, a superlinear or quadratic convergence rate may be attained if a (projected) Newton method is used ([11]). If $r < \dim F^* + 1$, however, then the algorithm is only asymptotically convergent, and the rate of convergence is the same as that of the *Frank–Wolfe algorithm* (or, *conditional gradient method*; which is actually obtained as a special case of RSD when $r := 1$), that is, the convergence rate is sublinear. Since the threshold value for finite convergence cannot be estimated from the original data and thus is unknown a priori, the proper value of r must in general be based on computational experience.

In column generation methods for linear programs, such as *Dantzig–Wolfe decomposition* ([16]), the columns are generated through the *pricing* operation of the *simplex method* in linear programming, which utilizes an estimate of the dual optimal solution. Its extension to nonlinear programming, *nonlinear Dantzig–Wolfe decomposition* (e. g., [18]), also utilizes the pricing operation in the construction of the column generation subproblem, which in both cases is equivalent to the result of performing a Lagrangian relaxation of the original problem using the current estimate of the vector of Lagrange multipliers; while the column generation subproblem is nonlinear in the latter algorithm, the RMP are in both cases linear programs. In contrast, the class of SD algorithms are column generation methods where the columns are generated through the solution of the primal, linearized problem (4), and which thus does not utilize dual information. However, it is established in [14] that Dantzig–Wolfe decomposition is in fact a special case of simplicial decomposition, when the latter is applied to a primal-dual (saddle point) reformulation of the linear program. Also for linearly constrained nonlinear programs of the form (1), simplicial decomposition may be based on the *pricing-out* of a subset of the linear constraints. Identifying a subset of the constraints defining X as complicating, these may be priced-out (that is, Lagrangian relaxed) in the column generation subproblem, and instead included in the master problem, just as in Dantzig–Wolfe decomposition for linear and nonlinear programming problems. Such methods have been devised in [20,28]. It should be noted, however, that just as in the original (primal) SD method, the column generation subproblems in these methods are based on the linearization of the original objective function, and are therefore linear programs, and their RMP are nonlinear; this is precisely the opposite to the case of nonlinear Dantzig–Wolfe decomposition.

The RSD algorithm has been successfully applied to large scale, structured nonlinear optimization problems, in particular mathematical programming models of various nonlinear network flow problems, where the column generation subproblem reduces to efficiently solvable linear network flow problems (e. g., [11,15,23]).

Other special structures in the feasible set X may also be taken into account efficiently in the construction of an SD method. For example, assume that the set X is a Cartesian product of polyhedral sets X_k in smaller dimensions $\mathbf{R}^{n_k}$, with $k \in \mathcal{K}$ and $\sum_{k\in\mathcal{K}} n_k = n$. (In network flow problems, k could denote a commodity of goods to be transported or a pair of origin and destination in an urban transportation network.) Noting that the linear column generation subproblem decomposes into $|\mathcal{K}|$ independent linear column generation problems, it is possible to store extreme points and directions of the individual (smaller-dimensional) sets X_k rather than extreme points and directions of X. The RMP of such a *disaggregate simplicial decomposition* (DSD) method ([15]) would then have variables of the form λ_{ik}, $i \in \mathcal{P}_k$, $k \in \mathcal{K}$, and likewise for μ_{ik}, and $|\mathcal{K}|$ convexity constraints (2b) instead of only one as in the SD method. The total number of extreme points and directions is much less in the disaggregated representation ($\sum_{k\in\mathcal{K}}\{\mathcal{P}_k + \mathcal{D}_k\}$ in the disaggregated case, and $\prod_{k\in\mathcal{K}}\{\mathcal{P}_k + \mathcal{D}_k\}$ in the aggregated case; [24]). On the other hand, according to Carathéodory's theorem, the total number of columns needed to express an optimal solution is in this case bounded above by $\sum_{k\in\mathcal{K}}(\dim F_k^* + 1)$, which may be a much higher number than $\dim F^* + 1$. This result notwithstanding, it has been observed in applications of Dantzig–Wolfe decomposition to linear multicommodity network flow problems ([13]) that a disaggregated representation of the solution (in this case, as commodity route flows instead of as aggregated link flows) speeds up the convergence of the method. The same conclusion has been drawn from applications to nonlinear multicommodity network flow problems ([15]) of the DSD algorithm.

Experience with the RSD method has shown that it makes rapid progress initially, quickly reaching a near-optimal solution, especially when relatively large values of r are used and when second order methods are used for the solution of the RMP, but that it slows down close to an optimal solution. It is also relatively less efficient for larger values of $\dim F^*$.

The explanation for this behavior is to be found in the construction of the column generation subproblem, which utilizes first order approximations of f. The column generation subproblem of RSD is the same as that of the Frank–Wolfe (FW) method mentioned earlier, the quality of whose search directions are known to deteriorate rapidly. The reason is that as the sequence $\{x^t\}$

tends to an optimal solution, the sequence $\{\nabla f(x^t)^\top d^t\}$ of directional derivatives of the search directions $d^t := y^t - x^t$ tends to zero whereas $\{d^t\}$ does not; thus, the search directions rapidly tend to become orthogonal to the gradient of f, and the result of the deteriorating descent property is a decreasing step length in the line search of this algorithm.

We then make the observation that the RSD method is similar to FW, since the same descent direction-generating subproblem is used, and that the only difference between FW and RSD lies in the updating phase, the latter algorithm using a multidimensional search (when $r \geq 2$) rather than a one-dimensional search, which one may interpret as a devise for reducing the zig-zagging effect inherent in the FW algorithm. It is a natural conclusion from this discussion that better approximations of f could be exploited in the column generation phase of SD methods, since then the columns generated would be of better quality, thus leading to larger improvements in the inner approximations of the feasible set. (The counterpart in line search methods for (1) is that improved approximations of f in a direction-finding subproblem yield better search directions.)

An extension of the RSD algorithm was made by T. Larsson, M. Patriksson and C. Rydergren [16] based on this observation. The motivation behind the *nonlinear simplicial decomposition* (NSD) method is that by generating columns based on better approximations of the objective function, the sensitivity of the method to the dimension of the optimal face will be reduced, fewer columns will be needed to describe an optimal solution, resulting in fewer iterations, and enabling a smaller value of the parameter r to be chosen.

The NSD method is obtained from the RSD method by replacing the linear column generation subproblem (4) with

$$\min_{y \in X} \{\nabla f(x^t)^\top y + \varphi(y, x^t)\}, \tag{6}$$

where φ: $X \times X \to \mathbf{R}$ is a continuous function of the form $\varphi(y, x)$, convex and continuously differentiable with respect to y for all $x \in X$, and with the property that $\nabla_y \varphi(x, x) = 0^n$ for all $x \in X$. Among the possible choices for φ we mention the following, where diag denotes the diagonal part of the matrix and where $\gamma_t > 0$:

$\varphi(y, x^t)$	Subproblem
0	Frank-Wolfe
$\frac{1}{2} y^\top \nabla^2 f(x^t) y$	Newton
$\frac{1}{2} y^\top [\mathrm{diag} \nabla^2 f(x^t)] y$	Diag. Newton
$\frac{1}{2\gamma^t} y^\top y$	Projection

Even though the finite convergence property will be lost (because nonextremal points will be generated), one may expect a more rapid convergence of the NSD method than the RSD method, both in terms of the number of iterations needed to reach a given solution accuracy and in terms of the required solution time, provided however that the nonlinear column generation subproblems can be efficiently solved, at least approximately. In numerical experiments performed on large scale nonlinear network flow problems, such conclusions were indeed made. It was particularly observed that the NSD method is relatively much less sensitive to the value of $\dim F^*$ than is RSD, which permits the use of a much smaller value of r in the NSD method.

Convergence results for SD methods allow for both the column generation subproblem and the RMP to be solved inexactly, thus facilitating its practical use. In [11], convergence is established for the RSD method, wherein the RMP is solved using one iteration of Newton's method only. Further developments along these lines are found in [8,14] for a general class of SD methods and in [25, Chap. 9] for the NSD method. The convergence results established in the latter reference not only validate inexact computations but also quite arbitrary rules both for defining and for dropping columns.

SD algorithms have been extended to handle nonlinear constraints as well. In [30], the column generation subproblem is made linear by approximating the nonlinear constraint functions by piecewise linear functions, reminiscent to the Topkis–Veinott scheme ([29]). The NSD method of [16,25] applies to general convex sets X directly. A combination of sequential quadratic programming (SQP) and NSD is devised in [25, Chap. 9]. There, in the column generation subproblem one replaces the nonlinear constraints with linear approximations, and dual information about these constraints is included in the objective function. SD methods for nonlinearly constrained problems are believed to be efficient, if the nonlinearity is mild, as

concluded from the computational results of [30]; however, few applications of SD methods to nonlinearly constrained problems have been reported.

The class of SD methods has furthermore been extended to the solution of the general *variational inequality problem* (VIP) of finding $x^* \in X$ such that

$$F(x^*)^\top (x - x^*) \geq 0, \qquad \forall x \in X, \tag{7}$$

where $F: X \to \mathbf{R}^n$ is a continuous and monotone mapping. If the mapping F satisfies $F \equiv \nabla f$, that is, it is the gradient of f, then the VIP defines the first order optimality conditions for x^* in (1), and the SD method for the VIP becomes that for (1). If this is not the case, then F replaces ∇f in the column generation subproblem (4), and the RMP is defined as the restriction of the variational inequality problem (7) to the currently known inner approximation of X. Further, in this case there is no objective function (or, *merit function*) immediately available for monitoring the convergence of the SD method. Column dropping rules in SD methods for the VIP must however be based on the improvement of the method in terms of some merit function, without which the method may cycle. (This is evidenced by the nonconvergence of the FW method applied to the VIP; see [9].) S. Lawphongpanich and D.W. Hearn [19] utilize the *primal gap function*,

$$\psi(x) := \max_{y \in X} F(x)^\top (x - y),$$

which is zero at solutions to VIP and positive elsewhere in X, to guide the dropping of columns. A related merit function is used in [27]. The NSD and NSD/SQP methods are extended to VIP in [25, Chap. 9], there using the merit function

$$\psi(x) := \max_{y \in X} \{F(x)^\top (x - y) - \varphi(y, x)\}.$$

In contrast to the case of the problem (1), the sequence of solutions to the RMP in SD methods applied to (7) does not necessarily yield a monotonically decreasing sequence of values of any merit function for the VIP unless very restrictive assumptions are made on the original data, whence the theoretical properties of SD methods for VIP are less strong; for example, a consequence of the property just mentioned is that the finite convergence result for the RSD method cannot be transferred to the VIP. The solution methods that have been considered for the RMP for the VIP are generalizations of those used for the RMP of (1); the most popular ones are projection algorithms, due to a large degree to the simple form of the constraints of the RMP; see, e. g., [22] for numerical investigations of different algorithmic approaches to the RMP.

Larsson, Patriksson and A.-B. Strömberg [17] develop an SD scheme for *nondifferentiable convex optimization*. There, the gradient $\nabla f(x)$ is replaced by the set of subgradients, the *subdifferential*, $\partial f(x)$, defined by

$$\partial f(x) = \left\{ \xi_f \in \mathbb{R}^n : \quad \begin{array}{c} f(y) \geq f(x) + \xi_f^\top (y - x), \\ \forall y \in \mathbb{R}^n \end{array} \right\}.$$

It is shown that the utilization of an arbitrary subgradient $\xi_f \in \partial f(x)$ in place of the gradient in the column generation subproblem may lead to the termination of the algorithm at a nonoptimal solution, since not all subgradients define descent directions. A modification of the SD scheme is therefore made, wherein the subgradients evaluated in the course of solving the RMP are averaged with weights proportional to the step lengths used in the solution method for RMP; this vector of averaged (or, ergodic) subgradients is then shown to yield an improved inner approximation. An alternative, and probably computationally much more efficient, means to define a linear column generation subproblem with the properties required is through the generation of (approximately) shortest ε-subgradients.

The first simplicial decomposition type methods evolved from the experience of the poor convergence of the FW method (e. g., [5,12,21]). The perhaps first thorough theoretical investigation of SD methods is due to C.A. Holloway [12]. Its close relationships to column generation methods however make it difficult to trace its earliest history; for example, already the first urban transportation planning studies in the 1950s applied heuristics resembling the DSD algorithm (see [24] for an overview of the history of such methods). A related class of methods also exist for least-distance and other problems in quadratic programming (e. g., [7]).

Further surveys on simplicial decomposition methods, their history and their relationships to column generation, are found in [15,24,25].

See also

- ► Decomposition Principle of Linear Programming
- ► Generalized Benders Decomposition

- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Logic-based Methods
- ▶ Simplicial Decomposition
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Successive Quadratic Programming: Decomposition Methods

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs, NJ
2. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: Theory and algorithms, 2nd edn. Wiley, New York
3. Bertsekas DP (1982) Constrained optimization and Lagrange multiplier methods. Acad. Press, New York
4. Bertsekas DP (1995) Nonlinear programming. Athena Sci., Belmont, MA
5. Cantor DG, Gerla M (1974) Optimal routing in a packet-switched computer network. JEEE Trans Computers c-23:1062–1069
6. Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8:101–111
7. Djang A (1980) Algorithmic equivalence in quadratic programming. PhD Thesis Dept. Oper. Res. Stanford Univ
8. García R, Marín A, Patriksson M (1999) A class of column generation/simplicial decomposition algorithms in convex differentiable optimization. Techn. Report Dept. Math. Chalmers Univ. Technol., Gothenburg, Sweden
9. Hammond JH (1984) Solving asymmetric variational inequality problems and systems of equations with generalized nonlinear programming algorithms. PhD Thesis Dept. Math. Massachusetts Inst. Techn.
10. Hearn DW, Lawphongpanich S, Ventura JA (1985) Finiteness in restricted simplicial decomposition. Oper Res Lett 4:125–130
11. Hearn DW, Lawphongpanich S, Ventura JA (1987) Restricted simplicial decomposition: computation and extensions. Math Program Stud 31:99–118
12. Holloway CA (1974) An extension of the Frank and Wolfe method of feasible directions. Math Program 6:14–27
13. Jones KL, Lustig IJ, Farvolden SM, Powell WB (1993) Multicommodity network flows: the impact of formulation on decomposition. Math Program 62:95–117
14. Larsson T, Migdalas A, Patriksson M (1994) A generic column generation scheme. Report Dept. Math. Linköping Inst. Techn. LiTH-MAT-R-94-18
15. Larsson T, Patriksson M (1992) Simplicial decomposition with disaggregated representation for the traffic assignment problem. Transport Sci 26:4–17
16. Larsson T, Patriksson M, Rydergren C (1997) Applications of simplicial decomposition with nonlinear column generation to nonlinear network flows. In: Pardalos PM, Hager WW, Hearn DW (eds) Network Qptimization. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 346–373
17. Larsson T, Patriksson M, Strömberg A-B (1997) Ergodic convergence in subgradient optimization. Report Dept. Math. Linköping Inst. Techn.
18. Lasdon LS (1970) Optimization theory for large systems. MacMillan, New York
19. Lawphongpanich S, Hearn DW (1984) Simplicial decomposition of the asymmetric traffic assignment problem. Transport Res 18B:123–133
20. Marín A (1995) Restricted simplicial decomposition with side constraints. Networks 26:199–215
21. Meyer GGL (1974) Accelerated Frank–Wolfe algorithms. SIAM J Control 12:655–663
22. Montero L (1991) A simplicial decomposition approach for solving the variational inequality formulation of the general traffic assignment problem for large scale networks. PhD Thesis Dept. d'Estadística i Invest. Oper. Fac. Inform. Univ. Politec. Catalunya, Barcelona
23. Mulvey JM, Zenlos SA, Ahlfeld DP (1990) Simplicial decomposition for convex generalized networks. J Inform Optim Sci 11:359–387
24. Patriksson M (1994) The traffic assignment problem–models and methods. Topics in Transportation. VSP, Utrecht
25. Patriksson M (1998) Nonlinear programming and variational inequality problems–A unified approach. Applied Optim Kluwer, Dordrecht
26. Rookafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton
27. Smith MS (1983) An algorithm for solving asymmetric equilibrium problems with a continuous cost-flow function. Transport Res 17B:365–371
28. Stefek D (1989) Extensions of simplicial decomposition for solving the multicommodity flow problem with bounded arc flows and convex costs. PhD Thesis Univ. Pennsylvania
29. Topkis DM, Veinott AF (1967) On the convergence of some feasible direction algorithms for nonlinear programming. SIAM J Control 5:268–279
30. Ventura SA, Hearn DW (1993) Restricted simplicial decomposition for convex constrained problems. Math Program 59:71–85
31. Von Hohenbalken B (1977) Simplicial decomposition in nonlinear programming algorithms. Math Program 13:49–68

Simplicial Pivoting Algorithms for Integer Programming

HANS VAN MAAREN
Delft University Technol., Delft, Netherlands

MSC2000: 90C10, 90C05

Article Outline

Keywords
Triangulations
Labelings
Integer Labeling
Vector Labeling
Pivoting
Termination and Noncycling Arguments
Max-Closed Sets
Unimodular Max-Closed Form Transformations
See also
References

Keywords

Simplicial algorithms; Integer programming; Knapsack problem; Horn formulas

Simplicial *path following methods* are relatively new in the area of integer programming (cf. ► Integer programming). They are based on a *triangulation* of Euclidean space and a *pivoting algorithm* which, for a restrictive class of problems, terminates with an integral solution or shows that no such solution exists. The path constructed consists of a sequence of neighboring *simplices*, the vertices of which are integral lattice points.

Simplicial methods originated in *fixed point theory*, where they are used to approximate fixed points of continuous mappings [7,18].

In the area of continuous mathematics they have been applied successfully in disciplines such as game theory [19], the approximation of roots of systems of complex polynomials [8,13], economics and econometrics [20] and they have provided a useful machinery to prove a variety of intersection lemmas [10,16].

Returning to discrete problems again, the essentials of the method consist of the following ingredients:

- triangulations
- *labelings*
- pivoting
- termination- and noncycling arguments.

The simplicial algorithms developed so far yield a conclusive answer to the feasibility problem: 'Does a given bounded set in Euclidean space contain a lattice point?' for a restrictive class of sets, the so called *max-closed sets*. In fact, for this class, they provide a polynomial time algorithm (for polyhedral cases) which generalizes earlier work of A. Pnueli [17], F. Glover [11], R. Chandrasekaran [1] and R.W. Cottle and A.F. Veinott [3].

In the sequel we shall pay attention to the notions quoted above. Also, some intrigueing complexity issues which arise when studying *unimodular max-closed form transformations* are discussed briefly. Finally, we discuss the use of simplicial methods outside this tractable class of max-closed sets, namely to locate regions of specific interest, and their possible incorporation into *branching algorithms*.

Triangulations

A *triangulation of Euclidean space* is a set of simplices which union covers the space and moreover satisfies the condition that any two simplices from this set intersect in a member of this set.

For our purposes we need a triangulation which uses *all* lattice points as zero-dimensional elements and triangulates the unit-cube together with all its integral translates. There are various ways to triangulate space in such a manner. An extensive study on triangulations and simplicial methods is [4].

Labelings

The labelings form the most crucial part of the simplicial methods. It is through the labeling device that the original problem is translated to a format where the arguments of the pivoting algorithms adapt to. There are two cases to consider, integer labeling and vector labeling.

Integer Labeling

This part needs some introductory notations and conventions. Given $a \in \mathbf{R}^n$ the following cones play a crucial role:

$$P(a) = \{x \in \mathbb{R}^n : \ x_i \geq a_i \text{ for } i = 1, \ldots, n\},$$
$$P_k(a) = \{x \in \mathbb{R}^n : \ x \in P(a) \text{ and } x_k = a_k\},$$
$$N(a) = \{x \in \mathbb{R}^n : \ x_i \leq a_i \text{ for } i = 1, \ldots, n\},$$
$$N_k(a) = \{x \in \mathbb{R}^n : \ x \in N(a) \text{ and } x_k = a_k\}.$$

Now suppose a set $S \subset \mathbf{R}^n$ is given. A point $a \in \mathbf{R}^n$ *carries label* k if (following [15])

$$S \cap N_k(a) = \emptyset.$$

According to the above, a point may carry more labels (in which case the smallest such is chosen, for example) or it may carry no label at all. Drawing some pictures, one sees that a labeling device as above partitions Euclidean space into n regions, except for points in S itself and except for some part which is located near the so called Pareto boundary of S. The latter part might vanish if S has a specific shape, which shall be precisely the case for the max-closed sets that will be discussed later on.

Of course, it is understood in this context that the set S is accessible to questions whether some point carries label k. In case S is polyhedral, linear programming answers these questions. If S is convex, one needs convex programming to do so. In all cases of interest we assume that membership questions related to S are relatively easy as long they are *not* specified to lattice points of S, that is, as long as they deal with S as a continuous set!

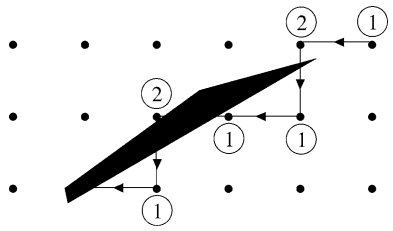
In [5,6] it is shown that if S is a simplex (*knapsack problem*) one can even avoid the use of linear programs in establishing labels. An explicit device is available in that case.

Vector Labeling

Relative to a set $S \subset \mathbf{R}^n$ we can associate to any point $a \in \mathbf{R}^n$ the vector which starts at a and ends at a point of S nearest to a. Here S is assumed to be closed, and preferably convex, in which case this vector is unique. The reader may notice that now all points carry a label! This labeling device was introduced in [15] where it is shown that it can be used to obtain similar results as provided by integer labeling rules but, moreover, enables a pivoting algorithm to continue where it would terminate in the former case due to the lack of labels!

Pivoting

We shall explain the pivoting structure only in the case where integer labelings are used. See [15] for the more complicated process involving vectors. Also, we restrict



Simplicial Pivoting Algorithms for Integer Programming, Figure 1
Pnueli's algorithm

ourselves here to full-dimensional pivots, which lead from a $n+1$-dimensional simplex of the triangulation to a neighboring one. Although *varying dimension pivoting algorithms* [6] are essential in speeding up performance, they involve rather technical details and go beyond the scope of this overview.

Before going into simplicial pivoting we discuss *Pnueli's algorithm* [17] first. Because of introductory purposes we will present a strongly modified form of it.

A set $S \subset \mathbf{R}^n$ is specified, together with a lattice point u, being an *upper bound* for S; this means that $S \subset N(u)$. A *lower bound* for S is any lattice point ℓ for which $N(\ell) \cap S = \emptyset$. Clearly, if S is bounded, it has both upper and lower bounds in the above cone-like sense. In the sequel e_k denotes the kth unit vector. Now the modified form of Pnueli's algorithm (see Fig. 1 for a two-dimensional example) consists of the following iterative procedure:

1 | let $x_0 = u$
2 | if x_j has label k let $x_{j+1} = x_j - e_k$.

It is not at all that hard to see that, *as long as the algorithm runs*, we have all integral lattice points of S contained in $N(x_j)$. This means that the algorithm terminates in three possible states:

1) The iterate x_j is recognized as a lower bound. S is proven to contain no integral solutions.
2) The iterate x_j has no label. If this is because $x_j \in S$ we have located a solution.
3) The iterate x_j has no label and, unfortunately, $x_j \notin S$.

Start A simplex of the triangulation is located with the following property: all labels from $1, \ldots, n$ are present as labels of the $n+1$ vertices of the simplex involved. Such a simplex can easily be found near an upper bound of S (as is done in Fig. 2). In the case of an arbitrary starting, varying dimension algorithm [6] such a simplex shows up during the execution of the algorithm itself!

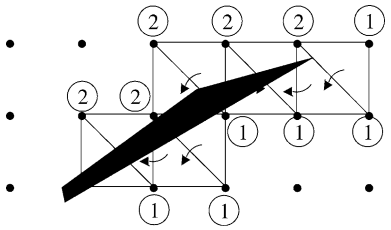
Pivot If a simplex carrying all labels is found, two vertices of this simplex must carry the *same* label. These two vertices define two neighboring simplices of the triangulation: those two full dimensional simplices which intersect the given one in one of the two facets opposed to the two vertices involved. Now one of those simplices is supposed to represent the previous state of the algorithm. The next state is defined as the other simplex. Thus a new lattice point is found, its label is calculated, and when it has one, a new simplex is found carrying all labels, one of which occurs twice at a vertex.

As shall become clear later, 3) cannot happen whenever S is max-closed. Pnueli's algorithm in the above form is extremely simple and it runs (as long as it runs) in polynomial time if S is polyhedral: it uses at most

$$n \sum_{i=1}^{n} (u_i - \ell_i)$$

linear programs to travel from an upper bound u to a lower bound ℓ. A disadvantage however is that it cannot start at an arbitrary point, in the neighborhood of which an integral solution is expected for some reason whatsoever (this remark will turn out to be even more relevant when we will deal with the unimodular transformations later). Simplicial algorithms are much more flexible with respect to starting conditions and this is precisely the reason why they deserve attention as a substitute for Pnueli's algorithm.

We now turn to the pivoting structure (see Fig. 2) (Comparing Fig. 1 and Fig. 2, one might prefer Pnueli's algorithm above the simplicial one; however, note the latter can start everywhere.)



Simplicial Pivoting Algorithms for Integer Programming, Figure 2
A simplicial algorithm

The above construction of a path of neighboring simplices forms the basic idea of each simplicial algorithm. Again we emphasize that in arbitrary starting, variable dimension algorithms the construction is considerably more complicated. However sophisticated a simplicial algorithm may be, the ultimate goal is to create a sequence of almost solutions, thereby carefully avoiding *cycling*.

Termination and Noncycling Arguments

An elegant feature of simplicial algorithms is that, if special care is taken in the construction of the sequence of simplices, cycling is impossible. Moreover, they are designed in such a way that they cannot tend to infinity without passing an upper or lower bound of the set S. This means that, as long as the algorithm runs, it creates new candidate solutions on every iteration. Using these arguments one can prove that

1) The algorithm reaches a recognizable lattice point (which can be an upper bound or a lower bound, depending on the starting position and the state of the algorithm) in which case S is proven not to contain any lattice point. The argument is similar but slightly more involved than in Case 1 to Pnueli's algorithm and was first used in [5]:
   If v is a lattice point of S the set $P_k(v)$ clearly does not contain points carrying label k. Therefore the algorithm cannot pass this set! In order to pass it, it would have needed *all* labels there to be present. As a consequence, the algorithm cannot pass from $P(v)$ to a lower bound or vice versa, without hitting v or another solution. Hence it cannot run between an upper bound and a lower bound without hitting a solution when there is one.

2) The algorithm encounters a point carrying no label. If S is max-closed (see below) this must be a solution.

Max-Closed Sets

Following [15] a set S is called *max-closed* if it satisfies

$$x, y \in S \quad \Rightarrow \quad \max(x, y) \in S$$

where

$$\max(x, y) = (\max(x_1, y_1), \ldots, \max(x_n, y_n)).$$

As one may verify, the following generalities are easily established:

1) Translations map max-closed sets to such sets.
2) Intersections of max-closed sets are max-closed.
3) Inequalities of the form $x_i \leq \alpha_i$ and $x_i \geq \beta_i$ define max-closed sets.
4) An inequality of the form

$$c_1 x_1 + \cdots + c_n x_n \leq c$$

defines a max-closed set whenever *at most one* of the c_i is positive. Special features around these kinds of inequalities in integer programming were already investigated in [1,11,12,17]. Of special interest are the inequalities of this type on two variables: They arise as integer programming reformulations of the *simultaneous Diophantine approximation problem* [14].
5) If f is a monotone increasing function in each of its variables, the set $\{(x, z): f(x) \geq z\}$ is max-closed.
6) Max-closed sets need not be polyhedral, nor need they be convex. They need not even be connected!
7) A function g is called *max-closed* whenever it satisfies

$$g(\max(x, y)) \leq \max(g(x), g(y)).$$

Such functions define max-closed sets by the inequalities $g(x) \leq \gamma$.
8) A bounded max-closed set contains a *unique* coordinatewise maximum point, that is, a u with $S \subset N(u)$.

Based on the above properties it is seen that the following algorithm solves the integer feasibility program whenever S is defined by inequalities of the type mentioned in 4):

1 | $\max_{x \in S} \sum x_i$.
2 | If x solves Step 1 let $u = \lfloor x \rfloor$, the lattice point obtained by rounding the x_i downwards.
3 | Let $S := S \cap N(u)$ and repeat.

The algorithm sketched above can be found in more or less comparable form in [1,11,17]. It only uses

$$\sum_{i=1}^{n} (u_i - \ell_i)$$

linear programs but it is subjected to the *recognition problem*: 'Is a given set S definable by inequalities of the form mentioned in 4)? Or, more generally, is S max-closed?'

Our modified form presented earlier requires more linear programs to run, but it provides a correct answer whenever it has run without halting between an upper and lower bound, without having checked S first on max-closedness. In other words: it avoids the recognition problem.

Max-closed sets have an important analogue in the area of computational logic. There, so called *Horn formulas*, constituting the basics of data-base-reasoning, play an important algorithmic role. The reader familiar with Horn formulas shall certainly recognize property 4) above. Also, Pnueli's algorithm in the form presented above recalls the *single-lookahead-unit-resolution* procedure. See [2,9,21]. V. Chandru and J.N. Hooker, using results of Chandrasekaran established this interesting link.

Resuming the results so far: Pnueli's algorithm, or a simplicial substitute of it, conclusively answers the question whether a max-closed set S contains a lattice point. This is because of the following theorem.

Theorem 1 *If S is max-closed, the only points carrying no label are the points of S itself.*

We emphasize that max-closedness is a sufficient condition for the above algorithms to run in a conclusive manner. But for these algorithms to run it is not a necessary one: they might even occasionally run conclusively (that is, never encountering a non labeled point which is not a solution) whenever S is not max-closed.

Unimodular Max-Closed Form Transformations

Bounded linear integer programming should have turned out to be an easy job in case polyhedral sets would have been automatically max-closed. Unfortunately, this is not the case. An important result, however, is that simplices can be brought into max-closed form through a unimodular transformation [17]. This transformation can be found in a polynomial number of arithmetical operations. This is the more intrigueing since the problem of finding a lattice point in a simplex (knapsack problem) is *NP*-complete. The breakdown of this transformation approach is that after transforming a simplex into max-closed form the number

$$\sum_{i=1}^{n}(u_i - l_i)$$

might grow exponentially with the dimension. For this reason, an arbitrary starting algorithm certainly is to be favored over one which has to start at an upper bound, at least in cases where solutions exist.

Transforming arbitrary polyhedral sets (unimodularly) into max-closed form is not possible. Therefore, simplicial methods clearly are *incomplete methods* when they are applied to arbitrary sets. They return no answer in case a point is encountered which carries no label and is not a solution, before they have entered the upper or lower bound area. As indicated earlier, vector labeling avoids this situation and simplicial methods using this kind of labeling keep running until either a solution is encountered or a specific region of interest is reached: a so called *twinplex* [15].

A twinplex consists of a simplex of the triangulated region which is very specific in the sense that the $n + 1$ associated distance vectors point in all possible directions. To be precise, these $n + 1$ vectors $a_1, \ldots, a_{n+1}$ satisfy

$$\sum_{i=1}^{n+1}\lambda_i a_i = 0$$

for a nontrivial $\lambda \geq 0$. Moreover, these vectors $a_1, \ldots, a_{n+1}$ determine $n+1$ valid inequalities, each of which is violated by at least one of the vertices of the simplex involved. Intuitively, a twinplex is located where the set S is locally the fattest. Based on this intuition it is suggested in [15] to cut S at this place in two or more components, thereby creating a branching algorithm where simplicial methods are incorporated. The various components are transformed then in a manner which make them more likely to be max-closed. Thus a branching tree is designed where tighter max-closed relaxations of the problem are solved with increasing depth.

Yet another possibility, when meeting a nonsolution point a without (integer) label is to go into a *recursion*: If $S \cap N_k(a) \neq \phi$, decrease dimension and find out whether $S \cap N_k(a)$ contains a lattice point. If not, a can be assigned label k after all and the simplicial search continues again in full dimension. In this approach it is suggested to select an index k for the recursion which has a large effect on the non max-closedness of the problem. For example, if S is polyhedral, one selects an index k which appears most often with a nonnegative coefficient. Exploiting twinplexes and incorporating simplicial algorithms in branch and bound search trees is still an area under investigation. In the *satisfiability* area, however, branching algorithms which tend to create Horn-like subproblems with increasing depth have been studied with success.

See also

- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Criss-cross Pivoting Rules
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Graph Coloring
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Least-index Anticycling Rules
- ▶ Lexicographic Pivoting Rules
- ▶ Linear Programming
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Mixed Integer Linear Programming

- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Pivoting Algorithms for Linear Programming Generating Two Paths
- ▶ Principal Pivoting Methods for Linear Complementarity Problems
- ▶ Probabilistic Analysis of Simplex Algorithms
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem

References

1. Chandrasekaran R (1984) Integer programming problems for which a simple rounding type algorithm works. In: Progress in Combinatorial Optimization (Waterloo, Ont., 1982). Acad. Press, New York, pp 101–106
2. Chandru V, Hooker JN (1991) Extended Horn sets in propositional logic. J ACM 38(1):205–221
3. Cottle RW, Veinott AF Jr (1972) Polyhedral sets having a least element. Math Program 3:238–249
4. Dang Chuang Yin (1995) Triangulations and simplicial methods. Lecture Notes Economics and Math Systems, vol 421. Springer, Berlin
5. Dang Chuang Yin, Maaren H van (1998) A simplicial approach to the determination of an integral point of a simplex. Math Oper Res 23(2):403–415
6. Dang Chuang Yin, Maaren H van (1999) An arbitrary starting variable dimension algorithm for computing an integer point of a simplex. Comput Optim Appl 14(1):133–155
7. Eaves CB (1971) Computing Kakutani fixed points. SIAM J Appl Math 21:236–244
8. Forster W (1987) Computing "all" solutions of systems of polynomial equations by simplicial fixed point algorithms: The Computation And Modelling Of Economic Equilibria (Tilburg, 1985). In: Contrib Econom Anal, vol 167. North-Holland, Amsterdam, pp 39–57
9. Franco J (1997) Relative size of certain polynomial time solvable subclasses of satisfiability. In: Satisfiability problem: theory and applications (Piscataway, NJ, 1996). In: DIMACS 35. Amer. Math. Soc., Providence, RI, pp 211–223
10. Freund RM (1984) Variable dimension complexes. II. A unified approach to some combinatorial lemmas in topology. Math Oper Res 9(4):498–509
11. Glover F (1964) A bound escalation method for the solution of integer linear programs. Cahiers CERO 6:131–168
12. Hochbaum DS, Naor J (1994) Simple and fast algorithms for linear and integer programs with two variables per inequality. SIAM J Comput 23(6):1179–1192
13. Kuhn HW (1977) Finding roots of polynomials by pivoting. In: Fixed Points: Algorithms And Applications (Proc. First Internat. Conf., Clemson Univ., Clemson, S.C., 1974). Acad. Press, New York, pp 11–39
14. Lagarias JC (1985) The computational complexity of simultaneous Diophantine approximation problems. SIAM J Comput 14(1):196–209
15. Maaren H van (1998) A simplicial algorithm for a tractable class of integer programs. Techn. Report Delft Univ. Technol. 98-33
16. Meyerson MD, Wright AH (1979) A new and constructive proof of the Borsuk–Ulam theorem. Proc Amer Math Soc 73(1):134–136
17. Pnueli A (1968) A method of truncated relaxation for integer programming. IBM Res. Paper
18. Scarf HE (1967) The approximation of fixed points of a continuous mapping. SIAM J Appl Math 15:1328–1343
19. Scarf HE (1967) The core of an *N* person game. Econometrica 35:50–69
20. Scarf HE (1973) The computation of economic equilibria. Cowles Foundation Monograph, vol 24. Yale Univ. Press, New Haven, CT, With the collaboration of Terje Hansen.
21. Schlipf JS, Annexstein FS, Franco JV, Swaminathan RP (1995) On finding solutions for extended Horn formulas. Inform Process Lett 54(3):133–137

Simulated Annealing

SA

PANOS M. PARDALOS[1], THELMA D. MAVRIDOU[2]

[1] Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

[2] University Florida, Gainesville, USA

MSC2000: 90C90, 90C27

Article Outline

Keywords
See also
References

Keywords

Optimization; Approximation algorithms

For *NP*-hard optimization problems, the use of exact algorithms for the evaluation of the optimal solution is computationally intensive requiring an effort that increases exponentially with the size of the problem. In practice, exact algorithms are used for solv-

ing only moderately sized problem instances. This results in the development of heuristic optimization techniques which provide good quality solutions in a reasonable amount of computational time. One such popular technique is simulated annealing (SA) which has been widely applied in both discrete and continuous optimization problems ([1,4]). SA is a *stochastic search method* modeled according to the physical annealing process which is found in the field of thermodynamics. *Annealing* refers to the process of a thermal system initially melting at high temperature and then cooling slowly by lowering the temperature until it reaches a stable state (*ground state*), in which the system has its lowest energy. S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi [7] initially proposed an effective connection between simulated annealing and combinatorial optimization (cf. ► Evolutionary algorithms in combinatorial optimization), based on the original work in [9].

The main advantage of the SA algorithm is its ability to escape from local optima by using a mechanism which allows deterioration in the objective function value (OFV). That is, in the optimization process SA accepts not only better than previous solutions, but also worse quality solutions controlled probabilistically through the temperature parameter T. More particularly, in the first stages of SA where T is relatively high, the search of the solution space is widely 'explored' so that different solution directions are identified, and often 'bad' solutions are accepted with high probability. During the course of the algorithm the temperature T decreases in order to steadily reduce the probability P of accepting solutions that lead to worse objective function values. With the allowance of controlled 'uphill' movements one can avoid the entrapment to local optima and, eventually, higher quality solutions can be obtained.

There are many factors that have a strong impact on the performance of the SA algorithm:

- The *initial temperature* T_1. A high value of T_1 means that the probability P of accepting inferior solutions is also high, leading to a diversified search in the first iterations of the algorithm. With low values of the initial temperature the search becomes more localized.
- The *thermal equilibrium*. This is the condition in which further improvement in the solution cannot be expected with high probability.
- The *annealing schedule*, which determines in what point of the algorithm and by how much the temperature T is to be reduced.

Now consider a minimization process. Let ΔE denote the change of the OFV between the current state and the state under consideration that occurs as T decreases. This change corresponds to the change in the energy level in the analogy with physical annealing. Then the probability P of accepting a worse quality solution is equal to $e^{-\Delta E/(k_B T)}$, where k_B is the *Boltzmann constant*. Simulated annealing is presented below in pseudocode:

```
PROCEDURE simulated annealing()
    InputInstance();
    Generate randomly an initial solution;
    initialize T;
    DO T > 0
        DO thermal equilibrium not reached →
            Generate a neighbor state randomly;
            evaluate ΔE;
            update current state
            IF ΔE < 0 with new state;
            IF ΔE ≥ 0 with new state with probability e^{-ΔE/(k_B T)};
        OD;
    Decrease T using annealing schedule;
    OD;
    RETURN (solution with the lowest energy)
END simulated annealing:
```

A pseudocode for simulated annealing procedure

The following example (the *quadratic assignment problem*, QAP) illustrates the basic principles of SA in combinatorial optimization. The QAP is defined as follows.

Given a set $N = \{1, \ldots, n\}$ and two $(n \times n)$ matrices $F = (f_{ij})$ and $D = (d_{kl})$, find a permutation p of the set N that minimizes the following function:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{p(i)p(j)}.$$

In the context of location theory one uses the QAP formulation to model the problem of allocating n facilities to n locations with the objective to minimize the cost

associated not only with the distance between locations but also with the flow. F and D correspond to the flow and distance matrices respectively [11].

Let T_i represent the temperature at stage i of the procedure and $T_1 > \ldots > T_f$ represent the annealing schedule. Then the application of SA to the QAP ([2,12]) can be described with the following steps:

- Start with a feasible solution (permutation). Make an exchange between two randomly selected permutation elements (2-exchange). Evaluate the consequent change ΔE.
- While $\Delta E < 0$ repeat the above step. If $\Delta E \geq 0$ then randomly select a variable x from a *uniform distribution* $U(0, 1)$. Accept the pair exchange if $x < \mathrm{P}(\Delta E) = e^{\frac{-\Delta E}{T_i}}$, and repeat the process.
- The system remains at stage i until a fixed number of pair exchanges (equilibrium) has taken place before going to the next stage.
- The procedure stops when all the temperatures in the annealing schedule have been used, i. e. when $i > f$.

Simulated annealing has been used to solve a wide variety of combinatorial optimization problems, such as graph partitioning and graph coloring ([5,6]), VLSI design [7], quadratic assignment problems [2], image processing [3] and many others. In addition, implementations of SA in parallel environments have recently appeared, with applications in cell placement problems, traveling salesman problems, quadratic assignment problems, and others [10]. General references on simulated annealing can be found in [1] and in [8].

See also

- ► Adaptive Simulated Annealing and its Application to Protein Folding
- ► Bayesian Global Optimization
- ► Global Optimization Based on Statistical Models
- ► Monte-Carlo Simulated Annealing in Protein Folding
- ► Packet Annealing
- ► Random Search Methods
- ► Simulated Annealing Methods in Protein Folding
- ► Stochastic Global Optimization: Stopping Rules
- ► Stochastic Global Optimization: Two-phase Methods

References

1. Aarts EHL, Korst JHM (1989) Simulated annealing and Boltzmann machines. Wiley, New York
2. Burkard RE, Rendl F (1984) A thermodynamically motivated simulation procedure for combinatorial optimization problems. Europ J Oper Res 17:169–174
3. Geman S, Geman D (1984) Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Trans Pattern Anal Machine Intell 6:721–741
4. Horst R, Pardalos PM (1995) Handbook of global optimization. Kluwer, Dordrecht
5. Johnson DS, Aragon CR, McGeoch LA, Schevon C (1989) Optimization by simulated annealing: An experimental evaluation. Part I: graph partitioning. Oper Res 37:865–892
6. Johnson DS, Aragon CR, McGeoch LA, Schevon C (1991) Optimization by simulated annealing: An experimental evaluation. Part II: Graph coloring and number partitioning. Oper Res 39:378–395
7. Kirkpatrick S, Gelatt CD, Vecchi MP (1989) Optimization by simulated annealing. Science 220:671–680
8. Laarhoven PJM van, Aarts EHL (1987) Simulated annealing, theory and practice. Kluwer, Dordrecht
9. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:1087–1092
10. Pardalos PM, Pitsoulis LS, Mavridou TD, Resende MGC (1995) Parallel search for combinatorial optimization: Genetic algorithms, simulated annealing, tabu search and GRASP. Lecture Notes Computer Sci, vol 980. Springer, Berlin, pp 317–331
11. Pardalos PM, Wolkowicz H (eds) (1994) Quadratic assignment and related problems. DIMACS. Amer Math Soc, Providence, RI
12. Wilhelm MR, Ward TL (1987) Solving quadratic assignment problems by simulated annealing. IEEE Trans 19:107–119

Simulated Annealing Methods in Protein Folding

IOAN ANDRICIOAEI, JOHN STRAUB
Boston University, Boston, USA

MSC2000: 90C90

Article Outline

Keywords
See also
References

Keywords

Molecular optimization; Protein folding

Proteins are essential molecules for the functioning of biological systems. They are linear polymers, with the monomeric units drawn from a set of 20 amino-acids. The number of amino-acids in a protein ranges from tens to thousands. In their biologically active state, *proteins* assume, out of the immense number of possible configurations, a unique state, the so-called native state. How the protein chain folds into this native state is the essence of the *protein folding* problem. Leaving aside the actual kinetics of the folding process, of great interest and pragmatic value is even the prediction, by any means, of the three-dimensional structure from the knowledge of only the sequence of amino-acids and the potential of interaction between the atoms in the amino-acids.

If one adopts as a working hypothesis that the native state is that which minimizes the protein's potential energy, then the above prediction now reduces, given a potential energy as a function of the positions of the atoms in the protein, to an *optimization* problem. This has been a view which has been advocated by H.A. Scheraga as the ab initio solution of the protein structure prediction problem [7].

How can one estimate the potential energy, which will serve as the objective function in the optimization procedure? Commonly used *empirical potentials* for proteins [5] usually have the form

$$
\begin{aligned}
&U(\mathbf{r}^N)\\
&= \sum_{\text{pairs}} \frac{1}{2} k_b (b - b_0)^2 + \sum_{\text{bond angles}} \frac{1}{2} k_\theta (\theta - \theta_0)^2\\
&\quad + \sum_{\text{dihedral angles}} k_\phi [1 + \cos(n\phi - \delta)]\\
&\quad + \sum_{\text{nonbonded pairs } i,j} 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r} \right)^{12} - \left(\frac{\sigma_{ij}}{r} \right)^{6} \right]\\
&\quad + \sum_{\text{nonbonded pairs } i,j} \frac{q_i q_j}{\epsilon r}.
\end{aligned}
$$

The potential energy $U(\mathbf{r}^N)$ of a protein with N atoms having a certain configuration $\mathbf{r}^N$ in the $3N$-dimensional configuration space is modeled by harmonic bond terms with force constant k_b and bond length b, bond angle terms with force constant k_θ, dihedral angle terms with force constant k_ϕ, multiplicity n and phase δ, van der Waals terms of the Lennard–Jones type with parameters σ_{ij} and ϵ_{ij}, and Coulombic terms, with q_i the charges and ϵ the dielectric constant. The exact value of the parameters will depend on the identity of the participating atoms.

Several related physical problems have been shown to suffer from *NP*-hardness, i. e., to find their optimal solution, a search algorithm requires a simulation time which scales with the size of the system faster than any polynomial function. Among them are the ground state determination of atomic clusters [24] or various spin glass models [4]; demonstrations that the protein folding problem is *NP*-hard have also been given [13].

Since the problem is *NP*-hard, *heuristic algorithms* must be sought for its solution. From the variety of approaches developed (for a review see [20]), we focus on the method of *simulated annealing* [9] as applied to proteins. It has an obvious and physically interpretable application. Once a cooling schedule is chosen, representative configurations of the allowed micro-states are generated by methods either of the *molecular dynamics* (MD) or *Monte-Carlo* (MC) type [6]. For proteins, simulated annealing is traditionally built on an MD approach where the dynamics of the system is simulated by integrating the Newtonian equations of motion and the temperature is controlled through some form of coupling to a heat-bath. If the MC approach is used, after having drawn a new configuration, it is accepted or rejected according to an updating probability of, for example, the Metropolis type [12]

$$
p = \min\left[1, \exp\left(-\beta \Delta V\right)\right], \tag{1}
$$

where $\beta = 1/kT$ and ΔV is the change in potential energy. This acceptance probability has the desirable features that

i) it obeys detailed balance; and
ii) it reduces to a steepest descent minimizer at low temperature (where only moves which decrease the potential energy are accepted).

In addition to the standard Metropolis Monte-Carlo protocol, several other smarter MC algorithms have been designed, based on atomic moves biased by the forces acting upon the atoms in the molecule [16,18].

Substantial improvements to the method of simulated annealing can be obtained by propagating not just one phase point through configuration space, but a whole distribution of points. In the method titled *Gaussian density annealing* [11], the probability den-

sity is approximated by a single Gaussian distribution which is a product of the individual distributions for each atom. This Gaussian distribution is propagated as the temperature decreases according to the Bloch equation, and the dependence of the first two moments of the distribution (i. e., measures of its center position and of its width) is followed as a function of the inverse temperature.

With $\mathbf{r}$ the d-dimensional vector of Gaussian random variables, the multivariate probability density function in the spherically symmetric case has the form

$$\rho(\mathbf{r}, \beta) = (2\pi\sigma^2)^{\frac{-d}{4}} e^{\frac{(\mathbf{r}-\mathbf{r}_0)^2}{4\sigma^2}},$$

where the center of the packet is at $\mathbf{r}_0$, and the second moment is $M_2 = d\sigma^2$. For many degrees of freedom, the many body density distribution can be approximated as a product of single body density distributions, each with its own center and variance.

The differential equations describing the evolution explicitly in the inverse temperature, obtained from the reduced Bloch equation, are found to be

$$\frac{d\mathbf{r}_0}{d\beta} = -\frac{1}{d} M_2 \nabla_{\mathbf{r}_0} \langle U \rangle , \tag{2}$$

$$\frac{dM_2}{d\beta} = -\frac{1}{d^2} M_2^2 \nabla_{\mathbf{r}_0}^2 \langle U \rangle , \tag{3}$$

where $\langle U \rangle$ is the potential energy averaged by weighting with the Gaussian probability density. Note that the center $\mathbf{r}_0$ moves in a steepest descent on the effective potential $\langle U \rangle$ while the width of the distribution adjusts to the curvature of the effective potential.

A variant of the Gaussian density annealing has been applied by C. Tsoo and C.L. Brooks [23], for the study of optimization of water clusters. The popular diffusion equation method of Scheraga and coworkers [10] exists as a special case of the Gaussian density annealing method when all Gaussian packets are characterized by the same variance.

Of particular interest and promise are two related methods. The first is the elegant packet annealing method of D. Shalloway and coworkers [15] of which the Gaussian density annealing method can be shown to be a special case. The second is the locally enhanced sampling (LES) method of R. Elber and coworkers [17]. The LES method has has been applied to complex systems such as solvated peptides with excellent results. It has the additional advantage that it is relatively simple to implement. These and other related methods have recently been reviewed [20].

The *potential smoothing algorithms* based on a Gaussian smoothing transform of the potential energy surface are quite effective for a large number of systems. For a complicated biomolecular potential energy surface it is possible to carry out the smoothing transformation approximately by fitting the nonbonded potential functions to Gaussians or exponentials. However, there is computational overhead associated with computing these transformed functions. It has recently been shown that it is possible to derive all the benefits of the Gaussian smoothing transform while carrying out no explicit transform of the potential energy function. The method substitutes a 'top hat' (impulse) function for the Gaussian in the smoothing transform. In one dimension the result is that the force on the smoothed potential is simply the difference in the potential energy evaluated at each side of the top hat divided by twice the top hat's width – that is, a finite difference formula for the force. Since the width of the smoothing function is not always small, this exact force derived for the smoothed potential can be thought of as a 'bad derivative'. Most significantly, it is possible to use this method to smooth the Boltzmann probability distribution function. This approach is similar in spirit to the packet annealing algorithm of Shalloway and coworkers, which involves a Gaussian smoothing of the Boltzmann distribution [15]. However, the *method of bad derivatives* requires no explicit integral transform. The method was applied to isolate low lying energy minima for a small peptide with excellent results demonstrating the superiority of the method to standard Gaussian smoothing algorithms [3].

Other recent improvements of the simulated annealing method include the use of the *Tsallis probability distributions* [19]. In the Tsallis formalism, a generalized statistics is built from the generalized entropy

$$S_q = k \frac{1 - \sum p_i^q}{q - 1}, \tag{4}$$

where q is a real number and S_q tends to the information entropy

$$S = -k \sum p_i \ln p_i \tag{5}$$

when $q \to 1$. By means of maximizing the Tsallis entropy with the constraints

$$\sum p_i = 1 \quad \text{and} \quad \sum p_i^q \epsilon_i = \text{const},$$

where ϵ_i is the energy spectrum, the generalized probability distribution is found to be

$$p_i = \frac{\left[1 - (1-q)\beta\epsilon_i\right]}{Z_q}, \tag{6}$$

where Z_q is the generalized partition function. This distribution goes to the Gibbs–Boltzmann distribution when q tends to 1. But for $q \neq 1$, the probability distributions have power-law tails, and are thus broader than the Gibbs–Boltzmann distributions. This delocalization of the distribution is the essential feature that allows more ample exploration of the configuration space and faster cooling schedules.

A typical implementation of a generalized simulated annealing algorithm has been proposed by I. Andricioaei and J. Straub [2] (for a variety of MD and MC based sampling algorithms of the Tsallis class, see [21]) and has the following structure:

1) generate trial moves by the method of choice;
2) accept trial moves with probability

$$p = \min\left[1, \left(\frac{1-(1-q(T))\beta E_{\text{new}}}{1-(1-q(T))\beta E_{\text{old}}}\right)^{\frac{q(T)}{1-q(T)}}\right] \tag{7}$$

that obeys detailed balance;
3) reduce the temperature and reduce q such that $\lim_{T\to 0} q(T) = 1$ and go to 1).

At constant temperature, the sampling converges towards the Tsallis equilibrium probability distribution in (6). As temperature decreases, the parameter q is varied as a monotonically decreasing function of temperature. The steepest descent behavior is imposed by starting with a convenient value of q at the initial temperature and by having q tend towards 1 as the temperature decreases during the annealing schedule. Since $q \to 1$ at low T, the desirable reduction to a steepest descent at low temperature is preserved.

Interestingly enough, it was shown [1], that when the maximum entropy formalism is applied to the entropy postulated by Tsallis (4) one is able to recover the more general *Levy probability distributions* (corresponding to fractal random walk, the dimension of which is determined by q), which a variational entropic formalism based on the Boltzmann entropy is unable to do. At initial values of $q(T) > 1$, a Markov chain generated at the initial temperatures would converge towards a Levy distribution. For example, in the particular case of $q = 2$ and a harmonic potential, the Levy distribution is a Cauchy–Lorentz distribution which is the same distribution used for trial moves in the fast simulated annealing method of H. Szu and R. Hartley [22].

Of importance is to study how does the simulated annealing time depend on the features of the potential energy surface of the proteins. One can derive a simple scaling relation for the optimal cooling schedule in a simulated annealing optimization protocol. The relevant energy scales of $U(\mathbf{r}^N)$ are ΔU, the difference in energy between the ground and first excited state minima, and $U^{\ddagger}$, the highest barrier on the potential surface accessed from the global energy minimum. The final temperature reached in a simulated annealing run must be small enough so that at equilibrium the mole fraction in the global energy minimum basin is significant.

The time that the trajectory must spend at the lowest temperature to ensure that the equilibrium distribution is sampled is at least $\tau_{\min}$, the time required to surmount the largest barrier separating the global energy minimum from other thermodynamically important states, which can be shown to go as

$$\tau_{\min} \sim \left(\frac{U^{\ddagger}}{\Delta U}\right)^{\frac{q}{(q-1)}}.$$

In the limit $q \to 1$ of Gibbs–Boltzmann statistics, one finds that

$$\tau_{\min} \sim e^{\frac{U^{\ddagger}}{\Delta U}}.$$

The time for classical simulated annealing increases exponentially as a function of the ratio of the energy scales $U^{\ddagger}/\Delta U$. However, for $q > 1$ the situation is qualitatively different. As a result of the weak temperature dependence in the barrier crossing times, the time for simulated annealing increases only weakly as a power law [21]. Given the large separation in energy scales on the potential energy surface of proteins, and the large value of $U^{\ddagger}/\Delta\ U$, the generalized simulated annealing is expected to be well suited for problems of global optimization as one encounters in protein folding.

The algorithm based on the generalized Tsallis probability distribution has been recently adopted and employed for proteins by U.H.E. Hansmann and Y. Okamoto [8]; see [14] for a review of other generalized ensembles as applied to the protein folding problem.

See also

- Adaptive Simulated Annealing and its Application to Protein Folding
- Bayesian Global Optimization
- Genetic Algorithms
- Genetic Algorithms for Protein Structure Prediction
- Global Optimization Based on Statistical Models
- Global Optimization in Lennard–Jones and Morse Clusters
- Global Optimization in Protein Folding
- Molecular Structure Determination: Convex Global Underestimation
- Monte-Carlo Simulated Annealing in Protein Folding
- Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach
- Packet Annealing
- Phase Problem in X-ray Crystallography: Shake and Bake Approach
- Protein Folding: Generalized-ensemble Algorithms
- Random Search Methods
- Simulated Annealing
- Stochastic Global Optimization: Stopping Rules
- Stochastic Global Optimization: Two-phase Methods

References

1. Alemany PA, Zanette DH (1994) Fractal random walks from a variational formalism for Tsallis entropies. Phys Rev E 49:R956–R958
2. Andricioaei I, Straub JE (1996) Generalized simulated annealing algorithm using Tsallis statistics: Application to conformational optimization of a tetrapeptide. Phys Rev E 53:R3055–R3058
3. Andricioaei I, Straub JE (1998) Global optimization using bad derivatives: A derivative-free method for molecular energy minimization. J Comput Chem 19:1445–1455
4. Bachas CP (1984) Computer-intractability of the frustration model of a spin glass. J Phys A: Math Gen 17:L709–L712
5. Brooks III CL, Karplus M, Montgomery Pettitt B (1988) Proteins: A theoretical perspective of dynamics, structure, and thermodynamics. Wiley, New York
6. Frenkel D, Smit B (1996) Understanding molecular simulation: From algorithms to applications. Acad. Press, New York
7. Gibson KD, Scheraga HA (1988) The multiple-minima problem in protein folding. In: Sarma MH, Sarma RH (eds) Structure and Expression: From Proteins to Ribosomes, vol 1. Adenine Press, Schenectady, NY
8. Hansmann UHE, Okamoto Y (1998) Stochastic dynamics simulations in a new generalized ensemble. Chem Phys Lett 297:374–382
9. Kirkpatrick S, Gelatt CD, Vecchi MP (1989) Optimization by simulated annealing. Science 220:671–680
10. Kostrowicki J, Piela L, Cherayil BJ, Scheraga HA (1991) Performance of the diffusion equation method in searches for optimum structures of clusters of Lennard–Jones atoms. J Phys Chem 95:4113–4119; Kostrowicki J, Scheraga HA (1992) J Phys Chem 96:7442–7449
11. Ma J, Straub JE (1994) Simulated annealing using the classical density distribution. J Chem Phys 101:533–541; ibid (1995) 103:9113–9113
12. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:1087–1092
13. Ngo JT, Marks J, Karplus M (1994) Computational complexity, protein structure prediction, and the Levinthal paradox. In: Merz K, LeGrand S (eds) The protein folding problem and tertiary structure prediction. Birkhäuser, Basel, pp 433–506
14. Okamoto Y (1998) Protein folding problem as studied by new simulation algorithms. Recent Res Developm Pure Appl Chem 2:1–22
15. Orešič M, Shalloway D (1994) Hierarchical characterization of energy landscapes using Gaussian packet states. J Chem Phys 101:9844–9857
16. Pangali C, Rao M, Berne BJ (1978) On a novel Monte Carlo scheme for simulating water and aqueous solutions. Chem Phys Lett 55:413–417
17. Roitberg A, Elber R (1991) Modeling side chains in peptides and proteins: Application of the Locally Enhanced Sampling (LES) and the simulated annealing methods to find minimum energy conformations. J Chem Phys 95:9277–9287
18. Rossky PJ, Doll JD, Friedman HL (1978) Brownian dynamics as smart Monte Carlo simulation. J Chem Phys 69:4628–4633
19. Stariolo DA, Tsallis C (1995) Optimization by simulated annealing: Recent progress. In: Stauffer D (ed) Anual Reviews of Computational Physics II. World Sci., Singapore p 343
20. Straub JE (1996) Optimization techniques with applications to proteins. In: Elber R (ed) New Developments in Theoretical Studies of Proteins. World Sci., Singapore, pp 137–196
21. Straub JE, Andricioaei I (1998) Exploiting Tsallis statistics. In: Deuflhard P, Hermans J, Leimkuhler B, Mark A, Reich S, Skeel RD (eds) Algorithms for Macromolecular Modelling.

Lecture Notes Computational Sci and Engin. Springer, Berlin, pp 189–204
22. Szu H, Hartley R (1987) Fast simulated annealing. Phys Lett A 122:157–162
23. Tsoo C, Brooks III CL (1994) Cluster structure determination using Gaussian density distribution global minimization methods. J Chem Phys 101:6405–6411
24. Wille LT, Vennik J (1985) Computational complexity of the ground-state determination of atomic clusters. J Phys A: Math Gen 18:L419–L422

Simultaneous Estimation and Optimization of Nonlinear Problems

LAURA DI GIACOMO, GIACOMO PATRIZI
Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università di Roma "La Sapienza", Roma, Italy

MSC2000: 93E20, 93E12, 49J15, 62J02, 62M10, 62M20, 91B28

Article Outline

Introduction
Definitions
  Definition and Properties of the Traditional Estimation Approach
  The Simultaneous Estimation and Optimization Approach
Formulation
Methods and Applications
Models
Cases
  Financial Prediction
  Optimal Control in Drilling of Oil
Conclusions
See also
References

Introduction

In applied work, there is often no a priori unique functional form and parametrization for models of empirical processes, nor are there generally accepted ones. For many empirical processes the functional form and the parameters of the system are not known and must be estimated from the data and the optimal control derived [10].

The model of a phenomenon must consider suitable functional forms for the merit function and for the constraints, and determine suitable parameters for these and suitable inputs of the control variables to obtain a specified set of output variables which will render the merit function a minimum (maximum), possibly global. Since the estimation problem is usually posed as an unconstrained optimization problem and determining the optimal control is an optimization problem, two optimization problems must be solved.

Except for simple model categories, interactions occur between the estimation space, where the values of the variables are determined, and the control space, where the optimal control is derived [7], so there is a variational aspect with regard to the optimal control to be analyzed. To avoid severe suboptimization all the unknowns must be determined at the same time, by solving a single more general optimization problem.

To achieve an efficient control of an empirical process, a mathematical programming problem must be solved simultaneously in the estimation and control spaces, to determine a sufficiently accurate estimate of the functional, form and the parameters and the least-cost solution of the optimal control problem.

Statistical conditions have been studied [19], and empirical aspects of the approach have been considered [2,3].

Definitions

To control phenomena of any type [10], including problems of pathological conditions [3], a set of decisions regarding what are the best actions for their control, so as to render optimal the merit function, are considered. Three approaches can be indicated by which the problem could be specified and solved:

1. Ad hoc methods may be used, such as calibration, simulation, experience or intuition, which are limited and may lead to wrong formulations of the problem.
2. The classic three-stage approach: determine the model to implement, estimate the parameters of the form adopted and then solve the quantified optimization problem.
3. Solve the estimation and the optimization problem simultaneously [19].

Whatever the approach, it must be ensured that the estimation of the functional forms and the parameters satisfies the following statistical properties to ensure that the estimates are the best, correct and consistent ones that can be formulated. Statistical estimation methods are important, because, when implemented correctly, with regard to an accurately specified functional form, they will provide estimates of parameters that have the following properties [1,15]:

1. The parameter estimates are unbiased:
   - As the size of the data set grows larger, the estimated parameters tend to their true values.
2. The parameter estimates are consistent, which then will satisfy the following conditions:
   - The estimated parameters are asymptotically unbiased.
   - The variance of the parameter estimate must tend to zero as the data set tends to infinity.
3. The parameter estimates are asymptotically efficient:
   - The estimated parameters are consistent.
   - The estimated parameters have smaller asymptotic variance compared with any other consistent estimator.
4. The residuals have minimum variance, which is ensured by:
   - The variance of the residuals must be a minimum.
   - The residuals must be homoscedastic.
   - The residuals must not be serially correlated.
5. The residuals are unbiased (have zero mean).
6. The residuals have a noninformative distribution (usually, a Gaussian distribution).
   - If the distribution of the residuals is informative, the extra information could somehow be obtained, reducing the variance of the residuals, their bias, etc. with the result that better estimates are obtained.

Through a correct implementation of statistical estimation techniques the estimates are as close as possible to their true values, all the information that is available is applied and the uncertainty surrounding the estimates and the data fit is reduced to the maximum extent possible. Thus, the estimates of the parameters, which satisfy all these conditions, are the "best" possible in a "technical sense" [1].

Definition and Properties of the Traditional Estimation Approach

The estimation and optimization of an empirical model by the traditional three-stage statistical method [20] considers:

- A functional form is posited and an error structure for the residuals is assumed.
- Under these hypotheses a data set is used to determine the values of the parameters, by solving an unconstrained optimization problem and then the estimates determined are checked to verify that they satisfy the properties indicated above [13,1]. If this is so, it is possible to proceed to the next stage, otherwise a new functional form must be specified and a new unconstrained optimization problem must be solved.
- An optimization problem is solved in the space of the control variables, to determine the optimal policy.

With regard to empirical processes, which require the estimation of nonlinear models, there may be many alternative models to represent the behavior of a phenomenon [4]. Therefore the efficacy of the control policy to be adopted will depend on which of the alternative models is chosen and how the parameter values and the functional form are selected [12]. This leads to a number of problems in applications:

- For nonlinear and dynamic estimation problems, the likelihood function to be maximized is usually not concave, so there are many local maxima, each leading to a different set of estimation coefficients. Determining the global maximum of the function may not be helpful.
- Certain statistical properties must be satisfied to ensure a statistically correct estimate. Such conditions may hold at some local maxima, but not at others. There is no assurance that the global maximum, even if unique, will satisfy these conditions; thus, all local maxima must be verified.
- Since there may be many alternative models, the model chosen may not have a different optimal control policy, or the optimization may not yield the best policy for the possible parametrization, while other candidate models could satisfy both criteria.

For estimation problems linear in the parameters, the whole process is simplified. In general, however, the

stages of such a procedure are interrelated and so enumerative solution techniques over all stationary points must be adopted, providing serious limitations to this approach.

The Simultaneous Estimation and Optimization Approach

It would appear more plausible, to avoid suboptimization and to balance the eventual imprecision, to solve both problems simultaneously, imposing the statistical conditions that must be satisfied as constraints [19].

Let the data set of a phenomenon consist of sufficient measurements (y_n,x_n,u_n) over ($n = 1, 2, \ldots, N$) periods, where it is assumed, that $y_n \in R^p$ is a p-dimensional vector to be explained, while $x_n \in R^r$ is an r-dimensional vector of explanatory or state variables of the dynamic process and u_n is a q-dimensional vector of control variables and a horizon $\mathcal{T}$ is indicated, over which the merit function must be optimized and the control variables must be determined. Thus, the whole period considered is composed of a historical period {1,2,… N} and a future period for which policy must be determined, given by $\{N+1, N+2, \ldots, \mathcal{T}\}$. Further, let $w_i \in R^r$ and $v_i \in R^P$ be random variables to be determined with mean zero and finite variance.

It is desired to determine functional forms $\varphi\colon R^{r+q} \to R^r$ and $\eta\colon R^{r+1} \to R^p$ and a set of suitable coefficients $\Theta \in R^m$ such that

$$\text{Min } J = \sum_{i=N+1}^{\mathcal{T}} c(x_i, u_i, y_i)\,, \tag{1}$$

$$x_{i+1} = \varphi(x_i, u_i, y_i, w_i\colon \theta_1) \quad i = 1, 2, \ldots, \mathcal{T}\,, \tag{2}$$

$$y_{i+1} = \eta(x_i, u_i, v_i\colon \theta_2) \quad i = 1, 2, \ldots, \mathcal{T}\,. \tag{3}$$

Equation (1) is the objective function for the model and (2) and (3) are the state space formulation of the problem, while a similar representation may be adopted for the input–output formulation, [14]. To ensure that all the statistical properties that the given estimates of the residuals must fulfill are satisfied at every iteration, instead of solving an unconstrained maximum-likelihood or least-squares problem [13], the required statistical properties of the estimates are set up as constraints, together with the specification of the model of the phenomenon and this global optimization problem is solved for all the undetermined variables.

Formulation

A phenomenon can be represented by a model which will capture all the prescribed input–output relations, at a preset level of precision. Suppose that it is desired to determine an optimal control for the system (1)–(3) over a given period $i =, N+1, \ldots, \mathcal{T}-1$ based on a historical period $i = 1, 2, \ldots, N$ for which a suitable data set is given.

To achieve this, a set of constraints must be added to enforce that the parameter values that will be estimated have the required properties.

The unknowns to be determined are the input and output variables considered and the parameters of the functional form specified in the current iteration, indicated as $\Theta = \{\theta_1, \theta_2\} \subset R^m$. Note that m may be much larger than $2r + q + p + 1$, the number of variables present in each system, since the system is nonlinear.

The mathematical program is formulated with respect to the residual variables, but it is immediate that for a given functional form the unknown parameters will be specified and thus the unknowns of the problem will also be defined and available. Hence, the mathematical program is fully specified for each functional form to be considered, as the residual terms can be stated so:

$$w_i = \hat{x}_{i+1} - \varphi_h(\hat{x}_i, \hat{u}_i, \hat{y}_i\colon \theta_1) \quad i = 1, 2, \ldots, N\,, \quad h \in \mathcal{H} \tag{4}$$

$$v_i = \hat{y}_{i+1} - \eta_k(x_i, u_i, v_i\colon \theta_2) \quad i = 1, 2, \ldots, N\,, \quad k \in \mathcal{K} \tag{5}$$

where $\hat{}$, as usual indicates the historical values of a variable and thus suitable values of θ_1,θ_2 must be determined by the mathematical program, such that all the constraints expressed in terms of $w_i, v_i \forall i$ are specified and $\mathcal{H}, \mathcal{K}$ are suitable function spaces.

The mathematical programming problem, in the notation defined above in "Definitions," to be solved for each functional form in the given sets, with a suitable optimization routine, is the following:

$$\text{Min } J = \sum_{i=N+1}^{\mathcal{T}} c(x_i, u_i, y_i)\,, \tag{6}$$

$$x_{i+1} = \varphi(x_i, u_i, y_i, w_i\colon \theta_1)\,, \tag{7}$$

$$y_{i+1} = \eta(x_i, u_i, v_i : \theta_2) \,, \tag{8}$$

$$\frac{1}{N}\sum_{i=1}^{N} w_i = 0 \,, \tag{9}$$

$$\frac{1}{N}\sum_{i=1}^{N} v_i = 0 \,, \tag{10}$$

$$\frac{1}{N}\sum_{i=1}^{N} w_i^2 \le k_w \,, \tag{11}$$

$$\frac{1}{N}\sum_{i=1}^{N} v_i^2 \le k_v \,, \tag{12}$$

$$-\epsilon_0 \le \frac{1}{N}\sum_{i=1}^{N} v_i w_i \le \epsilon_0 \,, \tag{13}$$

$$-\epsilon_1 \le \frac{1}{N}\sum_{i=1}^{N} w_i w_{i-1} \le \epsilon_1 \,, \tag{14}$$

$$-\epsilon_2 \le \frac{1}{N}\sum_{i=1}^{N} v_i v_{i-1} \le \epsilon_2 \,, \tag{15}$$

$$-\epsilon_3 \le \frac{1}{N}\sum_{i=1}^{N} v_i w_{i-1} \le \epsilon_3 \,, \tag{16}$$

$$-\epsilon_4 \le \frac{1}{N}\sum_{i=1}^{N} w_i v_{i-1} \le \epsilon_4 \,, \tag{17}$$

. . . . . .

$$-\epsilon_{2s} \le \frac{1}{N}\sum_{i=1}^{N} v_{i-s} w_{i-s} \le \epsilon_{2s} \,, \tag{18}$$

$$-\epsilon_{2s+1} \le \frac{1}{N}\sum_{i=1}^{N} w_i w_{i-s} \le \epsilon_{2s+1} \,, \tag{19}$$

$$-\epsilon_{2s+2} \le \frac{1}{N}\sum_{i=1}^{N} v_i v_{i-s} \le \epsilon_{2s+2} \,, \tag{20}$$

$$-\epsilon_{2s+3} \le \frac{1}{N}\sum_{i=1}^{N} v_i w_{i-s} \le \epsilon_{2s+3} \,, \tag{21}$$

$$-\epsilon_{2s+4} \le \frac{1}{N}\sum_{i=1}^{N} w_i v_{i-s} \le \epsilon_{2s+4} \,, \tag{22}$$

$$\frac{1}{2} g_w^T \Psi (\Psi^T \Psi)^{-1} \Psi^T g_w - \frac{N}{2} \le \chi^2_{1-\alpha:\, p-1} \,, \tag{23}$$

$$\frac{1}{2} g_v^T \Psi (\Psi^T \Psi)^{-1} \Psi^T g_v - \frac{N}{2} \le \chi^2_{1-\alpha:\, p-1} \,, \tag{24}$$

$$-\epsilon_{2r+1} \le \frac{1}{N}\sum_{i=1}^{N} w_i^{2r+1} \le \epsilon_{2r+1} r = 3, 4, \ldots \,, \tag{25}$$

$$\frac{1}{N}\sum_{i=1}^{N} w_i^{2r} \le \frac{2r!}{r!2^r}\sigma_w^{2r} r = 3, 4, \ldots \,, \tag{26}$$

$$-\epsilon_{2r+1} \le \frac{1}{N}\sum_{i=1}^{N} v_i^{2r+1} \le \epsilon_{2r+1} r = 3, 4, \ldots \,, \tag{27}$$

$$\frac{1}{N}\sum_{i=1}^{N} v_i^{2r} \le \frac{2r!}{r!2^r}\sigma_v^{2r} r = 3, 4, \ldots \,, \tag{28}$$

$$x_i \in X, y_i \in Y, u_i \in U, w_i \in W, v_i \in V \,. \tag{29}$$

The abstract model of the dynamical system Eqs. (1)–(3) given by the system of Eqs. (7) and (8) is to be optimized with regard to a given merit function Eq. (6) subject to the means of the residuals that should be zero Eqs. (19), (10), and the sum of squares of the residuals should be less than critical values k_w,k_v Eqs. (11), (12), which can be decreased by dichotomous search at every iteration, until the problem does not yield a feasible solution.

The least values obtained for these parameters, while retaining a feasible solution to the whole problem, are equivalent to a minimization of the statistical estimation error and of a maximization of the maximum likelihood, under appropriate distributional assumptions concerning the residuals.

Further, all the serial correlations between the residual must not be significantly different from zero, as enforced by constraints Eqs. (13)–(23).

To ensure that these conditions hold throughout the possible variation of the independent variables, the residuals must be homoscedastic and thus satisfy Eqs. (23) and (24).

The homoscedasticity condition on the residuals is obtained by regressing the original variables of the problem, indicated by the data matrix Ψ, on the normalized square of the residuals, which are indicated, respectively, by g_w,g_v. This leads to a set of nonlinear equations in the squared residuals to be determined. The χ^2 test is applied at a confidence level of $(1-\alpha)$ with $m-1$ degrees of freedom and a significance level of α [6].

Constraints Eqs. (25)–(28) are sample moments of the probability distribution function of the residuals which are made to assume given values in terms of the variance σ^2 and its higher powers. These constraints enforce the residuals to have a noninformative distribution, here a Gaussian.

This constrained minimization problem Eqs. (6)–(29) will dominate the solutions obtainable by the traditional three-phase procedure, since whenever the latter has a solution, the new procedure will also have a solution.

Theorem 1 *Let the given optimal control problem as described in Eqs. (1)–(3) have a unique solution and let all the statistical conditions indicated above hold, so that the solution of the maximum-likelihood estimate of the unconstrained problem is well defined. Then, the solution of optimal control problem Eqs. (1)–(3) will be equal to the solution of the constrained optimization problem Eqs. (6)–(29).*

Methods and Applications

Nonlinear optimization routines which use line search methods may cause difficulties, since some of the constraints may be highly nonlinear, so trust region methods may be better.

A specialized technique based on complementarity theory is used to actually solve this problem.

Consider the following optimization problem which represents in summary form the problem Eqs. (6)–(29).

$$\text{Min } Z = f(w) \quad f\colon R^n \to R\,, \tag{30}$$

$$g(w) \geq 0 \quad g\colon R^n \to R^p\,, \tag{31}$$

$$h(w) = 0 \quad h\colon R^n \to R^q\,. \tag{32}$$

The proposed algorithm consists in defining a quadratic approximation to the objective function, a linear approximation to the constraints and determining a critical point of the approximation by solving a linear complementarity problem, as given in [18].

Expanding the functions in a Taylor series, at the given iteration point w^k, one may eliminate the equality constraints simply by converting them into $p+1$ inequality constraints. Thus,

$$h(w) = h(w^k) + \nabla h(w^k)(w - w^k) \geq 0\,, \tag{33}$$

$$-e_q^T h(w) = -e_q^T (h(w^k) + \nabla h(w^k)(w - w^k) \geq 0\,. \tag{34}$$

Unconstrained variables must be transformed into nonnegative variables for the linear complementarity problem (LCP) algorithm. So let

$$\zeta = \text{Inf}\{w_i \mid w_i \in \{g(w) \geq 0, h(w) = 0\}\} \tag{35}$$

where ζ is a suitable lower bound to the unrestricted variables, which will be expressed as

$$x_i = w_i - \zeta \geq 0\,. \tag{36}$$

Should there be no lower bound specifiable for a variable, then as it is well known, the variable can be represented as the difference of two nonnegative variables.

A set of trust region constraints can be imposed on the problem as a system of linear inequalities centered around the iteration point, to limit the change in the possible solution. Note that such a set of inequalities has properties quite different from those of the usual trust region constraint imposed in nonlinear optimization [8]:

$$Dx + d \geq 0 \tag{37}$$

where $D \in R^{n\times n}$ is a suitable matrix which may be changed at every iteration and $d \in R^n$ a suitable vector. These can be included in the inequalities.

The resulting quadratic problem is easily transformed into a linear complementarity problem, which can be solved by linear programming routines [18]. A new solution point will always exist whenever the trust region is included in the problem, and will lie either inside the trust region or on a trust region constraint.

Whenever this point occurs inside the trust region, it is an approximate stationary point. If the solution point occurs on a trust region constraint and the solution is feasible while a reduction in the objective function has occurred, the solution point is taken as the new starting point and a new iteration is started. Otherwise, if the new point is infeasible, the trust region is reduced. Finally, if there has been an increase in the objective function, the trust region is enlarged and the iteration is repeated, with suitable safeguards to provoke a reduction in the objective function.

If the objective function is bounded for all values of the variables which satisfy the constraints, then a local minimum point will be found. Thus, the convergence of the algorithm can be proved without difficulty by standard techniques [19].

Models

The data sets of empirical processes of phenomena may be of various types, such as dynamic, cross-sectional or both cross-sectional and dynamic, but the model is suitable and accurate for all types of models, replacing the dynamic model indicated in Eqs. (2) and (3) by an appropriate model of the phenomenon, suitable for the data available and the purposes of the application.

In Table 1 three well-known examples are indicated and their salient estimation characteristics are presented. The model for naphthalene arose in an investigation of the oxidation of naphthalene to phthalic anhydride [5,11]. The model fitted was a full quadratic cross-sectional model. As can be seen, the results for the two estimations are identical and all the statistical conditions are satisfied, so confirming Theorem 1.

The model for the tire composition studies the effect of three process variables. The data are very scanty, but the full quadratic model applied indicated that the model as a whole was significant, although the coefficients of the squared terms and the cross-product terms under the null hypothesis are not significant. Heteroscedasticity is found to be significant, by the appropriate test. Our algorithm indicates that lagged variables should be included and the estimates of such a model result have a very low residual.

The model specified for the Constant Elasticity of Substitution (CES) function in economics is a function which is nonlinear in the parameters although it can be almost linearized by a logarithm transformation. Log transformations gave good results for traditional routines, while no transformations were required with this algorithm, which is a definite advantage.

Further computational experiments are given in Table 2 for some important chemical processes and similar results are reported. For example, in the chlorides experiment it is indicated that more variables are required by defining suitable cross products and lagged terms of the original set of variables.

In Table 3 a time series implementation is presented with a number of variants. The experiment is very well known and the original data were analyzed [9] and a polynomial model was fitted. A dynamic model with an exponential term was subsequently fitted [16,17] and autocorrelated terms were added to the series, in the third instance. Traditional techniques provide limited results, whereas this algorithm solves the problem well.

Cases

While in the previous section the application of various models was presented and the results were shown in all cases to be good, here we shall consider dynamic models with exogenous and endogenous variables and examine their performance.

Two general instances will be considered: financial prediction models and optimal control in drilling for oil.

Simultaneous Estimation and Optimization of Nonlinear Problems, Table 1
SAS Institute statistical package and Socrates algorithm: performance comparison of statistical conditions for examples nonlinear in the variables

	Naphthalene		Tire composition		CES function		
	SAS	Socrates	SAS	Socrates	SAS nonlinear	SAS logs	Socrates
No. of observations	80	80	20	20	30	30	30
No. of parameters	10	10	10	10	4	4	10
No. of iterations	–	3	–	6	51	Varying	8
Dynamic	No	No	No	Yes	No	No	Yes
Mean of residuals	0.0	0.0	0.0	0.0	0.0135	0.006	0.09
Residuals variance	6.05	6.05	5.402	1.6E–07	0.1115	0.06	0.1
Heteroscedasticity	NS	NS	S	NS	S	NS	NS
Autocorrelation	NS	NS	NS	NS	U	NS	NS
Lack of normality	NS	NS	S	NS	S	NS	NS

S: significant, NS not significant at confidence level 0.95 U unverifiable

Simultaneous Estimation and Optimization of Nonlinear Problems, Table 2
SAS Institute statistical package and Socrates algorithm: performance comparison of statistical conditions for examples nonlinear in the variables

	Chemical inversion		Isomerization		Chlorides		
	SAS	Socrates	SAS	Socrates	SAS nonlinear	SAS logs	Socrates
No. of observations	38	38	24	24	54	54	54
No. of parameters	2	6	4	4	3	4	10
No. of iterations	4	7	8	4	32	4	8
Dynamic	No	Yes	No	No	No	Yes	Yes
Mean of residuals	0.0	0.0	0.09	0.044	0.0	0.0	0.0
Residuals variance	0.19	0.33	3.23	3.26	1.82	0.98	0.86
Heteroscedasticity	NS	NS	NS	NS	NS	NS	NS
Autocorrelation	S	NS	NS	NS	S	NS	NS
Lack of normality	NS	NS	NS	NS	NS	NS	NS

S: significant, NS not significant at confidence level 0.95 U unverifiable

Simultaneous Estimation and Optimization of Nonlinear Problems, Table 3
SAS Institute statistical package and Socrates algorithm: performance comparison of statistical conditions for dynamic systems

	Exponential model		Polynomial model		Autocorrelated	
	SAS	Socrates	SAS	Socrates	SAS	Socrates
No. of observations	110	110	110	110	110	110
No. of parameters	6	10	5	20	5	10
No. of iterations	3	15	2	31	2	15
Mean of residuals	2.3E-3	4.5E-14	-2.2E-3	9.4E-16	2.2E-2	1.2E-14
Residuals variance	0.404	0.335	0.388	0.241	1.98	1.82
Heteroscedasticity	S	NS	S	NS	S	NS
Autocorrelation	NS	NS	U	NS	NS	NS
Lack of normality	NS	NS	S	NS	NS	NS

S: significant, NS not significant at confidence level 0.95 U undeterminable

Financial Prediction

A number of chronological series of quotations on various stock exchanges were gathered and it was attempted to make predictions one period ahead for the series. The series considered are:

1. The Standard and Poor 500 common stock index (SPX).
2. The Dow Jones Euro Stock Index, consisting of 50 stocks expressed in euros (SX5E).
3. The Nikkei 225 stock average (NKY).
4. US Government bonds 10-year index (USGG10YR).
5. The 10-year fixed US dollar fixed swap rate (USSWAP10).
6. German government bonds 10-year index (GDBR10).
7. Deutsche mark–euro exchange rate (DM-EUR).
8. Gold spot price (GOLDS).
9. Chicago Board Options Exchange: Volatility Index (VIX).
10. Reuters Jeffries CRB Futures Price Index (CRY).

The results of fitting suitable dynamical models to this data, available for over 12 years, are given in Table 4.

Notice how the estimation system (here there is no concurrent optimization system) through the specification of the constraints which impose the statistical properties on the estimates results in very precise predictions one period ahead.

Simultaneous Estimation and Optimization of Nonlinear Problems, Table 4
Variances and residual variances for the single lagged variable dynamic systems, 647 periods

Name of financial series	Series		Residuals		Goodness of fit		
	Mean value	Standard deviation	Mean value	Standard deviation	Absolute mean error	χ^2 value	No. of freedom
SPX	1055.0	270.909	1.1328e-04	0.0836	0.0612	2.8622	580
SX5E	1032.8	3082.0	1.1017e-04	0.4617	0.3057	31.9584	580
NKY	14816.0	3703.1	-0.0137	1.4617	1.0236	62.5824	580
USGG10YR	5.2399	0.9479	-4.7074E-06	2.6441E-04	1.9255E-04	5.58760E-03	580
USSWAP10	5.8226	0.9990	-4.9551E-06	2.7227E-04	2.0133E-04	5.31960E-03	580
GDBR10	4.8036	1.0002	-2.0893E-06	2.3075E-04	1.7287E-04	4.68289E-03	580
DEM-EUR	1.7522	0.2526	-1.1848E-07	5.1837E-05	4.2083E-05	6.33564E-04	580
GOLDS	369.5799	101.1419	-2.4613E-05	0.0237	0.0167	6.53197E-01	580
VIX	19.6402	6.7262	6.3632E-05	0.0093	0.0070	2.00086	580
CRY	245.4010	41.9699	-1.0608E-05	0.0085	0.0068	0.12247	580

Results have also been obtained for predictions two periods ahead and up to five periods ahead with comparable results. Thus, this algorithm is extremely robust.

Optimal Control in Drilling of Oil

Determining optimal control policies for petroleum drilling is a very interesting problem, since through the mudlogging data bases which are compiled for every well, the underlying empirical process can be studied in detail.

In Table 5 the actual time series of the drilling process is compared with the best predictions obtained from an endogenous model for each process. The best model indicated a differing number of endogenous variables, but for all 16 processes the model with five lags was the most accurate.

The determination of optimal control policies in processes for the extraction of oil from underground sources requires that they be formulated as formal procedures. In each process during 1 week, the original mudlogging data set was resampled every 30 elementary periods, which were of 5 s, so the fundamental period considered was 150 s.

The period over which to determine the optimal controls was chosen to be 192 periods, or 8 h, while the historical period was relatively long. The results are shown in Table 6, in which the optimal increment determined on the basis of this algorithm with a closed-loop policy is compared with what actually happened.

In Table 6 the six instances determining optimal controls are indicated and each entry is concerned with the drilling experience of the given well for that week with regard to the given period. From the active perforation intervals an initial period was selected randomly and the optimal control was defined for the next 192 periods (8 h). The average predicted increment over the actual increment was more than 30%.

Conclusions

Simultaneous estimation and optimization provides very robust and versatile algorithms for nonlinear estimation and optimization of all types of empirical data sets. Here a more complex unique optimization problem was used instead of a succession of optimization problems, the first being unconstrained and the second being a constrained decision or allocation optimization problem.

This approach always imposes the satisfaction of the required statistical properties, so as to ensure that at every iteration a statistically correct solution is determined for that functional form. By iterating on the functional forms and on the specification of the problem, one will always obtain better solutions, until a lower bound is reached.

The lower bound indicates the pure noise of the empirical process, but as can be seen it is usually much lower than the noise components determined by traditional methods.

Simultaneous Estimation and Optimization of Nonlinear Problems, Table 5
Best endogenous prediction results of five lag models for the drilling processes

Process	No. of exogenous variables	Residual variances	χ^2	No. of degrees of freedom	Time (s)
AX01	4	84002408	782709.488774	2660	470.89
AXY02	4	0.11496404E-02	0.002329	3236	982.77
AZC03	8	0.41461787E-01	0.311272	5069	5073.40
BL01D	4	0.10572470E-01	0.012292	3664	1056.66
BE01	6	0.27926209	5.930955	3080	1839.50
CP03	8	0.12768791E-01	0.057533	2744	2069.71
CC09	4	0.76346211E-03	0.000425	3230	575.76
DY02	6	0.30245159E-02	0.014091	3092	1518.98
FT02D	6	0.19508212E-03	0.000238	3848	10601.91
GX01	6	0.63263313E-03	0.001045	3611	8576.97
MAR08	8	0.88143523E-02	0.007543	1909	1278.91
PW01	6	0.58546597E-01	0.132707	2105	658.40
P01	6	0.55842518	1.509173	2105	478.83
RGR68O	6	0.99603892E-03	0.002172	2981	987.23
RGR69O	8	0.27631189E-02	0.005723	2756	1254.82
RED01	6	0.21829931E-02	0.001563	3122	908.94
VF03	4	0.84573218E-03	0.000488	3213	531.59

Simultaneous Estimation and Optimization of Nonlinear Problems, Table 6
Optimal predicted versus actual increment for six oil wells over 192 periods (8 h)

Well and week	Optimal increment	Real increment	Difference (%)
FT02D 9	114.0	73.3	35.70
FT02D 16	116.7	83.8	28.19
FT02D 23	73.7	13.45	81.75
GX01 3	94.8	72.2	23.84
GX01 11	57.9	18.8	67.53
BE01 1	181.4	160.25	11.66

Thus, the simultaneous estimation and optimization algorithm for nonlinear and dynamic problems is an extremely powerful instrument.

See also

- ► Complementarity Algorithms in Pattern Recognition
- ► Generalizations of Interior Point Methods for the Linear Complementarity Problem
- ► Mathematical Programming Methods in Supply Chain Management

References

1. Amemiya T (1985) Advanced Econometrics. Blackwell, Oxford
2. Arcangeli G, Benassi M, Nieddu L, Passi C, Patrizi G, Russo MT (2002) Optimal Adaptive control of treatment planning in radiation therapy. Eur J Oper Res 140:399–412
3. Bartolozzi F, De Gaetano A, Di Lena E, Marino S, Nieddu L , Patrizi G (2000) Operational research techniques in medical treatment and diagnosis: A review. Eur J Oper Res 121:435–466
4. Box GE, Jenkins GM (1970) Time Series Analysis: forecasting and control. Holden-Day, San Francisco
5. Box GEP, Draper NR (1987) Empirical Model-Building and Response Surfaces. Wiley, New York
6. Breusch TS, Pagan AR (1979) A simple test for heteroscedasticity and random coefficient variation. Econometrica 47:1287–1294
7. Bunke H, Bunke O (1986) Statistical Inference in Linear Models. Wiley, New York
8. Conn AR, Gould NIM, Toint PL (2000) Trust-Region Methods. MPS-SIAM, Philadelphia
9. Cox DR (1977) Discussion of papers by campbell and walker, tong and morris. J Royal Stat Soc A140:453–454
10. Di Giacomo L, Patrizi G (2006) Dynamic Nonlinear Modelization of Operational Supply Chain Systems. J Glob Optim 34:503–534
11. Franklin NL, Pinchbeck PH, Popper F (1956) A statistical approach to catalyst development: the effect of process vari-

ables in the oxidation phase of naphthalene, part i. Trans Inst Chem Eng 34:280–293
12. Gallant AR (1987) Nonlinear Statistical Models. Wiley, New York
13. Jennrich RI (1969) Asymptotic properties of non-linear least squares estimators. Ann Math Stat 40:633–643
14. Kalman RE, Falb PL, Arbib MA (1969) Topics in Mathematical System Theory. McGraw-Hill, New York
15. Malinvaud E (1978) Méthodes Statistiques de l'économétrie, 3eme edn. Dunod, Paris
16. Moran PAP (1953) The statistical analysis of the Canadian lynnx cycle, i: Structure and prediction. Austr J Zoology 1:163–173
17. Osaki T (1982) The statistical analysis of the perturbed limit cycle process sinf nonlinear time series models. J Time Ser Anal 3:29–41
18. Patrizi G (1991) The equivalence of an lcp to a parametric linear program with a scalar parameter. Eur J Oper Res 51:367–386
19. Patrizi G (2000) **S.O.C.R.A.t.E.S.** simultaneous **o**ptimal **c**ontrol by **r**ecursive and **a**dap**t**ive **e**stimation system: Problem formulation and computational results. In: Lassonde M Optimization and Approximation, Vth International Conference on Approximation and Optimization in the Carribean. Springer, Berlin, pp 245–254
20. Seber GAF, Wild CJ (1989) Nonlinear Regression. Wiley, New York

Single Facility Location: Circle Covering Problem

Sylvester's Problem

DONALD W. HEARN
University Florida, Gainesville, USA

MSC2000: 90B85, 90C27

Article Outline

Keywords
See also
References

Keywords

Minimum circle; Minimum sphere; Sylvester's problem

In 1857 the British mathematician J.J. Sylvester, in a one sentence article [28], posed the problem:

> It is required to find the *least circle* which shall contain a given set of points in the plane.

From then until the 1960s, the problem attracted the occasional interest of mathematicians who proposed algorithms [1,5,21], applications [21,29] and related theory [17,26], both for the problem in the plane and for the *minimum sphere problem* in higher dimensions. The references, especially [1,14,26], cover the history in more detail.

Starting in 1971, the problem attracted greater interest in the context of location theory because finding the center of the circle of minimum radius is equivalent to locating a central facility for which the maximum distance to any service point is minimized [8,9,11,12,16,23]. The problem was also introduced in the 1970s computer science literature as one of the closest point problems of computational geometry [27]. This article provides an optimization formulation, characterization of the solution, one of the primary algorithms for the problem in the plane and references to extensions.

A minimax statement of the minimum sphere problem in $\mathbf{R}^n$ is:

$$\min_{x\in\mathbb{R}^n} \max_{i=1,\ldots,m} \|x - a_i\| ,$$

where there are m given points $a_i \in \mathbf{R}^n$ and $\|\cdot\|$ denotes the Euclidean norm. Converting this to a constrained optimization problem with differentiable functions is accomplished by squaring the norm term and introducing a new variable s for the squared radius of the minimum sphere:

$$\begin{cases} \min\limits_{(s,x)} & s \\ \text{s.t.} & \|x - a_i\|^2 \le s, \quad i = 1,\ldots,m, \\ & (s,x) \in \mathbb{R}^{n+1}. \end{cases}$$

In this form the problem is a convex program for which the *Karush–Kuhn–Tucker conditions* [20] are both necessary and sufficient. Applying these conditions shows that there exist nonnegative multipliers u_i, $i = 1,\ldots,m$ such that

$$\sum_{i=1}^{m} u_i a_i = x^*,$$

$$\sum_{i=1}^{m} u_i = 1,$$

$$u_i(s^* - \|x - a_i\|^2) = 0, \quad i = 1,\ldots,m,$$

$$u_i \ge 0, \quad i = 1,\ldots,m.$$

Thus x^*, the center of the minimum sphere with radius $\sqrt{s^*}$, is a convex combination of points on its boundary since, for points within the sphere, the associated u_i must be zero. Further, by *Carathéodory's theorem* (cf. ► Carathéodory theorem), at most $n + 1$ of the given points are required for this convex combination. In the plane, therefore, the minimum circle is defined by either

- two points of maximum separation, or
- three of the points which form a nonobtuse triangle.

This characterization of the solution is also evident from geometrical arguments.

Given this characterization, a finite procedure could be devised in which all circles defined by two or three points are constructed. In this procedure, three points would be discarded if they formed an obtuse triangle. From the circles constructed, the smallest covering circle would be chosen. However, this total enumerative approach would not be effective for large values of m – the number of two and three point combinations grows rapidly with the number of points. If, for example, m = 100, over 160,000 combinations would be considered.

Below is an outline of a more efficient method due to J. Elzinga and D.W. Hearn [8]:

0.	Choose any two of the given points and go to Step 1.
1.	Let these two points define the diameter of a circle. IF this circle covers all points THEN STOP ELSE find an outside point and go to Step 2 with these three points.
2.	Solve the minimum circle problem for these three points. IF the minimum circle is defined by two points THEN go to Step 1. IF three points define it THEN go to Step 3.
3.	IF the circle defined by three points covers all points THEN STOP ELSE find an outside point (e.g., the farthest one) and go to Step 4.
4.	Solve the minimum circle problem for these four points. IF it is defined by two points THEN go to Step 1. IF defined by three points THEN go to Step 3.

Elzinga–Hearn algorithm

This outline omits many details of how the different steps are executed and efficiencies such as reducing the given set of m points to their *convex hull*. See [2,8,14] for a discussion of those details and [15] for a computer code. In computational testing on random data, the most effective versions compute the solution of a problem with 100 points in fewer than 10 iterations [14]. Empirically, the computational effort goes up linearly with the number of points, however, an example given in [6] requires $O(m^2)$ time. In theory, there are methods of lower complexity, e. g., methods based on construction of *Voronoi diagrams* [25,27] and the $O(m)$ method of N. Megiddo [22], but effective implementations of these methods have not been developed. E. Welzl [30] gives a random method of *expected* complexity $O(m)$ and reports on the implementation of an effective heuristic variation.

The *Elzinga–Hearn algorithm* can be classified as a *dual* procedure [14] – only the final circle covers all points and is therefore feasible. Primal methods, in particular, the first algorithm, due to G. Chrystal [5] and, independently, to B. Peirce [29] also exist, and the most efficient implementations are competitive with that of Elzinga and Hearn. In [14] these implementations are described and a classification scheme is given which shows the equivalence of a number of other proposed algorithms.

In the location theory context positive weights indicating relative importance may be associated with the given service points [11,14]. Then the problem is one of minimizing the *maximum weighted distance* and it loses its covering circle interpretation [14]. The algorithm given above has been extended to this weighted problem in [13,14].

By a change of variables of the form $s = v + x^\top x$, the above constrained problem becomes

$$\begin{cases} \min\limits_{(v,x)} & v + x^\top x \\ \text{s.t.} & v + 2a_i^\top x - a_i^\top a_i \geq 0, \\ & \quad i = 1, \ldots, m, \\ & (v, x) \in \mathbb{R}^{n+1}, \end{cases}$$

which is recognized to be a *convex quadratic program* [9,19,23]. Thus there are many general purpose algorithms which can solve the minimum sphere problem. A pivoting method reminiscent of the revised simplex

method for linear programming, and with storage requirements that depend only on n, is given in [9].

Extensions of the basic problem include covering a convex polyhedron defined by algebraic inequalities [10,18] and *minimax location on a sphere* or hemisphere using great circle distances [7,24]. See also the $O(m \log m)$ method of G. Xue and S. Sun [31].

See also

- Combinatorial Optimization Algorithms in Resource Allocation Problems
- Competitive Facility Location
- Facility Location with Externalities
- Facility Location Problems with Spatial Interaction
- Facility Location with Staircase Costs
- Global Optimization in Weber's Problem with Attraction and Repulsion
- MINLP: Application in Facility Location-allocation
- Multifacility and Restricted Location Problems
- Network Location: Covering Problems
- Optimizing Facility Location with Rectilinear Distances
- Production-distribution System Design Problem
- Resource Allocation for Epidemic Control
- Single Facility Location: Multi-objective Euclidean Distance Location
- Single Facility Location: Multi-objective Rectilinear Distance Location
- Stochastic Transportation and Location Problems
- Voronoi Diagrams in Facility Location
- Warehouse Location Problem

References

1. Blumenthal LM, Wahlin GE (1941) On the spherical surface of smallest radius enclosing a bounded subset of n-dimensional Euclidean space. Amer Math Soc Bull 47:771–777
2. Boffey TB, Karkazis J (1983) Speeding up the Elzinga-Hearn algorithm for finding 1-centres. J Oper Res Soc 34:1119–1121
3. Chakraborty RK, Chaudhuri PK (1981) A note on geometrical solutions for some minimax location problems. Trans Sci 15:164–166
4. Charalambous C (1981) An iterative algorithm for the minimax multifacility location problem with Euclidean distance. Naval Res Logist Quart 28:325–337
5. Chrystal G (1885) On the problem to construct the minimum circle enclosing n given points in the plane. Proc Edinburgh Math Soc 3:30–33
6. Drezner Z, Shelah G (1987) On the complexity of the Elzinga–Hearn algorithm for the 1-center problem. Math Oper Res 12:255–261
7. Drezner Z, Wesolowsky GO (1983) Minimax and maximin facility location problems on a sphere. Naval Res Logist Quart 30:305–312
8. Elzinga J, Hearn DW (1972) Geometrical solutions for some minimax location problems. Trans Sci 6:379–394
9. Elzinga J, Hearn DW (1972) The minimum covering sphere problem. Managem Sci 19:96–104
10. Elzinga J, Hearn DW (1974) The minimum sphere covering a convex polyhedron. Naval Res Logist Quart 21:715–718
11. Francis RL, McGinnis LF Jr, White JA (1992) Facility and location: An analytical approach, 2nd edn. Prentice-Hall, Englewood Cliffs, NJ
12. Hearn DW (1971) Minimum covering spheres. PhD Thesis Johns Hopkins Univ.,).
13. Hearn DW, Vijay J (1981) A geometrical solution for the weighted minimum circle problem. Res. Report ISE Dept. Univ. Florida 81(2)
14. Hearn DW, Vijay J (1982) Efficient algorithms for the (weighted) minimum circle problem. Oper Res 30:777–795
15. Hearn DW, Vijay J (1995) Codes of geometrical algorithms for the (weighted) minimum circle problem. Europ J Oper Res 80:236–237
16. Jacobsen SK (1981) An algorithm for the minimax Weber problem. Europ J Oper Res 6:144–148
17. John F (1948) Extremum problems with inequalities as subsidiary conditions. Courant Anniv. Vol. Interscience 187–204
18. Konno H, Yajima Y, Ban A (1994) Calculating a minimal sphere containing a polytope defined by a system of linear inequalities. Comput Optim Appl 3:181–191
19. Kuhn HW (1975) Nonlinear programming: A historical view. SIAM-AMS Proc 9:1–26
20. Kuhn HW, Tucker AW (1950) Nonlinear programming. In: Neyman J (ed) Proc. Second Berkeley Symp. Math. Statistics and Probability. Univ. Calif. Press, Berkeley, CA, pp 481–492
21. Lawson CL (1965) The smallest covering cone or sphere. SIAM Rev 7:415–417
22. Megiddo N (1983) Linear time algorithms for linear programming in R3 and related problems. SIAM J Comput 12:759–776
23. Nair KPK, Chandrasekaran R (1971) Optimal location of a single service center of certain types. Naval Res Logist Quart 18:503–510
24. Patel MH (1995) Spherical minimax location problem using the Euclidean norm: Formulation and optimization. Comput Optim Appl 4:79–90
25. Preparata FP, Shamos. MI (1985) Computational geometry: An introduction. Springer, Berlin
26. Rademacher H, Toeplitz O (1957) The enjoyment of mathematics. Princeton Univ. Press, Princeton. Transl. from: Von Zahlen and Figuren. Springer, 1933

27. Shamos MI, Hoey D (1975) Closest point problems. In: Proc. 16th IEEE Annual Symp., Found. of Comput. Sci., Berkeley, pp 151–162
28. Sylvester JJ (1857) A question in the geometry of situation. Quart J Pure Appl Math 1:79
29. Sylvester JJ (1860) On Poncelet's approximate linear valuation of surd froms. Philosophical Mag (Fourth Ser) 20:203–222
30. Welzl E (1991) Smallest enclosing disks (balls and ellipsoids). Lecture Notes Computer Sci, vol 555. Springer, Berlin, pp 359–370
31. Xue G, Sun S (1995) The spherical one-center problem. In: Du D-Z and Pardalos PM (eds) Minimax and Applications. Kluwer, Dordrecht, pp 153–156

Single Facility Location: Multi-objective Euclidean Distance Location

MARIANTHI IERAPETRITOU
Department Chemical and Biochemical Engineering, Rutgers University, Piscataway, USA

MSC2000: 90B85

Article Outline

Keywords
Mathematical Model
Solution Approach
Duality
Discrete Location Problem
Objectives
See also
References

Keywords

Facility location; Optimization

The problem of *single facility location* can be stated as follows: Determine the location of a single new facility with respect to a number of existing facilities that minimizes an appropriate defined total cost function which is chosen to be proportional to distance. Typical examples are the location of a new:

a) machine in a manufacturing facility;
b) warehouse relative to production;
c) pump in chemical operations;
d) well in an oil field development.

A generalization of this problem involves the *multifacility location-allocation* problem, [5]. A mathematical formulation of the single-facility problem is as follows: m existing facilities are located at known distinct points $P_1, \ldots, P_m$, a new facility is to be located at a point X, costs of 'transportation' nature are incurred and are directly proportional to an appropriately defined distance between the new facility and the existing ones. Let $d(X, P_i)$ represent the distance between points X and P_i and let w_i represent the cost of transportation between the new facility and existing facility i at P_i. Then the total 'transportation' cost between the new facility and the existing facilities is given by:

$$f(X) = \sum_{i=1}^{m} w_i d(X, P_i),$$

where terms w_i are referred to as 'weights'. The single facility location problem is to determine the location of a new facility, say X^*, that minimizes $f(X)$. In many applications the cost per unit distance is a constant thus the minimization problem often reduces to a determination of the location that minimizes distance. The l_p norm is a popular distance measure in facility location theory. If the coordinates for the new facility are x_1, x_2, so that $X = (x_1, x_2)$, and for the existing facility i the coordinates are a_i, b_i, so that $P_i = (a_i, b_i)$, the l_p norm is as follows:

$$l_p = ((x_1 - a_i)^p + (x_2 - b_i)^p)^{\frac{1}{p}}, \quad p \geq 1.$$

Typically in the literature it is assumed that $p = 1$ or 2, for which we obtain the rectangular and Euclidean norms, respectively. Examples of facility location problems where Euclidean distance applies are the network location problems as well as the pipeline design problems. Examples of facility location problems where *rectilinear distance* applies are machine location problems where transportation occurs along a set of aisles arranged in a rectangular pattern as well as the well location problem in an oil field where the no flow boundaries are located in the midpoint between the extraction wells, [6].

Mathematical Model

The Euclidean distance location problem can be stated mathematically as:

$$\min f(X) = \sum_{i=1}^{m} w_i((x_1 - a_i)^2 + (x_2 - b_i)^2)^{\frac{1}{2}}. \quad (1)$$

It is variously referred to as the *Steiner–Weber problem* or the *general Fermat problem*. The first important property of the problem is that since $w_i l_2(X, a_i)$ is a convex function of X and the sum of convex functions is a convex function as well, it follows that $f(X)$ is a convex function. This means that local minima are global optima of problem (1), [8]. With this information we are assured that the following *extremal equations* for $f(X)$ can produce only global optima of problem (1):

$$\frac{\partial f(X)}{\partial x_k} = \sum_{i=1}^{m} \frac{w_i((x_k - a_i k)}{l_2(X, a_i)} = 0, \quad k = 1, 2. \quad (2)$$

One difficulty that the above equation has is that the derivatives are undefined if $l_2(X, a_i) = 0$. Therefore, if the optimal location for the new facility coincides with that of an existing facility, equation (2) cannot be used to check optimality. However, each existing facility can be easily checked for optimality by utilizing the following property, [8]: $f(X)$ is minimized at the rth existing facility location (a_{r1}, a_{r2}), if and only if

$$CR_r^2 = \left(\sum_{i=1 \neq r}^{m} \frac{w_i(a_{r1} - a_{i1})}{l_2(a_r, a_i)} \right)^2 + \left(\sum_{i=1, \neq r}^{m} \frac{w_i(a_{r2} - a_{i2})}{l_2(a_r, a_i)} \right)^2 \leq w_r. \quad (3)$$

Solution Approach

An iterative procedure has been proposed for the solution of problem (1) which is known as *Weiszfeld procedure*. This iterative procedure is based on the following equation, which can be obtained from (2) so as to get x_k:

$$x_k = \frac{\sum_{i=1}^{m} \frac{w_i a_{i1}}{l_2(X, a_i)}}{\sum_{i=1}^{m} \frac{w_i}{l_2(X, a_i)}}, \quad k = 1, 2. \quad (4)$$

Note that x_k is also involved in the right-hand side of (4) so that an iterative scheme should be employed to solve (4). The following form holds at iteration $l + 1$:

$$x_k^{l+1} = \frac{\sum_{i=1}^{m} \frac{w_i a_{i1}}{l_2(X, a_i)}}{\sum_{i=1}^{m} \frac{w_i}{l_2(X^l, a_i)}}, \quad k = 1, 2. \quad (5)$$

A good initial point comes from the solution of the *squared Euclidean problem*, that is the same as the Euclidean location problem except that each distance $l_2(X, a_i)$ is squared. The solution to this problem was found to be the *center of gravity location* given by:

$$x_k^0 = \frac{\sum_{i=1}^{m} w_i a_{i1}}{\sum_{i=1}^{m} w_i}, \quad k = 1, 2. \quad (6)$$

The iterative procedure is guaranteed to converge to the optimum location. Discussion about the convergence is given in [7].

Duality

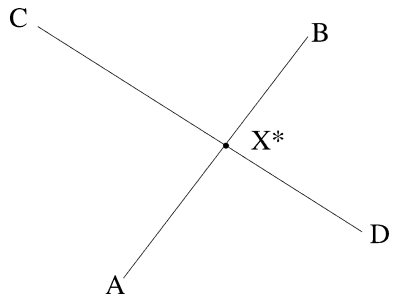
A strict convex approximation function to eliminate the problem of discontinuities of the derivatives of $f(X)$ is the following:

$$\min f(X) = \sum_{i=1}^{m} w_i((x_1 - a_i)^2 + (x_2 - b_i)^2 + \epsilon^2)^{(\frac{1}{2})}. \quad (7)$$

Based on this problem one can derive, [8], the dual limit as $\epsilon \to 0$:

$$\begin{cases} \max\limits_{U,Z} & D(U) = -\sum\limits_{i=1}^{m}(a_{i1}u_i + a_{i2}v_i) \\ \text{s.t.} & \sum\limits_{i=1}^{m} u_i = 0, \\ & \sum\limits_{i=1}^{m} v_i = 0, \\ & u_i^2 + v_i^2 \leq w_i, \quad i = 1, \ldots, m. \end{cases} \quad (8)$$

The inequality constraints can be equivalently posed as $u_i^2 + v_i^2 \leq w_i^2$ to produce a differentiable problem that can be solved using standard nonlinear programming algorithm. The optimal *Lagrange multipliers* of problem (5) solve the original problem (1), [8]. It is of interest to notice the geometric interpretation of optimal facility location. Figure 1 gives the case of four existing facilities at points A, B, C and D. For the case of equal weights suppose that the four points can be arranged in pairs (A, B) and (C, D) so that the straight



Single Facility Location: Multi-objective Euclidean Distance Location, Figure 1
Graphical solution for the Euclidean distance problem

lines AB and CD intersect. The intersection point X^* is the optimum location for the new facility. However, if a point, for example D, lies within the triangle ABC, then the optimal location of new facility coincides with the location D.

Discrete Location Problem

In the *discrete location problem*, [4], the new facility is to be located and a single choice must be made from among a finite number of sites, say n. The mathematical model for this *assignment problem* is the following:

$$\begin{cases} \min \quad f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,\ldots,n, \\ \qquad \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,\ldots,n, \\ \qquad x_{ij} = 0 \text{ or } 1, \quad \forall i,j. \end{cases} \tag{9}$$

As an example of the use of the *assignment model* in a context similar to continuous location problem, suppose that there are p existing facilities and d_{kj} represents the Euclidean distance between existing facility k and site j and that w_{ik} represents a total cost per unit distance incurred in transporting items between a new facility i and existing facility k. Then, for a given assignment, $w_{ik}\sum_{j=1}^{n} d_{kj}x_{ij}$ represents the total cost incurred transporting items between new facility j and existing facility k. The total cost of transportation of all items is then given by:

$$\sum_{i=1}^{n}\sum_{k=1}^{p} w_{ik}\sum_{j=1}^{n} d_{kj}x_{ij} = \sum_{j=1}^{n} c_{ij}x_{ij}, \tag{10}$$

where $c_{ij} = \sum_{k=1}^{p} w_{ik}d_{kj}$. With the above transformation the location problem can then be solved as an assignment problem, [4].

Objectives

In the previous sections the objective considered for the location problem was the most commonly used minimization of cost that can be translated to minimization of distance since the cost has been considered directly proportional to distance. However, there are many other objectives used in the literature concerning the location problem. A complete review could be found in [3]. One can distinguish three different objective categories:

- the *push objectives*,
- the *pull objectives* and
- the *balancing objectives*.

The pull objectives are based on the assumption that the facility to be located is desirable and so the distance from the 'costumers' has to be minimized. In this category belong the objectives mentioned before as well as objectives of profit maximization where there is a price associated with each demand cite. In the push objectives the facility is undesirable as for example a dangerous facility due to leak possibility, and so is located to maximize the distance from the 'costumer' cites. The balancing objectives try to minimize the weighted (balance) distance between the new facility and the costumers. If we consider the distribution of all facility-costumer distances for any given solution, push and pull objectives optimize some function of the mean i. e., the first central moment of this distribution. In contrast most balancing objectives target on minimizing the variability of the distribution of distances i. e., the second moment. A number of criteria exist to evaluate the selection of the balancing objectives, [9]. Among them are the *scale invariance criterion*, according to which the optimal facility location remains the same irrespective of the type of measure applied to the problem; the *principle of transfers*, that states that the distri-

bution will become suboptimal if a facility from above the average distance is transfered to the facility that is below the average. Other criteria include *analytical tractability*, *normalization of measures*, *appropriateness*, *sensitivity* and *Pareto optimality*. For the latter one considers a single facility location problem where the three points are almost collinear. The distances to the three points are equal if the points lie on the circumference of a circle with its center at the facility. However, the closer the points are to the line the larger the radious of a circle is. Customers in all three locations will benefit from getting the facility closer to them until it reaches the central point. The above case is an example where the equality objective alone is not meaningful. Based on the problems associated with balancing objectives a number of authors, [1], have considered the trade-offs between equality and efficiency in the objective function. Concluding the facility location problem is usually multi-objective in nature. Location under any of the aforementioned objectives satisfies but a single objective. Specialized solution approaches can be used to guide the optimization when more than one objective is considered, [2].

See also

- Combinatorial Optimization Algorithms in Resource Allocation Problems
- Competitive Facility Location
- Facility Location with Externalities
- Facility Location Problems with Spatial Interaction
- Facility Location with Staircase Costs
- Global Optimization in Weber's Problem with Attraction and Repulsion
- MINLP: Application in Facility Location-allocation
- Multifacility and Restricted Location Problems
- Network Location: Covering Problems
- Optimizing Facility Location with Rectilinear Distances
- Production-distribution System Design Problem
- Resource Allocation for Epidemic Control
- Single Facility Location: Circle Covering Problem
- Single Facility Location: Multi-objective Rectilinear Distance Location
- Stochastic Transportation and Location Problems
- Voronoi Diagrams in Facility Location
- Warehouse Location Problem

References

1. Bodily SE (1978) Police sector design incorporating preferences of interest groups for equality and efficiency. Managem Sci 24:1301–1313
2. Clark PA, Westerberg AW (1983) Optimization for design problems having more than one objective. Comput Chem Eng 7:259–278
3. Drezner Z (1995) Facility location: A survey of applications and methods. Springer, Berlin
4. Francis RL, White JA (1974) Facility layout and location. Prentice-Hall, Englewood Cliffs, NJ
5. Ierapetritou MG (2000) MINLP: Application in facility location-allocation. In: Encycl. Global Optim. Kluwer, Dordrecht
6. Ierapetritou MG (2000) Single facility location: Multiobjective Euclidean location. In: Encycl. Global Optim. Kluwer, Dordrecht
7. Kuhn HW (1973) A note on Fermat's problem. Math Program 4:98–107
8. Love RF, Morris JG, Wesolowsky GO (1988) Facilities location: Models & methods. North-Holland, Amsterdam
9. Marsh M, Schilling D (1994) Equity measurement in facility location analysis: A review and framework. Europ J Oper Res 74:1–17

Single Facility Location: Multi-objective Rectilinear Distance Location

MARIANTHI IERAPETRITOU
Department Chemical and Biochemical Engineering, Rutgers University, Piscataway, USA

MSC2000: 90B85

Article Outline

Keywords
Mathematical Model
Uncertainty
Constraints
Dynamic Location
Objectives
See also
References

Keywords

Facility location; Optimization

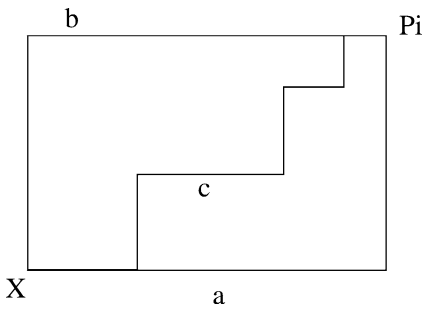
The problem of *single facility location* can be stated as follows: Determine the location of a single new facility with respect to a number of existing facilities that minimizes an appropriate defined total cost function which is chosen to be proportional to distance. A mathematical formulation of the single facility problem is as follows: m existing facilities are located at known distinct points $P_1, \ldots, P_m$, a new facility is to be located at a point X, costs of 'transportation' nature are incurred that are directly proportional to an appropriately defined distance between the new facility and the existing ones. Let $d(X, P_i)$ represent the distance between points X and P_i and w_i represent the cost of transportation between the new facility and existing facility i at P_i, then the total 'transportation' cost between the new facility and the existing facilities is given by:

$$f(X) = \sum_{i=1}^{m} w_i d(X, P_i),$$

where the terms w_i are referred to as 'weights'. The single facility location problems is to determine the location of a new facility, say X^*, that minimizes $f(X)$. In many applications the cost per unit distance is a constant thus the minimization problem often reduces to a determination of the location that minimizes distance. The appropriate determined distance can be a straight line (i. e. *Euclidean distance*), [7], or a rectilinear distance. Examples of facility location problems where Euclidean distance applies are the network location problems as well as the pipeline design problems. If the coordinates for the new facility are x, y so that $X = (x, y)$ and the coordinates for the existing facility i are a_i, b_i so that $P_i = (a_i, b_i)$, the rectilinear distance between X and P_i is defined as:

$$d(X, P_i) = |x - a_i| + |y - b_i|\,.$$

Examples of facility location problems where *rectilinear distance* applies are machine location problems where transportation occurs along a set of aisles arranged in a rectangular pattern as well as the well location problem in an oil field where the no flow boundaries are located in the midpoint between the extraction wells. A distinct difference between the rectilinear and Euclidean distance is illustrated in Fig. 1, which illustrates that there are several different paths between X and P_i for which the rectilinear distance is the same. The number of such paths is, of course, infinite. This is not the case with Euclidean distance where the path is unique.



Single Facility Location: Multi-objective Rectilinear Distance Location, Figure 1
Different rectilinear paths between *X* and P_i

Mathematical Model

The rectilinear distance location problem can be stated mathematically as:

$$\min f(x, y) = \sum_{i=1}^{m} w_i(|x - a_i| + |y - b_i|), \qquad (1)$$

which is equivalently stated as:

$$\min_x \sum_{i=1}^{m} w_i(|x - a_i|) + \min_y \sum_{i=1}^{m} w_i(|y - b_i|),$$

where each quantity can be treated as separate optimization problem [4]:

$$\min_x \sum_{i=1}^{m} w_i(|x - a_i|),$$

$$\min_y \sum_{i=1}^{m} w_i(|y - b_i|).$$

Some properties if the optimum solution are:

a) The x-coordinate of the new facility will be the same as the x-coordinate of some existing facility. Similarly, the y-coordinate of the new facility will coincide with the y-coordinate of some existing facility. It is not necessary however, that both coordinates be for the same existing facility.
b) The optimum x-coordinate (y-coordinate) location for the new facility is a *median location*.

A different way to addressing the same problem is proposed in [10], where the problem is formulated as a linear programming model. Let us consider the absolute distance between the new and the existing facility i. This is given by:

$$\begin{aligned} d_{i1} &= a_i - x \quad \text{if } x \le a_i, \\ d_{i2} &= x - a_i \quad \text{if } x \ge a_i, \\ d_{i1}, d_{i2} &\ge 0, \quad i = 1, \ldots, m. \end{aligned}$$

Using these expressions:

$$\begin{aligned} & x + d_{i1} - d_{i2} = a_i, \\ & d_{i1} \times d_{i2} = 0, \\ & d_{i1}, d_{i2} \ge 0, \quad i = 1, \ldots, m, \end{aligned}$$

and the location problem (1) become a linear programming problem:

$$\begin{cases} \min & \sum_{i=1}^{m} w_i (d_{i1} + d_{i2}) \\ \text{s.t.} & x + d_{i1} - d_{i2} = a_i, \\ & d_{i1}, d_{i2} \ge 0, \quad i = 1, \ldots, m. \end{cases} \tag{2}$$

Note that the condition $d_{i1} \times d_{i2} = 0$, which is not included in the above formulation, is always satisfied at the optimal solution of problem (2). For example, let the current values of d_{i1}, d_{i2} be given by d'_{i1}, d'_{i2}, where $d'_{i1} > d'_{i2}$ and $d'_{i2} \neq 0$. Then a solution which reduces the value of the objective function is given by $d_{i1} = d'_{i1} - d'_{i2}$ and $d_{i2} = 0$. Problem (2) represents a linear programming problem with $2m + 1$ variables and m nontrivial constraints. The dual of problem (2) is:

$$\min \sum_{i=1}^{m} a_i s_i - s_{m+1} \sum_{i=1}^{m} a_i .$$

By defining a new variable $z_i = s_i - s_{m+1} + \max_i [w_i]$, $i = 1, \ldots, m$, the dual becomes:

$$\begin{cases} \min & \sum_{i=1}^{m} (z_i - \max_i [w_i]) a_i \\ \text{s.t.} & \sum_{i=1}^{m} (z_i - \max_i [w_i]) \ge 0, \\ & \max_i [w_i] - w_i \le z_i \le w_i + \max_i [w_i], \\ & i = 1, \ldots, m. \end{cases} \tag{3}$$

Problem (3) has all its constraints but one of the bounded-variable type and can therefore solved efficiently by linear programming techniques.

See [8], for a proof that the function $W_k(x_k) = \sum_{i=1}^{m} w_i(|x_k - a_{jk}|)$, where $k = 1, 2$ for the two-dimensional problems, is a convex function and for the following optimality conditions: at some t^*,

$$\sum_{i=1}^{t-1} w_i^k - \sum_{i=t}^{n_k} w_i^k < 0, \tag{4}$$

$$\sum_{i=1}^{t} w_i^k - \sum_{i=t+1}^{n_k} w_i^k \ge 0 \tag{5}$$

are satisfied. If condition (5) is met as a strict inequality, then $x_k^* = a_{t^* k}$. If condition (5) is met as an equality, then $x_k^* \in [a_{t^* k}, a_{t^*+1 k}]$. Based on the above conditions, [8] propose an iterative procedure to determine t^*.

Uncertainty

Uncertainty may appear in the destinations, which are also called *regional demand*, and the weights. Regional demand is modeled by a continuous spatial distribution of one or more destinations. The location objective then corresponds to minimizing the expected value of the distance of the facility to the random destinations. The analytical evaluation of the integral type of the objective is only possible in the simple cases of rectangular or circular regions that however lead to objectives that are not easy to optimize, [1]. Let us consider the single facility location problem with rectangular distances. Let each of the m existing facilities have random coordinates (Y_{j1}, Y_{j2}), described by bivariate normal probability density function $f(y_{j1}, y_{j2})$. We want to find a facility location (x_1^*, x_2^*) that minimizes the expected sum of weighted rectangular distances. The total expected cost is:

$$\mathsf{E}W(X) = \mathsf{E}_Y \left(\sum_{j=1}^{m} w_j d(X, Y_j) \right), \tag{6}$$

where $Y_j = (Y_{j1}, Y_{j2})$ and $Y = (Y_{11}, Y_{12}, \ldots, Y_{n1}, Y_{n2})$. It follows that:

$$\begin{aligned} \mathsf{E}W(X) &= \sum_{j=1}^{m} w_j \mathsf{E}_Y d(X, Y_j)) \\ &= \sum_{j=1}^{m} w_j \mathsf{E}_{Y_j} |x_1 - Y_{j1}| + \sum_{j=1}^{m} w_j \mathsf{E}_{Y_j} |x_2 - Y_{j2}| \end{aligned}$$

$$= \sum_{j=1}^{m} w_j \mathsf{E}_{Y_{j1}} |x_1 - Y_{j1}| + \sum_{j=1}^{m} w_j \mathsf{E}_{Y_{j2}} |x_2 - Y_{j2}|$$
$$= \mathsf{E}W_1(x_1) + \mathsf{E}W_1(x_2),$$

which means that the location problem becomes:

$$\min_X \mathsf{E}W(X) = \min_{x_1} \mathsf{E}W_1(x_1) + \min_{x_2} \mathsf{E}W_1(x_2). \quad (7)$$

Using the density function f we can evaluate each one of these terms:

$$\mathsf{E}W_1(x_1) = \sum_{j=1}^{m} w_j \int_{-\infty}^{\infty} |x_1 - y_{j1}| f(y_{j1})\, dy_{j1} \quad (8)$$

Let Y_{j1} have mean μ_{j1} and standard deviation σ_{j1}. Then:

$$f(y_{j1}) = \frac{1}{(2\pi)\sigma_{j1}} \exp\left(-\frac{1}{2}\left(\frac{y_{j1} - \mu_{j1}}{\sigma_{j1}}\right)^2\right). \quad (9)$$

As the derivative of $\mathsf{E}W_1(x_1)$ is easily evaluated as follows:

$$\mathsf{E}W_1'(x_1) = \sum_{j=1}^{m} w_j \left(1 - 2\mathsf{P}_r\left(z \geq \frac{x_1 - \mu_{j1}}{\sigma_{j1}}\right)\right), \quad (10)$$

we may use a method such as interval bisection to find x_1^*.

Another approach to overcome the regional demand is to replace each region by a representative point, a centroid, and solve the resulting classical location problem, [2]. Although this approach is relative simple it raises questions regarding the involved aggregation error due to the arbitrary way of selecting the used centroids. In case of uncertainty in the weights the most typical question is to determine all the points which may be optimal for any choice of weights. Assuming a distribution to be known for each one of them H.C.L.W. Williams [11] derived the probability for each one of the destinations of being optimal. Also the degree of nonoptimality of a given site is an important information that can be extracted from the uncertainty analysis. This type of information is important when considering a possible relocation decision of a facility, [5].

Constraints

Many practical applications need methods able to handle feasible regions of any shape even disconnected ones. See [6], for a special method for location problems with l_p-distance within a finite union of convex polygons; this method was subsequently extended in [9] to general objectives and arbitrary polygonal shaped feasible regions.

Dynamic Location

Let us assume that a facility is expected to serve over r periods of time during which may be repeatedly relocated. The problem of *dynamic facility location* is to find the single facility location but for each of the r periods. Let the weights w_{jk} be the present value of the cost per unit distance between the new facility and existing facility j in period k, and let c_k be the present value of the cost. The objective is then to find a series of locations $X_k = (x_{1k}, x_{2k})$, $k = 1, \ldots, r$, that minimize the present value of the location plan. The dynamic location problem to be considered is [8]:

$$\min \sum_{k=1}^{r} \sum_{j=1}^{m} w_{jk} d(X_k, a_j) + \sum_{k=2}^{r} c_k z_k, \quad (11)$$

where $z_k = 1$ if $X_k \neq X_{k-1}$ is allowed and $z_k = 0$ otherwise. The variables z_k serve as indicators if the facility is permitted to move from where it was the previous period.

Objectives

The problem of facility location is usually a multiobjective problem since more than one objectives have to be optimized simultaneously. The models described above result in the location solution where only the objective of minimizing a distance function is being considered. However, a number of alternative objectives have been considered in the literature that can be classified in the *pull objectives*, *push objectives* and *balancing objectives*, [7]. The first category involve the constraints that minimize the distance to the new facility assuming that it is a desirable unit whereas, the push objectives maximize the corresponding distances assuming the location of undesirable unit. The balancing objectives try to weight the distances from the new facility to the existing ones. An alternative to the objective of minimizing the distance is the *maxmin objective*, [8], that has

the following form:

$$\begin{cases} \max & f(x) \\ \text{s.t.} & f(x) = \min w_j l_1(X, a_j), \quad j = 1, \ldots, m. \end{cases}$$

In order to incorporate more than one objective in the facility location problem, *multi-objective optimization* techniques should be applied, [3]. The basic idea of these techniques is the systematic generation of the *Pareto optimal solution* set that involves the points in which one objective can be improved only at the expense of other objectives.

See also

- ▶ Combinatorial Optimization Algorithms in Resource Allocation Problems
- ▶ Competitive Facility Location
- ▶ Facility Location with Externalities
- ▶ Facility Location Problems with Spatial Interaction
- ▶ Facility Location with Staircase Costs
- ▶ Global Optimization in Weber's Problem with Attraction and Repulsion
- ▶ MINLP: Application in Facility Location-allocation
- ▶ Multifacility and Restricted Location Problems
- ▶ Network Location: Covering Problems
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Production-distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Stochastic Transportation and Location Problems
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Aly AA, Marucheck AS (1982) Generalized Weber problem with rectangular regions. J Oper Res Soc 33:983–989
2. Bennett CD, Mirakhor A (1974) Optimal facility location with respect to several regions. J Reg Sci 14:131–136
3. Clark PA, Westerberg AW (1983) Optimization for design problems having more than one objective. Comput Chem Eng 7:259–278
4. Francis RL, White JA (1974) Facility layout and location. Prentice-Hall, Englewood Cliffs, NJ
5. Hall RW (1988) Median, mean and optimum as facility locations. J Reg Sci 28:65–82
6. Hansen P, Peeters D, Thisse J-F (1982) An algorithm for a constrained Weber problem. Managem Sci 28:1285–1295
7. Ierapetritou MG (2000) Single facility location: Multiobjective Euclidean location. In: Encycl. Global Optim. Kluwer, Dordrecht
8. Love RF, Morris JG, Wesolowsky GO (1988) Facilities location: Models & methods. North-Holland, Amsterdam
9. Plastria F (1992) GBSSS: The generalized big square small square method for planar single facility location. Europ J Oper Res 62:163–174
10. Wesolowsky GO, Love RF (1971) The optimal location of new facilities using rectangular distances. Oper Res 18:124–130
11. Williams HCWL (1977) On the formation of travel demand models and economic evaluation measures of user benefit. Environm Plan 9:285–344

Smoothing Methods for Semi-Infinite Optimization

Hubertus T. Jongen[1], Oliver Stein[2]

[1] Department of Mathematics, RWTH Aachen University, Aachen, Germany

[2] School of Economics and Business Engineering, University of Karlsruhe, Karlsruhe, Germany

MSC2000: 90C34, 90C31, 57R12

Article Outline

Introduction
- Smoothing Approaches Motivated by Nonlinear Programming
- Smoothing Approaches for Semi-Infinite Programs

Definitions
- The Extended Mangasarian–Fromovitz Constraint Qualification
- The Reduction Ansatz and Nondegenerate KKT Points
- Mollifiers

Formulation

Conclusions

See also

References

Introduction

Optimization problems for a finite-dimensional decision variable under infinitely many inequality constraints are called *semi-infinite* (see [8,19] for reviews).

For smoothing methods one considers them in the form

$$\text{SIP:} \quad \min_{x\in\mathbb{R}^n} f(x) \text{ subject to } g(x,y) \geq 0 \text{ for all } y \in Y,$$

with the objective function $f \in C^2(\mathbb{R}^n, \mathbb{R})$, the constraint function $g \in C^2(\mathbb{R}^n \times \mathbb{R}^m, \mathbb{R})$, and a nonempty and compact index set $Y \subset \mathbb{R}^m$. Moreover, Y is assumed to be described by finitely many inequality constraints,

$$Y = \{y \in \mathbb{R}^m \mid v(y) \geq 0\},$$

with $v \in C^2(\mathbb{R}^m, \mathbb{R}^s)$ and $s \in \mathbb{N}$. The feasible set of SIP is denoted by

$$M = \{x \in \mathbb{R}^n \mid g(x,y) \geq 0 \text{ for all } y \in Y\}.$$

A basic problem in semi-infinite optimization is to check whether a point $x \in \mathbb{R}^n$ is feasible, since this involves the verification of infinitely many inequality constraints.

The semi-infinite constraint in SIP is equivalent to

$$G(x) := \min_{y\in Y} g(x,y) \geq 0,$$

which means that the feasible set M is the upper level set of the, in general, nonsmooth function G. Smoothing methods try to replace G by a smooth function, but to keep important properties of the original problem under this modification. More explicitly, one wishes that *under weak assumptions* a nonempty and compact feasible set M can be approximated arbitrarily well by a level set of a *single smooth function* with certain *regularity properties*. Moreover, there should be a *correspondence between Karush–Kuhn–Tucker points* of the original and of the smoothed problem, along with their *Morse indices*.

Smoothing Approaches Motivated by Nonlinear Programming

A smoothing procedure for *finite* optimization problems is given in [12]. There the main idea is to use the logarithmic barrier approach to approximate finitely many inequality constraints $g_i(x) \geq 0$, $i \in I$, $|I| < \infty$, by one smooth and nondegenerate constraint $\sum_{i\in I} \ln(g_i(x)) \geq \ln(\varepsilon)$ for $\varepsilon > 0$. A similar approach is taken in [5] to smooth finite maximum functions. However, obvious generalizations of this approach to semi-infinite programming are not successful.

In fact, there are two standard arguments which connect SIP to finite optimization problems. First, a sufficiently fine discretization of the index set leads to an arbitrarily accurate outer approximation of M by finitely many inequality constraints which could, in a next step, be smoothed by the logarithmic barrier approach. Unfortunately, the so-called second-order shift terms of semi-infinite programs are ignored by the discretized problem, so correspondences of Morse indices cannot even be established between the original and the discretized problem, let alone the smoothed discretized problem.

Second, assuming the so-called reduction ansatz at some point $\bar{x} \in M$, the feasible set can locally be described by finitely many smooth inequality constraints. The logarithmic barrier approach for this locally reduced SIP is used in [10]. While Morse indices are modeled well in this approach, the assumption of the reduction ansatz in the whole feasible set is not generic [16].

Another obvious generalization of the approach from finite programming is to directly use the barrier term $\int_Y \ln(g(x,y))\,\mathrm{d}y$ for SIP. For infinite quadratic programming problems a related interior point approach is presented in [18]. For nonlinear SIP, however, this logarithmic barrier term is neither self-concordant nor does it necessarily enforce interior points, as an example from [10] shows. The main problem is that in some situations even the singularity of the logarithm is smoothed by the integral, and boundary points can become feasible for the approximation (note that $\int \ln(y) = y\ln(y) - y$ can be continuously extended to $y = 0$ with value 0).

Smoothing Approaches for Semi-Infinite Programs

An approximation of the feasible set in semi-infinite optimization by a quadratic distance function is presented in [6]. While smoothness of the approximating problem is shown, it is inherently degenerate, so no results on Morse indices can be expected from this approach.

A smoothing method for semi-infinite programs adhering to *all* the above criteria is given in [14,15]. There the function G is smoothed by mollification, as is explained in the remainder of this contribution.

Definitions

The auxiliary function G is the optimal value function of the so-called *lower level problem*:

$$Q(x): \quad \min_{y \in \mathbb{R}^m} g(x, y) \text{ subject to } v(y) \geq 0 .$$

Points x from the topological boundary ∂M of M satisfy $G(x) = 0$, and the corresponding globally minimal points of $Q(x)$ are denoted by

$$Y_0(x) = \{y \in Y | g(x, y) = 0\} .$$

The set $Y_0(x)$ is also called the *active index set* of x for SIP.

The Extended Mangasarian–Fromovitz Constraint Qualification

A nice topological structure of M at its boundary points can be guaranteed under constraint qualifications. Since G is directionally differentiable [1], according to [21] the natural extension of the well-known and basic Mangasarian–Fromovitz constraint qualification [17] at a zero $\bar{x}$ of G is

$$\{d \in \mathbb{R}^n | \, G'(\bar{x}, d) > 0\} \neq \emptyset .$$

With the formula for the directional derivative from [1], one obtains the following explicit condition which is well known for semi-infinite programs [9,16].

Definition 1 At $\bar{x} \in M$ the *extended Mangasarian–Fromovitz constraint qualification (EMFCQ)* is said to hold if there exists some vector $d \in \mathbb{R}^n$ with

$$D_x g(\bar{x}, y) d > 0 \text{ for all } y \in Y_0(\bar{x}) .$$

Here $D_x g$ denotes the row vector of partial derivatives of g with respect to x. In [16] it is shown that EMFCQ holds generically in semi-infinite programming and is, thus, a weak assumption.

The Reduction Ansatz and Nondegenerate KKT Points

For theoretical as well as numerical purposes it is of crucial importance to keep track of the elements of the active index set $Y_0(x)$ for varying x. Recall that each $y \in Y_0(x)$ is a global minimizer of $Q(x)$. The *reduction ansatz* [7,22] is said to hold at $\bar{x} \in M$ if all global minimizers of $Q(\bar{x})$ are *nondegenerate* in the sense of Jongen et al. [11]. Since nondegenerate minimizers are isolated, and Y is a compact set, the closed set $Y_0(\bar{x})$ can only contain finitely many points, say, $Y_0(\bar{x}) = \{\bar{y}^1, \ldots, \bar{y}^p\}$ with $p \in \mathbb{N}$. By a result from [3] the local variation of these points with x can be described by the implicit function theorem.

In fact, for x locally around $\bar{x}$ there exist continuously differentiable functions $y^i(x)$, $1 \leq i \leq p$, with $y^i(\bar{x}) = \bar{y}^i$ such that $y^i(x)$ is the locally unique local minimizer of $Q(x)$ around $\bar{y}^i$. It turns out that the functions $G_i(x) := g(x, y^i(x))$ are even C^2 in a neighborhood of $\bar{x}$.

A major consequence of the reduction ansatz is the so-called reduction lemma: if the reduction ansatz holds at $\bar{x}$, then for all x from a neighborhood U of $\bar{x}$ we have

$$G(x) = \min_{1 \leq i \leq p} G_i(x) .$$

This means that M can locally be described by finitely many C^2 constraints, that is, SIP is locally equivalent to the smooth finite optimization problem:

$$\text{SIP}_{\text{red}}: \quad \min_{x \in \mathbb{R}^n} f(x) \text{ subject to } G_i(x) \geq 0, \quad i = 1, \ldots, p .$$

Examples show that the reduction ansatz cannot be expected to hold everywhere in the feasible set of a generic semi-infinite program [16]. As nondegeneracy in the sense of Jongen et al. is a local property, one can, however, define a nondegenerate KKT point of the SIP via the locally reduced problem SIP_{red}. Let

$$L(x, \lambda) = f(x) - \sum_{i=1}^{p} \lambda_i G_i(x)$$

denote the Lagrangian of SIP_{red} with multiplier vector $\lambda \in \mathbb{R}^p$.

Definition 2 A point $\bar{x} \in M$ is called *a nondegenerate Karush–Kuhn–Tucker point of SIP* if the reduction ansatz holds at $\bar{x}$ and if $\bar{x}$ is a nondegenerate Karush–Kuhn–Tucker point of SIP_{red}, that is, the following three conditions hold:

1. The linear independence constraint qualification holds at $\bar{x}$, and there exists a (unique) multiplier vector $\bar{\lambda} \geq 0$ with $D_x L(\bar{x}, \bar{\lambda}) = 0$.

2. The multipliers satisfy $\bar{\lambda}_i > 0,\ i = 1, \ldots, p$.
3. The matrix $D_x^2 L(\bar{x}, \bar{\lambda})|_{T_{\bar{x}} M_{\mathrm{red}}}$, that is, the Hessian of the Lagrangian, restricted to the tangent space to M_{red} at $\bar{x}$, is nonsingular.

The number of negative eigenvalues of the matrix in condition 3 is called the *Morse index* of $\bar{x}$.

For generic SIP all Karush–Kuhn–Tucker points are nondegenerate [20,24]. In this sense, nondegeneracy of KKT points for SIP is a weak assumption.

Mollifiers

With the Euclidean norm $||\cdot||_2$ on $\mathbb{R}^n$ the *standard mollifier* [2] is the C^∞ function:

$$\eta(x) = \begin{cases} C \exp\left(\frac{1}{||x||_2^2 - 1}\right), & ||x||_2 < 1\,, \\ 0, & ||x||_2 \geq 1\,, \end{cases}$$

where $C > 0$ is chosen such that $\int_{\mathbb{R}^n} \eta(x)\,\mathrm{d}x = 1$. For $\varepsilon > 0$ put

$$\eta_\varepsilon(x) = \frac{1}{\varepsilon^n}\,\eta\left(\frac{x}{\varepsilon}\right)\,.$$

The function η_ε is also C^∞, it satisfies $\int_{\mathbb{R}^n} \eta_\varepsilon(x)\,\mathrm{d}x = 1$, and its support is the closed ball $\overline{B(0,\varepsilon)}$ with $B(0,\varepsilon) = \{x \in \mathbb{R}^n \,|\, ||x||_2 < \varepsilon\}$.

Definition 3 For $\varepsilon > 0$ the *ε-mollification* of a locally integrable function $F\colon \mathbb{R}^n \to \mathbb{R}$ is the convolution $F^\varepsilon = \eta_\varepsilon * F$ on $\mathbb{R}^n$, that is,

$$F^\varepsilon(x) = \int_{\mathbb{R}^n} \eta_\varepsilon(x-z)F(z)\,\mathrm{d}z = \int_{B(0,\varepsilon)} \eta_\varepsilon(z)F(x-z)\,\mathrm{d}z$$

for all $x \in \mathbb{R}^n$.

Theorem 1 ([2])
1. *For all $\varepsilon > 0$, the ε-mollification F^ε is in $C^\infty(\mathbb{R}^n, \mathbb{R})$.*
2. *If F is continuous on $\mathbb{R}^n$, then F^ε converges to F uniformly on compact sets for $\varepsilon \to 0$.*

Formulation

To formulate the smoothing method, the following three weak assumptions are made in [14,15].

Assumption 1 The feasible set M of *SIP* is nonempty and compact.

Assumption 2 The EMFCQ holds everywhere in M.

Assumption 3 All KKT points of *SIP* are nondegenerate.

The smoothing approach is based on the mollification of the optimal value function G:

$$G^\varepsilon = \eta_\varepsilon * G = \eta_\varepsilon * \min_{y \in Y} g(\cdot, y)\,.$$

In view of Theorem 1, the function G^ε is C^∞ for each $\varepsilon > 0$, and G^ε converges to G uniformly on compact sets for $\varepsilon \to 0$.

Intuitively, for sufficiently small $\varepsilon > 0$ the set

$$M^\varepsilon = \{x \in \mathbb{R}^n | G^\varepsilon(x) \geq 0\}$$

and the smooth finite optimization problem

$$\mathrm{SIP}^\varepsilon\colon \quad \min_{x \in \mathbb{R}^n} f(x) \text{ subject to } G^\varepsilon(x) \geq 0$$

should be strongly related to M and SIP, respectively. This statement is made precise in the following theorems.

Theorem 2 ([14]) *M^ε converges to M in the Hausdorff distance for $\varepsilon \to 0$.*

Theorem 3 ([14]) *For all sufficiently small $\varepsilon > 0$, EMFCQ holds everywhere in the set M^ε.*

Theorem 4 ([14]) *For all sufficiently small $\varepsilon > 0$, the set M^ε is homeomorphic with M.*

Theorem 5 ([14,15])
1. *The set $KKT(f, M)$ of Karush–Kuhn–Tucker points of SIP is finite.*
2. *For each $\bar{x} \in KKT(f, M)$ let $U(\bar{x})$ be some neighborhood of $\bar{x}$. Then outside the sets $U(\bar{x}), \bar{x} \in KKT(f, M)$, the problem SIP^ε has no KKT points for sufficiently small $\varepsilon > 0$.*
3. *The neighborhoods $U(\bar{x}), \bar{x} \in KKT(f, M)$, from part 2 can be chosen such that each $U(\bar{x})$ contains exactly one KKT point x^ε of SIP^ε for sufficiently small $\varepsilon > 0$. Moreover, x^ε is nondegenerate, and the Morse index of $\bar{x}$ in SIP and the Morse index of x^ε in SIP^ε coincide.*

Conclusions

As an application of smoothing by mollifiers, Jongen and Stein [14] showed an important topological property of semi-infinite optimization problems. In fact, assume that at any $x \in M$ one can define ascent and

descent directions for f. Then these define ascent and descent flows for f, respectively. For compact M suppose that all local minima and maxima of f on M are isolated critical points. Starting in a neighborhood of a local minimum one follows the ascent flow and might reach a local maximum. From there one steps downwards via the descent flow and might reach a local minimum. Perhaps the latter minimum is different from the former one, and the previous procedure is repeated. In this way one obtains a kind of "bang-bang" path in M which connects certain local minima and local maxima. The main question that arises is whether one can reach all local minima via such a bang-bang strategy or, equivalently, if a certain "min–max digraph" is strongly connected [12]. Of course, M has to be assumed to be connected, since only local information is used.

Even for finitely many constraints, in general the answer to the latter question is negative. A two-dimensional counterexample was given in [23], and the general mechanism which generates obstructions is presented in [4]. On the other hand, a special global adaptation of the metric, constructed in [12], gives a positive result. Moreover, Jongen and Stein [13] presented an automatic adaptation of the metric based on local information which generically gives a positive result.

Smoothing by mollifiers allows one to derive a similar result for SIP. In fact, for generic Riemannian metrics and sufficiently small $\varepsilon > 0$ the corresponding min–max digraph of SIP$^\varepsilon$ is strongly connected [14]. In view of Theorem 5 the corresponding KKT points (especially the local minima and maxima) of SIP$^\varepsilon$ are arbitrarily close to those of the unperturbed SIP. This shows that SIP can be approximated arbitrarily well by a smooth finite SIP$^\varepsilon$ with strongly connected min–max digraph.

See also

- ► Generalized Semi-infinite Programming: Optimality Conditions
- ► Nonsmooth Analysis: Weak Stationarity
- ► Nonsmooth and Smoothing Methods for Nonlinear Complementarity Problems and Variational Inequalities
- ► Semi-infinite Programming, Semidefinite Programming and Perfect Duality

References

1. Danskin JM (1967) The Theory of Max-Min and its Applications to Weapons Allocation Problems. Springer, New York
2. Evans LC (1998) Partial Differential Equations. American Mathematical Society, Providence, Rhode Island
3. Fiacco AV, McCormick GP (1968) Nonlinear Programming: Sequential Unconstrained Minimization Techniques. Wiley, New York
4. Günzel H, Jongen HT (2005) On absorbing cycles in min-max digraphs. J Glob Optim 31:85–92
5. Guerra Vasquez F, Günzel H, Jongen HT (2001) On logarithmic smoothing of the maximum function. Ann Oper Res 101:209–220
6. Guerra Vasquez F, Rückmann J-J (2002) An approximation of feasible sets in semi-infinite optimization. Top 10:325–336
7. Hettich R, Jongen HT (1978) Semi-infinite programming: conditions of optimality and applications. In: Stoer J (ed) Optimization Techniques, Part 2, Lecture Notes in Control and Information Sciences, vol 7. Springer, Berlin, pp 1–11
8. Hettich R, Kortanek KO (1993) Semi-infinite programming: theory, methods, and applications. SIAM Rev 35:380–429
9. Hettich R, Zencke P (1982) Numerische Methoden der Approximation und semi-infiniten Optimierung. Teubner, Stuttgart
10. Jarre F (1999) Comparing two interior-point approaches for semi-infinite programs, Preprint, University of Trier, http://www.opt.uni-duesseldorf.de/~jarre/papers/semi2.ps
11. Jongen HT, Jonker P, Twilt F (1986) Critical sets in parametric optimization. Math Program 34:333–353
12. Jongen HT, Ruiz Jhones A (1999) Nonlinear Optimization: On the min-max digraph and global smoothing. In: Ioffe A, Reich S, Shafrir I (eds) Calculus of Variations and Differential Equations, Chapman and Hall/CRC Research Notes in Mathematics Series, vol 410. CRC Press, Boca Raton, pp 119–135
13. Jongen HT, Stein O (2004) Constrained global optimization: adaptive gradient flows. In: Floudas CA, Pardalos PM (eds) Frontiers in Global Optimization, Kluwer, Boston, pp 223–236
14. Jongen HT, Stein O (2006) Smoothing by mollifiers. J Glob Optim. doi:10.1007/s10898-007-9232-3
15. Jongen HT, Stein O (2006) Smoothing by mollifiers. J Glob Optim. doi:10.1007/s10898-007-9231-4
16. Jongen HT, Twilt F, Weber G-W (1992) Semi-infinite optimization: structure and stability of the feasible set. J Optim Theory Appl 72:529–552
17. Mangasarian OL, Fromovitz S (1967) The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. J Math Anal Appl 17:37–47
18. Lim AEB, Moore JB (1998) A path following algorithm for infinite quadratic programming on a Hilbert space. Discret Continuous Dyn Syst 4:653–670

19. Polak E (1987) On the mathematical foundation of nondifferentiable optimization in engineering design. SIAM Rev 29:21–89
20. Stein O (1997) On parametric semi-infinite optimization. Thesis, Shaker, Aachen
21. Stein O (2004) On constraint qualifications in non-smooth optimization. J Optim Theory Appl 121:647–671
22. Wetterling W (1970) Definitheitsbedingungen für relative Extrema bei Optimierungs- und Approximationsaufgaben. Num Math 15:122–136
23. Zank H (1994) Personal communication
24. Zwier G (1987) Structural Analysis in Semi-Infinite Programming. Thesis, University of Twente

Smooth Nonlinear Nonconvex Optimization

Tamás Rapcsák
Computer and Automation Institute,
Hungarian Acad. Sci., Budapest, Hungary

MSC2000: 90C26

Article Outline

Keywords
See also
References

Keywords

Smooth nonlinear optimization; Nonconvex optimization; Lagrange multiplier rule; Optimality conditions; Geodesic convexity

Smooth *optimization problems* can be considered in the form of

$$\begin{cases} \min & f(\mathbf{x}) \\ \text{s.t.} & h_j(\mathbf{x}) = 0, \quad j = 1, \dots, n-k, \\ & g_i(\mathbf{x}) \le 0, \quad i = 1, \dots, m, \\ & \mathbf{x} \in \mathbb{R}^n, \end{cases} \tag{1}$$

where $k > 0$, $\mathbf{R}^n$ is the n-dimensional Euclidean space, $f, h_j, j = 1, \dots, n-k, g_i, i = 1, \dots, m$, are at least twice continuously differentiable real valued functions and the aim is to find a solution point and/or the optimal value of the objective function f. Note that the underlying space can be more general than the Euclidean one, e. g., Hilbert space. A smooth optimization problem (1) is *nonlinear* if the objective function f or at least one *constraint* function is nonlinear. Problem (1) is *nonconvex* if at least one function from $f, g_i, i = 1, \dots, m$, is not convex or at least one function from $h_j, j = 1, \dots, n-k$, is not affine. An important class of problems (1) is that of the *unconstrained optimization problems*, where the constraints g_i and h_j, for all i, j are not present or where every point in the domain of f is *feasible*, i. e., satisfies the constraints. If the number of constraints is infinite, then (1) result in *semi-infinite optimization problems*, and if the variables are restricted to a subset of the integers, then *integer optimization problems* are obtained. Since minimization and maximization are mathematically equivalent, without loss of generality, maximization should be replaced with minimization in (1). The practical applications of *nonlinear optimization* are incredibly vast, and moreover, smooth nonlinear optimization has very good properties with respect to structural investigations and *computational performances*.

Problem (1) can be considered a representation of models providing tools to describe real-life constraints of different types. For theoretical investigations, other representations could be helpful. Let $\mathbf{h}$ denote the map from $\mathbf{R}^n$ into $\mathbf{R}^{n-k}$ of components $h_j, j = 1, \dots, n-k$; furthermore, assume that the following *regularity condition* holds: 0 is a *regular value* of $\mathbf{h}$, i. e., the Jacobian matrix $J\mathbf{h}(\mathbf{x}) \in \mathcal{L}(\mathbf{R}^n, \mathbf{R}^{n-k})$ of $\mathbf{h}$ at $\mathbf{x}$ is of full rank $(n-k)$ for all $\mathbf{x} \in M = \{\mathbf{x} \in \mathbf{R}^n : h_j(\mathbf{x}) = 0, j = 1, \dots, n-k\}$. Under this assumption, the *feasible set*

$$A = \{\mathbf{x} \in M : \ g_i(\mathbf{x}) \le 0, \ i = 1, \dots, m\} \tag{2}$$

is a subset of the k-dimensional submanifold M of class C^2 in $\mathbf{R}^n$ which can be endowed with a *Riemannian metric* (e. g., the one induced by the Euclidean structure of $\mathbf{R}^n$). Assume, furthermore, that A is connected. In order to better see the structure of problem (1), we reformulate it into the following form:

$$\begin{cases} \min & f(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in A \subseteq M \subseteq \mathcal{M}, \end{cases} \tag{3}$$

where M is a k-dimensional *Riemannian manifold* and $\mathcal{M}$ is the n-dimensional differentiable manifold $\mathbf{R}^n$ endowed with the Riemannian metric $G_1(\mathbf{x}) = I$, $\mathbf{x} \in \mathbf{R}^n$, which induces the Riemannian metric of M defined as the restriction of the $n \times n$ identity matrix to all the tangent spaces of M. The speciality of problem (1) is that

the representation of the manifold M is not a *curvilinear coordinate system* in the sense of differential geometry and the essential condition $M \subseteq \mathcal{M}$ holds, which motivates the investigation of the common curvilinear coordinate representations of $\mathcal{M}$ and M from the point of view of nonlinear optimization.

A *local minimizer* of problem (1) or (3) is a feasible point $\mathbf{x}^* \in A$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x}$ in a feasible neighborhood of $\mathbf{x}^*$. If $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all feasible $\mathbf{x}$, then $\mathbf{x}^*$ is called a *global optimizer*. If $\mathbf{x}^*$ is a local optimizer and $f(\mathbf{x}^*) < f(\mathbf{x})$ for all $\mathbf{x} \neq \mathbf{x}^*$ in a feasible neighborhood of $\mathbf{x}^*$, then $\mathbf{x}^*$ is called a *strict local optimizer*. If $\mathbf{x}^*$ is the only local minimizer in some feasible neighborhood of $\mathbf{x}^*$, it is called an *isolated local minimizer*.

A fundamental result due to K. Weierstrass is the fact that a feasible global minimizer of a continuous function f exists in a nonempty and compact feasible region A. If f is once continuously differentiable and $\mathbf{x}^*$ is a local unconstrained minimizer, then the gradient $\nabla f(\mathbf{x}^*) = 0$. If f is twice continuously differentiable and $\mathbf{x}^*$ is a local unconstrained minimizer, then $\nabla f(\mathbf{x}^*) = 0$ and the Hessian matrix $Hf(\mathbf{x}^*)$ is positive semidefinite. If $\nabla f(\mathbf{x}^*) = 0$ and $Hf(\mathbf{x}^*)$ is positive definite, then $\mathbf{x}^*$ is an isolated (hence, also strict) local minimizer.

In the case of a finite number of equalities, the local optimality conditions are deduced by the *Lagrange multiplier rule*, [9,10]. This classical rule was independently extended to constraints including a finite number of equalities and inequalities in [2,3,6,7,8]. A transparent description of the smooth local optimality conditions can be found, e. g., in [1,4,5,11]. By improving these rules [13], global optimality conditions can be obtained which are formulated as follows.

Let $\mathbf{x}^* \in A$ be a given point, $I(\mathbf{x}^*)$ denote the index set of the *active inequality constraints* at $\mathbf{x}^*$, $|I(\mathbf{x}^*)|$ the number of active constraints, and $\mathbf{g}_{I(x^*)}\colon \mathbf{R}^n \to \mathbf{R}^{|I(x^*)|}$ the mapping of the active constraints at $\mathbf{x}^*$. (An inequality constraint is active at $\mathbf{x}^*$ if equality holds.)

Let us introduce:

- M_{x^*} as the set

$$\left\{(\mathbf{x}, \mathbf{z}) \in \mathbb{R}^{n+|I(\mathbf{x}^*)|}\colon \begin{array}{c} h_j(\mathbf{x}) = 0, \\ j = 1, \ldots, n-k, \\ g_i(\mathbf{x}) + \frac{1}{2} z_i^2 = 0, \\ i \in I(\mathbf{x}^*) \end{array}\right\}, \quad (4)$$

- the set TM_{x^*}:

$$\left\{(\mathbf{v}_1, \mathbf{v}_2) \in \mathbb{R}^{n+|I(\mathbf{x}^*)|}\colon \begin{array}{c} \nabla h_j(\mathbf{x})\mathbf{v}_1 = 0, \\ j = 1, \ldots, n-k, \\ \nabla g_i(\mathbf{x})\mathbf{v}_1 + z_i v_{2i} = 0, \\ i \in I(\mathbf{x}^*), \\ (\mathbf{x}, \mathbf{z}) \in M_{\mathbf{x}^*} \end{array}\right\}, \quad (5)$$

- a regularity condition

$$\begin{aligned} &r([J\mathbf{h}(\mathbf{x}), 0]^\top, [J\mathbf{g}_{I(\mathbf{x}^*)}(\mathbf{x}), D_z]^\top) \\ &= n - k + |I(\mathbf{x}^*)|, \\ &(\mathbf{x}, \mathbf{z}) \in M_{\mathbf{x}^*}, \end{aligned} \quad (6)$$

where $J\mathbf{h}$ and $J\mathbf{g}_{I(x^*)}$ are the Jacobian matrices of the mappings $\mathbf{h}\colon \mathbf{R}^n \to \mathbf{R}^{n-k}$ and $\mathbf{g}_{I(x^*)}\colon \mathbf{R}^n \to \mathbf{R}^{|I(x^*)|}$, respectively, and $D_{\mathbf{z}} = \operatorname{diag}(z_1, \ldots, z_{I(x^*)|})$ the diagonal matrix with the components of the vector $\mathbf{z}$.

Here, problems satisfying (6), i. e., regular problems, are considered for which the inequality $n \geq k$ holds. It is well-known that there cannot exist Lagrange multipliers for a local minimum in an irregular problem. Instead of (1) or (3), let us consider the problem

$$\min_{(\mathbf{x},\mathbf{z}) \in M_{\mathbf{x}^*}} f(\mathbf{x}). \quad (7)$$

As a point $\mathbf{x}^* \in A$ is a local optimal solution of problem (1) if and only if $(\mathbf{x}^*, 0) \in M_{x^*}$ is a local optimal solution of (7), and since the orthogonal projection of M_{x^*} to $\mathbf{R}^n$ with respect to the Euclidean metric contains A, a point $\mathbf{x}^* \in A$ is a global optimal solution of problem (1) if $(\mathbf{x}^*, 0) \in M_{x^*}$ is a global optimal solution of (7), we deal with this latter problem only.

Let the *Lagrangian function* associated with f and M_{x^*} be defined as

$$\begin{aligned} &L(\mathbf{x}, \mathbf{z}, \mu(\mathbf{x}), \lambda(\mathbf{x}, \mathbf{z})) \\ &= f(\mathbf{x}) - \sum_{j=1}^{n-k} \mu_j(\mathbf{x}) h_j(\mathbf{x}) \\ &- \sum_{i \in I(\mathbf{x}^*)} \lambda_i(\mathbf{x}, \mathbf{z}) \left(g_i(\mathbf{x}) + \frac{1}{2} z_i^2\right), \\ &(\mathbf{x}, \mathbf{z}) \in M_{\mathbf{x}^*}, \\ &\mu\colon \mathbb{R}^n \to \mathbb{R}^{n-k}, \\ &\lambda\colon \mathbb{R}^{n+|I(\mathbf{x}^*)|} \to \mathbb{R}^{|I(\mathbf{x}^*)|}, \end{aligned} \quad (8)$$

where

$$\mu(\mathbf{x})^\top = \nabla f(\mathbf{x}) J\mathbf{h}^\top(\mathbf{x}) \left[J\mathbf{h}(\mathbf{x}) J\mathbf{h}(\mathbf{x})^\top\right]^{-1},$$
$$\lambda(\mathbf{x},\mathbf{z})^\top = \nabla f(\mathbf{x}) J\mathbf{g}^\top_{I(\mathbf{x}^*)}(\mathbf{x})$$
$$\times \left[[J\mathbf{g}_{I(\mathbf{x}^*)}(\mathbf{x}), D_z][J\mathbf{g}_{I(\mathbf{x}^*)}(\mathbf{x}), D_z]^\top\right]^{-1}.$$

Let the *geodesic gradient vector* and the *geodesic Hessian matrix* of the Lagrangian function (8) be defined as

$$\nabla^g_{\mathbf{x}} L(\mathbf{x},\mathbf{z},\mu(\mathbf{x}),\lambda(\mathbf{x},\mathbf{z}))$$
$$= \nabla f(\mathbf{x}) - \sum_{j=1}^{n-k} \mu_j(\mathbf{x}) \nabla h_j(\mathbf{x})$$
$$- \sum_{i\in I(\mathbf{x}^*)} \lambda_i(\mathbf{x},\mathbf{z}) \nabla g_i(\mathbf{x}),$$
$$(\mathbf{x},\mathbf{z}) \in M_{\mathbf{x}^*},$$
$$\nabla^g_{\mathbf{z}} L(\mathbf{x},\mathbf{z},\mu(\mathbf{x}),\lambda(\mathbf{x},\mathbf{z}))$$
$$= - \sum_{i\in I(\mathbf{x}^*)} \lambda_i(\mathbf{x},\mathbf{z}) z_i \mathbf{e}_i^\top,$$
$$(\mathbf{x},\mathbf{z}) \in M_{\mathbf{x}^*},$$

where $\mathbf{e}_i$, $i = 1,\ldots,|I(\mathbf{x}^*)|$, are the unit vectors,

$$H^g_{(\mathbf{x},\mathbf{z})} L(\mathbf{x},\mathbf{z},\mu(\mathbf{x}),\lambda(\mathbf{x},\mathbf{z}))$$
$$= \begin{pmatrix} Hf(\mathbf{x}) - \sum\limits_{j=1}^{n-k} \mu_j(\mathbf{x}) Hh_j(\mathbf{x}) & \\ - \sum\limits_{i\in I(\mathbf{x}^*)} \lambda_i(\mathbf{x},\mathbf{z}) Hg_i(\mathbf{x}) & 0 \\ 0 & -D_\lambda \end{pmatrix}_{|TM_{\mathbf{x}^*}}, \quad (9)$$
$$(\mathbf{x},\mathbf{z}) \in M_{\mathbf{x}^*},$$

where the symbol $|_{TM_{x^*}}$ denotes the restriction to the tangent spaces of M_{x^*} and D_λ is the diagonal matrix with components $\lambda_i(\mathbf{x},\mathbf{z})$, $i = 1,\ldots,|I(\mathbf{x}^*)|$, at $(\mathbf{x},\mathbf{z})$.

Now, the global Lagrange multiplier rule is formulated for the case of equality and inequality constraints. First, a definition of *geodesic convex sets* is recalled where the geodesic is used in the classical meaning. If M is a Riemannian manifold, then a set $A \subseteq M$ is geodesic convex if any two points of A are joined by a geodesic belonging to A, moreover, a singleton is geodesic convex. It is emphasized that every Riemannian metric generates a geodesic convexity notion. In optimization theory related to the Lagrange multiplier rule, the induced Riemannian metric seems to be the most important.

Theorem 1 (*Global Lagrange multiplier rule*) *If the point $(\mathbf{x}^*, 0) \in M_{x^*}$ is a (strict) local or global minimum of problem (7), then*

$$\nabla^g_{(\mathbf{x},\mathbf{z})} L(\mathbf{x}^*, 0, \mu(\mathbf{x}^*), \lambda(\mathbf{x}^*, 0)) = 0,$$
$$(\mathbf{v}_1,\mathbf{v}_2)^\top H^g_{(\mathbf{x},\mathbf{z})} L(\mathbf{x}^*, 0, \mu(\mathbf{x}^*), \lambda(\mathbf{x}^*, 0))$$
$$\times (\mathbf{v}_1,\mathbf{v}_2) \geq (>)\, 0,$$
$$(\mathbf{v}_1,\mathbf{v}_2) \in TM_{\mathbf{x}^*}(\mathbf{x}^*, 0),$$
$$((\mathbf{v}_1,\mathbf{v}_2) \neq 0).$$

If $\widehat{A} \subseteq M_{\mathbf{x}^*}$ is an open geodesic convex set with respect to the induced Riemannian metric and

$$\begin{aligned} &\nabla^g_{(\mathbf{x},\mathbf{z})} L(\mathbf{x}^*, 0, \mu(\mathbf{x}^*), \lambda(\mathbf{x}^*, 0)) = 0,\\ &(\mathbf{v}_1,\mathbf{v}_2)^\top H^g_{(\mathbf{x},\mathbf{z})} L(\mathbf{x},\mathbf{z},\mu(\mathbf{x}),\lambda(\mathbf{x},\mathbf{z}))\\ &\times (\mathbf{v}_1,\mathbf{v}_2) \geq (>)\, 0,\\ &(\mathbf{v}_1,\mathbf{v}_2) \in TM_{\mathbf{x}^*}(\mathbf{x},\mathbf{z}),\\ &((\mathbf{v}_1,\mathbf{v}_2) \neq 0),\\ &(\mathbf{x},\mathbf{z}) \in M_{\mathbf{x}^*}, \end{aligned} \quad (10)$$

then the point $(\mathbf{x}^*, 0)$ is a (strict) global minimum of the function f on $\widehat{A}$. Moreover,

$$\nabla^g_{(\mathbf{x},\mathbf{z})} L(\mathbf{x},\mathbf{z},\mu(\mathbf{x}),\lambda(\mathbf{x},\mathbf{z})) = Df(\mathbf{x},\mathbf{z}), \quad (\mathbf{x},\mathbf{z}) \in M_{\mathbf{x}^*}, \quad (11)$$
$$H^g_{(\mathbf{x},\mathbf{z})} L(\mathbf{x},\mathbf{z},\mu(\mathbf{x}),\lambda(\mathbf{x},\mathbf{z})) = D^2 f(\mathbf{x},\mathbf{z}), \quad (\mathbf{x},\mathbf{z}) \in M_{\mathbf{x}^*}, \quad (12)$$

where Df and $D^2 f$ are the first and second covariant derivatives of the function f with respect to the induced Riemannian metric of the manifold M_{x^*}, respectively.

Because of the linear independence of the active gradients, the first order optimality condition is equivalent to the classical one, and in the case of equality constrained problems, the second order optimality conditions coincide with the classical ones at the stationary points as well, moreover, a geodesic convex feasible neighborhood always exists around a stationary point in a Riemannian manifold, so this latter condition does not mean a new restriction. If inequality constraints are present, then from (5) and (9), the classical optimality conditions and the nonpositivity of the Lagrange multipliers at the stationary points can be deduced. In this approach, neither Farkas' lemma in the necessary part

nor strict complementarity assumption ($\lambda_i(\mathbf{x}^*, 0) > 0$, $i \in I(\mathbf{x}^*)$) in the sufficiency part are used; they are replaced by the regularity condition and the smoothness of the functions.

By the global Lagrange multiplier rule, the necessary and sufficient optimality conditions are given by the same tensor formulae based on the first and second covariant derivatives, only the domains are different where the second order formula holds. The second order conditions (10) define a class of functions on geodesic convex sets with respect to the induced Riemannian metric, the *geodesic convex functions* with respect to the same metric, introduced in optimization theory in [12]. It is recalled that if M is a Riemannian manifold and $A \subseteq M$ a geodesic convex set, then a function $f: A \to \mathbf{R}$ is geodesic (strictly) convex if its restrictions to all geodesic arcs belonging to A are (strictly) convex in the arc length parameter. From the point of view of geometry, the existence of a constrained minimum is equivalent to the existence of a geodesic convex function with respect to the induced Riemannian metric. It follows that the Lagrange method can be considered the transformation of a constrained problem in $\mathbf{R}^n$ into an unconstrained problem on the constraint submanifold with the induced Riemannian metric in $\mathbf{R}^n$. In the case of a Euclidean space, the geodesic convexity coincides with the classical one. The application of the Riemannian geometry highlighted the geometric background of smooth optimization and provides it with strong mathematical tools to study structural properties (e. g., geodesic convexity) and to deepen the theory of algorithms (e. g., *variable metric methods*).

See also

- ▶ αBB Algorithm
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Global Optimization in Batch Design Under Uncertainty
- ▶ Global Optimization in Generalized Geometric Programming
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization in Phase and Chemical Reaction Equilibrium
- ▶ Interval Global Optimization
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Global Optimization with αBB

References

1. Bertsekas DP (1995) Nonlinear programming. Athena Sci., Belmont, MA
2. Farkas J (1901) Theorie der einfachen Ungleichungen. J Reine Angew Math 124:1–27
3. Farkas J (1906) Beiträge zu den Grundlagen der analytischen Mechanik. J Reine Angew Math 131:165–201
4. Fiacco AV, McCormick GP (1968) Nonlinear programming, sequential unconstrained minimization techniques. Wiley, New York
5. Hestenes MR (1975) Optimization theory. The finite dimensional case. Wiley, New York
6. John F (1948) Extremum problems with inequalities as subsidary conditions. In: Friedricks KD ET AL (eds) Studies and Essays, Courant Anniv. Vol. Interscience, New York, pp 187–204
7. Karush W (1939) Minima of functions of several variables with inequalities as side conditions. PhD Thesis Dept. Math. Univ. Chicago
8. Kuhn HW, Tucker AW (1951) Nonlinear programming. In: Neyman J (ed) Proc. Second Berkeley Symp. Math. Stat. Probab. Univ. Calif. Press, Berkeley, CA, pp 481–492
9. Lagrange JL (1762) Essai sur une nouvelle mèthode pour determiner les maxima et minima des formules integrales indefinies. Miscellanea Taurinensia 2:173–195
10. Lagrange JL (1788) Mècanique analytique, vol I-II
11. Luenberger DG (1973) Introduction to linear and nonlinear programming. Addison-Wesley, Reading, MA
12. Rapcsàk T (1991) Geodesic convexity in nonlinear optimization. J Optim Th Appl 69:169–183
13. Rapcsàk T (1997) Smooth nonlinear optimization in Rn. Kluwer, Dordrecht

Solution Methods for Multivalued Variational Inequalities

Igor V. Konnov
Department of System Analysis and Information Technologies, Kazan University, Kazan, Russia

MSC2000: 47J20, 49J40, 90C33, 65K10

Article Outline

Keywords and Phrases
Problem Formulation and Basic Facts

Projection Methods for GVIs
Projection Method
Basic Solution Methods for GVIs
Averaging and Regularization Type Methods
Averaging Method
Regularization Methods
Proximal Point Method
Direct Iterative Methods for GVIs
Center-Type Methods
Combined Relaxation Methods
Iterative Methods
for Generalized Complementarity Problems
Properties of Multivalued Z-Mappings
Extended Jacobi Algorithm
for Multivalued Mixed Complementarity Problems
Iterative Methods for MVIs
Descent Methods for MVIs
Combined Relaxation Methods for MVIs
References

Keywords and Phrases

Variational inequalities; Multivalued mappings; Solution methods

Problem Formulation and Basic Facts

Let X be a nonempty convex set in the real n-dimensional space $\mathbb{R}^n$, and let $G\colon X \to \Pi(\mathbb{R}^n)$ be a multivalued mapping. Here and below $\Pi(A)$ denotes the family of all nonempty subsets of a set A. Then one can define the multivalued or *generalized variational inequality* (GVI) *problem*, which is to find an element $x^* \in X$ such that

$$\exists g^* \in G(x^*), \langle g^*, y - x^* \rangle \geq 0 \quad \forall y \in X\,. \tag{1}$$

If the cost mapping G is single-valued, GVI (1) reduces to the following usual *variational inequality* (VI) *problem*: Find an element $x^* \in X$ such that

$$\langle G(x^*), y - x^* \rangle \geq 0 \quad \forall y \in X\,, \tag{2}$$

where $G\colon \mathbb{R}^n \to \mathbb{R}^n$ is a given mapping.

VIs are now regarded as very useful and powerful tools for investigation and solution of various equilibrium-type problems arising in economics, engineering, operations research and mathematical physics. Many such applied problems involve multivalued mappings with rather weak continuity properties. The article is devoted to the construction of solution methods for VIs with multivalued cost mappings. These problems involve in particular multivalued inclusions, complementarity and fixed-point problems, nonsmooth optimization and game problems, and mixed VIs (MVIs). Problem (1) was originally considered by Browder [5].

It is well known that the multivaluedness creates certain difficulties for providing convergence of many iterative methods, which are applied successfully to single-valued problems. This fact leads to the necessity of construction of new solution methods. In this article, we outline briefly the current situation and describe some new advances in this field.

First we consider some existence results for GVI (1) which are based on certain continuity-type properties of multivalued mappings; see, e. g., [11,16,17].

Definition 1 A multivalued mapping $Q\colon \mathbb{R}^n \to \Pi(\mathbb{R}^n)$ is said to be a K (Kakutani) mapping on X if it is upper semicontinuous on X and has nonempty, convex, and compact values.

Proposition 1 *Let $G\colon X \to \Pi(\mathbb{R}^n)$ be a K-mapping. Suppose at least one of the following assumptions holds:*

(a) The set X is bounded;

(b) there exists a nonempty bounded subset Y of X such that for every $x \in X \setminus Y$ there is $y \in Y$ with

$$\langle g, x - y \rangle \geq 0 \quad \forall g \in G(x)\,.$$

Then GVI (1) has a solution.

The solution of GVI (1) is closely related to that of the corresponding *dual (or Minty) GVI* (DGVI) *problem*, which is to find a point $\tilde{x} \in X$ such that

$$\forall x \in X \text{ and } \forall g \in G(x)\colon \langle g, x - \tilde{x} \rangle \geq 0\,. \tag{3}$$

We denote by X^* (respectively, by X^d) the solution set of problem (1) (respectively, problem (3)). Recall certain monotonicity-type properties for multivalued mappings.

Definition 2 Let $Q\colon \mathbb{R}^n \to \Pi(\mathbb{R}^n)$ be a multivalued mapping. The mapping Q is said to be

(a) *strongly monotone* on X with constant $\tau > 0$ if for each pair of points $x, y \in X$ and for all $q' \in Q(x)$, $q'' \in Q(y)$, we have

$$\langle q' - q'', x - y \rangle \geq \tau \|x - y\|^2\,;$$

(b) *strictly monotone* on X if for all distinct $x, y \in X$ and for all $q' \in Q(x)$, $q'' \in Q(y)$, we have

$$\langle q' - q'', x - y\rangle > 0\,;$$

(c) *monotone* on X if for each pair of points $x, y \in X$ and for all $q' \in Q(x)$, $q'' \in Q(y)$, we have

$$\langle q' - q'', x - y\rangle \geq 0\,;$$

(d) *pseudomonotone* on X if for each pair of points $x, y \in X$ and for all $q' \in Q(x)$, $q'' \in Q(y)$, we have

$$\langle q'', x - y\rangle \geq 0 \;\text{ implies }\; \langle q', x - y\rangle \geq 0\,.$$

From the definitions we have the following implications:

$$(a) \Longrightarrow (b) \Longrightarrow (c) \Longrightarrow (d)\,.$$

The reverse assertions are not true in general.

Now we give an extension of the Minty lemma for the multivalued case.

Proposition 2
(i) The set X^d is convex and closed.
(ii) If G is a K-mapping, then $X^d \subseteq X^*$.
(iii) If G is pseudomonotone, then $X^* \subseteq X^d$.

We also recall some conditions under which GVI (1) has a unique solution.

Proposition 3
(i) If G is strictly monotone, then GVI (1) has at most one solution.
(ii) If G is a strongly monotone K-mapping, then GVI (1) has a unique solution.

Of course, there exist various modifications and extensions of the above results; see, e. g., [3,16] for more details.

Projection Methods for GVIs

We observe that the existence and uniqueness results for multivalued problems are very similar to those for single-valued VIs, but this is not the case for solution methods in general. That is, the substantiation of convergence and derivation of rates of convergence for iterative methods applied to multivalued problems meet certain difficulties in comparison with those in the single-valued case. This reduces essentially the number of approaches to the creation of efficient solution methods. To illustrate this assertion, we first outline the behavior of projection-type methods.

Unless otherwise stated, throughout the article we suppose that

(C1) *X is a nonempty, convex and closed subset of the real n-dimensional space $\mathbb{R}^n$, $G\colon X \to \Pi(\mathbb{R}^n)$ is a K-mapping.*

Projection Method

Let us consider the standard *projection method*

$$x^{k+1} := \pi_X[x^k - \lambda_k g^k]\,,\quad g^k \in G(x^k)\,,\quad \lambda_k > 0\,, \tag{4}$$

where $\pi_X(\cdot)$ denotes the projection mapping onto X. Usually, during the computation process we can find at least one element from $G(x^k)$ at the current point x^k, but the whole set $G(x^k)$ is not determined explicitly. The problem is to find a suitable rule for choosing the step size $\lambda_k > 0$, which provides convergence under mild assumptions and a good rate of convergence. We recall that in the single-valued case, where **(C1)** means that G is continuous, the above method is rewritten as follows

$$x^{k+1} := \pi_X[x^k - \lambda_k G(x^k)]\,,\quad \lambda_k > 0\,, \tag{5}$$

and its convergence requires either integrability, or strengthened monotonicity (co-coercivity, strong monotonicity) and Lipschitz continuity assumptions. That is, if G is of the form $G = \nabla f$, where f is a given function, the step size λ_k can be chosen in conformity with the known exact or inexact (Armijo-type) rules. Then method (5) generates a sequence whose limit points are solutions of VI (2) with $G = \nabla f$; moreover, it attains a linear rate of convergence if G is strongly monotone and Lipschitz continuous. The superlinear rate of convergence can be obtained within the conjugate gradient approach. However, this is not the case if G is not integrable. In fact, the same method (5) does not provide convergence even in the nonintegrable monotone case, for instance, when $G(x) = Ax + b$ with A being skew-symmetric, regardless of the step-size choice. Therefore, we have to utilize different step-size rules and impose certain additional assumptions, such as co-coercivity.

Definition 3 A mapping $Q\colon \mathbb{R}^n \to \mathbb{R}^n$ is said to be co-coercive with constant $\mu > 0$ on X, if for each pair

of points $x, y \in X$, we have

$$\langle Q(x) - Q(y), x - y\rangle \geq \mu \|Q(x) - Q(y)\|^2 .$$

In fact, if G is co-coercive and VI (2) is solvable, then method (5) with the fixed step size $\lambda_k = \lambda \in (0, 2\mu)$ generates a sequence $\{x^k\}$ which converges to a solution of VI (2); see [15]. At the same time, the co-coercivity of G again implies the single-valuedness and even Lipschitz continuity of G.

Nevertheless, method (4) becomes convergent if we utilize the *divergent series step-size rule*

$$\lambda_k = \frac{\alpha_k}{\|g^k\|}, \quad \alpha_k > 0, \quad \sum_{k=0}^{\infty} \alpha_k = \infty, \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty, \quad (6)$$

and replace monotonicity of G with the *acute angle condition*:

(C2′) *GVI* (1) *is solvable, and for every* $x^* \in X^*$, *it holds that*

$$\forall x \in X \backslash X^*, \ \forall g \in G(x), \langle g, x - x^*\rangle > 0 ; \quad (7)$$

see [15]. This property has clear geometric sense: the angle between $-G(x)$ and $x^* - x$ has to be acute at each nonoptimal point x. For instance, (7) holds if G is strictly monotone.

However, rule (6) leads to very slow convergence and prevents the method from attaining a linear rate of convergence. For this reason, we have to apply other approaches

- *to utilize more efficient step-size rules,*
- *to attain more rapid convergence,*
- *to weaken sufficient conditions for convergence.*

Basic Solution Methods for GVIs

So, we intend to describe some other solution methods for GVI (1). First of all, in addition to **(C1)** we will utilize the following weakened condition

(C2) *GVI* (1) *is solvable, and for every* $x^* \in X^*$, *it holds that*

$$\forall x \in X, \ \forall g \in G(x), \langle g, x - x^*\rangle \geq 0 \quad (8)$$

(cf. (3) and (7)) or the somewhat more restrictive, but simplified version

(C3) *GVI* (1) *is solvable, and G is monotone.*

Clearly, we have the following implications

$$\textbf{(C2}') \Longrightarrow \textbf{(C2)} \quad \text{and} \quad \textbf{(C3)} \Longrightarrow \textbf{(C2)}$$

but the reverse assertions are not true.

Owing to Proposition 2, we see that **(C2)** is equivalent to

$$X^* = X^d \neq \emptyset$$

for GVI (1) if **(C1)** is fulfilled. Also note that (8) may in principle be called the *nonobtuse angle condition*; see [26] for more details.

We divide the basic solution methods into the following families:

- **Averaging methods;**
- **Center-type methods;**
- **Combined relaxation methods;**
- **Proximal point methods;**
- **Regularization methods.**

In the next sections, we describe properties of these general approaches to enhance the convergence properties of methods (4) and (6).

Averaging and Regularization Type Methods

We now consider the methods which utilize modifications of the initial problems or some other kind of convergence.

Averaging Method

The idea of the *averaging method* consists in replacing the usual convergence of $\{x^k\}$ with an ergodic convergence. It utilizes the same divergent series step-size rule (6). In fact, the sequence

$$z^k = \sum_{i=0}^{k} \alpha_i x^i \Big/ \sum_{i=0}^{k} \alpha_i , \quad (9)$$

enjoys stronger convergence properties than $\{x^k\}$. This idea leads to the so-called *averaging method*, which is due to Bruck [8].

Method (AVR). Choose a point $x^0 \in X$ and a positive sequence $\{\alpha_k\}$. Set $z^0 := x^0$, $\beta_0 := \alpha_0$. At the kth iteration, $k = 0, 1, \ldots$, set

$$\beta_{k+1} := \beta_k + \alpha_{k+1}, \tau_{k+1} = \alpha_{k+1}/\beta_{k+1} ;$$
$$x^{k+1} := \pi_X(x^k + \alpha_k g^k) , g^k \in G(x^k) ;$$
$$z^{k+1} := \tau_{k+1} x^{k+1} + (1 - \tau_{k+1}) z^k .$$

From the description it follows that the sequence $\{z^k\}$ generated by (AVR) satisfies (9).

Theorem 1 *Suppose that* ***(C1)*** *and* ***(C3)*** *hold and that sequences $\{x^k\}$ and $\{z^k\}$ are constructed by (AVR) and that the sequence $\{\alpha_k\}$ satisfies the following conditions:*

$$\sum_{k=0}^{\infty} \alpha_k = \infty\,, \quad \sum_{k=0}^{\infty} (\alpha_k \|g^k\|)^2 < \infty\,.$$

Then there exist limit points of $\{z^k\}$ and all these points belong to X^*.

The rate of convergence of the averaging method was investigated by several authors. It was shown by Nemirovskii [40] that $\|z^k - x^*\| = O(1/\sqrt{k})$, where $x^* \in X^*$.

Regularization Methods

The idea of the earliest and most popular regularization method consists in replacing the initial GVI (1) with a sequence of the following auxiliary problems: Find a point $x^\varepsilon \in X$ such that

$$\exists g^\varepsilon \in G(x^\varepsilon),\ \langle g^\varepsilon + \varepsilon x^\varepsilon, x - x^\varepsilon\rangle \geq 0 \quad \forall x \in X\,, \quad (10)$$

where $\varepsilon > 0$ is a regularization parameter. It was first proposed by Tikhonov [46] and was adjusted to VIs by Browder [7]. Suppose that **(C1)** and **(C3)** hold. Then G is monotone, $G + \varepsilon I$ is strongly monotone and, by Proposition 3, (10) has a unique solution, which can be found by one of the versions of the above projection method within a given accuracy. The basic approximation property of the exact regularization method is formulated as follows:

Theorem 2 *Suppose that* ***(C1)*** *and* ***(C3)*** *are fulfilled and that the sequence $\{x^{\varepsilon_k}\}$ is obtained from (10) with $\{\varepsilon_k\} \searrow 0$. Then the following assertions are true:*

(i) each auxiliary GVI (10) has a unique solution;

(ii) the sequence $\{x^{\varepsilon_k}\}$ converges to the solution x_n^* of (1) nearest to the origin.

We also can replace **(C3)** with **(C2)** and obtain similar convergence properties despite the fact that the cost mapping in (10) is not monotone. We present a strengthened version of the result from [34].

Theorem 3 *Suppose that* ***(C1)*** *and* ***(C2)*** *are fulfilled and that the sequence $\{x^{\varepsilon_k}\}$ is obtained from (10) with $\{\varepsilon_k\} \searrow 0$. Then the following assertions are true:*

(i) each auxiliary GVI (10) has a solution;

(ii) $\{x^{\varepsilon_k}\}$ converges to the solution x_n^* of (1) nearest to origin.

Moreover, we can obtain convergence results for (RM) under even weaker conditions which are utilized for providing existence results for GVIs; see [33]. Namely, let us consider the following coercivity condition (see e. g. [3]):

(C2′′) *There exists a number $r > 0$ such that for any point $x \in X \setminus X_r$ there is a point $y \in X$, $\|y\| < \|x\|$ such that $\langle g, y - x\rangle \leq 0$, $\forall g \in G(x)$, where*

$$X_r = \{x \in X \mid \|x\| \leq r\}\,.$$

The basic approximation properties of the regularization method are then formulated as follows:

Theorem 4 *Suppose conditions* ***(C1)*** *and* ***(C2")*** *are fulfilled. Then:*

(i) GVI (1) has a solution;

(ii) GVI (10) has a solution for each $\varepsilon > 0$;

(iii) Each sequence $\{x^{\varepsilon_k}\}$ of solutions of GVI (10) has limit points and if $\{\varepsilon_k\} \searrow 0$ all these limit points are solutions of GVI (1).

The regularization approach allows various modifications. One of them was proposed by Bakushinskii and Polyak [2] and is called the *iterative regularization method*. The idea of this approach consists in simultaneous changes of the regularization parameters and the step sizes of an approximation method, i. e., it is intermediate between the above averaging and regularization methods, and has similar convergence properties.

Proximal Point Method

The idea of the *proximal point method*, which was suggested by Martinet [39], also consists in replacing the initial GVI (1) with a sequence of auxiliary problems. The essential features of the proximal point method are that the regularization parameter may in principle be fixed and that the perturbed mapping depends on the previous iteration point. We first recall the convergence result for the exact version of the proximal point method applied to monotone problems.

Theorem 5 *Suppose that* ***(C1)*** *and* ***(C3)*** *are fulfilled and that a sequence $\{x^k\}$ is generated in conformity with*

the rules

$$\exists g^k \in G(x^k),\ \langle g^k + \theta^{-1}(x^k - x^{k-1}), y - x^k\rangle \geq 0 \quad \forall y \in Y\,, \quad (11)$$

where $\theta > 0$ is a regularization parameter. Then the following assertions are true:
(i) each auxiliary GVI (11) has a unique solution;
(ii) the sequence $\{x^k\}$ converges to a solution of GVI (1).

In fact, since **(C1)** and **(C3)** hold, the cost mapping in (11) is strongly monotone and, by Proposition 3, (11) has a unique solution. However, we can replace **(C3)** with **(C2)** and obtain similar convergence properties; see e. g., [1,4,12,28]. We now give such a strengthened result for the proximal point method.

Theorem 6 *Suppose that **(C1)** and **(C2)** are fulfilled and that a sequence $\{x^k\}$ is generated in conformity with the rules in (11) with $\theta > 0$. Then the following assertions are true:*
(i) each auxiliary GVI (11) has a solution;
(ii) the sequence $\{x^k\}$ converges to a solution of GVI (1).

Observe that the cost mapping in (11) need not be strongly monotone but (11) is still solvable. Under the additional Lipschitz continuity type condition on G we can choose θ large enough for the cost mapping in (11) to be strongly monotone, thus providing the uniqueness of a solution as well.

The exact proximal point method attains linear and even superlinear convergence rates as was shown by Rockafellar [44]. At the same time, the total rates of convergence of both the proximal point method and the regularization method, involving expenses for approximate solutions of auxiliary problems, need further investigations.

Direct Iterative Methods for GVIs

We now present iterative methods for solving GVI (1) without any explicit monotonicity assumptions.

Center-Type Methods

The best known of the center-type methods is the famous ellipsoid method, which was proposed first by Yudin and Nemirovskii [47] and by Shor [45] for convex programming and afterwards adjusted for saddle point problems and VIs [40]. In this method, each iterate x^k is associated with an ellipsoid U_k centered at x^k and containing at least one solution point. After finding a half-space $H^+{}_k$ containing this solution point, the next ellipsoid U_{k+1} is precisely the smallest ellipsoid containing the set $U_k \bigcap H_k^+$. Set

$$P(z) = \begin{cases} G(z) & \text{if } z \in X\,, \\ \{p \in \mathbb{R}^n \mid \langle p, y - z\rangle \leq 0\ \forall y \in X\} & \text{if } z \notin X\,. \end{cases}$$

Method (EM). Choose a point $x^0 \in \mathbb{R}^n$, a number $\lambda > 0$ such that $\|x^0 - x^*\| \leq \lambda$ for some $x^* \in X^*$ and set $A_0 := \lambda^2 I$. At the kth iteration, $k = 0, 1, \ldots$, choose $p^k \in P(x^k)$ and set

$$x^{k+1} := x^k - \frac{1}{n+1}\frac{A_k p^k}{\sqrt{(p^k)^T A_k p^k}}\,,$$

$$A_{k+1} := \frac{n^2}{n^2-1}\left(A_k - \frac{2}{n+1}\frac{A_k p^k (A_k p^k)^T}{(p^k)^T A_k p^k}\right)\,,$$

and $k := k + 1$.

If the basic assumptions **(C1)** and **(C2)** are fulfilled, the process is well-defined. Namely, then

$$U_k = \{x \in \mathbb{R}^n \mid \langle A_k^{-1}(x - x^k), x - x^k\rangle \leq 1\}$$

and

$$H_k^+ = \{x \in \mathbb{R}^n \mid \langle p^k, x - x^k\rangle \leq 0\}\,.$$

The implementation of (EM) is similar to that of variable metric methods. It is well known that the volumes of U_k will also tend to zero at a linear rate, which depends on the dimensionality of the problem. These properties yield the convergence of the sequence $\{x^k\}$ to a solution.

The idea of various proximal-level methods is rather close to that of the center methods [14,18,36]. In fact, the methods are based on sequential updating of a polyhedral approximation of a nonsmooth merit function for GVI and computing the prox-center of the corresponding level sets. These methods possess similar convergence properties.

Combined Relaxation Methods

The idea of *combined relaxation methods* consists in defining the next iterate x^{k+1} as the projection of the

current iterate x^k onto a hyperplane H_k which separates strictly x^k and the solution set and is computed by an auxiliary procedure. This approach to solve VIs was proposed first by Konnov [19], where it was also noticed that the parameters of the hyperplane H_k can be found with the help of an iteration of any relaxation method. Afterwards, combined relaxation methods were developed in several directions. In [20], a combined relaxation method for GVI of form (1) was proposed and linear convergence rates were established. All these methods ensure convergence to a solution of GVI under assumption **(C2)** or **(C3)**. Within the general combined relaxation framework, different rules for determining the separating hyperplane and auxiliary procedures were presented; see [26] and references therein.

Combined Relaxation Method for GVIs We now consider a combined relaxation method for solving GVI (1) with explicit usage of constraints [20,22]. In addition to the basic assumptions **(C1)** and **(C2)**, we suppose that

- *X is defined by*

$$X = \{x \in \mathbb{R}^n \mid h(x) \le 0\} ,$$

where $h\colon \mathbb{R}^n \to \mathbb{R}$ is a convex and subdifferentiable function;

- *the Slater condition is satisfied, i. e., there exists a point $\bar{x}$ such that $h(\bar{x}) < 0$.*

Let us define the mapping $Q\colon \mathbb{R}^n \to \Pi(\mathbb{R}^n)$ by

$$Q(x) = \begin{cases} G(x) & \text{if } h(x) \le 0 , \\ \partial h(x) & \text{if } h(x) > 0 . \end{cases}$$

Definition 4 A mapping $P\colon \mathbb{R}^n \to \mathbb{R}^n$ is said to be a *pseudo-projection* onto X, if for every $y \in \mathbb{R}^n$ it holds that

$$P(y) \in X \text{ and } \|P(y) - x\| \le \|y - x\| \quad \forall x \in X .$$

We denote by $\mathcal{F}$ the class of all pseudo-projection mappings onto X. Clearly, $\pi_X \in \mathcal{F}$. That is, the pseudo-projection is weaker but it can be implemented in the case where h is essentially nonlinear; see [26] for more details.

Method (CRM). *Step 0 (initialization)*: Choose a point $x^0 \in X$, bounded positive sequences $\{\varepsilon_l\}$ and $\{\eta_l\}$, and a sequence of mappings $\{P_k\}$, where $P_k \in \mathcal{F}$. Also, choose numbers $\theta \in (0,1)$, and $\gamma \in (0,2)$. Set $k := 0$, $l := 1$.

Step 1 (auxiliary procedure):

Step 1.1: Choose q^0 from $Q(x^k)$, set $i := 0$, $p^i := q^i$, $w^{k,0} := x^k$.

Step 1.2: If $\|p^i\| \le \eta_l$, set $x^{k+1} := x^k$, $k := k+1$, $l := l+1$ and go to step 1. (*null step*)

Step 1.3: Set $w^{k,i+1} := w^{k,0} - \varepsilon_l p^i / \|p^i\|$, choose $q^{i+1} \in Q(w^{k,i+1})$. If $\langle q^{i+1}, p^i \rangle > \theta \|p^i\|^2$, then set $y^k := w^{k,i+1}$, $g^k := q^{i+1}$, and go to step 2. (*descent step*)

Step 1.4: Set $p^{i+1} := \mathrm{Nr}\,\mathrm{conv}\{p^i, q^{i+1}\}$, $i := i+1$ and go to step 1.2.

Step 2 (Main iteration): Set $\omega_k := \langle g^k, x^k - y^k \rangle$,

$$x^{k+1} := P_k[x^k - \gamma(\omega_k / \|g^k\|^2) g^k] ,$$

$k := k+1$ and go to step 1.

Here $\mathrm{Nr}S$ denotes the element of S nearest to the origin. We will call one increase of the index i an inner step, so that the number of inner steps gives the number of computations of elements from $Q(\cdot)$ at the corresponding points.

Theorem 7 *Let a sequence $\{x^k\}$ be generated by (CRM) and let $\{\varepsilon_l\}$ and $\{\eta_l\}$ satisfy the following relations:*

$$\{\varepsilon_l\} \searrow 0, \{\eta_l\} \searrow 0 . \tag{12}$$

Then:

(i) The number of inner steps at each iteration is finite.

(ii) It holds that

$$\lim_{k\to\infty} x^k = x^* \in X^* .$$

Given a starting point x^0 and a number $\delta > 0$, we define the complexity of the method, denoted by $N(\delta)$, as the total number of inner steps t which ensures finding a point $\bar{x} \in X$ such that

$$\|\bar{x} - x^*\| / \|x^0 - x^*\| \le \delta .$$

Therefore, since the computational expense per inner step can be evaluated for each problem under examination, this estimate in fact gives the total amount of work. We thus proceed to obtain an upper bound for $N(\delta)$.

Theorem 8 (Konnov [26], Theorem 2.3.3) *Suppose G is monotone and there exists $x^* \in X^*$ such that*

for every $x \in X$ and for every $g \in G(x)$,

$$\langle g, x - x^* \rangle \ge \mu \|x - x^*\| ,$$

for some $\mu > 0$. Let a sequence $\{x^k\}$ be generated by (CRM) where

$$\varepsilon_l = \nu^l \varepsilon', \eta_l = \eta', l = 0, 1, \ldots ; \quad \nu \in (0, 1) .$$

Then, there exist some constants $\bar{\varepsilon} > 0$ and $\bar{\eta} > 0$ such that

$$N(\delta) \le B_1 \nu^{-2} (\ln(B_0/\delta)/\ln \nu^{-1} + 1) ,$$

where $0 < B_0, B_1 < \infty$, whenever $0 < \varepsilon' \le \bar{\varepsilon}$ and $0 < \eta' \le \bar{\eta}$, B_0 and B_1 being independent of ν.

The assertion of Theorem 8 remains valid without the additional monotonicity assumption on G if $X = \mathbb{R}^n$. Thus, (CRM) attains a logarithmic complexity estimate, which corresponds to a linear rate of convergence with respect to inner steps. We can give a similar upper bound for $N(\delta)$ in the single-valued case.

Theorem 9 (Konnov [26], Theorem 2.3.4) *Suppose that $X = \mathbb{R}^n$ and that G is strongly monotone and Lipschitz continuous. Let a sequence $\{x^k\}$ be generated by (CRM), where*

$$\varepsilon_l = \nu^l \varepsilon', \eta_l = \nu^l \eta', l = 0, 1, \ldots ; \varepsilon' > 0, \eta' > 0; \quad \nu \in (0, 1) .$$

Then,

$$N(\delta) \le B_1 \nu^{-6} (\ln(B_0/\delta)/\ln \nu^{-1} + 1) ,$$

where $0 < B_0, B_1 < \infty$, B_0 and B_1 being independent of ν.

Combined Relaxation Method for Multivalued Inclusions To solve GVI (1), we can also apply (CRM) for finding stationary points of the mapping P defined as follows:

$$P(x) = \begin{cases} G(x) & \text{if } h(x) < 0 , \\ \mathrm{conv}\{G(x) \bigcup \partial h(x)\} & \text{if } h(x) = 0 , \\ \partial h(x) & \text{if } h(x) > 0 . \end{cases} \quad (13)$$

Such a method need not include (pseudo)projections and is based on the following observations [21,26].

We note P in (13) is a K-mapping. Next, GVI (1) is equivalent to the multivalued inclusion

$$0 \in P(x^*) . \quad (14)$$

We denote by S^* the solution set of problem (14). In order to apply (CRM) to this problem we have to show that its dual problem is solvable. Namely, let us consider the problem of finding a point x^* of $\mathbb{R}^n$ such that

$$\forall x \in \mathbb{R}^n, \ \forall p \in P(u), \ \langle p, x - x^* \rangle \ge 0 ,$$

which can be viewed as the dual problem of (14). We denote by S^d the solution set of this problem.

Theorem 10 (Konnov [26], Theorem 2.3.1 and Proposition 2.4.1) *It holds that*
(i) $X^* = S^*$,
(ii) $X^d = S^d$.

Therefore, we can apply (CRM) by replacing G, X, and P_k by P, $\mathbb{R}^n$, and I, respectively, to the multivalued inclusion (14) under the same blanket assumptions. We call this modification (CRMIS).

Theorem 11 *Let a sequence $\{x^k\}$ be generated by (CRMIS) and let $\{\varepsilon_l\}$ and $\{\eta_l\}$ satisfy (15). Then:*
(i) The number of inner steps at each iteration is finite.
(ii) It holds that

$$\lim_{k \to \infty} x^k = x^* \in S^* = X^* .$$

Iterative Methods for Generalized Complementarity Problems

It is well known that taking into account additional peculiarities of the problem under examination could yield more efficient solution methods in comparison with those in the general case. We intend to describe several recent results for certain classes of multivalued VIs.

Let us consider problem (1) where the feasible set X coincides with the nonnegative orthant $\mathbb{R}^n_+ = \{x \in \mathbb{R}^n \mid x_i \ge 0 \ \forall i = 1, \ldots, n\}$. Then it can be rewritten in the equivalent *generalized complementarity problem* (GCP) format:

$$x^* \ge 0 , \ \exists g^* \in G(x^*), g^* \ge 0, \ \langle g^*, x^* \rangle = 0 . \quad (15)$$

Owing to the special form of the constraint sets of these problems, their existence and uniqueness results of solutions can be based upon rather weak order monotonicity properties instead of the previous norm monotonicity ones [9,10,17]. We recall several order monotonicity properties of single-valued mappings.

Definition 5 A mapping $F: X \to \mathbb{R}^n$ is said to be

(a) a P_0-*mapping*, if for each pair of points $x', x'' \in X$ there exists an index i such that $x'_i \neq x''_i$ and

$$(x'_i - x''_i)(F_i(x') - F_i(x'')) \geq 0\,;$$

(b) a *P-mapping*, if for each pair of points $x', x'' \in X$ such that $x' \neq x''$ it holds that

$$\max_{1\leq i\leq n} (x'_i - x''_i)\left[F_i(x') - F_i(x'')\right] > 0\,;$$

(c) a *Z-mapping* if for each pair of points $x', x'' \in X$ such that $x' \geq x''$ it holds that $F_k(x') \leq F_k(x'')$ for each index k with $x'_k = x''_k$.

Clearly, each monotone (respectively, strictly monotone) mapping is a P_0-mapping (respectively, P-mapping).

One of the most useful and fruitful concepts is that of the Z-mapping (or off-diagonal antitone mapping). However, the creation of efficient solution methods and even the generalization of this concept for multivalued mappings meet considerable difficulties.

Following [29] and [32], we consider some kinds of multivalued Z-mappings and discuss their properties. For rather a general class of GCPs of form (15), we suggest an extension of the Jacobi algorithm and obtain its convergence to a solution, thus presenting an existence result.

Properties of Multivalued Z-Mappings

We present streamlined extensions of the above concepts for the multivalued case.

Definition 6 A multivalued mapping $G: X \to \Pi(\mathbb{R}^n)$ is said to be

(a) a P_0-*mapping*, if for each pair of points $x', x'' \in X$, and for each pair of vectors $g' \in G(x')$, $g'' \in G(x'')$ there exists an index i such that $x'_i \neq x''_i$ and

$$(x'_i - x''_i)(g'_i - g''_i) \geq 0\,;$$

(b) a *P-mapping*, if for each pair of points $x', x'' \in X$ such that $x' \neq x''$ and for each pair of vectors $g' \in G(x')$, $g'' \in G(x'')$ there exists an index i such that

$$(x'_i - x''_i)(g'_i - g''_i) > 0\,;$$

(c) a *Z-mapping* if for each pair of points $x', x'' \in X$ such that $x' \geq x''$, $x' \neq x''$ it holds that $g'_k \leq g''_k$ for all $g' \in G(x')$, $g'' \in G(x'')$ and for each index k such that $x'_k = x''_k$.

Note that the additional condition $x' \neq x''$ cannot be dropped in Definition 6c since otherwise the Z-mapping becomes single-valued. Hence, the above concept of the Z-mapping may appear too restrictive.

Definition 7 A mapping $G: \mathbb{R}^n \to \Pi(\mathbb{R}^n)$ is said to be

(a) *diagonal* if $G(x) = \prod_{i=1}^{n} G_i(x_i)$;

(b) *quasi-diagonal* if $G(x) = \prod_{i=1}^{n} G_i(x)$.

Clearly, (a)$\Longrightarrow$(b). Moreover, each single-valued mapping is quasi-diagonal. Next, observe that each diagonal single-valued mapping is Z, but this is not the case if it is multivalued; hence, various compositions of multivalued diagonal and Z-mappings may not possess the Z property as well.

We now present modified order monotonicity concepts of multivalued Z-mappings which enable us to remove these difficulties.

Definition 8 A mapping $G: X \to \Pi(\mathbb{R}^n)$ is said to be *an upper (a lower) Z-mapping* if for each pair of points $x', x'' \in D$ such that $x' \geq x''$ and for each $g' \in G(x')$ there exists $g'' \in G(x'')$ (respectively, for each $g'' \in G(x'')$ there exists $g' \in G(x')$) such that $g'_k \leq g''_k$ for every index k such that $x'_k = x''_k$.

Obviously, these concepts extend the similar one from Definition 6 and the condition $x' \neq x''$ is now unnecessary. They are also additive. Moreover, each diagonal mapping is both an upper and a lower Z-mapping.

Extended Jacobi Algorithm for Multivalued Mixed Complementarity Problems

Let us consider GCP (15), where $G: X \to \Pi(\mathbb{R}^n)$ is of the form

$$G(x) = \sum_{s=1}^{l} F^{(s)} \circ H^{(s)}(x)\,, \tag{16}$$

where $F^{(s)}: \mathbb{R}^n \to \Pi(\mathbb{R}^n)$ *is a quasi-diagonal, an upper Z- and a K-mapping on some rectangle containing* $H^{(s)}(X)$, $H^{(s)}: X \to \Pi(\mathbb{R}^n)$ *is a diagonal monotone*

K-mapping for each $s = 1, \dots, l$. Let us introduce the auxiliary set for GCP (15) and (16) as follows:

$$\mathbb{Q} = \{x \geq 0 \mid \exists g \in G(x), g \geq 0\} \ .$$

Given a vector $x \in \mathbb{R}^n$ and a number y_i, we set

$$(x_{-i}, y_i) = (x_1, \dots, x_{i-1}, y_i, x_{i+1}, \dots, x_n) \ .$$

Algorithm (Jacobi). Choose a point $\tilde{x} \in \mathbb{Q}$ and, beginning from the point $x^0 = \tilde{x}$, construct a sequence $\{x^k\}$ in conformity with the following rules.

At the kth iteration, $k = 0, 1, \dots$, we have a point $x^k \in \mathbb{Q}$ such that $x^k \leq x^0$ and that there exists $g^k \in \sum_{s=1}^{l} F^{(s)}(h^{(s),k})$ for some $h^{(s),k} \in H^{(s)}(x^k), s = 1, \dots, l$, such that $g^k \geq 0$.

For each separate index $i = 1, \dots, n$, we determine numbers $x_i^{k+1}, p_i^{(1)}, \dots, p_i^{(l)}$ such that

$$0 \leq x_i^{k+1} \leq x_i^k, \exists \tilde{g}_i^k \in \sum_{s=1}^{l} F_i^{(s)}(h_{-i}^{(s),k}, p_i^{(s)}), \tilde{g}_i^k \geq 0,$$
$$x_i^{k+1} \tilde{g}_i^k = 0 \ , \quad (17)$$

$$p_i^{(s)} \in H_i^{(s)}(x_i^{k+1}), p_i^{(s)} \leq h_i^{(s),k} \text{ for } s = 1, \dots, l \ , \quad (18)$$

with the help of the bisection procedure below. Afterwards, set $h^{(s),k+1} = p^{(s)}$ for $s = 1, \dots, l$ and go to the $(k+1)$th iteration.

Procedure (Bisection). It is applied when the indices k and i are fixed and consists of the following sequence of steps.

Step 1: If $g_i^k = 0$ or $x_i^k = 0$, set $x_i^{k+1} = x_i^k$, $\tilde{g}_i^k = g_i^k$, $p_i^{(s)} = h_i^{(s),k}$ for $s = 1, \dots, l$ and stop. Otherwise go to step 2.

Step 2: Choose $p_i^{(s)} \in H_i^{(s)}(0)$ for $s = 1, \dots, l$ and compute an element $\tilde{g}_i^k \in \sum_{s=1}^{l} F_i^{(s)}(h_{-i}^{(s),k}, p_i^{(s)})$. If $\tilde{g}_i^k \geq 0$, then set $x_i^{k+1} = 0$ and stop. Otherwise set $x_i' = 0$, $x_i'' = x_i^k$ and $\alpha_i^{(s)} = p_i^{(s)}, \beta_i^{(s)} = h_i^{(s),k}$ for $s = 1, \dots, l$.

Step 3: Generate a sequence of inscribed segments $[x_i', x_i'']$ contracting to a point z_i by choosing $y_i = \frac{1}{2}(x_i' + x_i'')$, computing $\tilde{\beta}_i^{(s)} \in H_i^{(s)}(y_i)$ for $s = 1, \dots, l$ and $\tilde{g}_i \in \sum_{s=1}^{l} F_i^{(s)}(h_{-i}^{(s),k}, \tilde{\beta}_i^{(s)})$, and setting $x_i'' = y_i$, $\beta_i^{(s)} = \tilde{\beta}_i^{(s)}$ for $s = 1, \dots, l$ if $\tilde{g}_i \geq 0$ and $x_i' = y_i, \alpha_i^{(s)} = \tilde{\beta}_i^{(s)}$ for $s = 1, \dots, l$ if $\tilde{g}_i < 0$.

Step 4: Set $x_i^{k+1} = z_i$ and compute numbers $p_i^{(s)} \in H_i^{(s)}(z_i)$ for $s = 1, \dots, l$ such that conditions (17) and (18) are satisfied.

We present a convergence result for the Jacobi algorithm.

Theorem 12 *Suppose that the set* $\mathbb{Q}$ *is nonempty. Then the Jacobi algorithm with the bisection procedure is well defined and generates a sequence* $\{x^k\}$ *converging to a solution* x^* *of GCP (15) and (16) such that* $0 \leq x^* \leq \tilde{x}$.

Clearly, the corresponding modification of the Gauss–Seidel algorithm will possess similar convergence properties. Note that the above theorem in fact contains also the existence result.

Corollary 1 *If the set* $\mathbb{Q}$ *is nonempty, then GCP (15) and (16) has a solution.*

It was also shown in [32] that the auxiliary set $\mathbb{Q}$ is a *meet semisublattice*, i. e., for each pair of points $x, y \in \mathbb{Q}$ it contains their minimal point (meet) $z = \min\{x, y\}$ with $z_i = \min\{x_i, y_i\}$ for $i = 1, \dots, n$; if (16) is replaced by

$$G(x) = F \circ H(x) + V(x) \ ,$$

$V\colon X \to \Pi(\mathbb{R}^n)$ *is a quasi-diagonal, an upper Z- and a K-mapping,* $H\colon X \to \Pi(\mathbb{R}^n)$ *is a diagonal monotone K-mapping, and* $F\colon \mathbb{R}^n \to \Pi(\mathbb{R}^n)$ *is a quasi-diagonal, an upper Z- and a K-mapping on a rectangle containing* $H(X)$. Hence, the set $\mathbb{Q}$ has the least element $\min \mathbb{Q}$ which is a solution of the GCP.

The above Jacobi algorithm can be extended to a more general class of problems. In fact, we can consider problem (1) where the feasible set X is defined as follows:

$$X = \{x \in \mathbb{R}^n \mid -\infty < a_i \leq x_i \leq b_i \leq +\infty \quad i = 1, \dots, n\} \ .$$

It is called the *generalized mixed complementarity problem* (GMCP) and can be also equivalently rewritten as follows: Find a point $x^* \in X$ such that

$$\exists g^* \in G(x^*), \quad g_i^* \begin{cases} \geq 0 & \text{if} \quad x_i^* = a_i, \\ = 0 & \text{if} \quad x_i^* \in (a_i, b_i) \ , \\ \leq 0 & \text{if} \quad x_i^* = b_i \ , \end{cases} \quad (19)$$
$$\text{for} \quad i = 1, \dots, n \ .$$

Then we should define the auxiliary set for the GMCP as follows

$$\mathbb{Q} = \{x \in X \mid \exists g \in G(x), x_i < b_i \Rightarrow g_i \geq 0 \quad \forall i = 1, \ldots, n\},$$

and all the above results remain true.

We can enhance the assertions of the theorems from Sects. "Regularization Methods" and "Proximal Point Method" for the regularization-type methods applied to GCP (15) or to GMCP (19) with order monotonicity (P_0) properties. In fact, if G is P_0, then the auxiliary mappings in (10) and (11) are P and the corresponding auxiliary problems will have a unique solution, thus strengthening assertions (i) of Theorem 3, (ii) of Theorem 4, and (i) of Theorem 6 [1,30,34].

Iterative Methods for MVIs

Let $Q\colon \mathbb{R}^n \to \mathbb{R}^n$ be a continuous single-valued mapping and $f\colon \mathbb{R}^n \to \mathbb{R}$ be a convex, proper and lower semicontinuous function. The *MVI problem* is the problem of finding a point $x^* \in X$ such that

$$\langle Q(x^*), x - x^* \rangle + f(x) - f(x^*) \geq 0 \quad \forall x \in X. \quad (20)$$

In this section, we denote by X^* the solution set of problem (20). Problem (20) was originally considered by Lescarret [37] and Browder [6] and was studied by many authors owing to its various applications. In the case $f \equiv 0$, it corresponds to the usual VI (2). If f is subdifferentiable, MVI (20) becomes equivalent to the problem of finding $x^* \in X$ such that

$$\exists h^* \in \partial f(x^*), \quad \langle Q(x^*) + h^*, x - x^* \rangle \geq 0 \quad \forall x \in X, \quad (21)$$

i. e., to GVI (1) with $G = Q + \partial f$, where ∂f denotes the subdifferential mapping of f. Also, GVI (21) is a particular case of the problem: Find $x^* \in X$ such that

$$\exists h^* \in H(x^*), \quad \langle Q(x^*) + h^*, x - x^* \rangle \geq 0 \quad \forall x \in X, \quad (22)$$

where $H\colon X \to \Pi(\mathbb{R}^n)$ is a monotone multivalued mapping. In turn, GVI (22) is a particular case of GVI (1) with $G = Q + H$.

In order to construct an efficient solution method for multivalued GVI (22) (or (21)) we can utilize the so-called splitting approach as a basis. In fact, if the mapping H in GVI (22) (respectively, ∂f in GVI (21)) is invertible rather easily, then one can apply the *forward–backward splitting method* which is due to Lions and Mercier [38] and consists in constructing a sequence $\{x^k\}$ as follows: $x^{k+1} \in X$ such that

$$\exists h^{k+1} \in H(x^{k+1}), \quad \langle Q(x^k) + \theta^{-1}(x^{k+1} - x^k) + h^{k+1}, y - x^{k+1} \rangle \geq 0 \quad \forall y \in X, \quad (23)$$

where $\theta > 0$, i. e., each iteration is explicit with respect to Q and implicit with respect to H. Method (23) is clearly simpler than the general proximal point method with respect to $Q + H$, but it requires strengthened monotonicity (co-coercivity) assumptions on Q for convergence [13]. The combined averaging and splitting method (see [41]) allows for establishing convergence if Q is only monotone, but it also utilizes the divergent series step-size rule.

Descent Methods for MVIs

In order to enhance the step-size rule we can utilize the descent approach with respect to some artificial merit (or otherwise, gap) function, which enables one to convert the MVI problem into an optimization problem.

Gap Function Approach for MVIs The simplest regularized gap function can be defined as follows:

$$\varphi_\alpha(x) = \max_{y \in X} \Phi_\alpha(x, y),$$

where

$$\Phi_\alpha(x, y) = \langle G(x), x - y \rangle - 0.5\alpha \|x - y\|^2 + f(x) - f(y), \quad \alpha > 0.$$

The function $\Phi_\alpha(x, \cdot)$ is strongly concave; hence, there exists the unique element $y_\alpha(x) \in X$ such that $\Phi_\alpha(x, y_\alpha(x)) = \varphi_\alpha(x)$. Observe that the computation of $y_\alpha(x)$ is equivalent to an iteration of the forward–backward splitting method applied to MVI (20).

From the definition we have that the following properties are equivalent:

(a) $\varphi_\alpha(x) = 0$,
(b) $x = y_\alpha(x)$,
(c) x is a solution of problem (20).
i. e., ϕ_α is a gap function for MVI (20) and MVI (20) is equivalent to the optimization problem

$$\min_{x\in X} \to \varphi_\alpha(x)\,. \tag{24}$$

Despite the fact that ϕ_α is nondifferentiable and nonconvex, we can describe descent methods with respect to ϕ_α without computation of its derivatives. Such a descent method with exact linesearch was proposed by Patriksson [42]. Moreover, it generates a sequence which converges to a unique solution of MVI (20) if the mapping Q is strongly monotone. At the same time, inexact linesearch procedures are more suitable for implementation. For this reason we describe a descent method with an inexact Armijo-type linesearch procedure.

Method (DIG). Choose a point $x^0 \in X$ and numbers $\alpha > 0$, $\beta \in (0,1)$, and $\gamma \in (0,1)$.

At the kth iteration, $k = 0, 1, \ldots$, we have a point $x^k \in X$, compute $y_\alpha(x^k)$ and set $d^k := y_\alpha(x^k) - x^k$. If $d^k = 0$, stop. Otherwise, we find m as the smallest nonnegative integer such that

$$\varphi_\alpha(x^k + \gamma^m d^k) \le \varphi_\alpha(x^k) - \beta\gamma^m \|d^k\|^2\,,$$

set $\lambda_k := \gamma^m$, $x^{k+1} := x^k + \lambda_k d^k$ and go to the next iteration.

Theorem 13 *If the mapping Q is continuously differentiable and strongly monotone with constant τ, and $\beta < \tau$, (DIG) generates a sequence $\{x^k\}$ which converges to a unique solution of MVI (20).*

***D*-Gap Function Approach for MVIs** For the usual VI (2), Peng [43] introduced the so-called D-gap function, which allows one to convert it into an unconstrained optimization problem. Following this approach, Konnov [23] proposed the D-gap function for MVI (20) and showed that, unlike the usual gap functions, it becomes differentiable if the mapping Q is so, regardless of the properties of the function f. Hence, we can apply the rapidly convergent algorithms in order to find a solution of the initial MVI.

The D-gap function is defined as follows:

$$\psi_{\alpha\beta}(x) = \varphi_\alpha(x) - \varphi_\beta(x)\,,$$

where $0 < \alpha < \beta$. It follows that MVI (20) is equivalent to the unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} \to \psi_{\alpha\beta}(x)\,.$$

Next, if Q a continuously differentiable, so is $\psi_{\alpha\beta}$ and

$$\nabla\psi_{\alpha\beta}(x) = \nabla Q(x)[y_\beta(x) - y_\alpha(x)] + \beta(x - y_\beta(x)) - \alpha(x - y_\alpha(x))\,.$$

If $\nabla Q(x)$ is positive definite on $\mathbb{R}^n$, then MVI (20) is equivalent to the equation

$$\nabla\psi_{\alpha\beta}(x) = 0\,.$$

Utilizing the above properties, we can describe a descent method with respect to $\psi_{\alpha\beta}$ without computation of its derivatives.

Method (DIDG). Choose a point $x^0 \in \mathbb{R}^n$ and numbers $\beta > \alpha > 0$, $\mu > 0$, $\gamma \in (0,1)$, $\theta > 0$.

At the kth iteration, $k = 0, 1, \ldots$, we have a point x^k, compute $y_\alpha(x^k)$ and $y_\beta(x^k)$, set

$$r(x^k) := y_\alpha(x^k) - y_\beta(x^k)\,,$$
$$s(x^k) := \alpha(x^k - y_\alpha(x^k)) - \beta(x^k - y_\beta(x^k))$$

and $d^k := r(x^k) + \mu s(x^k)$. If $d^k = 0$, stop. Otherwise, we compute m as the smallest nonnegative integer such that

$$\psi_{\alpha\beta}(x^k + \gamma^m d^k) \le \psi_{\alpha\beta}(x^k) - \gamma^m\theta(\|r(x^k)\| + \mu\|s(x^k)\|)^2$$

set $\lambda_k := \gamma^m$, $x^{k+1} := x^k + \lambda_k d^k$ and go to the next iteration.

If the mapping Q is strongly monotone, (DIDG) also generates a sequence $\{x^k\}$ which converges to a unique solution of MVI (20).

In [25], this approach was extended for MVI (20) with order monotonicity (P) properties. In the case when the mapping Q is only monotone (or P_0), but not strongly monotone, the above descent methods can be combined with either regularization or proximal point methods, such that their auxiliary subproblems are solved approximately.

Combined Relaxation Methods for MVIs

We describe a combined relaxation method for solving monotone MVI (20) which utilizes a similar iteration of

the forward–backward splitting method as an auxiliary procedure [24,26]. In this subsection we suppose that X is a nonempty, closed and convex subset of the space $\mathbb{R}^n$, $Q\colon \mathbb{R}^n \to \mathbb{R}^n$ is a continuous monotone mapping and $f\colon \mathbb{R}^n \to \mathbb{R}$ is a convex and subdifferentiable function. For the sake of clarity, we describe a simplified version of the method.

Method (CRS). *Step 0 (initialization)*: Choose a point $x^0 \in \mathbb{R}^n$ and a sequence of $n \times n$ symmetric matrices $\{A_k\}$ such that

$$\tau'\|p\|^2 \le \langle A_k p, p\rangle \le \tau''\|p\|^2 \quad \forall p \in \mathbb{R}^n, \quad 0 < \tau' \le \tau'' < \infty\,. \tag{25}$$

Choose numbers $\alpha \in (0,1)$, $\beta \in (0,1)$, and $\gamma \in (0,2)$. Set $k := 0$.

Step 1 (auxiliary procedure):

Step 1.1: Determine m as the smallest nonnegative integer such that

$$\langle Q(x^k) - Q(z^{k,m}), x^k - z^{k,m}\rangle \le (1-\alpha)\beta^{-m}\langle A_k(z^{k,m} - x^k), z^{k,m} - x^k\rangle\,,$$

where $z^{k,m}$ is a solution of the auxiliary problem: Find $z^{k,m} \in X$ such that

$$\langle Q(x^k) + \beta^{-m}A_k(z^{k,m} - x^k), x - z^{k,m}\rangle + f(x) - f(z^{k,m}) \ge 0 \quad \forall x \in X\,. \tag{26}$$

Step 1.2: Set $\theta_k := \beta^m$, $y^k := z^{k,m}$. If $x^k = y^k$, stop. Otherwise set

$$g^k := Q(y^k) - Q(x^k) - \theta_k^{-1}A_k(y^k - x^k)\,,$$
$$\omega_k := \langle g^k, x^k - y^k\rangle\,.$$

Step 2 (main iteration): Set

$$x^{k+1} := x^k - \gamma\omega_k g^k/\|g^k\|^2\,,$$

$k := k+1$ and go to step 1.

Obviously, there exist a number of rules for choosing the sequence $\{A_k\}$ satisfying condition (25). The simplest is $A_k \equiv I$, which yields the usual forward–backward splitting iteration.

Theorem 14 *Let a sequence $\{x^k\}$ be constructed by (CRS). If the method terminates at the kth iteration, then $x^k \in X^*$. Otherwise, if $\{x^k\}$ is infinite, then*

$$\lim_{k\to\infty} x^k = x^* \in X^*\,.$$

Note that problem (20) has a unique solution if Q is strongly monotone. Then (CRS) converges at least linearly.

Theorem 15 *Suppose that Q is strongly monotone. If (CRS) generates an infinite sequence $\{x^k\}$, then $\{x^k\}$ converges to a solution of problem (20) at a linear rate.*

This approach admits various extensions and modifications. For instance, we can adjust the previous combined relaxation method to problem (20) with the function f having the form

$$f(x) = \max_{i=1,\dots,m} f_i(x)\,, \tag{27}$$

where $f_i\colon \mathbb{R}^n \to \mathbb{R}$, $i = 1,\dots,m$ are continuously differentiable convex functions [27]. In this method, the function f in (26) is replaced by its lower approximation:

$$\tau_k(x) = \max_{i=1,\dots,m}\left\{f_i(x^k) + \langle f_i'(x^k), x - x^k\rangle\right\}\,.$$

Hence, if the feasible set X is defined by affine functions, then the auxiliary problem is equivalent to a convex quadratic programming problem and can be solved exactly by one of the finite algorithms. At the same time, the modified method possesses the same convergence properties.

The combined relaxation methods described above based on the auxiliary splitting iterations can be applied to nonmonotone multivalued GVIs of form (22) and to nonmonotone mixed-equilibrium problems; see [24,31,35] for more details.

References

1. Allevi E, Gnudi A, Konnov IV (2006) The proximal point method for nonmonotone variational inequalities. Math Meth Oper Res 63:553–565
2. Bakushinskii AB, Polyak BT (1974) On the solution of variational inequalities. Soviet Math Doklady 15:1705–1710
3. Bianchi M, Hadjisavvas N, Schaible S (2004) Minimal coercivity conditions and exceptional families of elements in quasimonotone variational inequalities. J Optim Theory Appl 122:1–17
4. Billups SC, Ferris MC (1997) QPCOMP: A quadratic programming based solver for mixed complementarity problems. J Optim Theory Appl 76:533–562
5. Browder FE (1965) Multivalued monotone nonlinear mappings and duality mappings in Banach spaces. Trans Am Math Soc 71:780–785

6. Browder FE (1966) On the unification of the calculus of variations and the theory of monotone nonlinear operators in Banach spaces. Proc Natl Acad Sci USA 56:419–425
7. Browder FE (1966) Existence and approximation of solutions of nonlinear variational inequalities. Proc Natl Acad Sci USA 56:1080–1086
8. Bruck R (1977) On weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in Hilbert space. J Math Anal Appl 61:159–164
9. Cottle RW, Pang JS, Stone RE (1992) The Linear Complementarity Problem. Academic Press, Boston
10. Facchinei F, Pang J-S (2003) Finite-Dimensional Variational Inequalities and Complementarity Problems. Springer, Berlin (two volumes)
11. Fang SC, Petersen EL (1982) Generalized variational inequalities. J Optim Theory Appl 38:363–383
12. El Farouq N (2001) Pseudomonotone variational inequalities: convergence of proximal methods. J Optim Theory Appl 109:311–326
13. Gabay D (1983) Application of the method of multipliers to variational inequalities. In: Fortin M, Glowinski R (eds) Augmented Lagrangian Methods: Application to the Numerical Solution of Boundary-Value Problems. North-Holland, Amsterdam, pp 299–331
14. Gol'shtein EG, Nemirovskii AS, Nesterov YE (1995) Level method, its extensions and applications. Ekon Matem Metody 31:164–181
15. Gol'shtein EG, Tret'yakov NV (1989) Augmented Lagrange Functions, Nauka, Moscow, (Engl. transl. in Modified Lagrangians and Monotone Maps in Optimization. John Wiley and Sons, New York (1996))
16. Hadjisavvas N, Komlósi S, Schaible S (eds) (2005) Handbook of Generalized Convexity and Generalized Monotonicity. Nonconvex Optimization and Applications, 76. Springer, New York
17. Isac G (1992) Complementarity Problems. Springer, Berlin
18. Kiwiel KC (1995) Proximal level bundle methods for convex nondifferentiale optimization, saddle point problems and variational inequalities. Math Program 69:89–109
19. Konnov IV (1993) Combined relaxation methods for finding equilibrium points and solving related problems. Russian Math (Iz. VUZ) 37(2):44–51
20. Konnov IV (1993) On combined relaxation methods' convergence rates. Russian Math (Iz. VUZ) 37(12):89–92
21. Konnov IV (1996) A general approach to finding stationary points and the solution of related problems. Comput Math Math Phys 36:585–593
22. Konnov IV (1998) A combined relaxation method for variational inequalities with nonlinear constraints. Math Program 80:239–252
23. Konnov IV (1999) On a class of D-gap functions for mixed variational inequalities. Russian Math (Iz. VUZ) 43(12):60–64
24. Konnov IV (1999) A combined method for variational inequalities with monotone operators. Comp Math Math Phys 39:1051–1056
25. Konnov IV (2000) Properties of gap functions for mixed variational inequalities. Siberian J Numer Math 3:259–270
26. Konnov IV (2001) Combined Relaxation Methods for Variational Inequalities, Lecture Notes in Economics and Mathematical Systems, 495. Springer, Berlin
27. Konnov IV (2002) A combined relaxation method for a class of nonlinear variational inequalities. Optimization 51:127–143
28. Konnov IV (2003) Application of the proximal point method to nonmonotone equilibrium problems. J Optim Theory Appl 119:317–333
29. Konnov IV (2005) An extension of the Jacobi algorithm for the complementary problem in the presence of multivalence. Comp Math Math Phys 45:1127–1132
30. Konnov IV (2006) On the convergence of a regularization method for variational inequalities. Comp Math Math Phys 46:541–547
31. Konnov IV (2006) Splitting-type method for systems of variational inequalities. Comput Oper Res 33:520–534
32. Konnov IV (2007) An extension of the Jacobi algorithm for multi-valued mixed complementarity problems. Optimization 56:399–416
33. Konnov IV (2007) Regularization method for nonmonotone equilibrium problems. J Nonlin Conv Anal (accepted)
34. Konnov IV, Ali MSS, Mazurkevich EO (2006) Regularization of nonmonotone variational inequalities. Appl Math Optim 53:311–330
35. Konnov IV, Schaible S, Yao JC (2005) Combined relaxation method for equilibrium problems. J Optim Theory Appl 126:309–322
36. Lemaréchal C, Nemirovskii A, Nesterov Y (1995) New variants of bundle methods. Math Program 69:111–147
37. Lescarret C (1965) Cas d'addition des applications monotones maximales dan un espace de Hilbert. Compt Rend Acad Sci 261:1160–1163
38. Lions PL, Mercier B (1979) Splitting algorithms for the sum of two monotone operators. SIAM J Numer Anal 16:964–979
39. Martinet B (1970) Regularization d'inéquations variationnelles par approximations successives. Rev Fr Inform Rech Opér 4:154–159
40. Nemirovskii AS (1981) Effective iterative methods for solving equations with monotone operators. Ekon Matem Metody (Matecon) 17:344–359
41. Passty GB (1979) Ergodic convergence to zero of the sum of two monotone operators in Hilbert space. J Math Anal Appl 72:383–390
42. Patriksson M (1997) Merit functions and descent algorithms for a class of variational inequality problems. Optimization 41:37–55

43. Peng J-M (1997) Equivalence of variational inequality problems to unconstrained minimization. Math Program 78:347–355
44. Rockafellar RT (1976) Monotone operators and the proximal point algorithm. SIAM J Contr Optim 14:877–898
45. Shor NZ (1977) Cut-off method with space dilation in convex programming problems. Cybernetics 13:94–96
46. Tikhonov AN (1963) On the solution of ill-posed problems and regularization method. Dokl Akad Nauk SSSR 151:501–504
47. Yudin DB, Nemirovskii AS (1977) Informational complexity and efficient methods for the solution of convex extremal problems, I, II. Matecon 13:25–45, 550–559

Solving Hemivariational Inequalities by Nonsmooth Optimization Methods

MARKKU MIETTINEN, MARKO M. MÄKELÄ
University Jyväskylä, Jyväskylä, Finland

MSC2000: 49J40, 49J52, 90C30, 65K05

Article Outline

Keywords
Discrete Problem
Numerical Realization
Nonsmooth Optimization Methods
Numerical Experience
See also
References

Keywords

Hemivariational inequalities; Substationary point; Nonsmooth optimization; Nonconvex optimization; Bundle methods

Hemivariational inequalities are generalizations of variational inequalities. They are used to model mathematically problems from mechanics, engineering and economics whenever nonconvex energy functionals are present. Typical applications, e. g. in mechanics are contact problems of elastic bodies in which nonmonotone friction laws or adhesive contact laws are involved or delamination of adhesively connected plates. The basic hemivariational inequality is of the following form: Find $u \in K$ (K is a convex subset of a Hilbert space X) such that

$$a(u, v-u) + \int_\Omega j^\circ(u; v-u)\,\mathrm{d}x \geq \langle F, v-u\rangle\,, \qquad \forall v \in K\,, \quad (1)$$

where $a\colon X \times X \to \mathbf{R}$ is a bilinear form, Ω a subset of $\mathbf{R}^N$, $j^\circ(\cdot;\cdot)$ the *Clarke generalized directional derivative* of the locally Lipschitz function $j\colon \mathbf{R}^M \to \mathbf{R}$ defined in [3] by

$$j^\circ(\xi;\eta) = \limsup_{\substack{\xi' \to \xi,\\ t \downarrow 0}} \frac{j(\xi' + t\eta) - j(\xi')}{t},$$

$\langle\cdot,\cdot\rangle$ the duality pairing between X and X^* (X^* is the dual space of X) and $F \in X^*$. If the function j is convex then the hemivariational inequality (1) is reduced to the classical variational inequality: Find $u \in K$ such that

$$a(u, v-u) + \int_\Omega j(v)\,\mathrm{d}x - \int_\Omega j(u)\,\mathrm{d}x \geq \langle F, v-u\rangle\,, \qquad \forall v \in K\,.$$

The concept of the hemivariational inequality was introduced by P.D. Panagiotopoulos. The mathematical theory and the applications are studied in [14,15].

Discrete Problem

For the discretization of the hemivariational inequalities it is used the finite element technique [2]. By means of it the following fully discrete problem is formulated: Find $u \in K$ such that

$$u^\top A(v-u) + \sum_{i \in I} c_i\, j^\circ(u_i; v_i - u_i) \geq F^\top (v-u), \qquad \forall v \in K\,, \quad (2)$$

where A is the stiffness matrix, F the load vector, c_i the coefficients of the appropriate numerical integration formula, K a convex subset of the finite element space $X_h \subset X$, h the discretization parameter connected to the mesh size of the triangulation of X_h and I the set of the components of u for which the function j has an effect. The following stability and convergence result has been proved for the above approximation scheme [9,10]:

Theorem 1 *The problems (2) are solvable for every h. Further, their solutions converge in subsequences to the solutions of the continuous problem (1).*

Due to the nonconvex character of the function j the solutions of the continuous problem (1) and the discrete problems (2) are not in general unique. This also explains that in the above result the convergence is guaranteed only for subsequences and it has not been proved any convergence rate estimate.

Numerical Realization

Because of the nonsmoothness and the nonconvexity of j the numerical realization of (2) is a challenging problem. There are different approaches for that (see, e. g. [15]). The most obvious one is to regularize the nonsmoothness and use methods for smooth problems. The other possibility is to approximate (2) by a convex, possibly nonsmooth, problem and apply numerical methods for the classical variational inequalities. Both approaches are typically iterative methods: in the former one the problem is solved with many regularization parameters and in the latter one the convex approximation is updated in every iteration step. In order to solve (2) directly in its original form one can use *nonsmooth nonconvex optimization* methods.

Next it is explained in detail how the discrete hemivariational inequality (2) can be transformed to a nonsmooth optimization problem. The following concepts from nonsmooth analysis are needed [3]:

- Suppose that $f\colon \mathbf{R}^M \to \mathbf{R}$ is locally Lipschitz continuous. Then ξ^* is called a *substationary point* of f on K if

$$0 \in \partial f(\xi^*) + N_K(\xi^*),$$

where $N_K(\xi)$ is the *normal cone* of K at ξ, defined by

$$N_K(\xi) = \mathrm{cl}\left\{\bigcup_{\lambda\geq 0} \lambda \partial d_K(\xi)\right\},$$

d_K the distance function of K, and $\partial f(\xi)$ is the *Clarke subdifferential*, defined by

$$\partial f(\xi) = \left\{\eta \in \mathbb{R}^M\colon\ f^\circ(\xi;\zeta) \geq \eta^\top \zeta, \quad \forall \zeta \in \mathbb{R}^M\right\}.$$

- The function $f\colon \mathbf{R}^M \to \mathbf{R}$ is said to be *upper semismooth* if for any $\xi \in \mathbf{R}^M$, $\zeta \in \mathbf{R}^M$ and sequences $\{\eta_i\} \subset \mathbf{R}^M$ and $\{t_i\} \subset (0, \infty)$ satisfying $\eta_i \in \partial f(\xi + t_i\zeta)$ and $t_i \downarrow 0$, one has

$$\limsup_{i\to\infty} \eta_i^\top \zeta \geq \liminf_{i\to\infty} \frac{f(\xi + t_i\zeta) - f(\xi)}{t_i}.$$

In the sequel it is assumed that the stiffness matrix A is symmetric. Then the following discrete energy function can be defined:

$$f(u) = \frac{1}{2}u^\top Au + \sum_{i\in I} c_i j(u_i) - F^\top u.$$

And, consequently, the following optimization problem

$$\min_{u\in K} f(u) \tag{3}$$

is formulated.

The main question is now what is the relation between the optimization problem (3) and the inequality problem (2). Under reasonable assumptions, which are generally satisfied for real applications, the subdifferential of f is equal to

$$\partial f(u) = Au + \sum_{i\in I} c_i \partial j(u_i) - F.$$

Then from the definition of the subdifferential and the upper semismoothness it follows the result (see the proofs in [2,3]):

Theorem 2 *Every substationary point of f on K is a solution of the discrete hemivariational inequality (2). Moreover, the functional f is upper semismooth.*

This result gives the theoretical basis and the motivation for the use of nonsmooth optimization methods for the numerical solution of hemivariational inequalities. In what follows optimization methods are introduced for nonsmooth nonconvex functionals which are convergent under the condition that the functional is upper semismooth. Furthermore, some observations are presented of the numerical tests being performed in [11,12].

Nonsmooth Optimization Methods

The methods for solving the nonsmooth optimization problem (3) can be divided into two main classes: *subgradient methods* and *bundle methods*. The basic idea behind the subgradient methods is to generalize the smooth methods by replacing the gradient by an arbitrary subgradient (see [17]). Due to this simple structure they are widely used, but suffer from some theoretical and numerical drawbacks.

Bundle methods have their origin in cutting planes and can be stated, at the moment, the most efficient

and promising methods in nonsmooth optimization (see [13]). The aim is to produce a sequence $\{u_k\}_{k=1}^{\infty} \subset \mathbf{R}^N$ converging to a local minimum of (3) being also a substationary point of f on K. Suppose that in addition to the current iteration point u_k there exist some trial points $y_j \in \mathbf{R}^N$ (from past iterations) and subgradients $g_j \in \partial f(y_j)$ for $j \in J_k$, where the index set J_k is a nonempty subset of $\{1, \ldots, k\}$.

The idea behind the bundle methods is to approximate the objective function below by a piecewise linear function, in other words, f is replaced by so called *cutting plane model*

$$\widehat{f}_k(u) = \max_{j \in J_k} \{f(y_j) + g_j^\top (u - y_j)\}, \tag{4}$$

which equivalently can be written in the form

$$\widehat{f}_k(u) = \max_{j \in J_k} \{f(u_k) + g_j^\top (u - u_k) - \alpha_j^k\},$$

with the *linearization error*

$$\alpha_j^k = f(u_k) - f(y_j) - g_j^\top (u_k - y_j). \tag{5}$$

If f is convex, then $\widehat{f}_k(u) \le f(u)$ for all $u \in \mathbf{R}^N$ and $\alpha_j^k \ge 0$ for all $\in J_k$. In other words, the cutting plane model $\widehat{f}_k$ is an under estimate for f and the nonnegative linearization error α_j^k measures how good an approximation the model is to the original problem. In the nonconvex case these facts are not valid anymore and thus the linearization error (5) is replaced by so called *subgradient locality measure* (cf. [5])

$$\beta_j^k = \max \left\{ \left| \alpha_j^k \right|, \gamma \left| u_k - y_j \right|^2 \right\}, \tag{6}$$

where $\gamma \ge 0$ ($\gamma = 0$ if f is convex). Then obviously $\min_{u \in K} \widehat{f}_k(u) \le f(u_k)$ and $\beta_j^k \ge 0$ for all $j \in J_k$. The search direction is then calculated by

$$d_k = \arg \min_{u_k + d \in K} \left\{ \widehat{f}_k(u_k + d) + \frac{1}{2} d^\top M_k d \right\}. \tag{7}$$

The role of the stabilizing term $\frac{1}{2} d^\top M_k d$ is to guarantee the existence of the solution d_k and keep the approximation local enough. The $n \times n$ matrix M_k is intended to accumulate some second order information about the curvature of f around u_k.

The different bundle methods deviate mostly in the choice of M_k. Roughly speaking, the following methods can be distinguish.

- *Cutting plane method* [4] with $M_k \equiv 0$.
- *Conjugate subgradient method* [18] with $M_k \equiv I$ and $\beta_j^k \equiv 0$.
- *ε-steepest descent* [7] and *generalized cutting plane method* [5] with $M_k \equiv I$.
- *Bundle trust region* [16] and *proximal bundle method* [6] with $M_k = \lambda_k I$.
- *Variable metric bundle method* [1] with M_k as a full matrix.

Although the more advanced bundle methods try to accumulate the second order information, they are based on first order (sub)gradient information and thus have to considered as first order methods. The 'real' second order method, called *bundle-Newton method*, was derived in [8]. Instead of piecewise linear cutting pane model (4) a quadratic model was introduced in the form

$$\widetilde{f}_k(u) = \max_{j \in J_k} \left\{ f(y_j) + g_j^\top (u - y_j) + \frac{1}{2} \varrho_j (u - y_j)^\top M_j (u - y_j) \right\},$$

where $M_j \approx \nabla^2 f(y_j)$. The search direction finding problem (7) is then replaced by the problem

$$d_k = \arg \min_{u_k + d \in K} \{\widetilde{f}_k(u_k + d)\}. \tag{8}$$

Next the problem of determining the stepsize into search direction d_k is considered. Assume that $m_L \in (0, 1/2)$, $m_R \in (m_L, 1)$ and $\overline{t} \in (0, 1]$ are some fixed line search parameters. First search for the largest number $t_L^k \in [0, 1]$ such that $t_L^k \ge \overline{t}$ and

$$f(u_k + t_L^k d_k) \le f(u_k) + m_L t_L^k v_k, \tag{9}$$

where v_k is the predicted amount of descent

$$v_k = \widehat{f}_k(u_k + d_k) - f(u_k) < 0.$$

If such a parameter exists take a *long serious step*

$$u_{k+1} = u_k + t_L^k d_k \quad \text{and} \quad y_{k+1} = u_{k+1}.$$

Otherwise, if (9) holds but $0 < t_L^k < \overline{t}$ then take a *short serious step*

$$u_{k+1} = u_k + t_L^k d_k \quad \text{and} \quad y_{k+1} = u_k + t_R^k d_k$$

and if $t_L^k = 0$ take a *null step*

$$u_{k+1} = u_k \quad \text{and} \quad y_{k+1} = u_k + t_R^k d_k,$$

where $t_R^k > t_L^k$ is such that

$$-\beta_{k+1}^{k+1} + g_{k+1}^\top d_k \geq m_R v_k. \tag{10}$$

In short serious steps and null steps there exists discontinuity in the gradient of f. Then the requirement (10) ensures that u_k and y_{k+1} lie on the opposite sides of this discontinuity and the new subgradient $g_{k+1} \in \partial f(y_{k+1})$ will force a remarkable modification of the next search direction finding problem. The iteration is terminated if $|v_k|$ is small enough.

The pseudocode of general bundle method is the following:

```
PROCEDURE bundle method()
InputInstance();
Generate    an initial solution u_k;
Initialize  the bundle J_k and v_k;
Set         k = 1;
DO          | v_k | ≥ ε
            Generate the search direction d_k;
            Find stepsizes t_L^k and t_R^k;
            Update u_k and J_k;
            Set k = k + 1;
            Evaluate f(u_k) and g_k ∈ ∂f(u_k);
OD;
RETURN      (final solution u_k)
END bundle method;
```

Numerical Experience

The numerical tests in [12] indicate the applicability of bundle methods for hemivariational inequlities. Especially the second order bundle-Newton method based on the piecewise quadratic model works very reliable and efficiently way. This is natural, since the optimization problem arising from hemivariational inequalities has a dominated quadratic part. The most promising feature of bundle-Newton method discovered was the independence of the iteration number and function evaluations from the dimension of the problem even in the large scale case.

See also

- Composite Nonsmooth Optimization
- Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- Hemivariational Inequalities: Applications in Mechanics
- Hemivariational Inequalities: Eigenvalue Problems
- Hemivariational Inequalities: Static Problems
- Nonconvex Energy Functions: Hemivariational Inequalities
- Nonconvex-nonsmooth Calculus of Variations
- Nonsmooth and Smoothing Methods for Nonlinear Complementarity Problems and Variational Inequalities
- Quasidifferentiable Optimization
- Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- Quasidifferentiable Optimization: Algorithms for QD Functions
- Quasidifferentiable Optimization: Applications
- Quasidifferentiable Optimization: Applications to Thermoelasticity
- Quasidifferentiable Optimization: Calculus of Quasidifferentials
- Quasidifferentiable Optimization: Codifferentiable Functions
- Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- Quasidifferentiable Optimization: Exact Penalty Methods
- Quasidifferentiable Optimization: Optimality Conditions
- Quasidifferentiable Optimization: Stability of Dynamic Systems
- Quasidifferentiable Optimization: Variational Formulations
- Quasivariational Inequalities
- Sensitivity Analysis of Variational Inequality Problems
- Variational Inequalities
- Variational Inequalities: F. E. Approach
- Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- Variational Inequalities: Projected Dynamical System
- Variational Principles

References

1. Bonnans JF, Gilbert JC, Lemarèchal C, Sagastizàbal C (1995) A family of variable metric proximal methods. Math Program 68:15–47
2. Ciarlet PG (1978) The finite element method for elliptic problems. North-Holland, Amsterdam
3. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
4. Kelley JE (1960) The cutting plane method for solving convex programs. SIAM J 8:703–712
5. Kiwiel KC (1985) Methods of descent for nondifferentiable optimization. Springer, Berlin
6. Kiwiel KC (1990) Proximity control in bundle methods for convex nondifferentiable optimization. Math Program 46:105–122
7. Lemarèchal C, Strodiot J-J, Bihain A (1981) On a bundle algorithm for nonsmooth optimization. In: Mangasarian OL, Mayer RR, Robinson SM (eds) Nonlinear Programming. Acad. Press, New York, pp 245–281
8. Lukšan L, Vlček J (1995) A bundle-Newton method for nonsmooth unconstrained minimization. Techn. Report Inst. Computer Sci. Acad. Sci. Czech Republic 654
9. Miettinen M, Haslinger J (1995) Approximation of nonmonotone multivalued differential inclusions. IMA J Numer Anal 15:475–503
10. Miettinen M, Haslinger J (1997) Finite element approximation of vector-valued hemivariational inequalities. J Global Optim 10:17–35
11. Miettinen M, Mäkelä MM, Haslinger J (1995) On numerical solution of hemivariational inequalities by nonsmooth optimization methods. J Global Optim 6:401–425
12. Mäkelä MM, Miettinen M, Lukšan L, Vlček J (1999) Comparing nonsmooth nonconvex bundle methods in solving hemivariational inequalities. J Global Optim 14:117–135
13. Mäkelä MM, Neittaanmäki P (1992) Nonsmooth optimization: Analysis and algorithms with applications to optimal control. World Sci., Singapore
14. Naniewicz Z, Panagiotopoulos PD (1995) Mathematical theory of hemivariational inequalities and applications. M. Dekker, New York
15. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Berlin
16. Schramm H, Zowe J (1992) A version of the bundle idea for minimizing a nonsmooth functions: Conceptual idea, convergence analysis, numerical results. SIAM J Optim 2:121–152
17. Shor NZ (1985) Minimization methods for non-differentiable functions. Springer, Berlin
18. Wolfe P (1975) A method of conjugate subgradients for minimizing nondifferentiable functions. Math Program Stud 3:145–173

Solving Large Scale and Sparse Semidefinite Programs

YINYU YE
Department Management Sci., University Iowa, Iowa City, USA

MSC2000: 90C25, 90C30

Article Outline

Keywords
See also
References

Keywords

Semidefinite programming; Sparse matrix; Primal-dual; Potential function; Potential reduction

The standard semidefinite program has the form:

$$\text{(SDP)} \quad \begin{cases} \min & C \bullet X \\ \text{s.t.} & A_i \bullet X = b_i, \\ & i = 1, \ldots, m, \\ & X \succeq 0, \end{cases}$$

where the given matrices $A_i \in \mathbf{R}^{n\times n}$ and $C \in \mathbf{R}^{n\times n}$ are symmetric, $b \in \mathbf{R}^m$, and unknown $X \in \mathbf{R}^{n\times n}$ is also symmetric. Furthermore, $C \bullet X = \operatorname{tr} C^{\mathsf{T}} X = \sum_{jk} C_{jk} X_{jk}$, and $X \succcurlyeq 0$ means that X is positive semidefinite. In most applications, $A_i = a_i a_i^{\top}$, $a_i \in \mathbf{R}^n$, is a *rank-one matrix* and C is sparse.

The dual of (SDP) can be written as:

$$\text{(DSDP)} \quad \begin{cases} \max & b^{\top} y \\ \text{s.t.} & \sum_{i=1}^{m} y_i A_i + S = C, \\ & S \succeq 0, \end{cases}$$

where y_i, $i = 1, \ldots, m$ are scalar variables.

This pair of semidefinite programs can be solved in 'polynomial time'. There are actually several polynomial algorithms. One is the *primal-scaling algorithm* ([1,13,16,17]), which is the analogue of the primal potential reduction algorithm for linear programming. This algorithm uses X to generate the next iterate direction. Another is the *dual-scaling algorithm*

([2,9,16,17]), which is the analogue of the dual-scaling algorithm for linear programming. The dual-scaling algorithm uses only S to generate the iterate direction. The third is the primal-dual scaling algorithm, which uses both X and S to generate iterate directions, including Alizadeh-Haeberly-Overton, Helmberg-Rendl-Vanderbei-Wolkowicz/Kojima-Shida-Hara/Monteiro, Nesterov-Todd, Gu, and Toh directions, as well as directions called the MTW and Half directions (see [6,15], and references therein). All these algorithm possess $O(\sqrt{n}\log(\frac{1}{\epsilon}))$ iteration complexity to yield duality gap accuracy ϵ.

Although they are 'polynomially' solvable, semidefinite programs with dimension n above 1000 have been extremely hard to solve in practice, due to the density of matrices involved in computation. Thus, exploiting the structure and sparsity characteristic of large scale semidefinite programs becomes critical to the efficient computation of their solution.

Many large scale semidefinite programs, such as the relaxations of combinatorial and quadratic optimization problems, have features which make the dual-scaling algorithm the most suitable choice:

1) For large scale problems, S tends to be very *sparse* and *structured* since it is the linear combination of C and the A_is. This sparsity allows considerable savings in both memory and computation time. On the other hand, X, the primal matrix, may be much less sparse and its structure not known beforehand. Thus, primal or primal-dual algorithms cannot fully exploit the sparseness and structure of the data.
2) Many problems under consideration require less accuracy than some other applications. Therefore, the superlinear convergence, exhibited by the primal-dual algorithm, may not be utilized in our applications. The *dual-scaling* linear programming algorithm has been shown to perform equally well when only a lower precision answer is required.
3) In most combinatorial applications, we need only a lower bound for the optimal objective value of (SDP. Solving (DSDP) alone would be sufficient to provide such a lower bound. Moreover, in most applications an interior-feasible point is available to start with. Thus, we may not need to generate and store X at all.
4) Even if an optimal primal solution is necessary, the dual-scaling algorithm can generate a sparsely structured optimal X at the termination of the algorithm.

The dual-scaling algorithm, which is a an extension of the linear programming algorithm, is to reduce the Tanabe-Todd-Ye *primal-dual potential function*

$$\Psi(X,S) = \rho \ln(X \bullet S) - \ln\det X - \ln\det S,$$

where $\rho \geq n + \sqrt{n}$, by a constant at each iteration. Since

$$n\ln(X \bullet S) - \ln\det X - \ln\det \geq n\ln n,$$

the reduction of the potential leads the duality gap, $X \bullet S$, converging 0.

Let

$$\mathcal{A}(X) = \begin{pmatrix} A_1 \bullet X \\ \vdots \\ A_l \bullet X \end{pmatrix} \quad \text{and} \quad \mathcal{A}^\top(y) = \sum_{i=1}^{m} y_i A_i,$$

and let $\overline{z} = C \bullet X$ for some feasible X. Consider the dual potential function

$$\psi(y,\overline{z}) = \rho \ln(\overline{z} - b^\top y) - \ln\det S.$$

Note the relation between the two potential functions:

$$\Psi(X,S) = \psi(y,\overline{z}) - \ln\det X.$$

The *gradient* of ψ with respect to y is:

$$\nabla\psi(y,\overline{z}) = -\frac{\rho}{\overline{z} - b^\top y} b + \mathcal{A}(S^{-1})$$

Each step in the dual-scaling algorithm minimizes the linearized *dual potential function* subject to an *ellipsoidal constraint* that keeps $S \succ 0$ and the quadratic error term small. More precisely, beginning with a strictly feasible dual point (y^k, S^k) and a $\overline{z}^k$, each iteration solves the problem:

$$\begin{cases} \min & \nabla\psi^\top(y^k,\overline{z}^k)(y - y^k) \\ \text{s.t.} & \left\| (S^k)^{-.5} \left(\mathcal{A}^\top(y - y^k)\right) (S^k)^{-.5} \right\| \leq \alpha, \end{cases} \tag{1}$$

where α is a constant in (0, 1). Here, all matrix norms used here will be the Frobenius norm.

Define

$$\widetilde{m}(i,j) = \mathrm{tr}(S^k)^{-1} A_i (S^k)^{-1} A_j$$

and the positive definite matrix

$$M^k = \begin{pmatrix} \widetilde{m}(1,1) & \cdots & \widetilde{m}(1,m) \\ \vdots & \ddots & \vdots \\ \widetilde{m}(m,1) & \cdots & \widetilde{m}(m,m) \end{pmatrix}. \tag{2}$$

In particular, if $A_i = a_i a_i^\top$ is a rank-one matrix, where $a_i \in \mathbf{R}^n$, for $i = 1, \ldots, n$, then

$$M^k = \begin{pmatrix} (a_1^\top (S^k)^{-1} a_1)^2 & \cdots & (a_1^\top (S^k)^{-1} a_m)^2 \\ \vdots & \ddots & \vdots \\ (a_m^\top (S^k)^{-1} a_1)^2 & \cdots & (a_m^\top (S^k)^{-1} a_m)^2 \end{pmatrix}.$$

The minimal solution, y^{k+1}, of (1) is given by

$$y^{k+1} - y^k = \frac{\alpha}{\sqrt{\nabla\psi^\top(y^k, \bar{z}^k) M^{-1} \nabla\psi(y^k, \bar{z}^k)}}\, d(\bar{z}^k)_y$$

where

$$d(\bar{z}^k)_y = -(M^k)^{-1} \nabla\psi(y^k, \bar{z}^k). \tag{3}$$

Let

$$\begin{aligned} P(\bar{z}^k) &= (S^k)^{-.5} \mathcal{A}^\top \left((M^k)^{-1} \nabla\psi(y^k, \bar{z}^k) \right) (S^k)^{-.5} \\ &= (S^k)^{-.5} \mathcal{A}^\top \left(-d(\bar{z}^k)_y \right) (S^k)^{-.5}. \end{aligned}$$

Then

$$\begin{aligned} \nabla\psi^\top(y^k, \bar{z}^k)\, d(\bar{z}^k)_y &= -\left\| P(\bar{z}^k) \right\|^2, \\ \nabla\psi^\top(y^k, \bar{z}^k)(y^{k+1} - y^k) &= -\alpha \left\| P(\bar{z}^k) \right\|, \end{aligned}$$

and the reduction in the potential function satisfies the inequality

$$\psi(y^{k+1}, \bar{z}^k) - \psi(y^k, \bar{z}^k) \le -\alpha \left\| P(\bar{z}^k) \right\| + \frac{\alpha^2}{2(1-\alpha)}.$$

Focusing on the expression of $P(\bar{z}^k)$, it can be rewritten as

$$P(\bar{z}^k) = -\frac{\rho}{\bar{z}^k - b^\top y^k} (S^k)^{.5} X(\bar{z}^k) (S^k)^{.5} + I$$

with

$$\begin{aligned} &X(\bar{z}^k) \\ &= \frac{\bar{z}^k - b^\top y^k}{\rho} (S^k)^{-1} \left(\mathcal{A}^\top (\, d(\bar{z}^k)_y) + S^k \right) (S^k)^{-1}. \end{aligned} \tag{4}$$

Note that $\mathcal{A}(X(\bar{z}^k)) = b$, and $X(\bar{z}^k)$ is a primal feasible solution if and only if $X(\bar{z}^k) \succeq 0$. Furthermore, from the multiplicative structure of (4), $X(\bar{z}^k) \succeq 0$ if and only if

$$\mathcal{A}^\top (d(\bar{z}^k)_y) + S^k \succeq 0,$$

which is a sparse matrix in many applications. Also note that

$$\begin{aligned} C \bullet X(\bar{z}^k) &= S^k \bullet X(\bar{z}^k) + b^\top y^k \\ &= \frac{\bar{z}^k - b^\top y^k}{\rho} \\ &\quad \cdot \left(\mathcal{A}^\top (d(\bar{z}^k)_y) \bullet (S^k)^{-1} + n \right) + b^\top y^k, \end{aligned}$$

which can be efficiently computed.

One can show that, when $\left\| P(\bar{z}^k) \right\|$ is small, then $(X(\bar{z}^k), y^k, S^k)$ is in the neighborhood of the *central path* and $C \bullet X(z^k) < \bar{z}^k$. Thus, we can decrease $\bar{z}^k$ to $C \bullet X(z^k)$. Moreover, $\Psi(X(\bar{z}^k), S^k)$ is reduced from $\Psi(X^k, S^k)$ by a constant.

The theoretical algorithm can be stated as follows.

Dual algorithm

Given $\mathcal{A}(X^0) = b$, $X^0 \succ 0$, $\bar{z}^0 = C \bullet X^0$, $S^0 = C - \mathcal{A}^\top y^0 \succ 0$, and $k := 0$.
do the following:
WHILE $\bar{z}^k - b^\top y^k \ge \epsilon$ DO
1 Compute the matrix M^k of (2).
2 Solve (3) for the dual step direction $d(\bar{z}^k)_y$.
3 IF $\bar{z}^k > C \bullet X(\bar{z}^k)$ AND $\mathcal{A}^\top (d(\bar{z}^k)_y) + S^k \succeq 0$,
THEN $X^{k+1} = X(\bar{z}^k)$ and $\bar{z}^{k+1} = C \bullet X^{k+1}$
ELSE $X^{k+1} = X^k$ and $\bar{z}^{k+1} = \bar{z}^k$
ENDIF
4 Let $y^{k+1} = y^k + \frac{\alpha}{\|P(\bar{z}^{k+1})\|} d(\bar{z}^{k+1})_y$ and $S^{k+1} = C - \mathcal{A}^\top (y^{k+1})$.
5 Set $k := k + 1$ and return to Step 1.

The algorithm occasionally updates the primal solution X and its objective value, but it does not need X in computation. We can derive the following potential reduction theorem:

Theorem 1

$$\Psi(X^{k+1}, S^{k+1}) \leq \Psi(X^k, S^k) - \delta$$

where $\delta > 1/20$ for a suitable α.

This theorem leads to

Corollary 2 *Let $\rho \geq n + \sqrt{n}$. Then, the algorithm terminates in at most $O((\rho - n)\log(X^0 \bullet S^0/\epsilon))$ iterations.*

To accelerate the convergence of the algorithm, one may increase the value of ρ in practice and consider a bigger stepsize α, see [4]. The stopping criterion is often

$$\frac{\bar{z}^k - b^\top y^k}{1 + \left|b^\top y^k\right|} \leq \epsilon,$$

that is, when the *relative duality gap* is less than prescribed accuracy ϵ.

The *dual-scaling* algorithm, described above, has been implemented for solving semidefinite programs, arisen from maximum-cut and 'box'-constrained quadratic optimization, with dimension up to 10000. The computational results are promising.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- Cholesky Factorization
- Duality for Semidefinite Programming
- Interior Point Methods for Semidefinite Programming
- Interval Linear Systems
- Large Scale Trust Region Problems
- Large Scale Unconstrained Optimization
- Linear Programming
- Orthogonal Triangularization
- Overdetermined Systems of Linear Equations
- QR Factorization
- Semidefinite Programming and Determinant Maximization
- Semidefinite Programming: Optimality Conditions and Stability
- Semidefinite Programming and Structural Optimization
- Semi-infinite Programming, Semidefinite Programming and Perfect Duality
- Symmetric Systems of Linear Equations

References

1. Alizadeh F (1991) Combinatorial optimization with interior point methods and semi-definite matrices. PhD Thesis Univ. Minnesota, Minneapolis, MN
2. Anstreicher KM, Fampa M (1996) A long-step path following algorithm for semidefinite programming problems. Working Paper Dept. Management Sci. Univ. Iowa
3. Atkinson KE (1989) An introduction to numerical analysis, 2nd edn. Wiley, New York
4. Benson S, Ye Y, Zhang X (1997) Solving sparse, large scale positive semidefinite programs. SIAM J. Optim. (to appear), Working Paper Dept. Management Sci. Univ. Iowa
5. Fujie T, Kojima M (1997) Semidefinite programming relaxation for nonconvex quadratic programs. J Global Optim 10(4):367–380
6. Fujisawa K, Fukuda M, Kojima M, Nakata K (1997) Numerical evaluation of SDPA (SemiDefinite Programming Algorithm). Res. Report Dept. Math. Computing Sci. Tokyo Inst. Techn. Oh-Okayama B-330
7. Goemans MX, Williamson DP (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J ACM 42:1115–1145
8. Helmberg C, Rendl F (1997) A spectral bundle method for semidefinite programming. ZIB Preprint SC Konrad-Zuse-Zentrum Informationstechnik Berlin, 97–37
9. Jarre F (1993) An interior-point method for minimizing the largest eigenvalue of a linear combination of matrices. SIAM J Control Optim 31:1360–1377
10. Karisch SE, Rendl F, Clausen J (1997) Solving graph bisection problems with semidefinite programming. Techn. Report DIKU Dept. Computer Sci. Univ. Copenhagen 9
11. Lovàsz L, Schrijver A (1990) Cones of matrices and setfunctions, and 0 — 1 optimization. SIAM J Optim 1:166–190
12. Nesterov YuE (1998) Semidefinite relaxation and nonconvex quadratic optimization. Optim Methods Softw 9:141–160
13. Nesterov YuE, Nemirovskii AS (1993) Interior point polynomial methods in convex programming: Theory and algorithms. SIAM, Philadelphia
14. Polijak S, Rendl F, Wolkowicz H (1995) A recipe for semidefinite relaxation for 0-1 quadratic programming. J Global Optim 7:51–73
15. Todd MJ (1997) On search directions in interior-point methods for semidefinite programming. Techn Report School Oper Res Industr Engin Cornell Univ Ithaca, NY, 1205:14853–3801
16. Vandenberghe L, Boyd S (1996) Semidefinite programming. SIAM Rev 38(1):49–95
17. Ye Y (1997) Interior point algorithms: Theory and analysis. Discrete Math and Optim. Wiley, New York
18. Ye Y (1999) Approximating quadratic programming with bound and quadratic constraints. Math Program 84:219–226

Spatial Price Equilibrium

SPE

Anna Nagurney
University Massachusetts, Amherst, USA

MSC2000: 91B50, 91B28

Article Outline

Keywords
The Quantity Model
The Price Model
A Dynamic Model
See also
References

Keywords

Perfectly competitive; Partial equilibrium; Spatial markets; Quantity model; Price model; Variational inequality formulations

Spatial price equilibrium modeling and computation are concerned with the prediction of commodity trade flow patterns between spatially separated supply and demand markets as well as the market commodity prices. The distinctive character of spatial price problems lies in the recognition of transportation costs associated with shipping the commodities between producing and consuming locations or regions. Such models are perfectly competitive partial equilibrium models; *perfectly competitive* in the sense that it is assumed that there are many producers and consumers with no individual being able to affect the market prices and *partial* (as opposed to general) in the sense that only a subset of the commodities in the economy is assumed to be modeled.

In particular, in the *spatial price equilibrium problem*, one seeks to compute the commodity supply prices, demand prices, and trade flows satisfying the equilibrium condition that the demand price is equal to the supply price plus the cost of transportation, if there is trade between the pair of supply and demand markets; if the demand price is less than the supply price plus the transportation cost, then there will be no trade. Spatial price equilibrium problems arise in agricultural markets, energy markets, and financial markets and such models provide the basis for interregional and international trade modeling (see, e. g., [14,16,19,20,21,30]).

The first reference in the literature to such problems was by A. Cournot [2] in 1838, who considered two spatially separated markets. S. Enke [8], more than a century later, used an analogy between spatial price equilibrium and electronic circuits to give the first computational approach, albeit analogue, to such problems, in the case of linear and separable supply and demand functions.

P.A. Samuelson [28] subsequently initiated the rigorous treatment of such problems by establishing that the solution to the spatial price equilibrium problem, as posed by Enke, could be obtained by solving an optimization problem in which the objective function, although artificial, had the interpretation of a net social pay-off function. The spatial price equilibrium, in this case, coincided with the Kuhn-Tucker conditions of the appropriately constructed optimization problem. Samuelson also related Enke's specification to a standard problem in linear programming, the Hitchcock-Koopmans transportation problem and noted that the spatial price equilibrium problem was more general in the sense that the supplies and demands were not known a priori. Finally, Samuelson also identified the network structure of such problems.

T. Takayama and G.C. Judge [29,30] further expanded on the work of Samuelson [28] and showed that the prices and commodity flows satisfying the spatial price equilibrium conditions could be determined by solving a quadratic programming problem in the case of linear supply and demand price functions for which the Jacobians were symmetric and not necessarily diagonal. This theoretical advance enabled not only the qualitative study of equilibrium patterns, but also opened up the possibility for the development of effective computational procedures, based on convex programming, as well as, the exploitation of the network structure (see [5,12,15]).

As noted in Takayama and Judge [30], who developed a variety of spatial price equilibrium models, distinct model formulations are needed, in particular, both quantity and price formulations, depending upon the availability and format of the data. In a *quantity* formulation it is assumed that the supply price functions

and demand price functions are given (and these are a function, respectively, the quantities produced and consumed) whereas in a *price* formulation it is assumed that the supply and demand functions are given and these are a function, respectively, of the supply and demand prices. Moreover, Takayama and Judge [30] realized that a pure optimization framework was not sufficient to handle, for example, multicommodity spatial price equilibrium problems in which the Jacobians of the supply and demand price functions were no longer symmetric.

Towards that end, new formulations were proposed for the spatial price equilibrium problem under more general settings, including fixed point, complementarity, and variational inequality formulations. J.G. MacKinnon [18] gave a fixed point formulation which was then used by H.W. Kuhn and MacKinnon [17] for computational purposes. R. Asmuth, B.C. Eaves, and E.L. Peterson [1] considered the linear asymmetric spatial price equilibrium problem formulated as a linear complementarity problem and proposed Lemke's algorithm for the computation of the spatial price equilibrium. J.S. Pang and P.L. Lee [27] developed special-purpose algorithms based on the complementarity formulation of the problem. M. Florian and M. Los [10] and S.C. Dafermos and A. Nagurney [4] addressed the variational inequality formulations of general spatial price equilibrium models with the latter authors providing sensitivity analysis results. The interrelationships between variational inequality, complementarity, and extremal formulations of spatial price equilibrium problems are given in [11].

Dafermos and Nagurney [5] established the equivalence of the spatial price equilibrium problem with the traffic network equilibrium problem. This identification stimulated further research in network equilibria (cf. [9,21], and the references therein) and in algorithm development for such problems. Computational testing of different algorithms for spatial price equilibrium problems can be found in [11] and [20]. Spatial price equilibrium models have also been used for policy analysis (see, e. g., [16,23,26], and the references therein).

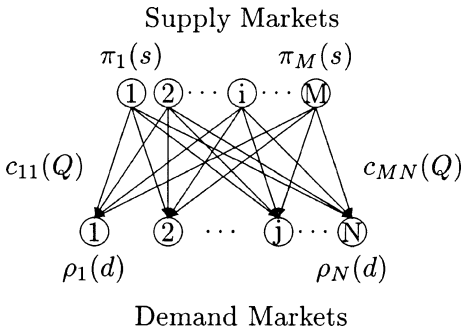
Since spatial price equilibrium problems can be large scale in practice parallel computational approaches have been implemented to solve such problems (cf. [13,22,23]). Recently, general dynamic spatial price equilibrium models have been developed (cf. [24,25]), based on the connection between solutions to variational inequality problems and the stationary points of projected dynamical systems (cf. [7]), and solved using parallel computers.

For definiteness, we first present the quantity model and then the price model and provide the variational inequality formulations of the governing equilibrium conditions. We then present a dynamic quantity model. For additional background, including qualitative and computational results, see [21] and [25].

The Quantity Model

Consider the spatial price equilibrium problem in quantity variables with M supply markets and N demand markets involved in the production and consumption of a homogeneous commodity under perfect competition. Denote a typical supply market by i and a typical demand market by j. Let s_i denote the supply and π_i the supply price of the commodity at supply market i. Let d_j denote the demand and ρ_j the demand price at demand market j. Group the supplies and supply prices, respectively, into a column vector $s \in \mathbf{R}^M$ and a row vector $\pi \in \mathbf{R}^M$. Similarly, group the demands and demand prices, respectively, into a column vector $d \in \mathbf{R}^N$ and a row vector $\rho \in \mathbf{R}^N$. Let Q_{ij} denote the nonnegative commodity shipment between the supply and demand market pair (i, j), and let c_{ij} denote the unit transaction cost associated with trading the commodity between (i, j). The unit transaction costs are assumed to include the unit costs of transportation from supply markets to demand markets, and, depending upon the application, may also include a tax/tariff, duty, or subsidy incorporated into these costs. Group the commodity shipments into a column vector $Q \in \mathbf{R}^{MN}$ and the transaction costs into a row vector $c \in \mathbf{R}^{MN}$. The *network structure of the problem* is depicted in Fig. 1.

Assume that the supply price at any supply market may, in general, depend upon the supply of the commodity at every supply market, that is, $\pi = \pi(s)$, where π is a known smooth function. Similarly, the demand price at any demand market may depend upon, in general, the demand of the commodity at every demand market, that is, $\rho = \rho(d)$, where ρ is a known smooth function. The unit transaction cost between a pair of supply and demand markets may depend upon the shipments of the commodity between every pair of



Spatial Price Equilibrium, Figure 1
Network structure of spatial market problem

markets, that is, $c = c(Q)$, where c is a known smooth function.

The supplies, demands, and shipments of the commodity, in turn, must satisfy the following feasibility conditions, which are also referred to as the *conservation of flow equations*:

$$s_i = \sum_{j=1}^{N} Q_{ij}, \quad i = 1, \ldots, M,$$

$$d_j = \sum_{i=1}^{M} Q_{ij}, \quad j = 1, \ldots, N,$$

$$Q_{ij} \geq 0, \quad i = 1, \ldots, M; j = 1, \ldots, N.$$

In other words, the supply at each supply market is equal to the commodity shipments out of that supply market to all the demand markets. Similarly, the demand at each demand market is equal to the commodity shipments from all the supply markets into that demand market.

Definition 1 (*spatial price equilibrium*) Following [28] and [30], the supply, demand, and commodity shipment pattern (s^*, Q^*, d^*) constitutes a spatial price equilibrium, if it is feasible, and for all pairs of supply and demand markets (i, j), it satisfies the conditions:

$$\pi_i(s^*) + c_{ij}(Q^*) \begin{cases} = \rho_j(d^*), & \text{if } Q_{ij}^* > 0 \\ \geq \rho_j(d^*), & \text{if } Q_{ij}^* = 0. \end{cases}$$

Hence, if the commodity shipment between a pair of supply and demand markets is positive at equilibrium, then the demand price at the demand market must be equal to the supply price at the originating supply market plus the unit transaction cost. If the commodity shipment is zero in equilibrium, then the supply price plus the unit transaction cost can exceed the demand price.

The spatial price equilibrium can be formulated as a variational inequality problem (cf. [3,10], and [21] for proofs). Precisely, we have

Theorem 2 (*variational inequality formulation*) *A commodity supply, shipment, and demand pattern $(s^*, Q^*, d^*) \in K$ is a spatial price equilibrium if and only if it satisfies the following variational inequality problem:*

$$\langle \pi(s^*), s - s^* \rangle + \langle c(Q^*), Q - Q^* \rangle + \langle -\rho(d^*), d - d^* \rangle \geq 0, \quad \forall (s, Q, d) \in K,$$

where

$$K \equiv \{(s, Q, d) : \textit{ feasibility conditions hold}\}$$

and $\langle \cdot, \cdot \rangle$ denotes the inner product.

Example 3 For illustrative purposes, we now present a small example. Consider the spatial price equilibrium problem consisting of two supply markets and two demand markets. Assume that the functions are as follows:

$$\begin{aligned} \pi_1(s) &= 5s_1 + s_2 + 1, \\ \pi_2(s) &= 4s_2 + s_1 + 2, \\ c_{11}(Q) &= 2Q_{11} + Q_{12} + 3, \\ c_{12}(Q) &= Q_{12} + 5, \\ c_{21}(Q) &= 3Q_{21} + Q_{22} + 5, \\ c_{22}(Q) &= 3Q_{22} + 2Q_{21} + 9, \\ \rho_1(d) &= -2d_1 - d_2 + 21, \\ \rho_2(d) &= -5d_2 - 3d_1 + 29. \end{aligned}$$

It is easy to verify that the spatial price equilibrium pattern is given by:

$$\begin{matrix} & s_1^* = 2, & s_2^* = 1, & \\ Q_{11}^* = 1, & Q_{12}^* = 1, & Q_{21}^* = 1, & Q_{22}^* = 0, \\ & d_1^* = 2, & d_2^* = 1. & \end{matrix}$$

In one of the simplest models, in which the Jacobians of the supply price functions, $[\partial \pi / \partial s]$, the transportation (or transaction) cost functions, $[\partial c / \partial Q]$, and minus the demand price functions, $-[\partial \rho / \partial d]$ are di-

agonal and positive definite, then the spatial price equilibrium pattern coincides with the Kuhn-Tucker conditions of the strictly convex optimization problem:

$$\min_{Q\in\mathbb{R}^{MN}_{+}} \left[\sum_{i=1}^{M} \int_{0}^{\sum_{j=1}^{N} Q_{ij}} \pi_i(x)\, dx - \sum_{i=1}^{M}\sum_{j=1}^{N} \int_{0}^{Q_{ij}} c_{ij}(y)\, dy - \sum_{j=1}^{N} \int_{0}^{\sum_{i=1}^{M} Q_{ij}} \rho_j(z)\, dz \right].$$

The Price Model

We now describe briefly the price model. The notation is as for the quantity model except now we consider the situation where the supplies at the supply markets, denoted by the row vector s may, in general, depend upon the column vector of supply prices π, that is, $s = s(\pi)$. Similarly, assume that the demands at the demand markets, denoted by the row vector d, may, in general, depend upon the column vector of demand prices ρ, that is, $d = d(\rho)$. The transaction/transportation costs are of the same form as in the quantity model.

The spatial equilibrium conditions now take the following form: For all pairs of supply and demand markets (i, j), $i = 1, \ldots, M$; $j = 1, \ldots, N$:

$$\pi_i^* + c_{ij}(Q^*) \begin{cases} = \rho_j^* & \text{if } Q_{ij}^* > 0, \\ \geq \rho_j^* & \text{if } Q_{ij}^* = 0, \end{cases}$$

where

$$s_i(\pi^*) \begin{cases} = \sum_{j=1}^{N} Q_{ij}^* & \text{if } \pi_i^* > 0, \\ \geq \sum_{j=1}^{N} Q_{ij}^* & \text{if } \pi_i^* = 0, \end{cases}$$

and

$$d_j(\rho^*) \begin{cases} = \sum_{i=1}^{M} Q_{ij}^* & \text{if } \rho_j^* > 0, \\ \leq \sum_{i=1}^{M} Q_{ij}^* & \text{if } \rho_j^* = 0. \end{cases}$$

The first equilibrium condition is as in the quantity model with the exception that the prices are now variables. The other two conditions allow for the possibility that if the equilibrium prices are zero, then one may have excess supply and/or excess demand at the respective market(s). If the prices are positive, then the markets will clear.

The variational inequality formulation of the equilibrium conditions governing the price model is now given (for a proof, see [21]).

Theorem 4 (*variational inequality formulation*) *The vector $x^* \equiv (Q^*, \pi^*, \rho^*) \in \boldsymbol{R}_{+}^{MN+M+N}$ is an equilibrium shipment and price vector if and only if it satisfies the variational inequality:*

$$\langle F(x^*), x - x^* \rangle \geq 0, \qquad \forall x \in \mathbb{R}_{+}^{MN+M+N},$$

where $F: K \rightarrow \boldsymbol{R}^{MN+M+N}$ is the row vector: $F(x) \equiv (T(x), S(x), D(x))$, where $T: \boldsymbol{R}_{+}^{MN+M+N} \rightarrow \boldsymbol{R}^{MN}$, $S: \boldsymbol{R}_{+}^{MN+M} \rightarrow \boldsymbol{R}^{M}$, and $D: \boldsymbol{R}_{+}^{MN+N} \rightarrow \boldsymbol{R}^{N}$ are defined by:

$$T_{ij} = \pi_i + c_{ij}(Q) - \rho_j,$$

$$S_i = s_i(\pi) - \sum_{j=1}^{N} Q_{ij},$$

$$D_j = \sum_{i=1}^{M} Q_{ij} - d_j(\rho).$$

A Dynamic Model

We now present the projected dynamical system model of the latter spatial price problem. For additional background, qualitative properties, as well as computational results, see [25] and the references therein. In view of variational inequality governing the price model, we may write the dynamical system as:

$$\begin{pmatrix} \dot{Q} \\ \dot{\pi} \\ \dot{\rho} \end{pmatrix} = \Pi_{\mathbb{R}_{+}^{MN+M+N}} \left(\begin{pmatrix} Q \\ \pi \\ \rho \end{pmatrix}, \begin{pmatrix} -T(Q,\pi,\rho) \\ -S(Q,\pi) \\ -D(Q,\rho) \end{pmatrix} \right),$$

where assuming that the feasible set K is a convex polyhedron (as is the case here), and given $x \in K$ and $v \in \mathbf{R}^n$, we define the projection of the vector v at x (with respect to K) by

$$\Pi_K(x, v) = \lim_{\delta \rightarrow 0} \frac{P_K(x + \delta v) - x}{\delta},$$

where P_K is defined as:

$$P_K(x) = \arg\min_{z \in K} \|x - z\|,$$

and $\|\cdot\|$ denotes the Euclidean norm.

More explicitly, if the demand price at a demand market exceeds the supply price plus the unit transaction cost associated with shipping the commodity between a pair of supply and demand markets, then the commodity shipment between this pair of markets will increase. On the other hand, if the supply price plus unit transaction cost exceeds the demand price, then the commodity shipment between the pair of supply and demand markets will decrease. If the supply at a supply market exceeds (is exceeded by) the commodity shipments out of the market, then the supply price will decrease (increase). In contrast, if the demand at a demand market exceeds (is exceeded by) the commodity shipments into the market, then the demand price will increase (decrease).

However, if at the boundary the vector field $-F$ points 'out' of the feasible set, the right-hand side of the ordinary differential equation becomes the projection of F onto the boundary. In other words, if the commodity shipments, and/or the supply prices, and/or the demand prices are driven to be negative, then the projection ensures that the commodity shipments and the prices will be nonnegative, by setting the values equal to zero. The solution to the projected dynamical system then evolves along a 'section' of the boundary of the feasible set. At a later time, the solution may re-enter the interior of the constraint set, or it may enter a lower-dimensional part of its boundary, with, ultimately, the spatial price equilibrium conditions being reached at a stationary point, that is, when $\dot{x} = 0$.

See also

- ► Equilibrium Networks
- ► Financial Equilibrium
- ► Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ► Oligopolistic Market Equilibrium
- ► Traffic Network Equilibrium
- ► Walrasian Price Equilibrium

References

1. Asmuth R, Eaves BC, Peterson EL (1979) Computing economic equilibria on affine networks. Math Oper Res 4:209–214
2. Cournot A (1897) Researches into the mathematical principles of wealth. MacMillan, New York, English translation; original: 1838.
3. Dafermos S (1983) An iterative scheme for variational inequalities. Math Program 28:57–72
4. Dafermos S, Nagurney A (1984) Sensitivity analysis for the general spatial economics equilibrium problem. Oper Res 32:1069–1086
5. Dafermos S, Nagurney A (1985) Isomorphism between spatial price and traffic network equilibrium models. Lefschetz Center Dynamical Systems, vol 85-17. Brown Univ., Providence, RI
6. Dafermos S, Nagurney A (1989) Supply and demand equilibration algorithms for a class of market equilibrium problems. Transport Sci 23:118–124
7. Dafermos S, Nagurney A (1993) Dynamical systems and variational inequalities. Ann Oper Res 44:9–42
8. Enke S (1951) Equilibrium among spatially separated markets: solution by electronic analogue. Econometrica 10:40–47
9. Florian M, Hearn D (1995) Network equilibrium models and algorithms. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing. Handbook Oper Res and Management Sci. Elsevier, Amsterdam, pp 485–550
10. Florian M, Los M (1982) A new look at static spatial price equilibrium models. Regional Sci and Urban Economics 12:579–597
11. Friesz TL, Harker PT, Tobin RL (1984) Alternative algorithms for the general network spatial price equilibrium problem. J Reg Sci 24:473–507
12. Glassey CR (1978) A quadratic network optimization model for equilibrium single commodity trade flows. Math Program 14:98–107
13. Guder F, Morris JG, Yoon SH (1992) Parallel and serial successive overrelaxation for multicommodity spatial price equilibrium problems. Transport Sci 26:48–58
14. Harker PT (ed) (1985) Spatial price equilibrium: Advances in theory, computation and application. Lecture Notes Economics and Math Systems. Springer, Berlin
15. Jones PC, Saigal R, Schneider MC (1984) Computing nonlinear network equilibria. Math Program 31:57–66
16. Judge GG, Takayama T (eds) (1973) Studies in economic planning over space and time. North-Holland, Amsterdam
17. Kuhn HW, MacKinnon JG (1975) Sandwich method for finding fixed points. J Optim Th Appl 17:189–204
18. MacKinnon JG (1975) An algorithm for the generalized transportation problem. Regional Sci and Urban Economics 5:445–464
19. Nagurney A (1987) Competitive equilibrium problems, variational inequalities, and regional science. J Reg Sci 27:503–514
20. Nagurney A (1987) Computational comparisons of spatial price equilibrium methods. J Reg Sci 27:55–76
21. Nagurney A (1999) Network economics: A variational inequality approach, 2nd edn. Kluwer, Dordrecht
22. Nagurney A, Kim DS (1989) Parallel and serial variational inequality decomposition algorithms for multicommodity

market equilibrium problems. Internat J Supercomputer Appl 3:34–59
23. Nagurney A, Nicholson CF, Bishop PM (1996) Massively parallel computation of large-scale spatial price equilibrium models with discriminatory ad valorem tariffs. Ann Oper Res 68:281–300
24. Nagurney A, Takayama T, Zhang D (1995) Projected dynamical systems modeling and computation of spatial network equilibria. Networks 26:69–85
25. Nagurney A, Zhang D (1997) Projected dynamical systems and variational inequalities with applications. Kluwer, Dordrecht
26. Nagurney A, Zhao L (1991) A network equilibrium formulation of market disequilibrium an variational inequalities. Networks 21:109–132
27. Pang JS, Lee PL (1981) A parametric linear complementarity technique for the computation of equilibrium prices in a single commodity spatial model. Math Program 20:81–102
28. Samuelson PA (1952) Spatial price equilibrium and linear programming. Amer Economic Rev 42:283–303
29. Takayama T, Judge GG (1964) An intertemporal price equilibrium model. J Farm Economics 46:477–484
30. Takayama T, Judge GG (1971) Spatial and temporal price and allocation models. North-Holland, Amsterdam

Spectral Projected Gradient Methods

Ernesto G. Birgin[1], J. M. Martínez[2], Marcos Raydan[3]
[1] Department of Computer Science IME-USP, University of São Paulo, São Paulo, Brazil
[2] Department of Applied Mathematics, University of Campinas, Campinas, Brazil
[3] Departamento de Computación, Facultad de Ciencias, Universidad Central de Venezuela, Caracas, Venezuela

MSC2000: 49M07, 49M10, 65K, 90C06, 90C20

Article Outline

Keywords
Introduction
Method
The Secant Connection
The Line Search
General Form and Convergence
Numerical Example
Further Developments
References

Keywords

Spectral projected gradient methods; Projected gradients; Nonmonotone line search; Large scale problems; Convex constrained problems; Bound constrained problems; Spectral gradient method

Introduction

Cauchy's steepest descent algorithm [22] is the most ancient method for multidimensional unconstrained minimization. Given f, a real smooth function defined on $\mathbb{R}^n$, the idea is to iterate according to:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)\,, \tag{1}$$

with the expectancy that the sequence $\{x_k\}$ would approximate a minimizer of f. The greedy choice of the steplength α_k is

$$f(x_k - \alpha_k \nabla f(x_k)) = \min_{\alpha \geq 0} f(x_k - \alpha \nabla f(x_k))\,. \tag{2}$$

The poor practical behavior of (1)–(2) has been known for many years. If the level sets of f resemble long valleys, the sequence $\{x_k\}$ displays a typical zig-zagging trajectory and the speed of convergence is very slow. In the simplest case, in which f is a strictly convex quadratic, the method converges to the solution with a Q-linear rate of convergence whose factor tends to 1 when the condition number of the Hessian tends to infinity.

Nevertheless, the structure of the iteration (1) is very attractive, especially when one deals with large-scale (many variables) problems. Each iteration only needs the computation of the gradient $\nabla f(x_k)$ and the number of algebraic operations is linear in terms of n. As a consequence, a simple paper by Barzilai and Borwein published in 1988 [4] attracted some justified attention. Barzilai and Borwein discovered that, for some choices of α_k, Cauchy's method converges superlinearly to the solution, if $f : \mathbb{R}^2 \to \mathbb{R}$ is a convex quadratic. Some members of the optimization community began to believe that the existence of an efficient method for large-scale minimization based only on gradient directions could be possible.

In 1993, Raydan [60] proved the convergence of the Barzilai–Borwein method for arbitrary strictly convex quadratics. He showed that the method was far more efficient than the steepest descent algorithm (1)–(2) although it was not competitive with the Conjugate Gradient method of Hestenes and Stiefel [49] for quadratic

problems. The possibility of obtaining superlinear convergence for arbitrary n was discarded by Fletcher's work [40] (see also [60]) and a bizarre behavior of the method seemed to discourage the application to general (not necessarily quadratic) unconstrained minimization: the sequence of functional values $f(x_k)$ did not decrease monotonically and, sometimes, monotonicity was severely violated.

However, starting with the work by Grippo, Lampariello and Lucidi [47], nonmonotone strategies for function minimization began to become popular. These strategies made it possible to define globally convergent algorithms without monotone decrease requirements. The philosophy behind nonmonotone strategies is that, many times, the first choice of a trial point by a minimization algorithm hides a lot of wisdom about the problem structure and that such knowledge can be destroyed by the decrease imposition. For example, if one applies Newton's method to a problem in which several components of the gradient are linear, these components vanish at the first trial point of each iteration, but the objective function value does not necessarily decrease at this trial point.

Therefore, the conditions were given for the implementation of the Barzilai–Borwein method for general unconstrained minimization with the help of a nonmonotone strategy. Raydan [61] defined this method in 1997 using the GLL strategy [47]. He proved global convergence and exhibited numerical experiments that showed that, perhaps surprisingly, the method was more efficient than classical conjugate gradient methods for minimizing general functions. These nice comparative numerical results were possible because, albeit the Conjugate Gradient method of Hestenes and Stiefel continued to be the rule of choice for solving many convex quadratic problems, its efficiency was hardly inherited by generalizations for minimizing general functions. Therefore, there existed a wide space for variations of the Barzilai–Borwein idea.

The Spectral Projected Gradient (SPG) method [16,17,18] was born from the marriage of the Barzila-Borwein (spectral) nonmonotone ideas with classical projected gradient strategies [7,46,53]. This method is applicable to convex constrained problems in which the projection on the feasible set is easy to compute. Since its appearance, the method has been intensively used in applications [3,6,10,14,15,19,20,24,26,35,42,50,59,63,64,65,69]. Moreover, it has been the object of several spectral-parameter modifications, alternative nonmonotone strategies have been suggested, convergence and stability properties have been elucidated and it has been combined with other algorithms for different optimization problems.

Method

The Secant Connection

Quasi-Newton secant methods for unconstrained optimization [36,37] obey the recursive formula

$$x_{k+1} = x_k + \alpha_k B_k^{-1} \nabla f(x_k)\,. \tag{3}$$

The sequence of matrices $\{B_k\}$ satisfy the *secant equation*

$$B_{k+1} s_k = y_k\,, \tag{4}$$

where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. By (4), it can be shown that, at the *trial point* $x_k - B_k^{-1}\nabla f(x_k)$, the affine approximation of $\nabla f(x)$ that interpolates the gradient at x_k and x_{k-1} vanishes for all $k \geq 1$.

Now assume that we want a matrix B_{k+1} with a very simple structure that satisfies (4). More precisely, we wish

$$B_{k+1} = \sigma_{k+1} I\,,$$

with $\sigma_{k+1} \in \mathbb{R}$. (4) becomes:

$$\sigma_{k+1} s_k = y_k\,.$$

In general, this equation cannot be solved. However, accepting the least-squares solution that minimizes $\|\sigma s_k - y_k\|_2^2$, we obtain:

$$\sigma_{k+1} = \frac{s_k^T y_k}{s_k^T s_k}\,. \tag{5}$$

This formula defines the most popular Barzilai–Borwein method [61]. Namely, the method for unconstrained minimization is of the form (3), where, at each iteration,

$$d_k = -\frac{1}{\sigma_k}\nabla f(x_k)$$

and formula (5) is used to generate the coefficients σ_k provided that they are bounded away from zero and

that they are not very large. In other words, the method uses safeguards $0 < \sigma_{\min} < \sigma_{\max} < \infty$ and defines, at each iteration:

$$\sigma_{k+1} = \min\left\{\sigma_{\max}, \max\left\{\sigma_{\min}, \frac{s_k^T y_k}{s_k^T s_k}\right\}\right\} .$$

By the Mean-Value Theorem of Integral Calculus, one has:

$$y_k = \left(\int_0^1 \nabla^2 f(x_k + t s_k)\, dt\right) s_k .$$

Therefore, formula (5) defines a Rayleigh quotient relative to the average Hessian matrix $(\int_0^1 \nabla^2 f(x_k + t s_k) dt) s_k$. This coefficient is between the minimum and the maximum eigenvalue of the average Hessian, which motivates the denomination of Spectral Method [16].

Writing the secant equation as $H y_k = s_k$, which is also standard in the Quasi-Newton tradition, we arrive to a different spectral coefficient: $\frac{y_k^T y_k}{s_k^T y_k}$. Curiously, both this dual and the primal coefficient had been used for many years in practical quasi-Newton methods to define the initial matrices B_0 [58].

The Line Search

The Spectral Projected Gradient method (SPG) aims to minimize f on a closed and convex set Ω. The method, as well as its unconstrained counterpart [61] has the form

$$x_{k+1} = x_k + \alpha_k d_k . \tag{6}$$

The search direction d_k has been defined in [16] as

$$d_k = P(x_k - \frac{1}{\sigma_k}\nabla f(x_k)) - x_k ,$$

where P denotes the Euclidean projection on Ω. A related method with approximate projections has been defined in [18]. The direction d_k is a descent direction, which means that, if $d_k \neq 0$, one has that $f(x_k + \alpha d_k) \ll f(x_k)$ for α small enough. This means that, in principle, one could define convergent methods imposing sufficient decrease at every iteration. However, this leads to disastrous practical results. For this reason, the spectral methods employ a nonmonotone line search that does not impose functional decrease at every iteration. In [16,17,18] the GLL [47] search is used. This line search depends on an integer parameter $M \geq 1$ and imposes a functional decrease every M iterations (if $M = 1$ then GLL line search reduces to a monotone line search).

The line search is based on a safeguarded quadratic interpolation and aims to satisfy an Armijo-type criterion with a sufficient decrease parameter $\gamma \in (0, 1)$. The safeguarding procedure acts when the minimum of the one-dimensional quadratic lies outside $[\tau_1, \tau_2\alpha]$, and not when it lies outside $[\tau_1\alpha, \tau_2\alpha]$ as usually implemented. This means that, when interpolation tends to reject 90% (for $\sigma_1 = 0.1$) of the original search interval $[0, 1]$, we judge that its prediction is not reliable and we prefer the more conservative bisection. The complete line search procedure is described below.

Algorithm 3.1: Line Search
Compute $f_{\max} = \max\{f(x_{k-j}) | 0 \leq j \leq \min\{k, M-1\}\}$, $x_+ \leftarrow x_k + d_k$, $\delta \leftarrow \langle \nabla f(x_k)), d_k\rangle$ and set $\alpha \leftarrow 1$.
Step 1. *Test nonmonotone Armijo-like criterion*
If $f(x_+) \leq f_{\max} + \gamma\alpha\delta$ then set $\alpha_k \leftarrow \alpha$ and stop.

Step 2. *Compute a safeguarded new trial steplength*
Compute $\alpha_{tmp} \leftarrow -\frac{1}{2}\alpha^2\delta / (f(x_+) - f(x_k) - \alpha\delta)$.
If $\alpha_{tmp} \geq \sigma_1$ and $\alpha_{tmp} \leq \sigma_2\alpha$ then set $\alpha \leftarrow \alpha_{tmp}$. Otherwise, set $\alpha \leftarrow \alpha/2$.

Step 3. Compute $x_+ \leftarrow x_k + \alpha d_k$ and go to Step 1.

Remark. In the case of rejection of the first trial point, the next ones are computed along the same direction. As a consequence, the projection operation is performed only once per iteration.

General Form and Convergence

The Spectral Projected Gradient method SPG aims to solve the problem

$$\text{Minimize } f(x) \text{ subject to } x \in \Omega , \tag{7}$$

where f admits continuous first derivatives and $\Omega \subset \mathbb{R}^n$ is closed and convex.

We say that a point $x \in \Omega$ is *stationary* if

$$\nabla f(x)^T d \geq 0$$

for all $d \in \mathbb{R}^n$ such that $x + d \in \Omega$.

In [18], SPG method has been presented as a member of a wider family of "Inexact Variable Metric" methods for solving (7). Let $\mathbb{B}$ be the set of $n \times n$ positive definite matrices such that $\|B\| \leq L$ and $\|B^{-1}\| \leq L$. Therefore, $\mathbb{B}$ is a compact set of $\mathbb{R}^{n\times n}$. In the spectral gradient approach, the matrices will be of the form σI, with $\sigma \in [\sigma_{\min}, \sigma_{\max}]$.

Algorithm 4.1: Inexact Variable Metric Method
Assume $\eta \in (0, 1]$, $\gamma \in (0, 1)$, $0 < \tau_1 < \tau_2 < 1$, $M \geq 1$ an integer number. Let $x_0 \in \Omega$ be an arbitrary initial point. We denote $g_k = \nabla f(x_k)$ for all $k = 0, 1, 2, \ldots$ Given $x_k \in \Omega$, $B_k \in \mathbb{B}$, the steps of the kth iteration of the algorithm are:

Step 1. *Compute the search direction*
Consider the subproblem

$$\text{Minimize } Q_k(d) \quad \text{subject to} \quad x_k + d \in \Omega\,, \tag{8}$$

where

$$Q_k(d) = \frac{1}{2} d^T B_k d + g_k^T d\,.$$

Let $\bar{d}_k$ be the minimizer of (8). (This minimizer exists and is unique by the strict convexity of the subproblem (8), but does not need to be computed.)

Let d_k be such that $x_k + d_k \in \Omega$ and

$$Q_k(d_k) \leq \eta\, Q_k(\bar{d}_k)\,.$$

If $d_k = 0$, stop the execution of the algorithm declaring that x_k is a stationary point.

Step 2. *Compute the steplength*
Compute $f_{\max} = \max\{f(x_{k-j}) | 0 \leq j \leq \min\{k, M - 1\}\}$, $\delta \leftarrow \langle g_k, d_k\rangle$ and set $\alpha \leftarrow 1$.
If

$$f(x_k + \alpha d_k) \leq f_{\max} + \gamma\alpha\delta\,, \tag{9}$$

set $\alpha_k = \alpha$, $x_{k+1} = x_k + \alpha_k d_k$ and finish the iteration. Otherwise, choose $\alpha_{\text{new}} \in [\tau_1\alpha, \tau_2\alpha]$, set $\alpha \leftarrow \alpha_{\text{new}}$ and repeat test (9).

Remarks. (a) Algorithm 3.1 is a particular case of Step 2 of Algorithm 4.1. (b) In the definition of Algorithm 4.1 the possibility $\eta = 1$ corresponds to the case in which the subproblem (8) is solved exactly.

The main theoretical results [18] are stated below. Firstly, it is shown that an iteration necessarily finishes and then it is shown that every limit point of a sequence generated by the algorithm is stationary.

Theorem 4.1. *The algorithm is well defined.*

Theorem 4.2. *Assume that the level set* $\{x \in \Omega \mid f(x) \leq f(x_0)\}$ *is bounded. Then, either the algorithm stops at some stationary point* x_k*, or every limit point of the generated sequence is stationary.*

Numerical Example

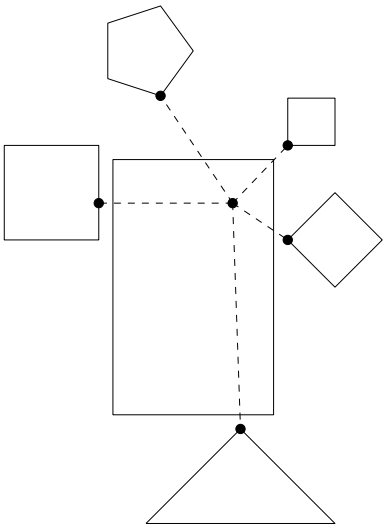
In [17] a family of *location* problems was introduced. Given a set of *npol* disjoint polygons in $\mathbb{R}^2$ we wish to find the point y that minimizes the sum of the distances to those polygons. Therefore, the problem is

$$\min_{z^i, y} \sum_{i=1}^{npol} \|z^i - y\|_2$$
$$\text{subject to } z^i \in P_i\,, \quad i = 1, \ldots, npol\,.$$

Let us write $x = (z_1^1, z_2^1, \ldots, z_1^{npol}, z_2^{npol}, y_1, y_2)$. We observe that the problem has $2 \times (npol + 1)$ variables. The number of (linear inequality) constraints is $\sum_{i=1}^{npol} \nu_i$, where ν_i is the number of vertices of the polygon P_i. Each constraint defines a half-plane in $\mathbb{R}^2$. Figure 1 shows the solution of a small five-polygons problem.

For projecting x onto the feasible set observe that we only need to project each z^i separately onto the corresponding polygon P_i. In the projection subroutine we consider the half-planes that define the polygon. If z^i belongs to all these half-planes, then z^i is the projection onto P_i. Otherwise, we consider the set of half-planes to which z^i does not belong. We project z^i onto these half-planes and we discard the projected points that do not belong to P_i. Let A_i be the (finite) set of nondiscarded half-plane projections and let V_i be the set of vertices of P_i. Then, the projection of z^i onto P_i is the point of $A_i \cup V_i$ that is closest to z^i. The projection subroutine are included in the test driver for SPG method [17].

Varying *npol* and choosing randomly the localization of the polygons and the number of vertices of each one, several test problems were generated and solved by the SPG method in [17]. The biggest prob-



Spectral Projected Gradient Methods, Figure 1
Five-polygons problem

lem had 48,126 polygons, 96,254 variables and 578,648 constraints. Using the origin as initial approximation, it was solved by the SPG method in 17 iterations, using 19 function evaluations and 12.97 s of CPU time on a Sun SparcStation 20 with the following main characteristics: 128 Mbytes of RAM, 70 MHz, 204.7 mips, 44.4 Mflops. (Codes were in Fortran 77 and the compiler option adopted was "-O".)

Further Developments

Developments on spectral gradient and spectral projected gradient ideas include:

1. Application and implementation of the spectral methods to particular optimization problems: Linear inequality constraints were considered in [1]. In [38] the SPG method was used to solve Augmented Lagrangian subproblems. The spectral gradient method solves the subproblems originated by the application of an exponential penalty method to linearly constrained optimization in [56].
2. Preconditioning: The necessity of preconditioning for very ill-conditioned problems has been recognized in several works [5,23,45,54,57].
3. Extensions: The spectral residual direction was defined in [51] to introduce a new method that aims to solve nonlinear systems of equations using only the vector of residues. See, also, [48,52,70]. The SPG method has been extended for solving nondifferentiable convex constrained problems [25].
4. Association with other methods: The SPG method has been used in the context of active-set methods for box-constrained optimization in [2,13,12]. Namely, SPG iterations are used in these methods for abandoning faces whereas Newtonian iterations are employed inside the faces of the feasible region. The opposite idea was used in [44], where spectral directions were used inside the faces and a different orthogonal strategy was employed to modify the set of current active constraints (see also [9]). Spectral ideas were also used in association with conjugate gradients in [11]. Combinations of the spectral gradient method with other descent directions were suggested in [21,28].
5. Nonmonotone alternative rules: Dai and Fletcher [30] observed that, in some cases, even the descent GLL strategy was very conservative and, so, more chance should be given to the pure spectral method ($\alpha_k = 1$ for all k). However, they showed that, for quadratic minimization with box constraints, the pure method is not convergent. Therefore, alternative tolerant nonmonotone rules were suggested. Dai and Zhang [31] noted that the behavior of spectral methods heavily depend on the choice of the parameter M of the GLL search and proposed an adaptive nonmonotone search independent of M. Over and under relaxations of the spectral step were studied by Raydan and Svaiter [62].
6. Alternative choices of the spectral parameter: In [43] it was observed that the convergence theory of the spectral gradient method for quadratics remains valid when one uses Rayleigh coefficients originated in retarded iterations (see also [55]). In [32], for unconstrained optimization problems, the same stepsize is reused for m consecutive iterations (CBB method). This cyclic method is locally linearly convergent to a local minimizer with positive definite Hessian. Numerical evidence indicates that when $m > n/2 >= 3$, where n is the problem dimension, CBB is locally superlinearly convergent. In the special case $m = 3, n = 2$, the convergence rate is, in general, no better than linear [32].
   In [34] the stepsize in the spectral gradient method was interpreted from the point of view of interpolation and two competitive modified spectral-like

gradient methods were defined. Yuan [68] defined a new stepsize for unconstrained optimization that seems to possess spectral gradient properties preserving monotonicity.

7. Asymptotic behavior analysis: A careful consideration of the asymptotic practical and theoretical behavior of the Barzilai–Borwein method may be found in [41]. Dai and Fletcher [29] showed interesting transition properties of the spectral gradient method for quadratic functions as depending on the number of variables. Dai and Liao [33] proved the *R*-linear convergence of the spectral gradient method for general functions and, as a consequence, established that the spectral stepsize is always accepted by the non-monotone line search when the iterate is close to the solution. The convergence of the inexact SPG method was established in [66,67] under different assumptions than the ones used in [18]. Assuming Lipschitz-continuity of the objective functions, these authors eliminated the bounded-level-set assumption of [18].

References

1. Andreani R, Birgin EG, Martínez JM, Yuan J (2005) Spectral projected gradient and variable metric methods for optimization with linear inequalities. IMA J Numer Anal 25:221–252
2. Andretta M, Birgin EG, Martínez JM (2005) Practical active-set Euclidian trust-region method with spectral projected gradients for bound-constrained minimization. Optimization 54:305–325
3. Azofeifa D, Clark N, Vargas W (2005) Optical and electrical properties of terbium films as a function of hydrogen concentration. Physica Status Solidi B Basic – Solid State Physics 242:2005–2009
4. Barzilai J, Borwein JM (1988) Two point step size gradient methods. Numer IMAJ Anal 8:141–148
5. Bello L, Raydan M (2005) Preconditioned spectral projected gradient method on convex sets. Comput J Math 23:225–232
6. Bello L, Raydan M (2007) Convex constrained optimization for the seismic reflection tomography problem. Appl J Geophys 62:158–166
7. Bertsekas DP (1976) On the Goldstein-Levitin-Polyak gradient projection method. Trans IEEE Auto Control 21:174–184
8. Bertsekas DP (1999) Nonlinear Programming, 2nd edn. Athena Scientific, Belmont
9. Bielschowsky RH, Friedlander A, Gomes FAM, Martínez JM, Raydan M (1997) An adaptive algorithm for bound constrained quadratic minimization. Investig Oper 7:67–102
10. Birgin EG, Chambouleyron IE, Martínez JM (2003) Optimization problems in the estimation of parameters of thin films and the elimination of the influence of the substrate. Comput J Appl Math 152:35–50
11. Birgin EG, Martínez JM (2001) A spectral conjugate gradient method for unconstrained optimization. Appl Math Optim 43:117–128
12. Birgin EG, Martínez JM (2002) Large-scale active-set box-constrained optimization method with spectral projected gradients. Comput Optim Appl 23:101–125
13. Birgin EG, Martínez JM (2001) A box constrained optimization algorithm with negative curvature directions and spectral projected gradients. Computing 15(Supl):49–60
14. Birgin EG, Martínez JM, Mascarenhas WF, Ronconi DP (2006) Method of sentinels for packing items within arbitrary convex regions. Oper J Res Soc 57:735–746
15. Birgin EG, Martínez JM, Nishihara FH, Ronconi DP (2006) Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization. Comput Oper Res 33:3535–3548
16. Birgin EG, Martínez JM, Raydan M (2000) Nonmonotone spectral projected gradient methods on convex sets. J SIAM Optim 10:1196–1211
17. Birgin EG, Martínez JM, Raydan M (2001) Algorithm 813: SPG Software – for convex-constrained optimization. Trans ACM Math Softw 27:340–349
18. Birgin EG, Martínez JM, Raydan M (2003) Inexact Spectral Projected Gradient methods on convex sets. J IMA Numer Anal 23:539–559
19. Birgin EG, Martínez JM, Ronconi DP (2003) Minimization subproblems and heuristics for an applied clustering problem. Eur J Oper Res 146:19–34
20. Birgin EG, Martínez JM, Ronconi DP (2005) Optimizing the packing of cylinders into a rectangular container: A nonlinear approach. Eur J Oper Res 160:19–33
21. Brezinski C (2002) Block descent methods and hybrid procedures for linear systems. Numer Algorithm 29:21–32
22. Cauchy A (1847) Méthode générale pour la résolution des systèmes d'équations simultanées. Compte-Rendu de l'Académie des Sciences 27:536–538
23. Chehab JP, Raydan M (2005) Implicit and adaptive inverse preconditioned gradient methods for nonlinear problems. Appl Numer Math 55:32–47
24. Cores D, Loreto M (2007) A generalized two-point ellipsoidal anisotropic ray tracing for converted waves. Optim Eng 8:373–396
25. Crema A, Loreto M, Raydan M (2007) Spectral projected subgradient with a momentum term for the Lagrangean dual approach. Comput Oper Res 34:3174–3186
26. Curiel F, Vargas WE, Barrera RG (2002) Visible spectral dependence of the scattering and absorption coefficients of pigmented coatings from inversion of diffuse reflectance spectra. Appl Optics 41:5969–5978

27. Dai YH (2002) On the nonmonotone line search. J Optim Theory Appl 112:315–330
28. Dai YH (2003) Alternate step gradient method. Optimization 52:395–415
29. Dai YH, Fletcher R (2005) On the asymptotic behaviour of some new gradient methods. Math Program 103:541–559
30. Dai YH, Fletcher R (2005) Projected Barzilai–Borwein methods for large-scale box-constrained quadratic programming. Numer Math 100:21–47
31. Dai YH, Zhang HC (2001) Adaptive two-point stepsize gradient algorithm. Numer Algorithms 27:377–385
32. Dai YH, Hager WW, Schittkowski K, Zhang HC (2006) The cyclic Barzilai–Borwein method for unconstrained optimization. J IMA Numer Anal 26:604–627
33. Dai YH, Liao LZ (2002) R-linear convergence of the Barzilai and Borwein gradient method. J IMA Numer Anal 22:1–10
34. Dai YH, Yuan JY, Yuan YX (2002) Modified two-point stepsize gradient methods for unconstrained optimization. Comput Optim Appl 22:103–109
35. Deidda GP, Bonomi E, Manzi C (2003) Inversion of electrical conductivity data with Tikhonov regularization approach: some considerations. Ann Geophys 46:549–558
36. Dennis JE, Moré JJ (1977) Quasi-Newton methods, motivation and theory. Rev SIAM 19:46–89
37. Dennis JE, Schnabel RB (1983) Methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall, Englewood Cliffs
38. Diniz-Ehrhardt MA, Gomes-Ruggiero MA, Martínez JM, Santos SA (2004) Augmented Lagrangian algorithms based on the spectral projected gradient method for solving nonlinear programming problems. J Optim Theory Appl 123:497–517
39. Fletcher R (1987) Practical methods of Optimization. Wiley, New York
40. Fletcher R (1990) Low storage methods for unconstrained optimization. Lect Appl Math (AMS) 26:165–179
41. Fletcher R (2001) On the Barzilai–Borwein method. Department of Mathematics. University of Dundee NA/207, Dundee
42. Francisco JB, Martínez JM, Martínez L (2006) Density-based globally convergent trust-region methods for self-consistent field electronic structure calculations. J Math Chem 40:349–377
43. Friedlander A, Martínez JM, Molina B, Raydan M (1998) Gradient method with retards and generalizations. J SIAM Numer Anal 36:275–289
44. Friedlander A, Martínez JM, Raydan M (1995) A new method for large-scale box constrained convex quadratic minimization problems. Optim Methods Softw 5:57–74
45. Glunt W, Hayden TL, Raydan M (1994) Preconditioners for distance matrix algorithms. Comp J Chem 15:227–232
46. Goldstein AA (1964) Convex Programming in Hilbert Space. Bull Am Math Soc 70:709–710
47. Grippo L, Lampariello F, Lucidi S (1986) A nonmonotone line search technique for Newton's method. J SIAM Numer Anal 23:707–716
48. Grippo L, Sciandrone M (2007) Nonmonotone Derivative Free Methods for Nonlinear Equations. Comput Optim Appl 37:297–328
49. Hestenes MR, Stiefel E (1952) Methods of conjugate gradients for solving linear systems. Res J NBS 49:409–436
50. Jiang Z (2006) Applications of conditional nonlinear optimal perturbation to the study of the stability and sensitivity of the Jovian atmosphere. Adv Atmos Sci 23:775–783
51. La W Cruz, Raydan M (2003) Nonmonotone spectral methods for large-scale nonlinear systems. Optim Methods Softw 18:583–599
52. La Cruz W, Martínez JM, Raydan M (2006) Spectral residual method without gradient information for solving large-scale nonlinear systems of equations. Math Comput 75:1429–1448
53. Levitin ES, Polyak BT, Constrained Minimization Problems. Comput USSR Math Math Physics 6:1–50
54. Luengo F, Raydan M, Glunt W, Hayden TL (2002) Preconditioned Spectral Gradient Method. Numer Algorithms 30:241–258
55. Luengo F, Raydan M (2003) Gradient method with dynamical retards for large-scale optimization problems. Electron Trans Numer Anal (ETNA) 16:186–193
56. Martínez JM, Pilotta EA, Raydan M (2005) Spectral gradient methods for linearly constrained optimization. J Optim Theory Appl 125:629–651
57. Molina B, Raydan M (1996) Preconditioned Barzilai–Borwein method for the numerical solution of partial differential equations. Numer Algorithms 13:45–60
58. Oren SS (1974) On the selection of parameters in self-scaling variable metric algorithms. Math Program 7:351–367
59. Ramirez-Porras A, Vargas-Castro WE (2004) Transmission of visible light through oxidized copper films: feasibility of using a spectral projected gradient method. Appl Optics 43:1508–1514
60. Raydan M (1993) On the Barzilai and Borwein choice of steplength for the gradient method. J IMA Numer Anal 13:321–326
61. Raydan M (1997) The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. J SIAM Optim 7:26–33
62. Raydan M, Svaiter BF (2002) Relaxed steepest descent and Cauchy-Barzilai–Borwein method. Comput Optim Appl 21:155–167
63. Serafini T, Zanghirati G, Zanni L (2005) Gradient projection methods for quadratic programs and applications in training support vector machines. Optim Methods Softw 20:347–372
64. Vargas WE (2002) Inversion methods from Kiabelka–Munk analysis. J Opt A Pure Appl Opt 4:452–456
65. Vargas WE, Azofeifa DE, Clark N (2003) Retrieved optical properties of thin films on absorbing substrates from

transmittance measurements by application of a spectral projected gradient method. Thin Solid Films 425:1–8
66. Wang CY, Liu Q (2006) Convergence properties of inexact projected gradient methods. Optimization 55:301–310
67. Wang CY, Liu Q, Yang XM (2005) Convergence properties of nonmonotone spectral projected gradient methods. J Comput Appl Math 182:51–66
68. Yuan Y-X (2006) A new stepsize for the steepest descent method. J Comput Math 24:149–156
69. Zeev N, Savasta O, Cores D (2006) Non-monotone spectral projected gradient method applied to full waveform inversion. Geophys Prospect 54:525–534
70. Zhang L, Zhou WJ (2006) Spectral gradient projection method for solving nonlinear monotone equations. J Comput Appl Math 196:478–484
71. Zhou B, Gao L, Dai YH (2006) Gradient methods with adaptive step-sizes. Comput Optim Appl 35:69–86
72. Zhou B, Gao L, Dai YH (2006) Monotone projected gradient methods for large-scale box-constrained quadratic programming. Sci China Ser A Math 49:688–702

Splitting Method for Linear Complementarity Problems

PAUL TSENG
Department Math., University Washington, Seattle, USA

MSC2000: 90C25, 90C55, 90C25, 90C33

Article Outline

Keywords
See also
References

Keywords

LCP; Splitting method; SOR method

Splitting methods were originally proposed as a generalization of the classical SOR method for solving a system of linear equations [8,25], and in the late 1970s they were extended to the linear complementarity problem (LCP; cf. ► Linear complementarity problem) [1,2, Chap. 5], [10,13,18]. These methods are iterative and are best suited for problems in which exploitation of sparsity is important, such as large sparse linear programs and the discretization of certain elliptic boundary value problems with obstacle.

To describe the splitting methods, we formulate the LCP (with bound constraints) as the problem of finding an $x = (x_1, \ldots, x_n) \in \mathbf{R}^n$ solving the following system of nonlinear equations:

$$x = \max\left[l, \min[u, x - (Mx + q)]\right], \tag{1}$$

where $M = [m_{ij}]_{i,j=1,\ldots,n} \in \mathbf{R}^{n\times n}$, $q = (q_1, \ldots, q_n) \in \mathbf{R}^n$ and the lower bound $l = (l_1, \ldots, l_n)$ and upper bound $u = (u_1, \ldots, u_n)$ are given. (Here max and min are understood to be taken componentwise and we allow $l_i = -\infty$ or $u_i = \infty$ for some i. The case of $l_i = 0$ and $u_i = \infty$ for all i corresponds to the standard LCP.) In the splitting methods, we express

$$M = B + C$$

for some $B \in \mathbf{R}^{n\times n}$ and $C \in \mathbf{R}^{n\times n}$; then, starting with any $x \in \mathbf{R}^n$, we iteratively update x by solving the following equation for x':

$$x' = \max\left[l, \min[u, x' - (Bx' + Cx + q)]\right], \tag{2}$$

and then replacing x with x'. Thus, at each iteration, we effectively replace M and q in the original problem by, respectively, B and $Cx + q$ which we then solve to obtain the new iterate.

A key to the performance of the splitting methods lies in the choice of the matrix B. We should choose B to be a good approximation of M so that the methods have rapid convergence and, at the same time, such that x' is easy to compute at each iteration (e. g., B is diagonal or upper/lower triangular). The best known choice, corresponding to the SOR method of C. Hildreth [3,7], is

$$B = \frac{1}{\omega}D + L, \tag{3}$$

where D and L denote, respectively, the diagonal and the strictly lower-triangular part of M and $\omega \in (0, 2)$ is a relaxation parameter (see [2, p. 397], [13]). For this choice of B and assuming M has positive diagonal entries, the components of x' can be computed using n-step backsolve:

$$x_i' = \max\left[l_i, \min\left[u_i, x_i - \frac{\omega}{m_{ii}} \times \left(\sum_{j<i} m_{ij}x_j' + \sum_{j\geq i} m_{ij}x_j + q_i\right)\right]\right], \quad i = 1, \ldots, n.$$

In the case where $l_i = -\infty$ and $u_i = \infty$ for all i, the above iteration reduces to the classical SOR method for solving the system of linear equations $Mx + q = 0$. More generally, we can choose B to be block-lower/upper-triangular, e. g.,

$$B = \begin{pmatrix} B_{11} & & & \\ B_{21} & B_{22} & & \\ \vdots & \vdots & \ddots & \\ B_{p1} & B_{p2} & \cdots & B_{pp} \end{pmatrix}$$

for some $1 < p \leq n$, with the diagonal and triangular blocks possibly coming from M. Then, each block of components of x' can be computed recursively by solving an LCP of dimension equal to the block size. Other choices of B are discussed in [2, Chap. 5], [13] and below. Computation with the (block) SOR method on solving sparse linear programs and LCP with symmetric positive definite M is investigated in [1,4,14,16].

An original application of the SOR method is to the solution of convex quadratic programs of the form

$$\begin{cases} \min & \frac{1}{2} y^\top y \\ \text{s.t.} & Ay \leq b, \end{cases}$$

where $A \in \mathbf{R}^{n \times m}$ and $b \in \mathbf{R}^n$ are given, with A having nonzero rows. (Here, $^\intercal$ denotes the transpose.) Specifically, by attaching nonnegative Lagrange multipliers (cf. also ► Lagrangian multipliers methods for convex programming) $x = (x_1, \ldots, x_n)$ to the constraints $Ay \leq b$, we obtain the following dual problem in x:

$$\max_{x \geq 0} \left\{ \min_y \left\{ \frac{1}{2} y^\top y + x^\top (Ay - b) \right\} \right\} = \max_{x \geq 0} \left\{ -\frac{1}{2} x^\top A A^\top x - x^\top b \right\}$$

whose optimal solution, related to the optimal solution of the original problem by $y + A^\intercal x = 0$ [7], solves the LCP (1) with $M = AA^\intercal$, $q = b$ and $l_i = 0$, $u_i = \infty$ for all i [7, p. 4]. In this case, M is symmetric positive semidefinite with positive diagonal entries and x' computed in the SOR method is alternatively given by the formula:

$$\begin{aligned} \Delta_i &= \max \left[-x_i, \frac{\omega}{A_i A_i^\top} \left(A_i y^i - b_i \right) \right], \\ x_i' &= x_i + \Delta_i, \\ y^{i+1} &= y^i - A_i^\top \Delta_i, \qquad i = 1, \ldots, n, \end{aligned}$$

where $y^1 = -A^\intercal x$ and A_i denotes the ith row of A. The above iteration is reminiscent of the *Agmon–Motzkin–Fourier relaxation method* for solving the inequalities $Ay \leq b$ and, in fact, differs from the latter only in that the term $-x_i$, rather than zero, appears inside the max.

Convergence of the splitting methods, despite their relatively long history, was more fully analyzed only in the last ten years. In particular, if M is symmetric (not necessarily positive semidefinite) and the function

$$f(x) = \frac{1}{2} x^\top M x + q^\top x \tag{4}$$

is bounded below on the box $l \leq x \leq u$, then it is known that x generated by the splitting method (2) converges to a solution of the LCP at a linear rate (in the root sense [17]), provided that (B, C) is a *regular Q-splitting* in the sense that

$B - C$ is positive definite and for every x there exists a solution x' to (2)

[12, Thm. 3.2]. (Earlier results of this kind that further assumed M is positive semidefinite or nondegenerate can be found in [2 Chap. 5], [5,11,19,20] and references therein.) For the SOR method, corresponding to B given by (3) with $\omega \in (0, 2)$, it can be verified that (B, C) is a regular Q-splitting provided M has positive diagonal entries. The proof of the above convergence result uses two key facts about the LCP, namely, that $f(x)$ assumes only a finite number of values on the solution set and that the distance to the solution set from any point x near the solution set is in the order of the'residual' at x, defined to be the difference in the two sides of (1). In addition, the function $f(x)$ can be used in a line-search strategy to accelerate convergence of the splitting methods [2, Sec. 5.5].

If M is not symmetric but positive semidefinite, it is known that x generated by the splitting method (2) converges to a solution of the LCP at a linear rate (in the root sense), provided that

$B - M$ is symmetric positive definite

[2, Thm. 5.6.1], [24, Cor. 5.3]. One choice of B that satisfies the above assumption is

$$B = M + \widehat{D} - L - L^\top,$$

where L denotes the strictly lower-triangular part of M

and $\widehat{D}$ is any $n\times n$ diagonal matrix such that $\widehat{D}-L-L^{\top}$ is positive definite. This choice of B is upper-triangular and hence the corresponding x' can be computed in the order of n^2 arithmetic operations using n-step backsolve [22, Sec. 6], [24]. Computationally, the asymmetry of M makes it difficult to incorporate line-search strategies since no 'natural' merit function analogous to (4) is known. As a result, on problems where M is highly asymmetric, such as the LCP formulation of linear programs, the convergence of the splitting methods can be slow. Thus, accelerating convergence of the splitting methods on asymmetric problems remains a challenge. In this direction, we point out related methods based on projection or operator splitting (see [6,21] and references therein). These methods are applicable to the case where M is positive semidefinite (not necessarily symmetric) and the major part of their iterations also involves solving a matrix-splitting equation of the form (2), except the solution x' must undergo additional transformations to yield the new iterate x. These methods, which may be viewed as a hybrid form of the splitting methods, admit some forms of line search and show good promise in computation.

In summary, building on the early work of Hildreth and H.B. Keller and others, splitting methods have been well developed in the last twenty years to solve the LCP (1) when the matrix M is either symmetric or positive semidefinite. Computationally, these methods are best suited when M is symmetric, possibly having some sparsity structure (e. g., $M = AA^{\top}$ with A sparse), and the function (4) is used in a line-search strategy to accelerate convergence. Extensions of these methods to problems where the box $l \leq x \leq u$ is replaced by a general polyhedral set, including as special cases the *extended linear/quadratic programming problem* of R.T. Rockafellar and R.J-B. Wets and the quadratic program formulation of the LCP with *row sufficient matrix*, have also been studied [2, Sec. 5.5], [6,12,22,23]. Inexact computation of x' is discussed in [2, Sec. 5.7], [9,12,15]. Acceleration of the methods in the case where M is not symmetric remains an open issue. In fact, if M is not symmetric nor positive semidefinite, convergence of the splitting methods is known only for the case where M is an *H-matrix* with positive diagonal entries and B is likewise, with the comparison matrix of B having a contractive property [2, p. 418], [18,19]. Thus, even if M is a *P-matrix*, it is not known whether the splitting methods converge for some practical choice of B.

See also

► Linear Complementarity Problem

References

1. Cottle RW, Golub GH, Sacher RS (1978) On the solution of large, structured linear complementarity problems: the block partitioned case. Appl Math Optim, 4:347–363
2. Cottle RW, Pang J-S, Stone RE (1992) The linear complementarity problem. Acad. Press, New York
3. Cryer CW (1971) The solution of a quadratic programming problem using systematic overrelaxation. SIAM J Control Optim, 9:385–392
4. DeLeone R, Mangasarian OL (1988) Asynchronous parallel successive overrelaxation for the symmetric linear complementarity problem. Math Program, 42:347–361
5. DePierro AR, Iusem AN (1993) Convergence properties of iterative methods for symmetric positive semidefinite linear complementarity problems. Math Oper Res, 18:317–333
6. Eckstein J, Ferris MC (1994) Operator splitting methods for monotone affine variational inequalities, with a parallel application to optimal control. Techn. Report Thinking Machines Corp
7. Hildreth C (1957) A quadratic programming procedure. Naval Res Logist Quart, 4:79–85
8. Keller HB (1965) On the solution of singular and semidefinite linear systems by iteration. SIAM J Numer Anal, 2:281–290
9. Li W (1993) Remarks on convergence of matrix splitting algorithm for the symmetric linear complementarity problem. SIAM J Optim, 3:155–163
10. Lin YY, Pang J-S (1987) Iterative methods for large convex quadratic programs: A survey. SIAM J Control Optim, 25:383–411
11. Luo Z-Q, Tseng P (1991) On the convergence of a matrix: splitting algorithm for the symmetric monotone linear complementarity problem. SIAM J Control Optim, 29:1037–1060
12. Luo Z-Q, Tseng P (1992) Error bound and convergence analysis of matrix splitting algorithms for the affine variational inequality problem. SIAM J Optim, 2:43–54
13. Mangasarian OL (1977) Solution of symmetric linear complementarity problems by iterative methods. J Optim Th Appl, 22:465–485
14. Mangasarian OL (1984) Sparsity-preserving SOR algorithms for separable quadratic and linear programming. Comput Oper Res, 11:105–112
15. Mangasarian OL (1991) Convergence of iterates of an inexact matrix splitting algorithm for the symmetric monotone linear complementarity problem. SIAM J Optim, 1:114–122

16. Mangasarian OL, DeLeone R (1988) Parallel gradient projection successive overrelaxation for symmetric linear complementarity problems and linear programs. Ann Oper Res, 14:41–59
17. Ortega JM, Rheinboldt WC (1970) Iterative solution of nonlinear equations in several variables. Acad. Press, New York
18. Pang J-S (1982) On the convergence of a basic iterative method for the implicit complementarity problem. J Optim Th Appl, 37:149–162
19. Pang J-S (1984) Necessary and sufficient conditions for the convergence of iterative methods for the linear complementarity problem. J Optim Th Appl, 42:1–17
20. Pang J-S (1986) More results on the convergence of iterative methods for the symmetric linear complementarity problem. J Optim Th Appl, 49:107–134
21. Solodov MV, Tseng P (1996) Modified projection-type methods for monotone variational inequalities. SIAM J Control Optim, 34:1814–1830
22. Tseng P (1990) Further applications of a splitting algorithm to decomposition in variational inequalities and convex programming. Math Program, 48:249–263
23. Tseng P (1991) Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. SIAM J Control Optim, 29:119–138
24. Tseng P (1995) On linear convergence of iterative methods for the variational inequality problem. J Comput Appl Math, 60:237–252
25. Young DM (1971) Iterative solution of large linear systems. Acad. Press, New York

SSC Minimization Algorithms

supervisor and searcher cooperation minimization algorithms, SSC

Wenbin Liu, K. Sirlantzis
University Kent, Canterbury, England

MSC2000: 90C30, 90C15, 90C99

Article Outline

Keywords
See also
References

Keywords

Optimization with noises; Hybrid algorithm; Robustness

SSC algorithms are a class of local minimization algorithms developed within the framework of supervisor and searcher cooperation (SSC). One of the distinct characteristics of this class of algorithms is that they are both efficient and robust, and therefore suitable for minimization problems with strong noises. Some of these algorithms have been studied in [5] and [6].

Most of the existing fast minimization algorithms are not suitable for noisy minimization problems. One of the main reasons seems that the effectiveness of the line search procedures, which are usually incorporated into these algorithms, are rather sensitive to the accuracy of function or gradient value evaluations. One may increase the accuracy of these evaluations by, for example, using an averaging process to diminish the effects of stochastic noises. However this will normally increase the computational work required.

Most of the existing robust minimization algorithms, such as the stochastic approximation (SA) method, the N-M simplex method, the genetic algorithms, the Hooke–Jeeves method, do not use the classical line search, see [7] and [2]. Consequently they have been widely employed to tackle noisy optimization problems. Unfortunately, these algorithms are, in general, very slow and inefficient even for the noise free case. The stochastic approximation method, for example, is very robust. Its global convergence has been established under various assumptions for the noise. However, even for deterministic problems without noises, SA has been known to be very slow in comparison with efficient gradient algorithms like the conjugate gradient (CG) method and the GBB algorithm (see [8]). This is mainly due to the use of pre-assigned stepsizes in the search for optimizers.

The SSC framework provides an effective mechanism to address the above issues. The essential idea adopted in SSC algorithms is to combine the desirable characteristics of two algorithms: a supervisor algorithm (SR) and a searcher (or search engine) algorithm (SE). The former is used to ensure global convergence and to enhance robustness, while the latter is employed to increase the efficiency of the solution searching work. The SSC framework can be viewed as a systematic way of exploring possible combinations of some existing minimization algorithms. The resulting algorithms are a class of piecewise or hybrid algorithms.

Assume that we wish to minimize an objective function $f(x)$ on $\mathbf{R}^n$. Assume, for given $x_0, \ldots, x_l$, that we have an iterative algorithm, called *search engine* (SE):

$$x_{k+1} = x_k - se_k(x_k, \ldots, x_{k-l}, k, f),$$
$$k = l, l+1, \ldots.$$

This particular form of se_k may depend on k and the values (or estimates) of $\{f(x_{k-i})\}_{i=0}^{l}$, $\{\nabla f(x_{k-i})\}_{i=0}^{l}$, $\{H(x_{k-i})\}_{i=0}^{l}$, etc, where $H(x)$ is the Hessian matrix of f at the point x. Suppose that this algorithm is convergent to a local minimizer of f provided the starting points are close enough to the minimizer.

Assume that we also have a *supervisor algorithm* (SR): Given $x_0, \ldots, x_l$

$$x_{k+1} = x_k - sr_k(x_k, \ldots, x_{k-l}, k, f), \qquad k = l, l+1, \ldots.$$

In general, an SR is slow but robust, while an SE is fast but only convergent locally. For the two algorithms to work together efficiently, a principle is needed to regulate their relationship. In [5] the following framework was introduced, based on the idea of cooperation between the supervisor and the searcher (SSC): the supervisor acts only if it believes that the performance of the search engine is not satisfactory while the search engine does most of the solution searching work.

There are many different ways of designing new algorithms in this framework. In [5] the following simple implementation has been proposed: Assume $f \geq 0$. Given $x_0, \ldots, x_l$, define $(k = \mathrm{l}, \mathrm{l}+1, \ldots)$ the following (SSC) algorithm:

$$x_{k+1} = \begin{cases} x_k - sr_k & \text{if } T_k f(x_k - sr_k) \\ & \quad \leq f(x_k - se_k), \\ x_k - se_k & \text{otherwise,} \end{cases}$$

where $\{T_k\}$ is a given sequence of nonnegative real numbers.

Note that as far as minimization is concerned, one can always assume that $f \geq 0$ by adding a positive constant to the original function. Alternatively, one can use the following SSC algorithm for the general case:

$$x_{k+1} = \begin{cases} x_k - sr_k & \text{if } T_k^{\mathrm{sign}(f(x_k - sr_k))} \\ & \quad \times f(x_k - sr_k) \\ & \quad \leq f(x_k - se_k), \\ x_k - se_k & \text{otherwise.} \end{cases}$$

where $\{T_k\}$ is a given sequence of nonnegative real numbers.

Other forms of implementation or extensions have also been studied in [5,6]. For example, the following nonmonotone version of SSC algorithm was proposed in [6] (NOMSCC algorithm): For a fixed positive integer $m > 0$ and a real number $\delta > 0$, define x_k as follows:

$$x_{k+1} = \begin{cases} x_k - se_k & \text{if } f(x_k - se_k) \leq \max\limits_{0 \leq j \leq m(k)} \\ & \quad \{f(x_{k-j})\} - \epsilon_k |g_k|^2 \\ & \text{or if } f(x_k - se_k) \\ & \quad \leq T_k f(x_k - sr_k), \\ x_k - sr_k & \text{otherwise,} \end{cases}$$

where $g_k = \nabla f(x_k)$, $m(k) = \min(k, m)$ and $\{\epsilon_k\}$ is a given sequence of nonnegative real numbers such that $\epsilon_k \geq \delta$ $(k = 0, 1, \ldots)$.

It was found that in some cases the nonmonotone version performs better than the simpler original version (see [5]).

In principle, any globally convergent iterative algorithm, e. g. CG, BFGS, etc., could be used as a supervisor. In general, one wishes that supervisors should have simpler structures to increase robustness. Two classes of supervisors have been examined in [6].

The first class uses pre-assigned steplengths as in SA. Let $\{t_k\}(k = 0, 1, \ldots)$ be such that

i) $t_k > 0$;

ii) $\sum_{k=0}^{\infty} t_k = +\infty$.

Let $x_k \in \mathbf{R}^n$ $(k = 0, 1, \ldots)$ and let $g_k = \nabla f(x_k)$. Let $l \geq 0$ and let $\{d_k(x_k, \ldots, x_{k-l}, f)\}$ be a sequence of $\mathbf{R}^n$ vectors so that there exist $c, C > 0$ such that the following assumptions hold:

$$\text{(D)} \quad d_k^{\top} g_k \geq c\,|g_k|^2\,;\ |d_k| \leq C\,|g_k|\,,\ k = 0, 1, \ldots.$$

One of the most frequently used forms of d_k in this paper is $d_k = g_k$, though there are other possible choices as well.

The first class of supervisors have the following form: for given $x_0, \ldots, x_l$,

$$x_{k+1} = x_k - t_k d_k, \quad k = l, l+1, \ldots.$$

When $d_k = g_k$, it is the SA algorithm.

The second class of supervisors have the following form: for given $x_0, \ldots, x_l$,

$$x_{k+1} = x_k - t_k d_k, \quad k = l, l+1, \ldots,$$

where $\{t_k\}$, $\{d_k\}$ are chosen so that there exists an $\alpha > 0$, independent of k, such that

$$f(x_k - t_k d_k) - f(x_k) \leq -\alpha \frac{(d_k^\top \nabla f(x_k))^2}{|d_k|^2},$$

and

$$\sum_l^\infty \cos^2(\theta_k) = +\infty,$$

where

$$\cos(\theta_k) = \frac{(g_k, d_k)}{|g_k|\,|d_k|}, \quad k = l, l+1, \ldots.$$

For instance, let $d_k = g_k$ and let $\{t_k$ be generated using the Wolfe line search procedure. Then the above assumptions hold. Clearly other line search methods can also be considered in the same way.

Again, in principle, any locally convergent iterative algorithm could be used as a search engine. Some possible examples, which have been already investigated, are as follows:

1) *Newton search engine*: Given x_0,

$$x_{k+1} = x_k - se_k, \quad k = 0, 1, \ldots,$$

where $se_k = H_k^{-1} g_k$, and H_k and g_k are the Hessian matrix and gradient of f at the point x_k. If H_k is not invertible, then the SSC switches to SR.

2) *Quasi-Newton search engine*: Given x_0,

$$x_{k+1} = x_k - se_k, \quad k = 0, 1, \ldots,$$

where $se_k = B_k g_k$ and B_k is given by a quasi-Newton recursive formula (see [4] and [3] for the details). It has been found that a straightforward use of this class of search engines may lead to poor performance of the resulting algorithms.

3) *BB search engine*: Given x_0,

$$x_{k+1} = x_k - se_k, \quad k = 0, 1, \ldots,$$

where $se_k = \alpha_k g_k$, where $\alpha_0 = 1$ and for $k \geq 1$

$$\alpha_k = \frac{|y_k|^2}{y_k^\top s_k},$$

where

$$y_k = x_k - x_{k-1},$$
$$s_k = \nabla f(x_k) - \nabla f(x_{k-1}).$$

This steplength α_k was first proposed in [1], where the BB algorithm was introduced. This search engine has been studied in detail in [5].

Three SSC algorithms have been studied and tested in [6]: SSC-SABB, SSC-SANEWTON, and SSC-SABFGS, which use SA as the supervisor, and, BB, Newton's, and BFGS as the search engines, respectively. For example the SSC-SABB algorithm is as follows:

Let $x \in \mathbf{R}^n$ be given and $\alpha_0 = 1$.
Let $T_k \geq 0$, $k = 0, 1, \ldots$, be given, and assume that $f \geq 0$.
Then (the SSC-SABB algorithm):

$$x_{k+1} = x_k - \begin{cases} t_k g_k & \text{if } T_k f(x_k - t_k g_k) \leq f(x_k - \alpha_k g_k), \\ \alpha_k g_k & \text{otherwise,} \end{cases}$$

where, for $k \geq 1$,

$$\alpha_k = \frac{|y_k|^2}{y_k^t s_k},$$

with

$$y_k = x_k - x_{k-1},$$

$$s_k = \nabla f(x_k) - \nabla f(x_{k-1}).$$

These algorithms were found to be efficient and very robust. For instance, SSC-SABB was able to solve difficult stochastic optimization problems efficiently, while for the noise free case, it was comparable with some fastest gradient algorithms like CG and GBB (see [5]). In [6], it was also reported that SSC-SANEWTON and SSC-SABFGS can solve difficult optimization problems, using rough approximations of the gradient and the Hessian, while they are very fast when the errors are small. In these experiments, $t_k = \min(0.01, \frac{1}{\sqrt{k}})$ and $t_k = \min(0.01, 1/k)$ are tested, while connection T_k is fixed as a constant T>0. For example, one may take $T = 3$ or $T = 5$ when there is little noise in an optimization problem, and one has to let $T = 1$ if the noise is strong.

In the noise free case the following global convergence result has been established in [6] for SSC algorithms.

Let f be twice continuously differentiable and bounded below. Assume that ∇f is Lipschitz with a global Lipschitz constant. Let $\{x_k\}$ be generated by the

SSC algorithm. Assume that $\prod_0^\infty \max(T_k, 1) < \infty$. If the supervisor belongs to the first class, then there is an $\epsilon(f) > 0$ such that $\{\sum_l^k t_k |\nabla f(x_k)|^2\}$is convergent as $k \to \infty$ for any sequence $\{t_k\}$ such that there is a $N > 0$ satisfying that $t_k \le \epsilon(f)$ after $k \ge N$. If the supervisor belongs to the second class, then $\{\sum_l^k \cos^2(\theta_k) |\nabla f(x_k)|^2\}$ is convergent as $k \to \infty$.

It should be noted that the above convergence is independent of the search engines used. The result can be further extended. For instance, SSC-SABB has been shown to be globally convergent when only finite $T_k = 1$ and the rest $T_k = T > 1$. It was also shown that the speed of SSC-SABB is R-linear.

In general, the SSC algorithms are at least as fast as their SE provided $T_k > T > 1$ after k large enough. Therefore, the global convergence of SSC algorithms is ensured by the SR, while their speed is largely decided by the SE. More details can be found in [6].

The above basic SCC algorithms have been extended for different applications, like training of feedforward neural networks (FNNs). For example, the following extended version of the SSC algorithms, which forces the search engine to run m iteration before attempting switching, is proposed. Let $x_0 \in \mathbf{R}^n$ be given. Let $T_k \ge 0$ be given for $k = 0, 1, \ldots$, and let $m > 0$. Assume that $f \ge 0$ and that we have x_k. Then define

$$x_{k+m}^1 = x_k - sr_k,$$
$$x_{k+m}^2 = y_{k+m},$$

where y_{k+m} is defined by $y_k = x_k$,

$$y_{k+l+1} = y_{k+l} - se_{k+l}, \quad l = 0, \ldots, m-1.$$

Then define

$$x_{k+1} = \begin{cases} x_{k+m}^1 & \text{if } T_k f(x_{k+m}^1) \\ & \quad \le f(x_{k+m}^2). \\ x_{k+m}^2 & \text{otherwise.} \end{cases}$$

The purpose of the above extension is to create a 'memory' effect which may be important for the efficient performance of some search engines.

It seems that the supervisor-searcher cooperation framework offers a promising way for combining characteristics of existing algorithms. It seems that the new algorithms can retain the desirable properties of their 'parents'. Consequently these algorithms are both efficient and robust. This makes them suitable for both deterministic and stochastic minimization problems. This unique feature is largely due to the fact that the algorithms use neither line search nor pre-assigned stepsizes *all* the time. Finally, the additional computational work required is also very light.

See also

- ► Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- ► Inequality-constrained Nonlinear Optimization
- ► SSC Minimization Algorithms for Nonsmooth and Stochastic Optimization

References

1. Barzilai J, Borwein M (1988) Two point step size gradient methods. IMA J Numer Anal, 8:141–148
2. Conn AR, Scheinberg K, Toint PL (1997) Recent progress in unconstrained nonlinear optimization without derivatives. Math Program, 79:397–414
3. Dennis JE, Schnabel RB (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs, NJ
4. Fletcher R (1987) Practical methods of optimization, 2nd edn. Wiley, New York
5. Liu WB, Dai YH (2000) Minimization algorithms based on supervisor and searcher co-operation: I – Faster and robust gradient algorithms for minimization problems with strong noises. JOTA, accepted
6. Liu WB, Dai YH (2000) Minimization algorithms based on supervisor and searcher co-operation: II – Faster and robust gradient algorithms for minimization problems with strong noises. JOTA, submitted
7. Powell MJD (1998) Direct search algorithms for optimization calculations. Acta Numer 7(287–336
8. Raydan M (1993) On the BB choice of steplength for the gradient method. IMA J Numer Anal, 13:321–326

SSC Minimization Algorithms for Nonsmooth and Stochastic Optimization

SSC

Wenbin Liu[1], F. J. Feng[2]

[1] University Kent, Canterbury, England

[2] University Sussex, Sussex, England

MSC2000: 90C30, 90C15, 90C99

Article Outline

Keywords
See also
References

Keywords

Stochastic optimization; Nonsmooth optimization; Global optimization; Robustness

SSC algorithms are a class of local minimization algorithms developed within the framework of supervisor and searcher cooperation, see ► SSC minimization algorithms for definitions. Some of the algorithms are both efficient and robust, and therefore suitable for minimization problems with strong noise. Further studies of SSC algorithms can be found in [4,5], and [7].

Nonsmooth optimization problems, where the objective functions may not be differentiable, are frequently met in many important applications. There exist some minimization algorithms, which are applicable to certain types of nonsmooth optimization problems, such as the subgradient method, the conjugate subgradient method, and the cutting plane method (see, [3,8], and [9]). However, more research is still much needed to develop fast minimization algorithms for nonsmooth optimization problems.

Some of the SSC algorithms have been extended to nonsmooth optimization problems, see [6]. It is clear that the SSC framework described in ► SSC minimization algorithms can be used to design minimization algorithms for nonoptimization problems, though it is not straightforward to select suitable supervisors and searchers. In [6], the SSC-SABB (see ► SSC minimization algorithms) algorithm was extended to nonsmooth optimization: the supervisor was replaced by the subgradient method, while the BB search engine was still used. This gives the *SSC-SBB algorithm*, described below.

Assume that we wish to minimise an objective function $f(x)$ on $\mathbf{R}^n$. Assume that $f \geq 0$ is Lipschitz. Let $t_k \geq 0$ $(k = 0, 1, \ldots)$ be such that $\sum_0^\infty t_k = \infty$ and $\sum_0^\infty t_k^2 < \infty$. Define $\partial f(x)$, the *general gradient* at x, as in [9]:

$$\partial f(x) = \operatorname{conv}(V_f(x)),$$

with

$$V_f(x) = \left\{ v \in \mathbb{R}^n : \begin{array}{c} v = \lim_{x_i \to x} \nabla f(x_i), \\ \text{where } \nabla f(x_i) \text{ exists} \end{array} \right\},$$

where $\operatorname{conv}(V_f(x))$ represents the close convex hull of $V_f(x)$. Since every Lipschitz function on $\mathbf{R}^n$ is differentiable almost everywhere, the above general gradient is well-defined. The *SSC-SBB algorithm* is then defined as follows: For given x_0 and $T_k > 0$ $(k = 0, 1, \ldots)$, define

$$x_{k+1} = \begin{cases} y_{k+1} = x_k - t_k \xi_k & \text{if } T_k f(y_{k+1}) \\ & \quad \leq f(z_{k+1}), \\ z_{k+1} = x_k - \alpha_k \xi_k & \text{otherwise,} \end{cases}$$

where $\xi_k \in V_f(x_k)$, $\alpha_0 = 1$, and for $k \geq 1$

$$\alpha_k = \frac{|y_k|^2}{y_k^t s_k},$$

where

$$y_k = x_k - x_{k-1}, \quad s_k = \xi_k - \xi_{k-1}.$$

Note that in the above definition, ξ_k is taken from $V_f(x_k)$, instead of $\partial f(x_k)$ as in the subgradient method. In general, this causes no extra computational work. For instance, for the most common case where $f(x) = \max_{1 \leq i \leq J}(f_i(x))$, where f_i $(i = 1, \ldots, J)$ is smooth, one can simply use $\xi_k = \nabla f_j(x_k)$, where j is such that $f_j(x_k) = \max_{1 \leq i \leq J}(f_i(x_k))$. In fact, it is also possible to simply require $\xi_k \in \partial f(x_k)$ with $|\xi_k| = 1$ in the above definition, as in the subgradient method. Then the formula of α_k has to be modified because of the normalization of ξ_k.

Some global convergence results have been established in [6] for the SSC-SBB algorithm. The following is just an example. Assume that

a) f is continuous and strictly convex;
b) x^* is the global minimizer of f such that $f(x^*) = \min_{x \in R^n} f(x) > 0$; and
c) there exist constants $c, m > 0$ such that for all $x \in \mathbf{R}^n$

$$c\,|x - x^*| \leq f(x) - f(x^*) \leq m\,|x - x^*|\,.$$

Assume that $\{\beta_k\}$ is a sequence of real numbers. Let x_0 be given and let $T_k = T > 0$ $(k = 0, 1, \ldots)$. Let $\{x_k\}$ be defined by $(k = 1, 2, \ldots)$

$$x_{k+1} = \begin{cases} y_{k+1} = x_k - t_k \xi_k & \text{if } T f(y_{k+1}) \\ & \quad \leq f(z_{k+1}), \\ z_{k+1} = x_k - \beta_k \xi_k & \text{otherwise,} \end{cases}$$

where $\xi_k \in \partial f(x_k)$ with $|\xi_k = 1$, and $t_k \geq 0$ such that $\sum_0^\infty t_k = \infty$, and $\sum_0^\infty t_k^2 < \infty$. Then $\lim_{k\to\infty}|x_k - x^*| = 0$ if $T \leq c/m$. If we further have $U = \sup_{x,y}(f(x) - f(y)) < \infty$, then for any $0 < T < 1$ there exists a $C_T > 0$ such that $\lim_{k\to 0} |x_k - x^*| = 0$, where x_k is generated from the above SSC-SBB algorithm for any shifted objective function $f_C = f + C$, where $C \geq C_T$.

The above result basically says that if $0 < T < 1$, then by adding a suitable positive number C to the objective function f (this does not change the minimizer of f), the SSC-SBB algorithm will converge. It does not, however, give any estimate for the constant C. In [6], convergence was observed for all $C > 0$. More general convergence results have also been established in [6], where some of the above conditions have been further relaxed.

Numerical experiments indicated that the SSC-SBB is quite fast when $\{x_k\}$ are far away from a minimizer. It then slows down when the approximations are near the minimizer, if it happens that the objective function is not differentiable at that point. The main reason seems that the BB search engine is frequently redundant in the computation when x_k is near a nonsmooth minimizer. This problem is yet to be solved. Nevertheless, in some cases, a considerable improvement on the overall speed over the subgradient method has been observed in [6].

It has been found that some SSC algorithms seem quite suitable for stochastic optimization:

$$\begin{cases} \min & f(x), \\ \text{with} & f(x) = F(x) + \epsilon, \end{cases}$$

where F is the smooth underlying exact mathematical model, and ϵ is stochastic noise, which may depend on x. Of course, here we really mean to find a minimizer of F. Among the algorithms, a stochastic version of SSC-SABB has been much studied and tested, see, [4] and [7].

Assume that we have some estimators for $F(x)$ and nabla; $F(x)$. Whenever we refer to $f(x)$ and $\nabla f(x)$, we mean the estimators of the value or the gradient of F at the point x. When $\epsilon = 0$, all the estimators are assumed to be identical to the exact values.

Then the stochastic version of SSC-SABB is as follows: Let $x_0 \in \mathbf{R}^n$ be given and $\alpha_0 = 1$. For $k = 0, 1, \ldots$, let $T_k \geq 0$ be given. Define the *stochastic SSC-SABB algorithm*

$$x_{k+1} = \begin{cases} x_k - t_k g_k & \text{if } T_k f(x_k - t_k g_k) \\ & \quad \leq f(x_k - \alpha_k g_k), \\ x_k - \alpha_k g_k & \text{otherwise,} \end{cases}$$

where $g_k = \nabla f(x_k)$, and for $k \geq 1$,

$$\alpha_k = \frac{|y_k|^2}{y_k^t s_k},$$

with

$$y_k = x_k - x_{k-1}, \quad s_k = \nabla f(x_k) - \nabla f(x_{k-1}).$$

Actually it has the exactly same form as the deterministic version of SSC-SABB (see ► SSC minimization algorithms) but with a different interpretation for f and ∇f. More importantly, the parameters have to be selected differently. For example, it has been found in [4] that if one takes $t_k = \min(c, \frac{1}{\sqrt{k}})$ or $t_k = \min(c, 1/k)$, then the constant c may be very much problem dependent to ensure a fast convergence. It was reported in [4] and [7] that $T_k \equiv 1$ was always a safe choice for the stochastic problems studied there. One may use any estimator for F or ∇F in the above algorithm. In [4] and [7], the linear or quadratic approximation for F, and the second order central difference estimator for ∇F have been tested.

Performance of the algorithm was found to be promising. A considerable improvement over the stochastic approximation algorithm (SA) was observed in [4] and [7]. It was found to be able to solve some hard stochastic optimization problems efficiently. This improvement seems due to its unique adaptive (switching) feature in calculating its steplengths – they are neither pre-assigned nor determined by line search.

However, this unique feature leads to a major difficulty in establishing convergence for the algorithm (indeed for this class of algorithms) in the stochastic case. In the literature, most existing convergence results for the SA type of algorithms are established on the assumption that the size of the steplengths tends to zero. For instance, in the case of the stochastic approximation algorithm, the steplength at the kth iteration is just t_k, which goes to zero as $k \to \infty$. However this assumption is certainly not true for the stochastic version of SSC-SABB, since at the kth iteration, the

steplength is either t_k or α_k. In fact, it was observed that the steplengths in the stochastic version of SSC-SABB often jumped, and this actually speeded up the algorithm.

Some convergence results were established for this algorithm in [2], based on the observation that the shifted minimization problem: $\min_{R^n}(f + C)$ has weaker noise relative to the value of $F + C$, if C is larger. One of the convergence results will be very briefly described below. For ease of exposition, we will only state the result for the one-dimensional case.

Assume that $\{\xi_k\}_{\{k \geq 1\}}$, $\{\eta_k\}_{\{k \geq g1\}}$, $\{\zeta_k\}_{\{k \geq 1\}}$, and $\{\chi_k\}_{\{k \geq 1\}}$ are independent random sequences defined on the probability space $(\Omega, \mathrm{P}, \mathcal{F})$ with bounded variances. For $n = 1, 2, \ldots$, define the sub-σ-algebra $\mathcal{F}_n$ by

$$\mathcal{F}_n = \sigma(\{\xi_k\}_{\{k \leq n\}}, \{\eta_k\}_{\{k \leq n\}}, \{\zeta_k\}_{\{k \leq n\}}, \{\chi_k\}_{\{k \leq n\}}).$$

Assume that $\chi_k \sim \mathrm{N}(0, \gamma(k))$ and $\zeta_k \sim N(0, \theta(k))$ for $\gamma(k) > 0$ and $\theta(k) > 0$, respectively. Assume that the function $F \in C^2(\mathbf{R})$ with bounded second derivatives. Let $\{t_k\}$ be defined as before. Let $\{\beta_k\}$ be a sequence of positive numbers, and let $T > 0$. Then for a given starting point $X_0 = x_0$, define the random process ($k = 1, 2, \ldots$)

$$X_{k+1} = Y_{k+1}$$

if

$$T[F(X_k - t_k(\nabla F(X_k) + \xi_{k+1})) + \zeta_{k+1}] \leq [F(X_k - \beta_k(\nabla F(X_k) + \eta_{k+1})) + \chi_{k+1}],$$

and otherwise by

$$X_{k+1} = Z_{k+1},$$

where

$$\begin{cases} Y_{k+1} = X_k - t_k(\nabla F(X_k) + \xi_{k+1}), \\ Z_{k+1} = X_k - \beta_k(\nabla F(X_k) + \eta_{k+1}). \end{cases}$$

Suppose that $\{\xi_k, \mathcal{F}_k\}$ is a sequence of martingale differences. Then it was shown in [2] that for any given $T < 1$, one has that

$$\lim_{k \to \infty} F(X_k)$$

exists almost surely, and

$$P(\liminf_{k \to \infty} \mathrm{dis}(X_k, \Omega) = 0) = 1,$$

provided C is large enough and

$$\gamma(k) + \theta(k) \leq \frac{[(1 - T)C]^2}{10 \log(k + 1)},$$

where

$$\Omega = \{x : \nabla F(x) = 0\}.$$

Convergence has also been established when all the noises are bounded, see [2]. It should be noted that the above convergence is independent of the selection of the sequence $\{\beta_k\}$. However, the speed of the algorithm will very much depend on the selection. The assumptions of the results are in fact quite weak, and can be met easily, e. g. by averaging two or three samples of χ_k and ζ_k, or by taking a larger value of C. For instance, in [4], C was fixed to be 250 when $F \geq 0$.

See also

- ► Equality-constrained Nonlinear Programming: KKT Necessary Optimality Conditions
- ► Inequality-constrained Nonlinear Optimization
- ► SSC Minimization Algorithms

References

1. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley/Interscience, New York
2. Feng FJ, Liu WB, Sirlantzis K: Minimization algorithms based on supervisor and searcher co-operation: III–New fast algorithms for stochastic optimization (in preparation)
3. Kelley JE (1960) The cutting plane method for solving convex programs. J SIAM, 8:703–712
4. Liu WB, Dai YH: Minimization algorithms based on supervisor and searcher co-operation: I–Fast and robust gradient algorithms for minimization problems with strong noises. JOTA (accepted)
5. Liu WB, Dai YH: Minimization algorithms based on supervisor and searcher co-operation: II–A class of novel algorithms for minimization problems with strong noises. JOTA (submitted)
6. Liu WB, Liao LZ: Minimization algorithms based on supervisor and searcher co-operation: IV–New subgradient algorithms for non-smooth optimization (in preparation)
7. Liu WB, Sirlantzis K: Novel gradient algorithms and their extensions for deterministic and stochastic minimization problems – I. Math Oper Res (submitted)

8. Polyak BT (1978) Subgradient method: A survey of Soviet research. In: Lemarechal C, Mifflin R (eds) Nonsmooth Optimization. Pergamon, Oxford, pp 5–30
9. Shor NZ (1968) On the rate of convergence of the generalized gradient descent. Kibernetika, 3:98–99 (in Russian)

Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems

SUVRAJEET SEN, JULIA L. HIGLE
Systems and Industrial Engineering,
University Arizona, Tucson, USA

MSC2000: 90C15, 90C06

Article Outline

Keywords
Introduction
Alternative Methods
for Approximating The Recourse Function
Two Basic Tools: Incumbent Solutions
and Regularization
Regularization of Deterministic Cutting Plane
Methods
A Regularized Stochastic Decomposition Algorithm
Conclusions
Thanks
See also
References

Keywords

Cutting plane algorithm; Stochastic programming

Decomposition methods for stochastic linear programming problems often use cutting planes to develop piecewise linear approximations of the objective function. This article provides a summary of techniques that can be used to stabilize these algorithms. By providing a unified treatment of deterministic as well as stochastic cutting plane algorithms, this article provides a common ground for both classes of methods.

Introduction

Stochastic programming problems often give rise to highly structured optimization problems that are solved using decomposition techniques. Many of these methods rely on cutting plane algorithms that generate successive approximations of the objective function. By and large, these algorithms generate deterministic approximations (e. g. [1,14]). More recently, new randomized versions of cutting plane methods have also been studied ([6] and [10]). The approximations used by such algorithms (deterministic or stochastic) are piecewise linear functions, which have the potential to provide arbitrarily close approximations, especially near an optimum solution. However, these basic methods suffer from the following drawback: cutting planes are derived in each iteration, and proofs of convergence are often based on retaining all cuts generated during the course of the algorithm. Since the cuts often result in dense linear inequalities, the unfettered proliferation of cuts leads to scarcity of computer memory. On the other hand, deleting cutting planes indiscriminately leads to instability in the approximations, and consequently in the solutions as well. This article is devoted to the discussion of algorithmic methods to curtail the size of the approximations without a degradation in the convergence properties of the algorithm.

Cutting plane algorithms for the solution of stochastic linear programming (SLP) problems may use one of a class of alternative representations of the SLP objective function. There is a great deal of flexibility in the manner in which cutting plane algorithms are designed. For stochastic programming problems, each iteration may generate anywhere from 1 to S cutting planes, where S is the number of possible outcomes associated with the random events of interest. Our discussion of stabilization methods will focus on two basic tools: incumbent solutions and regularization ([12,13] and [7]). These tools are then used within deterministic as well as stochastic cutting plane algorithms to provide well defined schemes for the deletion of cutting planes that do not compromise the integrity of the algorithm. For problems with n decision variables, deterministic algorithms that generate q cuts ($1 \leq q \leq S$) can be shown to be convergent using a maximum of $n + 2q$ deterministic cuts. In case of a stochastic cutting plane algorithm that uses q cuts per iteration, convergence results can be obtained by using a maximum of $n + 3q$ cuts. We conclude this article with an illustration of these stabilization techniques.

Alternative Methods for Approximating The Recourse Function

A two-stage stochastic linear program (SLP) may be stated as

$$\text{(SLP)} \quad \begin{cases} \min & c^\top x + \mathsf{E}[h(x,\widetilde{\omega})] \\ \text{s.t.} & x \in X, \end{cases}$$

where $\widetilde{\omega}$ is a random variable and

$$h(x,\omega) = \begin{cases} \min & g^\top y \\ \text{s.t.} & Wy = r(\omega) - T(\omega)x, \\ & y \geq 0, \end{cases}$$

and $X \subseteq \mathbf{R}^n$ is a convex polyhedral set. Note that h is a value function of a linear program. For the sake of simplicity in this presentation, we assume that for all $x \in X$, $h(x,\widetilde{\omega}) < \infty$ with probability one.

Most deterministic methods for SP work with the entire sample space of scenarios. For these methods it is therefore customary to to assume that the random variable $\widetilde{\omega}$ is discrete, so that the possible outcomes may be numbered $\{\omega_s\}_{s=1}^S$. In such cases, the expectation in the objective function may be written as

$$E[h(x,\widetilde{\omega})] = \sum_{s=1}^{S} h(x,\omega_s)p_s.$$

where $p_s = \mathsf{P}\{\widetilde{\omega} = \omega_s\}$, the probability of scenario s.

The structure of the SLP problem is well suited for Benders' decomposition which, in the SLP literature, goes under the name of the *L-shaped method* ([14]). At each iteration of the L-shaped method one constructs a supporting hyperplane of the recourse function $\mathsf{E}[h(x,\widetilde{\omega})]$. This hyperplane is then is added to the collection of previously generated hyperplanes, and the method proceeds in this manner until a stopping rule is satisfied. The sequence of hyperplanes are called 'cuts' and they provide a piecewise linear lower bounding envelope of $\mathsf{E}[h(x,\widetilde{\omega})]$. Some variants of this method develop cutting plane approximations of the value functions $\{h(x,\omega_s)\}_{s=1}^S$ so that S cuts are generated at a time (see [2,13]). These types of cutting plane algorithms are often called *multicut methods* since each iteration requires the development of as many cuts as there are scenarios in S. Since these multicut methods use sums of piecewise linear approximations, rather than one aggregated approximation (as in the L-shaped method), they provide better lower bounds than the L-shaped method, although there is no similar guarantee regarding the quality of the upper bounds derived.

Two Basic Tools: Incumbent Solutions and Regularization

Consider optimization problems of the form $\min\{f(x):x \in X\}$, where $f:\mathbf{R}^n \to \mathbf{R}$ is convex and $X \subseteq \mathbf{R}^n$ is a convex set. At each iteration, we assume that a point x^k is given, and we generate a lower bounding linearization $\alpha_k + \beta_k^\top x$, where the cut coefficients $\beta_k \in \partial f(x^k)$ and $\alpha_k = f(x^k) - \beta_k^\top x^k$. (Since we are dealing with convex functions, the Clarke subdifferential, [3], will suffice for the purposes of this exposition.) The collection of all previously generated cuts define a piecewise linear lower bounding function, denoted $f_k(x)$, and is represented as

$$f_k(x) = \max_{t \in J_k} \alpha_t + \beta_t^\top x, \tag{1}$$

where J_k is an index set, $\alpha_t \in \mathbf{R}$, $\beta_t \in \mathbf{R}^n$ were generated in iteration t.

For the case of SLP, the cut coefficients are obtained by recognizing that as long as $h(x^t,\omega)$ is finite, LP duality implies that

$$h(x,\omega) = \begin{cases} \max & [r(\omega) - T(\omega)x^t]^\top \pi \\ \text{s.t.} & W^\top \pi \leq g. \end{cases}$$

Letting $\pi^t(\omega)$ denote an optimal dual solution, a subgradient of h is

$$-T(\omega)^\top \pi^t(\omega) \in \partial h(x^t,\omega).$$

Hence the cut coefficients for the SLP problem are defined as

$$\begin{aligned} \beta_t &= c - \mathsf{E}[T(\widetilde{\omega})^\top \pi^t(\widetilde{\omega})], \\ \alpha_t &= \mathsf{E}[r(\widetilde{\omega})^\top \pi^t(\widetilde{\omega})]. \end{aligned} \tag{2}$$

A more detailed explanation of this cut is provided in ► Stochastic linear programming: Decomposition and cutting planes. It is worth noting at this point that one of the major handicaps associated with deterministic cutting plane algorithms for SLP arises from the calculations necessary in (2), which involves multidimensional integration. For problems involving a large number of outcomes, this operation becomes computationally burdensome, and one must then resort to sampling

based methods such as those discussed in the next section.

The iterates obtained by cutting plane methods may be generated as

$$x^{k+1} \in \operatorname{argmin}\{f_k(x):\ x \in X\}. \tag{3}$$

We will designate these points as 'candidate' solutions. Since f_k has the form shown in (1), it can be written as a linear programming problem. In the language of Benders' decomposition ([1]), this problem is called a *master problem* (program).

There are several criticisms that may be leveled against cutting plane methods. For example, in the earliest versions of these methods (including [11]) the index set J_k was defined as $\{1, \ldots, k\}$; consequently, the size of the LP and its data structures would grow indefinitely. In order to curtail the growth of the LP size, one may resort to dropping some of the inequalities. As one might expect, it is important to derive mathematically justifiable cut dropping rules, for otherwise, the sequence generated by (3) may not converge to an optimal solution. In this article we will summarize *stabilization* schemes that guarantee convergence, while ensuring that there is a fixed upper limit on the size of the master program. Another criticism is that the sequence $\{f(x^k)\}$ is not a decreasing sequence. As a result, it is advisable to monitor decreases in $f(x^k)$ and record points that suggest significant improvements. Such iterates will be designated as 'incumbent' solutions, so that the sequence of incumbent solutions is a subsequence of the sequence of candidate solutions.

We now proceed to a discussion of regularization, in which the piecewise linear approximation in (1) is augmented with a strictly convex casting function [15] or an auxiliary function [4]. The most commonly used casting function is a quadratic proximal term (see [12,13]). In iteration k, let $\overline{x}^k$ denote the incumbent solution. The regularized master program is defined as

$$x^{k+1} \in \operatorname{argmin}\left\{ f_k(x) + \frac{\sigma}{2}\left\|x - \overline{x}^k\right\|^2 :\ x \in X \right\},$$

where $\sigma > 0$ is a parameter that may be chosen during the course of the algorithm. Note that if we define

$$\rho_k(x;\sigma) = f_k(x) + \frac{\sigma}{2}\left\|x - \overline{x}^k\right\|^2$$

then

$$\partial\rho_k(\overline{x}^k;\sigma) = \partial f_k(\overline{x}^k).$$

Hence, if 0 belongs to the set $\partial\rho_k(\overline{x}^k;\sigma)$, then $0 \in \partial f_k(\overline{x}^k)$. Moreover, if $\partial f_k(\overline{x}^k) \subseteq \partial f(\overline{x}^k)$, then $0 \in \partial\rho_k(\overline{x}^k;\sigma)$ implies $0 \in \partial f(\overline{x}^k)$, the first order optimality condition at $\overline{x}^k$.

Regularization of Deterministic Cutting Plane Methods

In the following we shall work with the understanding that X is a convex polyhedral set represented in the form $\{x\colon Ax \le b\}$. For the remainder of the development, it will be convenient to rewrite the approximation in terms of the displacement, d, from the incumbent solution, $\overline{x}^k$. Let $x \leftarrow \overline{x}^k + d$. In order to write the cuts as a function of d, we define $\overline{f}_t^k = \alpha_t + \beta_t^\top \overline{x}^k$ that is, $\overline{f}_t^k$ denotes the value of the cut $\alpha_t + \beta_t^\top x$ at the incumbent point $\overline{x}^k$. Hence the approximation in terms of the direction vector d may be written as

$$v_k(d) = f_k(\overline{x}^k + d) = \max_{t \in J_k}\{\overline{f}_t^k + \beta_t^\top d\}. \tag{4}$$

The *primal master problem* (direction finding problem) can now be written as

$$(\mathrm{PRM}^k) \quad v_k = \begin{cases} \min & v + \frac{\sigma}{2}\|d\|^2 \\ \text{s.t.} & v - \beta_t^\top d \ge \overline{f}_t^k, \quad \forall t \in J_k, \\ & \overline{x}^k + d \in X. \end{cases}$$

Since this section is devoted to a deterministic algorithm, we work with the assumption that

$$v_k(0) = f_k(\overline{x}^k) = f(\overline{x}^k). \tag{5}$$

We now state a prototypical regularized deterministic cutting plane algorithm.

This algorithm has several important properties. First note from (4) and (5), and the definition of d^k (a solution to PMK^k) that

$$\begin{aligned} v_k(d^k) &= f_k(\overline{x}^k + d^k) \\ &\le f_k(\overline{x}^k) - \frac{\sigma}{2}\left\|d^k\right\|^2 = f(\overline{x}^k) - \frac{\sigma}{2}\left\|d^k\right\|^2. \end{aligned} \tag{6}$$

Hence $\Delta_k \ge 0$, with strict inequality if $d^k \ne 0$. Consequently, the sequence of incumbents defined in Step 2 yields a descending sequence of objective values. Furthermore, the stopping rule is based on the observation that if $\Delta_{k+1} \le \epsilon$, then $\sigma\,\|d\|^2/2 \le \epsilon$. Thus with an appropriate choice of ϵ, the stopping rule tests whether the direction d^k has a sufficiently small norm.

0	(Initialize)
0a	(Problem parameters) A convex function f:$\mathbf{R}^n \to \mathbf{R}$ and a convex polyhedral set $X = \{x: Ax \leq b\}$ are given. For SLP problems, $f(x) = c^\top x + \mathbf{E}[h(x,\tilde{\omega})]$.
0b	(Algorithm parameters) $\sigma > 0$, $\gamma \in (0,1)$ and $\epsilon > 0$ are given.
0c	$x^1 \in X$ is given, $k \leftarrow 0$, $J_0 \leftarrow \emptyset$, $\bar{x}^0 \leftarrow x^1$, $\Delta k = \infty$.
1	(Define/Update the piecewise linear approximation) $k \leftarrow k + 1$. Evaluate $\beta_k \in \partial f(x^k)$ and $\alpha_k = f(x^k) - \beta_k^\top x^k$. (For SLP these quantities are calculated as in (2).) $J_k = \{t \in J_{k-1} : \theta_t^{k-1} > 0\} \cup \{k\}$.
2	(Update the incumbent) IF $f(x^k) \leq f(\bar{x}^{k-1}) - \gamma\Delta_k$. THEN $\bar{x}^k = x^k$. ELSE $\bar{x}^k = \bar{x}^{k-1}$.
3	(Solve the regularized master) Let $x^{k+1} = \bar{x}^k + d^k$, where d^k denotes an optimal solution to (PRM^k). Similarly let θ_t^k denote the optimal Lagrange multipliers associated with the cuts indexed by $t \in J_k$.
4	(Stopping rule) Let $\Delta_{k+1} = f(\bar{x}^k) - v_k(d^k)$. IF $\Delta_{k+1} \leq \varepsilon$, THEN stop ELSE repeat from Step 1.

A regularized deterministic cutting plane algorithm

In Step 4, Δ_{k+1} is defined using the objective function approximation, $v_k(d^k)$. Thus, the decision to update the incumbent is based on how well Δ_{k+1} predicts the actual change in the objective value. For higher values of γ an incumbent change is accepted only when the prediction is accurate, whereas smaller values of γ yield a less stringent criterion.

The analysis of this algorithm relies heavily on a dual statement of the regularized master (PRM^k). In fact, [12] states the algorithm in terms of the dual to the master problem specified above. To write the dual, let F_k denote the vector of scalars $\{\overline{f}_t^k\}_{t\in J_k}$, and similarly, let B_k denote a matrix whose rows are given by the cut coefficients $\{\beta_t^\top\}_{t\in J_k}$. Furthermore, let b_k denote the vector $A\bar{x}^k - b$. Then the dual to (PRM^k) is

$$(\text{DRM}^k) \quad \begin{cases} \max & F_k^\top\theta + b_k^\top\lambda - \frac{1}{2\sigma}\left\|B_k^\top\theta + A^\top\lambda\right\|^2 \\ \text{s.t.} & \mathbf{1}^\top\theta = 1, \quad \theta \geq 0, \quad \lambda \geq 0. \end{cases}$$

One of the important relationships between the primal and dual optimal solutions for the pair (PRM^k) and (DRM^k) is

$$d = -\frac{1}{\sigma}(B_k^\top\theta + A^\top\lambda). \tag{7}$$

Finally, we note that the total number of cutting planes retained in Step 1 cannot exceed $n + 2$. To see this, note that since $\sigma > 0$, the primal and dual master problems are strictly convex programs, and consequently both have unique optimal solutions. Suppose now that d^k and (θ^k, λ^k) are the optimal solutions for the primal-dual pair. Clearly these solutions must satisfy (7) which in combination with the convexity constraint in (DRM^k), yields $n + 1$ linear equations involving the dual variables (θ, λ). Note that (θ^k, λ^k) (the optimal dual solution) must be an extreme point of the resulting polyhedron; for if not, then (θ^k, λ^k) can be written as a convex combination of two other points with the same dual objective value. The latter, of course, contradicts the uniqueness of the dual optimum. Hence (θ^k, λ^k) must be an extreme point of the polyhedron determined by (7) and the dual feasibility restrictions. Since there are at most $n + 1$ equations, it follows that there can be at most $n + 1$ components of θ^k which can be positive. Thus by including the new cut, we conclude that the cardinality of J_k cannot exceed $n + 2$.

We should mention that similar benefits can also be realized in multicut methods. For instance, if we use the sum of q piecewise linear approximations as in above, then one needs to carry at most $n + q$ cuts from one iteration to the next. By accounting for the q cuts generated in each iteration of a multicut method, it follows that the size of the master problem can be restricted to at most $n + 2q$. In applying this class of methods to SLP problems, the regularized decomposition algorithm [13] uses $q = S$, the total number of outcomes. Hence the resulting master problem requires $n + 2S$ cuts.

A Regularized Stochastic Decomposition Algorithm

In contrast to the previous sections, this one is dedicated to SLP problems. For this class of problems, the calculation of multidimensional integrals in (2) creates a computational bottleneck for deterministic algorithms. In order to overcome this bottleneck, the

Stochastic Decomposition (SD) algorithm combines sampling within a cutting plane algorithm. In this section we discuss a regularized version of stochastic decomposition (SD).

It is clear that the primary change in going from the deterministic method of the previous section to the stochastic method is the inclusion of sampling. Referring to the algorithmic statement of the previous section, we augment the statement of step 1 with the inclusion of a sampled outcome (generated independently of previous observations). Further differences between SD and other cutting plane algorithms arise from the manner in which the approximations are generated/updated (Step 1) as well as the rule for accepting an incumbent (Step 2). The process for creating the cutting plane coefficients is presented in ► Stochastic linear programming: Decomposition and cutting planes, we do not reproduce those details here; instead, we summarize the changes to step 1 as follows:

1 | (Define/Update the piecewise linear approximation)
$k \leftarrow k + 1$. Generate ω^k, an outcome of $\tilde{\omega}$. Evaluate cut coefficients (α_k^k, β_k^k) for a new cut, and update coefficients of previously generated cuts. Denote the updated cuts by (α_t^k, β_t^k). Update J_k.

Next we motivate the reason for updating J_k in a manner that is slightly different that that used in a deterministic algorithm. We also argue the need to update the incumbent (in Step 2) with a slightly different, though analogous rule.

Unlike the regularized deterministic algorithm which uses at most $n + 2$ cuts in the master problem, convergence results for the regularized SD method requires $n + 3$ cuts in the master program. This is because of the inherent inaccuracy of the objective function estimates used in a sampling algorithm. Recall that Step 2 of the deterministic method uses objective function values $f(x^k)$ and $f(\bar{x}^{k-1})$ to decide whether the incumbent needs to be updated. In a sample based algorithm these quantities cannot be calculated, and the choice of an incumbent is necessarily based on sampled information.

In order to prove asymptotic results regarding the sequence of incumbent solutions, one needs to have asymptotic accuracy of a subgradient and the function value at an accumulation point of the incumbent sequence. One way to accomplish this is to require that the cut associated with an incumbent becomes asymptotically accurate (with probability one). This property may be attained algorithmically by periodically updating the cut associated with the incumbent solution. To clarify the process, suppose that at iteration k, we know that the most recent iteration at which the incumbent cut was generated (or updated) was $\bar{i}_k < k$. Furthermore, let us suppose that we intend to update the incumbent cut every τ iterations ($\infty > \tau \geq 1$). Then, at an iteration in which $k = \bar{i}_k + \tau$, the incumbent cut is updated to reflect the impact of outcomes as well as the dual vertices that may have been generated since iteration $\bar{i}_k$. These updates then guarantee that the sample size used for the incumbent cut grows indefinitely as required by the law of large numbers.

As suggested by the above discussion, the rule for cut retention ought to maintain a cut associated with the incumbent solution. At iteration k, let t_k denote the index associated with a cut generated at the incumbent solution. Then the rule used for cut retention is the following:

$$J_k = \left\{t \in J_{k-1}\colon\ \theta_t^{k-1} > 0\right\} \cup \{k, t_k\}.$$

Hence the maximum number of cuts used in the regularized master for SD is $n + 3$. With these changes included in the definition of the approximation f_k, we now provide the rule used for updating incumbents in Step 2.

2 | (Update the incumbent)
IF $k = 1$, THEN put $\Delta_k = \infty$,
ELSE $\Delta_k = f_{k-1}(\bar{x}^{k-1}) - v_{k-1}(d^{k-1})$.
IF $f_k(x^k) \leq f_k(\bar{x}^{k-1}) - \gamma\Delta_k$,
THEN $\bar{x}^k = x^k$, ELSE $\bar{x}^k = \bar{x}^{k-1}$.

Finally, there is one additional issue that arises whenever one uses sampled information. In some cases, as in SD, sampling is incorporated within the decomposition algorithm. In other cases, sampling is undertaken prior to using the optimization algorithm. Nevertheless, since each case uses sampled data, one should explore the possibility of error. We refer the reader to two articles in this area. The first of these ([5]) is based

on designing stopping rules that work with 'in-sample' information, and is tailored for the SD algorithm. Another approach, using 'out-of-sample' scenarios is provided in [8].

Conclusions

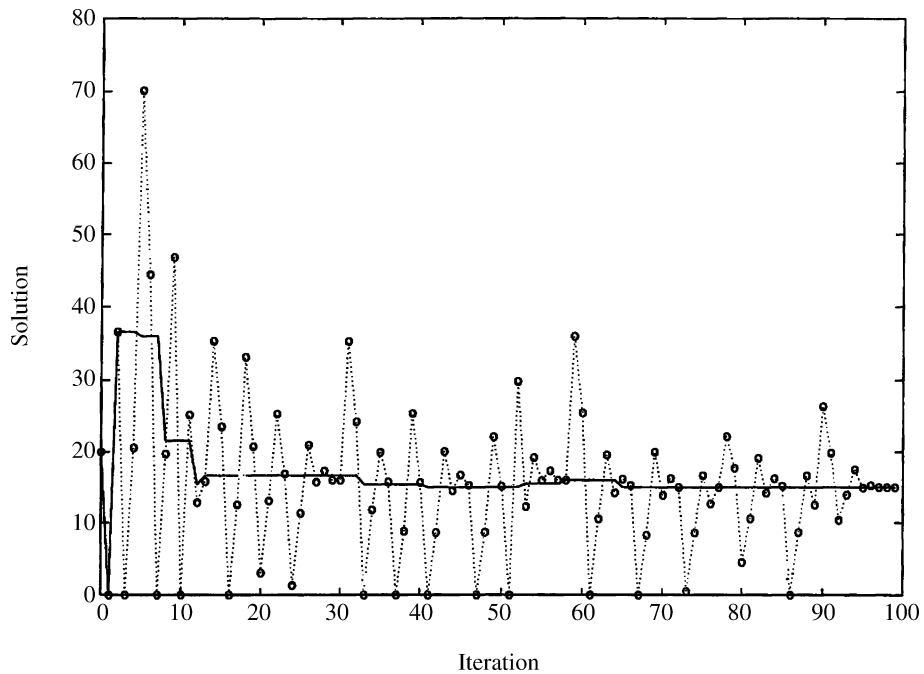
We conclude this article with two examples taken from [9]. The illustrative application for these examples deals with an electricity reserve planning problem that finds an optimal trade-off between the cost of reserve capacity and the cost of unmet demand [9]. In Fig. 1, we illustrate two trajectories: the solid line is the trajectory of incumbent solutions and the dotted line is the set of candidate solutions. In order to isolate the impact of using an incumbent solution from the impact of the quadratic term, we used only the LP master in generating Fig. 1. It is clear that the solid line of incumbent solutions provides a far more stable trajectory.

Next we discuss the impact of the quadratic term with reference to the same reserve planning example. In order to isolate the impact of the quadratic term, we examine the candidate sequences from two implementations: one in which the weight on the quadratic term is very small ($\sigma = 10^{-8}$) and another in which $\sigma = 1$. The candidate solutions from the former implementation are depicted by dotted lines in Fig. 2, whereas, the candidates from the latter implementation are shown by the solid line. Once again, the solid line, representing the impact of regularization, exhibits a more stable trajectory than the trajectory associated with the dotted line.

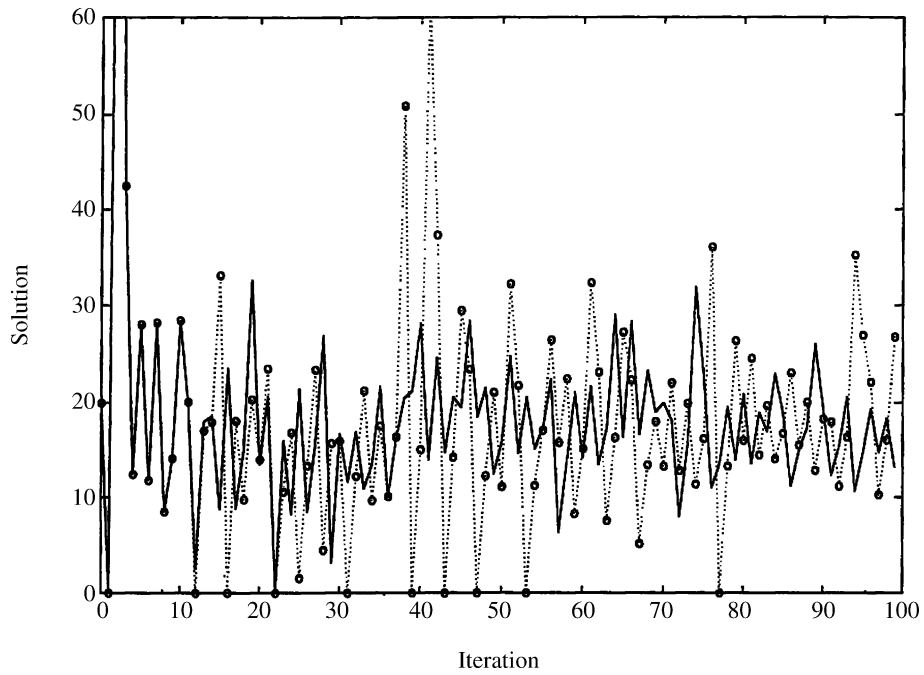
Before we close the article, we should mention one acceleration technique that often helps speed-up regularized algorithms in practice. The idea is to change the magnitude of σ dynamically: σ is reduced when the incumbent changes and $\left\| x^{k+1} - \overline{x}^k \right\|$ increases; on the other hand, σ is increased when the incumbent does not change. This allows the algorithm to take smaller steps (higher σ) when the approximation seems to be poor, whereas, the steps are allowed to more 'daring' (lower σ) when progress is rapid.

Thanks

This work is based on research funded by the National Science Foundation.



Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems, Figure 1
Trajectories of candidate ··· and incumbent __ solutions



Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems, Figure 2
Trajectories of candidate solutions from unregularized ··· and regularized __ algorithms

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Benders JF (1962) Partitioning procedures for solving mixed variables programming problems. Numerische Math, 4:238–252
2. Birge JR, Louveaux F (1988) A multi-cut algorithm for two stage linear programs. Europ J Oper Res 34:384–392
3. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
4. Culioli C, Cohen G (1990) Decomposition/coordination algorithms in stochastic optimization. SIAM J Control Optim, 28:1372–1403
5. Higle JL, Sen S (1991) Statistical verification of optimality conditions. Ann Oper Res, 30:215–240
6. Higle JL, Sen S (1991) Stochastic decomposition: An algorithm for two-stage linear programs with recourse. Math Oper Res, 16:650–669
7. Higle JL, Sen S (1994) Finite master programs in regularized stochastic decomposition. Math Program, 67:143–168
8. Higle JL, Sen S (1996) Duality and statistical tests of optimality for two stage stochastic programs. Math Program, 75:257–275
9. Higle JL, Sen S (1996) Stochastic decomposition: A statistical method for large scale stochastic linear programming. Kluwer, Dordrecht
10. Infanger G (1992) Monte Carlo (importance) sampling within a Benders' decomposition for stochastic linear programs. Ann Oper Res, 39:69–95
11. Kelley JE (1960) The cutting plane method for convex programs. J SIAM, 8:703–712
12. Kiwiel KC (1985) Methods of descent for nondifferentiable optimization. Lecture Notes Math, vol 1133. Springer, Berlin
13. Ruszczyński A (1986) A regularized decomposition method for minimizing a sum of polyhedral functions. Math Program, 35:309–333
14. Slyke R Van, Wets RJ-B (1969) L-Shaped linear programs with application to optimal control and stochastic programming. SIAM J Appl Math, 17:638–663
15. Wets RJ-B (1980) Convergence of convex functions, variational inequalities and convex optimization problems. In: Lions JL, Cottle R, Giannessi F (eds) Variational Inequalities and Complementarity Problems. Wiley, New York pp 375–403

Stable Set Problem: Branch & Cut Algorithms

Steffen Rebennack
Center of Applied Optimization, University of Florida, Gainesville, USA

MSC2000: 90C27, 90C35

Article Outline

Abstract
Keywords
Introduction
Method
- Stable Set Polytope
- Separation
- Branching
- Implementation Aspects

Conclusion
See also
References

Abstract

In this article, we provide an overview on how the maximum weighted stable set problem can be solved exactly with Branch & Cut techniques. In addition, we provide selected references to other exact methods. We start with a brief introduction of the stable set problem and a few basic definitions but assuming that the reader is already familiar with the basic concepts. The main stress of this article lies in the review of polyhedral results for the stable set polytope in Sect. "Stable Set Polytope" and the discussion of separation procedures, Sect. "Separation". An efficient Branch & Cut algorithm needs, in addition to strong separation routines, also a good branching strategy. This is discussed in Sect. "Branching". At the end, some implementation aspects are considered.

Keywords

Stable set; Independent set; Maximum clique; Vertex packing; Branch & Cut; Separation; Exact method

Introduction

Let $G = (V, E)$ be an undirected graph consisting of a nonempty finite set V, the node set; and a finite set E, the edge set, of unordered pairs of distinct elements of V. A stable set of graph G is defined as a set of nodes S with the property that the nodes of S are pairwise non adjacent; two nodes are called adjacent if there is an edge in E connecting them. In the literature, stable set is also called independent set, vertex packing, co-clique or anticlique. If each node v_i of a graph G is assigned a weight c_i, then the graph is called

weighted. In this case, the maximum weighted stable set problem looks for a stable set S which maximizes the sum of the weights corresponding to the nodes in S, $\sum_{v_i \in S} c_i$. In the case when G is not weighted, or all $c_i = 1$, we are interested in a stable set with the maximum number of nodes, which is called maximum cardinality stable set. The size of a maximum cardinality stable set is called the stability number of graph G and is denoted by $\alpha(G)$. Throughout this article, references to the maximum stable set problem, or just stable set problem, consider the weighted case unless otherwise noted.

Following from its definition, the stable set problem has many applications in various fields, [15]. Especially when "conflict(s)" between some objects occur, it is an indicator that the stable set problem is applicable. Next to the Traveling Salesman Problem, it is one of the most important combinatorial optimization problems. It is well known that it is $\mathcal{NP}$-*hard* to determine a maximum stable set in an arbitrary graph, [37]. This holds true for the cardinality case. Furthermore, it is also hard to approximate the stable set number: It can be shown that for any fixed $\varepsilon > 0$ there is no polynomial time algorithm for approximating the stability number within a factor of $|V|^{\varepsilon}$, under the assumption that $\mathcal{P} \neq \mathcal{NP}$, [3,36].

Let us briefly and informally introduce some polyhedral terminologies. In our case it is sufficient to define a polyhedron as the solution set of a system of linear inequalities. If the solution set is bounded, it is called a polytope. Graphically speaking, a polytope in $\mathbb{R}^n$ is of full-dimension if it contains an n-dimensional sphere completely; in 2-dimensions it is therefore forbidden that the polytope is empty, one point or a line segment. A linear inequality $\beta^\top x \leq b_0$ is valid with respect to a polyhedron P, if P is a subset of $\{x \,|\, \beta^\top x \leq b_0\}$. We call a set $F \subseteq P$ a facet of P if there is a valid inequality $\beta^\top x \leq b_0$ for P such that $F = \{x \in P \,|\, \beta^\top x = b_0\}$, and the inequality is not dominated by any other valid inequality. This inequality is called a facet-defining inequality for P. In the case when v is a point in the polyhedron P and $F = \{v\}$, we call v a vertex of polyhedron P. Now, let P be a polytope and x^* be a given point. The task to decide if this point lies in P or if not to find a valid inequality $\beta^\top x \leq b_0$ for P which is violated by x^*, is called the separation problem for polytope P. The convex hull of points $y_1, \ldots, y_n \in \mathbb{R}^d$ is the set of points x satisfying $x = \sum_{i=1}^n \lambda_i y_i$ with $\sum_{i=1}^n \lambda_i = 1$ and $\lambda_i \geq 0 \;\; \forall i$. It is denoted by $conv\{y_1, \ldots, y_n\}$. More details can be found for instance in [20,42,82].

We introduce now, in addition to the ones above, several graph theoretic definitions and notations needed throughout this article. A node v is incident to an edge e, if $e = uv$. The two nodes incident to an edge are its endnodes. A node is isolated if it has no neighbor in the graph, which means that it is not an endnode of any edge of the graph. The neighborhood of a node v is the collection of all its neighbor ans is abbreviated with $\Gamma(v)$. If a graph has no isolated nodes, it is called connected. A graph is said to be complete if it contains an edge connecting each pair of its nodes. A clique is the node set of a complete subgraph. If a clique has three nodes it is also called a triangle. The cardinality of a graph G is abbreviated by $|G|$ and denotes the number of nodes in the graph. The complement graph $\overline{G}$ of the graph G has the same node set as G and contains an edge between two nodes, iff no edge is contained in G. We call a graph G bipartite if its node set V can be partitioned into two disjoint sets V_1, V_2 with $V = V_1 \cup V_2$ such that neither two nodes of set V_1 nor two nodes of set V_2 are neighbors. We call $H = (W, F)$ a subgraph of G, and write $H \subseteq G$, when $W \subseteq V$ and $F \subseteq E$ is the set of edges of graph G with both endnodes in W. Two graphs $G = (V, E)$ and $H = (W, F)$ are called isomorphic, if there is a bijection $\phi \colon V \to W$ such that $uv \in E \Leftrightarrow \phi(u)\phi(v) \in F$. A matching is a collection of pairwise disjoint edges. If in a matching M every node of G is incident with exactly one edge M, then it is a perfect matching. More about graph theory can be found in [33,78].

We do not describe the Branch & Cut algorithm in general here, as we assume the reader is familiar with its basic ideas. For more information we refer to [40, 46,61,79].

Before we discuss some aspects of a Branch & Cut algorithm to solve the maximum stable set problem, we provide a list of some other exact solution methods. Clearly, this list does not claim to be complete. A more detailed list of exact methods can be found in [15,22]. In this context, we want to point out that the stable set problem is equivalent to the maximum clique problem in the complement graph. Hence, each method solving the maximum clique problem can also

be used to solve the stable set problem. For polynomial time algorithms for some special classes of graphs, see [2,6,8,14,17,32,48,54,55,59,63] and Sect. "Stable Set Polytope". Algorithms finding all maximum stable sets in a graph are considered in [19,29,44,49,73]. In the literature, many variants of Branch & Bound algorithms have been discussed, [4,12,35,51,60,65,71,76,80]. Other methods using, for instance, continuous formulations, column generation or constraint programming can be found in [5,16,18,21,25,53,64,68,75,77].

Benchmark instances are provided by the second DIMACS Challenge, [34], from 1992/1993 and by the BHOS library from 2000, [13]. Note that some of these stable set instances are still unsolved. A test case generator was introduced by Hasselberg et al. [45].

Method

Let us now formulate the maximum stable set problem as a linear program. Therefore, one choice could be to introduce variables x_i for each node $v_i \in V$, which have value one, if node v_i is in a stable set, say S, and otherwise zero. Such a vector is called an incidence vector. Obviously, for each edge, only one endnode can be in a stable set and hence, we get the so called edge-inequalities

$$x_i + x_j \leq 1 \quad \forall\, ij \in E. \tag{1}$$

It is easy to see, that if vector x has a positive integer domain (or more precisely, binary domain), each vector satisfying inequalities (1) induces a stable set and vice versa. Hence, if c denotes the (positive) weight vector of the nodes, one gets the following integer program

$$\begin{aligned} \max_x \quad & c^\top x \\ s.t. \quad & x_i + x_j \leq 1 \qquad \forall\, ij \in E \\ & x \in \{0,1\}^{|V|}, \end{aligned} \tag{2}$$

which solves the maximum weighted stable set problem. We recognize, that this formulation has only $|E|$ constraints and $|V|$ variables. So the formulation is quite compact. Unfortunately, the binary constraints on x make it hard to solve this linear program. We will discuss some relaxations of this problem in the next section which is mainly based on [9,39,42,58,70].

Stable Set Polytope

The stable set polytope of graph $G = (V, E)$ is defined as the convex hull of the incidence vectors of all stable sets in G. It is denoted by

$$P_{\mathrm{STAB}}(G) := conv\{\, \chi^S \mid S \subseteq V \text{ stable set}\},$$

where χ^S is the incidence vector of set S. From the integer program formulation (2) we see that $P_{\mathrm{STAB}}(G)$ is a polyhedron. As it is bounded by the $|V|$-dimensional unit cube, it is indeed a polytope. The definition of a stable set implies that the unit vectors are always stable sets. Trivially, the zero vector is a stable set, the empty set, therefore, the stable set polytope is full-dimensional. This implies that all facets of $P_{\mathrm{STAB}}(G)$ are inequalities, and hence, we do not have to consider equalities, [58,67].

Let us now discuss some relaxations of the integer program formulation (2) which will also give us relaxations of the stable set polytope. The obvious idea is to relax the binary condition on x, and instead make them continuous which leads to

$$0 \leq x_i \leq 1 \quad \forall v_i \in V. \tag{3}$$

The integer problem reduces to a linear program which can be solved in polynomial time. This relaxation leads to the so called stable set polytope relaxation

$$\begin{aligned} &P_{\mathrm{RSTAB}}(G) \\ &:= \{x \in \mathbb{R}^n \mid x_i + x_j \leq 1,\ 0 \leq x \leq 1\ \forall ij \in E\}. \end{aligned} \tag{4}$$

From its construction, we get that $P_{\mathrm{STAB}}(G) \subseteq P_{\mathrm{RSTAB}}(G)$. For a complete graph with cardinality ≥ 3, the x vector with value 1/2 in each component is a vertex of $P_{\mathrm{RSTAB}}(G)$, but it cannot be contained in $P_{\mathrm{STAB}}(G)$ as a maximum cardinality stable set in a complete graph has cardinality one. This example shows that the relaxation above is very weak. Note, the vector whose entries are all 1/2 is always contained in $P_{\mathrm{RSTAB}}(G)$ – independent of the structure of the graph G. The following corollary generalizes this observation. It was first indicated by Balinski [11].

Corollary 1 *The vertices of $P_{RSTAB}(G)$ are $(0, \frac{1}{2}, 1)$-valued.*

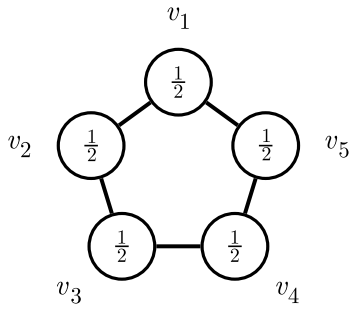
We saw that for a complete graph G with cardinality ≥ 3, $P_{\mathrm{STAB}}(G) \subset P_{\mathrm{RSTAB}}(G)$. The next theorem states

that this holds except for connected bipartite graphs. In this case the stable set polytope and the stable set polytope relaxation are equal, that is $P_{\text{STAB}}(G) = P_{\text{RSTAB}}(G)$.

Theorem 2 [42] *The non-negativity inequalities, $x_i \geq 0 \ \forall v_i \in V$, together with the edge inequalities (1) are sufficient to describe $P_{STAB}(G)$, iff G is bipartite and has no isolated nodes.*

Theorem 2 has the following important implication: It states that the maximum stable set problem for bipartite graphs can be solved in polynomial time by solving the stable set problem over (4). As a consequence, a Branch & Cut algorithm using the stable set polytope relaxation (4), will terminate for bipartite graphs in the root node of the branching tree after solving one linear program. However, we already have indicated that the stable set polytope relaxation is very week and hence not a good choice in a Branch & Cut framework for general graphs. Obviously, it can be checked in linear time whether a graph is bipartite or not. Exact polynomial time algorithms for bipartite graphs can be found in [42,47].

As the restriction to bipartite graphs is very tough, we want to find some ways to strengthen the stable set polytope relaxation. One idea is to add additional valid inequalities to $P_{\text{RSTAB}}(G)$. Therefore, let us consider the Fig. 1. It shows a graph with the five nodes $v_1, v_2, \ldots, v_5$. Such a graph is called odd-cycle. In general, whenever a (sub-)graph H has an odd number of nodes, say n, and there are n adjacent edges in the edge set such that each node is incident to exactly two nodes, then we call H an odd cycle. Notice that an odd cycle can have more than n edges. In this case, any additional edge is called chord. For the graph of the Fig. 1 , the



Stable Set Problem: Branch & Cut Algorithms, Figure 1
Odd cycle with five nodes

stable set polytope relaxation allows the fractional solution with all entries of 1/2, as illustrated. This solution is optimal, and for the cardinality stable set problem, the objective function value is 5/2, which is greater than any optimal stable set which has cardinality two. Now, summing up all edge inequalities corresponding to the five edges in the graph, one gets

$$2x_1 + 2x_2 + 2x_3 + 2x_4 + 2x_5 \leq 5 \, .$$

In this case each node is incident to exactly two edges, giving the coefficients for the variables, and there are five edge inequalities, providing the right-hand side. This inequality can be divided by two and as all variables are binary, one gets

$$x_1 + x_2 + x_3 + x_4 + x_5 \leq \left\lfloor \frac{5}{2} \right\rfloor \, .$$

This inequality can be generalized to the so called odd-cycle inequalities

$$\sum_{v_i \in \widetilde{V}} x_i \leq \frac{|C| - 1}{2} \qquad (5)$$

for each odd cycle $C = (\widetilde{V}, \widetilde{E}) \subseteq G$.

From the construction above, it is obvious that the odd-cycle inequalities are valid for the stable set polytope. If, in addition, the stable set polytope relaxation satisfies all odd-cycle inequalities of G, then it is called a cycle-constraint stable set polytope and is denoted by

$$P_{\text{CSTAB}}(G) := \left\{ x \in \mathbb{R}^{|V|} \mid x \text{ satisfies (1), (3) and (5)} \right\} .$$

If you consider, again, a complete graph, you see that there is no constant which relates the optimal solution over $P_{\text{CSTAB}}(G)$ to an optimal stable set. However, the graphs for which $P_{\text{CSTAB}}(G) = P_{\text{STAB}}(G)$ are called t-perfect; the "t" stands for "trou" – the French word for "hole". Two examples for t-perfect graphs are bipartite graphs and almost bipartite graphs, where a graph G is called almost bipartite if there is a node v such that graph G without v is bipartite. The problem of checking whether a graph is t-perfect or not belongs to $co\text{-}\mathcal{NP}$. The special structure of t-perfect graphs helps to find a maximum stable set. This is stated by the next corollary.

Corollary 3 *The maximum stable set problem in a t-perfect graph can be solved in polynomial time.*

We will see in Sect. "Separation" that the odd-cycle inequalities can be separated in polynomial time. This proves together with the Equivalence of Optimization and Separation the Corollary 3. Polynomial time algorithms for the class of t-perfect graphs can be found in [42,43].

We are mainly interested in facets of $P_{\mathrm{STAB}}(G)$ since they are not dominated by any valid inequality of $P_{\mathrm{STAB}}(G)$. The odd-cycle inequalities can only be facet-defining if their odd cycles are chordless. If there is a chord, one gets a smaller odd cycle and an even cycle. The smaller odd-cycle inequality together with the edge inequalities dominate the odd-cycle inequality which shows that it cannot be facet-defining. A graph which is a chordless cycle is called a hole. If an odd cycle induces an odd hole, the corresponding odd-cycle inequality is called an odd-hole inequality. Consider the following

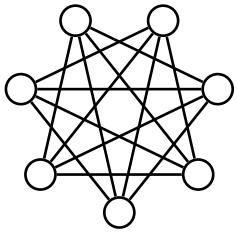
Corollary 4 [57] *Let G be an odd hole. Then $\sum_{v_i \in V} x_i \leq \frac{|V|-1}{2}$ is facet-defining for $P_{STAB}(G)$.*

A counterpart of the odd-cycle inequalities are the antihole inequalities. They are valid for antiholes, which is the complement graph of an odd hole with at least five nodes. From the Fig. 2, we recognize that each stable set of an antihole with n nodes can contain at most two nodes as each node is adjacent to exactly $n-2$ nodes. Therefore, the following inequalities hold

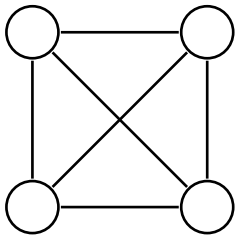
$$\sum_{v_i \in \widetilde{V}} x_i \leq 2 \quad \text{for each antihole } A = (\widetilde{V}, \widetilde{E}) \subseteq G\,. \tag{6}$$

Note that an antihole with 5 nodes is isomorphic to an odd hole with 5 nodes. The separation problem for the antihole inequalities is not known whether it belongs to P or not.

Another group of inequalities for the stable set polytope builds the clique inequalities. From the Fig. 3 we



Stable Set Problem: Branch & Cut Algorithms, Figure 2
Odd antihole



Stable Set Problem: Branch & Cut Algorithms, Figure 3
Complete graph with four nodes a clique

get

$$\sum_{v_i \in Q} x_i \leq 1 \quad \text{for each clique } Q\,. \tag{7}$$

In 1979, Padberg showed the following important

Theorem 5 [62] *Let G be a graph with node set V and $Q \subseteq V$. Inequality (7) is valid for $P_{STAB}(G)$. An inequality $\sum_{v_j \in Q} x_j \leq 1$ is a facet of P_{STAB}, iff Q is a maximal clique in G.*

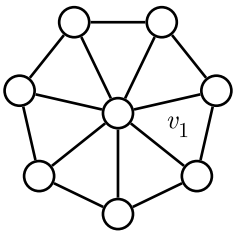
Theorem 5 shows that the edge inequalities (1) are only facet-defining for $P_{\mathrm{STAB}}(G)$, if they build a maximal clique. Hence, they are dominated by the clique inequalities. We will use this observation later in Sect. "Implementation". Note that for triangles, the clique inequality and the odd-cycle inequality are the same. We define the so called clique-constraint stable set polytope as

$$P_{\mathrm{QSTAB}}(G) := \left\{ x \in \mathbb{R}^{|V|} \mid x \text{ satisfies (1), (3) and (7)} \right\}.$$

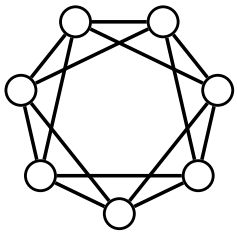
A graph G is called perfect if $P_{\mathrm{QSTAB}}(G) = P_{\mathrm{STAB}}(G)$. Originally, in 1961 Berge called a graph perfect if the coloring number is equal to the clique number. This definition is equivalent to the polyhedral one given above. The maximum clique and the stable set problem are very closely related. Therefore, it is not surprising that it is $\mathcal{NP}$*-hard* to separate the clique inequalities in an arbitrary graph. With this fact, it is quite remarkable that the following theorem holds.

Theorem 6 [42] *The maximum stable set problem for perfect graphs can be solved in polynomial time.*

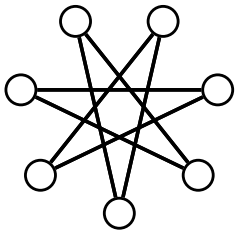
We do not go into the details of the proof here, but nevertheless, we give a rough explanation. It is possible to generalize the clique inequalities to a class of so called orthonormal representation constraints which



Stable Set Problem: Branch & Cut Algorithms, Figure 4
Odd wheel



Stable Set Problem: Branch & Cut Algorithms, Figure 5
(7,3)-web



Stable Set Problem: Branch & Cut Algorithms, Figure 6
(7,3)-antiweb

are polynomially separable. The convex set of all vectors satisfying them and the non-negativity inequalities build the so called theta body, [50,81]. In the case of perfect graphs, this theta body is a polytope which equals $P_{\mathrm{STAB}}(G)$. This implies Theorem 6. However, it is even $\mathcal{NP}$- *hard* to determine an optimal solution over $P_{\mathrm{QSTAB}}(G)$, in general. More about perfect graphs can be found, for instance, in [30,31,66].

If we consider the Fig. 4, we recognize an odd hole with cardinality seven with the additional node v_1 which is adjacent to all other nodes. Such a graph is called wheel and node v_1 is its hub. We see that if node v_1 is contained in a stable set, no other node of the wheel can be contained in it. Hence, we get the following odd-wheel inequalities

$$\sum_{v_i \in \widetilde{V}} x_i + \tfrac{|C|-1}{2} x_u \leq \tfrac{|C|-1}{2} \tag{8}$$

for each odd wheel $C = (\widetilde{V}, \widetilde{E}) \subseteq G$ with hub u .

From its construction, inequality (8) is valid for $P_{\mathrm{STAB}}(G)$. It defines a facet if G is isomorphic to an odd wheel. Recognize that the wheel inequality dominates the odd-hole inequality. Generalizations of the wheel inequalities can be found in [27].

Another class of inequalities are the web and antiweb inequalities. Let p and q be integers satisfying $p > 2q + 1$ and $q > 1$. A graph G is called a web if G is isomorphic to the graph consisting of the nodes $\{v_1, \ldots, v_p\}$ with an edge $v_i v_j$, iff $|i - j| \equiv r < q$ modulo $(n - 2)$. A web is abbreviated with $W(p, q)$. A graph is called antiweb, denoted by $AW(p, q)$, iff $AW(p, q)$ is isomorphic to $\overline{W}(p, q)$. Examples can be seen in the Fig. 5 and 6, respectively. The following inequalities

$$\sum_{v_i \in W(p,q)} x_i \leq q\,, \tag{9}$$

$$\sum_{v_i \in AW(p,q)} x_i \leq \left\lfloor \tfrac{p}{q} \right\rfloor \tag{10}$$

are called web inequalities and antiweb inequalities, respectively. Both types of inequalities are valid for $P_{\mathrm{STAB}}(G)$. The web inequalities (9) define facets if p and q are relatively prime and $G = W(p, q)$ – two natural numbers are called relatively prime if their greatest common divisor is 1, or in formula: $\gcd(p, q) = 1$; while the antiweb inequalities (10) are facet-defining for $P_{\mathrm{STAB}}(AW(p, q))$ if there is no $k \in \mathbb{N}$ with $p = k \cdot q$. More details can be found, for instance, in [28,74].

Now, consider the following class of inequalities for a graph $G = (V, E)$ and $W \subseteq V$

$$x(W) := \sum_{v_i \in W} x_i \leq \alpha(G[W]). \tag{11}$$

They are called rank inequalities. From their construction, inequalities (11) are valid for $P_{\mathrm{STAB}}(G)$. The edge, odd-cycle, clique, antihole, web and antiweb inequalities belong to this class. Therefore, these inequalities are not facet-defining for $P_{\mathrm{STAB}}(G)$ in general. For instance, an odd-wheel with 5 or more nodes does not lead to a rank inequality. Let us now have a closer look at the separation of the discussed inequalities.

Separation

In order to separate the odd-cycle inequalities (5) for a graph G and a vector x^*, one has to find an odd cycle for which x^* violates the corresponding inequality, or one has to prove that such cycles do not exist. In other words, we have to find a minimum-weight odd cycle in a graph, with an appropriate weighting function. If this cycle satisfies the corresponding inequality (5), it is proven that all odd-cycle inequalities are satisfied. Otherwise, one has found a maximal violated odd-cycle inequality. Therefore, we first recognize

Proposition 7 *A minimum-weight odd cycle in a graph G with edge weights can be computed in $O(|V| \cdot |E| \cdot \log(V))$.*

The idea is to construct an auxiliary bipartite graph H. This node set of H consists of two copies of the node set of the original graph G, and there is an edge between two nodes of H if the corresponding original nodes in G are adjacent. The edge weights are copied with the edges. From the construction of H, a minimum odd-cycle with respect to the edge weighting in G corresponds to a shortest path in H and vice versa. Hence, calculation of a shortest path for each node of the original graph G to its copy, gives a minimum weight odd cycle in G. Computing the shortest paths with Dijkstra's algorithm yields to the running time of Proposition 7. Now, define the following edge weighting of graph G depending on x^* as

$$c(v_i v_j) := \frac{1 - x_i^* - x_j^*}{2} . \tag{12}$$

With this weighting, it can be shown that an odd-cycle inequality in G is violated by vector x^*, if and only if

$$\exists C = (\widetilde{V}, \widetilde{E}) \subseteq G \text{ with } C \text{ odd cycle and } \sum_{v_i v_j \in \widetilde{E}} c(v_i v_j) < \frac{1}{2} .$$

This yields directly to the following theorem.

Theorem 8 *The separation problem for the class of odd-cycle inequalities can be solved in $O(|V| \cdot |E| \cdot \log(|V|))$.*

One has to mention that x^* has to satisfy all inequalities (3) before the odd-cycle inequalities can be separated with the above procedure. Otherwise, the weights defined in (12) can become negative, and the shortest path cannot be calculated with Dijkstra's algorithm anymore. More details and the description of the algorithms can be found in [27,38,67]. It is interesting to realize that, in general, there are exponentially many odd cycles in a graph, but on the contrary, the separation of them is polynomial. We want to mention that Grötschel and Pulleyblank introduced in 1981 another method to separate the odd-cycle inequalities with the use of perfect matchings resulting in a running time of $O(|E|^4)$, [41].

Finding a maximum clique is $\mathcal{NP}$-*hard*, while computing an arbitrary maximal clique as well as an arbitrary maximal stable set can be done in linear time. We want to point out that we distinguish between maximum and maximal. Maximal means that there is no better solution containing the particular one; so maximal can be seen as locally best while maximum is global. The separation problem for the clique inequalities asks to find a violated clique inequality in a particular graph G with a given linear program solution or to state that all clique inequalities are satisfied. This is equivalent to finding a maximum clique in G with the linear program solution as node-weighting c. Hence, the separation problem for the clique inequalities is $\mathcal{NP}$-*hard*. Computational tests show that the clique inequalities are very important for polyhedral approaches to the stable set problem, cf. [69]. As an exact separation cannot be considered, one idea could be to fix the size of the cliques to be separated, as then the problem becomes polynomially solvable. Another observation is that it is enough to consider maximal cliques. The reason is that the resulting clique inequality dominates all clique inequalities corresponding to contained cliques. Practically, it is more successful to separate over larger classes of inequalities containing the clique inequalities. We discuss that later for the case of rank inequalities. However, the best computational results, so far, are achieved with heuristic separation methods.

A separation algorithm for the wheel inequalities is given by Cheng and Cunningham, [26]. The appealing idea is to treat each node of the graph as a possible hub of an odd wheel and define appropriate weights for all nodes which are adjacent to this hub. Then, on the new graph, the odd-cycle inequalities can be separated. Hence, this algorithm results in a total running time of $O(|V|^2 \cdot |E| \cdot \log(|V|))$ or $O(|V|^4)$, dependent on the

shortest path algorithm. For practical Branch & Cut algorithms, this complexity is already too high compared to the number of violated inequalities one can expect. Even more sophisticated are the separation routines for the web and antiweb inequalities. They are discussed by Cheng and Vries, [28]. Although they can also be separated in polynomial time, the complexity of the separation algorithms are again too large for practical use. In addition, these type of inequalities usually occur in graphs with high density, e. g. greater than 0.7, which makes its separation doubtful for graphs with low density; where the density of graph $G = (V, E)$ is defined as $\frac{2\cdot|E|}{|V|\cdot(|V|-1)}$.

Next to the special type of inequalities for the stable set polytope discussed in the Sect. "Stable Set Polytope", one can use general classes of inequalities. In this context, we want to mention two of such a type. The first are the so called mod-k cuts which belong to the Chvátal-Gomory cuts with rank one. The appealing idea of the mod-k cuts is to find a multiplier μ, such that a particular inequality system $Ax \leq b$ multiplied with this vector μ can be strengthened by dividing it by a positive integer k. Therefore, let $k > 1$ be integral, and suppose that we have given a system of linear inequalities $Ax \leq b$ with integral coefficients. Let μ be a vector with positive integer entries and appropriate dimension such that

$$\mu^\top A \equiv \quad 0 \pmod{k},$$
$$\mu^\top b \equiv \quad k-1 \pmod{k}.$$

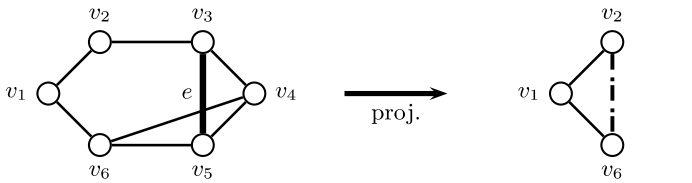
From this, one can obtain the mod-k inequalities

$$\frac{1}{k}\mu^\top Ax \leq \frac{1}{k}(\mu^\top b - (k-1)).$$

Examples for the case of $k = 2$ are the odd-cycle inequalities or the wheel inequalities. The separation of the mod-k inequalities and more details can be found in [23,67].

The second class of general inequalities that we discuss here in brief are the so called local cuts. The principal idea was introduced by Applegate et al. in 2001 for the Traveling Salesman Problem, [1]. Suppose we have given the set of all m feasible solutions to the stable set problem, for instance of a subgraph with n nodes. The idea is to check if a given (solution) vector lies in its convex hull or if it lies outside, to compute a facet which separates this point from the convex hull. This can be achieved by solving a linear program with m variables and n constraints. Its optimal objective function value provides the information if the point lies inside the polytope, and the optimal dual variables give the coefficients of the separation inequality, called local cut. Obviously, this method has some weaknesses. One first has to find a 'good' subgraph and then enumerate all stable sets. In addition, the linear program to be solved can be large, as the number of stable sets in a graph can be exponential. Nevertheless, the resulting cuts can be quite strong, especially if all other separation procedures do not bring new cuts. More details and computational results can be found in [67].

At the end of this subsection, we introduce the idea of separating rank inequalities. We do not go into full detail here but instead focus on the discussion of the principle ideas of the beautiful results of Mannino and Sassano, [52]. The appealing idea is to reduce the size of the graph and to make it denser at the same time. In general, any node of the graph can be selected and its two endnodes will be removed from the graph. In addition, new so called false edges are added to the graph and some other nodes may also be removed. Therefore, this procedure is called edge projection. Now, after a few iterations of this procedure, when the graph is small enough, one can separate any type of rank inequality, for instance, the clique inequalities or the odd-cycle inequalities. If a violated inequality has been found, it must be projected to be valid for the polytope of the original graph. This process is called anti-projection. We have to mention that it is possible that a projected inequality is no longer violated by a solution vector, even if it was in the projected graph. The edge-projection and the anti-projection can be done in linear time which makes this method very fast. Let us now consider a small example indicated in the Fig. 7. In Fig. 7a you see a small graph with six nodes. It is an odd hole with the additional node v_4. If we select edge $e = v_3v_5$ to project on, then in this case, nodes v_3, v_5 and, in addition, v_4 are removed from the graph (the reason therefore is that v_4 is the common neighbor of nodes v_3 and v_5); as well as all edges incident with any of these nodes. False edges are added between the set of nodes which are only neighbors of v_3 and not of v_5 and the set of nodes which are only neighbors of v_5 and not of v_3. Hence, false edge v_2v_6 is added, and one gets the triangle shown in Fig. 7b. We recognize that the re-



a Graph G

b Graph $\overline{G}$ after edge-projection of edge v_3v_5

Stable Set Problem: Branch & Cut Algorithms, Figure 7
Edge protection

sulting graph is smaller and, indeed, more dense. We realize that the triangle inequality

$$x_1 + x_2 + x_6 \leq 1$$

is valid for $P_{\mathrm{STAB}}(\overline{G})$ but not for $P_{\mathrm{STAB}}(G)$. In most cases, the anti-projection adds the deleted nodes to the inequality and increases the right hand side by value one1. In this case we get

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 2\,,$$

which is a valid inequality for $P_{\mathrm{STAB}}(G)$. We recognize that it is a lifted odd-hole inequality, which is facet-defining for $P_{\mathrm{STAB}}(G)$; where the extension of a valid inequality for P to a valid inequality for a higher dimensional polytope $\overline{P} \supseteq P$ is called lifting. In general, it turns out that facet-defining inequalities for the polytope of the projected graph might not be facet-defining for the polytope of the original graph and vice versa. The method was successfully developed and implemented by Rossi and Smriglio, [69]. More details and polyhedral results can be found in [67].

Branching

The branching strategy in a Branch & Cut algorithm influences the overall performance of the algorithm very much. In general, it is very difficult to find a good strategy. Various techniques have been explored and none of them can always outperform the others. But for special problems, there are different strategies that help to reduce the size of the Branch & Bound tree and speedup the algorithm. We start this section with a motivation for the need of special techniques and present the branching idea of Balas and Yu from 1986, [7]. It still remains the most successful strategy in practice.

One standard branching idea for a problem with binary variables is to generate two subproblems. One variable is set to value one in one subproblem and to value zero in the other one. However, this branching strategy leads to a very unbalanced Branch & Bound tree for the maximum stable set problem. This can be seen by the following argument. Setting a variable to value one sets all nodes of its neighborhood to value zero. In contrast, setting a variable to value zero has no consequence for all other variables of the graph. To avoid this drawback, one could think to set in each subproblem of the tree at least one variable to value one. Basically, this is the key property of the branching strategy by Balas and Yu.

Let $G' = (V', E')$ be the subgraph induced by the set of nodes which are not fixed in a current subproblem. In each subproblem, the goal is to find a maximum stable set in the particular subgraph given by the tree, or to prove that $\alpha(G') < LB$, with the lower bound LB. Let $W \subseteq V'$, and assume that we can show that $\alpha(G[W]) \leq LB$. Clearly, if $W = V'$, the subproblem can be fathomed; otherwise, if $\alpha(G') > LB$ any maximum stable set must contain at least one node of set $Z{:=}V' \setminus W = \{v_1, \ldots, v_p\}$. Based on this observation, Balas and Yu showed that every maximum-cardinality stable set with greater weight than the lower bound must be contained in one of the sets

$$V_i'{:=}\{v_i\} \cup V' \setminus (\Gamma(v_i) \cup \{v_{i+1}, \ldots, v_p\}) \qquad \text{for } i = 1, \ldots, p\,.$$

Note, that this strategy is also true for the weighted case with $c \neq 1$. This branching leads to p new subproblems in one branching step. In each subproblem, node v_i is set to value one, and all nodes of $\Gamma(v_i) \cup \{v_{i+1}, \ldots, v_p\}$ are set to value zero.

Let us now discuss some properties of this strategy. The size of W and the ordering of the nodes in Z can affect the total number of subproblems to be solved. Of course, the larger W is, the fewer subproblems will be generated in that state. The size of W is strongly effected by the quality of the computed lower bound. Determining W is quite crucial and can be done for the cardinality stable set problem, for instance, with a clique covering, cf. [7,69]. Also other methods such as matchings or holes [71] have been considered. In addition, the choice of the branching variable also has a great impact on the number of subproblems being solved. The size of the tree can be reduced by branching on nodes with a high degree, which was empirically shown by Carraghan and Pardalos [25]. The reason is the previously mentioned observation that with the branching node its neighborhood is also set. To sort the nodes in each subproblem in ascending order of degree seems to be computationally expensive as the degree of all nodes has to be calculated in each branching step prior to sorting. However, Sewell [71] showed experimentally that for sparse graphs this is still convenient.

Implementation Aspects

In general, when implementing a solver for the stable set problem, the following two things are crucial. First, one has to obtain a good lower bound, which means in the case of maximization, a feasible solution. One should use one of the many suggested heuristics in literature. We refer to the article ► heuristics for maximum clique and independent set. Second, it is recommended to have a strong preprocessing. This becomes even more important when the graphs result from applications. Many contributions have been made for this purpose. Among them are, for instance, fixing of nodes, fixing of cliques and treating connected components separately, [56,67,72].

For the case of a Branch & Cut algorithm in particular, one first needs a formulation of the stable set problem. We discussed some of the relaxations in Sect. "Stable Set Polytope". For practical efficiency, it is not recommended to start with the optimization over $P_{\mathrm{RSTAB}}(G)$. The reasons are that this relaxation is very weak and contains relatively many constraints. A better idea is to start with maximal cliques which contain all edges. Such a covering can be found in linear time. The resulting relaxation is stronger, as each maximal clique is facet-defining and dominates all the edge inequalities contained. Recognize that for bipartite graphs the relaxations is the same for both methods.

If one decides to separate several classes of inequalities within a Branch & Cut framework, one needs an order in which the separation routines are called. It is recommended to first call the polynomial separation routines, then the ones which take higher computational effort. However, practical tests show that the clique inequalities are very important. Therefore, a Branch & Cut solver should focus on fast heuristic separation of the clique inequalities combined with the very powerful tool of edge-projection. This leads to the best computational results so far.

Moreover, it is recommended to focus on facet-defining inequalities. Therefore, each clique should be lifted to a maximal clique before its inequality is added to the formulation. Accordingly, each odd-cycle should be checked not to contain any chords, and if so, the smaller odd-cycle would be added instead. These transformations can be done in linear time.

More details regarding implementation and Branch & Cut modules for the stable set problem can be found, for instance, in [10,24,67,69].

Conclusion

Many contributions have been made to solve the stable set problem exactly. One of the exact algorithms is based on the Branch & Cut method. However, the stable set polytope is not yet fully understood, and the known inequalities are either easy to separate with little impact, or they can only be separated with a large amount of computational effort and are very crucial for polyhedral approaches to the stable set problem. Therefore, it is not a surprise that there are some stable set instances with less than 1000 nodes which cannot be solved exactly with current methods.

See also

- ► Heuristics for Maximum Clique and Independent Set
- ► Integer Programming
- ► Integer Programming: Branch and Bound Methods
- ► Integer Programming: Branch and Cut Algorithms
- ► Integer Programming: Cutting Plane Algorithms

► Lovász Number
► Simplicial Pivoting Algorithms for Integer Programming

References

1. Applegate D, Bixby R, Chvátal V, Cook W (2001) TSP Cuts Which Do Not Conform to the Template Paradigm. Comput Comb Optim LNCS, vol 2241:157–222
2. Alekseev VE (2003) On easy and hard hereditary classes of graphs with respect to the independent set problem. Discret Appl Math 132(1–3):17–26
3. Arora S, Safra S (1992) Probabilistic Checking of Proofs; a new Characterization of NP. In Proceedings 33rd IEEE Symposium on Foundations of Computer Science, pp 2–13. IEEE Computer Society, Los Angeles
4. Babel L, Tinhofer G (1990) A branch and bound algorithm for the maximum clique problem. ZOR-Methods Models Oper Res 34:207–217
5. Balas E, Xue J (1996) Weighted and Unweighted Maximum Clique Algorithms with Upper Bounds from Fractional Coloring. Algorithmica 15(5):397–412
6. Balas E, Yu CS (1989) On graphs with polynomially solvable maximum-weight clique problem. Networks 19(2):247–253
7. Balas E, Yu CS (1986) Finding a Maximum Clique in an Arbitrary Graph. SIAM J 14(4):1054–1068
8. Balas E, Chvátala V, Nešetřil J (1987) On the Maximum Weight Clique Problem. Math Oper Res 12(3):522–535
9. Balas E, Padberg MW (1976) Set Partitioning: A Survey. SIAM Rev 18(4):710–761
10. Balasa E, Ceria S, Cornuejols G, Pataki G (1996) Polyhedral methods for the maximum clique problem. In: Johnson DS, Trick MA (eds) American Mathematical Society. DIMACS vol 26, pp 11–28
11. Balinski ML (1970) On Maximum Matching, Minimum Covering and their Connections. In: Kuhn HW (ed) Proceedings of the Princeton symposium on mathematical programming. Princeton University Press, Princeton, pp 303–312
12. Barnes ER (2000) A Branch-and-Bound Procedure for the Largest Clique in a Graph. Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems. Kluwer, Boston
13. BHOSLIB (2000) Benchmarks with Hidden Optimum Solutions for Graph Problems (Maximum Clique, Maximum Independent Set, Minimum Vertex Cover and Vertex Coloring) – Hiding Exact Solutions in Random Graphs. http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm
14. Bhattacharya BK, Kaller D (1997) An O(m + n log n) Algorithm for the Maximum-Clique Problem in Circular-Arc Graphs. J Algorithms 25(3):336–358
15. Bomze IM, Budinich M, Pardalos PM, Pelillo M (1999) The Maximum Clique Problem. Handbook of Combinatorial Optimization. Kluwer, Boston
16. Bomze IM, Stix V (1999) Genetic engineering via negative fitness: Evolutionary dynamics for global optimization. Annals Oper Res 89:297–318
17. Bonomo F, Durán G, Lin MC, Szwarcfiter JL (2005) On Balanced Graphs. Math Program 105(2–3):233–250
18. Bourjolly J-M, Laporte G, Mercure H (1997) A combinatorial column generation algorithm for the maximum stable set problem. Oper Res Lett 20(1):21–29
19. Bron C, Kerbosch J (1973) Algorithm 457: Finding all cliques on an undirected graph. Commun ACM 16:575–57
20. Brønsted A (1983) An introduction to Convex Polytopes. Graduate Texts in Mathematics, vol 90. Springer, New York
21. Burer S, Monteiro RDC, Zhang Y (2002) Maximum stable set formulations and heuristics based on continuous optimization. Math Program 94(1):137–166
22. Butenko S (2003) Maximum Independent Set and Related Problems, with Applications. PhD thesis, University of Florida
23. Caprara A, Fiscetti M, Letchford AN (2000) On the Separation of Maximally Violated mod-*k* Cuts. Math Program 87(1):37–56
24. Carr RD, Lancia G, Istrail S (2000) Branch-and-Cut Algorithms for Independent Set Problems: Integrality Gap and An Application to Protein Structure Alignment. Technical report, Sandia National Laboratories, Albuquerque, US; Sandia National Laboratories, Livermore
25. Carraghan R, Pardalos PM (1990) An exact algorithm for the maximum clique problem. Oper Res Lett 9(6):375–382
26. Cheng E, Cunningham WH (1995) Separation problems for the stable set polytope. In: Balas E, Clausen J (eds) The 4th Integer Programming and Combinatorial Optimization Conference Proceedings. pp 65–79
27. Cheng E, Cunningham WH (1997) Wheel Inequalities for Stable Set Polytopes. Math Program 77:389–421
28. Cheng E, de Vries S (2002) Antiweb-wheel inequalities and their separation problems over the stable set polytopes. Math Program 92(1):153–175
29. Chiba N, Nishizeki T (1985) Arboricity and subgraph listing algorithms. SIAM J 14:210–223
30. Chudnovsky M, Cornuéjols G, Liu X, Seymour P, Vušković K (2005) Recognizing Berge Graphs. Comb 25(2):143–186
31. Chudnovsky M, Robertson N, Seymour P, Thomas R (2004) The strong perfect graph theorem. Ann Math 164:51229
32. Cogisa O, Thierry E (2005) Computing maximum stable sets for distance-hereditary graphs. Discret Optim 2(3):185–188
33. Diestel R (2000) Graph Theory. Electronic Edition 2000. Springer, New York
34. Second DIMACS Challenge, 1992/1993. http://mat.gsia.cmu.edu/challenge.html
35. Fahle T (2002) Simple and Fast: Improving a Branch-And-Bound Algorithm for Maximum Clique, vol 2461/2002 Lecture Notes in Computer Science pp 485–498
36. Fujisawa K, Morito S, Kubo M (1995) Experimental Analyses of the Life Span Method for the Maximum Stable Set Prob-

lem. The Institute of Statistical Mathematics Cooperative Research Report 75:135–165
37. Garey MR, Johnson DS (1979) Computers and Intractability, A guide to the Theory of NP-Completeness. In: Klee V (ed) A series of books in the mathematical sciences. Freeman WH and Company, New York
38. Gerards AMH, Schrijver A (1986) Matrices with the Edmonds-Johnson property. Comb 6(4):365–379
39. Giandomenico M, Letchford AN (2006) Exploring the Relationship Between Max-Cut and Stable Set Relaxations. Math Program 106(1):159–175
40. Grötschel M, Jünger M, Reinelt G (1984) A Cutting Plane Algorithm for the Linear Ordering Problem. Oper Res 32:1195–1220
41. Grötschel M, Pulleyblank WR (1981) Weakly Bipartite Graphs and the Max-cut Problem. Oper Res Lett 1(1):23–27
42. Grötschel M, Lovász L, Schrijver A (1988) Geometric Algorithms and Combinatorial Optimization. Algorithms and Combinatorics 2. Springer, Berlin
43. Grötschel M, Lovász L, Schrijver A (1981) The Ellipsoid Method and Its Consequences in Combinatorial Optimization. Comb 1:169–197
44. Harary F, Ross IC (1957) A procedure for clique detection using the group matrix. Sociom 20:205–215
45. Hasselberg J, Pardalos PM, Vairaktarakis G (1993) Test case generators and computational results for the maximum clique problem. J Glob Optim 3(4):463–482
46. Kallrath J, Wilson JM (1997) Business Optimization using Mathematical Programming. Macmillan, New York
47. Lawler E (2001) Combinatorial Optimization: Networks and Matroids. Reprint of the 1976 original. Dover Publications, Inc., Mineola
48. Lehmann KA, Kaufmann M, Steigele S, Nieselt K (2006) On the maximal cliques in c-max-tolerance graphs and their application in clustering molecular sequences. Algorithm Molecular Biol 1:9:1–17
49. Loukakis E, Tsouros C (1981) A depth first serach algorithm to generate the family of maximal independet sets of a graph lexicographically. Comput 27:249–266
50. Lovász L (1979) On the Shannon capacity of a graph. IEEE Trans Inform Theory 25(1):1–7
51. Mannino C, Sassano A (2005) An exact algorithm for the maximum stable set problem. Comput Optim Appl 3(3):243–258
52. Mannino C, Sassano A (1996) Edge Projection and the Maximum Cardinality Stable Set Problem. DIMACS Series Discret Math Theor Comput Sci 26:249–261
53. Mannino C, Stefanutti E (1999) An augmentation algorithm for the maximum weighted stable set problem. Comput Optim Appl 14(3):367–381
54. Masuda S, Nakajima K, Kashiwabara T, Fujisawa T (1990) Efficient algorithms for finding maximum cliques of an overlap graph. Networks 20(2):157–171
55. Mosca R (1997) Polynomial algorithms for the maximum stable set problem on particular classes of p5-free graphs. Inf Process Lett 61(3):137–143
56. Nemhauser GL, Trotter LE Jr (1975) Vertex Packings: Structural Properties and Algorithms. Math Program 8:232–248
57. Nemhauser GL, Trotter LE Jr (1974) Properties of Vertex Packing and Independence System Polyhedra. Math Program 6:48–61
58. Nemhauser GL, Wolsey LA (1988) Integer and Combinatorial Optimization. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, New York
59. Olariu S (1989) Weak bipolarizable graphs. Discret Math 74(1–2):159–171
60. Östergård PRJ (2002) A fast algorithm for the maximum clique problem. Discret Appl Math 120(1–3):197–207
61. Padberg MW, Rinaldi G (1987) Optimization of a 532 City Symmetric Traveling Salesman Problem by Branch and Cut. Oper Res Lett 6:1–7
62. Padberg MW (1973) On the Facial Structure of Set Packing Polyhedra. Math Prog 5:199–215
63. Papadimitriou CH, Yannakakis M (1981) The clique problem for planar graphs. Inf Process Lett 13(4–5):131–133
64. Pardalos PM, Phillips AT (1990) A global optimization approach for solving the maximum clique problem. Int J Comput Math 33(3–4):209–216
65. Pardalos PM, Rodgers GP (1992) A branch and bound algorithm for the maximum clique problem. Comput Oper Res 19(5):363–375
66. Ramírez-Alfonsín JL, Reed BA (eds) (2001) Perfect Graphs, Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, New York
67. Rebennack S (2006) Maximum Stable Set Problem: A Branch & Cut Solver. Diplomarbeit, Ruprecht–Karls Universität Heidelberg, Heidelberg, Germany
68. Régin J-C (2003) Using constraint Programming to Solve the Maximum Clique Problem. Lecture Notes in Computer Science. Springer, Berlin, pp 634–648
69. Rossi F, Smriglio S (2001) A Branch-and-Cut Algorithm for the Maximum Cardinality Stable Set Problem. Oper Res Lett 28:63–74
70. Schrijver A (2003) Combinatorial Optimization: Polyhedra and Efficiency, vol 24 of Algorithms and Combinatorics. Springer, Berlin
71. Sewell EC (1998) A Branch and Bound Algorithm for the Stability Number of a Sparse Graph. INFORMS J Comput 10(4):438–447
72. Strijk T, Verweij B, Aardal K (2000) Algorithms for maximum independent set applied to map labelling. Technical Report UU-CS-2000–22, http://citeseer.ist.psu.edu/article/strijk00algorithms.html
73. Tomita E, Tanaka A, Takahashi H (1988) The worst-time complexity for finding all the cliques. Technical report, University of Electro-Communications, Tokyo, Japan
74. Trotter LE (1975) A class of facet producing graphs for vertex packing polyhedra. Discret Math 12(4):373–388

75. Verweij B, Aardal K (1999) An Optimisation Algorithm for Maximum Independent Set with Applications in Map Labelling, vol 1643/1999 Lecture Notes in Computer Science, pp 426–437
76. Warren JS, Hicks IV (2006) Combinatorial Branch-and-Bound for the Maximum Weight Independent Set Problem. working paper, August 7
77. Warrier D, Wilhelm WE, Warren JS, Hicks IV (2005) A branch-and-price approach for the maximum weight independent set problem. Network 46(4):198–209
78. West DB (2000) Introduction to Graph Theory, 2nd edn. Prentice Hall
79. Wolsey LA (1998) Integer Programming. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, New York
80. Wood DR (1997) An algorithm for finding a maximum clique in a graph. Oper Res Lett 21(5):211–217
81. Yildirim EA, Fan-Orzechowski X (2006) On Extracting Maximum Stable Sets in Perfect Graphs Using Lovász's Theta Function. Comput Optim Appl 33(2–3):229–247
82. Ziegler GM (1995) Lecture on Polytopes. Graduate Texts in Mathematics. Springer, New York

Standard Quadratic Optimization Problems: Algorithms

IMMANUEL M. BOMZE
University Vienna, Wien, Austria

MSC2000: 90C20

Article Outline

Keywords
See also
References

Keywords

Global optimization; Interior point; Copositivity; Escape step; Replicator dynamics

A *standard quadratic optimization problem* (StQP) consists of finding (global) maximizers of a quadratic form over the *standard simplex* Δ in n-dimensional Euclidean space $\mathbf{R}^n$,

$$\Delta = \left\{\mathbf{x} \in \mathbb{R}^n : \; x_i \geq 0 \text{ for all } i \in N, \; \mathbf{e}^\top \mathbf{x} = 1\right\},$$

where $N = \{1, \ldots, n\}$; a $^\top$ denotes transposition; and $\mathbf{e} = [1, \ldots, 1]^\top \in \mathbf{R}^n$. Hence a StQP can be written as a (global) quadratic optimization problem of the form

$$\max\left\{f(\mathbf{x}) = \mathbf{x}^\top R\mathbf{x} : \; \mathbf{x} \in \Delta\right\}, \tag{1}$$

where R is an arbitrary symmetric $n \times n$ matrix. Quadratic optimization problems like (1) are *NP*-hard [2], even regarding the detection of local solutions. Nevertheless, there are several procedures which try to exploit favorable data constellations in a systematic way, and to avoid the worst-case behavior whenever possible. Examples for this type of algorithms are specified below.

First we concentrate on the evolutionary approach to finding local solutions of StQPs. To this end, consider the following *dynamical system* operating on Δ:

$$\begin{aligned} &\dot{x}_i(\tau) = x_i(\tau)[(R\mathbf{x}(\tau))_i - \mathbf{x}(\tau)^\top R\mathbf{x}(\tau)], \\ &i \in N, \end{aligned} \tag{2}$$

where a dot signifies derivative w.r.t. time τ, and a discrete time version

$$x_i(\tau + 1) = x_i(\tau)\frac{[R\mathbf{x}(\tau)]_i}{\mathbf{x}(\tau)^\top R\mathbf{x}(\tau)}, \quad i \in N. \tag{3}$$

The *stationary points* under (2) and (3) coincide, and all local solutions of (1) are among these (see below). A stationary point $\mathbf{x}$ is said to be *asymptotically stable*, if every solution to (2) or (3) which starts close enough to $\overline{\mathbf{x}}$, will converge to $\overline{\mathbf{x}}$ as $\tau \nearrow \infty$. Now the following results hold (for proofs and further characterization results linking optimization theory, evolutionary game theory, and qualitative theory of dynamical systems, see [1] and the references therein):

- the objective $f(\mathbf{x}(\tau))$ increases strictly along non-constant trajectories of (2) and (3);
- all trajectories converge to a stationary point;
- all Karush–Kuhn–Tucker points and hence all local solutions of (1) are stationary points under (2) and (3);
- if no principal minor of $R = R^\intercal$ vanishes, then with probability one (regarding the choice of $\mathbf{x}(0)$, the starting point), any trajectory of (2) converges to one of the strict local solutions $\overline{\mathbf{x}}$ of (1), which coincide with the asymptotically stable points under (2) and (3);
- further, $\mathbf{y}^\top R\mathbf{y} < \overline{\mathbf{x}}^\top R\overline{\mathbf{x}}$ for all $\mathbf{y} \in \Delta$ with $\mathbf{y} \neq \overline{\mathbf{x}}$ but $y_i = 0$ if $\overline{x}_i = 0$;.

1 Given a local solution $\bar{\mathbf{x}}$ to (1), remove all alleles which are not unfit, i.e. all $i \in S = \{i \in N : \bar{x}_i > 0\}$;

2 determine a (local) *fitness minimizer* $\mathbf{y}$ in the reduced problem, i.e. consider problem (1) with R replaced by

$$\bar{R} = [\gamma s - r_{ij}]_{i,j \in N \setminus S},$$

where $\gamma s = \max x_{i,j \in N \setminus S}\, r_{ij}$;

3 with a local solution $\mathbf{y}$ of this auxiliary problem, put

$$J = \{j \in N \setminus S : y_i > 0\}$$

and denote by m the cardinality of J;

4 for all $s \in S$ and $t \in J$, consider the reduced problem $\mathcal{P}_{t \to s}$, i.e. problem (1) in $n - m$ variables for the $(n - m) \times (n - m)$ matrix $R_{t \to s}$ obtained from R as follows: replace r_{si} with r_{ti} and remove all other $j \in J$;

5 $\bar{\mathbf{x}}$ is a global solution to the master problem (1) if and only if for all $(s, t) \in S \times J$, the maximum of $\mathcal{P}_{t \to s}$ does not exceed the current best value $\bar{\mathbf{x}}^\top R \bar{\mathbf{x}}$;

6 in the negative, i.e. if $\mathbf{u}^\top R_{t \to s} \mathbf{u} > \bar{\mathbf{x}}^\top R \bar{\mathbf{x}}$ for some $\mathbf{u} \in \Delta \subset \mathbf{R}^{n-m}$, and if $j \in J$ is chosen such that for all $q \in J$

$$\sum_{p \notin J \cup \{s\}} (r_{jp} - r_{qp}) u_p \geq \frac{1}{2}(r_{qq} - r_{jj}) u_s\,,$$

then a strictly improving feasible point $\tilde{\mathbf{x}}$ is obtained as follows:

$$\tilde{\mathbf{x}}_q = \begin{cases} u_t & \text{if } q = j, \\ 0 & \text{if } q \in J \cup \{s\} \setminus \{j\}, \\ u_q & \text{if } q \in N \setminus J. \end{cases}$$

GENF procedure to escape from inefficient local solutions in StQPs

Although strictly increasing objective values are guaranteed as trajectories under (2) or (3) are followed, one could get stuck in an inefficient local solution of (1). One possibility to escape is the *genetic engineering via negative fitness* (GENF) approach [1] described in the sequel. From the properties above, a strict local solution $\widetilde{\mathbf{x}}$ of(1) must be a global one if all $\bar{x}_i > 0$. Consequently, at an inefficient local solution necessarily $\bar{x}_i = 0$ for some i. In the usual genetic interpretation of the dynamics (2) and (3), this means that some alleles die out during the natural selection process, and these are therefore unfit in the environment currently prevailing. The escape step now artificially re-introduces some alleles which would have gone extinct during the natural selection process, and restarts with a smaller subproblem which will yield an improvement if $\bar{\mathbf{x}}$ is inefficient, see the table above.

In view of the possible combinatorial explosion in effort with increasing number of variables, this dimension reducing strategy seems to be promising: if k is the size of S, the above result yields a series of km StQPs in $n - m$ variables rather than in n. We are now ready to describe the algorithm which stops after finitely many repetitions, since it yields strict local solutions with strictly increasing objective values [1].

1 Start with $\mathbf{x}(0) = [1/n, ..., 1/n]^\top$ or nearby, iterate (3) until convergence;

2 the limit $\bar{\mathbf{x}} = \lim_{\tau \to \infty} \mathbf{x}(\tau)$ is a strict local solution with probability one; call the GENF procedure to improve the objective, if possible; denote the improving point by $\tilde{\mathbf{x}}$;

3 repeat 1), starting with $\mathbf{x}(0) = \tilde{\mathbf{x}}$

Replicator dynamics algorithm for StQPs

A different approach towards global solutions of StQPs uses familiar branch and bound schemes. For ease of exposition, now consider the minimization StQP

$$\min \left\{ \mathbf{x}^\top Q \mathbf{x} : \ \mathbf{x} \in \Delta \right\}, \tag{4}$$

and assume without loss of generality that Q has only positive entries (otherwise replace Q with $Q + \gamma \mathbf{e}\mathbf{e}^\top$ where γ is suitably large). If one applies a usual simplicial partition [2] (cf. also ► Simplicial decomposition) to Δ, all subproblems are again StQPs. To obtain lower bounds for these problems, convex minorants for the objective $\mathbf{x}^\top Q \mathbf{x}$ on Δ may be used, e. g.quadratic forms $\mathbf{x}^\top F \mathbf{x}$ with F positive semidefinite (or some related matrix $\widehat{F}$ which ensures that the minorant is convex necessarily over Δ only), where F is chosen such that the gap between the objectives is small. This can be accomplished by requiring diag F = diag$Q$ and that $\sum^{i,j}[q_{ij} -$

f_{ij}] is small, while the minorant condition is guaranteed by the requirement $f_{ij} \le q_{ij}$ for all $i, j \in N$. Therefore one arrives at a semidefinite optimization problem (SDP; cf. ► Semidefinite programming: Optimality conditions and stability) which can be solved by the usual methods. The resulting matrix F then gives a convex problem, so that the desired lower bound for(4), $\{\min \mathbf{x}^\mathsf{T} F \mathbf{x}: \mathbf{x} \in \Delta\}$ can be obtained efficiently, e. g. via local search techniques or linear complementarity approaches (cf. also ► Interval analysis: Eigenvalue bounds of interval matrices). For details and results see [3].

See also

- ► Complexity Theory: Quadratic Programming
- ► Quadratic Assignment Problem
- ► Quadratic Fractional Programming: Dinkelbach Method
- ► Quadratic Knapsack
- ► Quadratic Programming with Bound Constraints
- ► Quadratic Programming Over an Ellipsoid
- ► Standard Quadratic Optimization Problems: Applications
- ► Standard Quadratic Optimization Problems: Theory

References

1. Bomze IM, Stix V (1999) Genetical engineering via negative fitness: evolutionary dynamics for global optimization. Ann Oper Res 89:297–318
2. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
3. Nowak I (1999) A new semidefinite programming bound for indefinite quadratic forms over a simplex. J Global Optim 14:357–364

Standard Quadratic Optimization Problems: Applications

Immanuel M. Bomze
University Vienna, Wien, Austria

MSC2000: 90C20

Article Outline

Keywords
See also
References

Keywords

Maximum weight clique; Portfolio selection; Simplicial partition; Branch and bound

A *standard quadratic optimization problem* (StQP) consists of finding (global) maximizers of a quadratic form over the *standard simplex* in n-dimensional Euclidean space $\mathbf{R}^n$,

$$\Delta = \left\{\mathbf{x} \in \mathbb{R}^n : \; x_i \ge 0 \quad \text{for all } i \in N, \; \mathbf{e}^\top \mathbf{x} = 1\right\},$$

where $N = \{1, \ldots, n\}$; a $^\mathsf{T}$ denotes transposition; and $\mathbf{e} = \sum_{i \in N} \mathbf{e}_i = [1, \ldots, 1]^\mathsf{T} \in \mathbf{R}^n$, with $\{\mathbf{e}_i : i \in N\}$ the vertices of Δ. Hence a StQP can be written as a (global) quadratic optimization problem of the form

$$\max \left\{\mathbf{x}^\top R \mathbf{x} : \; \mathbf{x} \in \Delta\right\}, \tag{1}$$

where R is an arbitrary symmetric $n \times n$ matrix.

An important application for StQPs is the search for a maximum weight clique arising in computer vision, pattern recognition and robotics (see [2] for a more detailed account): consider an undirected graph $\mathcal{G} = (N, \mathcal{E})$ with n nodes, and a weight vector $\mathbf{w} = [w_1, \ldots, w_n]^\mathsf{T}$ of positive weights w_i associated to the nodes $i \in N$. A *clique* S is a subset of N which corresponds to a complete subgraph of $\mathcal{G}$ (i. e. any pair of different nodes in S is an edge in $\mathcal{E}$). A clique S is said to be *maximal* if there is no larger clique containing S. Every clique S in $\mathcal{G}$ has a weight $W(S) = \sum_{i \in S} w_i$. The *maximum weight clique problem* (MWCP) consists of finding a clique in the graph which has largest total weight. The classical (unweighted) *maximum clique problem* is a special case with $\mathbf{w} = \mathbf{e}$. To reformulate the MWCP as a StQP, one may exploit an idea of L. Lovász in considering the following class of symmetric $n \times n$ matrices: let

$$\mathcal{C}(\mathcal{E}) = \left\{(c_{ij})_{i,j \in N} : \; c_{ij} = 0 \quad \text{if } (i,j) \in \mathcal{E}\right\},$$

as well as $\mathcal{C}_+(\mathcal{G}) = C \in \mathcal{C}(\mathcal{E})$: $C^\mathsf{T} = \{C$ and $c_{ij} \ge c_{ii} + c_{jj}$ if $(i,j) \notin \mathcal{E}\}$, and form the class

$$\mathcal{C}(\mathcal{G}, \mathbf{w}) = \left\{ C \in \mathcal{C}_+(\mathcal{G}) : \; c_{ii} = \frac{1}{2w_i} \text{ for all } i \right\}.$$

Now consider the (minimization) StQP

$$\min \left\{\mathbf{x}^\top C \mathbf{x} : \; \mathbf{x} \in \Delta\right\} \tag{2}$$

for some matrix $C \in \mathcal{C}(\mathcal{G}, \mathbf{w})$. Given a subset $S \subseteq N$, define the *S-face* of Δ as

$$\Delta_S = \{\mathbf{x} \in \Delta\colon\ x_i = 0 \text{ if } i \notin S\}$$

and its *weighted barycenter* as $\mathbf{x}^S = \sum_{i \in S} (w_i / W(S))\mathbf{e}_i \in \Delta_S$. Then the following assertions hold [2]:

- A point $\mathbf{x} \in \Delta$ is a local solution to problem (2) if and only if $\mathbf{x} = \mathbf{x}^S$, where S is a maximal clique.
- A vector $\mathbf{x} \in \Delta$ is a global solution to problem (2) if and only if $\mathbf{x} = \mathbf{x}^S$, where S is a maximum weight clique.
- Moreover, all local (and hence global) solutions to (2) are strict.

Note that a different class used in [3] does not share these properties. The class $\mathcal{C}(\mathcal{G}, \mathbf{w})$ is isomorphic to the positive orthant in $\binom{n}{2} - e$ dimensions where e is the cardinality of $\mathcal{E}$. This class is a polyhedral pointed cone with its apex given by the matrix $C^{\mathcal{G},\mathbf{w}}$ with entries

$$c_{ij}^{\mathcal{G},\mathbf{w}} = \begin{cases} \frac{1}{2w_i} & \text{if } i = j, \\ \frac{1}{2w_i} + \frac{1}{2w_j} & \text{if } i \neq j \text{ and } (i,j) \notin \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$$

In the unweighted case $\mathbf{w} = \mathbf{e}$, we get $C^{\mathcal{G},\mathbf{e}} = (\frac{1}{2})I + A_{\overline{\mathcal{G}}}$ where $A_{\overline{\mathcal{G}}}$ is the adjacency matrix of the complement graph $\overline{\mathcal{G}}$, and I the $n \times n$ identity matrix. Therefore (2) can be seen as a regularized generalization of the original approach of T.S. Motzkin and E.G. Straus [7].

Another application of StQP is concerned with the mean/variance *portfolio selection* problem (see, e. g. [6]; ► Portfolio selection and multicriteria analysis) which can be formalized as follows: suppose there are n securities to invest in, at an amount expressed in relative shares $x_i \geq 0$ of an investor's budget. Thus, the budget constraint reads $\mathbf{e}^\mathsf{T}\mathbf{x} = 1$, and the set of all feasible portfolios (investment plans) is given by Δ. Now, given the expected return m_i of security i during the forthcoming period, and an $n \times n$ covariance matrix V across all securities, the investor faces the multi-objective problem to maximize the expected return $\mathbf{m}^\mathsf{T}\mathbf{x}$ and simultaneously minimize the risk $\mathbf{x}^\mathsf{T}\ V\mathbf{x}$ associated with her decision $\mathbf{x}$.

One of the most popular approaches to such type of problems in general applications is that the user prespecifies a parameter β which in her eyes balances the benefit of high return and low risk, i. e. consider the parametric QP

$$\max\left\{f_\beta(\mathbf{x}) = \mathbf{m}^\top\mathbf{x} - \beta\mathbf{x}^\top V\mathbf{x}\colon\ \mathbf{x} \in \Delta\right\}.$$

For fixed β, this is a StQP. Anyhow, the question remains how to choose β. In finance applications, the notion of market portfolio is used to determine a reasonable value for this parameter. This emerges more or less from an exogenous artefact, namely by introducing a completely risk-free asset which is used to scale return versus risk [5]. An alternative, purely endogenous derivation of market portfolio could use a result of M.J. Best and B. Ding [1] who consider the problem

$$\max_{\beta > 0} \max_{\mathbf{x} \in \Delta} \frac{1}{\beta} f_\beta(\mathbf{x}), \tag{3}$$

and show how optimal solutions $(\beta^*, \mathbf{x}^*)$ for (3) emerge from a single StQP (1) with, e. g. $R = 2\mathbf{m}\mathbf{m}^\mathsf{T} - V$.

A general application of StQPs arises if one applies branch and bound schemes with simplicial partitions [4] (cf. also ► Simplicial decomposition) to general quadratic problems of the form

$$\max\left\{g(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top Q\mathbf{x} + \mathbf{c}^\top\mathbf{x}\colon\ \mathbf{x} \in M\right\},$$

where $M = \{\mathbf{x} \in \mathbf{R}^n\colon A\mathbf{x} \leq \mathbf{b}\}$ with A an $m \times n$ matrix and Q a symmetric $n \times n$ matrix. A subproblem then is of the form $\max\{g(\mathbf{x})\colon \mathbf{x} \in P\}$ with $P \cap M \neq \emptyset$ and $P = \text{conv}\{\mathbf{v}_0, \ldots, \mathbf{v}_n\}$ for some points $\mathbf{v}_i \in \mathbf{R}^n$ (if all vertices of M are easy to determine, one could even take them rather than the $\mathbf{v}_i$). With the $n \times (n+1)$ matrix $U = [\mathbf{v}_0, \ldots, \mathbf{v}_n]$, the subproblem reduces to the StQP

$$\max\left\{\mathbf{y}^\top R\mathbf{y}\colon\ \mathbf{y} \in \Delta\right\}$$

where $R = U^\mathsf{T}\ QU + \mathbf{e}\mathbf{c}^\mathsf{T}\ U + U^\mathsf{T}\ \mathbf{c}\mathbf{e}^\mathsf{T}$ is a symmetric $(n+1) \times (n+1)$ matrix and $\Delta \subseteq \mathbf{R}^{n+1}$. Efficient bounds can thus be obtained with one of the algorithms for obtaining (local) solutions to a StQP.

See also

- ► Complexity Theory: Quadratic Programming
- ► Quadratic Assignment Problem
- ► Quadratic Fractional Programming: Dinkelbach Method
- ► Quadratic Knapsack
- ► Quadratic Programming with Bound Constraints

- ▶ Quadratic Programming Over an Ellipsoid
- ▶ Standard Quadratic Optimization Problems: Algorithms
- ▶ Standard Quadratic Optimization Problems: Theory

References

1. Best MJ, Ding B (1997) Global and local quadratic minimization. J Global Optim 10:77–90
2. Bomze IM (1998) On standard quadratic optimization problems. J Global Optim 13:369–387
3. Gibbons LE, Hearn DW, Pardalos PM, Ramana MV (1997) Continuous characterizations of the maximum clique problem. Math Oper Res 22:754–768
4. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
5. Huang C-F, Litzenberger RH (1988) Foundations for financial economics, 6th printing. North-Holland, Amsterdam
6. Markowitz HM (1995) The general mean-variance portfolio selection problem. In: Howison SD, Kelly FP, Wilmott P (eds) Mathematical models in finance. Chapman and Hall, London, pp 93–99
7. Motzkin TS, Straus EG (1965) Maxima for graphs and a new proof of a theorem of Turán. Canad J Math 17:533–540

Standard Quadratic Optimization Problems: Theory

IMMANUEL M. BOMZE
University Vienna, Wien, Austria

MSC2000: 90C20

Article Outline

Keywords
See also
References

Keywords

Global optimization; Interior point; Copositivity

A *standard quadratic optimization problem* (StQP) consists of finding (global) maximizers of a (possibly indefinite) quadratic form over the *standard simplex* Δ in n-dimensional Euclidean space $\mathbf{R}^n$,

$$\Delta = \left\{\mathbf{x} \in \mathbf{R}^n\colon\ x_i \geq 0 \text{ for all } i \in N,\ \mathbf{e}^\top \mathbf{x} = 1\right\},$$

where $N = \{1, \ldots, n\}$; $a^\top$ denotes transposition; and $\mathbf{e} = [1, \ldots, 1]^\top \in \mathbf{R}^n$. Hence a StQP can be written as a (global) quadratic optimization problem of the form

$$\max\left\{\mathbf{x}^\top R\mathbf{x}\colon\ \mathbf{x} \in \Delta\right\}, \tag{1}$$

where R is an arbitrary symmetric $n \times n$ matrix. Since the maximizers of (1) remain the same if R is replaced with $R + \gamma \mathbf{e}\mathbf{e}^\top$ where γ is an arbitrary constant, one may assume without loss of generality that all entries of R are positive. Furthermore, the question of finding maximizers of a general quadratic function $\mathbf{x}^\top Q\mathbf{x} + 2\mathbf{c}^\top \mathbf{x}$ over Δ can be homogenized in a similar way by considering the rank-two update $R = Q + \mathbf{e}\mathbf{c}^\top + \mathbf{c}\mathbf{e}^\top$ in (1) which has the same objective values on Δ.

StQPs arise in procedures which enable an escape from inefficient local solutions of general quadratic optimization problems (QPs): consider the general quadratic maximization problem

$$\max\left\{f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top Q\mathbf{x} + \mathbf{c}^\top \mathbf{x}\colon\ \mathbf{x} \in M\right\}, \tag{2}$$

where $M = \{\mathbf{x} \in \mathbf{R}^n\colon A\mathbf{x} \leq \mathbf{b}\}$x with A an $m \times n$ matrix and Q a symmetric $n \times n$ matrix. To formulate a characterization of global optimality of a Karush–Kuhn–Tucker point $\overline{\mathbf{x}}$ for (2), first add a trivial nonbinding constraint, i. e. the most elementary strict inequality $0 < 1$, to obtain slacks $\overline{\mathbf{u}}$ as follows: denote by $\mathbf{a}_i^\top$ the ith row of A and put $\mathbf{a}_0 = \mathbf{o}$. Similarly put $b_0 = 1$ and enrich $\overline{A} = [\mathbf{a}_0 | A^\top]^\top = [\mathbf{o}, \mathbf{a}_1, \ldots, \mathbf{a}_m]^\top$ as well as $\overline{\mathbf{b}} = [b_0 | \mathbf{b}^\top]^\top = [1, b_1, \ldots, b_m]^\top$ Finally, define $\overline{\mathbf{u}} = \overline{\mathbf{b}} - \overline{A}\overline{\mathbf{x}} \geq \mathbf{o}$ Then perform, for any $i \in \{0, \ldots, m\}$, a rank-one update of $\overline{A}$ and a rank-two update of Q, using the current gradient $\mathbf{g} = \nabla f(\overline{\mathbf{x}}) = Q\overline{\mathbf{x}} + \mathbf{c}$ of the objective:

$$D_i = \overline{\mathbf{u}}\mathbf{a}_i^\top - \overline{u}_i \overline{A},$$
$$Q_i = -\mathbf{a}_i \mathbf{g}^\top - \mathbf{g}\mathbf{a}_i^\top - \overline{u}_i Q.$$

This gives a symmetric $n \times n$ matrix Q_i and a matrix D_i which is effectively $m \times n$ since its ith row is zero. Denoting by $J(\overline{\mathbf{x}})$ the set of all nonbinding constraints, the following result is proved in [1]:

Theorem 1 *A Karush–Kuhn–Tucker point $\overline{\mathbf{x}}$ of (2) is a global solution if and only if for all $i \in J(\overline{\mathbf{x}}) = \{i \in N \cup \{0\}\colon\ \overline{u}_i > 0\}$,*

$$\mathbf{v}^\top Q_i \mathbf{v} \geq 0 \quad \text{if } D_i \mathbf{v} \geq \mathbf{o}. \tag{3}$$

If $\boldsymbol{v}^\top Q_i \boldsymbol{v} < 0$ *for some* $\boldsymbol{v}$ *with* $D_i \boldsymbol{v} \geq \boldsymbol{o}$, *then*

$$\widetilde{\mathbf{x}} = \overline{\mathbf{x}} + \lambda \mathbf{v} \tag{4}$$

is an improving feasible point for $\lambda = \overline{u}_i/(\mathbf{a}_i^\top \mathbf{v})$ *(if* $\boldsymbol{a}_i^\top \boldsymbol{v} = 0$, *i. e.* $\lambda = +\infty$, *this means that (2) is unbounded).*

Determining whether or not (3) is satisfied, amounts to the question whether or not $\max\{\mathbf{v}^\top(-Q_i)\mathbf{v}\colon D_i\mathbf{v} \geq \mathbf{o}\} \leq 0$. Now this homogeneous problem is decomposable [2,3] into problems of the form $\max\{\mathbf{x}^\top R\mathbf{x}\colon \mathbf{x} \geq \mathbf{o}\}$, where the constraint $\mathbf{e}^\top\mathbf{x} = \sum_i x_i = 1$ can be added without loss of generality, rendering a StQP. In fact, in order to determine an improving feasible direction (4) it is not necessary to solve the latter problem to optimality, but rather sufficient to determine a feasible point $\mathbf{x} \in \Delta$ with $\mathbf{x}^\top R\mathbf{x} > 0$.

If the original problem (2) is itself already a StQP, then all checks of (3) can be reduced to a single one: if $\overline{\mathbf{x}} \in \Delta$ is any feasible point, then $\overline{\mathbf{x}}$ is a global maximizer of $\mathbf{x}^\top Q\mathbf{x}$ over Δ if and only if the matrix $\overline{Q} = (\overline{\mathbf{x}}^\top Q\overline{\mathbf{x}})\mathbf{e}\mathbf{e}^\top - Q$ satisfies $\mathbf{v}^\top\overline{Q}\mathbf{v} \geq 0$ if $\mathbf{v} \geq \mathbf{o}$, i. e. $\overline{Q}$ is *copositive*.

The close connection between StQPs and copositivity becomes evident if the usual semidefinite programming (SDP) approach is enlarged to recast a StQP into a linear optimization problem on a cone $\mathcal{K}$ which is the (pre) dual of the cone $\mathcal{K}^*$ of all copositive symmetric n × n matrices, with respect to the duality $\langle R, S\rangle$ = trace(RS) operating on pairs (R, S) of such matrices. This formulation allows to employ interior point algorithms (cf. also ► Sequential quadratic programming: Interior point methods for distributed optimal control problems; ► Interior point methods for semidefinite programming), similar to the methods used in SDP. Both cones $\mathcal{K}^*$ and $\mathcal{K}$ have nonempty interiors, and the latter can be described as follows [4,6]:

$$\mathcal{K} = \operatorname{conv}\left\{\mathbf{x}\mathbf{x}^\top\colon\ \mathbf{x} \geq \mathbf{o}\right\},$$

the convex hull of all symmetric rank-one matrices, i. e. dyadic products, generated by nonnegative vectors. Note that dropping the nonnegativity requirement, we would arrive at the positive semidefinite case. Now let $E = \mathbf{e}\mathbf{e}^\top$ be the $n \times n$ matrix of all ones. Since the extremal points of the set $\mathcal{L} = \{X \in \mathcal{K}\colon \langle E, X\rangle = 1\}$ are exactly the dyadic products $\mathbf{x}\mathbf{x}^\top$ with $\mathbf{x} \in \Delta$, a maximizer of $\langle R, X\rangle$ over $\mathcal{L}$ can be found which is of this form, and hence the StQP (1) is equivalent to the linear problem

$$\max\{\langle R, X\rangle\colon\ X \in \mathcal{K}, \langle E, X\rangle = 1\}.$$

It is easy to see that the dual formulation [5] of this problem is

$$\min\{y \in \mathbf{R}\colon\ yE - R \in \mathcal{K}^*\},$$

which is the task to find the smallest y such that $yE - R$ is copositive. Thus the dual problem is related to the question of eigenvalue bounds (replace E with the identity matrix and 'copositive' with 'positive semidefinite').

See also

- ► αBB Algorithm
- ► Complexity Theory: Quadratic Programming
- ► D.C. Programming
- ► Quadratic Assignment Problem
- ► Quadratic Fractional Programming: Dinkelbach Method
- ► Quadratic Knapsack
- ► Quadratic Programming with Bound Constraints
- ► Quadratic Programming Over an Ellipsoid
- ► Reverse Convex Optimization
- ► Standard Quadratic Optimization Problems: Algorithms
- ► Standard Quadratic Optimization Problems: Applications

References

1. Bomze IM (1992) Copositivity conditions for global optimality in indefinite quadratic programming problems. Czechoslovak J Oper Res 1:7–19
2. Cohen J, Hickey T (1979) Two algorithms for determining volumes of convex polyhedra. J ACM 26:401–414
3. Danninger G (1990) A recursive algorithm to detect (strict) copositivity of a symmetric matrix. In: Rieder U, Gessner P, Peyerimhoff A, Radermacher FJ (eds) Operations Research, Proc 14th Symp (Ulm, FRG, 1989), Methods Oper Res., vol 62. Anton Hain Verlag, Bodenheim, pp 45–52
4. Hall M Jr, Newman M (1963) Copositive and completely positive quadratic forms. Proc Cambridge Philos Soc 59:329–339
5. Nesterov YE, Nemirovskii AS (1994) Interior point methods in convex programming: Theory and applications. SIAM, Philadelphia
6. Quist AJ, de Klerk E, Roos C, Terlaky T (1998) Copositive relaxation for general quadratic programming. Optim Methods Softw 9:185–209

State of the Art in Modeling Agricultural Systems

PETRAQ PAPAJORGJI
IFAS, Information Technology Office,
University of Florida, Gainesville, USA

Article Outline

Introduction
The Unified Modeling Language
  Examples of UML Models in Agriculture
  The OCL
  Examples of Using OCL in Agriculture
The Design Patterns
  Example 1 of Using Design Patterns in Agriculture
  Example 2 of Using Design Patterns in Agriculture
The MDA Approach
Conclusion
References

Introduction

Historically, the use of new technologies in agriculture and related sciences has been relatively behind that in the industrial sector. Usually, for a technology that is already part of the mainstream technologies in the industrial sector, it takes time to be accepted by the community of researchers in agriculture-related areas. One of the reasons for the technological gap between the industrial and agricultural sectors could be related to the modest amounts of investment made in the field of agriculture compared with the impressive amounts and efforts the industrial sector invests in new technologies. Another reason could be the relatively slow pace of updating the student curriculum with the new technologies in university departments that prepare the future specialists in the field of agriculture [19].

The level of complexity of the problems researchers in agriculture-related areas need to address is constantly increasing. The advent of the Internet forces researchers to move their models and applications in new programming platforms. As agriculture occurs in time and space, aside from the technical issues presented by the particular problem, researchers need to take into account spatial and temporary considerations related to the problem. Therefore, researchers in agriculture-related fields are obliged to address more and more complex problems and their solution requires a wide collaboration between specialists of different disciplines and the integration of different technologies. Thus, the software systems they need to develop and maintain are complex and challenging.

In order to successfully overcome the challenges of developing flexible and complex agricultural systems, researchers are required to master and use modern software engineering disciplines. The following is a short inventory of some of the most advanced software engineering techniques used in developing software systems in agriculture and related fields.

The Unified Modeling Language

The Unified Modeling Language (UML) was born as a support for modeling software using the object-oriented programming paradigm. Before UML, several object-oriented modeling languages were used, each with its own set of notations, and there was some confusion among the object-oriented community about which language to use [4]. By the mid-1990s, an important event had occurred that impacted the development of object-oriented modeling languages in a very positive manner. Grady Booch, Ivar Jacobson and James Rumbaugh joined Rational Rose (http://www-306.ibm.com/software/rational/) with the goal of creating a standard modeling language for specifying, visualizing, constructing, and documenting all the artifacts of a software system [4].

According to Wikipedia, in the field of software engineering, the UML is a nonproprietary specification language for object modeling. UML is a general-purpose modeling language that includes a standardized graphical notation used to create an abstract model of a system, referred to as a UML model. UML is extendable, offering the following mechanisms for customization: profiles and stereotype (http://en.wikipedia.org/wiki/Unified_Modeling_Language). The current version, UML2.0, contains 13 types of diagrams that can be grouped in three categories such as structure, behavior, and interaction diagrams; they are used to express static and dynamic aspects of the system under study. UML is the Object Management Group's most-used specification, and the way the world models not only application structure, behavior, and architecture, but also business process and data structure.

Examples of UML Models in Agriculture

The use of UML in modeling agricultural systems is a recent phenomenon. Initially, the use of UML was to make a general presentation of the application's model. Hutchings used a simple class diagram to represent relationships between classes in a framework for grazing livestock. Drouet and Pages [7] discussed the benefits of using the object-oriented paradigm and UML to express the relationships between growth and assimilate partitioning from plant organs to the whole plant. This use of UML was limited as the diagrams presented lacked many details, which make it difficult to understand the role of classes/objects and their behavior. A good model should represent not only the relationships between classes, but also their structure and behavior and the role of each class involved in an association.

Later, a number of authors made UML part of their modeling approach [21] used UML to analyze several irrigation-scheduling models and water-balance models to identify common elements and relationships in order to propose a general template for creating new models and maintaining existing ones. Papajorgji and Pardalos [19] presented a detailed UML and object-oriented approach to model software for agricultural systems. Pinet et al. [22] used UML and Object Constraint Language (OCL) to model spatial constraints of an environmental information system monitoring the spreading of organic matter. Hasenohr and Pinet [12] used UML and OCL to develop a spatial decision support system to implement common agricultural policy. Martin and Vigler [13] used UML to set up a shared geographic information system for agricultural quality and environmental management. Miralles [15] used UML to present geographic information system (GIS) patterns that express relationships between spatial and temporal concepts and to automatically generate the corresponding code. Figure 1 shows the class diagram of the irrigation-scheduling model as presented in [21].

The OCL

OCL is a notational language, a subset of UML that allows specifying constraints over entities representing concepts from the problem domain [17,27]. It integrates notations close to a spoken language to express constraints. OCL was first developed by a group of IBM scientists around 1995 during a business modeling project. It was influenced by Syntropy, which is an object-oriented modeling language that makes heavy use of mathematical concepts. OCL is now part of the UML standard supported by the OMG and it plays a crucial role in the model-driven architecture (MDA) approach [24].

Examples of Using OCL in Agriculture

OCL is used to express spatial constraints in an environmental information system developed by researchers at Cemagref, France, and is described in detail in [22]. This system monitors the spreading of organic matter in France.

Spreading on the croplands is an excellent way of recycling organic matter (manure, sewage sludge, etc.) but the agricultural practices used require a fastidious monitoring system. An excessive and ill-planed spreading practice could lead to damage to soils owing to pollution. It is very important to model a set of spatial constraints that define precisely where spreading of organic matter can take place; as an example, organic matter can never be spread inside certain protected natural areas. Designing an environmental information system that controls the spreading of organic matter requires some spatial constraints be strictly respected. Figure 2 shows the UML model for the spreading problem.

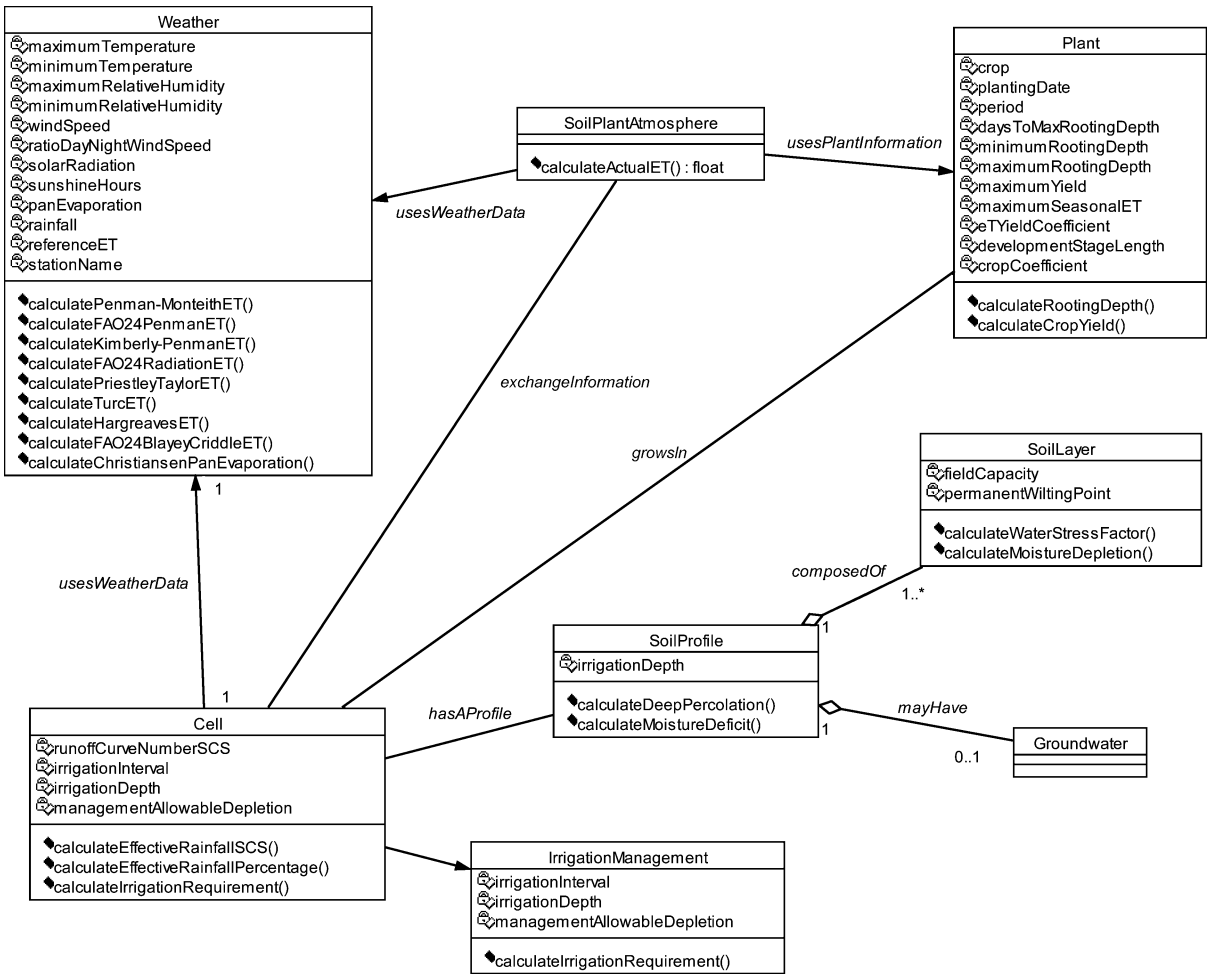
The *Allowed_Area* class models the area on which the regulation allows the spreading of organic matter. The *Spread_Area* class models the area on which the spreading has already been carried out by the farmers. In the ideal case, the organic matter is organized into groups before being spread on the fields; each group has an ID in order to improve traceability. The spreading model presented in this example includes only one of the potential organic matter providers (*Purification_Station*).

Constraints can be expressed using OCL. The following constraint says that a spread area should not overlap with its associated allowed area:

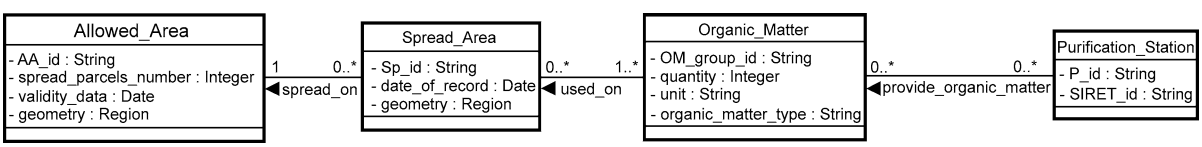
context Spread_Area **inv**:
not (self.geometry.overlap(self.spread_on.geometry))

The following constraint says that all allowed areas must be spatially disjoint from built areas:

context Allowed_Area **inv**:



State of the Art in Modeling Agricultural Systems, Figure 1
Class diagram representation of the irrigation-scheduling model



State of the Art in Modeling Agricultural Systems, Figure 2
Agricultural spreading model

1. **Built_Area.allInstances->forAll**
   (built_area_instance|
2. **built_area_instance.geometry->forAll(building|**
3. **self.geometry.disjoint(building)))**

In the above constraint, the use of set-based operation is needed because the geometry of a built area can be composed of several simple regions (i. e., several buildings). A complete expression in natural language of this constraint is "1. **for each built_area_instance in the *Built_Area* class** and 2. **for each building in the built_area_instance geometry**, 3. **the geometry of an *Allowed_Area* instance (denoted by self) must always be spatially disjoint from building**." Code can be automatically generated from OCL constraints [6,22]. This code allows also the evaluation of the quality of the data stored in the database; i. e., verify if the data

stored in the database satisfy the constraints defined using OCL.

The Design Patterns

Well before software engineers started using patterns, an architect named Christopher Alexander wrote two books that describe the use of patterns in building architecture and urban planning. The first book is titled *A Pattern Language: Towns, Buildings, Construction* [1], published in 1977. The second one is titled *The Timeless Way of Building* [2], published in 1979. These two books not only changed the way structures were built, but they had a significant impact in another not closely related field, the field of software engineering. According to Alexander [1], a pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that one can use this solution a million of times over, without ever doing it the same way twice. Although Alexander refers to buildings and towns, his conclusion can be successfully applied in the process of object-oriented design.

Design patterns are well-thought solutions for a large number of problems that have been built by experienced designers to be easily used by novice programmers. They started being used in the mid-1990s, when a group of four software engineers [10] wrote the book titled *Design Patterns Elements of Reusable Object-Oriented Software*. The book had a significant impact on the way software design was carried out. A design pattern names, abstracts, and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design [10]. The same way an architect uses prefabricated blocks for building complex constructions, a programmer will use patterns to develop complex software. Using patterns makes the process of designing complex systems easier.

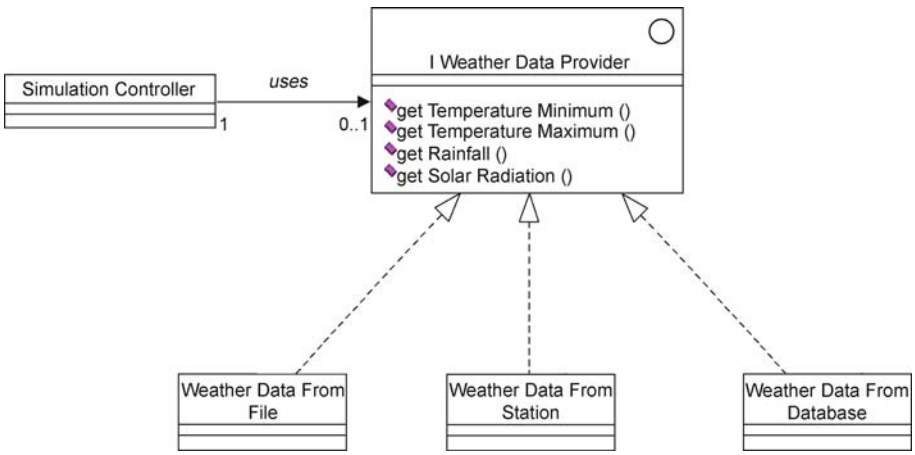
Design patterns are divided into three categories: creational, structural, and behavioral patterns. Creational patterns deal with the process of creating objects. They describe optimal ways of creating new objects. Structural patterns describe how to compose classes or objects. Behavioral patterns describe how to distribute behavior among classes and how classes interact with each other.

Example 1 of Using Design Patterns in Agriculture

Very often programmers have to solve the same problem that occurs in different applications regardless of the problem domain. An example of this type of problem could be providing an application with the same type of data using different data sources and the system has to decide at run time what the particular data source is. Such a problem can be solved using the *strategy pattern*.

The intent for the *strategy pattern* is to define a family of algorithms, encapsulate each one, and make them interchangeable [10]; therefore, algorithms can vary independently from the clients that use them. This pattern is useful in cases where several strategies are available for use and the choice of the right strategy is done at run time. To better understand the context in which the strategy pattern can be used, let us consider a simple simulation model as presented in [20]. In a crop simulation model the weather data can be obtained using different sources, such as using a text file, reading them from a database system, or using an on-line system of weather stations. In a system developed in a traditional programming language such as FORTRAN, the ability to choose between several options requires the use of complex *if-then-else* or *switch* statements. All the options available are hardwired into code. As new data sources are available, their use will require changes to the code. Therefore, traditional programming languages offer rather limited and rigid solutions to this problem. A well-thought system should not only provide access to several sources of data, but additionally it should provide for ways of obtaining them when available in the future without affecting the existing system.

The object-oriented paradigm and the design patterns solve this problem by offering a flexible and elegant solution. Figure 3 shows classes that are involved in the strategy pattern as described in [19]. The *SimulationController* is a client that uses the weather data. The *IWeatherDataProvider* is an interface that represents the common behavior of all classes providing access to a particular source of weather data. *SimulationController* has a unidirectional association with *IWeatherDataProvider*. The multiplicity of this association allows one controller to use one or no weather data provider. Classes *WeatherDataFromFile*, *WeatherDataFromStation*, and *WeatherDataFromDatabase*



State of the Art in Modeling Agricultural Systems, Figure 3
Class diagram for the strategy pattern

provide behavior for extracting data from a particular source such as a text file, a network of weather stations, or a database. These classes implement the same interface, the *IWeatherDataProvider*; therefore, any one of them can be used to provide the weather data requested by *SimulationController*. Note that the user of the data, in this case *SimulationController*, does not have access to or knowledge of the data providers; therefore, the data providers can change the data extraction algorithm without affecting the data user.

Example 2 of Using Design Patterns in Agriculture

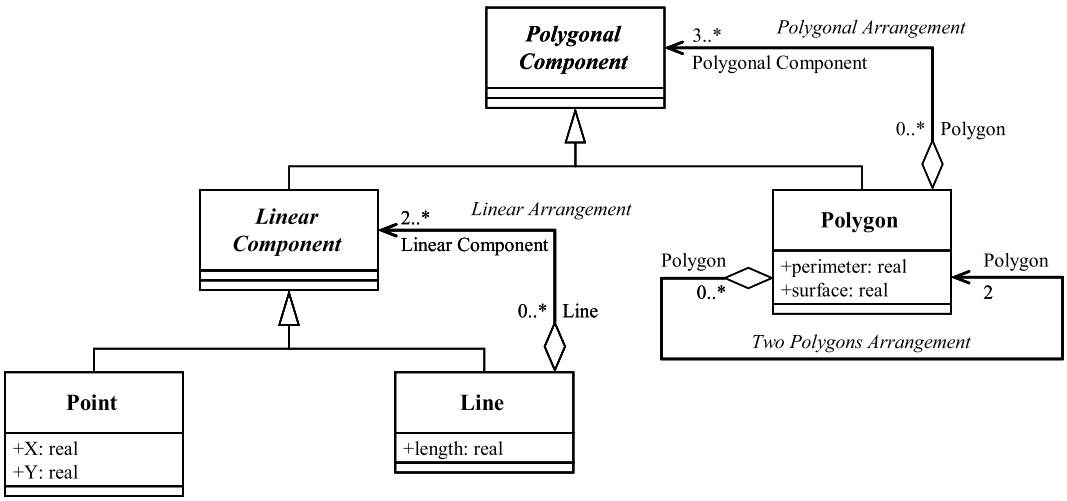
While developing a GIS, designers pay particular attention to the spatial properties of thematic concepts such as *Plot*, *Spread_Area*, etc. In GIS-based systems the spatial concepts that are manipulated the most are *Point*, *Line*, and *Polygon*. These concepts have their own characteristics and some of them can be combined to create other concepts; as an example, a *Line* can be represented as a set of *Point* and a *Polygon* can be considered as a set of *Point* or as a succession of *Line*. Furthermore, the nature of the relationship between *Point*, *Line*, and *Polygon* is static; it never changes over time. The relationship between these spatial concepts can be represented by design patterns as described in [10].

Miralles [15] described a *GIS design pattern based model* that is a recurrent model for a coordinate system: 1D, 2D, and 3D. Figure 4 shows the specific design pattern for a 2D coordinate system. This GIS model is structured around two composite patterns. The first composite pattern depicts the relationship between the spatial properties *Point* and *Line*. A *Line* is composed at least of two *Linear Component*, which can be *Point*, *Line*, or a mixture of *Point* and *Line*. The *Point* has two properties: an abscissa and an ordinate. The *Line* has as property its length. The second composite pattern provides the possibility of representing a *Polygon* as *Polygonal Component*, which can be either *Linear Component* or *Polygon* or even a combination of the latter. The polygonal properties are the perimeter and the surface. Like the simplest polygon (triangle) is composed of three points or three line segments, the cardinality of the *Polygonal Arrangement* association should be at least equal to (3 or more). In this case, the polygonal arrangement of two polygons cannot be done. In order to do it, the *Two Polygons Arrangement* association has been added.

The *GIS design pattern based model* describing the spatial properties currently used for GIS modeling could be considered as a structural pattern. Considering the static nature of the relationships among concepts involved in the pattern, code can be easily generated in any programming language.

The MDA Approach

The MDA is a framework for software development defined by the OMG [25]. At the center of this approach are models; the software development process is driven



State of the Art in Modeling Agricultural Systems, Figure 4
Geographical information system design pattern based model for a 2D coordinate system

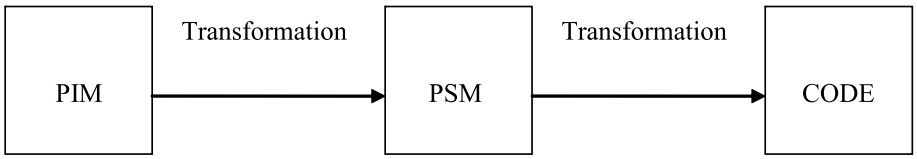
by constructing models representing the software under development. The MDA approach is often referred to as a model-centric approach as it focuses on the business logic rather than on implementation technicalities of the system in a particular programming environment. This separation allows both business knowledge and technology to continue to be developed without necessitating a complete rework of existing systems [14].

MDA uses UML to construct visual representations of models. UML is an industry standard for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system [4], and it has a set of advantages that makes it fit to be the heart of the MDA approach. First, by its nature, UML allows for developing models that are platform-independent. These models depict concepts from the problem domain and the relationships between them and then represent the concepts as objects provided with the appropriate data and behavior. A model specified with UML can be translated into any implementation environment. The valuable business and systems knowledge captured in models can then be leveraged, reused, shared, and implemented in any programming language [5]. A second advantage is that UML has built-in extension mechanisms that allow the creation of specialized, UML-based languages referred to as UML profiles [8]. If modelling agricultural systems requires special modelling artifacts, then an agricultural UML profile would be created and plugged into the UML core system.

The MDA approach consists of three levels of models as shown in Fig. 5. As shown in this figure, a set of transformations are needed to transform a model from the current level to the next one.

The approach starts with construction of a *conceptual diagram* that represents our knowledge of the problem domain expressed through concepts, abstractions, and their relationships. Conceptual diagrams are the result of an activity referred to as *conceptual modeling*. Conceptual modeling can be defined as the process of organizing our knowledge of an application domain into hierarchical rankings or ordering of abstractions, in order to obtain a better understanding of the phenomena under consideration [7]. Conceptual diagrams have the advantage of presenting concepts and relationships in an abstract way, independent of any computing platform or programming language that may be used for their implementation. During this phase, the focus is on depicting the concepts of the system and providing them with the right data and behavior. The fact that the implementation technology may be Java, a relational database, or .NET is irrelevant at this point. Therefore, the intellectual capital invested in the model is not affected by changes in the implementation technologies. A conceptual model thus is a platform-independent model (PIM).

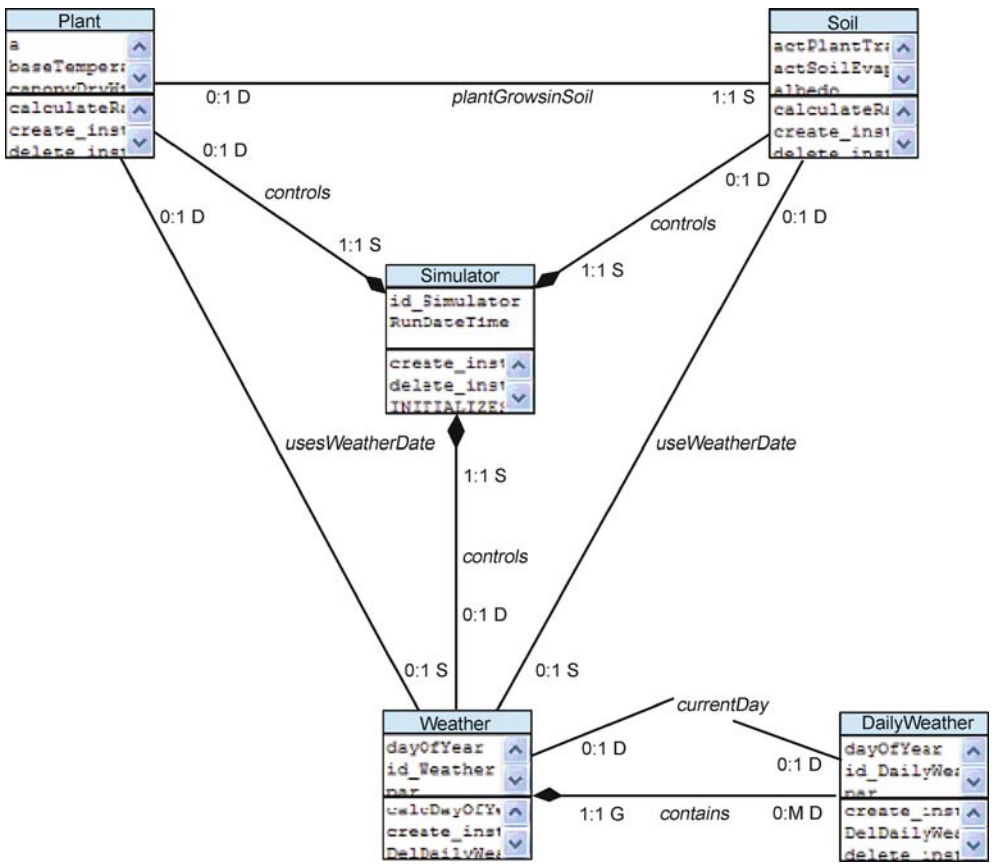
Because of the nature of a PIM (no implementation details are considered at this phase) and because the model construction is done visually using UML,



State of the Art in Modeling Agricultural Systems, Figure 5
Transformations are applied to a model level to obtain the next level. *PSM* platform-specific model

the participation of domain specialists in the model construction process is greatly facilitated. The MDA approach frees domain specialists from the necessity of knowing a programming language in order to be an active participant. PIMs are developed in UML, which is visual and uses plain English that can be easily understood by programmers and nonprogrammers alike [21]. A PIM is the only model that developers will have to create "by hand." Executable models will be obtained automatically by applying a set of transformations to the PIM.

Figure 6 shows a PIM for a simple crop simulation model. Details on the implementation of this model can be found at http://mda.ifas.ufl.edu. Concepts from the simulation domain are depicted in an abstract manner and their relationships are presented. At the center of the model is the *Simulator* object, which has access to entity objects *Plant*, *Soil*, and *Weather*. The na-



State of the Art in Modeling Agricultural Systems, Figure 6
Conceptual model for the crop simulation model

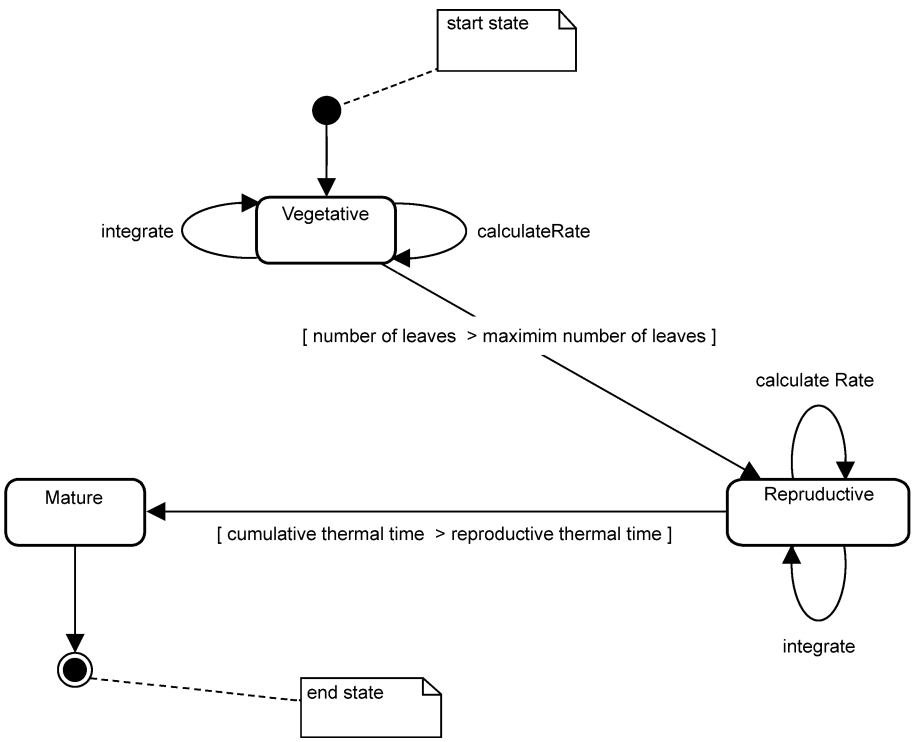
ture of these relationships is a composition, meaning that it is *Simulator*'s responsibility to create instances of these objects and destroy them at the end of the simulation. *Simulator* is provided with a method named *simulate(list of parameters)* that runs the simulation using as initial values the list of parameters. *Simulator* plays the role of a control class; it sends the right message to the right object to carry out the simulation [19].

Providing objects of the conceptual diagram with behavior is one of the most exciting features of the MDA approach. In the world of the simulation models, most of the behavior that objects should provide is expressed in the form of equations. Equations are constructed in a declarative way using attributes of objects participating in the conceptual diagram.

The simulation process is controlled by the behavior of the object *Plant*. A state-transition diagram is used to model the behavior of *Plant* [4]. This diagram shows the valid execution order of the services of the class and the set of possible lifecycles of *Plant*. Figure 7 shows the state-transition diagram of *Plant*. The diagram has two types of elements: *states* and *transitions*. States represent the different situations through which an object of type *Plant* can pass, depending on the value of its attributes. Transitions represent executed services, events, or transactions, which produce state changes and modify the value of the object's attributes.

According to the state-transition diagram, *Plant* will remain in the state *vegetative* and will continue to receive messages *calculateRate* and *integrate* as long as the guard condition *number of leaves* > *maximum number of leaves* is not satisfied. When the guard condition is satisfied *Plant* will move to state *reproductive*. For this transition, the source state is *vegetative* and the target state is *reproductive*. Plant will remain in the state *reproductive* as long as the guard condition *cumulative thermal time* > *reproductive thermal time* is not satisfied. When the guard condition is satisfied, *Plant* will move to state *mature* and the simulation will terminate.

MDA-based tools provide ample capabilities to check the correctness of the conceptual model, the behavior of it objects, and the relationships between them.



State of the Art in Modeling Agricultural Systems, Figure 7
State chart describing the behavior of *Plant*

An XML file is created that contains a detailed specification for the model that can be used by code engines to generate code in several programming languages. Several scenarios can be considered as different parts of the system can be implemented in different languages. For example, the user can choose the C# environment for developing the user interface and a CORBA-EJB, Java-based environment for the implementation of the server. Because the conceptual model is detailed and precise, code generators can find all the information needed to translate the model into several programming environments. Besides the code representing objects of the conceptual model, code generators will provide all the wiring code that links the client and the server applications.

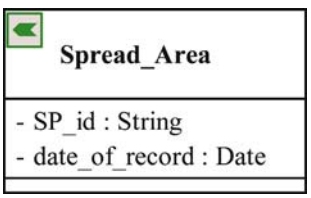
In the domain of GIS there are several modeling formalisms to express the spatial properties of thematic concepts: Aigle [16], CONGOO [18], GeoFrame [9], MADS [23], Perceptory [3], POLLEN [11], etc. Some of these formalisms use a visual language based on pictograms (Perceptory, MADS, Aigle, etc.). The visual language was introduced to improve the communication between the GIS designer and users.

Similarly, Miralles [15] has implemented the visual language of Perceptory in a professional case tool using the profile mechanism, a mechanism that extends the UML metamodel. Pictograms are attached as annotations to the UML notation of *Class*. For example, the polygonal geometry of the thematic concept *Spread_Area* showed in Fig. 8 can be expressed by a polygonal pictogram. Figure 8 shows the UML notation for class *Spread_Area* using the Perceptory language [3]. The polygonal pictogram used in *Spread_Area* shows that *Spread_Area* is kind of *Polygon*.

The GIS design formalisms are used during the analysis phase of the GIS development, the phase during which the model is created "by hand." So, the model built is a PIM. Miralles [15] provided two transformations on the PIM to convert a pictogram into model elements used by code generators. The first one is a transformation that generates the *GIS design pattern based model* presented in "The Design Patterns" (Fig. 4). It automatically creates classes (*Point*, *Line*, and *Polygon*), the corresponding attributes (*X*, *Y*, *length*, *perimeter*, and *surface*), and also the generalization and the association links (*Linear Arrangement*, *Polygonal Arrangement*, and *Two Polygons Arrangement*). This transformation is called *GIS design pattern based model generation*. At this step, the thematic concept *Spread_Area* and the spatial concept *Polygon* are totally disassociated. The second transformation implements the relationship between these two concepts and is also a PIM/PIM transformation. This transformation is referred to as *pictogram translation mapping technique*. The goal of this transformation is to automatically establish an association between *Spread_Area* and *Polygon* (Fig. 9) referred to as *Spatial Characteristic*.

By default, the role of *Polygon* is set to *Geometry* and its cardinality is set to 1. At the other end of the association *Spatial Characteristic*, the class, and its role share the same name, *Spread_Area*, and its cardinality is set to (0 or 1). These default values can be modified later by the designer if necessary. Once the association has been created, the pictogram is not used any longer as the information it conveys becomes redundant.

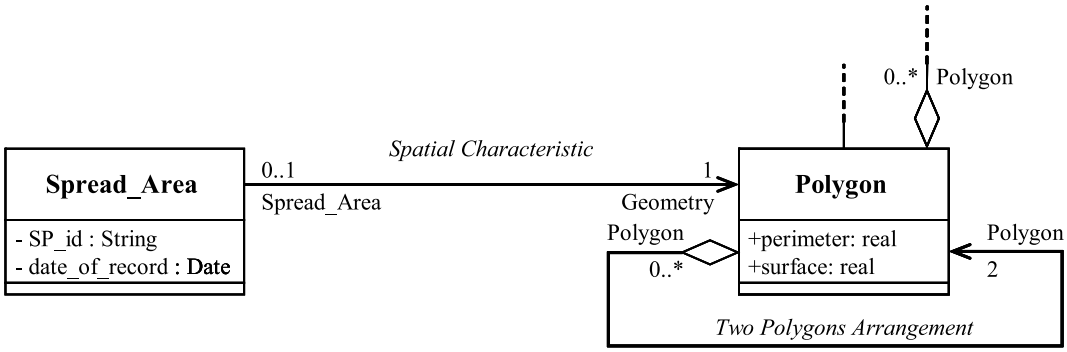
The two transformations described above, GIS design pattern based model generation and pictogram translation mapping technique are examples of application of the MDA principles in the domain of GIS.



State of the Art in Modeling Agricultural Systems, Figure 8
***Spread_Area* concept annotated with a polygonal pictogram**

Conclusion

Developing a successful software project in agriculture requires the collaboration of researchers from different scientific domains with different scientific backgrounds. Therefore, it is very important for a team of different backgrounds to have a common communication language. UML is an excellent tool for analyzing, designing, and documenting software projects. Models are developed visually using plain English (or any other spoken language for that matter) and can be understood by programmers and nonprogrammers alike. Thus, collaboration between team members is greatly



State of the Art in Modeling Agricultural Systems, Figure 9
Association spatial characteristic created by the pictogram translation mapping technique

improved by increasing the number of specialists involved in project development. Furthermore, the advent of MDA makes the process of design and analysis more accessible to specialists, as MDA is a specialist-centric approach. The model is developed visually using knowledge from the problem domain, thus making the specialist of the domain the center of the application development.

References

1. Alexander CA (1977) Pattern Language: Towns, Buildings, Constructions. Oxford University Press, New York
2. Alexander C (1979) The Timeless Way of Building. Oxford University Press, Oxford, UK
3. Bedard Y, Larrivee S, Proulx M-J, Nadeau M (2004) Modeling Geospatial Databases with Plug-ins for Visual Languages: A Pragmatic Approach and the Impacts of 16 Years of Research and Experimentations on Perceptory. ER Workshops CoMoGIS, Shanghai (China), pp 17–30
4. Booch G, Rumbaugh J, Jacobson I (1999) The Unified Modeling Language User Guide. Addison-Wesley, Reading, MA
5. Clark R, Papajorgji P (2005) Model Driven Architecture for Software Generation. World Congress of Computers in Agriculture, Vila Real, Portugal, July 2005
6. Demuth B, Hußmann H, Loecher S, Zschaler S (2004) Structure of the Dresden OCL Toolkit. In: 2nd international Fujaba days "MDA with UML and rule-based object manipulation". Darmstadt, Germany, September 15–17
7. Drouet J-L, Pages L (2001) Object-Oriented modeling of the relationships between growth and assimilates partitioning from the organ to the whole plant. In: Second International Symposium Modelling cropping systems, Firence, Italy, July 16–18, pp 19–20
8. Frankel SD (2003) Model Driven Architecture. Applying MDA to Enterprise Computing. Wiley, Indianapolis
9. Filho JL, Iochpe C (1999) Specifying analysis patterns for geographic databases on the basis of a conceptual framework. In: Proceedings of the 7th ACM international symposium on Advances in geographic information systems. ACM Press, Kansas City, pp 7–13
10. Gamma E, Helm R, Johnson R, Vlissides J (1995) Design Patterns Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading
11. Gayte O, Libourel T, Cheylan J-P, Lardon S (1997) Conception des Systèmes d'information sur environnement. Hermés, Paris
12. Hasenohr P, Pinet F (2006) Modeling of a Spatial DSS Template in Support to the Common Agricultural Policy. J Decision Syst 15(2):181–196
13. Martin C, Vigier F (2003) Setting up a Shared Geographic Information System for Agricultural Quality and Environmental Management at Production Level – Context – Methodology – Concrete Application. Fourth Conference of the European Federation for Information Technology in Agriculture Food and the Environment (EFITA), Budapest, Hungary, July 5–9
14. Miller J, Mukerj J (eds) (2003) MDA Guide Version 1.0.1. Document Number omg/2003–06-01, Object Modeling Group. Object Management Group (OMG). http://omg.org/
15. Miralles A (2006) Ingénierie des modèles pour les applications environmentales. Thèse de doctorat, Université des Sciences et Techniques du Languedoc, Montpellier, France
16. Lbath A (1997) AIGLE: Un environnement pour la conception et la génération automatique d'applications géomatiques. Thèse de doctorat, Institut National des Sciences Appliquées, Lyon
17. OCL 2.0 specification version 2.0. OMG specification, 185p, http://www.omg.org/docs/ptc/05-06-06.pdf. Accessed June 2005
18. Pantazis D, Donnay JP (1996) La conception SIG: Méthode et formalisme. Hermés, Paris, p 343

19. Papajorgji P, Pardalos P (2005) Software Engineering Techniques Applied to Agricultural Systems an Object-Oriented and UML Approach. Springer, New York
20. Papajorgji P (2005) A plug and play approach for developing environmental models. Environ Modelling Softw 20(10):1353–1357
21. Papajorgji P, Shatar T (2004) Using the Unified Modelling Language to develop soil water-balance and irrigation-scheduling models. Environ Modelling Softw 19:451–459
22. Pinet F, Duboisset M, Soulignac V (2007) Using UML and OCL to Maintain the Consistency of Spatial Data in Environmental Information Systems. Environ Modelling Softw 22(8):1217–1220
23. Parent C, Spaccapietra S, Zimanyi E (2006) Conceptual Modeling for traditional and Spatio-Temporal Applications: The MADS Approach. Springer, New York
24. Kleppe A, Warmer J (2003) The Object Constraint Language, getting your models ready for MDA. Addison-Wesley
25. Kleppe A, Warmer J, Bast W (2003) MDA Explained The Model Driven Architecture: Practice and Promise. Addison-Wesley
26. Taivalsaari A (1996) On the notion of inheritance. ACM Comput Surveys 28(3):438–479
27. Warmer J, Kleppe A (1999) The Object Constraint Language, Precise Modeling with UML. Addison-Wesley

Static Resource Constrained Project Scheduling

Arif Volkan Vural, Burak Eksioglu
Department of Industrial and Systems Engineering,
Mississippi State University,
Starkville, USA

Article Outline

Keywords and Phrases
Introduction
Formulation
Methods
Conclusion
References

Keywords and Phrases

Project scheduling; Scheduling; Resource constrained project scheduling; Resource scheduling

Introduction

This review paper considers the resource constrained project-scheduling problem (RCPSP) with static nature and renewable resources and aims to provide a recent survey of related work and heuristics employed. 'Static scheduling' refers to determining a solution to a scheduling problem instance with fixed resources and precedence constraints. 'Renewable resources' implies that resources may be used during the whole scheduling process and planning timespan without degradation in capacity or work pace. Thus, the resources are not single-use type. The solutions will basically consist of starting times of a known set of activities. For an introduction and overview of different formulations of project scheduling problems the reader is referred to [7,16,17,25]. Ozdamar and Ulusoy [30] provide an elaborate review of RCPSP with both renewable and non-renewable resource constraints and time/cost based objectives.

The RCPSP is known to be NP-hard. Thus only instances with a very limited number of activities can currently be solved to optimality. For larger problems, heuristics are utilized that provide robust, high performance, extensible and easy to apply solutions. Some benchmark results are also available in the literature and among those, the most recent are the ones supplied by [1,15,23,24]. Bouleimen and Lecocq [6] (along with [1] and [13]) give the best performing results to supplied benchmark instances.

The activity-on-node (AON) based flow representation of the RCPSP is given in the next section. Following the formulation, a brief summary of the recent approaches proposed is provided.

Formulation

Artigues et al. [2] provide the following AON-flow network based formulation for the static case of RCPSP with renewable resources:

It is assumed that a project composed of a set of activities $V = 1, \ldots, n$ has to be scheduled on a set of renewable resources $\Re = 1, \ldots, m$. Each resource $k \in \Re$ has a finite capacity R_k. Precedence constraints of activities within the project are modeled by a set of project arcs E such that $(i, j) \in E$ means that activity j has to start after completion of activity i. Each activity $i \in V$ requires a non-negative amount r_{ik} of each resource

$k \in \Re$ and has a duration p_i. The scheduling problem lies in characterizing an n-tuple $S = \{S_1, \ldots, S_n\}$ where S_i is the starting time of the activity i, while minimizing the total project duration (makespan) denoted by C_{max}. This problem, known as the RCPSP may be defined by the triple $(V, E, \Re)$. Its difficulty comes from the resource limitation constraints that prevent some activities requiring the same resources from being scheduled simultaneously. These constraints can be modeled by defining each resource k as the union of R_k resource units, such that a given resource unit cannot be allocated to more than one activity at the same time. In other words, each resource is assumed to have a capacity of one activity. If a resource has a capacity greater than one activity, then it is divided into several resources each with a capacity of one. Hence, in any feasible solution, a resource unit allocated to an activity i has to be directly transferred after the completion of i to a unique activity j. However, since all the units of the same resource are equivalent, one only has to know the number of units directly transferred from one activity to another. For an elaborate discussion of the model, the graphical representation, and the mathematical formulation of the problem, refer to [2].

There are two main classifications for the objective function: time based and cost based. Time based objectives in the literature comprise instances such as minimizing makespan, mean lateness, mean completion time, and weighted tardiness in an environment where multiple projects are dealt simultaneously. However, cost based objectives don't necessarily yield the same results as time based objective functions. Maximizing the net present value of a project, minimizing total cost of a project considering all costs including variable costs due to resource consumption and other overhead summed with tardiness costs, and finally maximizing the efficient usage of cash over the project span are some instances of cost based objectives we can observe in the contemporary literature. Ozdamar and Ulusoy [30] study these different objective functions in greater detail.

Brucker et al. [7] introduce a classification scheme and a common notation for the RCPSP. The need for such an effort is to remove the widening gap between the contemporary machine scheduling literature and the RCPSP literature in terms of notation and classification. Indeed, both problems have so many commonalities that one may be converted to the other with ease (both are *NP*-hard in nature). Due to these commonalities, notation and heuristics developed for one can easily be adapted for the other. Brucker et al. [7] also try to form a standard structural base to maintain future research within a coherent literature.

Methods

The surveys by Herroelen et al. [16] and Kolisch and Padman [25] provide detailed descriptions of the characteristics, representations and classification schemes for the solution approaches proposed for the RCPSP. Bouleimen and Lecocq [6] group the suggested solution methods into three, as follows:

1. priority rules ([4,12,21,22,26]);
2. exact methods ([8,10,11,28]); and
3. metaheuristics such as tabu search ([2,3,31]) genetic algorithms ([1,13,20]), and simulated annealing method ([5,6]).

Brucker et al. [7] claim that the first heuristic methods are the priority-rule based scheduling methods. In these methods, the main idea is extending a partially generated schedule by stepwise insertions of new activities either in sequential or parallel order. At each step, a set of feasible nodes for insertion is generated based on starting time constraints, priority constraints, or other resource constraints. The selection of the next activity for insertion (from the decision set) is based on a priority assessment mechanism, usually specific to the problem type and objectives. Brucker et al. [7] emphasize the advantage of priority-based heuristics as being intuitive, easy to implement and fast in computational effort. However, a shortcoming of these methods is that they do not excel with respect to the average deviation from the optimal objective function value. Brucker et al. [7] also point out that recent effort has shifted to exact methods, local search [32] and meta-heuristics. They also provide an overview of computational results by reporting the size of the problems solved and giving details about computational specifications.

Among the exact algorithms proposed for the RCPSP, most promising progress has been attained using the branch-and-bound mechanism. Herroelen et al. [16] highlight seven points in the conclusion of their review on usage of branch-and-bound methods for solving the RCPSP. They comment that those seven

points reveal a number of desirable attributes of an efficient optimal solution procedure for the RCPSP. It is also added that the seven points constitute the very basis of computational efficiency of a method they introduce for the RCPSP. Herroelen et al. [16] and Icmeli et al. [19] list instances of exact applications from the literature while providing an extensive review of solution methods used for different versions of RCPSP.

Kolisch and Padman [25] cluster heuristic approaches for the RCPSP with makespan minimization objective basically to four different solutions methodologies: (1) priority-rule based scheduling, (2) truncated branch-and-bound, (3) disjunctive arc concepts, and (4) meta-heuristic techniques. However, for their efficiency, robustness and improvement potential, heuristic algorithms and meta-heuristic approaches (rather than exact algorithms) will be addressed.

Meta-heuristic techniques for solving combinatorial optimization problems have emerged in recent years. Kolisch and Padman [25] address that all heuristic approaches encode the solution as a list with length equal the number of jobs. The generated list can be mapped into a schedule using priority-based approaches. For a detailed study in encoding schemes one may refer to Kolisch and Hartmann [24].

In the procedure of Sampson and Weiss [32] each element of the generated list is an integer. In their scheduling mechanism, each element starts at the maximum of the completion times of its immediate predecessors plus a specific integer value. This ensures feasibility in the time domain. To prevent excess usage of renewable resources they also add a penalizing mechanism.

Hartmann [14], Leon and Ramamoorthy [27], Naphade et al. [29], Lee and Kim [26], Cho and Kim [9], and Kohlmorgen et al. [20] basically encode the solution as a list of numbers that assigns each task a priority value. By using these priority values within a schedule-generating scheme, one obtains a feasible schedule and the associated objective function value. This encoding has the potential to be applied to meta-heuristics such as simulated annealing, tabu search and genetic algorithms.

Baar et al. [3], Bouleimen and Lecocq [6], Hartmann [13], and Pinson et al. [31] use an 'activity list' where a schedule is generated by scheduling the activities in the order prescribed by the list. Baar et al. [3] use two different neighborhood search mechanisms for a tabu-search procedure. The first one encodes a solution as an activity list which is mapped to a schedule with the serial scheduling scheme. The neighborhood is defined as all activity lists which can be reached by shifting a resource-critical job to a new position. The second neighborhood builds up on the exact solution procedure of Brucker et al. [8]. Essentially, activity pairs are either forced or let to be processed in parallel via the so called 'parallelity relations'. For a fixed parallelity relation a schedule is obtained by forward recursion. Bouleimen and Lecocq [6] use simulated annealing together with a shift operator. Hartmann [13] uses a genetic algorithm with two-point crossover. Pinson et al. [31] propose a tabu-search with pair wise interchange and shift within a neighborhood.

An alternative objective function for the RCPSP is to maximize the net present value of the project. Kolisch and Padman [25] provide a classification of heuristics developed for the RCPSP. They group heuristics into three categories: (1) optimization guided, (2) parameter based, and (3) meta-heuristic approaches.

Icmeli and Erenguc [18] apply a tabu-search procedure to a starting feasible solution generated using a simple, single-pass algorithm. They improve the initial solution over several iterations by moving each activity one time unit early or late from its current completion time without violating the earliest and latest completion time constraints for the activity. They also test the usage of long time memory concept in their algorithm. The computational results are found to be both efficient and close to optimal.

Zhu and Padman [33,34] introduce a notion of replacing single pass complex optimization-based heuristics with a blend of multiple but simple heuristics. They report superior performance of their method to other works employing unique but more complex heuristics. In their initial work [33], they utilize a multi-agent based approach with six simple rules used in random order to exploit changing conditions of the project environment. In their latter work [34], they use distributed computation concepts through the use of an Asynchronous Team (A-team) approach. This approach facilitates cooperation of multiple heuristic algorithms so that together they produce better results than if they were acting alone.

Kolisch and Hartmann [24] provide an updated and extended version of their previous [23] review on solution methods for project scheduling problems. They review existing solution methods under priority based rules, classical metaheuristics, non-standard metaheuristics, and other heuristics. They test different algorithms they picked from the literature on problem sets generated and report average deviation from critical path lower bound.

Conclusion

In this study, the reader is provided with a brief introduction to the RCPSP and supplied with most recent improvements in solution mechanisms based on heuristics. With the potential of providing high quality solutions within reasonable time frames, heuristics and meta-heuristics seem to be one step ahead of the exact algorithms. Thus, this work aims to focus its inquiry domain within the meta-heuristics field. While trying to keep the content brief, the reader is provided with guides to elaborate and most cited references.

References

1. Alcaraz J, Maroto C (2001) A robust genetic algorithm for resource allocation in project scheduling. Ann Oper Res 102:83–109
2. Artigues C, Michelon P, Reusser S (2003) Insertion techniques for static and dynamic resource-constrained project scheduling. Eur J Oper Res 149(2):249–267
3. Baar T, Brucker P, Knust S (1998) Tabu-search algorithms for the resource-constrained project scheduling problems. In: Voss S, Martello S, Osman I, Roucairol C (eds) Metaheuristics: Advances and Trends in Local Search Paradigms for Optimisation. Kluwer, Boston, pp 1–18
4. Boctor FF (1990) Some efficient multi-heuristic procedures for the resource-constrained project scheduling problem. Eur J Oper Res 49(1):3–13
5. Boctor FF (1996) Resource-constrained project scheduling by simulated annealing. Int J Prod Res 34(8):2335–2351
6. Bouleimen K, Lecocq H (2003) A new efficient simulated annealing algorithm for the resource-constrained project management problem and its multiple mode version. Eur J Oper Res 149(2):268–281
7. Brucker P, Drexl A, Möhring R, Neumann K, Pesch E (1999) Resource-constrained project scheduling: Notation, models, and methods. Eur J Oper Res 112(1):3–41
8. Brucker P, Knust S, Schoo A, Thiele O (1998) A branch and bound algorithm for the resource-constrained project scheduling problem. Eur J Oper Res 107:272–288
9. Cho JH, Kim YD (1997) A simulated annealing algorithm for resource constrained project scheduling problems. J Oper Res Soc 48:735–744
10. Christofides N, Alvarez-Valdes R, Tamarit JM (1987) Project scheduling with resource constraints: A branch and bound approach. Eur J Oper Res 29(2):262–273
11. Demeulemeester E, Herroelen W (1992) A branch and bound procedure for the multiple resource constrained project scheduling problem. Manage Sci 38(12):1803–1818
12. Demeulemeester E, Herroelen W (1995) New benchmark results for the resource-constrained project scheduling problem. Manage Sci 43(11):1485–1492
13. Hartmann S (1998) A competitive genetic algorithm for the resource-constrained project scheduling. Naval Res Logist 456:733–750
14. Hartmann S (1999) Project scheduling under limited resources: models, methods, and applications. Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, p 478
15. Hartmann S, Kolisch R (2000) Experimental evaluation of state-of-the-art heuristics for the resource constrained project scheduling problem. Eur J Oper Res 127:394–407
16. Herroelen W, Demeulemeester E, De Reyck B (1998) Resource-constrained project scheduling – A survey of recent developments. Comput Oper Res 25(4):279–302
17. Herroelen W, Demeulemeester E, De Reyck B (1999) A classification scheme for project scheduling. In: Weglarz J (ed) Project Scheduling: Recent Models, Algorithms and Applications. Kluwer, Boston, pp 1–26
18. Icmeli O, Erenguc S (1994) A tabu-search procedure for the resource constrained project scheduling problem with discounted cash flows. Comput Oper Res 21(8):842–853
19. Icmeli O, Erenguc S, Zappe C (1993) Project scheduling problems: A survey. Int J Oper Prod Manage 13(11):80–91
20. Kohlmorgen U, Schmeck H, Haase K (1999) Experiences with fine grained parallel genetic algorithms. Ann Oper Res 90:203–219
21. Kolisch R (1996) Efficient priority rule for the resource constrained project-scheduling problem. J Oper Manage 14:179–192
22. Kolisch R, Drexl A (1996) Adaptive search for solving hard scheduling problems. Naval Res Logist 43:23–40
23. Kolisch R, Hartmann S (1999) Heuristic algorithms for solving the resource constrained project scheduling problem: Classification and computational analysis. In: Weglarz J (ed) Project Scheduling: Recent Models, Algorithms, and Applications. Kluwer, Boston, pp 147–178
24. Kolisch R, Hartmann S (2006) Experimental investigation of heuristics for resource constrained project scheduling: An update. Eur J Oper Res 174(1):23–37
25. Kolisch R, Padman R (2001) An integrated survey of project deterministic scheduling. OMEGA Int J Manage Sci 29(3):249–272

26. Lee JK, Kim YD (1996) Search heuristics for the resource constrained project scheduling problem. J Oper Res Soc 47:678–689
27. Leon VJ, Ramamoorthy B (1995) Strength and adaptability of problem-space based neighborhoods for resource constrained scheduling. OR Spektrum 17(2):173–182
28. Mingozzi A, Maniezzo V, Ricardelli S, Bianco L (1998) An exact algorithm for project scheduling with resources constraints based on a new mathematical formulation. Manage Sci 44:714–729
29. Naphade KS, Wu SD, Storer RH (1997) Problem space search algorithms for resource constrained project scheduling. Ann Oper Res 70:307–326
30. Ozdamar L, Ulusoy G (1995) A survey on the resource-constrained project scheduling problem. IIE Trans 27:574–586
31. Pinson E, Prins C, Ruiller F (1994) Using tabu search for solving the resource constrained project scheduling problem. Proceedings of the 4th International Workshop on Project Management and Scheduling. Leuven, Belgium, pp 102–106
32. Sampson SE, Weiss EN (1993) Local search techniques for the generalized resource constrained project scheduling problem. Naval Res Logist 40:665–675
33. Zhu D, Padman R (1997) A cooperative multi-agent approach to constrained project scheduling. In: Barr RS, Helgason RV, Kennington JL (eds) Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies. Kluwer, Norwell MA, pp 367–381
34. Zhu D, Padman R (1999) A metaheuristic scheduling procedure for resource-constrained projects with cash flows. Naval Res Logist 46:1–18

Static Stochastic Programming Models

ANDRÁS PRÉKOPA
RUTCOR, Rutgers Center for Operations Research, Piscataway, USA

MSC2000: 90C15

Article Outline

Keywords
See also
References

Keywords

Stochastic programming problem; Underlying deterministic problem; Probabilistic constraint; Individual probabilistic constraints; Joint probabilistic constraint; Programming under probabilistic constraint; Recourse model; Hybrid model

Stochastic programming is the science that offers solutions for problems in connection with stochastic systems, where the resulting numerical problem to be solved is a mathematical programming problem. When formulating a stochastic programming problem, in most cases we start from a deterministic mathematical programming problem that we call base problem or underlying deterministic problem. Then, observing that some of the parameters in it are random, we formulate another decision problem, the stochastic programming problem, by taking into account the probability distribution of the random variables involved.

Any stochastic programming problem formulation depends on a decision-observation scheme that tells us in what order decisions and observations follow each other. If this scheme is: decision making on the system design or control variables (usually contained in the decision vector $\mathbf{x}$), observation of the random variables influencing the system performance, then the model is called static. If there is at least one observation of random variables followed by a decision making, then the model is called dynamic. From another point of view, a stochastic programming problem (static or dynamic) may contain reliability provision or allows for the violation of the constraints with some penalty that is included into the objective function. The reliability provision typically manifests itself in the use of probabilistic constraint(s), where we prescribe that the random constraint(s) should hold (when the random variables realize and can be observed) with prescribed probability (probabilities). The first type of stochastic programming model is called probabilistic constrained model while the second type is called recourse model. The two model construction principles can be used simultaneously in a hybrid model. The use of probabilistic constraints is an old statistical decision principle. For example, A. Wald used it in the sequential analysis context [6]. Its combined use with mathematical programming, however, appeared first in [1]. The general form of a static probabilistic constrained model has the form:

$$\begin{cases} \min & h(\mathbf{x}) \\ \text{s.t.} & h_0(\mathbf{x}) \geq 0, \ldots, h_m(\mathbf{x}) \geq 0, \end{cases}$$

where $h, h_1, \ldots, h_m$ are some functions of $\mathbf{x} \in \mathbf{R}^n$,

$$h_0(\mathbf{x}) = \mathsf{P}\left(g_1(\mathbf{x}, \boldsymbol{\xi}) \geq 0, \ldots, g_r(\mathbf{x}, \boldsymbol{\xi}) \geq 0\right) - p,$$

$\boldsymbol{\xi} \in \mathbf{R}^q$ is a random vector, $g_1, \ldots, g_r$ are functions in R^{n+q} and p is a prescribed probability. In practice, p is chosen near 1, e. g., $p = 0.9, 0.95, 0.99$.

In the simplest case $g_i(\mathbf{x}, \boldsymbol{\xi}) = T_i\mathbf{x} - y_i$, $i = 1, \ldots, r$, where T_i is the ith row of an $r \times n$ matrix T and $h, h_1, \ldots, h_m$ are linear functions. This model can be written as

$$\begin{cases} \min & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} \geq \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}, \\ & \mathsf{P}(T\mathbf{x} \geq \boldsymbol{\xi}) \geq p. \end{cases}$$

The probabilistic constraints in the above models are joint constraints. Sometimes instead of $\mathsf{P}(T\mathbf{x} \geq \boldsymbol{\xi}) \geq p$, the individual probabilistic constraints $\mathsf{P}(T_i\mathbf{x} \geq \xi_i) \geq p_i$, $i = 1, \ldots, r$, are used. This was the case in the originating paper [1]. Joint probabilistic constraint was first used by L.B. Miller and H. Wagner [2]. They assumed, however, that in $\mathsf{P}(T\mathbf{x} \geq \boldsymbol{\xi})$ the components of the random vector $\boldsymbol{\xi}$ are stochastically independent. General probabilistic constrained stochastic programming models have been formulated in [3,4].

A related model construction contains maximization of a probability subject to some constraints. Programming under probabilistic constraint and maximizing a probability under constraints have many applications in many engineering (power systems, water resources, telecommunication, engineering structures, etc.) and economic (insurance, finance, economic planning, etc.) problems. For the mathematical theory, solution techniques and applications of these models, see [5].

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Charnes A, Cooper WW, Symonds GH (1958) Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil. Managem Sci 4:183–195
2. Miller LB, Wagner H (1965) Chance-constrained programming with joint constraints. Oper Res 13:930–945

3. Prékopa A (1970) On porbabilistic constrained programming. In: Proc. Princeton Symp. Math. Program., Princeton Univ. Press, Princeton, pp 113–138
4. Prékopa A (1973) Contributions to the theory of stochastic programming. Math Program 4:202–221
5. Prékopa A (1995) Stochastic programming. Kluwer, Dordrecht
6. Wald A (1945) Sequential tests of statistical hypotheses. Ann Math Statist 16:117–186

Static Stochastic Programming Models: Conditional Expectations

ANDRÁS PRÉKOPA
RUTCOR, Rutgers Center for Operations Research, Piscataway, USA

MSC2000: 90C15

Article Outline

Keywords
See also
References

Keywords

Conditional expectation constraint; Integrated probabilistic constraint

Given the constraints $g_i(\mathbf{x}, \boldsymbol{\xi}) \geq 0, i = 1, \ldots, r$, where $\boldsymbol{\xi}$ is a random vector, one way to create part of a stochastic programming problem, based on them, is to introduce the constraints involving conditional expectations:

$$g_i(t) = \mathsf{E}\{g_i(\mathbf{x}, \boldsymbol{\xi}) | g_i(\mathbf{x}, \boldsymbol{\xi}) < 0\} \leq d_i, \quad i = 1, \ldots, r,$$

where d_i, $i = 1, \ldots, r$, are some bounds chosen by ourselves. In the simplest and from the practical point of view most important case $g_i(\mathbf{x}, \boldsymbol{\xi}) = T_i\mathbf{x} - y_i$, where T_i is the ith row of a matrix. In this case the above constraints take the form:

$$g_i(T_i\mathbf{x}) = \mathsf{E}\{\xi_i - T_i\mathbf{x} | \xi_i - T_i\mathbf{x} > 0\} \leq d_i, i = 1, \ldots, r.$$

The practical meaning of these constraints is that violations of the stochastic constraints $T_i\mathbf{x} \geq \xi_i$, $i = 1, \ldots, r$, are allowed but the average magnitude of violation, given that violation occurs, is bounded from above, in each constraint. If, e. g., $T_i\mathbf{x} \geq \xi_i$ means in a diet problem that the meal composition should satisfy the demand for the ith nutrient in a population (where the randomness of the nutrient demand is due to the inhomogeneous nature of the population), then $\mathsf{E}\{\xi_i - T_i\mathbf{x} | \xi_i - T_i\mathbf{x} > 0\}$ is the average unsatisfied demand for nutrient i, among those whose demands are not satisfied. Constraints of this type have been introduced first in [2]; see also [3]. The conditional expectation constraint is closely related to the expected residual lifetime in reliability theory or total remaining life in insurance, these being defined as $g(t) = \mathsf{E}\{\eta - t | \eta - t > 0\}$, in connection with the random lifetime η. It is well-known (see, e. g., [3]) that if η has continuous distribution with logconcave probability distribution function, then $g(t)$ is a decreasing function. Using this fact, we can convert the conditional expectation constraints into linear ones, provided that ξ_i has continuous distribution with logconcave probability distribution function, and $g_i(\mathbf{x}, \mathbf{y}) = T_i\mathbf{x} - y_i$, for every $i = 1, \ldots, r$. The equivalent constraints are:

$$T_i\mathbf{x} \geq g_i^{-1}(d_i), \quad i = 1, \ldots, r.$$

A closely related stochastic programming constraint formulation, based on the stochastic constraints $T_i\mathbf{x} - \xi_i \geq 0$, $i = 1, \ldots, r$, provides us with the following:

$$\mathsf{E}\{\xi_i - T_i\mathbf{x} | \xi_i - T_i\mathbf{x} > 0\}\mathsf{P}(\xi_i - T_i\mathbf{x} > 0) \leq d_i, \quad i = 1, \ldots, r.$$

These are equivalent to

$$l_i(T_i, \mathbf{x}) = \int_{T_i\mathbf{x}}^{\infty} (1 - F_i(z))\, dz \leq d_i, \quad i = 1, \ldots, r,$$

where $F_i(z)$ is the distribution function of the random variable ξ_i, $i = 1, \ldots, r$. The new constraints are called *integrated probabilistic constraints* and have been introduced in [1]. In the above-mentioned diet problem these constraints mean that the average unsatisfied demand is taken in the whole population and is limited form above, in each nutrient.

The advantage of the integrated probabilistic constraints is that the functions l_i, $i = 1, \ldots, r$, are decreasing, regardless of the type of probability distributions

involved, and the constraints are equivalent to

$$T_i \mathbf{x} \geq l_i^{-1}(d_i), \quad i = 1, \ldots, r.$$

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Klein Haneveld WK (1986) Duality in stochastic and dynamic programming. Lecture Notes Economics and Math Systems, vol 274. Springer, Berlin
2. Prékopa A (1973) Contributions to stochastic programming. Math Program 4:202–221
3. Prékopa A (1995) Stochastic programming. Kluwer, Dordrecht

Statistical Classification: Optimization Approaches

William V. Gehrlein
University of Delaware, Newark, USA

MSC2000: 62H30, 90C11

Article Outline

Keywords
See also
References

Keywords

Statistical classification; Mathematical programming

Statistical classification is used when it is of interest to partition a set of subjects or observations into groups or categories, based on observed attributes that are associated with each of the subjects. For example, a lending institution may wish to partition a set of loan applicants into one of the two categories of probable payers and defaulters, based on observed characteristics for the applicants. The characteristics might include: size of the loan requested, total available income, total amount of credit available to the applicant at other sources, number of years with current employer, and others.

To give a formal definition to the problem, there are n subjects to be partitioned into k categories, based on m different observed characteristics. The proper category of classification is assumed to be known for each of these subjects, and X_{ij} denotes the measured value

of characteristic j for subject i. A *classification function* is to be obtained for each category of classification to represent the strength of association of any subject with that category. Let $F_a(X_{i\cdot})$ denote the classification function for a given category a with $1 \le a \le k$, for any given subject, say for subject i. We assume a linear form for $F_a(X_{i\cdot})$ and

$$F_a(X_{i\cdot}) = c_0^a + \sum_{j=1}^{m} c_j^a X_{ij}.$$

The n known observations are used as a training set to obtain coefficients for the classification functions that accurately model the relationship between the strength of association of the classification functions and the actual group membership for the given observations. These functions are then used in the future to classify subjects for which the proper category is not known, and for which a prediction of category membership is being sought. In particular, a future observation, with associated measured values for X_{ij}s, will be predicted to be a member of category a when $F_a(X_{i\cdot}) > F_b(X_{i\cdot})$, for all $1 \le b \le k$ with $b \ne a$. R.A. Fisher [1] and C.A.B. Smith [3] developed classical statistical techniques to approach this problem, with standard assumptions about the distributions of the X_{ij}s.

More recently, optimization approaches have been used to develop techniques to obtain coefficients for the classification functions that directly maximize the number of correct classifications in the training set. A. Stam [4] presents an exhaustive survey of most of the early work in this area. Most of these approaches are based on mathematical programming techniques. These approaches are of interest since they will maximize the number of correct classifications in the training set, which standard statistical approaches will not necessar- ily do. In addition, the mathematical programming techniques are very useful when standard statistical assumptions about the distributions of X_{ij}s are not valid.

W.V. Gehrlein [2] presents elementary mathematical programming formulations of the generalized classification problem to obtain classification functions that directly maximize the number of correct classifications in the training set. The primary variables in these formulations are the given c_i^a coefficients that should be used in the classification functions.

Each of the observations in the training set will have a binary (0–1) variable, I_i, associated with it, such that observation i will be correctly categorized when $I_i = 0$ and observation i will be incorrectly categorized when $I_i = 1$. The objective function is given by

$$\text{Minimize} \quad \sum_{i=1}^{n} I_i.$$

There are $k - 1$ constraints that are associated with the categorization of each observation. Observation i is known to be a member of some category, say a. Then for each b with $1 \le b \le k$ and $b \ne a$ there is a constraint of the form

$$c_0^a + \sum_{j=1}^{m} c_j^a X_{ij} - c_0^b - \sum_{j=1}^{m} c_j^b X_{ij} + MI_i \ge e,$$

in which M is a very large number and e is a very small number. The values of M and e remain the same in all constraints. By the nature of M and e, this constraint will be met trivially if $I_i = 1$, and we must have $F_a(X_{i\cdot}) > F_b(X_{i\cdot})$ if $I_i = 0$.

It is also possible to develop a classification procedure that has only one classification function. The category of group membership is then determined by where the value of the computed classification function value falls on the number line. That is, the number line is partitioned into k segments, with each of the segments corresponding to an associated group membership. The line segments are established for each group, say a, with an upper limit UL_a and a lower limit LL_a. As above, a binary variable, I_i, is associated with each observation such that observation i is correctly categorized when $I_i = 0$ and observation i will be incorrectly categorized when $I_i = 1$. The objective function remains the same as above. With this formulation there are only two constraints that are associated with the categorization of each observation. Observation i is known to be a member of some category, say a, and the associated constraints are of the form

$$c_0^a + \sum_{j=1}^{m} c_j^a X_{ij} - MI_i \le \text{UL}_a,$$

$$c_0^a + \sum_{j=1}^{m} c_j^a X_{ij} + MI_i \ge \text{LL}_a.$$

As above, M is a very large number, and e will be a very small number. These constraints will be met trivially if

$I_i = 1$, and we must have $\mathrm{LL}_a \le F_a(X_{i\cdot}) \le \mathrm{UL}_a$ whenever $I_i = 0$. Additional constraints are needed to ensure a logical consistency of the line segment partition. For each a with $1 \le a \le k$, we need a constraint of the form

$$\mathrm{UL}_a - \mathrm{LL}_a \ge e.$$

To be certain that there is no overlap of the line segments, each combination of groups, say a and b, has two binary variables I_{ab} and I_{ba} defined for them, with associated constraints of the form

$$\begin{aligned} \mathrm{LL}_a - \mathrm{UL}_b + MI_{ab} &\ge e, \\ \mathrm{LL}_b - \mathrm{UL}_a + MI_{ba} &\ge e, \\ I_{ab} + I_{ba} &= 1. \end{aligned}$$

Extensions of these elementary formulations of optimization techniques go in several different directions. A particularly useful variation deals with the notion of minimizing the total cost of misclassification, when there are different costs or penalties that are associated with the different ways in which a subject could be incorrectly classified. Multiple stage classification schemes are also considered, in which a subject is initially either placed in a category with existing information, or no classification is made. If a classification is not made in the first stage, then additional information is used to make a classification in a second stage. In addition, specialized heuristic techniques have been developed to obtain solutions to these mathematical programming formulations in an efficient manner, when the number of possible categories of classification is restricted. Much of the current work on these extensions is given in [5].

See also

- ► Linear Programming Models for Classification
- ► Mixed Integer Classification Problems
- ► Optimization in Boolean Classification Problems
- ► Optimization in Classifying Text Documents

References

1. Fisher RA (1936) The use of multiple measurements in taxonomic problems. Ann Eugenics 7:179–188
2. Gehrlein WV (1986) General mathematical programming formulations for the statistical classification problem. Oper Res Lett 5:299–304
3. Smith CAB (1947) Some examples of discrimination. Ann Eugenics 13:272–282
4. Stam A (1997) Nontraditional approaches to statistical classification: Some perspectives on the Lp-norm method. Ann Oper Res 74:1–36
5. (1997) Nontraditional approaches to the statistical classification an regression problems. Ann Oper Res 74

Statistical Convergence and Turnpike Theory

Musa Mammadov (Mamedov)
Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Ballarat, Australia

MSC2000: 40A05, 49J24

Article Outline

Keywords and Phrases
Introduction
Turnpike Theory
Statistical Cluster Points and Statistical Convergence
Problem 1
Problem 2
A Challenging Problem
See also
References

Keywords and Phrases

Turnpike; Statistical convergence; Asymptotical stability; Optimal trajectories

Introduction

This article considers the application of the notion of statistical convergence in turnpike theory. The first results have been obtained recently [14,15,19]. We briefly discuss the importance of this conjunction, present some results obtained and, finally, we formulate a challenging problem for future investigations.

We will consider discrete dynamical systems. Trajectories of these systems are some sequences of real numbers. Turnpike property, in a simple case, states that there is a certain stationary point that attracts all optimal trajectories not depending on the initial state. In other words, all optimal trajectories converge to this

stationary point. We can say that any optimal trajectory spends "almost" all time in some ε-neighborhood of that point. The term "almost" in this case means that only a finite number of elements of optimal trajectory may remain outside the ε-neighborhood of that point (ε is any small number).

It turns out that, for some practical problems this property does not hold. The relaxation of the term "almost" might be helpful to extend the class of problems that the turnpike property holds. One such a relaxation is the use of statistical convergence instead of an ordinary convergence. In this case, an infinite number of elements of optimal trajectory may remain outside the ε-neighborhood of a stationary point; however, the number of these elements in comparison with the number of elements in the ε-neighborhood is so small that we can say the optimal trajectory "almost" remains in this neighborhood.

This article adopts the notion of statistical convergence to describe the turnpike property.

Turnpike Theory

Turnpike theory studies asymptotical behavior (often stability) of optimal trajectories of dynamical systems. It has many applications in economics and engineering. We refer to [8,16,18,25] for more detailed information about this theory and its various applications.

The first result in this area was obtained by J. von Neumann, in 1945. However, the main meaning of this result that led to turnpike property was discovered by Paul A. Samuelson, in 1948–1949, who also introduced this terminology. A clearer description of this property was provided by Dorfman et al. [3] in the chapter "Efficient Programs of Capital Accumulation" of *Linear Programming and Economic Analysis*. The following is the famous quote from [3], p. 331, that describes the meaning of the turnpike property:

"Thus in this unexpected way, we have found a real normative significance for steady growth – not steady growth in general, but maximal von Neumann growth. It is, in a sense, the single most effective way for the system to grow, so that if we are planning long-run growth, no matter where we start and where we desire to end up it will pay in the intermediate stages to get into a growth phase of this kind. It is exactly like a turnpike paralleled by a network of minor roads. There is a fastest route between any two points; and if the origin and destination are close together and far from the turnpike, the best route may not touch the turnpike. But if origin and destination are far enough apart, it will always pay to get on to the turnpike and cover distance at the best rate of travel, even if this means adding a little mileage at either end. The best intermediate capital configuration is one which will grow most rapidly, even if it is not the desired one, it is temporarily optimal".

After this book, theorems about the asymptotic behavior of optimal (or efficient) trajectories of dynamical systems are called "turnpike theorems." Asymptotic behavior of optimal trajectories may be described in different ways.

In this article, we consider trajectories that are sequences of numbers from $\mathbb{R}^m$. The turnpike property in this case can be formulated as a convergence of optimal trajectories to some stationary point. We reformulate this property using statistical convergence instead of an ordinary convergence.

Statistical Cluster Points and Statistical Convergence

The idea of statistical convergence was introduced by Steinhaus [23] and also independently by Fast [4] and Buck [1] for sequences of real and complex numbers. Later, this notion was developed by Salat [22], Maddox [7], Connor [2], Fridy [5,6] and others.

Fridy [6] introduced the notion of a statistical limit point and a statistical cluster point and gave some properties of a set of statistical limit and cluster points. In particular, it was shown that the set of statistical cluster points of a bounded sequence is not empty; moreover, if this set consists of one point, then the sequence is statistically convergent to this point. Because of this property, the notion of statistical cluster points, and, consequently, the statistical convergence, became a suitable tool that could be used in turnpike theory.

First we present some notations. We denote by $|A|$ the cardinality of a subset $A \subset \{1, 2, \dots\}$. Consider a sequence (x_k), where $x_k \in \mathbb{R}^m$, $k = 1, 2, \dots$

Definition 1 The sequence (x_k) is said to be statistically convergent to $x^* \in \mathbb{R}^m$ if for every $\varepsilon > 0$

$$\limsup_{n\to\infty} \frac{1}{n} |\{k \le n : ||x_k - x^*|| \ge \varepsilon\}| = 0 .$$

We use the notation $\mathrm{st}-\lim_{k\to\infty} x_k = x^*$ in this case.

Definition 2 $\xi \in \mathbb{R}^m$ is said to be a cluster point of sequence (x_k) if for every $\varepsilon > 0$

$$\limsup_{n\to\infty} \frac{1}{n}|\{k \le n : ||x_k - \xi|| < \varepsilon\}| > 0 \ .$$

Given a sequence $x = (x_k)$, we denote by Γ_x the set of all statistical cluster points. If this set consists of one point, then the sequence is statistically convergent to that point [6].

The set of ordinary limit points is defined as

$$L_x = \{\xi \in \mathbb{R}^m : \text{ there exists a subsequence } x_{k_n} \to \xi \text{ as } k_n \to \infty\} \ .$$

It is clear that if x is a bounded sequence, then L_x is a nonempty compact set and for every $\varepsilon > 0$ there exists a number $N_\varepsilon < +\infty$ such that

$$\rho(L_x, x_k) < \varepsilon \text{ for every } k \ge N_\varepsilon \ .$$

Here $\rho(A, \xi) = \min_{y\in A} ||y - \xi||$ is the distance from ξ to the closed set A.

It turns out that the set Γ_x possesses a similar property. The following is a very useful and important result proved in [6] for the case $m = 1$. It is not difficult to generalize it for $m > 1$.

Lemma 1. *Assume that $x = (x_k)$ is a bounded sequence. Then:*

1. *There exists a sequence $y = (y_k)$ such that*
   - $\Gamma_x = L_x$,
   - $\lim_{n\to\infty} \frac{1}{n}|\{k \le n : x_k \ne y_k\}| = 0$.
2. *The set of statistical cluster points Γ_x is not empty and compact.*
3. $\lim_{n\to\infty} \frac{1}{n}|\{k \le n : \rho(\Gamma_x, x_k) < \varepsilon\}| = 0$ *for all* $\varepsilon > 0$.

Let $\alpha = (\alpha_k)$ be a sequence of bounded real numbers and Γ_α be the set of statistical cluster points on this sequence. From Lemma 1. we know that the set Γ_α has a minimal element. We denote by $C - \liminf_{k\to\infty} \alpha_k$ the minimal element in Γ_α.

This notation is similar to the notion of $\liminf_{k\to\infty} \alpha_k$ being equal to a minimal number of the set of ordinary limit points L_α.

Let $g : \mathbb{R}^m \to \mathbb{R}$ be a continuous function and $x = (x_k)$, $x_k \in \mathbb{R}^m$, be a given bounded sequence. Then the sequence of real numbers $\gamma = (g(x_k))$ is bounded. We define the following functional

$$J(x) = C - \liminf_{k\to\infty} g(x_k) \doteq \min \Gamma_\gamma \tag{1}$$

as a minimal number of Γ_γ. We have the following useful representation: given any bounded sequence $x = (x_k)$

$$C - \liminf_{k\to\infty} g(x_k) = \min_{\xi\in\Gamma_x} g(\xi) \ . \tag{2}$$

Below we consider two problems where the turnpike theorems are formulated in terms of statistical convergence. For details see [14,15,19].

Problem 1

Consider the problem

$$x_{k+1} \in a(x_k) + x_k, k = 1, 2, \dots \ ; \tag{3}$$

$$J(x) = C - \liminf_{k\to\infty} g(x_k) \to \max \ . \tag{4}$$

We assume that set-valued mapping $a\colon \mathbb{R}^m \to \Pi_c(\mathbb{R}^m)$ is continuous in the Hausdorff metric and $g\colon \mathbb{R}^m \to \mathbb{R}$ is a continuous function. Here $\Pi_c(\mathbb{R}^m)$ stands for the set of all compact subsets of $\mathbb{R}^m$.

A sequence $x = (x_k)$ satisfying (3) will be called a trajectory of this system. From Lemma 1. we know that the functional (4) is well defined for bounded trajectories.

Definition 3 $\xi \in \mathbb{R}^m$ is called a stationary point if $0 \in a(\xi)$.

Note that if ξ is a stationary point, then the sequence (x_k), where $x_k = \xi$ for all $k = 1, 2, \dots$, is a stationary trajectory to system (3). Throughout this article, we denote the set of stationary points by $\mathcal{M}$:

$$\mathcal{M} = \{x : 0 \in a(x)\} \ .$$

The set $\mathcal{M}$ may be empty or unbounded. If it is not empty, then it is a closed set as mapping a is continuous. Denote

$$J^* = \sup_{\xi\in\mathcal{M}} g(\xi) \ .$$

Definition 4 Trajectory $x = (x_k)$ is called optimal if $J(x) \ge J(\tilde{x})$ holds for all trajectories $\tilde{x}$ starting from the same initial state: $\tilde{x}_0 = x_0$.

We use the notation $a(A) = \cup_{x\in A} a(x)$. The following is the main condition imposed on mapping a.

Condition A: *Given any set* $A \subset \mathbb{R}^m$

$$\text{if } 0 \in \text{co } a(A) \text{ then } 0 \in a(\text{co } A)\,. \tag{5}$$

If mapping a has a convex graph then this condition holds. The main results are presented in the following two theorems.

Theorem 1 *Assume that Condition A holds and g is concave. Then for every bounded trajectory $x = (x_k)$ to system (1) the inequality $J(x(\cdot)) \le J^*$ is satisfied.*

Now assume that the set $\mathcal{M}$ is convex and bounded, and function g is strictly concave. Then there is a unique point ξ^* such that $g(\xi^*) = J^*$.

Condition B: *The set* $a(\xi^*)$ is a strictly convex body; that is, $\int a(\xi^*) \neq \emptyset$, and for every two points $\xi_1, \xi_2 \in \partial a(\xi^*)$ and for all $\lambda \in (0,1)$ the following holds

$$\lambda\xi_1 + (1-\lambda)\xi_2 \in \int a(\xi^*)\,.$$

Here $\partial(\cdot)$ and $\int(\cdot)$ stand for the boundary and the interior of a set, respectively.

Theorem 2 *Assume that $\mathcal{M}$ is convex and bounded, function g is strictly concave and Conditions A and B hold. If a bounded trajectory $x = (x_k)$ such that $J(x) = J^*$, then $\Gamma_x = \{\xi^*\}$; that is, trajectory x is statistically convergent to $\xi^* : st - \lim_{k\to\infty} x_k = \xi^*$.*

If $J(x) = J^*$ then from the first theorem it follows that trajectory $x = (x_k)$ is optimal. The second theorem provides the turnpike property: all optimal trajectories satisfying $J(x) = J^*$ statistically converge to ξ^*.

The proof of these theorems based on techniques developed for continuous systems in [9,10,11,12]. These studies did not use an assumption similar to Condition B. The following example shows that Condition B is necessary when dealing with discrete systems.

Example 1. Let mapping a and function g be defined on the box given by $\{(x_1, x_2) : |x_i| \le 1, i = 1, 2\}$ as follows

$$a(x_1, x_2) = \{(y_1, y_2) : y_1 = x_2(x_2 - 1),$$
$$y_2 = [-2x_2, 1 - 2x_2]\}\,;$$
$$g = -x_1^2 - (1 - x_2)^2\,.$$

We have

$$\mathcal{M} = \{(x_1, x_2) : |x_1| \le 1, x_2 = 0\}\,;$$

$$J^* = \max_{\xi\in\mathcal{M}} g(\xi) = g(0,0) = -1\,, \quad \xi^* = (0,0)\,.$$

It is not difficult to see that all the conditions of Theorem 2 hold except Condition B. Consider the sequence $x = (x^k)$ where $x^k = (0,0)$ for $k = 1, 3, 5, \ldots$, and $x^k = (0,1)$ for $k = 2, 4, 6, \ldots$ It is a trajectory to (3) because $(0,1) \in a(0,0)$ and $(0,-1) \in a(0,1)$. Moreover, the set $\Gamma_x = \{(0,0), (0,1)\}$ consists of two points. We have

$$J(x) = \min_{\xi\in\Gamma_x} = -1 = J^*\,,$$

however x^k is not statistically convergent to $\xi^* : \text{st-lim}_{k\to\infty} x^k \neq \xi^*$.

Problem 2

Consider the problem

$$x_{k+1} = f(x_k, u_k), x_1 = \xi^0, u_k \in U\,; \tag{6}$$

$$J(x) = C - \liminf_{k\to\infty} g(x_k) \to \max\,. \tag{7}$$

Here ξ^0 is a fixed initial point, function $f(x,u) : \mathbb{R}^m \times \mathbb{R}^r \to \mathbb{R}^m$ is continuous, $U \subset \mathbb{R}^r$ is a compact set and $g : \mathbb{R}^m \to \mathbb{R}$ is a continuous function.

The pair (u, x) is called a process if the sequences $x = (x_k)$ and $u = (u_k)$ satisfy (6) for all $k = 1, 2, \ldots$; $x = (x_k)$ is called a trajectory and $u = (u_k)$ is called a control.

We assume that there is a bounded closed set $C \subset \mathbb{R}^m$ such that $x_k \in C$ for all trajectories; that is, we assume that trajectories are uniformly bounded.

Definition 5 $\xi \in \mathbb{R}^m$ is called a stationary point if there exists $u \in U$ such that $f(\xi, u) = \xi$.

We denote the set of stationary points by M. It is clear that M is a closed set.

We formulate the main conditions as follows:

Condition 1. Function g has a unique maximizer on set M denoted by $\xi^* : \max_{\xi\in M} g(\xi) = g(\xi^*)$.

Condition 2. There exists a process (u^*, x^*) such that $x_k^* \to \xi^*$ as $k \to \infty$.

Denote $B = \{\xi \in C : g(\xi) \ge g(\xi^*)\}$.

Condition 3. There exists a vector $p \in \mathbb{R}^m$ such that $p\, f(x,u) < p\, x$ for all $x \in B$, $x \neq \xi^*$ and $u \in U$.

The turnpike property is formulated in the following theorem.

Theorem 3 *Let Conditions 1–3 hold and (u, x) be an optimal process in problem (6) and (7). Then*

1. $\Gamma_x = \{\xi^*\}$; *that is,* $st - \lim_{k\to\infty} x_k = \xi^*$.
2. *If* $u^* \in U$ *is a unique point in* U *such that* $f(\xi^*, u^*) = \xi^*$, *then we also have* $st - \lim_{k\to\infty} u_k = u^*$.

In this theorem, turnpike property is established not only for optimal trajectories but also for optimal controls. In both cases this property is satisfied in terms of statistical convergence.

A Challenging Problem

The functional in the above problems is defined by statistical cluster points (see (4)). The following functional would be of great interest in terms of turnpike theory and statistical convergence

$$J(x) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} g(x_k) \to \max \ . \tag{8}$$

In the literature on turnpike theory many functional have been considered, including terminal functionals, integral (summation) functionals with and without discount factors [3,8,9,10,11,12,13,16,18,20,21,24,25]. They usually are defined by utility functions (g in our case).

The functional (8) also has a useful meaning: it aims to maximize the limit of average utilities. However, this functional is not considered in the literature; the reason is very simple – for functional (8) the turnpike property in terms of (ordinary) convergence is in order not true!

We explain how this may happen in the following example.

Example 2. Consider the system $x_{k+1} \in a(x_k)$, $k = 1, 2, \ldots$, where $x_k \in (-\infty, +\infty)$. We only require that sets $a(0)$ and $a(1)$ contain at least points 0 and 1: $\{0,1\} \in a(0), \{0,1\} \in a(1)$. Function g is defined as $g(x) = -x^2$.

It is clear that a stationary trajectory $\xi_k^* = 0$, $k = 1, 2, \ldots$, is an optimal trajectory and $J^* = 0$. For any other trajectory $\tilde{x} = (\tilde{x}_k)$ we have $J(\tilde{x}) \le J^*$.

Consider a sequence $x = (x_k)$, where $x_k = 1$ for all $k = i^2$, $i = 1, 2, \ldots$, and $x_k = 0$ otherwise. We know that

$$\lim_{k\to\infty} x_k \ \text{ does not exist; however, } \ st - \lim_{k\to\infty} x_k = 0 \ .$$

It is easy to see that this sequence is a trajectory to the system. Moreover, it is not difficult to show that

$$\frac{1}{n} \sum_{k=1}^{n} g(x_k) \to 0 \quad \text{as} \quad n \to \infty \ .$$

Therefore, x is an optimal trajectory and it does not converge to 0; meanwhile, the statistical convergence to 0 is valid.

This example shows that the turnpike property for functional (8) should use something different from ordinary convergence. We believe that the statistical convergence will be suitable for this aim.

To prove the turnpike property, in terms of statistical convergence, for a wide range of systems with functional (8) would be a challenging problem.

See also

► Turnpike Theory: Stability of Optimal Trajectories

References

1. Buck RC (1953) Generalized asymptotic density. Am J Math 75:335–346
2. Connor JS (1988) The statistical and strong *p*-Cesaro convergence of sequences. Analysis 8:47–63
3. Dorfman R, Samuelson PA, Solow RM (1958) Linear Programming and Economic Analysis. McGraw Hill, New York
4. Fast H (1951) Sur la convergence statistique. Colloq Math 2:241–244
5. Fridy JA (1985) On statistical convergence. Analysis 5:301–313
6. Fridy JA (1993) Statistical limit points. Proc Am Math Soc 4:1187–1192
7. Maddox IJ (1988) Statistical convergence in a locally convex sequence space. Math Proc Camp Phil Soc 104: 141–145
8. Makarov VL, Rubinov AM (1977) Mathematical Theory of Economic Dynamics and Equilibria. Springer, New York
9. Mamedov MA (1985) Asymptotical optimal paths in models with environment pollution being taken into account. Optimization, Novosibirsk 36(53):101–112
10. Mamedov MA (1992) Turnpike theorems in continuous systems with integral functionals. Russ Acad Sci Dokl Math 45(2):432–435

11. Mamedov MA (1993) Turnpike theorems for integral functionals. Russ Acad Sci Dokl Math 46(1):174–177
12. Mamedov MA (2003) Turnpike Theorem for Continuous-time Control Systems when Optimal Stationary Point is not unique. Abstr Appl Anal 11:631–650
13. Mamedov MA, Borisov KY (1988) A simple model of economic grow and pollution control. Vestn Leningr Univ, 5(5):120–124
14. Mamedov MA, Pehlivan S (2000) Statistical convergence of optimal paths. Math Jpn 52(1):51–55
15. Mamedov MA, Pehlivan S (2001) Statistical cluster points and turnpike theorem in nonconvex problems. J Math Anal Appl 256:686–693
16. McKenzie LW (1976) Turnpike theory. Econometrica 44:841–866
17. von Neumann J (1945–1946) A Model of General Economic Equilibrium. Rev Econ Stud 13:1–9
18. Panasyuk AI, Panasyuk VI (1986) Asymptotic turnpike optimization of control systems. Nauka Techn
19. Pehlivan S, Mamedov MA (2000) Statistical cluster points and turnpike. Optimization 48:93–106
20. Rockafellar RT (1973) Saddle points of Hamiltonian systems in convex problems of Lagrange. J Optim Theory Appl 12:367–390
21. Rockafellar RT (1976) Saddle points of Hamiltonian systems in convex problems having a nonzero discount rate. J Econ Theory 12:71–113
22. Śalăt T (1980) On statistically convergent sequences of real numbers. Math Slovaca 30:139–150
23. Steinhaus H (1951) Sur la convergence ordinarie et la convergence asymptotique. Colloq Math 2:73–74
24. Scheinkman JA (1976) On optimal steady states of *n*-sector growth models when utility is discounted. J Econ Theory 12:11–30
25. Zaslavski A (2005) Turnpike Properties in the Calculus of Variations and Optimal Control. Series: Nonconvex Optimization and Its Applications, vol 80 XXII. Springer, New York, p 396

Steiner Ratio of Biomolecular Structures

RUBEM P. MONDAINI
BIOMAT Institute for Advanced Studies of Biosystems, Federal University of Rio de Janeiro, Centre of Technology/COPPE, Rio de Janeiro, Brazil

MSC2000: 90C27

Article Outline

Introduction
The Steiner Ratio of a Metric Manifold
Evenly Spaced Consecutive Points – Spanning and Steiner Trees
Steiner Trees
The Steiner Ratio Function
Concluding Remarks
References

Introduction

The careful observation of protein data banks [6] has been one of the motivations for modelling the biomolecular structure. The present development of this subject led to the conviction that the placement of some atoms in the structure of a protein is the same as that of Steiner points of a minimal Steiner Tree [3,4]. The experimental internal radius of a DNA molecule and a molecular aggregate like the tobacco mosaic virus as well as the pitch of the helices in a helical model of the placement of their atoms have also been in good agreement with this Steiner modelling. It seems that there is a deep correlation between the potential energy of the molecular configuration and the length of the Steiner Tree. The search for the minima of the energy could then be conducted by solving the associated Steiner problem. Even molecular clusters can be studied with this approach by starting from an existing correlation of their potential energies with the length of a generic Fermat problem. It can be thought that Nature is following mathematical principles of local energy minimization in order to build the present form of these structures and to keep them looking for stability through unstable stages of molecular evolution [5].

The Steiner Ratio of a Metric Manifold

We consider a finite set of points A in a metric manifold M. Let us consider the subsets of A such that each pair of points on them could be connected by an edge of minimal length of a subset. These edges are geodesic arcs of the manifold M. A tree is a collection of points and their connecting edges. A tree that connects all the points of a subset is a spanning tree (SP) of this subset. Among all the possible STs s of a set A with length $l_{\mathrm{SP}}(s, A)$, there is at least one whose overall length is minimum. This will be the minimal SP of

set A, MST(A). Its length will be given by

$$l_{\mathrm{MST}(A)} = \min_{(s-\mathrm{trees})} l_{\mathrm{SP}}(s, A)\,. \tag{1}$$

If we allow for the introduction of additional points of the manifold M on each set A, we get SPs of smaller overall length. A Steiner tree (ST) is obtained with the additional requirement of three tangent lines to only three geodesic edges meeting at an angle of 120° on each additional (Steiner) point. Among all these Steiner trees t of a set A with length $l_{\mathrm{ST}}(t, A)$, there is one whose overall length is minimum. This is called the Steiner minimal tree of set A, SMT(A). Its length is

$$l_{\mathrm{SMT}(A)} = \min_{(t-\mathrm{trees})} l_{\mathrm{ST}}(t, A)\,. \tag{2}$$

The minimum spanning tree MST(A) is the worst approximation to the Steiner minimal tree, SMT(A), or the "worst cut" for each set $A \subset M$. A common measure of this approximation is the Steiner ratio of the set $A \subset M$

$$\rho(A) = \frac{l_{\mathrm{SMT}(A)}}{l_{\mathrm{MST}(A)}}\,. \tag{3}$$

The Steiner ratio ρ_n of the manifold M is then defined as the infimum of all values $\rho(A)$ for all sets A, or

$$\rho_n = \inf_{A \subset M} \rho(A)\,. \tag{4}$$

We henceforth adopt the three-dimensional Euclidean space as the metric manifold M.

Evenly Spaced Consecutive Points – Spanning and Steiner Trees

For each set A of points in $\mathbb{R}^3$ we suppose a continuous and differentiable curve to pass by all these points. If the points along the curve are evenly spaced in terms of the Euclidean metric, we have for their position vectors

$$\|\vec{r}_{j+2} - \vec{r}_{j+1}\| = \|\vec{r}_{j+1} - \vec{r}_j\|, \tag{5}$$

where $\|\cdot\|$ represents the Euclidean norm.

A convenient representation of these vectors will be

$$\vec{r}_j = (r(\omega)\cos(j\omega), r(\omega)\sin(j\omega), jh(\omega))\,, \quad 0 \le j \le n-1\,. \tag{6}$$

The functions $r(\omega)$ and $h(\omega)$ are continuous and twice differentiable.

The position vectors $\vec{r}_j$ above have an interesting property: four of them are enough to generate all the others, or

$$\vec{r}_{j+4} = \mu\vec{r}_{j+3} + \nu\vec{r}_{j+2} + \xi\vec{r}_{j+1} + \zeta\vec{r}_j\,, \quad 0 \le j \le n-1\,, \tag{7}$$

where μ, ν, ξ, ζ are functions to be found.

This is a well-posed problem with a unique solution. We write the corresponding relations for the coordinates of the vectors in Eq. (7) as

$$\begin{aligned}(j+4)h &= \mu(j+3)h + \nu(j+2)h + \xi(j+1)h + \zeta jh\,,\\ &\qquad 0 \le j \le n-1\end{aligned} \tag{8}$$

and with an Argand representation in the $x^1 - x^2$ plane,

$$rz_{j+4} = \mu\, rz_{j+3} + \nu\, rz_{j+2} + \xi\, rz_{j+1} + \zeta\, rz_j\,, \quad 0 \le j \le n-1\,, \tag{9}$$

where

$$z_j = (z)^j = e^{ij\omega}\,. \tag{10}$$

From Eq. (9) we have

$$z^4 - \mu z^3 - \nu z^2 - \xi z - \zeta = 0\,. \tag{11}$$

We write Eq. (8) for two points j and $j + l$, and we get

$$\mu + \nu + \xi + \zeta = 1\,. \tag{12}$$

From Eqs. (12) and (8) we can write

$$\mu + 2\nu + 3\xi + 4\zeta = 0\,. \tag{13}$$

The two last equations are enough for the existence of a double $z = 1$ root of Eq. (11), or

$$(z-1)^2(z^2 + (2-\mu)z + 3 - 2\mu - \nu) = 0\,. \tag{14}$$

For complex roots of unit modulus $|z| = 1$ according to Eq. (10), we have

$$\mu^2 + 4\mu + 4\nu - 8 < 0 \tag{15}$$

$$\mu = 2(1 + \cos\omega)\,; \quad 0 \le \mu \le 4\,. \tag{16}$$

Equations (15) and (16) lead us to write

$$2\mu + \nu = 2\,. \tag{17}$$

From Eqs. (11), (12), (13) and (17) we get

$$\nu = 2 - 2\mu\,, \quad \xi = \mu\,; \quad \zeta = -1\,. \tag{18}$$

We can then write

$$\vec{r}_{j+4} = \mu\,\vec{r}_{j+3} + 2(1-\mu)\vec{r}_{j+2} + \mu\,\vec{r}_{j+1} - \zeta\,\vec{r}_j\,. \tag{19}$$

This relation will give us the motivation of thinking about tetrahedra as the fundamental pieces of this modelling. There are $n_{\mathrm{T}} = (n-3)$ tetrahedra for n points.

The values

$$\omega = \omega_{\mathrm{R}} = \pi \pm \arccos\left(\frac{2}{3}\right) \tag{20}$$

correspond to vertices of two sequences of regular tetrahedra joined together at common faces. In the literature, it is known usually by the name "3-sausage" [2]. Actually, these two values correspond to the same structure. It is itself chiral for $n \geq 6$, since for $n = 3, 4, 5$ (0, 1, 2 tetrahedra) there is a two-fold ($n = 3$) or a three-fold ($n = 4, 5$) axis of symmetry. The structures correspond to ω and $-\omega$, with $\omega \neq \omega_R$ being sequences of non-regular tetrahedra and they are chiral themselves and chiral to each other for all $n \geq 3$.

After this digression, we go back to the problem of constructing SPs for the set of n vertices of Eq. (5). A first candidate is the sequence itself. Its total length is given by

$$l_{\mathrm{SP}} = (n-1)[h^2 + r^2(A+1)]^{1/2}\,, \tag{21}$$

where

$$A = 1 - 2\cos\omega\,. \tag{22}$$

There is a necessary restriction on this spanning tree if we require that the STs to be formed below be full STs ($n-2$ Steiner points). The smallest angle between consecutive edges with the points $\vec{r}_j$ of Eq. (6) as vertices should be less than 120°. We write

$$-\frac{1}{2} < \cos\theta = -1 + \frac{r^2(A+1)^2}{2[h^2 + r^2(A+1)]}\,, \tag{23}$$

or

$$h^2 < r^2 A(A+1)\,. \tag{24}$$

This is the first restriction imposed on the position vectors $\vec{r}_j$. Actually, $r(\omega)$ is an arbitrary function for the present modelling, as will be seen in the forthcoming development.

A generalization of the last formula can be obtained by introducing subsequences of evenly spaced but non-consecutive points [4]. These subsequences are given by

$$(P_j)_{m,\,l_{P_{j\max}}}:\ \vec{r}_j, \vec{r}_{j+m}, \vec{r}_{j+2m}, \ldots, \vec{r}_{j+l_P m}, \ldots, \vec{r}_{j+l_{P_{j\max}} m}\,, \tag{25}$$

where $(m-1)$ is the number of skipped points necessary to form this sequence. The indices of the subsequences above should be restricted by

$$j + l_P m \leq n - 1\,. \tag{26}$$

We can write

$$l_{P_{j\max}} = \left[\frac{n-j-1}{m}\right]\,. \tag{27}$$

The square brackets $[x]$ stand for the greatest integer value $\leq x$.

There are m subsequences P_j, $0 \leq j \leq m-1$, and each subsequence has $(l_{P_{j\max}} + 1)$ points. We now define a new sequence by the union set of the sequences above or

$$\mathbb{P}_m = \bigcup_{j=0}^{m-1} (P_j)_{m,\,l_{P_{j\max}}}\,. \tag{28}$$

As a check of the consistency of this scheme we can see that each sequence $\mathbb{P}_m$ has n points, like the original sequence, Eq. (6). From Eq. (27) and a mathematical identity we have

$$\sum_{j=0}^{m-1}\left(l_{P_{j\max}} + 1\right) = m + \sum_{j=0}^{m-1}\left[\frac{n-j-1}{m}\right] = m + n - m = n\,. \tag{29}$$

By completeness, the scheme should also include the original sequence of consecutive points, given by Eq. (6). It is effectively given by $\mathbb{P}_1 = (P_0)_{1,\,n-1}$.

Each sequence $\mathbb{P}_m$ (Eq. 28) has an associated SP. We shall proceed now to the calculation of its length. We stress that the union of subsequences of the definition

of $\mathbb{P}_m$ is accomplished by joining two subsequences by an edge of consecutive points as far as the calculation of length is concerned.

The scheme is valid for a generic set of points on a given curve. We should write a generic Ansatz for the coordinates of the subsequences instead of Eq. (6). We have

$$\vec{r}_{j+l_P m} = (r(\omega)\cos(j + l_P m), r(\omega)\sin(j + l_P m), \cdot (j + l_P m)h(\omega)) \,. \quad (30)$$

With the prescriptions above, the length of the SP for a sequence $\mathbb{P}_m$ is

$$l_{\mathrm{SP}}^{(m)} = [m^2h^2 + r^2(A_m + 1)]^{1/2} \sum_{j=0}^{m-1} \left[\frac{n - j - 1}{m}\right] + (m - 1)[h^2 + r^2(A_1 + 1)]^{1/2} \,, \quad (31)$$

where

$$A_1 = A \text{ and } A_m = 1 - 2\cos(m\omega) \,. \quad (32)$$

After using the mathematical identity in the last equality of Eq. (29), we get

$$l_{\mathrm{SP}}^{(m)} = (n - m)[m^2h^2 + r^2(A_m + 1)]^{1/2} + (m - 1)[h^2 + r^2(A_1 + 1)]^{1/2} \,. \quad (33)$$

There is an analogous restriction to Eq. (24) on the angles between edges of subsequences. It is written as

$$m^2h^2 < r^2A_m(A_m + 1) \,, \quad \forall m \,. \quad (34)$$

The length of the MST will be given by

$$l_{\mathrm{SP}} = \min_{(m)} \left\{ l_{\mathrm{SP}}^{(m)} \right\} \,. \quad (35)$$

The $\min_{(m)}\{\ldots\}$ process above should be understood in the sense of formation of a piecewise function by the functions corresponding to values $m = 1, 2, 3, \ldots$

We can apply the same scheme to Steiner points and their connecting edges. The original sequence is

$$\vec{S}_k = (R(\omega)\cos k\omega, R(\omega)\sin k\omega, kH(\omega)) \,, \quad 1 \le k \le n - 2 \,, \quad (36)$$

where $R(\omega)$ and $H(\omega)$ are also continuous and twice differentiable functions.

We now form the subsequences

$$(S_k)_{m,\, l_{S_k\max}} : \quad \vec{S}_k, \vec{S}_{k+m}, \vec{S}_{k+2m}, \ldots, \vec{S}_{k+l_S m}, \ldots, \vec{S}_{k+l_{S_j\max} m} \,, \quad (37)$$

where $(m - 1)$ is the number of skipped points.

The restriction on the indices is

$$k + l_S m \le n - 2 \,, \quad (38)$$

and we have

$$l_{S_k\max} = \left[\frac{n - k - 2}{m}\right] \,. \quad (39)$$

We also have m subsequences S_k, $1 \le k \le m$, each of which has $(l_{S_k\max} + 1)$ points. We define a new sequence of Steiner points by the union set of the sequences S_k, $1 \le k \le m$:

$$\mathbb{S}_m = \bigcup_{k=1}^{m} (S_k)_{m,\, l_{S_k\max}} \,. \quad (40)$$

These new sequences have $(n - 2)$ points each. This can be checked by using Eq. (38), or

$$\sum_{k=1}^{m} (l_{S_k\max} + 1) = m + \sum_{k=1}^{m} \left[\frac{n - k - 2}{m}\right] = m + n - m - 2 = n - 2 \,. \quad (41)$$

The scheme includes trivially the original sequence of Eq. (36). It is given by $\mathbb{S}_1 = (S_1)_{1,\, n-1}$.

The coordinates of these subsequences should be written generically as

$$\vec{S}_{k+l_S m} = (R_m(\omega)\cos(k + l_S m), R_m(\omega) \cdot \sin(k + l_S m), (k + l_S m)H_m(\omega)) \,, \quad (42)$$

where $R_m(\omega)$ and $H_m(\omega)$ are continuous and twice differentiable functions.

Steiner Trees

The ST for each subsequence will be organized as follows: first the points $\vec{S}_{k+l_S m}$ of the subsequence $(S_k)_{m,\, l_{S_k\max}}$ for a given m will be connected consecutively to each other. The first of these points, $\vec{S}_{k+m}$, will be connected to the first two points, $\vec{r}_j$ and $\vec{r}_{j+m}$, of the

subsequence $(P_j)_{m,\, l_{P_{j\max}}}$. The last point, $S_{k+l_{S_k\max}m}$, will be connected to the last two points, $\vec{r}_{j+(l_{P_{j\max}}-1)m}$ and $\vec{r}_{j+l_{P_{j\max}}m}$. All the intermediary points $\vec{S}_{k+l_S m}$ will be connected to the intermediary points $r_{j+l_p m}$, for $j = k$. This means that we assume a path topology [2] for each subsequence.

The requirement of edges meeting at an angle of 120° on each Steiner point leads to the following relations:

$$H_m = h\,; \quad m^2 H_m^2 = R_m^2 A_m (A_m + 1)\,, \quad \forall m\,. \tag{43}$$

From Eqs. (34) and (43) we have trivially,

$$R_m < r\,. \tag{44}$$

From Eq. (44) the length of the ST corresponding to the sequences $\mathbb{P}_m$ and $\mathbb{S}_m$ and a path topology will be given by

$$\begin{aligned} l_{\mathrm{ST}}^{(m)} &= (r - R_m)\left(m + \sum_{k=1}^{m}\left[\frac{n-k-2}{m}\right]\right) \\ &\quad + \left[m^2 H_m^2 + R_m^2 (A_m + 1)\right]^{1/2} \sum_{k=1}^{m}\left[\frac{n-k-2}{m}\right] \\ &\quad + 2\left[m^2 H_m^2 + (r - R_m)^2 + r R_m (A_m + 1)\right]^{1/2}\,. \end{aligned} \tag{45}$$

From Eqs. (43) and the mathematical identity used in Eq. (41) we write

$$\begin{aligned} l_{\mathrm{ST}}^{(m)} &= n(r - R_m) + (n - m - 2) R_m (A_m + 1) \\ &\quad + 2\left[(r - R_m)^2 + R_m (r + R_m A_m)(A_m + 1)\right]^{1/2}\,. \end{aligned} \tag{46}$$

Actually, we have used Eq. (43) for the two ends of the tree. Their contribution to the length is the last term in Eq. (46). In order to satisfy the condition of meeting edges at 120° there, we need to take a special limit $R_m \to r$. It is worthwhile for future modelling applications to notice that this procedure leads to the same result as the limit for $l_{\mathrm{ST}}^{(m)}$ for large numbers n of atoms. This is easy to see from Eq. (46).

$$l_{\mathrm{ST}}^{(m)}\ (n \gg 1 \text{ or } \underbrace{R_m \to r}_{\text{at ends}}) = nr + [(n-m)A_m - m] R_m\,. \tag{47}$$

The Steiner Ratio Function

We follow the prescription of Eq. (3) for writing an expression for the Steiner ratio function. It will be given by

$$\begin{aligned} \rho = \frac{\min_{(m)} \left\{ n + (A_m (A_m + 1))^{-1/2} \right.}{\min_{(m)} \left\{ (n-m)[A_m + 1 + m^2 (F(\omega))^2]^{1/2} \right.} \cdots \\ \cdots \frac{\left. [(n-m)A_m - m] m F(\omega) \right\}}{\left. + (m-1)[A_1 + 1 + (F(\omega))^2]^{1/2} \right\}} \end{aligned} \tag{48}$$

where $F(\omega) \equiv \frac{h(\omega)}{r(\omega)}$ is a function restricted by

$$F(\omega) < \min_{(m)} \left\{ \frac{1}{m} [A_m (A_m + 1)]^{1/2} \right\}\,. \tag{49}$$

The application to protein modelling could be done by classifying the minimum energy values (minima of ρ) of protein structures on protein data banks [6] after choosing a convenient function $F(\omega)$.

For application to the Steiner ratio problem of discrete mathematics, we can see that for very large set of points ($n \gg 1$), the ratio function Eq. (48) can be written

$$\rho(n \gg 1) = \frac{\min_{(m)} \left\{ 1 + m F(\omega) \left(\frac{A_m}{A_m + 1} \right)^{1/2} \right\}}{\min_{(m)} \left\{ \left[A_m + 1 + m^2 \left(F(\omega) \right)^2 \right]^{1/2} \right\}}\,. \tag{50}$$

Let the function $F(\omega)$ be chosen such that the $\min_{(m)}$ process in the numerator of Eq. (49) is dominated by the term corresponding to a fixed value of $m = \tilde{m}$. The problem of Eq. (49) will then be solved by

$$\rho(\omega, F, \tilde{m}) = \mathrm{Max}_{(m)} \left\{ \frac{1 + \tilde{m} F(\omega) \left(\frac{A_{\tilde{m}}}{A_{\tilde{m}} + 1} \right)^{1/2}}{[A_m + 1 + m^2 (F(\omega))^2]^{1/2}} \right\}\,. \tag{51}$$

For $\tilde{m} = 1$ the function of Eq. (50) has the global minimum

$$\omega_1 = \pi - \arccos \frac{2}{3}\,; \quad F(\omega_1) = \frac{\sqrt{30}}{9} \tag{52}$$

and

$$\begin{aligned} \rho(\omega_1, F(\omega_1), \tilde{m} = 1) &= \frac{1}{10}(3\sqrt{3} + \sqrt{7}) \\ &= 0.78419037337\,. \end{aligned} \tag{53}$$

This is the value that was conjectured in the literature [2] as the best upper bound of the Steiner ratio for three-dimensional Euclidean space.

We also report that the value $m = \tilde{m} = 1$ can be selected by a linear function $F(\omega) = \alpha\omega$, which means right circular helices for the geometrical locus of the points $\vec{r}_j$ and $\vec{S}_k$ of our modelling. Furthermore, the function ρ obtained by taking also $m = \tilde{m} = 1$ in the denominator is a convex envelope in this case.

Concluding Remarks

The successful application of the scheme introduced in the foregoing pages reinforce the idea of studying their consequences as well as classifying biomolecular structures in terms of associated Steiner trees. It is a geometrical approach to the fundamental problem of energy minimization of these structures, and it can shed some light on the problem of biomolecular formation and evolution. Some additional knowledge of protein structure [1,7] should be introduced into our analysis, like amide planes and their twisting angles and a residual Fermat problem already solved by Nature for the placement of α-carbon atoms. The only information we used was the placement of carbon and nitrogen atoms as Steiner points. We hope that the scheme developed here can be extended to create a useful definition of molecular chirality and a well-posed optimization problem.

References

1. Brandon C, Tooze J (1991) Introduction to Protein Structure. Garland, New York
2. Du D-Z, Smith WD (1996) Disproofs of Generalized Gilbert-Pollak Conjecture on the Steiner Ratio in Three or more Dimensions. J Comb Theory A74:115–130
3. MacGregor SJ (2006) Steiner Minimal Trees, Twist Angles, and the Protein Folding Problem. In: BIOMAT 2005, International Symposium on Mathematical and Computational Biology, World Scientific Co. Pte. Ltd., pp 299–326
4. Mondaini RP (2006) Steiner Trees as Intramolecular Networks of the Biomacromolecular Structures. In: BIOMAT 2005, International Symposium on Mathematical and Computational Biology, World Scientific, pp 327–342
5. Mondaini RP (2007) Euclidean Full Steiner Trees and the Modelling of Biomolecular Structures. In: BIOMAT 2006, International Symposium on Mathematical and Computational Biology, World Scientific Co. Pte. Ltd., pp 247–258
6. Protein Data Bank (2004) Education Section. http://www.rothamsted.bbsrc.ac.uk/notebook/courses/guide/aa.htm
7. Voet D, Voet J (1990) Biochemistry. Wiley, New York

Steiner Tree Problems

STP

DING-ZHU DU[1], BING LU[2], HUANG NGO[3], PANOS M. PARDALOS[4]

[1] Department of Computer Science and Engineering, University of Texas at Dallas, Richardson, USA
[2] Department Computer Sci. and Engineering, University Minnesota, Minneapolis, USA
[3] Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, USA
[4] Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 90C27

Article Outline

Keywords
On the Proof of Gilbert–Pollak's Conjecture
- Chung–Gilbert's Conjecture
- Graham–Hwang's Conjecture
- The Steiner Ratio in Banach Spaces

On Better Approximations
- Chang's Idea
- Zelikovsky's Idea
- The *k*-Steiner Ratio ρ_k
- Variable Metric Method

On PTAS
- Arora's PTAS
- Mitchell's PTAS

Variations of Steiner Trees
- Steiner Arborescence
- Edge-length and Number of Steiner Points
- Multiphase

See also
References

Keywords

Combinatorial optimization; Steiner tree; Steiner ratio; Approximation algorithms; variations of Steiner trees

The Steiner tree is an *NP*-hard combinatorial optimization problem [50] with a long history [11,66,93]. The study of Steiner trees received great attention in the 1990s since many important open problems, including

the *Gilbert–Pollak conjecture* on the *Euclidean Steiner ratio*, the existence of better approximation, and the existence of polynomial time approximation schemes (PTAS), have been solved with influence in the general theory of designs and analysis of approximation algorithms for combinatorial optimization, and also, many new important applications in VLSI designs, optical networks, wireless communications, etc. have been discovered and studied extensively. Those applications usually require some modifications on classical Steiner tree problems and hence require new techniques for solving them. Therefore, studying various variations of Steiner trees became an exciting activity recently. In this article, we will review important developments in the 1990s and discuss some open problems which may motivate important developments in this century.

On the Proof of Gilbert–Pollak's Conjecture

Given a set of points in a metric space, the problem is finding a shortest network interconnecting the points in the set. Such a shortest network is called a *Steiner minimum tree* on the point set. The Steiner tree problem can be seen as a generalization of *Fermat's problem*. Around 1700, P. Fermat proposed a problem of finding a point to minimize the total distance from this point to three given points in the Euclidean plane; its solution is exactly the Steiner minimum tree on the three points. The general form of Steiner minimum tree problem was proposed by C.F. Gauss [26]. However, R. Courant and H. Robbins [27] referred to it as the Steiner problem. The popularity of their book was responsible for bringing the Steiner tree problem to people's attention. Two important papers in the 1960s further laid a solid groundwork for additional study. Z.A. Melzak [75] first gave a finite algorithm for the Euclidean Steiner trees. E.N. Gilbert and H.O. Pollak [52] produced an excellent survey of the problem, raised many new topics including Steiner ratio problem, and extended the problem to other metric space. Since then, more than three hundred research papers have been written contributing to the Steiner tree problem. For an excellent survey, see [55].

An important development on the Steiner tree problem that took place in the beginning of the 1990s is the proof of Gilbert–Pollak's conjecture on the Euclidean Steiner ratio [32,33]. This new development is based on the discovery of a new approach with a new *minimax theorem*.

A *minimum spanning tree* on a set of points is the shortest network interconnecting the points in the set with all edges between the points. While the Steiner tree problem is intractable, the minimum spanning tree can be computed pretty fast. The Steiner ratio in a metric space is the largest lower bound for the ratio between lengths of a minimum Steiner tree and a minimum spanning tree for the same set of points in the metric space, which is a measure of performance for the minimum spanning tree as a polynomial time approximation of the minimum Steiner tree. Determin- ing the Steiner ratio in each metric space is a traditional problem on Steiner trees. In 1976, F.K. Hwang [54] determined that the Steiner ratio in a rectilinear plane is 2/3. However, it took 22 years to complete the story of determining the Steiner ratio in the Euclidean plane. In 1968, Gilbert and Pollak conjectured that the Steiner ratio in the Euclidean plane is $\sqrt{3}/2$. Through efforts made by several authors [14,24,31,31,36,48,53,80,86,87], and [23], the conjecture was finally proved by D.-Z. Du and Hwang [32,33] in 1990. The significance of their proof stems also from the potential applications of the new approach included in the proof.

In their approach, the central part is a new minimax theorem about minimizing the maximum value of several concave functions over a simplex as follows.

Theorem 1 (*Du–Hwang minimax theorem*) *Let $f(x) = \max_{i \in I} g_i(x)$, where I is a finite set and $g_i(x)$ is a continuous, concave function in a polytope X. Then the minimum value of $f(x)$ over the polytope X is achieved at some critical point, namely, a point satisfying the following property:*

**) There exists an extreme subset Y of X such that $x \in Y$ and the index set $M(x)$ ($= \{i: f(x) = g_i(x)\}$) is maximal over Y.*

The Steiner ratio problem is first transferred to such a minimax problem ($g_i(x)$ = (the length of a Steiner tree)−(the Steiner ratio)·(the length of a spanning tree with graph structure i), where x is a vector whose components are edge-lengths of the Steiner tree) and the minimax theorem reduces the minimax problem to the problem of finding the minimax value of the concave functions at critical points. Then each critical point is transferred back to an input set of points with special

geometric structure; it is a subset of a lattice formed by equilateral triangles. This special structure enables us to verify the conjecture corresponding to the nonnegativeness of minimax value of the concave functions.

Clearly, in order to use the minimax approach, for each problem three questions will be addressed:

1) How do we transfer the problem to such a minimax problem meeting the condition that the functions are concave?
2) How do we determine the critical geometric structure?
3) How do we verify the function value on the critical structure?

Developing techniques for answering these three questions will enable us to solve more open problems. Let us explain it by some examples in the following.

Chung–Gilbert's Conjecture

Steiner trees in Euclidean spaces have an application in constructing phylogenetic trees [17]. It was also conjectured by Gilbert and Pollak [52] that in any Euclidean space the Steiner ratio is achieved by the vertex set of a regular simplex. F.R.K. Chung and Gilbert [22] constructed a sequence of Steiner trees on regular simplices. The lengths of constructed Steiner trees goes decreasingly to $\sqrt{3}/(4-\sqrt{2})$. Although the constructed trees are not known to be Steiner minimum trees, Chung and Gilbert conjectured that $\sqrt{3}/(4-\sqrt{2})$ is the best lower bound for Steiner ratios in Euclidean spaces. Clearly, if $\sqrt{3}/(4-\sqrt{2})$ is the limiting Steiner ratio in d-dimensional Euclidean space as d goes to infinity, then *Chung–Gilbert's conjecture* is a corollary of Gilbert and Pollak's general conjecture. However, this general conjecture of Gilbert and Pollak has been disproved by J.M. Smith [92] for dimension from three to nine and by Du and Smith for dimension larger than two. Now, interesting questions which arise in this situation are about Chung and Gilbert's conjecture. Could Chung–Gilbert's conjecture also be false? If the conjecture is not false, can we prove it by the minimax approach?

First, we claim that Chung–Gilbert's conjecture could be true. In fact, we could get rid of Gilbert–Pollak's general conjecture, and use another way to reach the conclusion that the limiting Steiner ratio for regular simplex is the best lower bound for Steiner ratios in Euclidean spaces. To support our viewpoint, let us analyze a possible proof of such a conclusion as follows.

Consider n points in $(n-1)$-dimensional Euclidean space. Then all of $n(n-1)/2$ distances between the n points are independent. Suppose that we could do a similar transformation and the minimax theorem could apply to these n points to obtain a similar result in the proof of Gilbert–Pollak's conjecture for Euclidean plane, i. e. a point set with critical geometric structure has the property that the union of all minimum spanning trees contains as many equilateral triangles as possible. Then such a critical structure must be a regular simplex.

The above observation tells us two facts:

a) Chung–Gilbert's conjecture can follow from the following two conjectures.

Conjecture 2 The Steiner ratio for n points in a Euclidean space is not smaller than the Steiner ratio for the vertex set of $(n-1)$-dimensional regular simplex.

Conjecture 3 (Smith's conjecture [92]) $\sqrt{3}/(4-\sqrt{2})$ is the limiting Steiner ratio for simplex.

b) It may be possible to prove Conjecture 2 by the minimax approach if we could find the right transformation.

One may wonder why we need to find a right transformation. What happens to the transformation used in the proof of Gilbert–Pollak's conjecture in the Euclidean plane? Here, we remark that such a transformation does not work for Conjecture 2. In fact, in the Euclidean plane, with a fixed graph structure, all edge-lengths of a full Steiner tree can determine the set of original points and furthermore the length of a spanning tree for a fixed graph structure is a convex function of the edges-lengths of the Steiner tree. However, in Euclidean spaces of dimension more than two, edge-lengths of a full Steiner tree are not enough to determine the set of original points. Moreover, adding other parameters may destroy the convexity of the length of a spanning tree as a function of the parameters.

Smith [92] showed by an exhaustive computation that for $d = 3, \ldots, 7$, the Steiner trees constructed by Chung and Gilbert are actually minimum Steiner trees, but, for $d = 8$, their Steiner tree is not minimum. He also conjectured that the trees of Chung and Gilbert are minimum if d is of the form $d = 3\cdot 2^p$. Conjecture 3 is a corollary of this more specific conjecture.

From the above, we see that proving Chung–Gilbert's conjecture requires a further development of the minimax approach.

Graham–Hwang's Conjecture

A Steiner tree with rectilinear distance is called a *rectilinear Steiner tree*. While rectilinear Steiner trees in plane have many applications on CAT and VLSI, rectilinear Steiner trees in high-dimensional space can be found in biology [17,47] and optimal traffic multicasting for some communication networks [13,18]. Although the Steiner ratio in rectilinear plane was determined by Hwang [54] in an earlier stage of the study of Steiner trees, there is still (as of 2000) no progress on the Steiner ratio in rectilinear spaces by now. The Steiner ratio in a d-dimensional rectilinear space was conjectured to be $d/(2d-1)$ by Graham and Hwang [53]. The difficulty for extending Hwang's approach to proving *Graham–Hwang's conjecture* is due to the lack of knowledge on the full rectilinear Steiner trees in high-dimensional spaces. (A full Steiner tree has a property that all original points are leaves.) In fact, for a full rectilinear Steiner tree in plane, all Steiner points lie on a path. However, it is not known whether a similar result holds for full rectilinear Steiner trees in a space of dimension more than two.

Graham–Hwang's conjecture can be easily transferred to a minimax problem required by our minimax approach. For example, choose lengths of all straight segments of a Steiner tree. When the connection pattern of the Steiner tree is fixed, the set of original points can be determined by such segments-lengths, the length of the Steiner tree is a linear function and the length of a spanning tree is a convex function of such segment-lengths, so that g_i is a concave function of such segment-lengths. However, for this transformation, it is hard to determine the critical structure. To explain the difficulty, we notice that in general the critical points could exist in both the boundary and interior of the polytope. (See the minimax theorem.) In the proof of Gilbert–Pollak's conjecture in plane, a crucial fact is that only interior critical points need to be considered in a contradiction argument. The critical structure of interior critical points are relatively easy to be determined. However, for the current transformation on Graham–Hwang's conjecture, we have to consider some critical points on the boundary. It requires a new technique, either determine critical structure for such critical points or eliminate them from our consideration.

One possible idea is to combine the minimax approach and Hwang's method. In fact, by the minimax approach, we may get a useful condition on the set of original points. With such a condition, the point set can have only certain type of full Steiner trees. This may reduce the difficulty of extending Hwang's method to high dimension.

The significance of developing techniques for determining critical structure corresponding to critical points on the boundary is not only for solving Graham–Hwang's conjecture, but also for solving some other problems. For example, it can be immediately applied to some packing problems. One of the typical packing problems is to find the maximum number of objects which can be put in a certain container. When the objects are discs or spheres, the problem can be transferred to a minimax problem that meets our requirement. To determine such a number exactly, we have also to deal with critical points on the boundary of the polytope.

The Steiner Ratio in Banach Spaces

Examining the proof of Gilbert–Pollak's conjecture in the Euclidean plane, we observe that the proof has nothing concerning the property of Euclidean norm except the last part, verification of the conjecture on point sets of critical structure. This means that using the minimax approach to determine the Steiner ratio in *Minkowski plane* (2-dimensional Banach space), we would have no problem in finding a transformation and determining critical structures. We would meet only a problem on verification for point sets with critical structure.

Steiner minimum trees in Minkowski planes have been studied by [1,25,30,34,70,91]. In these papers, some fundamental properties of Steiner minimum trees in Minkowski planes have been established. Two nice conjectures about the Steiner ratio in Minkowski planes were proposed respectively by [25,30] and [30] as follows:

Conjecture 4 In any Minkowski plane, the Steiner ratio is between 2/3 and $\sqrt{3}/2$.

Conjecture 5 The Steiner ratio in a Minkowski plane equals that in its dual plane.

With new techniques in the critical structures, B. Gao, Du and Graham [49] proved the first half of Conjecture 4 that in any Minkowski plane, the Steiner ratio is at least 2/3, and P.-J. Wan, Du and Graham [97] showed that Conjecture 5 is true for three, four, and five points. With a different approach, Du and others [30] also proved that in any Minkowski plane, the Steiner ratio is at most 0.8766.

The Chung–Gilbert conjecture and Conjecture 5 can be extended to high-dimensional Banach spaces as follows.

Conjecture 6 In any infinite-dimensional Banach space, the Steiner ratio is between 1/2 and $\sqrt{3}/(2-\sqrt{2})$.

Conjecture 7 The Steiner ratio in any Banach space equals that in its dual space.

Significant results on these two conjectures could be produced by further developments of minimax approach from a successful application in two-dimensional problems to high-dimension.

On Better Approximations

Starting from a minimum spanning tree, improve it by adding Steiner points. This is a natural idea to obtain an approximation solution for the Steiner minimum tree. Every approximation solution obtained in this way would have a performance ratio at most the inverse of the Steiner ratio. The problem is how much better than the inverse of the Steiner ratio one can make.

From the 1980s onwards numerous heuristics [6,13,19,44,61,63,64,65,67,94,100] for Steiner minimum trees have been proposed for points in various metric spaces. Their superiority over minimum spanning trees were often claimed by computation experiments. But no theoretical proof of superiority was ever given. It was a long-standing problem whether there exists a polynomial time approximation with a performance ratio better than the inverse of the Steiner ratio or not. For simplicity, a polynomial time approximation with performance ratio smaller than the inverse of the Steiner ratio will be called a *better approximation*. The first significant work on better approximations was made by M.W. Bern [10]. He proved that for the rectilinear metric and Poisson distributed regular points, a greedy approximation obtained by a very simple improvement over a minimum spanning tree has a shorter average length. Later, Hwang and Y.C. Yao [56] extended this result to the usual case when the number of regular points is fixed.

In 1991, A.Z. Zelikovsky [102] made the first breakthrough to the problem by giving a better heuristic for the Steiner minimum trees in graph. This is the second important development on Steiner trees in 1990s. To explain his idea and review further development from his work, let us start from comparing his work with a previous work with a similar idea.

Chang's Idea

Chang [18,19] proposed the following approximation algorithm for Steiner minimum trees in the Euclidean plane: Start from a minimum spanning tree and at each iteration choose a Steiner point such that using this Steiner point to connect three vertices in the current tree could replace two edges in the minimum spanning tree and this replacement achieves the maximum saving among such possible replacements.

Smith, D.T. Lee and J.S. Liebman [90] also use the idea of the greedy improvement. But, they start with Delaunay triangulation instead of a minimum spanning tree. Since every minimum spanning tree is contained in Delaunay triangulation, the performance ratio of their approximation algorithm can also be bounded by the inverse of the Steiner ratio. The advantage of Smith–Lee–Liebman algorithm is on the running time. While Chang's algorithm runs in $O(n^3)$ time, Smith–Leeh–Liebman algorithm runs only in $O(n \log n)$ time.

A. Kahng and G. Robin [60] proposed an approximation algorithm for Steiner minimum trees in the rectilinear plane by using the same idea as that of Chang. For these three algorithms, it can be proved that for any particular set of points, the ratio of lengths of the approximation solution and the Steiner minimum tree is smaller than the inverse of the Steiner ratio. Some experimental results also show that the approximation solution obtained by these algorithms are very good. However, no proof has been found to show any one of them being a better approximation.

Zelikovsky's Idea

Zelikovsky's idea[102] is based on the decomposition of a Steiner tree (namely, a tree, not necessarily minimum, interconnecting original points): An original point in a Steiner tree can be either a leaf or a junction. In the latter case, the Steiner tree can be decomposed at this point. In this way, every Steiner tree can be decomposed into edge-disjoint union of several Steiner trees for subsets of original points; each of them has no junction being an original point. A Steiner tree with no original point being a junction is called a *full Steiner tree*. The full Steiner trees in the decomposition are called *full components*. The size of a full component is the number of original points in the component.

Clearly, for any $k \geq 3$, a k-size Steiner minimum tree usually has shorter length compared with a minimum spanning tree. It is natural to think about using a minimum k-size Steiner tree to approximate the Steiner minimum tree. However, this does not work because computing a k-size Steiner minimum tree is still an intractable problem. Zelikovsky's idea is to approximate the Steiner minimum tree by a 3-size Steiner tree generated by a polynomial time greedy algorithm. The key fact is that the length of such a heuristic is smaller than the arithmetic mean of lengths of a minimum spanning tree and a 3-size Steiner minimum tree; that is, the performance ratio of his approximation satisfies

$$\mathrm{PR} \leq \frac{\rho_2^{-1} + \rho_3^{-1}}{2},$$

where ρ_k is the k-Steiner ratio. Thus, if the 3-Steiner ratio ρ_3 is bigger than the Steiner ratio ρ_2, then this greedy algorithm is a better approximation for the Steiner minimum tree. Zelikovsky was able to prove that 3-Steiner ratio in graphs is at least 3/5 which is bigger than 1/2, the Steiner ratio in graphs [61]. So, he solved the better approximation problem in graphs. Zelikovsky's idea has been extensively studied in the literature.

Du, Zhang, and Q. Feng [40] generalized Zelikovsky's idea to the k-size Steiner tree. They showed that a generalized Zelikovsky's algorithm has performance ratio

$$\mathrm{PR} \leq \frac{(k-2)\rho_2^{-1} + \rho_k^{-1}}{k-1}.$$

P. Berman and V. Ramaiyer [9] employed a different idea to generalize Zelikovsky's result. They obtained an algorithm with the performance ratio satisfying

$$\mathrm{PR} \leq \rho_2^{-1} - \sum_{i=3}^{k} \frac{\rho_{i-1}^{-1} - \rho_i^{-1}}{i-1}.$$

They also showed that in the rectilinear plane, the 3-Steiner ratio is at least 72/94 which is bigger than 2/3 [54], the Steiner ratio in rectilinear plane. So, they solved the better heuristic problem in rectilinear plane.

Du, Zhang, and Feng [40] proved a lower bound for the k-Steiner ratio in any metric space. This lower bound goes to one as k goes to infinity. So, in any metric space with the Steiner ratio less than one, there exists a k-Steiner ratio bigger than the Steiner ratio. Thus, they proved that the better heuristic exists in any metric space satisfying the following conditions:

1) the Steiner ratio is smaller than one;
2) the Steiner minimum tree on any fixed number of points can be computed in polynomial time.

These metric spaces include Euclidean plane and Euclidean spaces.

Zelikovsky [104] used a different potential function in his greedy approximation and obtained an approximation with performance ratio satisfying

$$\mathrm{PR} \leq \rho_k^{-1}(1 - \ln \rho_2).$$

Although Zelikovsky's idea starts from a point different from Chang's one, the two approximations are actually similar. To see this, let us describe Zelikovsky's algorithm as follows: Start from a minimum spanning tree and at each iteration choose a Steiner point such that using this Steiner point to connect three regular points could replace two edges in the minimum spanning tree and this replacement achieves the maximum saving among such possible replacements.

Clearly, they both start from a minimum spanning tree and improve it step by step by using a greedy principal to choose a Steiner point to connect a triple of vertices. The difference is only that this triple in Chang's algorithm may contain some Steiner points while it contains only regular points in Zelikovsky's algorithm. This difference makes Chang's approximation difficult to be analyzed. Which one will give a better approximation solution? This is an interesting problem.

The k-Steiner Ratio ρ_k

While the determination of the k-Steiner ratio plays an important role in estimation of the performance ratio of several recent better approximations, A. Borchers and Du [15] completely determined the k-Steiner ratio in graphs that for $k = 2^r + h \geq 2$,

$$\rho_k = \frac{r2^r + h}{(r+1)2^r + h}$$

and Borchers, Du, Gao, and Wan [16] completely determined the k-Steiner ratio in the rectilinear plane that $\rho_2 = 2/3$, $\rho_3 = 4/5$, and for $k \geq 4$, $\rho_k = (2k-1)/(2k)$. However, the k-Steiner ratio in the Euclidean plane for $k \geq 3$ is still (as of 2000) an open problem. Du, Zhang, and Feng [40] conjectured that the 3-Steiner ratio in the Euclidean plane is

$$\frac{(1+\sqrt{3})\sqrt{2}}{1+\sqrt{2}+\sqrt{3}}.$$

They also analyzed that the k-Steiner ratio in the Euclidean plane might be determined in a similar way to the proof of Gilbert–Pollak conjecture. The difficulty appears only in the description of 'critical structure'.

Variable Metric Method

Berman and Ramaiyer [9] introduced an interesting approach to generalize Zelikovsky's greedy approximation. Let us call the Steiner minimum tree for a subset of k regular points as a k-tree. Their approach consists of two steps. The first step processes all i-trees, $3 \leq i \leq k$, sequentially in the following way: For each i-tree T with positive saving in the current graph, put T in a stack and if two leaves x and y of T are connected by a path p in a minimum spanning tree without passing any other leaf of T, then put an edge between x and y with weight equal to the length of the longest edge in p minus the saving of T. In the second step, it repeatedly pops i-trees from the stack remodifying the original minimum spanning tree for all regular points and keeping only i-trees with the current positive saving. Adding weighted edges to a point set would change the metric on the points set. Let E be an arbitrary set of weighted edges such that adding them to the input metric space makes all i-trees for $3 \leq i \leq k$ have nonpositive saving in the resulting metric space M_E. Denote by $t_k(P)$ a supremum of the length of a minimum spanning tree for the point set P in metric space M_E over all such E. Then Berman–Ramaiyer's algorithm produces a k-size Steiner tree with total length at most

$$t_2(P) - \sum_{i=3}^{k} \frac{t_{i-1}(P) - t_i(P)}{i-1} = \frac{t_2(P)}{2} + \sum_{i=3}^{k-1} \frac{t_i(P)}{(i-1)i} + \frac{t_k(P)}{k-1}.$$

The bound for the performance ratio of Berman–Ramaiyer's approximation above is obtained from this bound and the fact that $t_k(P) \leq \rho_k^{-1}\,\mathrm{SMT}(P)$ where $\mathrm{SMT}(P)$ is the length of the Steiner minimum tree for point set P.

Based on the above observation, we may have the following questions. Could we find another way to vary metric for a better bound? Could we forget the greedy idea and design a better approximation with only a variable metric idea? Answering these questions requires deeper understanding the of variable metric method. We attempt to obtain new algorithms from this study.

M. Karpinski and Zelikovsky [62] proposed a preprocessing procedure to improve existing better approximations. First, they use this procedure to choose some Steiner points and then run a better approximation algorithm on the union of the set of regular points and the set of chosen Steiner points. This preprocessing improves the performance ratio for every known better approximation that we mentioned previously.

The preprocessing procedure is similar to the algorithm of Berman and Ramaiyer. But, it uses a 'related gain' instead of the saving as the greedy function. One of our current ideas is to modify Chang's algorithm in the following way: At each iteration, if a Steiner point is introduced, then computes its related gain, and later consider only triples of regular points and Steiner points with positive related gain. Would this approximation perform better? We attempt to get the answer.

Although many better approximations have been found in recent years, none of them has performance ratio smaller than the inverse of 3-Steiner ratio. The inverse of the 3-Steiner ratio seems to be the limit for the performance ratio of polynomial time approximations for Steiner minimum trees to be able to reach.

S. Arora and others [5] conjectured that their backtrack greedy technique gives a polynomial time approx-

imation scheme to 3-size Steiner minimum trees. If their conjecture is true, then their algorithms also give approximations for Steiner minimum trees with performance ratio approach to the inverse of the 3-Steiner ratio. This probably is the best possible performance ratio. Thus, the conjecture of Arora and others is an attractive problem to our further research.

A more accurate analysis [62,101,104] for the performance ratios of Berman–Ramaiyer's algorithm and Karpinski–Zelikovsky's preprocessing requires bounds for t_k and a similar number t^k. The techniques in [15,16] for determining the k-Steiner ratio seems very promising for establishing tight upper bounds for t_k and t^k.

The knowledge for the lower bound of the performance ratio is an open problem (as of 2000). One knows only that for Steiner minimum trees in graphs, if $NP \neq P$, then a lower bound larger than one exists, because the problem in this case is MAX SNP-complete [12].

On PTAS

T. Jiang and others [58,59] brought a quite different idea from previous ones to Steiner minimum trees. They decompose the set of regular points based on the lengths of edges in a minimum spanning tree. By an interesting analysis, they proved that if the ratio of lengths between the longest edge and the shortest edge in a minimum spanning tree is bounded by a constant, then there is a polynomial time approximation scheme (PTAS) for Steiner minimum trees in the rectilinear plane and in the Euclidean plane. This idea can also be used in other geometric optimization problems, in particular, some variations of Steiner tree problems described in the next section.

In 1995, Arora and J.S.B. Mitchell independently discovered powerful techniques to establish polynomial time approximation schemes for geometric optimization problems, including Euclidean and rectilinear Steiner tree problems. Their results constitute the third important development on Steiner trees in 1990s. The significance of their results is not only on Steiner trees, but also on the design and analysis of approximation algorithms in combinatorial optimization. Let us review these two remarkable techniques in the following.

Arora's PTAS

It is quite interesting to note that Arora [4] appeared only one week before Mitchell [76]. Any way, they use very different techniques to reach the same goal. Therefore, both are very interesting. Arora's technique is based on recursive partition. In Jiang and others [58,59], although partition can be moved parallelly, the size of each cell is fixed. It cannot be varied according to local information about distribution of terminals. Therefore, only in case that terminals are distributed almost evenly, could the partition work well. This is why such a condition that the ratio of lengths between the longest edge and the shortest edge in a minimum spanning tree is bounded by a constant is required.

However, in Arora's recursive partition, each big cell is partitioned into small cells independently from other big cells. How to cut only depends on the situation inside of itself. This advantage enables him to discard the condition in Jiang and others [58,59].

Mitchell's PTAS

Mitchell's technique was initiated from studying a minimum length *rectangular partition problem*. Given a rectilinear region R surrounded by a rectilinear polygon and some rectilinear holes, a *rectangular partition* of R is a set of segments in R, which divide R into small rectangles each of which does not contain any hole in its interior. The problem is to find such a rectangular partition with the minimum total length. This problem is *NP*-hard.

Du and others [38] introduced a concept of *guillotine subdivision*. A guillotine subdivision is a sequence of cuts performed recursively such that each cut partitions a piece into at least two. Du and others [38] showed that the minimum length guillotine rectangular partition can be computed in polynomial time. However, they were only able to show that this guillotine subdivision is an approximation of the minimum length rectangular partition problem with performance ratio two in a special case that the region R is surrounded by a rectangle with some points as holes in it. Mitchell [77] showed that this is actually true in general. He also successfully utilized this technique to obtain constant approximations for other geometric optimization problems. With the same technique, C. Mata [74] obtained a constant-factor approximation algo-

rithm for red-blue separation problem improving previous result $O(logn)$.

Inspired by this success, Mitchell [76] extended guillotine subdivision to m-guillotine subdivision, a rectangular polygonal subdivision such that there exists a cut whose intersection with the subdivision edges consists of a small number ($O(m)$) of connected components and the subdivisions on either side of the cut are also m-guillotine. With a minor change of the proof of [77], Mitchell established a PTAS for minimum length rectangular partition problem. Mitchell [78,79] further extended this m-guillotine subdivision technique to other geometric optimization problems, including Euclidean and rectilinear Steiner tree problems, and obtained PTAS for them.

Variations of Steiner Trees

Successful researches on classical Steiner tree problems encourage extensive study on variations of Steiner trees with various application backgrounds. Currently, they form a quite active research direction in Steiner trees.

In VLSI design, one considers several sets of terminals and finds a minimum total length packing of Steiner trees for these sets under the following situation [82]: The edges of the Steiner trees are required to lie in channels between cells. Each channel has a capacity which tells at most how many edges can run through it.

A complicated computer network usually consists of several nets of different speeds. The following problem was proposed based on such a back- ground: Consider an undirected network with multiple edge weights $(c_1(e), \ldots, c_k(e))$ $(c_1(e) > \cdots > c_k(e))$. Given a subset N of vertices and a partition $\{N_1, \ldots, N_k\}$ of N with $|N_1| \geq 2$, find a subnetwork interconnecting N with minimum total weight such that the length of any edge e on a path between a pair of vertices in N_j is at least $c_j(e)$ [43,57].

To construct roads of minimum total length to interconnect n highways under the constraint that the roads can intersect each highway only at one point in a designated interval which is a line-segment, a generalization of Euclidean Steiner trees has been proposed and studied. Du, Hwang, and Xue [35] presented a set of optimality conditions for the problem and showed how to construct a solution to meet this set of optimality conditions.

Constructing phylogenetic trees is an important topic in computer biology. One of formulations is as follows: For a fixed alphabet A, let d denote the Hamming distance on A^n, i. e. $d((a_1, \ldots, a_n), (b_1, \ldots, b_n))$ equals the number of indices i such that $a_i \neq b_i$. Given a set P of points in the metric space (A^n, d), find a Steiner minimum tree for P. This problem is known to be NP-hard. (See [47].)

When a new customer is out of original telephone network, the company has to build a new line to connect the customer into the network. This situation brings us an on-line Steiner tree problem as follow: Assume that a sequence of points in a metric space are given step by step. In the ith step, only locations of the first n_i points in the sequence are known. The problem is to construct a shorter network at each step based on the network constructed in previous steps. The study of on-line problems was initiated by [89] and [73]. A criterion for the performance of an on-line algorithm is to compare the solution generated by the on-line algorithm with the solution of corresponding off-line problem. In the Euclidean plane, it has been known that the worst-case ratio of lengths between on-line solution and off-line solution is between $O(n \log n / \log \log n)$ and $O(n \log n)$ [2,96,99].

Listing all variations and reviewing each of them would take tremendous time and space. It should not be the purpose of this short article. Therefore, we next review a few for which some significant results have been recently obtained.

Steiner Arborescence

Given a weighted directed graph G, a vertex r, and a subset P of n vertices, a *Steiner arborescence* is a directed tree with root r such that for each $x \in P$ there exists a path from r to x. The shortest Steiner arborescence is also called a *minimum Steiner arborescence*. Computing minimum Steiner arborescence is an NP-hard problem. Also, one knows that if $NP \neq P$, then the best possible performance ratio of polynomial time approximation for this problem is $O(\log n)$. This means that although, like the minimum spanning tree, the minimum arborescence as a shortest arborescence tree without Steiner points can be computed in polynomial time, the Steiner ratio (the maximum lower bound for the ratio of lengths between the minimum Steiner arbores-

cence and the minimum arborescence for the same set of given points) in directed graphs is zero. Z. Dai and others [20,28] applied Arora's techniques to this problem and obtained the best known result that for any $\varepsilon > 0$ there exists a polynomial time approximation with performance ratio $O(n^{\varepsilon})$. An open problem (as of 2000) remains for closing the gap between the lower bound and the upper bound for the performance ratio.

A version of this problem in the rectilinear plane has a great interest in VLSI designs and an interesting story in the literature. Given a set P of n points in the first quadrant of the rectilinear plane, a *rectilinear Steiner arborescence tree* is a directed tree rooted at the origin, consisting of all paths from the root to points in P with horizontal edges oriented in left-to-right direction and vertical edges oriented in bottom-up direction. What is the complexity of computing the minimum rectilinear arborescence? First, it was claimed that a polynomial time algorithm was found. However, S.K. Rao, P. Sadayappan, Hwang, and P.W. Shor [84] found a serious flow in this algorithm. Although they could not show the *NP*-completeness of the problem, they pointed out the difficulties of computing the minimum rectilinear arborescence in polynomial time. They also showed that while the ratio of lengths between a minimum arbores- cence tree and a minimum Steiner tree for the same set of points tends to infinity, there is a polynomial time approximation with performance two. Recently (2000), W. Shi and C. Su [88] showed that computing the minimum rectilinear arborescence is *NP*-hard. B. Lu and L. Ruan [72] showed, by employing Arora's techniques, that there is a polynomial time approximation scheme for the problem.

Edge-length and Number of Steiner Points

In wavelength-division multiplexing (WDM) optical network design [68,83], suppose we need to connect n sites located at $p_1, \ldots, p_n$ with WDM optical network. Due to the limit in transmission power, signals can only travel a limited distance (say R) for guaranteed correct transmission. If some of the intersite distances are greater than R, we need to provide some amplifiers or receivers/transmitters at some locations in order to break it into shorter pieces. This situation requires us to consider the problem of minimizing the maximum edge-length and the number of *Steiner points* in design of WDM optical network. To do so, two variations of Steiner trees have been studied.

The first is to minimize the number of Steiner points under upper bound for edge-length. That is, given a set of n terminals $X = \{p_1, \ldots, p_n\}$ in the Euclidean plane R^2, and a positive constant R, the problem is to compute a tree T spanning a superset of X such that each edge in the tree has a length no more than R and with the minimum number $C(T)$ of points other than those in X, called *Steiner points*. This problem is called *Steiner tree problem with minimum number of Steiner points*, denoted by *STP-MSP* for short. G.-L. Lin and G.H. Xue [69] showed that the STP-MSP problem is *NP*-hard. They also showed that the approximation obtained from the minimum spanning tree by simply breaking each edge into small pieces within the upper bound (called steinerized spanning tree) has a worst-case performance ratio at most five. D. Chen and others [21] showed that this approximation has a performance ratio exactly four. They also presented a new polynomial time approximation with a performance ratio at most three and a polynomial time approximation scheme under certain conditions. Lu and others [71] studied the STP-MSP in rectilinear plane. They showed that in the rectilinear plane, the steinerized spanning tree has performance ratio exactly three and there exists a polynomial time approximation two.

The second is to minimize the maximum edge-length under an upper bound on the number of Steiner points. That is, given a set $P = \{p_1, \ldots, p_n\}$ of n terminals and an positive integer k, we want to find a Steiner tree with at most k Steiner points such that the length of the longest edges in the tree is minimized. This is one of the bottleneck Steiner tree problems. Wang and Du [98] showed that:

a) if $NP \neq P$, then the performance ratio of any polynomial time approximation for the problem in the Euclidean plane is at least $\sqrt{2}$;
b) if $NP \neq P$, then the performance ratio of any polynomial time approximation for the problem in the rectilinear plane is at least two;
c) there exists a polynomial time approximation with performance ratio two for the problem in both rectilinear and Euclidean planes.

Multiphase

Given an edge-weighed complete graph with vertex set X ($|X| = n$) and subsets $X_1, \ldots, X_m$ of vertices, the problem is to find a minimum weighed subgraph G such that for every $i = 1, \ldots, m$, G contains a spanning tree for X_i. This problem is called *subset interconnection designs* or *multiphase spanning network* problem [29,37]. Du and others [41] showed that if $NP \neq P$, then the best performance ratio of polynomial time approximation for this problem is $\ln n + O(1)$.

Given an edge-weighed graph B with vertex set X and subsets $X_1, Y_1, \ldots, X_m, Y_m$ of X with $X_i \cap Y_i = \emptyset$, the problem is to find a minimum weighed subgraph G such that for every $i = 1, \ldots, m$, G contains a Steiner tree for X_i without using vertices not in Y_i. This problem is called *multiphase Steiner network* problem. Both multiphase spanning network and Steiner network problems arose in communication network design [81] and vacuum system design [37]. For the former one, when the solution is a forest, the system $(X_1, \ldots, X_m)$ is called *subtree hypergraph*. Such a system has various applications in computer database schemes [7] and statistics. It is also related to chordal graphs [42,45]. R.E. Tarjan and M. Yannakakis [95] gave a $O(m+n)$-time algorithm to tell whether a set system is a subtree hypergraph or not.

Comparing the phylogenetic tree problem with multiphase Steiner network problem, we would find some similarities between them if we look at each coordinate like a phase. For multiphase Steiner tree problem, if the solution is a tree, then we have either a good heuristic or a polynomial time computable exact solution [37]. This suggests that studying the relationship between the two problems will hopefully find a new construction of phylogenetic trees.

L. Ruan and others [85] found that multiweight Steine tree problem can be transformed to multiphase Steiner tree problem. This initiates new line to study both problems.

See also

- ▶ Auction Algorithms
- ▶ Bottleneck Steiner Tree Problems
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Alfaro M, Conger M, Hodges K, Levy A, Kochar R, Kuklinski L, Mahmood Z, von Haam K (1991) The structure of singularities of phi;-minimizing networks in R2. Pacific J Math 149:201–210
2. Alon N, Azar Y (1993) On-line Steiner trees in the Euclidean plane. Discrete Comput Geom 10(2):113–121
3. Arora S More efficient approximation schemes for Euclidean TSP and other geometric problems
4. Arora S (1996) Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In: Proc 37th FOCS, pp 2–12
5. Arora V, Santosh V, Saran H, Vazirani V (1994) A limited-backtrack greedy schema for approximation algorithms. Manuscript
6. Beasley JE (1989) A heuristic for the Euclidean and rectilinear Steiner problems. Techn Report Managem School Imperial College London
7. Beeri C, Fagin R, Maier D, Yannakakis M (1983) On the desirability of acyclic database schemes. J ACM 30:479–513
8. Berman P, Fössmeier U, Karpinski M, Kaufmann M, Zelikovsky A (1994) Approaching the 5/4-approximations for rectilinear Steiner trees. Lecture Notes Computer Sci, vol 855. Springer, Berlin, pp 60–71
9. Berman P, Ramaiyer V (1994) Improved approximations for the Steiner tree problem. J Algorithms 17:381–408
10. Bern M (1986) Two probabilistic results on rectilinear Steiner trees. In: Proc 18th STOC, pp 433–441
11. Bern MW, Graham RL (1988) The shortest-network problem. Scientif Amer jan.:84–89
12. Bern M, Plassmann P (1989) The Steiner problem with edge lengths 1 and 2. Inform Process Lett 32:171–176
13. Bharath-Kumar K, Jaffe JM (1983) Routing to multiple destinations in computer networks. IEEE Trans Communications COM-31:343–351

14. Booth RS (1991) Analytic formulas for full Steiner trees. Discrete Comput Geom 6(1):69–82
15. Borchers A, Du D-Z (1995) The k-Steiner ratio in graphs. Proc. 27th ACM Symp. Theory of Computing,
16. Borchers A, Du D-Z, Gao B, Wan P-J (1998) The k-Steiner ratio in the rectilinear plane. J Algorithms 29:1–17
17. Cavalli-Sforza LL, Edwards AW (1967) Phylogenetic analysis: models and estimation procedures. Amer J Human Genetics 19:233–257
18. Chang S-K (1972) The design of network configurations with linear or piecewise linear cost functions. In: Symp Computer-Communications, Networks, and Teletraffic, pp 363–369
19. Chang S-K (1972) The generation of minimal trees with a Steiner topology. J ACM 19:699–711
20. Charikar M, Chekuri C, Cheung TY, Dai A, Goel A, Guha S, Li M (1998) Approximation algorithms for directed Steiner problems. In: Proc. 9th SODA, pp 199–209
21. Chen D, Du D-Z, Hu X, Lin G-H, Wang L, Xue G Approximations for Steiner trees with minimum number of Steiner points. J Global Optim (accepted)
22. Chung FRK, Gilbert EN (1976) Steiner trees for the regular simplex. Bull Inst Math Acad Sinica 4:313–325
23. Chung FRK, Graham RL (1985) A new bound for Euclidean Steiner minimum trees. Ann New York Acad Sci 440:328–346
24. Chung FRK, Hwang FK (1978) A lower bound for the Steiner tree problem. SIAM J Appl Math 34:27–36
25. Cieslik D (1990) The Steiner ratio of Banach-Minkowski planes. In: Bodendiek R (ed) Contemporary Methods in Graph Theory. BI Wissenschaftsverlag, Mannheim, pp 231–248
26. Cieslik D (1998) Steiner minimal trees. Kluwer, Dordrecht
27. Courant R, Robbins H (1941) What is mathematics? Oxford Univ. Press, Oxford
28. Dai Z (1998) Algorithm design and analysis for the Steiner and p-median problems. PhD Thesis Dept. Computer Sci. City Univ. Hong Kong
29. Du D-Z (1986) An optimization problem. Discrete Appl Math 14:101–104
30. Du D-Z, Gao B, Graham RL, Liu Z-C, Wan P-J (1993) Minimum Steiner trees in normed planes. Discrete Comput Geom 9:351–370
31. Du D-Z, Hwang FK (1983) A new bound for the Steiner ratio. Trans Amer Math Soc 278:137–148
32. Du D-Z, Hwang FK (1990) An approach for proving lower bounds: solution of Gilbert–Pollak's conjecture on Steiner ratio. Proc 31th FOCS:76–85
33. Du D-Z, Hwang FK (1990) The Steiner ratio conjecture of Gilbert–Pollak is true. Proc Nat Acad Sci USA 87:9464–9466
34. Du D-Z, Hwang FK (1992) Reducing the Steiner Problem in a normed space with a d-dimensional polytope as its unit sphere. SIAM J Comput 21:1001–1007
35. Du D-Z, Hwang FK, Xue G (1999) Intersecting highways. SIAM Discret Math 12:252–261
36. Du D-Z, Hwang FK, Yao EY (1985) The Steiner ratio conjecture is true for five points. J Combin Th A 38:230–240
37. Du D-Z, Miller Z (1988) Matroids and subset interconnection design. SIAM J Discret Math 1:416–424
38. Du D-Z, Pan L-Q, Shing M-T (1986) Minimum edge length guillotine rectangular partition. Report Math Sci Res Inst Univ California 02418–86
39. Du D-Z, Yao EN, Hwang FK (1982) A short proof of a result of Pollak on Steiner minimal trees. J Combin Th A 32:396–400
40. Du D-Z, Zhang Y, Feng Q (1991) On better heuristic for Euclidean Steiner minimum trees. Proc 32nd FOCS
41. Du X, Wu W, Kelley D (1998) Approximations for subset interconnection designs. In memoriam of Ronald V. Book. Theoret Computer Sci 207(1):171–180
42. Duchet P (1978) Propriété Helly et problèmes de répresentation. In: Colloque Internat Paris–Orsay, 260, pp 117–118
43. Duin C, Volgenant T (1991) The multi-weighted Steiner tree problem. Ann Oper Res 33:451–469
44. El-Arbi C (1978) Une heuristique pour le probleme de l'arbre de Steiner. RAIRO 12:207–212
45. Flament C (1978) Hypergraphes arbores. Discret Math 21:223–227
46. Fössmeier U, Kaufman M, Zelikovsky A (1993) Fast approximation algorithms for the rectilinear Steiner tree problem. Lecture Notes Computer Sci, vol 762. Springer, Berlin, pp 533–542
47. Foulds LR, Graham RL (1982) The Steiner problem in phylogeny is NP-complete. Adv Appl Math 3:43–49
48. Friedel J, Widmayer P (1989) A simple proof of the Steiner ratio conjecture for five points. SIAM J Appl Math 49:960–967
49. Gao B, Du D-Z, Graham RL (1995) A tight lower bound for the Steiner ratio in Minkowski planes. Discret Math 142:49–63
50. Garey MR, Graham RL, Johnson DS (1977) The complexity of computing Steiner minimal trees. SIAM J Appl Math 32:835–859
51. Georgakopoulos G, Papadimitriou CH (1987) The 1-Steiner tree problem. J Algorithms 8:122–130
52. Gilbert EN, Pollak HO (1968) Steiner minimal trees. SIAM J Appl Math 16:1–29
53. Graham RL, Hwang FK (1976) Remarks on Steiner minimal trees. Bull Inst Math Acad Sinica 4:177–182
54. Hwang FK (1976) On Steiner minimal trees with rectilinear distance. SIAM J Appl Math 30:104–114
55. Hwang FK, Richards DS, Winter P (1992) Steiner tree problems. North-Holland, Amsterdam
56. Hwang FK, Yao YC (1990) Comments on Bern's probabilistic results on rectilinear Steiner trees. Algorithmica 5:591–598

57. Iwainsky A (1985) Optimal trees-a short overview on problem formulations. In: Iwainsky A (ed) Optimization of Connections Structures in Graphs, CICIP, East Berlin, GDR., pp 121–133
58. Jiang T, Lawler EL, Wang L (1994) Aligning sequences via an evolutionary tree: complexity and approximation. In: Proc. 26th STOC
59. Jiang T, Wang L (1994) An approximation scheme for some Steiner tree problems in the plane. Lecture Notes Computer Sci, vol 834. Springer, Berlin, pp 414–422
60. Kahng A, Robins G (1990) A new family of Steiner tree heuristics with good performance: the iterated 1-Steiner approach. Techn Report UCLA CS Dept
61. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computation. Plenum, New York, pp 85–103
62. Karpinski M, Zelikovsky AZ (1997) New approximation algorithms for Steiner tree problems. J Combin Optim 1:47–65
63. Korthonen P (1979) An algorithm for transforming a spanning tree into a Steiner tree. In: Survey of Math. Programming, vol 2. North Holland, Amsterdam, pp 349–357
64. Kou L, Makki K (1987) An even faster approximation algorithm for the Steiner tree problem in graph. Congressus Numerantium 59:147–154
65. Kou L, Markowsky G, Berman L (1981) A fast algorithm for Steiner trees. Acta Informatica 15:141–145
66. Kuhn HW (1975) Steiner's problem revisited. In: Dantzig GB, Eaves BC (eds) Studies in Optimization. Math. Assoc. Amer., Washington, DC, pp 98–107
67. Levin AJ (1971) Algorithm for the shortest connection of a group of vertices. Soviet Math Dokl 12:1477–1481
68. Li C-S, Tong FF, Georgiou CJ, Chen M (1994) Gain equalization in metropolitan and wide area optical networks using optical amplifiers. Proc IEEE INFOCOM'94 June, pp 130–137
69. Lin G-H, Xue GL (1999) Steiner tree problem with minimum number of Steiner points and bounded edge-length. Inform Process Lett 69:53–57
70. Liu ZC, Du DZ (1991) On Steiner minimal trees with Lp distance. Algorithmica
71. Lu B, Gu J, Hu X, Shragowitz E Wire segmenting for bd-Buffer insertion based on RSTP-MSP. Theoret Comput Sci (accepted).
72. Lu B, Ruan L Polynomial-time approximation scheme for rectilinear Steiner arborescence problem. J Combin Optim (accepted).
73. Manase MS, McGeoch LA, Sleator DD (1988) Competitive algorithms for on-line problems. In: Proc. 20th STOC, Chicago,
74. Mata C (1995) A constant factor approximation algorithm for the red-blue separation problem. Techn Report Dept Computer Sci SUNY
75. Melzak ZA (1961) On the problem of Steiner. Canad Math Bull 4:143–148
76. Mitchell JSB Guillotine subdivisions approximate polygonal subdivisions: Part II-A simple polynomial-time approximation scheme for geometric k-MST, TSP, and related problems. SIAM J Comput
77. Mitchell JSB (1996) Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k-MST problem. In: Proc. 7th ACM-SIAM Sympos. Discrete Algorithms, pp 402–408
78. Mitchell JSB (1997) Guillotine subdivisions approximate polygonal subdivisions: Part III-Faster polynomial-time approximation scheme for geometric network optimization. In: Proc. Ninth Canadian Conf. Computational Geometry, pp 229–232
79. Mitchell JSB, Blum A, Chalasani P, Vempala S A constant-factor approximation for the geometric k-MST problem in the plane. SIAM J Comput
80. Pollak HO (1978) Some remarks on the Steiner problem. J Combin Th A 24:278–295
81. Prisner E (1992) Two algorithms for the subset interconnection design. Networks 22:385–395
82. Pulleyblank WR Two Steiner tree packing problems. 27th STOC 383–387
83. Ramamurthy B, Iness J, Mukherjee B (1997) Minimizing the number of optical amplifiers needed to support a multi-wavelength optical LAN/MAN. In: Proc. IEEE INFOCOM'97, Apr, pp 261–268
84. Rao SK, Sadayappan P, Hwang FK, Shor PW (1992) The rectilinear Steiner arborescence problem. Algorithmica 7:277–288
85. Ruan L, Lu B Approximations for multi-weight Steiner trees. Manuscript
86. Rubinstein JH, Thomas DA (1989) A variational approach to the Steiner network problem. In: Proc NATO Workshop Topological Networks, Copenhagen, Denmark,
87. Rubinstein JH, Thomas DA (1991) The Steiner ratio conjecture for six points. J Combin Th A 58:54–77
88. Shi W, Su C (2000) The rectilinear Steiner arboresence problem is NP-complete. In: Proc. SODA
89. Sleator DD, Tarjan RE (1985) Amortized efficiency of list update and paging rules. Comm ACM 28:202–208
90. Smith JM, Lee DT, Liebman JS (1981) An O(N log N) heuristic for Steiner minimal tree problems in the Euclidean metric. Networks 11:23–39
91. Smith JM, Liebman JS (1979) Steiner trees, Steiner circuits and interference problem in building design. Eng Optim 4:15–36
92. Smith WD (1992) How to find Steiner minimal trees in Euclidean d-space. Algorithmica 7:137–177
93. Stewart I (1991) Trees, telephones and tiles. New Scientist 16:26–29
94. Suzuki A, Iri M (1986) A heuristic method for Euclidean Steiner problem as a geographical optimization problem. Asia–Pacific J Oper Res 3:106–122
95. Tarjan RE, Yannakakis M (1984) Simple linear-time algorithms to test chordality of graphs, test acyclicity of hy-

pergraphs, and selectively reduce acyclic hypergraphs. SIAM J Comput 13:566–579
96. Tsai YT, Tang CY, Chen YY (1996) An average case analysis of a greedy algorithm for the on-line Steiner tree problem. Comput Math Appl 31(11):121–131
97. Wan P-J, Du D-Z, Graham RL (1997) The Steiner ratio on the dual normed plane. Discret Math 171:261–275
98. Wang L, Du D-Z Approximations for a bottleneck Steiner tree problem
99. Westbrook J, Yan DCK (1995) The performance of greedy algorithms for the on-line Steiner tree and related problems. Math Syst Theory 28(5):451–468
100. Wu YF, Widmayer P, Wong CK (1986) A faster approximation algorithm for the Steiner problem in graphs. Acta Informatica 23:223–229
101. Zelikovsky AZ (1992) An 11/8-approximation algorithm for the Steiner problem on networks with rectilinear distance. Coll Math Soc J Bolyai 60:733–745
102. Zelikovsky AZ (1993) The 11/6-approximation algorithm for the Steiner problem on networks. Algorithmica 9:463–470
103. Zelikovsky AZ (1993) A fast approximation algorithm for the Steiner tree problem in graphs. Inform Process Lett 46:79–83
104. Zelikovsky AZ (1995) Better approximation bounds for the network and Euclidean Steiner tree problem. Manuscript

Stochastic Bilevel Programs

SBP

LAURA WYNTER
PRISM, Computer Sci. Department, University Versailles, Versailles-Cedex, France

MSC2000: 90C15, 90C26, 90C33

Article Outline

Keywords
Principal Features of Bilevel Programs
Examples of Bilevel Programs
Properties of the Stochastic Bilevel Program
Algorithms for Stochastic Bilevel Programs
See also
References

Keywords

Stochastic programming; Mathematical program with equilibrium constraints; Stackelberg game; Hierarchical optimization; Variational inequality problem; Nondifferentiable nonconvex program

A stochastic bilevel program (SBP) is a generalization of an ordinary bilevel program (BP; cf. ► Bilevel programming: Introduction, history and overview), which allows the uncertainty in the values of the problem parameters to be expressed by a probability distribution on some or all of the variables of the model. The introduction of these random variables in the BP modifies some of its properties, both its mathematical properties, as well as the resolution time needed to find a satisfactory solution.

Nevertheless, a stochastic BP is essentially a bilevel program, and it is therefore important to summarize the essential features of this class of models before moving on to the effects of incorporating uncertainty.

Principal Features of Bilevel Programs

Bilevel programs are optimization problems with two objectives, one at each level, which interact through the sharing of some of the problem variables. BP can be perhaps most easily understood through comparison with the leader-follower (or Stackelberg) paradigm in game theory. In this context, the leader is represented through the upper level optimization problem. In particular, the leader seeks to optimize a function of two vectors, one which he explicitly controls, x, and another vector which describes the reactions of the followers to his actions. These reactions are described by y. Since the reactions of the followers have an impact on the objective of the leader, the leader's optimization problem is to solve $\min f(x, y)$, subject to $x \in \mathcal{X}$, $y \in \mathcal{Y}$. Note that the leader may be subject to constraints on his action, given by $\mathcal{X}$. In addition, in some cases, there is more than one possible reaction y from the followers in response to a given x; if this happens, the leader may take different strategies for accepting a single y when he decides on the optimal x, or y will be restricted to a set, $\mathcal{Y}$.

Just as the leader seeks to minimize a function, $f(x, y)$, the followers' behavior is also described by an optimization process, $t(x, y)$. In this case, the followers accept the leader's decision, x, as a parameter, and seek to optimize their objective over y, subject to some constraints, described by the set $\mathcal{Z}(x)$, which may also depend on the parameter x. That is, the lower level, fol-

lowers' problem is given by min $t(x, y)$, $y \in \mathcal{Z}(x)$, which is clearly a parametric program with x as a parameter.

In summary, then, BP is expressed by a pair of optimization programs, coupled through the passing of an upper-level variable as a parameter to the lower level problem. One can express quite generally the vector y as a possibly nonunique solution to the lower-level problem, whose (parametric) solution set is given by $S(x)$:

$$\begin{cases} \min\limits_{\substack{x\in X\\ y\in Y}} & f(x, y) \\ \text{s.t.} & y \in S(x), \end{cases}$$

where $S(x) = \{y\colon\ y \in \operatorname{argmin}_{z\in\mathcal{Z}(x)} t(x, z)\}$.

The definition of BP in terms of a generic, lower-level parametric solution set $S(x)$ allows us to introduce a further generality into the model: in many interesting applications, the lower-level, or followers' behavior, cannot be expressed as an optimization problem, but can be described by an *equilibrium process*, which is given mathematically by a *variational inequality problem* (VIP). That is, $S(x) = \{y \in \mathcal{Z}(x)\colon\ T(x, y)^\top(y - \widehat{y}) \le 0,\ \widehat{y} \in \mathcal{Z}(x)\}$. These bilevel programs are often referred to as *mathematical programs with equilibrium constraints*, (MPEC). In fact, MPEC can be considered as a more general form of BP, since any optimization problem can be expressed as a VIP, but the converse is true only when $T(x, \cdot) = \nabla_y t(x, \cdot)$.

Bilevel programs have a number of mathematical and computational particularities with respect to standard one-level optimization programs. The most striking characteristic of BP is that the upper-level function is not differentiable, even in the case where the lower-level response vector, y, is unique as a function of x, that is $y = S(x)$. Indeed, y is an implicit function in terms of x. A further observation of BP shows that, for each evaluation of $f(x, y)$, it is necessary to solve the lower-level problem, just to obtain an iterate for y. The computational complexity of BP is thus increased dramatically, since each iteration in the resolution of f requires the resolution of t.

Examples of Bilevel Programs

While the leader-follower paradigm is useful for illustrating the hierarchical nature of the two levels in a bilevel program, it does not provide a sufficient scope of the range of problems included in the class of bilevel programs. Indeed, bilevel programs describe problems in many areas of engineering and management, as well as problems in game theory. Following are a few examples of bilevel programs, which will furthermore help illustrate how uncertainty can be explicitly taken into account. (See also [10].)

Example 1 (Optimal pricing problem) In a number of application areas, especially the *transportation* and *telecommunications* sectors, a central operator seeks to maximize profit, given that the market that he is targeting is competitive, and furthermore that his potential clients can refuse to participate should the price be set too high, or service quality too low. These problems have an inherent bilevel form. Determining the optimal tolls to set on a highway, or the price of a long-distance phone service are problems of this type.

The upper-level describes the manager's problem: it may consist in determining the prices so as to achieve profit maximization, or the problem of determining the level of service to offer on the infrastructure which optimizes a given performance criterion, perhaps taking into account the cost of offering such a service level. The policy instrument, x, is then the price to be set by the manager for use of the infrastructure, and/or the service level of the infrastructure (capacity, travel time, or access time improvements, etc.). The feasible set $\mathcal{X}$ generally contains bounds on the possible values of x. When the manager seeks to determine the optimal level of capacity improvements to an existing infrastructure, the resulting bilevel problem is known as the *network design problem*.

The lower-level problem describes the users' responses to the prices and/or service levels set by the manager. The users' response is given by the level of use on each link of the infrastructure, y. In general, one assumes that the users' behavior follows an equilibrium principle, given by an cost operator (or utility function), $T(x, y)$. The cost operator gives the cost of using the infrastructure as a function of the price vector x and the amount of use y. The interpretation of (Nash) *equilibrium* is the following: A usage pattern y is in equilibrium (stable) if no single user can reduce his own cost (or increase his utility) by modifying his current usage pattern. The lower-level feasible set, $\mathcal{Z}(x)$ will then include demand satisfaction constraints (where the total demand level may be a function of the price or service

level, x) and may include capacity constraints on the infrastructure.

The deterministic bilevel pricing model is then given by:

$$\begin{cases} \max\limits_{x \in X} & f(x, y), \\ \text{s.t.} & y \in S(x), \end{cases} \tag{1}$$

where $S(x) := \{y \in Z(x) \colon T(x, y)^\top (y - \widehat{y}) \leq 0, \widehat{y} \in Z(x)\}$.

Permitting the incorporation of random variables into the optimal infrastructure pricing model can be quite useful for increasing the realism of the model. In particular, the demand for the use of the infrastructure can be estimated through historical data or surveys, but generally with some random error. In this case, demand can be expressed as a random variable, and then $Z(x) := Z(x, \omega)$, where ω belongs to a probability space $(\Omega, \mathcal{A}, \mathcal{P})$. Similarly, the maximum usage level of the infrastructure, that is, the link capacities included in $Z(x)$, can depend on a number of factors which cannot be described precisely, but whose effect is to modify upper usage limits according to a known probabilistic law. In addition, the user's cost function on the infrastructure, $T(x, y) := T(x, y, \omega)$ can vary according to a known distribution.

Example 2 (Stackelberg–Nash equilibrium) The game theoretic model of a *Stackelberg–Nash* (or *Stackelberg–Nash–Cournot*) equilibrium permits representing a number of important market phenomena. The model assumes a market in which N firms produce a single good, each competing for maximum market share. In addition, there is a single firm (or government) that also produces the good, but is capable of reacting to the N other firms' production when determining how to set its own production level.

This paradigm is successfully applied, for example, to utility markets, in which both private and public firms compete to sell the same utility; power generation is one such example. The upper-level optimization problem, $\max_x f(x, y)$ represents the leader's profit function, and x the leader's production level. The lower-level represents the Nash equilibrium problem among the N firms, known as the followers, with $y := y(x)$ being the followers' production levels, given the leader's production decision. Each follower then solves $\max_{y_i} t_i(x, y)$; and the equilibrium of the noncooperative Nash game can therefore be expressed as a VIP, as in the example above.

The incorporation of uncertainty in the Stackelberg–Nash model would then take the form of a randomly varying demand from the market, and possibly uncertainty in the profit functions themselves.

Example 3 (Structural optimization) Among the large number of potential applications of bilevel optimization in engineering, one which has been the subject of much research attention is that of finding the design of a mechanical structure which has the best performance under the influence of external forces: structural optimization.

As is the case in the examples above, *structural optimization* problems also have an inherent bilevel form. The upper level objective function $f(x, y)$ measures some characteristic of the structure, such as its construction cost, weight, or stiffness, with y representing the performance measure. This objective function is optimized by selecting design parameters, x, which express the shape of the structure, and the choice and amount of material to be used in the design. In addition, the structure may be subject to behavioral constraints within the upper-level problem, such as bounds on the displacements, stresses and contact forces. These constraints define the feasible region $\mathcal{Y}$. Budget limits on the amount of available material, if present, would be included in a set $\mathcal{X}$.

The lower-level problem describes the behavior of the structure given the choices of the design variables, possible contact conditions with foundations or boundaries, (the set $\mathcal{Z}$), and the external forces acting on it, F. For elastic structures, the behavior is given by the equilibrium law of minimal potential energy, $\Pi(x, y)$, which determines the values of the (lower-level) state variables, that is, the displacements, stresses and contact forces, y. The matrix $K(x)$ represents the stiffness of the material, and is symmetric and positive semidefinite.

The deterministic structural optimization problem thus described then is:

$$\begin{cases} \min\limits_{\substack{x \in X \\ y \in \mathcal{Y}}} & f(x, y), \\ \text{s.t.} & y \in \operatorname{argmin}_{z \in Z} \Pi(x, z), \end{cases} \tag{2}$$

where $\Pi(x, y) := (1/2) y^\top K(x) y - F^\top y$.

The introduction of uncertainty into the structural optimization model can be of great use in permitting the modeler to take into account variations in external forces (due for example to wind and other weather conditions, varying traffic on a bridge, etc.) and variations in the material properties. In the first case, the variation in external forces can be described by setting $F = F(\omega)$, where $\omega \in \Omega$. Note that a failure to take this variability into account may lead to structures with unwanted vibration under certain weather conditions, as the topology optimization will assign bars only where needed to sustain the described forces; hence random forces left out of the optimization will not be accounted for by the resulting structure. Note further that the structure that will result from this stochastic bilevel optimization will be one that responds best 'on average' to the range of possible forces. In certain cases, where any failure is unacceptable (e. g., when designing bridges), it may be preferable to use a *worst-case approach*, which will result in more costly designs that minimize however the risk of failure (rather than minimizing cost or some other characteristic of the structure).

Uncertainty in other problem parameters can be accommodated in a similar manner. Taking into account the variability in material properties, one would set $K(x) = K(x, \omega)$.

Properties of the Stochastic Bilevel Program

The presence of random variables in the bilevel program means that one can no longer calculate exactly the vectors x and y, since their values depend on parameters which vary randomly. Instead, one can calculate the values of x and y that optimize f on average; the objective is then to minimize the expected value of $f(x, y, \omega)$. The stochastic bilevel program is defined below, with the more general lower-level VIP:

$$\text{SBP} \quad \begin{cases} \min & \mathsf{E}_\omega \left[f(x, y, \omega) \right] \\ \text{s.t.} & x \in \mathcal{X}, \\ & y(\omega) \in S(x, \omega), \quad \omega \in \Omega, \end{cases}$$

where $S(x, \omega) := \left\{ y \in \mathcal{Z}(x, \omega) \colon T(x, y, \omega)^\top (y - \widehat{y}) \leq 0, \widehat{y} \in \mathcal{Z}(x, \omega) \right\}$ denotes the set of solutions to the lower-level variational inequality defined by the parameterized mapping $T(x, \cdot, \omega)$ and feasible set $\mathcal{Z}(x, \omega)$ (presumed convex). The random variable ω is defined on a probability space $(\Omega, \mathcal{A}, \mathcal{P})$.

In its general form, the objective function of the stochastic bilevel program can be a multiple integral when ω is a vector of continuous random variables. That is, $\mathsf{E}[f(x, y, \omega)] := \int_\Omega f(x, y, \omega) dF(\omega)$. However, this integral is in most cases very difficult to evaluate. For that reason, as is the case in the majority of stochastic programs at this time, a discretization of the random distributions is used: one lets $\mathcal{L}$ represent the discrete set of random observations obtained from Ω, numbered $\ell = 1, \ldots, |\mathcal{L}|$, and ρ_ℓ the probability of each scenario $\ell \in \mathcal{L}$, with $\sum_{\ell \in \mathcal{L}} \rho_\ell = 1$. This allows one to express the expected value in the objective function as a sum: $\mathsf{E}[f(x, y_\ell)] := \sum_{\ell \in \mathcal{L}} \rho_\ell f(x, y_\ell)$. The resulting problem is referred to as SBP-$\mathcal{L}$.

In what follows, it is assumed that the set of random observations has been expressed as a discrete set. (For information on the additional assumptions needed in the case of a continuous distribution, see [1,12,16]).

Consider the following assumptions:

i) $\mathcal{X}$ is nonempty and closed.

ii) The lower-level constraint set is of the form $\mathcal{Z}_\ell(x) := \{y \colon g^i_\ell(x, y) \leq 0, i = 1, \ldots, k\}$, $\ell \in \mathcal{L}$, where each function g^i_ℓ is continuous and convex in y for each $x \in \mathcal{X}$. Further, either $g^i_\ell(x, \cdot) = g^i_\ell(\cdot)$, $i = 1, \ldots, k$, $\ell \in \mathcal{L}$, that is, $\mathcal{Z}_\ell(x) = \mathcal{Z}_\ell$, or for each $x \in \mathcal{X}$, $\ell \in \mathcal{L}$, there is a y such that $g^i_\ell(x, y) < 0$, $i = 1, \ldots, k$.

iii) There exists an $(x, y_\ell) \in \mathcal{P}_\ell := \{(x, y) \in \text{gr } S_\ell \colon x \in \mathcal{X}\}$ with $f(x, y_\ell) < \infty$ for all $\ell \in \mathcal{L}$, where gr $S_\ell := \{(x, y) \colon y \in S_\ell\}$ is the graph of S_ℓ.

iv) (Inf-compactness) f is lower semicontinuous, proper, and has bounded level sets on $\cup_{\ell \in \mathcal{L}} \mathcal{P}_\ell$.

Perhaps the first property of interest is that of the existence of a solution.

Theorem 4 (Existence of optimal solutions to SBP-$\mathcal{L}$) *Let the assumption ii) hold, and the mapping T_ℓ be continuous. Then, the graph of S_ℓ, gr S_ℓ, is closed for each $\ell \in \mathcal{L}$. Hence, under the additional assumptions i), iii), and iv), there exists at least one optimal solution to SBP-$\mathcal{L}$.*

Proof If gr S_ℓ is closed for each $\ell \in \mathcal{L}$, then the assumptions imply the inf-compactness of the extended function $f + \delta_{\widehat{\mathcal{Z}}}$, where $\widehat{\mathcal{Z}} := \cup_{\ell \in \mathcal{L}} \mathcal{Z}_\ell$. From this, the existence of a solution follows from Weierstrass' theorem. But by condition ii), either $\mathcal{Z}_\ell(x) = \mathcal{Z}_\ell$, in which case the closedness of gr S_ℓ follows from the continuity

of T_ℓ, or the Slater condition holds, in which case the closedness of $\mathcal{Z}_\ell$ follows from [7, Lemma 1].

The conditions required in the preceding existence result are weaker than those of some previously considered requirements on bilevel model formulations, and as such, may be particularly interesting for a number of important applications. One such example can be found in (stochastic or deterministic) structural optimization problems; in this case, for example, it can be shown, that there exists an optimal solution even in the presence of zero design bounds. (See [2] for further details.)

A second property of interest is whether or not the problem is a *convex optimization problem*. In most cases, neither deterministic not stochastic bilevel problems will be convex. However, there is a special form of the stochastic bilevel program which may possess the desired convexity property.

Consider the following special case of SBP, in which the upper-level objective function f depends on the lower-level solution only in the sense of its optimal value. The deterministic form of this problem has been analyzed in great detail in [15], and is defined as follows:

$$\text{SBPOV} \quad \begin{cases} \min & \mathsf{E}_\omega \left[f(x, p(x,\omega)) \right], \\ \text{s.t.} & x \in \mathcal{X}, \\ & p(x,\omega) := \inf\limits_{y \in \mathcal{Z}(x,\omega)} t(x,y,\omega). \end{cases}$$

The discretized formulation, SBPOV-$\mathcal{L}$, of this special case is analogous to that of the general bilevel program.

Theorem 5 (Convexity of SBPOV-$\mathcal{L}$) *In addition to the assumptions i)–iv), assume, for each $\ell \in \mathcal{L}$, that t_ℓ is convex and continuous, and g^i_ℓ, $i = 1, \ldots, k$, are convex. Then, each function $p_\ell := \inf_{y \in \mathcal{Z}_\ell(x)} t_\ell(x,y)$ is convex on $\mathcal{X}$. Further, assume that $\mathcal{X}$ is convex, and that the function f is convex and increasing in its second argument. Then, the implicit upper-level objective function $x \mapsto \sum_{\ell \in \mathcal{L}} \rho_\ell f(x, p_\ell(x))$ is convex on $\mathcal{X}$, so that SBPOV-$\mathcal{L}$ is a convex problem.*

Proof One needs only to establish the convexity of p_ℓ on $\mathcal{X}$ for every $\ell \in \mathcal{L}$, but this result follows from the assumptions and [5, Thm. 5].

Note that two-stage stochastic programs are in fact equivalent to the problem SBPOV: two-stage stochastic linear programs (2S-SLP) are obtained as a particular form of this problem, that is, the *right-hand side perturbation model*, as discussed by [15, p. 189], in which the upper-level variable is located only on the right-hand side of the lower-level constraints. Consequently, the convexity of the two-stage stochastic programs can be obtained directly from the preceding result on SBPOV. (These and other such relations are explored in [12].)

A third property of interest for the resolution of the model concerns the differentiability of the upper-level objective function.

Let us first present some additional (and stronger) assumptions, that, among other things, will guarantee the uniqueness of the lower-level solution, y_ℓ for each $x \in \mathcal{X}$ and each $\ell \in \mathcal{L}$.

a) f is continuously differentiable.
b) The lower-level constraint set is of the form $\mathcal{Z}_\ell(x) := \{y: g^i_\ell(x,y) \le 0, i = 1, \ldots, k\}$, where each function g^i_ℓ is twice continuously differentiable and convex in y for each $x \in \mathcal{X}$, $\ell \in \mathcal{L}$. Furthermore, for each $x \in \mathcal{X}$, $\ell \in \mathcal{L}$, $\mathcal{Z}_\ell(x) \neq \emptyset$, and $\mathcal{Z}_\ell(x) \subset B_\ell$, for some open and bounded set B_ℓ.
c) Let $\mathcal{I}_\ell(x,y) := \{i = 1, \ldots, k: g^i_\ell(x,y) = 0\}$. Then, for each $x \in \mathcal{X}$, $\ell \in \mathcal{L}$, and $y \in S_\ell(x)$, the partial gradients $\nabla_y g^i_\ell(x,y)$, $i \in \mathcal{I}_\ell(x,y)$, are linearly independent.
d) T_ℓ is continuously differentiable and strongly monotone in y_ℓ for each $x \in \mathcal{X}$, $\ell \in \mathcal{L}$.

Theorem 6 (Directional differentiability of SBP-$\mathcal{L}$) *Suppose that SBP-$\mathcal{L}$ has a solution and let the assumptions a)–d) be satisfied. Then, the implicit function $x \mapsto \sum_{\ell \in \mathcal{L}} \rho_\ell f(x, y_\ell(x))$ of SBP-$\mathcal{L}$) is locally Lipschitz continuous and directionally differentiable on $\mathcal{X}$.*

Proof By [13, Thm. 2.1], the assumptions imply that the implicit mapping $x \longmapsto S_\ell(x)$ is locally Lipschitz continuous, for each $\ell \in \mathcal{L}$. Then, the result follows directly from [14].

Algorithms for Stochastic Bilevel Programs

The last essential ingredient needed in order to study and apply stochastic bilevel programs to real problems is an efficient algorithm for solving the model. As mentioned earlier, the deterministic bilevel problem is already quite difficult and time-consuming to solve; the introduction of a discrete random distribution on some

or all of its parameters causes an even greater increase in the problem size. For this reason, the development of efficient methods is primordial, as are the use of *decomposition* and *parallel strategies*, whenever possible.

Next, one method for generating subgradients of f is presented (based on [11]). Note that the algorithm utilizes the local Lipschitz continuity and directional differentiability of the implicit function f, and thus requires the additional assumptions presented above. Then, a penalty method making use of a merit function reformulation of the VIP is discussed, followed by a perturbation method which uses the *Karush–Kuhn–Tucker conditions* (KKT) of the VIP.

For a given $x \in \mathcal{X}$ and $\ell \in \mathcal{L}$, let y_ℓ be the (unique) solution to the lower-level problem, and let $\mathcal{I}(x, \ell) := \{i = 1, \ldots, m: g^i_\ell(x, y) = 0\}$. One then introduces the subsets $\mathcal{I}_+(x, \ell)$ and $\mathcal{I}_0(x, \ell)$ of $\mathcal{I}(x, \ell)$ for the active constraints whose (unique) multipliers satisfy $\lambda_i > 0$ and $\lambda_i = 0$, respectively. In the event that $\mathcal{I}_0(x, \ell)$ is nonempty (that is, strict complementarity does not hold at y_ℓ) one can further introduce a subset $\mathcal{J}(x, \ell)$ of $\mathcal{I}(x, \ell)$ for which the requirement is that $\mathcal{I}(x, \ell) \supseteq \mathcal{J}(x, \ell) \supseteq \mathcal{I}_+(x, \ell)$ holds. Let $g_\mathcal{I}$ and $d_\mathcal{I}$ denote the subvector of g and subvector of d where only rows $i \in \mathcal{I}$ are included.

Applying the analysis of [11], a subgradient of f can be calculated as follows.

First, for each $\ell \in \mathcal{L}$, solve the following linear system of equations:

$$\begin{pmatrix} \nabla_y L_\ell(x, y_\ell, \lambda_\ell) & -\nabla_y g_{\mathcal{J}(x,\ell)}(x, y_\ell)^\top \\ \nabla_y g_{\mathcal{J}(x,\ell)}(x, y_\ell) & 0^{m \times |\mathcal{J}(x,\ell)|} \end{pmatrix} \times \begin{pmatrix} d_{y_\ell} \\ d_{\lambda_{\mathcal{J}(x,\ell)}} \end{pmatrix} = \begin{pmatrix} -\nabla_y f(x, y_\ell) \\ 0^m \end{pmatrix}$$

in order to obtain d_{y_ℓ}, where $L_\ell(x, y_\ell, \lambda_\ell) := T_\ell(x, y_\ell) + \nabla_y g(x, y_\ell)^\top \lambda_\ell$, and $y_\ell := y_\ell(x)$.

Then, a subgradient of f at x is given by the formula $\xi_f(x) := \sum_{\ell \in \mathcal{L}} \rho_\ell [\nabla_x f(x, y_\ell) + \nabla_x L_\ell(x, y_\ell, \lambda_\ell)^\top d_{y_\ell} - \nabla_x g_{\mathcal{J}(x,\ell)}(x, y_\ell)^\top d_{\lambda_{\mathcal{J}(x,\ell)}}]$.

The subgradient can be used in an algorithm for the heuristic solution of the problem or embedded within a more sophisticated algorithm. The following *subgradient projection algorithm* utilizes the fact that f is differentiable almost everywhere and in particular when the y_ℓ, $\ell \in \mathcal{L}$, are strictly complementary:

Given $x \in \mathcal{X}$
An initial step in the direction of an arbitrary element $-\xi_f(x) \in -\partial f(x)$ is taken,
followed by a Euclidian projection onto $\mathcal{X}$:
a backtracking line search in this steplength is made so that
either the resulting feasible solution has a sufficiently lower objective value,
or a predetermined steplength is applied, whichever is greater.

Subgradient projection method

Note that, at points of nondifferentiability, the traditional projection method may break down because the negative of the subgradient may then not be a descent direction; in order to obtain a well-defined iteration at such points, one therefore utilizes a steplength which is the maximum of the one supplied by the backtracking line search and the result of a predetermined steplength formula used in traditional *subgradient optimization* techniques.

See [11] for the deterministic analog of the above analysis for calculating subgradients in a *bundle method* for the solution of a deterministic bilevel problem. This can be viewed as a more advanced technique than the above which ensures convergence to a stationary point.

Consider the following parallel resolution strategy for this model. In some cases, one may identify a cluster of scenarios with similar values. Then, by allocating these to the same processor, one may solve the corresponding lower-level problems utilizing efficient reoptimization procedures given that any of them have been solved to optimality, since the optimal solution to any one of them is feasible and near-optimal to all the others. Further, for scenarios with slightly differing sets, $\mathcal{J}(x, \ell)$, consider sorting the scenarios in each set so that $|\mathcal{J}(x, \ell)|$ is increasing. Then one may solve the preceding linear systems in sequence, expanding the matrix with the necessary rows and columns and utilizing the solution to the former system as a starting point in the search for the next. The fact that the choice of $\mathcal{J}(x, \ell)$ is arbitrary in the range of active constraints may be used to minimize the number of scenarios with distinct values of $\mathcal{J}(x, \ell)$.

By reformulating the variational inequality in the lower level through the use of a *merit* (or *gap*) func-

tion and introducing a penalty parameter, it is possible to devise a different method for solving the stochastic bilevel program. The merit function is defined as follows: Assume that for each scenario, $\ell \in \mathcal{L}$, one can find a continuous function ψ_ℓ that satisfies the following criteria:

$$\psi_\ell \geq 0$$

for every $x \in \mathcal{X}$ and all y, and

$$\psi_\ell = 0 \quad \Leftrightarrow \quad y \in S_\ell(x).$$

Then, the method is based on including the new function ψ_ℓ in the objective with a penalty parameter forcing it to zero, that is, for $\eta_\ell > 0$, one solves

$$\begin{cases} \min & \sum_{\ell \in \mathcal{L}} \rho_\ell [f(x, y_\ell) + \eta_\ell \psi_\ell(x, y_\ell)], \\ \text{s.t.} & x \in \mathcal{X}. \end{cases}$$

Note that the objective function remains separable with respect to the scenarios and can thus be decomposed and solved in parallel. For more details on this method and possible merit functions see [4,6,8,9].

A still different approach for solving MPEC or bilevel programs involves rewriting the solution set mapping of the lower-level variational inequality in terms of its KKT conditions. Letting λ_ℓ be the (unique, under the assumptions above) vector of multipliers for the lower-level constraint set $\mathcal{Z}_\ell(x)$, the KKT conditions are, for every $\ell \in \mathcal{L}$:

$$T_\ell(x, y) - \nabla_y g_\ell(x, y)\lambda_\ell = 0,$$
$$g_\ell(x, y) \leq 0, \quad \lambda_\ell \geq 0, \quad \lambda_\ell^\top g_\ell(x, y) = 0.$$

Then, the stochastic bilevel program is written as before with the constraints above replacing the constraints $y_\ell \in S_\ell(x)$.

The resulting program is an equivalent one-level reformulation of SBP, but is intractable due to the presence of the complementarity constraints. In [3] the above model was reformulated by expressing the lower-level constraints as $g_\ell(x, y) \geq 0$, $\ell \in \mathcal{L}$, and then writing the complementarity constraints as

$$g_\ell(x, y) - z_\ell = 0, \quad -2\min(z_\ell, \lambda_\ell) = 0,$$

for every $\ell \in \mathcal{L}$, where z_ℓ is a vector of the same dimension as λ_ℓ, and the min operator is applied to the vectors componentwise.

The resulting problem is tractable but nonsmooth, due to the min operator. The authors in [3] then reformulated the *nonsmooth optimization* problem by using a *perturbative approximation* to the min operation. This results in a sequence of smooth optimization problems converging to the nonsmooth problem as the perturbation parameters, η_ℓ, tend to zero.

Since the equations above are all separable with respect to the scenarios $\ell \in \mathcal{L}$, the same decomposition approaches can be applied to this method.

It should be noted that decomposition across scenarios may still prove insufficient for permitting the resolution of realistic stochastic bilevel programs. The use of *random sampling*, such as has been used in *stochastic quasigradient methods* and *stochastic decomposition*, as well as the development of approximation strategies, are lines of research that should be pursued in the future.

See also

- ▶ Bilevel Fractional Programming
- ▶ Bilevel Linear Programming
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Bilevel Programming
- ▶ Bilevel Programming: Applications
- ▶ Bilevel Programming: Applications in Engineering
- ▶ Bilevel Programming: Implicit Function Approach
- ▶ Bilevel Programming: Introduction, History and Overview
- ▶ Bilevel Programming in Management
- ▶ Bilevel Programming: Optimality Conditions and Duality
- ▶ Multilevel Methods for Optimal Design
- ▶ Multilevel Optimization in Mechanics

References

1. Birge JR, Louveaux F (1997) Stochastic programming. Springer, Berlin
2. Christiansen S, Patriksson M, Wynter L (2001) Stochastic bilevel programming in structural optimization. Structural Optim
3. Facchinei F, Jiang H, Qi L (1999) A smoothing method for mathematical programs with equilibrium constraints. Math Program 85:107–134

4. Facchinei F, Soares J (1997) A new merit function for nonlinear complementarity problems and a related algorithm. SIAM J Optim 7:225–247
5. Geoffrion AM (1970) Elements of large-scale mathematical programming. Managem Sci 16:652–691
6. Harker PT, Choi S-C (1991) A penalty function approach for mathematical programs with variational inequality constraints. Inform and Decision Techn 17:41–50
7. Hogan WW (1973) Directional derivatives for extremal-value functions with applications to the completely convex case. Oper Res 21:188–209
8. Larsson T, Patriksson M (1994) A class of gap functions for variational inequalities. Math Program 64:53–79
9. Marcotte P, Zhu D (1996) Exact and inexact penalty methods for the generalized bilevel programming problem. Math Program 74:141–157
10. Migdalas A, Pardalos PM, Varbrand P (eds) (1998) Multilevel optimization: Algorithms and applications. Kluwer, Dordrecht
11. Outrata J, Zowe J (1995) A numerical approach to optimization problems with variational inequality constraints. Math Program 68:105–130
12. Patriksson M, Wynter L (1999) Stochastic mathematical programs with equilibrium constraints. Oper Res Lett 25:159–167
13. Robinson SM (1980) Strongly regular generalized equations. Math Oper Res 5:43–62
14. Robinson SM (1991) An implicit-function theorem for a class of nonsmooth functions. Math Oper Res 16:282–309
15. Shimizu K, Ishizuka Y, Bard J (1996) Nondifferentiable and two-level mathematical programming. Kluwer, Dordrecht
16. Wets RJ-B (1989) Stochastic programming. In: Nemhauser GL, Rinnooy Kan AHG, Todd MJ (eds) Handbook Oper. Res. and Management Sci., vol 1. North-Holland, Amsterdam, pp 573–629

Stochastic Global Optimization: Stopping Rules

Fabio Schoen
Dip. Sistemi e Informatica, Universita Firenze, Firenze, Italy

MSC2000: 65K05, 90C26, 90C30, 65Cxx, 65C30, 65C40, 65C50, 65C60

Article Outline

Keywords
See also
References

Keywords

Global optimization; Random sampling; Optimal stopping; Look-ahead rules; Bayesian stopping rule

Every well-designed *global optimization* algorithm is usually equipped with three main components:

1) a sampling (or global) phase, whose aim is to investigate as thoroughly as possible the feasible region;
2) a local phase, designed to approximate good local optima;
3) a stopping rule, through which the algorithm is terminated either with a certificate of optimality (or an estimate of the error incurred) or with some kind of probabilistic measure of the error itself.

Many deterministic algorithm for global optimization problems are equipped with stopping criteria which enable to prove global optimality or to give a precise error bound. Unfortunately, from one side those algorithms are applicable only to strongly structured problems, like, e. g., the optimization of Lipschitz-continuous functions (with known Lipschitz constant), the minimization of concave functions (cf. ► Concave programming), or of functions which are explicitly representable as the difference of two convex functions (cf. ► D.C. programming); thus those stopping rules, based upon duality results and lower-bounding techniques, as well as the algorithms designed for those problems, cannot be applied to problems which do not possess that specific structure. On the other side, even for those strongly structured problems, it has been frequently observed that, in practice, it is quite likely that an algorithm will find the global optimum relatively quickly, but the vast majority of computational time is devoted to the proof of optimality. So, in some situations, it might be advisable to relax the requirement of certificated optimality, and to stop an algorithm as soon as there is sufficient evidence that the optimum has been found.

Stopping criteria based upon this idea are usually built after a stochastic model and are inspired from classical stopping rules developed within the field of statistical decision theory. Most, if not all, of the research in statistical stopping rules is based on the assumption that the algorithm used is either the *pure random search* or *pure Monte-Carlo* method, or *multistart*. The former is the most basic global optimization algorithm which consists only in generating a uniform ran-

dom sample in the feasible region and keeping track of the best observed function value (the record). The latter, multistart, differs from pure random sampling as it prescribes to apply a local optimization routine from each sampled point (thus it implements an effective, although computationally inefficient, local phase). Extension of the stopping rules built for these algorithms to different methods, like e. g. other two-phase methods (cf. also ► Stochastic global optimization: Two-phase methods) known as *multilevel single-linkage*, or *simple linkage*, has to be considered just as an heuristic.

It is possible to distinguish three main types of stopping rules, which mainly differ upon the criteria for stopping and the sophistication of the underlying stochastic model:

1) global exploration: stop as soon as there is sufficient evidence that all of the feasible region has been sampled;
2) enumeration of local optima: stop as soon as there is sufficient evidence that all local optima have been observed;
3) enumeration of good local optima: stop as soon as there is sufficient evidence that no local optimum better than the best so far will be observed.

It can be easily understood that the last criterion is the best one, but its practical realization poses very difficult modelization problems. In what follows, a brief account of the main models and methods for each of the three models will be given.

Let $S \subseteq \mathbf{R}^d$ ($d \in \mathbf{N}$) be the feasible region for the global optimization problem

$$f^* = \min_{x \in S} f(x). \tag{1}$$

It is assumed that the Lebesgue measure $\mu(S)$ of S is finite and strictly positive. From elementary probability, it is easy to derive that, after N uniform points have been sampled, the probability that at least a point in the sample falls within a prescribed subset of S whose Lebesgue measure is ϵ is given by

$$1 - \left(1 - \frac{\epsilon}{\mu(S)}\right)^N; \tag{2}$$

thus, given a prefixed probability level $p \in (0, 1)$, if sampling is terminated as soon as

$$N \geq \frac{\log(1-p)}{\log\left(1 - \frac{\epsilon}{\mu(S)}\right)} \tag{3}$$

then every region of volume greater than ϵ will contain a sample point with probability at least p. Letting

$$L^\eta := \{x \in S\colon\ f(x) \leq f^* + \eta\} \tag{4}$$

be the η-level set of f, it is in principle possible, although extremely difficult in practice, once an error level η has been chosen, to let $\epsilon = \mu(L^\eta)$. The main disadvantage of this very simple stopping rule is that it usually prescribes to stop very late, when every region of volume ϵ, and not just L^η, has been observed. In principle, this method might be applied with success to multistart: this algorithm can be seen as a pure random search method applied to the composition of the objective function f and the mapping of S into itself which arises associating to each feasible point $x \in S$ the point which is obtained starting a local optimization routine from x. This way, being the resulting composite function piecewise constant, a (usually much) larger value of ϵ can be safely used. Unfortunately no practical method is known which enables to approximate the correct value of ϵ. Some attempts have been reported in the literature (see e. g. [6]), but the rules proposed, although quite simple, seem to be very inefficient in providing a quick and reliable stopping time.

The second class of methods for stopping, designed for the complete enumeration of local optima through multistart, originates from ideas introduced in[9]. If the starting points in multistart are stochastically independent, then, letting the local optima be arbitrarily numbered as $1, \ldots, T$, the probability of hitting n_1 times the first, n_2 times the second, and so on, is given by

$$\frac{n!}{n_1! \cdots n_T!} \theta_1^{n_1} \cdots \theta_T^{n_T}. \tag{5}$$

Parameters θ_j in (5) represent the *share* of the jth local optimum, that is, the relative volume of its *region of attraction*, or, equivalently, the probability that a local search started from a point in the sample leads to the jth local minimum. In practice, neither T, nor the shares are known. A *Bayesian decision-theoretic framework* is used, in which a prior probability distribution is given on the unknown parameters of the probabilistic model; after each observation is made, through a Bayesian updating, an a posteriori distribution is quite easily computed. Based upon this model, several rules can be obtained for stopping, depending upon different cost structures. Following a standard procedure in

Bayesian sequential decision theory, a loss function is defined which gives a measure of the trade-off between stopping and continuing sampling; then, sequentially, at each iteration the posterior expected loss is computed and the alternatives of stopping and continuing are compared. For example, one might define a loss by specifying a price to be paid either for stopping before all local optima have been found or for continuing after the global optimum has been observed. Alternatively, one can define a cost for each local search and a gain (a negative loss) incurred if a new local optimum is discovered. Stopping rules derived from this framework have the advantage of being very simple to implement; unfortunately, the model and the resulting stopping rules just depend on the number of different local optima discovered; the value of the objective function at the local optima never plays a role, so that these criteria tend to be quite conservative and highly sensible to the total number of local optima, irrespective of their value. Well known and widely used results in this framework can be found in [3]; an ingenious trick is presented in [8] which consists in ordering the observed function values at local optima; a term which takes into account the desire to stop if no optima better than the best so far is observed is thus included in the loss. Here again function values do not explicitly enter neither the model nor the stopping rules, but the relative rank of each local optimum is considered. The resulting rules, although more complex than those in [3], display a significantly better behavior.

In all of the above methods for stopping, a prior distribution has to be given on the total number of local optima and the shares; while it is quite natural to assume a Dirichlet distribution for the shares, the question of choosing a sensible prior for T is still to be answered; a seemingly interesting choice, consisting of an improper prior, giving constant weight to all positive integers, was proven to give incorrect results in [8].

The last generation of stopping rules for multistart derive from even more complex models. The idea of explicitly including function values in the model raises extremely difficult theoretical problems. In fact it is very hard to identify a sensible stochastic models for the observed function values at randomly chosen points or, even worse, at local optima obtained by starting a local optimization routine from a random starting point. Again, the best one can hope, is to define a prior model and adapt the resulting probability distribution by means of Bayes' theorem. An attempt in this direction can be found in [1] and [2]; there it is assumed that function values at local optima obtained from multistart follow a probability distribution which is largely unknown. Thus this probability distribution is modeled as one out of an infinite class of possible probability distributions: the prior knowledge is made explicit in the definition of a prior probability distribution over a class of distribution functions. From the literature on Bayesian nonparametric inference the so called *simple homogeneous process* [5] was selected, due to its representativeness (its realizations are dense in the space of continuous distributions) and to its computational manageability. Accordingly, stopping rules were derived which prescribe to stop as soon as

$$\int_{-\infty}^{f_n^*} (f_n^* - y)\, d\widehat{F}_n(y) \le c, \tag{6}$$

where $f_n^\star$ is the record value observed after the first n samples (more precisely, after the first n local searches), $\widehat{F}_n$ is the expected posterior distribution, computed through the prior process and the n observations of local optima, c is a threshold. The left-hand side in (3) is the expected improvement after the the next observation with respect to the current record. The effectiveness of the resulting stopping rules has been tested and very good computational results have been reported; unfortunately the rules are quite cumbersome to implement and require the setting of several parameters in the definition of the prior.

An alternative approach is described in [4], where the problem of specifying a prior distribution on the set of probability distributions is simplified assuming that function values are discretized; this way the underlying probabilistic model becomes a parametric one, although with a possibly huge number of parameters, namely the global minimum and global maximum values of f, and the probability of observing any of the discretized values between them. Although the resulting rules are attractively simple, the idea of discretizing the range is far from being a satisfactory one, as a discretization which is too large might lead to incorrect decisions on the global optimum, while a narrow one enlarges the dimension of the parameter space. Again the problem of specifying a prior on the parameters, in particular of

the global minimum and maximum, is a difficult one; moreover the sensitivity of the rules to function values is questionable: the authors proof that, using a loss function which is the difference between a cost associated to sampling and the ratio between the possible improvement over the record value and the range of values for f, the optimal stopping rule depends only on the iteration number; thus no information on the observed function values is used, thus easily leading to incorrect decisions.

Unfortunately the main difficulty in stopping a stochastic global optimization algorithm is not any of the above mentioned ones, like e. g., the difficulty of choosing parameters, the insensibility of some rules to function values or the cumbersome implementation. The real weakness of all the above mentioned approaches is the fact that all of them are based upon the analysis of stochastically independent samples in the feasible region. While this assumption is natural for pure random search and for multistart, it becomes false as soon as more refined methods are used, notably two-phase methods. Thus all of these rules become simply heuristic stopping criteria. What is worse is that they do not provide a reliable estimate of the error incurred after stopping; the user is thus left with a heuristic rule with no guarantee. It is not a surprise that most stochastic global optimization users just let their algorithm run until some time limit is exhausted. Unfortunately deriving stopping rules for more clever algorithms is an extremely hard task; some attempts have been reported in [7] when dealing with Bayesian global optimization methods (cf. ► Bayesian global optimization), but even if good stopping rules can be derived in that framework, the results are again only heuristic, as they are based upon models of the objective function which are usually not justifiable.

In conclusion, while stopping is a crucial component in stochastic global optimization and late stopping is generally the main cause of inefficiency, research in this field seems to have stopped in the last years; there is still a need for good criteria for general algorithms, capable of producing a reliable estimate of the error.

See also

- ► Adaptive Simulated Annealing and its Application to Protein Folding
- ► Bayesian Global Optimization
- ► Genetic Algorithms for Protein Structure Prediction
- ► Global Optimization Based on Statistical Models
- ► Global Optimization: Hit and Run Methods
- ► Monte-Carlo Simulated Annealing in Protein Folding
- ► Packet Annealing
- ► Random Search Methods
- ► Simulated Annealing
- ► Simulated Annealing Methods in Protein Folding
- ► Stochastic Global Optimization: Two-phase Methods

References

1. Betrò B, Schoen F (1987) Sequential stopping rules for the multistart algorithm in global optimisation. Math Program 38:271–286
2. Betrò B, Schoen F (1992) Optimal and suboptimal stopping rules for the multistart algorithm in global optimisation. Math Program 57:445–458
3. Boender CGE, Rinnooy Kan AHG (1987) Bayesian stopping rules for multistart global optimization methods. Math Program 37:59–80
4. Boender CGE, Rinnooy Kan AHG (1991) On when to stop sampling for the maximum. J Global Optim 1(4):331–340
5. Ferguson TS, Phadia EG (1979) Bayesian nonparametric estimation based on censored data. Ann Statist 7:163–186
6. Hart WE (1998) Sequential stopping rules for random optimization methods with applications to multistart local search. SIAM J Optim 9:270–290
7. Locatelli M, Schoen F (1995) An adaptive stochastic global optimization algorithm for one-dimensional functions. Ann Oper Res 58:263–278
8. Piccioni M, Ramponi A (1990) Stopping rules for the multistart method when different local minima have different function values. Optim 21:697–707
9. Zieliński R (1981) A statistical estimate of the structure of multiextremal functions. Math Program 21:348–356

Stochastic Global Optimization: Two-Phase Methods

Fabio Schoen
Dip. Sistemi e Informatica, Universita Firenze, Firenze, Italy

MSC2000: 65K05, 90C26, 90C30, 65Cxx, 65C30, 65C40, 65C50, 65C60

Article Outline

Keywords
See also
References

Keywords

Global optimization; Random sampling; Acceptance/rejection; Multilevel methods

Two-phase methods are *global optimization* algorithms which consist of sampling (*global phase*) coupled with refinement or approximation of local optima (*local phase*). Although this definition is extremely general, as to encompass virtually all known methods of global optimization, the term 'two-phase' is generally used in connection with methods based upon random sampling and local optimization started from selected points in the sample.

To fix the notation, let the problem under consideration be that of finding

$$f^* = \min_{x \in S} f(x), \tag{1}$$

where $S \subseteq \mathbf{R}^d$ is a d-dimensional set (usually closed, often compact) and $f\colon S \to \mathbf{R}$ is a continuous objective function. On f no other special assumption is generally made except that a local search algorithm is available, which is capable of producing a *local optimum* once started from a *feasible point*. Thus, depending on the local optimization routine employed, f might, for example, be required to be continuously differentiable.

Two-phase methods are best suited for *global optimization problems* with no special structure, while different, often deterministic, methods are preferred when dealing with strongly structured global optimization problems like, e. g., concave minimization (cf. also ► Concave programming), d.c. problems (cf. also ► D.C. programming), Lipschitz continuous problems; for a general reference on these particular classes of structured global optimization problems, besides this volume, the reader might wish to consult [3].

Two-phase methods display sufficiently good behavior when the following conditions are met:

- sampling in the feasible set is not too difficult (frequently S is assumed to be a hyper-rectangle);
- the dimension d is not too high: until a few years ago $d = 10$ seemed already to be a high dimension; more recent developments pushed this limit to more than 60;
- the Lebesgue measure of the *region of attraction* of the global optima is not too small. The 'region of attraction' of a local (and, hence, also the global) optimum is defined as the maximal subset $A \subseteq S$ characterized by the fact that a local search started from any point in A leads to that local optimum;
- the computational cost of evaluating f at a feasible point is substantially lower than that of performing a local search, and it is not extremely expensive by itself.

For low-dimensional problems, when the last condition is not met and function observation is a computationally demanding task it is preferable to switch to methods based upon approximate models of the objective function, the most well-known representative of which is the class of Bayesian algorithms (cf. also ► Bayesian global optimization).

What has come to be known as the class of *two-phase* methods consists of the following very general optimization scheme:

```
PROCEDURE two-phase()
    let k := 1;
    WHILE (StoppingRule() == FALSE) do:
        choose N_k > 0;
        (Begin Phase I):
        sample a set S_k of N_k points from S;
        let S^k := ⋃_{h=1}^{k} S_h
        (Begin Phase II):
        choose S* ⊆ S^k;
        FOR EACH x ∈ S* DO
            start a local search from x;
        END FOR;
        let k := k + 1;
    END WHILE;
END two-phase;
```

In practice Phase I, or *global phase*, aims at exploring as thoroughly as possible the feasible region, while in Phase II, or *local phase*, the approximation of good local optima is carried out. It can be easily seen that most methods for global optimization can be seen as special instances of the above scheme.

Phase I usually consists of *uniform random sampling*, even if some attempts have been made to use

quasirandom sequences which have, among others, the advantage of producing smaller gaps between sampled points.

As the main computational burden is assumed to be connected with local searches, it is natural that the main distinction between different two-phase methods lies in the decision regarding the choice of $S^\star$, the set of starting points for local optimization. The following are some of the most well-known strategies for defining such a set:

- *pure random search*: $S^\star = \emptyset$, that is, no local searches are performed. This method is a basic, although very inefficient, global optimization strategy. It is also known as the *pure Monte-Carlo method*;
- *best start*: $S^\star \subseteq S_k$ is equal to the set of new 'record points', i. e. to the set of points at which, during the current iteration, a function value strictly lower than the best so far has been observed;
- *multistart*: $S^\star = S_k$. This method prescribes that a local search is started from every sampled point;
- *clustering*: $S^\star$ is defined as the set of record points in suitable subsets of the sample built according to a clustering procedure. The idea of clustering for global optimization dates back to two papers in the 1970s, [2] and [10], where the idea of concentrating the sample in order to approximate the regions of attraction of low-level local optima was introduced. Concentration of the sample is achieved either by discarding a fixed fraction of the points with highest function value, or by performing a single, or just a few, descent steps from points in the sample. These procedures transform the sample into a nonuniform one: clustering techniques are then employed to identify subsets of the sample with a higher-than-average concentration of points. This idea is further exploited in *density clustering* methods (see [9] and [7]) where clusters are grown around suitable 'seed points' by progressively enlarging an ellipsoidal set until the relative density of points of the transformed sample which fall inside the ellipsoid becomes smaller than a threshold;
- *multilevel single-linkage* (in short: MLSL). In this method the user specifies a constant sample size $N_k = N$ and two positive reals σ and ν. Then $S^\star$ is defined as follows: a point $x \in S^k$ (the whole sample) is included in $S^\star$ if no point $y \neq x$ in the sample S^k exists such that:

$$f(y) \leq f(x) \quad \text{and} \quad \|x - y\| \leq r_{N,k,\sigma}.$$

It is moreover required that points in $S^\star$ are 'sufficiently far' from the boundary, that is, given the threshold $\nu > 0$, it is required that

$$\operatorname{dist}(x, \partial S) \geq \nu, \qquad \forall\, x \in S^*,$$

and that neither a local search had been already applied to x, nor x is too near to a previously discovered local optimum.
Here dist is the Euclidean distance and ∂S denotes the boundary of S; the variable threshold $r_{N,k,\sigma}$ is defined as:

$$\pi^{-1/2} \left(\Gamma\left(1 + \frac{d}{2}\right) \mu(S) \sigma \frac{\log kN}{kN} \right)^{1/d}. \tag{2}$$

In (2), Γ is the gamma function, μ represents the Lebesgue measure, σ is a positive constant. The basic idea of this method, which was analyzed in [8], is that, instead of building clusters with prescribed shape (e. g., ellipsoidal), points are clustered by means of a distance criterion. In particular a point is clustered to another in the sample if this latter is near enough and has a better function value. Local searches are started only from unclustered points. This way, ideally linking points which are clustered together, a forest is built and local searches are started only from the root of each tree; it is hoped that each tree in the forest connects points within the region of attraction of a single local optimum. As the threshold in (2) decreases to 0, a single tree will eventually be broken into two or more connected components, and it may happen that local searches are started also from points which, in previous iterations, were not considered to be good candidates.
- *simple linkage* (in short: SL). Here $N_k = 1$ and either $S^\star = \emptyset$ or $S^\star = S_k$ (a single point). In particular a local search will be started from the most recently sampled point x if and only if no point $y \neq x$ in the sample S^k exists such that:

$$f(y) \leq f(x) + \epsilon \quad \text{and} \quad \|x - y\| \leq r_{k,\sigma},$$

where $r_{k,\sigma}$ is defined as

$$\pi^{-1/2} \left(\Gamma\left(1 + d/2\right) \mu(S) \sigma \frac{\log k}{k} \right)^{1/d}. \tag{3}$$

Symbols used in (3) have the same meaning as before; ϵ is a small positive constant. This method was introduced in [5] and theoretically compared with MLSL in [4]. The main differences with MLSL are in the choice of sampling a single point at each iteration and of letting the possibility of starting a local search only from the last single sampled point, instead of having the necessity, at each iteration, of reconsidering the whole sample again;

- *Topographical search*. $S^\star$ is defined as the set of local record points in the sample: given an integer $m \geq 1$ a point is a local record if its functional value is lower than that of its m nearest neighbors. A variant of this idea was presented in [11].

In all these methods, provided that sampling produces a dense set of points in S, convergence of the best observed function value to the global optimum is achieved with probability 1. It was proven in [1] that producing a dense set of observations is, in a certain sense, also necessary. Moreover, if a prefixed accuracy $\eta > 0$ is given and if it is assumed that the level set

$$L^\eta = \{x \in S\colon\ f(x) \leq f^* + \eta\}$$

has a positive Lebesgue measure, then all of the above algorithms will almost surely place an observation in L^η in a finite number of iterations. What distinguishes poorly performing methods like Pure Random Search from MLSL or Simple Linkage is the fact that, through local searches, these methods attempt to place observations in L^η as soon as a point is sampled in the (hopefully much larger) region of attraction of the global optimum.

Multistart succeeds in reaching this goal, starting a local search from *every* sampled point. This has the negative effect of wasting a huge quantity of computational power both during local searches leading to local (nonglobal) optima, and in discovering each local optimum more than once. Both MLSL and Simple Linkage try to reach a compromise between Pure Random Search and Multistart by starting a few local searches only from a selected set of promising points. The main theoretical properties of MLSL and SL are the following:

- the probability of starting a local search at iteration k decreases to 0 provided that $\sigma > 2$ in MLSL and $\sigma > 0$ in SL;
- the total expected number of local searches started, even if the algorithm is run forever without stopping, is finite, provided that $\sigma > 4$ in MLSL and $\sigma > 2^d/\mathrm{d}$ in SL (these results hold when S is the d-dimensional hypercube).

These two properties are crucial in evaluating the efficiency of MLSL and SL: they state that the total effort devoted to local searches is kept at a low level. The different conditions imposed on σ in the last property come from an important difference in the assumptions of MLSL and SL: while the former forbids local searches to be started from sampled points which are within a prefixed distance from the boundary, the latter allows local searches to be started from any point in S, including the boundary. As the dimension d increases, given, as prescribed in MLSL, $\nu > 0$, the probability of sampling a point whose distance from the boundary is less than ν increases. This fact might help to explain the fact that MLSL was successfully applied only to quite low-dimensional global optimization problems. On the other hand SL was applied with success to high-dimensional global optimization problems like, e. g., the minimization of *Lennard–Jones potential energy function*, a classical test for global optimization derived from computational chemistry, characterized by the presence of a number of local optima which increases exponentially with the dimension of the problem.

Recent developments in two-phase methods aim at:

- analyzing their finite time behavior (all of the properties mentioned above were asymptotic ones), possibly leading to different definitions of the thresholds used for deciding whether to start or not local searches;
- improving the sampling phase, in order to avoid the possibility that a local search is started from a point just because no other point was sampled in a suitable neighborhood.

This last point can be tackled by building the sample in such a way that large gaps between sampled points are avoided; as an example, *quasirandom* sequences might be employed (for a general reference see [6]). It should be observed however that the substitution of pseudo-random points with quasirandom ones needs a thorough redefinition of the criteria used for starting local searches. As an alternative, the idea itself of using a threshold might be abandoned, leading to methods similar to Topographical Search.

As a concluding remark, it should be observed that all the effort in these methods is devoted to placing, at relatively low computational cost, observations in a small neighborhood of the global optimum. This effort is wasted if the algorithm is not equipped with a good stopping rule (cf. also ► Stochastic global optimization: Stopping rules). Unfortunately, no stopping rule is sensible for such poorly structured problems; the only possibility up to now has been that of using heuristic stopping criteria, some of which particularly complex, derived from simplified stochastic models. Regrettably, research in the field of stopping rules for two phase methods seems to have stalled in the last few years, probably as a consequence of the low confidence users put into statistical stopping rules which never can give a guarantee, or even an estimate, about the error in approximating the global optimum.

See also

- ► Adaptive Simulated Annealing and its Application to Protein Folding
- ► Bayesian Global Optimization
- ► Genetic Algorithms for Protein Structure Prediction
- ► Global Optimization Based on Statistical Models
- ► Global Optimization: Hit and Run Methods
- ► Monte-Carlo Simulated Annealing in Protein Folding
- ► Packet Annealing
- ► Random Search Methods
- ► Simulated Annealing
- ► Simulated Annealing Methods in Protein Folding
- ► Stochastic Global Optimization: Stopping Rules

References

1. Baritompa W, Stephens C (1998) Global optimization requires global information. J Optim Th Appl 96(3):575–588
2. Becker RW, Lago GV (1970) A global optimization algorithm. In: Proc. 8th Allerton Conf. on Circuits and Systems Theory, pp 3–12
3. Horst R, Pardalos PM (eds) (1995) Handbook global optimization. Kluwer, Dordrecht
4. Locatelli M, Schoen F (1996) Simple linkage: Analysis of a threshold-accepting global optimization method. J Global Optim 9:95–111
5. Locatelli M, Schoen F (1999) Random linkage: A family of acceptance/rejection algorithms for global optimisation. Math Program 85(2):379–396
6. Niederreiter H (1992) Random number generation and quasi-Monte Carlo methods. SIAM, Philadelphia, pp 241
7. Rinnooy Kan AHG, Timmer GT (1987) Stochastic global optimization methods. Part I: Clustering methods. Math Program 39:27–56
8. Rinnooy Kan AHG, Timmer GT (1987) Stochastic global optimization methods. Part II: Multi level methods. Math Program 39:57–78
9. Töorn AA (1977) Cluster analysis using seed points and density determined hyperspheres with an application to global optimization. IEEE Trans Syst, Man Cybern 610–616
10. Törn AA (1978) A search clustering approach to global optimization. In: Dixon LCW, Szegö GP (eds) Towards global optimization 2. North-Holland, Amsterdam
11. Törn AA, Viitanen S (1994) Topographical global optimization using pre-sampled points. J Global Optim 5:267–276

Stochastic Integer Programming: Continuity, Stability, Rates of Convergence

RÜDIGER SCHULTZ
Department Math., Gerhard-Mercator University, Duisburg, Germany

MSC2000: 90C15, 90C11, 90C31

Article Outline

Keywords
See also
References

Keywords

Stochastic integer programming; Stability analysis

As in many other branches of mathematical optimization, stochastic programming theory and algorithms have to be rethought completely when including integer requirements. Among stochastic integer programs so far the *linear two-stage model* is best understood, both structurally and algorithmically. It is the problem

$$\min\left\{cx + Q(x,\mu):\ Bx = b,\ x \in \mathbf{Z}_+^n \times \mathbf{R}_+^{n'}\right\},$$

where

$$Q(x,\mu) = \int_{\mathbf{R}^{\bar{s}}} \Phi(z - Ax)\mu(\,d(z,A))$$

and

$$\Phi(t) = \min\left\{ qy + q'y' : \begin{array}{l} Wy + W'y' = t, \\ y' \in \mathbf{R}^{s'}_{+},\ y \in \mathbf{Z}^{s}_{+} \end{array} \right\}.$$

Structural properties of the above model are mainly determined by the interaction of the second-stage value function Φ and the integrating probability measure μ. Due to the integer requirement on y the value function Φ is no longer convex, as would have been the case with the integer-free counterpart model. Studies of the *mixed integer value function* Φ date back to the 1970s [5]. Under mild assumptions including in particular the rationality of W and W' it holds that $\Phi(\mathrm{t}) \in \mathbf{R}$ for all t, and the function Φ is lower semicontinuous. Moreover, the following properties of Φ (established in [3,5]) are useful prerequisites for the analysis of the above model:

1) There exists a countable partition $\cup_{i=1}^{\infty} T_i$ of the domain of Φ such that the restrictions of Φ to T_i are piecewise linear and Lipschitz continuous with a uniform constant $L > 0$ not depending on i.
2) Each of the sets T_i has a representation $T_i = \{t_i + K\} \setminus \cup_{j=1}^{N} \{t_{ij} + K\}$, where K denotes the polyhedral cone $W'(\mathbf{R}^{s'}_{+})$, t_i, t_{ij} are suitable points from the argument space, and N does not depend on i.
3) There exist positive constants β, γ such that $|\Phi(t_1) - \Phi(t_2)| \leq \beta \parallel t_1 - t_2 \parallel + \gamma$ for arbitrary t_1, t_2.

Although discontinuous, the function Φ hence is not 'too bad': discontinuities only occur in subsets of hyperplanes, in its continuity regions the function is even Lipschitzian with uniform modulus, and the overall growth of the function is bounded by an affine expression.

The combination of these properties with tools from probability theory leads to statements on the *joint continuity* of Q as a function both of the decision variable x and the integrating probability measure μ. The latter, of course, needs a proper convergence notion in the space of probability measures. Here *weak convergence of probability measures* [4] has turned out to be sufficiently broad to cover relevant applications and sufficiently strict to allow substantial conclusions. Continuity of Q both in x and μ has direct consequences for the *stability* of the stochastic integer program when perturbing the underlying probability measure μ. Such perturbations are motivated by two reasons: In practical modeling the probability distribution of the random parameters is always subjective. The modeler hence wants to be sure that slight modifications of the distribution do not lead to drastic changes in the solution. Secondly, the integral defining Q is multivariate with a dimension governed by the dimension of the underlying random vector which usually is quite big. Numerical integration hence fails if μ is nondiscrete. Approximating μ by discrete measures then turns integrals into sums which are numerically feasible. Of course, this has to be accompanied by the safeguard that 'close' approximations of model data (the measure μ) end up in 'close' approximations of the model output (the optimal value and the solution set).

Under mild technical assumptions that basically ensure Φ and Q to be well defined real-valued functions the following results on continuity, stability and rates of convergence for stochastic integer programs are known.

Fatou's lemma and the lower semicontinuity of Φ imply lower semicontinuity of $Q(\cdot, \mu)$ [11,13]. Via Lebesgue's dominated convergence theorem this extends to continuity of $Q(\cdot, \mu)$ at a given x provided the exceptional set $E(x)$ of all (z, A) such that Φ is discontinuous at $z - Ax$ has μ-measure zero [11,13]. Since discontinuities of Φ are located in a set of Lebesgue measure zero (cf. property 2) above) the condition on $E(x)$ is fulfilled if μ has a density. This also covers the first continuity result in the field obtained by L. Stougie [15]. Adding boundedness and monotonicity assumptions on densities of one-dimensional marginal distributions of linear transforms of μ leads to *Lipschitz continuity* of $Q(\cdot, \mu)$ [12,13]. Here the above properties 1) and 3) of Φ are essential for the proof.

A particular problem class is given by *two-stage stochastic programs with simple integer recourse*. Here, the function Q is much better understood since it enjoys separability properties and essential parts of the analysis can be done in dimension one. Results comprise sufficient conditions for differentiability of $Q(\cdot, \mu)$, an algorithm for constructing the convex hull of the epigraph of $Q(\cdot, \mu)$ and convexification procedures for $Q(\cdot, \mu)$ based on proper modifications of μ [6,7,8,16].

Continuity of Q as a function jointly in x and μ can be obtained by adding elements from the theory of weak convergence of probability measures [4]. Using Rubin's theorem on weak convergence of image measures induced by discontinuous transformations [4] it is possible to show that $Q(\cdot, \cdot)$ is continuous at (x, μ) if the

above mentioned exceptional set $E(x)$ has μ-measure zero [13]. In [1] the authors study upper and lower semicontinuity of integral functionals with discontinuous integrands of which $Q(\cdot, \cdot)$ is a special case. The role of the discontinuity set $E(x)$ is then taken by properly defined exceptional sets of missing upper and lower semicontinuity. *Semicontinuity* of the integral functional then essentially follows if the corresponding exceptional set has μ-measure zero.

When heading for *rates of quantitative continuity* of Q as a function of μ, e. g., Lipschitz or *Hölder continuity*, it is essential to select a metric on the space of probability measures (*probability metric*, [9]) that, on the one hand, fits to the discontinuous integrand and, on the other hand, metrizes weak convergence of probability measures under mild assumptions. In [14] a specific variational distance meeting these requirements is proposed and a Hölder continuity result for $Q(x, \cdot)$ is established. The polyhedral cone K arising in the above property 2) of Φ enters the definition of the variational distance as a crucial ingredient.

By standard arguments from the *stability analysis of optimization problems* with parameters in general topological or metric spaces [2,10] the above continuity statements for Q can be turned into stability results for the optimal value and the set of optimal solutions of the underlying stochastic integer program. In particular, results of this type were obtained for (Hölder) continuity of the optimal value and for upper semicontinuity of the solution set mapping [1,11,13,14].

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds

- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Time-dependent Traveling Salesman Problem
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Artstein Z, Wets RJ-B (1994) Stability results for stochastic programs and sensors, allowing for discontinuous objective functions. SIAM J Optim 4:537–550
2. Bank B, Guddat J, Klatte D, Kummer B, Tammer K (1982) Nonlinear parametric optimization. Akad. Verlag, Berlin
3. Bank B, Mandel R (1988) Parametric integer optimization. Akad. Verlag, Berlin
4. Billingsley P (1976) Convergence of probability measures. Wiley, New York
5. Blair CE, Jeroslow RG (1977) The value function of a mixed integer program: I. Discret Math 19:121–138
6. Klein Haneveld WK, Stougie L, van der Vlerk MH (1996) An algorithm for the construction of convex hulls in simple integer recourse programming. Ann Oper Res 64:67–81
7. Klein Haneveld WK, van der Vlerk MH (1994) On the expected value function of a simple integer recourse problem with random technology matrix. J Comput Appl Math 56:45–54
8. Louveaux FV, van der Vlerk MH (1993) Stochastic programs with simple integer recourse. Math Program 61:301–326
9. Rachev ST (1991) Probability metrics and the stability of stochastic models. Wiley, New York
10. Rockafellar RT, Wets RJ-B (1997) Variational analysis. Springer, Berlin
11. Schultz R (1992) Continuity and stability in two-stage stochastic integer programming. In: Marti K (ed) Stochastic Optimization - Numerical Methods and Techn. Applications. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 81–92
12. Schultz R (1993) Continuity properties of expectation functions in stochastic integer programming. Math Oper Res 18:578–589
13. Schultz R (1995) On structure and stability in stochastic programs with random technology matrix and complete integer recourse. Math Program 70:73–89
14. Schultz R (1996) Rates of convergence in stochastic programs with complete integer recourse. SIAM J Optim 6:1138–1152
15. Stougie L (1987) Design and analysis of algorithms for stochastic integer programming. CWI Tract, vol 37. Center Math. and Computer Sci., Amsterdam
16. van der Vlerk MH (1995) Stochastic programming with integer recourse. PhD Thesis, Univ. Groningen

Stochastic Integer Programs

SIP

FRANCOIS LOUVEAUX[1], JOHN R. BIRGE[2]
[1] University Namur, Namur, Belgium
[2] Northwestern University, Evanston, USA

MSC2000: 90C15, 90C10

Article Outline

Keywords
See also
References

Keywords

Combinatorial optimization; Stochastic programming

A *stochastic linear program with recourse* is a program of the form

$$\begin{cases} \min & cx + Q(x) \\ \text{s.t.} & x \in X \end{cases}$$

where $Q(x) = \mathcal{E}_\xi Q(x, \xi)$, $Q(x, \xi) = \min_{y(\xi) \in Y(\xi)} qy(\xi)$ and $\mathcal{E}_\xi$ denotes the mathematical expectation with respect to ξ. X and $Y(\xi)$ are usually polyhedral convex sets. In recourse programs, some decisions (x), called *first-stage decisions*, must be taken before knowing the particular values taken by the random variables (ξ) while some other decisions ($y(\xi)$), called *second-stage decisions* or *recourse actions*, can be taken after the realizations of the random variables are known. In this representation, $Q(x, \xi)$ is the second-stage value function for a given ξ and $Q(x)$ the expected value-function or expected recourse.

A *stochastic integer program with recourse* (SIP) is a stochastic program, where some of the decisions are restricted to be integer, either in the first-or in the second-stage problem. It is an extension of integer programming or combinatorial optimization, where some of the problem data are random variables. Any application of combinatorial optimization can thus be extended to a stochastic integer program. Typical applications are in the energy sector [16], resource acquisition [1], location problems [10], stochastic scheduling [2], stochastic knapsack for yield management [15]. We

concentrate here on the *recourse* formulation of SIP although some models incorporate instead probabilistic constraints.

SIPs are notoriously difficult unless the only integrality requirements are restricted in the first-stage. Indeed, the *expected recourse function* is known to be nonconvex, discontinuous (with some exceptions when the random variables are absolutely continuous [10,13]. Similarly, the set of first-stage decisions that yield second-stage feasible solutions is in general nonconvex. It follows that the available methods and properties are scarce for this problem. Research has thus concentrated on some specific problems.

One major situation where some properties are available is the *simple integer recourse* problem, defined as follows:

$$Q(x,\xi) = \min \left\{ q^+ \cdot y^+ + q^- \cdot y^- : \begin{array}{c} y^+ \geq h - Tx, \\ -y^- \leq h - Tx, y^+, \\ y^- \geq 0 \text{ and integer} \end{array} \right\},$$

where ξ is formed by the stochastic components of q^+, q^-, h and T. Here, any difference in $h - Tx$ with respect to zero must be compensated by an integer quantity y^+ or y^-. This compensation is calculated componentwise. The expected value of a simple integer recourse problem can be computed either exactly or by an approximation whose error bound can be controlled. Moreover, a componentwise convexity property can be derived between points that are at an integer distance so that an exact algorithm can be obtained, in particular in the case where the first-stage variables are integer [11]. Also, in several cases, the convex hull of the expected recourse can be obtained [6].

Another line of approach is to use the hierarchy between aggregate level decisions, which are typically those restricted to be integer, and detailed level decisions, which are very often continuous. *Hierarchy* has been used either through *Benders decomposition* [9] or within the framework of *asymptotic analysis* [8].

Bender's decomposition has also been applied to the case when the first-stage variables are binary variables [7]. Those methods sometimes use the terminology 'integer L-shaped' to stress the similarity with what is done in linear (continuous) stochastic programs. Applications have mainly be in the routing area as many examples exist where the expected second-stage recourse functions is computable. Of particular importance, is the possibility to develop lower bounding functionals that are also valid at fractional solutions (see ► Fractional combinatorial optimization).

A description of Bender's decomposition for SIP in a more general framework is available in [5]. An interesting alternative is to combine *dual decomposition* and *Lagrangian relaxation* [4].

Clearly, this field is only in its infancy (as of 1999) so that we may expect many more results in the coming years. A bibliography is available in [14] and a general presentation in [3].

See also

- ► Approximation of Extremum Problems with Probability Functionals
- ► Approximation of Multivariate Probability Integrals
- ► Branch and Price: Integer Programming with Column Generation
- ► Decomposition Techniques for MILP: Lagrangian Relaxation
- ► Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ► Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ► General Moment Optimization Problems
- ► Integer Linear Complementary Problem
- ► Integer Programming
- ► Integer Programming: Algebraic Methods
- ► Integer Programming: Branch and Bound Methods
- ► Integer Programming: Branch and Cut Algorithms
- ► Integer Programming: Cutting Plane Algorithms
- ► Integer Programming Duality
- ► Integer Programming: Lagrangian Relaxation
- ► LCP: Pardalos–Rosen Mixed Integer Formulation
- ► Logconcave Measures, Logconvexity
- ► Logconcavity of Discrete Distributions
- ► L-shaped Method for Two-stage Stochastic Programs with Recourse
- ► Mixed Integer Classification Problems
- ► Multi-objective Integer Linear Programming
- ► Multi-objective Mixed Integer Programming
- ► Multistage Stochastic Programming: Barycentric Approximation
- ► Preprocessing in Stochastic Programming

- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Time-dependent Traveling Salesman Problem
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Bienstock D, Shapiro JF (1988) Optimizing resource acquisition decisions by stochastic programming. Managem Sci 34:215–229
2. Birge JR, Dempster MAH (1996) Stochastic programming approaches to stochastic scheduling. J Global Optim 9:383–409
3. Birge JR, Louveaux FV (1997) Introduction to stochastic programming. Springer, Berlin
4. Caroe CC, Schultz R (1999) Dual decomposition in stochastic integer programming. Oper Res Lett 24(1-2):37-45
5. Caroe CC, Tind J (1998) L-shaped decomposition of two-stage stochastic programming with integer recourse. Math Program 83:451-464
6. Klein Haneveld WK, Stougie L, van der Vlerk MH (1995) An algorithm for the construction of convex hulls in simple integer recourse programming. Ann Oper Res 56:209–224
7. Laporte G, Louveaux FV (1993) The integer L-shaped method for stochastic integer programs with complete recourse. Oper Res Lett 13:133–142
8. Lenstra JK, Rinnooy Kan AHG, Stougie L (1984) A framework for the probabilistic analysis of hierarchical planning systems. Ann Oper Res 1:23–42
9. Louveaux FV (1986) Stochastic programming with block-separable recourse. Math Program Stud 28:48–62
10. Louveaux FV (1993) Stochastic location analysis. Location Sci 1:127–154
11. Louveaux FV, van der Vlerk MH (1993) Stochastic programming with simple integer recourse. Math Program 61:301–325
12. Schultz R (1995) On structure and stability in stochastic programs with random technology matrix and complete integer recourse. Math Program 70:73–89
13. Stougie L (1987) Design and analysis of algorithms for stochastic integer programming. CWI Tract 37
14. Stougie L, van der Vlerk MH (1997) Stochastic integer programming. In: Dell'Amico M, Maffioli F, Martello S (eds) Annotated Bibliography in Combinatorial Optimization. Wiley, New York
15. Vanslyke R, Young Yi (2000) Finite horizon stochastic knapsacks with applications to yield management. Oper Res 48:155–172
16. Wollmer RM (1980) Two-stage linear programming under uncertainty with, 0-1 first-stage variables. Math Program 19:279–288

Stochastic Linear Programming: Decomposition and Cutting Planes

JULIA L. HIGLE, SUVRAJEET SEN
Systems and Industrial Engineering,
University Arizona, Tucson, USA

MSC2000: 90C15, 90C06

Article Outline

Keywords
Introduction
Decomposition of SLP
Statistical Representation of Cutting Plane Coefficients
Stochastic Decomposition

Conclusions
Thanks
See also
References

Keywords

Cutting plane algorithm; Stochastic programming

Stochastic linear programs are typically characterized as extremely large scale linear programs. In general, they cannot be solved without the use of specialized methods that are designed to exploit their special structure. In this chapter, we describe a commonly used decomposition technique due to J.F. Benders, and discuss the manner in which it is used to solve stochastic linear programs. We also discuss the manner in which statistical techniques can be used in combination with Benders' decomposition. This combination forms the basis of the stochastic decomposition algorithm, which is a powerful mechanism for solving large scale stochastic linear programs.

Introduction

In deterministic activity analysis, planning consists of choosing activity levels which satisfy resource constraints while maximizing total profit (or minimizing total cost). Note that all the information necessary for decision making is assumed to be available at the time of planning. Under uncertainty, not all the information is available, and parameters such as resources are often modeled by random variables. However, in the absence of appropriate modeling and algorithmic tools for planning under uncertainty, practitioners have often resorted to using deterministic methodology by replacing the random variables by their expected values. In general, this is inappropriate. In circumstances where all the information is not known with certainty, it is advisable to plan only those activities that cannot be postponed to a future date, while some others may be postponed until better information becomes available. Since information is revealed sequentially over time, decision making under uncertainty naturally becomes a multistage process. The earliest LP models for planning under uncertainty may be credited to G.B. Dantzig [3] and E.M.L. Beale [1], and are often referred to as *two-stage stochastic programs with recourse*.

A two-stage stochastic linear program with recourse may be stated as

$$(SLP)\begin{cases}\min & cx + \mathsf{E}[h(x,\widetilde{\omega})]\\ \text{s.t.} & Ax = b,\\ & x \geq 0,\end{cases}$$

where $\widetilde{\omega}$ is a random variable defined on the probability space $(\Omega, \mathcal{A}, \mathsf{P})$, and for each $\omega \in \Omega$,

$$h(x,\omega) = \begin{cases}\min & g_\omega y\\ \text{s.t.} & W_\omega y = r_\omega - T_\omega x\\ & y \geq 0.\end{cases} \tag{1}$$

Note that the randomness in data elements appears in the second stage, whereas data in the first stage is assumed to be known with certainty.

Two-stage stochastic linear programs arise in a variety of settings. They commonly appear in situations in which the first-stage decision, x, corresponds to a long term, or 'planning', decision that must be made immediately (i. e., prior to any realization of $\widetilde{\omega}$). Following the implementation of this decision, one is faced with a collection of short term, or 'operational', decisions, which vary with the outcome of $\widetilde{\omega}$. Thus, for example, in a manufacturing environment one might make decisions regarding the acquisition of productive capacity. These, of course, are long term decisions made prior to knowing the precise nature of the demand profile. Actual production decisions are made after information regarding demand has been revealed. As such, these decisions are short term. Naturally, the objective function will attempt to strike a balance between the two types of costs.

Note that the explicit representation of $\mathsf{E}[h(x,\widetilde{\omega})]$ requires the solution of (1) for each possible outcome of $\widetilde{\omega}$. Thus, problems such as SLP are typically quite large – too large to solve directly as linear programs. It follows that the efficient solution of SLP requires the development of specialized solution procedures that exploit the structure of the problem. In order to bring the problem to a computationally viable size, decomposition schemes are used. As problem size increases, these schemes are often used in conjunction with statistically motivated methods. Thus, in this article, we will explore the development of cutting plane methods based on a Benders' decomposition of SLP, and will discuss

the development of deterministic cutting planes, as well as their statistically motivated counterparts.

Decomposition of SLP

For brevity in exposition, we assume throughout that $h(x, \widetilde{\omega}) < \infty$ with probability one. This is equivalent to assuming that for all x satisfying $Ax = b$, $x \geq 0$, (1) is almost surely feasible. In the stochastic programming literature, this property is known as 'relatively complete recourse'. As stated in (1), the function $h(x, \omega)$ is easily verified as a convex function of x (see [9]). It follows that the recourse function,

$$\mathsf{E}[h(x, \widetilde{\omega})] = \int_{\Omega} h(x, \omega)\, \mathsf{P}(d\omega)$$

is also a convex function of x. As such, it lends itself to solution via Kelley's cutting plane algorithm [7]. Define

$$f(x) = cx + \mathsf{E}[h(x, \widetilde{\omega})]$$
$$\mathbf{X} = \{x\colon\ Ax = b,\ x \geq 0\}.$$

Assuming $\mathbf{X}$ is bounded, Kelley's algorithm may be stated as:

0 $x^1 \in \mathbf{X}$ is given, $f^0(x) = -\infty$, $k \leftarrow 0$.
1 $k \leftarrow k + 1$.
Find (α^k, β^k) such that $\forall x \in \mathbf{X}$:
$f(x^k) = \alpha^k + (c + \beta^k)x^k$,
$f(x) \geq \alpha^k + (c + \beta^k)x$.
2 $f_k(x) = \max\{f_{k-1}(x),\ \alpha^k + (c + \beta^k)x\}$.
3 Solve $\min_{x \in \mathbf{X}} f_k(x)$ to obtain x^{k+1}.
Repeat from Step 1.

Kelley's cutting plane algorithm

While it is not difficult to include a stopping rule in Kelley's method, we have not done so in the above statement because we wish to draw parallels between deterministic and stochastic cutting plane methods. We note that stopping rules for stochastic methods are beyond the scope of this article.

The manner in which (α^k, β^k) are specified in Step 1 of the algorithm is critical to ensuring that an optimal solution to $\min\{f(x)\colon x \in \mathbf{X}\}$ is eventually identified through Step 3 of the algorithm. Coefficients of the supporting hyperplane required in Step 2 may be obtained from a dual solution of (1). That is, assuming that (1) has a finite optimum, we have

$$h(x, \omega) = \begin{cases} \max & \pi(r_\omega - T_\omega x) \\ \text{s.t.} & \pi W_\omega \leq g_\omega. \end{cases} \tag{2}$$

Thus, if $\overline{x} \in \mathbf{X}$ is given and if we let $\pi_\omega \in \arg\max\{\pi(r_\omega - T_\omega \overline{x})\colon\ \pi W_\omega \leq g_\omega\}$ then

$$h(x, \omega) \geq \pi_\omega(r_\omega - T_\omega x), \quad \forall x \in \mathbf{X},$$

with equality ensured if $x = \overline{x}$. Note that π_ω actually depends on both $\overline{x}$ and ω. Thus, given x^k and ω, let

$$\pi_\omega^k \in \arg\max\left\{\pi(r_\omega - T_\omega x^k)\colon\ \pi W_\omega \leq g_\omega\right\}$$

and note that we may obtain the subgradient coefficients in Step 2 of Kelley's method using

$$\alpha^k = \int_{\Omega} \pi_\omega^k r_\omega\, \mathsf{P}(d\omega),$$
$$\beta^k = -\int_{\Omega} \pi_\omega^k T_\omega\, \mathsf{P}(d\omega),$$

or equivalently

$$\alpha^k = \mathsf{E}[\pi_{\widetilde{\omega}}^k r_{\widetilde{\omega}}], \quad \beta^k = -\mathsf{E}[\pi_{\widetilde{\omega}}^k T_{\widetilde{\omega}}]. \tag{3}$$

Note that this algorithm can be interpreted as a decomposition method for block angular linear programs. Hence it is sometimes referred to as Benders' decomposition [2]. In the stochastic programming literature, this is also known as the *L-shaped method* [8].

In order to appreciate the computational challenges inherent to the solution of SLP, it is important to recognize the magnitude of the requirements associated with (3). Specifically, note that the subgradient coefficients specified in (3) require the implicit solution of the linear program in (1)-(2) for *every* possible realization of the random variable $\widetilde{\omega}$. If there are only a few possible realizations, this poses no computational burden. However, in most cases this exact evaluation easily exceeds computational capabilities. For example, if there are only 10 independent random variables with 3 outcomes each, then there are a total of 3^{10} or 59,049 possible outcomes. This figure represents the number of linear programs that would have to be solved *in each iteration* of Kelley's method. To solve problems of realistic sizes, it is

necessary to resort to approximations of the subgradient coefficients. Given their representation as expected values in (3), it is natural to resort to statistically based approximation schemes.

Statistical Representation of Cutting Plane Coefficients

The most simplistic use of sampled data within the cutting plane coefficients is quite straightforward and easily implemented. Unfortunately, it is prone to substantial error, and should not, in general be used except with caution. We present it here only as an introduction to a more stable methodology.

We begin by noting that one may obtain statistical estimates of the cutting plane coefficients by using randomly sampled observations of $\widetilde{\omega}$ and computing the appropriate sample means. That is, suppose that $\{\omega^t\}_{t=1}^n$ is a collection of independent and identically distributed observations of $\widetilde{\omega}$ and $\pi_t^k \in \arg\max\{\pi(r^t - T^t x^k : \pi W^t \le g^t\}$, where $(r^t, T^t, g^t, W^t) = (r_{\omega^t}, T_{\omega^t}, g_{\omega^t}, W_{\omega^t})$. Then the sample means

$$\widehat{\alpha}^k = \frac{1}{n}\sum_{t=1}^{n} \pi_t^k r^t, \quad \widehat{\beta}^k = \frac{-1}{n}\sum_{t=1}^{n} \pi_t^k T^t$$

can be used as estimates of the cutting plane coefficients. Application of Kelley's method using these estimated cut coefficients is equivalent to solving

$$\begin{cases} \min & cx + \dfrac{1}{n}\displaystyle\sum_{t=1}^{n} h(x, \omega^t) \\ \text{s.t.} & Ax = b, \\ & x \ge 0. \end{cases} \tag{4}$$

If we let $\overline{x}_n$ denote an optimal solution to (4), then it is clear that $\overline{x}_n$ need not be an optimal solution to SLP. Moreover, it is also clear that $\overline{x}_n$ will depend on the sample used – different sets of observations will lead to different solutions.

The drawback to simply solving (4) in place of SLP lies in the inability to judge the quality of the solutions produced. That is, if x^* denotes an optimal solution to SLP, one would naturally be interested in assessing $f(\overline{x}_n) - f(x^*)$. This turns out to be a fairly difficult undertaking, and can be computationally intensive (see [5]). In addition, we note that cutting plane algorithms commonly generate cuts in early iterations which support the objective function 'peripherally' (i. e., in regions that are far removed from the optimal solution). With that observation, we note that it is possible to ease the computational effort required by using less accurate cuts in the early iterations. The *stochastic decomposition algorithm* [4] was designed to circumvent these drawbacks.

Stochastic Decomposition

Recognizing that cutting planes derived early in the iterative process will tend to have little bearing on the optimal solution, stochastic decomposition (SD) iterates with a variable sample size. As iterations progress, the sample size used increases. That is, in the kth iteration, k observations are used. This requires the generation of one new observation of $\widetilde{\omega}$ in each iteration. In addition, SD creates computational efficiencies by using approximation of the subproblem (2).

In the kth iteration, the SD algorithm approximates one support of the following function:

$$\frac{1}{k}\sum_{t=1}^{k} h(x, \omega^t).$$

Note that cuts generated in earlier iterations were based on fewer observations of $\widetilde{\omega}$. In order to ensure that all cuts are asymptotically valid (i. e., underestimate the actual objective function, as required in Step 2 of Kelley's method), the SD algorithm updates previous cuts by including a lower bounding constant. For example, suppose that $h(x, \widetilde{\omega}) \ge 0$ (with probability 1). Then

$$\begin{aligned} &\frac{1}{k}\sum_{t=1}^{k} h(x, \omega^t) \\ &= \frac{k-1}{k}\left\{\frac{1}{k-1}\sum_{t=1}^{k-1} h(x, \omega^t)\right\} + \frac{1}{k} h(x, \omega^k) \\ &\ge \frac{k-1}{k}\left\{\frac{1}{k-1}\sum_{t=1}^{k-1} h(x, \omega^t)\right\}. \end{aligned} \tag{5}$$

Now, suppose that in the first $k-1$ iterations we have accumulated a collection of cutting plane coefficients,

$\{(\alpha_t^{k-1}, \beta_t^{k-1})\}_{t=1}^{k-1}$, such that

$$\frac{1}{k-1}\sum_{t=1}^{k-1} h(x,\omega^t) \geq \alpha_t^{k-1} + \beta_t^{k-1}x,$$
$$\forall x \in \mathbf{X}, \quad t = 1,\ldots,k-1.$$

Combining the above inequality with (5), we have

$$\frac{1}{k}\sum_{t=1}^{k} h(x,\omega^t) \geq \frac{k-1}{k}\{\alpha_t^{k-1} + \beta_t^{k-1}x\},$$
$$\forall x \in \mathbf{X}, \quad t = 1,\ldots,k-1.$$

This leads to a simple mechanism for updating previously derived cutting plane coefficients which preserves the required lower bounding nature as iterations progress (and hence, as the sample size increases). That is, we simply require

$$\alpha_t^k \leftarrow \frac{k-1}{k}\alpha_t^{k-1}, \quad \beta_t^k \leftarrow \frac{k-1}{k}\beta_t^{k-1}, \quad t = 1,\ldots,k-1\,.$$

Of course, if the lower bound is given as $\ell \neq 0$, then we use

$$\alpha_t^k \leftarrow \frac{k-1}{k}\alpha_t^{k-1} + \frac{1}{k}\ell,$$

and the update of β_t^k is not altered.

To illustrate the subproblem approximation, suppose that $g_{\widetilde{\omega}}$ and $W_{\widetilde{\omega}}$ are constant so that $g_\omega = g$ and $W_\omega = W$ for all $\omega \in \Omega$. In this case, the set of dual feasible solutions in (2) is the same for all $\omega \in \Omega$, so that

$$h(x,\omega) = \max\{\pi(r_\omega - T_\omega x)\colon\ \pi W \leq g\}\,.$$

Noting that we may restrict our attention to extreme point solutions, let V denote the set of extreme points of $\{\pi\colon \pi W \leq g\}$. Then

$$h(x,\omega) = \max\{\pi(r_\omega - T_\omega x)\colon\ \pi \in V\}\,.$$

SD iteratively constructs a subset of V based upon observed dual solutions. Thus, if $V_k \subset V$ is the subset as it appears in the kth iteration, then SD estimates the cutting plane coefficients using

$$\pi_t^k \in \arg\max\left\{\pi(r^t - T^t x^k)\colon\ \pi \in V_k\right\}.$$

Unlike the simplistic sample-based method previously described, specific guarantees of optimality can be obtained through stochastic decomposition. The expository details associated with safeguards can be somewhat lengthy, and thus, [4] and [6] for a detailed explanation.

Conclusions

The representation of uncertainty in linear programming models easily leads to problems of extremely large magnitude. As such, the ability to decompose stochastic linear programs is critical to the development of viable solution procedures. Indeed, Benders' decomposition, and the cutting planes derived from it, lie at the heart of a wide variety of stochastic linear programming solution methods. Moreover, as the number of possible outcomes associated with the random variables increases, it is necessary to incorporate additional techniques. One of the most promising avenues of exploration to date involves the incorporation of random sampling methods within a decomposition procedure. This provides a mechanism for combining proven methods for obtaining computational efficiencies. That is, the benefits of using decomposition techniques for large scale linear programs have been well established, as have the benefits of using statistical summaries of sampled data in the estimation of expected values. In the solution of large scale stochastic linear programs, their combination proves to be quite powerful.

Thanks

This work is based on research funded by the National Science Foundation.

See also

- Approximation of Extremum Problems with Probability Functionals
- Approximation of Multivariate Probability Integrals
- Decomposition Principle of Linear Programming
- Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- Extremum Problems with Probability Functions: Kernel Type Solution Methods
- Generalized Benders Decomposition
- General Moment Optimization Problems
- Logconcave Measures, Logconvexity
- Logconcavity of Discrete Distributions
- L-shaped Method for Two-stage Stochastic Programs with Recourse
- MINLP: Generalized Cross Decomposition
- MINLP: Logic-based Methods

- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Simplicial Decomposition
- ▶ Simplicial Decomposition Algorithms
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Successive Quadratic Programming: Decomposition Methods
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Beale EM (1955) On minimizing a convex function subject to linear inequalities. J Royal Statist Soc 17B:173–184
2. Benders JF (1962) Partitioning procedures for solving mixed variables programming problems. Numerische Math 4:238–252
3. Dantzig GB (1955) Linear programming under uncertainty. Managem Sci 1:197–206
4. Higle JL, Sen S (1991) Stochastic decomposition: An algorithm for stage linear programs with recourse. Math Oper Res 16:650–669
5. Higle JL, Sen S (1996) Duality and statistical tests of optimality for two stage stochastic programs. Math Program B 75:257–275
6. Higle JL, Sen S (1996) Stochastic decomposition: A statistical method for large scale stochastic linear programming. Kluwer, Dordrecht
7. Kelley JE (1960) The cutting plane method for convex programs. J SIAM 8:703–712
8. Slyke R Van, Wets RJ-B (1969) L-Shaped linear programs with application to optimal control and stochastic programming. SIAM J Appl Math 17:638–663
9. Wets RJ-B (1974) Stochastic programs with fixed recourse: the equivalent deterministic problem. SIAM Rev 16:309–339

Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions

KARL FRAUENDORFER, MICHAEL SCHÜRLE
Institute Operations Res., University St. Gallen, St. Gallen, Switzerland

MSC2000: 90C15

Article Outline

Keywords
The Jensen Lower Bound
The Edmundson–Madansky Upper Bound (Independent Case)
The Edmundson–Madansky Upper Bound (Dependent Case)
Bounds on Simplices
Improving Bounds
Generalizations and Alternative Methods
See also
References

Keywords

Stochastic programming; Approximation; Bounding the expectation

Stochastic linear programs (SLPs) can be seen as a generalization of linear programming problems where at least some coefficients in the objective function and/or

the constraints are random. The motivation for such a formulation is that in many practical applications, the problem data are not known with certainty, for instance because they represent information about the future.

As an example, consider a problem statement from the area of production planning. Today (at a first stage), a decision maker has to decide upon an input plan x which yields goods Tx by means of some technological process in order to meet the uncertain demand h in the future (a second stage). Since it is likely that the number of produced goods fails to meet the demand, a recourse action y is required that allows to compensate the discrepancy '$h - Tx$' as soon as the demand is known. Such a correction induces additional cost $q'y$ in the second stage. The objective is to find a decision x that minimizes the direct cost $c'x$ of the first stage plus the expected cost $\mathcal{Q}(x)$ induced by x for compensations. Formally, this can be written as

$$\begin{cases} \min & c'x + \mathcal{Q}(x) \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{cases} \tag{1}$$

where $\mathcal{Q}(x) = \int_{\Xi} Q(x, \xi)\, dF(\xi)$ is the expectation of the *recourse function*

$$Q(x, \xi) = \min_{y} \{q'y : \; Wy = h(\xi) - T(\xi)x\}. \tag{2}$$

The second-stage cost (2) depends on x and the realization of some K-dimensional random vector ξ with values in the compact convex set $\Xi \subset \mathbf{R}^K$ and distribution function F. Note that $Q(x, \xi)$: $\Xi \to \mathbf{R}$ is convex for all x satisfying the constraints in (1). While the recourse action y may be different for each ξ, the first-stage decision x is independent of which event actually occurs. This property is known as *nonanticipativity*. The meaning is that the current decision is only based on what is known today.

Throughout this article, it is assumed that the cost coefficients q as well as the recourse matrix W are fixed since the penalization for compensating the discrepancy $h(\xi) - T(\xi)x$ is likely to be of a deterministic nature. The case of a nonrandom W is known as *fixed recourse*. In particular, $W = (-I, I)$ with I being the identical matrix is called *simple recourse*, meaning that each deviation of $h(\xi) - T(\xi)x$ from zero is penalized by its absolute value. However, in a more general formulation both q and W may also depend on the realization of ξ (see [1,12] for a more thorough discussion). Furthermore, it is assumed that the demand h and the technology matrix T are linear-affine in ξ, i. e.

$$h(\xi) = h_0 + h_1\xi_1 + \cdots + h_K\xi_K,$$
$$T(\xi) = T_0 + T_1\xi_1 + \cdots + T_K\xi_K.$$

If the distribution of ξ is discrete, the *stochastic linear program with recourse* given by (1) and (2) can be written as a large deterministic problem where the expectations are written as a finite sum, and all constraints are duplicated for each realization of ξ. The resulting *deterministic equivalent problem* may be solved by straightforward application of standard linear programming methods (provided that the discrete set of possible outcomes for ξ is of relatively low cardinality). However, it exhibits a typical block structure that can be exploited by special *decomposition algorithms* (see also [1,12] or [13]).

Otherwise, if the distribution of ξ is continuous, one can use *approximation techniques* ([2,10,11]) where the original random vector ξ is replaced by another one $\widehat{\xi}$. Typically, $\widehat{\xi}$ is discrete, and the problem reduces to a discretely distributed stochastic program. These techniques take advantage of the convexity of the recourse function, yielding upper and lower bounds for the expected recourse cost. This allows to quantify the accuracy of the approximation and, if not sufficient, to improve it by constructing a better approximation to $\mathcal{Q}(x)$.

The Jensen Lower Bound

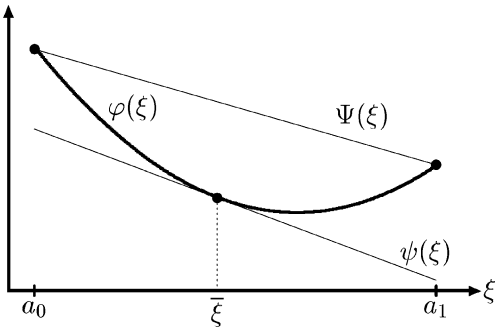
In order to outline the basic concepts (see e. g. [12]), assume that ξ is one-dimensional and denote the expectation by $\overline{\xi} := \mathsf{E}\xi = \int_{\Xi} \xi\, dF(\xi)$. Recall that $\varphi(\xi) := Q(\widehat{x}, \xi)$ is convex in ξ for any fixed $\widehat{x}$. Therefore, it can be bounded from below by a linear function $\psi(\xi)$ that supports φ at some point $\widehat{\xi}$, i. e. (assuming that φ is (sub)differentiable in $\widehat{\xi}$)

$$\psi(\xi) = \varphi(\widehat{\xi}) + \varphi'(\widehat{\xi})(\xi - \widehat{\xi}).$$

Due to linearity, the expected value of this lower bound is given by

$$E\psi(\xi) = \varphi(\widehat{\xi}) + \varphi'(\widehat{\xi})(\mathsf{E}\xi - \widehat{\xi}) = \psi(\overline{\xi}).$$

Obviously, the best lower bound is given by $\widehat{\xi} = \overline{\xi}$ since no linear function supporting φ has a value larger than



Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions, Figure 1

$\varphi(\overline{\xi})$ in $\overline{\xi}$. This is stated by *Jensen's inequality* $\varphi(\mathsf{E}\xi) \leq \mathsf{E}\varphi(\xi)$ for convex functions of random variables. Applied to the situation where $\varphi(\xi)$ is convex over the domain of ξ, one obtains a lower bound to the recourse function from

$$\varphi(\overline{\xi}) \leq \int_{\Xi} \varphi(\xi)\, dF(\xi). \tag{3}$$

Note that (3) also holds for multidimensional $\xi \in \Xi \subset \mathbf{R}^K$, $K \geq 1$, regardless of any correlation between the components of ξ.

For one-dimensional random variables, the lower bounding function $\psi(\xi)$ is illustrated in Fig. 1, together with an upper bound which can be derived as follows.

The Edmundson–Madansky Upper Bound (Independent Case)

For simplicity, the one-dimensional case is considered first. Let the support of ξ be given by the interval $\Xi = [a_0, a_1] \subset \mathbf{R}$. The idea of the *Edmundson–Madansky upper bound* is to introduce a discrete random variable $\widehat{\xi}$ with the same expectation, i. e. $\mathsf{E}\widehat{\xi} = \overline{\xi}$, attaining the values a_0 and a_1 with probabilities

$$p_{a_0} := \mathsf{P}(\widehat{\xi} = a_0) = \frac{a_1 - \overline{\xi}}{a_1 - a_0},$$
$$p_{a_1} := \mathsf{P}(\widehat{\xi} = a_1) = \frac{\overline{\xi} - a_0}{a_1 - a_0} = (-1)\frac{a_0 - \overline{\xi}}{a_1 - a_0}.$$

Obviously, the linear function $\Psi(\xi)$ through the points $(a_0, \varphi(a_0))$ and $(a_1, \varphi(a_1))$ is above $\varphi(\xi)$ for all $\xi \in \Xi$ due to convexity. This implies

$$\varphi(\xi) \leq \frac{a_1 - \xi}{a_1 - a_0} \cdot \varphi(a_0) + \frac{\xi - a_0}{a_1 - a_0} \cdot \varphi(a_1)$$

for all $\xi \in \Xi$, and integrating this inequality yields

$$\begin{aligned} &\int_{\Xi} \varphi(\xi)\, dF(\xi) \\ &\leq \frac{a_1 - \overline{\xi}}{a_1 - a_0} \cdot \varphi(a_0) + \frac{\overline{\xi} - a_0}{a_1 - a_0} \cdot \varphi(a_1) \\ &= p_{a_0} \cdot \varphi(a_0) + p_{a_1} \cdot \varphi(a_1) = \mathsf{E}\varphi(\widehat{\xi}) \end{aligned}$$

as an upper bound for the expectation $\mathsf{E}\varphi(\xi)$. The EM-bound on intervals can be extended easily to multivariate distributions, where $\Xi = [a_{10}, a_{11}] \times \cdots \times [a_{K0}, a_{K1}] \subset \mathbf{R}^K$ is the rectangular support of ξ, if either $Q(x, \cdot)$ is separable in the components of ξ or the elements of ξ are stochastically independent. In the former case, the bound may be applied to each component separately. Here, the more general latter case is of interest.

Denote the vertices of Ξ by a^ν, $\nu = (\nu_1, \ldots, \nu_K)$, $\nu_i \in \{0, 1\}$, such that $a_i^\nu = a_{i\nu_i}$. Analogously to the above, $\widehat{\xi}$ is a discrete random vector with independent components and $\mathsf{E}\widehat{\xi}_i = \overline{\xi}_i$, attaining values a^ν with probability

$$\begin{aligned} p(a^\nu) &:= \mathsf{P}\left(\widehat{\xi} = a^\nu\right) \\ &= \frac{\prod_{i=1}^K (-1)^{\nu_i}(a_{i\overline{\nu}_i} - \overline{\xi}_i)}{\prod_{i=1}^K (a_{i1} - a_{i0})}, \end{aligned} \tag{4}$$

where $\overline{\nu}_i = 1 - \nu_i$, and the EM-inequality contains the product of all combinations of each interval bound, i. e.

$$\int_{\Xi} \varphi(\xi)\, dF(\xi) \leq \sum_{\nu} p(a^\nu) \cdot \varphi(a^\nu) = \mathsf{E}\varphi(\widehat{\xi}).$$

The Edmundson–Madansky Upper Bound (Dependent Case)

If the components of ξ are dependent, the EM-bound is more difficult to evaluate (unfortunately, the notation is also somewhat cumbersome). For $\mathcal{B} := \{\Lambda\colon \Lambda \subset \{1, \ldots, K\}$ and $\delta_\Lambda(\nu) := \prod_{i \in \Lambda}(-1)^{\overline{\nu}_i}$, $\Lambda \in \mathcal{B}$,

$$m_\Lambda := \int_{\Xi} \left(\prod_{i \in \Lambda} \xi_i\right) dF(\xi)$$

are the crossmoments of ξ for any $\Lambda \in \mathcal{B}$, and $\rho_\Lambda := m_\Lambda - \prod_{i \in \Lambda} \overline{\xi}_i$. Using these definitions and $\overline{\Lambda} = \{1, \ldots, K\} \setminus \Lambda$, it has been shown in [6], that the prob-

abilities of the discrete outcomes a^ν are determined by

$$p(a^\nu) = \prod_{i=1}^{K}(a_{i1} - a_{i0})^{-1} \times \left[\prod_{i=1}^{K}(-1)^{\nu_i}(a_{i\overline{\nu}_i} - \overline{\xi}_i) + \sum_{\Lambda\in\mathcal{B}}\left(\delta_{\overline{\Lambda}}(\overline{\nu})\prod_{i\in\overline{\Lambda}} a_{i\overline{\nu}_i}\right)\delta_\Lambda(\nu)\rho_\Lambda\right]$$

Obviously, $\rho_\Lambda = 0$ if the components of ξ are independent which is equivalent to the bound in (4). The calculation of the EM-bound requires to evaluate 2^K points and, in the dependent case, the same number of cross-moments. Therefore, for higher dimensions not only the computational effort but also the number of discrete outcomes increases exponentially, yielding a deterministic equivalent problem that might be too large to be solved.

Bounds on Simplices

Instead of rectangles, one can use a regular simplex $\Delta \supset \Xi$ containing the support of ξ. In this case, Jensen's inequality where Ξ is replaced by Δ in (3) can be applied immediately to obtain a lower bound. To derive an upper bound, the affine independent vertices $v_0, \ldots, v_K$ of Δ are considered as discrete outcomes. Since this are only $K + 1$ points, the complexity is no longer exponential in the dimension of ξ. Note that for any $\xi \in \Delta$, the system of linear equations

$$p_0(\xi) + \cdots + p_K(\xi) = 1,$$
$$v_0 p_0(\xi) + \cdots + v_K p_K(\xi) = \xi$$

or briefly

$$Vp(\xi) = \begin{pmatrix}1\\ \xi\end{pmatrix} \quad \text{with } V = \begin{pmatrix}1 & \cdots & 1\\ v_0 & \cdots & v_K\end{pmatrix}$$

has the unique solution

$$p(\xi) = V^{-1}\begin{pmatrix}1\\ \xi\end{pmatrix}.$$

Analogously to the above, a discrete random variable $\widehat{\xi}$ is constructed attaining values $v_0, \ldots, v_K$ with probabilities $p_0, \ldots, p_K$ so that the expectation of the discrete distribution is equivalent to those of the original one, i. e. $\mathsf{E}\widehat{\xi} = p_0 v_0 + \cdots + p_n v_n = \overline{\xi}$. This yields the following version of the Edmundson–Madansky inequality:

$$\int_\Delta \varphi(\xi)\, dF(\xi) \le \sum_{i=0}^{K} p_i \varphi(v_i),$$

where the vector of probabilities is given by

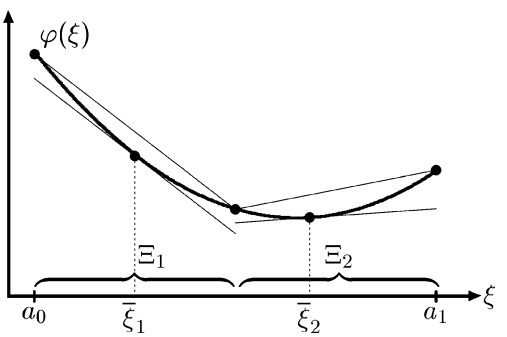
$$p = \int_\Delta V^{-1}\begin{pmatrix}1\\ \xi\end{pmatrix} dF(\xi) = V^{-1}\begin{pmatrix}1\\ \overline{\xi}\end{pmatrix}$$

Note that this equation holds for both the independent and the dependent case.

Improving Bounds

The advantageous feature of approximation techniques is that the accuracy can be quantified by the difference between the Jensen and the Edmundson–Madansky bound. If not sufficient, the approximation can be improved by dividing the rectangular support Ξ (or the simplex Δ, respectively) into convex disjunct subsets like in Fig. 2.

A finite collection of such subsets is called a *partition*. Dividing an element of an existing partition yields a 'refined' partition, and the associated bound is at least as good as the former one (monotonicity of bounds). If the subsets become arbitrary small, the approximated recourse function converges to the original one. However, for computational reasons the partition cannot become arbitrarily small. Also, dividing Ξ (or Δ, respectively) without strategy may increase the computational effort dramatically. Hence, sophisticated refinement strategies are required that analyze the accuracy



Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions, Figure 2

of the bounds for each subset and determine those that should be further divided. This yields a sequential approximation to the original problem.

Generalizations and Alternative Methods

When deriving the upper and lower bounds, only randomness in the right-hand side was taken into account so far. If the latter are deterministic (i. e. $h(\xi) := h_0$) but the coefficients $q(\xi)$ in the objective are uncertain, one can apply the same procedure to the dual problem. Since this is a maximization problem with concave recourse function, the Jensen inequality provides an upper while the Edmundson–Madansky inequality yields a lower bound. For uncertainty in both the objective and the right-hand side of constraints, extensions of the approximation scheme described above are required (see e. g. [7]). It has to be mentioned that there are other concepts apart from the bounds derived here which are also applicable for noncompact support of the random data [4] or derive sharper lower bounds by exploiting second moment information [3].

Alternatively, one may approximate the original problem by sampling from continuous distributions to obtain a deterministic equivalent, or one may use *Benders decomposition* together with variance reduction techniques to handle a large number of scenarios [9]. Other sophisticated approaches like stochastic quasigradient methods [5] or stochastic decomposition [8] combine sampling with techniques known from convex optimization, for example subgradient or cutting plane procedures.

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer, Berlin
2. Birge JR, Wets RJ-B (1986) Designing approximation schemes for stochastic optimization problems, in particular for stochastic programs with recourse. Math Program Stud 27:54–102
3. Edirisinghe NCP (1996) New second-order bounds on the expectation of saddle functions with applications to stochastic linear programming. Oper Res 44:909–922
4. Edirisinghe NCP, Ziemba WT (1994) Bounding the expectation of a saddle function with application to stochastic programming. Math Oper Res 19:314–340
5. Ermoliev Y (1988) Stochastic quasigradient methods. In: Ermoliev Y, Wets RJ-B (eds) Numerical Techniques for Stochastic Optimization. Springer, Berlin, pp 141–185
6. Frauendorfer K (1873) Solving SLP recourse problems with arbitrary multivariate Coefficienten. Math Ann 6:23–28

7. Frauendorfer K (1992) Stochastic two-stage programming. Springer, Berlin
8. Higle JL, Sen S (1996) Stochastic decomposition – a statistical method for large scale stochastic linear programming. Kluwer, Dordrecht
9. Infanger G (1994) Planning under uncertainty. Boyd & Fraser, Danvers
10. Kall P (1998) Bounds for and approximations to stochastic linear programs with recourse. In: Marti K, Kall P (eds) Stochastic programming methods and technical applications. Springer, Berlin, pp 1–21
11. Kall P, Ruszczyski A, Frauendorfer K (1988) Approximation techniques in stochastic programming. In: Ermoliev Y, Wets RJ-B (eds) Numerical Techniques for Stochastic Optimization. Springer, Berlin, pp 33–64
12. Kall P, Wallace SW (1994) Stochastic programming. Wiley, New York
13. Ruszczyski A (1997) Decomposition methods in stochastic programming. Math Program 79:333–353

Stochastic Network Problems: Massively Parallel Solution

Søren S. Nielsen
Department Stat. & Operations Res.,
University Copenhagen, Copenhagen, Denmark

MSC2000: 90B15, 68W10, 90C06, 90C30

Article Outline

Keywords
Problem Formulation
Row-Action Algorithm
Specialization to Quadratic Generalized Network Problems
Proximal Minimization with *D*-functions
Parallelism in the Row-Action Algorithm
Simple Bounds
Disjoint Constraints
Stochastic Problems
Jacobi Algorithms
Massively Parallel Implementations
Extension to Multistage SPs
Computational Experiments
Nonlinear Two-stage Problems
Linear Two-Stage Problems
Multistage Problems
See also
References

Keywords

Stochastic programming; Optimization; Network

Stochastic programming provides a framework for sequential decision making, or planning, under uncertainty. Uncertainty naturally arises when the future consequences of present decisions are unknown, and is typically represented by a set of scenarios covering possible future events. A *stochastic program* (SP) seeks to find a decision which is, in some sense, optimal with respect to the scenario set. Depending upon the number of scenarios, and upon the size of the underlying (deterministic) model, an SP may become very large and computationally challenging to solve. Hence, there is an ongoing effort to device efficient algorithms tailored to the special structures of SPs, and to exploit novel, parallel computer architectures in their solution.

This article focuses on two- and multistage SPs with generalized network structure. The two-stage SP is the simplest program which captures the dynamic decision process, while the network structure arises naturally in many applications, such as financial decision making (e. g., [11]). This problem structure was studied already in [24] and [25]. We first describe algorithms for the solution of network problems with strictly convex objective functions. These algorithms are then adapted to the specially structured stochastic network problems, and are also extended to general, convex (such as linear) objective functions.

Problem Formulation

A *stochastic program* models a situation where a decision maker must make a decision here and now (time 0), facing future uncertainty (the first stage). At later time points, τ_n, further corrective decision ares made. These decisions are made dependently on prior decisions and on the actual realizations of uncertain events between times 0 and τ_n, but in the face of further uncertainty of events after τ_n. The *two-stage* SP consists of the initial and one corrective decision and is naturally generalized to the *multistage* SP with an arbitrary number of decisions.

The two-stage SP where uncertainty is represented by a finite *scenario set* $S = \{1, \ldots, N\}$ with probabilities $p^s > 0$, can be formulated mathematically in the deter-

ministic, equivalent form, [26]:

$$\begin{cases} \min & f(x) + \sum_{s \in S} p^s g^s(y^s) \\ \text{s.t.} & Ax = b, \\ & C^s x + B^s y^s = d^s, \quad \forall s \in S, \\ & 0 \le x \le u^x, \quad \forall s \in S, \\ & 0 \le y^s \le v^x, \quad \forall s \in S. \end{cases}$$

The first- and second-stage variables are denoted x and y^s, respectively. Any (scenario-independent) constraints on the first-stage decision are represented by the $Ax = b$ constraint. The second-stage decisions y^s depend upon both the scenario (hence the superscript s) and on the first-stage decisions x through the matrix C^s. All decision variable may be subject to simple bounds. The objective functions f and g^s are assumed to be convex, and continuously differentiable.

The L-shaped decomposition algorithm [21], based on Benders decomposition, applies directly to SP and is well-suited for course-grained, parallel solution, [17]. We consider here the equivalent *split-variable formulation*, obtained by replicating the first-stage variables x into copies x^s for each scenario, then adding *nonanticipativity constraints* (NA constraints)

$$x^1 = x^s, \quad \forall s \in \{2, \ldots, N\} \tag{1}$$

to force the replicated variables to have the same value:

$$[RNLP] \begin{cases} \min & \sum_{s \in S} p^s (f(x^s) + g^s(y^s)) \\ \text{s.t} & Ax^s = b, \quad \forall s \in S, \\ & C^s x^s + B^s y^s = d^s, \quad \forall s \in S, \\ & 0 \le x^s \le u, \quad \forall s \in S, \\ & 0 \le y^s \le v^s, \quad \forall s \in S, \\ & x^1 = x^s, \quad \forall s \in \{2, \ldots, S\}. \end{cases}$$

The advantage of this formulation is that the problem decomposes into N independent subproblems when the NA constraints are ignored. This fact is exploited algorithmically in the progressive hedging ([10,20]) and the diagonal quadratic approximation ([1,9]) algorithms, as well as in the row-action algorithms discussed above.

Row-Action Algorithm

Row-Action Algorithm The row-action algorithm (RA algorithm) [4] is a primal-dual algorithm for solving the general nonlinear optimization problem

$$[NLP] \begin{cases} \min & F(z) \\ \text{s.t.} & Hz = r, \\ & 0 \le z \le u, \end{cases}$$

where the objective function $F(z)$ is strictly convex and continuously differentiable, $z \in \mathbf{R}^n$ and $H \in \mathbf{R}^{m \times n}$.

A solution to [NLP] consists of real vectors, $z \in \mathbf{R}^n$, $\pi \in \mathbf{R}^m$ and $\lambda \in \mathbf{R}^n$, which satisfy the standard optimality conditions:

- *primal feasibility*: $Hz = r$ and $0 \le z \le u$;
- *dual feasibility*: $\nabla F(z) = -H^\top \pi - \lambda$;
- *complementary slackness*: for $j = 1, \ldots, n$,

$$\lambda_j > 0 \implies z_j = 0,$$
$$\lambda_j < 0 \implies z_j = u_j.$$

Starting from an initial primal-dual point (z, π, λ) that satisfies complementary slackness, the row-action algorithm iteratively operates on a single constraint (or row) at a time, simultaneously updating the primal variables occurring in the constraint, and the constraint's dual price. This update causes the constraint to be satisfied (primal feasibility) while maintaining complementary slackness. The algorithm terminates when primal feasibility is satisfied for all constraints, within some tolerance. The order in which the constraints are operated on is not formally important as long as no constraint is ignored indefinitely, although the ordering may influence the rate of convergence.

The algorithmic step can be viewed as a projection of the current primal iterate upon the hyperplane defined by a nonsatisfied constraint, with a simultaneous update of the constraint's dual price so that the three optimality conditions stated above are satisfied for that constraint. Indeed, let h_i denote the ith row of H, so that $h_i^\top z = r_i$ is the ith constraint from $Hz = r$. Then the update is defined as the solution $z^{\nu+1} \in \mathbf{R}^n$, $\beta \in \mathbf{R}$ to the system

$$\nabla F(z^{\nu+1}) = \nabla F(z^\nu) + \beta \cdot h_i, \tag{2}$$

$$h_i^\top z^{\nu+1} = r_i \tag{3}$$

where ν is the iteration counter, z^ν is the current primal point and the *Bregman parameter* β is the change in the constraint's dual price, π_i. Under mild conditions

(strict convexity and *zone consistency* of F, see [4]), this system can be shown to have a uniquely determined solution that lies in the domain of F. The projection upon a simple bounds constraint is similar: For instance, if $z_j > u_j$, the system

$$\nabla F(z^{\nu+1}) = \nabla F(z^{\nu}) + \beta, \quad z_j^{\nu+1} = u_j$$

defines the next iterate. The algorithm is summarized below, where e_i denotes the ith unit vector.

Initialization
Set $\nu = 0$. Initialize (z^0, π^0, λ^0) such that $\nabla F(z^0) = -H^\top \pi^0 - \lambda^0$.
Projection on equality constraint:
To project on the ith equality constraint, $h_i^\top z = r_i$, solve the equations

$$\nabla F(z^{\nu+1}) = \nabla F(z^{\nu}) + \beta \cdot h_i\,,$$
$$h_i^\top z^{\nu+1} = r_i$$

for $z^{\nu+1} \in \mathbf{R}^n$ and $\beta \in \mathbf{R}$.
Update the dual price: $\pi^{\nu+1} = \pi^{\nu} - \beta \cdot e_i$.
Projection on bounds:
To project on the jth simple bound constraint, $0 \le z_j \le u_j$:
If $z_j^{\nu} < 0$, let β and $z_j^{\nu+1}$ solve:

$$\nabla F(z^{\nu+1}) = \nabla F(z^{\nu}) + \beta \cdot e_j\,,$$
$$z_j^{\nu+1} = 0\,.$$

If $z_j^{\nu} > u_j$, let β and $z_j^{\nu+1}$ solve:

$$\nabla F(z^{\nu+1}) = \nabla F(z^{\nu}) + \beta \cdot e_j\,,$$
$$z_j^{\nu+1} = u_j\,.$$

If $0 < z_j^{\nu} < u_j$, let β and $z_j^{\nu+1}$ solve:

$$\nabla F(z^{\nu+1}) = \nabla F(z^{\nu}) + \beta \cdot e_j\,,$$
$$\beta = \lambda_j^{\nu}\,.$$

Let $\lambda^{\nu+1} = \lambda^{\nu} - \beta \cdot e_j$.
Termination: IF convergence: STOP.
ELSE Set $\nu \leftarrow \nu + 1$ and continue.

Row-action algorithm for [NLP]

We note that the initialization step is usually trivial, by setting $\lambda^0 = -\nabla F(z^0) - H^\top \pi^0$ for any z^0 in the domain of F and any π^0. The convergence test is based on a measure of violation of primal feasibility.

The RA algorithm has found application in a variety of areas, such as matrix estimation, image reconstruction and multicommodity network flow problems. For an extensive textbook treatment of these and related topics, and for further references, see [6].

Specialization to Quadratic Generalized Network Problems

We now specialize the RA algorithm to the case where [NLP] is a network problem with a quadratic objective function,

$$F(z) = \frac{1}{2} z^\top W z + q^\top z.$$

Let the network structure be defined by a graph $G = (V, E)$ with a set V of nodes (or vertices), and a set $E \subseteq V \times V$ of arcs (or edges). Let $\delta_i^+ = \{j \in V: (i, j) \in E\}$ be the set of nodes having an arc coming from i, and $\delta_j^- = \{i \in V: (i, j) \in E\}$ be the set of nodes having an arc going to j, respectively. The decision variables are then the flows z_{ij} from node i to node j, for $(i, j) \in E$. We allow the network to be generalized with arc multipliers $m_{ij} > 0$:

$$[QNP] \begin{cases} \min & \displaystyle\sum_{(i,j)\in E} \frac{1}{2} w_{ij} z_{ij}^2 + q_{ij} z_{ij} \\ \text{s.t.} & \displaystyle\sum_{j\in\delta_j^+} z_{ij} - \sum_{k\in\delta_j^-} m_{ki} z_{ki} = r_i, \\ & \forall i \in N, \\ & 0 \le z_{ij} \le u_{ij}, \quad \forall (i, j) \in E. \end{cases}$$

The elements of W are here denoted by w_{ij} and r_i is the supply at node i (demands are represented as negative supplies).

The algorithmic steps of the RA algorithm can now be stated for [QNP]. For the *flow conservation constraint* for node i,

$$\sum_{j\in\delta_j^+} z_{ij} - \sum_{i\in\delta_j^-} m_{ki} z_{ki} = r_i,$$

(2)–(3) leads to the updating formula for z_{ij}:

$$\begin{cases} z_{ij}^{\nu+1} = z_{ij}^{\nu} + \frac{\beta}{w_{ij}} & \text{for } j \in \delta_i^+, \\ z_{ki}^{\nu+1} = z_{ki}^{\nu} - \frac{\beta \cdot m_{ki}}{w_{ki}} & \text{for } k \in \delta_i^-, \end{cases}$$

where

$$\beta = \frac{r_i - \left(\sum_{j\in\delta_i^+} z_{ij}^{\nu} - \sum_{k\in\delta_i^-} m_{ki} z_{ki}^{\nu}\right)}{\sum_{j\in\delta_i^+} \frac{1}{w_{ij}} + \sum_{k\in\delta_i^-} \frac{m_{ki}^2}{w_{ki}}}. \tag{4}$$

The dual variable for node i is updated by subtracting β from its current value, $\pi_i^{\nu+1} = \pi_i^{\nu} - \beta$.

Similarly, the simple bound constraints lead to the updates:

- if $z_{ij}^{\nu} < 0$: Let $z_{ij}^{\nu+1} = 0$ and $\beta = -w_{ij} z_{ij}^{\nu}$;
- if $z_{ij}^{\nu} > u_j$: Let $z_{ij}^{\nu+1} = u_{ij}$ and $\beta = w_{ij}(u_{ij} - z_{ij}^{\nu})$;
- if $0 < z_{ij}^{\nu} < u_j$: Let $z_{ij}^{\nu+1} = z_{ij}^{ij} + \lambda_{ij}^{\nu}/w_{ij}$ and $\beta = -\lambda_{ij}^{\nu}$.

In each case update $\lambda_{ij}^{\nu+1} = \lambda_{ij}^{\nu} - \beta$.

The *stochastic quadratic network problem*, is now obtained by adding NA constraints (1) to [QNP]:

[SQNP]

$$\begin{cases} \min & \displaystyle\sum_{s\in S}\sum_{(i,j)\in E} p^s \left(\frac{1}{2} w_{ij} (z_{ij}^s)^2 + q_{ij} z_{ij}^s\right) \\ \text{s.t.} & \displaystyle\sum_{j\in\delta_j^+} z_{ij}^s - \sum_{i\in\delta_j^-} m_{ki} z_{ki} = r_i, \\ & \quad \forall i \in N, \\ & z_{ij}^1 = z_{ij}^s, \quad \forall s \in \{2,\ldots,N\}, \\ & \quad \forall (i,j) \in F, \\ & 0 \le z_{ij}^s \le u_{ij}^s, \quad \forall (i,j) \in E. \end{cases}$$

As in [RNLP] the superscripts s denote scenario-dependent quantities. The NA constraints apply to the subset F of (replicated) first-stage variables. A specialization of (2)–(3) upon one such constraint changes the two components of the current iterate $z_{ij}^{1,\nu}$ and $z_{ij}^{s,\nu}$ to the common value

$$z_{ij}^{1,\nu+1} = z_{ij}^{s,\nu+1} = \frac{p^1 z_{ij}^{1,\nu} + p^s z_{ij}^{z,\nu}}{p^1 + p^s}$$

which is their probability-weighted average. However, it was shown in [13] that the values of first-stage variable (i, j) across all scenarios can be updated in a single step — equivalent to an infinite number of projections upon the NA-constraints for $z_{i,j}$ — to their probability-weighted average:

$$z_{ij}^{s,\nu+1} = \sum_{s\in S} p^s z_{ij}^{s,\nu}, \quad \forall s \in S. \tag{5}$$

This observation leads to considerably faster agreement among scenarios.

Proximal Minimization with *D*-functions

The row-action algorithm only applies to problems with a strictly convex objective function. Consider the linear program

$$[LP] \begin{cases} \min & c^{\top} z \\ \text{s.t.} & z \in X, \end{cases}$$

where the *feasible region* is denoted by

$$X = \{z \in \mathbf{R}^n : \ Hz = r \text{ and } 0 \le z \le u\}.$$

The *proximal minimization with D-functions algorithm* (PMD) solves linear programs by perturbing the objective into a strictly convex form, then solving the resulting subproblem using RA. The process is repeated with updated perturbations until the solution to the original LP is approached. PMD was proposed in [5], where its convergence was established.

Let $S \neq \emptyset$ be an open convex set. Let f: $\Lambda \subseteq \mathbf{R}^n \to \mathbf{R}$ be an *auxiliary function*. We assume that $\overline{S} \subseteq \Lambda$, where $\overline{S}$ is the closure of S, and that f is strictly convex and continuous on $\overline{S}$ and continuously differentiable on S. The set S is called the *zone* of f. The *D-function* corresponding to f is defined as

$$D_f(x, y) = f(x) - f(y) - \nabla f(y)^T (x - y). \tag{6}$$

For some suitable choice of the auxiliary function f and a positive nondecreasing sequence $\{\gamma^k\}_{k=0}^{\infty}$ with $\liminf_{k\to\infty} \gamma^k = \gamma < \infty$, the proximal minimization algorithm with D-functions proceeds from an arbitrary starting point, $z^0 \in S$, with the following iteration (the PMD algorithm):

$$z^{k+1} \leftarrow \arg\min_{z\in X\cap\overline{S}} F(z) + \frac{1}{\gamma^k} D_f(z, z^k). \tag{7}$$

PMD was used in [14] and [15] to solve linear and stochastic network problems using two auxiliary functions: the quadratic function,

$$f_Q(x) = \frac{1}{2} x^{\top} x, \tag{8}$$

with zone $\mathbf{R}^n$ and D-function

$$D_{f_Q}(x, y) = \frac{1}{2}(x - y)^{\top}(x - y),$$

and the negative of Shannon's entropy function

$$f_E(x) = \sum_{j=1}^{n} x_j \cdot \log x_j \tag{9}$$

with the positive orthant $S = \{x \in \mathbf{R}^n : x_j > 0\}$ as zone and with D-function

$$D_{f_E}(x, y) = \sum_{j=1}^{n} x_j \left[\log \frac{x_j}{y_j} - 1\right] + \sum_{j=1}^{n} y_j.$$

With the quadratic auxiliary function f_Q, PMD specializes to the familiar *quadratic proximal point algorithm* (QPP algorithm) of [18,19]:

$$z^{k+1} \leftarrow \arg\min_{z \in X} F_Q(z) \doteq e^T z + \frac{1}{2\gamma^k} \left\| z - z^k \right\|^2.$$

QPP hence consists of solving a series of quadratic programs (using, e. g., RA) while iteratively updating the proximal point z^k.

QPP and the corresponding algorithm obtained from PMD by using f_E, the *entropic proximal point algorithm* EPP, [7], were implemented and compared with exact algorithms for deterministic network problems in [15]. It was found that while neither algorithm was comparable to specialized, exact network algorithms (based on simplex or relaxation) for small or medium-sized problems, the PMD algorithms were able to solve extremely large problems, with up to 16 million variables, which could not be solved using the exact algorithms.

Parallelism in the Row-Action Algorithm

The RA algorithm lends itself naturally to parallel execution on a computer with a large number of interconnected processors. In the context of stochastic network problems the potential for parallelism manifests itself at several levels. The key to parallelizing an algorithm is to identify parts of the algorithm which can be executed simultaneously without interfering with each other. Hence, two calculations can be executed simultaneously (by different processors) if they do not depend upon each other's results, that is, there are no data dependencies between them.

Simple Bounds

The projection on simple bounds constraints is the simplest example of natural parallelism: Each projection changes the value of (at most) one primal and one dual variable. All the n bounds projections of a problem having n variables can be executed in parallel without data dependencies.

Disjoint Constraints

By *disjoint constraints* we mean constraints that do not have primary variables in common. Projections (2)-(3) upon a set of such constraints can be performed simultaneously since the data involved in each projection do not depend upon the other projections. For network problems, equality constraints correspond to nodes and are disjoint for sets of nonadjacent nodes, i. e., nodes that have no arcs in common. The identification of such sets is a graph-coloring problem [28], where nodes with the same 'color' can all be updated simultaneously.

Stochastic Problems

For a stochastic network it is evident that nodes belonging to different scenario subproblems are independent. This is true even for first-stage nodes within our framework of variable-splitting as in RNLP (and is our primary reason for using splitting). The NA projections (5) can in turn be executed in parallel for each (set of replicated) first-stage variable, z^s_{ij}, $(i, j) \in F$.

Jacobi Algorithms

The kinds of parallelism mentioned so far all define algorithms which are equivalent to the strictly sequential RA algorithm, i. e., *Gauss–Seidel algorithms*. In contrast, *Jacobi algorithms* allow simultaneous operations on constraints even if they are not disjoint. This increases the potential for parallelism because projections on all the primal constraints can be calculated in parallel. However, projections on nondisjoint constraints will generally lead to conflicting updates of the (primal) variables common to the constraints. The conflict can be resolved by solving first for the projections for all constraints simultaneously, but retaining only the dual solutions and discarding the primal variables. Then, common values of the primal variables are calculated from the duals using dual feasibility. The convergence of this algorithm, using suitable underrelaxation, is established in [2]. Jacobi algorithms generally allow more parallelism than Gauss-Seidel algorithms but have poorer convergence properties because they operate on partially outdated data.

Massively Parallel Implementations

The RA algorithm was implemented on the massively parallel Connection Machine CM-2, [8], which is a single-instruction, multiple-data (SIMD) computer having up to 65536 processors implemented as 4096 physical chips organized as a 12-dimensional hypercube. Each processor has 32Kbytes local memory, and there is a floating-point unit for each cluster of 32 processors. The processors, which operate at 7MHz, can each simulate a number of *virtual processors*, VPs, which allows the programmer to address the machine as if it had a number of processors required by a specific parallel algorithm.

Stochastic network problems were mapped upon the machine following the scheme of [27] for each scenario network problem. This mapping was found in [12] to be the most efficient data structure. A linearly-organized cluster of VPs were assigned to each node i, consisting of $|\delta_i^+| + |\delta_i^-| + 1$ VPs which calculate, in parallel, dual feasible flows on the node's incident arcs (satisfying arc bounds), and cooperate efficiently in calculating the resulting node surplus/deficit, and β in (4), leading to an updated dual node price. Clusters of processors associated with adjacent nodes then exchange dual prices through a global *send* operation. This operation is the only operation that does not use the efficient hypercube communication pattern.

Each scenario subproblem is represented the same way but in such a way that processors associated with corresponding variables in different scenarios have direct communication links. This allows for efficient calculation of the NA projections (5) across scenarios. The algorithm hence alternates between flow conservation and bounds constraints projections within each scenario network — in parallel across nodes and scenarios — and enforcement of nonanticipativity constraints across scenarios. Experimentation with the balancing between these two constraint types are reported in [13] and [16]; generally 25-100 network iterations between nonanticipativity projections worked well. [16] also reports on a choice of penalty parameter values, γ^k, in (7).

Extension to Multistage SPs

Although two-stage stochastic programs go a long way toward properly incorporating uncertainty, they suffer from the problem of 'anticipativity': At the time of the second-stage decision, all uncertainties, even those beyond the second stage, are known to the program, permitting in effect a super-optimal decision. Addressing the realistic requirement that there should be more than two stages, so that decisions at any (but the last) stage are still made under further uncertainty, leads to *multistage SPs*, MSP.

A T-stage stochastic programming problem can be formulated as follows [3]:

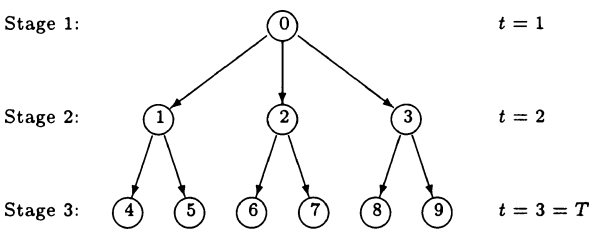
$$
\begin{cases}
\min\limits_{x_1} & c_1 x_1 \\
& \quad + \mathsf{E}_{\xi_2}\left(\min\limits_{x_2} c_2 x_2 \right. \\
& \quad + \mathsf{E}_{\xi_3|\xi_2}\left(\min\limits_{x_3} c_3 x_3 + \cdots \right. \\
& \quad \left. \left. + \mathsf{E}_{\xi_3|\xi_2,\ldots,\xi_{T-1}} \min\limits_{x_T} c_T x_T \right) \right) \\
\text{s.t.} & A_1 x_1 = b_1, \\
& B_2 x_1 + A_2 x_2 = b_2, \\
& \quad B_3 x_2 + A_3 x_3 = b_3, \\
& \qquad \ddots \\
& \qquad\quad B_T x_{T-1} + A_T x_T = b_T, \\
& 0 \le x_t \le u_t \text{ for } t = 1, \ldots, T,
\end{cases}
$$

where

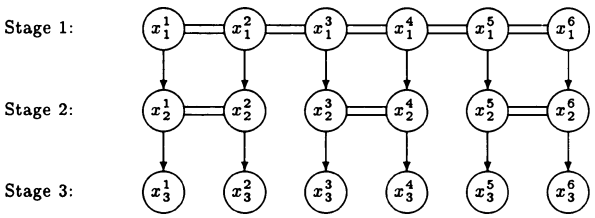
$$\xi_t = (A_t, B_t, b_t, c_t), \quad t = 2, \ldots, T,$$

are random variables, i. e., $\mathcal{F}_t$-measurable functions ξ_t: $A_t \to \mathbf{R}^{M_t}$ on some probability spaces $(\Omega_t, \mathcal{F}_t, \mathcal{P}_t)$.

The decision variables $x_t \in \mathbf{R}^{n_t}$, for $t = 2, \ldots, T$, are stochastic variables measurable on the σ-field generated by ξ_t. The notation E_ξ denotes mathematical expectation with respect to ξ, and $\mathsf{E}_{\xi_i|\xi_j}$ similarly denotes conditional expectation. The sequential nature of the decision process is apparent from this formulation: At stage t, x_t is decided to minimize the expected cost of the subsequent stages, conditional upon events realized up to that stage.



A scenario tree can be used to represent the way in which the stochastic variables ξ_t evolve: The root of the tree corresponds to the immediately observable, deterministic data, A_1, B_1, b_1 and c_1. The nodes of the tree at level $t \geq 2$ corresponds to possible realizations of ξ_t. In this tree, a scenario corresponds to a complete path from the root to a leaf. Similarly to the two-stage case, variables belonging to stages prior to the last are replicated for each scenario, and appropriate NA constraints used to enforce the proper correspondence among scenarios:



The nodes here represent sets of decision variables belonging to various stages (top to bottom) and scenarios (left to right). The double lines represent NA constraints; cf. the scenario tree above. This problem representation maps naturally upon a rectangular communication pattern, as on the CM-2, and was solved, [16], as in the two-stage case, by iterating alternatingly on scenario networks and NA constraints.

Computational Experiments

The algorithms covered in this article were subjected to extensive numerical experimentation on the major types of stochastic network problems:

Nonlinear Two-stage Problems

The row-action algorithm was implemented on the CM-2 and used to solve large scale, quadratic problems in [13]. They report that the algorithm scales very effectively in problem size and number of processors: For instance, doubling both results in virtually the same solution time. The largest problem solved had 8, 192 scenarios and a deterministic, nonlinear equivalent of 868,367 constraints and 2,747,017 variables, and was solved to a tight primal tolerance in about 11 minutes using 32K processors, and achieving a computational rate of 276MFLOPS. The algorithm's performance is sensitive to the range of multipliers occurring in the generalized networks, and deteriorates as this range increases.

Problems of this size could not be solved using any other available algorithm. The RA algorithm was, however, competitive with standard algorithms on smaller problems.

Linear Two-Stage Problems

The PMD algorithm in conjunction with the row-action algorithm was used in [14] on two-stage problems with linear objectives. They conclude that the relative advantage of this algorithm compared to standard codes (Minos 5.3 simplex and OSL interior point) is in the solution of large and very large problems. Solving the largest problems, with 2, 048 scenarios (deterministic equivalent of 217,103 constraints and 618,529 variables), took more than 3 hours of 32K CM-2 processing time, but a problem of this size could not be solved using the simplex or interior point algorithms. It also appears that the PMD/RA solution times scale nearly linearly in problem size, whereas the comparison codes had close to quadratic time complexity. It is apparent that the solution of linear, as opposed to strictly convex, problems takes substantially longer due to the overhead of the nested PMD/RA algorithms.

Multistage Problems

Finally, linear multistage problems with up to 9 stages were solved in [16]. Results mirror those stated above, namely effective scalability and the ability to make progress on very large scale problems, even with the complex nonanticipativity structure of a 9-stage problem. OSL is generally superior to the PMD/RA implementation for small to medium-sized problems, but cannot solve the large instances.

For further material on the parallel and massively parallel solution of large scale stochastic programs, see also [22] and [23].

See also

- ► Approximation of Extremum Problems with Probability Functionals
- ► Approximation of Multivariate Probability Integrals
- ► Asynchronous Distributed Optimization Algorithms

- ▶ Auction Algorithms
- ▶ Automatic Differentiation: Parallel Computation
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ Generalized Networks
- ▶ General Moment Optimization Problems
- ▶ Heuristic Search
- ▶ Interval Analysis: Parallel Methods for Global Optimization
- ▶ Load Balancing for Parallel Optimization Techniques
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Parallel Computing: Complexity Classes
- ▶ Parallel Computing: Models
- ▶ Parallel Heuristic Search
- ▶ Piecewise Linear Network Flow Problems
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Shortest Path Tree Algorithms
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Steiner Tree Problems
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Berger AJ, Mulvey JM, Ruszczyski A (1994) An extension of the DQA algorithm to convex stochastic programs. SIAM J Optim 4(4):735–753
2. Bertsekas DP, Hossein P, Tseng P (1987) Relaxation methods for network flow problems with convex arc costs. SIAM J Control Optim 25:1219–1243
3. Birge JR (1985) Decomposition and partitioning methods for multistage stochastic linear programs. Oper Res 33(5):989–1007
4. Censor Y, Lent A (1981) An iterative row-action method for interval convex programming. J Optim Th Appl 34:321–353
5. Censor Y, Zenios SA (1992) The proximal minimization algorithm with D-functions. J Optim Th Appl 73(3):455–468
6. Censor Y, Zenios SA (1997) Parallel optimization: Theory, algorithms, and applications. Oxford Univ. Press, Oxford
7. Eriksson JR (1985) An iterative primal-dual algorithm for linear programming. Techn Report Dept Math Linköping Univ, no. LiTH-MAT-R-1985-10
8. Hillis WD (1985) The connection machine. MIT, Cambridge, MA
9. Mulvey JM, Ruszczyski A (1992) A diagonal quadratic approximation method for large scale linear programs. Oper Res Lett 12:205–215
10. Mulvey JM, Vladimirou H (1991) Applying the progressive

hedging algorithm to stochastic generalized networks. Ann Oper Res 31:399–424
11. Mulvey JM, Vladimirou H (1992) Stochastic network programming for financial planning problems. Managem Sci 38(11):1642–1664
12. Nielsen SS, Zenios SA (1992) Data structures for network algorithms on massively parallel architectures. Parallel Comput 18:1033–1052
13. Nielsen SS, Zenios SA (1993) A massively parallel algorithm for nonlinear stochastic network problems. Oper Res 41(2):319–337
14. Nielsen SS, Zenios SA (1993) Massively parallel proximal algorithms for solving linear stochastic network programs. Internat J Supercomputer Appl 4:349–364
15. Nielsen SS, Zenios SA (1993) Massively parallel solution of linear network programs. Comput Optim Appl 1(4):375–398
16. Nielsen SS, Zenios SA (1996) Solving multistage stochastic network programs on massively parallel computers. Math Program 73:227–250
17. Nielsen SS, Zenios SA (1997) Scalable parallel Benders decomposition for stochastic linear programming. Parallel Comput 23:1069–1088
18. Rockafellar RT (1976) Augmented Lagrangians and applications to proximal point algorithms in convex programming. Math Oper Res 1:97–116
19. Rockafellar RT (1976) Monotone operators and the proximal point algorithm. SIAM J Control Optim 14:877–898
20. Rockafellar RT, Wets RJ-B (1991) Scenarios and policy aggregation in optimization under uncertainty. Math Oper Res 16(1):119–147
21. Slyke R Van, Wets RJ-B (1966) Programming under uncertainty and stochastic optimal control. SIAM J Control Optim 4:179–193
22. Vladimirou H, Zenios SA (1997) Parallel algorithms for large-scale stochastic programming. In: Migdales A, Pardalos PM, Storöy S (eds) Parallel Computing in Optimization. Kluwer, Dordrecht, pp 413–469
23. Vladimirou H, Zenios SA (1999) Scalable parallel computations for large-scale stochastic programming. Ann Oper Res
24. Wallace SW (1986) Solving stochastic programs with network recourse. Networks 16:295–317
25. Wallace SW, Wets RJ-B (1989) Preprocessing in stochastic programming: The case of uncapacitated networks. ORSA J Comput 1:252–270
26. Wets RJ-B (1974) Stochastic programs with fixed resources: The equivalent deterministic problem. SIAM Rev 16:309–339
27. Zenios SA, Lasken RA (1988) Nonlinear network optimization on a massively parallel connection machine. Ann Oper Res 14:147–165
28. Zenios SA, Mulvey JM (1988) A distributed algorithm for convex network optimization problems. Parallel Comput 6:45–56

Stochastic Optimal Stopping: Numerical Methods

FARID AIT SAHLIA
University of Florida, Gainesville, USA

Article Outline

Introduction
Discrete-Time Models
Continuous-Time Models
References

Introduction

A significant number of optimal stopping problems of practical interest may only be solved through numerical schemes. As many of them have surfaced in the area of mathematical finance, illustrations drawn from that field will be used to describe some of these numerical approaches. Specifically, we consider problems in the expected-value maximization form

$$\sup_{\tau\in\mathcal{T}} E[f(X_\tau,\tau)]\,, \tag{1}$$

where $\mathcal{T}$ is a set of stopping times, f a measurable function and $\{X_t\}_{t\in I}\equiv X$ a Markov process, where I is a time index set that can be either discrete or continuous (see AitSahlia [1] for additional details).

Under technical conditions for its existence, a solution for (1) consists of

- the value function
  $V(x,t)=\sup_{\tau\in\mathcal{T}_t} E[f(X_\tau,\tau)|X_t=x]$, where $\mathcal{T}_t$ is the set of stopping times subsequent to t in $\mathcal{T}$
- the optimal stopping time
  $\tau_t^*=\operatorname{argmax}_{\tau\in\mathcal{T}_t} E[f(X_\tau,\tau)]$.

In this context, with E denoting the state space of X, the set $E\times I$ is partitioned into a closed set S and its complement C labeled, respectively, stopping and continuation regions. Then

$$\tau_t^*=\inf\{s\geq t: X_s\in S\}\,. \tag{2}$$

Discrete-Time Models

When $\mathcal{T}=\{0,1,\ldots,N\}$ for some given $N<\infty$, the most straightforward numerical device is the back-

wards recursive dynamic algorithm

$$V(x, N) = f(x, N)\,, \tag{3}$$

$$V(x, n) = \max\{E[V(X_{n+1}, n+1)|X_n = x], f(x, n)\}, \quad 0 \le n \le N-1\,. \tag{4}$$

An issue with the above might be implementing the proper numerical scheme to estimate $E[V(X_{n+1}, n+1)|X_n = x]$, especially in light of the so-called *curse of dimensionality* that makes this algorithm inefficient in high dimensions. There are potentially two remedies to this problem: one, for finite $\mathcal{T}$, based on Monte-Carlo simulation, and another, for infinite $\mathcal{T}$, based on large-scale linear programming (LP). For the former, an efficient and popular algorithm is that of Longstaff and Schwartz [10], which is now viewed within the wider context of approximate dynamic programming (see also [3,11]). The basic idea of this algorithm is to use Monte Carlo simulation and least-squares regression to estimate $E[V(X_{n+1}, n+1)|X_n = x]$.

For infinite $\mathcal{T}$, the value function is time-homogeneous when X and f are. In this case the value function solves

$$V(x) = \sup_{\tau \in \mathcal{T}_t} E[f(X_\tau)|X_t = x] \tag{5}$$

for all $t \in \{0, 1, \dots\}$ and may be obtained through a LP algorithm thanks to its Snell envelope characterization [1]. Assuming a transition matrix P for X and a finite state space, which might be genuine or the result of a truncation, the resulting LP is

$$\text{Minimize} \quad \sum_x V(x)$$

subject to

$$V(x) \ge \sum_y P(x, y)V(y)\,,$$
$$V(x) \ge f(x)\,,$$
$$V(x) \ge 0\,.$$

See Çinlar [4] and Dynkin and Yushkevich [5] for proofs and further details.

Continuous-Time Models

When both X and its time index I are continuous, there are a number of numerical schemes to generate solutions for (1). Overall, they approximate either the underlying diffusion process X by a discrete version or the value function and its derivatives in its characterizing expression (e. g., integral representation, partial differential equation.)

- Weak-convergence approximation approach: The most general scheme concerning this approach is to approximate the infinitesimal operator $\mathcal{L}$ of X in the free-boundary problem that characterizes the solution of (1). For example, a finite-differences approximation of derivatives in the free-boundary problem

$$\mathcal{L}V = 0 \quad \text{in } C\,,$$
$$V = f \quad \text{on } E \times \{T\}\,,$$
$$\frac{\partial V}{\partial x} = \frac{\partial f}{\partial x} \quad \text{on } \partial S$$

  leads to the formulation of an optimal stopping problem for a Markov chain (see Kushner and Dupuis [8]).
  If the process X is explicitly expressed in terms of Brownian motion, then random-walk approximations can directly be used on the latter. This is a fairly well understood procedure for which rates of convergence have been developed (see Lamberton [9]).
- Integral equation approach: In this scheme, one makes use of the Doob–Meyer decomposition formula for submartingales (see Karatzas and Shreve [7]) to express the value function V in terms of the boundary, which itself solves a related integral equation. For example, consider a case in American option pricing , with horizon T, payoff function $f(x, t) = e^{-rt}\max(K - x, 0)$, and $X_t = X_0 \exp\{(r - \sigma^2/2)t + \sigma W_t\}$, where $K > 0, r > 0$, and $\sigma > 0$ are given and $\{W_t\}_t$ is a standard Brownian motion started at 0. Then the value function V can be decomposed as

$$V(x, t) = U(x, t) + \int_t^T [rK\Phi(-d(X, B(t), \tau - t))]\mathrm{d}\tau\,, \tag{6}$$

  where Φ is the cumulative standard normal distribution function, $d(x, y, \tau) = (\ln(x/y) + (r +$

$\sigma^2/2)\tau)/\sigma\sqrt{\tau} - \sigma\sqrt{\tau}$, and $U(x,t) = Ke^{(T-t)}$ $\Phi(-d(x,K,T-t))$. This formula requires the knowledge of the boundary $B = \partial S$, where S is the stopping region that identifies the optimal stopping time (2), and which is obtained as the solution of the integral equation

$$(K - B(t)) = U(B(t),t) + \int_t^T [rK\Phi(-d(B(t),B(t),\tau - t)]\mathrm{d}\tau \,. \quad (7)$$

Efficient and accurate spline approximations of B can be found in AitSahlia and Lai [2].

- Linear complementarity approach: An alternative that does not require the explicit determination of the optimal stopping boundary relies on the variational inequality formulation

$$\min\{V, V - f\} = 0, \quad \text{on } E \times [0,T) \,,$$
$$V = f, \quad \text{on } E \times \{T\} \,.$$

Finite-difference approximations then lead to a linear complementarity problem (see Huang and Pang [6] and Wilmott et al. [12]).

References

1. AitSahlia F Stochastic optimal stopping: problem formulations, this Encyclopedia
2. AitSahlia F, Lai TL (2001) Exercise boundaries and efficient approximations to American option prices and hedge parameters. J Comput Finance 4:85–103
3. Carrière J (1996) Valuation of early-exercice price of options using simulations and nonparametric regression. Insurance Math Econ 19:19–30
4. Çinlar E (1975) Introduction to Stochastic Processes. Prentice-Hall, Englewood Cliffs
5. Dynkin EB, Yushkevich AA (1969) Markov Processes: Theorems and Problems. Plenum Press, New York
6. Huang J, Pang J-S (1998) Option Pricing and Linear Complementarity. J Comput Finance 2:31–60
7. Karatzas I, Shreve SE (1991) Brownian Motion and Stochastic Calculus. Springer, New York
8. Kushner HJ, Dupuis P (2001) Numerical Methods for Stochastic Control Problems in Continuous Time, 2nd edn. Springer, New York
9. Lamberton D (2002) Brownian optimal stopping and random walks. Appl Math Optim 45:283–324
10. Longstaff FA, Schwartz ES (2001) Valuing American options by simulation: a simple least-squares approach. Rev Financial Stud 14:113–148
11. Tsitsiklis JN, Van Roy B (2001) Regression methods for pricing complex American-style options. IEEE Trans Neural Networks 12(4):694–703
12. Wilmott P, Dewynne JN, Howison S (1993) Option Pricing: Mathematical Models and Computation. Oxford Financial Press, Oxford

Stochastic Optimal Stopping: Problem Formulations

Farid Ait Sahlia
Industrial and Systems Engineering,
University of Florida, Gainesville, USA

Article Outline

Introduction
Definitions
Solution Methods
References

Introduction

A typical stochastic optimal stopping problem of practical interest consists of the following optimization:

$$\sup \mathrm{E}[f(X_\tau,\tau)] \quad \text{s.t.} \quad \tau \in \mathcal{T} \,, \quad (1)$$

where $\{X_t\} \equiv X$ is a stochastic process known as the state process, E its associated expectation operator, f a function measurable with respect to the probability law induced by X, and $\mathcal{T}$ a set of stopping times to be defined shortly. In many applications $f(X_\tau,\tau)$ is interpreted as the gain resulting from stopping at time τ when the state value is X_τ.

A financial example that has been the subject of great interest in mathematical finance/financial engineering is one with $f(x,t) = e^{-rt}\max(K-x,0)$, where $K > 0$ is given, and X is the geometric Brownian process

$$X_t = X_0 \exp\{(r - \sigma^2/2)t + \sigma B_t\} \,,$$

where r and σ are given positive constants and $\{B_t\}$ is a standard Brownian motion started at 0. In finance $f(X_t,t)$ represents the discounted payoff that results from the exercise at time t of a put stock option by its holder who is allowed to sell this stock at share price K when it is traded at price X_t. The option holder's prob-

lem is to find the best time to exercise this option, thus maximizing its payoff, a problem that is mathematically expressed as (1). As will be made precise soon, this optimal exercise time (or more generally stopping time) must be determined only on the basis of pastz observations. It should be mentioned here that $\{B_t\}$ can also be considered the state process, instead of X.

Note that the payoff function f as expressed in (1) does include payoffs that are path-dependent through the usual introduction of additional variables to render a problem Markovian. For example, still in the financial realm, one may consider the payoffs $e^{-rt}(M_t - X_t)$ or $e^{-rt}\max(K - A_t, 0)$ that depend on the maximum process $M_t = \max_{s\le t} X_s$ or the average process $A_t = (1/t)\int_0^t X_s\,\mathrm{d}s$.

Stochastic optimal stopping theory, or optimal stopping as it is customarily known, is a specialized type of the (stochastic) dynamic programming approach devised by Bellman [1] in the 1950s. However, actual optimal stopping problems originated in Wald's work on sequential statistical inference (Wald [4]), where the problem is to determine sequentially the sample size that will decide between two statistical hypotheses. Ever since these early days, this field has experienced several developments in both theory and applications as described for example in the book of Peskir and Shiryaev [3].

Optimal stopping problems are generally approached from a probabilistic perspective through martingales and Markov processes. When the underlying process X in (1) is a diffusion, they also lead to free-boundary problems for partial differential equations. Optimal stopping problems are rarely solved in closed-form and numerical methods abound, a topic addressed in a companion entry in this Encyclopedia.

Definitions

This section sets up basic definitions that lead to the notion of stopping time. As mentioned before, the decision to stop at time t must be based only on information available up to t. In this respect the concept of information set in the form of filter is first formally presented, followed by that of stopping time.

- Discrete-time filtration
  Given a probability space $(\Omega, \mathcal{F}, P)$, a discrete-time filtration is a collection $(\mathcal{F}_n)_{n\ge 0}$ where each $\mathcal{F}_n$ is a σ-algebra of subsets of Ω such that $\mathcal{F}_0 \subset \mathcal{F}_1 \subset \cdots \mathcal{F}$. $\mathcal{F}_n$ represents the information available up to time n. It generally consists of at least the set of events that have been determined by the realized values of X_t up to time n. The latter is called the natural filtration of X and is often augmented to form $(\mathcal{F}_n)_{n\ge 0}$.
- Continuous-time filtration
  Here the definition is essentially identical to the previous modulo a technical condition. Given a probability space $(\Omega, \mathcal{F}, P)$, a continuous-time filtration is a collection $(\mathcal{F}_t)_{t\ge 0}$ where each $\mathcal{F}_t$ is a σ-algebra of subsets of Ω such that $\mathcal{F}_s \subset \mathcal{F}_t \subset \mathcal{F}$ for $s \le$ t. As in the discrete-time case, $\mathcal{F}_t$ also represents information up to time t. Additionally, it is assumed that each $\mathcal{F}_t$ contains all P-null sets in $\mathcal{F}$ and that $(\mathcal{F}_t)_{t\ge 0}$ is right-continuous; i. e., $\mathcal{F}_t = \bigcap_{t\le s} \mathcal{F}_s$ for all $t \ge 0$.
- Stopping time
  Let $I = \{0, 1, 2, \ldots\}$ and $I = [0, \infty]$ when X is, respectively, a discrete-time process and a continuous-time process. A random variable $\tau\colon \Omega \to I$ is a stopping time if $P\{\tau < \infty\} = 1$ and $\{\tau \le t\} \in \mathcal{F}_t$ for all $t \ge 0$. Often the set I is bounded and therefore the former condition is obviously true. The latter condition expresses the fact that the decision to stop at time t must be based solely on information up to time t. In this case τ is adapted to the filtration $(\mathcal{F}_t)_{t\ge 0}$.

Solution Methods

There are generally two approaches to solving (1): one based on probabilistic tools and another on partial differential equations (PDE) techniques. However, both start by using the dynamic programming principle of optimality to derive the so-called Bellman equation. When the interval I is of the form $[0, T]$ or $\{1, 2, \ldots, N\}$ define $\mathcal{T}_t$ to be, respectively, the set of stopping times in $[t, T]$ and $\{t, t+1, \ldots, N\}$. When I is infinite, $\mathcal{T}_t$ is defined as the set of stopping times in I that are $\ge t$. Then solving (1) is tantamount to determining

- the value function $V(x, t) = \sup_{\tau\in\mathcal{T}_t} \mathrm{E}[f(X_\tau, \tau)| X_0 = x]$, and
- the optimal stopping time $\tau_t^* = \mathrm{argmax}_{\tau\in\mathcal{T}_t} \mathrm{E} f(X_\tau, \tau)$.

A sufficient condition guaranteeing the finiteness of the expectation in (1) is

$$E(\sup_{t\in I}|f(X_t,t)|)<\infty\,,$$

which can in fact be relaxed a number of ways.

(I) The Probabilistic Approach: Martingales
When $I=\{0,1,\ldots,N\}$, then by the optimality principle of dynamic programming we can write the recursion

$$V(x,N)=f(x,N) \tag{2}$$

$$V(x,n)=\max\{E[V(X_{n+1},n+1)|X_n=x]\,,\ f(x,n)\}\,,\quad 0\le n\le N-1\,. \tag{3}$$

The solution for the system (2)–(3) induces a sequence of random variables $S_n=V(X_n,n)$ that satisfies the following properties:

(i) $S_n=\max\{E[S_{n+1}|\mathcal{F}_n],f(X_n,n)\}$;
(ii) $(S_n)_{k\le n\le N}$ is the smallest super-martingale that dominates the gain process $G_n=f(X_n,n)_{k\le n\le N}$ (i. e.; $S_n\ge G_n$ P-a.s.);
(iii) the stopping time $\tau_n^*=\inf\{n\le k\le N: S_n=G_n\}$ is optimal for $0\le n\le N$;
(iv) the stopped sequence $(S_{k\wedge\tau_n^*})_{n\le k\le N}$ is a martingale.

We recall here that a discrete-time process $(M_n)_n$ is a martingale with respect to a filtration $(\mathcal{F}_n)_n$ (martingale for short) if $E|M_n|<\infty$ for $n\ge 0$ and $E(M_{n+1}|\mathcal{F}_n)=M_n$, P-a.s., for $n\ge 0$. Correspondingly, $(M_n)_n$ is a super-martingale if $E(M_{n+1}|\mathcal{F}_n)\le M_n$, P-a.s., $n\ge 0$. The process $(S_n)_{k\le n\le N}$ is called the Snell envelope and the above characterization is particularly useful to obtain the value function V through linear programming when the state space is finite (see Çinlar p. 212 in [2]).

The generalization of the above result to the case where I is countably infinite requires that the sequence $S_n=V(X_n,n)$ be characterized differently through the concept of essential supremum below, which generalizes in some sense that of deterministic supremum.

Essential Supremum. Let I be an arbitrary set and $(Z_n)_{n\in I}$ be a collection of random variables defined on the same probability space. Then there exists a countable subset $J\subset I$ such that $Z^*=\sup_{n\in J}Z_n$ satisfies

(a) $Z_n\le Z^*$ P-a.s. for each $n\in I$;
(b) for any other random variable $\tilde{Z}$ such that $Z_n\le\tilde{Z}$ P-a.s. for each $n\in I$, we have $Z^*\le\tilde{Z}$ P-a.s.

The random variable Z^* is labeled *essential supremum* and is denoted by $\operatorname{esssup}_{n\in I}Z_n$.

As a consequence, we can now rewrite the above Snell envelope when $I=\{1,2,\ldots,N\}$ as

$$S_n=\operatorname{esssup}_{\tau\in\mathcal{T}_n}\mathrm{E}[f(X_\tau,\tau)|\mathcal{F}_n]\,,\quad n\in I\,, \tag{4}$$

where $\mathcal{T}_n$ is the set of stopping times in $\{n,n+1,\ldots,N\}$. When I is countable infinite then S_n is correspondingly defined with $\mathcal{T}_n$ as the set of stopping times in $\{n,n+1,\ldots\}$. Similarly, S_n satisfies both conditions (a) and (b) and the optimality property (i) above for all $n\ge 0$.

For the continuous-time case, where I is an interval, the value function for problem (1) is the Snell envelope of the gain process $(f(X_t,t))_t$ defined as

$$S_t=\operatorname{esssup}_{\tau\in\mathcal{T}_t}\mathrm{E}[f(X_\tau,\tau)|\mathcal{F}_t]\,,\quad t\in I\,, \tag{5}$$

where $\mathcal{T}_t$ is the set of stopping times in $[t,T]$ for a finite horizon problem or $[t,\infty)$ otherwise. The Bellman equation in its discrete form (3) is now replaced by

$$V(x,t)\ge\max\{E[V(X_s,s)|X_t=x],f(X_t,t)\},\quad\text{for } s\ge t.$$

Formulation (1) has cast the problem of optimal stopping in a Markovian framework. This is in fact the most common situation in practice and the set-up is not too restrictive as it mirrors well the generic martingale situation fully described in Peskir and Shiryaev [3].

(II) The Probabilistic Approach: Markov Property and Stopping Boundary
When X is a Markov process (in discrete- or continuous-time) with state space E the optimal stopping time is defined as

$$\tau^*=\inf\{t\in I: X_t\in S\}\,,$$

where S is a closed subset in $I\times E$. S and its complement C in $I\times E$ are such that

$$V(x,t)>f(x,t)\text{ on }C\,,$$
$$V(x,t)=f(x,t)\text{ on }S\,.$$

C and S are respectively called the continuation and stopping regions. The intersection B of their closures is called the stopping boundary. It is time-dependent when I is bounded and time-homogeneous when I is unbounded. When I is countably finite and E is discrete, then B can be obtained through the backward recursion (2)–(3).

(III) The PDE Approach

When the state process X is a diffusion the boundary B and the value function V can be obtained by solving a free-boundary problem. Alternatively, when only the value function is of interest then it can be obtained as the solution of a variational inequality. If we let $\mathcal{L}$ the infinitesimal operator associated with X, then assuming regularity and differentiability where necessary, the free-boundary problem when $I = [0, \infty)$ is stated as

$$\begin{aligned} \mathcal{L}V &= 0 \quad \text{in } C\,, \\ \frac{\partial V}{\partial x} &= \frac{\partial f}{\partial x} \quad \text{on } B\,. \end{aligned} \tag{6}$$

The latter condition is called smooth-fit. It is in a sense the condition that characterizes the optimality of a solution V of the PDE (6). When $I = [0, T]$ the free-boundary problem becomes:

$$\begin{aligned} \mathcal{L}V &= 0 \quad \text{in } C\,, \\ V &= f \quad \text{on } E \times \{T\}\,, \\ \frac{\partial V}{\partial x} &= \frac{\partial f}{\partial x} \quad \text{on } B\,. \end{aligned} \tag{7}$$

One way to avoid reference to the free boundary B is through the use of variational equality. For example, the latter problem with finite horizon T can be re-expressed as

$$\begin{aligned} \min\{V, V - f\} &= 0\,, \quad \text{on } E \times [0, T) \\ V &= f\,, \quad \text{on } E \times \{T\}. \end{aligned}$$

References

1. Bellman R (1957) Dynamic Programming. Princeton University Press, Princeton
2. Çinlar E (1975) Introduction to Stochastic Processes. Prentice-Hall, Englewood Cliffs
3. Peskir G, Shiryaev A (2006) Optimal Stopping and Free-Boundary Problems. Birkhäuser Verlag, Basel
4. Wald A (1950) Statistical Decisions Functions. Wiley, New York; Chapman and Hall, London

Stochastic Programming: Minimax Approach

JITKA DUPAČOVÁ
Charles University, Prague, Czech Republic

MSC2000: 90C15, 62C20

Article Outline

Keywords
See also
References

Keywords

Stochastic programming; Incomplete information; Game theory; Minimax decision rule; Worst-case analysis; Bounds

In many applications of stochastic programming there is some uncertainty about the probability distribution P of the random parameters. The *incomplete knowledge of the probability distribution* can be described by assuming that P belongs to a specified class $\mathcal{P}$ of probability distributions. This in turn suggests to use the *minimax decision rule*.

The first results were concerned with stochastic linear programs with recourse; they can be treated within the following more general framework

$$\begin{cases} \min & F(\mathbf{x}; P) := \mathsf{E}_P f(\mathbf{x}; \omega) \\ \text{on} & \text{the set } \mathcal{X} \subset \mathbf{R}^n, \end{cases} \tag{1}$$

with $\mathcal{X}$ a given set of decisions, P a probability distribution on (Ω, Σ), $\Omega \subset \mathbf{R}^m$ and P known to belong to a class $\mathcal{P}$. The random outcome of a decision $\mathbf{x} \in \mathcal{X}$ is quantified by a function f defined on $\mathcal{X} \times \Omega$, E_P denotes the expectation under P.

These results were formulated in terms of the *two-person zero-sum game*

$$(\mathcal{X}, \mathcal{P}, F(\mathbf{x}; P)). \tag{2}$$

M. Iosifescu and R. Theodorescu [11] suggested to use an optimal mixed strategy of the first player in the game (2). J. Žáčková [18] introduced the notion of *minimax solution* as an optimal pure strategy of the first player

in the game (2). Under quite general assumptions on $\mathcal{P}$ and F, a minimax solution exists and

$$\inf_{\mathbf{x}\in\mathcal{X}} \max_{P\in\mathcal{P}} F(\mathbf{x}; P) = \max_{P\in\mathcal{P}} \inf_{\mathbf{x}\in\mathcal{X}} F(\mathbf{x}; P). \tag{3}$$

The minimax decision rule can be applied also in cases when the minimax theorem for the game (2) does not hold true. It means to solve the problem

$$\begin{cases} \min & \max\limits_{P\in\mathcal{P}} F(\mathbf{x}; P) \\ \text{on} & \text{the set } \mathcal{X} \subset \mathbf{R}^n \end{cases} \tag{4}$$

hence, to apply the best possible decision obtained for the most adverse considered circumstances. This provides a tool for the *worst-case analysis* for program (1) and allows for constructing bounds for the optimal value of (1) valid for all $P \in \mathcal{P}$.

Applicability of the results depends on the assumed form of the class $\mathcal{P}$ which describes the level of the available information about the probability distribution of the random parameters and also on the properties of the random objective function $f(\mathbf{x}; \omega)$. Let us list some of the most frequent choices of $\mathcal{P}$:

- $\mathcal{P}$ consists of probability distributions carried by $\Omega \subset \mathbf{R}^m$ which fulfill certain *moment conditions*, e. g.,

$$\mathcal{P} = \{P:\ \mathsf{E}_P g_j(\omega) = y_j,\ j = 1, \dots, J\} \tag{5}$$

  with prescribed values y_j, $\forall j$, [4,6,8,17,18].
- $\mathcal{P}$ contains probability distributions on (Ω, Σ) with fixed marginals [15].
- Additional qualitative information, such as unimodality of P, is taken into account [6,8].
- $\mathcal{P}$ consists of probability distributions P with known finite support, i. e., to specify P means to fix the probabilities of the considered atoms (scenarios) taking into account a prior knowledge about their partial ordering, etc.; see e. g. [2].
- $\mathcal{P}$ is a neighborhood of a hypothetical probability distribution P_0 [4].
- In principle, $\mathcal{P}$ can be also a parametric family of probability distributions with an incomplete knowledge of parameter values.

For convex, compact $\mathcal{P}$, the expectation $F(\mathbf{x}; P) = \mathsf{E}_P f(\mathbf{x}; \omega)$ attains its maximal (and minimal) value at extremal points of $\mathcal{P}$; the extremal probability distributions can be characterized independently of the form of the random objective f, however, the worst-case probability distribution, say, $P^* \in \mathcal{P}$ independent of f (and thus independent of the decisions $\mathbf{x}$) appears only exceptionally. If this is possible the objective function in (4) $\max_{P\in\mathcal{P}} F(\mathbf{x}; P) = F(\mathbf{x}; P^*)$ is just an objective function of a standard stochastic program which is relatively easy to solve due to a relatively simple structure of P^*. There are also instances when one can succeed to get the explicit form of $\max_{P\in\mathcal{P}} F(\mathbf{x}; P)$ [6,12]. They relate to classes of one-dimensional probability distributions and to special functions f.

The general methodology for solution of the inner optimization problem $\max_{P\in\mathcal{P}} F(\mathbf{x}; P)$ for a fixed decision $\mathbf{x}$ has been elaborated in detail for the classes of probability distributions defined by moment conditions (5), both in the form of equations and inequalities: The extremal probability distributions have finite supports, cf. [14,17], and the solution of the inner problem

$$\max_P \int_\Omega f(\mathbf{x}; \mathbf{z})\, dP \tag{6}$$

subject to

$$\begin{cases} \int\limits_\Omega dP = 1, \\ \int\limits_\Omega g_j(\mathbf{z})\, dP = y_j, \quad j = 1, \dots, J, \end{cases} \tag{7}$$

reduces to solution of a generalized linear program (cf. [3,4,7,9,17]), provided that Ω is compact and $f(\mathbf{x}; \cdot)$, g_j, $\forall j$, are continuous on Ω. The procedure provides both the atoms of the sought worst-case probability distribution and their probabilities. In some cases, it is expedient to analyze the dual program to (6), (7), which reads

$$\min_{\mathbf{u}} \sum_{j=1}^{J} u_j y_j + u_0 \tag{8}$$

subject to

$$u_0 + \sum_{j=1}^{J} u_j g_j(\mathbf{z}) \geq f(\mathbf{x}; \mathbf{z}), \quad \forall \mathbf{z} \in \Omega. \tag{9}$$

For details and various applications consult [3,4,5,6,7,8,9,13,17].

As an example, let $f(\mathbf{x}; \cdot)$ be a convex function on a bounded convex polyhedron $\Omega \subset \mathbf{R}^m$, say, $\Omega =$

$\mathrm{conv}\{\omega^{(1)}, \ldots, \omega^{(K)}\}$ and

$$\mathcal{P} = \{P:\ \mathsf{E}_P\omega_j = y_j,\ j = 1, \ldots, m\} \tag{10}$$

with $\mathbf{y}$ a given interior point of Ω. The constraints of (9),

$$u_0 + \sum_{j=1}^{m} u_j z_j \geq f(\mathbf{x}; \mathbf{z}),$$

hold true for all $\mathbf{z} \in \Omega$ if and only if they are fulfilled for the extremal points $\omega^{(1)}, \ldots, \omega^{(K)}$. Duality properties imply that only suitable subsets of the set of extremal points of Ω need to be considered in construction the finite supports of the worst-case distributions. The generalized linear program (6), (7) reduces to the linear program

$$\max_{\mathbf{p}} \sum_{k=1}^{K} p_k f(\mathbf{x}; \omega^{(k)}) \tag{11}$$

subject to

$$\begin{cases} \sum_{k=1}^{K} p_k \omega_j^{(k)} = y_j, & j = 1, \ldots, m, \\ \sum_{k=1}^{K} p_k = 1, & p_k \geq 0 \quad \forall k. \end{cases} \tag{12}$$

Convexity of f with respect to ω is essential for the above result. Generalization to piecewise convex functions $f(\mathbf{x}, \cdot)$ (cf. [5]) is possible; on the other hand, the worst-case probability distribution from the class (10) for f concave in ω is the degenerated distribution concentrated at the prescribed expected value $\mathsf{E}_P\omega$. This degenerated distribution provides the best (i. e., the minimal possible) expectation for convex functions $f(\mathbf{x}, \cdot)$ under P belonging to the class $\mathcal{P}$; compare with the Jensen inequality.

If the set of feasible solutions of (12) is a singleton, the worst-case distribution P^* does not depend on f and we obtain bounds for the optimal values of the stochastic programs (12) under an arbitrary probability distribution P from the class (10) and an arbitrary function f which is convex in ω:

$$\begin{aligned} &\min_{\mathbf{x}\in X} f(\mathbf{x}, \mathsf{E}_P\omega) \leq \min_{\mathbf{x}\in X} \mathsf{E}_P f(\mathbf{x}, \omega) \\ &\leq \min_{\mathbf{x}\in X} \mathsf{E}_{P^*} f(\mathbf{x}, \omega), \quad \forall P \in \mathcal{P}, \end{aligned} \tag{13}$$

provided that the minima exist. Such bounds are numerically tractable, are tight and provide an information about sensitivity of the optimal value of stochastic program (1) on the choice of a probability distribution P belonging to the considered class $\mathcal{P}$. The well-known instance is the class of probability distributions carried by a closed interval $[a, b]$ on the real line with a prescribed value $y \in (a, b)$ of the expectation $\mathsf{E}_P\omega$. The worst-case distribution is carried by the endpoints of the given interval $[a, b]$ and the only solution of the system

$$p_1 a + p_2 b = y, \quad p_1 + p_2 = 1, \quad p_1, p_2 \geq 0$$

is $p_1 = (b - y)/(b - a)$ and $p_2 = 1 - p_1$. The result agrees with the well-known Edmundson-Madansky inequality and the minimax approach guarantees that this bound is tight within the considered class of probability distributions and for convex functions $f(\mathbf{x}, \cdot)$.

There is a host of papers devoted to designing various bounds for the objective function $F(\mathbf{x}, P)$ of stochastic programs (1) under various assumptions about the class $\mathcal{P}$ and the function $f(\mathbf{x}, \cdot)$; for a review of the related results see [3,4,6,13,17] and the references therein. These bounds proved to be useful also in designing algorithms and this is at present the main field of successful applications of the minimax approach.

On the other hand, to get minimax decisions is rather demanding, as it requires the solution of the full minimax problem (4). Except for the simple special cases, such as a unique feasible discrete distribution that fulfills (7) or the optimal value of the objective function (8) obtained in an explicit form, one has to rely on special numerical procedures such as the stochastic quasigradient methods designed for this purpose in [9,10]. The numerical difficulties are behind the fact that, in spite of a sound motivation, real life applications of the minimax approach have been rare and have consisted of the simple special cases e. g., [1,2,6,8,15,16].

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs

- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Minimax: Directional Differentiability
- ▶ Minimax Theorems
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Anandalingam G (1987) A stochastic programming process model for investment planning. Comput Oper Res 14:521–536
2. Bühler W (1981) Capital budgeting under qualitative data information. In: Crum RL, Derkinderen FGJ (eds) Capital Budgeting under Conditions of Uncertainty. Nijhoff, pp 81–117
3. Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer, Berlin
4. Birge JR, Wets RJ-B (1986) Designing approximation schemes for stochastic optimization problems, in particular for stochastic programs with recourse. Math Program Stud 27:54–102
5. Dupačová J (1976) Minimax stochastic programs with nonconvex nonseparable penalty functions. In: Prékopa A (ed) Progress in Operations Research. J. Bolyai Math. Soc. and Univ. Press, pp 303–316
6. Dupačová J (1977) Minimax approach to stochastic linear programming and the moment problem. EMO 13:279–307, extended abstract: ZAMM 58, T466–T467. (In Czech)
7. Dupačová J (1980) Minimax stochastic programs with nonseparable penalties. In: Iracki R, Malanowski R, Waluiewicz S (eds) Optimization Techniques. Lecture Notes Control and Information. Springer, Berlin, pp 157–163
8. Dupačová J (1987) The minimax approach to stochastic programming and an illustrative application. Stochastics 20:73–88
9. Ermoliev Y, Gaivoronski A, Nedeva C (1985) Stochastic optimization problems with partially known distribution functions. SIAM J Control Optim 23:696–716
10. Gaivoronski A (1991) A numerical method for solving stochastic programming problems with moment constraints on a distribution function. Ann Oper Res 31:347–369
11. Iosifescu M, Theodorescu R (1963) Sur la programmation linéaire. CR Acad Sci Paris 256:4831–4833
12. Jagannathan R (1977) Minimax procedure for a class of linear programs under uncertainty. Oper Res 25:173–177
13. Kall P (1988) Stochastic programming with recourse: Upper bounds and moment problems - a review. In: Guddat J et al (eds) Advances in Mathematical Optimization. Akademie-Verlag, Berlin, pp 86–103
14. Kemperman JMB (1968) The general moment problem, a geometric approach. Ann Math Statist 39:93–122
15. Klein Haneveld W (1986) Robustness against dependence in PERT: An application of duality and distributions with known marginals. Math Program Stud 27:153–182

16. Nedeva C (1988) Some applications of stochastic optimization methods to the electric power system. In: Ermoliev Yu, Wets RJ-B (eds) Numerical Techniques for Stochastic Optimization Problems. Springer, Berlin, pp 455–464
17. Prékopa A (1995) Stochastic programming. Kluwer, Dordrecht
18. Žáčková J (1966) On minimax solutions of stochastic linear programming problems. Časopis Mat 91:423–430

Stochastic Programming Models: Random Objective

ANDRÁS PRÉKOPA
RUTCOR, Rutgers Center for Operations Research, Piscataway, USA

MSC2000: 90C15

Article Outline

Keywords
See also
References

Keywords

Random objective function; Efficient point; Kataoka principle

We consider an objective function $h(\mathbf{x}, \boldsymbol{\xi})$, to be maximized in a mathematical programming problem, where $\boldsymbol{\xi}$ is a random vector. There are two prominent proposals to incorporate it into a stochastic programming model formulation. The first one is due to H.M. Markowitz [2,3] and it advocates for the maximization, with respect to $\mathbf{x}$ and subject to the given constraints, of the function

$$\alpha \, \mathsf{E}\left[h(\mathbf{x}, \boldsymbol{\xi})\right] - \beta \sqrt{\mathrm{Var}\left[h(\mathbf{x}, \boldsymbol{\xi})\right]},$$

where $\alpha > 0$ and $\beta > 0$ are constants. We may choose $\alpha = 1$. Let $\mu(\mathbf{x}) = \mathsf{E}[h(\mathbf{x}, \boldsymbol{\xi})]$, $\sigma(\mathbf{x}) = \sqrt{\mathrm{Var}\left[h(\mathbf{x}, \boldsymbol{\xi})\right]}$. An optimal solution $\mathbf{x}_0$ can be characterized by the following two statements:

a) there is no feasible $\mathbf{x}$ such that $\mu(\mathbf{x}) = \mu(\mathbf{x}_0)$, $\sigma(\mathbf{x}) < \sigma(\mathbf{x}_0)$;
b) there is no feasible $\mathbf{x}$ such that $\sigma(\mathbf{x}) = \sigma(\mathbf{x}_0)$, $\mu(\mathbf{x}) > \mu(\mathbf{x}_0)$.

We say that the pair $(\mu(\mathbf{x}_0), \sigma(\mathbf{x}_0))$ is an *efficient point* among all pairs $(\mu(\mathbf{x}), \sigma(\mathbf{x}))$.

An important special case is the random objective function $h(\mathbf{x}, \boldsymbol{\xi}) = \boldsymbol{\xi}^\top \mathbf{x}$ and it comes up in portfolio composition problems, where the components of $\boldsymbol{\xi}$ are the random returns of the assets. If we introduce the notations $\boldsymbol{\mu} = \mathsf{E}(\boldsymbol{\xi})$, $C = \mathsf{E}[(\boldsymbol{\xi} - \boldsymbol{\mu})(\boldsymbol{\xi} - \boldsymbol{\mu})^\top]$, then the objective function of the stochastic programming problem takes the form:

$$\boldsymbol{\mu}^\top \mathbf{x} - \beta \sqrt{\mathbf{x}^\top C \mathbf{x}}.$$

Sometimes $\mathbf{x}^\top C \mathbf{x}$ is replaced for $\sqrt{\mathbf{x}^\top C \mathbf{x}}$, in order to obtain a convex quadratic programming problem.

Sometimes $\boldsymbol{\mu}^\top \mathbf{x}$ is fixed (or a lower bound is prescribed for it) and $\mathbf{x}^\top C \mathbf{x}$ is minimized, or $\mathbf{x}^\top C \mathbf{x}$ is fixed (or an upper bound is prescribed for it) and $\boldsymbol{\mu}^\top \mathbf{x}$ is maximized.

The second principle to handle $h(\mathbf{x}, \boldsymbol{\xi})$ is due to S. Kataoka [1]. In this case we formulate the problem

$$\begin{cases} \max & d \\ \text{s.t.} & \mathsf{P}\left(h(\mathbf{x}, \boldsymbol{\xi}) \geq d\right) \geq p, \\ & \mathbf{x} \in D, \end{cases}$$

where p is a prescribed probability and D is the set of feasible solutions in the original problem. In the special case $h(\mathbf{x}, \boldsymbol{\xi}) = \boldsymbol{\xi}^\top \mathbf{x}$, and under the assumption that $\boldsymbol{\xi}$ has a normal distribution, the above problem can be shown to be equivalent to

$$\begin{cases} \max & \left\{\boldsymbol{\mu}^\top \mathbf{x} + \Phi^{-1}(1-p)\sqrt{\mathbf{x}^\top C \mathbf{x}}\right\} \\ \text{s.t.} & \mathbf{x} \in D, \end{cases}$$

where Φ is the univariate standard normal probability distribution function.

See also

- Approximation of Extremum Problems with Probability Functionals
- Approximation of Multivariate Probability Integrals
- Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- Extremum Problems with Probability Functions: Kernel Type Solution Methods

- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Kataoka S (1963) A stochastic programming model. Econometrica 31:181–196
2. Markowitz H (1952) Portfolio selection. J Finance 7:77–91
3. Markowitz H (1996) Portfolio selection: Efficient diversification of investment. Wiley, New York

Stochastic Programming: Nonanticipativity and Lagrange Multipliers

IGOR EVSTIGNEEV[1], SJUR DIDRIK FLÅM[2]
[1] Acad. Sci. Russia, Moscow, Russia
[2] University Bergen, Bergen, Norway

MSC2000: 90C15

Article Outline

Keywords
The Basic Model
Results
Lagrange Multipliers for Phase Constraints
Lagrange Multipliers for Nonanticipativity Constraints
Synthesis
See also
References

Keywords

Stochastic programming; Lagrange multipliers; Nonanticipativity; Fritz John conditions; Yosida-Hewitt decomposition

Decision making under uncertainty can often be formalized as a *stochastic program*, constrained not merely in material terms (by bounded resources, capacities, technological possibilities etc.), but also by limited information (*nonanticipativity*). The former type of constraints, accounting for material bounds, is usually described by inequalities required to hold almost surely. The latter type, reflecting informational restrictions, often assumes the form of linear equations involving conditional expectation operators. Each sort of constraint generates its own Lagrange multipliers. These Lagrange multipliers have various applications. In particular, they play key roles in algorithms for solving stochastic programs employing decomposition (cf. also ▶ Stochastic linear programming: Decomposition and cutting planes) or constraint relaxation techniques [12,13,17,19]. Besides their computational role, these auxiliary variables also figure prominently in optimality conditions, duality theory and stability analysis (cf. also ▶ Stochastic integer programming: Continuity, stability, rates of convergence). Present randomness, they

take on specific features [1,3,4,9,11,15,16,18], which are reviewed in this article.

The Basic Model

Stochastic programs often assume the following form. Minimize a functional $f(x) = f(x(\cdot))$ over some given set in a linear space $\mathcal{L}$ containing finite sequences $x(\cdot) = (x_1(\cdot), \ldots, x_T(\cdot))$ of random vectors $x_t(\omega) \in \mathbf{R}^{n_t}$. These vectors represent constrained choices (made sequentially, one at each stage or time $t = 1, \ldots, T < \infty$) under imperfect knowledge about the state $\omega \in \Omega$ of the world. Although ω is not known a priori, its probability distribution, P, is given exogenously and defined on some *sigma-field* $\mathcal{F}_{T+1}$ in Ω. The knowledge of ω increases over time. Specifically, there is an expanding family $\mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_{T+1}$ of sigma-fields which describes the information flow. At time t one may ascertain for any event in $\mathcal{F}_t$ (and such events only) whether it has happened or not. In particular, a finite $\mathcal{F}_t$ partitions Ω into minimal events (atoms, information sets, decision nodes) on each of which x_t must be constant. The inclusion $\mathcal{F}_t \subseteq \mathcal{F}_{t+1}$, $t \leq T$, reflecting progressive acquisition of knowledge, says that the partition becomes finer as time evolves.

At time t the decision-maker implements the part x_t of his overall decision $x = (x_1, \ldots, x_T)$. That part is supposed to be an $\mathcal{F}_t$-measurable strategy (policy, behavioral rule) $x_t \colon \Omega \to \mathbf{R}^{n_t}$. This means that only available information is used any stage; decisions are based on realized rather than future events. If so, the process $x = (x_1, \ldots, x_T)$ is called *nonanticipative with respect to the filtration* $\mathcal{F} = (\mathcal{F}_t)_{t=1}^T$, and we write $x \in \mathcal{F}$ for brevity. For example, let $\theta_1, \ldots, \theta_T$ be a stochastic process, defined on Ω, and let $\mathcal{F}_t$ be generated by $\theta_1, \ldots, \theta_t$. Then $x \in \mathcal{F}$ means that x_t depends on $\theta_1, \ldots, \theta_t$ only.

Besides the informational limitation $x \in \mathcal{F}$, there are other restrictions $x \in \mathcal{G} \cap \mathcal{X}$ 'material' in nature, which are defined as follows: $x \in \mathcal{L}$ belongs to the set $\mathcal{G}$ (and is said to satisfy the *phase constraints*) if and only if

$$g_t(\omega, x(\omega)) := g_t(\omega, x_1(\omega), \ldots, x_t(\omega)) \in -K_t(\omega) \quad (1)$$

almost surely (a.s.); $x \in \mathcal{L}$ belongs to $\mathcal{X}$ if and only if

$$x_t(\omega) \in X_t(\omega) \quad \text{a.s.} \quad (2)$$

for all t. Here, $g_t \colon \Omega \times \mathbf{R}^{n_1 + \cdots + n_t} \to \mathbf{R}^{m_t}$ is $\mathcal{F}_t \times \mathcal{B}$-measurable, and $K_t(\omega) \subseteq \mathbf{R}^{m_t}$, $X_t(\omega) \subseteq \mathbf{R}^{n_t}$ are $\mathcal{F}_t$-measurable random sets (see, e. g., [2] and [1]); $\mathcal{B}$ stands for the Borel σ-algebra. Both $X_t(\omega)$ and $K_t(\omega)$ are nonempty and closed; $K_t(\omega)$ is a convex cone. Define the relation $\preceq_{t,\omega}$ on $\mathbf{R}^{m_t}$ by $a \preceq_{t,\omega} b \Leftrightarrow b - a \in K_t(\omega)$. Then (1) can be written in the form

$$g_t(\omega, x(\omega)) \preceq_{t,\omega} 0 \quad \text{a.s.}. \quad (3)$$

The basic optimization problem is stated as follows.

$$(P) \quad \begin{cases} \text{Find} & \inf f(x) \\ \text{s.t.} & x \in \mathcal{F} \cap \mathcal{G} \cap \mathcal{X}. \end{cases}$$

Problem (P) is supposed to be feasible (i. e., $\mathcal{F} \cap \mathcal{G} \cap \mathcal{X} \neq \emptyset$) with finite optimal value.

Important examples of objective functionals include integral functionals of the form

$$f(x) := \mathsf{E}F(\omega, x(\omega)) = \int F(\omega, x(\omega))\mathsf{P}(d\omega), \quad (4)$$

where F is some $\mathcal{F}_{T+1} \times \mathcal{B}$-measurable integrand for which $f(x)$, $x \in \mathcal{X}$, is well-defined and finite.

In the general setting, the objective f is a real-valued functional on the set $\mathcal{X} \subseteq \mathcal{L}$, where $\mathcal{L}$ is the given linear space of $\mathcal{F}_{T+1}$-measurable vector functions $x(\omega) = (x_1(\omega), \ldots, x_T(\omega))$. As $\mathcal{L}$, one often takes $L^p(\mathcal{F}_{T+1}, \mathsf{P}; \mathbf{R}^n)$, $n := \sum n_t$, with $p \in [1, +\infty]$. This space consists of (equivalence classes of) $\mathcal{F}_{T+1}$-measurable functions $x \colon \Omega \to \mathbf{R}^n$ with finite norm $\| x \|_p := [\int |x(\omega)|^p \mathsf{P}(d\omega)]^{1/p}$ if $p \in [1, +\infty)$, and $\| x \|_\infty := \operatorname{ess\,sup} |x(\omega)|$, where $|\cdot|$ is any fixed norm on a finite-dimensional vector space.

It will be convenient to assume that all the σ-algebras $\mathcal{F}_t$, are completed by all subsets of null-sets in Ω. In many applications, this assumption does not lead to a significant loss in generality.

Results

For the most part, it will not be supposed that problem (P) has optimal solutions. For completeness, however, an existence result is provided.

Theorem 1 (Existence of optimal solutions.) *Suppose that, for each ω, the set*

$$A(\omega) := \left\{ a = (a_1, \ldots, a_T) : \begin{array}{c} a_t \in X_t(\omega), \\ g_t(\omega, a) \in -K_t(\omega), \\ \forall t \end{array} \right\}$$

is closed, convex and bounded in the norm $|\cdot|$ by some number $\beta(\omega) \geq 0$, where $\mathsf{E}\beta < \infty$. Assume $\mathcal{L}$ contains a $x \in \mathcal{F}$ such that $x(\omega) \in A(\omega)$ a.s. Also, suppose f is convex and lower semicontinuous with respect to L^1 convergence. Then problem (P) admits an optimal solution.

Proof The feasible set $\mathcal{F} \cap \mathcal{G} \cap \mathcal{X}$ is convex, closed in L^1 and uniformly integrable, hence weakly sequentially compact, while f is weakly lower semicontinuous [6].

Some notation must now be fixed. If $\mathcal{Y}$ is a topological vector space, we write $y^* \in \mathcal{Y}^*$ when y^* is a continuous linear mapping of $\mathcal{Y}$ into $\mathbf{R}$. The value of $y^* \in \mathcal{Y}^*$ at $y \in \mathcal{Y}$ is denoted by y^*y. If $\mathcal{K} \subseteq \mathcal{Y}$ is a convex cone, its *dual cone* is defined as $\mathcal{K}^+ := \{y^* \in \mathcal{Y}^* : y^*y \geq 0, \forall y \in \mathcal{K}\}$.

The following general fact will serve as the basis for further presentation.

Proposition 2 (A *Fritz John rule*) *Consider an optimization problem:*

$$(P1)\quad \begin{cases} \text{Find} & \inf f(x) \\ \text{s.t.} & x \in \mathcal{X}, \\ & g(x) \in -\mathcal{K}, \\ & h(x) = 0, \end{cases}$$

having finite optimal value inf(P1) for the given functions $f: \mathcal{X} \to \mathbf{R}$, $g: \mathcal{X} \to \mathcal{Y}$ and $h: \mathcal{X} \to \mathcal{Z}$. Here $\mathcal{X}$ is a set, $\mathcal{K} \subseteq \mathcal{Y}$ is a convex cone, and $\mathcal{Y}$, $\mathcal{Z}$ are Hausdorff linear topological spaces. Suppose the convex hull conv C of the set

$$C := \left\{ \begin{array}{c} (r, y, z) \\ \in \mathbf{R} \times \mathcal{Y} \times \mathcal{Z} \end{array} : \begin{array}{c} f(x) \leq \inf(P1) + r, \\ g(x) \in -\mathcal{K} + y, \\ h(x) = z \\ \text{for some } x \in \mathcal{X} \end{array} \right\}$$

has nonempty interior and (0, 0, 0) at its boundary. Then there exists a nonzero continuous linear functional $(r^*, y^*, z^*) \in \mathbf{R}_+ \times \mathcal{K}^+ \times \mathcal{Z}^*$ such that

$$\begin{aligned} & r^* \inf(P1) \\ & = \inf\{r^* f(x) + y^* g(x) + z^* h(x) : \ x \in \mathcal{X}\}. \end{aligned} \tag{5}$$

If $\mathcal{Y}$, $\mathcal{Z}$ are both finite-dimensional, it suffices for (5) that (0, 0, 0) lies at the boundary of conv C.

Proof The convex hull of C has a closed supporting hyperplane through its boundary point (0, 0, 0). Hence there is a nonzero $(r^*, y^*, z^*) \in \mathbf{R}^* \times \mathcal{Y}^* \times \mathcal{Z}^*$ such that $r^*r + y^*y + z^*z \geq 0$ for all $(r, y, z) \in C$. It is straightforward to see that $r^* \geq 0$, and that y^* must belong to the dual cone $\mathcal{K}^+$. Thus $r^*[f(x) - \inf(\text{P1})] + y^*g(x) + z^*h(x) \geq 0$ for all $x \in \mathcal{X}$, implying $\inf\{r^*f(x) + y^*g(x) + z^*h(x): x \in \mathcal{X}\} \geq r^* \inf(\text{P1})$. The reverse inequality holds trivially.

The above result can be employed, in particular, if

a) C is convex, and

b) the interior int$C$ of the set C is not empty.

Observe that (0, 0, 0) is always on the boundary of C. Condition a) is fulfilled if, for any $x^i \in \mathcal{X}$, $y^i \in g(x^i) + \mathcal{K}$, $z^i = h(x^i)$, $i = 1, 2$, and $\rho \in [0, 1]$, there exists $x \in \mathcal{X}$ such that $f(x) \leq \rho f(x^1) + (1 - \rho) f(x^2)$, $g(x) \in -\mathcal{K} + \rho y^1 + (1 - \rho) y^2$, and $h(x) = \rho z^1 + (1 - \rho) z^2$. In turn, this property holds if $\mathcal{X}$ is a convex set in some linear space, f is a convex functional on $\mathcal{X}$, $h: \mathcal{X} \to \mathcal{Z}$ is affine, and the mapping $g: \mathcal{X} \to \mathcal{Y}$ is convex with respect to the cone $\mathcal{K}$, i. e., $\rho g(x^1) + (1 - \rho) g(x^2) - g(\rho x^1 + (1 - \rho) x^2) \in \mathcal{K}$ for all $x^1, x^2 \in \mathcal{X}$ and $\rho \in [0, 1]$.

In this article, Proposition 2 is applied to versions of problem (P1) in which one of the constraints $g(x) \in -\mathcal{K}$ or $h(x) = 0$ is not present. Observe that int$C \neq \emptyset$ in any of the following cases:

i) h is absent, and f is bounded above on a set $\mathcal{X}_1 \subseteq \mathcal{X}$ with int$[g(\mathcal{X}_1) + \mathcal{K}] \neq \emptyset$ (this is so if int$\mathcal{K} \neq \emptyset$);

ii) g is absent and f is bounded above on some $\mathcal{X}_2 \subseteq \mathcal{X}$ for which int $h(\mathcal{X}_2) \neq \emptyset$.

In the applications below, the functional f and the set $\mathcal{X}$ will be those involved in the basic model (see the previous section). Furthermore, fix some $q \in [1, +\infty]$ and set

$$\begin{aligned} & \mathcal{Y} := \prod_{t=1}^{T} L^q(\mathcal{F}_t, \mathsf{P}; \mathbf{R}^{m_t}), \\ & g(x) := \big(g_t(\cdot, x)\big), \\ & \mathcal{K} := \{y \in \mathcal{Y}: \ y_t(\omega) \in K_t(\omega) \text{ a.s. } \forall t\}. \end{aligned} \tag{6}$$

Suppose $\| g(x) \|_q < \infty$ for all $x \in \mathcal{X}$. Also, following [16], define

$$h(x) := x - (\mathsf{E}_1 x_1, \ldots, \mathsf{E}_T x_T), \tag{7}$$

and $\mathcal{Z} := h(\mathcal{L})$, assuming that h (when this operator comes into action) is well-defined on $\mathcal{L} \supseteq \mathcal{X}$. Here E_t stands for the conditional expectation given $\mathcal{F}_t$. Observe that $x \in \mathcal{F}$ if and only if $h(x) = 0$.

In the subsequent analysis, the following hypothesis regarding problem (P) will be used:

FJ) There exists a nonzero $(r^*, y^*, z^*) \in \mathbf{R}_+ \times \mathcal{K}^+ \times \mathcal{Z}^*$ such that the Fritz John condition (5) holds with inf(P) finite.

It is understood that if some constraint $g(x) \in -\mathcal{K}$ or $h(x) = 0$ is absent in (P), or automatically satisfied, then the corresponding part of (y^*, z^*) should be omitted.

Conditions which guarantee applicability of Proposition 2, and hence the truth of FJ), are presented above. Hypothesis FJ) is especially well motivated for convex problems. Absent convexity, FJ) frequently obtains when $\mathcal{X}$ is a neighborhood of some local optimum.

Lagrange Multipliers for Phase Constraints

Throughout this subsection, it is assumed that $\mathcal{L} \subseteq \mathcal{F}$. The information constraint is thus satisfied automatically, and problem (P) reduces to finding $\inf f$ over $\mathcal{G} \cap \mathcal{X}$. Associate the *Lagrangian*

$$L(x, \lambda) := f(x) + \lambda g(x), \quad x \in \mathcal{X}, \quad \lambda \in \mathcal{K}^+,$$

to constraint (3).

Theorem 3 (Lagrange multipliers for the phase constraints.) *Assume FJ) and the following strict feasibility condition: For any $y = (y_t)_{t=1}^T \in \mathcal{Y}$, belonging to some neighborhood of 0, one can find $x \in \mathcal{X}$ satisfying*

$$g_t(\omega, x(\omega)) \leq_{t,\omega} y_t(\omega) \text{ a.s.}, \quad \forall t.$$

Then there exists $\lambda = (\lambda_t)_{t=1}^T \in \mathcal{K}^+$ such that

$$\inf(P) = \inf_{\mathcal{X}} L(\cdot, \lambda). \tag{8}$$

Proof The strict feasibility condition ensures that the number r^* involved in FJ) is strictly positive. Divide (5) by r^* and set $\lambda := y^*/r^*$.

It is often important to obtain an integral representation of λ,

$$\lambda y = \mathsf{E}\lambda(\omega) y(\omega), \quad y \in \mathcal{Y}, \tag{9}$$

with an appropriate function $\lambda(\omega)$. This is immediate if $q \in [1, +\infty)$ because then $L^q(\mathcal{F}_t, \mathsf{P}; \mathbf{R}^{m_t})^* = L^{q^*}(\mathcal{F}_t, \mathsf{P}; \mathbf{R}^{m_t})$, where $q^* := q/(q-1)$, $1^* = +\infty$. However, not every functional in the dual of L^∞ is of the form (9) (those which admit representation (9) with $\lambda(\cdot) \in L^1$ are called *absolutely continuous*). Therefore the case $q = +\infty$ requires special consideration. The analysis of that case is based on the following continuity property of f:

- For any pair $x, \widetilde{x} \in \mathcal{X}$ and any sequence of $\mathcal{F}_T$-measurable indicators $\chi^k\ \Omega \to \{0, 1\}$ satisfying $\mathsf{E}\chi^k \to 0$, we have

$$f(\chi^k \widetilde{x} + (1 - \chi^k)x) \to f(x), \tag{10}$$

provided $\chi^k \widetilde{x} + (1 - \chi^k)x \in \mathcal{X}$.

Clearly (10) holds when f is of the form (4) (with finite values).

One speaks of *complete recourse* if:

CR) for any $x \in \mathcal{X}$ and $1 \leq t \leq T$, there exists $\widetilde{x}_t \in \mathcal{F}_t \cap \mathcal{X}_t$ such that

$$g_t(\omega, x_1(\omega), \ldots, x_{t-1}(\omega), \widetilde{x}_t(\omega)) \leq_{t,\omega} 0 \text{ a.s.} \tag{11}$$

and $(x_1, \ldots, (1 - \mathbf{1}_S)x_t + \mathbf{1}_S \widetilde{x}_t, \ldots, x_T) \in \mathcal{X}$ for each $S \in \mathcal{F}_t$.

Here $\mathbf{1}_S(\omega) = 1$ if $\omega \in S$ and 0 otherwise; the notation $\widetilde{x}_t \in \mathcal{F}_t \cap \mathcal{X}_t$ means that $\widetilde{x}_t$ is $\mathcal{F}_t$-measurable and satisfies (2).

Theorem 4 (Absolutely continuous Lagrange multipliers for essentially bounded phase constraints.) *Consider the problem*

$$\begin{cases} \min & f \\ s.t. & x \in \mathcal{G} \cap \mathcal{X}. \end{cases}$$

Assume FJ), CR), strict feasibility, and the continuity property (10). Then, there exists $\lambda = (\lambda_t)_{t=1}^T \in \mathcal{K}^+$ with $\lambda_t \in L^1(\mathcal{F}_t, \mathsf{P}; \mathbf{R}^{m_t})$ satisfying (8).

Proof By virtue of the *Yosida-Hewitt theorem* [20], any $\lambda \in L^{\infty *}$ admits a unique decomposition $\lambda = \lambda^a + \lambda^s$ into an absolutely continuous component $\lambda^a \in L^1$ and a *singular component* λ^s. The last notion means that there exist measurable sets $S^1, S^2, \ldots$ such that $\mathsf{P}(S^k) \to 0$ and $\lambda^s(\cdot) = \lambda^s(\chi^k \cdot)$, $\forall k$, where $\chi^k := \mathbf{1}_{S^k}$.

Consider $\lambda = (\lambda_t) \in \mathcal{K}^+$ satisfying (8) and decompose λ_t into $\lambda_t^a + \lambda_t^s$. We claim that, in (8), one may replace λ by $\lambda^a = (\lambda_t^a)$, i. e., set all $\lambda_t^s = 0$. Indeed, by way of induction, suppose $\lambda_\tau^s = 0$ for all $\tau > t$:

$$f(x) + \sum_{\tau \leq t} \lambda_\tau g_\tau(x) + \sum_{\tau \geq t+1} \lambda_\tau^a g_\tau(x) \geq \inf(\mathrm{P}) \tag{12}$$

for any $x \in \mathcal{X}$, where $1 < t < T$. (If $t = T$, the last sum is zero.) Then the analogous inequality also holds for $t-1$. To prove this, select a sequence $S^1, S^2, \ldots$ of $\mathcal{F}_t$-measurable sets such that $\mathsf{P}(S^k) \to 0$ and $\lambda_t^s(\cdot) = \lambda_t^s(\chi^k \cdot)$, $\forall k$, where χ^k is the indicator of S^k. Fix any $x \in \mathcal{X}$. Consider the function $\widetilde{x}_t \in \mathcal{F}_t \cap \mathcal{X}_t$ described in CR). Let $x^k \in \mathcal{X}$ be obtained from x by substituting $\chi^k \widetilde{x}_t + (1-\chi^k)x_t$ in place of x_t (in coordinate t only). Then

$$
\begin{aligned}
&f(x^k) + \sum_{\tau \le t-1} \lambda_\tau g_\tau(x^k) + \sum_{\tau \ge t} \lambda_\tau^{\mathrm{a}} g_\tau(x^k) \\
&\ge f(x^k) + \sum_{\tau \le t-1} \lambda_\tau g_\tau(x^k) + \lambda_t^s g_t(x^k) \\
&\quad + \sum_{\tau \ge t} \lambda_\tau^{\mathrm{a}} g_\tau(x^k) \ge \inf(\mathrm{P}),
\end{aligned}
$$

because $\lambda_t^s g_t(x^k) = \lambda_t^s(\chi^k g_t(x^k)) \le 0$. To obtain (12) for $t-1$, it suffices to pass to the limit, employing the continuity property (10).

Finally, observe that $\lambda^{\mathrm{a}} = (\lambda_t^a) \in \mathcal{K}^+$, since, for any $y_t \ge_{t,\omega} 0$, we have $0 \le \lambda_t[(1-\chi^k)y_t] = \lambda_t^a[(1-\chi^k)y_t] \to \lambda_t^a y_t \ge 0$.

Methods using similar arguments were first proposed by A.Ya. Dubovitskii and A.A. Milyutin [5] in (deterministic) optimal control theory. Applications to stochastic extremal problems were developed in [8,9,10,14,15,16], and [1], where various versions and extensions of Theorem 4 can be found. Another approach, relying on a direct analysis of perturbation functions in L_1 and yielding Lagrange multipliers representable as functions in L_∞, was suggested by E.B. Dynkin [7].

Lagrange Multipliers for Nonanticipativity Constraints

In this subsection, let $\mathcal{L} := L^p(\mathcal{F}_{T+1}, \mathsf{P}; \mathbf{R}^n)$ for some $p \in [1, +\infty]$ and $\mathcal{Z} := \{z \in \mathcal{L}: (\mathsf{E}_1 z_1, \ldots, \mathsf{E}_T z_T) = 0\} = h(\mathcal{L})$, where h is given by (7).

The next goal is to relax the constraint $x \in \mathcal{F}$ ($\Leftrightarrow h(x) = 0$). Constraints of this type were first examined systematically by R.T. Rockafellar and R.J-B. Wets [16]. To separate different issues, suppose here that all the phase constraints of type (3) are absent, or already relaxed, as described above. Then (P) reduces to minimizing f over $\mathcal{F} \cap \mathcal{X}$. To deal with the constraint $h(x) = 0$, consider a different Lagrangian

$$
\Lambda(x, \pi) := f(x) + \pi h(x), \qquad x \in \mathcal{X}, \quad \pi \in \mathcal{Z}^*. \tag{13}
$$

Theorem 5 (Lagrange multipliers for the nonanticipativity constraints.) *Consider the problem of minimizing f over $\mathcal{F} \cap \mathcal{X}$. Assume FJ). If $\mathcal{X}$ has nonempty interior, then there is a linear functional $\pi \in \mathcal{Z}^*$ such that*

$$
\inf(P) = \inf_{\mathcal{X}} \Lambda(\cdot, \pi). \tag{14}
$$

Proof Evidently the linear mapping $h: \mathcal{L} \to \mathcal{Z}$ defined in (7) is surjective. Both spaces $\mathcal{L}$, $\mathcal{Z}$ are Banach. Since the set $\mathcal{U} := \mathrm{int}\mathcal{X} \ne \emptyset$ is open, the open mapping theorem implies that $h(\mathcal{U})$ is open. Furthermore, $0 \in h(\mathcal{U})$ (see Remark 6), and so $0 \in \mathrm{int}\, h(\mathcal{X})$. As a result, r^* must be strictly positive. Divide 5 by r^* and set $\pi := z^*/r^*$.

Remark 6 Note that if $\mathrm{int}\,\mathcal{X} \ne \emptyset$ and $p < \infty$, then $X_t(\omega) = \mathbf{R}^{n_t}$ a.s., and so $\mathcal{X} = \mathcal{L}$. Furthermore, if $\mathrm{int}\,\mathcal{X} \ne \emptyset$, then, for any p, $\mathcal{F} \cap \mathrm{int}\,\mathcal{X} \ne \emptyset$, which implies $0 \in h(\mathrm{int}\,\mathcal{X})$.

Remark 7 By the *Hahn-Banach theorem*, the functional constructed in Theorem 5 can be extended to a continuous functional $\pi \in \mathcal{L}^*$.

Again, it is of importance to obtain an integral representation of π. Like before, when $p \in [1, +\infty)$, this representation is immediate, since $\mathcal{L}^* = L^{p^*}$ with $p^* = p/(p-1)$, $1^* = +\infty$. Suppose $p = +\infty$.

Theorem 8 (Absolutely continuous multipliers for the nonanticipativity constraints: the L^∞ case.) *Consider the problem*

$$
\begin{cases} \min & f \\ \textit{over} & \mathcal{F} \cap \mathcal{X}. \end{cases}
$$

Assume FJ) together with the continuity condition (10). Suppose that $\mathcal{X}$ is convex and $\mathrm{int}\,\mathcal{X} \ne \emptyset$. Then there exists an absolutely continuous functional on $\mathcal{L} = L^\infty(\mathcal{F}_{T+1}, \mathsf{P}; \boldsymbol{R}^n)$ satisfying (14).

Proof Let π be the functional described in Theorem 5 and Remark 7. By the Yosida-Hewitt theorem, each coordinate π_t of π decomposes into the sum $\pi_t^a + \pi_t^s$, where $\pi_t^a \in L^1$ and π_t^s is singular. Let $S^1, S^2, \ldots \in \mathcal{F}_{T+1}$, $\mathsf{P}(S^k) \to 0$ and $\pi_t^s(\cdot) = \pi_t^s(\chi^k \cdot)$, $\forall k$, where $\chi^k = \mathbf{1}_{S^k}$. We

may assume, additionally, that $\| (1-\chi^k)\mathsf{E}_t\chi^k \|_\infty \to 0$ [9]. Consider any $x \in \mathcal{X}$. Construct x^k from x by substituting $\chi^k \mathsf{E}_t x_t + (1-\chi^k)x_t$ in place of x_t in coordinate t only. Then $x^k \in \mathcal{X}$ by virtue of the convexity and $\mathcal{F}_t$-measurability of $X_t(\omega)$ (see [1, App. II]; $X_t(\omega)$ is convex a.s. since $\mathcal{X}$ is convex). Finally, $f(x^k) \to f(x)$ and $\pi_t(x_t^k - \mathsf{E}_t x_t^k) \to \pi_t^a(x_t - \mathsf{E}_t x_t)$, because $\pi_t^s(x_t^k - \mathsf{E}_t x_t^k) = \pi_t^s[\mathsf{E}_t x_t(\chi^k - \mathsf{E}_t\chi^k)]$, where $\mathsf{E}_t|\chi^k - \mathsf{E}_t\chi^k| \le 2\mathsf{E}_t(1-\chi^k)\mathsf{E}_t\chi^k \to 0$ in $\| \cdot \|_\infty$. This shows that $\Lambda(x, \pi^a) \ge \inf$ (P) for all $x \in \mathcal{X}$.

Remark 9 If $\pi = (\pi_t)$ admits an integral representation, then the Lagrangian (13) can be written

$$\begin{aligned}\Lambda(x,\pi) &= f(x) + \mathsf{E}\sum_{t=1}^{T} \mathsf{E}_t\left[\pi_t(x_t - \mathsf{E}_t x_t)\right] \\ &= f(x) + \mathsf{E}\sum_{t=1}^{T}\left[(\pi_t - \mathsf{E}_t\pi_t)x_t\right] .\end{aligned}$$

This allows one to interpret $\pi_t - \mathsf{E}_t\pi_t$ as a 'shadow price' of information [3,4,9,16].

Synthesis

Combination of the results presented above allows one to examine both the phase and the nonanticipativity constraints simultaneously. The next theorem provides a criterion of optimality in terms of pointwise minimization of a Lagrangian associated with the two constraints. Consider problem (P) with $\mathcal{L} = L^p(\mathcal{F}_{T+1}, \mathsf{P}; \mathbf{R}^n)$ and f defined by (4). Suppose that the following hypotheses hold:

C) For each ω, the set $X(\omega) := X_1(\omega) \times \cdots \times X_T(\omega)$ and the function $F(\omega, a)$, $a \in X(\omega)$, are convex; the mapping $g(\omega, a)$, $a \in X(\omega)$, is convex with respect to the cone $K_1(\omega) \times \cdots \times K_T(\omega)$.

G) The functional $f(x)$ is bounded above on some set $\mathcal{X}_1 \subseteq \mathcal{X} \cap \mathcal{F}$ with $\operatorname{int}[g(\mathcal{X}_1) + \mathcal{K}] \ne \emptyset$; furthermore, $0 \in \operatorname{int}[g(\mathcal{X} \cap \mathcal{F}) + \mathcal{K}]$.

H) For any integral linear functional $\lambda \in \mathcal{K}^+$, $f(x) + \lambda g(x)$ is bounded above on some $\mathcal{X}_2 \subseteq \mathcal{X}$ with int $h(\mathcal{X}_2) \ne \emptyset$, and we have int $\mathcal{X} \ne \emptyset$.

The sets of interior points involved in G), int $h(\mathcal{X}_2)$, and int $\mathcal{X}$ are defined in terms of the spaces $\mathcal{Y}$, $\mathcal{Z} = h(\mathcal{L})$, and $\mathcal{L}$, respectively (for the definition of $\mathcal{Y}$ and $\mathcal{K}$ see (6)). Additionally, if $q = \infty$, assume CR).

Theorem 10 (Pointwise optimality.) *Let $x \in \mathcal{X} \cap \mathcal{F} \cap \mathcal{G}$. Then x is a solution to (P) if and only if there exist functionals $\lambda \in \mathcal{K}^+$ and $\pi \in L^p{}^*$ of integral form (9) such that a.s.*

$$x(\omega) \in \arg\min \left\{ \begin{array}{c} F(\omega,a) + \lambda(\omega)g(\omega,a) \\ + \sum_t \left[\pi_t(\omega) - E_t\pi_t(\omega)\right] a_t \end{array} \;:\; a_t \in X_t(\omega) \right\} \quad (15)$$

and $\lambda(\omega)g(\omega, x(\omega)) = 0$.

Proof Let x be a solution to (P). It is sufficient to construct functionals $\lambda \in \mathcal{K}^+$ and $\pi \in L^p{}^*$ of integral form such that

$$\mathsf{E}(\omega, x) \le \mathsf{E}\left\{ \begin{array}{c} F(\omega,x') + \lambda(\omega)g(\omega,x') \\ + \sum_t [\pi_t - \mathsf{E}_t\pi_t]\, x_t' \end{array} \right\}$$

for all $x' \in \mathcal{X}$. Then a suitable measurable selection argument (see, e. g., [1, App. I]) yields (15)). To construct λ, use Theorems 3 and 4. Then, to prove the existence of π, apply Theorems 5 and 8 to a modified optimization problem with the objective functional $f(x) + \lambda\, g(x)$. (The truth of FJ) follows from i) and ii).) The 'if' statement is straightforward.

Stochastic programming, as presented above, can easily accommodate integral constraints of the form $\int \varphi(\omega, x(\omega))\mathsf{P}(d\omega) \in M$, where $\varphi: \Omega \times \mathbf{R}^n \to \mathbf{R}^d$ is an integrand satisfying appropriate conditions and M is a cone in $\mathbf{R}^d$.

Of course this article is only a brief glance at the large and rapidly developing field of study. Many relevant aspects have not been discussed. Perhaps the most important of such aspects is the tight connection between the theory of stochastic Lagrange multipliers and the theory of stochastic economic models. The former provides technical tools for the latter. The latter serves as a source of problems and often as a 'proving ground' for new developments regarding stochastic Lagrange multipliers. For an introduction into this subject, see [1].

See also

► Approximation of Extremum Problems with Probability Functionals

- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Arkin VI, Evstigneev IV (1987) Stochastic models of control and economic dynamics. Acad. Press, New York
2. Aubin JP, Frankowska H (1990) Set-valued analysis. Birkhäuser, Basel
3. Back K, Pliska SR (1987) The shadow price of information constraints in continuous time decision problems. Stochastics 22:151–186
4. Dempster MAH (1988) On stochastic programming: II Dynamic problems under risk. Stochastics 25:15–42
5. Dubovitskii AYa, Milyutin AA (1968) Necessary conditions for a weak extremum in optimal control problems with mixed inequality constraints. USSR Comput Math Math Phys 8:378–395
6. Dunford N, Schwartz JT (1957) Linear operators. I, Interscience, New York
7. Dynkin EB (1972) Stochastic concave dynamic programming. Math USSR Sb 16:501–515
8. Evstigneev IV (1972) Optimal economic planning taking account of stationary random factors. Math USSR Dokl 13:1357–1359
9. Evstigneev IV (1976) Lagrange multipliers for the problems of stochastic programming. In: Los MW, Los J, Wieczorek A (eds) Warsaw Fall Seminars in Mathematical Economics. Lecture Notes Economics and Math Systems, vol 133. Springer, Berlin, pp 34–48
10. Flåm SD (1985) Non-anticipativity in stochastic programming. J Optim Th Appl 46:23–30
11. Flåm SD (1992) Lagrange multipliers in stochastic programming. SIAM J Control Optim 30:1–10
12. Higle J, Sen S (1996) Stochastic decomposition. Kluwer, Dordrecht
13. Kall P, Wallace SW (1994) Stochastic programming. Wiley, New York
14. Radner R (1973) Optimal stationary consumption with stochastic production and resources. J Econom Theory 6:68–90
15. Rockafellar RT, Wets RJ-B (1975) Stochastic concave programming: Kuhn–Tucker conditions. J Math Economics 2:349–370
16. Rockafellar RT, Wets RJ-B (1976) Nonanticipativity and L1-martingales in stochastic optimization problems. In: Wets RJ-B (ed) Stochastic Systems: Modeling, Identification and Optimization II. Math Program Stud. North-Holland, Amsterdam, pp 170–187
17. Rockafellar RT, Wets RJ-B (1991) Scenarios and policy aggregation in optimization under uncertainty. Math Oper Res 16:1–23
18. Rö W, Schultz R (1991) Stability analysis for stochastic programs. Ann Oper Res 30:241–266
19. Ruszczyński A (1997) Decomposition methods in stochastic programming. Math Program 70:333–353
20. Yosida K, Hewitt E (1952) Finitely additive measures. Trans Amer Math Soc 72:46–66

Stochastic Programming: Parallel Factorization of Structured Matrices

HERCULES VLADIMIROU, STAVROS A. ZENIOS
University Cyprus, Nicosia, Cyprus

MSC2000: 90C15

Article Outline

Keywords
Problem Formulation
Interior Point Algorithm
Parallel Matrix Factorization
Computational Experience
References

Keywords

Stochastic programming; Interior point methods; Structured matrix factorization; Parallel computing

Uncertainty is pervasive in many decision making problems on which it often plays a key role. Uncertainty in the input data of mathematical programs is generally modeled by postulating a probability distribution (usually discrete) for the unknown parameters which is then incorporated in an appropriate optimization model. Based on this approach, stochastic programs (SP) provide a constructive and prescriptive framework for incorporating, ex-ante, uncertainty in decision making models. *Stochastic programming* has evolved into an effective framework for modeling sequential decision problems under uncertainty in diverse applications: e. g., investment management, production and logistics, capacity and operational planning for electric power generation, management of natural resources, network design, etc.

Attention is focused here on two-stage stochastic linear programs with recourse which address the following situation: Certain decisions must be made at present in the face of uncertainty. At a later time uncertainty is resolved by observing a joint realization (outcome) of the values of all uncertain parameters. At that time, further corrective (recourse) actions can be taken in response to the outcome that materializes. Each postulated realization of the uncertain parameters constitutes a particular *scenario*. The objective is to minimize the expected value of a total cost functional, which includes the direct cost of the initial decisions and the expected cost of the recourse actions.

Problem Formulation

Two-stage SP with recourse distinguish between two sets of decision variables:

- $x_0 \in \mathbf{R}^{n_0}$ denotes the *first-stage decisions*. These decisions are made before the values of the random variables are observed, but they should anticipate the consequent cost of recourse actions.
- $y_s \in \mathbf{R}^{n_1}$ denotes the *second-stage decisions* under a particular scenario s. These are the adaptive decisions, representing *recourse* actions that are taken after the random variables have been observed. They depend on the first-stage decisions and on the realization of the random variables.

Uncertainty is represented by a discrete set of scenarios $S = \{1, \ldots, S\}$ with associated probabilities $p_s > 0$, $\sum_{s=1}^{S} p_s = 1$. The two stage SP with recourse can then be stated in the following deterministic equivalent program, [13]:

$$\min_{x_0 \in \mathbf{R}_+^{n_0},\, y_s \in \mathbf{R}_+^{n_1}} c^\top x_0 + \sum_{s=1}^{S} p_s q_s^\top y_s, \tag{1}$$

such that

$$A_0 x_0 = b, \tag{2}$$

$$T_s x_0 + W_s y_s = h_s, \quad \forall s \in S. \tag{3}$$

Any deterministic constraints on the first-stage decisions are depicted by equation (2) the coefficients of which (i. e., the $m_0 \times n_0$ matrix A_0 and the vector $b \in \mathbf{R}^{m_0}$) are scenario invariant. Each scenario $s \in S$ is associated with a corresponding instance of the input data, that is, the $m_1 \times n_0$ technology matrix T_s, the $m_1 \times n_1$ recourse matrix W_s, and the vectors $q_s \in \mathbf{R}^{n_1}$ and $h_s \in \mathbf{R}^{m_1}$.

In this compact representation of the deterministic equivalent program, the constraints matrix has a *dual block-angular structure*:

$$\begin{pmatrix} A_0 & & & \\ T_1 & W_1 & & \\ \vdots & & \ddots & \\ T_S & & & W_S \end{pmatrix}. \tag{4}$$

The above formulation has $n = n_0 + S \cdot n_1$ variables and $m = m_0 + S \cdot m_1$ constraints. Hence, the inclusion of a large number of scenarios, so as to account for many possible contingencies, inevitably leads to very large optimization programs. Substantial research effort has been directed toward the development of effective solution methods that exploit the special block-structures of SP and capitalize on the capabilities of high-performance computing systems, including parallel multiprocessors.

Reformulations are sometimes applied to yield structures that are more suitable for some parallel algorithms. These reformulations employ variable-splitting to replicate the first-stage solutions into distinct vectors $x_s \in \mathbf{R}^{n_1}$ for each scenario $s \in S$. Explicit *nonanticipativity constraints* are then added to the program in order to ensure that the values of these distinct vectors are scenario-invariant. Nonanticipativity constraints can be of a staircase form:

$$x_s - x_{s-1} = 0, \quad s = 2, \ldots, S. \tag{5}$$

Similarly, the distinct vectors x_s may be equated with an auxiliary vector x_0, yielding a primal block-angular structure:

$$x_s - x_0 = 0, \quad s = 1, \ldots, S. \tag{6}$$

Recent reviews of alternative parallel algorithms for solving stochastic programs can be found in [2,12].

Interior Point Algorithm

Interior point algorithms directly address the deterministic equivalent program (1)–(3). Let us focus on the primal-dual, path following interior point method (e. g., [11]) which solves simultaneously the following pair of dual programs:

$$(P) \quad \min_{x \geq 0} c^\top x \quad \text{s.t. } Ax = b,$$

$$(D) \quad \max_{z \geq 0} b^\top y \quad \text{s.t. } A^\top y + z = c.$$

The $m \times n$ constraint matrix A is assumed to have full row rank. The method applies a logarithmic barrier to enforce the nonnegativity constraints. Each iteration involves a Newton step for the system of linear equations that represents first-order conditions for a critical point of the associated Lagrangian functions for the barrier forms of (P) and (D). The algorithm is given below.

X and Z are the $n \times n$ positive definite diagonal matrices $X = \mathrm{diag}(x_1, \ldots, x_n)$, $Z = \mathrm{diag}(z_1, \ldots, z_n)$. The steplengths α_P, α_D are computed so as to keep the primal variables (x) and the dual slack iterates (z) positive:

$$\alpha_P \doteq \delta \cdot \min_{\alpha_j > 0} \{\alpha_j : \; x_j + \alpha_j \Delta x_j \geq 0\}, \tag{7}$$

$$\alpha_D \doteq \delta \cdot \min_{\alpha_j > 0} \{\alpha_j : \; z_j + \alpha_j \Delta z_j \geq 0\}, \tag{8}$$

where $\delta \in (0, 1)$. A typical value of δ, used in practice is 0.9995. An updating formula for setting the barrier parameter μ^ν that works well in practice is:

$$\mu^\nu = \frac{c^\top x^\nu - b^\top y^\nu}{n^2}.$$

Initialization
Set $\nu = 0$. Start with an interior point ($x^\nu \in \mathbf{R}^n_+$, $z^\nu \in \mathbf{R}^n_+$, $y^\nu \in \mathbf{R}^m$), and $\mu^\nu > 0$.
Iterative step
Solve for the dual step Δy:

$$(A\Theta A^\top)\Delta y = \psi, \tag{9}$$

where $\Theta = XZ^{-1}$, $\psi = \rho + A\Theta(\sigma - X^{-1}\phi)$, $\rho = b - Ax$,
$\sigma = c - A^\top y - z$, $\phi = \mu \mathbf{1} - XZ\mathbf{1}$, and $\mathbf{1}$ is a conformable vector of ones.
Compute the primal step Δx, and the slack variable step Δz, from

$$\Delta x = -\Theta(\sigma - X^{-1}\varnothing - A^\top \Delta y), \tag{10}$$

$$\Delta z = -X^{-1}(\varnothing - Z\Delta x). \tag{11}$$

Update:

$$x^{\nu+1} = x^\nu + \alpha_P \Delta x, \tag{12}$$

$$y^{\nu+1} = y^\nu + \alpha_D \Delta y, \tag{13}$$

$$z^{\nu+1} = z^\nu + \alpha_D \Delta z, \tag{14}$$

where α_P, $\alpha_D \in (0, 1)$ are steplengths. reduce μ^ν to $\mu^{\nu+1}$, and increment the iteration counter $\nu \leftarrow \nu + 1$.

Primal-dual path following algorithm

Hence, the barrier parameter is kept large when far from the optimum (as measured by a large duality gap in the numerator) and the search direction points away from the boundary of the feasible region, so as to allow large steps. The barrier parameter is reduced as the optimum is approached so that the iterates may approach the boundary of the feasible region. Practically the same computations are applied – with fairly minor extensions – to solve separable, convex quadratic programs (see, e. g., [5,11]).

The major effort involves the solution of the $n \times n$ symmetric, positive definite system of linear equations $(A\Theta A^\top)\Delta y = \psi$. This system is commonly encountered in interior point methods. So, the parallel matrix factorization procedures discussed here are directly applicable in other interior point algorithms as well. Interior point methods for stochastic programs are reviewed in [10].

The constraints matrix A of a two-stage SP has the dual block-angular structure (4). In the discussion above, the vector x encompasses all primal decision variables, that is, the first-stage decisions x_0, as well as the recourse decisions y_s for all scenarios. Also, the scenario probabilities p_s are incorporated by scaling the objective coefficients. Despite the sparsity of the constraints matrix A, the product matrix $A\Theta A^\top$ can be very dense due to the presence of the coupling columns associated with the first-stage variables (see (4)). Hence, the direct application of interior point methods to stochastic programs is not particularly effective.

A first approach to overcoming this problem focuses on staircase formulations (cf. (5)). This significantly reduces fill-in and produces banded product matrices which can be factorized efficiently [8]. Schur complements have also been tested as a means for overcoming the problem with the dense columns of the first-stage variables [3,8]. These procedures can improve substantially the performance of interior point algorithms on SP. However, they can not be effectively parallelized. Moreover, the Schur complement approach suffers from numerical instabilities, particularly in problems with many dense columns [3].

An alternative, is to directly parallelize the matrix operations in interior point methods. Such procedures typically treat the optimization programs as fully dense and can not exploit the sparse block structure of SP. Consequently, they are effective only for moderate-size problems [6,9].

A third approach is to specialize a *matrix factorization* procedure so as to capitalize on the structure of SP. The method is based on a generalization of the *Sherman–Morrison–Woodbury formula*. It was proposed for stochastic programs by J.R. Birge and L. Qi [4], and was further extended by Birge and D.F. Holmes [3], who also reported numerical experiments. Implementations of this factorization procedure on hypercubes and other parallel computers are reported in [7,14].

Parallel Matrix Factorization

Partition the vectors Δy and ψ in (9) into subvectors $[\Delta y_0^\top, \ldots, \Delta y_S^\top]^\top$ and $[\psi_0^\top, \ldots, \psi_S^\top]^\top$, respectively, with Δy_l, $\psi_l \in \mathbf{R}^{m_l}$, for $l = 0, \ldots, S$. Here Δy_0 represents the dual step corresponding to the first-stage constraints, and Δy_l represents the dual step corresponding to the second-stage constraints for the lth scenario. Hence, $m_l = m_1$, for $l = 1, \ldots, S$; also denote $n_l = n_1$ for $l = 1, \ldots, S$. The matrix factorization procedure that solves for the dual step Δy in (9) is based on the following lemma; for a proof of the lemma, see [4].

Lemma 1 *Let $M \doteq A\,\Theta\,A^\top$, where Θ is diagonal, and $R \doteq diag_{l=0,\ldots,S}(R_l)$, where $R_0 = I$ is an $\mathbf{R}^{m_0} \times \mathbf{R}^{m_0}$ identity matrix, $R_l = W_l\Theta_l W_l^\top \in \mathbf{R}^{m_l \times m_l}$, $l = 1, \ldots, S$, and $\Theta_l \in \mathbf{R}^{n_l \times n_l}$ is the (diagonal) submatrix of Θ corresponding to the lth block. Also, let*

$$
\begin{aligned}
G_1 &\doteq \Theta_0^{-2} + A_0^\top A_0 + \sum_{l=1}^{S} T_l^\top R_l^{-1} T_l, \\
G &\doteq \begin{pmatrix} G_1 & A_0^\top \\ -A_0 & 0 \end{pmatrix} \\
U &\doteq \begin{pmatrix} A_0 & I \\ T_1 & 0 \\ \vdots & \vdots \\ T_S & 0 \end{pmatrix}, \quad V \doteq \begin{pmatrix} A_0 & -I \\ T_1 & 0 \\ \vdots & \vdots \\ T_S & 0 \end{pmatrix}.
\end{aligned}
\tag{9}
$$

If A_0 and W_l, $l = 1, \ldots, S$, have full row rank then M and $G_2 \doteq -A_0\,G_1^{-1}A_0^\top$ are invertible, and

$$
M^{-1} = R^{-1} - R^{-1}UG^{-1}V^\top R^{-1}. \tag{10}
$$

Equation (10) indicates that the solution of the linear system $M\Delta y = (A\Theta A^\top)\Delta y = \psi$ can be expressed as Δy

$= p - r$, where p solves $Rp = \psi$, and r is obtained from the system

$$Gq = V^\top p, \quad Rr = Uq. \tag{11}$$

The vector p can be computed componentwise by solving $R_l p_l = \psi_l$, for $l = 0, \ldots, S$. The block structure of G is exploited in solving for q. One can write:

$$Gq = \begin{pmatrix} G_1 & A_0^\top \\ -A_0 & 0 \end{pmatrix} \begin{pmatrix} q^1 \\ q^2 \end{pmatrix} = \begin{pmatrix} \widehat{p}^1 \\ \widehat{p}^2 \end{pmatrix},$$

where

$$\begin{pmatrix} \widehat{p}^1 \\ \widehat{p}^2 \end{pmatrix} \doteq V^\top p = \begin{pmatrix} A_0^\top p_0 + \sum_{l=1}^{S} T_l^\top p_l \\ -p_0 \end{pmatrix}. \tag{12}$$

Hence,

$$q^2 = -G_2^{-1}(\widehat{p}^2 + A_0 G_1^{-1} \widehat{p}^1), \tag{13}$$

$$q^1 = G_1^{-1}(\widehat{p}^1 - A_0^\top q^2). \tag{14}$$

Once q is known, r can be computed componentwise from (11).

The required operations rely extensively on matrix subblock computations that can be performed independently of one another. A parallel procedure for computing the dual step Δy is summarized in the following box (denote by $A_{i\cdot}$ the ith row of A, and by $A_{\cdot j}$ the jth column of A).

Interprocessor data communication is necessary at only three points. After forming the terms $T_l^\top R_l^{-1} T_l$ in Steps 2a–2b the processors communicate to form the matrix G_1 and the vectors $\widehat{p}^1$ and $\widehat{p}^2$, in Step 2b. The results can be accumulated at a single master processor which executes serially the computations involving the dense matrices G_1 and G_2 in Steps 2c–2e. The computed vector q is then broadcasted to all other processors. Steps 3 and 4 require only the distributed data R_l, T_l, and p_l on the lth processor and can be carried out with full parallelism. A final communication step accumulates the partial vectors Δy_l at the master processor to form Δy. This vector can then be made available to all processors for use in the subsequent calculation of the directions Δx and Δz; these computations involve only matrix-vector products and vector additions that can be parallelized in a rather straightforward manner [14]. This algorithm is suitable for implementation on distributed memory, as well as on shared memory multiprocessors.

Begin with the following data distribution. Processor l holds W_l, T_l, Θ_l, and ψ_l, $l = 1, \ldots, S$. A designated master processor also holds A_0, R_0, Θ_0, and ψ_0.

1 (Parallel solution of $Rp = \psi$.)
The master processor sets $R_0 p_0 = I p_0 = \psi_0$.
Processors $l = 1, \ldots, S$, form $R_l = W_l \Theta_l W_l^\top$ and solve $R_l p_l = \psi_l$ for p_l.

2 (Solution of $G_q = V^\top p$.)

2a Processors $l = 1, \ldots, S$, solve $R_l (u_l)^i = (T_l)_{\cdot i}$ for $(u_l)^i$, $i = 1, \ldots, n_0$, thus computing the columns of the matrix $u_l = [(u_l)^1, \ldots, (u_l)^{n_0}] = R_l^{-1} T_l$.

2b Processors $l = 1, \ldots, S$, multiply $v_l = T_l^\top u_l$ to form $v_l = T_l^\top R_l^{-1} T_l$ and also compute $\varsigma_l = T_l^\top p_l$. Communicate $v_l \in \mathbf{R}^{n_0 \times n_0}$ and $\varsigma_l \in \mathbf{R}^{n_0}$ to form G_1 (cf. (15)) and $\hat{p}^1$ (cf. (18)) on the master processor.
The master processor sets $\hat{p}^2 = -p_0$.

2c The master processor solves $G_1 u = \hat{p}^1$ for u and sets $v = \hat{p}^2 + A_0 u$ (cf. (19)).

2d The master processor forms G_2 by solving $(G_1) w^i = (A_0^\top)_{\cdot i}$ for w^i, $i = 1, \ldots, m_0$, and sets $G_2 = -A_0 [w^1, \ldots, w^{m_0}]$.

2e The master processor solves $G_2 q^2 = -v$ for q^2 (cf. (19)), and solves $G_1 q^1 = \hat{p}^1 - A_0^\top q^2$ for q^1 (cf. (20)).
Communicate to distribute $q^1 \in \mathbf{R}^{n_0}$ to all processors.

3 (Parallel solution of $Rr = U_q$.)
The master processor sets $r_0 = A_0 q^1 + q^2$.
Processors $l = 1, \ldots, S$, solve $R_l r_l = T_l q^1$ for r_l.

4 (Form Δy in parallel.)
The master processor sets $\Delta y_0 = p_0 - r_0$. Processors $l = 1, \ldots, S$, set $\Delta y_l = p_l - r_l$.
Communicate to gather the vector Δy on the master processor.

Parallel matrix factorization for dual step calculation

An alternative parallel implementation is to distribute the matrices G_1 and G_2 to all processors, and let the processors proceed locally (and redundantly) with all calculations involving these matrices. The master processor approach uses an all-to-one communication

step to accumulate the dense matrices at the master, followed by a one-to-all communication step to distribute the results to the processors. The alternative approach combines these two communication steps into a single all-to-all communication step that distributes the dense matrices to all processors which compute locally (and redundantly) their own copies of the vector q. Both of these alternatives are very efficient on present day distributed memory machines with high-bandwidth interconnections.

Yet another alternative is to distribute the dense matrices across processors, and use parallel dense linear algebra techniques for all calculations that involve these matrices. This approach is more suitable for shared-memory, tightly coupled multiprocessors, and when the dense matrices G_1 and G_2 are large.

A.J. Berger et al. [1] proposed another matrix factorization procedure that exploits the block-structure form of stochastic programs in interior point algorithms. The method, termed *tree dissection*, operates on the split-variable formulation and is applicable to multistage stochastic programs with recourse and convex, block-separable objective functions. A serial implementation of the method demonstrated very competitive computational performance on large scale problems in comparison to direct applications of interior point algorithms.

Computational Experience

Interior point algorithms can be applied to solve SP with linear or separable, convex objective functions. Separability is important to maintain sparsity in the projection matrices. Nonseparable problems can yield full projection matrices, thus dramatically increasing the computational complexity. In such cases, it is possible to treat a problem as fully dense and directly parallelize the matrix operations involved in interior point methods without regard to problem structure [6,9]. However, such an approach is effective only for moderate size problems due to its substantial computational and storage requirements.

Interior point algorithms have proved very robust on two-stage SP with linear, or separable, convex quadratic objectives. The required number of iterations is neither particularly influenced in going from linear to separable quadratic SP, nor is it significantly affected by the size of the problem or by the conditioning of the objective function. Hence, the algorithms can solve to a high accuracy very large SP in a moderate number of iterations. However, they may exhibit numerical difficulties if the constraint matrix does not have full row rank. Even if the entire constraint matrix has full row rank, the parallel factorization procedure may suffer from numerical instabilities if the recourse matrices W_s do not have full row rank as well. Thus, care must be exercised in implementations to test and account for situations in which the recourse matrices are rank deficient.

The parallel matrix factorization procedure presented in this article has been subjected to extensive numerical experimentation on hypercubes and other parallel computing systems [7,14], exhibiting a high level of scalability on large scale problems. In the implementations, each parallel task factorized the part of the projection matrix corresponding to a scenario. Moreover, the matrix operations involved in the factorization procedure, and throughout the interior point algorithm, can be executed efficiently on vector processors, or be further parallelized on massively parallel multiprocessors. Particularly, the operations on the small dense matrices that constitute the coordination step of the algorithm can be vectorized or parallelized to effectively eliminate any serial bottleneck.

References

1. Berger AJ, Mulvey JM, Rothberg E, Vanderbei RJ (1995) Solving multistage stochastic programs using tree dissection. Report Dept Civil Engin and Oper Res, Princeton Univ SOR-95-05
2. Birge JR (1997) Stochastic programming computation and applications. INFORMS J Comput 9:111–133
3. Birge JR, Holmes DF (1992) Efficient solution of two-stage stochastic linear programs using interior point methods. Comput Optim Appl 1:245–276
4. Birge JR, Qi L (1988) Computing block-angular Karmarkar projections with applications to stochastic programming. Managem Sci 34:1472–1479
5. Censor Y, Zenios SA (1997) Parallel optimization: Theory, algorithms, and applications. Oxford Univ. Press, Oxford
6. Eckstein J, Qi R, Ragulin VI, Zenios SA (1992) Data-parallel implementations of dense linear programming algorithms. Report Decision Sci Dept Wharton School, Univ Pennsylvania 92-05-06
7. Jessup ER, Yang D, Zenios SA (1994) Parallel factorization of structured matrices arising in stochastic programming. SIAM J Optim 4:833–846

8. Lustig I, Mulvey JM, Carpenter TJ (1991) Formulating stochastic programs for interior point methods. Oper Res 39:757–770
9. Qi R-J, Zenios SA (1994) On the scalability of data-parallel decomposition algorithms for stochastic programs. J Parallel Distributed Comput 22:565–570
10. Ruszczyński A (1993) Interior point methods in stochastic programming. Report Internat Inst Appl Systems Anal, Laxenburg, Austria WP-93-8
11. Vanderbei RJ, Carpenter TJ (1993) Symmetric indefinite systems for interior point methods. Math Program 58:1–32
12. Vladimirou H, Zenios SA (1999) Scalable parallel computations for large-scale stochastic programming. Ann Oper Res 90:87–129
13. Wets RJ-B (1974) Stochastic programs with fixed recourse: The equivalent deterministic problem. SIAM Rev 16:309–339
14. Yang D, Zenios SA (1997) A scalable parallel interior point algorithm for stochastic linear programming and robust optimization. Comput Optim Appl 7:143–158

Stochastic Programming with Simple Integer Recourse

MAARTEN H. VAN DER VLERK
University of Groningen, Groningen, The Netherlands

MSC2000: 90C15, 90C11

Article Outline

Keywords
References

Keywords

Stochastic programming; Simple recourse; Integer recourse

The *simple integer recourse* model with fixed technology matrix is defined as

$$\inf_x \{cx + \bar{Q}(x) : Ax = b\ ,\ x \in R^{n_1}_+\}\ ,$$

where the *expected value function* $\bar{Q}$ is

$$\bar{Q}(x) = E_\xi\ v(\xi - Tx)\ ,$$

and v is the value function of the second-stage problem

$$v(s) = \inf_{y^+, y^-} \{q^+y^+ + q^-y^- : y^+ \geq s\ , y^- \geq -s, y^+, y^- \in Z^{m_2}_+\}$$

for $s \in R^{m_2}$. Here c, A, b, q^+, q^- and T are vectors/matrices of the appropriate size, $q^+, q^- \geq 0$, $q^+ + q^- > 0$, and ξ is a random vector in R^{m_2}.

As suggested by the name, this model has the same structure as the well-known continuous **simple recourse** model, in which the second-stage decision variables $y = (y^+, y^-)$ are non-negative reals. These models are indeed the most simple **recourse models**, both analytically and conceptually. In both models the recourse actions (that is, the compensations for observed deviations from the constraints $Tx = \xi$) are straightforward. For example, let Tx represent production to meet uncertain demand ξ. Then in the continuous model the recourse actions may represent buying or selling any shortage or surplus, whereas in the integer recourse model buying and selling is only possible in batches of a certain size. In both models the objective function reflects the direct costs cx and the expected recourse costs $\bar{Q}(x)$.

Using separability which is due to the simple recourse structure, $\bar{Q}$ is completely characterized by the one-dimensional generic function Q, given by

$$Q(z) = q^+E_\xi\lceil\xi - z\rceil^+ + q^-E_\xi\lfloor\xi - z\rfloor^-\ ,\ z \in R\ ,$$

with $q^+, q^- \in R_+$, $q^+ + q^- > 0$, ξ a random variable, and $\lceil s\rceil^+ = \max\{0, \lceil s\rceil\}$, $\lfloor s\rfloor^- = \max\{0, -\lfloor s\rfloor\}$, $s \in R$. Below we present results for the one-dimensional function Q; the extension to the n_1-dimensional case is straightforward.

In [11] structural properties of the function Q are presented. A closed-form expression for Q is given,

$$Q(z) = q^+ \sum_{k=0}^{\infty} \Pr\{\xi > z + k\} + q^- \sum_{k=0}^{\infty} \Pr\{\xi < z - k\}\ ,\quad z \in R\ ,$$

and conditions for (**Lipschitz**) continuity and (one-sided) differentiability are derived. (Corresponding results for the model in which also the technology matrix T is random are given in [4].) The function Q is continuous if and only if ξ follows a continuous distribution. If ξ is a discrete random variable with realizations ξ^j, $j = 1, \ldots, r$, then Q is **lower semicontinuous** with discontinuity points $\cup_j\{\xi^j + Z\}$. Moreover, even if Q is continuous, it is non-**convex** in general. This has led to

the study of conditions on the distribution of ξ such that the function Q is convex, and to the construction of convex approximations of Q.

Since Q is continuous (and finite) precisely if ξ is continuously distributed (with finite mean value), it follows that Q can only be convex if ξ has a probability density function, say f. In [9] it is shown that, under some mild technical conditions, Q is convex if and only if $f(s) = G(s+1) - G(s)$, $s \in R$, where G is an arbitrary cumulative distribution function with finite mean value. For example, if G corresponds to the degenerate distribution in 1 then f is a probability density function of the **uniform distribution** on $[0, 1]$. Formulated in terms of random variables, we have that Q is convex if and only if there exists a random variable η with finite mean value, such that for all $s \in R$ the conditional distribution of ξ given $\eta = s$ is uniform on $[s-1, s]$. From this we see that the uniform distribution with unit support plays a central role here.

In [7] it is shown that any reasonable convex approximation of the function Q can be represented as a one-dimensional expected value function of a continuous simple recourse model, with a random right-hand side parameter whose distribution is known. Consequently, given a convex approximation of Q, the integer recourse model can be solved (at least approximately) by well-known algorithms for continuous simple recourse models (see e. g. [1,3,12,13,19]).

For the case that ξ follows a finite discrete distribution, a **strongly polynomial algorithm** to construct the **convex hull** of the function Q is given in [8]. In [6] it is shown that if the matrix T has full row rank, then the resulting one-dimensional functions can be used as building blocks for the convex hull of the n_1-dimensional expected value function $\bar{Q}$. If this condition is not satisfied, a convex lower bound for $\bar{Q}$ is obtained.

If ξ is a continuous random variable, convex approximations of Q can be obtained by perturbing its distribution. In [9] a class of such approximations, defined by their probability density functions $f_\alpha(s) = F(\lfloor s \rfloor_\alpha + 1) - F(\lfloor s \rfloor_\alpha)$, $s \in R$, is analyzed. Here F is the cumulative distribution function of ξ, $\alpha \in [0, 1)$ is a shift parameter, and $\lfloor \cdot \rfloor_\alpha$ denotes round down with respect to the set $\{\alpha + Z\}$ (the case $\alpha = 0$ corresponds to the usual integer round down). For each $\alpha \in [0, 1)$, the function $Q_\alpha(z) = q^+ E\lceil \xi_\alpha - z \rceil^+ + q^- E\lfloor \xi_\alpha - z \rfloor^-$, $z \in R$, with the random variable ξ_α distributed according to f_α, is a piecewise linear convex approximation of Q. It is shown that

$$\|Q_\alpha - Q\|_\infty \le (q^+ + q^-)\frac{|\Delta| f}{4},$$

where $|\Delta| f$ is the **total variation** of f. By taking convex combinations this uniform error bound can be improved by a factor two at most, which is obtained by using $f_{\alpha\beta} = (f_\alpha + f_\beta)/2$ with $|\alpha - \beta| = 1/2$ as the approximating distribution. For many distributions the total variation of f decreases as the variance of the distribution increases. In these cases the approximation becomes better accordingly.

The continuous simple recourse representations of the approximations presented above have discretely distributed right-hand side parameters, and can therefore be solved efficiently. Algorithms to compute these distributions (and standard solution methods) are implemented in the model management system SLP-IOR [2].

Most results referred to above can be found in [17]. For an overview of the field of **stochastic (mixed-)integer programming** beyond the simple recourse case, we refer to [5,10,14,15,16]. An extensive *bibliography of* ***stochastic programming*** in general can be found in [18], which also contains a separate listing of stochastic integer programming literature.

References

1. Birge J, Louveaux F (1997) Introduction to Stochastic Programming. Springer, New York
2. Kall P, Mayer J (1996) SLP-IOR: An interactive model management system for stochastic linear programs. Math Program 75(2):221–240
3. Kall P, Wallace S (1994) Stochastic Programming. Wiley, Chichester
4. Klein Haneveld W, van der Vlerk M (1994) On the expected value function of a simple integer recourse problem with random technology matrix. J Comput Appl Math 56:45–53
5. Klein Haneveld W, van der Vlerk M (1999) Stochastic integer programming: General models and algorithms. Ann Oper Res 85:39–57
6. Klein Haneveld W, Stougie L, van der Vlerk M (1994) Some theorems on the convex hull of a function with an application to stochastic integer programming. Discussion Paper TI 94-81, Tinbergen Institute, Amsterdam-Rotterdam
7. Klein Haneveld W, Stougie L, van der Vlerk M (1995) On the convex hull of the simple integer recourse objective function. Ann Oper Res 56:209–224

8. Klein Haneveld W, Stougie L, van der Vlerk M (1996) An algorithm for the construction of convex hulls in simple integer recourse programming. Ann Oper Res 64:67–81
9. Klein Haneveld W, Stougie L, van der Vlerk M (2006) Simple integer recourse models: convexity and convex approximations. Math Program 108(2-3):435–473
10. Louveaux F, Schultz R (2003) Stochastic integer programming. In: Ruszczynski A, Shapiro A (eds) Stochastic Programming. Handbooks in Operations Research and Management Science, vol 10. Elsevier, pp 213–266
11. Louveaux F, van der Vlerk M (1993) Stochastic programming with simple integer recourse. Math Program 61:301–325
12. Prékopa A (1995) Stochastic Programming. Kluwer, Dordrecht
13. Ruszczynski A, Shapiro A (eds) (2003) Stochastic Programming, Handbooks in Operations Research and Management Science, vol 10. Elsevier, Amsterdam
14. Schultz R, Stougie L, van der Vlerk M (1996) Two-stage stochastic integer programming: a survey. Statistica Neerl 50(3):404–416
15. Stougie L, van der Vlerk M (1997) Stochastic integer programming. In: Dell'Amico M, Maffioli F, Martello S (eds) Annotated Bibliographies in Combinatorial Optimization, Wiley, Chichester, pp 127–141
16. Stougie L, van der Vlerk M (to appear) Approximation in stochastic integer programming. In: Johnson D, Lenstra J, Shmoys D (eds) Handbook on Approximation and Heuristics, Elsevier, handbooks in Operations Research and Management Science
17. van der Vlerk M (1995) Stochastic programming with integer recourse. PhD thesis, University of Groningen, The Netherlands
18. van der Vlerk M (1996–2007) Stochastic Programming Bibliography. World Wide Web, http://mally.eco.rug.nl/spbib.html. Accessed 13 April 2008
19. Wets R (1984) Solving stochastic programs with simple recourse. Stochastics 10:219–242

Stochastic Programs with Recourse: Upper Bounds

STEIN W. WALLACE[1], CHANAKA EDIRISINGHE[2]
[1] Department Indust. Econ. and Techn. Management Gløshaugen, Norwegian University Sci. and Techn., Trondheim, Norway
[2] Management Sci. Program, University of Tennessee, Knoxville, USA

MSC2000: 90C15

Article Outline

Keywords
The Edmundson–Madansky Upper Bound
The Piecewise Linear Upper Bound
Restrictions
Upper Bounds for a Convex-Concave Case
See also
References

Keywords

Stochastic programming; Upper bound

A two-stage stochastic linear program with recourse (SLPR) with random right-hand sides and objective costs is normally written as follows:

$$\begin{cases} \min\limits_{x\geq 0} & cx + Q(x) \\ \text{s.t.} & Ax = b \end{cases}$$

where

$$Q(x) = \int_{\Omega} Q(x,\omega)g(\omega)\,d\omega,$$

and

$$Q(x,\omega) = \min_{y\geq 0}\{q(\eta)y:\ Wy = h(\xi) - T(\xi)x\}.$$

Here, $g(\omega)$ is the density function for the joint random vector $\widetilde{\omega} := (\widetilde{\xi},\widetilde{\eta})$ whose support is the set Ω and W is an $(m \times n)$ matrix. This way of formulating a two-stage stochastic program is motivated partly by solution procedures, and partly by the time structure of the problem. For this article, the former is more important. The interpretation of the problem is that first (now) we make a decision x, then we observe a value of the joint vector $\widetilde{\omega}$, and finally we make a *recourse decision* y based on our earlier decision x and the observed value of $\widetilde{\omega}$.

In a direct approach for solution of the above problem, such as *Benders decomposition* [1] (or equivalently the *L-shaped decomposition* [12]), a master problem is created to determine a first stage solution x (say x_0), along with a subproblem to determine the second stage value function $Q(x_0)$ by integrating:

$$\int_{\Omega} Q(x_0,\omega)g(\omega)\,d\omega \equiv \int_{\Omega} f(\omega)g(\omega)\,d\omega.$$

For most functions f and densities g this is impossible to do exactly. Numerical integration when the dimension of the random vector $\widetilde{\omega}$ is beyond 5 or 6 is computationally intractable. Hence, one has to resort to bounds or approximations. Normally, the integral is replaced by a lower bounding expression, either by replacing $f(\omega)$ by some simpler $L(\omega)$, $g(\omega)$ by some simpler $l(\omega)$, or both. Benders decomposition may then be applied to this simplified problem to arrive at an optimal lower bounding solution x_0. To check if this solution is good enough as an optimal solution to the given problem, an upper bound U to $Q(x_0)$ must be found. Such upper bounds are usually determined by either relying on the solution of a certain moment problem that would essentially yield a discretization of the support Ω or using a functional approximation $U(\omega)$ that serves as an upper bounding function to $f(\omega)$. The latter case is discussed in this article. In this case, the resulting approximations may require either univariate integration on marginal domains, or simple discretizations of the support to allow efficient computation.

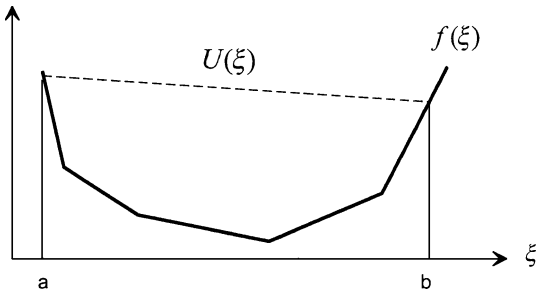
Upper bounds of interest here can be categorized depending on whether SLPR has randomness only in the right-hand sides, or it has randomness in both the objective costs and right-hand sides. In the former case, the function Q is convex in the random vector, while in the latter case, it is convex-concave in the random vectors.

We first consider the convex case, by restricting η to be a degenerate random vector, and thus, using the notation that $\xi \equiv \omega$ and $q \equiv q(\eta)$. An easy upper bound in this case is available due to H.P. Edmundson and A. Madansky.

The Edmundson–Madansky Upper Bound

The *Edmundson–Madansky upper bound* (EM-bound) is based on articles by Edmundson [7] and Madansky [9]. This bound can be interpreted in terms of a moment problem with first moment condition, as well as in terms of an upper bounding function $U(\xi)$ on $f(\xi)$. We consider the latter construction, as illustrated in Fig. 1.

The upper bound $U(\xi)$ can be written as $U(\xi) = r\xi + s$, with $r = (f(b) - f(a))/(b - a)$ and $s = b/(b - a)f(a) - a/(b - a)f(b)$. Upon integration, one obtains the upper bound as



Stochastic Programs with Recourse: Upper Bounds, Figure 1
The basis for the EM-bound

$$\int_a^b U(\xi)g(\xi)\,d\xi = f(a)\frac{b - \mathsf{E}\widetilde{\xi}}{b - a} + f(b)\frac{\mathsf{E}\widetilde{\xi} - a}{b - a}.$$

In other words, by integrating $U(\xi)$ instead of $f(\xi)$ one obtains an upper bound which amounts to just evaluating the function f at the extreme values of the support $[a, b]$, using the weights

$$p = \frac{b - \mathsf{E}\widetilde{\xi}}{b - a} \quad \text{and} \quad 1 - p = \frac{\mathsf{E}\widetilde{\xi} - a}{b - a}.$$

If the random vector ξ has K independent random components, the above reduction will leave us with a total of 2^K points to evaluate. Hence, with more than about 10 random variables or so, this approximation scheme becomes computationally unattractive. Hence, there is a need for an upper bound whose complexity is not exponential in the number of random variables.

The Piecewise Linear Upper Bound

The *piecewise linear upper bound* (PL-bound) is based on the articles independently developed by S.W. Wallace [13] and J.R. Birge and R.J-B. Wets [4], later combined in [3]. Assume now that $f(\xi)$ is given by

$$f(\xi) = \min_y \{qy:\ Wy = b + H\xi,\ 0 \le y \le c\},$$

that is, $T(\xi) \equiv T$, H is a deterministic matrix with n_1 columns, the n_1 random variables are independent, and there is no randomness in the upper limits, c. For simplicity, assume that the support of ξ is $[0, B]$, with $B = (B_1, \ldots, B_{n_1})$. More complex situations can also be

treated. The goal is to find an upper bounding function

$$U(\xi) = f(0) + \sum_{i=1}^{n_1} d_i \xi_i .$$

This function $U(\xi)$ is useful in that for most $g(\xi)$ it is simple to integrate. We first solve

$$f(0) = \min_y \{qy:\ Wy = b,\ 0 \le y \le c\} = qy^0 .$$

This is the base case, and we define $\alpha^1 = -y^0$, and $\beta^1 = c - y^0$. Define a counter r and let $r := 1$. Now, solve (letting H_r be column r of the matrix H)

$$\min_y \{qy:\ Wy = H_r B_r, \alpha^r \le y \le \beta^r\} = qy^r = d^r B_r .$$

Then, update the bounds to obtain

$$\alpha_i^{r+1} = \alpha_i^r - \min\{y_i^r, 0\},$$
$$\beta_i^{r+1} = \beta_i^r - \max\{y_i^r, 0\}.$$

Now, increment r by one and repeat until all n_1 random variables have been treated. The PL-bound, as outlined here, requires the solution of $n_1 + 1$ linear programs, in contrast to the EM-bound which needed 2^{n_1} linear programs in the same setting. Many other versions of this bound exist, see, for example, [8, Sect. 3.4.4; 6.5.1].

Note that the EM-bound and the PL-bound are not comparable, in the sense that either one can be better than the other. The PL-bound bound may be infinite even if the true expected value is finite, whereas the EM-bound is finite if and only if the true value is finite. If the function $f(\xi)$ turns out to be linear in ξ, both bounds are exact.

Restrictions

Upper bounds can be found by adding *restrictions* to the solution set of a problem. The PL-bound above is an example of that. In that case the restriction amounts to reserving certain parts of the upper limits for certain random variables. Another type of thinking about restrictions can be found in [10]. He points out that

$$\int_a^b f(\xi) g(\xi)\, d\xi$$
$$= \int_a^b \min_{y \ge 0} \{qy:\ Wy = b + h(\xi)\}\, g(\xi)\, d\xi$$
$$\le \min_{y \ge 0} \int_a^b \{qy:\ Wy = b + h(\xi)\}\, g(\xi)\, d\xi .$$

The logic of this bound is that if we allow only one y for all values of $\widetilde{\xi}$, rather than a function of ξ, we *restrict* the problem, and hence obtain an upper bound. The usefulness of this bound depends on our ability to evaluate the final expression. In [11] this expression is used in connection with another very useful observation, to arrive a *restricted-recourse bound*. Let

$$z_1^* = \min_{y \ge 0} \{qy:\ Wy \ge \xi\},$$

and define π^* to be an optimal dual solution to this problem. With $\pi' \ge \pi^*$, we get that $z_1^* = z_2^*$ where

$$z_2^* = \min_{y \ge 0} \{qy + \pi'(\xi - Wy)^+\}.$$

Combining the two results we get that the following yields an upper bound:

$$\min_{y \ge 0} \left\{ qy + \pi' \int_a^b (\xi - Wy)^+ g(\xi)\, d\xi \right\}.$$

Solving this problem amounts to solving a stochastic program with simple recourse, and is not particularly hard. The quality of the bound depends to a large extent on the ability to find tight dual solutions. Note that this bound does not require convexity, namely, the matrix W is allowed to have random elements, and that it can be used for much more general situations than here.

As D.P. Morton and R.K. Wood noted, the restricted recourse bounds provide improvements over the penalty-based aggregation bounds developed in [2] and [6]. Restrictions are also used to bound a multistage problem in [14].

Upper Bounds for a Convex-Concave Case

When the right-hand side and objective cost vectors (of the second stage problem) are dependent random

vectors, upper bounds are developed in [5]. The first essential idea is to enclose the joint support Ω within a known $(K + L)$-dimensional simplex $\widehat{\Omega}$, where K and L are, respectively, the number of ξ and η random variables. Such a simplex can generally be determined quite easily, however, the quality of the bound can be affected by a particular choice. Let the chosen simplex $\widehat{\Omega}$ have the extreme points given by $w^i \equiv (u^i, v^i)$ for $i = 1, \ldots, K + L + 1$.

The second important idea is to develop an upper bounding function $U(x, \omega)$ to $Q(x, \omega)$ by using the convexity property of Q in ξ vector, given a fixed first stage decision x. Towards this, let $\widehat{\Omega}_\eta$ be the conditional domain of $\widehat{\Omega}$ for any fixed η value. Under the convexity, the following inequality holds:

$$Q(x, \omega) \le U(x, \omega) := \sum_{i=1}^{K+L+1} p_i(\omega) Q(x, u^i, \eta), \quad \text{for } \omega \in \widehat{\Omega}_\eta,$$

where the nonnegative multipliers $p_i(\omega)$ satisfy the convexity constraints for any $\omega \in \widehat{\Omega}$:

$$\sum_{i=1}^{K+L+1} w^i p_i(\omega) = \omega, \qquad \sum_{i=1}^{K+L+1} p_i(\omega) = 1.$$

Taking expectations under the conditioning argument with respect to the 'true' density $g(\omega)$ yields $Q(x) \le \mathsf{E}U(x, \omega)$. However, $\mathsf{E}U(x, \omega)$ in itself is not easy to evaluate. Hence, a simple inequality is utilized to upper bound the latter expectation. Notice that

$$\mathsf{E}U(x, \omega) = \sum_{i=1}^{K+L+1} \int_{\Omega} G(\omega, x, i) g(\omega)\, d\omega,$$

where

$$G(\omega, x, i) = \min_{y^i \ge 0} \left\{ p_i(\omega) q(\eta) y^i : \; Wy = h(u^i) - T(u^i)x \right\},$$

in which each minimization involves only random objective coefficients. Then, apply Jensen's inequality on the inner minimization to obtain the upper bound as

$$Q(x) \le \sum_{i=1}^{K+L+1} \widehat{p}_i \min_{y^i \ge 0} \left\{ q(\widehat{\eta}^i) y^i : \; Wy = h(u^i) - T(u^i)x \right\},$$

provided the certainty equivalents $\widehat{p}_i$ and $\widehat{\eta}^i$ satisfy the condition:

$$\widehat{p}_i q(\widehat{\eta}^i) := \int_{\Omega} [p_i(\omega) q(\eta)] g(\omega)\, d\omega.$$

N.C.P. Edirisinghe [5] shows that the latter equivalent representation can be uniquely determined when the objective cost vector $q(\eta)$ is linear affine in η. To compute these certainty equivalents, consider the (nonsingular) vertex matrix V of the simplex $\widehat{\Omega}$, whose ith column is given by

$$(w_1^i, \ldots, w_{K+L}^i, 1)'.$$

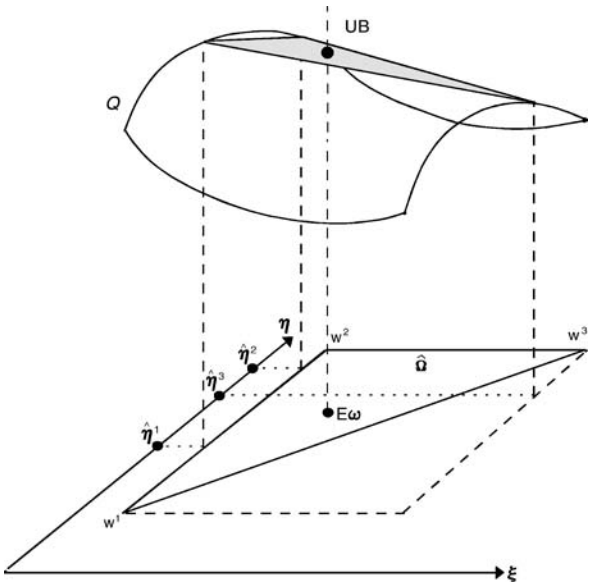
The inverse matrix of V is denoted by V^{-1} whose ith row is V_i^{-1}. Then, it can be shown that

$$\widehat{p}_i = V_i^{-1} \left(\mathsf{E}[\xi_1], \ldots, \mathsf{E}[\xi_K], \mathsf{E}[\eta_1], \ldots, \mathsf{E}[\eta_L], 1 \right)'.$$

Moreover, $\widehat{\eta}^i$ is evaluated for each coordinate $l (= 1, \ldots, L)$ by $\widehat{\eta}_l^i = r_l^i / \widehat{p}_i$, where r_l^i is the ith element of the $(K + L + 1)$-column vector,

$$V^{-1} \left(\mathsf{E}[\xi_1 \eta_l], \ldots, \mathsf{E}[\xi_K \eta_l], \mathsf{E}[\eta_1 \eta_l], \ldots, \mathsf{E}[\eta_L \eta_l] \right)'.$$

Consequently, the upper bound requires all first moments and second order moments including the variance information of η. The upper bound computations



Stochastic Programs with Recourse: Upper Bounds, Figure 2
Convex-concave upper bound in two dimensions

require solving only $K + L + 1$ linear programs. Therefore, the complexity of the bound does not grow exponentially with the number of random variables. This bound is illustrated in Fig. 2.

It can be verified through counterexamples that this upper bound is not associated with the solution of a moment problem having the concerned moment conditions. In contrast, generally, upper bounds in the convex-concave case are associated with solutions to moment problems. Also, when applied to the case of $K = 1$ and $L = 0$ – when only the right-hand side is random –, this upper bound reduces to the previously discussed Edmundson–Madansky upper bound.

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Two-stage Stochastic Programs with Recourse

References

1. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Math 4:238–252
2. Birge JR (1985) Aggregation bounds in stochastic linear programming. Math Program 31:25–41
3. Birge JR, Wallace SW (1988) A separable piecewise linear upper bound for stochastic linear programs. SIAM J Control Optim 26:725–739
4. Birge JR, Wets RJ-B (1989) Sublinear upper bounds for stochastic programs with recourse. Math Program 43:131–149
5. Edirisinghe NCP (1996) New second-order bounds on the expectation of saddle functions with applications to stochastic linear programming. Oper Res 44:909–922
6. Edirisinghe NCP, Ziemba WT (1992) Tight bounds for stochastic convex programs. Oper Res 40:660–677
7. Edmundson HP (1957) Bounds on the expectation of a convex function of a random variable. Techn Report RAND Corp P-382
8. Kall P, Wallace SW (1994) Stochastic programming. Wiley, New York
9. Madansky A (1959) Bounds on the expectation of a convex function of a multivariate random variable. A-STAT 30:743–746
10. Madansky A (1960) Inequalities for stochastic linear programming problems. Managem Sci 6:197–204
11. Morton DP, Wood RK (1999) Restricted-recourse bounds for stochastic linear programming. Oper Res 47:943–956
12. Slyke Rvan, Wets RJ-B (1969) L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J Appl Math 17:638–663
13. Wallace SW (1987) A piecewise linear upper bound on the network recourse function. Math Program 38:133–146
14. Wallace SW, Yan T (1993) Bounding multistage stochastic linear programs from above. Math Program 61(1):111–130

Stochastic Quasigradient Methods

YURI ERMOLIEV
IIASA, Laxenburg, Austria

MSC2000: 90C15

Article Outline

Keywords and Phrases
Introduction
Calculation of SQG
Convergence Properties
Nonsmooth Problems
Averaging Operations
General Constraints
References

Keywords and Phrases

Stochastic quasigradient (SQG) methods; Stochastic quasigradients (SQG); Nonstationary optimization; Generalized differentiable (GD) function; Stochastic approximation; Global optimization

Introduction

Traditional deterministic optimization methods are used for well defined objective and constraint functions, i. e., when it is possible to calculate exactly $F_0(x)$ to be minimized (or maximized) and to verify constraints

$$F_i(x) \le 0\,, \quad i = 1:m \tag{1}$$

for each decision vector $x = (x_1, \dots, x_n) \in X$, where the set X has a "simple" structure (for example, defined by linear constraints). Usually it is also assumed that gradients or subgradients (for nonsmooth functions) F_{ix} of the functions F_i, $i = 0, 1, \dots, m$ are easily calculated. *Stochastic Quasigradient (SQG) methods* have been developed for solving general optimization problems without exact calculation of F_i, F_{ix}. They incorporate basic ideas of standard optimization methods, random search procedures, stochastic approximation and statistical estimation. There are at least three main applications areas for SQG methods:

- Deterministic problems for which the calculation of descent directions is difficult (large-scale, nonsmooth, distributed, and nonstationary optimization models).
- Multiextremal problems where it is important to bypass locally optimal solutions.
- Problems involving uncertainties or/and difficulties in the evaluation of functions and their subgradients (stochastic, spatial, and dynamic optimization problems with multidimensional integrals, simulation and other analytically intractable models).

Thus, SQG methods are used in situations where the structure of the problem does not permit the application of one the many tools of deterministic optimization. They only require modest computer resources per iteration and reach with reasonable speed the vicinity of optimal solutions, with an accuracy that is sufficient for many applications. Further details on SQG methods can be found in references.

The main idea of the SQG methods as proposed in [1,2,3] (see also [5,6,7,8,9]) is to use statistical (biased and unbiased) estimates of objective and constraints functions and/or their gradients (subgradients). In other words, a sequence of approximate solutions $x^0, x^1, \dots$ is constructed by using random variables $\eta_i(k)$, and random vectors $\xi^i(k)$, $i = 0, \dots, m$ such that the conditional mathematical expectation for a given "history" $\mathcal{B}_k$ (say, $(x^0, \dots, x^k)$)

$$E[\eta_i(k)|\mathcal{B}_k] = F_i(x^k) + a_i(k)\,, \tag{2}$$

$$E[\xi^i(k)|\mathcal{B}_k] = F_{ix}(x^k) + b^i(k)\,, \tag{3}$$

where $a_i(k)$, $b^i(k)$ are "errors" (bias) of the estimates $\eta_i(k)$, $\xi^i(k)$. For the exact convergence of the sequence $\{x^k\}$ to optimal solutions $a_i(k)$, $b^i(k)$ must tend (in some sense) to 0 when $k \to \infty$. Vectors $\xi^i(k)$ are called *stochastic quasigradients*. If $b^i(k) \equiv 0$, then they are also called stochastic gradients for continuously differentiable $F_i(x)$ and stochastic subgradients (generalized gradients) for nonsmooth $F_i(x)$. In what follows notations $F(x), \eta(k), \xi(k)$ are also used instead of $F_0(x), \eta_0(k), \xi^0(k)$.

Consider the simplest SQG method. Assume that there are no constraints (1), X is a closed bounded (compact) convex set such that the orthogonal projection $\Pi_X(y)$ of a point y on X is easily calculated: $\Pi X(y) = Arg\min\{\|y - x\|^2 : x \in X\}$, for example, $\Pi_{a \le x \le b}(y) = \max\left[a, \min\{y, b\}\right]$. The SQG pro-

jection method is defined iteratively as following

$$x^{k+1} = \Pi_X \left[x^k - \rho_k \xi(k) \right], k = 0, 1, \dots , \qquad (4)$$

where ρ_k is a positive stepsize, x^0 is an arbitrary initial approximation (guess).

Calculation of SQG

Let us consider some important typical examples of SQG.

Example 1. Monte Carlo Optimization Various practical problems are so complicated that only a Monte Carlo simulation model is available [2,4,6] to indicate how the system might react to any given choice of the decision variable x. We always can view a given simulation run of such a model as the observation of an "environment" ω from a sample space Ω. To simplify matters, let us assume that only a single quantity $f(x, \omega)$ summarizes the output of the simulation ω for given x. The problem is then to minimize the expected performance (cost, risk, profit, a "distance" from given goals or a reference point, etc.):

$$F(x) = Ef(x, \omega). \qquad (5)$$

This is a typical stochastic optimization problem. Exact values of $F(x)$ are unknown explicitly. Available information at each current solution x^k and simulation run ω is $\eta(k) = f(x^k, \omega)$ satisfying (2) for $a(k) \equiv 0$. The vector $\xi(k)$ can be calculated as in the standard stochastic approximation procedures: At each step k for given x^k simulate random outcomes $f(x^k, \omega^{k0})$, $f(x^k + \Delta_k e^j, \omega^{k,j})$, $j = 1, \dots, n$, where $\Delta_k e^j$ is a positive increment in the direction e^j of jth coordinate axis; calculate

$$\xi(k) = \sum_{j=1}^{n} \Delta_k^{-1} \cdot \left[f(x^k + \Delta_k e^j, \omega^{k,j}) - f(x^k, \omega^{k,0}) \right] e^j . \qquad (6)$$

Simulations $\omega^{k,0}, \cdots, \omega^{k,n}$ are not necessarily independent: one possibility is to use only one simulation ω^k at each step $k : \omega^{k,0} = \dots = \omega^{k,n} = \omega^k$. The variance of such a single run estimate of SQG converges to 0 as $k \to \infty$, whereas for independent simulations it goes to ∞. Since $E\left[\xi(k)|x^k\right] = \sum_{j=1}^{n} \Delta_k^{-1} \left[F(x^k + \Delta_k e^j) - F(x^k) \right] e^j$, then for continuously differentiable $F(\cdot)$:

$$E\left[\xi(k)|x^k\right] = F_x(x^k) + C(k)\Delta_k, \qquad (7)$$

where $\|C(k)\| < const < \infty$ for all x^k from a bounded set X.

Example 2. Optimization by random search Suppose that $F(x)$ can be evaluated exactly but this is time consuming, say, because $F(x)$ is defined on solutions of differential equations or on solutions of other optimization problems. A purely random trial-and-error method (with $x^{k+1} \in X$ drawn at random until that $F(x^{k+1}) < F(x^k)$, and so on) may be time consuming since the probability to "hit" at random even a subspace as large as nonnegative orthant of n-dimensional Euclidean space is 2^{-n}. The traditional finite difference approximation

$$F_x(x^k) \approx \sum_{j=1}^{n} \Delta_k^{-1} \left[F(x^k + \Delta_k e^j) - F(x^k) \right] e^j \qquad (8)$$

requires $n + 1$ evaluations of $F(\cdot)$ and this also may be time-consuming. The SQG

$$\xi(k) = 3/2\Delta_k^{-1} \left[F(x^k + \Delta_k \varsigma^k) - F(x^k) \right] \zeta^k, \qquad (9)$$

where ς^k has independent uniformly distributed on $[-1, 1]$ components, requires only two evaluations of $F(x)$ at points x^k and $x^k + \Delta_k \varsigma^k$ independently of the dimensionality n. It is easy to see that vector (9) satisfies (7) for continuously differentiable $F(x)$.

Example 3. Finite difference approximations of subgradients The finite difference approximations (6), (8), (9) can not be used for nondifferentiable functions, e. g., for stochastic two-stage and minimax problems. SQG methods allow to develop simple finite-difference subgradient approximations for general (deterministic and stochastic) nonsmooth optimization problems. The slight randomization of (6), (8), (9) by substituting, roughly speaking, the current point x^k by a random point $\overline{x}^k = x^k + \nu^k$, where the random vector ν^k has a density and $\left\| \nu^k \right\| \to 0$ with probability 1, ensures their convergence even for locally Lipschitz and discontinuous functions [2,4,5], pp. 151, 320, [6,7].

Assume that $F(x)$ is a locally integrable (possibly discontinuous) function and the vector ν^k has sufficiently smooth density concentrated in a bounded set.

Then

$$\xi(k) = 3/2\Delta_k^{-1}\left[F(\overline{x}^k + \Delta_k\varsigma^k) - F(\overline{x}^k)\right]\zeta^k, \quad (10)$$

$$\xi(k) = 3/2\Delta_k^{-1}\left[f(\overline{x}^k + \Delta_k\varsigma^k, \omega^{k1}) - f(\overline{x}^k, \omega^{k,0})\right]\zeta^k \quad (11)$$

are SQG of $F(k,x) = EF(x+\nu^k)$ or so-called stochastic mollifier quasigradient (SMQG) of $F(x)$, which converges (in some sense) to $F(x)$ and for which $F_x(k,x)$ converges [4,6] to the set of subgradients $F_x(x)$. We have

$$E\left[\xi(k)|x^k\right] = F_x(k,x^k) + C(k)\Delta_k, \quad (12)$$

where $\|C(k)\| < const < \infty$ for all x^k from a bounded set. The analysis of convergence of x^k involves general ideas of nonstationary optimization (see Example 5). The important advantage of this approach is that $F(k,x)$ smoothes out rapid oscillations of $F(x)$ and reflects general trend of $F(x)$. In this sense $F(k,x)$ provides a "bird's eye" point of view on the "landscape" $F(x)$ enabling $\{x^k\}$ to bypass inessential local solutions. "Large" enough ν^k force the procedure to concentrate on essential (global) solutions.

Example 4. Global optimization The simplest way to introduce the "inertia" in the gradient type procedure to bypass some local solutions is to perturb the gradient $F_x(x^k)$ by a random vector ν^k, i. e. to consider $\xi(k) = F_x(x^k) + \nu^k$, $E\nu^k = 0$. A special choice of ν^k corresponds to the simulated annealing. Another approach is to cut off local solutions by sequential convex approximations [6].

Example 5. Nonstationary optimization Many applied problems, [2,5], pp. 152–156, [6], such as in Example 3, can be formulated as optimization problems with objective function $F_0(k,x)$ and constraints functions $F_i(k,x)$ changing at each step $k = 0, \ldots$ In this case a SQG - method on step k performs one step of the minimization of $F_0(k,x)$ using estimates of $F_{ix}(k,x)$, $i = 0, \ldots, m$. An important case arises when $F_i(k,x) \to F_i(x)$ in some sense. Then it is possible to prove that $F_0(k,x^k) \to \min F_0(x)$. In the general case, it is possible to specify wide variety of situations for which $\left|F_0(k,x^k) - \min F_0(x,k)\right| \to 0$, $k \to \infty$.

Convergence Properties

SQG methods generate random sequences of approximate solutions $\{x^k(\omega)\}$ and values $\{F(x^k(\omega))\}$ indexed by ω from an appropriately defined probability space. Most important, from practical point of view, is the convergence of $x^k(\omega)$ (or $F(x^k)$) to the set of local (in general) solutions X^* (set $F^* = F^*(X^*)$) for almost all ω (with probability 1). The convergence with probability 1 of the sequence $\{F(x^k)\}$ to the set F^* was proved for rather general nonsmooth (generalized differentiable, locally Lipschitz and even semicontinuous) functions covering a wide range of applications. The limit points of $\{x^k(\omega)\}$ for each ω form a connected set from X^*. The convergence $x^k(\omega) \to X^*$ with probability 1 takes place under "convexity" assumptions. The global convergence in general cases requires special stochastic mechanisms [2,6]. In all cases the convergence requires special choice of the stepsize ρ_k. Due to the complexity of the problems, ρ_k can not be chosen in a way, that guarantees the monotonic decrease of $F(x) : F(x^{k+1}) < F(x^k)$, $k = 0, 1, \ldots$ A relatively flexible requirement that often guarantees the convergence of the sequence $\{F(x^k)\}$ with probability 1 is that with probability 1

$$\rho_k \geq 0, \sum_{k=0}^{\infty}\rho_k = \infty\,, \quad \sum_{k=0}^{\infty} E\left[\rho_k\|b(k)\| + \rho_k^2\|\xi(k)\|^2\right] < \infty. \quad (13)$$

For example, consider (6) with dependent observations $\omega^{k,0} = \omega^{k,1} = \cdots = \omega^k$ and $f(x,\omega) < const < \infty$. In this case we can always assume in practice that $\|\xi(k)\| \leq const < \infty$. Then condition (13) leads to the requirement $\rho_k \geq 0, \sum_{k=0}^{\infty}\rho_k = \infty, \sum_{k=0}^{\infty} E(\rho_k\Delta_k + \rho_k^2) < \infty$, which is satisfied for $\rho_k = C/(k+1)$, $\Delta_k = D/(k+1)$ with constants C, D. In practice C, D are usually adjusted [2,6] at each step by taking into account the history $\mathcal{B}_k$, for example by using values $\overline{F}(k) = (k+1)^{-1}\sum_{s=0}^{k} f(x^s,\omega^s)$. Different adaptive SQG methods with adaptive adjustment of ρ_k as a function of $\mathcal{B}_k$ have been studied in [5], pp. 373–385, 316–322, [11]. One idea is to choose ρ_k, more or less so as minimize $E\left[F(x^k - \rho\xi(k))|x^k\right]$. It leads to adaptive modifications of ρ_k that are proportional to the product $< \xi(k+1), \xi(k) >$. The important issues

of determining the moment of termination (stopping time) and the confidence intervals for approximate solutions have also been studied [5], pp. 353–373.

SQG methods require appropriate techniques to prove their convergence. These techniques ([2,5], pp. 155–156), [7,9] can be viewed as stochastic Lyapunov method for the stability analysis of nonsmooth dynamic systems. The main idea is to show that $\{x^k(\omega)\}$ for each ω leaves any neighborhood of points which do not belong to X^* with decreasing values of some (in general nonsmooth) Lyapunov function.

Nonsmooth Problems

The common sense arguments in using SQG methods for nonsmooth problems may be misleading (see applications of SQG methods). An exception, in a sense, is the class of problems with so-called generalized differentiable (GD) functions [4,7]. The class of GD functions is closed with respect to *min* and *max* operators and superpositions. The following important formula holds for the set ∂f of subgradients

$$\partial \max\{f_1(x), f_2(x)\} = co\left\{\partial f_i(x) = \max\left(f_1, f_2\right)\right\} \tag{14}$$

and subgradients of a composite function $\Psi(f_1, \ldots, f_r)$ are calculated by intuitively obvious chain rules. The class of GD functions is also closed with respect to the expectation operator:

$$\partial F(x) = E\partial f(x, \omega), F(x) = Ef(x, \omega), \tag{15}$$

where $f(\cdot, \omega)$ is a GD function.

Formulas (14), (15) provide a useful tool for calculating subgradients. Unfortunately, for general classes of nonsmooth functions their direct use becomes tedious and in some cases (14), (15) invalid. The most promising approach seems to use SMQG similar to (10), (11).

Averaging Operations

The methods (4) and many other SQG methods have the same basic structure as their deterministic counterparts. The following stochastic linearization method possesses an essential new feature. Consider again the minimization of $F(x), x \in X$. Assume that $F(x)$ is a continuously differentiable function, and that X is a convex compact. The method is defined iteratively by

$$x^{k+1} = x^k + \rho_k(\widehat{x}^k - x^k), \quad 0 < \rho_k < 1\,, \tag{16}$$

$$\widehat{\xi}(k+1) = \widehat{\xi}(k) + \delta_k(\xi(k+1) - \widehat{\xi}(k))\,,$$
$$\widehat{x}^k \in \arg\min\left\{\langle\widehat{\xi}(k), x\rangle : x \in X\right\}, \tag{17}$$

where x^0 is an arbitrary initial approximation from X.

The well-known deterministic counterpart has $\delta_k \equiv 0$, $\xi(k) \equiv F_x(x^k)$. Simple examples show that without the averaging operation (17), i. e., $\delta_k \equiv 0$, method (16) does not converge. For convergence, it is required in addition to (13) that

$$\delta_k \geq 0, \quad \rho_k/\delta_k \to 0, \sum_{k=0}^{\infty} E\delta_k^2 < \infty. \tag{18}$$

Method (16) is generally used when X is defined by linear constraints. In this case a linear subproblem is solved at each step k. In contrast, the projection method (4), requires the solution of quadratic subproblem. Note that in both cases only small perturbations occur at each step in the objective functions of subproblems. Therefore, only small adjustments of the preceding solutions are needed. This method can be modified for nondifferentiable functions and constraints (1). In particular, it is possible to use SMQG as in (10), (11) (stochastic finite-difference approximations) for locally Lipschitz functions.

The use of averaging operations similar to (17) is often crucial for the convergence of SQG methods as well as their efficiency and robustness.

This operation is also applied to directions $\xi(k)$ in (4), i. e. instead of $\xi(k)$ the vector $\widehat{\xi}(k)$ is used such that for $k = 0, 1, \ldots$

$$\widehat{\xi}(k+1) = \widehat{\xi}(k) + \delta_k\left[\xi(k+1) - \widehat{\xi}(k)\right]\,,$$
$$\widehat{\xi}(0) = \xi(0)\,.$$

It introduces inertia or "heavy ball" properties for procedure (4) in addition to its inherent global features due to involving stochastic mechanisms. It may also reduce the variance of the SQG. The averaging of approximate solutions x^k may also improve the asymptotic properties [10].

General Constraints

Constraints (1) for which the functions $F_i(x)$ are not known explicitly can be treated by using penalty functions, the averaging operation, and the ► Lagrange multipliers. Consider the minimization of $F_0(x), x \in X$, subject to constraints (1). Instead we can minimize a penalty function, for example

$$\Psi(x,c) = F_0(x) + c\sum_{i=1}^{m} \max\{0, F_i(x)\} \tag{19}$$

on X, where c is a large number. If exact values $F_i(x)$ are not available, then $\max\{0, F_i(x)\}$ is unknown. Note that if F_i is GD functions, then $\Psi(\cdot, c)$ is also GD function with subgradient $\Psi_x = F_{0x} + c\sum_{i=1}^{m} \max\{0, F_i\} F_{ix}$, where F_{0x}, F_{ix} are subgradients of F_0, F_i.

Assume there are available the statistical estimates $\xi^0(k), \xi^i(k)$ satisfying (3). Consider the SQG procedure with embedded averaging operation (21)

$$x^{k+1} = \Pi_X\Big[x^k - \rho_k\big(\xi^0(k) + c\sum_{i=1}^{m} \max\{0, \widehat{F}_i(x)\}\xi^i(k)\big)\Big] \tag{20}$$

$$\widehat{F}_i(k+1) = \widehat{F}_i(k) + \delta_k\Big[\eta^i(k) - \widehat{F}_i(k)\Big], \quad i = 1:m. \tag{21}$$

The convergence of this method with probability 1 requires conditions similar to those in (13), (18). The following procedure converges also under conditions similar to (13), (18):

$$x^{k+1} = \Pi_X\Big[x^k - \rho_k\alpha^k\Big], \tag{22}$$

$$\alpha^k = \begin{cases} \xi^0(k), & \text{if} \quad \max \widehat{F}_i(k) = \widehat{F}_{i_k}(k) \le 0, \\ \xi^{i_k}(k), & \text{if} \quad \widehat{F}_{i_k}(k) > 0\,. \end{cases}$$

Assume that $F_i(x), i = 0, 1, \ldots, m$ are convex functions and X is a convex compact. The SQG Lagrange multiplier method is characterized by the relations

$$x^{k+1} = \Pi_X\Big[x^k - \rho_k\big[\xi^0(k) + \sum_{i=1}^{m} \lambda_i(k)\xi^i(k)\big]\Big], \tag{23}$$

$$\lambda_i(k+1) = \min\big[\max\{0, \lambda_i(k) + \delta_k\eta_i(k)\}, C\big], \tag{24}$$

where $\eta_i(k), \xi^i(k)$ are estimates of $F_i(x^k), F_{ix}(x^k)$ as in (2), (3); F_{ix} are subgradients of $F_i(x)$; C is a large enough number; ρ_k, δ_k are stepsizes.

The procedure (23) can be interpreted in the context of nonstationary optimization: x^{k+1} is the result of the one-step of the procedure (4) applied to the nonstationary function $\Psi(k,x) = F_0(x) + \sum_{i=1}^{m} \lambda_i(k)F_i(x)$ with SQG $\xi^0(k) + \sum_{i=1}^{m} \lambda_i(k)\xi^i(k)$. It was proved [9] that $\min_{s\le k} F_0(x^s)$ converges to $\min F_0(x)$ (in the feasible set) with probability 1, provided that $F_0(x)$ is strictly convex, $\rho_k = \delta_k$, and (13) is satisfied with $\|\xi(k)\|^2$ substituted by $\sum_{i=0}^{m} \|\xi^i(k)\|^2$. The convergence for the convex functions $F_0(x)$ – not necessary strictly convex – was established under additional assumptions on δ_k similar to (18). The convergence of the sequences $\sum_{s=0}^{k} \rho_s x^s / \sum_{s=0}^{k} \rho_s$, $\sum_{s=0}^{k} \rho_s \lambda(s) / \sum_{s=0}^{k} \rho_s$, $\lambda(s) = (\lambda_1(s), \ldots, \lambda_m(s))$, to the saddle points of the Lagrange function does not require the strict convexity of $F_0(x)$ and different stepsizes ρ_k, δ_k (see stochastic quasigradient methods in minimax problems).

References

1. Ermoliev YM (1969) On the method of the generalized stochastic gradients and stochastic quasi-Fjer sequences, Kibernetica, No. 2, 1969, pp 73–84 (in Russian), English translation in Cybernetics 5(2):208–220
2. Ermoliev YM (1976) Methods of stochastic programming, Nauka, Moscow
3. Ermoliev YM, Nekrylova ZV (1967) The method of stochastic gradients and its applications, Proceedings of the Seminar: Theory of Optimal Solutions. Akad Nauk Ukraine, Kiev 1:24–47
4. Ermoliev YM, Norkin VI (1997) On nonsmooth and discontinuous problems of stochastic systems optimization. Eur J Oper Res 101(2):230–243.
5. Ermoliev YM, Wets R (1988) Numerical techniques for stochastic optimization. Springer Ser Comput Math, vol 10. Springer, Berlin
6. Gaivoronski A (2005) SQG Software for solving stochastic programming problems with stochastic quasigradient methods. In: Wallace SW, Ziemba WT (eds) Applications of Stochastic Programming. SIAM MPS 38–60.
7. Gupal AM, Michalevich VS, Norkin VI (1987) Methods on Nonconvex Optimization. Nauka, Moscow (in Russian)
8. Marti K (2005) Stochastic optimization methods. Springer, Berlin
9. Nurminski E (1979) Numerical methods for deterministic and stochastic minimax problems. Naukova Dumka, Kiev

10. Polyak BT, Juditsky A (1992) Accelaration of stochastic approximation by averaging. SIAM J Control Optim 30:838–855
11. Uryasev S (1992) A stochastic quasigradient algorithm with variable metric. Ann Oper Res 39:251–267

Stochastic Quasigradient Methods: Applications

YURI ERMOLIEV
IIASA, Laxenburg, Austria

MSC2000: 90C15

Article Outline

Keywords and Phrases
Maximization via Ordering Rules
Expected Utility Maximization
  Example 1. Portfolio Selection Problems
Stochastic Optimization Problems
"Hit-or-Miss" Decision Problems
Pollution Control
Queuing Networks
Stochastic Dynamic Systems
Optimization of Discrete Event Dynamic Systems
  Example 2. Managing Catastrophic Risks
  Neural Nets
  Automaton Learning Problem
References

Keywords and Phrases

Expected utility; Stochastic optimization (STO) problem; Standard stochastic approximation; Pollution control; Queuing systems; Catastrophic risks; Neural nets; Automaton learning

Let us consider some optimization problems that require SQG methods:

$$\min\{F_0(x)|F_i(x) \le 0, i = 1:m, x \in X\}\,. \tag{1}$$

Maximization via Ordering Rules

Realistic decision problems involve multiple objectives and inherent uncertainty. Generally, it is not possible to optimize several objectives simultaneously; for instance, minimizing cost while maximizing reliability at the same time. Therefore, it is necessary to strike a balance between various objectives and if we can specify some (utility) function $U(x)$ that combines all objectives into a scale index of preferability, then the problem of decision making can be cast in the format of the standard optimization problem of maximizing $U(x)$. Unfortunately, finding such a function may be a very difficult task. It is often much easier to arrive at a preference ordering, [4], p. 176, among feasible decisions (based on some rules or direct judgements by decision makers). Therefore, let us assume that instead of $U(x)$ there is a given consistent "mechanism" (ordering $\succcurlyeq$) that can verify whether a vector x is preferred to y ($x \succ y$), and yields outcomes that are equivalent to those of the unknown continuous function $U(x) > U(y)$. Let us define $F(x) = -U(x)$ and

$$\xi(k) = \begin{cases} -\varsigma(k) \text{ if } & x^k + \Delta_k\varsigma(k) \succ x^k\,, \\ \varsigma(k) \text{ if } & x^k + \Delta_k\varsigma(k) \preccurlyeq x^k\,, \end{cases}$$

where $\Delta_k \to 0$ and $\varsigma(k)$, $k = 0, 1, \dots$ are independent samples of the random vector ς uniformly distributed over the unit sphere. Then, $E[\xi(k)|x^k] = -\alpha U_x(x^k)/\left\|U_x(x^k)\right\|$ for continuously differentiable $U(x)$, where α is a positive number, i. e., the vector $\xi(k)$ estimates the direction of gradient $F_x(x^k)$ and can be used in SQG methods (also *Stochastic quasigradient methods*) to maximize $F(x)$ without knowing this function.

Expected Utility Maximization

In practice, a given decision x often results in different outcomes $g(x,\omega) = (g_1(x,\omega), g_2(x,\omega), \dots, g_r(x,\omega))$ affected by uncertainty ω ("environment", parameters). Using either objective or subjective probability it is possible to treat ω as a stochastic variable that is characterized by the priori probability measure $P(\cdot)$. The expected utility is an evaluation

$$\begin{aligned} U(x) &= Eu(g_1(x,\omega), g_2(x,\omega), \dots, g_r(x,\omega)) \\ &= \int u(g(x,\omega))P(d(\omega))\,, \end{aligned} \tag{2}$$

which is linear with respect to P, i. e., if ω is a mixture of ω' and ω'' with probability α and $1-\alpha$, $0 < \alpha < 1$, then

$$U(x) = \alpha Eu(g(x,\omega') + (1-\alpha)Eu(g(x,\omega''))\,. \tag{3}$$

The maximization of (2) is a special case of the general stochastic optimization (STO) problem, which does not necessarily satisfy (3). The expected utility theory neglects the major difficulties involved in the maximization of $U(x)$: exact evaluation of $U(x)$ as the integral (2), analytically or numerically, is only possible in exceptional cases. Consequently, applications suffer from highly restrictive assumptions, for example that ω has a discrete probability distribution with a small number N of possible states $\omega = 1, \dots, N$. SQG methods avoid the calculation of integrals (2). Take $F(x) = -Eu(g(x,\omega))$. Assume that functions $u(z_1, \dots, z_r)$, $z_i = g_i(x,\omega)$ are known explicitly and gradients $u_z(\cdot)$, $g_{ix}(\cdot)$ are calculated exactly for given z, x, ω. Then, the SQG of $F(x)$ is

$$\xi(k) = (\xi_1(k), \dots, \xi_n(k)), \xi_j(k) = -\sum_{i=1}^{r} u_{z_i}(g_1(x^k,\omega), \dots, g_r(x^k,\omega)) \, g_{ix_j}(x^k,\omega) \,. \quad (4)$$

Example 1. Portfolio Selection Problems

The advantage of using (4) is evident even in a simple single-period model. Assume that at the beginning of the period (week, month, year, etc.) an investor allocates funds among different investment alternatives with random rates of returns. He may, for example, be in charge of investing the foreign currency reserve of a central bank, decide on projects financing, or manage a mutual fund. Let $j = 1, \dots, n$ denotes assets (or classes of investment) with random rates of return ω_j; x_j is a share of asset j to be included in the portfolio; c_j is the current price; W is the initial fund. The net portfolio future value is now $g(x,\omega) = W - \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} \omega_j x_j$, where $x = (x_1, \dots, x_n)$ satisfies feasibility constraints. The expected utility $U(x) = Eu(W + \sum_{j=1}^{n}(\omega_j - c_j)x_j) = \int \dots \int u(g(x,\omega))P(d\omega)$, where $u(z)$ is assumed to be a monotonically increasing function of z. If each ω_j is characterized by a finite number M of states, the expectation $U(x)$ is reduced to the sum of $N = M^n$ terms. The number M^n is astronomically large even for $M = 10$, $n = 10$, i. e. although ω is characterized by a finite number N of states, the exact evaluation of $U(x)$ is still a tedious task. The vector $\xi(k)$ in (4) has components $\xi_j(k) = -u'(W + \sum_{j=1}^{n}(\omega_j^k - c_j)x_j^k)(\omega_j^k - c_j)$, where $\omega^k, k = 0, 1, \dots$ are independent samples of ω from a probability distribution.

Stochastic Optimization Problems

It is often impossible to summarize the outcomes $g(x,\omega)$ of a decision x into a single index of preferability (2). Such cases lead to the following general STO problem: Given a probability space that gives a description of the possible environments $\omega \in \Omega$ with associated probability measure P, a stochastic optimization (STO) problem is to find $x \in X \subset R^n$ such that constraints

$$F_i(x) = E[f_i(x,\omega)] = \int f_i(x,\omega)P(x,d\omega) \le 0, \quad i = 1:m \,, \quad (5)$$

are satisfied and an objective function

$$F_0(x) = Ef_0(x,\omega)] = \int f_0(x,\omega)P(x,d\omega) \quad (6)$$

is minimized. Functions $f_i(x,\omega)$ and $F_i(x)$, $i = 0, 1, \dots, m$, are called correspondingly *sample and expectation functions*. In some problems functions $f_i(x,\omega)$ depend, [4], pp. 17, 173, not only on the outcomes $g(x,\omega)$ but also on their expectations $Eg(x,\omega)$:

$$f_i(x,\omega) = \Psi_i(x, Eg(x,\omega), \omega) \,. \quad (7)$$

In this case, the calculation of $f_i(x,\omega)$ requires the calculation of the expectation $Eg(x,\omega)$, i. e. functions $f_i(x,\omega)$ themselves are not known explicitly.

Functions $f_i(x,\omega)$ and even $F_i(x)$ in (5) are often discontinuous. For example, if we set $f_i(x,\omega) = 1 - r_i$ when an outcome $g_i(x,\omega) \ge 0$, and $f_i(x,\omega) = -r_i$ otherwise, then for the given i constraint (5) corresponds to the safety or chance constraint

$$F_i(x) = Pr[g_i(x,\omega) \ge 0] - r_i \le 0 \,, \quad (8)$$

where $0 \le r_i \le 1$ is a safety level (risk factor). For a discrete distribution of ω, function $F_i(x)$ in (8) is a discontinuous function.

A rather general approach, besides SQG methods, consists of approximating the distribution P in (5), (6) by a discrete distribution $p^N = (p_1, \dots, p_N)$, i. g., by an empirical distribution $p_s = 1/s$, $s = 1, \dots, N$. As a result, the integrals in (5), (6) are replaced by sums, i. e., functions $F_i(x)$ are approximated

by $F_i^N(x) = \sum_{s=1}^N p_s f_i(x, \omega_s)$, and resulting problem could be solved, if possible, by standard deterministic methods. This approach can be used only when P does not depend on x, N is a small number (as in Example 1), and $f_i(\cdot, \omega)$ are analytically tractable functions. Besides, the deterministic approximations $F_i^N(x)$ may destroy the smoothness, the continuity and even the convexity [3] of functions $F_i(x)$. The convergence of $\min F_0^N$ to $\min F(x)$, $N \to \infty$, is established in practically all important cases. Despite this, the number of discontinuities and local optimal solutions of approximating problems as we can see from (8) and further examples, may tend to ∞ without having connections with solutions of the original problem. The SQG methods have advantages in such cases. They deal directly with functions $F_i(x)$ allowing to utilize a remarkable specific of many STO problems: despite nonsmooth, discontinuous and even nonconvex functions $f_i(\cdot, \omega)$, and hence $F_i^N(x)$, functions $F_i(x)$ are often continuous and convex.

If P in (5) does not depend on x and subgradients $f_{ix}(\cdot, \omega)$ are calculated exactly, then, under certain regularity assumptions that ensure the interchangeability of differentiation and integration operations, $f_{ix}(x^k, \omega)$ is a SQG of $F_i(x)$ at $x = x^k$. In case $P(x, d\omega) = p(x, \omega)d\omega$ and $p(x, \omega)$, $f_i(x, \omega)$ are calculated exactly, a SQG of $F_i(x)$ is computed as

$$\xi^i(k) = f_{ix}(x^k, \omega) + f_i(x^k, \omega)\frac{p_x(x^k, \omega^k)}{p(x^k, \omega^k)}\,. \qquad (9)$$

If $P(x, d\omega)$ is not known but the function $\int f_i(x, \omega)P(y, d\omega)$ is continuously differentiable with respect to y at $y = x$, then the following SQG has the bias $\|b^i(k)\| \le const\,\Delta_k$ for all x^k from a bounded set:

$$\xi^i(k) = f_{ix}(x^k, \omega^{k,0}) + \sum_{j=1}^n \Delta_k^{-1}[f_i(x^k, \omega^{k,j}) - f_i(x^k, \omega^{k,0})]e^j\,, \qquad (10)$$

where $\omega^{k,j}$, $j = 1, \dots, n$, are independent observations of ω from $P(x^k + \Delta_k e^j, d\omega)$, and $\omega^{k,0}$ from $P(x^k, d\omega)$.

For general nonsmooth $F_i(x)$ the SQG are calculated by using random points $\overline{x}^k$ instead of x^k as in (11), (12) in *Stochastic quasigradient methods*:

$$\xi^i(k) = \sum_{j=1}^n \Delta_k^{-1}[f_i(\overline{x}^k, \omega^{k,j}) - f(\overline{x}^k, \omega^{k,0})]e^j\,, \qquad (11)$$

$$\xi^i(k) = 3/2\Delta_k^{-1}[f_i(\overline{x}^k + \Delta_k\varsigma^k, \omega^{k,1}) - f(\overline{x}^k, \omega^{k,0})]\varsigma^k\,, \qquad (12)$$

where ς^k has independent uniformly distributed components on $[-1, 1]$.

The choice (11) with $\overline{x}^k = x^k$ corresponds to the standard Stochastic Approximation originating from the classical papers of Robbins–Monro and Kiefer–Wolfowitz (see, for example, [10,11]). It was proposed for unconstrained minimization $F(x) = \int f(x, \omega)P(x, d\omega)$, where $F(x)$ is a twice continuously differentiable convex function.

Vectors (10)–(12) have (for fixed $x^k, \overline{x}^k$) unbounded variance $Var\xi(k) = O(\Delta_k^{-2}) \to \infty$, $k \to \infty$, assuming $Var f_i(x, \omega) < const$. If P does not depend on x, then for dependent $\omega^{k,0} = \dots = \omega^k$ (*single run SQG*) they have $Var\xi(k) = O(\Delta^k) \to 0$, $k \to \infty$.

An *averaging operation* similar to (17) in ▶ stochastic quasigradient methods, is used to confront the complexity of the sample function (7). Assume that $\Psi_i(x, y, \omega)$ is calculated exactly for a given (x, y, ω) and consider the sequence

$$\widehat{g}(k+1) = \widehat{g}(k) + \delta_k(g(x^k, \omega^k) - g(x^k)), \quad k = 0, 1, \dots\,,$$

where x^k is the current approximate solution, and ω^k, $k = 0, 1, \dots$, are independent samples of ω. Then, under general requirements on δ_k, with probability 1 $\|\widehat{g}(k) - E[g(x^k, \omega^k)|x^k]\| \to \infty$, $k \to \infty$. Therefore, $\Psi(x^k, \widehat{g}(k), \omega^k)$ can be used as an estimate of $f_i(x^k, \omega^k)$ and $\xi^i(k)$ can be calculated by *chain rules* such as

$$\xi^i(k) = \Psi_{ix}(x^k, \widehat{g}(k), \omega^k) + \sum_{l=1}^r \Psi_{igl}(x^k, \widehat{g}(k), \omega^k)g_{lx}(x^k, \omega^k) \qquad (13)$$

for GD functions, or by using finite-difference approximations of Ψ_{ix}, Ψ_{igl}, g_{lx} as in (11), (12). Single-run SQG of type (11), (12) for dependent $\omega^{k,0} = \dots = \omega^k$ provide surprisingly more accurate estimates [2].

"Hit-or-Miss" Decision Problems

This problem [3] illustrates typical difficulties in dealing with optimization of continuously differentiable expectation functions $F(x)$ when sample functions $f(x,\omega)$ are nonsmooth. Assume that at some point in the evolution of a system (ecosystem, nuclear power plant, economic system), a catastrophe at a random time ω could occur if the decision maker does not intervene and control ongoing processes at a time moment $x \in [0,T]$. The profit in the absence of a catastrophe, $x < \omega$, is proportional to $\tau = \min(x,\omega)$, but $\omega \le x$ leads to high losses b. Suppose that ω is distributed on the interval $[0,T]$ with a probability density function $\mu(\omega)$ and the sample performance function

$$f(x,\omega) = \begin{cases} -ax & \text{if} \quad 0 \le x < \omega \\ b - a\omega & \text{if} \quad \omega \le x \le T\,. \end{cases}$$

The function $f(x,\omega)$ is discontinuous with respect to both variables. The expected performance function has the form

$$\begin{aligned} F(x) &= Ef(x,\omega) \\ &= E[f(x,\omega)I_{x<\omega}] + E[f(x,\omega)I_{x\ge\omega}]\,, \end{aligned} \tag{14}$$

where $I_A = I_A(\omega)$ is the *indicator function* of the event A: $I_A(\omega) = 1$ if $\omega \in A$ and $I_A(\omega) = 0$, otherwise. The gradient of $f(\cdot,\omega)$ exits everywhere except for $x = \omega$. Define

$$f_x(x,\omega) = \begin{cases} -a, 0 \le x < \omega \\ 0, \omega \le x \le T\,. \end{cases}$$

Obviously, the expectation $Ef_x(x,\omega)$ exists, but the *"interchangeability"* formula is not valid: $F_x(x) \ne Ef_x(x,\omega)$. Indeed the direct differentiation of both sides in (14) yields $F_x(x) = (f(x,x) - f(x,x_{+0}))\mu(x) + Ef_x(x,\omega)$ where $f(x,x_{+0}) = \lim_{x\to x+0} f(x,y)$. Therefore the discontinuity of $f(x,\omega)$ results in a new additional to $f_x(x,\omega)$ term

$$\xi(k) = \left(f(x^k,x^k) - f(x^k,x^k_{+0})\right)\mu(x) + f_x(x,\omega). \tag{15}$$

It is clear that the approximations $F^N(x)$ of function (14) have increasing number of discontinuities and local optimal solutions.

Pollution Control

A feature common to most of the models applied for the design of *pollution control* polices is the use of transfer coefficients a_{ij} linking the amount of pollution x_i emitted by source i to the resulting pollution concentrations y_j at location j as $y_j = \sum_{i=1}^n a_{ij}x_i$, $j = 1,\dots,m$. The coefficients a_{ij} are often computed by means of Gaussian type diffusion equations. These equations are solved over all possible meteorological conditions, and the outputs are then weighted by the frequencies of the meteorological inputs over a given time interval, yielding average a_{ij}. The deterministic models determine cost-effective emission strategies subject to achieving exogenously specified environmental goals, such as ambient standards q_j at receptors: $y_j \le q_j$. The natural improvement of deterministic models is the inclusion of constraints that account for the random nature of the coefficients a_{ij} in order to reduce the occurrence of extreme events:

$$\begin{aligned} F_i(x) = \sum_{i=1}^n x_j\overline{a}_{ij} + \gamma_i E\max\left\{0, \sum_{i=1}^n x_j(a_{ji} - \overline{a}_{ji})\right\} \\ - q_j \le 0, i = 1:m, \overline{a}_{ji} = Ea_{ji}\,, \end{aligned}$$

where γ_i is a risk coefficient which enforces the constraints to reduce the chance for actual deposition to exceed the average value. The function $F_i(x)$ does not satisfy the linearity requirements (3) and, in general, is not continuously differentiable (although it is a convex function). It's a SQG

$$\xi^i(k) = \overline{a}_{ji} + \gamma_i \begin{cases} 0, \sum_{i=1}^n x_j^k a_{ij}^k < \sum_{i=1}^n x_j^k \overline{a}_{ji} \\ a_{ij}^k - \overline{a}_{ji}, \text{ otherwise,} \end{cases}$$

where a_{ij}^k, $k = 0,1,\dots$ are independent observations of a_{ij}.

Queuing Networks

A typical situation with implicitly given nonsmooth sample functions $f(x,\omega)$ occurs in the optimization of *queuing networks* [3]. A network consists of nodes (devices) which "serve" flows of "messages". At any moment the device $i = 1,2,\dots$ serves only one message, which is then transferred to another node in accordance with a certain routing procedure defining a destination node for jth message served at the ith device. If the device is busy, then the message is waiting in the queue

and is served according to the rule: first come – first served. Let $\tau_{ij}(x,\omega)$ be (random) service time of message j at i depending on some control parameter x and uncontrolled (random) parameter ω; $\alpha_{ij}(x,\omega)$ – the arrival time of message j to node i; $\beta_{ij}(x,\omega)$ – the time when i starts to serve j; $\gamma_{ij}(x,\omega)$ – the time when i finishes servicing j. The logic of a node operation is described by the following recurrent relations: $\gamma_{ij} = \beta_{ij} + \tau_{ij}$, $\beta_{ij} = \max\{\gamma_{i(j-1)}, a_{ij}\}$, $j = 1,2,\ldots$. From this follows that various important indicators (waiting times, queue length node loads and etc.) of network performance can be expressed through functions $\tau_{ij}(x,\omega)$ by max and min operations, i. e. they are GD functions assuming $\tau_{ij}(\cdot,\omega)$ are such functions. The calculation of SQG can be based on (15) in "Stochastic quasigradient methods".

Stochastic Dynamic Systems

Stochastic dynamic systems are usually defined by implicitly given sample performance functions $f_i(x,\omega)$. The decision vector x represents a sequence of decisions (control actions) $x(t)$ over a given time horizon $t = 0,\ldots,T$: $x = (x(0),\ldots,x(T-1))$. In addition to x, there may also be a group of state variables $z = (z(0),\ldots,z(T))$ that record the state of the system. The variables x, z are connected through a system of equations:

$$z(t+1) = g(t,z(t),x(t),\omega)\,,$$
$$t = 0,\ldots,T-1, z(0) = z^0\,. \quad (16)$$

Objective and constraints functions are defined as expectations of some sample performance functions $h_i(z,x,\omega)$, $i = 0,1,\ldots,m$.

Due to (16), variables z are implicit functions of (x,ω), i. e. $z = z(x,\omega)$. Therefore, h_i are also implicit functions of (x,ω): $f_i(x,\omega) = h_i(z(x,\omega),x,\omega)$, $i = 0,1,\ldots,m$, and the resulting *stochastic dynamic optimization problem* can be viewed as a stochastic optimization problem of the type (5)–(6) with implicitly given sample performance functions $f_i(x,\omega)$. A way to solve this problem is to use the SQG (11), (12). In particular (12) requires the calculation of only two "trajectories" $z(t)$, $t = 0,1,\ldots,T$ at each step of SQG procedures. If functions $g(\cdot,\omega)$, $h_i(\cdot,\omega)$ have well-defined analytical structure and the probability measure P does not depend on x, then subgradients $f_{ix}(x^k,\omega)$ are calculated (for fixed ω) using analytical formulas from *nonsmooth analysis* [1][pp. 17, 175 in 4].

Optimization of Discrete Event Dynamic Systems

The well-defined analytical structure of functions $g(\cdot,\omega)$, $h_i(\cdot,\omega)$, in (16) is typical for applications in mechanics and physics. Important problems in operation research, economics, ecology, finance, reliability theory, communicational networks [3,4,8,9] deal with cases where these functions and the probability measure are composed of so many components involving logical variables (as in queuing systems) that no explicit "smooth" analytical expression can be derived. Discrete events may change the state of the *discrete event dynamic system* in a discontinuous fashion, implying that the functions $g(\cdot,\omega)$, $h_i(\cdot,\omega)$ are nonsmooth. This often rules out the interchangeability of differentiation and integration operations, as in the *"hit-or-miss" decision problems*. Nonetheless, it is possible to develop various techniques for calculating stochastic quasigradients [2,3,8]. Let us consider a typical situation.

Example 2. Managing Catastrophic Risks

Increasing likelihood of extreme catastrophic events which may affect large territories and communities dominates the discussion of global change processes. The analysis of robust catastrophic risk management decisions [6] requires new approaches based on explicit analysis of endogenous risk processes involving various agents such as governments, insurers, and individuals. Risk processes describing the ruin of an agent or depletion of resources have similar structure. Consider a typical simple example. At time $t = 0,1,\ldots$ risk reserve of an insurer is characterized as $R(x,t) = M(t) + xt - S(t)$, $t = 0,1,\ldots$, where $M(t)$ is the "normal" part of the reserve, associated with ordinary (noncatastrophic claims); a catastrophic scenario occurs at time t with probability p; $S(t) = \sum_{l=1}^{N_t} D_{t_l}$ is the accumulated catastrophic claim from catastrophes at random time moments $t_1, t_2, \ldots$; xt is the accumulated premium (inflow) from catastrophic risk. In more general risk processes inflows xt and outflows $S(t)$ at t are described by some random functions $I(x,t)$, $O(x,t)$ dependent on a vector x of feasible decisions [6]. The long term stability can be characterized by the prob-

ability of ruin $q(x) = \Pr[R(x,t) \le 0 \text{ for some } t]$ or $q(x) = EI_{R(x,\tau)\le 0}$, where $I_{R\le 0} = 1$ if $R \le 0$, $I_{R\le 0} = 0$ otherwise, and τ is the first moment t when $R(x,t) \le 0$. Assume $d(x)$ is the welfare generated by x. The calculation of $d(x)$ requires to consider all relevant agents [6]. The problem is to find $x \ge 0$ maximizing a trade-off between profit $d(x)$ and the risk $F(x) = d(x) + \gamma E[I_{R(x,\tau)\le 0}]$, where γ is a risk coefficient. The function $f(x,\omega) = d(x) + \gamma I_{R(x,\tau)\le 0}$ is an implicit function of x and ω, and it is also a discontinuous function. Assume that the probability $V_t(y) = \Pr[M(t) \le y]$ is an explicitly known function. By taking the conditional expectation for given $D_{t_1}, D_{t_2}, \dots$, the function $F(x)$ can be written as

$$F(x) = d(x) + \gamma E \sum_{t=1}^{\infty} p^{N_t}(1-p)^{t-N_t} \times V_t\left(\sum_{l=1}^{N_t} D_{t_l} - xt\right). \quad (17)$$

A SQG of $F(x)$ in (17) can be calculated [6] by using auxiliary random variables. At step k sample random variable $\varsigma_k \in \{1,2,\dots\}$ distributed according to arbitrary $\mu(t)$, $\sum_{t=1}^{\infty}\mu(t) = 1, \mu(t) > 0$, sample $D_{t_l}, l = 1,\dots,N_{\varsigma_k}$ and take $\xi(k) = d'(x^k) - \gamma p^{N_{\varsigma_k}}(1-p)^{1-N_{\varsigma_k}} V'_{\zeta_k}\left(\sum_{l=1}^{N_{\zeta_k}} D_{t_l} - x^k\zeta_k\right)\varsigma_k/\mu(\varsigma_k)$, where d', V'_t are the derivatives of $d(\cdot), V_t(\cdot)$. It is easy to see that $E[\xi(k)|x^k] = F_x(x^k)$. More general situations are discussed in [5].

Neural Nets

These models emerge in image processing, classification and behavioral sciences. From a formal point of view the training of a neural net is equivalent to the minimization of the error function $F(x) = \sum_{i=1}^{N} H(i,x)$, where each function $H(i,x)$ corresponds to one training object. At each step $k = 0,1,\dots,$ it is possible to experiment only with one object. Assume at the step k action $i = i(k)$ is chosen with probability $\mu(i) > 0$ among N alternatives. The most frequently used algorithm is so-called *back propagation*, where the current vector x^k of parameters x is adjusted [7] in the direction opposite to the gradient $\xi(k) = 1/\mu(i(k))H_x(i(k),x^k)$. It is easy to see that $E[\xi(k)|x^k] = F_x(x^k)$.

Automaton Learning Problem

Let $j = \{1,\dots,n\}$ be the automaton action set and $\beta(j)$ be the random response to action j. The distribution of $\beta(j)$ depends j but it is unknown. The automaton attempts to improve its behavior (current action) based on the random responses to a particular action chosen. In other words, the goal is to find an action with the largest expected outcome $E\beta(j)$. This problem can be formulated as a rather simple stochastic optimization problem: maximize $F(x) = \sum_{j=1}^{n} E\beta(j)x_j$, s.t. $\sum_{j=1}^{n} x_j = 1$, $x_j \ge 0$. Let $x^k = (x_1^k,\dots,x_n^k)$ is the current approximate solution to this problem. Choose an action $i = i(k)$ with the probability x_i^k, observe response $\beta^k(i(k))$ and calculate $\xi(k) = (0,\dots,0,\beta^k(i(k))/x_{i_k}^k,0,\dots,0)$. Then $E[\xi(k)|x^k] = F_x(x^k)$.

References

1. Ermoliev Y (1976) Methods of stochastic programming. Nauka, Moscow (in Russian)
2. Ermoliev Y, Gaivoronski A (1992) Stochastic quasigradient methods for optimization of discrete event systems. Annals Oper Res I39:1–39
3. Ermoliev Y, Norkin V (1997) On nonsmooth and discontinuous problems of stochastic systems optimization. Eur J Oper Res 101:(2):230–243
4. Ermoliev Y, Wets R (eds) (1988) Numerical techniques for stochastic optimization. Ser Comput Math, vol 10. Springer, Berlin
5. Ermoliev YM, Norkin VI (2004) Stochastic optimization of risk functions. In: Marti K, Ermoliev Y, Pflug G (eds) Dynamic stochastic optimization. Springer, Berlin, pp 225–249
6. Ermolieva T, Ermoliev Y (2005) Catastrophic risk management: flood and seismic risk case studies. In: Wallace SW, Ziemba WT (eds) Applications of stochastic programming. SIAM, MPS, pp 425–444
7. Gaivoronski A (1994) Convergence properties of backpropagation for neural nets via theory of stochastic gradient methods, Part 1. Optim Methods Softw 4:117–134
8. Gaivoronski A (2005) SQG: Software for solving stochastic programming problems with stochastic quasigradient methods. In: Wallace SW, Ziemba WT (eds) Applications of Stochastic Programming. SIAM, MPS, pp 38–60
9. Ho Y-C, Cao X-R (1996) Perturbation analysis of discrete event dynamic systems. Kluwer, Dordrecht
10. Kushner H, Clark D (1978) Stochastic approximation methods for constrained and unconstrained systems. Springer, Berlin
11. Pflug G (1996) Optimization of Stochastic Models: The interface between simulation and optimization. Kluwer, Boston

Stochastic Quasigradient Methods in Minimax Problems

YURI ERMOLIEV
IIASA, International Institute for Applied Systems Analysis, Laxenburg, Austria

MSC2000: 90C15

Article Outline

Keywords
Introduction
Decision Making Under Extreme Events
Assymetric Information
Convex–Concave Expectation
Stochastic Nash Equilibrium
Stochastic Optimization with Unknown Distributions
See also
References

Keywords

Stochastic minimax problem; Incomplete information; Decision making; Extreme events; Facility location; Stochastic nash equilibrium; Unknown distributions

Introduction

Stochastic quasigradient (SQG) methods are applicable to both deterministic and stochastic minimax (SMM) problems. SMM problems, which are similar to the two-stage stochastic programing, have nested nonsmooth sample (random) objective functions. Some examples of SMM applications are discussed in 'Stochastic quasigradient methods: Applications'. An important class of SMM problems takes on the form [4,9], pp. 165–168, [14]: minimize the expectation function

$$F(x) = E \max_{y \in Y} g(x, y, \omega), \quad x \in X, \tag{1}$$

where $f(x, \omega) = \max_{y \in Y} g(x, y, \omega)$ is the sample (random) objective function, $X \subseteq R^n$, $Y \subseteq R^r$ and ω is an element of a probability space (Ω, A, P), i. e. $\omega \in \Omega$, and A is the set of events (subsets of Ω) measurable with respect to the probability measure P. If Ω contains only one point, then the minimization of (1) corresponds to the standard deterministic minimax problem. Besides deterministic constraints of the type $x \in X$, problem (1) may have general constraints of the STO problems given in terms of expectation functions, some of which may have the same structure as the expectation function $F(x)$ in (1). The set Y may also depend on (x, ω) as in the two-stage stochastic programing. Functions $f(x, \omega)$ in (1) often have more general nested structure as in catastrophic risk management [7]. First of all, let us note that the sample function $f(\cdot, \omega)$ is an implicitly given nonsmooth function even for linear $g(\cdot, y, \omega)$. Therefore, all general purpose SQG methods developed for general nonsmooth problems (such as stochastic finite-difference approximations) are applicable to problem (1). Specific SQG methods utilize the structure of the sample function $f(x, \omega)$ by using the following idea. Let $y(x, \omega)$ be a solution of the inner maximization problem in (1), i. e. $f(x, \omega) = g(x, y(x, \omega), \omega)$. Often it is possible to show that $g(\cdot, y, \omega)$ is an analyticaly tractable **generalized differentiable (GD) function** and, hence, $f(\cdot, \omega) = \max_{y \in Y} g(\cdot, y, \omega)$ is also a generalized differentiable function and its subgradient

$$f_x(\cdot, \omega) = g_x(\cdot, y(x, \omega), \omega) \tag{2}$$

is an SQG of $F(x)$. For example, if $g(\cdot, y, \omega)$ is a convex function, then $f(\cdot, \omega)$ is also a convex function and (2) defines its subgradient. Let us note that although the two-stage model has similar objective function $f(x, \omega) = \min_{y \in Y} g(x, y, \omega)$, but in this case the convexity of $f(\cdot, \omega)$ requires much stronger assumptions: $g(x, y, \omega)$ has to be a convex function in both variables (x, y). As it is discussed further, these two classes of models are oriented on rather different decision situations under uncertainty. The vector $f_x(x, \omega)$ or its approximation can be used in various SQG methods. For example, if $f(\cdot, \omega)$ is a GD function, then the SQG projection method is defined as

$$x^{k+1} = \pi_X \left[x^k - \rho_k f_x(\overline{x}^k, \omega^k) \right], \quad k = 0, 1, \ldots, \tag{3}$$

where $\overline{x}^k = x^k + \varepsilon^k$ and $\varepsilon^0, \varepsilon^1, \ldots$ are independent random vectors with densities, $\|\varepsilon^k\| > 0$, $\varepsilon^k \to 0$ for $k \to \infty$ with probability 1, and $\omega^0, \omega^1, \ldots$ are independent samples of ω. The vector $f_x(\overline{x}^k, \omega^k)$ is a **stochastic mollifier quasigradient** of $F(x)$ at $x = x^k$. If $\rho_k \geq 0, \sum_{k=0}^{\infty} \rho_k = \infty$ with probability 1,

$\sum_{k=0}^{\infty} E\rho_k^2 < \infty$, and X is a convex compact, then $\{F(x^k)\}$ converges with probability 1 and the cluster points of random sequences $\{x^k\}$ belong with probability 1 to a connected set of local solutions [8,11]. The convergence of (3) for $\varepsilon^k \equiv 0$ takes place for so-called [4,9], p. 151, [15] weakly convex functions $F(x)$, i. e., functions such that $F(y) - F(x) \geq (F_x(x), y - x) + r(x, y)$, where $r(x, y)/\|x - y\| \to 0$, $x \to z$, $y \to z$. For convex $f(\cdot, \omega)$, the sequence $\{x^k\}$ converges with probability 1 to the set of optimal solutions for $\varepsilon^k \equiv 0$. If the probability distribution of ω is concentrated at one point, then (1) reduces to the standard deterministic minimax problem and (3) is a SQG procedure for this problem.

Example 1. Production planning under uncertainty. As long as there is uncertainty about future demand, prices, input-output coefficients, available resources, etc., the choice of a production level $\overline{x} \geq x \geq 0$ for foreseeable demand ω is a "hit-or-miss" type decision problem. The cost $f(x, \omega)$ associated with overestimation and underestimation of ω is, in the simplest case, a random piesewise linear function

$$f(x, \omega) = \max\{\alpha(x - \omega), \beta(\omega - x)\},$$

where α is the unit surplus cost and β is the unit shortage cost. The problem is to find the level x that is "optimal", in a sense, for all foreseeable demands ω rather than a function $x \to x(\omega)$ specifying the optimal production level should be in every possible "scenario" ω. The expected cost criterion leads to the minimization

$$F(x) = E \max\{\alpha(x - \omega), \beta(\omega - x)\} \tag{4}$$

subject to $\overline{x} \geq x \geq 0$ for a given upper bound $\overline{x}$. This stochastic minimax problem is also reformulated as a two-stage stochastic programming model known as the *newsboy problem*.

Function $F(x)$ in (4) is convex, therefore method (3) with $\varepsilon^k \equiv 0$ is reduced to the following

$$x^{k+1} = \min\{\max\{0, x^k - \rho_k \xi(k)\}, \overline{x}\}, \quad k = 0, 1, \ldots, \tag{5}$$

where $\xi(k) = \alpha$, if the current level of production x^k exceeds the observed demand $\omega^k (x^k \geq \omega^k)$ and $\xi(k) = -\beta$ otherwise. The method (5) can be viewed as an adaptive procedure which is able to learn the optimal production level through sequential adjustments of its current levels $x^0, x^1, \ldots$ to observable demands $\omega^0, \omega^1, \ldots$. Let us note that the optimal solution of (4) and more general SMM problems [6] defines quantile type characteristics of solutions, e. g., CVaR risk measures [16] (see also "Two-stage stochastic programming: Stochastic Quasigradient methods"). For example, if the distribution of ω has a density, $\alpha > 0$, $\beta > 0$, then the optimal solution x minimizing (4) is the quantile defined as $Pr[\omega \leq x] = \beta/(\alpha + \beta)$. Therefore, the process (5) is a constraint *sequential estimation procedure* as in [9]. Problem (4) illustrates the essential difference between so-called *scenario analysis* aiming at the straightforward calculation of $x(\omega)$ and the STO optimization approach: instead of producing trivial optimal "bang-bang" solutions $x(\omega) = \omega$ for each scenario ω (a Pareto optimal solution w.r.t. potential ω), an STO model as (4) produces one solution that is optimal ("robust") against all possible ω.

Example 2. Stochastic facility location model. This model [6,9], pp. 413–435, generalizes Example 1 and illustrates the possible implicit character of underlying probability distributions. Assume that customers living in a district i choose their destination j at random with probability P_{ij} related to the cost of travel between (i, j) and (or) other factors. Let ε_{ij} be a random number of customers traveling from i to j and τ_j is the total number of customers attracted by j: $\tau j = \sum_{i=1}^{m} \varepsilon_{ij}$, $j = 1{:}\, n$, $\sum_{j=1}^{n} \varepsilon_{ij} = a_i$, $i = 1{:}\, m$. The actual number τ_j of customers attracted by j may not be equal x_j. The random cost connected with overestimating or underestimating of the demand τ_j in district j may be a convex function $\alpha_j(x_j - \tau_j)$ for $x_j \geq \tau_j$ or $\beta_j(\tau_j - x_j)$ for $x_j < \tau_j$. The problem is to determine the size x_j that minimizes the expected cost

$$F(x) = \sum_{j=1}^{n} E \max\{\alpha_j(x_j - \tau_j), \beta_j(\tau_j - x_j)\},$$

where $\overline{x}_j \geq x \geq 0$. The SQG procedure for solving this problem is similar to (5). It is remarkable that applications of SQG methods to spatial minimax allocation problems [12] do not destroy their convexity in contrast to discrete approximation schemes.

Decision Making Under Extreme Events

The standard theory of extreme events studies the behaviors of the maxima, $\max(\theta_1, \ldots, \theta_n)$, for an iid sequence $\theta_1, \ldots, \theta_n$, $n \geq 2$. The objective function (1) is, in general, defined by the maxima of mutually dependent random variables $g(x, y, \omega)$, $y \in Y$, which also depends on decision variables x. Problem (1) can be viewed as a model for *decision making under extreme events*, when two types of uncertain variables y, ω affect the result $g(x, y, \omega)$ of decision x. The uncertainty with respect to y is evaluated from the extreme random scenario, whereas ω is considered as a random variable. Therefore (1) is a hybrid between a purely deterministic minimax approach that takes the form of minimizing $F(x) = \max\{g(x, y, \omega) | y \in Y, \omega \in \Omega\}$ and the purely probabilistic Bayesian approach of minimizing the expectation function $F(x) = Eg(x, y, \omega)$ for some joint probability distribution of (y, ω). Such SQG procedures as (3) can be viewed as an adaptive search of robust decisions by learning from environmental responses (simulations) $\omega^0, \omega^1, \ldots$.

Assymetric Information

The following interpretation leads to various important generalizations of the SMM problem (1). Consider two agents and the objective function $g(x, y, \omega)$. Agent 1 chooses his action $x \in X$ without knowing the choice $y \in Y$ of agent 2 and the state of nature (environment ω. Agent 2 chooses his action y after agent 1 and is fully informed about x, ω. $F(x)$ in (1) is the guaranteed expected result of agent 1 for action x. If agent 2 does not know the state ω before choosing his action y, the problem for agent 1 is to minimize

$$F(x) = \max_{y \in Y} Eg(x, y, \omega)\,. \tag{6}$$

The function $F(x)$ in (6) is nonlinear in probability measure P, i. e. it differes from the expected utility. The calculation of F at any point requires the solution of a STO subproblem, i. e. it is a **nested STO problem**, that often can be solved by using SQG methods with the **averaging operation**.

Example 3. Exact penalty function. See also ► stochastic quasigradient methods for a discussion of the method. For solving a particular case of problem (6): the minimization of the *exact penalty function*

$$F(x) = Ef_0(x, \omega) + C \sum_{i=1}^{m} \max\{0, Ef_i(x, \omega)\}$$

for general STO problems.

Convex–Concave Expectation

Assume that the function $\varphi(x, y) = Eg(x, y, \omega)$ is convex in x and concave in y, and X, Y are compact convex sets. Let

$$\begin{aligned} x^{k+1} &= \pi_X[x^k - \rho_k g_x(x^k, y^k, \omega^k)]\,, \\ y^{k+1} &= \pi_X[y^k + \rho_k g_y(x^k, y^k, \omega^k)]\,, \end{aligned} \tag{7}$$

where g_x,g_y are subgradients of $g(x,y,\omega)$ with respect to x,y correspondingly. This is an SQG projection method for the search of **saddle points** of $\varphi(x, y)$ in $X \times Y$. It is a stochastic version of the Arrow–Hurwicz method. The convergence of sequences $\{x^k\}, \{y^k\}$ to a saddle point requires rather strong assumptions on $\varphi(x, y)$, such as strict convex–concavity. It is possible to show that for linear functions $\varphi(\cdot, y)$, $\varphi(x, \cdot)$ the sequences $\{x^k\}, \{y^k\}$ do not converge under any choice of the step-size multiplier ρ_k, besides some exceptional cases. The convergence of the sequences $\sum_{s=0}^{k} \rho_s x^s \big/ \sum_{s=0}^{k} \rho_s$, $\sum_{s=0}^{k} \rho_s y^s \big/ \sum_{s=0}^{k} \rho_s$, $k \to \infty$, with probability 1 to a saddle point of $\varphi(x,y)$ takes place under standard assumptions on ρ_k ensuring the convergence of SQG projection method for convex problem [14,17]. Another possibility is to modify (7) by using general ideas of the proximal method or its variations [3,13].

Example 4. Finite set Y. Consider problem (6) with a finite set Y, i. e. assume that $F(x) = \max_{1 \leq i \leq r} Eg_i(x, \omega)$. This problem is equivalent to the minimization of $F(x) = \max_{y \in Y} E \sum_{i=1}^{r} y_i g_i(x, \omega)$, with convex–concave expectation, $Y = \{y_i \geq 0, \sum_{i=1}^{r} y_i = 1\}$.

Further refinements of stochastic minimax problems are possible. A hybrid of the models (1), (6) is the minimization

$$F(x) = \max_{y \in Y} E \max_{z \in Z} g(x, y, z, \omega)\,. \tag{8}$$

If $g(x, y, z, \omega)$ is convex in x and concave in y, and X, Y are convex compacts, then the procedure (7)

is also applicable for solving (8) with $g_x(x^k, y^k, \omega^k)$, $g^y(x^k, y^k, \omega^k)$ replaced by $g_x(x^k, y^k, z^k, \omega^k)$, $g^y(x^k, y^k, z^k, \omega^k)$, where z^k is a solution of the deterministic subproblem $g(x^k, y^k, z^k, \omega^k) = \max_{z \in Z} g(x^k, y^k, z, \omega^k)$.

Stochastic Nash Equilibrium

A stochastic equilibrium of an N person game on $X = X_1 \times X_2 \times \ldots \times X_N$ can be defined by using pay-off functions $\Psi_i(x) = E\psi_i(x, \omega)$, $x \in X$. Let us denote $(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$ by $(y_i|x)$. The point $x^* = (x_1^*, \ldots, x_N^*) \in X$ is refered to as the *Nash equilibrium* if $\Psi_i(x^*) = \min\{\Psi_i(y_i|x^*)|(y_i|x^*) \in X\}$. Let us introduce **KyFan function** $L(x, y) = \sum_{i=1}^N \left[\Psi_i(x) - \Psi_i(y_i|x)\right]$, $y = (y_1, \ldots, y_N)$, $x = (x_1, \ldots, x_N)$. A point $x^* \in X$ is a normalized equilibrium if $\max_{y \in X} L(x^*, y) = 0$. Since $\max_{y \in X} L(x, y) \geq 0$, the search of a stochastic normalized equilibrium reduces to an SMM problem: minimize

$$F(x) = \max_{y \in X} L(x, y), \quad x \in X\,.$$

The important additional information that $\min_{x \in X} F(x) = 0$ can be effectively utilized in the search of global solutions by SQG methods. Assume now that $L(x, y)$ is a convex–concave function for $x \in X$, $y \in X$. Then, the procedure (7) takes the form

$$x_i^{k+1} = \pi_X\left[x_i^k - \rho_k \psi_{ix_i}(x^k, \omega^k)\right], \quad i = 1\colon N, k = 0, 1, \ldots, \quad (9)$$

where ψ_{ixi} is a subgradient of $\varphi_i(x, \omega)$ with respect to x. The convergence conditions are similar to those of method (7). This method is an adaptive adjustment procedure for learning a Nash equilibrium [10,17].

Example 5. Stochastic Cournot oligopoly. The classical *oligopoly model* of Cournot [1,10] remains a key model within modern theories of industrial organization. Generalizing it to comprise the different goods and uncertainty, the model takes on the following form. Firm i produces the commodity bundle $x_i \in R^n$, thus incurring convex random production cost $c_i(x_i, \omega)$ and gaining random market revenues $(p(\sum_{j=1}^n x_j, \omega), x_i)$, where $p(Q, \omega)$ is the price at which total demand Q equals the aggregate supply $\sum_{j=1}^n x_j$. Suppose that $p(Q, \omega) = a - A(\omega)Q$, where $a \in R^n$ and $A(\omega)$ is a $n \times n$ positive semidefinite matrix (almost for all ω). Then for $\Psi_i(x, \omega) = c_i(x_i, \omega) - (a, x_i) + \sum_{j=1}^n (A(\omega)x_j, x_i)$ the function $L(x,y)$ is convex–concave.

Stochastic Optimization with Unknown Distributions

The probability measure P of the standard STO problem is assumed to be well defined on subsets A of Ω in the sense that it is possible to generate samples $\omega^0, \omega^1, \ldots$ of random variables ω from P. In practice the probability [2] measure P may not be known exactly: there is only information on some of its characteristics, in particular bounds for the mean value or other moments. Such information can often be written in terms of constraints

$$Q_s(x, P) = \int_\Omega q_k(x, \omega) P(\mathrm{d}\omega) \leq 0\,, \quad s = 1\colon K, \quad (10)$$

$$\int_\Omega P(\mathrm{d}\omega) = 1\,, \quad (11)$$

where $q_k(x, \omega)$ are known functions (which often do not depend on x), for example, as in constraints on joint moments $c_{r_1,\ldots,r_l} \leq E\omega_1^{r_1} \ldots \omega_l^{r_l} \leq C_{r_1,\ldots,r_l}$ with known constants c, C. If the unknown probability measure is evaluated from the worst case perspective within constraints (10), (11), then the STO problem is formalized as a SMM problem: find a vector x that minimizes

$$F(x) = \max_{P \in \mathcal{P}} \int f(x, \omega) P(\mathrm{d}\omega)\,, \quad (12)$$

where $\mathcal{P}$ is the family of probability measures defined by (10), (11). The solution of "inner" subproblem in (12) definesan implicit probability measure $P^*(x, \mathrm{d}\omega)$. The important fact is that $P^*(x,\cdot)$ is concentrated at not more than $K + 1$ points [4,5], and this fact can be utilized effectively in the design of solution procedures. Another important approach is to use the following duality relations [5]

$$F(x) = \max_{u \geq 0, \omega \in \Omega} \left[f(x, \omega) + \sum_{s=1}^K u_s q_s(x, \omega)\right] \quad (13)$$

for the inner maximization problem in (12).

Other important cases arise with further specification of uncertainties associated with measure P. Assume that random parameters can be separated

into two groups (ω, ν) with a joint distribution function $H(\omega,\nu)$ of the form $dH(\omega,\nu) = h(\omega,\nu) P(\mathrm{d}\omega)\mu(\mathrm{d}\nu)$, where $P(\mathrm{d}\omega)$ is not known exactly and satisfies (10), (11). If P is taken from the worst case, then the problem is formulated as the minimization of

$$F(x) = E\left[\max_{P\in\mathcal{P}} E_{x,y} f(x,\omega,\nu)\right] = \int\left[\max_{P\in\mathcal{P}}\int f(x,\omega,\nu)h(\omega,\nu)P(\mathrm{d}\omega)\right]\mathrm{d}\mu(\nu)\,. \tag{14}$$

By using duality relations similar to (13), this can be reformulated as the minimization of type (1) function

$$F(x) = \int \max_{u\geq 0,\omega\in\Omega}\Big[f(x,\omega,\nu)h(\omega,\nu) + \sum_{s=1}^{K} u_s q_s(x,\omega,\nu)\Big]\mathrm{d}\mu(\nu)\,. \tag{15}$$

Example 6. Incomplete information on cost functions. Consider (12) for cost function $F(x) = (Ec, x), x \in X$, where c is a random vector, $c = c(\omega,\nu)$, and the functions q_k in (10) do not depend on x. Then problem (15) is formulated as the minimization

$$F(x) = \int \max_{u\geq 0,\omega\in\Omega}\Big[(c(\omega,\nu),x)h(\omega,\nu) + \sum_{s=1}^{K} u_s q_s(\omega)\Big]\mu(\mathrm{d}\nu)\,.$$

Here $F(x)$ is a convex function. A SQG is defined by formula (2).

The complexity of SMM problems discussed here is due to nested structure of their objective functions. Many of them involve deterministic optimization subproblems under the sign of mathematical expectations. In applications, these subproblems often have a special structure, for example, the feasible set may be reduced to a finite number of alternatives (as in Example 4). In the case of infinite feasible sets they can be adaptively approximated by random finite sets with constant number of elements at each step $k = 0, 1, \ldots$ of SQG procedures, as it was proposed in [5]. For practical applications it is important to realize that models (1), (6), (8), (12), (14) are formulated, in fact, under different assumptions on the worst case scenario. For example, in (12), the evaluation of uncertainty is taken from the worst case expected outcomes, whereas (1) deals with the worst case random scenarios.

See also

- ▶ Minimax Theorems
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Stochastic Quasigradient Methods
- ▶ Stochastic Quasigradient Methods: Applications
- ▶ Two-Stage Stochastic Programming: Quasigradient Method
- ▶ Two-Stage Stochastic Programs with Recourse

References

1. Cournot A (1897) Researches into the mathematical principles of the theory of wealth. Macmillan, New York
2. Dupacova J (1987) The Minimax approach to stochastic programming and illustrative application. Stochastics 20:73–88
3. Eckstein J, Bertsekas DP (1992) On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math Program 55:293–318
4. Ermoliev YM (1976) Methods of stochastic programming. Nauka, Moscow
5. Ermoliev Y, Gaivoronski A, Nedeva C (1985) Stochastic optimization problems with incomplete Information on distribution functions. SIAM J Control Optim 23(5):697–716
6. Ermoliev Y, Leonardi G (1982) Some proposals for stochastic facility locations models. Math Model 3:407–420
7. Ermoliev YM, Norkin VI (2004) Stochastic optimization of risk functions. In: Marti K, Ermoliev Y, Pflug G (eds) Dynamic stochastic optimization. Springer, Berlin. 532 Lecture Notes in Economics and Mathematical Systems, pp 225–249
8. Ermoliev YM, Norkin VI (1998) Stochastic generalized gradient method with application to insurance risk management. Kibernetika I Sistemnii Analiz 2:50–71
9. Ermoliev Y, Wets R (eds) (1988) Numerical techniques for stochastic optimization. Springer Ser Comput Math 10:1–571
10. Flam SD (1999) Learning equilibrium play: a myopic approach. Comput Optim Appl 14:87–100
11. Gaivoronski A (2005) SQG: Software for solving stochastic programming problems with stochastic quasigradient methods. In: Wallace SW, Ziemba WT (eds) Applications of Stochastic Programming. SIAM, MPS, pp 38–60
12. Keyzer M, Ermoliev Y (1999) Modeling producer decisions in a spatial continuum. In: Herings P, Van der Laan G, Talman A (eds) The Theory of Markets. North-Holland, Amsterdam, Oxford

13. Martinet B (1970) Regularization d'inequations variationelles par approximations successives. Rev Francaise Inf Rech Oper 4:154–159
14. Nemirovskii AS, Yudin DB (1978) Cesaro convergence of the gradient method for approximating saddle points of convex–concave functions. Dokl Akad Nauk SSSR 239(5):1056–1059
15. Nurminskii E (1979) Numerical methods for solving deterministic and stochastic minimax problems. Naukova Dumka, Kiev
16. Rockafellar T, Uryasev S (2000) Optimization of conditional-value-at-risk. J Risk 2:21–41
17. Uryasev S (1990) Adaptive algorithms of stochastic optimization and game theory. Nauka, Moscow

Stochastic Scheduling

José Niño-Mora
Department of Statistics,
Universidad Carlos III de Madrid, Getafe, Spain

MSC2000: 90B36

Article Outline

Introduction
Models
Scheduling a Batch of Stochastic Jobs
Multi-Armed Bandits
Scheduling Queueing Systems
References

Introduction

The field of stochastic scheduling is motivated by problems of priority assignment arising in a variety of systems where jobs with random features (e. g., arrival or processing times) vie over time for access to shared service resources. Two prominent application areas are the dynamic scheduling of flexible manufacturing and computer-communication systems. Think, e. g., of a manufacturing workstation whose capacity is shared by multiple part types. Or consider a packet-switched communication network's channel whose bandwidth is shared by multiple traffic classes. Another rich set of applications is furnished by problems concerning the dynamic scheduling of multiple projects, whose states evolve randomly over time (e. g., research and development projects, or clinical trials).

The performance of such systems, as evaluated by a criterion such as the average time that jobs stay in the system (*flowtime*), can be significantly affected by the *scheduling policy* adopted to prioritize over time the access of jobs to resources. This motivates the interest of finding scheduling policies that optimize performance objectives of concern (e. g., minimizing the average flowtime). Yet, the high degree of discretionality allowed in the design of such policies gives rise to a combinatorial explosion rendering intractable an exhaustive search to determine an optimal policy. Instead, a goal of practical interest is to design relatively simple scheduling policies that achieve an optimal or nearly optimal performance.

The theory of stochastic scheduling addresses such a goal in the idealized setting of stochastic system models. Real-world random features such as job interarrival or processing times are thus modeled by specifying their probability distributions. Model assumptions vary across several dimensions, including the class of scheduling policies considered to be admissible, job interarrival and processing-time distributions, type and arrangement of service resources, and performance objective to be optimized. Typically, admissible policies are required to be nonanticipative, meaning that they cannot make use of future information, such as the unknown total duration of a job whose processing has not yet finished.

Regarding solution methods and techniques, it seems fair to say that no unified and practical approach is yet available to design and analyze optimal or near-optimal policies across the entire range of stochastic scheduling models. Although many such models can be cast in the framework of dynamic programming, straightforward application of this technique typically results in intractable formulations (*curse of dimensionality*). Classical results in the area were obtained through insightful ad hoc ideas, often based on interchange arguments (cf. [41]), whose extension to seemingly close model variations is hard or elusive. Yet the last two decades have witnessed major advances in promising research fronts, such as the use of Brownian or of fluid approximations, the use of mathematical programming formulations, and the development of priority-index policies.

Stochastic scheduling problems can be classified into three broad types, which have evolved with sub-

stantial autonomy: problems concerning the scheduling of a batch of stochastic jobs, multi-armed bandit problems, and problems concerning the scheduling of queueing systems.

The historical development of each such area has followed a similar three-stage pattern. In the first, earlier stage, researchers elucidated the optimal policies in relatively simple models. Such policies were often found to be based on priority indices: an index is computed for each job type or project, as a function of its state; then, at each decision epoch jobs or projects with larger index values are awarded higher priority for access to service. In the second stage, research efforts focused on identifying optimal policies in more complex models, often at the expense of introducing rather restrictive conditions on model parameters, such as symmetry assumptions. In the third, current stage, the main focus has shifted to develop computationally tractable methods capable of addressing large-scale models, which yield guidelines for designing good dynamic scheduling policies.

Models

Scheduling a Batch of Stochastic Jobs

In models of this class, a set of m machines is available to process a batch of n jobs with random processing times having known distributions, in order to optimize a given performance objective. The simplest such problem is to sequence a set of n stochastic jobs on a single machine ($m = 1$) to minimize the expected weighted flowtime. Job processing times are independent random variables, having a general distribution $G_i(\cdot)$ with mean p_i for job i. Admissible scheduling policies are required to be nonanticipative and *nonpreemptive* (processing of a job, once started, must proceed uninterruptedly to completion). Let $w_i \geq 0$ denote the cost rate incurred per unit time in the system (waiting or being processed) for job i, and let $\tilde{C}_i$ denote its random *completion time*. Let Π denote the class of all admissible policies, and let $E^{\pi}[\cdot]$ denote expectation under policy $\pi \in \Pi$. The problem can be formulated as

$$\min_{\pi \in \Pi} E^{\pi}\left[\sum_{j=1}^{n} w_j \tilde{C}_j\right]. \tag{1}$$

In the special case where job durations are deterministic, Smith first showed in [60] that it is optimal to sequence jobs in nonincreasing order of the priority index w_i/p_i. Such a rule is also optimal in the general stochastic case (1), as shown in [57]. References [36,37] identify conditions under which such an index rule is optimal when there are multiple identical parallel machines and processing times are exponentially distributed.

The model extension where policies are allowed to be *preemptive* (processing of a job may be interrupted at any time, to be later resumed) was solved by Sevcik in [58]. The optimal policy is again characterized by a priority index for each job, which in this case is a function of the cumulative processing time received so far.

Optimal index policies have also been identified for scheduling a batch of jobs on identical parallel machines, yet only under rather stringent conditions. The main performance objectives investigated in such a setting are: (i) minimize the total expected flowtime,

$$\min_{\pi \in \Pi} E^{\pi}\left[\sum_{j=1}^{n} \tilde{C}_j\right]; \tag{2}$$

and (ii) minimize the expected *makespan* (time to finish the last job),

$$\min_{\pi \in \Pi} E^{\pi}\left[\max_{1 \leq j \leq n} \tilde{C}_j\right]. \tag{3}$$

The index rule that assigns higher priority to jobs with shorter expected processing time (SEPT) has been shown to be optimal for (2) under the following assumptions: when job processing time distributions are exponential [15,29,72]; when jobs have the same general processing time distribution (having possibly received different amounts of processing prior to start) with a nondecreasing hazard rate function [65]; and, more generally, when job processing time distributions are stochastically ordered [67].

As for the expected makespan objective (3), the index rule that assigns higher priority to jobs with longer expected processing times (LEPT) has been shown to be optimal in the following cases: under exponential processing time distributions [15,72]; and when jobs have a common processing time distribution (with possibly different amounts of processing prior to start) with a nonincreasing hazard rate function [65].

Other models incorporate more complex features. Thus, the optimality of the preemptive version of Smith's index rule is extended in [53] to models with stochastic release dates or due dates. Also, in models with *uniform* parallel machines, which differ in speed rates, researchers have characterized optimal policies exhibiting a threshold structure: see [1,55] for the problem of expected flowtime minimization, and [18] for the problem of expected makespan minimization. An optimal policy for the problem of scheduling a batch of stochastic jobs in a *flow shop* (with m machines in series) is identified in [75].

The optimality of the simple policies identified in the work reviewed above typically does not extend to models that violate the required assumptions [19]. Finding an optimal policy in such cases appears to be a computationally intractable goal (see [50] for a study on the complexity of decision-making problems under uncertainty, such as stochastic scheduling). This fact has motivated the analysis of suboptimal heuristic index policies.

For example, it has been shown in [71] that, under mild assumptions, the suboptimality gap for Smith's rule, when used as a heuristic for stochastic scheduling on parallel machines, is bounded above by a quantity that is independent of the number of jobs. Therefore, as the latter grows to infinity the rule's relative suboptimality gap vanishes, yielding a form of asymptotic optimality. An earlier asymptotic optimality result in the same vein for a model of parallel machines stochastic scheduling with in-tree precedence constraints was obtained in [51].

A recent line of work uses optimal solutions to linear programming relaxations to design and analyze scheduling rules with performance guarantees for hard stochastic scheduling problems [40].

Multi-Armed Bandits

Models in this class are concerned with optimally allocating effort over time to a collection of projects, which change state in a random fashion depending on whether they are engaged or not. A classic example is the multi-armed bandit problem which, in its discrete-time version, can be described as follows: there is a collection of K projects labeled by $k = 1, \ldots, K$, exactly one of which must be engaged at each discrete time period $t = 0, 1, \ldots$. Project k can be in a finite number of states $i_k \in N_k$, where N_k is the project's state space. If at period t project k occupies state i_k and is engaged, an active reward $R_k^1(i_k)$ is earned, geometrically discounted by factor $0 < \beta < 1$; then, the project state changes in a Markovian fashion to j_k with active transition probability $p_k^1(i_k, j_k)$ for $j_k \in N_k$. Projects not engaged do not earn rewards (i. e., passive rewards are $R_k^0(i_k) \equiv 0$) and remain frozen. The problem is to find a nonanticipative scheduling policy for selecting the project to be engaged at each period, so as to maximize the total expected discounted reward earned over an infinite horizon. Denoting by Π the class of such admissible policies, and denoting by $X_k(t)$ and by $a_k(t)$ the state and the action ($a_k(t) = 1$: active; $a_k(t) = 0$: passive) for project k at period t, the problem can be formulated as

$$\max_{\pi \in \Pi} E^{\pi} \left[\sum_{t=0}^{\infty} \beta^t \sum_{k=1}^{K} R_k^{a_k(t)}\big(X_k(t)\big) \right].$$

Such a classic problem, whose name refers to a slot machine with multiple arms, one of which must be pulled at each time, has its origins in problems of sequential design of experiments [56,62]. After being long considered intractable, the problem was solved in a celebrated result by Gittins and Jones [28]. The optimal policy is given by an index rule: an index $\gamma_k(i_k)$ is defined for each project k as a function of its state i_k; then, at each time a project with currently largest index is engaged, breaking ties arbitrarily. The Gittins index generalizes that introduced in [7] for Bayesian Bernoulli bandits, which in turn was based on the index introduced in [13] for finite-horizon Bayesian bandits.

The optimality of such an index rule, for the original model and extensions, has a rich history of proofs yielding complementary insights. Such proofs are based on different techniques, including interchange arguments [26,28,64,70], dynamic programming [73], intuitive arguments [66], induction arguments [63], and conservation laws/linear programming [8]. See [27] for a comprehensive reference. For efficient methods to compute the Gittins index, see [17,35,46].

The important model extension where projects not engaged continue to evolve, typically with different transition probabilities, was introduced by Whittle in [74]. Its greatly improved modeling power comes,

however, at the cost of tractability [52]. In the setting of a time-average version of such a *multi-armed restless bandit problem*, he deployed a Lagrangian approach to obtain a heuristic index rule that reduces to Gittins' in the classic model. His conjecture regarding the asymptotic optimality of such an index rule as both the number of projects and the required number of projects to be engaged grow to infinity in a constant ratio was established, under appropriate conditions, in [68]. Yet his proposed index for restless bandits only exists for a restricted class of bandits, termed indexable, which raises the issue of finding sufficient conditions for indexability.

The results in [74] were based on introduction of a tractable problem relaxation, which also yields useful bounds on the optimal value. Improved bounds based on a hierarchy of linear programming relaxations were introduced in [11].

A framework for the analysis and computation of restless bandit indices, leading to the unifying concept of marginal productivity index, has been recently developed and deployed in several applications in [42,43,44,45]. See [47] for an accessible review of such a line of research.

The incorporation of penalties (costs or delays) for switching projects also yields an important yet intractable model extension of classic bandits [34], as it is no longer solved by index policies [5]. Yet, [3] introduced an intuitive index that partially characterizes optimal policies. An efficient algorithm to compute such an index, based on the natural formulation of classic bandits with switching costs as restless bandits without them, along with extensive computational experience showing that the resulting index policy is nearly optimal, is reported in [49].

Scheduling Queueing Systems

Models in this class concern the design of optimal policies for dynamic allocation of jobs to servers, where jobs arrive over time according to given stochastic processes The main class of models in this setting is that of *multiclass queueing networks* (MQNs), widely applied as versatile models of computer-communication and manufacturing systems.

The simplest types of MQNs involve scheduling a number of job classes in a single server. Similarly as in the two problem categories discussed above, simple priority-index rules have been shown to be optimal for a variety of such models. Consider the case of a multiclass $M/G/1$ queue, where K job classes vie for the attention of a single server: Jobs of class $k = 1, \ldots, K$ arrive at the system as a Poisson process with rate λ_k, and their service times are drawn independently from a common distribution $G_k(\cdot)$ with mean $1/\mu_k$. Class j jobs incur linear holding costs at rate $c_k \geq 0$ per unit time that a job resides in the system (waiting or in service). The goal is to find a nonanticipative and nonpreemptive scheduling policy prescribing which job class to serve at each decision epoch, in order to minimize the long-run average holding cost rate per unit time. Let Π denote the class of all such admissible policies, and let $E^{\pi}[L_k]$ denote the expected number of class k jobs in the system under policy $\pi \in \Pi$. The problem can be stated as

$$\min_{\pi \in \Pi} \sum_{k=1}^{K} c_k E^{\pi}[L_k] .$$

Its solution is given by the classic $c\mu$-rule [21], which is the same as the Smith index rule discussed above: award highest service priority at each time to a job with largest index $c_k \mu_k$. The $c\mu$-rule is also optimal among preemptive policies when service times are exponential.

The optimality of an index policy for the model extension that incorporates Markovian job feedback (when a class k job completes service it changes class to l with probability p_{kl}, and leaves the system with probability $1 - \sum_{l=1}^{K} p_{kl}$) was established by Klimov in [38]. The optimal priority index is efficiently computed by the K-step Klimov algorithm. The result was extended to the discounted criterion in [61].

An account of these results based on the *achievable region method*, which seeks to characterize the region of achievable performance vectors (e. g., mean queue lengths for each class) by means of linear programming constraints that formulate *conservation laws*, has been given in [8,20,25,59] (in increasing levels of generality). The performance of Klimov's index rule, when used as a heuristic in the model extension that includes identical parallel servers, has been analyzed using such an approach in [31]: a *relaxed* linear programming formulation of the performance region is shown to yield closed-form suboptimality bounds, which imply the rule's asymptotic optimality in heavy-traffic.

More general MQN models involve features such as changeover times for changing service from one job class to another [39], or multiple processing stations, which provide service to corresponding nonoverlapping subsets of job classes. Due to the intractability of such models, researchers have aimed to design relatively simple heuristic policies which achieve a performance close to optimal. The accomplishment of such a goal has been hindered by formidable technical challenges, including the *stability problem* for multiclass queueing networks with multiple stations [14,24]: in general it is not known what conditions on model parameters ensure that a given policy is *stable* (the time-average number of jobs in the system is finite). As a result, computer simulation remains the most widely used tool in applications of these models. Theoretical approaches currently under active development include the development of heuristic scheduling policies based on: diffusion approximations of the original system under heavy-traffic conditions [6,32,33,54,69]; fluid approximations [4,16,23]; mathematical programming formulations [9,10,12,22,30,31]; and restless bandit indexation [2,43,44,45,47,48].

References

1. Agrawala AK, Coffman Jr EG, Garey MR, Tripathi SK (1984) A stochastic optimization algorithm minimizing expected flow times on uniform processors. IEEE Trans Comput c-33:351–356
2. Ansell PS, Glazebrook KD, Niño-Mora J, O'Keeffe M (2003) Whittle's index policy for a multi-class queueing system with convex holding costs. Math Methods Oper Res 57:21–39
3. Asawa M, Teneketzis D (1996) Multi-armed bandits with switching penalties. IEEE Trans Autom Control 41:328–348
4. Atkins D, Chen H (1995) Performance evaluation and scheduling control of queueing networks: fluid model heuristics. Queueing Sys Theory Appl 21:391–413
5. Banks JS, Sundaram RK (1994) Switching costs and the Gittins index. Econometrica 62:687–694
6. Bell SL, and Williams RJ (2001) Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: asymptotic optimality of a threshold policy. Ann Appl Probab 11:608–649
7. Bellman R (1956) A problem in the sequential design of experiments. Sankhyā 16:221–229
8. Bertsimas D, Niño-Mora J (1996) Conservation laws, extended polymatroids and multiarmed bandit problems; a polyhedral approach to indexable systems. Math Oper Res 21:257–306
9. Bertsimas D, Niño-Mora J (1999) Optimization of multiclass queueing networks with changeover times via the achievable region approach: Part I, the single-station case. Math Oper Res 24:306–330
10. Bertsimas D, Niño-Mora J (1999) Optimization of multiclass queueing networks with changeover times via the achievable region approach: Part II, the multi-station case. Math Oper Res 24:331–361
11. Bertsimas D, Niño-Mora J (2000) Restless bandits, linear programming relaxations and a primal-dual index heuristic. Oper Res 48:80–90
12. Bertsimas D, Paschalidis IC, Tsitsiklis JN (1994) Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance. Ann Appl Probab 4:43–75
13. Bradt RN, Johnson SM, Karlin S (1956) On sequential designs for maximizing the sum of *n* observations. Ann Math Statist 27:1060–1074
14. Bramson M (1994) Instability of FIFO queueing networks. Ann Appl Probab 4:414–431
15. Bruno J, Downey P, Frederickson GN (1981) Sequencing tasks with exponential service times to minimize the expected flow time or makespan. J ACM 28:100–113
16. Chen H, Yao DD (1993) Dynamic scheduling of a multiclass fluid network. Oper Res 41:1104–1115
17. Chen YR, Katehakis MN (1986) Linear programming for finite state multi-armed bandit problems. Math Oper Res 11:180–183
18. Coffman Jr EG, Flatto L, Garey MR, Weber RR (1987) Minimizing expected makespans on uniform processor systems. Adv Appl Probab 19:177–201
19. Coffman Jr EG, Hofri M, Weiss G (1989) Scheduling stochastic jobs with a two-point distribution on two parallel machines. Probab Eng Inform Sci 3:89–116
20. Coffman Jr EG, Mitrani I (1980) A characterization of waiting time performance realizable by single server queues. Oper Res 28:810–821
21. Cox DR, Smith WL (1961) Queues. Methuen, London
22. Dacre M, Glazebrook K, Niño-Mora J (1999) The achievable region approach to the optimal control of stochastic systems (with discussion). J Royal Stat Soc Ser B Methodol 61:747–791
23. Dai JG, Lin W (2005) Maximum pressure policies in stochastic processing networks. Oper Res 53:197–218
24. Dai JG, Weiss G (1996) Stability and instability of fluid models for reentrant lines. Math Oper Res 21:115–134
25. Federgruen A, Groenevelt H (1988) Characterization and optimization of achievable performance in general queueing systems. Oper Res 36:733–741
26. Gittins JC (1979) Bandit processes and dynamic allocation indices (with discussion). J Royal Stat Soc Ser B Methodol 41:148–177
27. Gittins JC (1989) Multi-Armed Bandit Allocation Indices. Wiley, Chichester

28. Gittins JC, Jones DM (1974) A dynamic allocation index for the sequential design of experiments. In: Gani J, Sarkadi K, Vincze I (eds) Progress in Statistics (European Meeting of Statisticians, Budapest, 1972). North-Holland, Amsterdam, pp 241–266
29. Glazebrook KD (1979) Scheduling tasks with exponential service times on parallel processors. J Appl Probab 16: 685–689
30. Glazebrook KD, Niño-Mora J (1999) A linear programming approach to stability, optimisation and performance analysis for Markovian multiclass queueing networks. Ann Oper Res 92:1–18
31. Glazebrook KD, Niño-Mora J (2001) Parallel scheduling of multiclass *M*/*M*/*m* queues: approximate and heavy-traffic optimization of achievable performance. Oper Res 49:609–623
32. Harrison JM (1988) Brownian models of queueing networks with heterogeneous customer populations. In: Fleming W, Lions PL (eds) Stochastic Differential Systems, Stochastic Control Theory and Their Applications, vol 10. IMA Volumes in Mathematics and Its Applications. Springer, New York, pp 147–186
33. Harrison JM, Zeevi A (2004) Dynamic scheduling of a multiclass queue in the Halfin-Whitt heavy traffic regime. Oper Res 52:243–257
34. Jun T (2004) A survey on the bandit problem with switching costs. Economist 152:513–541
35. Kallenberg LCM (1986) A note on M. N. Katehakis' and Y.-R. Chen's computation of the Gittins index. Math Oper Res 11:184–186
36. Kämpke T (1987) On the optimality of static priority policies in stochastic scheduling on parallel machines. J Appl Probab 24:430–448
37. Kämpke T (1989) Optimal scheduling of jobs with exponential service times on identical parallel processors. Oper Res 37:126–133
38. Klimov GP (1974) Time-sharing service systems. I. Theory Probab Appl 19:532–551
39. Levy H, Sidi M (1990) Polling systems: applications, modeling, and optimization. IEEE Trans Commun 38:1750–1760
40. Möhring RH, Schulz AS, Uetz M (1999) Approximations in stochastic scheduling: The power of LP-based priority policies. J ACM 46:924–942
41. Nain P, Tsoucas P, Walrand J (1989) Interchange arguments in stochastic scheduling. J Appl Probab 27:815–826
42. Niño-Mora J (2001) Restless bandits, partial conservation laws and indexability. Adv Appl Probab 33:77–98
43. Niño-Mora J (2002) Dynamic allocation indices for restless projects and queueing admission control: a polyhedral approach. Math Program Ser A 93:361–413
44. Niño-Mora J (2006) Marginal productivity index policies for scheduling a multiclass delay-/loss-sensitive queue. Queueing Syst 54:281–312
45. Niño-Mora J (2006) Restless bandit marginal productivity indices, diminishing returns and optimal control of make-to-order/make-to-stock *M*/*G*/1 queues. Math Oper Res 31: 50–84
46. Niño-Mora J (2007) A (2/3)n^3 fast-pivoting algorithm for the Gittins index and optimal stopping of a Markov chain. INFORMS J Comput 19:596–606
47. Niño-Mora J (2007) Dynamic priority allocation via restless bandit marginal productivity indices (with discussion). Top 15:161–198
48. Niño-Mora J (2007) Marginal productivity index policies for admission control and routing to parallel multi-server loss queues with reneging. In: Network Control and Optimization: Proceedings of the First EuroFGI International Conference (NET-COOP 2007, Avignon, France), vol 4465. Lecture Notes in Computer Science. Springer, Berlin, pp 138–149
49. Niño-Mora J (2008) A faster index algorithm and a computational study for bandits with switching costs. INFORMS J Comput 20: in press
50. Papadimitriou CH (1985) Games against nature. J Comput Syst Sci 31:288–301
51. Papadimitriou CH, Tsitsiklis JN (1987) On stochastic scheduling with in-tree precedence constraints. SIAM J Comput 16:1–6
52. Papadimitriou CH, Tsitsiklis JN (1999) The complexity of optimal queuing network control. Math Oper Res 24: 293–305
53. Pinedo M (1983) Stochastic scheduling with release dates and due dates. Oper Res 31:559–572
54. Reiman MI, Wein LM (1998) Dynamic scheduling of a two-class queue with setups. Oper Res 4:532–547
55. Righter R (1988) Job scheduling to minimize expected weighted flowtime on uniform processors. Syst Control Lett 10:211–216
56. Robbins H (1952) Some aspects of the sequential design of experiments. Bull Am Math Soc 58:527–535
57. Rothkopf MH (1966) Scheduling with random service times. Manag Sci 12:707–713
58. Sevcik KC (1974) Scheduling for minimum total loss using service time distributions. J ACM 21:66–75
59. Shanthikumar JG, Yao DD (1992) Multiclass queueing systems: Polymatroidal structure and optimal scheduling control. Oper Res 40:S293–S299
60. Smith WE (1956) Various optimizers for single-stage production. Nav Res Logist Quart 3:59–66
61. Tcha DW, Pliska SR (1977) Optimal control of single-server queueing networks and multiclass *M*/*G*/1 queues with feedback. Oper Res 25:248–258
62. Thompson WR (1933) On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika 25:285–294
63. Tsitsiklis JN (1994) A short proof of the Gittins index theorem. Ann Appl Probab 4:194–199
64. Varaiya PP, Walrand JC, Buyukkoc C (1985) Extensions of the multiarmed bandit problem: the discounted case. IEEE Trans Autom Control 30:426–439
65. Weber RR (1992) Scheduling jobs with stochastic pro-

cessing requirements on parallel machines to minimize makespan or flowtime. J Appl Probab 19:167–182
66. Weber RR (1992) On the Gittins index for multiarmed bandits. Ann Appl Probab 2:1024–1033
67. Weber RR, Varaiya P, Walrand J (1986) Scheduling jobs with stochastically ordered processing times on parallel machines to minimize expected flowtime. J Appl Probab 23:841–847
68. Weber RR, Weiss G (1990) On an index policy for restless bandits. J Appl Probab 27:637–648
69. Wein LM (1992) Dynamic scheduling of a multiclass make-to-stock queue. Oper Res 40:724–735
70. Weiss G (1988) Branching bandit processes. Probab Eng Inf Sci 2:269–278
71. Weiss G (1992) Turnpike optimality of Smith's rule in parallel machines stochastic scheduling. Math Oper Res 17:255–270
72. Weiss G, Pinedo M (1980) Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. J Appl Probab 17:197–202
73. Whittle P (1980) Multiarmed bandits and the Gittins index. J Royal Stat Soc Ser B Method 42:143–149
74. Whittle P (1988) Restless bandits: Activity allocation in a changing world. In: Gani J (ed) A celebration of applied probability, spec vol 25A. Appl Probab Trust, Sheffield, pp 287–298
75. Wie SH, Pinedo M (1986) On minimizing the expected makespan and flow time in stochastic flow shops with blocking. Math Oper Res 11:336–342

Stochastic Transportation and Location Problems

KAJ HOLMBERG
Department Math.,
Linköping Institute Technol.,
Linköping, Sweden

MSC2000: 90B80, 90C11

Article Outline

Keywords
The Problem
Solution Method
How to Solve the Subproblem
Computational Results
See also
References

Keywords

Location; Transportation; Fixed charge; Branch and bound

The basic *transportation problem* (a minimal cost network flow problem in a bipartite graph) is a very well-known problem, which can be efficiently solved with existing methods. It is also an important problem in practice; transporting goods from a set of supply points (factories) to a set of demand points (customers) so as to minimize transportation costs is a situation that often faces planners.

However, in practice the demand of the customers is often not known exactly. In many cases it is best seen as a stochastic amount, with a certain probability function and a certain expected value. Models of this situation also exist in the literature, and quite efficient solution methods have been developed, see for example [1,10] and [9]. The problem is called the *stochastic transportation problem*, (STP), and is a transportation problem with the demand constraints replaced by nonlinear convex costs.

Considering the other end of the transportation problem, there are *facility location* models, which deal with the question of whether or not a certain supply point should be utilized. In such problems, a supply point (facility) is available only if a certain fixed cost is paid. Such models, with linear costs for transportation, are also well known andseveral efficient solution methods exist, see for example [3,6,17], and [21]. (Other variants of location problems are treated in ► Facility location with staircase costs and ► Facility location problems with spatial interaction.)

Obviously both these aspects can be interesting to consider simultaneously, i. e. planning the location of supply facilities and transportation of goods to the customers, while considering the demand as stochastic.This is what we call the *stochastic transportation and location problem*. This problem has received little attention until now. Only a few suggestions for solution methods can be found in the literature, [2,11,12,14]. The latter two papers actually consider a further generalization with generalconcave costs at the supply points, together with the convex costs at thedemand points.

The Problem

Let m supply points (facilities) and n demand points (customers) be given. The variables are x_{ij}, the amount shipped from supply point i to demand point j, z_i, the amount shipped out of supply point i, and y_j, the amount shipped into demand point j. The maximal capacity at supply point i is S_i, and the cost of production is $g_i(z_i)$, where $g_i(0) = 0$, and $g_i(z_i)$ is assumed to be lower semicontinuous, nondecreasing and concave due to economies of scale. Assuming stochastic at demand point j, d_j, we get a penalty cost $\varsigma_j(y_j, d_j)$, such that the expected penalty $f_j(y_j) = \mathsf{E}[\varsigma_j(y_j, d_j)]$ is a convex function, see [1].

Let us assume a probability density function, $\phi_j(d_j)$, which gives an expected demand as $\mathsf{E}[d_j] = \int_0^\infty v\,\phi_j(v)\,dv$, and a continuous distribution function as $F_j(d_j) = \int_0^{d_j} \phi_j(v)dv$. There are unit holding costs, $\alpha_j > 0$, and unit shortage costs, $\xi_j > 0$, which gives a total cost as

$$\begin{aligned} f_j(y_j) &= \xi_j \int_{y_j}^{\infty} (v - y_j)\phi_j(v)\,\mathrm{d}v \\ &\quad + \alpha_j \int_0^{y_j} (y_j - v)\phi_j(v)\,\mathrm{d}v \\ &= \xi_j(\mathsf{E}[d_j] - y_j) + (\xi_j + \alpha_j) \int_0^{y_j} F_j(v)\,\mathrm{d}v\,, \end{aligned}$$

which can be shown to be a convex function.

The costs for transportation are linear, with unit cost c_{ij} from supply point i to demand point j. The problem to minimize the total costs is:

$$(P)\begin{cases} v^* = \min \sum_{i=1}^m \sum_{j=1}^n c_{ij}x_{ij} \\ \qquad + \sum_{i=1}^m g_i(z_i) + \sum_{j=1}^n f_j(y_j) \\ \text{s.t.} \quad \sum_{i=1}^m x_{ij} = y_j, \quad j = 1,\dots,n, \\ \qquad \sum_{j=1}^n x_{ij} = z_i, \quad i = 1,\dots,m, \\ \qquad 0 \le z_i \le S_i, \quad i = 1,\dots,m, \\ \qquad x_{ij} \ge 0, \quad \forall i, j. \end{cases}$$

The objective function is a sum of three terms: a linear transportation cost, a convex shortage penalty and a concave production cost. It isneither convex nor concave, but a *d.c. function*, i. e. a function that can be represented as a difference of two convex functions, [13]. Minimizing such a function under linear constraints is a nonconvex global optimization problem, which may have multiple local minima.

We can add the following redundant constraints,

$$\begin{cases} \sum_{j=1}^n y_j \le S_{\mathrm{TOT}}, \\ y_j \ge 0, \quad \forall j, \\ x_{ij} \le S_i, \quad \forall i, \end{cases}$$

where $S_{\mathrm{TOT}} = \sum_{i=1}^m S_i$, to ensure that the feasibleset is bounded.

The problem (P) is quite general, and here we are mainly interested in the stochastic transportation-location problem, (STLP), which is the special case of (P) that occurs if g_i consists of a fixed charge (and possibly a linear cost). This problem can also be formulated as

$$(STLP)\begin{cases} v^* = \min \sum_{i=1}^m \sum_{j=1}^n c_{ij}x_{ij} \\ \qquad + \sum_{i=1}^m r_i\delta_i + \sum_{j=1}^n f_j(y_j) \\ \text{s.t.} \quad \sum_{i=1}^m x_{ij} = y_j, \\ \qquad\qquad j = 1,\dots,n, \\ \qquad \sum_{j=1}^n x_{ij} \le S_i\delta_i, \\ \qquad\qquad i = 1,\dots,m, \\ \qquad \delta_i \in \{0,1\}, \\ \qquad\qquad i = 1,\dots,m, \\ \qquad y_j \ge 0, \quad \forall j, \\ \qquad x_{ij} \ge 0, \quad \forall i, j, \end{cases}$$

where r_i is the fixed cost for starting production at supply point i, and δ_i is equal to 1 if something is produced

at supply point i and 0 if not. This problem has been solved by a heuristic approach in [14] and by generalized Benders decomposition, [5], in [2].

A much simpler special case of (P) occurs if $g_i(z_i)$ is linear (and thus can be included in the transportation costs), namely the *stochastic transportation problem*, see [15]:

$$(STP)\quad \begin{cases} v^* = \min \sum_{i=1}^{m}\sum_{j=1}^{n} \bar{c}_{ij}x_{ij} \\ \qquad + \sum_{j=1}^{n} f_j(y_j) \\ \text{s.t.} \quad \sum_{i=1}^{m} x_{ij} = y_j, \\ \qquad\qquad j = 1,\dots,n, \\ \sum_{j=1}^{n} x_{ij} \le S_i, \\ \qquad\qquad i = 1,\dots,m, \\ x_{ij} \ge 0, \quad \forall i,j, \\ y_j \ge 0, \quad \forall j. \end{cases}$$

The objective function of (STEP) is convex, so the problem can be solved efficiently by methods for convex problems.

Solution Method

We will here describe the method proposed in [12], which solves both (P) and (STLP). It exploits the facts that the constraints are of transportation type and the objective function is separable. Furthermore, of the $mn + m + n$ variables, only m variables, z, appear in nonconvex functions. Therefore one can reduce the problem to a much smaller d.c. optimization problem in only z. This reduced problem can be solved by a branch and bound procedure in which branching is performed by partitioning the space into rectangles, while bounding is based on linearization of the concave functions $g(z) = \sum_{i=1}^{m} g_i(z_i)$.

One can show, [12], that(P) is equivalent to

$$(P^*)\quad \min\{\varphi(z) + g(z):\ z \in \Omega\}$$

in the sense that the two problems have equal optimal values and if (x^*, y^*, z^*) solves (P), then z^* solves (P*), and conversely if z^* solves (P*), then (x^*, y^*, z^*) solves (P), where (x^*, y^*) is an optimal solution of (Q(z)), where

$$(Q(z))\quad \begin{cases} \varphi(z) = \min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \qquad + \sum_{j=1}^{n} f_j(y_j) \\ \text{s.t.} \quad \sum_{i=1}^{m} x_{ij} = y_j, \\ \qquad\qquad j = 1,\dots,n, \\ \sum_{j=1}^{n} x_{ij} = z_i, \\ \qquad\qquad i = 1,\dots,m, \\ y_j \ge 0, \\ \qquad\qquad j = 1,\dots,n, \\ x_{ij} \ge 0, \quad \forall i,j \end{cases}$$

for a given z in the rectangle $\Omega = \{z: 0 \le z \le S := (S_1, \dots, S_m)\}$. One can show that $\varphi(z)$ is a convex function.

(P*) is still a d.c. optimization problem, but much smaller than (P). Below we present a branch and bound solution method for (P*), based on the following.

- Rectangular subdivision of the feasible domain Ω.
- Linearization of the concave costs, yielding a subproblem similar to (STP).

Let $M = [p, q]$ be a rectangle contained in Ω. Any point $w \in M$, together with an index $i \in \{1, \dots, m\}$, determines a subdivision of M into two subrectangles $\{z: p_i \le z_i \le w_i, p_t \le z_t \le q_t\ (\forall t \ne i)\}$ and $\{z: w_i \le z_i \le q_i, p_t \le z_t \le q_t\ (\forall t \ne i)\}$. This subdivision is called a *subdivision via* (w, i).

For any rectangle $M = [p, q]$ contained in the feasible domain Ω, let $L_{M,i}(\cdot)$ be the affine function which agrees with $g_i(\cdot)$ at the endpoints of the segment $[p_i, q_i]$, i. e.

$$L_{M,i}(z_i) = g_i(p_i) - \left(\frac{p_i}{q_i - p_i}\right)\left(g_i(q_i) - g_i(p_i)\right) + z_i\left(\frac{g_i(q_i) - g_i(p_i)}{q_i - p_i}\right).$$

Define the convex problem

$$(CP(M))\begin{cases}\beta(M) = \min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \qquad + \sum_{j=1}^{n} f_j(y_j) \\ \qquad + \sum_{i=1}^{m} L_{M,i}(z_i) \\ \text{s.t.} \quad \sum_{i=1}^{m} x_{ij} = y_j, \\ \qquad j = 1,\dots,n, \\ \sum_{j=1}^{n} x_{ij} = z_i, \\ \qquad i = 1,\dots,m, \\ y_j \geq 0, \quad \forall j, \\ x_{ij} \geq 0, \quad \forall i,j, \\ z \in M.\end{cases}$$

Clearly $\beta(M) \leq \min\{\varphi(z) + g(z): z \in M\}$, and if an optimal solution $(\overline{x}(M); \overline{y}(M), \overline{z}(M))$ of (CP(M)) satisfies $L_M(\overline{z}(M)) = g(\overline{z}(M))$ then $\beta(M) = \min\{\varphi(z) + g(z): z \in M\}$.

Since the convex problem (CP(M)) is an STP with additional constraints $z \in M$, i. e. $p \leq z \leq q$, the lower bounding subproblem for each rectangle can be solved by slightly modified versions of algorithms for (STP), see [9].

M_k, the subrectangle in which the approximation is to be improved, is chosen to be the subrectangle with smallest lower bound.

$$M_k \in \arg\min_{M \in W_k'} \beta(M),$$

where W_k' denotes the current partition. This implies $\beta(M_k) \leq \min\{\varphi(z) + g(z): z \in \Omega\}$, and if $\overline{z}^k = \overline{z}(M_k)$ satisfies $L_{M_k}(\overline{z}^k) = g(\overline{z}^k)$ then equality holds, i. e. $\overline{z}^k$ solves (P*). Otherwise, $g_i(\overline{z}_i^k) - L_{M_k i}(\overline{z}_i^k) > 0$ for at least one i and we subdivide M via $(\overline{z}^k, i_k)$, where i_k is the index i that achieves the maximal difference $g_i(\overline{z}_i^k) - L_{M_k,i}(\overline{z}_i^k)$ between the actual cost and the linear approximation. This subdivision rule ensures that eventually this maximal difference will tend to zero, and the lower bound $\beta(M_k)$ will tend to the exact minimum of the objective function on M_k.

The method is given in algorithmic form:

Initialization
Choose ε and a subrectangle M_1 of Ω which is known to contain an optional solution of (P*). Let z^1 be the best feasible point available, and $\bar{\nu} = \varphi(z^1) + g(z^1)$. Set $W_1 = P_1 = \{M_1\}$, $k = 1$.

Iteration $k = 1, 2, \dots$

i For each $M \in P_k$ solve (CP(M)), yielding $\beta(M)$ and $(\bar{x}(M); \bar{y}(M), \bar{z}(M))$. Update $\bar{\nu}$ and z^k.

ii Delete all $M \in W_k$ such that $\beta(M) \geq \bar{\tau} - \varepsilon$, and let W_k' be the remaining members of W_k.
IF $W_k' = \emptyset$ THEN terminate: z^k is a global ε-optimal solution of (P*). ELSE choose $M_k \in \arg\min\{\beta(M) : M \in W_k'\}$.

iii Let $\bar{z}^k = \bar{z}(M_k)$. Select $i_k \in \arg\max_i \{g_i(\bar{z}_i^k) - L_{M_k,i}(\bar{z}^k)\}$.
IF $g_{i_k}(\bar{z}_{i_k}^k) - L_{M_k,i_k}(\bar{z}^k) = 0$
THEN terminate: $\bar{z}^k$ is a global optional solution.
ELSE divide M_k via $(\bar{z}^k, i_k)$. Let P_{k+1} be the partition of M_k and $W_{k+1} = (W_k' \setminus \{M_k\}) \cup P_{k+1}$.
Set $k \to k+1$ and go back to i).

In [12] convergence of this algorithm is proved. If $g_i(t)$ is discontinuous at $t = 0$, then the problem does not change if $g_i(t)$ is replaced by a continuous function $\widetilde{g}_i(t)$ that is linear for $0 \leq t \leq \tau_i$ and equal to $g_i(t)$ for $t \geq \tau_i$, where τ_i is a certain positive number (for details, see [12]).

Let us now discuss the case when $g_i(t)$ are concave piecewise linear nondecreasing functions. First we can, as mentioned above, assume that they are continuous at $t = 0$, hence continuous at every point. Furthermore, in Step iii) of the Algorithm, instead of dividing M_{k,i_k} via $\overline{z}_{i_k}^k$ we can divide it via the breakpoint u_{k,i_k} of $g_{i_k}(t)$ nearest to $\overline{z}_{i_k}^k$ that satisfies

$$g_{i_k}(u_{k,i_k}) - L_{M_k,i_k}(u_{k,i_k}) \geq g_{i_k}(\overline{z}_{i_k}^k) - L_{M_k,i_k}(\overline{z}_{i_k}^k).$$

Since the number of breakpoints of each function $g_i(t)$ is finite, the algorithm must terminate after finitely many steps. In this case the method has similarities to the method proposed in [19].

If $g_i(t)$ is of fixed charge type, e. g. $g_i(0) = 0$ and $g_i(t) = r_i + \rho_i t > 0$ for $t > 0$, it can be replaced by a continuous

concave piecewise linear nondecreasing function with one breakpoint u_i arbitrarily near to 0. The subdivision of the interval $[0, S_i]$ via u_i then amounts to branching over the boolean variable δ_i, with $[0, u_i]$ corresponding to $\delta_i = 0$, and $[u_i, S_i]$ corresponding to $\delta_i = 1$.

How to Solve the Subproblem

There are several possibilities for solving the subproblem, (CP(M)), which is a problem of the type (STP). It can be solved by the Frank–Wolfe method (cf. also ► Frank–Wolfe algorithm, [4]), in [1] and [16], by cross decomposition, [20], in [10], by separable programming in [7], by the forest iteration method in [18] and by mean value cross decomposition, [9], and combinations of separable programming, Lagrangian relaxation with subgradient optimization and mean value cross decomposition in [9].

In computational tests in [9] separable programming combined with mean value cross decomposition, here denoted by SM, is found to be the quickest method, followed by a modified version of the Frank–Wolfe method, [16], here denoted by FW.

In [12] two branch and bound methods, BB-SM where the subproblem is solved by the SM method, and BB-FW, where the subproblem is solved by the FW method, are compared. BB-FW is found to be rather stable, while BB-SM is less so. Reasons for this can be the scaling and rounding required in the primal subproblem in SM, if the linear minimal cost network flow code used requires integer costs and capacities. If after branching the interval $[p_i, q_i]$ is small, the rounding may have large effects on the dual solution (used as input to subsequent subproblems).

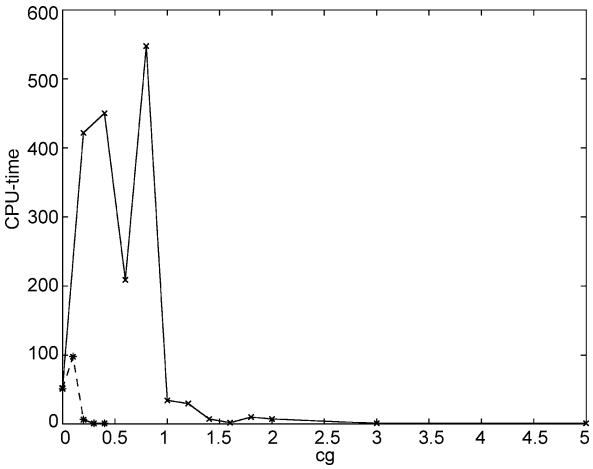
Computational Results

In [12] a number of randomly generated problems with up to 100 origins and 500 destinations have been solved. The probability density functions used are exponential distribution functions, which yields

$$f_j(y_j) = \alpha_j \left(y_j - \frac{1}{\lambda_j} \right) + \left(\frac{\xi_j + \alpha_j}{\lambda_j} \right) \exp(-\lambda_j y_j).$$

The concave cost functions are chosen to be of the form

$$g_i(z_i) = \begin{cases} b_i z_i^{a_i} & \text{if } z_i > 0, \\ 0 & \text{if } z_i = 0. \end{cases}$$



Stochastic Transportation and Location Problems, Figure 1

The relations between the convex costs and the concave costs are decided by a weight, c_g, on the concave part, which seems to affect the difficulty of the problem very much. For small values of c_g the concave part of the cost is dominated by the convex part, and the problems are easy. (For $c_g = 0$, we get the (STP).) For large values of c_g, the concave part dominates, and these problems are also easy (solvable in fractions of a second). However, somewhere between these extremes, we find a sharp increase in difficulty. These effects are illustrated for two groups of problems in Fig. 1.

A general conclusion of the computational tests is that the branch and bound trees are fairly limited in size, in spite of the risk for unnecessary branching due to the asymptotic convergence of the methods used for the subproblem. This is indicates that the bounding subproblems are strong enough.

See also

- ► Combinatorial Optimization Algorithms in Resource Allocation Problems
- ► Competitive Facility Location
- ► Facility Location with Externalities
- ► Facility Location Problems with Spatial Interaction
- ► Facility Location with Staircase Costs
- ► Global Optimization in Weber's Problem with Attraction and Repulsion
- ► Minimum Concave Transportation Problems
- ► MINLP: Application in Facility Location-allocation
- ► Motzkin Transposition Theorem

- ▶ Multifacility and Restricted Location Problems
- ▶ Multi-index Transportation Problems
- ▶ Network Location: Covering Problems
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Production-distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Cooper L, LeBlanc LJ (1977) Stochastic transportation problems and other network related convex problems. Naval Res Logist Quart 24:327–336
2. Franca PM, Luna HP (1982) Solving stochastic transportation-location problems by generalized Benders decomposition. Transport Sci 16(2):113–126
3. Francis RL, McGinnis LF, White JA (1983) Locational analysis. Europ J Oper Res 12:220–252
4. Frank M, Wolfe P (1956) An algorithm for quadratic programming. Naval Res Logist Quart 3:95–110
5. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
6. Geoffrion A, McBride R (1978) Lagrangean relaxation applied to capacitated facility location problems. AIIE Trans 10(1):40–47
7. Holmberg K (1984) Separable programming applied to the stochastic transportation problem. Res Report Dept Math Linköping Inst Techn no. LiTH-MAT-R-1984-15
8. Holmberg K (1992) Linear mean value cross decomposition: A generalization of the Kornai–Liptak method. Europ J Oper Res 62:55–73
9. Holmberg K (1995) Efficient decomposition and linearization methods for the stochastic transportation problem. Comput Optim Appl 4:293–316
10. Holmberg K, Jörnsten K (1984) Cross decomposition applied to the stochastic transportation problem. Europ J Oper Res 17:361–368
11. Holmberg K, Tuy H (1994) A production-transportation problem with stochastic demand and concave production costs. In: Bachem A, Derigs U, Jünger M, Schrader R (eds) Operations Research '93. GMÖOR, pp 266–269
12. Holmberg K, Tuy H (1999) A production-transportation problem with stochastic demand and concave production costs. Math Program 85:157–179
13. Horst R, Tuy H (1993) Global optimization, 2nd edn. Springer, Berlin
14. LeBlanc LJ (1977) A heuristic approach for large scale discrete stochastic transportation-location problems. Comput Math Appl 3:87–94
15. LeBlanc LJ, Cooper L (1974) The transportation-production problem. Transport Sci 8:344–354
16. LeBlanc LJ, Helgason RV, Boyce DE (1985) Improved efficiency of the Frank–Wolfe algorithm for convex network programs. Transport Sci 19:445–462
17. Nauss RM (1978) An improved algorithm for the capacitated facility location problem. J Oper Soc 29:1195–1201
18. Qi L (1985) Forest iteration method for stochastic transportation problem. Math Program Stud 25:142–163
19. Rech P, Barton LG (1970) A non-convex transportation algorithm. In: Beale EM (ed) Applications of Mathematical Programming Techniques, pp 250–260
20. Van Roy TJ (1983) Cross decomposition for mixed integer programming. Math Program 25:46–63
21. Van Roy TJ (1986) A cross decomposition algorithm for capacitated facility location. Oper Res 34:145–163

Stochastic Vehicle Routing Problems

SVRP

Francois Louveaux[1], Gilbert Laporte[2]

[1] University Namur, Namur, Belgium

[2] HEC Montreal, Montreal, Canada

MSC2000: 90C15, 90C10

Article Outline

Keywords
See also
References

Keywords

Vehicle routing; Stochastic programming; Chance constraint programming

The deterministic *vehicle routing problem* (VRP) is defined on a graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is a vertex set and $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ is an edge set. Vertex v_1 represents a depot at which are based m identical vehicles of capacity D. The other vertices are customers requiring a visit, which may consist of either

collection or delivery of goods or of providing some service like in the repair industry. The VRP consists of designing a set of m least cost routes starting and ending at the depot, such that each customer is visited exactly once. In practice, additional constraints need to be satisfied. Two important examples are the following:

- capacity constraints: with each customer v_i is associated a demand d_i. Then the total demand of a vehicle route may not exceed D.
- time constraints: with each customer v_i is associated a service time s_i. Also, with each edge (v_i, v_j) is associated a travel time t_{ij}. Then, the total duration of each route, including service and travel times, may not exceed a given bound B.

Several other constraints can be encountered in practical applications. For a recent survey and a bibliography on the deterministic VRP, see [5] and [10].

In many situations, some components of the problem are random. Then, the problem becomes a *stochastic vehicle routing problem* (SVRP). At the moment (1999), three main cases have been considered:

1) stochastic customers: customer v_i is present with probability p_i and absent with probability $1 - p_i$.
2) stochastic demands: the demand d_i of customer v_i is a random variable.
3) stochastic times: the service time s_i of customer v_i and the travel time t_{ij} of edge (v_i, v_j) are random variables.

This randomness may apply to only some customers or edges. When some data are random, it is no longer possible to require that all constraints be satisfied for all realizations of the random variables. As is classical in stochastic programming (see also [3]), two main approaches are considered.

In *chance constraint programming* (CCP), the decision maker requires that the constraints must be satisfied with a given probability, typically 90% or 95%. This line of research for the SVRP was initiated in [18]. While in general, the approach in CCP is to obtain a (usually nonlinear) deterministic equivalent of the probabilistic constraint, it turns out that, for the SVRP, linear constraints can be found to eliminate routes that violate the probabilistic constraints. Those constraints are stronger for the SVRP with stochastic demands (as they apply to sets of customers) and, in general, weaker in the case of stochastic travel times (as they only apply to the routes traveled). These constraints are usually embedded within a branch and cut (cf. also ► Integer programming: Branch and cut algorithms) procedure, in exactly the same way as this is done for the subtour elimination constraints in the deterministic VRP [13].

In *stochastic program with recourse* (SPR), the set of decisions is divided into two groups, the de- cisions made before the realizations of the random variables are known are called *first-stage decisions*, those made after the realizations are known are called *second-stage* (or *recourse*, or *corrective*) actions. In the SVRP, the first-stage decision typically consists of planning the various routes. In the second stage, the routes are followed as planned, with simple rules for possible corrective actions. When customers are present with some probability, the recourse action consists of skipping absent customers (this problem is then known as the *probabilistic traveling salesman problem* , or PTSP). For the case where demands are random, while following the planned route to make the collections, the vehicle may be unable to load some customer's demand as the vehicle becomes full. The recourse action may then consist of a return trip to the depot to unload, such that the vehicle may be able to resume its trip. Similarly in the SVRP with *stochastic travel times*, the recourse action may simply consist of paying some charge (or penalty) when the effective travel time exceeds B. Such situations do not occur in a deterministic setting where demands (or travel times) are supposed to take precisely the value forecasted for planning the route. The aim of an SPR is to find a solution of least expected total length (for the PTSP) or least expected total cost (for SVRP with stochastic demands or times). The framework of planning the routes in the first-stage and having a simplified recourse policy in the second-stage is known as *a priori optimization*.

Solution methodologies include the asymptotic analysis of a priori optimization [2,9], heuristics such as the modified savings algorithm [4], metaheuristics such as the tabu search [7] and exact algorithms [12]. Based on the integer L-shaped method [11], exact algorithms assume the capability of computing the expected recourse (or penalty) function and the availability of a lower bound on this function. Efficiency is greatly improved when lower bounding functionals can be derived that also apply at fractional solutions. Such is the case for the PTSP [14] and for the SVRP with stochastic demands [8].

More elaborate versions of SPR can easily be modeled. They typically include more diversified recourse policies or even multistage recourse policies. They are in general much more difficult to solve. Example of successful solutions are available in dynamic vehicle allocation [16], dynamic routing [17] and re-optimization strategies [1]. A survey is available in [6].

See also

- Approximation of Extremum Problems with Probability Functionals
- Approximation of Multivariate Probability Integrals
- Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- Extremum Problems with Probability Functions: Kernel Type Solution Methods
- General Moment Optimization Problems
- General Routing Problem
- Logconcave Measures, Logconvexity
- Logconcavity of Discrete Distributions
- L-shaped Method for Two-stage Stochastic Programs with Recourse
- Multistage Stochastic Programming: Barycentric Approximation
- Preprocessing in Stochastic Programming
- Probabilistic Constrained Linear Programming: Duality Theory
- Probabilistic Constrained Problems: Convexity Theory
- Simple Recourse Problem: Dual Method
- Simple Recourse Problem: Primal Method
- Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- Static Stochastic Programming Models
- Static Stochastic Programming Models: Conditional Expectations
- Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- Stochastic Integer Programs
- Stochastic Linear Programming: Decomposition and Cutting Planes
- Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- Stochastic Network Problems: Massively Parallel Solution
- Stochastic Programming: Minimax Approach
- Stochastic Programming Models: Random Objective
- Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- Stochastic Programming with Simple Integer Recourse
- Stochastic Programs with Recourse: Upper Bounds
- Stochastic Quasigradient Methods in Minimax Problems
- Two-stage Stochastic Programming: Quasigradient Method
- Two-stage Stochastic Programs with Recourse
- Vehicle Routing
- Vehicle Scheduling

References

1. Bertsimas DJ (1992) A vehicle routing problem with stochastic demands. Oper Res 40:574–585
2. Bertsimas DJ, Jaillet P, Odoni AR (1990) A priori optimization. Oper Res 28:1019–1033
3. Birge JR, Louveaux FV (1997) Introduction to stochastic programming. Springer, Berlin
4. Dror M, Trudeau P (1986) Stochastic vehicle routing with modified savings algorithm. Europ J Oper Res 23:228–235
5. Fisher ML (1995) Vehicle routing. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing. Handbook Oper Res and Management Sci. North-Holland, Amsterdam
6. Gendreau M, Laporte G, Séguin R (1996) Stochastic vehicle routing. Europ J Oper Res 88:3–12
7. Gendreau M, Laporte G, Séguin R (1996) A tabu-search heuristic for the vehicle routing problem with stochastic demands and customers. Oper Res 44:469–477
8. Hjorring C, Holt J (1997) New optimality cuts for a single-vehicle stochastic routing problem. In: Ann. Oper. Res.: Recent Advances in Combinatorial Optimization, Theory and Applications. Baltzer, Basel
9. Jaillet P (1988) A priori solution of a traveling salesman problem in which a random subset of customers are visited. Oper Res 36:929–936
10. Laporte G (1997) Vehicle routing. In: Dell'Amico M, Maffioli F, Martello S (eds) Annotated Bibliography in Combinatorial Optimization. Wiley, New York
11. Laporte G, Louveaux FV (1993) The integer L-shaped method for stochastic integer programs with complete recourse. Oper Res Lett 13:133–142
12. Laporte G, Louveaux FV (1998) Solving stochastic routing problems with the integer L-shaped method. In: Crainic TG, Laporte G (eds) Fleet Management and Logistics. Kluwer, Dordrecht
13. Laporte G, Louveaux FV, Mercure H (1992) The vehicle routing problem with stochastic travel times. Transport Sci 26:161–170

14. Laporte G, Louveaux FV, Mercure H (1994) A priori optimization of the probabilistic traveling salesman problem. Oper Res 42:543–549
15. Powell WB (1988) A comparative review of alternative algorithms for the dynamic vehicle allocation problem. In: Golden BL, Assad AA (eds) Vehicle Routing: Methods and Studies. North-Holland, Amsterdam
16. Powell WB, Jaillet P, Odoni AR (1995) Stochastic and dynamic network and routing. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing. Handbook Oper Res and Management Sci. North-Holland, Amsterdam
17. Psaraftis HN (1988) Dynamic vehicle routing problems. In: Golden BL, Assad AA (eds) Vehicle Routing: Methods and Studies. North-Holland, Amsterdam
18. Stewart WR Jr, Golden BL (1983) Stochastic vehicle routing: A comprehensive approach. Europ J Oper Res 14:371–385

Structural Optimization

KRISTER SVANBERG
Royal Institute of Technology (KTH),
Stockholm, Sweden

MSC2000: 49M37, 65K05, 90C30

Article Outline

References

Structural optimization is a discipline dealing with optimal design of load-carrying structures. Some examples of such problems are to find an optimal shape of a suspension arm in a car, to find an optimal material distribution in the wall of a centrifugal separator, to find optimal thicknesses of composite material layers in wing panels of an aircraft, or to find optimal cross sectional dimensions of the different beams in a new Eiffel tower!

Structural optimization problems are often modeled as nonlinear programming problems of the form

$$\begin{aligned} &\text{minimize } f_0(x) \\ &\text{subject to } f_i(x) \le b_i, \quad i = 1, \dots, m, \\ &\qquad\qquad x \in X\,. \end{aligned} \tag{1}$$

Here, $x = (x_1, \dots, x_n)^{\mathrm{T}}$ is the vector of design variables, like cross sectional dimensions of bars and beams, thicknesses of membranes and plates, variables describing the shape of the structure, etc. The objective function $f_0(x)$ is typically the structural weight, while the inequalities $f_i(x) \le b_i$ model constraints on displacements, stresses, moments of inertia, eigenfrequencies, etc. Finally, X is a given rectangular subset of $\mathbb{R}^n$, defining simple lower and upper bounds on the variables.

The functions $f_i(x)$ are not explicit. Instead, they are typically of the form

$$f_i(x) = h_i(x, u(x))\,, \tag{2}$$

where $h_i(x, u)$ are explicit functions while the "state vector" (u) depends implicitly on the design variable vector x through some system of state equations. Often, u is the nodal displacement vector in a *finite element* model and the state equations are of the form

$$Ku = p\,, \tag{3}$$

where $K = K(x)$ is the *stiffness matrix* of the structure (in the finite element model), while $p = p(x)$ is a vector describing the loads on the structure. This means that each time the constraint functions should be evaluated at some point x, the state equations must be generated and solved, typically by some finite element package. The function evaluations could therefore become very time consuming.

An encouraging fact, however, is that for many problems of this type it is possible to calculate gradients of the constraint functions in an efficient way. Since the possibility of calculating gradients is a key point for solving structural optimization problems, we now describe in some detail a so called adjoint method for such calculations, assuming that the considered problem is on the above form.

First, the chain rule gives

$$\frac{\partial f_i}{\partial x_j} = \frac{\partial h_i}{\partial x_j} + q_i^{\mathrm{T}} \frac{\partial u}{\partial x_j}\,, \tag{4}$$

where the components of the row vector q_i^{T} are the partial derivatives of h_i with respect to the components of u, calculated at the current point (x, u).

Next, with q_i from above, let the vectors v_i be obtained from the systems

$$Kv_i = q_i, \quad i = 1, \dots, m\,. \tag{5}$$

When the system (3) was recently solved for obtaining the displacement vector u corresponding to the current

x, the matrix K was generated and factorized, typically by a banded Cholesky method. This calculated factorization should naturally be used again when solving (5).

By differentiating both sides of (3), one obtains

$$K\frac{\partial u}{\partial x_j} + \frac{\partial K}{\partial x_j}u = \frac{\partial p}{\partial x_j}\,, \tag{6}$$

from which it follows, after multiplying by v_i^{T}, that

$$q_i^{\mathrm{T}}\frac{\partial u}{\partial x_j} = v_i^{\mathrm{T}}\frac{\partial p}{\partial x_j} - v_i^{\mathrm{T}}\frac{\partial K}{\partial x_j}u\,. \tag{7}$$

Together with (4), this implies that

$$\frac{\partial f_i}{\partial x_j} = \frac{\partial h_i}{\partial x_j} + v_i^{\mathrm{T}}\frac{\partial p}{\partial x_j} - v_i^{\mathrm{T}}\frac{\partial K}{\partial x_j}u\,, \tag{8}$$

where all the terms on the right hand side are fairly straightforward to calculate.

In most applications, the considered structure should be able to carry several different loads. This means that instead of a single load vector p in (3), there are several given load vectors $p_1, \ldots, p_L$, and corresponding displacement vectors $u_1, \ldots, u_L$. After obvious modifications of the notations, the above description of gradient calculations remains valid.

The major computational work when calculating function values and gradients thus consists in solving the systems (3) and (5) for a possibly large number of given right hand side vectors. However, the stiffness matrix $K(x)$ only has to be factorized once for the current x.

Because function evaluations are very expensive, and because gradients can be calculated almost at the same time as function values, the following iterative approximation approach has become well established for solving a large class of structural optimization problems on the above form.

Step 0. Choose a starting point $x^{(1)} \in X$ and set the iteration index $k = 1$.

Step 1. Given $x^{(k)}$, calculate $f_i(x^{(k)})$ and gradients $\nabla f_i(x^{(k)})$ for $i = 0, 1, \ldots, m$.

Step 2. Generate an approximating subproblem of the form

$$\begin{aligned} &\text{minimize } g_0^{(k)}(x) \\ &\text{subject to } g_i^{(k)}(x) \le b_i, \quad i = 1, \ldots, m, \\ &\qquad\qquad x \in X^{(k)}\,, \end{aligned} \tag{9}$$

where $g_i^{(k)}(x)$ are explicit functions which approximate the implicit functions $f_i(x)$, while $X^{(k)}$ is a rectangular subset of X containing $x^{(k)}$.

Step 3. Solve this explicit subproblem with some suitable method and let the optimal solution be the next iteration point $x^{(k+1)}$. Then set $k = k + 1$ and go to Step 1 again.

The process is terminated when some reasonable convergence criteria have been fulfilled, or (in practice) when the marginal improvements from the latest iterations have become so small that the user does not find it worthwhile to continue. As mentioned above, each single iteration may take a considerable time.

A crucial step in this approach is to make a clever choice of approximating functions $g_i^{(k)}(x)$. The main information available for doing that are the calculated function values and gradients, at the current iteration point $x^{(k)}$ as well as at previous points. In addition, some relevant properties of the considered problem may be known. As an example, it is known that the normal stress in a truss element decreases approximately as $1/x_j$ if the cross section area x_j of the element increases, and a related type of behavior holds also for the nodal displacements. This kind of general information could be most valuable when the approximating functions should be chosen. Finally, it is important that the subproblem (9) does not become too hard to solve numerically. For this reason, and to avoid nonglobal local optima of the subproblem, it is to prefer that the chosen approximating functions $g_i^{(k)}(x)$ are convex. It should be noted, though, that the original functions $f_i(x)$ may very well be nonconvex. This implies that the optimality conditions used for terminating the process do not, in general, guarantee a global optimum of the original problem.

The approach above, Step 0 – Step 3, will now be exemplified by a specific method, further discussed in e. g., [2], which is well suited for problems of the following type: The design variables x_j are assumed to be transverse sizes of structural elements, such as cross section areas of truss elements or thicknesses of membrane elements. This makes the stiffness matrix $K(x)$ linear in x. It is further assumed that there are strictly positive lower bounds defined for these variables, which implies that $K(x)$ is always positive definite. The objective function is assumed to be the structural weight,

which is linear in x. Finally, the constraint functions are assumed to model given limitations on stresses and displacements at different given points in the structure, for different given load vectors.

For this type of problems, the constraint functions f_i satisfy the relation $f_i(\alpha x) = (1/\alpha) f_i(x)$ for every vector x with strictly positive components and every scalar $\alpha > 0$. This makes it reasonable to approximate each constraint functions by a linearization in the inverse design variables $1/x_j$. The approximating functions $g_i^{(k)}(x)$ are thus chosen as follows, for $i = 1, \ldots, m$,

$$g_i^{(k)}(x) = f_i(x^{(k)}) + \sum_{j=1}^{n} a_{ij} \left(\frac{1}{x_j} - \frac{1}{x_j^{(k)}} \right) , \qquad (10)$$

where

$$a_{ij} = \frac{\partial f_i}{\partial (1/x_j)} = -x_j^2 \frac{\partial f_i}{\partial x_j} , \qquad (11)$$

calculated at $x = x^{(k)}$.

There is no need to approximate the objective function, since it is already linear in x. Thus,

$$g_0^{(k)}(x) = f_0(x) = \sum_{j=1}^{n} c_j x_j . \qquad (12)$$

Therefore, the subproblem (9) becomes as follows, where a temporary change of variables to $y_j = 1/x_j$ has been made,

$$\begin{aligned} &\text{minimize} \quad \sum_{j=1}^{n} \frac{c_j}{y_j} \\ &\text{subject to} \quad \sum_{j=1}^{n} a_{ij} y_j \le b_i, \ i = 1, \ldots, m \\ &\qquad\qquad y \in Y^{(k)} . \end{aligned} \qquad (13)$$

This is a tractable problem with a separable and strictly convex objective functions, linear inequality constraints and simple bounds on the variables. One of several possible ways of solving this subproblem is to form the corresponding dual problem, which is of the form

$$\begin{aligned} &\text{maximize} \quad \varphi(\lambda) \\ &\text{subject to} \quad \lambda_i \ge 0, \quad i = 1, \ldots, m \end{aligned} \qquad (14)$$

where $\varphi(\lambda)$ is a concave, continuously differentiable, explicit function. By solving this dual problem by some suitable method, like a modified Newton method, one also obtain the optimal solution of the primal subproblem (13).

If somewhat more elaborate approximating functions $g_i^{(k)}(x)$ are used, the above approach of solving a sequence of separable convex subproblems can in fact be made *globally convergent*, so that the sequence of generated iteration points $x^{(k)}$ always converges towards the set of KKT points of the original problem (1), see [3].

An expanding subfield of structural optimization is so called *topology optimization*, where a central ingredient is the interest for the holes in the structure. In addition to the optimal outer shape of the structure, one now also search for the optimal number, location and shape of these holes. The corresponding optimization models typically involve a large number of binary variables which indicate presence ($x_j = 1$) or absence ($x_j = 0$) of material in various points of the considered structure. For an excellent survey on topology optimization, see [1].

References

1. Bendsøe MP, Sigmund O (2003) Topology optimization: theory, methods and applications. Springer, Berlin
2. Fleury C (1979) A unified approach to structural weight minimization. Comput Method Appl Mech Eng 20:17–38
3. Svanberg K (2002) A class of globally convergent optimization methods based on conservative convex separable approximations. SIAM J Optim 12:555–573

Structural Optimization: History

Raphael T. Haftka[1],
Jaroslaw Sobieszczanski-Sobieski[2]
[1] University Florida, Gainesville, USA
[2] NASA Langley Research Center, Hampton, USA

MSC2000: 90C90, 90C26

Article Outline

Keywords
See also
References

Keywords

Structural optimization; Structural shape optimization; Structural topology optimization; Sequential approximate optimization; Sensitivity derivatives

The term *structural optimization* is commonly used for the optimization of *engineering structures*, such as building, automobile, or airplane structures for improved strength or stiffness properties and reduced weight or cost. Before computer based optimization became widely used, structural components, such as *beams* and *plates* were optimized by using the calculus of variations [5].

The computerized analysis of structures, via models that discretize the structure into a large number of pieces, known as *finite elements*, has become prevalent in the 1960s. Numerical optimization based on finite element models started in the early 1960s by L. Schmit and his students [6]. The early years were characterized by applications for civil engineering truss structures, with the design variables being cross-sectional areas of the elements. Later these variables were generalized to cross-sectional dimensions of beams and thicknesses of plates. This class of design variables, so called *sizing variables*, has the distinction that the optimization can be carried out with only superficial changes in the finite element model.

More recently, structural optimization research has focused on changing the *shape* (geometry) and *topology* of the structural configuration. Geometrical changes require redefinition of the finite element mesh. Topological changes, which consist of adding or removing parts as well as creating holes, pose even more difficult challenges in converting the structural design into a manageable optimization problem [1,2].

A major driving force in the development of structural optimization methodology has been the need to accommodate very large number of design variables (hundreds or thousands), while a single structural analysis (evaluation of objective function and constraints) requires the solution of thousands to hundred of thousands of algebraic equations derived from the finite element method. This computational challenge has been addressed by several devices, many quite unique to structural optimization.

- For optimization problems subject only to stress limit constraints, an intuitive *optimality criterion* has been employed, that stipulates that each part of the structure is stressed to its limit, imposed by material properties or by buckling, under at least one loading condition. This optimality condition is accompanied by techniques that remove material from regions that are under stressed and add material to regions that are overstressed, without the need to calculate derivatives. This approach is often termed *fully stressed design*.
- In many problems the number of active constraints, aside from lower limits on the design variables, can be made much smaller than the number of design variables. Various dual optimization formulations then become more effective than direct formulations [1].
- The most popular approach for solving structural optimization problems of high dimension is *sequential approximate optimization*, a generalization of sequential linear programming. In this approach the objective function and constraints are replaced by approximations using first derivatives, which are linear in either the design variables or their reciprocals. Convex approximations are particularly popular [3].
- Efficient calculation of *derivatives of structural response* with respect to design variables is a major field of active research. Methods that differentiate the continuum equations and then discretize compete with methods which differentiate the discretized finite element equations. *Adjoint derivative methods* are usually superior when the number of differentiated response quantities is less than the number of the design variables.
- For *topology optimization* problems, where the number of design variables is extremely large, global compliance is often used as a single measure of structural performance. This allows the development of efficient specialized algorithms, or at least extremely cheap calculation of derivatives.
- In the structural design problem, the overall design is usually carried out first, using a coarse analysis and optimization models to determine the overall material distribution, and possibly shape and topology. This is followed by, detail design of different parts of the structure, for example, individual spars and panels in aircraft structures. In principle, there ought to be feedback from the second stage to the

first, but the iterations implied by such feedback are often impractical for cost and time reasons. Unfortunately, there is still no completely satisfactory method to such two-stage design in a rigorous and computationally efficient way.

While most structural optimization problems are formulated as continuous problems, there is also substantial interest in *discrete design variables*, and these fall in two categories: those that appear as continuous in the analysis but are available for actual implementation in limited sets; and those that appear as discrete in the analysis. An example of the first category are civil engineering applications of *beam cross-sectional shapes* and dimensions, which are readily available only in standardized sets, and using other shapes increases the cost substantially. An example of the second category is are choices of material and topology. The increasing usage of fiber reinforced *laminated composite materials* also introduces variables of both categories, creating discrete and *combinatorial problems*. This is due to the fact that thicknesses have to be integer multiples of the basic ply thickness, and fiber angles are usually limited to a small discrete set by the availability of test data. Genetic algorithms have been popular for such applications [4].

See also

- ► Bilevel Programming: Applications in Engineering
- ► Design Optimization in Computational Fluid Dynamics
- ► Interval Analysis: Application to Chemical Engineering Design Problems
- ► Multidisciplinary Design Optimization
- ► Multilevel Methods for Optimal Design
- ► Optimal Design of Composite Structures
- ► Optimal Design in Nonlinear Optics
- ► Structural Optimization
- ► Topology Optimization

References

1. Beckers M, Fleury C (1997) Topology optimization involving discrete variables. In: Gutkowski W, Mroz Z (eds) Proc. Second World Congress of Structural and Multidisciplinary Optimization. Inst. Fundum. Techn. Res., Warsaw
2. Bendse MP (1995) Optimization of structural topology, shape, and material. Springer, Berlin
3. Haftka RT, Gürdal Z (1992) Elements of structural optimization, 3rd edn. Kluwer, Dordrecht
4. Nagendra S, Jestin D, Gürdal Z, Haftka RT, Watson LT (1996) Improved genetic algo- rithms for the design of stiffened composite panels. Comput Structures 58(3):543–555
5. Rozvany GIN (1989) Structural design via optimality criteria. Kluwer, Dordrecht
6. Schmit LA Jr (1971) Structural synthesis 1959–1969: A decade of progress, Recent Advances in Matrix Methods of Structural Analysis and Design. Univ. Alabama Press, Tuscaloosa, pp 565–634

Suboptimal Control

Rein Luus
Department Chemical Engineering,
University Toronto, Toronto, Canada

MSC2000: 90C30

Article Outline

Keywords
Time Suboptimal Control
Use of Suboptimal Control in Complex Systems
Suboptimal Control in Other Situations
See also
References

Keywords

Optimization; Direct search; Suboptimal control; Time optimal control; Approximation; Model reduction

Frequently, optimal control of engineering processes is difficult to achieve, or the resulting structure of the optimal control policy may not be in an appropriate form for application. This leads us to make certain approximations to the formulation of the problem, to simplify the model describing the process, or to impose some structure on the nature of the control policy. Therefore, instead of solving the original optimal control problem, the *optimal control policy* is established to a closely related problem. The solution to the related problem is said to be *suboptimal control* policy of the original problem.

Time Suboptimal Control

To illustrate the suboptimal control in one important area, let us consider the *time optimal control* problem, where the system is described by the differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad \text{with } \mathbf{x}(0) \text{ given}, \tag{1}$$

where $\mathbf{x}$ is an n-dimensional state vector and $\mathbf{u}$ is an r-dimensional control vector bounded by

$$\alpha_j \le u_j(t) \le \beta_j, \quad j = 1, \ldots, r. \tag{2}$$

The time optimal control problem is to determine the control $\mathbf{u}$ in the time interval $0 \le t < t_f$, so that the origin $\mathbf{x}(t_f) = \mathbf{0}$, is reached in minimum time t_f. The origin is reached when

$$\|x_i(t_f)\| < \epsilon_i, \quad i = 1, \ldots, n, \tag{3}$$

where ϵ_i is some specified tolerance, such as accuracy of measurement of the state variables.

For the case of scalar control ($r = 1$), and $n = 2$, this time optimal control problem is easily solved by establishing the switching curves in the phase plane and then to follow the particular trajectory starting from the given initial condition to the *switching curve* and then following the switching curve to the origin. This is illustrated in [8, pp. 146–150].

Of interest is to solve the time optimal control problem for higher-dimensional problems. For the general case the problem is very difficult, even with *iterative dynamic programming* (IDP) [3,4]. The special case of time optimal control of a linear system with $n = 6$ and $r = 2$ was solved in [9] by using linear programming on the discretized form of the system to seek the minimum number of time steps to provide a feasible solution. The computational aspects and results of using *linear programming* on a 6-plate gas absorber are given in [8, pp. 212–223] and [1]. The optimal control policy involves switching the two control variables from bound to bound several times. Therefore, we may be interested in allowing the final time to be increased somewhat if we get a 'more stable' control policy, and one that would not be very sensitive to modeling errors.

In order to stabilize the control policy for the discrete-time version of the problem and still drive the system to the origin, R. Koepcke and L. Lapidus [7], suggested the construction of a *positive definite quadratic function* of state

$$V(k) = \mathbf{x}^\top(k)\mathbf{Q}\mathbf{x}(k) \tag{4}$$

and to choose the control policy to minimize the forward difference

$$\Delta V(k) = V(k+1) - V(k). \tag{5}$$

If any of the calculated control variables are beyond the boundary specified by (2), then the boundary values are used. Such *clipping technique* is widely used in optimal control to handle control constraints. If $\Delta V(k)$ is negative, then we have the added benefit of having *asymptotic stability* stability, and $V(k)$ is called a *Lyapunov function*.

There is a certain amount of freedom in choosing an appropriate positive definite matrix $\mathbf{Q}$. Although $\mathbf{Q}$ may be chosen to be the identity matrix, such a choice is not the best. For a six-plate gas absorber model, Lapidus and R. Luus [8, pp. 363–369] suggested the use of the diagonal matrix

$$\mathbf{Q} = \text{diag}[1, 7.39, 230, 230, 7.39, 1], \tag{6}$$

rather than an identity matrix. The elements in $\mathbf{Q}$ were chosen to put more weight on the inner stages to counterbalance the logarithmic damping produced by each stage. Thus, $7.39 = e^2$, and $230 = (e^e)^2$ where e is the base for the natural logarithm. The criterion that $|x_i(t_f)| \le 0.001$, $i = 1, \ldots, 6$ was satisfied with $t_f = 9.0$ minutes, which is reasonably close to the value $t_f = 6.0$ minutes obtained by linear programming for the original problem [1,9]. To improve the result obtained by suboptimal procedure, different values for $\mathbf{Q}$ were examined [2,5]. Instead of using *Rosenbrock's hillclimbing procedure* [24], Luus [12] found that the use of *direct search* [16] gave surprisingly good results. The results were surprising, since $t_f = 4.8$ minutes that was obtained, was better than the previously accepted value of 6.0. This apparent paradox, where the suboptimal control yielded better results than the optimal control was resolved in [23], where the high *sensitivity* of t_f on the final state specification is shown. With optimal control we were driving the system to the mathematical origin, rather than the practical origin where all variables had to be less than 0.001 in value. When this relaxed condition was incorporated into the linear programming algorithm, a minimum time of $t_f = 4.5$ minutes resulted.

Therefore, the suboptimal control here served as a good means to check the results obtained in solving the original problem, and provided a nice means of reinterpreting the original optimal control problem. The use of the quadratic function $V = \mathbf{x}^{\mathsf{T}}(k)\mathbf{Q}\mathbf{x}(\mathrm{k})$ in general optimal control problems to simplify the formulation and to provide good suboptimal results have been reported [6,11,25,27].

Another approach to time suboptimal control is to simply carry out search on the elements of the feedback gain matrix [10], or choose the gain matrix elements so that the eigenvalues of the linearized system are shifted as far left as possible in the imaginary versus real graph [18]. With direct search optimization this search can be readily accomplished. These two approaches were combined into a two-step procedure in [20], to enable the desired state to be approached very rapidly, and yet in a stable manner.

Use of Suboptimal Control in Complex Systems

When the given system is very complex, or of very high dimension, very good results can be obtained by reducing the dimensionality or complexity of the system before attempting to determine the optimal control policy. This is similar to the idea in nonlinear analysis where averaging technique may be used to average out the noncontributing terms and yet provide accurate stability information [17]. The optimal control policy obtained for the simplified system is then the suboptimal control for the original system. To get good results, one usually tries to get the best simplified or *reduced model*. Optimization has been used to obtain excellent models even when the original system has been of very high dimension [13,33,32].

Another useful approach is to use *orthogonal collocation* to change a partial differential equation into a set of ordinary differential equations [21,29,31] or by using *coordinate transformation* [30]. K.T. Wong and Luus [28] showed that a very good simplified model for a staged system, such as a gas absorber, that is modeled as a large number of differential equations, can be established by first converting the ordinary differential equations into a partial differential equation and then to use orthogonal collocation to yield a small number of ordinary differential equations that depict the behavior of the system quite accurately.

Suboptimal Control in Other Situations

Suppose that the optimal control policy requires the use of all the state variables in the control law, but it is impractical to measure all the variables. To handle that situation we can establish a control law which uses only the variables that can be measured. We have in essence an *incomplete state feedback*, and it is important to examine what is lost by not measuring all the state variables. This is an important area, and good progress has been made [26].

In *time-delay systems*, if the time delay is small, then, as was shown in [21], *Taylor series* approximation may be used to convert the time-delay system into a set of ordinary differential equations. The establishment of optimal control for the latter is much easier, but results in suboptimal control for the original system. The degree of *suboptimality* for realistic values for the delay terms was found to be quite small [21]. Such an approach was also used in [19,15], to obtain *suboptimal control* policy for a time-delay system. This suboptimal control policy was then used as the initial control policy in using iterative dynamic programming with piecewise linear control.

Another area where suboptimal control has been used is in the choice of the final time t_f. When t_f is relatively small, the choice of t_f is very important, because the optimal control policy is quite sensitive to t_f [14]. If, however, t_f is relatively large, the system is linear and the performance index is quadratic, the choice of $t_f = \infty$ simplifies the solution of the optimal control problem, since the resulting *Riccati equation* then is an algebraic equation and not a differential equation that has to be integrated backward.

Suboptimal control therefore serves a very useful role in the optimal control field.

See also

- ► Control vector iteration
- ► Duality in optimal control with first order differential equations
- ► Dynamic programming: Continuous-time optimal control
- ► Dynamic programming and Newton's method in unconstrained optimal control
- ► Dynamic programming: Optimal control applications

- ▶ Hamilton–Jacobi–Bellman equation
- ▶ Infinite horizon control and dynamic games
- ▶ MINLP: Applications in the interaction of design and control
- ▶ Multi-objective optimization: Interaction of design and control
- ▶ Optimal control of a flexible arm
- ▶ Optimization strategies for dynamic systems
- ▶ Robust control
- ▶ Robust control: Schur stability of polytopes of polynomials
- ▶ Semi-infinite programming and control problems
- ▶ Sequential quadratic programming: Interior point methods for distributed optimal control problems

References

1. Bashein G (1971) A simplex algorithm for on-line computation of time optimal controls. IEEE Trans Autom Control AC-16:479–482
2. Bennett HW, Luus R (1971) Application of numerical hill-climbing in control of systems via Liapunov's direct method. Canad J Chem Eng 49:685–690
3. Bojkov B, Luus R (1994) Time-optimal control by iterative dynamic programming. Industr Eng Chem Res 33:1486–1492
4. Bojkov B, Luus R (1995) Time-optimal control of high dimensional systems by iterative dynamic programming. Canad J Chem Eng 72:380–390
5. Chant VG, Luus R (1968) Time suboptimal control of the gas absorber. Canad J Chem Eng 46:376–381
6. Cormack DE, Luus R (1972) Suboptimal control of chemical engineering systems. Canad J Chem Eng 50:390–398
7. Koepcke R, Lapidus L (1961) Dynamic control of chemical engineering processes using a method of Lyapunov. Chem Eng Sci 16:252–258
8. Lapidus L, Luus R (1967) Optimal control of engineering processes. Blaisdell, Waltham, pp 354–369
9. Lesser HA, Lapidus L (1966) The time-optimal control of discrete-time linear systems with bounded controls. AIChE J 12:143–152
10. Luus R (1974) Optimal control by direct search on feedback gain matrix. Chem Eng Sci 29:1013–1017
11. Luus R (1974) A practical approach to time-optimal control of nonlinear systems. Industr Eng Chem Process Des Developm 13:405–408
12. Luus R (1974) Time-optimal control of linear systems. Canad J Chem Eng 52:98–102
13. Luus R (1980) Optimization in model reduction. Internat J Control 32:741–747
14. Luus R (1991) Effect of the choice of final time in optimal control of nonlinear systems. Canad J Chem Eng 69:144–151
15. Luus R (2000) Iterative dynamic programming. Chapman and Hall/CRC, London, pp 143–148
16. Luus R, Jaakola THI (1973) Optimization by direct search and systematic reduction in the size of search region. AIChE J 19:760–766
17. Luus R, Lapidus L (1966) Use of averaging technique for stability analysis. Chem Eng Sci 21:159–181
18. Luus R, Mutharasan R (1974) Stabilization of linear system behaviour by pole shifting. Internat J Control 20:395–405
19. Luus R, Zhang X, Hartig F, Keil FJ (1995) Use of piecewise linear continuous control for time–delay systems. Industr Eng Chem Res 34:4136–4139
20. Oh SH, Luus R (1975) Two-step procedure for time-suboptimal control. Internat J Control 22:749–762
21. Oh SH, Luus R (1976) Optiml feedback control of time-delay systems. AIChE J 22:140–147
22. Oh SH, Luus R (1977) Use of orthogonal collocation method in optimal control problems. Internat J Control 26:657–673
23. Rosen O, Imanudin, Luus R (1987) Sensitivity of the final state specification for time-optimal control problems. Internat J Control 45:1371–1381
24. Rosenbrock HH (1960) An automatic way of finding the greatest or least value of a function. Comput J 3:175–184
25. Schlossmacher EJ, Lapidus L (1971) The suboptimal control of nonlinear systems using Liapunov-like functions. AIChE J 17:1330–1341
26. Therapos CP (1982) Suboptimal control with incomplete state feedback. Electronics Lett 18:996–997
27. Tsang ACC, Luus R (1973) Approximation in control of nonlinear dynamic systems. AIChE J 19:327–334
28. Wong KT, Luus R (1980) Model reduction of high-order multistage systems by the method of orthogonal collocation. Canad J Chem Eng 58:382–388
29. Wong KT, Luus R (1982) Time suboptimal feedback control of systems described by linear parabolic partial differential equations. Optimal Control Appl Meth 3:177–185
30. Wong KT, Luus R (1983) Time suboptimal feedback control design through coordinate transformation. Internat J Control 37:723–729
31. Wong KT, Luus R (1983) Time suboptimal feedback control of high order linear systems. Internat J Control 37:89–109
32. Yang SM, Luus R (1983) A note on model reduction of digital control systems. Internat J Control 37:437–439
33. Yang SM, Luus R (1983) Optimization in linear system reduction. Electronics Lett 19:635–637

Successive Quadratic Programming

Angelo Lucia
Department Chemical Engineering,
University Rhode Island, Kingston, USA

MSC2000: 90C30

Article Outline

Keywords
Some Nonlinear Programming Basics
The Fundamental Building Blocks
of Successive Quadratic Programming
Estimating the Hessian Matrix
of the Lagrangian Function
Methods for Solving
the Recursive Quadratic Programs
Active Set Methods
Interior Point Methods
Global Convergence and Stabilization Techniques
Line Searching
Trust Region Methods
A Generic SQP Algorithm
Some Brief Comments on Numerical Performance
See also
References

Keywords

Successive quadratic programming; Lagrangian function; Kuhn–Tucker conditions; Quadratic programming subproblem; Hessian matrix of a Lagrangian function; Range and null space decomposition; Full space methods; Active set methods; Interior point methods; Line searching; Trust region

Successive quadratic programming (or SQP) methods are a class of methods for finding a local optimum to nonlinearly constrained optimization (or nonlinear programming) problems. Introduced by R.B. Wilson [19] in the early 1960s, followed by variants by W. Murray [13] and M.C. Biggs [2], and then popularized and refined by S.P. Han [10] and M.J.D. Powell [16], SQP methods are based on the recursive use of quadratic programming to calculate iterative improvements to the estimates of the constrained optimum and corresponding Lagrange and Kuhn–Tucker multipliers. This use of recursive quadratic programming can be thought of as a means of balancing the tasks of satisfying the nonlinear constraints and optimizing the objective function. That is, starting estimates of the unknown variables and iterates do not have to satisfy the nonlinear constraints at each iteration, as they do in many other methods for nonlinearly constrained optimization. Rather the nonlinear constraints are satisfied as the iterates approach the optimum. Consequently, this together with the use of analytical and/or quasi-Newton estimates of the (Hessian) matrix of second derivatives of the Lagrangian function which account for nonlinear constraint curvature, are often cited as the two primary reasons why SQP methods are more reliable and more efficient (usually requiring fewer function and gradient evaluations) than other nonlinear programming techniques in solving nonlinearly constrained optimization problems. Successive quadratic programming methods have, in more recent years, been extended to large scale and nonconvex, nonlinearly constrained optimization [11,14] and applied successfully in various mathematical, scientific, and engineering disciplines ([1,11,17]).

The fundamental building blocks of any SQP method generally include:

- methods for calculating or estimating the Hessian matrix of the Lagrangian function,
- procedures for solving the successive quadratic programs, and
- stabilization techniques for forcing convergence from 'poor' starting points.

Some Nonlinear Programming Basics

Successive quadratic programming and other local constrained optimization techniques seek to find a local solution to the following nonlinear programming (NLP) problem:

$$\begin{cases} \min & f(x) \\ \text{s.t.} & c(x) \leq 0, \end{cases}$$

where x is a vector of length n which represents an estimate of the local minimum, $f(x)$ is a twice continuously differentiable objective function and $c(x)$, the vector of equality and/or inequality constraints, is also twice continuously differentiable and nonlinear. This constrained optimization problem is commonly recast in terms of the *Lagrangian function*, $L(x)$, defined by

$$L(x) = f(x) + \lambda^{\top} c(x) + \mu^{\top} c(x),$$

where λ and μ are vectors of Lagrange and Kuhn–Tucker multipliers associated with the equality and inequality constraints respectively. The conditions that

define local solutions to the original nonlinearly constrained optimization problem are called Kuhn–Tucker (or Karush–Kuhn–Tucker) conditions and are represented by stationarity of the Lagrangian function. The *Kuhn–Tucker conditions* for the nonlinearly constrained optimization problem shown above are

$$g_L(x) = g(x) + \lambda^\top J_E(x) + \mu^\top J_I(x) = 0\,,$$
$$c(x) = 0, \qquad \mu^\top c(x) = 0, \quad \mu \geq 0\,,$$

where $g(x)$ is the gradient of the objective function, $J_E(x)$ is the Jacobian matrix (or matrix of first partial derivatives) of the equality constraints, $J_I(x)$ is the Jacobian matrix of the inequality constraints, $g_L(x)$ is the gradient of the Lagrangian function and the complementarity conditions, $\mu^\top c(x) = 0$, for the inequality constraints are interpreted as follows: if $c_i(x) = 0$, then $\mu_i > 0$ or if $c_i(x) < 0$, then $\mu_i = 0$ for each inequality separately. Successive quadratic programming methods can be thought of as an application of Newton or quasi-Newton methods to the NLP Kuhn–Tucker conditions, with one important difference. Direct application of Newton or quasi-Newton methods to the Kuhn–Tucker conditions for the nonlinear program requires a priori knowledge of the set of active constraints (i. e., the equalities plus the inequalities that hold as equalities) and this is further complicated by the fact that the active set can change from iteration to iteration. What Wilson [19] recognized was that the active set (and therefore the Lagrange and Kuhn–Tucker multipliers) and the Newton correction in the x variables at any iteration could be determined simultaneously by solving an appropriately-posed quadratic programming subproblem. This iterative quadratic programming subproblem, which is based on a quadratic approximation to the Lagrangian function subject to linearized constraints, is given by

$$\begin{cases} \min & g(x_k)^\top \Delta x_k + \frac{1}{2}\Delta x_k^\top B_k \Delta x_k \\ \text{s.t.} & c(x_k) + J(x_k)\Delta x_k \leq 0 \end{cases}$$

where $\Delta\, x_k$ is the change in the unknown variables, B_k is some approximation to the true *Hessian matrix of the Lagrangian function*, $H(x) = H_f(x) + \sum \lambda_i H_{ci}(x) + \sum \mu_i H_{ci}(x)$, and where $H_f(x)$ and $H_{ci}(x)$ refer to the true Hessian matrices of the objective function and ith constraint respectively and J is the Jacobia matrix of the constraints. Solving this quadratic programming subproblem produces precisely the same change in the unknown variables and estimates of the Lagrange and Kuhn–Tucker multipliers as does Newton's (or a quasi-Newton) method applied to the Kuhn–Tucker conditions for the original nonlinear program provided the active set is known. Therefore, the use of successive quadratic programming has the distinct advantage of not requiring a priori knowledge of the active set! However, perhaps the single biggest disadvantage of SQP methods is that linearization of the constraints (or the use of trust regions) can sometimes make these quadratic programming subproblems infeasible (i. e., result in a feasible region for the linearized constraints that is empty).

The Fundamental Building Blocks of Successive Quadratic Programming

Computational tools for the implementation of successive quadratic programming methods require means of estimating the Hessian matrix of the Lagrangian function (i. e., by analytical, finite difference, or quasi-Newton second derivatives or a mixture thereof), methods for solving the recursive quadratic programming subproblems (i. e., active set or interior point methods), and stabilization techniques (i. e., line searching or trust region methods) for forcing convergence from 'poor' starting points.

Estimating the Hessian Matrix of the Lagrangian Function

Wilson [19] originally suggested the use of analytical second derivatives of the objective function and the constraints for approximating the Hessian matrix of the Lagrangian function. However, in the 1960s and 1970s, quasi-Newton methods for approximating second derivatives were introduced and became popular. This led to superlinear convergence results for a number of quasi-Newton updates including the *Davidon–Fletcher–Powell* (DFP) and *Powell-symmetric–Broyden* (PSB) updates [10], and eventually the use of a modified *Broyden–Fletcher–Goldfarb–Shanno* (BFGS) update [16] for calculating a new approximation to the Hessian matrix of the Lagrangian function given by

$$B_{k+1} = B_k + \frac{[\eta_k \eta_k^\top]}{(\eta_k^\top s_k)} - \frac{[B_k s_k s_k^\top B_k]}{(s_k^\top B_k s_k)},$$

where the new approximation to the Hessian matrix, B_k+ 1, is computed from the old approximation, B_k, the change in the unknown or x variables, $s_k = \Delta x_k$, and the vector $\eta_k = \theta y_k - (1 - \theta) B_k s_k$, where $y_k = g_L(x_k+1) - g_L(x_k)$ and θ depends on the relative size of $\eta_k^\top s_k$ and $s_k^\top B_k s_k$. In particular, $\theta = 1$ unless $s_k^\top s_k \leq 0.2\, s_k^\top B_k s_k$, in which case $\theta = 0.8 s_k^\top B_k s_k / (s_k^\top B_k s_k - \eta_k^\top s_k)$ was suggested in [16].

Presently, both analytical and quasi-Newton, as well as finite difference, second derivatives of the objective function and the nonlinear constraints are used for estimating $H(x)$. However, all methods for estimating second derivatives have advantages and disadvantages. Quasi-Newton updates are computationally inexpensive and can be suitably modified to maintain the positive definiteness of $H(x)$, which makes the recursive quadratic programs convex (bowl-shaped). This, in turns, guarantees that each recursive quadratic program has a unique solution and that descent in a suitably chosen stabilization procedure like line searching can be maintained in order to force convergence. However, the use of quasi-Newton methods provides only two-step *R*-superlinear convergence under reasonable conditions and the particular use of the modified BFGS update forces the Hessian matrix of the Lagrangian function to be positive definite in the full space of the variables, which is unnatural. Usually the true Hessian matrix is indefinite and only positive definite on the tangent subspace (i. e., a hyperplane) defined by the linearized constraints, even at a local constrained minimum. Moreover, SQP methods that use quasi-Newton derivatives generally require more function and gradient evaluations than SQP methods based on analytical second derivatives. The use of analytical (or finite difference) second derivative to approximate the Hessian matrix of the Lagrangian function provide faster quadratic convergence, but only at a price. That is, SQP methods that use analytical or finite difference second derivatives usually converge in fewer function and gradient evaluations than SQP methods that use quasi-Newton approximations to $H(x)$, but, in doing so, they sacrifice the guaranteed convexity of the recursive quadratic programs. This, in turn, can introduce multiple Kuhn–Tucker points into the quadratic programming subproblems, and result in the loss of descent properties associated with suitably chosen line search functions. Some application-based SQP methods [11] have used judicious mixtures of analytical and quasi-Newton second derivative information and these methods have the same properties as SQP methods based analytical or finite difference second derivatives, only the rate of convergence is still theoretically two-step *R*-superlinear.

More recent SQP methods for solving large scale problems are either based on sparsity-preserving estimates of $H(x)$, so-called full space methods, or range and null space decomposition (RND), which eliminate x variables by substitution using the equality constraints and require approximations of the projection of $H(x)$ onto the linear (tangent) subspace (i. e., the hyperplane) defined by the linearized constraints. Full space methods ([11,14]), as their name implies, result in quadratic programming subproblems in the full space of the x variables and often use analytical, finite difference, quasi-Newton or a mixture of second derivatives. RND methods, on the other hand, result in smaller quadratic programming subproblems because they eliminate x variables using the equality constraints. However, RND methods [15] must use quasi-Newton updates to build approximations to the projection of $H(x)$ on the tangent subspace, which tend to track curvature less effectively than analytical second derivative approximations.

Methods for Solving the Recursive Quadratic Programs

Given B_k, an approximation to $H(x)$, the quadratic program can be assembled and the corresponding Kuhn–Tucker conditions for the quadratic programming subproblem,

$$B_k \Delta x_k + J_E^\top \lambda_k + J_I^\top \mu_k = -g(x_k)$$

and

$$[J_E^\top J_I^\top]^\top \Delta x_k = -c(x_k),$$

where J_E is the Jacobian matrix of the equality constraints and J_I is the Jacobian matrix for the active inequality constraints, can be solved for Δx_k, λ_k and μ_k. Methods for solving quadratic programming problems can be divided into two broad categories, active set methods and interior point methods.

Active Set Methods

Active set methods iteratively determine the inequalities that hold as equalities to solve the associated quadratic program Kuhn–Tucker conditions for Δx_k, λ_k and μ_k. This is typically accomplished by repeatedly applying a set of rules for adding and deleting constraints from the estimate of the active set (i. e., an active set strategy) until a valid Kuhn–Tucker point (or solution) is determined. Moreover, the Kuhn–Tucker conditions for the quadratic program constitute a set of symmetric linear equations, can be solved using a variety of symmetric matrix factorization methods including Cholesky factorization, and the matrix factors can be modified (or updated) to accommodate changes in the active set without the need for complete refactorization at each quadratic programming iteration. Complete refactorization is only required at each SQP iteration When the quadratic program is convex, descent in the quadratic program can be maintained, the associated linear Kuhn–Tucker conditions have a positive definite projection of the Hessian matrix of the Lagrangian function onto the subspace defined by any active set of constraints, are reasonably easy to solve, and some guarantee of convergence to a unique quadratic programming solution (i. e., Kuhn–Tucker point) can be given. When the quadratic program is indefinite (i. e., due to nonconvexities), special factorization methods must be used to handle the potential indefiniteness of the projected Hessian matrix of the Lagrangian function, new rules for adding and deleting constraints to and from the active set are required, and no guarantees of convergence can usually be made [12] [3]. Also active set methods can suffer from a potential combinatorial explosion in computational overhead under certain circumstances, particularly on large quadratic programming problems.

Interior Point Methods

Recently, interior point methods have been suggested for quadratic programming [8,9,20] because they have the potential to solve problems with many variables. However limited experience in solving quadratic programming problems is currently available, and even less is available for large scale problems. Interior point methods for quadratic programming are primal-dual path following algorithms that employ a logarithmic barrier function, are based on Newton's method and permit the use of iterative methods to solve the associated Kuhn–Tucker conditions for the quadratic program. However, convexity in the quadratic program is generally required. Typical iterative linear equation-solving techniques used to solve the associated Kuhn–Tucker conditions include preconditioned conjugate gradient, generalized minimum residuals, and other so-called Krylov (or expanding) subspace methods and preconditioning techniques often are based on some partial LU factorization of the Hessian matrix. These linear equation-solving methods are particularly advantageous in solving large scale problems because they avoid fill-in in the coefficient matrix (i. e., turning zero elements into nonzero elements through the elimination process) and thus, in principle, reduce both storage and overall computational workload.

Global Convergence and Stabilization Techniques

Often times, the starting point chosen for initiating SQP computations is not within the theoretical region of convergence for a given local solution. Thus it has become standard practice in the use of SQP methods, as well as in nonlinear programming in general, to use some type of technique for forcing convergence from these so-called 'poor' starting points. The most common stabilization techniques used in SQP methods are based on either line searching or the use of trust regions.

Line Searching

The underlying concept of using line search functions in successive quadratic programming is to generate a monotonically decreasing sequence of line search (or merit) function values that maintain a compromise between satisfying the constraints and minimizing the objective function and guarantee convergence to a stationary point of the Lagrangian function. Many line searching techniques in SQP methods are based on the application of Armijo's rule to exact l_1 penalty functions and augmented Lagrangian merit functions. For example, Powell [16] uses the nondifferentiable line search function

$$\phi(x, \lambda, \mu) = f(x) + \sum |\lambda_j| \, |c_j(x)| + \sum |\mu_j| \, |c_j(x)| \,,$$

which was suggested in [10] and chooses the line search parameter, α, as the first number in the sequence (1, 0.1, 0.001, ...) for which $\phi(x_k + \alpha\Delta x_k, \lambda, \mu) < \phi(x_k, \lambda, \mu)$ and where Δx_k, λ and μ are solution of the current quadratic programming subproblem. Local superlinear convergence occurs when $\alpha = 1$ is chosen in the neighborhood of the solution. A further modification of this exact penalty function that chooses $|\lambda_j| = \max[|\lambda_j|, (|\lambda'_j| + |\lambda_j|)/]$, where λ'_j is the Lagrange multiplier for the jth constraint on the previous iteration, has been suggested in [16] in order to avoid placing too much emphasis on satisfying the constraints in the line search function. While this modification frequently gives good numerical performance, convergence guarantees can not be given because upper bounds on the multipliers are required but not known in advance. Cycling has been observed in SQP methods using nondifferentiable line search functions [5] and this, in turn, has led to the development of such things as the watchdog technique [6], which allows the line search function values to increase on some iterations.

A monotonic decreasing sequence of line search function values can also be maintained through a property known as descent (i. e., that $g_{LS}^{\top}\Delta x_k < 0$, where g_{LS} is the gradient of the line search function). This requires that the line search function be differentiable and has led to the use of the differentiable augmented Lagrangian type line search functions [17] given by

$$\phi(x, \lambda, \mu, r) = f(x) - [\lambda^{\top} c(x) - rc(x)^{\top} c(x)] - P(x, \mu, r),$$

where r is a penalty parameter that is adjusted iteratively, where $P(x, \mu, r) = \sum p_i(x, \mu, r)$ and where $p_i(x, \mu_i, r) = [\mu_i c_i(x) - rc_i(x)^2]$ if the ith inequality is active and $p_i(x, \mu_i, r) = \mu_i^2/r$ if it is inactive.

Trust Region Methods

Trust region methods for nonlinearly equality constrained optimization [4,18] are another way of forcing convergence from 'poor' starting points. More recent work can be found in [7]. These techniques add a single trust region bound to the set of linearized equality constraints and solve the modified quadratic programming subproblem defined by

$$\begin{cases} \min & g(x_k)^{\top}\Delta x_k + \frac{1}{2}\Delta x_k^{\top} B_k \Delta x_k \\ \text{s.t.} & c(x_k) + J(x_k)\Delta x_k = 0, \quad \|\Delta x_k\| \le \Delta_k, \end{cases}$$

where Δ_k is a trust region radius that is adjusted from information gathered from one iteration to the next. The primary advantages of trust region methods are that they are relatively straightforward to implement and permit the Hessian matrix of the Lagrangian function to be indefinite. While often successful, trust region methods have the particular disadvantage of causing infeasible linearized constraints. That is, when the trust region is too small, it may not be possible to satisfy the linearized constraints within the trust region. Other trust region methods for constrained optimization that define the trust region in terms of a set of bounds on each variable have been suggested [11] and thus applicable to problems involving inequality constraints. However, these also can lead to infeasible linearized constraint sets.

A Generic SQP Algorithm

A typical, generic successive quadratic programming algorithm is shown below.

1) Initialize x and B_0; define a convergence tolerance, $\epsilon > 0$, and set $k = 0$.
2) Evaluate $f(x_k)$, $g(x_k)$, $c(x_k)$ and $J(x_k)$.
3) If $\| [g_L(x_k), c(x_k)]^{\top} \|_2 < \epsilon$, $c(x_k) \le 0$ and $\mu \ge 0$, then stop; else go to step 4.
4) Construct the quadratic program

$$\begin{cases} \min & g(x_k)^{\top}\Delta x_k + \frac{1}{2}\Delta x_k^{\top} B_k \Delta x_k \\ \text{s.t.} & c(x_k) + J(x_k)^{\top}\Delta x_k \le 0 \end{cases}$$

and solve it for Δx_k, λ, and μ.
5) Determine x_{k+1} from by either line searching, trust regions or some other means.
6) Calculate a new approximation to the Hessian matrix, B_{k+1}, from analytical, finite difference, quasi-Newton or mixed second derivatives.
7) Set $k = k + 1$ and go to step 2.

Some Brief Comments on Numerical Performance

The usual measures of numerical performance in nonlinear programming, as well as other areas of numerical mathematics, are algorithmic reliability and efficiency. Reliability refers to the ability of an algorithm to find a local optimum from starting points that are 'far away' from the solution. That is, the issue of reliabil-

ity is equivalent to the question: will the SQP method converge to a local constrained optimum if the starting point is 'far away' from this solution, for whatever reasons? Clearly the answer to this question is strongly related to the use of stabilization techniques and is the primary motivation for the interest in the global convergence characteristic of SQP methods. Efficiency, on the other hand, is usually measured in terms of function and gradient evaluations (or iterations) and is related to the local convergence properties of SQP methods. When the SQP algorithm gets close to the solution, rapid convergence to the optimum that requires fewer function and gradient evaluations is desired. Many numerical studies by the principle authors of SQP methods, as well as others in the mathematical sciences [17], and various branches of engineering [1,11], have clearly demonstrated that successive quadratic programming methods are among the most reliable and efficient algorithms presently available for solving nonlinearly constrained optimization problems.

However, there are many subtle and interrelated issues (e. g., sparsity, nonconvexity, constraint infeasibility, as well as problem-specific issues like model inconsistencies and limitations) that have bearing on numerical performance and therefore considerable care must be exercised during both implementation and problem-solving. Often trades between advantages and disadvantages must be accepted solely out of necessity.

See also

- ► Feasible Sequential Quadratic Programming
- ► Optimization with Equilibrium Constraints: A Piecewise SQP Approach
- ► Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ► Successive Quadratic Programming: Applications in Distillation Systems
- ► Successive Quadratic Programming: Applications in the Process Industry
- ► Successive Quadratic Programming: Decomposition Methods
- ► Successive Quadratic Programming: Full Space Methods
- ► Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Betts JT (1995) The application of optimization techniques to aerospace systems. In: Biegler LT, Doherty MF (eds) Foundations of Computer Aided Process Design, AIChE Sym Ser. vol 304-91, pp 169–177
2. Biggs MC (1972) Constrained minimization using recursive equality quadratic programming. In: Lootsma FA (ed) Numerical Methods in Nonlinear Optimization, Acad. Press, New York, pp 411–428
3. Bunch JR, Kaufman L (1980) A computational method for indefinite quadratic programming problems. Linear Alg Its Appl 34:341–370
4. Celis MR, Dennis JE, Tapia RA (1984) A trust region strategy for nonlinear equality constrained optimization. In: Boggs PT, Byrd RH, Schnabel RB (eds) Numerical Optimization, pp 71–82
5. Chamberlain RM (1979) Some examples of cycling in variable metric methods for constrained minimization. Math Program 16:378–383
6. Chamberlain RM, Lemarechal C, Pedersen HC, Powell MJD (1982) The watchdog technique for forcing convergence in algorithms for constrained optimization. Math Program Stud 16:1–17
7. Dennis JE, Vincente LN (1997) On the convergence theory of trust region-based algorithms for equality constrained optimization. SIAM J Optim 7:927–950
8. Gill PE, Murray W, Ponceleon DB, Saunders MA (1991) Solving reduced KKT systems in barrier methods for linear and quadratic programming. SOL Report Dept Oper Res Stanford Univ 91-7
9. Goldfarb D, Liu S (1991) An $O(n^3L)$ primal interior point algorithm for convex quadratic programming. Math Program 49:325–340
10. Han SP (1976) Superlinearly convergent variable metric algorithms for general nonlinear programming problems. Math Program 11:263–282
11. Lucia A, Xu J (1990) Chemical process optimization using Newton-like methods. Comput Chem Eng 14:119–138
12. Lucia A, Xu J, D'Couto GC (1993) Sparse quadratic programming in chemical process optimization. Ann Oper Res 42:55–83
13. Murray W (1969) An algorithm for constrained minimisation. In: Fletcher R (ed) Optimization, Acad. Press, New York, pp 247–258
14. Nickel RH, Tolle JW (1989) A sparse sequential quadratic programming algorithm. J Optim Th Appl 60:453–473
15. Nocedal J, Overton ML (1985) Projected Hessian updating algorithms for nonlinearly constrained optimization. SIAM J Numer Anal 22:821–850
16. Powell MJD (1978) A fast algorithm for nonlinearly constrained optimization calculations. In: Watson G (ed) Numerical Analysis, Dundee 1977, Lecture Notes Math. Springer, Berlin, pp 144–157

17. Schittkowski K (1981) The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function, Part 1: Convergence analysis. Numer Math 38:83–114
18. Vardi A (1985) A trust region algorithm for equality constrained minimization: convergence properties and implementation. SIAM J Numer Anal 22:575–591
19. Wilson RB (1963) A simplicial method for concave programming. PhD Diss Harvard Univ, Cambridge, MA
20. Ye YY (1989) An extension of Karmarkar's algorithm and the trust region method for quadratic programming. In: Megiddo N (ed) Progress in Mathematical Programming. Springer, Berlin, pp 49–64

Successive Quadratic Programming: Applications in Distillation Systems

ANGELO LUCIA
Department Chemical Engineering,
University Rhode Island, Kingston, USA

MSC2000: 90C30, 90C90

Article Outline

Keywords
SQP Formulation
Mathematical Model
Objective Function
Equality Constraints
Inequality Constraints
Lagrangian Hessian Matrix Approximations
Initialization of the Unknown Variables and Multipliers
Algorithmic and Other Implementation Issues
Comments on the Numerical Performance of SQP Methods on Distillation Problems
See also
References

Keywords

Successive quadratic programming; Lagrangian function; Quadratic programming subproblem; Hessian matrix of a Lagrangian function; Range and null space decomposition; Full space methods; Distillation; Mass and energy balance equations; Ideal and nonideal phase equilibrium equations

The separation of the chemical components of a mixture into a variety of useful product streams is a common task in the chemical and petroleum industries and *distillation* represents the most common method for this task. Some distillation products may be sold while others may be sent to different parts of the same chemical plant for further processing. Distillation takes place in a distillation column (or large tower) that usually has a number of stages (or trays) on which vapor flowing up the column is brought into contact with liquid flowing down the column. It is this contact between liquid and vapor that affects the separation. Unfortunately distillation is often very energy intensive and thus very costly. As a result, optimal operation of a distillation or sequence of distillation columns is desirable.

In a mathematical programming framework, the optimal design of a distillation column is a mixed integer nonlinear program (MINLP) because it involves both discrete variables (i. e., the number of stages in the column, the feed tray location, etc.) and continuous variables (i. e., flow rates, compositions, temperatures, pressures, etc. on all trays). Once the design or discrete variables are fixed, the optimal operation of a given column configuration becomes a nonlinear programming (NLP) problem since then it only involves continuous variables. These NLP problems tend to be highly nonlinear and nonconvex due to the nature of the equations of conservation of energy and phase equilibrium.

In recent years, there has been some work on the application of full space [7,8,9] and decomposition methods [2,11] of *successive quadratic programming* (SQP) to distillation. In particular, A. Kumar and A. Lucia [6] proposed a full space SQP method based on thermodynamically consistent quasi-Newton formulae that exploit the homogeneity of the second derivatives of the energy balance and phase equilibrium equations. These thermodynamically consistent quasi-Newton updates were used to build appropriate parts of the Hessian matrix of the Lagrangian function and shown to result in better numerical performance than traditional secant updates on a number of distillation examples. Lucia and J. Xu [8] developed an indefinite quadratic programming method based on *Bunch and Parlett factorization* [3] of the entire coefficient matrix of the Kuhn–Tucker conditions, an active set strategy, and the use of trust regions to address the strong constraint nonconvexities inherent in distillation optimization problems.

Results from this work showed that permitting indefiniteness often provides a better local quadratic model for the Lagrangian function and that the resulting algorithms were capable of easily solving distillation examples with which [7] had difficulty. In [9] a refined active set strategy, constrained pivoting and numerical matrix factor updating for indefinite quadratic programming were proposed and good numerical performance was obtained for a family of SQP methods on a set of 15 distillations, some of which were extractive and azeotropic distillations.

In contrast, L.T. Biegler and C. Schmid [2,11] have applied range and null space decomposition (RND) SQP methods to distillation optimization problems. They 'tailor' an RND method for use with an existing simulation model through the use of an interface in order to illustrate that decomposition SQP methods can be easily applied to process models like distillation. In particular, [2] reports good numerical performance for a set of four distillation examples involving ideal binary and ternary mixtures, along with a discussion of issues such as the need for preprocessing and quadratic programming constraint infeasibility. See [2] for a discussion of the need for and ways in which range space curvature can be obtained.

The issues that are important in the application of SQP methods to distillation systems include:

- formulation;
- the mathematical model;
- Hessian matrix approximations, sparsity and other exploitable properties;
- initialization procedures for the unknown variables and multipliers; and
- algorithmic and other implementation issues.

SQP Formulation

Distillation optimization problems usually contain between 100 and 500 unknown variables, roughly the same number of equality constraints and twice that number of inequality constraints. Thus the number of degrees of freedom is often small compared to the number of unknowns. As a result, both full space and decomposition SQP methods can be used and the question of which approach is better still remains open. Regardless of the approach, a general mathematical representation of the distillation optimization problem is given by

$$\begin{cases} \min & f(x) \\ \text{s.t.} & c(x) \le 0, \end{cases}$$

where x is a vector of unknown variables of length n which represents an estimate of the local minimum, $f(x)$ is a twice continuously differentiable objective function and $c(x)$, the vector of equality and/or inequality constraints, is also twice continuously differentiable and nonlinear. This constrained optimization problem is commonly recast in terms of the *Lagrangian function*, $L(x)$, defined by

$$L(x) = f(x) + \lambda^\top c(x) + \mu^\top c(x),$$

where λ and μ are vectors of Lagrange and Kuhn–Tucker multipliers associated with the equality and inequality constraints respectively. The Kuhn–Tucker conditions are solved iteratively using a recursive quadratic programming formulation to define the change in the unknown variables and multipliers. This iterative *quadratic programming subproblem*, which is based on a quadratic approximation to the Lagrangian function subject to linearized constraints, is given by

$$\begin{cases} \min & g(x_k)^\top \Delta x_k + \frac{1}{2} x_k^\top B_k \Delta x_k \\ \text{s.t} & c(x_k) + J(x_k)\Delta x_k \le 0, \end{cases}$$

where Δx_k is the change in the unknown variables, $g(x_k)$ is the gradient of the objective function, $J(x_k)$ is the Jacobian matrix of the constraint functions, B_k is some approximation to the true *Hessian matrix of the Lagrangian function*, $H(x) = H_f(x) + \sum \lambda_i H_{ci}(x) + \sum \mu_i H_{ci}(x)$, where $H_f(x)$ and $H_{ci}(x)$ refer to the true Hessian matrices of the objective function and ith constraint respectively. When *full space SQP methods* are used, this recursive quadratic programming problem is solved directly. When *range and null space decomposition* (RND) methods are used, the equality constraints are used to 'eliminate' variables and a 'reduced' quadratic programming problem, given by

$$\begin{cases} \min & (Z_k^\top g(x_k) \\ & \quad - Z_k^\top B_k [Y_k [J_E Y_k]^{-1} c(x_k)])^\top \Delta z_k \\ & + \frac{1}{2} z_k^\top (Z_k^\top B_k Z_k) \Delta z_k \\ \text{s.t.} & x_L + Y_k [J_E Y_k]^{-1} [c(x_k)] \le Z_k \Delta z_k \\ & \le x_U + Y_k [J_E Y_k]^{-1} [c(x_k)] \end{cases}$$

is solved. Here Y and Z represent bases for the range and null space of the Jacobian matrix respectively, $\Delta x_k = Y\Delta y_k + Z\Delta z_k$, where Δz_k and $\Delta y_k = -[J_E Y_k]^{-1}[c(x_k)]$ are the change in the unknown variables in the null space and range respectively, J_E is the Jacobian matrix of the equality constraints, $(Z_k^\top B_k Z_k)$ is the symmetric projection of the Hessian matrix of the Lagrangian function onto the linearized constraints, and x_L and x_U are lower and upper bounds on the x variables.

Mathematical Model

The objective function in distillation optimization can be linear or nonlinear while the equality constraints are usually a mixture of highly nonlinear and linear algebraic equations. The inequality constraints, on the other hand, are simple bounds on variables but other nonlinear inequalities can occur.

Objective Function

The objective function for a typical distillation optimization is usually some function that represents a balance between the energy-related (or other operating) costs of the column and the profit obtained from the sale (or credits) of products. One example, taken from [7], might be

$$\min f = c_1(-Q^{\mathrm{C}} + Q^{\mathrm{R}}) - (v_{\mathrm{lk}}^{\mathrm{C}} + l_{\mathrm{hk}}^{\mathrm{R}})$$

where Q^{C} and Q^{R} are the condenser and reboiler heat duties (or energy demands) respectively and v_{lk}^{C} and l_{hk}^{R} are the component flow rates of the overhead and bottoms products. Here the subscripts lk and hk denote the light key component (or low boiling component) and heavy key component (or high boiling component) respectively, the superscripts C and R denote the condenser (or top stage) and reboiler (or bottom stage) of the column and c_1 is a scaling factor that helps balance the scale between the energy costs and product flow variables. Usually the energy demands consist of cooling water requirements for the condenser and the steam demands for the reboiler. Moreover, the negative sign in the condenser duty in the above objective function merely accounts for the thermodynamic convention associated with heat transfer and does not represent subtraction of the condenser duty because the value of Q^{C} is always negative. Other objective functional forms exist as well.

Equality Constraints

The equality constraints in distillation optimization consist of the conservation of mass and energy as well as the *phase equilibrium equations*, the latter of which relates the composition of the vapor to that of the liquid leaving each stage. The equations for the jth equilibrium stage are the component *mass balances*

$$f_{ij} + l_{i,j-1} - l_{ij} - v_{ij} + v_{i,j+1} = 0, \qquad i = 1, \ldots, n_c,$$

the phase equilibrium relationships

$$K_{ij}\frac{l_{ij}}{\sum l_{kj}} - \frac{v_{ij}}{\sum v_{kj}} = 0, i = 1, \ldots, n_c.$$

and the conservation of energy

$$\left(\sum f_{ij}\right) H_{fj} + \left(\sum l_{i,j-1}\right) H_{j-1} - \left(\sum l_{ij}\right) H_j - \left(\sum v_{ij}\right) h_j + \left(\sum v_{i,j+1}\right) h_{j+1} + Q_j = 0.$$

In these equations, l_{ij} and v_{ij} are the flow rates of component i in the liquid and vapor respectively on the jth stage, K_{ij} is the equilibrium ratio (or K-value) for the ith component on the jth stage, H_j and h_j are the corresponding liquid and vapor enthalpies, f_{ij} is the ith component feed flow rate to the jth stage, Q_j is the heat duty to the jth stage and n_c is the number of components. Moreover, the equilibrium ratios, K_{ij}, and enthalpies, H_j and h_j, are strongly nonlinear functions of the component flow rates, temperature, T_j, and pressure, p_j on the jth stage. Finally, for a column with n_s equilibrium stages, there are $n_s(2n_c + 1)$ equality constraints and $n_s(2n_c + 2)$ unknown variables (i. e., the l_{ij}s, v_{ij}s, T_js and Q_js). Usually, the pressure on each stage is fixed in some manner so pressure is not an unknown variable. Thus there are n_s degrees of freedom. However, it is usual, but not strictly necessary, that adiabatic operation (i. e., no heat withdrawn or added) for all trays except the top and bottom tray is assumed. This gives the additional equality constraints

$$Q_j = 0, \quad j = 2, \ldots, n_s - 1,$$

and results in just two degrees of freedom for the simplest and most common column configuration. Columns with heat withdrawal or addition would necessarily have more degrees of freedom.

Inequality Constraints

The inequality constraints in distillation optimization problems are usually comprised of simple bounds on variables and product flows. In particular, for the *j*th equilibrium stage there are nonnegativity bounds on the component flow rates

$$l_{ij}, v_{ij} \geq 0, \quad i = 1, \ldots, n_c,$$

upper and lower bounds on the temperature

$$T_{\min} \leq T_j \leq T_{\max}$$

in order to keep the calculations of K_{ij}, H_j and h_j physically meaningful, and in some cases explicit bounds on product component flow rates

$$l_{i,n_s} v_{i1} \leq \sum f_{ij}, \quad i = 1, \ldots, n_c,$$

to ensure that the mass balance around the column is satisfied.

Lagrangian Hessian Matrix Approximations

The Hessian matrix of the Lagrangian function requires that second derivatives of the objective function and nonlinear constraints be approximated and this can be done in any number of ways. When full space SQP methods are used to solve distillation optimization problems, analytical second derivative, finite difference second derivatives, quasi-Newton approximations or a mixture of analytical and quasi-Newton derivatives of the objective function and constraints (hybrid methods) can be used. In contrast, when decomposition SQP methods are used [1], the modified Broyden–Fletcher–Goldfarb–Shanno (BFGS) update (see [10]) is usually used to approximate the projection of the Lagrangian Hessian matrix, $Z_k^\top B_k Z_k$, to avoid explicit representation of B_k and the computation of a matrix triple product.

In full space SQP methods, all techniques for estimating the Hessian matrix of the Lagrangian function can be put in the form

$$B_k = C(x_k) + A_k$$

where $C(x_k)$ is called the computed part of the Hessian matrix and is calculated from analytical derivatives and A_k, the approximated part, can be computed either analytically, from finite differences or from an appropriate quasi-Newton formula [6]. Note that the objective function and the phase equilibrium and energy balance constraints can have both computed and approximated parts, and in particular, the second derivatives of the phase equilibrium and energy balance constraints have a natural division into thermodynamically *ideal and nonideal ideal and nonideal* (or *excess*) parts. The ideal parts are readily available in analytical form and constitute much of $C(x_k)$ while the nonideal parts depend on 'models' for the activity coefficient and/or fugacity coefficient and excess enthalpy and are usually contained in A_k. Furthermore, because of the stagewise structure of distillation columns, both the computed and approximated parts are sparse and tend to be comprised of many small dense blocks. Thus quasi-Newton formulas such as the Powell-symmetric-Broyden (PSB) update can be used to build the second derivative approximations of each block [6,7]. Also, certain parts of the equilibrium ratios, K_{ij}, and enthalpies, H_j and h_j, are homogeneous functions of the unknown variables (i. e., the component molar flow rates) and this gives rise to other matrix constraints that can be exploited when building quasi-Newton approximations to the blocks of A. That is, the equilibrium ratio is commonly defined by

$$K_{ij} = \frac{\gamma_{ij} f_{ij}^0}{\phi_{ij} p_j},$$

where γ_{ij} is the liquid activity coefficient, f_{ij}^0 is the pure component fugacity and ϕ_{ij} is the vapor fugacity coefficient for the *i*th component on the *j*th stage. The liquid and vapor molar enthalpies, on the other hand, usually have the form

$$H_j = H_j^{\mathrm{ID}} + H_j^{\mathrm{E}},$$
$$h_j = h_j^{\mathrm{ID}} + h_j^{\mathrm{E}},$$

where the superscripts ID and E denote the ideal and excess (or nonideal) parts respectively. The functions $\ln \gamma_{ij}$ and H_j^E are homogeneous functions of the liquid component molar flow rates on stage j while $\ln \phi_{ij}$ and h_j^E are homogeneous functions of the vapor component molar flow rates from the *j*th stage. Because these thermodynamic properties are homoge-

neous functions, they give rise to the matrix conditions

$$[\nabla^2 \ln \gamma_{ij}]l_j = -\ln \gamma_{ij} \quad \text{and} \quad [\nabla^2 H_j^{\mathrm{E}}]l_j = 0,$$

for the liquid phase on the jth stage and

$$[\nabla^2 \ln \phi_{ij}]v_j = -\ln \phi_j, \qquad [\nabla^2 h_j^{\mathrm{E}}]v_j = 0,$$

for the vapor phase on stage j. In [6] it is suggested that these thermodynamic constraints be used, in conjunction with traditional secant conditions, to build better approximations of the appropriate blocks of A_k and use one iterated projection [4] from the space of secant matrices to the space of thermodynamically constrained matrices to approximate the second derivatives of the activity coefficients, fugacity coefficients and excess enthalpies. In [9] a variety of techniques was used for approximating the blocks of A_k, including a partial Newton strategy in which each block of A_k is zero at each SQP iteration, a secant only hybrid (SOH) method in which each block of A_k is approximated by the PSB formula and only associated secant information for that block, a thermodynamically constrained hybrid (TCH) method in which both secant and thermodynamic constraints are used in conjunction with iterated projections for each block of A_k, and Newton's (or Wilson's [12]) method, in which analytical or finite difference second derivatives of each block of A_k are used.

Biegler et al. [2,11] use the modified BFGS formula exclusively to approximate the projection of the Lagrangian Hessian matrix on the tangent subspace defined by the linearized constraints in RND SQP methods. Curvature information in the range space is generally neglected; however [2] suggests the use of and techniques for obtaining range space curvature.

Initialization of the Unknown Variables and Multipliers

The initialization of the unknown variables and Lagrange and Kuhn–Tucker multipliers is an extremely important aspect of the successful implementation and application of SQP methods to distillation optimization, regardless of whether full space or decomposition SQP methods are used, and can represent the difference between success and failure in problem solving. 'Good' initial values of the unknowns and multipliers often prevent infeasible quadratic programming subproblems. In many cases, a base design or simulation is available, in which the equality constraints are solved for a given set of specifications for the column (i. e., additional (usually two) equality constraints that exhaust the number of degrees of freedom). See [9]. This base case simulation provides both feasible and qualitatively correct initial estimates of the unknown variables and, while feasibility is not strictly necessary, it does usually significantly improve the numerical performance of full space SQP methods on distillation optimization problems. This is because much of the strong nonlinearity in distillation optimization is contained in the equality constraints (i. e., phase equilibrium and energy balance equations) and feasible starting points usually result in iterates that track the constraint surface more closely than infeasible starting points. Both full space and decomposition SQP methods benefit from feasible starting points. The base case simulation also identifies any active inequalities at the feasible starting point.

Decomposition methods for distillation optimization can also make use of simulations to initialize the unknown variables. However, [11] uses an initialization procedure from the simulation program to give an infeasible but linearly consistent starting point for distillation optimization problems solved by the 'tailored' RND SQP method.

Good initial estimates of the Lagrange and Kuhn–Tucker multipliers are also important to the application of SQP methods to distillation optimization. In [6] all initial Kuhn–Tucker multipliers are set to zero (unless the base case simulation suggests otherwise) and the Lagrange multipliers are initialized by solving the equations

$$J_E J_E^{\top} \lambda = -J_E g(x_k),$$

where again J_E is the part of the Jacobian matrix corresponding to the equality (or active) constraints. Because the number of equality constraints can be large and because the sparsity of $J_E J_E^{\top}$ need not be anything like the sparsity of J_E, In [6] it is suggested that the above equation be solved for a small 'model' column consisting of a condenser (top stage), reboiler (bottom stage), all feed stages and one stage between the condenser, reboiler and each feed tray and then the resulting multiplier values be distributed by equation-type and section throughout the actual larger column. Thus for a 'conventional' column with one feed stage, there is one rectifying stage (between the feed and the con-

denser) and one stripping tray (between the feed and the reboiler) and five total stages. Lagrange multipliers for this 'model' column are calculated and then distributed by equation-type within different sections in the column. That is, the Lagrange multipliers for the mass balance, energy balance and phase equilibrium equations for the condenser, reboiler and all feed stages are assigned their values calculated for the model column while the Lagrange multipliers for the mass balance, energy balance and phase equilibrium equations in the rectifying section are all assigned the same respective values calculated for the single rectifying tray in the model column. The same exact distribution procedure is used for all stages in the stripping section of the column.

Initial estimates of the the Lagrange multiplier in decomposition SQP methods are not strictly required since the initial projected Hessian matrix, $Z_k^\top B_k Z_k$, does not strictly require these values to be known. Moreover, the Lagrange multipliers, as well as any active Kuhn–Tucker multipliers, can be easily calculated from the relationship

$$Y_k^\top J_E^\top \lambda_k = -Y_k^\top g(x_k)$$

once the unknown variables have been initialized.

Algorithmic and Other Implementation Issues

The sparsity of the constraint Jacobian matrix, characteristics of the resulting quadratic programming subproblems and the use of stabilization techniques such as line searching are also important in assembling the correct set of computer tools for the application of SQP methods to distillation optimization problems.

The Jacobian matrix for the equality constraints in distillation optimization usually has a block tridiagonal structure, unless there are pumparounds. The inequality constraints, on the other hand, result in a diagonal structure for their part of the constraint Jacobian matrix. As a result, both full space and decomposition SQP methods must exploit the sparsity of the Jacobian matrix in distillation problems to keep storage (i. e., fill-in) and computational effort (arithmetic operations) tractible. Exploiting the sparsity of the constraint Jacobian matrix is necessary in full space SQP method in order to effectively store the linear operators used in solving the large recursive quadratic programming subproblems that occur. That is, sparsity must be exploited in the matrix factorizations in active set methods or the natural operators in interior point methods for solving large quadratic programming problems. In contrast, the use of sparse matrix techniques significantly reduces both storage of the Jacobian matrix and the storage and computational effort required to form the (factors of the) matrices Y (i. e., the basis for the range), $[J_E Y_k]^{-1}$, and $Y_k[J_E Y_k]^{-1}$ in decomposition SQP methods. D. Goldfarb [5] provides a good set of general guidelines for many of the issues related to the sparsity of both the constraint Jacobian and Hessian matrices in recursive quadratic programming.

In decomposition SQP methods, the projection of the Lagrangian Hessian matrix is almost always approximated by the modified BFGS formula [10] and thus the 'reduced' quadratic programming subproblems are positive definite and have a unique solution at each SQP iteration. This is a significant advantage in some respects but the BFGS formula can give slower convergence than desired at the SQP level of the computations in distillation optimization because it often has difficulty tracking the strong curvature of the nonconvex constraint surface. In full space SQP methods, the projected Hessian Lagrangian matrix can be either positive definite or indefinite depending on the way in which it is approximated. If Levenberg–Marquardt or modified Schur complements are used in conjunction with sparse factorizations like Cholesky factorization, positive definiteness can be maintained and convex quadratic programs result. On the other hand, when a hybrid approach is used, the blocks of A_k usually have no sign definiteness in distillation optimization because the phase equilibrium and energy balance equations are nonconvex and strongly nonlinear. Thus the resulting (projection of the) Hessian matrix of the Lagrangian function can be indefinite and indefinite quadratic programs result, which can be difficult to solve.

Stabilization techniques such as line searching [2] and trust region methods [8,9] have been used in distillation optimization. Biegler [2] suggests the use of line searching techniques such as the Armijo rule (see [10]) but gives few details on associated numerical performance. Lucia et al. [9] recommend the use of 'asymmetric' trust region methods in distillation optimization to improve numerical performance and also alleviate difficulties associated with infeasible quadratic programs.

Comments on the Numerical Performance of SQP Methods on Distillation Problems

There are a limited number of papers in the literature on the optimization of distillation systems using SQP methods. Lucia and Kumar [6] minimized the operating costs of a methanol recovery column with 10 equilibrium stage and two components using SQP methods in which the Lagrangian Hessian matrix was approximated by a partial Newton method, SOH and TCH methods and all analytical second derivative (i. e., Wilson's method). They report failure for all methods except the thermodynamically constrained hybrid method on this relatively small problem involving 50 unknown variables. They [7] subsequently applied the same SQP methods, with the exception of the partial Newton method, to a set of five distillation examples ranging in size from 35 to 176 unknown variables and report good numerical performance for the thermodynamically constrained hybrid method. In particular, they discuss the advantages of using feasible starting points for the unknown variables and 'good' qualitative estimates of the Lagrange multipliers. In [8] the need for feasible starting points is reiterated and also contains a discussion of the occurrence of line searching difficulties and uphill search directions in the examples studied in [7]. See [9] for the numerical performance of the partial Newton, SOH, TCH, Wilson and range and null space decomposition (RND) SQP methods on a set of 15 examples, some of which contain strongly nonideal (and therefore nonlinear) extractive and azeotropic distillations. In this study, most methods performed quite well with a slight advantage going to Wilson's method over the TCH method. The RND method performed worst of all on this set of examples, followed by partial Newton and then the SOH and TCH methods in terms of reliability and efficiency. See [9] also for the failure of line searching techniques such as Armijo's rule and an augmented Lagrangian line search function as well as the occurrence of infeasible linearized constraint sets and the usefulness of 'asymmetric' trust regions in forcing convergence when difficulties arise. See, on the other hand, [11] for numerical results for two binary distillations and two ternary distillations, ranging in size from 60 to 252 unknown variables. Few numerical details are presented with regard to the physical properties models used, although the mixtures studied can be considered ideal, and some discussion of infeasible quadratic programs from 'poor' starting points is given.

See also

- ▶ Feasible Sequential Quadratic Programming
- ▶ Optimization with Equilibrium Constraints: A Piecewise SQP Approach
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Successive Quadratic Programming
- ▶ Successive Quadratic Programming: Applications in the Process Industry
- ▶ Successive Quadratic Programming: Decomposition Methods
- ▶ Successive Quadratic Programming: Full Space Methods
- ▶ Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Berna TJ, Locke MH, Westerberg AW (1980) A new approach to optimization of chemical processes. AIChE J 26:37–43
2. Biegler LT (1992) Tailoring optimization algorithms to process applications. Comput Chem Eng 16S:81–95
3. Bunch JR, Kaufman L (1980) A computational method for indefinite quadratic programming problems. Linear Alg Appl 34:341–370
4. Dennis JE, Schnabel RB (1979) Least change secant updates for quasi-Newton methods. SIAM Rev 21:443–459
5. Goldfarb D (1975) Matrix factorizations in optimization of nonlinear functions subject to linear constraints. Math Program 10:1–31
6. Kumar A, Lucia A (1987) Separation process optimization calculations. Appl Numer Math 3:409–425
7. Lucia A, Kumar A (1988) Distillation optimization. Comput Chem Eng 12:1263–1266
8. Lucia A, Xu J (1990) Chemical process optimization using Newton-like methods. Comput Chem Eng 14:119–138
9. Lucia A, Xu J, D'Couto GC (1993) Sparse quadratic programming in chemical process optimization. Ann Oper Res 42:55–83
10. Powell MJD (1978) A fast algorithm for nonlinearly constrained optimization calculations. In: Watson G (ed) Numerical analysis (Dundee, 1977). Lecture Notes Math. Springer, Berlin, pp 144–157
11. Schmid C, Biegler LT (1992) Quadratic programming methods for tailored reduced Hessian SQP. Comput Chem Eng 12:1087–1095

12. Wilson RB (1963) A simplicial method for concave programming. PhD Thesis, Harvard University

Successive Quadratic Programming: Applications in the Process Industry

DILEK ALKAYA[1], SRIRAM VASANTHARAJAN[2], L. T. BIEGLER[1]
[1] Chemical Engineering Department, Carnegie Mellon University, Pittsburgh, USA
[2] Upstream Strategic Research Center, Mobil Technology Company, Dallas, USA

MSC2000: 90C30, 90C90

Article Outline

Keywords
Introduction
Successive Quadratic Programming
Reduced Hessian SQP Methods
Multiplier-Free Reduced Hessian SQP
Solving the QP Subproblem
Modeling Modes for Process Flowsheets
  Modular (Closed Form) Approach
  Equation Oriented (Open Form) Approach
Application of SQP Optimization in Industrial Problems
On-line Process Optimization
Off-Line Process Optimization
Conclusions And Future Work
See also
References

Keywords

SQP; Model/optimizer coupling; Modular; Equation-based; Tailored optimization; rSQP; Flowsheet optimization

This article discusses the use of successive quadratic programming (SQP) in industry, together with techniques for mathematical optimization and process modeling to improve economic performance of plants in process industries. First, different types of flowsheeting optimization problems based on SQP methods and process models are introduced briefly. Then a number of process optimization formulations and strategies are discussed, along with how the SQP algorithm needs to be developed and extended to take advantage of large scale systems. In particular, the development of reduced Hessian SQP (rSQP) is presented along with different variants. Finally, literature on industrial and academical applications of SQP and rSQP is given.

Introduction

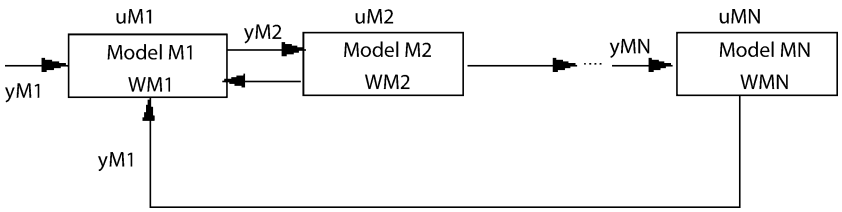
Complex engineering models can be formulated as large systems of differential and algebraic equations, constructed by linking smaller submodels. These complex engineering models can constitute a larger system which leads to flowsheet optimization. This optimization problem can be posed as considering a general problem with different models and connections shown in Fig. 1 schematically.

The mathematical representation of the flowsheet optimization and simulation problem shown symbolically in Fig. 1 can be written as

$$\begin{cases} \min & \sum_i f(w_{Mi}, y_{Mi}, u_{Mi}) \\ \text{s.t.} & M_1(w_{M1}, y_{M1}, u_{M1}) = 0 \\ & \quad \vdots \\ & M_n(W_{Mn}, y_{Mn}, u_{Mn}) = 0 \\ & C(w_{Mi}, y_{Mi}, u_{Mi}) = 0 \\ & \quad \text{for } i = 1, \ldots, \#\text{units} \\ & w \in W, \quad y \in Y, \quad u \in U, \end{cases} \tag{1}$$

where M_i are the chemical process models that can be solved with specialized solution strategies. Also, w_{Mi} are the internal variables inside of each model M_i; y_{Mi} are the input stream variables, and u_{Mi} are the decision variables. Here $C(w, y, u) = 0$ includes the additional constraints that arise from coupling of models and the sets W, Y and U represent lower and upper bounds on their respective variables.

In these optimization problems, the overall equation system usually results in very large systems of algebraic equations and variables (typically, $10^4 - 10^6$) with relatively few degrees of freedom (typically, < 100). The solution and optimization of these models are frequently effected by calculation procedures that exploit their equation structure. Because it requires fewer iterations and because of its flexibility in interfacing to process models, successive quadratic programming has



Successive Quadratic Programming: Applications in the Process Industry, Figure 1
General representation of flowsheet optimization problem

arguably become the most popular method for solving these nonlinearly constrained optimization problems.

The objective of this article is to characterize usage of SQP in different techniques for process flowsheeting optimization problem and to give an overview on industrial applications of these techniques. We also point out some remaining difficulties which still prevent application of simultaneous optimization for very large systems. In Section 2 we present the SQP algorithm and discuss how this is interfaced to process models. Section 3 discusses some large scale extensions to SQP and focuses on the rSQP technique. Here, the flowsheeting modes, modular and equation oriented (EO), are described in Section 4 in order to provide more detail on process optimization problems. This is followed in Section 5 with some industrial examples for both off-line and on-line optimization. Conclusions and directions for future research are then given in Section 6.

Successive Quadratic Programming

In this section we examine the underlying ideas of the SQP method and the theory that establishes it as a framework from which effective algorithms can be derived. In addition, an excellent review of the development of the SQP algorithm can be found in [9]. For process optimization we describe the most popular manifestations of the method, discuss the theoretical properties, and comment on their practical implementations. The nonlinear programming problem to be solved can be formulated as

$$(NLP)\begin{cases}\min & f(x)\\ \text{s.t} & c(x)=0\\ & x^l \le x \le x^u\end{cases}$$

where the objective function f: $\mathbf{R}^n \to \mathbf{R}$, equality constraints c: $\mathbf{R}^n \to \mathbf{R}^m$ and any nonlinear inequality constraints can be expressed through simple bounds and additional equality constraints by adding slack variables. Here the great strength of the SQP method is its ability to solve problems with nonlinear constraints. For this reason, it is assumed that (NLP) contains at least one nonlinear constraint function.

The basic idea of SQP is to model (NLP) at a given point, say x_k, by a quadratic programming subproblem, and then to use the solution to this subproblem to construct a better approximation x_{k+1}. This process is iterated to create a sequence of approximations that, it is hoped, will converge to a solution x^*. The key to understanding the performance and theory of SQP is the fact that, with an appropriate choice of quadratic subproblem, the method can be viewed as the natural extension of Newton and quasi-Newton methods [16] to the constrained optimization setting. Thus one would expect SQP methods to share the characteristics of Newton-like methods, namely, rapid convergence when iterates are close to the solution, but it is possible to have erratic behavior that needs to be carefully controlled when iterates are far from a solution. While this correspondence is valid, in general, the presence of constraints makes both the analysis and implementation of SQP methods significantly more complex.

Two additional properties of the SQP methods should be pointed out. First, SQP is not a feasible-point method; that is, neither the initial point nor any of the subsequent iterates need to be feasible (a feasible point satisfies all of the constraints of (NLP)). This is a major advantage since finding a feasible point for nonlinear constraints may be nearly as hard as solving (NLP) itself. SQP methods can be easily modified so that linear constraints, including simple bounds, are always satisfied. Second, the success of the SQP methods depends on the existence of rapid and accurate algorithms for solving quadratic programs. Fortunately, there are

good procedures for solving quadratic programs. Indeed, when there are only equality constraints, the solution to a quadratic program reduces to the solution of a linear system of equations. When there are inequality constraints a sequence of these systems is to be solved, in principle.

A successful SQP algorithm also needs adaptive safeguards that deal with general problems. The algorithmic details to overcome such difficulties, as well as more mundane questions – how to choose parameters, how to recognize convergence, and how to carry out the numerical linear algebra – are lumped under the term 'implementation'. Some description of SQP implementations for SQP is provided in [6].

The basic algorithm for the SQP method can be summarized as follows:

	For each iteration k:
1	Evaluate the objective and functions, $f(x_k)$ and $c(x_k)$ and their gradients
2	Solve a quadratic programming (QP) problem to determine a search direction, d_k for the variables, x_k. If a termination criterion is satisfied (i.e., x_k is a KKT point), STOP.
3	Find a steplength that leads to a sufficient improvement toward the solution of (NLP).This is done either by a trust region or a line search algorithm. In the case of a line search, set $x_{k+1} = x_k + \alpha_k d_k$, where α_k is a steplength parameter. For the trust region method, we constrain $d_k \in \triangle$, where $\triangle$ is adjusted and $x_{k+1} = x_k + d_k$.

Basic algorithm for the SQP method

To consider two components of this algorithm, the QP subproblem for problem(NLP) method can be formulated as follows

$$\begin{cases} \min\limits_{d\in\mathbb{R}^n} & g(x_k)^\top d + \frac{1}{2} d^\top W(x_k) d \\ \text{s.t.} & c(x_k) + A(x_k)^\top d = 0 \\ & x^L \le x_k + d \le x^u \end{cases} \tag{2}$$

where g denotes the gradient of f, $W(x)$ denotesthe Hessian of the Lagrangian function, $L(x, \lambda) = f(x) + \lambda^\top c(x)$ and A denotes the $n \times m$ matrix of constraint gradients,

$$A(x_k) = [\nabla c_1(x_k), \ldots, \nabla c_m(x_k)].$$

To establish global convergence for constrained optimization algorithms, i. e., convergence to KKT points from poor starting points, a way of measuring progress towards a solution is needed. For SQP this is done by constructing a merit function, a reduction in which implies that an acceptable step has been taken. For determination of the steplength, either with a trust region or line search method, a merit function is used to balance the two goals of decreasing the objective function and satisfying the constraints of the nonlinear program (NLP). Choices for the merit function include the nondifferentiable ℓ_1 merit function

$$\varphi_\mu(x) = f(x) + \mu \|c(x)\|_1 \tag{3}$$

from [24], and the augmented Lagrangian function

$$\varphi_\mu(x) = f(x) + \lambda(x)^\top c(x) + \mu \frac{\|c(x)\|^2}{2} \tag{4}$$

from [18].

Using these components, S.P. Han [24] proved that if a line search stepsize, α_k, is chosen by decreasing an exact penalty function along the QP computed search direction d_k, and the QPs are convex, solvable and bounded below, then the SQP algorithm converges to a KKT point from any starting point, implying global convergence. However, employing this line search function often led to very small stepsizes, and consequently, slow convergence rates in the neighborhood of the solution. M.J.D. Powell [34] modified this procedure to introduce a less stringent line search function. However, this function was neither globally convergent, nor locally superlinear. Several researchers have considered the line search strategies for SQP. K. Schittkowski [37,38] and H. Yamashita [48] proposed an augmented Lagrangian merit function for the line search.

R.H. Byrd and J. Nocedal [11] give an analysis of the two merit functions and their convergence properties for a reduced Hessian algorithm. The augmented Lagrange function has been used widely but its performance is sensitive to the multiplier estimates and the penalty parameter, μ. On the other hand, the ℓ_1 merit function can suffer from the Maratos effect (slow convergence) near the solution, although a nonmonotonic

line search such as watchdog technique [13] can be used to avoid this effect. This merit function also has the advantage of not requiring estimates of multiplier values at each iteration, even though the penalty parameter is usually based on Lagrange multiplier estimates. For this reason, in [8] a simpler measure is considered that does not require Lagrange multiplier estimates, but still maintains descent properties for $\varphi_\mu(x)$ as discussed later.

Finally, Byrd and Nocedal [11] summarize and extend local convergence properties of SQP. In particular, if full steps are taken in the neighborhood of the solution, a variety of superlinear convergence rates can be classified and these depend on how $W(x)$ is calculated or approximated.

Efficient SQP algorithms in the large scale case depend on carefully addressing many factors. Problems are considered large if, to solve them efficiently, either their structure must be exploited or the storage and manipulation of the matrices involved must be handled in a special manner. The most obvious structure, and the most commonly considered, is the sparsity of the matrices. Typically in large scale problems most constraints depend only on a few of the variables and the objective function is 'partially separable', i. e., it is made of a sum of functions each of which depends only on a few of the variables. In such cases matrices are sparse.

The formulation of the problem in terms of SQP and QP usage is the same as shown in (2). But the solution strategy and passing of information about the Hessian make a significant difference. For the SQP method described above, the Hessian matrix is usually supplied in a dense form, since it is frequently approximated by a quasi-Newton updating formula. The search direction is determined by using the dense Hessian approximation. When the problem becomes very large, passingthis information and solving the QP can become prohibitively expensive.

For large scale process optimization problems, we can distinguish twosignificant kinds of SQP algorithms, full space and reduced space methods. The first approach applies sparse, full space QP solvers, where natural problem structure can be exploited [28] based on analytic first and second derivative matrices. For large process models, however, second derivatives may be difficult to obtain and there are generally few decision variables (i. e., degrees of freedom) despite the large model size. As a result, we therefore consider a reduced space SQP (rSQP) decomposition strategy, as described in the next section.

Reduced Hessian SQP Methods

The reduced Hessian methods approximate only the portion of the Hessian relevant to a subspace of the variables. The advantages of these methodsare that quasi-Newton positive definite updates can be used and that the dimension of the problem is reduced to $n - m$ (possibly a significant reduction). Several versions of a reduced Hessian type of algorithm have been proposed; they differ in the ways the multiplier vectors are chosen and the way the reduced Hessian approximation is updated.

In rSQP, the quadratic programming (QP) subproblem is reduced to solving a smaller QP in the space of the independent variables by introducing a nonsingular matrix of order n, which consists of two basis matrices and is written as $[Y_k\ Z_k]$, where $Y_k \in \mathbf{R}^{n\times m}$, $Z_k \in \mathbf{R}^{n\times(n-m)}$ and it is assumed that $A_k^\top Z_k = 0$. Thus the Z_k matrix becomes a basis for the tangent space of the constraintsand the solution can be expressed as

$$d_k = Y_k p_Y + Z_k p_Z \tag{5}$$

for vectors $p_Y \in \mathbf{R}^m$ and $p_Z \in \mathbf{R}^{n-m}$. From (5) the linear constraint defined in (2) becomes

$$c_k + A_k^\top Y_k p_Y = 0. \tag{6}$$

If A_k is assumed to have full column rank then the nonsingularity of $[Y_k\ Z_k]$ implies that the matrix $A_k^\top Y_k$ is nonsingular, so that p_Y can be determined by (7) as:

$$p_Y = -[A_k^\top Y_k]^{-1} c_k. \tag{7}$$

Loosely speaking, the p_Y step serves to improve the solution of the equality constraints, while p_Z acts in the null space of these constraints and serves to minimize the objective function.

There are a number of ways to form the basis vectors Y and Z. One of the cheapest ways, that is well-suited to large scale decomposition, is to partition the variables, x, into m basic or dependent variables (which are reordered to be the last m variables) and $n - m$ decision variables. This induces the partition

$$A(x)^\top = [N(x)\ C(x)], \tag{8}$$

where $m \times m$ basis matrix $C(x)$ is assumed to be nonsingular. $Z(x)$ and $Y(x)$ are now defined to be

$$Z(x)^\top = [I| - N^\top C^{-1\top}], \quad Y^\top = [0|I]. \tag{9}$$

This choice is particularly popular [18,32] and advantageous when $A(x)$ is large and sparse, because a sparse LU decomposition of $C(x)$ can often be computed efficiently.

If the variables are partitioned into independent (z_I) and dependent (z_D) variables, then the corresponding search direction can be defined as follows:

$$d^\top = [d_\mathrm{I}^\top \; d_\mathrm{D}^\top], \tag{10}$$

$$d_\mathrm{I} = p_z, \tag{11}$$

$$d_\mathrm{D} = p_y - C^{-1} N d_I, \tag{12}$$

where p_y corresponds to a Newton step for the m dependent variables and m equality constraints and p_z is computed by solving a much smaller QP subproblem than the original problem, as given below. The QP subproblem can then be expressed exclusively in terms of the variable p_Z.

$$\begin{cases} \min & (Z_k^\top g_k + w_k)^\top p_Z + \frac{1}{2} p_Z^\top B_k p_Z \\ \text{s.t.} & x_L - x_k - Y_k p_Y \le Z_k p_Z \\ & \quad \le x^U - x_k - Y_k p_Y \end{cases} \tag{13}$$

where the reduced matrices $Z^\top WZ$ and vector $Z^\top WYp_Y$ are given (or approximated) by B_k and w_k, respectively. This decomposition reduces the Hessian matrix from order n to order $n - m$ but we are still left with $n - m$ simple bounds on the variables p_Z and m bounds from the dependent variables, which are projected into the space of the independent variables. Details of the reduced Hessian SQP can be found in [11,27] and in [8,39].

Multiplier-Free Reduced Hessian SQP

In the conventional rSQP method (see, e. g., [11]), Lagrange multipliers are calculated by

$$\lambda = -(Y^\top A)^{-1} Y^\top (g_k + \eta_k), \tag{14}$$

where η_k are the bound multipliers. In process optimization models where the model equations, variables and the constraint gradient matrices *are not accessible directly*, we can develop a nonlinear programming method that *requires neither second derivative nor calculates Lagrange multiplier estimates for the model equations*. In [7] this condition is taken into account, the 'multiplier free' reduced Hessian algorithm is derived and is presented formally for problem (NLP). In this approach, f and c are assumed to be smooth functions with $n, m \gg n - m$ and the first derivatives of the f and c are available. The SQP method for solving equations (7)–(13) generates, at iterate x_k, a search direction d_k by solving the QP subproblem with an exact penalty linesearch method. Generally the condition $\mu_k > ||\lambda_k||$ is used to ensure a descent property [11]. Instead, the multiplier-free approach can be used by noting that

$$\lambda^\top c_k = (g_k + \eta_k)^\top Y_k p_Y. \tag{15}$$

So to ensure a descent property, one only needs to choose:

$$\mu_k > \frac{\left|(g_k + \eta_k)^\top Y_k p_Y\right|}{\|c_k\|}. \tag{16}$$

Finally, current SQP methods incorporate either trust region or line search strategies to promote global convergence behavior. The line search is more efficient in determining proper steplengths while the trust region is essential to avoid poor search directions. Because of the trade-offs in using either method, D. Ternet and L.T. Biegler [42] incorporated a combined line search and trust region approach. Details for the application of trust region and line search methods can be seen from [42].

Solving the QP Subproblem

At the heart of the SQP optimization algorithm is the formulation and solution of the quadratic program (QP1). This step strongly influences theaccuracy and efficiency of the algorithm. Aside from the effort required to evaluate the function values and gradients, this step is usually the most computationally intensive step in the SQP or rSQP algorithm.

QP algorithms based on active set strategies can be classified into primal and dual approaches. In the primal space approach, a feasible point is determined first

and succeeding directions are then taken to reduce the quadratic objective function. This approach requires a positive definite projected Hessian. An early primal code was VE02AD, developed by R. Fletcher [18] and incorporated in the Harwell subroutine library in 1972 [25]. A popular and very reliable primal QP code, QPSOL, was developed by P.E. Gill and W. Murray [21], who also extended this strategy in the codes LSSOL (1988) and QPOPT (1996).

A very efficient dual space QP strategy was developed by D. Goldfarb and A. Idnani [22]. Here, we require a positive Hessian matrix, but no initial feasible point is required, instead a dual feasible point is first calculated. This can save considerable effort in the SQP or rSQP algorithms. This approach was incorporated into two QP codes, the Harwell code-VE17AD by Powell and QPKWIK [39]. In addition, QPKWIK allows for direct updating of the inverse Cholesky factor of the reduced Hessian matrix. This and other features within-QPKWIK allow the reduced Hessian method to perform better than both QPSOL and VE17AD, as $n- m$ becomes larger.

Modeling Modes for Process Flowsheets

Using the process models described in (1) and in Fig. 1, thereare three problem types, *simulation*, *design* and *optimization problems*, frequently considered by process engineers. In the simulation problem, the variables associated with the feed streams and the design variables (u in the constraints in (1)) of the units are specified. The unknowns are the remaining variables (y and w in the constraints in (1)). In this procedure, it is implicitly assumed thatthe number of variables to be determined is equal to the number of equations, so that the system is solved; any adjustment of remaining decision variables can be left to an outer optimization loop. In addition, design problems require the specification of additional constraints (such as production rates, product yields and purities) in the flowsheet and freeing up additional decision variables (u) to satisfy these constraints.

In the optimization problem for process flowsheets, variables associated with the feed streams and design variables may be left unspecified and a cost function is added to the model in (1). The unspecified variables (u) are determined so as to minimize the cost function. In this case, both equality and inequality constraints may be present and their number may be different from the number ofthe unspecified parameters. At the simplest (and least efficient) level, the optimization approach is an iterative procedure consisting of the following steps:

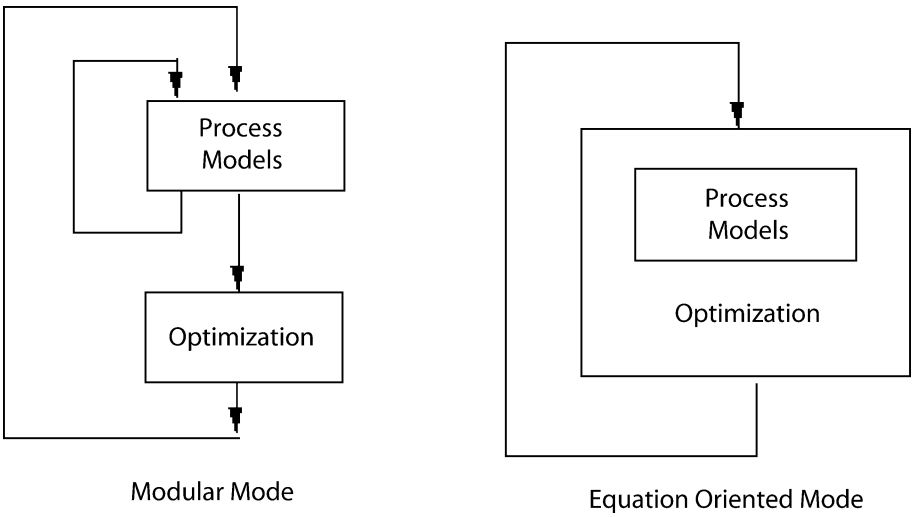
a) Fix $(n - m)$ degrees of freedom (independent variables, u)
b) Solve the m equations for the m remaining variables(w and y). This is the flowsheet simulation problem.
c) Evaluate the objective function (1), and adjust the $n- m$ variables to minimize this and satisfy the bounds in (1).
d) Repeat from step a).

Since the variables are determined by the solution of the equations in step b), the equations themselves are satisfied exactly within the convergence tolerance of the solution procedure. We may regard the equations as serving to eliminate m variables from the optimization problem. This approach treats the process model as a 'black box' and requires the repeated solution of the simulation problem. However, this approach can fail if decision variables are chosen at intermediate points, for which there is no solution to the simulation model.

The advantage to SQP is that it can be interfaced more flexibly than with the *black-box strategy* and it allows a simultaneous optimization and simulation capability. This can be seen by considering two different modes, the *modular* and *equation oriented approaches*, for model formulation and solution. These are illustrated in Fig. 2 and described next.

Modular (Closed Form) Approach

With this approach, the modeling equations are grouped according the individual units in the process and specialized solution strategies are applied to each unit. Calculation then proceeds sequentially from one unit to the next. With modular approach, there are many widely tested models and procedures. Solution procedures are unit specific and locally robust. Initialization is also straightforward. However, this modeling mode requires tearing of the recycle streams. Here one iterates on tear variables that are sufficient to permit the remaining variables to be calculated. Since the flowsheet consists of modular modules, recycle convergence is usually performed by slow convergence tech-



Successive Quadratic Programming: Applications in the Process Industry, Figure 2
Modeling modes and process optimization

niques. Moreover, the calculation of derivatives from individual process models involves perturbing and re-simulating the entire flowsheet with respect to the decision variables. This process is both time-consuming and subject to errors due to probable internal convergence failures during model solution stage.

The modular approach is the most common simulation technique employed in industrial environments for off-line design and analysis. For example in three common process simulation codes (ASPEN, PRO/II and HYSYS) the optimization problem, is solved by first calculating the process models before evaluating the constraints and objective function. This black-box technique is referred to as *feasible path* (FP) approach and represents a two-tiered strategy to optimization. The optimization problem is solved in an outer loop, while the simulation equations are converged in an inner loop. Note that recycle equations need to be solved every time the objective function is evaluated.

On the other hand, these tear stream equations and variables can also be added as constraints in the optimization problem, and this leads to a more efficient NLP strategy than with the black-box approach. Termed the *infeasible path approach*, this strategy performs convergence of the recycle loops simultaneously with optimization of the flowsheet. This capability has been added to a number of process simulators (see Table 2) and the optimizer enjoys some success in industry for the optimization of novel process and equipment designs. A detailed derivation of this approach along with description of several flowsheeting case studies is given in [6].

Equation Oriented (Open Form) Approach

With the equation oriented approach, the process model equations are considered as a single large set of equations to be solved with a large scale nonlinear algorithm. For process optimization, B.A. Murtagh [31] offered the viewpoint where the optimization is embedded within the solution procedure. Here the nonlinear equations describing the entire system become a set of nonlinear equality constraints, giving rise to a large nonlinear programming problem with a mixed set of large sparse linear and nonlinear constraints. To distinguish it from the modular or closed form approach it is named as the *simultaneous* or *open form approach*. This approach involves the simultaneous linearization of all the equations and iteration on all the variables, using the Newton–Raphson method or some variation thereof. In this case, we must be able to solve efficiently very large systems of sparse linear equations. In solving such systems the use of sparse matrix techniques is a necessity.

The EO approach, when used with multiple models, does not exploit individual model structure and the

entire burden for solution is on a general purpose Newton solver. Initialization might be difficult, but very efficient methods exist for the partitioning and tearing large sets of algebraic equations. Also, objective function and constraint derivatives are usually available analytically from the large system. Therefore, the advantage of the equation oriented approach is that it avoids multiple levels of iteration, one for solving the systems describing equations and one for optimization. On the other hand, the describing equations are not necessarily satisfied exactly until convergence is approached (although it is possible to allow for instances where this causes difficulty).

Several EO programs were developed, including ASCEND (Carnegie MellonUniv.), QUASILIN (Cambridge Univ.), FLOWSIM (Univ. Connecticut), and SPEEDUP (Imperial College). Given recent advances in software engineering and object oriented structures, equation oriented process simulation packages havealso been made commercially available (e. g. RTOPT, SPEEDUP, NOVA, gProms). However, because equation-based process models are harder to set up and initialize, these packages are generally more difficult to use than modular simulators.

With these two simulation modes, we see a number of trade-offs. The modular mode deals with large detailed models and convergence of the optimization problem occurs at multiple levels and can be time-consuming. However, initialization and problem formulation is generally easy and intuitive to the process engineer and the solution strategies are robust. Consequently, this approach is used as a general purpose optimization strategy for off-line design and analysis for large scale chemical processes.

On the other hand, the equation oriented strategy provides a truly simultaneous strategy to process optimization and can be much more efficient. Nevertheless, initialization and process modeling are somewhat specialized to the process application, and general-purpose detailed models may often be simplified. As a result, EO approaches are more common for on-line optimization, including refineries, olefin plants and power stations.

In the next section, we provide a brief history on the application ofSQP to these problem types. We then describe a number of examples for bothon-line and off-line process optimization.

Application of SQP Optimization in Industrial Problems

The first appearance of SQP can be traced back to [45] and [4], butnumerical difficulties hampered widespread application. In particular, this was due to conceptual weaknesses, such as lack of global convergence, nonconvex QPs and unreliable QP solvers. As a result there was little initial development of SQP until the late seventies. Nevertheless, in 1968, J.D. Simon developed a general purpose nonlinear optimizer within Exxon, based upon the successive solution of quadratic programming (QP) problems approximating the given problem. The program called ECO (Exxon Computerized Optimizer) was put into production status in 1969 and made available to Exxon's worldwide affiliates. In 1970, the code was revised to handle a gas field optimization problem which required over 300 variables for the Exxon Production Research Company. The code proved to be so successful that a special version has linked to a reservoir simulation system and marketed outside of Exxon. In 1972, a revised Exxon production version was released to incorporate additional features and be more user friendly. However, this software was not pursued during the 1980s due to lack of economic justification [40].

On the other hand, by 1977, the application of quasi-Newton methods and analysis of exact penalty functions led to the efficient SQP algorithms by Han and Powell. From this starting point, the next decade saw algorithmic developments by A. Conn, Fletcher, Gill, Murray, Nocedal and many others, which led to advanced features including convergence properties for a variety of merit functions, applications of trust region and line search globalizations for constrained optimization, and efficient factorization and decomposition for large scale problems.

Applications of SQP in process engineering begin in 1980 and include contributions from A. Westerberg and coworkers, R.W.H. Sargent, Biegler, A. Lucia, S. Macchietto, M.A. Stadtherr, W. Morton, B. Kalitventzeff and others. New algorithms have been developed, existing ones have been refined, some good software has been developed, and there has been some computational experience and practical applications. All of these academic efforts have paved the way for effective large scale, on-line optimization strategies, discussed next.

On-line Process Optimization

On-line (or real-time) optimization requires the solution of nonlinear programs that describe the steady state operation of a chemical process. This problem can be solved every few hours and operating conditions (e. g., setpoints to the control system) can be updated in the process to improve operation based on a profit function. Current industrial applications of model-based real time optimization (RTO) address complex chemical plants. T.E. Marlin and A.N. Hrymak [29] list the following features of plant which favor the application of RTO:

1) adjustable optimization variables exist after higher priority safety, quality and production rate objectives have been achieved;
2) profit changes significantly as values of the optimization variables are changed;
3) disturbances occur frequently enough for real-time adjustments to be required;
4) determining the proper values for the optimization variables is too complex to be achieved by selecting from several standard operating procedures.

Systems for on-line optimization have been developed and used since about 1980, but success or failure in industrial applications has largely gone unnoticed until the 1990s. After that, a few publications appeared in the literature (e. g. [3]; [2]). Still, much of the mathematical programming technology has not been documented outside of industrial corporations. Therefore, it is impossible to fully survey and discuss all the industrial applications here. Rather, we highlight a few major areas in which progress is being made, and point out a few references for further detail.

In particular, applications grew in scope and size as computing power to support such activities became available. From 1990 onwards, there has been a significant growth in the application of on-line optimization systems in the process industries. The following applications milestones show the growth of SQP-based applications for real-time optimization [2]:

- 1980s: in house developments at DSM, ICI, Shell ($\geq$ 20000 equations);
- 1986: Shell 'Opera' package ethylene plants;
- 1988: First DMO application: SUNOCO Hydrocracker;
- 1991: Lyondell Integrated refinery;
- 1994: Mobil and Mitsubishi Chemical applications (over 200000 equations);
- 1996: Aspen/DMC/Setpoint mergers.

Note that with the 1996 merger, 80–90% of real-time optimization applications are implementations by Aspen Technology, Inc. A recent and comprehensive review of the issues related to real-time optimization (RTO) may be found in [29]. The components of the current research to RTO of large scale continuous processes based on steady state models are reviewed in [33]. In that article, the issues involved in the design of RTO systems are discussed, particularly with respect to structural decisions, e. g., the choice of measurements to be used to monitor plant performance and update the optimization model, and the level of model complexity to be used in the RTO system. From published studies, summarized in Table 1, a successful RTO application delivers about 3% of the value added by

Successive Quadratic Programming: Applications in the Process Industry, Table 1
Some industrial case studies with SQP optimization

REAL-TIME OPTIMIZATION		
Company	Application	Results
Shell Oil (1986)	Ethylene Plant	$ 4.0M/yr
Wilton (1988)	Power Station	2–6%
Amoco (1990)	Gas Plant	$ 4.0M/yr
British Petroleum	Refinery	$ 2.5M/yr
Chevron USA (1990)	Ethylene Plant	5–10%
Star Enterprise (1990)	Crude Unit	$ 3.0M/yr
Shell Oil	Refinery	9% in gasoline production
Texaco (1990)	Refinery	$ 4.0M/yr
Lyondell (1991)	Ethylene Plant	9 month payout
OMV Deutschland (1991)	Ethylene Plant	1–3%
Conkwright (1994)	Petroleum Crude Distillation	$ 1–2M/yr
ICI	Industrial Steam & power	$ 1.5M/yr
DMC	Ethylene plant	Payback in less than one year
Sunoco (1995)	Hydrocracker	$ 1M/yr
Divekar & Lepore (1991, [17])	Ethylbenzene / styrene	$ 1–2.6M/yr

the plant in economic benefits ([15]; [20,35,43]; [26]). However, it should benoted that published applications cover a small spectrum of the full range of manufacturing plants employed in the process industries, i. e., large scale continuous plants in the petroleum and petrochemical sectors [33].

As seen in Table 1, many successful industrial applications of RTO have been reported. For instance, impressive optimizations were implemented at the SUNOCO Sarnia Canada refinery, which was recognized by a 1995 Computerworld Smithsonian Award for innovative information technology in manufacturing. The following examples with economic benefits suggest the wide range of processes on which RTO has been successful.

Successive Quadratic Programming: Applications in the Process Industry, Table 2
SQP optimization for process design

OFF-LINE PROC OPTIMIZATION		
Company	Application	Remarks
Linde	Optisim	SQP with EO mode
Bayer	VTPLAN	rSQP with EO mode
Bayer	Simulation Manager	rSQP with EO mode
ASPEN	SPEEDUP	rSQP with EO mode
ICI	Flowpack	SQP with Modular mode
BP	Genesys	SQP with Modular mode
Mobil	QUIKBAL	SQP with Modular moed
SimSci	PRO/II	SQP with Modular mode
Hyprotech	HYSYS	SQP with Modular mode
ASPEN	ASPEN Plus	SQP with Modular mode

Off-Line Process Optimization

Optimization for process design is a difficult and complicated task. Here, most discussion of process optimization in the literature focuses on the problem: *given certain operating objectives such as throughput, utilities availabilities, product specifications, what are the best sizes of equipment and operating conditions to minimize an appropriate combination of capital and operating costs?*

To aid in the design task, detailed, comprehensive simulation platforms have been developed (see [6], for a review.) Moreover, over the past two decades there has been an almost complete shift from in-house development and maintenance of simulation packages, e. g., within an operating petrochemical company, to vendor supplied software. Table 2 presents a short summary of current process simulation tools with SQP optimization. All but the last three are in-house packages; the last three entries are vendor software which command most of the usage for design and optimization.

Unlike real-time optimization, these off-line *process simulation programs* are now part of every process engineer's toolkit and have also been widely integrated into the chemical engineering academic curriculum. Moreover, while RTO models remain specialized applications with only a small group of model developers, process simulation tools are available on every engineer's desk, at least in large operating companies, and are used for most day-to-day modeling tasks. These include design of new processes, retrofits of existing ones, debottlenecking the process operations and analysis for operability and control.

While both RTO and off-line process simulation represent steady stateprocess models, off-line models tend to be much more detailed and rigorous.This is due to the fact that these models need to serve much more general applications and also because there are no online data with which to adjustparameters. As a result, these models are much more difficult to solve andarobust modular mode is preferred, particularly if detailed sizing and costing programs are involved.

Broadly speaking, modular process simulation tools can be classified into four levels:

1) At the lowest level, basic physical properties and thermodynamic relationships (e. g., phase equilibrium, energy balance terms, transport relationships) have been incorporated. These contain the vast majority of process equations and these are solved with specialized solution algorithms.
2) At the next level, are the basic *building blocks for the process units*, including distillation, heat exchange, reaction and material transfer. These blocks consist of mass and energy balances as well as constitutive equations, solved with specialized procedures; they also rely heavily on underlying physical property equations.
3) This level deals with the *convergence of the overall flowsheet*. Here process units are sequenced, tear streams are chosen, their values are updated and recycle loops are converged. It is at this level that flow-

sheet optimization is introduced since the SQP algorithm extends the overall convergence function by incorporating tear equations as equality constraints in the optimization problem. Often this problem is fairly small (fewer than 100 variables and constraints) and a dense SQP algorithm without decomposition is usually satisfactory. Here the dominant computational cost for the optimization lies in the evaluation of the objective and constraint functions (and gradients) from the process model.

4) The process simulator is capped with a *graphical user interface* that communicates with the process engineer in setting up and solving the simulation and optimization problem.

For off-line process optimization, the list of successful applications is too numerous to mention, as it is currently a routine task, distributed across all sectors of the chemical industry. A number of case studies for flowsheet optimization have been summarized in [6]. Moreover, the user guides for ASPEN+, PRO/II and HYSYSprovide ample documentation and examples on the use of their SQP-based optimization tools.

Conclusions And Future Work

This article provides a brief review of nonlinear programming strategies and applications in chemical process optimization. In many industrial applications, the NLP algorithm of choice is successive quadratic programming (SQP) and a description of the algorithm, and its variations, is provided. In particular, we develop the basic SQP algorithm and then concentrate on large scale extensions. For process optimization, we take advantage of two model characteristics: these problems have few decision variables (≤ 100) despite their large model size and, despite advances in software development, second derivatives are often hard to evaluate. As a result, reduced space decompositions for SQP (rSQP) have been developed for a number of industrial applications.

Process models that are formulated for optimization can be classified as modular and equation oriented modes. In the first mode, function values are expensive as most of the process equations are solved internally with specialized solution procedures. As a result, the optimization problem seen by SQP is relatively small and can be solved without decomposition. In the equation oriented mode, the process equations are integrated into the optimization problem and the burden of the solution is passed on to the NLP solver. Here, decomposition strategies such as rSQP are essential for efficient process optimization.

Finally, we classify process applications as off-line, devoted to design and analysis studies and on-line, devoted to monitoring and optimization of an operating process in real time. Currently, off-line optimization tasks are often performed with modular simulation tools that incorporate SQP strategies without decomposition. In contrast, on-line process optimization is performed almost entirely with equation oriented models and require the implementation of decomposition strategies like rSQP. A number of applications in both categories are cited in this article.

Future work related to NLP applications in process optimization deals with further development of the SQP algorithm, extension of large scale decomposition strategies and larger, more sophisticated problem formulations for process application.

Fundamental development of SQP algorithms deals with improving the local and global convergence properties of the algorithm. These properties have been strengthened through the analysis of trust region strategies as well as additional safeguards in dealing with rank deficiency and inequality constraints. Related to this are the application of interior point (IP) strategies that improve the efficiency and reliability of large scale, highly constrained NLPs. These IP (or barrier) methods can be applied at the level of the QP subproblem (see [47]; [44]) or the barrier terms can be applied directly to the NLP problem [10]. Since the computational effort of barrier methods (either at the QP or the NLP level) does not increase greatly with an increase in the number of inequality constraints, this approach seems to be essential to deal with ever increasing problem sizes.

Moreover, decomposition strategies for large scale NLP can be considered in two categories. For full space SQP, decomposition occurs at the QP and the linear algebra level, and effective strategies have been developed to factorize indefinite, sparse systems. These also require first and second derivatives from the process model. Future developments in process modeling systems need to provide these capabilities. Also, further conceptual development is needed to deal with large, full space QPs with indefinite Hessian matrices.

For reduced space methods, the multiplier free approach can be applied to a tailoring of process models, where existing modular models (if solved with Newton-based procedures) can be solved simultaneously with the NLP, using rSQP. With this approach, the best of the modular and equation oriented modes can be achieved; reliable, detailed models with specialized initializations and solvers can be optimized quickly and simultaneously. This approach has been demonstrated in a number of process applications in [12,39,41] and [1].

Finally, with the development of improved NLP solvers and decomposition strategies there are a number of process applications that extend beyond process optimization, both for on-line and off-line optimization. For on-line optimization, the current challenges lies in combining the control and RTO layers in a chemical process. The resulting formulation is a differential-algebraic optimization problem, which can be posed as a large scale NLP with many decision variables. These problems require novel decomposition approaches that are beyond the scope of this article (see e. g., [5]). Related to control and optimization are the problems of state estimation and parameter estimation. These tasks are essential to identify the optimization model and have the same structure as the differential-algebraic optimization problem.

For off-line optimization, a number of capabilities are required that extend beyond process optimization. Once an optimal flowsheet has been found, a number of questions still need to be answered, before the solution can be implemented. These issues can be summarized by the following items:

- *Sensitivity of optimal flowsheets*: How does the optimum flowsheet change with changes in input conditions and model uncertainty? ([46])
- *Design under uncertainty*: What is the optimal process that can accommodate a range of uncertainties? ([23])
- Operability and flexibility analysis of flowsheets Operability and flexibility analysis of flowsheets: Over what range of uncertainty does an existing process function? ([19])
- *Integration of dynamic considerations and controllability*: How well can the designed or existing process reject disturbances and move from one desired setpoint to another? ([33])

As a result of these open questions, process optimization still appears to be an active and fertile area for future research.

See also

- ▶ Feasible Sequential Quadratic Programming
- ▶ Optimization with Equilibrium Constraints: A Piecewise SQP Approach
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Successive Quadratic Programming
- ▶ Successive Quadratic Programming: Applications in Distillation Systems
- ▶ Successive Quadratic Programming: Decomposition Methods
- ▶ Successive Quadratic Programming: Full Space Methods
- ▶ Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Alkaya D, Vasantharajan S, Biegler LT (2000) Generalization and application of tailored approach for flowsheet optimization. I EC Res 39(6):1731–1742
2. Ayala J (1997) Realtime optimization. Aspenworld
3. Bailey JK, Hrymak AN, Treiber SS, Hawkins RB (1993) Nonlinear optimization of a hydrocracker fractionation plant. Comput Chem Eng 17:123
4. Beale EML (1967) Numerical methods. In: Abadie J (ed) Nonlinear Programming. North-Holland, Amsterdam, p 189
5. Biegler LT (1998) Advances in nonlinear programming concepts for process control. J Process Control 8(5-6):301 See also: Proceedings ADCHEM 1997
6. Biegler LT, Grossmann IE, Westerberg AW (1997) Systematic methods of chemical process design. Prentice-Hall, Englewood Cliffs, NJ
7. Biegler LT, Nocedal J, Schmid C (1995) A reduced Hessian method for large scale constrained optimization. SIAM J Optim 5(2):314
8. Biegler LT, Schmid C, Ternet D (1997) A multiplier-free, reduced Hessian method for process optimization. In: Biegler LT, Coleman TF, Conn AR, Santosa FN (eds) Large-Scale Optimization with Applications, Part II: Optimal Design and Control. Springer, Berlin, p 101
9. Boggs PT, Tolle JW (1996) Sequentialquadratic programming. Acta Numer:1–100
10. Byrd RH, Gilbert JC, Nocedal J (1998) A trust region method based on interior point techniques for nonlinear programming. Techn Report Northwestern Univ

11. Byrd RH, Nocedal J (1991) An analysis of reduced Hessian methods for constrained optimization. Math Program 49:285–323
12. Cervantes A, Biegler LT (1998) Large scale DAE optimization using simultaneous nonlinear programming formulations. AIChE J 44(5):1038
13. Chamberlain RM, Lemarechal C, Pedersen HC, Powell MJD (1982) The watchdog technique for forcing convergence in algorithms for constrained optimization. Math Program Stud 16:1–17
14. Cronkwright M (1994) Experience with online optimization. Proceed. Canad. Chem. Confer. Annual Meet., Calgary, pp 477–478
15. Cutler CR, Perry RT (1983) Real time optimization with multivariable control is required to maximize profits. Comput Chem Eng 7(5):663–667
16. Dennis JE, More JJ (1977) Quasi-Newton methods, motivation and theory. SIAM Rev 19(1):46–89
17. Divekar S, Lepore J (1991) Operational improvements in Ethylbenze/Styrene plant due to advanced process control and on-line optimization. EB/SM Tecnology Conference
18. Fletcher R (1987) Practical methods of optimization
19. Floudas CA, Grossmann IE (1987) Active set strategy for flexibility analysis in chemical processes. Comput Chem Eng 11:675
20. Foster D (1987) Economic performance optimization of a combined heat and power station. Proc ImechE 201(A3):201–206
21. Gill PE, Murray W, Wright MH (1981) Practical optimization. Acad. Press, New York
22. Goldfarb D, Idnani A (1983) A numerically stable dual method for solving strictly convex quadratic problems. Math Program 27:1–33
23. Grossmann IE, Straub D (1991) Recent developments in the evaluation and optimization of flexible chemical processes. In: Puigjaner L, Espuna A (eds) Proc. COPE-91, Barcelona, Spain
24. Han SP (1977) A globally convergent method for nonlinear programming. J Optim Th Appl 22/23:297–309
25. (1995) Harwell Subroutine Library. Report Atomic Energy Res Establishment
26. Lauks VE, Vasbinder RJ, Vallenburg PJ, van Leeuwen C (1992) On line optimization of an ethylene plant. Comput Chem Eng 16S:S213–220
27. Locke MH, Edahl R, Westerberg AW (1983) An improved successive quadratic programming optimization algorithm for engineering design problems. AIChE J 29(5)
28. Lucia A, Xu J, D'Couto GC (1990) Sparse quadratic programming in chemical process optimization. Ann Oper Res 42:55
29. Marlin TE, Hrymak AN (1997) Real-time operations optimization of continuous processes. In: Kantor JC, Garcia CE, Carnahan B (eds) AIChE Symp. 316: Chemical Process Control, vol 93, pp 156–164
30. Murray W, Wright M (1978) Projected Lagrangian methods based on trajectories of barrier and penalty methods. SOL Techn Report Stanford Univ no. 78-23
31. Murtagh BA (1982) On the simultaneous solution and optimization of large scale engineering systems. Comput Chem Eng 6(1):1–5
32. Murtagh BA, Saunders MA (1982) A projected lagrangian algorithm and its implementation for sparse nonlinear constraints. Math Program Stud 16:84–117
33. Perkins JD (1998) Plant wide optimization-opportunities and challenges. Proc. AIChE Symp., vol 32c, p 15
34. Powell MJD (1978) A fast algorithm for nonlinearly constrained optimization calculations. Dundee Conference on Numerical Analysis, Dundee, Scotland, 1977 In: Lecture Notes Math, vol 630. Springer, Berlin p 144
35. Saha LE, Chonots AJ, Hatch DR (May 1990) Optimization at Wyoming gas plant improves quality. Oil and Gas J
36. Sargent R (1997) The development of sequential quadratic programming. In: Biegler LT, Coleman T, Conn AR, Santosa F (eds) Large Scale Optimization with Applications Part II: Optimal Design and Control. IMA Vol Math Appl, vol 93. Springer, Berlin
37. Schittkowski K (1981) The nonlinear programming algorithm of Wilson, Han and Powell with an augmented Lagrangian type line search function, Part 1: Convergence analysis. Numer Math 38:83
38. Schittkowski K (1981) The nonlinear programming algorithm of Wilson, Han and Powell with an augmented Lagrangian type line search function, Part 2: An efficient implementation with linear least squares subproblems. Numer Math 38:115
39. Schmid C, Biegler LT (1993) Quadratic programming methods for tailored reduced Hessian SQP. Comput Chem Eng 18(9):817–832
40. Simon JD, Azma HM (1983) Exxon experience with large-scale linear and nonlinear programming applications. Comput Chem Eng 7(5):605–614
41. Tanartkit P, Biegler LT (1996) Reformulating ill-conditioned differential-algebraic equation optimization problems. Industr Eng Chem Res 35:1853–1865
42. Ternet D, Biegler LT (1998) Recent improvements to a multiplier-free reduced Hessian successive quadratic programming algorithm. Comput Chem Eng 22(7-8):963
43. van Wijk RA, Pope MR (1992) Advanced process control and on-line optimization in Shell refineries. Comput Chem Eng 16S:S69–80
44. Vassiliadis V, Brooks SA (1998) Application of the modified barrier method to large-scale quadratic programming. Comput Chem Eng 22(9):1197
45. Wilson RB (1963) A simplicial algorithm for concave programming. PhD Thesis, Harvard University
46. Wolbert D, Joulia X, Koehret B, Biegler LT (1994) Flowsheet optimization and optimal sensitivity analysis using exact derivatives. Comput Chem Eng 18(11-12):1083

47. Wright SJ (1997) Primal-dual interior point methods. SIAM, Philadelphia
48. Yamashita H (1982) A globally convergent constrained quasi-Newton method with an augmented Lagrangian type penalty function. Math Program 23:75

Successive Quadratic Programming: Decomposition Methods

ANGELO LUCIA
Department Chemical Engineering,
University Rhode Island, Kingston, USA

MSC2000: 90C30, 90C20

Article Outline

Keywords
Nonlinear Programming
Decomposition of the Quadratic Program
Quadratic Programming Kuhn–Tucker Conditions for Decomposition Methods
Choice of Basis
Decoupling the Stationary Conditions
Methods for Approximating the Projected Lagrangian Hessian Matrix
Factoring the Jacobian and Projected Lagrangian Hessian Matrices
An SQP Algorithm for Decomposition Methods
Some Comments on Numerical Performance of Decomposition Methods
See also
References

Keywords

Successive quadratic programming; Lagrangian function; Kuhn–Tucker conditions; QR factorization; Range space; Null space; Decomposition; Reduced quadratic programming subproblem; Projected Hessian matrix of a Lagrangian function

Decomposition methods of *successive quadratic programming* (SQP) are methods that reduce the size of the recursive quadratic programming (QP) subproblems by using the equality constraints to 'eliminate' variables. Decomposition methods are particularly useful when the number of degrees of freedom, $n - m_{eq}$, is small, where n is the number of unknown variables and m_{eq} is the number of equality constraints, because this results in small and tractible quadratic programming subproblems in which there is little need to be concerned with sparsity, fill-in and other issues in large scale quadratic and successive quadratic programming. Thus quasi-Newton updates like the modified Broyden–Fletcher–Goldfarb–Shanno (BFGS) formula can be used to maintain hereditary positive definite approximations to the projection of the Hessian matrix of the Lagrangian function on the linearized constraint surface (tangent plane) and the solutions to the recursive quadratic programs are unique. However, when decomposition methods are used, it is often necessary to use quasi-Newton approximations to the Lagrangian Hessian matrix and thus the asymptotic rate of convergence is at best two-step Q-superlinear as opposed to quadratic if analytical or finite difference second derivatives are used. Moreover, some curvature (or second derivative) information is ultimately lost as the linearized constraint surface orientation changes because second derivative information is only being gathered or approximated on the tangent subspace, while it is neglected in directions orthogonal to the tangent subspace. Additional techniques for recovering this 'lost' curvature information and for preserving sparsity in the Jacobian matrix of the constraints have also been proposed.

All decomposition methods are based on a choice of basis for the vector space defined by the x variables in the optimization problem and are best suited for nonlinear programming problems with equality constraints and simple bounds on variables. More general nonlinear inequalities are usually handled by converting these inequalities to equalities using slack variables. Some early decomposition methods [1,6] in engineering used canonical bases while more recent range and null space decomposition (RND) methods [7], choose basis vectors that align with the range and null space of the Jacobian matrix of the constraints. Range and null space decomposition methods were introduced by W. Murray and M.H. Wright in the late 1970s (see [4]). The null space, Z, of the Jacobian matrix, J_E, is a vector space that satisfies the condition $J_E w_k = 0$ for any nonzero vector $w_k = \sum \alpha_j z_j$, where $\{z_j\}$ are the basis vectors for the null space and where the dimension of the null space is equal to the number of distinct vectors z_j.

The null space of the Jacobian matrix of the constraints is the tangent subspace of the constraints. The range space, Y, of the Jacobian matrix, on the other hand, is the space orthogonal to Z such that the direct sum $Y + Z = R^n$, where n denotes the number of x variables. This is where the name range and null space decomposition (RND) comes from and this particular choice of basis also leads to some simplified algebra. RND methods are the decomposition methods currently in use.

The issues that are central to decomposition methods of successive quadratic programming include:

- the choice of basis for the linearized constraint surface;
- decoupling, simplifying assumptions and other related concerns;
- methods for approximating the projection of the Lagrangian Hessian matrix;
- the methods used in factoring the Jacobian and Lagrangian Hessian matrices.

Nonlinear Programming

Successive quadratic programming methods address the problem of finding a local solution to the following nonlinear programming (NLP) problem:

$$\begin{cases} \min & f(x) \\ \text{s.t.} & c(x) < 0, \end{cases}$$

where x is a vector of length n which represents an estimate of the local minimum, $f(x)$ is a twice continuously differentiable objective function and $c(x)$, the vector of equality and/or inequality constraints, is also twice continuously differentiable and nonlinear. The associated *Lagrangian function* for this nonlinear programming problem is

$$L(x) = f(x) + \lambda^\top c(x) + \mu^\top c(x),$$

where λ and μ are vectors of Lagrange and Kuhn–Tucker multipliers associated with the equality and inequality constraints, respectively. The corresponding gradient of the Lagrangian function, $g_L(x)$, is defined by

$$g_L(x) = g(x) + \lambda^\top g_c(x) + \mu^\top g_c(x),$$

where $g(x)$ is the gradient of the objective function, $g_c(x)$ is the gradient (or vector of first partial derivatives) of the constraint functions.

Decomposition of the Quadratic Program

All successive quadratic programming methods solve nonlinear programs by recursively solving quadratic programming subproblems based on a quadratic approximation of the Lagrangian function and decomposition methods are no different. Consider then the recursive quadratic program on the kth SQP iteration given by

$$\begin{cases} \min & g(x_k)^\top \Delta x_k + \frac{1}{2}\Delta x_k^\top B_k \Delta x_k \\ \text{s.t.} & c(x_k) + J_E \Delta x_k = 0, \\ & \text{and} \quad x_L \le \Delta x_k \le x_U, \end{cases}$$

where $g(x_k)$ is the gradient of the nonlinear programming objective function, $f(x)$, evaluated at x_k, $c(x_k)$ is the set of equality constraints, J_E is the $(m_{\text{eq}} \times n)$ Jacobian (or first partial derivative) matrix for the equality constraints, B_k is an $(n \times n)$ approximation to the Hessian (or second partial derivative) matrix of the Lagrangian function, x_L and x_U are the lower and upper bounds on the variables respectively and Δx_k, λ_k, and μ_k represent the desired solution or change in the unknown variables, Lagrange and Kuhn–Tucker multipliers respectively. Remember, general inequalities are converted into equalities using slack variables so they are present as part of the set of equalities in this formulation. Only the bounds on variables remain as inequalities.

Decomposition methods are based on the idea of splitting (or decomposing) the unknown variables, x, into two groups, m_{eq} dependent variables, say y, and $n - m_{\text{eq}}$ independent variables, z. Once this framework is established the change in the unknown variables, Δx_k, can be represented by the matrix equation

$$\Delta x_k = Y_k \Delta y_k + Z_k \Delta z_k,$$

where the matrices Y_k and Z_k are $n \times m_{\text{eq}}$ and $n \times (n - m_{\text{eq}})$, respectively. Substitution of this expression for Δx_k into the quadratic program gives

$$\begin{cases} \min & g(x_k)^\top [Y_k \Delta y_k + Z_k \Delta z_k] \\ & \quad + \frac{1}{2}[Y_k \Delta y_k + Z_k \Delta z_k]^\top B_k [Y_k \Delta y_k + Z_k \Delta z_k] \\ \text{s.t.} & c(x_k) + J_E [Y_k \Delta y_k + Z_k \Delta z_k] = 0 \\ & x_L \le Y_k \Delta y_k + Z_k \Delta z_k \le x_U. \end{cases}$$

The reformulated equality constraints can be rearranged to give $\Delta y_k = -[J_E Y_k]^{-1}[c(x_k) + J_E Z_k \Delta z_k]$,

which in turn can be substituted into the quadratic approximation of the Lagrangian function to give a 'reduced' objective function expression and a *'reduced' quadratic program* in the $n - m_{\text{eq}}$ variables Δz_k only. That is, substitution of this last expression for Δy_k gives the 'reduced' quadratic program defined by

$$\begin{cases} \min & g(x_k)^\top [Y_k(-[J_E Y_k]^{-1}[c(x_k) + J_E Z_k \Delta z_k]) \\ & \quad + Z_k \Delta z_k] \\ & + \frac{1}{2}[Y_k(-[J_E Y_k]^{-1}[c(x_k) + J_E Z_k \Delta z_k]) \\ & \quad + Z_k \Delta z_k]^\top \\ & \times B_k[Y_k(-[J_E Y_k]^{-1}[c(x_k) + J_E Z_k \Delta z_k]) \\ & \quad + Z_k \Delta z_k] \\ \text{s.t.} & x_L + Y_k([J_E Y_k]^{-1}[c(x_k) + J_E Z_k \Delta z_k]) \\ & \leq Z_k \Delta z_k \\ & \leq x_U + Y_k([J_E Y_k]^{-1}[c(x_k) + J_E Z_k \Delta z_k]). \end{cases}$$

Note that the bounds on Δy_k effect the bounds on Δz_k through the $Y_k[J_E Y_k]^{-1}[c(x_k)]$ term.

Quadratic Programming Kuhn–Tucker Conditions for Decomposition Methods

For decomposition methods, the *Kuhn–Tucker conditions* for the original quadratic program are given by the generalized stationary conditions for the 'reduced' quadratic program

$$\begin{aligned} & [C_k + Z_k]^\top g(x_k) + (C_k + Z_k)^\top B_k a_k \\ & + [C_k^\top B_k C_k + 2C_k^\top B_k Z_k + Z_k^\top B_k Z_k]\Delta z_k \\ & + J_I^\top \mu_k = 0, \\ & J_I \Delta z_k = 0, \\ & \mu_k \geq 0, \end{aligned}$$

where the vector $a_k = -Y_k[J_E Y_k]^{-1} c(x_k)$ and the $(n \times (n - m_{\text{eq}}))$-matrix $C_k = -Y_k[J_E Y_k]^{-1} J_E Z_k$, J_I is the Jacobian matrix for the active bounds on Δz_k and μ_k are the Kuhn–Tucker multipliers associated with those active bounds.

There are also the equations defining the change in the dependent variables

$$\Delta y_k = -[J_E Y_k]^{-1}[c(x_k) + J_E Z_k \Delta z_k]$$

and the conditions defining the Lagrange multipliers for the equality constraints

$$Y_k^\top B_k[Y_k \Delta y_k + Z_k \Delta z_k] + Y_k^\top J_E^\top \lambda_k = -Y_k^\top g(x_k)\,.$$

This last condition comes from the Kuhn–Tucker conditions for the quadratic program formulated in terms of both Δy_k and Δz_k.

Choice of Basis

Choices for the matrices Y_k and Z_k give different decomposition methods, some of which are more convenient algebraically than others. For example, the choice of basis in the decomposition method in [1] corresponds to $Y_k = (I - J_y^{-1} J_z)$ and $Z_k = (0\ I)$, where the Jacobian matrix of the equality constraints, J_E, is partitioned into $(J_y\ J_z)$ such that the $(m_{\text{eq}} \times m_{\text{eq}})$-submatrix J_y is invertible or nonsingular and the order of the matrix partitions are such that they are consistent with matrix-vector and matrix-matrix multiplication. For *range and null space decomposition* (RND) methods [7], the choice of basis is given by $Y_k = (I - J_y^{-1} J_x)$ and $Z_k = (J_x J_y^{-1})$ such that $J_E Z_k = [0]$. The condition $J_E Z_k = [0]$ means that the matrix product $J_E Z_k$ gives the zero matrix and therefore each column of the matrix Z_k is carried to the zero vector by the Jacobian matrix.

Use of the *null space* condition $J_E Z_k = [0]$ simplifies the reduced quadratic program Kuhn–Tucker conditions to

$$\begin{aligned} & Z_k^\top g(x_k) - Z_k^\top B_k[Y_k[J_E Y_k]^{-1} c(x_k)] \\ & + Z_k^\top B_k Z_k \Delta z_k + [Z_k^\top, -Z_k^\top]\mu_k = 0, \\ & Z_k \Delta z_k - x_U - Y_k[J_E Y_k]^{-1}[c(x_k)] \leq 0, \\ & -Z_k \Delta z_k + x_L + Y_k[J_E Y_k]^{-1}[c(x_k)] \leq 0, \end{aligned}$$

since the $(n \times (n - m_{\text{eq}}))$-matrix $C_k = -Y_k[J_E Y_k]^{-1} J_E Z_k = [0]$,

$$\Delta y_k + [J_E Y_k]^{-1}[c(x_k)] = 0$$

and

$$Y_k^\top B_k[Y_k \Delta y_k + Z_k \Delta z_k] + Y_k^\top J_E^\top \lambda_k = -Y_k^\top g(x_k)\,.$$

Decoupling the Stationary Conditions

The last four equations defining Δz_k, Δy_k, λ_k and μ_k are coupled because of the last condition and require projected second derivative information from both the *range space* ($Y_k^\top B_k Y_k$) and the cross product of the range and null spaces ($Y_k^\top B_k Z_k$). To remove the need for this information and thereby decouple the stationary conditions, simplifying assumptions are necessary. In particular, it is assumed that because Δz_k and Δy_k are zero at any constrained local minimum, $\Delta y_k = 0$ and $\Delta z_k = 0$ can be used at all iterations, which reduces the last equation to

$$Y_k^\top J_E T \lambda_k = -Y_k^\top g(x_k).$$

This decouples the variables z_k and μ_k from Δy_k and λ_k. However, it should be pointed out that this simplification results in a loss of curvature information in the range space of the Jacobian matrix and slightly slower asymptotic rates of convergence. That is, second derivatives information associated with $Y_k^\top B_k Y_k$ and $Y_k^\top B_k Z_k$, which is not constant for general nonlinear constraints, is lost and the resulting rate of convergence is two-step Q-superlinear.

There are also other concerns related to the bounds on the change in the dependent variables, Δy_k, that require attention. The relationship $\Delta y_k = -[J_E Y_k]^{-1}[c(x_k)]$ that defines the change in the dependent variables as a function of the change in the independent variables also directly effects the bounds on the independent variables in the reduced quadratic program, which are given by

$$\begin{aligned} x_L + Y_k[J_E Y_k]^{-1}[c(x_k)] &\le Z_k \Delta z_k \\ &\le x_U + Y_k[J_E Y_k]^{-1}[c(x_k)]\,. \end{aligned}$$

Some care must be exercised to avoid conflicting bounds and infeasible reduced quadratic programs.

Methods for Approximating the Projected Lagrangian Hessian Matrix

With the above simplifications, the only second derivative information required in the Kuhn–Tucker conditions for the reduced quadratic program is the $((n - m_{\text{eq}}) \times (n - m_{\text{eq}}))$-matrix $Z_k^\top B_k Z_k$, which is the *projection* of the full Lagrangian Hessian matrix onto the tangent subspace defined by the linearized constraints. Moreover, because the matrix $Z_k^\top B_k Z_k$ must be positive definite at any constrained local minimum and can be much smaller in dimension than B_k in many applications, $Z_k^\top B_k Z_k$ is often approximated using the BFGS formula or some other suitable quasi-Newton update that preserves hereditary positive definiteness [7]. The primary advantage in doing this is that the change in the independent variables and associated Kuhn–Tucker multipliers for any bounds can be determined by solving a smaller, convex quadratic program. This results in computational savings as well guaranteeing an iteratively unique Δz_k. However, remember the trade-off for this is loss of curvature in the range space and two-step Q-superlinear convergence, which is a bit slower than Q-superlinear or quadratic convergence.

Using analytical or finite difference second derivatives in full space methods is straightforward; using analytical or finite difference second derivatives in decomposition methods is not. With quasi-Newton approximations, the projection $Z_k^\top B_k Z_k$ can be easily formed and stored as a small dense $((n - m_{\text{eq}}) \times (n - m_{\text{eq}}))$-matrix. Moreover, there is no need to explicitly calculate or store B_k. On the other hand, to use analytical or finite difference second derivatives in decomposition methods the matrix triple product $Z_k^\top B_k Z_k$ must be explicitly formed and therefore the entire Hessian matrix of the Lagrangian function, B_k, must be evaluated and stored. Clearly this is counter to the overall purpose of decomposition.

Factoring the Jacobian and Projected Lagrangian Hessian Matrices

The iterative computation of the range and null space of the Jacobian matrix is normally accomplished by QR factorization. That is, the Jacobian matrix of the constraints is factored iteratively according to the rule

$$\begin{aligned} J_E(x_k) &= Q(x_k) \begin{pmatrix} R^\top(x_k) & 0 \end{pmatrix}^\top \\ &= \begin{pmatrix} Y_k & Z_k \end{pmatrix} \begin{pmatrix} R_k^\top & 0 \end{pmatrix}^\top, \end{aligned}$$

where $Q(x_k)$, which is the product of Householder transformation matrices, is an orthonormal matrix and partitioned into Y_k and Z_k and where $R(x_k)$ is an upper triangular $(m_{\text{eq}} \times m_{\text{eq}})$-matrix. In general, the matrices $Q(x_k)$ and $R^\top(x_k)$ will be dense matrices and therefore not well suited for large problems in which $J_E(x_k)$ is sparse. Other factorizations such as LQ factorization

can be used and it is also possible to update the sparse factors of the Jacobian matrix to reduce storage requirements [5].

When $Z_k^\top B_k Z_k$ is approximated using the BFGS or some equivalent hereditary positive definite quasi-Newton update, it can be factored reliably and efficiently using $LDL^\top$ factorization, where L and D are lower triangular and diagonal factors, respectively. In fact, techniques exist for updating the quasi-Newton lower and diagonal factors directly to avoid the expense of factorization altogether [5].

An SQP Algorithm for Decomposition Methods

A generic successive quadratic programming algorithm using decomposition methods is shown below:

1) Initialize x, λ, μ and $(Z_k^\top B_k Z_k)$; define a convergence tolerance, $\epsilon > 0$, and set $k = 0$.
2) Evaluate $f(x_k)$, $g(x_k)$, $c(x_k)$ and $J_E(x_k)$.
3) If $\| (g_L(x_k)\ c(x_k))^\top \|_2 < \epsilon$, $c(x_k) = 0$ and $\mu_k \geq 0$, then stop. Otherwise, go to step 4.
4) Factor $J_E(x_k)$ by QR (or some other equivalent) factorization so that $J_E(x_k) = (Y_k\ Z_k)\ (R_k^\top\ 0)^\top$.
5) Define the dependent variables, y_k, and independent variables, z_k.
6) Determine the change in the dependent variables from $\Delta y_k + [J_E Y_k]^{-1}[c(x_k)] = 0$ and set $y_{k+1} = y_k + \Delta y_k$.
7) Solve the equation $Y_k^\top J_E^\top \lambda_k = -\ Y_k^\top g(x_k)$ for the Lagrange multipliers λ_k.
8) Construct the reduced quadratic program

$$\begin{cases} \min & (Z_k^\top g(x_k) \\ & -Z_k^\top B_k [Y_k [J_E Y_k]^{-1} c(x_k)])^\top \Delta z_k \\ & +\frac{1}{2} \Delta z_k^\top (Z_k^\top B_k Z_k) \Delta z_k \\ \text{s.t.} & x_L + Y_k [J_E Y_k]^{-1} [c(x_k)] \\ & \leq Z_k \Delta z_k \\ & \leq x_U + Y_k [J_E Y_k]^{-1} [c(x_k)] \end{cases}$$

and solve it for Δz_k and μ_k. Set $z_{k+1} = z_k + \Delta z_k$.
9) Determine $x_{k+1} = (y_{k+1}, z_{k+1})$ from by either line searching, trust regions or some other means.
10) Calculate a new approximation to the projected Hessian matrix, $Z_{k+1}^\top B_{k+1} Z_{k+1}$, using the BFGS formula or some equivalent hereditary positive definite quasi-Newton update.
11) Set $k = k + 1$ and go to step 2.

Some Comments on Numerical Performance of Decomposition Methods

Decomposition methods have been applied to a variety of small and large scale nonlinearly constrained optimization problems in both the mathematical sciences [7] and engineering [1,2,3]. Most agree that decomposition methods are best suited for applications in which the number of degrees of freedom or number of independent variables (i. e., $n - m_{\text{eq}}$) is small compared to the total number of variables, a situation that occurs in many practical applications. Good numerical results have been reported for mathematical benchmark problem [7] and small and large chemical process engineering problems [2]. In particular, J. Nocedal and M.L. Overton [7] report that RND methods compare favorably with the full space SQP method of M.J.D. Powell [8] on a set of small mathematical benchmark problems involving up to eight variables and four equality constraints. L.T. Biegler [2] and H.S. Chen and M.A. Stadtherr [3] show that decomposition methods work well on a variety of chemical process problems including multicomponent, multistage distillation optimization problems involving up to 1000 variables.

See also

- ▶ Decomposition Principle of Linear Programming
- ▶ Feasible Sequential Quadratic Programming
- ▶ Generalized Benders Decomposition
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Logic-based Methods
- ▶ Optimization with Equilibrium Constraints: A Piecewise SQP Approach
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Simplicial Decomposition
- ▶ Simplicial Decomposition Algorithms
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Successive Quadratic Programming
- ▶ Successive Quadratic Programming: Applications in Distillation Systems
- ▶ Successive Quadratic Programming: Applications in the Process Industry
- ▶ Successive Quadratic Programming: Full Space Methods

► Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Berna TJ, Locke MH, Westerberg AW (1980) A new approach to optimization of chemical processes. AIChE J 26:37–43
2. Biegler LT (1992) Tailoring optimization algorithms to process applications. Comput Chem Eng 16S:81–95
3. Chen HS, Stadtherr MA (1985) A simultaneous modular approach to process flowsheeting and optimization. AIChE J 31:1868–1881
4. Gill PE, Murray W, Wright MH (1981) Practical optimization. Acad. Press, New York
5. Goldfarb D (1975) Matrix factorizations in optimization of nonlinear functions subject to linear constraints. Math Program 10:1–31
6. Locke MH, Edahl R, Westerberg AW (1983) An improved successive quadratic programming optimization algorithm for engineering design problems. AIChE J 29:871–874
7. Nocedal J, Overton ML (1985) Projected Hessian updating algorithms for nonlinearly constrained optimization. SIAM J Numer Anal 22:821–850
8. Powell MJD (1978) A fast algorithm for nonlinearly constrained optimization calculations. In: Watson G (ed) Numerical analysis, Dundee 1977, Lecture Notes Math. Springer, Berlin, pp 144–157

Successive Quadratic Programming: Full Space Methods

ANGELO LUCIA
Department Chemical Engineering,
University Rhode Island, Kingston, USA

MSC2000: 90C30, 90C25

Article Outline

Keywords
Nonlinearly Constrained Optimization
Kuhn–Tucker Conditions
for the Quadratic Program
Estimating the Hessian Matrix
of the Lagrangian Function
Convexity
Nonconvexity
Solving the Kuhn–Tucker Conditions
for Quadratic Programming Subproblems
Active Set Methods
Interior Point Methods
Multiple QP Kuhn–Tucker Points
and Related Issues
Loss of Descent in the Nonlinear Program
Initializing the Unknown Variables
Related Numerical Studies
See also
References

Keywords

Full space successive quadratic programming; Full space of x variables; Sparsity; Matrix of second partial derivatives; Jacobian matrix; Convex; Nonconvex; Indefinite quadratic programs; Multiple Kuhn–Tucker points; Loss of descent; Active set methods; Interior point methods

Full space SQP methods are *successive quadratic programming* methods that build approximations to the *Hessian matrix* of the *Lagrangian function* and solve the resulting quadratic programming subproblems in the full space of the unknown variables. The original SQP methods of R.B. Wilson [25], S.P. Han [12] and M.J.D. Powell [21] are formulated in terms of all of the x variables and are therefore full space methods. Usually, however, 'full space methods' refers to those SQP methods that operate in the full space of the x variables when the number of variables, say n, is large and, as a result, are simultaneously concerned with the sparsity (the relative number of zero and nonzero elements) of the matrix of second partial derivatives of the Lagrangian function and the Jacobian matrix of the constraint functions, as well as techniques for factoring, updating and solving the Kuhn–Tucker conditions for the recursive quadratic programs and other related issues. A *sparse matrix* is one in which the number of nonzero elements is a small fraction of the total, and performing arithmetic operations with only these nonzero elements reduces the overall computational workload. Clearly, the SQP methods of Han and Powell were not intended for problems in which the number of x variables is large. Full space SQP methods can be further categorized as 'convex' or 'nonconvex' and this characterization has bearing on the techniques used to estimate the Hessian matrix of the Lagrangian function, its result-

ing curvature, the nature of the recursive quadratic programs (i. e., whether they are positive definite or indefinite), and the methods needed to factor and solve the Kuhn–Tucker conditions for the quadratic programming subproblems. In particular, nonconvexity (or indefiniteness) requires special factorization techniques and more complex active set methods, can give rise to indefinite quadratic programs and multiple Kuhn–Tucker points (or solutions) to the recursive quadratic programming problems, and can result in a loss of descent in the parent nonlinear programming problem causing *line searching* and other difficulties. Virtually all of these difficulties disappear when convexity can be guaranteed.

Thus the issues that are important when the number of x variables is large are:

- estimating the Hessian matrix of the Lagrangian function,
- various aspects of solving the Kuhn–Tucker conditions for the quadratic program,
- solution (Kuhn–Tucker point) multiplicity in the quadratic program,
- loss of descent in the parent nonlinear program, and
- initializing the unknown variables.

Nonlinearly Constrained Optimization

The general *nonlinear programming* (NLP) problem is given by

$$\begin{cases} \min & f(x) \\ \text{s.t.} & c(x) \le 0, \end{cases}$$

where x is a vector of length n which represents an estimate of the local minimum, $f(x)$ is a twice continuously differentiable objective function and $c(x)$, the vector of equality and/or inequality constraints, is also twice continuously differentiable and nonlinear. Successive quadratic programming methods are based on a quadratic approximation of the Lagrangian function, $L(x)$, defined by

$$L(x) = f(x) + \lambda^\top c(x) + \mu^\top c(x),$$

where λ and μ are vectors of Lagrange and Kuhn–Tucker multipliers associated with the equality and inequality constraints respectively, and attempt to solve the NLP by recursively solving a quadratic programming subproblem

$$\begin{cases} \min & g(x_k)^\top \Delta x_k + \frac{1}{2} \Delta x_k^\top B_k \Delta x_k \\ \text{s.t.} & c(x_k) + J(x_k) \Delta x_k \le 0, \end{cases}$$

where $\Delta\, x_k$ is the change in the unknown variables, B_k is some approximation to the true Hessian matrix of the Lagrangian function, $H(x) = H_f(x) + \sum \lambda_i H_{ci}(x) + \sum \mu_i H_{ci}(x)$, and where $H_f(x)$ and $H_{ci}(x)$ refer to the true Hessian matrices of the objective function and ith constraint respectively and J is the Jacobian matrix of the constraints.

Kuhn–Tucker Conditions for the Quadratic Program

The Kuhn–Tucker conditions that define stationarity in the recursive quadratic program are given by

$$\begin{aligned} B_k \Delta x_k + J_E^\top \lambda_k + J_I^\top \mu_k &= -g(x_k)\,, \\ [J_E^\top J_I^\top]^\top \Delta x_k &= -c(x_k)\,, \\ \mu_k &\ge 0\,, \end{aligned}$$

where $g(x_k)$ is the gradient of the nonlinear programming objective function, $f(x)$, evaluated at x_k, $c(x_k)$ is the set of active constraints (i. e., equalities plus inequalities that hold as equalities), J_E and J_I are the Jacobian (or first partial derivative) matrices for the equality and active inequality constraints respectively, B_k is an approximation to the Hessian (or second partial derivative) matrix of the Lagrangian function, and $\Delta\, x_k$, λ_k and μ_k represent the desired solution or change in the unknown variables, Lagrange and Kuhn–Tucker multipliers respectively. Remember, the number of inequalities in the active set can change from one quadratic programming iteration to the next as well as from one SQP iteration to the next. In many larger applications, the matrices B_k, J_E and J_I have relatively few nonzero elements (or are sparse) with a sparsity pattern that is often naturally banded with wide bandwidth. Efforts to account for this sparsity to reduce both storage and computation give rise to many auxiliary issues that must be resolved in order to produce reliable and efficient full space SQP methods.

Estimating the Hessian Matrix of the Lagrangian Function

When the number of variables is small, rank-two, quasi-Newton updating formulas like the Broyden–Fletcher–Goldfard–Shanno (BFGS) and Davidon–Fletcher Powell (DFP) updates that preserve the desired property of positive definiteness can be used to approximate the Hessian matrix of the Lagrangian function because sparsity is of no concern. On the other hand, when n is large, these and other 'full' updates can not be used because they result in Hessian matrix approximations that have essentially all nonzero elements, increasing both storage requirements and associated computational effort. Moreover, while both the sparsity of the constraint Jacobian matrix and the Hessian matrix of the Lagrangian function must be considered in developing full space SQP methods for large scale problems, it is the characteristics of the second derivatives of the Lagrangian function that most strongly effects the nature of the resulting recursive quadratic programs and the techniques that must be used to solve them.

In order to account for the sparsity of the Hessian matrix of the Lagrangian function, second partial derivatives of the objective function and nonlinear constraints are usually estimated using analytical or finite difference derivatives, sparse quasi-Newton updates like the sparse Powell-symmetric Broyden (PSB) update [23], or a mixture of analytical and quasi-Newton derivatives [13,15]. In fact, all techniques for estimating the Hessian matrix of the Lagrangian function can be put within a common framework [5] by representing the matrix B_k in the form

$$B_k = C(x_k) + A_k,$$

where $C(x_k)$ is called the computed part of the Hessian matrix, and is calculated from analytical derivatives, and A_k, the approximated part, can be computed either analytically, from finite differences or from an appropriate quasi-Newton formula. This division of the Hessian matrix of the Lagrangian function is both natural and convenient in many applications and allows the approximation of the Lagrangian Hessian matrix to be tailored for any given situation. When n is small and $C(x_k) = 0$ and A_k is updated by the 'full' DFP or full modified BFGS updates, the resulting SQP methods are those of Han and Powell. Wilson's method, which can be used for either small or large scale problems, results when $C(x_k)$ and $A_k = A(x_k)$ are calculated from analytical and/or finite difference second derivatives. On the other hand, $C(x_k) = 0$ and A_k can be updated by the sparse PSB update [23] to give a full space SQP (quasi-Newton) method that accounts for sparsity. Finally, hybrid SQP methods [15], which have their foundation in nonlinear least squares [5], result when $C(x_k)$ is computed from analytical second derivatives and A_k is calculated using some quasi-Newton update. For example, A. Kumar and A. Lucia [13] suggest a number of hybrid full space SQP methods that calculate $C(x_k)$ from analytical second derivatives and build approximations to each of the many small dense blocks of A_k by the full PSB update and iterated projections [6] using both traditional secant and auxiliary (thermodynamic) matrix constraint information. Like Wilson's method, these hybrid methods for approximating the Hessian matrix of the Lagrangian function can be used for both small and large scale problems.

Convexity

For small problems, the use of the modified BFGS update to approximate B_k gives hereditary positive definite approximations to the Hessian matrix of the Lagrangian matrix and is preferred because this guarantees that the projection of the Lagrangian Hessian matrix is positive definite on the tangent subspace defined by the linearized constraints. As a result, the recursive quadratic programs are convex (bowl-shaped) and have unique solutions, and these unique solutions usually provide descent in the nonlinear programming line search function. In contrast, when analytical and/or other quasi-Newton updates are used to build iterative approximations of the Hessian matrix of the Lagrangian function, regardless of whether the problem size is small or large, it is often difficult to guarantee hereditary positive definiteness unless the problem has certain intrinsic convexity properties or the Hessian matrix is 'corrected' to force positive definiteness. The most common type of correction for forcing positive definiteness is one in which a scalar multiple of the identity matrix, I, is added to the current approximation to the Hessian matrix of the Lagrangian function. That is, if B_k (or its projection) is determined to be indefinite (by monitoring the appropriate factors), then it

is corrected by the *Levenberg–Marquardt rule*

$$B_k = B_k + \gamma I,$$

where γ is a scalar determined from the diagonal factor of B_k. Other modifications or corrections to ensure that B_k is positive definite using Schur complements have also been suggested in [10] and applied in [1]. However, it has been shown that forcing positive definiteness can often lead to convergence to undesired (trivial) solutions [19].

Nonconvexity

In many cases, the true Hessian matrix of the Lagrangian function is positive definite on the tangent subspace defined by the constraints at a local constrained minimum but indefinite in the full space of the variables. To build a 'better' approximation to the Hessian matrix of the Lagrangian function, it is possible to allow B_k to be indefinite. However, this can also cause the projection of the Hessian matrix onto the tangent subspace of the linearized constraints to be indefinite. Consequently, the resulting recursive quadratic programs can be indefinite and require special techniques [2,16] or global optimization methods to be solved reliably. Moreover, loss of descent in the nonlinear program can still occur even though these indefinite quadratic programs are solved successfully.

Thus in many large scale applications, where sparsity must be exploited and some combination of analytical, finite difference and/or quasi-Newton second derivatives must be used, considerable attention must be paid to the resulting convexity or nonconvexity implied by the approximations of the Hessian matrix of the Lagrangian function because this will have significant ramifications, both in the methods used to solve the recursive quadratic programs and in the use of stabilization techniques to farce convergence from 'poor' (or remote) starting points.

Solving the Kuhn–Tucker Conditions for Quadratic Programming Subproblems

Full space SQP methods give rise to recursive quadratic programs in the full space of the x variables. When n is small and sparsity is of little concern, active set strategies are usually used to solve these small but full recursive quadratic programs. Moreover, when the approximation to the Hessian matrix of the Lagrangian function is hereditary positive definite, the quadratic program is convex, the rules for constraint addition and deletion are simple, both feasibility and descent in the quadratic program can be maintained under mild restrictions, and convergence to the unique solution of the quadratic program can be guaranteed [8]. This is why full quasi-Newton updates like the modified BFGS formula are preferred for small problems. Even when n is large, there is still strong incentive to 'correct' the Hessian matrix approximations for positive definiteness so that the recursive quadratic program is convex, although techniques for both large scale convex and indefinite quadratic programming and both active set (direct) and interior paint (iterative) methods for solving the recursive quadratic programs exist.

Active Set Methods

Current active set strategies are exchange-type methods [8] and are usually based on some factorization of the coefficient matrix of the Kuhn–Tucker conditions for the quadratic program depending on the properties of B_k. This means that inequalities are brought in and out of the active set (or exchanged) as part of the procedure for solving the given recursive quadratic program. This is accomplished using a combination of 'line searching' in the direction given by the current estimate of the quadratic program solution and by monitoring the signs of the Kuhn–Tucker multipliers for the inequalities. When the Hessian matrix of the Lagrangian function is positive definite, an $LDL^\top$ or Cholesky factorization of B_k (or its projection) is often used and the factors are 'updated' both symbolically and numerically as the active set within a given recursive quadratic program changes. Updating avoids complete refactorization at each quadratic program iteration and diagonal pivoting can be used to maintain numerical stability. This is true for both small and large problems. Moreover, at the beginning of any given recursive quadratic program, an initial feasible solution to the quadratic program is often generated using linear programming. While this may be desirable, in some sense, feasible starting points for the quadratic program are not strictly required. In fact, the usual strategy of initializing the active set for the current recursive quadratic program to be the final

active set on the previous SQP iteration often generates an infeasible starting point for the quadratic program, particularly when the SQP iterates are far from the optimal solution.

When the Hessian matrix of the Lagrangian function is permitted to be indefinite on the tangent subspace defined by the linearized constraints, $LDL^{\top}$ or Cholesky factorization is no longer necessarily stable and therefore can not be used. To correctly handle projected indefiniteness in solving the Kuhn–Tucker conditions for a given choice of active set, symmetric indefinite factorization [3] has been suggested [2] and modified for sparsity [14,16]. A number of pivoting strategies also exist, including threshold [14] and constrained pivoting [18], to both reduce fill-in and ensure numerical stability. Within the active set strategy, projected indefiniteness often causes 'incorrect' constraint addition and deletion, places restrictions on the order in which inequalities can be added to and/or deleted from the active set (even among members of the true final active set), can cause redundant active sets, and gives rise to multiple Kuhn–Tucker points. Thus more complicated logic is required within the active set strategy when projected indefiniteness is permitted. In particular, projected indefiniteness must be monitored from one quadratic programming iteration to the next and used as a guide for constraint addition and deletion, line searching must be permitted in both the positive and negative direction, and an inequality must not necessarily be deleted from the active set because its associated Kuhn–Tucker multiplier is negative. See [19].

Interior Point Methods

Interior point methods for solving the recursive quadratic programs have been suggested [9,11,26] and differ from active set methods in that they usually use iterative methods with some type of preconditioning (or scaling of the elements of the Jacobian and Hessian matrices) to solve the Kuhn–Tucker conditions for a modified quadratic program. In the context of quadratic programming, interior point methods are usually primal-dual (sometimes called predictor-corrector) path following algorithms that use logarithmic barrier functions. The use of a logarithmic barrier function changes the linear Kuhn–Tucker conditions for the quadratic program into a set of parameterized nonlinear Kuhn–Tucker conditions. These nonlinear stationary conditions are usually solved using some type of (truncated) Newton method, whose linear subproblems are often solved by iterative linear equation-solving techniques. Approximate linear solutions are often used in the beginning and the accuracy 'tightened' as the solution to the nonlinear equations is approached. Convexity in the quadratic program is also generally required and preconditioning of the linear equations is needed for numerical stability. Typical iterative linear equation-solving techniques used to determine the (truncated) Newton corrections include preconditioned conjugate gradient, generalized minimum residuals, and other so-called Krylov (or expanding) subspace methods, while common preconditioning techniques often are based on some partial LU factorization of the coefficient matrix. The use of iterative linear equation-solving methods is particularly advantageous in solving large scale problems because fill-in (i. e., turning zero elements into nonzero elements) is avoided in the coefficient matrix, thereby reduce both storage and overall computational workload.

Multiple QP Kuhn–Tucker Points and Related Issues

When the projection of the Lagrangian Hessian matrix is hereditary positive definite on the tangent subspace defined by the linearized constraints, the recursive quadratic programs have unique solutions (or Kuhn–Tucker points). Perhaps the single biggest difficulty associated with projected indefiniteness of the Hessian matrix of the Lagrangian function in full space SQP methods is the potential for multiple solutions to the recursive (indefinite) quadratic programs. There can be local and global minima of the quadratic programming objective function on the linearized constraint surface, as well as a saddle point solutions, each corresponding to a different set of active constraints. Moreover, the particular solution that is found is very often a function of the way in which the quadratic programming calculations are initiated (or the initial active set chosen for the quadratic program).

However, more importantly, many of these multiple solutions (even the global solution) to a quadratic program may not be descent directions for a vari-

ety of common line search functions used in nonlinearly constrained optimization, including an l_1 exact penalty function and an augmented Lagrangian function. Thus finding the global solution to a given recursive quadratic program using global optimization techniques is unjustified [17].

Loss of Descent in the Nonlinear Program

Given a valid Kuhn–Tucker point from an indefinite recursive quadratic program, loss of descent (or an uphill search direction) in the nonlinear program can still occur. However, this is a problem that has received relatively little attention in the literature because many full space SQP methods modify the Lagrangian Hessian matrix so that it is positive definite on the tangent subspace defined by the linearized constraints, thereby providing descent.

Descent in any nonlinear programming algorithm means that $g_{LS}(x_k)^\top \Delta x_k < 0$, where $g_{LS}(x_k)$ is the gradient of some suitably chosen merit function that balances satisfying the constraints with minimizing the objective function. Loss of descent means that $g_{LS}(x_k)^\top \Delta x_k \geq 0$. When line searching is used as the stabilization technique, it is essential that the change in the unknown variables provided by the quadratic programming subproblem be a descent direction of the chosen line search merit function at each SQP iteration. Failure to provide a descent direction in the context of line searching often leads to termination of the SQP algorithm due to the failure to find a 'better' point with regard to the line search function. Thus, it seems appropriate to modify the Hessian matrix of the Lagrangian function for (projected) positive definiteness to avoid subsequent failure in the line searching phase of a full space SQP method, as is done in [1,20].

In contrast, when trust region methods are used for stabilization [4,16,22,24] [7], descent at each SQP iteration is not strictly required but is desirable. Thus, many trust region methods for full space SQP methods still use positive definite Hessian matrix approximations, as well as a merit function on the trust region [22], to help ensure descent even though uniform boundedness is all that is required. Other trust region methods [17] use a linear programming subproblem to recapture descent at any given SQP iteration. For example, Lucia and J. Xu [17] solve the linear programming problem given by

$$\begin{cases} \min & g(x_k)^\top \Delta x_k \\ \text{s.t.} & J\Delta x_k \leq c(x_k), \end{cases}$$

where the Jacobian matrix, J, is comprised of the first partial derivatives for all active constraints from the quadratic programming subproblem. While this approach does sacrifice curvature information, it does have the distinct advantages of having a unique solution and providing information on how to adjust the trust region to obtain descent.

Initializing the Unknown Variables

Many studies of full space SQP methods give little attention to the way in which initial values for the unknown variables and Lagrange and Kuhn–Tucker multipliers are initialized. In fact, remote or 'poor' initial values for the unknown variables are often chosen in mathematical studies involving 'small' problems in order to test the global convergence properties of the SQP algorithm. While this provides very useful information, in large scale application-based optimization problems, the choice of starting point is an important issue with a slightly different perspective that often represents the difference between success and failure. Qualitatively correct physical information is frequently just as important as, or even more important than, the numerical values used for the unknown variables. In many physically-based optimization applications, the mathematical model (i. e., the constraints and the objective function) can wander into regions in which the model is not properly defined when initial values are chosen arbitrarily and lead to difficulties such as infeasibility, singularity, and other related problems. In addition, many physically-based nonlinear programming problems can exhibit multiple optima. However, some of these solutions are clearly undesirable in the sense that they do not represent the desired operational state of the model. Some solutions may represent local optima or saddle point solutions when a global optimum is the desired solution. To improve the chances of calculating desired optima, 'better' initial values are often used and/or coupled with the use of global optimization techniques.

Despite the fact that SQP methods are 'infeasible path' algorithms since they do not usually satisfy the nonlinear constraints at each iteration, it is often helpful to at least initiate the computations using a set of initial values that satisfies the constraint set and, if possible, represents something 'close' to the desired optimal solution. In many applications, this is often possible by solving a 'simulation' problem, in which the degrees af freedom in the optimization problem are exhausted by defining a set of 'specifications' (i. e., by adding simple equality constraints that fix values of certain variables) to give a set of n constraint equations in n unknown variables. The solution to this set of nonlinear algebraic equations often provides useful and physically meaningful initial values for the unknown variables. The initial values for the Lagrange and Kuhn–Tucker multipliers are also important in a problem-solving setting because their values effect the Hessian matrix of the Lagrangian function, the quadratic programming solution, and the line search function (if one is used). Initial values for the Lagrange and Kuhn–Tucker multipliers are often determined by computing a least squares solution to the Kuhn–Tucker conditions for the nonlinear programming using the initial values of the unknown variables and some knowledge or assumption of the initial active set of constraints. That is, initial Lagrange and Kuhn–Tucker multipliers can be obtained by solving the set of equations given by

$$JJ^{\top}\lambda = -Jg(x_k),$$

where J is the Jacobian matrix of the active constraints and λ represents the associated Lagrange and Kuhn–Tucker multipliers. Special techniques are also often employed to reduce the size and storage associated with this system of nonlinear equations [16].

Related Numerical Studies

There have been relatively few numerical studies of full space SQP methods specifically directed at large scale problems. Application areas have included mathematical [19], chemical process optimization [16,18] and aerospace problems [1] to name a few. The number of unknown variables in these studies has ranged from 5 to 60 in the mathematical studies, from 100 to 500 unknown variables in the chemical process optimization problems and to as many as 13,000 unknowns in the aerospace examples. A variety of techniques for estimating the Hessian matrix, including analytical, finite difference, quasi-Newton and a mixture of second derivatives, have been used. In some studies positive definiteness of the Lagrangian Hessian matrix has been enforced [1,20], while in others indefiniteness of the projected Hessian matrix has been permitted [16,17,18,19].

See also

- ▶ Feasible Sequential Quadratic Programming
- ▶ Optimization with Equilibrium Constraints: A Piecewise SQP Approach
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Successive Quadratic Programming
- ▶ Successive Quadratic Programming: Applications in Distillation Systems
- ▶ Successive Quadratic Programming: Applications in the Process Industry
- ▶ Successive Quadratic Programming: Decomposition Methods
- ▶ Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

References

1. Betts JT (1995) The application of optimization techniques to aerospace systems. In: Biegler LT, Doherty MF (eds) Foundations of computer aided process design. AIChE Sym Ser 304., vol 91, pp 169–177
2. Bunch JR, Kaufman L (1980) A computational method for indefinite quadratic programming problems. Linear Alg Appl 34:341–370
3. Bunch JR, Parlett BN (1971) Direct method for solving a symmetric indefinite system of linear equations. SIAM J Numer Anal 8:639–655
4. Celis MR, Dennis JE, Tapia RA (1984) A trust region strategy for nonlinear equality constrained optimization. In: Boggs PT, Byrd RH, Schnabel RB (eds) Numer. Optim., pp 71–82
5. Dennis JE, Gay DM, Welsch RE (1981) An adaptive nonlinear least-squares algorithm. ACM Trans Math Softw 7:348–368
6. Dennis JE, Schnabel RB (1979) Least change secant updates for quasi-Newton methods. SIAM Rev 21:443–459
7. Dennis JE, Vincente LN (1997) On the convergence theory

of trust region-based algorithms for equality constrained optimization. SIAM J Optim 7:927–950
8. Fletcher R (1971) A general quadratic programming algorithm. J Inst Math Appl 7:76–91
9. Gill PE, Murray W, Ponceleon DB, Saunders MA (1991) Solving reduced KKT systems in barrier methods for linear and quadratic programming. SOL Report Dept Oper Res Stanford Univ 91-7
10. Gill PE, Murray W, Saunders MA, Wright MH (1987) A Schur complement method for sparse quadratic programming. SOL Report Dept Oper Res Stanford Univ 87(12)
11. Goldfarb D, Liu S (1991) An O(n3L) primal interior point algorithm for convex quadratic programming. Math Program 49:325–340
12. Han SP (1976) Superlinearly convergent variable metric algorithms for general nonlinear programming problems. Math Program 11:263–282
13. Kumar A, Lucia A (1987) Separation process optimization calculations. Appl Numer Math 3:409–425
14. Liu JWH (1987) A partial pivoting strategy for sparse symmetric matrix decomposition. ACM Trans Math Softw 13:173–182
15. Lucia A, Miller DC, Kumar A (1985) Thermodynamically consistent quasi-Newton formulae. AIChE J 31:138 1–1388
16. Lucia A, Xu J (1990) Chemical process optimization using Newton-like methods. Comput Chem Eng 14:119–138
17. Lucia A, Xu J (1994) Methods of successive quadratic programming. Comput Chem Eng 18:S211–S215
18. Lucia A, Xu J, D'Couto GC (1993) Sparse quadratic programming in chemical process optimization. Ann Oper Res 42:55–83
19. Lucia A, Xu J, Layn KM (1996) Nonconvex process optimization. Comput Chem Eng 20:1375–1398
20. Nickel RH, Tolle JW (1989) A sparse sequential quadratic programming algorithm. J Optim Th Appl 60:453–473
21. Powell MJD (1978) A fast algorithm for nonlinearly constrained optimization calculations. In: Watson G (ed) Numerical analysis, Dundee 1977. Lecture Notes Math. Springer, Berlin, pp 144–157
22. Powell MJD, Yuan Y (1991) A trust region algorithm for equality constrained optimization. Math Program 49:189–211
23. Toint PHL (1979) On the superlinear convergence of an algorithm for solving a sparse minimization problem. SIAM J Numer Anal 16:1036–1045
24. Vardi A (1985) A trust region algorithm for equality constrained mininiization: convergence properties and implementation. SIAM J Numer Anal 22:575–59 1
25. Wilson RB (1963) A simplicial method for concave programming. PhD Diss Harvard Univ, Cambridge, MA
26. Ye YY (1989) An extension of Karmarkar's algorithm and the trust region method for quadratic programming. In: Megiddo N (ed) Progress in Mathematical Programming. Springer, Berlin, pp 49–64

Successive Quadratic Programming: Solution by Active Sets and Interior Point Methods

ANGELO LUCIA
Department Chemical Engineering,
University Rhode Island, Kingston, USA

MSC2000: 90C30, 90C25

Article Outline

Keywords
The Quadratic Program
Linear Kuhn–Tucker Conditions
for the Quadratic Program
Active Set Methods
Initialization
Addition and Deletion of Inequality Constraints
Infeasible and Redundant Constraint Sets
Matrix Factorizations and Updating
Interior Point Methods
Interior Point Formulation
Solution of the Interior Point
Optimality Conditions
Penalty Parameter Updating
Numerical Studies
See also
References

Keywords

Quadratic programming; Active set methods; Interior point methods; Sparsity; Matrix of second partial derivatives; Jacobian matrix; Convex; Nonconvex; Indefinite quadratic programs; Barrier function; Primal-dual algorithm; Central path; Truncated Newton method; Iterative linear equation-solving

Quadratic programming (QP) subproblems arise in both full space and decomposition methods of *successive quadratic programming* (SQP) where the solution to the QP is used to define the step in the unknown variables for the *nonlinear programming* (NLP) phase of the calculations. Thus reliable and efficient methods for solving quadratic programs are needed. Early methods for quadratic programming were based on a linear programming (LP) approach [1,3,17] but current

methods are either *active set strategies* [5] and [2,13] or *interior point methods* [7,11,17]. An active set is a set of equality and/or inequality constraints that hold as equalities and active set methods [5] for quadratic programming are based on a strategy for moving inequality constraints in and out of the active set during the solution of the quadratic program until a valid Kuhn–Tucker point (solution) is found. Sometimes feasible starting points are used to initiate the QP calculations but this is not strictly necessary and often impractical in the context of SQP methods. Inequalities are added to the active set based on constraint violations in the current Kuhn–Tucker direction using a 'line searching' procedure while inequalities are deleted from the active set based on the signs of Kuhn–Tucker multipliers. Constraint addition, deletion and/or exchange usually occurs one at a time. Furthermore, many active set methods require convexity (or a positive definite projection of the matrix of the quadratic objective function); however QP methods for indefinite quadratic programs [2,13] and large scale problems [13,14] also exist. For large scale problems, sparsity of the coefficient (Jacobian) matrix of the constraints and second-derivative matrix need to be exploited. In contrast, interior point methods convert a convex quadratic program into a set of nonlinear equations using logarithmic barrier functions, and solve the resulting nonlinear system of equations using Newton's method and iterative linear equation-solving techniques. The use of iterative methods to solve the linearized equations usually requires preconditioning but incurs no fill-in, making it possible to solve very large problems. Convexity is also a strict requirement for current interior point methods.

Issues that are important in solving quadratic programs by either active set methods or interior point methods include

- the rules for constraint addition, deletion and exchange, cycling, infeasibility, redundancy;
- the use of matrix factorizations, updating techniques and sparsity considerations;
- convexity and nonconvexity.

The Quadratic Program

Recursive quadratic programming problems arise from the application of SQP methods to the following nonlinear programming problem

$$\begin{cases} \min & f(x) \\ \text{s.t.} & c(x) \le 0, \end{cases}$$

where x is a vector of length n which represents an estimate of the local minimum, $f(x)$ is a twice continuously differentiable objective function and $c(x)$, the vector of equality and/or inequality constraints, is also twice continuously differentiable and nonlinear. A Lagrangian formulation

$$L(x) = f(x) + \lambda^\top c(x) + \mu^\top c(x),$$

where $L(x)$ is the Lagrangian function and λ and μ are vectors of Lagrange and Kuhn–Tucker multipliers associated with the equality and inequality constraints, respectively, gives rise to the following successive quadratic program:

$$\begin{cases} \min & g(x_k)^\top \Delta x_k + \frac{1}{2}\Delta x_k^\top B_k \Delta x_k \\ \text{s.t.} & c(x_k) + J\Delta x_k \le 0, \end{cases}$$

where $\Delta\, x_k$ is the change in the unknown variables, $g(x)$ is the gradient of the objective function, J is the first partial derivative (or Jacobian) matrix of the constraint functions, and B_k is some approximation to the true Hessian matrix of the Lagrangian function, $H(x) = H_f(x) + \sum \lambda_i H_{ci}(x) + \sum \mu_i H_{ci}(x)$, and where $H_f(x)$ and $H_{ci}(x)$ refer to the true Hessian matrices of the objective function and ith constraint respectively. It is this recursive quadratic program that is solved by active set or interior point methods, whose solution is complicated by the fact that any inequalities that hold as equalities must be determined as part of the solution procedure.

Linear Kuhn–Tucker Conditions for the Quadratic Program

The solution of a quadratic program is defined by the stationary (or linear Kuhn–Tucker) conditions for the Lagrangian function of the quadratic program given by

$$B_k \Delta x_k + J_E^\top \lambda_k + J_I^\top \mu_k = -g(x_k)$$

and

$$[J_E^\top J_I^\top]^\top \Delta x_k = -c(x_k)$$

with $\mu_k > 0$, where J_E is the Jacobian matrix of the equality constraints, J_I is the Jacobian matrix for the

active inequality constraints, and where the unknown variables in this set of linear equations are Δx_k, λ_k, and μ_k. For *convex* quadratic programs, the projection of the Hessian matrix onto the constraint surface is positive definite and the solution to the quadratic programming problem is unique. On the other hand, for *nonconvex* (or indefinite) quadratic programs multiple solutions can exist. Therefore, reliable active set or interior point methods for solving quadratic programs must find the global solution to convex QP's and at least a local solution to indefinite quadratic programming problems.

Active Set Methods

Equality constraints are always members of the active set on all QP iterations. If the active inequality constraints at the solution were known, the linear Kuhn–Tucker for the quadratic program would only have to be solved once (per SQP iteration). However, the active inequalities and therefore J_I are not known until the solution is found and this is what makes solving quadratic programs something more than just solving linear systems of equations. The linear Kuhn–Tucker conditions for the quadratic program must be repeatedly solved with a different number of equations and different number of variables, both of which reflect the number of active inequalities.

Initialization

Many active set methods for quadratic programming use a feasible starting point to initiate the calculations, which is usually determined by solving a phase I linear programming problem [5,6]. The reasons for this are that:

i) constraint feasibility can then be maintained throughout the iteration procedure;
ii) iterative comparisons of the quadratic programming objective function are meaningful since all iterates are feasible and
iii) descent from one iteration to the next can be enforced.

However, the use of a feasible starting point is not strictly necessary for convex quadratic programs and is actually impractical in SQP methods. In fact, it is a common practice in SQP methods to choose the active set for each QP subproblem to be the final active set from the previous SQP iteration and this usually gives an infeasible starting point, particularly during early nonlinear programming iterations. However, the use of the active set from the previous SQP iteration has been found to work well and therefore the calculation of a feasible starting point at each SQP iteration is an unnecessary computational expense. Direct comparison of feasible and infeasible QP starting points in SQP calculations has also shown this to be true [13].

Addition and Deletion of Inequality Constraints

The heart of any active set method for quadratic programming is the rules for adding, deleting or exchanging inequalities and in most active set strategies one 'iteration' consists of either the addition or deletion of a single inequality or the exchange of one inequality for another [5,6]. However, strategies for adding and deleting multiple inequalities have also been suggested.

Constraint addition, deletion and exchange usually occur in the following way. Given some active set, say $\{A_j\}$, which is simply a collection of inequality constraint indices, and a solution to the associated quadratic programming Kuhn–Tucker conditions for that active set, Δx_k, λ_k and μ_k, a search in the direction Δx_k is performed to determine if any of the inequalities not in the current active set are violated. This is easily accomplished by comparing the elements of the product $J_I \Delta x_k$ to the elements of the vector $-c_i(x_k)$ for each inequality not in the current active set. If $e_i^\top J_I \Delta x_k > -c(x_k)$ for $i \notin \{A_j\}$, then that inequality constraint is violated and temporarily flagged as one that must be considered for addition to the active set. Otherwise, it is not violated. Once all inequalities not in the active set have been tested, the most violated inequality (or the one in which the ratio is the smallest) is selected as the inequality constraint to add to the current active set. Constraint deletion, on the other hand, is more straightforward. The Kuhn–Tucker multipliers for all inequalities in the current active set are checked and the one with the multiplier that is most negative is identified as the inequality that should be removed from the active set. For iterations in which both addition and deletion are indicated, an exchange is usually made in which the inequality to be added replaces the one to be deleted from the current active set. These rules define the next active set $\{A_{j+1}\}$. See [5] for a more de-

tailed description of the standard rules for constraint addition, deletion and exchange.

For nonconvex or indefinite quadratic programming problems, A. Lucia et al. [13] have shown that standard rules can determine incorrect active sets when the projection of the matrix B_k onto the linear constraint surface is indefinite. In particular, nonconvexity together with the standard rules can lead to the addition of inequalities not in the final active set and/or the deletion of inequalities that truly belong to the final active set. This often leads to cycling, in which the same estimate of the active set occurs periodically after one or more active set changes, and subsequently to failure of the active set strategy and therefore the SQP method. This is further complicated by the fact that there can be multiple Kuhn–Tucker points to indefinite quadratic programs, each of which corresponds to a different active set. Thus the rules for constraint addition, deletion and exchange for indefinite or nonconvex quadratic programming problems are more complicated and often include:

i) monitoring the projected Hessian matrix and using projected indefiniteness to guide the addition of inequalities into the active set;
ii) permitting line searching for negative values of the line search parameter when the current QP Kuhn–Tucker point is in a direction of nondescent; and
iii) deleting an inequality only if the projected Hessian matrix is positive definite or the degrees of freedom are exhausted [13].

Infeasible and Redundant Constraint Sets

Many active set strategies also contain safeguards for infeasible constraint sets and constraint redundancy to avoid singularity in iterative estimates of the active set. An infeasible constraint set is one in which the collection of points satisfying the constraints is empty; thus there is no solution to the quadratic program. It is in this regard that the use of phase I linear programming techniques for generating feasible initial values for the quadratic program offer advantages since they readily identify constraint infeasibilities; however other techniques for identifying constraint infeasibilities have been proposed [13]. SQP calculations in which infeasible constraint sets have been encountered are usually continued by using some 'least error' solution for the step in the unknown variables. Otherwise, the calculations are simply terminated [15]. Redundant constraints, on the other hand, give rise to singularity in the constraint Jacobian matrix for the active set and techniques for 'trapping' and removing linearly dependent constraints are available.

Matrix Factorizations and Updating

The linear Kuhn–Tucker conditions for the quadratic program are usually solved using matrix factorizations. When the quadratic program is small in size and is guaranteed to be convex, QR, LQ or TU factorization can be used to factor the Jacobian matrix of the constraints and Cholesky factorization is used to factor either the Hessian matrix, B_k, or its projection onto the constraint surface, $Z^\top B_k Z$ [9]. Note, however, that B_k does not have to be positive definite in order for $Z^\top B_k Z$ to be positive definite, but that some care must be exercised since Cholesky factorization requires positive definiteness. For large scale quadratic programs, usually both the constraint Jacobian matrix and the Hessian matrix are sparse (i. e., contain relatively few nonzero elements). To exploit sparsity, QR factorization can not be used to factor the constraint Jacobian matrix since Q formed from Householder transformations is usually a full matrix and Z can destroy any sparsity in B_k. In this case, TU factorization is used for the Jacobian matrix and Cholesky factorization is used for the projected Hessian matrix when it is positive definite [10]. For indefinite quadratic programs, Bunch and Parlett factorization is usually used in place of Cholesky factorization of the Hessian matrix or its projection [2] to maintain numerical stability. Lucia, J. Xu and K.M. Layn [13] use Bunch and Parlett factorization for the entire coefficient matrix of the Kuhn–Tucker conditions by exploiting the sparsity of the constraint Jacobian and Hessian matrices. See [10] for guidelines for choosing factorization techniques in quadratic programming.

In order for the procedure of repeatedly solving the linear Kuhn–Tucker conditions to be efficient, complete refactorization of the associated matrices must be avoided. When constraint addition, deletion and exchange is limited to the change of one or at most two inequalities from one active set to the next, techniques that modify the necessary factors using relatively few

arithmetic operations are available. R. Fletcher [5] gives recursion relations for modifying the fundamental linear operators in quadratic programming. Other explicit and implicit formulas for modifying Cholesky, Bunch and Parlett factors [12] exist, as well as updating techniques based on Schur complement [8].

Interior Point Methods

Most interior point methods are primal-dual (sometimes called predictor-corrector) path following algorithms that use logarithmic barrier functions to change the linear Kuhn–Tucker conditions for the quadratic program into a set of parameterized nonlinear Kuhn–Tucker conditions [7,11,18]. The resulting nonlinear equations are solved using some type of (truncated) Newton method, whose linear subproblems are, in turn, solved by iterative linear equation-solving techniques. That is, approximate linear solutions are often used in the beginning and the accuracy 'tightened' as the solution to the nonlinear equations is approached. Convexity in the quadratic program is also generally required and preconditioning of the linear equations is needed for numerical stability. Iterative linear equation-solving techniques used to determine the (truncated) Newton corrections include preconditioned conjugate gradient, generalized minimum residuals, and other so-called Krylov (or expanding) subspace methods, while preconditioning is often based on some incomplete (or partial) LU factorization of the coefficient matrix. The use of iterative linear equation-solving methods is particularly advantageous in solving large scale quadratic programming problems because fill-in (i. e., turning zero elements into nonzero elements) is avoided in the coefficient matrix, thereby reducing both storage and overall computational workload.

Interior Point Formulation

Interior point methods use logarithmic barrier functions to convert a quadratic program of the form

$$\begin{cases} \min & g(x_k)^\top \Delta x_k + \frac{1}{2}\Delta x_k^\top B_k \Delta x_k \\ \text{s.t} & c(x_k) + J\Delta x_k = 0 \\ & \Delta x_k \geq 0 \end{cases}$$

into the nonlinear program

$$\begin{cases} \min & g(x_k)^\top \Delta x_k \\ & \quad + \frac{1}{2}\Delta x_k^\top B_k \Delta x_k - P\sum \log(\Delta x_i)_k \\ \text{s.t.} & c(x_k) + J\Delta x_k = 0, \end{cases}$$

where $(\Delta\ x_i)_k$ denotes the ith element of the vector Δx on the kth iteration and P is a positive penalty parameter that tends to zero as the solution to the quadratic program is approached. Note that the inequality bounds on the variables appear as part of the objective function and that the purpose of the logarithmic functions is to add a penalty to the objective function as the variables $\Delta\ x_i$ approach their bounds. Since the logarithm of a small number is negative, the term $-\ P\ \log(\Delta\ x_i)_x$ for any variable near its bound is positive. This increases the value of the objective function and thus places a 'barrier' in the way of the iterates in order to prevent them from hitting the boundaries and keep them in the 'interior' of the feasible region. Also, general inequalities can be converted to equality constraints using slack variables. Because the logarithmic barrier functions are nonlinear, the resulting Kuhn–Tucker conditions

$$B_k\Delta x_k + J^\top \lambda_k - P\sum \frac{1}{(\Delta x_i)_k} = -g(x_k)$$

and

$$J\Delta x_k = -c(x_k)$$

are also nonlinear because of the terms $1/(\Delta\ x_i)_k$. These equations are then converted into an equivalent formulation using dual variables, z_k, defined by $P\sum 1/(\Delta\ x_i)_k$, and the resulting system of nonlinear equations

$$B_k\Delta x_k + g(x_k) + J^\top \lambda_k - z_k = 0,$$
$$J\Delta x_k + c(x_k) = 0$$

and

$$z_k - P\sum \frac{1}{(\Delta x_i)_k} = 0,$$

which provide a primal-dual central path, are solved using some form of Newton's method.

Solution of the Interior Point Optimality Conditions

This last set of nonlinear equations, which define optimality for an interior point formulation of a quadratic

program, must be solved iteratively and usually Newton's method is used for this task (i. e., these conditions for the interior point formulation are linearized and corrections to the variables Δx_k, $\Delta \lambda_k$ and Δz_k are calculated). In practice, these equations are usually solved in a predictor-corrector fashion in which predictor steps account for scaling and feasibility and corrector steps perform centering to keep the iterates interior to the feasible region. Within these predictor and corrector iterations, line searching (i. e., a fraction of the Newton step) is also used to maintain feasibility and the rules for choosing the appropriate stepsizes are such that direct prediction Newton steps are taken in the limit.

The nonlinear optimality conditions for an interior point formulation can be solved accurately at each iteration. However, to improve the efficiency of interior point methods, truncated Newton methods [4] have been suggested for approximately solving these conditions. That is, optimality conditions are only solved to a 'loose' tolerance during early iterations in order to produce acceptable corrections in the variables and this tolerance is tightened as the solution to the interior point optimality conditions is approached. The primary justification for this approach is that there is no apparent need to solve the nonlinear optimality conditions accurately when the iterates are far from the solution.

To further improve efficiency, the linearized equations that come from the application of Newton's method to the optimality conditions for interior point methods are often solved using iterative linear equation-solving techniques such as preconditioned conjugate gradients or other Krylov subspace methods such as the Generalized Minimum RESiduals (GMRES) techniques of [16]. These iterative methods usually use a small number of basis vectors, to conserve storage, and are 'restarted' each time the number of iterations exceeds the number of basis vectors being stored. In addition, preconditioning is usually required for numerical stability. Preconditioning techniques are intended to improve the condition number (i. e., the ratio of the absolute value of the largest and smallest eigenvalue) of the coefficient matrix and typically some form of incomplete LU factorization is used for this purpose. The primary advantage of using iterative methods to solve the linearized equations is that they incur no fill-in and thus keep storage requirements at an acceptable level, even for very large problems.

Penalty Parameter Updating

The penalty parameter, P, must tend to zero as the solution to the quadratic program is approached to achieve rapid convergence and for this usually some 'updating' procedure is used. That is, given some initial penalty parameter, say P_0, iterative values of the penalty parameter are calculated using some formula that drives P to zero quickly. To accomplish this, generic updating formulas of the form

$$P_{k+1} = \beta P_k$$

are usually used, where β is a function of the stepsizes determined during the corrector phase of the solution to the optimality conditions.

Numerical Studies

Limited numerical results for quadratic programming can be found in the papers that introduce various aspects of active set methods [2,5] and [13]. Few general numerical results for quadratic programming in SQP methods exist and those again usually deal with specific concerns such as infeasibility, factorizations, matrix updating, etc. There are even fewer numerical results regarding the use of interior point methods, although interest in the latter is beginning to grow, and although much has been written regarding the comparison of active set and interior point methods no definitive study has been published.

See also

- ► Entropy Optimization: Interior Point Methods
- ► Feasible Sequential Quadratic Programming
- ► Homogeneous Selfdual Methods for Linear Programming
- ► Interior Point Methods for Semidefinite Programming
- ► Linear Programming: Interior Point Methods
- ► Linear Programming: Karmarkar Projective Algorithm
- ► Optimization with Equilibrium Constraints: A Piecewise SQP Approach

- ▶ Potential Reduction Methods for Linear Programming
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Successive Quadratic Programming
- ▶ Successive Quadratic Programming: Applications in Distillation Systems
- ▶ Successive Quadratic Programming: Applications in the Process Industry
- ▶ Successive Quadratic Programming: Decomposition Methods
- ▶ Successive Quadratic Programming: Full Space Methods

References

1. Beale EML (1967) An introduction to Beale's method of quadratic programming. In: Abadie J (ed) Nonlinear programming. North-Holland, Amsterdam, pp 143–153
2. Bunch JR, Kaufman L (1980) A computational method for indefinite quadratic programming problems. Linear Alg Appl 34:341–370
3. Dantzig G (1963) Linear programming and its extensions. Princeton Univ. Press, Princeton
4. Dembo RS, Steihaug T (1987) Truncated Newton algorithms for large-scale unconstrained optimization. Math Program Stud 31:43–71
5. Fletcher R (1971) A general quadratic programming algorithm. J Inst Math Appl 7:76–91
6. Gill PE, Murray W (1978) Numerically stable methods for quadratic programming. Math Program 14:349–372
7. Gill PE, Murray W, Ponceleon DB, Saunders MA (1991) Solving reduced KKT systems in barrier methods for linear and quadratic programming. SOL Report Dept Oper Res Stanford Univ 91-7
8. Gill PE, Murray W, Saunders MA, Wright MH (1987) A Schur complement method for sparse quadratic programming. SOL Report Dept Oper Res Stanford Univ 87(12)
9. Gill PE, Murray W, Wright MH (1981) Practical optimization. Acad. Press, New York
10. Goldfarb D (1975) Matrix factorizations in optimization of nonlinear functions subject to linear constraints. Math Program 10:1–31
11. Goldfarb D, Liu S (1991) An $O(n^3)$ primal interior point algorithm for convex quadratic programming. Math Program 49:325–340
12. Lucia A, Xu J, D'Couto GC (1993) Sparse quadratic programming in chemical process optimization. Ann Oper Res 42:55–83
13. Lucia A, Xu J, Layn KM (1996) Nonconvex process optimization. Comput Chem Eng 20:1375–1398
14. Nickel RH, Tolle JW (1989) A sparse sequential quadratic programming algorithm. J Optim Th Appl 60:453–473
15. Powell MJD (1978) A fast algorithm for nonlinearly constrained optimization: calculations. In: Watson G (ed) Numerical analysis, Dundee, 1977. Lecture Notes Math. Springer, Berlin, pp 144–157
16. Saad Y, Schultz MH (1986) GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear equations. SIAM J Sci Statist Comput 7:856–870
17. Wolfe P (1959) The simplex method for quadratic programming. Econometrica 27:382–398
18. Ye YY (1989) An extension of Karmarkar's algorithm and the trust region method for quadratic programming. In: Megiddo N (ed) Progress in Mathematical Programming. Springer, Berlin, pp 49–64

Supply Chain Performance Measurement

Ismail Capar[1], Burak Eksioglu[2]

[1] Department of Engineering Technology and Industrial Distribution, Texas A&M University, College Station, USA

[2] Department of Industrial and Systems Engineering, Mississippi State University, Mississippi State, USA

MSC2000: 00-02, 01-02, 03-02

Article Outline

Keywords
Introduction
Applications
Conclusions
References

Keywords

Supply chain performance measurement

Introduction

During the last two decades, many new strategies in logistics management have emerged. By introduction of these strategies, not only concepts of logistics engineering have been broadened but also new concepts such as "logistics integration" have been introduced. As the new concepts were introduced, many companies began to realize that in optimizing their logistics costs it is not sufficient to focus only on the organization itself.

It is rather compulsory to include other companies that have direct or indirect relationships with the organization. The challenge for logistics managers has become the integration of all operations across all facets of the business to improve the overall performance. The concept has become to be known as supply chain management (SCM).

As companies began managing their supply chain, they realized that they were in need of tools that will measure the combined performance of the supply chain instead of their organization alone. These much-needed tools can be referred to as "supply chain performance measurement systems" [6]. When traditional performance measurement systems are reviewed, it is seen that performance metrics are usually based on accounting systems which are not sufficient to measure overall supply chain performance. In order to overcome this insufficiency, several performance measurement systems have been developed during the last decade.

Applications

Kaplan and Norton [17] developed the concept of the balanced scorecard (BSC) that complements financial measures of past performance with measures of the drivers of future performance. According to Kaplan and Norton [17], financial and nonfinancial measures must be parts of an information system that is available to employees at all levels in an organization. Based on this approach, the BSC provides executives with a comprehensive framework that matches a company's vision and strategy with a set of performance measures which are organized into four different perspectives: financial perspective, customer perspective, internal business process perspective, and learning and growth perspective.

The supply chain operation reference (SCOR) [9] model links process elements, metrics, best practices, and the features associated with the execution of a supply chain in a unique format and provides a balanced approach to measure overall supply chain performance. The SCOR model is hierarchical with specific boundaries in regard to scope and gives definitions of every performance measure included in the model. In version 5.0 of the model, the performance measures are also intended to be hierarchical. Under the SCOR model, performance measures are classified into five groups, which are reliability, responsiveness, flexibility, cost, and assets.

Beamon [2] categorized performance measures into two groups, qualitative and quantitative measures, and uses these performance measures for supply chain design and analysis. Qualitative performance measures are those for which there is no single direct numerical measurement. Quantitative performance measures are the measures that may be directly described numerically. Beamon [2] also categorized quantitative measures as objectives based on cost and objectives based on customer responsiveness. Fill rate, product lateness, customer response time, and lead time are examples of measures based on customer responsiveness, while cost, sales, profit, inventory investment, and return on investment are defined as measures based on cost.

According to Gunasekaran et al. [14], companies often lack the insight for the development of effective measures and metrics to achieve fully integrated supply chains. The lack of insight is not only the result of an unbalanced approach between financial and nonfinancial performance measures, but is also because of an insufficient distinction among metrics at strategic, tactical, and operational levels. The authors also present a framework for measuring the performance of a supply chain after discussing some of the most appropriate performance metrics and measures. The metrics discussed in this framework are classified into strategic, tactical, and operational levels of management. The metrics are also distinguished as financial and nonfinancial so that a suitable costing method based on activity analysis can be applied.

Beamon and Chen [4] categorized performance measures into three groups: resource, output, and flexibility. The resource performance measures determine the level of resources in the system that are used to meet the objectives. The output performance measures show the effectiveness of the supply chain. The flexibility measures describe the range of possible operating conditions that are profitably achievable in the supply chain. Beamon and Chen [4] defined performance measures for each category and also ran simulations concerning the performance behavior of conjoined supply chains. In their paper, a conjoined supply chain is defined as a combination of divergent and arborescent structures. The simulation results show that the system stock-out risk, the probability distribution of the de-

mand, and the transportation time are the most important metrics in determining the effectiveness of a supply chain.

Basu [1] made a comparison of performance measures when performance criteria shift from the enterprise itself to the integrated supply chain. According to the author, the collaborative culture of the integrated supply chain has triggered the emergence of new measures, especially in five areas: external focus, power to customer, value-based competition, network performance, and intellectual capital. The author also recommends a six-step cycle to implement a performance management system and sustain the benefits of a performance management system with the new measures.

Beamon [3] evaluated and identified the limitations of supply chain performance measures such as cost, activity time, responsiveness, and flexibility. She also evaluated the use of single performance measures. According to Beamon [3], the single supply chain performance measures are attractive because of their simplicity. In addition, the author claims that current supply chain performance measurement systems are inadequate since they rely on the use of cost as a primary measure; they are not inclusive; they are often inconsistent with the strategic goals of the organization; and they do not consider the effects of uncertainty. On the basis of these insufficiencies, the author proposed a framework for measuring supply chain performance that relates supply chain performance measures to strategic goals. In addition, Beamon [3] gave a list of supply chain performance measures and their respective definition. The author also presented a quantitative approach to flexibility measurement and stated that flexibility measures are different from cost, activity time, and responsiveness measures.

Ramdas and Spekman [21] measured supply chain performance using a set of variables that capture the impact of SCM on both system-wide revenues and costs. Their evaluation was based on responses to a survey of 22 extended supply chains across five industry groups, which included life sciences, oil and gas, consumer products, agricultural and food processing utilities, and manufacturing (high-tech electronic and automotive). The authors defined six variables that reflect different approaches to measure the supply chain performance. These variables are inventory, time, order fulfillment, quality, customer focus, and customer satisfaction. Ramdas and Spekman [21] also compared functional and innovative respondents and concluded that functional product supply chains and innovative product supply chains differ significantly in thinking and practices.

Stewart [24] claims that the integration of a supply chain requires philosophical, operational, and systems changes. The author also claims that the objective of an integrated supply chain structure is minimizing non-value-adding activities and their associated structure. The author suggests that during the integration of a supply chain, four categories of operational change must be considered. These categories are structure, policy, systems, and organization. Systems should enable performance measurement. In addition, the author pointed out that the business performance metrics must support a balanced view, and that a "balanced metric" framework is necessary to measure supply chain performance. Stewart [24] also provided PRTM's Third Annual Supply Chain Performance Benchmarking Study results. The data collected for the benchmarking study cover four areas: delivery performance, flexibility and responsiveness, logistics cost, and asset management. The author identified "keys" to unlock the supply chain excellence.

Stainer [23] included productivity in the context of logistics operations and showed how productivity can be measured in this context. The author states that productivity can be seen as management of resource utilization and then proposes a framework for logistics productivity analysis that consists of five distinct dimensions of service performance: tangibles, reliability, responsiveness, assurance, and empathy. In addition, Stainer [23] states that the dimensions must be incorporated in the strategic thinking.

Bowersox and Closs [5] discussed logistics performance measures. The authors offer a framework for measuring integrated supply chain performance and benchmarking across an organization. They propose three objectives for developing and implementing performance measurement systems: monitoring measures, controlling measures, and directing measures. In addition, they defined activity-based measures and process measures. While activity-based measures focus on an individual task or process, process measures focus on the overall process throughout the supply chain. Bow-

ersox and Closs [5] defined three levels of performance measurement: internal performance measurement, external performance measurement, and comprehensive supply chain measurement. Each of these measurement systems is classified into subcategories, and the logistics performance measures are placed into these subcategories.

Miller [19] presented a hierarchical framework for capturing and linking all key performance measures. In this framework, the author differentiates measures both by their individual level in the hierarchy and by their focus. There are three hierarchical levels: strategic, tactical, and operational. Within these hierarchical levels, performance measures are differentiated into two categories: internal and external measures. While internal measures focus on efficiency and productivity, external measures focus on the effectiveness of an activity. On the basis of the framework developed, at the strategic level, a few key performance metrics will be measured to assess a company's overall performance; at the tactical level, the performance of each function will be monitored; and at the operational level, the performance of each subfunction will be monitored.

Handfield and Nichols [15] discussed the key elements in establishing a successful supply chain reengineering effort and an effective performance measurement. They defined the properties of an effective supply chain performance measurement system and gave an example framework of the BSC approach for a supply chain performance measurement system.

Lapide [18] identified that companies generally fall into the following developmental stages:

1. Functional excellence – a stage in which a company needs to develop excellence within each of its operating units, such as manufacturing, customer service, and logistics departments.
2. Enterprise-wide integration – a stage in which a company needs to develop excellence in its cross-functional processes rather than within its individual functional departments.
3. Extended enterprise integration – a stage in which a company needs to develop excellence in interenterprise processes.

Another important aspect of performance measurement is setting the correct performance targets, which should always be jointly set in the context of strategic objectives. Lapide [18] identified four methods that can be used to set performance targets: historical data based targets, external benchmark, internal benchmark, and theoretical target setting.

Hausman [16] gave information about the effects of the Internet on the supply chain. The author claims that new performance metrics should capture costs and benefits of the Internet. Hausman [16] claims that a supply chain needs to perform on three key dimensions: service, assets, and speed. In addition, the author emphasizes that supply chain performance metrics must be aligned with the business strategy.

Rolstadås [22] states that although many different performance definitions exist in the literature, these definitions can be defined by three dimensions:

1. Effectiveness: to what extent are customers' needs met.
2. Efficiency: how economically are the resources of the company utilized.
3. Changeability: to what extent is the company prepared for future changes.

Chan et al. [8] identified some in-depth problems of performance measurement systems in the supply chain context in their literature review. These problems are (1) the lack of a balanced approach in integrating financial and nonfinancial measures, (2) the lack of system thinking, in which a supply chain must be viewed as one whole entity and the measurement system should span the entire supply chain, and (3) loss of a supply chain context. Thus, the authors conclude that the existing performance measurement systems lead to local optimization. To overcome local optimization, Chan et al. [8] propose a supply chain performance measurement system with the assistance of the "analytic hierarchy process" (AHP) method. The proposed system is supposed to assess the performance of all the nodes involved along the supply chain on the basis of the core process in the simplified supply chain model. The authors propose an eight-step method that identifies and decomposes the processes involved and measures the performance. Chan et al. [7] extended the previously proposed supply chain performance measurement system by using the fuzzy set theory.

Research papers related to supply chain performance measurement have also appeared in many books for different supply chain environments, such as Geunes et al. [11,13], Geunes and Pardalos [12], and Pardalos and Tsitsiringos [20].

Similar to previous researchers, Capar [6] also proposes a supply chain performance measurement framework. In addition to customer satisfaction and financial perspectives, the author presented a new perspective. The new perspective, referred to as the supply chain collaboration perspective, considers new trends in SCM. Capar [6] also presented performance measures for the supply chain collaboration perspective. The metrics are classified as strategic, tactical, and operational in order to determine the corresponding management level that deals with the metrics. Furthermore, the author discussed an appropriate supply chain performance measurement system for a large Turkish automotive company that manufactures passenger cars, light commercial vehicles, and related components.

Conclusions

According to a multiyear study of supply chain excellence at Michigan State University, performance measurement is one of the top four drivers of supply chain excellence [10]. This study also brings out the importance of neglected supply chain performance measurement during the supply chain transformation efforts. The research study reached the conclusion that successful supply chain transformation efforts via effective supply chain performance measurement are increasing.

Easton et al. [10] also mentioned that sufficient performance measurement systems should possess the following properties:

1. Measures should be directly tied to operational effectiveness and efficiency.
2. Measures should relate important strategic objectives and nonfinancial performance.
3. Measures should provide a forward-looking perspective.

As the authors state, efficiency is the most weighted dimension of the majority of measurement systems, and it is intuitive for companies to focus on efficiency. However, companies should not neglect the measurement of the integrated supply chain performance.

References

1. Basu R (2001) New criteria of performance management: A transition from enterprise to collaborative supply chain. Meas Bus Excell 5(4):7–12
2. Beamon BM (1998) Supply chain design and analysis: models and methods. Int J Product Econom 55(3):281–294
3. Beamon BM (1999) Measuring supply chain performance. Int J Oper Product Manag 19(3):275–292
4. Beamon BM, Chen VCP (2001) Performance analysis of conjoined supply chains. Int J Product Res 39(14):3195–3128
5. Bowersox DJ, Closs DJ (1996) Logistical management: The integrated supply chain process. McGraw-Hill, New York
6. Capar I (2002) A supply chain performance measurement system: A case study in automotive industry. Master's thesis, Sabanci University
7. Chan FTS, Qi HJ, Lau HCW, Ip RWL (2003) A conceptual model of performance measurement for supply chains. Management Decision 41(7):635–642
8. Chan FTS, Qi HJ, Lau HCW, Ip RWL (2004) Supply Chain Performance Management – Concepts and Cases, chapter An evaluation method on the performance measurement of supply chains. ICFAI University Press, June
9. Supply Chain Council (2002) Supply-chain operations reference-model. http://www.supply-chain.org, Version 5.0
10. Easton R, Thurwachter B, Zhang TB (2002) Seizing the supply chain opportunity in asia. Achiev Supply Chain Excell Technol 4:52–56
11. Geunes J, Akcali E, Pardalos PM, Romeijn HE, Shen ZJ (eds) (2004) Applications of Supply Chain Management and E-commerce Research. Kluwer, Dordrecht
12. Geunes J, Pardalos PM (eds) (2003) Supply Chain Optimization. Kluwer, Dordrecht
13. Geunes J, Pardalos PM, Romeijn HE (eds) (2002) Supply Chain Optimization: Applications and Algorithms. Kluwer, Dordrecht
14. Gunasekaran A, Patel C, Tirtiroglu E (2001) Performance measures and metrics in a supply chain environment. Int J Oper Product Manag 21:71–87
15. Handfield RB, Nichols EL (1999) Introduction to supply chain management. Prentice-Hall, New Jersey
16. Hausman WH (2003) The practice of supply chain management: Where theory and application converge. In: Supply chain performance metrics. Kluwer, Dordrecht, pp 61–76
17. Kaplan RS, Norton DP (1996) Translating strategy into action: The Balanced Scorecard. Harvard Business School Press, Boston
18. Lapide L (1998) What about measuring supply chain performance. Advanced Manufacturing Research, pp 287–297
19. Miller T (2002) Hierarchical Operations and Supply Chain Planning. Springer, Berlin
20. Pardalos PM, Tsitsiringos VK (eds) (2002) Financial Engineering, E-commerce and Supply Chain. Kluwer, Dordrecht
21. Ramdas K, Spekman RE (2000) Chain or shackles: Understanding what drives supply chain performance. Interfaces 30(4):3–21
22. Rolstadås A (1995) Performance management: A business process benchmarking approach. Chapman and Hall, London

23. Stainer A (1997) Logistics- a productivity and performance perspective. Supply Chain Management: Int J 2(2):53–62
24. Stewart G (1995) Supply chain performance benchmarking study reveals keys to supply chain excellence. Logistic Inform Manag 8(2):38–44

Survivable Networks

Chi-Geun Han
KyungHee University, Seoul, Korea

MSC2000: 90-XX

Article Outline

Keywords
SNDP with Traffic Capacity
A Heuristic for SNDP
See also
References

Keywords

Survivability; Local search; Node-disjoint path; Edge-disjoint path; Network design

As the modern telecommunications need networks that support fast and reliable data transmissions, the optic fiber networks (SONET: Synchronous Optic Networks) are widely used and replace the old copper based networks. According to the characteristics of the optic fiber networks, designing networks so that it can survive from any failure of the networks which comes from a node failure or a link failure is an important issue that we have to consider [1].

Malfunctions of the networks result from node or link failures and these failures come from natural disasters such as earthquakes or incidents such as cutting by ground digging or fire. Since much of modern business depends on the telecommunication networks, the networks should be safe even if there is a damage in some place of the networks. The survivable network is a network that can perform its function properly even if there are node or link failures on the network. Practically, the node means a telecommunication center, a switching point, or a city and the link denotes a cable between pair of nodes.

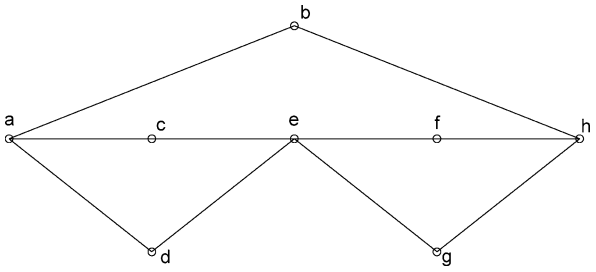
For an undirected graph $G = (V, E)$, where V is a set of vertices (nodes) and E is a set of edges (links) in the graph, the survivability of the networks is defined as the number of *node-disjoint paths*, r_{ij}, for pairs of nodes i, j. In other words, from i to j or from j to i in the graph, the communication path is safe until r_{ij}-1 nodes of the network are malfunctioning.

Also, *edge-disjoint paths* can be defined similarly. If two edge-disjoint paths are found between i and j, the two paths do not share some edges in these paths, but they may use the same node in the paths. Hence, the node-disjoint constraint is stricter than the edge-disjoint constraint. Note that if there are r_{ij} node-disjoint paths between i and j, we can always find r_{ij} edge-disjoint paths between them. But, the reverse is not true.

In Fig. 1, for example, three paths, $a - b - h$, $a - c - e - f - h$, and $a - d - e - g - h$ between a and h, are edge-disjoint and $a - b - h$ and $a - d - e - g - h$ are node-disjoint paths. That means at least three edge failures or two node failures are needed for interrupting telecommunications service between a and h.

In general, the number of node(edge)-disjoint paths for some specific nodes, a and b, can be found easily using *max–flow algorithms*. If flow capacities of all edges are infinite and those of all nodes on the network is 1, maximum flow obtained by a max–flow algorithm is equal to the number of node-disjoint paths between a and b. If flow capacities of all edges are 1 and flow capacities of all nodes are infinite, the maximum flow should be the number of edge-disjoint paths between a and b on the network.

Now, we consider only the node-disjoint constraints since the edge-disjoint constraints are included in the node-disjoint constraints. The node(edge)-dis-



Survivable Networks, Figure 1
Survivable networks

joint constraints may be called connectivity constraints. If $r_{ij} = n - 1$, $\forall i, j \in V$, where n is the number of nodes in the graph, the survivable network is a complete graph and there are $n - 1$ node disjoint paths including a direct connection between a and b. However, it is not a good idea to establish such an expensive network. Since implementing a network that has more links than an optimal network that has the necessary number of links to provide desirable survivability is not cost effective, we have to consider the trade-off between the cost and network redundancy in designing a network topology.

For a given $G = (V, E)$, cost matrix $c(i, j)$, $i, j = 1, \ldots, n$, and the requirements of the number of node-disjoint paths between each pair of nodes $i, j \in V$, r_{ij}, the *Survivable network design problem* (SNDP) is to find the minimum cost edge set such that it guarantees that the numbers of node-disjoint paths between pairs of nodes are all greater than or equal to r_{ij}, for all $i, j = 1, \ldots, n$. If $r_{ij} = 2$, for all $i, j = 1, \ldots, n$, then a *ring topology* is a solution of the network. Since all pairs of nodes in a ring should have two node-disjoint paths, the network is safe with any one node failure. For designing such a ring structure, Net Solver was developed by L.M. Gardner, I.H. Sudborough, and I.G. Tollis [2].

Many studies have been done for these problems and they can be categorized into two classes:

- algorithms that obtain the optimum solution
- *heuristic algorithms* that get a near optimum solution

For the former, various *integer programming* (*IP*) and *linear programming* (*LP*) techniques are used and for the latter, *local search methods* are studied.

M. Grötschel and C.L. Monma [4] formulated the problems as integer program and studied integer polyhedra of k-edge(node) connected network design problems. Also, Grötschel, Monma, and M. Stoer [5] presented computational results obtained by solving the SNDP using a *cutting plane* method of the IP. They considered the SNDP with low connectivity constraints, i. e., $r_{ij} = 0, 1, 2$, for all i, j. M.X. Goemans and D.J. Bertsimas [3] established the parsimonious property of the problems formulated in the LP relaxation.

K. Steiglitz, P. Weiner, and D.J. Kleitman [9] proposed a local search method for the general SNDP. Monma and D.F. Shallcross [6] used several heuristic techniques for obtaining initial solutions and for improving the solutions of two-connected survivability constraints problems. T.S. Wu explained the network survivability in detail in [10] and many features of hardware aspects are included in the book.

SNDP with Traffic Capacity

In the practical situation, we also have to consider telecommunication traffic on the network in addition to the network topology. In order to guarantee the steady service when some node or link failures occur on the network, we have to decide the cable capacity in a link. For some specific pair of large cities, the traffic may be higher than that of some pair of small cities. Consider the following situation: a link that has large capacity between two large cities A and B is broken down and a secondary path that the survivable network provides is used. But the secondary path uses a link that has not an enough capacity for the traffic between A and B. If such a case happens, the service between A and B should be interrupted and the network is not safe any more. Hence, the capacity of a link should be determined in designing the network as well as the survivability.

For this kind of problems, many approaches have been proposed. J. Yamada [11] proposes an algorithm to design efficient spare path networks. It is a heuristic and to achieve near-optimization. J. Shi and J. Fonseka [8] studied a class of traffic-based survivability measures and survivability analysis of telecommunications networks. I. Ouveysi and A. Wirth [7] used a maximum *spanning tree* with traffic requirements and could provide a survivable network that can operate with at most two link failures.

A Heuristic for SNDP

Now, we are going to explain a heuristic method [9] (called Alg1) for the general SNDP in detail.

Alg1 is composed of 2 parts:

1) stage of obtaining initial feasible solutions
2) stage of local improvements of the initial solutions

Since the SNDP has many local minima, by performing a local search starting from several initial solutions and obtaining many local minima, Alg1 may get a near optimal solution of the SNDP. The local search method is trying to find a direction to minimize an objective function. If such a point is found and it is feasible, it moves

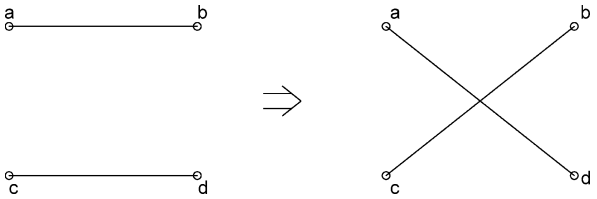
to the new point and continues the same procedure until no further improvements can be made. Since the SNDP has multiple local minima, the procedure may stop at a local minimum and further improvements are impossible. If then, it finds a local minimum and the whole procedure starts again with another initial feasible solution. Since it searches local area, it is called the local search. In order to increase the possibility of finding the global minimum, it has to search whole area of the feasible region and it needs many initial solutions. However, it is not cost-effective to investigate the whole feasible region and the trade-off between computation time and quality of solutions. The following is the detail procedure of Alg1.

```
DO k = 1 (number of local minima to be found)
    find an initial feasible solution
    WHILE local improvements are possible
        DO X-change (local improvement)
        move to the new feasible solution
    END WHILE
    keep the local minimum
END DO
RETURN the lowest cost solution among the local minima
```

Program Alg1

For obtaining an initial solution, Alg1 defines a requirement array p, $p_i = \max_j r_{ij}$, $j = 1, \ldots, n$, and we select at least p_i edges connected to a node i for constructing an initial solution. It uses the property that the degree of node i must be at least the maximum of node requirements between i and all other nodes in order to guarantee that there are r_{ij} node-disjoint paths between nodes i and j. By randomizing the sequence of the node numbers, Alg1 may get many distinct initial solutions. These initial solutions may satisfy the node connectivity constraints or not. The node connectivity constraints can be tested by using a max–flow algorithm and if there is a max-flow $f \geq r_{ij}$ between nodes i and j, the initial solution that satisfies the node connectivity constraints becomes an initial feasible solution and is used for the next stage, local improvements.

The local improvements are mainly composed of a series of X-changes (see Fig. 2).



Survivable Networks, Figure 2
X-change

In Fig. 2, the lines between two nodes mean the edges are selected for cabling. If $c(a, b) + c(c, d) > c(a, d) + c(b, c)$, the X-change can be done and the total cost can be decreased. However, it is necessary to test the network after an X-change still keeps its feasibility of the node-connectivity constraints since two links are deleted and it can affect the number of node-disjoint paths for some pair of nodes. If we have to test node-connectivity constraints for all pairs of nodes $i, j = 1, \ldots, n$, it must take $O(n^2)$ steps and be time consuming. But, according to [9, Thm. 2], only some of $O(n^2)$ pairs could be tested after an X-change for ensuring the node-connectivity constraints.

For the SNDP with connectivity constraints $r_{ij} = 1, 2$ for all $i, j \in V$, Monma and Shallcross [6] proposed a specific method for obtaining initial solutions and local improvements. For the SNDP with node connectivity $r_{ij} > 2$ for some pair of nodes i, j, a special method for initial solutions and local improvements has not been studied until now.

As much of the modern life depends on telecommunications, the importance of the survivable networks is increasing rapidly. Also the structure of the SNDP will be complex in order to model the complicated requirements of telecommunications in the real world. Hence, new classes of SNDP will be introduced and they should deserver to be challenged by many researchers.

See also

- ► Auction Algorithms
- ► Communication Network Assignment Problem
- ► Directed Tree Networks
- ► Dynamic Traffic Networks
- ► Equilibrium Networks
- ► Evacuation Networks
- ► Generalized Networks
- ► Maximum Flow Problem

- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Traffic Network Equilibrium

References

1. Cardwell RH, Monma CL, Wu TH (1989) Computer-aided design procedures for survivable fiber optic networks. IEEE J Selected Areas in Comm 7(8):1188–1197
2. Gardner LM, Sudborough IH, Tollis IG (1995) Net solver: A software tool for the design of survivable networks. In: Proc. Globecom'95, vol 2, pp 926–930
3. Goemans MX, Bertsimas DJ (1993) Survivable networks, programming relaxations and the parsimonious property. Math Program 60:145–166
4. Grötschel M, Monma CL (1990) Integer polyhedra arising from certain network design problems with connectivity constraints. SIAM J Discret Math 3(4):502–523
5. Grötschel M, Monma CL, Stoer M (1992) Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. Oper Res 40(2):309–330
6. Monma CL, Shallcross DF (1989) Methods for designing communications network with certain two-connected survivability constraints. Oper Res 37(4):531–541
7. Ouveysi I, Wirth A (1995) An efficient heuristic for the design of a survivable network architecture for dynamic routing. In: Proc. Globecom'95, vol 2, pp 921–925
8. Shi J, Fonseka J (1995) Traffic-based survivability analysis of telecommunication networks. In: Proc. Globecom '95, vol 2, pp 936–940
9. Steiglitz K, Weiner P, Kleitman DJ (1969) The design of minimum-cost survivable networks. IEEE Trans Circuit Theory ct-16(4):455–460
10. Wu TS (1992) Fiber network service survivability. Artech House, Norwood
11. Yamada J (1995) A spare capacity design method for restorable networks. In: Proc. Globecom'95, vol 2, pp 931–935

Symmetric Systems of Linear Equations

ALLEN HOLDER
University Colorado, Denver, USA

MSC2000: 65K05, 90Cxx

Article Outline

Keywords
See also
References

Keywords

Symmetric matrix; Optimization

Let A be an $n \times n$ *symmetric matrix* and b be a column vector of length n. Then the system of linear equations

$$Ax = b,$$

where x is a column vector of length n, is a symmetric system of linear equations. To demonstrate what a symmetric matrix is, consider the following two matrices,

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & -1 & -4 \\ 3 & -4 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 3 \\ 7 & -1 & -4 \\ 5 & -3 & 1 \end{pmatrix}.$$

The matrix on the left is symmetric because an element in row i and column j is equivalent to the element in row j and column i. The matrix on the right does not have this property and is therefore not symmetric. Although the Chinese investigated linear systems of equations around 250 BC, the modern study of systems of linear equations was begun in the late 17th century by G.W. Leibniz. The solution techniques of this time were developed through the use of determinants, and the idea of a matrix was not introduced until 1850 by J.J. Sylvester. In 1855, the English mathematician A. Caley published the first article concerned with the algebra of matrices and it was Caley that defined what it meant for a matrix to be symmetric.

Symmetric systems of linear equations often arise when dealing with optimization problems. For example, many common optimization algorithms, such as *gradient descent*, *quasi-Newton methods*, and *Newton's method*, use a solution to a symmetric linear system to decide a direction in which to search for the next iterate. Consider the unconstrained optimization problem

$$\min_{x \in \mathbf{R}^n} f(x),$$

where f is a twice continuously differentiable function from $\mathbf{R}^n$ to $\mathbf{R}$. For any $x \in \mathbf{R}^n$, the search direction, Δx, which solves

$$S\Delta x = -\nabla f(x),$$

leads to gradient descent if S is the identity matrix, Newton's method if S is the Hessian of f at x, and quasi-Newton methods if S is an approximation of the Hessian of f. In all of these cases S is symmetric, and one can see that common optimization routines require the solution to a symmetric system of linear equations. Another optimization problem requiring the solution to a symmetric system of linear equations is the unconstrained, full rank, *least squares problem*. This problem is

$$\min_{x \in \mathbf{R}^n} \|Bx - b\|_2 ,$$

where B is an $(m \times n)$-matrix, $m \geq n$, and the rank of B is n. The unique solution to this problem is the solution to the symmetric system

$$B^\top Bx = B^\top b.$$

Symmetric matrices have several desirable eigenvalue and eigenvector properties. One of these properties is that the 2-norm of a symmetric matrix is the *spectral radius*. In other words, if $\lambda_{\max}$ is the largest eigenvalue of the symmetric matrix A, then

$$\|A\|_2 = \lambda_{\max}.$$

Two of the most important eigenvalue properties, shown by Ch. Hermite in 1855, are that symmetric matrices have real eigenvalues and are unitarily similar to a diagonal matrix, whose main diagonal contains the eigenvalues of the matrix. This last result relies on the fact that the eigenspaces of an n by n symmetric matrix, A, contains an orthonormal bases for $\mathbf{R}^n$. Using such a basis as the columns of a matrix, say Q, we have the similarity relation,

$$A = QDQ^\top.$$

If this factorization of A is known, then the system $Ax = b$ may be easily solved by first solving

$$Dy = Q^\top b,$$

and then setting $x = Qy$. Although this diagonal factorization is in general expensive, it is useful when dealing with quasi-Newton methods. These methods often update the current approximation of the Hessian by adding a rank one matrix, so that, in the following iteration, the new matrix produces a 'better' search direction. Hence, if S is the symmetric approximation to the Hessian of f for the current iteration, the next iteration uses $S + vv^\top$, where v is defined by the algorithm. The result needed to analyze and implement these quasi-Newton methods is due to Ch. Loewner. The theorem is known as the *interlocking eigenvalue theorem*, and it shows the relationship between the eigenvalues of the symmetric matrices S and $S + vv^\top$. It states that if the eigenvalues of S are

$$\lambda_1 \leq \cdots \leq \lambda_n,$$

and the eigenvalues of $S + vv^\top$ are

$$\sigma_1 \leq \cdots \leq \sigma_n,$$

then

$$\lambda_1 \leq \sigma_1 \leq \lambda_2 \leq \sigma_2 \leq \cdots \leq \lambda_n \leq \sigma_n.$$

Essentially this demonstrates that if a symmetric matrix is formed as the sum of two symmetric matrices, one of which is rank one, then the eigenvalue structure remains somewhat intact. This leads to preconditioning and scaling routines that make quasi-Newton methods robust. Furthermore, subsequent results show how to efficiently obtain the diagonal factorization for $S + vv^\top$ from the factorization of S. This means that solving the symmetric system of equations for the next iteration is relatively cheap when the diagonal factorization of S is known.

Many methods, other than the diagonal factorization used above, have been suggested to solve symmetric systems of linear equations. When the A matrix is positive definite, the factorization of choice is usually the *Cholesky factorization*. However, when the matrix is indefinite, other schemes must be used. In 1846 C.G. Jacobi showed how to use *rotations* to solve a symmetric system of linear equations. This method is receiving recent attention because it is inherently parallel. Another method, presented by J. Aasen in 1971, is to permute the rows and columns of A and then decompose this

matrix into a *tridiagonal matrix* instead of a diagonal matrix. The exact factorization is

$$PAP^\top = LTL^\top,$$

where P is a permutation matrix, L is a lower triangular matrix, and T is a tridiagonal matrix. Now, to solve $Ax = b$ one solves the following sequence of problems: $Lz = Pb$, $Tw = z$, $L^\top y = w$, and then sets $x = Py$.

See also

- ▶ ABS Algorithms for Linear Equations and Linear Least Squares
- ▶ Cholesky Factorization
- ▶ Gauss, Carl Friedrich
- ▶ Interval Linear Systems
- ▶ Large Scale Trust Region Problems
- ▶ Large Scale Unconstrained Optimization
- ▶ Linear Programming
- ▶ Orthogonal Triangularization
- ▶ Overdetermined Systems of Linear Equations
- ▶ QR Factorization
- ▶ Solving Large Scale and Sparse Semidefinite Programs

References

1. Aasen J (1971) On the reduction of a symmetric matrix to tridiagonal form. BIT 11:233–242
2. Gittleman A (1975) History of mathematics. Charles E. Merrill, Columbus
3. Golub G, Charles V (1989) Matrix computations. Johns Hopkins Univ. Press, Baltimore, MD
4. Kline M (1972) Mathematical thought from ancient to modern times. Oxford Univ. Press, Oxford
5. Loewner C (1934) Uber monotone Matrixfunktionen. Math Z 38:177–216
6. Luenberger D (1984) Linear and nonlinear programming. Addison-Wesley, Reading, MA

T

Theorems of the Alternative and Optimization

Franco Giannessi
Department Math., University of Pisa, Pisa, Italy

MSC2000: 90C05, 90C30

Article Outline

Keywords
Farkas Lemma
See also
References

Keywords

Theorem of the alternative; Separation theorems; Image space; Transposition theorem; Lagrange multipliers; Lagrange function; Dual space; Dual problem

In a general format a *theorem of the alternative* (TA) claims that, between two given propositions, say S and S^*, one and only one is true. In mathematics S and S^* are, in general, systems of equalities or inequalities. A TA for linear algebraic systems was established as early as 1873 by P. Gordan [11]; then there was the celebrated Farkas lemma in 1902 [7] (cf. also ► Farkas lemma; ► Farkas lemma: Generalizations); indeed, such a lemma does not appear as a TA, but an obvious reformulation shows it as a TA. Some further important TA were established in 1915 by E. Stiemke [22], in 1936 by T.S. Motzkin [19], in 1951 by M.L. Slater [21], in 1956 by A.W. Tucker and in 1956 by R.J. Duffin (see [17]). Subsequently, due mainly to the development of the optimization theory, there has been a blooming of TAs; they have been extended to not necessarily algebraic systems, to systems in an infinite-dimensional space, to systems in a complex space, and even to systems for point-to-set maps. TA (sometimes called *transposition theorems*) have been conceived as tools for proving some theorems of linear algebra (this is the reason why the Farkas TA is known as a lemma) or to prove the existence and uniqueness of solutions of differential and integral equations [24].

It is interesting to note that, a few years later, in a completely different field of mathematics, some ideas mature which lead to state so-called separation theorems (ST). Indeed, here too, the first important result does not look like an ST: on the basis of some ideas of E. Helly in 1912 [14], S. Banach in 1925 [2] and H. Hahn in 1927 [12], independently of each other, establish the celebrated *Hahn–Banach linear extension theorem*; by means of an obvious reformulation it shows itself to be a ST. Here too the purpose is to have lemmas for proving other theorems – in functional analysis and geometry.

Over several years TA and ST have been carried on as disjoint theories. Recently, thanks to the great development of optimization and to the increasing use of TA and ST in the theory of optimization, it has been recognized that TA and ST are different 'languages' for expressing thesame 'structural' property (this does not imply that one of them should be deleted; on the contrary, different languages let us achieve more properties) and, overall, that they are not only tools for proving theorems; indeed, they have been raised to the basis for the theory of constrained extrema.

After a short review of some TA, their application to prove fundamental theorems of optimization will be shown. Then, we will briefly describe the recent approach to the theory of constrained extrema which is

based on TA and ST. Matrices and vectors will be real-valued.

Farkas Lemma

Let A be a matrix of the order $m \times n$, a be a row n-vector, and x a column n-vector. $Ax \geq 0$ implies $ax \geq 0$ if and only if there exists a nonnegative row m-vector z such that $zA = a$.

This lemma receives a useful vector interpretation. The rows of A can be seen as vectors of $\mathbf{R}^n$; call C the (convex) cone generated by them, and set $C^* := \{x \in \mathbf{R}^n : Ax \geq 0\}$. Since the elements of C are the only vectors which have a nonnegative scalar product with each vector of C^*, then a must belong to C.

Farkas lemma can be equivalently formulated as TA:

Theorem 1 *Let us adopt the same notation of Farkas lemma, and let z be a row m-vector. Between the systems (in the unknowns x and z):*

$$S_1 : \quad Ax \geq 0, \quad ax < 0$$

and

$$S_1^* : \quad zA = a, \quad z \geq 0$$

one and only one has solutions.

System S_1^* introduces a new variable and a new space – i. e., that where z runs – which can be called *dual space* of that where x runs, as we will see later.

From Therorem 1 we immediately deduce another TA.

Theorem 2 *Let A and B be matrices respectively of the orders $m \times n$ and $p \times n$, u a row m-vector, and v a row p-vector. Between the systems (in the unknowns x and (u, v)):*

$$S_2 : \quad Ax \leq 0, \quad Bx < 0$$

and

$$S_2^* : \begin{cases} uA + vB = 0, \\ u \geq 0, \quad v \geq 0, \quad v \neq 0 \end{cases}$$

one and only one has solutions.

The possibility of both S_2 and S_2^* leads to that of inequality $(uA + vB)\ x < 0$ which contradicts the equation in S_2^*. Let e be the column p-vector whose entries equal 1; because of Therorem 1 the impossibility of S_2(which is equivalent to that of system $Ax \leq 0$, $Bx + et \leq 0$, $t > 0$ with $t \in \mathbf{R}$ such a system is easily identified to be of type S_1) implies the possibility of system (we set $z = (u, v)) - uA - Bv = 0, -ev = -1, u \geq 0, v \geq 0$, which shows the possibility of S_2^*.

A vector interpretation quite analogous to the one above can be given for Therorem 2. At $A = 0$ Therorem 2 becomes the TA stated by Gordan. Now, let us show, by means of classic instances, how TA have been exploited for proving fundamental theorems on constrained extrema. To this end, consider the following minimization problem with bilateral constraints:

$$P_1 : \quad \begin{cases} \min & f(x), \\ \text{s.t.} & g(x) = 0, \end{cases}$$

where the function f:$\mathbf{R}^n \to \mathbf{R}$ and the column vector function g:$\mathbf{R}^n \to \mathbf{R}^m$ are differentiable at least at $\overline{x} \in \mathbf{R}^n$. Let ∇ f(x) denote the row n-vector gradient of f at x and $\nabla\, g(x)$ the $m \times n$ Jacobian matrix of g at x.

It is well known (see [17,20]) that, under suitable assumptions (e. g., $\nabla g(\overline{x})$ has maximum rank and ∇ g(x) is continuous around $\overline{x}$, if $\overline{x}$ is a (local) minimum point for P_1, then the directional derivative of f is nonnegative along each direction d of the linear manifold $\nabla g(\overline{x})d = 0)\ d = 0$ (beside, of course, along each feasible direction; s is a column n-vector) which is tangent, at $(\overline{x})$, to the (nonlinear) manifold g(x) = 0. This means that ∇ g$(\overline{x})d = 0$ implies $\nabla f(\overline{x})d \geq 0$, or

$$\begin{pmatrix} \nabla g(\overline{x}) \\ -\nabla g(\overline{x}) \end{pmatrix} d \geq 0 \quad \text{implies} \quad \nabla f(\overline{x})d \geq 0.$$

By setting $A = \begin{pmatrix} \nabla g(\overline{x}) \\ -\nabla g(\overline{x}) \end{pmatrix}$ and $a = \nabla f(\overline{x})$, Farkas' lemma can be applied and its thesis means now the existence of a nonnegative row (2m) -vector, say $y = (y', y'')$ with $y', y'' \in \mathbf{R}_+^m$, such that $(y' - y'')\ \nabla g(\overline{x}) = \nabla f(\overline{x})$, or

$$\nabla f(\overline{x}) = \overline{\lambda} \nabla g(\overline{x}),$$

where $\overline{\lambda} := y' - y''$. Hence, by means of a TA, we have achieved the existence of a vector $\overline{\lambda}$) (whose elements are known as *Lagrange multipliers*), such that the pair $(\overline{x}, \overline{\lambda})$ is a stationary point of the Lagrangian function $L(x; \lambda) :=$ f(x) $- \lambda$ g(x).

Now, consider the following minimization problem with unilateral constraints:

$$P_2: \quad \begin{cases} \min & f(x), \\ \text{s.t.} & g(x) \geq 0 \end{cases}$$

where f and g are as in P_1.

In 1948 F. John, under the assumption that f and g be differentiable at least at $\overline{x}$, proved the following necessary condition (see [17,20]): if $\overline{x}$ is a (local) minimum point for P_2, then there exist $\overline{\theta} \in \mathbf{R}_+$ and $\overline{\lambda} \in \mathbf{R}_+^m$ with $\overline{\theta}, \overline{\lambda}) \neq 0$, such that $(\overline{\theta}, \overline{\lambda})$ is a solution of the system (in the unknowns $\theta \in \mathbf{R}$ and the row vector $\lambda \in \mathbf{R}^m$):

$$FJ: \quad \theta \nabla f(\overline{x}) - \lambda \nabla g(\overline{x}) = 0, \quad \lambda g(\overline{x}) = 0.$$

To show this condition, let us set ($\top$ as superscript denotes transposition) $g(x)^\top = (g_1(x), \ldots, g_m(x))$, $\lambda = (\lambda_1, \ldots, \lambda_m)$ and introduce the sets

$$I := \{1, \ldots, m\},$$
$$I^0 = I^0(\overline{x}) := \{i \in I: \ g_i(\overline{x}) = 0\}.$$

If the thesis is false, then the (linear homogeneous) system

$$\begin{cases} \theta \nabla f(\overline{x}) - \sum_{i \in I^0} \lambda_i \nabla g_i(\overline{x}) = 0, \\ \theta \geq 0, \quad \lambda_i \geq 0, \quad i \in I^0, \\ (\theta, \lambda_i, \ I \in I^0) \neq 0 \end{cases} \tag{1}$$

has no solution (in fact, if (1) had a solution, say $(\theta^*, \lambda_i^*, i \in I^0)$, then, by setting $\lambda_i^* = 0$, $i \in I \backslash I^0$, and $\lambda^* = (\lambda_1^*, \ldots, \lambda_m^*)$, the pair (θ^*, λ^*) should be a solution of *FJ*). With the positions

$$A = 0, \quad B = \begin{pmatrix} \nabla f(\overline{x}) \\ -\nabla g_i(\overline{x}), \ i \in I^0 \end{pmatrix},$$

$v = (\theta, \lambda_i, i \in I^0)$, (1) is identified with S_2^*. Hence, from Therorem 2 we deduce that S_2 has solutions, or that there exists a column n-vector $\overline{y}$, such that

$$\nabla f(\overline{x})\overline{y} < 0, \quad \nabla g_i(\overline{x})\overline{y} > 0, \quad i \in I^\circ.$$

These inequalities would mean that $\overline{y}$ is a feasible direction along which the directional derivative of f at $\overline{x}$ is negative; this fact, according to a well known Linearization Lemma [1,17], contradicts the assumption that $\overline{x}$ be a minimum point.

The above Lagrange and John necessary optimality conditions show how a TA has been classically used, and hence the reason why they have been conceived. However, TA (and ST) possess a much greater potential than that exploited for proving theorems. To explain, even if in short, this aspect let us consider again P_2 and assume now that f be convex and g concave, but not necessarily differentiable, so that P_2 is a convex problem (minimization of a convex function over a convex domain). Let us set $\varphi(\overline{x}; x) := f(\overline{x}) - f(x) :=$ by the very definition of minimum it is trivial to claim that a feasible $\overline{x} \in \mathbf{R}^n$ is a minimum point for P_2 if and only if the system (in the unknown x):

$$S_3: \quad \varphi(\overline{x}; x) > 0, \quad g(x) \geq 0, \quad x \in \mathbf{R}^n,$$

is impossible, or

$$S_3': \quad \mathcal{H} \cap \mathcal{K}(\overline{x}) = \emptyset,$$

where $\mathcal{H} := \{(u, v) \in \mathbf{R} \times \mathbf{R}^m : u > 0, v \geq 0\}$ and $\mathcal{K}(\overline{x}) := \{(u, v) \in \mathbf{R} \times \mathbf{R}^m : \ u = \varphi(\overline{x}; x), \ v = g(x), \ x \in \mathbf{R}^n\} = F(\overline{x}; \mathbf{R}^n)$, with $F(\overline{x}; x) := (\varphi(\overline{x}; x), g(x))$.It is easy to see that S_3' holds if and only if

$$S_3'': \quad \mathcal{H} \cap (\mathcal{K}(\overline{x}) - \operatorname{clos} \mathcal{H}) = \emptyset,$$

where the difference is in vector sense and clos denotes closure. $\mathcal{K}(\overline{x})$, which is called the *image* of P_2, is such that $\mathcal{K}(\overline{x}) - \operatorname{clos} \mathcal{H})$is convex [9] as well as $\mathcal{H}$. In the *image* space *(u, v)*, consider the family of hyperplanes H defined by the equation

$$\ell(u, v; \theta, \lambda) := \theta u + \lambda v = 0$$

with

$$\theta \geq 0, \quad \lambda \geq 0, \quad (\theta, \lambda) \neq 0,$$

where the scalar θ and the row m-vector λ are parameters which describe the family. Denote by $H^{\underline{0}}$ the closed halfspace defined by the nonpositive level set of ℓ, and by H^+ the open halfspace defined by the positive level set of ℓ. We should like to be able to claim that 'S_3'' (and thus S'_3) holds if and only if there exists a hyperplane H such that $\mathcal{K}(\overline{x}) - \operatorname{clos} \mathcal{H} \subseteq H^{\underline{0}}$ (and thus $\mathcal{K}(\overline{x}) \subseteq H^{\underline{0}}$. While the necessity is an obvious consequence of the convexity of $\mathcal{H}$ and $\mathcal{K}(\overline{x}) - \operatorname{clos} \mathcal{H}$, the sufficiency unfortunately does not hold. In fact, if $\theta = 0$, then the above inclusion does not exclude that elements

of $\mathcal{K}(\overline{x}) - \operatorname{clos} \mathcal{H}$ (and thus of $\mathcal{K}(\overline{x})$, which belongs to H^0, may belong to $\mathcal{H}$(more precisely to $\mathcal{H} \cap \operatorname{frt} \mathcal{H}$; frt denotes frontier) or that $\mathcal{H} \cap F(\overline{x}; X(\lambda)) \neq \emptyset$, where

$$X(\lambda) := \left\{ x \in \mathbf{R}^n : \begin{array}{l} \varphi(\overline{x}; x) > 0, \\ g(x) \geq 0, \\ \lambda g(x) = 0 \end{array} \right\}.$$

This can be expressed saying that the above inclusion assures separation, but not disjunctive separation, between $\mathcal{H}$ and $\mathcal{K}(\overline{x}) - \operatorname{clos} \mathcal{H}$ (and thus $\mathcal{K}(\overline{x})$). Instead, if $\theta = 1$ (if $\theta > 0$, due to the homogeneity of l, θ can be reduced to be 1), then the above inclusion implies disjunctive separation so that the sufficiency holds. Therefore, in general the sufficiency does not hold. This drawback can be overcome in two ways. In the former (a priori) we restrict the class of functions φ, g in order to guarantee the existence of a separating hyperplane having $\theta > 0$; this is done by making suitable assumptions, which are called *constraint qualifications* (see [17]) if they implicate only g (e. g., Slater's condition which requires the existence of a $\widehat{x} \in \mathbf{R}^n$ such that $g(\widehat{x}) > 0$), and are called *regularity conditions* if they implicate both φ and g (see [9,17,18]). In the latter (a posteriori) we must check that $\mathcal{H} \cap F(\overline{x}; X(\lambda)) = \emptyset$.

In the preceding claim we have considered the inclusion in H^0, since it is a closed halfspace. When φ and g are affine and $\theta = 0$, if we replace H^0 with the negative level set of H, say H^-, then the sufficient part of the above claim becomes selfevident since $H^- \cap \mathcal{H} = \emptyset$. The necessity holds since S_3' and the 'parallelism' between $\mathcal{K}(\overline{x})$ and u-axis (the case when $\theta = 0$) imply $\mathcal{K}(\overline{x}) \subset H^-$.

We have obtained the following theorem; in the sequel we will understand the dependence of clos and $\mathcal{K}$ (and hence of F) on $\overline{x}$: to stress the fact that Theorem 3 holds independently of $\overline{x}$; namely, it holds whatever the concave function $\varphi(\overline{x}; \cdot)$ may be and not only when $\varphi(\overline{x}, x) = f(\overline{x}) - f(x)$. In fact, in going from S_3 to S_3^* or $S_3^{*\prime}$ or $S_3^{*\prime\prime}$, $\overline{x}$ does not play any substantial role: a change of $\overline{x}$ produces merely a translation of $\mathcal{K}$ in the direction of the u-axis and does not affect the conclusion.

Theorem 3 *Let* $\varphi\colon \mathbf{R}^n \to \mathbf{R}$ *and* $g\colon \mathbf{R}^n \to \mathbf{R}^m$.

i) Assume that φ and g be affine. S_3 is impossible if and only if there exist $\theta \in \boldsymbol{R}$ and $\lambda \in \boldsymbol{R}^m$, such that

$$S_3^*: \quad \begin{cases} \theta\varphi(x) + \lambda g(x) \leq 0, & \forall x \in \mathbf{R}^n, \\ \theta \geq 0, \quad \lambda \geq 0, & (\theta, \lambda) \neq 0, \end{cases}$$

where the first inequality must be verified in strict sense if $\theta = 0$.

ii) Assume that φ and g are concave, and that there exists $\widehat{x} \in \mathbf{R}^n$ such that $g(\widehat{x}) > 0$. S_3 is impossible if there exists $\lambda \in \boldsymbol{R}^m$, such that

$$S_3^{*\prime}: \quad \varphi(x) + \lambda g(x) \leq 0, \quad \forall x \in \mathbf{R}^n, \ \lambda \geq 0.$$

iii) S_3 is impossible if and only if there exist $\theta \in \boldsymbol{R}$ and $\lambda \in \boldsymbol{R}^m$, such that

$$S_3^{*\prime\prime}: \quad \begin{cases} \theta\varphi(x) + \lambda g(x) \leq 0 & \forall x \in \mathbf{R}^n, \\ \textit{with} & \theta \geq 0, \\ & \lambda \geq 0, \\ & (\theta, \lambda) \neq 0, \\ \textit{and} & X(\lambda) = \emptyset \\ \textit{when} & \theta = 0. \end{cases}$$

Before touching on the consequences for P_2 of the above approach, let us show how Theorem 3 can be used as a source for deriving TA; this will be done by deducing some classic linear TA from Theorem 3 even if, historically, these have been established directly.

With the notation of Theorem 1, set $\varphi(x) = -ax$ and $g(x) = Ax$, so that S_1 becomes a particular case of S_3. Theorem 3i) can be applied. At $\theta = 0$ S_3^* becomes $\lambda \geq 0$, $\lambda Ax < 0$, $\forall x \in \mathbf{R}^n$, and is obviously impossible; at $\theta = 1$ S_3^* becomes $\lambda \geq 0$, $-ax + \lambda Ax \leq 0$, $\forall x \in \mathbf{R}^n$, and holds if and only if $\lambda \geq 0$, $-a + \lambda A = 0$, which is equivalent to S_1^*. Theorem 1 follows from Theorem 3.

With the notation of Theorem 2 and its proof, S_2 turns out to be equivalent to system $t > 0$, $Ax \leq 0$, $Bx + et \leq 0$, which is easily identified as a particular case of S_3 where x is replaced by (x, t), $\varphi(x)$ by t, $g(x)$ by

$$\begin{pmatrix} -A & 0 \\ -B & -e \end{pmatrix} \begin{pmatrix} x \\ t \end{pmatrix}.$$

Theorem 3i) can be applied. Thus, by setting $\lambda = (u, v)$ at $\theta = 0$, S_3^* is obviously impossible; at $\theta = 1$ it becomes:

$$u \geq 0, \quad v \geq 0, \quad t - (uA + vB)x - vet \leq 0, \quad \forall x \in \mathbf{R}^n, \quad \forall t \in \mathbf{R},$$

and holds if and only if $u \geq 0$, $v \geq 0$, $ve = 1$, $uA + vB = 0$, which is equivalent to S_2^*. Theorem 2 follows from Theorem 3.

Theorem 4 (*Stiemke's theorem*) *Let A be an $m \times n$ matrix. Between the systems (in the unknowns x and u):*

$$S_4: \quad Ax \geq 0, \quad Ax \neq 0,$$

and

$$S_4^*: \quad uA = 0, \quad u > 0,$$

one and only one has solutions.

By observing that S_4^* is equivalent to $\begin{pmatrix} A^\top \\ -A^\top \end{pmatrix} u^\top \leq 0$, $-u^\top < 0$, S_4^* can be seen as a special case of S_2. Then the application of Theorem 2 leads quickly to the thesis.

Theorem 5 (*Motzkin's theorem*) *Let A, B, C be matrices of the orders $m \times n$, $p \times n$, $q \times n$, respectively. Between the systems (in the unknowns x and (u, v, y)):*

$$S_5: \quad Ax > 0, \quad Bx \geq 0, \quad Cx = 0$$

and

$$S_5^*: \quad \begin{cases} uA + vB + yC = 0, \\ u \geq 0, \quad u \neq 0, \\ v \geq 0 \end{cases}$$

one and only one has solutions.

It is immediate to see that S_5 is impossible if and only if the same happens to the system (in the unknown $\begin{pmatrix} x \\ t \end{pmatrix} \in \mathbf{R}^n \times \mathbf{R}$; e is a column m-vector whose entries equal 1): $t > 0$, $Ax \geq et$, $Bx \geq 0$, $Cx = 0$, which is easily identified as a particular case of S_3 where x is replaced by $\begin{pmatrix} x \\ t \end{pmatrix}$, $\varphi(x)$ by t, $g(x)$ by

$$\begin{pmatrix} A & -e \\ B & 0 \\ C & 0 \\ -C & 0 \end{pmatrix} \begin{pmatrix} x \\ t \end{pmatrix}.$$

Theorem 3i) can be applied. Thus, by setting $\lambda = (u, v, y', y'')$, at $\theta = 0$ S_3^* is obviously impossible; at $\theta = 1$ it becomes:

$$\begin{aligned} &u \geq 0, \quad v \geq 0, \quad y' \geq 0, \quad y'' \geq 0, \\ &t + [uA + vB + (y' - y'')C]x - uet \leq 0, \\ &\forall x \in \mathbf{R}^n, \quad \forall t \in \mathbf{R}, \end{aligned}$$

and holds if and only if $u \geq 0$, $v \geq 0$, $uA + vB + yC = 0$, $ue = 1$, which is equivalent to S_5^*. Theorem 5 follows from Theorem 3.

Theorem 6 (*Slater's theorem*) *Let A, B, C, D be matrices of the orders $m \times n$, $p \times n$, $q \times n$, $r \times n$, respectively. Between the systems (in the unknowns x and (u, v, y, z)):*

$$S_6: \quad \begin{cases} Ax > 0, \quad Bx \geq 0, \quad Bx \neq 0, \\ Cx \geq 0, \quad Dx = 0, \end{cases}$$

$$S_6^*: \quad \begin{cases} uA + vB + yC + zD = 0 \\ \text{and either} \\ \quad u \geq 0, \quad u \neq 0, \quad v \geq 0, \quad y \geq 0 \\ \text{or } u \geq 0, \quad v > 0, \quad y \geq 0, \end{cases}$$

one and only one has solutions.

It is easy to see that S_6 is equivalent to the system (in the unknown $\begin{pmatrix} x \\ t \end{pmatrix}$):

$$S_6': \quad \begin{cases} t > 0, \quad Ax \geq e_m t, \\ Bx \geq 0, \quad e_p Bx \geq t, \\ Cx \geq 0, \quad Dx = 0 \quad (t \in \mathbf{R}), \end{cases}$$

where e_m and e_p are respectively a column m-vector and a row p-vector both with entries equal to 1. S_6' is quickly identified as a particular case of S_3 where x is replaced by $\begin{pmatrix} x \\ t \end{pmatrix}$, $\varphi(x)$ by t, $g(x)$ by

$$\begin{pmatrix} A & -e_m \\ B & 0 \\ e_p B & -1 \\ C & 0 \\ D & 0 \\ -D & 0 \end{pmatrix} \begin{pmatrix} x \\ t \end{pmatrix}.$$

Theorem 3i) can be applied. Thus, by setting $\lambda = (u, v', v_0, y, z', z'')$, at $\theta = 0$ S_3^* is obviously impossible; at $\theta = 1$ S_3^* becomes:

$$\begin{aligned} &u \geq 0, \quad v' \geq 0, \quad v_0 \geq 0, \\ &y \geq 0, \quad z' \geq 0, \quad z'' \geq 0, \\ &t + [uA + (v' + v_0 e_p)B + yC + (z' - z'')D]x \\ &\quad - (ue_m + v_0)t \leq 0, \\ &\forall x \in \mathbf{R}^n, \quad \forall t \in \mathbf{R} \end{aligned}$$

and holds if and only if ($v := v' + v_0\, e_p$, $z := z' - z''$)

$$u \geq 0, \quad v \geq 0, \quad v_0 \geq 0, \quad y \geq 0,$$
$$uA + vB + yC + zD = 0, \quad ue_m + v_0 = 1.$$

This system is equivalent to S_6^* since $v_0 = 0 \Rightarrow u \neq 0$ and $v_0 > 0 \Rightarrow v > 0$. Theorem 6 follows from Theorem 3.

Theorem 7 (*Tucker's theorem*) *Let A, B, C be matrices of the orders $m \times n$, $p \times n$, $q \times n$, respectively. Between the systems (in the unknowns x and (u, v, y)):*

$$S_7: \quad Ax \geq 0, \quad Ax \neq 0, \quad Bx \geq 0, \quad Cx = 0$$

and

$$S_7^*: \quad uA + vB + yC = 0, \quad u > 0, \quad v \geq 0$$

one and only one has solutions.

It is immediate to see that S_7 is possible if and only if the same happens to the system (e is a row m-vector whose entries equal 1): $eAx > 0$, $Ax \geq 0$, $Bx \geq 0$, $Cx = 0$, which is easily identified as a particular case of S_3 where $\varphi(x) = eAx$ and

$$g(x) = \begin{pmatrix} A \\ B \\ C \\ -C \end{pmatrix} x.$$

Theorem 3i) can be applied. Thus, by setting $\lambda = (u', v, y', y'')$, at $\theta = 0$ S_3^* is obviously impossible; at $\theta = 1$ it becomes:

$$eAx + [u'A + vB + (y' - y'')C]x \leq 0,$$
$$u' \geq 0, \quad v \geq 0, \quad y' \geq 0, \quad y'' \geq 0,$$
$$\forall x \in \mathbf{R}^n,$$

and holds if and only if ($y := y' - y''$)

$$u' \geq 0, \quad v \geq 0, \quad (e + u')A + vB + yC = 0,$$

which is equivalent to S_7^* since $u := e + u' > 0$. Theorem 7 follows from Theorem 3.

At $B = 0$ and $C = 0$ Theorem 7 collapses to Theorem 4.

Theorem 8 (*Duffin's theorem*) *Let A be an $m \times n$ matrix, b a column m-vector, a a row n-vector, and α a scalar. The system (in the unknown x):*

$$S_8: \quad ax > \alpha, \quad Ax \leq b$$

is impossible if and only if at least one of the systems (in the unknown λ):

$$S_8^*: \quad \lambda A = a, \quad \lambda b \leq \alpha, \quad \lambda \geq 0$$

and

$$S_8^{*\prime}: \quad \lambda A = 0, \quad \lambda b < 0, \quad \lambda \geq 0,$$

is possible.

S_8 is easily identified as a particular case of S_3 where $\varphi(x) = ax - \alpha$, and $g(x) = b - Ax$. Theorem 3i) can be applied. Thus, at $\theta = 0$ S_3^* becomes $\lambda \geq 0$, $\lambda\, b - \lambda\, Ax < 0$, $\forall x \in \boldsymbol{R}^n$, and is equivalent to $S_8^{*\prime}$ at $\theta = 1$ S_3^* becomes $\lambda \geq 0$, $ax - \alpha + \lambda(b - Ax) \leq 0$, $\forall x \in \boldsymbol{R}^n$, and is equivalent to S_8^*. Theorem 8 follows from Theorem 3.

In quite similar ways other TA can be obtained from Theorem 3.

Theorem 3 can be stated directly, i. e. without introducing the image space (u, v) and separation, as classically done in [6]. It is satisfactory if a TA aims to play the role of lemma for some theorems. Instead, the above outlined way raises TA to the basis for developing most of the topics of the Theory of Optimization, and for obtaining TA under weaker assumptions than those of Theorem 3. A few comments will now be added on these two aspects.

When the assumptions of Theorem 3ii) are not fulfilled, then of course S_3 and $S_3^{*\prime}$ are not necessarily in alternative. However, by taking into account the geometric meaning of $S_3^{*\prime}$, it is easy to note that the feasibility of $S_3^{*\prime}$ is a sufficient condition (without any assumption on φ and g) for S_3 to be impossible and then (when $\varphi(x) = f(\overline{x}) - f(x)$) for $\overline{x}$ to be a minimum point of P_2. These facts lead us to a generalization of Theorem 3. Indeed, it is immediate to see that any condition, which assures the convexity of $\mathcal{K}-$clos $\mathcal{H}$ (like the concavity of φ and g), assures the existence of a separating hyperplane between $\mathcal{H}$ and $\mathcal{K}$. It has been proved [23] that $\mathcal{K}-$ clos $\mathcal{H}$ is convex if and only if the (1+ m)-vector function $F(x) := -\varphi(x) - g(x)$ is convex like (F is a convex-like function if and only if $\forall x', x''; \in \boldsymbol{R}^n$ and $\forall \alpha \in [0, 1]$ there exists $z \in \boldsymbol{R}^n$ such that $F(z) \leq (1 - \alpha)\, F(x') + \alpha\, F(x'')$). Therefore, we have:

Theorem 9 *Let the (1+ m)-vector function F be convex-like, and suppose that there exists $\widehat{x} \in \mathbf{R}^n$ such that $g(\widehat{x}) > 0$. Then, between S_3 and $S_3^{*\prime}$ one and only one has solutions.*

If φ and g are concave (in particular affine), then F is convex-like; thus Theorem 3ii) is a special case of Theorem 9.

Now, let us briefly show how $S_3^{*\prime}$ originates most of the topics of the theory of optimization. Since the feasibility of $S_3^{*\prime}$ implies that $\overline{x}$ is a minimum point for P_2, and since $S_3^{*\prime}$ is equivalent to (now continue to we use the position $\varphi(x) = f(\overline{x}) - f(x)$)

$$(\Delta): \quad \inf_{\lambda \geq 0} \sup_{x \in \mathbf{R}^n} [f(\overline{x}) - f(x) + \lambda g(x)] \leq 0,$$

we are led to consider the problem

$$P_2^*: \quad \sup_{\lambda \geq 0} \inf_{x \in \mathbf{R}^n} L(x; \overline{\lambda}),$$

where $L(x;\lambda) := f(x) - \lambda\, g(x)$ is called *Lagrangian function*, and the elements of a vector λ, which fulfills the above inequality and thus $S_3^{*\prime}$, are called *Lagrangian multipliers* and therefore receive the interesting interpretation as elements of the gradient of the hyperplane which separates the image $\mathcal{K}$ of P_2 from $\mathcal{H}$ to within an obvious transformation, the Lagrangian function is strictly related to such a hyperplane. P_2^* is known as *dual problem* of P_2 which, in its turn, receives the name of *primal problem*. The space of the (linear) separation functionals ℓ is called *dual space*; being here in a finite-dimensional space, it is isomorphic to the space $\mathbf{R}^m$ where λ runs; this is the reason why the elements of vector λ are often identified as *dual variables*. Without any assumption it is possible to prove the following inequality [9]:

$$\begin{aligned} \text{PD}: \quad \sup_{\lambda \geq 0} \inf_{x \in \mathbf{R}^n} L(x; \lambda) &\leq \inf_{x \in \mathbf{R}^n} \sup_{\lambda \geq 0} L(x; \lambda) \\ &= \inf_{x \in R} f(x), \end{aligned}$$

where the equality holds if $R := \{ x \in \mathbf{R}^n : g(x) \geq 0\} \neq \emptyset$. The difference between the second and the first term of the above inequality is called *duality gap* and is always nonnegative. Under the assumptions of Theorem 9 it is possible to prove [9,23] that the duality gap between P_2 and P_2^* is zero; this result recovers the well-known duality theorem for linear programming (see [4,17]) when f and g are affine. P_2^* has shown itself to be very useful to improve algorithms for solving P_2; a classic instance is offered by the so-called Hitchcock linear transportation problem (see [4]), where the use of dual variables (i. e., λ) drastically reduces the computational steps of simplex algorithm; another classic instance is offered by the dual decomposition methods for solving the so-called mixed integer linear programs (see [4]), which heavily exploit the dual problem under the assumption that the duality gap be zero. Theory and solving algorithms are not the only fields of application of duality. Indeed, a solution of the dual problem contains always an important piece of information, often even more important than that of the *primal problem*. This fact is proved by classic instances. When P_2 is the format for finding the maximum flow in a network, then the dual variables give the potentials at nodes and arcs (see [4]) which are crucial information for the design and management of the network. When P_2 is the format for finding the optimal production in an industry, then the dual variables represent the so-called shadow prices which lead to deep information on how the resources are exploited in the production process (see [4]). Many other applications might be mentioned. In all cases the introduction of the dual problem leads to a deep mathematical analysis which would have been inconceivable if the primal problem only had been introduced.

Now, let us go back to (Δ). From this condition it is possible to derive the classic necessary and sufficient condition for $\overline{x}$ to be a minimum point of P_2 which is expressed in terms of generalized multipliers [17,20]. Condition (Δ)has stimulated the development of several other theories, like minimax theory and game theory [5], saddle point theory [9], penalization theory [9].

When $\mathcal{K} - \text{clos}\,\mathcal{H}$ is not convex, $S_3^{*\prime}$ may not hold even if $S_3{}'$ does. In such a case, the above separation scheme suggests the introduction of a nonlinear functional to replace $\ell(u, v; \lambda)$; in other words, we can try to show $S_3{}'$ by means of nonlinear separation if the linear one fails (see ► Image space approach to optimization). The nonlinear separation has led to generalize all the above results [9], and has allowed us to extend TA to more general situations, like systems of point-to-set maps [8,10], or to systems in a complex space, or to systems in an infinite-dimensional space where the first contribution is due to J. Farkas [7].

See also

► Farkas Lemma
► Linear Optimization: Theorems of the Alternative

References

1. Abadie J (1967) On the Kuhn–Tucker theorem. In: Abadie J (ed) Nonlinear Programming. North Holland, Amsterdam, pp 19–36
2. Banach S (1925) Sur les lignes rectifiables et les surfaces dont l'aire est finie. Fundam Math 7:225–237
3. Craven BD, Gwinner J, Jeyakumar V (1987) Nonconvex theorems of the alternative and minimization. Optim 18(2):151–163
4. Dantzig GB (1963) Linear programming and extensions. Princeton Univ. Press, Princeton
5. Du D-Z, Pardalos PM (1995) Minimax and applications. Kluwer, Dordrecht
6. Fan K, Glicksberg I, Hoffman AJ (1957) Systems of inequalities involving convex function. Proc 8:617–622
7. Farkas J (1902) Uber die Theorie der einfachen Ungleichungen. J Reine Angew Math 124:1–27
8. Ferrero O (1989) Theorems of the alternative for set-valued functions in infinite-dimensional spaces. Optim 2(2):167–175
9. Giannessi F (1984) Theorems of the alternative and optimality conditions. J Optim Th Appl 42(3):331–365
10. Giannessi F (1987) Theorems of the alternative for multifunctions with applications to optimization: General results. J Optim Th Appl 55:233–256
11. Gordan P (1873) Über die Auflösungen linearer Gleichungen mit reelen Coefficienten. Math Ann 6:23–28
12. Hahn H (1927) Über lineare Gleichungen in linearen Räumen. J Math 157:214–229
13. Helly E (1912) Über lineare Funktional Operationen. Sitzungsber Akad Wiss Wien 121:265–297
14. Heinecke G, Oettli W (1990) A nonlinear theorem of the alternative without regularity assumptions. J Math Anal Appl 146:580–590
15. Jeyakumar V (1985) Convexlike alternative theorems and mathematical programming. Optim 16(5):643–652
16. Lehmann R, Oettli W (1975) The theorem of the alternative: Key-theorem, and the vector maximum problem. Math Program, vol 8. North-Holland, Amsterdam, 332–344
17. Mangasarian OL (1994) Nonlinear programming. Classics Appl Math. SIAM, Philadelphia
18. Mastroeni G, Pappalardo M (1998) Separation and regularity in the image space. New trends in Mathematical Programming. Kluwer, Dordrecht
19. Motzkin TS (1936) Beiträge zur Theorie der Linearen Ungleichungen. Inaugural Diss Basel, Jerusalem
20. Rapcsák T (1997) Smooth nonlinear optimization in $\mathbf{R}^n$. Nonconvex Optim Appl, no. 19. Kluwer, Dordrecht
21. Slater ML (1951) A note on Motzkin's transposition theorem. Econometrica 19:185–186
22. Stiemke E (1915) Über positive Lösungen homogener linearer Gleichungen. Math Ann 76:340–342
23. Tardella F (1989) On the image of a constrained extremum problem and some applications to the existence of a minimum. J Optim Th Appl 60(1):93–104
24. Tricomi FG (1957) Integral equations. Interscience, New York

Time-Dependent Traveling Salesman Problem

MOHIT TAWARMALANI, NIKOLAOS V. SAHINIDIS
University Illinois, Urbana-Champaign, USA

MSC2000: 90C27

Article Outline

Keywords
Formulations of TDTSP
Envelopes and Tight Formulations
Network Interpretation of TDTSP
Decomposition Algorithm for TDTSP
Heuristics for TDTSP
Conclusion
See also
References

Keywords

TDTSP; Benders decomposition; Convexification; Multistage optimization

Given a list of cities, the classical traveling salesman problem is aimed at finding the least cost tour through the cities. The time-dependent traveling salesman problem is a generalization of the traveling salesman problem where the cost of travel between cities is also dependent on the order in which they are traversed. We now provide a more formal description of the two problems. Consider a set of cities $\mathcal{N} = \{1, \ldots, n\}$ and a mapping, D: $\mathcal{N} \times \mathcal{N} \to \mathbf{R}$, that associates with each ordered city-pair a cost incurred when a travel/transition is undertaken starting from the first city and ending at the second city. The data may be pictorially visualized on a complete directed graph of n nodes, where the nodes represent the cities and the arcs are labeled with the transition costs of the incident node pair. The cost function may be extended from single transitions to paths by summing the cost of travel over the arcs

comprising the path. As there is a unique arc joining any two nodes, a path in the graph may be identified with a sequence of nodes. Let us then restrict our attention to simple circuits that pass through all the nodes. These are the Hamiltonian cycles of the directed graph. Cyclic permutations of the node set produce these Hamiltonian cycles and there are, therefore, $(n-1)!$ possible candidates. The *classical traveling salesman problem* (TSP) is aimed at finding the cyclic permutation/Hamiltonian cycle of cities with the minimum travel cost.

Given a Hamiltonian cycle $\mathcal{P}$, associate with every arc its ordinality in $\mathcal{P}$. A variant of the TSP spawns out if the contribution of a transition on the arc towards the cost of $\mathcal{P}$ is not only dependent on the ordered node-pair incident to the arc but also on the ordinality of the arc in $\mathcal{P}$. The ordinality of the arc shall be referred to as the time-period of the associated transition following the intuition that the cost of travel between cities varies with time and assuming that it takes one time unit to travel between any two cities. In other words, the cost of transition is specified as a mapping $C: \mathcal{N} \times \mathcal{N} \times \mathcal{T} \rightarrow \mathbf{R}$ where $\mathcal{N}$ is the set of possible ordinal values for an arc in any Hamiltonian cycle $\mathcal{P}$. Note that $|\mathcal{T}| = |\mathcal{N}|$ since we restrict attention to Hamiltonian cycles. The *time-dependent traveling salesman problem* (TDTSP) is aimed at finding the minimum cost Hamiltonian cycle under the cost structure defined above. It should be apparent that TSP is a special case of TDTSP.

Computational experience indicates that the TDTSP is a significantly more difficult problem in comparison to the TSP. However, the flexibility obtained by using the more elaborate cost structure allows one to model additional interesting applications.

Applications of the TDTSP have been proposed in: sequence dependent scheduling with time-dependent set-up costs [10], scheduling with precedence constraints [3] and timetabling [2].

Studies on the TDTSP have been made by J.C. Picard and M. Queyranne [10]. Exact solution approaches for a special case of the TDTSP, namely the delivery man problem, have been reported by A. Lucena [7]. Time-dependent vehicle routing problems have been studied by C. Malandraki and M.S. Daskin [9]. Various formulations for the TDTSP have been compared by L. Gouveia and S. Voß [4]. Benders' partitioning scheme has been used to derive an exact algorithm for the TDTSP [15]. Heuristics to accelerate convergence of such an algorithm are discussed in [14].

In this article, we present existing formulations and solution methodologies for the TDTSP. First, we review various formulations of the TDTSP and a comparison of these formulations in regards to the tightness of their relaxations. We then present a technique of constructing tight relaxations by employing convex and concave envelopes of product terms. On the algorithmic side, we outline a modification of the Benders decomposition algorithm for the TDTSP. Then, using multistage network optimization we present an acceleration technique for the above algorithm. Finally, a variable depth search heuristic is briefly outlined.

Formulations of TDTSP

The TDTSP is a special case of the *quadratic assignment problem* (QAP). The formulation for the QAP may hence be employed to model the TDTSP as follows:

$$\begin{cases} \min & y^{\top} Q y \\ \text{s.t.} & y \in \mathrm{AP}_n, \end{cases}$$

where AP_n is the assignment polytope for n assignments,

$$Q = \begin{pmatrix} 0 & Q_{1\,2} & \cdots & Q_{1\,n-1} & Q_{1\,n} \\ 0 & 0 & \cdots & Q_{2\,n-1} & Q_{2\,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & Q_{n-1\,n} \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix},$$

where $Q_{ik} \in \mathbf{R}^{n \times n}$ and assumes the form

$$\begin{pmatrix} 0 & c_{ik\,1} & 0 & 0 & 0 & \cdots & c_{ik\,n} \\ c_{ki\,1} & 0 & c_{ik\,2} & 0 & 0 & \cdots & 0 \\ 0 & c_{ki\,2} & 0 & c_{ik\,3} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & c_{ki\,n-3} & 0 & c_{ki\,n-2} & 0 \\ 0 & \cdots & 0 & 0 & c_{ki\,n-2} & 0 & c_{ki\,n-1} \\ c_{ki\,n} & \cdots & 0 & 0 & 0 & c_{ki\,n-1} & 0 \end{pmatrix}$$

in terms of the mapping $C : \mathcal{N} \times \mathcal{N} \times \mathcal{T} \rightarrow \mathbf{R}$.

In order to derive linearized versions of the above formulation, we define two sets of binary variables:

$$Y_{it} = \begin{cases} 1 & \text{city } i \text{ visited in period } t, \\ 0 & \text{otherwise,} \end{cases}$$

$$X_{ijt} = \begin{cases} 1 & \text{transition from city } i \\ & \text{to city } j \text{ in period } t, \\ 0 & \text{otherwise.} \end{cases}$$

The above formulation then takes the following form:

$$(Q) \begin{cases} \min & \sum_i \sum_j \sum_t C_{ijt} Y_{it-1} Y_{it}, \\ \text{s.t.} & \sum_t Y_{it} = 1, \quad i \in \mathcal{N}, \\ & \sum_i Y_{it} = 1 \quad t \in \mathcal{T} \\ & Y_{it} \in \{0,1\}, \quad i \in \mathcal{N},\ t \in \mathcal{T}. \end{cases}$$

Linearized formulations are derived by introducing the transition variables X. Most linearized models assume that the tour begins and ends in city 1. This condition does not pose any additional restriction since a dummy city may be added with zero transition costs to and from other cities and then be treated as the starting city.

In [5], a linearized reformulation of the TDTSP was derived by using the X variables:

$$\min \sum_i \sum_j \sum_t C_{ijt} X_{ijt}$$

such that:

$$\sum_t Y_{it} = 1, \quad i \in \mathcal{N}, \tag{1}$$

$$\sum_i Y_{it} = 1, \quad t \in \mathcal{T}, \tag{2}$$

$$Y_{it} + Y_{j\,t-1} - 2X_{ijt} \geq 0, \quad i, j \in \mathcal{N},\ t \in \mathcal{T}, \tag{3}$$

$$\sum_i \sum_j \sum_t X_{ijt} = n, \quad j \in \mathcal{N}, \tag{4}$$

$$Y_{it} \in \{0,1\}, \quad i \in \mathcal{N},\ t \in \mathcal{T}, \tag{5}$$

$$X_{ijt} \in \{0,1\}, \quad i, j \in \mathcal{N},\ t \in \mathcal{T}. \tag{6}$$

The constraints (1) and (2) are the assignment constraints. Constraints (3) state that no transition from city i to city j can take place in time-period t if city i was not visited in time-period $t-1$ and city j was not visited in time-period t. Note that this constraint could be tightened by replacing it by the equivalent two constraints $X_{ijt} \leq Y_{it}$ and $X_{ijt} \leq Y_{jpt}$. Constraint (4) just states that there are n transitions in any feasible solution.

The linear formulation presented by Picard and Queyranne [10] is described below:

$$\min \sum_i \sum_j \sum_t C_{ijt} X_{ijt}$$

such that:

$$\sum_j X_{1j1} = 1, \tag{7}$$

$$\sum_i X_{ijt} = \sum_i X_{ij\,t+1}, \quad i \in \mathcal{N},\ t \in \mathcal{T}\backslash\{1\}, \tag{8}$$

$$\sum_i \sum_t X_{ijt} = 1, \quad j \in \mathcal{N}, \tag{9}$$

$$X_{ijt} \in \{0,1\}, \quad i, j \in \mathcal{N},\ t \in \mathcal{T}. \tag{10}$$

Constraint (7) fixes the starting city as 1. Constraints (8) require an entry to be followed by exit from any city in the following time-period. Constraints (9) allow only one entry to a city.

Another model for the TDTSP has been proposed by K. Fox, B. Gavish and S. Graves [3] based on the assumption that the tour begins and ends in city 1:

$$\min \sum_i \sum_j \sum_t C_{ijt} X_{ijt}$$

such that:

$$\sum_j \sum_t X_{ijt} = 1, \quad i \in \mathcal{N}, \tag{11}$$

$$\sum_t \sum_j X_{ijt} = 1, \quad j \in \mathcal{N}, \tag{12}$$

$$\sum_i \sum_j X_{ijt} = 1, \quad t \in \mathcal{T}, \tag{13}$$

$$\sum_j \sum_{t=2}^{n} t X_{ijt} - \sum_j \sum_{t=1}^{n-1} t X_{ijt} = 1, \quad i \in \mathcal{N}\backslash\{1\}, \tag{14}$$

$$X_{ijt} \in \{0,1\}, \quad i, j \in \mathcal{N},\ t \in \mathcal{T}. \tag{15}$$

Constraints (11), (12) and (13) are assignment relationships which, respectively, state that a city is left, entered

and visited exactly once. Constraints (14) are subtour elimination constraints that force leaving the city (except the starting one) in the time-period following the entering time-period.

We present below a linear formulation of the TDTSP proposed in N.V. Sahinidis and I.E. Grossmann [11] and R.J. Vander Wiel and Sahinidis [14]:

$$(P)\begin{cases}\min & \sum_i \sum_j \sum_t C_{ijt} X_{ijt},\\ \text{s.t.} & \sum_t Y_{it} = 1, \quad i \in \mathcal{N},\\ & \sum_i Y_{it} = 1, \quad t \in \mathcal{T},\\ & \sum_i X_{ijt} = Y_{jt}, \quad j \in \mathcal{N},\ t \in \mathcal{T},\\ & \sum_j X_{ijt} = Y_{i\,t-1}, \quad i \in \mathcal{N},\ t \in \mathcal{T},\\ & Y_{it} \in \{0,1\}, \quad i \in \mathcal{N},\ t \in \mathcal{T},\\ & X_{ijt} \geq 0.\end{cases}$$

This last formulation does not require the X variables to take integral values as the constraints enforce the integrality of X when the Y variables take integral values.

The strengths of the various formulations are compared in [4]. It turns out that formulation P has the tightest linear relaxation amongst all formulations. The formulation in [10] achieves the same objective function value. However, its feasible region is larger. The formulations in [3] and [5] are dominated by formulation (P).

Envelopes and Tight Formulations

We provide some insight into the tight relaxations for the TDTSP. By introducing the X variables in the TDTSP formulation (Q), we obtain the following mathematical program:

$$\begin{cases}\min & \sum_i \sum_j \sum_t C_{ijt} X_{ijt}\\ \text{s.t.} & \sum_t Y_{it} = 1, \quad i \in \mathcal{N},\\ & \sum_i Y_{it} = 1, \quad t \in \mathcal{T},\\ & X_{ijt} = Y_{i\,t-1} Y_{it},\\ & Y_{it} \in \{0,1\}, \quad i \in \mathcal{N},\ t \in \mathcal{T}.\end{cases} \tag{16}$$

Note that, in the above formulation, there are no integrality restrictions on the X variables. These are automatically enforced by the bilinear constraints (16) in the formulation. However, these constraints are nonconvex and therefore the mathematical program given above is a nonconvex nonlinear program. A convex relaxation may be developed by replacing the bilinear constraints by convex constraints that properly contain all the feasible points to the above program. In particular, if constraints (16) are replaced by linear constraints then the linear programming relaxations of TDTSP are obtained.

We present a general methodology for constructing tight linear relaxations of 0–1 programs containing product terms of 0–1 variables. Product terms of 0–1 variables have linear concave and convex envelopes over the unit hypercube as shown in [13]. Therefore, we may introduce new variables for the product terms and restrict them to lie in the convex set formed by the concave and convex envelope of the corresponding product term. Note that the product terms take integral values at the extreme points. However, as the convex and concave envelopes are exact at the extreme points of the unit hypercube, the integrality restrictions on the newly introduced variables become redundant and may be dropped. This technique may be used in conjunction with the *reformulation-linearization technique* (RLT) introduced by H.D. Sherali and W.P. Adams [12].

In the case of the TDTSP, the product terms are bilinear. The convex envelope is, therefore, given by:

$$X_{ijt} \geq \max\{Y_{i\,t-1} + Y_{it} - 1, 0\}.$$

The concave envelope is given by:

$$X_{ijt} \leq \min\{Y_{i\,t-1}, Y_{it}\}.$$

It was shown in [13] that the above constraints are implied in the formulation (P) described above. Furthermore, this formulation may be derived by using the scheme described in this section.

Network Interpretation of TDTSP

We now provide a network interpretation of the TDTSP. If the Y variables in the linear relaxation of (P) are fixed, the formulation (P) reduces to a network flow

model. Formally, the problem is:

$$(S)\begin{cases}\min & \sum_i \sum_j \sum_t C_{ijt} X_{ijt} \\ \text{s.t.} & \sum_i X_{ijt} = Y_{jt}, \quad j \in \mathcal{N},\ t \in \mathcal{T}, \\ & \sum_j X_{ijt} = Y_{i\,t-1}, \quad i \in \mathcal{N},\ t \in \mathcal{T}, \\ & X_{ijt} \geq 0. \end{cases}$$

It follows from the problem definition that the above problem decomposes by time-periods. In each time period, (S) takes the form of a transportation problem. Hence, a series of transportation problems may be used to solve the above problem. An alternate way of visualizing this structure is by juxtaposing the transportation problems to form a $2n$-partite graph. An illustration of the graph appears as Fig. 1. The destination node representing city i in time-period $t-1$ is connected to the source node representing city i in time period t by an arc with capacity interval $[Y_{i\,t-1}, Y_{i\,t-1}]$. In the new framework, the problem reduces to a feasible circulation problem. For more on network flow problems, see [1].

Fixing the Y variables amounts to identifying a Hamiltonian cycle of the $2n$-partite graph. Hence, the TDTSP reduces to the problem of identifying the minimum cost Hamiltonian cycle on this graph. Note that it is possible to combine the ith destination node in period $t-1$ and ith source node in period t to produce an n-partite graph.

Decomposition Algorithm for TDTSP

Based on the network interpretation of the problem described above, it is possible to arrive at a decomposition algorithm to solve the dual of the linear relaxation of (P).

This algorithm employs ideas of *Benders decomposition*. The master problem is defined in the space of the Y variables. Once the Y variables are fixed, the subproblem is a set of n transportation problems. They are solved and one of their dual optimal solutions is picked to construct a cut for the master problem. This procedure is iterated producing a series of master problems that are increasingly tighter approximations of the projection of (P) on the space of the Y variables. When the upper bound from the subproblem and lower bound from the master problem converge, the solution to (P) is obtained. For a detailed description of the algorithm, see [15] and [13].

Note that the cutting plane introduced into the Benders master problem depends on the optimal dual solution selected from the subproblem to construct it. T.L. Magnanti and R.T. Wong [8] proved that *Pareto optimal* solutions to a Benders problem may be constructed by solving a second-stage optimization problem on the set of optimal dual solutions from the subproblem. It was shown in [13] that, whenever the subproblem is a network-flow problem, the Pareto optimal problem can be recast as a network-flow problem. Using this idea, the linear programming relaxation may be solved by introducing Pareto optimal cuts at each iteration. Computational experience shows that this methodology gives faster convergence characteristics for the Benders algorithm.

Once we have a solution methodology for the dual of the linear relaxation, we can incorporate it in the *branch and bound* framework to derive an exact algorithm for the TDTSP. Note that the dual of the linear relaxation does not need to be solved to optimality to produce a valid lower bound to construct the enumeration tree.

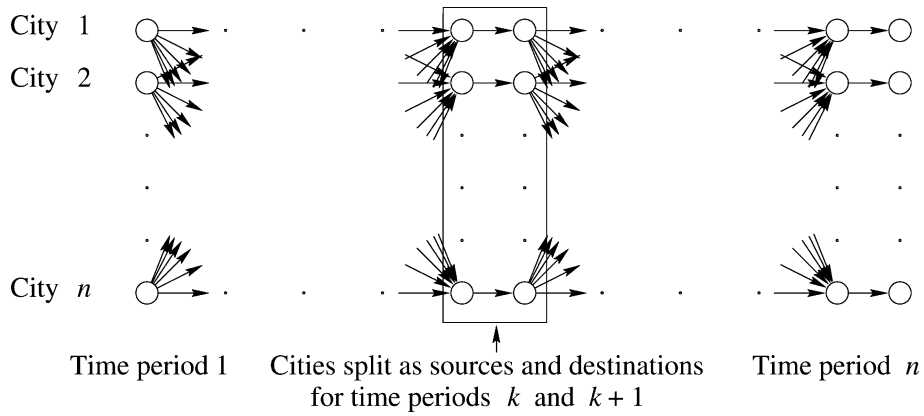
Heuristics for TDTSP

Heuristics for the TDTSP are natural extensions of heuristics for the TSP. Probably one of the most successful heuristics for the TSP is the *R-opt heuristic* developed by S. Lin and B.W. Kernighan [6] and extended for the TDTSP by Vander Wiel and Sahinidis [14].

We describe the variable R-opt heuristic as applied to the TDTSP. This is an improvement heuristic and assumes an initial tour has been constructed already.

- Step 1 selects a transition (l, k) for removal.
- Step 2 identifies an arc (k, i) to replace arc (l, k).
- The selection is done in a way that maximizes a certain estimate of cost improvement which takes into account the time-dependence of the transition costs.
- The path between j and k is then reversed and arc (l, j) is added to complete the tour.
- Step 5 iterates trying to accomplish a similar reduction with (j, i) as the starting arc instead of (l, k).

For details, see [14].



Time-Dependent Traveling Salesman Problem, Figure 1
The 2*n*-partite TDTSP graph

Conclusion

TDTSP is a computationally difficult problem. Since the linear programming relaxations of the formulations for this problem are large, it is important to identify structured constraints and solve the problems efficiently using some decomposition based ideas. We presented one such algorithm that exploits the network substructure of the TDTSP formulation. Furthermore, valid inequalities for the TDTSP polytope may help improve its formulation and allow us to develop more efficient solution techniques. However, as of today, the TDTSP continues to be an intractable problem for large instances.

See also

- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Graph Coloring
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows. Prentice-Hall, Englewood Cliffs, NJ
2. Balakrishnan N, Lucena A, Wong RT (1992) Scheduling examinations to reduce second-order conflicts. Comput Oper Res 19:353–361
3. Fox K, Gavish B, Graves S (1980) An N-constraint formulation of the (time-dependent) traveling salesman problem. Oper Res 28(4):1018–1021
4. Gouveia L, Voß S (1995) A classification of formulations for the (time-dependent) traveling salesman problem. Europ J Oper Res 83:69–82
5. Lawler EL (1963) The quadratic assignment problem. Managem Sci 19:586–599
6. Lin S, Kernighan BW (1973) An effective heuristic algorithm for traveling salesman problem. Oper Res 21:498–516
7. Lucena A (1990) Time-dependent traveling salesman problem - The deliveryman case. Networks 20:753–763
8. Magnanti TL, Wong RT (May-June 1981) Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. Oper Res 29(3) 464–484

9. Malandraski C, Daskin MS (1992) Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. Transport Sci 26(3):185–200
10. Picard JC, Queyranne M (1978) The time-dependent traveling salesman problem and its applications to the tardiness problem in one-machine scheduling. Oper Res 26:86–110
11. Sahinidis NV, Grossmann IE (1991) MINLP model for cyclic multiproduct scheduling on continuous parallel lines. Comput Chem Eng 15(2):85–103
12. Sherali HD, Adams WP (Aug. 1990) A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J Discret Math 3(3):411–430
13. Tawarmalani M (1997) Multistage network optimization and decomposition algorithms. Techn Report Univ Illinois, Urbana-Champaign
14. Vander Wiel RJ, Sahinidis NV (1995) Heuristic bounds and test problem generation for the time-dependent traveling salesman problem. Transport Sci 29(2):167–183
15. Vander Wiel RJ, Sahinidis NV (1996) An exact solution approach for the time-dependent traveling salesman problem. Naval Res Logist 43(6):797–820

Topological Derivative in Shape Optimization

JAN SOKOLOWSKI[1], ANTONI ZOCHOWSKI[2]
[1] Laboratoire de Mathématiques, Université Henri Poincaré, Nancy, France
[2] Systems Research Institute of the Polish Academy of Sciences, Warsaw, Poland

MSC2000: 49Q10, 49Q12, 74P05, 35J85

Article Outline

Introduction
Definitions
Formulation
  Three-Dimensional Anisotropic Elastic Body with a Small Cavity
  Contact Problem for Plane Elasticity
  Solution of the Elasticity System in the Ring
Cases
  Plane Isotropic Elasticity System
  Three-Dimensional Isotropic Elasticity Systems
Conclusions
References

Introduction

Topological derivatives of shape functionals are introduced in [35] for elliptic boundary value problems. The construction is based on the technique [15,24] of singular perturbations of geometrical domains, and the mathematical framework for topological differentiability in the general case can be found in [27] within the method of compound asymptotic expansions. There are numerous applications of topological derivatives to the resolution of shape optimization and inverse problems. The asymptotic analysis of boundary value problems in singularly perturbed geometrical domains is performed in the monographs [15,24,26], and, e. g., in the papers [17,28,29,30]. The derivation of topological derivatives for integral functionals is presented, e. g., in the papers [11,19,20,21,22,23,27,34,35,36,37,38,40,41], and in the Ph.D. dissertations [18,31].

In this chapter we perform an asymptotic analysis for boundary value problems in elasticity in two and three spatial dimensions. The results are borrowed from papers of the authors, [35,38,40,41]; see also [27], where the complete proofs of the presented results can be found.

Numerical methods of optimization with topological derivatives are considered in, e. g., the papers [1,2,3,4,6,7,9,10,13,14,16,18,34].

Definitions

The topological derivative for a shape functional is defined in the following way.

Assume that $\Omega \subset \mathbb{R}^N$, $N = 2, 3$, is an open set and that there is given a shape functional

$$\mathcal{J}\colon \Omega \setminus K \to \mathbb{R}$$

for any compact subset $K \subset \overline{\Omega}$. We denote by $B_\rho(x)$, $x \in \Omega$, the open ball of radius $\rho > 0$ around x, and $\omega_\rho(x) = \overline{B_\rho(x)}$. The domain with a void will be denoted $\Omega(\rho, x) = \Omega \setminus \omega_\rho(x)$. Assume that there exists the following limit:

$$\mathcal{T}(x) = \lim_{\rho \downarrow 0} \frac{\mathcal{J}(\Omega(\rho, x)) - \mathcal{J}(\Omega)}{|\omega_\rho(x)|} ,$$

which can be defined in an equivalent way by

$$\tilde{\mathcal{T}}(x) = \lim_{\rho \downarrow 0} \frac{\mathcal{J}(\Omega(\rho, x)) - \mathcal{J}(\Omega)}{\rho^N} .$$

The function $\mathcal{T}(x), x \in \Omega$, is called the topological derivative of $\mathcal{J}(\Omega)$ and provides the information on the infinitesimal variation of the shape functional $\mathcal{J}$ if a small hole is created at $x \in \Omega$. We shall show in the sequel that the method is constructive, i. e., the topological derivative can be evaluated for shape functionals depending on solutions of elasticity equations defined in Ω.

The partial differential equation for $u_\rho = u_{\Omega(\rho,x)}$ is called the state equation for the shape optimization problems under consideration. We show that for a class of shape functionals it is sufficient to solve in the unperturbed domain Ω the state equation as well as the appropriate adjoint state equation in order to evaluate the topological derivative $\mathcal{T}(x), x \in \Omega$. This means that the derivative can be used in shape optimization for broad classes of shape functionals and partial differential equations. Some examples of where the derivative is explicitly given for model problems are provided.

Our results can be described in the form of the following expansion:

$$\mathcal{J}(\Omega(\rho, x)) = \mathcal{J}(\Omega) + |\omega_\rho(x)|\mathcal{T}(x) + o(\rho^N) .$$

In the very special case of the energy functional, the so-called compliance functional in linear elasticity, the topological derivative is in fact considered in [8]. The derivative is used, for the first time, in numerical methods of optimal design for the specific choice of shape functional [8]. In order to differentiate the energy functional with respect to the variations of the boundary of the domain of integration, knowledge of the shape derivative of the state equation with respect to the boundary variations is not required. Therefore, the results obtained for the particular case of the energy functional cannot be directly generalized to the case of an arbitrary shape functional.

In the sequel we shall drop x from the notation, assuming that the cavity surrounds $x = \mathcal{O} \in \Omega$.

Formulation

Three-Dimensional Anisotropic Elastic Body with a Small Cavity

Let us consider the elasticity problem written in the matrix/column form

$$\mathcal{L}u = D(-\nabla_x)^\top A{\sim}D(\nabla_x)u = 0 \quad \text{in } \Omega(\rho) , \qquad (1)$$

$$\mathcal{N}^\Omega u = D(n)^\top A{\sim}D(\nabla_x)u = g^\Omega \quad \text{on } \partial\Omega , \qquad (2)$$

$$\mathcal{N}^\omega u = D(n)^\top A{\sim}D(\nabla_x)u = 0 \quad \text{on } \partial\omega_\rho , \qquad (3)$$

where A is a symmetric positive definite matrix of size 6×6, consisting of the elastic material moduli (the Hooke's matrix) $\alpha = 1/\sqrt{2}$ and $D(\nabla_x)$ is a 6×3 matrix of the first-order differential operators ($\xi_i = \partial/\partial x_i$):

$$D(\xi)^\top = \begin{bmatrix} \xi_1 & 0 & 0 & 0 & \alpha\xi_3 & \alpha\xi_2 \\ 0 & \xi_2 & 0 & \alpha\xi_3 & 0 & \alpha\xi_1 \\ 0 & 0 & \xi_3 & \alpha\xi_2 & \alpha\xi_1 & 0 \end{bmatrix} \qquad (4)$$

u is the displacement column, and $n = (n_1, n_2, n_3)^\top$ is the unit outward normal vector on $\partial\Omega(\rho)$, i. e., unit column. In this notation the strain and stress columns are given respectively by $\epsilon(u) = D(\nabla_x)u$ and $\sigma(u) = A{\sim}D(\nabla_x)u$, which gives

$$\epsilon(u) = \left(\epsilon_{11}, \epsilon_{22}, \epsilon_{33}, \sqrt{2}\epsilon_{23}, \sqrt{2}\epsilon_{31}, \sqrt{2}\epsilon_{12}\right)^\top ,$$
$$\sigma(u) = \left(\sigma_{11}, \sigma_{22}, \sigma_{33}, \sqrt{2}\sigma_{23}, \sqrt{2}\sigma_{31}, \sqrt{2}\sigma_{12}\right)^\top .$$

The load g^Ω is supposed to be self-equilibrated in order to assure the existence of a solution to the elasticity problem,

$$\int_{\partial\Omega} \mathrm{d}(x)^\top g^\Omega(x)\, \mathrm{d}s_x = 0 \in \mathbb{R}^6, \qquad (5)$$

where

$$d(x) = \begin{bmatrix} 1 & 0 & 0 & 0 & -\alpha x_3 & \alpha x_2 \\ 0 & 1 & 0 & \alpha x_3 & 0 & -\alpha x_1 \\ 0 & 0 & 1 & -\alpha x_2 & \alpha x_1 & 0 \end{bmatrix} \qquad (6)$$

represents rigid body motion.

The general theory presented in the article can be applied to a broad class of shape functionals; however, to fix the ideas we deal only with one representative example.

Let us consider the functional

$$\mathbb{J}^1_\rho(u) = \int_{\Omega(\rho)} \sigma(u;x)^\top B(x)\sigma(u;x)\, \mathrm{d}x . \qquad (7)$$

Functional (7) looks like the elastic energy functional

$$\mathbb{E}(u;\Omega(\rho)) = \frac{1}{2}\int_{\Omega(\rho)} \epsilon(u;x)^\top A\epsilon(u;x)\, \mathrm{d}x$$
$$= \frac{1}{2}\int_{\Omega(\rho)} \sigma(u;x)^\top A^{-1}\sigma(u;x)\, \mathrm{d}x \qquad (8)$$

but can contain a certain symmetric 6×6-matrix function B. In the case of a constant, diagonal matrix B functional (7) is related to the square of the $L_2(\Omega)$-norm of the stress tensor or of its components. On the other hand, if $A(x)^{-1}B(x)A(x)^{-1}$ becomes a constant diagonal matrix with our choice of B, then in (7) the similar strain norms are obtained. For problem (1)–(3) the explicit dependence of the integrand on the displacement vector $u(\rho,x)$ makes no sense, since such a displacement field is defined up to rigid motions.

From condition (5) it follows that both problems, problem (1)–(3) in the body $\Omega(\rho)$ with the cavity ω_ρ and the first limit problem in the entire body Ω,

$$\begin{aligned} D(-\nabla_x)^\top AD(\nabla_x)v = 0 \text{ in } \Omega\,, \\ D(n)^\top AD(\nabla_x)v = g^\Omega \text{ in } \partial\Omega\,, \end{aligned} \tag{9}$$

admit the solutions $u(\rho,x) \in C^{2,\alpha}(\Omega(\rho))^3$ and $v \in C^{2,\alpha}(\Omega)^3$, respectively, under the loading $g^\Omega \in C^{1,\alpha}(\partial\Omega)^3$. Freedom in selection of such solutions up to the rigid motions has no influence on functional (7) and therefore can be neglected (we recall only that using additional conditions we can pass to uniquely solvable problems).

Before presenting the result for functional (7), we recall some facts. First of all, the adjoint state $W \in C^{2,\alpha}(\Omega)^3$ has the form

$$\begin{aligned} &D(-\nabla_x)^\top AD(\nabla_x)W = \\ &\quad -2D(\nabla_x)^\top BAD(\nabla_x)v \quad \text{in } \Omega\,, \\ &D(n)AD(\nabla_x)W = \\ &\quad 2D(n)AD(\nabla_x)^\top BAD(\nabla_x)v \quad \text{on } \partial\Omega\,. \end{aligned} \tag{10}$$

Furthermore, we define the special functions $\mathbf{z}^j$ solving the exterior elasticity problem

$$D(-\nabla_\xi)^\top AD(\nabla_\xi)\mathbf{z}^j = 0 \quad \text{in } G = \mathbb{R}^3 \setminus \omega_1\,, \tag{11}$$

$$D(n(\xi))^\top AD(\nabla_\xi)\mathbf{z}^j = \mathbf{g}^j \quad \text{on } \partial\omega_1 \tag{12}$$

with the special right-hand sides

$$\mathbf{g}^j(\xi) = -D(n(\xi))^\top A\mathbf{e}^j\,, \tag{13}$$

where $j = 1, \ldots, 6$ and $\mathbf{e}^j = \left(\delta_{j,1}, \ldots, \delta_{j,6}\right)^\top$ is an element of the canonical basis in $\mathbb{R}^6$.

The final formula has the following form.

Theorem 1 *The following formula holds true:*

$$\begin{aligned} \mathbb{J}^1_\rho(u) = \mathbb{J}^1_0(v) + \rho^3\Big\{&\epsilon^0(v)^\top ABA\epsilon^0(v)|\omega_1| \\ &+ (ABAD(\nabla_\xi)\mathbf{z}\epsilon^0(v), D(\nabla_\xi)\mathbf{z}\epsilon^0(v))_{\mathbb{R}^3\setminus\omega_1} \\ &+ (\epsilon^0(W) - 2BA\epsilon^0(v))^\top \mathbf{m}^\omega\epsilon^0(v)\Big\} \\ &+ O(\rho^{3+\delta})\,, \end{aligned} \tag{14}$$

where $\epsilon^0(v) = D(\nabla_\xi)v(\mathcal{O})$ *and* $\epsilon^0(W) = D(\nabla_\xi)W(\mathcal{O})$ *are strain columns evaluated at the point* $x = \mathcal{O}$ *for the solutions of problems (9) and (10);* $\mathbf{m}^\omega$ *is the polarization matrix of size* 6×6 *for the cavity* ω *in the elastic space with Hooke's matrix* A*, and* $\mathbf{z} = (\mathbf{z}^1, \ldots, \mathbf{z}^6)$ *is the row of energy components of the special solutions to homogeneous exterior elasticity problem (11)–(12).*

In the particular case of $B(x) = \frac{1}{2}A^{-1}$, functional (7) coincides with the elastic energy (8). In addition, we have $W = v$; thus $\epsilon^0(W) - 2BA\epsilon^0(v) = 0$ and the last term in parentheses in (14) vanishes, and by (16) the sum of the first two terms equals $\frac{1}{2}\epsilon^0(v)^\top \mathbf{m}^\omega\epsilon^0(v)$. Thus, we have the relation

$$\begin{aligned} \mathbb{E}(u;\Omega(\rho)) = \mathbb{E}(u;\Omega) + \frac{1}{2}\rho^3\epsilon^0(v)^\top \mathbf{m}^\omega\epsilon^0(v) \\ + O(\rho^{3+\delta})\,. \end{aligned} \tag{15}$$

The 6×6 polarization matrix may be computed explicitly using the result given below.

Theorem 2 *The following integral representation holds true:*

$$\mathbf{m}^\omega_{jk} = \left(AD(\nabla_\xi)\mathbf{z}^j, D(\nabla_\xi)\mathbf{z}^k\right)_G + A_{jk}|\omega_1|\,. \tag{16}$$

We consider only the operator $\mathcal{L}(\nabla_x)$ with the constant coefficients; however, the main results of the article remain the same for the operators with variable coefficients.

Contact Problem for Plane Elasticity

We consider the two-dimensional elasticity problem in plane stress formulation. Unlike in (1)–(3) on a part Γ_0 of $\partial\Omega$, we assume clamped condition $u = 0$, on part Γ_g the load $\mathcal{N}^\Omega u = \{\sigma_{ij}n_j\}_{i=1,2} = g$, and on part Γ_c the condition of frictionless contact

$$\begin{aligned} u_n \geq 0, \quad \sigma_n \leq 0\,, \\ \sigma_n u_n = 0, \quad \sigma_\tau = \sigma.n - \sigma_n n = 0\,. \end{aligned} \tag{17}$$

Here $u_n = u_i n_i, \sigma_n = n_i \sigma_{ij} n_j, \sigma.n = \{\sigma_{ij} n_j\}_{i=1,2}$. We define also the ring $C(R,\rho) = \omega_R \setminus \omega_\rho$ with $R > \rho$ and such that $\omega_R \subset \Omega$.

In contrast to the previous section, it is now impossible to compute topological derivatives of shape functionals by means of adjoint variables without additional assumption of strict complementarity for unknown solutions. Therefore, we shall derive a method for computing the perturbation caused by ω_ρ in the solution itself.

The bilinear form corresponding to the elastic energy may be written as

$$a(\rho; u, v) = \frac{1}{2} \int_{\Omega(\rho)} \sigma^\top(u) \epsilon(v) \, \mathrm{d}x \tag{18}$$

for $u, v \in H^1(\Omega)^2$, and the linear form responsible for the work of external forces is

$$L(u) = \int_{\Gamma_g} u^\top g \, \mathrm{d}s \,. \tag{19}$$

We will use also the Steklov–Poincaré operator $\mathcal{A}_\rho$ defined in the following way. Consider the boundary value problem

$$\begin{aligned} \mathcal{L} w &= 0 \quad \text{in } C(R,\rho) \,, \\ \mathcal{N}^\omega w &= 0 \quad \text{on } \partial\omega_\rho \,, \\ w &= v \quad \text{on } \partial\omega_R \,. \end{aligned} \tag{20}$$

Then we set

$$\mathcal{A}_\rho(v) = \sigma_n(w) \quad \text{on } \partial\omega_R \,. \tag{21}$$

Thus $\mathcal{A}_\rho$ is a mapping

$$\mathcal{A}_\rho \colon H^{1/2}(\partial\omega_R)^2 \mapsto H^{-1/2}(\partial\omega_R)^2 \,. \tag{22}$$

In the latter part of the article it will be demonstrated constructively that

$$\mathcal{A}_\rho = \mathcal{A}_0 + \rho^2 \mathcal{B} + O(\rho^4) \tag{23}$$

in the linear operator norm corresponding to (22). Using this notation we have

$$\begin{aligned} a(\rho; u, u) = \frac{1}{2} \int_{\Omega(R)} \sigma^\top(u) \epsilon(u) \, \mathrm{d}x \\ + \frac{1}{2} \int_{C(R,\rho)} \sigma^\top(u) \epsilon(u) \, \mathrm{d}x \end{aligned} \tag{24}$$

as well as

$$\begin{aligned} \frac{1}{2} \int_{C(R,\rho)} \sigma^\top(u) \epsilon(u) \, \mathrm{d}x &= \frac{1}{2} \langle \mathcal{A}_\rho u, u \rangle_{\partial\omega_R} \\ &= \frac{1}{2} \langle \mathcal{A}_0 u, u \rangle_{\partial\omega_R} + \frac{1}{2} \rho^2 \langle \mathcal{B} u, u \rangle_{\partial\omega_R} + \mathcal{R}(u,u) \,, \end{aligned} \tag{25}$$

where $\mathcal{R}(u,u)$ is of the order $O(\rho^4)$ on bounded sets in $H^{1/2}(\partial\omega_R)^2$.

With $\mathcal{B}$ we associate the bilinear form

$$b(u,v) = \frac{1}{2} \langle \mathcal{B} u, u \rangle_{\partial\omega_R} \tag{26}$$

and observe that

$$a(0; u, u) = \frac{1}{2} \langle \mathcal{A}_0 u, u \rangle_{\partial\omega_R} \tag{27}$$

corresponds to the internal elastic energy in the entire domain. Denote also by u_0 the solution to the contact problem in the domain without a hole. We have thus the approximation of the energy form

$$a(\rho; u, u) = a(0; u, u) + \rho^2 b(u,u) \,. \tag{28}$$

Let also

$$H^1_{\Gamma_0}(\Omega) = \{ v \in H^1(\Omega)^2 | v = 0 \quad \text{on } \Gamma_0 \}$$

and K be the convex cone

$$K = \{ v \in H^1_{\Gamma_0}(\Omega) | v_n \geq 0 \quad \text{on } \Gamma_c \} \,.$$

Then the following variational inequality solves our contact problem in $\Omega(\rho)$:

$$u \in K \colon \; a(\rho; u, u - v) \geq L(v - u) \quad \forall v \in K \,. \tag{29}$$

Taking into account approximation (28) and using abstract results on the differentiability of metric projection onto the polyhedric convex sets in Dirichlet space [33] we have the following result.

Theorem 3 *For ρ sufficiently small we have on Ω_R the following expansion of the solution u with respect to the parameter ρ at $0+$:*

$$u = u_0 + \rho^2 q + o(\rho^2) \quad \text{in } H^1(\Omega_R)^2 \,, \tag{30}$$

where the topological derivative q of the solution u to the contact problem is given by the unique solution of the

following variational inequality:

$$q \in S_K(u)\colon\ a(0;q,v-q)+b(u,v-q) \geq 0 \quad \forall v \in S_K(u), \tag{31}$$

where

$$S_K(u) = \Big\{ v \in (H^1_{\Gamma_0}(\Omega)^2 | v_n \leq 0 \quad on\ \Xi(u)\,, \\ a(0;u,v) = 0 \Big\}\,. \tag{32}$$

The coincidence set

$$\Xi(u) = \{x \in \Gamma_c | u_n(x) = 0\}$$

is well defined [33,42], *for any function* $u \in H^1(\Omega)^2$, *and* $u_0 \in K$ *is the solution of variational inequality* (29) *for* $\rho = 0$.

The perturbation q gives an approximation of u outside ω_R. In the ring $C(R,\rho)$ one can, as we shall see, compute the solution separately.

Solution of the Elasticity System in the Ring

Let us consider the plane elasticity problem in the ring $C(R,\rho)$. We use polar coordinates (r,θ) with $\mathbf{e}_r$ pointing outward and $\mathbf{e}_\theta$ perpendicularly in the counterclockwise direction. Assume that the displacement on the outer boundary is given, while the inner boundary is free. We want to compare the solution to such a problem to one defined in the full circle, with the same displacement data. To this end we shall construct the exact representation of both solutions, using the complex variable method of [25]. It was shown there that

$$\begin{aligned} \sigma_{rr} - i\sigma_{r\theta} &= 2\Re\phi' - e^{2i\theta}(\bar{z}\phi'' + \psi')\,, \\ \sigma_{rr} + i\sigma_{\theta\theta} &= 4\Re\phi'\,, \\ 2\mu(u_r + iu_\theta) &= e^{-i\theta}(\kappa\phi - z\bar{\phi}' - \bar{\psi})\,, \end{aligned} \tag{33}$$

where φ and ψ are given by complex series

$$\begin{aligned} \phi &= A\log(z) + \sum_{k=-\infty}^{k=+\infty} a_k z^k\,, \\ \psi &= -\kappa\bar{A}\log(z) + \sum_{k=-\infty}^{k=+\infty} b_k z^k\,. \end{aligned} \tag{34}$$

Here μ is the Lame constant, ν is the Poisson ratio, $\kappa = 3-4\nu$ in the plain strain case, and $\kappa = (3-\nu)/(1+\nu)$ for plane stress.

The displacement data are given in the form of Fourier series

$$2\mu(u_r + iu_\theta) = \sum_{k=-\infty}^{k=+\infty} A_k e^{ik\theta}\,. \tag{35}$$

The traction-free condition on some circle means $\sigma_{rr} = \sigma_{r\theta} = 0$. From (33) and (34) we get for displacements the formula

$$\begin{aligned} 2\mu(u_r + iu_\theta) &= 2\kappa A \sim r\log(r)\frac{1}{z} - \bar{A}\frac{1}{r}z \\ &+ \sum_{p=-\infty}^{p=+\infty}\Big[\kappa r a_{p+1} - (1-p)\bar{a}_{1-p}r^{-2p+1} \\ &- \bar{b}_{-(p+1)}r^{-2p-1}\Big]z^p\,. \end{aligned} \tag{36}$$

Similarly we obtain a representation of tractions on some circle

$$\begin{aligned} \sigma_{rr} - i\sigma_{r\theta} &= 2A\frac{1}{z} + (\kappa+1)\frac{1}{r^2}\bar{A}z \\ &+ \sum_{p=-\infty}^{p=+\infty}(1-p)\Big[(1+p)a_{p+1} + \bar{a}_{1-p}r^{-2p} \\ &+ \frac{1}{r^2}b_{p-1}\Big]z^p\,. \end{aligned} \tag{37}$$

Denote $d_0 = \kappa a_0 - \bar{b}_0$. For the full circle we must eliminate singularities, i. e., $b_{-k} = a_{-k} = A = 0$ for $k = 1, 2, \ldots$ and then, using (36), obtain

$$\begin{aligned} d_0^0 &= A_{-1} + \frac{2}{\kappa}\bar{A}_1\,, \\ \Re a_1^0 &= \frac{1}{(\kappa-1)R}\Re A_0\,, \\ \Im a_1^0 &= \frac{1}{(\kappa+1)R}\Im A_0\,, \\ a_k^0 &= \frac{1}{\kappa R^k}A_{k-1}, \quad k > 1\,, \\ b_k^0 &= -\frac{1}{R^k}[(k+2)\frac{1}{\kappa}A_{k+1} + \bar{A}_{-(k+1)}], \quad k > 1\,. \end{aligned} \tag{38}$$

In further analysis we consider the ring and take for the sake of simplicity $R = 1$ as well as $\rho < 0.5$. This is only rescaling and does not diminish generality. Then from (36) for $r = R = 1$ and (37) for $r = \rho$, we get

$A = 0$ and

$$
\begin{aligned}
d_0 &= A_{-1} + \frac{2R^4}{\kappa R^4 + \rho^4}\bar{A}_1 \,, \\
a_2 &= \frac{R^2}{\kappa R^4 + \rho^4} A_1 \,, \\
\Re a_1 &= \frac{R}{(\kappa-1)R^2 + 2\rho^2}\Re A_0 \,, \\
\Im a_1 &= \frac{1}{\kappa+1}\Im A_0 \,, \\
b_{-1} &= -2\rho^2 \Re a_1 = -\frac{2\rho^2 R}{(\kappa-1)R^2 + 2\rho^2}\Re A_0 \,, \\
b_{-2} &= -\rho^4 \bar{a}_2 = -\frac{\rho^4 R^2}{\kappa R^4 + \rho^4}\bar{A}_1 \,.
\end{aligned}
$$

Observe that

$$
\begin{aligned}
d_0 - d_0^0 &= -\rho^4 \frac{2}{\kappa(\kappa R^4 + \rho^4)}\bar{A}_1 \,, \\
a_1 - a_1^0 &= -\rho^2 \frac{2}{(\kappa-1)R((\kappa-1)R^2 + 2\rho^2)}\Re A_0 \,, \\
a_2 - a_2^0 &= -\rho^4 \frac{1}{\kappa R^2(\kappa R^4 + \rho^4)} A_1 \,.
\end{aligned}
\tag{39}
$$

Again using (36) and (37) we obtain for $k \geq 2$

$$
\begin{bmatrix} a_{-(k-1)} \\ b_{-(k+1)} \end{bmatrix} = T_k(\rho) \cdot \begin{bmatrix} \bar{a}_{k+1} \\ \bar{b}_{k-1} \end{bmatrix} \tag{40}
$$

where

$$
T_k(\rho) = \begin{bmatrix} -(k+1)\rho^{2k} & , & -\rho^{2(k-1)} \\ -k^2\rho^{2(k+1)} & , & -(k-1)\rho^{2k} \end{bmatrix}
$$

and the system which may be rewritten as

$$
S_k(\rho) \cdot \begin{bmatrix} a_{k+1} \\ b_{k-1} \end{bmatrix} = \begin{bmatrix} A_k \\ \bar{A}_{-k} \end{bmatrix} \tag{41}
$$

with entries

$$
\begin{aligned}
S_k(\rho)_{11} &= \kappa R^{k+1} - (k^2-1)R^{1-k}\rho^{2k} \\
&\quad + k^2 R^{-(k+1)}\rho^{2(k+1)} \,, \\
S_k(\rho)_{12} &= -(k-1)(R^{1-k}\rho^{2(k-1)} - R^{-(k+1)}\rho^{2k}) \,, \\
S_k(\rho)_{21} &= -(k+1)(R^{k+1} + \kappa R^{1-k}\rho^{2k}) \,, \\
S_k(\rho)_{22} &= -R^{k-1} - \kappa R^{1-k}\rho^{2(k-1)} \,.
\end{aligned}
\tag{42}
$$

In fact formulas (40) and (42) are correct also for $k = 0, 1$ and $\rho > 0$. Together with initial values d_0,a_1, a_2,b_{-1},b_{-2} they allow us to compute all a_k,b_k for any $-\infty < k < +\infty$.

The matrix $S_k(\rho)$ is a perturbation of $S_{k(0)}$, which would produce the solution for the full circle, namely, a_{k+1}^0, b_{k-1}^0. Observe that $T_k(0) = 0$. Direct computations lead to estimates

$$
|a_3 - a_3^0| \leq \Lambda\left(|A_2|\rho^4 + |A_{-2}|\rho^2\right) \tag{43}
$$

and for $k = 4, 5, \ldots$

$$
|a_k - a_k^0| \leq \Lambda\left(|A_{k-1}|\rho^{3(k-1)/2} + |A_{1-k}|\rho^{3(k-2)/2}\right) \,, \tag{44}
$$

where the exponent $k/2$ has been used to counteract the growth of k^2 in terms like $k^2\rho^{k/2}$. Similarly,

$$
|b_1 - b_1^0| \leq \Lambda\left(|A_2|\rho^4 + |A_{-2}|\rho^2\right) \,, \tag{45}
$$

and for $k = 2, 3, \ldots$

$$
|b_k - b_k^0| \leq \Lambda\left(|A_{k+1}|\rho^{3(k+1)/2} + |A_{-(k+1)}|\rho^{3k/2}\right) \,. \tag{46}
$$

From relation (40) we get another estimate

$$
|a_{-k}| \leq \Lambda\rho^{2k}\left(|A_{k+1}| + |A_{-(k+1)}|\right), \quad k = 1, 2, \ldots \tag{47}
$$

and

$$
|b_{-k}| \leq \Lambda\rho^{2(k-1)}\left(|A_{k-1}| + |A_{1-k}|\right), \quad k = 3, 4, \ldots \,. \tag{48}
$$

Here Λ is a constant independent of ρ and A_i. Observe that the corrections proportional to ρ^2 are present only in a_1, a_3, b_{-1}, a_{-1}. The rest is at least of the order $O(\rho^3)$ (in fact $O(\rho^4)$).

Explicit Expansion of Elastic Energy The elastic energy contained in the ring has the form

$$
\begin{aligned}
2\mathbb{E}(\rho, R) &= \int_{C(\rho,R)} \sigma(u_\rho) : \epsilon(u_\rho)\, \mathrm{d}x \\
&= \int_{\partial\omega_R} u_\rho \sigma(u_\rho).n \, \mathrm{d}s \,.
\end{aligned}
\tag{49}
$$

Since $u_\rho = u$ on $\partial\omega_R$,

$$
2\mathbb{E}(\rho, R) = \int_{\partial\omega_R} u\sigma(u_\rho).n \, \mathrm{d}s \,. \tag{50}
$$

Now $\sigma(u_\rho)$ is in fact of the form $\sigma(u_\rho) = \sigma_\rho(u)$, because $u_\rho = u$ on $\partial\omega_R$, which means that $u_\rho = u_\rho(u)$. If we split σ_ρ into

$$\sigma_\rho(u) = \sigma^0 + \rho^2\sigma^1(u) + O(\rho^4)\,, \tag{51}$$

then

$$\mathbb{E}(\rho, R) = \mathbb{E}(0, R) + \rho^2 \int_{\partial\omega_R} u\sigma^1(u).n \; \mathrm{d}s + O(\rho^4). \tag{52}$$

Thus the problem of defining the operator $\mathcal{B}$ reduces to finding $\sigma^1(u)$.

From (33) and (34) we know that $\sigma_\rho(u)$ is a linear function of infinite vectors $\boldsymbol{a}, \boldsymbol{b}$, while $\sigma^0(u)$ is the same function of $\boldsymbol{a}^0, \boldsymbol{b}^0$. Here $\boldsymbol{a}^0, \boldsymbol{b}^0$ are computed for ω^R, while $\boldsymbol{a}, \boldsymbol{b}$ correspond to $C(\rho,R)$. To obtain $\sigma^1(u)$ it is enough to express $\boldsymbol{a}, \boldsymbol{b}$ as

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{a}^0 + \rho^2\boldsymbol{a}^1 + O(\rho^4)\,, \\ \boldsymbol{b} &= \boldsymbol{b}^0 + \rho^2\boldsymbol{b}^1 + O(\rho^4)\,, \end{aligned} \tag{53}$$

because then

$$\sigma^1(u) = \sigma^1(\boldsymbol{a}^1, \boldsymbol{b}^1; u)\,.$$

Let us observe as well that

$$\begin{aligned} \int_{\partial\omega_R} u\sigma^1(u).n \; \mathrm{d}s &= R\int_0^{2\pi} (\sigma^1_{rr}u_r + \sigma^1_{r\theta}u_\theta)\; \mathrm{d}\theta \\ &= R\int_0^{2\pi} \Re[(\sigma^1_{rr} - i\sigma^1_{r\theta})(u_r + iu_\theta)]\; \mathrm{d}\theta\,. \end{aligned} \tag{54}$$

The analysis of formulae (38) for $\boldsymbol{a}^0, \boldsymbol{b}^0$ and their counterparts $\boldsymbol{a}, \boldsymbol{b}$ leads to the conclusion that the only nonzero terms in $\boldsymbol{a}^1, \boldsymbol{b}^1$ will be $a^1_3, a^1_1, a^1_{-1}, b^1_{-1}, b^1_1$.

Taking into account that $A = 0$ in (34) for our problem,

$$\begin{aligned} \phi &= \phi^0 + \rho^2\phi^1 + O(\rho^4)\,, \\ \psi &= \psi^0 + \rho^2\psi^1 + O(\rho^4)\,, \end{aligned} \tag{55}$$

where

$$\phi^1 = a^1_{-1}\frac{1}{z} + a^1_1 z + a^1_3 z^3, \quad \psi^1 = b^1_{-1}\frac{1}{z} + b^1_1 z\,. \tag{56}$$

Using all the results collected so far gives the final expression for $\mathcal{B}$:

$$\begin{aligned} \int_{\partial\omega_R} u\sigma^1(u).n \; \mathrm{d}s = \frac{1}{R^2}\Big[&\frac{2(\kappa-2)}{(\kappa-1)^2}(\Re A_0)^2 \\ &- (\kappa+1)|A_{-2}|^2 - \frac{9(\kappa+1)}{\kappa^2}|A_2|^2 \\ &- \frac{6(\kappa+1)}{\kappa}\Re(A_2A_{-2})\Big]\,. \end{aligned} \tag{57}$$

Taking into account the formulae for Fourier coefficients

$$\begin{aligned} A_0 &= \frac{\mu}{\pi}\int_0^{2\pi} (u_1 + iu_2)e^{-i\theta}\; \mathrm{d}\theta\,, \\ A_2 &= \frac{\mu}{\pi}\int_0^{2\pi} (u_1 + iu_2)e^{-3i\theta}\; \mathrm{d}\theta\,, \\ A_{-2} &= \frac{\mu}{\pi}\int_0^{2\pi} (u_1 + iu_2)e^{+i\theta}\; \mathrm{d}\theta\,, \end{aligned} \tag{58}$$

we conclude that $\mathcal{B}$ is indeed the well-defined bilinear form which contains squares of integrals of u over $\partial\omega_r$. In addition, from (56) follows the theorem below.

Theorem 4 *If $u \in H^{1/2}(\partial\omega_R)^2$, which is equivalent to*

$$\sum_{k=-\infty}^{k=+\infty} \sqrt{1+k^2}|A_k|^2 \le \Lambda_0\,, \tag{59}$$

then the rest $\mathcal{R}(u,u)$ in formula (25) is uniformly bounded by some constant depending only on Λ_0.

The derivation sketched above allows one also to obtain a higher-order expansion of the Steklov–Poincaré operator.

Cases

Plane Isotropic Elasticity System

Let us consider the isotropic elasticity equations in the plane

$$\begin{aligned} \mathcal{L}u_0 &= f \quad \text{in} \quad \Omega\,, \\ u_0 &= g \quad \text{on} \quad \Gamma_1\,, \\ \mathcal{N}^\Omega u_0 &= h \quad \text{on} \quad \Gamma_2\,, \end{aligned}$$

and the same system for u in the domain with the circular hole ω_ρ centered at $x_0 \in \Omega$, with the additional condition

$$\mathcal{N}^\omega u = 0 \quad \text{on} \quad \partial\omega_\rho\,.$$

Observe the presence of the volume forces denoted by f. Isotropicity means here that the matrix of material coefficients has a particular form

$$A = \begin{bmatrix} \lambda+2\mu & , & \lambda & , & 0 \\ \lambda & , & \lambda+2\mu & , & 0 \\ 0 & , & 0 & , & 2\mu \end{bmatrix}\,.$$

We introduce the yield functional

$$J_\sigma(\rho) = \int_{\Omega(\rho)} \sigma(u^\rho)^\top S\sigma(u^\rho)\,\mathrm{d}x\,, \tag{60}$$

where S is an isotropic matrix. Again, isotropicity means that S may be expressed as follows:

$$S = [s_{ij}] = \begin{bmatrix} l+2m & l & 0 \\ l & l+2m & 0 \\ 0 & 0 & 2m \end{bmatrix},$$

where l,m are any real constants. Their values depend on the particular yield criterion (e. g., maximal shear stress or Huber). The following assumption assures that the problem is well defined.

(A) The domain Ω has a piecewise smooth boundary, but pure cracks are admissible, even having different types of boundary conditions prescribed on both edges (i. e., tractions and displacements). Then g,h must be compatible with $u \in H^1(\Omega)^2$.

The interior regularity of u in Ω is determined by the regularity of the right-hand side f of the elasticity system. For such a problem the formulae given in Sect. "Three-Dimensional Anisotropic Elastic Body with a Small Cavity" may be computed exactly, even in the more general case of the presence of volume forces [35].

In this case the adjoint state $v \in H^1_{\Gamma_1}(\Omega)$ satisfies for all test functions $\phi \in H^1_{\Gamma_1}(\Omega)$ the following integral identity:

$$-\int (D(\nabla_x)v)^\top A\sim D(\nabla_x)\phi\,\mathrm{d}x = 2\int \sigma(u)^\top S\sigma(\phi)\,\mathrm{d}x. \tag{61}$$

Denote $\eta = l(1+4\lambda\frac{\nu}{E}+4\mu\frac{\nu}{E})+2m(1+2\lambda\frac{\nu}{E})$. Now we may formulate the following result:

Theorem 5 *Assume that the distributed force is sufficiently regular, $f \in C^1(\Omega)^2$, and (A) then the topological derivative of the functional J_σ is given by*

$$\mathcal{T}(x_0) = -\Big[\eta(a_u^2+2b_u^2)+2f^\top v + \frac{1}{E}(a_u a_v + 2b_u b_v\cos 2\delta)\Big]_{x=x_0}. \tag{62}$$

Here

$$\eta = l\left(1+4\lambda\frac{\nu}{E}+4\mu\frac{\nu}{E}\right)+2m\left(1+4\lambda\frac{\nu}{E}\right).$$

Some of the terms in (62) require explanation. In the reference frame tied to the principal stress directions for the displacement fields u,w,v, they are given by the expressions:

$$a_u = \sigma_{11}(u)+\sigma_{22}(u),\quad b_u = \sigma_{11}(u)-\sigma_{22}(u)\,,$$
$$a_v = \sigma_{11}(v)+\sigma_{22}(v),\quad b_v = \sigma_{11}(v)-\sigma_{22}(v)\,.$$

Finally, the angle δ denotes the angle between principal stress directions for displacement fields u and v in (62) and E, ν stands for Young's modulus and Poisson constant. By principal stress directions we mean, as usually, the coordinate system in which the stress tensor is diagonal.

Three-Dimensional Isotropic Elasticity Systems

Now we consider the same system as in Sect. "Plane Isotropic Elasticity System", only in $\mathbb{R}^3$. The isotropic matrix of material (Lame) coefficients is now

$$A = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\mu \end{bmatrix}.$$

The yield functional is similar,

$$J_\sigma(\rho) = \int_{\Omega(\rho)} [\sigma(u^\rho)^\top S\sigma(u^\rho)]\,\mathrm{d}x\,, \tag{63}$$

where S is an isotropic matrix. Isotropicity means here again that S may be expressed as follows:

$$S = \begin{bmatrix} l+2m & l & l & 0 & 0 & 0 \\ l & l+2m & l & 0 & 0 & 0 \\ l & l & l+2m & 0 & 0 & 0 \\ 0 & 0 & 0 & 2m & 0 & 0 \\ 0 & 0 & 0 & 0 & 2m & 0 \\ 0 & 0 & 0 & 0 & 0 & 2m \end{bmatrix},$$

where l,m are real constants. Some yield criteria fit into this framework, but not maximal shear stress. The following assumption assures that J_σ is well defined for solutions of the elasticity system.

(A) The domain Ω has a piecewise smooth boundary, which may have reentrant corners with $\alpha < 2\pi$ created by the intersection of two planes. In addition,

g,h must be compatible with $u \in H^1(\Omega)^3$. With respect to f we assume that it is a continuous vector field and $f \in H^1(\Omega)^3$.

The interior regularity of the displacement field u in Ω is determined by the regularity of the right-hand side f of the elasticity system. To be precise: Let $\delta > 0$ be given and $\Omega^\delta = \Omega \setminus (\Gamma + B_\delta(0))$. Then $u \in H^3(\Omega^\delta)^3$, see [26].

The adjoint state equation for shape functional J_σ takes on exactly the same form as in (61)

Now we may formulate the following result, giving the constructive method for computing the topological derivative:

Theorem 6 *Assume that the distributed force is sufficiently regular, $f \in C^1(\Omega;\mathbb{R}^3)$, and (A) is satisfied, then*

$$\mathcal{T} J_\sigma(x_0) = -\frac{1}{4\pi}[K(S;\sigma(u),\sigma(u)) + 4\pi f^T v + K(A^{-1};\sigma(u),\sigma(v))]_{x=x_0}\,, \tag{64}$$

where $v \in H^1_{\Gamma_1}(\Omega)$ is the adjoint state satisfying the integral identities (61).

Function K is defined as an integral over the unit sphere $\partial B_1(0) = \{x \in \mathbb{R}^3 | \|x\| = 1\}$ of the following functions:

$$K(S;\sigma(u(x_0)),\sigma(u(x_0))) = \int_{\partial B_1(0)} \sigma^\infty(u(x_0);x)^T \cdot S \cdot \sigma^\infty(u(x_0);x)\,\mathrm{d}S$$

$$K(A^{-1};\sigma(u(x_0)),\sigma(v(x_0))) = \int_{\partial B_1(0)} \sigma^\infty(u(x_0);x)^T \cdot A^{-1} \cdot \sigma^\infty(v(x_0);x)\,\mathrm{d}S\,.$$

The symbol $\sigma^\infty(u(x_0);x)$ denotes the explicit solution to the exterior elasticity problem, constructed from the so-called Leon solutions [12] in the way specified below. It satisfies the following boundary conditions in the infinite exterior domain $\mathbb{R}^3 \setminus \overline{B_1(0)}$:

- No tractions are applied on the surface $\partial B_1(0)$ of the ball;
- The stresses $\sigma^\infty(u(x_0);x)$ tend to the constant value $\sigma(u(x_0))$ as $\|x\| \to \infty$.

In this notation $\sigma^\infty(u(x_0);x)$ is a function of space variables depending on the functional parameter $u(x_0)$, while $\sigma(u(x_0))$ is a value of the stress tensor computed at the point x_0 for the displacement field u. The dependence between $\sigma^\infty(u(x_0);x)$ and $\sigma(u(x_0))$ results from the boundary condition at infinity listed above. The method for obtaining such solutions (and the displacement field u^∞), based on [12], is given in [36].

The main difficulty is related to the computation of the values of the functions denoted above as $K(S;\sigma(u(x_0)),\sigma(u(x_0)))$ and $K(A^{-1};\sigma(u(x_0)),\sigma(w(x_0)))$, which cannot be obtained in the closed form, in contrast with the two-dimensional case. This is due to the fact that the principal stress directions for u and v may be rotated with respect to each other, and this rotation is not specified by a single parameter δ.

Therefore we must approximate these functions using numerical quadrature formulae. It is possible, because we may calculate the values of integrands defining K at any point on the sphere. This makes the computations more involved but does not increase the numerical complexity in comparison to evaluating single closed form expression in the case of two dimensions. The detailed procedure is given in [36].

Conclusions

We list some applications of topological derivatives in numerical methods of resolution for shape-optimization and inverse problems.

A numerical coupling of two methods, boundary variations by a level set method and topological derivatives, in shape and topology optimization of structures is proposed in [1,4,7,9,10,13]. On the one hand, the level set method, based on the classical shape derivative, is known to easily handle boundary propagation with topological changes. However, in practice it does not allow for the nucleation of new holes (at least in two spatial dimensions). On the other hand, the topological derivative method is precisely designed for introducing new holes in the optimization process. Therefore, the coupling of these two methods yields an efficient algorithm that can escape from local minima in a given topological class of shapes. Both methods rely on a notion of gradient computed through an adjoint analysis and have a low CPU cost since they capture a shape on a fixed Eulerian mesh. The main advantage of our coupled algorithm is to make the resulting optimal design largely independent of the initial guess.

The paper [2] is devoted to minimum stress design in structural optimization. The efficient numerical algorithm for shape and topology optimization is based

on the level set method coupled with the topological derivative. Several numerical examples in two and three spatial dimensions are discussed.

The Brazilian group has been working on topological derivatives since early 2000. Novotny in his Ph.D. dissertation proposes a method to calculate the topological derivative based on classical shape sensitivity analysis [31].

The topological derivatives for partial differential equations on graphs are introduced in [19].

Topological sensitivity analysis can be performed in the framework of the piecewise constant Mumford–Shah functional. Topological and shape derivatives can be combined to derive a fast algorithm for image segmentation, without any initialization required. The general Mumford–Shah functional is also investigated, see, e. g., [14], see also [5].

References

1. Allaire G, de Gournay F, Jouve F, Toader A-M (2005) Structural optimization using topological and shape sensitivity via a level set method. Control Cybern 34:59–80
2. Allaire G, Jouve F (2007) Minimum stress optimal design with the level set method, to appear in Int J Boundary Elem Appl
3. Amstutz S, Horchani I, Masmoudi M (2005) Crack detection by the topological gradient method. Control Cybern 34:119–138
4. Amstutz S, Andrä H (2006) A new algorithm for topology optimization using a level-set method. J Comput Phys 216(2):573–588
5. Auroux D, Masmoudi M Image processing by topological asymptotic expansion. J Math Imag Vis, to appear
6. Bonnet M, Guzina BB (2004) Sounding of finite solid bodies by way of topological derivative. Int J Numer Methods Eng 61:2344–2373
7. Burger M, Hackl B, Ring W (2004) Incorporating Topological Derivatives into Level Set Methods. J Comput Phys 194:344–362
8. Eschenauer HA, Kobelev VV, Schumacher A (1994) Bubble method for topology and shape optimization of structures. Struct Optim 8:42–51
9. Fulmanski P, Laurain A, Scheid J-F, Sokolowski J (2007) A levelset method in shape and topology optimization for variational inequalities. Int J Appl Math Comput Sci 17(3):413–430
10. Fulmanski P, Laurain A, Scheid J-F, Sokolowski J (2007) Level set method with topological derivatives in shape optimization Les Prépublications de l'Institut Elie Cartan
11. Garreau S, Guillaume P, Masmoudi M (2001) The topological asymptotic for PDE systems: the elasticity case. SIAM J Control Optim 39(6):1756–1778
12. Hahn HG (1985) Elastizitätstheorie. Teubner, Stuttgart
13. Hintermüller M (2005) Fast level-set based algorithms using shape and topological sensitivity information. Control Cybern 34(1):305–324
14. Hintermüller M (2006) A combined shape Newton and topology optimization technique in real-time image segmentation. In: Biegler L, Ghattas O, Heinkenschloss M, Keyes D, van Bloemen Waanders B (eds) Real-Time PDE-Constrained Optimization, SIAM
15. Il'in AM (1992) Matching of Asymptotic Expansions of Solutions of Boundary Value Problems. AMS, p 281
16. Jackowska-Strumiłło L, Sokołowski J, Żochowski A, Henrot A (2002) On numerical solution of shape inverse problems. Comput Optim Appl 23:231–255
17. Kamotski IV, Nazarov SA (1998) Spectral problems in singular perturbed domains and selfadjoint extensions of differential operators. Trudy St.-Petersburg Mat Obshch 6: 151–212 (Engl. transl. in: Proceedings of the St. Petersburg Mathematical Society (2000), Am Math Soc Transl Ser 2 6:127–181, 199, Am Math Soc, Providence, RI)
18. Laurain A (2006) Domaines singulierements perturbes en optimisation de formes. Ph.D. dissertation, Institut Elie Cartan, Nancy, http://tel.archives-ouvertes.fr/tel-00139595
19. Leugering G, Sokolowski J Topological derivatives for ellipticv problems on graphs. In: Novotny A, Tarocco E, de Souza Neto E (eds) The book for Raul Feijoo, to be published by International Center for Numerical, Methods in Engineering (CIMNE), Barcelona, in press
20. Lewinski T, Sokolowski J (1998) Optimal shells formed on a sphere. The topological derivative method. RR-3495, INRIA-Lorraine
21. Lewinski T, Sokolowski J (2000) Topological derivative for nucleation of non-circular voids Contemporary Mathematics. Am Math Soc 268:341–361
22. Lewinski T, Sokolowski J (2003) Energy change due to appearing of cavities in elastic solids. Int J Solids Struct 40:1765–1803
23. Lewinski T, Sokolowski J, Żochowski A (1999) Justification of the bubble method for the compliance minimization problems of plates and spherical shells. CD-Rom, 3rd World Congress of Structural and Multidisciplinary Optimization (WCSMO-3) Buffalo/Niagara Falls, NY, May 17–21
24. Mazja WG, Nazarov SA, Plamenevskii BA (1991) Asymptotische Theorie elliptischer Randwertaufgaben in singulär gestörten Gebieten. 1, 2. Akademie-Verlag, Berlin. (English transl.: Asymptotic theory of elliptic boundary value problems in singularly perturbed domains, vol 1, 2. Birkhäuser, Basel (2000))
25. Muskhelishvili NI (1953) Some Basic Problems on the Mathematical Theory of Elasticity. Noordhoff, Groningen
26. Nazarov SA, Plamenevsky BA (1994) Elliptic Problems in Domains with Piecewise Smooth Boundaries. De

Gruyter Exposition in Mathematics 13. Walter de Gruyter, Berlin
27. Nazarov SA, Sokołowski J (2003) Asymptotic analysis of shape functionals. J Math Pures Appl 82(2):125–196
28. Nazarov SA, Sokołowski J (2004) Topological derivative of the Dirichlet integral due to formation of a thin ligaments. Siberian Math J 45:341–355
29. Nazarov SA, Sokołowski J (2006) Self adjoint extensons for the Neumann laplacian in application to shape optimization. Act Math Sin Engl Ser 22:879–906
30. Nazarov SA, Slutskij AS, Sokołowski J (2005) Topological derivative of the energy functional due to formation of a thin ligament on a spatial body. Folia Math 12:39–72
31. Novotny AA (2003) Anàlise de Sensibilidade Topológica. Ph.D. Thesis. LNCC/MCT, Petrópolis
32. Novotny AA, Feijo RA, Padra C, Taroco E (2005) Topological-Shape Sensitivity Analysis Applied to Topology Design of Kirchhoff's Plate Bending Problem. Control Cybern 34(1):339–361
33. Rao M, Sokołowski J (2000) Tangent sets in Banach spaces and applications to variational inequalities. Les prépublications de l'Institut Élie Cartan 42
34. Rocha de Faria J, Novotny AA, Feijo RA, Taroco E, Padra C (2007) Second Order Topological Sensitivity Analysis. Int J Solids Struct 44(14–15):4958–4977
35. Sokołowski J, Żochowski A (1999) On topological derivative in shape optimization. SIAM J Control Optim 37(4):1251–1272
36. Sokołowski J, Żochowski A (1999) Topological derivative for optimal control problems. Control Cybern 28(3):611–626
37. Sokołowski J, Żochowski A (1999) Topological derivatives for elliptic problems. Inverse Problems 15(1):123–134
38. Sokołowski J, Żochowski A (2001) Topological derivatives of shape functionals for elasticity systems. Mech Struct Mach 29(3):333–351
39. Sokołowski J, Żochowski A (2003) Optimality conditions for simultaneous topology and shape optimization. SIAM J Control Optim 42:1198–1221
40. Sokołowski J, Żochowski A (2005) Modelling of topological derivatives for contact problems. Num Math 102:145–179
41. Sokołowski J, Żochowski A (2007) Asymptotic analysis and topological derivatives for shape and topology optimization of elasticity problems in two spatial dimensions. Prépublication IECN, 16
42. Sokołowski J, Zolesio J-P (1992) Introduction to Shape Optimization. Shape Sensitivity Analysis. Springer, Berlin

Topological Methods in Complementarity Theory

GEORGE ISAC
Royal Military College of Canada, Kingston, Canada

MSC2000: 90C33

Article Outline

Keywords
Preliminaries
Topological Degree
and Complementarity Problems
Exceptional Families of Elements
and Complementarity Problems
Homotopy Continuation Method
Zero-Epi Mappings
and Complementarity Problems
Properties
Conclusions
See also
References

Keywords

Topological methods; Degree theory; Zero-epi mapping; Complementarity

Complementarity theory is dedicated to the study of complementarity problems. The concept of *complementarity* is fundamental to the study of many optimization problems and to the analysis and computation of equilibria in the physical and economical sense. It is well known that the complementarity theory has also many and remarkable applications in Engineering, Elasticity, Mechanics, Game Theory etc. The solution set of a complementarity problem can be empty or nonempty, stable or unstable. When the *solution set* is nonempty, the problem is, how can we compute a solution. The classical existence results for complementarity problems were proved using the Hartman – Stampacchia theorem, Karamardian 's theorem, some fixed point theorems and for the linear complementarity problem using algebraic tools. A class of powerful methods used recently in complementarity theory is the class of *topological methods*. By topological methods we can prove existence theorems, we can study the *stability of the solution set* or we can study some particular topological properties of the solution set. In what follows, we shall present some known topological methods.

Preliminaries

Denote by $(\mathbf{R}^n, \langle \cdot, \cdot \rangle)$ the n-dimensional Euclidean space, by $(H, \langle \cdot, \cdot \rangle)$ a Hilbert space and by $(E, \| \cdot \|)$

a real Banach space. If $(E, \|\cdot\|)$ is a Banach space, denote by E^* the topological dual of E and by $\langle E, E^*\rangle$ a duality defined by a canonical bilinear form $\langle \cdot, \cdot \rangle$ defined on $E \times E^*$. We say that $\mathbf{K} \subset E$ is a *pointed closed convex cone* if and only if $\mathbf{K}$ is a closed subset and the following properties are satisfied:

1) $\mathbf{K} + \mathbf{K} \subseteq \mathbf{K}$;
2) $\lambda \mathbf{K} \subseteq \mathbf{K}$ for all $\lambda \in \mathbf{R}_+$;
3) $\mathbf{K} \cap (-K) = \{0\}$.

Whenever a pointed closed convex cone $\mathbf{K} \subset E$ is defined, we have an ordering on E defined by $x \leq y$ if and only if $y - x \in \mathbf{K}$. By definition the dual of $\mathbf{K}$ is

$$\mathbf{K} = \{y \in E^* : \ \langle x, y\rangle \geq 0 \text{ for all } x \in \mathbf{K}\}.$$

Note that $\mathbf{K}^*$ is also a closed convex cone. We say that the ordered Banach space $(E, \|\cdot\|, \mathbf{K})$ is a *vector lattice* if and only if for every pair (x, y) of elements of E, the supremum $x \wedge y$ and the infimum $x \vee y$ both exist in E. If $(H, \langle \cdot, \cdot \rangle, \mathbf{K})$ is an ordered Hilbert space, we say that the inner-product $\langle \cdot, \cdot \rangle$ is $\mathbf{K}$-*local* if whenever $x \wedge y = 0$ $(x, y \in \mathbf{K})$, we have $\langle x, y \rangle = 0$. If $(H, \langle \cdot, \cdot \rangle)$ is a Hilbert space and $\mathbf{K} \subset H$ is a closed convex cone, we denote the projection onto $\mathbf{K}$ by $P_{\mathbf{K}}$. The projection $P_{\mathbf{K}}$ is defined for every $x \in H$ by $\| x - P_{\mathbf{K}}(x) \| = \min_{y \in \mathbf{K}} \| x - y \|$.

If $E = \mathbf{R}^n$ and $\langle \cdot, \cdot \rangle$ is the Euclidean inner-product, the cone $\mathbf{R}^n_+$ is closed, pointed, self-adjoint (i. e., $(\mathbf{R}^n_+)^* = \mathbf{R}^n_+$) and the inner-product $\langle \cdot, \cdot \rangle$ is $\mathbf{R}^n_+$-local. The ordered space $(\mathbf{R}^n, \langle \cdot, \cdot \rangle, \mathbf{R}^n_+)$ is a vector lattice. Let $\langle E, E^* \rangle$ be a duality of Banach spaces and let $\mathbf{K} \subset E$ be a pointed closed convex cone. Given the mappings $f: E \to E^*$ and $g: E \to E$, consider the following *implicit complementarity problem*:

$$\mathrm{ICP}(f, g, \mathbf{K}) \quad \begin{cases} \text{find} & x_0 \in E \\ \text{s.t} & g(x_0) \in \mathbf{K}, f(x_0) \in \mathbf{K}^*, \ \text{ and} \\ & \langle g(x_0), f(x_0)\rangle = 0. \end{cases}$$

If $g(x) = x$ for all $x \in E$, we obtain the *nonlinear complementarity problem*:

$$\mathrm{NCP}(f, \mathbf{K}) \quad \begin{cases} \text{find} & x_0 \in \mathbf{K} \\ \text{s.t} & f(x_0) \in \mathbf{K}^* \quad \text{and} \\ & \langle x_0, f(x_0)\rangle = 0. \end{cases}$$

If E is a vector lattice with respect to the ordering defined by $\mathbf{K}$ and $f_1, \ldots, f_n$ are mappings from E into E we consider the *general order complementarity problem*: $\mathrm{GOCP}(\{f_i\}_{i=1}^n, \mathbf{K})$

$$\begin{cases} \text{find} & x_0 \in \mathbf{K} \\ \text{s.t} & \wedge(f_1(x_0), \ldots, f_n(x_0)) = 0. \end{cases}$$

Topological Degree and Complementarity Problems

A powerful topological method used in complementarity theory is based on the concept of *topological degree* of a continuous mapping. A standard reference for degree theory is [23]. Corresponding to a bounded open set $\Omega \subset \mathbf{R}^n$, a continuous function $f: \overline{\Omega} \to \mathbf{R}^n$, and an n-vector $y \notin f(\partial\Omega)$, we associate an integer number denoted by $\deg(f, \Omega, y)$. We say that $\deg(f, \Omega, y)$ is the degree of f at y relative to Ω. Always for our applications we take $y = 0$. The topological degreehas the following properties:

1) (Existence property). If $\deg(f, \Omega, 0) \neq 0$, then the equation $f(x) = 0$ has a solution in Ω.
2) (Nearness property). If $\deg(f, \Omega, 0)$ is defined and $g \in C(\overline{\Omega}, \mathbf{R}^n)$ is such that $\sup_{x \in \Omega} \| f(x) - g(x) \| < \operatorname{dist}(0, f(\partial\Omega))$, then $\deg(g, \Omega, 0)$ is defined and $\deg(g, \Omega, 0) = \deg(f, \Omega, 0)$.
3) (Homotopy invariance property). If $H: [0,1] \times \overline{\Omega} \to \mathbf{R}^n$ is continuous and $0 \neq H(t, \partial\Omega)$ for all $t \in [0, 1]$, then $\deg(H(0, \cdot), \Omega, 0) = \deg(H(1, \cdot), \Omega, 0)$.
4) (Excision property). Suppose that $\deg(f, \Omega, 0)$ is defined and D is a compact subset of Ω such that there are no solutions of $f(x) = 0$ in D. Then $\deg(f, \Omega, 0) = \deg(f, \Omega \setminus D, 0)$.
5) (Domain decomposition property). If $\deg(f, \Omega, 0)$ is defined and Ω is a disjoint union of finite number of open sets Ω_l, then
$$\deg(f, \Omega, 0) = \sum_i \deg(f, \Omega_i, 0)$$
6) (Index at a zero). Let x_* be an isolated solution of the equation $f(x) = 0$. Then $\deg(f, \Omega, 0)$ is the same for any bounded open set Ω containing x_* with the property that $\overline{\Omega}$ contains no other solution of $f(x) = 0$. In this case we call $\deg(f, \Omega, 0)$ the index of f at x_*, i. e., $\operatorname{index}(f, x_*) = \deg(f, \Omega, 0)$. If f is differentiable at x_* with a nonsingular Jacobian matrix $f'(x_*)$, then $\operatorname{index}(f, x_*) = \operatorname{sgn} \det f'(x_*)$.

The topological degree defined above is the *Brouwer degree*, which can be extended to infinite-dimensional case by the concept of *Leray–Schauder degree*. It is evident that, when the problem NCP(f, $\mathbf{K}$) is equivalent to an equation of the form $\Phi(x) = 0$, and $\deg(\Phi, \Omega, 0)$ is well defined, we can obtain existence theorems for the problem NCP(f, $\mathbf{K}$). This is the situation when the problem NCP(f, $\mathbf{K}$) is defined on a Hilbert space $(H \langle \cdot, \cdot \rangle)$ ordered by a closed pointed convex cone $\mathbf{K} \subset H$. It is known [13,15,25,26] that in this case, the problem NCP(f, $\mathbf{K}$) is equivalent to the solvability of the equation

$$\Phi_1(x) = x - P_{\mathbf{K}}(x - f(x)) = 0. \tag{1}$$

In [7,25,26] several results were proved using (1) and the topological degree in the Euclidean space. Suppose that the ordered Hilbert space $(H, \langle \cdot, \cdot \rangle, \mathbf{K})$ is a vector lattice, $\mathbf{K}$ is self-adjoint (i. e., $\mathbf{K} = \mathbf{K}^*$) and the inner-product $\langle \cdot, \cdot \rangle$ is $\mathbf{K}$-local. In this case the problem NCP(f, $\mathbf{K}$) is equivalent to the equation:

$$\Phi_2(x) = x \wedge f(x) = 0. \tag{2}$$

This is the case when $(H, \langle \cdot, \cdot \rangle, \mathbf{K})$ is the Euclidean space $\mathbf{R}^n$, $\langle \cdot, \cdot \rangle$, $\mathbf{R}^n_+$). The topological degree of the mapping Φ_2 can be used.

Generally, the problem GOCP($\{f_i\}_{i=1}^n$, $\mathbf{K}$) can be studied using the topological degree and the equation

$$\Phi_3(x) = \wedge(f_1(x), \ldots, f_n(x)) = 0. \tag{3}$$

Using (2) and (3) and the topological degree many results were proved in [6,7,8,25,26].

The particular case of affine functions (i. e., the case of linear complementarity problems) has been considered in many papers as for example: [4,6,7,8,11,12,24, 27,28,29]. The topological degree can be also used to study, the cardinality of solution set [7,21,22], to study the stability of solutions [7,10], or to study the connectedness of solution set [9,18]. Finally, we note the paper [5] where the topological degree is applied to the study of a particular complementarity problem which is important in Elasticity Theory.

Exceptional Families of Elements and Complementarity Problems

Let $(\mathbf{R}^n, \langle \cdot, \cdot \rangle)$ be the Euclidean space, $\mathbf{K} \subset E$ a closed pointed convex cone and f: $\mathbf{R}^n \to \mathbf{R}^n$ a function.

Definition 1 We say that the family of elements $\{x^r\}_{r>0} \subset \mathbf{K}$ is an *exceptional family* of elements for f with respect to $\mathbf{K}$ if and only if for every real number $r > 0$ there exists a real number $\mu_r > 0$ such that the vector $u_r = f(x^r) + \mu_r x^r$ satisfies the following conditions:
1) $u_r \in \mathbf{K}^*$;
2) $\langle u_r, x^r \rangle = 0$;
3) $\| x^r \| \to +\infty$ as $r \to +\infty$.

This was introduced in [1] and [19] and it is a new variant of a similar notion introduced initially in [15]. By the topological degree we can prove the following alternative.

Theorem 2 *For any continuous function f: $\boldsymbol{R}^n \to \boldsymbol{R}^n$ there exists either a solution for the problem NCP(f, $\boldsymbol{K}$), or an exceptional family of elements for f with respect to $\boldsymbol{K}$.*

Proof The proof is in [1,15] or [19].

Corollary 3 *If a continuous function f: $\boldsymbol{R}^n \to \boldsymbol{R}^n$ is without exceptional families of elements with respect to $\boldsymbol{K}$, then the problem NCP(f, $\boldsymbol{K}$) is solvable.*

In the papers [1,15,16,17] and [19] are proved several existence theorems based on Corollary of Theorem 1 for explicit and implicit complementarity problems. We note that Theorem 1 can be extended to infinite-dimensional Hilbert spaces, replacing the function f by a compact field.

Homotopy Continuation Method

Let $(\mathbf{R}^n, \langle \cdot, \cdot \rangle)$ be the Euclidean space ordered by $\mathbf{R}^n_+$, and f: $\mathbf{R}^n \to \mathbf{R}^n$ a continuous function. The homotopy continuation method is the following. Let $D(x) = \mathrm{diag}(x)$ be the diagonal $(n \times n)$-matrix with the coordinates of $x \in \mathbf{R}^n$. Define the mapping $\Phi : \mathbf{R}^{2^n} \to \mathbf{R}^n_+ \times \mathbf{R}^n$ by $\Phi(z) = (D(x)y, y\text{-}f(x))$ for every $z = (x, y) \geq 0$. The problem NCP(f, $\mathbf{R}^n_+$) is equivalent to the system of equations:

$$\Phi(z) = 0 \quad \text{and} \quad z = (x, y) \geq 0. \tag{4}$$

Consider the family of systems of equations

$$\Phi(z) = tc \quad \text{and} \quad z = (x, y) \geq 0. \tag{5}$$

where $c = (a, b) \in (\mathbf{R}^n_+ \setminus 0) \times \mathbf{R}^n$ and $t \in \mathbf{R}_+$. Let $C = \{ tc : t > 0 \}$. Under certain assumptions $\Phi^{-1}(C)$ exists and

forms a trajectory, a one-dimensional curve, $\{z(t):t>0\}$. Furthermore, $z(t)$ leads to a solution of the system (4) as t tends to 0. By the homotopy continuation method we can study the existence of the trajectory $\Phi^{-1}(C)$ and we can develop numerical methods for tracing this trajectory. About the results obtained by this method, the reader is referred to the paper [20] and to its references.

Zero-Epi Mappings and Complementarity Problems

The concept of *zero-epi mapping* was introduced in [3] and it is more simple and more refined as the topological degree [3,13]. By this notion we can obtain the solvability of a nonlinear equation in a Banach space, even when the topological degree is zero. Let $(E, \|\cdot\|)$ and $(F, \|\cdot\|)$ be Banach spaces and $\Omega \subset E$ a bounded open subset.

Definition 4 We say that a continuous mapping $f: \overline{\Omega} \to F$ is *zero-epi* (shortly 0-epi) if and only if:
1) $0 \notin f(\partial\Omega)$ (i. e., f is 0-admissible);
2) for any continuous compact mapping $h: \overline{\Omega} \to F$ such that $h(x) = 0$ for every $x \in \partial\Omega$, the equation $f(x) = h(x)$ has a solution in Ω.

If $f: \overline{\Omega} \to F$ is p-admissible (for $p \in F$), i. e., $p \notin f(\partial\Omega)$ and the mapping $f - p$, defined by $(f - p)(x) = f(x) - p$ is 0-epi, we say that f is *p-epi*.

Properties

i) *Existence property*. If $f: \overline{\Omega} \to F$ is p-epi, then the equation $f(x) = p$ has a solution in Ω.
ii) *Normalization property*. The inclusion $i: \overline{\Omega} \to E$ is p-epi if and only if $p \in \Omega$.
iii) *Localization property*. If $f: \overline{\Omega} \to F$ is 0-epi, $\Omega_1 \subset \Omega$ is an open set and $f^{-1}(0) \subset \Omega_1$, then the restriction of f to Ω_1, i. e., $f|_{\overline{\Omega}_1}: \overline{\Omega}_1 \to F$ is 0-epi.
iv) *Homotopy property*. Let $f: \overline{\Omega} \to F$ be 0-epi and let $h: \overline{\Omega} \times [0, 1] \to F$ be a continuous and compact mapping such that $h(x, 0) = 0$ for any $x \in \overline{\Omega}$. If $f(x) + h(x, t) \neq 0$ for all $x \in \partial\Omega$ and for any $t \in [0, 1]$, then the mapping $f(\cdot) + h(\cdot, 1): \overline{\Omega} \to F$ is 0-epi.
v) *Boundary dependence property*. If $f: \overline{\Omega} \to F$ is 0-epi and $g: \overline{\Omega} \to F$ is a continuous compact mapping such that $g(x) = 0$ for all $x \in \Omega$, then $f + g: \overline{\Omega} \to F$ is 0-epi.

Remark 5 If $f: \overline{\Omega} \to E$ is a p-admissible compact vector field and the Leray–Schauder degree $\deg(f, \Omega, p) \neq 0$, then f is p-epi. The converse is not true.

In [13, Chap. 3] several existence theorems are proved for the (implicit and explicit) nonlinear complementarity problem applying the concept of 0-epi mapping. The connectedness of solution set of a nonlinear complementarity problem depending of multiparameters can be studied also applying the concept of 0-epi mapping [13]. The next result extends to the infinite-dimensional case and to the case when the topological degree is zero, the main result proved in [26].

Theorem 6 *Let $(H, \langle\cdot,\cdot\rangle)$ be a Hilbert space, $\boldsymbol{K} \subset H$ a closed pointed convex cone and $f, g: H \to H$ completely continuous mappings. Suppose given a completely continuous mapping $\phi: H \to H$ and $\Omega \subset H$ a bounded open set such that:*
1) the mapping $\Psi: \overline{\Omega} \to H$ defined by $\Psi(x) = g(x) - P_{\boldsymbol{K}}[g(x) - \phi(x)]$ for all $x \in \overline{\Omega}$ is 0-epi;
2) for every $\mu > 0$ and $x \in \partial\Omega \cap g^{-1}(\boldsymbol{K})$ we have $f(x) + \mu\,\phi(x) \notin (\boldsymbol{K} - g(x))^*$.
Then the problem ICP(f, g, $\boldsymbol{K}$) has a solution $x_* \in \overline{\Omega}$.

Proof A proof of this result is in [13].

Conclusions

The application of topological methods to the study of complementarity problems represents probably, the most recent activity in complementarity theory. Another argument to support this idea is the topological index on cones used recently in the paper [14].

See also

- Convex-simplex Algorithm
- Equivalence Between Nonlinear Complementarity Problem and Fixed Point Problem
- Generalized Nonlinear Complementarity Problem
- Integer Linear Complementary Problem
- LCP: Pardalos–Rosen Mixed Integer Formulation
- Lemke Method
- Linear Complementarity Problem
- Linear Programming
- Order Complementarity
- Parametric Linear Programming: Cost Simplex Algorithm

► Principal Pivoting Methods for Linear Complementarity Problems
► Sequential Simplex Method

References

1. Bulavski VA, Isac G, Kalashnikov V (1998) Application of topological degree to complementarity problems. In: Migdalas A (eds) Multilevel Optimization: Algorithms and Applications. Kluwer, Dordrecht, pp 333–358
2. Carbone A, Isac G (1998) The generalized order complementarity problem. Applications to economics. An existence result. Nonlinear Stud 5(2):129–151
3. Furi M, Martelli M, Vignoli A (1980) On the solvability of nonlinear operators equations in normed spaces. Ann Mat Pura Appl 124:321–343
4. Garcia CB, Gould FJ, Turnbull TR (1983) Relations between PL maps, complementarity cones and degree in linear complementarity problems. In: Eaves, Gould, Peitgen, Todd (eds) Homotopy Methods and Global Convergence. Plenum, New York, pp 91–144
5. Goeleven D, Nguyen VH, Théra M (1993) Nonlinear eigenvalue problems governed by a variational inequality of von Karman's type: A degree theoretic approach. Topol Meth Nonlinear Anal 2:253–276
6. Gowda MS (1991) A degree formula of Stewart. Math Res Report Univ Maryland
7. Gowda MS (1993) Applications of degree theory to linear complementarity problems. Math Oper Res 18(4):868–879
8. Gowda MS, Sznajder R (1994) The generalized order linear complementarity problem. SIAM J Matrix Anal Appl 15(3):779–795
9. Gowda MS, Sznajder R (1997) Weak univalence and connectedness of inverse images of continuous functions. Preprint Univ Maryland, Baltimore County
10. Ha CD (1987) Application of degree theory in stability of the complementarity problem. Math Oper Res 12:368–376
11. How R (1983) On a class of linear complementarity problem of variable degree. In: Eaves, Gould, Peitgen, Todd (eds) Homotopy Methods and Global Convergence. Plenum, New York, pp 155–177
12. How R, Stone RE (1983) Linear complementarity and the degree of mappings. In: Eaves, Gould, Peitgen, Todd (eds) Homotopy Methods and Global Convergence. Plenum, New York, pp 179–223
13. Hyers DH, Isac G, Rassias TM (1997) Topics in nonlinear analysis and applications. World Sci., Singapore
14. Isac G (1996) The fold complementarity problem and the order complementarity problem. Topol Meth Nonlinear Anal 8:343–358
15. Isac G, Bulavski V, Kalashnikov V (1997) Exceptional families, topological degree and complementarity problems. J Global Optim 10:207–225
16. Isac G, Carbone A (1999) Exceptional families of elements for continuous functions. Some applications to complementarity theory. J Global Optim 15:181–196
17. Isac G, Obushowska T (1998) Functions without exceptional family of elements and complementarity problems. J Optim Th Appl 99:147–163
18. Jones C, Gowda MS (1997) On the connectedness of solution set in linear complementarity problems. Preprint Univ Maryland, Baltimore County
19. Kalashnikov V (1995) Complementarity problem and the generalized oligopoly model. Habilitation Thesis CEMI, Moscow
20. Kojima M, Megiddo N, Noma T (1991) Homotopy continuation method for nonlinear complementarity problems. Math Oper Res 16(4):754–774
21. Kojima M, Saigal R (1979) On the number of solutions to a class of linear complementarity problem. Math Program 17:136–139
22. Kojima M, Saigal R (1981) On the number of solutions to a class of complementarity problems. Math Program 21:190–203
23. Lloyd NG (1978) Degree theory. Cambridge Univ. Press, Cambridge
24. Morris WD (1990) On the maximum degree of an LCP map. Math Oper Res 15:423–429
25. Pang JS (1995) Complementarity problems. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 271–338
26. Pang JS, Yao JC (1995) On a generalization of a normal map and equation. SIAM J Control Optim 33(1):168–184
27. Sridhar R (1996) The degree on an exact order matrix. Math Oper Res 21(2):427–441
28. Stewart DE (1991) A degree theory approach to degeneracy of LCPS. Res Report Dept Math Univ Queensland, Australia 4072
29. Sznajder R (1994) Degree-theoretic analysis of the vertical and horizontal linear complementarity problems. PhD Thesis, Graduate School Univ. Maryland

Topology of Global Optimization

HUBERTUS TH. JONGEN[1], ALINA RUIZ JHONES[2]
[1] Department Math., Aachen University Technol., Aachen, Germany
[2] Fac. Math. and Computer Sci., University Havana San Lázaro y L, Ciudad Habana, Cuba

MSC2000: 90C30, 58E05

Article Outline

Keywords
Introduction, Critical Points, Nondegeneracy
Relations Between KKT Points: Morse Relations
Projected Gradients
Global Gradient Flows: Equality Con-straints Only
Global Gradient Flows: The General Case
See also
References

Keywords

Morse theory; Karush–Kuhn–Tucker point; Morse relations; Euler formula; Global optimization; Continuous selection of functions; Min-max graph; Min-max digraph; Projected positive gradient; Projected negative gradient; Ascent flow; Descent flow

Introduction, Critical Points, Nondegeneracy

In this article we describe the basic idea of Morse theory in finite-dimensional smooth optimization. This is concerned with critical points (in particular, Karush–Kuhn–Tucker points) and relations between them. An extension to certain nonsmooth problems is indicated. Then, we turn to gradient flows and focus on the fundamental problem: how to get from one local minimum to (all) other ones.

In this paper we consider optimization problems of the type (P):

$$(P)\quad \begin{cases} \min & f \text{ on the feasible set } M, \\ \text{where} & M = \left\{ x \in \mathbf{R}^n : \begin{array}{l} h_i(x) = 0, \\ i \in I, \\ g_j(x) \geq 0, \\ j \in J \end{array} \right\} \end{cases}$$

and where $f, h_i, g_j \colon \mathbf{R}^n \to \mathbf{R}$ are C^2-functions, $|I| \to < n$, $|J| < \infty$.

For simplicity we assume that M is compact and that the linear independence constraint qualification (LICQ) is satisfied at all points of M. The LICQ is said to hold at $\overline{x} \in M$ if the vectors $Dh_i(\overline{x})$, $i \in I$, $Dg_j(\overline{x})$, $j \in J_0(\overline{x})$ are linearly independent. Here, Dh stands for the row vector of partial derivatives of h and $J_0(\overline{x}) = \{ j \in J \colon g_j(\overline{x}) = 0 \}$.

In virtue of LICQ we can take the constraint functions h_i, $i \in I$, g_j, $j \in J_0(\overline{x})$, as new coordinates in a neighborhood of $\overline{x}$. In these coordinates, the set M locally takes the form $\mathbf{H}^p \times \mathbf{R}^q$, where $p = |J_0(\overline{x})|$, $q = n - |I| - p$, and $\mathbf{H}^p = \{ y \in \mathbf{R}^p \colon y_i \geq 0, \quad i = 1, \ldots, p \}$

A point $\overline{x} \in M$ is called a *critical point* for $f \mid_M$ if there exist real numbers $overline\lambda_i, \overline{\mu}_j$, such that

$$Df = \sum_{i \in I} \overline{\lambda}_i Dh_i + \sum_{j \in J_0(\overline{x})} \overline{\mu}_j Dg_j|_{\overline{x}}.$$

A critical point is called a *Karush–Kuhn–Tucker point* (shortly, *KKT point*) if $\overline{\mu}_j \geq 0$, $j \in J_0(\overline{x})$. Moreover, a critical point is said to be *nondegenerate* if the following two conditions hold:

ND1) (*linear*) $\overline{\mu}_j \neq 0$, $j \in J_0(\overline{x})$.

ND2) (*quadratic*) $D^2 L(\overline{x})|_{T_{\overline{x}} M}$ is nonsingular.

The matrix $D^2 L$ stands for the Hessian of the Lagrange function L,

$$L(x) := f(x) - \sum_{i \in I} \overline{\lambda}_i h_i(x) - \sum_{j \in J_0(\overline{x})} \overline{\mu}_j g_j(x),$$

and $T_{\overline{x}} M$ denotes the tangent space at $\overline{x}$,

$$T_{\overline{x}} M = \{ \xi \in \mathbf{R}^n \colon Dh_i(\overline{x})\xi = 0,\ i \in I,$$
$$Dg_j(\overline{x})\xi = 0,\ j \in J_0(\overline{x}) \}.$$

Condition ND2) means that the matrix $V^\top D^2 L(\overline{x}) V V$ is nonsingular, where V is some matrix whose columns form a basis for the tangent space $T_{\overline{x}} M$.

In a neighborhood of a nondegenerate critical point $\overline{x}$ there exist new C^1-coordinates, such that M locally takes the form $\mathbf{H}^p \times \mathbf{R}^q$ and $f \mid_M$ becomes (*equivariant Morse lemma*; see [11]):

$$f \sim f(\overline{x}) + \sum_i -y_i + \sum_j y_j + \sum_k -y_k^2 + \sum_l y_l^2$$

where the coordinates y_i and y_j in the first two sums are nonnegative. The number of negative/positive linear terms corresponds to the number of negative/positive multipliers $\overline{\mu}_r$, $r \in J_0(\overline{x})$, whereas the number of negative/positive squares is equal to the number of negative/positive eigenvalues of $V^\top D^2 L(\overline{x}) V$). The number of negative linear (quadratic) terms is called the *linear index LI* (*quadratic index QI*). In particular, a nondegenerate critical point is a KKT point (local minimum) if and only if $LI = 0$ ($LI = QI = 0$). Basic references for this article are [11,12,15].

Relations Between KKT Points: Morse Relations

From now on we assume that all critical points of $f|_M$ are nondegenerate with pairwise different f-values. For details we refer to [11,16]. In order to study relations between critical points, we consider lower level sets $M^a = \{x \in M : f(x) \le a\}$ for increasing values of a. If the intermediate set $M_a^b := \{x \in M: a \le f(x) \le b\}$ does not contain a KKT point, then the set M^b can be continuously deformed into the lower level set M^a. The crucial point is the following: a point $\overline{x} \in M$ M is not a KKT point if and only if there exists a feasible direction of linear descent for f at $\overline{x}$. The desired deformation can now be accomplished via such descent directions.

Next, suppose that the intermediate set M_a^b contains exactly one KKT point $\overline{x}$. Moreover, let $a < f(\overline{x}) < b$ and $QI = k$. Then, the lower level set M^b has the homotopy type of $M^a \cup D^k$. Here, the notation $M^a \cup D^k$ means that a k-dimensional ball D^k is attached (glued) to the set M^a along its boundary ∂D^k.

In particular, if $k = 0$ (local minimum!), then $M^a \cup D^0$ is just the disjoint union of M^a and a point (hence, a new component is created). Next, the 1-dimensional ball D^1 is an interval, and its boundary ∂D^1 consists of two points. There are two possibilities. Either the two boundary points are glued onto two different components of M^a (hence, the number of connected components decreases by one), or both boundary points are mapped onto the same component of M^a (now, the number of 2-dimensional 'holes' is increased by one). Speaking in terms of holes, we have the following general alternative when passing a value of a KKT point with $QI = k$:

- either the number of k-dim holes of M^a goes down by one; or
- the number of $(k+1)$-dim holes of M^a goes up by one.

To be precise: by a k-dim hole of a topological space X we mean a generator of $H_{k-1}(X)$, the $(k-1)$ singular homology space of X over the real number field; in particular, $H_0(X)$ counts the number of path-connected components of X.

The number of k-dim holes is invariant under continuous deformations. Hence, that number can only change when passing a functional level corresponding to a KKT point.

Let r_k (the kth Betti number) denote the number of the $(k+1)$-dim holes of the feasible set M. Moreover, let c_k^- (c_k^+) be the number of KKT points with $QI = k$ at which level the number of k-dim holes of M^a goes down (the number of $(k+1)$-dim holes goes up) for increasing values of a. If we reach the global maximum value of $f|_M$, then $M^a = M$, and we should have created precisely r_k holes of dimension $(k+1)$, $k = 0, 1, 2, \cdots$. Consequently, if (in between) more than r_k holes of dimension $(k+1)$ are created, then some of these holes should be closed before reaching the global maximum value of $f|_M$. Together with the aforementioned alternative, this results into the following topological balance equations (*Morse relations*):

$$c_i^+ - c_{i+1}^- = r_i, \quad i = 0, 1, \ldots, \tag{1}$$

where $c_0^+ = c_0$ and $0 = c_s := c_s^+ + c_s^-$ for $s > n - |I|$ (the dimension of M).

If M is connected, then $r_0 = 1$ and the first relation in (1) becomes $c_0 - c_1^{-=1}$.

It guarantees the existence of at least $(c_0 - 1)$ KKT points with $QI = 1$. (*mountain pass theorem*). For this reason we call the KKT points with $QI = 1$ of $(-)$ type *decomposition points*. In fact, when lowering the functional level of a decomposition point, the corresponding component of the lower level set splits up into two components, thereby separating the local minima contained in them.

We can get rid of the $(+)$ and $(-)$ signs in (1) by adding all equations in (1) with alternating signs. This leads to the equation ($s = n - |I|$):

$$c_0 - c_1 + c_2 - \cdots + (-1)^s c_s = r_0 - r_1 + r_2 - \cdots + (-1)^s r_s \tag{2}$$

Remark 1 In the deformation part (along feasible directions of linear descent) we can weaken the LICQ assumption. For example, the Mangasarian–Fromowitz constraint qualification suffices (see [7]). Also, nonsmooth aspects can be taken into account (see [4,9,13]). Since only KKT points play a role in the Morse relations (1), the nondegenerate KKT points can be replaced by strongly stable stationary points (in the sense of Kojima); see[7].

Remark 2 In case that M is a polytope P, relation (2) reflects the famous Euler 's formula. In fact, using the

logarithmic barrier function, a function f can be constructed such that each k-dim face of the polytope P has exactly one KKT point for $f\,|_P$ with $QI = k$. For this particular function f, the number c_k in (2) equals the number of k-dim faces of P. Since a polytope can be continuously deformed into a point, we have $r_0 = 1$, and $r_i = 0, i \geq 1$. Altogether, formula (2) then becomes the Euler 's formula for polytopes (see [8]).

Remark 3 The ideas of deformation and cell attachment can be generalized to functions of maximum type (cf. [2]) and minimum type (cf. [3]). Both cases are special continuous selections of functions. In fact, let $\Psi = CS(f_1, \ldots, f_s)$, where $f_1, \ldots, f_s \colon \mathbf{R}^n \to \mathbf{R}$ are C^2-functions and CS means 'continuous selection'. Note that Ψ is nonsmooth in general. In [13] the concept of nondegenerate critical point for Ψ is introduced. It is shown that, locally around such a point $\overline{z}$, there exist new *continuous* coordinates such that Ψ takes the form:

$$\Psi \sim \Psi(\overline{z}) + CS\left(x_1, \ldots, x_r, -\sum_{i=1}^{r} x_i\right) - \sum_{j=r+1}^{r+k} x_j^2 + \sum_{l=r+k+1}^{n} x_l^2. \tag{3}$$

It is easily seen that

$$\max\left(x_1, \ldots, x_r, -\sum_{i=1}^{r} x_i\right) \sim \sum_{i=1}^{r} x_i^2$$

and

$$\min\left(x_1, \ldots, x_r, -\sum_{i=1}^{r} x_i\right) \sim -\sum_{i=1}^{r} x_i^2.$$

Now, consider the lower level set of Ψ when passing the value $\Psi(\overline{z})$. Then, in case that Ψ is of max (resp. min) type, a k-cell (resp. $(k + r)$-cell) will be attached to the lower level set.

If Ψ is not of max (or min) type, the situation becomes more complicated; with respect to the 'linear part' it is to be expected that more cells have to be attached simultaneously. The negative squares in (3) will raise the dimension of the latter cells. A precise study is presented in [1].

Projected Gradients

A symmetric positive definite (n, n)-matrix R defines a scalar product $\langle \cdot \rangle_R$, where $\langle x, y \rangle_R := x^\top R y$. The *gradient* $\operatorname{grad}_R f(x)$ of f with respect to R is defined to be the vector solving the system $\langle v, \operatorname{grad}_R f(x)\rangle_R = Df(x)v$, $v \in \mathbf{R}^n$. It follows that $\operatorname{grad}_R f(x) = R^{-1}\, D^\top f(x)$.

For $\overline{x} \in M$ let

$$C_{\overline{x}}M := \left\{ \xi \in \mathbf{R}^n : \begin{array}{ll} Dh_i(\overline{x})\xi = 0, & i \in I, \\ Dg_j(\overline{x})\xi \geq 0, & j \in J_0(\overline{x}) \end{array} \right\}$$

denote the tangent cone of M at $\overline{x}$. The *projected positive gradient* $(+)grad_{R,M} f(\overline{x})$ at $\overline{x} \in M$ is defined to be the unique solution vector of the following 'primal' optimization problem:

$$\begin{cases} \min & \left\| \xi - \operatorname{grad}_R f(\overline{x}) \right\|_R, \\ \text{s.t.} & \xi \in C_{\overline{x}}M, \end{cases}$$

where $\|y\|_R = \sqrt{\langle y, y\rangle_R}$.

We point out that $(+)\operatorname{grad}_{R,M} f(\overline{x})$ is equal to the vector obtained by inserting the solution $(\overline{\lambda}, \overline{\mu})$ of the 'dual' problem:

$$\begin{cases} \min & \left\| \operatorname{grad}_R \left(f + \sum_{i \in I} \lambda_i h_i + \sum_{j \in J_0(\overline{x})} \mu_j g_j \right)(\overline{x}) \right\|_R \\ \text{s.t.} & \mu \geq 0. \end{cases}$$

In case that $J_0(\overline{x}) = \emptyset$, we have the formula

$$(+)\operatorname{grad}_{R,M} f(\overline{x}) = (A - AH(H^\top AH)^{-1}H^\top A)D^\top f(\overline{x}),$$

where the columns of H are formed by the vectors $D^\top h_i(\overline{x})$, $i \in I$, and $A = R^{-1}$.

The *projected negative gradient* $(-)\operatorname{grad}_{R,M} f$ is defined to be the projected positive gradient corresponding to the function $(-f)$.

We note that $(-)\operatorname{grad}_{R,M} f(\overline{x}) = 0$ if and only if $\overline{x}$ is a KKT point for $f|_M$. Moreover,

$$(+)\operatorname{grad}_{R,M} f(\overline{x}) = -(-)\operatorname{grad}_{R,M} f(\overline{x})$$

if $J_0(\overline{x}) = \emptyset$.

A C^k-*Riemannian metric* (or *variable metric*)$R{:}x \to R(x)$ is a C^k-mapping from $\mathbf{R}^n$ into the space of symmetric positive definite (n, n)-matrices. It induces (pointwise) a projected positive (negative) gradient field of f on M.

Global Gradient Flows: Equality Con-straints Only

Here we assume that there are no inequality constraints, i. e. $J = \emptyset$.

Let all functions f, h_i, $i \in I$, be smooth, i. e. of class C^∞ and let R be a smooth Riemannian metric. Now, M is a smooth manifold without boundary and the concepts of critical point and KKT point coincide. We assume, in addition, that all critical points of $f\,|_M$ are nondegenerate having pairwise different f-values. Consider the vector field $(+)\,\mathrm{grad}_{R,M} f$ on M. It defines a smooth flow [12] Ψ: $\mathbf{R} \times M \to M$, where $\Psi(t, x)$ is the point which is reached from x when integrating the vector field during the time t. Let $\overline{x} \in M$ be a critical point. Then, the stable manifold $W^s_{\overline{x}} := \{x \in M\colon \lim_{t\to\infty} \Psi(t, x) = \overline{x}\}$ and the unstable manifold $W^u_{\overline{x}} := \{x \in M\colon \lim_{t\to-\infty} \Psi(t, x) = \overline{x}\}$ are well defined. From the fundamental work of S. Smale [17] we know that for generic Riemannian metrics (resp. for generic f) all stable and unstable manifolds corresponding to critical points intersect transversally.

Now we focus on the fundamental question: how to get from one local minimum to (all) other ones. To this aim we introduce two bipartite graphs:

- The 0–1–0 *graph*. The set of nodes is partitioned into the set of local minima of $f\,|_M$ and the set of critical points of $f\,|_M$ with $QI = 1$. There exists an edge between $\overline{x}$ (local minimum) and $\overline{y}$ (critical point with $QI = 1$) if and only if $W^u_{\overline{x}} \cap W^s_{\overline{y}} \neq \emptyset$ (i. e. if there exists a trajectory of $(+)\,\mathrm{grad}_{R,M} f$ which connects the local minimum $\overline{x}$ and the critical point $\overline{y}$).
- The *min-max graph*. The set of nodes is partitioned into the set of local minima and the set of local maxima of $f|_M$. There exists an edge between $\overline{x}$ (local minimum) and $\overline{y}$ (local maximum) iff $W^u_{\overline{x}} \cap W^s_{\overline{y}} \neq \emptyset$ (i. e. if there exists a trajectory of $(+)\,\mathrm{grad}_{R,M} f$ connecting the local minimum $\overline{x}$ and the local maximum $\overline{y}$).

Theorem 4 ([10,11]) *Let M be connected. Then both the 0–1–0 graph and the min-max graph are generically connected.*

The connectedness of the 0–1–0 graph follows from the fact that the subgraph of local minima and *decomposition points* is already (generically) connected. This also induces the connectedness of the min-max graph. In fact, let the decomposition point $\overline{x}$ connect the different local minima $\overline{y}_1$ and $\overline{y}_2$. The unstable manifold $W^u_{\overline{x}}$ (generically) intersects $W^s_{\overline{z}}$ for some local maximum $\overline{z}$. But then, $W^s_{\overline{z}} \cap W^u_{\overline{y}_i} \neq \emptyset$, $i = 1, 2$.

We emphasize that the connectedness of the aforementioned graphs lies at the heart of the problem of global optimization.

Global Gradient Flows: The General Case

The appearance of inequality constraints makes things much more difficult and, up to now, the theory on global flows is far from complete. Now we are dealing with two types of differential equations on M:

$$\dot{x} = (+)\,\mathrm{grad}_{R,M} f(x) \quad \text{(the ascent flow)} \tag{4}$$

$$\dot{x} = (-)\,\mathrm{grad}_{R,M} f(x) \quad \text{(the descent flow)} \tag{5}$$

Both equations (4), (5), may have discontinuities in the right-hand side along the boundary ∂M. A solution of (4), (5) is a function $x(\cdot)$ which is absolutely continuous on compact time intervals and which satisfies (4), (5) almost everywhere. Uniqueness for the associated initial value problems can only be guaranteed for *positive* time intervals (for details see [5,6]). Hence, we will integrate (4), (5) only in positive time, and then, the functional value will increase (decrease). We note that KKT points on $\partial\, M$ may be reached (via the descent flow) in finite time and that integral curves can be tangent to the boundary ∂M. Moreover, an integral curve may move along the boundary ∂M, thereby changing the active constraints.

Now, let us focus on the concept of a min-max graph. We assume again that all critical points of $f|_M$ are nondegenerated. Let $\overline{x}_1, \ldots, \overline{x}_p$ and $\overline{y}_1, \ldots, \overline{y}_q$ be the local minima and the local maxima of $f\,|_M$ respectively. Choose small neighborhoods (germs) $U_{\overline{x}_1}, \ldots, U_{\overline{y}_q}$, of $\overline{x}_1, \ldots, \overline{y}_q$ in M. These neighborhoods will be kept fixed in the sequel. The min-max digraph is defined to be the following directed bipartite graph:

- The *min-max digraph*. The set of nodes is partitioned into the set of local minima $\{\overline{x}_1, \ldots, \overline{x}_p\}$ and the set of local maxima $\{\overline{y}_1, \ldots, \overline{y}_q\}$. There exists an arc from $\overline{x}_i$ to $\overline{y}_j$ (from $\overline{y}_j$ to $\overline{x}_i$) if the ascent flow (descent flow) connects some point from $U_{\overline{x}_i}$ ($U_{\overline{y}_j}$) with a point from $U_{\overline{y}_j}$ ($U_{\overline{x}_i}$).
  Note: In case of equality constraints only, an arc from $\overline{x}_i$ to $\overline{y}_j$ always generates an arc from $\overline{y}_j$ to $\overline{x}_i$ (just by reversing the integration time).

Now, let M be connected. In contrast to the theorem on min-max graphs, the min-max digraph need not be strongly connected (i. e. connected as a directed graph) in presence of inequality constraints. Moreover, the disconnectedness may be stable (with respect to small C^1 −perturbations of Df or R).

The simplest example of this phenomenon can be constructed on the 2-dimensional disc M [18]; the function $f|_M$ should have five critical points: two local minima, two local maxima (all of them on the boundary ∂M) and one saddlepoint (in the interior of M). Moreover, the separatrices of the saddlepoint should intersect ∂M in points outside the chosen neighborhhoods of the local minima (maxima).

Although this result seems to be disappointing at first glance, a different Riemannian metric may be constructed such that the associated min-max digraph becomes strongly connected.

In fact, consider the example above. By means of adapting the Riemannian metric, one might move the four points of intersection of the saddle-separatrices with the boundary ∂M towards the set of local minima/maxima. But then, the associated min-max digraph becomes strongly connected.

We end up with the following theorem [14].

Theorem 5 *For connected M (and given f) there exists a smooth Riemannian metric R such that the resulting min-max digraph is strongly connected.*

See also

- ▶ αBB Algorithm
- ▶ Continuous Global Optimization: Applications
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Differential Equations and Global Optimization
- ▶ DIRECT Global Optimization Algorithm
- ▶ Globally Convergent Homotopy Methods
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Binary Star Astronomy
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization Using Space Filling
- ▶ Parametric Optimization: Embeddings, Path Following and Singularities
- ▶ Semidefinite Programming and Structural Optimization
- ▶ Structural Optimization
- ▶ Structural Optimization: History
- ▶ Topology Optimization

References

1. Agrachev AA, Pallaschke D, Scholtes S (1997) On Morse theory for piecewise smooth functions. J Dynamical and Control Systems 3:449–469
2. Brink-Spalink J, Jongen HTh (1979) Morse theory for optimization problems with functions of maximum type. Methods Oper Res 31:121–134
3. Craven BD, Gershkovich V, Ralph D (1994) Morse theory and relations between smooth and nonsmooth optimization. In: Glover BM, Jeyakumar V, Anderson GT (eds) Proc. Optimization Miniconference, Univ. Ballarat, pp 39–46
4. Degiovanni M (1990) Homotopical properties of a class of nonsmooth functions. Ann Mat Pura Appl (IV) CLVI:37–71
5. Dupuis P, Nagurney A (1993) Dynamical systems and variational inequalities. Ann Oper Res 44:9–42
6. Filippov AF (1988) Differential equations with discontinuous righthand sides. Kluwer, Dordrecht
7. Guddat J, Jongen HTh, Rückmann J-J (1986) On stability and stationary points in nonlinear optimization. J Austral Math Soc (Ser B) 28:36–56
8. Hirabayashi R, Jongen HTh, Shida M (1994) Euler's formula via potential functions. J Oper Res Japan 37(3):228–231
9. Ioffe A, Schwartzman E (1996) Metric critical point theory 1. Morse regularity and homotopic stability of a minimum. J Math Pures Appl 75:125–153
10. JongenHTh (1977) Zur Geometrie endlichdimensionaler nichtkonvexer Optimierungsaufgaben. Internat Ser Numer Math 36:111–136
11. Jongen HTh, Jonker P, Twilt F (1983) Nonlinear optimization in $\mathbf{R}^n$, I. Morse theory, Chebyshev approximation. P. Lang, Frankfurt am Main
12. JongenHTh, Jonker P, Twilt F (1986) Nonlinear optimization in $\mathbf{R}^n$, II. transversality, flows, parametric aspects. P. Lang, Frankfurt am Main
13. JongenHTh, Pallaschke D (1988) On linearization and continuous selections of functions. Optim 19:343–353
14. Jongen HTh, Ruiz Jhones A (2000) Nonlinear optimization: On the min-max digraph and global smoothing. In: Ioffe A, Reich S, Shafrir I (eds) Calculus of Variations and Differential Equations, vol 410. Res Notes in Math. Chapman and Hall/CRC, London/Boca Raton, FL, pp 119–135
15. Jongen HTh, Weber G-W (1992) Nonconvex optimization and its structural frontiers. In: Krabs W, Zowe J (eds) Modern Methods of Optimization, vol 378. Lecture Notes Economics and Math Systems, pp 151–203
16. Milnor J (1963) Morse theory. Ann of Math Stud, vol 51. Princeton Univ. Press, Princeton
17. Smale S (1961) On gradient dynamical systems. A-MATH 74(1):199–206
18. Zank H (1994) Personal communication

Topology Optimization

MARTIN P. BENDSØE
Department of Mathematics,
Technical University Denmark, Lyngby, Denmark

MSC2000: 90C90, 90C99

Article Outline

Keywords
See also
References

Keywords

Optimization; Topology; Structures; Relaxation

The term "topology optimization" is usually used for a certain type of problem appearing in structural optimization (► structural optimization; ► structural optimization: history) where the choice of design variables allows for a prediction of a general distribution of material in space. Alternatively, this type of problem is called generalized shape design problems or layout design problems. The concepts of the area are not, however, restricted to problems in structural mechanics, and much of the basic understanding of the variational nature of these problems in a continuum setting derives from studies of conduction problems (heat or electric current). Also, much current research is related to the application of the techniques in multiphysics settings [4,5].

Inherently topology design problems are large-scale (infinite) discrete optimization problems. Most work in the field has been concerned with formulations where the prediction of topology can be performed in the framework of differentiable optimization. Initial studies were performed in the early twentieth century for problems where layout is described in terms of densities of fields of stringers at the plastic limit, with variational calculus being the setting of the mathematical analysis [9]. Works involving the tools of mathematical programming techniques were initiated in the 1960s, based on similar mechanical models for truss structures, leading to linear programming problems, and with the fundamental solutions being the basis for obtaining considerable insight into the mechanical nature of optimal topologies [8].

In the last two decades (as of 2007), work in the area of shape design in a variational setting has led to a revival of topology design and it is now one of the most active areas of design optimization. As it is broadly recognized that structural layout has an immense influence on structural performance, the technology is now quite standard in industrial contexts, typically based on the standard use of finite-element software. In discrete form the problems treated are similar in structure to other structural optimization problems, i. e., with objective and constraint functions given in terms of the design variables and correlated state variables, which in turn are given implicitly as solutions to variational problems depending on the design variables (that is, the problem is a mathematical programming problem with equilibrium constraints, and in a continuum mechanics formulation it is a so-called partial differential equation constrained optimization problem). In topology design the number of design variables is large, usually leading to simplifications made in terms of the number of constraints and in the complications involved in the variational problem defining the state. In mathematical programming terms, sequential convex approximations and dual methods play an important role [10], while for problems of special structure (see later), interior point methods for semidefinite programming problems and methods of nonsmooth optimization have led to efficient computational procedures [3].

In its general continuum setting the mathematical analysis of topology design problems is based on the tools of variational analysis, as seen in problems of optimal control of partial differential equations. Methods of variational convergence (G-convergence, Γ-convergence, etc.) and relaxation are central to the area, and as relaxed controls can be understood in terms of composite materials a close interaction between the area and the field of theoretical material science has been fruitful and of mutual benefit [2,7].

Seen from a mathematical programming perspective, the most thoroughly studied topology design problem is the so-called truss topology problem. Here the optimization of the geometry and topology of trusses can conveniently be formulated in terms of the well-known ground structure method. In this approach the

layout of the truss structure is found by allowing a certain set of connections between a fixed set of nodal points as potential structural or vanishing members (bars in tension or compression only). Allowing for continuously varying cross-sectional bar areas, including the possibility of zero bar areas, gives a continuous optimization problem which permits the prediction of topology; that is, the prediction of which bars should be part of an optimal structure. A similar problem structure can be achieved in some finite-element versions of continuum problems, such as variable thickness membrane problems and problems involving the prediction of topology as well as material. If we consider the simplest possible optimal design problem, namely, the minimization of compliance (maximization of stiffness) for a given total mass of the structure, the topology optimization problem has a linear objective function in the vector of nodal displacements $\mathbf{u}$ and a bilinear constraint equation determining the displacements as functions of the design variables t_i, resulting in the following problem statement:

$$\begin{aligned} &\min_{t\in\boldsymbol{R},\boldsymbol{u}\in\boldsymbol{R}} \boldsymbol{f}^T\boldsymbol{u} \\ &\text{subject to} \sum_{i=1}^{m} t_i\ \boldsymbol{K}_i\boldsymbol{u} = \boldsymbol{f} \\ &\qquad\qquad \boldsymbol{t} \geq \boldsymbol{0} \\ &\qquad\qquad \sum_{i=1}^{m} t_i \leq V\,; \end{aligned} \tag{1}$$

where $\boldsymbol{K}_\mathrm{i}$ is the positive semidefinite element stiffness matrix of the ith element, and the vector $\boldsymbol{f}$ denotes the given external loads. It is for this problem possible, through duality principles, to derive a number of equivalent problem statements, for example, in semidefinite programming form. With these formulations at hand it is thus possible to devise algorithms which can handle large-scale problems, and it is possible to devise algorithms for finding global optima [1].

A survey of the area can be found in [6]; this reference also includes an extensive bibliography.

See also

- ► Semidefinite Programming and Structural Optimization
- ► Structural Optimization
- ► Structural Optimization: History
- ► Topology of Global Optimization

References

1. Achtziger W, Stolpe M (2007) Truss topology optimization with discrete design variables – Guaranteed global optimality and benchmark examples. Struct Multidiscip Optim 34:1–20
2. Allaire G (2002) Shape Optimization by the Homogenization Method. Springer, New York
3. Ben-Tal A, Kocvara M, Nemirovski A, Zowe J (2000) Free material design via semidefinite programming: the multiload case with contact conditions. SIAM Rev 42:695–715
4. Bendsoe MP (2006) Computational Challenges for Multi-Physics Topology Optimization. In: Mota Soares CA et al (eds) Computational Mechanics — Solids, Structures and Coupled Problems. Springer, Dordrecht, pp 1–20
5. Bendsoe MP, Lund E, Olhoff N, Sigmund O (2005) Topology Optimization – broadening the areas of application. Control Cybern 34:7–35
6. Bendsøe MP, Sigmund O (2003) Topology Optimization – Theory, Methods and Applications. Springer, Heidelberg
7. Cherkaev AV (2000) Variational Methods for Structural Optimization. Springer, New York
8. Dorn W, Gomory R, Greenberg M (1964) Automatic design of optimal structures. J Mécanique 3:25–52
9. Hemp WS (1973) Optimum structures. Clarendon Press, Oxford
10. Svanberg KA (2002) Class of Globally Convergent Optimization Methods Based on Conservative Convex Separable Approximations. SIAM J Optim 12:555–573

Traffic Network Equilibrium

TNE

Anna Nagurney
University Massachusetts, Amherst, USA

MSC2000: 90B06, 90B20, 91B50

Article Outline

Keywords
Traffic Network Equilibrium
with Travel Disutility Functions
Elastic Demand Traffic Network Problems
with Known Travel Demand Functions
Fixed Demand Traffic Network Problems
See also
References

Keywords

User-optimization; System-optimization; Traffic assignment; Variational inequality formulations; Equilibration; Congested network; Braess paradox

The traffic network equilibrium problem, sometimes also referred to as the *traffic assignment* problem, addresses the problem of users of a congested transportation network seeking to determine their minimal cost travel paths from their origins to their respective destinations. It is a classical *network equilibrium* problem and was studied by A.C. Pigou [29], who considered a two-node, two-link (or path) transportation network, and was further developed by F.H. Knight [21]. The congestion on a link is modeled by having the travel cost as perceived by the user be a nonlinear function; in many applications the cost is convex or monotone.

The main objective in the study of traffic network equilibria is the determination of traffic patterns characterized by the property that, once, established, no user or potential user may decrease his travel cost or disutility by changing his travel arrangements. The traffic network equilibrium conditions were stated by J.G. Wardrop [33] through two principles:

First principle: The journey times of all routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route.

Second principle: The average journey time is minimal.

In the *standard* traffic equilibrium problem, the travel cost on a link depends solely upon the flow on that link whereas the travel demand associated with an O/D pair may be either *fixed*, that is given, or *elastic*, that is, it depends upon the travel cost associated with the particular origin/destination (O/D) pair. M.J. Beckmann, C.B. McGuire, and C.B. Winsten [2] in their seminal work showed that the equilibrium conditions in the case of separable (and increasing) functions coincided with the optimality conditions of an appropriately constructed convex optimization problem. Such a reformulation also holds in the nonseparable case provided that the Jacobian of the functions is symmetric. The reformulation of the equilibrium conditions in the *symmetric* case as a convex optimization problem was also done in the case of the spatial price equilibrium problem by P.A. Samuelson [30].

S.C. Dafermos and F.T. Sparrow [14] coined the terms *user-optimized* and *system-optimized* transportation networks to distinguish between two distinct situations. In the user-optimized problem users act unilaterally, in their own self-interest, in selecting their routes, and the equilibrium pattern satisfies Wardrop's first principle, whereas is the system-optimized problem users select routes according to what is optimal from a societal point of view, in that the total costs in the system are minimized. In the latter problem, the marginal total costs rather than the average user costs are equilibrated. She also introduced *equilibration* algorithms based on the path formulation of the problem which exploited the network structure of the problem (see also [23]). Another algorithm that is widely used in practice for the symmetric TNE problem is the Frank–Wolfe algorithm (cf. ► Frank–Wolfe algorithm) [19].

Such a symmetry assumption was limiting, however, from both modeling and application standpoints. The discovery of Dafermos [6] that the traffic equilibrium conditions as formulated by M.J. Smith [31] defined a variational inequality problem allowed for such modeling extensions as: asymmetric link travel costs, link interactions, and multiple modes of transportation and classes of users. It also stimulated the development of rigorous algorithms for the computation of solutions to such problems as well as the qualitative study of equilibrium patterns in terms of the existence and uniqueness of solutions in addition to sensitivity analysis and stability issues.

Algorithms that have been applied to solve general traffic network equilibrium problems include projection and relaxation methods (cf. [1,3,4,6,7,8,9,17,22,25,27]) and simplicial decomposition (cf. [16], and the references therein). Projection and relaxation methods resolve the variational inequality problem into a series of convex optimization problems, with projection methods yielding quadratic programming problems and relaxation methods, typically, nonlinear programming problems. Hence, the overall effectiveness of a variational inequality-based method for the computation of traffic network equilibria will depend upon the algorithm used at each iteration.

Sensitivity analysis for traffic networks was conducted by M.A. Hall [20] and R. Steinberg and W. Zangwill [32] and in a variational inequality framework by Dafermos and A. Nagurney [10,11,12]. Some

of the work was, in part, an attempt to explain traffic network paradoxes such as the *Braess paradox* [5] (see also [15,18,24]) in which the addition of a link results in all users of the transportation network being worse off.

For background and additional material, including theoretical results, algorithms, and computational examples, see [16,25,26], and [28].

A variety of traffic network equilibrium models are now presented along with the variational inequality formulations of the governing equilibrium conditions.

Traffic Network Equilibrium with Travel Disutility Functions

The first model described here is due to Dafermos [7]. In this model, the travel demands are not known and fixed but are variables. The spatial price equilibrium problem (cf. [13]) is equivalent to this problem.

We consider a network $[N, L]$ consisting of nodes $[N]$ and directed links $[L]$. Let a denote a link of the network connecting a pair of nodes, and let p denote a path (assumed to be acyclic) consisting of a sequence of links connecting an O/D pair w. P_w denotes the set of paths connecting the O/D pair w with n_{P_w} paths. We let W denote the set of O/D pairs and P the set of paths in the network. We assume that there are J O/D pairs, n_A links, and n_p paths.

Let x_p represent the flow on path p and let f_a denote the load on link a. The following conservation of flow equation must hold for each link a:

$$f_a = \sum_p x_p \delta_{ap},$$

where $\delta_{ap} = 1$, if link a is contained in path p, and 0 otherwise. Hence, the load on a link a is equal to the sum of all the path flows on paths that contain the link a.

Moreover, if we let d_w denote the demand associated with an O/D pair w, then we must have that for each O/D pair w:

$$d_w = \sum_{p \in P_w} x_p,$$

where $x_p \geq 0$, for all p, that is, the sum of all the path flows on paths connecting the O/D pair w must be equal to the demand d_w. We refer to this expression as the demand feasibility condition. Let x denote the column vector of path flows with dimension n_P.

Let c_a denote the user cost associated with traversing link a, and let C_p the user cost associated with traversing path p. Then

$$C_p = \sum_a c_a \delta_{ap}.$$

In other words, the cost of a path is equal to the sum of the costs on the links comprising that path. We group the link costs into the row vector c with n_A components, and the path costs into the row vector C with n_P components. We also assume that we are given a travel disutility function λ_w for each O/D pair w. We group the travel disutilities into the column vector λ with J components.

We assume that, in general, the cost associated with a link may depend upon the entire link load pattern, that is, $c_a = c_a(f)$ and that the travel disutility associated with an O/D pair may depend upon the entire demand pattern, that is, $\lambda_w = \lambda_w(d)$, where f is the n_A-dimensional column vector of link loads and d is the J-dimensional column vector of travel demands.

Definition 1 (*traffic network equilibrium*; [2,7]) A vector $x^* \in \mathbf{R}_+^{n_P}$, which induces a vector d^*, through the demand feasibility condition, is a traffic network equilibrium if for each path $p \in P_w$ and every O/D pair w:

$$C_p(x^*) \begin{cases} = \lambda_w(d^*) & \text{if } x_p^* > 0 \\ \geq \lambda_w(d^*) & \text{if } x_p^* = 0. \end{cases}$$

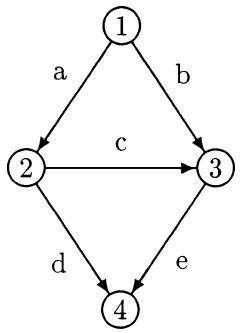
In equilibrium, only those paths connecting an O/D pair that have minimal user costs are used, and their costs are equal to the travel disutility associated with traveling between the O/D pair.

The equilibrium conditions have been formulated as a variational inequality problem by Dafermos [7]. In particular, we have:

Theorem 2 (cf. [7]) *$(x^*, d^*) \in K^1$ is a traffic network equilibrium pattern, that is, satisfies the equilibrium conditions if and only if it satisfies the variational inequality problem:*

path flow formulation

$$\langle C(x^*), x - x^* \rangle - \langle \lambda(d^*), d - d^* \rangle \geq 0, \quad \forall (x, d) \in K^1,$$



Traffic Network Equilibrium, Figure 1
A traffic network equilibrium example

where $K^1 \equiv \{(x, d) : x \geq 0$; and the demand feasibility condition holds$\}$, or, equivalently, $(f^*, d^*) \in K^2$ satisfies the variational inequality problem:

link flow formulation

$$\langle c(f^*), f - f^* \rangle - \langle \lambda(d^*), d - d^* \rangle \geq 0, \quad \forall (f, d) \in K^2,$$

where $K^2 \equiv \{(f, d)$: $x \geq 0$; and the conservation of flow and demand feasibility conditions hold$\}$ and $\langle \cdot, \cdot \rangle$ denotes the inner product.

Example 3 For illustrative purposes, we now present an example, which is illustrated in Fig. 1. Assume that there are 4 nodes and 5 links in the network as depicted in the figure and a single O/D pair $w = (1, 4)$. Define the paths connecting the O/D pair a: $p_1 = (a, d)$, $p_2 = (b, e)$, and $p_3 = (a, c, e)$.

Assume that the link travel cost functions are given by:

$$c_a(f) = 5f_a + 5f_c + 5,$$
$$c_b(f) = 10f_b + f_a + 5,$$
$$c_c(f) = 10f_c + 5f_b + 10,$$
$$c_d(f) = 7f_d + 2f_e + 1,$$
$$c_e(f) = 10f_e + f_c + 21,$$

and the travel disutility function is given by:

$$\lambda_w(d) = -3d_w + 181.$$

The equilibrium path flow pattern is: $x^*_{p1} = 10$, $x^*_{p2} = 5$, $x^*_{p3} = 0$, with induced link loads: $f^*_a = 10$, $f^*_b = 5$, $f^*_c = 0$, $f^*_d = 10$, $f^*_e = 5$, and the equilibrium travel demand: $d^*_w = 15$.

The incurred travel costs are: $C_{p1} = C_{p2} = 136$, $C_{p3} = 161$, and the incurred travel disutility $\lambda_w = 136$.

In the special case (cf. [2]), where the user link cost functions are separable, that is, $c_a = c_a(f_a)$, and the travel disutility functions are also separable, that is, $\lambda_w = \lambda_w(d_w)$, then the traffic network equilibrium pattern can be obtained as the solution to the optimization problem:

$$\min_{(f,d)\in K^2} \sum_a \int_0^{f_a} c_a(x)\, dx - \sum_w \int_0^{d_w} \lambda_w(y)\, dy.$$

Elastic Demand Traffic Network Problems with Known Travel Demand Functions

We now consider elastic demand traffic network problems in which the travel demand functions rather than the travel disutility functions are assumed to be given. The model is due to Dafermos and Nagurney [12] We retain the notation of the preceding model except for the following changes. We assume now that the demand d_w, associated with traveling between O/D pair w, is now a function, in general, of the travel disutilities associated with traveling between all the O/D pairs, that is, $d_w = d_w(\lambda)$. We assume now that the vector d is a row vector and the vector λ is a column vector.

Note that the expression relating the link loads to the path flows is still valid, as is the nonnegativity assumption on the path flows. In addition, the link cost and path cost functions are as defined previously.

The traffic network equilibrium conditions are now the following (cf. [2] and [12]):

Definition 4 (*traffic network equilibrium*) A path flow pattern x^* and a travel disutility pattern λ^* is a traffic network equilibrium pattern if, for every O/D pair w and each path $p \in P_w$, the following equalities and inequalities hold:

$$C_p(x^*) \begin{cases} = \lambda^*_w & \text{if } x^*_p > 0 \\ \geq \lambda^*_w & \text{if } x^*_p = 0, \end{cases}$$

and

$$d_w(\lambda^*) \begin{cases} = \sum_{p\in P_w} x^*_p & \text{if } \lambda^*_w > 0 \\ \leq \sum_{p\in P_w} x^*_p & \text{if } \lambda^*_w = 0. \end{cases}$$

The first system of equalities and inequalities above is analogous to the traffic network equilibrium conditions for the preceding model where now the equilibrium travel disutilities λ^* are to be determined, rather than the equilibrium travel demand d^*.

The second set of equalities and inequalities, in turn, has the following interpretation: if the travel disutility (or price) associated with traveling between an O/D pair w is positive, then the 'market' clears for that O/D pair, that is, the sum of the path flows on paths connecting that O/D pair are equal to the demand associated with that O/D pair; if the travel disutility (or price) is zero, then the sum of the path flows can exceed the demand.

Here we can immediately write down the governing variational inequality formulation in path flow and travel disutility variables (see, also, e. g., [12,25]).

Theorem 5 (*variational inequality formulation*) *$(x^*, \lambda^*) \in \boldsymbol{R}_+^{n_P+J}$ is a traffic network equilibrium if and only if it satisfies the variational inequality problem:*

$$\sum_w \sum_{p\in P_w} [C_p(x^*) - \lambda_w^*] \times [x_p - x_p^*] - \sum_w [d_w(\lambda^*) - \sum_{p\in P_w} x_p^*] \times [\lambda_w - \lambda_w^*] \geq 0, \quad \forall (x,\lambda) \in \mathbf{R}_+^{n_P+J},$$

or, in vector form:

$$\langle (C(x^*) - \widetilde{B}^\top \lambda^*)^\top, x - x^* \rangle - \langle (d(\lambda^*) - \widetilde{B}x^*)^\top, \lambda - \lambda^* \rangle \geq 0, \quad \forall (x,\lambda) \in \mathbf{R}_+^{n_P+J},$$

where $\widetilde{B}$ is the $(J \times n_P)$-dimensional matrix with element $(w, p) = 1$, if $p \in P_w$, and 0 otherwise.

Fixed Demand Traffic Network Problems

We now present the path flow and link load variational inequality formulations of the traffic network equilibrium conditions in the case of fixed travel demands, introduced in [31] and [6].

We retain the notation of the preceding two models. However, in contrast, it is assumed now that there is a fixed and known travel demand associated with traveling between each O/D pair in the network. Let d_w denote the traffic demand between O/D pair w, which is assumed to be known and fixed. The demand must satisfy, for each $w \in W$,

$$d_w = \sum_{p\in P_w} x_p,$$

where $x_p \geq 0$, $\forall p$, that is, the sum of the path flows between an O/D pair w must be equal to the demand d_w; such a path flow pattern is termed *feasible*.

Following [33] and [2], the traffic network equilibrium conditions are given as follows.

Definition 6 (*fixed demand traffic network equilibrium*) A path flow pattern x^*, which satisfies the demand, is a traffic network equilibrium, if, for every O/D pair w and each path $p \in P_w$, the following equalities and inequalities hold:

$$C_p(x^*) \begin{cases} = \lambda_w & \text{if } x_p^* > 0 \\ \geq \lambda_w & \text{if } x_p^* = 0, \end{cases}$$

where λ_w is the travel disutility incurred in equilibrium.

Again, as in the elastic demand models, in equilibrium, only those paths connecting an O/D pair that have minimal user travel costs are used, and those paths that are not used have costs that are higher than or equal to these minimal travel costs. However, here the demands and travel disutilities are no longer functions.

The equilibrium conditions have been formulated as a variational inequality problem by Smith [31] and Dafermos [6]. In particular, we present two formulations, in path flows and link loads, respectively.

Theorem 7 (*variational inequality formulation in path flows*) *$x^* \in K^3$ is a traffic network equilibrium in path flows if and only if it solves the following variational inequality problem:*

$$\langle C(x^*), x - x^* \rangle \geq 0, \quad \forall x \in K^3,$$

where $K^3 \equiv \{x \in \boldsymbol{R}_+^{n_P}$: the path flow pattern is feasible$\}$.

Theorem 8 (*variational inequality formulation in link loads*) *$f^* \in K^4$ is a traffic network equilibrium in link loads if and only if it satisfies the following variational inequality problem:*

$$\langle c(f^*), f - f^* \rangle \geq 0, \quad \forall f \in K^4,$$

where $K^4 \equiv \{f : \exists x \geq 0$, the path flow pattern is feasible and induces a link load pattern$\}$.

In the case where the Jacobian of the link travel cost functions is symmetric, i. e.,

$$\frac{\partial c_a(f)}{\partial f_b} = \frac{\partial c_b(f)}{\partial f_a},$$

for all links $a, b \in L$, then by Green's lemma the vector $c(f)$ is the gradient of the line vector $\int_0^f c(x)\,dx$. Moreover, if the Jacobian is positive semidefinite, then the traffic equilibrium pattern (f^*) coincides with the solution of the convex optimization problem:

$$\min_{f \in K^4} \int_0^f c(x)\,dx.$$

In particular, when the link travel cost functions c_a are separable, that is, $c_a = c_a(f_a)$ for all links a, then one obtains the objective function:

$$\min_{f \in K^4} \sum_a \int_0^{f_a} c_a(x)\,dx,$$

which is the classical and standard traffic network equilibrium problem with fixed travel demands (cf. [2]).

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Financial Equilibrium
- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Generalized Networks
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Oligopolistic Market Equilibrium
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Spatial Price Equilibrium
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Walrasian Price Equilibrium

References

1. Aashtiani HZ, Magnanti TL (1981) Equilibria on a congested transportation network. SIAM J Alg Discrete Meth 2:213–226
2. Beckmann MJ, McGuire CB, Winsten CB (1956) Studies in the economics of transportation. Yale Univ. Press, New Haven, CT
3. Bertsekas DP, Gafni EM (1982) Projection methods for variational inequalities with application to the traffic assignment problem. Math Program Stud 17:139–159
4. Boyce DE (1984) Urban transportation network-equilibrium and design models: recent achievements and future prospects. Environm Plan 16A:1445–1474
5. Braess D (1968) Über ein Paradoxon der Verkehrsplanung. Unternehmensforschung 12:256–268
6. Dafermos S (1980) Traffic equilibrium and variational inequalities. Transport Sci 14:42–54
7. Dafermos S (1982) The general multimodal traffic equilibrium problem with elastic demand. Networks 12:57–72
8. Dafermos S (1982) Relaxation algorithms for the general asymmetric traffic equilibrium problem. Transport Sci 16:231–240
9. Dafermos S (1983) An iterative scheme for variational inequalities. Math Program 26:40–47
10. Dafermos S, Nagurney A (1984) On some traffic equilibrium theory paradoxes. Transport Res 18B:101–110
11. Dafermos S, Nagurney A (1984) Sensitivity analysis for the asymmetric network equilibrium problem. Math Program 28:174–184
12. Dafermos S, Nagurney A (1984) Stability and sensitivity analysis for the general network equilibrium-travel choice model. In: Volmuller J, Hamerslag R (eds) Proc. 9th Internat. Symp. Transportation and Traffic Theory, VNU Sci. Press, Utrecht, pp 217–234
13. Dafermos S, Nagurney A (1985) Isomorphism between spatial price and traffic equilibrium models. Lefschetz Center Dynamical Systems, vol 85-17. Brown Univ., Providence, RI
14. Dafermos SC, Sparrow FT (1968) The traffic assignment problem for a general network. J Res Nat Bureau Standards 73B:91–118
15. Fisk C (1979) More paradoxes in the equilibrium assignment problem. Transport Res 13B:305–309
16. Florian M, Hearn D (1995) Network equilibrium models and algorithms. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing. Handbook Oper Res and Management Sci. Elsevier, Amsterdam, pp 485–550

17. Florian M, Nguyen S (1976) Recent experience with equilibrium methods for the study of a congested urban area. Lecture Notes Economics and Math Systems, vol 118. Springer, Berlin, pp 382–395
18. Frank M (1981) The Braess paradox. Math Program 20:283–302
19. Frank M, Wolfe P (1956) An algorithm for quadratic programming. Naval Res Logist Quart 3:95–110
20. Hall MA (1978) Properties of the equilibrium state in transportation networks. Transport Sci 12:208–216
21. Knight FH (1924) Some fallacies in the interpretations of social costs. Quart J Econom 38:582–606
22. Lawphongpanich S, Hearn DW (1984) Simplicial decomposition of the asymmetric traffic assignment problem. Transport Res 18B:123–133
23. Leventhal TL, Nemhauser GL, Trotter LE Jr (1973) A column generation algorithm for optimal traffic assignment. Transport Sci 7:168–176
24. Murchland JD (1970) Braess paradox of traffic flow. Transport Res 4:391–394
25. Nagurney A (1999) Network economics: A variational inequality approach. second, Kluwer, Dordrecht
26. Nagurney A, Zhang D (1996) Projected dynamical systems and variational inequalities with applications. Kluwer, Dordrecht
27. Pang JS, Chan D (1982) Iterative methods for variational and complementarity problems. Math Program 24:284–331
28. Patriksson M (1994) The traffic assignment problem. VSP, Utrecht
29. Pigou AC (1920) The economics of welfare. MacMillan, New York
30. Samuelson PA (1952) Spatial price equilibrium and linear programming. Amer Economic Rev 42:283–303
31. Smith MJ (1979) Existence, uniqueness, and stability of traffic equilibria. Transport Res 13B:295–304
32. Steinberg R, Zangwill W (1983) The prevalence of Braess' paradox. Transport Sci 17:301–318
33. Wardrop JG (1952) Some theoretical aspects of road traffic research. Proc Inst Civil Engineers, Part II, pp 325–278

Traveling Salesman Problem

Gregory Gutin
Department of Computer Science, Royal Holloway, University of London, Egham, UK

MSC2000: 90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90

Article Outline

Keywords and Phrases
Introduction
  Basic Definitions and Notation
  Computational Complexity
Formulations
Applications
Methods
  Exact Algorithms
See also
References

Keywords and Phrases

Traveling salesman problem, Asymmetric traveling salesman problem, Symmetric traveling salesman problem

Introduction

The Traveling Salesman Problem (TSP) is perhaps the most studied discrete optimization problem. Its popularity is due to the facts that TSP is easy to formulate, difficult to solve, and has a large number of applications. It appears that K. Menger [31] was the first researcher to consider the Traveling Salesman Problem (TSP). He observed that the problem can be solved by examining all permutations one by one. Realizing that the complete enumeration of all permutations was not possible for graphs with a large number of vertices, he looked at the most natural *nearest neighbor* strategy and pointed out that this *heuristic*, in general, does not produce the shortest route. (In fact, the nearest neighbor heuristic will generate the worst possible route for some problem instances of each size [17].) For interesting overviews of TSP history, see [20,40].

Basic Definitions and Notation

In applications, both the symmetric and asymmetric versions of the TSP are important. In the *Symmetric TSP (STSP)*, given a complete (undirected) graph K_n with weights on the edges, our aim is to find a Hamiltonian cycle in K_n of minimum weight (the weight a cycle is the sum of the weights of its edges). In the *Asymmetric TSP (ATSP)*, given a complete directed graph K_n^* with weights on the arcs, find a Hamiltonian cycle in K_n^* of minimum weight. The *Euclidean TSP* is a special case of STSP in which the vertices are points

in the Euclidean plane and the weight on each edge is the Euclidean distance between its endpoints. A Hamiltonian cycle in K_n or K_n^* is often called a *tour*. Notice that ATSP has $(n-1)!$ tours, but STSP has $(n-1)!/2$ tours (changing the direction of the tour in STSP does not change the tour). By *TSP* we refer to both STSP and ATSP simultaneously.

Throughout this entry, the set $[n] = \{1, 2, \ldots, n\}$ denotes the vertices of K_n or K_n^* or any other n-vertex graph under consideration. The weight of an edge (arc) ij is denoted by w_{ij} or $w(i, j)$. We also call w_{ij} the *distance from i to j* and the *length* of ij. A *cycle factor* is a collection of vertex-disjoint cycles in K_n^* covering all vertices of K_n^*.

Computational Complexity

The Hamiltonian cycle problem on an n-vertex graph G can be transformed into STSP by converting G to an edge-weighted K_n as follows: assign weight 0 to each edge of G; and assign weight 1 to each edge in the complement of G. A similar transformation can be used for digraphs and ATSP. This implies that TSP is NP-hard, even if the triangle inequality holds. By replacing the weights 0 by 1 and the weights 1 by $1 + nr$ in this transformation, we obtain the following result:

Proposition 1 *For an arbitrary constant r, unless $P = NP$, there is no polynomial time algorithm that always produces a tour of total weight at most r times the optimal.*

It was proved in [12,38] that even Euclidean TSP is NP-hard. Despite this result, there was a feeling among some researchers that the Euclidean TSP is somewhat simpler than the general STSP. More precisely, Proposition 1 does not hold for the Euclidean TSP. This was confirmed by Arora [1] in 1996, see Theorem 2. Mitchell [33] independently made a similar discovery a few months later (see [2]).

Theorem 2 *For every $\epsilon > 0$, there is a polynomial time algorithm $\mathcal{A}_\epsilon$ that, for any instance of the Euclidean TSP, finds a tour at most $1 + \epsilon$ times longer than the optimal one.*

As of this writing, the fastest algorithm $\mathcal{A}_\epsilon$ has time complexity $O(n \log n + n/\text{poly}(\epsilon))$ [42]. These $\mathcal{A}_\epsilon$ algorithms have been implemented, but, in their current form, they are not competitive with best TSP heuristics [2].

Arora's result can be generalized to d-dimensional Euclidean space for any constant d. However, the next theorem limits the scope of this generalization.

Theorem 3 *[45] There exists a constant $r > 1$ such that, for the Euclidean TSP in $O(\log n)$-dimensional Euclidean space, the problem of finding a tour that is at most r times longer than the optimal tour is NP-hard.*

We finish this subsection with a result from [16] that indicates another limitation for 'approximation' ATSP algorithms. The *domination number* of an ATSP heuristic $\mathcal{H}$ is the maximum $d(n)$ such that for each instance of ATSP on n vertices, $\mathcal{H}$ produces a tour T which is not worse than at least $d(n)$ tours including T itself.

Theorem 4 *Unless $P = NP$, there is no polynomial time ATSP heuristic of domination number at least $(n-1)! - \lfloor n - n^\alpha \rfloor !$ for any constant $\alpha < 1$.*

Formulations

Perhaps, the simplest combinatorial formulation of ATSP is as follows: given an $n \times n$-matrix $W = [w_{ij}]$ find a permutation π of $[n]$ that minimizes the sum

$$w_{\pi(n),\pi(1)} + \sum_{i=1}^{n-1} w_{\pi(i),\pi(i+1)}.$$

For STSP, we require that W is symmetric.

The earliest (and very useful) integer programming formulation of ATSP is due to Dantzig, Fulkerson and Johnson [10]. Define $n^2 - n$ zero-one variables x_{ij} by $x_{ij} = 1$, if the tour traverses arc ij and $x_{ij} = 0$, otherwise. Then ATSP can be expressed as:

$$\min z = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_{ij}$$

$$\text{such that} \sum_{i=1}^{n} x_{ij} = 1, \quad j \in [n]$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i \in [n]$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \le |S| - 1 \text{ for all } |S| < n$$

$$x_{ij} = 0 \text{ or } 1, \ i \ne j \in [n].$$

The first set of constraints ensures that a tour must come into vertex j exactly once, and the second set of constraints indicates that a tour must leave every vertex i exactly once. These two sets of constraints ensure that there are two arcs adjacent to each vertex, one in and one out. However, this does not prevent non-Hamiltonian cycles. Instead of having one tour, the solution can consist of two of more vertex-disjoint cycles (called *sub-tours*), i. e., be a cycle factor with $t \geq 2$ cycles. The third set of constraints, called *sub-tour elimination constraints*, requires that no proper subset of vertices, S, can have a total of $|S|$ arcs.

For STSP, we can get the following similar formulation:

$$\min z = \sum_{1 \leq i < j \leq n} w_{ij} x_{ij} \tag{1}$$

$$\text{such that} \sum_{i=1}^{n} x_{ij} = 2, \quad j \in [n] \tag{2}$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 2 \text{ for all } 3 \leq |S| \leq n/2 \tag{3}$$

$$0 \leq x_{ij} \leq 1, \quad i \neq j \in [n] \tag{4}$$

$$x_{ij} \text{ is integral for all } i \neq j \in [n]. \tag{5}$$

While the Dantzig–Fulkerson–Johnson formulation of ATSP has an exponential number of sub-tour elimination constraints and, thus, of all constraints, there are other integer programming ATSP formulations that contain only a polynomial number of constraints. One such example is the formulation of Miller, Tucker and Zemlin [32]. In this formulation, we use $(n-1)(n-2)$ additional constraints and $n-1$ additional variables. The following constraints replace the sub-tour elimination constraints in the Dantzig–Fulkerson–Johnson formulation:

$$(n-1)x_{ij} + u_i - u_j \leq (n-2) \text{ for all } i \neq j = 2, 3, \ldots, n,$$

where u_i, $i = 2, 3, \ldots, n$ are unrestricted real variables. If a solution is not a tour, it contains a cycle C without vertex 1. By adding the inequalities above corresponding to all arcs ij of C, we arrive at a contradiction.

Notice that the Dantzig–Fulkerson–Johnson formulation of ATSP is stronger than the Miller–Tucker–Zemlin formulation in the following sense: the optimal value of the linear relaxation of the former is larger than that of the latter [37]. As for the STSP, there are two other formulations that are as strong as the the Dantzig–Fulkerson–Johnson formulation for STSP, but only the latter have been used in computational practice. For more information on various formulations of TSP, see [40].

Applications

It appears that the most natural and well-studied application area of the TSP is machine scheduling. A simple scheduling application can be described as follows. Suppose there are n jobs $1, 2, \ldots, n$ to be processed sequentially on a machine. Let w_{ij} be the set up cost required for processing job j immediately after job i. When all the jobs are processed, the machine is reset to its initial state at a cost of w_{j1}, where j is the last job processed. The aim of the *Sequencing Problem* is to find an order in which the jobs are to be processed so as to minimize the total setup cost. Observe that finding a permutation π of $[n]$ that minimizes $w_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} w_{\pi(i)\pi(i+1)}$ solves the problem. Thus, the Sequencing Problem is equivalent to ATSP.

Now consider a more interesting application introduced and studied by Gutin et al. [15]. The *Seismic Vessel Problem (SVP)* is defined by a set of line segments (survey lines) on the plane, all of which need to be traversed exactly once. Some lines can be traversed in either direction, other have directional constraints imposed on them. The objective is to minimize the travel time between lines by choosing an optimal ordering of lines (and specifying in which direction each line has to be traversed). The function that defines the travel time between lines can be of arbitrary complexity and in general is defined as a matrix of 'line change' weights for all combinations of pairs of lines and traversing directions.

More formally, SVP can be stated as follows: We are given a weighted complete digraph K_n^*, whose vertices are partitioned into pairs P (representing survey lines). Each pair $\{u, v\} \in P$ is assigned a set F_{uv} such that $\emptyset \neq F_{uv} \subseteq \{uv, vu\}$. If $F_{uv} = \{uv\}$, then we must traverse the survey line corresponding to uv from u to v, and if $F_{uv} = \{uv, vu\}$, either traversing direction is possible. Let $F = \{uv : \{u, v\} \in P, uv \in F_{uv}\}$. Every arc in F is assigned weight zero (as we must traverse all survey lines and we assume that the time of traversal of a survey line in both directions is the same).

We are required to find a minimum weight Hamilton cycle in K_n^* that traverses one arc from F_{uv} for every pair $\{u, v\} \in P$.

The *Stacker Crane Problem (SCP)* studied in [18,21] is a special case of SVP. In SCP, F_{uv} consists of one arc for every pair $\{u, v\} \in P$. To see that SCP is equivalent to ATSP it suffices to contract all arcs of F.

Consider SVP. In order to enforce the requirement that a Hamilton cycle has to traverse one arc in F_{uv} for each pair $\{u, v\} \in P$, we apply a transformation which results in a weighted complete undirected graph. Solving STSP on the transformed graph provides a solution to the original problem. The transformation replaces each pair $\{u, v\} \in P$ with a graph. We consider only the more interesting case when the line $\{u, v\}$ is undirected (that is, $|F_{uv}| = 2$). In this case, the two vertices are replaced with the so-called diamond graph D_8 with $V(D_8) = \{N, W, E, S, a, b, c, d\}$ and $E(D_8) = \{Na, aW, Nb, bE, Wc, cS, Ed, dS, bc\}$. The diamond graph can be traversed in two possible ways, $N-S$ and $W-E$ (see Chapter 19 in [39]). These correspond to traversing the original pair of vertices, $\{u, v\}$ via arcs uv and vu, respectively. To make the weight of the tour consistent with the original graph:

- We set the weight of edges incident to W to be the same as the weight of the corresponding original arcs entering vertex v;
- The weight of edges incident to E are taken to be the same as weight of arcs leaving u;
- The weight of edges incident to N are taken to be the same as weight of arcs entering u;
- The weight of edges incident to S are taken to be the same as weight of arcs leaving v;
- Since arcs uv and vu have zero weight, all edges inside the diamond graph have their weight set to 0.
- The vertices a, b, c, d are not adjacent to any vertices outside their copy of D_8.

For more TSP applications, see, e. g., [40].

Methods

The methods to solve TSP can be divided into two large classes: *exact algorithms* that solve the problem or its special cases to optimality and the algorithms that normally provide non-optimal tours. The members of the second class are called TSP *heuristics* or TSP *approximation algorithms* (the latter is often used if there is some kind of approximation guarantee). Exact algorithms are used when we want to obtain an optimal tour. This may not be possible as exact algorithms may well require several hours or days of running time even for instances of moderate size (for example, the authors of [11] found out that no state-of-the art exact algorithm could solve some ATSP instances with 316 vertices within the limit of 10^4 sec.). When running time is limited or the data of the instance is not exact, one can use TSP heuristics. For discussion of TSP software implementing both exact algorithms and heuristics, see [30] and the site http://www.or.deis.unibo.it/research.html.

Exact Algorithms

The *brute-force* method of explicitly examining all possible TSP tours is impractical even for moderately sized problem instances because there are $(n-1)!/2$ different tours in K_n and $(n-1)!$ different tours in K_n^*. The well-known dynamic programming algorithm of Hell and Karp [19] reduces the running time to $O(n^2 2^n)$ only, but this time complexity is still far too large to solve even TSP instances of moderate size. On the other hand, branch-and-bound, branch-and-cut and other branching algorithms are proved to be quite efficient in practice; branch-and-bound algorithms will be discussed in this subsection.

While every STSP instance can be considered as an ATSP instance and, thus, solved using ATSP algorithms, normally STSP-specialized algorithms are used for STSP instances as such algorithms are often more efficient that their ATSP counterparts (partially because they exploit a more special structure of STSP and partially because STSP algorithms have received significantly more attention than their ATSP counterparts). Moreover, in many cases ATSP instances are transformed into STSP instances and subsequently solved using STSP algorithms. (This situation may change in the future when advanced ATSP solvers will have been developed.)

In this subsection, we will consider two ATSP-to-STSP transformations and basic ideas behind STSP branch-and-bound algorithms. We will not consider special polynomial-time solvable cases of TSP; instead we refer the reader to [5,26] which are excellent surveys on the topic.

The following are well-known ATSP-to-STSP transformations:

The 2-node transformation: Replace every vertex i of K_n^* by a pair i^-, i^+ of vertices to form K_{2n}. The weights of edges of K_{2n} are defined as follows: all weights are equal to $+\infty$ apart from $w(i^-, i^+) = 0$ and $w(i^+, j^-) = w(i, j) + M$ for all $i \neq j \in [n]$, where $w(i, j)$ is the weight of arc ij in K_n^* and M is a sufficiently large constant. The transformation value nM has to be subtracted from the STSP optimal weight to obtain the ATSP optimal weight. The transformation was introduced by Jonker and Volgenant [24].

The 3-node transformation: Replace every vertex i of K_n^* by a triple i^-, i^0, i^+ of vertices to form K_{3n}. The weights of edges of K_{3n} are defined as follows: all weights are equal to $+\infty$ apart from $w(i^-, i^0) = w(i^0, i^+) = 0$ and $w(i^+, j^-) = w(i, j)$ for all $i \neq j \in [n]$, where $w(i, j)$ is the weight of arc ij in K_n^*. The transformation was introduced by Karp [28].

Each transformation has its pros and cons, see [11,21].

Now we consider basic ideas behind TSP branch-and-bound and branch-and-cut algorithms using the Dantzig–Fulkerson–Johnson formulation of STSP. The formulation allows us to treat STSP as an integer programming problem. If we drop (5), we will get a linear programming problem whose solution will give us a lower bound to STSP. The linear program is called the *linear relaxation* of STSP. A branch-and-bound algorithm for STSP could be as follows.

Step 1 A list L of problems to solve is initialized by including into it the linear program discussed above. This problem is called the *root problem*.

Step 2 If $L = \emptyset$, then the best known feasible solution (tour) is optimal. Otherwise, choose a problem P and delete it from the list.

Step 3 (a) Solve the linear relaxation of P. If the solution is integral, return to Step 2 after eventually updating the best known integral solution and the best known solution value.

(b) If the value of the objective function exceeds that of the best known feasible solution, return to Step 2.

(c) Otherwise, using some linear inequality, partition the current problem into two new problems which are added to L. The union of the feasible (integral) solutions to each of these two problems contains all the feasible solutions of the problem that has been partitioned. This is commonly done by choosing a variable with a current fractional value $\overline{x}_{ij}$ and imposing $x_{ij} \geq 1$ in one problem and $x_{ij} \leq 0$ in the other. Return to Step 2.

Due to computer memory limitations, the branch-and-bound algorithm is appropriate for an STSP formulation with a polynomial number of constraints, but this is not the case for the Dantzig–Fulkerson–Johnson formulation of STSP. Thus, we need to use a method that allows to store only a small number of constraints at any given moment of time. One such method is *row generation*. Using row generation, we replace Step 1 of the above algorithm by the following: we initially solve the problem consisting of (1), (2) and (4) obtaining a solution $\overline{x}$. Now we try to find a set $S \subset [n]$ such that $\sum_{i \in S} \sum_{j \notin S} \overline{x}_{ij} < 2$. To do that we can use an efficient algorithm for computing a minimum cut in a weighted undirected graph applied to K_n with weight function $\overline{x} : E(K_n) \to \mathbb{R}$ (see, e. g., [7,25]). If a desired set S is found, we add the constraint to the current linear program and solve it to find a new vector $\overline{x}$, and continue as above. If no desired set S has been found, the problem is solved.

Similarly, one can solve other problems from the list L. In practice, we need to apply a minimum cut algorithm very few times, see, e. g., [36]. The solution found in Step 1 usually provides a good lower bound called the *Held–Karp bound* in the literature.

STSP computational practice indicates that while branch-and-bound algorithms are fairly efficient in solving STSP, branch-and-cut algorithms are normally much more efficient. For an excellent overview of STSP branch-and-cut algorithms, see [36].

TSP Heuristics TSP heuristics can be roughly partitioned into two classes: construction heuristics, and improvement heuristics. Both classes and their performances in computational experiments are discussed below. More comprehensive overviews of TSP heuristics can be found in [14,21] and [22]. Notice that [21] and [22] discuss families of instances on which many TSP heuristics have been tested. Many new heuristics are now tested using these families. This allows one to

compare new heuristics with many known ones without too much effort.

We finalize the Heuristic subsection by a brief discussion of approximation analysis of TSP heuristics.

Construction Heuristics *Construction heuristics* build a tour from scratch and stop when one is produced. The simplest and most obvious construction heuristic is *nearest neighbor (NN)*: the tour starts at any vertex x of the complete directed or undirected graph; we repeat the following loop until all vertices have been included in the tour: add to the tour a vertex (among vertices not yet in the tour) closest to the vertex last added to the tour. The *greedy algorithm* is based on the observation that a vertex-disjoint collection of paths in K_n^* (K_n) can be extended to a tour in K_n^* (K_n). In the ATSP greedy algorithm, we order all arcs $a_1, a_2, \ldots, a_{n(n-1)}$ such that $w(a_i) \leq w(a_{i+1})$ for each $i = 1, 2, \ldots, n(n-1)-1$, set $C := \emptyset$ and, in the ith iteration, we check whether the arcs of C and a_i form a vertex-disjoint collection of paths or a tour, and if it is so, we add a_i to C.

Computational experiments in [21] indicate that, in fact, on most real-world-like problem instances of ATSP, NN performs better than the greedy algorithm; the greedy algorithm fails completely on one family of instances, where the average greedy-tour is more than 2000% above the optimum. Computational experiments for STSP in [22] show that both the greedy algorithm and NN perform relatively well on Euclidean instances and perform poorly for general STSP. The greedy algorithm appears to perform better than NN for STSP.

Vertex insertion (VI) is another type of TSP construction heuristic. For ATSP, the insertion algorithm begins with a cycle of length 2, and in each iteration, inserts a new vertex into the cycle. For STSP, the algorithm beings with a cycle of length 3. We describe only the ATSP vertex insertion, but the STSP algorithm is similar. Let C be a cycle in K_n^*, and let v be a vertex not on C. For any arc ab on cycle C, the *insertion of vertex v at* arc ab is the operation of replacing arc ab with the arcs av and vb. The resulting cycle is denoted $C(a, v, b)$. Observe that the difference between the wights of $C(a, v, b)$ and C equals $w(a, v) + w(v, b) - w(a, b)$. The VI algorithm always inserts a vertex v at arc ab of C for which $w(a, v) + w(v, b) - w(a, b)$ minimum.

Random vertex insertion (RVI), *nearest vertex insertion (NVI)*, and *farthest vertex insertion (FVI)*, which are defined below, are three different versions of algorithm VI. Each one of them is determined by how it chooses vertex v to be inserted into the current cycle C. Given a vertex v and a cycle C in K_n^*, $d(v, C)$ denotes the distance from v to C, that is, $d(v, C) = \min\{w(v, x) : x \in V(C)\}$. The algorithm RVI chooses vertex v randomly. The algorithm NVI chooses vertex v so that its distance to cycle C is a minimum. That is, $d(v, C) = \min\{d(u, C) : u \notin V(C)\}$. The algorithm FVI chooses vertex v so that its distance to cycle C is a maximum. That is, $d(v, C) = \max\{d(u, C) : u \notin V(C)\}$.

The vertex insertion heuristics described above perform quite well for Euclidean TSP (see [22]). Computational experiments with RVI for ATSP in [13] show that RVI is good only for instances close to Euclidean.

The following heuristic was initially suggested, in a different form, for the Vehicle Routing Problem by Clark and Write [9]. In the *savings heuristic*, we choose one vertex, say, n and compute new weights $w'(i, j) = w(i, j) - w(i, n) - w(n, j)$ for all $i \neq j \in [n-1]$. Then the greedy algorithm is applied for the new weights in $K_n - n$ ($K_n^* - n$) until all vertices (but n) are included in a path. Then n is added to the path to form a tour. The savings heuristic showed very good results for STSP in the computational experiments discussed in [22], in which the heuristic clearly outperformed the greedy algorithm, NN, RVI, NVI, FVI and a large number other heuristics. (The saving heuristic has not been tested for ATSP in [21].)

The only heuristic, some versions of which could successfully compete with the savings heuristic in the experiments in [22], was the well-known Christofides heuristic [8]. The Christofides heuristic is designed only for STSP and proceeds as follows: First we find a minimum weight spanning tree T in K_n. Let X be the vertices of odd degree in T. It is well-known that $|X|$ is even and, thus, the subgraph G of K_n induced by X has a perfect matching. We compute a minimum weight perfect matching M in G. The edges of T and M for an Euler graph H as all vertex degrees are even. We find an Euler trail R of H and 'short-cut' it, i. e., delete all repetitions of the same vertex in R. As a result, we obtain a tour. The way of short-cutting is very important for getting good quality tours [22].

According to [21] the best ATSP construction heuristics are based on finding a minimum weight cycle factor (a vertex-disjoint collection of cycles covering all vertices of K_n^*) and merging the cycles (the process often called *patching* in the literature) to obtain a tour. The operation of patching of two cycles C and Z deletes an arc in each of the cycles and adds an arc from C to Z and an arc from Z to C such that we obtain a cycle containing all vertices of C and Z. Often patching of cycles $C = i_1 i_2 \dots i_s i_1$ and $Z = j_1 j_2 \dots j_t j_1$ is done *optimally*, i. e., we delete arcs $i_p i_{p+1}$ and $j_q j_{q+1}$ such that the cycle

$$i_1 i_2 \dots i_p j_{q+1} j_{q+2} \dots j_t j_1 \dots j_q i_{p+1} i_{p+2} \dots i_s i_1$$

is of minimum possible weight.

The following simple yet very successful patching heuristic was introduced by Karp and Steele [29]. In the Karp–Steele heuristic, we always choose a pair of cycles (in the current cycle factor) with maximum number of vertices and patch them optimally. The Karp–Steele heuristic performs not so good when the minimum weight cycle factor has many cycles with just two vertices. In such cases, another patching heuristic, *contract-or-patch (COP)* gives better results [21]. COP partitions the cycles of the cycle factor into short and long cycles (a short cycle has at most t vertices for some fixed t). COP deletes the heaviest arc from each short cycle and contracts each such path using the operation of path-contraction defined shortly. COP finds a minimum cost cycle factor in the new complete digraph and continues as above until the current cycle factor has no short cycles. In the last case, COP applies the Karp–Steele heuristic, computes a tour and 'extends' it to a tour in K_n^* in the obvious way. For a directed path $P = x_1 x_2 \dots x_p$ in K_n^*, the operation of *path-contraction* (see [3] for the case of general weighted digraphs) consists of replacing all vertices of P in K_n^* with a single new vertex v and assigning weights in the new digraph K_{n-p+1}^* as follows: the weight between vertices not including v is the same as in K_n^*, the weight $w(v, u)$ in K_{n-p+1}^* equals $w(x_p, u)$ in K_n^* and the weight $w(u, v)$ in K_{n-p+1}^* equals $w(u, x_1)$ in K_n^* for each $u \in V(K_n^*) \setminus V(P)$. The contract-or-patch heuristic was introduced by Glover et al. [13].

Improvement Heuristics *Improvement heuristics* start from a tour normally obtained using a construction heuristic and iteratively improve it by changing some parts of it at each iteration. Improvement heuristics are typically much faster than the exact algorithms, yet often produce solutions very close to the optimal one.

It appears that currently the best improvement heuristics are based on local search, on genetic algorithm approach, or on a mixture of the two, which is often called *memetic algorithms*. The most developed TSP improvement algorithms are local search algorithms that use *edge exchange*, in which a tour is improved by replacing k its edges with k edges not in the solution. For STSP, the *2-opt* algorithm starts from an initial tour T and tries to improve T by replacing two of its nonadjacent edges with two other edges to form another tour. Once an improvement is obtained, it becomes the new T. The procedure is repeated as long as an improvement is possible (or a time limit is exceeded). For $k \geq 3$, the *k-opt* algorithm is the same as 2-opt except that k edges are replaced at each iteration.

The best local search algorithms use a variable k-opt search called the *Lin–Kernighan local search*, where at each iteration the actual value of k varies depending on which value of k gives the best improvement, for details see, e. g., [43]. Although the Lin–Kernighan local search can be applied only to STSP, ATSP can be transformed into STSP (see above). However, there is an approach, the *ejection chain methods*, which include the Lin–Kernighan search, that are applicable to ATSP. Recently, Rego et al. [44] developed a new ejection chain method, the doubly-rooted Stem-and-Cycle method that can be directly applied to ATSP. Computational experiments in [44] clearly demonstrated high efficiency of the new method. One interesting aspect of the method indicated in [44] is the fact that the method allows one to construct tours, in polynomial time, that are better than an exponential number of other tours.

The main problem with any kind of local search is that no further improvement is possible once we have found a local optimum. To get around this problem, one can restart the local search from another tour and repeat this many times. In the end, the best of all found tours gives us a solution. In practice, two ways to obtain restarting tours have been used. In the first (called *iterated local search*), a restarting tour is produced by a construction heuristic as before. In the second (*chained local search*), a kind of perturbation is applied to the current or previous local optimum to obtain a restarting tour. It seems Baum [6] was the first

to introduce chained local search; this method proved to be significantly better than the iterated local search for large instances of STSP (see, e. g., Johnson and McGeoch [22]).

Genetic algorithms operate with a large number of tours at any given time. They produce the initial *population* of tours and consecutively several other populations such that the best tour in the previous population is not worse than the best tour in the current population. *Genetic operators* that change tours include mutations (a *mutation* makes small changes to a single tour) and crossovers. A *crossover* selects two tours and produces a new tour from them. It appears that the currently most efficient crossovers are variations of the *edge assembly crossover (EAX)* introduced by Nagata and Kobayashi [35]. In EAX, we identify a set A of edges from the first tour and a set B of edges from the second tour such that $A \cup B$ forms a collection of alternating cycles (i. e., cycles in which edges alternate between the first and second tours) and replace all edges from A by the edges of B resulting in a cycle factor. Then an operation of patching is applied to the cycle factor. Recently, Nagata [34] reported on very impressive results for large instances of STSP achieved by a genetic algorithm using a new version of EAX and no local search.

Worst Case Analysis of Heuristics While computational experiments are important in the evaluation of heuristics, they cannot cover all possible families of instances of TSP and, in particular, they normally do not cover the most difficult instances. Moreover, certain applications may produce families of instances that are much harder than those normally used in computational experiments. For example, such instances can arise when the Generalized TSP is transformed into TSP. Thus, theoretical analysis of the worst possible cases is also important in evaluating and comparing TSP heuristics. One way to analyze worst cases of heuristics is Domination Analysis, see its entry in this book.

We provide only a brief overview of the second approach to the worst case analysis of heuristics, Approximation Analysis. For the STSP with *triangle inequality* (i. e., $w_{ij} + w_{jk} \geq w_{ik}$ for all vertices i, j, k), the best known approximation is 3/2 provided by the Christofides algorithm discussed earlier. The performance guarantee 3/2 means that a tour produced by the heuristic has weight which is at most 50% larger than that of an optimal tour. For the Euclidean TSP, we can obtain much better approximation as we saw earlier. For ATSP with triangle inequality, no algorithm with constant approximation guarantee is known. The best approximation ratio so far was obtained by Kaplan et al. [27]: $0.841 \cdot \log n$. Recently, Blaeser et al. [4] obtained a constant approximation guarantee when a *strengthen triangle inequality* holds: for some $\gamma \in [1/2, 1)$ we have $\gamma \cdot (w_{ij} + w_{jk}) \geq w_{ik}$ for all vertices i, j, k. The authors of [4] proved that their algorithm always produces a tour at most $(1 + \gamma) / (2 - \gamma - \gamma^3)$ times longer than an optimal one.

We saw above that, if no triangle inequality is imposed, there is no polynomial-time TSP algorithm with constant approximation guarantee (unless P=NP). We can overcome the inapproximability, by using another measure of performance guarantee. One such measure was defined by Zemel [46] who provided some mathematical arguments to show that his measure is better, in some sense, than the traditional performance (approximation) ratio. Let $\mathcal{A}$ be a heuristic for TSP and I a problem instance. Then $w_{\min}(I)$, $w_{\max}(I)$, $w_{\mathcal{A}}(I)$ denote the weights, respectively, of an optimal tour, a heaviest tour, and a tour produced by $\mathcal{A}$ for instance I. The *Zemel measure* of $\mathcal{A}$, denoted $\rho_z(\mathcal{A})$, is the supremum of $(w_{\mathcal{A}}(I) - w_{\min}(I))/(w_{\max}(I) - w_{\min}(I))$, taken over all TSP instances I for which $w_{\max}(I) \neq w_{\min}(I)$. The following theorem was proved by Hassin and Khuller [18].

Theorem 5 *There is a polynomial-time heuristic $\mathcal{A}$ for ATSP with $\rho_z(\mathcal{A}) \leq 1/2$, and one for STSP with $\rho_z(\mathcal{A}) \leq 1/3$.*

See also

- ▶ Domination Analysis in Combinatorial Optimization
- ▶ Evolutionary Algorithms in Combinatorial Optimization
- ▶ Heuristic and Metaheuristic Algorithms for the Traveling Salesman Problem

References

1. Arora S (1998) Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. J ACM 45:753–782

2. Arora S (2002) Approximation algorithms for geometric TSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 207–221
3. Bang-Jensen J, Gutin G (2000) Digraphs: Theory, Algorithms and Applications. Springer, London
4. Blaeser M, Manthey B, Sgall J (2006) An Improved Approximation Algorithm for the Asymmetric TSP with Strengthened Triangle Inequality. J Discret Algorithms 4:623–632
5. Burkard RE, Deineko VG, van Dal R, van der Veen JAA, Woeginger GJ (1998) Well-solvable special cases of the traveling salesman problem: a survey. SIAM Rev 40:496–546
6. Baum EB (1986) Iterated descent: A better algorithm for local search in combinatorial optimization problems. unpublished manuscript
7. Chekuri C, Goldberg A, Karger D, Levine M, Stein C (1997) Experimental Study of Minimum Cut Algorithms. In: Proc. 8th ACM-SIAM Symp. Discrete Algorithms (SODA'97). ACM/SIAM, pp 324–333
8. Christofides N (1976) Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report CS-93–13, Carnegie Mellon University
9. Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 12:568–581
10. Dantzig GB, Fulkerson DR, Johnson SM (1954) Solution of large scale traveling salesman problem. Oper Res 2:393–410
11. Fischetti M, Lodi A, Toth P (2002) Exact Methods for the Asymmetric Traveling Salesman Problem. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 169–205
12. Garey MR, Graham RL, Johnson DS (1976) Some NP-complete geometric problems. In: Proc. 8th ACM Symp. Theory Comput. ACM, pp 10–22
13. Glover F, Gutin G, Yeo A, Zverovich A (2001) Construction heuristics for the asymmetric TSP. Eur J Oper Res 129:555–568
14. Golden BL, Stewart WR (1985) Empirical Analysis of Heuristics. In: Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (eds) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York, pp 207–249
15. Gutin G, Jakubowicz H, Ronen S, Zverovitch A (2005) Seismic vessel problem. Commun DQM 8:13–20
16. Gutin G, Koller A, Yeo A (2006) Note on Upper Bounds for TSP Domination Number. Algorithmic Oper Res 1:52–54
17. Gutin G, Yeo A, Zverovich A (2002) Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. Discrete Appl Math 117:81–86
18. Hassin R, Khuller S (2001) z-Approximations. J Algorithms 41:429–442
19. Held M, Karp RM (1962) A dynamic programming approach to sequencing problems. J Soc Ind Appl Math 10:196–210
20. Hoffman AJ, Wolfe P (1985) History. In: Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (eds) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York, pp 1–16
21. Johnson DS, Gutin G, McGeoch LA, Yeo A, Zhang W, Zverovitch A (2002) Experimental Analysis of Heuristics for ATSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht
22. Johnson DS, McGeoch LA (2002) Experimental Analysis of Heuristics for STSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 369–443
23. Johnson DS, Papadimitriou CH (1985) Performance guarantees for heuristics. In: Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (eds) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York, pp 145–180
24. Jonker R, Volgenant T (1983) Transforming asymmetric into symmetric traveling salesman problems. Oper Res Lett 2:161–163
25. Jünger M, Rinaldi G, Thienel S (2000) Practical Performance of efficient minimum Cut Algorithms. Algorithmica 26:172–195
26. Kabadi SN (2002) Polynomially solvable cases of the TSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 489–583
27. Kaplan H, Lewenstein M, Shafrir N, Sviridenko M (2005) Approximation Algorithms for the Asymmetric TSP by Decomposing Regular Multigraphs. J ACM 52:602–626
28. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations. Plenum Press, New York, pp 85–103
29. Karp RM, Steele JM (1985) Probabilistic analysis of heuristics. In: Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (eds) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York, pp 181–205
30. Lodi A, Punnen AP (2002) TSP Software. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 737–749
31. Menger K (1932) Das Botenproblem. Ergebnisse eines Mathematischen Kolloquiums 2:11–12
32. Miller CE, Tucker AW, Zemlin RA (1960) Integer Programming formulations and traveling salesman problems. J Assoc Comput Mach 7:326–329
33. Mitchell JCB (1999) Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial time approximation scheme for geometric TSP, k-MST, and related problem. SIAM J Comput 28:1298–1309
34. Nagata Y (2006) New EAX crossover for large TSP instances. Lecture Notes Comp Sci 4193:372–381
35. Nagata Y, Kobayashi S (1997) Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem. In: Proc. of the 7th Int. Conference on Genetic Algorithms, pp 450–457

36. Naddef D (2002) Polyhedral Theory and Branch-and-Cut Algorithms for the Symmetric TSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 29–116
37. Padberg MW, Sung TY (1991) An analytical comparison of different formulations of the traveling salesman problem. Math Program Ser A 52:315–357
38. Papadimitriou CH (1977) The Euclidean traveling salesman problem is NP-complete. Theoret Comput Sci 4:237–244
39. Papadimitriou CH, Steiglitz K (1982) Combinatorial Optimization. Prentice-Hall, New Jersey
40. Punnen AP (2002) The Traveling Salesman Problem: Aplications, Formulations and Variations. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 1–28
41. Punnen AP, Margot F, Kabadi S (2003) TSP heuristics: domination analysis and complexity. Algorithmica 35:111–127
42. Rao S, Smith W (1998) Approximating geometric graphs via "spanners" and "banyans". Proc. 30th Ann. ACM Symp. Theory Comput. ACM, pp 540–550
43. Rego C, Glover F (2002) Local Search and Metaheuristics. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 309–368
44. Rego C, Glover F, Gamboa D, Osterman C: A Doubly-Rooted Stem-and-Cycle Ejection Chain Algorithm for Asymmetric Traveling Salesman Problems. submitted
45. Trevisan L (1997) When Hamming meets Euclid: the approximability of geometric TSP and MST. In: Proc. 29th ACM Symp. Theory Comput. ACM, pp 21–39
46. Zemel E (1981) Measuring the quality of approximate solutions to zero-one programming problems. Math Oper Res 6:319–332

Tucker Homogeneous Systems of Linear Relations

THS

KEES ROOS
Department ITS/TWI/SSOR,
Delft University Technol., AJ Delft, Netherlands

MSC2000: 15A39, 90C05

Article Outline

Keywords
See also
References

Keywords

Inequality systems; Homogeneous dual systems; Theorem of the alternative; Transposition theorem; Strict complementarity

In a sequence of path-breaking papers A.W. Tucker and A.J. Goldman systematically investigated the relation between the theory of linear programming (cf. ► Linear programming) on the one hand, and *theorems of the alternative* (cf. ► Linear optimization: Theorems of the alternative) on the other hand [3,4,5,12]. In these papers they develop a comprehensive theory, covering many old and new results, often with new proofs. Thus they sharpen and consolidate the classical theorems of the alternative of J. Farkas [2], P. Gordan [6], E. Stiemke [11] and T.S. Motzkin [8] and the duality theory for linear optimization as first developed by G.B. Dantzig and J. von Neumann and O. Morgenstern. New is the emphasis they put on the property of *complementary slackness*. They derive the above results from properties of *homogeneous systems* of linear equality and inequality relations.

In its most general form such a so-called *dual system* consists of two systems, as follows:

$$\begin{cases} u \text{ unrestricted} \\ v \geq 0 \\ A^\top u + C^\top v \geq 0 \\ B^\top u + D^\top v = 0 \end{cases} \qquad \begin{cases} -Ax - By = 0 \\ -Cx - Dy \geq 0 \\ x \geq 0 \\ y \text{ unrestricted} \end{cases} \tag{1}$$

The matrices and vectors in (1) are such that all expressions are well-defined, in particular the matrices A and C have the same number of columns and similarly for the matrices B and D. In the left system the variables are the entries in the vectors y and v, and in the right system these are the entries in the vectors x and z. Note that the lines in (1) define a natural one-to-one correspondence between the variables in one system and the inequalities in the other system. Also, if a relation is of equality type then the corresponding variable is unrestricted (or free) and if it is of inequality type then the corresponding variable is nonnegative.

One easily verifies that any solution of (1) will satisfy

$$u^\top(-Ax - By) = 0, \tag{2}$$

$$v^\top(-Cx - Dy) \geq 0, \tag{3}$$

$$x^\top(A^\top u + C^\top v) \geq 0, \tag{4}$$

$$y^\top(B^\top u + D^\top v) = 0. \tag{5}$$

Adding (2) and (4), and also (3) and (5), one gets

$$-u^\top By + v^\top Cx \geq 0, \quad u^\top By - v^\top Cx \geq 0.$$

So $u^\top By = v^\top Cx$. Combining this with (2) and (5), one obtains

$$-u^\top Ax = u^\top By = v^\top Cx = -v^\top Dy.$$

This implies that (3) and (4) hold with equality:

$$v^\top(-Cx - Dy) = 0, \tag{6}$$

$$x^\top(A^\top u + C^\top v) = 0. \tag{7}$$

These relations are called the *complementary slackness relations*. They imply that if one of the nonnegative variables is positive then the corresponding inequality in the system necessarily holds with equality.

Note that it is not excluded that a nonnegative variable is zero and the corresponding inequality in the system holds with equality. In general this may certainly occur. For example, the trivial solution $x = y = u = v = 0$ has this behavior. The main result in [12, Thm. 4], however, states that there exists a solution of (1) with the property that a nonnegative variable is positive if and only if the corresponding inequality in the system holds with equality. Such a solution is called *strictly complementary* and can be characterized by the fact that it satisfies the *strictly complementary conditions*:

$$v - Cx - Dy > 0, \tag{8}$$

$$x + A^\top u + C^\top v > 0. \tag{9}$$

In [12] Tucker proves this result in a number of steps. Only the first step is nontrivial; the other steps consist of rather elementary algebraic arguments.

In the first step, he considers the simple dual system

$$A^\top u \geq 0, \quad Ax = 0, \quad x \geq 0. \tag{10}$$

Adapting arguments of D. Gale, in an unpublished proof of the *fundamental theorem* of H. Weyl [14] – that the convex hull of finitely many halflines is the intersection of finitely many halfspaces – he shows the existence of a solution of (10) such that the first coordinate of the vector $x + A^\top u$ is positive. This result is basic for the rest of Tucker's paper [12]. As Tucker shows, it already implies Farkas' lemma and, as he remarks, it recurs in geometric form in [4] as the theorem stating that 'a polyhedral cone is the polar of its polar' and in [3] as the separation theorem for a polyhedral convex cone and an individual vector. Tucker's proof exploits only algebraic arguments and uses induction to the number of columns of A. It may be noted that he could also have used Farkas' lemma (cf. ► Farkas lemma). Because if there does not exist a solution with $x_1 > 0$, then writing $x_1 = e_1^\top x$, where e_1 denotes the first unit vector, the system $Ax = 0$, $x \geq 0$, $-e_1^\top x < 0$ does not have a solution; then Farkas' lemma states that the system $A^\top z \leq -e_1$ has a solution. Hence, with $u = -z$, one has a solution of (10) such that the first coordinate of $x + A^\top u$ is positive.

Of course, there is nothing special with the first coordinate of $x + A^\top u$. For each of the other coordinates one can also obtain solutions x and u such that this particular coordinate of $x + A^\top u$ is positive. By adding these solutions one gets a solution of (10) all of whose coordinates are positive, i. e., such that

$$x + A^\top u > 0. \tag{11}$$

Thus the main result has now been proved for the special case of system (10). At this stage Tucker shows that the Stiemke and Gordan transposition theorems easily follow. Indeed, if there is no u such that $A^\top u \neq 0$ then there must exist an $x > 0$, with $Ax = 0$, which is *Stiemke's theorem*; and if there is no nonzero $x \geq 0$ such that $Ax = 0$ then there must exist a u such that $A^\top u > 0$, which is *Gordan's theorem*.

When applying the above result with the matrix A replaced by $(A\ B\ C\ -C)$ it immediately follows that the system

$$A^\top u \geq 0, \quad B^\top u \geq 0, \quad C^\top u = 0, \tag{12}$$

$$Ax + By + Cz = 0, \quad x \geq 0, \quad y \geq 0 \tag{13}$$

has a solution such that

$$x + A^\top u > 0, \quad v + B^\top u > 0.$$

Hence, if every solution u of (12) satisfies $A^\top$ u = 0 then (13) must have a solution with $x > 0$. By the complementary slackness property, each solution will satisfy

$x^\top A^\top u = 0$. Therefore, either the system

$$A^\top u \geq 0, \quad A^\top u \neq 0, \quad B^\top u \geq 0, \quad C^\top u = 0$$

has a solution or the system

$$Ax + By + Cz = 0, \quad x > 0, \quad y \geq 0$$

has a solution, but not both. This result is known as *Tucker's transposition theorem* [7]. On the other hand, and in a similar way, it follows that either the system

$$A^\top u \geq 0, \quad B^\top u > 0, \quad C^\top u = 0$$

or the system

$$Ax + By + Cz = 0, \quad x \geq 0, \quad y \geq 0, \quad y \neq 0$$

has a solution, but not both. This is *Motzkin's transposition theorem*. When C is vacuous, these results are also known as 'theorems of the alternative' for the pair A, B of matrices [1].

When replacing the matrix A in (10) by $(I\ K)$ one obtains that the system

$$K^\top u \geq 0, \quad -Kx \geq 0, \quad u \geq 0, \quad x \geq 0 \tag{14}$$

has a solution such that

$$u - Kx > 0, \quad x + K^\top u > 0.$$

Tucker notes that by applying this result to the pay-off matrix of a 'fair' zero-sum two-person game, one may easily derive a well-known theorem of von Neumann and Morgenstern [9]. One easily sees that the following alternatives hold:

$$\begin{aligned} &K^\top u \neq 0 \quad \text{or} \quad x > 0, \\ &K^\top u > 0 \quad \text{or} \quad x \neq 0, \end{aligned} \tag{15}$$

$$\begin{aligned} &u > 0 \quad \text{or} \quad -Kx \neq 0, \\ &u \neq 0 \quad \text{or} \quad -Kx > 0. \end{aligned} \tag{16}$$

The above alternatives are mutually exclusive because $u^\top Kx = 0$ for all solutions of (14); (15) and (16) are dual forms of the theorem of the alternative for matrices in [9]. It also follows that if the system $-Kx \geq 0$, $x \geq 0$ has no nonzero solution then the system $K^\top u > 0$, $u > 0$ has a solution; this result is due to J. Ville [13].

The existence of a strictly complementary solution of the most general dual system, as given by (1), now straightforwardly follows by replacing the matrix K in (14) by the matrix

$$\begin{pmatrix} -A & -B & B \\ A & B & -B \\ C & D & -D \end{pmatrix}.$$

This yields the existence of nonnegative vectors u_1, u_2, v, (x, y_1, and y_2 such that

$$\begin{aligned} -A^\top u_1 + A^\top u_2 + C^\top v &\geq 0, \\ -B^\top u_1 + B^\top u_2 + D^\top v &\geq 0, \\ B^\top u_1 - B^\top u_2 - D^\top v &\geq 0, \\ x - A^\top u_1 + A^\top u_2 + C^\top v &> 0, \end{aligned}$$

and

$$\begin{aligned} A^\top x + B^\top y_1 - B^\top y_2 &\geq 0, \\ -A^\top x - B^\top y_1 + B^\top y_2 &\geq 0, \\ -C^\top x - D^\top y_1 + D^\top y_2 &\geq 0, \\ v - C^\top x - D^\top y_1 + D^\top y_2 &> 0. \end{aligned}$$

Take $u = u_2 - u_1$ and $y = y_1 - y_2$. Then

$$\begin{aligned} A^\top u + C^\top v &\geq 0, \\ B^\top u + D^\top v &= 0, \\ x + A^\top u + C^\top v &> 0, \end{aligned}$$

and

$$\begin{aligned} A^\top x + B^\top y &\geq 0, \\ -A^\top x - B^\top y &= 0, \\ v - C^\top x - D^\top y &> 0, \end{aligned}$$

showing that u, v, x and y solve the dual system (1), and also satisfy the strictly complementarity conditions (8) and (9).

An interesting and important special case of the dual system (14) occurs when the matrix K is skew-symmetric. Then the conditions on x and u are the same and the system becomes a *selfdual system*. Taking $z = x + u$ and replacing K by $K^\top$ it then follows that there exists a vector z such that

$$Kz \geq 0, \quad z \geq 0, \quad z + Kz > 0. \tag{17}$$

In fact, the result for this special case is strong enough to recover the more general result for the system (14): if K is an arbitrary matrix then one simply

applies (17) to the skew-symmetric matrix

$$\begin{pmatrix} 0 & -K \\ K^\top & 0 \end{pmatrix}.$$

The existence of a strictly complementary solution to a selfdual system is used in [5] 'as an omnibus means of proving the basic duality and existence theorems of linear programming'; a new proof is given [10], where this result is used for the same purpose. The derivation of the duality theorem goes as follows.

For a given matrix A and column vectors b and c of appropriate size consider the pair of *dual linear programs*

$$\text{(P)} \quad \begin{cases} \min & c^\top x \\ \text{s.t.} & Ax \geq b \\ & x \geq 0 \end{cases}$$

$$\text{(D)} \quad \begin{cases} \max & b^\top y \\ \text{s.t.} & A^\top y \leq c \\ & y \geq 0. \end{cases}$$

If x is feasible for (P) and y for (D) then

$$c^\top x \geq y^\top Ax \geq b^\top y.$$

This is known as the *weak duality result* for linear optimization. The *strong duality result* states that if one of the two problems (P) and (D) has an optimal solution, then so has the other and the optimal values coincide.

Define the skew-symmetric matrix K by

$$K := \begin{pmatrix} 0 & A & -b \\ -A^\top & 0 & c \\ b^\top & -c^\top & 0 \end{pmatrix}.$$

Applying (17) to K one obtains nonnegative vectors y and x and a nonnegative scalar t such that

$$Ax - tb \geq 0, \tag{18}$$

$$-A^\top y + tc \geq 0, \tag{19}$$

$$b^\top y - c^\top x \geq 0, \tag{20}$$

$$y + Ax - tb > 0, \tag{21}$$

$$x - A^\top y + tc > 0, \tag{22}$$

$$t + b^\top y - c^\top x > 0. \tag{23}$$

Recall that these relations imply the complementarity relations, which are given by

$$y^\top (Ax - tb) = 0, \tag{24}$$

$$x^\top (-A^\top y + tc) = 0, \tag{25}$$

$$t(b^\top y - c^\top x) = 0. \tag{26}$$

Note that (24) and (25) are equivalent to

$$y^\top Ax = t\, b^\top y = t\, c^\top x \tag{27}$$

and these relation imply (26).

The relations (18)–(23) are homogeneous in t, x and y. Hence, a solution with $t > 0$ exists if and only if a solution with $t = 1$ exists. Thus two cases have to be distinguished: either $t = 0$ or $t = 1$.

If $t = 0$ then one has

$$Ax \geq 0, \quad A^\top y \leq 0, \quad b^\top y - c^\top x > 0.$$

Thus one has either $b^\top y > 0$ or $c^\top x < 0$ or both. First consider the case $b^\top y > 0$. Then (P) cannot have a feasible solution x, for this would yield the contradiction

$$0 \geq x^\top (A^\top y) = (Ax)^\top y \geq b^\top y > 0.$$

Moreover, if (D) has a feasible solution y', then $A^\top y \leq 0$ implies that $y' + \alpha y$ is feasible for (D) for any nonnegative α. From

$$b^\top y' + \alpha y = b^\top y' + \alpha b^\top y,$$

it follows that the dual objective value can attain arbitrarily large valus, since $b^\top y > 0$. The dual problem (D) is *unbounded* in this case. Thus, if $b^\top y > 0$, then (P) is infeasible and (D) can be either infeasible or unbounded.

If $c^\top x < 0$, similar arguments can be used to show that (D) is infeasible and (P) can be either infeasible or unbounded.

If $t = 1$ then x is feasible for (P) and y for (D), whereas $c^\top x = b^\top y$, proving that x is an optimal solution for (P) and y is an optimal solution for (D). Hence, the duality theorem for linear optimization has been proved.

The above approach to the duality theory for linear optimization yields a little more than the classical approach, namely; if (P) and (D) are feasible, then there exist strictly complementary optimal solutions x and y (5, [Coroll. 2A]). This is due to (21) and (22) which give (for $t = 1$):

$$y + (Ax - b) > 0,$$
$$x + (c - A^{\top} y) > 0.$$

See also

- ▶ Farkas Lemma
- ▶ Linear Optimization: Theorems of the Alternative
- ▶ Linear Programming
- ▶ Motzkin Transposition Theorem

References

1. Antosiewicz HA (1955) A theorem of the alternative for pairs of matrices. Pacific J Math 5:641–642
2. Farkas J (1902) Theorie der Einfachen Ungleichungen. J Reine Angew Math 124:1–27
3. Goldman AJ (1956) Resolution and separation theorems for polyhedral convex sets. In: Kuhn HW, Tucker AW (eds) Linear Inequalities and Related Systems. Ann Math Stud. Princeton Univ. Press, Princeton, pp 41–52
4. Goldman AJ, Tucker AW (1956) Polyhedral convex cones. In: Kuhn HW, Tucker AW (eds) Linear Inequalities and Related Systems. Ann Math Stud. Princeton Univ. Press, Princeton, pp 19–40
5. Goldman AJ, Tucker AW (1956) Theory of linear programming. In: Kuhn HW, Tucker AW (eds) Linear Inequalities and Related Systems. Ann Math Stud. Princeton Univ. Press, Princeton, pp 53–97
6. Gordan P (1873) Über die Auflösung Linearer Gleichungen mit Reelen Coefficienten. Math Ann 6:23–28
7. Mangasarian OL (1994) Nonlinear programming. Classics Appl Math, no 10. SIAM, Philadelphia
8. Motzkin TS (1936) Beiträge zur Theorie der Linearen Ungleichungen. PhD Thesis Azriel, Jerusalem
9. Neumann J von, Morgenstern O (1953) Theory of games and economic behavior. third, Princeton Univ. Press, Princeton
10. Roos C, Terlaky T, Vial J-P (1997) Theory and algorithms for linear optimization. An interior approach. Wiley, New York
11. Stiemke E (1915) Über Positive Lösungen Homogener Linearer Gleichungen. Math Ann 76:340–342
12. Tucker AW (1956) Dual systems of homogeneous linear relations. In: Kuhn HW, Tucker AW (eds) Linear Inequalities and Related Systems. Ann Math Stud. Princeton Univ. Press, Princeton, pp 3–18
13. Ville J (1938) Sur la théorie générale des jeux où intervient l'habileté des joueurs. In: Ville J (ed) Applications aux Jeux de Hasard. Gauthier-Villars, pp 105–113
14. Weyl H (1935) Elementare Theorie der konvexen Polyeder. Comment Math Helvetici 7:290–306 English translation: (1950) The elementary theory of convex polyhedra. In: Kuhn W, Tucker AW (eds) Contribution to the Theory of Games I. Princeton Univ. Press, Princeton, pp 3–18.

Turnpike Theory: Stability of Optimal Trajectories

Musa Mammadov (Mamedov)
Centre for Informatics
and Applied Optimization School of Information
Technology and Mathematical Sciences,
University of Ballarat, Ballarat, Australia

MSC2000: 49J24, 35B40, 37C70

Article Outline

Keywords and Phrases
Introduction
Definitions
Turnpike Theorems for Terminal Functionals
Functional $\liminf_{t\to\infty} u(x(t))$
Functional $\liminf_{t\to\infty} u(x(t), \dot{x}(t))$
Turnpike Theorems for Integral Functionals
Convex Problems
Other Results
See also
References

Keywords and Phrases

Turnpike property; Differential inclusion; Asymptotical stability; Stationary points; Convex sets

Introduction

This article considers the problem of asymptotical stability of optimal trajectories of dynamical systems described by differential inclusions. In the literature, the results obtained in this area are called "turnpike theorems."

Turnpike theory has many applications in economics and engineering. We refer to [9,17,18,25] for more detailed information about this theory and its various applications.

The first result in this area was obtained by J. von Neumann, in 1945. However, the main meaning of this result that led to turnpike property was discovered by Paul A. Samuelson, in 1948–1949, who also introduced this terminology. These results were obtained for optimal trajectories of models of economic dynamics determined by convex processes (the von Neumann model). A clearer description of this property was provided by Dorfman et al. [1] in the Chap. "Efficient Programs of Capital Accumulation" of *Linear Programming and Economic Analysis*. The following is the famous quote from [1], p. 331, that describes the meaning of the turnpike property:

"Thus in this unexpected way, we have found a real normative significance for steady growth – not steady growth in general, but maximal von Neumann growth. It is, in a sense, the single most effective way for the system to grow, so that if we are planning long-run growth, no matter where we start and where we desire to end up it will pay in the intermediate stages to get into a growth phase of this kind. It is exactly like a turnpike paralleled by a network of minor roads. There is a fastest route between any two points; and if the origin and destination are close together and far from the turnpike, the best route may not touch the turnpike. But if origin and destination are far enough apart, it will always pay to get on to the turnpike and cover distance at the best rate of travel, even if this means adding a little mileage at either end. The best intermediate capital configuration is one which will grow most rapidly, even if it is not the desired one, it is temporarily optimal".

In a simple case, when the trajectories of the system under consideration are uniformly bounded, the following formulation could be considered as a turnpike property.

Let $\{x_T(t)\}$ be a set of optimal trajectories defined on the intervals $[0, T]$, $T > 0$, and x^* be a fixed point. In the applications, x^* is usually an optimal stationary point.

Turnpike property: *For any $\varepsilon > 0$ there is a finite number $K_\varepsilon > 0$ such that for all $T > 0$ the following inequality holds*:

$$\text{meas}\{t \in [0, T] \ : ||x_T(t) - x^*|| \geq \varepsilon\} \leq \ K_\varepsilon \ .$$

The meaning of this statement is as follows: the time that optimal trajectories spend outside the ε-neighborhood of x^* is bounded by some finite number K_ε that does not depend on T and optimal trajectories.

If the system is considered on the interval $[0, \infty)$, the turnpike property can be formulated as a convergence of all optimal trajectories to x^*.

Historically, the turnpike theory was first studied for optimal control problems in discrete time. The general formulation of these problems can be presented as

$$\begin{aligned} &\text{Maximize } J(\{x_t\}_{t=1}^T) \, ; \quad \text{subject to } x_{t+1} \in a(x_t) \, , \\ &\quad t = 1, \ldots, T \, . \end{aligned} \tag{1}$$

Set-valued mapping $a : \Omega \to \Pi_c(\mathbb{R}^n)$ is usually assumed to be continuous in the Hausdorff metric. Here $\Omega \subset \mathbb{R}^n$ (in a particular case $\Omega = \mathbb{R}^n$) and $\Pi_c(\mathbb{R}^n)$ stands for the set of all compact subsets of $\mathbb{R}^n$. The graph of the mapping a is defined by

$$\text{graph } \ a = \{(x, y) : x \in \Omega, \ y \in a(x)\} \, .$$

The objective function - functional $J(\{x_t\}_{t=1}^T)$ can be defined in different forms. It is usually defined by some utility function $u(x_t, x_{t-1})$. In some cases it is assumed that $u(x_t, x_{t-1}) = u(x_t)$; that is, utility function u does not depend on x_{t-1}.

Terminal and integral type functionals with and without discount factors are most commonly used in the literature.

Many approaches have been developed to study the turnpike property for different classes of problems (1). Good surveys of these approaches developed by the 1970s can be found in [9,17]. The main achievement of these approaches can be summarized as follows. The turnpike property is valid under the following convexity assumptions:

$$\begin{aligned} &\text{graph } a \text{ is convex, and the function } u \\ &\text{is strictly concave} \, . \end{aligned} \tag{2}$$

We say that problem (1) is convex if condition (2) holds. Many problems arising in economics are convex problems. Thus, the methods developed for discrete systems in the form of (1) can be successfully applied to such piratical problems.

The study of turnpike property for continuous systems started in the 1970s. It turned out that the methods developed for discrete systems were not applicable for continuous systems; thus, new methods were required. In order to prove the turnpike property for continuous systems, together with the convexity assumptions (2),

very restrictive additional assumptions were used. Since then, to prove turnpike property assuming only convexity conditions (2) became a very difficult and challenging problem. This article gives a brief survey of the results obtained in this area.

Definitions

Let $x^0 \in \Omega$ be a given initial point. We will consider the following optimal control problem

$$\text{Maximize } J(x(\cdot)); \quad \text{subject to } \dot{x} \in a(x)\,. \tag{3}$$

We assume that set-valued mapping $a : \Omega \to \Pi_c(R^n)$ has compact images and is continuous in the Hausdorff metric. Here $\Omega \subset R^n$ or $\Omega = R^n$.

Definition 1 An absolutely continuous function $x(\cdot)$ is called a trajectory defined on the interval $[0, T]$, if for almost all $t \in [0, T]$ the inclusion $\dot{x}(t) \in a(x(t))$ holds.

Definition 2 $x \in R^n$ is called a stationary point if $0 \in a(x)$.

Stationary points play an important role in the study of asymptotical behavior of optimal trajectories. Throughout this article, we denote the set of stationary points by M:

$$M = \{x \in \Omega : 0 \in a(x)\}\,.$$

If mapping $a(x)$ is continuous, then M is a compact set. We note that this set may be empty.

We will consider different classes of functionals $J(x(\cdot))$. The following are the most commonly used functionals considered in the literature:

$$\liminf_{t\to\infty} u(x(t)), \quad \liminf_{t\to\infty} u(x(t), \dot{x}(t))\,; \tag{4}$$

$$\int_0^T u(x)\,\mathrm{d}t, \qquad \int_0^T u(x, \dot{x})\,\mathrm{d}t\,; \tag{5}$$

$$\int_0^\infty e^{-rt}\,u(x)\,\mathrm{d}t, \quad \int_0^\infty e^{-rt}\,u(x, \dot{x})\,\mathrm{d}t\,. \tag{6}$$

In (4), $\liminf_{t\to\infty} u(x(t), \dot{x}(t))$ is taken over the points t where $\dot{x}(t)$ exists.

Several approaches have been developed to study turnpike property for continuous systems. These approaches use the Hamiltonian of problem (3) and the necessary conditions of optimality in various versions. We specially mention the approaches developed by Rockafellar [20,21] and Scheinkman [22,23]. They considered a convex problem with integral functionals (with and without a discount factor). The additional assumptions (together with convexity) that were used in these approaches involve the derivatives of the Hamiltonian. That is why these assumptions are very difficult to check in practical problems.

Among the other approaches developed for problem (3), we mention the results of Gusev and Yakubovich [4,5], Panasyuk and Panasyuk [18] and Zelikina [26]. They considered some special classes of optimal control problems defined by differential equations and the turnpike property was established under some restrictive assumptions. The approaches developed in [4,5,26] were based on Pontryagin's maximum principle. The results obtained by Panasyuk and Panasyuk [18] found some interesting applications in engineering where the corresponding restrictive assumptions hold.

Summarizing these results, we observe that the techniques developed for continuous systems in the form of (3) have not been successful in the establishment of turnpike property for convex problems. The additional assumptions were too restrictive in terms of application to a wide range of practical problems.

However, it was the common opinion that this flaw was due to the drawbacks of the techniques developed. As mentioned above, these techniques were based on the necessary conditions of optimality. We think that the use of necessary conditions (for example, Pontryagin's maximum principle) generates serious difficulties in the proof of turnpike property. New techniques were required that could allow us to avoid the use of necessary conditions.

We note that there are many studies in the literature that aimed to study the behavior of optimal trajectories for different systems. We refer to [25] for more information and references. For example, Zaslavski [25] obtained important results for variational problems regarding the turnpike behavior of optimal trajectories.

In the following we present some results obtained in [10,11,12,13,15]. These studies introduced new techniques for problem (3) that did not use necessary conditions. In this way, we succeeded in establishing the turnpike property not only for convex problems but also for some classes of nonconvex problems.

Turnpike Theorems for Terminal Functionals

Functional (4) was first introduced by Lyapunov [8], in 1983, for discrete systems. It can be considered as an analog of the terminal functional reformulated for the interval $(0, \infty)$. It turned out that this functional was very convenient in terms of turnpike property.

The turnpike property for this functional was established in [10,11] even for nonconvex problems in continuous time. These results were obtained on the basis of new techniques that had been developed. They applied to some nonconvex practical optimal control problems. Some applications of these techniques to discrete time systems can be found in [14,16,19].

We consider the system $\dot{x} \in a(x)$ on the interval $[0, \infty)$.

Definition 3 Trajectory $x(t)$ is called optimal if $J(x(\cdot)) \geq J(\tilde{x}(\cdot))$ holds for all trajectories $\tilde{x}(t)$ starting from the same initial state: $\tilde{x}(0) = x(0)$.

Definition 4 The set

$$\mathfrak{M} = \{x \in \mathbb{R}^n : 0 \in \operatorname{co} a(x)\}$$

is called the set of generalized stationary points.

Here "co" stands for the convex hull. Clearly $M \subset \mathfrak{M}$. We will also use the notation

$$a(A) = \cup_{x \in A} a(x)\,.$$

Functional $\liminf_{t\to\infty} u(x(t))$

In this section we consider the problem

$$\dot{x} \in a(x), \quad J(x(\cdot)) = \lim_{t\to\infty} \inf u(x(t)) \to \max \quad (7)$$

The main condition that will be imposed on mapping a is the following:

Condition A: *Given any set* $A \subset \mathbb{R}^n$

$$\begin{aligned} &\text{if} \quad 0 \in \operatorname{co} a(A) \quad \text{then} \quad 0 \in \operatorname{co} a(x) \\ &\qquad\qquad\qquad\qquad\qquad\qquad \text{for some} \quad x \in A\,. \end{aligned} \quad (8)$$

If mapping a has convex images $a(x)$ for all x, then (8) can be reformulated as

$$\text{if} \quad 0 \in \operatorname{co} a(A) \quad \text{then} \quad 0 \in a(\operatorname{co} A)\,. \quad (9)$$

We denote by $\mathfrak{A}$ the class of continuous set-valued mappings a satisfying Condition A. The following lemma shows that $\mathfrak{A}$ contains the class of mappings having a convex graph.

Lemma 1 *If graph a is convex then Condition A holds.*

Denote $J^* = \max_{x\in\mathfrak{M}} u(x)$. Let $x^* \in \mathfrak{M}$ be a point for which $u(x^*) = J^*$. If set $\mathfrak{M}$ is convex and $u(x)$ is strictly concave, then point x^* is unique.

The main results are combined in the following theorems.

Theorem 1 *Assume that $a \in \mathfrak{A}$ and function $u(x)$ is concave. Then the inequality $J(x(\cdot)) \leq J^*$ holds for all trajectories $x(t)$.*

Theorem 2 *Assume that $a \in \mathfrak{A}$, function $u(x)$ is strictly concave and $\mathfrak{M}$ is convex and compact. If trajectory $x(t)$ is such that $J(x(\cdot)) = J^*$, then $\lim_{t\to\infty} x(t) = x^*$.*

If $J(x(\cdot)) = J^*$, then from the first theorem it follows that trajectory $x(t)$ is optimal. The second theorem provides the turnpike property: all optimal trajectories satisfying $J(x(\cdot)) = J^*$ converge to x^*.

It is important to note that, in this way, the turnpike property is established for a special class $\mathfrak{A}$ of *nonconvex* set-valued mappings a. This class contains mappings a having convex graphs. Therefore, for convex problem (7) the turnpike property is true without any additional assumptions.

Functional $\liminf_{t\to\infty} u(x(t), \dot{x}(t))$

Now we consider the problem

$$\dot{x} \in a(x), \quad J_1(x(\cdot)) = \lim_{t\to\infty} \inf u(x(t), \dot{x}(t)) \to \max \quad (10)$$

The main condition in this case is the following:

Condition B: *Given any set* $Q \subset$ graph a

$$\begin{aligned} &\text{if} \quad \operatorname{co} Q \cap (\mathbb{R}^n \times 0) \quad \neq \quad \emptyset \\ &\text{then} \quad \operatorname{co} Q \cap (\mathfrak{M} \times 0) \quad \neq \quad \emptyset\,. \end{aligned} \quad (11)$$

We denote by $\mathfrak{B}$ the class of continuous set-valued mappings a satisfying Condition B. We have the following properties:

Lemma 2 *If graph a is convex, then Condition B holds. If Condition A holds then Condition B holds too.*

Denote the set of continuous set-valued mapping with a convex graph by $\mathfrak{C}$. From this lemma we have the

relation

$$\mathfrak{C} \subset \mathfrak{B} \subset \mathfrak{A} .$$

Denote $J_1^* = \max_{x \in \mathfrak{M}} u(x, 0)$, where $\mathfrak{M}$ is defined in Definition 4.

Let $x^* \in \mathfrak{M}$ be a point for which $u(x^*, 0) = J_1^*$. If set $\mathfrak{M}$ is convex and compact and $u(x,y)$ is strictly concave, then point x^* is unique.

Similar to Theorems 1 and 2 we have the following results.

Theorem 3 *Assume that $a \in \mathfrak{B}$ and function $u(x,y)$ is concave. Then the inequality $J_1(x(\cdot)) \le J_1^*$ holds for all trajectories $x(t)$.*

Theorem 4 *Assume that $a \in \mathfrak{B}$, function $u(x,y)$ is strictly concave and $\mathfrak{M}$ is convex and compact. If trajectory $x(t)$ is such that $J_1(x(\cdot)) = J_1^*$ then $\lim_{t\to\infty} x(t) = x^*$.*

Therefore, for problem (10), the turnpike property is established for a special class $\mathfrak{B}$ of *nonconvex* set-valued mappings a. This class contains mappings a having convex graphs.

Now we present some interesting examples related to Theorems 1–4 and classes $\mathfrak{A}$ and $\mathfrak{B}$.

Example 1 Let A be an $n \times n$ matrix, B be an $n \times r$ matrix and $V \subset \mathbb{R}^r$ be a closed set (not necessarily convex). Then, the mapping defined by

$$a(x) = \{Ax + Bv : v \in V\}$$

belongs to class $\mathfrak{B}$ (and, consequently, to $\mathfrak{A}$).

Example 2 Let $x = (x_1, x_2) \in \mathbb{R}^2$ and $a(x) = \{-(x_1, x_2), (x_1, 0)\}$. It is not difficult to show that $a \in \mathfrak{B}$.

The following example shows that, in Theorem 4, the convergence $\dot{x}(t) \to 0$ may not be true while $x(t) \to x^*$.

Example 3 Let $x \in \mathbb{R} u(x, y) = \sqrt{x} + \sqrt{y+1}$, and $a(x) = [-1, 1]$ if $x \in [0, 1]$, $a(x) = 1$ if $x > 1$. We have $\mathfrak{M} = [0, 1]$ and $J_1^* = 2$. Consider trajectory $x(t)$ defined as follows: on each interval $[m, m+1]$, $m = 1, 2, \cdots$, we set $x(t) = t - m$ if $t \in [m, m + 1/(2m)]$, and $x(t) = -t/(2m-1) + (m+1)/(2m-1)$ if $t \in [m + 1/(2m), m+1]$. It is not difficult to show that $J_1(x(\cdot)) = 2 = J_1^*$ (i. e., turnpike property is true). However, $\dot{x}(t)$ does not converge to 0.

Turnpike Theorems for Integral Functionals

In this section we consider problem (3) with integral functionals. For the sake of simplicity, we will only consider the following problem:

$$\begin{aligned} &\dot{x} \in a(x), \quad x(0) = x^0; \\ &J_T(x(\cdot)) = \int_0^T u(x)\, \mathrm{d}t \to \max \end{aligned} \tag{12}$$

We denote by X_T the set of trajectories defined on the interval [0,T]. Let

$$J_T^* = \sup_{x(\cdot) \in X_T} J_T(x(\cdot)) .$$

Definition 5 Trajectory $x(\cdot)$ is called optimal if $J_T(x(\cdot)) = J_T^*$ and is called ξ-optimal ($\xi > 0$) if

$$J_T(x(\cdot)) \ge J_T^* - \xi .$$

Definition 6 $x^* \in M$ is called an optimal stationary point if

$$u(x^*) = u^* \triangleq \max_{x \in M} u(x) .$$

Here M is the set of all stationary points. We assume set M is not empty.

The turnpike theorem is proved under two main conditions: Conditions M and H given below. The first condition concerns the existence of "good" trajectories starting from the initial state x^0. The second is the main condition that provides the turnpike property.

Condition M: *There exists $b < +\infty$ such that for every $T > 0$ there is a trajectory $x(\cdot) \in X_T$ satisfying the inequality*

$$J_T(x(\cdot)) \ge u^* T - b .$$

Set

$$\mathcal{B} = \{x \in \Omega : \ u(x) \ge u^*\} .$$

We fix $p \in R^n$, $p \neq 0$, and define a support function

$$c(x) = \max_{y \in a(x)} py .$$

Here the notation py means the scalar product of the vectors p and y. By $|c|$ we will denote the absolute value of c. We also define the function

$$\varphi(x, y) = \frac{u(x) - u^*}{|c(x)|} + \frac{u(y) - u^*}{c(y)} .$$

Condition H: *There exists a vector* $p \in R^n$ *such that*
H1 $c(x) < 0$ *for all* $x \in \mathcal{B}, x \neq x^*$;
H2 *there exists point* $\tilde{x} \in \Omega$ *such that* $p\tilde{x} = px^*$ and $c(\tilde{x}) > 0$;
H3 *for all points* x, y, for which $px = py, c(x) < 0$, $c(y) > 0$, *the inequality* $\varphi(x, y) < 0$ *holds. Moreover, if*

$$x_k \to x^*,\ y_k \to y' \neq x^*,$$
$$px_k = py_k, c(x_k) < 0, c(y_k) > 0\,,$$
$$\text{then} \quad \limsup_{k\to\infty} \varphi(x_k, y_k) < 0\,.$$

Now we formulate the main result.

Theorem 5 *Assume that Conditions M and H are satisfied and the optimal stationary point* x^* *is unique. Then:*
1) There exists $C < +\infty$ *such that*

$$\int_0^T [u(x(t)) - u^*]\,\mathrm{d}t \le C$$

for all $T > 0$ *and all trajectories* $x(\cdot) \in X_T$.
2) For every $\varepsilon > 0$ *there exists* $K_{\varepsilon,\xi} < +\infty$ *such that*

$$meas\{t \in [0, T] : ||x(t) - x^*|| \ge \varepsilon\} \le K_{\varepsilon,\xi}$$

for all $T > 0$ *and all* ξ*-optimal trajectories* $x(\cdot) \in X_T$.
3) If $x(\cdot)$ *is an optimal trajectory and* $x(t_1) = x(t_2) = x^*$, *then* $x(t) = x^*$ *for all* $t \in [t_1, t_2]$.

This theorem has two major advantages compared with the results obtained by others, including [20,21,22,23]:

1. Theorem 5 does not use the Hamiltonian. It uses conditions that directly imposed on mapping a and function u. Thus, these conditions can be verified for a given particular problem.
2. The main condition in Theorem 5 is H3. It can be considered as a relation between mapping a and function u which provides the turnpike property.

We will see below that Conditions **H1** and **H3** hold if the graph of the mapping a is a convex set (in $R^n \times R^n$) and the function u is strictly concave. On the other hand Condition **H** may hold for mappings a having nonconvex graphs and for functions u that are not strictly concave. Therefore, Theorem 5 establishes turnpike property for nonconvex problems.

Convex Problems

Now we consider problem (12) assuming that graph a of the mapping a is a convex set and the function $u : \Omega \to \mathbb{R}$ is strictly concave. Let

$$J_T^* = \sup_{x(\cdot)\in X_T} J_T(x(\cdot))\,.$$

In this section, we present a result showing that Theorem 5 is valid for a convex problem without assuming Condition **H**. In particular, this means that the turnpike property is true for convex problem (12) without any restrictive additional assumptions.

We have the following result.

Lemma 3 *Assume that graph* a *is a compact set, function* u *is strictly concave and*

$$0 \in int\, a(\tilde{x}) \quad \text{for some} \quad \tilde{x} \in M\,. \tag{13}$$

Then Conditions ***H1*** *and* ***H3*** *hold.*

We note that Condition **H2** may not be satisfied even if condition (13) holds. This can be seen from the following example.

Example 4 Let $\Omega = [-1, 1] \subset R^1$ and $a(x) = [-1, \xi(x)]$,

where

$$\xi(x) = -\frac{4}{10}\left(x + \frac{1}{2}\right)^2 + \frac{1}{10}, \quad x \in [-1, +1]\,.$$

Consider the function $u(x) = 1 - (x - 1)^2$.

For this problem function u is strictly concave, the graph of the mapping a is a convex set. We have $M = [-1, 0]$, $u^* = \max_{x\in M} u(x) = 0$ and $x^* = 0$.

It is not difficult to observe that for the point $\tilde{x} = -1/2$ condition (13) holds.

Consider Condition **H**. We have $\mathcal{B} = [0, 1]$. Condition **H1** is satisfied for the points $p \in R^1$, $p > 0$.

Now we check Condition **H2**. Take any $p \in R^1$, $p \neq 0$. If $p\tilde{x} = px^*$, then $\tilde{x} = x^* = 0$ and

$$c(\tilde{x}) = \max_{y\in a(\tilde{x})} py = \max_{y\in[-1,0]} y = 0\,.$$

Therefore, Condition **H2** is not satisfied for any $p \in R^1$, $p \neq 0$. The main result of this section is the following

Theorem 6 *Assume that function u is strictly concave and Conditions* ***M*** *and (13) hold. Then an optimal stationary point x^* exists, is unique and all assertions of Theorem 5 are true.*

Condition (13) is important. Example 5 presented shows that if this condition does not hold, then Theorem 6 may be not true.

Example 5 Let $\Omega = [-1, 1] \subset R^1$,

$$a(x) = \begin{cases} \{-x^4 + v : v \in [x^4 - 1, 0]\} \text{ if } 0 \le x \le 1\,; \\ \{v : v \in [-1, 0]\} \text{ if } -1 \le x \le 0\,, \end{cases}$$

and $u(x) = 1 - (x-1)^2$. v is the control.

It is clear that function u is strictly concave, the graph of the mapping a is a convex set. We have $M = [-1, 0], u^* = \max_{x \in M} u(x) = 0$ and $x^* = 0$.

It is not difficult to observe that condition (13) is not satisfied. We will show that Theorem 6 is not true in this case.

We take an initial point $x^0 = 1$ and consider a trajectory corresponding to the control $v(t) = 0$. This trajectory can be calculated as a solution to the following differential equation:

$$\dot{x} = -x^4,\ x(0) = 1.$$

We have $x(t) = (3t + 1)^{-\frac{1}{3}}$. Clearly $0 \le x(t) \le 1$ and $x(t) \to 0$ as $t \to \infty$. Therefore, $x(t)$ is a trajectory. We have

$$\int_0^T (u(x(t)) - u^*)\mathrm{d}t = \int_0^T [1 - (x(t) - 1)^2]\mathrm{d}t \ge \int_0^T x(t)\mathrm{d}t = \int_0^T (3t + 1)^{-\frac{1}{3}}\mathrm{d}t \to +\infty\,,$$

as $T \to \infty$. Therefore, the first assertion of Theorem 6 is not true.

Other Results

Some generalizations of the results presented above, involving functionals in (5) and (6), can be found in [11,12,13,15]. Moreover the case when the optimal stationary point is not unique is also considered.

See also

► Statistical Convergence and Turnpike Theory

References

1. Dorfman R, Samuelson PA, Solow RM (1958) Linear Programming and Economic Analysis. McGraw Hill, New York
2. Filippov AF (1988) Differential Equations with Discontinuous Righthand Sides. Mathematics and its Applications (Soviet series). Kluwer, Dordrecht
3. Gale D (1967) On optimal development in a multisector economy. Rev Econ Stud 34:119
4. Gusev DE, Yakubovich VA (1983) Turnpike Theorem in the Problem of Continuous Optimization. Vestn Leningr Univ 1(2):2027
5. Gusev DE, Yakubovich VA (1983) Turnpike theorem in the problem of continuous optimization with phase restrictions. Syst Control Lett 3(4):221–226
6. Leizarowitz A (1985) Infinite horizon autonomous systems with unbounded cost. Appl Math Opt 13:19–43
7. Leizarowitz A (1989) Optimal trajectories on infinite horizon deterministic control systems. Appl Math Opt 19:11–32
8. Lyapunov AN (1983) Asymptotical Optimal Trajectories for Convex Mappings. Optim Model Syst Anal, VNIISI 9:74–80
9. Makarov VL, Rubinov AM (1977) Mathematical Theory of Economic Dynamics and Equilibria. Springer, New York
10. Mamedov MA (1985) Asymptotical optimal paths in models with environment pollution being taken into account. Optimization 36(53):101–112
11. Mamedov MA (1987) Asymptotical stability of optimal paths for some differential inclusions. In: On problems of uniformly approximations and asymptotical paths of dynamical systems. NGPI, Novosibirsk, pp 124–131
12. Mamedov MA (1992) Turnpike theorems in continuous systems with integral functionals. Russ Acad Sci Dokl Math 45(2):432–435
13. Mamedov MA (1993) Turnpike theorems for integral functionals. Russ Acad Sci Dokl Math 46(1):174–177
14. Mamedov MA, Pehlivan S (2001) Statistical cluster points and turnpike theorem in nonconvex problems. J Math Anal Appl 256:686–693
15. Mamedov MA (2003) Turnpike Theorem for Continuous-time Control Systems when Optimal Stationary Point is not unique. Abstr Appl Anal 2003(11):631–650
16. Mamedov MA, Borisov KY (1988) A simple model of economic grow and pollution control. Vestn Leningr Univ 5(5):120–124
17. McKenzie LW (1976) Turnpike theory. Econometrica 44:841–866
18. Panasyuk AI, Panasyuk VI (1986) Asymptotic turnpike optimization of control systems. Nauka i Technika, Minsk
19. Pehlivan S, Mamedov MA (2000) Statistical cluster points and turnpike. Optimization 48:93–106

20. Rockafellar RT (1973) Saddle points of Hamiltonian systems in convex problems of Lagrange. J Optim Theory Appl 12:367–390
21. Rockafellar RT (1976) Saddle points of Hamiltonian systems in convex problems having a nonzero discount rate. J Econ Theory 12:71–113
22. Scheinkman JA (1976) On optimal steady states of *n*-sector growth models when utility is discounted. J Econ Theory 12:11–30
23. Scheinkman JA (1979) Stability of regular equilibria and the correspondence principle for symmetric variational problems. Int Econ Rev 20:279–315
24. von Neumann J (1945–46) A Model of General Economic Equilibrium. Rev Econ Stud 13:19
25. Zaslavski A (2005) Turnpike Properties in the Calculus of Variations and Optimal Control. Nonconvex Optim Appl 80(22):396
26. Zelikina LF (1977) Multidimensional synthesis and the turnpike theorem in optimal control problems in economics. In: Arkin VI (ed) Probabilistic Methods of Control in Economics. Nauka, Moscow, pp 33–114

Two-Stage Stochastic Programming: Quasigradient Method

YURI ERMOLIEV
International Institute for Applied Systems Analysis, Laxenburg, Austria

MSC2000: 90C15

Article Outline

Keywords and Phrases
Anticipation, Learning, and Adaptation
Safety Constraints and CVaR Risk Measures
General Model
Convex Case
Stochastic Decomposition Techniques
Dynamic Two-Stage Problem
Decision Processes with Rolling Horizon
See also
References

Keywords and Phrases

Two-stage stochastic programming problem; Dynamic two-stage stochastic programming problem; Stochastic decomposition; Anticipation; Learning and adaptation; Conditional-value-at-risk; Safety constraints

Anticipation, Learning, and Adaptation

Two-stage stochastic programming models incorporate three major mechanisms facilitating our response (or survival) to uncertainty and changing conditions: anticipation, learning, and adaptation. Uncertainty and potential abrupt changes are pervasive characteristics of most on-going socio-economic and environmental changes. In order to manage such processes we must develop robust strategies incorporating all these mechanisms: the long term anticipative (forward looking, ex-ante) actions (policy setting, allocation of resources, engineering design, pre-disaster planning, etc.); learning (by-doing, researches, observations); and the short-term adaptive adjustments (defensive driving, marketing, inventory, control, post-disaster adaptation, etc.). The standard expected utility theory considers these mechanisms independently suggesting either anticipative (risk averse) or adaptive (risk prone) decisions. This decision paradigm often directs real policy debate, e. g., on CO^2 stabilization strategies, emphasizing either immediate actions or wait-and-see adaptation after full information become available.

The following simple example illustrates that according to the two-stage modeling approach, in general, only a part of the risk is managed by anticipative decisions whereas the other part is managed by connected with them properly designed adaptive decisions. It shows that strong interdependencies among ex-ante and ex-post decisions induce endogenous risk aversion even in linear models. The example illustrates also potential advantages of SQG methods.

Safety Constraints and CVaR Risk Measures

A stylized climate stabilization (two-stage stochastic programming) problem [8] can be formulated as follows: let x denotes an amount of emission reduction and let a random variable β denotes an uncertain critical level of required emission reduction. Ex-ante emission reductions $x \geq 0$ with costs cx may underestimate β. A linear total adaptation cost is dy, where y is an ex-post adaptation. Let us assume that ex-post adaptive capacity is unlimited (in general, it must be developed ex-ante), and $c < d$. The two-stage model is formulated as the minimization of expected total cost $cx + dEy$ subject to the constraint $x + y \geq \beta$. This problem is equivalent to the minimization of function

$F(x) = cx + E\min\{dy | x + y \geq \beta\}$ or $F(x) = cx + dE\max\{0, \beta - x\}$, which is a simple minimax problem. Optimality conditions for these types of problems show [4,5], pp 107, 416, that the optimal ex-ante solution is the critical quantile $x^* = \beta_p$ satisfying the safety constraint $Pr[x \geq \beta] \geq p$ for $p = 1 - c/d$ assuming the distribution of β has a density. This is a remarkable result: highly non-linear and even often discontinuous safety (or chance) constraint is derived (justified) from an explicit introduction of ex-post second stage decisions y. In other words, although the two stage model is linear in variables x, y, the strong risk aversion is induced among ex-ante decisions characterized by the critical quantile β_p. Only the slice β_p of the risk is managed ex-ante, whereas the rest $y^* = \beta - \beta_p$ is adapted ex-post. It is easy to see that the optimal value $F(x^*) = dE\beta I(\beta > x^*)$, where $I(\cdot)$ is the indicator function. This is the expected shortfall or Conditional Value-at-Risk (CVaR) risk measure [9].

In more realistic models, β is defined as a rather complex process dependent on scenarios of future global energy system, land use changes, demographic dynamics, etc. In these cases it is practically impossible to derive the distribution of β analytically. Instead, only random scenarios of $\beta, \beta^0, \beta^1, \beta^2, \ldots$, can be generated providing sufficient information for SQG methods. $F(x)$ is a convex and nonsmooth function, its SQG is $\xi = \xi(x, \beta) = c - d$ for $\beta < x$; and $\xi = c$ otherwise. Therefore, the SQG projection method for $k = 0, 1, \ldots$ is defined as the following

$$x(k+1) = \max\left\{0, x^k - \rho_k \xi(x^k, \beta^k)\right\}. \tag{1}$$

General Model

The model incorporates two types of independent decisions. The *ex-ante (risk averse, anticipative) decision* $x \in R^n$ of the first-stage is made on the basis of a priory information about random uncertain variables ω. The second-stage *ex-post (risk prone, adaptive) decision* $y \in R^r$ is chosen after making an additional observation on ω. For known ω, decisions x,y are evaluated by some functions $g_i(x, y, \omega), i = 0, 1, \ldots, m$, which define the constraints

$$g_i(x, y, \omega) \leq 0, \quad i = 1 : m \tag{2}$$

and the objective function $g_0(x,y,\omega)$. The ex-ante decision x which is chosen before the observation of ω cannot properly anticipate ω and, hence, satisfy (2) exactly. The ex-post decision y creates the possibility to fulfill (2) after revealing information on ω. It minimizes $g_0(x,y,\omega)$ for given x,ω subject to (2) and some additional constraints $y \in Y$ such as $y \geq 0$. Let us denote the feasible set of this standard deterministic problem as $Y(x,\omega)$ and an optimal solution as $y(x,\omega)$. In various important applications $y(x,\omega)$ is easily calculated and its existence can be easily ensured by introducing some auxiliary variables. The function $g_0(x,y,\omega)$ reflects a trade-off between choosing some options x now "and postponing other options y after" full information on ω becomes available. The general two-stage problem is to find $x \in X \subseteq R^n$ minimizing

$$F(x) = Ef(x, \omega), \tag{3}$$

where $f(x, \omega) = g_0(x, y(x, \omega), \omega)$. Besides deterministic constraints of type $x \in X$, there may also be general constraints of STO problems formulated in terms of some other random functions $f_l(x, \omega), l = 1, 2, \ldots$.

The random objective function $f(x,\omega)$ in (3) is a rather general implicitly defined nonsmooth function even for linear in (x,y) functions $g_i(x, y, \omega), i = 0, 1, \ldots, m$. Hence $F(x)$, in general, is also a nonsmooth function and general purpose *SQG methods* designed for *nonsmooth optimization problems* are applicable for minimizing (3). In fact, as the following sections show, there are fundamental obstacles in using other solution techniques even for problems with linear functions. Now consider specific SQG methods which exploit the structure of the function (3).

Convex Case

Assume that $g_i(x, y, \omega), i = 0 : m$ are convex in (x,y) functions and that a solution $\lambda(x,\omega)$ dual to the solution $y(x,\omega)$ is given. Let (g_{ix}, g_{iy}) be a subgradient of the function $g_i(\cdot, \omega)$ in variables (x,y) at a point $(x^k, y^k), k = 0, 1, \ldots$. A stochastic subgradient of the function (3) takes the form [2,5], pp 16, 171,

$$\xi(k) = g_{0x}(x^k, y^k, \omega^k) + \sum_{i=1}^{m} \lambda_i(x^k, \omega^k) g_{ix}(x^k, y^k, \omega^k), \tag{4}$$

where $y^k = y(x^k, \omega^k)$; $\omega^0, \ldots, \omega^k$ are independent samples of ω. This vector can now be used in various SQG methods.

Example 1 *Linear functions* Assume that $x \in X$, $Y = R_+^m$, and $g_0 = (c, x) + (d, y)$, $g_i = (a^i, x) + (W^i, y) - b_i$, where d, b, a^i, W^i are random vectors, i. e. $\omega = \{d, b, a^i, W^i, i = 1 : m\}$. Let us introduce matrices $A = (a^1, \ldots, a^m)$, $W = (W^1, \ldots, W^m)$. By using *SQG projection methods* with $\xi(k)$ defined as (4), we obtain the following procedure. Let x^0 be an arbitrary initial approximate solution and $\omega^0, \ldots, \omega^k$ are independent observations of ω, $\omega^k = (d^k, b^k, A^k, W^k)$, where d^k,b^k,A^k,W^k are observations of random vectors d,b and matrices A,W. Solve the linear problem (for given x^k, ω^k): $\min\{(d^k, y)\colon W^k y \le b^k - A^k x^k, y \ge 0\}$; calculate the dual variables $\lambda(x^k,\omega^k)$, $\xi(k) = c + \lambda(x^k, \omega^k)A^k$, and new

$$x^{k+1} = \pi_X \left[x^k - \rho_k \xi(k) \right], \tag{5}$$

where $k = 0, 1, \ldots$. This method was first proposed in [1,2] (see also references in [5], pp 169–171, [7], pp 213–215). It is important to note that the SQG method (5) can be regarded as a *stochastic decomposition* procedure for extremely large scale problems which often can not be solved by conventional deterministic techniques [6].

Stochastic Decomposition Techniques

Assume that $\omega = (d, b, A, W)$ has only a finite number of possible states (scenarios) $\omega = (d^s, b^s, A^s, W^s)$, $s = 1 : N$, with probabilities p_s, $\sum_{s=1}^N p_s = 1$. Then the problem with linear functions and $X = R_+^n$ is equivalent to the following deterministic large scale linear problem: minimize

$$(c, x) + p_1(d^1, y(1)) + \ldots + p_N(d^N, y(N)), \tag{6}$$

$$A^1 x + W^1 y(1) \le b^1,$$

$$- - - -$$

$$A^N x + \ldots W^N y(N) \le b^N,$$

$$x \ge 0, \quad y(1) \ge 0, \ldots, \quad y(N) \ge 0.$$

The number N may be very large: if only the vector $b = (b_1, \ldots, b_m)$ is random and each component $b_1, \ldots, b_m$ has two independent outcomes, then $N = 2^m$. Hence deterministic problem (6) can not be solved by the standard optimization techniques even with small number of constraints $m = 100$ and general random matrix A. The SQG procedure (5) is applicable also to other deterministic problems with an arbitrary block-diagonal structure of type (6), since any objective function $(\alpha, x) + (\beta^1, y(1)) + \ldots + (\beta^N, y(N))$ can be rewritten in the form of expectation (6) with $c = \alpha$, $d^s = \beta^s / p_s$, $p_s > 0$, $\sum_{s=1}^N p_s = 1$.

Example 2 Managing *agricultural risks* This example illustrates the nonsmooth character of the objective function (3), which prohibits the use of the standard stochastic approximation procedures. The main issue is to evaluate the need for an irrigation system. If the river water level is characterized by its average value, the decision to use irrigation is trivial and depends, in particular, on whether the profit per hectare of irrigated area d_1 is greater than the profit d_3 from a hectare without irrigation. The stochastic variation of the river water level creates essential difficulties. In situations of low water levels the land prepared in advance can only be partially supplemented with additional water, resulting in a profit d_2 per hectare on the remainder of the land. Besides this, the situation may also be affected by variations in water prices: it is easy to imagine a scenario for which in a dry season the use of irrigation water may become unprofitable although irrigation is profitable under average conditions. Now suppose that Q is the level of available water; q is the amount of water required for irrigation of a hectare. Denote by $x, x \le a$, the area which must be prepared in advance for irrigation, where a is the total irrigable acreage. There may be two types of risks: in situations when $Q < xq$ there is the risk to forego the profit per hectare of land that irrigates. In the case when $Q > xq$ there is the risk to forego the profit per hectare of land not prepared in advance for irrigation. These risks depend on the choice of ex-post decisions $y = (y_1, y_2, y_3)$, where y_1 is the use of irrigated land, y_2 is the use of land that was prepared for irrigated cultivation but cannot be irrigated, y_3 is the use of land that was not prepared for irrigation. Let $\omega = (Q, d_1, d_2, d_3)$, and c be the cost per hectare of irrigated land. Ex-ante and ex-post decisions x, y_1, y_2, y_3 are connected by the equations $y_1 + y_2 \le x$, $0 \le x \le a$, $y_1 + y_2 + y_3 \le a$, $Q/q \ge y_1$, $y_1 \ge 0$, $y_2 \ge 0$, $y_3 \ge 0$. The decision vector $y(x, \omega)$ maximizes the profit $d_1 y_1 + d_2 y_2 + d_3 y_3$ subject to these constraints. The sample objective function can be defined as $f(x, \omega) = -r(x, \omega)$, where $r(\cdot)$ is the revenue function $r(x, \omega) = -cx + d_1 y_1(x, \omega) +$

$d_2 y_2(x,\omega) + d_3 y_3(x,\omega)$. This function has complex nonsmooth character because $y_i(x,\omega), i = 1,2,3$ are discontinuous functions. Thus if, by a chance, $Q > xq$ and $d_1 \geq d_2$, then $y_1(x,\omega) = x$, $y_2(x,\omega) = 0$, $y_3(x,\omega) = a - x$. But if $Q > xq$ and $d_1 < d_2$, then $y_1(x,\omega) = 0$, $y_2(x,\omega) = x$, $y_3(x,\omega) = a - x$. In the case $Q \leq xq$, $d_1 \geq d_2$ the values are $y_1(x,\omega) = Q/q$, $y_2(x,\omega) = x - Q/q$, $y_3(x,\omega) = a - x$. The SQG method is defined by (5).

Dynamic Two-Stage Problem

It must be emphasized that the "stages" of the two-stage problem do not necessarily refer to two time units [2,5], pp 16–20, [7]. The x,y vectors may represent sequences of actions $x(t),y(t)$ over a given time horizon $x = (x(0), x(1), \ldots)$, $y = (y(0), y(1), \ldots)$, and in addition to the x,y decision variables, there may also be a group of variables $z = (z(0), z(1), \ldots)$ that record the state of the system at $t = 0, 1, \ldots$. The variables x,y,z,ω are often connected through a system of equations: $z(t+1) = z(t) + h(t, z(t), x(t), y(t), \omega)$, $t = 0, \ldots, T-1$. The essential new feature of such a *dynamic two-stage stochastic programming problem* is that the variables z are implicit functions of x,y,ω besides the already rather complex implicit structure of $y(x,\omega)$. This often rules out the use of deterministic optimization techniques.

Example 3 Optimal investments Consider a typical problem of optimal investments under uncertainty. Let $x_i(t)$ be the new capacity made available for electricity producing technology i at time t and $z_i(t)$ be the total capacity of i at time t. Obviously $z_i(t) = z_i(t-1) + x_i(t) - x_i(t - L_i)$, where L_i is the life-time of i. If $d_j(t)$ is different possible demand modes (scenarios) j, at time $t = 0, 1, \ldots, T$; $y_{ij}(t)$ is capacity of i (effectively) used at time t in mode j, then $\sum_j y_{ij}(t) \leq z_i(t)$ and $\sum_i y_{ij}(t) = d_j$. Let $c_i(t)$ be the unit investment cost for i at time t and $q_{ij}(t)$ be the unite production cost. The future cost and total demand can be considered truly random, i. e., elements forming ω are $d_j(t), q_{ij}(t)$. The resulting random objective function is the sum of investment and production costs: $f(x,\omega) = \sum_{i,t} c_i(t) z_i(t) + \min \sum_{i,j,t} q_{ij}(t) y_{ij}(t)$. Nonnegative variables $z_i(t)$ are uniquely defined by variables $x(t)$, i. e. $f(x,\omega)$ is an implicit function of x. The general scheme for calculation SQG is the following. Assume for simplicity $x_i(t - L_i) = 0$, $t = 0, \ldots, L_j - 1$. Suppose that at step k we have arrived at an approximate ex-ante decision variables $x_i^k(t)$, $t = 0, \ldots, T$. Next, simulate ω^k composed of $d_j^k(t), q_i j^k(t)$; calculate $z_i^k(t)$ and ex-post variables $y_{ij}^k(t)$. Let $\lambda_i^k(t)$ be the dual variables for constraints $\sum_j y_{ij}(t) \leq z_i^k(t)$. Here we suppose that these demand constraints can always be fulfilled by introducing a fictitious unlimited energy source with high operating cost. A SQG of $f(x,\omega)$ at $x = x^k$ is defined by using adjoint variables $u_i^k(t)$ (commonly used in the control theory) to dynamic equations for $z_i(t)$, [2], pp 173–175. In our case they obey simple equations: $u_i^k(T) = -c_i(T), u_i^k(t) = u_i^k(t+1) - c_i(t) + \lambda_i^k(t)$ for $t = T-1, \ldots, 1, 0$. The SQG ξ^k consists of components $\xi_i^k(t) = u_i^k(t + L_i) - u_i^k(t)$ for $t = 0, 1, \ldots, T - L_i$ and $\xi_i^k(t) = -u_i^k(t)$ for $t = T - L_i + 1, \ldots, T$.

Decision Processes with Rolling Horizon

In the dynamic two-stage problem, the learning (observation) of $\omega = (\omega(0), \ldots, \omega(t), \ldots)$ takes place only in one step before making ex-post decision $y = (y(0), \ldots, y(t), \ldots)$. In reality the learning and the decision making processes may be of a sequential character. At step $t = 0, 1, \ldots$ some uncertainties $\omega(t)$ are revealed followed by ex-post decisions $y(t,x,\omega)$, that are chosen to adapt to new information. The whole decision process proceeds in alternating steps: decision - learning - decision - .... The dependence of $y(t,x,\omega)$ on x is highly nonlinear, i. e. these functions do not posses, in general, the separability properties necessary to permit the use of the conventional recursive equations of dynamic programming. There are even more serious obstacles to the use of such recursive equations: a tremendous increase of the dimensionality and the computation of mathematical expectations. The dynamic two-stage model provides a powerful approach to dynamic decision making problems under uncertainty. At time $t = 0$ an optimal long term ex-ante strategy $x[0, T-1]$ is computed by using a priori information about uncertainty within the interval $[0, T-1]$. The decision $x(0)$ from $x[0, T-1]$ is chosen to be implemented at $t = 0$ and the new a priori information is designed for interval $[1, T]$ conditioned on the learned $\omega(0)$; a new ex-ante strategy $x[1, T]$ is computed and the decision $x(1)$ from $x[1, T]$ is chosen for the implementation at $t = 1$, and so on. This approach to *de-*

cision making with rolling horizon avoids the computation of decisions at time t as a function of all previous to t decisions, what enormously reduces the computational burden of the *recursive dynamic programming equations* and *multy-stage stochastic programs*. The decision path (strategy) $x[t, T+t-1]$ for each $t = 0, 1, \ldots$ can be viewed as a robust strategic plan over a time horizon of duration T (weeks, months, years). At each $t = 0, 1, \ldots$ this plan is revised to incorporate adaptively new information and new time horizon.

The duration T must be properly defined in order to justify strategies that may turn into benefits over long and uncertain time horizons. For example, how can we justify investments, say, in a flood defense system to cope with foreseen extreme 100-, 250-, 500-, and 1000-year floods. In such cases, T can be a random variable, so-called stopping time, associated with the occurrence of a catastrophic event. SQG methods allow to design adaptive Monte Carlo optimization procedures (learning-by-simulations) combining fast generators of catastrophes with adaptive adjustments of robust risk management decisions [3].

See also

- ▶ *L*-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Quasigradient Methods
- ▶ Stochastic Quasigradient Methods: Applications
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Two-Stage Stochastic Programs with Recourse

References

1. Ermoliev YM, Shor NZ (1968) Method of a random search for two-stage stochastic programming problems and its generalizations. Kibernetica 1:90–92
2. Ermoliev YM (1976) Methods of stochastic programming. Nauka, Moscow
3. Ermolieva T, Ermoliev YM (2005) Catastrophic risk management: flood and seismic risk case studies. In: Wallace SW, Ziemba WT (eds) Applications of stochastic programming. SIAM, MPS, pp 425–445
4. Ermoliev YM, Leonardi G (1982) Some proposals for stochastic facility locations models. Math Model 3:407–420
5. Ermoliev YM, Wets R (eds) (1988) Numerical techniques for stochastic optimization. Springer Ser Comput Math 10:1–571
6. Gaivoronski A (2005) SQG: software for solving stochastic programming problems with stochastic quasigradient methods. In: Wallace SW Ziemba WT (eds) Applications of stochastic programming, SIAM, MPS, pp 38–60
7. Kall P, Wallace S (1994) Stochastic programming. Wiley interscience series in systems and optimization. John Wiley and Sons, New York
8. O'Neil B, Ermoliev YM, Ermolieva T (2005) Endogenous risks and learning. In: Marti K, Ermoliev Y, Pflug G (eds) Coping with uncertainty. Lecture Notes in Economics and Mathematical Systems, vol 581. Springer, Berlin, pp 1–330
9. Rockafellar T, Uryasev S (2000) Optimization of Conditional-Value-at-Risk. J Risk 2:21–41

Two-Stage Stochastic Programs with Recourse

SPR

FRANCOIS LOUVEAUX[1], JOHN R. BIRGE[2]

[1] University Namur, Namur, Belgium

[2] Northwestern University, Evanston, USA

MSC2000: 90C15

Article Outline

Keywords
See also
References

Keywords

Optimization; Stochastic programming; Probabilistic programming; Uncertainty

Many real world decision problems are faced with some uncertainty. Typical examples are production/inventory problems with uncertain future demands or energy models with uncertain future fuel prices. Models that take uncertainty into account are known as stochastic models to differentiate them from

deterministic models which assume all data to be known with certainty. *Stochastic programming* is precisely the field of mathematical programming where some of the data are random variables.

In two-stage stochastic programs, some decisions, x, called *first-stage decisions*, must be taken before knowing the particular values taken by the random variables ξ while some other decisions, $y(\xi)$, *called second-stage decisions* or *corrective actions*, can be taken after the realizations of the random variables are known. In this representation, first- and second-stage are differentiated as the periods of time before and after the random data are known.

A *two-stage stochastic linear program* or *stochastic linear program with recourse* is a mathematical program of the form

$$\begin{cases} \min & c.x + Q(x) \\ \text{s.t.} & x \in X, \end{cases}$$

where $Q(x) = \mathbf{E}_{\xi} Q(x, \xi)$, $Q(x, \xi) = \min_{y(\xi)\in} Y(\xi)\, q.y(\xi)$, and $\mathbf{E}_{\xi}$ denotes the mathematical expectation with respect to the random vector ξ. X and $Y(\xi)$ are usually polyhedral convex sets. In this representation, $Q(x, \xi)$ is the second-stage value function for a given ξ and $Q(x)$ the expected value-function or *expected recourse*. It measures the impact in the second-stage and in expected terms, of a first-stage decision.

Many different situations can be represented by a recourse program. Two extreme situations for the random variables are the following. First, the random vector may represent a limited number of well studied *scenarios*. These are obtained as the best judgment experts can form about the future. In its simplest version, this may correspond to an optimistic scenario, a pessimistic scenario and a mean scenario. The stochastic solution will hedge against these scenarios, to find a solution that performs well under three scenarios, although only one will realize. On the other extreme, the random vector may represent uncertainties that recur frequently on a short-term basis. Then, the expectation somehow represents a mean over possible values of which many occur so that the expectation will match closely e. g. the mean yearly revenue/cost.

Much of the difficulty of solving a two-stage program depends on the properties of the expected recourse function and on the so-called *second-stage feasibility set*, denoted by K_2, which represents the set of first-stage decisions yielding feasible decisions in the second stage. In the case where random vectors are described by discrete distributions, $Q(x)$ is a piecewise linear convex function of x and K_2 is convex and polyhedral in x, so that classical *decomposition techniques* may apply (see ► *L*-shaped method for two-stage stochastic programs with recourse). When the random variables are not discrete, some technicalities may occur which result in difficulties over feasibility sets [5]. Those situations are fortunately infrequent. In the case of W being fixed and under weak assumptions, $Q(x)$ is convex. It is also differentiable if ξ has an absolutely continuous cumulative distribution, so that techniques from nonlinear programming can be applied. Note that continuous random variables may be approximated by discrete ones. For this process, known as discretization, see ► Semi-infinite programming: Discretization methods.

Even when decision makers realize the existence of uncertainty, in practice, they may choose to solve a deterministic model. The reason for such a choice is that stochastic models are seen as more difficult to solve. Now, as perfect forecasting does not exist, real data are very often different from the data used in the models. This results in poor decisions being taken. It is thus very often advisable to develop smaller size models that include some stochastic elements, instead of very large detailed deterministic ones that neglect the presence of uncertainty.

Measures have been developed to quantify the importance of solving a stochastic program instead of a deterministic one. The *expected value of perfect information* (EVPI) measures the maximum amount a decision maker would be ready to pay in return for complete information about the future. This concept has been developed in the context of *decision analysis*. It compares the expected objective when all decisions can be taken after the random vector is observed (the so-called *wait-and-see* solutions) and the two-stage situation. The *value of stochastic solution* (VSS) measures how much would be lost by not solving the recourse problem, but, instead, by solving some substitute deterministic model. Those concepts are studied in [1]. Unfortunately, they can only be calculated a posteriori, so that is it usually not possible to evaluate beforehand the benefit of solving a stochastic program.

A classical alternative to two-stage or recourse programs is to require that the constraints should be sat-

isfied with some level of probability. This is known as chance-constraint or *probabilistic programming*. See [4] for an extensive treatment.

Finally, one can observe that other fields also include uncertainty into their models. Examples are decision analysis, Markov decision processes or stochastic optimal control. To illustrate the difference, we may say that, typically, a two-stage stochastic program is an extension of a linear mathematical program. It involves many decision variables and constraints, discrete time periods, linear expectation functionals for the objective and known distributions for the random variables. For a general presentation of stochastic programming, see [2] or [3].

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method

References

1. Birge JR (1982) The value of stochastic solution in stochastic linear programming. Math Program 24:314–325
2. Birge JR, Louveaux FV (1997) Introduction to stochastic programming. Springer, Berlin
3. Kall P, Wallace SW (1994) Stochastic programming. Wiley, New York
4. Prékopa A (1995) Stochastic programming. Kluwer, Dordrecht
5. Walkup D, Wets RJ-B (1967) Stochastic programs with recourse. SIAM J Appl Math 15:1299–1314

U

Unconstrained Nonlinear Optimization: Newton–Cauchy Framework

J. L. NAZARETH[1,2]
[1] Department Pure and Applied Math., Washington State University, Pullman, USA
[2] Department Applied Math., University Washington, Seattle, USA

MSC2000: 90C30

Article Outline

Keywords
Notation
Model-Based Perspective
  Newton's Method
  Quasi-Newton Method
  Limited-Memory Approach
  Modified Cauchy Approach
  Summary
Metric-Based Perspective
  Cauchy Method
  Variable Metric Method
  Limited-Memory Approach
  Modified Newton Method
  Summary
Newton–Cauchy Framework
  Positive Definite Quadratic Models
  NC Method
See also
References

Keywords

Unconstrained minimization; Newton–Cauchy framework; Model-based method; Metric-based method; NC method

Unconstrained optimization methods seek a minimizing point of a nonlinear function $f\colon \mathbf{R}^n \to \mathbf{R}$, where f is smooth. The classical techniques named for I. Newton and A.-L. Cauchy view this fundamental problem from complementary perspectives, *model-based* and *metric-based*, respectively. They provide a coherent framework that relates the basic algorithms of the subject to one another and reveals hitherto unexplored avenues for further algorithmic development.

Notation

Lowercase boldface letters denote vectors, e. g., $\mathbf{x}$, and uppercase boldface letters denote matrices, e. g., $\mathbf{M}$. A matrix that is necessarily positive definite and symmetric has a + superscript attached, e. g., $\mathbf{D}^+$. Calligraphic letters, e. g., $\mathcal{H}$, denote certain distinguished matrix variables.

Model-Based Perspective

Model-based methods approximate f at a current iterate $\mathbf{x}_k$ by a local approximating model or direction-finding problem (DfP), which is used to obtain an improving point.

Newton's Method

In Newton's method the DfP is as follows:

$$\begin{cases} \min & \mathbf{g}_k^\top(\mathbf{x}-\mathbf{x}_k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_k)^\top \mathbf{H}_k(\mathbf{x}-\mathbf{x}_k) \\ \text{s.t.} & \|\mathbf{x}-\mathbf{x}_k\|_{\mathbf{D}^+} \le \delta_k, \end{cases} \tag{1}$$

where $\mathbf{g}_k$ denotes the gradient vector of f at $\mathbf{x}_k$, and $\mathbf{H}_k$ denotes its (possibly indefinite) Hessian matrix, i. e., the $n \times n$ matrix of second partial derivatives $\partial^2 f / \partial x_i \partial x_j$ at $\mathbf{x}_k$. The points $\mathbf{x}$ that satisfy the quadratic constraint form the *trust region*. The quantity $\|\cdot\|_{\mathbf{D}^+}$ de-

notes a vector norm defined by a positive definite, symmetric matrix $\mathbf{D}^+$ that determines the scaling of variables, i. e., $\| \mathbf{z} \|_{\mathbf{D}^+} = (\mathbf{z}^\top \mathbf{D}^+ \mathbf{z})^{1/2}$ for any vector $\mathbf{z}$. Common choices include the Euclidean norm $\mathbf{D}^+ = \mathbf{I}$ (where $\mathbf{I}$ denotes the $n \times n$ identity matrix) and the norm defined by a fixed diagonal scaling matrix (independent of k). The quantity δ_k is an adaptively-updated parameter that defines the size of the trust region.

It can be shown that a point $\mathbf{x}_*$ is the global solution of (1) if and only if there is a scalar $\lambda_k \geq 0$ (Lagrange multiplier) such that

$$\begin{aligned} (\mathbf{H}_k + \lambda_k \mathbf{D}^+)(\mathbf{x}_* - \mathbf{x}_k) &= -\mathbf{g}_k, \\ \lambda_k(\|\mathbf{x}_* - \mathbf{x}_k\|_{\mathbf{D}^+} - \delta_k) &= 0, \end{aligned} \tag{2}$$

with $(\mathbf{H}_k + \lambda_k \mathbf{D}^+)$ positive semidefinite.

For convenience of discussion, assume that the constraint holds as an equality at $\mathbf{x}_*$ and the matrix $(\mathbf{H}_k + \lambda_k \mathbf{D}^+)$ is positive definite. (An 'easy' case occurs when $\mathbf{x}_*$ is in the interior of the trust region so the DfP is essentially unconstrained with $\lambda_k = 0$; the other infrequent and so-called 'hard case' arises when the matrix is only positive semidefinite, and it requires a deeper analysis and refinement of the algorithmic techniques. For details, see [12].) Then the optimal multiplier is the solution of the following *one-dimensional nonlinear equation* in the variable $\lambda \geq 0$, which is derived directly from (2), namely,

$$\|w(\lambda)\|_{\mathbf{D}^+} = \delta_k, \quad w(\lambda) = -(\mathbf{H}_k + \lambda \mathbf{D}^+)^{-1}\mathbf{g}_k.$$

Also, the vector $\mathbf{x}_* - \mathbf{x}_k$ is a *direction of descent* at the point $\mathbf{x}_k$. A variety of strategies can be devised for defining the new current iterate $\mathbf{x}_{k+1}$. A *pure trust region strategy* (TR strategy) evaluates the function at $\mathbf{x}_*$. If it is not suitably improving then the current iterate is not updated, δ_k is reduced, and the procedure repeated. If $\mathbf{x}_*$ is improving then $\mathbf{x}_{k+1} = \mathbf{x}_*$, and δ_k is updated (usually by comparing function reduction predicted by the model against actual reduction). Alternatively, the foregoing strategy can be augmented by a line search along the direction of descent $\mathbf{d}_k = \mathbf{x}_* - \mathbf{x}_k$ to find an improving point, and again δ_k is revised (TR/LS strategy). See also [16] for strategies that explicitly use the dual of (1).

Quasi-Newton Method

When $\mathbf{H}_k$ is unavailable or too expensive to compute, it can be approximated by an $n \times n$ symmetric matrix, say, $\mathbf{M}_k$, which is used in the foregoing model-based approach (1) in place of $\mathbf{H}_k$. This approximation is then revised as follows. Suppose the next iterate is $\mathbf{x}_{k+1}$ and the corresponding gradient vector is $\mathbf{g}_{k+1}$, and define $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$. A standard mean value theorem for vector-valued functions states that

$$\left[\int_0^1 \mathbf{H}(\mathbf{x}_k + \theta \mathbf{s}_k)\, d\theta\right] \mathbf{s}_k = \mathbf{y}_k, \tag{3}$$

i. e., the averaged Hessian matrix over the current step transforms the vector $\mathbf{s}_k$ into $\mathbf{y}_k$. In revising $\mathbf{M}_k$ to incorporate new information, it is natural to require that the updated matrix $\mathbf{M}_{k+1}$, has the same property, i. e., that it satisfies the so-called *quasi-Newton relation* or *secant relation*:

$$\mathbf{M}_{k+1}\mathbf{s}_k = \mathbf{y}_k. \tag{4}$$

The *symmetric rank-one update* (*SR1 update*) makes the simplest possible modification to $\mathbf{M}_k$, adding to it a matrix $\kappa\ \mathbf{u}\mathbf{u}^\top$, where κ is a real number and $\mathbf{u}$ is an n-vector. The unique matrix $\mathbf{M}_{k+1}$ of this form that also satisfies (4) is as follows:

$$\mathbf{M}_{k+1} = \mathbf{M}_k + \frac{(\mathbf{y}_k - \mathbf{M}_k\mathbf{s}_k)(\mathbf{y}_k - \mathbf{M}_k\mathbf{s}_k)^\top}{(\mathbf{y}_k - \mathbf{M}_k\mathbf{s}_k)^\top \mathbf{s}_k}. \tag{5}$$

This update can be safeguarded when the denominator in the last expression is close to zero. A local approximating model analogous to (1) can be defined using the Hessian approximation in place of $\mathbf{H}_k$. The resulting model-based method is called the *symmetric rank-one quasi-Newton method* (*SR1 quasi-Newton method*). For additional detail, see [3].

Limited-Memory Approach

When storage is at a premium and it is not possible to store an $n \times n$ matrix, a *limited-memory symmetric rank-one approach* (L-SR1 approach) uses the current step and a remembered set of prior steps (usually much fewer than n in number), along with their associated gradient changes, to form a *compact representation* of the approximated Hessian. We will denote this approximation by L-$\mathbf{M}_k$. Details can be found in [2]. An alternative approach, called a *limited-memory affine reduced Hessian* or *successive affine reduction* (SAR) technique,

develops Hessian information in an affine subspace defined by the current gradient vector, the current step and a set of zero, one or more previous steps. Curvature estimates can be obtained in a Newton or a quasi-Newton sense. The associated methods are identified by the acronyms L-RH-N and L-RH-SR1 and the corresponding Hessian approximations by L-RH-$\mathbf{H}_k$ and L-RH-$\mathbf{M}_k$. The underlying updates can be patterned after analogous techniques described in [14] and [8], but they have not been fully explored, to date (1999).

Modified Cauchy Approach

Finally, when Hessian approximations in (1) are restricted to (possibly indefinite) diagonal matrices $\mathbf{D}_k$, whose elements are obtained by finite differences or updating techniques, one obtains a simple method that has also not been fully explored to date (1999). We attach the name *modified Cauchy method* to it for reasons that will become apparent in the next Section.

Summary

Each of the foregoing model-based methods utilize a DfP at the current iterate $\mathbf{x}_k$ of the form:

$$\begin{cases} \min & \mathbf{g}_k^\top(\mathbf{x}-\mathbf{x}_k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_k)^\top \mathcal{H}_k(\mathbf{x}-\mathbf{x}_k) \\ \text{s.t.} & \|\mathbf{x}-\mathbf{x}_k\|_{\mathbf{D}^+} \le \delta_k, \end{cases} \tag{6}$$

where $\mathcal{H}_k$ is one of the following: the Hessian matrix $\mathbf{H}_k$; an SR1 approximation $\mathbf{M}_k$ to the Hessian; a compact representation L-$\mathbf{M}_k$, L-RH-$\mathbf{H}_k$ or L-RH-$\mathbf{M}_k$; a diagonal matrix $\mathbf{D}_k$. (The other quantities in (6) were defined earlier.) This DfP is used in a TR or TR/LS strategy to obtain an improving point.

Metric-Based Perspective

Metric-based methods explicitly or implicitly perform a transformation of variables (or reconditioning) and employ a steepest descent search vector in the transformed space. Use of the negative gradient (steepest descent) direction to obtain an improving point was originally proposed by Cauchy.

Consider a change of variables, $\widetilde{\mathbf{x}} = \mathbf{R}\mathbf{x}$, where $\mathbf{R}$ is any $n \times n$ *nonsingular matrix*. Then $\mathbf{g}$, the gradient vector at the point $\mathbf{x}$, transforms to $\widetilde{\mathbf{g}} = \mathbf{R}^{-\top}\mathbf{g}$, which is easily verified by the chain rule. (Henceforth, we attach the symbol 'tilde' to transformed quantities, and whenever it is necessary to explicitly identify the matrix used to define the transformation, we write $\widetilde{\mathbf{x}}[\mathbf{R}]$ or $\widetilde{\mathbf{g}}[\mathbf{R}]$.) The steepest descent direction at the current iterate $\widetilde{\mathbf{x}}_k$ in the transformed space is $-\mathbf{R}^{-\top}\mathbf{g}_k$, and the corresponding direction in the original space is $-[\mathbf{R}^\top\mathbf{R}]^{-1}\mathbf{g}_k$.

Cauchy Method

If $\mathbf{R}$ is taken to be a nonsingular *diagonal matrix* $\mathbf{D}_k^+$, corresponding to a rescaling of the variables that is either fixed or is varied at each iteration, this defines a Cauchy method. A line search along the search direction $-(\mathbf{D}_k^+)^{-2}\mathbf{g}_k$ yields an improving point, and the procedure is then repeated.

Variable Metric Method

Consider next the case when the matrix defining the transformation of variables is an $n \times n$ matrix $\mathbf{R}_k$ that can be changed at each iteration. Suppose a line search procedure along the corresponding direction $-[\mathbf{R}_k^\top\mathbf{R}_k]^{-1}\mathbf{g}_k$ yields a step to an improving point $\mathbf{x}_{k+1}$ and again define $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$. How should we revise the *reconditioner* $\mathbf{R}_k$ to $\mathbf{R}_{k+1}$ in order to reflect new information? Ideally, the transformed function, say $\widetilde{f}$, should have concentric contour lines, i. e., *in the metric defined by an 'ideal reconditioner'* $\mathbf{R}_{k+1}$, the next iterate, obtained by a unit step from the transformed point $\widetilde{\mathbf{x}}_{k+1}$ along the steepest descent direction, should be *independent* of where $\widetilde{\mathbf{x}}_{k+1}$ lies along $\widetilde{\mathbf{s}}_k$. Such a reconditioner will not normally exist when f is nonquadratic, but it should at least have the aforementioned property at the two points $\widetilde{\mathbf{x}}_k$ and $\widetilde{\mathbf{x}}_{k+1}$. Thus, it is reasonable to require that $\mathbf{R}_{k+1}$ be chosen to satisfy

$$\widetilde{\mathbf{x}}_k[\mathbf{R}_{k+1}] - \widetilde{\mathbf{g}}_k[\mathbf{R}_{k+1}] = \widetilde{\mathbf{x}}_{k+1}[\mathbf{R}_{k+1}] - \widetilde{\mathbf{g}}_{k+1}[\mathbf{R}_{k+1}]. \tag{7}$$

This equation can be re-expressed in the original variables as follows:

$$\mathbf{R}_{k+1}\mathbf{s}_k = \mathbf{R}_{k+1}^{-\top}\mathbf{y}_k. \tag{8}$$

For a matrix $\mathbf{R}_{k+1}$ satisfying (8) to exist, it is necessary and sufficient that $\mathbf{y}_k^\top\mathbf{s}_k > 0$, which can always be assured by a line search procedure. Since $\mathbf{R}_{k+1}^\top\mathbf{R}_{k+1}\mathbf{s}_k = \mathbf{y}_k$, we see that (8) is equivalent to the *quasi-Newton*

relation (4) when we impose the added restriction that the Hessian approximation is positive definite.

Consider the question of how to revise $\mathbf{R}_k$. The so-called *BFGS update* (named after the first letter of the surname of its four co-discoverers) makes the simplest possible augmentation of $\mathbf{R}_k$ by adding to it a matrix $\mathbf{u}\mathbf{v}^\top$ of rank one. The updated matrix $\mathbf{R}_{k+1}$ is required to satisfy (8) and is chosen as close as possible to $\mathbf{R}_k$, as measured by the Frobenius or spectral norm of the difference $(\mathbf{R}_{k+1} - \mathbf{R}_k)$. This update can be shown to be as follows:

$$\mathbf{R}_{k+1} = \mathbf{R}_k + \frac{\mathbf{R}_k \mathbf{s}_k}{\|\mathbf{R}_k \mathbf{s}_k\|} \left(\frac{\mathbf{y}_k}{(\mathbf{y}_k^\top \mathbf{s}_k)^{1/2}} - \frac{\mathbf{R}_k^\top \mathbf{R}_k \mathbf{s}_k}{\|\mathbf{R}_k \mathbf{s}_k\|} \right)^\top, \tag{9}$$

where $\|\cdot\|$ denotes the Euclidean vector norm. The descent search direction is defined as before by

$$\mathbf{d}_{k+1} = -[\mathbf{R}_{k+1}^\top \mathbf{R}_{k+1}]^{-1} \mathbf{g}_{k+1},$$

and a line search along it will yield an improving point. The foregoing BFGS algorithm is an outgrowth of seminal variable-metric ideas pioneered by W.C. Davidon [4] and clarified by R. Fletcher and M.J.D. Powell [6]. For other original references, see, for example, the bibliographies in [1] or [5].

Let $\mathbf{M}_{k+1}^+ = \mathbf{R}_{k+1}^\top \mathbf{R}_{k+1}$ and $\mathbf{W}_{k+1}^+ = [\mathbf{M}_{k+1}^+]^{-1}$. These quantities can be updated directly as follows:

$$\mathbf{M}_{k+1}^+ = \mathbf{M}_k^+ - \frac{\mathbf{M}_k^+ \mathbf{s}_k \mathbf{s}_k^\top \mathbf{M}_k^+}{\mathbf{s}_k^\top \mathbf{M}_k^+ \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^\top}{\mathbf{y}_k^\top \mathbf{s}_k} \tag{10}$$

$$\mathbf{W}_{k+1}^+ = \mathbf{E}_k \mathbf{W}_k^+ \mathbf{E}_k^\top + \frac{\mathbf{s}_k \mathbf{s}_k^\top}{\mathbf{y}_k^\top \mathbf{s}_k}, \tag{11}$$

where

$$\mathbf{E}_k = \mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^\top}{\mathbf{y}_k^\top \mathbf{s}_k}.$$

Limited-Memory Approach

When storage is at a premium, a version of the BFGS algorithm (*L-BFGS*) preserves steps and corresponding changes in gradients over a limited number of prior iterations, and defines the matrix $\mathbf{M}_k^+$ or $\mathbf{W}_k^+$ implicitly in terms of these vectors instead of explicitly by forming a square matrix. Consider the simplest case where a single step and gradient change are preserved (one-step memory). The update is then defined implicitly by (11) with $\mathbf{W}_k^+ = \gamma_k \mathbf{I}$, where γ_k is a scaling constant often taken to be $\mathbf{y}_k^\top \mathbf{s}_k / \mathbf{y}_k^\top \mathbf{y}_k$. Thus the search direction is defined by

$$\mathbf{d}_{k+1} = -\left[\mathbf{E}_k (\gamma_k \mathbf{I}) \mathbf{E}_k^\top + \frac{\mathbf{s}_k \mathbf{s}_k^\top}{\mathbf{y}_k^\top \mathbf{s}_k} \right] \mathbf{g}_{k+1}.$$

Under the assumption of exact line searches, i. e., $\mathbf{g}_{k+1}^\top \mathbf{s}_k = 0$ it follows immediately that the search direction is parallel to the following vector:

$$-\mathbf{g}_{k+1} + \frac{\mathbf{y}_k^\top \mathbf{g}_{k+1}}{\mathbf{y}_k^\top \mathbf{s}_k} \mathbf{s}_k. \tag{12}$$

This is the search direction used in the *conjugate gradient method*, pioneered by M.R. Hestenes and E.L. Stiefel [9] and later suitably adapted to nonlinear optimization by Fletcher and C. Reeves [7]. We say that L-BFGS is a *CG-related algorithm*.

More generally, a set of prior steps and gradient changes can be preserved and the update defined recursively (see [11]). Key implementation issues are addressed in [8]. An alternative compact representation for L-BFGS is given in [2]. Henceforth, let us denote the Hessian and inverse Hessian approximations in L-BFGS by L-$\mathbf{M}_k^+$ and L-$\mathbf{W}_k^+$, respectively.

A *limited-memory reduced-Hessian* or *successive affine reduction* version of the BFGS algorithm develops curvature approximations in an affine subspace defined by the current gradient vector and a set of prior steps. We will denote this algorithm by *L-RH-BFGS* and its Hessian approximation by L-RH-$\mathbf{M}_k^+$. It too can be shown to be CG-related. For details, see [8,14] and references given therein.

Modified Newton Method

If $\mathbf{H}_k$ is available and possibly indefinite, it can be modified to a positive definite matrix, $\mathbf{H}_k^+$, in a variety of ways (see [1]). This modified matrix can be factored as $\mathbf{H}_k^+ = \mathbf{R}_k^\top \mathbf{R}_k$ with $\mathbf{R}_k$ nonsingular, for example, by using a Cholesky factorization or an eigendecomposition. The factor $\mathbf{R}_k$ then defines a metric-based algorithm as above called a *modified Newton method* (MN).

A limited-memory modified Newton algorithm analogous to L-RH-BFGS can also be formulated. For details, see [14]. Denote this CG-related algorithm by L-RH-MN and its Hessian approximation by L-RH-$\mathbf{H}_k^+$.

Summary

The *steepest descent direction* in each of the foregoing methods is the direction $\mathbf{d}_k$ that minimizes $\mathbf{g}_k^\top \mathbf{d}$ over all vectors $\mathbf{d}$ of constant length δ_k in an appropriate metric. (Typically $\delta_k = 1$.) Let $\mathbf{d} = \mathbf{x} - \mathbf{x}_k$. Then this DfP can equivalently be stated as follows:

$$\begin{cases} \min & \mathbf{g}_k^\top (\mathbf{x} - \mathbf{x}_k) \\ \text{s.t.} & \|\mathbf{x} - \mathbf{x}_k\|_{\mathcal{M}_k^+} \le \delta_k, \end{cases} \tag{13}$$

where $\mathcal{M}_k^+$ is given by one of the following: $\mathbf{I}$; $(\mathbf{D}_k^+)^2$; $\mathbf{M}_k^+$; L-$\mathbf{M}_k^+$; L-RH-$\mathbf{H}_k^+$; L-RH-$\mathbf{M}_k^+$; $\mathbf{H}_k^+$. The quantity δ_k determines the length of the initial step along the search direction, and a line search (LS) strategy yields an improving point.

Let us denote the inverse of $\mathcal{M}_k^+$ by $\mathcal{W}_k^+$. It is sometimes computationally more efficient to maintain the latter matrix, for example, within a limited-memory BFGS algorithm.

Newton–Cauchy Framework

A simple and elegant picture emerges from the development in the two Sections above, which is summarized by Fig. 1 (see also [15]). It is often convenient to straddle this 'two-lane highway', so to speak, and to formulate algorithms based on a 'middle-of-the-road' approach. We now describe the traditional synthesis based on positive definite, unconstrained models and a new synthesis, called the NC method, based on $\mathcal{M}_k^+$-metric trust regions.

Positive Definite Quadratic Models

At the current iterate $\mathbf{x}_k$, use an unconstrained DfP of the following form:

$$\min \mathbf{g}_k^\top (\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^\top \mathcal{H}_k^+ (\mathbf{x} - \mathbf{x}_k), \tag{14}$$

where $\mathcal{H}_k^+$ is one of the following: $\mathbf{I}$; $(\mathbf{D}_k^+)^2$; $\mathbf{M}_k^+$; L-$\mathbf{M}_k^+$; L-RH-$\mathbf{H}_k^+$; L-RH-$\mathbf{M}_k^+$; $\mathbf{H}_k^+$. Note that the DfP uses the options available in (13), and indeed is a Lagrangian relaxation of the latter. Often these options for $\mathcal{H}_k^+$ are derived directly in model-based terms. The search direction obtained from (14) is $\mathbf{d}_k = -\,[\mathcal{H}_k^+]^{-1}\,\mathbf{g}_k$ and a line search along it yields an improving iterate. A good discussion of this line of development can be found in [1].

model-based	metric-based
Newton (N)	modified Newton (MN)
SR1	BFGS
L-SR1; L-RH-N#; L-RH-SR1#	L-BFGS*; L-RH-MN*; L-RH-BFGS*
modified Cauchy (MC)#	Cauchy (C)

Unconstrained Nonlinear Optimization: Newton–Cauchy Framework, Figure 1
Newton–Cauchy framework. Legend: * – CG-related; # – not fully explored

NC Method

Substantial order can be brought to computational unconstrained nonlinear minimization by recognizing the existence of a *single underlying method*, henceforth called the Newton–Cauchy or NC method, which is based on a model of the form (6), but with its trust region now employing a metric corresponding to (13). This DfP takes the form

$$\begin{cases} \min & \mathbf{g}_k^\top (\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^\top \mathcal{H}_k (\mathbf{x} - \mathbf{x}_k) \\ \text{s.t.} & \|\mathbf{x} - \mathbf{x}_k\|_{\mathcal{M}_k^+} \le \delta_k, \end{cases} \tag{15}$$

where the matrix $\mathcal{H}_k$ is one of the following: the zero matrix $\mathbf{0}$; $\mathbf{H}_k$; $\mathbf{M}_k$; L-$\mathbf{M}_k$; L-RH-$\mathbf{H}_k$; L-RH-$\mathbf{M}_k$; $\mathbf{D}_k$. Note that the objective is permitted to be linear. The matrix $\mathcal{M}_k^+$ is one of the following: $\mathbf{I}$; $(\mathbf{D}_k^+)^2$; $\mathbf{M}_k^+$; L-$\mathbf{M}_k^+$; L-RH-$\mathbf{H}_k^+$; L-RH-$\mathbf{M}_k^+$; $\mathbf{H}_k^+$. Despite numerical drawbacks, it is sometimes computationally more convenient to maintain $\mathcal{W}_k^+$; also note $\mathcal{H}_k = \mathcal{M}_k^+$ gives a model equivalent to $\mathcal{H}_k = \mathbf{0}$.

The underlying theory for trust region subproblems of the form (15) and techniques for computing the associated multiplier λ_k can be derived, for example, from [13] or [16], and the next iterate is obtained by a TR/LS strategy as discussed in the first Section; in particular, a line search can ensure that $\mathbf{y}_k^\top \mathbf{s}_k > 0$, which is needed whenever a variable metric is updated. Note also when the objective function is linear that (15) reduces to (13), and the TR/LS strategy reduces to a line-search strategy.

The NC method can be formulated into a large variety of individual *NC algorithms*. These include all the standard ones in current use, along with many new and potentially useful algorithms, each a particular algorithmic expression of the same underlying method. A sample of a few algorithms from

$\mathcal{H}_k$	$\mathcal{M}_k^+$	Strategy	Algorithm
$\mathbf{H}_k$	$\mathbf{D}^+$	TR	Newton
0	$(\mathbf{D}_k^+)^2$	LS	Cauchy
$\mathbf{M}_k$	$\mathbf{D}^+$	TR	SR1
0	$\mathbf{M}_k^+$	LS	BFGS
$\mathbf{H}_k$	$\mathbf{M}_k^+$	TR/LS	*
$\mathbf{M}_k$	$\mathbf{M}_k^+$	TR/LS	*
L-$\mathbf{M}_k$	$\mathbf{D}^+$	TR	L-SR1
0	L-$\mathbf{M}_k^+$	LS	L-BFGS
0	L-RH-$\mathbf{M}_k^+$	LS	L-RH-BFGS
L-$\mathbf{M}_k$	L-$\mathbf{M}_k^+$	TR/LS	*
L-RH-$\mathbf{H}_k$	L-$\mathbf{M}_k^+$	TR/LS	*
L-RH-$\mathbf{H}_k$	L-RH-$\mathbf{M}_k^+$	TR/LS	*

Unconstrained Nonlinear Optimization: Newton–Cauchy Framework, Figure 2
Examples of NC algorithms

amongst the many possibilities (combinations of $\mathcal{H}_k$ and $\mathcal{M}_k^+$) is given in Fig. 2. The last column identifies each algorithm, and the symbol $*$ indicates that it is new.

We see that even in this relatively mature field there are still ample opportunities for new algorithmic contributions, and for associated convergence and rate of convergence analysis, numerical experimentation and the development of quality software.

See also

- ► Automatic Differentiation: Calculation of Newton Steps
- ► Broyden Family of Methods and the BFGS Update
- ► Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ► Interval Newton Methods
- ► Large Scale Unconstrained Optimization
- ► Nondifferentiable Optimization: Newton Method
- ► Numerical Methods for Unary Optimization
- ► Unconstrained Optimization in Neural Network Training

References

1. Bertsekas DP (1999) Nonlinear programming, 2nd edn. Athena Sci., Belmont, MA
2. Byrd RH, Nocedal J, Schnabel RB (1994) Representations of quasi-Newton matrices and their use in limited-memory methods. Math Program 63:129–156
3. Conn AR, Gould NIM, Toint PhL (1991) Convergence of quasi-Newton matrices generated by the symmetric rank one update. Math Program 50:177–196
4. Davidon WC (1991) Variable metric method for minimization. SIAM J Optim 1:1–17 (Original (with different preface): Argonne Nat. Lab. Report ANL-5990 (Rev., Argonne, Illinois))
5. Dennis JE, Schnabel RB (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs, NJ
6. Fletcher R, Powell MJD (1963) A rapidly convergent descent method for minimization. Comput J 6:163–168
7. Fletcher R, Reeves C (1964) Function minimization by conjugate gradients. Comput J 6:149–154
8. Gilbert JC, Lemaréchal C (1989) Some numerical experiments with variable–storage quasi-Newton algorithms. Math Program B 45:407–435
9. Hestenes MR, Stiefel EL (1952) Methods of conjugate gradients for solving linear systems. J Res Nat Bureau Standards (B) 49:409–436
10. Leonard MW (1995) Reduced Hessian quasi–Newton methods for optimization. PhD Diss Univ Calif, San Diego, CA
11. Liu DC, Nocedal J (1989) On the limited memory BFGS method for large-scale optimization. Math Program B 45:503–528
12. Moré JJ (1983) Recent developments in algorithms and software for trust region methods. In: Bachem A, Grötschel M, Korte B (eds) Mathematical Programming: The State of the Art (Bonn, 1982). Springer, Berlin, pp 258–287
13. Moré JJ (1993) Generalizations of the trust region problem. Optim Methods Softw 2:189–209
14. Nazareth JL (1986) Conjugate gradient algorithms less dependent on conjugacy. SIAM Rev 28:501–511
15. Nazareth JL (1994) The Newton–Cauchy framework: A unified approach to unconstrained nonlinear minimization. Lecture Notes Computer Sci, vol 769. Springer, Berlin
16. Rendl F, Wolkowicz H (1997) A semidefinite framework for trust region subproblems with applications to large scale minimization. Math Program 77:273–300

Unconstrained Optimization in Neural Network Training

UONNT

LUIGI GRIPPO
Rome University La Sapienza, Rome, Italy

MSC2000: 90C30, 90C30, 90C52, 90C53, 90C55, 65K05, 68T05

Article Outline

Keywords
See also
References

Keywords

Neural networks; Unconstrained optimization; Training algorithms

The *training* (or *supervised learning*) problem for a neural network [5,13] of given topology can be formulated as the problem of determining the network parameters by minimizing some measure of the error between the desired output and the actual output, in correspondence to a given set of input data. More specifically, consider a static network and suppose that a set of desired input/output patterns (training set) is given:

$$T = \{(\xi_j, t_j) : \ j = 1, \ldots, M\},$$

where it is assumed that $\xi_j \in \mathbf{R}^p$ and $t_j \in \mathbf{R}$.

Denoting by $w \in \mathbf{R}^n$ the vector of network parameters and by $y(w; \xi) \in \mathbf{R}$ the output of the network in response to an input ξ, the training problem can be formulated, for instance, as the (nonlinear) least squares problem (cf. also ▶ Least squares problems):

$$\min_w E(w) := \sum_{j=1}^{M} \big(y(w; \xi_j) - t_j\big)^2 .$$

As an example, consider a simple *two-layer feed-forward network* with one input $\xi \in \mathbf{R}$ and one output $y \in \mathbf{R}$, having a 'hidden layer' constituted by two neural units with a sigmoidal input-output function ('activation function') defined by:

$$\phi(s) = \frac{1}{1 + e^{-s}},$$

and an output layer consisting of one linear unit.

Let u_1, u_2 be the weights on the input connections, let θ_1, θ_2 be the 'thresholds' associated to the hidden units and let v_1, v_2 be the weights on the output connections. Then the input/output map realized by the network is given by

$$y = v_1\phi(u_1\xi_j - \theta_1) + v_2\phi(u_2\xi_j - \theta_2).$$

Therefore, given a set $\{(\xi_j, t_j)\}$ of training pairs, the training problem may consist in determining the parameter vector

$$w = (u_1, u_2, \theta_1, \theta_2, v_1, v_2)$$

that minimizes the error function:

$$\sum_{j=1}^{M} (v_1\phi(u_1\xi_j - \theta_1) + v_2\phi(u_2\xi_j - \theta_2) - t_j)^2.$$

Problems of this form constitute challenging unconstrained optimization problems, which typically exhibit almost all the difficulties that may be encountered in the minimization of a nonlinear function, and in particular:

- multiple local minima;
- steep sided valleys;
- extensive flat regions;
- singularities in the Hessian matrix.

Moreover, the number n of unknown parameters can be large in many practical applications and the number M of error terms can be huge, since it corresponds to the number of training examples used in the learning process.

It is also important to realize that the minimization of the error function E itself is not the ultimate goal of network training. In fact, the quality of learning would rather depend on the network's ability of making good predictions for new inputs (not considered in the training data), that is, on the 'generalization capability' of the network, [5,13]. Thus, a considerable amount of experimentation may be required in order to choose properly the model complexity in relation to the available data and to evaluate the results of the training phase. In practice, this implies that very often the minimization process has to be repeated, in correspondence to different architectures, different training sets, different stopping rules, and possibly also in correspondence to different error functions. Then, although learning cannot be simply reduced to an optimization problem, the availability of efficient optimization algorithms can be crucial for a successful learning.

Deterministic iterative methods that attempt to find a minimizer of E can be categorized into:

- batch methods, which use at each step information on the global error function E;

- *on-line methods*, which use information on a single error term

$$E_j(w) = \big(y(w;\xi_j) - t_j\big)^2$$

at time.

Batch methods can be used only for *off-line learning*, that is when the whole training set is available before that learning is started, whereas on-line methods can be employed both for off-line learning and for *on-line learning*, that is when the training set is formed during real time operations of the network.

In the neural network literature the best known training method is the so-called *backpropagation method* (BP) [22], which owes its name essentially to the technique used for computing the derivatives of E in a multilayer network, a technique based on an efficient use of the chain rule, which can be identified with the reverse mode technique of automatic differentiation [23] (cf. also ► Automatic differentiation: Introduction, history and rounding error estimation). The BP method has been implemented both in batch mode ('batch BP' or 'off-line BP') and in an on-line mode ('on-line BP' or 'stochastic BP' or 'pattern mode training').

In the batch version, the BP method can be viewed as an heuristic implementation of the gradient method (or steepest descent method) and it can be described by the iteration:

$$w^{k+1} = w^k - \alpha \nabla E(w^k), \tag{1}$$

where ∇E is the gradient of E and the stepsize $\alpha > 0$ is termed the 'learning rate'. Global convergence and rate of convergence analyses of iteration (1) with a constant stepsize can be found, for instance, in [2]. In particular, it is known that a convergent implementation would require the stepsize to satisfy the bound

$$0 < \alpha < \frac{2}{L},$$

under the assumption that ∇E is Lipschitz-continuous on $\mathbf{R}^n$ with constant L. As the value of L is unknown, it may be difficult to find an appropriate value for α. This may suggest the use of an inexact line search technique for computing the stepsize along the search direction. With a suitable implementation, the increase in the computational cost per iteration due to the line searches would be compensated by a definite improvement in the overall efficiency and reliability.

Modifications of the BP method and heuristic rules for choosing and updating α have been also extensively studied in the neural network literature [7,25].

An improved version of the basic BP method is the so-called *momentum updating rule* [22], which consists in the iteration:

$$w^{k+1} = w^k - \alpha \nabla E(w^k) + \beta(w^k - w^{k-1}), \tag{2}$$

where $\beta > 0$ is a suitable parameter. In the optimization literature, this method is known as the *heavy ball method* [20], because of a physical analogy with the motion of a body in a potential field, subject to an energy loss caused by friction. Under appropriate assumptions, it can be shown [21] that the convergence rate of this method is superior to that of the gradient method, but there is again the difficulty of choosing suitable values for the parameters α and β.

On the other hand, when batch training is adopted, many well-known unconstrained minimization methods are available for computing stationary points of E, such as

- conjugate gradient methods (CGM; cf. ► Conjugate-gradient methods);
- quasi-Newton methods;
- Newton-type methods.

Moreover, as the training problem is a nonlinear least squares problem, also the use of some modified form of the Gauss–Newton method is suitable, at least when small residues are expected.

It can be observed that an iteration of the CGM can be placed equivalently into the form

$$w^{k+1} = w^k - \alpha^k \nabla E(w^k) + \frac{\alpha^k \beta^k}{\alpha^{k-1}}(w^k - w^{k-1}),$$

where α^k is the stepsize (to be computed through a line search) and β^k is the parameter appearing in the particular CGM formula we may adopt. This would correspond to a momentum updating rule where the choice of the parameters α^k and β^k can be made on a sound

theoretical basis. In fact, the use of the CGMhas been suggested since the early papers on neural networks and good results have been obtained [6], for instance, by employing the Polak–Ribière CGM, which corresponds to the choice:

$$\beta^k = \frac{\nabla E(w^k)^\top (\nabla E(w^k) - \nabla E(w^{k-1}))}{\left\| \nabla E(w^{k-1}) \right\|^2}.$$

Various applications and adaptations of the CGM can be found in the recent neural network literature [18,25]. For large scale training problems a viable alternative can be also the use of some reduced memory quasi-Newton method [19]. In particular, training algorithms employing a *memoryless BFGS method* (also known as *Shanno conjugate gradient method* [24]) have been considered [1]. This method can be defined through the iteration

$$w^{k+1} = w^k + \alpha^k d^k,$$

where the stepsize α^k is computed through an (inaccurate) line search and the search direction d^k is obtained by taking initially $d^0 = -\nabla E(w^0)$ and then computing for $k > 0$ the vectors

$$s = w^k - w^{k-1},$$
$$y = \nabla E(w^k) - \nabla E(w^{k-1})$$

and letting

$$d^k = -\nabla E(w^k) + a^k y^k + b^k s^k,$$

where:

$$a^k = \frac{(s^k)^\top \nabla E(w^k)}{(s^k)^\top y^k},$$
$$b^k = -a^k \left(1 + \frac{(y^k)^\top y^k}{(s^k)^\top y^k}\right) + \frac{(y^k)^\top \nabla E(w^k)}{(s^k)^\top y^k}.$$

It can be shown that the search directions are descent directions, provided that

$$(s^k)^\top y^k > 0,$$

which can be enforced through an appropriate line search.

Several experiments have been also made by using Newton-type methods employing second order derivatives of E. In particular, truncated Newton methods appear to be valuable for large-dimensional training problems [8], but the use of second order methods may be not so convenient in case of singularities in the Hessian matrix of E at the solutions, since the superlinear convergence rate usually associated to Newton-type methods may be lost. However, singularities are most likely to occur when the number of free parameters is too large in relation to the available data, a situation which would suggest the need of 'pruning' the network.

On the whole, whenever batch learning is viable, it can be safely said that unconstrained minimization methods are of some order of magnitude faster with respect early heuristic training methods and the CGM method appears to be the technique of election. Special cautions are required in the choice of the starting point w^0, which, in principle, should be such that the level set

$$\mathcal{L} = \left\{ w : \; E(w) \leq E(w^0) \right\}$$

is bounded. A useful device may be that of minimizing a modified objective function of the form

$$\widetilde{E}(w) = E(w) + \varepsilon \, \|w\|^2$$

where ε is a small parameter. This ensures that all level sets are compact and may prevent the algorithms from reaching flat regions corresponding to very large values of w. The addition of the term $\varepsilon \parallel \mathrm{w} \parallel^2$ may also have an important motivation in the context of learning theory since it corresponds, in essence, to regularizing the error function by introducing a 'complexity penalty' that 'encourages the excess weights to assume values closer to zero, and thereby improve generalization' [13].

Moreover, in association with a local method, some form of multistart method may be required for searching a global minimizer or, alternatively, the use of a deterministic technique of global optimization can be attempted. Training algorithms employing *homotopic methods* and *tunneling methods* are described in [25].

However, batch methods are not suitable in on-line learning problems, since the objective function is not known when training is started, and hence on-line methods have to be adopted.

In the on-line version, the BP method can be described as a sequence of cycles ('epochs'), each consisting of M iterations that update the vector w by employing the negative gradient of a single error term E_j at time (possibly in a random order). This can be described by means of the following simplified scheme.

1. Choose w^0 and set $k = 0$;
2. Set $y_0 = w^k$ and choose α;
3. For $j = 1, \ldots, M$:
   set $y_j = y_{j-1} - \alpha \nabla E_j(y_{j-1})$
4. Set $w^{k+1} = y_M$;
5. Set $k = k + 1$ and return to 2.

On-line backpropagation

It is also possible to introduce a 'momentum term' by replacing, for $j > 1$, the update considered at Step 3 of the preceding scheme with an update of the form:

$$y_j = y_{j-1} - \alpha \nabla E_j(y_{j-1}) + \beta(y_{j-1} - y_{j-2}),$$

or else by introducing a memory of the preceding step also when passing to a new epoch.

For on-line BP, the problem of selecting an appropriate value for α becomes more critical, since the method may not converge with a fixed stepsize. In spite of this, various heuristic versions of on-line BP have been employed with satisfactory results in many neural network applications and the interesting fact is that on-line BP employing simple heuristics is often superior to more sophisticated batch unconstrained optimization methods even for off-line learning, at least when the number of training pairs is very large and the training set is highly redundant [28]. In fact, in this case, it could be wasteful to spend much time in computing exactly the gradient of E when far from the solution. In addition, it would seem that an on-line procedure introduces some sort of randomness that may help escaping from local minimizers. In fact, on-line BP can be viewed as a *stochastic process* that introduces a random error on the gradient, which may prevent the algorithm from being trapped at irrelevant local minimizers.

Neural network applications have stimulated a considerable interest in the field of unconstrained optimization towards the convergence analysis of on-line methods (often termed *incremental gradient methods*) and the development of new algorithms. From a deterministic point of view, incremental gradient methods can be viewed as algorithms for minimizing a sum of M differentiable functions, in which the computation of derivatives is split into a set of M (or more) consecutive steps. An early contribution to the study of this problem (with a different motivation) has been given in [14], where the case of a convex objective function is studied. More recently, convergence results have been obtained by giving rules for the stepsizes, which, under suitable assumptions, may ensure convergence towards stationary points. In particular, stepsize rules for nonconvex problems have been established in [9,29], by using stochastic approximation ideas and in [15,16], by employing deterministic approaches.

A thorough analysis of incremental gradient methods and a description of an incremental version of the Gauss–Newton method, which leads to a discrete version of the extended Kalman filter, can be found in [4].

In particular, in the case of on-line BP, under the assumption that the gradients ∇E_j are Lipschitz continuous and that the sequence $\{w^k\}$ generated by on-line BP is bounded (or else that $\| \nabla E_j \|$ grows at most linearly with $\| \nabla E \|$) it can be shown that every limit point of $\{w^k\}$ is a stationary point of E, provided that the stepsizes α^k are such that

$$\sum_{k=0}^{\infty} \alpha^k = \infty \quad \text{and} \quad \sum_{k=0}^{\infty} (\alpha^k)^2 < \infty.$$

As an example, the choice

$$\alpha^k = \frac{c}{k},$$

for some $c > 0$, which is considered in the stochastic approximation literature, would satisfy these conditions. Similar convergence results have been also established in connection with the momentum updating rule [16].

In any case, we have that the stepsize must be driven, in principle, to zero for ensuring convergence of on-line methods, and this could be highly inefficient in comparison with a batch gradient method. Some attempts have been made in order to overcome this difficulty. A possible compromise is the use of hybrid on-

line-batch BP techniques, called *bold-driver methods* in the neural network literature, where on-line BP is used with a fixed stepsize for one (or more epochs), but the stepsize is revised periodically by evaluating the behavior of the global error function E [27]. Globally convergent algorithms based on this idea have been also considered in [10] and [26]. These techniques may improve the performance of on-line BP in case of off-line learning with a moderately large and redundant training set, but still has the disadvantage that the whole objective function has to be evaluated, which is expensive when M is very large and is unsuitable in case of 'truly' on-line learning. A different approach can be that of reducing the degree of incrementalism as the method progresses and to gradually switch from the incremental gradient method, which can be quite effective at early stages of the process, to the steepest descent method, which has a much better ultimate convergence rate [3]. Still another approach can be that of constructing multiple copies of the network, each trained (possibly in a batch mode) with different data blocks (added as they become available), and then penalizing the disagreement between the various solutions in a way that ultimate convergence can be achieved [11,12]. However, an extensive computational testing of these approaches is not available and research on incremental gradient methods is still an active field, so that further progresses may be expected.

Suggested general references on the application of optimization methods to neural network training are [4,5,13,25], and [17].

See also

- ▶ Broyden Family of Methods and the BFGS Update
- ▶ Conjugate-gradient Methods
- ▶ Large Scale Unconstrained Optimization
- ▶ Neural Networks for Combinatorial Optimization
- ▶ Neuro-dynamic Programming
- ▶ Numerical Methods for Unary Optimization
- ▶ Replicator Dynamics in Combinatorial Optimization
- ▶ Unconstrained Nonlinear Optimization: Newton–Cauchy Framework

References

1. Battiti R, Masulli F (1990) BFGS optimization for faster and automated supervised learning. In: Proc. Internat. Neural Network Conf. Paris, pp 757–760
2. Bertsekas DP (1995) Nonlinear programming. Athena Sci., Belmont, MA
3. Bertsekas DP (1997) A new class of incremental gradient methods for least squares problems. SIAM J Optim 7:913–926
4. Bertsekas DP, Tsitsiklis JN (1996) Neuro-dynamic programming. Athena Sci., Belmont, MA
5. Bishop CM (1995) Neural networks for pattern recognition. Clarendon Press, Oxford
6. Charalambous C (1992) Conjugate gradient algorithm for efficient training of artificial neural networks. IEEE Proc Part G 139:301–310
7. Cichocki A, Unbehauen R (1993) Neural networks for optimization and signal processing. Wiley, New York
8. Dixon LCW (1993) Neural networks and unconstrained optimization. In: Spedicato E (ed) Algorithms for continuous optimization: the state of the art. NATO ASI Ser. Kluwer, Dordrecht, pp 513–530
9. Gaivoronski AA (1994) Convergence analysis of parallel back-propagation algorithm for neural networks. Optim Methods Softw 4:117–134
10. Grippo L (1994) A class of unconstrained minimization methods for neural network training. Optim Methods Softw 4:135–150
11. Grippo L (1996) Globally convergent online minimization algorithms for neural network training. In: Di Pillo G, Gianessi F (eds) Nonlinear Optimization and Applications. Plenum, New York, pp 181–195
12. Grippo L (1999) Convergent online algorithms for supervised learning in neural networks. Techn Report DIS, Univ Roma La Sapienza, no 24-99
13. Haykin S (1994) Neural networks. MacMillan, New York
14. Kibardin VM (1980) Decomposition into functions in the minimization problems. Automation and Remote Control 40:1311–1323
15. Luo Z-Q, Tseng P (1994) Analysis of an approximate gradient projection method with applications to the backpropagation algorithm. Optim Methods Softw 4:85–102
16. Mangasarian OL, Solodov MV (1994) Serial and parallel backpropagation convergence via nonmonotone perturbed minimization. Optim Methods and Software 4:103–116
17. Masters T (1997) Advanced algorithms for neural networks. A C++ sourcebook. Wiley, New York
18. Møller M (1993) A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks 6:525–533
19. Nash SG, Nocedal J (1991) A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization. SIAM J Optim 14:358–372

20. Polijak BT (1964) Some methods of speeding up the convergence of iteration methods. Z VyCisl Mat Mat Fiz 4:1–17
21. Polijak BT (1987) Introduction to optimization. Optim. Software, New York
22. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representation by error backpropagation. In: Rumelhart DE, McClelland (eds) Parallel Distributed Processing-Explorations in the microstructure of cognition. MIT, Cambridge, MA, pp 318–362
23. Saarinen S, Bramley RB, Cybenko G (1991) Neural networks, backpropagation and automatic differentiation. In: Griewank A, Corliss GF (eds) Automatic differentiation of algorithms. SIAM, Philadelphia, pp 31–42
24. Shanno DF (1978) Conjugate gradient methods with inexact line searches. Math Oper Res 3:244–256
25. Shepherd AJ (1997) Second-order methods for neural networks. Springer, Berlin
26. Tseng P (1998) Incremental gradient (-projection) method with momentum term and adptive stepsize rule. SIAM J Optim 8:506–531
27. Vogl TP, Mangis JK, Rigler AK, Zink WT, Alkon DL (1988) Accelerating the convergence of the back-propagation method. Biol Cybern 59:257–263
28. Werbos PJ (1994) Supervised learning: can it escape its local minimum? In: Roychowdhury K-YSiu V, Orlitsky A (eds) Theoretical advances in neural computation and learning. Kluwer, Dordrecht, pp 449–461
29. White H (1989) Some asymptotic results for learning in single hidden-layer feedforward network models. J Amer Statist Assoc 84:1003–1013

V

Variable Neighborhood Search Methods

PIERRE HANSEN[1], NENAD MLADENOVIĆ[2]
[1] GERAD and HEC Montréal, Montreal, Canada
[2] GERAD and School of Mathematics, Brunel University, Uxbridge, UK

MSC2000: 9008, 90C59, 90C27, 90C26

Article Outline

Introduction
Background
  Variable Metric Method
  Local Search
Basic Schemes
  Variable Neighborhood Descent
  Reduced VNS
  Basic VNS
  General VNS
  Skewed VNS
  Some Extensions of Basic VNS
  Variable Neighborhood Decomposition Search
  Parallel VNS
  Primal–Dual VNS
  Variable Neighborhood Formulation Space Search
Applications
References

Introduction

Variable neighborhood search (VNS) is a metaheuristic, or framework for building heuristics, aimed at solving combinatorial and global optimization problems. Its basic idea is systematic change of a neighborhood combined with a local search. Since its inception, VNS has undergone many developments and been applied in numerous fields. We review here the basic rules of VNS and of its main extensions. Moreover, some of the most successful applications are briefly summarized. Pointers to many other ones are given in the reference list.

A deterministic optimization problem may be formulated as

$$\min\{f(x)|x \in X, X \subseteq S\}, \tag{1}$$

where S, X, x and f denote respectively the *solution space* and *feasible set*, a *feasible solution* and a real-valued *objective function*, respectively. If S is a finite but large set, a *combinatorial optimization* problem is defined. If $S = R^n$, we talk about *continuous optimization*. A solution $x^* \in S$ is *optimal* if

$$f(x^*) \leq f(x), \ \forall x \in S;$$

an *exact algorithm* for problem (1), if one exists, finds an optimal solution x^*, together with the proof of its optimality, or shows that there is no feasible solution, i. e., $S = \emptyset$. Moreover, in practice, the time to do so should be finite (and not too large); if one deals with a continuous function one must admit a degree of tolerance i. e., stop when a feasible solution x^* has been found such that

$$f(x^*) < f(x) + \varepsilon, \ \forall x \in S \ or$$
$$\frac{f(x^*) - f(x)}{f(x^*)} < \varepsilon, \ \forall x \in S$$

for some small positive ε.

Many practical instances of problems of the form (1), arising in operations research and other fields, are too large for an exact solution to be found in reasonable time. It is well known from complexity theory [46,85] that thousands of problems are nondeterministic polynomial-time hard (*NP-hard*), that no algorithm with a number of steps polynomial in the size

of the instances is known for solving any of them and that finding one would entail obtaining one for each and all of them. Moreover, in some cases where a problem admits a polynomial algorithm, the power of this polynomial may be so large that realistic size instances cannot be solved in reasonable time in the worst case, and sometimes also in the average case or most of the time.

So one is often forced to resort to *heuristics*, which yield quickly an approximate solution, or sometimes an optimal solution but without proof of its optimality. Some of these heuristics have a worst-case guarantee, i. e., the solution x_h obtained satisfies

$$\frac{f(x_h) - f(x)}{f(x_h)} \leq \varepsilon, \ \forall x \in X \tag{2}$$

for some ε, which is, however, rarely small. Moreover, this ε is usually much larger than the error observed in practice and may therefore be a bad guide in selecting a heuristic. In addition to avoiding excessive computing time, heuristics address another problem: local optima. A local optimum x_L of (1) is such that

$$f(x_L) \leq f(x), \ \forall x \in N(x_L) \cap X\,, \tag{3}$$

where $N(x_L)$ denotes a neighborhood of x_L (ways to define such a neighborhood will be discussed later). If there are many local minima, the range of values they span may be large. Moreover, the globally optimum value $f(x^*)$ may differ substantially from the average value of a local minimum, or even from the best such value among many, obtained by some simple heuristic such as multistart (a phenomenon called the Tchebycheff catastrophe in [7]). There are, however, many ways to get out of local optima and, more precisely, the valleys which contain them (or set of solutions from which the descent method under consideration leads to them).

Metaheuristics are general frameworks to build heuristics for combinatorial and global optimization problems. For discussion of the best known of them the reader is referred to surveys [17,49,91]. Some of the many successful applications of metaheuristics are also mentioned there.

Variable neighborhood search (VNS) [55,56,59,78] is a metaheuristic which exploits systematically the idea of neighborhood change, both in descent to local minima and in escape from the valleys which contain them. VNS relies heavily upon the following observations:

Fact 1 A local minimum with respect to one neighborhood structure is not necessarily so for another;

Fact 2 A global minimum is a local minimum with respect to all possible neighborhood structures;

Fact 3 For many problems local minima with respect to one or several neighborhoods are relatively close to each other.

This last observation, which is empirical, implies that a local optimum often provides some information about the global one. This may, for instance, be several variables with the same value in both. However, it is usually not known which ones are such. An organized study of the neighborhood of this local optimum is therefore in order, until a better one is found.

Unlike many other metaheuristics, the basic schemes of VNS and its extensions are simple and require few, and sometimes no parameters. Therefore in addition to providing very good solutions, often in simpler ways than other methods, VNS gives insight into the reasons for such a performance, which in turn can lead to more efficient and sophisticated implementations.

Background

VNS embeds a local search heuristic for solving combinatorial and global optimization problems. There are predecessors of this idea. It allows change of the neighborhood structures within this search. In this section we give a brief introduction to the variable metric algorithm for solving continuous convex problems and local search heuristics for solving combinatorial and global optimization problems.

Variable Metric Method

The variable metric method for solving unconstrained continuous optimization problem (1) was suggested by Davidon [27] and Fletcher and Powell [43]. The idea is to change the metric (and thus the neighborhood) in each iteration such that the search direction (steepest descent with respect to the current metric) adapts better to the local shape of the function. In the first iteration a Euclidean unit ball in n-dimensional space is used

and the steepest descent (antigradient) direction found; in the next iterations, ellipsoidal balls are used and the steepest descent direction with respect to a new metric is obtained after a linear transformation. The purpose of such changes is to build up, iteratively, a good approximation to the inverse of the Hessian matrix A^{-1} of f, that is, to construct a sequence of matrices H_i with the property,

$$\lim_{t \leftarrow \infty} H_i = A^{-1} .$$

In the convex quadratic programming case the limit is achieved after n iterations instead of ∞. In that way the so-called Newton search direction is obtained. The advantages are (1) it is not necessary to find the inverse of the Hessian (which requires $O(n^3)$ operations) in each iteration; (2) the second -order information is not demanded.

Assume that the function $f(x)$ is approximated by its Taylor series

$$f(x) = \frac{1}{2} x^T A\, x - b^T x \tag{4}$$

with positive-definite matrix A ($A > 0$). Applying the first-order condition $\nabla f(x) = Ax - b = 0$, we have $Ax_{\text{opt}} = b$, where x_{opt} is a minimum point. At the current point we have $A\, x_i = \nabla f(x_i) + b$. We will not rigorously derive here the Davidon–Fletcher–Powell algorithm for taking H_i to H_{i+1}. Let us just mention that subtracting these two equations and multiplying (from the left) by the inverse matrix A^{-1}, we have

$$x_{\text{opt}} - x_i = -A^{-1} \nabla f(x_i).$$

Subtracting the latest equation at x_{i+1} from the same equation at x_i gives

$$x_{i+1} - x_i = -A^{-1}(\nabla f(x_{i+1}) - \nabla f(x_i)) . \tag{5}$$

Having made the step from x_i to x_{i+1}, we might reasonably want to require that the new approximation H_{i+1} satisfies (5) as if it were actually A^{-1}, that is,

$$x_{i+1} - x_i = -H_{i+1}(\nabla f(x_{i+1}) - \nabla f(x_i)) . \tag{6}$$

We might also assume that the updating formula for matrix H_i should be of the form $H_{i+1} = H_i + U$, where U is a correction. It is possible to get different updating formulas for U and thus for H_{i+1}, keeping H_{i+1} positive-definite ($H_{i+1} > 0$). In fact, there exists a whole family of updates, the Broyden family. From practical experience the Broyden–Fletcher–Goldfarb–Shanno method seem to be most popular (see [48] for details). Pseudo-code is given in Algorithm 1.

```
Function VarMetric(x);
let x ∈ R^n be an initial solution
H ← I; g ← −∇f(x)
for i = 1 to n do
    α* ← arg min_α f(x + α · Hg)
    x ← x + α* · Hg; g ← −∇f(x)
    H ← H + U
end
```

Variable Neighborhood Search Methods, Algorithm 1
Variable metric algorithm

```
Function BestImprovement(x)
repeat
    x′ ← x
    x ← arg min_{y∈N(x)} f(y)
until (f(x) ≥ f(x′)) ;
```

Variable Neighborhood Search Methods, Algorithm 2
Best improvement (steepest descent) heuristic

From the above one can conclude that even in solving a convex program a change of metric, and thus change of the neighborhoods induced by that metric, may produce more efficient algorithms. Thus, using the idea of neighborhood change for solving NP-hard problems could well lead to even greater benefits.

Local Search

A *local search* heuristic consists in choosing an initial solution x, finding a direction of descent from x, within a neighborhood $N(x)$, and moving to the minimum of $f(x)$ within $N(x)$ along that direction; if there is no direction of descent, the heuristic stops, and otherwise it is iterated. Usually the steepest descent direction, also referred to as *best improvement*, is used. This set of rules is summarized in Algorithm 2, where we assume that an initial solution x is given. The output consists of a local minimum, also denoted with x, and its value.

Observe that a neighborhood structure $N(x)$ is defined for all $x \in X$; in discrete optimization problems it

```
Function FirstImprovement(x)
1  repeat
2     x' ← x;  i ← 0
3     repeat
4        i ← i + 1
5        x ← arg min{f(x), f(x_i)},  x_i ∈ N(x)
      until (f(x) < f(x_i) or i = |N(x)|) ;
   until (f(x) ≥ f(x')) ;
```

Variable Neighborhood Search Methods, Algorithm 3
First improvement heuristic

usually consists of all vectors obtained from x by some simple modification, e. g., complementing one or two components of a 0–1 vector. Then, at each step, the neighborhood $N(x)$ of x is explored completely. As this may be time-consuming, an alternative is to use the *first descent* heuristic. Vectors $x_i \in N(x)$ are then enumerated systematically and a move is made as soon as a descent direction is found. This is summarized in Algorithm 3.

Basic Schemes

Let us denote with $\mathcal{N}_k$, $(k = 1, \ldots, k_{\max})$, a finite set of preselected neighborhood structures, and with $\mathcal{N}_k(x)$ the set of solutions in the kth neighborhood of x. We will also use notation $\mathcal{N}'_k$, $k = 1, \ldots, k'_{\max}$, when describing local descent. Neighborhoods $\mathcal{N}_k$ or $\mathcal{N}'_k$ may be induced from one or more metric (or quasi-metric) functions introduced into a solution space S. An *optimal solution* x_{opt} (or global minimum) is a feasible solution where a minimum of (1) is reached. We call $x' \in X$ a *local minimum* of (1) with respect to $\mathcal{N}_k$, if there is no solution $x \in \mathcal{N}_k(x') \subseteq X$ such that $f(x) < f(x')$.

In order to solve (1) by using several neighborhoods, facts 1–3 can be used in three different ways: (i) deterministic; (ii) stochastic; (iii) both deterministic and stochastic. We first give in Algorithm 4 steps of the neighborhood change function that will be used later.

The function `NeighborhoodChange()` compares the new value $f(x')$ with the incumbent value $f(x)$ obtained in the neighborhood k (line 1). If an improvement is obtained, k is returned to its initial value and the new incumbent updated (line 2). Otherwise, the next neighborhood is considered (line 3).

```
Function NeighborhoodChange(x, x', k)
1  if f(x') < f(x) then
2     x ← x'; k ← 1 /* Make a move */
   else
3     k ← k + 1 /* Next neighborhood */
   end
```

Variable Neighborhood Search Methods, Algorithm 4
Neighborhood change or move or not function

```
Function VND (x, k'_max)
1  repeat
2     k ← 1
3     repeat
4        x' ← arg min_{y∈N'_k(x)} f(x)
         /* Find the best neighbor in N_k(x) */
5        NeighborhoodChange(x, x', k)
         /* Change neighborhood */
      until k = k'_max ;
   until no improvement is obtained ;
```

Variable Neighborhood Search Methods, Algorithm 5
Steps of the basic variable neighborhood descent (*VND*)

Variable Neighborhood Descent

The *variable neighborhood descent* (VND) method is obtained if the change of neighborhoods is performed in a deterministic way. Its steps are presented in Algorithm 5. In the descriptions of all algorithms that follow we assume that an initial solution x is given.

Most local search heuristics use in their descents a single or sometimes two neighborhoods ($k'_{\max} \leq 2$). Note that the final solution should be a local minimum with respect to all $k'_{\max}$ neighborhoods, and thus the chances of reaching a global one are larger when using VND than with a single neighborhood structure. Beside this *sequential* order of neighborhood structures in VND above, one can develop a *nested* strategy. Assume, e. g., that $k'_{\max} = 3$; then a possible nested strategy is: perform VND from Fig. 8 for the first two neighborhoods, in each point x' that belongs to the third ($x' \in \mathcal{N}_3(x)$). Such an approach is applied, e. g., in [14,57].

Reduced VNS

The *reduced* VNS (RVNS) method is obtained if random points are selected from $\mathcal{N}_k(x)$ and no descent

```
   Function RVNS (x, k_max, t_max)
1  repeat
2      k ← 1
3      repeat
4          x' ← Shake(x, k)
5          NeighborhoodChange(x, x', k)
       until k = k_max ;
6      t ← CpuTime()
   until t > t_max ;
```

Variable Neighborhood Search Methods, Algorithm 6
Steps of the reduced variable neighborhood search (*RVNS*)

is made. Rather, the values of these new points are compared with that of the incumbent and updating takes place in the case of improvement. We assume that a stopping condition has been chosen, among various possibilities, e. g., the maximum CPU time allowed $t_{\max}$, or the maximum number of iterations between two improvements. To simplify the description of the algorithms we always use $t_{\max}$ below. Therefore, RVNS uses two parameters: $t_{\max}$ and $k_{\max}$. Its steps are presented in Algorithm 6. With the function `Shake` represented in line 4, we generate a point x' at random from the kth neighborhood of x, i. e., $x' \in \mathcal{N}_k(x)$.

RVNS is useful for very large instances for which local search is costly. It is observed that the best value for the parameter $k_{\max}$ is often 2. In addition, the maximum number of iterations between two improvements is usually used as a stopping condition. RVNS is akin to a Monte Carlo method, but is more systematic (see [80], where results obtained by RVNS were 30% better than those of the Monte Carlo method in solving a continuous min–max problem). When applied to the p-median problem, RVNS gave equally good solutions as the *fast interchange* heuristic of [104] in 20–40 times less time [60].

Basic VNS

The *basic VNS* method [78] combines deterministic and stochastic changes of neighborhood. Its steps are given in Algorithm 7.

Often successive neighborhoods $\mathcal{N}_k$ will be nested. Observe that point x' is generated at random in step 4 in order to avoid cycling, which might occur if any deterministic rule was applied. In step 5 the first improvement local search (Algorithm 3) is usually adopted;

```
   Function VNS (x, k_max, t_max)
1  repeat
2      k ← 1
3      repeat
4          x' ← Shake(x, k)
           /* Shaking */
5          x'' ← FirstImprovement(x')
           /* Local search */
6          NeighborhoodChange(x, x'', k)
           /* Change neighborhood */
       until k = k_max ;
7      t ← CpuTime()
   until t > t_max ;
```

Variable Neighborhood Search Methods, Algorithm 7
Steps of the basic variable neighborhood search (*VNS*)

however, it can be replaced with best improvement (Algorithm 2).

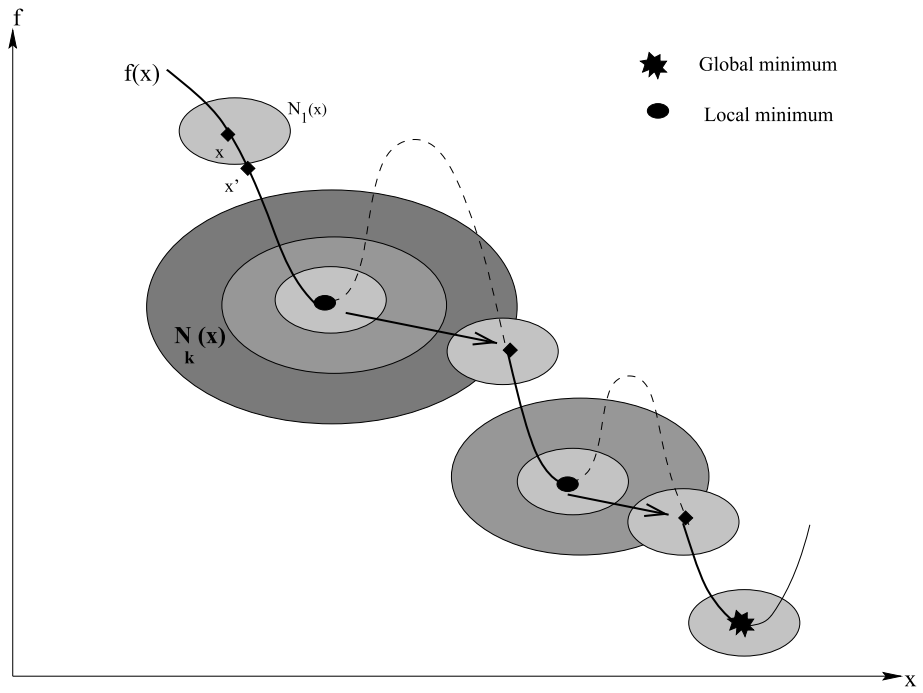
General VNS

Note that the local search step 5 may be also replaced by VND (Algorithm 4). Using this general VNS (VNS/VND) approach led to the most successful applications reported [3,14,18,20,21,22,23,57,61,94,96]. The steps of the general VNS are given in Algorithm 8.

Skewed VNS

The skewed VNS (SVNS) method [53] addresses the problem of exploring valleys far from the incumbent solution. Indeed, once the best solution in a large region has been found it is necessary to go quite far to obtain an improved one. Solutions drawn at random in far-away neighborhoods may differ substantially from the incumbent and VNS can then degenerate, to some extent, into the multistart heuristic (in which descents are made iteratively from solutions generated at random, and which is known not to be very efficient). So some compensation for distance from the incumbent must be made and a scheme called SVNS is proposed for that purpose. Its steps are presented in Algorithms 9 and 10, where the `KeepBest`(x, x') function simply keeps the better between x and x': **if** $f(x') < f(x)$ **then** $x \leftarrow x'$.

SVNS makes use of a function $\rho(x, x'')$ to measure the distance between the incumbent solution x and the local optimum found x''. The distance used to define



Variable Neighborhood Search Methods, Figure 1
General VNS (*GVNS*)

Function GVNS $(x, k'_{max}, k_{max}, t_{max})$
1 **repeat**
2 $k \leftarrow 1$
3 **repeat**
4 $x' \leftarrow$ Shake(x, k)
5 $x'' \leftarrow$ VND(x', k'_{max})
6 NeighborhoodChange(x, x'', k)
until $k = k_{max}$;
7 $t \leftarrow$ CpuTime ( )
until $t > t_{max}$;

Variable Neighborhood Search Methods, Algorithm 8
Steps of the GVNS

Function NeighborhoodChangeS(x, x'', k, α)
1 **if** $f(x'') - \alpha\rho(x, x'') < f(x)$ **then**
2 $x \leftarrow x''$; $k \leftarrow 1$
else
3 $k \leftarrow k + 1$
end

Variable Neighborhood Search Methods, Algorithm 9
Steps of neighborhood change for the skewed VNS (*SVNS*)

Function SVNS $(x, k_{max}, t_{max}, \alpha)$
1 **repeat**
2 $k \leftarrow 1$; $x_{best} \leftarrow x$
3 **repeat**
4 $x' \leftarrow$ Shake(x, k)
5 $x'' \leftarrow$ FirstImprovement(x')
6 KeepBest (x_{best}, x)
7 NeighborhoodChangeS(x, x'', k, α)
until $k = k_{max}$;
8 $x \leftarrow x_{best}$
9 $t \leftarrow$ CpuTime()
until $t > t_{max}$;

Variable Neighborhood Search Methods, Algorithm 10
Steps of the SVNS

the $\mathcal{N}_k$, as in the above examples, could be used also for this purpose. The parameter α must be chosen in order to accept exploring valleys far from x when $f(x'')$ is larger than $f(x)$ but not too much larger (otherwise one will always leave x). A good value is to be found experimentally in each case. Moreover, in order to avoid frequent moves from x to a close solution, one may take

a large value for α when $\rho(x, x'')$ is small. More sophisticated choices for a function of $\alpha\rho(x, x'')$ could be made through some learning process.

Some Extensions of Basic VNS

Several easy ways to extend the basic VNS are now discussed. The basic VNS is a descent, first improvement method with randomization. Without much additional effort it could be transformed into a descent–ascent method: in the `NeighborhoodChange()` function set also $x \leftarrow x''$ with some probability even if the solution is worse than the incumbent (or the best solution found so far). It could also be changed into a best improvement method: make a move to the best neighborhood k^* among all $k_{\max}$ of them. Its steps are given in Algorithm 11.

Another variant of the basic VNS could be to find a solution x' in step 4 as the best among b (a parameter) randomly generated solutions from the kth neighborhood. There are two possible variants of this extension: (i) perform only one local search from the best point among b; (ii) perform all b local searches and then choose the best. We now give an algorithm of a second type suggested by Fleiszar and Hindi [41]. There, the value of parameter b is set to k. In that way no new parameter is introduced (see Algorithm 12).

It is also possible to introduce $k_{\min}$ and k_{step}, two parameters that control the change of the neighborhood process: in the previous algorithms instead of $k \leftarrow 1$ set $k \leftarrow k_{\min}$ and instead of $k \leftarrow k + 1$ set $k \leftarrow k + k_{\text{step}}$. Steps of jump VNS are given in Algorithms 13 and 14.

```
   Function BI-VNS (x, k_max, t_max)
1  repeat
2      k ← 1  x_best ← x
3      repeat
4          x' ← Shake(x, k)
5          x'' ← FirstImprovement(x')
6          KeepBest(x_best, x'')
7          k ← k + 1
       until k = k_max ;
8      x ← x_best
9      t ← CpuTime()
   until t > t_max ;
```

Variable Neighborhood Search Methods, Algorithm 11
Steps of the basic best improvement VNS (*BI-VNS*)

```
   Function FH-VNS (x, k_max, t_max)
1  repeat
2      k ← 1
3      repeat
4          for ℓ = 1 to k do
5              x' ← Shake(x, k)
6              x'' ← FirstImprovement(x')
7              KeepBest(x, x'')
           end
8          NeighborhoodChange(x, x'', k)
       until k = k_max ;
9      t ← CpuTime()
   until t > t_max ;
```

Variable Neighborhood Search Methods, Algorithm 12
Steps of the Fleszar–Hindi extension of the basic VNS (*FH-VNS*)

Variable Neighborhood Decomposition Search

While the basic VNS is clearly useful for approximate solution of many combinatorial and global optimization problems, it remains difficult or takes a long time to solve very large instances. Often the size of the problems considered is limited in practice by the tools available to solve them more than by the needs of potential users of these tools. Hence, improvements appear to be highly desirable. Moreover, when heuristics are applied to really large instances their strengths and weaknesses become clearly apparent. Three improvements of the basic VNS for solving large instances are now considered.

The variable neighborhood decomposition search (VNDS) method [60] extends the basic VNS into a two-level VNS scheme based upon decomposition of the problem. Its steps are presented in Algorithm 15, where t_d is an additional parameter and represents the running time given for solving decomposed (smaller-sized) problems by VNS.

For ease of presentation, but without loss of generality, we assumed that the solution x represents the set of some elements. In step 4 we denote with y a set of k solution attributes present in x' but not in x ($y = x' \setminus x$). In step 5 we find the local optimum y' in the space of y; then we denote with x'' the corresponding solution in the whole space S ($x'' = (x' \setminus y) \cup y'$). We noticed

```
   Function J-VNS (x, k_min, k_step, k_max, t_max)
1  repeat
2      k ← k_min
3      repeat
4          x' ← Shake(x, k)
5          x'' ← FirstImprovement(x')
6          NeighborhoodChangeJ(x, x'', k, k_min, k_step)
       until k = k_max ;
7      t ← CpuTime()
   until t > t_max ;
```

Variable Neighborhood Search Methods, Algorithm 13
Steps of the jump VNS (*J-VNS*)

```
   Function NeighborhoodChangeJ (x, x', k, k_min, k_step)
1  if f(x') < f(x) then
2      x ← x'; k ← k_min
   else
3      k ← k + k_step
   end
```

Variable Neighborhood Search Methods, Algorithm 14
Neighborhood change or move or not function

```
   Function VNDS (x, k_max, t_max, t_d)
1  repeat
2      k ← 2
3      repeat
4          x' ← Shake (x, k); y ← x' \ x
5          y' ← VNS(y, k, t_d); x'' = (x' \ y) ∪ y'
6          x''' ← FirstImprovement(x'')
7          NeighborhoodChange(x, x''', k)
       until k = k_max ;
   until t > t_max ;
```

Variable Neighborhood Search Methods, Algorithm 15
Steps of variable neighborhood decomposition search (*VNDS*)

that exploiting some *boundary effects* in a new solution can significantly improve the solution quality. That is why, in step 6, we find the local optimum x''' in the whole space S using x'' as an initial solution. If this is time-consuming, then at least a few local search iterations should be performed.

VNDS can be viewed as embedding the classical successive approximation scheme (which has been used in combinatorial optimization at least since the 1960s [50]) in the VNS framework.

Parallel VNS

Parallel VNS methods are another extension. Several ways for parallelizing VNS have recently been proposed [26,71] in solving the p-median problem. In [71] three of them were tested: (i) parallelize local search; (ii) augment the number of solutions drawn from the current neighborhood and do local search in parallel from each of them and (iii) do the same as for method 2 but update the information on the best solution found. The second version gave the best results. It was shown in [26] that assigning different neighborhoods to each processor and interrupting their work as soon as an improved solution is found gives very good results: the best known solutions have been found on several large instances taken from TSP-LIB [92]. Three parallel VNS strategies are also suggested for solving the traveling purchaser problem in [83].

Primal–Dual VNS

For most modern heuristics the difference in value between the optimal solution and the one obtained is completely unknown. Guaranteed performance of the primal heuristic may be determined if a lower bound on

Function PD-VNS $(x, k'_{max}, k_{max}, t_{max})$
1 BVNS $(x, k'_{max}, k_{max}, t_{max})$
/* Solve primal by VNS */
2 DualFeasible(x, y)
/* Find (infeasible) dual such that $f_P = f_D$ */
3 DualVNS(y)
/* Use VNS do decrease infeasibility */
4 DualExact(y)
/* Find exact (relaxed) dual */
5 BandB(x, y)
/* Apply branch-and-bound method */

Variable Neighborhood Search Methods, Algorithm 16
Steps of the basic primal–dual VNS (*PD-VNS*)

the objective function value is known. To that end the standard approach is to relax the integrality condition on the primal variables, based on a mathematical programming formulation of the problem. However, when the dimension of the problem is large, even the relaxed problem may be impossible to solve exactly by standard commercial solvers. Therefore, it looks to be a good idea to solve dual relaxed problems heuristically as well. In that way we get guaranteed bounds on the primal heuristics performance. The next problem arises if we want to get exact solution within a branch-and-bound framework since having the approximate value of the relaxed dual does not allow us to branch in an easy way, e. g., exploiting complementary slackness conditions. Thus, the exact value of the dual is necessary.

In primal–dual VNS [52] we propose one possible general way to get both the guaranteed bounds and the exact solution. Its steps are given in Algorithm 16.

In the first stage a heuristic procedure based on VNS is used to obtain a near-optimal solution. In [52] we showed that VNS with decomposition is a very powerful technique for large-scale simple plant location problems (SUPPL) up to 15,000 facilities × 15,000 users. In the second phase, our approach is designed to find an exact solution of the relaxed dual problem. For solving SPLP, this is accomplished in three stages: (i) find an initial dual solution (generally infeasible) using the primal heuristic solution and complementary slackness conditions; (ii) improve the solution by applying VNS on the unconstrained nonlinear form of the dual; (iii) finally, solve the dual exactly using a customized "sliding simplex" algorithm that applies "windows" on the dual variables to reduce substantially the size of the problem. In all problems tested, including instances much larger than previously reported in the literature, our procedure was able to find the exact dual solution in reasonable computing time. In the third and final phase armed with tight upper and lower bounds, obtained, respectively, from the heuristic primal solution in phase 1 and the exact dual solution in phase 2, we apply a standard branch-and-bound algorithm to find an optimal solution of the original problem. The lower bounds are updated with the dual sliding simplex method and the upper bounds whenever new integer solutions are obtained at the nodes of the branching tree. In this way we were able to solve exactly problem instances with up to $7,000 \times 7,000$ for uniform fixed costs and $15{,}000 \times 15{,}000$ otherwise.

Variable Neighborhood Formulation Space Search

Traditional ways to tackle an optimization problem consider a given formulation and search in some way through its feasible set S. The fact that the same problem may often be formulated in different ways allows us to extend search paradigms to include jumps from one formulation to another. Each formulation should lend itself to some traditional search method, its "local search" that works totally within this formulation, and yields a final solution when started from some initial solution. Any solution found in one formulation should easily be translatable to its equivalent formulation in any other formulation. We may then move from one formulation to another using the solution resulting from the former's local search as the initial solution for the latter's local search. Such a strategy will of course only be useful ff local searches in different formulations behave differently.

This idea was recently investigated in [81] using an approach that systematically changes formulations for solving circle packing problems (CPP). It is shown there that a stationary point of a nonlinear programming formulation of CPP in Cartesian coordinates is not necessarily also a stationary point in a polar coordinate system. The method *reformulation descent* that alternates between these two formulations until the final solution is stationary with respect to both is suggested. The results obtained were comparable with the best known values, but they were achieved some 150

Function FormulationChange$(x, x', \phi, \phi', \ell)$
1 **if** $f(\phi', x') < f(\phi, x)$ **then**
2 $\quad \phi \leftarrow \phi'; x \leftarrow x'; \ell \leftarrow \ell_{min}$
else
3 $\quad \ell \leftarrow \ell + \ell_{step}$
end

Variable Neighborhood Search Methods, Algorithm 17
Formulation change function

Function VNFSS(x, ϕ, ℓ_{max})
1 **repeat**
2 $\quad \ell \leftarrow 1$
$\quad$ /* Initialize formulation in $\mathcal{F}$ */
3 $\quad$ **while** $\ell \leq \ell_{max}$ **do**
4 $\quad\quad$ ShakeFormulation(x,x',ϕ,ϕ',ℓ) /* Take $(\phi',x') \in (N_\ell(\phi),\mathcal{N}(x))$ at random*/
5 $\quad\quad$ FormulationChange(x,x',ϕ,ϕ',ℓ) /* Change formulation */
$\quad$ **end**
until *some stopping condition is met* ;

Variable Neighborhood Search Methods, Algorithm 18
Reduced variable neighborhood formulation space search (*VNFSS*)

times faster than by an alternative single formulation approach. In that same paper the idea suggested above of *formulation space search* was also introduced, using more than two formulations. Some research in that direction has been reported in [64,75,84]. One algorithm that uses the variable neighborhood idea in searching through the formulation space is given in Algorithms 17 and 18.

In Fig. 2 we consider the CPP case with $n = 50$. The set consists of all mixed formulations, in which some circle centers are given in Cartesian coordinates, while the others are given in polar coordinates. The distance between two formulations is then the number of centers whose coordinates are expressed in different systems in each formulation. Our formulation space search starts with the reformulation descent solution i. e., with $r_{\text{curr}} = 0.121858$. The values of k_{min} and k_{step} are set to 3 and the value of k_{max} is set to $n = 50$. We did not get an improvement with $k_{\text{curr}} = 3, 6$ and 9. The next improvement was obtained for $k_{\text{curr}} = 12$. This means that a "mixed" formulation with 12 polar and 38 Cartesian coordinates is used. Then we turn again to the formulation with three randomly chosen circle centers, which was unsuccessful, but obtained a better solution with six, etc. After 11 improvements we ended up with a solution with radius $r_{\text{max}} = 0.125798$.

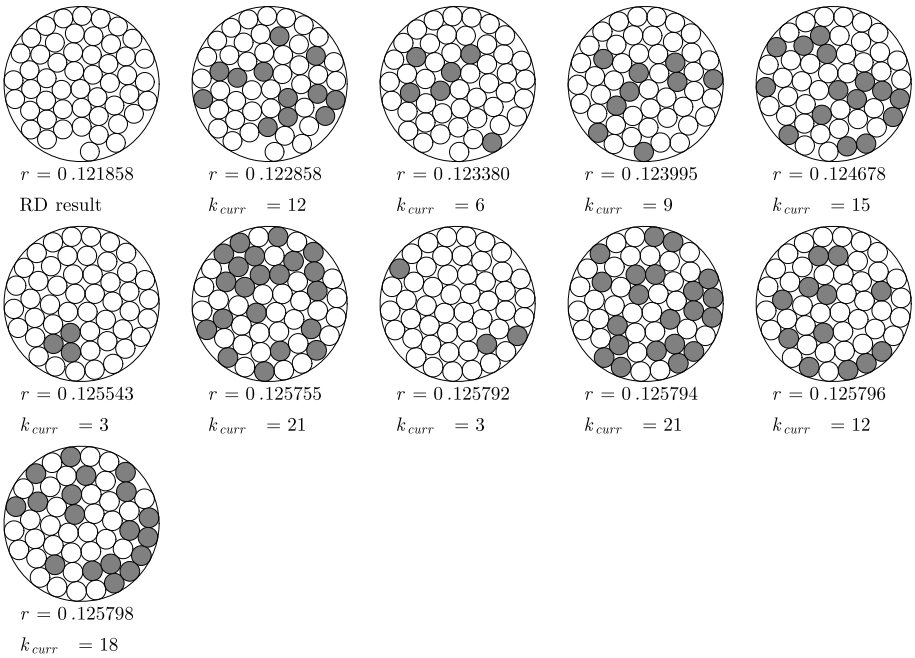
Applications

Applications of VNS or of hybrids of VNS and other metaheuristics are diverse and numerous. We next review some of them. Considering first industrial applications, the oil industry provided many problems. These include scheduling of walkover rigs for Petrobras [2], the design of an offshore pipeline network [13] and the pooling problem [5]. Other design problems include cable layout [24], synchronous digital hierarchy/wavelength-division multiplexing networks [73], surface acoustic wave filters [100], topological design of a yottabit-per-second lattice network [29], the ring star problem [31], distribution networks [67] and supply chain planning [68]. Location problems have also attracted much attention. Among discrete models the p-median has been the most studied [15,26,44,54,71,76] together with its variants [32,37]; the p-center problems [77] and the maximum capture problem [10] have also been examined. Among continuous models the multisource Weber problem is addressed in [14]. Use of VNS to solve the quadratic assignment problem is discussed in [33,34,106].

VNS proved to be a very efficient tool in cluster analysis. In particular, the J-means heuristic combined with VNS appears to be the state of the art for heuristic solution of minimum sum-of-square clustering [8,9,57]. Combined with stabilized column generation [36] it leads to the presently most efficient exact algorithm for that problem [35].

Other combinatorial optimization problems on graphs to which VNS has been applied include the degree-constrained spanning tree problem [16,94,102], the clique problem [62], the max-cut problem [38], the median cycle problem [86] and vertex coloring [6,64]. Some further discrete combinatorial optimization problems, unrelated to graphs, to which VNS has been applied are the linear ordering problem [45], bin packing [42] and the multidimensional knapsack problem [89].

Heuristics may help to find a feasible solution or an improved and possibly optimal solution to large and difficult mixed-integer programs. The local branching



Variable Neighborhood Search Methods, Figure 2
Reduced formulation space search for the circle packing problem and $n = 50$

method of Fischetti and Lodi [39] does that, in the spirit of VNS. For further developments see [40,61].

Timetabling and related manpower organization problems can be well solved with VNS. They include the team-orienteering problem [4], the examination proximity problem [25], the design of balanced MBA student teams [30] and apportioning the European Parliament [103].

Various vehicle-routing problems were solved by VNS or hybrids [12,63,66,87,88,93,97,105]. This led to interesting developments such as the reactive VNS of Braysy [12]. Use of VNS to solve machine scheduling problems was studied in many papers [11,27,28,41,51, 70,82,90,98].

Miscellaneous other problems solved with VNS include study of the dynamics of handwriting [19], the capacitated lot-sizing problem with setup times [65], the location-routing problem with nonlinear costs [72] and continuous time-constrained optimization problems [101].

In all these applications VNS is used as an optimization tool. It can also lead to results in "discovery science," i. e., help in the development of theories. This has been done for graph theory in a long series of papers reporting on development and applications of the system AutoGraphiX [20,21]. See also [22,23] for applications to chemistry and [1] for a survey with many further references. This system addresses the following problems:

- Find a graph satisfying given constraints;
- Find optimal or near-optimal graphs for an invariant subject to constraints;
- Refute a conjecture;
- Suggest a conjecture (or sharpen one);
- Suggest a proof.

This is done by applying VNS to find extremal graphs using a VND with many neighborhoods defined by modifications of the graphs such as removal or addition of an edge, rotation of an edge, and so forth. Once a set of extremal graphs, parametrized by their order, has been found their properties are explored with various data-mining techniques and lead to conjectures, refutations and simple proofs or ideas of proof.

Note finally that a series of papers on VNS presented at the 18th EURO Mini-Conference on Variable Neighborhood Search, Tenerife, November 2005, will appear soon in special issues of the *European Journal of Operational Research*, *IMA Journal of Management Mathematics* and *Journal of Heuristics*.

References

1. Aouchiche M, Caporossi G, Hansen P, Laffay M (2005) AutoGraphiX: A Survey. Electron Notes Discret Math 22: 515–520
2. Aloise DJ, Aloise D, Rocha CTM, Ribeiro CC, Ribeiro JC, Moura LSS (2006) Scheduling workover rigs for onshore oil production. Discret Appl Math 154(5):695–702
3. Andreatta A, Ribeiro C (2002) Heuristics for the phylogeny problem. J Heuristics 8(4):429–447
4. Archetti C, Hertz A, Speranza MG (2007) Metaheuristics for the team orienteering problem. J Heuristics 13(1): 49–76
5. Audet C, Brimberg J, Hansen P, Mladenović N (2004) Pooling problem: alternate formulation and solution methods. Manag Sci 50:761–776
6. Avanthay C, Hertz A, Zufferey N (2003) A variable neighborhood search for graph coloring. Eur J Oper Res 151(2): 379–388
7. Baum EB (1987) Toward practical 'neural' computation for combinatorial optimization problems. In: AIP Conference Proceedings 151 on Neural Networks for Computing, Snowbird, USA, March 1987, pp 53–58
8. Belacel N, Cuperlovic-Culf M, Ouellette R (2004) Fuzzy J-Means and VNS methods for clustering genes from microarray data. Bioinformatics 20(11):1690–1701
9. Belacel N, Hansen P, Mladenović N (2002) Fuzzy J-Means: a new heuristic for fuzzy clustering. Pattern Recognit 35(10):2193–2200
10. Benati S, Hansen P (2002) The maximum capture problem with random utilities: Problem formulation and algorithms. Eur J Oper Res 143(3):518–530
11. Blazewicz J, Pesch E, Sterna M, Werner F (2005) Metaheuristics for late work minimization in two-machine flow shop with common due date. KI2005: Advances in Artificial Intelligence, Proceedings Lecture Notes in Artificial Intelligence 3698:222–234 (2005)
12. Braysy O (2003) A reactive variable neighborhood search for the vehicle routing problem with time windows. INFORMS J Comput 15(4):347–368
13. Brimberg J, Hansen P, Lih KW, Mladenović N, Breton M (2003) An oil pipeline design problem. Oper Res 51(2):228–239
14. Brimberg J, Hansen P, Mladenović N, Taillard É (2000) Improvements and comparison of heuristics for solving the Multisource Weber problem. Oper Res 48(3):444–460
15. Brimberg J, Mladenović N (1996) A variable neighborhood algorithm for solving the continuous location-allocation problem. Stud Locat Anal 10:1–12
16. Brimberg J, Urosević D, Mladenović N (2006) Variable neighborhood search for the vertex weighted k-cardinality tree problem. Eur J Oper Res 171(1):74–84
17. Burke E, Kendall G (2005) Search Methodologies. Introductory tutorials in optimization and decision support techniques. Springer, Berlin
18. Canuto S, Resende M, Ribeiro C (2001) Local search with perturbations for the prize-collecting Steiner tree problem in graphs. Networks 31(3):201–206
19. Caporossi G, Alamargot D, Chesnet D (2004) Using the computer to study the dynamics of the handwriting processes. Discov Sci, Proc Lect Notes Comp Sci 3245:242–254 (2004)
20. Caporossi G, Hansen P (2000) Variable neighborhood search for extremal graphs. 1. The AutoGraphiX system. Discret Math 212:29–44
21. Caporossi G, Hansen P (2004) Variable neighborhood search for extremal graphs. 5. Three ways to automate finding conjectures. Discret Math 276(1–3):81–94
22. Caporossi G, Cvetković D, Gutman I, Hansen P (1999) Variable neighborhood search for extremal graphs. 2. Finding graphs with extremal energy. J Chem Inf Comput Sci 39:984–996
23. Caporossi G, Gutman I, Hansen P (1999) Variable neighborhood search for extremal graphs. IV: Chemical trees with extremal connectivity index. Comput Chem 23(5):469–477
24. Costa MC, Monclar FR, Zrikem M (2002) Variable neighborhood decomposition search for the optimization of power plant cable layout. J Intell Manuf 13(5):353–365
25. Cote P, Wong T, Sabourin R (2005) A hybrid multi-objective evolutionary algorithm for the uncapacitated exam proximity problem. Practice and Theory of Automated Timetabling V Lect Notes Comput Sci 3616:294–312
26. Crainic T, Gendreau M, Hansen P, Mladenović N (2004) Cooperative parallel variable neighborhood search for the p-median. J Heuristics 10:289–310
27. Davidović T (2000) Scheduling heuristic for dense task graphs. Yugoslav J Oper Res 10:113–136
28. Davidović T, Hansen P, Mladenović N (2005) Permutation-based genetic, tabu, and variable neighborhood search heuristics for multiprocessor scheduling with communication delays. Asia-Pacific J Oper Res 22(3):297–326
29. Degila JR, Sansò B (2004) Topological design optimization of a Yottabit-per-second lattice network. IEEE J Sel Areas Commun 22(9):1613–1625
30. Desrosiers J, Mladenović N, Villeneuve D (2005) Design of balanced MBA student teams. J Oper Res Soc 56(1):60–66
31. Dias TCS, De Sousa GF, Macambira EM, Cabral LDAF, Fampa MHC (2006) An efficient heuristic for the ring star problem. Exp Alg, Proc Lect Notes Comput Sci 4007: 24–35
32. Dominguez-Marin P, Nickel S, Hansen P, Mladenović N (2005) Heuristic procedures for solving the Discret Ordered Median Problem. Annals Oper Res 136(1):145–173
33. Drezner Z (2005) The extended concentric tabu for the quadratic assignment problem. Eur J Oper Res 160(2): 416–422
34. Drezner Z, Hahn PM, Taillard ED (2005) Recent advances for the quadratic assignment problem with special em-

phasis on instances that are difficult for meta-heuristic methods. Annals Oper Res 139(1):65–94
35. du Merle O, Hansen P, Jaumard B, Mladenović N (2000) An interior point algorithm for Minimum sum-of-squares clustering. SIAM J Sci Comput 21:1485–1505
36. du Merle O, Villeneuve D, Desrosiers J et al (1999) Stabilized column generation. Discret Math 194(1–3):229–237
37. Fathali J, Kakhki HT (2006) Solving the p-median problem with pos/neg weights by variable neighborhood search and some results for special cases. Eur J Oper Res 170(2):440–462
38. Festa P, Pardalos PM, Resende MGC, Ribeiro CC (2002) Randomized heuristics for the MAX-CUT problem. Optim Methods Softw 17(6):1033–1058
39. Fischetti M, Lodi A (2003) Local branching. Math Program 98(1–3):23–47
40. Fischetti M, Polo C, Scantamburlo M (2004) A local branching heuristic for mixed-integer programs with 2-level variables, with an application to a telecommunication network design problem. Networks 44(2):61–72
41. Fleszar K, Hindi KS (2004) Solving the resource-constrained project scheduling problem by a variable neighborhood search. Eur J Oper Res 155(2):402–413
42. Fleszar K, Hindi KS (2002) New heuristics for one-dimensional bin-packing. Comput Oper Res 29:821–839
43. Fletcher R, Powell MJD (1963) Rapidly convergence method for minimization. Comput J 6:163–168
44. Garcia-Lopez F, Melian-Batista B, Moreno-Perez JA, Moreno-Vega JM (2002) The parallel variable neighborhood search for the p-median problem. J Heuristics 8(3):375–388
45. Garcia CG, Perez-Brito D, Campos V, Marti R (2006) Variable neighborhood search for the linear ordering problem. Comput Oper Res 33(12):3549–3565
46. Garey MR, Johnson DS (1978) Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York
47. Gendreau M, Potvin JY (2005) Metaheuristics in combinatorial optimization. Annals Oper Res 140(1):189–213
48. Gill P, Murray W, Wright M (1981) Practical optimization. Academic Press, London
49. Glover F, Kochenberger G (eds) (2003) Handbook of Metaheuristics. Kluwer, Boston
50. Griffith RE, Stewart RA (1961) A nonlinear programming technique for the optimization of continuous processing systems. Manag Sci 7:379–392
51. Gupta SR, Smith JS (2006) Algorithms for single machine total tardiness scheduling with sequence dependent setups. Eur J Oper Res 175(2):722–739
52. Hansen P, Brimberg J, Urosevic D, Mladenovic N (2007) Primal-Dual Variable Neighborhood Search for the Simple Plant Location Problem. INFORMS J Comput, doi:10.1287/ijoc.1060.0196
53. Hansen P, Jaumard B, Mladenović N, Parreira A (2000) Variable neighborhood search for Weighted maximum satisfiability problem. Les Cahiers du GERAD G–2000–62, HEC Montréal, Canada
54. Hansen P, Mladenović N (1997) Variable neighborhood search for the *p*-median. Locat Sci 5:207–226
55. Hansen P, Mladenović N (1999) An introduction to variable neighborhood search. In: Voss S et al (eds) Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization. Kluwer, Dordrecht, pp 433–458
56. Hansen P, Mladenović N (2001) Variable neighborhood search: Principles and applications. Eur J Oper Res 130:449–467
57. Hansen P, Mladenović N (2001) J-Means: A new local search heuristic for minimum sum-of-squares clustering. Pattern Recognit 34:405–413
58. Hansen P, Mladenović N (2001) Developments of variable neighborhood search. In: Ribeiro C, Hansen P (eds) Essays and surveys in metaheuristics. Kluwer, Boston, pp 415–440
59. Hansen P, Mladenović N (2003) Variable Neighborhood Search. In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 145–184
60. Hansen P, Mladenović N, Perez-Brito D (2001) Variable neighborhood decomposition search. J Heuristics 7(4):335–350
61. Hansen P, Mladenović N, Urosević D (2006) Variable neighborhood search and local branching. Comput Oper Res 33(10):3034–3045
62. Hansen P, Mladenović N, Urosević D (2004) Variable neighborhood search for the maximum clique. Discret Appl Math 145(1):117–125
63. Hertz A, Mittaz M (2001) A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. Transp Sci 35(4):425–434
64. Hertz A, Plumettaz M, Zufferey N (2006) Variable space search for graph coloring. Les Cahiers du GERAD G–2006–81. Res Report, HEC Montréal, Canada
65. Hindi KS, Fleszar K, Charalambous C (2003) An effective heuristic for the CLSP with setup times. J Oper Res Soc 54(5):490–498
66. Kytojoki J, Nuortio T, Braysy O, Gendreau M (2007) An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. Comput Oper Res 34(9):2743–2757
67. Lapierre SD, Ruiz AB, Soriano P (2004) Designing distribution networks: Formulations and solution heuristic. Transp Sci 38(2):174–187
68. Lejeune MA (2006) A variable neighborhood decomposition search method for supply chain management planning problems. Eur J Oper Res 175(2):959–976
69. Liang YC, Chen YC (2007) Redundancy allocation of series-parallel systems using a variable neighborhood search algorithm. Reliab Eng System Saf 92(3):323–331
70. Liu HB, Abraham A, Choi O, Moon SH (2006) Variable neighborhood particle swarm optimization for multi-

objective flexible job-shop scheduling problems. Simulated Evolution and Learning, Proc Lect Notes Comput Sci 4247:197–204
71. Lopez FG, Batista BM, Moreno Pérez JA, Moreno Vega JM (2002) The parallel variable neighborhood search for the p-median problem. J Heuristic 8(3):375–388
72. Melechovsky J, Prins C, Calvo R (2005) A metaheuristic to solve a location-routing problem with non-linear costs. J Heuristics 11(5–6):375–391
73. Melian B (2006) Using memory to improve the VNS metaheuristic for the design of SDH/WDM networks. Hybrid Metaheuristics, Proc Lect Notes Comput Sci 4030: 82–93
74. Mladenović N (1995) A variable neighborhood algorithm – a new metaheuristic for combinatorial optimization. In: Abstracts of papers presented at Optimization Days. GERAD, Montréal, p 112
75. Mladenović N (2005) Formulation space search – a new approach to optimization (plenary talk). In: Vuleta J (eds) Proceedings of XXXII SYMOPIS'05. Vrnjacka Banja, Serbia, p 3
76. Mladenović N, Brimberg J, Hansen P, Moreno Perez JA (2007) The p-median problem: A survey of metaheuristic approaches. Eur J Oper Res 179(3):927–939
77. Mladenović N, Labbé M, Hansen P (2003) Solving the p-center problem by Tabu search and Variable Neighborhood Search. Networks 42:48–64
78. Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24:1097–1100
79. Mladenović N, Moreno JP, Moreno-Vega J (1996) A Chain-interchange heuristic method. Yugoslav J Oper Res 6: 41–54
80. Mladenović N, Petrović J, Kovačević-Vujčić V, Čangalović M (2003) Solving Spread spectrum radar polyphase code design problem by Tabu search and Variable neighborhood search. Eur J Oper Res 151:389–399
81. Mladenović N, Plastria F, Urošević D (2005) Reformulation descent applied to circle packing problems. Comput Oper Res 32:2419–2434
82. Nuortio T, Kytojoki J, Niska H, Braysy O (2006) Improved route planning and scheduling of waste collection and transport. Expert Syst Appl 30(2):223–232
83. Ochi LS, Silva MB, Drummond L (2001) Metaheuristics based on GRASP and VNS for solving Traveling purchaser problem. MIC'2001. Porto University, pp 489–494
84. Plastria F, Mladenović N, Urošević D (2005) Variable neighborhood formulation space search for circle packing. In: 18th Mini Euro Conference VNS, Tenerife, Spain, pp 137–141
85. Papadimitriou C (1994) Computational Complexity. Addison Wesley, New Jersey
86. Perez JAM, Moreno-Vega JM, Martin IR (2003) Variable neighborhood tabu search and its application to the median cycle problem. Eur J Oper Res 151(2):365–378
87. Polacek M, Doerner KF, Hard RF, Kiechle G, Reimann M (2007) Scheduling periodic customer visits for a traveling salesperson. Eur J Oper Res 179(3):823–837
88. Polacek M, Hartl RF, Doemer K (2004) A variable neighborhood search for the multi depot vehicle routing problem with time windows. J Heuristics 10(6):613–627
89. Puchinger J, Raidl GR, Pferschy U (2006) The core concept for the Multidimensional Knapsack Problem. Evolutionary Computation in Combinatorial Optimization, Proc Lect Notes Comput Sci 3906:195–208
90. Qian B, Wang L, Huang DX, Wang X (2006) Multi-objective flow shop scheduling using differential evolution. Intell Computing in Signal Processing and Pattern Recognition. Lect Notes Control Inf Scis 345:1125–1136
91. Reeves CR (ed)(1992) Modern heuristic techniques for combinatorial problems. Blackwell Scientific Press, Oxford
92. Reinelt G (1991) TSLIB – A Traveling salesman library. ORSA J Comput 3:376–384
93. Repoussis PP, Paraskevopoulos DC, Tarantilis CD, Ioannou G (2006) A reactive greedy randomized variable neighborhood Tabu search for the vehicle routing problem with time windows. Hybrid Metaheuristics, Proc Lect Notes Comput Sci 4030:124–138
94. Ribeiro CC, Souza MC (2002) Variable neighborhood search for the degree-constrained minimum spanning tree problem. Discret Appl Math 118(1–2):43–54
95. Ribeiro CC, Martins SL, Rosseti I (2007) Metaheuristics for optimization problems in computer communications. Comput Commun 30(4):656–669
96. Ribeiro CC, Uchoa E, Werneck R (2002) A hybrid GRASP with perturbations for the Steiner problem in graphs. INFORMS J Comput 14(3):228–246
97. Rousseau LM, Gendreau M, Pesant G (2002) Using constraint-based operators to solve the vehicle routing problem with time windows. J Heuristics 8(1):43–58
98. Sevkli M, Aydin ME (2006) A variable neighbourhood search algorithm for job shop scheduling problems. Lect Notes Comput Sci 3906:261–271
99. Stummeer C, Sun MH (2005) New multiobjective metaheuristic solution procedures for capital investment planning. J Heuristics 11(3):183–199
100. Tagawa K, Ohtani T, Igaki T, Seki S, Inoue K (2007) Robust optimum design of SAW filters by the penalty function method. Electr Eng Japan 158(3):45–54
101. Toksari AD, Guner E (2007) Solving the unconstrained optimization problem by a variable neighborhood search. J Math Anal Appl 328(2):1178–1187
102. Urosević D, Brimberg J, Mladenović N (2004) Variable neighborhood decomposition search for the edge weighted k-cardinality tree problem. Comput Oper Res 31(8):1205–1213
103. Villa G, Lozano S, Racero J, Canca D (2006) A hybrid VNS/Tabu Search algorithm for apportioning the European Parliament. Evolutionary Computation in Com-

binatorial Optimization. Proc Lect Notes Comput Sci 3906:284–292
104. Whittaker R (1983) A fast algorithm for the greedy interchange for large-scale clustering and median location problems. INFOR 21:95–108
105. Yepes V, Medina J (2006) Economic heuristic optimization for heterogeneous fleet VRPHESTW. J Transp engineering-Asce 132(4):303–311
106. Zhang C, Lin ZG, Lin ZQ (2005) Variable neighborhood search with permutation distance for QAP. Knowledge-Based Intelligent Information and Engineering Systems, PT 4. Proc Lecture Notes Artif Intell 3684:81–88

Variational Inequalities

Anna Nagurney
University Massachusetts, Amherst, USA

MSC2000: 65K10, 65M60

Article Outline

Keywords
Problem Classes
Systems of Equations
Optimization Problems
Complementarity Problems
Fixed Point Problems
See also
References

Keywords

Projection; System of equations; Optimization problem; Complementarity problem; Fixed point problem; Nonexpansive; Variational inequality formulations; Market equilibrium conditions

Equilibrium in a fundamental concept in the study of competitive problems arising in such fields as operations research and management science, engineering, and economics, regional science, and finance. Methodologies that have been applied to the study of equilibrium problems include: systems of equations, optimization theory, complementarity theory, as well as fixed point theory. Variational inequality theory, in particular, has become a powerful technique for equilibrium analysis and computation.

Variational inequalities were introduced by P. Hartman and G. Stampacchia [3], principally, for the study of partial differential equation problems drawn from mechanics. That research focused on infinite-dimensional variational inequalities. An exposition of infinite-dimensional variational inequalities and references can be found in [4].

M.J. Smith [9] provided a formulation of the traffic network equilibrium problem which was then shown by S.C. Dafermos [2] to satisfy a finite-dimensional variational inequality problem. This connection allowed for the construction of more realistic models as well as rigorous computational techniques for equilibrium problems including: traffic network equilibrium problems, spatial price equilibrium problems, oligopolistic market equilibrium problems, as well as economic and financial equilibrium problems (cf. [5,6], and the references therein).

Many mathematical problems can be formulated as variational inequality problems and, hence, this formulation is particularly convenient since it allows for a unified treatment of equilibrium and optimization problems.

Definition 1 (*variational inequality problem*) The finite-dimensional variational inequality problem, VI(F, K), is to determine a vector $x^* \in K \subset \mathbf{R}^n$, such that

$$\langle F(x^*)^\top, x - x^* \rangle \geq 0, \quad \forall x \in K,$$

where F is a given continuous function from K to $\mathbf{R}^n$, K is a given closed convex set, and $\langle \cdot, \cdot \rangle$ denotes the inner product in $\mathbf{R}^n$.

We now discuss some basic problem types and their relationships to the variational inequality problem. We also provide examples. Proofs of the theoretical results may be found in [4,5]. For algorithms for the computation of variational inequalities, see also [1,5,7,8].

We begin with systems of equations, which have been used to formulate certain equilibrium problems. We then discuss optimization problems, both unconstrained and constrained, as well as complementarity problems. We conclude with a fixed point problem and its relationship with the variational inequality problem.

Problem Classes

We here briefly review certain problem classes, which appear frequently in equilibrium modeling, and identify their relationships to the variational inequality problem.

Systems of Equations

Systems of equations are common in equilibrium analysis, expressing, for example, that the demand is equal to the supply of various commodities at the equilibrium price levels. Let $K = \mathbf{R}^n$ and let F: $\mathbf{R}^n \to \mathbf{R}^n$ be a given function. A vector $x^* \in \mathbf{R}^n$ is said to solve a *system of equations* if

$$F(x^*) = 0.$$

The relationship to a variational inequality problem is stated in the following

Proposition 2 *Let $K = \boldsymbol{R}^n$ and let $F : \boldsymbol{R}^n \to \boldsymbol{R}^n$ be a given vector function. Then $x^* \in \boldsymbol{R}^n$ solves the variational inequality problem VI(F, K) if and only if x^* solves the system of equations*

$$F(x^*) = 0.$$

Example 3 (Market equilibrium with equalities only) As an illustration, we now present an example of a system of equations. Consider m consumers, with a typical consumer denoted by j, and n commodities, with a typical commodity denoted by i. We let p denote the n-dimensional column vector of the commodity prices with components: $\{ p_1, \ldots, p_n \}$.

Assume that the demand for a commodity i, d_i, may, in general, depend upon the prices of all the commodities, that is,

$$d_i(p) = \sum_{j=1}^{m} d_i^j(p),$$

where $d_i^j(p)$ denotes the demand for commodity i by consumer j at the price vector p.

Similarly, the supply of a commodity i, s_i, may, in general, depend upon the prices of all the commodities, that is,

$$s_i(p) = \sum_{j=1}^{m} s_i^j(p),$$

where $s_i^j(p)$ denotes the supply of commodity i of consumer j at the price vector p.

We group the aggregate demands for the commodities into the n-dimensional column vector d with components: $\{ d_1, \ldots, d_n \}$ and the aggregate supplies of the commodities into the n-dimensional column vector s with components: $\{ s_1, \ldots, s_n \}$.

The market equilibrium conditions that require that the supply of each commodity must be equal to the demand for each commodity at the equilibrium price vector p^*, are equivalent to the following system of equations:

$$s(p^*) - d(p^*) = 0.$$

Clearly, this expression into the standard nonlinear equation form, if we define the vectors $x \equiv p$ and $F(x) \equiv s(p) - d(p)$.

Note, however, that the problem class of nonlinear equations is not sufficiently general to guarantee, for example, that $x^* \geq 0$, which may be desirable in this example in which the vector x refers to prices.

Optimization Problems

Optimization problems, on the other hand, consider explicitly an objective function to be minimized (or maximized), subject to constraints that may consist of both equalities and inequalities. Let f be a continuously differentiable function where f: $\mathbf{K} \to :\mathbf{R}$. Mathematically, the statement of an *optimization problem* is:

$$\begin{cases} \min & f(x) \\ \text{s.t.} & x \in K. \end{cases}$$

The relationship between an optimization problem and a variational inequality problem is now highlighted.

Proposition 4 *Let x^* be a solution to the optimization problem:*

$$\begin{cases} \min f(x) \\ s.t. \qquad x \in K, \end{cases}$$

where f is continuously differentiable and K is closed and convex. Then x^* is a solution of the variational inequality problem:

$$\left\langle \nabla f(x^*)^\top, x - x^* \right\rangle \geq 0, \quad \forall x \in K.$$

Furthermore, we have the following:

Proposition 5 *If $f(x)$ is a convex function and x^* is a solution to VI(∇f, K), then x^* is a solution to the above optimization problem.*

If the feasible set $K = \mathbf{R}^n$, then the unconstrained optimization problem is also a variational inequality problem.

On the other hand, in the case where a certain symmetry condition holds, the variational inequality problem can be reformulated as an optimization problem. In other words, in the case that the variational inequality formulation of the equilibrium conditions underlying a specific problem is characterized by a function with a symmetric Jacobian, then the solution of the equilibrium conditions and the solution of a particular optimization problem are one and the same. We first introduce the following definition and then fix this relationship in a theorem.

Definition 6 An $n \times n$ matrix $M(x)$, whose elements $m_{ij}(x)$; $i = 1, \ldots, n$; $j = 1, \ldots, n$, are functions defined on the set $S \subset \mathbf{R}^n$, is said to be *positive semidefinite* on S if

$$v^\top M(x)v \geq 0, \qquad \forall v \in \mathbf{R}^n, \quad x \in S.$$

It is said to be *positive definite* on S if

$$v^\top M(x)v > 0, \qquad \forall v \neq 0, \quad v \in \mathbf{R}^n, \quad x \in S.$$

It is said to be *strongly positive definite* on S if

$$v^\top M(x)v \geq \alpha \|v\|^2,$$

for some $\alpha > 0$, $\forall v \in \mathbf{R}^n$, $x \in S$.

Note that if $\gamma(x)$ is the smallest eigenvalue, which is necessarily real, of the symmetric part of $M(x)$, that is, $[M(x) + M(x)^\top]/2$, then it follows that:

i) $M(x)$ is positive semidefinite on S if and only if $\gamma(x) \geq 0$, for all $x \in S$;
ii) $M(x)$ is positive definite on S if and only if $\gamma(x) > 0$, for all $x \in S$;
iii) $M(x)$ is strongly positive definite on S if and only if $\gamma(x) \geq \alpha > 0$, for all $x \in S$.

Theorem 7 *Assume that $F(x)$ is continuously differentiable on K and that the Jacobian matrix*

$$\nabla F(x) = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial F_n}{\partial x_1} & \cdots & \frac{\partial F_n}{\partial x_n} \end{pmatrix}$$

is symmetric and positive semidefinite. Then there is a real-valued convex function $f: K \to \boldsymbol{R}^1$ satisfying

$$\nabla f(x) = F(x)$$

with x^* the solution of VI(F, K) also being the solution of the mathematical programming problem:

$$\begin{cases} \min & f(x) \\ s.t. & x \in K. \end{cases}$$

Hence, although the variational inequality problem encompasses the optimization problem, a variational inequality problem can be reformulated as a convex optimization problem, only when the symmetry condition and the positive semidefiniteness condition hold.

Therefore, the variational inequality is the more general problem in that it can also handle a function $F(x)$ with an asymmetric Jacobian. Historically, many equilibrium problems were reformulated as optimization problems, under precisely such a symmetry assumption. The assumption, however, in terms of applications was restrictive and precluded the more realistic modeling of multiple commodities, multiple modes and/or classes in competition. Moreover, the objective function that resulted was sometimes artificial, without a clear economic interpretation, and simply a mathematical device.

Complementarity Problems

The variational inequality problem also contains the complementarity problem as a special case. Complementarity problems are defined on the nonnegative orthant.

Let $\mathbf{R}^n_+$ denote the nonnegative orthant in $\mathbf{R}^n$, and let F: $\mathbf{R}^n \to \mathbf{R}^n$. The *nonlinear complementarity problem* over $\mathbf{R}^n_+$ is a system of equations and inequalities stated as:

$$\begin{cases} \text{Find} & x^* \geq 0 \\ \text{s.t.} & F(x^*) \geq 0 \quad \text{and} \quad \langle F(x^*)^\top, x^* \rangle = 0. \end{cases}$$

Whenever the mapping F is affine, that is, whenever $F(x) = Mx + b$, where M is an $n \times n$ matrix and b an $n \times 1$ vector, the problem is then known as the *linear complementarity* problem.

The relationship between the complementarity problem and the variational inequality problem is as follows.

Proposition 8 *VI (F, $\mathbf{R}^n_+$) and the complementarity problem have precisely the same solutions, if any.*

Example 9 (Market equilibrium with equalities and inequalities) We now present a nonlinear complementarity formulation of market equilibrium. We assume that the prices must now be nonnegative in equilibrium. Hence, we consider the following situation, in which the demand functions are given as previously as are the supply functions, but now, instead of the market equilibrium conditions, which are represented of a system of equations, we have the following equilibrium conditions: For each commodity i; $i = 1, \ldots, n$:

$$s_i(p^*) - d_i(p^*) \begin{cases} = 0 & \text{if } p_i^* > 0, \\ \geq 0 & \text{if } p_i^* = 0. \end{cases}$$

Note that these equilibrium conditions state that if the price of a commodity is positive in equilibrium then the supply of that commodity must be equal to the demand for that commodity. On the other hand, if the price of a commodity at equilibrium is zero, then there may be an excess supply of that commodity at equilibrium, that is, $s_i(p^*) - d_i(p^*) > 0$, or the market clears. Furthermore, this system of equalities and inequalities guarantees that the prices of the instruments do not take on negative values, which may occur in the system of equations expressing the market clearing conditions.

We now give the nonlinear complementarity formulation of this problem:

$$\begin{cases} \text{Determine} & p^* \in \mathbf{R}^n_+ \\ \text{satisfying} & s(p^*) - d(p^*) \geq 0 \\ & \langle (s(p^*) - d(p^*))^\top, p^* \rangle = 0. \end{cases}$$

Moreover, since a nonlinear complementarity problem is a special case of a variational inequality problem, we may rewrite the nonlinear complementarity formulation of the market equilibrium problem above as a variational inequality problem:

$$\begin{cases} \text{Determine} & p^* \in \mathbf{R}^n_+, \\ \text{s.t.} & \langle (s(p^*) - d(p^*))^\top, p - p^* \rangle \geq 0, \\ & \quad \forall p \in \mathbf{R}^n_+. \end{cases}$$

Note, first, that in the special case of demand functions and supply functions which are separable, the Jacobians of these functions are symmetric since they are diagonal and given, respectively, by

$$\nabla s(p) = \begin{pmatrix} \frac{\partial s_1}{\partial p_1} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial s_n}{\partial p_n} \end{pmatrix},$$

$$\nabla d(p) = \begin{pmatrix} \frac{\partial d_1}{\partial p_1} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial d_n}{\partial p_n} \end{pmatrix}.$$

Indeed, in this special case model, the supply of a commodity depends only upon the price of that commodity and, similarly, the demand for a commodity depends only upon the price of that commodity.

Hence, in this special case, the price vector p^* that satisfies the equilibrium conditions can be obtained by solving the following optimization problem:

$$\begin{cases} \min & \sum_{i=1}^{n} \int_0^{p_i} s_i(x)\, dx - \sum_{i=1}^{n} \int_0^{p_i} d_i(y)\, dy \\ \text{s.t.} & p_i \geq 0, \qquad i = 1, \ldots, n. \end{cases}$$

Note that one also obtains an optimization reformulation of the equilibrium conditions, provided that the following symmetry condition holds: $\partial s_i / \partial p_k = \partial s_k / \partial p_i$ and $\partial d_i / \partial p_k = \partial d_k / \partial p_i$ for all commodities i, k. In other words, the price of a commodity k affects the supply of a commodity i in the same way that the price of a commodity i affects the price of a commodity k. A similar situation must hold for the demands for the commodities.

However, such conditions are limiting from the application standpoint and, hence, the appeal of variational inequality problem that enables the formulation and, ultimately, the computation of equilibria where such restrictive symmetry assumptions on the underlying functions need no longer hold. Indeed, such symmetry assumptions were not imposed in the variational inequality problem.

Example 10 (Market equilibrium with equalities and inequalities and policy interventions) We now provide a generalization of the preceding market equilibrium model to allow for price policy interventions in the

form of price floors and ceilings. In particular, we let p_i^C denote the imposed price ceiling on the price of commodity i, and we let p_i^F denote the imposed price floor on the price of commodity i. Then we have the following equilibrium conditions: For each commodity i; $i = 1, \ldots, n$:

$$s_i(p^*) - d_i(p^*) \begin{cases} \leq 0 & \text{if} \quad p_i^* = p_i^C \\ = 0 & \text{if} \quad p_i^F < p_i^* < p_i^C \\ \geq 0 & \text{if} \quad p_i^* = p_i^F. \end{cases}$$

Note that these equilibrium conditions state that if the price of a commodity in equilibrium lies between the imposed price floor and ceiling, then the supply of that commodity must be equal to the demand for that commodity. On the other hand, if the price of a commodity at equilibrium is at the imposed floor, then there may be an excess supply of that commodity at equilibrium, that is, $s_i(p^*) - d_i(p^*) > 0$, or the market clears. In contrast, if the price of a commodity in equilibrium is at the imposed ceiling, then there may be an excess demand of the commodity in equilibrium.

We now provide a variational inequality formulation of the governing equilibrium conditions:

Determine $p^* \in K$, such that

$$\langle (s(p^*) - d(p^*))^\top, p - p^* \rangle \geq 0, \quad \forall p \in K,$$

where the feasible set $K \equiv \{ p \mid p^F \leq p \leq p^C \}$, where p^F and p^C denote, respectively, the n-dimensional column vectors of imposed price floors and ceilings.

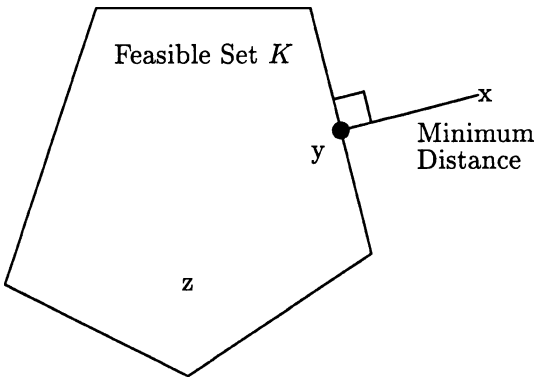
Fixed Point Problems

We now turn to a discussion of fixed point problems in conjunction with variational inequality problems. We also provide the geometric interpretation of the variational inequality problem and its relationship to a fixed point problem.

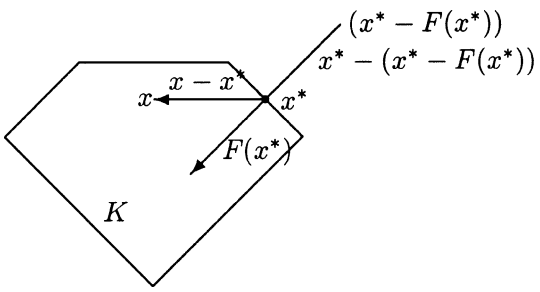
We first define a projection. For a graphical depiction, see Fig. 1 .

Definition 11 (a *projection*) Let K be a closed convex set in $\mathbf{R}^n$. Then for each $x \in \mathbf{R}^n$, there is a unique point $y \in K$, such that

$$\|x - y\| \leq \|x - z\|, \quad \forall z \in K,$$



Variational Inequalities, Figure 1
The projection *y* of *x* on the set *K*



Variational Inequalities, Figure 2
Geometric depiction of the variational inequality problem and its fixed point equivalence (with $\gamma = 1$)

and y is known as the *orthogonal projection* of x on the set K with respect to the Euclidean norm, that is,

$$y = P_K(x) = \arg\min_{z \in K} \|x - z\|.$$

In other words, the closest point to x lying in the set K is given by y.

We now present a property of the projection operator that is useful both in the qualitative analysis of equilibria and in their computation. Let K again be a closed convex set. Then the projection operator P_K is *nonexpansive*, that is,

$$\|P_K x - P_K x'\| \leq \|x - x'\|, \quad \forall x, x' \in \mathbf{R}^n.$$

The relationship between a variational inequality and a fixed point problem can now be stated (see Fig. 2).

Theorem 12 *Assume that K is closed and convex. Then $x^* \in K$ is a solution of the variational inequality problem VI(F, K) if and only if x^* is a fixed point of the map: $P_K(I$*

$-\gamma F): K \to K$, *for* $\gamma > 0$, *that is,*

$$x^* = P_K(x^* - \gamma F(x^*)).$$

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Bertsekas DP, Tsitsiklis JN (1989) Parallel and distributed computation: Numerical methods. Prentice-Hall, Englewood Cliffs, NJ
2. Dafermos S (1980) Traffic equilibria and variational inequalities. Transport Sci 14:42–54
3. Hartman P, Stampacchia G (1966) On some nonlinear elliptic differential functional equations. Acta Math 115:271–310
4. Kinderlehrer D, Stampacchia G (1980) An introduction to variational inequalities and their applications. Acad. Press, New York
5. Nagurney A (1999) Network economics: A variational inequality approach, 2nd edn. Kluwer, Dordrecht
6. Nagurney A, Siokos S (1997) Financial networks: Statics and dynamics. Springer, Berlin
7. Nagurney A, Zhang D (1997) Projected dynamical systems and variational inequalities with applications. Kluwer, Dordrecht
8. Patriksson M (1994) The traffic assignment problem. VSP, Utrecht
9. Smith MJ (1979) Existence, uniqueness, and stability of traffic equilibria. Transport Res 13B:295–304

Variational Inequalities: F. E. Approach

JAROSLAV HASLINGER
Charles University, Prague, CZ

MSC2000: 65M60

Article Outline

Keywords
See also
References

Keywords

Approximation of variational inequalities; Mixed finite element approximation; Ritz–Galerkin method

Let V be a Hilbert space with the norm $\|\cdot\|$ and V' its dual space with the duality pairing denoted by $\langle \cdot, \cdot \rangle$. Let

a: $V \times V \to \mathbf{R}^1$ be a *bilinear form* satisfying:

$$(\mathcal{H}) \begin{cases} \text{boundedness:} \quad \exists M = \text{const} > 0 : \\ \quad |a(u,v)| \le M \|u\| \, \|v\|, \quad \forall u, v \in V; \\ V\text{-ellipticity:} \quad \exists \alpha = \text{const} > 0 : \\ \quad a(v,v) \ge \alpha \|v\|^2, \quad \forall v \in V. \end{cases}$$

Finally, let K be a nonempty, closed and convex subset of V and $f \in V'$ be given.

By an *abstract variational inequality of elliptic type* we mean a triple $\{ K, a, f \}$ and the element $u \in K$ satisfying

$$(\mathcal{P}) a(u, v-u) \ge \langle f, v-u \rangle, \quad \forall v \in K,$$

its solution. It is known (see [7]) that if $\underline{a}$ satisfies $(\mathcal{H})$, then $(\mathcal{P})$ has a unique solution for any $f \in V'$. If moreover $\underline{a}$ is *symmetric* in V, i. e. $a(u, v) = a(v, u)$ for any $u, v \in V$, then $(\mathcal{P})$ is equivalent to the following minimization problem:

$$(\mathcal{P}') \text{ Find } u \in K : J(u) \le J(v), \quad \forall v \in K,$$

where $J(v) = 1/2 a(v, v) - \langle f, v \rangle$ is the quadratic functional.

In order to define the approximation of $(\mathcal{P})$, we introduce a family $\{ V_h \}$ of *finite-dimensional subspaces* $V_h \subset V$, dim $V_h = n(h)$, where $h > 0$ is a discretization parameter and $n(h) \to +\infty$ when $h \to 0+$. Let $K_h \subset V_h$ be a nonempty, closed and convex set, not necessarily a subset of K.

By the *approximation* of $(\mathcal{P})$ we call the problem

$(\mathcal{P}_h)$ Find $u_h \in K_h$:

$$a(u_h, v_h - u_h) \ge \langle f, v_h - u_h \rangle, \quad \forall v_h \in K_h,$$

or, in the case when $\underline{a}$ is symmetric:

$$(\mathcal{P}_h') \text{ Find } u_h \in K_h : J(u_h) \le J(v_h), \quad \forall v_h \in K_h.$$

Such approach is known as the *Ritz–Galerkin method* for the approximation of $(\mathcal{P})$.

Let $V_h = \{ \phi_1, \dots, \phi_{n(h)} \}$ be a basis of V_h and denote by $\mathcal{I}$ the isomorphism between V_h and $\mathbf{R}^{n(h)}$: $\mathcal{I}(V_h) = \mathbf{R}^{n(h)}$ defined in the standard way. Then K_h can be identified with a nonempty, closed and convex subset $\mathcal{K} \subset \mathbf{R}^{n(h)}$, where $\mathcal{K} = \mathcal{I}(K_h)$ and problem $(\mathcal{P}_h)$ can be written in the following algebraic form:

$$(\vec{\mathcal{P}}) \text{ Find } \vec{x} \in \mathcal{K} : (\mathbf{A}\vec{x}, \vec{y} - \vec{x}) \ge (\vec{F}, \vec{y} - \vec{x}), \forall \vec{y} \in \mathcal{K},$$

where $\mathbf{A} = (a_{ij})_{i,j=1}^{n(h)}$ is the matrix with the elements $a_{ij} = a(\phi_j, \phi_i)$, $\vec{F} = (F_i)_{i=1}^{n(h)} \in \mathbf{R}^{n(h)}$ with $F_i = \langle f, \phi_i \rangle$, $i = 1, \dots, n(h)$, and $(\cdot, \cdot)$ stands for the scalar product in $\mathbf{R}^{n(h)}$. In addition, if $\underline{a}$ is symmetric in V, then $(\mathcal{P}_h')$ is equivalent to the *constrained minimization problem*:

$$(\vec{\mathcal{P}}') \text{ Find } \vec{x} \in \mathcal{K} : \mathcal{J}(\vec{x}) \le \mathcal{J}(\vec{y}), \quad \forall \vec{y} \in \mathcal{K},$$

where

$$\mathcal{J}(\vec{y}) = \frac{1}{2}(\mathbf{A}\vec{y}, \vec{y}) - (\vec{F}, \vec{y}).$$

A natural question arises, namely how to estimate the error between u and u_h. It holds:

Theorem 1 *Let u and u_h be the solution to $(\mathcal{P})$ and $(\mathcal{P}_h)$, respectively, and let $(\mathcal{H})$ be satisfied. Then*

$$\begin{aligned} &\alpha \|u - u_h\|^2 \le a(u - u_h, u - u_h) \\ &\le \langle f, u - v_h \rangle + \langle f, u_h - v \rangle \\ &+ a(u_h - u, v_h - u) + a(u, v - u_h) \\ &+ a(u, v_h - u), \\ &\forall v \in K, \quad \forall v_h \in K_h. \end{aligned} \tag{1}$$

For the proof, see [3,5].

Remark 2 If $K_h \subset K$ for any $h > 0$, then choosing $v = u_h$ in (1) we obtain:

$$\begin{aligned} &\alpha \|u - u_h\|^2 \le a(u - u_h, u - u_h) \\ &\le \langle f, u - v_h \rangle + a(u_h - u, v_h - u) + a(u, v_h - u), \\ &\qquad \forall v_h \in K_h. \end{aligned} \tag{2}$$

In order to guarantee that $\| u_h - u \| \to 0$, $h \to 0+$, the following properties of the system $\{K_h\}$ are needed:

$$\forall v \in K \; \exists \{v_h\}, \; v_h \in K_h : \; v_h \to v, \; h \to 0+, \tag{3}$$

$$\begin{aligned} &\text{if } \{v_h\}, \; v_h \in K_h, \text{ is such that} \\ &\qquad v_h \rightharpoonup v \text{ (weakly) in } V, \text{ then } v \in K. \end{aligned} \tag{4}$$

Then one has:

Theorem 3 *Let $(\mathcal{H})$ and (3), (4) be satisfied. Then the Ritz–Galerkin method is convergent, i. e.*

$$\|u - u_h\| \to 0, \quad h \to 0+.$$

The proof easily follows from (1) and (2).

Remark 4 If $K_h \subset K$ for any $h > 0$, then the condition (4) is automatically satisfied.

In practice, the sets V_h and K_h are constructed by using *finite element methods*. To illustrate such a construction we consider the following model example.

Example 5 Let $\{ K, a, f \}$ be the variational inequality with

$$K = \{v \in H_0^1(\Omega):\ v \geq \phi \text{ a.e. in } \Omega\},$$
$$a(u,v) = \int_\Omega \operatorname{grad} u \cdot \operatorname{grad} v \, dx,$$
$$\langle f, v\rangle = \int_\Omega f v \, dx,$$

where $\phi \in C(\overline{\Omega})$ is a given function, $\phi \leq 0$ on $\partial\Omega$, $f \in L^2(\Omega)$ and $H_0^1(\Omega)$ is the standard Sobolev space of functions vanishing on the boundary $\partial\Omega$.

Since $\underline{a}$ is bounded and elliptic in $H_0^1(\Omega)$ and K is a nonempty, closed convex subset of $H_0^1(\Omega)$, $\{K, a, f\}$ has a unique solution $u \in K$:

$$\int_\Omega \operatorname{grad} u \cdot \operatorname{grad}(v-u)\, dx \geq \int_\Omega f(v-u)\, dx, \quad \forall v \in K. \tag{5}$$

Let us suppose that Ω is a plane polygonal domain. Let $\{\mathcal{T}_h\}$ be a *regular family of triangulations* of $\overline{\Omega}$ (see [2]). With any $\mathcal{T}_h$ we associate the space of *piecewise linear functions* $V_h \subset H_0^1(\Omega)$:

$$V_h = \left\{ v_h \in C(\overline{\Omega}):\ \begin{array}{l} v_h|_T \in P_1(T),\ \forall T \in \mathcal{T}_h, \\ v_h = 0 \text{ on } \partial\Omega \end{array} \right\}$$

and its closed convex subset K_h:

$$K_h = \{v_h \in V_h:\ v_h(A_i) \geq \phi(A_i),\ \forall A_i \in \mathcal{N}_h\},$$

where $\mathcal{N}_h$ is the set of all interior nodes of $\mathcal{T}_h$. Note that $K_h \not\subset K$, in general.

The approximation of (5) is defined by

$$\text{Find } u_h \in K_h:$$
$$\int_\Omega \operatorname{grad} u_h \cdot \operatorname{grad}(v_h - u_h)\, dx \geq \int_\Omega f(v_h - u_h)\, dx, \quad \forall v_h \in K_h. \tag{6}$$

Since $\underline{a}$ is symmetric, u_h can be equivalently characterized by

$$u_h \in K_h: \quad J(u_h) \leq J(v_h), \quad \forall v_h \in K_h, \tag{7}$$

where

$$J(v_h) = \frac{1}{2}\int_\Omega |\operatorname{grad} v_h|^2 \, dx - \int_\Omega f v_h \, dx.$$

The algebraic form of (7) reads as follows:

$$\text{Find } \vec{x} \in \mathcal{K}: \quad \mathcal{J}(\vec{x}) \leq \mathcal{J}(\vec{y}), \quad \forall \vec{y} \in \mathcal{K}, \tag{8}$$

where

$$\mathcal{K} = \{\vec{y} = (y_1, \ldots, y_{n(h)}) \in \mathbf{R}^{n(h)}:\ y_i \geq \phi(A_i), i = 1, \ldots, n(h)\},$$
$$n(h) = \operatorname{card} \mathcal{N}_h,$$
$$\mathcal{J}(\vec{y}) = \frac{1}{2}(\mathbf{A}\vec{y}, \vec{y}) - (\vec{F}, \vec{y}),$$

with

$$\mathbf{A} = (a_{ij})_{i,j=1}^{n(h)},$$
$$a_{ij} = \int_\Omega \operatorname{grad}\phi_i \cdot \operatorname{grad}\phi_j \, dx,$$
$$\vec{F} = (F_i)_{i=1}^{n(h)}, \quad F_i = \int_\Omega f\phi_i \, dx$$

and $\{\phi_i\}_{i=1}^{n(h)}$ being the basis of V_h. Using Theorems 1 and 3 one can prove the following convergence result:

Theorem 6 *It holds:*
i) if $\phi \in H^2(\Omega)$ and $u \in H^2(\Omega) \cap K$ then

$$\|u - u_h\|_{H^1(\Omega)} \leq ch,$$

where $c > 0$ does not depend on h;
ii) if $\phi \in C(\overline{\Omega})$, then

$$\|u - u_h\|_{H^1(\Omega)} \to 0, \quad h \to 0+,$$

without any regularity assumption on u.

To release the constraint $v \in K$, the duality approach may be used. Such a formulation involving besides the primal variable also Lagrange multipliers is the basis for the so-called *mixed finite element* methods.

Let Y be another Hilbert space and $\Lambda \subset Y$ be a closed, convex cone containing the zero element of

$Y, g \in Y'$ be an element of the dual space to Y. The duality pairing between Y' and Y is denoted by $[\,\cdot,\cdot]$. Let us suppose that the convex set K is characterized by:

$$K = \{v \in V:\ b(v,\mu) \geq [g,\mu], \quad \forall \mu \in \Lambda\},$$

where b: $V \times Y \to \mathbf{R}^1$ is a continuous bilinear form.

We shall define the new problem by:

($\mathcal{M}$) Find $(u,\lambda) \in V \times \Lambda$ such that

$$a(u,v) + b(v,\lambda) = \langle f, v\rangle, \quad \forall v \in V,$$
$$b(u,\mu-\lambda) \geq [g,\mu-\lambda], \quad \forall \mu \in \Lambda,$$

where a: $V \times V \to \mathbf{R}^1$ is the bilinear form satisfying ($\mathcal{H}$).

Problem ($\mathcal{M}$) will be called the *mixed variational formulation* to ($\mathcal{P}$). In order to guarantee the existence and the uniqueness of its solution we suppose that (see [1]):

$$\exists \beta > 0: \quad \sup_{\substack{v \in V,\\ v \neq 0}} \frac{b(v,\mu)}{\|v\|} \geq \beta \|\mu\|_Y, \quad \forall \mu \in Y. \tag{9}$$

Remark 7 If a is symmetric on V, then ($\mathcal{M}$) is equivalent to the following *saddle-point formulation* (see [5]):

($\mathcal{M}'$) Find $(u,\lambda) \in V \times \Lambda$ such that

$$\mathcal{L}(u,\mu) \leq \mathcal{L}(u,\lambda) \leq \mathcal{L}(v,\lambda),$$
$$\forall (v,\mu) \in V \times \Lambda,$$

where $\mathcal{L}(v,\mu) \equiv J(v) - b(v,\mu) + [g,\mu]$.

Let $\{V_h\}$, $\{Y_H\}$ be two families of finite-dimensional subspaces of V and Y, respectively. Let $\Lambda_H \subset Y_H$ be a closed, convex cone, containing the zero element of Y.

By the *approximation* of ($\mathcal{M}$) we call the problem

($\mathcal{M}_h^H$) Find $(u_h,\lambda_H) \in V_h \times \Lambda_H$ such that:

$$a(u_h,v_h) + b(v_h,\lambda_H) = \langle f, v_h\rangle,$$
$$\forall v_h \in V_h,$$
$$b(u_h,\mu_H - \lambda_H) \geq [g,\mu_H - \lambda_H],$$
$$\forall \mu_H \in \Lambda_H.$$

One can formulate conditions under which the sequence $\{(u_h,\lambda_H)\}$ of solutions to ($\mathcal{M}_h^H$) tends to the solution (u,λ) of ($\mathcal{M}$) (see [3,4]). Such a mixed formulation is useful since:

j) there are no constraints imposed on the primal variable u;

jj) it makes possible to approximate not only the primal but also the dual variable λ.

Example 8 Let us consider the variational inequality $\{K, a, f\}$, where:

$$K = \{v \in H^1(\Omega):\ v \geq 0 \text{ on } \partial\Omega\},$$
$$a(u,v) = \int_\Omega (\operatorname{grad} u \cdot \operatorname{grad} v + uv)\,dx,$$
$$\langle f, v\rangle = \int_\Omega fv\,dx, \quad f \in L^2(\Omega).$$

Then the convex set K can be equivalently characterized as follows:

$$K = \{v \in H^1(\Omega) : [v,\mu] \geq 0,$$
$$\forall \mu \in H^{-1/2}(\partial\Omega),\ \mu \geq 0\},$$

where $H^{-1/2}(\partial\Omega)$ is the dual space to

$$H^{1/2}(\partial\Omega) = \{\phi:\ \partial\Omega \to \mathbf{R}^1 : \exists v \in H^1(\Omega) : v = \phi \text{ on } \partial\Omega\}.$$

The symbol $[\,\cdot,\cdot\,]$ stands for the corresponding duality and the ordering '$\geq$' is defined in a usual way: $\mu \geq 0$ if and only if $[v,\mu] \geq 0$ for any $v \in K$. Denote by Λ the convex cone of all nonnegative functionals over $H^{1/2}(\partial\Omega)$. The mixed formulation of $\{K, a, f\}$ is given by:

$$\begin{cases} \text{Find} & (u,\lambda) \in H^1(\Omega) \times \Lambda \\ \text{s.t.} & a(u,v) + [v,\lambda] = \langle f, v\rangle, \\ & \quad \forall v \in H^1(\Omega), \\ & [u,\mu-\lambda] \geq 0, \quad \forall \mu \in \Lambda. \end{cases} \tag{10}$$

The approximation of (10) will be defined by a finite element method.

To this end we suppose that $\Omega \subset \mathbf{R}^2$ is a polygonal domain. Let $\{\mathcal{T}_h\}$, $h \to 0+$, be a regular family of triangulations of $\overline{\Omega}$ and let

$$V_h = \{v_h \in C(\overline{\Omega}):\ v_h|_T \in P_1(T),\ \forall T \in \mathcal{T}_h\}$$

be the space of piecewise linear functions over $\mathcal{T}_h$. Further, let $\{\mathcal{T}_H\}$ be a regular family of partitions of $\partial\Omega$ into segments I, the length of which does not exceed the number $H > 0$. We define

$$\Lambda_H = \{\mu_H \in L^2(\partial\Omega) : \mu_H|_I \in P_0(I),$$
$$\forall I \in \mathcal{T}_H,\ \mu_H \geq 0 \text{ on } \partial\Omega\},$$

i. e. Λ_H is the set of all nonnegative piecewise constant functions over $\mathcal{T}_H$. The approximation of (10) is defined as follows:

$$\begin{cases} \text{Find} & (u_h, \lambda_H) \in V_h \times \Lambda_H \\ \text{s.t.} & a(u_h, v_h) = (v_h, \lambda_H)_{0,\partial\Omega} = \langle f, v_h \rangle\,, \\ & \quad \forall v_h \in V_h, \\ & (u_h, \mu_H - \lambda_H)_{0,\partial\Omega} \geq 0, \\ & \quad \forall \mu_H \in \Lambda_H, \end{cases} \tag{11}$$

where $(u,\, \mu)_{0,\,\partial\Omega} \equiv \int_{\partial\Omega} u\ \mu ds$. The relation between (10) and (11) is studied in [4].

Since $\underline{a}$ is symmetric, problem (11) is equivalent to the saddle-point formulation:

$$\begin{cases} \text{Find} & (u_h, \lambda_H) \in V_h \times \Lambda_H \\ \text{s.t.} & \mathcal{L}(u_h, \mu_H) \leq \mathcal{L}(u_h, \lambda_H) \leq \mathcal{L}(v_h, \lambda_H) \\ & \quad \forall (v_h, \mu_H) \in V_h \times \Lambda_H, \end{cases} \tag{12}$$

where

$$\mathcal{L}(v_h, \mu_H) = \frac{1}{2} \int_{\Omega} \left(|\text{grad}\, v_h|^2 + v_h^2 \right)\, \mathrm{d}x - \int_{\partial\Omega} v_h \mu_H\, \mathrm{d}s - \int_{\Omega} f v_h\, \mathrm{d}x.$$

The approximation of elliptic variational inequalities describing problems in mechanics of solids (contact problems, problems involving friction, different models of plasticity) can be found in [4,5,6]. The approximation of time dependent variational inequalities is studied in [3].

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Brezzi F, Hager WW, Raviart PA (1977) Error estimates for the finite element solution of variational inequalities. Numer Math 28:431–443
2. Ciarlet PG (1978) The finite element method for elliptic problems. Stud Math Appl, vol 4. North-Holland, Amsterdam
3. Glowinski R, Lions JL, TrÈmoliÉres R (1981) Numerical analysis of variational inequalities. Stud Math Appl. North-Holland, Amsterdam
4. Haslinger J, HlavÀcek I, Necas J (1996) Numerical methods for unilateral problems in solid mechanics. In: Handbook Numerical Analysis, Part 2 IV, Elsevier, Amsterdam
5. HlavÀcek I, Haslinger J, Necas J, LovÌsek J (1988) Numerical solution of variational inequalities. Appl Math Sci, vol 66. Springer, Berlin

6. Kikuchi N, Oden JT (1988) Contact problems in elasticity: A study of variational inequalities and finite element methods. Stud Appl Math. SIAM, Philadelphia
7. Lions JL (1969) Quelques mèthodes de rèsolution des problèmes aux limites non linèaires. Dunod, Paris

Variational Inequalities: Geometric Interpretation, Existence and Uniqueness

ANNA NAGURNEY
University Massachusetts, Amherst, USA

MSC2000: 65K10, 65M60

Article Outline

Keywords
See also
References

Keywords

Geometric interpretation; Coercivity condition; Monotonicity; Strict monotonicity; Strong monotonicity; Lipschitz continuity

Variational inequality theory is a powerful tool in the qualitative analysis of equilibria. Here we provide a geometric interpretation of the variational inequality problem and conditions for existence and uniqueness of solutions. For proofs of the theoretical results stated herein, see [1,2]. For stability and sensitivity analysis of variational inequalities, including applications, see [2], and the references therein.

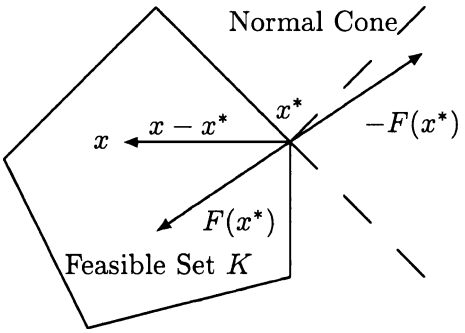
In particular, here we consider the *finite-dimensional variational inequality problem* VI(F, K): Determine $x^* \in K$, such that

$$\langle F(x^*)^\top, x - x^* \rangle \geq 0, \quad \forall x \in K,$$

where $K \subset \mathbf{R}^n$ is a closed convex set and F is the vector function: F: $K \to \mathbf{R}^n$, where $\langle \cdot, \cdot \rangle$ denotes the inner product in $\mathbf{R}^n$.

From the definition one can deduce that the necessary and sufficient condition for x^* to be a solution to VI(F, K) is that

$$-F(x^*) \in C(x^*),$$



Variational Inequalities: Geometric Interpretation, Existence and Uniqueness, Figure 1
Geometric interpretation of VI(F, K)

where $C(x)$ denotes the *normal cone* of K at x, defined by

$$C(x) \equiv \{y \in \mathbf{R}^n : \langle y^\top, x' - x \rangle \leq 0, \quad \forall x' \in K\}.$$

A geometric depiction of the variational inequality problem is given in Fig. 1.

Existence of a solution to a variational inequality problem follows from continuity of the function F entering the variational inequality, provided that the feasible set K is compact. Indeed, we have the following:

Theorem 1 *If K is a compact convex set and $F(x)$ is continuous on K, then the variational inequality problem admits at least one solution x^*.*

In the case of an unbounded feasible set K, this theorem is no longer applicable; the existence of a solution to a variational inequality problem can, nevertheless, be established under the subsequent condition.

Let $B_R(0)$ denote a closed ball with radius R centered at 0 and let $K_R = K \cap B_R(0)$. K_R is then bounded. By VI_R is denoted then the variational inequality problem:

Determine $x_R^* \in K_R$, such that

$$\langle F(x_R^*)^\top, y - x_R^* \rangle \geq 0, \quad \forall y \in K_R.$$

We now state

Theorem 2 *VI (F, K) admits a solution if and only if there exists an $R > 0$ and a solution of VI_R, x_R^*, such that $\| x_R^* \| < R$.*

Although $\| x_R^* \| < R$ may be difficult to check, one may be able to identify an appropriate R based on the particular application.

Existence of a solution to a variational inequality problem may also be established under the coercivity condition, as in the subsequent corollary.

Corollary 3 *Suppose that F(x) satisfies the coercivity condition:*

$$\frac{\langle (F(x) - F(x_0))^\top, x - x_0 \rangle}{\|x - x_0\|} \to \infty$$

as $\|x\| \to \infty$ *for* $x \in K$ *and for some* $x_0 \in K$. *Then VI(F, K) always has a solution.*

Corollary 4 *Suppose that* x^* *is a solution of VI(F, K) and* $x^* \in K^0$, *the interior of K. Then* $F(x^*) = 0$.

Qualitative properties of existence and uniqueness become easily obtainable under certain monotonicity conditions. First we outline the definitions and then present the results.

Definition 5 (*monotonicity*) $F(x)$ is *monotone* on K if

$$\langle (F(x^1) - F(x^2))^\top, x^1 - x^2 \rangle \geq 0, \quad \forall x^1, x^2 \in K.$$

Definition 6 (*strict monotonicity*) $F(x)$ is *strictly monotone* on K if

$$\langle (F(x^1) - F(x^2))^\top, x^1 - x^2 \rangle > 0, \quad \forall x^1, x^2 \in K, \quad x^1 \neq x^2.$$

Definition 7 (*strong monotonicity*) $F(x)$ is *strongly monotone* if for some $\alpha > 0$

$$\langle (F(x^1) - F(x^2))^\top, x^1 - x^2 \rangle \geq \alpha \left\| x^1 - x^2 \right\|^2, \quad \forall x^1, x^2 \in K.$$

Definition 8 (*Lipschitz continuity*) $F(x)$ is *Lipschitz continuous* if there exists an $L > 0$, such that

$$\left\| F(x^1) - F(x^2) \right\| \leq L \left\| x^1 - x^2 \right\|, \quad \forall x^1, x^2 \in K.$$

Similarly, one may define local monotonicity (strict monotonicity, strong monotonicity) if one restricts the points: x^1, x^2 in the neighborhood of a certain point $\overline{x}$. Let $B(\overline{x})$ denote a ball in $\mathbf{R}^n$ centered at $\overline{x}$.

Definition 9 (*local monotonicity*) $F(x)$ is *locally monotone* at $\overline{x}$ if

$$\langle (F(x^1) - F(x^2))^\top, x^1 - x^2 \rangle \geq 0, \quad \forall x^1, y^1 \in K \cap B(\overline{x}).$$

Definition 10 (*local strict monotonicity*) $F(x)$ is *locally strictly monotone* at $\overline{x}$ if

$$\langle (F(x^1) - F(x^2))^\top, x^1 - x^2 \rangle > 0, \quad \forall x^1, x^2 \in K \cap B(\overline{x}), \quad x^1 \neq x^2.$$

Definition 11 (*local strong monotonicity*) $F(x)$ is *locally strongly monotone* at $\overline{x}$ if for some $\alpha > 0$

$$\langle (F(x^1) - F(x^2))^\top, x^1 - x^2 \rangle \geq \alpha \left\| x^1 - x^2 \right\|^2, \quad \forall x^1, x^2 \in K \cap B(\overline{x}).$$

A uniqueness result is presented in the subsequent theorem.

Theorem 12 *Suppose that F(x) is strictly monotone on K. Then the solution is unique, if one exists.*

Similarly, one can show that if F is locally strictly monotone on K, then VI(F, K) has at most one local solution.

Monotonicity is closely related to positive definiteness.

Theorem 13 *Suppose that F(x) is continuously differentiable on K and the Jacobian matrix*

$$\nabla F(x) = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial F_n}{\partial x_1} & \cdots & \frac{\partial F_n}{\partial x_n} \end{pmatrix},$$

which need not be symmetric, is positive semidefinite (positive definite). Then F(x) is monotone (strictly monotone).

Proposition 14 *Assume that F(x) is continuously differentiable on K and that* $\nabla F(x)$ *is strongly positive definite. Then F(x) is strongly monotone.*

One obtains a stronger result in the special case where $F(x)$ is linear.

Corollary 15 *Suppose that* $F(x) = Mx + b$, *where M is an* $n \times n$ *matrix and b is a constant vector in* $\mathbf{R}^n$. *The function F is monotone if and only if M is positive semidefinite. F is strongly monotone if and only if M is positive definite.*

Proposition 16 *Assume that* $F: K \to \mathbf{R}^n$ *is continuously differentiable at* $\overline{x}$. *Then F(x) is locally strictly (strongly)*

monotone at $\overline{x}$ if $\nabla F(\overline{x})$ is positive definite (strongly positive definite), that is,

$$v^\top F(\overline{x})v > 0, \quad \forall v \in \mathbf{R}^n, \quad v \neq 0,$$
$$v^\top \nabla F(\overline{x})v \geq \alpha \|v\|^2,$$
$$\text{for some} \quad \alpha > 0, \quad \forall v \in \mathbf{R}^n.$$

The following theorem provides a condition under which both existence and uniqueness of the solution to the variational inequality problem are guaranteed. Here no assumption on the compactness of the feasible set K is made.

Theorem 17 *Assume that $F(x)$ is strongly monotone. Then there exists precisely one solution x^* to VI(F, K).*

Hence, in the case of an unbounded feasible set K, strong monotonicity of the function F guarantees both existence and uniqueness. If K is compact, then existence is guaranteed if F is continuous, and only the strict monotonicity condition needs to hold for uniqueness to be guaranteed.

Assume now that $F(x)$ is both strongly monotone and Lipschitz continuous. Then the projection $P_K\,[x - \gamma\, F(x)]$ is a contraction with respect to x, that is, we have the following:

Theorem 18 *Fix $0 < \gamma \leq \alpha\,/L^2$ where α and L are the constants appearing, respectively, in the strong monotonicity and the Lipschitz continuity condition definitions. Then*

$$\|P_K(x - \gamma F(x)) - P_K(y - \gamma F(y))\| \leq \beta \|x - y\|$$

for all $x, y \in K$, where

$$(1 - \gamma\alpha)^{1/2} \leq \beta < 1.$$

An immediate consequence of the theorem and the Banach fixed point theorem is:

Corollary 19 *The operator $P_K(x - \gamma\, F(x))$ has a unique fixed point x^*.*

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Kinderleher D, Stampacchia G (1980) An introduction to variational inequalities. Acad. Press, New York
2. Nagurney A (1993) Network economics: A variational inequality approach. Kluwer, Dordrecht

Variational Inequalities: Projected Dynamical System

PDS

ANNA NAGURNEY
University Massachusetts, Amherst, USA

MSC2000: 65K10, 90C90

Article Outline

Keywords
The Variational Inequality Problem and a Projected Dynamical System.
See also
References

Keywords

Projected dynamical system; Stationary point; Equilibrium point; Skorokhod problem; Initial value problem

A plethora of equilibrium problems, including network equilibrium problems, can be uniformly formulated and studied as finite-dimensional variational inequality problems (cf. [11] and the references therein). Indeed, it was precisely the traffic network equilibrium problem, as stated by M. Smith [15], and identified by S.C. Dafermos [3] to be a variational inequality problem, that gave birth to the ensuing research activity in variational inequality theory and applications in transportation science, regional science, operations research/management science, and, more recently, in economics.

Usually, using this methodology, one first formulates the governing equilibrium conditions as a variational inequality problem. Qualitative properties of existence and uniqueness of solutions to a variational inequality problem can then be studied using the standard theory (cf. [9]) or by exploiting problem structure (cf. [11]). Finally, a variety of algorithms for the computation of solutions to finite-dimensional variational inequality problems are now available (see, e. g., [1,4,11], and the references therein).

Finite-dimensional variational inequality theory by itself, however, provides no framework for the study of the dynamics of competitive systems. Rather, it captures the system at its equilibrium state and, hence, the focus of this tool is static in nature.

Recently, P. Dupuis and A. Nagurney [6] proved that, given a variational inequality problem, there is a naturally associated dynamical system, the stationary points of which correspond precisely to the solutions of the variational inequality problem. This association was first noted by Dupuis and H. Ishii [5]. This dynamical system, first referred to as a *projected dynamical system* by D. Zhang and Nagurney [16], is nonclassical in that its right-hand side, which is a projection operator, is discontinuous. The discontinuities arise because of the constraints underlying the variational inequality problem modeling the application in question. Hence, classical dynamical systems theory (cf. [2,7,8,10,13]) is no longer applicable.

Nevertheless, as demonstrated rigorously in [6], a projected dynamical system may be studied through the use of the *Skorokhod problem* [14], a tool originally introduced for the study of stochastic differential equations with a reflecting boundary condition. Existence and uniqueness of a solution path, which is essential for the dynamical system to provide a reasonable model, were also established therein.

Here we present some recent results in the development of a new tool for the study of equilibrium problems in a dynamic setting, which has been termed *projected dynamical systems* theory (cf. [16]). One of the notable features of this tool, whose rigorous theoretical foundations were laid in [6], is its relationship to the variational inequality problem. Projected dynamical systems theory, however, goes further than finite-dimensional variational inequality theory in that it extends the static study of equilibrium states by introducing an additional time dimension in order to allow for the analysis of disequilibrium behavior that precedes the equilibrium.

In particular, we associate with a given variational inequality problem, a nonclassical dynamical system, called a projected dynamical system. The projected dynamical system is interesting both as a dynamical model for the system whose equilibrium behavior is described by the variational inequality, and, also, because its set of stationary points coincides with the set of solutions to a variational inequality problem. In this framework, the feasibility constraints in the variational in-

equality problem correspond to discontinuities in the right-hand side of the differential equation, which is a projection operator. Consequently, the projected dynamical system is not amenable to analysis via the classical theory of dynamical systems.

We first recall the variational inequality problem. We then present the definition of a projected dynamical system, which evolves within a constraint set K. Its stationary points are identified with the solutions to the corresponding variational inequality problem with the same constraint set. We then state in a theorem the fundamental properties of such a projected dynamical system in regards to the existence and uniqueness of solution paths to the governing ordinary differential equation. We subsequently provide an interpretation of the ordinary differential equation that defines the projected dynamical system, along with a description of how the solutions may be expected to behave.

For additional qualitative results, in particular, stability analysis results, see [16]. For a discussion of the general iterative scheme and proof of convergence, see [6]. For applications to dynamic spatial price equilibrium problems, oligopolistic market equilibrium problems, and traffic network equilibrium problems, see [12], and the references therein.

The Variational Inequality Problem and a Projected Dynamical System.

We now present the definition of a variational inequality problem (VI) and that of a projected dynamical system (PDS).

Definition 1 (***variational inequality problem***) For a closed convex set $K \subset \mathbf{R}^n$ and vector function F: $K \rightarrow \mathbf{R}^n$, the variational inequality problem, VI(F, K), is to determine a vector $x^* \in K$, such that

$$\left\langle F(x^*)^\top, x - x^* \right\rangle \geq 0, \quad \forall x \in K,$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product in $\mathbf{R}^n$.

As is well-known, the variational inequality has been used to formulate a plethora of equilibrium problems ranging from traffic network equilibrium problems to spatial oligopolistic market equilibrium problems (cf. [11] and the references therein).

Finite-dimensional variational inequality theory, however, provides no framework for studying the underlying dynamics of systems, since it considers only equilibrium solutions in its formulation. Hence, in a sense, it provides a static representation of a system at its 'steady state'. One would, therefore, like a theoretical framework that permits one to study a system not only at its equilibrium point, but also in a dynamical setting.

The definition of a projected dynamical system (PDS) is given with respect to a closed convex set K, which is usually the constraint set underlying a particular application, such as, for example, network equilibrium problems, and a vector field F whose domain contains K. As noted in [6], it is expected that such projected dynamical systems will provide mathematically convenient approximations to more 'realistic' dynamical models that might be used to describe nonstatic behavior. The relationship between a projected dynamical system and its associated variational inequality problem with the same constraint set is then highlighted. For completeness, we also recall the fundamental properties of existence and uniqueness of the solution to the ordinary differential equation (ODE) that defines such a projected dynamical system.

Let $K \subset \mathbf{R}^n$ be closed and convex. Denote the boundary and interior of K, respectively, by ∂K and K^0. Given $x \in \partial K$, define the set of inward normals to K at x by

$$N(x) = \left\{\gamma\colon \ \|\gamma\| = 1, \text{ and } \left\langle \gamma^\top, x - y \right\rangle \leq 0, \ \forall y \in K\right\}.$$

We define $N(x)$ to be $\{\gamma\colon \| \gamma \| = 1\}$ for x in the interior of K.

When K is a convex polyhedron (for example, when K consists of linear constraints), K takes the form $\cap_{i=1}^{Z} K_i$, where each K_i is a closed half-space with inward normal N_i. Let P_K be the norm projection. Then P_K projects onto K 'along N', in that if $y \in K$, then $P(y) = y$, and if $y \notin K$, then $P(y) \in \partial K$, and $P(y) - y = \alpha\gamma$ for some $\alpha > 0$ and $\gamma \in N(P(y))$.

Definition 2 Given $x \in K$ and $v \in \mathbf{R}^n$, define the projection of the vector v at x (with respect to K) by

$$\Pi_K(x, v) = \lim_{\delta \rightarrow 0} \frac{(P_K(x + \delta v) - x)}{\delta}.$$

The class of ordinary differential equations that are of interest here take the following form:

$$\dot{x} = \Pi_K(x, -F(x)),$$

where K is a closed convex set, corresponding to the constraint set in a particular application, and $F(x)$ is a vector field defined on K.

Note that a classical dynamical system, in contrast, is of the form

$$\dot{x} = -F(x).$$

We have the following results (cf. [6]):

i) If $x \in K^0$, then

$$\Pi_K(x, -F(x)) = -F(x).$$

ii) If $x \in \partial K$, then

$$\Pi_K(x, -F(x)) = -F(x) + \beta(x)N^*(x),$$

where

$$N^*(x) = \arg \max_{N \in N(x)} \langle (-F(x))^\top, -N \rangle,$$

and

$$\beta(x) = \max\{0, \langle (-F(x))^\top, -N^*(x) \rangle\}.$$

Note that since the right-hand side of the ordinary differential equation is associated with a projection operator, it is discontinuous on the boundary of K. Therefore, one needs to explicitly state what one means by a solution to an ODE with a discontinuous right-hand side.

Definition 3 We say that the function x: $[0, \infty) \to K$ is a solution to the equation $\dot{x} = \Pi_K(x, -F(x))$ if $x(\cdot)$ is absolutely continuous and $\dot{x}(t) = \Pi_K(x(t), -F(x(t)))$, $-F(x(t)))$, save on a set of Lebesgue measure zero.

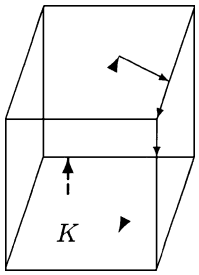
In order to distinguish between the pertinent ODEs from the classical ODEs with continuous right-hand sides, we refer to the above as ODE(F, K).

Definition 4 (*initial value problem*) For any $x_0 \in K$ as an initial value, we associate with ODE(F, K) an initial value problem, IVP(F, K, x_0), defined as:

$$\dot{x} = \Pi_K(x, -F(x)), \quad x(0) = x_0.$$

Note that if there is a solution $\phi_{x_0}(t)$ to the initial value problem IVP(F, K, x_0), with $\phi_{x_0}(0) = x_0 \in K$, then $\phi_{x_0}(t)$ always stays in the constraint set K for $t \geq 0$.

We now present the definition of a projected dynamical system, governed by such an ODE(F, K), which, correspondingly, will be denoted by PDS(F, K).



Variational Inequalities: Projected Dynamical System, Figure 1
A trajectory of a projected dynamical system that evolves both on the interior and on the boundary of the constraint set *K*

Definition 5 (*projected dynamical system*) Define the *projected dynamical system* PDS(F, K) as the map Φ: $K \times \mathbf{R} \to K$ where

$$\Phi(x, t) = \phi_x(t)$$

solves IVP(F, K, x), that is,

$$\dot{\phi}_x(t) = \Pi_K(\phi_x(t), -F(\phi_x(t))),$$
$$\phi_x(0) = x.$$

The behavior of the dynamical system is now described. One may refer to Fig. 1 for an illustration of this behavior. If $x(t) \in K^0$, then the evolution of the solution is directly given in terms of F: $\dot{x} = -F(x)$. However, if the vector field $-F$ drives x to ∂K (that is, for some t one has $x(t) \in \partial K$ and $-F(x(t))$ points 'out' of K) the right-hand side of the ODE becomes the projection of $-F$ onto ∂K. The solution to the ODE then evolves along a 'section' of ∂K, e. g., ∂K_i for some i. At a later time the solution may re-enter K^0, or it may enter a lower-dimensional part of ∂K, e. g., $\partial K_i \cap \partial K_j$. Depending on the particular vector field F, it may then evolve within the set $\partial K_i \cap \partial K_j$, re-enter ∂K_i, enter ∂K_j, etc.

We now define a stationary or an equilibrium point.

Definition 6 (*stationary point* or *equilibrium point*) The vector $x^* \in K$ is a *stationary point* or an *equilibrium point* of the projected dynamical system PDS(F, K) if

$$0 = \Pi_K(x^*, -F(x^*)).$$

In other words, we say that x^* is a stationary point or an equilibrium point if, once the projected dynamical system is at x^*, it will remain at x^* for all future times.

From the definition it is apparent that x^* is an equilibrium point of the projected dynamical system PDS(F, K) if the vector field F vanishes at x^*. The contrary, however, is only true when x^* is an interior point of the constraint set K. Indeed, when x^* lies on the boundary of K, we may have $F(x^*) \neq 0$.

Note that for classical dynamical systems, the necessary and sufficient condition for an equilibrium point is that the vector field vanish at that point, that is, that $0 = -F(x^*)$.

The following theorem states a basic connection between the static world of finite-dimensional variational inequality problems and the dynamic world of projected dynamical systems.

Theorem 7 [6] *Assume that K is a convex polyhedron. Then the equilibrium points of the PDS(F, K) coincide with the solutions of VI(F, K). Hence, for $x^* \in K$ and satisfying*

$$0 = \Pi_K(x^*, -F(x^*))$$

also satisfies

$$\left\langle F(x^*)^\top, x - x^* \right\rangle \geq 0, \quad \forall x \in K.$$

This theorem establishes the equivalence between the set of equilibria of a projected dynamical system and the set of solutions of a variational inequality problem. Moreover, it provides a natural underlying dynamics (out of equilibrium) of such systems.

Before stating the fundamental theorem about projected dynamical systems, we introduce the following assumption needed for the theorem.

Assumption 8 (linear growth condition) There exists a $B < \infty$ such that the vector field $-F$: $\mathbf{R}^n \to \mathbf{R}^n$ satisfies the linear growth condition: $\| F(x) \| \leq B(1 + \| x \|)$ for $x \in K$, and also

$$\left\langle (-F(x) + F(y))^\top, x - y \right\rangle \leq B \|x - y\|^2, \quad \forall x, y \in K.$$

Theorem 9 (existence, uniqueness, and continuous dependence) *Assume that the linear growth condition holds. Then*

i) For any $x_0 \in K$, there exists a unique solution $x_0(t)$ to the initial value problem.

ii) If $x_k \to x_0$ as $k \to \infty$, then $x_k(t)$ converges to $x_0(t)$ uniformly on every compact set of $[0, \infty)$.

The second statement of this theorem is sometimes called the *continuous dependence* of the solution path to ODE(F, K) on the initial value. By virtue of the theorem, PDS(F, K) is well-defined and inhabits K whenever the assumption holds.

Lipschitz continuity is a condition that plays an important role in the study of variational inequality problems. It also is a critical concept in the classical study of dynamical systems.

Definition 10 (*Lipschitz continuity*) F: $K \to \mathbf{R}^n$ is *locally Lipschitz continuous* if for every $x \in K$ there are a neighborhood $\eta(x)$ and a positive number $L(x) > 0$ such that

$$\left\| F(x') - F(x'') \right\| \leq L(x) \left\| x' - x'' \right\|, \quad \forall x', x'' \in \eta(x).$$

When this condition holds uniformly on K for some constant $L > 0$, that is,

$$\left\| F(x') - F(x'') \right\| \leq L \left\| x' - x'' \right\|, \quad \forall x', x'' \in K,$$

then F is said to be *Lipschitz continuous* on K.

Lipschitz continuity implies the Assumption and is, therefore, a sufficient condition for the fundamental properties of projected dynamical systems stated in the theorem.

Example 11 (Tatonnement or adjustment process) Consider the market equilibrium model in which there are n commodities. We denote the price of commodity i by p_i, and group the prices into the n-dimensional column vector p. The supply of commodity i is denoted by $s_i(p)$, and the demand for commodity i is denoted by $d_i(p)$. We are interested in determining the equilibrium pattern that satisfies the following

market equilibrium conditions: For each commodity i; $i = 1, \ldots, n$:

$$s_i(p^*) - d_i(p^*) \begin{cases} = 0 & \text{if} \quad p_i^* > 0, \\ \geq 0 & \text{if} \quad p_i^* = 0. \end{cases}$$

For this problem we propose the following *adjustment* or *tatonnement* process: For each commodity i; $i = 1$,

$\ldots, n$:

$$\dot{p}_i = \begin{cases} d_i(p) - s_i(p) & \text{if} \quad p_i > 0 \\ \max\{0, d_i(p) - s_i(p)\} & \text{if} \quad p_i = 0. \end{cases}$$

In other words, a price of an instrument will increase if the demand for that instrument exceeds the supply of that instrument; the price will decrease if the demand for that instrument is less than the supply for that instrument. However, if the price of an instrument is equal to zero, and the supply of that instrument exceeds the demand, then the price will not change since one cannot have negative prices according to equilibrium conditions.

In vector form, we may express the above as

$$\dot{p} = \Pi_K(p, d(p) - s(p)),$$

where $K = \mathbf{R}^n_+$, $s(p)$ is the n-dimensional column vector of supply functions, and $d(p)$ is the n-dimensional column vector of demand functions. Note that this adjustment process can be put into the standard form of a PDS, if we define the column vectors: $x \equiv p$ and $F(x) \equiv s(p) - d(p)$.

On the other hand, if we do not constrain the instrument prices to be nonnegative, then $K = \mathbf{R}^n$, and the above tatonnement process would take the form:

$$\dot{p} = d(p) - s(p).$$

This would then be an example of a classical dynamical system.

In the context of the example, we have then that, according to the theorem, the stationary point of prices, p^*, that is, those prices that satisfy

$$0 = \Pi_K(p^*, d(p^*) - s(p^*))$$

also satisfy the variational inequality problem

$$\langle (s(p^*) - d(p^*))^\top, p - p^* \rangle \geq 0, \quad \forall p \in K.$$

Hence, there is a natural underlying dynamics for the prices, and the equilibrium point satisfies the variational inequality problem; equivalently, is a stationary point of the projected dynamical system.

See also

- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Principles

References

1. Bertsekas DP, Tsitsiklis JN (1989) Parallel and distributed computation. Prentice-Hall, New Jersey

2. Coddington EA, Levinson N (1955) Theory of ordinary differential equations. McGraw-Hill, New York
3. Dafermos S (1980) Traffic equilibrium and variational inequalities. Transport Sci 14:42–54
4. Dafermos S (1983) An iterative scheme for variational inequalities. Math Program 26:40–47
5. Dupuis P, Ishii H (1991) On Lipschitz continuity of the solution mapping to the Skorokhod Problem, with applications. Stochastics and Stochastic Reports 35:31–62
6. Dupuis P, Nagurney A (1993) Dynamical systems and variational inequalities. Ann Oper Res 44:9–42
7. Hartman P (1964) Ordinary differential equations. Wiley, New York
8. Hirsch MW, Smale S (1974) Differential equations, dynamical systems, and linear algebra. Acad. Press, New York
9. Kinderlehrer D, Stampacchia G (1980) An introduction to variational inequalities and their applications. Acad. Press, New York
10. Lefschetz S (1957) Differential equations. Geometric theory. Interscience, New York
11. Nagurney A (1999) Network economics: A variational inequality approach. second, Kluwer, Dordrecht
12. Nagurney A, Zhang D (1996) Projected dynamical systems and variational inequalities with applications. Kluwer, Dordrecht
13. Perko L (1991) Differential equations and dynamical systems. Springer, Berlin
14. Skorokhod AV (1961) Stochastic equations for diffusions in a bounded region. Theory Probab Appl 6:264–274
15. Smith MJ (1979) Existence, uniqueness, and stability of traffic equilibria. Transport Res 13B:295–304
16. Zhang D, Nagurney A (1995) On the stability of projected dynamical systems. J Optim Th Appl 85:97–124

Variational Principles

MUHAMMAD ASLAM NOOR[1],
THEMISTOCLES M. RASSIAS[2]
[1] Math. and Statist., Dalhousie University Halifax, Halifax, Canada
[2] Math. National Techn., University Athens Zografou Campus, Athens, Greece

MSC2000: 62H30, 65Cxx, 65C30, 65C40, 65C50, 65C60, 90C05, 49J40

Article Outline

Keywords
Variational-Like Inequalities
Open Problems
See also
References

Keywords

Variational-like inequalities; Variational principles; Merit function; Invex function; Pre-invex function; Resolvent equations; Wiener–Hopf equations

The theory of variational principles is a branch of mathematical sciences with a wide range of applications in industry, physical, social, regional and engineering sciences. Researches in this theory have shown important and novel connections with all areas of pure and applied sciences. The general theory of the calculus of variations started soon after the introduction of differential and integral calculus by I. Newton and G.W. Leibniz, although some individual optimization problems had been investigated before that, the determination of the paths of light by P. Fermat. To be more specific, the brothers Jakob Bernoulli and Johann Bernoulli (1697) were the first, who considered the variational problems in mathematical terms. It is worth mentioning that the first phase of the development of the calculus of variations was characterized by a combination of philosophical concepts, mathematical methods and physical problems. L. Euler (eighteenth century) created a new branch of mathematics known as the calculus of variations. Motivated by geometrical considerations, he deduced its first principle which is now referred to as Euler's differential equation for the determination of maximizing or minimizing arcs. By variational principles, we mean: maximum and minimum problems arising in game theory, approximation theory, mechanics, geometrical optics, general relativity theory, economics, transportation, differential geometry and related areas. In fact, the history of variational principles comprises the following distinct stages:

1) The basic search for solutions of variational problems, led through the work of Euler, J.L. Lagrange, A.M. Legendre, C.G. Jacobi, K. Weierstrass and many others, to develop along the lines of differential and integral equations as well as functional analysis.
2) The Hamiltonian–Jacobi theory represents a general framework for the mathematical description of

the propagation of actions in nature and the optimal modeling of control processes in daily life. Using the ideas and techniques of Hamiltonian–Jacobi theory in mechanics, E. Cartan introduced differential geometry and his exterior calculus in the calculus of variations. Many basic equations of mathematical physics result from variational problems. It is known that the gauge fields theories are a continuation of Einstein's concept of describing physical effects mathematically in terms of differential geometry. These theories play a fundamental role in the modern theory of elementary particles and are right tool of building up a unified theory of elementary particles, which includes all kind of known interactions. For example, the Weinberg–Salam theory unifies weak and electromagnetic interactions. It is also known that the variational formulation of field theories allows for a degree of unification absent their versions in terms of differential equations. Variational principles play an important part in the existence and stability of soliton, which occur in almost every branch of physics.

3) Optimization that came into being because of equilibrium problems arising in economics and transportation from the 1950s onwards, for example, linear optimization, Kuhn–Tucker theory, Bellman dynamic optimization, Ekeland's principle and its variant forms.
4) Variational and quasivariational inequalities theory with their applications to mathematical physics, pure and applied sciences, which was introduced in 1964. Theory of variational inequalities provides us with a simple, natural, efficient and unified framework to study a wide class of unrelated problems. This theory combines the theory of extremal problems and monotone operators under a unified viewpoint. Note that every monotone operator is not a potential operator.

A last problem of great interest is the so-called *inverse problem of the calculus of variations*. A detailed exposition of the single integral problem shows the crucial role of the concept of variational selfadjoints. Selfadjointness of the linear differential operators is well known to be the key property in the inverse problems of the calculus of variations. However, E. Tonti [23] have emphasized the role played by the inner product with regard to the selfadjointness. Variational principles for nonsymmetric nonpotential operators have not been widely used either by mathematicians or in applications. One of the basic reasons for this is apparently the complexity of a constructive approach to the necessary symmetrizing operators. After Hilbert's paper, [7,24], the variational methods for investigating boundary value problems for partial differential equations, were developed and received theoretical justification. It is known that, if, for a linear nonsymmetric and nonpositive operator T, there exists an inverse operator $T-1$ on a Hilbert space H, then there exist an infinite number of auxiliary operators g such that T is g-symmetric and g-positive. For the theoretical foundation of the formulation and investigation for variational principles for both linear and nonlinear equations, see [7,23], where it is shown that the construction of a variational principle is closely related with the choice of the classes of functionals and the space.

The direct methods for solving primal variational problems provide only upper bounds, whereas the solution of the dual (complementary) problem will give lower bounds. The idea of transforming the original variational problem of a minimization of a functional into a corresponding problem of maximization and of obtaining a posteriori estimate of approximate solution goes back to C. Zaremba, E. Trefftz and K. Friedrich, see [7,24] which incidently forms the basis of three directions of obtaining dual variational principles: geometric, operator and functional. In recent years (as of 2000) with the help of operator theory, interesting and important results have been obtained in the applications of dual variational principles. The important significance of dual variational principles of obtaining them by means of Fenchel–Rockafellar inequality has been emphasized in [8], where among the basic drawbacks of other techniques have been pointed out. It has been shown in [8] that dual techniques have more favorable properties than the primal ones for nonlinear and nonsmooth systems.

It is perhaps part of the fascination of the subject that so many branches of pure and applied sciences are involved. The task of becoming conversant with a wide spectrum of knowledge is indeed a real challenge. The framework chosen should be seen as a model setting for more general results. In this article, we will consider the variational-like inequalities to describe some results in the setting of Hilbert space and list some very interest-

ing and open (as of 1999) problems for the future research.

Variational-Like Inequalities

Let H be a real Hilbert space whose inner product and norm are denoted by $\langle\cdot\rangle$ and $\|\cdot\|$ respectively. Let K be a nonempty subset of H and η: $K \times K \to H$ be a single-valued operator. Let F: $K \to H$ be a function. We now recall the following concepts and results, see, for example, [13,14,15].

Definition 1 Let $u \in K$. Then, the set K is said to be *invex* at u with respect to η, if, for each $v \in K$, and $t \in [0, 1]$, $u + t\eta(v, u) \in K$. K is said to be *invex* with respect to η, if K is invex at each $u \in K$.

From now onward, the set K is an invex set, unless otherwise specified.

Definition 2 The function F: $K \to H$ is said to be *pre-invex* with respect to η, if, for all u, $v \in K$ and $t \in [0, 1]$,

$$F(u + t\eta(v, u)) \leq (1 - t)F(u) + tF(v).$$

Definition 3 For all u, $v \in K$, the differentiable function F: $K \to H$ is said to be an *invex function* with respect to η if

$$F(v) - F(u) \geq \langle F'(u), \eta(v, u)\rangle,$$

where $F'(u)$ is the differential of F at u.

Remark 4 It is known that every differentiable pre-invex function is an invex function, but the converse is not true. However, if $\eta(v, u) = v - u$, then both pre-invex and invex functions are convex functions and the invex set is a convex set. If F is a differentiable pre-invex function and φ is a pre-invex function, then it is known [13,15] that the minimum u of the functional $I[v]$, where

$$I[v] = F(v) + \varphi(v) \quad \text{for all } v \in K, \tag{1}$$

on the invex set K in H can be characterized by the variational-like inequality

$$\langle F'(u), \eta(v, u)\rangle + \varphi(v) - \varphi(u) \geq 0 \quad \text{for all } v \in K. \tag{2}$$

It is well known that in many important applications, variational-like inequalities (2) occur, which do not arise as a result of extremum problems. This motivates the interest of studying problem like (2) on its own, that is, without assuming a priori that this comes out as an Euler inequality of an extremum problem.

For a given nonlinear operator T: $H \to H$, we consider the problem of finding $u \in H$ such that

$$\langle Tu, \eta(v, u)\rangle + \varphi(v) - \varphi(u) \geq 0 \quad \text{for all } v \in H. \tag{3}$$

Clearly problem (2) is a special case of problem (3). First of all, we discuss some special important cases: In particular, if $\eta(v, u) = v - u$, then problem (3) is equivalent to finding $u \in H$ such that

$$\langle Tu, v - u\rangle + \varphi(v) - \varphi(u) \geq 0 \quad \text{for all } v \in H, \tag{4}$$

which is known as the *mixed variational inequality*. Note that the function φ: $H \to \mathbf{R} \cup \{+\infty\}$ is a proper, convex and lower semicontinuous, whose subdifferential $\partial\varphi(u)$ is a maximal monotone operator. For applications of problem (4), see [3,4,5,6,9,10,12,13,14,15,17,18] and the references therein. Problem (4) can be written in the equivalent form as: Find $u \in H$ such that

$$0 \in Tu + \partial\varphi(u), \tag{5}$$

which is equivalent to finding $u \in H$ such that

$$u = J_\varphi[u - \rho Tu], \tag{6}$$

where $J_\varphi = (I + \rho\,\partial\varphi)^{-1}$ is the *resolvent operator* associated with the maximal monotone operator $\partial\varphi$, a subdifferential of the proper, convex and lower semicontinuous function φ; and $\rho > 0$ is a constant. Problem (5) is also known as the *variational inclusion*; see [16] and the references therein for more details.

If φ^* is the *conjugate function* of φ, then its subdifferential $\partial\varphi^*$ is also a maximal monotone operator and $J_{\varphi^*} = (I + \partial\varphi^*)^{-1}$ is the resolvent operator associated with $\partial\varphi^*$. From the definitions of the resolvent operators, we have, for all u, $v \in H$,

$$\begin{aligned} u = J_\varphi(u + v) &\Leftrightarrow v \in \partial\varphi(u) \\ &\Leftrightarrow \varphi^*(v) + \varphi(u) = \langle v, u\rangle \\ &\Leftrightarrow u \in \partial\varphi^*(v) \Leftrightarrow v = J_\varphi(u + v). \end{aligned}$$

From this result, we have

$$z = J_{\varphi}(z) + J_{\varphi^*}(z) \quad \text{for all } z \in H,$$

a beautiful and useful relationship between the resolvent operators. It has been shown [14] that the problem (4) is equivalent to finding $z \in H$ such that

$$\rho T J_{\varphi} z + J_{\varphi *} z = 0. \tag{7}$$

Equations (7) are called the *resolvent equations*. Such an equivalent interplay has played an important part in suggesting various iterative methods for solving mixed variational inequalities. If φ is an indicator function of a closed convex set K in H, then $J_{\varphi} \equiv P_K$, the projection of H onto K; as a consequence, resolvent equations are equivalent to the *Wiener–Hopf equations*, introduced and studied in [21] and [20] in connection with classical variational inequalities. See [14,17,18] for the physical formulation and numerical methods of the Wiener–Hopf equations.

Remark 5 Above, we have tried to emphasize the role played by the concepts of the invexity theory in variational-like inequalities. Unfortunately, all the existence theory for variational-like inequalities has been developed in the setting of the standard convexity up to now. It is right time to study the variational-like inequalities in context of invex functions and invex sets. We would like to point out that the projection and resolvent equations techniques cannot be extended and modified to study the existence results and to suggest iterative methods for variational-like inequalities due to the presence of the function η and the nonlinear pre-invex function φ. See [13,15] for the auxiliary principle technique to suggest a general iterative method and a merit function for solving variational-like inequalities.

Open Problems

In this section, we list a number of open problems which can play an important role in the development of variational-like inequalities.

1) Is the subdifferential of a preinvex function a maximal monotone operator?
2) Does there exist a resolvent(projection) operator associated with the subdifferential of a proper, pre-invex (invex) and lower semicontinuous function?
3) There are a number of merit (gap) functions for variational inequalities and complementarity problems. Is it possible to construct similar merit (gap) functions for variational-like inequalities? M.A. Noor [15] has constructed a merit (gap) function for variational-like inequalities under some conditions.
4) Study the sensitivity analysis for variational-like inequalities.
5) Can one apply the Ky Fan inequality or any other minimax theory to study the existence of a solution of variational-like inequalities in the context of invexity theory?
6) In recent years (as of 2000), Ekeland's principle has played a significant part in various branches of pure and applied sciences, see, for example, [2,6,12] and the references therein. Is it possible to find a similar variational principle for pre-invex (invex) functions?

In this article, we have given only a brief introduction of variational-like inequalities. This theory does not appear to have developed to an extent that it provides a complete framework for studying various problems. This field has been continuing and will continue to foster new, innovative and novel applications. The interested reader is advised to explore this fascinating field further and discover interesting and significant applications.

It is not practical to quote sufficient up-to-date references. We shall therefore constrain to various references with which the authors have recently (as of 1999) been associated. Perhaps some of these point to future possibilities.

See also

- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonconvex-nonsmooth Calculus of Variations
- ▶ Nonsmooth and Smoothing Methods for Nonlinear Complementarity Problems and Variational Inequalities

- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System

References

1. Arthurs AM (1980) Complementary variational principles. second, Clarendon Press, Oxford
2. Aubin JP (1993) Optima and equilibria. Springer, Berlin
3. Baiocchi C, Capelo A (1984) Variational and quasi variational inequalities. Wiley, New York
4. Brezis H (1973) Operateur maximaux monotone et semigroupes de contractions dans les espaces de Hilbert. North-Holland, Amsterdam
5. Crank J (1984) Free and boundary problems. Clarendon Press, Oxford
6. Ekeland I, Temam R (1976) Convex analysis and variational problems. North-Holland, Amsterdam
7. Filippov VM (1989) Variational principles for nonpotential operators. Monograph, vol 77. Amer. Math. Soc., Providence, RI
8. Gao DY (1999) Duality principles in non-convex systems: Theory, methods and applications. Kluwer, Dordrecht
9. Giannessi F, Maugeri A (1995) Variational inequalities and network equilibrium problems. Plenum, New York
10. Glowinski R, Lions JL, Tremolieres R (1981) Numerical analysis of variational inequalities. North-Holland, Amsterdam
11. Herrera I, Sewell MJ (1978) Dual extremum principles for non-negative unsymmetric operators. J Inst Math Appl 21:95–115
12. Hyers DH, Isac G, Rassias ThM (1997) Topics in nonlinear analysis and applications. World Sci., Singapore
13. Noor MA (1994) Variational-like inequalities. Optim 30:323–330
14. Noor MA (1997) Some recent advances in variational inequalities, Part I (basic concepts), Part II (other concepts). New Zealand J Math 26:53–80; 229–255
15. Noor MA (2000) Merit function for variational-like inequalities. Math Ineq Appl 3(1):117–128
16. Noor MA (2001) Three-step iterative algorithms for multivalued quasi variational inclusions. J Math Anal Appl
17. Noor MA, Noor KI, Rassias ThM (1993) Some aspects of variational inequalities. J Comput Appl Math 47:285–312
18. Noor MA, Noor KI, Rassias ThM (1998) Set-valued resolvent equations and mixed variational inequalities. J Math Anal Appl 220:741–759
19. Rassias GM, Rassias ThM (1985) Differential geometry, calculus of variations and their applications. M. Dekker, New York
20. Robinson SM (1992) Normal maps induced by linear transformations. Math Oper Res 17:691–714
21. Shi P (1991) Equivalence of variational inequalities with Wiener–Hopf equations. Proc Amer Math Soc 111:339–346
22. Srivastava HM, Rassias ThM (1993) Analysis, geometry and groups: A Riemann legacy volume. Hadronic Press, Palm Harbor
23. Tonti E (1984) Variational formulation for every nonlinear problem. Inter J Eng Sci 22:1343–1371
24. Zeidler E (1985) Nonlinear functional analysis and its applications III, Variational methods and optimization. Springer, Berlin

Vector Optimization

MASSIMO PAPPALARDO
Department Applied Math., University of Pisa, Pisa, Italy

MSC2000: 90C29

Article Outline

Keywords
See also
References

Keywords

Pareto optimality; Ordering cones; Scalarization

Vector optimization is the discipline which studies the approaches for selecting optimal decisions from a given admissible (feasible) set in presence of two (bicriteria) or more (multicriteria) conflicting objectives. The increasing interest, in the last decades, towards this discipline is mainly due to the fact that, in optimization processes, it seems more realistic to accept the presence of more than one objective. This, in fact, happens in economic systems (maximization of the profit and minimization of the risk in portfolio selection problems), in engineering systems an in physical systems (see, for example, [23]) and so on.

From the pioneering works of V. Pareto and M.-E.-L. Walras at the end of the nineteenth century, where the authors introduce a (vector) equilibrium concept for economical systems, vector optimization gained mathematical recognition with the Kuhn–Tucker definition of vector maximum given in 1951. Such a definition, as we will see later, clarifies that the mathematical foundations of this discipline must be found in the fundamental works of G. Cantor and F. Hausdorff concerning orderings and ordered set in vector spaces. The birth of the discipline in the economic context provides the justification of the presence of many typical terms like utility functions, decision process, preferring order, equilibrium model.

The first mathematical step consists in introducing a preference in a set, which will be called *decision set* D. This, mathematically speaking, can be translated in considering a binary relation R (i. e. a subset of $D \times D$) on D. For economists, a *preference* is a *partial order* R, i. e. a binary relation satisfying the following two properties

a) $(x, x) \in R$, $\forall x \in D$ (reflexivity);
b) $(x, y) \in R$, $(y, z) \in R \Rightarrow (x, z) \in R$ (transitivity).

In order to have compatibility between the partial order R and the structure of vector space of D it is common to assume that the following two axioms hold:

c) $\forall \alpha \geq 0$: $(x, y) \in R \Rightarrow (\alpha x, \alpha y) \in R$;
d) $[(x, y) \in R, (z, w) \in R] \Rightarrow (x + z, y + w) \in R$.

A fundamental property, when it holds, is the antisymmetry of R:

e) $(x, y) \in R, (y, x) \in R \Rightarrow x = y$.

The following theorem shows the strict relationship between partial order and cones, and it gives the reason of the common term 'the ordering cone is…'.

Theorem 1

a) If R is a partial order then

$$C = \{x \in D\colon\ (x, 0) \in R\}$$

is a convex cone; if, in addition, R is antisymmetric, then C is pointed.

b) If C is a convex cone, then

$$R = \{(x, y) \in D \times D\colon\ x - y \in C\}$$

is a partial order on D; if, in addition, C is pointed, then R is antisymmetric.

For the sake of simplicity from now on we shall suppose that $D \subset \mathbf{R}^n$ and we shall consider, in this context, the most often used and best known ordering cone in $\mathbf{R}^n$, which is called the *Paretian cone*, that is $C = \mathbf{R}^n_+ \backslash \{0\} = \{x \in \mathbf{R}^n\colon x_i \geq 0,\ i = 1, \ldots, n\} \backslash \{0\}$. Naturally many other types of convex cones have been considered in literature in different situations but our treatment can be easily generalized to those cases.

The following definition is crucial in this framework.

Definition 2 y is a minimum of the set A with respect to the cone C, and we will write $y \in \min_C A$, if and only if the system

$$y - x \in C, \quad x \in A,$$

is impossible.

From the above definition, the natural consequence is contained in the following:

Definition 3 Given $f\colon \mathbf{R}^m \to \mathbf{R}^n$ and a subset $D \subset \mathbf{R}^m$, a point $\widehat{x} \in D$ is called a *Pareto* (or efficient) solution of

$$(P) \begin{cases} \min\limits_C f(x) \\ x \in D \end{cases}$$

if and only if $f(\widehat{x})$ is a minimum of $f(D)$ with respect to the cone C.

These minimum points are called Pareto (or efficient) points. If we replace C with int$C$ we have the classical relaxed definition of weak Pareto (or weak efficient) points. The notion of efficiency has been restricted to *proper efficiency* in order to avoid some undesirable situations.

Unfortunately, there are several different definitions of proper efficiency [4,6,11] and this makes more difficult the development of the analysis with respect to this aspect.

After having given the definition of minimum point many relevant questions come:

1) Under what conditions can we ensure the existence of a solution of problem (P)?
2) What conditions can be established for a minimum point (necessary or sufficient optimality conditions)?
3) How can we determine the minimum (when it exists)?

For giving the fundamental ideas for answering the above questions we can restrict ourselves to the most classical case in which

$$D = \{x \in \mathbf{R}^m :\ g(x) \leq 0,\ h(x) = 0\},$$

where g: $\mathbf{R}^m \to \mathbf{R}^s$ and $\mathbf{R}^m \to \mathbf{R}^k$.

A first classical theorem for the existence of the minimum needs the following definition.

Definition 4 f is called $\mathbf{R}^n_+$-upper semicontinuous at $x_0 \in D$ if and only if for every neighborhood V of $f(x_0)$, there exists a neighborhood I of x_0 such that $f(x) \in V - \mathbf{R}^n_+$, $\forall x \in I \cap D$.

Now we are able to state the following:

Theorem 5 *Let us suppose that D is compact and $-f$ is $\mathbf{R}^n_+$-upper semicontinuous in D. Then the set of optimal solution of (P) is nonempty.*

This theorem is a generalization to vector optimization of the well known Weierstrass theorem. Many generalizations of it can be found in the literature (see, for example, [18]).

Necessary optimality conditions of Lagrangian type can be established under classical assumptions of continuous differentiability of f, g and h. The following theorem holds.

Theorem 6 *Suppose that (P) satisfies the classical Kuhn–Tucker constraints qualification (or some generalization of it) at $\widehat{x} \in D$ [2,21]. Then, a necessary condition for $\widehat{x}$ to be a weak Pareto solution of (P) is that there exist*

$$\widehat{\mu} \in \mathbf{R}^n, \quad \widehat{\lambda} \in \mathbf{R}^s, \quad \widehat{\sigma} \in \mathbf{R}^k$$

such that

$$(\widehat{\mu}, \widehat{\lambda}, \widehat{\sigma}) \neq (0, 0, 0)$$

and

$$(A) \begin{cases} \langle \widehat{\mu}, \nabla f(\widehat{x}) \rangle + \left\langle \widehat{\lambda}, \nabla g(\widehat{x}) \right\rangle \\ \quad + \langle \widehat{\sigma}, \nabla h(\widehat{x}) \rangle = 0, \\ \left\langle \widehat{\lambda}, g(\widehat{x}) \right\rangle = 0, \\ \widehat{\mu} \geq 0, \quad \widehat{\lambda} \geq 0. \end{cases}$$

Addition of convexity to the assumption of the Theorems leads us to sufficient optimality conditions:

Theorem 7 *If all f_i and g_j are convex and all h_k are affine, then condition (A) in Theorem 6 is sufficient for $\widehat{x} \in D$ to be a weak Pareto solution to (P).*

Theorems 6 and 7 are classical results and they are the starting point in the field of optimality conditions. Developments of such theorems can be found in literature (see, for example, [16,18,21]). We can observe that the generalizations go in several different directions:

a) to remove the assumptions of differentiability;
b) to weaken the constraint qualifications assumptions;
c) to strengthen the optimality conditions for other types of optimal solutions.

Another research field is the characterization of efficient points. Most well known results regarding the characterization of efficient points are via *scalarization* by means of vectors of weights belonging to the polar cone of the ordering cone. This leads to find an 'equivalent' scalar optimization problem in the following sense:

Theorem 8 *Suppose that all f_i are convex. Then $\widehat{x}$ is a weak Pareto solution if and only if there exists $\mu \in \mathbf{R}^n_+$, $\mu \neq 0$ such that $\widehat{x}$ is a minimum point of the function $\langle \mu, f \rangle$ on the set D.*

Let us observe that, in Theorem 8, the assumption of convexity of all f_i is not required in the proof of sufficiency but only in the proof of the necessity.

When the objective functions and the constraints are defined by linear or affine functions we have the so-called *multi-objective linear programming*. In this case the set of efficient points is connected and it is possible to derive an algorithm, which is a generalization of the simplex method, in order to locate the entire set of efficient points.

Finally, we recall that it is possible to develop a duality theory for vector optimization like in the scalar case.

In fact, it is well known from scalar optimization that, under suitable assumptions, a minimization problem can be associated to a maximization problem such that both problems have the same optimal solutions. This scheme is called, in literature, duality and it provides useful tools in order to have a deeper knowledge of the given problem and, moreover, it provides important informations in order to develop algorithms for solving the given problem. A similar general duality principle holds for vector optimization problems and it can be specialized to linear vector problems.

See also

► Image Space Approach to Optimization

References

1. Arrow KJ (1968) Economic equilibrium. In: Sills DL (ed) Internat Encycl Social Sci, 4 MacMillan, London, pp 376–389
2. Bigi G, Pappalardo M (1999) Regularity conditions in vector optimization. J Optim Th Appl 102(1)
3. Borel E (1953) The theory of play and of integral equations with skew symmetric kernels. CR Acad Sci 173:1304–1308. English translation in: Econometrica 21: 1:97–100 (In French)
4. Borwein JM (1977) Proper efficient points for maximization with respect to cones. SIAM J Control Optim 15:57–63
5. Borwein JM (1980) The geometry of Pareto efficiency over cones. Math Operationsforsch Statist 11:235–248
6. Borwein JM, Zhuang D (1993) Superefficiency in vector optimization. Trans Amer Math Soc 338:105–122
7. Cantor G (1895) Contributions to the foundation of transfinite set theory. Math Ann 46:491–512. (1987) 49:207–246 (In German)
8. Dauer JP, Stadler W (1986) A survey of vector optimization in infinite-dimensional spaces, Part 2. J Optim Th Appl 51:205–251
9. Edgeworth FY (1881) Mathematical physics. C. Kegan Paul, London
10. Geoffrion AM (1968) Proper efficiency and the theory of vector maximization. J Math Anal Appl 22:618–630
11. Guerraggio A, Molho E, Zaffaroni A (1994) On the nature of proper efficiency in vector optimization. J Optim Appl 82(1):1–21
12. Hartley R (1978) On cone efficiency, cone convexity, and cone compactness. SIAM J Appl Math 34:211–222
13. Hausdorff F (1906) Investigations concerning order types. Berichte Verhandl K Sächsischen Gesellschaft Wissenschaft Leipzig, Math–Phys Kl 58:106–169. (In German.)
14. Hurwicz L (1958) Programming in linear spaces. In: Arrow KJ Hurwicz L, Uzawa H (eds) Studies in Linear and Nonlinear Programming. Stanford Univ. Press, Palo Alto, CA Second Printing, Oxford UP 1964, pp 38–102
15. Jahn J (1985) A characterization of properly minimal elements of a set. SIAM J Control Optim 23:649–656
16. Jahn J (1986) Mathematical vector optimization in partially ordered linear spaces. P. Lang, Frankfurt am Main
17. Kuhn HW, Tucker AW (1951) Nonlinear Programming: Proc. 2nd Berkeley Symp. Math. Statistics and Probability. Berkeley, pp 481–492
18. Luc DT (1989) Theory of vector optimization. Springer, Berlin
19. Neumann J von (1928) On the theory of parlor games. Math Ann 100:295–320. (In German.)
20. Pareto V (1906) Manuale di economia politica. Soc. Ed. Libraria, Milano. Translated into French, with revised Mathematical Appendix, by Girard and Brière, as Manual D'Economie Politique, Giard, Paris, 1st edn. 1909 and 2nd edn. 1927. Translated into English by Schwier AS (1971) Manual of Political Economy. The MacMillan Company, New York
21. Sawaragi Y, Nakayama H, Tanino T (1985) Theory of multiobjective optimization. Acad. Press, New York
22. Stadler W (1979) A survey of multicriteria optimization or the vector maximal problem, Part 1: 1776–1960. J Optim Th Appl 29:1–52
23. Stadler W (1984) Multicriteria optimization in mechanics (a survey). Applied Mechanics Reviews 37(3):277–286

Vector Variational Inequalities

X. Q. YANG
Department of Applied Mathematics,
The Hong Kong Polytechnic University,
Kowloon, China

MSC2000: 58E35, 90C29

Article Outline

References

The vector variational inequality is a mathematics model which is designed to account for equilibrium situations where the multicriteria consideration is important. The concept of a vector variational inequality was introduced in [5]. In recent years, the vector variational inequality problem has received extensive attentions and found many applications in vector optimization and vector network equilibrium problems. The theory of vector variational inequalities has been summarized in the edited book [7] and one chapter of the monograph [1].

Let X and Y be Hausdorff topological vector spaces. By $L(X, Y)$, we denote the set of all linear continuous functions from X into Y. For $l \in L(X, Y)$, the value of linear function l at x is denoted by $\langle l, x\rangle$. Let $C \subset Y$ be a nonempty, pointed, closed and convex cone with $intC \neq \emptyset$. For convenience, we will denote $C \setminus \{0\}$ and $intC$ by C_o and $\hat{C}$ respectively. Then (Y,C) is an ordered Hausdorff topological vector space with a partial ordering defined by, for $y_1, y_2 \in Y$,

$$y_1 \leq_C y_2 \Longleftrightarrow y_2 - y_1 \in C\,.$$

Moreover, we also define

$$y_1 \not\leq_{C_o} y_2 \Longleftrightarrow y_2 - y_1 \notin C_o\,;$$
$$y_1 \not\leq_{\hat{C}} y_2 \Longleftrightarrow y_2 - y_1 \notin \hat{C}\,.$$

These orderings can also be applied to sets where the ordering is understood as element-wise.

Let $T : K \to L(X, Y)$ and $K \subset X$ be a nonempty closed and convex subset.

A weak vector variational inequality (WVVI) is a problem of finding $x^* \in K$ such that

$$\langle T(x^*), x - x^*\rangle \not\leq_{\hat{C}} 0, \quad \forall x \in K. \tag{WVVI}$$

A vector variational inequality (VVI) is a problem of finding $x^* \in K$ such that

$$\langle T(x^*), x - x^*\rangle \not\leq_{C_o} 0, \quad \forall x \in K. \tag{VVI}$$

It is clear that "$\not\leq_{\hat{C}}$" is a closed ordering, that is, $x_k \not\leq_{\hat{C}} 0$ and $x_k \to x$ imply $x \not\leq_{\hat{C}} 0$, but "$\not\leq_{C_o}$" is not. As such, the set of solutions for (WVVI) is closed and that for (VVI) is not. When $Y = \mathbb{R}$ and $X = \mathbb{R}^n$, (WVVI) and (VVI) reduce to the variational inequality, see [8].

Consider a vector optimization problem:

$$\min_{\substack{C \\ x \in K}} f(x), \tag{VOP}_K$$

where $f : X \to Y$ is a vector-valued function. The point $x^* \in K$ is said to be a weakly minimal solution of f on K if and only if $f(K) \not\leq_{\hat{C}} f(x^*)$ and a minimal solution of f on K if and only if $f(K) \not\leq_{C_o} f(x^*)$.

Let $X = \mathbb{R}^n$, $Y = \mathbb{R}^\ell$ and $C = \mathbb{R}^\ell_+$. Let $f(x) := (f_1(x), \cdots, f_\ell(x))^\top$. A point $x^* \in K$ is said to be a Geoffrion properly minimal solution of $(\text{VOP})_K$ if and only if there exists a scalar $M > 0$ such that, for each i,

$$\frac{f_i(x^*) - f_i(x)}{f_j(x) - f_j(x^*)} \leq M\,,$$

for some j such that $f_j(x) > f_j(x^*)$ whenever $x \in K$ and $f_i(x) < f_i(x^*)$. Every Geoffrion properly minimal solution is a minimal solution.

$f : X \to Y$ is C-convex on K if and only if, for any $x_1, x_2 \in K, \lambda \in [0, 1]$,

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq_C \lambda f(x_1) + (1 - \lambda) f(x_2)\,.$$

The following proposition summarizes relationships between (WVVI)/ (VVI) and the vector optimization problem $(\text{VOP})_K$. See [2,13].

Proposition 1 *Assume that f is Gâteaux differentiable with Gâteaux derivative Df. Let $T = Df$. We have*

(i) If x is a weakly minimal solution of $(\text{VOP})_K$, then x solves (WVVI).

(ii) If f is C-convex and x solves (WVVI), then x is a weakly minimal solution of $(\text{VOP})_K$.

(iii) If f is C-convex and x solves (VVI), then x is a minimal solution of $(\text{VOP})_K$.

(iv) If $-f$ is C-convex and x^* is a minimal solution of $(\text{VOP})_K$, then x solves (VVI).

(v) Let $T(x) = \nabla f(x) := (\nabla f_1(x), \cdots, \nabla f_\ell(x))^\top$ be the Jacobian (an $\ell \times n$ matrix) of the vector-valued function f at x. If f is $C = \mathbb{R}^\ell_+$-convex and x^* is a Geoffrion properly minimal solution for $(\text{VOP})_K$, then x^* solves (VVI).

Without the C-convexity of $-f$, (iv) may not be true. Let $X = \mathbb{R}$, $Y = \mathbb{R}^2$ and $C = \mathbb{R}^2_+$. Consider the problem $\min_C f(x)$, subject to $x \in [-1, 0]$ where $f(x) =$

$(x, x^2+1)^\top$. It is clear that every $x \in [-1, 0]$ is a minimal solution of the problem. But $x = 0$ is not a solution of (VVI). The set of solutions for (WVVI) and (VVI) is [−1,0] and [−1,0) respectively.

A Minty vector variational inequality (Minty VVI, in short) is a problem of finding $x^* \in K$ such that

$$\langle T(x), x - x^* \rangle \not\leq_{C_o} 0, \ \forall x \in K . \tag{1}$$

The following result shows that a minimal solution of $(\mathrm{VOP})_K$ can be completely characterized by the Minty VVI when $C = \mathbb{R}^\ell_+$.

Theorem 1 [6] *Let $X = \mathbb{R}^n$, $Y = \mathbb{R}^\ell$ and $C = \mathbb{R}^\ell_+$. Let $T(x) = \nabla f(x)$. Let f be $\mathbb{R}^n_+$-convex and v-hemicontinuous on K. Then, x^* is a minimal solution of $(\mathrm{VOP})_K$ if and only if it is a solution of the Minty VVI.*

The following generalized linearization lemma and Knaster, Kuratowski and Mazurkiewicz Theorem (KKM Theorem, in short) have played a key role in the establishment of the existence of a solution for (WVVI).

Lemma 1 (Generalized Linearization Lemma) *Let the mapping $T : X \to L(X, Y)$ be monotone and v-hemicontinuous. Then the following two problems are equivalent:*

1. $x \in K, \langle T(x), y - x \rangle \not\leq_{\hat{C}} 0, \quad \forall y \in K;$
2. $x \in K, \langle T(y), y - x \rangle \not\leq_{\hat{C}} 0, \quad \forall y \in K.$

Lemma 2 (KKM Theorem) *Let K be a subset of a topological vector space V. For each $x \in K$, let a closed and convex set F(x) in V be given such that F(x) is compact for at least one $x \in K$. If the convex hull of every finite subset $\{x_1, x_2, \cdots, x_k\}$ of K is contained in the corresponding union $\cup_{i=1}^n F(x_i)$, then $\cap_{x \in K} F(x) \neq \emptyset$.*

Assume that K is compact. We set

$$F_1(y) = \{x \in K : \langle T(x), y - x \rangle \not\leq_{\hat{C}} 0\}, \quad y \in K ,$$
$$F_2(y) = \{x \in K : \langle T(y), y - x \rangle \not\leq_{\hat{C}} 0\}, \quad y \in K .$$

It can be shown that the convex hull of every finite subset $\{x_1, x_2, \cdots, x_k\}$ of K is contained in the corresponding union $\cup_{i=1}^n F_1(x_i)$. Since $F_1(y) \subset F_2(y)$ for all $y \in K$, this is also true for F_2. By Lemma 1, we have

$$\cap_{y \in K} F_1(y) = \cap_{y \in K} F_2(y) .$$

We observe that for each $y \in K$, $F_2(y)$ is a (weakly) compact subset in K.

By Lemma 2, we have

$$\cap_{y \in K} F_1(y) = \cap_{y \in K} F_2(y) \neq \emptyset .$$

Hence, there exists an $x^* \in K$ such that

$$\langle T(x^*), x - x^* \rangle \not\leq_{\hat{C}} 0 , \quad \forall x \in K .$$

Assume that K is unbounded and $T : K \to L(X, Y)$ is weakly coercive on K, that is, there exist $x_0 \in K$ and $c \in intC^*$ such that

$$\langle c \circ T(x) - c \circ T(x_0), x - x_0 \rangle / ||x - x_0|| \to +\infty,$$

whenever $x \in K$ and $||x|| \to +\infty$. In a similar way, we can show that $\cap_{y \in K} F_1(y) \neq \emptyset$.

As such we have the following result, where the weak topology of X and the norm topology of Y are used.

Theorem 2 [2] *Assume that X is a reflexive Banach space and $K \subset X$ is convex. Assume that (Y,C) is an ordered Banach space with $\hat{C} \neq \varnothing$ and $intC^* \neq \varnothing$. Let the mapping $T : K \to L(X, Y)$ be monotone, v-hemicontinuous and let, for any $y \in K$, T(y) be completely continuous on X. If*

1. *K is compact, or*
2. *K is closed, T is weakly coercive on K,*

then the weak vector variational inequality (WVVI) is solvable.

KKM Theorem cannot be applied to the establishment of the existence of (VVI) as the sets $F_1(x)$ and $F_2(x)$ where $\not\leq_{\hat{C}}$ is replaced by $\not\leq_{C_o}$ are not closed anymore.

Only very recently, an existence of a solution for (VVI) has been obtained by using the Browder fixed point theorem.

Theorem 3 [4] *Assume that X is a reflexive Banach space and $K \subset X$ is convex. Assume that (Y,C) is an ordered Banach space with $\hat{C} \neq \varnothing$ and $intC^* \neq \varnothing$. Let the mapping $T : K \to L(X, Y)$. If*

1. *K is compact, and for each $y \in K$, the set $\{x \in K : \langle T(x), y - x \rangle \leq_{C_o} 0\}$ is open in K, or*
2. *K is closed, T is continuous, and weakly coercive on K,*

then the vector variational inequality (VVI) is solvable.

The study of a vector variational inequality has also been pursued by introducing another model with a similar form and using the tool of conjugate function of

a vector-valued function. Here such a model is called a primitive of a VVI. We remark that this model was called a dual VVI in [12] and an inverse VVI in [1] respectively.

Let $T : X \longrightarrow L(X, Y)$ be a function, and $h : X \to Y$ is a function. The (VVI$_h$) problem consists in finding $x^* \in X$, such that

$$\langle T(x^*), x - x^* \rangle \not\leq_{C_o} h(x^*) - h(x), \quad \forall x \in X .$$

Assume that T is one-to-one (injective). Define $T' : L(X, Y) \to X$ as follows:

$$T'(l) := -T^{-1}(-l), \quad \forall l \in \text{Domain}(T') = -\text{Range}(T).$$

If T is linear, then $T' = T^{-1}$.

The primitive of (VVI$_h$) problem is defined as: finding $l^* \in \text{Domain}(T')$, such that

$$\langle l - l^*, T'(l^*) \rangle \not\leq_{C_o} h^*_{\leq}(l^*) - h^*_{\leq}(l), \quad \forall l \in L(X, Y) , \quad (\text{IVVI}_h)$$

where $h^*_{\leq}(l) := \text{Max}_C\{\langle l, x\rangle - h(x) : x \in X\}$ is the vector conjugate function of h.

Let $h : X \to Y$ and $x^* \in X$. We define the subgradient of h at x^* by

$$\partial_{\leq} h(x^*) = \{l \in L(X, Y) : h(x) - h(x^*) \not\leq_{C_o} \langle l, x - x^*\rangle, \quad \forall x \in X\} .$$

Theorem 4 [12] *Let X be a Hausdorff topological vector space and (Y,C) be an ordered Hausdorff topological vector space. The function T is one-to-one and $h : X \to Y$ is continuous. Assume that $h^*_{\leq}(l) \neq \varnothing, \forall l \in L(X, Y)$.*

(i) If x^* is a solution of (VVI$_h$), then $l^* = -T(x^*)$ is a solution of (IVVI$_h$) and the following relation is satisfied:

$$\langle l^*, x^*\rangle \in h(x^*) + h^*_{\leq}(l^*) .$$

(ii) If l^* is a solution of (IVVI$_h$), C is connected, i. e., $C \cup (-C) = Y$, and $\partial_{\leq} h(x^*) \neq \varnothing$, where $x^* = -T'(l^*)$, then x^* is a solution of (VVI$_h$).

Consider the (VOP) with $X = K = \mathbb{R}^n, Y = \mathbb{R}^\ell, f$ being differentiable. Let $h : \mathbb{R}^n \rightrightarrows \mathbb{R}^\ell$ be a set-valued function and $x^* \in X$. We define the weak subgradient of h at x^* by

$$\partial^w h(x^*) = \{l \in \mathbb{R}^n \times \mathbb{R}^\ell : h(x) - h(x^*) \not\leq_{\hat{C}} \langle l, x - x^*\rangle, \quad \forall x \in X\} .$$

Let $\phi : \mathbb{R}^n \times \mathbb{R}^\ell \longrightarrow \mathbb{R}^\ell$ be a perturbation function satisfying

$$\phi(x, 0) = h(x), \quad \forall x \in \mathbb{R}^n ,$$

and

$$W(u) = -\text{Max}_{\hat{C}}\{-\phi(x, u) : x \in \mathbb{R}^n\} .$$

Now we construct the dual problem (for short, DVOP) of (VOP) as follows

$$\min_C -\phi^*_{<}(0, \Gamma), \quad \text{subject to} \quad \Gamma \in \mathbb{R}^{n\times\ell} .$$

Proposition 2 *Assume that W has a weak subgradient at $u = 0$ and C is connected. If x^* is a solution of (VOP), then there exists $\Gamma_0 \in \mathbb{R}^{n\times\ell}$ such that $l^* = -\nabla f(x^*)$ is a solution of the primitive of a vector variational inequality and Γ_0 is a solution of (DVOP) and satisfy the inclusion*

$$(l^{*\top}, \Gamma_0) \in \partial^w \phi(x^*, 0) .$$

The concept of a gap function is well-known both in the context of convex optimization and variational inequalities. The minimization of gap functions is a viable approach for solving variational inequalities.

A set-valued function $\phi_w : K \rightrightarrows Y$ is said to be a gap function of (WVVI) if and only if (i) $0 \in \phi_w(x^*)$ if and only if x^* solves (WVVI); and (ii) $0 \not\geq_{\hat{C}} \phi_w(x), \forall x \in K$. A set-valued function $\phi : K \rightrightarrows Y$ is said to be a gap function of (VVI) if and only if (i) $0 \in \phi(x^*)$ if and only if x^* solves (VVI); and (ii) $0 \not\geq_{C\setminus\{0\}} \phi(x), x \in K$.

Proposition 3 *Let C be a pointed and convex cone in Y. We have*

(i) The set-valued function $\phi_w(x) := \text{Max}_{\hat{C}}\langle T(x), x - K\rangle$ is a gap function for (WVVI).

(ii) The set-valued function $\phi(x) := \text{Max}_C\langle T(x), x - K\rangle$ is a gap function for (VVI).

The above gap functions are of set-valued nature. Special single-valued gap functions can be constructed in terms of nonlinear scalarization functions. Given a fixed $e \in \hat{C}$ and $a \in Y$, the nonlinear scalarization function is defined by:

$$\xi_{ea}(y) = \min\{t \in \mathbb{R} : y \in a + te - C\}, \quad y \in Y .$$

Proposition 4 *Let $e \in \hat{C}$. Then $x^* \in K$ solves (WVVI) if and only if the non-positive function $g(x) = \min_{y\in K} \xi_{e0}(\langle T(x), y - x\rangle)$ has a zero at x^*.*

In the special case where $Y = \mathbb{R}^\ell$, $C = \mathbb{R}^\ell_+$ and $T(x) = [T_1(x), \cdots, T_\ell(x)]^\top$, the nonlinear scalarization function may be expressed in the following equivalent form:

$$\xi_{ea}(y) = \max_{1\le i\le \ell} \frac{y_i - a_i}{e_i} .$$

Thus $g(x) = \min_{y\in K} \max_{1\le i\le \ell}\{\langle T_i(x), y - x\rangle\}, x \in K$. The value of each $g(x)$ amounts to solving a linear minimax optimization problem.

Next we construct a gap function for a set-valued WVVI.

Let $Y = \mathbb{R}^\ell$, $C = \mathbb{R}^\ell_+$ and $K \subset X$ a compact subset. Assume that $T : K \rightrightarrows L(X, \mathbb{R}^\ell)$ is a set-valued mapping with a compact set $T(x)$ for each x.

Consider the set-valued WVVI with the set-valued mapping T [9], which consists in finding $x^* \in K$, and $\bar{t} \in T(x^*)$ such that

$$\langle \bar{t}, x - x^*\rangle \not\le_{\hat{C}} 0, \quad \forall x \in K . \tag{2}$$

Let $x, y \in K$ and $t \in T(x)$. Denote

$$\langle t, y\rangle = (\langle\langle t, y\rangle\rangle_1, \cdots, \langle\langle t, y\rangle\rangle_\ell) ,$$

i. e., $\langle\langle t, y\rangle\rangle_i$ is the i-th component of $\langle t, y\rangle$, $i = 1, \cdots, \ell$. We define two mappings $\phi_1 : K \times L(X, \mathbb{R}^\ell) \to R$ and $\phi : K \to \mathbb{R}$ as follows

$$\phi_1(x, t) = \min_{y\in K} \max_{1\le i\le \ell} \langle\langle t, y - x\rangle\rangle_i \tag{3}$$

and

$$\phi(x) = \max\{\phi_1(x, t) | t \in T(x)\} . \tag{4}$$

Since K is compact, $\phi_1(x,t)$ is well-defined. If X is a Hausdorff topological vector space, then $g_1(x,t)$ is a lower semi-continuous function in x. Since $T(x)$ is a compact set, $\phi(x)$ is well-defined.

Theorem 5 *$\phi(x)$ defined by (4) is a gap function of the set-valued WVVI.*

By Theorem 5, the solution of set-valued WVVI is equivalent to finding a global solution x^* to the following generalized semi-infinite programming problem

$$\begin{aligned} \max_{x,s} \quad & s \\ \text{s.t.} \quad & \phi_1(x, t) \le s, \quad \forall t \in T(x) , \\ & \phi_1(x, t_1) = s, \quad \exists t_1 \in T(x) , \\ & x \in K . \end{aligned}$$

The concept of vector complementarity problems was introduced in [2,11]. If $K = D$ is a convex cone of X, then, by letting $x = 0$ and $x = 2x^*$ in (WVVI) respectively, we have

$$0 \not\le_{\hat{C}} \langle T(x^*), x^*\rangle \not\le_{\hat{C}} 0 , \tag{5}$$

and by letting $x = y + x^*$ with $y \in D$, we have

$$\langle T(x^*), y\rangle \not\le_{\hat{C}} 0 , \quad \forall y \in D . \tag{6}$$

(5) and (6) together are called a weak vector complementarity problem (WVCP). Let the weak C-dual cone D_C^{w+} of D be defined by

$$D_C^{w+} = \{g \in L(X, Y) : \langle g, x\rangle \not\le_{\hat{C}} 0 , \quad \forall x \in D\} .$$

Then (WVCP) can be rewritten as a problem of finding $x^* \in D$, such that

$$\langle T(x^*), x^*\rangle \not\ge_{\hat{C}} 0 , \quad T(x^*) \in D_C^{w+} .$$

Thus a solution of (WVVI) is one for (WVCP), but the fact that the inverse implication is in general not true can be shown by some simple example. Nevertheless, the inverse implication can be guaranteed by the usual positiveness property on T. Indeed, let the strong C-dual cone D_C^{s+} of D be defined by

$$D_C^{s+} = \{g \in L(X, Y) : \langle g, x\rangle \ge_C 0 , \quad \forall x \in D\} .$$

The positive vector complementarity problem (PVCP) is defined to be a problem of finding an $x^* \in D$ such that

$$\langle T(x^*), x^*\rangle \not\ge_{\hat{C}} 0 , \quad T(x^*) \in D_C^{s+} .$$

It is obvious that D_C^{w+} and D_C^{s+} are nonempty, since the null linear function in $L(X,Y)$ belongs to D_C^{w+} and D_C^{s+}. It is easy to prove that $D_C^{s+} \subset D_C^{w+}$ if C is pointed. When $Y = \mathbb{R}$, the weak and strong C-dual cones of D reduce to the dual cone D^* of D. The weak and strong C-dual cones of D can be shown to be algebraically closed and the strong C-dual cone of D is convex.

Thus, it is clear that if C is pointed, then a solution of (PVCP) is one for (VCP). Moreover, by noting the ordering implication of $0 \leq_C a \not\geq_{\hat{C}} b \Longrightarrow b \not\leq_{\hat{C}} 0$, a solution of (PVCP) is one for (WVVI).

References

1. Chen GY, Huang XX, Yang XQ (2005) Vector optimization. Set-valued and variational analysis. Lecture Notes in Economics and Mathematical Systems, 541. Springer, Berlin
2. Chen GY, Yang XQ (1990) The vector complementary problem and its equivalences with vector minimal element in ordered spaces. J Math Anal Appl 153:136–158
3. Chen GY, Yen ND (1993) On the variational inequality model for network equilibrium. Internal Report, Department of Mathematics, University of Pisa, 3. 196 (724)
4. Fang YP, Huang NJ (2006) Strong vector variational inequalities in Banach spaces. Appl Math Lett 19:362–368
5. Giannessi F (1980) Theorems of alternative, quadratic programs and complementary problems. In: Cottle RW, Giannessi F, Lions JL (eds) Variational Inequality and Complementary Problems. Wiley, New York
6. Giannessi F (1998) On Minty variational principle. In: Giannessi F, Komlósi S, Rapcsák T (eds) New Trends in Mathematical Programming. Kluwer, Boston, pp 93–99
7. Giannessi F (ed) (2000) Vector Variational Inequalities and Vector Equilibrium. Kluwer, Dordrecht, Boston, London
8. Harker PT, Pang JS (1990) Finite-dimensional variational inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications. Math Program 48(2 Ser B):161–220
9. Konnov IV, Yao JC (1997) On the generalized vector variational inequality problem. J Math Anal Appl 206(1):42–58
10. Lee GM, Kim DS, Lee BS, Yen ND (1998) Vector variational inequality as a tool for studying vector optimization problems. Nonlinear Anal 34(5):745–765
11. Yang XQ (1993) Vector complementarity and minimal element problems. J Optim Theory Appl 77(3):483–495
12. Yang XQ (1993) Vector variational inequalities and its duality. Nonlinear Anal, TMA 21:867–877
13. Yang XQ, Goh CJ (1997) On vector variational inequalities: application to vector equilibria. J Optim Theory Appl 95:431–443

Vehicle Routing

JEAN-YVES POTVIN
University Montréal, Montréal, Canada

MSC2000: 90B06

Article Outline

Keywords
Node Routing
- Static Deterministic Problems
- Static Stochastic Problems
- Dynamic Problems
- Methodologies

Arc Routing
- Chinese Postman Problem
- Rural Postman Problem
- Capacitated Arc Routing Problem
- Methodologies

See also
References

Keywords

Network; Node routing; Arc routing; Static; Dynamic; Exact methods; Metaheuristics

Vehicle routing consists in determining optimal collection or delivery routes for a fleet of vehicles on a transportation network [7,11,14,15]. The customers to be serviced may be associated with vertices (*node routing*) or arcs (*arc routing*) of the network. Problems are deterministic or stochastic depending on the certainty or uncertainty associated with the data. They are static or dynamic depending on the time of availability of the data. When all information is known in advance, a solution can be constructed beforehand and the problem is said to be static. Conversely, when new information (e. g., new customer requests) is continuously revealed over time, the problem is said to be dynamic.

In the following, we examine both node and arc routing problems.

Node Routing

These *NP*-hard problems are found in transportation activities where the service occurs at the nodes (customer sites) of the transportation network. Along the logistics chain, they are associated with movement of raw material from suppliers to plants, movement of finished products from plants to warehouses or depots, and delivery of products to final customers. In the service sector, they are found in *dial-a-ride* systems (e. g., transportation-on-demand for people with specific needs), school bus routing, courier services, etc.

The various classes of node routing problems are now presented.

Static Deterministic Problems

This is the most widely studied class of problems in the vehicle routing literature. It may be formally described as follows. Let $G = (V, A)$ be a graph where $V = \{ 1, \ldots, n \}$ is the set of vertices, vertex 1 is the depot and A is the set of arcs. A nonnegative cost c_{ij} is associated with every arc (i, j), $i \neq j$. This cost may be interpreted as a travel distance or travel time, depending on the context. Also, a fleet of m vehicles is based at the depot with m being either fixed or variable. The *vehicle routing problem* (VRP) then consists in determining a set of least-cost vehicle routes such that:

- each vertex, apart from vertex 1, is visited exactly once by exactly one vehicle;
- all vehicle routes start and end at vertex 1;
- side constraints may have to be satisfied.

When no side constraints are found, an *m-traveling salesman problem* (m-TSP) is obtained, with the classical TSP corresponding to the special case $m = 1$ [17]. When side constraints are present, there may be one or more of the followings:

- capacity constraints (*capacitated vehicle routing problem*, CVRP): a nonnegative demand or load q_i is associated with each vertex i (apart from vertex 1) and the total load on a route cannot exceed vehicle capacity Q;
- distance or travel time constraints (*distance-constrained vehicle routing problem*, DVRP): the length or travel time of a route must not exceed a prespecified bound;
- time windows (*vehicle routing problem with time windows*, VRPTW): each vertex i must be visited within a time interval $[a_i, b_i]$; the upper bound b_i may be soft or hard, and a waiting time is typically allowed if the vehicle arrives before a_i;
- precedence constraints: a partial ordering may be imposed on the sequence of vertices to be serviced. For example, a subset of vertices may have to be visited before another subset, as in the *vehicle routing problem with backhauls* (VRPB). There may also be a precedence relationship between pairs of vertices, as in the subscriber dial-a-ride problem (for transportation-on-demand services), where each transportation request includes both a pick-up and a delivery point and the pick-up must precede the delivery.

Time-constrained vehicle routing and scheduling problems have been widely studied in the literature (see [4], for a comprehensive survey). *Pick-up and delivery* problems are studied in [20]. Other, more complex, variants are also reported in the literature. Without being exhaustive, we mention the following [7]:

- *multiple depot* problems where vehicles may start (end) their route from (to) different depots;
- *mixed fleet* problems where vehicles have different characteristics (e. g., different capacities);
- *location-routing* problems where strategic decisions about the location of different facilities (e. g., depots, warehouses) must be taken concurrently with the determination of the delivery routes;
- *period routing* where daily routes are determined over an horizon that spans a few days (e. g., a 5-day week) and where each customer must receive deliveries at a designated frequency;
- *inventory routing* where each customer has an inventory of a product and a distributor should determine delivery routes so that no customer runs out of the product.

Static Stochastic Problems

Stochastic vehicle routing problems typically involve a stochastic demand or stochastic customers [16]. These problems also belong to the class of static problems since all information is known in advance, even if this information is a probability distribution. Furthermore, recourse actions are predefined in case of failure (e. g., when the vehicle capacity is exceeded or when a customer does not show). Hence, a solution that minimizes some expected measure, like expected total travel distance, can be constructed beforehand.

Dynamic Problems

Dynamic problems emerge when information about the problem is continuously revealed over time [18]. Usually, such problems occur when new customer requests must be dispatched in 'real-time' into the current routes. These problems are found in many different application domains, like delivery of petroleum products and industrial gases, truckload and less-than-truckload

trucking, dial-a-ride systems, courier services, emergency services, etc. [10] Dynamic problems exhibit distinctive features with regard to their static counterpart. For one thing, the time issue is crucial as the system must react promptly to new occurring events. A discussion on this topic may be found in [19].

Methodologies

Exact methods for solving vehicle routing problems include [14,21]:

- direct tree search methods (e. g., branch and bound);
- dynamic programming;
- integer linear programming (e. g., set partitioning and column generation).

Approximate methods may be classified as [2]:

- *constructive methods*;
  - cluster-first, route-second: clusters or groups of customers are first identified; then, these customers are sequenced within each route;
  - route-first, cluster-second: a large route that includes all customers is first constructed; then, it is partitioned into a number of smaller feasible routes;
  - savings/insertion methods;
- *improvement methods* based on exchange procedures;
- *mixed methods* which include both a constructive and an improvement phase. The latter include *metaheuristics* like tabu search, simulated annealing, genetic algorithms, GRASP, ant systems and hybrids [8,9,12].

Parallel implementations of the above methods, in particular metaheuristics, are also reported in the literature [3]. Exploitation of multiple processors in parallel is vital in the case of dynamic vehicle routing problems where fast response times are required.

Arc Routing

As opposed to node routing, the key service activity in arc routing problems is to cover arcs of a transportation network [1,5,6]. These problems are found in real-world applications like street maintenance, garbage collection, snow plowing, meter reading, etc. In the following, we examine the various subclasses of problems.

Chinese Postman Problem

The Chinese postman problem (CPP) is the canonical problem in arc routing [13]. It is defined as follows. Let $G = (V, E \cup A)$ be a graph where $V = \{1, \ldots, n\}$ is the set of vertices, E is a set of undirected edges and A is a set of directed arcs. A nonnegative cost c_{ij} is associated with every edge or arc (i, j), $i \neq j$. The CPP then consists of determining a least cost traversal of all edges and arcs of G. Several special cases should be mentioned:

- the undirected CPP, when $A = \emptyset$;
- the directed CPP, when $E = \emptyset$;
- the mixed CPP, when $A \neq \emptyset$ and $E \neq \emptyset$;
- the windy CPP, when $A = \emptyset$ but the cost of travel on each edge is not the same in both directions;
- the hierarchical CPP, when $A \cup E$ is partitioned into several classes and a precedence relationship is defined among the classes. If a particular class C_i precedes another class C_j, then all edges in C_i must be visited before C_j.

Rural Postman Problem

In the rural postman problem (RPP), only a subset of $E \cup A$ is required to be serviced, although other edges or arcs may be in the solution. As for the CPP, different variants may be considered, depending if the underlying graph is directed, undirected or mixed.

Capacitated Arc Routing Problem

In the capacitated arc routing problem (CARP), a nonnegative quantity q_{ij} is associated with each arc or edge (i, j). A fleet of m vehicles, each of capacity Q, must visit all edges or arcs of the graph subject to the capacity constraint.

Methodologies

Polynomial algorithms have been developed for the undirected and directed variants of CPP. The other variants are *NP*-hard, although polynomially solvable cases have been identified. For the undirected CPP, the problem solving methods are based on the following observation: a cycle that contains each edge exactly once in a graph can be found if and only if all vertices have even degree (such a graph is said to be *Eulerian* or *unicursal*). Thus, the basic problem is to find a least cost way of adding edges to the graph to make it uni-

cursal. This is done by calculating the shortest paths between odd-degree vertices and to use these costs to determine a least-cost matching of the odd-degree vertices. The same augmentation problem appears in the directed CPP, in terms of balancing the in-degree and out-degree of each vertex. This is solved through a minimum cost network flow problem. Similar approaches are reported for the RPP although, in this case, approximate solutions are produced.

Many heuristic methods have been proposed for the CARP. These methods are often derived from their VRP counterparts: insertion methods, constructive methods, improvement methods and mixed methods [6].

Exact algorithms based on branch and bound techniques are also reported in the literature for solving some *NP*-hard variants [1,5,6].

See also

- ► General Routing Problem
- ► Stochastic Vehicle Routing Problems
- ► Vehicle Scheduling

References

1. Assad AA, Golden BL (1995) Arc routing methods and applications. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing. North-Holland, Amsterdam, pp 375–483
2. Bodin L, Golden BL, Assad AA, Ball M (1983) Routing and scheduling of vehicles and crews: The state of the art. Comput Oper Res 10:63–211
3. Crainic TG, Toulouse M (1998) Parallel metaheuristics. In: Crainic TG, Laporte G (eds) Fleet Management and Logistics. Kluwer, Dordrecht
4. Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995) Time constrained routing and scheduling. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing. North-Holland, Amsterdam, pp 35–139
5. Eiselt HA, Gendreau M, Laporte G (1995) Arc routing problems, Part I: The Chinese postman problem. Oper Res 43:231–242
6. Eiselt HA, Gendreau M, Laporte G (1995) Arc routing problems, Part II: The rural postman problem. Oper Res 43:399–414
7. Fisher M (1995) Vehicle routing. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing. North-Holland, Amsterdam, pp 1–33
8. Gendreau M, Laporte G, Potvin JY (1997) Vehicle routing: Modern heuristics. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, New York, pp 311–336
9. Gendreau M, Laporte G, Potvin JY (1998) Metaheuristics for the vehicle routing problem. Techn Report G-98-52 Groupe d'études et de recherche en analyse des décisions, Études Commerciales, Montréal
10. Gendreau M, Potvin JY (1998) Dynamic vehicle routing and dispatching. In: Crainic TG, Laporte G (eds) Fleet Management and Logistics. Kluwer, Dordrecht, pp 115–226
11. Golden BL, Assad AA (eds) (1988) Vehicle routing: Methods and studies. North-Holland, Amsterdam
12. Golden BL, Wasil EA, Kelly JP, Chao I-M (1998) Metaheuristics in vehicle routing. In: Crainic TG, Laporte G (eds) Fleet Management and Logistics. Kluwer, Dordrecht, pp 33–56
13. Guan M (1962) Graphic programming using odd and even points. Chinese Math 1:273–277
14. Laporte G (1992) The vehicle routing problem: An overview of exact and approximate algorithms. Europ J Oper Res 59:345–358
15. Laporte G (1997) Vehicle routing. In: Dell'Amico M, Maffioli F, Martello S (eds) Annotated Bibliographies in Combinatorial Optimization. Wiley, New York, pp 223–240
16. Laporte G, Louveaux F (1998) Solving stochastic routing problems. In: Crainic TG, Laporte G (eds) Fleet Management and Logistics. Kluwer, Dordrecht, pp 159–167
17. Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (eds) (1985) The traveling salesman problem: A guided tour of combinatorial optimization. Wiley, New York
18. Powell WB, Jaillet P, Odoni A (1995) Stochastic and dynamic networks and routing. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing. North-Holland, Amsterdam, pp 141–295
19. Psaraftis HN (1995) Dynamic vehicle routing: Status and prospects. Ann Oper Res 61:143–164
20. Savelsbergh MWP, Sol M (1995) The general pickup and delivery problem. Transport Sci 29:17–29
21. Toth P, Vigo M (1998) Exact solution of the vehicle routing problem. In: Crainic TG, Laporte G (eds) Fleet Management and Logistics. Kluwer, Dordrecht, pp 1–31

Vehicle Routing Problem with Simultaneous Pickups and Deliveries

ARIF VOLKAN VURAL, BURAK EKSIOGLU
Department of Industrial and Systems Engineering, Mississippi State University, Mississippi State, USA

MSC2000: 00-02, 01-02, 03-02

Article Outline

Introduction
Formulation
Applications
References

Introduction

Since it was first formulated in 1959 by Dantzig and Ramser [5], the vehicle routing problem (VRP) has attracted much attention from operations research academia. The VRP is similar to the traveling salesman problem (TSP). In the TSP, the goal is to find the shortest trip for a salesman who has to visit all customers (nodes) exactly once, starting at an initial node and returning to the same initial node. The VRP is the name given to the class of problems which includes the TSP and adds usage of multiple vehicles, each with a uniform limited capacity.

The VRP designates a wide range of set-ups to the original problem of the TSP rather than addressing a specific problem. Several versions of the problem may be defined, depending on a number of factors, constraints, and objectives addressed in the context of the problem. It is fairly important to clarify the borders and content of the problem prior to developing an analytical approach towards a solution. The main attributes within the configuration of most VRPs published are listed as follows:

- *Number of vehicles*: The upper limit on number of vehicles available for routing.
- *Vehicles' homogeneity/heterogeneity*: The condition on the vehicles' capacity, whether it is uniform for all vehicles or not.
- *Time windows*: The imposed time constraint for servicing a customer.
- *Backhauls*: Besides feeding a customer with its demand, the customer loads the truck with some load to be carried back to the depot.
- *Splitting/unsplitting of load*: The load to be delivered or picked up at any node may be divided into any number of groups in the splitting case and this is strictly forbidden in the unsplitting case. Splitting puts forward multiple trips to any node rather than a single one during the routing process.
- *Single depot/multiple depots*: The distribution or collection process is constructed considering a single depot or multiple ones; even distribution and collection centers may be different.
- *Static/dynamic service needs*: The demand values are either known in whole or unknown to some level prior to establishing a route for the service vehicle.
- *Precedence/coupling constraints*: If a node's demand must be satisfied with anything picked up at a node other than the depot, then a coupling constraint is used to handle this. The latter node has precedence in service over the prior node (precedence constraint).

The VRP with deliveries and pickup (VRPDP) is a subset of the general VRP such that rather than providing either a delivery or a pickup service, the nodes provide both services, sometimes even simultaneously. Salhi and Nagy [15] gave a clear identification of three types of problem that may be addressed under the VRPDP subset depending on the the service provided as follows:

1. VRP with backhauls (VRPB): The nodes are identified either as linehaul (the nodes with deliveries originating from the depot) or backhaul (nodes with items to be picked up and destined for the main depot) nodes. The linehaul nodes are served prior to the backhaul nodes. The logic behind this is given by Chen and Wu [4] as the design of old trucks only allowed rear-load functions.
2. VRP with mixed load (VRPM): Upon introduction of a side-loading function on trucks, rearrangement of loads on board became easier; thus, the need for serving linehaul customers to free some space was no longer necessary. The result was the introduction of service routes with backhaul nodes at any sequence before the last linehaul node. Establishing routes composed of a mixed sequence of linehaul and backhaul nodes constitutes the VRPM.
3. VRP with simultaneous delivery and pickup (VRPSDP): The VRPM setting caused inconvenience when some nodes requested both a pickup and a delivery service. Those nodes had to be visited twice, which added to the tour length, resulting in inferior total routing results. The VRPSDP setting lets each node have a delivery and pickup service during the same stop. The delivery operation precedes the pickup. The VRPM is a special case of VRPSDP such that either the delivery or the pickup quantity at each node is defined as zero.

The VRPSDP model differs from the *m-dial-a-ride problem with nonunit capacity* in the sense that the traffic of goods between nodes other than the depot is strictly avoided. In the latter type of problem, m routes are created using multiunit capacity vehicles to accommodate transportation of items from and to nodes existing on a route.

Montane and Galvao [10] introduces a fourth different type of VRPDP. The problem, which is referred to as the *express delivery problem*, constructs separate delivery routes and pickup routes to serve customers with both pickup and delivery requirements. These delivery routes, which are traversed initially, do not necessarily cover the same nodes while vehicles move along the pickup routes. Thus, each node is visited twice, possibly by two different vehicles, to satisfy the two types of service.

In this review, only the VRPSDP configuration is discussed. The problem comprises many customers or "nodes" to be served by a fleet of vehicles of homogeneous type and limited capacity. The vehicles deliver items to customers from the depot and pickup loads are collected to be delivered back to the depot at the end of the trip. The unit sizes of the picked-up and delivered items are identical and they consume the same amount of capacity on each truck. However, the amounts picked up and delivered at each location may not necessarily be the same. Delivery and pickup locations are unique and feeding a customer with anything picked up at a node other than the main depot is strictly avoided. The objective is to minimize the total distance covered by the fleet during service. Some instances of this type of problem may be observed in distribution networks of bottled spring water in recollectable containers, industrial gas distribution/collection in refillable tanks, liquefied petroleum gas distribution in commercial containers from wholesalers to retailers, and crew transportation between mainland and offshore oil rigs using helicopters.

Formulation

The graph-theoretical definition of the VRPSDP is as follows.

Instance: A graph $G = (V, E)$ with edge weights w_e for all $e \in E$ and vertex weights d_v and p_v for all $v \in V$, a distinguished node, i. e., depot d, and a parameter k denoting the upper limit on the number of vehicles available, and a parameter C denoting uniform capacity of each of the trucks.

Objective: Find a partition of the nodes in $V\backslash\{d\}$ to $V_1, \ldots, V_k$ and a subset of edges $T_k \subseteq E$ forming k tours each containing node d and each node of V_i exactly once, so that $\sum_{e \in Tk} w_e$ is minimized without violating $\sum_{j \in Vh} d_j \leq C$, $\sum_{j \in Vh} p_j \leq C$ for $h \in \{1, \ldots, k\}$ and $p^*_{vt} + d^*_{vt} + p_v \leq C$ for $v \in V$, $t \in \{1, \ldots, k\}$, where $p_{vt}{}^*$ denotes all the load picked up at some partition V_t prior to some definite node $v \in V_t$; and $d_{vt}{}^*$ denotes all the load to be delivered at some partition V_t after some definite node $v \in V_t$.

Applications

Very little attention has been paid to the VRPSDP. This problem was first introduced in the literature by Min [9]. In his work, Min studied book distribution and recollection activities between a central library and 22 remote libraries in a county in Ohio. Each and every day, a central depot is responsible for supplying remote libraries with ordered books and recollecting previously delivered books from them in return. There are two trucks, which are assigned for this distribution and recollection activity, with limited capacity. The article also provides a symmetric cost matrix in terms of distances between the library locations. Min [9] gives the solution for his problem as 94. He uses a method of clustering the nodes into two groups, then solving a relaxation of the problem using branch and bound. After determining the constraint violations, he penalizes moves leading to such violations and solves the relaxed form of the problem iteratively until no violations of the constraints are observed.

Halse [8] studied this special case VRPSDP as well as many others in the VRP literature. In his work, cases with a single depot and multiple vehicles and number of nodes varying between 22 and 150 are studied. Halse [8] utilized a Lagrangian relaxation and a column-generating approach. A cluster-first-route-second type heuristic is developed in which nodes are first distributed to vehicles and then the problem is solved using a 3-opt approach.

Angelelli and Mansini [1] studied the VRPSDP with time window constraints. They implemented a branch-and-price approach based on a set-covering formulation for the master problem. A relaxation of the elementary shortest path problem with time windows and capacity constraints was used as the pricing problem. Branch-and-bound was applied to obtain integer solutions. Angelelli and Mansini [1] provided further profound guidance about exact algorithms based on column generation and branch-and-price algorithms.

Gendreau et al. [7] studied the VRPSDP for a single vehicle case. They derived 26 problem instances based on some formerly published instances and they tested the performance of their two newly developed heuristics with those previously introduced heuristics in the VRP literature. In their problem instances, the number of nodes varied between six and 261, including the depot. The first algorithm presented by the authors constructs a sequence and serves nodes with positive demands (they describe positive demand as the case when the pickup quantity is greater than the delivery quantity) until the truck capacity is violated. In other words, when the truck is handling the next customer with positive demand, if the residual capacity is not enough they stop serving the customer and begin serving the next available customer with negative demand. When there is enough room available to serve the next customer, the truck returns to the node where the former capacity violation occurred, and the next node with a positive demand is served.

The main difficulty associated with Lagrangian relaxation is due to the cardinality of the relaxed constraints, which does not allow for the explicit inclusion of all of them in the objective function. To overcome this difficulty Toth and Vigo [17] proposed including only a limited set of the relaxed constraints initially and iteratively added other constraints which are violated by the current solution of the Lagrangian problem. Besides this mechanism, to avoid complexity of the objective function, they also proposed purging the relaxed constraints from the Lagrangian relaxation in case they become slack by the current solution. This process is repeated until no violated constraints are detected (hence, feasibility is obtained) or a prefixed number of subgradient iterations have been executed.

Dethloff [6] also studied the VRPSDP problem. In his study, Dethloff utilized dynamic programming to calculate net savings attainable by imbedding the future steps and the course of actions to follow during those steps. The aim is to keep higher residual capacities on the vehicles to provide higher freedom for future servings of nodes while dealing with a current node. Higher residual capacities can be achieved by serving customers with a small (large) delivery amount and a large (small) pickup amount late (early) in the route. Each of those residuals is more advantageous if it is valid for a long part of the route. Additionally, the residual values are prospectively more advantageous if a higher cumulative demand for delivery and pickup of the yet unvisited customers for future insertions exists. In his work, Dethloff developed 40 VRPSDP instances to test his algorithm. He also reported an improvement on the solution given by Min [9]. Then, he compared the results of his algorithm with those of Salhi and Nagy [15], based on their problem instances and problem structure. In the problems given in [15], nodes are separated into disjoint delivery or pickup nodes with 0 distance vector in between and they are provided with either a delivery or a pickup service, but not both at the same time. Thus, a node may be visited more than once when the coupling of nodes in the solution is collapsed into single ones. Besides, the problem puts a limit on the maximum route length and introduces multiple depots rather than a single depot case.

The mathematical formulation of the VRPSDP is omitted. The interested reader is referred to [6]. A relaxation of the VRPSDP may be obtained by separating pickup and delivery processes such that at any node either pickup or a delivery occurs. This relaxation has been commented on to be at least as hard as an NP-hard problem [12]. Thus, the VRPSDP is also NP-hard in the strong sense.

The algorithm presented by Montane and Galvao [10] for the VRPSDP starts with two well-known heuristics: tour partitioning and sweep. The primal problem is divided into TSP with simultaneous deliver and pickup subproblems and those are solved using cycle, minimum spanning tree, and cheapest insertion heuristics. Node exchange operators are used to overcome route infeasibilities and improve solution quality. With use of the proposed methods, eight heuristics were generated and tested on 27 problems. The number of nodes in the problems ranged between 32 and 80.

Minimum, maximum, and average solution values were provided rather than the best results for each problem instance.

Vural [18] proposed a genetic algorithm for the solution of the VRPSDP described elaborately above. In that work, some previously published genetic structures and mechanisms were used to build a good performing mechanism. The Dethloff [6] instances were tested and the results were found to be relatively better. The genetic mechanism introduced uses the "*random keys method*" of Bean [2] in order to establish the initial population of the chromosomes and uses a modified version of a cross-over mechanism introduced by Topcuoglu and Sevilmis [16]. With these two mechanisms, computational efficiency and reduction of complexity is sought. Further route improvement is performed using a local search mechanism based on Or-opt [14].

Nagy and Salhi [13] revisited their previous research and extended as well as updated it. They studied the VRPSDP together with the VRPDP, again considering both single and multiple depot instances. They provided a list of VRPDP articles with their main features. They improved their previous [13] constructive heuristics by adding more node operators, leading to better solution refinement. They introduced three new heuristic methods and compared their performance with that of three other methods in addition to their best performing method of a previous study.

Chen and Wu [4] provided a recent study on the VRPSDP. They developed two algorithms; one is an insertion-based heuristic, which also provides the initial solution for the second algorithm; the second algorithm is a hybrid metaheuristic which works like the "*simulated annealing*" method, but eliminates the probabilistic moves with a deterministic rule. The algorithm also employs a "*tabu*" mechanism to avoid recurrence of previous local optima and finally refines the solutions for any improvement using a node swap and exchange operators. The algorithm runs as long as an improvement is realized within a specific number of trials. The algorithm was run on 14 Nagy and Salhi [13] instances without an upper limit on route length. They claim better results over Salhi and Nagy's [15] results but they provided no comparison either with improved results [13] or with those provided by Dethloff [6]. They further generated some problem instances by modifying Solomon instances using the method of Salhi and Nagy [15].

The study presented by Montane and Galvao [11] was inspired by transportation between mainland Brazil and open-sea oil platforms using helicopters. Since the distance a helicopter may fly between two spots is restricted by factors such as fuel needs, the original VRPSDP comes with an additional constraint on the maximum length of move between nodes. Swapping people between platforms is avoided, which makes this problem a fine VRPSDP instance from a real-life situation. Montane and Galvao [11] used modified versions of sweep and tour partitioning heuristics. The tours are filled with nodes or closed depending on the net change of the total load and maximum distance constraint. For this phase, four different selection rules are devised. The initial solutions constitute an input to the tabu search mechanism. The tabu search stops either when no more feasible movement exists or when an upper limit on a number of iterations is met. The procedure creates new initial solutions and the tabu phase follows until this cycle is run for a fixed number of times. The authors tested their problem on 87 instances, provided by Dethloff [6], Salhi and Nagy [15], and Min [9], and 18 newly modified Solomon and extended Solomon instances from the literature.

Bianchessi and Righini [3] applied local search and tabu search algorithms on selfcreated random instances for VRPM. They further applied their algorithms on Dethloff [6] instances and compared their average values with those of Dethloff, reporting an improvement. However, they did not compare their results with those provided by previous researchers for the same instances. They first constructed initial solutions using four different node selection rules, based on tolerance to capacity violations and overall tour feasibility. They further applied local search by node exchanges on different neighborhoods they defined. Although they applied a variable neighborhood search technique, they did not use it for VRPSDP solution generation.

References

1. Angelelli E, Mansini R (2002) The vehicle routing problem with time windows and simultaneous pick-up and delivery, Quantitative Approaches to Distribution Logistics and Supply Chain Management. Series: Lecture Notes in

Economics and Mathematical Systems. Springer, pp 249–267
2. Bean JC (1994) Genetic algorithms and random keys for sequencing and optimization. ORSA J Comput 6(2):154–160
3. Bianchessi N, Righini G (2007) Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. Comput Oper Res 34:578–594
4. Chen JF, Wu TH (2006) Vehicle routing problem with simultaneous deliveries and pickups. J Oper Res Soc 57:579–587
5. Dantzig GB, Ramser JH (1959) The truck dispatching problem. Manag Sci 6(1):58–64
6. Dethloff J (2001) Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. OR Spektrum 23:79–96
7. Gendrau M, Laporte G, Vigo D (1999) Heuristics for the traveling salesman problem with pick-up and delivery delay. Comput Oper Res 26:699–714
8. Halse K (1992) Modeling and solving complex vehicle routing problems. PhD Thesis, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, Lyngby
9. Min H (1989) The multiple vehicle routing problem with simultaneous delivery and pick-up points. Transp Res 23-A:377–386
10. Montane FAT, Galvao RD (2002) Vehicle routing problems with simultaneous pick-up and delivery service. OPSEARCH 39(1):19–33
11. Montane FAT, Galvao RD (2006) A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. Comput Oper Res 33:595–619
12. Mosheiov G (1998) Vehicle routing with pick up and delivery: Tour partitioning heuristics. Comput Indust Eng 34(3):669–684
13. Nagy G, Salhi S (2005) Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. Eur J Oper Res 162:126–141
14. Or I (1976) Traveling salesman-type combinatorial optimization problems and their relation to the logistics of blood banking. Phd. Thesis, Department of Industrial Engineering and Management Science, Northwestern University, Evanston
15. Salhi S, Nagy G (1999) A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. J Oper Res Soc 50:1034–1042
16. Topcuoglu H, Sevilmis C (2002) Task scheduling with conflicting objectives Lecture Notes on Computer Science, vol 2457. Springer, Heidelberg, pp 346–355
17. Toth P, Vigo D (2002) Models, relaxations and exact approaches for the capacitated vehicle routing problem. Discr Appl Math 23:487–512
18. Vural AV (2003) A GA Based Meta-heuristic for Capacitated Vehicle Routing Problem with Simultaneous Pick-up and Deliveries, MSc. Thesis. Graduate School of Engineering and Natural Sciences, Sabanci University, Istanbul

Vehicle Scheduling

JOACHIM R. DADUNA
Faculty of Business and Economics,
Fachhochschule fur Wirtschaft (FHW) Berlin,
Berlin, Germany

MSC2000: 90B06, 90B35, 68M20, 90C27, 90B80, 90B10, 90C10

Article Outline

Keywords
See also
References

Keywords

Transportation; Logistics; Scheduling theory; Combinatorial optimization; Discrete location and assignment; Flows in networks; Deterministic

The *vehicle scheduling problem* (VSP) is concerned with the combination of a number of (passenger) *trips*, which are given by a timetable, to (vehicle) *blocks*. These blocks are sequences of trips operated by one vehicle after leaving the (origin or) home *depot* with a *pull-out trip* until its return to the same depot with an *pull-in trip*. For public transit companies the solution of the VSP is of great importance for the efficiency in planning processes. This is valid for both vehicle operations as well as for manpower planning. From the maximal number of blocks (which usually arises during the morning peak hours) one obtains a lower bound on the vehicles required, consequently, this bound strongly influences the fleet size a company has to maintain. Furthermore, the number of vehicles and the total duration of all blocks also determine considerably the manpower requirements.

The basic problem is to assign each trip of a given timetable to exactly one block in accordance with the *in-company policies* and various *operational restrictions*, while a *minimization problem* has to be solved with a given objective function [1,9]. For real-world problems, the restrictions which have to be considered are usually varying for each company, and they also lead in

many cases to a great complexity in the problem structure. That is, these optimization problems tend to be *NP*-hard optimization problems (cf. ► Complexity theory; ► Complexity classes in optimization).

A range of basic characteristics can be determined, which mainly specify the (company-dependent) problem structures of a VSP [5,6,18].

- Number of depots. *Single-depot vehicle scheduling problems* (SDVSP) and *multi-depot vehicle scheduling problems* (MDVSP) have to be differentiated. However, a MDVSP arises only if there are intersections between the service areas assigned to the depots. Otherwise, a certain number of independent SDVSP has to be solved. Making use of a *cluster first-schedule second strategy* a MDVSP can also be partitioned into a number of SDVSP.
- Assignment of vehicles to a depot. Each vehicle is usually assigned to a specific depot. Therefore, after finishing a block the vehicle has to return to its origin. However, in some cases exceptions may be allowed by interchanges between different depots.
- Number of trips. The number of (passenger) trips is fixed by a given timetable. In some cases alterations (trip shortening, trip shifting, trip cancellation) are possible or necessary to attain a reduction in the necessary number of vehicles or to adhere to capacity constraints.
- Assignment of lines/trips to depots. In multi-depot cases, the lines and/or (single) trips have to be assigned to the depots, preferably in accordance to the spatial structure. If no overlaps occur, a number of SDVSP is generated, otherwise a MDVSP has to be solved.
- Multiple types of vehicles and type-dependent assignment of lines/trips. Various restrictions (e. g., demand structure, differentiation on service, technical conditions) may necessitate type-dependent assignments of lines/trips.
- Type-independent and type-dependent capacity constraints. At the depots certain restrictions usually arise resulting from the number of stationed vehicles. However, based on the planning results, in some cases interchanges between different types may be allowed.
- Technical restrictions. For those means of transport, which are operating in a guided or tracked mode (trolley bus, tram, light rail, etc.), additional technical restrictions have to be considered. These specific restrictions mainly result from the reduced room for passing.
- Manpower capacity restrictions. Besides quantitative manpower restrictions, the qualification of the drivers also has to be taken into consideration. Of special interest are the varying knowledge levels regarding line networks, which often occurs in greater public transit areas. In addition, in many cases the drivers are not instructed on each type of vehicle.
- In-company restrictions. Here the different *layovers* (which are defined as a minimum time between the end of a trip i and the beginning of the following trip j carried out by the same vehicle) are considered. The duration of a layover results from the attributes describing the types of trips, which have to be linked. Furthermore, other specific in-company targets can be included, as e. g., *interlining* (within one block there should not be trips of only one line) or a given range for the length of a block.

This short overview clearly shows the difficulties to describe a real-world situation in a formal model, which may be taken as a necessary basis to employ quantitative methods on these scheduling problems. The availability of efficient algorithms usually depends on the formulation of a model with explicitly or implicitly included constraints. However, in some cases a simplification of the description can be considered which leads to significant impacts on the complexity of the formal model [9,16].

Besides the in-company framework, the operational restrictions have to be considered, which describe the basis for the admissibility to link two trips i and j within a block. To determine whether a link is admissible, the following conditions based on the fundamentals of the (standard) VSP have to be met:

$$s_i + d_i + \Psi_i + \Delta_{i(j)} + \delta_{ij} + \Delta_{j(i)} \leq s_j, \tag{1}$$

$$s_j - (s_i + d_i) \leq T_{\max}, \tag{2}$$

with Ψ_i as a buffer time to cover possible delays, δ_{ij} as the running time of a possibly required deadhead trip, Δ_{ij} or Δ_{ji} as a (trip dependent) layover, and $T_{\max}$ as the (given) maximum *idle time* between the end of a trip i (expressed as the starting time s_i of a trip i plus its duration d_i) and the departure of the following trip j.

In the modeling process for all admissible links, an (individual) weight has to be defined, based on the quantifiable idle time as well as the various nonquantifiable in-company restrictions. From the planning staff's point of view, the valuation of idle times in many cases leads to a conflicting situation. To attain efficient vehicle schedules the idle time usually should be minimized, but considering the *duty scheduling process*, longer idle times may be favorable for the determination of the necessary breaks in a duty. Therefore, it has to be decided whether the vehicle scheduling process should be regarded first of all as an independent step in the planning process, which is followed by a (separate) duty scheduling process [16], or as an integrated process of simultaneous vehicle and duty scheduling [25,48].

Based on these formal restrictions and on different in-company targets, usually two basic objectives can be formulated:

- minimization of the number of operating vehicles;
- minimization of the nonproductive idle times (e. g., layovers or running time of deadhead trips).

These two objectives are only complementary in a small number of specific cases [40]. Usually, a determination of a lexicographic order of these objectives is necessary. As the needed fleet size has most important influence on operational cost, as mentioned above, the main objective consequently has to be the minimization of the number of vehicles, which is necessary to operate a given timetable, especially during peak hours.

Based on these various and complex in-company structures, different models of the VSP have been formulated and appropriate solution procedures are developed [15,21,22,46,49,50]. The most important of these models is described in the following. The objective for these problems is, if another formulation is not explicitly given, to minimize the number of vehicles needed to operate a given timetable or to minimize the overall operational cost.

- *Single-depot vehicle scheduling problem* (SDVSP). In the SDVSP, all trips are operated by vehicles stationed at a single depot. The first solution procedures for these problems are based on *assignment models* [38,41] and *transportation models* [28,29]. Later, network flow formulations [9,34,35], matching formulations [2,3] and quasi-assignment formulations [42,43] are used.
- *Vehicle scheduling problems with a fixed number of vehicles* (*p*-VSP). The *p*-VSP appears in three different cases [18]. First, it exists in a two-phase solution process of a SDVSP, where based on the optimal number of vehicles, the overall operational cost has to be minimized. In the second case, there are more vehicles available than the calculated optimal number. Therefore, an additional target that all vehicles have to be used, has to be considered. The third case deals with the problem that the available fleet is not large enough to operate a given timetable, so that a certain number of trips cannot be performed. In the first two cases, the *p*-VSP can be solved based on a transportation model, a *network flow model*, a *quasi-assignment model*, and a *matching model*, whereas the third case can only be handled making use of a quasi-assignment model.
- *Multi-depot vehicle scheduling problem* (MDVSP). In the MDVSP the trips have to be operated by vehicles stationed at a certain number of depots with given vehicle and manpower capacities. Each used vehicle has to start and end at its home depot. Based on defined *depot groups*, which represent *virtual depots* showing a unique allocation of lines and/or trips, different solution approaches are developed. The earliest efficient version, a two-stage *schedule first-cluster second* solution procedure, which used an assignment model, is described in [16,38]. However, most of the known approaches proceed from a *multicommodity flow* formulation [34]. To solve the MDVSP based on this formulation, various optimization strategies are employed, while especially *branch and bound* [8,24,37,45], *set partitioning* [4,23,33,34,45], and *Lagrange relaxation* [3,30,32,34,36], methods have to be mentioned.
- *Vehicle scheduling with trip shifting* (TSVSP). As opposed to classical optimization approaches, in the TSVSP input data is modified within the solution process. The main advantage of such strategy lies in an extension of the degree of freedom for the combinatorial process, which leads to better results, especially with respect to the minimization of the number of vehicles during peak hour. Various approaches are described in [7,11,12,14,17,20,31,47].
- *Vehicle scheduling problems with multiple types of vehicles* (MTVSP). The MTVSP considers that

a varying demand on the lines and/or on (single) trips do not necessarily require a unique type of vehicle. Therefore, if a fleet with different vehicle capacities is available, the VSP may be extended. In this case the objective is to determine a schedule that minimizes the operational cost while the assignment of vehicles to trips is not fixed but part of the optimization process. Solution procedures to solve a MTVSP are given in [10,13].

- *Integrated vehicle and duty scheduling problems* (VDSP). In the VDSP, two steps of the planning process, the vehicle scheduling and the duty scheduling, become integrated and solved simultaneously. These problems, which arise mainly in *extra-urban transit planning*, result in solutions where each block usually corresponds to exactly one duty. Solution methods for this problem are described in [2,25,27,39,44,48]. Similar to the VDSP are *vehicle scheduling problems with time constraints* (TCVSP) [26]. In this case, a time constraint may arise from technical restrictions (fuel capacity, etc.) or legal and in-company restrictions such as the maximum length of a duty period.

The described picture shows the basics of the VSP and some specific characteristics, especially with respect to the various in-company dependent problem structures. Beginning several decades ago with very simple solution procedures, which were a result of restrictions in computer technology and in availability of algorithms, better and better results in vehicle scheduling are being attained. Until now, the procedure proposed in [16,19,38] leads to the best results for real-world problems with a great number of trips. However, based on network flow formulations in connection with branch and bound, set partitioning, and Lagrange relaxation methods, some improvements seem to be possible [34]. Furthermore, further research activities are focused on the TSVSP and also on the VDSP.

See also

- ► Airline Optimization
- ► General Routing Problem
- ► Integer Programming
- ► Job-shop Scheduling Problem
- ► MINLP: Design and Scheduling of Batch Processes
- ► Stochastic Scheduling
- ► Stochastic Vehicle Routing Problems
- ► Vehicle Routing

References

1. Assad A, Ball MO, Bodin L, Golden B (1993) Routing and scheduling of vehicles and crews. Comput Oper Res 10:63–211
2. Ball MO, Bodin L, Dial R (1983) A matching based heuristic for scheduling mass transit crews and vehicles. Transport Sci 17:4–31
3. Bertossi A, Carraresi P, Gallo G (1987) On some matching problems arising in vehicle scheduling. Networks 17:271–281
4. Bianco L, Mingozzi A, Riccardelli S (1994) A set partitioning approach to the multi-depot vehicle scheduling problem. Optim Problems Softw 3:163–194
5. Bodin L, Gloden B, Assad A, Ball MO (1983) Routing and scheduling of vehicles and crews: The state of the art. Comput Oper Res 10:63–211
6. Bodin L, Golden B (1981) Classification in vehicle routing and scheduling. Networks 11:97–108
7. Bokinge U, Hasselström D (1980) Improved vehicle scheduling in public transport through systematic changes in the time-table. Europ J Oper Res 5:388–395
8. Carpaneto G, Dell'Amico M, Fischetti M, Toth P (1989) A branch and bound algorithm for the multiple depot vehicle scheduling problem. Networks 19:531–548
9. Carraresi P, Gallo G (1984) Network models for vehicle and crew scheduling. Europ J Oper Res 16:139–151
10. Ceder A (1995) Minimum cost vehicle scheduling with different types of transit vehicles. In: Daduna JR, Branco I, Paixão J (eds) Computer-aided transit scheduling. Springer, Berlin, pp 102–114
11. Ceder A, Stern HI (1981) Deficit function bus scheduling with deadheading trip intersections for fleet size reduction. Transport Sci 15:338–363
12. Ceder A, Stern HI (1985) The variable trip procedure used in the AUTOBUS vehicle scheduler. In: Rousseau JM (ed) Computer scheduling in public transport, vol 2. North-Holland, Amsterdam, pp 371–390
13. Costa A, Branco I, Paixão J (1995) Vehicle scheduling with multiple types of vehicles and a single depot. In: Daduna JR, Branco I, Paixão J (eds) Computer-Aided Transit Scheduling. Springer, Berlin, pp 115–129
14. Daduna JR (1988) A decision support system for vehicle scheduling in public transport. In: Gaul W, Schader M (eds) Data, expert knowledge and decisions. Springer, Berlin, pp 93–102
15. Daduna JR, Branco I, Paixão J (ed) (1995) Computer-aided transit scheduling. Springer, Berlin
16. Daduna JR, Mojsilovic M (1988) Computer-aided vehicle and duty scheduling using the HOT programme system. In: Daduna JR, Wren A (eds) Computer-aided transit scheduling. Springer, Berlin, pp 133–146

17. Daduna JR, Mojsilovic M, Schütze P (1993) Practical experiences using an interactive optimization procedure for vehicle scheduling. In: Du D-Z, Pardalos PM (eds) Network optimization problems: Algorithms, applications and complexity. World Sci., Singapore, pp 37–52
18. Daduna JR, Paixã J (1995) Vehicle scheduling for public mass transit. In: Daduna JR, Branco I, Paixão J (eds) Computer-aided transit scheduling. Springer, Berlin, pp 37–52
19. Daduna JR, Völker M (1996) Operations research methods and their application within the HOT II system of computer-aided planning for public transport. In: Fortuin L, van Beek P, Van Wassenhove L (eds) OR at work–Case studies in the application of OR in industry, service, agriculture and health care. Francis & Taylor, Milton Park, pp 147–162
20. Daduna JR, Völker M (1997) Fahrzeugumlaufbildung im ÖPNV mit unscharfen Abfahrtszeiten. Der Nahverkehr 11:39–43
21. Daduna JR, Wren A (eds) (1988) Computer-aided transit scheduling. Springer, Berlin
22. Desrochers M, Rousseau J-M (1992) Computer-aided transit scheduling. Springer, Berlin
23. Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995) Time constraint routing and scheduling. In: Ball MO, Magnanti TL, Momna CL, Nemhauser GL (eds) Network Routing. Elsevier, Amsterdam, pp 35–139
24. Forbes MA, Holt JN, Watts AM (1994) An exact algorithm for multiple depot bus scheduling problems. Europ J Oper Res 72:115–124
25. Freling R (1997) Models and techniques for integrating vehicle and crew scheduling. Tinbergen Inst Res Ser
26. Freling R, Paixão J (1995) Vehicle scheduling with time constraint. In: Daduna JR, Branco I, Paixão J (eds) Computer-Aided Transit Scheduling. Springer, Berlin, pp 130–144
27. Friberg C, Haase K (1999) An exact algorithm for the vehicle and crew scheduling problem. In: Wilson NHM (ed) Computer-Aided Transit Scheduling. Springer, Berlin, pp 63–80
28. Gavish B, Schweitzer P, Shilfer E (1978) Assigning buses to schedules in a metropolitan area. Comput Oper Res 5:129–138
29. Gavish B, Shilfer E (1978) An approach for solving a class of transportation scheduling problems. Europ J Oper Res 3:122–134
30. Grötschel M, Löbel A, Völker M (1997) Optimierung des Fahrzeugumlaufs im öffentlichen Nahverkehr. In: Hofmann KH, Jäger W, Lohmann T, Schunk H (eds) Mathematik–Schlüsseltechnologie für die Zukunft. Springer, Berlin, pp 609–624
31. Hamer N, Se´guin L (1992) The HASTUS system: New algorithms and modules for the 90s. In: Desrochers M, Rousseau J-M (eds) Computer-aided transit scheduling. Springer, Berlin, pp 17–29
32. Lamatsch A (1992) An approach to vehicle scheduling with depot capacity constraints. In: Desrochers M, Rousseau J-M (eds) Computer-aided transit scheduling. Springer, Berlin, pp 181–195
33. Löbel A (1997) Experiments with a Dantzig–Wolfe decomposition for multi-depot vehicle scheduling problems. Preprint SC –Zuse–Zentrum Informationstechnik, 97-16. www.zib.de
34. Löbel A (1998) Optimal vehicle scheduling. Shaker, Maastricht
35. Luedtke LK (1985) RUCUS II: A review of system capabilities. In: Rousseau J-M (ed) Computer scheduling in public transport, vol 2. North-Holland, Amsterdam, pp 61–115
36. Mesquita M, Paixão J (1992) Multiple depot vehicle scheduling problem: A new heuristic based on quasi-assignment algorithms. In: Desrochers M, Rousseau J-M (eds) Computer-aided transit scheduling. Springer, Berlin, pp 167–180
37. Mesquita M, Paixão J (1999) Exact algorithms for the multi-depot vehicle scheduling problem based on multicommodity network flow type formulations. In: Wilson NHM (ed) Computer-Aided Transit Scheduling. Springer, Berlin, pp 221–243
38. Mojsilovic M (1983) Verfahren für die Bildung von Fahrzeugumlä, Dienstplänen und Dienstreihenfolgenplänen in Verkehr und Transport. HEUREKA 83, Karlsruhe, pp 178–191
39. Nonato M, Gaffi A (1999) An integrated approach to extra-urban crew and vehicle scheduling. In: Wilson NHM (ed) Computer-Aided Transit Scheduling. Springer, Berlin, pp 103–128
40. Odoni AR, Rousseau J-M, Wilson NHM (1994) Models in urban and air transportation. In: Pollock SM, Rothkopf MH, Barnett A (eds) Handbook Oper. Res. and Management Sci., vol 6. North-Holland, Amsterdam, pp 107–150
41. Orloff CS (1976) Route constraint fleet scheduling. Transport Sci 10:149–168
42. Patrikalakis G, Xerocostas D (1992) A new decomposition scheme of the urban public transport scheduling problem. In: Desrochers M, Rousseau J-M (eds) Computer-aided transit scheduling. Springer, Berlin, pp 407–425
43. Paixão J, Branco I (1987) A quasi-assignment algorithm for bus scheduling. Networks 17:249–269
44. Paixão J, Branco I (1988) Bus scheduling with a fixed number of vehicles. In: Daduna JR, Wren A (eds) Computer-aided transit scheduling. Springer, Berlin, pp 28–40
45. Ribeiro CC, Soumis F (1994) A column generation approach to the multiple-depot vehicle scheduling problem. Oper Res 42:41–52
46. Rousseau JM (ed) (1985) Computer scheduling in public transport, vol 2. North-Holland, Amsterdam
47. Soumis F, Desrosiers M, Desrochers M (1985) Optimal urban bus routing with scheduling flexibilities. In: Thoft-

Christensen P (ed) System modelling and optimization. Springer, Berlin, pp 155–165
48. Tosini E, Vercellis C (1988) An interactive system for extra urban vehicle and crew scheduling problems. In: Daduna JR, Wren A (eds) Computer-aided transit scheduling. Springer, Berlin, pp 41–53
49. Wilson NHM (ed) (1999) Computer-aided transit scheduling. Springer, Berlin
50. Wren A (ed) (1981) Computer scheduling in public transport. North-Holland, Amsterdam

Volume Computation for Polytopes: Strategies and Performances

Andreas Enge
Lehrstuhl Diskrete Math.,
Optim. und OR Institute Math., University Augsburg, Augsburg, Germany

MSC2000: 52B11, 52B45, 52B55

Article Outline

Keywords
Polytopes and Volume
Complexity Results
Basic Approaches and Duality
  Triangulations
  Signed Decompositions
  Duality
Algorithms
  Delaunay Triangulation
  Boundary Triangulation
  Triangulation by Cohen and Hickey
  Lawrence's Signed Decomposition
  Lasserre's Signed Decomposition
  Hybrid Orthonormalization Technique
Choosing an Algorithm
  Low Dimension
  Near-Simple and Near-Simplicial Polytopes
  Double Representation
  Representation Conversion
See also
References

Keywords

Volume; Convex polytope; Algorithms

Polytopes and Volume

A *convex polytope* P in dimension d can be described in two ways: First, it is the convex hull of a finite set of points $\{v_1, \ldots, v_n\} \subseteq \mathbf{R}^d$. Second, it is the bounded intersection of a finite number of half spaces, i. e. given as $\{x\colon Ax \leq b\}$ for a matrix $A \in \mathbf{R}^{m\times d}$ and a vector $b \in \mathbf{R}^m$. In the first case, the polytope is given by its $\mathcal{V}$-*representation*, in the second case by its $\mathcal{H}$-*representation*.

Only full dimensional polytopes, i. e. polytopes which are not contained in any hyperplane of $\mathbf{R}^d$, are of interest in the context of volume computation. If P is such a polytope, then its volume $\mathrm{Vol}(P) = \mathrm{Vol}_d(P)$, as usual defined by the d-dimensional Lebesgue measure, is not zero. A minimal $\mathcal{V}$-representation of P is unique and given by the vertices of P. Also a minimal $\mathcal{H}$-representation is unique (up to multiplication of rows of A and the corresponding entries of b by positive scalars), and the sets $P \cap \{x\colon a_i\, x = b\}$, where a_i is the ith row of A in such a minimal representation, define the facets of P.

There are numerous applications of polytope volume computation, ranging from estimating the size of the solution space of a linear program to counting the number of roots of a system of complex polynomial equations. To date, most practical applications concern low dimensions. This is partially due to the fact that the complexity of volume computation algorithms increases exponentially when the dimension grows, partially to limited experience with these algorithms.

This contribution is concerned with exact algorithms for computing polytope volumes which have been implemented successfully. Randomized approximation algorithms as described in [7] are omitted from the discussion because so far they have not been programmed.

Complexity Results

Simple examples show that the minimal $\mathcal{V}$-representation of a polytope can have exponential size with respect to its minimal $\mathcal{H}$-representation, and vice versa. Hypercubes, for instance, have $2d$ facets and 2^d vertices in dimension d, and the converse holds for their duals, the *cross polytopes* (see below for more about polytope duality). Hence it is not surprising that the complexity of volume computation depends on the polytope rep-

resentation. The complexity results of this section are valid for polytopes represented over **Z** or **Q** and are formulated with respect to the bit length of the input. Detailed proofs can be found in [6].

For some classes of polytopes there are polynomial time algorithms computing their volumes. An important such class is formed by the polytopes in fixed dimension, given either in $\mathcal{H}$- or $\mathcal{V}$-representation, which reflects that geometric 2- or 3-dimensional problems can be solved efficiently in practice.

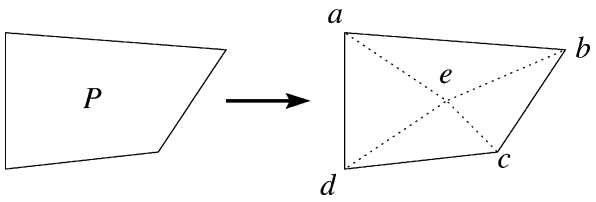
Moreover, there are polynomial algorithms for polytopes P satisfying the following condition: P is given in $\mathcal{H}$-representation, and there is a constant δ such that each facet of P contains at most δ vertices (the *near-simplicial* case). Or, P is given in $\mathcal{H}$-representation, and there is a constant δ such that each vertex of P is contained in at most δ facets (the *near-simple* case). In particular, the case $\delta = d$ covers *simplicial* $\mathcal{H}$- and *simple* $\mathcal{V}$-polytopes. However, this result seems to be of little practical value. For instance, simple polytopes are usually given by their $\mathcal{H}$-representation, and their $\mathcal{V}$-representation already tends to be exponential. Examples for this phenomenon are provided by hypercubes or polytopes constructed from randomly chosen halfspaces.

All known algorithms are exponential in d respectively δ. This is inevitable for $\mathcal{H}$-polytopes, since the binary size of the volume of a polytope may be exponential in its $\mathcal{H}$-representation. Some of the triangulation methods described below (boundary triangulation, for instance) show that the volume of a polytope has polynomial size with respect to the $\mathcal{V}$-representation. However, even in this case volume computation is not an easy task. Indeed, the problem is #P-hard for polytopes in either $\mathcal{H}$- or $\mathcal{V}$-representation (see [4]).

The complexity of the volume computation problem is unknown when both representations are given. This problem might even be solvable in polynomial time with one of the known algorithms.

Basic Approaches and Duality

All deterministic volume computation methods decompose a given polytope into polytopes whose volumes are easier to compute. Especially apt for volume computation purposes are *simplices*, i. e. d-dimensional polytopes with $d+1$ vertices. Their volume is given by



Volume Computation for Polytopes: Strategies and Performances, Figure 1

a determinant, precisely,

$$\mathrm{Vol}(\mathrm{conv}(\{v_0, \ldots, v_d\})) = \frac{|\det(v_1 - v_0, \ldots, v_d - v_0)|}{d!}.$$

Depending on the decomposition into simplices, two basic classes of algorithms can be distinguished.

Triangulations

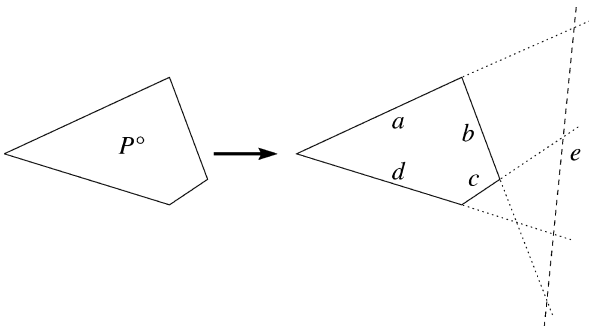
A triangulation of a polytope P is a finite collection $\{\Delta_i : i = 1, \ldots, s\}$ of simplices such that $P = \cup_{i=1}^{s} \Delta_i$ and the intersection of any two simplices is their common face. (This is the usual definition of a triangulation; in fact, for volume computation purposes it would be enough to demand that no two simplices share an interior point.) Then clearly

$$\mathrm{Vol}(P) = \sum_{i=1}^{s} \mathrm{Vol}(\Delta_i).$$

For example, if P is simplicial, i. e., all its facets are $d-1$-dimensional simplices, then linking these facets to a fixed interior point as illustrated in Fig. 1 yields a triangulation of the whole polytope, called *boundary triangulation*. Other algorithms are presented below.

Signed Decompositions

The restriction that no two simplices may share an interior point is in fact not necessary. If it is violated, then some parts of the polytope are covered more than once, which must be corrected by subtracting their respective volumes. A *signed decomposition* of P is thus a finite collection $\{(\Delta_i, \sigma_i) : i = 1, \ldots, s\}$ of simplices Δ_i and signs $\sigma_i \in \{\pm 1\}$ with the following property: If a point of P does not lie on the boundary of any simplex, then the number of positive simplices containing it exceeds the number of negative simplices containing it by 1. A point



Volume Computation for Polytopes: Strategies and Performances, Figure 2

outside P is contained in as many positive as negative simplices. It follows that

$$\mathrm{Vol}(P) = \sum_{i=1}^{s} \sigma_i \,\mathrm{Vol}(\Delta_i).$$

An example for a signed decomposition is given by the algorithm of J. Lawrence [9], see Fig. 2. Let P be simple, and introduce an additional hyperplane e not parallel to any edge of the polytope. The signed decomposition consists of one simplex for each vertex, which is formed by e and the hyperplanes incident with this vertex. How the sign is determined will be described below. In Fig. 2, the signed decomposition is {(*ade*, +), (*abe*, −), (*cde*, −), (*bce*, +)}. In fact, triangulations are special cases of signed decompositions with all signs being +1. Due to duality results a distinction between these two classes is reasonable nevertheless.

Duality

Assume that the origin lies in the interior of P, which may be obtained by a translation into any interior point. Then the *polar dual* of P is defined to be $P^\circ = \{x \in \mathbf{R}^d : y^\top x \leq 1 \text{for all vertices } y \text{ of } P\}$. Hence the vertices of P are in bijection with the facets of P°, and the converse also holds. This means that the $\mathcal{V}$-representation of P corresponds to the $\mathcal{H}$-representation of P° (with the right-hand side b normalized to the all one vector), and vice versa. For instance, the polytopes of Fig. 1 and Fig. 2 are dual to each other.

Polarity reverses inclusion, so that the duals of simple polytopes are simplicial and vice versa, and the combinatorial structure of P° depends only on the combinatorial structure of P. The exact geometry of P° and its volume, however, depend on the location of the origin in P. A result due to P. Filliman [5] shows that to each triangulation of P corresponds a signed decomposition of P°, if the origin is located conveniently. Precisely, let $\mathcal{T}$ be a triangulation of P such that the origin does not lie on any hyperplane spanned by the vertices of the triangulation. Then the following procedure yields a signed decomposition of P°: For a simplex $\Delta \in \mathcal{T}$ let its *separation number* s be the number of its facets separating it from the origin. Let Δ° be the simplex bounded by the hyperplanes corresponding by polarity to the vertices of Δ, and let its sign be $\sigma = (-1)^s$.

In this sense, Lawrence's decomposition of P° in Fig. 2 is induced by the boundary triangulation of P in Fig. 1.

Suppose now that the complexity of a decomposition is measured by the number of simplices it generates (which is a simplified model since it does not take the sizes of the occurring numbers into account). Let $\mathcal{P}$ be a class of polytopes containing the origin in their interiors, and let $\mathcal{P}^\circ$ be composed of the duals of the elements in $\mathcal{P}$. Then for each triangulation algorithm on $\mathcal{P}$ (working with the $\mathcal{V}$-, $\mathcal{H}$- or both representations), there is a signed decomposition algorithm on $\mathcal{P}^\circ$ (working with the $\mathcal{H}$-, $\mathcal{V}$- or both representations) with the same complexity. This explains why most of the complexity results above are symmetric with respect to duality, and that asymmetry may only occur when the binary lengths of the numbers involved are taken into account.

Algorithms

In this section, some volume computation algorithms are presented in more detail. While the following list is not exhaustive, all of the described methods have been implemented and tested on different examples (see [2]).

Delaunay Triangulation

The classical *Delaunay triangulation* is obtained by lifting the polytope vertices onto the surface of a $d+1$-dimensional convex body. Precisely, let $f:\mathbf{R}^d \to \mathbf{R}$ be a strictly convex function. (Traditionally, $f(x) = \sum_{i=1}^{d} x_i^2$.) Lift each vertex v to $(v, f(v))$. Then the lifted vertices will usually be in *general position*, i. e., their convex hull will be simplicial. Interpreting these simplicial facets in terms of the original vertices yields a triangulation. If

the lifted points are not in general position – which is frequent for the traditional lifting function on 0– 1-polytopes – then perturbation or the choice of a different lift resolves the problem. The algorithm needs the $\mathcal{V}$-representation. Its time and space complexity and numerical behavior depend on the convex hull algorithm used for computing the facets of the lifted polytope.

Boundary Triangulation

The algorithm for simplicial polytopes has been described above. If P is not simplicial itself, then perturbation of its vertices leads to a simplicial polytope, whose facets can be determined by any convex hull algorithm. Identifying the perturbed vertices with the original ones leads to a triangulation of P. Again, only the $\mathcal{V}$-representation is needed, and the behavior of the algorithm is determined by the underlying method for computing the convex hull.

Triangulation by Cohen and Hickey

An obvious improvement of any boundary triangulation is obtained by using a vertex instead of an interior point, thus dropping all simplices of the boundary triangulation containing this vertex. If in Fig. 1, for instance, e were translated into a, then only the two simplices abc and acd would be left over. Together with a very efficient scheme of handling nonsimpliciality, this observation is the basis of the triangulation algorithm described by J. Cohen and T. Hickey [3]. Suppose that $e_1, \ldots, e_m$ are the facets of P, given by the sets of vertices they contain. (This requirement is equivalent to the knowledge of both the $\mathcal{V}$- and the $\mathcal{H}$-representation of P.) Fix a vertex $v(P)$. Then the set $\{\text{conv}(v(P), e_i\ v(P) \notin e_i\}$ is a decomposition of P into pyramids with bases e_i and apex v. If the polytope is simplicial, then the pyramids are in fact simplices. Otherwise their bases are triangulated recursively, and the resulting $d-$ 1-dimensional simplices, together with the fixed vertex $v(P)$, form a triangulation of P. More precisely, let b be such a base, again represented by a set of vertices. Fix a vertex $v(b) \in$ b. As above, $\{\text{conv}(v(b), \text{e}):v(b) \notin e\}$, where e varies over the facets of b, i. e. the $b-1$-dimensional faces of P contained in b, forms a decomposition of b into pyramids. The recursion continues with the e's until a simplicial face is reached, which is the case at the latest in dimension 1. While the e's are not known a priori for $b \neq P$, they are exactly the $b-$ 1-dimensional sets within $\{b \cap e_i: v(b) \notin e_i\}$. Instead of testing the dimensions of these sets, which turns out to be too costly in practice, one can just continue the algorithm with all of them. If the dimension of $b \cap e_i$ is lower than needed, then this recursion branch will end prematurely with an empty face and not contribute a simplex to the triangulation.

Since Cohen and Hickey's triangulation scheme is purely combinatorial – it uses only incidence information and not the vertex coordinates to obtain a triangulation – it raises no numerical problems. Storing faces as sets of their vertices facilitates the detection of simplicial faces, as a face in (supposed) dimension d' is simplicial if and only if it has $d'+1$ vertices. Similarly, faces with supposed dimension d' and less than $d'+1$ vertices can be dropped, as their dimension is in fact lower than d'.

Lawrence's Signed Decomposition

Assume first that P is simple. To apply the scheme presented in the section about signed decompositions, it must be known how to compute the volume of the simplex associated with a vertex v of P. Lawrence [9] derives the following formula. Let the additional hyperplane be $\{x: c^\top x = 0\}$, where $x \longmapsto c^\top x$ is nonconstant on any edge of P. Denote by A_v the $d \times d$-matrix corresponding to the facets containing v. (This matrix can be derived if both the $\mathcal{H}$- and the $\mathcal{V}$-representation of P are known.) Then A_v is invertible, $\gamma^v = (A_v^\top)^{-1}c$ is well defined up to permutations of its entries, and none of its entries is zero. Finally,

$$\text{Vol}(P) = \sum_v \frac{(c^\top v)^d}{d!\,|\det A_v| \prod_{i=1}^d \gamma_i^v}.$$

If P is not simple, then a vertex v is contained in more than d facets, and A_v is not square any more. In this case one perturbs the $\mathcal{H}$-representation lexicographically and runs a vertex enumeration algorithm. This results in possibly several new simple vertices for each vertex of the original polytope. Each new vertex corresponds to a lexicographically feasible cobasis A_v of the original vertex and contributes a term to the sum above. The same procedure must be applied if only the $\mathcal{H}$-representation of P is known.

An unresolved problem is the choice of the additional hyperplane, i. e. of c. For rational polytopes it is possible to choose c of size polynomial in the $\mathcal{H}$-representation, so that each term of the sum has polynomial size. However, the nontrivial denominators often result in prohibitively large numbers after a few terms are summed up. On the other hand, using floating point arithmetic often leads to severe numerical instabilities when the denominator is close to zero in some vertices, so that numbers of drastically different magnitudes are summed up.

Lasserre's Signed Decomposition

Like Cohen and Hickey's triangulation, the algorithm of J.B. Lasserre [8] is based on a decomposition of the polytope into pyramids and proceeds by recursion. However, it uses only the $\mathcal{H}$-representation. Let a_i be the ith row of A, b_i the ith entry of b and $e_i = \{x \in P: a_i x = b_i\}$. Suppose that if e_i is parallel to e_j for some $i \neq j$, then b_i and b_j have different signs. (This is no restriction, since otherwise one of them is redundant, but it has to be tested in an implementation.) Then

$$\mathrm{Vol}(P) = \frac{1}{d} \sum_{i=1}^{m} \frac{b_i}{\|a_i\|} \mathrm{Vol}_{d-1}(e_i),$$

where $|b_i|/||a_i||$ is the distance of the origin from e_i. A negative sign occurs for negative b_i, i. e., when the hyperplane corresponding to e_i separates the origin from the polytope. This formula does not yet allow a recursive implementation, since the $d-1$-dimensional volumes of facets embedded into d-dimensional space are needed. Thus e_i is projected onto a suitable subspace of $d-1$ coordinates. Let $a_{ij} \neq 0$. Substituting $x_j = (b_i - \sum_k \neq j\, a_{ik} x_i)/a_{ij}$ in the system of linear inequalities $Ax \leq b$ yields a new system $A^{(i)}x^{(i)} \leq b^{(i)}$ with $m-1$ inequalities in $d-1$ variables, which describes the projection of e_i onto the coordinate subspace $\{x: x_j = 0\}$. Taking the distorting effect of this projection into account, the following formula, which can be implemented recursively, is derived:

$$\mathrm{Vol}_d(P) = \frac{1}{d} \sum_{i=1}^{m} \frac{b_i}{a_{ij}} \mathrm{Vol}_{d-1}\left(\left\{x^{(i)}: A^{(i)}x^{(i)} \leq b^{(i)}\right\}\right).$$

In the same way as Cohen and Hickey's triangulation improves on boundary triangulation, the computational effort for Lasserre's algorithm can be reduced by translating each intermediate polytope into one of its vertices. In fact, the algorithm then implicitly constructs Cohen and Hickey's triangulation, with the determinant computation spread over the recursion tree. Since finding a vertex is a rather costly linear programming step, one instead translates into a (possibly infeasible) basis solution, which preserves the character of a signed decomposition.

Noticing that the same face may be considered at several places in the recursion tree, the efficiency of the algorithm can be increased by storing the volume of a face when it is computed for the first time, and retrieving the volume when a face reappears. Some care must be taken, however, when a face is projected onto different coordinate subspaces; then a determinant computation is required. For details, see [2].

Unlike in Cohen and Hickey's triangulation algorithm, there is no cheap way of testing whether an intermediate face is empty or of a too low dimension.

Hybrid Orthonormalization Technique

In [2] an algorithm is described which combines the advantages of Cohen and Hickey's and Lasserre's methods. On one hand, it exploits the information of both the $\mathcal{H}$- and the $\mathcal{V}$-representation, on the other hand, it stores and retrieves intermediate results.

Consider again the dissection into pyramids

$$\{\mathrm{conv}(v(P), e_i): \; v(P) \notin e_i\}$$

of Cohen and Hickey's triangulation algorithm, where as before the e_i's are given as the sets of their vertices. The volume formula for pyramids yields

$$\mathrm{Vol}_d(P) = \sum_{v(P) \notin e_i} \frac{1}{d} \mathrm{dist}(v(P), \mathrm{aff}(e_i)) \, \mathrm{Vol}_{d-1}(e_i),$$

where $\mathrm{dist}(v(P), \mathrm{aff}(e_i))$ denotes the distance of the pyramid apex to the affine subspace corresponding to the pyramid base. Suppose that $\mathrm{Vol}_{d-1}(e_i)$ and on orthonormal basis of the linear subspace associated with e_i are known. Then the required distance as well as an orthonormal basis of the linear subspace corresponding to the pyramid with basis e_i can be computed easily.

Since the volume and an orthonormal basis of a one-dimensional polytope are trivial to determine, a recursive approach analogous to Cohen and Hickey's algorithm can be adopted.

So far, there is no advantage over the triangulation scheme. (In fact, the overhead of basis computation makes this algorithm slower in practice.) However, the strategy of storing and reusing intermediate results, which has been successfully employed to accelerate Lasserre's algorithm, is applicable now. Volumes and orthonormal bases of already visited faces can be stored and retrieved as soon as the face is reconsidered. As the bases require an enormous amount of storage space, it is reasonable to store only the face volumes and to recompute orthonormal bases from scratch when necessary.

The algorithm requires both the $\mathcal{H}$- and the $\mathcal{V}$-representation. Unlike all other methods presented above, it can only be implemented using floating point arithmetic and not rational arithmetic because orthonormalization involves square roots. Care must be taken to choose a numerically stable orthonormalization technique, e. g. using Householder transformations.

Choosing an Algorithm

Experience with volume computation in higher dimensions is very limited; to date, only one study has been published [2]. Hence, while the following recommendations reflect the state of the art, they may soon be affected by algorithmic progress or new experimental results.

Low Dimension

In accordance with the theoretical complexity results, volume computation is a simple task in practice if one restricts oneself to low dimensions (up to about 5). An algorithm should be chosen which works with the representation in which the polytopes are given.

Near-Simple and Near-Simplicial Polytopes

It can be observed that in practice triangulation methods behave particularly well on near-simplicial $\mathcal{V}$-polytopes, and that signed decomposition methods behave particularly well on near-simple $\mathcal{H}$-polytopes. This is in accordance with the general duality result above, but at first sight surprising when compared to the complexity result stating that these problems are polynomial when the polytopes are given by their converse representations. However, the polynomial complexity is obtained by solving a large number of linear programs, which is apparently not competitive for problems of a tractable size.

Lawrence's method, which generates a signed decomposition with especially few simplices for near-simple $\mathcal{H}$-polytopes, suffers from numerical instabilities as outlined above, so that Lasserre's algorithm is preferable in general.

Double Representation

If both representations are known, then the hybrid orthonormalization technique proves to be the most efficient algorithm in practice. Although it is closer in spirit to a triangulation method, it is usually faster even than signed decomposition algorithms on near-simple polytopes since it efficiently exploits the additional structural information from the $\mathcal{V}$-representation.

Representation Conversion

The cases left over as difficult are those of polytopes given by the 'wrong' representation. An experimental finding is that $\mathcal{V}$-polytopes with a large ratio n/m of vertices to facets and $\mathcal{H}$-polytopes with a small ratio n/m pose problems. Unfortunately, most algorithms for converting between the representations face an 'easy' and a 'hard' direction, and exactly the hard direction is needed here. It has been observed, however, that on suitable classes of polytopes both directions have essentially the same complexity; the result is obtained using the easy transformation as an oracle for the hard one [1]. This technique will probably allow to efficiently apply the hybrid orthonormalization technique to those polytopes whose volumes are particularly hard to compute today.

See also

► Quadratic Programming Over an Ellipsoid

References

1. Bremner D, Fukuda K, Marzetta A (1998) Primal-dual methods for vertex and facet enumeration. Discrete Comput Geom 20:333–357
2. Büeler B, Enge A, Fukuda K (2000) Exact volume computation for polytopes: A practical study. In: Kalai G, Ziegler GM (eds) Polytopes – Combinatorics and Computation, DMV Seminar 29. Birkhäuser, Basel
3. Cohen J, Hickey T (1979) Two algorithms for determining volumes of convex polyhedra. J ACM 26(3):401–414
4. Dyer ME, Frieze AM (1988) On the complexity of computing the volume of a polyhedron. SIAM J Comput 17(5):967–974
5. Filliman P (1992) The volume of duals and sections of polytopes. Mathematika 39:67–80
6. Gritzmann P, Klee V (1994) On the complexity of some basic problems in computational convexity: II. Volume and mixed volumes. In: Bisztriczky T, McMullen P, Schneider R, Weiss AI (eds) POLYTOPES: Abstract, Convex and Computational. Kluwer, Dordrecht, pp 373–466
7. Kannan R, Lovász L, Simonovits M (1997) Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. Random Struct Algorithms 11(1):1–50
8. Lasserre JB (1983) An analytical expression and an algorithm for the volume of a convex polyhedron in Rn. J Optim Th Appl 39(3):363–377
9. Lawrence J (1991) Polytope volume computation. Math Comput 57(195):259–271

Von Neumann, John

PAVEENA CHAOVALITWONGSE
University Florida, Gainesville, USA

MSC2000: 01A99, 90C99

Article Outline

Keywords
See also
References

Keywords

Theory of games; Hilbert space; Logical design; Numerical analysis; Duality theorem; Theory of automata

> *Young man, in mathematics you don't understand things, you just get used to them.*
>
> John von Neumann

A very intelligent mathematician and scientist, John von Neumann (1903–1957) worked in the area of set theory, *game theory*, economic behavior, operator algebra, quantum mechanics, computer science, neural network, and the *theory of automata*. He was also one of the first five professors at the Institute for Advanced Studies (IAS), whose purpose was 'the pursuit of advanced learning and exploration in fields of pure science and high scholarship' [5].

Von Neumann was born in Budapest on December 28, 1903. He was a mathematical prodigy with extraordinarily fast thinking and an effectively 'photographic memory'. In 1921 he was sent to study at the Lutheran Gymnasium (Agostai Hitvallasu Evangelikus Fogimnazium), one of three well-respected high schools in Budapest [1]. During the eight years of Von Neumann's high school career, his father arranged a professional mathematician to tutor him at home so that his remarkable mathematical talent would be advanced.

At the age of 18, von Neumann had his first mathematical paper published, with M. Fekete, his tutor. This paper showed how to solve a problem on the location of zeroes of certain minimal polynomials. He was awarded 'excellence in mathematics and scientific reasoning' in a nationwide high school competition [4].

In 1921, von Neumann enrolled in both the Univ. of Budapest and the Univ. of Berlin to study mathematics. In 1925, he was awarded the Bachelor degree in chemical engineering from the Eidgenossische Technische Hochschule (ETH) or Swiss Federal Institute of Technology. A year later, he was awarded the Ph.D. in mathematics at the Univ. of Budapest.

During the 1920s, while von Neumann was in Europe, he focused hisworks in two main areas: 'set theory and logical foundations of mathematics', and 'Hilbert space theory, operator theory, and the mathematical foundations of quantum mechanics' [7]. On top of that he successfully gained a reputation for his work on set theory and quantum mechanics especially the theory of measurement [1].

In 1930, von Neumann was invited to work at Princeton Univ. which was recognized as a global center for mathematicians during the early 1930s [3]. A few years later, he was appointed to teach at the IAS. Like other mathematicians at that time, von Neumann was fascinated by Hilbert research. One of his important works in Hilbert space was the theory of rings of op-

erators (*W∗-algebras* or *von Neumann algebras*) which are algebras of bounded operators in a separable Hilbert space [8]. He applied 'modern algebra to algebras of operators in a Hilbert space'. Later on, working with F.J. Murray, von Neumann examined the continuous geometry associated with the ring of operators [2,9,10].

Von Neumann was also interested in the theory of games [12]. His great achievement in this area was the book [14], written with the Austrian economist O. Morgenstern. The book contains the mathematics of game theory and its application to a variety of economic problems.

During World War II, the need for advanced computing technology increased within various military research programs. Several scientists, physicists, and mathematicians, who worked in those programs, faced problems that could not be solved analytically. As a result some experiments or numerical methods were used to determine solutions to these difficult. Like other mathematicians, von Neumann also participated in those research programs as a consultant. He got involved in aerodynamics, high explosives, atomic bomb, electronics, the development of high-speed calculating machines, etc. Von Neumann wanted a powerful calculating system that could solve nonlinear partial differential equations of more than one independent variable.

At the Univ. of Pennsylvania's Moore School of Electrical Engineering, the ENIAC (Electronic Numerical Integrator and Computer), a programmable electronic calculator, was completely designed. The ENIAC, regarded as the first modern computer, was able to work 'at electronic speed'. As soon as von Neumann heard about this machine, he rushed to see the ENIAC despite of its full completion. During the time he visited the Moore School, he discussed with the staff about the design of the EDVAC (Electronic Discrete Variable Arithmetic Computer), a stored-program computer; so he decided to participate in developing this machine. While working on EDVAC project, von Neumann was writing [6] in 1945 to describe the stored-program computer, particularly in its logical control [1,2]. His report was successfully approved throughout the United States and Britain.

After the war, von Neumann directed his attention to the Electronic Computer Project because he expected an increase on computer need in scientific research. One of his concerns on computing systems was their speed. In [13], he and his co-author defined factors that affected overall speed [1,2]. With A. Burks and H. Goldstine, von Neumann also wrote a report, 'Preliminary Discussion of the Logical Design of an Electronic Computing Instrument', on logical design known as *von Neumann architecture* [1,2].

As the power of computer increased, the use of numerical analysis was stimulated after a declining state during the 1930s. Although the computer gave mathematical scientists an opportunity to study larger and more complex systems of linear equations, partial differential equations, etc., existing iterative methods were ineffective to solve problems using computer. For this reason, von Neumann began examining more efficient and more reliable algorithms for the computer [11]. The methods of Monte-Carlo and the *duality theorem* in linear programming are the two most distinguished results that he contributed in computer-oriented numerical analysis.

The computer has been a necessary tool for the achievement of many scientific and engineering research as von Neumann's predicted. He used computer to solve problems in fluid dynamics, meteorology, atomic and nuclear physics, partial differential equations, numerical analysis, linear programming, etc. [11,12]. Von Neumann interest in computer seemed to be never lessened. The theory of natural and artificial automata [11] is also one of his computing research accomplishment during the last years of his life. It examines general solutions to the problems of organization, structure, language, information, and control [1,2].

Von Neumann's problem-solving ability and wide-ranging interests enabled him to produce a large number of contributions to various fields. Moreover his fast thinking and effective memory allowed him to reduce the complexity of many problems. He had never stopped thinking about mathematics. Thus it is not surprising that von Neumann quickly became an intellectual in both pure and applied mathematics.

See also

- ► Duality Theory: Biduality in Nonconvex Optimization
- ► Duality Theory: Monoduality in Convex Optimization

► Duality Theory: Triduality in Global Optimization
► History of Optimization

References

1. Aspray W (1990) John von Neumann and the origins of modern computing. MIT, Cambridge
2. Aspray W, Burks A (eds) (1987) Papers of John von Neumann on computing and computer theory. MIT, Cambridge
3. Halperin I (1990) The extraordinary inspiration of John von Neumann. The Legacy of John von Neumann. Amer. Math. Soc., Providence
4. Heims HJ (1980) John Neumann and Norbert Wiener: From mathematics to the technology of life and death. MIT, Cambridge
5. Leitch A (1978) A Princeton companion. Princeton Univ. Press, Princeton
6. von Neumann J (1945) First draft of a report on the EDVAC. Moore School Electrical Engin. Univ. Pennsylvania
7. von Neumann J (1961) Logic theory of sets and quantum mechanics, vol 1. Pergamon, Oxford
8. von Neumann J (1961) Operators, ergodic theory and almost periodic functions in a group, vol 2. Pergamon, Oxford
9. von Neumann J (1961) Rings of operators, vol 3. Pergamon, Oxford
10. von Neumann J (1962) Continuous geometry and other topics, vol 4. Pergamon, Oxford
11. von Neumann J (1963) Design of computers, theory of automata and numerical analysis, vol 5. Pergamon, Oxford
12. von Neumann J (1963) Theory of games, astrophysics, hydrodynamics and meteorology, vol 6. Pergamon, Oxford
13. von Neumann J, et al On the principles of large scale computing machines
14. von Neumann J, Morgenstern O (1953) Theory of games and economic behavior, 3rd edn. Princeton Univ. Press, Princeton

Voronoi Diagrams in Facility Location

KOKICHI SUGIHARA
Department Math. Engineering
and Information Physics, Graduate School
of Engineering University Tokyo, Tokyo, Japan

MSC2000: 90C27, 90B80

Article Outline

Keywords
Facility Location Problems
Voronoi Diagrams
Farthest-Point Voronoi Diagram
Variations in the Distance
Multiple-Facility Location
Concluding Remarks
See also
References

Keywords

Voronoi diagram; Farthest-point Voronoi diagram; Largest empty circle; Smallest enclosing circle; L1-distance; L∞-distance; Lp-distance; Elliptic distance; Facility location; Convex hull; Manhattan distance

Facility Location Problems

Let T be a convexpolygon in the plane $\mathbf{R}^2$, and $S = \{P_1, \ldots, P_n\}$ be a set of n points in T. Let us denote by $d(P, Q)$ the Euclidean distance between P and Q. The following are typical location problems.

Problem 1 Find a point $P = P^*$ that attains

$$\max_{P \in T} \min_{P_i \in S} d(P, P_i). \tag{1}$$

Problem 2 Find a point $P = P^{**}$ that attains

$$\min_{P \in \mathbf{R}^2} \max_{P_i \in S} d(P, P_i). \tag{2}$$

Problem 1 is called the *largest empty-circle problem*. This is because the solution P^* of Problem 1 gives the center of the largest circle that does not contain any point of S in the interior while the center is in T; the value of (1) is the radius of the largest empty circle. Problem 2, on the other hand, is called the *smallest enclosing-circle problem*, because the solution P^{**} of Problem 2, gives the center of the smallest circle containing all the points in S; the value in (2) is the radius of the smallest enclosing circle.

These problems can be considered as facility location problems in the following sense (cf. also ► Multi-facility and restricted location problems). Suppose that T represents the shape of a city, and that $P_1, \ldots, P_n$ represent the locations of hospitals in the city. We assume that citizens go to the nearest hospitals when they need

medical care. Then, Problem 1 is to find the inhabitant who is the farthest from the nearest hospital. Hence, if the city has a budget to build another hospital, the solution P^* of Problem 1 gives the optimal location to build it in the sense that the least convenient person is benefited the most by the new hospital.

Next suppose that the city government wants to build a blood-supply center that keeps all the types of blood and delivers them to the hospitals when needed. Then, the solution P^{**} of Problem 2 gives the optimal location to build it in the sense that the longest distance to a hospital is minimized.

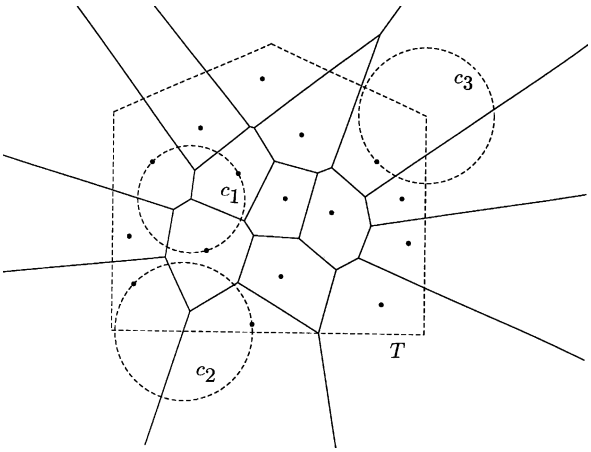
Voronoi Diagrams

For a given set $S = \{P_1, \ldots, P_n\}$ of n points, we define region $R(S; P_i)$ by

$$R(S; P_i) = \left\{ P \in \mathbf{R}^2 : \begin{array}{l} d(P, P_i) < d(P, P_j) \\ \text{for any } j \neq i \end{array} \right\}. \quad (3)$$

$R(S;P_i)$ consists of points P such that, among S, P_i is the nearest point from P. $R(S; P_i)$ is called the *Voronoi region* of P_i.

The plane $\mathbf{R}^2$ is partitioned into the Voronoi regions $R(S; P_1), \ldots, R(S; P_n)$ and their boundaries. This partition is called the *Voronoi diagram* for S. Elements of S are called the *generators* of the Voronoi diagram. Figure 1 shows an example of a Voronoi diagram, where small dots represent elements of S and solid lines represent the Voronoi diagram.



Voronoi Diagrams in Facility Location, Figure 1
Voronoi diagram and empty circles

A line, a half line or a line segment shared by the boundaries of two Voronoi regions is called a *Voronoi edge*, and a point shared by the boundaries of three or more Voronoi regions is called a *Voronoi point*.

The following properties are the direct consequences of the definition [1,7].

Lemma 3 *A Voronoi edge is on the perpendicular bisector of the associated two generators.*

Lemma 4 *A Voronoi point is the center of the circle that passes through the associated three or more generators, and there is no generator in the interior of this circle.*

An example of such a circle is shown by circle c_1 in Fig. 1.

The Voronoi diagram can be constructed by many efficient algorithms. Among them, the divide-and-conquer algorithm [10] and the plane-sweep algorithm [3] require $O(n \log n)$ time, and this time complexity is worst-case optimal. The incremental algorithm requires $O(n)$ time on the average for a wide class of distributions of the generators [6]. Numerically robust algorithms are also obtained [7,11].

Now let us return to Problem 1. Suppose that we start with a circle with radius 0 centered at an arbitrary point in T, and try to make the circle as large as possible by changing the radius and the center continuously provided that the center is in T and the circle contains no element of S in the interior. The situations in which we cannot make the circle larger can be classified into three types.

The first type is that the circle hits three points in S, as circle c_1 in Fig. 1. The second type is that the circle hits two points and the center is on the boundary of T, as circle c_2 in Fig. 1. The third type is that the circle hits one point and the center is at the corner of T, as circle c_3 in Fig. 1.

Thus, the solution P^* of Problem 1 is either:

i) a Voronoi point;
ii) a point of intersection between a Voronoi edge and the boundary of T; or
iii) a vertex of the polygon T.

Hence, we can solve Problem 1 by first constructing the Voronoi diagram and next checking all the candidate points. Since the number of Voronoi points and that of Voronoi edges are of $O(n)$, Problem 1 can be solved in $O(n \log n)$ time if the number of vertices of the polygon T is of $O(n)$.

Farthest-Point Voronoi Diagram

Reversing the inequality in (3), we define another region $R_f(S; P_i)$ by

$$R_f(S; P_i) = \left\{ P \in \mathbf{R}^2 : \begin{array}{l} d(P, P_i) > d(P, P_j) \\ \text{for any } j \neq i \end{array} \right\}. \quad (4)$$

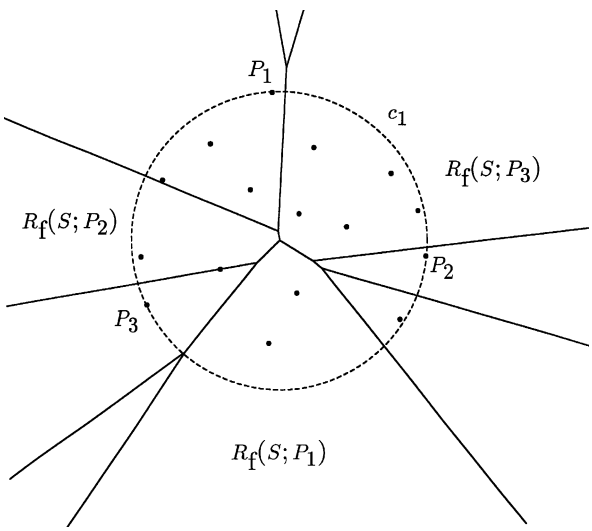
$R_f(S; P_i)$ consists of points P such that, among S, P_i is the farthest point from P. The plane is partitioned into $R_f(S; P_1), \ldots, R_f(S; P_n)$ and their boundaries. This partition is called the *farthest-point Voronoi diagram* for S. The farthest-point Voronoi diagram for the same set of points as in Fig. 1 isshown in Fig. 2.

The next property is a direct consequence of the definition [7].

Lemma 5 *A Voronoi point of the farthest-point Voronoi diagram is the center of the circle that passes through the associated three generators, and this circle contains all the elements of S.*

An example of such a circle is shown by circle c_1 in Fig. 2.

Suppose that we choose an arbitrary circle containing S, and shrink it smaller by changing both the radius and the center continuously without violating the condition that the circle contains all the elements of S. The locally minimal circle thus obtained can be classified into two types. One type is a circle hitting three or more points in S, as is shown by the circle c_1 in Fig. 2, and the other is a circle hitting two points that form a diameter of the circle.



Voronoi Diagrams in Facility Location, Figure 2
Farthest-point Voronoi diagram and the candidates of the smallest enclosing circle

Therefore, the solution P^{**} of Problem 2 is either:

i) a Voronoi point of the farthest-point Voronoi diagram for S; or
ii) the midpoint of two vertices on the boundary of the convex hull of S, where the *convex hull* of S is defined as the smallest convex region containing S.

Both the convex hull and the farthest-point Voronoi diagrams can be constructed in $O(n \log n)$time [2], and consequently Problem 2 can be solved in $O(n \log n)$ time.

Variations in the Distance

We have considered the facility location in the framework of the Euclidean distance. Sometimes, however, other distances are more realistic. For many variants of the distance, the above discussion can be applied with slight changes.

Let (x_i, y_i) be the coordinates of P_i, and (x, y) the coordinates of P. Typical variants of the distance are the following:

- *L_1-distance* (also called the *Manhattan distance*)

$$d(P, P_i) = |x - x_i| + |y - y_i|; \quad (5)$$

- *L_∞-distance*

$$d(P, P_i) = \max\{|x - x_i|, |y - y_i|\}; \quad (6)$$

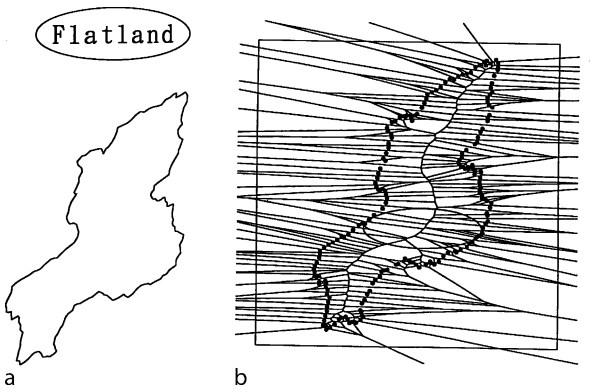
- *L_p-distance*

$$d(P, P_i) = \{|x - x_i|^p + |y - y_i|^p\}^{1/p}; \quad (7)$$

- *elliptic distance*

$$d(P, P_i) = a(x - x_i)^2 + 2b(x - x_i)(y - y_i) + c(y - y_i)^2, \quad (8)$$
$$\text{where } a > 0, ac - b^2 > 0.$$

The L_1-distance is a good approximation of the actual cost to move along the avenues in the North–South direction and along the streets in the East–West direction, just as we do in Manhattan in New York City or in the central area of Kyoto City. In this distance, the 'circle' is a square whose sides are slanted in the 45° direction with respect to the x and y axes.



Voronoi Diagrams in Facility Location, Figure 3
Elliptic-distance Voronoi diagram and the largest empty ellipse

The L_∞-distance can be observed in a mechanical x-y plotter. In the x-y plotter, a pen is moved by two step motors, one for the x direction and the other for the y direction; these two motors are controlled independently when the pen is moved in the pen-up mode. Hence, the time required to move the pen from one position to another is proportional to the L_∞-distance. A similar distance can also be found in the parts-supply system on the ceiling of a production factory.

The L_p-distance represents a general distance, including the Euclidean distance for $p = 2$, the Manhattan distance for $p = 1$ and the L_∞-distance as the limit for $p \to \infty$.

For each of these distances, Problems 1 and 2 are defined. Also the Voronoi diagram and the farthest-point Voronoi diagram can be defined similarly [1,7], and hence can be used to solve Problems 1 and 2.

It might be difficult to find an example of the elliptic distance in the actual world, but this distance is also useful for some location problems. An example is shown in Fig. 3. Suppose that, as shown in (a), we are given a map, and we want to insert the name of a place in this map in such a way that the name should be as large as possible while it should not intersect the existing symbols or lines. To solve this problem, we first find an ellipse enclosing the name text as shown in (a), next construct the Voronoi diagram with respect to the elliptic distance as shown in (b), and finally solve the largest empty-ellipse problem using this Voronoi diagram.

Note that the elliptic-distance Voronoi diagram can be constructed easily. Actually, we first transform the plane by the affine transformation that maps the given ellipse to a circle, next construct the Euclidean-distance Voronoi diagram, and finally inversely transform it to the original plane.

Multiple-Facility Location

So far we have considered the optimal location of a single new facility. Another type of the location problem is to find a given number, say n, of points that altogether attain a certain optimality. This type of the problem arises when two or more facilities can be constructed simultaneously.

A typical problem in this type is the following.

Problem 6 For a given convex region T and given number n, find the locations of n points $P_1 = P_1^*, \ldots, P_n = P_n^*$ that attain

$$\min_{P_1,\ldots,P_n \in T} \max_{P \in T} \min_{i \in \{1,\ldots,n\}} d(P, P_i). \tag{9}$$

This problem may arise in the situation that the city T has no hospital, and the city government wants to construct n hospitals (simultaneously or one by one) in such a way that, when all the n hospitals are built, the maximum distance from an inhabitant to the nearest hospital is minimized.

A more general situation is that the city already has k hospitals and the government wants to construct n more hospitals. Thus, we get the next problem.

Problem 7 For given convex regions T, number n, and k points $P_{n+1}, \ldots, P_{n+k} \in T$, find the locations of n points $P_1 = P_1^*, \ldots, P_n = P_n^*$ that attain

$$\min_{P_1,\ldots,P_n \in T} \max_{P \in T} \min_{i \in \{1,\ldots,n+k\}} d(P, P_i). \tag{10}$$

Still another situation is the following.

Let μ (P) denote the population density at P in the city T, and let $d(P, P_i)$ be the cost for an inhabitant at P to go to the hospital at P_i. Assume that all the citizens need medical care in equal probability, and the government wants to build n hospitals in such a way that the total cost for the citizens to go to the hospitals is minimized. Thus we get the next problem.

Problem 8 For given T, μ, n, find the locations of n points $P_1 = P_1^*, \ldots, P_n = P_n^*$ that attain

$$\min_{P_1,\ldots,P_n \in T} \int_T \mu(P) \min_{i \in \{1,\ldots,n\}} d(P, P_i)\, dT. \tag{11}$$

Note that Expressions (9), (10), (11) have the same form:

$$\min_{P_1,\ldots,P_n \in T} F(P_1, \ldots, P_n). \tag{12}$$

However the objective function

$$F(P_1, \ldots, P_n)$$

is nonconvex, and the three problems are in general difficult to solve strictly. However, we can find an approximate solution using Voronoi diagrams in the following way.

As we have seen, $F(P_1, \ldots, P_n)$ in Problems 6 and 7 can be obtained by solving the largest empty-circle problem and hence the Voronoi diagram associated with the distance d can be used. Moreover, (11) is rewritten to

$$\min_{P_1,\ldots,P_n \in T} \sum_{i=1}^{n} \int_{R(S;P_i)\cap T} \mu(P) d(P, P_i)\, dT, \tag{13}$$

where $R(S;P_i)$ denotes the Voronoi region of P_i in the associated Voronoi diagram. Hence, the objective function $F(P_1, \ldots, P_n)$ in Problem 8 can also be computed via the Voronoi diagram.

Thus, we have the following iterative strategy to solve Problems 6, 7 and 8 approximately. First, we choose $P_1, \ldots, P_n$ in T arbitrarily. Next we construct the Voronoi diagram for $\{P_1, \ldots, P_n\}$ with respect to the given distance d, and compute the value of $F(P_1, \ldots, P_n)$ together with $\partial F/\partial x_i$, $\partial F/\partial y_i$ $(i = 1, \ldots, n)$. Then, we move $P_1, \ldots, P_n$ in the direction that decreases the objective function. We repeat this until $F(P_1, \ldots, P_n)$ converges.

The detailed descriptions of the strategies and their experimental evaluations for individual types of the problems can be found in [4,8,12]. For the fast automatic differentiation method to get $\partial F/\partial x_i$ and $\partial F/\partial y_i$, refer to [5].

Concluding Remarks

We have seen typical problems in facility location for which the concept of the Voronoi diagram and its generalization are useful. There are many other variants in facility location problems. For the details, also refer to other surveys [7,9].

See also

- ▶ Combinatorial Optimization Algorithms in Resource Allocation Problems
- ▶ Competitive Facility Location
- ▶ Facility Location with Externalities
- ▶ Facility Location Problems with Spatial Interaction
- ▶ Facility Location with Staircase Costs
- ▶ Global Optimization in Weber's Problem with Attraction and Repulsion
- ▶ MINLP: Application in Facility Location-allocation
- ▶ Multifacility and Restricted Location Problems
- ▶ Network Location: Covering Problems
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Production-distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Stochastic Transportation and Location Problems
- ▶ Warehouse Location Problem

References

1. Aurenhammer F (1991) Voronoi diagram: A survey of a fundamental geometric data structure. ACM Computing Surveys 23:345–405
2. Edelsbrunner H (1987) Algorithms in combinatorial geometry. Springer, Berlin
3. Fortune S (1987) A sweepline algorithm for Voronoi diagrams. Algorithmica 2:153–174
4. Iri M, Murota K, Ohya T (1984) A fast Voronoi diagram algorithm with applications to geographical optimization problems. In: Thoft-Christensen P (ed) Proc. IFIP Conf. System Modelling and Optimization (1983, Copenhagen), Lecture Notes Control Inform Sci. Springer, Berlin, 273–288
5. Kubota K, Iri M (1991) Estimates of rounding errors with fast automatic differentiation and interval analysis. J Inform Process 14:508–515
6. Ohya T, Iri M, Murota K (1984) Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms. J Oper Res Soc Japan 27:306–336
7. Okabe A, Boots B, Sugihara K, Chui SN (2000) Spatial tessellations: Concepts and applications of Voronoi diagrams. Wiley, New York
8. Okabe A, Suzuki A (1987) Stability of spatial competition for a large number of firms on a bounded two-dimensional space. Environm Plan A 19:1067–1082

9. Okabe A, Suzuki A (1997) Locational optimization problems solved through Voronoi diagrams. Europ J Oper Res 98:445–456
10. Preparata FP, Shamos MI (1985) Computational geometry: An introduction. Springer, Berlin
11. Sugihara K, Iri M (1994) A robust topology-oriented incremental algorithm for Voronoi diagrams. Internat J Comput Geom Appl 4:179–228
12. Suzuki A, Drezner Z (1996) 'The p-center location problem in an area. Location Sci 4:69–82

W

Walrasian Price Equilibrium

WPE

Anna Nagurney
University Massachusetts, Amherst, USA

MSC2000: 91B50

Article Outline

Keywords
The Iterative Scheme
The Projection Method
The Relaxation Method
See also
References

Keywords

Pure exchange; Pure trade; General economic equilibrium; Perfect competition; Walras law; Aggregate excess demand function; Variational inequality formulation; Projection method; Relaxation method

The *Walrasian price* or *pure exchange* equilibrium problem is a *general* as opposed to *partial* equilibrium problem in that all commodities in the economy are treated. In addition, it is an example of *perfect* competition in that it is assumed that producers take the prices as given and can not individually influence the prices. This problem has been extensively studied in the economics literature dating to L. Walras [20]; see also [2,4,10,19].

In the pure exchange model the consumer side of the economy is modeled by the excess demand functions and it is assumed that production is absent and consumers exchange commodities that they initially own. Production can be introduced into this basic framework in a variety of ways by including, for example, an activity analysis model to describe the productive techniques in the economy (cf. [14,15]). The excess demand functions are aggregated demand functions over the individual consumers in the economy. They represent the difference between the market demands for the commodities and the supply of the commodities (based on the initial endowments of the consumers).

Computation of economic equilibria has been, typically, based either on classical algorithms for solving nonlinear systems of equations (see, e. g., [6]), or, on simplicial approximation methods pioneered by H. Scarf [15] (see also the contributions of M.J. Todd [17,18], J.B. Shoven [16], J. Whalley [21], G. van der Laan and A.J.J. Talman [7]). The former techniques are applicable only when the equilibrium lies in the interior of the feasible set, while the latter techniques are general-purpose algorithms, and are capable of handling inequality constraints, such as the requirement that the prices of the commodities be nonnegative. Nevertheless, in their present state of development, they may be unable to handle large scale problems (cf. [11]).

General economic equilibrium problems have been formulated as nonlinear complementarity problems (see, e. g., [8]) and a Newton-type method based on this formulation has been used by many researchers for the computation of equilibria (see, e. g., [5,9,13]). Although this approach has been proven to be more effective than fixed point methods, its convergence has not been proven theoretically (see, e. g., [11]).

K.C. Border [1] provides a variational inequality formulation of Walrasian price equilibrium. Qualitative results using a variational inequality framework can be

found in [3]. Algorithms, as well as numerical examples, as well as the relationship of the Walrasian price equilibrium problem to a network equilibrium problem can be found in [23].

Here we present the *pure exchange* (or *pure trade*) economic equilibrium model and gives its variational inequality formulation. Some fundamental theoretical results are then presented. For proofs and additional background and results, see [3,12,22], and [23].

Consider a *pure exchange* economy with l commodities, and with column price vector p taking values in $\mathbf{R}^l_+$ and with components $p_1, \ldots, p_l$. Denote the induced aggregate excess demand function $z(p)$, which is a row vector with components $z_1(p), \ldots, z_l(p)$. Assume that $z(p)$ is generally defined on a subcone C of $\mathbf{R}^l_+$ which contains the interior $\mathbf{R}^l_{++}$ of $\mathbf{R}^l_+$, that is, $\mathbf{R}_{++} \subset C \subset \mathbf{R}^l_+$. Hence, the possibility that the aggregate excess demand function may become unbounded when the price of a certain commodity vanishes is allowed. Assume that $z(p)$ satisfies *Walras' law*, that is, $\langle z(p), p\langle = 0$ on C and that $z(p)$ is *homogeneous of degree zero* in p on C, that is, $z(\alpha p) = z(p)$ for all $p \in C$, $\alpha > 0$. Because of homogeneity, one may normalize the prices so that they take values in the simplex:

$$S^l = \left\{ p:\ p \in \mathbf{R}^l_+, \sum_{i=1}^{l} p_i = 1 \right\},$$

and, therefore, one may restrict the aggregate excess demand function to the intersection D on S^l with C. Let

$$S^l_+ = \left\{ p:\ p > 0, p \in S^l \right\},$$

and note that $S^l_+ \subset D \subset S^l$.

As is standard in general economic equilibrium theory, assume that:

i) the function $z(p)$: $D \Rightarrow \mathbf{R}^l$ is continuous;

ii) the function $z(p)$ satisfies Walras' law:

$$\langle z(p), p\rangle = 0, \quad \forall p \in D.$$

The definition of a Walrasian equilibrium is now stated.

Theorem 1 (*Walrasian price equilibrium*) *A price vector $p^* \in \boldsymbol{R}^l_+$ is a Walrasian equilibrium price vector if*

$$z(p^*) \leq 0.$$

The following theorem establishes that Walrasian price vectors can be characterized as solutions of a variational inequality.

Theorem 2 (*variational inequality formulation*) *A price vector $p^* \in D$ is a Walrasian equilibrium if and only if its satisfies the variational inequality*

$$\langle z(p^*), p - p^*\rangle \leq 0, \quad \forall p \in S^l.$$

The interpretation in the above variational inequality model geometrically is that $z(p^*)$ is 'orthogonal' to the set S^l and points away from the set S^l. In particular, the result is the following:

Proposition 3 *A price vector p^* is a Walrasian equilibrium, or, equivalently, a solution of the above variational inequality, if and only if, it is a fixed point of the projection map*

$$G(p) = P_{Sl}(p + \rho z(p)),$$

where $\rho > 0$ and P_{S^l} indicates the projection map onto the compact convex set S^l.

Note that if the aggregate excess demand function $z(p)$ is defined and is continuous on all of S^l, that is, $D = S^l$, then the existence of at least one Walrasian equilibrium price vector in S^l follows immediately from the standard theory of variational inequalities.

However, since D is not necessarily compact, one may still be able to deduce that $z(p)$ exhibits the needed behavior near the boundary of S^l, in particular, that at least some of the components of $z(p)$ become in a sense 'large' as p approaches points on the boundary of S^l that are not contained in D. Several existence proofs of this type can be found in [1]. We now provide the result proven in [3]:

Theorem 4 *Assume that the aggregate excess demand function $z(p)$ satisfies the following assumption: If $S^l \setminus D$ is nonempty, then with any sequence $\{p_n\}$ in S^l_+ which converges to a point of $S^l \setminus D$ there is associated a point $\overline{p} \in S^l_+$, generally dependent on $\{p_n\}$, such that the sequence $z(p_n) \cdot \overline{p}$ contains infinitely many positive terms.*

Then there exists a Walrasian equilibrium price vector $p^* \in D$.

A special class of aggregate excess demand functions is now considered, for which the following result holds true:

Theorem 5 *Assume that $-z(p)$ is continuous and monotone on D. Then $p^*; \in D$ is a Walrasian equilibrium price vector if and only if*

$$\langle z(p), p - p* \rangle \leq 0, \quad \forall p \in D,$$

or, equivalently, if and only if,

$$\langle z(p), p* \rangle \geq 0, \quad \forall p \in D.$$

An immediate consequence of the above is the following:

Corollary 6 *Assume that $-z(p)$ is continuous and monotone on D and D is compact. Then the set of Walrasian equilibrium price vectors is a convex subset of D.*

The uniqueness issue is now investigated; specifically, if one strengthens the monotonicity assumption somewhat, one obtains the following result.

Theorem 7 *Assume that $-z(p)$ is strictly monotone on D, that is,*

$$\langle z(p^1) - z(p^2), p^1 - p^2 \rangle < 0, \quad \forall p^1, p^2 \in D, p^1 \neq p^2.$$

Then there exists at most a single Walrasian price equilibrium vector p^*.

We now present a general iterative scheme for the computation of Walrasian price equilibria is described. The scheme is based on the general iterative scheme of S.C. Dafermos (cf. [12], and the references therein).

In the study of algorithms and their convergence, the standard assumption in the economics literature (cf. [15]) is that the aggregate excess demand function $z(p)$ is well-defined and continuous on all of S^l. Here this assumption is also made. The scheme is as follows.

The Iterative Scheme

Construct a smooth function $g(p, q)$: $S^l \times S^l \Rightarrow \mathbf{R}^l$ with the following properties:

i) $g(p, p) = -z(p)$, $\forall p \in S^l$;
ii) for every $p, q \in S^l$, the $(l \times l)$-matrix $\nabla_p g(p, q)$ is positive definite.

Any smooth function $g(p, q)$ with the above properties generates the following algorithm:

0	*Initialization*: Start with some $p^0 \in S^l$. Set $k := 1$.
1	*Construction and computation*: Compute p^k by solving the variational inequality $\langle g(p^k, p^{k-1})^\top, p - p^k \rangle \geq 0, \quad \forall p \in S^l.$
2	*Convergence verification*: IF $\mid p^k - p^{k-1} \mid \leq \epsilon$, with $\epsilon > 0$, a prespecified tolerance, THEN STOP; ELSE, set $k := k + 1$, and go to Step 1.

For simplicity, we denote the above variational inequality by $VI^k(g, S^l)$. Since $\nabla_p g(p, q)$ is positive definite, $VI^k(g, S^l)$ admits a unique solution p^k. Thus, we obtain a well-defined sequence $\{p^k\}$. It is easy to verify that if the sequence $\{p^k\}$ is convergent, say $p^k \to p^*$, as $k \to \infty$, then p^* is an equilibrium price vector, that is, it is a solution of the variational inequality. In fact, on account of the continuity of $g(p, q)$, $VI^k(g, S^l)$ yields

$$\langle -z(p^*), p - p^* \rangle = \langle g(p^*, p^*)^\top, p - p^* \rangle = \lim_{k \to \infty} \langle g(p^k, p^{k-1})^\top, p - p^k \rangle \geq 0, \quad \forall p \in S^l,$$

so that p^* is a solution of the original variational inequality.

Conditions for convergence may be found in [12] and [23].

We now show that the general iterative scheme induces a projection method and a relaxation method. In the context of the pure exchange model both the projection method and the relaxation method resolve the variational inequality problem into simpler subproblems, which can then be solved using equilibration algorithms (cf. [23]).

The Projection Method

The projection method corresponds to the choice:

$$g(p, q) = -z(q) + \frac{1}{\rho} G(p - q),$$

where ρ is a positive scalar and G is a fixed, symmetric positive definite matrix. In this case properties i) and ii) are satisfied. In fact,

i) $g(p, q) = -z(p) + \frac{1}{\rho}G(p - p) = -z(p)$
ii) $\nabla_p\, g(p, q) = \rho^{-1}G$, is positive definite and symmetric.

The Relaxation Method

The relaxation method corresponds to the choice:

$$g_i(p, q) = -z_i(q_1, \ldots, q_{i-1}, p_i, q_{i+1}, \ldots, q_l),$$
$$\forall i = 1, 2, \ldots, l.$$

In this case properties i) and ii) are also satisfied. In fact,
i) $g(p, p) = -z(p)$,
ii)

$$\nabla_p g(p, q) = \begin{pmatrix} -\frac{\partial z_1}{\partial p_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -\frac{\partial z_l}{\partial p_l} \end{pmatrix}$$

is a diagonal matrix.

See also

- ▶ Equilibrium Networks
- ▶ Financial Equilibrium
- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Oligopolistic Market Equilibrium
- ▶ Spatial Price Equilibrium
- ▶ Traffic Network Equilibrium

References

1. Border KC (1985) Fixed point theorems with application to economics and game theory. Cambridge Univ. Press, Cambridge
2. Cornwell R (1984) Introduction to the use of general equilibrium analysis. North-Holland, Amsterdam
3. Dafermos S (1990) Exchange price equilibria and variational inequalities. Math Program 46:391–402
4. Debreu G (1959) The theory of value. Wiley, New York
5. Eaves BC (1983) Where solving for stationary points by LCPs is mixing Newton iterates. In: Eaves BC, Gould FJ, Peitgen HO, Todd MJ (eds) Homotopy Methods and Global Convergence. Plenum, New York
6. Ginsburgh V, Waelbrock JL (1981) Activity analysis and general equilibrium modelling. In: Contributions to Economic Analysis, vol 125. North-Holland, Amsterdam
7. van der Laan G, Talman AJJ (1987) Simplicial approximation of solutions to the nonlinear complementarity problem with lower and upper bounds. Math Program 38:1–15
8. Manne AS (1985) On the formulation and solution of economic equilibrium models. Math Program Stud 23:1–22
9. Manne AS, Preckel PV (1985) A three-region intertemporal model of energy, international trade, and capital flows. Math Program Stud 23:56–74
10. Mas-Colell A (1985) The theory of general economic equilibrium: A differentiable approach. In: Econometric Soc Monographs, vol 9. Cambridge Univ. Press, Cambridge
11. Mathiesen L (1987) An algorithm based on a sequence of linear complementarity problems applied to a Walrasian equilibrium model: An example. Math Program 37:1–18
12. Nagurney A (1999) Network economics: A variational inequality approach, 2nd edn. Kluwer, Dordrecht
13. Rutherford T (1987) A modeling system for applied general equilibrium analysis. Cowles Foundation Discussion Paper 836
14. Scarf H (1982) The computation of equilibrium prices: An exposition. In: Arrow KJ, Intriligator MD (eds) Handbook Math. Economics, vol 3. North-Holland, Amsterdam, pp 1007–1061
15. Scarf H, Hansen T (1973) Computation of economic equilibria. Yale Univ. Press, New Haven, CT
16. Shoven JB (1983) The application of fixed point methods to economics. In: Eaves BCGould FJPeitgen HO, Todd MJ (eds) Homotopy Methods and Global Convergence, Plenum, New York, pp 249–262
17. Todd MJ (1976) The computation of fixed points and applications. Lecture Notes Economics and Math Systems, vol 124. Springer, Berlin
18. Todd MJ (1979) A note on computing equilibria in economics with activity models of production. J Math Economics 6:135–144
19. Wald A (1951) On some systems of equations in mathematical economics. Econometrica 19:368–403
20. Walras L (1874) Elements d'economique politique pure, economique politique pure. Corbaz
21. Whalley J (1977) Fiscal harmonization in the EEC: Some preliminary findings of fixed point calculations. Fixed Points: Algorithms and Appl. Acad. Press, New York, pp 435–472
22. Zhao L, Dafermos S (1991) General economic equilibrium and variational inequalities. Oper Res Lett 10:369–376
23. Zhao L, Nagurney A (1993) A network framework for general economic equilibrium. In: Du D-Z, Pardalos PM (eds) Network Optimization Problems: Algorithms, Complexity, and Applications. World Sci., Singapore, pp 363–386

Warehouse Location Problem

JAKOB KRARUP
DIKU Universitetsparken 1, Copenhagen, Denmark

MSC2000: 90B80, 90B85

Article Outline

Keywords
Three Prototype Location Problems
p-MP
p-CP
SPLP
See also
References

Keywords

Prototype location problem; p-median problem; p-center problem; Simple plant location problem; Uncapacitated plant location problem; Discrete single-commodity single-criterion uncapacitated static multifacility

Within the broad interface between computer science and operational research (OR) a major application area is *locational decisions*. Roughly speaking, one can claim that the location of any physical object whatsoever, partially or totally created by some living organism, represents the solution to a location problem. Even if we restrict 'living organism' to encompass human beings only and even if we assume some kind of 'intellectual process' behind the choice of a solution, the entire field of location problems is still overwhelming and its story dates back as far as the story of mankind itself. Further limitations are necessary in order to find a suitable framework for the pesentation of the subject, and the next subset of location problems emerges naturally if the rather vague phrase 'intellectual process' is replaced by some systematical approach based on what has become commonly accepted as *OR-methodology*. This brings *models* in the focal point as a convenient tool for locational analyses.

A substantial proportion of any developed country's gross national product (GNP) is spent simply on 'moving things around'. The location of the *facilities* (such as town halls, hospitals, factories, depots, retailers, supermarkets, or components in electrical circuits, etc.) in relation to the *customers* or other facilities is therefore of crucial importance to the success of both private or public enterprises or to the outcome of the nowadays computer-monitored military operations.

Several schemes have been proposed for classifying the wealth of models developed for locational decisions. To account for all the factors separating such models from one another is far beyond the scope of the present article. For our purpose, however, we need to emphasize the distinction between *continuous* and *discrete* models. Continuous models typically presume that the facilities to be located can be placed *anywhere*, that is, within the context of a continuous space, for example, in a plane. Discrete models, on the other hand, deal with situations where the set of potential sites for the facilities to be placed is finite and often represented by the vertices of a network.

Single-commodity models as opposed to *multicommodity* models deal with the location of one or more facilities each providing the *same* kind of service to the customers allocated to it. For such models, the *weight of a customer* represents the amount demanded (per time unit) for the kind of commodity supplied by the facilities. It may well occur that all customers can be viewed as having the same demand. If the so-called *single assignment property* applies, which means that each customer is supplied by a single facility only, we can then transform the original data of the problem such that all customers have *unity demand*, or weight equal to 1. Problems where the weight associated with each customer equals 1 are characterized as *unweighted* as opposed to *weighted*.

As its name indicates, *single-facility problems* involves the placement of a single facility only as opposed to *multifacility models*. Likewise, *single-criterion models* consider a single criterion only when the quality of feasible solutions is assessed. In contrast to *multicriteria decision making*, it is here meaningful to talk of an *objective function* which is to be optimized.

How should a feasible solution comprising both location and allocation be assessed? Among the most frequently encountered constituents of an objective function are: the overall *cost structure* and the *distance measure* employed.

To exemplify some of the notions introduced above, we can consider the location of a regional wine depot which is to supply a chain of n supermarkets located in n towns. Disregarding the fact that wine bottles indeed may contain highly different fluids, we shall assume that the yearly demands can be expressed in terms of the total number of wine cases. The problem thereby reduces to a single-commodity problem.

It is rather unlikely that the regional depot to be located can be placed 'anywhere'. A discrete model is thus more appropriate in this case; let us say that the set of potential location sites is a subset of the vertices in the road network connecting the n cities. Now, will the trucks visit only a single supermarket at a time or a sequence of supermarkets before it returns to the depot? For simplicity, let us disregard the *routing* aspect (how to combine location with the design of routes visiting two or more customers in some order) and accordingly assume that the objective function to be minimized can be expressed as the product sum

$$\sum (\text{annual demand}) \times (\text{shortest path distance between supermarket and depot}),$$

where the summation is taken over the n supermarkets.

This model representing a very simplified picture of the'real world' is known in the literature as the *1-median problem in a network* or 1-MP for short.

Admittedly, the inclusion of other factors would contribute a lot to the model's realism. Examples of such factors are: seasonal variations around Christmas and other peak events, the number and capacities of the trucks used, the distance measure employed (road lengths or travel time?), the fixed costs which may vary from one potential location site to another, et cetera. Nevertheless, it is a well-documented fact that fairly simple models often are able to capture the essential parts of a realistic problem whereas the additional contributions to cost savings or profit achieved via more complex models, viewed against the time and cost invested in their development, need not give 'value for money'.

A multifacility location problem arises, as mentioned above, when two or more facilities are to be located simultaneously, each interacting with the customers or existing facilities or with each other. Suppose for example that the chain of supermarkets may wish to expand its activities by entering a new region. How many new supermarkets should be opened and where should they be located? Should they all necessarily provide the same kind of service to their customers, say, in terms of assortment? Considering the *competitive environment* we here are dealing with, what will the expected market share become? Should the supermarkets be opened in different time periods and, if yes, in which order?

The last question addressed does also relate to the distinction between *static* and *dynamic* models, where the latter explicitly include time and thus can be viewed as examples of *multiperiod planning*. The usual aim of such dynamic models would be to investigate how best to incorporate additional new facilities in an existing structure, to rearrange an existing layout, or to plan a completely new system.

The facilities to be located are normally regarded as 'friendly' in the sense that 'closeness' is viewed as an attractive property. For example, real estate dealers praise easy access to shopping centers, schools, public transportation, and recreational areas when a house is announced for sale. Locational decisions, however, do also encompass the counterpart: the location of so-called 'obnoxious' facilities like nuclear power plants, shooting ranges, and polluting factories which are needed for the society although they produce an undesirable effect or represent a threat to their surroundings. Here, one frequently used criterion is the maximum distance between a facility and the closest customer. It should in this context be noted that even a friendly facility may well become obnoxious unless 'closeness' is taken with a grain of salt. Thus, optimal closeness to a noisy elementary school is rather 'reachable within a few minutes' walk' than 'next door', a feature known as the *NIMBY syndrome* (not in my back yard).

The investigation of models with such truly antagonistic criteria capturing both the friendly and the obnoxious aspect of a locational decision problem have attracted several researchers, notably in the 1990's, and the field is still gaining further momentum. 'Semiobnoxious' is among the new adjectives created. Whereas a nuclear power plant indeed is being considered as obnoxious by the vast majority of people, a typical semiobnoxious facility could be an airport, disliked by its neighbors for its environmental pollution and appreciated by its users for its reachability.

Three Prototype Location Problems

When is a discrete model more appropriate than a continuous one? Both options are frequently available to practitioners and the following issues are often crucial when a choice is to be made:

a) Is the transportation network so well developed in the region being considered and so free from barriers that a continuous formulation is reasonable?
b) Is there a relatively small set of identifiable facility sites so that a discrete formulation should be advocated?
c) Are the optimal solutions to a continuous model readily transferable to a set of possible locations without resulting in serious errors in the measure(s) of performance used to evaluate solutions?
d) Do either of the two model types offer computational advantages?

Although the answers to such questions may be ambiguous, and although the analyst will often have considerable flexibility in her/his choice, experience from practice indicates that these answers most often lead to the choice of a discrete model. The major reasons are that in most cases decision-makers consider a discrete representation to be a more realistic and a more accurate portrayal of the problem at hand, and that continuous models appear to be relatively difficult to solve.

Among the myriads of models considered in discrete location theory, only three of these: the *p-median problem* p-MP, the *p-center problem* (p-CP), and the *simple plant location problem* (SPLP) — at times referred to as *prototype location problems* — have played a particularly dominant role. Despite the seeming simplicity of their underlying assumptions, these models have provided important, quantitative bases for the investigation of numerous practical locational decision problems. They have been used both as optimization models in their own right or have been employed as subroutines in more integrated models. Finally, due to the large number of extensions available, each of these three prototype problems can be viewed as the foremost member of a family of location problems.

We now present p-MP, p-CP, and SPLP in their most general forms and provide concise, symbolic formulations of each of these within a common framework.

Let

- m be the finite number of customers, indexed by $i \in I = \{1, \ldots, m\}$;
- n be the finite number of sites for potential facilities, indexed by $j \in J = \{1, \ldots, n\}$;
- p be the number of facilities to be opened or established, $1 \leq p \leq n$.

Whereas the locations of the p facilities to be established is to be decided upon, the locations of the m customers are assumed known and invariant. These customers have prespecified demands for a common good which in principle can be provided by any potential facility.

For each of the mn facility-customer pairs, define

- c_{ij} as the total variable cost of serving all of customer i's demands from facility j.

The 'cost' c_{ij} may include measures of the distance from customer i to facility j as well as of the time or cost of serving customer i from facility j. For example, the c_{ij} may be interpreted as $c_{ij} = w_i(h_j + t_{ij})$ where

- w_i is the number of units demanded by customer i;
- h_j is the per unit cost of operating facility j (including variable production and administrative costs, etc.), and
- t_{ij} is the transportation cost of shipping one unit to customer j from facility i.

Cost t_{ij} may also be interpreted as $t_{ij} = d_{ij}$ where

- d_{ij} is the physical distance (or its time or monetary equivalent) of a shortest path from customer i to facility j

Then, for $h_j = 0$, c_{ij} reduces to $c_{ij} = w_i d_{ij}$. Thus, c_{ij} captures the various notions of distance, time, and variable costs referred to so far.

With c_{ij} so defined, we can without loss of generality assume all customers to have unity demand. Furthermore, for each of the three problems, no *capacity constraints* are imposed on the number of costumers that each potential facility can serve. Finally, also without loss of generality, all data are assumed nonnegative.

As will be explained shortly, we can conveniently express the locational decisions to be made in terms of Q:

- $Q \subseteq J$ is a subset of potential facilities to be opened and from which all customers are to be served. The cardinality of Q is denoted by $|Q|$.

Conceptually, an approach to identify an optimal solution to each of the three problems can be said to involve two phases,

1) determination of a *location pattern* in terms of Q specifying the location of the facilities, and
2) an *allocation phase* in which each customer is assigned to exactly one open facility and hence is assumed to receive all of its demand from that facility such that a certain objective function is minimized.

Computationally, however, these two phases cannot in general be separated from one another but may, depending on how a specific algorithm is designed, be carried out simultaneously. The conceptual decomposition into two phases is here suggested solely to facilitate comprehension of the ensuing compact formulations.

p-MP

Data instance: m, n, p, $C = \{c_{ij}\}$.

Open p facilities and assign each customer to exactly one of them such that the total variable cost is minimized. For Q given, an assignment minimizing total variable cost can be determined 'by inspection': customer i is assigned to an established facility corresponding to the smallest c_{ij} (up to ties), that is, to facility k where $c_{ik} = \min_{j \in Q} c_{ij}$. Upon assigning all customers in this manner, the resulting total variable cost becomes $\sum_{i \in I}\{\min_{j \in Q} c_{ij}\}$. Hence, p–MP reads:

$$p\text{—MP}: \min_{Q \subseteq J,\ |Q|=p} \left\{ \sum_{i \in I} \min_{j \in Q} c_{ij} \right\}$$

The *p-center problem* p-CP differs significantly from p-MP in several respects, primarily with respect to the criterion used for assessing the quality of a feasible solution. Whereas 1-MP as exemplified above by the wine depot location problem and the more general p-MP are *minisum problems*, p-CP has a *minimax objective*: open p facilities and assign each customer to exactly one of them such that the maximum distance (unweighted case) or the maximum weighted distance from any open facility to any of the customers assigned to it is a minimum.

p-CP is often a suitable model for analyzing locational decision problems for emergency services such as police, fire, and ambulance services. A common criterion for the effectiveness of such service coverage is that any demand point may be reached from the facility nearest it within a given weighted distance, time or cost.

p-CP

Data instance: m, n, p, $C = \{c_{ij}\}$.

Open p facilities and assign each customer to exactly one of them such that the maximum variable cost of serving any customer is a minimum. Suppose Q is known. We can then do no better than assigning the ith customer to that open facility from which the cost c_{ij} is a minimum, that is, to facility k where $c_{ik} = \min_{j \in Q} c_{ij}$ and where ties are resolved arbitrarily. Upon assigning all customers in this manner, the resulting maximum cost becomes $\max_{i \in I}\{\min_{j \in Q} c_{ij}\}$. Hence, the following formulation obtains:

$$p\text{—CP}: \min_{Q \subseteq J,|Q|=p} \max_{i \in I} \left\{ \min_{j \in Q} c_{ij} \right\}.$$

Like p-MP, also the third prototype location problem, the *simple plant location problem* (SPLP) is a minisum problem. Two features, however, separate p-MP from SPLP:

a) the inclusion of fixed costs associated with each potential facility, and
b) the number of facilities to be established which no longer is prespecified but results from an optimal solution.

For the jth potential facility define

- f_j as the fixed cost of establishing facility j

'Fixed' means that f_j is to be paid only if facility j actually is established and f_j is then independent of the number of customers (≥ 1) served by that facility.

SPLP

Data instance: m, n, $C = \{c_{ij}\}$, $f = (f_j)$.

Open a subset $Q \subseteq J$ of facilities and assign each customer to exactly one of them such that the sum of the fixed and the variable costs is minimized, that is,

$$\text{SPLP}: \min_{Q \subseteq J} \left\{ \sum_{j \in Q} f_j + \sum_{i \in I} \min_{j \in Q} c_{ij} \right\}.$$

We note in passing, that while most well-defined problems bear unambiguous names, SPLP has been dealt with in the literature under a wide variety of different titles, usually composed of an adjective (simple, uncapacitated, optimal) and a noun (plant, warehouse, facility, site) followed by *location problem*. It is furthermore somewhat confusing that 'simple' in this context is synonymous with 'uncapacitated' since also p-MP and p-CP assume that the facilities to be located have unlimited capacities.

This ultra-short sketch of the 'nature of locational decisions' does hardly reveal even the tip of the ice-

berg though hopefully enough to leave an impression of an area of great practical importance not to forget the wealth of theoretical challenges and open questions that still remain.

The literature is already huge and rapidly growing. Among pertinent 'broad-coverage' references are the three textbooks [1,3,4]. Also, [2] deserves to be mentioned. The idea is here to consider decisions as regards location and design of production facilities as being interrelated, that is, the optimal plant design (input mix and output level) depends on the location of the plant, and the optimal location of the plant depends on its design.

See also

- Combinatorial Optimization Algorithms in Resource Allocation Problems
- Competitive Facility Location
- Facility Location with Externalities
- Facility Location Problems with Spatial Interaction
- Facility Location with Staircase Costs
- Global Optimization in Weber's Problem with Attraction and Repulsion
- MINLP: Application in Facility Location-allocation
- Multifacility and Restricted Location Problems
- Network Location: Covering Problems
- Optimizing Facility Location with Rectilinear Distances
- Production-distribution System Design Problem
- Resource Allocation for Epidemic Control
- Single Facility Location: Circle Covering Problem
- Single Facility Location: Multi-objective Euclidean Distance Location
- Single Facility Location: Multi-objective Rectilinear Distance Location
- Stochastic Transportation and Location Problems
- Voronoi Diagrams in Facility Location

References

1. Drezner Z (ed) (1995) Facility location. Springer, Berlin
2. Hurter AP, Martinich JS (1989) Facility location and the theory of production. Kluwer, Dordrecht
3. Love RF, Morris JG, Wesolowsky GO (1988) Facilities location. North-Holland, Amsterdam
4. Mirchandani PB, Francis RL (eds) (1990) Discrete location theory. Wiley/Interscience, New York

Wastewater System, Optimization of

MAGDALENE MARINAKI
Department of Production Engineering and Management, Industrial Systems Control Laboratory, Technical University of Crete, Chania, Greece

MSC2000: 76D55

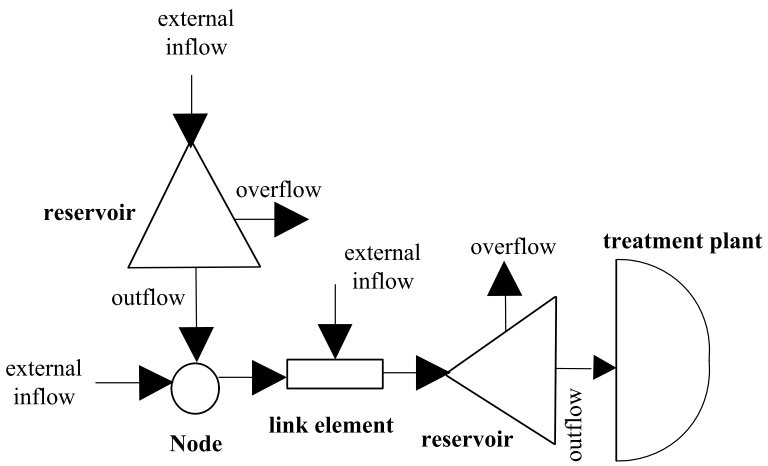
Article Outline

Introduction
Optimization of Wastewater Systems
References

Introduction

In recent decades an increased interest in protecting the environment from everything that could lead to its degradation and destruction has been observed. Pollution (discharge of materials or energy and discharge of microorganisms that are pathogenic for people and animals) of the groundwater and underwater is one of the most important problems facing ordinary people and authorities around the world. Many ecological consequences result from groundwater pollution. For example, the physicochemical characteristics of water are changed, leading to severe economic consequences for people, e. g., an increase in the cost of water processing for its reuse.

The most important problems surrounding pollution concern water (lakes, rivers, and oceans), which suffers the strongest exploitation and use. One of these uses is as receivers of the outflows of combined sewer networks [10,26,55]. The construction of treatment plants, to enable sewage treatment before disposal in a body of water, protects the quality of the water that receives the outflows of the sewage networks. However, urban combined sewer networks do not have separate collectors for domestic and industrial sewage and rainwater drainage. Therefore, during rainfall, networks and/or treatment plants may be overloaded, and overflows may take place upstream of overloaded stretches, causing the pollution of receiving waters. Placing retention reservoirs at appropriate locations along the network (by constructing special basins (offline storage) or by installing throttle gates at the end of long sewer stretches (in-line storage)) is a cost-efficient way



Wastewater System, Optimization of, Figure 1
Schematic representation of a small sewer network

to avoid overflows during moderate rainfall events and to reduce them in stronger rainfall as the water is stored in the reservoirs during the rainfall and is directed toward the treatment plant after the rain has stopped.

Optimal operation of the combined sewer network (which contains retention reservoirs) (Fig. 1) implies that for each rain event the whole retention capacity of all reservoirs will be used before overflows take place somewhere in the network. This, however, cannot be guaranteed by fixed gate settings, such as fixed weirs or manually adjustable gates for the filling and emptying of these storage spaces. Especially if the rainfall is distributed unevenly over an urban area, there may be reservoirs that are not totally filled, while overflows already occur elsewhere in the network. In these cases, a further considerable reduction of overflows can be obtained by real-time operation of the reservoirs, e. g., by use of controllable gates. The decision on how to move the gates during a certain rain event may be made by a human operator or by some automatic control strategy to be applied in real time. An efficient control strategy can reduce substantially the overflows from a sewer network. In addition, it may lead to substantial cost savings as the number and storage capacities of the reservoirs required to keep overflows below a certain (usually legislatively defined [22,60]) limit depends upon the efficiency of the applied control strategy.

Optimization of Wastewater Systems

The development of a control system for combined sewer networks has as a goal the protection of the quality of waters that receive the outflows of the networks. Thus, the main task of the control system is the minimization of overflows for any rainfall event. The development of optimization techniques for the planning, design, and management of complex water resource systems has been the subject of many investigations [40] around the world. The choice of optimization method to be used depends on the characteristics of the reservoir system being considered, on the availability of data, and on the specific control objectives and constraints. Many researchers in the field have considered methods such as linear programming, dynamic programming, nonlinear programming, linear-quadratic control, genetic algorithms, and combinations of these methods. Nonlinear optimal control is the most efficient approach due to direct consideration of inflow predictions, process nonlinearities, and constraints. On the other hand, nonlinear optimal control implies development and implementation of sophisticated codes for the real-time numerical solution of the optimal control problem. Multivariable regulators, if designed properly, may approximate the efficiency of nonlinear optimal control, based on much simpler calculation instructions. The approaches that are based on dynamic programming are difficult to ap-

ply to large-scale networks due to the "curse of dimensionality," while approaches based on linear programming do not include the nonlinearities of the process. Expert systems, fuzzy control, and further heuristic approaches have also been applied with remarkable results.

Linear programming is a very powerful and easy-to-use form of optimization and is most efficient for problems that can be expressed in linear terms. For sewer network control, linear programming is used in [4] for the development of a control algorithm for automatic control of detention storage in a large-scale combined sewer system and in [42] for the real-time control of urban drainage systems where the nonlinear programming problem is replaced by a succession of linear programming problems. In [1] the optimization of the discharge hydrograph of a pumping station located at the downstream end of a storm drainage channel located in the southeastern portion of Mexico City is considered and the initial nonlinear optimization problem is reduced to a series of linear programming problems whose solution determines the desired optimal discharge hydrograph. In [8] a new approach to the optimal design of wastewater treatment systems is presented. An algorithm that can be divided into two parts is proposed for finding global optimal solutions to the problem. The first part comprises a new linear program formulation that is used to generate good starting points for the solution of the general nonlinear program (second part).

Dynamic programming has been used extensively in the optimization of water resource systems, as the nonlinear and stochastic features, which characterize a large number of water resource systems, can be translated into a dynamic programming formulation. However, when dynamic programming is applied to multiple reservoir systems, the usefulness of the technique is limited, as the computer memory requirement is quite large. In such cases, dynamic programming can only be applied if the complex problems with the large number of variables are decomposed into a series of subproblems, which are solved recursively. In the context of sewer network control, dynamic programming has been used for optimizing the design of drainage systems [51], for designing the least expensive network of sewers that will drain water from a number of discrete sources [56], for designing the lowest-cost drainage networks which include storage elements [17], and for control of the combined sewer network of the city and county of San Francisco [24].

Nonlinear programming offers a more general mathematical formulation than linear and dynamic programming and can effectively handle nonlinear objective functions and nonlinear constraints. In [16] an algorithm that combines elements of discrete dynamic programming (i. e., discrete state space, backward stagewise optimization) with elements of constrained optimization (i. e., nonlinear programming with equality constraints) is used for the optimal control of a multireservoir system. In [3], optimal control theory is used for real-time automated control of combined sewers, in [41,45,46,47] nonlinear programming is applied for the flow control of Québec Urban Community sewer network, in [20] a model-predictive control strategy that uses a mixed linear/quadratic objective function is applied in the Seattle metropolitan area to minimize combined sewer overflows, while a solution algorithm developed for the sewer network control problem applying the discrete maximum principle is used in [27,28,29,31,32,33,35,38].

Linear-quadratic control theory has been extensively applied in many fields, and a number of investigators have incorporated various aspects of linear-quadratic theory in their solutions to reservoir operations problems. In [59] a multivariable feedback controller is used for the control of combined storm-sewer systems, while in [27,28,29,30,31,34,36,37,38] a linear multivariable feedback regulator designed using the linear-quadratic methodology is used for the sewer network control problem.

Genetic algorithms have been proposed as a means of global optimization for a variety of engineering design problems. They mimic the natural genetic processes of evolution, deliberately keeping a range of good solutions to avoid being drawn into false local optima. Genetic algorithms are robust methods for searching the optimum solution to complex problems. In [57] they are applied to a four-reservoir, deterministic finite-horizon problem. To achieve water quality goals and wastewater treatment cost optimization in the Youngsan River, where water quality has decreased due to heavy pollutant loads from Kwangju City and surrounding areas, a water quality management model [11] has been developed through the integration

of a genetic algorithm and a mathematical water quality model with remarkable results.

Conventional rule-based control and **fuzzy logic** for real-time flow control of sewer systems have also been used with success. Conventional rule-based control systems are based on a large number of rules, while control systems based on fuzzy logic combine the simple rules of an expert system with a flexible specification of output parameters. In [23] a comparison of conventional rule-based flow control systems with a control system based on fuzzy logic is conducted for a combined sewer system, while in [18] a study was carried out for a part of the sewer system of the city of Flensburg using fuzzy logic for the real-time control of the sewer system. An interactive fuzzy approach has worked suitably [25] in water quality management when applied to developing a water quality management plan in the Tou-Chen River Basin in northern Taiwan for solving a multiobjective optimization problem involving vague and imprecise information related to data, model formulation, and the decision maker's preferences.

The **real-time control (RTC)** of wastewater systems has been a topic of research and application for many years, and the benefits of applying RTC strategies to various wastewater systems are presented in many research papers. In [9] a global optimal control prototype for the Barcelona urban drainage system is presented. [7] presents the results of the application of RTC strategies to the Roma-Cecchignola combined sewer system. In [58] the RTC is applied to sewer systems in Germany, while in [19] the analysis of the performance improvement of a new automatic central control procedure applied to the sewer system of Rotterdam is presented. The results of a study [48] showed that the Trebic sewer system is suitable for combined runoff control. The optimized control of a Moscow sewer subnetwork enabled significant improvements in the sewer network operation as shown in [14]. A global optimal control system was implemented on the Québec Urban Community's Westerly sewer network [45] to manage flows and water levels in real time and managed to decrease combined sewer overflow (CSO) volumes at four overflow sites by more than 85% for seven rainfall events recorded during the summer of 1999. In [44] fault-tolerant-model predictive-control strategies of sewer networks are investigated and applied to a portion of the Barcelona sewage network under realistic rain and fault scenarios, while in [43] hybrid model predictive control (HMPC) for sewer networks is introduced and applied to the same sewer network. In [15] the CORAL offline, a new tool for sewerage network modeling, simulation, and optimal strategy computation, is demonstrated for a test catchment of the Barcelona sewer network for the purpose of performing a global optimal control.

It should be noted that in the recent past the three parts of the urban wastewater system (sewer system, wastewater treatment plant (WWTP), and receiving water) have been considered as separate units in water quality management, and the aims of optimum performance were considered individually as well. The conventional RTC of sewer systems mainly aims at minimization of overflow volumes and loads, while treatment plant operation traditionally is mainly concerned with maintaining effluent standards. However, recent years have seen increased attention being paid to the integrated analysis of sewer networks, wastewater treatment plants and receiving waters, and many researchers have focused their work on **integrated modeling** and **integrated control**. Integrated control is characterized by two aspects [6]:

- Integration of objectives: control objectives within one subsystem may be based on criteria measured in other subsystems.
- Integration of information: control decisions taken in one subsystem may be based on information about the state of other subsystems.

One of the most important improvements in the field of integrated modeling and integrated control is due to Schütze and Butler's work [5,6,52,53,54,61]. However, other researchers have also studied the integrated modeling and integrated control of wastewater systems [2,12,13,21,39,49,50].

References

1. Aldama AA (1991) Pumping rate optimization in a storm drainage system through the combined use of numerical simulation and linear programming. Adv Water Resour 14(4):192–202
2. Bauwens W, Vanrolleghem P, Fronteau C, Smeets C (1995) An integrated methodology for the impact assessment of the design and operation of the sewer-wastewater treatment plant system on the receiving water qual-

ity. Med Fac Landbouwwetenschappen, Universiteit Gent, 60(4b):2447–2450
3. Bell W, Johnson G, Winn CB (1973) Simulation and control of flow in combined sewers. 6th Annual Simulation Symposium, Tampa, pp 26–47
4. Bradford B (1977) Optimal storage control in a combined sewer system. J Water Resour Plan Manag Div, Proc ASCE 103:1–15
5. Butler D, Katebi R, Jeppsson U, Baeza JA, Marinaki M, Mikkelsen PS, Capodaglio AG (2004) Towards a systems engineering approach to integrated catchment management. 6th International Conference on Systems Analysis and Integrated Assessment in Water Management, Beijing, China, November 3–5 (Poster)
6. Butler D, Schütze M (2005) Integrating simulation models with a view to optimal control of urban wastewater systems. Environ Model Softw 20:415–426
7. Campisano A, Schilling W, Modica C (2000) Regulators' setup with application to the Roma-Cecchignola combined sewer system. Urban Water 2:235–242
8. Castro PM, Matos HA, Novais AQ (2007) An efficient heuristic procedure for the optimal design of wastewater treatment systems. Resour Conserv Recycl 50(2):158–185
9. Cembrano G, Quevedo J, Salamero M, Puig V, Figueras J, Martí J (2004) Optimal control of urban drainage systems. A case study. Control Eng Pract 12:1–9
10. Chebbo G, Gromaire MC, Ahyerre M, Garnaud S (2001) Protection and transport of urban wet weather pollution in combined sewer systems: the 'Marais' experimental urban catchment in Paris. Urban Water 3:3–15
11. Cho JH, Sung KS, Ha SR (2004) A river water quality management model for optimising regional wastewater treatment using a genetic algorithm. J Environ Manag 73:229–242
12. Erbe V, Risholt LP (2000) Integrated optimisation of wastewater systems by real time control using an open simulation environment for integrated modelling. Proceedings Integrated Modellers User Group Conference, Prague, 12–14 April
13. Erbe V, Risholt LP, Schilling W, Londong J (2002) Integrated modelling for analysis and optimisation of wastewater systems – the Odenthal case. Urban Water 4:63–71
14. Ermolin YA (1999) Mathematical modelling for optimized control of Moscow's sewer network. Appl Math Model 23:543–556
15. Figueras J, Cembrano G, Puig V, Quevedo J, Salamero M, Martì J (2002) CORAL off-line: An object-oriented tool for optimal control of sewer networks. IEEE International Symposium on Computer Aided Control System Design Proceedings, September 18–20, 2002, Glasgow, pp 224–229
16. Foufoula-Georgiou E, Kitanidis PK (1988) Gradient dynamic programming for stochastic optimal control of multidimensional water resources systems. Water Resour Res 24(8):1345–1359
17. Froise S, Burges SJ (1978) Least-cost design of urban-drainage networks. J Water Resour Plan Manag Div, Proc ASCE 104:75–92
18. Fuchs L, Beeneken T, Spönemann P, Scheffer C (1997) Model Based Real-Time Control of Sewer System Using Fuzzy-Logic. Water Sci Technol 36(8–9):343–347
19. Geerse JMU, Lobbrecht AH (2002) Assessing the performance of urban drainage systems: 'general approach' applied to the city of Rotterdam. Urban Water 4:199–209
20. Gelormino MS, Ricker NL (1994) Model-predictive control of a combined sewer system. Int J Control 59(3):793–816
21. Harremoës P, Hvitved-Jacobsen T, Lynggaard-Jensen A, Nielsen B (1994) Municipal wastewater systems, integrated approach to design, monitoring and control. Water Sci Technol 29(1–2):419–426
22. Kallis G, Butler D (2001) Implications of the EU Water Framework Directive. Water Policy 3:125–142
23. Klepiszewski K, Schmitt TG (2002) Comparison of conventional rule based flow control with control processes based on fuzzy logic in a combined sewer system. Water Sci Technol 46(6–7):77–84
24. Labadie JW, Morrow DM, Chen YH (1980) Optimal control of unsteady combined sewer flow. J Water Resour Plan Manag Div, Proc ASCE 106:205–223
25. Lee C-S, Chang S-P (2005) Interactive fuzzy optimization for an economic and environmental balance in a river system. Water Resour 39:221–231
26. Lee JM, Bang KW (2000) Characterization of urban stormwater runoff. Water Resour 34(6):1773–1780
27. Marinaki M (1995) Central flow control in sewer networks, MSc thesis. Technical University of Crete, Chania
28. Marinaki M (2002) Optimal real-time control of sewer networks, PhD thesis. Technical University of Crete, Chania
29. Marinaki M, Marinakis Y (2006) Real-time central flow control for a large-scale sewer network. Submitted to Environ Model Assess
30. Marinaki M, Papageorgiou M (1996) A LQ-Regulator with feedforward terms applied to sewer network flow control. 4th International Conference on Control, Automation, Robotics and Vision (ICARCV'96), Singapore, December 3–6, pp 1441–1445
31. Marinaki M, Papageorgiou M (1997) Central flow control in sewer networks. ASCE J Water Resour Plan Manag 123(5):274–283
32. Marinaki M, Papageorgiou M (1998) Nonlinear optimal flow control for sewer networks. 1998 American Control Conference, Philadelphia, PA, June 24–26, pp 1289–1293
33. Marinaki M, Papageorgiou M (1999) A non-linear optimal control approach to central sewer network flow control. Int J Control 72(5):418–429
34. Marinaki M, Papageorgiou M (1999) Multivariable regulator approach to sewer network flow control. ASCE J Environ Eng 125(3):267–276

35. Marinaki M, Papageorgiou M (2001) Rolling-Horizon Optimal Control of Sewer Networks. Proceedings of the 2001 IEEE International Conference on Control Applications, Mexico City, September 5–7, pp 594–599
36. Marinaki M, Papageorgiou M (2003) Application of linear-quadratic regulators to sewer network control. In: Cabrera EE, Cabrera J (eds) Pumps, Electromechanical Devices and Systems Applied to Urban Water Management, vol 2. Swets and Zeitlinger, Lisse, pp 879–886
37. Marinaki M, Papageorgiou M (2003) Linear-quadratic regulators applied to sewer network flow control. European Control Conference ECC2003, University of Cambridge, September 1–4
38. Marinaki M, Papageorgiou M (2005) Optimal real-time control of sewer networks. Springer, London
39. Marsili-Libelli S (1995) Operational wastewater treatment control in the context of flexible standards. Mededelingen Faculteit Landbouwwetenschappen, Universiteit Gent 60:2503–2506
40. Melo JJ de, Câmara AS (1994) Models for the optimization of regional wastewater treatment systems. Eur J Oper Res 73:1–16
41. Méthot JF, Pleau M (1997) The effects of uncertainties on the control performance of sewer networks. Water Sci Technol 36(5):309–315
42. Nelen F (1994) A model to assess the performance of controlled urban drainage systems. Water Sci Technol 29(1–2):437–444
43. Ocampo-Martìnez C, Ingimundarson A, Puig V, Quevedo J (2006) Hybrid model predictive control applied on sewer networks The Barcelona Case Study. CTS-HYCON Workshop on Nonlinear and Hybrid Control, July 10–12, Paris
44. Ocampo-Martìnez C, Puig V, Quevedo J, Ingimundarson A (2005) Fault tolerant model predictive control applied on the Barcelona sewer network. Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference 2005, Seville, Spain, December 12–15, pp 1349–1354
45. Pleau M, Colas H, Lavallèe P, Pelletier G, Bonin R (2005) Global optimal real-time control of the Quebec Urban Drainage System. Environ Model Softw 20:401–413
46. Pleau M, Méthot F, Lebrun AM, Colas H (1996) Minimizing combined sewer overflows in real-time control applications. Water Qual Res J Can 31(4):775–786
47. Pleau M, Pelletier G, Colas H, Lavallée P, Bonin R (2001) Global predictive real-time control of Quebec Urban Community's Westerly sewer network. Water Sci Technol 43(7):123–130
48. Polaskova K, Hlavinek P, Haloun R (2006) Integrated approach for protection of an urban catchment area. Desalination 188:51–59
49. Rauch W, Aalderink H, Krebs P, Schilling W, Vanrolleghem P (1998) Requirements for integrated wastewater models – driven by receiving water objectives. Water Sci Technol 38(11):97–104
50. Rauch W, Harremoës P (1995) Integrated urban water quality management, Mededelingen Faculteit Landbouwkundige en toegepaste biologische wetenschappen. Universiteit Gent, 60 (4b), pp 2345–2352
51. Robinson DK, Labadie JW (1981) Optimal design of urban storm water drainage systems. International Symposium on Urban Hydrology, Hydraulics, and Sediment Control. Lexington, KY, 1981, pp 145–156
52. Schütze M, Butler D, Beck B (1999) Optimisation of control strategies for the urban wastewater system – an integrated approach. Water Sci Technol 39(9):209–216
53. Schütze M, Butler D, Beck MB (2001) Parameter optimisation of real-time control strategies for urban wastewater systems. Water Sci Technol 43(7):139–146
54. Schütze M, Butler D, Beck MB (2002) Modelling, simulation and control of urban wastewater systems. Springer, London
55. Seidl M, Servais P, Mouchel JM (1998)(2006) Organic matter transport and degradation in the river Seine (France) after a combined sewer overflow. Water Resour 32(12):3569–3580
56. Walters GA (1985) The Design of the Optimal Layout for a Sewer Network. Eng Optim 9:37–50
57. Wardlaw R, Sharif M (1999) Evaluation of Genetic Algorithms for Optimal Reservoir System Operation. J Water Resour Plan Manag 125(1):25–33
58. Weyand M (2002) Real-time control in combined sewer systems in Germany – Some case studies. Urban Water 4:347–354
59. Winn CB, Moore JB (1973) The Application of Optimal Linear Regulator Theory to a Problem in Water Pollution. IEEE Trans Syst Man Cybernet 3(5):450–455
60. Zabel T, Milne I, Mckay G (2001) Approaches adopted by the European Union and selected Member States for the control of urban pollution. Urban Water 3:25–32
61. Zacharof AI, Butler D, Schütze M, Beck MB (2004) Screening for real-time control potential of urban wastewater systems. J Hydrology 299:349–362

Y

Young Programming

PÉTER KAS[1], EMIL KLAFSZKY[2], LEVENTE MÁLYUSZ[2], GÖ KHAN İZBIRAK[1]
[1] Department Math., Eastern Mediterranean University, Mersin-10, Turkey
[2] Department Building Management, Techn. University, Budapest, Hungary

MSC2000: 90C25, 90C05

Article Outline

Keywords
The Young Inequality
Linear Programming
Young Programming
Approximation of the LP Problems
Algorithms
See also
References

Keywords

Analytical approximation of linear programming; Convex programming; Row-action method

The concept of *Young programming* is based on the Young inequality. Wide range applications in mechanics [9], statistics and decision theory [4,7] and information theory [3] give some beautiful interpretations of the Young inequality. In section 1 we shortly recall the inequality in a form that is convenient to use in the rest of the paper. In section 2 some basic facts of linear programming are restated in a form that is easy to generalize to Young programming. In section 3 the Young programming is introduced and the duality is discussed. In section 4 a parametric form Young programming is shown to be an *analytical approximation of the corresponding linear programming problem*. Finally in section 5 a *row-action method* for the solution of Young programming problems is presented. The algorithm interpreted in terms of the dual problem leads to an alternative method, we call it dir-action method.

The Young Inequality

The Young inequality was first published by W.H. Young in 1912. A generalized form of the inequality can be found in [5]. We recall the inequality in a form that is convenient to use in the rest of the paper. Let $\varphi\colon \mathbf{R} \to \mathbf{R}$ be a continuous, strictly decreasing function, and consider the curve $\gamma_\varphi = \{(x, \varphi(x))\colon x \in \mathbf{R}\}$. The following definition offers a way to describe how much an arbitrary point (u, v) of the plane is 'away' from the curve~γ_φ.

Definition 1 Let $(u, v) \in \mathbf{R}^2$ and denote $\psi = \varphi^{-1}$, then

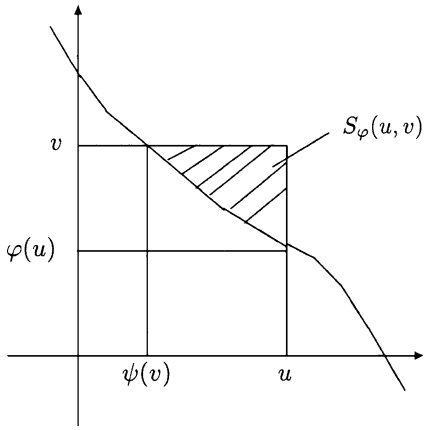
$$S_\varphi(u, v) = (u - \psi(v))v - \int_{\psi(v)}^{u} \varphi(t)\, dt$$

$$\left(= (v - \varphi(u))u - \int_{\varphi(u)}^{v} \psi(t)\, dt \right).$$

is called the *inaccuracy* of (u, v) with respect to γ_φ.

Geometrically $S_\varphi(u, v)$ is the area of the shaded region in Fig. 1. The *Young inequality* states that

$$S_\varphi(u, v) \geq 0 \quad \text{for every } (u, v) \in \mathbf{R}^2$$



Young Programming, Figure 1

and equality holds if and only if $v = \varphi(u)$. The geometric interpretation or a straightforward computation proves the statement. The inaccuracy can be termed in $\mathbf{R}^{2n}$ as follows.

Definition 2 Let $(\mathbf{u}, \mathbf{v}) \in \mathbf{R}^{2n}$, $\mathbf{u} = (u_1, \ldots, u_n)$, $\mathbf{v} = (v_1, \ldots, v_n)$, then

$$S_\varphi(\mathbf{u}, \mathbf{v}) := \sum_{j=1}^{n} S_\varphi(u_j, v_j)$$

is called the *inaccuracy* of $(\mathbf{u}, \mathbf{v}) \in \mathbf{R}^{2n}$, i. e. the inaccuracy is computed coordinatewisely.

Interesting examples of functions of the form $S_\varphi(u, v)$ are displayed by certain discrepancy or divergence functions. Let $f: \mathbf{R} \to \mathbf{R}$ be a strictly concave and differentiable function. Then a class of divergence functions can be generated by the following mapping $D_f: \mathbf{R} \times \mathbf{R} \to \mathbf{R}$,

$$D_f(u||v) = f(v) + f'(v)(u - v) - f(u), \qquad u, v \in \mathbf{R}.$$

D_f can serve as a 'measure of distance' between u and $v \in \mathbf{R}$ with respect to f, although strictly speaking D_f is not a distance function for it is not symmetric and the triangle inequality does not hold. Nevertheless the family of functions of the form D_f (f being not necessarily of a sum form) was introduced by L.M. Bregman [1].

Obviously

$$D_f(u||v) = f(v) + f'(v)(u - v) - f(u) = f'(v)(u - v) + \int_u^v f'(t)\, dt, \qquad u, v \in \mathbf{R}.$$

Let $f: \mathbf{R} \to \mathbf{R}$ be defined by $f' = \varphi$. It is easy to verify that

$$S_\varphi(u, \varphi(v)) = D_f(u||v) \quad \text{for all } u, v \in \mathbf{R},$$

or alternatively

$$S_\varphi(u, v) = D_f(u||\psi(v)) \quad \text{for all } u, v \in \mathbf{R}.$$

Some special cases, divergence functions known in the literature, are obtained by choosing

- $\varphi(t) = -\ln t - 1$, then $S_\varphi(u, \varphi(v)) = u \ln(u/v) - u + v$ known as I-divergence, introduced by S. Kullback~[10];
- $\varphi(t) = 1/t$, then $S_\varphi(u, \varphi(v)) = \ln(v/u) - (u/v) - 1$, known as *Itakura–Saito divergence* [6];
- $\varphi(t) = t^{\alpha-1}$, $\alpha < 1$, $\alpha \neq 0$, then $S_\varphi(u, \varphi(v)) = (v^\alpha - u^\alpha + \alpha v^{\alpha-1}(u - v))/\alpha$ known as *Csiszar's α-divergence*~[4].

Linear Programming

Since the Young programming will be introduced as an analytical approximation of linear programming, it is convenient to recall some basic facts of linear programming. Let A be an m × n matrix and denote $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(m)} \in \mathbf{R}^n$ the rows of A. Denote ℓ and $\ell^\perp$ the rowspace of matrix A, and the solution space of $A\mathbf{x} = \mathbf{0}$, respectively. Let $\widehat{\mathbf{x}}, \widehat{\mathbf{z}} \in \mathbf{R}^n$ be arbitrary but fixed vectors. Let us define the affine subspaces $\widehat{\mathbf{z}} \oplus \mathcal{L} = \{\mathbf{z} \in \mathbf{R}^n : \mathbf{z} = \widehat{\mathbf{z}} + \mathbf{w} \text{ for some } \mathbf{w} \in \mathcal{L}\}$ and $\widehat{\mathbf{x}} \oplus \mathcal{L}^\perp = \{\mathbf{x} \in \mathbf{R}^n : \mathbf{x} = \widehat{\mathbf{x}} + \mathbf{w} \text{ for some } \mathbf{w} \in \mathcal{L}^\perp\}$. Then clearly

$$\mathbf{x} \in \widehat{\mathbf{x}} \oplus \mathcal{L}^\perp \Leftrightarrow A\mathbf{x} = A\widehat{\mathbf{x}},$$

and

$$\mathbf{z} \in \widehat{\mathbf{z}} \oplus \mathcal{L} \Leftrightarrow \mathbf{z} = \widehat{\mathbf{z}} + \mathbf{y}A$$

for some $\mathbf{y} \in \mathbf{R}^m$.

Denote x_j, z_j the jth coordinate of $\mathbf{x}$, $\mathbf{z}$, respectively, and consider the following feasibility problem, which is

in fact the linear programming problem in an equilibrium form.

Problem 3 Let $\hat{\mathbf{x}}, \hat{\mathbf{z}} \in \mathbf{R}^n$ be arbitrary but fixed vectors. Find a feasible solution (if any) to the set of constraints

$$\mathbf{x} \in \hat{\mathbf{x}} \oplus \mathcal{L}^{\perp}, \quad \mathbf{z} \in \hat{\mathbf{z}} \oplus \mathcal{L}, \tag{1}$$

$$\mathbf{x} \geq \mathbf{0}, \quad \mathbf{z} \geq \mathbf{0}, \tag{2}$$

$$x_j z_j = 0, \quad j = 1, \ldots, n. \tag{3}$$

The next lemma states some elementary but crucial observations about Problem 3.

Lemma 4 *Let $\hat{\mathbf{x}}, \hat{\mathbf{z}} \in \mathbf{R}^n$ be arbitrary but fixed vectors.*
i) If $\boldsymbol{x}$ and $\boldsymbol{z}$ satisfy (1), then $\mathbf{x}\hat{\mathbf{z}} + \hat{\mathbf{x}}\mathbf{z} = \mathbf{x}\mathbf{z} + \hat{\mathbf{x}}\hat{\mathbf{z}}$.
ii) If $\boldsymbol{x}$ and $\boldsymbol{z}$ satisfy (2)–(2), then $\mathbf{x}\hat{\mathbf{z}} + \hat{\mathbf{x}}\mathbf{z} \geq \hat{\mathbf{x}}\hat{\mathbf{z}}$.
iii) If $\boldsymbol{x}$ and $\boldsymbol{z}$ satisfy (1)–(3), then $\mathbf{x}\hat{\mathbf{z}} + \hat{\mathbf{x}}\mathbf{z} = \hat{\mathbf{x}}\hat{\mathbf{z}}$.

Proof Elementary computation proves that i), ii) and iii) are obvious.

Let $\mathcal{P} = \{\mathbf{x}\colon\ \mathbf{x} \in \hat{\mathbf{x}} \oplus \mathcal{L}^{\perp},\ \mathbf{x} \geq \mathbf{0}\}$, and $\mathcal{D} = \{\mathbf{z}\colon\ \mathbf{z} \in \hat{\mathbf{z}} \oplus \mathcal{L},\ \mathbf{z} \geq \mathbf{0}\}$. The following problem presents three equivalent settings of the linear programming problem.

Problem 5 Let $\hat{\mathbf{x}}, \hat{\mathbf{z}} \in \mathbf{R}^n$ be arbitrary but fixed vectors.
i) (Equilibrium form) Find a feasible solution to (1)–(3).
ii) (Optimization form) Find a feasible solution to (1)–(2) such that $\sum_{j=1}^{n} x_j z_j$ is minimal.
iii) (Primal-dual form) Find solutions to both problems:

$$\frac{\begin{matrix}\textit{primal}\\ \min\left\{\sum_{j=1}^{n} x_j \hat{z}_j\colon\ \mathbf{x} \in \mathcal{P}\right\}\end{matrix}}{\begin{matrix}\textit{dual}\\ \min\left\{\sum_{j=1}^{n} \hat{x}_j z_j\colon\ \mathbf{z} \in \mathcal{D}\right\}\end{matrix}}.$$

The next theorem restates the well-known duality theorem of linear programming in three equivalent forms corresponding to problem settings in Problem 5.

Theorem 6 (Duality theorem) *Let $\hat{\mathbf{x}}, \hat{\mathbf{z}} \in \mathbf{R}^n$ be arbitrary but fixed vectors.*
i) If (1)–(2) is feasible, then (1)–(3) is feasible.
ii) If (1)–(2) is feasible, then there are $\boldsymbol{x}^* \in \mathcal{P}$, $\boldsymbol{z}^* \in \mathcal{D}$ such that $\sum_{j=1}^{n} x_j^* z_j^* = 0$.
iii) If $\mathcal{P} \neq \emptyset$ and $\mathcal{D} \neq \emptyset$, then there are optimal solutions $\boldsymbol{x}^* \in \mathcal{P}$ and $\boldsymbol{z}^* \in \mathcal{D}$ to both the primal and the dual problems, respectively; furthermore $\boldsymbol{x}^*\boldsymbol{z}^* = 0$.

The standard way in the literature is to prove Theorem 6iii) directly, then i) and ii) are easy corollaries. Finally let us point out that if we drop the assumption that curve γ is defined to be the graph of a function φ in Definition 1, then the term $x_j z_j$ can be interpreted as the inaccuracy of (x_j, z_j) with respect to $\gamma = \{(x, z)\colon x \geq 0, z \geq 0, xz = 0\}$. In the next section an equilibrium function, $\varphi\colon \mathbf{R}_+ \to \mathbf{R}_+$ will be introduced as an analytical approximation of γ.

Young Programming

Let $\varphi\colon \mathbf{R}_+ \to \mathbf{R}_+$ be a continuous, strictly decreasing function with $\lim_{x \to 0+} \varphi(x) = \infty$, $\lim_{x \to \infty} \varphi(x) = 0$. Let $\gamma_\varphi = \{(x, \varphi(x))\colon x \in \mathbf{R}_+\}$ and denote $\psi = \varphi^{-1}$. According to Definition 1, the inaccuracy of $(x, z) \in \mathbf{R}^2$, $x > 0$, $z > 0$ with respect to γ_φ is

$$S_\varphi(x, z) = (x - \psi(z))z - \int_{\psi(z)}^{x} \varphi(t)\, dt.$$

The basic properties of $S_\varphi(x, z)$ are summarized in the following lemma.

Lemma 7 *Let $\varphi\colon \mathbf{R}_+ \to \mathbf{R}_+$ be a continuous, strictly decreasing function with $\lim_{x \to 0+} \varphi(x) = \infty$, $\lim_{x \to \infty} \varphi(x) = 0$. Let $\psi = \varphi^{-1}$. Then*
i) $S_\varphi(x, z)$ is strictly convex function in $x > 0$ and $z > 0$, respectively.
ii) $S_\varphi(x, z) \geq 0$ for every $x > 0$, $z > 0$, and $S_\varphi(x, z) = 0$ if and only if $x = \psi(z)$.
iii) $\partial/(\partial x) S_\varphi(x, z) = z - \varphi(x)$ for every $x > 0$, $z > 0$.
iv) $\partial/(\partial z) S_\varphi(x, z) = x - \psi(z)$ for every $x > 0$, $z > 0$.
v) $\lim_{x \to 0+} \partial/(\partial x) S_\varphi(x, z) = -\infty$ for every $z > 0$.
vi) $\lim_{z \to 0+} \partial/(\partial z) S_\varphi(x, z) = -\infty$ for every $x > 0$.
vii) $\lim_{x \to \infty} \partial/(\partial x) S_\varphi(x, z) = z$ for every $z > 0$.
viii) $\lim_{z \to \infty} \partial/(\partial x) S_\varphi(x, z) = x$ for every $x > 0$.
ix) $\lim_{x \to \infty} S_\varphi(x, z) = \infty$ for every $z > 0$.
x) $\lim_{z \to \infty} S_\varphi(x, z) = \infty$ for every $x > 0$.
xi) $S_\varphi(x, z) = S_\psi(z, x)$ for every $x > 0$ and $z > 0$.

Proof Elementary computation proves each property.

The inaccuracy of $(\mathbf{x}, \mathbf{z}) \in \mathbf{R}^{2n}$, $\mathbf{x} > \mathbf{0}$, $\mathbf{z} > \mathbf{0}$ with respect to γ_φ, as introduced in Definition 2, is computed coordinatewisely, i. e. for every $\mathbf{x} = (x_1, \ldots, x_n)$, $x_j > 0$, $j = 1$,

$\ldots, n$, and $\mathbf{z} = (z_1, \ldots, z_n)$, $z_j > 0$, $j = 1, \ldots, n$,

$$S_\varphi(\mathbf{x}, \mathbf{z}) := \sum_{j=1}^{n} S_\varphi(x_j, z_j).$$

In the rest of this section the Young programming is presented by a complete analogy to linear programming as is discussed in section 2. Let us consider the following feasibility problem, which is in fact the Young programming problem in an equilibrium form.

Problem 8 Let $\widehat{\mathbf{x}} > \mathbf{0}, \widehat{\mathbf{z}} > \mathbf{0}$ be arbitrary but fixed vectors. Find a feasible solution to the set of constraints below.

$$\mathbf{x} \in \widehat{\mathbf{x}} \oplus \mathcal{L}^\perp, \quad \mathbf{z} \in \widehat{\mathbf{z}} \oplus \mathcal{L}, \tag{4}$$

$$\mathbf{x} > \mathbf{0}, \quad \mathbf{z} > \mathbf{0}, \tag{5}$$

$$x_j = \psi(z_j), \quad j = 1, \ldots, n. \tag{6}$$

The next lemma states some elementary but crucial observations about Problem 6.

Lemma 9 *Let $\widehat{\mathbf{x}} > \mathbf{0}, \widehat{\mathbf{z}} > \mathbf{0}$ be arbitrary but fixed vectors.*

i) If $\boldsymbol{x}$ and $\boldsymbol{z}$ satisfy (4)–(5), then $S_\varphi(\mathbf{x}, \widehat{\mathbf{z}}) + S_\varphi(\widehat{\mathbf{x}}, \mathbf{z}) = S_\varphi(\mathbf{x}, \mathbf{z}) + S_\varphi(\widehat{\mathbf{x}}, \widehat{\mathbf{z}})$.

ii) If $\boldsymbol{x}$ and $\boldsymbol{z}$ satisfy (4)–(5), then $S_\varphi(\mathbf{x}, \widehat{\mathbf{z}}) + S_\varphi(\widehat{\mathbf{x}}, \mathbf{z}) \geq S_\varphi(\widehat{\mathbf{x}}, \widehat{\mathbf{z}})$.

iii) If $\boldsymbol{x}$ and $\boldsymbol{z}$ satisfy (4)–(6), then $S_\varphi(\mathbf{x}, \widehat{\mathbf{z}}) + S_\varphi(\widehat{\mathbf{x}}, \mathbf{z}) = S_\varphi(\widehat{\mathbf{x}}, \widehat{\mathbf{z}})$.

Proof Elementary computation proves that i), ii) and iii) are obvious.

Let $\mathcal{P}_+ = \{\mathbf{x}\colon\ \mathbf{x} \in \widehat{\mathbf{x}} \oplus \mathcal{L}^\perp,\ \mathbf{x} \geq \mathbf{0}\}$, and $\mathcal{D}_+ = \{\mathbf{z}\colon\ \mathbf{z} \in \widehat{\mathbf{z}} \oplus \mathcal{L},\ \mathbf{z} \geq \mathbf{0}\}$. The following problem presents three equivalent settings of the Young programming problem.

Problem 10 Let $\widehat{\mathbf{x}} > \mathbf{0}, \widehat{\mathbf{z}} > \mathbf{0}$ be arbitrary but fixed vectors.

i) (Equilibrium form) Find a feasible solution to (4)–(6).

ii) (Optimization form) Find a feasible solution to (4)–(5) such that $\sum_{j=1}^{n} S_\varphi(x_j, z_j)$ is minimal.

iii) (Primal-dual form) Find solutions to both problems below:

$$\frac{\begin{array}{c}\textit{primal}\\ \min\left\{\sum_{j=1}^{n} S_\varphi(x_j, \widehat{z}_j)\colon\ \mathbf{x} \in \mathcal{P}_+\right\},\end{array}}{\begin{array}{c}\textit{dual}\\ \min\left\{\sum_{j=1}^{n} S_\varphi(\widehat{x}_j, z_j)\colon\ \mathbf{z} \in \mathcal{D}_+\right\}.\end{array}}$$

The primal and dual Young programming problems as defined in Problem 10 are symmetrical in the sense that dual of the dual is the primal. For a proof of this statement, see [8]. The next theorem presents three equivalent forms of the duality theorem corresponding to problem settings in Problem 10.

Theorem 11 (Duality theorem) *Let $\widehat{\mathbf{x}} > \mathbf{0}, \widehat{\mathbf{z}} > \mathbf{0}$ be arbitrary but fixed vectors.*

i) The system (4)–(6) is feasible.

ii) There are unique $\boldsymbol{x}^* \in \mathcal{P}_+$ and $\boldsymbol{z}^* \in D_+$ such that $\sum_{j=1}^{n} S_\varphi(x_j^*, z_j^*) = 0$.

iii) There exist $\boldsymbol{x}^* \in \mathcal{P}_+$ and $\boldsymbol{z}^* \in D_+$ unique optimal solutions to the primal and the dual problems, respectively. Furthermore $x_j^* = \psi(z_j^*)$, $j = 1, \ldots, n$.

Instead of referring to proofs of more general statements (duality theorems of convex programming, see e. g. [11]) we prefer to give a short proof of Theorem 11iii).

Proof The proof can be formulated on both the primal and the dual sides. To emphasize the symmetry we show the proof in both cases.

Proof on the dual side. Because of Lemma 7i), 7vi) and 7x) the dual objective function attains its unique minimum in D_+. Denote $\mathbf{z}^*$ the minimum and let $x_j^* := \psi(z_j^*) > 0$, $j = 1, \ldots, n$. Suppose that $\mathbf{a}^{(i)}\mathbf{x}^* \neq \mathbf{a}^{(i)}\widehat{\mathbf{x}}$ for some i. Let $\mathbf{z}(\theta) := \mathbf{z}^* + \theta\, \mathbf{a}^{(i)}$. Clearly $\mathbf{z}(\theta) \in \mathcal{D}_+$ for small enough θ, and

$$\frac{d}{d\theta} S(\widehat{\mathbf{x}}, \mathbf{z}(\theta))\Big|_{\theta=0} = \mathbf{a}^{(i)}\widehat{\mathbf{x}} - \mathbf{a}^{(i)}\mathbf{x}^* \neq 0$$

what is in contradiction with the assumption that $\mathbf{z}^*$ is dual optimal solution. Therefore $\mathbf{x}^*$ is primal optimal solution.

Proof on the primal side. Because of Lemma 7i, 7v) and 7ix the primal objective function attains its unique minimum in $\mathcal{P}_+$. Denote $\mathbf{x}^*$ the minimum and let $z_j^* := \varphi(x_j^*) > 0$, $j = 1, \ldots, n$. Suppose that $\mathbf{z}^* - \widehat{\mathbf{z}} \neq \mathbf{y}A$

i. e. $\mathbf{z}^* - \widehat{\mathbf{z}} \notin \mathcal{L}(\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(n)})$. Then there exists an $\widetilde{\mathbf{x}} \in \mathcal{L}^{\perp}(\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(n)})$ such that $\widetilde{\mathbf{x}}(\mathbf{z}^* - \widehat{\mathbf{z}}) \neq 0$. Let $\mathbf{x}(\theta) := \mathbf{x}^* + \theta\widetilde{\mathbf{x}}$. Clearly $\mathbf{x}(\theta) \in \mathcal{P}_+$ for small enough θ, and

$$\frac{d}{d\theta} S(\mathbf{x}(\theta), \widehat{\mathbf{z}})\Big|_{\theta=0} = \widetilde{\mathbf{x}}(\widehat{\mathbf{z}} - \mathbf{z}^*) \neq 0$$

what is in contradiction with the assumption that $\mathbf{x}^*$ is primal optimal solution. Therefore $\mathbf{z}^*$ is dual optimal solution. This completes the proof of iii).

To show that i), ii), iii) are equivalent statements it is enough to note that $S_\varphi(\mathbf{x}, \mathbf{z}) \geq 0$ for every $\mathbf{x} > \mathbf{0}, \mathbf{z} > \mathbf{0}$ and $S_\varphi(\mathbf{x}, \mathbf{z}) = 0$ if and only if $x_j = \psi(z_j)$, $j = 1, \ldots, n$.

The next corollary points out that the optimal solutions of the primal and dual problems do not depend on the choice of parameter vectors $\widehat{\mathbf{x}} \in \widehat{\mathbf{x}} \oplus \mathcal{L}^{\perp}, \widehat{\mathbf{x}} > \mathbf{0}$ and $\widehat{\mathbf{z}} \in \widehat{\mathbf{z}} \oplus \mathcal{L}, \widehat{\mathbf{z}} > \mathbf{0}$.

Corollary 12 *Let $\widehat{\mathbf{x}}, \widehat{\mathbf{z}}$ be given. There exist unique $\mathbf{x}^* \in \widehat{\mathbf{x}} \oplus \mathcal{L}^{\perp}$, $\mathbf{z}^* \in \widehat{\mathbf{z}} \oplus \mathcal{L}$ such that $x_j^* = \psi(z_j^*)$, $j = 1, \ldots, n$, and*

$$S_\varphi(\mathbf{x}^*, \mathbf{z}) + S_\varphi(\mathbf{x}, \mathbf{z}^*) = S_\varphi(\mathbf{x}, \mathbf{z}),$$
$$\forall \mathbf{x} \in \widehat{\mathbf{x}} \oplus \mathcal{L}^{\perp}, \quad \forall \mathbf{z} \in \widehat{\mathbf{z}} \oplus \mathcal{L}.$$

Proof Obvious from Theorem 11.

Another important implication of the duality theorem is noted in the following corollary.

Corollary 13 *Let $\widehat{\mathbf{x}}$, $\widehat{\mathbf{z}}$ be given and denote $\mathbf{x}^* = \operatorname{argmin}\{S_\varphi(\mathbf{x}, \widehat{\mathbf{z}}) \colon \mathbf{x} \in \widehat{\mathbf{x}} \oplus \mathcal{L}^{\perp}, \mathbf{x} > \mathbf{0}\}$ and $\mathbf{z}^* = \operatorname{argmin}\{S_\varphi(\widehat{\mathbf{x}}, \mathbf{z}) \colon \mathbf{z} \in \widehat{\mathbf{z}} \oplus \mathcal{L}, \mathbf{z} > \mathbf{0}\}$. Suppose that $\mathcal{L}' \subseteq \mathcal{L}$ and denote $\mathbf{z}'^* = \operatorname{argmin}\{S_\varphi(\widehat{\mathbf{x}}, \mathbf{z}) \colon \mathbf{z} \in \widehat{\mathbf{z}} \oplus \mathcal{L}', \mathbf{z} > \mathbf{0}\}$. Then*

$$\mathbf{x}^* = \operatorname{argmin}\{S_\varphi(\mathbf{x}, \mathbf{z}'^*) \colon \mathbf{x} \in \widehat{\mathbf{x}} \oplus \mathcal{L}^{\perp}, \mathbf{x} > \mathbf{0}\}.$$

Proof Since $\mathbf{z}'^* \in \widehat{\mathbf{z}} \oplus \mathcal{L}' \subseteq \widehat{\mathbf{z}} \oplus \mathcal{L}$, the statement follows from Corollary 12.

Observations of Corollary 12 and Corollary 13 give rise to the row-action method proposed in section 5.

Approximation of the LP Problems

In this section we introduce a parameter $\epsilon > 0$ in the Young programming problem, and prove that the sequence of optimal solutions of parametric Young programming problems converges to an optimal solution of the corresponding linear programming problem.

Definition 14 Let $\epsilon > 0$, φ: $\mathbf{R}_+ \to \mathbf{R}_+$ continuous, strictly decreasing function with $\lim_{x \to 0+} \varphi(x) = \infty$, $\lim_{x \to \infty} \varphi(x) = 0$. Define $\varphi_\epsilon(x) := \epsilon\, \varphi(x)$.

Denote $\psi_\epsilon = \varphi_\epsilon^{-1}$. Clearly $\psi_\epsilon(x) = \psi(x/\epsilon)$, where $\psi = \varphi^{-1}$. Let us consider the following parametric version of the Young programming primal-dual pair.

Problem 15 Let $\widehat{\mathbf{x}} > \mathbf{0}, \widehat{\mathbf{z}} > \mathbf{0}$ be arbitrary but fixed vectors. Find a solution to both problems below

$$\begin{array}{c} \textit{primal} \\ \mathbf{x} \in \widehat{\mathbf{x}} \oplus \mathcal{L}^{\perp} \\ \mathbf{x} > \mathbf{0} \\ \min \sum_{j=1}^n S_{\varphi_\epsilon}(x_j, \widehat{z}_j) \\ \hline \textit{dual} \\ \mathbf{z} \in \widehat{\mathbf{z}} \oplus \mathcal{L} \\ \mathbf{z} > \mathbf{0} \\ \min \sum_{j=1}^n S_{\varphi_\epsilon}(\widehat{x}_j, z_j). \end{array}$$

For every given $\epsilon > 0$, denote $\mathbf{x}^*(\epsilon)$, $\mathbf{z}^*(\epsilon)$ the optimal solutions to Problem 15. The next theorem points out that the sets of optimal solutions $\{\mathbf{x}^*(\epsilon) \colon \epsilon < \epsilon_0\}$, $\{\mathbf{z}^*(\epsilon) \colon \epsilon < \epsilon_0\}$ are bounded for a small enough $\epsilon_0 > 0$.

Theorem 16 *Let $\epsilon < \epsilon_0 = \min_j \widehat{z}_j/(\varphi(\widehat{x}_j))$. Then there exists a $K \in \boldsymbol{R}_+$ such that $x_j^*(\epsilon) < K$ and $z_j^*(\epsilon) < K$ for $j = 1, \ldots, n$.*

Proof The optimality of $\mathbf{x}^*(\epsilon)$ and $\mathbf{z}^*(\epsilon)$ implies that

$$\sum_{j=1}^n S_{\varphi_\epsilon}(x_j^*(\epsilon), \widehat{z}_j) + \sum_{j=1}^n S_{\varphi_\epsilon}(\widehat{x}_j, z_j^*(\epsilon)) = \sum_{j=1}^n S_{\varphi_\epsilon}(\widehat{x}_j, \widehat{z}_j).$$

Let $\epsilon < \epsilon_0 = \min_j \widehat{z}_j/(\varphi(\widehat{x}_j))$. It is clear from the geometric interpretation of $S_\varphi(x, z)$ in Fig. 1 that $S_{\varphi_\epsilon}(\widehat{x}_j, \widehat{z}_j) < \widehat{x}_j\widehat{z}_j$ for every $\epsilon < \epsilon_0$ and $j = 1, \ldots, n$. Since $S_{\varphi_\epsilon}(\widehat{x}_j, z_j^*(\epsilon)) \geq 0$, $j = 1, \ldots, n$, we have that

$$\sum_{j=1}^n S_{\varphi_\epsilon}(x_j^*(\epsilon), \widehat{z}_j) < \sum_{j=1}^n \widehat{x}_j\widehat{z}_j$$

for every $\epsilon < \epsilon_0$, which implies the boundedness of $x_j^*(\epsilon)$, $j = 1, \ldots, n$, for $\lim_{x \to \infty} S_{\varphi_\epsilon}(x_j, \widehat{z}_j) = \infty$. The proof to show the boundedness of $z_j^*(\epsilon)$, $j = 1, \ldots, n$, follows the same line of thoughts

Corollary 17 *Let $\mathbf{x}^*(\epsilon)$, $\mathbf{z}^*(\epsilon)$ be optimal solutions of Problem 10. Then*

$$\lim_{\epsilon \to 0} x_j^*(\epsilon) z_j^*(\epsilon) = 0, \quad j = 1, \ldots, n.$$

Proof If $\lim_{\epsilon \to 0} z_j^*(\epsilon) = 0$, then due to the boundedness of $x_j^*(\epsilon)$ the corollary follows. If $z_j^*(\epsilon) \geq a > 0$ for all $\epsilon < \epsilon_0$, then $\lim_{\epsilon \to 0} x_j^*(\epsilon) = \lim_{\epsilon \to 0} \varphi(z_j^*(\epsilon)/\epsilon) = 0$ and the corollary follows

Putting together the observations of this section we get that $\lim_{\epsilon \to 0} \mathbf{x}^*(\epsilon) = \mathbf{x}^*$ and that $\lim_{\epsilon \to 0} \mathbf{z}^*(\epsilon) = \mathbf{z}^*$ are optimal solutions of the linear programming primal-dual pair corresponding to Problem 15, i. e.

$$\begin{array}{c} \textit{primal} \\ \mathbf{x} \in \widehat{\mathbf{x}} \oplus \mathcal{L}^{\perp} \\ \mathbf{x} > \mathbf{0} \\ \min \sum_{j=1}^{n} x_j \widehat{z}_j \\ \hline \textit{dual} \\ \mathbf{z} \in \widehat{\mathbf{z}} \oplus \mathcal{L} \\ \mathbf{z} > \mathbf{0} \\ \min \sum_{j=1}^{n} \widehat{x}_j z_j. \end{array}$$

Algorithms

Row-action algorithms as defined in [2] are ones that use only the previous iterate in each iterative step, and access is required to only one row of the system of equations of the constraint set. A row-action method was first suggested in the pioneering work of Bregman [1]. The following algorithm is a row-action method for the solution of the Young programming problem as stated in Problem 10iii).

Initialize:
$\mathbf{z}^0 = \widehat{\mathbf{z}}$
$i := k \pmod m$.
Step k:
$\mathbf{x}^k = \operatorname{argmin}\{S_\varphi(\mathbf{x}, \mathbf{z}^{k-1}) : \mathbf{a}^{(i)}\mathbf{x} = \mathbf{a}^{(i)}\widehat{\mathbf{x}}\}$,
$z_j^k = \varphi(x_j^k)$, $j = 1, \ldots, n$.

Algorithm 1: row-action method

It may be interesting to point out that in terms of the dual, Step k reads as follows

Step k:
$\mathbf{z}^k = \mathbf{z}^{k-1} + \vartheta_k \mathbf{a}^{(i)}$
$\vartheta_k = \operatorname{argmin}\{S_\varphi(\widehat{\mathbf{x}}, \mathbf{z}^{k-1} + \vartheta \mathbf{a}^{(i)}) : \vartheta \in \mathbf{R}\}$.

That is, in dual terms, Step k is a one-dimensional minimization problem along the row vector $\mathbf{a}^{(i)}$, $i = k \pmod m$ at each step. The convergence of this algorithm was shown by I. Csiszar [4] if any of the following two assumptions holds:

[A1] the set $\{\mathbf{x}\colon\ \mathbf{a}^{(i)}\mathbf{x} = \mathbf{a}^{(i)}\widehat{\mathbf{x}}, \mathbf{x} \geq \mathbf{0}\}$ is bounded for at least one i, $1 \leq i \leq m$;

[A2] $\int_0^a \varphi(t)dt = \text{infin;}$, for some $a > 0$.

Typically A1) holds for problems involving discrete random variables, where the sum of the components is one. A2) holds for example for $\varphi(x) = x^{\alpha - 1}$, $0 < \alpha < 1$. Note that, due to Lemma 7ix), the dual objective function can be rewritten as $S_\psi(\mathbf{z}, \widehat{\mathbf{x}})$, where $\psi = \varphi^{-1}$. Then the duality theorem (Theorem 11) enables us to add the remark that convergence is also ensured if

[A2'] $\int_0^a \psi(t)\, dt = \infty$, for some $a > 0$.

For example, A2') holds for $\varphi(x) = x^{\alpha - 1}$, $\alpha < 0$.

The convergence of Algorithm 1 without any further assumption, although is likely to be true, remains an open question according to the best knowledge of the authors. Finally we present a small numerical example to display the steps of Algorithm 1.

Example 18 Let us consider the following Young programming problem.

$$\begin{cases} \min & \displaystyle\sum_{j=1}^{4} S_\varphi(x_j, \widehat{z}_j) \\ \text{s.t.} & A\mathbf{x} = A\widehat{\mathbf{x}} \\ & \mathbf{x} > \mathbf{0} \end{cases}$$

where

$$A = \begin{pmatrix} -8 & -2 & 5 & 8 \\ 7 & -18 & -22 & 13 \end{pmatrix},$$
$$\widehat{\mathbf{x}}^{\top} = \begin{pmatrix} 6 & 10 & 3 & 1 \end{pmatrix},$$
$$\widehat{\mathbf{z}}^{\top} = \begin{pmatrix} \frac{1}{8} & \frac{1}{12} & \frac{1}{5} & \frac{1}{3} \end{pmatrix}$$

and $\varphi(t) = 1/t$. The steps that Algorithm 1 takes on this example are arranged in the table below (iterations were stopped when $|x_j^{(k)} - x_j^{(k+1)}| < 10^{-8}$, $j = 1, 2, 3, 4$).

Step	x_1	x_2	x_3	x_4	$\sum_{j=1}^{4} S_\varphi(x_j, \widehat{z}_j)$
0	6.000	10.000	3.000	1.000	0.596108
1	9.093	8.585	5.747	2.023	0.137325
2	9.655	10.002	5.426	3.139	0.038966
3	8.815	10.386	4.679	2.861	0.018048
4	8.898	10.597	4.632	3.028	0.016183
5	8.827	10.630	4.568	3.004	0.016004
6	8.834	10.648	4.564	3.018	0.016003
7	8.828	10.650	4.559	3.016	0.016003
8	8.828	10.651	4.564	3.017	0.016003
9	8.828	10.652	4.558	3.017	0.016003

So, the optimal solution is $(\mathbf{x}^*)^\mathsf{T} = (8.828\ 10.652\ 4.558\ 3.017)$. If we solve the dual problem, then $z_j^{(k)} = 1/x_j^{(k)}$, $j = 1, 2, 3, 4$, for every $k = 1, \ldots, 9$, so the optimal solution for the dual problem is $(\mathbf{z}^*)^\mathsf{T} = (0.113\ 0.094\ 0.219\ 0.331)$.

See also

► Linear Programming

References

1. Bregman LM (1967) The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. USSR Comput Math Math Phys 7:200–217
2. Censor Y (1981) Row-action methods for huge and sparse systems and their applications. SIAM Rev 23:444–466
3. Cover TM, Thomas JA (1991) Elements of information theory. Wiley, New York
4. Csiszar I (1975) I-divergence geometry of probability distributions and minimization problems. Ann of Probab 3:146–156
5. Hiriart-Urruty JB, Lemarechal C (1993) Convex analysis and minimization algorithms, vol I. Springer, Berlin
6. Itakura F, Saito S (1968) Analysis synthesis telephony based on the maximum likelihood method. In: Kohasi Y (ed) Reports Sixth Internat. Congress on Acoustics. pp 17–20
7. Kas P, Klafszky E (1993) On the duality of the mixed entropy programming. Optim 27:253–258
8. Kas P, Klafszky E (1996) On the dual of linear inverse problems. Europ J Oper Res 91:634–639
9. Khincsin AI (1949) Mathematical foundation of statistical mechanics. Dover, Mineola, NY
10. Kullback S (1977) Information theory and statistics. Wiley, New York
11. Rockafellar R T (1970) Convex analysis. Princeton Univ. Press, Princeton

Preface to the Index Volume

This volume comprises the index to volumes 1–6 of the ENCYCLOPEDIA OF OPTIMIZATION. It contains two indices: a Subject Index, and a Name Index. In these indices, part of the information in the articles has been 'inverted'.

To understand the contents of these indices, recall that each article in the first six volumes has the following global structure:

- title (in **bold**)
- text of the article; important notions are printed in *italics*, references to other articles are printed in **bold**, and the first mention of a scientist includes his/her initials
- bibliography
- author(s)
- AMS 2000 classification code
- list of keywords and phrases

Name Index. The Name Index has an entry for each scientist explicitly mentioned in the text of an article. The entry lists the titles of the articles in which that Person is mentioned. The Name Index is alphabetically sorted according to the scientist's last name.

Subject Index. The Subject Index contains entries of four types, using different fonts:

- article titles (in **bold**)
- phrases marked in the articles as being important (in *italics*, as in the articles)
- keywords and phrases as listed underneath each article (in a plain font)
- rotations of the three entries above (in a sans-serif font)

Article Titles. For each article title we first list the AMS 2000 subject classification code, then the titles of articles that refer to the article (i. e., those that mention the article), and, last, the titles of articles to which the article refers (i. e., those mentioned and printed in a bold font in the text of the article).

Important Phrases. For each such phrase we give the list of article titles in which the phrase (or a standard form of it) appears, and all the AMS classification codes associated to these articles. These codes are thus taken from the articles.

Keywords and Phrases. For each such entry we give the list of article titles having exactly this word or phrase in the *Keywords and phrases* section, and all the AMS classification codes associated with these articles.

Rotations. The *rotation set of a phrase* is formed by phrases obtained by successively moving the first part of the initial phrase to the end. If the phrase thus obtained does not start with an uninformative word (like 'the', 'its', 'to'), the phrase belongs to the rotation set of the initial phrase.

Rotation entries list the last part and refer to the actual index entry. They allow you to locate an exact phrase when you only know a word occurring somewhere in it.

An example to clarify this follows:
Suppose the initial phrase is 'Adaptive simulated annealing and its application to protein folding'. Then its rotations are

1) simulated annealing and its application to protein folding *see:* Adaptive –
2) annealing and its application to protein folding *see:* Adaptive simulated –
3) and its application to protein folding *see:* Adaptive simulated annealing –
4) its application to protein folding *see:* Adaptive simulated annealing and –
5) application to protein folding *see:* Adaptive simulated annealing and its –
6) to protein folding *see:* Adaptive simulated annealing and its application –
7) protein folding *see:* Adaptive simulated annealing and its application to –
8) folding *see:* Adaptive simulated annealing and its application to protein –

Now, 3), 4), 6) clearly are not very helpful as regards the index, and only 1), 2), 5), 7), 8) remain and form the rotation set. So, by looking in the Subject Index for entries starting with any of the words 'simulated', 'annealing', 'application', 'protein', 'folding', you will find the phrase 'Adaptive simulated annealing and its application to protein folding' as well as others!

Order of Entries. Both indices are in alphabetical order, with numerals and mathematics symbols first. Sub-/superscript and small uninformative words (such as "in", "of", "the") are ignored. Punctuation signs and the symbols '-' and '–' count as space.

Please note that greek letters are sorted as if spelled out, e. g., ζ as 'zeta'.

When two phrases are exactly the same but are in different fonts, the order is: bold, italics, plain, sans-serif (or: article titles, important notions, keywords and phrases, rotations).

It is hoped that both indices will lead you quickly to the wealth of information provided in the six volumes of the ENCYCLOPEDIA OF OPTIMIZATION.

July 2008

Subject Index

0–1–0 *graph*
[58E05, 90C30]
(*see:* **Topology of global optimization**)

0–1 knapsack *see:* fractional —

0-1 linear programming approach for DNA transcription element identification *see:* Mixed —

0–1 *mixed integer problems*
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)

0-1 programming problem *see:* fractional —; hyperbolic —; single-ratio fractional (hyperbolic) —

0–1 programs *see:* mixed integer —

0-diagonal operator *see:* block- —; off- —

1 knapsack *see:* fractional 0– —

1 linear programming approach for DNA transcription element identification *see:* Mixed 0- —

1-*median problem in a network*
[90B80, 90B85]
(*see:* **Warehouse location problem**)

1 mixed integer problems *see:* 0– —

1-*MP*
[90B80, 90B85]
(*see:* **Warehouse location problem**)

1 programming problem *see:* fractional 0- —; hyperbolic 0- —; single-ratio fractional (hyperbolic) 0- —

1 programs *see:* mixed integer 0– —

1D-diffusion fluxes *see:* estimation of —

2 *see:* SSS- —

2-*dimensional grid*
[65K05, 65Y05]
(*see:* **Parallel computing: models**)

2-*dimensional torus*
[65K05, 65Y05]
(*see:* **Parallel computing: models**)

2-*matching problem*
[90C05, 90C10, 90C11, 90C27, 90C35, 90C57]
(*see:* **Assignment and matching**; **Integer programming**)

2-opt
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)

2-*opt neighborhood*
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)

2-*partition*
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)

2-SAT *see:* MAX- —

2-separated
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)

2-*step superlinear*
[65K05, 65K10, 90C06, 90C30, 90C34, 90Cxx]
(*see:* **Discontinuous optimization**; **Feasible sequential quadratic programming**)

2-valued function *see:* Boolean —

2-valued logic algebra *see:* Boolean —

2-valued normal forms *see:* PI-algebras and —

2B-consistency
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)

3-colorability
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)

3-*DIMENSIONAL MATCHING*
[90C60]
(*see:* **Complexity classes in optimization**)

3-dimensional matching problem
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)

3-*partition*
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)

#3 problem *see:* Gomez —

3-*SAT*
[68Q25, 90C60]
(*see:* **Complexity classes in optimization**; **NP-complete problems and proof methodology**)

3-*satisfiability*
[68Q25, 90C60]
(*see:* **Complexity classes in optimization**; **NP-complete problems and proof methodology**)

3B-consistency
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)

3D-transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)

3PM process
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: differential equations**)
4-element group *see:* Klein —
6000 *see:* EasyModeler/ —
= N P *see:* P —
∂^+*-critical point*
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
∂^-*-critical point*
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
∂^+*-function*
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
∂^-*-function*
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
∞*-stationary point*
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
∞-stationary point *see:* Hadamard —
$\in$-subdifferential
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)

A

a priori
[90C25, 94A17]
(*see:* **Bilevel programming: applications in engineering**; **Entropy optimization: shannon measure of entropy and its properties**)
a priori method
[65K05, 90B50, 90C05, 90C29, 91B06]
(*see:* **Multi-objective optimization and decision support systems**)
a priori optimization
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
A-weighted Euclidean norm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
A search algorithm*
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
Abadie CQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
Abaffi–Broyden–Spedicato algorithms for linear equations and linear least squares
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
Abaffian
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
Abaffian matrices
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
Abaffians
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
abnormal extremal
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
abnormal extremals *see:* High-order maximum principle for —
abnormal points
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
abnormal points *see:* High-order necessary conditions for optimality for —
abnormal processes
[41A10, 46N10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**; **High-order necessary conditions for optimality for abnormal points**)
abnormal weak extremal
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
ABS algorithms for linear equations and linear least squares
(65K05, 65K10)
(*referred to in:* **ABS algorithms for optimization**; **Cholesky factorization**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Least squares orthogonal polynomials**; **Least squares problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
(*refers to:* **ABS algorithms for optimization**; **Cholesky factorization**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Least squares orthogonal polynomials**; **Least squares problems**; **Linear programming**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
ABS algorithms for optimization
(65K05, 65K10)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Least squares orthogonal polynomials**; **Least**

squares problems; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Least squares orthogonal polynomials**; **Least squares problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**)
ABS class *see:* basic —; scaled —; unsealed —
ABS class of algorithms *see:* scaled —
ABS methods
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**)
absolute deviation *see:* least —; maximum —; mean —
absolute estimation
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)
absolute limit
[01A99]
(*see:* **Gauss, Carl Friedrich**)
absolute qualification rule
[90C35]
(*see:* **Feedback set problems**)
absolutely continuous functional
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
abstract constraint
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
abstract constraints
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
abstract convex analysis
[90C26]
(*see:* **Global optimization: envelope representation**)
abstract convex function
[90C26]
(*see:* **Global optimization: envelope representation**)
abstract convexity
[90C26]
(*see:* **Global optimization: envelope representation**)
abstract convexity
[90C26]
(*see:* **Global optimization: envelope representation**)
abstract group *see:* realization of an —
abstract hemivariational inequality
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
abstract variational inequality of elliptic type
[65M60]
(*see:* **Variational inequalities: F. E. approach**)
AC3
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
acceleration devices and related techniques
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
acceleration function *see:* the mid-point —
acceleration steps
[90C30]
(*see:* **Cyclic coordinate method**)
accelleration of algorithms
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
acceptance measure
(*see:* **Bayesian networks**)
acceptance/rejection
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C10, 90C26, 90C27, 90C30]
(*see:* **Multidimensional knapsack problems**; **Stochastic global optimization: two-phase methods**)
accepted by a Turing machine *see:* language —
accepting *see:* threshold —
accepting algorithms *see:* threshold —
accepting computation of a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
accepting state of a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
access machine *see:* parallel random —
accessibility form of CEP *see:* restricted —
accessible form of CEP *see:* universally —
accessible state
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
accessory minimum problem
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
ACCPM
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
accumulate
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
accumulation of the Jacobian
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
accuracy
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
achievable region method
[90B36]
(*see:* **Stochastic scheduling**)
achievement
[90C29]
(*see:* **Multiple objective programming support**)
achievement function
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**; **Multiple objective programming support**)
achievement scalarizing program
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)

acid *see:* amino —
acquisitions *see:* Multicriteria methods for mergers and —
across a fault *see:* jump —
across an s—t-cut *see:* flow —
action *see:* Clarke dual —; corrective —; recourse —; total —
action algorithm *see:* row- —
action method *see:* row- —
actions *see:* recourse —
activation function
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
active
[05C85, 46N10, 47N10, 49M37, 65K10, 90C10, 90C26, 90C30, 90C46, 90C60]
(*see:* **Complexity of degeneracy**; **Directed tree networks**; Global optimization: tight convex underestimators; **Integer programming duality**; **Railroad locomotive scheduling**)
active *see:* p-order —
active constraint
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
active constraints
[90C26, 90C30, 90C39]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Kuhn–Tucker optimality conditions**; **Second order optimality conditions for nonlinear optimization**)
active constraints
[90C26, 90C60]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Complexity of degeneracy**)
active constraints *see:* strongly —
active function
[49K35, 49M27, 65K10, 90C25]
(*see:* **Convex max-functions**)
active index
[65K05, 90C26, 90C33, 90C34]
(*see:* **Adaptive convexification in semi-infinite optimization**)
active index set
[49J52, 49K35, 49M27, 49Q10, 57R12, 65K10, 74G60, 74H99, 74K99, 74Pxx, 90C25, 90C31, 90C34, 90C46, 90C90]
(*see:* **Convex max-functions**; Generalized semi-infinite programming: optimality conditions; **Quasidifferentiable optimization: stability of dynamic systems**; **Semi-infinite programming: second order optimality conditions**; **Smoothing methods for semi-infinite optimization**)
active index set *see:* essentially —
active inequality constraints
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
active points *see:* set of ε-most —
active ridge
[90Cxx]
(*see:* **Discontinuous optimization**)
active set
[65K05, 65K10, 90C20, 90C30]
(*see:* **ABS algorithms for optimization**; **Quadratic programming with bound constraints**; **Rosen's method, global convergence, and Powell's conjecture**)
active set algorithm
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
active set method
[90Cxx]
(*see:* **Discontinuous optimization**)
active set methods
[49M37, 65K05, 90C25, 90C30, 90C60]
(*see:* **Complexity of degeneracy**; **Inequality-constrained nonlinear optimization**; **Successive quadratic programming: full space methods**)
active set methods
[90C25, 90C30, 90C60, 90Cxx]
(*see:* **Complexity of degeneracy**; **Discontinuous optimization**; **Successive quadratic programming**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
active set quadratic programming methods
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
active set strategies
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
active set strategy
[90C25, 90C30]
(*see:* **Successive quadratic programming: solution by active sets and interior point methods**)
active set strategy *see:* Goldfarb–Idnani —
active sets and interior point methods *see:* Successive quadratic programming: solution by —
active site
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
active site
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
activities *see:* matrix of —
activity *see:* direction, preserving an —
activity coefficient
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
actual
[65H20]
(*see:* **Multi-scale global optimization using terrain/funneling methods**)
acute angle condition
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
acyclic oriented matroid
[90C09, 90C10]
(*see:* **Oriented matroids**)
acyclic oriented matroid *see:* totally —
acyclic subdigraph problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
acyclic tournament *see:* spanning —

AD
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
AD *see:* error estimates for —; forward mode of —; point —; reverse mode of —
AD algorithm *see:* forward mode of an —; reverse mode of an —
AD-enabled parallelism
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
ad hoc networks *see:* Optimization in —
AD intermediate form
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
AD of parallel programs
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
AD tools *see:* parallel —
AD01
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
Adams–Johnson linearization
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
adaptation
(*see:* **Bayesian networks**)
adaptation *see:* subinterval —
adaptive aggregation method
[49L20, 90C39]
(*see:* **Dynamic programming: discounted problems**)
adaptive algorithm
[60J65, 68Q25]
(*see:* **Adaptive global search**)
adaptive algorithm
[60J65, 68Q25]
(*see:* **Adaptive global search**)
adaptive computational method
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
adaptive computational method
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
Adaptive convexification in semi-infinite optimization
(90C34, 90C33, 90C26, 65K05)
(*refers to:* **αBB algorithm**; **Bilevel optimization: feasibility test and flexibility index**; **Convex discrete optimization**; Generalized semi-infinite programming: optimality conditions)
adaptive) decision *see:* ex-post (risk prone —
Adaptive global search
(60J65, 68Q25)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Global optimization based on statistical models**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Global optimization based on statistical models**)
adaptive homotopy
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
adaptive homotopy
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
adaptive memory
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
adaptive methods
[49M37, 65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**; **Nonlinear least squares: Newton-type methods**)
adaptive methods
[49M37, 60J65, 68Q25]
(*see:* **Adaptive global search**; **Nonlinear least squares: Newton-type methods**)
adaptive partitioning
[65K05, 90C26, 90C30]
(*see:* **Bounding derivative ranges**; **Direct global optimization algorithm**)
adaptive random search method
[65K05, 90C30]
(*see:* **Random search methods**)
adaptive search
[65K05, 90C26, 90C30, 90C90]
(*see:* **Global optimization: hit and run methods**; **Random search methods**)
adaptive search
[90C08, 90C11, 90C26, 90C27, 90C90]
(*see:* **Biquadratic assignment problem**; **Global optimization: hit and run methods**)
adaptive search *see:* greedy randomized —; hesitant —; pure —
adaptive search procedure *see:* greedy randomized —
adaptive search procedures *see:* Greedy randomized —
adaptive simulated annealing
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
adaptive simulated annealing
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
Adaptive simulated annealing and its application to protein folding
(92C05)
(*referred to in:* **Adaptive global search**; **Bayesian global optimization**; **Genetic algorithms**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization in Lennard–Jones and morse clusters**; **Graph coloring**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Phase problem in X-ray**

crystallography: Shake and bake approach; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Adaptive global search**; **Bayesian global optimization**; **Genetic algorithms**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization in Lennard–Jones and morse clusters**; Global optimization in protein folding; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Protein folding: generalized-ensemble algorithms**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)

adaptive subdivision rule
[90C26]
(*see:* **D.C. programming**)

addition with order *see:* first order theory of real —

additional reverse convex constraint *see:* linear program with an —

additive tree
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)

additive utility functions
[90C26, 91B28]
(*see:* **Portfolio selection and multicriteria analysis**)

additive utility functions
[90C26, 90C29, 91B28]
(*see:* **Decision support systems with multiple criteria**; **Portfolio selection and multicriteria analysis**)

adic assignments problems *see:* N- —

aDIFOR
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: point and interval taylor operators**)

adjacency graph
[90B80]
(*see:* **Facilities layout problems**)

adjacency graph
[90B80]
(*see:* **Facilities layout problems**)

adjacency matrix
[05C15, 05C17, 05C35, 05C60, 05C69, 37B25, 90C20, 90C22, 90C27, 90C35, 90C59, 91A22]
(*see:* **Lovász number**; **Replicator dynamics in combinatorial optimization**)

adjacent
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Graph coloring**; **Lovász number**)

adjacent channel constrained frequency assignment
[05-XX]
(*see:* **Frequency assignment problem**)

adjacent vertices in a graph
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)

adjacent violators algorithm *see:* pool —

adjoint
[65K05, 65L99, 90C30, 93-XX]
(*see:* **Automatic differentiation: calculation of Newton steps**; **Optimization strategies for dynamic systems**)

adjoint-based gradient
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)

adjoint derivative method
[90C26, 90C90]
(*see:* **Structural optimization: history**)

adjoint equation *see:* extended —

adjoint linear map
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)

adjoint methods
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)

adjoint problem
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)

adjoint program
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)

adjoint recursion
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)

adjoint variables
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)

adjoints
[65H99, 65K99]
(*see:* **Automatic differentiation: point and interval**)

adjoints *see:* second order —

adjustment
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)

adjustment *see:* multiplier —; simultaneous —

adjustment process
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)

adjustment process *see:* trip-route choice —

admissible arc
[90C35]
(*see:* **Maximum flow problem**)

admissible cluster
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)

admissible displacement *see:* kinematically —

admissible domain
[49J20, 49J52]
(*see:* **Shape optimization**)

admissible pair of a monomial ideal
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
admissible pair of trajectory and control functions *see:* asymptotically —
admissible pair of trajectory-function and control-function
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
admissible pivot
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
admissible policy
[49L20, 90C39]
(*see:* **Dynamic programming: discounted problems**)
admissible solution
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
admissible solution
[90C15, 90C29]
(*see:* **Approximation of extremum problems with probability functionals**; **Discretely distributed stochastic programs: descent directions and efficient points**)
admissible space *see:* kinetically —
admissible trajectory-control pair
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
ADOL-C
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
ADOL-F
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: point and interval taylor operators**)
advance
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
advanced basis
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
advanced search heuristics
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
advanced warmstart
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
adversary
[05C85]
(*see:* **Directed tree networks**)
aEL
[74A40, 90C26]
(*see:* **Shape selective zeolite separation and catalysis: optimization methods**)
affine
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
affine equilibrium constraints *see:* mathematical program with —
affine function
[32B15, 51E15, 51N20, 90C26, 90C39]
(*see:* **Affine sets and functions**; **Second order optimality conditions for nonlinear optimization**)
affine functions *see:* product of —; program of minimizing a product of two —
affine-reduced-Hessian
[90C30]
(*see:* **Conjugate-gradient methods**)
affine reduced Hessian *see:* limited-memory —
affine-reduced-Hessian algorithm
[90C30]
(*see:* **Conjugate-gradient methods**)
affine reduction *see:* successive —
affine reduction BFGS algorithm *see:* successive —
affine scaling algorithm
[90C05]
(*see:* **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**)
affine scaling SQPIP methods
[49K20, 49M99, 90C55]
(*see:* **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
affine set
[32B15, 51E15, 51N20]
(*see:* **Affine sets and functions**)
Affine sets and functions
(51E15, 32B15, 51N20)
(*referred to in:* **Linear programming**; **Linear space**)
(*refers to:* **Convex max-functions**; **Linear programming**; **Linear space**)
after-arrival *see:* duty- —
afterset
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
afterset representation of relations *see:* foreset and —
against all *see:* one —
against one *see:* one —
agent *see:* principal —
agents *see:* mass separating —
aggregate excess demand function
[91B50]
(*see:* **Walrasian price equilibrium**)
aggregate excess demand function
[91B50]
(*see:* **Walrasian price equilibrium**)
aggregation
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
Aggregation
(*see:* **Optimal planning of offshore oilfield infrastructure**)
aggregation *see:* feature-based —; scenario —
aggregation function
[90C30, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**)
aggregation heuristic
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
aggregation method *see:* adaptive —

aggregation schemes
[90C30, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**)
Agmon–Motzkin–Fourier relaxation method
[90C25, 90C33, 90C55]
(*see:* **Splitting method for linear complementarity problems**)
agricultural risks
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
agricultural risks
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
agricultural systems *see:* State of the art in modeling —
agriculture
[90C29, 90C30, 90C90]
(*see:* **Decision support systems with multiple criteria**; **MINLP: applications in blending and pooling problems**)
ahead rules *see:* look- —
AHP
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
AHP
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
aHT
[74A40, 90C26]
(*see:* **Shape selective zeolite separation and catalysis: optimization methods**)
aid *see:* multicriteria decision —
*AIDA**
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
aided techniques *see:* computer —
AIF
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
air pollution
[90C05, 90C34]
(*see:* **Semi-infinite programming: methods for linear problems**)
air pollution
[90C05, 90C34]
(*see:* **Semi-infinite programming: methods for linear problems**)
air traffic control *see:* ground delay problem in —
air traffic control and ground delay programs
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
aircraft routing
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
aircraft routing
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
airline crew scheduling
[90C35]
(*see:* **Multicommodity flow problems**)
airline fleet assignment
[90C35]
(*see:* **Multicommodity flow problems**)
airline maintenance routing problem
[90C35]
(*see:* **Multicommodity flow problems**)
Airline optimization
(90B06, 90C06, 90C08, 90C35, 90C90)
(*referred to in:* **Integer programming**; **Vehicle scheduling**)
(*refers to:* **Integer programming**; **Vehicle scheduling**)
airplane hopping problem
[90C35]
(*see:* **Minimum cost flow problem**)
Aitken double sweep method
[90C30]
(*see:* **Cyclic coordinate method**)
Aitken double sweep method
[90C30]
(*see:* **Cyclic coordinate method**)
Aizenberg–Rabinovich system
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
Akaike information criterion
[62F10, 94A17]
(*see:* **Entropy optimization: parameter estimation**)
Alanine) *see:* Poly(L- —
algebra *see:* Boolean —; Boolean 2-valued logic —; computer —; Fundamental theorem of —; Lie —; linear —; many-valued logic —; MV- —; Orlik–Solomon —; Pi- —; Pinkava —; Pinkava logical —; relational interval —; V —; von Neumann —; W∗- —; Zhegalkin —
algebra connective *see:* logic —
algebra framework *see:* linear —
algebra package *see:* computer —
algebraic equations
[01A60, 03B30, 54C70, 68Q17]
(*see:* **Hilbert's thirteenth problem**)
algebraic equations
[01A60, 03B30, 54C70, 68Q17]
(*see:* **Hilbert's thirteenth problem**)
algebraic equations *see:* differential and —; linear —
algebraic expressions
(*see:* **Planning in the process industry**)
algebraic methods *see:* Integer programming: —
algebraic modeling language
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
algebraic modeling languages
(*see:* **Planning in the process industry**)
algebraic QAP
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
algebraic quadratic assignment problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)

algebraic statistics
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
algebraically decreasing tail *see:* RSM-distribution with —
algebras *see:* application of PI- —; complexity theory of PI- —; families of Pi- —; Finite complete systems of many-valued logic —; functional completeness of PI- —; functionally complete normal forms of Pi- —; many-valued families of the Pinkava logic —; PI-logic —; taxonomy of Pi-logic —; use of PI- —
algebras and 2-valued normal forms *see:* PI- —
algebras of many-valued logics *see:* taxonomy of the PI- —
algorithm
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Decomposition principle of linear programming**; **Modeling difficult optimization problems**)
algorithm *see:* A* search —; active set —; adaptive —; affine-reduced-Hessian —; affine scaling —; alpha-beta —; αBB —; αBB global optimization —; approximation —; asynchronous —; asynchronous parallel CA —; auction —; augmenting path —; Balas —; Bialas–Karwan Kth-best —; binary search —; branch and bound —; branch and contract —; branching —; Buchberger —; bundle —; cCOMB —; CG-related —; CGU —; clustering —; combinatorial —; complexity of an —; conical —; conjugate residual —; consistent labeling —; Conti–Traverso —; continuous-time equivalent of the dynamic programming —; convergent —; Convex-simplex —; copolyblock —; Corley–Moon —; Craig —; Craig conjugate gradient type —; cross decomposition —; cutting plane —; cycle-canceling —; cycling —; Dai–Yuan —; Daniel–Gragg–Kaufmann–Stewart reorthogonalized Gram–Schmidt —; descent —; descent in a nonlinear programming —; deterministic global optimization —; dimension-by-dimension —; Dinkelbach —; Direct global optimization —; discrete polyblock —; distributed game tree search —; division —; dual exterior point —; dual-scaling —; dual-scalings —; dual simplex —; dynamic programming —; efficient —; efficient polynomially bounded polynomial time —; EGOP —; ellipsoid —; Elzinga–Hearn —; EM —; entropic proximal point —; equilibration —; Esau–Williams —; evolutionary —; exact —; exact penalty function based —; expectation-maximization —; exponential —; exponential time —; Extended cutting plane —; extra-gradient —; Feed —; Fletcher–Reeves —; Ford–Fulkerson —; forward mode of an AD —; Frank–Wolfe —; Gauss–Seidel —; general —; general structure mixed integer αBB —; generalized bisection —; generalized game tree search —; generalized primal-relaxed dual —; generic augmenting path —; generic preflow-push —; generic vertex insertion —; globally convergent —; globally convergent probability-one homotopy —; GMIN-αBB —; Goldfarb–Wang —; Gomory cutting plane —; gOP —; gradient-free —; gradient-free minimization —; gradient projection —; graph collapsing auction —; greedy —; grouping genetic —; Gsat —; heavy ball —; Hestenes–Stiefel —; heuristic —; hide-and-seek —; high failure of the alpha-beta —; hit and run —; homogeneous —; Hopcroft–Tarjan planarity-testing —; Huang —; Hungarian —; hybrid —; implicit Choleski —; Implicit LU —; Implicit LX —; implicit QR —; incremental —; incremental-iterative solution —; incremental negamax —; infeasible-start interior-point —; Ingber —; interior point —; interval Newton —; Jacobi —; Jünger–Mutzel branch and cut —; K-iterated tour partitioning —; Karmarkar —; Kruskal —; Lanczos —; learning —; Lemke's —; Levenberg–Marquardt —; lexicographic search —; limited-memory —; limited-memory reduced-Hessian BFGS —; linear —; Linear programming: karmarkar projective —; low failure of the alpha-beta —; machine-learning —; mandatory work first —; Martin —; max–flow —; minimax —; minimum lower set —; MINLP: branch and bound global optimization —; MINLP: outer approximation —; modified Huang —; modified Kruskal —; modified Prim —; modified standard auction —; Monte-Carlo simulation —; multilevel —; naive auction —; NC —; nDOMB —; nearest insertion optimal partitioning —; Nelder–Mead —; network simplex —; nondeterministic polynomial —; nondeterministic polynomial time —; nonsmooth SSC-SABB —; on-line —; one clause at a time —; operator splitting —; optimal —; optimal state space search —; outer approximation —; P- —; parallel —; parallel minimax tree —; parallel routing —; parallel savings —; parallel-tangents —; Parametric linear programming: cost simplex —; parametric objective simplex —; parametric right-hand side simplex —; PARTAN —; partial proximal point —; path following —; perceptron —; pivot —; pivoting —; Piyavskii–Shubert —; Pnueli —; Polyak–Polak–Ribiére —; polyblock —; polynomial —; polynomial time —; polynomial time deterministic —; pool adjacent violators —; potential reduction —; potential smoothing —; predictor-corrector —; preflow-push —; primal-dual —; primal-dual potential reduction —; primal-dual scaling —; primal potential reduction —; primal-scaling —; primal simplex —; principal pivot —; principal variation splitting —; probabilistic analysis of an —; projected gradient —; projective —; proximal point —; pseudopolynomial —; pseudopolynomial time —; QPP —; quadratic proximal point —; RA —; randomized —; recursive —; recursive least squares —; recursive state space search —; reduced gradient —; regularized Frank–Wolfe —; regularized stochastic decomposition —; relative positioning —; relaxation —; relaxation labeling —; reverse mode of an AD —; reverse polyblock —; revised polyblock —; revised reverse polyblock (copolyblock) —; row-action —; Schaible —; sequential CA —; Sequential cutting plane —; sequential deterministic —; sequential minimax game tree —; shadow-vertex —; shake and bake —; simplex —; simplex type —; simulated annealing and genetic —; SMIN-αBB —; Smith–Walford-one —; special structure mixed integer αBB —; SQP type —; SSC-SABB —; sSC-SBB —; state space search —; steepest descent —; stochastic decomposition —; strongly polynomial —; strongly polynomial time —; subgradient projection —; successive affine reduction BFGS —; successive shortest path —; supervisor —; sweep —; synchronized distributed state space search —; synchronized parallel CA —; synchronous implementation of the auction —; TCF of an —; three phase —; three-term-recurrence —; time complexity function of

an —; totally asynchronous implementation of the auction —; tree-splitting —; truncated Buchberger —; unified —; UTASTAR —; variable-storage —; variant of the simplex —; virtual source —; weakly polynomial time —; Zangwill —
algorithm analysis *see:* nondegeneracy assumption for —
algorithm for axial MITPs *see:* greedy —
algorithm of complexity $\mathcal{O}(n^c)$
[90C60]
(*see:* **Computational complexity theory**)
algorithm (definition) *see:* optimization —
algorithm design
[05-04, 65K05, 65Y05, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**; **Parallel computing: models**)
algorithm design *see:* model for parallel —
algorithm for entropy optimization *see:* path following —
algorithm greedy-expanding
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
algorithm partition-flipping
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
algorithm partition-matching-I
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
algorithm polynomial of degree c
[90C60]
(*see:* **Computational complexity theory**)
algorithm pre-matching
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
algorithm and robust stopping criteria *see:* Dykstra's —
algorithm running in $\mathcal{O}(n^c)$ *time*
[90C60]
(*see:* **Computational complexity theory**)
algorithm-SCG
(*see:* **Railroad crew scheduling**)
algorithm for solving CAP on trees *see:* exact —
algorithm solving a problem instance in time m
[90C60]
(*see:* **Computational complexity theory**)
algorithm for weighted graph planarization *see:* branch and bound —
algorithmic approximation
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
algorithmic complexity
[90C60]
(*see:* **Kolmogorov complexity**)
Algorithmic complexity
[90C60]
(*see:* **Kolmogorov complexity**)
algorithmic definition
[65H99, 65K99]
(*see:* **Automatic differentiation: point and interval**)
algorithmic development
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)
algorithmic differentiation
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
algorithmic entropy
[90C60]
(*see:* **Kolmogorov complexity**)
Algorithmic entropy
[90C60]
(*see:* **Kolmogorov complexity**)
Algorithmic improvements using a heuristic parameter, reject index for interval optimization
(65K05, 90C30)
(*refers to:* **Interval analysis: unconstrained and constrained optimization**; **Interval Newton methods**)
algorithmic information
[90C60]
(*see:* **Kolmogorov complexity**)
Algorithmic information
[90C60]
(*see:* **Kolmogorov complexity**)
algorithmic knowledge
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
algorithmic language
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
algorithmic language
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
algorithmic randomness
[90C60]
(*see:* **Kolmogorov complexity**)
Algorithmic randomness
[90C60]
(*see:* **Kolmogorov complexity**)
algorithms
[05C69, 05C85, 49J35, 49K35, 49M37, 52B11, 52B45, 52B55, 62C20, 62G07, 62G30, 65K05, 65K10, 68Q25, 68W01, 90C26, 90C27, 90C30, 90C35, 90C59, 90C60, 91A05, 91A12, 91A40, 91B28]
(*see:* **αBB algorithm**; **Combinatorial optimization games**; **Competitive ratio for portfolio management**; **Graph coloring**; **Heuristics for maximum clique and independent set**; **Inequality-constrained nonlinear optimization**; **Isotonic regression problems**; **Minimax game tree searching**; **Volume computation for polytopes: strategies and performances**)
algorithms *see:* accelleration of —; approximation —; Asynchronous distributed optimization —; asynchronous iterative —; auction —; average case complexity of —; branch and bound —; bundle —; CA —; complexity theory of —; Cost approximation —; decomposition —; decomposition CA —; discrete-time —; efficient —; evolutionary —; exact —; fixed parameter tractable —; generic shortest path —; genetic —; geometric —; graph collapsing in auction —; graph reduction in auction —; greedy —; heuristic —; heuristic–metaheuristic —; hit and run —; inexact proximal point —; Integer programming: branch and cut —; Integer programming: cutting plane —; interior point —; local greedy —; local search —; memetic —; nonlinear CG-related —; numerical —;

optimal —; optimization —; pair assignment —; parallel —; polynomial time —; potential reduction —; primal and dual simplex —; Probabilistic analysis of simplex —; Protein folding: generalized-ensemble —; proximal —; random search —; randomized —; reducibility of —; robust —; scaled ABS class of —; search —; Shortest path tree —; simplicial —; Simplicial decomposition —; single assignment —; SLP —; smoothing —; solution —; SSC minimization —; Stable set problem: branch & cut —; Standard quadratic optimization problems: —; supervisor and searcher cooperation minimization —; threshold accepting —; training —; unconstrained optimization —; varying dimension pivoting —; virtual source concept in auction —
algorithms in combinatorial optimization *see:* Evolutionary —
algorithms and complexity *see:* Regression by special functions: —
algorithms for entropy optimization
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
algorithms for entropy optimization *see:* interior point —
algorithms for financial planning problems *see:* Global optimization —
algorithms for GAP *see:* approximation —
Algorithms for genomic analysis
(90C27, 90C35, 90C11, 65K05, 90-08, 90-00)
algorithms for hypodifferentiable functions *see:* Quasidifferentiable optimization: —
algorithms for integer programming *see:* Simplicial pivoting —
algorithms for isotonic regression problems
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
algorithms for linear equations and linear least squares *see:* Abaffi–Broyden–Spedicato —; ABS —
algorithms for linear programming generating two paths *see:* Pivoting —
algorithms for nonconvex minimization problems *see:* decomposition —
algorithms for nonsmooth and stochastic optimization *see:* SSC minimization —
algorithms for optimization *see:* ABS —
algorithms in pattern recognition *see:* Complementarity —
algorithms for protein structure prediction *see:* Genetic —
algorithms for QD functions *see:* Quasidifferentiable optimization: —
algorithms in resource allocation problems *see:* Combinatorial optimization —
algorithms and software *see:* Continuous global optimization: models —
algorithms for the solution of multistage mean-variance optimization problems *see:* Decomposition —
algorithms for stochastic bilevel programs
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
algorithms for stochastic linear programming problems *see:* Stabilization of cutting plane —
algorithms for the traveling salesman problem *see:* Heuristic and metaheuristic —
algorithms for unconstrained minimization
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
algorithms for unconstrained optimization *see:* New hybrid conjugate gradient —; Performance profiles of conjugate-gradient —
algorithms for the vehicle routing problem *see:* Metaheuristic —
aligned ellipsoid *see:* coordinate- —
alignment *see:* communication-free —; multiple sequence —; trace of an —
alignment constraint
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
alignment-distribution graph
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
alignment graph *see:* extended —
Alignment problem
(05-02, 05-04, 15A04, 15A06, 68U99)
(*referred to in:* **Integer programming**)
(*refers to:* **Integer programming**)
alignment problem
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
alignment problem
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
alignment problem *see:* communication-free —; constant degree parallelism —; solution of the —
alignment via mixed-integer linear optimization *see:* Global pairwise protein sequence —
all *see:* find one, find —; one against —
all-atom
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
all azeotropes *see:* Nonlinear systems of equations: application to the enclosure of —
all edge-directions of P *see:* covers —
all instances *see:* all-to- —; one-to- —
all-optical networks
[05C85]
(*see:* **Directed tree networks**)
all-to-all instances
[05C85]
(*see:* **Directed tree networks**)
allele
(*see:* **Broadcast scheduling problem**)
allocation *see:* facility location- —; marginal —; median location- —; MINLP: application in facility location- —; multifacility location- —; resource —; task —
allocation for epidemic control *see:* Resource —
allocation of gas
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
allocation model *see:* location- —
allocation phase
[90B80, 90B85]
(*see:* **Warehouse location problem**)

allocation problem *see:* discrete resource —; location- —; p-median location- —; resource —
allocation problems *see:* Combinatorial optimization algorithms in resource —
allocation scheme *see:* randomized —
allocation subproblem
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
allowed neighbor in tabu search
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
almost complementary solutions
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
almost empty spaces *see:* analyzing —
almost at equilibrium of an assignment and a set of prices
[90C30, 90C35]
(*see:* **Auction algorithms**)
along-rays functions on topological vector spaces *see:* Increasing and convex- —
alpha-beta algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
alpha-beta algorithm *see:* high failure of the —; low failure of the —
α-concave function
[90C15]
(*see:* **Logconcave measures, logconvexity**)
α-concave function
[90C15]
(*see:* **Logconcave measures, logconvexity**)
α-concave measure
[90C15]
(*see:* **Logconcave measures, logconvexity**)
α-cut of a fuzzy relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
α-divergence *see:* Csiszar —
alpha-helical proteins *see:* Predictive method for interhelical contacts in —
α-helix
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
α-helix
[92C40]
(*see:* **Monte-Carlo simulated annealing in protein folding**)
αBB
[49M37, 65K10, 90C26, 90C30, 92C40]
(*see:* **αBB algorithm**; **Multiple minima problem in protein folding: αBB global optimization approach**)
αBB *see:* GMIN- —; MINLP: global optimization with —; SMIN- —
αBB algorithm
(49M37, 65K10, 90C26, 90C30)
(*referred to in:* **Adaptive convexification in semi-infinite optimization**; **Bisection global optimization methods**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Convex envelopes in optimization problems**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Eigenvalue enclosures for ordinary differential equations**; **Generalized primal-relaxed dual approach**; **Global optimization based on statistical models**; **Global optimization in batch design under uncertainty**; **Global optimization in binary star astronomy**; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Global optimization using space filling**; **Hemivariational inequalities: eigenvalue problems**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Quadratic knapsack**; **Reverse convex optimization**; **Semidefinite programming and determinant maximization**; **Smooth nonlinear nonconvex optimization**; **Standard quadratic optimization problems: theory**; **Topology of global optimization**)
(*refers to:* **Bisection global optimization methods**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Convex envelopes in optimization problems**; **D.C. programming**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Eigenvalue enclosures for ordinary differential equations**; **Generalized primal-relaxed dual approach**; **Global optimization based on statistical models**; **Global optimization in batch design under uncertainty**; **Global optimization in binary star astronomy**; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Global optimization using space filling**; **Hemivariational inequalities: eigenvalue problems**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Reformulation-linearization technique for global optimization**; **Reverse convex optimization**; **Semidefinite programming and determinant maximization**; **Smooth nonlinear nonconvex optimization**; **Topology of global optimization**)
αBB algorithm
[49M37, 65K05, 65K10, 90C11, 90C26, 90C30]
(*see:* **αBB algorithm**; **MINLP: global optimization with αBB**)
αBB algorithm *see:* general structure mixed integer —; GMIN- —; SMIN- —; special structure mixed integer —
αBB approach *see:* Global optimization: g- —; Global optimization: p- —
αBB global optimization algorithm
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)
αBB global optimization approach *see:* Multiple minima problem in protein folding: —
alphabet *see:* finite —
alphabet of a Turing machine *see:* input —

alternance *see:* Chebyshev —
alternating procedure
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
alternating Turing machine
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
alternation *see:* Chebyshev —
alternative *see:* Linear optimization: theorems of the —; maximal —; set of decision —; theorem of the —
alternative linear system
[15A39, 90C05]
(*see:* **Linear optimization: theorems of the alternative**)
alternative and optimization *see:* Theorems of the —
Alternative set theory
(03E70, 03H05, 91B16)
(*referred to in:* **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
(*refers to:* **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
alternative set theory *see:* axioms of —
alternative systems
[15A39, 90C05]
(*see:* **Motzkin transposition theorem**)
alternative theorem
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
alternative theorem
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
alternative theorem *see:* basic —
alternatives *see:* finite set of the —; set of —; theorem of the —
alternatives to CG
[90C30]
(*see:* **Conjugate-gradient methods**)
amino acid
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
amino acid
[92B05, 92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms for protein structure prediction**)
analog of the dynamic programming equation *see:* continuous-time —
analyses *see:* post-optimality —
analysing declarative program structure
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
analysis *see:* abstract convex —; Algorithms for genomic —; applications of sensitivity —; approximation —; asymptotic —; automated Fortran program for nonlocal sensitivity —; average case —; cluster —; Combinatorial matrix —; competitive —; convex —; data envelopment —; decision —; dependence —; design —; discrete convex —; discriminant —; domination —; equilibrium —; exploratory statistical —; Financial applications of multicriteria —; functional —; infinitesimal perturbation —; interval —; investment —; linear Programming and Economic —; marginal —; matrix —; mean-variance portfolio —; model-based experimental —; monotonic —; multicriteria —; nondegeneracy assumption for algorithm —; nonlocal sensitivity —; nonsmooth —; nonstandard —; numerical —; perturbation —; Portfolio selection and multicriteria —; post-optimality —; post-optimality sensitivity —; preference disaggregation —; probabilistic —; range- —; regression —; relational —; robust stability —; robustness —; scenario —; sensitivity —; set-valued —; Shape reconstruction methods for nonconvex feasibility —; shape sensitivity —; Short-term scheduling under uncertainty: sensitivity —; stability —; target —; time series —; value —; worst-case —
analysis of an algorithm *see:* probabilistic —
analysis: application to chemical engineering design problems *see:* Interval —
analysis with automatic differentiation *see:* Nonlocal sensitivity —
analysis and balanced interval arithmetic *see:* Global optimization: interval —
analysis of cable structures *see:* structural —
analysis in combinatorial optimization *see:* Domination —
analysis of complementarity problems *see:* Sensitivity —
analysis: differential equations *see:* Interval —
analysis: eigenvalue bounds of interval matrices *see:* Interval —
analysis of flowsheets *see:* flexibility —; operability —
analysis: Fréchet subdifferentials *see:* Nonsmooth —
analysis: intermediate terms *see:* Interval —
analysis and management of environmental systems *see:* Global optimization in the —
analysis methodologies *see:* semantic —
analysis: nondifferentiable problems *see:* Interval —
analysis and optimization *see:* nonsmooth —
analysis for optimization of dynamical systems *see:* Interval —
analysis of optimization problems *see:* stability —
analysis: parallel methods for global optimization *see:* Interval —
analysis with respect to changes in cost coefficients *see:* sensitivity —
analysis with respect to right-hand side changes *see:* sensitivity —
analysis of simplex algorithms *see:* Probabilistic —
analysis Step
[90B15]
(*see:* **Evacuation networks**)
analysis: subdivision directions in interval branch and bound methods *see:* Interval —
analysis system *see:* stability of a structural —
analysis: systems of nonlinear equations *see:* Interval —
analysis: unconstrained and constrained optimization *see:* Interval —
analysis of variance *see:* one-way —
analysis of variational inequality problems *see:* Sensitivity —
analysis: verifying feasibility *see:* Interval —
analysis: weak stationarity *see:* Nonsmooth —

analytic center
[46N10, 49M20, 90-00, 90-08, 90C25, 90C47]
(*see:* **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**)
analytic center cutting plane method
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
analytic hierarchy process
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
analytic hierarchy process
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
analytical approximation of linear programming
[90C05, 90C25]
(*see:* **Young programming**)
analytical approximation of a linear programming problem
[90C05, 90C25]
(*see:* **Young programming**)
analytical differentiation
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
analytical tractability
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
analyzing almost empty spaces
(*see:* **Selection of maximally informative genes**)
anchor
(*see:* **Semidefinite programming and the sensor network localization problem, SNLP**)
anchor *see:* non- —
AND-ing
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
angle *see:* bond —; dihedral —
angle condition *see:* acute —; nonobtuse —; uniform —
angle method *see:* cutting —; Global optimization: cutting —
angle optimization *see:* beam —
angle selection *see:* beam —
angle selection and wedge orientation optimization *see:* beam —
angles *see:* direction —
angular form *see:* block —
angular structure *see:* block- —; dual block- —
annealed replication heuristic
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
annealing
[60J65, 68Q25, 90C27, 90C90]
(*see:* **Adaptive global search**; **Simulated annealing**)
annealing *see:* adaptive simulated —; Gaussian density —; Packet —; re- —; simulated —; simulating —; stochastic simulated —
annealing and genetic algorithm *see:* simulated —
annealing and its application to protein folding *see:* Adaptive simulated —
annealing methods in protein folding *see:* Simulated —
annealing in protein folding *see:* Monte-Carlo simulated —
annealing schedule
[90C27, 90C90]
(*see:* **Laplace method and applications to optimization problems**; **Simulated annealing**)
annealing temperature *see:* initial —
annexation *see:* polyhedral —
another) *see:* pseudomonotone bifunction (with respect to —
Ansatz *see:* reduction —
ant colony
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
ant system
[68T20, 68T99, 90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Metaheuristics**; **Quadratic assignment problem**)
ant system *see:* MAX-MIN —
ante (risk averse, anticipative) decision *see:* ex- —
anti-cycling procedure
[90C60]
(*see:* **Complexity of degeneracy**)
anti-Monge inequalities
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
anti-Monge matrix
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
anti-Robinson
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
anti-Robinson matrix
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
anticipative *see:* non- —
anticipative) decision *see:* ex-ante (risk averse —
anticycling
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Lexicographic pivoting rules**)
anticycling rules
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
anticycling rules *see:* Least-index —
antisymmetric partial order
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
antisymmetric relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
antisymmetric relation *see:* strictly —
antitone Boolean function
[90C09]
(*see:* **Inference of monotone boolean functions**)
antitone monotone Boolean function
[90C09]
(*see:* **Inference of monotone boolean functions**)
antitone operator
[90C33]
(*see:* **Order complementarity**)

APF
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
appearance of control function *see:* linear —
application to chemical engineering design problems *see:* Interval analysis: —
application to the enclosure of all azeotropes *see:* Nonlinear systems of equations: —
application in facility location-allocation *see:* MINLP: —
application to phase equilibrium problems *see:* Global optimization: —
application of PI-algebras
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
application process
[68T20, 68T99, 90C27, 90C59]
(*see:* **Capacitated minimum spanning trees**; **Metaheuristics**)
application to protein folding *see:* Adaptive simulated annealing and its —
applications
[49M37, 65K10, 90-01, 90B30, 90B50, 90C26, 90C27, 90C30, 91B06, 91B32, 91B52, 91B60, 91B74]
(*see:* **αBB algorithm**; **Bilevel programming in management**; **Financial applications of multicriteria analysis**; **Operations research and financial markets**)
applications *see:* Bilevel programming: —; Continuous global optimization: —; Dynamic programming: optimal control —; economic —; engineering —; Invexity and its —; medical —; minimization Methods for Non-Differentiable Functions and —; Multi-quadratic integer programming: models and —; multistage —; noneconomic —; Pseudomonotone maps: properties and —; Quasidifferentiable optimization: —; Robust linear programming with right-hand-side uncertainty, duality and —; scientific —; Standard quadratic optimization problems: —; Stochastic quasigradient methods: —
applications in blending and pooling problems *see:* MINLP: —
applications in distillation systems *see:* Successive quadratic programming: —
applications in engineering *see:* Bilevel programming: —
applications in environmental systems modeling and management
[90C05]
(*see:* **Global optimization in the analysis and management of environmental systems**)
applications in finance *see:* Semi-infinite programming and —
applications in the interaction of design and control *see:* MINLP: —
applications in mechanics
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
applications in mechanics *see:* Hemivariational inequalities: —
applications of multicriteria analysis *see:* Financial —
applications to optimization problems *see:* Laplace method and —
applications of parametric programming
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
applications in the process industry *see:* Successive quadratic programming: —
applications of sensitivity analysis
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
applications in the supply chain management *see:* Bilinear programming: —
applications to thermoelasticity *see:* Quasidifferentiable optimization: —
applications to variational inequalities and equilibrium problems *see:* Generalized monotonicity: —
approach *see:* Archimedian —; auction —; augmented Lagrangian decomposition —; axiomatic —; Bayesian —; Bayesian heuristic —; Benders decomposition —; Bilevel programming: implicit function —; closed form —; continuously differentiable exact penalty function —; cutting plane —; direct —; equation oriented —; Everett generalized Lagrange multiplier —; feasibility —; feasible —; feasible path —; Generalized primal-relaxed dual —; Global optimization: g-αBB —; Global optimization: p-αBB —; gradient based —; GRASP —; implicit function —; index —; infeasible path —; Kuhn–Tucker —; lexicographic —; limited-memory —; limited-memory symmetric rank-one —; material derivative —; Mixed-integer nonlinear optimization: A disjunctive cutting plane —; modified Cauchy —; modular —; Multiple minima problem in protein folding: αBB global optimization —; one clause at a time —; open form —; Optimization with equilibrium constraints: A piecewise SQP —; outranking relations —; parabolic curve —; parametric —; path following —; penalty —; Petrov–Galerkin —; Phase problem in X-ray crystallography: Shake and bake —; preference disaggregation —; primal-relaxed dual —; proximal point —; semidefinite programming —; simultaneous —; stochastic —; Stochastic programming: minimax —; subgraph —; Tikhonov's regularization —; trust region —; value function —; Variational inequalities: F. E. —; worst-case —
approach: basic features, examples from financial decision making *see:* Preference disaggregation —
approach to bilevel programming *see:* implicit function —
approach to clustering *see:* Nonsmooth optimization —
approach for DNA transcription element identification *see:* Mixed 0-1 linear programming —
approach to fractional optimization *see:* parametric —
approach: global optimum search with enhanced positioning *see:* Gene clustering: A novel decomposition-based clustering —
approach to image reconstruction from projection data *see:* feasibility —; optimization —
approach to optimality *see:* parametric —
approach to optimization *see:* Image space —
approach to optimization in water resources *see:* stochastic —
approach to solving CAP on trees *see:* heuristic —
approaches *see:* cutting plane —; equation based —; heuristic —; logic-based —; Optimal solvent design —; Statistical classification: optimization —
appropriateness
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)

approximate
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
approximate continuous
[90C10, 90C11, 90C27, 90C33]
(*see:* **Continuous reformulations of discrete-continuous optimization problems**)
approximate gradient *see:* ν- —
approximate inference *see:* interval-valued —
approximate inverse
[65H10, 65J15]
(*see:* **Contraction-mapping**)
approximate Jacobian
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
approximate methods for solving vehicle routing problems
[90B06]
(*see:* **Vehicle routing**)
approximate Newton method
[90C30]
(*see:* **Generalized total least squares**)
approximate optimization *see:* sequential —
approximate reasoning
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
approximate reasoning
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
approximate reasoning *see:* interval logic system of —; point-based logic system of —
approximate solutions of nonlinear systems of equations *see:* error bound for —
approximately *see:* exactly or —
approximating cone *see:* high-order —; tangent high-order —
approximating cone of decrease *see:* high-order —
approximating cones *see:* feasible high-order —; tangent high-order —
approximating curve *see:* feasible high-order —; high-order —; tangent high-order —
approximating the recourse function
[90C06, 90C15]
(*see:* **Stabilization of cutting plane algorithms for stochastic linear programming problems**)
approximating vector *see:* feasible high-order —; high-order tangent —
approximating vector of decrease *see:* high-order —
approximating vectors *see:* high-order —
approximation
[49J20, 49J52, 65H20, 65M60]
(*see:* **Multi-scale global optimization using terrain/funneling methods**; **Shape optimization**; **Variational inequalities: F. E. approach**)
approximation
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D10, 65D30, 65K05, 65K10, 90C15, 90C25, 90C26, 90C34, 90C35]
(*see:* **ABS algorithms for optimization**; **Approximation of multivariate probability integrals**; **Graph coloring**; **Multistage stochastic programming: barycentric approximation**; **Overdetermined systems of linear equations**; **Semi-infinite programming: numerical methods**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
approximation *see:* algorithmic —; barycentric —; best —; better —; Chebyshev best —; cost —; discrete —; ellipsoidal —; finite-difference —; finite element —; Generalized outer —; hybrid branch and bound and outer —; inner —; linear —; linear outer —; logic of —; Logic-based outer —; maximal best —; mean field —; minimal best —; mixed finite element —; multipoint —; Multistage stochastic programming: barycentric —; outer —; Padé —; Padé-type —; perturbative —; point-based —; polyblock —; polynomial of best —; proximal —; quadratic outer —; second order —; Sensitivity and stability in NLP: —; stochastic —; successive —; truncated Taylor —
approximation algorithm
[90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**)
approximation algorithm *see:* MINLP: outer —; outer —
approximation algorithms
[05C05, 05C85, 68Q25, 90B06, 90B35, 90B80, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Bottleneck steiner tree problems**; **Directed tree networks**; **Traveling salesman problem**)
approximation algorithms
[03B05, 05C05, 05C85, 68P10, 68Q25, 68R05, 68T15, 68T20, 90B80, 90C09, 90C27, 90C35, 90C60, 90C90, 94C10]
(*see:* **Bottleneck steiner tree problems**; **Complexity theory: quadratic programming**; **Maximum satisfiability problem**; **Multi-index transportation problems**; **Simulated annealing**; **Steiner tree problems**)
approximation algorithms *see:* Cost —
approximation algorithms for GAP
[90-00]
(*see:* **Generalized assignment problem**)
approximation Analysis
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
approximation by bounded or continuous functions *see:* Lipschitzian operators in best —
approximation with equality relaxation *see:* outer —
approximation with equality relaxation and augmented penalty *see:* outer —
Approximation of extremum problems with probability functionals
(90C15)
(*referred to in:* **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic**

programming models; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
approximation of a function *see:* first order —
approximation of linear programming *see:* analytical —
approximation of a linear programming problem *see:* analytical —
approximation measure *see:* contraction/ —
approximation method *see:* Logic-based outer- —; outer —; polyblock —; Vogel —
approximation methods *see:* Gaussian —; Semi-infinite programming: —
Approximation of multivariate probability integrals (65C05, 65D30, 65Cxx, 65C30, 65C40, 65C50, 65C60, 90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
approximation of nonsmooth mappings [49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
approximation operator *see:* best —

approximation in ordered normed linear spaces *see:* Best —
approximation to the problem
[41A30, 47A99, 65K10, 93-XX]
(*see:* **Boundary condition iteration BCI**; **Lipschitzian operators in best approximation by bounded or continuous functions**)
approximation problem *see:* simultaneous Diophantine —
approximation ratio
[05C05, 05C85, 68Q25, 90B80]
(*see:* **Bottleneck steiner tree problems**; **Directed tree networks**)
approximation scheme *see:* fully polynomial time —; polynomial time —
approximation of space filling curves
[90C26]
(*see:* **Global optimization using space filling**)
approximation techniques
[90C15]
(*see:* **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
approximation in the uniform norm
[90C34]
(*see:* **Semi-infinite programming: approximation methods**)
approximation of variational inequalities
[65M60]
(*see:* **Variational inequalities: F. E. approach**)
approximations *see:* nonsmooth local —
approximations of nonsmooth mappings
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
Approximations to robust conic optimization problems
approximations to subdifferentials *see:* Continuous —
approximator
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
aquifers
[90C30, 90C35]
(*see:* **Optimization in water resources**)
arbitrage pricing theory
[91B50]
(*see:* **Financial equilibrium**)
arbitrary
[90C60]
(*see:* **Complexity classes in optimization**)
arbitrary multivariate distributions *see:* Stochastic linear programs with recourse and —
arborescence *see:* minimum Steiner —; Steiner —
arborescence problem *see:* capacitated minimum spanning —
arborescence system *see:* multi-echelon —
arborescence tree *see:* rectilinear Steiner —
arboricity
[90C35]
(*see:* **Fractional zero-one programming**; **Optimization in leveled graphs**)
arc *see:* admissible —; arrival-ground connection —; backward —; central —; conjunction —; disjunction —; dual —; endpoint of an —; entering —; forward —; ground-departure connection —; inadmissible —; incoming —; multiplier associated with an —; network —; outgoing —; primal —; root —; train —
arc capacity
[90C35]
(*see:* **Maximum flow problem**)
arc coloring
[05C85]
(*see:* **Directed tree networks**)
arc consistency
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
arc construction procedure *see:* best —
arc cost *see:* piecewise linear —
arc cost function *see:* sawtooth —; staircase —
(arc) deletion problem *see:* vertex —
arc in a directed network *see:* directed —; endpoint of an —
arc flow bounds
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)
arc flows *see:* capacity constraint on —
arc formulation *see:* node- —
arc formulation of the problem *see:* node- —
arc incidence matrix *see:* node- —
arc legend
(*see:* **Railroad crew scheduling**)
arc length vector
[90C31, 90C39]
(*see:* **Multiple objective dynamic programming**)
arc in a network *see:* capacity of an —; cost of an —; directed —
arc oriented branch and bound method
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
arc oriented construction procedure
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
arc routing
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
arc routing
[90B06]
(*see:* **Vehicle routing**)
arc routing problem *see:* capacitated —
arc separation procedure
[90B10]
(*see:* **Piecewise linear network flow problems**)
(arc) set problem *see:* feedback —; minimum feedback —; minimum feedback vertex —; minimum weight feedback —; subset feedback vertex —; subset minimum feedback vertex —
Archimedes and the foundations of industrial engineering (01A20)
archimedian
(*see:* **Planning in the process industry**)
Archimedian approach
(*see:* **Planning in the process industry**)
architecture *see:* selection of —; von Neumann —
archive
[34-xx, 34Bxx, 34Lxx, 93E24]
(*see:* **Complexity and large-scale least squares problems**)
arcs *see:* bold —; critical —; deadhead —; demand —; ground —; natural stream —; rest —; sequence of —; train-train connection —

are under control *see:* rounding errors —
area computer network *see:* local- —
areas *see:* software package for specific mathematical —
argon atoms
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
argument *see:* noncompensatory —; ordinal —
argument function *see:* four- —; three- —
argument principle
[01A50, 01A55, 01A60]
(*see:* **Fundamental theorem of algebra**)
argument principle
[01A50, 01A55, 01A60]
(*see:* **Fundamental theorem of algebra**)
arithmetic *see:* balanced interval —; balanced random interval —; differentiation —; Global optimization: interval analysis and balanced interval —; inclusion principle of machine interval —; inner interval —; Interval —; Kaucher —; machine interval —; random interval —; slope —
arithmetic degree of a monomial ideal
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
arithmetic operation *see:* interval —
arithmetic operations on fuzzy numbers
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
arity of a constraint
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
arm *see:* flexible —; Optimal control of a flexible —
armed restless bandit problem *see:* multi- —
Armijo-like criterion *see:* test nonmonotone —
Armijo rule
[90C30]
(*see:* **Convex-simplex algorithm**; **Cost approximation algorithms**)
Armijo steplength rule
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
Arora PTAS
[90C27]
(*see:* **Steiner tree problems**)
ARR
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
arr-station
(*see:* **Railroad locomotive scheduling**)
arr-time
(*see:* **Railroad locomotive scheduling**)
arrangement
[05B35, 20F36, 20F55, 26A24, 52C35, 57N65, 65K99, 85-08]
(*see:* **Automatic differentiation: geometry of satellites and tracking stations**; **Hyperplane arrangements in optimization**)
arrangement *see:* face of an —; hyperplane —; linear —; polygonal —; simple —; two Polygons —
arrangement of hyperplane
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
arrangement of hyperplanes *see:* Boolean —; braid —; cohomology of an —; complement of an —; divisor of an —; free —; reflection —; singularity of an —
arrangement problem *see:* linear —
arrangements *see:* Hyperplane —
arrangements in optimization *see:* Hyperplane —
Arrhenius constants
[90C30, 90C52, 90C53, 90C55]
(*see:* **Gauss–Newton method: Least squares, relation to Newton's method**)
arrival *see:* duty-after- —
arrival-ground connection arc
(*see:* **Railroad locomotive scheduling**)
arrival-ground node
(*see:* **Railroad locomotive scheduling**)
arrival node
(*see:* **Railroad locomotive scheduling**)
arrival-station
(*see:* **Railroad crew scheduling**)
Arrow–Hurwicz gradient method
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
art in modeling agricultural systems *see:* State of the —
artificial centering hit and run
[90C26, 90C90]
(*see:* **Global optimization: hit and run methods**)
artificial intelligence
[65G20, 65G30, 65G40, 65K05, 68T20, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C26, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**; **Forecasting**; **Interval constraints**)
artificial intelligence
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 90C26, 90C30, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Forecasting**)
ary relation *see:* n- —
AS
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
as conic convex program *see:* semidefinite program —
ASA
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
ascendant direction
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
ascendant direction *see:* feasible —
ascent *see:* dual —; rate of steepest —; rule of steepest —
ascent direction *see:* Dini steepest —; Hadamard steepest —; steepest —
ascent flow
[58E05, 90C30]
(*see:* **Topology of global optimization**)
asking strategy *see:* binary search-Hansel chains question- —; question- —; sequential Hansel chains question- —

ASOG equation
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
aspatial oligopoly problem
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
aspatial and spatial markets
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
aspiration criteria
[68M20, 90B06, 90B35, 90B80, 90C59]
(*see:* **Flow shop scheduling problem**; **Heuristic and metaheuristic algorithms for the traveling salesman problem**; **Location routing problem**)
aspiration level
[05C69, 05C85, 68W01, 90C29, 90C59]
(*see:* **Heuristics for maximum clique and independent set**; **Multiple objective programming support**)
aspiration search
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
aspiration search *see:* parallel —
asplund
[49K27, 58C20, 58E30, 90C46, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**; **Nonsmooth analysis: weak stationarity**)
assembly crossover (EAX) *see:* edge —
assessment *see:* comparative efficiency —; fuzzy truth —
asset *see:* risk-free —
asset Liability Management
[65K99, 90C29]
(*see:* **Asset liability management decision support system**)
Asset liability management decision support system
(90C29, 65K99)
asset pricing model *see:* capital —
asset selling problem
[49L20, 90C39]
(*see:* **Dynamic programming: discounted problems**)
assignment
[90C29]
(*see:* **Decision support systems with multiple criteria**; **Mixed integer programming/constraint programming hybrid methods**)
assignment
[90C10, 90C35]
(*see:* **Bi-objective assignment problem**)
assignment *see:* adjacent channel constrained frequency —; airline fleet —; co-channel constrained frequency —; discrete location and —; dynamic traffic —; feasible —; fleet —; free —; optimal —; order of a T-coloring frequency —; partial —; quadratic —; single —; span of a T-coloring frequency —; traffic —; variable —
assignment algorithms *see:* pair —; single —
assignment constraints *see:* semi- —
Assignment and matching
(90C35, 90C27, 90C10, 90C05)
(*referred to in:* **Assignment methods in clustering**; **Bi-objective assignment problem**; **Communication network assignment problem**; **Frequency assignment problem**; **Linear ordering problem**; **Maximum partition matching**; **Multidimensional assignment problem**; **Quadratic assignment problem**)
(*refers to:* **Assignment methods in clustering**; **Bi-objective assignment problem**; **Communication network assignment problem**; **Frequency assignment problem**; **Maximum partition matching**; **Quadratic assignment problem**)
assignment method
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
Assignment methods in clustering
(62H30, 90C27)
(*referred to in:* **Assignment and matching**; **Bi-objective assignment problem**; **Communication network assignment problem**; **Frequency assignment problem**; **Linear ordering problem**; **Maximum partition matching**; **Quadratic assignment problem**)
(*refers to:* **Assignment and matching**; **Bi-objective assignment problem**; **Communication network assignment problem**; **Frequency assignment problem**; **Maximum partition matching**; **Quadratic assignment problem**)
assignment model
[68M20, 90B06, 90B10, 90B35, 90B80, 90B85, 90C10, 90C27]
(*see:* **Single facility location: multi-objective euclidean distance location**; **Vehicle scheduling**)
assignment model *see:* the multi-resource weighted —; quasi- —
assignment models *see:* locomotive —
assignment problem
[68Q25, 68R10, 68W40, 90B85, 90C05, 90C06, 90C08, 90C10, 90C11, 90C27, 90C30, 90C35, 90C59, 90C60]
(*see:* **Assignment and matching**; **Auction algorithms**; **Bi-objective assignment problem**; **Complexity of degeneracy**; **Domination analysis in combinatorial optimization**; **Integer programming: cutting plane algorithms**; **Single facility location: multi-objective euclidean distance location**)
assignment problem
[68Q25, 90B80, 90C05, 90C27, 90C30, 90C35]
(*see:* **Auction algorithms**; **Communication network assignment problem**)
assignment problem *see:* algebraic quadratic —; Asymptotic properties of random multidimensional —; Bi-objective —; Biquadratic —; bottleneck quadratic —; Communication network —; fleet —; Frequency —; general quadratic —; generalized —; Koopmans–Beckmann quadratic —; Multidimensional —; multilevel generalized —; multiperiod —; optimal —; order preserving —; quadratic —; Quadratic semi- —; radio link frequency —; traffic —
assignment problems *see:* multi-index —; nonlinear —; three-index —
assignment property *see:* single —
assignment ranking
[90C60]
(*see:* **Complexity of degeneracy**)
assignment and a set of prices *see:* almost at equilibrium of an —; equilibrium of an —
assignment of wavelengths
[05C85]
(*see:* **Directed tree networks**)
assignments problems *see:* N-adic —

associated with an arc *see:* multiplier —
associated with Λ *see:* canonical function —
Association *see:* atlas of the International Zeolite —
association graph
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
association graph *see:* tree —; weighted tree —
association problem *see:* data- —
associative connective
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
associativity of products of relations *see:* pseudo- —
assumption *see:* backlog —; human rationality —; lost sales —; nondegeneracy —; separability —
assumption for algorithm analysis *see:* nondegeneracy —
assumption stability
[90C31]
(*see:* **Sensitivity and stability in NLP: continuity and differential stability**)
assumptions *see:* regularity —; under weak —
AST
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
astrodynamics
[26A24, 65K99, 85-08]
(*see:* **Automatic differentiation: geometry of satellites and tracking stations**)
Astrodynamics
[26A24, 65K99, 85-08]
(*see:* **Automatic differentiation: geometry of satellites and tracking stations**)
astronomical problem
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
astronomy
[49M37, 65K10, 90C26, 90C30, 90C90]
(*see:* **αBB algorithm**; **Global optimization in binary star astronomy**)
astronomy *see:* Global optimization in binary star —
asymmetric Traveling Salesman Problem (ATSP)
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
asymmetric TSP
[90C59]
(*see:* **Heuristic and metaheuristic algorithms for the traveling salesman problem**)
asymmetric TSP (ATSP)
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
asymmetrical information
[68Q25, 91B28]
(*see:* **Competitive ratio for portfolio management**)
asymptotic analysis
[90C10, 90C15]
(*see:* **Stochastic integer programs**)
asymptotic analysis
[90Cxx]
(*see:* **Discontinuous optimization**)
asymptotic behavior
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
asymptotic behavior
[62C10, 65K05, 68Q25, 90B80, 90C05, 90C10, 90C15, 90C26, 90C27]
(*see:* **Bayesian global optimization**; **Communication network assignment problem**)
asymptotic behavior of CAP on trees
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
asymptotic case of integral evaluation
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
asymptotic convergence
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
asymptotic convergence rates
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
asymptotic CQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
Asymptotic properties of random multidimensional assignment problem
(90C27, 34E05)
asymptotic results for RSM-distributions
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
asymptotic stability
[90C30]
(*see:* **Suboptimal control**)
asymptotical stability at an equilibrium
[90B15]
(*see:* **Dynamic traffic networks**)
asymptotical stability of a system
[90B15]
(*see:* **Dynamic traffic networks**)
asymptotical system stability
[90B15]
(*see:* **Dynamic traffic networks**)
asymptotically admissible pair of trajectory and control functions
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
asymptotically stable
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
asymptotically stable stationary point
[90C20]
(*see:* **Standard quadratic optimization problems: algorithms**)
asynchronous algorithm
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
asynchronous computation
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**; **Cost approximation algorithms**)
asynchronous computation
[90C30]
(*see:* **Cost approximation algorithms**)

asynchronous computation *see:* partially —
asynchronous convergence theorem
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
Asynchronous distributed optimization algorithms
(90C30, 90C30, 90C52, 90C53, 90C55)
(*referred to in:* **Automatic differentiation: parallel computation**; **Heuristic search**; **Load balancing for parallel optimization techniques**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Parallel heuristic search**; **Stochastic network problems: massively parallel solution**)
(*refers to:* **Automatic differentiation: parallel computation**; **Heuristic search**; **Interval analysis: parallel methods for global optimization**; **Load balancing for parallel optimization techniques**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Parallel heuristic search**; **Stochastic network problems: massively parallel solution**)
asynchronous implementation of the auction algorithm *see:* totally —
asynchronous iterative algorithms
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
asynchronous iterative method *see:* partially —
asynchronous operation *see:* partially —; totally —
asynchronous parallel CA algorithm
[90C30]
(*see:* **Cost approximation algorithms**)
asynchronous round robin balancing scheme
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
Atkinson–Brakhage preconditioner
[65H10, 65J15]
(*see:* **Contraction-mapping**)
atlas of the International Zeolite Association
[74A40, 90C26]
(*see:* **Shape selective zeolite separation and catalysis: optimization methods**)
atom *see:* all- —
ATOMFT
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
atoms *see:* argon —
(ATSP) *see:* asymmetric Traveling Salesman Problem —; asymmetric TSP —
attentive convergence *see:* f- —
attraction *see:* basins of —; region of —
attraction and repulsion *see:* Global optimization in Weber's problem with —; Weber problem with —
attractor *see:* local —
attractors *see:* singular local —
attractors for gradient-related descent iterations *see:* Local —
attribute utility theory *see:* multi- —
attributed tree
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
auction
[90C30, 90C35]
(*see:* **Auction algorithms**)
auction algorithm
[90C30, 90C35]
(*see:* **Auction algorithms**)
auction algorithm *see:* graph collapsing —; modified standard —; naive —; synchronous implementation of the —; totally asynchronous implementation of the —
Auction algorithms
(90C30, 90C35)
(*referred to in:* **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
(*refers to:* **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
auction algorithms
[90B10, 90C27, 90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**; **Shortest path tree algorithms**)
auction algorithms *see:* graph collapsing in —; graph reduction in —; virtual source concept in —
auction approach
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
auction technique
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
auction technique
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
auditing decisions *see:* Multicriteria decision support methodologies for —
augmentation *see:* planar —
augmentation oracle
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
augmentation oracle *see:* oriented —
augmented Lagrange functions
[90C30]
(*see:* **Nonlinear least squares problems**)
augmented Lagrangian
[90C25, 90C30, 90C33]
(*see:* **Implicit lagrangian**; **Lagrangian multipliers methods for convex programming**)
augmented Lagrangian decomposition approach
[90C30, 90C35]
(*see:* **Optimization in water resources**)

augmented Lagrangian function
[90C30]
(*see:* **Image space approach to optimization**)
augmented Lagrangian methods *see:* Practical —
augmented Lagrangians
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
augmented network
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)
augmented penalty *see:* outer approximation with equality relaxation and —
augmented performance index
[93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**)
augmenting flows
[90C35]
(*see:* **Minimum cost flow problem**)
augmenting path algorithm
[90C35]
(*see:* **Maximum flow problem**)
augmenting path algorithm
[90C35]
(*see:* **Maximum flow problem**)
augmenting path algorithm *see:* generic —
augmenting vector
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
automata *see:* theory of —
automated design optimization process
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
automated Fortran program for nonlocal sensitivity analysis
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
automated hypothesis formation
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
automatic classification of documents
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
automatic differentiation
[26A24, 34-XX, 49-04, 49-XX, 65-XX, 65D25, 65G20, 65G30, 65G40, 65H20, 65K05, 65K99, 65L99, 65Y05, 68-XX, 68N20, 68W30, 85-08, 90-XX, 90C26, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: root problem and branch problem**; **Bounding derivative ranges**; **Complexity of gradients, Jacobians, and Hessians**; **Interval analysis: differential equations**; **Interval analysis: intermediate terms**; **Interval global optimization**; **Nonlocal sensitivity analysis with automatic differentiation**)
automatic differentiation
[26A24, 34-XX, 49-04, 49-XX, 65-XX, 65D25, 65K05, 65K99, 65Y05, 68-XX, 68N20, 68W30, 85-08, 90-XX, 90C26, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Bounding derivative ranges**; **Complexity of gradients, Jacobians, and Hessians**; **Nonlocal sensitivity analysis with automatic differentiation**)
automatic differentiation *see:* backward mode in —; forward mode of —; interval —; Nonlocal sensitivity analysis with —; reverse mode —; vector forward —
Automatic differentiation: calculation of the Hessian
(90C30, 65K05)
(*referred to in:* **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Nonlocal sensitivity analysis with automatic differentiation**)
(*refers to:* **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Nonlocal sensitivity analysis with automatic differentiation**)
Automatic differentiation: calculation of Newton steps
(90C30, 65K05)
(*referred to in:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Interval Newton methods**; **Nondifferentiable optimization: Newton method**; **Nonlinear least squares: Newton-type methods**; **Nonlocal sensitivity analysis with automatic differentiation**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
(*refers to:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**;

Automatic differentiation: root problem and branch problem; **Dynamic programming and Newton's method in unconstrained optimal control**; **Interval Newton methods**; **Nondifferentiable optimization: Newton method**; **Nonlocal sensitivity analysis with automatic differentiation**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)

Automatic differentiation: geometry of satellites and tracking stations
(26A24, 65K99, 85-08)
(*referred to in:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Nonlocal sensitivity analysis with automatic differentiation**)
(*refers to:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Nonlocal sensitivity analysis with automatic differentiation**)

Automatic differentiation: introduction, history and rounding error estimation
(65D25, 26A24)
(*referred to in:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Interval analysis: intermediate terms**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Nonlocal sensitivity analysis with automatic differentiation**; **Unconstrained optimization in neural network training**)
(*refers to:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Nonlocal sensitivity analysis with automatic differentiation**)

Automatic differentiation: parallel computation
(65Y05, 68N20, 49-04)
(*referred to in:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Heuristic search**; **Load balancing for parallel optimization techniques**; **Nonlocal sensitivity analysis with automatic differentiation**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Parallel heuristic search**; **Stochastic network problems: massively parallel solution**)
(*refers to:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Heuristic search**; **Interval analysis: parallel methods for global optimization**; **Load balancing for parallel optimization techniques**; **Nonlocal sensitivity analysis with automatic differentiation**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Parallel heuristic search**; **Stochastic network problems: massively parallel solution**)

Automatic differentiation: point and interval
(65H99, 65K99)
(*referred to in:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Nonlocal sensitivity analysis with automatic differentiation**)
(*refers to:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis:**

nondifferentiable problems; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Nonlocal sensitivity analysis with automatic differentiation**)

Automatic differentiation: point and interval taylor operators (65K05, 90C30)
(*referred to in:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: root problem and branch problem**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Nonlocal sensitivity analysis with automatic differentiation**)
(*refers to:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: root problem and branch problem**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Nonlocal sensitivity analysis with automatic differentiation**)

Automatic differentiation: root problem and branch problem (65K05)
(*referred to in:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Nonlocal sensitivity analysis with automatic differentiation**)
(*refers to:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Nonlocal sensitivity analysis with automatic differentiation**)

automatic document classification
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)

automatic graph drawing
[90C35]
(*see:* **Optimization in leveled graphs**)

automatic parallelization
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)

automatic result verification
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval fixed point theory**; **Interval Newton methods**)

automation
[93D09]
(*see:* **Robust control**)

autonomy
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)

auxiliary function
[65K05, 68W10, 90B15, 90C06, 90C30]
(*see:* **Direct global optimization algorithm**; **Stochastic network problems: massively parallel solution**)

auxiliary problem principle
[90C30]
(*see:* **Cost approximation algorithms**)

auxiliary problem principle
[90C30]
(*see:* **Cost approximation algorithms**)

auxiliary variable
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)

availability *see:* upper bound on gas lift —

average
[65H20]
(*see:* **Multi-scale global optimization using terrain/funneling methods**)

average *see:* on —

average behavior
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)

average case analysis
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
average case behavior
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
average case complexity
[60J65, 68Q25]
(*see:* **Adaptive global search**)
average case complexity
[60J65, 68Q25]
(*see:* **Adaptive global search**)
average case complexity of algorithms
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
average case setting
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
average cost per stage
[49L99]
(*see:* **Dynamic programming: average cost per stage problems**)
average cost per stage problem
[49L20, 90C39, 90C40]
(*see:* **Dynamic programming: infinite horizon problems, overview**)
average cost per stage problems
[49L99]
(*see:* **Dynamic programming: average cost per stage problems**)
average cost per stage problems *see:* Dynamic programming: —
average model *see:* moving —
average nonredundancy rate
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
average number of pivot steps
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
average redundancy rate
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
average rmsds by energy
[92C05, 92C40]
(*see:* **Protein loop structure prediction methods**)
averaged Navier–Stokes code *see:* Reynolds- —
averaging
[55R15, 55R35, 65K05, 90C11]
(*see:* **Deterministic and probabilistic optimization models for data classification**)
averaging method
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
averaging operation
[90C15]
(*see:* **Stochastic quasigradient methods: applications**)
averaging operation
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)
averse, anticipative) decision *see:* ex-ante (risk —
avoidance
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
A.W. *see:* Tucker —
away direction
[90C30]
(*see:* **Frank–Wolfe algorithm**)
away terminals
(*see:* **Railroad crew scheduling**)
axes of coordinates
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
axial MITPs *see:* greedy algorithm for —; hub heuristics for —
axial multi-index transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)
axiom *see:* choice —; existence of classes —; extensionality —; induction —; prolongation —; regularity —; two cardinalities —
axiom of extensionality
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
axiom of prolongation
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
axiom systems for oriented matroids
[90C09, 90C10]
(*see:* **Oriented matroids**)
axiomatic approach
[90C30]
(*see:* **Global optimization based on statistical models**)
axiomatic derivation of cross-entropy
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
axiomatic derivation of entropy
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
axiomatic derivation of the principle of maximum entropy
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
axiomatic derivation of the principle of minimum cross-entropy
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
axioms *see:* existence of sets —; painting —
axioms of alternative set theory
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
azeotrope
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
azeotrope
[90C30]

(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
azeotropes *see:* heterogeneous —; Nonlinear systems of equations: application to the enclosure of all —; reactive —
azuma's inequality
[05C85]
(*see:* **Directed tree networks**)

B

B *see:* gas lift wells of type —; level —; model —; naturally flowing wells of type —
B-derivative
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
b-matching
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
b-matching problem
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
b-matching problem *see:* perfect —
B-subdifferential
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
B well *see:* type —
B wells *see:* type —
BA
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
back-off
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)
back propagation
[90C15]
(*see:* **Stochastic quasigradient methods: applications**)
backboard wiring problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
backhauls
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
backhauls *see:* vehicle routing problem with —
backlog assumption
[49L20]
(*see:* **Dynamic programming: inventory control**)
backpropagation method
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
backsolving *see:* Gaussian elimination with —
backtrack
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
backtrack phases
[90C35]
(*see:* **Graph coloring**)
backtracking
[90C10, 90C29, 90C30]
(*see:* **Multi-objective integer linear programming**; **Nonlinear least squares problems**)
backtracking *see:* depth-first search with —
backward arc
[90C35]
(*see:* **Maximum flow problem**)
backward compatibility
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
backward mode in automatic differentiation
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
bad derivatives *see:* method of —
Baire measure
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
bake algorithm *see:* shake and —
bake approach *see:* Phase problem in X-ray crystallography: Shake and —
balance *see:* power —
balance constraints *see:* mass —
balance equation
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
balance equations *see:* mass and energy —; node flow —
balance equations for material flows
(*see:* **Planning in the process industry**)
balance sheet
[91B50]
(*see:* **Financial equilibrium**)
balanced interval arithmetic
[65G30, 65G40, 65K05, 90C30, 90C57]
(*see:* **Global optimization: interval analysis and balanced interval arithmetic**)
balanced interval arithmetic *see:* Global optimization: interval analysis and —
balanced node
[90C35]
(*see:* **Minimum cost flow problem**)
balanced random interval arithmetic
[65G30, 65G40, 65K05, 90C30, 90C57]
(*see:* **Global optimization: interval analysis and balanced interval arithmetic**)
balances *see:* mass —; mass, energy and momentum —
balancing *see:* dynamic load —; load —; static load —
balancing objectives
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**)
balancing for parallel optimization techniques *see:* Load —
balancing problem *see:* turbine —
balancing scheme *see:* asynchronous round robin —; near-neighbor load —
balancing technique *see:* dynamic load —
Balas algorithm
[90C10, 90C29]
(*see:* **Multi-objective integer linear programming**)

ball algorithm *see:* heavy —
ball-constrained linear problem
[37A35, 90C05]
(*see:* **Potential reduction methods for linear programming**)
ball constraint
[90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**)
ball method *see:* heavy —
ball quotient
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
Banach linear extension theorem *see:* Hahn– —
Banach space *see:* monotone operator on a —
Banach spaces *see:* Steiner ratio in —
Banach theorem *see:* Hahn– —; Mazur–Orlicz version of the Hahn– —
banded matrix
[65Fxx]
(*see:* **Least squares problems**)
bandit problem *see:* multi-armed restless —
Bandler–Kohout compatibility theorem
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
bandwidth
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
bandwidth *see:* lower —; optical —; row —; upper —
bandwidth of interdisciplinary coupling
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
bandwidth packing problem
[90C35]
(*see:* **Multicommodity flow problems**)
bandwidth problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
bank *see:* first —
bar of a truss *see:* elastic —
BARON
[90C11]
(*see:* **MINLP: branch and bound methods**)
barrier
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
barrier function
[90C05, 90C22, 90C25, 90C30, 90C51]
(*see:* **Interior point methods for semidefinite programming**)
barrier function
[90C25, 90C30]
(*see:* **Successive quadratic programming: solution by active sets and interior point methods**)
barrier function *see:* logarithmic —
barrier location problems
[90B85]
(*see:* **Multifacility and restricted location problems**)
barrier location problems
[90B85]
(*see:* **Multifacility and restricted location problems**)
barrier method *see:* interior point logarithmic —
barrier methods
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
barrier-penalty function *see:* logarithmic-quadratic —
barrier policy
[90B05, 90B06]
(*see:* **Global supply chain models**)
barycenter *see:* weighted —
barycenters *see:* generalized —
barycentric approximation
[90C15]
(*see:* **Multistage stochastic programming: barycentric approximation**)
barycentric approximation *see:* Multistage stochastic programming: —
barycentric scenario trees
[90C15]
(*see:* **Multistage stochastic programming: barycentric approximation**)
barycentric weights
[90C15]
(*see:* **Multistage stochastic programming: barycentric approximation**)
base polytope *see:* matroid —
based *see:* index- —; method- —; metric- —; model- —
based aggregation *see:* feature- —
based algorithm *see:* exact penalty function —
based approach *see:* gradient —
based approaches *see:* equation —; logic- —
based approximation *see:* point- —
based branch and bound *see:* IP/NLP —; QP/NLP —
based clustering approach: global optimum search with enhanced positioning *see:* Gene clustering: A novel decomposition- —
based complexity *see:* information- —
based complexity and information-based optimization *see:* Information- —
based control for drug delivery systems *see:* Model —
based controllers via parametric programming *see:* Design of robust model- —
based experimental analysis *see:* model- —
based framework *see:* graph —
based frameworkfor radiation therapy *see:* Optimization —
based gradient *see:* adjoint- —; sensitivity- —
based heuristics *see:* continuous —
based implementation *see:* PVM- —
based implementations *see:* MPI- —
based logic system of approximate reasoning *see:* point- —
based lower bounds *see:* eigenvalue —
based method *see:* KKT- —; penalty- —; reduction —
based methods *see:* descent- —; Disease diagnosis: optimization- —; MINLP: logic- —
based model *see:* gIS design pattern —; information- —
based NP methods *see:* knowledge- —
based optimization *see:* information- —; Information-based complexity and information- —; simulation- —
based outer approximation *see:* Logic- —
based perspective *see:* metric- —; model- —
based procedures *see:* gradient —
based on semidefinite relaxations *see:* bounds —

based on single-crystal X-ray diffraction data *see:* Optimization techniques for phase retrieval —
based on statistical models *see:* Global optimization —
based system *see:* rule- —
based theorem prover *see:* resolution —
based visualization *see:* Optimization- —
based yield *see:* Ω- —
bases *see:* neighboring —
bases of a matroid *see:* set of —
bases of an oriented matroid
[90C09, 90C10]
(*see:* **Oriented matroids**)
bases for polynomial equations *see:* Gröbner —
basic ABS class
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**)
basic alternative theorem
[90C26]
(*see:* **Invexity and its applications**)
basic alternative theorem
[90C26]
(*see:* **Invexity and its applications**)
basic column
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
basic component
[90C30]
(*see:* **Convex-simplex algorithm**)
basic component
[90C30]
(*see:* **Convex-simplex algorithm**)
basic constraint qualification
[46A20, 49K27, 49K40, 52A01, 90C30, 90C31]
(*see:* **Composite nonsmooth optimization**; **First order constraint qualifications**)
basic feasible solution
[90C05, 90C35, 90C60]
(*see:* **Complexity of degeneracy**; **Generalized networks**; **Linear programming**; **Linear programming: Klee–Minty examples**)
basic feasible solution
[90C60]
(*see:* **Complexity of degeneracy**)
basic features, examples from financial decision making *see:* Preference disaggregation approach: —
basic GRASP
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
basic matrix
[90C05, 90C33]
(*see:* **Linear programming: Klee–Minty examples**; **Pivoting algorithms for linear programming generating two paths**)
basic operations in a program
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)
basic outline of filled function methods
[65K05, 90C26, 90C30, 90C59]
(*see:* **Global optimization: filled function methods**)
basic QC
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
basic rules of branch and bound
[90C10, 90C29]
(*see:* **Multi-objective integer linear programming**)
basic sensitivity theorem
[90C31]
(*see:* **Bounds and solution vector estimates for parametric NLPS**)
basic software routines *see:* package of —
basic solution
[05B35, 65K05, 90C05, 90C20, 90C33, 90C35]
(*see:* **Criss-cross pivoting rules**; **Generalized networks**; **Linear programming**; **Linear programming: Klee–Minty examples**; **Pivoting algorithms for linear programming generating two paths**)
basic solution
[90C05]
(*see:* **Linear programming**)
basic solution *see:* degenerate —
basic solutions
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**; **Lexicographic pivoting rules**)
basic variables
[49M37, 90C05, 90C11]
(*see:* **Linear programming**; **Mixed integer nonlinear programming**)
basic VNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
BASICS *see:* Lagrangian duality: —
basin *see:* g- —; local —
basins of attraction
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
basis
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
basis
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
basis *see:* advanced —; complementary —; dual degenerate —; extremal —; graver —; Gröbner —; Hilbert —; optimal —; primal degenerate —; primal feasible —; reduced Gröbner —; tangle —; universal Gröbner —; working —
basis forest
[90C35]
(*see:* **Generalized networks**)
basis method *see:* extremal —
basis orientation of an oriented matroid
[90C09, 90C10]
(*see:* **Oriented matroids**)
basis tableau *see:* lexico-positive —
Bassett–Maybee–Quirk theorem
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
batch design under uncertainty *see:* Global optimization in —

batch learning
(*see:* **Bayesian networks**)
batch mode
(*see:* **Planning in the process industry**)
batch plant *see:* multiproduct —
batch plant design
[90C05, 90C26]
(*see:* **Continuous global optimization: applications**; **Global optimization in batch design under uncertainty**)
batch process
[90C26]
(*see:* **Global optimization in batch design under uncertainty**)
batch process
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
batch processes *see:* Medium-term scheduling of —; MINLP: design and scheduling of —; Reactive scheduling of —
batch processes with resources *see:* Short-term scheduling of —
batch production systems
(*see:* **Planning in the process industry**)
batch reactor *see:* fed- —
batch scheduling
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
Bauer formula
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
Bayes *see:* naïve —; simple —
Bayes optimal rule
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
Bayesian approach
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
Bayesian approach
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
Bayesian decision-theoretic framework
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**)
Bayesian global optimization
(90C26, 90C10, 90C15, 65K05, 62C10)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian networks**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
Bayesian heuristic approach
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
Bayesian inference
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
Bayesian methods
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
Bayesian networks
(*refers to:* **Bayesian global optimization**; **Evolutionary algorithms in combinatorial optimization**; **Neural networks for combinatorial optimization**)
Bayesian networks *see:* chain rule for —; dynamical —
Bayesian parameter estimation
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
Bayesian stopping rule
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
Bayesian stopping rule
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**)
bB procedure
[90C26]
(*see:* **D.C. programming**)
BB search engine
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms**)
BCI
[93-XX]
(*see:* **Boundary condition iteration BCI**)
BCI *see:* Boundary condition iteration —
BDMST
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
be stable without pivoting *see:* guaranteed to —
beacon
(*see:* **Semidefinite programming and the sensor network localization problem, SNLP**)
beam angle optimization
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
beam angle selection
[90C11, 90C59]
(*see:* **Nested partitions optimization**)
beam angle selection and wedge orientation optimization
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
beam cross-sectional shapes
[90C26, 90C90]
(*see:* **Structural optimization: history**)
beam's-eye-view
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)

beam segmentation problem
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
Beam selection in radiotherapy treatment design
(*refers to:* **Credit rating and optimization methods**; **Evolutionary algorithms in combinatorial optimization**; **Optimization based frameworkfor radiation therapy**)
beam weight optimization
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
beams
[90C26, 90C90]
(*see:* **Structural optimization: history**)
Beckmann QAP *see:* Koopmans– —
Beckmann quadratic assignment problem *see:* Koopmans– —
before-departure *see:* duty- —
behavior *see:* asymptotic —; average —; average case —; day-to-day dynamic travel —; exponential —; noncooperative —; random —
behavior of CAP on trees *see:* asymptotic —
behavior of a generally nonhomogeneous and nonisotropic body *see:* linear thermoelastic —
beliefs *see:* homogeneous —
Bellman's equation
[49L20, 49L99, 90C39, 90C40]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Neuro-dynamic programming**)
Bellman equation *see:* derivation of the Hamilton–Jacobi– —; Hamilton–Jacobi– —; solution of the Hamilton–Jacobi– —; sufficiency theorem for the Hamilton–Jacobi– —
Bellman–Ford method
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
ben-Tal SOCQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
Benders decomposition
[90-02, 90C10, 90C15, 90C27, 90C90]
(*see:* **Chemical process planning**; **L-shaped method for two-stage stochastic programs with recourse**; **Operations research models for supply chain management and design**; **Stochastic integer programs**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic programs with recourse: upper bounds**; **Time-dependent traveling salesman problem**)
Benders decomposition
[90C26, 90C27]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Time-dependent traveling salesman problem**)
Benders decomposition *see:* generalized —; nested —
Benders decomposition approach
[90C30, 90C35]
(*see:* **Optimization in water resources**)
Benders method *see:* generalized —
best algorithm *see:* Bialas–Karwan Kth- —
best approximation
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
best approximation *see:* Chebyshev —; maximal —; minimal —; polynomial of —
best approximation by bounded or continuous functions *see:* Lipschitzian operators in —
best approximation operator
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
Best approximation in ordered normed linear spaces
(90C46, 46B40, 41A50, 41A65)
best arc construction procedure
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
best bound
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
best bound rule
[90C11]
(*see:* **MINLP: branch and bound methods**)
best-compromise solution
[90C11, 90C29, 90C90]
(*see:* **Multi-objective optimization: interaction of design and control**)
best estimate
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
best estimate using pseudocosts
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
best estimate using pseudoshadow prices
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
best-first
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)
best-first tree search
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
Best-First Tree Search *see:* Parallel —
best fit
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
best fitting to data
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
best improvement
[68Q25, 68R10, 68W40, 9008, 90C26, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**; **Variable neighborhood search methods**)
Best improvement
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
best node construction procedure
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)

best projection
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
best response mapping
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
best start
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: two-phase methods**)
best value
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
beta algorithm *see:* alpha- —; high failure of the alpha- —; low failure of the alpha- —
β-sheet
[92C40]
(*see:* **Monte-Carlo simulated annealing in protein folding**)
better approximation
[90C27]
(*see:* **Steiner tree problems**)
between nonlinear complementarity problem and fixed point problem *see:* Equivalence —
between primal and dual solutions *see:* exploiting the interplay —
BF *see:* level —; model —
BFGS *see:* L- —; L-RH- —
BFGS algorithm *see:* limited-memory reduced-Hessian —; successive affine reduction —
BFGS-CG relationship
[90C30]
(*see:* **Conjugate-gradient methods**)
BFGS method
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
BFGS method *see:* limited-memory —; memoryless —
BFGS quasi-Newton update
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
BFGS update
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
BFGS update
[90C30]
(*see:* **Broyden family of methods and the BFGS update**)
BFGS update *see:* Broyden family of methods and the —
BFM
[90C30]
(*see:* **Broyden family of methods and the BFGS update**)
BFR *see:* level —; model —
BFS
[90C60]
(*see:* **Complexity of degeneracy**)
BFS *see:* degenerate —; nearly degenerate —; nondegenerate —
bi-knapsack problem
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)

Bi-objective assignment problem
(90C35, 90C10)
(*referred to in:* **Assignment and matching**; **Assignment methods in clustering**; **Communication network assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Frequency assignment problem**; **Fuzzy multi-objective linear programming**; **Linear ordering problem**; **Maximum partition matching**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**; **Quadratic assignment problem**)
(*refers to:* **Assignment and matching**; **Assignment methods in clustering**; **Communication network assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Frequency assignment problem**; **Fuzzy multi-objective linear programming**; **Maximum partition matching**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**; **Quadratic assignment problem**)

bI-VNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)

Bialas–Karwan Kth-best algorithm
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)

bias function
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)

bibliography of stochastic programming
[90C11, 90C15]
(*see:* **Stochastic programming with simple integer recourse**)

bidding increment
[90C30, 90C35]
(*see:* **Auction algorithms**)
bidding system *see:* preferential —
bidimensional knapsack problem
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
bidual problem
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
biduality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
biduality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
biduality in nonconvex optimization *see:* Duality theory: —
biduality theorem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
bifunction *see:* monotone —; pseudomonotone —; quasimonotone —
bifunction (with respect to another) *see:* pseudomonotone —
big-M
[90C09, 90C10, 90C11]
(*see:* **MINLP: logic-based methods**)
bigraph
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
bigraph *see:* signed —
bilateral boundary value problem
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
Bilevel fractional programming
(90C32, 90C26)
(*referred to in:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Fractional combinatorial optimization**; **Fractional programming**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Quadratic fractional programming: Dinkelbach method**; **Stochastic bilevel programs**)
Bilevel linear programming
(49-01, 49K10, 49M37, 90-01, 91B52, 90C05, 90C27)
(*referred to in:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
bilevel linear programming
[49-01, 49K10, 49K45, 49M37, 49N10, 90-01, 90C05, 90C20, 90C27, 91B52]
(*see:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**)
Bilevel linear programming: complexity, equivalence to minmax, concave programs
(49-01, 49K45, 49N10, 90-01, 91B52, 90C20, 90C27)
(*referred to in:* **Bilevel linear programming**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Concave programming**; **Minimax: directional differentiability**; **Minimax theorems**; **Minimum concave transportation problems**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Nondifferentiable optimization: minimax problems**; **Stochastic bilevel programs**; **Stochastic programming: minimax approach**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Concave programming**; **Minimax: directional differentiability**; **Minimax theorems**; **Minimum concave transportation problems**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Nondifferentiable optimization: minimax problems**; **Stochastic bilevel programs**; **Stochastic programming: minimax approach**; **Stochastic quasigradient methods in minimax problems**)
bilevel optimization
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
bilevel optimization
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Mixed integer nonlinear bilevel programming: deterministic global optimization**)

Bilevel Optimization *see:* Mixed Integer —
Bilevel optimization: feasibility test and flexibility index (90C26)
(*referred to in:* **Adaptive convexification in semi-infinite optimization**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; Generalized semi-infinite programming: optimality conditions; **Minimax: directional differentiability**; **Minimax theorems**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Nondifferentiable optimization: minimax problems**; **Stochastic bilevel programs**; **Stochastic programming: minimax approach**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Minimax: directional differentiability**; **Minimax theorems**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Nondifferentiable optimization: minimax problems**; **Stochastic bilevel programs**; **Stochastic programming: minimax approach**; **Stochastic quasigradient methods in minimax problems**)
bilevel program
[49-01, 49K10, 49M37, 90-01, 90B30, 90B50, 90C05, 90C25, 90C26, 90C27, 90C29, 90C30, 90C31, 91A10, 91B32, 91B52, 91B74]
(*see:* **Bilevel linear programming**; **Bilevel programming**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**)
bilevel program *see:* stochastic —
Bilevel programming (49M37, 90C26, 91A10)
(*referred to in:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
bilevel programming
[90C26, 90C30, 90C31]
(*see:* **Bilevel programming: introduction, history and overview**)
bilevel programming
[90-01, 90B30, 90B50, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C90, 91B32, 91B52, 91B74]
(*see:* **Bilevel programming: global optimization**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **MINLP: reactive distillation column synthesis**; **Mixed integer nonlinear bilevel programming: deterministic global optimization**; **Optimization with equilibrium constraints: A piecewise SQP approach**)
bilevel programming *see:* complexity of —; duality for —; enumeration in —; implicit function approach to —
Bilevel programming: applications (91B99, 90C90, 91A65)
(*referred to in:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
Bilevel programming: applications in engineering
(*referred to in:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Design optimization in computational fluid dynamics**; **Interval analysis: application to chemical engineering design problems**; **Multidisciplinary design optimization**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Optimal design of composite**

structures; **Optimal design in nonlinear optics**; **Stochastic bilevel programs**; **Structural optimization: history**)
bilevel programming: deterministic global optimization *see:* Mixed integer nonlinear —
Bilevel programming framework for enterprise-wide process networks under uncertainty
Bilevel programming: global optimization
(90C90, 90C30)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
Bilevel programming: implicit function approach
(90C26, 90C31, 91A65)
(*referred to in:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
Bilevel programming: introduction, history and overview
(90C26, 90C30, 90C31)
(*referred to in:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Nondifferentiable optimization: parametric programming**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Stochastic bilevel programs**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
Bilevel programming in management
(90-01, 91B52, 91B74, 91B32, 90B30, 90B50)
(*referred to in:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
Bilevel programming: optimality conditions and duality
(90C25, 90C29, 90C30, 90C31)
(*referred to in:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Stochastic bilevel programs**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**; **Nondifferentiable optimization: parametric programming**; **Stochastic bilevel programs**)
bilevel programming problem
[90C25, 90C26, 90C29, 90C30, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**; **Bilevel programming: optimality conditions and duality**)
bilevel programming problem
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)

bilevel programming problem *see:* generalized —; linear —
bilevel programming problems
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)
bilevel programming problems *see:* solution of —
bilevel programs
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
bilevel programs *see:* algorithms for stochastic —; Stochastic —
bilinear
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
bilinear
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
bilinear form *see:* K-local —
bilinear forms
[49-XX, 65M60, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**; **Variational inequalities: F. E. approach**)
bilinear function
[90C26]
(*see:* **Convex envelopes in optimization problems**)
bilinear matrix inequality
[93D09]
(*see:* **Robust control**)
Bilinear programming
bilinear programming
[49M37, 90C09, 90C10, 90C11, 90C25, 90C26, 91A10]
(*see:* **Bilevel programming**; **Concave programming**; **Disjunctive programming**; **MINLP: branch and bound methods**)
bilinear programming
[49-01, 49K45, 49N10, 90-01, 90C20, 90C27, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**)
bilinear programming *see:* difficulties in —; optimality in —; stable —
Bilinear programming: applications in the supply chain management
bilinear programming problem
[49-01, 49K45, 49N10, 90-01, 90C20, 90C27, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**)
bilinear symmetric continuous form *see:* coercive —
bilinear terms
[90C26, 90C90]
(*see:* **Global optimization of heat exchanger networks**)
bimatrix games
[90C11, 90C33]
(*see:* **LCP: Pardalos–Rosen mixed integer formulation**)
bimatrix games
[90C11, 90C33]
(*see:* **LCP: Pardalos–Rosen mixed integer formulation**; **Linear complementarity problem**)
bin packing problem
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
binary
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
binary constraint satisfaction problem
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
binary CSPs
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
binary encoding
[92B05]
(*see:* **Genetic algorithms**)
binary encoding
[92B05]
(*see:* **Genetic algorithms**)
binary heap
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
binary length
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
binary linear programming
[90C30]
(*see:* **Lagrangian duality: BASICS**)
binary matroid
[90C09, 90C10]
(*see:* **Matroids**)
binary noninterference constraints
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
binary operations on relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
binary programming *see:* positive semi-definite quadratic —
binary relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
binary search
[90C09]
(*see:* **Inference of monotone boolean functions**)
binary search algorithm
[90C09]
(*see:* **Inference of monotone boolean functions**)
binary search-Hansel chains question-asking strategy
[90C09]
(*see:* **Inference of monotone boolean functions**)
binary search-Hansel chains question-asking strategy
[90C09]
(*see:* **Inference of monotone boolean functions**)
binary search method
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
binary star
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)

binary star *see:* spectroscopic visual —; visual —
binary star astronomy *see:* Global optimization in —
binary surrogates
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
binary tree
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)
binary trees
[05C85]
(*see:* **Directed tree networks**)
binary variables
[90C11]
(*see:* **MINLP: branch and bound methods**)
binary vectors *see:* ordering on —
binomial *see:* degree of a —
binomial distribution
[90C15]
(*see:* **Logconcavity of discrete distributions**)
binomial moments
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
biochemical processes
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
biological constraints *see:* incorporation of —
biology *see:* computational —
biomolecular structures *see:* Steiner ratio of —
biorthogonal polynomial
[33C45, 65F20, 65F22, 65K10]
(*see:* **Least squares orthogonal polynomials**)
bipartite
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
bipartite chordal graph
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
bipartite graph
[05C85, 90C09, 90C10]
(*see:* **Combinatorial matrix analysis**; **Directed tree networks**)
bipartite graph *see:* convex —
bipartite matching
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
bipartite matching problem *see:* weighted —
bipartite network
[90B06, 90B10, 90C26, 90C35]
(*see:* **Minimum concave transportation problems**)
bipartite subgraph *see:* maximum —
bipartite tournament
[90C35]
(*see:* **Feedback set problems**)
bipartitioning problem *see:* graph —
bipartization *see:* graph —
bipartization problem *see:* graph —; minimum weighted graph —
BiQAP
[90C08, 90C11, 90C27]
(*see:* **Biquadratic assignment problem**)
Biquadratic assignment problem
(90C27, 90C11, 90C08)
(*refers to:* **Feedback set problems**; **Generalized assignment problem**; **Graph coloring**; **Graph planarization**; **Greedy randomized adaptive search procedures**; **Integer programming: branch and bound methods**; **Quadratic assignment problem**)
biquadratic assignment problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Biquadratic assignment problem**; **Quadratic assignment problem**)
birkhoff's theorem
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
bisection
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
bisection
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
bisection *see:* max- —; multidimensional —
bisection algorithm *see:* generalized —
Bisection global optimization methods
(90C30, 65K05)
(*referred to in:* **αBB algorithm**; **Global optimization: interval analysis and balanced interval arithmetic**)
(*refers to:* **αBB algorithm**)
bisection method
[65K05, 90C20, 90C25, 90C30]
(*see:* **Bisection global optimization methods**; **Quadratic programming over an ellipsoid**)
bisection method
[90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**)
Bisection Problem *see:* minimum —
bisection search
[90C30]
(*see:* **Convex-simplex algorithm**)
bisection search
[90C30]
(*see:* **Frank–Wolfe algorithm**)
bisectored
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
bisectored unit disk graphs
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
bisubmodular system
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
bisymmetric matrix
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
bisymmetric positive semidefinite matrix
[65K05, 90C20, 90C33]

(*see:* **Principal pivoting methods for linear complementarity problems**)
BK-product of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
bK-products
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
BK-products
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
BL-logic
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
black-box
[65K99]
(*see:* **Global optimization of planar multilayered dielectric structures**; **Maximum cut problem, MAX-CUT**)
black-box global optimization
[65K05]
(*see:* **Direct global optimization algorithm**)
black-box optimization
[90C05]
(*see:* **Global optimization in the analysis and management of environmental systems**)
black-box optimization
[65K05]
(*see:* **Direct global optimization algorithm**)
black-box strategy
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
black oil model
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
Black–Scholes model
[91B50]
(*see:* **Financial equilibrium**)
blackball number
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
Bland *see:* rule of —
Bland least index pivoting rule
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
Bland rule
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
Bland technique
[90C60]
(*see:* **Complexity of degeneracy**)
blended panel
[90C26, 90C29]
(*see:* **Optimal design of composite structures**)
blending
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
blending *see:* nonlinear —
blending and distribution scheduling: an MILP model *see:* Gasoline —
blending index
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
blending and pooling problems *see:* MINLP: applications in —
blending problems *see:* pooling and —
bliss
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
block
[90B35]
(*see:* **Job-shop scheduling problem**)
block *see:* solution —; vehicle —
block-0-diagonal operator
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
block angular form
[65Fxx]
(*see:* **Least squares problems**)
block-angular structure
[90C06]
(*see:* **Decomposition principle of linear programming**)
block-angular structure *see:* dual —
block-clique graph
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
block of a partition
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
block pivot
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
block plan
[90B80]
(*see:* **Facilities layout problems**)
block theorem
[90B35]
(*see:* **Job-shop scheduling problem**)
block truncated Newton software package
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
blocks for the process units *see:* building —
blocks of variables *see:* eliminating —
blossom
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
blossom inequalities
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
BLPP
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)
BLPP *see:* complexity of the linear —

BMI
[93D09]
(*see:* **Robust control**)
BNR-Prolog
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
board matrix *see:* chess- —
body *see:* linear thermoelastic behavior of a generally nonhomogeneous and nonisotropic —
body-tail problem *see:* head- —
bold arcs
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
bold connective
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
bold-driver method
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
bold strategy
[49L20, 90C40]
(*see:* **Dynamic programming: undiscounted problems**)
boldface
[90B15]
(*see:* **Evacuation networks**)
Boltzmann constant
[90C27, 90C90]
(*see:* **Simulated annealing**)
boltzmann density
(*see:* **Laplace method and applications to optimization problems**)
Boltzmann distribution
[65K05, 90C30]
(*see:* **Random search methods**)
bond angle
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
bond angle
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
bond distance
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
bond distance
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
bonds with constant maturities *see:* estimating the spot rate for —
Bonferroni bounds *see:* Boole– —
Boole–Bonferroni bounds
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
Boole–Bonferroni bounds
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
Boolean 2-valued function
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
Boolean 2-valued logic algebra
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
Boolean algebra
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
Boolean arrangement of hyperplanes
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
Boolean circuit *see:* depth of a —
Boolean classification problem
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
Boolean classification problem
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
boolean classification problems *see:* Optimization in —
Boolean connective
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
Boolean formula *see:* satisfiable —
Boolean formula in conjunctive normal form
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
Boolean formulas *see:* satisfiability of —
Boolean function
[90C09]
(*see:* **Inference of monotone boolean functions**)
Boolean function
[90C09]
(*see:* **Inference of monotone boolean functions**)
Boolean function *see:* antitone —; antitone monotone —; isotone —; isotone monotone —; monotone —; nondecreasing monotone —; nonincreasing monotone —
Boolean function inference *see:* monotone —
Boolean function inference problem
[90C09]
(*see:* **Inference of monotone boolean functions**)
Boolean function inference problem
[90C09]
(*see:* **Inference of monotone boolean functions**)
boolean functions *see:* Inference of monotone —; interactive learning of —
Boolean and fuzzy relations
(03E72, 03B52, 47S40, 68T27, 68T35, 68Uxx, 91B06, 90Bxx, 91Axx, 92C60)
(*referred to in:* **Alternative set theory**; **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
(*refers to:* **Alternative set theory**; **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
Boolean satisfiability
[90C60]
(*see:* **Complexity theory**)

Boolean variables
(*see:* **Logic-based outer approximation**)
bootstrapping
[90C35]
(*see:* **Feedback set problems**)
border node
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
border nodes set
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
bordering method
[33C45, 65F20, 65F22, 65K10]
(*see:* **Least squares orthogonal polynomials**)
bore model *see:* well —
Borel set
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
BOT types of logical connectives *see:* TOP and —
both water environment *see:* minimizing the degradation in quality of —
both-way compatibility
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
bottleneck machine
[90B35]
(*see:* **Job-shop scheduling problem**)
bottleneck measure
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
bottleneck quadratic assignment problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
bottleneck Steiner ratio
[05C05, 05C85, 68Q25, 90B80]
(*see:* **Bottleneck steiner tree problems**)
Bottleneck steiner tree problems
(05C05, 05C85, 68Q25, 90B80)
(*referred to in:* **Capacitated minimum spanning trees**; **Minimax game tree searching**; **Shortest path tree algorithms**; **Steiner tree problems**)
(*refers to:* **Capacitated minimum spanning trees**; **Directed tree networks**; **Minimax game tree searching**; **Shortest path tree algorithms**; **Steiner tree problems**)
bottleneck Steiner trees
[05C05, 05C85, 68Q25, 90B80]
(*see:* **Bottleneck steiner tree problems**)
bottleneck Steiner trees
[05C05, 05C85, 68Q25, 90B80]
(*see:* **Bottleneck steiner tree problems**)
bottlenecks in NLP solvers
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
Bouligand cone
[65K05, 90C30, 90Cxx]
(*see:* **Nondifferentiable optimization: minimax problems**; **Quasidifferentiable optimization: optimality conditions**)
Bouligand cone
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
bouligand tangent
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
Bouligand tangent cone
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
bound *see:* basic rules of branch and —; best —; Branch and —; branch-and- —; Edmundson–Madansky upper —; error —; Gilmore–Lawler lower —; global error —; guaranteed —; guaranteed lower —; Hunter–Worsley upper —; Jensen lower —; Lehmann–Goerisch —; lower —; lP/NLP based branch and —; optimal componentwise —; parametric lower —; parametric upper —; piecewise linear upper —; polynomial upper —; QP/NLP based branch and —; Rayleigh–Ritz —; reformulation/spatial branch and —; restricted-recourse —; stochastic branch and —; upper —; valid lower —; valid upper —
bound algorithm *see:* branch and —
bound algorithm for weighted graph planarization *see:* branch and —
bound algorithms *see:* branch and —
bound for approximate solutions of nonlinear systems of equations *see:* error —
bound consistency
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
bound constrained quadratic problem
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
bound constraints
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: unconstrained and constrained optimization**)
bound constraints *see:* flow —; Quadratic programming with —
bound dichotomy *see:* generalized-upper- —
bound enumerative techniques *see:* branch and —
bound-factors
[90C26]
(*see:* **Reformulation-linearization technique for global optimization**)
bound function *see:* lower —
bound on gas lift availability *see:* upper —
bound global optimization algorithm *see:* MINLP: branch and —
bound-improvement *see:* dual —; primal —
bound method *see:* arc oriented branch and —; branch and —; node oriented branch and —
bound methods *see:* branch and —; Integer programming: branch and —; Interval analysis: subdivision directions in interval branch and —; MINLP: branch and —
bound to optimality *see:* guaranteed —
bound and outer approximation *see:* hybrid branch and —
bound principle *see:* branch and —
bound rule *see:* best —
bound scheme *see:* branch and —
bound for a set *see:* lower —; upper —
bound for solutions of nonlinear systems of equations *see:* rigorous —
bound strategy *see:* branch and —
bound techniques *see:* branch and —

bound test *see:* lower —; upper- —
bound for unconstrained optimization *see:* branch and —
boundary *see:* lower —; upper —
Boundary condition iteration BCI
(93-XX)
(*referred to in:* **Control vector iteration CVI**)
(*refers to:* **Control vector iteration CVI**)
boundary condition iteration method
[93-XX]
(*see:* **Boundary condition iteration BCI**)
boundary conditions
[03H10, 49J27, 49K05, 49K10, 49K15, 49K20, 90C34]
(*see:* **Duality in optimal control with first order differential equations**; **Semi-infinite programming and control problems**)
boundary conditions *see:* elastostatics with nonlinear —; quasidifferential elastic —; quasidifferential thermal —; variational formulation of quasidifferential thermal —
boundary dependence property
[90C33]
(*see:* **Topological methods in complementarity theory**)
boundary effects
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
boundary flux estimation in distributed systems
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
boundary of a function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
boundary laws and variational equalities *see:* single-valued —
boundary point *see:* lower —; upper —
boundary triangulation
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
boundary value conditions
[34A55, 78A60, 90C30]
(*see:* **Optimal design in nonlinear optics**)
boundary value problem *see:* bilateral —; ODE two-point —; two-point —; unilateral —
boundary variation technique
[49J20, 49J52]
(*see:* **Shape optimization**)
bounded
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
bounded
[90C30]
(*see:* **Frank–Wolfe algorithm**)
bounded or continuous functions *see:* Lipschitzian operators in best approximation by —
bounded cost per stage *see:* discounted problem with —
bounded degree minimum spanning tree problem
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
bounded integer variable *see:* multiple branches for —
bounded level set
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
bounded polynomial time algorithm *see:* efficient polynomially —
bounded ratio disk graphs
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
bounded rationality
[90-01, 90B30, 90B50, 91B32, 91B52, 91B74]
(*see:* **Bilevel programming in management**)
bounded Turing machine *see:* exponentially space- —; exponentially time- —; polynomially space- —; polynomially time- —
bounding *see:* lower —; upper —
Bounding derivative ranges
(90C30, 90C26)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
(*refers to:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
bounding the expectation
[90C15]
(*see:* **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
bounding Hessian *see:* lower —
bounding step
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)
bounding structure *see:* generalized upper —
bounds
[62C20, 90B80, 90C15, 90C31]
(*see:* **Facilities layout problems**; **Sensitivity and stability in NLP: approximation**; **Stochastic programming: minimax approach**)

bounds *see:* arc flow —; Boole–Bonferroni —; computable optimal value —; constructive lower —; eigenvalue based lower —; Gilmore–Lawler type lower —; Hunter–Worsley —; Lagrangian —; lower —; lower weight —; Maximum constraint satisfaction: relaxations and upper —; maximum flow problem with nonnegative lower —; parametric —; parametric upper and lower —; Stochastic programs with recourse: upper —; upper —; variance reduction lower —
bounds based on semidefinite relaxations
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
bounds constraints
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
bounds constraints *see:* generalized upper —; lower and upper —
bounds on the distance of a feasible point to a solution point
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
bounds to eigenvalues *see:* upper and lower —
bounds of interval matrices *see:* Interval analysis: eigenvalue —
bounds for linear equations
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
bounds for multivariate probability integrals *see:* lower —; upper —
bounds for NLP *see:* solution-point —
bounds on simplices
[90C15]
(*see:* **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
Bounds and solution vector estimates for parametric NLPS
(90C31)
(*referred to in:* **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**)
(*refers to:* **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**)
bounds subject to moment conditions *see:* optimal integral —
box
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
box *see:* black- —; feasible —; indeterminate —; process- —; reduced —
box(2)*-consistency*
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
box consistency
[65G20, 65G30, 65G40, 65K05, 68T20, 90C30]
(*see:* **Interval constraints**; **Interval global optimization**)
box constraints
[90C60]
(*see:* **Complexity theory: quadratic programming**)
box constraints
[90C60]
(*see:* **Complexity theory: quadratic programming**)
box global optimization *see:* black- —
box optimization *see:* black- —
box strategy *see:* black- —
boxes *see:* canonical —; leading —
BP
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
BPP
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)
BQAP
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
brachytherapy
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
bracket
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
bracket *see:* multidimensional —
bracketing
[90C30]
(*see:* **Nonlinear least squares problems**)
Braess paradox
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
Braess paradox
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
braid arrangement of hyperplanes
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
braid group
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
Brakhage preconditioner *see:* Atkinson– —
branch
[90C11]
(*see:* **MINLP: branch and bound methods**)
branch *see:* cut-and- —
branch & cut algorithms *see:* Stable set problem: —
branch-and-bound
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
branch-and-price
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
branch-and-price
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
branch and bound
[49M37, 65G30, 65G40, 65K05, 65K10, 68M20, 68Q25,

68Q99, 90B06, 90B10, 90B35, 90B50, 90B80, 90C05, 90C10, 90C11, 90C25, 90C26, 90C27, 90C29, 90C30, 90C35, 90C57, 90C90, 92C40]
(*see:* **Branch and price: Integer programming with column generation**; **Chemical process planning**; **Communication network assignment problem**; **Concave programming**; **Global optimization in batch design under uncertainty**; **Global optimization of heat exchanger networks**; **Global optimization: interval analysis and balanced interval arithmetic**; **Graph coloring**; **Integer programming: lagrangian relaxation**; **Inventory management in supply chains**; **Lagrangian duality: BASICS**; **MINLP: branch and bound methods**; **Mixed integer nonlinear programming**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**; **Set covering, packing and partitioning problems**; **Time-dependent traveling salesman problem**; **Vehicle scheduling**)

Branch and bound
[49M20, 49M37, 65K05, 65K10, 90B80, 90C05, 90C06, 90C08, 90C10, 90C11, 90C20, 90C26, 90C30, 90C31]
(*see:* **αBB algorithm**; **Facilities layout problems**; **Generalized outer approximation**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: lagrangian relaxation**; **Linear ordering problem**; **MINLP: branch and bound methods**; **MINLP: global optimization with αBB**; **Mixed integer nonlinear programming**; **Multiparametric mixed integer linear programming**; **Standard quadratic optimization problems: applications**; **Stochastic transportation and location problems**)

branch and bound *see:* basic rules of —; IP/NLP based —; QP/NLP based —; reformulation/spatial —; stochastic —

branch and bound algorithm
[65K05, 90C11, 90C26]
(*see:* **MINLP: global optimization with αBB**)

branch and bound algorithm for weighted graph planarization
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)

branch and bound algorithms
[65G20, 65G30, 65G40, 65H20, 65K99, 90B35]
(*see:* **Interval Newton methods**; **Job-shop scheduling problem**)

branch and bound algorithms
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)

branch and bound enumerative techniques
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)

branch and bound global optimization algorithm *see:* MINLP: —

branch and bound method
[90B10]
(*see:* **Piecewise linear network flow problems**)

branch and bound method *see:* arc oriented —; node oriented —

branch and bound methods
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)

branch and bound methods *see:* Integer programming: —; Interval analysis: subdivision directions in interval —; MINLP: —

branch and bound and outer approximation *see:* hybrid —

branch and bound principle
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)

branch and bound scheme
[90C26]
(*see:* **Convex envelopes in optimization problems**)

branch and bound strategy
[49-01, 49K10, 49M37, 90-01, 90C05, 90C27, 91B52]
(*see:* **Bilevel linear programming**)

branch and bound techniques
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)

branch and bound for unconstrained optimization
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: unconstrained and constrained optimization**)

branch and contract algorithm
[90C26, 90C90]
(*see:* **Global optimization of heat exchanger networks**)

branch and cut
[90C10, 90C11, 90C26, 90C27, 90C35, 90C57]
(*see:* **Cutting plane methods for global optimization**; **MINLP: branch and bound methods**; **Multicommodity flow problems**; **Optimization in leveled graphs**; **Set covering, packing and partitioning problems**)

branch and cut algorithm *see:* Jünger–Mutzel —

branch and cut algorithms *see:* Integer programming: —

branch and cut procedure
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)

branch decomposition
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)

branch decompositions *see:* Branchwidth and —

branch of a feasible set
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)

branch and Infer
(*see:* **Mixed integer programming/constraint programming hybrid methods**)

branch and price
[90C35]
(*see:* **Multicommodity flow problems**)

branch and price and cut
[90C35]
(*see:* **Multicommodity flow problems**)

Branch and price: Integer programming with column generation
(68Q99)
(*referred to in:* **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming:**

branch and bound methods; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Nonoriented multicommodity flow problems**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Decomposition techniques for MILP: lagrangian relaxation**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)

branch problem
[65K05]
(*see:* **Automatic differentiation: root problem and branch problem**)

branch problem
[65K05]
(*see:* **Automatic differentiation: root problem and branch problem**)

branch problem *see:* Automatic differentiation: root problem and —

branch and prune
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)

branch and reduce
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)

branches for bounded integer variable *see:* multiple —

branching
[49M37, 65K10, 90C05, 90C06, 90C08, 90C10, 90C11, 90C26, 90C27, 90C30, 90C57, 90C59]
(*see:* **αBB algorithm**; **Integer programming: branch and bound methods**; **Quadratic assignment problem**)

branching *see:* strong —

branching algorithm
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)

branching step
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)

branchpoint of a graph
[90C35]
(*see:* **Feedback set problems**)

branchwidth
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)

Branchwidth and branch decompositions
(90C27, 68R10)

breadth-first
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)

breaking rule *see:* tie —

breakpoint
[90C11, 90C31]
(*see:* **Parametric mixed integer nonlinear optimization**)

breast cancer diagnosis
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)

breast tumors
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)

Bregman parameter
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)

bridges
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)

bridging model
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)

brief review *see:* Generalized variational inequalities: A —

(Brier) scoring rule *see:* quadratic —

Broadcast scheduling problem
(*refers to:* **Frequency assignment problem**; **Genetic algorithms**; **Graph coloring**; **Greedy randomized adaptive search procedures**; **Multi-objective integer linear programming**; **Optimization problems in unit-disk graphs**; **Simulated annealing**)

Broeckx linearization *see:* Kaufman– —

brother waits *see:* younger —

Brouwer degree
[90C33]
(*see:* **Topological methods in complementarity theory**)

brouwer fixed point theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 65G20, 65G30, 65G40, 65H20, 91A05]
(*see:* **Interval fixed point theory**; **Minimax theorems**)

Brownian motion
[60G35, 65K05]
(*see:* **Differential equations and global optimization**)

Brownian motion *see:* N-dimensional —

Broyden class *see:* quasi-Newton method of —

Broyden family
[49M37, 90C30]
(*see:* **Nonlinear least squares: Newton-type methods**; **Rosen's method, global convergence, and Powell's conjecture**)

Broyden family of methods
[90C30]
(*see:* **Broyden family of methods and the BFGS update**)
Broyden family of methods and the BFGS update
(90C30)
(*referred to in:* **Conjugate-gradient methods**; **Large scale unconstrained optimization**; **Numerical methods for unary optimization**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**; **Unconstrained optimization in neural network training**)
(*refers to:* **Conjugate-gradient methods**; **Large scale unconstrained optimization**; **Numerical methods for unary optimization**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**; **Unconstrained optimization in neural network training**)
Broyden–Fletcher–Goldfarb–Shanno method
[90C30]
(*see:* **Successive quadratic programming**)
Broyden–Fletcher–Goldfarb–Shanno quasi-Newton update
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
Broyden–Fletcher–Goldfarb–Shanno update
[90C30]
(*see:* **Broyden family of methods and the BFGS update**)
Broyden method *see:* Powell-symmetric– —
Broyden methods
[90C30]
(*see:* **Broyden family of methods and the BFGS update**)
Broyden–Spedicato algorithms for linear equations and linear least squares *see:* Abaffi– —
Broyden theorem
[15A39, 90C05]
(*see:* **Farkas lemma**)
brute-force
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
BSM
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
BSP
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
BSP
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
bSP model
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
BSP model
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
BSTP
[05C05, 05C85, 68Q25, 90B80]
(*see:* **Bottleneck steiner tree problems**)
Buchberger algorithm
[12D10, 12Y05, 13Cxx, 13P10, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Gröbner bases for polynomial equations**; **Integer programming: algebraic methods**)
Buchberger algorithm *see:* truncated —
bucket
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
budget constraint
[78M50, 90B50, 91B28]
(*see:* Global optimization algorithms for financial planning problems)
budget of uncertainty
(*see:* **Price of robustness for linear optimization problems**)
building *see:* model —
building blocks for the process units
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
bulk synchronous parallel
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
bulk synchronous parallel computer
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
bulk synchronous parallel model
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
bulk synchronous parallel model
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
bullwhip effect
[90-02]
(*see:* **Operations research models for supply chain management and design**)
Bunch and Parlett factorization
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in distillation systems**)
bundle algorithm
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
bundle algorithms
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
bundle method *see:* proximal —; proximal point —; variable metric —
bundle methods
[46N10, 49J40, 49J52, 65K05, 90-00, 90C15, 90C26, 90C30, 90C33, 90C47]
(*see:* **Nondifferentiable optimization**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Stochastic bilevel programs**)
bundle methods
[49J40, 49J52, 65K05, 90C30]
(*see:* **Nondifferentiable optimization: relaxation methods**; **Solving hemivariational inequalities by nonsmooth optimization methods**)
bundle-Newton method
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
bundle trust region
[49J40, 49J52, 65K05, 90C30]

(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
Burke local dualization *see:* Ioffe– —
Burke–Poliquin reduction
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
Burmeister function *see:* Fischer– —
business failure risk
[91B06, 91B60]
(*see:* **Financial applications of multicriteria analysis**)
busting *see:* consist- —
butterfly
[90C35]
(*see:* **Feedback set problems**)
butterfly *see:* toroidal —

C

C *see:* ADOL- —; algorithm polynomial of degree —
C-differentiable function
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
C-differential
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
(C_m^J)*-efficient point*
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
(C_m^J)*-efficient solution*
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
$C^{1,1}$ *optimization problem*
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
C^k*-Riemannian metric*
[58E05, 90C30]
(*see:* **Topology of global optimization**)
C-subdifferential
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
CA algorithm *see:* asynchronous parallel —; sequential —; synchronized parallel —
CA algorithms
[90C30]
(*see:* **Cost approximation algorithms**)
CA algorithms *see:* decomposition —
cable *see:* slack —
cable structures *see:* structural analysis of —
cables
[51K05, 52C25, 68Q25, 68U05, 90C22, 90C35]
(*see:* Graph realization via semidefinite programming)
calculation of the Hessian *see:* Automatic differentiation: —
calculation of Newton steps *see:* Automatic differentiation: —
calculus *see:* infinitesimal —; quasidifferential —
calculus of quasidifferentials
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
calculus of quasidifferentials *see:* Quasidifferentiable optimization: —
calculus of variations
[01A99]
(*see:* **Carathéodory, Constantine**; **Lagrange, Joseph-Louis**)
calculus of variations
[01A99, 03H10, 49J27, 90C34]
(*see:* **Carathéodory, Constantine**; **Lagrange, Joseph-Louis**; **Semi-infinite programming and control problems**)
calculus of variations *see:* inverse problem of the —; Nonconvex-nonsmooth —
calibration *see:* model —
called
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)
calm problem
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
calmness
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
calmness condition
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
calmness condition *see:* partial —
campaign
(*see:* **Planning in the process industry**)
campaign *see:* mixed-product —; single-product —
campaigns *see:* long —
canceling algorithm *see:* cycle- —
cancer chemotherapy
[93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**)
cancer diagnosis *see:* breast —
candidate list
[68T20, 68T99, 90B10, 90C27, 90C59]
(*see:* **Metaheuristics**; **Shortest path tree algorithms**)
Candidate List *see:* restricted —
canonical boxes
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
canonical dual transformation
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
canonical dual transformation method
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
canonical form
[49-XX, 90-XX, 90C26, 93-XX]
(*see:* **D.C. programming**; **Duality theory: biduality in nonconvex optimization**)
canonical function associated with Λ
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
canonical function space
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
canonical function space *see:* extended —

canonical monotonic optimization problem
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
canonical normal form
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
canonical transformation
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
Cantor set theory
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
CAP
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
CAP on trees
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
CAP on trees *see:* asymptotic behavior of —; exact algorithm for solving —; heuristic approach to solving —
capacitated arc routing problem
[90B06]
(*see:* **Vehicle routing**)
capacitated lot-sizing problem
[90C90]
(*see:* **Chemical process planning**)
capacitated minimum spanning arborescence problem
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
capacitated minimum spanning tree problem
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
capacitated minimum spanning tree problem
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
Capacitated minimum spanning trees
(90C27, 68T99)
(*referred to in:* **Bottleneck steiner tree problems**; **Shortest path tree algorithms**)
(*refers to:* **Bottleneck steiner tree problems**; **Directed tree networks**; **Minimax game tree searching**; **Shortest path tree algorithms**)
capacitated network *see:* directed —
capacitated transportation problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
capacitated vehicle routing problem
[90B06]
(*see:* **Vehicle routing**)
capacity *see:* arc —; nodes with water storage —; problem with nonunit —; residual —; shannon zero-error —
capacity of an arc in a network
[90C35]
(*see:* **Minimum cost flow problem**)
capacity constraint
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
capacity constraint on arc flows
[90B10]
(*see:* **Piecewise linear network flow problems**)
capacity constraints
[90B80, 90B85, 90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**; **Warehouse location problem**)
capacity constraints *see:* maximum oil, gas and water —; single fixed cost with —; single fixed cost with no —
capacity of a cut
[90C35]
(*see:* **Maximum flow problem**)
capital asset pricing model
[91B50]
(*see:* **Financial equilibrium**)
capital investment *see:* venture —
capital market line
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
Carathéodory
[01A99]
(*see:* **Carathéodory, Constantine**)
Carathéodory, Constantine
(01A99)
(*referred to in:* **Carathéodory theorem**; **History of optimization**)
(*refers to:* **Carathéodory theorem**; **History of optimization**)
Carathéodory principle
[01A99]
(*see:* **Carathéodory, Constantine**)
Carathéodory theorem
(90C05)
(*referred to in:* **Carathéodory, Constantine**; **History of optimization**; **Krein–Milman theorem**; **Linear programming**; **Single facility location: circle covering problem**)
(*refers to:* **Carathéodory, Constantine**; **Krein–Milman theorem**; **Linear programming**)
Carathéodory theorem
[90B85, 90C06, 90C25, 90C27, 90C30, 90C35]
(*see:* **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Single facility location: circle covering problem**)
Carathéodory theorem
[90C06, 90C25, 90C30, 90C35]
(*see:* **Simplicial decomposition**; **Simplicial decomposition algorithms**)
cardinalities axiom *see:* two —
cardinality of a graph
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
cardinality matching problem *see:* maximum —
cardinality of a node
[90C35]
(*see:* **Generalized networks**)
cargo routing problems
(*see:* **Maritime inventory routing problems**)
Carl Friedrich *see:* Gauss —
Carlo *see:* Monte- —; pure Monte- —
Carlo configuration *see:* Monte- —
Carlo method *see:* metropolis Monte —; Monte- —; pure Monte- —
Carlo simulated annealing in protein folding *see:* Monte- —
Carlo simulation *see:* monte- —
Carlo simulation algorithm *see:* Monte- —

Carlo simulation procedure *see:* Monte- —
Carlo simulations for stochastic optimization *see:* Monte- —
CARP
[90B06]
(*see:* **Vehicle routing**)
carrying *see:* label —
Cartesian coordinates
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
Cartesian coordinates
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
Cartesian product
[90C30]
(*see:* **Cost approximation algorithms**)
Cartesian product set
[90C30]
(*see:* **Cost approximation algorithms**)
cascade *see:* temperature —
case *see:* convex-concave —; perfectly consistent —
case analysis *see:* average —; worst- —
case approach *see:* worst- —
case behavior *see:* average —
case complexity *see:* average —; worst- —
case complexity of algorithms *see:* average —
case of integral evaluation *see:* asymptotic —
case optimality *see:* worst- —
case performance guarantee *see:* worst- —
case setting *see:* average —
case of the trust region problem *see:* general —; hard —; Newton step —
case of a two-person game *see:* cooperative —
cash flow *see:* maximize operating —
catalysis: optimization methods *see:* Shape selective zeolite separation and —
catchment management
[90C30, 90C35]
(*see:* **Optimization in water resources**)
Cauchy approach *see:* modified —
Cauchy formula
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
Cauchy framework *see:* Newton– —; Unconstrained nonlinear optimization: Newton– —
Cauchy inequality
[15A39, 90C05]
(*see:* **Motzkin transposition theorem**)
Cauchy method
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
Cauchy method *see:* modified —
Cauchy point
[90C06]
(*see:* **Large scale unconstrained optimization**)
Cayley transform
[15A39, 90C05]
(*see:* **Farkas lemma**)
CCM
[90C30]
(*see:* **Cyclic coordinate method**)
cCOMB algorithm
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)
CD *see:* IS- —
c.d. function
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
CDPAP
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
cell
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements in optimization**)
cell of a function
[90Cxx]
(*see:* **Discontinuous optimization**)
cell of a polyhedral subdivision
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
cell sectorization
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
cell of a Turing machine *see:* tape —
center *see:* analytic —; uncapacitated —
center cutting plane method *see:* analytic —
center of gravity
[49M20, 90-08, 90C25]
(*see:* **Nondifferentiable optimization: cutting plane methods**)
center of gravity location
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
center of gravity method
[49M20, 90-08, 90C25]
(*see:* **Nondifferentiable optimization: cutting plane methods**)
center of an interval linear system
[15A99, 65G20, 65G30, 65G40, 90C26]
(*see:* **Interval linear systems**)
center node
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
center path
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
center problem *see:* p- —
center problem on a network *see:* p- —
centering *see:* design —
centering direction
[90C05]
(*see:* **Linear programming: interior point methods**)
centering hit and run *see:* artificial —
central arc
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)

central component
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
central path
[37A35, 49-XX, 90-XX, 90C05, 90C25, 90C30, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**; **Linear programming: interior point methods**; **Potential reduction methods for linear programming**; **Solving large scale and sparse semidefinite programs**)
central path
[90C25, 90C30]
(*see:* **Successive quadratic programming: solution by active sets and interior point methods**)
central trajectory
[90C05]
(*see:* **Linear programming: interior point methods**)
centroid of a simplex
[90C30]
(*see:* **Sequential simplex method**)
CEP
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
CEP *see:* RAF of —; restricted accessibility form of —; UAF of —; universally accessible form of —
cerevisiae *see:* saccharomyces —
certainty *see:* mathematical and computational —
certificate
[15A39, 90C05, 90C60]
(*see:* **Complexity theory**; **Linear optimization: theorems of the alternative**; **Motzkin transposition theorem**)
certificate
[15A39, 90C05]
(*see:* **Farkas lemma**; **Linear optimization: theorems of the alternative**; **Motzkin transposition theorem**)
CFAP
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
CFD
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
CFD *see:* design optimization in —
CG *see:* alternatives to —
CG family *see:* two-parameter —
CG method *see:* linear —; nonlinear —
CG-related algorithm
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
CG-related algorithms *see:* nonlinear —
CG relationship *see:* BFGS- —
CG-standard
[90C30]
(*see:* **Conjugate-gradient methods**)
CG-standard for minimizing q
[90C30]
(*see:* **Conjugate-gradient methods**)
CGM
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
CGM
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
cGS
[74A40, 90C26]
(*see:* **Shape selective zeolite separation and catalysis: optimization methods**)
CGU algorithm
[65K05, 90C26]
(*see:* **Molecular structure determination: convex global underestimation**)
chain *see:* ejection —; finite-state Markov —; global supply —; Hansel —; markov —; operational decisions in a supply —; stationary-state Markov —; strategic design of a supply —; supply —; two-stranded —
chain design *see:* supply —
chain justification
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
chain justification *see:* left- —; right- —
chain management *see:* Bilinear programming: applications in the supply —; Mathematical programming methods in supply —; operational supply —; strategic supply —; supply —
chain management and design *see:* Operations research models for supply —
chain methods *see:* ejection —
chain models *see:* Global supply —
chain networks
[05C85]
(*see:* **Directed tree networks**)
chain optimization *see:* supply —
chain performance measurement *see:* Supply —
chain rule
[90C30]
(*see:* **Generalized total least squares**)
chain rule for Bayesian networks
(*see:* **Bayesian networks**)
chain rules
[90C15]
(*see:* **Stochastic quasigradient methods: applications**)
chain sampling *see:* Markov —
chain simulation models *see:* supply —
chained local search
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
chains *see:* Inventory management in supply —; Markov —
chains question-asking strategy *see:* binary search-Hansel —; sequential Hansel —
Chaitin complexity *see:* Solomonoff–Kolmogorov– —
Chaitin in Omega
[90C60]
(*see:* **Kolmogorov complexity**)
challenge *see:* grand —
challenges in MINLP
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
challenges for OR
[90C27]
(*see:* **Operations research and financial markets**)

chamber
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
Chan method
[65Fxx]
(*see:* **Least squares problems**)
chance constraint programming
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
chance constraint programming
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
changes *see:* sensitivity analysis with respect to right-hand side —; up to first order —
changes in cost coefficients *see:* sensitivity analysis with respect to —
channel constrained frequency assignment *see:* adjacent —; co- —
chaotic iterative scheme
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
Characteristic *see:* spatial —
characteristic equation
[93D09]
(*see:* **Robust control**)
characteristic function
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
characteristic polynomial
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
characteristic vector
[05C60, 05C69, 05C85, 37B25, 68W01, 90C10, 90C20, 90C25, 90C27, 90C35, 90C59, 91A22]
(*see:* **Heuristics for maximum clique and independent set**; **L-convex functions and M-convex functions**; **Replicator dynamics in combinatorial optimization**)
characteristic vector *see:* weighted —
characteristics *see:* method of —
characterization of *see:* Convexifiable functions —
characterization of $E_{D,J}$
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
characterizing momments
[94A17]
(*see:* **Jaynes' maximum entropy principle**)
characterstic polynomial
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)
charactertstic vector
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
charge *see:* fixed —; linear fixed —
charge function *see:* fixed —
charge network flow problem *see:* fixed —
charge networks *see:* fixed —
charge problem *see:* fixed —
charge transportation problem *see:* fixed —
chart scores *see:* REL —
Chebyshev alternance
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
Chebyshev alternation
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
Chebyshev best approximation
[49K35, 49M27, 65K10, 90C25]
(*see:* **Convex max-functions**)
Chebyshev iterative method
[90C05, 90C25]
(*see:* **Metropolis, Nicholas Constantine**)
Chebyshev polynomial
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
Chebyshev problem
[65D10, 65K05]
(*see:* **Overdetermined systems of linear equations**)
Chebyshev set
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
checklist *see:* valuation of a —
checklist confirmation
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
checklist denial
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
checklist modus ponens
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
checklist modus tollens
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
checklist paradigm
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
checklist paradigm
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
Checklist paradigm semantics for fuzzy logics
(03B52, 03B50, 03C80, 62F30, 62Gxx, 68T27)
(*referred to in:* **Alternative set theory**; **Boolean and fuzzy relations**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
(*refers to:* **Alternative set theory**; **Boolean and fuzzy relations**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
checklist template
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
checkpointing
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
chemical engineering design problems *see:* Interval analysis: application to —

chemical equilibrium *see:* multiphase —; Optimality criteria for multiphase —
chemical equilibrium problem
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
chemical potential
[49K99, 65K05, 80A10, 90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**; **Optimality criteria for multiphase chemical equilibrium**)
Chemical process planning
(90C90)
(*referred to in:* **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)
(*refers to:* **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)
chemical reaction equilibrium *see:* Global optimization in phase and —
chemotherapy *see:* cancer —
Chen–Harker–Kanzow–Smale function
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
chess-board matrix
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Chevron method
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
chi-square statistic *see:* Pearson —
child of a vertex
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
Chinese postman problem
[90B20]
(*see:* **General routing problem**)
Chinese postman problem *see:* directed —
chirotope
[90C09, 90C10]
(*see:* **Oriented matroids**)
choice *see:* greedy —; rational —; rule of random —
choice adjustment process *see:* trip-route —
choice axiom
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
choice of the entering variable
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
choice knapsack *see:* multiple —
choice knapsack problem *see:* linear multiple- —; multidimensional multiple- —; multiple- —
choice of the leaving variable
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
choice property *see:* greedy- —
choices *see:* linguistic —
Choleski algorithm *see:* implicit —
Cholesky factorization
(15-XX, 65-XX, 90-XX)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Least squares problems**; **Linear programming**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
cholesky factorization
[15-XX, 65-XX, 65Fxx, 65K05, 90-XX, 90Cxx]
(*see:* **Cholesky factorization**; **Least squares problems**; **Symmetric systems of linear equations**)
Cholesky triangle
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
chordal
[90C35]
(*see:* **Feedback set problems**)
chordal graph
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
chordal graph *see:* bipartite —
chromatic number
[05-XX, 05C15, 05C17, 05C35, 05C62, 05C69, 05C85, 90C22, 90C27, 90C35, 90C59]
(*see:* **Frequency assignment problem**; **Lovász number**; **Optimization problems in unit-disk graphs**)

chromosome
[92B05]
(*see:* **Genetic algorithms**)
chromosome
[92B05]
(*see:* **Genetic algorithms**)
Chung–Gilbert conjecture
[90C27]
(*see:* **Steiner tree problems**)
Chvátal function
[90C10, 90C46]
(*see:* **Integer programming duality**)
Chvátal–Gomory cut
[90C05, 90C06, 90C08, 90C10, 90C11, 90C46]
(*see:* **Integer programming: branch and cut algorithms**; **Integer programming duality**)
Chvátal–Gomory cutting plane
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
Chvátal rank
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
CI
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
CID
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
circle *see:* largest empty —; least —; minimum —; smallest enclosing —
circle covering problem *see:* Single facility location: —
circle problem *see:* largest empty —; smallest enclosing- —
circle product of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
circles in a square *see:* equal —
circuit
[13Cxx, 13Pxx, 14Qxx, 90C09, 90C10, 90Cxx]
(*see:* **Integer programming: algebraic methods**; **Oriented matroids**)
circuit *see:* combinatorial switching —; depth of a Boolean —; HAMILTON —; Hamiltonian —; sign of a —
circuit design
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
circuit of a digraph
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
circuit orientation
[90C09, 90C10]
(*see:* **Oriented matroids**)
circuit problem *see:* Hamiltonian —
circuits
[90C09, 90C10, 90C35]
(*see:* **Matroids**; **Optimization in leveled graphs**)
circuits *see:* signed —
circulant matrix
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
circular path
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
circular unidimensional scale
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
Clarke *see:* generalized subdifferential of F.H. —
Clarke derivative
[26E25, 49J52, 52A27, 90C99]
(*see:* **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
Clarke derivative *see:* directional —
clarke derivatives *see:* Quasidifferentiable optimization: Dini derivatives —
Clarke directional differential
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
Clarke dual action
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
Clarke duality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
Clarke duality theorem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
clarke generalized derivative
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX, 90C26]
(*see:* **Generalized monotone multivalued maps**; **Nonconvex energy functions: hemivariational inequalities**)
Clarke generalized directional derivative
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
Clarke generalized gradient
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)
Clarke generalized gradient
[26E25, 49J52, 52A27, 90C99]
(*see:* **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
Clarke generalized Jacobian
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
Clarke generalized subdifferential
[49J52, 49Q10, 74G60, 74H99, 74K99, 74Pxx, 90C90]
(*see:* **Quasidifferentiable optimization: stability of dynamic systems**)
Clarke–Rockafellar generalized derivative
[90C26]
(*see:* **Generalized monotone multivalued maps**)
Clarke subdifferential
[49J40, 49J52, 65K05, 90C30]
(*see:* **Nonconvex-nonsmooth calculus of variations**; **Solving hemivariational inequalities by nonsmooth optimization methods**)

class
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
class *see:* basic ABS —; closed —; color —; connected —; countable —; finite —; infinite —; quasi-Newton method of Broyden —; scaled ABS —; source —; universal —; unsealed ABS —
class of algorithms *see:* scaled ABS —
class data classification via mixed-integer optimization *see:* Multi- —
class distance *see:* inter- —; intra- —
class invariant under principal pivoting *see:* matrix —
class of a matrix *see:* qualitative —
class P *see:* complexity —
class software package *see:* multiple- —; single- —
class of states *see:* recurrent —; transient —
classes *see:* complexity —; equivalent —; matrix —; Parallel computing: complexity —; phase —; π- —; Sd- —; separable —; set-definable —; σ- —
classes axiom *see:* existence of —
classes in optimization *see:* Complexity —
classes of problems *see:* equivalence —
classical cutting plane method *see:* Kelley's —
classical Gram–Schmidt orthogonalization
[65Fxx]
(*see:* **Least squares problems**)
classical inference
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
classical linear regression model
[90C26, 90C30]
(*see:* **Forecasting**)
classical logic *see:* evaluation in —
classical LU factorization
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
classical Lyusternik theorem
[41A10, 46N10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**; **High-order necessary conditions for optimality for abnormal points**)
classical oligopoly problem
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
classical thermoelastic model
[35R70, 47S40, 74B99, 74D99, 74G99, 74H99]
(*see:* **Quasidifferentiable optimization: applications to thermoelasticity**)
classical traveling salesman problem
[90C27]
(*see:* **Time-dependent traveling salesman problem**)
classification
[03B52, 03E72, 47S40, 62H30, 68T27, 68T35, 68Uxx, 90Bxx, 90C27, 90C29, 90C39, 91Axx, 91B06, 92C60]
(*see:* **Assignment methods in clustering**; **Boolean and fuzzy relations**; **Dynamic programming in clustering**; **Multicriteria sorting methods**)
classification
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
classification *see:* automatic document —; computational issues in —; Deterministic and probabilistic optimization models for data —; document —; Linear programming models for —; optimization in document —; statistical —; statistical pattern —; supervised —; text —; unsupervised —
classification of documents *see:* automatic —
classification error *see:* minimizing the overall —
classification of fractional programs
[90C32]
(*see:* **Fractional programming**)
classification function
[62H30, 90C11]
(*see:* **Statistical classification: optimization approaches**)
classification of hard problems
[90C60]
(*see:* **Computational complexity theory**)
classification of large collections of documents
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
classification of many-valued logics
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
classification matrix
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
classification: optimization approaches *see:* Statistical —
classification problem
[90C09]
(*see:* **Inference of monotone boolean functions**)
classification problem
[90C09]
(*see:* **Inference of monotone boolean functions**)
classification problem *see:* Boolean —; g-group —
classification problem (discriminant problem) *see:* g-group —
classification problems *see:* Mixed integer —; Optimization in boolean —
classification and regression trees
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
classification of text documents
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
classification via mixed-integer optimization *see:* Multi-class data —
classifying declarative programs
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
classifying text documents *see:* Optimization in —
clause
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
clause *see:* logical —
clause at a time *see:* one —
clause at a time algorithm *see:* one —
clause at a time approach *see:* one —

clauses *see:* minimal number of DNF —; minimum number of —
clearly defined end
(*see:* **Planning in the process industry**)
clipping technique
[90C30]
(*see:* **Suboptimal control**)
clique
[03B50, 05C15, 05C17, 05C35, 05C60, 05C62, 05C69, 05C85, 37B25, 65Fxx, 68Q25, 68R10, 68T15, 68T30, 68W40, 90C10, 90C20, 90C22, 90C27, 90C35, 90C59, 91A22]
(*see:* **Domination analysis in combinatorial optimization**; **Finite complete systems of many-valued logic algebras**; **Least squares problems**; **Lovász number**; **Maximum constraint satisfaction: relaxations and upper bounds**; **Multidimensional assignment problem**; **Optimization in leveled graphs**; **Optimization problems in unit-disk graphs**; **Replicator dynamics in combinatorial optimization**; **Standard quadratic optimization problems: applications**)
clique
[05C60, 05C69, 05C85, 37B25, 68W01, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Heuristics for maximum clique and independent set**; **Replicator dynamics in combinatorial optimization**)
clique *see:* maximal —; maximum —; maximum weight —
clique-cut
(*see:* **Contact map overlap maximization problem, CMO**)
clique graph *see:* block- —
clique and independent set *see:* Heuristics for maximum —
clique number
[05C15, 05C17, 05C35, 05C60, 05C62, 05C69, 05C85, 37B25, 68W01, 90C20, 90C22, 90C27, 90C35, 90C59, 91A22]
(*see:* **Heuristics for maximum clique and independent set**; **Lovász number**; **Optimization problems in unit-disk graphs**; **Replicator dynamics in combinatorial optimization**)
clique number *see:* weighted —
clique partition number
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
clique partitioning *see:* minimum —
clique Problem
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
clique problem *see:* max- —; maximum —; maximum weight —
closed *see:* minor —
closed class
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
closed convex cone *see:* pointed —
closed form approach
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
closed form transformation *see:* unimodular max- —
closed form transformations *see:* unimodular max- —
closed function *see:* max- —
closed list
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
closed-loop control
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
closed of a matroid
[90C09, 90C10]
(*see:* **Matroids**)
closed point-to-set mapping
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
closed selfadjoint operator
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
closed set *see:* max- —
closed sets *see:* max- —
closure
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
closure operator for a matroid
[90C09, 90C10]
(*see:* **Matroids**)
closure of a relation *see:* equivalence —; local equivalence —; local pre-order —; local tolerance —; pre-order —; property- —; reflexive —; tolerance —
closures
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
CLP
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
CLP(BNR)
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
cluster
[90C35]
(*see:* **Multi-index transportation problems**)
cluster *see:* admissible —; star —
cluster analysis
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 65K05, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 90-00, 90-08, 90C11, 90C27, 90C35, 92C40, 92E10]
(*see:* **Algorithms for genomic analysis**; Global optimization in protein folding)
cluster compactness
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
cluster first-schedule second strategy
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
cluster-heads
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
cluster isolation
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
cluster second *see:* schedule first- —
cluster statistic *see:* generalized single —
clustering
[62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C27, 90C30, 90C39, 90C90]

(*see:* **Assignment methods in clustering**; **Dynamic programming in clustering**; **Optimization in medical imaging**; **Stochastic global optimization: two-phase methods**)
clustering
[62H30, 90C27, 90C39]
(*see:* **Assignment methods in clustering**; **Dynamic programming in clustering**)
clustering *see:* Assignment methods in —; density —; Dynamic programming in —; fuzzy —; hard —; minimal variance —; Nonsmooth optimization approach to —; order constrained hierarchical —
clustering algorithm
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
clustering approach: global optimum search with enhanced positioning *see:* Gene clustering: A novel decomposition-based —
clustering: A novel decomposition-based clustering approach: global optimum search with enhanced positioning *see:* Gene —
clustering problem
[90C08, 90C11, 90C27]
(*see:* **Quadratic semi-assignment problem**)
clusters *see:* Determining the optimal number of —; Global optimization in Lennard–Jones and morse —; PC —
clusters size threshold *see:* determination of —
CMDT
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
CMO *see:* Contact map overlap maximization problem —
CMST
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
CMST *see:* equal demand —; nonunit weight —; unit weight —
CNF
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
CNF
[90C09, 90C10]
(*see:* **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
CNF *see:* k- —; SAT-k- —
CNF problem *see:* SAT- —
CNSO
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
CNSO *see:* extended real-valued —; multi-objective —; real-valued —
CNSO problems *see:* second order Lagrangian theory of —
CO
[90C27, 90C29]
(*see:* **Multi-objective combinatorial optimization**)
co-channel constrained frequency assignment
[05-XX]
(*see:* **Frequency assignment problem**)
co-coercive operator
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
co-generation plant
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
co-generation plant
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
co-index *see:* linear —; quadratic —
co-optimal path
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
co-optimal vertex
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
coalition *see:* concordant —; discordant —
coarse grained multicomputer
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
coarse grained multicomputer
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
coarse grid
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
coarse valuation structure
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
coarseness
[68Q20]
(*see:* **Optimal triangulations**)
cobipartite neighborhood edge elimination ordering
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
coboundary of a function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
cocircuits *see:* signed —
cocomparability graph
[05C15, 05C62, 05C69, 05C85, 90C27, 90C35, 90C59]
(*see:* **Feedback set problems**; **Optimization problems in unit-disk graphs**)
code *see:* Gray —; RANS —; Reynolds-averaged Navier–Stokes —
code list
[65G20, 65G30, 65G40, 65H20, 65H99, 65K05, 65K99, 90C26, 90C30]
(*see:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Interval analysis: intermediate terms**)
code list
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
code PCSP *see:* computer —
code transformation
[65H99, 65K99]
(*see:* **Automatic differentiation: point and interval**)
code transformation *see:* source —
coderivative *see:* limiting —

codifferentiability
[49J52, 65K99, 70-08, 90C25]
(*see:* **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: codifferentiable functions**)
codifferentiable
[49J52, 65K99]
(*see:* **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**)
codifferentiable *see:* continuously —; twice —; twice continuously —
codifferentiable function
[49J52, 65K99, 70-08, 90C25]
(*see:* **Quasidifferentiable optimization: codifferentiable functions**)
codifferentiable function
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
codifferentiable function *see:* continuously —; Dini —; Hadamard —; twice —; twice continuously —
codifferentiable functions *see:* Quasidifferentiable optimization: —
codifferential
[26B25, 26E25, 49J52, 65K99, 65Kxx, 70-08, 90C25, 90C30, 90C99, 90Cxx]
(*see:* **Nondifferentiable optimization: Newton method**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: codifferentiable functions**)
codifferential *see:* second order —
codifferential descent *see:* method of —
coefficient *see:* activity —; contraction —; expansion —; fugacity —; reflection —
coefficient generation *see:* one-at-a-time —
coefficient matrix *see:* ill-conditioned —
coefficient pivoting rule *see:* Dantzig largest —
coefficient reduction
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
coefficient rule *see:* largest —
coefficients *see:* estimation of kinetic —; flexible MOLP with fuzzy —; generalized linear programming with variable —; MOLP with fuzzy —; multi-objective linear programming with fuzzy —; real —; sensitivity analysis with respect to changes in cost —; statistical representation of cutting plane —; Taylor —
coercive
[90C25, 90C26]
(*see:* **Decomposition in global optimization**)
coercive bilinear symmetric continuous form
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
coercive hemivariational inequality problem
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
coercive operator *see:* co- —
coercivity condition
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
coercivity condition
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
cognitive construct
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
cognitive element
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
cognitive science
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
Cohen triangulation *see:* Hickey– —
cohomology *see:* local system —
cohomology of an arrangement of hyperplanes
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
coin graphs
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
coincidence theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
cold spot
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
cold spots
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
coli *see:* escherichia —
collaborative
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
collaborative optimization
[49M37, 65F10, 65F50, 65H10, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multidisciplinary design optimization**; **Multilevel methods for optimal design**)
collapse *see:* probabilistic —
collapsing auction algorithm *see:* graph —
collapsing in auction algorithms *see:* graph —
collecting traveling salesman problem *see:* prize —
collection
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
collection of margins *see:* hierarchical —
collection of a partition *see:* left- —; right- —
collection of subsets *see:* transversal of a —
collections of documents *see:* classification of large —
collectivety compact
[65H10, 65J15]
(*see:* **Contraction-mapping**)
collision *see:* direct —; hidden —

collocation
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
collocation
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
collocation *see:* orthogonal —
collocation conditions
[34A55, 78A60, 90C30]
(*see:* **Optimal design in nonlinear optics**)
(colloquial) *see:* optimization: definition —
colony *see:* ant —
coloop
[90C09, 90C10]
(*see:* **Matroids**; **Oriented matroids**)
color *see:* double —; single —
color class
[05C15, 05C62, 05C69, 05C85, 90C27, 90C35, 90C59]
(*see:* **Graph coloring**; **Optimization problems in unit-disk graphs**)
color-forced
[05C85]
(*see:* **Directed tree networks**)
colorability *see:* 3- —
colorable *see:* k- —
coloring
[90C35]
(*see:* **Graph coloring**)
coloring
[90C35]
(*see:* **Graph coloring**)
coloring *see:* arc —; conflict-free —; constrained edge —; edge —; frequency exhaustive sequential —; graph —; greedy —; hypergraph q- —; list —; proper —; requirement exhaustive sequential —; t- —; total —; uniform sequential —; weighted —
coloring extension
[05C85]
(*see:* **Directed tree networks**)
coloring frequency assignment *see:* order of a T- —; span of a T- —
coloring heuristic *see:* sequential greedy —
coloring problem *see:* edge —; graph —; m- —; path —; total —; weighted graph —
column *see:* basic —; critical —; nonbasic —
column dropping
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
column dropping rule
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
column generation
[68Q99, 90C06, 90C10, 90C11, 90C25, 90C27, 90C30, 90C35, 90C57]
(*see:* **Branch and price: Integer programming with column generation**; **Frank–Wolfe algorithm**; **Integer programming**; **Multicommodity flow problems**; **Simplicial decomposition**; **Simplicial decomposition algorithms**)
Column generation
[68Q99, 90B90, 90C06, 90C10, 90C11, 90C25, 90C30, 90C35, 90C57, 90C59, 90C90]
(*see:* **Branch and price: Integer programming with column generation**; **Cutting-stock problem**; **Frank–Wolfe algorithm**; **Modeling difficult optimization problems**; **Multicommodity flow problems**; **Simplicial decomposition**; **Simplicial decomposition algorithms**)
column generation *see:* Branch and price: Integer programming with —
column generation formulation
[90C35]
(*see:* **Multicommodity flow problems**)
column generation methods
[90C10, 90C11, 90C27, 90C57]
(*see:* **Set covering, packing and partitioning problems**)
column generation subproblem
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
column incidence graph
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
column-pivoting *see:* QR factorization with —
column sufficient
[90C33]
(*see:* **Linear complementarity problem**)
column sufficient matrix
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
column synthesis *see:* MINLP: reactive distillation —
combination of the extreme points *see:* convex —
combinations *see:* convex —
combinatorial
[90C60]
(*see:* **Computational complexity theory**)
combinatorial algorithm
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
combinatorial fractional programming
[90C32]
(*see:* **Fractional programming**)
Combinatorial matrix analysis
(90C10, 90C09)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Combinatorial optimization games**; **Evolutionary algorithms in combinatorial optimization**; **Fractional combinatorial optimization**; **Multi-objective combinatorial optimization**; **Replicator dynamics in combinatorial optimization**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Combinatorial optimization games**; **Evolutionary algorithms in combinatorial optimization**; **Fractional combinatorial optimization**; **Multi-objective combinatorial optimization**; **Neural networks for combinatorial optimization**; **Replicator dynamics in combinatorial optimization**)
combinatorial matrix analysis
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
combinatorial optimization
[01A99, 05A, 05C60, 05C69, 15A, 37B25, 51M, 52A, 52B, 52C, 60J15, 60J60, 60J70, 60K35, 62H, 62H30, 65C05, 65C10,

65C20, 65H20, 65K05, 68Q, 68R, 68U, 68U20, 68W, 70-08, 82B21, 82B31, 82B41, 82B80, 90-01, 9008, 90B, 90B40, 90C, 90C10, 90C11, 90C20, 90C26, 90C27, 90C29, 90C35, 90C39, 90C57, 90C59, 90C60, 91A12, 91A22, 92C40, 92E10, 94C15]
(*see:* **Combinatorial optimization games**; **Convex discrete optimization**; **Dynamic programming in clustering**; Global optimization in protein folding; **Greedy randomized adaptive search procedures**; **History of optimization**; **Integer programming**; **Multi-objective combinatorial optimization**; **Replicator dynamics in combinatorial optimization**; **Variable neighborhood search methods**)

combinatorial optimization
[05-04, 62H30, 65H20, 65K05, 68M20, 68T99, 90-01, 90B06, 90B10, 90B35, 90B40, 90B80, 90C08, 90C09, 90C10, 90C11, 90C15, 90C20, 90C27, 90C29, 90C30, 90C35, 90C39, 90C57, 90C59, 90C60, 91A12, 94C15]
(*see:* **Assignment methods in clustering**; **Bi-objective assignment problem**; **Capacitated minimum spanning trees**; **Combinatorial optimization games**; **Computational complexity theory**; **Dynamic programming in clustering**; **Evolutionary algorithms in combinatorial optimization**; **Feedback set problems**; **Greedy randomized adaptive search procedures**; **Integer programming**; **Linear ordering problem**; **Matroids**; **Multidimensional knapsack problems**; **Multi-objective combinatorial optimization**; **Neural networks for combinatorial optimization**; **Oriented matroids**; **Quadratic assignment problem**; **Set covering, packing and partitioning problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic integer programs**; **Vehicle scheduling**)

combinatorial optimization *see:* convex —; Domination analysis in —; Evolutionary algorithms in —; Fractional —; large-scale —; linear fractional —; multi-objective —; Neural networks for —; Replicator dynamics in —; stochastic —; uniform fractional —

Combinatorial optimization algorithms in resource allocation problems
(90C09, 90C10)
(*referred to in:* **Combinatorial matrix analysis**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Fractional combinatorial optimization**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Multi-objective combinatorial optimization**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Replicator dynamics in combinatorial optimization**; **Resource allocation for epidemic control**; **Simple recourse problem**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Combinatorial matrix analysis**; **Combinatorial optimization games**; **Competitive facility location**; **Evolutionary algorithms in combinatorial optimization**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Fractional combinatorial optimization**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Multi-objective combinatorial optimization**; **Network location: covering problems**; **Neural networks for combinatorial optimization**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Replicator dynamics in combinatorial optimization**; **Resource allocation for epidemic control**; **Simple recourse problem**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)

combinatorial optimization game
[90C27, 90C60, 91A12]
(*see:* **Combinatorial optimization games**)

Combinatorial optimization games
(91A12, 90C27, 90C60)
(*referred to in:* **Combinatorial matrix analysis**; **Combinatorial optimization algorithms in resource allocation problems**; **Evolutionary algorithms in combinatorial optimization**; **Fractional combinatorial optimization**; **Multi-objective combinatorial optimization**; **Replicator dynamics in combinatorial optimization**)
(*refers to:* **Combinatorial matrix analysis**; **Evolutionary algorithms in combinatorial optimization**; **Fractional combinatorial optimization**; **Multi-objective combinatorial optimization**; **Neural networks for combinatorial optimization**; **Replicator dynamics in combinatorial optimization**)

combinatorial optimization problem
[90C27, 90C30, 90C60]
(*see:* **Computational complexity theory**; **Neural networks for combinatorial optimization**)

combinatorial optimization problem *see:* fractional —; integral linear fractional —

combinatorial optimization problems
[90C11, 90C59]
(*see:* **Nested partitions optimization**)

combinatorial problem
[90C26, 90C90]
(*see:* **Structural optimization: history**)

combinatorial properties
[68Q20]
(*see:* **Optimal triangulations**)

combinatorial switching circuit
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)

Combinatorial test problems and problem generators
(90B99, 05A99)
(*referred to in:* **Maximum cut problem, MAX-CUT**)

combinatorics
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)

combinatorics *see:* polyhedral —

combine
[46N10, 47N10, 49M37, 65K10, 90C26, 90C30]
(*see:* Global optimization: tight convex underestimators)
combined method of feasible directions
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
combined relative measure
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
combined relaxation methods
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
commodity
[90C35]
(*see:* **Multicommodity flow problems**)
commodity flows *see:* Multi- —
commodity model in OR *see:* single- —
commodity network flow problem *see:* nonlinear single —
commodity single-criterion uncapacitated static multifacility *see:* discrete single- —
common dependency set
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
common mutated sequence *see:* minimum weight —
common random numbers
[62F12, 65C05, 65K05, 90C15, 90C27, 90C31]
(*see:* **Discrete stochastic optimization**; **Monte-Carlo simulations for stochastic optimization**)
communication
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
communication *see:* open —; rendez-vous —
communication costs
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
communication equilibrium
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
communication-free alignment
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
communication-free alignment problem
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
communication network
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
Communication network assignment problem
(90B80, 90C05, 90C27, 68Q25)
(*referred to in:* **Assignment and matching**; **Assignment methods in clustering**; **Auction algorithms**; **Bi-objective assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Frequency assignment problem**; **Generalized networks**; **Linear ordering problem**; **Maximum flow problem**; **Maximum partition matching**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Quadratic assignment problem**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
(*refers to:* **Assignment and matching**; **Assignment methods in clustering**; **Auction algorithms**; **Bi-objective assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Frequency assignment problem**; **Generalized networks**; **Maximum flow problem**; **Maximum partition matching**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Quadratic assignment problem**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
communication network assignment problem
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
communication protocol
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
commutator K
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
compact
[49L20, 49M29, 65K10, 90C06, 90C40]
(*see:* **Dynamic programming: undiscounted problems**; **Local attractors for gradient-related descent iterations**)
compact *see:* collectivety —
compact epi-Lipschitzness
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
compact graph
[49J20, 49J52]
(*see:* **Shape optimization**)
compact operator
[49J52]
(*see:* **Hemivariational inequalities: eigenvalue problems**)
compact representation
[90C30, 90C35]
(*see:* **Optimization in water resources**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
compact representations
[90C39]
(*see:* **Neuro-dynamic programming**)
compactness
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
compactness *see:* cluster —; partial sequential normal —; sequential normal —; weak —
company policies *see:* in- —
comparative efficiency assessment
[90B30, 90B50, 90C05, 91B82]
(*see:* **Data envelopment analysis**)
comparison *see:* paired —; sequence —; technological —
comparison of efficiency and nondomination
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
comparison oracle
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W,

90B, 90C]
(*see:* **Convex discrete optimization**)
comparison of parametric solutions
[90C11, 90C31]
(*see:* **Multiparametric mixed integer linear programming**)
comparisons *see:* missing —; pairwise —
compatibility *see:* backward —; both-way —; forward —
compatibility condition
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)
compatibility conditions
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
compatibility equations *see:* strain-displacement —
compatibility theorem *see:* Bandler–Kohout —
competition *see:* imperfect —; perfect —
competition facility location model *see:* spatial —
competitive *see:* perfectly —
competitive analysis
[68Q25, 91B28]
(*see:* **Competitive ratio for portfolio management**)
competitive environment
[90B80, 90B85]
(*see:* **Warehouse location problem**)
competitive equilibrium model *see:* perfectly —
Competitive facility location
(90B60, 90B80, 90B85)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
competitive ratio
[05C85, 68Q25, 91B28]
(*see:* **Competitive ratio for portfolio management**; **Directed tree networks**)
competitive ratio
[68Q25, 91B28]
(*see:* **Competitive ratio for portfolio management**)
Competitive ratio for portfolio management
(91B28, 68Q25)
(*referred to in:* **Financial applications of multicriteria analysis**; **Financial optimization**; **Portfolio selection and multicriteria analysis**; **Robust optimization**; **Semi-infinite programming and applications in finance**)
(*refers to:* **Financial applications of multicriteria analysis**; **Financial optimization**; **Portfolio selection and multicriteria analysis**; **Robust optimization**; **Semi-infinite programming and applications in finance**)
complement
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
complement
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
complement *see:* relative —; Schur —
complement of an arrangement of hyperplanes
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
complement graph
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
complementarity
[90C10, 90C11, 90C27, 90C30, 90C33]
(*see:* **Continuous reformulations of discrete-continuous optimization problems**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Topological methods in complementarity theory**)
complementarity
[49-01, 49-XX, 49K10, 49M37, 90-01, 90-XX, 90C05, 90C27, 90C30, 90C33, 91B52, 93-XX]
(*see:* **Bilevel linear programming**; **Duality for semidefinite programming**; **Duality theory: monoduality in convex optimization**; **Topological methods in complementarity theory**)
complementarity *see:* generalized —; linear —; nonlinear —; Order —; strict —
Complementarity algorithms in pattern recognition
(*referred to in:* **Generalizations of interior point methods for the linear complementarity problem**; **Simultaneous estimation and optimization of nonlinear problems**)
(*refers to:* **Generalizations of interior point methods for the linear complementarity problem**; Generalized eigenvalue proximal support vector machine problem)
complementarity condition
[49M37, 65K05, 90C30]
(*see:* **Image space approach to optimization**; **Inequality-constrained nonlinear optimization**)
complementarity condition
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
complementarity problem
[90C33]
(*see:* **Generalized nonlinear complementarity problem**; **Linear complementarity problem**)
complementarity problem
[65K10, 65M60, 90C30, 90C31, 90C33]
(*see:* **Implicit lagrangian**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Sensitivity analysis of complementarity problems**; **Variational inequalities**)
complementarity problem *see:* discrete dynamic —; dynamic —; extended linear —; general order —; Generalizations of interior point methods for the linear —; generalizations of the nonlinear —; generalized —; generalized linear order —; generalized mixed —; Generalized nonlinear —; generalized order —; horizontal linear —; implicit —; implicit general order —;

infinite-dimensional generalized order —; linear —; linear order —; mixed linear —; nonlinear —; nonlinear order —; order —; parametric linear —; parametric nonlinear —; vertical linear —
complementarity problem and fixed point problem *see:* Equivalence between nonlinear —
complementarity problems *see:* linear —; nonlinear —; parametric —; Principal pivoting methods for linear —; Sensitivity analysis of —; Splitting method for linear —
complementarity problems and variational inequalities *see:* Nonsmooth and smoothing methods for nonlinear —
complementarity slackness
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
complementarity slackness *see:* strict —
complementarity slackness condition *see:* strict —
complementarity theory *see:* Topological methods in —
complementary basis
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**; **Principal pivoting methods for linear complementarity problems**)
complementary conditions *see:* strictly —
complementary gap function
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
complementary gap function *see:* pure —
complementary graph
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
complementary ions
(*see:* **Peptide identification via mixed-integer optimization**)
complementary operator
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
complementary pair of variables
[90C33]
(*see:* **Lemke method**)
complementary pivot methods
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)
complementary pivot theory
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
complementary problem *see:* Integer linear —; linear —; nonlinear —
complementary region
[90C11, 90C59]
(*see:* **Nested partitions optimization**)
complementary relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
complementary slackness
[15A39, 68W10, 90B15, 90C05, 90C06, 90C10, 90C30, 90C31, 90C33, 90C34, 90C46, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **Linear complementarity problem**; **Semi-infinite programming: methods for linear problems**; **Sensitivity and stability in NLP: continuity and differential stability**; **Stochastic network problems: massively parallel solution**; **Tucker homogeneous systems of linear relations**)
complementary slackness *see:* ϵ- —; strict —
complementary slackness conditions
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
complementary slackness relations
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
complementary solution *see:* strictly —
complementary solutions *see:* almost —
complete
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
complete *see:* NP- —
complete completeness *see:* strong NP- —
complete digraph
[90C10, 90C11, 90C20]
(*see:* **Linear ordering problem**)
complete game
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
complete graph
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
complete language *see:* F- —
complete many-valued logic normal form
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
complete master problem
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
complete normal forms of Pi-algebras *see:* functionally —
complete orientation
[90B35]
(*see:* **Job-shop scheduling problem**)
complete orthogonal factorization
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **Orthogonal triangularization**)
complete orthogonal factorization
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **Orthogonal triangularization**; **QR factorization**)
complete problem *see:* NP- —
complete problems and proof methodology *see:* NP- —
complete recourse
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**; **Stochastic programming: nonanticipativity and lagrange multipliers**)
complete recourse *see:* relatively —
complete reduction
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
complete reductions *see:* ordinary NP- —
complete set of connectives
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
complete systems of many-valued logic algebras *see:* Finite —
completely continuous operator
[46N10, 49J40, 90C26]

(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
completely positive
[90C22, 90C25]
(*see:* **Copositive programming**)
completely positive and contraction matrices *see:* completion to —
completely positive matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
completely positive matrix *see:* partial —
completely regular set
[90C33]
(*see:* **Order complementarity**)
completeness *see:* F- —; functional —; NP- —; ordinary NP- —; strong NP- —; strong NP-complete —
completeness of PI-algebras *see:* functional —
completeness proofs *see:* NP- —
completion *see:* rank matrix —
completion to completely positive and contraction matrices
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
completion of matrices
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
completion of a partial matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
completion problem *see:* distance matrix —; Euclidean distance matrix —; matrix —; maximum rank —; minimum rank —; positive (semi) definite —; positive semidefinite matrix —
completion problems *see:* Matrix —
completion time
[90B36]
(*see:* **Stochastic scheduling**)
complex interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
complex interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
complexities *see:* predictability of —
complexity
[65K05, 68Q25, 90C08, 90C11, 90C27, 90C30, 90C57, 90C59, 90C60]
(*see:* **Automatic differentiation: point and interval taylor operators**; **NP-complete problems and proof methodology**; **Quadratic assignment problem**)
complexity
[90B35, 90C20, 90C25, 90C27, 90C30, 90C60, 90C90, 91A12]
(*see:* **Chemical process planning**; **Combinatorial optimization games**; **Complexity classes in optimization**; **Complexity theory**; **Complexity theory: quadratic programming**; **Job-shop scheduling problem**; **Kolmogorov complexity**; **Kuhn–Tucker optimality conditions**; **Quadratic programming over an ellipsoid**)
complexity *see:* Algorithmic —; average case —; computational —; conditional Kolmogorov —; Descriptional —; descriptive —; exponential —; graver —; information-based —; Kolmogorov —; PLS- —; polynomial —; Regression by special functions: algorithms and —; Solomonoff–Kolmogorov–Chaitin —; worst-case —
complexity of an algorithm
[90C60]
(*see:* **Computational complexity theory**)
complexity of algorithms *see:* average case —
complexity of bilevel programming
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)
complexity class P
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
complexity classes
[03D15, 68Q05, 68Q15, 90C60]
(*see:* **Complexity classes in optimization**; **Parallel computing: complexity classes**)
complexity classes
[03D15, 68Q05, 68Q15, 90C60]
(*see:* **Complexity classes in optimization**; **Parallel computing: complexity classes**)
complexity classes *see:* Parallel computing: —
Complexity classes in optimization
(90C60)
(*referred to in:* **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Facilities layout problems**; **Fractional combinatorial optimization**; **Global optimization in multiplicative programming**; **Information-based complexity and information-based optimization**; **Interval Newton methods**; **Job-shop scheduling problem**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **Multifacility and restricted location problems**; **Multiplicative programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**; **Vehicle scheduling**)
(*refers to:* **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**)
Complexity of degeneracy
(90C60)
(*referred to in:* **Complexity classes in optimization**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**)
(*refers to:* **Complexity classes in optimization**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**;

Computational complexity theory; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**)
complexity of a deterministic Turing machine *see:* space —; time —
complexity, equivalence to minmax, concave programs *see:* Bilevel linear programming: —
complexity and equivalent forms *see:* Quadratic integer programming: —
complexity function *see:* time —
complexity function of an algorithm *see:* time —
Complexity of gradients, Jacobians, and Hessians
(65D25, 68W30)
(*referred to in:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**)
(*refers to:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**)
complexity and information-based optimization *see:* Information-based —
Complexity and large-scale least squares problems
(93E24, 34-xx, 34Bxx, 34Lxx)
complexity of the linear BLPP
[49-01, 49K45, 49N10, 90-01, 90C20, 90C27, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**)
complexity of models
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
complexity of a nondeterministic Turing machine *see:* space —; time —
complexity $\mathcal{O}(n^c)$ *see:* algorithm of —
complexity of optimization problems *see:* computational —
Complexity theory
(90C60)
(*referred to in:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Facilities layout problems**; **Fractional combinatorial optimization**; **Global optimization in multiplicative programming**; **Information-based complexity and information-based optimization**; **Job-shop scheduling problem**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **Multifacility and restricted location problems**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**; **Quadratic assignment problem**; **Quadratic knapsack**; **Vehicle scheduling**)
(*refers to:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**; **Shortest path tree algorithms**)
complexity theory
[90C60]
(*see:* **Complexity theory**; **Computational complexity theory**)
complexity theory
[90C60]
(*see:* **Computational complexity theory**)
complexity theory *see:* Computational —
complexity theory of algorithms
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
complexity theory of PI-algebras
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
Complexity theory: quadratic programming
(90C60)
(*referred to in:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Linear ordering problem**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
(*refers to:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
complexity of Turing machines
[90C60]
(*see:* **Complexity classes in optimization**)

compliance
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
complicating variables
[90C26]
(*see:* **Generalized primal-relaxed dual approach**)
component
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
component *see:* basic —; central —; control —; feasible —; infeasible —; linear —; nonbasic —; noncentral —; polygonal —; singular —
component species
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
components *see:* full —
components of a digraph *see:* strongly connected —
components of a matrix *see:* irreducible —
componentwise bound *see:* optimal —
Composite Convexifiable Function *see:* integral Mean-Value for —
composite materials *see:* laminated —
Composite nonsmooth optimization
(46A20, 90C30, 52A01)
(*referred to in:* **Nonconvex-nonsmooth calculus of variations**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Solving hemivariational inequalities by nonsmooth optimization methods**)
(*refers to:* **Nonconvex-nonsmooth calculus of variations**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Vector optimization**)
composite programming *see:* convex —
composite structures *see:* design of —; Optimal design of —
composition
[90C09, 90C10]
(*see:* **Oriented matroids**)
composition difference *see:* minimum —
composition interval diagram
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
composition of relations *see:* round —; square —
composition theorem
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
compositional models
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
compositions *see:* equality of phase —; relational —
compression
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
compromise programming
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
compromise solution *see:* best- —
computability *see:* partial —
computable function *see:* polynomial time —
computable optimal value bounds
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
computable solution
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
computation
[33C45, 65F20, 65F22, 65K10, 90C10, 90C30]
(*see:* **Least squares orthogonal polynomials**; **Modeling languages in optimization: a new paradigm**)
Computation
[90C60]
(*see:* **Kolmogorov complexity**)
computation *see:* asynchronous —; Automatic differentiation: parallel —; conjugate gradient parameter —; direction —; fixed point —; model of —; parallel —; partially asynchronous —
computation and data mapping
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
computation in mechanics *see:* parallel —
computation for polytopes: strategies and performances *see:* Volume —
computation thesis *see:* parallel —
computation of a Turing machine *see:* accepting —; length of a partial —; nonaccepting —; partial —
computational biology
[90C35]
(*see:* **Optimization in leveled graphs**)
computational certainty *see:* mathematical and —
computational complexity
[41A30, 62J02, 90C08, 90C11, 90C26, 90C27, 90C57, 90C59, 90C60, 91A12]
(*see:* **Combinatorial optimization games**; **Computational complexity theory**; **Quadratic assignment problem**; **Regression by special functions: algorithms and complexity**)
computational complexity
[03B50, 49-01, 49K45, 49N10, 68Q25, 68T15, 68T30, 90-01, 90B80, 90C05, 90C20, 90C27, 90C60, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Communication network assignment problem**; **Computational complexity theory**; **Finite complete systems of many-valued logic algebras**; **NP-complete problems and proof methodology**)
computational complexity of optimization problems
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
Computational complexity theory
(90C60)
(*referred to in:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **Multiplicative**

programming; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**; **Quadratic assignment problem**; **Quadratic knapsack**)
(*refers to:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **Parallel computing: complexity classes**)
computational differentiation
[26A24, 34-XX, 49-XX, 65-XX, 65D25, 68-XX, 68W30, 90-XX]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**; **Complexity of gradients, Jacobians, and Hessians**; **Nonlocal sensitivity analysis with automatic differentiation**)
computational efficiency
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
computational equivalence
[90C34]
(*see:* **Semi-infinite programming: approximation methods**)
computational equivalent
[90C34]
(*see:* **Semi-infinite programming: approximation methods**)
computational fluid dynamics
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
computational fluid dynamics
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
computational fluid dynamics *see:* Design optimization in —
computational graph
[26A24, 65D25, 65K05, 68W30, 90C26, 90C30]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Complexity of gradients, Jacobians, and Hessians**)
computational issues in classification
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
computational linguistics
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
computational mechanics
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
computational method *see:* adaptive —
computational methods
[65H99, 65K99]
(*see:* **Automatic differentiation: point and interval**)
computational model
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
computational nonredundancy
[90C30]
(*see:* **Cost approximation algorithms**)
computational performance
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
computational performance *see:* optimization of —
computational plasticity
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
computational process
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)
computational solution *see:* practically feasible —
computational step
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)
computationally equivalent semi-infinite programs
[90C34]
(*see:* **Semi-infinite programming: approximation methods**)
computations *see:* interval —; parallel —; uniform —
compute *see:* easy-to- —
compute a safeguarded new trial steplength
[49M07, 49M10, 65K, 90C06, 90C20]
(*see:* **Spectral projected gradient methods**)
compute the search direction
[49M07, 49M10, 65K, 90C06, 90C20]
(*see:* **Spectral projected gradient methods**)
compute the steplength
[49M07, 49M10, 65K, 90C06, 90C20]
(*see:* **Spectral projected gradient methods**)
computer *see:* bulk synchronous parallel —; distributed memory parallel —; parallel —
computer aided techniques
[90C26, 90C30]
(*see:* **Forecasting**)
computer algebra
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
computer algebra
[13Cxx, 13Pxx, 14Qxx, 65D25, 68W30, 90Cxx]
(*see:* **Complexity of gradients, Jacobians, and Hessians**; **Integer programming: algebraic methods**)
computer algebra package
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
computer code PCSP
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
computer implementation example *see:* optimization —
Computer implementation of optimization
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
computer network *see:* local-area —
computerized tomography
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
computing
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)

computing *see:* distributed —; high performance —; interval —; massively parallel —; models for parallel —; parallel —
computing: complexity classes *see:* Parallel —
computing: models *see:* Parallel —
computing processes in interactive methods
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
computing system *see:* high performance —
concave case *see:* convex- —
concave fractional program
[90C32]
(*see:* **Fractional programming**)
concave function
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
concave function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
concave function *see:* α- —; U- —
concave functions *see:* product of —
concave increasing
[65D18, 90B85, 90C26]
(*see:* **Global optimization in location problems**)
concave measure *see:* α- —
concave minimization
[90B10, 90C25]
(*see:* **Concave programming**; **Piecewise linear network flow problems**)
concave probability measure *see:* γ- —
Concave programming
(90C25)
(*referred to in:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Global optimization in multiplicative programming**; **Minimum concave transportation problems**; **Multiplicative programming**; **Quadratic assignment problem**; **Reverse convex optimization**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Minimum concave transportation problems**)
concave programming
[90C25, 90C90]
(*see:* **Chemical process planning**; **Concave programming**)
concave programming
[49-01, 49K45, 49N10, 90-01, 90C20, 90C25, 90C27, 90C90, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Chemical process planning**; **Concave programming**)
concave programming *see:* quadratic —
concave programs *see:* Bilevel linear programming: complexity, equivalence to minmax —
concave quadratic programming
[90C60]
(*see:* **Complexity theory: quadratic programming**)
concave regression *see:* convex and —
concave transportation problem *see:* minimum —
concave transportation problems *see:* Minimum —
concavity *see:* property of —; vector generalized —
concavity cut
[90C26]
(*see:* **Cutting plane methods for global optimization**)
concavity cut
[90C26]
(*see:* **Cutting plane methods for global optimization**)
concavity in multi-objective optimization *see:* Generalized —
concentration theorem *see:* Jaynes entropy —
concept in auction algorithms *see:* virtual source —
conceptual design stage
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
conceptual diagram
(*see:* **State of the art in modeling agricultural systems**)
conceptual modeling
(*see:* **State of the art in modeling agricultural systems**)
conciseness
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
conclusion
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
concordance
[90-XX]
(*see:* **Outranking methods**)
concordance condition
[90-XX]
(*see:* **Outranking methods**)
concordance-discordance
[90-XX]
(*see:* **Outranking methods**)
concordance level
[90-XX]
(*see:* **Outranking methods**)
concordant coalition
[90-XX]
(*see:* **Outranking methods**)
concurrent subspace
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
concurrent subspace optimization
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
condensation *see:* posynomial —
condensing operator
[90C33]
(*see:* **Order complementarity**)
CoNDEXPTIME
[90C60]
(*see:* **Complexity classes in optimization**)
condition *see:* acute angle —; calmness —; coercivity —; compatibility —; complementarity —; concordance —; conjugacy —; diagonal dominance —; first order necessary —; Fritz John type —; general second order sufficient —; general strong second order sufficient —; high-order local minimum —; Karush–Kuhn–Tucker type —; Kirchhoff- —; Kuhn–Tucker optimality —; linear growth —; linear nondegeneracy —; maximum —; necessary optimality —; nondegeneracy —; nondiscordance —;

nonobtuse angle —; nontriviality —; orthogonal —; partial calmness —; quadratic nondegeneracy —; regularity —; saddle-point sufficient —; sandwich —; second order necessary —; second order optimality —; second order sufficient —; separation —; Signorini —; Slater's —; strict complementarity slackness —; strict feasibility —; strong second order sufficient —; sufficient —; sufficient optimality —; superlinear convergence —; uniform angle —; unilateral growth —
condition iteration BCI *see:* Boundary —
condition iteration method *see:* boundary —
condition for LDSU *see:* nonarbitrage —
condition measures
[90C05, 90C22, 90C25, 90C30, 90C51]
(*see:* **Interior point methods for semidefinite programming**)
condition number
[15-XX, 49M37, 65-XX, 90-XX, 90C31]
(*see:* **Cholesky factorization**; **Nonlinear least squares: Newton-type methods**; **Sensitivity and stability in NLP: approximation**)
condition number *see:* normwise relative —
condition number of a matrix
[65Fxx]
(*see:* **Least squares problems**)
condition for penalty methods *see:* regularity —
condition without using (sub)gradients parametric representations *see:* necessary optimality —
conditional *see:* logic —
conditional expectation constraint
[90C15]
(*see:* **Static stochastic programming models: conditional expectations**)
conditional expectations *see:* Static stochastic programming models: —
conditional gradient method
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
conditional Kolmogorov complexity
[90C60]
(*see:* **Kolmogorov complexity**)
conditional lower derivative *see:* Dini —; Hadamard —
conditional proximity data *see:* row —
conditional upper derivative *see:* Dini —; Hadamard —
conditionally differentiable function *see:* Dini —; Hadamard —
conditionally directionally differentiable function *see:* Dini —; Hadamard —
conditioned coefficient matrix *see:* ill- —
conditioned matrix *see:* ill- —; well- —
conditioned problem *see:* ill- —; well- —
conditions *see:* boundary —; boundary value —; collocation —; compatibility —; complementary slackness —; continuity —; cut —; economic system —; elastostatics with nonlinear boundary —; Equality-constrained nonlinear programming: KKT necessary optimality —; first order KKT —; first order necessary —; first order necessary optimality —; first order and second order optimality —; Fritz John —; Fritz John generalized —; fritz John necessary optimality —; generalized Karush–Kuhn–Tucker —; generalized necessary optimality —; Generalized semi-infinite programming: optimality —; Goldstein —; Hebden —; Karush–Kuhn–Tucker —; Karush–Kuhn–Tucker optimality —; KKT —; KKT necessary optimality —; KKT optimality —; KKT stationarity —; KT —; Kuhn–Tucker —; Kuhn–Tucker necessary optimality —; Kuhn–Tucker optimality —; Lagrangian —; market equilibrium —; matching of derivative —; moment —; necessary —; necessary optimality —; necessary and sufficient —; necessary and sufficient optimality —; nonstoichiometric form of KT —; optimal integral bounds subject to moment —; optimality —; Penrose —; point —; Post —; Quasidifferentiable optimization: optimality —; quasidifferential elastic boundary —; quasidifferential thermal boundary —; regularity —; Saddle point theory and optimality —; second order necessary —; second order necessary and sufficient optimality —; second order sufficient —; Semi-infinite programming: second order optimality —; stoichiometric form of KT —; strictly complementary —; sufficient —; sufficient decrease —; sufficient optimality —; uniform Hölder —; validity —; variational formulation of quasidifferential thermal boundary —
conditions for a constrained optimum
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
conditions and duality *see:* Bilevel programming: optimality —
conditions moment problem *see:* infinite many —
conditions on multipliers *see:* orthogonality —
conditions for nonlinear optimization *see:* Second order optimality —
conditions for optimality *see:* high-order necessary —
conditions for optimality for abnormal points *see:* High-order necessary —
conditions for quadratic programming sub-problems *see:* Kuhn–Tucker —
conditions and stability *see:* Semidefinite programming: optimality —
conditions for an unconstrained optimum
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
Condorcet paradox
[90-XX]
(*see:* **Outranking methods**)
conduction *see:* Fourier law of heat —; heat —
cone *see:* Bouligand —; Bouligand tangent —; contingent —; convex —; critical —; dual —; fréchet normal —; Galerkin —; high-order approximating —; inner linearization —; isotone projection —; limiting normal —; minimal —; normal —; order —; outer linearization —; Paretian —; pointed closed convex —; pointed convex —; polar —; second order —; secondary —; tangent —; tangent high-order approximating —; z-critical —
cone-convex map
[49K27, 90C29, 90C48]
(*see:* **Set-valued optimization**)
cone of critical directions
[90C31, 90C34]
(*see:* **Semi-infinite programming: second order optimality conditions**)
cone of decrease *see:* high-order approximating —

cone of feasible directions
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
cone property *see:* uniform —
cones *see:* convex —; feasible high-order approximating —; high-order feasible —; homogenous —; ordering —; tangent high-order approximating —
cones of decrease *see:* high-order —
confidence interval
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
configuration
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
configuration
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
configuration *see:* Monte-Carlo —; point —; search —; unyielding —; vector —
configuration space *see:* local properties of the —
confirmans *see:* modus —
confirmation
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
confirmation *see:* checklist —
conflict-free coloring
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
conflict graph
[05C85, 65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**; **Directed tree networks**)
conflicting populations *see:* Volterra model of —
conformation
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
conformation
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
conformation *see:* molecular —; native —
conformational search
[65K10, 92C40]
(*see:* **Multiple minima problem in protein folding: αBB global optimization approach**)
conformations *see:* discarding far-from-native —
conformity *see:* uniformity —
confusion matrix
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
congested network
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
conic convex program *see:* semidefinite program as —
conic convex programs
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
conic duality theorem
[90C22, 90C25]
(*see:* **Copositive programming**)
conic extension
[90C30]
(*see:* **Image space approach to optimization**)
conic optimization problems *see:* Approximations to robust —
conic program
[37A35, 90C05]
(*see:* **Potential reduction methods for linear programming**)
conical algorithm
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
conjecture *see:* Chung–Gilbert —; Gilbert–Pollak —; Graham–Hwang —; Jerrum —; powell's —; Rosen's method, global convergence, and Powell's —; Smith —
conjugacy condition
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)
conjugate *see:* Legendre —
conjugate direction subclass
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
conjugate function
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
conjugate functions
[46A20, 49J40, 52A01, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05, 90C30]
(*see:* **Farkas lemma: generalizations**; **Variational principles**)
conjugate functions *see:* Fenchel —
conjugate gradient
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**)
conjugate gradient algorithms for unconstrained optimization *see:* New hybrid —; Performance profiles of —
conjugate gradient method
[49M37, 90C30]
(*see:* **Nonlinear least squares: trust region methods**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
conjugate gradient method *see:* Shanno —
Conjugate-gradient methods
(90C30)
(*referred to in:* **Broyden family of methods and the BFGS update**; **Discontinuous optimization**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Local attractors for gradient-related descent iterations**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**; **Unconstrained optimization in neural network training**)
(*refers to:* **Broyden family of methods and the BFGS update**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Local attractors for gradient-related descent iterations**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**; **Unconstrained nonlinear optimization:**

Newton–Cauchy framework; **Unconstrained optimization in neural network training**)
conjugate gradient methods
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
conjugate gradient parameter computation
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)
conjugate gradient type algorithm *see:* Craig —
conjugate gradients
[90C30]
(*see:* **Conjugate-gradient methods**)
conjugate pair
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
conjugate residual algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
conjugate subgradient method
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
conjugation *see:* level sets —
conjunction arc
[90B35]
(*see:* **Job-shop scheduling problem**)
conjunctive normal form
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T15, 68T27, 68T30, 90C09, 90C10, 90C11]
(*see:* **Checklist paradigm semantics for fuzzy logics**; **Disjunctive programming**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
conjunctive normal form
[90C09, 90C10]
(*see:* **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
conjunctive normal form *see:* Boolean formula in —
conjunctive use of water resource systems
[90C30, 90C35]
(*see:* **Optimization in water resources**)
connected
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)
connected class
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
connected components of a digraph *see:* strongly —
connected cycle *see:* k-dimensional cube —
connected digraph *see:* strongly —
connected dominating set
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
connected edge list *see:* extended doubly —
connected graph
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
connected matroid *see:* infinitely —
connected network *see:* strongly —
connected set
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
connectedness
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 90C29, 91A05]
(*see:* **Minimax theorems**; **Multi-objective optimization: pareto optimal solutions, properties**)
connectedness
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 90C29, 91A05]
(*see:* **Generalized concavity in multi-objective optimization**; **Minimax theorems**)
connectedness of the efficient points sets
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
connecting path *see:* elementary —
connection *see:* train-to-train —
connection arc *see:* arrival-ground —; ground-departure —
connection arcs *see:* train-train —
connection of flow lines
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
connection of wells
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
connective *see:* associative —; bold —; Boolean —; logic algebra —; Łukasiewicz —; mid —
connectives *see:* complete set of —; emergence of logic —; logic —; semantics of MVL —; TOP and BOT types of logical —
connectivity *see:* matroid —; network —
connectivity graph
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
connectivity of a matroid
[90C09, 90C10]
(*see:* **Matroids**)
conorm *see:* t- —
conormal
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
conorms *see:* t- —
CoNP
[90C60]
(*see:* **Complexity classes in optimization**)
conquer *see:* divide-and- —
consecutive one constraint
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
conservation constraint *see:* flow —
conservation of flow
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
conservation of flow equation
[90C30]
(*see:* **Equilibrium networks**)
conservation of flow equations
[90B10, 91B28, 91B50]

(*see:* **Piecewise linear network flow problems**; **Spatial price equilibrium**)
conservation law *see:* flow —
conservation laws
[90B36]
(*see:* **Stochastic scheduling**)
considerations *see:* uncertainty —
considerations and controllability *see:* integration of dynamic —
consist-busting
(*see:* **Railroad locomotive scheduling**)
consist flow formulation
(*see:* **Railroad locomotive scheduling**)
consistency *see:* 2B- —; 3B- —; arc —; bound —; box —; box(2)- —; hull —; kB- —; local —; total —; zone —
consistency constraints
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
consistency index
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
consistency property *see:* Jacobian —
consistency ratio
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
consistent *see:* locally —
consistent case *see:* perfectly —
consistent judgment matrix
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
consistent labeling
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**; **Least-index anticycling rules**)
consistent labeling algorithm
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
consistent least squares problem
[65Fxx]
(*see:* **Least squares problems**)
consistent matrix
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
consistent rounding
[90C35]
(*see:* **Maximum flow problem**)
consistent variable
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
conspiracy number
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
constant
[68Q20]
(*see:* **Optimal triangulations**)
constant *see:* Boltzmann —; Lipschitz —
constant control *see:* piecewise —
constant degree parallelism alignment problem
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
constant maturities *see:* estimating the spot rate for bonds with —
constant permutation QAP
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
constant perturbations *see:* piecewise- —
constant rebalanced portfolio
[68Q25, 91B28]
(*see:* **Competitive ratio for portfolio management**)
Constantine *see:* Carathéodory —; Metropolis, Nicholas —
constants
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
constants *see:* Arrhenius —
constrained edge coloring
[05C85]
(*see:* **Directed tree networks**)
constrained frequency assignment *see:* adjacent channel —; co-channel —
constrained global optimization
[90C26]
(*see:* **Global optimization using space filling**)
constrained global optimum
[90C26]
(*see:* **Global optimization using space filling**)
constrained hierarchical clustering *see:* order —
constrained labeling *see:* distance —
constrained linear problem *see:* ball- —
constrained linear programming *see:* probabilistic —
constrained linear programming: duality theory *see:* Probabilistic —
constrained logic programming
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
constrained minimax problem
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
constrained minimax problems
[65K05, 65K10, 90C06, 90C30, 90C34]
(*see:* **Feasible sequential quadratic programming**)
constrained minimization
[90C26]
(*see:* **Invexity and its applications**)
constrained minimization problem
[65M60, 90C26]
(*see:* **Invexity and its applications**; **Variational inequalities: F. E. approach**)
constrained minimum spanning tree problem *see:* resource- —
constrained nonlinear optimization *see:* Inequality- —
constrained nonlinear programming: KKT necessary optimality conditions *see:* Equality- —
constrained nonlinear programming problem *see:* equality- —
constrained optimization
[65G20, 65G30, 65G40, 65H20, 65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**; **Interval analysis: unconstrained and constrained optimization**)

constrained optimization
[49M37, 65G20, 65G30, 65G40, 65H20, 65K05, 90C30]
(*see:* **Inequality-constrained nonlinear optimization**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**)
constrained optimization *see:* equality- —; general —; Interval analysis: unconstrained and —; nonlinearly —
constrained optimization problem *see:* global —
constrained optimization problems *see:* linearly —
constrained optimum *see:* conditions for a —
constrained partitioning *see:* order —
constrained path problem *see:* impossible pairs —
constrained problems: convexity theory *see:* Probabilistic —
constrained project scheduling *see:* Static resource —
constrained quadratic problem *see:* bound —
constrained stochastic programming *see:* probabilistic —
constrained subgraph problem *see:* degree- —
constrained: unified modeling frameworks *see:* Short-term scheduling, resource —
constrained vehicle routing problem *see:* distance- —
constraint *see:* abstract —; active —; alignment —; arity of a —; ball —; budget —; capacity —; conditional expectation —; consecutive one —; convex quadratic —; ellipsoid —; ellipsoidal —; flow conservation —; hidden —; implicit equality —; implied —; integral —; integral quadratic —; integrated probabilistic —; irredundant —; joint probabilistic —; knapsack —; linear —; linear program with an additional reverse convex —; locality —; marginal —; necessary —; nonlinear —; nonredundant —; probabilistic —; programming under probabilistic —; pseudoquadratic —; redundant —; relatively redundant —; resource —; Slater —; state —; state inequality —; surrogate —; tongue-and-groove —; weakly necessary —; weight of a —
constraint on arc flows *see:* capacity —
constraint-by-constraint method *see:* lexicographic variant of the —
constraint-factor
[90C26]
(*see:* **Reformulation-linearization technique for global optimization**)
constraint graph
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
constraint logic programming
[65G20, 65G30, 65G40, 68T20, 90C10, 90C30]
(*see:* **Interval constraints**; **Modeling languages in optimization: a new paradigm**)
constraint logic programming *see:* modeling language and —
constraint method *see:* ϵ- —; lexicographic variant of the constraint-by- —
constraint on a multiplicative function
[90C26]
(*see:* **Global optimization in multiplicative programming**)
constraint narrowing operator
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
constraint programming
[65G20, 65G30, 65G40, 68T20, 68T99, 90C27, 90C59]
(*see:* **Interval constraints**; **Metaheuristics**)
constraint programming
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
constraint programming *see:* chance —
constraint programming hybrid methods *see:* Mixed integer programming/ —
constraint propagation
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
constraint qualification
[49K27, 49K40, 49M37, 65K05, 65K10, 90C05, 90C25, 90C26, 90C29, 90C30, 90C31, 90C39, 93A13]
(*see:* **Bilevel programming: optimality conditions and duality**; **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Multilevel methods for optimal design**; **Second order optimality conditions for nonlinear optimization**; **Theorems of the alternative and optimization**)
constraint qualification
[49K27, 49K40, 90C26, 90C30, 90C31, 90C39]
(*see:* **Duality for semidefinite programming**; **First order constraint qualifications**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**; **Sensitivity and stability in NLP: continuity and differential stability**)
constraint qualification *see:* basic —; first order —; generalized Slater —; linear independence —; linear independency —; Mangasarian–Fromovitz —; second order —; Slater —
constraint qualification (LICQ) *see:* linear independence —
constraint qualifications
[90C30, 90C31]
(*see:* **Image space approach to optimization**; **Sensitivity and stability in NLP: continuity and differential stability**)
constraint qualifications *see:* First order —; input —; Second order —
constraint region *see:* relaxed —
constraint satisfaction
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
constraint satisfaction *see:* maximum —
constraint satisfaction problem
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
Constraint Satisfaction Problem *see:* binary —; max-r- —; maximum —; numerical —
constraint satisfaction problems
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
constraint satisfaction problems *see:* continuous —
constraint satisfaction: relaxations and upper bounds *see:* Maximum —
constraint satisfaction techniques
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)

constraint satisfaction techniques
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
constraint set
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
constraint set *see:* reduction of a —
constraint solving
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
constraint violating point *see:* index of a —
constraints
[65G20, 65G30, 65G40, 68T20, 90C05, 90C10, 90C20, 90C26, 90C30]
(*see:* **Global optimization using space filling**; **Interval constraints**; **Modeling languages in optimization: a new paradigm**; **Redundancy in nonlinear programs**; **Smooth nonlinear nonconvex optimization**)
constraints
[65K05, 90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Direct global optimization algorithm**; **Modeling difficult optimization problems**)
constraints *see:* abstract —; active —; active inequality —; binary noninterference —; bound —; bounds —; box —; capacity —; consistency —; control —; coupling —; disjoint —; equality —; feasibility of equality —; feasibility of inequality —; flow —; flow bound —; fuzzy —; general linear —; generalized upper bounds —; inactive —; incorporation of biological —; individual probabilistic —; inequality —; infeasibility of inequality —; intercommodity —; Interval —; Lagrange multipliers for nonanticipativity —; Lagrange multipliers for phase —; linearization of —; lower and upper bounds —; mass balance —; mathematical program with affine equilibrium —; mathematical program with equilibrium —; maximum function with dependent —; maximum oil, gas and water capacity —; multistage linking —; nested —; network —; nonanticipativity —; noninterference —; notation for —; odd-set —; optimization under network —; phase —; positive semidefiniteness —; precedence/coupling —; projection —; Quadratic programming with bound —; regular —; semi-assignment —; set-valued —; side —; simplicial —; single fixed cost with capacity —; single fixed cost with no capacity —; slack —; state —; strongly active —; structural —; sub-tour elimination —; submodular —; subtour elimination —; tight —; time window —; tree —; upper and lower well oil rate —; vehicle scheduling problems with time —; violation of —; weighted-sums programs with —
constraints: A piecewise SQP approach *see:* Optimization with equilibrium —
constraints in standard form
[90C60]
(*see:* **Complexity of degeneracy**)
constraints on variables
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: relaxation methods**)
construct *see:* cognitive —
construction *see:* greedy —; network design and schedule —; random —
construction of descent directions
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
construction of a dual problem
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
construction heuristic
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
construction heuristics
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
construction methods
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
construction phase
(*see:* **Maximum cut problem, MAX-CUT**)
construction phase in GRASP
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
construction procedure *see:* arc oriented —; best arc —; best node —; mixed —; mixed VAM —; node oriented —
construction procedures
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
construction procedures
[90B80]
(*see:* **Facilities layout problems**)
constructions *see:* linearly elastic mechanical —
constructive lower bounds
[90B35]
(*see:* **Job-shop scheduling problem**)
constructive methods for solving vehicle routing problems
[90B06]
(*see:* **Vehicle routing**)
constructive nonlinear dynamics *see:* Robust design of dynamic systems by —
consumption *see:* expected power —
consumption of utilities
(*see:* **Planning in the process industry**)
contact *see:* Signorini-Coulomb unilateral frictional —
contact map
(*see:* **Contact map overlap maximization problem, CMO**)
contact map overlap
(*see:* **Contact map overlap maximization problem, CMO**)
Contact map overlap maximization problem, CMO
contact point
[90Cxx]
(*see:* **Discontinuous optimization**)
contact problem with friction *see:* coupled unilateral —
contacts in alpha-helical proteins *see:* Predictive method for interhelical —
containment graph model
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
contaminated information
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]

(*see:* **Information-based complexity and information-based optimization**)
context descriptors
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
context descriptors
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
Conti–Traverso algorithm
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
contingency table
[03B50, 03B52, 03C80, 62F30, 62Gxx, 62H30, 68T27, 90C27]
(*see:* **Assignment methods in clustering**; **Checklist paradigm semantics for fuzzy logics**)
contingent
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
contingent cone
[90C31, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions)
contingent epiderivative
[49K27, 90C29, 90C48]
(*see:* **Set-valued optimization**)
continuation
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
continuation
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
continuation *see:* homotopy —
continuation method
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
continuation method
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
continuation method *see:* homotopy —
continuity *see:* Hölder —; joint —; Lipschitz —; rates of quantitative —
continuity conditions
[34A55, 78A60, 90C30]
(*see:* **Optimal design in nonlinear optics**)
continuity and differential stability *see:* Sensitivity and stability in NLP: —
continuity property of the objective function value
[90C31]
(*see:* **Bounds and solution vector estimates for parametric NLPS**)
continuity, stability, rates of convergence *see:* Stochastic integer programming: —
continuous
[58E05, 90C30]
(*see:* **Planning in the process industry**; **Topology of global optimization**)
continuous *see:* approximate —; exact —; Lipschitz —
Continuous approximations to subdifferentials
(65K05, 90C56)
continuous based heuristics
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
continuous constraint satisfaction problems
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
continuous dependence
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)
continuous and discrete free variables *see:* Generalized geometric programming: mixed —
continuous and discrete time models
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
continuous form *see:* coercive bilinear symmetric —
continuous function *see:* Lipschitz —; locally Lipschitz —; radially —; U- —
continuous functional *see:* absolutely —
continuous functions *see:* Lipschitzian operators in best approximation by bounded or —
continuous global optimization
[90C05]
(*see:* **Continuous global optimization: models, algorithms and software**; **Global optimization in the analysis and management of environmental systems**)
continuous global optimization *see:* mixed discrete- —
Continuous global optimization: applications
(90C05)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Forecasting**; **Global optimization in the analysis and management of environmental systems**; **Global optimization based on statistical models**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Interval global optimization**; **Mixed integer nonlinear programming**; **Topology of global optimization**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Forecasting**; **Global optimization in the analysis and management of environmental systems**; **Global optimization based on statistical models**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Interval global optimization**; **Mixed integer nonlinear programming**; **Topology of global optimization**)
continuous global optimization model
[90C05]
(*see:* **Continuous global optimization: models, algorithms and software**)
Continuous global optimization: models, algorithms and software
(90C05)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: applications**; **Differential equations and**

global optimization; **Direct global optimization algorithm**; **Global optimization in the analysis and management of environmental systems**; **Global optimization based on statistical models**; **Global optimization in batch design under uncertainty**; **Global optimization in binary star astronomy**; **Global optimization in generalized geometric programming**; **Global optimization: interval analysis and balanced interval arithmetic**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Global optimization using space filling**; **Interval global optimization**; **Large scale unconstrained optimization**; **Maximum cut problem, MAX-CUT**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Modeling languages in optimization: a new paradigm**; **Optimization-based visualization**; **Optimization software**; **Smooth nonlinear nonconvex optimization**; **Topology of global optimization**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: applications**; **Convex envelopes in optimization problems**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Global optimization in the analysis and management of environmental systems**; **Global optimization based on statistical models**; **Global optimization in batch design under uncertainty**; **Global optimization in binary star astronomy**; **Global optimization in generalized geometric programming**; **Global optimization of heat exchanger networks**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Global optimization using space filling**; **Interval global optimization**; **Large scale unconstrained optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: heat exchanger network synthesis**; **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Modeling languages in optimization: a new paradigm**; **Optimization software**; **Smooth nonlinear nonconvex optimization**; **Topology of global optimization**)

continuous location
[90B80, 90B85, 90Cxx, 91Axx, 91Bxx]
(*see:* **Facility location with externalities**)

continuous model in OR
[90B80, 90B85]
(*see:* **Warehouse location problem**)

continuous multiple criteria problem
[90C29]
(*see:* **Multiple objective programming support**)

continuous operator *see:* completely —

continuous optimization
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)

continuous optimization problems *see:* Continuous reformulations of discrete- —

continuous piecewise linear function *see:* decomposition of a —

continuous processes *see:* Short-term scheduling of —

continuous programming
[90C26]
(*see:* **Invexity and its applications**)

continuous programming
[90C26]
(*see:* **Invexity and its applications**)

Continuous reformulations of discrete-continuous optimization problems
(90C11, 90C10, 90C33, 90C27)
(*refers to:* **Disjunctive programming**; **Mixed integer programming/constraint programming hybrid methods**; **Order complementarity**)

continuous relaxation
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)

Continuous review inventory models: (*Q, R*) policy
(49-02, 90-02)

continuous review model
[90B50]
(*see:* **Inventory management in supply chains**)

continuous review model
[90B50]
(*see:* **Inventory management in supply chains**)

continuous selection
[49J52, 49Q10, 74G60, 74H99, 74K99, 74Pxx, 90C90]
(*see:* **Quasidifferentiable optimization: stability of dynamic systems**)

continuous selection of functions
[58E05, 90C30]
(*see:* **Topology of global optimization**)

continuous selection operator
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)

continuous-time analog of the dynamic programming equation
[34H05, 49L20, 90C39]
(*see:* **Dynamic programming: continuous-time optimal control**)

continuous-time equivalent of the dynamic programming algorithm
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)

continuous-time formulation
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)

continuous Time Model
(*see:* **Integrated planning and scheduling**)

continuous-time optimal control
[34H05, 49L20, 90C39]
(*see:* **Dynamic programming: continuous-time optimal control**; **Hamilton–Jacobi–Bellman equation**)

continuous-time optimal control *see:* Dynamic programming: —

continuous-time Riccati equation
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)

continuously codifferentiable
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)

continuously codifferentiable *see:* twice —
continuously codifferentiable function
[49J52, 65K99, 70-08, 90C25]
(*see:* **Quasidifferentiable optimization: codifferentiable functions**)
continuously codifferentiable function *see:* twice —
continuously differentiable exact penalty function approach
[90C30]
(*see:* **Large scale trust region problems**)
continuously differentiable function *see:* piecewise —
continuum
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
continuum
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
continuum *see:* true —
contract *see:* energy purchase —
contract algorithm *see:* branch and —
contract-or-patch (COP)
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
contracting matroid elements
[90C09, 90C10]
(*see:* **Matroids**)
contracting measure
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
contraction
[65H10, 65J15, 90C30]
(*see:* **Contraction-mapping**; **Sequential simplex method**)
contraction *see:* k-set- —; matroid —; path —; strict-set- —; weighter sup-norm —
contraction/approximation measure
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
contraction coefficient
[90C30]
(*see:* **Sequential simplex method**)
Contraction-mapping
(65H10, 65J15)
(*referred to in:* **Global optimization methods for systems of nonlinear equations**; **Gröbner bases for polynomial equations**; **Interval analysis: systems of nonlinear equations**; **Nonlinear least squares: Newton-type methods**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
(*refers to:* **Global optimization methods for systems of nonlinear equations**; **Interval analysis: systems of nonlinear equations**; **Nonlinear least squares: Newton-type methods**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
contraction mapping
[49L20, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**)
contraction mapping
[65H10, 65J15]
(*see:* **Contraction-mapping**)
contraction mappings
[49L20, 90C30, 90C39, 90C40, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**; **Dynamic programming: infinite horizon problems, overview**)
contraction matrices *see:* completion to completely positive and —
contraction matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
contraction matrix *see:* partial —
contraction of a matroid
[90C09, 90C10]
(*see:* **Matroids**)
contraction in matroids
[90C09, 90C10]
(*see:* **Oriented matroids**)
contraction method *see:* edge —
contraction operation
[90C35]
(*see:* **Feedback set problems**)
contraction operation
[90C30]
(*see:* **Sequential simplex method**)
contractive operator
[49L20, 90C39]
(*see:* **Dynamic programming: discounted problems**)
contradual transformation
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
contrapositivization
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
control
[49Jxx, 49K20, 49M99, 90C55, 91Axx]
(*see:* **Emergency evacuation, optimization modeling**; **Infinite horizon control and dynamic games**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
control
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
control *see:* closed-loop —; continuous-time optimal —; Discrete-Time Optimal —; Dynamic programming: continuous-time optimal —; Dynamic programming: inventory —; Dynamic programming and Newton's method in unconstrained optimal —; epidemic —; feedback —; ground delay problem in air traffic —; interaction of design and —; interaction of design, synthesis and —; inventory —; MINLP: applications in the interaction of design and —; model predictive —; mu synthesis —; Multi-objective optimization: interaction of design and —; open-loop —; optimal —; parametric optimal —; piecewise constant —; piecewise linear —; pollution —; process —; relaxed —; Resource allocation for epidemic —; Robust —; rounding errors are under —; Suboptimal —; systems theory and —; temperature —; time optimal —; unconstrained optimal —
control applications *see:* Dynamic programming: optimal —

control component
[90C90, 91B28]
(*see:* **Robust optimization**)
control constraints
[90C90, 91B28]
(*see:* **Robust optimization**)
control for drug delivery systems *see:* Model based —
control and dynamic games *see:* Infinite horizon —
control engineering
[93D09]
(*see:* **Robust control**)
control with first order differential equations *see:* Duality in optimal —
control of a flexible arm *see:* Optimal —
control function
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
control-function *see:* admissible pair of trajectory-function and —; linear appearance of —
control functions *see:* asymptotically admissible pair of trajectory and —
control and ground delay programs *see:* air traffic —
control model *see:* logistics —
control pair *see:* admissible trajectory- —
control parameterization
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)
control parametrization
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
control policy *see:* optimal —
control problem *see:* finite-dimensional —; inventory —; mixed integer optimal —; optimal —; relaxed —; singular —; time optimal —
Control problems
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
control problems *see:* discretized optimal —; distributed optimal —; Semi-infinite programming and —; Sequential quadratic programming: interior point methods for distributed optimal —
control restrictions
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
control: schur stability of polytopes of polynomials *see:* Robust —
control state of a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
control synthesis *see:* robust —
control theory
[49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX]
(*see:* **Resource allocation for epidemic control**)
control theory
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
control theory *see:* robust —
control variables
[90C90, 91B28]
(*see:* **Robust optimization**)
control variates
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
control vector
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
control vector iteration
[93-XX]
(*see:* **Boundary condition iteration BCI**)
Control vector iteration CVI
(93-XX)
(*referred to in:* **Boundary condition iteration BCI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Boundary condition iteration BCI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
controllability
[49M37, 90C11, 93-XX]
(*see:* **MINLP: applications in the interaction of design and control**; **Optimal control of a flexible arm**)
controllability *see:* integration of dynamic considerations and —; minimum norm —
controllability measure
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)
controlled recharge facilities
[90C30, 90C35]
(*see:* **Optimization in water resources**)

controlled selection
[90C35]
(*see:* **Multi-index transportation problems**)
controller *see:* feasible gradient —; nonfeasible gradient —
controllers via parametric programming *see:* Design of robust model-based —
controls *see:* suboptimal trajectories and —; unbounded —
controls and non standard methods *see:* unbounded —
convention *see:* extensionality —
conventional
[90C35]
(*see:* **Multicommodity flow problems**)
convergence
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
convergence *see:* $\mathcal{P}$ —; asymptotic —; discrete —; discrete Mosco —; discrete Painlevé–Kuratowski —; f-attentive —; finite ϵ- —; global —; linear —; polynomial time —; premature —; Q-quadratic —; Q-superlinear —; quadratic —; rate of —; Stochastic integer programming: continuity, stability, rates of —; superlinear —; weak —; weak discrete —
convergence condition *see:* superlinear —
convergence of GRASP *see:* global —
convergence of the overall flowsheet
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
convergence, and Powell's conjecture *see:* Rosen's method, global —
convergence of PPA
[90C30]
(*see:* **Relaxation in projection methods**)
convergence of probability measures *see:* weak —
convergence problem for the Rosen method *see:* global —
convergence rate
[90C06, 93-XX]
(*see:* **Boundary condition iteration BCI**; **Large scale unconstrained optimization**)
convergence rate
[90C30]
(*see:* **Frank–Wolfe algorithm**)
convergence rate *see:* geometric —; local —; r-linear —
convergence rates *see:* asymptotic —
convergence tests *see:* feasibility —; value —
convergence theorem
[60G35, 65K05]
(*see:* **Differential equations and global optimization**)
convergence theorem *see:* asynchronous —; local quadratic —; monotone —
convergence and turnpike theory *see:* Statistical —
convergent
[90C25, 90C26]
(*see:* **Decomposition in global optimization**)
convergent *see:* globally —
convergent algorithm
[90C26]
(*see:* **Cutting plane methods for global optimization**)
convergent algorithm *see:* globally —
convergent homotopies *see:* probability-one globally —
convergent homotopy methods *see:* Globally —
convergent probability-one homotopy algorithm *see:* globally —
convergent rate *see:* superlinear —
converges
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
converse relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
converting *see:* paper —
convex
[41A30, 47A99, 49K05, 49K10, 49K15, 49K20, 49M37, 65K10, 90C10, 90C11, 90C25, 90C26, 90C27, 90C30, 90C31, 90C33, 90C35]
(*see:* **αBB algorithm**; **Continuous reformulations of discrete-continuous optimization problems**; **Duality in optimal control with first order differential equations**; **Generalized monotone single valued maps**; **Global optimization: functional forms**; **L-convex functions and M-convex functions**; **Lipschitzian operators in best approximation by bounded or continuous functions**; **Robust global optimization**; **Successive quadratic programming: solution by active sets and interior point methods**)
convex
[90C05, 90C11, 90C15, 90C25, 90C30]
(*see:* **Krein–Milman theorem**; **Stochastic programming with simple integer recourse**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
convex *see:* η- —; invariant —; m- —; strongly —
convex-along-rays functions on topological vector spaces *see:* Increasing and —
convex analysis
[26B25, 26E25, 46A22, 49J35, 49J40, 49J52, 49Q10, 49S05, 54D05, 54H25, 55M20, 65K99, 70-08, 70-XX, 74G99, 74H99, 74K99, 74Pxx, 80-XX, 90C25, 90C33, 90C99, 91A05]
(*see:* **Hemivariational inequalities: applications in mechanics**; **Minimax theorems**; **Nonconvex energy functions: hemivariational inequalities**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: codifferentiable functions**)
convex analysis
[32B15, 51E15, 51N20]
(*see:* **Affine sets and functions**)
convex analysis *see:* abstract —; discrete —
convex bipartite graph
[90C35]
(*see:* **Feedback set problems**)
convex combination of the extreme points
[90C30]
(*see:* **Simplicial decomposition**)
convex combinations
[49M07, 49M10, 65K05, 68Q99, 90C06]
(*see:* **Branch and price: Integer programming with column generation**; **Performance profiles of conjugate-gradient algorithms for unconstrained optimization**)

convex combinations
[90C30]
(*see:* **Simplicial decomposition**)
convex combinatorial optimization
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
convex composite programming
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
convex-concave case
[90C15]
(*see:* **Stochastic programs with recourse: upper bounds**)
convex and concave regression
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
convex cone
[90C30]
(*see:* **Duality for semidefinite programming**)
convex cone *see:* pointed —; pointed closed —
convex cones
[90C22, 90C25]
(*see:* **Copositive programming**)
convex constraint *see:* linear program with an additional reverse —
convex decreasing
[65D18, 90B85, 90C26]
(*see:* **Global optimization in location problems**)
Convex discrete optimization
(05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C)
(*referred to in:* **Adaptive convexification in semi-infinite optimization**)
Convex envelopes in optimization problems
(90C26)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: models, algorithms and software**; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Lipschitzian operators in best approximation by bounded or continuous functions**; **MINLP: global optimization with αBB**)
(*refers to:* **αBB algorithm**; **Global optimization in generalized geometric programming**; **MINLP: global optimization with αBB**)
convex feasibility problem
[47H05, 65J15, 90C25, 90C30, 90C55]
(*see:* **Fejér monotonicity in convex optimization**; **Relaxation in projection methods**)
convex function
[49J52, 90C26, 90C30, 90C31, 90C39]
(*see:* **Nondifferentiable optimization: subgradient optimization methods**; **Second order optimality conditions for nonlinear optimization**; **Sensitivity and stability in NLP: approximation**)
convex function
[49J52, 65K05, 90C30, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Frank–Wolfe algorithm**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**)
convex function *see:* abstract —; difference —; geodesic —; H- —; K- —; L- —; M- —; program of minimizing a generalized —; strictly —; uniformly —
convex functions *see:* difference of —; Fenchel-type duality for M- and L- —; h- —; L-convex functions and M- —; product of —
convex functions and M-convex functions *see:* L- —
convex global underestimation
[65K05, 90C26]
(*see:* **Molecular structure determination: convex global underestimation**)
convex global underestimation *see:* Molecular structure determination: —
convex global underestimator
[65K05, 90C26]
(*see:* **Molecular structure determination: convex global underestimation**)
convex hull
[05A, 05C60, 05C69, 15A, 37B25, 41A30, 47A99, 51M, 52A, 52B, 52C, 62H, 65K10, 68Q, 68R, 68U, 68W, 90B, 90B80, 90B85, 90C, 90C05, 90C06, 90C08, 90C09, 90C10, 90C11, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Convex discrete optimization**; **Disjunctive programming**; **Integer programming: cutting plane algorithms**; **Lipschitzian operators in best approximation by bounded or continuous functions**; **Replicator dynamics in combinatorial optimization**; **Single facility location: circle covering problem**; **Voronoi diagrams in facility location**)
convex hull
[90B80, 90C05, 90C09, 90C10, 90C11, 90C15, 90C27, 90C30]
(*see:* **Carathéodory theorem**; **Disjunctive programming**; **Frank–Wolfe algorithm**; **Krein–Milman theorem**; **Simplicial decomposition**; **Stochastic programming with simple integer recourse**; **Voronoi diagrams in facility location**)
convex hull *see:* lower —
Convex hull disjunctions
(*see:* **Logic-based outer approximation**)
convex hull problem
[52B12, 68Q25]
(*see:* **Fourier–Motzkin elimination method**)
convex inequalities
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
convex inequality *see:* reverse —
convex inequality systems
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
convex integer programming
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
convex integer transportation problem
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)

convex intersection problem
[90C30]
(*see:* **Relaxation in projection methods**)
convex-like
[90C26]
(*see:* **Invexity and its applications**)
convex-like
[90C26]
(*see:* **Invexity and its applications**)
convex-like function
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
convex-like function pair
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
convex-like systems
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
convex map *see:* cone- —
convex max-function
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
Convex max-functions
(49K35, 49M27, 65K10, 90C25)
(*referred to in:* **Affine sets and functions**; **Lagrangian multipliers methods for convex programming**)
(*refers to:* **Lagrangian multipliers methods for convex programming**; **Successive quadratic programming**)
convex MINLP
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
convex minorant *see:* greatest —
convex model
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
convex moment problem
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
convex moment problem
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
convex moment problem *see:* solution of the —
convex multiplicative function *see:* program of minimizing a —
convex multiplicative functions *see:* sum of —
convex multiplicative program
[90C26]
(*see:* **Global optimization in multiplicative programming**)
convex NDO
[46N10, 90-00, 90C47]
(*see:* **Nondifferentiable optimization**)
convex and nonconvex programming problems
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
convex objective function *see:* separable —
convex optimization
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
convex optimization
[15A15, 49-XX, 49K35, 49M27, 65K10, 90-XX, 90C25, 90C30, 90C55, 90C90, 93-XX]
(*see:* **Convex max-functions**; **Duality theory: monoduality in convex optimization**; **Lagrangian multipliers methods for convex programming**; **Semidefinite programming and determinant maximization**)
convex optimization *see:* Duality theory: monoduality in —; Fejér monotonicity in —; multi-objective —; nondifferentiable —; Reverse —
convex optimization problem
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
convex parametric programming
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
convex piecewise linearization in facility location problems with staircase costs
[90B80, 90C11]
(*see:* **Facility location with staircase costs**)
convex polyhedral set
[90C05, 90C15]
(*see:* **Probabilistic constrained linear programming: duality theory**)
convex polytope
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
convex polytope
[52B11, 52B45, 52B55, 90B85]
(*see:* **Multifacility and restricted location problems**; **Volume computation for polytopes: strategies and performances**)
convex problem
[90C25, 90C30, 90C31]
(*see:* **Lagrangian multipliers methods for convex programming**; **Sensitivity and stability in NLP: continuity and differential stability**)
convex problem *see:* jointly —
convex problems
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
convex problems *see:* partly —
convex program *see:* nondifferentiable —; partly —; semidefinite program as conic —
convex programming
[90C05, 90C06, 90C22, 90C25, 90C30, 90C51]
(*see:* **Interior point methods for semidefinite programming**; **Saddle point theory and optimality conditions**)
convex programming
[47H05, 65J15, 90C05, 90C25, 90C30, 90C55]
(*see:* **Duality for semidefinite programming**; **Fejér monotonicity in convex optimization**; **Young programming**)
convex programming *see:* differentiable —; fundamental property in —; Lagrangian multipliers methods for —; reverse —
convex programming problem
[60G35, 65K05, 90C26, 90C39]
(*see:* **Differential equations and global optimization**; **Second order optimality conditions for nonlinear optimization**)

convex programs *see:* conic —; partly —; reverse —
convex quadratic constraint
[90C60]
(*see:* **Complexity theory: quadratic programming**)
convex quadratic function
[90C60]
(*see:* **Complexity theory: quadratic programming**)
convex quadratic knapsack problem
[90C60]
(*see:* **Complexity theory: quadratic programming**)
convex quadratic optimization
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
convex quadratic program
[90B85, 90C27]
(*see:* **Single facility location: circle covering problem**)
convex quadratic programming
[90C30]
(*see:* **Lagrangian duality: BASICS**)
convex regression problem
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
convex relaxation problem
[65H10, 90C26, 90C30]
(*see:* **Global optimization methods for systems of nonlinear equations**)
convex relaxations
[90C26, 90C90]
(*see:* **Global optimization of heat exchanger networks**)
convex semidefinite programming problem
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
convex set
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**)
convex set *see:* geodesic —; L- —; M- —; reverse —
convex sets *see:* differences of —
Convex-simplex algorithm
(90C30)
(*referred to in:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Order complementarity**; **Parametric linear programming: cost simplex algorithm**; **Principal pivoting methods for linear complementarity problems**; **Sequential simplex method**; **Topological methods in complementarity theory**)
(*refers to:* **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Parametric linear programming: cost simplex algorithm**; **Sequential simplex method**)
convex-simplex algorithm
[90C30]
(*see:* **Convex-simplex algorithm**)
convex-simplex algorithm
[90C30]
(*see:* **Convex-simplex algorithm**)
convex SIP
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
convex SQP
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
convex subdifferential
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
convex transformation
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
convex transformation
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
convex underestimator
[90C11, 90C26]
(*see:* **Convex envelopes in optimization problems**; **MINLP: branch and bound methods**)
convex underestimator
[90C26]
(*see:* **Convex envelopes in optimization problems**)
convex underestimators *see:* Global optimization: tight —
convex variational inequality for an elastostatic problem involving QD-superpotentials
[49J40, 49M05, 49S05, 74G99, 74H99, 74Pxx]
(*see:* **Quasidifferentiable optimization: variational formulations**)
convexifiable
[25A15, 34A05, 90C25, 90C26, 90C30, 90C31]
(*see:* **Convexifiable functions, characterization of**; **Invexity and its applications**)
convexifiable
[90C26]
(*see:* **Invexity and its applications**)
Convexifiable Function *see:* integral Mean-Value for Composite —
Convexifiable functions, characterization of
(90C25, 90C26, 90C30, 90C31, 25A15, 34A05)
convexifiable program *see:* sequentially —
convexification
[25A15, 34A05, 90C25, 90C26, 90C30, 90C31]
(*see:* **Convexifiable functions, characterization of**)
convexification
[90C27]
(*see:* **Time-dependent traveling salesman problem**)
convexification parameter
[65K05, 90C26, 90C33, 90C34]
(*see:* **Adaptive convexification in semi-infinite optimization**)
convexification/relaxation strategy
[65K05, 90C11, 90C26]
(*see:* **MINLP: global optimization with αBB**)
convexification in semi-infinite optimization *see:* Adaptive —
Convexification Technique *see:* reformulation-Linearization/ —
convexification techniques *see:* reformulation-linearization/ —

convexifier
[25A15, 34A05, 90C25, 90C26, 90C30, 90C31]
(*see:* **Convexifiable functions, characterization of**)
convexity
[90C26, 90C30, 90C33]
(*see:* **Generalized monotone single valued maps**; **Implicit lagrangian**)
convexity
[28-XX, 49-XX, 49M37, 60-XX, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**; **Frank–Wolfe algorithm**; **General moment optimization problems**)
convexity *see:* abstract —; discrete midpoint —; generalized —; geodesic —; K- —; L- —; M- —
convexity cut
[90C26]
(*see:* **Cutting plane methods for global optimization**)
convexity property of the objective function value
[90C31]
(*see:* **Bounds and solution vector estimates for parametric NLPS**)
convexity property of the solution space
[90C31]
(*see:* **Bounds and solution vector estimates for parametric NLPS**)
convexity theory *see:* Probabilistic constrained problems: —
convexized filled function *see:* globally —
Cook–Levin theorem
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
cook's theorem
[03B50, 68T15, 68T30, 90C60]
(*see:* **Computational complexity theory**; **Finite complete systems of many-valued logic algebras**)
cooling schedule
[65K05, 90C30]
(*see:* **Random search methods**)
cooperation *see:* region of —
cooperation minimization algorithms *see:* supervisor and searcher —
cooperative case of a two-person game
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)
cooperative equilibrium
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
Cooperative equilibrium
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
cooperative game
[90C27, 90C60, 91A12]
(*see:* **Combinatorial optimization games**)
cooperative game
[90C27, 90C60, 91A12]
(*see:* **Combinatorial optimization games**)
cooperative solution
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
coordinate
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
coordinate-aligned ellipsoid
[37A35, 90C05]
(*see:* **Potential reduction methods for linear programming**)
coordinate descent method
[90C30]
(*see:* **Cost approximation algorithms**)
coordinate direction
[90C26, 90C90]
(*see:* **Global optimization: hit and run methods**)
coordinate method *see:* Cyclic —
coordinate search *see:* cyclic —
coordinate system *see:* curvilinear —; moving —
coordinate transformation
[90C30]
(*see:* **Suboptimal control**)
coordinates
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
coordinates *see:* axes of —; Cartesian —; internal —; kth order form of —
coordinatewise increasing function
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
coordinatewise increasing utility function
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
coordination *see:* decomposition/ —
coordination method *see:* goal —; model —
coordination step
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
(COP) *see:* contract-or-patch —
copolyblock algorithm
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
(copolyblock) algorithm *see:* revised reverse polyblock —
copositive
[05C15, 05C17, 05C35, 05C69, 65K05, 90C20, 90C22, 90C25, 90C35]
(*see:* **Copositive programming**; **Lovász number**; **Quadratic programming with bound constraints**)
copositive matrix
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**; **Standard quadratic optimization problems: theory**)
copositive matrix *see:* strictly —
Copositive optimization
(90C20, 90C22, 90C26)
Copositive programming
(90C25, 90C22)
(*referred to in:* **Lovász number**)
copositive programming
[90C22, 90C25]
(*see:* **Copositive programming**)
copositivity
[90C20]
(*see:* **Standard quadratic optimization problems:**

algorithms; **Standard quadratic optimization problems: theory**)
copulas
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
core *see:* multi- —
Corley–Moon algorithm
[90C31, 90C39]
(*see:* **Multiple objective dynamic programming**)
corner rule *see:* North–West —
corner solution
[90C05]
(*see:* **Extension of the fundamental theorem of linear programming**)
cornered
[90C05]
(*see:* **Extension of the fundamental theorem of linear programming**)
corrected seminormal equation
[65Fxx]
(*see:* **Least squares problems**)
correcting methods *see:* label —
corrective
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
corrective action
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)
corrector
[90C05]
(*see:* **Linear programming: interior point methods**)
corrector *see:* predictor- —
corrector algorithm *see:* predictor- —
corridor method
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
cost
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
cost *see:* decomposable —; differential —; Hamiltonian path —; linear platform —; mean-weight —; minimizing network —; path —; piecewise linear arc —; production realizing with minimal social —; reduced —; setup —; staircase —; transportation —; unbounded —; variable —
cost approximation
[90C39]
(*see:* **Neuro-dynamic programming**)
Cost approximation algorithms
(90C30)
(*referred to in:* **Dynamic traffic networks**)
(*refers to:* **Dynamic traffic networks**; **Frank–Wolfe algorithm**)
cost of an arc in a network
[90C35]
(*see:* **Minimum cost flow problem**)
cost with capacity constraints *see:* single fixed —
cost coefficients *see:* sensitivity analysis with respect to changes in —
cost of a directed cycle
[90C35]
(*see:* **Minimum cost flow problem**)
cost fixing *see:* reduced —
cost flow problem *see:* minimum —
cost function
[90B10, 90C26, 90C30, 90C35, 93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**; **Nonconvex network flow problems**)
cost function *see:* regular —; regular link —; sawtooth arc —; staircase —; staircase arc —; total —
cost functional
[49J20, 49J52]
(*see:* **Shape optimization**)
cost functions in integer programming
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
cost index
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
cost infinite horizon problem *see:* total —
cost network flow *see:* minimum —
cost network flow problem *see:* minimum —; piecewise linear minimum —
cost with no capacity constraints *see:* single fixed —
cost per stage *see:* average —; discounted problem with bounded —
cost per stage problem *see:* average —
cost per stage problems *see:* average —; Dynamic programming: average —
cost row
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
cost scaling
[90C30, 90C35]
(*see:* **Auction algorithms**)
cost simplex algorithm *see:* Parametric linear programming: —
cost structure
[90B80, 90B85]
(*see:* **Warehouse location problem**)
cost terms
(*see:* **Planning in the process industry**)
cost/time *see:* minimization of —
cost-to-go
[49L20, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**)
cost-to-time ratio
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
cost-to-time ratio cycle *see:* minimum —
cost vector *see:* differential —; generic —
cost vectors *see:* equivalent —
COSTADE
[90C26, 90C29]
(*see:* **Optimal design of composite structures**)
costs *see:* communication —; convex piecewise linearization in facility location problems with staircase —; detention —; Facility location with staircase —; heuristics of facility location problems with staircase —; linearization in facility location problems with staircase —; reduction to finite —; solution of facility location problems with staircase —
Coulomb unilateral frictional contact *see:* Signorini- —

countability
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
countable class
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
countable set D
[49L99]
(*see:* **Dynamic programming: average cost per stage problems**)
counterpart *see:* robust —
counterpart method *see:* stochastic —
coupled fixed point
[90C33]
(*see:* **Order complementarity**)
coupled HMM
(*see:* **Bayesian networks**)
coupled unilateral contact problem with friction
[49J40, 49Q10, 70-08, 74K99, 74Pxx]
(*see:* **Quasivariational inequalities**)
coupling *see:* bandwidth of interdisciplinary —; model/optimizer —
coupling constraints
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
coupling constraints *see:* precedence/ —
Courant penalty function
[90C30]
(*see:* **Image space approach to optimization**)
Cournot equilibrium *see:* Stackelberg–Nash– —
Cournot–Nash equilibrium *see:* spatial —
Cournot–Nash oligopolistic equilibrium
[65K10, 90C31]
(*see:* **Sensitivity analysis of variational inequality problems**)
Cournot–Nash oligopolistic equilibrium model
[90C31, 90C33]
(*see:* **Sensitivity analysis of complementarity problems**)
covariance matrix estimation
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
covector
[90C09, 90C10]
(*see:* **Oriented matroids**)
cover
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
COVER *see:* minimum weighted vertex —; node —; universal —; VERTEX —
cover the extremal set
(*see:* **Planning in the process industry**)
Cover Problem *see:* minimum Vertex —
coverage location problem *see:* maximum —
covering, packing and partitioning problems *see:* Set —
covering problem
[90C35]
(*see:* **Feedback set problems**)
covering problem *see:* node —; set —; Single facility location: circle —
covering problem on a network
[90B10, 90B80, 90C35]
(*see:* **Network location: covering problems**)
covering problems *see:* Network location: —
covering relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
covering subset
[90C09, 90C10]
(*see:* **Matroids**)
covers all edge-directions of P
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
CP *see:* p- —
CPP
[90B20]
(*see:* **General routing problem**)
CQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
CQ *see:* Abadie —; asymptotic —; First order —; Gollan —; Kuhn–Tucker —; linear independence —; Mangasarian–Fromovitz —; Robinson —; second order —; Strong Slater —; Weak Slater —
CR
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
Craig algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
Craig conjugate gradient type algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
Crane Problem (SCP) *see:* stacker —
CRCW PRAM
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
credence *see:* degrees of —
Credit rating and optimization methods
(91B28 90C90 90C05 90C20 90C30)
(*referred to in:* **Beam selection in radiotherapy treatment design**)
crew deadheading
(*see:* **Railroad crew scheduling**)
crew district *see:* double-ended —; single-ended —
crew districts
(*see:* **Railroad crew scheduling**)
crew pairing
(*see:* **Railroad crew scheduling**)
crew pools
(*see:* **Railroad crew scheduling**)
CREW PRAM
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
crew rostering
(*see:* **Railroad crew scheduling**)

crew scheduling
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
crew scheduling
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**; **Railroad crew scheduling**)
crew scheduling *see:* airline —; Railroad —
crew-scheduling problem
[90C10, 90C11, 90C27, 90C57]
(*see:* **Set covering, packing and partitioning problems**)
crew types
(*see:* **Railroad crew scheduling**)
crisp relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
crisp relations *see:* special properties of —
criss-cross
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
criss-cross method
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
criss-cross method
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
criss-cross method *see:* least-index —; Terlaky —; Ziont —
Criss-cross pivoting rules
(90C05, 90C33, 90C20, 05B35, 65K05)
(*referred to in:* **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Linear programming**; **Linear programming: Klee–Minty examples**; **Pivoting algorithms for linear programming generating two paths**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**; **Simplicial pivoting algorithms for integer programming**)
(*refers to:* **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Linear complementarity problem**; **Linear programming**; **Pivoting algorithms for linear programming generating two paths**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**; **Simplicial pivoting algorithms for integer programming**)
criteria
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
criteria *see:* aspiration —; Decision support systems with multiple —; Dykstra's algorithm and robust stopping —
criteria decision making *see:* multiple —
criteria design problem *see:* multiple —
criteria evaluation *see:* multiple —
criteria for multiphase chemical equilibrium *see:* Optimality —
criteria problem *see:* continuous multiple —; discrete multiple —
criteria problems *see:* multi- —
criterion *see:* Akaike information —; dominance —; fuzzy —; infeasibility —; integrality —; k-means —; least squares —; measurable —; Metropolis —; minimum unfeasibility —; objective —; optimality —; ordinal —; probabilistic —; reaction tangent-plane —; scale invariance —; sector stability —; stopping —; tangent-plane —; termination —; test nonmonotone Armijo-like —
criterion problem in OR *see:* single- —
criterion space
[90C29]
(*see:* **Multiple objective programming support**)
criterion uncapacitated static multifacility *see:* discrete single-commodity single- —
critical
[90C05, 90C11, 90C25, 90C30, 90C31, 90C34]
(*see:* **Parametric mixed integer nonlinear optimization**; **Semi-infinite programming: discretization methods**)
critical arcs
(*see:* **Emergency evacuation, optimization modeling**)
critical column
[90C05, 90C06]
(*see:* **Selfdual parametric method for linear programs**)
critical cone
[90C22, 90C25, 90C30, 90C31, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**; **Semidefinite programming: optimality conditions and stability**)
critical cone *see:* z- —
critical direction
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
critical direction *see:* high-order —; high-regular —
critical directions *see:* cone of —
critical interval
[90C11, 90C31]
(*see:* **Parametric mixed integer nonlinear optimization**)
critical path
[90B35]
(*see:* **Job-shop scheduling problem**)
critical point
[49-XX, 49J52, 58E05, 90-XX, 90C30, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**; **Hemivariational inequalities: eigenvalue problems**; **Topology of global optimization**)
critical point *see:* ∂^+- —; generalized —; nondegenerate —
critical point of an energy functional *see:* generalized —
critical point set *see:* generalized —
critical point theory
[49J52]
(*see:* **Hemivariational inequalities: eigenvalue problems**)
critical points
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
critical points *see:* nondegenerate —
critical region
[90C05, 90C31]
(*see:* **Multiparametric linear programming**; **Parametric linear programming: cost simplex algorithm**)
critical region
[90C05, 90C31]
(*see:* **Multiparametric linear programming**; **Parametric linear programming: cost simplex algorithm**)
critical regions
[90C11, 90C31]
(*see:* **Multiparametric mixed integer linear programming**)

critical regions *see:* neighboring —
critical row
[90C05, 90C06]
(*see:* **Selfdual parametric method for linear programs**)
critical value
[90C05, 90C06]
(*see:* **Selfdual parametric method for linear programs**)
cross *see:* criss- —
cross decomposition
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
cross decomposition *see:* generalized —; mean value —; MINLP: generalized —
cross decomposition algorithm
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
cross-entropy
[62F10, 90C25, 94A17]
(*see:* **Entropy optimization: parameter estimation**; **Entropy optimization: shannon measure of entropy and its properties**)
cross-entropy
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
cross-entropy *see:* axiomatic derivation of —; axiomatic derivation of the principle of minimum —; Kullback–Leibler —; Kullback–Leibler measure of —; principle of minimum —
cross-entropy principle *see:* minimum —
cross method *see:* criss- —; least-index criss- —; Terlaky criss- —; Ziont criss- —
cross pivoting rules *see:* Criss- —
cross polytope
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
cross-sectional shapes *see:* beam —
cross-validation
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
crossing *see:* edge —; non- —
crossing minimization
[90C35]
(*see:* **Optimization in leveled graphs**)
crossing minimization *see:* k-level —; leveled —
crossing number
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
crossover
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90, 92B05]
(*see:* **Broadcast scheduling problem**; **Genetic algorithms**; **Traveling salesman problem**)
crossover
[92B05]
(*see:* **Genetic algorithms**)
crossover (EAX) *see:* edge assembly —
CRPM
[90C27, 90C60, 91A12]
(*see:* **Combinatorial optimization games**)
crystal structures *see:* prediction of —
crystal X-ray diffraction data *see:* Optimization techniques for phase retrieval based on single- —
crystallography: Shake and bake approach *see:* Phase problem in X-ray —
CSA
[90C30]
(*see:* **Convex-simplex algorithm**)
CSC
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
csd
[03B52, 03E72, 47S40, 62G07, 62G30, 65K05, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Isotonic regression problems**)
Csiszar α-divergence
[90C05, 90C25]
(*see:* **Young programming**)
CSP
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
CSP *see:* mAX- —; max-r- —
CSPs *see:* binary —
CST
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
cube connected cycle *see:* k-dimensional —
cuboctahedron
[90C35]
(*see:* **Optimization in leveled graphs**)
cumulative
(*see:* **Mixed integer programming/constraint programming hybrid methods**)
cumulative sum diagram
[41A30, 62G07, 62G30, 62J02, 65K05, 90C26]
(*see:* **Isotonic regression problems**; **Regression by special functions: algorithms and complexity**)
cures of dimensionality
[90C05]
(*see:* **Continuous global optimization: models, algorithms and software**)
curse of dimensionality
[65K05, 65T40, 68Q05, 68Q10, 68Q25, 90B36, 90C05, 90C25, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**; **Information-based complexity and information-based optimization**; **Stochastic optimal stopping: numerical methods**; **Stochastic scheduling**)
curse of dimensionality
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26, 90C34]
(*see:* **Information-based complexity and information-based optimization**; **Semi-infinite programming: methods for linear problems**)
curvature
[90C22, 90C25, 90C31]

(*see:* **Semidefinite programming: optimality conditions and stability**)
curvature *see:* direction of negative —; negative —
curve *see:* feasible high-order approximating —; high-order approximating —; interest rate yield —; load —; mobilization —; parabolic —; Peano —; space filling —; switching —; tangent high-order approximating —
curve approach *see:* parabolic —
curve fitting
[90C26, 90C30]
(*see:* **Forecasting**)
curve fitting *see:* subjective —
curve fitting and extrapolation *see:* subjective —
curves *see:* approximation of space filling —
curvilinear coordinate system
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
curvilinear line search
[90C06]
(*see:* **Large scale unconstrained optimization**)
customer
[90B80, 90B85]
(*see:* **Warehouse location problem**)
customer *see:* weight of a —
cut
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**)
cut *see:* branch and —; branch and price and —; capacity of a —; Chvátal–Gomory —; clique- —; concavity —; convexity —; feasibility —; flow across an s—t- —; global —; integer —; intersection —; knapsack —; lift-and-project —; lifting —; local —; max- —; maximum —; Maximum cut problem, MAX- —; maximum mean —; maximum mean-weight —; minimal —; minimum —; mixed integer rounding —; nonlinear —; odd-hole- —; optimality —; s—t- —; valid —
cut algorithm *see:* Jünger–Mutzel branch and —
cut algorithms *see:* Integer programming: branch and —; Stable set problem: branch —
cut-and-branch
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**)
cut conditions
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
cut of a fuzzy relation *see:* α- —
cut generation
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**; **Planning in the process industry**)
cut-improvement *see:* dual —; primal —
Cut (MC) *see:* max —
cut principle *see:* disjunctive —
cut problem *see:* minimum —
cut problem, MAX-CUT *see:* Maximum —
cut procedure *see:* branch and —
cut theorem *see:* max-flow min- —
cuts *see:* feasibility —; Fenchel —; lift-and-project —; nondominated —; parallel —; pool of —; quotient —; reduction —; value —
cutting angle method
[90C26]
(*see:* **Global optimization: envelope representation**)
cutting angle method
[90C26]
(*see:* **Global optimization: envelope representation**)
cutting angle method *see:* Global optimization: —
cutting pattern
[90B90, 90C59]
(*see:* **Cutting-stock problem**)
cutting patterns
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
cutting plane *see:* Chvátal–Gomory —; extended —; generalized —; strong —; trade-off —
cutting plane algorithm
[49M37, 90C08, 90C11, 90C27, 90C29, 90C57, 90C59, 90C90]
(*see:* **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Quadratic assignment problem**)
cutting plane algorithm
[90C06, 90C15]
(*see:* **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Stochastic linear programming: decomposition and cutting planes**)
cutting plane algorithm *see:* Extended —; Gomory —; Sequential —
cutting plane algorithms *see:* Integer programming: —
cutting plane algorithms for stochastic linear programming problems *see:* Stabilization of —
cutting plane approach
[90C25]
(*see:* **Concave programming**)
cutting plane approach *see:* Mixed-integer nonlinear optimization: A disjunctive —
cutting plane approaches
[90C10, 90C11, 90C27, 90C57]
(*see:* **Set covering, packing and partitioning problems**)
cutting plane coefficients *see:* statistical representation of —
cutting plane method
[46N10, 49J40, 49J52, 65K05, 90-00, 90C05, 90C10, 90C11, 90C25, 90C27, 90C30, 90C34, 90C47, 90C57]
(*see:* **Integer programming**; **Nondifferentiable optimization**; **Semi-infinite programming: discretization methods**; **Solving hemivariational inequalities by nonsmooth optimization methods**)
cutting plane method
[90C26]
(*see:* **Cutting plane methods for global optimization**)
cutting plane method *see:* analytic center —; extended —; generalized —; Kelley —; Kelley's classical —
cutting plane methods
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
cutting plane methods *see:* Nondifferentiable optimization: —; regularization of deterministic —
Cutting plane methods for global optimization
(90C26)
cutting plane model
[49J40, 49J52, 65K05, 90C30]

(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
cutting planes
[03B05, 49M37, 68P10, 68Q25, 68R05, 68T15, 68T20, 90-XX, 90C05, 90C06, 90C08, 90C09, 90C10, 90C11, 90C27, 94C10]
(*see:* **Integer programming: cutting plane algorithms**; **Maximum satisfiability problem**; **Mixed integer nonlinear programming**; **Survivable networks**)
cutting planes
[49M20, 90-08, 90C05, 90C06, 90C08, 90C09, 90C10, 90C11, 90C25]
(*see:* **Disjunctive programming**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Nondifferentiable optimization: cutting plane methods**)
cutting planes *see:* polyhedral —; Stochastic linear programming: decomposition and —
cutting stock
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
Cutting-stock problem
(90B90, 90C59)
(*refers to:* **Integer programming**)
cutting-stock problem
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68Q99, 68R, 68U, 68W, 90B, 90B50, 90C, 90C10, 90C11, 90C27, 90C57]
(*see:* **Branch and price: Integer programming with column generation**; **Convex discrete optimization**; **Optimization and decision support systems**; **Set covering, packing and partitioning problems**)
cutting-stock problem
[90B50, 90B90, 90C59]
(*see:* **Cutting-stock problem**; **Optimization and decision support systems**)
cutworthiness
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
cutworthy property
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
CVI *see:* Control vector iteration —
CVRP
[90B06]
(*see:* **Vehicle routing**)
cycle
[90C35]
(*see:* **Minimum cost flow problem**)
cycle *see:* cost of a directed —; fundamental —; k-dimensional cube connected —; maximum profit-to-time ratio —; minimum cost-to-time ratio —; minimum mean —; mixed —
cycle-canceling algorithm
[90C35]
(*see:* **Minimum cost flow problem**)
cycle-canceling algorithm
[90C35]
(*see:* **Minimum cost flow problem**)
cycle of a digraph *see:* directed —
cycle factor
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
cycle in a graph
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
cycle problem *see:* Hamiltonian —; hitting —
cycle time
[90C26]
(*see:* **Global optimization in batch design under uncertainty**)
cycles *see:* negative —
Cyclic coordinate method
(90C30)
(*referred to in:* **Powell method**; **Rosenbrock method**; **Sequential simplex method**)
(*refers to:* **Powell method**; **Rosenbrock method**; **Sequential simplex method**)
cyclic coordinate search
[90C30]
(*see:* **Cyclic coordinate method**)
cyclic rule
[90C30]
(*see:* **Cost approximation algorithms**)
cyclic shift function
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
cyclically reducible graph
[90C35]
(*see:* **Feedback set problems**)
cycling
[90C05, 90C10, 90C60]
(*see:* **Complexity of degeneracy**; **Simplicial pivoting algorithms for integer programming**)
cycling
[05B35, 65K05, 90C05, 90C20, 90C33, 90C60]
(*see:* **Complexity of degeneracy**; **Criss-cross pivoting rules**)
cycling *see:* nondegenerate —
cycling algorithm
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
cycling procedure *see:* anti- —
cZP
[74A40, 90C26]
(*see:* **Shape selective zeolite separation and catalysis: optimization methods**)

D

D *see:* countable set —
D_{DV} *see:* feasible for —
d-dimensional hypercube
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
d-dimensional torus
[65K05, 65Y05]
(*see:* **Parallel computing: models**)

D'Esopo–Pape method
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
D-function
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
D-functions *see:* proximal minimization with —
D-optimal design
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
DACE
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
DACE
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
DAE
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)
Dai–Yuan algorithm
[90C30]
(*see:* **Conjugate-gradient methods**)
damped Gauss–Newton method
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
damped Newton method
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
damped NM
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
Daniel–Gragg–Kaufmann–Stewart reorthogonalized Gram–Schmidt algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
Dantzig largest coefficient pivoting rule
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
Dantzig rule
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
Dantzig–Wolfe decomposition
[49M20, 90-08, 90C06, 90C25, 90C35]
(*see:* **Nondifferentiable optimization: cutting plane methods**; **Simplicial decomposition algorithms**)
Dantzig–Wolfe decomposition
[90C06, 90C25, 90C30, 90C35]
(*see:* **Frank–Wolfe algorithm**; **Simplicial decomposition**; **Simplicial decomposition algorithms**)
Dantzig–Wolfe decomposition *see:* nonlinear —
data
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
data *see:* best fitting to —; evaluation of empirical —; feasibility approach to image reconstruction from projection —; image reconstruction from projection —; length of input —; optimization approach to image reconstruction from projection —; Optimization techniques for phase retrieval based on single-crystal X-ray diffraction —; row conditional proximity —; size of input —; training —
data-association problem
[90C35]
(*see:* **Multi-index transportation problems**)
data classification *see:* Deterministic and probabilistic optimization models for —
data classification via mixed-integer optimization *see:* Multi-class —
data elicitation
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
data elicitation
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
Data envelopment analysis
(90B50, 90B30, 91B82, 90C05)
(*refers to:* **Optimization and decision support systems**)
data envelopment analysis
[90B30, 90B50, 90C05, 90C25, 90C29, 90C30, 90C31, 91B82]
(*see:* **Bilevel programming: optimality conditions and duality**; **Data envelopment analysis**)
data envelopment analysis
[90C27]
(*see:* **Operations research and financial markets**)
data fitting
[90C30]
(*see:* **Generalized total least squares**)
data mapping *see:* computation and —
Data mining
data Mining
(*see:* **Mathematical programming for data mining**)
data mining *see:* Mathematical programming for —
data for multicriteria decision making problems: optimization techniques *see:* Estimating —
data parallelism
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
data perturbation
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
data sets *see:* least squares problems with massive —
Davidon–Fletcher–Powell method
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
Davidon–Fletcher–Powell update
[90C30]
(*see:* **Broyden family of methods and the BFGS update**)
day dynamic travel behavior *see:* day-to- —
day-to-day dynamic travel behavior
[90B15]
(*see:* **Dynamic traffic networks**)
day-to-day dynamic travel behavior
[90B15]
(*see:* **Dynamic traffic networks**)

D.C. decomposition
[90C30]
(*see:* **Large scale trust region problems**)
d.c. function
[65H10, 90B80, 90C11, 90C26, 90C30, 90C31]
(*see:* **Global optimization methods for systems of nonlinear equations**; **Robust global optimization**; **Stochastic transportation and location problems**)
d.c. function
[46A20, 52A01, 65Kxx, 90C30, 90Cxx]
(*see:* **Farkas lemma: generalizations**; **Quasidifferentiable optimization: algorithms for QD functions**)
dc functions
[90C26]
(*see:* **D.C. programming**)
d.c. optimization
[90C25, 90C26, 90C31]
(*see:* **Concave programming**; **Robust global optimization**)
D.C. programming
(90C26)
(*referred to in:* **αBB algorithm**; **Global optimization methods for systems of nonlinear equations**; **Large scale trust region problems**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Reverse convex optimization**; **Standard quadratic optimization problems: theory**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
d.c. programming
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**; **Duality theory: monoduality in convex optimization**)
d.c. programming
[49-XX, 90-XX, 90C30, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**; **Large scale trust region problems**)
d.c. programming problem
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
d.c. set
[90C26]
(*see:* **Global optimization in multiplicative programming**)
DCA
[90C30]
(*see:* **Large scale trust region problems**)
De La Garza method
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
De novo protein design using flexible templates
(92D20, 46N10, 90C10)
De novo protein designUsing rigid templates
DEA
[90B30, 90B50, 90C05, 91B82]
(*see:* **Data envelopment analysis**)
DEA
[90B30, 90B50, 90C05, 91B82]
(*see:* **Data envelopment analysis**)
dead or dog-lawed
(*see:* **Railroad crew scheduling**)
dead point
[90Cxx]
(*see:* **Discontinuous optimization**)
dead-point iterate
[90Cxx]
(*see:* **Discontinuous optimization**)
deadhead arcs
(*see:* **Railroad crew scheduling**)
deadheading
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**; **Railroad locomotive scheduling**)
deadheading *see:* crew —
decision *see:* ex-ante (risk averse, anticipative) —; ex-post (risk prone, adaptive) —; expectation and —; first-stage —; funding —; fuzzy —; investment —; recourse —; second-stage —
decision aid *see:* multicriteria —
decision alternative *see:* set of —
decision analysis
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)
decision maker
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
decision making
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
decision making *see:* financial —; group —; hierarchical —; multicriteria —; multiple criteria —; Preference disaggregation approach: basic features, examples from financial —
decision making problems: optimization techniques *see:* Estimating data for multicriteria —
decision making with rolling horizon
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
decision making with rolling horizon
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
decision making under extreme events
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)
decision making under extreme events
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)
decision making under uncertainty
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
decision models *see:* nonlinear —
decision problem
[90C60]
(*see:* **Complexity theory**; **Computational complexity theory**)
decision problem
[90C60]
(*see:* **Complexity theory**; **Computational complexity theory**)

decision problem *see:* locational —; polynomially transformable —
decision problems *see:* "hit-or-miss" —
decision process *see:* Markov —
decision rule
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
decision rule *see:* minimax —
decision set
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**; **Vector optimization**)
decision support
[65K05, 90B50, 90C05, 90C29, 91B06, 91B60]
(*see:* **Financial applications of multicriteria analysis**; **Multi-objective optimization and decision support systems**; **Railroad crew scheduling**)
decision support methodologies for auditing decisions *see:* Multicriteria —
decision support system
[65K05, 90B50, 90B80, 90C05, 90C29, 91B06]
(*see:* **Facilities layout problems**; **Multi-objective optimization and decision support systems**)
decision support system
[90B80, 90C29, 91A99]
(*see:* **Decision support systems with multiple criteria**; **Facilities layout problems**; **Preference disaggregation**)
decision support system *see:* Asset liability management —; intelligent multicriteria —; multicriteria —; multicriteria group —
decision support systems
[90C29]
(*see:* **Decision support systems with multiple criteria**)
decision support systems *see:* intelligent multicriteria —; Multi-objective optimization and —; Optimization and —
Decision support systems with multiple criteria
(90C29)
(*referred to in:* **Bi-objective assignment problem**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
decision-theoretic framework *see:* Bayesian —
decision theory
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
decision tree
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
decision variable *see:* flow —
decision variables
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**; **Plant layout problems and optimization**)
decision variables x
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
decisions *see:* diversified investment —; first-stage —; inventory and transportation —; Multicriteria decision support methodologies for auditing —; second-stage —
decisions in dynamic optimization *see:* discrete —
decisions in a supply chain *see:* operational —
declarative knowledge
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
declarative language
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
declarative language
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
declarative languages
(*see:* **Planning in the process industry**)
declarative program *see:* pretty-printing a —
declarative program structure *see:* analysing —
declarative programs *see:* classifying —; symbolically transforming —
declarative representation
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
declared interval function *see:* pre- —
decomposable cost
[90C35]
(*see:* **Multi-index transportation problems**)
decompose
[90C25, 90C26]
(*see:* **Decomposition in global optimization**)

decomposition
[15-XX, 49M29, 65-XX, 68Q99, 90-XX, 90C11, 90C15, 90C26, 90C33, 90C35]
(*see:* **Branch and price: Integer programming with column generation**; **Cholesky factorization**; **Generalized benders decomposition**; **Multicommodity flow problems**; **Stochastic bilevel programs**)
Decomposition
[49M27, 49M29, 49M37, 68Q99, 90C06, 90C11, 90C20, 90C30, 90C35, 90C90]
(*see:* **Branch and price: Integer programming with column generation**; **Decomposition principle of linear programming**; **Decomposition techniques for MILP: lagrangian relaxation**; **Generalized benders decomposition**; **MINLP: generalized cross decomposition**; **Mixed integer nonlinear programming**; **Multicommodity flow problems**; **Railroad crew scheduling**; **Successive quadratic programming: decomposition methods**)
decomposition *see:* Benders —; branch —; cross —; Dantzig–Wolfe —; D.C. —; disaggregate simplicial —; dual —; generalized Benders —; generalized cross —; heat exchanger network synthesis without —; Jordan–Hahn —; L-shaped —; Lagrangian —; Lasserre signed —; Lawrence signed —; LU- —; mean value cross —; MINLP: generalized cross —; nested Benders —; nonlinear Dantzig–Wolfe —; operator —; path —; price-directive —; problem —; QR —; range and null space —; regularized Frank–Wolfe —; resource-directive —; restricted simplicial —; signed —; simplicial —; stochastic —; tree —; well-separated pair —; Yosida-Hewitt —
decomposition algorithm *see:* cross —; regularized stochastic —; stochastic —
decomposition algorithms
[90C15]
(*see:* **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
decomposition algorithms
[90B10, 90C05, 90C06, 90C11, 90C31, 90C35]
(*see:* **Nonoriented multicommodity flow problems**; **Parametric mixed integer nonlinear optimization**)
decomposition algorithms *see:* Simplicial —
decomposition algorithms for nonconvex minimization problems
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
Decomposition algorithms for the solution of multistage mean-variance optimization problems
(90C15, 90C90)
decomposition approach *see:* augmented Lagrangian —; Benders —
decomposition-based clustering approach: global optimum search with enhanced positioning *see:* Gene clustering: A novel —
decomposition CA algorithms
[90C30]
(*see:* **Cost approximation algorithms**)
decomposition of a continuous piecewise linear function
[90Cxx]
(*see:* **Discontinuous optimization**)
decomposition/coordination
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
decomposition and cutting planes *see:* Stochastic linear programming: —
decomposition of a function *see:* second order —
Decomposition in global optimization
(90C26, 90C25)
decomposition heuristic
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
decomposition method *see:* feasible —; nonfeasible —
decomposition methods
[90C20, 90C30]
(*see:* **Successive quadratic programming: decomposition methods**)
decomposition methods *see:* Successive quadratic programming: —
decomposition of a monomial ideal *see:* standard pair —
decomposition point
[58E05, 90C30]
(*see:* **Topology of global optimization**)
decomposition points
[58E05, 90C30]
(*see:* **Topology of global optimization**)
Decomposition principle of linear programming
(90C06)
(*referred to in:* **Generalized benders decomposition**; **MINLP: generalized cross decomposition**; **MINLP: logic-based methods**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)
(*refers to:* **Generalized benders decomposition**; **MINLP: generalized cross decomposition**; **MINLP: logic-based methods**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)
decomposition of SLP
[90C06, 90C15]
(*see:* **Stochastic linear programming: decomposition and cutting planes**)
decomposition solution *see:* truncated singular value —
decomposition step
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
decomposition techniques
[49K35, 49M27, 49Q10, 65K10, 74K99, 74Pxx, 90C15, 90C25, 90C90, 91A65]
(*see:* **Convex max-functions**; **Multilevel optimization in mechanics**; **Multistage stochastic programming: barycentric approximation**; **Two-stage stochastic programs with recourse**)
decomposition techniques
[90C15]
(*see:* **L-shaped method for two-stage stochastic programs with recourse**)
Decomposition techniques for MILP: lagrangian relaxation
(90C90, 90C30)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming:**

branch and bound methods; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **Lagrange, Joseph-Louis**; **Lagrangian multipliers methods for convex programming**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multi-objective optimization: lagrange duality**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **Lagrange, Joseph-Louis**; **Lagrangian multipliers methods for convex programming**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multi-objective optimization: lagrange duality**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
decompositions *see:* Branchwidth and branch —
decrease
[65K05, 90C30]
(*see:* **Random search methods**)
decrease *see:* high-order approximating cone of —; high-order approximating vector of —; high-order cones of —; high-order set of —
decrease conditions *see:* sufficient —
decreasing
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
decreasing *see:* convex —; nonlinear —
decreasing tail *see:* RSM-distribution with algebraically —
Dedekind number
[90C09]
(*see:* **Inference of monotone boolean functions**)
deepening *see:* iterative —
default strategies
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**; **Planning in the process industry**)
deficit of a network node
[90C35]
(*see:* **Minimum cost flow problem**)
definable classes *see:* set- —
defined end *see:* clearly —
defined start-ups *see:* well- —
definite *see:* positive —
definite completion problem *see:* positive (semi) —
definite matrices *see:* positive —
definite matrix *see:* partial —; positive —; strongly positive —
definite quadratic binary programming *see:* positive semi- —
definite quadratic function *see:* positive —
definite quadratic models *see:* positive —
definiteness *see:* positive —
definition *see:* algorithmic —; optimization algorithm —
definition (colloquial) *see:* optimization: —
deflected gradient methods
[90C30]
(*see:* **Cost approximation algorithms**)
deformable model
[90C90]
(*see:* **Optimization in medical imaging**)
deformable templates
[90C90]
(*see:* **Optimization in medical imaging**)
deformation process
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
degeneracy
[90C60]
(*see:* **Complexity of degeneracy**)
degeneracy
[90C60]
(*see:* **Complexity of degeneracy**)
degeneracy *see:* Complexity of —; near —; resolving —
degenerate
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
degenerate *see:* dual —; primal —
degenerate basic solution
[90C05]
(*see:* **Linear programming**)
degenerate basis *see:* dual —; primal —
degenerate BFS
[90C60]
(*see:* **Complexity of degeneracy**)
degenerate BFS *see:* nearly —
degenerate pivot operation
[90C35]
(*see:* **Minimum cost flow problem**)
degenerate problem
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
degenerate system
[90C60]
(*see:* **Complexity of degeneracy**)
degradation in quality of both water environment *see:* minimizing the —
degree *see:* Brouwer —; graph —; Leray–Schauder —; maximum —; motionless —; topological —
degree of a binomial
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
degree c *see:* algorithm polynomial of —

degree-constrained subgraph problem
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
degree deletion heuristic *see:* increasing- —
degree of flexibility *see:* fixed —; optimal —
degree of inclusion
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
degree of linearity
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
degree minimum spanning tree problem *see:* bounded —
degree of a monomial ideal *see:* arithmetic —
degree ordering *see:* minimum —
degree parallelism alignment problem *see:* constant —
degree theory
[90C33]
(*see:* **Topological methods in complementarity theory**)
degree zero *see:* homogeneous of —
degrees of credence
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
delaunay triangulation
[52B11, 52B45, 52B55, 68Q20]
(*see:* **Optimal triangulations**; **Volume computation for polytopes: strategies and performances**)
delay problem in air traffic control *see:* ground —
delay programs *see:* air traffic control and ground —
delay system *see:* time- —
delay systems *see:* time- —
deleting matroid elements
[90C09, 90C10]
(*see:* **Matroids**)
deletion
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
deletion heuristic *see:* increasing-degree —; incremental —
deletion in matroids
[90C09, 90C10]
(*see:* **Oriented matroids**)
deletion problem *see:* vertex (arc) —
deliveries *see:* Vehicle routing problem with simultaneous pickups and —
delivers *see:* pick-up and —
delivery *see:* express shipment —
delivery problem *see:* express —
delivery systems *see:* Model based control for drug —
Delphi method
[90C26, 90C30]
(*see:* **Forecasting**)
delta function
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
demand *see:* elastic travel —; fixed travel —; net —; regional —; unity —; water —; water resources planning under uncertainty on hydrological exogenous inflow and —
demand arcs
(*see:* **Railroad crew scheduling**)
demand CMST *see:* equal —
demand function *see:* aggregate excess —
demand functions *see:* elastic demand traffic network problems with travel —
demand node
[90C30, 90C35]
(*see:* **Minimum cost flow problem**; **Optimization in water resources**)
demand traffic network equilibrium *see:* fixed —
demand traffic network problems *see:* fixed —
demand traffic network problems with travel demand functions *see:* elastic —
denial
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
denial *see:* checklist —
density
[90C05, 90C25, 90C30, 90C34]
(*see:* **Fractional zero-one programming**; **Semi-infinite programming: discretization methods**)
density *see:* boltzmann —; steady-state distribution —; transition probability —
density annealing *see:* Gaussian —
density clustering
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: two-phase methods**)
density function *see:* logconcave probability —
dep-station
(*see:* **Railroad locomotive scheduling**)
dep-time
(*see:* **Railroad locomotive scheduling**)
departure *see:* duty-before- —
departure connection arc *see:* ground- —
departure-ground
(*see:* **Railroad locomotive scheduling**)
departure node
(*see:* **Railroad crew scheduling**; **Railroad locomotive scheduling**)
departure-station
(*see:* **Railroad crew scheduling**)
dependence *see:* continuous —; linear —; noisy functional —
dependence analysis
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
dependence property *see:* boundary —
dependency *see:* interval —
dependency set *see:* common —
dependent *see:* positively linearly —
dependent constraints *see:* maximum function with —
dependent hyperplanes
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
dependent property *see:* norm- —
dependent protein force field via linear optimization *see:* Distance —
dependent set
[90C09, 90C10]
(*see:* **Matroids**)
dependent set *see:* minimal —
dependent traveling salesman problem *see:* Time- —
dependent variables
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)

depot
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
depot *see:* multiple —; virtual —
depot group
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
depot/multiple depots *see:* single —
depot vehicle scheduling problem *see:* Multi- —; Single- —
depot vehicle scheduling problems *see:* multi- —; Single- —
depots *see:* single depot/multiple —
depth *see:* evaluation —
depth of a Boolean circuit
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
depth-first
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)
depth-first search
[05C85, 90C10, 90C29, 90C35]
(*see:* **Directed tree networks**; **Generalized networks**; **Multi-objective integer linear programming**)
depth-first search with backtracking
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
depth-first tree search
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
Depth-First Tree Search *see:* Parallel —
derangements
[34E05, 90C27]
(*see:* **Asymptotic properties of random multidimensional assignment problem**)
derivation of cross-entropy *see:* axiomatic —
derivation of entropy *see:* axiomatic —
derivation of the Hamilton–Jacobi–Bellman equation
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)
derivation of the principle of maximum entropy *see:* axiomatic —
derivation of the principle of minimum cross-entropy *see:* axiomatic —
derivative
[26E25, 49J52, 52A27, 90C99]
(*see:* **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
derivative *see:* B- —; Clarke —; clarke generalized —; Clarke generalized directional —; Clarke–Rockafellar generalized —; Dini —; Dini conditional lower —; Dini conditional upper —; Dini lower —; Dini lower directional —; Dini upper —; Dini upper directional —; directional —; directional Clarke —; generalized directional —; generalized second order directional —; Hadamard —; Hadamard conditional lower —; Hadamard conditional upper —; Hadamard lower directional —; Hadamard upper directional —; kth directional —; parameter —; upper —
derivative approach *see:* material —
derivative conditions *see:* matching of —
derivative-free descent method
[90C30, 90C33]
(*see:* **Implicit lagrangian**)
Derivative-free methods for non-smooth optimization
(65K05, 90C56)
(*referred to in:* **Maximum cut problem, MAX-CUT**)
derivative of a function
[26E25, 49J52, 52A27, 90C99]
(*see:* **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
derivative of an integral
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
derivative method *see:* adjoint —
derivative of a probability function
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
derivative of a probability function
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
derivative ranges *see:* Bounding —
derivative in shape optimization *see:* Topological —
derivatives
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
derivatives
[60J05, 90C15, 90C27]
(*see:* **Derivatives of markov processes and their simulation**; **Derivatives of probability measures**; **Discrete stochastic optimization**)
derivatives *see:* Dini directional —; directional —; distributional —; elementary partial —; evaluation of objective functions and/or —; Hadamard directional —; handcoded —; high-order directional —; higher-order —; higher-order directional —; lower and upper directional —; matrix of second partial —; method of bad —; pricing —; process —; Quasidifferentiable optimization: Dini derivatives, clarke —; sensitivity —; simulation of —
derivatives, clarke derivatives *see:* Quasidifferentiable optimization: Dini —
Derivatives of markov processes and their simulation
(90C15, 60J05)
(*referred to in:* **Derivatives of probability and integral functions: general theory and examples**; **Derivatives of probability measures**; **Discrete stochastic optimization**)
(*refers to:* **Derivatives of probability and integral functions: general theory and examples**; **Derivatives of probability measures**; **Discrete stochastic optimization**; **Optimization in operation of electric and energy power systems**; **Stochastic quasigradient methods**)
derivatives in optimization *see:* Dini and Hadamard —
Derivatives of probability and integral functions: general theory and examples
(90C15)
(*referred to in:* **Derivatives of markov processes and their simulation**; **Derivatives of probability measures**; **Discrete stochastic optimization**)
(*refers to:* **Derivatives of markov processes and their**

simulation; **Derivatives of probability measures**; **Discrete stochastic optimization**; **Optimization in operation of electric and energy power systems**)
Derivatives of probability measures
(90C15)
(*referred to in:* **Derivatives of markov processes and their simulation**; **Derivatives of probability and integral functions: general theory and examples**; **Discrete stochastic optimization**)
(*refers to:* **Derivatives of markov processes and their simulation**; **Derivatives of probability and integral functions: general theory and examples**; **Discrete stochastic optimization**; **Optimization in operation of electric and energy power systems**; **Stochastic quasigradient methods**)
derivatives of structural response
[90C26, 90C90]
(*see:* **Structural optimization: history**)
descending index
[60J05, 90C15]
(*see:* **Derivatives of markov processes and their simulation**)
descent
[90C30]
(*see:* **Nonlinear least squares problems**)
descent *see:* direction of —; dual —; ε-steepest —; first —; gradient —; gradient-related —; hypodifferential —; loss of —; method of codifferential —; method of hypodifferential —; method of steepest —; Newtonian —; rate of steepest —; reformulation —; steepest —; variable neighborhood —
descent algorithm
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
descent algorithm *see:* steepest —
descent-based methods
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
descent direction
[90C06, 90C30, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**; **Large scale unconstrained optimization**; **Sequential simplex method**)
descent direction
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
descent direction *see:* Dini steepest —; Hadamard steepest —; quasi-Newtonian —; steepest —
descent directions *see:* construction of —
descent directions and efficient points *see:* Discretely distributed stochastic programs: —
descent flow
[58E05, 90C30]
(*see:* **Topology of global optimization**)
descent flow
[58E05, 90C30]
(*see:* **Topology of global optimization**)
descent iterations *see:* Local attractors for gradient-related —
descent method
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
descent method
[90C30]
(*see:* **Nonlinear least squares problems**)
descent method *see:* coordinate —; derivative-free —; steepest —
descent in a nonlinear program *see:* loss of —
descent in a nonlinear programming algorithm
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
descent properties
[90C30]
(*see:* **Cost approximation algorithms**)
descent ray
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
descent step
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
descent vector
[49M29, 65K10, 90C05, 90C06, 90C25, 90C30, 90C34]
(*see:* **Local attractors for gradient-related descent iterations**; **Semi-infinite programming, semidefinite programming and perfect duality**)
descent vector
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
descent vector *see:* steepest —
description *see:* problem —
description method *see:* double —
descriptional complexity
[90C60]
(*see:* **Kolmogorov complexity**)
Descriptional complexity
[90C60]
(*see:* **Kolmogorov complexity**)
descriptive complexity
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
descriptive perspective
[90C29]
(*see:* **Preference modeling**)
descriptors *see:* context —
design
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
design
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
design *see:* algorithm —; batch plant —; Beam selection in radiotherapy treatment —; circuit —; D-optimal —; distribution system —; experiment —; experimental —; fully stressed —; global optimal —; logical —; model for parallel algorithm —; molecular —; multidisciplinary —; Multilevel methods for optimal —; multiload shape —; multiload truss —; network —; Operations research models for supply chain management and —; optimal —; optimal experimental —; optimal shape —; point —; process —;

robust obstacle-free shape —; robust obstacle-free truss —; sequential experimental —; shape —; structural —; supply chain —; truss —
design analysis
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
design approaches *see:* Optimal solvent —
design centering
[65D18, 90B85, 90C26]
(*see:* **Global optimization in location problems**)
design of composite structures
[90C26, 90C29]
(*see:* **Optimal design of composite structures**)
design of composite structures
[90C26, 90C29]
(*see:* **Optimal design of composite structures**)
design of composite structures *see:* Optimal —
design and control *see:* interaction of —; MINLP: applications in the interaction of —; Multi-objective optimization: interaction of —
design of dynamic systems by constructive nonlinear dynamics *see:* Robust —
design models *see:* strategic —
design in nonlinear optics *see:* Optimal —
design of operators
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
design of optimal shapes
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
design optimization *see:* Multidisciplinary —
design optimization in CFD
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
Design optimization in computational fluid dynamics
(90C90)
(*referred to in:* **Interval analysis: application to chemical engineering design problems**; **Multidisciplinary design optimization**; **Multilevel methods for optimal design**; **Optimal design of composite structures**; **Optimal design in nonlinear optics**; **Structural optimization: history**)
(*refers to:* **Bilevel programming: applications in engineering**; **Interval analysis: application to chemical engineering design problems**; **Multidisciplinary design optimization**; **Multilevel methods for optimal design**; **Optimal design of composite structures**; **Optimal design in nonlinear optics**; **Structural optimization: history**)
design optimization process *see:* automated —
design pattern based model *see:* gIS —
design problem
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
design problem *see:* multiple criteria —; network —; Production-distribution system —; survivable network —
design problems
[90C29]
(*see:* **Multiple objective programming support**)
design problems *see:* Interval analysis: application to chemical engineering —; Network —; optimal —
Design of robust model-based controllers via parametric programming
design and schedule construction *see:* network —
design and scheduling of batch processes *see:* MINLP: —
design space
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
design stage *see:* conceptual —; detailed —; preliminary —
design superstructure
[90C90]
(*see:* **MINLP: heat exchanger network synthesis**)
design of a supply chain *see:* strategic —
design, synthesis and control *see:* interaction of —
design under uncertainty
[49M37, 90C11, 90C30, 90C90]
(*see:* **Mixed integer nonlinear programming**; **Successive quadratic programming: applications in the process industry**)
design under uncertainty *see:* Global optimization in batch —; process synthesis and —
design using flexible templates *see:* De novo protein —
design variables
[90C90, 91B28]
(*see:* **Robust optimization**)
design variables *see:* discrete —
designs *see:* subset interconnection —
designUsing rigid templates *see:* De novo protein —
destructive method
[90B35]
(*see:* **Job-shop scheduling problem**)
det problem *see:* max- —
detailed design stage
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
detecting redundancy *see:* deterministic method for —; probabilistic method for —
detection *see:* low-level feature —
detection via semidefinite programming *see:* Maximum likelihood —
detention costs
(*see:* **Railroad crew scheduling**)
determinant expansion of a matrix *see:* standard —
determinant maximization *see:* Semidefinite programming and —
determination *see:* molecular structure —; orbits —
determination of clusters size threshold
[92C05, 92C40]
(*see:* **Protein loop structure prediction methods**)
determination: convex global underestimation *see:* Molecular structure —
determination of rmsd threshold
[92C05, 92C40]
(*see:* **Protein loop structure prediction methods**)
determined graph *see:* rank —
determined system of nonlinear equations *see:* well- —
determined variable *see:* strongly —
Determining the optimal number of clusters
(90C26, 91C20, 68T20, 68W10, 90C11, 92-08, 92C05, 92D10)
deterministic
[90C60]
(*see:* **Complexity classes in optimization**)

deterministic
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
deterministic algorithm *see:* polynomial time —; sequential —
deterministic cutting plane methods *see:* regularization of —
deterministic equivalent model
[90C30, 90C35]
(*see:* **Optimization in water resources**)
deterministic equivalent problem
[90C15]
(*see:* **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
deterministic global optimization *see:* LP strategy for interval-Newton method in —; Mixed integer nonlinear bilevel programming: —
deterministic global optimization algorithm
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
deterministic method for detecting redundancy
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
deterministic neural network
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
deterministic optimization
[65K05]
(*see:* **Direct global optimization algorithm**)
Deterministic and probabilistic optimization models for data classification
(65K05, 55R15, 55R35, 90C11)
(*referred to in:* **Linear programming models for classification**; **Mixed integer classification problems**)
deterministic problem *see:* static —; underlying —
deterministic shortest path problem
[49L20, 90C39, 90C40]
(*see:* **Dynamic programming: infinite horizon problems, overview**)
deterministic Turing machine
[90C60]
(*see:* **Complexity theory**)
deterministic Turing machine *see:* space complexity of a —; time complexity of a —
development *see:* algorithmic —; model —
development and evaluation *see:* software —
deviation *see:* external —; internal —; least absolute —; maximum absolute —; mean absolute —
device *see:* local search —
devices and related techniques *see:* acceleration —
DEXPTIME
[90C60]
(*see:* **Complexity classes in optimization**)
DFBB
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
DFP method
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
DFP update
[90C30]
(*see:* **Broyden family of methods and the BFGS update**)
DFP update
[90C30]
(*see:* **Broyden family of methods and the BFGS update**)
diagnosing and tracing infeasibilities
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**; **Planning in the process industry**)
diagnosis *see:* breast cancer —; medical —
diagnosis: optimization-based methods *see:* Disease —
diagnostic rotation
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
diagnostic rotations
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
diagonal
[47J20, 49J40, 65K10, 90C33]
(*see:* **Interval analysis for optimization of dynamical systems**; **Solution methods for multivalued variational inequalities**)
diagonal *see:* negative main —; quasi- —
diagonal dominance condition
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
diagonal matrix
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
diagonal model
[91B50]
(*see:* **Financial equilibrium**)
diagonal operator *see:* block-0- —; off-0- —
diagonal pivot
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
diagonal shift matrix
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**; **QBB global optimization method**)
diagonal underestimation matrix
[49M37, 65K10, 90C26, 90C30]
(*see:* **QBB global optimization method**)
diagram *see:* composition interval —; conceptual —; cumulative sum —; farthest-point Voronoi —; Hasse —; temperature interval —; Voronoi —
diagrams *see:* Voronoi —
diagrams in facility location *see:* Voronoi —
DIAL *see:* S- —
dial-a-ride
[90B06]
(*see:* **Vehicle routing**)
dial-a-ride *see:* m- —
diameter
[90C35]
(*see:* **Multi-index transportation problems**)

dichotomy *see:* generalized-upper-bound —; GUB —; variable —
dicycle
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
dielectric structures *see:* Global optimization of planar multilayered —
Dienes implication *see:* Kleene– —
difference *see:* minimum composition —; temporal —
difference approximation *see:* finite- —
difference convex function
[26B25, 26E25, 49J40, 49J52, 49M05, 49S05, 74G99, 74H99, 74Pxx, 90C99]
(*see:* **Quasidifferentiable optimization**; **Quasidifferentiable optimization: variational formulations**)
difference of convex functions
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
difference equation
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
difference estimate
[90C15]
(*see:* **Derivatives of probability measures**)
difference of max-type functions
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
difference methods *see:* finite —
difference of monotonic functions
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
difference quotients
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
difference of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
difference sublinear
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
difference sublinear function
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
differences *see:* divided —; finite —
differences of convex sets
[26E25, 49J52, 52A27, 90C99]
(*see:* **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
differencing
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
different
(*see:* **Planning in the process industry**)
differentiability *see:* direct —; inverse —; Minimax: directional —
differentiable
[65K05, 90C15, 90C30, 90Cxx]
(*see:* **Derivatives of probability measures**; **Dini and Hadamard derivatives in optimization**; **Image space approach to optimization**)
differentiable *see:* Dini —; dini directionally —; directionally —; Hadamard —; hadamard directionally —; process —; strictly —
differentiable convex programming
[90C30]
(*see:* **Lagrangian duality: BASICS**)
differentiable exact penalty function approach *see:* continuously —
differentiable family of measures *see:* weakly L_1 (v)- —
differentiable function
[26B25, 26E25, 49J52, 90C99]
(*see:* **Quasidifferentiable optimization**)
differentiable function
[90C30]
(*see:* **Frank–Wolfe algorithm**)
differentiable function *see:* C- —; Dini —; Dini conditionally —; Dini conditionally directionally —; Dini directionally —; Dini uniformly —; Dini uniformly directionally —; directionally —; Fréchet —; Hadamard —; Hadamard conditionally —; Hadamard conditionally directionally —; Hadamard directionally —; piecewise —; piecewise continuously —; piecewise twice- —
Differentiable Functions and Applications *see:* minimization Methods for Non- —
differentiable (GD) function *see:* generalized —
differentiable MINLPs *see:* twice- —
differentiable NLPs *see:* Twice- —
differentiable part of a function *see:* twice- —
differential
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
differential *see:* C- —; Clarke directional —; generalized directional —; limiting —; one-sided —; Rockafellar directional —
differential and algebraic equations
[49M37, 65L99, 90C11, 93-XX]
(*see:* **MINLP: applications in the interaction of design and control**; **Optimization strategies for dynamic systems**)
differential cost
[49L99]
(*see:* **Dynamic programming: average cost per stage problems**)
differential cost vector
[49L99]
(*see:* **Dynamic programming: average cost per stage problems**)
differential dynamic programming
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
differential equation
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: differential equations**)
differential equation
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: differential equations**)
differential equation *see:* Knizhnik–Zamolodchikov —; stochastic —

differential equations
[34-xx, 34Bxx, 34Lxx, 93E24]
(*see:* **Complexity and large-scale least squares problems**)
differential equations *see:* Duality in optimal control with first order —; Eigenvalue enclosures for ordinary —; First order partial —; Interval analysis: —; ordinary —; partial —
Differential equations and global optimization
(60G35, 65K05)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Direct global optimization algorithm**; **Global optimization based on statistical models**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Topology of global optimization**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Direct global optimization algorithm**; **Global optimization based on statistical models**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Simulated annealing methods in protein folding**; **Topology of global optimization**)
differential stability *see:* Sensitivity and stability in NLP: continuity and —
differentiation
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
differentiation
[01A99, 65H99, 65K99]
(*see:* **Automatic differentiation: point and interval**; **Leibniz, gottfried wilhelm**)
differentiation *see:* algorithmic —; analytical —; automatic —; backward mode in automatic —; computational —; forward mode of automatic —; goal-oriented —; internal numerical —; interval automatic —; Nonlocal sensitivity analysis with automatic —; numerical —; reverse —; reverse mode automatic —; symbolic —; vector forward automatic —
differentiation arithmetic
[90C26, 90C30]
(*see:* **Bounding derivative ranges**)
differentiation: calculation of the Hessian *see:* Automatic —
differentiation: calculation of Newton steps *see:* Automatic —
differentiation: geometry of satellites and tracking stations *see:* Automatic —
differentiation: introduction, history and rounding error estimation *see:* Automatic —
differentiation: parallel computation *see:* Automatic —
differentiation: point and interval *see:* Automatic —
differentiation: point and interval taylor operators *see:* Automatic —
differentiation: root problem and branch problem *see:* Automatic —
difficult optimization problems *see:* Modeling —
difficulties in bilinear programming
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
diffraction data *see:* Optimization techniques for phase retrieval based on single-crystal X-ray —
diffusion equation
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
diffusion equation *see:* nonlinear —
diffusion equation method
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
diffusion flux models *see:* estimation of —
diffusion fluxes *see:* estimation of 1D- —
diffusion process
[78M50, 90B50, 91B28]
(*see:* Global optimization algorithms for financial planning problems; **Laplace method and applications to optimization problems**)
digraph
[03B52, 03E72, 05C05, 05C40, 47S40, 68R10, 68T27, 68T35, 68Uxx, 90Bxx, 90C09, 90C10, 90C35, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Combinatorial matrix analysis**; **Network design problems**)
digraph *see:* circuit of a —; complete —; directed cycle of a —; min-max —; set of edges of a —; signed —; strongly connected —; strongly connected components of a —; vertex of a —
digraph representation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
dihedral angle
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
dihedral angle
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
dijoin
[90C35]
(*see:* **Feedback set problems**)
dilatation
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
dilation *see:* space —
dimension
[90C30]
(*see:* **Simplicial decomposition**)
dimension algorithm *see:* dimension-by- —
dimension-by-dimension algorithm
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
dimension pivoting algorithms *see:* varying —
dimensional Brownian motion *see:* N- —
dimensional control problem *see:* finite- —
dimensional cube connected cycle *see:* k- —
dimensional generalized order complementarity problem *see:* infinite- —
dimensional grid *see:* 2- —
dimensional hypercube *see:* d- —
dimensional integration *see:* high- —
dimensional knapsack problem *see:* m- —

dimensional linear program *see:* finite- —
dimensional linear programming *see:* infinite- —
dimensional marginal probability distribution function *see:* one- —; two- —
DIMENSIONAL MATCHING *see:* 3- —
dimensional matching problem *see:* 3- —
dimensional models for entropy optimization for image reconstruction *see:* finite- —
dimensional nonlinear equation *see:* one- —
dimensional optimization *see:* infinite- —
dimensional subspace *see:* finite- —
dimensional symmetric interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
dimensional torus *see:* 2- —; d- —
dimensional transportation problem *see:* three- —
dimensional variational inequality problem *see:* finite- —
dimensional vectors *see:* lexicographical ordering for n- —
dimensionality *see:* cures of —; curse of —
Dini codifferentiable function
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
Dini conditional lower derivative
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dini conditional upper derivative
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dini conditionally differentiable function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dini conditionally directionally differentiable function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dini derivative
[26E25, 49J52, 52A27, 65K05, 90C99, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: optimality conditions**)
Dini derivative
[26E25, 49J52, 52A27, 90C99]
(*see:* **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
Dini derivatives, clarke derivatives *see:* Quasidifferentiable optimization: —
Dini differentiable
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dini differentiable function
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
Dini directional derivatives
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dini directional derivatives
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
dini directionally differentiable
[65K05, 90C30, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Minimax: directional differentiability**)
Dini directionally differentiable function
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
Dini and Hadamard derivatives in optimization
(90Cxx, 65K05)
(*referred to in:* **Global optimization: envelope representation**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**; **Quasidifferentiable optimization: optimality conditions**)
(*refers to:* **Global optimization: envelope representation**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**)
Dini lower derivative
[26E25, 49J52, 52A27, 90C99]
(*see:* **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
Dini lower directional derivative
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dini quasidifferentiable function
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
Dini quasidifferential
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
Dini steepest ascent direction
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dini steepest descent direction
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dini sup-stationary point
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dini uniformly differentiable function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dini uniformly directionally differentiable function
[26E25, 49J52, 52A27, 90C99]
(*see:* **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
Dini upper derivative
[26E25, 49J52, 52A27, 90C99]

(*see:* **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
Dini upper directional derivative
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Dinkelbach algorithm
[90C32]
(*see:* **Quadratic fractional programming: Dinkelbach method**)
Dinkelbach method
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
Dinkelbach method
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**; **Quadratic fractional programming: Dinkelbach method**)
Dinkelbach method *see:* Quadratic fractional programming: —
Diophantine approximation problem *see:* simultaneous —
Diophantine equations
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
dipole moment
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
direct approach
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
direct collision
(*see:* **Broadcast scheduling problem**)
direct differentiability
[90C15]
(*see:* **Derivatives of probability measures**)
Direct global optimization algorithm
(65K05)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Global optimization based on statistical models**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Topology of global optimization**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Global optimization based on statistical models**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Topology of global optimization**)
direct iteration
[65H10, 65J15]
(*see:* **Contraction-mapping**)
direct search
[90C30]
(*see:* **Suboptimal control**)
Direct search Luus—Jaakola optimization procedure
(93-XX)
(*referred to in:* **Interval analysis: unconstrained and constrained optimization**)
(*refers to:* **Interval analysis: unconstrained and constrained optimization**)
direct search optimization
[93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**)
direct-sequential
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
directed arc in a directed network
[90C35]
(*see:* **Maximum flow problem**)
directed arc in a network
[90C35]
(*see:* **Minimum cost flow problem**)
directed capacitated network
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
directed Chinese postman problem
[90C35]
(*see:* **Minimum cost flow problem**)
directed cycle *see:* cost of a —
directed cycle of a digraph
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
directed divergence
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
directed graph
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**; **Oriented matroids**)
directed network
[90C35]
(*see:* **Maximum flow problem**; **Minimum cost flow problem**)
directed network *see:* directed arc in a —; endpoint of an arc in a —; node in a —
directed path
[90C35]
(*see:* **Maximum flow problem**; **Minimum cost flow problem**)
directed tree
[05C85]
(*see:* **Directed tree networks**)
Directed tree networks
(05C85)
(*referred to in:* **Auction algorithms**; **Bottleneck steiner tree problems**; **Capacitated minimum spanning trees**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Generalized networks**; **Maximum flow problem**; **Minimax game tree searching**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
directed walk
[90C35]
(*see:* **Minimum cost flow problem**)

direction
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
direction *see:* ascendant —; away —; centering —; compute the search —; coordinate —; critical —; descent —; Dini steepest ascent —; Dini steepest descent —; feasible —; feasible ascendant —; Hadamard steepest ascent —; Hadamard steepest descent —; high-order critical —; high-regular critical —; hyperspherical —; improving feasible —; jump —; quasi-Newtonian descent —; search —; soaring —; steepest ascent —; steepest descent —
direction angles
[26A24, 65K99, 85-08]
(*see:* **Automatic differentiation: geometry of satellites and tracking stations**)
direction computation
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)
direction of descent
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
direction finding problem
[90C30]
(*see:* **Frank–Wolfe algorithm**)
direction finding problem
[90C30]
(*see:* **Frank–Wolfe algorithm**)
direction finding problem *see:* regularized —
direction hit and run *see:* hyperspheres —
direction method *see:* reference —
direction method for nonlinear programming *see:* feasible —
direction methods *see:* feasible —
direction of negative curvature
[90C06]
(*see:* **Large scale unconstrained optimization**)
direction, preserving an activity
[90Cxx]
(*see:* **Discontinuous optimization**)
direction subclass *see:* conjugate —
direction vector *see:* reference —
directional Clarke derivative
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)
directional derivative
[26B25, 26E25, 46N10, 49J52, 90-00, 90C26, 90C30, 90C31, 90C47, 90C99]
(*see:* **Global optimization: envelope representation**; **Lagrangian duality: BASICS**; **Nondifferentiable optimization**; **Quasidifferentiable optimization**; **Sensitivity and stability in NLP: continuity and differential stability**)
directional derivative
[65K05, 90C30, 90Cxx]
(*see:* **Minimax: directional differentiability**; **Quasidifferentiable optimization: optimality conditions**)
directional derivative *see:* Clarke generalized —; Dini lower —; Dini upper —; generalized —; generalized second order —; Hadamard lower —; Hadamard upper —; kth —
directional derivatives
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
directional derivatives *see:* Dini —; Hadamard —; high-order —; higher-order —; lower and upper —
directional differentiability *see:* Minimax: —
directional differential *see:* Clarke —; generalized —; Rockafellar —
directional SOCQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
directionally differentiable
[90C30, 90C31, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions; **Image space approach to optimization**)
directionally differentiable *see:* dini —; hadamard —
directionally differentiable function
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
directionally differentiable function *see:* Dini —; Dini conditionally —; Dini uniformly —; Hadamard —; Hadamard conditionally —
directions *see:* combined method of feasible —; cone of critical —; cone of feasible —; construction of descent —; methods of feasible —; orthogonal search —; steep —
directions and efficient points *see:* Discretely distributed stochastic programs: descent —
directions in interval branch and bound methods *see:* Interval analysis: subdivision —
directions of P *see:* covers all edge- —
directive decomposition *see:* price- —; resource- —
directly left-reachable
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
directly right-reachable
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
Dirichlet distribution
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
Dirichlet problem *see:* nonsmooth —
disaggregate simplicial decomposition
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
disaggregated representation
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
disaggregated representation
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
Disaggregation
(*see:* **Optimal planning of offshore oilfield infrastructure**)
disaggregation *see:* preference —
disaggregation analysis *see:* preference —
disaggregation approach *see:* preference —
disaggregation approach: basic features, examples from financial decision making *see:* Preference —

disaggregation method
[90C29, 91A99]
(*see:* **Preference disaggregation**)
disaggregation in multi-objective optimization
[90C29, 91A99]
(*see:* **Preference disaggregation**)
disaggregation paradigm
[90C29, 91A99]
(*see:* **Preference disaggregation**)
disaggregation of preferences
[91B06, 91B60]
(*see:* **Financial applications of multicriteria analysis**)
disaggregation system *see:* interactive —
disaggregation under uncertainty
[90C29, 91A99]
(*see:* **Preference disaggregation**)
disallowed node
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
discarding far-from-native conformations
[92C05, 92C40]
(*see:* **Protein loop structure prediction methods**)
disconnected matroid
[90C09, 90C10]
(*see:* **Matroids**)
discontinuous function
[93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**)
Discontinuous optimization
(90Cxx)
(*referred to in:* **Nondifferentiable optimization**)
(*refers to:* **Conjugate-gradient methods**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Nondifferentiable optimization**)
discontinuous optimization
[90Cxx]
(*see:* **Discontinuous optimization**)
Discontinuous optimization
[90Cxx]
(*see:* **Discontinuous optimization**)
discordance
[91B06, 91B60]
(*see:* **Financial applications of multicriteria analysis**)
discordance
[90-XX]
(*see:* **Outranking methods**)
discordance *see:* concordance- —
discordant coalition
[90-XX]
(*see:* **Outranking methods**)
discount factor
[49L20, 49L99, 90C39, 90C40]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: undiscounted problems**)
discounted infinite horizon problem
[49L20]
(*see:* **Dynamic programming: inventory control**)
discounted problem
[49L20, 49L99, 90C40]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: stochastic shortest path problems**)
discounted problem
[49L20, 90C39, 90C40]
(*see:* **Dynamic programming: discounted problems**; **Dynamic programming: undiscounted problems**)
discounted problem with bounded cost per stage
[49L20, 90C39, 90C40]
(*see:* **Dynamic programming: infinite horizon problems, overview**)
discounted problems *see:* Dynamic programming: —
Discovery *see:* logic of Scientific —
discrepancy search *see:* limited —
discrete approximation
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
discrete approximation
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
discrete-continuous global optimization *see:* mixed —
discrete-continuous optimization problems *see:* Continuous reformulations of —
discrete convergence
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
discrete convergence *see:* weak —
discrete convex analysis
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
discrete convex analysis
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
discrete decisions in dynamic optimization
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
discrete design variables
[90C26, 90C90]
(*see:* **Structural optimization: history**)
discrete distributions *see:* Logconcavity of —
discrete dynamic complementarity problem
[90C33]
(*see:* **Order complementarity**)
discrete dynamical systems
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
discrete ε-global local maximizers *see:* set of —
discrete event dynamic system
[90C15]
(*see:* **Stochastic quasigradient methods: applications**)
discrete filled function
[65K05, 90C26, 90C30, 90C59]
(*see:* **Global optimization: filled function methods**)
discrete free variables *see:* Generalized geometric programming: mixed continuous and —

discrete function
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
discrete functions
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
discrete global maximizer
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
discrete left local maximizer
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
discrete location and assignment
[68M20, 90B06, 90B10, 90B35, 90B80, 90B85, 90C10, 90C27, 90Cxx, 91Axx, 91Bxx]
(*see:* **Facility location with externalities**; **Vehicle scheduling**)
discrete location problem
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
discrete logconcave distributions
[90C15]
(*see:* **Logconcavity of discrete distributions**)
discrete measure
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
discrete midpoint convexity
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
discrete model in OR
[90B80, 90B85]
(*see:* **Warehouse location problem**)
discrete monotonic optimization problems
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
discrete Mosco convergence
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
discrete multiple criteria problem
[90C29]
(*see:* **Multiple objective programming support**)
discrete neighborhood
[65K05, 90C05, 90C25, 90C26, 90C30, 90C34, 90C59]
(*see:* **Global optimization: filled function methods**; **Semi-infinite programming: discretization methods**)
discrete optimization
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
discrete optimization
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
discrete optimization *see:* Convex —
discrete optimization oracle *see:* linear —
discrete Painlevé–Kuratowski convergence
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
discrete polyblock algorithm
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
discrete probability distribution *see:* logconcave —; logconcave univariate —
discrete resource allocation problem
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
discrete separation theorem *see:* Frank —
discrete single-commodity single-criterion uncapacitated static multifacility
[90B80, 90B85]
(*see:* **Warehouse location problem**)
Discrete stochastic optimization
(90C15, 90C27)
(*referred to in:* **Derivatives of markov processes and their simulation**; **Derivatives of probability and integral functions: general theory and examples**; **Derivatives of probability measures**)
(*refers to:* **Derivatives of markov processes and their simulation**; **Derivatives of probability and integral functions: general theory and examples**; **Derivatives of probability measures**; **Integer programming: branch and bound methods**; **Optimization in operation of electric and energy power systems**; **Simulated annealing**; **Stochastic integer programming: continuity, stability, rates of convergence**)
discrete-time algorithms
[90B15]
(*see:* **Dynamic traffic networks**)
discrete-time algorithms
[90B15]
(*see:* **Dynamic traffic networks**)
discrete-time formulations
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
discrete Time Model
(*see:* **Integrated planning and scheduling**)
discrete-time models
(*see:* **Planning in the process industry**)
discrete time models *see:* continuous and —
Discrete-Time Optimal Control
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
discrete-time systems
[39A11, 93C55, 93D09]
(*see:* **Robust control: schur stability of polytopes of polynomials**)
discrete truncated Newton method
[90C06]
(*see:* **Large scale unconstrained optimization**)
discrete variables
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
discretely distributed stochastic programs
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)

Discretely distributed stochastic programs: descent directions and efficient points
(90C15, 90C29)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
discretization
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
discretization
[90B85]
(*see:* **Multifacility and restricted location problems**)
discretization *see:* full —; partial —; uniform time —
discretization method
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
discretization methods *see:* Semi-infinite programming: —
discretization of optimization problems
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
discretization procedure *see:* stochastic —
discretized hemivariational inequalities for nonlinear material laws
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
discretized optimal control problems
[49K20, 49M99, 90C55]
(*see:* **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
discretized SIP problem
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
discretized SIP problem *see:* nonlinear —
discriminant
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
discriminant analysis
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
discriminant functions
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
(discriminant problem) *see:* g-group classification problem —
discrimination
[90C29]
(*see:* **Multicriteria sorting methods**)
discrimination *see:* hierarchical —; multigroup hierarchical —
Disease diagnosis: optimization-based methods
(90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90, 90-08, 65K05)
disjoint
(*see:* **Bilinear programming**)
disjoint *see:* link-diverse/ —
disjoint constraints
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
disjoint path *see:* edge- —; node- —
disjunction
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)

disjunction arc
[90B35]
(*see:* **Job-shop scheduling problem**)
Disjunctions
(*see:* **Logic-based outer approximation**)
disjunctions *see:* Convex hull —
disjunctive cut principle
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
disjunctive cutting plane approach *see:* Mixed-integer nonlinear optimization: A —
disjunctive inequality
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**)
disjunctive normal form
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T15, 68T27, 68T30, 90C09, 90C10]
(*see:* **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
disjunctive normal form
[90C09, 90C10]
(*see:* **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
disjunctive OA master problem
[90C09, 90C10, 90C11]
(*see:* **MINLP: logic-based methods**)
disjunctive program *see:* facial —
Disjunctive programming
(90C09, 90C10, 90C11)
(*referred to in:* **Continuous reformulations of discrete-continuous optimization problems**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: global optimization with αBB**; **MINLP: logic-based methods**)
(*refers to:* **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: global optimization with αBB**; **MINLP: logic-based methods**; **Reformulation-linearization technique for global optimization**)
disjunctive programming
[90C10, 90C11, 90C27, 90C30, 90C33, 90C57]
(*see:* **Integer programming**; **Logic-based outer approximation**; **Optimization with equilibrium constraints: A piecewise SQP approach**)
disjunctive programming
[90C09, 90C10, 90C11, 90C27, 90C30, 90C33, 90C57]
(*see:* **Disjunctive programming**; **Integer programming**; **MINLP: logic-based methods**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Set covering, packing and partitioning problems**)
disjunctive programming *see:* Generalized —
disk graphs
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
disk graphs *see:* bisectored unit —; bounded ratio —; double —; Optimization problems in unit- —; unit- —
disk) representation *see:* geometric (or —
dispatch problem
[90B50]
(*see:* **Optimization and decision support systems**)
dispatcher *see:* load —
dispatching *see:* load —
displacement *see:* kinematically admissible —
displacement compatibility equations *see:* strain- —
displacements *see:* method of simultaneous —; method of successive —; virtual —
dissection *see:* nested —; tree —
dissimilarities
[65K05, 90C27, 90C30, 90C57, 91C15]
(*see:* **Optimization-based visualization**)
dissimilarity measure
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
distance *see:* bond —; elliptic —; euclidean —; inter-class —; intra-class —; L_p- —; Manhattan —; maximizing minimum —; maximum weighted —; method of optimal —; nonbonded —; rectilinear —; relative —
distance constrained labeling
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
distance-constrained vehicle routing problem
[90B06]
(*see:* **Vehicle routing**)
Distance dependent protein force field via linear optimization
distance of a feasible point to a solution point *see:* bounds on the —
distance function *see:* least squares —
distance functions
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
distance geometry problem
[65D18, 90B85, 90C26]
(*see:* **Global optimization in location problems**)
distance geometry problem *see:* Molecular —
distance in a graph
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
distance label
[90B10, 90C27, 90C35]
(*see:* **Maximum flow problem**; **Shortest path tree algorithms**)
distance location *see:* Single facility location: multi-objective euclidean —; Single facility location: multi-objective rectilinear —
distance location problem *see:* Euclidean —; iterative solution of the Euclidean —; rectilinear —; squared Euclidean —
distance matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
distance matrix *see:* Euclidean —; partial —
distance matrix completion problem
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
distance matrix completion problem *see:* Euclidean —
distance measure
[90B80, 90B85]
(*see:* **Warehouse location problem**)

distance scaling method
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
distances *see:* euclidean —; Manhattan —; minkowski —; Optimizing facility location with euclidean and rectilinear —; positiveness of —
distillation
[90C30, 90C90]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**; **Successive quadratic programming: applications in distillation systems**)
distillation
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in distillation systems**)
distillation *see:* reactive —
distillation column synthesis *see:* MINLP: reactive —
distillation superstructure
[90C90]
(*see:* **MINLP: reactive distillation column synthesis**)
distillation systems *see:* Successive quadratic programming: applications in —
distinguishable
[62H30, 68T10, 90C11]
(*see:* **Mixed integer classification problems**)
distinguished point
[90C26]
(*see:* **Cutting plane methods for global optimization**)
distinguished solution
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
distinguished tableau
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
distinguished variable
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
distributed computing
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
distributed computing
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
distributed game tree search algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
distributed memory parallel computer
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
distributed memory parallel machines
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
distributed optimal control problems
[49K20, 49M99, 90C55]
(*see:* **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
distributed optimal control problems *see:* Sequential quadratic programming: interior point methods for —
distributed optimization algorithms *see:* Asynchronous —
distributed state space search algorithm *see:* synchronized —
distributed stochastic programs *see:* discretely —
distributed stochastic programs: descent directions and efficient points *see:* Discretely —
distributed systems *see:* boundary flux estimation in —
distribution *see:* binomial —; Boltzmann —; Dirichlet —; Gaussian —; geometric —; hypergeometric —; incomplete knowledge of a probability —; law of normal —; Levy probability —; logconcave discrete probability —; logconcave univariate discrete probability —; multivariate gamma —; multivariate normal —; Poisson —; posterior —; prior —; probability —; quasiconcave probability —; trinomial —; Tsallis probability —; uncertainty embedded in a probability —; uniform —
distribution with algebraically decreasing tail *see:* RSM- —
distribution density *see:* steady-state —
distribution of efforts *see:* optimal —
distribution function
[60G35, 65K05]
(*see:* **Differential equations and global optimization**)
distribution function *see:* multivariate probability —; one-dimensional marginal probability —; two-dimensional marginal probability —
distribution functions *see:* gradient of multivariate —; marginal —
distribution graph *see:* alignment- —
distribution law *see:* Gauss —
distribution problems
[90C35]
(*see:* **Minimum cost flow problem**)
distribution scheduling: an MILP model *see:* Gasoline blending and —
distribution system design
[90-02]
(*see:* **Operations research models for supply chain management and design**)
distribution system design problem *see:* Production- —
distribution systems
[90-02]
(*see:* **Operations research models for supply chain management and design**)
distribution systems planning
[90C35]
(*see:* **Multicommodity flow problems**)
distributional derivatives
[60J05, 90C15]
(*see:* **Derivatives of markov processes and their simulation**; **Derivatives of probability measures**)
distributions *see:* asymptotic results for RSM- —; discrete logconcave —; Logconcavity of discrete —; Stochastic linear programs with recourse and arbitrary multivariate —
distributive lattice
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
distributive lattice *see:* free —
district *see:* double-ended crew —; single-ended crew —
districting problem *see:* political —
districts *see:* crew —

distrust region method
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
disutility functions *see:* traffic network equilibrium with travel —
dive-and-fix
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
divergence *see:* Csiszar α- —; directed —; Itakura–Saito —; Kullback–Leibler —
divergent series rule
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: subgradient optimization methods**)
divergent series step-size rule
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
divergent series steplength rule
[90C30]
(*see:* **Cost approximation algorithms**)
diverging trails
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
diverse/disjoint *see:* link- —
diversification
[68M20, 90B06, 90B35, 90B80, 90C59]
(*see:* **Flow shop scheduling problem**; **Heuristic and metaheuristic algorithms for the traveling salesman problem**; **Location routing problem**)
diversified investment decisions
[68Q25, 91B28]
(*see:* **Competitive ratio for portfolio management**)
divide-and-conquer
[90C09, 90C10, 90C26]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**; **MINLP: branch and bound global optimization algorithm**)
divided differences
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
divided differences
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
division algorithm
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
division multiplexing *see:* wavelength- —
divisor
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
divisor
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
divisor of an arrangement of hyperplanes
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
divorcing
(*see:* **Bayesian networks**)
d.m. function
[65Kxx, 90C26, 90C31, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**; **Robust global optimization**)
dM functions
[90C26]
(*see:* **D.C. programming**)
dM optimization
[90C26, 90C31]
(*see:* **D.C. programming**; **Robust global optimization**)
DNA mapping
[90C35]
(*see:* **Optimization in leveled graphs**)
dNA sequencing
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
DNA transcription element identification *see:* Mixed 0-1 linear programming approach for —
DNF
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
DNF
[90C09, 90C10]
(*see:* **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
DNF *see:* TAUT- —
DNF clauses *see:* minimal number of —
document
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
document classification
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
document classification *see:* automatic —; optimization in —
document surrogate
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
document surrogate
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
documentation
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
documents *see:* automatic classification of —; classification of large collections of —; classification of text —; Optimization in classifying text —
dog-lawed *see:* dead or —
dogleg method
[65C20, 65G20, 65G30, 65G40, 65H20, 90C06, 90C90]
(*see:* **Interval analysis: application to chemical engineering design problems**; **Large scale unconstrained optimization**)
dogleg path
[49M37, 90C30]
(*see:* **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**)
dogleg path
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
dogleg path *see:* multiple —
domain
[49J40; 49J53; 47H05; 47H04; 26B25]
(*see:* **Pseudomonotone maps: properties and applications**)

domain *see:* admissible —; feasible —; natural —
domain of a function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
domain of a function *see:* effective —
domain method *see:* fictitious —
domain of search
[90C26]
(*see:* **Global optimization using space filling**)
domains *see:* global optimization over unbounded —
dominance
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
dominance condition *see:* diagonal —
dominance criterion
[90C11, 90C31]
(*see:* **Multiparametric mixed integer linear programming**)
dominance relation
[90-XX]
(*see:* **Outranking methods**)
dominated *see:* not —
dominated family of measures
[90C15]
(*see:* **Derivatives of probability measures**)
dominated point
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
dominating set
[03B50, 05C15, 05C62, 05C69, 05C85, 68T15, 68T30, 90C27, 90C59]
(*see:* **Finite complete systems of many-valued logic algebras**; **Optimization problems in unit-disk graphs**)
dominating set *see:* connected —; finite —; independent —
domination Analysis
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
Domination analysis in combinatorial optimization
(90C27, 90C59, 68Q25, 68W40, 68R10)
(*referred to in:* **Traveling salesman problem**)
(*refers to:* **Traveling salesman problem**)
domination number
[05C15, 05C62, 05C69, 05C85, 68Q25, 68R10, 68W40, 90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Domination analysis in combinatorial optimization**; **Optimization problems in unit-disk graphs**; **Traveling salesman problem**)
domination property
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
domination ratio
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
door point *see:* trap- —
Doppler effect
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
dose (eud) *see:* equivalent uniform —
double color
[05C85]
(*see:* **Directed tree networks**)
double description method
[52B12, 68Q25]
(*see:* **Fourier–Motzkin elimination method**)
double disk graphs
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
double-ended
(*see:* **Railroad crew scheduling**)
double-ended crew district
(*see:* **Railroad crew scheduling**)
double-max duality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
double-min duality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
double pivot
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
double star
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
double sweep method *see:* Aitken —
double-well function
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
doublet
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**)
doublet
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**)
doublet *see:* sparse —
doubly connected edge list *see:* extended —
doubly nonnegative matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
doubly stochastic matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
Douglas—Rachford method
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
down penalty
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
downhill simplex method
[90C30]
(*see:* **Sequential simplex method**)
DP
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
drawing *see:* automatic graph —; graph —
Driebeck–Tomlin penalty
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
drifting *see:* population —

driven *see:* message- —
driver method *see:* bold- —
dropped negatively
[90Cxx]
(*see:* **Discontinuous optimization**)
dropped positively
[90Cxx]
(*see:* **Discontinuous optimization**)
dropping *see:* column —
dropping rule *see:* column —
drought out events
[90C30, 90C35]
(*see:* **Optimization in water resources**)
drug delivery systems *see:* Model based control for —
DS
[90C26]
(*see:* **Global optimization using space filling**)
DSD
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
DSL function
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
DSL system
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
DSPACE
[90C60]
(*see:* **Complexity classes in optimization**)
DSS *see:* multicriteria —
DTIME
[90C60]
(*see:* **Complexity classes in optimization**)
Du–Hwang minimax theorem
[90C27]
(*see:* **Steiner tree problems**)
dual
[90B85, 90C27]
(*see:* **Single facility location: circle covering problem**)
dual *see:* extended Lagrange–Slater —; formal perfect —; functional —; Lagrangian —; Mond–Weir —; primal- —; SSS∗ - —; strong —; superadditive —; surrogate —; Wolfe —
dual action *see:* Clarke —
dual algorithm *see:* generalized primal-relaxed —; primal- —
dual approach *see:* Generalized primal-relaxed —; primal-relaxed —
dual arc
[90B35]
(*see:* **Job-shop scheduling problem**)
dual ascent
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)
dual ascent
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)
dual block-angular structure
[90C15]
(*see:* **L-shaped method for two-stage stochastic programs with recourse**; **Stochastic programming: parallel factorization of structured matrices**)
dual bound-improvement
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
dual cone
[90C15, 90C22, 90C25]
(*see:* **Copositive programming**; **Stochastic programming: nonanticipativity and lagrange multipliers**)
dual cut-improvement
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
dual decomposition
[90C10, 90C15]
(*see:* **Stochastic integer programs**)
dual degenerate
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
dual degenerate basis
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
dual descent
[90C30, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**)
dual in entropy optimization *see:* unconstrained —
dual Euler–Lagrange equation
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
dual exterior point algorithm
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
dual feasibility
[68W10, 90B15, 90C05, 90C06, 90C30, 90C31]
(*see:* **Parametric linear programming: cost simplex algorithm**; **Stochastic network problems: massively parallel solution**)
dual feasible set
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
dual framework *see:* primal- —
dual information
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)
dual integral system *see:* totally —
dual interior-point methods *see:* primal- —
dual linear program
[90C30]
(*see:* **Lagrangian duality: BASICS**)
dual linear programs
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
dual matroid
[90C09, 90C10]
(*see:* **Matroids**; **Oriented matroids**)
dual method *see:* Simple recourse problem: —
dual method for the simple recourse problem
[90C06, 90C08, 90C15]
(*see:* **Simple recourse problem**)
dual methods *see:* primal- —

dual optimization problem
[90C30]
(*see:* **Lagrangian duality: BASICS**)
dual optimization problem
[90C30]
(*see:* **Lagrangian duality: BASICS**)
dual optimization problem *see:* Lagrangian —
dual (or Minty) GVI
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
dual pair
[90B35]
(*see:* **Job-shop scheduling problem**)
dual potential function
[37A35, 90C05, 90C25, 90C30]
(*see:* **Potential reduction methods for linear programming**; **Solving large scale and sparse semidefinite programs**)
dual potential function *see:* primal- —
dual potential reduction algorithm *see:* primal- —
dual price
[68Q99]
(*see:* **Branch and price: Integer programming with column generation**)
dual price increase
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
dual problem
[15A39, 49-XX, 49K05, 49K10, 49K15, 49K20, 49L99, 90-XX, 90C05, 90C29, 90C30, 93-XX]
(*see:* **Duality in optimal control with first order differential equations**; **Duality theory: monoduality in convex optimization**; **Dynamic programming: average cost per stage problems**; **Image space approach to optimization**; **Motzkin transposition theorem**; **Multi-objective optimization: lagrange duality**; **Theorems of the alternative and optimization**)
dual problem
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
dual problem *see:* construction of a —; generalized —; Lagrangian —; nonconvex —
dual problems *see:* primal and —
dual procedures
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
dual program
[90C06]
(*see:* **Saddle point theory and optimality conditions**)
dual programming problem
[90C06]
(*see:* **Saddle point theory and optimality conditions**)
dual properness
[90C25, 90C26]
(*see:* **Decomposition in global optimization**)
dual ray
[15A39, 90C05]
(*see:* **Motzkin transposition theorem**)
dual scaling
[90C25, 90C30]
(*see:* **Solving large scale and sparse semidefinite programs**)
dual-scaling algorithm
[90C25, 90C30]
(*see:* **Solving large scale and sparse semidefinite programs**)
dual scaling algorithm *see:* primal- —
dual-scalings algorithm
[90C25, 90C30]
(*see:* **Solving large scale and sparse semidefinite programs**)
dual SD problem
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
dual semi-infinite program
[90C05, 90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**; **Semi-infinite programming: methods for linear problems**)
dual semi-infinite program
[90C05, 90C34]
(*see:* **Semi-infinite programming: methods for linear problems**)
dual semidefinite program
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
dual side *see:* proof on the —
dual simplex
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
dual simplex algorithm
[90C35]
(*see:* **Generalized networks**)
dual simplex algorithms *see:* primal and —
dual simplex method *see:* lexicographic —
dual slacks
[49-XX, 90-XX, 90C05, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**; **Homogeneous selfdual methods for linear programming**)
dual solution *see:* primal- —
dual solutions *see:* exploiting the interplay between primal and —
dual space
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
dual space
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
dual SQPIP methods *see:* primal- —
dual system
[15A39, 90C05, 90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Tucker homogeneous systems of linear relations**)
dual systems *see:* homogeneous —
dual techniques
[90B80, 90C11]
(*see:* **Facility location with staircase costs**)
dual transformation
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
dual transformation *see:* canonical —
dual transformation method *see:* canonical —

dual variable
[90C30]
(*see:* **Image space approach to optimization**)
dual variables
[90C05, 90C10, 90C30, 90C35, 90C46]
(*see:* **Generalized networks**; **Integer programming duality**; **Lagrangian duality: BASICS**; **Theorems of the alternative and optimization**)
dual variational inequality problem
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
dual vector
[90C30]
(*see:* **Lagrangian duality: BASICS**)
duality
[49-XX, 90-XX, 90B85, 90C30, 90C33, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Large scale trust region problems**; **Single facility location: multi-objective euclidean distance location**)
duality
[15A39, 49-XX, 49K27, 49K40, 49M29, 90-XX, 90C05, 90C10, 90C11, 90C15, 90C22, 90C25, 90C29, 90C30, 90C31, 90C46, 93-XX]
(*see:* **Bilevel programming: optimality conditions and duality**; **Duality theory: biduality in nonconvex optimization**; **Duality theory: monoduality in convex optimization**; **Duality theory: triduality in global optimization**; **First order constraint qualifications**; **Generalized benders decomposition**; **Integer programming duality**; **Linear optimization: theorems of the alternative**; **Motzkin transposition theorem**; **Probabilistic constrained linear programming: duality theory**; **Second order constraint qualifications**; **Semidefinite programming: optimality conditions and stability**)
duality *see:* Bilevel programming: optimality conditions and —; Clarke —; double-max —; double-min —; Fenchel —; Fenchel–Moreau —; Fenchel–Rockafellar —; inference —; Integer programming —; Klötzler —; Lagrangian —; Legendre —; linear programming —; LP —; Minkowski —; Multi-objective optimization: lagrange —; perfect —; polar —; saddle Lagrange —; SDP —; Semi-infinite programming, semidefinite programming and perfect —; strong —; strong and weak —; superadditive —; superLagrangian —; surrogate —; weak —
duality and applications *see:* Robust linear programming with right-hand-side uncertainty —
duality: BASICS *see:* Lagrangian —
duality for bilevel programming
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
duality equality
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
duality equality
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
duality from the view of linear semi-infinite programming *see:* perfect —
duality gap
[49-XX, 90-XX, 90C05, 90C10, 90C22, 90C25, 90C30, 90C31, 90C34, 90C51, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**; **Duality theory: triduality in global optimization**; **Image space approach to optimization**; **Integer programming: lagrangian relaxation**; **Interior point methods for semidefinite programming**; **Lagrangian duality: BASICS**; **Semidefinite programming: optimality conditions and stability**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Theorems of the alternative and optimization**)
duality gap
[90C30]
(*see:* **Lagrangian duality: BASICS**)
duality gap *see:* relative —
Duality gaps in nonconvex optimization
(90B50, 78M50)
duality inequality
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
duality of the linear SIP problem
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
duality for M- and L-convex functions *see:* Fenchel-type —
duality of matroids
[90C09, 90C10]
(*see:* **Oriented matroids**)
duality and maximum principle
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
Duality in optimal control with first order differential equations
(49K05, 49K10, 49K15, 49K20)
(*referred to in:* **Control vector iteration CVI**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Control vector iteration CVI**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective**

optimization: interaction of design and control; **Optimal control of a flexible arm**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
duality pair *see:* Fenchel —; Legendre —
duality relation *see:* weak —
duality relations *see:* legendre —
duality result *see:* strong —; weak —
Duality for semidefinite programming
(90C30)
(*referred to in:* **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and structural optimization**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Solving large scale and sparse semidefinite programs**)
(*refers to:* **Interior point methods for semidefinite programming**; **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and structural optimization**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Solving large scale and sparse semidefinite programs**)
duality theorem
[01A99, 90C05, 90C15, 90C99]
(*see:* **Probabilistic constrained linear programming: duality theory**; **Von Neumann, John**)
duality theorem
[01A99, 90C99]
(*see:* **Von Neumann, John**)
duality theorem *see:* Clarke —; conic —; saddle —; strong —; superLagrangian —; weak —
duality theorem for linear optimization
[15A39, 90C05]
(*see:* **Motzkin transposition theorem**)
duality theory
[49M29, 90C11]
(*see:* **Generalized benders decomposition**)
Duality theory
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
duality theory *see:* Fenchel–Rockafellar —; Probabilistic constrained linear programming: —
Duality theory: biduality in nonconvex optimization
(49-XX, 90-XX, 93-XX)
(*referred to in:* **Duality theory: monoduality in convex optimization**; **Duality theory: triduality in global optimization**; **Von Neumann, John**)
(*refers to:* **Duality theory: monoduality in convex optimization**; **Duality theory: triduality in global optimization**; **History of optimization**; **Von Neumann, John**)
duality theory for entropy optimization
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
Duality theory: monoduality in convex optimization
(49-XX, 90-XX, 93-XX)
(*referred to in:* **Duality theory: biduality in nonconvex optimization**; **Duality theory: triduality in global optimization**; **Von Neumann, John**)
(*refers to:* **Duality theory: biduality in nonconvex optimization**; **Duality theory: triduality in global optimization**; **History of optimization**)
Duality theory: triduality in global optimization
(49-XX, 90-XX, 93-XX)
(*referred to in:* **Duality theory: biduality in nonconvex optimization**; **Duality theory: monoduality in convex optimization**; **Von Neumann, John**)
(*refers to:* **Duality theory: biduality in nonconvex optimization**; **Duality theory: monoduality in convex optimization**; **History of optimization**; **Von Neumann, John**)
dualization
[46A20, 52A01, 90C10, 90C30, 90C46]
(*see:* **Composite nonsmooth optimization**; **Integer programming duality**)
dualization *see:* Ioffe–Burke local —
Dubovitskii—Milyutin theorem
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
Duffin theorem
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
dummy nodes *see:* PSA with —
duopoly
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
duty-after-arrival
(*see:* **Railroad crew scheduling**)
duty-before-departure
(*see:* **Railroad crew scheduling**)
duty-period
(*see:* **Railroad crew scheduling**)
duty scheduling problems *see:* Integrated vehicle and —
duty scheduling process
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
duty time *see:* on- —
DVRP
[90B06]
(*see:* **Vehicle routing**)
Dykstra's algorithm and robust stopping criteria
(90C25, 65K05, 65G505)
dynamic
[90B06]
(*see:* **Vehicle routing**)
dynamic complementarity problem
[90C33]
(*see:* **Order complementarity**)
dynamic complementarity problem *see:* discrete —
dynamic considerations and controllability *see:* integration of —
dynamic facility location
[90B85]
(*see:* **Single facility location: multi-objective rectilinear distance location**)
dynamic games *see:* Infinite horizon control and —

dynamic load balancing
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)
dynamic load balancing technique
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
dynamic location problem
[90C35]
(*see:* **Multi-index transportation problems**)
dynamic model
[90B80, 90B85]
(*see:* **Warehouse location problem**)
dynamic network flow problem *see:* nonlinear —
dynamic network flow problems
[90C30]
(*see:* **Simplicial decomposition**)
dynamic optimization
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
Dynamic optimization
[49Jxx, 65L99, 91Axx, 93-XX]
(*see:* **Infinite horizon control and dynamic games**; **Optimization strategies for dynamic systems**)
dynamic optimization *see:* discrete decisions in —; mixed integer —
dynamic optimization problem *see:* stochastic —
dynamic programming
[34H05, 49Jxx, 49L20, 49L99, 62H30, 65L99, 68Q20, 90C10, 90C11, 90C20, 90C39, 90C40, 91Axx, 93-XX]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **Linear ordering problem**; **Neuro-dynamic programming**; **Optimal triangulations**; **Optimization strategies for dynamic systems**)
dynamic programming
[34H05, 49L20, 49L99, 49M29, 62H30, 65K10, 90C06, 90C27, 90C31, 90C39, 90C40]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Operations research and financial markets**)
dynamic programming *see:* differential —; iterative —; Multiple objective —; Neuro- —; stochastic —
dynamic programming algorithm
[49L20, 90C39, 90C40]
(*see:* **Dynamic programming: discounted problems**; **Dynamic programming: inventory control**; **Dynamic programming: stochastic shortest path problems**)
dynamic programming algorithm *see:* continuous-time equivalent of the —
Dynamic programming: average cost per stage problems
(49L99)
(*referred to in:* **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**)
(*refers to:* **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**)
Dynamic programming in clustering
(90C39, 62H30)
(*referred to in:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Optimization-based visualization**)
(*refers to:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**)
Dynamic programming: continuous-time optimal control
(49L20, 34H05, 90C39)
(*referred to in:* **Control vector iteration CVI**; **Duality in**

optimal control with first order differential equations; **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **High-order maximum principle for abnormal extremals**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Pontryagin maximum principle**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **High-order maximum principle for abnormal extremals**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Pontryagin maximum principle**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)

Dynamic programming: discounted problems
(49L20, 90C39)
(*referred to in:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**)
(*refers to:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**)

dynamic programming equation *see:* continuous-time analog of the —

dynamic programming equations *see:* recursive —

Dynamic programming: infinite horizon problems, overview
(49L20, 90C39, 90C40)
(*referred to in:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Optimization strategies for dynamic systems**)
(*refers to:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Optimization strategies for dynamic systems**)

Dynamic programming: inventory control
(49L20)
(*referred to in:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**)
(*refers to:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**;

Dynamic programming and Newton's method in unconstrained optimal control; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**)

Dynamic programming and Newton's method in unconstrained optimal control
(49M29, 65K10, 90C06)
(*referred to in:* **Automatic differentiation: calculation of Newton steps**; **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **Interval Newton methods**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Nondifferentiable optimization: Newton method**; **Nonlinear least squares: Newton-type methods**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
(*refers to:* **Automatic differentiation: calculation of Newton steps**; **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **Interval Newton methods**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Nondifferentiable optimization: Newton method**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)

Dynamic programming: optimal control applications
(93-XX)
(*referred to in:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)

dynamic programming paradigm *see:* general —

dynamic programming recursion
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)

dynamic programming recursions
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)

Dynamic programming: stochastic shortest path problems
(49L20, 90C40)

(*referred to in:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Optimization strategies for dynamic systems**)
(*refers to:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Optimization strategies for dynamic systems**)

Dynamic programming: undiscounted problems
(49L20, 90C40)
(*referred to in:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**)
(*refers to:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**)

dynamic service needs *see:* static/ —

dynamic simulation
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)

dynamic system *see:* discrete event —

dynamic systems *see:* Optimization strategies for —; Quasidifferentiable optimization: stability of —; stochastic —

dynamic systems by constructive nonlinear dynamics *see:* Robust design of —

dynamic traffic assignment
[90C35]
(*see:* **Multicommodity flow problems**)

dynamic traffic network model
[90B15]
(*see:* **Dynamic traffic networks**)

Dynamic traffic networks
(90B15)
(*referred to in:* **Auction algorithms**; **Communication network assignment problem**; **Cost approximation algorithms**; **Equilibrium networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
(*refers to:* **Auction algorithms**; **Communication network assignment problem**; **Cost approximation algorithms**; **Directed tree networks**; **Equilibrium networks**; **Evacuation networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)

dynamic travel behavior *see:* day-to-day —

dynamic two-stage stochastic programming problem
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)

dynamic two-stage stochastic programming problem
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)

dynamic vehicle routing problem
[90B06]
(*see:* **Vehicle routing**)

dynamical Bayesian networks
(*see:* **Bayesian networks**)

dynamical system
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**; **Standard quadratic optimization problems: algorithms**)

dynamical system
[05C60, 05C69, 37B25, 90C20, 90C27, 90C34, 90C35, 90C59, 91A22, 91B28]
(*see:* **Replicator dynamics in combinatorial optimization**; **Semi-infinite programming and applications in finance**)

dynamical system *see:* projected —; Variational inequalities: projected —; variational inequality problem and a projected —

dynamical systems *see:* discrete —; estimating uncertainty in —; Interval analysis for optimization of —; projected —

dynamics *see:* computational fluid —; Design optimization in computational fluid —; molecular —; process —;

replicator —; Robust design of dynamic systems by constructive nonlinear —
dynamics in combinatorial optimization *see:* Replicator —

E

$E_{D,J}$ *see:* characterization of —
E. approach *see:* Variational inequalities: F. —
EACO
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
earliness *see:* minimization of order —
easy-to-compute
(*see:* **Global optimization: functional forms**)
EasyModeler/6000
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
(EAX) *see:* edge assembly crossover —
echelon arborescence system *see:* multi- —
echelon form
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**)
echelon stock
[90B50]
(*see:* **Inventory management in supply chains**)
econometric methods
[90C26, 90C30]
(*see:* **Forecasting**)
econometric models
[90C26, 90C30]
(*see:* **Forecasting**)
econometrics
[90C26, 90C30]
(*see:* **Forecasting**)
Economic Analysis *see:* linear Programming and —
economic applications
[90C05]
(*see:* **Continuous global optimization: applications**)
economic equilibrium
[90C90, 91A65, 91B99]
(*see:* **Bilevel programming: applications**)
economic equilibrium *see:* general —
economic equilibrium model *see:* pure exchange —; pure trade —
economic growth *see:* Ramsey rule of —
Economic lot-sizing problem
(90C11, 90C39, 90C90, 90B10, 90B05, 55M05)
economic order quantity
[90B50]
(*see:* **Inventory management in supply chains**)
economic system conditions
[91B50]
(*see:* **Financial equilibrium**)
economics
[01A99]
(*see:* **Kantorovich, Leonid Vitalyevich**)
economics *see:* mathematical —
economies of scale
[90C25]
(*see:* **Concave programming**)
economy *see:* pure exchange —
economy of scale
[90C26, 90C30]
(*see:* **Reverse convex optimization**)
economy of scale
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)
edge *see:* light —; pale —; required —; shortest —; Voronoi —
edge assembly crossover (EAX)
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
edge coloring
[05C85, 90C35]
(*see:* **Directed tree networks**; **Graph coloring**)
edge coloring *see:* constrained —
edge coloring problem
[90C35]
(*see:* **Graph coloring**)
edge contraction method
(*see:* **Maximum cut problem, MAX-CUT**)
edge crossing
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
edge-directions of P *see:* covers all —
edge-disjoint path
[90-XX]
(*see:* **Survivable networks**)
edge-disjoint path
[90-XX]
(*see:* **Survivable networks**)
edge elimination ordering *see:* cobipartite neighborhood —
edge exchange
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
edge filter *see:* Sobel —
edge flips *see:* good —
edge of a graph
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
edge insertion paradigm
[68Q20]
(*see:* **Optimal triangulations**)
edge list *see:* extended doubly connected —
edge realization
[90C35]
(*see:* **Optimization in leveled graphs**)
edge set
[90C35]
(*see:* **Graph coloring**)
edge simplex method *see:* steepest —
edges
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
edges *see:* unavoidable —
edges of a digraph *see:* set of —

EDM
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
Edmundson–Madansky upper bound
[90C15]
(*see:* **Stochastic programs with recourse: upper bounds**)
effect *see:* bullwhip —; Doppler —; Maratos —; wrapping —
effective domain of a function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
effective set of a function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
effects *see:* boundary —; solvation —
effects in microclusters *see:* size —
efficiency
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
efficiency
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
efficiency *see:* computational —; local —; local strict —; local weak —; proper —; strict —; weak —
efficiency assessment *see:* comparative —
efficiency and nondomination *see:* comparison of —
efficient
[90B30, 90B50, 90C05, 90C10, 90C11, 90C29, 91B82]
(*see:* **Data envelopment analysis**; **Generalized concavity in multi-objective optimization**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**)
efficient *see:* extreme- —; input- —; nonextreme —; output- —; weakly —
efficient algorithm
[03D15, 68Q05, 68Q15, 90C60]
(*see:* **Computational complexity theory**; **Parallel computing: complexity classes**)
efficient algorithm
[90C60]
(*see:* **Computational complexity theory**)
efficient algorithms
[68Q25, 91B28]
(*see:* **Competitive ratio for portfolio management**)
efficient frontier
[91B50]
(*see:* **Financial equilibrium**)
efficient point
[90C15]
(*see:* **Stochastic programming models: random objective**)
efficient point
[90C15, 90C29, 90C30]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**; **Multi-objective optimization: lagrange duality**; **Stochastic programming models: random objective**)
efficient point *see:* (C_m^J)- —; local —; local strictly —; local weakly —; strictly —; weakly —
efficient point set
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
efficient points *see:* Discretely distributed stochastic programs: descent directions and —
efficient points sets *see:* connectedness of the —
efficient polynomially bounded polynomial time algorithm
[90C60]
(*see:* **Computational complexity theory**)
efficient portfolios *see:* frontier of —
efficient set
[90C31, 90C39]
(*see:* **Multiple objective dynamic programming**)
efficient solution
[65K05, 90B50, 90C05, 90C29, 90C30, 90C90, 91B06]
(*see:* **Bilevel programming: global optimization**; **Multi-objective optimization and decision support systems**; **Multiple objective programming support**)
efficient solution
[65K05, 90B50, 90C05, 90C15, 90C29, 91B06]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: pareto optimal solutions, properties**)
efficient solution *see:* (C_m^J)- —; nonsupported —; Pareto —; properly —; supported —; weakly —
efficient solutions
[90C27, 90C29]
(*see:* **Multi-objective combinatorial optimization**)
efficient solutions *see:* nonsupported —; set of potential —; supported —
effort *see:* principle of least —
efforts *see:* optimal distribution of —
EGOP algorithm
[90C26]
(*see:* **Generalized primal-relaxed dual approach**)
eigenvalue
[90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**)
eigenvalue
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
eigenvalue based lower bounds
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
eigenvalue bounds of interval matrices *see:* Interval analysis: —
Eigenvalue enclosures for ordinary differential equations
(49R50, 65L15, 65L60, 65G20, 65G30, 65G40)
(*referred to in:* **αBB algorithm**; **Hemivariational inequalities: eigenvalue problems**; **Interval analysis: eigenvalue bounds of interval matrices**; **Semidefinite programming and determinant maximization**)
(*refers to:* **αBB algorithm**; **Hemivariational inequalities: eigenvalue problems**; **Interval analysis: eigenvalue bounds of interval matrices**; **Semidefinite programming and determinant maximization**)
eigenvalue formulation *see:* inverse interpolation parametric —

eigenvalue of an interval matrix *see:* extreme —; interval of variation of an —
eigenvalue problem
[49J52, 90C30]
(*see:* **Hemivariational inequalities: eigenvalue problems**; **Large scale trust region problems**)
eigenvalue problem *see:* nonsmooth —
eigenvalue problems *see:* Hemivariational inequalities: —
eigenvalue proximal support vector machine *see:* generalized —
eigenvalue proximal support vector machine problem *see:* Generalized —
eigenvalue reformulation *see:* parametric —
eigenvalue theorem *see:* interlocking —
eigenvalues *see:* upper and lower bounds to —
eigenvector
[90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**)
eigenvector
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
ejection chain
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
ejection chain methods
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
Ekeland point *see:* local —
ekeland variational principle
[58C20, 58E30, 90C46, 90C48]
(*see:* **Nonsmooth analysis: weak stationarity**)
elastic bar of a truss
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
elastic boundary conditions *see:* quasidifferential —
elastic demand traffic network problems with travel demand functions
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
elastic mechanical constructions *see:* linearly —
elastic stability
[49J52, 49Q10, 74G60, 74H99, 74K99, 74Pxx, 90C90]
(*see:* **Quasidifferentiable optimization: stability of dynamic systems**)
elastic systems *see:* linearly —
elastic travel demand
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
elasticity *see:* price —
elastostatic problem involving QD-superpotentials
[49J40, 49M05, 49S05, 74G99, 74H99, 74Pxx]
(*see:* **Quasidifferentiable optimization: variational formulations**)
elastostatic problem involving QD-superpotentials *see:* convex variational inequality for an —; variational equality for an —
elastostatics
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
elastostatics with nonlinear boundary conditions
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
ELECTRE
[90-XX]
(*see:* **Outranking methods**)
ELECTRE I *see:* generalization of —
electric and energy power systems *see:* Optimization in operation of —
electric field
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
electric power system
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C30, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**; **Optimization in operation of electric and energy power systems**)
electric power system
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
electrostatic interactions
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
element *see:* cognitive —; finite —; left-paired —; left-unpaired —; mixed finite —; right-paired —; right-unpaired —
element approximation *see:* finite —; mixed finite —
element group *see:* Klein 4- —
element in a Hilbert space *see:* symmetric —
element identification *see:* Mixed 0-1 linear programming approach for DNA transcription —
element method *see:* finite —
elemental subset
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
elementary connecting path
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
elementary functions
[26A48, 26A51, 52A07]
(*see:* **Increasing and convex-along-rays functions on topological vector spaces**)
elementary functions *see:* set of —
elementary orthogonal transformations
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **QR factorization**)
elementary partial derivatives
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)
elementary transformations
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **Orthogonal triangularization**)

elements *see:* contracting matroid —; deleting matroid —
elevator problem
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
elicitation *see:* data —
eligible nonbasic variable
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
eliminating blocks of variables
[52B12, 68Q25]
(*see:* **Fourier–Motzkin elimination method**)
elimination
[13Cxx, 13Pxx, 14Qxx, 65K05, 90C30, 90Cxx]
(*see:* **Bisection global optimization methods**; **Integer programming: algebraic methods**)
elimination
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
elimination *see:* Fourier–Motzkin —; Gaussian —
elimination with backsolving *see:* Gaussian —
elimination constraints *see:* sub-tour —; subtour —
elimination graph
[65Fxx]
(*see:* **Least squares problems**)
elimination method *see:* Fourier–Motzkin —
elimination ordering *see:* cobipartite neighborhood edge —
elitism
[92B05]
(*see:* **Genetic algorithms**)
elitism
[92B05]
(*see:* **Genetic algorithms**)
ellipsoid *see:* coordinate-aligned —; maximum-volume —; minimum-volume —; Quadratic programming over an —
ellipsoid algorithm
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C06, 90C08, 90C10, 90C11, 90C20, 90C25, 90C26, 90C60]
(*see:* **Information-based complexity and information-based optimization**; **Integer programming: cutting plane algorithms**; **Linear programming: karmarkar projective algorithm**; **Quadratic knapsack**)
ellipsoid constraint
[90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**)
Ellipsoid method
(90C05)
(*refers to:* **Linear programming**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Linear programming: Klee–Minty examples**; **Volume computation for polytopes: strategies and performances**)
ellipsoid method
[90C60]
(*see:* **Complexity theory**; **Complexity theory: quadratic programming**)
ellipsoid method
[90C60]
(*see:* **Complexity theory: quadratic programming**)
ellipsoid Property
[90C05]
(*see:* **Ellipsoid method**)
ellipsoidal approximation
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
ellipsoidal approximation
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
ellipsoidal constraint
[90C25, 90C30, 90C60]
(*see:* **Complexity theory: quadratic programming**; **Solving large scale and sparse semidefinite programs**)
elliptic distance
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
elliptic distance
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
elliptic type *see:* abstract variational inequality of —
Elzinga–Hearn algorithm
[90B85, 90C27]
(*see:* **Single facility location: circle covering problem**)
EM algorithm
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
embedded family of preferences
[90C29]
(*see:* **Preference modeling**)
embedded in a probability distribution *see:* uncertainty —
embedding
[65K05, 65K10, 90C20, 90C25, 90C26, 90C27, 90C29, 90C30, 90C31, 90C33, 90C34, 90C57, 91C15]
(*see:* **Optimization-based visualization**; **Parametric optimization: embeddings, path following and singularities**)
embedding
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
embeddings, path following and singularities *see:* Parametric optimization: —
emergence of logic connectives
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
Emergency evacuation, optimization modeling
emergency facility location
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
empirical data *see:* evaluation of —
empirical measure
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
empirical method
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
empirical potential
[90C90]
(*see:* **Simulated annealing methods in protein folding**)

empirical potential
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
empirical potentials
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
empty circle *see:* largest —
empty circle problem *see:* largest —
empty neighborhood graphs
[68Q20]
(*see:* **Optimal triangulations**)
empty spaces *see:* analyzing almost —
empty tree
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
enabled parallelism *see:* AD- —
enclosing circle *see:* smallest —
enclosing-circle problem *see:* smallest —
enclosure *see:* interval —
enclosure of all azeotropes *see:* Nonlinear systems of equations: application to the —
enclosures for ordinary differential equations *see:* Eigenvalue —
encoding *see:* binary —
encyclopedia of Optimization
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
end *see:* clearly defined —
ended *see:* double- —; single- —
ended crew district *see:* double- —; single- —
endpoint of an arc
[90C35]
(*see:* **Minimum cost flow problem**)
endpoint of an arc in a directed network
[90C35]
(*see:* **Maximum flow problem**)
endpoint of a graph
[90C35]
(*see:* **Feedback set problems**)
energy
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
energy
[90C90, 91A65, 91B99]
(*see:* **Bilevel programming: applications**)
energy *see:* average rmsds by —; external —; free —; internal —; minimum potential —; molar Gibbs free —; smoothing of the potential —; total Gibbs free —
energy balance equations *see:* mass and —
energy function
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX, 90C90]
(*see:* **Nonconvex energy functions: hemivariational inequalities**; **Optimization in medical imaging**)
energy function *see:* Lennard-Jones potential —; nonconvex —; Optimization techniques for minimizing the —; potential —
energy functional *see:* generalized critical point of an —
energy functions: hemivariational inequalities *see:* Nonconvex —
energy minimization
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 65K10, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding; **Multiple minima problem in protein folding: αBB global optimization approach**)
energy minimum *see:* global —
energy model
[90B50]
(*see:* **Optimization and decision support systems**)
energy modeling
[90B50]
(*see:* **Optimization and decision support systems**)
energy and momentum balances *see:* mass —
energy power systems *see:* Optimization in operation of electric and —
energy purchase contract
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
energy purchase contract
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
engine *see:* BB search —; Newton search —; quasi-Newton search —; search —
engine routing and industrial in-plant railroads
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
engineering *see:* Archimedes and the foundations of industrial —; Bilevel programming: applications in —; control —
engineering applications
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 90C05, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Continuous global optimization: applications**)
engineering design problems *see:* Interval analysis: application to chemical —
engineering optimization
[65K05]
(*see:* **Direct global optimization algorithm**)
engineering structures
[90C26, 90C90]
(*see:* **Structural optimization: history**)
engineering via negative fitness *see:* genetic —
engines *see:* scheduling of switching —
enhance
(*see:* **Maximum cut problem, MAX-CUT**)
enhanced heuristic
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
enhanced positioning *see:* Gene clustering: A novel decomposition-based clustering approach: global optimum search with —
Enkephalin *see:* Met- —
enlargement
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
enlargement of a feasible region
[90C26]

(*see:* **Bilevel optimization: feasibility test and flexibility index**)
ensemble algorithms *see:* Protein folding: generalized- —
ensembles *see:* Generalized —
entering arc
[90C35]
(*see:* **Minimum cost flow problem**)
entering variable
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
entering variable *see:* choice of the —
enterprise-wide process networks under uncertainty *see:* Bilevel programming framework for —
entities *see:* multipurpose storage —
entropic proximal point algorithm
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
entropy
[90C25, 94A08, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**; **Jaynes' maximum entropy principle**; **Maximum entropy principle: image reconstruction**)
entropy
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**; **Jaynes' maximum entropy principle**)
entropy *see:* Algorithmic —; axiomatic derivation of —; axiomatic derivation of cross- —; axiomatic derivation of the principle of maximum —; axiomatic derivation of the principle of minimum cross- —; cross- —; ε- —; Jaynes maximum —; Kolmogorov ε- —; Kullback–Leibler cross- —; Kullback–Leibler measure of cross- —; maximum —; principle of maximum —; principle of minimum cross- —; relative —; Shannon —
entropy concentration theorem *see:* Jaynes —
entropy and game theory *see:* Maximum —
entropy and its properties *see:* Entropy optimization: shannon measure of —
entropy optimization
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
entropy optimization
[90C25, 90C51, 94A08, 94A17]
(*see:* **Entropy optimization: interior point methods**; **Maximum entropy principle: image reconstruction**)
entropy optimization *see:* algorithms for —; duality theory for —; interior point algorithms for —; path following algorithm for —; unconstrained dual in —
Entropy optimization for image reconstruction
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
entropy optimization for image reconstruction *see:* finite-dimensional models for —; vector-space models for —
Entropy optimization: interior point methods
(94A17, 90C51, 90C25)
(*referred to in:* **Entropy optimization: parameter estimation**; **Entropy optimization: shannon measure of entropy and its properties**; **Homogeneous selfdual methods for linear programming**; **Jaynes' maximum entropy principle**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Maximum entropy principle: image reconstruction**; **Potential reduction methods for linear programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Entropy optimization: parameter estimation**; **Entropy optimization: shannon measure of entropy and its properties**; **Homogeneous selfdual methods for linear programming**; **Interior point methods for semidefinite programming**; **Jaynes' maximum entropy principle**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Maximum entropy principle: image reconstruction**; **Potential reduction methods for linear programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: solution by active sets and interior point methods**)
Entropy optimization: parameter estimation
(94A17, 62F10)
(*referred to in:* **Entropy optimization: interior point methods**; **Entropy optimization: shannon measure of entropy and its properties**; **Jaynes' maximum entropy principle**; **Maximum entropy principle: image reconstruction**)
(*refers to:* **Entropy optimization: interior point methods**; **Entropy optimization: shannon measure of entropy and its properties**; **Jaynes' maximum entropy principle**; **Maximum entropy principle: image reconstruction**)
Entropy optimization: shannon measure of entropy and its properties
(94A17, 90C25)
(*referred to in:* **Entropy optimization: interior point methods**; **Entropy optimization: parameter estimation**; **Jaynes' maximum entropy principle**; **Maximum entropy principle: image reconstruction**; **Optimization in medical imaging**)
(*refers to:* **Entropy optimization: interior point methods**; **Entropy optimization: parameter estimation**; **Jaynes' maximum entropy principle**; **Maximum entropy principle: image reconstruction**; **Optimization in medical imaging**)
entropy principle *see:* Jaynes' maximum —; maximum —; minimum cross- —
entropy principle: image reconstruction *see:* Maximum —
entry-uniqueness problem
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
enumerating extreme point solutions
[90C60]
(*see:* **Complexity of degeneracy**)
enumeration
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**; **Set covering, packing and partitioning problems**)
enumeration *see:* extreme point —; implicit —; randomized —

enumeration in bilevel programming
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)
enumeration methods *see:* limited —
enumeration techniques
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
enumerative solution
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
enumerative techniques *see:* branch and bound —
envelope
[90C26]
(*see:* **Global optimization: envelope representation**)
envelope *see:* lower —; upper —
envelope representation
[90C26]
(*see:* **Global optimization: envelope representation**)
envelope representation *see:* Global optimization: —
envelopes *see:* theory of —
envelopes in optimization problems *see:* Convex —
envelopment analysis *see:* data —
envelops *see:* theory of —
environment *see:* competitive —; minimizing the degradation in quality of both water —; problem solving —; system-optimizing —; user-optimizing —
environmental systems *see:* Global optimization in the analysis and management of —
environmental systems modeling and management *see:* applications in —
environmental targets
[90C30, 90C35]
(*see:* **Optimization in water resources**)
EO
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
EOQ
[90B50]
(*see:* **Inventory management in supply chains**)
EOQ
[90B50]
(*see:* **Inventory management in supply chains**)
eor operator
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
EPA
[90C30]
(*see:* **Large scale trust region problems**)
epi-Lipschitzness *see:* compact —
epi mapping *see:* zero- —
epiconsistency
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
epiconvergence
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
epiconvergent sequence
[90C30]
(*see:* **Cost approximation algorithms**)
epidemic control
[49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX]
(*see:* **Resource allocation for epidemic control**)
epidemic control *see:* Resource allocation for —
epidemic model
[49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX]
(*see:* **Resource allocation for epidemic control**)
epidemiology
[49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX]
(*see:* **Resource allocation for epidemic control**)
epiderivative *see:* contingent —
epigraph
[49K27, 58C20, 58E30, 65K05, 90C10, 90C29, 90C30, 90C48]
(*see:* **Bisection global optimization methods**; **Integer programming: lagrangian relaxation**; **Nonsmooth analysis: Fréchet subdifferentials**; **Set-valued optimization**)
epigraph
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
epigraphs
[46A20, 52A01, 62F12, 65C05, 65K05, 90C15, 90C30, 90C31]
(*see:* **Farkas lemma: generalizations**; **Monte-Carlo simulations for stochastic optimization**)
epistemological interpretation
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
ϵ-complementary slackness
[90C30, 90C35]
(*see:* **Auction algorithms**)
ϵ-constraint method
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)
ϵ-convergence *see:* finite —
ε-entropy
[01A60, 03B30, 54C70, 68Q17]
(*see:* **Hilbert's thirteenth problem**)
ε-entropy
[01A60, 03B30, 54C70, 68Q17]
(*see:* **Hilbert's thirteenth problem**)
ε-entropy *see:* Kolmogorov —
ε-global local maximizers *see:* set of discrete —
ε-global points *see:* set of —
ϵ-minimizer
[90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**)
ε-most active points *see:* set of —
ε-reserved solution
[90C26]
(*see:* **Global optimization using space filling**)
ε-reserved solution
[90C26]
(*see:* **Global optimization using space filling**)
ϵ-scaling
[90C30, 90C35]
(*see:* **Auction algorithms**)
ε-stationary point
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
ε-steepest descent
[49J40, 49J52, 65K05, 90C30]

(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
ϵ-subdifferential
[46A20, 52A01, 65Kxx, 90C30, 90Cxx]
(*see:* **Farkas lemma: generalizations**; **Quasidifferentiable optimization: algorithms for QD functions**)
ϵ-subdifferential set
[46N10, 90-00, 90C47]
(*see:* **Nondifferentiable optimization**)
ϵ-subgradient method
[46N10, 90-00, 90C47]
(*see:* **Nondifferentiable optimization**)
EQNLP
[49M37, 65K05, 90C30]
(*see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**)
equal circles in a square
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
equal demand CMST
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
equalities *see:* single-valued boundary laws and variational —
equality *see:* duality —
Equality-constrained nonlinear programming: KKT necessary optimality conditions
(49M37, 65K05, 90C30)
(*referred to in:* **First order constraint qualifications**; **Globally convergent homotopy methods**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Redundancy in nonlinear programs**; **Relaxation in projection methods**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**; **SSC minimization algorithms**; **SSC minimization algorithms for nonsmooth and stochastic optimization**)
(*refers to:* **First order constraint qualifications**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Relaxation in projection methods**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**; **SSC minimization algorithms**; **SSC minimization algorithms for nonsmooth and stochastic optimization**)
equality-constrained nonlinear programming problem
[49M37, 65K05, 90C30]
(*see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**)
equality-constrained optimization
[49M37, 65K05, 90C26, 90C30, 90C39]
(*see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Second order optimality conditions for nonlinear optimization**)
equality-constrained optimization
[49M37, 65K05, 90C30]
(*see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**)
equality constraint *see:* implicit —
equality constraints
[41A10, 46N10, 47N10, 49K27, 93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**; **High-order necessary conditions for optimality for abnormal points**)
equality constraints *see:* feasibility of —
equality for an elastostatic problem involving QD-superpotentials *see:* variational —
equality of phase compositions
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
equality relation *see:* linear —
equality relaxation *see:* outer approximation with —
equality relaxation and augmented penalty *see:* outer approximation with —
equalization *see:* quality —
equation *see:* ASOG —; balance —; Bellman's —; characteristic —; conservation of flow —; continuous-time analog of the dynamic programming —; continuous-time Riccati —; corrected seminormal —; derivation of the Hamilton–Jacobi–Bellman —; difference —; differential —; diffusion —; dual Euler–Lagrange —; equilibrium —; Euler —; Euler–Lagrange —; extended adjoint —; generalized —; geometrical —; governing —; Hamilton–Jacobi —; Hamilton–Jacobi–Bellman —; Hammerstein —; HJB —; Knizhnik–Zamolodchikov differential —; Kremser —; Lagrange —; Langevin —; linear —; linear interval —; nonlinear diffusion —; normal —; NRTL —; one-dimensional nonlinear —; Poisson —; reaction —; regular solution of the Wilson —; replicator —; Riccati —; Schroedinger —; secant —; Smoluchowski–Kramers —; solution of the Hamilton–Jacobi–Bellman —; stochastic differential —; storage —; sufficiency theorem for the Hamilton–Jacobi–Bellman —; UNIFAC —; UNIQUAC —; Wilson —
equation-based
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
equation based approaches
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
equation method *see:* diffusion —
equation models *see:* simultaneous —
equation oriented approach
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
equation-solving *see:* iterative linear —
equation of state
[65H20, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**)
equations *see:* algebraic —; bounds for linear —; conservation of flow —; differential —; differential and algebraic —; Diophantine —; Duality in optimal control with first order differential —; Eigenvalue enclosures for ordinary

differential —; error bound for approximate solutions of nonlinear systems of —; Euler —; existence of solutions of nonlinear systems of —; extremal —; First order partial differential —; generalized state —; Global optimization methods for systems of nonlinear —; Gröbner bases for polynomial —; ideal and nonideal phase equilibrium —; integral —; Interval analysis: differential —; Interval analysis: systems of nonlinear —; Kantorovich–Karush–Kuhn–Tucker —; KT —; Kuhn–Tucker —; Lagrange —; linear —; linear algebraic —; linear systems of —; mass and energy balance —; node flow balance —; nonlinear —; nonlinear system of —; nonlinear systems of —; nonsmooth —; normal —; ordinary differential —; overdetermined system of nonlinear —; Overdetermined systems of linear —; partial differential —; phase equilibrium —; polynomial —; polynomial system of —; recursive dynamic programming —; replicator —; resolvent —; rigorous bound for solutions of nonlinear systems of —; rotation in the solution of —; selection —; sensitivity —; solvability of —; state —; strain-displacement compatibility —; stress equilibrium —; Symmetric systems of linear —; system of —; systems of nonlinear —; test for the existence of solutions of —; underdetermined system of nonlinear —; uniqueness of solutions of nonlinear systems of —; well-determined system of nonlinear —; Wiener–Hopf —

equations: application to the enclosure of all azeotropes *see:* Nonlinear systems of —

equations and global optimization *see:* Differential —

equations and linear least squares *see:* Abaffi–Broyden–Spedicato algorithms for linear —; ABS algorithms for linear —

equations for material flows *see:* balance —

equiangularity
[68Q20]
(*see:* **Optimal triangulations**)

equilibration
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)

equilibration algorithm
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)

equilibria *see:* reaction —

equilibrium
[49-XX, 49Jxx, 90-XX, 90B80, 90B85, 90C15, 90C26, 90C33, 90Cxx, 91Axx, 91Bxx, 93-XX]
(*see:* **Duality theory: triduality in global optimization**; **Facility location with externalities**; **Infinite horizon control and dynamic games**; **Stochastic bilevel programs**)

equilibrium
[49M37, 90C26, 91A10]
(*see:* **Bilevel programming**)

equilibrium *see:* asymptotical stability at an —; communication —; Cooperative —; Cournot–Nash oligopolistic —; economic —; feedback Nash —; Financial —; fixed demand traffic network —; general —; general economic —; Global optimization in phase and chemical reaction —; memory strategy —; memory strategy Nash —; migration —; multimodal traffic network —; multiphase chemical —; Nash —; network —; Noncooperative —; Oligopolistic market —; open-loop Nash —; Optimality criteria for multiphase chemical —; Overtaking —; partial —; phase —; pure exchange —; spatial Cournot–Nash —; spatial price —; stability at an —; Stackelberg–Nash —; Stackelberg–Nash–Cournot —; symmetric network —; thermal —; traffic network —; Walrasian price —

equilibrium analysis
[90-01, 90B30, 90B50, 91B32, 91B52, 91B74]
(*see:* **Bilevel programming in management**)

equilibrium of an assignment and a set of prices
[90C30, 90C35]
(*see:* **Auction algorithms**)

equilibrium of an assignment and a set of prices *see:* almost at —

equilibrium conditions *see:* market —

equilibrium constraints *see:* mathematical program with —; mathematical program with affine —

equilibrium constraints: A piecewise SQP approach *see:* Optimization with —

equilibrium equation
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)

equilibrium equations *see:* ideal and nonideal phase —; phase —; stress —

equilibrium model *see:* Cournot–Nash oligopolistic —; migration network —; multi-sector multi-instrument financial —; multimodal traffic network —; partial —; perfectly competitive —; pure exchange economic —; pure trade economic —

Equilibrium networks
(90C30)
(*referred to in:* **Auction algorithms**; **Communication network assignment problem**; **Dynamic traffic networks**; **Financial equilibrium**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Oligopolistic market equilibrium**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Spatial price equilibrium**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**; **Walrasian price equilibrium**)
(*refers to:* **Auction algorithms**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Evacuation networks**; **Financial equilibrium**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Oligopolistic market equilibrium**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Spatial price equilibrium**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**; **Walrasian price equilibrium**)

equilibrium point
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)

equilibrium point
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)
equilibrium problem *see:* chemical —; network structure of the spatial price —; phase —; spatial price —; standard traffic —
equilibrium problems
[65K10, 90C31, 90C33]
(*see:* **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**)
equilibrium problems
[46N10, 49J40, 90C26, 90C33]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Linear complementarity problem**)
equilibrium problems *see:* Generalized monotonicity: applications to variational inequalities and —; Global optimization: application to phase —
equilibrium process
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
equilibrium search *see:* Global —
equilibrium solutions
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
equilibrium solutions *see:* verifying —
equilibrium stress
[51K05, 52C25, 68Q25, 68U05, 90C22, 90C35]
(*see:* Graph realization via semidefinite programming)
equilibrium with travel disutility functions *see:* traffic network —
equivalence
[90C29]
(*see:* **Preference modeling**)
equivalence *see:* computational —; logical —; problem —
Equivalence between nonlinear complementarity problem and fixed point problem
(90C33)
(*referred to in:* **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Linear complementarity problem**; **Order complementarity**; **Principal pivoting methods for linear complementarity problems**; **Topological methods in complementarity theory**)
(*refers to:* **Convex-simplex algorithm**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Order complementarity**; **Parametric linear programming: cost simplex algorithm**; **Principal pivoting methods for linear complementarity problems**; **Sequential simplex method**; **Topological methods in complementarity theory**)
equivalence classes of problems
[90C60]
(*see:* **Computational complexity theory**)
equivalence closure of a relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
equivalence closure of a relation *see:* local —
equivalence to minmax, concave programs *see:* Bilevel linear programming: complexity —
equivalence relation
[26E25, 49J52, 52A27, 90C99]
(*see:* **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
equivalence relation *see:* local —
equivalence of SIPs
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
equivalence theorem
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
equivalences
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
equivalences
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
equivalent
[05C15, 05C62, 05C69, 05C85, 13Cxx, 13Pxx, 14Qxx, 90C27, 90C59, 90Cxx]
(*see:* **Integer programming: algebraic methods**; **Optimization problems in unit-disk graphs**)
equivalent *see:* computational —
equivalent classes
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
equivalent cost vectors
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
equivalent of the dynamic programming algorithm *see:* continuous-time —
equivalent forms *see:* Quadratic integer programming: complexity and —
equivalent model *see:* deterministic —
equivalent primal SD problem
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
equivalent problem *see:* deterministic —
equivalent semi-infinite programs *see:* computationally —
equivalent uniform dose (eud)
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
equivariant Morse lemma
[58E05, 90C30]
(*see:* **Topology of global optimization**)
EREW PRAM
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
error *see:* estimation —; feasibility —; linearization —; minimizing the overall classification —; round-off —; sum of squared —
error bound
[90C30, 90C33]
(*see:* **Implicit lagrangian**)
error bound *see:* global —

error bound for approximate solutions of nonlinear systems of equations
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)
error capacity *see:* shannon zero- —
error estimates for AD
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)
error estimation *see:* Automatic differentiation: introduction, history and rounding —
error minimization
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
errors are under control *see:* rounding —
errors-in-variables model
[65Fxx]
(*see:* **Least squares problems**)
Esau–Williams algorithm
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
escape step
[90C20]
(*see:* **Standard quadratic optimization problems: algorithms**)
escape step
[90C20]
(*see:* **Standard quadratic optimization problems: algorithms**)
escherichia coli
[90C08]
(*see:* **Mixed 0-1 linear programming approach for DNA transcription element identification**)
essential
[90C26, 90C31]
(*see:* **Robust global optimization**)
essential optimal solution
[90C26, 90C31]
(*see:* **Robust global optimization**)
essential polyhedron
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
essential supremum
(*see:* **Stochastic optimal stopping: problem formulations**)
essentially active index set
[49J52, 49Q10, 74G60, 74H99, 74K99, 74Pxx, 90C90]
(*see:* **Quasidifferentiable optimization: stability of dynamic systems**)
established node
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
establishing solution quality
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
estimate
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
estimate *see:* best —; difference —; Matula —; maximum likelihood —; probabilistic —; pseudocost —
estimate of the spot rate *see:* τ- —
estimate using pseudocosts *see:* best —
estimate using pseudoshadow prices *see:* best —
estimates *see:* kernel —; parameter —
estimates for AD *see:* error —
estimates for parametric NLPS *see:* Bounds and solution vector —
Estimating data for multicriteria decision making problems: optimization techniques
(90C29)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
estimating the spot rate for bonds with constant maturities
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
estimating uncertainty in dynamical systems
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
estimation
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
estimation *see:* absolute —; Automatic differentiation: introduction, history and rounding error —; Bayesian parameter —; covariance matrix —; Entropy optimization: parameter —; gradient —; maximum likelihood —; parameter —; τ-programmed problem of spot rate —

estimation of 1D-diffusion fluxes
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
estimation of diffusion flux models
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
estimation in distributed systems *see:* boundary flux —
estimation error
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
estimation of kinetic coefficients
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
estimation in lumped systems *see:* reaction flux —
estimation of model parameters
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
estimation and optimization of nonlinear problems *see:* Simultaneous —
estimation problem *see:* ℓ_1 —; sinusoidal parameter —
estimation procedure *see:* sequential —
estimation of reaction rates and stoichiometry
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
estimation risk
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
estimation of subdifferentials
[26E25, 49J52, 52A27, 90C99]
(*see:* **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
estimation of utility functions
[90C29]
(*see:* **Multicriteria sorting methods**)
estimator *see:* Huber M- —; robust —
estimators *see:* James–Stein —; Stein —
η-convex
[90C26]
(*see:* **Invexity and its applications**)
η-convex
[90C26]
(*see:* **Invexity and its applications**)
η-pseudoconvex
[90C26]
(*see:* **Invexity and its applications**)
euclidean distance
[90B85]
(*see:* **Bayesian networks**; **Single facility location: multi-objective rectilinear distance location**)
euclidean distance location *see:* Single facility location: multi-objective —
Euclidean distance location problem
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
Euclidean distance location problem *see:* iterative solution of the —; squared —
Euclidean distance matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
Euclidean distance matrix completion problem
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
Euclidean distance matrix completion problem
(*see:* **Semidefinite programming and the sensor network localization problem, SNLP**)
euclidean distances
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
Euclidean norm *see:* A-weighted —
euclidean and rectilinear distances *see:* Optimizing facility location with —
Euclidean representation
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
Euclidean representation *see:* unidimensional —
Euclidean space *see:* triangulation of —
Euclidean Steiner ratio
[90C27]
(*see:* **Steiner tree problems**)
euclidean TSP
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
(eud) *see:* equivalent uniform dose —
Euler equation
[90C30]
(*see:* **Image space approach to optimization**)
Euler equation
[90C30]
(*see:* **Image space approach to optimization**)
Euler equations
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
Euler formula
[58E05, 90C30]
(*see:* **Topology of global optimization**)
Euler–Lagrange equation
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
Euler–Lagrange equation *see:* dual —
Euler method
[90B15]
(*see:* **Dynamic traffic networks**)
Eulerian graph
[90B06]
(*see:* **Vehicle routing**)

european Journal of Operational Research
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
Evacuation networks
(90B15)
(*referred to in:* **Auction algorithms**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
evacuation, optimization modeling *see:* Emergency —
evaluation
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
evaluation *see:* asymptotic case of integral —; multiple criteria —; performance —; policy —; software development and —
evaluation in classical logic
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
evaluation depth
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
evaluation of empirical data
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
evaluation in multiple-valued logic
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
evaluation of objective functions and/or derivatives
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
evaluation problem
[90C29]
(*see:* **Multiple objective programming support**)
evaluator *see:* position —
even sequence
[05C85]
(*see:* **Directed tree networks**)
event dynamic system *see:* discrete —
events *see:* decision making under extreme —; drought out —
eventually exact
[90C25, 90C26]
(*see:* **Decomposition in global optimization**)
Everett generalized Lagrange multiplier approach
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
evidence *see:* expected weight of —; likelihood —
evolution
[92B05]
(*see:* **Genetic algorithms**; **Genetic algorithms for protein structure prediction**)
evolution strategy
[92B05]
(*see:* **Genetic algorithms**)
evolution strategy
[92B05]
(*see:* **Genetic algorithms**)
evolutionary algorithm
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
evolutionary algorithm
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
evolutionary algorithms
[05-04, 68T20, 68T99, 90C27, 90C59]
(*see:* **Evolutionary algorithms in combinatorial optimization**; **Metaheuristics**)
Evolutionary algorithms in combinatorial optimization
(90C27, 05-04)
(*referred to in:* **Bayesian networks**; **Beam selection in radiotherapy treatment design**; **Combinatorial matrix analysis**; **Combinatorial optimization algorithms in resource allocation problems**; **Combinatorial optimization games**; **Fractional combinatorial optimization**; **Multi-objective combinatorial optimization**; **Optimization-based visualization**; **Replicator dynamics in combinatorial optimization**; **Simulated annealing**; **Traveling salesman problem**)
(*refers to:* **Combinatorial matrix analysis**; **Combinatorial optimization games**; **Fractional combinatorial optimization**; **Genetic algorithms**; **Multi-objective combinatorial optimization**; **Neural networks for combinatorial optimization**; **Replicator dynamics in combinatorial optimization**)
evolutionary game theory
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
evolutionary game theory
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
evolutionary methods
(*see:* **Bayesian networks**)
evolutionary network
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
evolutionary strategies
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
Evtushenko method
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
EW
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
ex-ante (risk averse, anticipative) decision
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
ex-ante (risk averse, anticipative) decision
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
ex-post (risk prone, adaptive) decision
[90C15]

(*see:* **Two-stage stochastic programming: quasigradient method**)
ex-post (risk prone, adaptive) decision
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
exact *see:* eventually —
exact algorithm
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
exact algorithm for solving CAP on trees
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
exact algorithms
[68Q25, 68R10, 68W40, 90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Domination analysis in combinatorial optimization**; **Heuristic and metaheuristic algorithms for the traveling salesman problem**; **Traveling salesman problem**)
exact algorithms
[68Q25, 90B80, 90C05, 90C06, 90C08, 90C10, 90C11, 90C27, 94C15]
(*see:* **Communication network assignment problem**; **Graph planarization**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**)
exact continuous
[90C10, 90C11, 90C27, 90C33]
(*see:* **Continuous reformulations of discrete-continuous optimization problems**)
exact L_∞-penalty function
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
exact methods
[90B06]
(*see:* **Vehicle routing**)
exact methods for solving vehicle routing problems
[90B06]
(*see:* **Vehicle routing**)
exact penalty
[49K35, 49M27, 65K10, 90C25]
(*see:* **Convex max-functions**)
exact penalty function
[65L99, 90C15, 90C25, 90C29, 90C30, 90C31, 93-XX]
(*see:* **Bilevel programming: optimality conditions and duality**; **Optimization strategies for dynamic systems**; **Stochastic quasigradient methods in minimax problems**)
exact penalty function
[90C15, 90C30, 90Cxx]
(*see:* **Large scale trust region problems**; **Quasidifferentiable optimization: exact penalty methods**; **Stochastic quasigradient methods in minimax problems**)
exact penalty function *see:* l_1 —
exact penalty function approach *see:* continuously differentiable —
exact penalty function based algorithm
[90C30]
(*see:* **Large scale trust region problems**)
Exact penalty method
[90Cxx]
(*see:* **Discontinuous optimization**)
exact penalty methods *see:* Quasidifferentiable optimization: —
exact penalty parameter
[90Cxx]
(*see:* **Quasidifferentiable optimization: exact penalty methods**)
exact procedure
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
exact sampling
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
exact solution methods
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
exactly
(*see:* **LP strategy for interval-Newton method in deterministic global optimization**)
exactly or approximately
[90B10]
(*see:* **Piecewise linear network flow problems**)
example
[90C06, 90C10, 90C11, 90C27, 90C30, 90C57, 90C90, 94C15]
(*see:* **Graph planarization**; **Modeling difficult optimization problems**)
example *see:* optimization computer implementation —
example of a trim-loss problem *see:* numerical —
examples *see:* Derivatives of probability and integral functions: general theory and —; Klee–Minty —; Linear programming: Klee–Minty —; unclassifiable —
examples from financial decision making *see:* Preference disaggregation approach: basic features —
examples of quasidifferentiable functions
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
exceptional family
[90C33]
(*see:* **Topological methods in complementarity theory**)
excess demand function *see:* aggregate —
excess function
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
excess of a network node
[90C35]
(*see:* **Minimum cost flow problem**)
excess part
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in distillation systems**)
excess width
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
exchange *see:* edge —; mass and heat —; modeling mass —; pure —
exchange economic equilibrium model *see:* pure —
exchange economy *see:* pure —
exchange equilibrium *see:* pure —
exchange heuristic *see:* min- —

exchange matches *see:* mass —
exchange move
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
exchange neighborhood
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
exchange neighborhood *see:* k- —; pair- —
exchange network *see:* heat and mass —
exchange networks *see:* Flexible mass —
exchange pivot
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
exchange procedures *see:* local —
exchange property
[90C09, 90C10]
(*see:* **Matroids**)
exchanger *see:* heat —; mass —
exchanger network *see:* mass —; mass and heat —
exchanger network superstructure *see:* heat —
exchanger network synthesis *see:* heat —; MINLP: heat —; Mixed integer linear programming: heat —
exchanger network synthesis without decomposition *see:* heat —
exchanger networks *see:* Global optimization of heat —; heat —; MINLP: mass and heat —; Mixed integer linear programming: mass and heat —
exclusion region
[68Q20]
(*see:* **Optimal triangulations**)
exclusive OR
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
exclusively
[90C10, 90C11, 90C27, 90C33]
(*see:* **Continuous reformulations of discrete-continuous optimization problems**)
execution of a Turing machine
[90C60]
(*see:* **Complexity theory**)
exhaustion principle
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
exhaustive
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
exhaustive sequential coloring *see:* frequency —; requirement —
existence
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
existence of classes axiom
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
existence property
[90C33]
(*see:* **Topological methods in complementarity theory**)
existence-proving properties of interval Newton methods
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)
existence of sets axioms
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
existence of solutions of equations *see:* test for the —
existence of solutions of nonlinear systems of equations
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)
existence and uniqueness *see:* Variational inequalities: geometric interpretation —
exogenous inflow *see:* hydrological —
exogenous inflow and demand *see:* water resources planning under uncertainty on hydrological —
expanded transshipment model
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
expanding *see:* algorithm greedy- —
expanding grid *see:* menace of the —
expansion
[90C30]
(*see:* **Sequential simplex method**)
expansion *see:* first order Taylor series —
expansion coefficient
[90C30]
(*see:* **Sequential simplex method**)
expansion of a matrix *see:* standard determinant —
expansion operations
[90C30]
(*see:* **Sequential simplex method**)
expectation *see:* bounding the —
expectation constraint *see:* conditional —
expectation and decision
[90C26, 90C30]
(*see:* **Forecasting**)
expectation functions *see:* sample and —
expectation of an indicator function
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
expectation-maximization
(*see:* **Bayesian networks**)
expectation maximization
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
expectation-maximization algorithm
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
expectation-maximization interval
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
expectations *see:* Static stochastic programming models: conditional —
expected number of pivot steps
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
expected number of shadow-vertices
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)

expected power consumption
[68M12, 90B18, 90C11, 90C30]
(*see:* **Optimization in ad hoc networks**)
expected recourse
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)
expected recourse function
[90C10, 90C15]
(*see:* **Stochastic integer programs**)
expected Savings
[90B15]
(*see:* **Evacuation networks**)
expected value function
[90C11, 90C15]
(*see:* **Stochastic programming with simple integer recourse**)
expected value of perfect information
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)
expected weight of evidence
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
experiment
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
experiment design
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
experiment design
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
experimental analysis *see:* model-based —
experimental design
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
experimental design *see:* optimal —; sequential —
expert system
[90C26, 90C30]
(*see:* **Forecasting**)
expert systems
[90C26, 90C30]
(*see:* **Forecasting**)
exploiting the interplay between primal and dual solutions
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
exploration of minimax trees *see:* parallelizing the —
exploratory statistical analysis
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
exponential
[34E05, 90C27]
(*see:* **Asymptotic properties of random multidimensional assignment problem**)
exponential algorithm
[90C60]
(*see:* **Computational complexity theory**)
exponential algorithm
[90C60]
(*see:* **Computational complexity theory**)
exponential behavior
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
exponential complexity
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
exponential function
[90C60]
(*see:* **Complexity classes in optimization**)
exponential function *see:* parabolic- —
exponential scale
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
exponential smoothing
[90C26, 90C30]
(*see:* **Forecasting**)
exponential smoothing
[90C26, 90C30]
(*see:* **Forecasting**)
exponential time algorithm
[90C60]
(*see:* **Computational complexity theory**)
exponential transformation
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
exponentially space-bounded Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
exponentially time-bounded Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
export model
[90C35]
(*see:* **Multicommodity flow problems**)
exposure time *see:* radiation —
express delivery problem
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
express shipment delivery
[90C35]
(*see:* **Multicommodity flow problems**)
expression parsing
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
expressions *see:* algebraic —
EXSPACE
[90C60]
(*see:* **Complexity classes in optimization**)
extended adjoint equation
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)

extended alignment graph
[90C35]
(*see:* **Optimization in leveled graphs**)
extended canonical function space
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
extended cutting plane
[49M37, 90C11]
(*see:* **MINLP: outer approximation algorithm**; **Mixed integer nonlinear programming**)
Extended cutting plane algorithm
(90C11, 90C26)
(*referred to in:* **Chemical process planning**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Quadratic assignment problem**)
extended cutting plane method
[90C11]
(*see:* **MINLP: outer approximation algorithm**)
extended doubly connected edge list
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
extended Extremal Principle
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
extended group relaxations
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
extended Lagrange–Slater dual
[90C05, 90C25, 90C30, 90C34]
(*see:* **Duality for semidefinite programming**; **Semi-infinite programming, semidefinite programming and perfect duality**)
extended linear complementarity problem
[90C33]
(*see:* **Linear complementarity problem**)
extended linear programming problems
[90C25, 90C33, 90C55]
(*see:* **Splitting method for linear complementarity problems**)
extended matrix
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**)
extended quadratic programming problem
[90C25, 90C33, 90C55]
(*see:* **Splitting method for linear complementarity problems**)
extended real-valued CNSO
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
extended set of Lagrange multipliers
[49M37, 65K05, 90C30]
(*see:* **Inequality-constrained nonlinear optimization**)
extended support problems method
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
extension *see:* coloring —; conic —; interval —; Lovász —; mean value —; natural interval —; path —; uniform —; united —
Extension of the fundamental theorem of linear programming
(90C05)
extension set
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
extension theorem *see:* Hahn–Banach linear —
extensionality *see:* axiom of —
extensionality axiom
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
extensionality convention
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
extensive form
[90C15]
(*see:* **L-shaped method for two-stage stochastic programs with recourse**)
exterior point algorithm *see:* dual —
exterior point method
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
exterior point method
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
exteriority
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
external deviation
[62H30, 68T10, 90C05]
(*see:* **Linear programming models for classification**)
external energy
[90C90]
(*see:* **Optimization in medical imaging**)
externalities
[90B80, 90B85, 90Cxx, 91Axx, 91Bxx]
(*see:* **Facility location with externalities**)
externalities *see:* Facility location with —
extra-gradient algorithm
[90C30]
(*see:* **Cost approximation algorithms**)
extra-urban transit planning
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
extraction *see:* feature —

extrapolation
[90C26, 90C30]
(*see:* **Forecasting**)
extrapolation
[90C26, 90C30]
(*see:* **Forecasting**)
extrapolation *see:* subjective curve fitting and —
extrapolation methods
[90C26, 90C30]
(*see:* **Forecasting**)
extraTime
(*see:* **Medium-term scheduling of batch processes**)
extremal
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
extremal *see:* abnormal —; abnormal weak —; locally —; normal —; weak —
extremal basis
[49K35, 49M27, 65K10, 90C25]
(*see:* **Convex max-functions**)
extremal basis method
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
extremal basis method
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
extremal equations
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
extremal global optimization *see:* multi- —
extremal Principle
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
Extremal Principle *see:* extended —
extremal ray
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
extremal set *see:* cover the —
extremality *see:* multi- —
extremals *see:* High-order maximum principle for abnormal —
extreme-efficient
[90B30, 90B50, 90C05, 91B82]
(*see:* **Data envelopment analysis**)
extreme eigenvalue of an interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
extreme events *see:* decision making under —
extreme face
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
extreme feasible solution
[90C60]
(*see:* **Complexity of degeneracy**)
extreme point
[90C05, 90C30]
(*see:* **Convex-simplex algorithm**; **Frank–Wolfe algorithm**; **Krein–Milman theorem**; **Simplicial decomposition**)
extreme point enumeration
[90C60]
(*see:* **Complexity of degeneracy**)
extreme point mathematical program
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
extreme point ranking
[90C60]
(*see:* **Complexity of degeneracy**)
extreme point solution
[90C60]
(*see:* **Complexity of degeneracy**)
extreme point solutions *see:* enumerating —
extreme points *see:* convex combination of the —; ranking —
extremum principles
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
extremum problems with probability functionals *see:* Approximation of —
Extremum problems with probability functions: kernel type solution methods
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity**

theory; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
eye-view *see:* beam's- —

F

F *see:* ADOL- —
f-attentive convergence
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
F-complete language
[90C60]
(*see:* **Complexity classes in optimization**)
F-completeness
[90C60]
(*see:* **Complexity classes in optimization**)
F. E. approach *see:* Variational inequalities: —
F-hard language
[90C60]
(*see:* **Complexity classes in optimization**)
F-hardness
[90C60]
(*see:* **Complexity classes in optimization**)
face
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements in optimization**)
face *see:* extreme —; k- —; optimal —; points on the same —; S- —
face of an arrangement
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements in optimization**)
face of a polyhedral subdivision
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
faces *see:* incident —
facet
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
facet
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
facets
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
facial disjunctive program
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
facial disjunctive program
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
facial program
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
facilities *see:* controlled recharge —; groundwater pumping —; residents of special —; surface water pumping —
facilities layout
[90B80]
(*see:* **Facilities layout problems**)
Facilities layout problems
(90B80)
(*referred to in:* **Quadratic assignment problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Complexity classes in optimization**; **Complexity theory**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Genetic algorithms**; **Global optimization in Weber's problem with attraction and repulsion**; **Integer programming: branch and bound methods**; **Integer programming: lagrangian relaxation**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Quadratic assignment problem**; **Resource allocation for epidemic control**; **Simulated annealing methods in protein folding**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
facility layout
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
facility location
[90-02, 90B05, 90B06, 90B80, 90B85, 90C11, 90Cxx, 91Axx, 91Bxx]
(*see:* **Facility location with externalities**; **Global supply chain models**; **Operations research models for supply chain management and design**; **Stochastic transportation and location problems**; **Warehouse location problem**)
facility location
[05C05, 05C85, 68Q25, 90B80, 90B85, 90C08, 90C11, 90C26, 90C27, 90C57, 90C59, 90C90]
(*see:* **Bottleneck steiner tree problems**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **Quadratic assignment problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective**

rectilinear distance location; **Voronoi diagrams in facility location**)
facility location *see:* Competitive —; dynamic —; emergency —; multi-objective —; multiple- —; single —; Voronoi diagrams in —
facility location-allocation
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
facility location-allocation
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
facility location-allocation *see:* MINLP: application in —
facility location: circle covering problem *see:* Single —
facility location with euclidean and rectilinear distances *see:* Optimizing —
Facility location with externalities
(90B80, 90B85, 91Bxx, 90Cxx, 91Axx)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
facility location model
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)
facility location model *see:* spatial competition —; Stochastic —
facility location: multi-objective euclidean distance location *see:* Single —
facility location: multi-objective rectilinear distance location *see:* Single —
facility location problem
[90B80, 90C10, 90C11, 90C27, 90C57]
(*see:* **Facility location with staircase costs**; **Integer programming**)
facility location problem *see:* uncapacitated —
facility location problems
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)
Facility location problems with spatial interaction
(90B80, 90C10)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Facility location with externalities**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
facility location problems with staircase costs *see:* convex piecewise linearization in —; heuristics of —; linearization in —; solution of —
Facility location with staircase costs
(90B80, 90C11)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and**

rectilinear distances; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
facility location with staircase costs
[90B80, 90C11]
(*see:* **Facility location with staircase costs**)
facility planning and scheduling
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
facility problem in OR *see:* single- —
factor *see:* constraint- —; cycle —; discount —; fading —; human rationality —; Q- —; search overhead —
factor programming *see:* variable —
factorial HMM
(*see:* **Bayesian networks**)
factorization *see:* Bunch and Parlett —; Cholesky —; classical LU —; complete orthogonal —; matrix —; modifying matrix —; orthogonal —; parallel matrix —; qR —; rank revealing —; rank revealing QR —; rank revealing URV —; structured matrix —
factorization with column-pivoting *see:* QR —
factorization of structured matrices *see:* Stochastic programming: parallel —
factorization using Householder transformations *see:* QR —
factorized quasi-Newton methods
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
factors *see:* bound- —; normalized structure —
fading factor
(*see:* **Bayesian networks**)
failure of the alpha-beta algorithm *see:* high —; low —
failure risk *see:* business —
fair objective function
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
falsification
[34-xx, 34Bxx, 34Lxx, 93E24]
(*see:* **Complexity and large-scale least squares problems**)
families of Pi-algebras
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
families of the Pinkava logic algebras *see:* many-valued —
family *see:* Broyden —; exceptional —; finite nested —; laminar —; two-parameter CG —
family of measures *see:* dominated —; weakly L_1 (v)-differentiable —
family of methods *see:* Broyden —
family of methods and the BFGS update *see:* Broyden —
family of preferences *see:* embedded —
family of probability measures *see:* regular —
family of sets *see:* pseudoconnected —
family of triangulations *see:* regular —
fan *see:* Gröbner —; secondary —
FAP
[05-XX]
(*see:* **Frequency assignment problem**)
far-from-native conformations *see:* discarding —
Farkas lemma
(15A39, 90C05)
(*referred to in:* **Farkas lemma: generalizations**; **Fourier–Motzkin elimination method**; **Fractional programming**; **Global optimization: envelope representation**; **Gröbner bases for polynomial equations**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Least-index anticycling rules**; **Linear optimization: theorems of the alternative**; **Linear programming**; **Motzkin transposition theorem**; **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**)
(*refers to:* **Farkas lemma: generalizations**; **Linear optimization: theorems of the alternative**; **Linear programming**; **Motzkin transposition theorem**; **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**)
Farkas lemma
[05B35, 15A39, 90C05, 90C20, 90C30, 90C33]
(*see:* **Kuhn–Tucker optimality conditions**; **Least-index anticycling rules**; **Linear optimization: theorems of the alternative**; **Motzkin transposition theorem**; **Theorems of the alternative and optimization**)
Farkas lemma: generalizations
(46A20, 90C30, 52A01)
(*referred to in:* **Farkas lemma**; **Fourier–Motzkin elimination method**; **Fractional programming**; **Global optimization: envelope representation**; **Gröbner bases for polynomial equations**; **Lagrangian duality: BASICS**; **Least-index anticycling rules**; **Theorems of the alternative and optimization**)
(*refers to:* **Farkas lemma**)
farthest-point Voronoi diagram
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
farthest-point Voronoi diagram
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
farthest vertex insertion (FVI)
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
fast Givens transformation
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **QR factorization**)
fast interchange
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
fathom
[90C11]
(*see:* **MINLP: branch and bound methods**)
fathoming a node
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
fathoming step
[90C10, 90C26]

(*see:* **MINLP: branch and bound global optimization algorithm**)
fatness
[68Q20]
(*see:* **Optimal triangulations**)
fault *see:* jump across a —; negative —; positive —
fault ridge
[90Cxx]
(*see:* **Discontinuous optimization**)
faults *see:* set of —
fc and max-regret heuristics *see:* max-regret- —
FCO
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
FCOP
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
FCTP
[90B06, 90B10, 90C26, 90C35]
(*see:* **Minimum concave transportation problems**)
FDL
[90C09]
(*see:* **Inference of monotone boolean functions**)
FDS
[90B85]
(*see:* **Multifacility and restricted location problems**)
feasibility
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**)
feasibility *see:* dual —; interdisciplinary —; Interval analysis: verifying —; primal —
feasibility analysis *see:* Shape reconstruction methods for nonconvex —
feasibility approach
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
feasibility approach to image reconstruction from projection data
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
feasibility condition *see:* strict —
feasibility convergence tests
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
feasibility cut
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**)
feasibility cuts
[49M27, 90C11, 90C15, 90C30]
(*see:* **L-shaped method for two-stage stochastic programs with recourse**; **MINLP: generalized cross decomposition**)
feasibility of equality constraints
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: verifying feasibility**)
feasibility error
[90C90, 91B28]
(*see:* **Robust optimization**)
feasibility of inequality constraints
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: verifying feasibility**)
feasibility problem
[05B35, 46A20, 52A01, 90C05, 90C20, 90C30, 90C33]
(*see:* **Farkas lemma: generalizations**; **Least-index anticycling rules**)
feasibility problem *see:* convex —; nonlinear —; zero-one integer —
feasibility set *see:* second-stage —
feasibility test
[65G20, 65G30, 65G40, 65K05, 90C26, 90C30]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Interval global optimization**)
feasibility test and flexibility index *see:* Bilevel optimization: —
feasible
[49M30, 49M37, 65K05, 90C26, 90C30, 90C31]
(*see:* **Practical augmented Lagrangian methods**; **Robust global optimization**; **Smooth nonlinear nonconvex optimization**)
feasible approach
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
feasible ascendant direction
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
feasible assignment
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
feasible basis *see:* primal —
feasible box
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
feasible component
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
feasible computational solution *see:* practically —
feasible cones *see:* high-order —
feasible for D_{DV}
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
feasible decomposition method
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
feasible direction
[65K05, 65K10, 90C06, 90C30, 90C34]
(*see:* **Feasible sequential quadratic programming**; **Kuhn–Tucker optimality conditions**; **Rosen's method, global convergence, and Powell's conjecture**)
feasible direction
[90C30]
(*see:* **Frank–Wolfe algorithm**)
feasible direction *see:* improving —
feasible direction method for nonlinear programming
[90C30]
(*see:* **Frank–Wolfe algorithm**)
feasible direction methods
[90C29, 90C30]
(*see:* **Convex-simplex algorithm**; **Multi-objective optimization; Interactive methods for preference value functions**)
feasible direction methods
[65K05, 65K10, 90C30]

(*see:* **ABS algorithms for optimization**; **Convex-simplex algorithm**; **Frank–Wolfe algorithm**)
feasible directions *see:* combined method of —; cone of —; methods of —
feasible domain
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
feasible flow
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
feasible flow problem
[90C35]
(*see:* **Maximum flow problem**)
feasible flow vector
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
feasible gradient controller
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
feasible high-order approximating cones
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
feasible high-order approximating curve
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
feasible high-order approximating vector
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
feasible iterates
[65K05, 65K10, 90C06, 90C30, 90C34]
(*see:* **Feasible sequential quadratic programming**)
feasible iterates
[65K05, 65K10, 90C06, 90C30, 90C34]
(*see:* **Feasible sequential quadratic programming**)
feasible move
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
feasible node
[90C10, 90C29]
(*see:* **Multi-objective integer linear programming**)
feasible path approach
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
feasible path flow pattern
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
feasible point
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30, 90C31]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**; **Sensitivity and stability in NLP: approximation**; **Stochastic global optimization: two-phase methods**)
feasible point
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
feasible point *see:* regular —
feasible point to a solution point *see:* bounds on the distance of a —
feasible points *see:* set of —
feasible problem
[15A39, 90C05]
(*see:* **Linear optimization: theorems of the alternative**)
feasible region
[65G30, 65G40, 65K05, 68W10, 90B15, 90C05, 90C06, 90C20, 90C29, 90C30, 90C31, 90C57, 90C90]
(*see:* **Bilevel programming: global optimization**; **Global optimization: interval analysis and balanced interval arithmetic**; **Multiple objective programming support**; **Redundancy in nonlinear programs**; **Rosen's method, global convergence, and Powell's conjecture**; **Sensitivity and stability in NLP: continuity and differential stability**; **Stochastic network problems: massively parallel solution**)
feasible region *see:* enlargement of a —; minimal representation of a —; prime representation of a —; relaxation of a —
feasible region reduction
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
Feasible sequential quadratic programming
(65K05, 65K10, 90C06, 90C30, 90C34)
(*referred to in:* **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
feasible sequential quadratic programming
[65K05, 65K10, 90C06, 90C30, 90C34]
(*see:* **Feasible sequential quadratic programming**)
feasible set
[37A35, 49M37, 65K05, 65K10, 9008, 90C05, 90C26, 90C27, 90C29, 90C30, 90C59, 93A13]
(*see:* **Multilevel methods for optimal design**; **Multiple objective programming support**; **Potential reduction methods for linear programming**; **Smooth nonlinear nonconvex optimization**; **Variable neighborhood search methods**)
feasible set *see:* branch of a —; dual —; high-order —; p-order —; primal —; strictly —

feasible solution
[9008, 90C05, 90C25, 90C26, 90C27, 90C30, 90C31, 90C33, 90C59]
(*see:* **Lagrangian multipliers methods for convex programming**; **Pivoting algorithms for linear programming generating two paths**; **Robust global optimization**; **Variable neighborhood search methods**)
feasible solution *see:* basic —; extreme —
feasible solutions *see:* set of —
feasible spanning tree structure
[90C35]
(*see:* **Minimum cost flow problem**)
feasible underestimators
[90C11, 90C26]
(*see:* **Extended cutting plane algorithm**)
feature-based aggregation
[49L20, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**)
feature detection *see:* low-level —
feature extraction
[90C39]
(*see:* **Neuro-dynamic programming**)
feature segmentation
[90C90]
(*see:* **Optimization in medical imaging**)
feature selection
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
feature space
[90C90]
(*see:* **Optimization in medical imaging**)
feature vector
[90C39]
(*see:* **Neuro-dynamic programming**)
features *see:* special model —
features, examples from financial decision making *see:* Preference disaggregation approach: basic —
fed-batch reactor
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
Feed
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
Feed algorithm
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
feed-forward network *see:* two-layer —
feed-forward neural network
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
feedback
[93D09]
(*see:* **Robust control**)
feedback *see:* incomplete state —; off-line —; on-line —
feedback arc set problem
[90C35]
(*see:* **Feedback set problems**)
feedback arc set problem *see:* minimum —; minimum weight —
feedback control
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
feedback Nash equilibrium
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
feedback set problem
[90C35]
(*see:* **Feedback set problems**)
feedback set problem
[90C35]
(*see:* **Feedback set problems**)
Feedback set problems
(90C35)
(*referred to in:* **Biquadratic assignment problem**; **Graph coloring**; **Graph planarization**; **Greedy randomized adaptive search procedures**; **Linear ordering problem**; **Quadratic assignment problem**; **Quadratic semi-assignment problem**)
(*refers to:* **Generalized assignment problem**; **Graph coloring**; **Graph planarization**; **Greedy randomized adaptive search procedures**; **Quadratic assignment problem**; **Quadratic semi-assignment problem**)
feedback Stackelberg solution
(*see:* **Bilevel programming framework for enterprise-wide process networks under uncertainty**)
feedback vertex (arc) set problem *see:* minimum —; subset —; subset minimum —
feedback vertex set
[90C35]
(*see:* **Feedback set problems**)
feedback vertex set *see:* minimum —
feedback vertex set problem
[90C35]
(*see:* **Feedback set problems**)
feedback vertex set problem *see:* minimum weighted —; unweighted —
Fejér monotone sequence
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
Fejér monotonicity
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
Fejér monotonicity in convex optimization
(47H05, 90C25, 90C55, 65J15, 90C25)
(*referred to in:* **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
(*refers to:* **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
Fejérian *see:* S- —
fekete points problem
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)

Fenchel conjugate functions
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
Fenchel cuts
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
Fenchel duality
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
Fenchel duality pair
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
Fenchel–Legendre transformation *see:* integral —
Fenchel–Moreau duality
[90C26]
(*see:* **Global optimization: envelope representation**)
Fenchel–Moreau subdifferential
[90C26]
(*see:* **Generalized monotone multivalued maps**)
Fenchel–Rockafellar duality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**; **Duality theory: triduality in global optimization**)
Fenchel–Rockafellar duality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
Fenchel–Rockafellar duality theory
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
Fenchel transformation
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**; **Duality theory: monoduality in convex optimization**)
Fenchel-type duality for M- and L-convex functions
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
Fenchel–Young inequality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
Fermat problem
[90C27]
(*see:* **Steiner tree problems**)
Fermat problem *see:* general —
F.H. Clarke *see:* generalized subdifferential of —
fH-VNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
fiber *see:* Gröbner —
Fibonnaci section search
[90C30]
(*see:* **Nonlinear least squares problems**)
fictitious domain method
[49J20, 49J52]
(*see:* **Shape optimization**)
fictitious uncertainty
[93D09]
(*see:* **Robust control**)
field *see:* electric —; sigma- —; splitting —
field approximation *see:* mean —
field via linear optimization *see:* Distance dependent protein force —
fields *see:* force —; offshore oil —
Figure Legends
(*see:* **Mixed integer nonlinear bilevel programming: deterministic global optimization**)
figures
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
fill-in *see:* intermediate —
fill-in of a graph *see:* minimum —
filled function
[65K05, 90C26, 90C30, 90C59]
(*see:* **Global optimization: filled function methods**)
filled function *see:* discrete —; globally convexized —; locally —
filled function methods
[65K05, 90C26, 90C30, 90C59]
(*see:* **Global optimization: filled function methods**)
filled function methods *see:* basic outline of —; Global optimization: —
filling *see:* Global optimization using space —
filling curve *see:* space —
filling curves *see:* approximation of space —
FILO
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
filter *see:* kalman —; Sobel edge —; Volterra —
filters *see:* wedge —
filtration *see:* stochastic process nonanticipative with respect to a —
final state of a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
finance
[90C26, 90C27, 91B06, 91B28, 91B60]
(*see:* **Financial applications of multicriteria analysis**; **Operations research and financial markets**; **Portfolio selection: markowitz mean-variance model**; **Portfolio selection and multicriteria analysis**)
finance *see:* mathematical —; Semi-infinite programming and applications in —
Financial applications of multicriteria analysis
(91B06, 91B60)
(*referred to in:* **Bi-objective assignment problem**; **Competitive ratio for portfolio management**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial optimization**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation**

approach: basic features, examples from financial decision making; **Preference modeling**; **Robust optimization**; **Semi-infinite programming and applications in finance**)
(*refers to:* **Bi-objective assignment problem**; **Competitive ratio for portfolio management**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial optimization**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**; **Robust optimization**; **Semi-infinite programming and applications in finance**)
financial decision making
[90C29]
(*see:* **Preference disaggregation approach: basic features, examples from financial decision making**)
financial decision making
[90C29]
(*see:* **Preference disaggregation approach: basic features, examples from financial decision making**)
financial decision making *see:* Preference disaggregation approach: basic features, examples from —
Financial equilibrium
(91B50)
(*referred to in:* **Equilibrium networks**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Oligopolistic market equilibrium**; **Spatial price equilibrium**; **Traffic network equilibrium**; **Walrasian price equilibrium**)
(*refers to:* **Equilibrium networks**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Oligopolistic market equilibrium**; **Spatial price equilibrium**; **Traffic network equilibrium**; **Walrasian price equilibrium**)
financial equilibrium
[91B50]
(*see:* **Financial equilibrium**)
financial equilibrium model *see:* multi-sector multi-instrument —
financial leverage hypothesis
[90C05, 90C90, 91B28]
(*see:* **Multicriteria methods for mergers and acquisitions**)
financial markets *see:* Operations research and —
Financial optimization
(91B28)
(*referred to in:* **Competitive ratio for portfolio management**; **Financial applications of multicriteria analysis**; **Portfolio selection and multicriteria analysis**; **Robust optimization**; **Semi-infinite programming and applications in finance**)
(*refers to:* **Competitive ratio for portfolio management**; **Financial applications of multicriteria analysis**; **Portfolio selection and multicriteria analysis**; **Robust optimization**; **Semi-infinite programming and applications in finance**)
financial planning
[91B28]
(*see:* **Financial optimization**)
financial planning problems *see:* Global optimization algorithms for —
find all *see:* find one —
find one, find all
(*see:* **Planning in the process industry**)
finding a minimum
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: relaxation methods**)
finding problem *see:* direction —; regularized direction —
finding procedure *see:* model —
finding shortest paths *see:* problem of —
fine structures *see:* maxdiag —; mindiag —
fine valuation structure
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
finer grid
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
finite
[46N10, 47N10, 49M37, 57R12, 65K10, 90C26, 90C30, 90C31, 90C34]
(*see:* Global optimization: tight convex underestimators; **LP strategy for interval-Newton method in deterministic global optimization**; **Parametric global optimization: sensitivity**; **Smoothing methods for semi-infinite optimization**)
finite alphabet
[90C60]
(*see:* **Complexity classes in optimization**)
finite class
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
Finite complete systems of many-valued logic algebras
(03B50, 68T15, 68T30)
(*referred to in:* **Alternative set theory**; **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
(*refers to:* **Alternative set theory**; **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
finite costs *see:* reduction to —
finite-difference approximation
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
finite difference methods
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)
finite differences
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)

finite-dimensional control problem
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
finite-dimensional linear program
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
finite-dimensional models for entropy optimization for image reconstruction
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
finite-dimensional subspace
[65M60]
(*see:* **Variational inequalities: F. E. approach**)
finite-dimensional variational inequality problem
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
finite dominating set
[90B85]
(*see:* **Multifacility and restricted location problems**)
finite dominating set
[90B85]
(*see:* **Multifacility and restricted location problems**)
finite element
[49M37, 65K05, 90C26, 90C30, 90C90]
(*see:* **Structural optimization**; **Structural optimization: history**)
finite element
[49M37, 65K05, 90C30]
(*see:* **Structural optimization**)
finite element *see:* mixed —
finite element approximation
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
finite element approximation *see:* mixed —
finite element method
[49J40, 49J52, 49M05, 49Q10, 49S05, 65M60, 70-08, 74G99, 74H99, 74K99, 74Pxx, 90C33, 90C90, 91A65, 94A08, 94A17]
(*see:* **Hemivariational inequalities: applications in mechanics**; **Maximum entropy principle: image reconstruction**; **Multilevel optimization in mechanics**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Variational inequalities: F. E. approach**)
finite ϵ-convergence
[49M29, 90C11]
(*see:* **Generalized benders decomposition**)
finite generation method *see:* Lagrangian —
finite horizon
(*see:* **Bayesian networks**)
finite jump system
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
finite minimax problem
[49K35, 49M27, 65K10, 90C25]
(*see:* **Convex max-functions**)
finite moment problem
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
finite natural numbers
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
finite nested family
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
finite optimality
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
finite optimization problem *see:* one-parametric —
finite rational numbers
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
finite sequence *see:* generalized —
finite set *see:* hierarchy in a —
finite set of the alternatives
[90-XX]
(*see:* **Outranking methods**)
finite-state Markov chain
[49L20, 90C39]
(*see:* **Dynamic programming: discounted problems**)
Finsler theorem
[93D09]
(*see:* **Robust control**)
firmly nonexpansive operator
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
first *see:* best- —; breadth- —; depth- —
first algorithm *see:* mandatory work —
first bank
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
first-cluster second *see:* schedule —
first descent
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
first-In-First-Out
(*see:* **Railroad crew scheduling**)
first-in last-out rule
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
first level problem
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
first order approximation of a function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
first order changes *see:* up to —
first order constraint qualification
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
First order constraint qualifications
(90C30, 49K27, 90C31, 49K40)
(*referred to in:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Nondifferentiable optimization: parametric**

programming; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**)

first order constraint qualifications
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)

First order CQ
[49K27, 49K40, 90C26, 90C30, 90C31, 90C39]
(*see:* **First order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**)

first order differential equations *see:* Duality in optimal control with —

first order KKT conditions
[90C31]
(*see:* **Bounds and solution vector estimates for parametric NLPS**)

first order necessary condition
[49M29, 65K10, 90C06, 90C26, 90C39]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**; **Second order optimality conditions for nonlinear optimization**)

first order necessary conditions
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)

first order necessary optimality conditions
[49M37, 65K05, 90C30]
(*see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**)

first order optimality
[49M37, 65K05, 90C30]
(*see:* **Inequality-constrained nonlinear optimization**)

First order partial differential equations
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)

first order and second order optimality conditions
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)

first order tangent set
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)

first order Taylor series expansion
[90C30]
(*see:* **Simplicial decomposition**)

first order Taylor series expansion
[90C30]
(*see:* **Convex-simplex algorithm**; **Frank–Wolfe algorithm**; **Simplicial decomposition**)

first order theory of real addition with order
[52B12, 68Q25]
(*see:* **Fourier–Motzkin elimination method**)

First-Out *see:* first-In- —

first principle *see:* Wardrop —

first-schedule second strategy *see:* cluster —

first search *see:* depth- —

first search with backtracking *see:* depth- —

first slope lemma
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)

first-stage decision
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)

first-stage decisions
[90B10, 90B15, 90C10, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**; **Stochastic integer programs**; **Stochastic programming: parallel factorization of structured matrices**; **Stochastic vehicle routing problems**)

first tree search *see:* best- —; depth- —; Parallel Best- —; Parallel Depth- —

Fischer–Burmeister function
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)

fit *see:* best —

fitness
[92B05]
(*see:* **Broadcast scheduling problem**; **Genetic algorithms**)

fitness
[92B05]
(*see:* **Genetic algorithms**)

fitness *see:* genetic engineering via negative —

fitness function
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)

fittest *see:* survival of the —

fitting *see:* curve —; data —; subjective curve —

fitting to data *see:* best —

fitting and extrapolation *see:* subjective curve —

fix *see:* dive-and- —; near-integer- —; relax-and- —

fixed charge
[90C25]
(*see:* **Concave programming**)

fixed charge
[90B10, 90B80, 90C11]
(*see:* **Piecewise linear network flow problems**; **Stochastic transportation and location problems**)

fixed charge *see:* linear —

fixed charge function
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)

fixed charge network flow problem
[90B10]
(*see:* **Piecewise linear network flow problems**)

fixed charge networks
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)

fixed charge problem
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
fixed charge transportation problem
[90B06, 90B10, 90C26, 90C35]
(*see:* **Minimum concave transportation problems**)
fixed charge transportation problem
[90B06, 90B10, 90C26, 90C35]
(*see:* **Minimum concave transportation problems**)
fixed cost with capacity constraints *see:* single —
fixed cost with no capacity constraints *see:* single —
fixed degree of flexibility
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
fixed demand traffic network equilibrium
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
fixed demand traffic network problems
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
fixed number of vehicles *see:* Vehicle scheduling problems with a —
fixed parameter tractability
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
fixed parameter tractable algorithms
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)
fixed point
[49L20, 49M29, 65H10, 65J15, 65K10, 90C06, 90C30, 90C39, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**; **Contraction-mapping**; **Dynamic programming: discounted problems**; **Local attractors for gradient-related descent iterations**)
fixed point *see:* coupled —
fixed point computation
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
fixed point iteration
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval fixed point theory**)
fixed point problem
[47H05, 65J15, 90C25, 90C33, 90C55]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Fejér monotonicity in convex optimization**)
fixed point problem
[65K10, 65M60, 90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Variational inequalities**)
fixed point problem *see:* Equivalence between nonlinear complementarity problem and —
fixed point theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
fixed point theorem *see:* brouwer —; Miranda —; Schauder —; Tychonoff —
fixed point theory
[90C05, 90C10, 90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Simplicial pivoting algorithms for integer programming**)
fixed point theory *see:* Interval —
fixed recourse
[90C15]
(*see:* **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
fixed tabs search
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
fixed travel demand
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
fixedTime
(*see:* **Medium-term scheduling of batch processes**)
fixing *see:* reduced cost —
FL
[15A39, 90B80, 90C05]
(*see:* **Facilities layout problems**; **Farkas lemma**)
flat fuzzy number *see:* L-R —
fleet *see:* mixed —
fleet assignment
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
fleet assignment *see:* airline —
fleet assignment problem
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
Fletcher–Goldfarb–Shanno method *see:* Broyden– —
Fletcher–Goldfarb–Shanno quasi-Newton update *see:* Broyden– —
Fletcher–Goldfarb–Shanno update *see:* Broyden– —
Fletcher–Powell method *see:* Davidon– —
Fletcher–Powell update *see:* Davidon– —
Fletcher–Reeves algorithm
[90C30]
(*see:* **Conjugate-gradient methods**)
Fletcher–Reeves formula
[90C06]
(*see:* **Large scale unconstrained optimization**)
Fletcher–Reeves method
[90C06]
(*see:* **Large scale unconstrained optimization**)
flexibility
[90C26]
(*see:* **Global optimization in batch design under uncertainty**)
flexibility
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
flexibility *see:* fixed degree of —; optimal degree of —; stochastic —
flexibility analysis of flowsheets
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)

flexibility index
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
flexibility index *see:* Bilevel optimization: feasibility test and —
flexible arm
[93-XX]
(*see:* **Optimal control of a flexible arm**)
flexible arm *see:* Optimal control of a —
Flexible mass exchange networks
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**)
flexible MOLP with fuzzy coefficients
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
flexible programming
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
flexible templates *see:* De novo protein design using —
flight schedule
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
flipping *see:* algorithm partition- —; partition —
flipping model
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
flips *see:* good edge —
floating point intervals
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
floating point operation
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
floor function
[65K05]
(*see:* **Direct global optimization algorithm**)
flop
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
flow
[90C35]
(*see:* **Maximum flow problem**)
flow *see:* ascent —; conservation of —; descent —; feasible —; generalized —; material —; maximize operating cash —; maximum —; minimum cost network —; multicommodity —; relaxed multicommodity —; value of a —; value of a network —
flow across an s—t-cut
[90C35]
(*see:* **Maximum flow problem**)
flow algorithm *see:* max– —
flow balance equations *see:* node —
flow bound constraints
[90C35]
(*see:* **Maximum flow problem**; **Minimum cost flow problem**)
flow bounds *see:* arc —
flow conservation constraint
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
flow conservation law
(*see:* **Peptide identification via mixed-integer optimization**)
flow constraints
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
flow decision variable
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)
flow equation *see:* conservation of —
flow equations *see:* conservation of —
flow formulation *see:* consist —; link —; path —
flow lines *see:* connection of —
flow min-cut theorem *see:* max- —
flow model *see:* network —
flow models *see:* undirected multicommodity network —
flow pattern *see:* feasible path —
flow problem
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
flow problem *see:* feasible —; fixed charge network —; linear network —; maximal —; Maximum —; minimum cost —; minimum cost network —; multicommodity network —; network —; node-path formulation of the multicommodity —; nonconvex network —; nonlinear dynamic network —; nonlinear network —; nonlinear single commodity network —; package —; piecewise linear minimum cost network —; uncapacitated network —
flow problem with nonnegative lower bounds *see:* maximum —
flow problems *see:* dynamic network —; large nonlinear multicommodity —; maximum —; Multicommodity —; Nonconvex network —; nonlinear multicommodity —; nonlinear network —; Nonoriented multicommodity —; Piecewise linear network —
flow-shop
[05-04, 90B36, 90C26, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**; **MINLP: design and scheduling of batch processes**; **Stochastic scheduling**)
flow-shop problem
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
Flow shop scheduling problem
(68M20, 90B35)
flow solver
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
flow vector *see:* feasible —
flowing wells of type a *see:* naturally —
flowing wells of type b *see:* naturally —
flowlines *see:* set of —
flowmax
[90B35, 93A30]
(*see:* **Gasoline blending and distribution scheduling: an MILP model**)
flowmin
[90B35, 93A30]
(*see:* **Gasoline blending and distribution scheduling: an MILP model**)
flowrate *see:* well oil —

flows *see:* augmenting —; balance equations for material —; capacity constraint on arc —; global gradient —; logistics —; Multi-commodity —; multicommodity network —; network —; variational inequality formulation in path —
flows with gains
[90C35]
(*see:* **Generalized networks**)
flows in networks
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C26, 90C27, 90C30, 90C35]
(*see:* **Minimum concave transportation problems**; **Nonconvex network flow problems**; **Vehicle scheduling**)
flowsheet *see:* convergence of the overall —; process —
flowsheet optimization
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
flowsheets *see:* flexibility analysis of —; operability analysis of —; sensitivity of optimal —
flowtime
[90B36]
(*see:* **Stochastic scheduling**)
FLP
[90B80]
(*see:* **Facilities layout problems**)
fluctuations *see:* thermal —
fluence map optimization
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
fluid dynamics *see:* computational —; Design optimization in computational —
flux estimation in distributed systems *see:* boundary —
flux estimation in lumped systems *see:* reaction —
flux models *see:* estimation of diffusion —
fluxes *see:* estimation of 1D-diffusion —
FMOLP
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
fold integer programming *see:* n- —
fold matrix *see:* n- —
folding *see:* Adaptive simulated annealing and its application to protein —; Global optimization in protein —; Monte-Carlo simulated annealing in protein —; protein —; Simulated annealing methods in protein —
folding: αBB global optimization approach *see:* Multiple minima problem in protein —
folding: generalized-ensemble algorithms *see:* Protein —
folding problem *see:* protein —
folks theorem
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
follower problem
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
following
[90B60, 90B80, 90B85]
(*see:* **Competitive facility location**)
following *see:* path —
following algorithm *see:* path —
following algorithm for entropy optimization *see:* path —
following approach *see:* path —
following methods *see:* path —
following and singularities *see:* Parametric optimization: embeddings, path —
forbidden or tabu
[68M20, 90B06, 90B35, 90B80, 90C59]
(*see:* **Flow shop scheduling problem**; **Heuristic and metaheuristic algorithms for the traveling salesman problem**; **Location routing problem**; **Metaheuristic algorithms for the vehicle routing problem**)
force *see:* brute- —
force field via linear optimization *see:* Distance dependent protein —
force fields
[65K10, 92C40]
(*see:* **Multiple minima problem in protein folding: αBB global optimization approach**)
forced *see:* color- —
Ford–Fulkerson algorithm
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
Ford method *see:* Bellman– —
forecast *see:* judgemental —
Forecasting
(90C30, 90C26)
(*referred to in:* **Continuous global optimization: applications**)
(*refers to:* **Continuous global optimization: applications**; **Genetic algorithms**)
forecasting methods *see:* qualitative —; quantitative —
forecasting model
[90C26, 90C30]
(*see:* **Forecasting**)
foreset
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
foreset and afterset representation of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
forest *see:* basis —
forest management
[90C35]
(*see:* **Multicommodity flow problems**)
form *see:* AD intermediate —; block angular —; Boolean formula in conjunctive normal —; canonical —; canonical normal —; coercive bilinear symmetric continuous —; complete many-valued logic normal —; conjunctive normal —; constraints in standard —; disjunctive normal —; echelon —; extensive —; game in normal —; K-local bilinear —; Lagrange —; Lagrangian —; linear optimization problem in standard —; logarithmic —; logarithmic p- —; many-valued normal —; matrix in standard —; Mayer —; normal —; PI-normal —; rational p- —; standard —; standard greedy —; Taylor —
form approach *see:* closed —; open —
form of CEP *see:* restricted accessibility —; universally accessible —
form of coordinates *see:* kth order —

form of KT conditions *see:* nonstoichiometric —; stoichiometric —
form of a polynomial *see:* normal —
form test *see:* Taylor —
form transformation *see:* unimodular max-closed —
form transformations *see:* unimodular max-closed —
formal orthogonal polynomials *see:* least squares —
formal perfect dual
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
formation *see:* automated hypothesis —
formation values *see:* set of —
formats *see:* independent of solver —
forms *see:* bilinear —; Global optimization: functional —; minimization of Pinkava normal —; PI-algebras and 2-valued normal —; Quadratic integer programming: complexity and equivalent —; tricanonical —
forms of Pi-algebras *see:* functionally complete normal —
formula *see:* Bauer —; Cauchy —; Euler —; Fletcher–Reeves —; integral over surface —; integral over volume —; marginal value —; Moré updating —; Polak–Ribiére —; satisfiable Boolean —; selfdual rank one —; set- —; Sherman–Morrison —; Sherman-Morrison rank-one update —; Sherman–Morrison–Woodbury —
formula in conjunctive normal form *see:* Boolean —
formulas *see:* Horn —; satisfiability of Boolean —
formulation *see:* column generation —; consist flow —; continuous-time —; inverse interpolation parametric eigenvalue —; LCP: Pardalos–Rosen mixed integer —; least squares —; link flow —; mathematical —; mixed variational —; multilevel problem —; node-arc —; path —; path flow —; price —; problem —; quantity —; saddle-point —; Scarf —; separable —; variational inequality —
formulation in link loads *see:* variational inequality —
formulation of the multicommodity flow problem *see:* node-path —
formulation in path flows *see:* variational inequality —
formulation of the problem *see:* node-arc —
formulation of quasidifferential laws *see:* variational —
formulation of quasidifferential thermal boundary conditions *see:* variational —
formulation and solution of inverse problems
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
formulation of SP *see:* split-variable —
formulation space search
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
formulation of subdifferential laws *see:* variational —
formulations *see:* discrete-time —; Quasidifferentiable optimization: variational —; Stochastic optimal stopping: problem —; variational inequality —
Forrest–Goldfarb method
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
Fortran *see:* high performance —; Vienna —
Fortran program for nonlocal sensitivity analysis *see:* automated —
FORTRAN subroutines
[90C35]
(*see:* **Feedback set problems**)
forward arc
[90C35]
(*see:* **Maximum flow problem**)
forward automatic differentiation *see:* vector —
forward compatibility
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
forward mode
[65D25, 65H99, 65K99, 68W30]
(*see:* **Automatic differentiation: point and interval**; **Complexity of gradients, Jacobians, and Hessians**)
forward mode of AD
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
forward mode of an AD algorithm
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)
forward mode of automatic differentiation
[26A24, 65G20, 65G30, 65G40, 65H20, 65K99, 85-08]
(*see:* **Automatic differentiation: geometry of satellites and tracking stations**; **Interval analysis: intermediate terms**)
forward network *see:* two-layer feed- —
forward neural network *see:* feed- —
forward path
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
forward phases
[90C35]
(*see:* **Graph coloring**)
forward substitution
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
foundations of industrial engineering *see:* Archimedes and the —
four-argument function
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
Fourier *see:* mechanical principle of —
Fourier law of heat conduction
[35R70, 47S40, 74B99, 74D99, 74G99, 74H99]
(*see:* **Quasidifferentiable optimization: applications to thermoelasticity**)
Fourier–Motzkin elimination
[52B12, 68Q25]
(*see:* **Fourier–Motzkin elimination method**)
Fourier–Motzkin elimination method
(52B12, 68Q25)
(*refers to:* **Farkas lemma**; **Farkas lemma: generalizations**; **Linear programming**)
Fourier–Motzkin method
[52B12, 68Q25]
(*see:* **Fourier–Motzkin elimination method**)
Fourier relaxation method *see:* Agmon–Motzkin– —

FP
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
fractal interface
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
fractal set
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
fractional 0–1 knapsack
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
fractional 0-1 programming problem
(*see:* **Fractional zero-one programming**)
Fractional combinatorial optimization
(90-08, 90C27, 90C32, 68Q25, 68R05)
(*referred to in:* **Combinatorial matrix analysis**; **Combinatorial optimization algorithms in resource allocation problems**; **Combinatorial optimization games**; **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Evolutionary algorithms in combinatorial optimization**; **Fractional programming**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **Multi-objective combinatorial optimization**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**; **Quadratic fractional programming: Dinkelbach method**; **Replicator dynamics in combinatorial optimization**; **Stochastic integer programs**)
(*refers to:* **Bilevel fractional programming**; **Combinatorial matrix analysis**; **Combinatorial optimization algorithms in resource allocation problems**; **Combinatorial optimization games**; **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Evolutionary algorithms in combinatorial optimization**; **Fractional programming**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **Multi-objective combinatorial optimization**; **Neural networks for combinatorial optimization**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**; **Quadratic fractional programming: Dinkelbach method**; **Replicator dynamics in combinatorial optimization**)
fractional combinatorial optimization *see:* linear —; uniform —
fractional combinatorial optimization problem
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
fractional combinatorial optimization problem *see:* integral linear —
fractional (hyperbolic) 0-1 programming problem *see:* single-ratio —
fractional linear programming
[90C11]
(*see:* **MINLP: branch and bound methods**)
fractional optimization
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
fractional optimization
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
fractional optimization *see:* parametric approach to —
fractional program
[90C32]
(*see:* **Fractional programming**)
fractional program *see:* concave —; generalized —; linear —; max-min —; min-max —; multi-objective —; quadratic —; single-ratio —; sum-of-ratios —
Fractional programming
(90C32)
(*referred to in:* **Fractional combinatorial optimization**; **Quadratic fractional programming: Dinkelbach method**)
(*refers to:* **Bilevel fractional programming**; **Farkas lemma**; **Farkas lemma: generalizations**; **Fractional combinatorial optimization**; **Quadratic fractional programming: Dinkelbach method**)
fractional programming
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
fractional programming
[90C27, 90C32]
(*see:* **Fractional programming**; **Operations research and financial markets**; **Quadratic fractional programming: Dinkelbach method**)
fractional programming *see:* Bilevel —; combinatorial —; integer —; linear- —; multi-objective —
fractional programming: Dinkelbach method *see:* Quadratic —
fractional programming problem
[90C32]
(*see:* **Quadratic fractional programming: Dinkelbach method**)
fractional programming problems *see:* Multi-objective —
fractional programs *see:* classification of —
fractional routing pattern model
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
fractional terms *see:* linear —
fractional updating
(*see:* **Bayesian networks**)
Fractional zero-one programming
frame
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**)
framework *see:* Bayesian decision-theoretic —; graph based —; linear algebra —; multiperiod optimization modeling —; Newton–Cauchy —; nonstandard —; primal-dual —; proximal —; Unconstrained nonlinear optimization: Newton–Cauchy —
framework for enterprise-wide process networks under uncertainty *see:* Bilevel programming —
frameworkfor radiation therapy *see:* Optimization based —
frameworks *see:* Short-term scheduling, resource constrained: unified modeling —
Frank discrete separation theorem
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)

Frank–Wolfe
[90C06, 90C25, 90C30, 90C35]
(*see:* **Cost approximation algorithms**; **Frank–Wolfe algorithm**; **Simplicial decomposition algorithms**)
Frank–Wolfe algorithm
(90C30)
(*referred to in:* **Cost approximation algorithms**; **Simplicial decomposition**; **Stochastic transportation and location problems**; **Traffic network equilibrium**)
(*refers to:* **Rosen's method, global convergence, and Powell's conjecture**)
Frank–Wolfe algorithm
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
Frank–Wolfe algorithm
[90C30]
(*see:* **Cost approximation algorithms**; **Simplicial decomposition**)
Frank–Wolfe algorithm *see:* regularized —
Frank–Wolfe decomposition *see:* regularized —
Fréchet
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
Fréchet differentiable function
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
fréchet normal cone
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
Fréchet subdifferential
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
Fréchet subdifferential *see:* limiting —; singular —
Fréchet subdifferentials
[49K27, 58C20, 58E30, 90C46, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**; **Nonsmooth analysis: weak stationarity**)
Fréchet subdifferentials *see:* limiting —; Nonsmooth analysis: —
Fréchet superdifferential
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
free *see:* univariate gradient —
free algorithm *see:* gradient- —
free alignment *see:* communication- —
free alignment problem *see:* communication- —
free arrangement of hyperplanes
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
free asset *see:* risk- —
free assignment
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
free coloring *see:* conflict- —
free descent method *see:* derivative- —
free distributive lattice
[90C09]
(*see:* **Inference of monotone boolean functions**)
free distributive lattice
[90C09]
(*see:* **Inference of monotone boolean functions**)
free energy
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
free energy
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
free energy *see:* molar Gibbs —; total Gibbs —
free Givens transformation *see:* square-root- —
free lunch *see:* no —
free methods for non-smooth optimization *see:* Derivative- —
free minimization *see:* gradient- —
free minimization algorithm *see:* gradient- —
free reduced Hessian SQP *see:* multiplier- —
free shape design *see:* robust obstacle- —
free truss design *see:* robust obstacle- —
free variables *see:* Generalized geometric programming: mixed continuous and discrete —
freight operation
[90C35]
(*see:* **Multicommodity flow problems**)
frequency assignment *see:* adjacent channel constrained —; co-channel constrained —; order of a T-coloring —; span of a T-coloring —
Frequency assignment problem
(05-XX)
(*referred to in:* **Assignment and matching**; **Assignment methods in clustering**; **Bi-objective assignment problem**; **Broadcast scheduling problem**; **Communication network assignment problem**; **Graph coloring**; **Linear ordering problem**; **Maximum constraint satisfaction: relaxations and upper bounds**; **Maximum partition matching**; **Quadratic assignment problem**)
(*refers to:* **Assignment and matching**; **Assignment methods in clustering**; **Bi-objective assignment problem**; **Communication network assignment problem**; **Graph coloring**; **Maximum constraint satisfaction: relaxations and upper bounds**; **Maximum partition matching**; **Quadratic assignment problem**)
frequency assignment problem *see:* radio link —
frequency exhaustive sequential coloring
[05-XX]
(*see:* **Frequency assignment problem**)
frequentist
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
friction *see:* coupled unilateral contact problem with —
frictional contact *see:* Signorini-Coulomb unilateral —
Friedrich *see:* Gauss, Carl —
Frieze–Yadegar linearization
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Fritz John conditions
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: verifying feasibility**)
Fritz John conditions
[90C15]

(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
Fritz John generalized conditions
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
fritz John necessary optimality conditions
[49M37, 65K05, 90C29, 90C30]
(*see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Generalized concavity in multi-objective optimization**)
Fritz John rule
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
Fritz John system
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: verifying feasibility**)
Fritz John type condition
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
Frobenius theorem *see:* Perron– —
Fromovitz constraint qualification *see:* Mangasarian– —
Fromovitz CQ *see:* Mangasarian– —
frontier *see:* efficient —
frontier of efficient portfolios
[91B50]
(*see:* **Financial equilibrium**)
FSP
[90C35]
(*see:* **Feedback set problems**)
FSQP
[65K05, 65K10, 90C06, 90C30, 90C34]
(*see:* **Feasible sequential quadratic programming**)
fuel mixture problem
(*see:* **Planning in the process industry**)
fugacity coefficient
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
Fulkerson algorithm *see:* Ford– —
full components
[90C27]
(*see:* **Steiner tree problems**)
full discretization
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
full master problem
[90C06]
(*see:* **Decomposition principle of linear programming**)
full master program
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
full recourse
[90C30, 90C35]
(*see:* **Optimization in water resources**)
full recourse
[90C30, 90C35]
(*see:* **Optimization in water resources**)
full row rank
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
full space methods
[90C30, 90C90]
(*see:* **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**)
full space methods *see:* Successive quadratic programming: —
full space SQP
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
full space SQP method
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in distillation systems**)
full space successive quadratic programming
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
full space of x variables
[90C30]
(*see:* **Successive quadratic programming**)
full space of x variables
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
full Steiner tree
[90C27]
(*see:* **Steiner tree problems**)
full-step Gauss–Newton method
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
full-step Gauss–Newton method
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
fully indecomposable matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
fully nonlinear problem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
fully polynomial time approximation scheme
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
fully stressed design
[90C26, 90C90]
(*see:* **Structural optimization: history**)
function
[01A99]
(*see:* **Global optimization: functional forms**; **Leibniz, gottfried wilhelm**)
function *see:* ∂^+- —; abstract convex —; achievement —; activation —; active —; admissible pair of trajectory-function and control- —; affine —; aggregate excess demand —; aggregation —; α-concave —; antitone Boolean —; antitone monotone Boolean —; approximating the recourse —; augmented Lagrangian —; auxiliary —; barrier —; bias —; bilinear —; Boolean —; Boolean 2-valued —; boundary of a —; C-differentiable —; c.d. —; cell of a —; characteristic —; Chen–Harker–Kanzow–Smale —; Chvátal —;

classification —; coboundary of a —; codifferentiable —; complementary gap —; concave —; conjugate —; constraint on a multiplicative —; continuously codifferentiable —; control —; convex —; convex-like —; convex max- —; convex quadratic —; coordinatewise increasing —; coordinatewise increasing utility —; cost —; Courant penalty —; cyclic shift —; D- —; d.c. —; decomposition of a continuous piecewise linear —; delta —; derivative of a —; derivative of a probability —; difference convex —; difference sublinear —; differentiable —; Dini codifferentiable —; Dini conditionally differentiable —; Dini conditionally directionally differentiable —; Dini differentiable —; Dini directionally differentiable —; Dini quasidifferentiable —; Dini uniformly differentiable —; Dini uniformly directionally differentiable —; directionally differentiable —; discontinuous —; discrete —; discrete filled —; distribution —; d.m. —; domain of a —; double-well —; DSL —; dual potential —; effective domain of a —; effective set of a —; energy —; exact L_∞-penalty —; exact penalty —; excess —; expectation of an indicator —; expected recourse —; expected value —; exponential —; fair objective —; filled —; first order approximation of a —; Fischer–Burmeister —; fitness —; fixed charge —; floor —; four-argument —; Fréchet differentiable —; gap —; generalized differentiable (GD) —; geodesic convex —; Gibbs —; globally convexized filled —; good inclusion —; gradient of a probability —; gradient-related set —; greedy —; H-convex —; Hadamard codifferentiable —; Hadamard conditionally differentiable —; Hadamard conditionally directionally differentiable —; Hadamard differentiable —; Hadamard directionally differentiable —; Hadamard quasidifferentiable —; Hamiltonian —; Hessian matrix of a Lagrangian —; hyperdifferentiable —; hypodifferentiable —; implicit utility —; inclusion —; increasing —; indicator —; infimum of a Lagrangian —; int U-quasiconcave —; integral Mean-Value for Composite Convexifiable —; invex —; IPH —; isotone Boolean —; isotone inclusion —; isotone monotone Boolean —; isotonic —; K-convex —; Karmarkar potential —; kernel —; Kojima —; Kreisselmeier–Steinhauser —; KyFan —; L-convex —; l_1 exact penalty —; ℓ_1 penalty —; Lagrange —; Lagrangian —; least squares distance —; Lennard-Jones potential energy —; lexicographically minimax objective —; LFS —; likelihood —; linear appearance of control —; linear supporting —; Lipschitz —; Lipschitz continuous —; list square merit —; locally filled —; locally Lipschitz —; locally Lipschitz continuous —; locally monotone —; locally strictly monotone —; locally strongly monotone —; logarithmic barrier —; logarithmic-quadratic barrier-penalty —; logconcave —; logconcave probability density —; logconvex —; lower bound —; lower semicontinuous —; Luc U-quasiconcave —; Lyapunov —; M-convex —; marginal —; max- —; max-closed —; max-type —; maximin objective —; maximum —; maximum-type —; maxmin —; mean value —; membership —; merit —; the mid-point acceleration —; min-type —; minimal —; minimax objective —; minimum —; mixed integer value —; Moebius —; monotone —; monotone Boolean —; monotonic —; multicriteria objective —; multifacility Weber objective —; multifacility Weber–Rawls objective —; multivariate probability distribution —; nonconvex —; nonconvex energy —; nondecreasing —; nondecreasing monotone Boolean —; nondifferentiable —; nonincreasing monotone Boolean —; nonsmooth —; objective —; one-dimensional marginal probability distribution —; optimal value —; Optimization techniques for minimizing the energy —; order of an inclusion —; parabolic-exponential —; partially separable —; Peano —; penalty —; perturbation —; piecewise continuously differentiable —; piecewise differentiable —; piecewise linear —; piecewise linear quadratic —; piecewise twice-differentiable —; polynomial time computable —; positive definite quadratic —; positively homogeneous —; potential —; potential energy —; pre-declared interval —; pre-invex —; preference value —; primal-dual potential —; primal gap —; primal potential —; probability —; program of minimizing a convex multiplicative —; program of minimizing a generalized convex —; projected Hessian matrix of a Lagrangian —; pseudoconvex —; pure complementary gap —; quadratic —; quantile —; quasiconcave —; quasiconvex —; quasidifferentiable —; $\mathbf{R}^n_+$-upper semicontinuous —; radially continuous —; random objective —; recourse —; regular cost —; regular link cost —; regularized gap —; rounding —; saddle —; sawtooth arc cost —; scalarizing —; scale —; score —; scoring —; second order decomposition of a —; semicoercive —; semismooth —; semistrictly quasiconvex —; separable convex objective —; separable objective —; set-valued objective —; Shannon —; Sheffer —; sign —; single smooth —; social utility —; stable —; staircase arc cost —; staircase cost —; standard —; step —; strictly convex —; strictly monotone —; strictly pseudoconvex —; strictly quasiconvex —; strongly monotone —; strongly semismooth —; subconjugate —; subcritical —; subdifferentiable —; subdual —; sublinear —; submodular —; supconjugate —; superadditive —; supercritical —; superdifferentiable —; superlinear —; supermodular —; support —; support set of a —; Tanabe–Todd–Ye potential —; three-argument —; time complexity —; total cost —; trajectory —; twice codifferentiable —; twice continuously codifferentiable —; twice-differentiable part of a —; two-dimensional marginal probability distribution —; U-concave —; U-continuous —; U-pseudoconcave —; U-quasiconcave —; U-weakly pseudoconcave —; umbrella —; uniform P- —; uniformly convex —; upper semicontinuous —; upper semismooth —; utility —; value —; zone of a —
function of an algorithm *see:* time complexity —
function approach *see:* Bilevel programming: implicit —; continuously differentiable exact penalty —; implicit —; value —
function approach to bilevel programming *see:* implicit —
function associated with Λ *see:* canonical —
function based algorithm *see:* exact penalty —
function and control-function *see:* admissible pair of trajectory- —
function with dependent constraints *see:* maximum —
function inference *see:* monotone Boolean —
function inference problem *see:* Boolean —
function martingale *see:* score —

function of a matroid *see:* weight —
function method *see:* score —
function methods *see:* basic outline of filled —; filled —; Global optimization: filled —
function minimax inequality *see:* two- —
function optimization *see:* marginal —
function pair *see:* convex-like —
function parametrization *see:* objective —
function space
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
function space *see:* canonical —; extended canonical —
function system *see:* iterative —
function theorem *see:* implicit —
function value *see:* continuity property of the objective —; convexity property of the objective —
functional *see:* absolutely continuous —; cost —; generalized critical point of an energy —; Lagrange —; recession —; substationarity point of a —; truth- —
functional analysis
[01A99]
(*see:* **Kantorovich, Leonid Vitalyevich**)
functional analysis
[01A99]
(*see:* **Kantorovich, Leonid Vitalyevich**)
functional completeness
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
functional completeness
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
functional completeness of PI-algebras
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
functional dependence *see:* noisy —
functional dual
[90C10, 90C46]
(*see:* **Integer programming duality**)
functional forms *see:* Global optimization: —
functional paradigm
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
functional relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
functionally complete normal forms of Pi-algebras
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
functionals *see:* Approximation of extremum problems with probability —; probability —
functions *see:* additive utility —; Affine sets and —; asymptotically admissible pair of trajectory and control —; augmented Lagrange —; conjugate —; continuous selection of —; Convex max- —; dc —; difference of convex —; difference of max-type —; difference of monotonic —; discrete —; discriminant —; distance —; dM —; elastic demand traffic network problems with travel demand —; elementary —; estimation of utility —; examples of quasidifferentiable —; Fenchel conjugate —; Fenchel-type duality for M- and L-convex —; gradient of multivariate distribution —; h-convex —; homotopic —; Inference of monotone boolean —; interactive learning of Boolean —; isotone —; L-convex functions and M-convex —; Lagrange-type —; linear —; Lipschitzian operators in best approximation by bounded or continuous —; marginal —; marginal distribution —; minimizing —; Multi-objective optimization; Interactive methods for preference value —; multimodal —; natural level —; nondifferentiable objective —; nonsmooth —; notation for objective —; objective —; optimal value —; penalty —; probability —; product of affine —; product of concave —; product of convex —; production —; program of minimizing a product of two affine —; proximal minimization with D- —; quasidifferentiable —; Quasidifferentiable optimization: algorithms for hypodifferentiable —; Quasidifferentiable optimization: algorithms for QD —; Quasidifferentiable optimization: codifferentiable —; quasidifferential —; sample and expectation —; scheduling —; separation —; set of elementary —; smoothing —; sum of convex multiplicative —; superpositions of —; theory of generalized —; traffic network equilibrium with travel disutility —
functions: algorithms and complexity *see:* Regression by special —
functions and/or derivatives *see:* evaluation of objective —
Functions and Applications *see:* minimization Methods for Non-Differentiable —
functions, characterization of *see:* Convexifiable —
functions: general theory and examples *see:* Derivatives of probability and integral —
functions: hemivariational inequalities *see:* Nonconvex energy —
functions in integer programming *see:* cost —
functions: kernel type solution methods *see:* Extremum problems with probability —
functions and M-convex functions *see:* L-convex —
functions on topological vector spaces *see:* Increasing and convex-along-rays —; Increasing and positively homogeneous —
fundamental cycle
[90C35]
(*see:* **Minimum cost flow problem**)
fundamental group
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
fundamental indiscernibility
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
fundamental property in convex programming
[90C06]
(*see:* **Saddle point theory and optimality conditions**)
fundamental theorem *see:* Weyl —
Fundamental theorem of algebra
(01A55, 01A50, 01A60)
(*referred to in:* **Gröbner bases for polynomial equations**)
(*refers to:* **Gröbner bases for polynomial equations**)
fundamental theorem of algebra
[01A99]
(*see:* **Gauss, Carl Friedrich**)

fundamental theorem of algebra
[01A99]
(*see:* **Gauss, Carl Friedrich**)
fundamental theorem of linear programming *see:* Extension of the —
fundamental theorem of natural selection
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
funding decision
[90C27]
(*see:* **Operations research and financial markets**)
funnel
[65H20]
(*see:* **Multi-scale global optimization using terrain/funneling methods**)
funneling methods *see:* Multi-scale global optimization using terrain/ —
fuzzification
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
fuzziness
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
fuzziness *see:* unnormalized —
fuzzy
[94A17]
(*see:* **Jaynes' maximum entropy principle**)
fuzzy clustering
[65K05, 90C26, 90C56, 90C90]
(*see:* **Derivative-free methods for non-smooth optimization**; **Nonsmooth optimization approach to clustering**)
fuzzy coefficients *see:* flexible MOLP with —; MOLP with —; multi-objective linear programming with —
fuzzy constraints
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
fuzzy criterion
[90C29, 91A99]
(*see:* **Preference disaggregation**)
fuzzy decision
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
fuzzy goals
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
fuzzy interval inference
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
fuzzy interval pairs
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
fuzzy logic
[90C26, 90C30]
(*see:* **Forecasting**)
fuzzy logic
[90C26, 90C30]
(*see:* **Forecasting**)
fuzzy logics *see:* Checklist paradigm semantics for —
Fuzzy multi-objective linear programming
(90C70, 90C29)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
fuzzy number
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
fuzzy number *see:* L-R —; L-R flat —
fuzzy numbers
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
fuzzy numbers *see:* arithmetic operations on —
fuzzy outranking relation
[90-XX]
(*see:* **Outranking methods**)
fuzzy power set
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
fuzzy product *see:* harsh —
fuzzy programming
[90C90]
(*see:* **Chemical process planning**)
fuzzy relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
fuzzy relation *see:* α-cut of a —
fuzzy relational product
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)

fuzzy relations
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
fuzzy relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
fuzzy relations *see:* Boolean and —; special properties of —
fuzzy set
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
fuzzy set-inclusion operator
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
fuzzy sets
[03B50, 03B52, 03C80, 03E72, 47S40, 62F30, 62Gxx, 68T27, 68T35, 68Uxx, 90Bxx, 90C29, 90C70, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Fuzzy multi-objective linear programming**)
fuzzy sets
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27, 90C29, 90C70]
(*see:* **Checklist paradigm semantics for fuzzy logics**; **Fuzzy multi-objective linear programming**)
fuzzy sum rule
[58C20, 58E30, 90C46, 90C48]
(*see:* **Nonsmooth analysis: weak stationarity**)
fuzzy triangle product
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
fuzzy truth assessment
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
(FVI) *see:* farthest vertex insertion —

G

g-αBB approach *see:* Global optimization: —
g-basin
[65K05, 90C26, 90C30, 90C59]
(*see:* **Global optimization: filled function methods**)
g-group classification problem
[62H30, 68T10, 90C11]
(*see:* **Mixed integer classification problems**)
g-group classification problem (discriminant problem)
[62H30, 68T10, 90C05]
(*see:* **Linear programming models for classification**)
GA
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
gabriel graph
[68Q20]
(*see:* **Optimal triangulations**)
gadget
[05C85]
(*see:* **Directed tree networks**)
gain *see:* small —
gain legitimacy
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
Gaines implication *see:* Goguen– —
gains *see:* flows with —
Gale–Hoffman inequalities
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**)
Galerkin approach *see:* Petrov– —
Galerkin cone
[90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**)
Galerkin iteration *see:* Petrov– —
Galerkin method *see:* Ritz– —
Galerkin spectral method
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)
gambling *see:* optimal —
game *see:* combinatorial optimization —; complete —; cooperative —; cooperative case of a two-person —; minimax —; noncooperative —; nonzero-sum infinite horizon —; optimality in a —; packing —; polymatrix —; Stackelberg —; two-person —; two-person zero-sum —; two-player zero-sum perfect-information —; von Stackelberg —
game in normal form
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
game with side payments
[90C27, 90C60, 91A12]
(*see:* **Combinatorial optimization games**)
game of strategy
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
game theory
[01A99, 90C27, 90C60, 90C99, 91A12]
(*see:* **Combinatorial optimization games**; **Von Neumann, John**)
game theory
[49M37, 62C20, 90B80, 90B85, 90C15, 90C26, 90C30, 90C31, 90Cxx, 91A10, 91Axx, 91B06, 91B60, 91Bxx]
(*see:* **Bilevel programming**; **Bilevel programming: introduction, history and overview**; **Facility location with externalities**; **Oligopolistic market equilibrium**; **Stochastic programming: minimax approach**)
game theory *see:* evolutionary —; Maximum entropy and —; Stackelberg —
game tree
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
game tree algorithm *see:* sequential minimax —
game tree search algorithm *see:* distributed —; generalized —
game tree searching *see:* Minimax —
games
[01A99]
(*see:* **History of optimization**)
games
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)

games *see:* bimatrix —; Combinatorial optimization —; Infinite horizon control and dynamic —; noncooperative —; theory of —; von Stackelberg —
γ-concave probability measure
[90C15]
(*see:* **Logconcave measures, logconvexity**)
gamma distribution *see:* multivariate —
GAP
[68Q99]
(*see:* **Branch and price: Integer programming with column generation**)
gap *see:* approximation algorithms for —; duality —; integrality —; relative duality —
gap function
[90C15, 90C26, 90C30, 90C33]
(*see:* **Lagrangian duality: BASICS**; **Stochastic bilevel programs**)
gap function *see:* complementary —; primal —; pure complementary —; regularized —
gap theorem
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
gaps in nonconvex optimization *see:* Duality —
Garza method *see:* De La —
gas *see:* allocation of —
gas lift availability *see:* upper bound on —
gas lift wells of type a
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
gas lift wells of type b
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
gas and water capacity constraints *see:* maximum oil —
Gasoline blending and distribution scheduling: an MILP model
(90B35, 93A30)
gâteaux subdifferential
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
gates *see:* logic —
gauge
[90B85]
(*see:* **Multifacility and restricted location problems**)
gauge
[90B85]
(*see:* **Multifacility and restricted location problems**)
Gauss, Carl Friedrich
(01A99)
(*referred to in:* **Gauss–Newton method: Least squares, relation to Newton's method**; **Least squares problems**; **Linear programming**; **Symmetric systems of linear equations**)
(*refers to:* **Gauss–Newton method: Least squares, relation to Newton's method**; **Least squares problems**; **Linear programming**; **Symmetric systems of linear equations**)
Gauss distribution law
[01A99]
(*see:* **Gauss, Carl Friedrich**)
Gauss–Markoff theorem
[65Fxx]
(*see:* **Least squares problems**)
Gauss–Newton method
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
Gauss–Newton method
[90C30, 90C52, 90C53, 90C55]
(*see:* **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**)
Gauss–Newton method *see:* damped —; full-step —
Gauss–Newton method: Least squares, relation to Newton's method
(90C30, 90C30, 90C52, 90C53, 90C55)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Discontinuous optimization**; **Gauss, Carl Friedrich**; **Generalized total least squares**; **Least squares orthogonal polynomials**; **Least squares problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Gauss, Carl Friedrich**; **Generalized total least squares**; **Least squares orthogonal polynomials**; **Least squares problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**)
Gauss problem
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
Gauss quadrature rule *see:* generalized —
Gauss–Seidel
[90C30]
(*see:* **Cost approximation algorithms**)
Gauss–Seidel algorithm
[90C30]
(*see:* **Cost approximation algorithms**)
Gauss–Seidel iteration
[49L20, 90C39]
(*see:* **Dynamic programming: discounted problems**)
Gauss–Seidel method
[90C33]
(*see:* **Linear complementarity problem**)
Gauss–Seidel value iteration
[49L20, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**)
Gauss–Southwell method
[90C30]
(*see:* **Cyclic coordinate method**)
Gauss–Southwell method
[90C30]
(*see:* **Cyclic coordinate method**)
gaussian
(*see:* **Optimal sensor scheduling**)
Gaussian *see:* linear-quadratic —
Gaussian approximation methods
[01A99]
(*see:* **Gauss, Carl Friedrich**)
Gaussian density annealing
[90C90]
(*see:* **Simulated annealing methods in protein folding**)

Gaussian distribution
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
Gaussian elimination
[01A99, 65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Gauss, Carl Friedrich**)
Gaussian elimination
[01A99]
(*see:* **Gauss, Carl Friedrich**)
Gaussian elimination with backsolving
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
Gaussian measure
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
Gaussian quadrature
[33C45, 65F20, 65F22, 65K10, 90C26]
(*see:* **Global optimization in batch design under uncertainty**; **Least squares orthogonal polynomials**)
Gaussianity
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
Gauvin theorem
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
GBD
[49M29, 90C11]
(*see:* **Generalized benders decomposition**)
GBD *see:* variants of —
GC
[90C35]
(*see:* **Graph coloring**)
GCM
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
(GD) function *see:* generalized differentiable —
gDP
[90C10, 90C11, 90C27, 90C33]
(*see:* **Continuous reformulations of discrete-continuous optimization problems**)
Gene clustering: A novel decomposition-based clustering approach: global optimum search with enhanced positioning
(91C20, 90C11, 90C26)
general Algorithm
[90B15]
(*see:* **Evacuation networks**)
general case of the trust region problem
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
general constrained optimization
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
general dynamic programming paradigm
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
general economic equilibrium
[91B50]
(*see:* **Walrasian price equilibrium**)
general equilibrium
[91B50]
(*see:* **Walrasian price equilibrium**)
general Fermat problem
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
general gradient
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms for nonsmooth and stochastic optimization**)
general linear constraints
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
General moment optimization problems
(28-XX, 49-XX, 60-XX)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**;

Simple recourse problem: dual method; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

general order complementarity problem
[90C33]
(*see:* **Topological methods in complementarity theory**)

general order complementarity problem *see:* implicit —

general position
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)

general position of hyperplanes
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)

general purpose
[65H20, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**)

general-purpose software library
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)

general QAP
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)

general quadratic assignment problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)

General routing problem
(90B20)
(*referred to in:* **Stochastic vehicle routing problems**; **Vehicle routing**; **Vehicle scheduling**)
(*refers to:* **Stochastic vehicle routing problems**; **Vehicle routing**; **Vehicle scheduling**)

general routing problem
[90B20]
(*see:* **General routing problem**)

general second order sufficient condition
[90C31]
(*see:* **Sensitivity and stability in NLP: continuity and differential stability**)

general strong second order sufficient condition
[90C31]
(*see:* **Sensitivity and stability in NLP: continuity and differential stability**)

general structure mixed integer αBB algorithm
[65K05, 90C11, 90C26]
(*see:* **MINLP: global optimization with αBB**)

general theory and examples *see:* Derivatives of probability and integral functions: —

general univariate linear model
[65Fxx]
(*see:* **Least squares problems**)

generalization of ELECTRE I
[90-XX]
(*see:* **Outranking methods**)

generalization of Lyusternik theorem *see:* high-order —

generalizations *see:* Farkas lemma: —

Generalizations of interior point methods for the linear complementarity problem
(90C33, 90C51, 65K10)
(*referred to in:* **Complementarity algorithms in pattern recognition**; **Mathematical programming methods in supply chain management**; **Simultaneous estimation and optimization of nonlinear problems**)
(*refers to:* **Complementarity algorithms in pattern recognition**; **Mathematical programming methods in supply chain management**; **Simultaneous estimation and optimization of nonlinear problems**)

generalizations of the nonlinear complementarity problem
[90C33]
(*see:* **Generalized nonlinear complementarity problem**)

generalized
[90C31, 90C34]
(*see:* **Semi-infinite programming: second order optimality conditions**)

Generalized assignment problem
(90-00)
(*referred to in:* **Biquadratic assignment problem**; **Feedback set problems**; **Graph coloring**; **Graph planarization**; **Greedy randomized adaptive search procedures**; **Linear ordering problem**; **Multi-index transportation problems**; **Quadratic assignment problem**; **Quadratic semi-assignment problem**)

generalized assignment problem
[68Q99, 90-00]
(*see:* **Branch and price: Integer programming with column generation**; **Generalized assignment problem**)

generalized assignment problem *see:* multilevel —

generalized barycenters
[90C15]
(*see:* **Multistage stochastic programming: barycentric approximation**)

Generalized benders decomposition
(49M29, 90C11)
(*referred to in:* **Chemical process planning**; **Decomposition principle of linear programming**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation**

column synthesis; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Nondifferentiable optimization**; **Preprocessing in stochastic programming**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)
(*refers to:* **Chemical process planning**; **Decomposition principle of linear programming**; **Extended cutting plane algorithm**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)
generalized Benders decomposition
[49M37, 65K05, 90C10, 90C11, 90C26, 90C29, 90C90]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: global optimization with αBB**; **MINLP: outer approximation algorithm**; **Mixed integer nonlinear programming**; **Multi-objective optimization: interaction of design and control**)
generalized Benders decomposition
[90C09, 90C10, 90C11]
(*see:* **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**)
generalized Benders method
[90C09, 90C10, 90C11]
(*see:* **MINLP: logic-based methods**)
generalized bilevel programming problem
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
generalized bisection algorithm
[65H20, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**)
generalized complementarity
[90C33]
(*see:* **Generalized nonlinear complementarity problem**)
generalized complementarity problem
[47J20, 49J40, 65K10, 90C33]
(*see:* **Generalized nonlinear complementarity problem**; **Solution methods for multivalued variational inequalities**)
generalized concavity *see:* vector —
Generalized concavity in multi-objective optimization
(90C29)
(*referred to in:* **Invexity and its applications**; **L-convex functions and M-convex functions**)
(*refers to:* **Invexity and its applications**; **Isotonic regression problems**)
generalized conditions *see:* Fritz John —
generalized convex function *see:* program of minimizing a —
generalized convexity
[90C26]
(*see:* **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**)
generalized critical point
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
generalized critical point
[49J40]
(*see:* **Nonconvex-nonsmooth calculus of variations**)
generalized critical point of an energy functional
[49J40]
(*see:* **Nonconvex-nonsmooth calculus of variations**)
generalized critical point set
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
generalized cross decomposition
[49M27, 49M37, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**; **Mixed integer nonlinear programming**)
generalized cross decomposition *see:* MINLP: —
generalized cutting plane
[90C26]
(*see:* **Global optimization: envelope representation**)
generalized cutting plane method
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
generalized derivative *see:* clarke —; Clarke–Rockafellar —
generalized differentiable (GD) function
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)
generalized directional derivative
[49J52]
(*see:* **Hemivariational inequalities: eigenvalue problems**)
generalized directional derivative *see:* Clarke —
generalized directional differential
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
Generalized Disjunctive Programming
(*see:* **Logic-based outer approximation**)
Generalized disjunctive programming
[90C09, 90C10, 90C11]
(*see:* Generalized disjunctive programming; **MINLP: logic-based methods**; **Optimal planning of offshore oilfield infrastructure**)
generalized dual problem
[90C30]
(*see:* **Image space approach to optimization**)
generalized eigenvalue proximal support vector machine
[68Q32, 68T10]

(*see:* Generalized eigenvalue proximal support vector machine problem)
Generalized eigenvalue proximal support vector machine problem
[68Q32, 68T10]
(*see:* Generalized eigenvalue proximal support vector machine problem)
generalized-ensemble algorithms *see:* Protein folding: —
Generalized ensembles
[92-08, 92C05, 92C40]
(*see:* **Protein folding: generalized-ensemble algorithms**)
generalized equation
[65K10, 90C31, 90C33]
(*see:* **Linear complementarity problem**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**)
generalized finite sequence
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
generalized flow
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
generalized fractional program
[90C32]
(*see:* **Fractional programming**)
generalized functions *see:* theory of —
generalized game tree search algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
generalized Gauss quadrature rule
[90C05, 90C34]
(*see:* **Semi-infinite programming: methods for linear problems**)
generalized geometric programming
[90C26, 90C90]
(*see:* **Global optimization in generalized geometric programming**)
generalized geometric programming
[90C26, 90C90]
(*see:* **Global optimization in generalized geometric programming**)
generalized geometric programming *see:* Global optimization in —
Generalized geometric programming: mixed continuous and discrete free variables
[49M37, 90C11, 90C30]
(*see:* Generalized geometric programming: mixed continuous and discrete free variables)
generalized gradient
[26E25, 46N10, 49J40, 49J52, 49Q10, 52A27, 65G20, 65G30, 65G40, 65K05, 70-XX, 74K99, 74Pxx, 80-XX, 90-00, 90C30, 90C47, 90C99]
(*see:* **Hemivariational inequalities: eigenvalue problems**; **Interval global optimization**; **Nonconvex energy functions: hemivariational inequalities**; **Nondifferentiable optimization**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
generalized gradient *see:* Clarke —
generalized inverses
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
generalized invex
[90C26]
(*see:* **Invexity and its applications**)
generalized invex
[90C26]
(*see:* **Invexity and its applications**)
generalized Jacobian *see:* Clarke —
generalized Karush–Kuhn–Tucker conditions
[65K10, 90C31]
(*see:* **Sensitivity analysis of variational inequality problems**)
generalized Lagrange multiplier approach *see:* Everett —
generalized least squares problem
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
generalized linear order complementarity problem
[90C33]
(*see:* **Order complementarity**)
generalized linear programming with variable coefficients
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
generalized minimizing sequence
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
generalized mixed complementarity problem
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
Generalized monotone multivalued maps
(90C26)
(*referred to in:* **Fejér monotonicity in convex optimization**; **Generalized monotone single valued maps**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Pseudomonotone maps: properties and applications**; **Set-valued optimization**)
(*refers to:* **Fejér monotonicity in convex optimization**; **Generalized monotone single valued maps**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Set-valued optimization**)
generalized monotone operator
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
Generalized monotone single valued maps
(90C26)
(*referred to in:* **Fejér monotonicity in convex optimization**; **Generalized monotone multivalued maps**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Pseudomonotone maps: properties and applications**; **Set-valued optimization**)
(*refers to:* **Fejér monotonicity in convex optimization**; **Generalized monotone multivalued maps**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Set-valued optimization**)
generalized monotonicity
[90C26]
(*see:* **Generalized monotone single valued maps**)

generalized monotonicity
[46N10, 49J40, 90C26]
(*see:* **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
Generalized monotonicity: applications to variational inequalities and equilibrium problems
(90C26, 49J40, 46N10)
(*referred to in:* **Equilibrium networks**; **Fejér monotonicity in convex optimization**; **Financial equilibrium**; **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Oligopolistic market equilibrium**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Spatial price equilibrium**; **Traffic network equilibrium**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Walrasian price equilibrium**)
(*refers to:* **Equilibrium networks**; **Fejér monotonicity in convex optimization**; **Financial equilibrium**; **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Oligopolistic market equilibrium**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Spatial price equilibrium**; **Traffic network equilibrium**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**; **Walrasian price equilibrium**)
generalized morphism
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
generalized morphism
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
generalized morphisms
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
generalized morphisms of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
generalized necessary optimality conditions
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
generalized network
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)
generalized network optimization system
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
generalized network problem
[90C35]
(*see:* **Generalized networks**)
generalized network problems *see:* quadratic —
Generalized networks
(90C35)
(*referred to in:* **Auction algorithms**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
(*refers to:* **Auction algorithms**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)

generalized networks
[90C35]
(*see:* **Generalized networks**)
generalized networks
[90C30, 90C35]
(*see:* **Convex-simplex algorithm**; **Generalized networks**)
Generalized nonlinear complementarity problem
(90C33)
(*referred to in:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Global optimization methods for systems of nonlinear equations**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Linear complementarity problem**; **Order complementarity**; **Principal pivoting methods for linear complementarity problems**; **Topological methods in complementarity theory**)
(*refers to:* **Convex-simplex algorithm**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Order complementarity**; **Parametric linear programming: cost simplex algorithm**; **Principal pivoting methods for linear complementarity problems**; **Sequential simplex method**; **Topological methods in complementarity theory**)
generalized nonlinear least squares
[90C30]
(*see:* **Generalized total least squares**)
generalized nonlinear least squares problem
[90C30]
(*see:* **Nonlinear least squares problems**)
generalized order complementarity problem
[90C33]
(*see:* **Order complementarity**)
generalized order complementarity problem *see:* infinite-dimensional —
Generalized outer approximation
(90C11, 90C30, 49M20)
(*referred to in:* **Chemical process planning**; **Generalized benders decomposition**; **Global optimization in multiplicative programming**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)
(*refers to:* **Chemical process planning**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)
generalized outer approximation
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
generalized primal-relaxed dual algorithm
[90C26]
(*see:* **Generalized primal-relaxed dual approach**)
Generalized primal-relaxed dual approach
(90C26)
(*referred to in:* **αBB algorithm**; **Global optimization in phase and chemical reaction equilibrium**)
(*refers to:* **αBB algorithm**; **Global optimization in phase and chemical reaction equilibrium**)
generalized quantifier
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
generalized quantifier
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
generalized second order directional derivative
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
generalized semi-infinite problem
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
Generalized semi-infinite programming: optimality conditions
[90C31, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions)
generalized single cluster statistic
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
generalized Slater constraint qualification
[90C31]
(*see:* **Sensitivity and stability in NLP: continuity and differential stability**)
generalized state equations
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
generalized subdifferential
[26B25, 26E25, 49J52, 90C99]
(*see:* **Quasidifferentiable optimization**)
generalized subdifferential *see:* Clarke —
generalized subdifferential of F.H. Clarke
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
Generalized total least squares
(90C30)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Gauss–Newton method: Least squares, relation to Newton's**

method; **Least squares orthogonal polynomials**; **Least squares problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Least squares orthogonal polynomials**; **Least squares problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**)
generalized total least squares
[90C30]
(*see:* **Generalized total least squares**)
generalized TSP
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
generalized-upper-bound dichotomy
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
generalized upper bounding structure
[90C30]
(*see:* **Convex-simplex algorithm**)
generalized upper bounds constraints
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
Generalized variational inequalities: A brief review
[49J53, 90C30]
(*see:* Generalized variational inequalities: A brief review)
generalized variational inequality
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
generalized Weber problem
[65D18, 90B85, 90C26]
(*see:* **Global optimization in location problems**)
generally nonhomogeneous and nonisotropic body *see:* linear thermoelastic behavior of a —
generated *see:* randomly —
generating polynomial
[33C45, 65F20, 65F22, 65K10]
(*see:* **Least squares orthogonal polynomials**)
generating two paths *see:* Pivoting algorithms for linear programming —
generation
[92B05]
(*see:* **Genetic algorithms**; **State of the art in modeling agricultural systems**)
generation
[92B05]
(*see:* **Genetic algorithms**)
generation *see:* Branch and price: Integer programming with column —; Column —; cut —; hydro- —; one-at-a-time coefficient —; row —; scenario —
generation formulation *see:* column —
generation method *see:* Lagrangian finite —
generation methods *see:* column —
generation modeling languages *see:* second —
generation plant *see:* co- —
generation subproblem *see:* column —
generations
(*see:* **Broadcast scheduling problem**)
generator *see:* hit and run —; Li–Pardalos —; Palubeckis —; supremal —
generators
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
generators *see:* Combinatorial test problems and problem —
generic
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
generic augmenting path algorithm
[90C35]
(*see:* **Maximum flow problem**)
generic cost vector
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
generic pivoting rule
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
generic preflow-push algorithm
[90C35]
(*see:* **Maximum flow problem**)
generic property
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
generic shortest path algorithms
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
generic singularities
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
generic transitions
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
generic vertex insertion algorithm
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
genericity
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
GENEROUS
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
genes *see:* Selection of maximally informative —
genetic algorithm *see:* grouping —; simulated annealing and —
Genetic algorithms
(92B05)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Broadcast scheduling problem**; **Evolutionary algorithms in combinatorial optimization**; **Facilities layout problems**; **Forecasting**; **Genetic algorithms for protein structure prediction**; **Global optimization in Lennard–Jones and morse clusters**; **Graph coloring**; **Integer programming: branch and bound methods**; **Job-shop scheduling problem**; **Molecular**

structure determination: convex global underestimation; **Monte-Carlo simulated annealing in protein folding**; **Multidisciplinary design optimization**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Optimization in medical imaging**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Set covering, packing and partitioning problems**; **Simulated annealing methods in protein folding**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms for protein structure prediction**; **Global optimization in Lennard–Jones and morse clusters**; Global optimization in protein folding; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Protein folding: generalized-ensemble algorithms**; **Simulated annealing**; **Simulated annealing methods in protein folding**)
genetic algorithms
[62C10, 65K05, 90B06, 90B35, 90C06, 90C08, 90C10, 90C11, 90C15, 90C26, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90, 92B05]
(*see:* **Bayesian global optimization**; **Design optimization in computational fluid dynamics**; **Genetic algorithms**; **Maximum constraint satisfaction: relaxations and upper bounds**; **Quadratic assignment problem**; **Traveling salesman problem**)
genetic algorithms
[90B80, 90C26, 90C30, 92B05]
(*see:* **Facilities layout problems**; **Forecasting**; **Genetic algorithms**)
Genetic algorithms for protein structure prediction
(92B05)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Genetic algorithms**; **Global optimization based on statistical models**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Genetic algorithms**; **Global optimization based on statistical models**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
genetic engineering via negative fitness
[90C20]
(*see:* **Standard quadratic optimization problems: algorithms**)
genetic operators
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
GENF
[90C20]
(*see:* **Standard quadratic optimization problems: algorithms**)
genomic analysis *see:* Algorithms for —
geodesic convex function
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
geodesic convex set
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
geodesic convexity
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
geodesic convexity
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
geodesic gradient vector
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
geodesic Hessian matrix
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
Geoffrion theorem
[90C10, 90C29]
(*see:* **Multi-objective integer linear programming**)
geometric
[68Q20]
(*see:* **Optimal triangulations**)
geometric algorithms
[05C05, 05C85, 68Q25, 90B80]
(*see:* **Bottleneck steiner tree problems**)
geometric convergence rate
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: subgradient optimization methods**)
geometric distribution
[90C15]
(*see:* **Logconcavity of discrete distributions**)
geometric interpretation
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
geometric interpretation, existence and uniqueness *see:* Variational inequalities: —
geometric mean method
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
geometric mean method *see:* revised —
geometric moment theory
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
geometric (or disk) representation
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
geometric programming
[01A99]
(*see:* **History of optimization**)
Geometric programming
[90C28, 90C30]
(*see:* Geometric programming)

geometric programming *see:* generalized —; Global optimization in generalized —
geometric programming: mixed continuous and discrete free variables *see:* Generalized —
geometric semilattice
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
geometric semilattice
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
geometric series rule
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: subgradient optimization methods**)
geometrical equation
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
geometrical operator
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
geometrical problem
[90C30]
(*see:* **Lagrangian duality: BASICS**)
geometrically
[90C30]
(*see:* **Frank–Wolfe algorithm**)
geometrically linear problem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
geometrically nonlinear problem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
geometry
(*see:* **State of the art in modeling agricultural systems**)
geometry *see:* stochastic —; vector —
geometry problem *see:* distance —; Molecular distance —
geometry of satellites and tracking stations *see:* Automatic differentiation: —
Gershgorin theorem
[49M37, 65K10, 90C09, 90C10, 90C26, 90C30]
(*see:* **αBB algorithm**; **Combinatorial matrix analysis**)
GGA
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
GGP
[90C26, 90C90]
(*see:* **Global optimization in generalized geometric programming**)
Gibbs free energy *see:* molar —; total —
Gibbs function
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
Gibbs sampler *see:* hidden Markov model and —
GIDEON
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
Gilbert conjecture *see:* Chung– —
Gilbert–Pollak conjecture
[90C27]
(*see:* **Steiner tree problems**)
Gilmore–Lawler lower bound
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Gilmore–Lawler type lower bounds
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
gIS design pattern based model
(*see:* **State of the art in modeling agricultural systems**)
given marginals *see:* table with —
Given transformation
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
Givens rotation
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **QR factorization**)
Givens transformation *see:* fast —; square-root-free —
glass model *see:* Ising —; Potts —
global
[65H20, 80A10, 80A22, 90C26, 90C31, 90C34, 90C90, 92-08, 92C05, 92C40]
(*see:* **Generalized primal-relaxed dual approach**; **Global optimization: application to phase equilibrium problems**; **Interval analysis for optimization of dynamical systems**; **Parametric global optimization: sensitivity**; **Protein folding: generalized-ensemble algorithms**)
global constrained optimization problem
[60G35, 65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Differential equations and global optimization**; **Interval global optimization**)
global convergence
[49J52, 49M37, 90C06, 90C30]
(*see:* **Large scale unconstrained optimization**; **Nondifferentiable optimization: Newton method**; **Nonlinear least squares: Newton-type methods**; **Rosen's method, global convergence, and Powell's conjecture**)
global convergence
[49M37, 90C30]
(*see:* **Nonlinear least squares: Newton-type methods**; **Rosen's method, global convergence, and Powell's conjecture**)
global convergence of GRASP
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
global convergence, and Powell's conjecture *see:* Rosen's method —
global convergence problem for the Rosen method
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
global cut
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**)
global energy minimum
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
Global equilibrium search
(*see:* Global equilibrium search)

global error bound
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
global gradient flows
[58E05, 90C30]
(*see:* **Topology of global optimization**)
global independence
(*see:* **Bayesian networks**)
global infimum
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
global Lagrange multiplier rule
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
global local maximizers *see:* set of discrete ε- —
global maximization problem
[90C05, 90C34]
(*see:* **Semi-infinite programming: methods for linear problems**)
global maximizer
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
global maximizer *see:* discrete —
global maximum point
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
global minimization
[03H10, 49J27, 65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26, 90C34]
(*see:* **Information-based complexity and information-based optimization**; **Semi-infinite programming and control problems**)
global minimization
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
global minimizer
[46A20, 52A01, 65G20, 65G30, 65G40, 65K05, 90C30, 90Cxx]
(*see:* **Composite nonsmooth optimization**; **Dini and Hadamard derivatives in optimization**; **Interval global optimization**)
global minimizers
[65G30, 65G40, 65K05, 90C30, 90C57]
(*see:* **Global optimization: interval analysis and balanced interval arithmetic**)
global minimum
[03H10, 49J27, 65G20, 65G30, 65G40, 65K05, 90C26, 90C30, 90C34, 90C39, 90C57]
(*see:* **Global optimization: interval analysis and balanced interval arithmetic**; **Interval global optimization**; **Second order optimality conditions for nonlinear optimization**; **Semi-infinite programming and control problems**)
global minimum
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
global minimum KKT point
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
global minimum of an NNFP
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)
global minimum point
[65K05, 90C30, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Image space approach to optimization**)
global minimum solution
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)
global MINLP
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
global nonlinear optimization
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
global optimal design
[90C26, 90C90]
(*see:* **Global optimization of heat exchanger networks**)
global optimal solution
[65H10, 90C26, 90C30]
(*see:* **Global optimization methods for systems of nonlinear equations**)
global optimality
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
global optimization
[01A99, 26E25, 46A20, 49-XX, 49J52, 52A01, 52A27, 60J15, 60J60, 60J65, 60J70, 60K35, 65C05, 65C10, 65C20, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65G20, 65G30, 65G40, 65K05, 65T40, 68Q25, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 90-XX, 90B50, 90C05, 90C10, 90C20, 90C26, 90C29, 90C30, 90C90, 90C99, 91B06, 92C40, 92E10, 93-XX]
(*see:* **Adaptive global search**; **Duality theory: triduality in global optimization**; **Farkas lemma: generalizations**; **Global optimization methods for harmonic retrieval**; Global optimization in protein folding; **History of optimization**; **Interval analysis: systems of nonlinear equations**; **MINLP: branch and bound global optimization algorithm**; **Multi-objective optimization and decision support systems**; **Quadratic programming with bound constraints**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Reverse convex optimization**; **Selection of maximally informative genes**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
Global optimization
[46A20, 49K99, 49M29, 49M37, 52A01, 58E05, 60G35, 62C10, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65G20, 65G30, 65G40, 65H10, 65H20, 65K05, 65K10, 65K99, 65T40, 80A10, 80A22, 90B06, 90B10, 90C05, 90C10, 90C11, 90C15, 90C20, 90C26, 90C27, 90C29, 90C30, 90C32, 90C35, 90C90, 90C99, 92C40]
(*see:* **αBB algorithm**; **Bayesian global optimization**; **Bilevel programming: global optimization**; **Continuous global optimization: applications**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Farkas lemma: generalizations**; **Generalized benders decomposition**; **Generalized primal-relaxed dual approach**; **Global optimization: application to phase equilibrium problems**; **Global optimization based on statistical models**; **Global optimization: envelope representation**; **Global optimization in generalized geometric programming**; **Global optimization of heat exchanger networks**; **Global**

optimization: hit and run methods; **Global optimization in Lennard–Jones and morse clusters**; **Global optimization methods for harmonic retrieval**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in multiplicative programming**; **Global optimization in phase and chemical reaction equilibrium**; **Global optimization in Weber's problem with attraction and repulsion**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval fixed point theory**; **Interval global optimization**; **Interval Newton methods**; **Minimum concave transportation problems**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Mixed integer nonlinear bilevel programming: deterministic global optimization**; **Mixed integer nonlinear programming**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Neural networks for combinatorial optimization**; **Nonconvex network flow problems**; **Optimal design of composite structures**; **Optimality criteria for multiphase chemical equilibrium**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Piecewise linear network flow problems**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic programming with bound constraints**; **Random search methods**; **Reverse convex optimization**; **SSC minimization algorithms for nonsmooth and stochastic optimization**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: theory**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**; **Topology of global optimization**)

global optimization *see:* Bayesian —; Bilevel programming: —; black-box —; constrained —; continuous —; Cutting plane methods for —; Decomposition in —; Differential equations and —; Duality theory: triduality in —; Interval —; Interval analysis: parallel methods for —; LP strategy for interval-Newton method in deterministic —; mixed discrete-continuous —; Mixed integer nonlinear bilevel programming: deterministic —; multi-extremal —; Reformulation-linearization technique for —; Robust —; stochastic —; Topology of —; unconstrained —

global optimization algorithm *see:* αBB —; deterministic —; Direct —; MINLP: branch and bound —

Global optimization algorithms for financial planning problems [78M50, 90B50, 91B28]
(*see:* Global optimization algorithms for financial planning problems)

global optimization with αBB *see:* MINLP: —

Global optimization in the analysis and management of environmental systems
(90C05)
(*referred to in:* **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Interval global optimization**; **Mixed integer nonlinear programming**; **Optimization in water resources**)
(*refers to:* **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Interval global optimization**; **Mixed integer nonlinear programming**; **Optimization in water resources**)

Global optimization: application to phase equilibrium problems
(80A10, 80A22, 90C90, 65H20)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization in phase and chemical reaction equilibrium**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Optimality criteria for multiphase chemical equilibrium**)
(*refers to:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization in phase and chemical reaction equilibrium**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Optimality criteria for multiphase chemical equilibrium**)

global optimization: applications *see:* Continuous —

global optimization approach *see:* Multiple minima problem in protein folding: αBB —

Global optimization based on statistical models
(90C30)
(*referred to in:* **Adaptive global search**; **Adaptive simulated annealing and its application to protein folding**; **αBB algorithm**; **Bayesian global optimization**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Genetic algorithms for protein structure prediction**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**; **Topology of global optimization**)
(*refers to:* **Adaptive global search**; **Adaptive simulated annealing and its application to protein folding**; **αBB**

algorithm; **Bayesian global optimization**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Genetic algorithms for protein structure prediction**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Integer programming: branch and bound methods**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**; **Topology of global optimization**)

Global optimization in batch design under uncertainty
(90C26)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: models, algorithms and software**; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Smooth nonlinear nonconvex optimization**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: models, algorithms and software**; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Smooth nonlinear nonconvex optimization**)

Global optimization in binary star astronomy
(90C26, 90C90)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Global optimization based on statistical models**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Topology of global optimization**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Global optimization based on statistical models**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Topology of global optimization**)

Global optimization: cutting angle method
(90C26, 65K05, 90C56, 65K10)

Global optimization: envelope representation
(90C26)
(*referred to in:* **Dini and Hadamard derivatives in optimization**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**)
(*refers to:* **Dini and Hadamard derivatives in optimization**; **Farkas lemma**; **Farkas lemma: generalizations**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**)

Global optimization: filled function methods
(90C26, 90C30, 90C59, 65K05)

Global optimization: functional forms

Global optimization: g-αBB approach
(49M37, 65K10, 90C26, 90C30, 46N10, 47N10)

Global optimization in generalized geometric programming
(90C26, 90C90)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: models, algorithms and software**; **Convex envelopes in optimization problems**; **Global optimization in batch design under uncertainty**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Smooth nonlinear nonconvex optimization**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: models, algorithms and software**; **Convex envelopes in optimization problems**; **Global optimization in batch design under uncertainty**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Smooth nonlinear nonconvex optimization**)

Global optimization of heat exchanger networks
(90C26, 90C90)
(*referred to in:* **Continuous global optimization: models, algorithms and software**; **Global optimization methods for systems of nonlinear equations**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**)
(*refers to:* **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**)

Global optimization: hit and run methods
(90C26, 90C90)
(*referred to in:* **Optimal design of composite structures**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Random search methods**; **Simulated annealing**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)

Global optimization: interval analysis and balanced interval arithmetic
(65K05, 90C30, 90C57, 65G30, 65G40)
(*refers to:* **Bisection global optimization methods**; **Continuous global optimization: models, algorithms and software**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: unconstrained and constrained optimization**; **Interval global optimization**; **Interval linear systems**)
Global optimization in Lennard–Jones and morse clusters
(90C26, 90C90)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms**; **Graph coloring**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Simulated annealing methods in protein folding**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms**; Global optimization in protein folding; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Protein folding: generalized-ensemble algorithms**; **Simulated annealing**; **Simulated annealing methods in protein folding**)
Global optimization in location problems
(90C26, 65D18, 90B85)
global optimization method *see:* QBB —
global optimization methods *see:* Bisection —
Global optimization methods for harmonic retrieval
(90C26, 65T40, 90C30, 90C90)
(*referred to in:* **Signal processing with higher order statistics**)
(*refers to:* **Signal processing with higher order statistics**)
Global optimization methods for systems of nonlinear equations
(65H10, 90C26, 90C30)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Contraction-mapping**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Global optimization based on statistical models**; **Global optimization in batch design under uncertainty**; **Global optimization in binary star astronomy**; **Global optimization in generalized geometric programming**; **Global optimization in phase and chemical reaction equilibrium**; **Global optimization using space filling**; **Gröbner bases for polynomial equations**; **Interval analysis: systems of nonlinear equations**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Nonlinear least squares: Newton-type methods**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**; **Smooth nonlinear nonconvex optimization**; **Topology of global optimization**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Contraction-mapping**; **Convex envelopes in optimization problems**; **D.C. programming**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Generalized nonlinear complementarity problem**; **Global optimization based on statistical models**; **Global optimization in batch design under uncertainty**; **Global optimization in binary star astronomy**; **Global optimization in generalized geometric programming**; **Global optimization of heat exchanger networks**; **Global optimization in phase and chemical reaction equilibrium**; **Global optimization using space filling**; **Interval analysis: systems of nonlinear equations**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Nonlinear least squares: Newton-type methods**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**; **Smooth nonlinear nonconvex optimization**; **Topology of global optimization**; **Variational inequalities**)
global optimization model *see:* continuous —
global optimization: models, algorithms and software *see:* Continuous —
Global optimization in multiplicative programming
(90C26)
(*referred to in:* **Linear programming**; **Multiparametric linear programming**; **Multiplicative programming**; **Parametric linear programming: cost simplex algorithm**)
(*refers to:* **Complexity classes in optimization**; **Complexity theory**; **Concave programming**; **Generalized outer approximation**; **Integer programming: branch and bound methods**; **Linear programming**; **Multiparametric linear programming**; **Multiplicative programming**; **Parametric linear programming: cost simplex algorithm**)
Global optimization: p-αBB approach
(49M37, 65K10, 90C26, 90C30, 46N10, 47N10)
Global optimization in phase and chemical reaction equilibrium
(90C26, 90C90)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: models, algorithms and software**; **Generalized primal-relaxed dual approach**; **Global optimization: application to phase equilibrium problems**; **Global optimization in batch design under uncertainty**; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Optimality criteria for multiphase chemical equilibrium**; **Smooth nonlinear nonconvex optimization**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: models, algorithms and software**; **Generalized primal-relaxed dual approach**; **Global optimization: application to phase equilibrium problems**; **Global**

optimization in batch design under uncertainty; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Optimality criteria for multiphase chemical equilibrium**; **Smooth nonlinear nonconvex optimization**)

Global optimization of planar multilayered dielectric structures
(65K99)

global optimization problem
[49K99, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65H10, 65K05, 80A10, 90C26, 90C27, 90C30]
(*see:* **Global optimization methods for systems of nonlinear equations**; **Neural networks for combinatorial optimization**; **Optimality criteria for multiphase chemical equilibrium**; **Random search methods**; **Stochastic global optimization: two-phase methods**)

Global optimization in protein folding
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)

global optimization: sensitivity *see:* Parametric —

global optimization: stopping rules *see:* Stochastic —

Global optimization: tight convex underestimators
[46N10, 47N10, 49M37, 65K10, 90C26, 90C30]
(*see:* Global optimization: tight convex underestimators)

global optimization: two-phase methods *see:* Stochastic —

global optimization over unbounded domains
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)

Global optimization using space filling
(90C26)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Global optimization based on statistical models**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Topology of global optimization**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Global optimization based on statistical models**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Topology of global optimization**)

global optimization using terrain/funneling methods *see:* Multi-scale —

Global optimization in Weber's problem with attraction and repulsion
(90C26, 90C90)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)

global optimizer
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)

global optimum
[65F10, 65F50, 65G20, 65G30, 65G40, 65H10, 65H20, 65K10, 93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**; **Dynamic programming: optimal control applications**; **Globally convergent homotopy methods**; **Interval analysis: unconstrained and constrained optimization**)

global optimum *see:* constrained —

global optimum search
[49M29, 90C11]
(*see:* **Generalized benders decomposition**)

global optimum search with enhanced positioning *see:* Gene clustering: A novel decomposition-based clustering approach: —

Global pairwise protein sequence alignment via mixed-integer linear optimization
[90C10, 92-08]
(*see:* Global pairwise protein sequence alignment via mixed-integer linear optimization)

global phase
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: two-phase methods**)

global points *see:* set of ε- —

global round robin request
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)

global search
[65K05, 90C26, 90C30, 90C59, 90C90]
(*see:* **Global optimization in binary star astronomy**; **Global optimization: filled function methods**)

global search *see:* Adaptive —

global solution
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)

global structual stability of SIP(f,h,g)
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
global structural stability
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
global supply chain
[90B05, 90B06]
(*see:* **Global supply chain models**)
Global supply chain models
(90B05, 90B06)
(*referred to in:* **Inventory management in supply chains**; **Nonconvex network flow problems**; **Operations research models for supply chain management and design**; **Piecewise linear network flow problems**)
(*refers to:* **Inventory management in supply chains**; **Nonconvex network flow problems**; **Operations research models for supply chain management and design**; **Piecewise linear network flow problems**)
Global terrain methods
(*see:* Global terrain methods)
global unconstrained optimization problem
[60G35, 65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Differential equations and global optimization**; **Interval global optimization**)
global underestimation *see:* convex —; Molecular structure determination: convex —
global underestimator
[90C11, 90C26]
(*see:* **Extended cutting plane algorithm**)
global underestimator *see:* convex —
globally convergent
[49M37, 65F10, 65F50, 65H10, 65K05, 65K10, 90C30]
(*see:* **Globally convergent homotopy methods**; **Nonlinear least squares: trust region methods**; **Structural optimization**)
globally convergent
[49M37, 65F10, 65F50, 65H10, 65K05, 65K10, 90C30]
(*see:* **Globally convergent homotopy methods**; **Structural optimization**)
globally convergent algorithm
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
globally convergent homotopies *see:* probability-one —
Globally convergent homotopy methods
(65F10, 65F50, 65H10, 65K10)
(*referred to in:* **Parametric optimization: embeddings, path following and singularities**; **Topology of global optimization**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Parametric optimization: embeddings, path following and singularities**; **Topology of global optimization**)
globally convergent probability-one homotopy algorithm
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
globally convexized filled function
[65K05, 90C26, 90C30, 90C59]
(*see:* **Global optimization: filled function methods**)
globally optimal
[90C30]
(*see:* **Frank–Wolfe algorithm**; **Simplicial decomposition**)
globally optimal parameter
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
GlobSol
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
GMIN
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)
GMIN-αBB
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
GMIN-αBB algorithm
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
GMMVM
[90C26]
(*see:* **Generalized monotone multivalued maps**)
GMRES
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
GMSVM
[90C26]
(*see:* **Generalized monotone single valued maps**)
GMVIPEP
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
GN
[90C35]
(*see:* **Generalized networks**)
go *see:* cost-to- —
GO for SNE
[65H10, 90C26, 90C30]
(*see:* **Global optimization methods for systems of nonlinear equations**)
GO4BSA
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
goal coordination method
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
goal-oriented differentiation
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
goal programming
[90C27, 90C29]
(*see:* **Decision support systems with multiple criteria**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**)
goal programming
[90C27, 90C29]
(*see:* **Multicriteria sorting methods**; **Operations research and financial markets**)

Goal Programming *see:* Lexicographic —
goals
(*see:* **Planning in the process industry**)
goals *see:* fuzzy —
Goerisch bound *see:* Lehmann– —
Goerisch method
[49R50, 65G20, 65G30, 65G40, 65L15, 65L60]
(*see:* **Eigenvalue enclosures for ordinary differential equations**)
Goguen–Gaines implication
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
golden section method
[90C30]
(*see:* **Convex-simplex algorithm**)
golden section method
[90C30]
(*see:* **Convex-simplex algorithm**; **Frank–Wolfe algorithm**)
golden section search
[90C30]
(*see:* **Nonlinear least squares problems**)
Goldfarb–Idnani active set strategy
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
Goldfarb–Idnani method
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
Goldfarb method *see:* Forrest– —
Goldfarb–Shanno method *see:* Broyden–Fletcher– —
Goldfarb–Shanno quasi-Newton update *see:* Broyden–Fletcher– —
Goldfarb–Shanno update *see:* Broyden–Fletcher– —
Goldfarb–Wang algorithm
[90C30]
(*see:* **Numerical methods for unary optimization**)
Goldfarb–Wang algorithm
[90C30]
(*see:* **Numerical methods for unary optimization**)
Goldstein conditions
[49M37, 90C30]
(*see:* **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**)
Gollan CQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
Gomez #3 problem
[65K05]
(*see:* **Direct global optimization algorithm**)
Gomory cut *see:* Chvátal– —
Gomory cutting plane *see:* Chvátal– —
Gomory cutting plane algorithm
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
Gomory relaxations
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
good edge flips
[68Q20]
(*see:* **Optimal triangulations**)
good inclusion function
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
good subset
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
Goodman–Kruskal τ_b statistic
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
gOP algorithm
[90C26, 90C30, 90C90]
(*see:* **Bilevel programming: global optimization**; **Generalized primal-relaxed dual approach**)
GOP algorithm
[90C26]
(*see:* **Generalized primal-relaxed dual approach**)
gOR
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**; **Optimal planning of offshore oilfield infrastructure**)
Gordan transposition theorem
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
gOS
[68T20, 68W10, 90C11, 90C26, 91C20, 92-08, 92C05, 92D10]
(*see:* **Determining the optimal number of clusters**)
GOSF
[90C26]
(*see:* **Global optimization using space filling**)
gottfried wilhelm *see:* Leibniz —
Gottfried Wilhelm Leibniz
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
governing equation
[49K20, 49M99, 90C55]
(*see:* **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
government regulation
[90-01, 90B30, 90B50, 91B32, 91B52, 91B74]
(*see:* **Bilevel programming in management**)
GPASP
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
GPP
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
GPRD
[90C26]
(*see:* **Generalized primal-relaxed dual approach**)
gradient
[49M29, 58E05, 65H99, 65K05, 65K10, 65K99, 90C06, 90C25, 90C30, 90C31]
(*see:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: point and interval**; **Local attractors for gradient-related descent iterations**; **Sensitivity and stability in NLP**; **Solving large scale and sparse semidefinite programs**; **Topology of global optimization**)
gradient
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**)

gradient *see:* adjoint-based —; Clarke generalized —; conjugate —; general —; generalized —; ν-approximate —; projected negative —; projected positive —; reduced —; restricted —; sensitivity-based —
gradient algorithm *see:* extra- —; projected —; reduced —
gradient algorithms for unconstrained optimization *see:* New hybrid conjugate —; Performance profiles of conjugate- —
gradient based approach
[90C30, 90C52, 90C53, 90C55]
(*see:* **Gauss–Newton method: Least squares, relation to Newton's method**)
gradient based procedures
[90C30]
(*see:* **Powell method**)
gradient controller *see:* feasible —; nonfeasible —
gradient descent
[65K05, 90Cxx]
(*see:* **Symmetric systems of linear equations**)
gradient estimation
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
gradient flows *see:* global —
gradient free *see:* univariate —
gradient-free algorithm
[90C30]
(*see:* **Cyclic coordinate method**)
gradient-free minimization
[90C30]
(*see:* **Powell method**; **Rosenbrock method**; **Sequential simplex method**)
gradient-free minimization algorithm
[90C30]
(*see:* **Powell method**; **Rosenbrock method**)
gradient index
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
gradient of an integral
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
gradient method *see:* Arrow–Hurwicz —; conditional —; conjugate —; incremental —; Shanno conjugate —; Wolfe reduced —
gradient methods
[90C30, 90C52, 90C53, 90C55]
(*see:* **Gauss–Newton method: Least squares, relation to Newton's method**)
gradient methods *see:* Conjugate- —; deflected —; Spectral projected —
gradient of multivariate distribution functions
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
gradient parameter computation *see:* conjugate —
gradient of a probability function
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
gradient of a probability function
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
gradient projection
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
gradient projection
[90C30]
(*see:* **Cost approximation algorithms**)
gradient projection algorithm
[90C30]
(*see:* **Cost approximation algorithms**)
gradient projection method *see:* Rosen —
gradient-related descent
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
gradient-related descent iterations *see:* Local attractors for —
gradient-related set function
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
gradient type algorithm *see:* Craig conjugate —
gradient vector
[37A35, 49M29, 65K10, 90C05, 90C06, 90C26, 90C39]
(*see:* **Local attractors for gradient-related descent iterations**; **Potential reduction methods for linear programming**; **Second order optimality conditions for nonlinear optimization**)
gradient vector *see:* geodesic —
gradients *see:* conjugate —
gradients, Jacobians, and Hessians *see:* Complexity of —
Gragg–Kaufmann–Stewart reorthogonalized Gram–Schmidt algorithm *see:* Daniel– —
Graham–Hwang conjecture
[90C27]
(*see:* **Steiner tree problems**)
graham-Scan
[46N10, 47N10, 49M37, 65K10, 90C26, 90C30]
(*see:* Global optimization: tight convex underestimators)
grained multicomputer *see:* coarse —
GRAM
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
Gram–Schmidt algorithm *see:* Daniel–Gragg–Kaufmann–Stewart reorthogonalized —
Gram–Schmidt orthogonalization
[65Fxx]
(*see:* **Least squares problems**)
Gram–Schmidt orthogonalization
[90C30]
(*see:* **Rosenbrock method**)
Gram–Schmidt orthogonalization *see:* classical —; modified —
Gram–Schmidt type iteration
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
grand challenge
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
graph
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)

graph
[05C60, 05C69, 37B25, 90B80, 90C10, 90C20, 90C27, 90C35, 90C59, 91A22, 94C15]
(*see:* **Facilities layout problems**; **Graph coloring**; **Graph planarization**; **Replicator dynamics in combinatorial optimization**)
graph *see:* 0–1–0 —; adjacency —; adjacent vertices in a —; alignment-distribution —; association —; bipartite —; bipartite chordal —; block-clique —; branchpoint of a —; cardinality of a —; chordal —; cocomparability —; column incidence —; compact —; complement —; complementary —; complete —; computational —; conflict —; connected —; connectivity —; constraint —; convex bipartite —; cycle in a —; cyclically reducible —; directed —; distance in a —; edge of a —; elimination —; endpoint of a —; Eulerian —; extended alignment —; gabriel —; homeomorph of a —; interval —; k-leveled —; kernel of a —; length of a path in a —; level in a leveled —; level planar —; leveled —; linkpoint of a —; maximum weighted planar —; min-max —; minimum fill-in of a —; mixed —; order of a —; overlap —; path in a —; permutation —; planar —; proper k-leveled —; rank determined —; series-parallel —; size of a —; Smith–Walford one-reducible —; trapezoid —; tree association —; unicursal —; vertex of a —; weighted —; weighted tree association —
graph based framework
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
graph bipartitioning problem
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
graph bipartization
[90C35]
(*see:* **Feedback set problems**)
graph bipartization problem
[90C35]
(*see:* **Feedback set problems**)
graph bipartization problem *see:* minimum weighted —
graph collapsing auction algorithm
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
graph collapsing in auction algorithms
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
Graph coloring
(90C35)
(*referred to in:* **Biquadratic assignment problem**; **Broadcast scheduling problem**; **Feedback set problems**; **Frequency assignment problem**; **Graph planarization**; **Greedy randomized adaptive search procedures**; **Heuristics for maximum clique and independent set**; **Integer programming**; **Linear ordering problem**; **Maximum constraint satisfaction: relaxations and upper bounds**; **Multi-objective mixed integer programming**; **Quadratic assignment problem**; **Quadratic semi-assignment problem**; **Replicator dynamics in combinatorial optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Time-dependent traveling salesman problem**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Feedback set problems**; **Frequency assignment problem**; **Generalized assignment problem**; **Genetic algorithms**; **Global optimization in Lennard–Jones and morse clusters**; Global optimization in protein folding; **Graph planarization**; **Greedy randomized adaptive search procedures**; **Heuristics for maximum clique and independent set**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Maximum constraint satisfaction: relaxations and upper bounds**; **Mixed integer classification problems**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Parametric mixed integer nonlinear optimization**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Protein folding: generalized-ensemble algorithms**; **Quadratic assignment problem**; **Quadratic semi-assignment problem**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Simulated annealing methods in protein folding**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
graph coloring
[05-XX]
(*see:* **Frequency assignment problem**)
graph coloring problem
[05-04, 05-XX, 90C27, 90C35]
(*see:* **Evolutionary algorithms in combinatorial optimization**; **Frequency assignment problem**; **Graph coloring**)
graph coloring problem *see:* weighted —
graph degree
[90C35]
(*see:* **Feedback set problems**)
graph drawing
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
graph drawing *see:* automatic —
graph isomorphism
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
graph isomorphism problem
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
graph of a matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)

graph model *see:* containment —; intersection —; proximity —
graph optimization
[90C09, 90C10, 90C11, 90C20]
(*see:* **Linear ordering problem**; **Matroids**; **Oriented matroids**)
graph packing problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Graph partitioning
(90C27, 05-04)
(*referred to in:* **Bayesian networks**; **Beam selection in radiotherapy treatment design**; **Combinatorial matrix analysis**; **Combinatorial optimization algorithms in resource allocation problems**; **Combinatorial optimization games**; **Fractional combinatorial optimization**; **Multi-objective combinatorial optimization**; **Optimization-based visualization**; **Replicator dynamics in combinatorial optimization**; **Simulated annealing**; **Traveling salesman problem**)
(*refers to:* **Combinatorial matrix analysis**; **Combinatorial optimization games**; **Fractional combinatorial optimization**; **Genetic algorithms**; **Multi-objective combinatorial optimization**; **Neural networks for combinatorial optimization**; **Replicator dynamics in combinatorial optimization**)
graph partitioning problem
[05C60, 05C69, 37B25, 90C08, 90C11, 90C20, 90C27, 90C35, 90C57, 90C59, 91A22]
(*see:* **Quadratic assignment problem**; **Replicator dynamics in combinatorial optimization**)
graph partitioning problem *see:* k-way —
Graph planarization
(94C15, 90C10, 90C27)
(*referred to in:* **Biquadratic assignment problem**; **Feedback set problems**; **Graph coloring**; **Greedy randomized adaptive search procedures**; **Linear ordering problem**; **Optimization in leveled graphs**; **Quadratic assignment problem**; **Quadratic semi-assignment problem**)
(*refers to:* **Feedback set problems**; **Generalized assignment problem**; **Graph coloring**; **Greedy randomized adaptive search procedures**; **Optimization in leveled graphs**; **Quadratic assignment problem**; **Quadratic semi-assignment problem**)
graph planarization
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
graph planarization *see:* branch and bound algorithm for weighted —
graph problem
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
graph Realization Problem
[05C50, 15A48, 15A57, 51K05, 52C25, 68Q25, 68U05, 90C22, 90C25, 90C35]
(*see:* Graph realization via semidefinite programming; **Matrix completion problems**)
Graph realization problem
(*see:* **Semidefinite programming and the sensor network localization problem, SNLP**)
Graph realization via semidefinite programming
[51K05, 52C25, 68Q25, 68U05, 90C22, 90C35]
(*see:* Graph realization via semidefinite programming)
graph reduction in auction algorithms
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
graph search
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
graph search
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
graph theorem *see:* strong perfect —
graph theory
[05-XX]
(*see:* **Frequency assignment problem**)
graph theory
[90B80, 90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**; **Facilities layout problems**)
graphic matroid
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C, 90C09, 90C10]
(*see:* **Convex discrete optimization**; **Matroids**)
graphical traveling salesman problem
[90B20]
(*see:* **General routing problem**)
graphical traveling salesman problem *see:* Steiner —
graphical user interface
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
graphs *see:* bisectored unit disk —; bounded ratio disk —; coin —; disk —; double disk —; empty neighborhood —; grid —; isomorphic —; leveled —; matrix patterns and —; Optimization in leveled —; Optimization problems in unit-disk —; searching state space —; unit-Disk —
GRASP
[03B05, 65H20, 65K05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90-01, 90B40, 90C09, 90C10, 90C27, 90C35, 94C10, 94C15]
(*see:* **Greedy randomized adaptive search procedures**; **Maximum satisfiability problem**)
GRASP
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Feedback set problems**; **Greedy randomized adaptive search procedures**)
GRASP *see:* basic —; construction phase in —; global convergence of —; local search phase in —; long-term memory in —; parallel —; reactive —
GRASP approach
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
GRASP in hybrid metaheuristics
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
GRASP in industry
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
GRASP in operations research
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)

graver basis
[05A, 13Cxx, 13Pxx, 14Qxx, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C, 90Cxx]
(*see:* **Convex discrete optimization**; **Integer programming: algebraic methods**)
graver complexity
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
gravity *see:* center of —
gravity location *see:* center of —
gravity method *see:* center of —
Gray code
[92B05]
(*see:* **Genetic algorithms**)
Gray code
[92B05]
(*see:* **Genetic algorithms**)
greater *see:* lexicographically —
greatest convex minorant
[41A30, 47A99, 62G07, 62G30, 62J02, 65K05, 65K10, 90C26]
(*see:* **Isotonic regression problems**; **Lipschitzian operators in best approximation by bounded or continuous functions**; **Regression by special functions: algorithms and complexity**)
greatest improvement *see:* rule of —
greatest K-minorant
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
greatest quasiconvex minorant
[41A30, 47A99, 62J02, 65K10, 90C26]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**; **Regression by special functions: algorithms and complexity**)
greedy algorithm
[68Q25, 68R10, 68W40, 90B06, 90B35, 90C06, 90C09, 90C10, 90C27, 90C35, 90C39, 90C57, 90C59, 90C60, 90C90, 94C15]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**; **Domination analysis in combinatorial optimization**; **Graph planarization**; **Multi-index transportation problems**; **Traveling salesman problem**)
greedy algorithm
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05, 90C35]
(*see:* **Maximum partition matching**; **Multi-index transportation problems**)
greedy algorithm *see:* the —
greedy algorithm for axial MITPs
[90C35]
(*see:* **Multi-index transportation problems**)
greedy algorithms
[05C85]
(*see:* **Directed tree networks**)
greedy algorithms *see:* local —
greedy choice
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
greedy-choice property
[90C09, 90C10]
(*see:* **Matroids**)
greedy coloring
[05-XX]
(*see:* **Frequency assignment problem**)
greedy coloring heuristic *see:* sequential —
greedy construction
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
greedy-expanding *see:* algorithm —
greedy form *see:* standard —
greedy function
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
greedy heuristics
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
greedy heuristics *see:* sequential —
greedy method
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
greedy randomized adaptive search
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
greedy randomized adaptive search procedure
[90C35]
(*see:* **Feedback set problems**)
Greedy randomized adaptive search procedures
(90-01, 90B40, 90C10, 90C35, 90C27, 94C15, 65H20, 65K05)
(*referred to in:* **Biquadratic assignment problem**; **Broadcast scheduling problem**; **Feedback set problems**; **Graph coloring**; **Graph planarization**; **Heuristics for maximum clique and independent set**; **Linear ordering problem**; **Maximum cut problem, MAX-CUT**; **Maximum satisfiability problem**; **Quadratic assignment problem**; **Quadratic semi-assignment problem**; **Replicator dynamics in combinatorial optimization**)
(*refers to:* **Feedback set problems**; **Generalized assignment problem**; **Graph coloring**; **Graph planarization**; **Heuristics for maximum clique and independent set**; **Maximum satisfiability problem**; **Quadratic assignment problem**; **Quadratic semi-assignment problem**)
greedy swap
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
greedy swaps *see:* monotone sequence of —
greedy technique
[90C09, 90C10]
(*see:* **Matroids**)
greedy technique
[90C09, 90C10, 90C11, 90C20]
(*see:* **Linear ordering problem**; **Matroids**; **Oriented matroids**)
greedy triangulation
[68Q20]
(*see:* **Optimal triangulations**)
grid
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
grid
[65K05, 65Y05]
(*see:* **Parallel computing: models**)

grid *see:* 2-dimensional —; coarse —; finer —; menace of the expanding —
grid graphs
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
grid point
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
grid search
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
grid technique *see:* uniform —
grids *see:* repertory —
Gröbner bases for polynomial equations
(12D10, 12Y05, 13P10)
(*referred to in:* **Fundamental theorem of algebra**)
(*refers to:* **Contraction-mapping**; **Farkas lemma**; **Farkas lemma: generalizations**; **Fundamental theorem of algebra**; **Global optimization methods for systems of nonlinear equations**; **Interval analysis: systems of nonlinear equations**; **Nonlinear least squares: Newton-type methods**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
Gröbner basis
[12D10, 12Y05, 13Cxx, 13P10, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Gröbner bases for polynomial equations**; **Integer programming: algebraic methods**)
Gröbner basis
[12D10, 12Y05, 13Cxx, 13P10, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Gröbner bases for polynomial equations**; **Integer programming: algebraic methods**)
Gröbner basis *see:* reduced —; universal —
Gröbner fan
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
Gröbner fiber
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
groove constraint *see:* tongue-and- —
ground *see:* departure- —
ground arcs
(*see:* **Railroad locomotive scheduling**)
ground connection arc *see:* arrival- —
ground delay problem in air traffic control
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
ground delay programs *see:* air traffic control and —
ground-departure connection arc
(*see:* **Railroad locomotive scheduling**)
ground node *see:* arrival- —
ground set
[90C09, 90C10]
(*see:* **Matroids**)
ground state
[90C27, 90C90]
(*see:* **Simulated annealing**)
groundwater *see:* rational use of —
groundwater pumping facilities
[90C30, 90C35]
(*see:* **Optimization in water resources**)
groundwater resources *see:* surface and —
groundwater systems *see:* surface and —
group *see:* braid —; depot —; fundamental —; higher homotopy —; Klein 4-element —; quantum —; realization of an abstract —; symmetric $S_{2\times 2\times 2}$ —
group classification problem *see:* g- —
group classification problem (discriminant problem) *see:* g- —
group decision making
[90B50]
(*see:* **Optimization and decision support systems**)
group decision making
[90B50]
(*see:* **Optimization and decision support systems**)
group decision support system *see:* multicriteria —
group relaxation
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
group relaxation
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
group relaxation in integer programming
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
group relaxations *see:* extended —
group of transformation *see:* Piaget —
grouping genetic algorithm
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
groupoid *see:* nonassociative —; noncommutative —
groups *see:* PCR —
growing technique *see:* sphere —
growth *see:* Ramsey rule of economic —; second order —
growth condition *see:* linear —; unilateral —
growth scheme
[90C26, 90C90]
(*see:* **Global optimization in Lennard–Jones and morse clusters**)
GRP
[90B20]
(*see:* **General routing problem**)
GRR
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
GRR-M
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
Gsat algorithm
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
GSC
[90B05, 90B06]
(*see:* **Global supply chain models**)
GSEARCH
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
GSIP
[90C31, 90C34]
(*see:* **Semi-infinite programming: second order optimality conditions**)

guarantee
(*see:* **Interval analysis for optimization of dynamical systems**)
guarantee *see:* performance —; worst-case performance —
guaranteed to be stable without pivoting
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
guaranteed bound
[90C26, 90C30]
(*see:* **Bounding derivative ranges**)
guaranteed bound to optimality
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
guaranteed lower bound
[65K05, 90C11, 90C26]
(*see:* **MINLP: global optimization with αBB**)
guardian
[90C05]
(*see:* **Ellipsoid method**)
GUB dichotomy
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
GUHA
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
guided learning
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
guiding process
[68T20, 68T99, 90C27, 90C59]
(*see:* **Capacitated minimum spanning trees**; **Metaheuristics**)
guillotine subdivision
[90C27]
(*see:* **Steiner tree problems**)
gVI
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
GVI *see:* dual (or Minty) —
gVNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)

H

H-convex function
[90C26]
(*see:* **Global optimization: envelope representation**)
h-convex functions
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
H-matrix
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)
$\mathcal{H}$-representation
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
H-subdifferential
[90C26]
(*see:* **Global optimization: envelope representation**)
H-subgradient
[90C26]
(*see:* **Global optimization: envelope representation**)
Hadamard
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hadamard ∞-stationary point
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hadamard codifferentiable function
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
Hadamard conditional lower derivative
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hadamard conditional upper derivative
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hadamard conditionally differentiable function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hadamard conditionally directionally differentiable function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hadamard derivative
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Quasidifferentiable optimization: optimality conditions**)
Hadamard derivatives in optimization *see:* Dini and —
Hadamard differentiable
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hadamard differentiable function
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
Hadamard directional derivatives
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
hadamard directionally differentiable
[65K05, 90C30, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Minimax: directional differentiability**)
Hadamard directionally differentiable function
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
Hadamard lower directional derivative
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hadamard quasidifferentiable function
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
Hadamard quasidifferential
[90Cxx]

(*see:* **Quasidifferentiable optimization: optimality conditions**)
Hadamard steepest ascent direction
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hadamard steepest descent direction
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hadamard sup-stationary point
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hadamard upper directional derivative
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Hahn–Banach linear extension theorem
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
Hahn–Banach theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 90C15, 91A05]
(*see:* **Minimax theorems**; **Stochastic programming: nonanticipativity and lagrange multipliers**)
Hahn–Banach theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
Hahn–Banach theorem *see:* Mazur–Orlicz version of the —
Hahn decomposition *see:* Jordan– —
HAMILTON CIRCUIT
[90C60]
(*see:* **Complexity classes in optimization**)
Hamilton–Jacobi–Bellman equation
(49L20, 34H05, 90C39)
(*referred to in:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **High-order maximum principle for abnormal extremals**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Pontryagin maximum principle**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **High-order maximum principle for abnormal extremals**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Multiple objective dynamic programming**; **Neuro-dynamic programming**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Pontryagin maximum principle**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
Hamilton–Jacobi–Bellman equation
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)
Hamilton–Jacobi–Bellman equation
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)
Hamilton–Jacobi–Bellman equation *see:* derivation of the —; solution of the —; sufficiency theorem for the —
Hamilton–Jacobi equation
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
Hamilton–Jacobi inequality
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
Hamiltonian
[49-XX, 49K05, 49K10, 49K15, 49K20, 65L99, 90-XX, 93-XX]
(*see:* **Boundary condition iteration BCI**; **Duality in optimal control with first order differential equations**; **Duality theory: biduality in nonconvex optimization**; **Optimization strategies for dynamic systems**)
Hamiltonian circuit
[90C60]
(*see:* **Computational complexity theory**)
Hamiltonian circuit problem
[90C60]
(*see:* **Computational complexity theory**)
Hamiltonian cycle problem
[90C60]
(*see:* **Complexity theory**)
Hamiltonian function
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
Hamiltonian path cost
[90C35]
(*see:* **Multi-index transportation problems**)
Hamiltonian system
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)

Hamiltonian system
[49-XX, 49J15, 49K15, 90-XX, 93-XX, 93C10]
(*see:* **Duality theory: biduality in nonconvex optimization**; **Pontryagin maximum principle**)
Hammerstein equation
[65H10, 65J15]
(*see:* **Contraction-mapping**)
Hamming-reactive tabu search
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
hand side changes *see:* sensitivity analysis with respect to right- —
hand side perturbation model *see:* right- —
hand side perturbation problem *see:* right- —
hand side problem *see:* right- —
hand side simplex algorithm *see:* parametric right- —
hand-side uncertainty, duality and applications *see:* Robust linear programming with right- —
handbook on Semidefinite Programming
[90C05, 90C22, 90C25, 90C30, 90C51]
(*see:* **Interior point methods for semidefinite programming**)
handcoded derivatives
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
handcoding
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
Hansel chain
[90C09]
(*see:* **Inference of monotone boolean functions**)
Hansel chain
[90C09]
(*see:* **Inference of monotone boolean functions**)
Hansel chains question-asking strategy *see:* binary search- —; sequential —
Hansel theorem
[90C09]
(*see:* **Inference of monotone boolean functions**)
Hansel theorem
[90C09]
(*see:* **Inference of monotone boolean functions**)
hard *see:* NP- —; strongly NP- —
hard case of the trust region problem
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
hard clustering
[65K05, 90C26, 90C56, 90C90]
(*see:* **Derivative-free methods for non-smooth optimization**; **Nonsmooth optimization approach to clustering**)
hard language *see:* F- —
hard problem *see:* NP- —
hard problems *see:* classification of —
hardness *see:* F- —
Hardy–Littlewood–Pólya theorem
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
Harker–Kanzow–Smale function *see:* Chen– —
harmonic retrieval
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
harmonic retrieval
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
harmonic retrieval *see:* Global optimization methods for —
harsh fuzzy product
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
Hasse diagram
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
HCP
[90C60]
(*see:* **Computational complexity theory**)
hDY
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)
hDYz
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)
head-body-tail problem
[90B35]
(*see:* **Job-shop scheduling problem**)
head(l)
(*see:* **Railroad crew scheduling**)
head of operation
[90B35]
(*see:* **Job-shop scheduling problem**)
heads *see:* cluster- —; tape —
HEAP *see:* binary —; S- —
Hearn algorithm *see:* Elzinga– —
heat conduction
[35R70, 47S40, 74B99, 74D99, 74G99, 74H99]
(*see:* **Quasidifferentiable optimization: applications to thermoelasticity**)
heat conduction *see:* Fourier law of —
heat exchange *see:* mass and —
heat exchanger
[90C90]
(*see:* **MINLP: heat exchanger network synthesis**)
heat exchanger network *see:* mass and —
heat exchanger network superstructure
[90C90]
(*see:* **MINLP: heat exchanger network synthesis**)
heat exchanger network synthesis
[90C90]
(*see:* **Mixed integer linear programming: heat exchanger network synthesis**)
heat exchanger network synthesis *see:* MINLP: —; Mixed integer linear programming: —
heat exchanger network synthesis without decomposition
[90C90]
(*see:* **MINLP: heat exchanger network synthesis**)
heat exchanger networks
[90C26, 90C90]
(*see:* **Global optimization of heat exchanger networks**)

heat exchanger networks *see:* Global optimization of —; MINLP: mass and —; Mixed integer linear programming: mass and —
heat and mass exchange network
[90C90]
(*see:* **MINLP: reactive distillation column synthesis**)
heat transfer module *see:* mass/ —
heater
[90B35, 90C11, 90C30]
(*see:* **Robust optimization: mixed-integer linear programs**)
heavy ball algorithm
[90C30]
(*see:* **Conjugate-gradient methods**)
heavy ball method
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
Hebden conditions
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
hedging
[90C90, 91B28]
(*see:* **Robust optimization**)
helical proteins *see:* Predictive method for interhelical contacts in alpha- —
helix *see:* α- —
hemicontinuous operator
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
hemicontinuous operator *see:* upper —
hemivariational inequalities
[26B25, 26E25, 35R70, 47S40, 49J35, 49J40, 49J52, 49M05, 49Q10, 49S05, 65K05, 65K99, 70-08, 70-XX, 74A55, 74B99, 74D99, 74G99, 74H99, 74K99, 74M10, 74M15, 74Pxx, 80-XX, 90C26, 90C30, 90C33, 90C90, 90C99, 91A65]
(*see:* **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Multilevel optimization in mechanics**; **Nonconvex energy functions: hemivariational inequalities**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Solving hemivariational inequalities by nonsmooth optimization methods**)
hemivariational inequalities
[49J40, 49J52, 49S05, 65K05, 74G99, 74H99, 74Pxx, 90C30, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**)
hemivariational inequalities *see:* multivalued nonmonotone laws and —; Nonconvex energy functions: —
Hemivariational inequalities: applications in mechanics
(49S05, 74G99, 74H99, 74Pxx, 49J52, 90C33)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
Hemivariational inequalities: eigenvalue problems
(49J52)
(*referred to in:* **αBB algorithm**; **Eigenvalue enclosures for ordinary differential equations**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Interval analysis: eigenvalue bounds of interval matrices**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD**

functions; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Semidefinite programming and determinant maximization**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **αBB algorithm**; **Eigenvalue enclosures for ordinary differential equations**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: static problems**; **Interval analysis: eigenvalue bounds of interval matrices**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Semidefinite programming and determinant maximization**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
hemivariational inequalities for nonlinear material laws *see:* discretized —
hemivariational inequalities by nonsmooth optimization methods *see:* Solving —
Hemivariational inequalities: static problems
(49J40, 47J20, 49J40, 35A15)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
hemivariational inequality *see:* abstract —; semicoercive —; variational- —
hemivariational inequality problem *see:* coercive —
HEN synthesis
[90C90]
(*see:* **MINLP: heat exchanger network synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**)
HEN synthesis using MINLP
[90C90]
(*see:* **MINLP: heat exchanger network synthesis**)
hereditary property
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
Hermitian interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
Hermitian interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
Hermitian matrix *see:* partial —
hesitant adaptive search
[65K05, 90C30]
(*see:* **Random search methods**)
Hessian
[49-04, 65H99, 65K05, 65K99, 65Y05, 68N20, 90C30, 90C31]
(*see:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Sensitivity and stability in NLP**)
Hessian
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**)
Hessian *see:* affine-reduced- —; Automatic differentiation: calculation of the —; limited-memory affine reduced —; lower bounding —; quasi- —; reduced —
Hessian algorithm *see:* affine-reduced- —

Hessian BFGS algorithm *see:* limited-memory reduced- —
Hessian of a Lagrangian *see:* reduced —
Hessian matrices
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
Hessian matrix
[90C25, 90C30, 90C90]
(*see:* **Design optimization in computational fluid dynamics**; **Successive quadratic programming: full space methods**)
Hessian matrix
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**)
Hessian matrix *see:* geodesic —; interval —; n —; projected Lagrangian —
Hessian matrix of the Lagrangian function
[90C25, 90C30, 90C90]
(*see:* **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: full space methods**)
Hessian matrix of a Lagrangian function
[90C30, 90C90]
(*see:* **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**)
Hessian matrix of a Lagrangian function *see:* projected —
Hessian SQP *see:* multiplier-free reduced —
Hessian SQP method *see:* reduced —
Hessian test
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)
Hessians *see:* Complexity of gradients, Jacobians, and —
Hestenes–Stiefel algorithm
[90C30]
(*see:* **Conjugate-gradient methods**)
heterogeneity *see:* subset —; vehicles' homogeneity/ —
heterogeneous
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
heterogeneous azeotropes
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
heterogeneous relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
heterogeneous relations *see:* special properties of —
heteroscedastic model
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
heterotonic operator
[90C33]
(*see:* **Order complementarity**)
Heun method
[90B15]
(*see:* **Dynamic traffic networks**)
heuristic
[68T20, 68T99, 90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Metaheuristics**; **Traveling salesman problem**)
heuristic *see:* aggregation —; annealed replication —; construction —; decomposition —; enhanced —; increasing-degree deletion —; incremental deletion —; maximal matching —; min-exchange —; multiple-hub —; R-opt —; rounding —; savings —; search —; semigreedy —; sequential greedy coloring —; single hub —; Toyoda primal —; Whitney savings —
heuristic algorithm
[90-XX]
(*see:* **Survivable networks**)
heuristic algorithms
[68Q20, 90C90]
(*see:* **Optimal triangulations**; **Simulated annealing methods in protein folding**)
heuristic approach *see:* Bayesian —
heuristic approach to solving CAP on trees
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
heuristic approaches
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
heuristic measure
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
heuristic–metaheuristic algorithms
[90C59]
(*see:* **Heuristic and metaheuristic algorithms for the traveling salesman problem**)
Heuristic and metaheuristic algorithms for the traveling salesman problem
(90C59)
(*referred to in:* **Traveling salesman problem**)
heuristic methods
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
heuristic optimization method
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
heuristic parameter, reject index for interval optimization *see:* Algorithmic improvements using a —
heuristic procedure
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
Heuristic search
(68T20, 90B40, 90C47)
(*referred to in:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: parallel computation**; **Load balancing for parallel optimization techniques**; **Maximum cut problem, MAX-CUT**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Parallel heuristic search**; **Quadratic assignment problem**; **Stochastic network problems: massively parallel solution**)
(*refers to:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: parallel computation**; **Load balancing for parallel optimization techniques**; **Parallel computing: complexity classes**; **Parallel**

computing: models; **Parallel heuristic search**; **Stochastic network problems: massively parallel solution**)
heuristic search *see:* Parallel —
heuristics
[9008, 90B06, 90B35, 90C06, 90C10, 90C11, 90C26, 90C27, 90C30, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Chemical process planning**; **Global optimization based on statistical models**; **Integer programming**; **Set covering, packing and partitioning problems**; **Traveling salesman problem**; **Variable neighborhood search methods**)
heuristics
[05-04, 05C60, 05C69, 05C85, 37B25, 62C10, 65K05, 68W01, 90B35, 90B80, 90C10, 90C11, 90C15, 90C20, 90C26, 90C27, 90C30, 90C35, 90C59, 91A22, 94C15]
(*see:* **Bayesian global optimization**; **Evolutionary algorithms in combinatorial optimization**; **Facilities layout problems**; **Facility location with staircase costs**; **Frank–Wolfe algorithm**; **Graph planarization**; **Heuristics for maximum clique and independent set**; **Job-shop scheduling problem**; **Replicator dynamics in combinatorial optimization**)
heuristics *see:* advanced search —; construction —; continuous based —; greedy —; history-sensitive —; improvement —; journal of —; k-interchange —; local search —; max-regret-fc and max-regret —; primal —; randomized —; sequential —; sequential greedy —
heuristics for axial MITPs *see:* hub —
heuristics of facility location problems with staircase costs
[90B80, 90C11]
(*see:* **Facility location with staircase costs**)
Heuristics for maximum clique and independent set
(90C59, 05C69, 05C85, 68W01)
(*referred to in:* **Graph coloring**; **Greedy randomized adaptive search procedures**; **Replicator dynamics in combinatorial optimization**; **Stable set problem: branch & cut algorithms**)
(*refers to:* **Graph coloring**; **Greedy randomized adaptive search procedures**; **Replicator dynamics in combinatorial optimization**)
Hewitt decomposition *see:* Yosida- —
Hewitt theorem *see:* Yosida– —
Hickey–Cohen triangulation
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
hidden collision
(*see:* **Broadcast scheduling problem**)
hidden constraint
[90C30]
(*see:* **Duality for semidefinite programming**)
hidden Markov model and Gibbs sampler
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
hidden Markov models
(*see:* **Bayesian networks**)
hide-and-seek algorithm
[90C26, 90C90]
(*see:* **Global optimization: hit and run methods**)
hierarchical
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
hierarchical clustering *see:* order constrained —
hierarchical collection of margins
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
hierarchical decision making
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)
hierarchical discrimination
[90C29]
(*see:* **Multicriteria sorting methods**)
hierarchical discrimination *see:* multigroup —
hierarchical optimization
[49-01, 49K10, 49K45, 49M37, 49N10, 90-01, 90B30, 90B50, 90C05, 90C15, 90C20, 90C26, 90C27, 90C30, 90C31, 90C33, 91B32, 91B52, 91B74]
(*see:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Stochastic bilevel programs**)
hierarchical programming problem
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
hierarchy
[90C10, 90C15]
(*see:* **Stochastic integer programs**)
hierarchy *see:* k-level —; lift-and-project —; partition —
hierarchy in a finite set
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
hierarchy process *see:* analytic —
high *see:* sufficiently —
high-dimensional integration
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
high-dimensional integration
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
high failure of the alpha-beta algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
high-level software
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
high-order approximating cone
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order approximating cone *see:* tangent —
high-order approximating cone of decrease
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order approximating cones *see:* feasible —; tangent —
high-order approximating curve
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)

high-order approximating curve *see:* feasible —; tangent —
high-order approximating vector *see:* feasible —
high-order approximating vector of decrease
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order approximating vectors
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order cones of decrease
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order critical direction
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
high-order directional derivatives
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order feasible cones
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order feasible set
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order generalization of Lyusternik theorem
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order local maximum principle
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
high-order local maximum principle for Lagrangian problems
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
high-order local minimum condition
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
High-order maximum principle for abnormal extremals
(49K15, 49K27, 41A10, 47N10)
(*referred to in:* **Dynamic programming: continuous-time optimal control**; **Hamilton–Jacobi–Bellman equation**; **Pontryagin maximum principle**)
(*refers to:* **Dynamic programming: continuous-time optimal control**; **Hamilton–Jacobi–Bellman equation**; **High-order necessary conditions for optimality for abnormal points**; **Pontryagin maximum principle**)
high-order necessary conditions for optimality
[41A10, 46N10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**; **High-order necessary conditions for optimality for abnormal points**)
High-order necessary conditions for optimality for abnormal points
(49K27, 46N10, 41A10, 47N10)
(*referred to in:* **High-order maximum principle for abnormal extremals**; **Kuhn–Tucker optimality conditions**)
(*refers to:* **Kuhn–Tucker optimality conditions**)
high-order set of decrease
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order tangent approximating vector
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order tangent sets
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
high-order tangent sets
[41A10, 46N10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**; **High-order necessary conditions for optimality for abnormal points**)
high performance computing
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
high performance computing
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
high performance computing system
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
high performance Fortran
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
high point problem
[49-01, 49K10, 49M37, 90-01, 90C05, 90C27, 91B52]
(*see:* **Bilevel linear programming**)
high-regular critical direction
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
higher homotopy group
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
higher-order derivatives
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
higher-order directional derivatives
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
higher-order spectrum
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
higher-order statistics
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
higher-order statistics
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
higher order statistics *see:* Signal processing with —

Hilbert basis
[90C10, 90C46]
(*see:* **Integer programming duality**)
Hilbert scheme
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
Hilbert space
[01A99, 90C99]
(*see:* **Von Neumann, John**)
Hilbert space *see:* symmetric element in a —
Hilbert tenth problem
[90C60]
(*see:* **Complexity classes in optimization**)
Hilbert's thirteenth problem
(01A60, 03B30, 54C70, 68Q17)
(*refers to:* **History of optimization**)
hillclimbing procedure *see:* Rosenbrock —
Hipparcos
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
Hirabayashi *see:* problem regular in the sense of Kojima– —
history
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
history
[01A99]
(*see:* **History of optimization**)
history *see:* Structural optimization: —
History of optimization
(01A99)
(*referred to in:* **Carathéodory, Constantine**; **Duality theory: biduality in nonconvex optimization**; **Duality theory: monoduality in convex optimization**; **Duality theory: triduality in global optimization**; **Hilbert's thirteenth problem**; **Inequality-constrained nonlinear optimization**; **Kantorovich, Leonid Vitalyevich**; **Leibniz, gottfried wilhelm**; **Linear programming**; **Operations research**; **Von Neumann, John**)
(*refers to:* **Carathéodory, Constantine**; **Carathéodory theorem**; **Inequality-constrained nonlinear optimization**; **Kantorovich, Leonid Vitalyevich**; **Leibniz, gottfried wilhelm**; **Linear programming**; **Operations research**; **Von Neumann, John**)
history and overview *see:* Bilevel programming: introduction —
history of parametric programming
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
history and rounding error estimation *see:* Automatic differentiation: introduction —
history of a search
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
history-sensitive heuristics
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
history-sensitive heuristics
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
"hit-or-miss" decision problems
[90C15]
(*see:* **Stochastic quasigradient methods: applications**)
hit and run
[65K05, 90C30]
(*see:* **Random search methods**)
hit and run *see:* artificial centering —; hyperspheres direction —; improving —
hit and run algorithm
[90C26, 90C90]
(*see:* **Global optimization: hit and run methods**)
hit and run algorithms
[90C26, 90C90]
(*see:* **Global optimization: hit and run methods**)
hit and run generator
[90C26, 90C90]
(*see:* **Global optimization: hit and run methods**)
hit and run methods
[90C26, 90C29, 90C90]
(*see:* **Global optimization: hit and run methods**; **Optimal design of composite structures**)
hit and run methods *see:* Global optimization: —
hitting cycle problem
[90C35]
(*see:* **Feedback set problems**)
HJB equation
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)
HMM *see:* coupled —; factorial —
hoc networks *see:* Optimization in ad —
Hoffman inequalities *see:* Gale– —
Hölder conditions *see:* uniform —
Hölder continuity
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)
hole-cut *see:* odd- —
home terminals
(*see:* **Railroad crew scheduling**)
homeomorph of a graph
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
homogeneity/heterogeneity *see:* vehicles' —
homogeneous
[90C05, 90C30]
(*see:* **Homogeneous selfdual methods for linear programming**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
homogeneous *see:* increasing and positively —; plus —; positively —
homogeneous algorithm
[90C05]
(*see:* **Homogeneous selfdual methods for linear programming**)
homogeneous beliefs
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)

homogeneous of degree zero
[91B50]
(*see:* **Walrasian price equilibrium**)
homogeneous dual systems
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
homogeneous function *see:* positively —
homogeneous functions on topological vector spaces *see:* Increasing and positively —
homogeneous process *see:* simple —
homogeneous relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
homogeneous relations *see:* special properties of —
Homogeneous selfdual methods for linear programming
(90C05)
(*referred to in:* **Entropy optimization: interior point methods**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Potential reduction methods for linear programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Entropy optimization: interior point methods**; **Interior point methods for semidefinite programming**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Potential reduction methods for linear programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: solution by active sets and interior point methods**)
homogeneous and selfdual model
[90C05]
(*see:* **Homogeneous selfdual methods for linear programming**)
homogeneous systems
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
homogeneous systems of linear relations *see:* Tucker —
homogenous
(*see:* **Approximations to robust conic optimization problems**)
homogenous cones
(*see:* **Approximations to robust conic optimization problems**)
homomorphism
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
homomorphism *see:* strong —; very strong —; weak —
homomorphisms
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
homoscedastic model
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
homotopic functions
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
homotopic maps
[01A50, 01A55, 01A60]
(*see:* **Fundamental theorem of algebra**)
homotopic method
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
homotopies *see:* optimization —; probability-one globally convergent —
homotopy
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
homotopy
[01A50, 01A55, 01A60, 65F10, 65F50, 65H10, 65K10]
(*see:* **Fundamental theorem of algebra**; **Globally convergent homotopy methods**)
homotopy *see:* adaptive —; probability-one —
homotopy algorithm *see:* globally convergent probability-one —
homotopy continuation
[65C20, 65G20, 65G30, 65G40, 65H20, 90C30, 90C90]
(*see:* **Interval analysis: application to chemical engineering design problems**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
homotopy continuation method
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
homotopy group *see:* higher —
homotopy method
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
homotopy method
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
homotopy methods
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
homotopy methods *see:* Globally convergent —; software for —
homotopy Newton method
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
homotopy property
[90C33]
(*see:* **Topological methods in complementarity theory**)
homotopy type
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
homotopy type
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)

HOMPACK90
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
Hook law
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)
hop neighboring stations *see:* one- —
hop neighbors *see:* two- —
Hopcroft–Tarjan planarity-testing algorithm
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
Hopf equations *see:* Wiener– —
hopping problem *see:* airplane —
horizon *see:* decision making with rolling —; finite —; infinite —; infinite time —; planning —
horizon control and dynamic games *see:* Infinite —
horizon game *see:* nonzero-sum infinite —
horizon problem *see:* discounted infinite —; total cost infinite —
horizon problems *see:* infinite —
horizon problems, overview *see:* Dynamic programming: infinite —
horizontal linear complementarity problem
[90C33]
(*see:* **Linear complementarity problem**)
Horn formulas
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
Horn formulas
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
HOS
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
hot spot
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
hot spots
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
Householder transformation
[15A23, 49M37, 65F05, 65F20, 65F22, 65F25]
(*see:* **Nonlinear least squares: Newton-type methods**; **QR factorization**)
Householder transformations *see:* QR factorization using —
HPF
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
hPS
(*see:* **Short-term scheduling of batch processes with resources**)
Huang algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**)
Huang algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
Huang algorithm *see:* modified —
Huang method
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
hub
[90C35]
(*see:* **Multi-index transportation problems**)
hub heuristic *see:* multiple- —; single —
hub heuristics for axial MITPs
[90C35]
(*see:* **Multi-index transportation problems**)
Huber M-estimator
[65D10, 65K05]
(*see:* **Overdetermined systems of linear equations**)
hull *see:* convex —; lower convex —; normal —; reverse normal —
hull consistency
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
hull disjunctions *see:* Convex —
hull problem *see:* convex —
Hull relaxation
(*see:* **Optimal planning of offshore oilfield infrastructure**)
human rationality assumption
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
human rationality factor
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
Hungarian algorithm
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
Hungarian method
[90C10, 90C35]
(*see:* **Bi-objective assignment problem**)
Hunter–Worsley bounds
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
Hunter–Worsley upper bound
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
Hurwicz gradient method *see:* Arrow– —
huS
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)
HVI
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)
Hwang conjecture *see:* Graham– —
Hwang minimax theorem *see:* Du– —
hybrid algorithm
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
hybrid algorithm
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms**)

hybrid branch and bound and outer approximation
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
hybrid conjugate gradient algorithms for unconstrained optimization *see:* New —
hybrid metaheuristic
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
hybrid metaheuristics *see:* GRASP in —
hybrid methods
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
hybrid methods *see:* Mixed integer programming/constraint programming —
hybrid model
[90C15]
(*see:* **Static stochastic programming models**)
hybrid NP methods
[90C11, 90C59]
(*see:* **Nested partitions optimization**)
hybrid orthonormalization
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
hydro-generation
[90C35]
(*see:* **Multicommodity flow problems**)
hydro plants
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
hydro-reservoir
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
hydrological exogenous inflow
[90C30, 90C35]
(*see:* **Optimization in water resources**)
hydrological exogenous inflow and demand *see:* water resources planning under uncertainty on —
hydropower nodes *see:* on-the-river —
hyperbolic 0-1 programming problem
(*see:* **Fractional zero-one programming**)
(hyperbolic) 0-1 programming problem *see:* single-ratio fractional —
hyperbolic programming
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
hypercube
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
hypercube
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
hypercube *see:* d-dimensional —
hyperdifferentiable
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
hyperdifferentiable function
[49J52, 65K99, 70-08, 90C25]
(*see:* **Quasidifferentiable optimization: codifferentiable functions**)
hyperdifferential
[49J52, 65K99, 65Kxx, 70-08, 90C25, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: codifferentiable functions**)
hyperdifferential *see:* second order —
hypergeometric distribution
[90C15]
(*see:* **Logconcavity of discrete distributions**)
hypergeometric integral
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
hypergeometric integral
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
hyperglycemia
(*see:* **Model based control for drug delivery systems**)
hypergraph *see:* subtree —
hypergraph q-coloring
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
hyperplane *see:* arrangement of —; separating —; support —; tangent —
hyperplane arrangement
[90C09, 90C10]
(*see:* **Oriented matroids**)
hyperplane arrangement
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**; **Hyperplane arrangements in optimization**)
Hyperplane arrangements
(52C35, 05B35, 57N65, 20F36, 20F55)
(*referred to in:* **Hyperplane arrangements in optimization**)
(*refers to:* **Hyperplane arrangements in optimization**)
Hyperplane arrangements in optimization
(05B35, 20F36, 20F55, 52C35, 57N65)
(*referred to in:* **Hyperplane arrangements**)
(*refers to:* **Hyperplane arrangements**)
hyperplanes *see:* Boolean arrangement of —; braid arrangement of —; cohomology of an arrangement of —; complement of an arrangement of —; dependent —; divisor of an arrangement of —; free arrangement of —; general position of —; reflection arrangement of —; singularity of an arrangement of —
hyperspheres direction hit and run
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
hyperspherical direction
[90C26, 90C90]
(*see:* **Global optimization: hit and run methods**)
hypervoxels
[90C90]
(*see:* **Optimization in medical imaging**)
hypodifferentiability
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)

hypodifferentiability
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
hypodifferentiable
[49J52, 65K99, 65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**)
hypodifferentiable function
[49J52, 65K99, 70-08, 90C25]
(*see:* **Quasidifferentiable optimization: codifferentiable functions**)
hypodifferentiable functions *see:* Quasidifferentiable optimization: algorithms for —
hypodifferentiable optimization
[49J52, 65K99]
(*see:* **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**)
hypodifferential
[49J52, 65K05, 65K99, 65Kxx, 70-08, 90C25, 90C30, 90Cxx]
(*see:* **Nondifferentiable optimization: minimax problems**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: codifferentiable functions**)
hypodifferential
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
hypodifferential *see:* kth order —; second —; second order —
hypodifferential descent
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
hypodifferential descent *see:* method of —
hypoglycemia
(*see:* **Model based control for drug delivery systems**)
hypothesis *see:* financial leverage —; inefficient management —
hypothesis formation *see:* automated —
hysteresis
[49J52]
(*see:* **Hemivariational inequalities: eigenvalue problems**)
hZaw
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)
hZw
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)

I

I *see:* algorithm partition-matching- —; generalization of ELECTRE —
I requirement *see:* Type —
IA
[90C26]
(*see:* **Cutting plane methods for global optimization**)
IBC
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
IDA∗
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
ideal *see:* admissible pair of a monomial —; arithmetic degree of a monomial —; initial —; lattice —; localization of an —; monomial —; polynomial —; standard pair decomposition of a monomial —; standard pair of a monomial —; Stanley–Reisner —; toric —
ideal and nonideal phase equilibrium equations
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in distillation systems**)
ideal part
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in distillation systems**)
identical machines
[68Q99]
(*see:* **Branch and price: Integer programming with column generation**)
identification
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
identification *see:* Mixed 0-1 linear programming approach for DNA transcription element —; model —; parameter —
Identification methods for reaction kinetics and transport
(34A55, 35R30, 62G05, 62G08, 62P30, 62P10, 62J02, 62K05, 76R50, 80A23, 80A30, 80A20)
identification problem *see:* parameter —
identification via mixed-integer optimization *see:* Peptide —
identities *see:* primitive partition —
identity transformation
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
idle time
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
Idnani active set strategy *see:* Goldfarb– —
Idnani method *see:* Goldfarb– —
IDP
[90C30]
(*see:* **Suboptimal control**)
IDP
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
IEQNO
[49M37, 65K05, 90C30]
(*see:* **Inequality-constrained nonlinear optimization**)
if-when scenarios *see:* what- —
IFS
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
IHDG
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)

Il rule *see:* polyak —
IIP
[90C05]
(*see:* **Linear programming: interior point methods**)
IL
[90C30, 90C33]
(*see:* **Implicit lagrangian**)
ill-conditioned coefficient matrix
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
ill-conditioned matrix
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
ill-conditioned problem
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
ill-posed
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
ill-posed problem
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
ill-posed problems
[90C30]
(*see:* **Cost approximation algorithms**)
ill-posed variational problem
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
Ill-posed variational problems
(65K05, 65M30, 65M32, 49M30, 49J40)
(*referred to in:* **Sensitivity and stability in NLP**)
(*refers to:* **Sensitivity and stability in NLP**)
IM *see:* multistage —; single-stage —
IM in SC
[90B50]
(*see:* **Inventory management in supply chains**)
iMA Journal of Management Mathematics
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
image
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
image problem
[90C30]
(*see:* **Image space approach to optimization**)
image processing *see:* optimization in medical —
image reconstruction
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
image reconstruction *see:* Entropy optimization for —; finite-dimensional models for entropy optimization for —; Maximum entropy principle: —; vector-space models for entropy optimization for —
image reconstruction from projection data
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
image reconstruction from projection data *see:* feasibility approach to —; optimization approach to —
image space
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
image space
[90C05, 90C30]
(*see:* **Image space approach to optimization**; **Theorems of the alternative and optimization**)
Image space approach to optimization
(90C30)
(*referred to in:* **Theorems of the alternative and optimization**; **Vector optimization**)
(*refers to:* **Theorems of the alternative and optimization**; **Vector optimization**)
images
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
imaging *see:* medical —; Optimization in medical —
imbalance
[90C35]
(*see:* **Minimum cost flow problem**)
immediate selection
[90B35]
(*see:* **Job-shop scheduling problem**)
imperative programming paradigm
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
imperfect competition
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
imperfect competition
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
implementation *see:* programmed —; PVM-based —
implementation of the auction algorithm *see:* synchronous —; totally asynchronous —
implementation example *see:* optimization computer —
implementation of optimization *see:* Computer —
implementations *see:* MPI-based —
implication
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
implication *see:* Goguen–Gaines —; Kleene–Dienes —; Łukasiewicz —; many-valued logic —; Reichenbach —
implication operator
[03B50, 03B52, 03C80, 03E72, 47S40, 62F30, 62Gxx, 68T15, 68T27, 68T30, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**)
implications *see:* logical —
implicit Choleski algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
implicit complementarity problem
[90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Topological methods in complementarity theory**)

implicit enumeration
[90C35]
(*see:* **Graph coloring**)
implicit equality constraint
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
implicit function approach
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
implicit function approach *see:* Bilevel programming: —
implicit function approach to bilevel programming
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
implicit function theorem
[90C30, 90C31]
(*see:* **Bounds and solution vector estimates for parametric NLPS**; **Generalized total least squares**)
implicit general order complementarity problem
[90C33]
(*see:* **Order complementarity**)
Implicit lagrangian
(90C33, 90C30)
(*referred to in:* **Kuhn–Tucker optimality conditions**)
(*refers to:* **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Variational inequalities**)
implicit Lagrangian *see:* restricted —; unconstrained —
implicit LU algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**)
Implicit LU algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
implicit LX algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**)
Implicit LX algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
implicit QR algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
implicit restarted Lanczos method
[90C30]
(*see:* **Large scale trust region problems**)
implicit utility function
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
implicit variational inequalities and quasivariational inequalities
[49J40, 49Q10, 70-08, 74K99, 74Pxx]
(*see:* **Quasivariational inequalities**)
implicit variational problems
[49J40, 49Q10, 70-08, 74K99, 74Pxx]
(*see:* **Quasivariational inequalities**)
implied constraint
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**)
implied inequality
[15A39, 90C05]
(*see:* **Linear optimization: theorems of the alternative**)
import model
[90C35]
(*see:* **Multicommodity flow problems**)
importance sampling
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
impossible pairs constrained path problem
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
imprecise information
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
improper
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
improper vertex
[90C26]
(*see:* **Cutting plane methods for global optimization**)
improved piecewise linearization
[90B80, 90C11]
(*see:* **Facility location with staircase costs**)
improved procedure
[90B80]
(*see:* **Facilities layout problems**)
improvement *see:* best —; dual bound- —; dual cut- —; local —; primal bound- —; primal cut- —; rule of greatest —
improvement heuristics
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
improvement of KKT points *see:* successive —
improvement methods
[90B06, 90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**; **Vehicle routing**)
improvements using a heuristic parameter, reject index for interval optimization *see:* Algorithmic —
improving
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
improving feasible direction
[90C30]
(*see:* **Convex-simplex algorithm**)
improving feasible direction
[90C30]
(*see:* **Convex-simplex algorithm**)
improving hit and run
[65K05, 90C26, 90C29, 90C30, 90C90]
(*see:* **Global optimization: hit and run methods**; **Optimal design of composite structures**; **Random search methods**)
improving hit and run
[90C26, 90C29, 90C90]
(*see:* **Global optimization: hit and run methods**; **Optimal design of composite structures**)
impulse perturbations *see:* Vasicek model with —
imputation
[90C27, 90C60, 91A12]
(*see:* **Combinatorial optimization games**)

IMSL subroutine library
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
imvex
[90C26]
(*see:* **Invexity and its applications**)
in-company policies
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
(in logic) *see:* literal —
inaccuracy
[90C05, 90C25]
(*see:* **Young programming**)
inaccuracy in observations
[65D10, 65K05]
(*see:* **Overdetermined systems of linear equations**)
inactive
[90C60]
(*see:* **Complexity of degeneracy**)
inactive constraints
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
inactive constraints
[90C60]
(*see:* **Complexity of degeneracy**)
inadmissible arc
[90C35]
(*see:* **Maximum flow problem**)
incidence
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements in optimization**)
incidence graph *see:* column —
incidence matrix
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
incidence matrix *see:* node-arc —
incidence in a network
[90C35]
(*see:* **Minimum cost flow problem**)
incidence vector
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
incident
[90C35]
(*see:* **Graph coloring**)
incident faces
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements in optimization**)
inclusion *see:* degree of —; variational —
inclusion function
[65G20, 65G30, 65G40, 65K05, 65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**; **Interval global optimization**)
inclusion function *see:* good —; isotone —; order of an —
Inclusion Method
[49R50, 65G20, 65G30, 65G40, 65L15, 65L60]
(*see:* **Eigenvalue enclosures for ordinary differential equations**)
inclusion operator *see:* fuzzy set- —
inclusion principle
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
inclusion principle of machine interval arithmetic
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
inclusion of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
incoming arc
[90C35]
(*see:* **Minimum cost flow problem**)
incomparability
[90C29]
(*see:* **Preference modeling**)
incomplete information
[68Q25, 90C09, 90C10, 91B28]
(*see:* **Competitive ratio for portfolio management**; **Optimization in boolean classification problems**)
incomplete information
[62C20, 90C15]
(*see:* **Stochastic programming: minimax approach**)
incomplete judgments
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
incomplete knowledge of a probability distribution
[62C20, 90C15]
(*see:* **Stochastic programming: minimax approach**)
incomplete methods
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
incomplete solution
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
incomplete state feedback
[90C30]
(*see:* **Suboptimal control**)
incorporation of biological constraints
(*see:* **Selection of maximally informative genes**)
increase *see:* dual price —
increasing
[65K05, 90B10, 90C26, 90C30]
(*see:* **Monotonic optimization**; **Piecewise linear network flow problems**)
increasing *see:* concave —; linear —
Increasing and convex-along-rays functions on topological vector spaces
(26A48, 52A07, 26A51)
increasing-degree deletion heuristic
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
increasing function
[90C26]
(*see:* **Cutting plane methods for global optimization**)
increasing function *see:* coordinatewise —
increasing and positively homogeneous
[26A48, 26A51, 52A07]

(*see:* **Increasing and convex-along-rays functions on topological vector spaces**)
Increasing and positively homogeneous functions on topological vector spaces
(26A48, 52A07, 26A51)
increasing utility function *see:* coordinatewise —
increment *see:* bidding —
incremental algorithm
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
incremental deletion heuristic
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
incremental gradient method
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
incremental-iterative solution algorithm
[49J52, 49Q10, 74G60, 74H99, 74K99, 74Pxx, 90C90]
(*see:* **Quasidifferentiable optimization: stability of dynamic systems**)
incremental negamax algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
incremental strategy for model structure refinement
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
incumbent objective value
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
incumbent solution
[90C06, 90C15, 90C25]
(*see:* **Concave programming**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**)
incumbent value
[90C10, 90C29]
(*see:* **Multi-objective integer linear programming**)
indecomposable matrix *see:* fully —
indefinite
[90C60]
(*see:* **Complexity theory: quadratic programming**)
indefinite integral
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
indefinite quadratic problems
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
indefinite quadratic programming
[90C11, 90C25]
(*see:* **Concave programming**; **MINLP: branch and bound methods**)
indefinite quadratic programs
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
independence
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
independence *see:* global —; linear —; local —
independence constraint qualification *see:* linear —
independence constraint qualification (LICQ) *see:* linear —
independence CQ *see:* linear —
independence number
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
independence system
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
independency constraint qualification *see:* linear —
independent
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
independent *see:* linearly —; model —; positively linearly —
independent dominating set
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
independent set
[05C15, 05C17, 05C35, 05C62, 05C69, 05C85, 68Q25, 68R10, 68W01, 68W40, 90C09, 90C10, 90C22, 90C27, 90C35, 90C59]
(*see:* **Domination analysis in combinatorial optimization**; **Graph coloring**; **Heuristics for maximum clique and independent set**; **Lovász number**; **Matroids**; **Optimization problems in unit-disk graphs**)
independent set
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
Independent Set *see:* Heuristics for maximum clique and —; maximal —; maximum —; maximum weighted —
Independent Set Problem *see:* maximum —
independent sets
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
independent sets *see:* maximum weight —
independent of solver formats
(*see:* **Planning in the process industry**)
independent subset
[90C09, 90C10]
(*see:* **Matroids**)
independent system
[90C09, 90C10]
(*see:* **Matroids**)
independent variables
[65D25, 65G20, 65G30, 65G40, 65H20, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**; **Interval analysis: intermediate terms**)
indeterminate box
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
index *see:* active —; augmented performance —; Bilevel optimization: feasibility test and flexibility —; blending —; consistency —; cost —; descending —; flexibility —; gradient —; least- —; linear —; linear co- —; merit —; morse —; performance —; quadratic —; quadratic co- —
index anticycling rules *see:* Least- —

index approach
[90C26]
(*see:* **Global optimization using space filling**)
index approach
[90C26]
(*see:* **Global optimization using space filling**)
index assignment problems *see:* multi- —; three- —
index-based
(*see:* **Planning in the process industry**)
index of a constraint violating point
[90C26]
(*see:* **Global optimization using space filling**)
index criss-cross method *see:* least- —
index for interval optimization *see:* Algorithmic improvements using a heuristic parameter, reject —
index market model *see:* Sharpe single —
index pivoting method *see:* least- —
index pivoting rule *see:* Bland least —
index refinement *see:* Murty least- —
index rule *see:* smallest —
index set
[90C34]
(*see:* **Semi-infinite programming: approximation methods**)
index set *see:* active —; essentially active —; SIP —; species —
index transportation problem *see:* axial multi- —; integer multi- —; k- —; multi- —; planar multi- —; symmetric multi- —; three- —
index transportation problems *see:* Multi- —
index tree
[34E05, 90C27]
(*see:* **Asymptotic properties of random multidimensional assignment problem**)
indexing terms
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
indexing terms
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
indexing vocabulary
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
indexing vocabulary
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
indexing vocabulary *see:* optimal —
iNDF
[65K10, 90C33, 90C51]
(*see:* **Generalizations of interior point methods for the linear complementarity problem**)
indicator
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
indicator function
[49J40, 49J52, 49Q10, 49S05, 62F12, 65C05, 65K05, 70-08, 70-XX, 74G99, 74H99, 74K99, 74Pxx, 80-XX, 90C15, 90C31, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**; **Monte-Carlo simulations for stochastic optimization**; **Nonconvex energy functions: hemivariational inequalities**; **Quasivariational inequalities**; **Stochastic quasigradient methods: applications**)
indicator function *see:* expectation of an —
indices
(*see:* **Planning in the process industry**; **Short-term scheduling under uncertainty: sensitivity analysis**)
indices *see:* morse —
indifference
[90C29]
(*see:* **Preference modeling**)
indifference threshold
[90-XX]
(*see:* **Outranking methods**)
indirect methods
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
indiscernibility
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
indiscernibility *see:* fundamental —; relation of —
individual
[92B05]
(*see:* **Genetic algorithms**)
individual
[92B05]
(*see:* **Genetic algorithms**)
individual probabilistic constraints
[90C15]
(*see:* **Static stochastic programming models**)
individual rationality
[90C27, 90C60, 91A12]
(*see:* **Combinatorial optimization games**)
individual software routine
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
individuals
(*see:* **Broadcast scheduling problem**)
induced region
[49M37, 90C26, 91A10]
(*see:* **Bilevel programming**)
induced subgraph
[05C50, 05C60, 05C69, 15A48, 15A57, 37B25, 90C20, 90C25, 90C27, 90C35, 90C59, 91A22]
(*see:* **Matrix completion problems**; **Replicator dynamics in combinatorial optimization**)
induced by a vertex subset *see:* subgraph —
inducible region
[49-01, 49K10, 49M37, 90-01, 90C05, 90C27, 90C30, 90C90, 91B52]
(*see:* **Bilevel linear programming**; **Bilevel programming: global optimization**)
induction axiom
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
inductive inference
[90C26, 90C30]
(*see:* **Forecasting**)
inductive inference
[90C26, 90C30]
(*see:* **Forecasting**)

inductive inference problem
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
inductive inference problem
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
inductive structure of an irreducible matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
industrial engineering *see:* Archimedes and the foundations of —
industrial in-plant railroads *see:* engine routing and —
industrial problems *see:* SQP optimization in —
industry *see:* GRASP in —; petrochemical —; Planning in the process —; refining —; Successive quadratic programming: applications in the process —
inefficient
[90B30, 90B50, 90C05, 91B82]
(*see:* **Data envelopment analysis**)
inefficient management hypothesis
[90C05, 90C90, 91B28]
(*see:* **Multicriteria methods for mergers and acquisitions**)
inequalities *see:* anti-Monge —; approximation of variational —; blossom —; convex —; Gale–Hoffman —; hemivariational —; implicit variational inequalities and quasivariational —; linear —; Monge —; multivalued monotone laws and variational —; multivalued nonmonotone laws and hemivariational —; Nonconvex energy functions: hemivariational —; Nonsmooth and smoothing methods for nonlinear complementarity problems and variational —; parametric variational —; QD laws and systems of variational —; Quasivariational —; saddle-point —; scalar variational —; Solution methods for multivalued variational —; system of —; system of variational —; valid —; variational —; variational-like —; vector variational —
inequalities: applications in mechanics *see:* Hemivariational —
inequalities: A brief review *see:* Generalized variational —
inequalities: eigenvalue problems *see:* Hemivariational —
inequalities and equilibrium problems *see:* Generalized monotonicity: applications to variational —
inequalities: F. E. approach *see:* Variational —
inequalities: geometric interpretation, existence and uniqueness *see:* Variational —
inequalities for nonlinear material laws *see:* discretized hemivariational —
inequalities by nonsmooth optimization methods *see:* Solving hemivariational —
inequalities: projected dynamical system *see:* Variational —
inequalities and quasivariational inequalities *see:* implicit variational —
inequalities: static problems *see:* Hemivariational —
inequality *see:* abstract hemivariational —; azuma's —; bilinear matrix —; Cauchy —; disjunctive —; duality —; Fenchel–Young —; generalized variational —; Hamilton–Jacobi —; implied —; jensen's —; linear matrix —; mixed variational —; nondominated valid —; quasivariational —; reverse convex —; semicoercive hemivariational —; strengthen triangle —; subgradient —; triangle —; two-function minimax —; variational —; variational-hemivariational —; vector —; Young —
Inequality-constrained nonlinear optimization
(49M37, 65K05, 90C30)
(*referred to in:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **History of optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Redundancy in nonlinear programs**; **Relaxation in projection methods**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**; **SSC minimization algorithms**; **SSC minimization algorithms for nonsmooth and stochastic optimization**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **History of optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Redundancy in nonlinear programs**; **Relaxation in projection methods**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**; **SSC minimization algorithms**; **SSC minimization algorithms for nonsmooth and stochastic optimization**)
inequality constraint *see:* state —
inequality constraints
[41A10, 46N10, 47N10, 49K27, 65K05]
(*see:* **Direct global optimization algorithm**; **High-order necessary conditions for optimality for abnormal points**)
inequality constraints *see:* active —; feasibility of —; infeasibility of —
inequality for an elastostatic problem involving QD-superpotentials *see:* convex variational —
inequality of elliptic type *see:* abstract variational —
inequality formulation *see:* variational —
inequality formulation in link loads *see:* variational —
inequality formulation in path flows *see:* variational —
inequality formulations *see:* variational —
inequality or nonsmooth mechanics
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
inequality problem *see:* coercive hemivariational —; dual variational —; finite-dimensional variational —; parametric variational —; variational —; vector variational —
inequality problem and a projected dynamical system *see:* variational —
inequality problems
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
inequality problems *see:* Sensitivity analysis of variational —; variational —
inequality relation *see:* linear —
inequality systems
[15A39, 46A20, 52A01, 90C05, 90C30]
(*see:* **Farkas lemma**; **Farkas lemma: generalizations**; **Linear optimization: theorems of the alternative**; **Motzkin transposition theorem**; **Tucker homogeneous systems of linear relations**)
inequality systems *see:* convex —

inertia of a matrix
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
inexact line search
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
inexact line search line
[90C30]
(*see:* **Frank–Wolfe algorithm**)
inexact line search technique
[90C30]
(*see:* **Convex-simplex algorithm**)
inexact Newton method
[90C30]
(*see:* **Numerical methods for unary optimization**)
inexact Newton method
[90C30]
(*see:* **Numerical methods for unary optimization**)
inexact Newton methods
[90C06]
(*see:* **Large scale unconstrained optimization**)
inexact proximal point algorithms
[90C30]
(*see:* **Cost approximation algorithms**)
inf-stationary
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
inf-stationary point
[65K05, 90C30, 90Cxx]
(*see:* **Nondifferentiable optimization: minimax problems**; **Quasidifferentiable optimization: optimality conditions**)
inf-stationary point
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
inf-stationary points
[49J52, 65K99]
(*see:* **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**)
infeasibilities *see:* diagnosing and tracing —; sum of integer —
infeasibility criterion
[90C11, 90C31]
(*see:* **Multiparametric mixed integer linear programming**)
infeasibility of inequality constraints
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: verifying feasibility**)
infeasibility proof
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
infeasibility test
[49M37, 65G20, 65G30, 65G40, 65K05, 90C11, 90C30]
(*see:* **Interval global optimization**; **Mixed integer nonlinear programming**)
infeasible
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
infeasible component
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
infeasible integer variable *see:* most/least —
infeasible interior point
[90C05]
(*see:* **Linear programming: interior point methods**)
infeasible node
[90C10, 90C29]
(*see:* **Multi-objective integer linear programming**)
infeasible path approach
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
infeasible problem
[15A39, 90C05]
(*see:* **Linear optimization: theorems of the alternative**)
infeasible program
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
infeasible solution
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
infeasible-start interior-point algorithm
[90C05, 90C22, 90C25, 90C30, 90C51]
(*see:* **Interior point methods for semidefinite programming**)
infeasible system
[15A39, 90C05]
(*see:* **Motzkin transposition theorem**)
Infer *see:* branch and —
inference *see:* Bayesian —; classical —; fuzzy interval —; inductive —; interval-valued approximate —; monotone Boolean function —; order restricted statistical —; premis of an —; visual —
inference duality
[90C10, 90C46]
(*see:* **Integer programming duality**)
Inference of monotone boolean functions
(90C09)
(*referred to in:* **Alternative set theory**; **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
(*refers to:* **Alternative set theory**; **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
inference problem *see:* Boolean function —; inductive —
infimum
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
infimum *see:* global —; local —
infimum of a Lagrangian function
[90C30]
(*see:* **Lagrangian duality: BASICS**)
infinite *see:* semi- —
infinite class
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)

infinite-dimensional generalized order complementarity problem
[90C33]
(*see:* **Generalized nonlinear complementarity problem**)
infinite-dimensional linear programming
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
infinite-dimensional optimization
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
infinite horizon
(*see:* **Bayesian networks**)
Infinite horizon control and dynamic games
(91Axx, 49Jxx)
(*referred to in:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
infinite horizon game *see:* nonzero-sum —
infinite horizon problem *see:* discounted —; total cost —
infinite horizon problems
[49L20, 90C39, 90C40]
(*see:* **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming: undiscounted problems**)
infinite horizon problems
[49L20, 49L99, 90C39, 90C40]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**)
infinite horizon problems, overview *see:* Dynamic programming: —
infinite linear programming *see:* semi- —
infinite many conditions moment problem
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
infinite moment problem
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
infinite optimization *see:* Adaptive convexification in semi- —; one-parametric semi- —; semi- —; Smoothing methods for semi- —
infinite optimization problem *see:* semi- —
infinite optimization problems *see:* semi- —
infinite problem *see:* generalized semi- —
infinite program *see:* dual semi- —; primal (linear) semi- —; semi- —
infinite programming *see:* linear semi- —; perfect duality from the view of linear semi- —; reduced problem in semi- —; semi- —
infinite programming and applications in finance *see:* Semi- —
infinite programming: approximation methods *see:* Semi- —
infinite programming and control problems *see:* Semi- —
infinite programming: discretization methods *see:* Semi- —
infinite programming: methods for linear problems *see:* Semi- —
infinite programming: numerical methods *see:* Semi- —
infinite programming: optimality conditions *see:* Generalized semi- —
infinite programming: second order optimality conditions *see:* Semi- —
infinite programming, semidefinite programming and perfect duality *see:* Semi- —
infinite programs *see:* computationally equivalent semi- —; nonlinear semi- —; semi- —
infinite time horizon
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
infinitely connected matroid
[90C09, 90C10]
(*see:* **Matroids**)
infinitely near rational numbers
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
infinitely small negative real numbers
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
infinitely small positive real numbers
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
infinitely small real numbers
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
infinitesimal
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
infinitesimal calculus
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
infinitesimal calculus
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)

infinitesimal perturbation analysis
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
infinity
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
infinity
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
inflow *see:* hydrological exogenous —
inflow and demand *see:* water resources planning under uncertainty on hydrological exogenous —
information
[01A60, 03B30, 54C70, 65K05, 68Q05, 68Q10, 68Q17, 68Q25, 90C05, 90C25, 90C26, 94A17]
(*see:* **Hilbert's thirteenth problem**; **Information-based complexity and information-based optimization**; **Jaynes' maximum entropy principle**)
information
[01A60, 03B30, 54C70, 68Q17, 90C60]
(*see:* **Hilbert's thirteenth problem**; **Kolmogorov complexity**)
information *see:* Algorithmic —; asymmetrical —; contaminated —; dual —; expected value of perfect —; imprecise —; incomplete —; missing —; mutual —; partial —; priced —; radius of —; uncertain —; unknown —
information-based complexity
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
information-based complexity
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26, 90C60]
(*see:* **Complexity theory**; **Information-based complexity and information-based optimization**)
Information-based complexity and information-based optimization
(65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26)
(*referred to in:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**)
(*refers to:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**)
information-based model
[90C60]
(*see:* **Complexity theory**)
information-based optimization
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
information-based optimization
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
information-based optimization *see:* Information-based complexity and —
information criterion *see:* Akaike —
information game *see:* two-player zero-sum perfect- —
information structure
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
informative genes *see:* Selection of maximally —
informed *see:* weakly —
infrastructure *see:* Optimal planning of offshore oilfield —
ing *see:* AND- —; OR- —
Ingber algorithm
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
inhibit procedure
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
inhibitor
(*see:* **Bayesian networks**)
initial annealing temperature
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
initial ideal
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
initial simplex
[90C30]
(*see:* **Sequential simplex method**)
initial solution
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
initial system
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
initial tableau
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
initial temperature
[90C27, 90C90]
(*see:* **Simulated annealing**)
initial term of a polynomial
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
initial value problem
[65G20, 65G30, 65G40, 65K10, 65L99, 90C90]
(*see:* **Interval analysis: differential equations**; **Variational inequalities: projected dynamical system**)
initial value problem
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)
initialization
[49L20, 90C05, 90C33, 90C39, 90C40, 92C05, 92C40]
(*see:* **Dynamic programming: infinite horizon problems, overview**; **Pivoting algorithms for linear programming generating two paths**; **Protein loop structure prediction methods**)
initializing unknown variables
[90C25, 90C30]

(*see:* **Successive quadratic programming: full space methods**)
initiated *see:* receiver- —; sender- —
initiated mapping technique *see:* receiver —; sender —
inner approximation
[90C06, 90C25, 90C26, 90C35]
(*see:* **Concave programming**; **Cutting plane methods for global optimization**; **Simplicial decomposition algorithms**)
inner approximation
[90C06, 90C25, 90C26, 90C35]
(*see:* **Cutting plane methods for global optimization**; **Simplicial decomposition algorithms**)
inner interval arithmetic
[65G30, 65G40, 65K05, 90C30, 90C57]
(*see:* **Global optimization: interval analysis and balanced interval arithmetic**)
inner linearization
[90C30]
(*see:* **Simplicial decomposition**)
inner linearization cone
[90C31, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions)
inner linearization/restriction
[90C30]
(*see:* **Simplicial decomposition**)
inner point
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
inner problem
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
inner product *see:* **K**-local —
inner regular measure
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
input *see:* interva —; maximization of output/ —
input alphabet of a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
input constraint qualifications
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
input data *see:* length of —; size of —
input-efficient
[90B30, 90B50, 90C05, 91B82]
(*see:* **Data envelopment analysis**)
input layer
(*see:* **Bayesian networks**)
input neurons
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
input optimization
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**; **Nondifferentiable optimization: parametric programming**)
input-output matrices *see:* updating —
input-output tables *see:* triangulation problem for —
input of a Turing machine *see:* size of the —
inscribed sphere method *see:* largest —
insertion
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
insertion algorithm *see:* generic vertex —
insertion (FVI) *see:* farthest vertex —
insertion (NVI) *see:* nearest vertex —
insertion optimal partitioning algorithm *see:* nearest —
insertion paradigm *see:* edge —
insertion (RVI) *see:* random vertex —
insertion step
[90C59]
(*see:* **Heuristic and metaheuristic algorithms for the traveling salesman problem**)
insertion supernode
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
insertion of vertex v at
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
insertion (VI) *see:* vertex —
insight
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
instability in parametric programming
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
instance
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
instance *see:* problem —; size of a problem —
instance in time m *see:* algorithm solving a problem —
instances *see:* all-to-all —; one-to-all —
instrument financial equilibrium model *see:* multi-sector multi- —
insufficient reason *see:* laplace's principle of —
insufficient reasoning *see:* Laplace principle of —
int *U-quasiconcave function*
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
integer
[65K05, 90C10, 90C11, 90C26, 90C46]
(*see:* **Direct global optimization algorithm**; **Integer programming duality**; **MINLP: global optimization with αBB**)
integer
[90C10, 90C29]
(*see:* **Multi-objective integer linear programming**)
integer *see:* mixed —
integer 0–1 programs *see:* mixed —
integer αBB algorithm *see:* general structure mixed —; special structure mixed —
Integer Bilevel Optimization *see:* Mixed —
integer classification problems *see:* Mixed —

integer cut
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
integer dynamic optimization *see:* mixed —
integer feasibility problem *see:* zero-one —
integer-fix *see:* near- —
integer formulation *see:* LCP: Pardalos–Rosen mixed —
integer fractional programming
[90C32]
(*see:* **Fractional programming**)
integer infeasibilities *see:* sum of —
integer L-shaped method
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
integer labeling
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
integer LCP
[90C25, 90C33]
(*see:* **Integer linear complementary problem**)
Integer linear complementary problem
(90C25, 90C33)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Graph coloring**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Linear complementarity problem**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Order complementarity**; **Parametric mixed integer nonlinear optimization**; **Principal pivoting methods for linear complementarity problems**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**; **Topological methods in complementarity theory**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Convex-simplex algorithm**; **Decomposition techniques for MILP: lagrangian relaxation**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Order complementarity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Principal pivoting methods for linear complementarity problems**; **Sequential simplex method**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**; **Topological methods in complementarity theory**)
integer linear complementary problem
[90C25, 90C33]
(*see:* **Integer linear complementary problem**)
integer linear optimization *see:* Global pairwise protein sequence alignment via mixed- —
integer linear program
[90C35]
(*see:* **Optimization in leveled graphs**)
integer linear program *see:* single parametric mixed —
integer linear programming
[90C10, 90C29]
(*see:* **Multi-objective integer linear programming**)
integer linear programming *see:* mixed —; Multi-objective —; Multiparametric mixed —
integer linear programming: heat exchanger network synthesis *see:* Mixed —
integer linear programming: mass and heat exchanger networks *see:* Mixed —
integer linear programs *see:* Robust optimization: mixed- —
Integer linear programs for routing and protection problems in optical networks
(68M10, 90B18, 90B25, 46N10)
integer MITPs
[90C35]
(*see:* **Multi-index transportation problems**)
integer multi-index transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)
integer nonconvex problem *see:* mixed —
integer nonlinear bilevel programming: deterministic global optimization *see:* Mixed —
integer nonlinear optimization *see:* mixed —; Parametric mixed —
integer nonlinear optimization: A disjunctive cutting plane approach *see:* Mixed- —
integer nonlinear program *see:* mixed —
integer nonlinear programming *see:* mixed —
integer nonlinear programming problem *see:* mixed —
integer optimal control problem *see:* mixed —
Integer Optimization *see:* Mixed —; Multi-class data classification via mixed- —; Peptide identification via mixed- —
integer optimization problem
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
integer optimization problem *see:* mixed —
integer optimization in well scheduling *see:* Mixed —
integer problem *see:* linear zero-one —; mixed —
integer problems *see:* 0–1 mixed —; linear mixed —

integer program
[05-XX, 90C10, 90C11, 90C27, 90C30, 90C57]
(*see:* **Frequency assignment problem**; **Integer programming**; **Lagrangian duality: BASICS**)

integer program
[90C05, 90C06, 90C08, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Modeling difficult optimization problems**)

integer program *see:* mixed —; zero-one —

integer program with recourse *see:* stochastic —

Integer programming
(90C10, 90C11, 90C27, 90C57)
(*referred to in:* **Airline optimization**; **Alignment problem**; **Branch and price: Integer programming with column generation**; **Cutting-stock problem**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **Large scale trust region problems**; **LCP: Pardalos–Rosen mixed integer formulation**; **Maximum cut problem, MAX-CUT**; **Maximum satisfiability problem**; **MINLP: trim-loss problem**; **Mixed integer classification problems**; **Multidimensional knapsack problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Optimization-based visualization**; **Optimization in leveled graphs**; **Parametric mixed integer nonlinear optimization**; **Quadratic knapsack**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stable set problem: branch & cut algorithms**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**; **Vehicle scheduling**)
(*refers to:* **Airline optimization**; **Alignment problem**; **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Maximum satisfiability problem**; **Mixed integer classification problems**; **Multidimensional knapsack problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Optimization in leveled graphs**; **Parametric mixed integer nonlinear optimization**; **Quadratic knapsack**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**; **Vehicle scheduling**)

integer programming
[01A99, 05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68Q20, 68Q99, 68R, 68U, 68W, 90-XX, 90B, 90B50, 90C]
(*see:* **Branch and price: Integer programming with column generation**; **Convex discrete optimization**; **History of optimization**; **Optimal triangulations**; **Optimization and decision support systems**; **Survivable networks**)

integer programming
[13Cxx, 13Pxx, 14Qxx, 68Q99, 90B50, 90B80, 90C05, 90C10, 90C11, 90C27, 90C30, 90C35, 90C46, 90C57, 90Cxx]
(*see:* **Assignment and matching**; **Branch and price: Integer programming with column generation**; **Facilities layout problems**; **Facility location problems with spatial interaction**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **Optimization and decision support systems**; **Optimization in leveled graphs**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**)

integer programming *see:* convex —; cost functions in —; group relaxation in —; mixed —; Multi-objective mixed —; multi-objective (multicriteria) mixed —; n-fold —; Simplicial pivoting algorithms for —; stochastic —; stochastic (mixed- —; test sets in —; zero-one —

Integer programming: algebraic methods
(13Cxx, 13Pxx, 14Qxx, 90Cxx)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic**

integer programming: continuity, stability, rates of convergence; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)

Integer programming: branch and bound methods
(90C10, 90C05, 90C08, 90C11, 90C06)
(*referred to in:* **Biquadratic assignment problem**; **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Discrete stochastic optimization**; **Facilities layout problems**; **Global optimization based on statistical models**; **Global optimization in multiplicative programming**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **Job-shop scheduling problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Maximum satisfiability problem**; **MINLP: trim-loss problem**; **Multidimensional assignment problem**; **Multidimensional knapsack problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Optimization-based visualization**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stable set problem: branch & cut algorithms**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Genetic algorithms**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Linear programming: interior point methods**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)

Integer programming: branch and cut algorithms
(90C10, 90C11, 90C05, 90C08, 90C06)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Quadratic assignment problem**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stable set problem: branch & cut algorithms**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic vehicle routing problems**; **Time-dependent traveling salesman problem**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)

Integer programming with column generation *see:* Branch and price: —

integer programming: complexity and equivalent forms *see:* Quadratic —

integer programming/constraint programming hybrid methods *see:* Mixed —

integer programming: continuity, stability, rates of convergence *see:* Stochastic —

Integer programming: cutting plane algorithms
(90C10, 90C05, 90C08, 90C11, 90C06, 90C08)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **Job-shop scheduling problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Quadratic assignment problem**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stable set problem: branch & cut algorithms**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP:**

lagrangian relaxation; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **NP-complete problems and proof methodology**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)

Integer programming duality
(90C10, 90C46)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Decomposition techniques for MILP: lagrangian relaxation**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **Simplicial pivoting algorithms for integer programming**; **Time-dependent traveling salesman problem**)

Integer programming: lagrangian relaxation
(90C10, 90C30)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Facilities layout problems**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Lagrange, Joseph-Louis**; **Lagrangian multipliers methods for convex programming**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multi-objective optimization: lagrange duality**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization**; **Nondifferentiable optimization: subgradient optimization methods**; **Parametric mixed integer nonlinear optimization**; **Quadratic assignment problem**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Lagrange, Joseph-Louis**; **Lagrangian multipliers methods for convex programming**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multi-objective optimization: lagrange duality**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)

integer programming: models and applications *see:* Multi-quadratic —

integer programming problem
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**; **Integer programming: cutting plane algorithms**)

integer programming problem *see:* large scale nonlinear mixed —; mixed nonlinear —

integer programming problems
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**)

integer programs *see:* mixed —; Stochastic —

integer quadratic programming *see:* mixed- —

integer recourse *see:* simple —; Stochastic programming with simple —; two-stage stochastic programs with simple —

integer rounding
[90C05, 90C06, 90C08, 90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**; **Integer programming: cutting plane algorithms**)

integer rounding cut *see:* mixed —

integer Solution
[90C11]
(*see:* **MINLP: branch and bound methods**)

integer transportation problem *see:* convex —

integer value function *see:* mixed —

integer variable *see:* most/least infeasible —; multiple branches for bounded —

integer variables
[90C11]
(*see:* **MINLP: branch and bound methods**)

integral
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
integral *see:* derivative of an —; gradient of an —; hypergeometric —; indefinite —; multivariate probability —
integral bounds subject to moment conditions *see:* optimal —
integral constraint
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
integral equations
[65H10, 65J15]
(*see:* **Contraction-mapping**)
integral evaluation *see:* asymptotic case of —
integral Fenchel–Legendre transformation
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
integral functions: general theory and examples *see:* Derivatives of probability and —
integral linear fractional combinatorial optimization problem
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
integral Mean-Value for Composite Convexifiable Function
[25A15, 34A05, 90C25, 90C26, 90C30, 90C31]
(*see:* **Convexifiable functions, characterization of**)
integral quadratic constraint
[93D09]
(*see:* **Robust control**)
integral relationships
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
integral over surface formula
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
integral system *see:* totally dual —
integral vector *see:* primitive —; support of an —
integral over a volume
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
integral over volume formula
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
integrality criterion
[90C11, 90C31]
(*see:* **Multiparametric mixed integer linear programming**)
integrality gap
[90C35]
(*see:* **Feedback set problems**)
integrality property
[90C30, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**)
integrality theorem
[90C35]
(*see:* **Maximum flow problem**)
integrals *see:* Approximation of multivariate probability —; lower bounds for multivariate probability —; probability —; upper bounds for multivariate probability —
integrate
(*see:* **State of the art in modeling agricultural systems**)
Integrated planning and scheduling
integrated probabilistic constraint
[90C15]
(*see:* **Static stochastic programming models: conditional expectations**)
integrated probabilistic constraint
[90C15]
(*see:* **Static stochastic programming models: conditional expectations**)
Integrated vehicle and duty scheduling problems
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
integration
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
integration
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
integration *see:* high-dimensional —; problem —
integration of dynamic considerations and controllability
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
integration of surveys
[90C35]
(*see:* **Multi-index transportation problems**)
intelligence *see:* artificial —
intelligent multicriteria decision support system
[90C29]
(*see:* **Decision support systems with multiple criteria**)
intelligent multicriteria decision support systems
[90C29]
(*see:* **Decision support systems with multiple criteria**)
intelligent search
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
intensification
[68M20, 90B06, 90B35, 90B80, 90C59]
(*see:* **Flow shop scheduling problem**; **Heuristic and metaheuristic algorithms for the traveling salesman problem**; **Location routing problem**)
intensification phase
(*see:* **Maximum cut problem, MAX-CUT**)
inter-class distance
[55R15, 55R35, 65K05, 90C11]
(*see:* **Deterministic and probabilistic optimization models for data classification**)
interaction *see:* Facility location problems with spatial —; level of —; spatial —; visual —
interaction of design and control
[49M37, 90C11, 90C29, 90C90]
(*see:* **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**)
interaction of design and control
[49M37, 90C11, 90C29, 90C90]
(*see:* **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**)

interaction of design and control *see:* MINLP: applications in the —; Multi-objective optimization: —
interaction of design, synthesis and control
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
interaction model *see:* spatial- —
interactions *see:* electrostatic —
interactive
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
interactive disaggregation system
[90C29, 91A99]
(*see:* **Preference disaggregation**)
interactive learning of Boolean functions
[90C09]
(*see:* **Inference of monotone boolean functions**)
interactive learning of Boolean functions
[90C09]
(*see:* **Inference of monotone boolean functions**)
interactive method
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**)
interactive method
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**; **Multi-objective optimization; Interactive methods for preference value functions**)
interactive method *see:* visual —
interactive methods
[65K05, 90B50, 90C05, 90C27, 90C29, 90C70, 91B06]
(*see:* **Fuzzy multi-objective linear programming**; **Multi-objective combinatorial optimization**; **Multi-objective optimization and decision support systems**)
interactive methods *see:* computing processes in —
Interactive methods for preference value functions *see:* Multi-objective optimization; —
interactive procedures
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
interactive sampling procedure
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
interactive versus noninteractive methods
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
interchange *see:* fast —
interchange heuristics *see:* k- —
intercommodity constraints
[90C35]
(*see:* **Feedback set problems**)
interconnection designs *see:* subset —
interdisciplinary coupling *see:* bandwidth of —
interdisciplinary feasibility
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
interest rate *see:* riskless —; spot —
interest rate yield curve
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
interest rates *see:* term structure of —
interface *see:* fractal —; graphical user —
interhelical contacts in alpha-helical proteins *see:* Predictive method for —
interior *see:* nonempty —
interior point
[49M29, 65K10, 65L99, 90C06, 90C33, 93-XX]
(*see:* **Linear complementarity problem**; **Local attractors for gradient-related descent iterations**; **Optimization strategies for dynamic systems**)
interior point
[90C20]
(*see:* **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: theory**)
interior point *see:* infeasible —
interior point algorithm
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
interior-point algorithm *see:* infeasible-start —
interior point algorithms
[90C15, 90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**; **Stochastic programming: parallel factorization of structured matrices**)
interior point algorithms
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
interior point algorithms for entropy optimization
[90C25, 90C51, 94A17]
(*see:* **Entropy optimization: interior point methods**)
interior point logarithmic barrier method
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
interior point method
[90C05, 90C09, 90C10]
(*see:* **Linear programming: karmarkar projective algorithm**; **Optimization in boolean classification problems**)
interior point methods
[15A15, 49J52, 65K05, 65K10, 65L99, 90C05, 90C06, 90C08, 90C10, 90C11, 90C20, 90C25, 90C30, 90C34, 90C51, 90C55, 90C60, 90C90, 93-XX, 94A17]
(*see:* **Complexity theory: quadratic programming**; **Entropy optimization: interior point methods**; **Feasible sequential quadratic programming**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Nondifferentiable optimization: Newton method**; **Optimization strategies for dynamic systems**; **Quadratic programming with bound constraints**; **Semidefinite programming and determinant maximization**; **Successive quadratic programming: solution by active sets and interior point methods**)
interior point methods
[15A15, 49K20, 49M99, 65K05, 65K10, 90C05, 90C15, 90C25, 90C27, 90C30, 90C51, 90C55, 90C60, 90C90, 94A17]
(*see:* **ABS algorithms for optimization**; **Complexity theory: quadratic programming**; **Entropy optimization: interior point methods**; **Homogeneous selfdual methods for linear**

programming; **Linear programming: karmarkar projective algorithm**; **Semidefinite programming and determinant maximization**; **Semidefinite programming and structural optimization**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Stochastic programming: parallel factorization of structured matrices**; **Successive quadratic programming**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
interior-point methods *see:* Entropy optimization: —; Linear programming: —; polynomial time —; primal-dual —; Successive quadratic programming: solution by active sets and —
interior point methods for distributed optimal control problems *see:* Sequential quadratic programming: —
interior point methods for the linear complementarity problem *see:* Generalizations of —
Interior point methods for semidefinite programming
(90C51, 90C22, 90C25, 90C05, 90C30)
(*referred to in:* **Duality for semidefinite programming**; **Entropy optimization: interior point methods**; **Homogeneous selfdual methods for linear programming**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Matrix completion problems**; **Potential reduction methods for linear programming**; **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and structural optimization**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Solving large scale and sparse semidefinite programs**; **Standard quadratic optimization problems: theory**; **Successive quadratic programming: solution by active sets and interior point methods**)
interior of a relation *see:* symmetric —
interior of a set
[37A35, 90C05]
(*see:* **Potential reduction methods for linear programming**)
interior solution
[90C25, 90C51, 94A17]
(*see:* **Entropy optimization: interior point methods**)
interiors
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
interlining
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
interlocking eigenvalue theorem
[65K05, 90Cxx]
(*see:* **Symmetric systems of linear equations**)
intermediate fill-in
[65Fxx]
(*see:* **Least squares problems**)
intermediate form *see:* AD —
intermediate scale network
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
intermediate storage *see:* unlimited —
intermediate term
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
intermediate terms *see:* Interval analysis: —
intermediate variables
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)
internal coordinates
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
internal coordinates
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
internal deviation
[62H30, 68T10, 90C05]
(*see:* **Linear programming models for classification**)
internal energy
[90C90]
(*see:* **Optimization in medical imaging**)
internal numerical differentiation
[34-xx, 34Bxx, 34Lxx, 93E24]
(*see:* **Complexity and large-scale least squares problems**)
International Zeolite Association *see:* atlas of the —
Internet
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
interplay between primal and dual solutions *see:* exploiting the —
interpolation *see:* Nyström —
interpolation parametric eigenvalue formulation *see:* inverse —
interpolation problem
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
interpolatory operator
[90C34]
(*see:* **Semi-infinite programming: approximation methods**)
interpolatory operator *see:* nonnegative —
interpretation
[65H99, 65K99]
(*see:* **Automatic differentiation: point and interval**)
interpretation *see:* epistemological —; geometric —; objective —; subjective —
interpretation, existence and uniqueness *see:* Variational inequalities: geometric —
intersatured vertices
[90C35]
(*see:* **Feedback set problems**)
intersection
[03B52, 03E72, 47S40, 65G20, 65G30, 65G40, 68T27, 68T35, 68Uxx, 90Bxx, 90C26, 90C30, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Bounding derivative ranges**; **Interval analysis: systems of nonlinear equations**)
intersection *see:* transversal —
intersection cut
[90C11]
(*see:* **MINLP: branch and bound methods**)

intersection cut
[90C26]
(*see:* **Cutting plane methods for global optimization**)
intersection graph model
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
intersection problem *see:* convex —
interva input
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: differential equations**)
interval *see:* Automatic differentiation: point and —; confidence —; critical —; expectation-maximization —; machine —
interval algebra *see:* relational —
interval analysis
[49M37, 65G20, 65G30, 65G40, 65H20, 65K99, 90C11]
(*see:* **Interval Newton methods**; **Mixed integer nonlinear programming**)
interval analysis
[65C20, 65G20, 65G30, 65G40, 65H20, 65L99, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**)
Interval analysis: application to chemical engineering design problems
(65C20, 65G20, 65G30, 65G40, 90C90, 65H20)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Design optimization in computational fluid dynamics**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Multidisciplinary design optimization**; **Multilevel methods for optimal design**; **Optimal design of composite structures**; **Optimal design in nonlinear optics**; **Structural optimization: history**)
(*refers to:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bilevel programming: applications in engineering**; **Bounding derivative ranges**; **Design optimization in computational fluid dynamics**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Multidisciplinary design optimization**; **Multilevel methods for optimal design**; **Optimal design of composite structures**; **Optimal design in nonlinear optics**; **Structural optimization: history**)
interval analysis and balanced interval arithmetic *see:* Global optimization: —
Interval analysis: differential equations
(65G20, 65G30, 65G40, 65L99)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
(*refers to:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
Interval analysis: eigenvalue bounds of interval matrices
(65G20, 65G30, 65G40, 65L99)
(*referred to in:* **αBB algorithm**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Eigenvalue enclosures for ordinary differential equations**; **Global optimization: application to phase equilibrium problems**; **Hemivariational inequalities: eigenvalue problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Semidefinite programming and determinant maximization**; **Standard quadratic optimization problems: algorithms**)
(*refers to:* **αBB algorithm**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Eigenvalue enclosures for ordinary differential equations**; **Global optimization: application to phase equilibrium problems**;

Hemivariational inequalities: eigenvalue problems; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Semidefinite programming and determinant maximization**)

Interval analysis: intermediate terms
(65G20, 65G30, 65G40, 65H20)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
(*refers to:* **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)

Interval analysis: nondifferentiable problems
(65G20, 65G30, 65G40, 65H20)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
(*refers to:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)

Interval analysis for optimization of dynamical systems

Interval analysis: parallel methods for global optimization
(65K05, 65Y05, 65Y10, 65Y20, 68W10)
(*referred to in:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Global optimization: interval analysis and balanced interval arithmetic**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Load balancing for parallel optimization techniques**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Stochastic network problems: massively parallel solution**)
(*refers to:* **Interval analysis: intermediate terms**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval global optimization**; **Interval Newton methods**)

Interval analysis: subdivision directions in interval branch and bound methods
(65K05, 90C30)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Global optimization: interval analysis and balanced interval arithmetic**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: systems of nonlinear equations**; **Interval analysis:**

unconstrained and constrained optimization; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
(*refers to:* **Automatic differentiation: introduction, history and rounding error estimation**; **Interval analysis: unconstrained and constrained optimization**; **Interval Newton methods**; **MINLP: branch and bound global optimization algorithm**)
Interval analysis: systems of nonlinear equations
(65G20, 65G30, 65G40)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Contraction-mapping**; **Global optimization: application to phase equilibrium problems**; **Global optimization methods for systems of nonlinear equations**; **Gröbner bases for polynomial equations**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Nonlinear least squares: Newton-type methods**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
(*refers to:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Contraction-mapping**; **Global optimization: application to phase equilibrium problems**; **Global optimization methods for systems of nonlinear equations**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**; **Nonlinear least squares: Newton-type methods**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
Interval analysis: unconstrained and constrained optimization
(65G20, 65G30, 65G40, 65H20)
(*referred to in:* **Algorithmic improvements using a heuristic parameter, reject index for interval optimization**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Direct search Luus—Jaakola optimization procedure**; **Global optimization: application to phase equilibrium problems**; **Global optimization: interval analysis and balanced interval arithmetic**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
(*refers to:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Direct search Luus—Jaakola optimization procedure**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
Interval analysis: verifying feasibility
(65G20, 65G30, 65G40, 65H20)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
(*refers to:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
interval arithmetic
[49M37, 65G20, 65G30, 65G40, 65H99, 65K05, 65K10, 65K99, 65T40, 68T20, 90C26, 90C30, 90C57, 90C90]

(*see:* **αBB algorithm**; **Automatic differentiation: point and interval**; **Bounding derivative ranges**; **Global optimization: interval analysis and balanced interval arithmetic**; **Global optimization methods for harmonic retrieval**; **Interval constraints**)
Interval arithmetic
[15A99, 49M37, 65G20, 65G30, 65G40, 65K05, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Interval linear systems**)
interval arithmetic *see:* balanced —; balanced random —; Global optimization: interval analysis and balanced —; inclusion principle of machine —; inner —; machine —; random —
interval arithmetic operation
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
interval automatic differentiation
[65H99, 65K99]
(*see:* **Automatic differentiation: point and interval**)
interval branch and bound methods *see:* Interval analysis: subdivision directions in —
interval computations
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval fixed point theory**; **Interval Newton methods**)
interval computing
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
Interval constraints
(68T20, 65G20, 65G30, 65G40)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
(*refers to:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
interval dependency
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
interval dependency
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
interval diagram *see:* composition —; temperature —
interval enclosure
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
interval equation *see:* linear —
interval extension
[65H20, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**; **LP strategy for interval-Newton method in deterministic global optimization**)
interval extension *see:* natural —
Interval fixed point theory
(65G20, 65G30, 65G40, 65H20)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
(*refers to:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
interval function *see:* pre-declared —
Interval global optimization
(65K05, 90C30, 65G20, 65G30, 65G40)
(*referred to in:* **αBB algorithm**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**;

Global optimization in the analysis and management of environmental systems; **Global optimization: application to phase equilibrium problems**; **Global optimization in batch design under uncertainty**; **Global optimization in generalized geometric programming**; **Global optimization: interval analysis and balanced interval arithmetic**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval linear systems**; **Interval Newton methods**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Mixed integer nonlinear programming**; **Smooth nonlinear nonconvex optimization**)
(*refers to:* **αBB algorithm**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Global optimization in the analysis and management of environmental systems**; **Global optimization: application to phase equilibrium problems**; **Global optimization in batch design under uncertainty**; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval linear systems**; **Interval Newton methods**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**; **Mixed integer nonlinear programming**; **Smooth nonlinear nonconvex optimization**)

interval graph
[05C85, 90C35]
(*see:* **Directed tree networks**; **Feedback set problems**)

interval Hessian matrix
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)

interval inference *see:* fuzzy —

interval linear system
[15A99, 65G20, 65G30, 65G40, 90C26]
(*see:* **Interval linear systems**)

interval linear system *see:* center of an —

Interval linear systems
(15A99, 65G20, 65G30, 65G40, 90C26)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Global optimization: interval analysis and balanced interval arithmetic**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval Newton methods**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Cholesky factorization**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval Newton methods**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Linear programming**; **Nonlinear least squares: trust region methods**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)

interval logic
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)

interval logic system of approximate reasoning
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)

interval matrices *see:* Interval analysis: eigenvalue bounds of —

interval matrix *see:* complex —; dimensional symmetric —; extreme eigenvalue of an —; Hermitian —; interval of variation of an eigenvalue of an —; real —; real symmetric —; vertex matrix of an —

interval methods
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)

interval methods
[65G20, 65G30, 65G40, 65K05, 65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**; **Interval global optimization**; **Random search methods**)
interval Newton
[65C20, 65G20, 65G30, 65G40, 65H20, 90C90]
(*see:* **Interval analysis: application to chemical engineering design problems**)
interval Newton
[65H20, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**)
interval Newton algorithm
[65H20, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**)
interval Newton iteration
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
interval Newton method
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval analysis: systems of nonlinear equations**; **Interval Newton methods**)
interval Newton method *see:* Krawczyk variation of the —; multivariate —; univariate —
interval-Newton method in deterministic global optimization *see:* LP strategy for —
Interval Newton methods
(65G20, 65G30, 65G40, 65H20, 65K99)
(*referred to in:* **Algorithmic improvements using a heuristic parameter, reject index for interval optimization**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Nondifferentiable optimization: Newton method**; **Nonlinear least squares: Newton-type methods**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
(*refers to:* **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Complexity classes in optimization**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Nondifferentiable optimization: Newton method**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
interval Newton methods
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: unconstrained and constrained optimization**)
interval Newton methods *see:* existence-proving properties of —
interval Newton operator
[65G20, 65G30, 65G40]
(*see:* **Interval analysis: systems of nonlinear equations**)
interval Newton operator
[65G20, 65G30, 65G40]
(*see:* **Interval analysis: systems of nonlinear equations**)
interval Newton operator *see:* univariate —
interval operator
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
interval optimization *see:* Algorithmic improvements using a heuristic parameter, reject index for —
interval order
[90C29]
(*see:* **Preference modeling**)
interval package *see:* variable precision —
interval pairs *see:* fuzzy —
Interval propagation
(68T20, 65G20, 65G30, 65G40)
(*referred to in:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**; **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
(*refers to:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**; **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Interval analysis: differential equations**; **Interval analysis: eigenvalue bounds of interval matrices**;

Interval analysis: intermediate terms; **Interval analysis: nondifferentiable problems**; **Interval analysis: parallel methods for global optimization**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval analysis: systems of nonlinear equations**; **Interval analysis: unconstrained and constrained optimization**; **Interval analysis: verifying feasibility**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Interval Newton methods**)
interval propagation
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
interval slopes
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: nondifferentiable problems**)
interval Taylor operator
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
interval taylor operators *see:* Automatic differentiation: point and —
interval-valued approximate inference
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
interval of variation of an eigenvalue of an interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
intervals *see:* floating point —; overlap of —
INTLIB
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
INTOPT_90
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
intra-class distance
[55R15, 55R35, 65K05, 90C11]
(*see:* **Deterministic and probabilistic optimization models for data classification**)
intractible problem
[90C60]
(*see:* **Complexity theory**)
introduction, history and overview *see:* Bilevel programming: —
introduction, history and rounding error estimation *see:* Automatic differentiation: —
iNV
(*see:* **Integrated planning and scheduling**)
invariance criterion *see:* scale —
invariance model *see:* sign- —
invariant *see:* scale- —; shift- —
invariant convex
[90C26]
(*see:* **Invexity and its applications**)
invariant set
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
invariant under principal pivoting *see:* matrix class —
invariants
[05C85]
(*see:* **Directed tree networks**)
invariants *see:* structure —
inventory control
[49L20]
(*see:* **Dynamic programming: inventory control**)
inventory control *see:* Dynamic programming: —
inventory control problem
[49L20]
(*see:* **Dynamic programming: inventory control**)
inventory management
[90B50]
(*see:* **Inventory management in supply chains**)
inventory management *see:* multistage —
inventory management models *see:* multistage —; single stage —
Inventory management in supply chains
(90B50)
(*referred to in:* **Global supply chain models**; **Nonconvex network flow problems**; **Operations research models for supply chain management and design**; **Piecewise linear network flow problems**)
(*refers to:* **Global supply chain models**; **Nonconvex network flow problems**; **Operations research models for supply chain management and design**; **Piecewise linear network flow problems**)
inventory models: (*QR*) policy *see:* Continuous review —
Inventory Ordering *see:* zero- —
inventory placement
[90-02]
(*see:* **Operations research models for supply chain management and design**)
inventory routing
[65H20, 65K05, 90-01, 90B06, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**; **Vehicle routing**)
inventory routing problems *see:* Maritime —
inventory ship routing problem
(*see:* **Maritime inventory routing problems**)
inventory systems
[90-02]
(*see:* **Operations research models for supply chain management and design**)
inventory and transportation decisions
[90-02]
(*see:* **Operations research models for supply chain management and design**)
inverse
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
inverse *see:* approximate —; Moore–Penrose pseudo- —; pseudo- —
inverse differentiability
[90C15]
(*see:* **Derivatives of probability measures**)
inverse interpolation parametric eigenvalue formulation
[90C30]
(*see:* **Large scale trust region problems**)

inverse problem of the calculus of variations
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
inverse problems *see:* formulation and solution of —
inverse product of relations *see:* self- —
inverse quasi-Newton updating
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
inverse relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
inverses *see:* generalized —
inverted transformation
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
investment *see:* maximization of return on —; venture capital —
investment analysis
[91B06, 91B60]
(*see:* **Financial applications of multicriteria analysis**)
investment decision
[91B06, 91B60]
(*see:* **Financial applications of multicriteria analysis**)
investment decisions *see:* diversified —
investments *see:* optimal —
invex
[90C26]
(*see:* **Invexity and its applications**)
invex *see:* generalized —; pseudo- —; quasi- —; V- —
invex function
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
invex function
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
invex function *see:* pre- —
invex set
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
invex set *see:* pre- —
Invexity and its applications
(90C26)
(*referred to in:* **Generalized concavity in multi-objective optimization**; **L-convex functions and M-convex functions**)
(*refers to:* **Generalized concavity in multi-objective optimization**; **Isotonic regression problems**; **L-convex functions and M-convex functions**)
invexity at a point
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
invexity with respect to a set
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
invexity with respect to a set *see:* pre- —
iNVO
(*see:* **Integrated planning and scheduling**)
involutory operator
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
involving QD-superpotentials *see:* convex variational inequality for an elastostatic problem —; elastostatic problem —; variational equality for an elastostatic problem —
Ioffe–Burke local dualization
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
ions *see:* complementary —
IP
[68Q99]
(*see:* **Branch and price: Integer programming with column generation**)
IPE
[90C30]
(*see:* **Large scale trust region problems**)
IPH function
[90C26]
(*see:* **Global optimization: envelope representation**)
iPMN
[65K10, 90C33, 90C51]
(*see:* **Generalizations of interior point methods for the linear complementarity problem**)
IPP
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
IQC
[93D09]
(*see:* **Robust control**)
IQML method
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
irreducible components of a matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
irreducible matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
irreducible matrix *see:* inductive structure of an —
irredundant constraint
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
irregular operations
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
irregular operations problem
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
Ising glass model
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
isodose
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
isolated local minimizer
[90C26, 90C31]
(*see:* **Sensitivity and stability in NLP: continuity and differential stability**; **Smooth nonlinear nonconvex optimization**)
isolated stationary point
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)

isolation *see:* cluster —
isomorphic graphs
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
isomorphism *see:* graph —; maximal similarity subtree —; maximal subtree —; maximum similarity subtree —; maximum subtree —; subtree —
isomorphism problem *see:* graph —
isotone Boolean function
[90C09]
(*see:* **Inference of monotone boolean functions**)
isotone functions
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
isotone inclusion function
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
isotone mapping *see:* Φ- —
isotone monotone Boolean function
[90C09]
(*see:* **Inference of monotone boolean functions**)
isotone operator
[90C33]
(*see:* **Order complementarity**)
isotone optimization
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
isotone projection
[90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**)
isotone projection cone
[90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**)
isotonic function
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
isotonic medium regression
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
isotonic regression
[41A30, 62G07, 62G30, 62J02, 65K05, 90C26]
(*see:* **Isotonic regression problems**; **Regression by special functions: algorithms and complexity**)
isotonic regression *see:* simple order —
isotonic regression problem
[41A30, 62G07, 62G30, 62J02, 65K05, 90C26]
(*see:* **Isotonic regression problems**; **Regression by special functions: algorithms and complexity**)
Isotonic regression problems
(62G07, 62G30, 65K05)
(*referred to in:* **Generalized concavity in multi-objective optimization**; **Invexity and its applications**; **L-convex functions and M-convex functions**; **Regression by special functions: algorithms and complexity**)
(*refers to:* **Regression by special functions: algorithms and complexity**)
isotonic regression problems *see:* algorithms for —
isotonicity property
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
issues in classification *see:* computational —
Itakura–Saito divergence
[90C05, 90C25]
(*see:* **Young programming**)
iterate *see:* dead-point —
iterated local search
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
iterated tour partitioning algorithm *see:* K- —
iterates *see:* feasible —
iteration
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
iteration *see:* control vector —; direct —; fixed point —; Gauss–Seidel —; Gauss–Seidel value —; Gram–Schmidt type —; interval Newton —; Jacobi —; Petrov–Galerkin —; policy —; relative value —; Richardson —; value —
iteration BCI *see:* Boundary condition —
iteration CVI *see:* Control vector —
iteration method *see:* boundary condition —
iterations *see:* Local attractors for gradient-related descent —
iterative algorithms *see:* asynchronous —
iterative deepening
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
iterative dynamic programming
[65L99, 90C30, 93-XX]
(*see:* **Boundary condition iteration BCI**; **Direct search Luus—Jaakola optimization procedure**; **Dynamic programming: optimal control applications**; **Optimization strategies for dynamic systems**; **Suboptimal control**)
iterative function system
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
iterative linear equation-solving
[90C25, 90C30]
(*see:* **Successive quadratic programming: solution by active sets and interior point methods**)
iterative method
[90C33]
(*see:* **Linear complementarity problem**)
iterative method
[65H10, 65J15]
(*see:* **Contraction-mapping**)
iterative method *see:* Chebyshev —; partially asynchronous —
iterative model refinement
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
iterative quadratic maximum likelihood method
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)

iterative regularization *see:* Tikhonov —
iterative regularization method
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
iterative scheme
[91B50]
(*see:* **Walrasian price equilibrium**)
iterative scheme *see:* chaotic —
iterative solution algorithm *see:* incremental- —
iterative solution of the Euclidean distance location problem
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
IV *see:* Prolog —
IVP
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)

J

J-normal primal problem
[90C29, 90C30]
(*see:* **Multi-objective optimization: lagrange duality**)
J-stable primal problem
[90C29, 90C30]
(*see:* **Multi-objective optimization: lagrange duality**)
j-VNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
Jacobi
[49L20, 90C39]
(*see:* **Dynamic programming: discounted problems**)
Jacobi
[90C30]
(*see:* **Cost approximation algorithms**)
Jacobi algorithm
[68W10, 90B15, 90C06, 90C30]
(*see:* **Cost approximation algorithms**; **Stochastic network problems: massively parallel solution**)
Jacobi–Bellman equation *see:* derivation of the Hamilton– —; Hamilton– —; solution of the Hamilton– —; sufficiency theorem for the Hamilton– —
Jacobi equation *see:* Hamilton– —
Jacobi inequality *see:* Hamilton– —
Jacobi iteration
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **QR factorization**)
Jacobi method
[90C33]
(*see:* **Linear complementarity problem**)
Jacobian *see:* accumulation of the —; approximate —; Clarke generalized —; preaccumulation of the —
Jacobian consistency property
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
Jacobian matrix
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**)
Jacobian matrix
[65K05, 90C25, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
Jacobians, and Hessians *see:* Complexity of gradients —
James–Stein estimators
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
James sup theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
Jaynes
[94A17]
(*see:* **Jaynes' maximum entropy principle**)
Jaynes entropy concentration theorem
[94A17]
(*see:* **Jaynes' maximum entropy principle**)
Jaynes maximum entropy
[62F10, 94A17]
(*see:* **Entropy optimization: parameter estimation**)
Jaynes' maximum entropy principle
(94A17)
(*referred to in:* **Entropy optimization: interior point methods**; **Entropy optimization: parameter estimation**; **Entropy optimization: shannon measure of entropy and its properties**; **Maximum entropy principle: image reconstruction**)
(*refers to:* **Entropy optimization: interior point methods**; **Entropy optimization: parameter estimation**; **Entropy optimization: shannon measure of entropy and its properties**; **Maximum entropy principle: image reconstruction**)
jensen's inequality
[90C15]
(*see:* **Multistage stochastic programming: barycentric approximation**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
Jensen lower bound
[90C15]
(*see:* **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
Jerrum conjecture
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
JJT-regular problem
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
job-shop
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
job-shop
[90B35]
(*see:* **Job-shop scheduling problem**)

job-shop problem
[62C10, 65K05, 90B35, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**; **Job-shop scheduling problem**)
Job-shop scheduling problem
(90B35)
(*referred to in:* **MINLP: design and scheduling of batch processes**; **Vehicle scheduling**)
(*refers to:* **Complexity classes in optimization**; **Complexity theory**; **Genetic algorithms**; **Integer programming: branch and bound methods**; **Integer programming: cutting plane algorithms**; **Linear programming**; **MINLP: design and scheduling of batch processes**; **Simulated annealing**; **Stochastic scheduling**; **Vehicle scheduling**)
John *see:* Von Neumann —
John conditions *see:* Fritz —
John generalized conditions *see:* Fritz —
John necessary optimality conditions *see:* fritz —
John rule *see:* Fritz —
John system *see:* Fritz —
John type condition *see:* Fritz —
Johnson linearization *see:* Adams– —
join
[90C35]
(*see:* **Multi-index transportation problems**)
join procedure
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
joined sets
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
joining *see:* neighbor —
joint continuity
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)
joint probabilistic constraint
[90C15]
(*see:* **Static stochastic programming models**)
jointly convex problem
[90C31]
(*see:* **Sensitivity and stability in NLP: continuity and differential stability**)
Jones *see:* Lennard- —
Jones microcluster *see:* Lennard– —
Jones and morse clusters *see:* Global optimization in Lennard– —
Jones potential energy function *see:* Lennard- —
Jongen–Jonker–Twilt *see:* problem regular in the sense of —
Jonker–Twilt *see:* problem regular in the sense of Jongen– —
Jordan–Hahn decomposition
[90C15]
(*see:* **Derivatives of probability measures**)
Joseph-Louis *see:* Lagrange —
journal of Heuristics
[68T20, 68T99, 9008, 90C26, 90C27, 90C59]
(*see:* **Metaheuristics**; **Variable neighborhood search methods**)
Journal of Management Mathematics *see:* iMA —
Journal of Operational Research *see:* european —
judgemental forecast
[90C26, 90C30]
(*see:* **Forecasting**)
judgment *see:* pairwise —
judgment matrix *see:* consistent —
judgments *see:* incomplete —
jump across a fault
[90Cxx]
(*see:* **Discontinuous optimization**)
jump direction
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
jump system *see:* finite —
jumps of optimal solutions
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
junction nodes *see:* physical —
Jünger–Mutzel branch and cut algorithm
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
justice *see:* rule of —
justification *see:* chain —; left-chain —; right-chain —

K

K *see:* commutator —; multivariable stability margin —
k-CNF
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
k-CNF *see:* SAT- —
k-colorable
[05C15, 05C62, 05C69, 05C85, 90C27, 90C35, 90C59]
(*see:* **Graph coloring**; **Optimization problems in unit-disk graphs**)
K-convex function
[49L20]
(*see:* **Dynamic programming: inventory control**)
K-convexity
[49L20]
(*see:* **Dynamic programming: inventory control**)
k-dimensional cube connected cycle
[90C35]
(*see:* **Feedback set problems**)
k-exchange neighborhood
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
k-face
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements in optimization**)
k-index transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)
k-interchange heuristics
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
K-iterated tour partitioning algorithm
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)

K-L type neighborhood structure for the QAP
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
k-level crossing minimization
[90C35]
(*see:* **Optimization in leveled graphs**)
k-level hierarchy
[90C35]
(*see:* **Optimization in leveled graphs**)
k-level planarization problem
[90C35]
(*see:* **Optimization in leveled graphs**)
k-leveled graph
[90C35]
(*see:* **Optimization in leveled graphs**)
k-leveled graph *see:* proper —
K-local bilinear form
[90C33]
(*see:* **Order complementarity**)
K-*local inner product*
[90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**)
K-majorant *see:* smallest —
k-means criterion
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
K-minorant *see:* greatest —
k-neighbor
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
k-neighborhood
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
k-Opt
[68Q25, 68R10, 68W40, 90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Domination analysis in combinatorial optimization**; **Traveling salesman problem**)
k-optimality
[68Q20]
(*see:* **Optimal triangulations**)
k-relations
[05C85]
(*see:* **Directed tree networks**)
k-restrictive
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
k-restrictive multilayer
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
k-separation
[90C09, 90C10]
(*see:* **Matroids**)
k-set-contraction
[90C33]
(*see:* **Order complementarity**)
k-Steiner ratio
[90C27]
(*see:* **Steiner tree problems**)
k-tree
[90C27]
(*see:* **Steiner tree problems**)
k-way graph partitioning problem
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
k-way polytope
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
k-way table
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
k-way transportation polytope
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
Kackmartz method
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
kalman filter
(*see:* **Bayesian networks**)
Kalmanson matrix
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Kantorovich
[01A99]
(*see:* **Kantorovich, Leonid Vitalyevich**)
Kantorovich–Karush–Kuhn–Tucker equations
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
Kantorovich, Leonid Vitalyevich
(01A99)
(*referred to in:* **History of optimization**; **Linear programming**)
(*refers to:* **History of optimization**; **Linear programming**)
Kantorovich scheme
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
Kanzow–Smale function *see:* Chen–Harker– —
Karmarkar algorithm
[90C05]
(*see:* **Linear programming: karmarkar projective algorithm**)
Karmarkar method
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
Karmarkar potential function
[37A35, 90C05]
(*see:* **Potential reduction methods for linear programming**)
karmarkar projective algorithm *see:* Linear programming: —
Karush–Kuhn–Tucker conditions
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
Karush–Kuhn–Tucker conditions
[90C30]
(*see:* **Convex-simplex algorithm**)
Karush–Kuhn–Tucker conditions *see:* generalized —
Karush–Kuhn–Tucker equations *see:* Kantorovich– —

Karush–Kuhn–Tucker optimality conditions
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
Karush–Kuhn–Tucker point
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
Karush–Kuhn–Tucker point
[58E05, 90C26, 90C30]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Topology of global optimization**)
Karush–Kuhn–Tucker type condition
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
Karwan Kth-best algorithm *see:* Bialas– —
Kataoka principle
[90C15]
(*see:* **Stochastic programming models: random objective**)
Kaucher arithmetic
[65G30, 65G40, 65K05, 90C30, 90C57]
(*see:* **Global optimization: interval analysis and balanced interval arithmetic**)
Kaufman–Broeckx linearization
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Kaufmann–Stewart reorthogonalized Gram–Schmidt algorithm *see:* Daniel–Gragg– —
kB-consistency
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
Keifer–Wolfowitz method
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
Kelley's classical cutting plane method
[46N10, 90-00, 90C47]
(*see:* **Nondifferentiable optimization**)
Kelley cutting plane method
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
Kelley method
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
kernel
[49-01, 49K45, 49N10, 90-01, 90C20, 90C27, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**)
kernel *see:* Markov —
kernel estimates
[90C15]
(*see:* **Extremum problems with probability functions: kernel type solution methods**)
kernel function
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
kernel of a graph
[90-XX]
(*see:* **Outranking methods**)
kernel transformation
[55R15, 55R35, 65K05, 90C11]
(*see:* **Deterministic and probabilistic optimization models for data classification**)
kernel type solution methods *see:* Extremum problems with probability functions: —
Kernighan neighborhood *see:* Lin– —
key variables
[49M25, 90-08, 90C05, 90C06, 90C08, 90C15]
(*see:* **Simple recourse problem: primal method**)
keys method *see:* random —
keywords
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
keywords
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
KH-regular problem
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
Kharitonov theorem
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)
Kimura maximum principle
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
kinematically admissible displacement
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
kinetic coefficients *see:* estimation of —
kinetically admissible space
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
kinetics and transport *see:* Identification methods for reaction —
Kirchhoff-condition
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
KKT
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
KKT-based method
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
KKT conditions
[49M37, 65K05, 90C26, 90C30, 90C39]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Second order optimality conditions for nonlinear optimization**)
KKT conditions
[90C26, 90C30, 90C33, 90C39]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**; **Second order optimality conditions for nonlinear optimization**)
KKT conditions *see:* first order —
kKT necessary optimality conditions
[49M37, 65K05, 90C30]
(*see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**)

KKT necessary optimality conditions
[49M37, 65K05, 90C30]
(*see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**)
KKT necessary optimality conditions *see:* Equality-constrained nonlinear programming: —
KKT optimality conditions
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
kKT point
[90C60]
(*see:* **Complexity theory: quadratic programming**)
KKT point *see:* global minimum —
KKT points
[58E05, 90C30]
(*see:* **Topology of global optimization**)
KKT points *see:* successive improvement of —
KKT stationarity conditions
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
Klee–Minty examples
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
Klee–Minty examples
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
Klee–Minty examples *see:* Linear programming: —
Kleene–Dienes implication
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
Klein 4-element group
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
Klötzler duality
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
knapsack *see:* fractional 0–1 —; multiconstraint —; multidimensional —; multiple choice —; Quadratic —
knapsack constraint
[68Q99, 90C10, 90C27]
(*see:* **Branch and price: Integer programming with column generation**; **Multidimensional knapsack problems**)
knapsack constraint
[90C20, 90C60]
(*see:* **Quadratic knapsack**)
knapsack cut
[90C11]
(*see:* **MINLP: branch and bound methods**)
knapsack problem
[05-04, 62C10, 65K05, 68Q99, 90C05, 90C10, 90C11, 90C15, 90C26, 90C27, 90C29, 90C30, 90C57, 90C90]
(*see:* **Bayesian global optimization**; **Branch and price: Integer programming with column generation**; **Chemical process planning**; **Evolutionary algorithms in combinatorial optimization**; **Integer programming**; **Kuhn–Tucker optimality conditions**; **Multi-objective combinatorial optimization**; **Simplicial pivoting algorithms for integer programming**)
knapsack problem
[90B90, 90C05, 90C10, 90C59, 90C60]
(*see:* **Complexity theory: quadratic programming**; **Cutting-stock problem**; **Simplicial pivoting algorithms for integer programming**)
knapsack problem *see:* bi- —; bidimensional —; convex quadratic —; linear multiple-choice —; m-dimensional —; multi- —; multiconstraint —; multidimensional —; multidimensional multiple-choice —; multidimensional zero-one —; multiple —; multiple-choice —; zero-one —
knapsack problems
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**)
knapsack problems *see:* Multidimensional —
Knaster–Kuratowski–Mazurkiewicz lemma
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
Knizhnik–Zamolodchikov differential equation
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
knowledge
(*see:* **Planning in the process industry**)
knowledge *see:* algorithmic —; declarative —; state of —
knowledge-based NP methods
[90C11, 90C59]
(*see:* **Nested partitions optimization**)
knowledge of a probability distribution *see:* incomplete —
Kohout compatibility theorem *see:* Bandler– —
Kojima function
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
Kojima–Hirabayashi *see:* problem regular in the sense of —
Kojima–Shindo method
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
Kolmogorov–Chaitin complexity *see:* Solomonoff– —
Kolmogorov complexity
(90C60)
(*referred to in:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**)
(*refers to:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Mixed integer nonlinear programming**; **Parallel computing: complexity classes**)
Kolmogorov complexity
[90C60]
(*see:* **Kolmogorov complexity**)
Kolmogorov complexity *see:* conditional —

Kolmogorov ε-entropy
[01A60, 03B30, 54C70, 68Q17]
(*see:* **Hilbert's thirteenth problem**)
Koopmans–Beckmann QAP
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Koopmans–Beckmann quadratic assignment problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
KP
[90B06, 90B10, 90C26, 90C35]
(*see:* **Minimum concave transportation problems**)
Kramers equation *see:* Smoluchowski– —
Krawczyk method
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval fixed point theory**; **Interval Newton methods**)
Krawczyk variation of the interval Newton method
[65G20, 65G30, 65G40]
(*see:* **Interval analysis: systems of nonlinear equations**)
Krein–Milman theorem
(90C05)
(*referred to in:* **Carathéodory theorem**; **Linear programming**)
(*refers to:* **Carathéodory theorem**; **Linear programming**)
Kreisselmeier–Steinhauser function
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
Kremser equation
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**)
Kruskal algorithm
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
Kruskal algorithm *see:* modified —
Kruskal τ_b statistic *see:* Goodman– —
Krylov space type methods
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
KSM
[90C15]
(*see:* **Extremum problems with probability functions: kernel type solution methods**)
KT conditions
[90C30]
(*see:* **Kuhn–Tucker optimality conditions**)
KT conditions *see:* nonstoichiometric form of —; stoichiometric form of —
KT equations
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
KT point
[90C30]
(*see:* **Nonlinear least squares problems**)
KT point
[90C30]
(*see:* **Kuhn–Tucker optimality conditions**)
Kth-best algorithm *see:* Bialas–Karwan —
kth directional derivative
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
kth order form of coordinates
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
kth order hypodifferential
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
Kuhn–Tucker approach
[49-01, 49K10, 49M37, 90-01, 90C05, 90C27, 91B52]
(*see:* **Bilevel linear programming**)
Kuhn–Tucker conditions
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
Kuhn–Tucker conditions
[90C20, 90C30]
(*see:* **Successive quadratic programming**; **Successive quadratic programming: decomposition methods**)
Kuhn–Tucker conditions *see:* generalized Karush– —; Karush– —
Kuhn–Tucker conditions for quadratic programming sub-problems
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
Kuhn–Tucker CQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
Kuhn–Tucker equations
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
Kuhn–Tucker equations *see:* Kantorovich–Karush– —
Kuhn–Tucker necessary optimality conditions
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
Kuhn–Tucker optimality condition
[90C30]
(*see:* **Nonlinear least squares problems**)
Kuhn–Tucker optimality condition
[90C30]
(*see:* **Nonlinear least squares problems**)
Kuhn–Tucker optimality conditions
(90C30)
(*referred to in:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **High-order necessary conditions for optimality for abnormal points**; **Implicit lagrangian**; **Inequality-constrained nonlinear optimization**; **Lagrangian duality: BASICS**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Farkas lemma**; **First order constraint qualifications**; **High-order necessary conditions for optimality for abnormal points**; **Implicit lagrangian**; **Inequality-constrained nonlinear optimization**; **Lagrangian duality: BASICS**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint**

qualifications; **Second order optimality conditions for nonlinear optimization**)
Kuhn–Tucker optimality conditions *see:* Karush– —
Kuhn–Tucker point
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
Kuhn–Tucker point *see:* Karush– —
Kuhn–Tucker points *see:* multiple —; multiple QP —
Kuhn–Tucker type condition *see:* Karush– —
Kullback–Leibler cross-entropy
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
Kullback–Leibler divergence
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
Kullback–Leibler measure of cross-entropy
[62F10, 94A17]
(*see:* **Entropy optimization: parameter estimation**)
Kuratowski convergence *see:* discrete Painlevé– —
Kuratowski–Mazurkiewicz lemma *see:* Knaster– —
kurtosis
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
KyFan function
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)

L

(l) *see:* tie-up-time —
L-BFGS
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
L-convex function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
L_2-convex function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
$L^\natural$-convex function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
L-convex functions *see:* Fenchel-type duality for M- and —
L-convex functions and M-convex functions
(90C27, 90C25, 90C10, 90C35)
(*referred to in:* **Invexity and its applications**)
(*refers to:* **Generalized concavity in multi-objective optimization**; **Invexity and its applications**; **Isotonic regression problems**)
L-convex set
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
$L^\natural$-convex set
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
L-convexity
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
L_1-distance
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
L_1-distance
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
L_∞-distance
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
L_∞-distance
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
L_p-distance
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
L_p-distance
[90B80, 90B85, 90C27]
(*see:* **Single facility location: multi-objective rectilinear distance location**; **Voronoi diagrams in facility location**)
ℓ_1 estimation problem
[65D10, 65K05]
(*see:* **Overdetermined systems of linear equations**)
l_1 exact penalty function
[90Cxx]
(*see:* **Discontinuous optimization**)
L-matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
L_1-norm
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
L_2-norm
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
L_∞-norm
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
ℓ_1 penalty function
[90C30]
(*see:* **Nonlinear least squares problems**)
L_∞-penalty function *see:* exact —
L-R flat fuzzy number
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
L-R fuzzy number
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
L-RH-BFGS
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
L-separation theorem
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
L-shaped decomposition
[68W10, 90B15, 90C06, 90C15, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**; **Stochastic programs with recourse: upper bounds**)

l-shaped method
[90C06, 90C15, 90C90]
(*see:* **Decomposition algorithms for the solution of multistage mean-variance optimization problems**; **L-shaped method for two-stage stochastic programs with recourse**; **Stochastic linear programming: decomposition and cutting planes**)
L-shaped method *see:* integer —
L-shaped method for two-stage stochastic programs with recourse
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
L type neighborhood structure for the QAP *see:* K- —
L_{free} of unused partitions *see:* set —
L_{reac} of used partitions *see:* set —
L_1 (v)-differentiable family of measures *see:* weakly —
La Garza method *see:* De —
label *see:* distance —
label carrying
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
label correcting methods
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
label setting methods
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
labeling
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
labeling *see:* consistent —; distance constrained —; integer —; vector —
labeling algorithm *see:* consistent —; relaxation —
labeling procedure
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
labeling processes *see:* relaxation —
labelings
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
lack of smoothness
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
Lagrange duality *see:* Multi-objective optimization: —; saddle —
Lagrange equation
[90C30]
(*see:* **Image space approach to optimization**)
Lagrange equation *see:* dual Euler– —; Euler– —
Lagrange equations
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
Lagrange form
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
Lagrange function
[49K27, 49K40, 90C30, 90C31, 90C34, 90C90]
(*see:* **MINLP: applications in blending and pooling**

problems; **Second order constraint qualifications**; **Semi-infinite programming: second order optimality conditions**)
Lagrange function
[90C05, 90C30]
(*see:* **Image space approach to optimization**; **Theorems of the alternative and optimization**)
Lagrange functional
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
Lagrange functions *see:* augmented —
Lagrange, Joseph-Louis
(01A99)
(*referred to in:* **Decomposition techniques for MILP: lagrangian relaxation**; **Integer programming: lagrangian relaxation**; **Lagrangian multipliers methods for convex programming**; **Multi-objective optimization: lagrange duality**)
(*refers to:* **Decomposition techniques for MILP: lagrangian relaxation**; **Integer programming: lagrangian relaxation**; **Lagrangian multipliers methods for convex programming**; **Multi-objective optimization: lagrange duality**)
Lagrange multiplier
[90C29, 90C30, 90C33]
(*see:* **Implicit lagrangian**; **Multi-objective optimization; Interactive methods for preference value functions**)
Lagrange multiplier approach *see:* Everett generalized —
Lagrange multiplier rule
[49K27, 90C26, 90C29, 90C48]
(*see:* **Set-valued optimization**; **Smooth nonlinear nonconvex optimization**)
Lagrange multiplier rule
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
Lagrange multiplier rule *see:* global —
Lagrange multiplier sets
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
Lagrange multiplier vector
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
Lagrange multipliers
[01A99, 49M37, 65G20, 65G30, 65G40, 65H20, 65K05, 65K10, 90B85, 90C05, 90C10, 90C22, 90C25, 90C30, 90C31, 93A13]
(*see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Image space approach to optimization**; **Integer programming: lagrangian relaxation**; **Interval analysis: verifying feasibility**; **Lagrange, Joseph-Louis**; **Lagrangian multipliers methods for convex programming**; **Multilevel methods for optimal design**; **Semidefinite programming: optimality conditions and stability**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**; **Single facility location: multi-objective euclidean distance location**; **Theorems of the alternative and optimization**)
Lagrange multipliers
[01A99, 90C05, 90C10, 90C15, 90C22, 90C25, 90C30, 90C31]
(*see:* **Image space approach to optimization**; **Integer programming: lagrangian relaxation**; **Lagrange, Joseph-Louis**; **Semidefinite programming: optimality conditions and stability**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Theorems of the alternative and optimization**)
Lagrange multipliers *see:* extended set of —; Stochastic programming: nonanticipativity and —
Lagrange multipliers for nonanticipativity constraints
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
Lagrange multipliers for phase constraints
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
Lagrange relaxation
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
Lagrange–Slater dual *see:* extended —
Lagrange-type functions
(90C30, 90C26, 90C46)
Lagrange-type functions
[90C26, 90C30, 90C46]
(*see:* **Lagrange-type functions**)
Lagrangian
[05C60, 05C69, 05C85, 37B25, 46A20, 49M37, 52A01, 65K05, 68W01, 90C15, 90C20, 90C22, 90C25, 90C27, 90C30, 90C31, 90C34, 90C35, 90C59, 91A22]
(*see:* **Composite nonsmooth optimization**; **Duality for semidefinite programming**; **Heuristics for maximum clique and independent set**; **Inequality-constrained nonlinear optimization**; **Replicator dynamics in combinatorial optimization**; **Semidefinite programming: optimality conditions and stability**; **Semi-infinite programming: second order optimality conditions**; **Sensitivity and stability in NLP: continuity and differential stability**; **Stochastic programming: nonanticipativity and lagrange multipliers**)
Lagrangian *see:* augmented —; Implicit —; modified —; MPEC —; quadratic —; reduced Hessian of a —; restricted implicit —; saddle —; sub —; unconstrained implicit —; vector valued —
Lagrangian bounds
[90C25, 90C26]
(*see:* **Decomposition in global optimization**)
Lagrangian conditions
[90C26]
(*see:* **Invexity and its applications**)
Lagrangian conditions
[90C26]
(*see:* **Invexity and its applications**)
Lagrangian decomposition
[90C30, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**)
Lagrangian decomposition
[90C30, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**)
Lagrangian decomposition approach *see:* augmented —

Lagrangian dual
[90C10, 90C30, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**; **Integer programming: lagrangian relaxation**)
Lagrangian dual
[90C10, 90C30]
(*see:* **Integer programming: lagrangian relaxation**)
Lagrangian dual optimization problem
[90C30]
(*see:* **Lagrangian duality: BASICS**)
Lagrangian dual problem
[90C10, 90C30, 90C46]
(*see:* **Integer programming duality**; **Lagrangian duality: BASICS**)
Lagrangian duality
[90C10, 90C46]
(*see:* **Integer programming duality**)
Lagrangian duality
[90C30]
(*see:* **Duality for semidefinite programming**)
Lagrangian duality: BASICS
(90C30)
(*referred to in:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **Implicit lagrangian**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Farkas lemma**; **Farkas lemma: generalizations**; **First order constraint qualifications**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**)
Lagrangian finite generation method
[90C15]
(*see:* **L-shaped method for two-stage stochastic programs with recourse**)
Lagrangian form
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
Lagrangian function
[90C05, 90C20, 90C25, 90C26, 90C30, 90C39, 90C90]
(*see:* **Image space approach to optimization**; **Lagrangian duality: BASICS**; **Second order optimality conditions for nonlinear optimization**; **Smooth nonlinear nonconvex optimization**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Theorems of the alternative and optimization**)
Lagrangian function
[90C20, 90C30, 90C90]
(*see:* **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: decomposition methods**)
Lagrangian function *see:* augmented —; Hessian matrix of a —; infimum of a —; projected Hessian matrix of a —
Lagrangian Hessian matrix *see:* projected —
Lagrangian methods *see:* Practical augmented —
Lagrangian multipliers
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
Lagrangian multipliers
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
Lagrangian multipliers methods for convex programming
(90C25, 90C30)
(*referred to in:* **Convex max-functions**; **Decomposition techniques for MILP: lagrangian relaxation**; **Integer programming: lagrangian relaxation**; **Lagrange, Joseph-Louis**; **Multi-objective optimization: lagrange duality**; **Splitting method for linear complementarity problems**)
(*refers to:* **Convex max-functions**; **Decomposition techniques for MILP: lagrangian relaxation**; **Integer programming: lagrangian relaxation**; **Lagrange, Joseph-Louis**; **Multi-objective optimization: lagrange duality**)
Lagrangian problems *see:* high-order local maximum principle for —
Lagrangian relaxation
[49J52, 68Q99, 90C10, 90C11, 90C15, 90C27, 90C30, 90C35, 90C46, 90C57, 90C90]
(*see:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Integer programming**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **Multicommodity flow problems**; **Multi-index transportation problems**; **Nondifferentiable optimization: relaxation methods**; **Stochastic integer programs**)
Lagrangian relaxation
[49J52, 90B80, 90C10, 90C30, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**; **Facilities layout problems**; **Integer programming: lagrangian relaxation**; **Nondifferentiable optimization: relaxation methods**)
lagrangian relaxation *see:* Decomposition techniques for MILP: —; Integer programming: —
Lagrangian relaxation with subgradient optimization
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)
Lagrangian theory
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
Lagrangian theory of CNSO problems *see:* second order —
Lagrangians *see:* augmented —
Λ *see:* canonical function associated with —
laminar family
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)

laminated composite materials
[90C26, 90C90]
(*see:* **Structural optimization: history**)
Lanczos algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
Lanczos method *see:* implicit restarted —
Landau notation
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
Langevin equation
[65K05, 90C30]
(*see:* **Random search methods**)
language
[90C60]
(*see:* **Complexity classes in optimization**)
language *see:* algebraic modeling —; algorithmic —; declarative —; F-complete —; F-hard —; modeling —
language accepted by a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
language and constraint logic programming *see:* modeling —
language recognition problem
[90C60]
(*see:* **Complexity classes in optimization**; **Complexity theory**)
language recognition problems
[90C60]
(*see:* **Complexity classes in optimization**)
languages *see:* algebraic modeling —; declarative —; second generation modeling —
languages in optimization: a new paradigm *see:* Modeling —
LAPACK
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
Laplace method and applications to optimization problems
laplace's principle of insufficient reason
[94A17]
(*see:* **Jaynes' maximum entropy principle**)
Laplace principle of insufficient reasoning
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
large *see:* sufficiently —
large collections of documents *see:* classification of —
large nonlinear multicommodity flow problems
[90C30]
(*see:* **Simplicial decomposition**)
large region network
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
large residual
[90C30]
(*see:* **Nonlinear least squares problems**)
large residual problem
[90C30]
(*see:* **Generalized total least squares**)
large-scale combinatorial optimization
(*see:* **Selection of maximally informative genes**)
large-scale least squares problems *see:* Complexity and —
large scale linear systems
[90C30]
(*see:* **Conjugate-gradient methods**)
large-scale neighborhoods
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
large scale nonlinear mixed integer programming problem
[90B05, 90B06]
(*see:* **Global supply chain models**)
large scale optimization
[01A99]
(*see:* **History of optimization**)
large scale problem
[90C06]
(*see:* **Large scale unconstrained optimization**)
large scale problem
[90C06]
(*see:* **Large scale unconstrained optimization**)
large scale and sparse semidefinite programs *see:* Solving —
large scale trust region
[90C30]
(*see:* **Large scale trust region problems**)
large scale trust region problem
[90C30]
(*see:* **Large scale trust region problems**)
Large scale trust region problems
(90C30)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Conjugate-gradient methods**; **Interval linear systems**; **Large scale unconstrained optimization**; **Local attractors for gradient-related descent iterations**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Conjugate-gradient methods**; **D.C. programming**; **Integer programming**; **Interval linear systems**; **Large scale unconstrained optimization**; **Linear programming**; **Lipschitzian operators in best approximation by bounded or continuous functions**; **Local attractors for gradient-related descent iterations**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
Large scale unconstrained optimization
(90C06)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **Broyden family of methods and the BFGS update**; **Cholesky factorization**; **Conjugate-gradient methods**; **Continuous global optimization: models, algorithms and software**; **Interval linear systems**; **Large scale trust region problems**; **Modeling languages in optimization: a new paradigm**; **Optimization software**; **Orthogonal triangularization**; **Overdetermined systems of**

linear equations; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**; **Unconstrained optimization in neural network training**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **Broyden family of methods and the BFGS update**; **Cholesky factorization**; **Conjugate-gradient methods**; **Continuous global optimization: models, algorithms and software**; **Interval linear systems**; **Large scale trust region problems**; **Linear programming**; **Modeling languages in optimization: a new paradigm**; **Nonlinear least squares: trust region methods**; **Optimization software**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**; **Unconstrained optimization in neural network training**)
largest coefficient pivoting rule *see:* Dantzig —
largest coefficient rule
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
largest empty circle
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
largest empty circle problem
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
largest inscribed sphere method
[49M20, 90-08, 90C25]
(*see:* **Nondifferentiable optimization: cutting plane methods**)
largest possible
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
Lasserre signed decomposition
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
last node
[90C35]
(*see:* **Generalized networks**)
last-out rule *see:* first-in —
lattice
[03B50, 03B52, 03C80, 13Cxx, 13Pxx, 14Qxx, 62F30, 62Gxx, 68T27, 90Cxx]
(*see:* **Checklist paradigm semantics for fuzzy logics**; **Integer programming: algebraic methods**)
lattice *see:* distributive —; free distributive —; vector —
lattice ideal
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
lattice program
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
lattice program
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
lattice-type many-valued logic system
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
law *see:* flow conservation —; Gauss distribution —; Hook —; Raoult —; Walras —
law of heat conduction *see:* Fourier —
law of normal distribution
[01A99]
(*see:* **Gauss, Carl Friedrich**)
lawed *see:* dead or dog- —
Lawler linearization
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Lawler lower bound *see:* Gilmore– —
Lawler type lower bounds *see:* Gilmore– —
Lawrence signed decomposition
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
laws *see:* conservation —; discretized hemivariational inequalities for nonlinear material —; variational formulation of quasidifferential —; variational formulation of subdifferential —
laws and hemivariational inequalities *see:* multivalued nonmonotone —
laws and systems of variational inequalities *see:* QD —
laws and variational equalities *see:* single-valued boundary —
laws and variational inequalities *see:* multivalued monotone —
layer *see:* input —
layer feed-forward network *see:* two- —
layer supergraph *see:* three- —
layout *see:* facilities —; facility —
layout manager
[90B80]
(*see:* **Facilities layout problems**)
layout manager
[90B80]
(*see:* **Facilities layout problems**)
layout problem *see:* terminal —
layout problems *see:* Facilities —
layout problems and optimization *see:* Plant —
layover
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
LBDOP
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
lCP
[65K10, 90C11, 90C33, 90C51]
(*see:* **Generalizations of interior point methods for the linear complementarity problem**; **LCP: Pardalos–Rosen mixed integer formulation**)
LCP
[05B35, 65K05, 90C05, 90C20, 90C25, 90C33, 90C55]
(*see:* **Integer linear complementary problem**; **Lexicographic pivoting rules**; **Principal pivoting methods for linear complementarity problems**; **Splitting method for linear complementarity problems**)
LCP *see:* integer —; PCP- —; process the —
LCP: Pardalos–Rosen mixed integer formulation
(90C33, 90C11)

(*referred to in:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **Linear complementarity problem**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Order complementarity**; **Parametric mixed integer nonlinear optimization**; **Principal pivoting methods for linear complementarity problems**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**; **Topological methods in complementarity theory**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Convex-simplex algorithm**; **Decomposition techniques for MILP: lagrangian relaxation**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Order complementarity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Principal pivoting methods for linear complementarity problems**; **Sequential simplex method**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**; **Topological methods in complementarity theory**)

lDL
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)

lDM
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)

LDSU *see:* nonarbitrage condition for —

leader problem
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)

leading boxes
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)

lean stream
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)

learning *see:* batch —; guided —; machine —; off-line —; on-line —; Q- —; reinforcement —; supervised —

learning algorithm
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)

learning algorithm
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)

learning algorithm *see:* machine- —

learning of Boolean functions *see:* interactive —

least absolute deviation
[65K05, 90C27, 90C30, 90C57, 91C15]
(*see:* **Optimization-based visualization**)

least circle
[90B85, 90C27]
(*see:* **Single facility location: circle covering problem**)

least effort *see:* principle of —

least-index
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)

Least-index anticycling rules
(90C05, 90C33, 90C20, 05B35)
(*referred to in:* **Criss-cross pivoting rules**; **Lexicographic pivoting rules**; **Linear programming**; **Linear programming: Klee–Minty examples**; **Pivoting algorithms for linear programming generating two paths**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**; **Simplicial pivoting algorithms for integer programming**)
(*refers to:* **Criss-cross pivoting rules**; **Farkas lemma**; **Farkas lemma: generalizations**; **Lexicographic pivoting rules**; **Linear complementarity problem**; **Linear programming**; **Oriented matroids**; **Pivoting algorithms for linear programming generating two paths**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**)

least-index anticycling rules
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)

least-index criss-cross method
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**; **Principal pivoting methods for linear complementarity problems**)

least-index pivoting method
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)

least index pivoting rule *see:* Bland —

least-index refinement *see:* Murty —

least infeasible integer variable *see:* most/ —

least squares
[33C45, 65F20, 65F22, 65K10, 65T40, 90C26, 90C29, 90C30, 90C90]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**; **Global optimization**

methods for harmonic retrieval; **Least squares orthogonal polynomials**)
least squares
[33C45, 65F20, 65F22, 65Fxx, 65K10, 90C30, 90C52, 90C53, 90C55]
(*see:* **Gauss–Newton method: Least squares, relation to Newton's method**; **Least squares orthogonal polynomials**; **Least squares problems**)
least squares *see:* Abaffi–Broyden–Spedicato algorithms for linear equations and linear —; ABS algorithms for linear equations and linear —; generalized nonlinear —; Generalized total —; linear —; method of —; nonlinear —; weighted —
least squares algorithm *see:* recursive —
least squares criterion
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
least squares distance function
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
least squares formal orthogonal polynomials
[33C45, 65F20, 65F22, 65K10]
(*see:* **Least squares orthogonal polynomials**)
least squares formulation
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
least squares: Newton-type methods *see:* Nonlinear —
Least squares orthogonal polynomials
(33C45, 65K10, 65F20, 65F22)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Least squares problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Least squares problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**)
least squares problem
[15-XX, 49K35, 49M27, 62G07, 62G30, 65-XX, 65D10, 65K05, 65K10, 90-XX, 90C25, 90Cxx]
(*see:* **Cholesky factorization**; **Convex max-functions**; **Isotonic regression problems**; **Overdetermined systems of linear equations**; **Symmetric systems of linear equations**)
least squares problem
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
least squares problem *see:* consistent —; generalized —; generalized nonlinear —; perturbed —; sparse —; total —; unconstrained nonlinear —; weighted —
Least squares problems
(65Fxx)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Cholesky factorization**; **Gauss, Carl Friedrich**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Least squares orthogonal polynomials**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**; **Unconstrained optimization in neural network training**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Gauss, Carl Friedrich**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Least squares orthogonal polynomials**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**)
least squares problems *see:* Complexity and large-scale —; Nonlinear —
least squares problems with massive data sets
[34-xx, 34Bxx, 34Lxx, 93E24]
(*see:* **Complexity and large-scale least squares problems**)
Least squares, relation to Newton's method *see:* Gauss–Newton method: —
least squares solutions
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
least squares: trust region methods *see:* Nonlinear —
leaving variable
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
leaving variable *see:* choice of the —
left-chain justification
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
left-collection of a partition
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
left local maximizer *see:* discrete —
left-paired element
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
left-paired set
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
left-pairs
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
left-reachable
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
left-reachable *see:* directly —
left saddle point
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
left-unpaired element
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
legal neighbor
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
legend *see:* arc —; node —

Legendre conjugate
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
Legendre duality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
Legendre duality pair
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
legendre duality relations
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**; **Duality theory: monoduality in convex optimization**)
Legendre transformation
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
Legendre transformation *see:* integral Fenchel– —
Legends *see:* Figure —
legitimacy *see:* gain —
Lehmann–Goerisch bound
[49R50, 65G20, 65G30, 65G40, 65L15, 65L60]
(*see:* **Eigenvalue enclosures for ordinary differential equations**)
Lehmann–Maehly method
[49R50, 65G20, 65G30, 65G40, 65L15, 65L60]
(*see:* **Eigenvalue enclosures for ordinary differential equations**)
Lehmann–Maehly method
[49R50, 65G20, 65G30, 65G40, 65L15, 65L60]
(*see:* **Eigenvalue enclosures for ordinary differential equations**)
Leibler cross-entropy *see:* Kullback– —
Leibler divergence *see:* Kullback– —
Leibler measure of cross-entropy *see:* Kullback– —
Leibniz *see:* Gottfried Wilhelm —
Leibniz, gottfried wilhelm
(01A99)
(*referred to in:* **History of optimization**)
(*refers to:* **History of optimization**)
Lemke's algorithm
[05B35, 52A22, 60D05, 65K05, 68Q25, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**; **Probabilistic analysis of simplex algorithms**)
Lemke method
(90C33)
(*referred to in:* **Convex-simplex algorithm**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Linear complementarity problem**; **Linear programming**; **Order complementarity**; **Parametric linear programming: cost simplex algorithm**; **Principal pivoting methods for linear complementarity problems**; **Sequential simplex method**; **Topological methods in complementarity theory**)
(*refers to:* **Convex-simplex algorithm**; **Linear complementarity problem**; **Linear programming**; **Parametric linear programming: cost simplex algorithm**; **Sequential simplex method**)
Lemke method
[90C33]
(*see:* **Linear complementarity problem**)
lemma *see:* equivariant Morse —; Farkas —; first slope —; Knaster–Kuratowski–Mazurkiewicz —; second slope —; third slope —
lemma: generalizations *see:* Farkas —
lemmas *see:* slope —
length *see:* binary —; maximin path —; maximum path —; minimax path —; minimum path —; path —; Shortest program —; unary —; variable stage- —
length of input data
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
length of a partial computation of a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
length of a path in a graph
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
length of a subgraph
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
length vector *see:* arc —
Lennard-Jones
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
Lennard–Jones microcluster
[90C26, 90C90]
(*see:* **Global optimization in Lennard–Jones and morse clusters**)
Lennard–Jones and morse clusters *see:* Global optimization in —
Lennard-Jones potential energy function
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**; **Stochastic global optimization: two-phase methods**)
Leonid Vitalyevich *see:* Kantorovich —
Leray–Schauder degree
[90C33]
(*see:* **Topological methods in complementarity theory**)
less-than-truckload
[90C35]
(*see:* **Multicommodity flow problems**)
lessPreferred
(*see:* **Railroad locomotive scheduling**)
level *see:* aspiration —; concordance —; lower- —
level B
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
level BF
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
level BFR
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]

(*see:* **Identification methods for reaction kinetics and transport**)
level crossing minimization *see:* k- —
level feature detection *see:* low- —
level functions *see:* natural —
level hierarchy *see:* k- —
level of interaction
[90B85]
(*see:* **Multifacility and restricted location problems**)
level in a leveled graph
[90C35]
(*see:* **Optimization in leveled graphs**)
level Optimization *see:* Two- —
level planar graph
[90C35]
(*see:* **Optimization in leveled graphs**)
level planarization
[90C35]
(*see:* **Optimization in leveled graphs**)
level planarization problem
[90C35]
(*see:* **Optimization in leveled graphs**)
level planarization problem *see:* k- —
level problem *see:* first —; lower- —; second —; upper —
level set
[62G07, 62G30, 65K05, 90C05, 90C25, 90C26, 90C30, 90C34]
(*see:* **Isotonic regression problems**; **Monotonic optimization**; **Random search methods**; **Semi-infinite programming: discretization methods**)
level set *see:* bounded —
level sets conjugation
[90C26]
(*see:* **Global optimization: envelope representation**)
level software *see:* high- —; low- —; medium- —
level of a vertex in a rooted tree
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
leveled crossing minimization
[90C35]
(*see:* **Optimization in leveled graphs**)
leveled graph
[90C35]
(*see:* **Optimization in leveled graphs**)
leveled graph *see:* k- —; level in a —; proper k- —
leveled graphs
[90C35]
(*see:* **Optimization in leveled graphs**)
leveled graphs *see:* Optimization in —
Levenberg–Marquardt
[90C30]
(*see:* **Cost approximation algorithms**)
Levenberg–Marquardt
[90C30]
(*see:* **Cost approximation algorithms**)
Levenberg–Marquardt algorithm
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
Levenberg–Marquardt method
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
Levenberg–Marquardt rule
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
leverage hypothesis *see:* financial —
Levin theorem *see:* Cook– —
Levitin–Polyak method
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
Levitin–Polyak minimizing sequence
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
Levitin–Polyak well-posed problem
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
Levy probability distribution
[90C90]
(*see:* **Simulated annealing methods in protein folding**)
lexico-positive basis tableau
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
lexico-positive vector
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
lexicographic
(*see:* **Planning in the process industry**)
lexicographic approach
(*see:* **Planning in the process industry**)
lexicographic dual simplex method
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
Lexicographic Goal Programming
(*see:* **Planning in the process industry**)
lexicographic ordering
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
lexicographic ordering
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
lexicographic ordering and perturbation
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
lexicographic pivot selection
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
lexicographic pivoting rule
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
Lexicographic pivoting rules
(90C05, 90C20, 90C33, 05B35, 65K05)
(*referred to in:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Linear programming**; **Linear programming: Klee–Minty examples**; **Pivoting algorithms for linear programming generating two paths**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**; **Simplicial pivoting algorithms for integer programming**)
(*refers to:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Linear complementarity problem**; **Linear**

programming; **Oriented matroids**; **Pivoting algorithms for linear programming generating two paths**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**)
lexicographic primal simplex method
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
lexicographic search algorithm
[90C10, 90C11, 90C20]
(*see:* **Linear ordering problem**)
lexicographic simplex
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
lexicographic simplex method
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
lexicographic variant of the constraint-by-constraint method
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
lexicographical order
[12D10, 12Y05, 13P10]
(*see:* **Gröbner bases for polynomial equations**)
lexicographical ordering for n-dimensional vectors
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
lexicographically greater
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
lexicographically minimax objective function
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
lexicographically positive vector
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
lexicographically smaller
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
LexPr
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
LFS function
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
Li–Pardalos generator
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Liability Management *see:* asset —
liability management decision support system *see:* Asset —
library *see:* general-purpose software —; IMSL subroutine —; NAG —; NAG parallel —; rotamer —
LICQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
(LICQ) *see:* linear independence constraint qualification —
Lie algebra
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
lift-and-project
[90C05, 90C06, 90C08, 90C10, 90C11, 90C26, 90C27, 90C57]
(*see:* **Integer programming**; **Integer programming: branch and cut algorithms**; **Reformulation-linearization technique for global optimization**)
lift-and-project
[90C09, 90C10, 90C11, 90C27, 90C57]
(*see:* **Disjunctive programming**; **Integer programming**; **Set covering, packing and partitioning problems**)
lift-and-project cut
[90C11]
(*see:* **MINLP: branch and bound methods**)
lift-and-project cuts
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
lift-and-project hierarchy
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
lift availability *see:* upper bound on gas —
lift wells of type a *see:* gas —
lift wells of type b *see:* gas —
lifting
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**)
lifting cut
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**)
lifting procedure
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
light edge
[68Q20]
(*see:* **Optimal triangulations**)
light travel
(*see:* **Railroad locomotive scheduling**)
like *see:* convex- —
like criterion *see:* test nonmonotone Armijo- —
like function *see:* convex- —
like function pair *see:* convex- —
like inequalities *see:* variational- —
like method *see:* proximal- —
likelihood *see:* maximum —
likelihood detection via semidefinite programming *see:* Maximum —
likelihood estimate *see:* maximum —
likelihood estimation *see:* maximum —
likelihood evidence
(*see:* **Bayesian networks**)
likelihood function
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
likelihood method *see:* iterative quadratic maximum —
likelihood principle *see:* maximum —
likelihood ratio method
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
limit *see:* absolute —

limited discrepancy search
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
limited enumeration methods
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
limited-memory
[90C30]
(*see:* **Conjugate-gradient methods**)
limited-memory affine reduced Hessian
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
limited-memory algorithm
[90C30]
(*see:* **Conjugate-gradient methods**)
limited-memory approach
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
limited-memory BFGS method
[90C06]
(*see:* **Large scale unconstrained optimization**)
limited-memory reduced-Hessian BFGS algorithm
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
limited-memory symmetric rank-one approach
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
limiting coderivative
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
limiting differential
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
limiting Fréchet subdifferential
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
limiting (Fréchet) subdifferentials
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
limiting normal cone
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
limiting subdifferential *see:* singular —
limiting superdifferential
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
limiting value
[01A99]
(*see:* **Gauss, Carl Friedrich**)
Lin–Kernighan neighborhood
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
LindAcR
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
line *see:* capital market —; inexact line search —; security market —
line algorithm *see:* on- —
line feedback *see:* off- —; on- —
line learning *see:* off- —; on- —
line method *see:* on- —
line process optimization *see:* off- —; on- —
line search
[90C25, 90C30, 90C33]
(*see:* **Conjugate-gradient methods**; **Implicit lagrangian**; **Nonlinear least squares problems**; **Successive quadratic programming: full space methods**)
line search
[90C30]
(*see:* **Nonlinear least squares problems**)
line search *see:* curvilinear —; inexact —; nonmonotone —
line search line *see:* inexact —
line search methods
[90C30]
(*see:* **Cyclic coordinate method**; **Powell method**; **Rosenbrock method**)
line search problem
[90C30]
(*see:* **Convex-simplex algorithm**)
line search problem
[90C30]
(*see:* **Convex-simplex algorithm**; **Frank–Wolfe algorithm**)
line search technique
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
line search technique *see:* inexact —
line searches
[49M37, 90C30]
(*see:* **Nonlinear least squares: trust region methods**; **Rosenbrock method**)
line searching
[90C30]
(*see:* **Successive quadratic programming**)
lineality space
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
linear
[34A55, 34E05, 35R30, 49M20, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30, 90C11, 90C27, 90C30]
(*see:* **Asymptotic properties of random multidimensional assignment problem**; **Generalized outer approximation**; **Global optimization: functional forms**; **Identification methods for reaction kinetics and transport**; **Optimal sensor scheduling**)
linear *see:* non- —
linear algebra
[14R10, 15A03, 32B15, 51E15, 51N20]
(*see:* **Affine sets and functions**; **Linear space**)
linear algebra framework
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
linear algebraic equations
[34-XX, 49-XX, 65-XX, 65K05, 65K10, 68-XX, 90-XX]
(*see:* **ABS algorithms for linear equations and linear least squares**; **Nonlocal sensitivity analysis with automatic differentiation**)

linear algorithm
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
linear appearance of control function
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
linear approximation
[90C31]
(*see:* **Bounds and solution vector estimates for parametric NLPS**)
linear arc cost *see:* piecewise —
linear Arrangement
(*see:* **State of the art in modeling agricultural systems**)
linear arrangement problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
linear bilevel programming problem
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)
linear BLPP *see:* complexity of the —
linear CG method
[90C30]
(*see:* **Conjugate-gradient methods**)
linear CG method
[90C30]
(*see:* **Conjugate-gradient methods**)
linear co-index
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
linear complementarity
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
linear complementarity
[90C33]
(*see:* **Lemke method**)
Linear complementarity problem
(90C33)
(*referred to in:* **Convex-simplex algorithm**; **Criss-cross pivoting rules**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Least-index anticycling rules**; **Lemke method**; **Lexicographic pivoting rules**; **Linear programming**; **Linear programming: interior point methods**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Order complementarity**; **Parametric linear programming: cost simplex algorithm**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**; **Quadratic programming with bound constraints**; **Sequential simplex method**; **Splitting method for linear complementarity problems**; **Topological methods in complementarity theory**)
(*refers to:* **Convex-simplex algorithm**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Lemke method**; **Linear programming**; **Order complementarity**; **Parametric linear programming: cost simplex algorithm**; **Principal pivoting methods for linear complementarity problems**; **Sequential simplex method**; **Splitting method for linear complementarity problems**; **Topological methods in complementarity theory**)
linear complementarity problem
[65K10, 65M60, 90C11, 90C30, 90C33]
(*see:* **Kuhn–Tucker optimality conditions**; **LCP: Pardalos–Rosen mixed integer formulation**; **Lemke method**; **Variational inequalities**)
linear complementarity problem
[90C11, 90C33]
(*see:* **LCP: Pardalos–Rosen mixed integer formulation**)
linear complementarity problem *see:* extended —; Generalizations of interior point methods for the —; horizontal —; mixed —; parametric —; vertical —
linear complementarity problems
[90C31, 90C33]
(*see:* **Sensitivity analysis of complementarity problems**)
linear complementarity problems *see:* Principal pivoting methods for —; Splitting method for —
linear complementary problem
[90C25, 90C33]
(*see:* **Integer linear complementary problem**)
linear complementary problem *see:* Integer —
linear Component
(*see:* **State of the art in modeling agricultural systems**)
linear constraint
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
linear constraints *see:* general —
linear control *see:* piecewise —
linear convergence
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
linear convergence rate *see:* r- —
linear dependence
[90C09, 90C10]
(*see:* **Matroids**; **Oriented matroids**)
linear discrete optimization oracle
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
linear equality relation
[15A39, 90C05]
(*see:* **Linear optimization: theorems of the alternative**)
linear equation
[15A39, 90C05]
(*see:* **Linear optimization: theorems of the alternative**)
linear equation-solving *see:* iterative —
linear equations
[65H10, 65J15, 65K05, 65K10]
(*see:* **ABS algorithms for optimization**; **Contraction-mapping**)
linear equations *see:* bounds for —; Overdetermined systems of —; Symmetric systems of —
linear equations and linear least squares *see:* Abaffi–Broyden–Spedicato algorithms for —; ABS algorithms for —
linear extension theorem *see:* Hahn–Banach —

linear fixed charge
[90C25]
(*see:* **Concave programming**)
linear fractional combinatorial optimization
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
linear fractional combinatorial optimization problem *see:* integral —
linear fractional program
[90C32]
(*see:* **Fractional programming**)
linear-fractional programming
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
linear fractional terms
[90C26, 90C90]
(*see:* **Global optimization of heat exchanger networks**)
linear function *see:* decomposition of a continuous piecewise —; piecewise —
linear functions
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
linear growth condition
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)
linear, increasing
[90B15]
(*see:* **Evacuation networks**)
linear independence
[90C31]
(*see:* **Sensitivity and stability in NLP: continuity and differential stability**)
linear independence constraint qualification
[49K20, 49M99, 65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34, 90C55]
(*see:* **Parametric global optimization: sensitivity**; **Parametric optimization: embeddings, path following and singularities**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
linear independence constraint qualification (LICQ)
[90C31, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions)
linear independence CQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
linear independency constraint qualification
[90C31, 90C34]
(*see:* **Semi-infinite programming: second order optimality conditions**)
linear index
[58E05, 90C30]
(*see:* **Topology of global optimization**)
linear inequalities
[15A39, 90C05]
(*see:* **Motzkin transposition theorem**)
linear inequalities
[52B12, 68Q25]
(*see:* **Fourier–Motzkin elimination method**)
linear inequality relation
[15A39, 90C05]
(*see:* **Linear optimization: theorems of the alternative**)
linear interval equation
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
linear least squares
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
linear least squares *see:* Abaffi–Broyden–Spedicato algorithms for linear equations and —; ABS algorithms for linear equations and —
linear map *see:* adjoint —
linear matrix inequality
[15A15, 90C25, 90C55, 90C90, 93D09]
(*see:* **Robust control**; **Semidefinite programming and determinant maximization**)
linear matrix inequality
[15A15, 90C25, 90C55, 90C90, 93D09]
(*see:* **Robust control**; **Semidefinite programming and determinant maximization**)
linear matroid
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
linear minimum cost network flow problem *see:* piecewise —
linear mixed integer problems
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
linear model
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
linear model *see:* general univariate —
linear multiple-choice knapsack problem
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
linear multiplicative program
[90C26]
(*see:* **Global optimization in multiplicative programming**)
linear network flow problem
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
linear network flow problems *see:* Piecewise —
linear nondegeneracy condition
[58E05, 90C30]
(*see:* **Topology of global optimization**)
linear optimization
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
linear optimization
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
linear optimization *see:* Distance dependent protein force field via —; duality theorem for —; Global pairwise protein sequence alignment via mixed-integer —
linear optimization problem
[15A39, 90C05]
(*see:* **Linear optimization: theorems of the alternative**; **Motzkin transposition theorem**)

linear optimization problem in standard form
[15A39, 90C05]
(*see:* **Linear optimization: theorems of the alternative**)
linear optimization problems *see:* Price of robustness for —; stochastic —
Linear optimization: theorems of the alternative
(15A39, 90C05)
(*referred to in:* **Farkas lemma**; **Linear programming**; **Motzkin transposition theorem**; **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**)
(*refers to:* **Farkas lemma**; **Linear programming**; **Motzkin transposition theorem**; **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**)
linear order complementarity problem
[90C33]
(*see:* **Order complementarity**)
linear order complementarity problem *see:* generalized —
linear ordering
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
Linear ordering problem
(90C10, 90C11, 90C20)
(*referred to in:* **Quadratic assignment problem**)
(*refers to:* **Assignment and matching**; **Assignment methods in clustering**; **Bi-objective assignment problem**; **Communication network assignment problem**; **Complexity theory: quadratic programming**; **Feedback set problems**; **Frequency assignment problem**; **Generalized assignment problem**; **Graph coloring**; **Graph planarization**; **Greedy randomized adaptive search procedures**; **Maximum partition matching**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Quadratic semi-assignment problem**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
linear ordering problem
[90C35]
(*see:* **Optimization in leveled graphs**)
linear outer approximation
[49M20, 90C11, 90C30]
(*see:* **Generalized outer approximation**)
linear path
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
linear platform cost
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
linear potential
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
linear problem *see:* ball-constrained —; geometrically —; physically —
linear problems *see:* Semi-infinite programming: methods for —
linear program
[68Q25, 90C05, 90C20, 90C30, 91B28]
(*see:* **Competitive ratio for portfolio management**; **Redundancy in nonlinear programs**; **Simplicial decomposition**)
linear program
[90C30]
(*see:* **Convex-simplex algorithm**; **Frank–Wolfe algorithm**; **Simplicial decomposition**)
linear program *see:* dual —; finite-dimensional —; integer —; single parametric mixed integer —; two-stage stochastic —
linear program with an additional reverse convex constraint
[90C26, 90C30]
(*see:* **Reverse convex optimization**)
linear program with recourse *see:* stochastic —
Linear programming
(90C05)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **Affine sets and functions**; **Carathéodory theorem**; **Cholesky factorization**; **Convex-simplex algorithm**; **Criss-cross pivoting rules**; **Ellipsoid method**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Farkas lemma**; **Fourier–Motzkin elimination method**; **Gauss, Carl Friedrich**; **Generalized nonlinear complementarity problem**; **Global optimization in multiplicative programming**; **History of optimization**; **Integer linear complementary problem**; **Interval linear systems**; **Job-shop scheduling problem**; **Kantorovich, Leonid Vitalyevich**; **Krein–Milman theorem**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **LCP: Pardalos–Rosen mixed integer formulation**; **Least-index anticycling rules**; **Lemke method**; **Lexicographic pivoting rules**; **Linear complementarity problem**; **Linear optimization: theorems of the alternative**; **Linear programming: Klee–Minty examples**; **Linear programming models for classification**; **Linear space**; **Motzkin transposition theorem**; **Multiparametric linear programming**; **Multiplicative programming**; **Optimization in medical imaging**; **Order complementarity**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **Parametric linear programming: cost simplex algorithm**; **Pivoting algorithms for linear programming generating two paths**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**; **QR factorization**; **Sequential simplex method**; **Simplicial pivoting algorithms for integer programming**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**; **Topological methods in complementarity theory**; **Tucker homogeneous systems of linear relations**; **Young programming**)
(*refers to:* **Affine sets and functions**; **Carathéodory theorem**; **Convex-simplex algorithm**; **Criss-cross pivoting rules**; **Farkas lemma**; **Gauss, Carl Friedrich**; **Global optimization in multiplicative programming**; **History of optimization**; **Kantorovich, Leonid Vitalyevich**; **Krein–Milman theorem**; **Least-index anticycling rules**; **Lemke method**; **Lexicographic pivoting rules**; **Linear complementarity problem**; **Linear optimization: theorems of the alternative**; **Linear space**; **Motzkin transposition theorem**; **Multiparametric linear programming**; **Multiplicative programming**; **Parametric linear programming: cost simplex algorithm**; **Pivoting algorithms for linear programming generating two paths**;

Principal pivoting methods for linear complementarity problems; **Probabilistic analysis of simplex algorithms**; **Sequential simplex method**; **Simplicial pivoting algorithms for integer programming**; **Tucker homogeneous systems of linear relations**)

linear programming
[01A99, 37A35, 65K05, 68Q05, 68Q10, 68Q20, 68Q25, 68Q99, 90-XX, 90B50, 90C05, 90C06, 90C10, 90C11, 90C15, 90C22, 90C25, 90C26, 90C29, 90C30, 90C57, 90C60, 90C90]
(*see:* **Branch and price: Integer programming with column generation**; **Complexity theory**; **Complexity theory: quadratic programming**; **Copositive programming**; **Information-based complexity and information-based optimization**; **Kantorovich, Leonid Vitalyevich**; **Lagrangian duality: BASICS**; **Modeling difficult optimization problems**; **Optimal triangulations**; **Optimization and decision support systems**; **Potential reduction methods for linear programming**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Probabilistic constrained linear programming: duality theory**; **Suboptimal control**; **Survivable networks**)

linear programming
[01A99, 37A35, 49-XX, 52A22, 52B12, 60D05, 65K05, 65K10, 68Q05, 68Q10, 68Q25, 90-XX, 90B30, 90B50, 90B85, 90C05, 90C06, 90C10, 90C11, 90C15, 90C20, 90C25, 90C26, 90C27, 90C29, 90C30, 90C31, 90C35, 91A99, 91B28, 91B82, 93-XX]
(*see:* **ABS algorithms for optimization**; **Auction algorithms**; **Data envelopment analysis**; **Duality theory: monoduality in convex optimization**; **Fourier–Motzkin elimination method**; **Homogeneous selfdual methods for linear programming**; **Information-based complexity and information-based optimization**; **Kantorovich, Leonid Vitalyevich**; **Linear ordering problem**; **Linear programming**; **Linear programming: karmarkar projective algorithm**; **Linear programming: Klee–Minty examples**; **Multifacility and restricted location problems**; **Multi-objective integer linear programming**; **Operations research and financial markets**; **Optimization and decision support systems**; **Portfolio selection: markowitz mean-variance model**; **Potential reduction methods for linear programming**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Probabilistic analysis of simplex algorithms**; **Probabilistic constrained linear programming: duality theory**; **Selfdual parametric method for linear programs**; **Sensitivity and stability in NLP**)

linear programming *see:* analytical approximation of —; Bilevel —; binary —; Decomposition principle of —; Extension of the fundamental theorem of —; fractional —; Fuzzy multi-objective —; Homogeneous selfdual methods for —; infinite-dimensional —; integer —; mixed integer —; multi-objective —; Multi-objective integer —; Multiparametric —; Multiparametric mixed integer —; multiple objective —; parametric —; Piecewise —; Potential reduction methods for —; probabilistic constrained —; semi-infinite —; stochastic —

linear programming approach for DNA transcription element identification *see:* Mixed 0-1 —

linear programming: complexity, equivalence to minmax, concave programs *see:* Bilevel —

linear programming: cost simplex algorithm *see:* Parametric —

linear programming: decomposition and cutting planes *see:* Stochastic —

linear programming duality
[90C10, 90C46]
(*see:* **Integer programming duality**)

linear programming: duality theory *see:* Probabilistic constrained —

linear Programming and Economic Analysis
[35B40, 37C70, 40A05, 49J24]
(*see:* **Statistical convergence and turnpike theory**; **Turnpike theory: stability of optimal trajectories**)

linear programming with fuzzy coefficients *see:* multi-objective —

linear programming generating two paths *see:* Pivoting algorithms for —

linear programming: heat exchanger network synthesis *see:* Mixed integer —

Linear programming: interior point methods
(90C05)
(*referred to in:* **Ellipsoid method**; **Entropy optimization: interior point methods**; **Homogeneous selfdual methods for linear programming**; **Integer programming: branch and bound methods**; **Linear programming: karmarkar projective algorithm**; **Potential reduction methods for linear programming**; **Principal pivoting methods for linear complementarity problems**; **Quadratic assignment problem**; **Quadratic programming with bound constraints**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Entropy optimization: interior point methods**; **Homogeneous selfdual methods for linear programming**; **Interior point methods for semidefinite programming**; **Linear complementarity problem**; **Linear programming: karmarkar projective algorithm**; **Potential reduction methods for linear programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: solution by active sets and interior point methods**)

Linear programming: karmarkar projective algorithm
(90C05)
(*referred to in:* **Ellipsoid method**; **Entropy optimization: interior point methods**; **Homogeneous selfdual methods for linear programming**; **Linear programming: interior point methods**; **Nondifferentiable optimization: cutting plane methods**; **Potential reduction methods for linear programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Entropy optimization: interior point methods**; **Homogeneous selfdual methods for linear programming**; **Interior point methods for semidefinite programming**; **Linear programming: interior point methods**; **Potential reduction methods for linear programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: solution by active sets and interior point methods**)

Linear programming: Klee–Minty examples
(90C05)
(*referred to in:* **Ellipsoid method**)
(*refers to:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Linear programming**)
linear programming: mass and heat exchanger networks *see:* Mixed integer —
Linear programming models for classification
(62H30, 68T10, 90C05)
(*referred to in:* **Mixed integer classification problems**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**; **Statistical classification: optimization approaches**)
(*refers to:* **Deterministic and probabilistic optimization models for data classification**; **Linear programming**; **Mixed integer classification problems**; **Statistical classification: optimization approaches**)
linear programming problem
[49L20, 52B12, 60G35, 65K05, 68Q25, 90C39, 90C90]
(*see:* **Design optimization in computational fluid dynamics**; **Differential equations and global optimization**; **Dynamic programming: discounted problems**; **Fourier–Motzkin elimination method**)
linear programming problem *see:* analytical approximation of a —
linear programming problems *see:* extended —; Stabilization of cutting plane algorithms for stochastic —
linear programming program
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
linear programming relaxation
[90C05, 90C06, 90C08, 90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Set covering, packing and partitioning problems**)
linear programming relaxations
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
linear programming with right-hand-side uncertainty, duality and applications *see:* Robust —
linear programming under uncertainty *see:* multi-objective —
linear programming with variable coefficients *see:* generalized —
linear programs *see:* dual —; Robust optimization: mixed-integer —; Selfdual parametric method for —
linear programs with recourse and arbitrary multivariate distributions *see:* Stochastic —
linear programs for routing and protection problems in optical networks *see:* Integer —
linear quadratic function *see:* piecewise —
linear-quadratic Gaussian
[93D09]
(*see:* **Robust control**)
linear-quadratic problem
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)
linear regression model *see:* classical —
linear relations
[15A39, 90C05]
(*see:* **Linear optimization: theorems of the alternative**)
linear relations *see:* Tucker homogeneous systems of —
linear relaxation
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
linear scales
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
(linear) semi-infinite program *see:* primal —
linear semi-infinite programming
[90C05, 90C34]
(*see:* **Semi-infinite programming: methods for linear problems**)
linear semi-infinite programming *see:* perfect duality from the view of —
linear semidefinite program
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
linear semidefinite programming problem
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
linear SIP
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
linear SIP problem *see:* duality of the —
linear SIP problems
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
Linear space
(15A03, 14R10, 51N20)
(*referred to in:* **Affine sets and functions**; **Linear programming**)
(*refers to:* **Affine sets and functions**; **Linear programming**)
linear space
[14R10, 15A03, 51N20]
(*see:* **Linear space**)
linear spaces *see:* Best approximation in ordered normed —
linear speedup
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
linear support
[90C30]
(*see:* **Lagrangian duality: BASICS**)
linear supporting function
[90C30]
(*see:* **Lagrangian duality: BASICS**)
linear system *see:* alternative —; center of an interval —; interval —; sign-solvable —
linear systems *see:* Interval —; large scale —
linear systems of equations
[15A99, 65G20, 65G30, 65G40, 90C26]
(*see:* **Interval linear systems**)
linear thermoelastic behavior of a generally nonhomogeneous and nonisotropic body
[35R70, 47S40, 74B99, 74D99, 74G99, 74H99]

(*see:* **Quasidifferentiable optimization: applications to thermoelasticity**)
linear topological space
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
linear transformation
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
linear two-stage model
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)
linear unidimensional scale
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
linear upper bound *see:* piecewise —
linear zero-one integer problem
[90C25, 90C33]
(*see:* **Integer linear complementary problem**)
linearity *see:* degree of —
linearization
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
linearization *see:* Adams–Johnson —; Frieze–Yadegar —; improved piecewise —; inner —; Kaufman–Broeckx —; Lawler —; partial —
linearization cone *see:* inner —; outer —
linearization of constraints
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
Linearization/Convexification Technique *see:* reformulation- —
linearization/convexification techniques *see:* reformulation- —
linearization error
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
linearization in facility location problems with staircase costs
[90B80, 90C11]
(*see:* **Facility location with staircase costs**)
linearization in facility location problems with staircase costs *see:* convex piecewise —
linearization methods
[90C30]
(*see:* **Cost approximation algorithms**)
linearization methods
[90C30]
(*see:* **Cost approximation algorithms**)
linearization of programs
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
linearization/restriction *see:* inner —
linearization technique *see:* reformulation- —
linearization technique for global optimization *see:* Reformulation- —
linearized reformulation
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
linearly constrained optimization problems
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
linearly dependent *see:* positively —
linearly elastic mechanical constructions
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
linearly elastic systems
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
linearly independent
[90C30]
(*see:* **Convex-simplex algorithm**)
linearly independent *see:* positively —
linearly monotonic over *see:* strongly —
lines *see:* connection of flow —; method of —
linguistic choices
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
linguistics *see:* computational —
link cost function *see:* regular —
link-diverse/disjoint
[46N10, 68M10, 90B18, 90B25]
(*see:* **Integer linear programs for routing and protection problems in optical networks**)
link flow formulation
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
link frequency assignment problem *see:* radio —
link loads *see:* variational inequality formulation in —
linkage *see:* multilevel single- —; simple —
linking constraints *see:* multistage —
linkpoint of a graph
[90C35]
(*see:* **Feedback set problems**)
links *see:* temporal —
LINPACK
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
Liouville theorem
[01A50, 01A55, 01A60]
(*see:* **Fundamental theorem of algebra**)
Lipschitz
[90C11, 90C15]
(*see:* **Stochastic programming with simple integer recourse**)
Lipschitz *see:* locally —
Lipschitz constant
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
Lipschitz continuity
[65K10, 65M60, 90C11, 90C15, 90C31, 90C90]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**)
Lipschitz continuity
[65K05, 65K10, 65M60, 90C30]
(*see:* **Bisection global optimization methods**; **Variational**

inequalities: geometric interpretation, existence and uniqueness)
Lipschitz continuous
[49J20, 49J52, 65K05, 65K10, 65M60, 90C30, 90C31, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**; **Bisection global optimization methods**; **Sensitivity and stability in NLP: approximation**; **Shape optimization**; **Variational inequalities: geometric interpretation, existence and uniqueness**)
Lipschitz continuous function
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)
Lipschitz continuous function *see:* locally —
Lipschitz function
[90C26]
(*see:* **Generalized monotone multivalued maps**; **Global optimization using space filling**)
Lipschitz function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
Lipschitz function *see:* locally —
Lipschitz optimization
[65G20, 65G30, 65G40, 65H20, 65K05, 90C26, 90C30]
(*see:* **Interval analysis: unconstrained and constrained optimization**; **Monotonic optimization**)
Lipschitz optimization
[90C26]
(*see:* **Global optimization using space filling**)
Lipschitz programming
[90C26]
(*see:* **Global optimization: envelope representation**)
Lipschitz programming
[90C26]
(*see:* **Global optimization: envelope representation**)
Lipschitz stability
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
Lipschitz stable solution
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
Lipschitzian operators in best approximation by bounded or continuous functions
(65K10, 41A30, 47A99)
(*referred to in:* **Large scale trust region problems**)
(*refers to:* **Convex envelopes in optimization problems**)
lipschitzian selection operator
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
Lipschitzian selection operator *see:* optimal —
Lipschitzness *see:* compact epi- —
liquid phases
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
list *see:* candidate —; closed —; code —; extended doubly connected edge —; open —; prediction —; restricted Candidate —; running —; tabu —
list coloring
[05-XX]
(*see:* **Frequency assignment problem**)
list size
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
list square merit function
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
literal
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
literal (in logic)
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
Littlewood–Pólya theorem *see:* Hardy– —
LJ optimization procedure
[93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**)
LMI
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
LMM
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
lMT-skeleton
[68Q20]
(*see:* **Optimal triangulations**)
load
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
load *see:* splitting/unsplitting of —
load balancing
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
load balancing *see:* dynamic —; static —
Load balancing for parallel optimization techniques
(68W10, 90C27)
(*referred to in:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: parallel computation**; **Heuristic search**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Parallel heuristic search**; **Stochastic network problems: massively parallel solution**)
(*refers to:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: parallel computation**; **Heuristic search**; **Interval analysis: parallel methods for global optimization**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Parallel heuristic search**; **Stochastic network problems: massively parallel solution**)
load balancing scheme *see:* near-neighbor —
load balancing technique *see:* dynamic —
load curve
[90C10, 90C30, 90C35]

(*see:* **Optimization in operation of electric and energy power systems**)
load curve
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
load dispatcher
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
load dispatcher
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
load dispatching
[90B50]
(*see:* **Optimization and decision support systems**)
load dispatching
[90B50]
(*see:* **Optimization and decision support systems**)
loadbalance
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)
loads *see:* set of —; variational inequality formulation in link —
local
[65H20]
(*see:* **Multi-scale global optimization using terrain/funneling methods**)
local approximations *see:* nonsmooth —
local-area computer network
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
local attractor
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
local attractors *see:* singular —
Local attractors for gradient-related descent iterations
(49M29, 65K10, 90C06)
(*referred to in:* **Conjugate-gradient methods**; **Large scale trust region problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**)
(*refers to:* **Conjugate-gradient methods**; **Large scale trust region problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**)
local basin
[65K05, 90C26, 90C30, 90C59]
(*see:* **Global optimization: filled function methods**)
local bilinear form *see:* K- —
local consistency
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
local convergence rate
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
local convergence rate
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
local cut
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**)
local dualization *see:* Ioffe–Burke —
local efficiency
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
local efficient point
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
local Ekeland point
[58C20, 58E30, 90C46, 90C48]
(*see:* **Nonsmooth analysis: weak stationarity**)
local equivalence closure of a relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
local equivalence relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
local exchange procedures
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
local greedy algorithms
[05C85]
(*see:* **Directed tree networks**)
local improvement
[68Q20]
(*see:* **Optimal triangulations**)
local independence
(*see:* **Bayesian networks**)
local infimum
[90Cxx]
(*see:* **Discontinuous optimization**)
local inner product *see:* **K**- —
local maximizer
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
local maximizer *see:* discrete left —; strict —
local maximizers *see:* set of discrete ε-global —
local maximum
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
local maximum point
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
local maximum point *see:* strict —
local maximum principle
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
local maximum principle
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
local maximum principle *see:* high-order —
local maximum principle for Lagrangian problems *see:* high-order —

local minima
[92B05]
(*see:* **Genetic algorithms**)
local minimization
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
local minimization
[90C60]
(*see:* **Complexity theory: quadratic programming**)
local minimizer
[65K05, 90C26, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Smooth nonlinear nonconvex optimization**)
local minimizer *see:* isolated —; nonsingular —; regular —; strict —; strong —
local minimizer problem
[90C60]
(*see:* **Complexity theory: quadratic programming**)
local minimizers
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
local minimum
[9008, 90C08, 90C11, 90C26, 90C27, 90C39, 90C57, 90C59]
(*see:* **Quadratic assignment problem**; **Second order optimality conditions for nonlinear optimization**; **Variable neighborhood search methods**)
local minimum
[90C26, 90C39, 92B05]
(*see:* **Genetic algorithms**; **Second order optimality conditions for nonlinear optimization**)
local minimum *see:* strict —; strong —
local minimum condition *see:* high-order —
local minimum point
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
local minimum point *see:* strict —
local MINLP
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
local monotonicity
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
local optima
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
local optimization
[90C30, 90C60, 90C90]
(*see:* **Complexity theory**; **MINLP: applications in blending and pooling problems**)
local optimizer *see:* strict —
local optimum
[03B05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C26, 90C27, 90C30, 94C10]
(*see:* **Maximum satisfiability problem**; **Stochastic global optimization: two-phase methods**)
local order relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
local phase
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: two-phase methods**)
local pre-order closure of a relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
local properties of the configuration space
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
local quadratic convergence theorem
[90C30]
(*see:* **Numerical methods for unary optimization**)
local-ratio principle
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
local relational properties
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
local search
[65K05, 9008, 90B35, 90C10, 90C26, 90C27, 90C30, 90C59, 94C15]
(*see:* **Global optimization: filled function methods**; **Graph planarization**; **Job-shop scheduling problem**; **Maximum constraint satisfaction: relaxations and upper bounds**; **Variable neighborhood search methods**)
local search
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90-XX, 90C09, 90C27, 90C35, 94C10]
(*see:* **Feedback set problems**; **Maximum satisfiability problem**; **Survivable networks**)
local search *see:* chained —; iterated —; nonoblivious —; stochastic —
local search algorithms
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
local search device
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
local search heuristics
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
local search method
[90-XX]
(*see:* **Survivable networks**)
local search phase in GRASP
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
local search problems *see:* polynomial time —
local solution
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
local solutions
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)

local strict efficiency
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
local strict monotonicity
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
local strictly efficient point
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
local strong monotonicity
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
local system cohomology
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
local tolerance closure of a relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
local tolerance relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
local underestimator
[90C11, 90C26]
(*see:* **Extended cutting plane algorithm**)
local weak efficiency
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
local weakly efficient point
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
locality constraint
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
locality measure *see:* subgradient —
localization
[65K05, 90C30]
(*see:* **Random search methods**)
localization of an ideal
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
localization problem *see:* Sensor network —
localization problem, SNLP *see:* Semidefinite programming and the sensor network —
localization property
[90C33]
(*see:* **Topological methods in complementarity theory**)
localization search
[65K05, 90C30]
(*see:* **Random search methods**)
localization search *see:* pure —
localization set
[49M20, 90-08, 90C25]
(*see:* **Nondifferentiable optimization: cutting plane methods**)
locally
[68T20, 68T99, 90C27, 90C31, 90C34, 90C59]
(*see:* **Metaheuristics**; **Parametric global optimization: sensitivity**)
locally consistent
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
locally extremal
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
locally filled function
[65K05, 90C26, 90C30, 90C59]
(*see:* **Global optimization: filled function methods**)
locally Lipschitz
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
locally Lipschitz continuous function
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)
locally Lipschitz function
[49J52, 65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Hemivariational inequalities: eigenvalue problems**; **Interval global optimization**)
locally minimal
[68Q20]
(*see:* **Optimal triangulations**)
locally monotone function
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
locally optimal
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
locally optimal parameter
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
locally optimal solution
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
locally reduced problem
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
locally reflexive relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
locally strictly monotone function
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
locally strongly monotone function
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
locally strongly monotonic at
[90C05]
(*see:* **Extension of the fundamental theorem of linear programming**)
location
[49M07, 49M10, 65K, 90B80, 90C06, 90C20]
(*see:* **Facilities layout problems**; **Spectral projected gradient methods**)

location
[90B80, 90C10, 90C11]
(*see:* **Facilities layout problems**; **Facility location problems with spatial interaction**; **Stochastic transportation and location problems**)
location *see:* center of gravity —; Competitive facility —; continuous —; dynamic facility —; emergency facility —; facility —; median —; multi-objective facility —; multifacilities —; multifacility —; multiple-facility —; single facility —; Single facility location: multi-objective euclidean distance —; Single facility location: multi-objective rectilinear distance —; Voronoi diagrams in facility —
location-allocation *see:* facility —; median —; MINLP: application in facility —; multifacility —
location-allocation model
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
location-allocation problem
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
location-allocation problem *see:* p-median —
location and assignment *see:* discrete —
location: circle covering problem *see:* Single facility —
location: covering problems *see:* Network —
location with euclidean and rectilinear distances *see:* Optimizing facility —
location with externalities *see:* Facility —
location model *see:* facility —; plant —; spatial competition facility —; Stochastic facility —
location: multi-objective euclidean distance location *see:* Single facility —
location: multi-objective rectilinear distance location *see:* Single facility —
location pattern
[90B80, 90B85]
(*see:* **Warehouse location problem**)
location problem
[90B80, 90B85]
(*see:* **Warehouse location problem**)
location problem *see:* discrete —; dynamic —; Euclidean distance —; facility —; iterative solution of the Euclidean distance —; maximum coverage —; objective for a —; prototype —; rectilinear distance —; restricted —; simple plant —; squared Euclidean distance —; stochastic transportation and —; uncapacitated facility —; uncapacitated plant —; warehouse —
location problems *see:* barrier —; facility —; Global optimization in —; Multifacility and restricted —; Stochastic transportation and —
location problems with spatial interaction *see:* Facility —
location problems with staircase costs *see:* convex piecewise linearization in facility —; heuristics of facility —; linearization in facility —; solution of facility —
location-routing
[90-02, 90B06]
(*see:* **Operations research models for supply chain management and design**; **Vehicle routing**)
location-routing models
[90-02]
(*see:* **Operations research models for supply chain management and design**)
Location routing problem
(90B06, 90B80)
location on a sphere *see:* minimax —
location with staircase costs *see:* Facility —
location theory
[90B85]
(*see:* **Multifacility and restricted location problems**)
locational decision problem
[90B80, 90B85]
(*see:* **Warehouse location problem**)
locomotive assignment models
(*see:* **Railroad locomotive scheduling**)
Locomotive scheduling
(*see:* **Railroad locomotive scheduling**)
locomotive scheduling *see:* Railroad —
locomotive scheduling models *see:* single —
locomotive type *see:* single —
locomotive type models *see:* multiple —
Loeb measure
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
logarithmic barrier function
[90C05]
(*see:* **Linear programming: interior point methods**)
logarithmic barrier method *see:* interior point —
logarithmic form
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
logarithmic p-form
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
logarithmic-quadratic barrier-penalty function
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
logarithmic scoring rule
(*see:* **Bayesian networks**)
logarithmic and square-root transformation
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
logarithmic volume
[37A35, 90C05]
(*see:* **Potential reduction methods for linear programming**)
logconcave discrete probability distribution
[90C15]
(*see:* **Logconcavity of discrete distributions**)
logconcave distributions *see:* discrete —
logconcave function
[90C15]
(*see:* **Logconcave measures, logconvexity**)
logconcave function
[90C15]
(*see:* **Logconcave measures, logconvexity**; **Probabilistic constrained problems: convexity theory**)
logconcave measure
[90C15]
(*see:* **Logconcave measures, logconvexity**)
Logconcave measures, logconvexity
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic**

programs: descent directions and efficient points; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

logconcave probability density function
[90C15]
(*see:* **Probabilistic constrained problems: convexity theory**)

logconcave probability measure
[90C15]
(*see:* **Logconcave measures, logconvexity**)

logconcave univariate discrete probability distribution
[90C15]
(*see:* **Logconcavity of discrete distributions**)

Logconcavity of discrete distributions
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **L-shaped method for two-stage stochastic programs with recourse**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic**

programming: quasigradient method; **Two-stage stochastic programs with recourse**)
logconvex function
[90C15]
(*see:* **Logconcave measures, logconvexity**)
logconvex function
[90C15]
(*see:* **Logconcave measures, logconvexity**)
logconvex measure
[90C15]
(*see:* **Logconcave measures, logconvexity**)
logconvex probability measure
[90C15]
(*see:* **Logconcave measures, logconvexity**)
logconvexity *see:* Logconcave measures —
logic *see:* BL- —; evaluation in classical —; evaluation in multiple-valued —; fuzzy —; interval —; literal (in —
logic algebra *see:* Boolean 2-valued —; many-valued —
logic algebra connective
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
logic algebras *see:* Finite complete systems of many-valued —; many-valued families of the Pinkava —; PI- —; taxonomy of Pi- —
logic of approximation
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
logic-based approaches
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
logic-based methods *see:* MINLP: —
Logic-based outer approximation
Logic-based outer-approximation method
(*see:* **Optimal planning of offshore oilfield infrastructure**)
logic conditional
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
logic connectives
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
logic connectives *see:* emergence of —
logic gates
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
logic implication *see:* many-valued —
logic normal form *see:* complete many-valued —
logic programming
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
logic programming *see:* constrained —; constraint —; modeling language and constraint —; paradigm of —
logic of Scientific Discovery
[34-xx, 34Bxx, 34Lxx, 93E24]
(*see:* **Complexity and large-scale least squares problems**)
logic system *see:* lattice-type many-valued —
logic system of approximate reasoning *see:* interval —; point-based —
logical algebra *see:* Pinkava —
logical clause
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
logical connectives *see:* TOP and BOT types of —
logical design
[01A99, 90C99]
(*see:* **Von Neumann, John**)
logical equivalence
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
logical implications
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
logics *see:* Checklist paradigm semantics for fuzzy —; classification of many-valued —; many-valued —; taxonomy of the PI-algebras of many-valued —
logistics
[90B05, 90B06]
(*see:* **Global supply chain models**)
logistics
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
logistics control model
[90-02]
(*see:* **Operations research models for supply chain management and design**)
logistics flows
[90-02]
(*see:* **Operations research models for supply chain management and design**)
logistics management
[90-02]
(*see:* **Operations research models for supply chain management and design**)
LogP model
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
logspace Turing machine
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
long campaigns
(*see:* **Planning in the process industry**)
long range planning
[90C90]
(*see:* **Chemical process planning**)
long serious step
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
long-term memory in GRASP
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
look-ahead rules
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**)
lookahead-unit-resolution *see:* single- —
lookup table representation
[90C39]
(*see:* **Neuro-dynamic programming**)
loop
[90C09, 90C10]
(*see:* **Matroids**)
loop control *see:* closed- —; open- —
loop Nash equilibrium *see:* open- —

loop structure prediction methods *see:* Protein —
loops *see:* nested —
LOP
[90C10, 90C11, 90C20]
(*see:* **Linear ordering problem**)
loss *see:* trim —
loss of descent
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
loss of descent in a nonlinear program
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
loss problem *see:* MINLP: trim- —; numerical example of a trim- —; trim- —
losses *see:* minimization of —
lost sales assumption
[49L20]
(*see:* **Dynamic programming: inventory control**)
lot sizing
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
lot-sizing problem *see:* capacitated —; Economic —
Louis *see:* Lagrange, Joseph- —
Lovász extension
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
Lovász number
(05C69, 05C15, 05C17, 05C35, 90C35, 90C22)
(*referred to in:* **Stable set problem: branch & cut algorithms**)
(*refers to:* **Copositive programming**)
lovász number
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
low failure of the alpha-beta algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
low-level feature detection
[90C90]
(*see:* **Optimization in medical imaging**)
low-level software
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
low-rank nonconvexity
[90C26, 90C31]
(*see:* **Global optimization in multiplicative programming**; **Multiplicative programming**)
low-rank nonconvexity
[90C26, 90C31]
(*see:* **Global optimization in multiplicative programming**; **Multiplicative programming**)
lower bandwidth
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
lower bound
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
lower bound *see:* Gilmore–Lawler —; guaranteed —; Jensen —; parametric —; valid —
lower bound function
[90C15, 90C27]
(*see:* **Discrete stochastic optimization**)
lower bound for a set
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
lower bound test
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
lower boundary
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
lower boundary point
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
lower bounding
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)
lower bounding Hessian
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)
lower bounds
[05C85, 90B35]
(*see:* **Directed tree networks**; **Job-shop scheduling problem**)
lower bounds *see:* constructive —; eigenvalue based —; Gilmore–Lawler type —; maximum flow problem with nonnegative —; parametric upper and —; variance reduction —
lower bounds to eigenvalues *see:* upper and —
lower bounds for multivariate probability integrals
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
lower convex hull
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
lower derivative *see:* Dini —; Dini conditional —; Hadamard conditional —
lower directional derivative *see:* Dini —; Hadamard —
lower envelope
[90C30]
(*see:* **Lagrangian duality: BASICS**)
lower-level
[49M30, 49M37, 65K05, 90C30]
(*see:* **Practical augmented Lagrangian methods**)
lower-level problem
[57R12, 90C31, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions; **Parametric global optimization: sensitivity**; **Smoothing methods for semi-infinite optimization**)
lower problem
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
lower semicontinuous
[46A22, 49J20, 49J35, 49J40, 49J52, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**; **Shape optimization**)

lower semicontinuous
[90C11, 90C15]
(*see:* **Stochastic programming with simple integer recourse**)
lower semicontinuous function
[03H10, 49J27, 90C26, 90C34]
(*see:* **Convex envelopes in optimization problems**; **Semi-infinite programming and control problems**)
lower set
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
lower set algorithm *see:* minimum —
lower sets *see:* minimum —
lower and upper bounds constraints
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
lower and upper directional derivatives
[90C31, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions)
lower weight bounds
[68Q20]
(*see:* **Optimal triangulations**)
lower well oil rate constraints *see:* upper and —
Löwner partial order
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
LP
[90C05]
(*see:* **Linear programming**)
LP
[05B35, 65K05, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C33, 90C57, 90C90]
(*see:* **Lexicographic pivoting rules**; **Modeling difficult optimization problems**)
LP duality
[90C05, 90C15]
(*see:* **Probabilistic constrained linear programming: duality theory**)
lP/NLP based branch and bound
[49M20, 90C11, 90C30]
(*see:* **Generalized outer approximation**)
lP relaxation
[68Q99, 90B80, 90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Branch and price: Integer programming with column generation**; **Facility location problems with spatial interaction**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**)
LP strategy for interval-Newton method in deterministic global optimization
lPS
(*see:* **Short-term scheduling of batch processes with resources**)
lS-CD
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)
LS problem
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
LSO
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
LSP
[65Fxx]
(*see:* **Least squares problems**)
LSTR
[90C30]
(*see:* **Large scale trust region problems**)
LSUO
[90C06]
(*see:* **Large scale unconstrained optimization**)
LU algorithm *see:* Implicit —
LU-decomposition
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
LU factorization *see:* classical —
Luc U-quasiconcave function
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
Łukasiewicz connective
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
Łukasiewicz implication
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
lumped systems *see:* reaction flux estimation in —
lunch *see:* no free —
Luus—Jaakola optimization procedure *see:* Direct search —
LX algorithm *see:* Implicit —
Lyapunov function
[90C30]
(*see:* **Suboptimal control**)
Lyusternik theorem
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
Lyusternik theorem *see:* classical —; high-order generalization of —

M

m *see:* algorithm solving a problem instance in time —; big- —; GRR- —; skew-symmetric matrix —
m-coloring problem
[90C08, 90C11, 90C27]
(*see:* **Quadratic semi-assignment problem**)
m-convex
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
M-convex function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
M_2-convex function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)

$M^\natural$-convex function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
M-convex functions *see:* L-convex functions and —
M-convex set
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
$M^\natural$-convex set
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
M-convexity
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
m-dial-a-ride
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
m-dimensional knapsack problem
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
M-estimator *see:* Huber —
M- and L-convex functions *see:* Fenchel-type duality for —
M-Pareto optimal solution
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
M-separation theorem
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
m-TSP
[90B06]
(*see:* **Vehicle routing**)
machine *see:* accepting computation of a Turing —; accepting state of a Turing —; alternating Turing —; bottleneck —; control state of a Turing —; deterministic Turing —; execution of a Turing —; exponentially space-bounded Turing —; exponentially time-bounded Turing —; final state of a Turing —; generalized eigenvalue proximal support vector —; input alphabet of a Turing —; language accepted by a Turing —; length of a partial computation of a Turing —; logspace Turing —; move of a Turing —; nonaccepting computation of a Turing —; nondeterministic Turing —; parallel random access —; partial computation of a Turing —; polynomially space-bounded Turing —; polynomially time-bounded Turing —; running time of a Turing —; size of the input of a Turing —; space complexity of a deterministic Turing —; space complexity of a nondeterministic Turing —; start state of a Turing —; state of a Turing —; tape cell of a Turing —; tape of a Turing —; time complexity of a deterministic Turing —; time complexity of a nondeterministic Turing —; transition rules of a Turing —; Turing —
machine interval
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
machine interval arithmetic
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
machine interval arithmetic *see:* inclusion principle of —
machine learning
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
machine-learning algorithm
[49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX]
(*see:* **Resource allocation for epidemic control**)
machine model *see:* Turing —
machine problem *see:* Generalized eigenvalue proximal support vector —
machine repetition
[90B35]
(*see:* **Job-shop scheduling problem**)
machine size
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
machine solving a problem *see:* Turing —
machines *see:* complexity of Turing —; distributed memory parallel —; identical —; nonidentical —; parallel —; shared memory parallel —; support vector —
macro scale network
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
macrostate
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)
Madansky upper bound *see:* Edmundson– —
Maehly method *see:* Lehmann– —
MAESTRO
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
magic numbers
[90C26, 90C90]
(*see:* **Global optimization in Lennard–Jones and morse clusters**)
magnitude *see:* order of —
main diagonal *see:* negative —
maintenance
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
maintenance routing problem *see:* airline —
majorant *see:* smallest K- —
majority
[90-XX]
(*see:* **Outranking methods**)
majority theorem
[90C26, 90C90]
(*see:* **Global optimization in Weber's problem with attraction and repulsion**)
majorization
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
maker *see:* decision —
makespan
[68Q25, 90B36, 90C60]
(*see:* **NP-complete problems and proof methodology**; **Stochastic scheduling**)
makespan *see:* minimization of —
making *see:* decision —; financial decision —; group decision —; hierarchical decision —; multicriteria decision —; multiple criteria decision —; Preference disaggregation approach: basic features, examples from financial decision —

making problems: optimization techniques *see:* Estimating data for multicriteria decision —
making with rolling horizon *see:* decision —
making under extreme events *see:* decision —
making under uncertainty *see:* decision —
mammography screening
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
management
[90C26, 90C30, 90C31]
(*see:* **Bilevel programming: introduction, history and overview**)
management
[90-01, 90B30, 90B50, 91B32, 91B52, 91B74]
(*see:* **Bilevel programming in management**)
management *see:* applications in environmental systems modeling and —; asset Liability —; Bilevel programming in —; Bilinear programming: applications in the supply chain —; catchment —; Competitive ratio for portfolio —; forest —; inventory —; logistics —; Mathematical programming methods in supply chain —; multistage inventory —; operational supply chain —; portfolio —; revenue —; strategic supply chain —; supply chain —
management decision support system *see:* Asset liability —
management and design *see:* Operations research models for supply chain —
management of environmental systems *see:* Global optimization in the analysis and —
management hypothesis *see:* inefficient —
Management Mathematics *see:* iMA Journal of —
management models *see:* multistage inventory —; single stage inventory —
management in supply chains *see:* Inventory —
manager *see:* layout —
mandatory work first algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
Mangasarian–Fromovitz constraint qualification
[90C31]
(*see:* **Sensitivity and stability in NLP: continuity and differential stability**)
Mangasarian–Fromovitz constraint qualification
[90C26, 90C31, 90C34, 90C39]
(*see:* **Parametric global optimization: sensitivity**; **Second order optimality conditions for nonlinear optimization**)
Mangasarian–Fromovitz CQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
Manhattan distance
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
Manhattan distance
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
Manhattan distances
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
manifold
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
manifold *see:* Riemannian —
manipulation *see:* symbolic —
Mann–Whitney statistic
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
many conditions moment problem *see:* infinite —
many-valued families of the Pinkava logic algebras
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
many-valued logic algebra
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
many-valued logic algebras *see:* Finite complete systems of —
many-valued logic implication
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
many-valued logic normal form *see:* complete —
many-valued logic system *see:* lattice-type —
many-valued logics
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
many-valued logics
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T15, 68T27, 68T30]
(*see:* **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**)
many-valued logics *see:* classification of —; taxonomy of the PI-algebras of —
many-valued normal form
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
MAP
[90C08, 90C10, 90C11, 90C27, 90C57, 90C59]
(*see:* **Multidimensional assignment problem**; **Quadratic assignment problem**)
map *see:* adjoint linear —; cone-convex —; contact —; maximal monotone —; monotone —; normal —; Peano —; proximal —; proximity —; quasimonotone —; semistrictly quasimonotone —; standard part —; strictly monotone —; strictly pseudomonotone —; strictly quasimonotone —
map optimization *see:* fluence —
map overlap *see:* contact —
map overlap maximization problem, CMO *see:* Contact —
mapping *see:* best response —; closed point-to-set —; computation and data —; Contraction- —; DNA —; nearest point —; optimal solution —; Φ-isotone —; point-to-set —; pseudomonotone —; semismooth —; strongly semismooth —; zero-epi —
mapping technique *see:* pictogram translation —; receiver initiated —; sender initiated —
mappings *see:* approximation of nonsmooth —; approximations of nonsmooth —; contraction —; method of —; point-to-set —
maps *see:* Generalized monotone multivalued —; Generalized monotone single valued —; homotopic —
maps: properties and applications *see:* Pseudomonotone —
Maratos effect
[65K05, 65K10, 90C06, 90C30, 90C34]
(*see:* **Feasible sequential quadratic programming**)
margin *see:* multivariable stability —; Stability —
margin K *see:* multivariable stability —

marginal allocation
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
marginal analysis
[90C60]
(*see:* **Complexity of degeneracy**)
marginal constraint
[90C35]
(*see:* **Multi-index transportation problems**)
marginal distribution functions
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
marginal function
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
marginal function optimization
[49J35, 65K99, 74A55, 74M10, 74M15, 90C26]
(*see:* **Quasidifferentiable optimization: applications**)
marginal functions
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
marginal probability distribution function *see:* one-dimensional —; two-dimensional —
marginal value
[90C60]
(*see:* **Complexity of degeneracy**)
marginal value *see:* positive —
marginal value formula
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
marginal values *see:* negative —; positive —
marginals *see:* table with given —
margins *see:* hierarchical collection of —
Margolin method *see:* Schruben– —
Maritime inventory routing problems
market equilibrium *see:* Oligopolistic —
market equilibrium conditions
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)
market equilibrium conditions
[65K10, 65M60]
(*see:* **Variational inequalities**)
market line *see:* capital —; security —
market model *see:* Sharpe single index —
market portfolio
[91B50]
(*see:* **Financial equilibrium**)
market portfolio
[91B50]
(*see:* **Financial equilibrium**)
markets *see:* aspatial and spatial —; Operations research and financial —; spatial —
Markoff theorem *see:* Gauss– —
markov chain
(*see:* **Bayesian networks**)
Markov chain *see:* finite-state —; stationary-state —
Markov chain sampling
[65K05, 90C30]
(*see:* **Random search methods**)
Markov chains
[90C27]
(*see:* **Operations research and financial markets**)
Markov decision process
[49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX]
(*see:* **Resource allocation for epidemic control**)
Markov kernel
[28-XX, 49-XX, 60-XX, 90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**; **General moment optimization problems**)
Markov kernel
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
Markov model and Gibbs sampler *see:* hidden —
Markov models *see:* hidden —
Markov process
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
Markov process
[49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX]
(*see:* **Resource allocation for epidemic control**)
markov processes and their simulation *see:* Derivatives of —
Markov strategy
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
Markov transformation
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
markowitz mean-variance model *see:* Portfolio selection: —
Marquardt *see:* Levenberg– —
Marquardt algorithm *see:* Levenberg– —
Marquardt method *see:* Levenberg– —
Marquardt rule *see:* Levenberg– —
marriage problem
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
marriage problem
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
marriage problem *see:* stable —
Martin algorithm
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
martingale *see:* score function —
mass
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
mass
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
mass balance constraints
[90C35]
(*see:* **Maximum flow problem**; **Minimum cost flow problem**)
mass balances
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in distillation systems**)

mass and energy balance equations
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in distillation systems**)
mass, energy and momentum balances
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
mass exchange *see:* modeling —
mass exchange matches
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: mass and heat exchanger networks**)
mass exchange network *see:* heat and —
mass exchange networks *see:* Flexible —
mass exchanger
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
mass exchanger network
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**)
mass and heat exchange
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: mass and heat exchanger networks**)
mass and heat exchanger network
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**)
mass and heat exchanger networks *see:* MINLP: —; Mixed integer linear programming: —
mass/heat transfer module
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**)
mass separating agents
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
massive data sets *see:* least squares problems with —
massively parallel computing
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
massively parallel solution *see:* Stochastic network problems: —
master problem
[49M29, 90C06, 90C10, 90C11, 90C15, 90C30, 90C35, 90C57, 90C90]
(*see:* **Decomposition algorithms for the solution of multistage mean-variance optimization problems**; **Generalized benders decomposition**; **Modeling difficult optimization problems**; **Multicommodity flow problems**; **Simplicial decomposition**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**)
master problem
[90C30]
(*see:* **Simplicial decomposition**)
master problem *see:* complete —; disjunctive OA —; full —; MILP —; MIQP —; primal —; relaxed —; relaxed primal —; restricted —
master program
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
master program *see:* full —; reduced —
master-slave scheme
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
master subproblem
[90C11, 90C31]
(*see:* **Parametric mixed integer nonlinear optimization**)
match-network problem
[90C90]
(*see:* **MINLP: heat exchanger network synthesis**)
matches *see:* mass exchange —
matching
[05C85, 90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**; **Directed tree networks**; **Maximum flow problem**)
matching *see:* 3-DIMENSIONAL —; algorithm pre- —; Assignment and —; b- —; bipartite —; maximum —; Maximum partition —; maximum pre- —; partition —; perfect —; pre- —
matching of derivative conditions
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
matching heuristic *see:* maximal —
matching-I *see:* algorithm partition- —
matching model
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
matching problem
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
matching problem *see:* 2- —; 3-dimensional —; b- —; maximum cardinality —; maximum partition —; maximum pre- —; perfect —; perfect b- —; weighted —; weighted bipartite —
Matching (ROM) *see:* recursive Opt —
Matching Subgraph Problem *see:* minMax —
matchings *see:* perfect —
material derivative approach
[49J20, 49J52]
(*see:* **Shape optimization**)
material flow
[90-02]
(*see:* **Operations research models for supply chain management and design**)
material flows *see:* balance equations for —
material laws *see:* discretized hemivariational inequalities for nonlinear —
materials *see:* laminated composite —
mathematical areas *see:* software package for specific —
mathematical and computational certainty
(*see:* **LP strategy for interval-Newton method in deterministic global optimization**)
mathematical economics
[90B80, 90B85, 90Cxx, 91Axx, 91Bxx]
(*see:* **Facility location with externalities**)
mathematical finance
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]

(*see:* **Information-based complexity and information-based optimization**)
mathematical finance
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
mathematical formulation
[90C11, 90C29, 90C90]
(*see:* **Multi-objective optimization: interaction of design and control**)
mathematical model
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30, 90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Identification methods for reaction kinetics and transport**; **Modeling difficult optimization problems**)
mathematical modeling
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)
mathematical models
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
mathematical program *see:* extreme point —
mathematical program with affine equilibrium constraints
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
mathematical program with equilibrium constraints
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
mathematical program with equilibrium constraints
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
mathematical Programming
[90C05, 90C06, 90C22, 90C25, 90C30, 90C51]
(*see:* **Interior point methods for semidefinite programming**; **Saddle point theory and optimality conditions**)
mathematical programming
[26B25, 26E25, 49J52, 62H30, 90B80, 90B85, 90C11, 90C27, 90C99, 90Cxx, 91Axx, 91Bxx]
(*see:* **Facility location with externalities**; **Operations research and financial markets**; **Quasidifferentiable optimization**; **Statistical classification: optimization approaches**)
mathematical programming *see:* multi-objective —
Mathematical programming for data mining
Mathematical programming methods in supply chain management
(*referred to in:* **Generalizations of interior point methods for the linear complementarity problem**; **Simultaneous estimation and optimization of nonlinear problems**)
(*refers to:* **Generalizations of interior point methods for the linear complementarity problem**; **Simultaneous estimation and optimization of nonlinear problems**)
mathematical programming problem *see:* nonlinear —
mathematical rigor *see:* with —
mathematical software
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
mathematical software
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
Mathematics *see:* iMA Journal of Management —
matric matroid
[90C09, 90C10]
(*see:* **Matroids**)
matrices
[90C33]
(*see:* **Linear complementarity problem**)
matrices *see:* Abaffian —; completion of —; completion to completely positive and contraction —; Hessian —; Interval analysis: eigenvalue bounds of interval —; positive definite —; positive semidefinite —; q- —; Stochastic programming: parallel factorization of structured —; updating input-output —
matrix
[90C09, 90C10, 90C25, 90C33, 90C55]
(*see:* **Combinatorial matrix analysis**; **Splitting method for linear complementarity problems**)
matrix
[90C30]
(*see:* **Frank–Wolfe algorithm**)
matrix *see:* adjacency —; anti-Monge —; anti-Robinson —; banded —; basic —; bisymmetric —; bisymmetric positive semidefinite —; chess-board —; circulant —; classification —; column sufficient —; completely positive —; completion of a partial —; complex interval —; condition number of a —; confusion —; consistent —; consistent judgment —; contraction —; copositive —; diagonal —; diagonal shift —; diagonal underestimation —; dimensional symmetric interval —; distance —; doubly nonnegative —; doubly stochastic —; Euclidean distance —; extended —; extreme eigenvalue of an interval —; fully indecomposable —; geodesic Hessian —; graph of a —; H- —; Hermitian interval —; Hessian —; ill-conditioned —; ill-conditioned coefficient —; incidence —; inductive structure of an irreducible —; inertia of a —; interval Hessian —; interval of variation of an eigenvalue of an interval —; irreducible —; irreducible components of a —; Jacobian —; Kalmanson —; L- —; Monge —; monotone —; n-fold —; n Hessian —; node-arc incidence —; nonbasic —; nonsingular —; oblique projection —; orthogonal —; p- —; partial —; partial completely positive —; partial contraction —; partial definite —; partial distance —; partial Hermitian —; partial semidefinite —; pattern of a —; permanent of a —; permutation —; polar —; polynomial —; positive definite —; positive semidefinite —; positive semidefinite symmetric —; product —; projected Lagrangian Hessian —; projection —; Q- —; qualitative class of a —; rank-one —; real interval —; real symmetric interval —; realization of a —; regular —; rotation —; row sufficient —; S*- —; seed —; sign —; sign-nonsingular —; sign pattern of a —; singular —; skew-symmetric —; sparse —; standard determinant expansion of a —; stiffness —; stochastic —; strictly copositive —; strongly nonsingular —; strongly positive definite —; strongly regular —; structured —; sufficient —; sum —; symmetric —; Toeplitz —; totally unimodular —; transition —; transition probability —;

tridiagonal —; unimodular —; vertex matrix of an interval —; well-conditioned —
matrix of activities
[90Cxx]
(*see:* **Discontinuous optimization**)
matrix analysis
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
matrix analysis
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
matrix analysis *see:* Combinatorial —
matrix class invariant under principal pivoting
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
matrix classes
[90C33]
(*see:* **Linear complementarity problem**)
matrix classes
[90C33]
(*see:* **Linear complementarity problem**)
matrix completion *see:* rank —
matrix completion problem
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
matrix completion problem *see:* distance —; Euclidean distance —; positive semidefinite —
Matrix completion problems
(05C50, 15A48, 15A57, 90C25)
(*referred to in:* **Semidefinite programming and determinant maximization**)
(*refers to:* **Interior point methods for semidefinite programming**; **Semidefinite programming and determinant maximization**)
matrix estimation *see:* covariance —
matrix factorization
[90C15]
(*see:* **Stochastic programming: parallel factorization of structured matrices**)
matrix factorization *see:* modifying —; parallel —; structured —
matrix inequality *see:* bilinear —; linear —
matrix of an interval matrix *see:* vertex —
matrix of a Lagrangian function *see:* Hessian —; projected Hessian —
matrix M *see:* skew-symmetric —
matrix notation *see:* relational —
matrix notation for relational operations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
matrix patterns and graphs
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
matrix representation of a relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
matrix rounding problem
[90C35]
(*see:* **Maximum flow problem**)
matrix of second partial derivatives
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
matrix splitting methods in quadratic programming
[90C30]
(*see:* **Cost approximation algorithms**)
matrix in standard form
[65Fxx]
(*see:* **Least squares problems**)
matroid
[90C09, 90C10]
(*see:* **Matroids**)
matroid
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
matroid *see:* acyclic oriented —; bases of an oriented —; basis orientation of an oriented —; binary —; closed of a —; closure operator for a —; connectivity of a —; contraction of a —; disconnected —; dual —; graphic —; infinitely connected —; linear —; matric —; minor of a —; orientable —; orthogonal —; partition —; rank of a —; regular —; representable —; restriction of a —; set of bases of a —; ternary —; totally acyclic oriented —; transversal —; underlying —; uniform —; vector of an oriented —; vectorial —; weight function of a —; weighted —
matroid base polytope
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
matroid connectivity
[90C09, 90C10]
(*see:* **Matroids**)
matroid contraction
[90C09, 90C10]
(*see:* **Matroids**)
matroid elements *see:* contracting —; deleting —
matroid minor
[90C09, 90C10]
(*see:* **Oriented matroids**)
matroid representation
[90C09, 90C10]
(*see:* **Matroids**)
matroid restriction
[90C09, 90C10]
(*see:* **Matroids**)
matroid theory
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
Matroids
(90C09, 90C10)
(*referred to in:* **Oriented matroids**)
(*refers to:* **Oriented matroids**)
matroids *see:* axiom systems for oriented —; contraction in —; deletion in —; duality of —; oriented —

Matula estimate
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
mature
(*see:* **State of the art in modeling agricultural systems**)
maturities *see:* estimating the spot rate for bonds with constant —
maturity *see:* term to —; yield to —
MAX-2-SAT
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
max-bisection
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
max-clique problem
[90C60]
(*see:* **Complexity theory**)
max-closed form transformation *see:* unimodular —
max-closed form transformations *see:* unimodular —
max-closed function
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
max-closed set
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
max-closed sets
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
mAX-CSP
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
max-cut
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
MAX-CUT *see:* Maximum cut problem —
max Cut (MC)
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
max-det problem
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
max digraph *see:* min- —
max duality *see:* double- —
max–flow algorithm
[90-XX]
(*see:* **Survivable networks**)
max-flow min-cut theorem
[05C05, 05C40, 68R10, 90C35]
(*see:* **Maximum flow problem**; **Network design problems**)
max-flow min-cut theorem
[90C35]
(*see:* **Maximum flow problem**)
max fractional program *see:* min- —
max-function
[46A20, 52A01, 65K05, 90C30]
(*see:* **Composite nonsmooth optimization**; **Minimax: directional differentiability**)
max-function
[49K35, 49M27, 65K05, 65K10, 90C25, 90C30]
(*see:* **Convex max-functions**; **Minimax: directional differentiability**)
max-function *see:* convex —
max-functions *see:* Convex —
max graph *see:* min- —
MAX-MIN ant system
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
max-min fractional program
[90C32]
(*see:* **Fractional programming**)
max-min-max optimization problem
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
max optimization problem *see:* max-min- —
max-r-Constraint Satisfaction Problem
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
max-r-CSP
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
max-regret-fc and max-regret heuristics
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
max-regret heuristics *see:* max-regret-fc and —
MAX-SAT
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C10, 90C27, 94C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**; **Maximum satisfiability problem**)
MAX-SAT problem *see:* weighted —
max Steiner tree *see:* min- —
max TSP
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
max-type function
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
max-type functions *see:* difference of —
maxdiag fine structures
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
MaxEnt
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
maxima
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
maximal
[05C15, 05C17, 05C35, 05C62, 05C69, 05C85, 90C22, 90C27, 90C35, 90C59]
(*see:* **Lovász number**; **Optimization problems in unit-disk graphs**)

maximal alternative
[90-XX]
(*see:* **Outranking methods**)
maximal best approximation
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
maximal clique
[05C69, 05C85, 68W01, 90C20, 90C59]
(*see:* **Heuristics for maximum clique and independent set**; **Standard quadratic optimization problems: applications**)
maximal flow problem
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
maximal independent set
[90C09, 90C10]
(*see:* **Matroids**)
maximal matching heuristic
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
maximal monotone map
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
maximal planar subgraph
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
maximal similarity subtree isomorphism
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
maximal subtree isomorphism
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
maximally informative genes *see:* Selection of —
maximin objective function
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
maximin path length
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
maximization
[90C60]
(*see:* **Computational complexity theory**)
maximization *see:* expectation- —; Semidefinite programming and determinant —
maximization algorithm *see:* expectation- —
maximization interval *see:* expectation- —
maximization method *see:* vector —
maximization of output/input
[90C32]
(*see:* **Fractional programming**)
maximization problem *see:* global —
maximization problem, CMO *see:* Contact map overlap —
maximization of productivity
[90C32]
(*see:* **Fractional programming**)
maximization of return on investment
[90C32]
(*see:* **Fractional programming**)
maximization of return/risk
[90C32]
(*see:* **Fractional programming**)
maximization of sales
(*see:* **Short-term scheduling of batch processes with resources**)
Maximization of the Smallest of Several Ratios
[90C32]
(*see:* **Fractional programming**)
maximize net present value
(*see:* **Planning in the process industry**)
maximize operating cash flow
(*see:* **Planning in the process industry**)
maximizer
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
maximizer
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
maximizer *see:* discrete global —; discrete left local —; global —; local —; strict local —
maximizers *see:* set of discrete ε-global local —
maximizing minimum distance
[90C29]
(*see:* **Multicriteria sorting methods**)
maximizing a sum of ratios
[90C32]
(*see:* **Fractional programming**)
maximum
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
maximum *see:* local —
maximum absolute deviation
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
maximum bipartite subgraph
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
maximum cardinality matching problem
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
maximum clique
[05-04, 05C60, 05C69, 05C85, 37B25, 68Q25, 68W01, 90C10, 90C20, 90C27, 90C35, 90C59, 90C60, 91A22]
(*see:* **Evolutionary algorithms in combinatorial optimization**; **Heuristics for maximum clique and independent set**; **Maximum constraint satisfaction: relaxations and upper bounds**; **NP-complete problems and proof methodology**; **Quadratic knapsack**; **Replicator dynamics in combinatorial optimization**)
maximum clique and independent set *see:* Heuristics for —
maximum clique problem
[05C15, 05C60, 05C62, 05C69, 05C85, 37B25, 60G35, 65K05, 68Q25, 68R10, 68W40, 90C08, 90C11, 90C20, 90C27, 90C35, 90C57, 90C59, 91A22]
(*see:* **Differential equations and global optimization**; **Domination analysis in combinatorial optimization**; **Optimization problems in unit-disk graphs**; **Quadratic assignment problem**; **Replicator dynamics in combinatorial**

optimization; **Standard quadratic optimization problems: applications**)
maximum condition
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
maximum constraint satisfaction
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
maximum constraint satisfaction problem
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
Maximum constraint satisfaction: relaxations and upper bounds
(90C10)
(*referred to in:* **Frequency assignment problem**; **Graph coloring**)
(*refers to:* **Frequency assignment problem**; **Graph coloring**)
maximum coverage location problem
[90B10, 90B80, 90C35]
(*see:* **Network location: covering problems**)
maximum coverage location problem
[90B10, 90B80, 90C35]
(*see:* **Network location: covering problems**)
maximum cut
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
Maximum cut problem, MAX-CUT
(*refers to:* **Combinatorial test problems and problem generators**; **Continuous global optimization: models, algorithms and software**; **Derivative-free methods for non-smooth optimization**; **Greedy randomized adaptive search procedures**; **Heuristic search**; **Integer programming**; **NP-complete problems and proof methodology**; **Quadratic integer programming: complexity and equivalent forms**; **Random search methods**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and the sensor network localization problem, SNLP**; **Solving large scale and sparse semidefinite programs**; **Variable neighborhood search methods**)
maximum degree
[90C35]
(*see:* **Graph coloring**)
maximum entropy
[62F10, 94A17]
(*see:* **Entropy optimization: parameter estimation**)
maximum entropy *see:* axiomatic derivation of the principle of —; Jaynes —; principle of —
Maximum entropy and game theory
maximum entropy principle
[90C25, 94A08, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**; **Jaynes' maximum entropy principle**; **Maximum entropy principle: image reconstruction**)
maximum entropy principle
[90C25, 94A08, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**; **Maximum entropy principle: image reconstruction**)
maximum entropy principle *see:* Jaynes' —
Maximum entropy principle: image reconstruction
(94A17, 94A08)
(*referred to in:* **Entropy optimization: interior point methods**; **Entropy optimization: parameter estimation**; **Entropy optimization: shannon measure of entropy and its properties**; **Jaynes' maximum entropy principle**; **Optimization in medical imaging**)
(*refers to:* **Entropy optimization: interior point methods**; **Entropy optimization: parameter estimation**; **Entropy optimization: shannon measure of entropy and its properties**; **Jaynes' maximum entropy principle**; **Optimization in medical imaging**)
maximum flow
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
Maximum flow problem
(90C35)
(*referred to in:* **Auction algorithms**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Generalized networks**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Nonoriented multicommodity flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)

(*refers to:* **Auction algorithms**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Generalized networks**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Nonoriented multicommodity flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
maximum flow problem
[90C35]
(*see:* **Maximum flow problem**)
maximum flow problem
[90C35]
(*see:* **Maximum flow problem**)
maximum flow problem with nonnegative lower bounds
[90C35]
(*see:* **Maximum flow problem**)
maximum flow problems
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
maximum function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
maximum function with dependent constraints
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
maximum Independent Set
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)

maximum Independent Set Problem
[68Q25, 68R10, 68W40, 90C20, 90C27, 90C59, 90C60]
(*see:* **Domination analysis in combinatorial optimization**; **Quadratic knapsack**)
maximum likelihood
[62F10, 65T40, 90C26, 90C30, 90C90, 94A17]
(*see:* **Entropy optimization: parameter estimation**; **Global optimization methods for harmonic retrieval**)
Maximum likelihood detection via semidefinite programming
(65Y20, 68W25, 90C27, 90C22, 49N15)
maximum likelihood estimate
[15A15, 34-xx, 34Bxx, 34Lxx, 90C25, 90C55, 90C90, 93E24]
(*see:* **Complexity and large-scale least squares problems**; **Semidefinite programming and determinant maximization**)
maximum likelihood estimation
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
maximum likelihood method *see:* iterative quadratic —
maximum likelihood principle
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
maximum matching
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
maximum mean cut
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
maximum mean-weight cut
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
maximum norm *see:* weighted —
maximum number of well switches
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
maximum oil, gas and water capacity constraints
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
Maximum partition matching
(05A18, 05D15, 68M07, 68M10, 68Q25, 68R05)
(*referred to in:* **Assignment and matching**; **Assignment methods in clustering**; **Bi-objective assignment problem**; **Communication network assignment problem**; **Frequency assignment problem**; **Linear ordering problem**; **Quadratic assignment problem**)
(*refers to:* **Assignment and matching**; **Assignment methods in clustering**; **Bi-objective assignment problem**; **Communication network assignment problem**; **Frequency assignment problem**; **Quadratic assignment problem**)
maximum partition matching problem
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
maximum path length
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
maximum planar subgraph
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
maximum point *see:* global —; local —; strict local —
maximum a posteriori principle
(*see:* **Bayesian networks**)
maximum pre-matching
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
maximum pre-matching problem
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
maximum principle
[49J15, 49K15, 93C10]
(*see:* **Pontryagin maximum principle**)
maximum principle *see:* duality and —; high-order local —; Kimura —; local —; pontryagin's —
maximum principle for abnormal extremals *see:* High-order —
maximum principle for Lagrangian problems *see:* high-order local —
maximum profit-to-time ratio cycle
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
maximum rank completion problem
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
maximum satisfiability
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
maximum satisfiability
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
Maximum satisfiability problem
(03B05, 68Q25, 90C09, 90C27, 68P10, 68R05, 68T15, 68T20, 94C10)
(*referred to in:* **Greedy randomized adaptive search procedures**; **Integer programming**)
(*refers to:* **Greedy randomized adaptive search procedures**; **Integer programming**; **Integer programming: branch and bound methods**; **Simulated annealing methods in protein folding**)
maximum similarity subtree isomorphism
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
maximum subtree isomorphism
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
maximum-type function
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
maximum Variance Unfolding
[51K05, 52C25, 68Q25, 68U05, 90C22, 90C35]
(*see:* Graph realization via semidefinite programming)
maximum-volume ellipsoid
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
maximum weight clique
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
maximum weight clique
[90C20]

(*see:* **Standard quadratic optimization problems: applications**)
maximum weight clique problem
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**; **Standard quadratic optimization problems: applications**)
maximum weight independent sets
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
maximum weight trace
[90C35]
(*see:* **Optimization in leveled graphs**)
maximum weighted distance
[90B85, 90C27]
(*see:* **Single facility location: circle covering problem**)
maximum weighted independent set
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
maximum weighted planar graph
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
maxmin function
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
maxmin function
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
maxmin objective
[90B85]
(*see:* **Single facility location: multi-objective rectilinear distance location**)
Maybee–Quirk theorem *see:* Bassett– —
Mayer form
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
Mazur–Orlicz theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
Mazur–Orlicz version of the Hahn–Banach theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
Mazurkiewicz lemma *see:* Knaster–Kuratowski– —
(MC) *see:* max Cut —
McCormick SOCQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
MCD
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
MCDM
[90C29]
(*see:* **Decision support systems with multiple criteria**)
MCDM
[90-XX, 90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**; **Outranking methods**)
mCl
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
MCP
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
MCTP
[90B06, 90B10, 90C26, 90C35]
(*see:* **Minimum concave transportation problems**)
MDO
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
MDO paradigm
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
MDVSP
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
Mead algorithm *see:* Nelder– —
mean absolute deviation
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
mean cut *see:* maximum —
mean cycle *see:* minimum —
mean field approximation
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
mean method *see:* geometric —; overall —; revised geometric —
mean product
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
mean value
[90C26, 90C30]
(*see:* **Bounding derivative ranges**)
Mean-Value for Composite Convexifiable Function *see:* integral —
mean value cross decomposition
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)
mean value extension
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval fixed point theory**)
mean value function
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
mean value function
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
mean value problem
[90C90, 91B28]
(*see:* **Robust optimization**)
mean value theorem
[65G20, 65G30, 65G40, 65H20, 65K05, 65K99, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Interval Newton methods**)
mean-variance
[90C27]
(*see:* **Operations research and financial markets**)

mean-variance model *see:* Portfolio selection: markowitz —
mean-variance optimization problems *see:* Decomposition algorithms for the solution of multistage —
mean-variance portfolio analysis
[91B50]
(*see:* **Financial equilibrium**)
mean-weight cost
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
mean-weight cut *see:* maximum —
meaningful words
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
means criterion *see:* k- —
measurable criterion
[90C29, 91A99]
(*see:* **Preference disaggregation**)
measure *see:* acceptance —; α-concave —; Baire —; bottleneck —; combined relative —; contracting —; contraction/approximation —; controllability —; discrete —; dissimilarity —; distance —; empirical —; γ-concave probability —; Gaussian —; heuristic —; inner regular —; Loeb —; logconcave —; logconcave probability —; logconvex —; logconvex probability —; quasiconcave —; quasiconcave probability —; Radon —; similarity —; subgradient locality —; Wiener —; Wiener probability —; zemel —
measure of cross-entropy *see:* Kullback–Leibler —
measure of entropy and its properties *see:* Entropy optimization: shannon —
measure of noncompactness
[90C33]
(*see:* **Order complementarity**)
measure space
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
measure space *see:* probability —
measure spaces
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
measure theory
[01A99]
(*see:* **Carathéodory, Constantine**)
measure theory
[01A99, 03H10, 49J27, 90C34]
(*see:* **Carathéodory, Constantine**; **Semi-infinite programming and control problems**)
measure of uncertainty
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
measurement
[90C29]
(*see:* **Preference modeling**)
measurement *see:* Supply chain performance —
measurement techniques
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
measures *see:* condition —; Derivatives of probability —; dominated family of —; normalization of —; Radon —; regular family of probability —; weak convergence of probability —; weakly L_1 (v)-differentiable family of —
measures, logconvexity *see:* Logconcave —
mechanical constructions *see:* linearly elastic —
mechanical models
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
mechanical principle of Fourier
[15A39, 90C05]
(*see:* **Farkas lemma**)
mechanics *see:* applications in —; computational —; Hemivariational inequalities: applications in —; inequality or nonsmooth —; molecular —; Multilevel optimization in —; nonsmooth —; parallel computation in —; smooth potentials and stability in —; unilateral —
mechanizing
[01A99]
(*see:* **History of optimization**)
median location
[90B85]
(*see:* **Single facility location: multi-objective rectilinear distance location**)
median location-allocation
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
median location-allocation problem *see:* p- —
median problem *see:* p- —
median problem in a network *see:* 1- —
medical applications
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
medical diagnosis
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 90C09, 90C10, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Optimization in boolean classification problems**)
medical diagnosis
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
medical image processing *see:* optimization in —
medical imaging
[90C90]
(*see:* **Optimization in medical imaging**)
medical imaging *see:* Optimization in —
medicine
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
medium-level software
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
medium regression *see:* isotonic —; quasiconvex —
Medium-term scheduling of batch processes
meet
[90C35]
(*see:* **Multi-index transportation problems**)
meet semisublattice
[47J20, 49J40, 65K10, 90C33]

(*see:* **Solution methods for multivalued variational inequalities**)
Megiddo's parametric search
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
Megiddo parametric search
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
membership function
[90C90]
(*see:* **Chemical process planning**)
membership oracle
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
memetic algorithms
[68T20, 68T99, 90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Metaheuristics**; **Traveling salesman problem**)
memory *see:* adaptive —; limited- —; short-term —
memory affine reduced Hessian *see:* limited- —
memory algorithm *see:* limited- —
memory approach *see:* limited- —
memory BFGS method *see:* limited- —
memory in GRASP *see:* long-term —
memory model *see:* queueing shared- —
memory parallel computer *see:* distributed —
memory parallel machines *see:* distributed —; shared —
memory reduced-Hessian BFGS algorithm *see:* limited- —
memory strategy equilibrium
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
memory strategy Nash equilibrium
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
memory symmetric rank-one approach *see:* limited- —
memoryless BFGS method
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
MEN
(93A30, 93B50)
(*referred to in:* **Continuous global optimization: models, algorithms and software**; **Global optimization of heat exchanger networks**; **Global optimization methods for systems of nonlinear equations**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**)
(*refers to:* **Global optimization of heat exchanger networks**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**)
MEN
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**)
MEN superstructure
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**)
MEN synthesis method *see:* sequential —
MEN synthesis model *see:* multiperiod MINLP —
menace of the expanding grid
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
mergers and acquisitions *see:* Multicriteria methods for —
merit function
[49M37, 65K05, 65K10, 90C06, 90C15, 90C25, 90C26, 90C30, 90C33, 90C34, 90C35]
(*see:* **Feasible sequential quadratic programming**; **Implicit lagrangian**; **Inequality-constrained nonlinear optimization**; **Nonlinear least squares problems**; **Simplicial decomposition algorithms**; **Stochastic bilevel programs**)
merit function
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05, 90C06, 90C25, 90C30, 90C33, 90C35]
(*see:* **Cost approximation algorithms**; **Implicit lagrangian**; **Simplicial decomposition algorithms**; **Variational principles**)
merit function *see:* list square —
merit index
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
mesh
[90C35]
(*see:* **Feedback set problems**)
mesh *see:* toroidal —
mesh networks
[05C85]
(*see:* **Directed tree networks**)
message-driven
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)
Met-Enkephalin
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
Met-Enkephalin
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
meta-UTA
[90C29, 91A99]
(*see:* **Preference disaggregation**)
metaheuristic
[90C27, 90C29]
(*see:* **Multi-objective combinatorial optimization**)
metaheuristic *see:* hybrid —
metaheuristic algorithms *see:* heuristic– —
metaheuristic algorithms for the traveling salesman problem *see:* Heuristic and —
Metaheuristic algorithms for the vehicle routing problem
(90B06, 90C59)
Metaheuristics
(68T20, 90C59, 90C27, 68T99)
metaheuristics
[90B06]
(*see:* **Vehicle routing**)

metaheuristics
[65H20, 65K05, 90-01, 90B06, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**; **Vehicle routing**)
metaheuristics *see:* GRASP in hybrid —
metaminimax theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
metamodel
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
method *see:* a priori —; achievable region —; active set —; adaptive aggregation —; adaptive computational —; adaptive random search —; adjoint derivative —; Agmon–Motzkin–Fourier relaxation —; Aitken double sweep —; analytic center cutting plane —; approximate Newton —; arc oriented branch and bound —; Arrow–Hurwicz gradient —; assignment —; averaging —; backpropagation —; Bellman–Ford —; BFGS —; binary search —; bisection —; bold-driver —; bordering —; boundary condition iteration —; branch and bound —; Broyden–Fletcher–Goldfarb–Shanno —; bundle-Newton —; canonical dual transformation —; Cauchy —; center of gravity —; Chan —; Chebyshev iterative —; Chevron —; conditional gradient —; conjugate gradient —; conjugate subgradient —; continuation —; coordinate descent —; corridor —; criss-cross —; cutting angle —; cutting plane —; Cyclic coordinate —; D'Esopo–Pape —; damped Gauss–Newton —; damped Newton —; Davidon–Fletcher–Powell —; De La Garza —; Delphi —; derivative-free descent —; descent —; destructive —; DFP —; diffusion equation —; Dinkelbach —; disaggregation —; discrete truncated Newton —; discretization —; distance scaling —; distrust region —; dogleg —; double description —; Douglas—Rachford —; downhill simplex —; edge contraction —; ellipsoid —; empirical —; ϵ-constraint —; ϵ-subgradient —; Euler —; Evtushenko —; Exact penalty —; extended cutting plane —; extended support problems —; exterior point —; extremal basis —; feasible decomposition —; fictitious domain —; finite element —; Fletcher–Reeves —; Forrest–Goldfarb —; Fourier–Motzkin —; Fourier–Motzkin elimination —; full space SQP —; full-step Gauss–Newton —; Galerkin spectral —; Gauss–Newton —; Gauss–Newton method: Least squares, relation to Newton's —; Gauss–Seidel —; Gauss–Southwell —; generalized Benders —; generalized cutting plane —; geometric mean —; global convergence problem for the Rosen —; Global optimization: cutting angle —; goal coordination —; Goerisch —; golden section —; Goldfarb–Idnani —; greedy —; heavy ball —; Heun —; heuristic optimization —; homotopic —; homotopy —; homotopy continuation —; homotopy Newton —; Huang —; Hungarian —; implicit restarted Lanczos —; Inclusion —; incremental gradient —; inexact Newton —; integer L-shaped —; interactive —; interior point —; interior point logarithmic barrier —; interval Newton —; IQML —; iterative —; iterative quadratic maximum likelihood —; iterative regularization —; Jacobi —; Kackmartz —; Karmarkar —; Keifer–Wolfowitz —; Kelley —; Kelley's classical cutting plane —; Kelley cutting plane —; KKT-based —; Kojima–Shindo —; Krawczyk —; Krawczyk variation of the interval Newton —; l-shaped —; Lagrangian finite generation —; largest inscribed sphere —; least-index criss-cross —; least-index pivoting —; Lehmann–Maehly —; Lemke —; Levenberg–Marquardt —; Levitin–Polyak —; lexicographic dual simplex —; lexicographic primal simplex —; lexicographic simplex —; lexicographic variant of the constraint-by-constraint —; likelihood ratio —; limited-memory BFGS —; linear CG —; local search —; Logic-based outer-approximation —; memoryless BFGS —; metric-based —; metropolis Monte Carlo —; model-based —; model coordination —; modified Cauchy —; modified Newton —; Monte-Carlo —; MOSA —; multicriteria sorting —; multifrontal —; multiplier —; multivariate interval Newton —; NC —; Newsam–Ramsdell —; Newton's —; Newton–Raphson —; Newton-type —; node oriented branch and bound —; noising —; nonadaptive —; Nondifferentiable optimization: Newton —; nonfeasible decomposition —; noninteractive —; nonlinear CG —; nonparametric statistical —; nonsmooth Newton —; OA —; on-line —; outer approximation —; overall mean —; overdetermined Yule–Walker —; parameterization —; partial-update Newton —; partially asynchronous iterative —; partitioned quasi-Newton —; partitioning —; penalty-based —; piecewise sequential quadratic programming —; Piela —; pilot —; pivot —; Polak–Ribiére —; polyblock approximation —; Powell —; Powell-symmetric–Broyden —; power —; primal —; principal pivoting —; projection —; proximal bundle —; proximal-like —; proximal point —; proximal point bundle —; pure Monte-Carlo —; pure NP —; QBB global optimization —; QR —; Quadratic fractional programming: Dinkelbach —; quasi-Newton —; random keys —; random search —; Rayleigh–Ritz —; reduced Hessian SQP —; reduction based —; reference direction —; regression —; regret —; relaxation —; response surface —; revised geometric mean —; Ritz–Galerkin —; Robbins–Monro —; Rodríguez —; rollout —; Rosen —; Rosen gradient projection —; Rosenbrock —; row-action —; Rudolph —; satisficing —; Schruben–Margolin —; score function —; separated Newton —; sequential —; sequential MEN synthesis —; Sequential simplex —; Shanno conjugate gradient —; shaped —; Simple recourse problem: dual —; Simple recourse problem: primal —; simplex —; single underlying —; smoothing Newton —; Solanki —; SOR —; splitting —; splitting Newton —; square-root —; SR1 quasi-Newton —; steepest descent —; steepest edge simplex —; stochastic counterpart —; stochastic search —; Stoica —; support problems solution —; supports problems —; symmetric rank-one quasi-Newton —; tensor —; Terlaky criss-cross —; topological —; truncated Newton —; trust region —; tunneling —; two-phase —; Two-stage stochastic programming: quasigradient —; univariate interval Newton —; UTA —; variable metric —; variable metric bundle —; vector maximization —; visual interactive —; Vogel approximation —; volumetric —; Wolfe reduced gradient —; Ziont criss-cross —
method and applications to optimization problems *see:* Laplace —

method of bad derivatives
[90C90]
(*see:* **Simulated annealing methods in protein folding**)
method-based
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
method of Broyden class *see:* quasi-Newton —
method of characteristics
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)
method of codifferential descent
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
method of codifferential descent
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
method for detecting redundancy *see:* deterministic —; probabilistic —
method in deterministic global optimization *see:* LP strategy for interval-Newton —
method of feasible directions *see:* combined —
method, global convergence, and Powell's conjecture *see:* Rosen's —
method of hypodifferential descent
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
method of hypodifferential descent
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
method for interhelical contacts in alpha-helical proteins *see:* Predictive —
method of least squares
[01A99]
(*see:* **Gauss, Carl Friedrich**)
method of least squares
[01A99]
(*see:* **Gauss, Carl Friedrich**)
method: Least squares, relation to Newton's method *see:* Gauss–Newton —
method for linear complementarity problems *see:* Splitting —
method for linear programs *see:* Selfdual parametric —
method of lines
[34-xx, 34Bxx, 34Lxx, 93E24]
(*see:* **Complexity and large-scale least squares problems**)
method of mappings
[49J20, 49J52]
(*see:* **Shape optimization**)
method for nonlinear programming *see:* feasible direction —
method of optimal distance
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
method of optimal ratio
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
method for the simple recourse problem *see:* dual —; primal —
method of simultaneous displacements
[90C30]
(*see:* **Cost approximation algorithms**)
method of steepest descent
[65K05, 90C30, 90C90]
(*see:* **Design optimization in computational fluid dynamics**; **Nondifferentiable optimization: minimax problems**)
method of successive displacements
[90C30]
(*see:* **Cost approximation algorithms**)
method for two-stage stochastic programs with recourse *see:* L-shaped —
method in unconstrained optimal control *see:* Dynamic programming and Newton's —
methodologies *see:* semantic analysis —; solution —
methodologies for auditing decisions *see:* Multicriteria decision support —
methodology *see:* NP-complete problems and proof —; OR- —; tabu search —; trust region —
methods *see:* ABS —; active set —; active set quadratic programming —; adaptive —; adjoint —; affine scaling SQPIP —; barrier —; basic outline of filled function —; Bayesian —; Bisection global optimization —; branch and bound —; Broyden —; Broyden family of —; bundle —; column generation —; combined relaxation —; complementary pivot —; computational —; computing processes in interactive —; Conjugate-gradient —; construction —; Credit rating and optimization —; cutting plane —; decomposition —; deflected gradient —; descent-based —; Disease diagnosis: optimization-based —; econometric —; ejection chain —; Entropy optimization: interior point —; evolutionary —; exact —; exact solution —; existence-proving properties of interval Newton —; extrapolation —; Extremum problems with probability functions: kernel type solution —; factorized quasi-Newton —; feasible direction —; filled function —; finite difference —; full space —; Gaussian approximation —; Global optimization: filled function —; Global optimization: hit and run —; Global terrain —; Globally convergent homotopy —; gradient —; heuristic —; hit and run —; homotopy —; hybrid —; hybrid NP —; improvement —; incomplete —; indirect —; inexact Newton —; Integer programming: algebraic —; Integer programming: branch and bound —; interactive —; interactive versus noninteractive —; interior point —; interval —; Interval analysis: subdivision directions in interval branch and bound —; interval Newton —; knowledge-based NP —; Krylov space type —; label correcting —; label setting —; limited enumeration —; line search —; Linear programming: interior point —; linearization —; MINLP: branch and bound —; MINLP: logic-based —; mixed —; Mixed integer programming/constraint programming hybrid —; Multi-scale global optimization using terrain/funneling —; Multicriteria sorting —; multicut —; multilevel —; multiplier —; Nondifferentiable optimization: cutting plane —; Nondifferentiable optimization: relaxation —; Nondifferentiable optimization: subgradient optimization —; Nonlinear least squares: Newton-type —; Nonlinear least squares: trust region —; nonsmooth —; numerical —; Outranking —; parallel —; parametric —; path following —; perturbation —; polyhedral —; polynomial time interior point —; posterior —; Practical augmented Lagrangian —; primal-dual —; primal-dual

interior-point —; primal-dual SQPIP —; Protein loop structure prediction —; proximal point —; quadrature —; qualitative forecasting —; quantitative forecasting —; quasi-Newton —; Quasidifferentiable optimization: exact penalty —; Random search —; regularity condition for penalty —; regularization of deterministic cutting plane —; Relaxation in projection —; Semi-infinite programming: approximation —; Semi-infinite programming: discretization —; Semi-infinite programming: numerical —; sequential quadratic programming —; Shape selective zeolite separation and catalysis: optimization —; smoothing —; software for homotopy —; solution —; Solving hemivariational inequalities by nonsmooth optimization —; Spectral projected gradient —; sQG —; sQG projection —; stochastic —; Stochastic global optimization: two-phase —; Stochastic optimal stopping: numerical —; stochastic quasigradient —; stochastic Quasigradient (SQG) —; Subgradient —; Successive quadratic programming: decomposition —; Successive quadratic programming: full space —; Successive quadratic programming: solution by active sets and interior point —; topological —; trust region —; unbounded controls and non standard —; variable metric —; Variable neighborhood search —; variational —
methods: applications *see:* Stochastic quasigradient —
methods and the BFGS update *see:* Broyden family of —
methods in clustering *see:* Assignment —
methods in complementarity theory *see:* Topological —
methods for convex programming *see:* Lagrangian multipliers —
methods for distributed optimal control problems *see:* Sequential quadratic programming: interior point —
methods of feasible directions
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
methods for global optimization *see:* Cutting plane —; Interval analysis: parallel —
methods for harmonic retrieval *see:* Global optimization —
methods for the linear complementarity problem *see:* Generalizations of interior point —
methods for linear complementarity problems *see:* Principal pivoting —
methods for linear problems *see:* Semi-infinite programming: —
methods for linear programming *see:* Homogeneous selfdual —; Potential reduction —
methods for mergers and acquisitions *see:* Multicriteria —
methods in minimax problems *see:* Stochastic quasigradient —
methods for multivalued variational inequalities *see:* Solution —
Methods for Non-Differentiable Functions and Applications *see:* minimization —
methods for non-smooth optimization *see:* Derivative-free —
methods for nonconvex feasibility analysis *see:* Shape reconstruction —
methods for nonlinear complementarity problems and variational inequalities *see:* Nonsmooth and smoothing —
methods for optimal design *see:* Multilevel —
methods for preference value functions *see:* Multi-objective optimization; Interactive —
methods in protein folding *see:* Simulated annealing —
methods in quadratic programming *see:* matrix splitting —
methods for reaction kinetics and transport *see:* Identification —
methods for semi-infinite optimization *see:* Smoothing —
methods for semidefinite programming *see:* Interior point —
methods for solving vehicle routing problems *see:* approximate —; constructive —; exact —
methods in supply chain management *see:* Mathematical programming —
methods for systems of nonlinear equations *see:* Global optimization —
methods for unary optimization *see:* Numerical —
metric
[90B50]
(*see:* **Inventory management in supply chains**)
metric *see:* C^k-Riemannian —; probability —; Riemannian —; Shahshahani —; variable —; w-weighted Tchebycheff —
metric-based
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
metric-based method
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
metric-based perspective
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
metric bundle method *see:* variable —
metric method *see:* variable —
metric methods *see:* variable —
metric projection
[41A30, 47A99, 49J52, 65K10, 90C30]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**; **Nondifferentiable optimization: Newton method**)
metric regularity
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
*metric regularity**
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
metric space
[90C35]
(*see:* **Multi-index transportation problems**)
metrically regular
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
Metropolis
[90C05, 90C25]
(*see:* **Metropolis, Nicholas Constantine**)
Metropolis criterion
[65K05, 90C30]
(*see:* **Random search methods**)
metropolis Monte Carlo method
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
Metropolis, Nicholas Constantine
(90C05, 90C25)

Metropolis process
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
MHD
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
MHEN
(93A30, 93B50)
(*referred to in:* **Continuous global optimization: models, algorithms and software**; **Global optimization of heat exchanger networks**; **Global optimization methods for systems of nonlinear equations**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**)
(*refers to:* **Global optimization of heat exchanger networks**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**)
MHEN
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**)
micro scale network
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
microcluster *see:* Lennard–Jones —; Morse —
microclusters *see:* size effects in —
microstate
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)
mid connective
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
the mid-point acceleration function
[25A15, 34A05, 90C25, 90C26, 90C30, 90C31]
(*see:* **Convexifiable functions, characterization of**)
mid-point acceleration function *see:* the —
middle set
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)
midpoint convexity *see:* discrete —
midpoint test
[65G20, 65G30, 65G40, 65H20, 65K05, 65Y05, 65Y10, 65Y20, 68W10, 90C30]
(*see:* **Interval analysis: parallel methods for global optimization**; **Interval analysis: unconstrained and constrained optimization**; **Interval global optimization**)
midpoint tests
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
migration *see:* multiclass —
migration equilibrium
[90C30]
(*see:* **Equilibrium networks**)
migration equilibrium
[90C30]
(*see:* **Equilibrium networks**)
migration network equilibrium model
[90C30]
(*see:* **Equilibrium networks**)
Milman theorem *see:* Krein– —
MILP
(*see:* **Logic-based outer approximation**)
MILP
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90, 93A30, 93B50]
(*see:* **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Modeling difficult optimization problems**; **Optimal planning of offshore oilfield infrastructure**)
MILP: lagrangian relaxation *see:* Decomposition techniques for —
MILP master problem
[49M20, 90C11, 90C30]
(*see:* **Generalized outer approximation**)
MILP model *see:* Gasoline blending and distribution scheduling: an —
MIN ant system *see:* MAX- —
min-cut theorem *see:* max-flow —
min duality *see:* double- —
min-exchange heuristic
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
min fractional program *see:* max- —
min-max digraph
[58E05, 90C30]
(*see:* **Topology of global optimization**)
min-max digraph
[58E05, 90C30]
(*see:* **Topology of global optimization**)
min-max fractional program
[90C32]
(*see:* **Fractional programming**)
min-max graph
[58E05, 90C30]
(*see:* **Topology of global optimization**)
min-max graph
[58E05, 90C30]
(*see:* **Topology of global optimization**)
min-max optimization problem *see:* max- —
min-max Steiner tree
[05C05, 05C85, 68Q25, 90B80]
(*see:* **Bottleneck steiner tree problems**)
min-type function
[90C26]
(*see:* **Global optimization: envelope representation**)
min-type function
[90C26]
(*see:* **Global optimization: envelope representation**)
mindiag fine structures
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
minima
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
minima *see:* local —; multiple —
minima problem in protein folding: αBB global optimization approach *see:* Multiple —

minimal
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
minimal *see:* locally —
minimal best approximation
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
minimal cone
[90C30]
(*see:* **Duality for semidefinite programming**)
minimal cut
[90C09, 90C10]
(*see:* **Oriented matroids**)
minimal dependent set
[90C09, 90C10]
(*see:* **Matroids**)
minimal function
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)
minimal function
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)
minimal number of DNF clauses
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
minimal principle
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)
minimal principle
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)
minimal representation
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
minimal representation of a feasible region
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
minimal runs
(*see:* **Planning in the process industry**)
minimal social cost *see:* production realizing with —
minimal tree *see:* Steiner —
minimal tree problem *see:* Steiner —
minimal value
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
minimal variance clustering
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
minimax
[65K05, 65K10, 90C06, 90C30, 90C34]
(*see:* **Feasible sequential quadratic programming**)
minimax
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
minimax algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
minimax approach *see:* Stochastic programming: —
minimax decision rule
[62C20, 90C15]
(*see:* **Stochastic programming: minimax approach**)
minimax decision rule
[62C20, 90C15]
(*see:* **Stochastic programming: minimax approach**)
Minimax: directional differentiability
(90C30, 65K05)
(*referred to in:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Minimax theorems**; **Nondifferentiable optimization: minimax problems**; **Stochastic programming: minimax approach**)
(*refers to:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Minimax theorems**; **Nondifferentiable optimization: minimax problems**; **Stochastic programming: minimax approach**; **Stochastic quasigradient methods in minimax problems**)
minimax game
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
minimax game tree algorithm *see:* sequential —
Minimax game tree searching
(49J35, 49K35, 62C20, 91A05, 91A40)
(*referred to in:* **Bottleneck steiner tree problems**; **Capacitated minimum spanning trees**; **Shortest path tree algorithms**)
(*refers to:* **Bottleneck steiner tree problems**; **Directed tree networks**; **Shortest path tree algorithms**)
minimax inequality *see:* two-function —
minimax location on a sphere
[90B85, 90C27]
(*see:* **Single facility location: circle covering problem**)
minimax objective
[90B80, 90B85]
(*see:* **Warehouse location problem**)
minimax objective function
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
minimax objective function *see:* lexicographically —
minimax observation problem
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
minimax observation problem under uncertainty with perturbations
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
minimax path length
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
minimax point *see:* saddle- —
minimax principles
[49J52]
(*see:* **Hemivariational inequalities: eigenvalue problems**)

minimax problem
[49K35, 49M27, 65K05, 65K10, 90C25, 90C30]
(*see:* **Convex max-functions**; **Nondifferentiable optimization: minimax problems**)
minimax problem
[49K35, 49M27, 65K05, 65K10, 90C25, 90C30]
(*see:* **Convex max-functions**; **Minimax: directional differentiability**; **Nondifferentiable optimization: minimax problems**)
minimax problem *see:* constrained —; finite —
minimax problems *see:* constrained —; Nondifferentiable optimization: —; Stochastic quasigradient methods in —
minimax solution
[62C20, 90C15]
(*see:* **Stochastic programming: minimax approach**)
minimax theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 90C27, 91A05]
(*see:* **Minimax theorems**; **Steiner tree problems**)
minimax theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
minimax theorem *see:* Du–Hwang —; mixed —; saddle- —
Minimax theorems
(46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05)
(*referred to in:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Minimax: directional differentiability**; **Nondifferentiable optimization: minimax problems**; **Stochastic programming: minimax approach**; **Stochastic quasigradient methods in minimax problems**)
(*refers to:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Minimax: directional differentiability**; **Nondifferentiable optimization: minimax problems**; **Stochastic programming: minimax approach**; **Stochastic quasigradient methods in minimax problems**)
minimax theory
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
minimax tree
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
minimax tree algorithm *see:* parallel —
minimax trees *see:* parallelizing the exploration of —
minimax value
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
minimization
[90C60]
(*see:* **Computational complexity theory**)
minimization *see:* algorithms for unconstrained —; concave —; constrained —; crossing —; energy —; error —; global —; gradient-free —; k-level crossing —; leveled crossing —; local —; nonconvex —; proximal —; unconstrained —; vector —
minimization algorithm *see:* gradient-free —
minimization algorithms *see:* SSC —; supervisor and searcher cooperation —
minimization algorithms for nonsmooth and stochastic optimization *see:* SSC —
minimization of cost/time
[90C32]
(*see:* **Fractional programming**)
minimization with D-functions *see:* proximal —
minimization of losses
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
minimization of makespan
(*see:* **Short-term scheduling of batch processes with resources**)
minimization Methods for Non-Differentiable Functions and Applications
[01A70, 90-03]
(*see:* **Shor, Naum Zuselevich**)
minimization of order earliness
(*see:* **Short-term scheduling of batch processes with resources**)
minimization of Pinkava normal forms
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
minimization problem
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
minimization problem *see:* constrained —
minimization problems *see:* decomposition algorithms for nonconvex —
minimization of regret
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
minimizer
[49K27, 90C05, 90C25, 90C29, 90C30, 90C31, 90C48]
(*see:* **Nondifferentiable optimization: parametric programming**; **Set-valued optimization**)
minimizer
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
minimizer *see:* ϵ- —; global —; isolated local —; local —; near- —; nonsingular local —; regular local —; strict local —; strong local —; weak —
minimizer problem *see:* local —
minimizers *see:* global —; local —
minimizing a convex multiplicative function *see:* program of —
minimizing the degradation in quality of both water environment
[90C30, 90C35]
(*see:* **Optimization in water resources**)
minimizing the energy function *see:* Optimization techniques for —
minimizing functions
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
minimizing a generalized convex function *see:* program of —
minimizing misclassifications
[90C29]
(*see:* **Multicriteria sorting methods**)
minimizing network cost
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**)

minimizing the overall classification error
[90C29]
(*see:* **Multicriteria sorting methods**)
minimizing a product of two affine functions *see:* program of —
minimizing q *see:* CG-standard for —
minimizing sequence
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
minimizing sequence *see:* generalized —; Levitin–Polyak —
minimum
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
minimum *see:* finding a —; global —; global energy —; local —; principle of —; relative —; strict local —; strict relative —; strong local —; strong relative —
minimum Bisection Problem
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
minimum circle
[90B85, 90C27]
(*see:* **Single facility location: circle covering problem**)
minimum clique partitioning
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
minimum composition difference
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: mass and heat exchanger networks**)
minimum concave transportation problem
[90B06, 90B10, 90C26, 90C35]
(*see:* **Minimum concave transportation problems**)
Minimum concave transportation problems
(90C26, 90C35, 90B06, 90B10)
(*referred to in:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Concave programming**; **Motzkin transposition theorem**; **Stochastic transportation and location problems**)
(*refers to:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Concave programming**; **Motzkin transposition theorem**; **Multi-index transportation problems**; **Stochastic transportation and location problems**)
minimum condition *see:* high-order local —
Minimum cost flow problem
(90C35)
(*referred to in:* **Auction algorithms**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Generalized networks**; **Maximum flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Nonoriented multicommodity flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
(*refers to:* **Auction algorithms**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Generalized networks**; **Maximum flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Nonoriented multicommodity flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
minimum cost flow problem
[05C05, 05C40, 68R10, 90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**; **Minimum cost flow problem**; **Network design problems**)
minimum cost flow problem
[90C35]
(*see:* **Minimum cost flow problem**)
minimum cost network flow
[90B10]
(*see:* **Piecewise linear network flow problems**)
minimum cost network flow problem
[90C35]
(*see:* **Generalized networks**)
minimum cost network flow problem *see:* piecewise linear —
minimum cost-to-time ratio cycle
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
minimum cross-entropy *see:* axiomatic derivation of the principle of —; principle of —
minimum cross-entropy principle
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
minimum cross-entropy principle
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
minimum cut
[90C35]
(*see:* **Maximum flow problem**)
minimum cut problem
[90C35]
(*see:* **Maximum flow problem**)
minimum cut problem
[90C35]
(*see:* **Maximum flow problem**)
minimum degree ordering
[65Fxx]
(*see:* **Least squares problems**)
minimum distance *see:* maximizing —
minimum feedback arc set problem
[90C35]
(*see:* **Feedback set problems**)
minimum feedback vertex (arc) set problem
[90C35]
(*see:* **Feedback set problems**)
minimum feedback vertex (arc) set problem *see:* subset —
minimum feedback vertex set
[90C35]
(*see:* **Feedback set problems**)
minimum fill-in of a graph
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)

minimum function
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
minimum KKT point *see:* global —
minimum lower set algorithm
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
minimum lower sets
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
minimum mean cycle
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
minimum Multiprocessor Scheduling Problem
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
minimum of an NNFP *see:* global —
minimum norm controllability
[93-XX]
(*see:* **Optimal control of a flexible arm**)
minimum norm solution
[90C11, 90C25, 90C33]
(*see:* **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**)
minimum norm solution
[90C11, 90C33]
(*see:* **LCP: Pardalos–Rosen mixed integer formulation**)
minimum number of clauses
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
minimum number of Steiner points *see:* Steiner tree problem with —
minimum Partition Problem (MP)
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
minimum path length
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
minimum phase
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
minimum point *see:* global —; local —; strict local —
minimum potential energy
[90C26, 90C90]
(*see:* **Global optimization in Lennard–Jones and morse clusters**)
minimum principle *see:* Pontryagin —
minimum problem *see:* accessory —
minimum rank completion problem
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
minimum ratio spanning-tree
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
minimum ratio test
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
minimum set
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
minimum solution *see:* global —
minimum spanning arborescence problem *see:* capacitated —
minimum spanning tree
[05C05, 05C40, 68R10, 90C27, 90C35]
(*see:* **Network design problems**; **Steiner tree problems**)
minimum spanning tree problem
[05C05, 05C40, 68R10, 68T99, 90C09, 90C10, 90C27, 90C35]
(*see:* **Capacitated minimum spanning trees**; **Matroids**; **Network design problems**)
minimum spanning tree problem *see:* bounded degree —; capacitated —; resource-constrained —
minimum spanning trees
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
minimum spanning trees *see:* Capacitated —
minimum sphere
[90B85, 90C27]
(*see:* **Single facility location: circle covering problem**)
minimum sphere problem
[90B85, 90C27]
(*see:* **Single facility location: circle covering problem**)
minimum Steiner arborescence
[90C27]
(*see:* **Steiner tree problems**)
minimum tree *see:* Steiner —
minimum unfeasibility criterion
[90C10, 90C29]
(*see:* **Multi-objective integer linear programming**)
minimum-units problem
[90C90]
(*see:* **Mixed integer linear programming: heat exchanger network synthesis**)
minimum value *see:* positive —
minimum Vertex Cover Problem
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
minimum-volume ellipsoid
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
minimum weight common mutated sequence
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
minimum weight feedback arc set problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
minimum weight Steiner triangulation
[68Q20]
(*see:* **Optimal triangulations**)
minimum weight triangulation
[68Q20]
(*see:* **Optimal triangulations**)
minimum weighted feedback vertex set problem
[90C35]
(*see:* **Feedback set problems**)
minimum weighted graph bipartization problem
[90C35]
(*see:* **Feedback set problems**)
minimum weighted vertex cover
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
mining *see:* Data —; Mathematical programming for data —

minisum
[65D18, 90B85, 90C26]
(*see:* **Global optimization in location problems**)
minisum problems
[90B80, 90B85]
(*see:* **Warehouse location problem**)
minkowski distances
[65K05, 90C27, 90C30, 90C57, 91C15]
(*see:* **Optimization-based visualization**)
Minkowski duality
[90C26]
(*see:* **Global optimization: envelope representation**)
Minkowski plane
[90C27]
(*see:* **Steiner tree problems**)
Minkowski sum
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
MINLP
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
MINLP
[90C06, 90C10, 90C11, 90C26, 90C30, 90C57, 90C90, 93A30, 93B50]
(*see:* **MINLP: application in facility location-allocation**; **MINLP: branch and bound methods**; **MINLP: heat exchanger network synthesis**; **MINLP: mass and heat exchanger networks**; **MINLP: reactive distillation column synthesis**; **Modeling difficult optimization problems**; **Optimal planning of offshore oilfield infrastructure**)
MINLP *see:* challenges in —; convex —; global —; HEN synthesis using —; local —; nonconvex —
MINLP: application in facility location-allocation
(90C26)
(*referred to in:* **Chemical process planning**; **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Chemical process planning**; **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Extended cutting plane algorithm**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
MINLP: applications in blending and pooling problems
(90C90, 90C30)
(*referred to in:* **Chemical process planning**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)
(*refers to:* **Chemical process planning**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)

MINLP: applications in the interaction of design and control (90C11, 49M37)
(*referred to in:* **Chemical process planning**; **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Generalized benders decomposition**; **Generalized outer approximation**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Chemical process planning**; **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)

MINLP: branch and bound global optimization algorithm (90C10, 90C26)
(*referred to in:* **αBB algorithm**; **Chemical process planning**; **Continuous global optimization: models, algorithms and software**; **Disjunctive programming**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization in batch design under uncertainty**; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval analysis: subdivision directions in interval branch and bound methods**; **Interval global optimization**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Smooth nonlinear nonconvex optimization**)
(*refers to:* **αBB algorithm**; **Chemical process planning**; **Continuous global optimization: models, algorithms and software**; **Disjunctive programming**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization in batch design under uncertainty**; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval global optimization**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Reformulation-linearization technique for global optimization**; **Smooth nonlinear nonconvex optimization**)

MINLP: branch and bound methods (90C11)
(*referred to in:* **Chemical process planning**; **Disjunctive programming**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming:**

mass and heat exchanger networks; **Mixed integer nonlinear programming**)
(*refers to:* **Chemical process planning**; **Disjunctive programming**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Reformulation-linearization technique for global optimization**)

MINLP: design and scheduling of batch processes
(90C26)
(*referred to in:* **Chemical process planning**; **Generalized benders decomposition**; **Generalized outer approximation**; **Job-shop scheduling problem**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Vehicle scheduling**)
(*refers to:* **Chemical process planning**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **Job-shop scheduling problem**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Stochastic scheduling**; **Vehicle scheduling**)

MINLP: generalized cross decomposition
(90C11, 90C30, 49M27)
(*referred to in:* **Chemical process planning**; **Decomposition principle of linear programming**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)
(*refers to:* **Chemical process planning**; **Decomposition principle of linear programming**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)

MINLP: global optimization with αBB
(65K05, 90C11, 90C26)
(*referred to in:* **αBB algorithm**; **Chemical process planning**; **Convex envelopes in optimization problems**; **Disjunctive programming**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization in batch design under uncertainty**; **Global optimization in generalized geometric programming**; **Global optimization of heat exchanger networks**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval global optimization**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: mass and heat exchanger networks**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Smooth nonlinear nonconvex optimization**)
(*refers to:* **αBB algorithm**; **Chemical process planning**; **Continuous global optimization: models, algorithms and software**; **Convex envelopes in optimization problems**;

Disjunctive programming; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization in batch design under uncertainty**; **Global optimization in generalized geometric programming**; **Global optimization of heat exchanger networks**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval global optimization**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: mass and heat exchanger networks**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Reformulation-linearization technique for global optimization**; **Smooth nonlinear nonconvex optimization**)

MINLP: heat exchanger network synthesis
(90C90)
(*referred to in:* **Chemical process planning**; **Continuous global optimization: models, algorithms and software**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization of heat exchanger networks**; **Global optimization methods for systems of nonlinear equations**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: logic-based methods**; **MINLP: mass and heat exchanger networks**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)
(*refers to:* **Chemical process planning**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization of heat exchanger networks**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: logic-based methods**; **MINLP: mass and heat exchanger networks**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)

MINLP: logic-based methods
(90C10, 90C09, 90C11)
(*referred to in:* **Chemical process planning**; **Decomposition principle of linear programming**; **Disjunctive programming**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)
(*refers to:* **Chemical process planning**; **Decomposition principle of linear programming**; **Disjunctive programming**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**; **Reformulation-linearization technique for global optimization**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)

MINLP: mass and heat exchanger networks
(93A30, 93B50)
(*referred to in:* **Continuous global optimization: models, algorithms and software**; **Global optimization of heat exchanger networks**; **Global optimization methods for systems of nonlinear equations**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**)
(*refers to:* **Global optimization of heat exchanger networks**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**)

MINLP MEN synthesis model *see:* multiperiod —

MINLP: outer approximation algorithm
(90C11)
(*referred to in:* **Chemical process planning**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)
(*refers to:* **Chemical process planning**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)
MINLP: reactive distillation column synthesis
(90C90)
(*referred to in:* **Chemical process planning**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)
(*refers to:* **Chemical process planning**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)
MINLP: trim-loss problem
(90C11, 90C90)
(*referred to in:* **Mixed integer nonlinear programming**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
mINLPs
[65K05, 90C11, 90C26]
(*see:* **MINLP: global optimization with αBB**)
MINLPs *see:* twice-differentiable —
minmax, concave programs *see:* Bilevel linear programming: complexity, equivalence to —
minMax Matching Subgraph Problem
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
minmax multicenter
[05C05, 05C85, 68Q25, 90B80]
(*see:* **Bottleneck steiner tree problems**)
minmax problem
[49-01, 49K45, 49N10, 90-01, 90C20, 90C27, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**)
minmax problem
[49-01, 49K45, 49N10, 90-01, 90C20, 90C27, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**)
minor *see:* matroid —
minor closed
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)
minor of a matroid
[90C09, 90C10]
(*see:* **Matroids**)
minorant *see:* greatest convex —; greatest K- —; greatest quasiconvex —
minority
[90-XX]
(*see:* **Outranking methods**)
MINOS
[49M37, 65K05, 90C30]
(*see:* **Inequality-constrained nonlinear optimization**)
mintasks
(*see:* **Medium-term scheduling of batch processes**)

Minty examples *see:* Klee– —; Linear programming: Klee– — Minty) GVI *see:* dual (or —
MinxEnt
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
MIP
[65K05, 90C09, 90C10, 90C11, 90C20]
(*see:* **Disjunctive programming**; **Multi-quadratic integer programming: models and applications**)
mipstart
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
MIQP
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
MIQP master problem
[49M20, 90C11, 90C30]
(*see:* **Generalized outer approximation**)
Miranda fixed point theorem
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval fixed point theory**)
mIS
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
misclassifications *see:* minimizing —
miss" decision problems *see:* "hit-or- —
missing comparisons
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
missing information
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
missing information
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
Mitchell PTAS
[90C27]
(*see:* **Steiner tree problems**)
mitigator
[65K05, 90C26, 90C30, 90C59]
(*see:* **Global optimization: filled function methods**)
MITP
[90C35]
(*see:* **Multi-index transportation problems**)
MITPs *see:* greedy algorithm for axial —; hub heuristics for axial —; integer —
Mixed 0-1 linear programming approach for DNA transcription element identification
(90C08)
mixed complementarity problem *see:* generalized —
mixed construction procedure
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
mixed continuous and discrete free variables *see:* Generalized geometric programming: —
mixed cycle
[90C35]
(*see:* **Optimization in leveled graphs**)
mixed discrete-continuous global optimization
[90C26, 90C29]
(*see:* **Optimal design of composite structures**)
mixed discrete-continuous global optimization
[90C26, 90C29, 90C90]
(*see:* **Global optimization: hit and run methods**; **Optimal design of composite structures**)
mixed finite element
[65M60]
(*see:* **Variational inequalities: F. E. approach**)
mixed finite element approximation
[65M60]
(*see:* **Variational inequalities: F. E. approach**)
mixed fleet
[90B06]
(*see:* **Vehicle routing**)
mixed graph
[90B35]
(*see:* **Job-shop scheduling problem**)
mixed integer
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
mixed integer 0–1 programs
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
mixed integer αBB algorithm *see:* general structure —; special structure —
Mixed Integer Bilevel Optimization
(*see:* **Mixed integer nonlinear bilevel programming: deterministic global optimization**)
Mixed integer classification problems
(62H30,68T10,90C11)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Linear programming models for classification**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Statistical classification: optimization approaches**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Deterministic and probabilistic optimization models for data classification**; **Integer programming**; **Linear programming models for classification**; **Optimization in boolean classification problems**; **Statistical classification: optimization approaches**)

mixed integer dynamic optimization
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
mixed integer formulation *see:* LCP: Pardalos–Rosen —
mixed-integer linear optimization *see:* Global pairwise protein sequence alignment via —
mixed integer linear program *see:* single parametric —
mixed integer/linear programming
[90C06, 90C10, 90C11, 90C30, 90C46, 90C57, 90C90]
(*see:* **Integer programming duality**; **Modeling difficult optimization problems**)
mixed integer linear programming
[90C90]
(*see:* **Chemical process planning**)
mixed integer linear programming *see:* Multiparametric —
Mixed integer linear programming: heat exchanger network synthesis
(90C90)
(*referred to in:* **Continuous global optimization: models, algorithms and software**; **Global optimization of heat exchanger networks**; **Global optimization methods for systems of nonlinear equations**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: mass and heat exchanger networks**)
(*refers to:* **Chemical process planning**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization of heat exchanger networks**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **Mixed integer nonlinear programming**)
Mixed integer linear programming: mass and heat exchanger networks
(93A30, 93B50)
(*referred to in:* **Chemical process planning**; **Continuous global optimization: models, algorithms and software**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization of heat exchanger networks**; **Global optimization methods for systems of nonlinear equations**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: mass and heat exchanger networks**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer nonlinear programming**)
(*refers to:* **Chemical process planning**; **Extended cutting plane algorithm**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization of heat exchanger networks**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: mass and heat exchanger networks**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer nonlinear programming**)
mixed-integer linear programs *see:* Robust optimization: —
mixed integer nonconvex problem
[65K05, 90C11, 90C26]
(*see:* **MINLP: global optimization with αBB**)
Mixed integer nonlinear bilevel programming: deterministic global optimization
mixed integer nonlinear optimization
[49M29, 49M37, 90C11, 90C26]
(*see:* **Generalized benders decomposition**; **Global optimization in batch design under uncertainty**; **Mixed integer nonlinear programming**)
mixed integer nonlinear optimization
[49M37, 90C11, 90C29, 90C90]
(*see:* **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**)
mixed integer nonlinear optimization *see:* Parametric —
Mixed-integer nonlinear optimization: A disjunctive cutting plane approach
(49M37, 90C11)
mixed integer nonlinear program
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)
Mixed integer nonlinear programming
(90C11, 49M37)
(*referred to in:* **Chemical process planning**; **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Continuous global optimization: applications**; **Fractional combinatorial optimization**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization in the analysis and management of environmental systems**; **Information-based complexity and information-based optimization**; **Interval global optimization**; **Kolmogorov complexity**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch**

processes; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**)
(*refers to:* **Chemical process planning**; **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Continuous global optimization: applications**; **Extended cutting plane algorithm**; **Fractional combinatorial optimization**; **Generalized benders decomposition**; **Generalized outer approximation**; **Global optimization in the analysis and management of environmental systems**; **Information-based complexity and information-based optimization**; **Interval global optimization**; **Kolmogorov complexity**; **MINLP: application in facility location-allocation**; **MINLP: applications in blending and pooling problems**; **MINLP: applications in the interaction of design and control**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **MINLP: generalized cross decomposition**; **MINLP: global optimization with αBB**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **MINLP: reactive distillation column synthesis**; **MINLP: trim-loss problem**; **Mixed integer linear programming: mass and heat exchanger networks**; **Parallel computing: complexity classes**)
mixed integer nonlinear programming
[49M20, 49M37, 90C06, 90C10, 90C11, 90C26, 90C29, 90C30, 90C57, 90C90]
(*see:* **Generalized outer approximation**; **MINLP: branch and bound global optimization algorithm**; **Mixed integer nonlinear programming**; **Modeling difficult optimization problems**; **Multi-objective optimization: interaction of design and control**)
mixed integer nonlinear programming
[49M20, 90C10, 90C11, 90C26, 90C30]
(*see:* **Generalized outer approximation**; **MINLP: branch and bound global optimization algorithm**; **MINLP: outer approximation algorithm**)
mixed integer nonlinear programming problem
[90C11, 90C90]
(*see:* **MINLP: branch and bound methods**; **Mixed integer linear programming: heat exchanger network synthesis**)
mixed integer optimal control problem
[49M37, 90C11, 90C29, 90C90]
(*see:* **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**)
Mixed Integer Optimization
(*see:* **Mixed integer nonlinear bilevel programming: deterministic global optimization**)
mixed-integer optimization *see:* Multi-class data classification via —; Peptide identification via —
mixed integer optimization problem
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
Mixed integer optimization in well scheduling
(76T30, 90C11, 90C90)
mixed integer problem
[90C11, 90C33]
(*see:* **LCP: Pardalos–Rosen mixed integer formulation**)
mixed integer problem
[90C11, 90C33]
(*see:* **LCP: Pardalos–Rosen mixed integer formulation**)
mixed integer problems *see:* 0–1 —; linear —
mixed integer program
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
Mixed-integer programming
[90B50, 90C29]
(*see:* **Logic-based outer approximation**; **Multicriteria sorting methods**; **Optimization and decision support systems**)
mixed integer programming
[90B50, 90B80, 90C09, 90C10, 90C11, 90C33]
(*see:* **Facility location with staircase costs**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: branch and bound methods**; **MINLP: logic-based methods**; **Mixed integer nonlinear bilevel programming: deterministic global optimization**; **Optimal planning of offshore oilfield infrastructure**; **Optimization and decision support systems**; **Railroad crew scheduling**; **Railroad locomotive scheduling**)
mixed integer programming *see:* Multi-objective —; multi-objective (multicriteria) —; stochastic —
Mixed integer programming/constraint programming hybrid methods
(*referred to in:* **Continuous reformulations of discrete-continuous optimization problems**)
mixed integer programming problem *see:* large scale nonlinear —
mixed integer programs
[90C11, 90C59]
(*see:* **Nested partitions optimization**)
mixed-integer quadratic programming
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
mixed integer rounding cut
[90C11]
(*see:* **MINLP: branch and bound methods**)
mixed integer value function
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)
mixed linear complementarity problem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
mixed methods
[90B06]
(*see:* **Vehicle routing**)
mixed minimax theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)

mixed nonlinear integer programming problem
[65K05, 90C26, 90C30, 90C59]
(*see:* **Global optimization: filled function methods**)
mixed-product campaign
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
mixed Time Representation
(*see:* **Integrated planning and scheduling**)
mixed VAM construction procedure
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
mixed variational formulation
[65M60]
(*see:* **Variational inequalities: F. E. approach**)
mixed variational inequality
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
mixture problem *see:* fuel —
MLP
[90C15]
(*see:* **L-shaped method for two-stage stochastic programs with recourse**)
MLS
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
MMP
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
mobilization curve
(*see:* **Emergency evacuation, optimization modeling**)
MOCO
[90C10, 90C35]
(*see:* **Bi-objective assignment problem**)
mode
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
mode *see:* batch —; forward —; reverse —
mode of AD *see:* forward —; reverse —
mode of an AD algorithm *see:* forward —; reverse —
mode of automatic differentiation *see:* backward —; forward —; reverse —
model
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
model *see:* assignment —; black oil —; Black–Scholes —; bridging —; BSP —; bulk synchronous parallel —; capital asset pricing —; classical linear regression —; classical thermoelastic —; computational —; containment graph —; continuous global optimization —; continuous review —; continuous Time —; convex —; Cournot–Nash oligopolistic equilibrium —; cutting plane —; deformable —; deterministic equivalent —; diagonal —; discrete Time —; dynamic —; dynamic traffic network —; energy —; epidemic —; errors-in-variables —; expanded transshipment —; export —; facility location —; flipping —; forecasting —; fractional routing pattern —; Gasoline blending and distribution scheduling: an MILP —; general univariate linear —; gIS design pattern based —; heteroscedastic —; homogeneous and selfdual —; homoscedastic —; hybrid —; import —; information-based —; intersection graph —; Ising glass —; linear —; linear two-stage —; location-allocation —; logistics control —; LogP —; matching —; mathematical —; migration network equilibrium —; moving average —; the multi-resource weighted assignment —; multi-sector multi-instrument financial equilibrium —; multimodal traffic network equilibrium —; multiperiod —; multiperiod MINLP MEN synthesis —; network flow —; newsboy —; oligopoly —; parametric programming —; partial equilibrium —; perfectly competitive equilibrium —; periodic review —; plant location —; Portfolio selection: markowitz mean-variance —; Potts glass —; price —; proximity graph —; pure exchange economic equilibrium —; pure trade economic equilibrium —; QSM —; quantity —; quasi-assignment —; queueing shared-memory —; Ramsey —; real number —; recourse —; reduced —; relational —; right-hand side perturbation —; rotation-symmetry —; Sharpe single index market —; sign-invariance —; single path routing pattern —; single-period —; spatial competition facility location —; spatial-interaction —; spatial oligopoly —; static —; stochastic —; Stochastic facility location —; superstructure —; transshipment —; trust region —; Turing machine —; vector space —; well bore —; Wiener —
model B
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
model-based
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
Model based control for drug delivery systems
(*refers to:* **Nondifferentiable optimization: parametric programming**)
model-based controllers via parametric programming *see:* Design of robust —
model-based experimental analysis
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
model-based method
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
model-based perspective
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
model BF
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
model BFR
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)

model building
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
model calibration
[90C05]
(*see:* **Global optimization in the analysis and management of environmental systems**)
model of computation
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
model of conflicting populations *see:* Volterra —
model coordination method
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
model development
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
model features *see:* special —
model finding procedure
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
model and Gibbs sampler *see:* hidden Markov —
model identification
[62F10, 94A17]
(*see:* **Entropy optimization: parameter estimation**)
model with impulse perturbations *see:* Vasicek —
model independent
[65H20, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**)
model nodes *see:* plant/ —; retailer/ —
model/optimizer coupling
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
model in OR
[90B80, 90B85]
(*see:* **Warehouse location problem**)
model in OR *see:* continuous —; discrete —; multicommodity —; single-commodity —
model for parallel algorithm design
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
model parameters *see:* estimation of —
model predictive control
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
model refinement *see:* iterative —
model reformulation
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
model robust
[90C90, 91B28]
(*see:* **Robust optimization**)
model structure refinement *see:* incremental strategy for —
model structures
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
model types
[90C05]
(*see:* **Continuous global optimization: models, algorithms and software**)
model validation
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
model world
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
modeling
[90C06, 90C10, 90C11, 90C27, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**; **Operations research and financial markets**)
modeling *see:* conceptual —; Emergency evacuation, optimization —; energy —; mathematical —; nonsmooth —; preference —; problem —; uncertainty —
modeling agricultural systems *see:* State of the art in —
Modeling difficult optimization problems
(90C06, 90C10, 90C11, 90C30, 90C57, 90C90)
modeling framework *see:* multiperiod optimization —
modeling frameworks *see:* Short-term scheduling, resource constrained: unified —
modeling language
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
modeling language
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
modeling language *see:* algebraic —
modeling language and constraint logic programming
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
modeling languages *see:* algebraic —; second generation —
Modeling languages in optimization: a new paradigm
(90C10, 90C30)
(*referred to in:* **Continuous global optimization: models, algorithms and software**; **Large scale unconstrained optimization**; **Optimization software**)
(*refers to:* **Continuous global optimization: models, algorithms and software**; **Large scale unconstrained optimization**; **Optimization software**)
modeling and management *see:* applications in environmental systems —
modeling mass exchange
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**)
modeling production
(*see:* **Planning in the process industry**)
modello
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
modellus
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)

models *see:* complexity of —; compositional —; continuous and discrete time —; discrete-time —; econometric —; estimation of diffusion flux —; Global optimization based on statistical —; Global supply chain —; hidden Markov —; location-routing —; locomotive assignment —; mathematical —; mechanical —; multiple locomotive type —; multipopulation replicator —; multistage inventory management —; nonlinear decision —; Parallel computing: —; parametric programming —; positive definite quadratic —; purpose of —; recourse —; representation of —; restricted recourse —; simulation —; simultaneous equation —; single locomotive scheduling —; single stage inventory management —; single versus Multiperiod —; Static stochastic programming —; statistical —; strategic design —; supply chain simulation —; thermodynamic —; time-stamped —; transportation —; two-stage stochastic programming —; undirected multicommodity network flow —
models, algorithms and software *see:* Continuous global optimization: —
models and applications *see:* Multi-quadratic integer programming: —
models for classification *see:* Linear programming —
models: conditional expectations *see:* Static stochastic programming —
models for data classification *see:* Deterministic and probabilistic optimization —
models for entropy optimization for image reconstruction *see:* finite-dimensional —; vector-space —
models for parallel computing
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
models: (*QR*) policy *see:* Continuous review inventory —
models: random objective *see:* Stochastic programming —
models for supply chain management and design *see:* Operations research —
modified
(*see:* **Emergency evacuation, optimization modeling**)
modified Cauchy approach
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
modified Cauchy method
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
modified Gram–Schmidt orthogonalization
[65Fxx]
(*see:* **Least squares problems**)
modified Huang algorithm
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**)
modified Kruskal algorithm
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
modified Lagrangian
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
modified Newton method
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
modified Prim algorithm
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
modified square-root transformation
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
modified standard auction algorithm
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
modifying matrix factorization
[65Fxx]
(*see:* **Least squares problems**)
MODP
[90C31, 90C39]
(*see:* **Multiple objective dynamic programming**)
MODP *see:* principle of Pareto optimality of —
modular
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
modular
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
modular approach
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
module *see:* mass/heat transfer —
modus confirmans
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
modus negans
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
modus ponens
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
modus ponens *see:* checklist —
modus tollens
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
modus tollens *see:* checklist —
Moebius function
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
MOILP
[90C27, 90C29]
(*see:* **Multi-objective combinatorial optimization**)
molar Gibbs free energy
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
molecular conformation
[65D18, 90B85, 90C26]
(*see:* **Global optimization in location problems**)

molecular design
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
Molecular distance geometry problem
(46N60)
molecular dynamics
[90C90]
(*see:* **Simulated annealing methods in protein folding**)
molecular mechanics
[65K10, 92C40]
(*see:* **Multiple minima problem in protein folding: αBB global optimization approach**)
molecular optimization
[90C90]
(*see:* **Simulated annealing methods in protein folding**)
molecular structure determination
[65K05, 90C26]
(*see:* **Molecular structure determination: convex global underestimation**)
Molecular structure determination: convex global underestimation
(65K05, 90C26)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms**; **Global optimization in Lennard–Jones and morse clusters**; **Graph coloring**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Simulated annealing methods in protein folding**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms**; **Global optimization in Lennard–Jones and morse clusters**; Global optimization in protein folding; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Protein folding: generalized-ensemble algorithms**; **Simulated annealing**; **Simulated annealing methods in protein folding**)
MOLFP
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
mollifier *see:* standard —
mollifier quasigradient *see:* stochastic —
MOLP with fuzzy coefficients
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
MOLP with fuzzy coefficients *see:* flexible —
moment *see:* dipole —
moment conditions
[62C20, 90C15]
(*see:* **Stochastic programming: minimax approach**)
moment conditions *see:* optimal integral bounds subject to —
moment optimization problems *see:* General —
moment problem *see:* convex —; finite —; infinite —; infinite many conditions —; solution of the convex —; standard —
moment theory
[93-XX]
(*see:* **Optimal control of a flexible arm**)
moment theory *see:* geometric —
moments *see:* binomial —
momentum balances *see:* mass, energy and —
momentum updating rule
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
MOMILP
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
MOMIP
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
momments *see:* characterizing —
monads
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
Mond–Weir dual
[90C26]
(*see:* **Invexity and its applications**)
Mond–Weir dual
[90C26]
(*see:* **Invexity and its applications**)
Monge inequalities
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Monge inequalities *see:* anti- —
Monge matrix
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Monge matrix *see:* anti- —
Monge property
[90C35]
(*see:* **Multi-index transportation problems**)
Monge property
[90C35]
(*see:* **Multi-index transportation problems**)
monitored
(*see:* **Emergency evacuation, optimization modeling**)
monoduality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
monoduality in convex optimization *see:* Duality theory: —
monomial
[12D10, 12Y05, 13P10]
(*see:* **Gröbner bases for polynomial equations**)
monomial ideal
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
monomial ideal *see:* admissible pair of a —; arithmetic degree of a —; standard pair of a —; standard pair decomposition of a —
monomials *see:* posynomial —; standard —
monopoly
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
monotone
[46N10, 47J20, 49J40, 49L20, 65K10, 90C26, 90C33, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**; **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**; **Generalized monotonicity: applications to variational inequalities and**

equilibrium problems; **Solution methods for multivalued variational inequalities**)
monotone *see:* strictly —; strongly —
monotone bifunction
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
monotone Boolean function
[90C09]
(*see:* **Inference of monotone boolean functions**)
monotone Boolean function
[90C09]
(*see:* **Inference of monotone boolean functions**)
monotone Boolean function *see:* antitone —; isotone —; nondecreasing —; nonincreasing —
monotone Boolean function inference
[90C09]
(*see:* **Inference of monotone boolean functions**)
monotone boolean functions *see:* Inference of —
monotone convergence theorem
[49L20, 90C40]
(*see:* **Dynamic programming: undiscounted problems**)
monotone function
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
monotone function *see:* locally —; locally strictly —; locally strongly —; strictly —; strongly —
monotone laws and variational inequalities *see:* multivalued —
monotone map
[90C26]
(*see:* **Generalized monotone single valued maps**)
monotone map *see:* maximal —; strictly —
monotone matrix
[90C33]
(*see:* **Linear complementarity problem**)
monotone multivalued maps *see:* Generalized —
monotone operator
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
monotone operator *see:* generalized —; strictly —
monotone operator on a Banach space
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
monotone sequence *see:* Fejér —
monotone sequence of greedy swaps
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
monotone single valued maps *see:* Generalized —
monotonic
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
monotonic analysis
[26A48, 26A51, 52A07]
(*see:* **Increasing and convex-along-rays functions on topological vector spaces**)
monotonic at *see:* locally strongly —
monotonic function
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
monotonic functions *see:* difference of —
Monotonic optimization
(90C26, 65K05, 90C30)
monotonic optimization
[65K05, 90C26, 90C30, 90C31]
(*see:* **Cutting plane methods for global optimization**; **D.C. programming**; **Monotonic optimization**; **Robust global optimization**)
monotonic optimization
[90C26]
(*see:* **Cutting plane methods for global optimization**)
monotonic optimization problem *see:* canonical —
monotonic optimization problems *see:* discrete —
monotonic over *see:* strongly linearly —
monotonicity
[65K10, 65M60, 90C09, 90C10, 90C26, 90C30]
(*see:* **Bounding derivative ranges**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Variational inequalities: geometric interpretation, existence and uniqueness**)
monotonicity
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
monotonicity *see:* Fejér —; generalized —; local —; local strict —; local strong —; partial —; strict —; strong —
monotonicity: applications to variational inequalities and equilibrium problems *see:* Generalized —
monotonicity in convex optimization *see:* Fejér —
monotonicity and nonconvexity test
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
monotonicity test
[49M37, 65G20, 65G30, 65G40, 65H20, 65K05, 65Y05, 65Y10, 65Y20, 68W10, 90C11, 90C30]
(*see:* **Interval analysis: parallel methods for global optimization**; **Interval analysis: unconstrained and constrained optimization**; **Interval global optimization**; **Mixed integer nonlinear programming**)
monotonous *see:* partially —
Monro method *see:* Robbins– —
Monte-Carlo
[65K10, 92C40]
(*see:* **Multiple minima problem in protein folding: αBB global optimization approach**)
Monte-Carlo *see:* pure —
Monte-Carlo configuration
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
Monte-Carlo method
[90C05, 90C25, 90C90]
(*see:* **Metropolis, Nicholas Constantine**; **Simulated annealing methods in protein folding**)
Monte-Carlo method
[62F12, 65C05, 65K05, 90C05, 90C15, 90C25, 90C31]
(*see:* **Metropolis, Nicholas Constantine**; **Monte-Carlo simulations for stochastic optimization**)
Monte Carlo method *see:* metropolis —; pure —

Monte-Carlo sampling and variance reduction
[90C27]
(*see:* **Operations research and financial markets**)
Monte-Carlo simulated annealing in protein folding
(92C40)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Genetic algorithms**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization in Lennard–Jones and morse clusters**; **Graph coloring**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulations for stochastic optimization**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Genetic algorithms**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization in Lennard–Jones and morse clusters**; Global optimization in protein folding; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulations for stochastic optimization**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Protein folding: generalized-ensemble algorithms**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
monte-Carlo simulation
[49L20, 49L99, 62F12, 65C05, 65K05, 90C15, 90C31, 90C40]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: stochastic shortest path problems**; **Monte-Carlo simulations for stochastic optimization**)
Monte-Carlo simulation algorithm
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
Monte-Carlo simulation procedure
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
Monte-Carlo simulations for stochastic optimization
(90C15, 65C05, 65K05, 90C31, 62F12)
(*referred to in:* **Monte-Carlo simulated annealing in protein folding**)
(*refers to:* **Monte-Carlo simulated annealing in protein folding**)
mood of play
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
Moon algorithm *see:* Corley– —
Moore–Penrose pseudo-inverse
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
Moré updating formula
[90C30]
(*see:* **Generalized total least squares**)
Moreau duality *see:* Fenchel– —
Moreau–Rockafellar subdifferential
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
Moreau subdifferential *see:* Fenchel– —
Moreau theorem
[90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**)
morphism *see:* generalized —
morphisms *see:* generalized —
morphisms of relations *see:* generalized —
Morrison formula *see:* Sherman– —
Morrison rank-one update formula *see:* Sherman- —
Morrison–Woodbury formula *see:* Sherman– —
morse clusters *see:* Global optimization in Lennard–Jones and —
morse index
[57R12, 90C31, 90C34]
(*see:* **Smoothing methods for semi-infinite optimization**)
morse indices
[57R12, 90C31, 90C34]
(*see:* **Smoothing methods for semi-infinite optimization**)
Morse lemma *see:* equivariant —
Morse microcluster
[90C26, 90C90]
(*see:* **Global optimization in Lennard–Jones and morse clusters**)
Morse relations
[58E05, 90C30]
(*see:* **Topology of global optimization**)
Morse relations
[58E05, 90C30]
(*see:* **Topology of global optimization**)
Morse theory
[58E05, 90C30]
(*see:* **Topology of global optimization**)
MOSA method
[90C27, 90C29]
(*see:* **Multi-objective combinatorial optimization**)
Mosco convergence *see:* discrete —
most active points *see:* set of ε- —
most/least infeasible integer variable
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
most preferred solution
[90C29]
(*see:* **Multiple objective programming support**)
most promising region
[90C11, 90C59]
(*see:* **Nested partitions optimization**)
mostPreferred
(*see:* **Railroad locomotive scheduling**)
motion
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)

motion
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
motion *see:* Brownian —; N-dimensional Brownian —
motion of a point
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
motion of a set
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
motionless degree
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
motivation
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**)
Motzkin elimination *see:* Fourier– —
Motzkin elimination method *see:* Fourier– —
Motzkin–Fourier relaxation method *see:* Agmon– —
Motzkin method *see:* Fourier– —
Motzkin theorem
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
Motzkin transposition theorem
(15A39, 90C05)
(*referred to in:* **Farkas lemma**; **Linear optimization: theorems of the alternative**; **Linear programming**; **Minimum concave transportation problems**; **Stochastic transportation and location problems**; **Tucker homogeneous systems of linear relations**)
(*refers to:* **Farkas lemma**; **Linear optimization: theorems of the alternative**; **Linear programming**; **Minimum concave transportation problems**; **Multi-index transportation problems**; **Stochastic transportation and location problems**; **Tucker homogeneous systems of linear relations**)
Motzkin transposition theorem
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
mountain pass theorem
[49J52, 58E05, 90C30]
(*see:* **Hemivariational inequalities: eigenvalue problems**; **Topology of global optimization**)
move
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
move *see:* exchange —; feasible —; shift —
move in a search
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
move of a Turing machine
[90C60]
(*see:* **Complexity theory**)
moving average model
[90C26, 90C30]
(*see:* **Forecasting**)
moving average model
[90C26, 90C30]
(*see:* **Forecasting**)
moving coordinate system
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: differential equations**)
(MP) *see:* 1- —; minimum Partition Problem —; p- —
MPC
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
mPCC
[65K05, 90C26, 90C33, 90C34]
(*see:* **Adaptive convexification in semi-infinite optimization**)
MPEC
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
MPEC
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
MPEC Lagrangian
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
MPEC multipliers
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
MPI
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
MPI-based implementations
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
MPM
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
MPS
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
MS scheme
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
MSIM
[90B50]
(*see:* **Inventory management in supply chains**)
MSP
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
MSP *see:* STP- —
MST
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
MTT
[15A39, 90C05]
(*see:* **Motzkin transposition theorem**)
MTVSP
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
mu synthesis control
[93D09]
(*see:* **Robust control**)

multi-armed restless bandit problem
[90B36]
(*see:* **Stochastic scheduling**)
multi-attribute utility theory
[90C29, 91B06, 91B60]
(*see:* **Decision support systems with multiple criteria**; **Financial applications of multicriteria analysis**; **Preference disaggregation approach: basic features, examples from financial decision making**)
Multi-class data classification via mixed-integer optimization
Multi-commodity flows
(*see:* **Railroad crew scheduling**; **Railroad locomotive scheduling**)
multi-core
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)
multi-criteria problems
(*see:* **Planning in the process industry**)
Multi-depot vehicle scheduling problem
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
multi-depot vehicle scheduling problems
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
multi-echelon arborescence system
[90B50]
(*see:* **Inventory management in supply chains**)
multi-extremal global optimization
[90C25]
(*see:* **Concave programming**)
multi-extremal global optimization
[90C25]
(*see:* **Concave programming**)
multi-extremality
[90C05]
(*see:* **Continuous global optimization: applications**; **Global optimization in the analysis and management of environmental systems**)
multi-index assignment problems
[90C08, 90C11, 90C27, 90C35, 90C57, 90C59]
(*see:* **Multi-index transportation problems**; **Quadratic assignment problem**)
multi-index transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)
multi-index transportation problem *see:* axial —; integer —; planar —; symmetric —
Multi-index transportation problems
(90C35)
(*referred to in:* **Minimum concave transportation problems**; **Motzkin transposition theorem**; **Multidimensional assignment problem**; **Stochastic transportation and location problems**)
(*refers to:* **Generalized assignment problem**; **Stochastic transportation and location problems**)
multi-instrument financial equilibrium model *see:* multi-sector —
multi-knapsack problem
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
multi-objective
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
multi-objective CNSO
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
Multi-objective combinatorial optimization
(90C29, 90C27)
(*referred to in:* **Bi-objective assignment problem**; **Combinatorial matrix analysis**; **Combinatorial optimization algorithms in resource allocation problems**; **Combinatorial optimization games**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Evolutionary algorithms in combinatorial optimization**; **Financial applications of multicriteria analysis**; **Fractional combinatorial optimization**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**; **Replicator dynamics in combinatorial optimization**)
(*refers to:* **Bi-objective assignment problem**; **Combinatorial matrix analysis**; **Combinatorial optimization algorithms in resource allocation problems**; **Combinatorial optimization games**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Evolutionary algorithms in combinatorial optimization**; **Financial applications of multicriteria analysis**; **Fractional combinatorial optimization**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Neural networks for combinatorial optimization**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**; **Replicator dynamics in combinatorial optimization**)
multi-objective combinatorial optimization
[90C10, 90C35]
(*see:* **Bi-objective assignment problem**)

multi-objective convex optimization
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
multi-objective euclidean distance location *see:* Single facility location: —
multi-objective facility location
[90B85]
(*see:* **Single facility location: multi-objective rectilinear distance location**)
multi-objective fractional program
[90C32]
(*see:* **Fractional programming**)
multi-objective fractional programming
[90C32]
(*see:* **Fractional programming**)
Multi-objective fractional programming problems
(90C29)
Multi-objective integer linear programming
(90C29, 90C10)
(*referred to in:* **Bi-objective assignment problem**; **Branch and price: Integer programming with column generation**; **Broadcast scheduling problem**; **Decision support systems with multiple criteria**; **Decomposition techniques for MILP: lagrangian relaxation**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective mixed integer programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiparametric mixed integer linear programming**; **Multiple objective programming support**; **Outranking methods**; **Parametric mixed integer nonlinear optimization**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Bi-objective assignment problem**; **Branch and price: Integer programming with column generation**; **Decision support systems with multiple criteria**; **Decomposition techniques for MILP: lagrangian relaxation**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective mixed integer programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiparametric mixed integer linear programming**; **Multiple objective programming support**; **Outranking methods**; **Parametric mixed integer nonlinear optimization**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
multi-objective linear programming
[90C10, 90C26, 90C29, 91B28]
(*see:* **Multi-objective integer linear programming**; **Portfolio selection and multicriteria analysis**; **Vector optimization**)
multi-objective linear programming
[90C26, 91B28]
(*see:* **Portfolio selection and multicriteria analysis**)
multi-objective linear programming *see:* Fuzzy —
multi-objective linear programming with fuzzy coefficients
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
multi-objective linear programming under uncertainty
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
multi-objective mathematical programming
[91B06, 91B60]
(*see:* **Financial applications of multicriteria analysis**)
multi-objective mathematical programming
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
Multi-objective mixed integer programming
(90C29, 90C11)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multiparametric mixed integer linear**

programming; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)

multi-objective mixed integer programming
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)

multi-objective (multicriteria) mixed integer programming
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)

multi-objective optimization
[49M37, 65K05, 65K10, 90B50, 90B85, 90C11, 90C29, 90C30, 93A13]
(*see:* **MINLP: applications in the interaction of design and control**; **Multilevel methods for optimal design**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**; **Optimization and decision support systems**; **Selection of maximally informative genes**; **Single facility location: multi-objective rectilinear distance location**)

multi-objective optimization
[90B50, 90C11, 90C29, 90C90]
(*see:* **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**; **Optimization and decision support systems**)

multi-objective optimization *see:* disaggregation in —; Generalized concavity in —

Multi-objective optimization and decision support systems
(90B50, 90C29, 65K05, 90C05, 91B06)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)

Multi-objective optimization: interaction of design and control
(90C29, 90C11, 90C90)
(*referred to in:* **Bi-objective assignment problem**; **Control vector iteration CVI**; **Decision support systems with multiple criteria**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Optimal control of a flexible arm**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Bi-objective assignment problem**; **Control vector iteration CVI**; **Decision support systems with multiple criteria**; **Duality in optimal control with first order**

differential equations; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Optimal control of a flexible arm**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)

multi-objective optimization in the interaction of design and control
[90C11, 90C29, 90C90]
(*see:* **Multi-objective optimization: interaction of design and control**)

Multi-objective optimization; Interactive methods for preference value functions
(90C29)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)

Multi-objective optimization: lagrange duality
(90C29, 90C30)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Decomposition techniques for MILP: lagrangian relaxation**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Integer programming: lagrangian relaxation**; **Lagrange, Joseph-Louis**; **Lagrangian multipliers methods for convex programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Decomposition techniques for MILP: lagrangian relaxation**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Integer programming: lagrangian relaxation**; **Lagrange, Joseph-Louis**; **Lagrangian multipliers methods for convex programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)

Multi-objective optimization: pareto optimal solutions, properties
(90C29)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**;

Multi-objective optimization: interaction of design and control; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)

multi-objective programming
[90C29]
(*see:* **Preference disaggregation approach: basic features, examples from financial decision making**)

multi-objective programming
[90C10, 90C27, 90C29, 90C35]
(*see:* **Bi-objective assignment problem**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**)

multi-objective rectilinear distance location *see:* Single facility location: —

Multi-quadratic integer programming: models and applications
(65K05, 90C11, 90C20)

the multi-resource weighted assignment model
[90-00]
(*see:* **Generalized assignment problem**)

multi-resource weighted assignment model *see:* the —

Multi-scale global optimization using terrain/funneling methods
(65H20)

multi-sector multi-instrument financial equilibrium model
[91B50]
(*see:* **Financial equilibrium**)

multicenter *see:* minmax —

multiclass migration
[90C30]
(*see:* **Equilibrium networks**)

multiclass migration
[90C30]
(*see:* **Equilibrium networks**)

multiclass queueing networks
[90B36]
(*see:* **Stochastic scheduling**)

multicoloring
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)

multicommodity flow
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)

multicommodity flow
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)

multicommodity flow *see:* relaxed —

multicommodity flow problem *see:* node-path formulation of the —

Multicommodity flow problems
(90C35)
(*referred to in:* **Minimum cost flow problem**; **Nonconvex network flow problems**; **Nonoriented multicommodity flow problems**)
(*refers to:* **Auction algorithms**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Nonoriented multicommodity flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)

multicommodity flow problems *see:* large nonlinear —; nonlinear —; Nonoriented —

multicommodity model in OR
[90B80, 90B85]
(*see:* **Warehouse location problem**)

multicommodity network
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)

multicommodity network flow models *see:* undirected —

multicommodity network flow problem
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)

multicommodity network flows
[90C35]
(*see:* **Multicommodity flow problems**)

multicomputer *see:* coarse grained —

multiconstraint knapsack
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)

multiconstraint knapsack problem
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)

multicriteria analysis
[90C29, 91A99, 91B06, 91B60]
(*see:* **Financial applications of multicriteria analysis**; **Preference disaggregation**)

multicriteria analysis
[90C11, 90C29, 91A99, 91B06, 91B60]
(*see:* **Decision support systems with multiple criteria**; **Financial applications of multicriteria analysis**; **Multicriteria sorting methods**; **Multi-objective mixed integer programming**; **Preference disaggregation**;

Preference disaggregation approach: basic features, examples from financial decision making)
multicriteria analysis *see:* Financial applications of —; Portfolio selection and —
multicriteria decision aid
[90C29, 91B06, 91B60]
(*see:* **Decision support systems with multiple criteria**; **Financial applications of multicriteria analysis**; **Multicriteria sorting methods**; **Preference disaggregation approach: basic features, examples from financial decision making**)
multicriteria decision making
[90B80, 90B85]
(*see:* **Warehouse location problem**)
multicriteria decision making
[90C29, 90C70]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**; **Fuzzy multi-objective linear programming**)
multicriteria decision making problems: optimization techniques *see:* Estimating data for —
Multicriteria decision support methodologies for auditing decisions
(90C90, 90C11, 91B28)
multicriteria decision support system
[90C29]
(*see:* **Decision support systems with multiple criteria**)
multicriteria decision support system *see:* intelligent —
multicriteria decision support systems *see:* intelligent —
multicriteria DSS
[90C29]
(*see:* **Decision support systems with multiple criteria**)
multicriteria group decision support system
[90C29]
(*see:* **Decision support systems with multiple criteria**)
multicriteria group decision support system
[90C29]
(*see:* **Decision support systems with multiple criteria**)
Multicriteria methods for mergers and acquisitions
(91B28, 90C05, 90C90)
(multicriteria) mixed integer programming *see:* multi-objective —
multicriteria objective function
[90B80]
(*see:* **Facilities layout problems**)
multicriteria sorting method
[90C29]
(*see:* **Multicriteria sorting methods**)
Multicriteria sorting methods
(90C29)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
multicut methods
[90C06, 90C15]
(*see:* **Stabilization of cutting plane algorithms for stochastic linear programming problems**)
Multidimensional assignment problem
(90C10, 90C27)
(*refers to:* **Assignment and matching**; **Integer programming: branch and bound methods**; **Multi-index transportation problems**)
multidimensional assignment problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
multidimensional assignment problem *see:* Asymptotic properties of random —
multidimensional bisection
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
multidimensional bisection
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
multidimensional bracket
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
multidimensional knapsack
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
multidimensional knapsack problem
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
Multidimensional knapsack problems
(90C27, 90C10)
(*referred to in:* **Integer programming**; **Quadratic knapsack**)
(*refers to:* **Integer programming**; **Integer programming: branch and bound methods**; **Quadratic knapsack**)
multidimensional multiple-choice knapsack problem
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)

multidimensional scaling
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
multidimensional scaling problem
[65D18, 90B85, 90C26]
(*see:* **Global optimization in location problems**)
multidimensional transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)
multidimensional zero-one knapsack problem
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
multidisciplinary design
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
Multidisciplinary design optimization
(65F10, 65F50, 65H10, 65K10)
(*referred to in:* **Design optimization in computational fluid dynamics**; **Interval analysis: application to chemical engineering design problems**; **Multilevel methods for optimal design**; **Optimal design of composite structures**; **Optimal design in nonlinear optics**; **Structural optimization: history**)
(*refers to:* **Bilevel programming: applications in engineering**; **Design optimization in computational fluid dynamics**; **Genetic algorithms**; **Interval analysis: application to chemical engineering design problems**; **Multilevel methods for optimal design**; **Optimal design of composite structures**; **Optimal design in nonlinear optics**; **Structural optimization: history**)
multidisciplinary design optimization
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
multidisciplinary optimization
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
multifacilities location
[90B80]
(*see:* **Facilities layout problems**)
multifacility *see:* discrete single-commodity single-criterion uncapacitated static —
multifacility location
[05C05, 05C85, 68Q25, 90B80]
(*see:* **Bottleneck steiner tree problems**)
multifacility location-allocation
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
multifacility problem in OR
[90B80, 90B85]
(*see:* **Warehouse location problem**)
Multifacility and restricted location problems
(90B85)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Complexity classes in optimization**; **Complexity theory**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
multifacility Weber objective function
[90B85]
(*see:* **Multifacility and restricted location problems**)
multifacility Weber problem
[90B85]
(*see:* **Multifacility and restricted location problems**)
multifacility Weber–Rawls objective function
[90B85]
(*see:* **Multifacility and restricted location problems**)
multifrontal method
[65Fxx]
(*see:* **Least squares problems**)
multigraph
[05-XX]
(*see:* **Frequency assignment problem**)
multigroup hierarchical discrimination
[90C29]
(*see:* **Multicriteria sorting methods**)
multilayer *see:* k-restrictive —
multilayered dielectric structures *see:* Global optimization of planar —
multilevel algorithm
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
multilevel generalized assignment problem
[90-00]
(*see:* **Generalized assignment problem**)
multilevel methods
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: two-phase methods**)
Multilevel methods for optimal design
(49M37, 65K05, 65K10, 90C30, 93A13)
(*referred to in:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function**

approach; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Design optimization in computational fluid dynamics**; **Interval analysis: application to chemical engineering design problems**; **Multidisciplinary design optimization**; **Multilevel optimization in mechanics**; **Optimal design of composite structures**; **Optimal design in nonlinear optics**; **Stochastic bilevel programs**; **Structural optimization: history**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Design optimization in computational fluid dynamics**; **Interval analysis: application to chemical engineering design problems**; **Multidisciplinary design optimization**; **Multilevel optimization in mechanics**; **Optimal design of composite structures**; **Optimal design in nonlinear optics**; **Stochastic bilevel programs**; **Structural optimization: history**)
multilevel optimization
[49J35, 65K99, 74A55, 74M10, 74M15, 90C26]
(*see:* **Quasidifferentiable optimization: applications**)
multilevel optimization
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
Multilevel optimization in mechanics
(49Q10, 74K99, 74Pxx, 90C90, 91A65)
(*referred to in:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Quasivariational inequalities**; **Stochastic bilevel programs**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Stochastic bilevel programs**)
multilevel problem formulation
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
multilevel programming
[90C26, 90C30, 90C31]
(*see:* **Bilevel programming: introduction, history and overview**)
multilevel programming problem
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
multilevel single-linkage
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
multiload shape design
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
multiload truss design
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
multimodal functions
[90C30]
(*see:* **Global optimization based on statistical models**)
multimodal networks
[90C30]
(*see:* **Equilibrium networks**)
multimodal traffic network equilibrium
[90C30]
(*see:* **Equilibrium networks**)
multimodal traffic network equilibrium model
[90C30]
(*see:* **Equilibrium networks**)
Multiparametric linear programming
(90C31, 90C05)
(*referred to in:* **Bounds and solution vector estimates for parametric NLPS**; **Global optimization in multiplicative programming**; **Linear programming**; **Multiparametric mixed integer linear programming**; **Multiplicative programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**)
(*refers to:* **Bounds and solution vector estimates for parametric NLPS**; **Global optimization in multiplicative programming**; **Linear programming**; **Multiparametric mixed integer linear programming**; **Multiplicative programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**)
Multiparametric mixed integer linear programming
(90C31, 90C11)
(*referred to in:* **Bounds and solution vector estimates for parametric NLPS**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch**

and cut algorithms; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric linear programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Time-dependent traveling salesman problem**)
(*refers to:* **Bounds and solution vector estimates for parametric NLPS**; **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric linear programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
multiperiod assignment problem
[90C35]
(*see:* **Multi-index transportation problems**)
multiperiod MINLP MEN synthesis model
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**)
multiperiod model
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
Multiperiod Models *see:* single versus —
multiperiod optimization
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**; **Optimal planning of offshore oilfield infrastructure**)
multiperiod optimization modeling framework
[90C30, 90C35]
(*see:* **Optimization in water resources**)
multiperiod planning
[90B80, 90B85]
(*see:* **Warehouse location problem**)
multiperiod stochastic program
[90B05, 90B06]
(*see:* **Global supply chain models**)
multiperiod stochastic program
[90B05, 90B06]
(*see:* **Global supply chain models**)
multiphase chemical equilibrium
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
multiphase chemical equilibrium *see:* Optimality criteria for —
multiphase spanning network
[90C27]
(*see:* **Steiner tree problems**)
multiphase Steiner network
[90C27]
(*see:* **Steiner tree problems**)
multiphase Steiner problems
[90C27]
(*see:* **Steiner tree problems**)
multiple
(*see:* **Railroad locomotive scheduling**)
multiple branches for bounded integer variable
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
multiple choice knapsack
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
multiple-choice knapsack problem
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
multiple-choice knapsack problem *see:* linear —; multidimensional —
multiple-class software package
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
multiple criteria *see:* Decision support systems with —
multiple criteria decision making
[65K05, 90-XX, 90B50, 90C05, 90C26, 90C29, 91B06, 91B28, 91B60]
(*see:* **Financial applications of multicriteria analysis**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**)
multiple criteria decision making
[65K05, 90B50, 90C05, 90C29, 91B06]
(*see:* **Multi-objective optimization and decision support systems**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**)
multiple criteria design problem
[90C29]
(*see:* **Multiple objective programming support**)
multiple criteria evaluation
[90C29]
(*see:* **Multiple objective programming support**)
multiple criteria problem *see:* continuous —; discrete —

multiple depot
[90B06]
(*see:* **Vehicle routing**)
multiple depots *see:* single depot/ —
multiple dogleg path
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
multiple-facility location
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
multiple-hub heuristic
[90C35]
(*see:* **Multi-index transportation problems**)
multiple knapsack problem
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
multiple Kuhn–Tucker points
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
multiple locomotive type models
(*see:* **Railroad locomotive scheduling**)
multiple minima
[65K10, 92C40]
(*see:* **Multiple minima problem in protein folding: αBB global optimization approach**)
multiple minima
[65K10, 92C40]
(*see:* **Multiple minima problem in protein folding: αBB global optimization approach**)
Multiple minima problem in protein folding: αBB global optimization approach
(92C40, 65K10)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms**; **Global optimization in Lennard–Jones and morse clusters**; **Graph coloring**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Simulated annealing methods in protein folding**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms**; **Global optimization in Lennard–Jones and morse clusters**; Global optimization in protein folding; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Protein folding: generalized-ensemble algorithms**; **Simulated annealing**; **Simulated annealing methods in protein folding**)
Multiple objective dynamic programming
(90C39, 90C31)
(*referred to in:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Neuro-dynamic programming**)
(*refers to:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Neuro-dynamic programming**)
multiple objective linear programming
[65K05, 90B50, 90C05, 90C29, 91B06]
(*see:* **Multi-objective optimization and decision support systems**; **Multiple objective programming support**)
multiple objective programming
[90C29]
(*see:* **Multiple objective programming support**)
multiple objective programming
[90C29, 90C31, 90C39]
(*see:* **Multiple objective dynamic programming**; **Multiple objective programming support**)
Multiple objective programming support
(90C29)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference**

disaggregation approach: basic features, examples from financial decision making; **Preference modeling**)
multiple objective programming support
[90C29]
(*see:* **Multiple objective programming support**)
multiple objective programming support
[90C29]
(*see:* **Multiple objective programming support**)
multiple objectives
[49-01, 49K10, 49M37, 90-01, 90C05, 90C27, 91B52]
(*see:* **Bilevel linear programming**)
multiple QP Kuhn–Tucker points
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
multiple runs
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
multiple sequence alignment
[90C35]
(*see:* **Optimization in leveled graphs**)
multiple sequence alignment
[90C35]
(*see:* **Optimization in leveled graphs**)
multiple shooting
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
multiple types of vehicles *see:* Vehicle scheduling problems with —
multiple-valued logic *see:* evaluation in —
multiplexing *see:* wavelength-division —
multiplicative function *see:* constraint on a —; program of minimizing a convex —
multiplicative functions *see:* sum of convex —
multiplicative program *see:* convex —; linear —
Multiplicative programming
(90C26, 90C31)
(*referred to in:* **Global optimization in multiplicative programming**; **Linear programming**; **Multiparametric linear programming**; **Parametric linear programming: cost simplex algorithm**)
(*refers to:* **Complexity classes in optimization**; **Computational complexity theory**; **Concave programming**; **Global optimization in multiplicative programming**; **Linear programming**; **Multiparametric linear programming**; **Parametric linear programming: cost simplex algorithm**)
multiplicative programming
[65K05, 90C25, 90C26, 90C30]
(*see:* **Concave programming**; **Monotonic optimization**)
multiplicative programming *see:* Global optimization in —
multiplicity of a prime
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
multiplier *see:* Lagrange —
multiplier adjustment
[90C30, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**)
multiplier approach *see:* Everett generalized Lagrange —
multiplier associated with an arc
[90C35]
(*see:* **Generalized networks**)
multiplier-free reduced Hessian SQP
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
multiplier method
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
multiplier methods
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
multiplier rule *see:* global Lagrange —; Lagrange —
multiplier sets *see:* Lagrange —
multiplier vector *see:* Lagrange —
multipliers *see:* extended set of Lagrange —; Lagrange —; Lagrangian —; MPEC —; orthogonality conditions on —; Stochastic programming: nonanticipativity and lagrange —
multipliers methods for convex programming *see:* Lagrangian —
multipliers for nonanticipativity constraints *see:* Lagrange —
multipliers for phase constraints *see:* Lagrange —
multipoint approximation
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
multipopulation replicator models
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
Multiprocessor Scheduling Problem *see:* minimum —
multiproduct
[49L20]
(*see:* **Dynamic programming: inventory control**)
multiproduct
[90C26]
(*see:* **Global optimization in batch design under uncertainty**)
multiproduct batch plant
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
multiproduct plant
[90C26]
(*see:* **Global optimization in batch design under uncertainty**)
multipurpose
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
multipurpose
[90C26]
(*see:* **Global optimization in batch design under uncertainty**)
multipurpose plant
[90C26]
(*see:* **Global optimization in batch design under uncertainty**)
multipurpose storage entities
(*see:* **Planning in the process industry**)

multiratio programs
[90C32]
(*see:* **Fractional programming**)
multistage
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
multistage applications
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
multistage IM
[90B50]
(*see:* **Inventory management in supply chains**)
multistage inventory management
[90B50]
(*see:* **Inventory management in supply chains**)
multistage inventory management
[90B50]
(*see:* **Inventory management in supply chains**)
multistage inventory management models
[90B50]
(*see:* **Inventory management in supply chains**)
multistage linking constraints
[90C30, 90C35]
(*see:* **Optimization in water resources**)
multistage mean-variance optimization problems *see:* Decomposition algorithms for the solution of —
multistage optimization
[90C27]
(*see:* **Time-dependent traveling salesman problem**)
multistage problems
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
multistage stochastic program
[90C15]
(*see:* **Multistage stochastic programming: barycentric approximation**)
multistage stochastic programming
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
Multistage stochastic programming: barycentric approximation
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **L-shaped method for two-stage stochastic programs with recourse**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
multistart
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
multistart process
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
multitarget tracking
[90C35]
(*see:* **Multi-index transportation problems**)
multivalued maps *see:* Generalized monotone —
multivalued monotone laws and variational inequalities
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
multivalued nonmonotone laws and hemivariational inequalities
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)

multivalued variational inequalities *see:* Solution methods for —
multivariable stability margin
[93D09]
(*see:* **Robust control**)
multivariable stability margin K
[93D09]
(*see:* **Robust control**)
multivariate distribution functions *see:* gradient of —
multivariate distributions *see:* Stochastic linear programs with recourse and arbitrary —
multivariate gamma distribution
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
multivariate interval Newton method
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)
multivariate normal distribution
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
multivariate normal distribution
[90C15]
(*see:* **Probabilistic constrained problems: convexity theory**)
multivariate probability distribution function
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
multivariate probability integral
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
multivariate probability integrals *see:* Approximation of —; lower bounds for —; upper bounds for —
multiWeber problem
[90B85]
(*see:* **Multifacility and restricted location problems**)
multiWeber problem
[90B85]
(*see:* **Multifacility and restricted location problems**)
multiWeber–Rawls problem
[90B85]
(*see:* **Multifacility and restricted location problems**)
multy-stage stochastic programs
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
multy-stage stochastic programs
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
Murty least-index refinement
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
mutated sequence *see:* minimum weight common —
mutation
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90, 92B05]
(*see:* **Genetic algorithms**; **Traveling salesman problem**)
mutation
[92B05]
(*see:* **Genetic algorithms**)
mutual information
[62F10, 94A17]
(*see:* **Entropy optimization: parameter estimation**)
Mutzel branch and cut algorithm *see:* Jünger– —
MV-algebra
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
mVC
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
mVI problem
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
MVL connectives *see:* semantics of —
MWCP
[90C20]
(*see:* **Standard quadratic optimization problems: applications**)
MWFAS
[90C35]
(*see:* **Feedback set problems**)
MWFVS
[90C35]
(*see:* **Feedback set problems**)
mWW
[74A40, 90C26]
(*see:* **Shape selective zeolite separation and catalysis: optimization methods**)
myopic
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)

N

N-adic assignments problems
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
n-ary relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
n-ary relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
N-dimensional Brownian motion
[60G35, 65K05]
(*see:* **Differential equations and global optimization**)
n-dimensional vectors *see:* lexicographical ordering for —
n-fold integer programming
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
n-fold matrix
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)

n Hessian matrix
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
n noninteracting
[92-08, 92C05, 92C40]
(*see:* **Protein folding: generalized-ensemble algorithms**)
N-normal primal problem
[90C29, 90C30]
(*see:* **Multi-objective optimization: lagrange duality**)
N P *see:* N P=co —; P = —
N P=co *N P*
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
n-queens problem
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
n-valued PI-systems *see:* subfamilies of —
NAG library
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
NAG parallel library
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
naive auction algorithm
[90C30, 90C35]
(*see:* **Auction algorithms**)
naïve Bayes
(*see:* **Bayesian networks**)
narrowing operator *see:* constraint —
Nasa
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
Nasa program
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
Nasa program
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
Nash–Cournot equilibrium *see:* Stackelberg– —
Nash equilibrium
[46A22, 49J35, 49J40, 49Jxx, 54D05, 54H25, 55M20, 90C15, 91A05, 91Axx, 91B06, 91B60]
(*see:* **Infinite horizon control and dynamic games**; **Minimax theorems**; **Oligopolistic market equilibrium**; **Stochastic quasigradient methods in minimax problems**)
Nash equilibrium
[90C15, 91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**; **Stochastic quasigradient methods in minimax problems**)
Nash equilibrium *see:* feedback —; memory strategy —; open-loop —; spatial Cournot– —; Stackelberg– —
Nash oligopolistic equilibrium *see:* Cournot– —
Nash oligopolistic equilibrium model *see:* Cournot– —
native conformation
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
native conformations *see:* discarding far-from- —
natural
[65H20, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**)
natural domain
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
natural interval extension
[65G20, 65G30, 65G40, 65K05, 90C26, 90C30]
(*see:* **Bounding derivative ranges**; **Interval global optimization**)
natural level functions
[34-xx, 34Bxx, 34Lxx, 93E24]
(*see:* **Complexity and large-scale least squares problems**)
natural numbers
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
natural numbers *see:* finite —
natural residual
[90C30, 90C33]
(*see:* **Implicit lagrangian**)
natural selection *see:* fundamental theorem of —
natural stream arcs
[90C30, 90C35]
(*see:* **Optimization in water resources**)
natural vector *see:* support of a —
naturally flowing wells of type a
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
naturally flowing wells of type b
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
Naum Zuselevich *see:* Shor —
Navier–Stokes code *see:* Reynolds-averaged —
NC
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
NC algorithm
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
NC method
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
NC method
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
NC3
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
NDEXPTIME
[90C60]
(*see:* **Complexity classes in optimization**)
NDO
[46N10, 90-00, 90C47]
(*see:* **Nondifferentiable optimization**)

NDO *see:* convex —
nDOMB algorithm
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)
NDP
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
NDSPACE
[90C60]
(*see:* **Complexity classes in optimization**)
NDTIME
[90C60]
(*see:* **Complexity classes in optimization**)
near degeneracy
[90C60]
(*see:* **Complexity of degeneracy**)
near-integer-fix
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
near-minimizer
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
near-neighbor load balancing scheme
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
near rational numbers *see:* infinitely —
near-simpliciality
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
near-simplicity
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
nearest insertion optimal partitioning algorithm
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
nearest-neighbor
[65K05, 90-08, 90B06, 90B35, 90C05, 90C06, 90C10, 90C11, 90C20, 90C27, 90C30, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**; **Traveling salesman problem**)
nearest neighbor (NN)
[68Q25, 68R10, 68W40, 90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Domination analysis in combinatorial optimization**; **Traveling salesman problem**)
nearest neighbor (RNN) *see:* repeated —
nearest point mapping
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
nearest vertex insertion (NVI)
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
nearly degenerate BFS
[90C60]
(*see:* **Complexity of degeneracy**)
necessary
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
necessary condition *see:* first order —; second order —
necessary conditions
[03H10, 49J27, 90C31, 90C34]
(*see:* **Semi-infinite programming and control problems**; **Semi-infinite programming: second order optimality conditions**)
necessary conditions
[90C30]
(*see:* **Image space approach to optimization**)
necessary conditions *see:* first order —; second order —
necessary conditions for optimality *see:* high-order —
necessary conditions for optimality for abnormal points *see:* High-order —
necessary constraint
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
necessary constraint *see:* weakly —
necessary optimality condition
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
necessary optimality condition without using (sub)gradients parametric representations
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
necessary optimality conditions
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
necessary optimality conditions
[90C26, 90C31, 90C39, 91A65]
(*see:* **Bilevel programming: implicit function approach**; **Second order optimality conditions for nonlinear optimization**)
necessary optimality conditions *see:* Equality-constrained nonlinear programming: KKT —; first order —; fritz John —; generalized —; KKT —; Kuhn–Tucker —
necessary and sufficient conditions
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
necessary and sufficient conditions
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
necessary and sufficient optimality conditions
[49K05, 49K10, 49K15, 49K20, 65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Duality in optimal control with first order differential equations**)
necessary and sufficient optimality conditions *see:* second order —
needs *see:* static/dynamic service —
negamax algorithm *see:* incremental —
negans *see:* modus —
negated relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx,

91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
negation
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
negation transformation
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
negative
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
negative curvature
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
negative curvature
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
negative curvature *see:* direction of —
negative cycles
[90C35]
(*see:* **Minimum cost flow problem**)
negative fault
[90Cxx]
(*see:* **Discontinuous optimization**)
negative fitness *see:* genetic engineering via —
negative gradient *see:* projected —
negative main diagonal
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
negative marginal values
[90C60]
(*see:* **Complexity of degeneracy**)
negative marginal values
[90C60]
(*see:* **Complexity of degeneracy**)
negative real numbers *see:* infinitely small —
negative-zero pattern *see:* positive- —
negatively *see:* dropped —
neighbor *see:* k- —; legal —; nearest- —
neighbor joining
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
neighbor load balancing scheme *see:* near- —
neighbor (NN) *see:* nearest —
neighbor (RNN) *see:* repeated nearest —
neighbor in tabu search *see:* allowed —; prohibited —
neighborhood
[05C15, 05C17, 05C35, 05C69, 65K05, 68T20, 68T99, 90C22, 90C26, 90C27, 90C30, 90C35, 90C59]
(*see:* **Global optimization: filled function methods**; **Lovász number**; **Metaheuristics**)
neighborhood *see:* 2-opt —; discrete —; exchange —; k- —; k-exchange —; Lin–Kernighan —; pair-exchange —
neighborhood descent *see:* variable —
neighborhood edge elimination ordering *see:* cobipartite —
neighborhood graphs *see:* empty —
neighborhood of a permutation
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
neighborhood search methods *see:* Variable —
neighborhood of a solution
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
neighborhood structure
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
neighborhood structure for the QAP *see:* K-L type —
neighborhoods *see:* large-scale —
neighboring bases
[90C05, 90C31]
(*see:* **Parametric linear programming: cost simplex algorithm**)
neighboring critical regions
[90C05, 90C31]
(*see:* **Parametric linear programming: cost simplex algorithm**)
neighboring stations *see:* one-hop —
neighbors
[90C05, 90C31]
(*see:* **Multiparametric linear programming**)
neighbors *see:* two-hop —
neighbors of the origin
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
Nelder–Mead algorithm
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
nested
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
nested Benders decomposition
[90C15, 90C90]
(*see:* **Decomposition algorithms for the solution of multistage mean-variance optimization problems**)
nested constraints
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
nested dissection
[65Fxx]
(*see:* **Least squares problems**)
nested family *see:* finite —
nested loops
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
nested loops
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
nested partitions
[90C11, 90C59]
(*see:* **Nested partitions optimization**)
Nested partitions optimization
(90C59, 90C11)
nested STO problem
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)
net demand
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)
net present value *see:* maximize —

net supply
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)
network
[05C05, 05C40, 68R10, 90C35]
(*see:* **Generalized networks**; **Network design problems**)
network
[05C05, 05C40, 68R10, 68W10, 90B06, 90B10, 90B15, 90C05, 90C06, 90C30, 90C35]
(*see:* **Frank–Wolfe algorithm**; **Maximum flow problem**; **Minimum cost flow problem**; **Network design problems**; **Nonoriented multicommodity flow problems**; **Stochastic network problems: massively parallel solution**; **Vehicle routing**)
network *see:* 1-median problem in a —; augmented —; bipartite —; capacity of an arc in a —; communication —; congested —; cost of an arc in a —; covering problem on a —; deterministic neural —; directed —; directed arc in a —; directed arc in a directed —; directed capacitated —; endpoint of an arc in a directed —; evolutionary —; feed-forward neural —; generalized —; heat and mass exchange —; incidence in a —; intermediate scale —; large region —; local-area computer —; macro scale —; mass exchanger —; mass and heat exchanger —; micro scale —; multicommodity —; multiphase spanning —; multiphase Steiner —; node in a —; node in a directed —; p-center problem on a —; recurrent neural —; regional —; residual —; routing of traffic in transmission —; Space-time —; star —; state-task- —; stochastic neural —; strongly connected —; survivable —; system-optimized transportation —; time replicated —; training a —; transformed —; two-layer feed-forward —; user-optimized transportation —; weekly space-time —
network arc
[90C35]
(*see:* **Generalized networks**)
network assignment problem *see:* Communication —
network connectivity
[90C35]
(*see:* **Maximum flow problem**)
network constraints
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
network constraints *see:* optimization under —
network cost *see:* minimizing —
network design
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
network design
[90-XX, 90B10, 90C26, 90C30, 90C35, 90C90, 91A65, 91B99]
(*see:* **Bilevel programming: applications**; **Nonconvex network flow problems**; **Survivable networks**)
network design problem
[90-01, 90B30, 90B50, 90C15, 90C26, 90C33, 91B32, 91B52, 91B74]
(*see:* **Bilevel programming in management**; **Stochastic bilevel programs**)
network design problem *see:* survivable —
Network design problems
(05C05, 05C40, 68R10, 90C35)
(*referred to in:* **Auction algorithms**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
(*refers to:* **Auction algorithms**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
network design and schedule construction
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
network design and schedule construction
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
network equilibrium
[90B06, 90B20, 90C30, 91B50]
(*see:* **Equilibrium networks**; **Traffic network equilibrium**)
network equilibrium *see:* fixed demand traffic —; multimodal traffic —; symmetric —; traffic —
network equilibrium model *see:* migration —; multimodal traffic —
network equilibrium with travel disutility functions *see:* traffic —
network flow *see:* minimum cost —; value of a —
network flow model
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
network flow models *see:* undirected multicommodity —
network flow problem
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)
network flow problem *see:* fixed charge —; linear —; minimum cost —; multicommodity —; nonconvex —; nonlinear —; nonlinear dynamic —; nonlinear single commodity —; piecewise linear minimum cost —; uncapacitated —
network flow problems *see:* dynamic —; Nonconvex —; nonlinear —; Piecewise linear —
network flows
[01A99]
(*see:* **History of optimization**)
network flows
[05C05, 05C40, 68R10, 90C35]
(*see:* **Generalized networks**; **Network design problems**; **Railroad crew scheduling**; **Railroad locomotive scheduling**)
network flows *see:* multicommodity —
network localization problem *see:* Sensor —
network localization problem, SNLP *see:* Semidefinite programming and the sensor —

Network location: covering problems
(90C35, 90B10, 90B80)
(*referred to in:* **Auction algorithms**; **Combinatorial optimization algorithms in resource allocation problems**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Generalized networks**; **Global optimization in Weber's problem with attraction and repulsion**; **Maximum flow problem**; **Minimum cost flow problem**; **MINLP: application in facility location-allocation**; **Multicommodity flow problems**; **Multifacility and restricted location problems**; **Network design problems**; **Nonconvex network flow problems**; **Nonoriented multicommodity flow problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Stochastic transportation and location problems**; **Survivable networks**; **Traffic network equilibrium**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Auction algorithms**; **Combinatorial optimization algorithms in resource allocation problems**; **Communication network assignment problem**; **Competitive facility location**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Generalized networks**; **Global optimization in Weber's problem with attraction and repulsion**; **Maximum flow problem**; **Minimum cost flow problem**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network design problems**; **Nonconvex network flow problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Piecewise linear network flow problems**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Shortest path tree algorithms**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Stochastic transportation and location problems**; **Survivable networks**; **Traffic network equilibrium**; **Voronoi diagrams in facility location**; **Warehouse location problem**)

network model *see:* dynamic traffic —

network node
[90C35]
(*see:* **Generalized networks**)

network node *see:* deficit of a —; excess of a —

network optimization
[90C30, 90C35]
(*see:* **Auction algorithms**)

network optimization
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)

network optimization system *see:* generalized —

network problem *see:* generalized —; match- —; pure —; stochastic —

network problems *see:* fixed demand traffic —; quadratic generalized —

network problems: massively parallel solution *see:* Stochastic —

network problems with travel demand functions *see:* elastic demand traffic —

network programming
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)

network programming
[91B28]
(*see:* **Financial optimization**)

network simplex algorithm
[90C35]
(*see:* **Minimum cost flow problem**)

network simplex algorithm
[90C35]
(*see:* **Minimum cost flow problem**)

network structure of the spatial price equilibrium problem
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)

network superstructure *see:* heat exchanger —

network synthesis
[90C05]
(*see:* **Continuous global optimization: applications**)

network synthesis
[90C90]
(*see:* **MINLP: heat exchanger network synthesis**)

network synthesis *see:* heat exchanger —; MINLP: heat exchanger —; Mixed integer linear programming: heat exchanger —

network synthesis without decomposition *see:* heat exchanger —

network topology
[65K05, 65Y05]
(*see:* **Parallel computing: models**)

network training *see:* Unconstrained optimization in neural —

networks *see:* all-optical —; Bayesian —; chain —; chain rule for Bayesian —; Directed tree —; Dynamic traffic —; dynamical Bayesian —; Equilibrium —; Evacuation —; fixed charge —; Flexible mass exchange —; flows in —; generalized —; Global optimization of heat exchanger —; heat exchanger —; Integer linear programs for routing and protection problems in optical —; mesh —; MINLP: mass and heat exchanger —; Mixed integer linear programming: mass and heat exchanger —; multiclass queueing —; multimodal —; neural —; Optimization in ad hoc —; queuing —; regeneration —; ring —; Survivable —; topology of transportation —

networks for combinatorial optimization *see:* Neural —

networks under uncertainty *see:* Bilevel programming framework for enterprise-wide process —

Neumann algebra *see:* von —

Neumann architecture *see:* von —

Neumann, John *see:* Von —

neural network *see:* deterministic —; feed-forward —; recurrent —; stochastic —
neural network training *see:* Unconstrained optimization in —
neural networks
[65K05, 68T20, 68T99, 90-08, 90C05, 90C06, 90C09, 90C10, 90C11, 90C20, 90C27, 90C30, 90C59, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**; **Metaheuristics**; **Optimization in boolean classification problems**)
neural networks
[65K05, 68T05, 90C26, 90C27, 90C30, 90C39, 90C52, 90C53, 90C55]
(*see:* **Forecasting**; **Neural networks for combinatorial optimization**; **Neuro-dynamic programming**; **Unconstrained optimization in neural network training**)
Neural networks for combinatorial optimization
(90C27, 90C30)
(*referred to in:* **Bayesian networks**; **Combinatorial matrix analysis**; **Combinatorial optimization algorithms in resource allocation problems**; **Combinatorial optimization games**; **Evolutionary algorithms in combinatorial optimization**; **Fractional combinatorial optimization**; **Multi-objective combinatorial optimization**; **Neuro-dynamic programming**; **Replicator dynamics in combinatorial optimization**; **Set covering, packing and partitioning problems**; **Unconstrained optimization in neural network training**)
(*refers to:* **Neuro-dynamic programming**; **Replicator dynamics in combinatorial optimization**; **Unconstrained optimization in neural network training**)
Neuro-dynamic programming
(90C39)
(*referred to in:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neural networks for combinatorial optimization**; **Replicator dynamics in combinatorial optimization**; **Unconstrained optimization in neural network training**)
(*refers to:* **Dynamic programming: average cost per stage problems**; **Dynamic programming in clustering**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: inventory control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**; **Hamilton–Jacobi–Bellman equation**; **Multiple objective dynamic programming**; **Neural networks for combinatorial optimization**; **Replicator dynamics in combinatorial optimization**; **Unconstrained optimization in neural network training**)
neurons
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
neurons *see:* input —; output —
New hybrid conjugate gradient algorithms for unconstrained optimization
(49M07, 49M10, 90C06, 65K)
new paradigm *see:* Modeling languages in optimization: a —
new trial steplength *see:* compute a safeguarded —
the New York Times
[90C05]
(*see:* **Ellipsoid method**)
New York Times *see:* the —
Newsam–Ramsdell method
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**)
newsboy model
[90B50]
(*see:* **Inventory management in supply chains**)
newsboy problem
[90C06, 90C08, 90C15]
(*see:* **Simple recourse problem**; **Stochastic quasigradient methods in minimax problems**)
newsboy problem
[90B50, 90C06, 90C08, 90C15]
(*see:* **Inventory management in supply chains**; **Simple recourse problem**; **Stochastic quasigradient methods in minimax problems**)
Newton *see:* interval —; quasi- —; truncated —
Newton algorithm *see:* interval —
Newton–Cauchy framework
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
Newton–Cauchy framework
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
Newton–Cauchy framework *see:* Unconstrained nonlinear optimization: —
Newton iteration *see:* interval —
Newton's method
[49J52, 49M37, 65K05, 68Q25, 68R05, 90-08, 90C05, 90C20, 90C22, 90C25, 90C27, 90C30, 90C32, 90C51, 90Cxx]
(*see:* **Cost approximation algorithms**; **Fractional combinatorial optimization**; **Interior point methods for semidefinite programming**; **Nondifferentiable optimization: Newton method**; **Nonlinear least squares: Newton-type methods**; **Quadratic programming over an ellipsoid**; **Symmetric systems of linear equations**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
Newton method
[49J52, 49M29, 49M37, 65K10, 68Q25, 68R05, 90-08, 90C06, 90C20, 90C25, 90C27, 90C30, 90C32]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**; **Fractional combinatorial optimization**; **Nondifferentiable optimization: Newton**

method; **Nonlinear least squares: Newton-type methods**; **Quadratic programming over an ellipsoid**)
Newton's method *see:* approximate —; bundle- —; damped —; damped Gauss– —; discrete truncated —; full-step Gauss– —; Gauss– —; Gauss–Newton method: Least squares, relation to —; homotopy —; inexact —; interval —; Krawczyk variation of the interval —; modified —; multivariate interval —; Nondifferentiable optimization: —; nonsmooth —; partial-update —; partitioned quasi- —; quasi- —; separated —; smoothing —; splitting —; SR1 quasi- —; symmetric rank-one quasi- —; truncated —; univariate interval —
Newton method of Broyden class *see:* quasi- —
Newton method in deterministic global optimization *see:* LP strategy for interval- —
Newton method: Least squares, relation to Newton's method *see:* Gauss– —
Newton's method in unconstrained optimal control *see:* Dynamic programming and —
Newton methods *see:* existence-proving properties of interval —; factorized quasi- —; inexact —; interval —; quasi- —
Newton operator *see:* interval —; univariate interval —
Newton procedure
[49M20, 90-08, 90C25]
(*see:* **Nondifferentiable optimization: cutting plane methods**)
Newton–Raphson method
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
Newton relation *see:* quasi- —
Newton search engine
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms**)
Newton search engine *see:* quasi- —
Newton software package *see:* block truncated —
Newton step
[37A35, 65K05, 90C05, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**; **Potential reduction methods for linear programming**)
Newton step
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**)
Newton step case of the trust region problem
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
Newton steps *see:* Automatic differentiation: calculation of —
Newton test
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)
Newton-type method
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
Newton-type methods *see:* Nonlinear least squares: —
Newton update *see:* BFGS quasi- —; Broyden–Fletcher–Goldfarb–Shanno quasi- —; quasi- —
Newton updates *see:* quasi- —
Newton updating *see:* inverse quasi- —
Newtonian descent
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
Newtonian descent
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
Newtonian descent direction *see:* quasi- —
next shortest path procedure
[90C35]
(*see:* **Multicommodity flow problems**)
Nicholas Constantine *see:* Metropolis —
NIMBY syndrome
[90B80, 90B85]
(*see:* **Warehouse location problem**)
NLP
[65L99, 90C06, 90C10, 90C11, 90C30, 90C57, 90C90, 93-XX]
(*see:* **Modeling difficult optimization problems**; **Optimization strategies for dynamic systems**)
NLP *see:* Sensitivity and stability in —; solution-point bounds for —
NLP: approximation *see:* Sensitivity and stability in —
NLP based branch and bound *see:* IP/ —; QP/ —
NLP: continuity and differential stability *see:* Sensitivity and stability in —
NLP solvers *see:* bottlenecks in —
NLP subproblem
[49M20, 90C11, 90C30]
(*see:* **Generalized outer approximation**)
NLP techniques
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
NLPs *see:* Bounds and solution vector estimates for parametric —; Twice-differentiable —
NLS
[90C30]
(*see:* **Nonlinear least squares problems**)
NM
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
NM *see:* damped —; smoothing —
NN
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
(NN) *see:* nearest neighbor —
NNFP
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)
NNFP *see:* global minimum of an —
no capacity constraints *see:* single fixed cost with —
no free lunch
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
no pivoting required
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
Nobel Prize
[01A99]
(*see:* **Kantorovich, Leonid Vitalyevich**)
node *see:* arrival —; arrival-ground —; balanced —; border —; cardinality of a —; center —; deficit of a network —; demand —; departure —; disallowed —; established —;

excess of a network —; fathoming a —; feasible —; infeasible —; last —; network —; root —; sink —; source —; supply —; transshipment —; tree —
node-arc formulation
[90C35]
(*see:* **Multicommodity flow problems**)
node-arc formulation of the problem
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
node-arc incidence matrix
[90C35]
(*see:* **Generalized networks**)
node-arc incidence matrix
[90C30]
(*see:* **Simplicial decomposition**)
node construction procedure *see:* best —
node cover
[90C35]
(*see:* **Maximum flow problem**)
node covering problem
[90C20, 90C60]
(*see:* **Quadratic knapsack**)
node in a directed network
[90C35]
(*see:* **Maximum flow problem**)
node-disjoint path
[90-XX]
(*see:* **Survivable networks**)
node-disjoint path
[90-XX]
(*see:* **Survivable networks**)
node flow balance equations
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)
node legend
(*see:* **Railroad crew scheduling**)
node in a network
[90C35]
(*see:* **Minimum cost flow problem**)
node oriented branch and bound method
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
node oriented construction procedure
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
node-path formulation of the multicommodity flow problem
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
node potentials
[90C35]
(*see:* **Minimum cost flow problem**)
node reconstruction *see:* parent —
node routing
[90B06]
(*see:* **Vehicle routing**)
node routing
[90B06]
(*see:* **Vehicle routing**)
node tightening
(*see:* **Fractional zero-one programming**)
node of a truss
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
nodes *see:* on-the-river hydropower —; physical junction —; plant —; plant/model —; PSA with dummy —; retailer —; retailer/model —; return —; Steiner —
nodes set *see:* border —; tree —
nodes with water storage capacity
[90C30, 90C35]
(*see:* **Optimization in water resources**)
noises *see:* optimization with —
noising method
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
noisy-AND
(*see:* **Bayesian networks**)
noisy functional dependence
(*see:* **Bayesian networks**)
noisy-OR
(*see:* **Bayesian networks**)
nomography
[01A60, 03B30, 54C70, 68Q17]
(*see:* **Hilbert's thirteenth problem**)
non-anchor
(*see:* **Semidefinite programming and the sensor network localization problem, SNLP**)
non-anticipative
[90C15, 90C90]
(*see:* **Decomposition algorithms for the solution of multistage mean-variance optimization problems**)
non-crossing
(*see:* **Contact map overlap maximization problem, CMO**)
Non-Differentiable Functions and Applications *see:* minimization Methods for —
non-linear
(*see:* **Global optimization: functional forms**)
non-smooth optimization *see:* Derivative-free methods for —
non standard methods *see:* unbounded controls and —
nonaccepting computation of a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
nonadaptive method
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
nonanticipative principle
[90C30, 90C35]
(*see:* **Optimization in water resources**)
nonanticipative with respect to a filtration *see:* stochastic process —
nonanticipative water resources policies
[90C30, 90C35]
(*see:* **Optimization in water resources**)
nonanticipativity
[90C15, 91B28]
(*see:* **Financial optimization**; **Multistage stochastic programming: barycentric approximation**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic programming: nonanticipativity and lagrange multipliers**)

nonanticipativity
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
nonanticipativity constraints
[68W10, 90B15, 90C06, 90C15, 90C30, 90C35]
(*see:* **Optimization in water resources**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: parallel factorization of structured matrices**)
nonanticipativity constraints *see:* Lagrange multipliers for —
nonanticipativity and lagrange multipliers *see:* Stochastic programming: —
nonanticipativity water resources policies
[90C30, 90C35]
(*see:* **Optimization in water resources**)
nonarbitrage condition for LDSU
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
nonassociative groupoid
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
nonassociative products
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
nonassociative products
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
nonassociativity
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
nonbasic
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
nonbasic column
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
nonbasic component
[90C30]
(*see:* **Convex-simplex algorithm**)
nonbasic component
[90C30]
(*see:* **Convex-simplex algorithm**)
nonbasic matrix
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
nonbasic variable *see:* eligible —
nonbasic variables
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
nonbonded distance
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
nonbonded distance
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
noncentral component
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
noncommutative groupoid
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
noncompactness *see:* measure of —
noncompensatory argument
[90-XX]
(*see:* **Outranking methods**)
nonconvex
[35B40, 37C70, 49J24, 90C10, 90C11, 90C25, 90C27, 90C30, 90C33]
(*see:* **Continuous reformulations of discrete-continuous optimization problems**; **Successive quadratic programming: solution by active sets and interior point methods**; **Turnpike theory: stability of optimal trajectories**)
nonconvex
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
nonconvex dual problem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
nonconvex energy function
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
Nonconvex energy functions: hemivariational inequalities
(49J40, 70-XX, 80-XX, 49J52, 49Q10, 74K99, 74Pxx)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**;

Hemivariational inequalities: static problems; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

nonconvex feasibility analysis *see:* Shape reconstruction methods for —

nonconvex function
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)

nonconvex minimization
[90C26]
(*see:* **Convex envelopes in optimization problems**)

nonconvex minimization
[90C26, 90C31]
(*see:* **Global optimization in multiplicative programming**; **Multiplicative programming**)

nonconvex minimization problems *see:* decomposition algorithms for —

nonconvex MINLP
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)

nonconvex network flow problem
[90B10, 90C26, 90C30, 90C35]
(*see:* **Nonconvex network flow problems**)

Nonconvex network flow problems
(90C26, 90C30, 90C35, 90B10)
(*referred to in:* **Auction algorithms**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Generalized networks**; **Global supply chain models**; **Inventory management in supply chains**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonoriented multicommodity flow problems**; **Operations research models for supply chain management and design**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
(*refers to:* **Auction algorithms**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Generalized networks**; **Global supply chain models**; **Inventory management in supply chains**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonoriented multicommodity flow problems**; **Operations research models for supply chain management and design**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)

Nonconvex-nonsmooth calculus of variations
(49J40)
(*referred to in:* **Composite nonsmooth optimization**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Composite nonsmooth optimization**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic**

systems; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
nonconvex optimization
[93D09]
(*see:* **Robust control**)
nonconvex optimization
[26B25, 26E25, 49-XX, 49J40, 49J52, 49M37, 65K05, 65K99, 70-08, 90-XX, 90C25, 90C26, 90C30, 90C99, 91A10, 93-XX]
(*see:* **Bilevel programming**; **Convex envelopes in optimization problems**; **Duality theory: biduality in nonconvex optimization**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: codifferentiable functions**; **Smooth nonlinear nonconvex optimization**; **Solving hemivariational inequalities by nonsmooth optimization methods**)
nonconvex optimization *see:* Duality gaps in —; Duality theory: biduality in —; nonsmooth —; Smooth nonlinear —
nonconvex optimization problem
[90C26]
(*see:* **Global optimization in batch design under uncertainty**; **Smooth nonlinear nonconvex optimization**)
nonconvex primal problem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
nonconvex problem *see:* mixed integer —
nonconvex program *see:* nondifferentiable —
nonconvex programming
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
nonconvex programming
[90B06, 90B10, 90C26, 90C30, 90C35]
(*see:* **Minimum concave transportation problems**; **Nonconvex network flow problems**; **Reverse convex optimization**)
nonconvex programming problem
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
nonconvex programming problems *see:* convex and —
nonconvex programs
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
nonconvex programs
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
nonconvex quadratic programming
[90C60]
(*see:* **Complexity theory**; **Complexity theory: quadratic programming**)
nonconvex set
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
nonconvex SQP
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
nonconvex superpotential
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
nonconvexity
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)
nonconvexity
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
nonconvexity *see:* low-rank —
nonconvexity test
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
nonconvexity test *see:* monotonicity and —
noncooperative behavior
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
noncooperative behavior
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
noncooperative equilibrium
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
Noncooperative equilibrium
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
noncooperative game
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
noncooperative games
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
noncooperative solution
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
noncycling
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
nondecreasing function
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
nondecreasing monotone Boolean function
[90C09]
(*see:* **Inference of monotone boolean functions**)
nondegeneracy
[90C60]
(*see:* **Complexity of degeneracy**)
nondegeneracy
[90C22, 90C25, 90C31, 90C60]
(*see:* **Complexity of degeneracy**; **Semidefinite programming: optimality conditions and stability**)
nondegeneracy assumption
[90C33]
(*see:* **Lemke method**)

nondegeneracy assumption for algorithm analysis
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
nondegeneracy condition
[90C33]
(*see:* **Linear complementarity problem**)
nondegeneracy condition *see:* linear —; quadratic —
nondegenerate
[41A10, 46N10, 47N10, 49K27, 57R12, 65K05, 90C20, 90C31, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions; **High-order necessary conditions for optimality for abnormal points**; **Quadratic programming with bound constraints**; **Smoothing methods for semi-infinite optimization**)
nondegenerate
[90C05]
(*see:* **Linear programming**)
nondegenerate BFS
[90C60]
(*see:* **Complexity of degeneracy**)
nondegenerate critical point
[49J52, 49Q10, 58E05, 65K05, 65K10, 74G60, 74H99, 74K99, 74Pxx, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34, 90C90]
(*see:* **Parametric optimization: embeddings, path following and singularities**; **Quasidifferentiable optimization: stability of dynamic systems**; **Topology of global optimization**)
nondegenerate critical points
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
nondegenerate cycling
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
nondegenerate pivot operation
[90C35]
(*see:* **Minimum cost flow problem**)
nondegenerate point
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
nondegenerate problems
[05B35, 65K05, 90C05, 90C20, 90C30, 90C33]
(*see:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Rosen's method, global convergence, and Powell's conjecture**)
nondegenerate solution
[90C60]
(*see:* **Complexity of degeneracy**)
nondegenerate systems
[90C60]
(*see:* **Complexity of degeneracy**)
nondeterministic
[90C60]
(*see:* **Complexity classes in optimization**)
nondeterministic polynomial algorithm
[90C60]
(*see:* **Computational complexity theory**)
nondeterministic polynomial algorithm
[90C60]
(*see:* **Computational complexity theory**)
nondeterministic polynomial time algorithm
[90C60]
(*see:* **Computational complexity theory**)
nondeterministic Turing machine
[90C60]
(*see:* **Complexity theory**)
nondeterministic Turing machine *see:* space complexity of a —; time complexity of a —
nondifferentiability
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
nondifferentiability
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: nondifferentiable problems**)
nondifferentiable convex optimization
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
nondifferentiable convex program
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
nondifferentiable function
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
nondifferentiable nonconvex program
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
nondifferentiable objective functions
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
nondifferentiable objective functions
[90C30]
(*see:* **Sequential simplex method**)
Nondifferentiable optimization
(90-00, 90C47, 46N10)
(*referred to in:* **Dini and Hadamard derivatives in optimization**; **Discontinuous optimization**; **Global optimization: envelope representation**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**)
(*refers to:* **Dini and Hadamard derivatives in optimization**; **Discontinuous optimization**; **Generalized benders decomposition**; **Global optimization: envelope representation**; **Integer programming: lagrangian relaxation**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**; **Quasidifferentiable optimization: exact penalty methods**)
nondifferentiable optimization
[46N10, 49M20, 65K05, 90-00, 90-08, 90C25, 90C26, 90C30, 90C31, 90C47, 90Cxx]

(*see:* **Bilevel programming: introduction, history and overview**; **Cyclic coordinate method**; **Dini and Hadamard derivatives in optimization**; **Discontinuous optimization**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**)

Nondifferentiable optimization: cutting plane methods
(49M20, 90C25, 90-08)
(*referred to in:* **Dini and Hadamard derivatives in optimization**; **Global optimization: envelope representation**; **Nondifferentiable optimization**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**)
(*refers to:* **Dini and Hadamard derivatives in optimization**; **Global optimization: envelope representation**; **Linear programming: karmarkar projective algorithm**; **Nondifferentiable optimization**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**)

Nondifferentiable optimization: minimax problems
(90C30, 65K05)
(*referred to in:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Dini and Hadamard derivatives in optimization**; **Global optimization: envelope representation**; **Minimax: directional differentiability**; **Minimax theorems**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**; **Stochastic programming: minimax approach**; **Stochastic quasigradient methods in minimax problems**)
(*refers to:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Dini and Hadamard derivatives in optimization**; **Global optimization: envelope representation**; **Minimax: directional differentiability**; **Minimax theorems**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**; **Stochastic programming: minimax approach**; **Stochastic quasigradient methods in minimax problems**)

Nondifferentiable optimization: Newton method
(49J52, 90C30)
(*referred to in:* **Automatic differentiation: calculation of Newton steps**; **Dini and Hadamard derivatives in optimization**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Global optimization: envelope representation**; **Interval Newton methods**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**; **Nonlinear least squares: Newton-type methods**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
(*refers to:* **Automatic differentiation: calculation of Newton steps**; **Dini and Hadamard derivatives in optimization**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Global optimization: envelope representation**; **Interval Newton methods**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)

Nondifferentiable optimization: parametric programming
(90C05, 90C25, 90C29, 90C30, 90C31)
(*referred to in:* **Bilevel programming: optimality conditions and duality**; **Bounds and solution vector estimates for parametric NLPS**; **Dini and Hadamard derivatives in optimization**; **Global optimization: envelope representation**; **Model based control for drug delivery systems**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**)
(*refers to:* **Bilevel programming: introduction, history and overview**; **Bounds and solution vector estimates for parametric NLPS**; **Dini and Hadamard derivatives in optimization**; **First order constraint qualifications**; **Global optimization: envelope representation**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and**

singularities; **Second order constraint qualifications**; **Selfdual parametric method for linear programs**)

Nondifferentiable optimization: relaxation methods
(49J52, 90C30)
(*referred to in:* **Dini and Hadamard derivatives in optimization**; **Global optimization: envelope representation**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: subgradient optimization methods**)
(*refers to:* **Dini and Hadamard derivatives in optimization**; **Global optimization: envelope representation**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: subgradient optimization methods**)

Nondifferentiable optimization: subgradient optimization methods
(49J52, 90C30)
(*referred to in:* **Dini and Hadamard derivatives in optimization**; **Global optimization: envelope representation**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**; **Quadratic assignment problem**)
(*refers to:* **Dini and Hadamard derivatives in optimization**; **Global optimization: envelope representation**; **Integer programming: lagrangian relaxation**; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: Newton method**; **Nondifferentiable optimization: parametric programming**; **Nondifferentiable optimization: relaxation methods**)

nondifferentiable problems *see:* Interval analysis: —

nondifferential optimization
[01A99]
(*see:* **History of optimization**)

nondiscordance condition
[90-XX]
(*see:* **Outranking methods**)

nondominance
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)

nondominated
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**)

nondominated cuts
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)

nondominated path
[90C31, 90C39]
(*see:* **Multiple objective dynamic programming**)

nondominated solution
[90C29]
(*see:* **Multiple objective programming support**)

nondominated solution
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)

nondominated solution *see:* unsupported —; weakly —

nondominated solution set
[90C11, 90C29, 90C90]
(*see:* **Multi-objective optimization: interaction of design and control**)

nondominated valid inequality
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)

nondomination *see:* comparison of efficiency and —

noneconomic applications
[90C32]
(*see:* **Fractional programming**)

nonempty
[90C30]
(*see:* **Frank–Wolfe algorithm**)

nonempty interior
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)

nonexpansive
[65K10, 65M60]
(*see:* **Variational inequalities**)

nonexpansive operator
[65K10, 65M60]
(*see:* **Variational inequalities**)

nonexpansive operator *see:* firmly —

nonextreme efficient
[90B30, 90B50, 90C05, 91B82]
(*see:* **Data envelopment analysis**)

nonfeasible decomposition method
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)

nonfeasible gradient controller
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)

nonGaussian signal processing
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)

nonhomogeneous and nonisotropic body *see:* linear thermoelastic behavior of a generally —

nonideal part
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in distillation systems**)

nonideal phase equilibrium equations *see:* ideal and —

nonidentical machines
[68Q99]
(*see:* **Branch and price: Integer programming with column generation**)

nonincreasing monotone Boolean function
[90C09]
(*see:* **Inference of monotone boolean functions**)
noninferior
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
noninferior solution
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
noninferior solution
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
noninferior solution *see:* weakly —
noninferior solution set
[49M37, 90C11, 90C29, 90C90]
(*see:* **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**)
noninteracting *see:* n —
noninteractive method
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
noninteractive methods *see:* interactive versus —
noninterference constraints
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
noninterference constraints *see:* binary —
nonisotropic body *see:* linear thermoelastic behavior of a generally nonhomogeneous and —
nonlinear
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
nonlinear
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
nonlinear assignment problems
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
nonlinear bilevel programming: deterministic global optimization *see:* Mixed integer —
nonlinear blending
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
nonlinear boundary conditions *see:* elastostatics with —
nonlinear CG method
[90C30]
(*see:* **Conjugate-gradient methods**)
nonlinear CG method
[90C30]
(*see:* **Conjugate-gradient methods**)
nonlinear CG-related algorithms
[90C30]
(*see:* **Conjugate-gradient methods**)
nonlinear CG-related algorithms
[90C30]
(*see:* **Conjugate-gradient methods**)
nonlinear complementarity
[49M37, 90C26, 91A10]
(*see:* **Bilevel programming**)
nonlinear complementarity problem
[65F10, 65F50, 65H10, 65K10, 65M60, 90C30, 90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Globally convergent homotopy methods**; **Implicit lagrangian**; **Linear complementarity problem**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Topological methods in complementarity theory**; **Variational inequalities**)
nonlinear complementarity problem
[90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**)
nonlinear complementarity problem *see:* generalizations of the —; Generalized —; parametric —
nonlinear complementarity problem and fixed point problem *see:* Equivalence between —
nonlinear complementarity problems
[90C31, 90C33]
(*see:* **Sensitivity analysis of complementarity problems**)
nonlinear complementarity problems and variational inequalities *see:* Nonsmooth and smoothing methods for —
nonlinear complementary problem
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
nonlinear complementary problem
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
nonlinear constraint
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
nonlinear cut
[90C26]
(*see:* **Cutting plane methods for global optimization**)
nonlinear cut
[90C26]
(*see:* **Cutting plane methods for global optimization**)
nonlinear Dantzig–Wolfe decomposition
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
nonlinear decision models
[90C05]
(*see:* **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Global optimization in the analysis and management of environmental systems**)
nonlinear, decreasing
[90B15]
(*see:* **Evacuation networks**)
nonlinear diffusion equation
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
nonlinear discretized SIP problem
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)

nonlinear dynamic network flow problem
[90C30]
(*see:* **Simplicial decomposition**)
nonlinear dynamics *see:* Robust design of dynamic systems by constructive —
nonlinear equation *see:* one-dimensional —
nonlinear equations
[65G20, 65G30, 65G40]
(*see:* **Interval analysis: systems of nonlinear equations**)
nonlinear equations
[65F10, 65F50, 65G20, 65G30, 65G40, 65H10, 65J15, 65K10]
(*see:* **Contraction-mapping**; **Globally convergent homotopy methods**; **Interval analysis: systems of nonlinear equations**)
nonlinear equations *see:* Global optimization methods for systems of —; Interval analysis: systems of —; overdetermined system of —; systems of —; underdetermined system of —; well-determined system of —
nonlinear feasibility problem
[49M20, 90C11, 90C30]
(*see:* **Generalized outer approximation**)
nonlinear integer programming problem *see:* mixed —
nonlinear least squares
[49M37, 90C30]
(*see:* **Generalized total least squares**; **Nonlinear least squares: trust region methods**)
nonlinear least squares
[90C30]
(*see:* **Nonlinear least squares problems**)
nonlinear least squares *see:* generalized —
Nonlinear least squares: Newton-type methods
(49M37)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Conjugate-gradient methods**; **Contraction-mapping**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Global optimization methods for systems of nonlinear equations**; **Gröbner bases for polynomial equations**; **Interval analysis: systems of nonlinear equations**; **Large scale trust region problems**; **Least squares orthogonal polynomials**; **Least squares problems**; **Local attractors for gradient-related descent iterations**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**; **Optimization-based visualization**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Automatic differentiation: calculation of Newton steps**; **Conjugate-gradient methods**; **Contraction-mapping**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Global optimization methods for systems of nonlinear equations**; **Interval analysis: systems of nonlinear equations**; **Interval Newton methods**; **Large scale trust region problems**; **Least squares orthogonal polynomials**; **Least squares problems**; **Local attractors for gradient-related descent iterations**; **Nondifferentiable optimization: Newton method**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**; **Nonlinear systems of equations: application to the enclosure of all azeotropes**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
nonlinear least squares problem *see:* generalized —; unconstrained —
Nonlinear least squares problems
(90C30)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Least squares orthogonal polynomials**; **Least squares problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Least squares orthogonal polynomials**; **Least squares problems**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**)
Nonlinear least squares: trust region methods
(49M37)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Conjugate-gradient methods**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Least squares orthogonal polynomials**; **Least squares problems**; **Local attractors for gradient-related descent iterations**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Overdetermined systems of linear equations**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **ABS algorithms for optimization**; **Conjugate-gradient methods**; **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**; **Large scale trust region problems**; **Least squares orthogonal polynomials**; **Least squares problems**; **Local attractors for gradient-related descent iterations**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**)
nonlinear material laws *see:* discretized hemivariational inequalities for —
nonlinear mathematical programming problem
[49J20, 49J52]
(*see:* **Shape optimization**)
nonlinear mixed integer programming problem *see:* large scale —
nonlinear multicommodity flow problems
[90C30]
(*see:* **Simplicial decomposition**)
nonlinear multicommodity flow problems *see:* large —
nonlinear network flow problem
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
nonlinear network flow problems
[90C30]
(*see:* **Convex-simplex algorithm**)
nonlinear nonconvex optimization *see:* Smooth —

nonlinear optics
[34A55, 78A60, 90C30]
(*see:* **Optimal design in nonlinear optics**)
nonlinear optics
[34A55, 78A60, 90C30]
(*see:* **Optimal design in nonlinear optics**)
nonlinear optics *see:* Optimal design in —
nonlinear optimization
[90B50, 90C26]
(*see:* **Optimization and decision support systems**; **Smooth nonlinear nonconvex optimization**)
nonlinear optimization
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26, 91B28]
(*see:* **Financial optimization**; **Information-based complexity and information-based optimization**)
nonlinear optimization *see:* global —; Inequality-constrained —; mixed integer —; parametric —; Parametric mixed integer —; Second order optimality conditions for —; smooth —
nonlinear optimization: A disjunctive cutting plane approach *see:* Mixed-integer —
nonlinear optimization: Newton–Cauchy framework *see:* Unconstrained —
nonlinear optimization problem
[49K99, 65K05, 80A10, 90C26, 90C90]
(*see:* **Design optimization in computational fluid dynamics**; **Optimality criteria for multiphase chemical equilibrium**; **Smooth nonlinear nonconvex optimization**)
nonlinear optimization problem
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
nonlinear optimization problems
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
nonlinear order complementarity problem
[90C33]
(*see:* **Order complementarity**)
nonlinear parametric optimization
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
nonlinear potential
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
nonlinear problem *see:* fully —; geometrically —; physically —
nonlinear problems *see:* Simultaneous estimation and optimization of —
nonlinear program
[90C05, 90C20, 90C26, 90C30]
(*see:* **Global optimization in batch design under uncertainty**; **Redundancy in nonlinear programs**; **Simplicial decomposition**)
nonlinear program
[90C30]
(*see:* **Simplicial decomposition**)
nonlinear program *see:* loss of descent in a —; mixed integer —; relaxed —
nonlinear programming
[90C06, 90C10, 90C11, 90C22, 90C25, 90C30, 90C31, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**; **Semidefinite programming: optimality conditions and stability**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
nonlinear programming
[65K05, 65K10, 90B50, 90C06, 90C27, 90C30, 90C31, 90C34, 91B28]
(*see:* **Convex-simplex algorithm**; **Feasible sequential quadratic programming**; **Frank–Wolfe algorithm**; **Neural networks for combinatorial optimization**; **Operations research and financial markets**; **Optimization and decision support systems**; **Portfolio selection: markowitz mean-variance model**; **Rosen's method, global convergence, and Powell's conjecture**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: continuity and differential stability**; **Simplicial decomposition**)
nonlinear programming *see:* feasible direction method for —; mixed integer —; sensitivity in —
nonlinear programming algorithm *see:* descent in a —
nonlinear programming: KKT necessary optimality conditions *see:* Equality-constrained —
nonlinear programming problem
[49K20, 49M99, 90C26, 90C55, 90C90]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Design optimization in computational fluid dynamics**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
nonlinear programming problem *see:* equality-constrained —; mixed integer —
nonlinear programs *see:* Redundancy in —
nonlinear semi-infinite programs
[90C34]
(*see:* **Semi-infinite programming: approximation methods**)
nonlinear semi-infinite programs
[90C34]
(*see:* **Semi-infinite programming: approximation methods**)
nonlinear signal processing
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
nonlinear simplicial
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
nonlinear single commodity network flow problem
[90C30]
(*see:* **Simplicial decomposition**)
nonlinear SIP
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
nonlinear system of equations
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
nonlinear system of equations
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)

nonlinear systems of equations
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)
nonlinear systems of equations
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
nonlinear systems of equations *see:* error bound for approximate solutions of —; existence of solutions of —; rigorous bound for solutions of —; uniqueness of solutions of —
Nonlinear systems of equations: application to the enclosure of all azeotropes
(90C30)
(*referred to in:* **Contraction-mapping**; **Global optimization methods for systems of nonlinear equations**; **Gröbner bases for polynomial equations**; **Interval analysis: systems of nonlinear equations**; **Nonlinear least squares: Newton-type methods**)
(*refers to:* **Contraction-mapping**; **Global optimization methods for systems of nonlinear equations**; **Interval analysis: systems of nonlinear equations**; **Nonlinear least squares: Newton-type methods**)
nonlinearly constrained optimization
[90C25, 90C30]
(*see:* **Successive quadratic programming: full space methods**)
nonlocal sensitivity analysis
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
nonlocal sensitivity analysis
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
nonlocal sensitivity analysis *see:* automated Fortran program for —
Nonlocal sensitivity analysis with automatic differentiation
(34-XX, 49-XX, 65-XX, 68-XX, 90-XX)
(*referred to in:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Parametric global optimization: sensitivity**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**)
(*refers to:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: geometry of satellites and tracking stations**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**; **Automatic differentiation: root problem and branch problem**; **Parametric global optimization: sensitivity**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**)
nonminimum phase
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
nonmonotone Armijo-like criterion *see:* test —
nonmonotone laws and hemivariational inequalities *see:* multivalued —
nonmonotone line search
[90C06]
(*see:* **Large scale unconstrained optimization**)
nonnegative interpolatory operator
[90C34]
(*see:* **Semi-infinite programming: approximation methods**)
nonnegative lower bounds *see:* maximum flow problem with —
nonnegative matrix *see:* doubly —
nonoblivious local search
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
nonobtuse angle condition
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
Nonoriented multicommodity flow problems
(90B10, 90C05, 90C06, 90C35)
(*referred to in:* **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**)
nonparametric
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
nonparametric statistical method
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
nonparticipant
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
nonpreemptive
[90B36]
(*see:* **Stochastic scheduling**)
nonredundancy *see:* computational —
nonredundancy rate *see:* average —
nonredundant constraint
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)

nonregular operator
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
nonsaturating push
[90C35]
(*see:* **Maximum flow problem**)
nonseparable optimization problem
[93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**)
nonseparable problem
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
nonsingular
[90C30]
(*see:* **Convex-simplex algorithm**)
nonsingular local minimizer
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
nonsingular matrix
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
nonsingular matrix *see:* sign- —; strongly —
nonsmooth
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
nonsmooth analysis
[26B25, 26E25, 49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX, 90C15, 90C26, 90C99]
(*see:* **Global optimization: envelope representation**; **Nonconvex energy functions: hemivariational inequalities**; **Quasidifferentiable optimization**; **Stochastic quasigradient methods: applications**)
nonsmooth analysis
[26B25, 26E25, 46A20, 49J52, 52A01, 52A27, 65K05, 65K99, 90C30, 90C90, 90C99, 90Cxx]
(*see:* **Composite nonsmooth optimization**; **Dini and Hadamard derivatives in optimization**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
Nonsmooth analysis: Fréchet subdifferentials
(49K27, 90C48, 58C20, 58E30)
(*referred to in:* **Nonsmooth analysis: weak stationarity**)
(*refers to:* **Nonsmooth analysis: weak stationarity**)
nonsmooth analysis and optimization
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
Nonsmooth analysis: weak stationarity
(90C46, 90C48, 58C20, 58E30)
(*referred to in:* **Nonsmooth analysis: Fréchet subdifferentials**; **Smoothing methods for semi-infinite optimization**)
(*refers to:* **Nonsmooth analysis: Fréchet subdifferentials**)
nonsmooth calculus of variations *see:* Nonconvex- —
nonsmooth Dirichlet problem
[49J52]
(*see:* **Hemivariational inequalities: eigenvalue problems**)
nonsmooth eigenvalue problem
[49J52]
(*see:* **Hemivariational inequalities: eigenvalue problems**)
nonsmooth equations
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
nonsmooth function
[26B25, 26E25, 49J52, 90C99]
(*see:* **Quasidifferentiable optimization**)
nonsmooth functions
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
nonsmooth local approximations
[49J52, 49Q10, 74G60, 74H99, 74K99, 74Pxx, 90C90]
(*see:* **Quasidifferentiable optimization: stability of dynamic systems**)
nonsmooth mappings *see:* approximation of —; approximations of —
nonsmooth mechanics
[49J35, 49J40, 49J52, 49Q10, 49S05, 65K99, 70-XX, 74A55, 74G99, 74H99, 74K99, 74M10, 74M15, 74Pxx, 80-XX, 90C26, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**; **Nonconvex energy functions: hemivariational inequalities**; **Quasidifferentiable optimization: applications**)
nonsmooth mechanics
[49J40, 49J52, 49M05, 49Q10, 49S05, 70-08, 74G60, 74G99, 74H99, 74K99, 74Pxx, 90C33, 90C90]
(*see:* **Hemivariational inequalities: applications in mechanics**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**)
nonsmooth mechanics *see:* inequality or —
nonsmooth methods
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
nonsmooth modeling
[49J35, 65K99, 74A55, 74M10, 74M15, 90C26]
(*see:* **Quasidifferentiable optimization: applications**)
nonsmooth modeling
[49J35, 49J52, 65K99, 74A55, 74M10, 74M15, 90C26, 90C90]
(*see:* **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: calculus of quasidifferentials**)
nonsmooth Newton method
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
nonsmooth nonconvex optimization
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
nonsmooth optimization
[49J20, 49J52, 90C15, 90C26, 90C33]
(*see:* **Shape optimization**; **Stochastic bilevel programs**)
nonsmooth optimization
[26B25, 26E25, 46A20, 46N10, 49J35, 49J40, 49J52, 49M05, 49M20, 49S05, 52A01, 65G20, 65G30, 65G40, 65K05, 65K99, 70-08, 74A55, 74G99, 74H99, 74M10, 74M15, 74Pxx, 90-00, 90-08, 90C15, 90C25, 90C26, 90C30, 90C47, 90C90, 90C99]
(*see:* **Composite nonsmooth optimization**; **Direct global optimization algorithm**; **Farkas lemma: generalizations**;

Interval global optimization; **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: variational formulations**; **Random search methods**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **SSC minimization algorithms for nonsmooth and stochastic optimization**)
nonsmooth optimization *see:* Composite —
Nonsmooth optimization approach to clustering
(90C26, 90C56, 90C90)
nonsmooth optimization methods *see:* Solving hemivariational inequalities by —
nonsmooth optimization problems
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
nonsmooth optimization problems
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
nonsmooth reformulation *see:* smoothing- —
Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities
(90C33, 90C30)
(*referred to in:* **Composite nonsmooth optimization**; **Nonconvex-nonsmooth calculus of variations**; **Smoothing methods for semi-infinite optimization**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational principles**)
(*refers to:* **Composite nonsmooth optimization**; **Nonconvex-nonsmooth calculus of variations**; **Solving hemivariational inequalities by nonsmooth optimization methods**)
nonsmooth SSC-SABB algorithm
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms for nonsmooth and stochastic optimization**)
nonsmooth and stochastic optimization *see:* SSC minimization algorithms for —
nonsmooth superpotential
[49J52, 49Q10, 74G60, 74H99, 74K99, 74Pxx, 90C90]
(*see:* **Quasidifferentiable optimization: stability of dynamic systems**)
nonstandard analysis
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
nonstandard analysis
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
nonstandard framework
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
nonstochastic uncertainty
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
nonstoichiometric form of KT conditions
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
nonsupported efficient solution
[90C10, 90C29]
(*see:* **Multi-objective integer linear programming**)
nonsupported efficient solutions
[90C10, 90C35]
(*see:* **Bi-objective assignment problem**)
nontriviality condition
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
nonuniform
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)
nonunit capacity *see:* problem with —
nonunit weight CMST
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
nonzero pattern *see:* zero- —
nonzero residual problem
[90C30]
(*see:* **Nonlinear least squares problems**)
nonzero-sum infinite horizon game
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
norm *see:* A-weighted Euclidean —; approximation in the uniform —; L_1- —; normalized —; t- —; weighted maximum —; weighter sup —
norm contraction *see:* weighter sup- —
norm controllability *see:* minimum —
norm-dependent property
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
norm solution *see:* minimum —
normal
[65K05, 90C26, 90C30, 90C31]
(*see:* **Minimax: directional differentiability**; **Robust global optimization**)
normal compactness *see:* partial sequential —; sequential —
normal cone
[05A, 15A, 49J40, 49J52, 49Q10, 51M, 52A, 52B, 52C, 62H, 65K05, 65K10, 65M60, 68Q, 68R, 68U, 68W, 70-XX, 74K99, 74Pxx, 80-XX, 90B, 90C, 90C30, 90C31, 90C33, 90Cxx]
(*see:* **Convex discrete optimization**; **Nonconvex energy functions: hemivariational inequalities**; **Quasidifferentiable optimization: exact penalty methods**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities: geometric interpretation, existence and uniqueness**)
normal cone *see:* fréchet —; limiting —
normal distribution *see:* law of —; multivariate —
normal equation
[65Fxx, 90C30]

(*see:* **Generalized total least squares**; **Least squares problems**)
normal equations
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
normal extremal
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
normal form
[12D10, 12Y05, 13P10]
(*see:* **Gröbner bases for polynomial equations**)
normal form
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
normal form *see:* Boolean formula in conjunctive —; canonical —; complete many-valued logic —; conjunctive —; disjunctive —; game in —; many-valued —; PI- —
normal form of a polynomial
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
normal forms *see:* minimization of Pinkava —; PI-algebras and 2-valued —
normal forms of Pi-algebras *see:* functionally complete —
normal hull
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
normal hull *see:* reverse —
normal map
[90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**)
normal primal problem *see:* J- —; N- —
normal set *see:* reverse —
normalization
[62H30, 68T10, 90C05]
(*see:* **Linear programming models for classification**)
normalization of measures
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
normalization property
[90C33]
(*see:* **Topological methods in complementarity theory**)
normalized norm
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)
normalized stress
[65K05, 90C27, 90C30, 90C57, 91C15]
(*see:* **Optimization-based visualization**)
normalized structure factors
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)
normalized structure factors
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)
normalized volume
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
normative perspective
[90C29]
(*see:* **Preference modeling**)
normed linear spaces *see:* Best approximation in ordered —
norms *see:* t- —
normwise relative condition number
[65Fxx]
(*see:* **Least squares problems**)
North–West corner rule
[90C35]
(*see:* **Multi-index transportation problems**)
not dominated
[90C27, 90C29]
(*see:* **Multi-objective combinatorial optimization**)
notation *see:* Landau —; relational matrix —
notation for constraints
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
notation for objective functions
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
notation for relational operations *see:* matrix —
novel decomposition-based clustering approach: global optimum search with enhanced positioning *see:* Gene clustering: A —
novo protein design using flexible templates *see:* De —
novo protein designUsing rigid templates *see:* De —
NP
[90C60]
(*see:* **Complexity classes in optimization**)
nP-complete
[49M37, 68Q25, 90C11, 90C60]
(*see:* **Complexity theory**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**)
NP-complete
[90C60]
(*see:* **Complexity of degeneracy**; **Complexity theory**; **Complexity theory: quadratic programming**)
NP-complete completeness *see:* strong —
nP-complete problem
[68Q25, 90C60]
(*see:* **Complexity theory**; **Computational complexity theory**; **NP-complete problems and proof methodology**)
NP-complete problem
[68Q25, 90C60]
(*see:* **Complexity of degeneracy**; **Computational complexity theory**; **NP-complete problems and proof methodology**)
NP-complete problems and proof methodology
(90C60, 68Q25)
(*referred to in:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Integer programming:**

cutting plane algorithms; **Maximum cut problem, MAX-CUT**)
(*refers to:* **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Mixed integer nonlinear programming**; **Parallel computing: complexity classes**)
NP-complete reductions *see:* ordinary —
NP-completeness
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
NP-completeness *see:* ordinary —; strong —
NP-completeness proofs
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
NP-hard
[49M37, 68T99, 9008, 90C05, 90C06, 90C08, 90C10, 90C11, 90C26, 90C27, 90C59, 90C60, 93D09]
(*see:* **Capacitated minimum spanning trees**; **Complexity theory**; **Integer programming: branch and bound methods**; **Mixed integer nonlinear programming**; **Price of robustness for linear optimization problems**; **Robust control**; **Variable neighborhood search methods**)
NP-hard
[90B80, 90B85, 90C60]
(*see:* **Complexity theory**; **Complexity theory: quadratic programming**; **Facilities layout problems**; **Multifacility and restricted location problems**)
NP-hard *see:* strongly —
nP-hard problem
[68Q25, 90C60]
(*see:* **Computational complexity theory**; **NP-complete problems and proof methodology**)
NP-hard problem
[68Q25, 90C60]
(*see:* **Computational complexity theory**; **NP-complete problems and proof methodology**)
NP method *see:* pure —
NP methods *see:* hybrid —; knowledge-based —
NPC
[90C60]
(*see:* **Computational complexity theory**)
NPH
[90C60]
(*see:* **Computational complexity theory**)
NRTL equation
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
nSD
[65K10, 90C06, 90C25, 90C33, 90C35, 90C51]
(*see:* **Generalizations of interior point methods for the linear complementarity problem**; **Simplicial decomposition algorithms**)
NSM
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
ν-approximate gradient
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
nulispace *see:* numerical —
null space
[49-XX, 90-XX, 90C20, 90C30, 90Cxx, 93-XX]
(*see:* **Discontinuous optimization**; **Duality theory: biduality in nonconvex optimization**; **Successive quadratic programming: decomposition methods**)
null space
[90C20, 90C30]
(*see:* **Successive quadratic programming: decomposition methods**)
null space decomposition *see:* range and —
null step
[47J20, 49J40, 49J52, 65K05, 65K10, 90C30, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**; **Solving hemivariational inequalities by nonsmooth optimization methods**)
number *see:* blackball —; chromatic —; clique —; clique partition —; condition —; conspiracy —; crossing —; Dedekind —; domination —; fuzzy —; independence —; L-R flat fuzzy —; L-R fuzzy —; Lovász —; normwise relative condition —; separation —; stability —; tangle —; weighted clique —; weighted stability —
number of clauses *see:* minimum —
number of clusters *see:* Determining the optimal —
number of DNF clauses *see:* minimal —
number of a matrix *see:* condition —
number model *see:* real —
number of operations
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
number of pivot steps *see:* average —; expected —
number of shadow-vertices *see:* expected —; variance of the —
number of Steiner points *see:* Steiner tree problem with minimum —
number of vehicles
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
number of vehicles *see:* Vehicle scheduling problems with a fixed —
number of well switches *see:* maximum —
numbers *see:* arithmetic operations on fuzzy —; common random —; finite natural —; finite rational —; fuzzy —; infinitely near rational —; infinitely small negative real —; infinitely small positive real —; infinitely small real —; magic —; natural —; rational —; real —
Numerica
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
numerical algorithms
[49J40, 49Q10, 70-08, 74K99, 74Pxx]
(*see:* **Quasivariational inequalities**)
numerical algorithms
[90C25, 90C26, 90C34]
(*see:* **Semi-infinite programming: numerical methods**)

numerical analysis
[01A99, 90C99]
(*see:* **Von Neumann, John**)
numerical constraint satisfaction problem
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
numerical differentiation
[26A24, 65D25, 68W30]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**; **Complexity of gradients, Jacobians, and Hessians**)
numerical differentiation *see:* internal —
numerical example of a trim-loss problem
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
numerical methods
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
numerical methods
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
numerical methods *see:* Semi-infinite programming: —; Stochastic optimal stopping: —
Numerical methods for unary optimization
(90C30)
(*referred to in:* **Broyden family of methods and the BFGS update**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**; **Unconstrained optimization in neural network training**)
(*refers to:* **Broyden family of methods and the BFGS update**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**; **Unconstrained optimization in neural network training**)
numerical nulispace
[65Fxx]
(*see:* **Least squares problems**)
numerical rank
[15A23, 65F05, 65F20, 65F22, 65F25, 65Fxx]
(*see:* **Least squares problems**; **Orthogonal triangularization**)
numerical results
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
numerical simulation
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
(NVI) *see:* nearest vertex insertion —
Nyström interpolation
[65H10, 65J15]
(*see:* **Contraction-mapping**)

O

$\mathcal{O}(n^c)$ *see:* algorithm of complexity —
$\mathcal{O}(n^c)$ time *see:* algorithm running in —
OA master problem *see:* disjunctive —
OA method
[90C26]
(*see:* **Cutting plane methods for global optimization**)
object
(*see:* **State of the art in modeling agricultural systems**)
objective
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
objective *see:* maxmin —; minimax —; multi- —; Stochastic programming models: random —
objective assignment problem *see:* Bi- —
objective CNSO *see:* multi- —
objective combinatorial optimization *see:* multi- —
objective convex optimization *see:* multi- —
objective criterion
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
objective dynamic programming *see:* Multiple —
objective euclidean distance location *see:* Single facility location: multi- —
objective facility location *see:* multi- —
objective fractional program *see:* multi- —
objective fractional programming *see:* multi- —
objective fractional programming problems *see:* Multi- —
objective function
[65G30, 65G40, 65K05, 68Q25, 9008, 90B10, 90B80, 90B85, 90C05, 90C20, 90C26, 90C27, 90C30, 90C35, 90C57, 90C59, 90C90, 91B28]
(*see:* **Competitive ratio for portfolio management**; **Global optimization: interval analysis and balanced interval arithmetic**; **Global optimization using space filling**; **MINLP: heat exchanger network synthesis**; **Nonconvex network flow problems**; **Redundancy in nonlinear programs**; **Rosen's method, global convergence, and Powell's conjecture**; **Variable neighborhood search methods**; **Warehouse location problem**)
objective function
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
objective function *see:* fair —; lexicographically minimax —; maximin —; minimax —; multicriteria —; multifacility Weber —; multifacility Weber–Rawls —; random —; separable —; separable convex —; set-valued —
objective function parametrization
[90C05, 90C31]
(*see:* **Parametric linear programming: cost simplex algorithm**)
objective function value *see:* continuity property of the —; convexity property of the —
objective functions
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**)
objective functions *see:* nondifferentiable —; notation for —
objective functions and/or derivatives *see:* evaluation of —
objective integer linear programming *see:* Multi- —

objective interpretation
[94A17]
(*see:* **Jaynes' maximum entropy principle**)
objective linear programming *see:* Fuzzy multi- —; multi- —; multiple —
objective linear programming with fuzzy coefficients *see:* multi- —
objective linear programming under uncertainty *see:* multi- —
objective for a location problem
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
objective mathematical programming *see:* multi- —
objective mixed integer programming *see:* Multi- —
objective (multicriteria) mixed integer programming *see:* multi- —
objective optimization *see:* disaggregation in multi- —; Generalized concavity in multi- —; multi- —
objective optimization and decision support systems *see:* Multi- —
objective optimization: interaction of design and control *see:* Multi- —
objective optimization; Interactive methods for preference value functions *see:* Multi- —
objective optimization: lagrange duality *see:* Multi- —
objective optimization: pareto optimal solutions, properties *see:* Multi- —
objective programming *see:* multi- —; multiple —
objective programming support *see:* Multiple —
objective rectilinear distance location *see:* Single facility location: multi- —
objective simplex algorithm *see:* parametric —
objective value *see:* incumbent —
objectives *see:* balancing —; multiple —; pull —; push —
objects
[65K05, 90C27, 90C30, 90C57, 91C15]
(*see:* **Optimization-based visualization**)
oblique projection matrix
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
observation problem *see:* minimax —
observation problem under uncertainty with perturbations *see:* minimax —
observational quantifiers
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
observations *see:* inaccuracy in —
obstacle
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
obstacle-free shape design *see:* robust —
obstacle-free truss design *see:* robust —
obstruction set
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)
OCAT
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
Occam razor
[90C60]
(*see:* **Kolmogorov complexity**)
odd-hole-cut
(*see:* **Contact map overlap maximization problem, CMO**)
odd sequence
[05C85]
(*see:* **Directed tree networks**)
odd-set constraints
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
ODE two-point boundary value problem
[34A55, 78A60, 90C30]
(*see:* **Optimal design in nonlinear optics**)
Odyssée
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
off *see:* back- —; tailing- —; trade- —
off-0-diagonal operator
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
off cutting plane *see:* trade- —
off error *see:* round- —
off-line feedback
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
off-line learning
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
off-line process optimization
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
off question *see:* trade- —
offs *see:* trade- —
offshore oil fields
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
offshore oilfield infrastructure *see:* Optimal planning of —
offspring *see:* perfect —
oil fields *see:* offshore —
oil flowrate *see:* well —
oil, gas and water capacity constraints *see:* maximum —
oil model *see:* black —
oil rate constraints *see:* upper and lower well —
oilfield infrastructure *see:* Optimal planning of offshore —
oligopolistic equilibrium *see:* Cournot–Nash —
oligopolistic equilibrium model *see:* Cournot–Nash —
Oligopolistic market equilibrium
(91B06, 91B60)
(*referred to in:* **Equilibrium networks**; **Financial equilibrium**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Spatial price equilibrium**; **Traffic network equilibrium**; **Walrasian price equilibrium**)
(*refers to:* **Equilibrium networks**; **Financial equilibrium**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Spatial price**

equilibrium; **Traffic network equilibrium**; **Walrasian price equilibrium**)
oligopoly
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
oligopoly model
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)
oligopoly model *see:* spatial —
oligopoly problem *see:* aspatial —; classical —
OLSO
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
OME
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
Omega *see:* Chaitin in —
Ω-based yield
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
on average
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
on-duty time
(*see:* **Railroad crew scheduling**)
on-line algorithm
[05C85]
(*see:* **Directed tree networks**)
on-line feedback
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
on-line learning
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
on-line method
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
on-line process optimization
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
on-the-river hydropower nodes
[90C30, 90C35]
(*see:* **Optimization in water resources**)
one *see:* one against —
one against all
(*see:* **Mathematical programming for data mining**)
one against one
(*see:* **Mathematical programming for data mining**)
one algorithm *see:* Smith–Walford- —
one approach *see:* limited-memory symmetric rank- —
one-at-a-time coefficient generation
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
one clause at a time
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
one clause at a time algorithm
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
one clause at a time approach
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
one constraint *see:* consecutive —
one-dimensional marginal probability distribution function
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
one-dimensional nonlinear equation
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
one, find all *see:* find —
one-for-one ordering policy
[90B50]
(*see:* **Inventory management in supply chains**)
one formula *see:* selfdual rank —
one globally convergent homotopies *see:* probability- —
one homotopy *see:* probability- —
one homotopy algorithm *see:* globally convergent probability- —
one-hop neighboring stations
(*see:* **Broadcast scheduling problem**)
one integer feasibility problem *see:* zero- —
one integer problem *see:* linear zero- —
one integer program *see:* zero- —
one integer programming *see:* zero- —
one knapsack problem *see:* multidimensional zero- —; zero- —
one matrix *see:* rank- —
one optimization *see:* zero- —
one ordering policy *see:* one-for- —
one-parametric finite optimization problem
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
one-parametric semi-infinite optimization
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
one problem *see:* quadratic zero- —
one programming *see:* Fractional zero- —; pure zero- —
one programming problem *see:* zero- —
one quasi-Newton method *see:* symmetric rank- —
one-reducible graph *see:* Smith–Walford —
one-sided differential
[26B25, 26E25, 49J52, 90C99]
(*see:* **Quasidifferentiable optimization**)
one-to-all instances
[05C85]
(*see:* **Directed tree networks**)
one-tree
[90C35]
(*see:* **Generalized networks**)
one update *see:* symmetric rank- —

one update formula *see:* Sherman-Morrison rank- —
one-way analysis of variance
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
onto relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
open communication
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
open form approach
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
open list
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
open-loop control
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
open-loop Nash equilibrium
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
open shop problem
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
operability
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)
operability analysis of flowsheets
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
operating cash flow *see:* maximize —
operation *see:* averaging —; contraction —; degenerate pivot —; floating point —; freight —; head of —; interval arithmetic —; nondegenerate pivot —; partially asynchronous —; pivot —; tail of —; totally asynchronous —
operation of electric and energy power systems *see:* Optimization in —
operation planning
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
operation planning
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
operational decisions in a supply chain
[90-02]
(*see:* **Operations research models for supply chain management and design**)
Operational Research *see:* european Journal of —
operational restrictions
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
operational status of the wells
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
operational supply chain management
[90-02]
(*see:* **Operations research models for supply chain management and design**)
operations *see:* expansion —; irregular —; matrix notation for relational —; number of —; process —; reduction —; reflection —
operations on fuzzy numbers *see:* arithmetic —
operations problem *see:* irregular —
operations in a program *see:* basic —
operations on relations *see:* binary —; unary —
Operations research
(90C27)
(*referred to in:* **History of optimization**)
(*refers to:* **History of optimization**)
operations research
[90C31]
(*see:* **Sensitivity and stability in NLP**)
operations research
[90C27]
(*see:* **Operations research**)
operations research *see:* GRASP in —
Operations research and financial markets
(90C27)
Operations research models for supply chain management and design
(90-02)
(*referred to in:* **Global supply chain models**; **Inventory management in supply chains**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**)
(*refers to:* **Global supply chain models**; **Inventory management in supply chains**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**)
operator *see:* antitone —; best approximation —; block-0-diagonal —; closed selfadjoint —; co-coercive —; compact —; complementary —; completely continuous —; condensing —; constraint narrowing —; continuous selection —; contractive —; eor —; firmly nonexpansive —; fuzzy set-inclusion —; generalized monotone —; geometrical —; hemicontinuous —; heterotonic —; implication —; interpolatory —; interval —; interval Newton —; interval Taylor —; involutory —; isotone —; lipschitzian selection —; monotone —; nonexpansive —; nonnegative interpolatory —; nonregular —; off-0-diagonal —; optimal Lipschitzian selection —; orthogonal projection —; overloaded —; p-regular —; Point Taylor —; properly quasimonotone —; pseudomonotone —; quasimonotone —; resolvent —; selection —; semistrictly quasimonotone —; strictly monotone —; strictly pseudomonotone —; strictly quasimonotone —; univariate interval Newton —; upper hemicontinuous —
operator on a Banach space *see:* monotone —
operator decomposition
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
operator for a matroid *see:* closure —

operator overloading
[65H99, 65K05, 65K99, 90C30]
(*see:* **Automatic differentiation: point and interval**; **Automatic differentiation: point and interval taylor operators**)
operator splitting
[90C30]
(*see:* **Cost approximation algorithms**)
operator splitting
[90C30]
(*see:* **Cost approximation algorithms**)
operator splitting algorithm
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
operator topology *see:* strong —
operators *see:* Automatic differentiation: point and interval taylor —; design of —; genetic —
operators in best approximation by bounded or continuous functions *see:* Lipschitzian —
OPFAD
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**)
opposite of a signed set
[90C09, 90C10]
(*see:* **Oriented matroids**)
OPRAD
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**)
Opt *see:* 2- —; k- —
opt heuristic *see:* R- —
Opt Matching (ROM) *see:* recursive —
opt neighborhood *see:* 2- —
optical bandwidth
[05C85]
(*see:* **Directed tree networks**)
optical networks *see:* all- —; Integer linear programs for routing and protection problems in —
optics *see:* nonlinear —; Optimal design in nonlinear —
optima *see:* local —
optimal
[05B35, 65K05, 68R10, 9008, 90C05, 90C20, 90C26, 90C27, 90C33, 90C59]
(*see:* **Branchwidth and branch decompositions**; **Criss-cross pivoting rules**; **Variable neighborhood search methods**)
optimal *see:* globally —; locally —; Pareto —
optimal algorithm
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
optimal algorithms
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
optimal assignment
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
optimal assignment problem
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
optimal basis
[90C05, 90C06, 90C08, 90C10, 90C11, 90C33]
(*see:* **Integer programming: branch and bound methods**; **Pivoting algorithms for linear programming generating two paths**)
optimal componentwise bound
[15A99, 65G20, 65G30, 65G40, 90C26]
(*see:* **Interval linear systems**)
optimal control
[03H10, 49J15, 49J27, 49K15, 90C26, 90C34, 93-XX, 93C10]
(*see:* **Boundary condition iteration BCI**; **Invexity and its applications**; **Pontryagin maximum principle**; **Semi-infinite programming and control problems**)
optimal control
[49-XX, 49J15, 49K15, 60Jxx, 65Lxx, 90C26, 91B32, 92D30, 93-XX, 93C10]
(*see:* **Invexity and its applications**; **Optimal control of a flexible arm**; **Pontryagin maximum principle**; **Resource allocation for epidemic control**)
optimal control *see:* continuous-time —; Discrete-Time —; Dynamic programming: continuous-time —; Dynamic programming and Newton's method in unconstrained —; parametric —; time —; unconstrained —
optimal control applications *see:* Dynamic programming: —
optimal control with first order differential equations *see:* Duality in —
Optimal control of a flexible arm
(93-XX)
(*referred to in:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
optimal control policy
[90C30]
(*see:* **Suboptimal control**)
optimal control problem
[93-XX]
(*see:* **Boundary condition iteration BCI**)
optimal control problem
[49K20, 49M99, 90C55]

(*see:* **Sequential quadratic programming: interior point methods for distributed optimal control problems**)

optimal control problem *see:* mixed integer —; time —

optimal control problems *see:* discretized —; distributed —; Sequential quadratic programming: interior point methods for distributed —

optimal degree of flexibility
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)

optimal design
[34A55, 78A60, 90C26, 90C30, 90C31]
(*see:* **Bilevel programming: introduction, history and overview**; **Optimal design in nonlinear optics**)

optimal design
[34A55, 78A60, 90C25, 90C27, 90C30, 90C90]
(*see:* **Optimal design in nonlinear optics**; **Semidefinite programming and structural optimization**)

optimal design *see:* D- —; global —; Multilevel methods for —

Optimal design of composite structures
(90C29, 90C26)
(*referred to in:* **Design optimization in computational fluid dynamics**; **Interval analysis: application to chemical engineering design problems**; **Multidisciplinary design optimization**; **Multilevel methods for optimal design**; **Optimal design in nonlinear optics**; **Structural optimization: history**)
(*refers to:* **Bilevel programming: applications in engineering**; **Design optimization in computational fluid dynamics**; **Global optimization: hit and run methods**; **Interval analysis: application to chemical engineering design problems**; **Multidisciplinary design optimization**; **Multilevel methods for optimal design**; **Optimal design in nonlinear optics**; **Random search methods**; **Structural optimization: history**)

Optimal design in nonlinear optics
(34A55, 90C30, 78A60)
(*referred to in:* **Design optimization in computational fluid dynamics**; **Interval analysis: application to chemical engineering design problems**; **Multidisciplinary design optimization**; **Multilevel methods for optimal design**; **Optimal design of composite structures**; **Structural optimization: history**)
(*refers to:* **Bilevel programming: applications in engineering**; **Design optimization in computational fluid dynamics**; **Interval analysis: application to chemical engineering design problems**; **Multidisciplinary design optimization**; **Multilevel methods for optimal design**; **Optimal design of composite structures**; **Structural optimization: history**)

optimal design problems
[49K20, 49M99, 90C55]
(*see:* **Sequential quadratic programming: interior point methods for distributed optimal control problems**)

optimal distance *see:* method of —

optimal distribution of efforts
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)

optimal experimental design
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)

optimal face
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)

optimal face
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)

optimal flowsheets *see:* sensitivity of —

optimal gambling
[49L20, 90C40]
(*see:* **Dynamic programming: undiscounted problems**)

optimal indexing vocabulary
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)

optimal integral bounds subject to moment conditions
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)

optimal investments
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)

Optimal investments
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)

optimal Lipschitzian selection operator
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)

optimal number of clusters *see:* Determining the —

optimal parameter
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)

optimal parameter
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)

optimal parameter *see:* globally —; locally —

optimal partitioning algorithm *see:* nearest insertion —

optimal path
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)

optimal path
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)

optimal path *see:* co- —

Optimal planning of offshore oilfield infrastructure

optimal policies
[90B50]
(*see:* **Inventory management in supply chains**)

optimal policies *see:* (s,S) —

optimal ratio *see:* method of —

optimal relaxation
[90C30]
(*see:* **Relaxation in projection methods**)

optimal rule *see:* Bayes —

Optimal sensor scheduling

optimal shape design
[49J20, 49J52]
(*see:* **Shape optimization**)
optimal shapes *see:* design of —
optimal solution
[9008, 90C06, 90C10, 90C11, 90C25, 90C26, 90C27, 90C30, 90C31, 90C57, 90C59, 90C90]
(*see:* **Lagrangian multipliers methods for convex programming**; **Modeling difficult optimization problems**; **Robust global optimization**; **Variable neighborhood search methods**)
optimal solution *see:* essential —; global —; locally —; M-Pareto —; Pareto —; quasi- —; strongly stable —; weakly Pareto —
optimal solution mapping
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
optimal solution of a program
[90C06]
(*see:* **Saddle point theory and optimality conditions**)
optimal solution set *see:* Pareto —
optimal solutions *see:* jumps of —; Pareto —
optimal solutions, properties *see:* Multi-objective optimization: pareto —
Optimal solvent design approaches
(65K99)
optimal spanning tree structure
[90C35]
(*see:* **Minimum cost flow problem**)
optimal state space search algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
optimal steady state
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
optimal stopping
[49L20, 90C40]
(*see:* **Dynamic programming: undiscounted problems**)
optimal stopping
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**)
optimal stopping: numerical methods *see:* Stochastic —
optimal stopping: problem formulations *see:* Stochastic —
optimal subset
[90C09, 90C10]
(*see:* **Matroids**)
optimal substructure property
[90C09, 90C10]
(*see:* **Matroids**)
optimal trajectories *see:* Turnpike theory: stability of —
optimal trajectory
[49J15, 49K15, 93C10]
(*see:* **Pontryagin maximum principle**)
Optimal triangulations
(68Q20)
optimal triangulations
[68Q20]
(*see:* **Optimal triangulations**)
optimal value bounds *see:* computable —
optimal value function
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Image space approach to optimization**; **Nondifferentiable optimization: parametric programming**; **Sensitivity and stability in NLP: continuity and differential stability**)
optimal value functions
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
optimal vertex *see:* co- —
optimal vocabulary
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
optimality *see:* finite —; first order —; global —; guaranteed bound to —; high-order necessary conditions for —; k- —; overtaking —; parametric approach to —; Pareto —; principle of —; test of —; weak principle of —; weakly overtaking —; worst-case —
optimality for abnormal points *see:* High-order necessary conditions for —
optimality analyses *see:* post- —
optimality analysis *see:* post- —
optimality in bilinear programming
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
optimality condition *see:* Kuhn–Tucker —; necessary —; second order —; sufficient —
optimality condition without using (sub)gradients parametric representations *see:* necessary —
optimality conditions
[90C06, 90C26, 90C31, 90C39]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Saddle point theory and optimality conditions**; **Second order optimality conditions for nonlinear optimization**; **Sensitivity and stability in NLP: continuity and differential stability**)
optimality conditions
[46A20, 49J15, 49K15, 49K27, 49K40, 49M37, 52A01, 65K05, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C34, 91B28, 93C10]
(*see:* **Bilevel programming: optimality conditions and duality**; **Composite nonsmooth optimization**; **First order constraint qualifications**; **Generalized concavity in multi-objective optimization**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Pontryagin maximum principle**; **Quadratic programming with bound constraints**; **Second order constraint qualifications**; **Semi-infinite programming and applications in finance**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: continuity and differential stability**; **Smooth nonlinear nonconvex optimization**)
optimality conditions *see:* Equality-constrained nonlinear programming: KKT necessary —; first order necessary —; first order and second order —; fritz John necessary —; generalized necessary —; Generalized semi-infinite programming: —; Karush–Kuhn–Tucker —; KKT —; KKT necessary —; Kuhn–Tucker —; Kuhn–Tucker necessary —; necessary —; necessary and sufficient —; Quasidifferentiable optimization: —; Saddle point theory

and —; second order necessary and sufficient —; Semi-infinite programming: second order —; sufficient —
optimality conditions and duality *see:* Bilevel programming: —
optimality conditions for nonlinear optimization *see:* Second order —
optimality conditions and stability *see:* Semidefinite programming: —
Optimality criteria for multiphase chemical equilibrium (49K99, 65K05, 80A10)
(*referred to in:* **Global optimization: application to phase equilibrium problems**; **Global optimization in phase and chemical reaction equilibrium**)
(*refers to:* **Global optimization: application to phase equilibrium problems**; **Global optimization in phase and chemical reaction equilibrium**)
optimality criterion
[90C26, 90C90]
(*see:* **Structural optimization: history**)
optimality cut
[90C15]
(*see:* **L-shaped method for two-stage stochastic programs with recourse**)
optimality in a game
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
optimality of MODP *see:* principle of Pareto —
optimality in parametric programming
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
optimality principal *see:* proximate —
optimality principle *see:* proximate —
optimality sensitivity analysis *see:* post- —
optimally
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
optimally scaled subclass
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
optimization
[65L99, 90C06, 90C10, 90C11, 90C30, 90C57, 90C90, 93-XX]
(*see:* **Modeling difficult optimization problems**; **Optimization strategies for dynamic systems**; **Simulated annealing methods in protein folding**)
optimization
[01A99, 05-XX, 15A99, 49K99, 62G07, 62G30, 65D10, 65G20, 65G30, 65G40, 65K05, 65K10, 68Q25, 68Q99, 68W10, 80A10, 90B15, 90B80, 90B85, 90C05, 90C06, 90C08, 90C10, 90C11, 90C15, 90C26, 90C27, 90C30, 90C31, 90C33, 90C35, 90C39, 90C52, 90C53, 90C55, 90C90, 90Cxx, 91B28, 92B05, 92C05, 94A17]
(*see:* **ABS algorithms for optimization**; **Adaptive simulated annealing and its application to protein folding**; **Assignment and matching**; **Asynchronous distributed optimization algorithms**; **Auction algorithms**; **Biquadratic assignment problem**; **Branch and price: Integer programming with column generation**; **Communication network assignment problem**; **Decomposition principle of linear programming**; **Design optimization in computational fluid dynamics**; **Frequency assignment problem**; **Genetic algorithms**; **Genetic algorithms for protein structure prediction**; **Graph coloring**; **History of optimization**; **Homogeneous selfdual methods for linear programming**; **Implicit lagrangian**; **Interval linear systems**; **Invexity and its applications**; **Isotonic regression problems**; **Jaynes' maximum entropy principle**; **Multiplicative programming**; **Neuro-dynamic programming**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Optimality criteria for multiphase chemical equilibrium**; **Optimization in medical imaging**; **Optimization software**; **Overdetermined systems of linear equations**; **Probabilistic constrained linear programming: duality theory**; **Quadratic semi-assignment problem**; **Robust optimization**; **Saddle point theory and optimality conditions**; **Simulated annealing**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic network problems: massively parallel solution**; **Symmetric systems of linear equations**; **Two-stage stochastic programs with recourse**)
optimization *see:* a priori —; ABS algorithms for —; Adaptive convexification in semi-infinite —; Airline —; Algorithmic improvements using a heuristic parameter, reject index for interval —; algorithms for entropy —; Bayesian global —; beam angle —; beam angle selection and wedge orientation —; beam weight —; bilevel —; Bilevel programming: global —; black-box —; black-box global —; branch and bound for unconstrained —; collaborative —; combinatorial —; Complexity classes in —; Composite nonsmooth —; Computer implementation of —; concurrent subspace —; constrained —; constrained global —; continuous —; continuous global —; convex —; convex combinatorial —; Convex discrete —; convex quadratic —; Copositive —; Cutting plane methods for global —; d.c. —; Decomposition in global —; Derivative-free methods for non-smooth —; deterministic —; Differential equations and global —; Dini and Hadamard derivatives in —; direct search —; disaggregation in multi-objective —; Discontinuous —; discrete —; discrete decisions in dynamic —; Discrete stochastic —; Distance dependent protein force field via linear —; dM —; Domination analysis in combinatorial —; Duality gaps in nonconvex —; duality theorem for linear —; Duality theory: biduality in nonconvex —; duality theory for entropy —; Duality theory: monoduality in convex —; Duality theory: triduality in global —; Dynamic —; encyclopedia of —; engineering —; entropy —; equality-constrained —; Evolutionary algorithms in combinatorial —; Fejér monotonicity in convex —; Financial —; flowsheet —; fluence map —; fractional —; Fractional combinatorial —; general constrained —; Generalized concavity in multi-objective —; global —; global nonlinear —; Global pairwise protein sequence alignment via mixed-integer linear —; graph —; hierarchical —; History of —; Hyperplane arrangements in —; hypodifferentiable —; Image space approach to —; Inequality-constrained nonlinear —; infinite-dimensional —; information-based —; Information-based complexity and information-based —; input —; interior point algorithms for entropy —; Interval analysis: parallel methods for global —; Interval analysis:

unconstrained and constrained —; Interval global —; isotone —; Lagrangian relaxation with subgradient —; large scale —; large-scale combinatorial —; Large scale unconstrained —; linear —; linear fractional combinatorial —; Lipschitz —; local —; LP strategy for interval-Newton method in deterministic global —; marginal function —; mixed discrete-continuous global —; Mixed Integer —; Mixed Integer Bilevel —; mixed integer dynamic —; mixed integer nonlinear —; Mixed integer nonlinear bilevel programming: deterministic global —; molecular —; monotonic —; Monte-Carlo simulations for stochastic —; Multi-class data classification via mixed-integer —; multi-extremal global —; multi-objective —; multi-objective combinatorial —; multi-objective convex —; multidisciplinary —; Multidisciplinary design —; multilevel —; multiperiod —; multistage —; Nested partitions —; network —; Neural networks for combinatorial —; New hybrid conjugate gradient algorithms for unconstrained —; nonconvex —; nondifferentiable —; nondifferentiable convex —; nondifferential —; nonlinear —; nonlinear parametric —; nonlinearly constrained —; nonsmooth —; nonsmooth analysis and —; nonsmooth nonconvex —; Numerical methods for unary —; off-line process —; on-line process —; one-parametric semi-infinite —; ordinal —; parallel —; parametric —; parametric approach to fractional —; Parametric mixed integer nonlinear —; parametric nonlinear —; path following algorithm for entropy —; Peptide identification via mixed-integer —; Performance profiles of conjugate-gradient algorithms for unconstrained —; Plant layout problems and —; portfolio —; process —; Quasidifferentiable —; Reformulation-linearization technique for global —; Replicator dynamics in combinatorial —; Reverse convex —; Robust —; Robust global —; sample-path —; Second order optimality conditions for nonlinear —; semi-infinite —; Semidefinite programming and structural —; separable —; sequential approximate —; Set-valued —; Shape —; simulation-based —; sizing —; smooth nonlinear —; Smooth nonlinear nonconvex —; Smoothing methods for semi-infinite —; SSC minimization algorithms for nonsmooth and stochastic —; stochastic —; stochastic combinatorial —; stochastic global —; stochasticglobal —; structural —; structural shape —; structural topology —; subgradient —; supply chain —; system- —; tailored —; Theorems of the alternative and —; Topological derivative in shape —; topology —; Topology of global —; Two-level —; type of —; unary —; unbounded —; unconstrained —; unconstrained dual in entropy —; unconstrained global —; uniform fractional combinatorial —; unstructured —; user- —; vector —; Wastewater system —; zero-one —

Optimization in ad hoc networks
(68M12, 90B18, 90C11, 90C30)

optimization algorithm *see:* αBB global —; deterministic global —; Direct global —; MINLP: branch and bound global —

optimization algorithm (definition)
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)

optimization algorithms
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)

optimization algorithms *see:* Asynchronous distributed —; unconstrained —

optimization algorithms for financial planning problems *see:* Global —

optimization: algorithms for hypodifferentiable functions *see:* Quasidifferentiable —

optimization: algorithms for QD functions *see:* Quasidifferentiable —

optimization algorithms in resource allocation problems *see:* Combinatorial —

optimization with αBB *see:* MINLP: global —

optimization in the analysis and management of environmental systems *see:* Global —

optimization: application to phase equilibrium problems *see:* Global —

optimization: applications *see:* Continuous global —; Quasidifferentiable —

optimization: applications to thermoelasticity *see:* Quasidifferentiable —

optimization approach *see:* Multiple minima problem in protein folding: αBB global —

optimization approach to clustering *see:* Nonsmooth —

optimization approach to image reconstruction from projection data
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)

optimization approaches *see:* Statistical classification: —

Optimization based frameworkfor radiation therapy
(68W01, 90-00, 90C90, 92-08, 92C50)
(*referred to in:* **Beam selection in radiotherapy treatment design**)

optimization-based methods *see:* Disease diagnosis: —

optimization based on statistical models *see:* Global —

Optimization-based visualization
(91C15, 65K05, 90C30, 90C27, 90C57)
(*refers to:* **Continuous global optimization: models, algorithms and software**; **Dynamic programming in clustering**; **Evolutionary algorithms in combinatorial optimization**; **Integer programming**; **Integer programming: branch and bound methods**; **Nonlinear least squares: Newton-type methods**; **Simulated annealing**)

optimization in batch design under uncertainty *see:* Global —

optimization in binary star astronomy *see:* Global —

Optimization in boolean classification problems
(90C09, 90C10)
(*referred to in:* **Alternative set theory**; **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Mixed integer classification problems**; **Optimization in classifying text documents**; **Statistical classification: optimization approaches**)
(*refers to:* **Alternative set theory**; **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Linear programming models for classification**; **Mixed integer classification problems**; **Optimization in classifying text**

documents; **Statistical classification: optimization approaches**)
optimization: calculus of quasidifferentials *see:* Quasidifferentiable —
optimization in CFD *see:* design —
Optimization in classifying text documents
(90C09, 90C10)
(*referred to in:* **Alternative set theory**; **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Optimization in boolean classification problems**; **Statistical classification: optimization approaches**)
(*refers to:* **Alternative set theory**; **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**; **Finite complete systems of many-valued logic algebras**; **Inference of monotone boolean functions**; **Linear programming models for classification**; **Mixed integer classification problems**; **Optimization in boolean classification problems**; **Statistical classification: optimization approaches**)
optimization: codifferentiable functions *see:* Quasidifferentiable —
optimization in computational fluid dynamics *see:* Design —
optimization of computational performance
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
optimization computer implementation example
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
optimization: cutting angle method *see:* Global —
optimization: cutting plane methods *see:* Nondifferentiable —
Optimization and decision support systems
(90B50)
(*referred to in:* **Data envelopment analysis**)
optimization and decision support systems *see:* Multi-objective —
optimization: definition (colloquial)
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
optimization: Dini derivatives, clarke derivatives *see:* Quasidifferentiable —
optimization: A disjunctive cutting plane approach *see:* Mixed-integer nonlinear —
optimization in document classification
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
optimization of dynamical systems *see:* Interval analysis for —
optimization: embeddings, path following and singularities *see:* Parametric —
optimization: envelope representation *see:* Global —
Optimization with equilibrium constraints: A piecewise SQP approach
(90C30, 90C33)
(*referred to in:* **Feasible sequential quadratic programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Bilevel programming: introduction, history and overview**; **Feasible sequential quadratic programming**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Linear complementarity problem**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**; **Variational inequalities**)
optimization: exact penalty methods *see:* Quasidifferentiable —
optimization: feasibility test and flexibility index *see:* Bilevel —
optimization: filled function methods *see:* Global —
optimization: functional forms *see:* Global —
optimization: g-αBB approach *see:* Global —
optimization game *see:* combinatorial —
optimization games *see:* Combinatorial —
optimization in generalized geometric programming *see:* Global —
optimization of heat exchanger networks *see:* Global —
optimization: history *see:* Structural —
optimization: hit and run methods *see:* Global —
optimization homotopies
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
optimization for image reconstruction *see:* Entropy —; finite-dimensional models for entropy —; vector-space models for entropy —
optimization in industrial problems *see:* SQP —
optimization: interaction of design and control *see:* Multi-objective —
optimization; Interactive methods for preference value functions *see:* Multi-objective —
optimization: interior point methods *see:* Entropy —
optimization: interval analysis and balanced interval arithmetic *see:* Global —
optimization: lagrange duality *see:* Multi-objective —
optimization in Lennard–Jones and morse clusters *see:* Global —
Optimization in leveled graphs
(90C35)
(*referred to in:* **Graph planarization**; **Integer programming**)
(*refers to:* **Graph planarization**; **Integer programming**)
optimization in location problems *see:* Global —
optimization in mechanics *see:* Multilevel —
optimization in medical image processing
[90C90]
(*see:* **Optimization in medical imaging**)

Optimization in medical imaging
(90C90)
(*referred to in:* **Entropy optimization: shannon measure of entropy and its properties**; **Maximum entropy principle: image reconstruction**)
(*refers to:* **Entropy optimization: shannon measure of entropy and its properties**; **Genetic algorithms**; **Linear programming**; **Maximum entropy principle: image reconstruction**; **Simulated annealing**)
optimization method *see:* heuristic —; QBB global —
optimization methods *see:* Bisection global —; Credit rating and —; Nondifferentiable optimization: subgradient —; Shape selective zeolite separation and catalysis: —; Solving hemivariational inequalities by nonsmooth —
optimization methods for harmonic retrieval *see:* Global —
optimization methods for systems of nonlinear equations *see:* Global —
optimization: minimax problems *see:* Nondifferentiable —
optimization: mixed-integer linear programs *see:* Robust —
optimization model *see:* continuous global —
optimization modeling *see:* Emergency evacuation —
optimization modeling framework *see:* multiperiod —
optimization: models, algorithms and software *see:* Continuous global —
optimization models for data classification *see:* Deterministic and probabilistic —
optimization in multiplicative programming *see:* Global —
optimization in neural network training *see:* Unconstrained —
optimization: a new paradigm *see:* Modeling languages in —
optimization: Newton–Cauchy framework *see:* Unconstrained nonlinear —
optimization: Newton method *see:* Nondifferentiable —
optimization with noises
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms**)
optimization of nonlinear problems *see:* Simultaneous estimation and —
Optimization in operation of electric and energy power systems
(90C35, 90C30, 90C10)
(*referred to in:* **Derivatives of markov processes and their simulation**; **Derivatives of probability and integral functions: general theory and examples**; **Derivatives of probability measures**; **Discrete stochastic optimization**)
optimization: optimality conditions *see:* Quasidifferentiable —
optimization oracle *see:* linear discrete —
optimization: p-αBB approach *see:* Global —
optimization paradigm
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
optimization: parameter estimation *see:* Entropy —
optimization: parametric programming *see:* Nondifferentiable —
optimization: pareto optimal solutions, properties *see:* Multi-objective —
optimization in phase and chemical reaction equilibrium *see:* Global —
optimization of planar multilayered dielectric structures *see:* Global —
optimization problem
[65K10, 65M60, 90C26, 90C31]
(*see:* **Robust global optimization**; **Variational inequalities**)
optimization problem
[65K10, 65M60]
(*see:* **Variational inequalities**)
optimization problem *see:* $C^{1,1}$ —; canonical monotonic —; combinatorial —; convex —; dual —; fractional combinatorial —; global —; global constrained —; global unconstrained —; integer —; integral linear fractional combinatorial —; Lagrangian dual —; linear —; max-min-max —; mixed integer —; nonconvex —; nonlinear —; nonseparable —; one-parametric finite —; parametric —; primal —; semi-infinite —; separable —; set-valued —; standard quadratic —; stochastic dynamic —; unary —; unconstrained —
optimization problem in standard form *see:* linear —
optimization problems
[68Q25, 68R10, 68W40, 90C26, 90C27, 90C30, 90C59, 90C90]
(*see:* **Domination analysis in combinatorial optimization**; **Planning in the process industry**; **Smooth nonlinear nonconvex optimization**; **Successive quadratic programming: applications in the process industry**)
optimization problems *see:* Approximations to robust conic —; combinatorial —; computational complexity of —; Continuous reformulations of discrete-continuous —; Convex envelopes in —; Decomposition algorithms for the solution of multistage mean-variance —; discrete monotonic —; discretization of —; General moment —; Laplace method and applications to —; linearly constrained —; Modeling difficult —; nonlinear —; nonsmooth —; Price of robustness for linear —; semi-infinite —; stability analysis of —; stochastic linear —
optimization problems: algorithms *see:* Standard quadratic —
optimization problems: applications *see:* Standard quadratic —
optimization problems: theory *see:* Standard quadratic —
Optimization problems in unit-disk graphs
(05C85, 05C69, 05C15, 05C62, 90C27, 90C59)
(*referred to in:* **Broadcast scheduling problem**)
optimization procedure *see:* Direct search Luus—Jaakola —; LJ —
optimization process *see:* automated design —
optimization in protein folding *see:* Global —
optimization: relaxation methods *see:* Nondifferentiable —
optimization: sensitivity *see:* Parametric global —
optimization: shannon measure of entropy and its properties *see:* Entropy —
optimization-simulation
(*see:* **Emergency evacuation, optimization modeling**)
Optimization software
(90C30, 90C26, 90C10)
(*referred to in:* **Continuous global optimization: models, algorithms and software**; **Large scale unconstrained optimization**; **Modeling languages in optimization: a new paradigm**)
(*refers to:* **Continuous global optimization: models, algorithms and software**; **Large scale unconstrained optimization**; **Modeling languages in optimization: a new paradigm**)

optimization: stability of dynamic systems *see:* Quasidifferentiable —
optimization: stopping rules *see:* Stochastic global —
Optimization strategies for dynamic systems
(93-XX, 65L99)
(*referred to in:* **Control vector iteration CVI**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **Quasidifferentiable optimization: stability of dynamic systems**; **Suboptimal control**)
(*refers to:* **Dynamic programming: continuous-time optimal control**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Dynamic programming: stochastic shortest path problems**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **Quasidifferentiable optimization: stability of dynamic systems**)
optimization: subgradient optimization methods *see:* Nondifferentiable —
optimization system
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
optimization system *see:* generalized network —
optimization techniques *see:* Estimating data for multicriteria decision making problems: —; Load balancing for parallel —
Optimization techniques for minimizing the energy function
[90C90]
(*see:* **Optimization in medical imaging**)
Optimization techniques for phase retrieval based on single-crystal X-ray diffraction data
optimization: theorems of the alternative *see:* Linear —
optimization: tight convex underestimators *see:* Global —
optimization over a trajectory
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
optimization: two-phase methods *see:* Stochastic global —
optimization over unbounded domains *see:* global —
optimization under network constraints
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
optimization using space filling *see:* Global —
optimization using terrain/funneling methods *see:* Multi-scale global —
optimization: variational formulations *see:* Quasidifferentiable —
optimization in a vector space
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
Optimization in water resources
(90C30, 90C35)
(*referred to in:* **Global optimization in the analysis and management of environmental systems**)
(*refers to:* **Global optimization in the analysis and management of environmental systems**)
optimization in water resources *see:* stochastic approach to —
optimization in Weber's problem with attraction and repulsion *see:* Global —
optimization in well scheduling *see:* Mixed integer —
optimized transportation network *see:* system- —; user- —
optimizer
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
optimizer *see:* global —; strict local —
optimizer coupling *see:* model/ —
optimizing environment *see:* system- —; user- —
Optimizing facility location with euclidean and rectilinear distances
(90B80, 90B85)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
optimum *see:* conditions for a constrained —; conditions for an unconstrained —; constrained global —; global —; local —; Pareto —; unconstrained —
optimum search *see:* global —
optimum search with enhanced positioning *see:* Gene clustering: A novel decomposition-based clustering approach: global —
optimum solution
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
option
[91B50]
(*see:* **Financial equilibrium**)

option
[91B50]
(*see:* **Financial equilibrium**)
(or disk) representation *see:* geometric —
OR-ing
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
OR-methodology
[90B80, 90B85]
(*see:* **Warehouse location problem**)
(or Minty) GVI *see:* dual —
oracle
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**; **Inference of monotone boolean functions**; **Maximum constraint satisfaction: relaxations and upper bounds**)
oracle *see:* augmentation —; comparison —; linear discrete optimization —; membership —; oriented augmentation —
orbit
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
orbit *see:* satellite —
orbital period
[26A24, 65K99, 85-08]
(*see:* **Automatic differentiation: geometry of satellites and tracking stations**)
orbits determination
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
order *see:* antisymmetric partial —; first order theory of real addition with —; interval —; lexicographical —; Löwner partial —; partial —; pre- —; pseudo- —; quasi- —; semi- —; weak —
order active *see:* p- —
order adjoints *see:* second —
order approximating cone *see:* high- —; tangent high- —
order approximating cone of decrease *see:* high- —
order approximating cones *see:* feasible high- —; tangent high- —
order approximating curve *see:* feasible high- —; high- —; tangent high- —
order approximating vector *see:* feasible high- —
order approximating vector of decrease *see:* high- —
order approximating vectors *see:* high- —
order approximation *see:* second —
order approximation of a function *see:* first —
order changes *see:* up to first —
order closure of a relation *see:* local pre- —; pre- —
order codifferential *see:* second —
Order complementarity
(90C33)
(*referred to in:* **Continuous reformulations of discrete-continuous optimization problems**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Linear complementarity problem**; **Principal pivoting methods for linear complementarity problems**; **Topological methods in complementarity theory**)
(*refers to:* **Convex-simplex algorithm**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Parametric linear programming: cost simplex algorithm**; **Principal pivoting methods for linear complementarity problems**; **Sequential simplex method**; **Topological methods in complementarity theory**)
order complementarity
[90C33]
(*see:* **Order complementarity**)
order complementarity problem
[90C33]
(*see:* **Order complementarity**)
order complementarity problem *see:* general —; generalized —; generalized linear —; implicit general —; infinite-dimensional generalized —; linear —; nonlinear —
order cone
[90C26]
(*see:* **Invexity and its applications**)
order cone *see:* second —
order cones of decrease *see:* high- —
order constrained hierarchical clustering
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
order constrained partitioning
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
order constraint qualification *see:* first —; second —
order constraint qualifications *see:* First —; Second —
order CQ *see:* First —; second —
order critical direction *see:* high- —
order decomposition of a function *see:* second —
order derivatives *see:* higher- —
order differential equations *see:* Duality in optimal control with first —
order directional derivative *see:* generalized second —
order directional derivatives *see:* high- —; higher- —
order earliness *see:* minimization of —
order feasible cones *see:* high- —
order feasible set *see:* high- —; p- —
order form of coordinates *see:* kth —
order generalization of Lyusternik theorem *see:* high- —
order of a graph
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
order growth *see:* second —
order hyperdifferential *see:* second —
order hypodifferential *see:* kth —; second —
order of an inclusion function
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
order isotonic regression *see:* simple —
order KKT conditions *see:* first —
order Lagrangian theory of CNSO problems *see:* second —
order local maximum principle *see:* high- —
order local maximum principle for Lagrangian problems *see:* high- —

order local minimum condition *see:* high- —
order of magnitude
[90C60]
(*see:* **Computational complexity theory**)
order maximum principle for abnormal extremals *see:* High- —
order necessary condition *see:* first —; second —
order necessary conditions *see:* first —; second —
order necessary conditions for optimality *see:* high- —
order necessary conditions for optimality for abnormal points *see:* High- —
order necessary optimality conditions *see:* first —
order necessary and sufficient optimality conditions *see:* second —
order optimality *see:* first —
order optimality condition *see:* second —
order optimality conditions *see:* first order and second —; Semi-infinite programming: second —
order optimality conditions for nonlinear optimization *see:* Second —
order partial differential equations *see:* First —
order preserving assignment problem
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
order procedures *see:* second —
order quantity *see:* economic —
order regular set *see:* second —
order relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
order relation *see:* local —; partial —; pre- —
order restricted statistical inference
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
order restriction
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
order and second order optimality conditions *see:* first —
order set of decrease *see:* high- —
order spectrum *see:* higher- —
order statistics *see:* higher- —; Signal processing with higher —
order sufficiency *see:* second —; strong second —
order sufficient condition *see:* general second —; general strong second —; second —; strong second —
order sufficient conditions *see:* second —
order of a T-coloring frequency assignment
[05-XX]
(*see:* **Frequency assignment problem**)
order tangent approximating vector *see:* high- —
order tangent set *see:* first —; second —
order tangent sets *see:* high- —
order Taylor series expansion *see:* first —
order theory of real addition with order *see:* first —
ordered normed linear spaces *see:* Best approximation in —
ordered partition
[62H30, 90C27, 90C39]
(*see:* **Assignment methods in clustering**; **Dynamic programming in clustering**)
ordered partitioning
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
ordered set
[90C29]
(*see:* **Preference modeling**)
ordered spaces *see:* semi- —
ordered vector spaces
[90C33]
(*see:* **Order complementarity**)
Ordering *see:* cobipartite neighborhood edge elimination —; lexicographic —; linear —; minimum degree —; zero-Inventory —
ordering on binary vectors
[90C09]
(*see:* **Inference of monotone boolean functions**)
ordering cones
[90C29]
(*see:* **Vector optimization**)
ordering for n-dimensional vectors *see:* lexicographical —
ordering and perturbation *see:* lexicographic —
ordering policy *see:* one-for-one —
ordering problem *see:* Linear —
orders *see:* partial —
ordinal argument
[90-XX]
(*see:* **Outranking methods**)
ordinal criterion
[90C29, 91A99]
(*see:* **Preference disaggregation**)
ordinal optimization
[90C15, 90C27]
(*see:* **Discrete stochastic optimization**)
ordinal regression
[90C26, 91B28]
(*see:* **Portfolio selection and multicriteria analysis**)
ordinary
[65G20, 65G30, 65G40]
(*see:* **Interval analysis: systems of nonlinear equations**)
ordinary differential equations
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
ordinary differential equations
[49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX]
(*see:* **Resource allocation for epidemic control**)
ordinary differential equations *see:* Eigenvalue enclosures for —
ordinary NP-complete reductions
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
ordinary NP-completeness
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
ordinary NP-completeness
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
organization *see:* self- —
orientable matroid
[90C09, 90C10]
(*see:* **Oriented matroids**)

orientation
[90B35]
(*see:* **Job-shop scheduling problem**)
orientation *see:* circuit —; complete —
orientation optimization *see:* beam angle selection and wedge —
orientation of an oriented matroid *see:* basis —
oriented approach *see:* equation —
oriented augmentation oracle
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
oriented branch and bound method *see:* arc —; node —
oriented construction procedure *see:* arc —; node —
oriented differentiation *see:* goal- —
oriented matroid *see:* acyclic —; bases of an —; basis orientation of an —; totally acyclic —; vector of an —
Oriented matroids
(90C09, 90C10)
(*referred to in:* **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Matroids**)
(*refers to:* **Matroids**)
oriented matroids
[05B35, 65K05, 90C05, 90C09, 90C10, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**; **Oriented matroids**)
oriented matroids
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Lexicographic pivoting rules**)
oriented matroids *see:* axiom systems for —
origin *see:* neighbors of the —
origin tracing
(*see:* **Planning in the process industry**)
Orlicz theorem *see:* Mazur– —
Orlicz version of the Hahn–Banach theorem *see:* Mazur– —
Orlik–Solomon algebra
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
Orlik–Solomon algebra
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
orthogonal collocation
[90C30]
(*see:* **Suboptimal control**)
orthogonal condition
[90C30]
(*see:* **Image space approach to optimization**)
orthogonal factorization
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **QR factorization**)
orthogonal factorization
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **QR factorization**)
orthogonal factorization *see:* complete —
orthogonal matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
orthogonal matrix
[15A39, 90C05]
(*see:* **Farkas lemma**)
orthogonal matroid
[90C09, 90C10]
(*see:* **Oriented matroids**)
orthogonal polynomials
[33C45, 65F20, 65F22, 65K10, 90C30]
(*see:* **Generalized total least squares**; **Least squares orthogonal polynomials**)
orthogonal polynomials
[33C45, 65F20, 65F22, 65K10]
(*see:* **Least squares orthogonal polynomials**)
orthogonal polynomials *see:* Least squares —; least squares formal —
orthogonal projection
[65K10, 65M60]
(*see:* **Variational inequalities**)
orthogonal projection operator
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
orthogonal search directions
[90C30]
(*see:* **Rosenbrock method**)
orthogonal signed sets
[90C09, 90C10]
(*see:* **Oriented matroids**)
orthogonal transform
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **QR factorization**)
orthogonal transformations *see:* elementary —
Orthogonal triangularization
(65F25, 15A23, 65F05, 65F20, 65F22)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Linear programming**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
orthogonality conditions on multipliers
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
orthogonalization *see:* classical Gram–Schmidt —; Gram–Schmidt —; modified Gram–Schmidt —
orthogonalization scheme *see:* sequential row —
orthogonally scaled subclass
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
orthonormal representation
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
orthonormalization *see:* hybrid —

OS
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
OSLE
[65D10, 65K05]
(*see:* **Overdetermined systems of linear equations**)
Out *see:* first-In-First- —; pricing- —
out events *see:* drought —
out rule *see:* first-in last- —
out trip *see:* pull- —
outcome
[90C26]
(*see:* **Global optimization using space filling**)
outcome set
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
outcome space
[65K05, 90B50, 90C05, 90C29, 91B06]
(*see:* **Multi-objective optimization and decision support systems**)
outer approximation
[49M37, 90C05, 90C11, 90C25, 90C26, 90C29, 90C30, 90C34, 90C90]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Concave programming**; **MINLP: branch and bound methods**; **MINLP: design and scheduling of batch processes**; **Mixed integer nonlinear programming**; **Multi-objective optimization: interaction of design and control**; **Semi-infinite programming: discretization methods**)
outer approximation
[49M20, 49M37, 90C11, 90C26, 90C30]
(*see:* **Cutting plane methods for global optimization**; **Generalized outer approximation**; **Mixed integer nonlinear programming**)
outer approximation *see:* Generalized —; hybrid branch and bound and —; linear —; Logic-based —; quadratic —
outer approximation algorithm
[90C10, 90C11, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**; **MINLP: outer approximation algorithm**)
outer approximation algorithm *see:* MINLP: —
outer approximation with equality relaxation
[65K05, 90C11, 90C26, 90C29, 90C90]
(*see:* **MINLP: global optimization with αBB**; **Multi-objective optimization: interaction of design and control**)
outer approximation with equality relaxation and augmented penalty
[90C11, 90C29, 90C90]
(*see:* **Multi-objective optimization: interaction of design and control**)
outer approximation method
[90C26]
(*see:* **Cutting plane methods for global optimization**)
outer approximation method
[90C09, 90C10, 90C11]
(*see:* **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**)
outer-approximation method *see:* Logic-based —
outer linearization cone
[90C31, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions)
outer problem
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
outgoing arc
[90C35]
(*see:* **Minimum cost flow problem**)
outline of filled function methods *see:* basic —
output-efficient
[90B30, 90B50, 90C05, 91B82]
(*see:* **Data envelopment analysis**)
output/input *see:* maximization of —
output matrices *see:* updating input- —
output neurons
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
output-polynomial
[52B12, 68Q25]
(*see:* **Fourier–Motzkin elimination method**)
output-polynomial time
[52B12, 68Q25]
(*see:* **Fourier–Motzkin elimination method**)
output tables *see:* triangulation problem for input- —
outranking
[90-XX]
(*see:* **Outranking methods**)
Outranking methods
(90-XX)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**;

Multiple objective programming support; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
outranking methods
[90-XX]
(*see:* **Outranking methods**)
outranking relation
[90-XX, 90C29]
(*see:* **Outranking methods**; **Preference disaggregation approach: basic features, examples from financial decision making**)
outranking relation
[90C26, 90C29, 91B28]
(*see:* **Multicriteria sorting methods**; **Portfolio selection and multicriteria analysis**)
outranking relation *see:* fuzzy —
outranking relations
[90C29, 91B06, 91B60]
(*see:* **Financial applications of multicriteria analysis**; **Multicriteria sorting methods**)
outranking relations approach
[90C29]
(*see:* **Decision support systems with multiple criteria**)
outward rounding
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval fixed point theory**)
over *see:* strongly linearly monotonic —
over an ellipsoid *see:* Quadratic programming —
over surface formula *see:* integral —
over a trajectory *see:* optimization —
over unbounded domains *see:* global optimization —
over a volume *see:* integral —
over volume formula *see:* integral —
overall classification error *see:* minimizing the —
overall flowsheet *see:* convergence of the —
overall mean method
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
overdetermined system of nonlinear equations
[90C30]
(*see:* **Nonlinear least squares problems**)
Overdetermined systems of linear equations
(65K05, 65D10)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Orthogonal triangularization**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Linear programming**; **Nonlinear least squares: trust region methods**; **Orthogonal triangularization**; **QR factorization**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
overdetermined Yule–Walker method
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
overhead
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
overhead factor *see:* search —
overlap *see:* contact map —
overlap graph
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
overlap of intervals
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
overlap maximization problem, CMO *see:* Contact map —
overloaded operator
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
overloaded operator
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
overloading *see:* operator —
overprojeclion
[90C30]
(*see:* **Relaxation in projection methods**)
overrelaxation *see:* successive —
Overtaking equilibrium
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
overtaking optimality
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
overtaking optimality *see:* weakly —
overview *see:* Bilevel programming: introduction, history and —; Dynamic programming: infinite horizon problems —

P

P
[90C60]
(*see:* **Complexity classes in optimization**)
P *see:* complexity class —; covers all edge-directions of —; N P=co N —; P = N —
P_*
[90C35]
(*see:* **Multicommodity flow problems**)
P = N P
[49-01, 49K45, 49N10, 90-01, 90C20, 90C27, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**)
P-algorithm
[60J65, 68Q25]
(*see:* **Adaptive global search**)
p-αBB approach *see:* Global optimization: —
p-center problem
[90B80, 90B85]
(*see:* **Warehouse location problem**)

p-center problem
[90B10, 90B80, 90B85, 90C35]
(*see:* **Network location: covering problems**; **Warehouse location problem**)
p-center problem on a network
[90B10, 90B80, 90C35]
(*see:* **Network location: covering problems**)
$\mathcal{P}$ convergence
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
p-CP
[90B80, 90B85]
(*see:* **Warehouse location problem**)
p-form *see:* logarithmic —; rational —
P-function *see:* uniform —
p-matrix
[05B35, 65K05, 90C05, 90C20, 90C25, 90C30, 90C33, 90C55]
(*see:* **Criss-cross pivoting rules**; **Implicit lagrangian**; **Least-index anticycling rules**; **Principal pivoting methods for linear complementarity problems**; **Splitting method for linear complementarity problems**)
P_0-matrix
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
P_*-matrix
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
p-median location-allocation problem
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
p-median problem
[90B80, 90B85]
(*see:* **Warehouse location problem**)
p-median problem
[9008, 90B80, 90B85, 90C26, 90C27, 90C59, 90Cxx, 91Axx, 91Bxx]
(*see:* **Facility location with externalities**; **Variable neighborhood search methods**; **Warehouse location problem**)
p-MP
[90B80, 90B85]
(*see:* **Warehouse location problem**)
p-order active
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
p-order feasible set
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
p-partition
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
p-regular
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
p-regular operator
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
p-simplex
[90C30]
(*see:* **Simplicial decomposition**)
p-simplex
[90C30]
(*see:* **Simplicial decomposition**)
p-VSP
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
P=co N P *see:* N —
PA
[90C26]
(*see:* **Cutting plane methods for global optimization**)
PA of SA
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
package *see:* block truncated Newton software —; computer algebra —; multiple-class software —; single-class software —; variable precision interval —
package of basic software routines
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
package flow problem
[90C35]
(*see:* **Multicommodity flow problems**)
package for specific mathematical areas *see:* software —
Packet annealing
(92B05)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Genetic algorithms**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization in Lennard–Jones and morse clusters**; **Graph coloring**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Bayesian global optimization**; **Genetic algorithms**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization in Lennard–Jones and morse clusters**; Global optimization in protein folding; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Protein folding: generalized-ensemble algorithms**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)

packing
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
packing *see:* vertex —
packing game
[90C27, 90C60, 91A12]
(*see:* **Combinatorial optimization games**)
packing and partitioning problems *see:* Set covering —
packing problem
[90C35]
(*see:* **Feedback set problems**)
packing problem *see:* bandwidth —; bin —; graph —; set —
Padé approximation
[33C45, 65F20, 65F22, 65K10]
(*see:* **Least squares orthogonal polynomials**)
Padé-type approximation
[33C45, 65F20, 65F22, 65K10]
(*see:* **Least squares orthogonal polynomials**)
Padé-type approximation
[33C45, 65F20, 65F22, 65K10]
(*see:* **Least squares orthogonal polynomials**)
pADRE2
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: point and interval taylor operators**)
Painlevé–Kuratowski convergence *see:* discrete —
painting axioms
[90C09, 90C10]
(*see:* **Oriented matroids**)
pair
(*see:* **Contact map overlap maximization problem, CMO**)
pair *see:* admissible trajectory-control —; conjugate —; convex-like function —; dual —; Fenchel duality —; Legendre duality —; primal —; quasimonotone —
pair assignment algorithms
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
pair decomposition *see:* well-separated —
pair decomposition of a monomial ideal *see:* standard —
pair-exchange neighborhood
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
pair of a monomial ideal *see:* admissible —; standard —
pair of trajectory and control functions *see:* asymptotically admissible —
pair of trajectory-function and control-function *see:* admissible —
pair of variables *see:* complementary —
paired comparison
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
paired element *see:* left- —; right- —
paired set *see:* left- —; right- —
pairing *see:* crew —
pairs *see:* fuzzy interval —; left- —; right- —
pairs constrained path problem *see:* impossible —
pairwise comparisons
[90-XX, 90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**; **Outranking methods**)
pairwise comparisons
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
pairwise judgment
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
pairwise protein sequence alignment via mixed-integer linear optimization *see:* Global —
pale edge
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
Palubeckis generator
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
panel *see:* blended —
Pape method *see:* D'Esopo– —
paper converting
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
parabolic curve
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
parabolic curve approach
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
parabolic-exponential function
[90C30]
(*see:* **Image space approach to optimization**)
paradigm *see:* checklist —; disaggregation —; edge insertion —; functional —; general dynamic programming —; imperative programming —; MDO —; Modeling languages in optimization: a new —; optimization —
paradigm of logic programming
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
paradigm semantics for fuzzy logics *see:* Checklist —
paradox *see:* Braess —; Condorcet —
parallax
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
parallel
[68T20, 68T99, 90C27, 90C59]
(*see:* **Contact map overlap maximization problem, CMO**; **Metaheuristics**)
parallel *see:* bulk synchronous —
parallel AD tools
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
parallel algorithm
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
parallel algorithm
[65K05, 65Y05, 68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**; **Parallel computing: models**)
parallel algorithm design *see:* model for —

parallel algorithms
[65G20, 65G30, 65G40, 65K05, 68T20, 68T99, 90C27, 90C30, 90C59]
(*see:* **Interval global optimization**; **Metaheuristics**)
parallel aspiration search
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
Parallel Best-First Tree Search
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
parallel CA algorithm *see:* asynchronous —; synchronized —
parallel computation
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
parallel computation *see:* Automatic differentiation: —
parallel computation in mechanics
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
parallel computation thesis
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
parallel computations
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
parallel computer
[65K05, 65Y05, 90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**; **Parallel computing: models**)
parallel computer *see:* bulk synchronous —; distributed memory —
parallel computing
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
parallel computing
[49-04, 65Y05, 68N20, 90C15]
(*see:* **Automatic differentiation: parallel computation**; **Stochastic programming: parallel factorization of structured matrices**)
parallel computing *see:* massively —; models for —
Parallel computing: complexity classes
(68Q05, 68Q15, 03D15)
(*referred to in:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: parallel computation**; **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Heuristic search**; **Information-based complexity and information-based optimization**; **Kolmogorov complexity**; **Load balancing for parallel optimization techniques**; **Mixed integer nonlinear programming**; **NP-complete problems and proof methodology**; **Parallel computing: models**; **Parallel heuristic search**; **Stochastic network problems: massively parallel solution**)
(*refers to:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: parallel computation**; **Complexity classes in optimization**; **Complexity of degeneracy**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Fractional combinatorial optimization**; **Heuristic search**; **Information-based complexity and information-based optimization**; **Interval analysis: parallel methods for global optimization**; **Kolmogorov complexity**; **Load balancing for parallel optimization techniques**; **Mixed integer nonlinear programming**; **Parallel computing: models**; **Parallel heuristic search**; **Stochastic network problems: massively parallel solution**)
Parallel computing: models
(65K05, 65Y05)
(*referred to in:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: parallel computation**; **Heuristic search**; **Load balancing for parallel optimization techniques**; **Parallel computing: complexity classes**; **Parallel heuristic search**; **Stochastic network problems: massively parallel solution**)
(*refers to:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: parallel computation**; **Heuristic search**; **Interval analysis: parallel methods for global optimization**; **Load balancing for parallel optimization techniques**; **Parallel computing: complexity classes**; **Parallel heuristic search**; **Stochastic network problems: massively parallel solution**)
parallel cuts
[90C05]
(*see:* **Ellipsoid method**)
Parallel Depth-First Tree Search
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
parallel factorization of structured matrices *see:* Stochastic programming: —
parallel graph *see:* series- —
parallel GRASP
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
Parallel heuristic search
(68W10, 68W15, 68R05, 68T20)
(*referred to in:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: parallel computation**; **Heuristic search**; **Load balancing for parallel optimization techniques**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Stochastic network problems: massively parallel solution**)
(*refers to:* **Asynchronous distributed optimization algorithms**; **Automatic differentiation: parallel computation**; **Heuristic search**; **Load balancing for parallel optimization techniques**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Stochastic network problems: massively parallel solution**)
parallel library *see:* NAG —
parallel machines
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
parallel machines *see:* distributed memory —; shared memory —
parallel matrix factorization
[90C15]
(*see:* **Stochastic programming: parallel factorization of structured matrices**)

parallel methods
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
parallel methods for global optimization *see:* Interval analysis: —
parallel minimax tree algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
parallel model *see:* bulk synchronous —
parallel optimization
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
parallel optimization techniques *see:* Load balancing for —
parallel programming
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: differential equations**)
parallel programs *see:* AD of —
parallel random access machine
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
parallel random access machine
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
parallel routing algorithm
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
parallel savings algorithm
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
parallel solution *see:* Stochastic network problems: massively —
parallel tangents
[90C30]
(*see:* **Frank–Wolfe algorithm**)
parallel-tangents algorithm
[90C30]
(*see:* **Conjugate-gradient methods**)
Parallel VNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
parallelism *see:* AD-enabled —; data —; time —
parallelism alignment problem *see:* constant degree —
parallelization
[65G20, 65G30, 65G40]
(*see:* **Interval analysis: systems of nonlinear equations**)
parallelization *see:* automatic —
parallelizing the exploration of minimax trees
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
parameter
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
parameter *see:* Bregman —; convexification —; exact penalty —; globally optimal —; locally optimal —; optimal —; penalty —; prohibition —; temperature —; tuning —
parameter CG family *see:* two- —
parameter computation *see:* conjugate gradient —
parameter derivative
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
parameter estimates
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
parameter estimation
[90C30, 90C52, 90C53, 90C55]
(*see:* **Gauss–Newton method: Least squares, relation to Newton's method**; **Generalized total least squares**)
parameter estimation
[62F10, 65D10, 65K05, 94A17]
(*see:* **Entropy optimization: parameter estimation**; **Overdetermined systems of linear equations**)
parameter estimation *see:* Bayesian —; Entropy optimization: —
parameter estimation problem *see:* sinusoidal —
parameter identification
[34A55, 78A60, 90C30]
(*see:* **Optimal design in nonlinear optics**)
parameter identification
[34A55, 78A60, 90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**; **Optimal design in nonlinear optics**)
parameter identification problem
[49K20, 49M99, 90C05, 90C25, 90C29, 90C30, 90C31, 90C55]
(*see:* **Nondifferentiable optimization: parametric programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
parameter, reject index for interval optimization *see:* Algorithmic improvements using a heuristic —
parameter T *see:* prohibition —
parameter tractability *see:* fixed —
parameter tractable algorithms *see:* fixed —
parameterization *see:* control —
parameterization method
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
parameters
(*see:* **Planning in the process industry**; **Short-term scheduling under uncertainty: sensitivity analysis**)
parameters *see:* estimation of model —; problem —; sensitivity —
parametric
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
parametric approach
(*see:* **Fractional zero-one programming**)
parametric approach to fractional optimization
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
parametric approach to optimality
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
parametric bounds
[90C11, 90C31]
(*see:* **Multiparametric mixed integer linear programming**)

parametric complementarity problems
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
parametric eigenvalue formulation *see:* inverse interpolation —
parametric eigenvalue reformulation
[90C30]
(*see:* **Large scale trust region problems**)
parametric finite optimization problem *see:* one- —
Parametric global optimization: sensitivity
(90C31, 90C34, 90C34)
(*referred to in:* **Bounds and solution vector estimates for parametric NLPS**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization: parametric programming**; **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**)
(*refers to:* **Bounds and solution vector estimates for parametric NLPS**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization: parametric programming**; **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**)
parametric linear complementarity problem
[90C31, 90C33]
(*see:* **Sensitivity analysis of complementarity problems**)
parametric linear programming
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**; **Parametric linear programming: cost simplex algorithm**)
parametric linear programming
[90C05, 90C31]
(*see:* **Parametric linear programming: cost simplex algorithm**)
Parametric linear programming: cost simplex algorithm
(90C05, 90C31)
(*referred to in:* **Bounds and solution vector estimates for parametric NLPS**; **Convex-simplex algorithm**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Global optimization in multiplicative programming**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Multiplicative programming**; **Nondifferentiable optimization: parametric programming**; **Order complementarity**; **Parametric global optimization: sensitivity**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**; **Principal pivoting methods for linear complementarity problems**; **Selfdual parametric method for linear programs**; **Sequential simplex method**; **Topological methods in complementarity theory**)
(*refers to:* **Bounds and solution vector estimates for parametric NLPS**; **Convex-simplex algorithm**; **Global optimization in multiplicative programming**; **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Multiplicative programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**; **Sequential simplex method**)
parametric lower bound
[90C11, 90C31]
(*see:* **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**)
parametric method for linear programs *see:* Selfdual —
parametric methods
[90C26]
(*see:* **Global optimization in multiplicative programming**)
parametric mixed integer linear program *see:* single —
Parametric mixed integer nonlinear optimization
(90C31, 90C11)
(*referred to in:* **Bounds and solution vector estimates for parametric NLPS**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Time-dependent traveling salesman problem**)
(*refers to:* **Bounds and solution vector estimates for parametric NLPS**; **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Integer linear complementary**

problem; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric optimization: embeddings, path following and singularities**; **Selfdual parametric method for linear programs**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)

parametric NLPS *see:* Bounds and solution vector estimates for —

parametric nonlinear complementarity problem
[90C31, 90C33]
(*see:* **Sensitivity analysis of complementarity problems**)

parametric nonlinear optimization
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)

parametric objective simplex algorithm
[90C26]
(*see:* **Global optimization in multiplicative programming**)

parametric optimal control
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)

parametric optimization
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Design of robust model-based controllers via parametric programming**; **Parametric optimization: embeddings, path following and singularities**)

parametric optimization
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)

parametric optimization *see:* nonlinear —

Parametric optimization: embeddings, path following and singularities
(90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34, 65K05, 65K10)
(*referred to in:* **Bounds and solution vector estimates for parametric NLPS**; Generalized semi-infinite programming: optimality conditions; **Globally convergent homotopy methods**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Selfdual parametric method for linear programs**; **Topology of global optimization**)
(*refers to:* **Bounds and solution vector estimates for parametric NLPS**; **Globally convergent homotopy methods**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Selfdual parametric method for linear programs**; **Topology of global optimization**)

parametric optimization problem
[90C31, 90C34]
(*see:* **Semi-infinite programming: second order optimality conditions**)

parametric problem
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)

parametric programming
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**; **Nondifferentiable optimization: parametric programming**)

parametric programming
[90C25, 90C29, 90C30, 90C31, 90C34]
(*see:* **Bilevel programming: optimality conditions and duality**; **Parametric global optimization: sensitivity**)

parametric programming *see:* applications of —; convex —; Design of robust model-based controllers via —; history of —; instability in —; Nondifferentiable optimization: —; optimality in —; stability on —; stable —; structural stability in —; topological stability in —

parametric programming model
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)

parametric programming model
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)

parametric programming models
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)

parametric programs *see:* robust —

parametric representation
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)

parametric representations *see:* necessary optimality condition without using (sub)gradients —

parametric right-hand side simplex algorithm
[90C26]
(*see:* **Global optimization in multiplicative programming**)

parametric rule
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)

parametric search *see:* Megiddo —

parametric semi-infinite optimization *see:* one- —

parametric solutions *see:* comparison of —
parametric upper bound
[90C11, 90C31]
(*see:* **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**)
parametric upper and lower bounds
[90C11, 90C31]
(*see:* **Bounds and solution vector estimates for parametric NLPS**; **Parametric mixed integer nonlinear optimization**)
parametric variational inequalities
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
parametric variational inequality problem
[65K10, 90C31]
(*see:* **Sensitivity analysis of variational inequality problems**)
parametrization *see:* control —; objective function —
parametrized Sard theorem
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
Pardalos generator *see:* Li– —
Pardalos–Rosen mixed integer formulation *see:* LCP: —
parent node reconstruction
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**)
parent of a vertex
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
parergon
[01A20]
(*see:* **Archimedes and the foundations of industrial engineering**)
Paretian cone
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**; **Vector optimization**)
Pareto efficient solution
[90C29]
(*see:* **Vector optimization**)
Pareto optimal
[49L20, 90C29, 90C39]
(*see:* **Dynamic programming: discounted problems**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: pareto optimal solutions, properties**; **Planning in the process industry**)
Pareto optimal
(*see:* **Planning in the process industry**)
Pareto optimal solution
[90B85, 90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**; **Single facility location: multi-objective rectilinear distance location**)
Pareto optimal solution
[90C11, 90C15, 90C29, 90C90]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization: pareto optimal solutions, properties**)
Pareto optimal solution *see:* M- —; weakly —
Pareto optimal solution set
[90C11, 90C29, 90C90]
(*see:* **Multi-objective optimization: interaction of design and control**)
Pareto optimal solutions
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
pareto optimal solutions, properties *see:* Multi-objective optimization: —
Pareto optimality
[90B85, 90C27]
(*see:* **Single facility location: multi-objective euclidean distance location**; **Time-dependent traveling salesman problem**)
Pareto optimality
[90C29]
(*see:* **Vector optimization**)
Pareto optimality of MODP *see:* principle of —
Pareto optimum
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
Pareto point
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
Pareto race
[90C29]
(*see:* **Multiple objective programming support**)
Pareto solution
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
Parlett factorization *see:* Bunch and —
Parrott theorem
[93D09]
(*see:* **Robust control**)
parsing *see:* expression —
part *see:* excess —; ideal —; nonideal —; seed —
part of a function *see:* twice-differentiable —
part map *see:* standard —
PARTAN
[90C30]
(*see:* **Frank–Wolfe algorithm**)
PARTAN algorithm
[90C30]
(*see:* **Conjugate-gradient methods**)
partial assignment
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
partial calmness condition
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
partial completely positive matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
partial computability
[90C26]
(*see:* **Global optimization using space filling**)
partial computability
[90C26]
(*see:* **Global optimization using space filling**)

partial computation of a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
partial computation of a Turing machine *see:* length of a —
partial contraction matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
partial definite matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
partial derivatives *see:* elementary —; matrix of second —
partial differential equations
[03H10, 49J27, 49K20, 49M99, 90C34, 90C55]
(*see:* **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
partial differential equations *see:* First order —
partial discretization
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
partial distance matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
partial equilibrium
[91B50]
(*see:* **Walrasian price equilibrium**)
partial equilibrium
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
partial equilibrium model
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
partial Hermitian matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
partial information
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
partial linearization
[90C30]
(*see:* **Cost approximation algorithms**)
partial matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
partial matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
partial matrix *see:* completion of a —
partial monotonicity
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
partial monotonicity
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
partial order
[41A30, 47A99, 62G07, 62G30, 65K05, 65K10, 90C29]
(*see:* **Isotonic regression problems**; **Lipschitzian operators in best approximation by bounded or continuous functions**; **Vector optimization**)
partial order *see:* antisymmetric —; Löwner —
partial order relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
partial orders
[90C29]
(*see:* **Preference modeling**)
partial proximal point algorithm
[90C30]
(*see:* **Cost approximation algorithms**)
partial semidefinite matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
partial sequential normal compactness
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
partial-update Newton method
[90C30]
(*see:* **Numerical methods for unary optimization**)
partial-update Newton method
[90C30]
(*see:* **Numerical methods for unary optimization**)
partial updating
[90C05]
(*see:* **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**)
partially
(*see:* **Mixed integer programming/constraint programming hybrid methods**)
partially asynchronous computation
[90C30]
(*see:* **Cost approximation algorithms**)
partially asynchronous iterative method
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
partially asynchronous operation
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
partially monotonous
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
partially separable function
[90C06]
(*see:* **Large scale unconstrained optimization**)
participant
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
participants *see:* subset of —
partition
[68Q25, 90C09, 90C10, 90C15, 90C60]
(*see:* **Matroids**; **NP-complete problems and proof methodology**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
partition *see:* 2- —; 3- —; block of a —; left-collection of a —; ordered —; p- —; rectangular —; right-collection of a —; simplicial —

partition flipping
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
partition-flipping *see:* algorithm —
partition hierarchy
[62H30, 90C27, 90C39]
(*see:* **Assignment methods in clustering**; **Dynamic programming in clustering**)
partition identities *see:* primitive —
partition matching
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
partition matching *see:* Maximum —
partition-matching-I *see:* algorithm —
partition matching problem *see:* maximum —
partition matroid
[90C09, 90C10]
(*see:* **Matroids**)
partition number *see:* clique —
partition problem *see:* rectangular —
Partition Problem (MP) *see:* minimum —
partition on a set
[03B52, 03E72, 41A30, 47S40, 62J02, 68T27, 68T35, 68Uxx, 90Bxx, 90C26, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Regression by special functions: algorithms and complexity**)
partitioned quasi-Newton method
[90C06]
(*see:* **Large scale unconstrained optimization**)
partitioning
[65K05]
(*see:* **Direct global optimization algorithm**)
partitioning *see:* adaptive —; Graph —; minimum clique —; order constrained —; ordered —; set —
partitioning algorithm *see:* K-iterated tour —; nearest insertion optimal —
partitioning method
[90C35]
(*see:* **Multicommodity flow problems**)
partitioning problem *see:* graph —; k-way graph —; set —
partitioning problems *see:* Set covering, packing and —
partitions
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
partitions *see:* nested —; set L_{free} of unused —; set L_{reac} of used —; set R_{free} of unused —; set R_{reac} of used —
partitions optimization *see:* Nested —
partly convex problems
[90C25, 90C26]
(*see:* **Decomposition in global optimization**)
partly convex program
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
partly convex programs
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
pass theorem *see:* mountain —
passenger trip
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
patch (COP) *see:* contract-or- —
patching
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
path
[90C35]
(*see:* **Minimum cost flow problem**)
path *see:* center —; central —; circular —; co-optimal —; critical —; directed —; dogleg —; edge-disjoint —; elementary connecting —; forward —; linear —; multiple dogleg —; node-disjoint —; nondominated —; optimal —; principal variation —; shortest —; stochastic shortest —; unique —
path algorithm *see:* augmenting —; generic augmenting —; successive shortest —
path algorithms *see:* generic shortest —
path approach *see:* feasible —; infeasible —
path coloring problem
[05C85]
(*see:* **Directed tree networks**)
path contraction
[90B06, 90B10, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Shortest path tree algorithms**; **Traveling salesman problem**)
path cost
[90C35]
(*see:* **Multi-index transportation problems**)
path cost *see:* Hamiltonian —
path decomposition
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)
path extension
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
path flow formulation
[90B06, 90B15, 90B20, 91B50]
(*see:* **Dynamic traffic networks**; **Traffic network equilibrium**)
path flow pattern *see:* feasible —
path flows *see:* variational inequality formulation in —
path following
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
path following
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
path following algorithm
[90C05]
(*see:* **Linear programming: interior point methods**)
path following algorithm for entropy optimization
[90C25, 90C51, 94A17]
(*see:* **Entropy optimization: interior point methods**)

path following approach
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
path following methods
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
path following and singularities *see:* Parametric optimization: embeddings —
path formulation
[90C35]
(*see:* **Multicommodity flow problems**)
path formulation of the multicommodity flow problem *see:* node- —
path in a graph
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
path in a graph *see:* length of a —
path length
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
path length *see:* maximin —; maximum —; minimax —; minimum —
path optimization *see:* sample- —
path problem *see:* deterministic shortest —; impossible pairs constrained —; shortest —; stochastic shortest —
path problems *see:* Dynamic programming: stochastic shortest —; stochastic shortest —
path procedure *see:* next shortest —
path-protection
[46N10, 68M10, 90B18, 90B25]
(*see:* **Integer linear programs for routing and protection problems in optical networks**)
path relinking
[65H20, 65K05, 68T20, 68T99, 90-01, 90B40, 90C10, 90C11, 90C20, 90C27, 90C35, 90C59, 94C15]
(*see:* **Greedy randomized adaptive search procedures**; **Linear ordering problem**; **Metaheuristics**)
path routing pattern model *see:* single —
path-string
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
path tree algorithms *see:* Shortest —
path tree problem *see:* single source shortest —
path tree problems *see:* shortest —
paths *see:* Pivoting algorithms for linear programming generating two —; problem of finding shortest —
pathwidth
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)
pattern *see:* cutting —; feasible path flow —; location —; positive-negative-zero —; strategy —; zero-nonzero —
pattern based model *see:* gIS design —
pattern classification *see:* statistical —
pattern of a matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
pattern of a matrix *see:* sign —
pattern model *see:* fractional routing —; single path routing —
pattern recognition
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
pattern recognition *see:* Complementarity algorithms in —; statistical —
pattern search
[90C30]
(*see:* **Rosenbrock method**)
pattern search
[90C30]
(*see:* **Cyclic coordinate method**; **Powell method**; **Rosenbrock method**)
pattern searches
[90C30]
(*see:* **Cyclic coordinate method**)
patterns *see:* cutting —; word —
patterns and graphs *see:* matrix —
PAV
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
payments *see:* game with side —
payoff space *see:* resource- —
PC clusters
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
PCP-LCP
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
PCR groups
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**)
PCSP *see:* computer code —
pd
[05C50, 15A48, 15A57, 90C25, 90C26, 90C39]
(*see:* **Matrix completion problems**; **Second order optimality conditions for nonlinear optimization**)
pD-VNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
PDS
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
Peano curve
[90C26]
(*see:* **Global optimization using space filling**)
Peano curve
[90C26]
(*see:* **Global optimization using space filling**)
Peano function
[90C30]
(*see:* **Image space approach to optimization**)
Peano map
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
Pearson chi-square statistic
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)

penalties
[90C26]
(*see:* **Global optimization using space filling**)
penalty
[90C11, 90C26]
(*see:* **Global optimization in batch design under uncertainty**; **MINLP: branch and bound methods**)
penalty *see:* down —; Driebeck–Tomlin —; exact —; outer approximation with equality relaxation and augmented —; up —
penalty approach
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
penalty-based method
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
penalty function
[05C69, 05C85, 65K05, 68W01, 90C59, 90C90, 91B28, 93-XX]
(*see:* **Direct global optimization algorithm**; **Dynamic programming: optimal control applications**; **Heuristics for maximum clique and independent set**; **Robust optimization**)
penalty function *see:* Courant —; exact —; exact L_∞- —; ℓ_1 —; l_1 exact —; logarithmic-quadratic barrier- —
penalty function approach *see:* continuously differentiable exact —
penalty function based algorithm *see:* exact —
penalty functions
[93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**)
penalty method *see:* Exact —
penalty methods *see:* Quasidifferentiable optimization: exact —; regularity condition for —
penalty parameter
[90Cxx]
(*see:* **Discontinuous optimization**)
penalty parameter *see:* exact —
penalty technique
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
Penrose conditions
[65Fxx]
(*see:* **Least squares problems**)
Penrose pseudo-inverse *see:* Moore– —
PEP
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
Peptide identification via mixed-integer optimization
per stage *see:* average cost —; discounted problem with bounded cost —
per stage problem *see:* average cost —
per stage problems *see:* average cost —; Dynamic programming: average cost —
perceptron algorithm
[62H30, 68T10, 90C05]
(*see:* **Linear programming models for classification**)
perfect
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
perfect *see:* subgame —
perfect b-matching problem
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
perfect competition
[91B50]
(*see:* **Financial equilibrium**; **Walrasian price equilibrium**)
perfect competition
[91B50]
(*see:* **Financial equilibrium**; **Walrasian price equilibrium**)
perfect dual *see:* formal —
perfect duality
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
perfect duality *see:* Semi-infinite programming, semidefinite programming and —
perfect duality from the view of linear semi-infinite programming
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
perfect graph theorem *see:* strong —
perfect information *see:* expected value of —
perfect-information game *see:* two-player zero-sum —
perfect matching
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
perfect matching problem
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
perfect matchings
[05C85]
(*see:* **Directed tree networks**)
perfect offspring
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
perfectly competitive
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
perfectly competitive equilibrium model
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
perfectly consistent case
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
performance *see:* computational —; optimization of computational —
performance computing *see:* high —
performance computing system *see:* high —
performance evaluation
[90B30, 90B50, 90C05, 91B82]
(*see:* **Data envelopment analysis**)
performance Fortran *see:* high —
performance guarantee
[05C85, 90C35]
(*see:* **Directed tree networks**; **Graph coloring**)
performance guarantee *see:* worst-case —
performance index
[93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**; **Dynamic programming: optimal control applications**)

performance index *see:* augmented —
performance measurement *see:* Supply chain —
Performance profiles of conjugate-gradient algorithms for unconstrained optimization
(49M07, 49M10, 90C06, 65K05)
performances *see:* Volume computation for polytopes: strategies and —
perimeter
(*see:* **State of the art in modeling agricultural systems**)
period *see:* duty- —; orbital —
period model *see:* single- —
period routing
[90B06]
(*see:* **Vehicle routing**)
periodic review model
[90B50]
(*see:* **Inventory management in supply chains**)
periodic review model
[90B50]
(*see:* **Inventory management in supply chains**)
permanent of a matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
permanent residents
(*see:* **Emergency evacuation, optimization modeling**)
permutation *see:* neighborhood of a —
permutation graph
[90C35]
(*see:* **Feedback set problems**)
permutation matrix
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
permutation QAP *see:* constant —
permutations
[05C85]
(*see:* **Directed tree networks**)
Perron–Frobenius theorem
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
person game *see:* cooperative case of a two- —; two- —
person zero-sum game *see:* two- —
perspective *see:* descriptive —; metric-based —; model-based —; normative —; prescriptive —
perturbation *see:* data —; lexicographic ordering and —
perturbation analysis
[90C15]
(*see:* **Derivatives of probability measures**)
perturbation analysis *see:* infinitesimal —
perturbation function
[90C30]
(*see:* **Image space approach to optimization**)
perturbation methods
[90C30]
(*see:* **Cost approximation algorithms**)
perturbation model *see:* right-hand side —
perturbation problem *see:* right-hand side —
perturbation technique
[05B35, 65K05, 90C05, 90C20, 90C33, 90C60]
(*see:* **Complexity of degeneracy**; **Lexicographic pivoting rules**)
perturbations
[90C05, 90C25, 90C30, 90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**; **Semi-infinite programming, semidefinite programming and perfect duality**)
perturbations *see:* minimax observation problem under uncertainty with —; piecewise-constant —; Vasicek model with impulse —
perturbative approximation
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
perturbed least squares problem
[65Fxx]
(*see:* **Least squares problems**)
perturbed system
[90C60]
(*see:* **Complexity of degeneracy**)
petrochemical industry
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
Petrov–Galerkin approach
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
Petrov–Galerkin iteration
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
phase *see:* allocation —; construction —; global —; intensification —; local —; minimum —; nonminimum —; two- —
phase algorithm *see:* three —
phase and chemical reaction equilibrium *see:* Global optimization in —
phase classes
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
phase compositions *see:* equality of —
phase constraints
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
phase constraints *see:* Lagrange multipliers for —
phase equilibrium
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
phase equilibrium
[65H20, 80A10, 80A22, 90C26, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**; **Global optimization in phase and chemical reaction equilibrium**)
phase equilibrium equations
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in distillation systems**)
phase equilibrium equations *see:* ideal and nonideal —
phase equilibrium problem
[49K99, 65K05, 80A10]

(*see:* **Optimality criteria for multiphase chemical equilibrium**)
phase equilibrium problems *see:* Global optimization: application to —
phase in GRASP *see:* construction —; local search —
phase method *see:* two- —
phase methods *see:* Stochastic global optimization: two- —
phase problem
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)
phase problem
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)
Phase problem in X-ray crystallography: Shake and bake approach
(90C26)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms**; **Global optimization in Lennard–Jones and morse clusters**; **Graph coloring**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Simulated annealing methods in protein folding**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms**; **Global optimization in Lennard–Jones and morse clusters**; Global optimization in protein folding; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Protein folding: generalized-ensemble algorithms**; **Simulated annealing**; **Simulated annealing methods in protein folding**)
phase procedure *see:* two- —
phase retrieval based on single-crystal X-ray diffraction data *see:* Optimization techniques for —
phase space
[49J15, 49K15, 93C10]
(*see:* **Pontryagin maximum principle**)
phase split
[65H20, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**)
phase stability
[65H20, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**)
phase stability
[65H20, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**)
phase stability problem
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
phases
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
phases *see:* backtrack —; forward —; liquid —; vapor —
Φ-isotone mapping
[90C33]
(*see:* **Order complementarity**)
physical junction nodes
[90C30, 90C35]
(*see:* **Optimization in water resources**)
physically linear problem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
physically nonlinear problem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
Pi-algebra
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
Pi-algebras *see:* application of —; complexity theory of —; families of —; functional completeness of —; functionally complete normal forms of —; use of —
PI-algebras and 2-valued normal forms
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
PI-algebras of many-valued logics *see:* taxonomy of the —
π-classes
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
PI-logic algebras
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
PI-logic algebras
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
Pi-logic algebras *see:* taxonomy of —
PI-normal form
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
PI-systems *see:* subfamilies of n-valued —
Piaget group of transformation
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
pick-up and delivers
[90B06]
(*see:* **Vehicle routing**)
pickups and deliveries *see:* Vehicle routing problem with simultaneous —
pictogram translation mapping technique
(*see:* **State of the art in modeling agricultural systems**)
pieces
[46N10, 47N10, 49M37, 65K10, 90C26, 90C30]
(*see:* Global optimization: tight convex underestimators)
piecewise
[46N10, 47N10, 49M37, 65K10, 90C26, 90C30]
(*see:* Global optimization: tight convex underestimators)
piecewise constant control
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
piecewise-constant perturbations
[90C34, 91B28]

(*see:* **Semi-infinite programming and applications in finance**)
piecewise continuously differentiable function
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
piecewise continuously differentiable function
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
piecewise differentiable function
[90Cxx]
(*see:* **Discontinuous optimization**)
piecewise linear arc cost
[90B10]
(*see:* **Piecewise linear network flow problems**)
piecewise linear control
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
piecewise linear function
[65M60, 90C26]
(*see:* **MINLP: application in facility location-allocation**; **Variational inequalities: F. E. approach**)
piecewise linear function *see:* decomposition of a continuous —
piecewise linear minimum cost network flow problem
[90B10]
(*see:* **Piecewise linear network flow problems**)
Piecewise linear network flow problems
(90B10)
(*referred to in:* **Auction algorithms**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Generalized networks**; **Global supply chain models**; **Inventory management in supply chains**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Nonoriented multicommodity flow problems**; **Operations research models for supply chain management and design**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
(*refers to:* **Auction algorithms**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Generalized networks**; **Global supply chain models**; **Inventory management in supply chains**; **Maximum flow problem**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Nonoriented multicommodity flow problems**; **Operations research models for supply chain management and design**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
Piecewise linear programming
[90Cxx]
(*see:* **Discontinuous optimization**)
piecewise linear quadratic function
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
piecewise linear upper bound
[90C15]
(*see:* **Stochastic programs with recourse: upper bounds**)
piecewise linearization *see:* improved —
piecewise linearization in facility location problems with staircase costs *see:* convex —
piecewise sequential quadratic programming method
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
piecewise sequential quadratic programming method
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
piecewise SQP approach *see:* Optimization with equilibrium constraints: A —
piecewise twice-differentiable function
[90Cxx]
(*see:* **Discontinuous optimization**)
Piela method
[90C26, 90C90]
(*see:* **Global optimization in Lennard–Jones and morse clusters**)
pilot method
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
pinch point
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
Pinkava algebra
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
Pinkava logic algebras *see:* many-valued families of the —
Pinkava logical algebra
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
Pinkava normal forms *see:* minimization of —
pinned
[51K05, 52C25, 68Q25, 68U05, 90C22, 90C35]
(*see:* Graph realization via semidefinite programming)
pitchfork
(*see:* Global terrain methods)
pivot
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
pivot *see:* admissible —; block —; diagonal —; double —; exchange —; simple principal —
pivot algorithm
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Lexicographic pivoting rules**)
pivot algorithm *see:* principal —
pivot method
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
pivot methods *see:* complementary —
pivot operation
[90C05, 90C33, 90C35]

(*see:* **Minimum cost flow problem**; **Pivoting algorithms for linear programming generating two paths**)
pivot operation *see:* degenerate —; nondegenerate —
pivot rules
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
pivot rules
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Principal pivoting methods for linear complementarity problems**)
pivot selection *see:* lexicographic —
pivot steps *see:* average number of —; expected number of —
pivot theory *see:* complementary —
pivotal transform *see:* principal —
pivotal transformation *see:* principal —
pivoting
[90C05, 90C10]
(*see:* **Linear programming**; **Simplicial pivoting algorithms for integer programming**)
pivoting
[90C05, 90C33]
(*see:* **Lemke method**; **Linear programming**)
pivoting *see:* guaranteed to be stable without —; matrix class invariant under principal —; principal —; QR factorization with column- —
pivoting algorithm
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
pivoting algorithms *see:* varying dimension —
pivoting algorithms for integer programming *see:* Simplicial —
Pivoting algorithms for linear programming generating two paths
(90C05, 90C33)
(*referred to in:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Linear programming**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**; **Simplicial pivoting algorithms for integer programming**)
(*refers to:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Linear programming**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**; **Simplicial pivoting algorithms for integer programming**)
pivoting method *see:* least-index —; principal —
pivoting methods for linear complementarity problems *see:* Principal —
pivoting property
[90C09, 90C10]
(*see:* **Oriented matroids**)
pivoting required *see:* no —
pivoting rule *see:* Bland least index —; Dantzig largest coefficient —; generic —; lexicographic —
pivoting rules
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**; **Linear programming: Klee–Minty examples**)
Pivoting rules
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
pivoting rules *see:* Criss-cross —; Lexicographic —
pixels
[90C90]
(*see:* **Optimization in medical imaging**)
Piyavskii–Shubert algorithm
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
placement *see:* inventory —
plan *see:* block —
planar
[90C10, 90C27]
(*see:* **Multidimensional assignment problem**)
planar augmentation
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
planar graph
[68R10, 90C10, 90C27, 94C15]
(*see:* **Branchwidth and branch decompositions**; **Graph planarization**)
planar graph
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
planar graph *see:* level —; maximum weighted —
planar multi-index transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)
planar multilayered dielectric structures *see:* Global optimization of —
planar subgraph
[90B80]
(*see:* **Facilities layout problems**)
planar subgraph *see:* maximal —; maximum —
planarity testing
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
planarity-testing algorithm *see:* Hopcroft–Tarjan —
planarization
[90C10, 90C27, 90C35, 94C15]
(*see:* **Graph planarization**; **Optimization in leveled graphs**)
planarization *see:* branch and bound algorithm for weighted graph —; Graph —; level —
planarization problem *see:* k-level —; level —
plane *see:* Chvátal–Gomory cutting —; extended cutting —; generalized cutting —; Minkowski —; strong cutting —; trade-off cutting —
plane algorithm *see:* cutting —; Extended cutting —; Gomory cutting —; Sequential cutting —
plane algorithms *see:* Integer programming: cutting —
plane algorithms for stochastic linear programming problems *see:* Stabilization of cutting —
plane approach *see:* cutting —; Mixed-integer nonlinear optimization: A disjunctive cutting —
plane approaches *see:* cutting —
plane coefficients *see:* statistical representation of cutting —
plane criterion *see:* reaction tangent- —; tangent- —
plane method *see:* analytic center cutting —; cutting —; extended cutting —; generalized cutting —; Kelley's classical cutting —; Kelley cutting —

plane methods *see:* cutting —; Nondifferentiable optimization: cutting —; regularization of deterministic cutting —
plane methods for global optimization *see:* Cutting —
plane model *see:* cutting —
planes *see:* cutting —; polyhedral cutting —; Stochastic linear programming: decomposition and cutting —
planning
[90C26, 90C30, 90C31]
(*see:* **Bilevel programming: introduction, history and overview**)
planning *see:* Chemical process —; distribution systems —; extra-urban transit —; financial —; long range —; multiperiod —; operation —; production —; requirements —; Resource —; scheduling (staff —; street —; water resource —
planning horizon
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
planning of offshore oilfield infrastructure *see:* Optimal —
planning problem *see:* process —
planning problems *see:* Global optimization algorithms for financial —
Planning in the process industry
planning and scheduling *see:* facility —; Integrated —
planning under uncertainty *see:* Production —
planning under uncertainty on hydrological exogenous inflow and demand *see:* water resources —
plant
(*see:* **State of the art in modeling agricultural systems**)
plant *see:* co-generation —; multiproduct —; multiproduct batch —; multipurpose —; run-of-river —; storage —; thermal —
plant design *see:* batch —
Plant layout problems and optimization
plant location model
[90-02]
(*see:* **Operations research models for supply chain management and design**)
plant location problem *see:* simple —; uncapacitated —
plant/model nodes
[90C35]
(*see:* **Minimum cost flow problem**)
plant nodes
[90C35]
(*see:* **Minimum cost flow problem**)
plant railroads *see:* engine routing and industrial in- —
plants *see:* hydro —; run-of-river —; storage —; tracing the states of —
plasticity *see:* computational —
plates
[90C26, 90C90]
(*see:* **Structural optimization: history**)
platform cost *see:* linear —
plausible rules
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
play *see:* mood of —
player
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
player zero-sum perfect-information game *see:* two- —
pLCP
[65K10, 90C33, 90C51]
(*see:* **Generalizations of interior point methods for the linear complementarity problem**)
PLE
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
PLE
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
PLNFP
[90B10]
(*see:* **Piecewise linear network flow problems**)
PLS-complexity
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
PLS problems
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
plus homogeneous
[41A50, 41A65, 46B40, 90C46]
(*see:* **Best approximation in ordered normed linear spaces**)
PM
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)
PMD
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
Pnueli algorithm
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
Poincaré polynomial
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
Poincaré polynomial
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
point *see:* ∂^+-critical —; ∞-stationary —; asymptotically stable stationary —; bounds on the distance of a feasible point to a solution —; (C_m^J)-efficient —; Cauchy —; contact —; coupled fixed —; critical —; dead —; decomposition —; Dini sup-stationary —; distinguished —; dominated —; efficient —; ε-stationary —; equilibrium —; extreme —; feasible —; fixed —; generalized critical —; global maximum —; global minimum —; global minimum KKT —; grid —; Hadamard ∞-stationary —; Hadamard sup-stationary —; index of a constraint violating —; inf-stationary —; infeasible interior —; inner —; interior —; invexity at a —; isolated stationary —; Karush–Kuhn–Tucker —; kKT —; KT —; Kuhn–Tucker —; left saddle —; local efficient —; local Ekeland —; local maximum —; local minimum —; local strictly efficient —; local weakly efficient —; lower boundary —; motion of a —; nondegenerate —; nondegenerate critical —; Pareto —; pinch —; proximal —; quadratic turning —; query —; reference —; regular —; regular feasible —; regular stationary —; right saddle —; saddle —; saddle-minimax —; stationary —; Steiner —; strict local maximum —; strict local minimum —; strictly efficient —; strong stability of

a stationary —; subcritical —; substationary —; sup-stationary —; supercritical —; supermaximum —; superminimax —; trap-door —; trial —; turning —; upper boundary —; Voronoi —; weakly efficient —
point acceleration function *see:* the mid- —
point AD
[65H99, 65K99]
(*see:* **Automatic differentiation: point and interval**)
point algorithm *see:* dual exterior —; entropic proximal —; infeasible-start interior- —; interior —; partial proximal —; proximal —; quadratic proximal —
point algorithms *see:* inexact proximal —; interior —
point algorithms for entropy optimization *see:* interior —
point approach *see:* proximal —
point-based approximation
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
point-based logic system of approximate reasoning
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
point boundary value problem *see:* ODE two- —; two- —
point bounds for NLP *see:* solution- —
point bundle method *see:* proximal —
point computation *see:* fixed —
point conditions
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
point configuration
[90C09, 90C10]
(*see:* **Oriented matroids**)
point design
[90C26, 90C29]
(*see:* **Optimal design of composite structures**)
point of an energy functional *see:* generalized critical —
point enumeration *see:* extreme —
point formulation *see:* saddle- —
point of a functional *see:* substationarity —
point inequalities *see:* saddle- —
point and interval *see:* Automatic differentiation: —
point and interval taylor operators *see:* Automatic differentiation: —
point intervals *see:* floating —
point iterate *see:* dead- —
point iteration *see:* fixed —
point logarithmic barrier method *see:* interior —
point mapping *see:* nearest —
point mathematical program *see:* extreme —
point method *see:* exterior —; interior —; proximal —
point methods *see:* Entropy optimization: interior —; interior —; Linear programming: interior —; polynomial time interior —; primal-dual interior- —; proximal —; Successive quadratic programming: solution by active sets and interior —
point methods for distributed optimal control problems *see:* Sequential quadratic programming: interior —
point methods for the linear complementarity problem *see:* Generalizations of interior —
point methods for semidefinite programming *see:* Interior —
point operation *see:* floating —
point problem *see:* Equivalence between nonlinear complementarity problem and fixed —; fixed —; high —; saddle- —
point ranking *see:* extreme —
point with respect to a set *see:* substationarity —
point set *see:* efficient —; generalized critical —
point solution *see:* extreme —
point to a solution point *see:* bounds on the distance of a feasible —
point solutions *see:* enumerating extreme —
point sufficient condition *see:* saddle- —
Point Taylor Operator
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
point theorem *see:* brouwer fixed —; fixed —; Miranda fixed —; right saddle- —; Schauder fixed —; supercritical —; Tychonoff fixed —
point theory *see:* critical —; fixed —; Interval fixed —
point theory and optimality conditions *see:* Saddle —
point-to-set mapping
[65K10, 90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**; **Rosen's method, global convergence, and Powell's conjecture**; **Sensitivity analysis of variational inequality problems**)
point-to-set mapping
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
point-to-set mapping *see:* closed —
point-to-set mappings
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
point Voronoi diagram *see:* farthest- —
pointed closed convex cone
[90C33]
(*see:* **Topological methods in complementarity theory**)
pointed convex cone
[90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Order complementarity**)
points *see:* abnormal —; convex combination of the extreme —; critical —; decomposition —; Discretely distributed stochastic programs: descent directions and efficient —; High-order necessary conditions for optimality for abnormal —; inf-stationary —; KKT —; multiple Kuhn–Tucker —; multiple QP Kuhn–Tucker —; nondegenerate critical —; ranking extreme —; saddle —; set of ε-global —; set of ε-most active —; set of feasible —; stationary —; Steiner —; Steiner tree problem with minimum number of Steiner —; successive improvement of KKT —
points problem *see:* fekete —
points on the same face
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements in optimization**)
points sets *see:* connectedness of the efficient —

Poisson distribution
[90C15]
(*see:* **Logconcavity of discrete distributions**)
Poisson equation
[60J05, 90C15]
(*see:* **Derivatives of markov processes and their simulation**)
Polak–Ribiére algorithm *see:* Polyak– —
Polak–Ribiére formula
[90C06]
(*see:* **Large scale unconstrained optimization**)
Polak–Ribiére method
[90C06]
(*see:* **Large scale unconstrained optimization**)
polar
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
polar cone
[41A10, 46N10, 47N10, 49K27, 90C30]
(*see:* **Duality for semidefinite programming**; **High-order necessary conditions for optimality for abnormal points**)
polar duality
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
polar matrix
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**)
polar polyhedron
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
policies *see:* in-company —; nonanticipative water resources —; nonanticipativity water resources —; optimal —; (s,S) optimal —
policy
[49Jxx, 49L20, 90B50, 91Axx]
(*see:* **Dynamic programming: inventory control**; **Infinite horizon control and dynamic games**; **Inventory management in supply chains**)
policy *see:* admissible —; barrier —; Continuous review inventory models: (*QR*) —; one-for-one ordering —; optimal control —; proper —; (Q,R) —; (s,S) —; scheduling —; stationary —; unichain —
policy evaluation
[49L20, 90C39, 90C40]
(*see:* **Dynamic programming: infinite horizon problems, overview**)
policy iteration
[49L20, 49L99, 90C39, 90C40]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: stochastic shortest path problems**)
Poliquin reduction *see:* Burke– —
political districting problem
[90C10, 90C11, 90C27, 90C57]
(*see:* **Set covering, packing and partitioning problems**)
Pollak conjecture *see:* Gilbert– —
polling scheme *see:* random —
pollution *see:* air —
pollution control
[90C15]
(*see:* **Stochastic quasigradient methods: applications**)
Poly(L-Alanine)
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
Poly(L-Alanine)
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
Pólya theorem *see:* Hardy–Littlewood– —
polyak II rule
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**)
Polyak method *see:* Levitin– —
Polyak minimizing sequence *see:* Levitin– —
Polyak–Polak–Ribiére algorithm
[90C30]
(*see:* **Conjugate-gradient methods**)
Polyak well-posed problem *see:* Levitin– —
polyblock
[90C26]
(*see:* **Cutting plane methods for global optimization**)
polyblock *see:* reduced —; reverse —
polyblock algorithm
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
polyblock algorithm *see:* discrete —; reverse —; revised —
polyblock approximation
[90C26]
(*see:* **Cutting plane methods for global optimization**)
polyblock approximation
[90C26]
(*see:* **Cutting plane methods for global optimization**)
polyblock approximation method
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
polyblock (copolyblock) algorithm *see:* revised reverse —
polyblocks
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
polygon
(*see:* **State of the art in modeling agricultural systems**)
polygonal Arrangement
(*see:* **State of the art in modeling agricultural systems**)
polygonal Component
(*see:* **State of the art in modeling agricultural systems**)
Polygons Arrangement *see:* two —
polyhedra *see:* segments of —
polyhedral annexation
[90C25, 90C26]
(*see:* **Concave programming**; **Cutting plane methods for global optimization**)
polyhedral annexation
[90C09, 90C10, 90C11, 90C26]
(*see:* **Cutting plane methods for global optimization**; **Disjunctive programming**)

polyhedral combinatorics
[90C05, 90C06, 90C08, 90C10, 90C11, 90C27, 90C35, 90C57]
(*see:* **Integer programming**; **Integer programming: cutting plane algorithms**; **Optimization in leveled graphs**)
polyhedral cutting planes
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
polyhedral methods
[05-XX]
(*see:* **Frequency assignment problem**)
polyhedral methods
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**; **Set covering, packing and partitioning problems**)
polyhedral set
[90C30]
(*see:* **Simplicial decomposition**)
polyhedral set
[90C30]
(*see:* **Simplicial decomposition**)
polyhedral set *see:* convex —
polyhedral subdivision
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
polyhedral subdivision *see:* cell of a —; face of a —
polyhedral theory
[90C10, 90C11, 90C27, 90C57]
(*see:* **Set covering, packing and partitioning problems**)
polyhedron
[90C30]
(*see:* **Convex-simplex algorithm**; **Simplicial decomposition**)
polyhedron
[90C30]
(*see:* **Convex-simplex algorithm**; **Frank–Wolfe algorithm**; **Simplicial decomposition**)
polyhedron *see:* essential —; polar —; regular —; segment of a —; simple —; state —; submodular —
polylogarithmic time
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
polymatrix game
[90C25, 90C33]
(*see:* **Integer linear complementary problem**)
polynomial
[05C15, 05C62, 05C69, 05C85, 34E05, 90C27, 90C59]
(*see:* **Asymptotic properties of random multidimensional assignment problem**; **Optimization problems in unit-disk graphs**)
polynomial *see:* biorthogonal —; characteristic —; characterstic —; Chebyshev —; generating —; initial term of a —; normal form of a —; output- —; Poincaré —
polynomial algorithm
[68Q25, 90C60]
(*see:* **Computational complexity theory**; **NP-complete problems and proof methodology**)
polynomial algorithm
[90C60]
(*see:* **Computational complexity theory**)
polynomial algorithm *see:* nondeterministic —; strongly —
polynomial of best approximation
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
polynomial complexity
[03B50, 41A30, 62J02, 68T15, 68T30, 90C26]
(*see:* **Finite complete systems of many-valued logic algebras**; **Regression by special functions: algorithms and complexity**)
polynomial of degree c *see:* algorithm —
polynomial equations
[12D10, 12Y05, 13P10]
(*see:* **Gröbner bases for polynomial equations**)
polynomial equations *see:* Gröbner bases for —
polynomial ideal
[12D10, 12Y05, 13P10]
(*see:* **Gröbner bases for polynomial equations**)
polynomial matrix
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
polynomial programming
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
polynomial programs
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
polynomial reducibility
[90C60]
(*see:* **Complexity classes in optimization**)
polynomial solution *see:* strongly —
polynomial solvability
[90C35]
(*see:* **Feedback set problems**)
polynomial system of equations
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
polynomial time
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68Q25, 68R, 68U, 68W, 90B, 90C, 90C35, 90C60, 91B28]
(*see:* **Competitive ratio for portfolio management**; **Complexity theory**; **Computational complexity theory**; **Convex discrete optimization**; **Graph coloring**)
polynomial time
[90C60]
(*see:* **Complexity theory**; **Complexity theory: quadratic programming**)
polynomial time *see:* output- —; strongly —; super- —
polynomial time algorithm
[49-01, 49K45, 49N10, 90-01, 90C05, 90C06, 90C08, 90C10, 90C11, 90C20, 90C27, 90C35, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Integer programming: branch and bound methods**; **Linear programming: karmarkar projective algorithm**; **Maximum flow problem**; **Minimum cost flow problem**)
polynomial time algorithm
[05B35, 20F36, 20F55, 52C35, 57N65, 90C05]
(*see:* **Hyperplane arrangements in optimization**; **Linear programming: karmarkar projective algorithm**)
polynomial time algorithm *see:* efficient polynomially bounded —; nondeterministic —; strongly —; weakly —

polynomial time algorithms
[90C05]
(*see:* **Linear programming: interior point methods**)
polynomial time approximation scheme
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59, 90C60]
(*see:* **Complexity classes in optimization**; **Optimization problems in unit-disk graphs**)
polynomial time approximation scheme *see:* fully —
polynomial time computable function
[90C60]
(*see:* **Complexity classes in optimization**)
polynomial time convergence
[90C25, 90C51, 94A17]
(*see:* **Entropy optimization: interior point methods**)
polynomial time convergence
[90C25, 90C51, 94A17]
(*see:* **Entropy optimization: interior point methods**)
polynomial time deterministic algorithm
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
polynomial time interior point methods
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
polynomial time local search problems
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
polynomial time problem *see:* strongly —
polynomial time reduction
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
polynomial time reduction
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
polynomial time solution
[90C60]
(*see:* **Complexity theory: quadratic programming**)
polynomial transformation
[90C60]
(*see:* **Computational complexity theory**)
polynomial Turing reducibility
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
polynomial upper bound
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
polynomially bounded polynomial time algorithm *see:* efficient —
polynomially space-bounded Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
polynomially time-bounded Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
polynomially transformable decision problem
[90C60]
(*see:* **Complexity theory**)
polynomials *see:* least squares formal orthogonal —; Least squares orthogonal —; orthogonal —; polytope —; Robust control: schur stability of polytopes of —
polyspectrum
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
polytope
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
polytope
[90C05]
(*see:* **Carathéodory theorem**)
polytope *see:* convex —; cross —; k-way —; k-way transportation —; matroid base —; secondary —; state —; trace —
polytope polynomials
[39A11, 93C55, 93D09]
(*see:* **Robust control: schur stability of polytopes of polynomials**)
polytopes of polynomials *see:* Robust control: schur stability of —
polytopes: strategies and performances *see:* Volume computation for —
ponens *see:* checklist modus —; modus —
Pontryagin maximum principle
(49J15, 49K15, 93C10)
(*referred to in:* **Dynamic programming: continuous-time optimal control**; **Hamilton–Jacobi–Bellman equation**; **High-order maximum principle for abnormal extremals**)
(*refers to:* **Dynamic programming: continuous-time optimal control**; **Hamilton–Jacobi–Bellman equation**; **High-order maximum principle for abnormal extremals**)
pontryagin's maximum principle
[41A10, 47N10, 49K15, 49K27, 65L99, 93-XX]
(*see:* **Boundary condition iteration BCI**; **Dynamic programming: optimal control applications**; **High-order maximum principle for abnormal extremals**; **Optimization strategies for dynamic systems**)
Pontryagin maximum principle
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
Pontryagin minimum principle
[34H05, 49L20, 90C39]
(*see:* **Dynamic programming: continuous-time optimal control**)
pool adjacent violators algorithm
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
pool of cuts
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**)
pool template
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
pooling
[90C05]
(*see:* **Continuous global optimization: applications**; **Planning in the process industry**)
pooling
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)

pooling and blending problems
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
pooling problems *see:* MINLP: applications in blending and —
pools *see:* crew —
POP
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
population
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90, 92B05]
(*see:* **Broadcast scheduling problem**; **Design optimization in computational fluid dynamics**; **Genetic algorithms**; **Traveling salesman problem**)
population
[92B05]
(*see:* **Genetic algorithms**)
population drifting
[90C11, 90C90, 91B28]
(*see:* **Multicriteria decision support methodologies for auditing decisions**)
population size
[92B05]
(*see:* **Genetic algorithms**)
population size
[92B05]
(*see:* **Genetic algorithms**)
populations *see:* Volterra model of conflicting —
portfolio *see:* constant rebalanced —; market —; universal —
portfolio analysis *see:* mean-variance —
portfolio management
[68Q25, 90C26, 91B06, 91B28, 91B60]
(*see:* **Competitive ratio for portfolio management**; **Financial applications of multicriteria analysis**; **Portfolio selection and multicriteria analysis**)
portfolio management
[68Q25, 90C26, 91B28]
(*see:* **Competitive ratio for portfolio management**; **Portfolio selection and multicriteria analysis**)
portfolio management *see:* Competitive ratio for —
portfolio optimization
[91B50]
(*see:* **Financial equilibrium**)
portfolio selection
[90C20, 90C29]
(*see:* **Decision support systems with multiple criteria**; **Standard quadratic optimization problems: applications**)
portfolio selection
[90C20]
(*see:* **Standard quadratic optimization problems: applications**)
Portfolio selection: markowitz mean-variance model
(91B28)
Portfolio selection and multicriteria analysis
(91B28, 90C26)
(*referred to in:* **Bi-objective assignment problem**; **Competitive ratio for portfolio management**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Financial optimization**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**; **Robust optimization**; **Semi-infinite programming and applications in finance**; **Standard quadratic optimization problems: applications**)
(*refers to:* **Bi-objective assignment problem**; **Competitive ratio for portfolio management**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Financial optimization**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**; **Robust optimization**; **Semi-infinite programming and applications in finance**)
portfolio selection problem
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
portfolio theory
[90C27]
(*see:* **Operations research and financial markets**)
portfolio theory
[90C27]
(*see:* **Operations research and financial markets**)
portfolios *see:* frontier of efficient —
posed *see:* ill- —; well- —
posed problem *see:* ill- —; Levitin–Polyak well- —; well- —
posed problems *see:* ill- —
posed variational problem *see:* ill- —
posed variational problems *see:* Ill- —
posedness *see:* well- —
position *see:* general —
position evaluator
[90C39]
(*see:* **Neuro-dynamic programming**)
position of hyperplanes *see:* general —
position vector
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements in optimization**)

positioning *see:* Gene clustering: A novel decomposition-based clustering approach: global optimum search with enhanced —
positioning algorithm *see:* relative —
positive
[58E05, 90C30]
(*see:* **Topology of global optimization**)
positive *see:* completely —
positive basis tableau *see:* lexico- —
positive and contraction matrices *see:* completion to completely —
positive definite
[05C50, 15-XX, 15A48, 15A57, 65-XX, 90-XX, 90C05, 90C22, 90C25, 90C30, 90C33, 90C51]
(*see:* **Cholesky factorization**; **Interior point methods for semidefinite programming**; **Linear complementarity problem**; **Matrix completion problems**)
positive definite matrices
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
positive definite matrix
[65K10, 65M60]
(*see:* **Variational inequalities**)
positive definite matrix
[90C30]
(*see:* **Frank–Wolfe algorithm**; **Simplicial decomposition**)
positive definite matrix *see:* strongly —
positive definite quadratic function
[90C30]
(*see:* **Suboptimal control**)
positive definite quadratic models
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
positive definiteness
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
positive fault
[90Cxx]
(*see:* **Discontinuous optimization**)
positive gradient *see:* projected —
positive marginal value
[90C60]
(*see:* **Complexity of degeneracy**)
positive marginal values
[90C60]
(*see:* **Complexity of degeneracy**)
positive matrix *see:* completely —; partial completely —
positive minimum value
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
positive-negative-zero pattern
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
positive real numbers *see:* infinitely small —
positive (semi) definite completion problem
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
positive semi-definite quadratic binary programming
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
positive semidefinite
[05C15, 05C17, 05C35, 05C50, 05C69, 15A48, 15A57, 90C05, 90C20, 90C22, 90C25, 90C30, 90C33, 90C35, 90C51, 90C60]
(*see:* **Copositive programming**; **Interior point methods for semidefinite programming**; **Linear complementarity problem**; **Lovász number**; **Matrix completion problems**; **Quadratic knapsack**)
positive semidefinite matrices
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
positive semidefinite matrix
[65K10, 65M60]
(*see:* **Variational inequalities**)
positive semidefinite matrix *see:* bisymmetric —
positive semidefinite matrix completion problem
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
positive semidefinite symmetric matrix
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
positive semidefiniteness constraints
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
positive vector *see:* lexico- —; lexicographically —
positively *see:* dropped —
positively homogeneous
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
positively homogeneous *see:* increasing and —
positively homogeneous function
[90C26]
(*see:* **Global optimization: envelope representation**)
positively homogeneous functions on topological vector spaces *see:* Increasing and —
positively linearly dependent
[49M30, 49M37, 65K05, 90C30]
(*see:* **Practical augmented Lagrangian methods**)
positively linearly independent
[49M30, 49M37, 65K05, 90C30]
(*see:* **Practical augmented Lagrangian methods**)
positiveness of distances
[65K05, 90C27, 90C30, 90C57, 91C15]
(*see:* **Optimization-based visualization**)
positivity
[93D09]
(*see:* **Robust control**)
possible *see:* largest —
Post conditions
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
post-optimality analyses
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)

post-optimality analysis
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
post-optimality sensitivity analysis
[90C31]
(*see:* **Sensitivity and stability in NLP**)
post (risk prone, adaptive) decision *see:* ex- —
Post system
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
posterior distribution
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
posterior methods
[65K05, 90B50, 90C05, 90C29, 91B06]
(*see:* **Multi-objective optimization and decision support systems**)
posteriori principle *see:* maximum a —
postman problem *see:* Chinese —; directed Chinese —; rural —
posynomial condensation
[90C26, 90C90]
(*see:* **Global optimization in generalized geometric programming**)
posynomial monomials
[90C26, 90C90]
(*see:* **Global optimization in generalized geometric programming**)
posynomial terms
[90C26, 90C90]
(*see:* **Global optimization in generalized geometric programming**)
posynomials
[90C26, 90C90]
(*see:* **Global optimization in generalized geometric programming**)
potential
[49J40, 49J52, 49Q10, 60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 70-XX, 74K99, 74Pxx, 80-XX, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding; **Nonconvex energy functions: hemivariational inequalities**)
potential *see:* chemical —; empirical —; linear —; nonlinear —
potential efficient solutions *see:* set of —
potential energy *see:* minimum —; smoothing of the —
potential energy function
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
potential energy function *see:* Lennard-Jones —
potential function
[37A35, 49M20, 90-08, 90C05, 90C22, 90C25, 90C30, 90C51]
(*see:* **Interior point methods for semidefinite programming**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Nondifferentiable optimization: cutting plane methods**; **Potential reduction methods for linear programming**)
potential function
[37A35, 90C05, 90C25, 90C30]
(*see:* **Linear programming: karmarkar projective algorithm**; **Potential reduction methods for linear programming**; **Solving large scale and sparse semidefinite programs**)
potential function *see:* dual —; Karmarkar —; primal —; primal-dual —; Tanabe–Todd–Ye —
potential reduction
[05-XX, 90C60]
(*see:* **Complexity of degeneracy**; **Frequency assignment problem**)
potential reduction
[37A35, 90C05, 90C25, 90C30]
(*see:* **Potential reduction methods for linear programming**; **Solving large scale and sparse semidefinite programs**)
potential reduction algorithm
[37A35, 90C05]
(*see:* **Potential reduction methods for linear programming**)
potential reduction algorithm *see:* primal —; primal-dual —
potential reduction algorithms
[90C05]
(*see:* **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**)
Potential reduction methods for linear programming
(90C05, 37A35)
(*referred to in:* **Entropy optimization: interior point methods**; **Homogeneous selfdual methods for linear programming**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Entropy optimization: interior point methods**; **Homogeneous selfdual methods for linear programming**; **Interior point methods for semidefinite programming**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: solution by active sets and interior point methods**)
potential smoothing algorithm
[90C90]
(*see:* **Simulated annealing methods in protein folding**)
potentials *see:* empirical —; node —
potentials and stability in mechanics *see:* smooth —
Potts glass model
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
powell's conjecture
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
Powell's conjecture *see:* Rosen's method, global convergence, and —
Powell method
(90C30)
(*referred to in:* **Cyclic coordinate method**; **Rosenbrock method**; **Sequential simplex method**)
(*refers to:* **Cyclic coordinate method**; **Rosenbrock method**; **Sequential simplex method**)

Powell method
[90C30]
(*see:* **Powell method**)
Powell method *see:* Davidon–Fletcher– —
Powell-symmetric–Broyden method
[90C30]
(*see:* **Successive quadratic programming**)
Powell update *see:* Davidon–Fletcher– —
power balance
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
power consumption *see:* expected —
power method
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
power set *see:* fuzzy —
power system *see:* electric —
power systems *see:* Optimization in operation of electric and energy —
PP
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
PPA *see:* convergence of —
PPM
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
practical
(*see:* **Global optimization: functional forms**)
Practical augmented Lagrangian methods
(90C30, 49M30, 49M37, 65K05)
practically feasible computational solution
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
PRAM
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
PRAM
[03D15, 65K05, 65Y05, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**; **Parallel computing: models**)
PRAM *see:* CRCW —; CREW —; EREW —
pre-declared interval function
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
pre-invex function
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
pre-invex function
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
pre-invex set
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
pre-invexity with respect to a set
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
pre-matching
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
pre-matching *see:* algorithm —; maximum —
pre-matching problem *see:* maximum —
pre-order
[90C35]
(*see:* **Generalized networks**)
pre-order
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
pre-order closure of a relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
pre-order closure of a relation *see:* local —
pre-order relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
preaccumulation of the Jacobian
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
precedence/coupling constraints
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
precision interval package *see:* variable —
preconditioner
[65H10, 65J15]
(*see:* **Contraction-mapping**)
preconditioner *see:* Atkinson–Brakhage —
preconditioning
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)
preconditioning step
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
predecessor
[90C35]
(*see:* **Generalized networks**)
predefined probabilities *see:* randomly with —
predictability of complexities
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
prediction *see:* Genetic algorithms for protein structure —; tertiary structure —
prediction of crystal structures
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
prediction list
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
prediction methods *see:* Protein loop structure —
predictive control *see:* model —
Predictive method for interhelical contacts in alpha-helical proteins
(90C11)

predictor-corrector
(*see:* Global terrain methods)
predictor-corrector algorithm
[90C05]
(*see:* **Linear programming: interior point methods**)
preemptive
[90B36]
(*see:* **Planning in the process industry**; **Stochastic scheduling**)
preference
[90C29]
(*see:* **Preference modeling**; **Vector optimization**)
preference
[90C29]
(*see:* **Preference modeling**)
Preference disaggregation
(90C29, 91A99)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Preference modeling**)
preference disaggregation
[90C29]
(*see:* **Multicriteria sorting methods**)
preference disaggregation
[90C29, 91A99]
(*see:* **Multicriteria sorting methods**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**)
preference disaggregation analysis
[90C29]
(*see:* **Decision support systems with multiple criteria**; **Preference disaggregation approach: basic features, examples from financial decision making**)
preference disaggregation approach
[90C29]
(*see:* **Decision support systems with multiple criteria**)
Preference disaggregation approach: basic features, examples from financial decision making
(90C29)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference modeling**)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference modeling**)
Preference modeling
(90C29)
(*referred to in:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation**

approach: basic features, examples from financial decision making)
(*refers to:* **Bi-objective assignment problem**; **Decision support systems with multiple criteria**; **Estimating data for multicriteria decision making problems: optimization techniques**; **Financial applications of multicriteria analysis**; **Fuzzy multi-objective linear programming**; **Multicriteria sorting methods**; **Multi-objective combinatorial optimization**; **Multi-objective integer linear programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization: interaction of design and control**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multi-objective optimization: lagrange duality**; **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**; **Outranking methods**; **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**)
preference modeling
[90-XX]
(*see:* **Outranking methods**)
preference relation
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
preference threshold
[90-XX]
(*see:* **Outranking methods**)
preference value function
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
preference value function
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
preference value functions *see:* Multi-objective optimization; Interactive methods for —
preferences
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
preferences *see:* disaggregation of —; embedded family of —
preferential bidding system
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
preferred solution *see:* most —
preflow
[90C35]
(*see:* **Maximum flow problem**)
preflow-push algorithm
[90C35]
(*see:* **Maximum flow problem**)
preflow-push algorithm
[90C35]
(*see:* **Maximum flow problem**)
preflow-push algorithm *see:* generic —
preliminary design stage
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
premature convergence
[92B05]
(*see:* **Genetic algorithms**)
premature convergence
[92B05]
(*see:* **Genetic algorithms**)
premis of an inference
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
preparation
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
preprocessing
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
preprocessing
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**)
preprocessing *see:* symbolic —
preprocessing and reformulation
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
Preprocessing in stochastic programming
(90C15, 90C35, 90B10, 90B15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **Generalized benders decomposition**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete**

distributions; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
prescriptive perspective
[90C29]
(*see:* **Preference modeling**)
present value *see:* maximize net —
preserving an activity *see:* direction —
preserving assignment problem *see:* order —
presolving techniques
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**; **Planning in the process industry**)
prespecification *see:* weak —
pressure *see:* Reid vapor —
pretty-printing a declarative program
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
price *see:* branch and —; branch-and- —; dual —; pseudoshadow —; shadow —
price and cut *see:* branch and —
price-directive decomposition
[90C35]
(*see:* **Multicommodity flow problems**)
price elasticity
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
price equilibrium *see:* spatial —; Walrasian —
price equilibrium problem *see:* network structure of the spatial —; spatial —
price formulation
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
price increase *see:* dual —
price: Integer programming with column generation *see:* Branch and —
price model
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
price model
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
Price of robustness for linear optimization problems
price taker
[91B50]
(*see:* **Financial equilibrium**)
priced information
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
prices *see:* almost at equilibrium of an assignment and a set of —; best estimate using pseudoshadow —; equilibrium of an assignment and a set of —; shadow —
pricing
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
pricing derivatives
[90C27]
(*see:* **Operations research and financial markets**)
pricing model *see:* capital asset —
pricing-out
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
pricing-out
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
pricing problem
[68Q99]
(*see:* **Branch and price: Integer programming with column generation**)
pricing theory *see:* arbitrage —
Prim algorithm *see:* modified —
primal
[90C22, 90C25]
(*see:* **Copositive programming**)
primal arc
[90B35]
(*see:* **Job-shop scheduling problem**)
primal bound-improvement
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
primal cut-improvement
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
primal degenerate
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
primal degenerate basis
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Lexicographic pivoting rules**)
primal-dual
[37A35, 49M27, 90C05, 90C11, 90C25, 90C30]
(*see:* **MINLP: generalized cross decomposition**; **Potential reduction methods for linear programming**; **Solving large scale and sparse semidefinite programs**)
primal-dual algorithm
[90C05]
(*see:* **Linear programming: interior point methods**)
primal-dual algorithm
[90C25, 90C30, 90C51, 94A17]

(*see:* **Entropy optimization: interior point methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
primal-dual framework
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
primal-dual interior-point methods
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
primal-dual methods
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
primal-dual methods
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
primal-dual potential function
[37A35, 90C05, 90C25, 90C30, 90C51, 94A17]
(*see:* **Entropy optimization: interior point methods**; **Potential reduction methods for linear programming**; **Solving large scale and sparse semidefinite programs**)
primal-dual potential reduction algorithm
[37A35, 90C05]
(*see:* **Potential reduction methods for linear programming**)
primal and dual problems
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
primal-dual scaling algorithm
[90C25, 90C30]
(*see:* **Solving large scale and sparse semidefinite programs**)
primal and dual simplex algorithms
[90C05, 90C06]
(*see:* **Selfdual parametric method for linear programs**)
primal-dual solution
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
primal and dual solutions *see:* exploiting the interplay between —
primal-dual SQPIP methods
[49K20, 49M99, 90C55]
(*see:* **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
primal feasibility
[68W10, 90B15, 90C05, 90C06, 90C30, 90C31]
(*see:* **Multiparametric linear programming**; **Parametric linear programming: cost simplex algorithm**; **Stochastic network problems: massively parallel solution**)
primal feasible basis
[90C05, 90C31]
(*see:* **Multiparametric linear programming**)
primal feasible set
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
primal gap function
[90C06, 90C25, 90C30, 90C35]
(*see:* **Cost approximation algorithms**; **Simplicial decomposition algorithms**)
primal heuristic *see:* Toyoda —
primal heuristics
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
primal linear semi-infinite program
[90C05, 90C34]
(*see:* **Semi-infinite programming: methods for linear problems**)
primal (linear) semi-infinite program
[90C05, 90C34]
(*see:* **Semi-infinite programming: methods for linear problems**)
primal master problem
[90C06, 90C15]
(*see:* **Stabilization of cutting plane algorithms for stochastic linear programming problems**)
primal master problem *see:* relaxed —
primal method
[90C06, 90C08, 90C15]
(*see:* **Simple recourse problem**)
primal method *see:* Simple recourse problem: —
primal method for the simple recourse problem
[90-08, 90C05, 90C06, 90C08, 90C15]
(*see:* **Simple recourse problem: dual method**)
primal optimization problem
[90C30]
(*see:* **Lagrangian duality: BASICS**)
primal optimization problem
[90C30]
(*see:* **Lagrangian duality: BASICS**)
primal pair
[90B35]
(*see:* **Job-shop scheduling problem**)
primal potential function
[37A35, 90C05]
(*see:* **Potential reduction methods for linear programming**)
primal potential reduction algorithm
[37A35, 90C05]
(*see:* **Potential reduction methods for linear programming**)
primal problem
[15A39, 49-XX, 49L99, 49M29, 90-XX, 90C05, 90C11, 90C22, 90C25, 90C29, 90C30, 90C31, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**; **Dynamic programming: average cost per stage problems**; **Generalized benders decomposition**; **Lagrangian duality: BASICS**; **Motzkin transposition theorem**; **Multi-objective optimization: lagrange duality**; **Semidefinite programming: optimality conditions and stability**; **Theorems of the alternative and optimization**)
primal problem *see:* J-normal —; J-stable —; N-normal —; nonconvex —
primal programming problem
[90C06]
(*see:* **Saddle point theory and optimality conditions**)
primal ray
[15A39, 90C05]
(*see:* **Motzkin transposition theorem**)
primal-relaxed dual algorithm *see:* generalized —
primal-relaxed dual approach
[90C26]
(*see:* **Generalized primal-relaxed dual approach**)
primal-relaxed dual approach *see:* Generalized —

primal-scaling algorithm
[90C25, 90C30]
(*see:* **Solving large scale and sparse semidefinite programs**)
primal SD problem
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
primal SD problem *see:* equivalent —
primal simplex algorithm
[90C35]
(*see:* **Generalized networks**)
primal simplex method *see:* lexicographic —
primal solution
[90B80, 90C11]
(*see:* **Facility location with staircase costs**)
primal subproblem
[90C11, 90C31]
(*see:* **Parametric mixed integer nonlinear optimization**)
primary structure
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
primary structure
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
prime *see:* multiplicity of a —
prime representation of a feasible region
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
primitive integral vector
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
primitive partition identities
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
principal *see:* proximate optimality —
principal agent
[90C90, 91A65, 91B99]
(*see:* **Bilevel programming: applications**)
principal pivot *see:* simple —
principal pivot algorithm
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
principal pivotal transform
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
principal pivotal transformation
[90C33]
(*see:* **Linear complementarity problem**)
principal pivoting
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
principal pivoting
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
principal pivoting *see:* matrix class invariant under —
principal pivoting method
[90C33]
(*see:* **Linear complementarity problem**)
Principal pivoting methods for linear complementarity problems
(90C33, 90C20, 65K05)
(*referred to in:* **Criss-cross pivoting rules**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Linear complementarity problem**; **Linear programming**; **Order complementarity**; **Pivoting algorithms for linear programming generating two paths**; **Probabilistic analysis of simplex algorithms**; **Simplicial pivoting algorithms for integer programming**; **Topological methods in complementarity theory**)
(*refers to:* **Convex-simplex algorithm**; **Criss-cross pivoting rules**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Least-index anticycling rules**; **Lemke method**; **Lexicographic pivoting rules**; **Linear complementarity problem**; **Linear programming**; **Linear programming: interior point methods**; **Order complementarity**; **Parametric linear programming: cost simplex algorithm**; **Pivoting algorithms for linear programming generating two paths**; **Probabilistic analysis of simplex algorithms**; **Sequential simplex method**; **Topological methods in complementarity theory**)
principal submatrix
[65K05, 90C20, 90C33]
(*see:* **Linear complementarity problem**; **Principal pivoting methods for linear complementarity problems**)
principal variation path
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
principal variation splitting algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
principle *see:* argument —; auxiliary problem —; branch and bound —; Carathéodory —; disjunctive cut —; duality and maximum —; ekeland variational —; exhaustion —; extended Extremal —; extremal —; high-order local maximum —; inclusion —; Jaynes' maximum entropy —; Kataoka —; Kimura maximum —; local maximum —; local-ratio —; maximum —; maximum entropy —; maximum likelihood —; maximum a posteriori —; minimal —; minimum cross-entropy —; nonanticipative —; pontryagin's maximum —; Pontryagin minimum —; proximate optimality —; subdifferential Variational —; Wardrop first —; Wardrop second —
principle for abnormal extremals *see:* High-order maximum —
principle of Fourier *see:* mechanical —
principle: image reconstruction *see:* Maximum entropy —
principle of insufficient reason *see:* laplace's —
principle of insufficient reasoning *see:* Laplace —
principle for Lagrangian problems *see:* high-order local maximum —
principle of least effort
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)

principle of least effort
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
principle of linear programming *see:* Decomposition —
principle of machine interval arithmetic *see:* inclusion —
principle of maximum entropy
[90C25, 94A08, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**; **Maximum entropy principle: image reconstruction**)
principle of maximum entropy
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
principle of maximum entropy *see:* axiomatic derivation of the —
principle of minimum
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
principle of minimum cross-entropy
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
principle of minimum cross-entropy *see:* axiomatic derivation of the —
principle of optimality
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
principle of optimality *see:* weak —
principle of Pareto optimality of MODP
[90C31, 90C39]
(*see:* **Multiple objective dynamic programming**)
principle of transfers
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
principle of virtual work
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
principles *see:* extremum —; minimax —; variational —
printing a declarative program *see:* pretty- —
prior distribution
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
prior probability
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
priori *see:* a —
priori method *see:* a —
priori optimization *see:* a —
priorities
(*see:* **Planning in the process industry**)
priorities *see:* relative —
priorities selection
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
Prize *see:* Nobel —
prize collecting traveling salesman problem
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
probabilistic analysis
[90C90]
(*see:* **Chemical process planning**)
probabilistic analysis
[90C90]
(*see:* **Chemical process planning**)
probabilistic analysis of an algorithm
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
Probabilistic analysis of simplex algorithms
(90C05, 68Q25, 60D05, 52A22)
(*referred to in:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Linear programming**; **Pivoting algorithms for linear programming generating two paths**; **Principal pivoting methods for linear complementarity problems**; **Simplicial pivoting algorithms for integer programming**)
(*refers to:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Linear complementarity problem**; **Linear programming**; **Pivoting algorithms for linear programming generating two paths**; **Principal pivoting methods for linear complementarity problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
probabilistic collapse
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
probabilistic constrained linear programming
[90C05, 90C15]
(*see:* **Probabilistic constrained linear programming: duality theory**)
probabilistic constrained linear programming
[90C05, 90C15]
(*see:* **Probabilistic constrained linear programming: duality theory**)
Probabilistic constrained linear programming: duality theory
(90C05, 90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models:**

random objective; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

Probabilistic constrained problems: convexity theory
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

probabilistic constrained stochastic programming
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)

probabilistic constraint
[90C15]
(*see:* **Probabilistic constrained problems: convexity theory**; **Static stochastic programming models**)

probabilistic constraint *see:* integrated —; joint —; programming under —

probabilistic constraints *see:* individual —

probabilistic criterion
[90C29, 91A99]
(*see:* **Preference disaggregation**)

probabilistic estimate
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)

probabilistic method for detecting redundancy
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)

probabilistic optimization models for data classification *see:* Deterministic and —

probabilistic programming
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)
probabilistic programming
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)
probabilistic traveling salesman
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
probabilistic uncertainty
[94A17]
(*see:* **Jaynes' maximum entropy principle**)
probabilities *see:* randomly with predefined —
probability
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
probability *see:* prior —; randomly with the same —
probability density *see:* transition —
probability density function *see:* logconcave —
probability distribution
[68Q25, 90C26, 91B28]
(*see:* **Competitive ratio for portfolio management**; **Emergency evacuation, optimization modeling**; **Global optimization in batch design under uncertainty**)
probability distribution *see:* incomplete knowledge of a —; Levy —; logconcave discrete —; logconcave univariate discrete —; quasiconcave —; Tsallis —; uncertainty embedded in a —
probability distribution function *see:* multivariate —; one-dimensional marginal —; two-dimensional marginal —
probability function
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**; **Derivatives of probability and integral functions: general theory and examples**)
probability function
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**; **Extremum problems with probability functions: kernel type solution methods**)
probability function *see:* derivative of a —; gradient of a —
probability functionals
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
probability functionals *see:* Approximation of extremum problems with —
probability functions
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
probability functions: kernel type solution methods *see:* Extremum problems with —
probability integral *see:* multivariate —
probability and integral functions: general theory and examples *see:* Derivatives of —
probability integrals
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
probability integrals *see:* Approximation of multivariate —; lower bounds for multivariate —; upper bounds for multivariate —
probability matrix *see:* transition —
probability measure *see:* γ-concave —; logconcave —; logconvex —; quasiconcave —; Wiener —
probability measure space
[60G35, 65K05]
(*see:* **Differential equations and global optimization**)
probability measures *see:* Derivatives of —; regular family of —; weak convergence of —
probability metric
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)
probability-one globally convergent homotopies
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
probability-one homotopy
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
probability-one homotopy algorithm *see:* globally convergent —
probing
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
problem
[47J20, 49J40, 65K10, 90C33, 90C60]
(*see:* **Computational complexity theory**; **Solution methods for multivalued variational inequalities**)
problem *see:* 2-matching —; 3-dimensional matching —; 3D-transportation —; accessory minimum —; acyclic subdigraph —; adjoint —; airline maintenance routing —; airplane hopping —; algebraic quadratic assignment —; Alignment —; analytical approximation of a linear programming —; approximation to the —; aspatial oligopoly —; asset selling —; assignment —; astronomical —; Asymptotic properties of random multidimensional assignment —; Automatic differentiation: root problem and branch —; average cost per stage —; axial multi-index transportation —; b-matching —; backboard wiring —; ball-constrained linear —; bandwidth —; bandwidth packing —; beam segmentation —; bi-knapsack —; Bi-objective assignment —; bidimensional knapsack —; bidual —; bilateral boundary value —; bilevel programming —; bilinear programming —; bin packing —; binary constraint satisfaction —; Biquadratic assignment —; Boolean classification —; Boolean function inference —; bottleneck quadratic assignment —; bound constrained quadratic —; bounded degree minimum spanning tree —; branch —; Broadcast scheduling —; $C^{1,1}$ optimization —; calm —; canonical monotonic optimization —; capacitated arc routing —; capacitated lot-sizing —; capacitated minimum spanning arborescence —; capacitated minimum spanning tree —; capacitated transportation —; capacitated vehicle routing —; Chebyshev —; chemical equilibrium —; Chinese postman —; classical oligopoly —; classical traveling salesman —; classification —; clique —; clustering —; coercive hemivariational inequality —; combinatorial —; combinatorial optimization —; communication-free alignment —; Communication

network assignment —; complementarity —; complete master —; consistent least squares —; constant degree parallelism alignment —; constrained minimax —; constrained minimization —; constraint satisfaction —; construction of a dual —; continuous multiple criteria —; convex —; convex feasibility —; convex hull —; convex integer transportation —; convex intersection —; convex moment —; convex optimization —; convex programming —; convex quadratic knapsack —; convex regression —; convex relaxation —; convex semidefinite programming —; covering —; crew-scheduling —; cutting-stock —; data-association —; d.c. programming —; decision —; degenerate —; degree-constrained subgraph —; design —; deterministic equivalent —; deterministic shortest path —; directed Chinese postman —; direction finding —; discounted —; discounted infinite horizon —; discrete dynamic complementarity —; discrete location —; discrete multiple criteria —; discrete resource allocation —; discretized SIP —; disjunctive OA master —; dispatch —; distance-constrained vehicle routing —; distance geometry —; distance matrix completion —; dual —; dual method for the simple recourse —; dual optimization —; dual programming —; dual SD —; dual variational inequality —; duality of the linear SIP —; dynamic complementarity —; dynamic location —; dynamic two-stage stochastic programming —; dynamic vehicle routing —; Economic lot-sizing —; edge coloring —; eigenvalue —; elevator —; entry-uniqueness —; equality-constrained nonlinear programming —; Equivalence between nonlinear complementarity problem and fixed point —; equivalent primal SD —; Euclidean distance location —; Euclidean distance matrix completion —; evaluation —; express delivery —; extended linear complementarity —; extended quadratic programming —; facility location —; feasibility —; feasible —; feasible flow —; feedback arc set —; feedback set —; feedback vertex set —; fekete points —; Fermat —; finite-dimensional control —; finite-dimensional variational inequality —; finite minimax —; finite moment —; first level —; fixed charge —; fixed charge network flow —; fixed charge transportation —; fixed point —; fleet assignment —; flow —; flow-shop —; Flow shop scheduling —; follower —; fractional 0-1 programming —; fractional combinatorial optimization —; fractional programming —; Frequency assignment —; fuel mixture —; full master —; fully nonlinear —; g-group classification —; g-group classification problem (discriminant —; Gauss —; general case of the trust region —; general Fermat —; general order complementarity —; general quadratic assignment —; General routing —; Generalizations of interior point methods for the linear complementarity —; generalizations of the nonlinear complementarity —; generalized assignment —; generalized bilevel programming —; generalized complementarity —; generalized dual —; Generalized eigenvalue proximal support vector machine —; generalized least squares —; generalized linear order complementarity —; generalized mixed complementarity —; generalized network —; Generalized nonlinear complementarity —; generalized nonlinear least squares —; generalized order complementarity —; generalized semi-infinite —; generalized Weber —; geometrical —; geometrically linear —; geometrically nonlinear —; global constrained optimization —; global maximization —; global optimization —; global unconstrained optimization —; Gomez #3 —; graph —; graph bipartitioning —; graph bipartization —; graph coloring —; graph isomorphism —; graph packing —; graph partitioning —; graph Realization —; graphical traveling salesman —; Hamiltonian circuit —; Hamiltonian cycle —; hard case of the trust region —; head-body-tail —; Heuristic and metaheuristic algorithms for the traveling salesman —; hierarchical programming —; high point —; Hilbert tenth —; Hilbert's thirteenth —; hitting cycle —; horizontal linear complementarity —; hyperbolic 0-1 programming —; ill-conditioned —; ill-posed —; ill-posed variational —; image —; implicit complementarity —; implicit general order complementarity —; impossible pairs constrained path —; inductive inference —; infeasible —; infinite-dimensional generalized order complementarity —; infinite many conditions moment —; infinite moment —; initial value —; inner —; Integer linear complementary —; integer multi-index transportation —; integer optimization —; integer programming —; integral linear fractional combinatorial optimization —; interpolation —; intractible —; inventory control —; inventory ship routing —; irregular operations —; isotonic regression —; iterative solution of the Euclidean distance location —; J-normal primal —; J-stable primal —; JJT-regular —; job-shop —; Job-shop scheduling —; jointly convex —; k-index transportation —; k-level planarization —; k-way graph partitioning —; KH-regular —; knapsack —; Koopmans–Beckmann quadratic assignment —; ℓ_1 estimation —; Lagrangian dual —; Lagrangian dual optimization —; language recognition —; large residual —; large scale —; large scale nonlinear mixed integer programming —; large scale trust region —; largest empty circle —; leader —; least squares —; level planarization —; Levitin–Polyak well-posed —; line search —; linear arrangement —; linear bilevel programming —; linear complementarity —; linear complementary —; linear multiple-choice knapsack —; linear network flow —; linear optimization —; linear order complementarity —; Linear ordering —; linear programming —; linear-quadratic —; linear semidefinite programming —; linear zero-one integer —; local minimizer —; locally reduced —; location —; location-allocation —; Location routing —; locational decision —; lower —; lower-level —; LS —; m-coloring —; m-dimensional knapsack —; marriage —; master —; match-network —; matching —; matrix completion —; matrix rounding —; max-clique —; max-det —; max-min-max optimization —; max-r-Constraint Satisfaction —; maximal flow —; maximum cardinality matching —; maximum clique —; maximum constraint satisfaction —; maximum coverage location —; Maximum flow —; maximum Independent Set —; maximum partition matching —; maximum pre-matching —; maximum rank completion —; Maximum satisfiability —; maximum weight clique —; mean value —; Metaheuristic algorithms for the vehicle routing —; MILP master —; minimax —; minimax observation —; minimization —; minimum Bisection —; minimum concave

transportation —; minimum cost flow —; minimum cost network flow —; minimum cut —; minimum feedback arc set —; minimum feedback vertex (arc) set —; minimum Multiprocessor Scheduling —; minimum rank completion —; minimum spanning tree —; minimum sphere —; minimum-units —; minimum Vertex Cover —; minimum weight feedback arc set —; minimum weighted feedback vertex set —; minimum weighted graph bipartization —; MINLP: trim-loss —; minmax —; minMax Matching Subgraph —; MIQP master —; mixed integer —; mixed integer nonconvex —; mixed integer nonlinear programming —; mixed integer optimal control —; mixed integer optimization —; mixed linear complementarity —; mixed nonlinear integer programming —; Molecular distance geometry —; multi-armed restless bandit —; Multi-depot vehicle scheduling —; multi-index transportation —; multi-knapsack —; multicommodity network flow —; multiconstraint knapsack —; Multidimensional assignment —; multidimensional knapsack —; multidimensional multiple-choice knapsack —; multidimensional scaling —; multidimensional transportation —; multidimensional zero-one knapsack —; multifacility Weber —; multilevel generalized assignment —; multilevel programming —; multiperiod assignment —; multiple-choice knapsack —; multiple criteria design —; multiple knapsack —; multiWeber —; multiWeber–Rawls —; mVI —; N-normal primal —; n-queens —; nested STO —; network design —; network flow —; network structure of the spatial price equilibrium —; newsboy —; Newton step case of the trust region —; node-arc formulation of the —; node covering —; node-path formulation of the multicommodity flow —; nonconvex dual —; nonconvex network flow —; nonconvex optimization —; nonconvex primal —; nonconvex programming —; nonlinear complementarity —; nonlinear complementary —; nonlinear discretized SIP —; nonlinear dynamic network flow —; nonlinear feasibility —; nonlinear mathematical programming —; nonlinear network flow —; nonlinear optimization —; nonlinear order complementarity —; nonlinear programming —; nonlinear single commodity network flow —; nonseparable —; nonseparable optimization —; nonsmooth Dirichlet —; nonsmooth eigenvalue —; nonzero residual —; NP-complete —; NP-hard —; numerical constraint satisfaction —; numerical example of a trim-loss —; objective for a location —; ODE two-point boundary value —; one-parametric finite optimization —; open shop —; optimal assignment —; optimal control —; optimization —; order complementarity —; order preserving assignment —; outer —; p-center —; p-median —; p-median location-allocation —; package flow —; packing —; parameter identification —; parametric —; parametric linear complementarity —; parametric nonlinear complementarity —; parametric optimization —; parametric variational inequality —; path coloring —; perfect b-matching —; perfect matching —; perturbed least squares —; phase —; phase equilibrium —; phase stability —; physically linear —; physically nonlinear —; piecewise linear minimum cost network flow —; planar multi-index transportation —; political districting —; polynomially transformable decision —; portfolio selection —; positive (semi) definite completion —; positive semidefinite matrix completion —; pricing —; primal —; primal master —; primal method for the simple recourse —; primal optimization —; primal programming —; primal SD —; prize collecting traveling salesman —; process planning —; Production-distribution system design —; programming —; protein folding —; prototype location —; pure network —; quadratic assignment —; quadratic programming —; Quadratic semi-assignment —; quadratic zero-one —; quasidifferentiable programming —; radio link frequency assignment —; real-world —; realisable —; recognition —; recourse —; rectangular partition —; rectilinear distance location —; regularized direction finding —; regularizing state —; relaxed —; relaxed control —; relaxed master —; relaxed primal master —; resource allocation —; resource-constrained minimum spanning tree —; restricted location —; restricted master —; ρ-regular —; right-hand side —; right-hand side perturbation —; road traveling salesman —; robust programming —; root —; rural postman —; saddle-point —; sample —; SAT-CNF —; satellite —; satisfiability —; scheduling —; second level —; selection —; selfdual —; semi-infinite optimization —; semidefinite programming —; Sensor network localization —; separable —; separable optimization —; separation —; sequencing —; set covering —; set packing —; set partitioning —; set-valued optimization —; shortest path —; simple plant location —; Simple recourse —; simultaneous Diophantine approximation —; Single-depot vehicle scheduling —; Single facility location: circle covering —; single-ratio fractional (hyperbolic) 0-1 programming —; single source shortest path tree —; singular control —; sinusoidal parameter estimation —; skorokhod —; smallest enclosing-circle —; solution of a —; solution of the alignment —; solution of the convex moment —; sorting —; sparse least squares —; spatial price equilibrium —; squared Euclidean distance location —; stability —; stable —; stable marriage —; standard moment —; standard quadratic optimization —; standard SD —; standard traffic equilibrium —; state —; static deterministic —; Steiner —; Steiner graphical traveling salesman —; Steiner minimal tree —; Steiner–Weber —; stiff —; stochastic dynamic optimization —; stochastic network —; stochastic programming —; stochastic shortest path —; stochastic transportation —; stochastic transportation and location —; stochastic vehicle routing —; strongly polynomial time —; subset feedback vertex (arc) set —; subset minimum feedback vertex (arc) set —; subset-sum —; survivable network design —; Sylvester —; symmetric multi-index transportation —; synthesis —; terminal layout —; three-dimensional transportation —; three-index transportation —; Time-dependent traveling salesman —; time optimal control —; total coloring —; total cost infinite horizon —; total least squares —; traffic assignment —; tramp steamer —; transportation —; transshipment —; Traveling purchaser —; traveling salesman —; traveling salesperson —; trim-loss —; trust region —; turbine balancing —; Turing machine solving a —; two-point boundary value —; unary optimization —; uncapacitated

facility location —; uncapacitated network flow —; uncapacitated plant location —; unconstrained —; unconstrained nonlinear least squares —; unconstrained optimization —; underlying deterministic —; undiscounted —; unilateral boundary value —; unweighted feedback vertex set —; upper —; upper level —; variational inequality —; vector variational inequality —; vehicle routing —; vehicle scheduling —; vertex (arc) deletion —; vertical linear complementarity —; warehouse location —; Weber —; Weber–Rawls —; weighted bipartite matching —; weighted graph coloring —; weighted least squares —; weighted matching —; weighted MAX-SAT —; well-conditioned —; well-posed —; zero-one integer feasibility —; zero-one knapsack —; zero-one programming —; zero residual —
problem in air traffic control *see:* ground delay —
Problem (ATSP) *see:* asymmetric Traveling Salesman —
problem with attraction and repulsion *see:* Global optimization in Weber's —; Weber —
problem with backhauls *see:* vehicle routing —
problem with bounded cost per stage *see:* discounted —
problem: branch & cut algorithms *see:* Stable set —
problem and branch problem *see:* Automatic differentiation: root —
problem of the calculus of variations *see:* inverse —
problem, CMO *see:* Contact map overlap maximization —
problem decomposition
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
problem description
[90C30, 90C35]
(*see:* **Optimization in water resources**)
problem (discriminant problem) *see:* g-group classification —
problem: dual method *see:* Simple recourse —
problem equivalence
[90C60]
(*see:* **Computational complexity theory**)
problem of finding shortest paths
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
problem and fixed point problem *see:* Equivalence between nonlinear complementarity —
problem formulation
[68M12, 90B18, 90C11, 90C30]
(*see:* **Optimization in ad hoc networks**)
problem formulation *see:* multilevel —
problem formulations *see:* Stochastic optimal stopping: —
problem with friction *see:* coupled unilateral contact —
problem generators *see:* Combinatorial test problems and —
problem for input-output tables *see:* triangulation —
problem instance
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
problem instance *see:* size of a —
problem instance in time m *see:* algorithm solving a —
problem integration
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
problem involving QD-superpotentials *see:* convex variational inequality for an elastostatic —; elastostatic —; variational equality for an elastostatic —
problem, MAX-CUT *see:* Maximum cut —
problem with minimum number of Steiner points *see:* Steiner tree —
problem modeling
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
Problem (MP) *see:* minimum Partition —
problem on a network *see:* 1-median —; covering —; p-center —
problem with nonnegative lower bounds *see:* maximum flow —
problem with nonunit capacity
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
problem in OR *see:* multifacility —; single-criterion —; single-facility —; unweighted —; weighted —
problem parameters
[90C60]
(*see:* **Computational complexity theory**)
problem: primal method *see:* Simple recourse —
problem principle *see:* auxiliary —
problem and a projected dynamical system *see:* variational inequality —
problem in protein folding: αBB global optimization approach *see:* Multiple minima —
problem regular in the sense of Jongen–Jonker–Twilt
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
problem regular in the sense of Kojima–Hirabayashi
[65K05, 65K10, 90C20, 90C25, 90C26, 90C29, 90C30, 90C31, 90C33, 90C34]
(*see:* **Parametric optimization: embeddings, path following and singularities**)
problem representation
[90C30]
(*see:* **Cost approximation algorithms**)
problem result
[90C60]
(*see:* **Complexity theory**)
problem for the Rosen method *see:* global convergence —
Problem (SCP) *see:* stacker Crane —
problem in semi-infinite programming *see:* reduced —
problem with simultaneous pickups and deliveries *see:* Vehicle routing —
problem, SNLP *see:* Semidefinite programming and the sensor network localization —
problem solution
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
problem solving *see:* restriction to the solution set in —
problem solving environment
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
problem of spot rate estimation *see:* τ-programmed —
problem in SQP *see:* quadratic programming —
problem in standard form *see:* linear optimization —
Problem (SVP) *see:* seismic Vessel —

problem synthesis
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
problem with time windows *see:* vehicle routing —
problem under uncertainty with perturbations *see:* minimax observation —
problem variables *see:* separable —
problem in X-ray crystallography: Shake and bake approach *see:* Phase —
problems *see:* 0–1 mixed integer —; algorithms for isotonic regression —; approximate methods for solving vehicle routing —; Approximations to robust conic optimization —; average cost per stage —; barrier location —; bilevel programming —; Bottleneck steiner tree —; cargo routing —; classification of hard —; combinatorial optimization —; Combinatorial optimization algorithms in resource allocation —; Complexity and large-scale least squares —; computational complexity of optimization —; constrained minimax —; constraint satisfaction —; constructive methods for solving vehicle routing —; continuous constraint satisfaction —; Continuous reformulations of discrete-continuous optimization —; Control —; convex —; Convex envelopes in optimization —; convex and nonconvex programming —; decomposition algorithms for nonconvex minimization —; Decomposition algorithms for the solution of multistage mean-variance optimization —; design —; discrete monotonic optimization —; discretization of optimization —; discretized optimal control —; distributed optimal control —; distribution —; dynamic network flow —; Dynamic programming: average cost per stage —; Dynamic programming: discounted —; Dynamic programming: stochastic shortest path —; Dynamic programming: undiscounted —; equilibrium —; equivalence classes of —; exact methods for solving vehicle routing —; extended linear programming —; Facilities layout —; facility location —; Feedback set —; fixed demand traffic network —; formulation and solution of inverse —; General moment optimization —; Generalized monotonicity: applications to variational inequalities and equilibrium —; Global optimization algorithms for financial planning —; Global optimization: application to phase equilibrium —; Global optimization in location —; Hemivariational inequalities: eigenvalue —; Hemivariational inequalities: static —; high-order local maximum principle for Lagrangian —; "hit-or-miss" decision —; ill-posed —; Ill-posed variational —; implicit variational —; indefinite quadratic —; inequality —; infinite horizon —; integer programming —; Integrated vehicle and duty scheduling —; Interval analysis: application to chemical engineering design —; Interval analysis: nondifferentiable —; Isotonic regression —; knapsack —; Kuhn–Tucker conditions for quadratic programming sub- —; language recognition —; Laplace method and applications to optimization —; large nonlinear multicommodity flow —; Large scale trust region —; Least squares —; linear complementarity —; linear mixed integer —; linear SIP —; linearly constrained optimization —; Maritime inventory routing —; Matrix completion —; maximum flow —; Minimum concave transportation —; minisum —; MINLP: applications in blending and pooling —; Mixed integer classification —; Modeling difficult optimization —; multi-criteria —; multi-depot vehicle scheduling —; multi-index assignment —; Multi-index transportation —; Multi-objective fractional programming —; Multicommodity flow —; Multidimensional knapsack —; Multifacility and restricted location —; multiphase Steiner —; multistage —; N-adic assignments —; Network design —; Network location: covering —; Nonconvex network flow —; nondegenerate —; Nondifferentiable optimization: minimax —; nonlinear assignment —; nonlinear complementarity —; Nonlinear least squares —; nonlinear multicommodity flow —; nonlinear network flow —; nonlinear optimization —; Nonoriented multicommodity flow —; nonsmooth optimization —; optimal design —; optimization —; Optimization in boolean classification —; parametric complementarity —; partly convex —; Piecewise linear network flow —; PLS —; polynomial time local search —; pooling and blending —; Price of robustness for linear optimization —; primal and dual —; Principal pivoting methods for linear complementarity —; quadratic generalized network —; quasidifferentiable —; reducibility of —; reducible —; second order Lagrangian theory of CNSO —; semi-infinite optimization —; Semi-infinite programming and control —; Semi-infinite programming: methods for linear —; Sensitivity analysis of complementarity —; Sensitivity analysis of variational inequality —; Sequential quadratic programming: interior point methods for distributed optimal control —; Set covering, packing and partitioning —; shortest path tree —; simulation —; Simultaneous estimation and optimization of nonlinear —; Single-depot vehicle scheduling —; solution of bilevel programming —; sorting —; Splitting method for linear complementarity —; SQP optimization in industrial —; stability analysis of optimization —; Stabilization of cutting plane algorithms for stochastic linear programming —; Stackelberg —; Steiner tree —; stochastic —; stochastic linear optimization —; Stochastic quasigradient methods in minimax —; stochastic shortest path —; Stochastic transportation and location —; Stochastic vehicle routing —; substationarity —; three-index assignment —; toy —; transformation of —; traveling salesman —; variational —; variational inequality —
problems: algorithms *see:* Standard quadratic optimization —
problems: applications *see:* Standard quadratic optimization —
problems: convexity theory *see:* Probabilistic constrained —
problems with a fixed number of vehicles *see:* Vehicle scheduling —
problems with massive data sets *see:* least squares —
problems: massively parallel solution *see:* Stochastic network —
problems method *see:* extended support —; supports —
problems with multiple types of vehicles *see:* Vehicle scheduling —
problems in optical networks *see:* Integer linear programs for routing and protection —
problems and optimization *see:* Plant layout —
problems: optimization techniques *see:* Estimating data for multicriteria decision making —

problems, overview *see:* Dynamic programming: infinite horizon —
problems with probability functionals *see:* Approximation of extremum —
problems with probability functions: kernel type solution methods *see:* Extremum —
problems and problem generators *see:* Combinatorial test —
problems and proof methodology *see:* NP-complete —
problems solution method *see:* support —
problems with spatial interaction *see:* Facility location —
problems with staircase costs *see:* convex piecewise linearization in facility location —; heuristics of facility location —; linearization in facility location —; solution of facility location —
problems: theory *see:* Standard quadratic optimization —
problems with time constraints *see:* vehicle scheduling —
problems with travel demand functions *see:* elastic demand traffic network —
problems in unit-disk graphs *see:* Optimization —
problems and variational inequalities *see:* Nonsmooth and smoothing methods for nonlinear complementarity —
procedure *see:* alternating —; anti-cycling —; arc oriented construction —; arc separation —; bB —; best arc construction —; best node construction —; branch and cut —; Direct search Luus—Jaakola optimization —; exact —; greedy randomized adaptive search —; heuristic —; improved —; inhibit —; interactive sampling —; join —; labeling —; lifting —; LJ optimization —; mixed construction —; mixed VAM construction —; model finding —; Monte-Carlo simulation —; Newton —; next shortest path —; node oriented construction —; rANDOMIZED ROUNDING —; recursive —; Rosenbrock hillclimbing —; roulette wheel —; S- —; sequential estimation —; stochastic discretization —; two-phase —; Weiszfeld —; Zionts–Wallenius —
procedures *see:* construction —; dual —; gradient based —; Greedy randomized adaptive search —; interactive —; local exchange —; savings —; second order —; solution —; statistical —
process *see:* 3PM —; adjustment —; analytic hierarchy —; application —; automated design optimization —; batch —; computational —; deformation —; diffusion —; duty scheduling —; equilibrium —; guiding —; Markov —; Markov decision —; Metropolis —; multistart —; simple homogeneous —; stochastic —; tatonnement —; trip-route choice adjustment —; Wiener —
process-Box
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)
process control
[49M37, 90C11, 90C29, 90C90]
(*see:* **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**)
process derivatives
[60J05, 90C15]
(*see:* **Derivatives of markov processes and their simulation**; **Derivatives of probability measures**)
process design
[49M37, 65C20, 65G20, 65G30, 65G40, 65H20, 90C11, 90C29, 90C90]
(*see:* **Interval analysis: application to chemical engineering design problems**; **MINLP: applications in the interaction of design and control**; **Mixed integer nonlinear programming**; **Multi-objective optimization: interaction of design and control**)
process design
[65C20, 65G20, 65G30, 65G40, 65H20, 90C90]
(*see:* **Interval analysis: application to chemical engineering design problems**)
process differentiable
[90C15]
(*see:* **Derivatives of probability measures**)
process dynamics
[49M37, 90C11]
(*see:* **MINLP: applications in the interaction of design and control**)
process flowsheet
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
process industry *see:* Planning in the —; Successive quadratic programming: applications in the —
process the LCP
[90C33]
(*see:* **Lemke method**)
process networks under uncertainty *see:* Bilevel programming framework for enterprise-wide —
process nonanticipative with respect to a filtration *see:* stochastic —
process operations
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
process optimization
[65C20, 65G20, 65G30, 65G40, 65H20, 90C90]
(*see:* **Interval analysis: application to chemical engineering design problems**)
process optimization
[65C20, 65G20, 65G30, 65G40, 65H20, 90C90]
(*see:* **Interval analysis: application to chemical engineering design problems**)
process optimization *see:* off-line —; on-line —
process planning *see:* Chemical —
process planning problem
[90C90]
(*see:* **Chemical process planning**)
process representation
[90C15]
(*see:* **Derivatives of probability measures**)
process simulation
[65C20, 65G20, 65G30, 65G40, 65H20, 90C90]
(*see:* **Interval analysis: application to chemical engineering design problems**)
process simulation
[65C20, 65G20, 65G30, 65G40, 65H20, 90C90]
(*see:* **Interval analysis: application to chemical engineering design problems**)
process simulation programs
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)

process synthesis
[49M37, 65C20, 65G20, 65G30, 65G40, 65H20, 90C11, 90C29, 90C90]
(*see:* **Interval analysis: application to chemical engineering design problems**; **Mixed integer nonlinear programming**; **Multi-objective optimization: interaction of design and control**)
process synthesis
[65C20, 65G20, 65G30, 65G40, 65H20, 90C90]
(*see:* **Interval analysis: application to chemical engineering design problems**)
process synthesis and design under uncertainty
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
process units *see:* building blocks for the —
processes *see:* abnormal —; biochemical —; Medium-term scheduling of batch —; MINLP: design and scheduling of batch —; Reactive scheduling of batch —; regenerative —; relaxation labeling —; Short-term scheduling of continuous —; synthesis of separation —
processes in interactive methods *see:* computing —
processes with resources *see:* Short-term scheduling of batch —
processes and their simulation *see:* Derivatives of markov —
processing *see:* nonGaussian signal —; nonlinear signal —; optimization in medical image —
processing with higher order statistics *see:* Signal —
processor *see:* virtual —
product *see:* Cartesian —; fuzzy relational —; fuzzy triangle —; harsh fuzzy —; **K**-local inner —; mean —; strong —
product of affine functions
[90C26, 90C31]
(*see:* **Multiplicative programming**)
product campaign *see:* mixed- —; single- —
product of concave functions
[90C26, 90C31]
(*see:* **Multiplicative programming**)
product of convex functions
[90C26, 90C31]
(*see:* **Multiplicative programming**)
product matrix
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
product of relations *see:* BK- —; circle —; self-inverse —; square —
product set *see:* Cartesian —
product of two affine functions *see:* program of minimizing a —
production *see:* modeling —
Production-distribution system design problem
(90B06)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
production functions
[49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX]
(*see:* **Resource allocation for epidemic control**)
production planning
[90-01, 90B30, 90B50, 91B32, 91B52, 91B74]
(*see:* **Bilevel programming in management**)
Production planning under uncertainty
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)
production realizing with minimal social cost
[90C33]
(*see:* **Order complementarity**)
production set
[90B30, 90B50, 90C05, 91B82]
(*see:* **Data envelopment analysis**)
production systems *see:* batch —
productivity *see:* maximization of —
products *see:* BK- —; nonassociative —
products of relations *see:* pseudo-associativity of —
profiles of conjugate-gradient algorithms for unconstrained optimization *see:* Performance —
profit
[90C26]
(*see:* **Global optimization in batch design under uncertainty**)
profit-to-time ratio cycle *see:* maximum —
program *see:* achievement scalarizing —; adjoint —; basic operations in a —; bilevel —; concave fractional —; conic —; convex multiplicative —; convex quadratic —; dual —; dual linear —; dual semi-infinite —; dual semidefinite —; extreme point mathematical —; facial —; facial disjunctive —; finite-dimensional linear —; fractional —; full master —; generalized fractional —; infeasible —; integer —; integer linear —; lattice —; linear —; linear fractional —; linear multiplicative —; linear programming —; linear semidefinite —; loss of descent in a nonlinear —; master —; max-min fractional —; min-max fractional —; mixed integer —; mixed integer nonlinear —; multi-objective fractional —; multiperiod stochastic —; multistage stochastic —; Nasa —; nondifferentiable convex —; nondifferentiable nonconvex —; nonlinear —; optimal solution of a —; partly convex —; pretty-printing a declarative —; primal (linear) semi-infinite —; quadratic —; quadratic fractional —; reduced master —; reduced quadratic —; relaxed nonlinear —; semi-infinite —; semidefinite —; semidefinite program as conic convex —; sequentially convexifiable —; simplex —; single parametric mixed integer linear —; single-ratio fractional —; stochastic —; stochastic bilevel —; sum-of-ratios fractional —; Tchebycheff —; two-stage stochastic linear —; unbounded —; weighted-sums —; zero-one integer —
program with an additional reverse convex constraint *see:* linear —
program with affine equilibrium constraints *see:* mathematical —

program as conic convex program *see:* semidefinite —
program with equilibrium constraints *see:* mathematical —
program length *see:* Shortest —
program of minimizing a convex multiplicative function [90C26, 90C31] (*see:* **Multiplicative programming**)
program of minimizing a generalized convex function [90C26] (*see:* **Global optimization in multiplicative programming**)
program of minimizing a product of two affine functions [90C26, 90C31] (*see:* **Multiplicative programming**)
program for nonlocal sensitivity analysis *see:* automated Fortran —
program with recourse *see:* stochastic —; stochastic integer —; stochastic linear —; two-stage stochastic —
program structure *see:* analysing declarative —
programmed implementation [65G20, 65G30, 65G40] (*see:* **Interval analysis: systems of nonlinear equations**)
programmed problem of spot rate estimation *see:* τ- —
programming *see:* analytical approximation of linear —; applications of parametric —; bibliography of stochastic —; Bilevel —; Bilevel fractional —; Bilevel linear —; bilinear —; binary linear —; chance constraint —; combinatorial fractional —; complexity of bilevel —; Complexity theory: quadratic —; compromise —; concave —; concave quadratic —; constrained logic —; constraint —; constraint logic —; continuous —; convex —; convex composite —; convex integer —; convex parametric —; convex quadratic —; Copositive —; cost functions in integer —; D.C. —; Decomposition principle of linear —; Design of robust model-based controllers via parametric —; differentiable convex —; differential dynamic —; difficulties in bilinear —; Disjunctive —; duality for bilevel —; Duality for semidefinite —; dynamic —; enumeration in bilevel —; Extension of the fundamental theorem of linear —; feasible direction method for nonlinear —; Feasible sequential quadratic —; flexible —; Fractional —; fractional linear —; Fractional zero-one —; full space successive quadratic —; fundamental property in convex —; fuzzy —; Fuzzy multi-objective linear —; Generalized disjunctive —; generalized geometric —; Geometric —; Global optimization in generalized geometric —; Global optimization in multiplicative —; goal —; Graph realization via semidefinite —; group relaxation in integer —; handbook on Semidefinite —; history of parametric —; Homogeneous selfdual methods for linear —; hyperbolic —; implicit function approach to bilevel —; indefinite quadratic —; infinite-dimensional linear —; instability in parametric —; integer —; integer fractional —; integer linear —; Interior point methods for semidefinite —; iterative dynamic —; Lagrangian multipliers methods for convex —; Lexicographic Goal —; linear —; linear-fractional —; linear semi-infinite —; Lipschitz —; logic —; mathematical —; matrix splitting methods in quadratic —; Maximum likelihood detection via semidefinite —; mixed integer —; mixed integer linear —; mixed integer nonlinear —; mixed-integer quadratic —; modeling language and constraint logic —; multi-objective —; multi-objective fractional —; Multi-objective integer linear —; multi-objective linear —; multi-objective mathematical —; Multi-objective mixed integer —; multi-objective (multicriteria) mixed integer —; multilevel —; Multiparametric linear —; Multiparametric mixed integer linear —; multiple objective —; Multiple objective dynamic —; multiple objective linear —; multiplicative —; multistage stochastic —; n-fold integer —; network —; Neuro-dynamic —; nonconvex —; nonconvex quadratic —; Nondifferentiable optimization: parametric —; nonlinear —; optimality in bilinear —; optimality in parametric —; paradigm of logic —; parallel —; parametric —; parametric linear —; perfect duality from the view of linear semi-infinite —; Piecewise linear —; polynomial —; positive semi-definite quadratic binary —; Potential reduction methods for linear —; Preprocessing in stochastic —; probabilistic —; probabilistic constrained linear —; probabilistic constrained stochastic —; pure zero-one —; quadratic —; quadratic concave —; reduced problem in semi-infinite —; reverse convex —; semi-infinite —; semi-infinite linear —; semidefinite —; sensitivity in nonlinear —; sequential quadratic —; signomial —; Simplicial pivoting algorithms for integer —; stability on parametric —; stable bilinear —; stable parametric —; stochastic —; stochastic dynamic —; stochastic integer —; stochastic linear —; stochastic (mixed-)integer —; structural stability in parametric —; successive quadratic —; test sets in integer —; topological stability in parametric —; two-stage stochastic —; variable factor —; Young —; zero-one integer —
programming: algebraic methods *see:* Integer —
programming algorithm *see:* continuous-time equivalent of the dynamic —; descent in a nonlinear —; dynamic —
programming: applications *see:* Bilevel —
programming: applications in distillation systems *see:* Successive quadratic —
programming: applications in engineering *see:* Bilevel —
programming and applications in finance *see:* Semi-infinite —
programming: applications in the process industry *see:* Successive quadratic —
programming: applications in the supply chain management *see:* Bilinear —
programming approach *see:* semidefinite —
programming approach for DNA transcription element identification *see:* Mixed 0-1 linear —
programming: approximation methods *see:* Semi-infinite —
programming: average cost per stage problems *see:* Dynamic —
programming: barycentric approximation *see:* Multistage stochastic —
programming with bound constraints *see:* Quadratic —
programming: branch and bound methods *see:* Integer —
programming: branch and cut algorithms *see:* Integer —
programming in clustering *see:* Dynamic —
programming with column generation *see:* Branch and price: Integer —
programming: complexity, equivalence to minmax, concave programs *see:* Bilevel linear —
programming: complexity and equivalent forms *see:* Quadratic integer —
programming/constraint programming hybrid methods *see:* Mixed integer —

programming: continuity, stability, rates of convergence *see:* Stochastic integer —
programming: continuous-time optimal control *see:* Dynamic —
programming and control problems *see:* Semi-infinite —
programming: cost simplex algorithm *see:* Parametric linear —
programming: cutting plane algorithms *see:* Integer —
programming for data mining *see:* Mathematical —
programming: decomposition and cutting planes *see:* Stochastic linear —
programming: decomposition methods *see:* Successive quadratic —
programming and determinant maximization *see:* Semidefinite —
programming: deterministic global optimization *see:* Mixed integer nonlinear bilevel —
programming: Dinkelbach method *see:* Quadratic fractional —
programming: discounted problems *see:* Dynamic —
programming: discretization methods *see:* Semi-infinite —
programming duality *see:* Integer —; linear —
programming: duality theory *see:* Probabilistic constrained linear —
Programming and Economic Analysis *see:* linear —
programming over an ellipsoid *see:* Quadratic —
programming equation *see:* continuous-time analog of the dynamic —
programming equations *see:* recursive dynamic —
programming framework for enterprise-wide process networks under uncertainty *see:* Bilevel —
programming: full space methods *see:* Successive quadratic —
programming with fuzzy coefficients *see:* multi-objective linear —
programming generating two paths *see:* Pivoting algorithms for linear —
programming: global optimization *see:* Bilevel —
programming: heat exchanger network synthesis *see:* Mixed integer linear —
programming hybrid methods *see:* Mixed integer programming/constraint —
programming: implicit function approach *see:* Bilevel —
programming: infinite horizon problems, overview *see:* Dynamic —
programming: interior point methods *see:* Linear —
programming: interior point methods for distributed optimal control problems *see:* Sequential quadratic —
programming: introduction, history and overview *see:* Bilevel —
programming: inventory control *see:* Dynamic —
programming: karmarkar projective algorithm *see:* Linear —
programming: KKT necessary optimality conditions *see:* Equality-constrained nonlinear —
programming: Klee–Minty examples *see:* Linear —
programming: lagrangian relaxation *see:* Integer —
programming in management *see:* Bilevel —
programming: mass and heat exchanger networks *see:* Mixed integer linear —
programming method *see:* piecewise sequential quadratic —
programming methods *see:* active set quadratic —; sequential quadratic —
programming: methods for linear problems *see:* Semi-infinite —
programming methods in supply chain management *see:* Mathematical —
programming: minimax approach *see:* Stochastic —
programming: mixed continuous and discrete free variables *see:* Generalized geometric —
programming model *see:* parametric —
programming models *see:* parametric —; Static stochastic —; two-stage stochastic —
programming: models and applications *see:* Multi-quadratic integer —
programming models for classification *see:* Linear —
programming models: conditional expectations *see:* Static stochastic —
programming models: random objective *see:* Stochastic —
programming and Newton's method in unconstrained optimal control *see:* Dynamic —
programming: nonanticipativity and lagrange multipliers *see:* Stochastic —
programming: numerical methods *see:* Semi-infinite —
programming: optimal control applications *see:* Dynamic —
programming: optimality conditions *see:* Generalized semi-infinite —
programming: optimality conditions and duality *see:* Bilevel —
programming: optimality conditions and stability *see:* Semidefinite —
programming paradigm *see:* general dynamic —; imperative —
programming: parallel factorization of structured matrices *see:* Stochastic —
programming and perfect duality *see:* Semi-infinite programming, semidefinite —
programming problem
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
programming problem *see:* analytical approximation of a linear —; bilevel —; bilinear —; convex —; convex semidefinite —; d.c. —; dual —; dynamic two-stage stochastic —; equality-constrained nonlinear —; extended quadratic —; fractional —; fractional 0-1 —; generalized bilevel —; hierarchical —; hyperbolic 0-1 —; integer —; large scale nonlinear mixed integer —; linear —; linear bilevel —; linear semidefinite —; mixed integer nonlinear —; mixed nonlinear integer —; multilevel —; nonconvex —; nonlinear —; nonlinear mathematical —; primal —; quadratic —; quasidifferentiable —; robust —; semidefinite —; single-ratio fractional (hyperbolic) 0-1 —; stochastic —; zero-one —
programming problem in SQP *see:* quadratic —
programming problems *see:* bilevel —; convex and nonconvex —; extended linear —; integer —; Multi-objective fractional —; solution of bilevel —; Stabilization of cutting plane algorithms for stochastic linear —
programming program *see:* linear —
programming: quasigradient method *see:* Two-stage stochastic —
programming recursion *see:* dynamic —
programming recursions *see:* dynamic —
programming relaxation *see:* linear —
programming relaxations *see:* linear —

programming with right-hand-side uncertainty, duality and applications *see:* Robust linear —
programming: second order optimality conditions *see:* Semi-infinite —
programming, semidefinite programming and perfect duality *see:* Semi-infinite —
programming and the sensor network localization problem, SNLP *see:* Semidefinite —
programming with simple integer recourse *see:* Stochastic —
programming: solution by active sets and interior point methods *see:* Successive quadratic —
programming: stochastic shortest path problems *see:* Dynamic —
programming and structural optimization *see:* Semidefinite —
programming sub-problems *see:* Kuhn–Tucker conditions for quadratic —
programming subproblem *see:* quadratic —; reduced quadratic —
programming support *see:* Multiple objective —
programming under probabilistic constraint
[90C15]
(*see:* **Static stochastic programming models**)
programming under uncertainty *see:* multi-objective linear —
programming: undiscounted problems *see:* Dynamic —
programming with variable coefficients *see:* generalized linear —
programs *see:* AD of parallel —; air traffic control and ground delay —; algorithms for stochastic bilevel —; bilevel —; Bilevel linear programming: complexity, equivalence to minmax, concave —; classification of fractional —; classifying declarative —; computationally equivalent semi-infinite —; conic convex —; discretely distributed stochastic —; dual linear —; indefinite quadratic —; linearization of —; mixed integer —; mixed integer 0–1 —; multiratio —; multy-stage stochastic —; nonconvex —; nonlinear semi-infinite —; partly convex —; polynomial —; process simulation —; Redundancy in nonlinear —; reverse convex —; Robust optimization: mixed-integer linear —; robust parametric —; Selfdual parametric method for linear —; semi-infinite —; single-ratio —; Solving large scale and sparse semidefinite —; Stochastic bilevel —; Stochastic integer —; symbolically transforming declarative —
programs with constraints *see:* weighted-sums —
programs: descent directions and efficient points *see:* Discretely distributed stochastic —
programs with recourse *see:* L-shaped method for two-stage stochastic —; two-stage stochastic —
programs with recourse and arbitrary multivariate distributions *see:* Stochastic linear —
programs with recourse: upper bounds *see:* Stochastic —
programs for routing and protection problems in optical networks *see:* Integer linear —
programs with simple integer recourse *see:* two-stage stochastic —
prohibited neighbor in tabu search
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
prohibition parameter
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
prohibition parameter T
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
project *see:* lift-and- —
project cut *see:* lift-and- —
project cuts *see:* lift-and- —
project hierarchy *see:* lift-and- —
project scheduling *see:* Static resource constrained —
projected dynamical system
[65K10, 90B15, 90C90]
(*see:* **Dynamic traffic networks**; **Variational inequalities: projected dynamical system**)
projected dynamical system
[65K10, 90B15, 90C90]
(*see:* **Dynamic traffic networks**; **Variational inequalities: projected dynamical system**)
projected dynamical system *see:* Variational inequalities: —; variational inequality problem and a —
projected dynamical systems
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)
projected gradient algorithm
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
projected gradient methods *see:* Spectral —
projected Hessian matrix of a Lagrangian function
[90C20, 90C30]
(*see:* **Successive quadratic programming: decomposition methods**)
projected Lagrangian Hessian matrix
[90C20, 90C30]
(*see:* **Successive quadratic programming: decomposition methods**)
projected negative gradient
[58E05, 90C30]
(*see:* **Topology of global optimization**)
projected negative gradient
[58E05, 90C30]
(*see:* **Topology of global optimization**)
projected positive gradient
[58E05, 90C30]
(*see:* **Topology of global optimization**)
projected positive gradient
[58E05, 90C30]
(*see:* **Topology of global optimization**)
projection
[49J52, 49M29, 65K10, 65M60, 90C11, 90C30]
(*see:* **Generalized benders decomposition**; **Nondifferentiable optimization: relaxation methods**; **Variational inequalities**)
projection
[52B12, 65K10, 65M60, 68Q25]
(*see:* **Fourier–Motzkin elimination method**; **Variational inequalities**)
projection *see:* best —; gradient —; isotone —; metric —; orthogonal —; subgradient —
projection algorithm *see:* gradient —; subgradient —
projection cone *see:* isotone —
projection constraints
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)

projection data *see:* feasibility approach to image reconstruction from —; image reconstruction from —; optimization approach to image reconstruction from —
projection matrix
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
projection matrix *see:* oblique —
projection method
[47J20, 49J40, 65K10, 90C33, 91B50]
(*see:* **Solution methods for multivalued variational inequalities**; **Walrasian price equilibrium**)
projection method
[90C30, 91B50]
(*see:* **Relaxation in projection methods**; **Walrasian price equilibrium**)
projection method *see:* Rosen gradient —
projection methods *see:* Relaxation in —; sQG —
projection operator *see:* orthogonal —
projection-restriction strategy
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
projective
[49M07, 49M10, 65K05, 90C06]
(*see:* **Performance profiles of conjugate-gradient algorithms for unconstrained optimization**)
projective algorithm
[90C05]
(*see:* **Linear programming: interior point methods**)
projective algorithm *see:* Linear programming: karmarkar —
projective transformation
[90C05]
(*see:* **Linear programming: karmarkar projective algorithm**)
projective transformation
[90C05]
(*see:* **Linear programming: karmarkar projective algorithm**)
Prolog
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
Prolog *see:* BNR- —
Prolog IV
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
prolongation *see:* axiom of —
prolongation axiom
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
promising region *see:* most —
prone, adaptive) decision *see:* ex-post (risk —
proof *see:* infeasibility —
proof on the dual side
[90C05, 90C25]
(*see:* **Young programming**)
proof methodology *see:* NP-complete problems and —
proof system *see:* propositional —
proofs *see:* NP-completeness —
propagation
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
propagation *see:* back —; constraint —; Interval —
proper
[51K05, 52C25, 65K05, 68Q25, 68U05, 90C22, 90C26, 90C30, 90C35]
(*see:* Graph realization via semidefinite programming; **Monotonic optimization**)
proper *see:* strictly —
proper coloring
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
proper efficiency
[90C29]
(*see:* **Vector optimization**)
proper k-leveled graph
[90C35]
(*see:* **Optimization in leveled graphs**)
proper policy
[49L20, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**)
proper reduction
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
properly efficient solution
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
properly efficient solution
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
properly quasimonotone operator
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
properness *see:* dual —
properties *see:* combinatorial —; descent —; Entropy optimization: shannon measure of entropy and its —; local relational —; Multi-objective optimization: pareto optimal solutions —; regularity —; testing relational —; thermodynamic —
properties and applications *see:* Pseudomonotone maps: —
properties of the configuration space *see:* local —
properties of crisp relations *see:* special —
properties of fuzzy relations *see:* special —
properties of heterogeneous relations *see:* special —
properties of homogeneous relations *see:* special —
properties of interval Newton methods *see:* existence-proving —
properties of random multidimensional assignment problem *see:* Asymptotic —
properties of relations *see:* special —; universal —
property *see:* boundary dependence —; cutworthy —; domination —; ellipsoid —; exchange —; existence —; generic —; greedy-choice —; hereditary —; homotopy —; integrality —; isotonicity —; Jacobian consistency —; localization —; Monge —; norm-dependent —; normalization —; optimal substructure —; pivoting —; scalarization —; single assignment —; uniform cone —

property-closure of a relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
property of concavity
[94A17]
(*see:* **Jaynes' maximum entropy principle**)
property in convex programming *see:* fundamental —
property of the objective function value *see:* continuity —; convexity —
property of the solution space *see:* convexity —
proposal vector
[90C15, 90C90]
(*see:* **Decomposition algorithms for the solution of multistage mean-variance optimization problems**)
proposition
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
propositional proof system
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
protection *see:* path- —
protection problems in optical networks *see:* Integer linear programs for routing and —
protein
[90C90]
(*see:* **Simulated annealing methods in protein folding**)
protein design using flexible templates *see:* De novo —
protein designUsing rigid templates *see:* De novo —
protein folding
[65K10, 90C90, 92C40]
(*see:* **Multiple minima problem in protein folding: αBB global optimization approach**; **Simulated annealing methods in protein folding**)
protein folding
[65K05, 65K10, 90C26, 90C90, 92C05, 92C40]
(*see:* **Adaptive simulated annealing and its application to protein folding**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Simulated annealing methods in protein folding**)
protein folding *see:* Adaptive simulated annealing and its application to —; Global optimization in —; Monte-Carlo simulated annealing in —; Simulated annealing methods in —
protein folding: αBB global optimization approach *see:* Multiple minima problem in —
Protein folding: generalized-ensemble algorithms
(92C05, 92C40, 92-08)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Genetic algorithms**; **Global optimization in Lennard–Jones and morse clusters**; **Graph coloring**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Simulated annealing methods in protein folding**)
protein folding problem
[65K05, 90C26]
(*see:* **Molecular structure determination: convex global underestimation**)
protein force field via linear optimization *see:* Distance dependent —
Protein loop structure prediction methods
(92C05, 92C40)
protein sequence alignment via mixed-integer linear optimization *see:* Global pairwise —
protein structure
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
protein structure prediction *see:* Genetic algorithms for —
proteins *see:* Predictive method for interhelical contacts in alpha-helical —
protocol *see:* communication —
protoconvex
[90C26]
(*see:* **Invexity and its applications**)
protoconvex
[90C26]
(*see:* **Invexity and its applications**)
prototype location problem
[90B80, 90B85]
(*see:* **Warehouse location problem**)
prototype location problem
[90B80, 90B85]
(*see:* **Warehouse location problem**)
prover *see:* resolution based theorem —
proving properties of interval Newton methods *see:* existence- —
proximal algorithms
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
proximal approximation
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
proximal bundle method
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
proximal framework
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
proximal-like method
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
proximal map
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
proximal minimization
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
proximal minimization with D-functions
[68W10, 90B15, 90C06, 90C30]

(*see:* **Stochastic network problems: massively parallel solution**)
proximal point
[90C30]
(*see:* **Cost approximation algorithms**)
proximal point algorithm
[47H05, 65J15, 90C25, 90C30, 90C55]
(*see:* **Cost approximation algorithms**; **Fejér monotonicity in convex optimization**; **Lagrangian multipliers methods for convex programming**)
proximal point algorithm *see:* entropic —; partial —; quadratic —
proximal point algorithms *see:* inexact —
proximal point approach
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
proximal point bundle method
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: relaxation methods**)
proximal point method
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
proximal point methods
[90C30]
(*see:* **Cost approximation algorithms**)
proximal point methods
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
proximal set
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
proximal support vector machine *see:* generalized eigenvalue —
proximal support vector machine problem *see:* Generalized eigenvalue —
proximate optimality principal
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
proximate optimality principle
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
proximinal
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
proximity *see:* skew-symmetric —; symmetric —
proximity data *see:* row conditional —
proximity graph model
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
proximity map
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
pRP
[49M07, 49M10, 65K, 90C06]
(*see:* **New hybrid conjugate gradient algorithms for unconstrained optimization**)
prune *see:* branch and —
pruning
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
PSA
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
PSA with dummy nodes
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
pSD
[05C50, 15A48, 15A57, 65K10, 90C25, 90C26, 90C33, 90C39, 90C51]
(*see:* **Generalizations of interior point methods for the linear complementarity problem**; **Matrix completion problems**; **Second order optimality conditions for nonlinear optimization**)
pseudo-associativity of products of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
pseudo-inverse
[65Fxx]
(*see:* **Least squares problems**)
pseudo-inverse *see:* Moore–Penrose —
pseudo-invex
[90C26]
(*see:* **Invexity and its applications**)
pseudo-invex
[90C26]
(*see:* **Invexity and its applications**)
pseudo-order
[90C29]
(*see:* **Preference modeling**)
pseudo-triangulations
[68Q20]
(*see:* **Optimal triangulations**)
pseudoconcave function *see:* U- —; U-weakly —
pseudoconnected family of sets
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
pseudoconnectedness
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
pseudoconvex
[90C26, 90C30]
(*see:* **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**; **Invexity and its applications**; **Simplicial decomposition**)
pseudoconvex
[90C26, 90C30]
(*see:* **Frank–Wolfe algorithm**; **Invexity and its applications**)
pseudoconvex *see:* η- —; strictly —
pseudoconvex function
[90C06, 90C25, 90C30, 90C35]
(*see:* **Convex-simplex algorithm**; **Simplicial decomposition algorithms**)
pseudoconvex function
[90C06, 90C25, 90C30, 90C35]
(*see:* **Convex-simplex algorithm**; **Simplicial decomposition algorithms**)
pseudoconvex function *see:* strictly —

pseudoconvexity
[90C30]
(*see:* **Simplicial decomposition**)
pseudocost
[90C11]
(*see:* **MINLP: branch and bound methods**)
pseudocost estimate
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
pseudocosts
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)
pseudocosts *see:* best estimate using —
pseudoflow
[90C35]
(*see:* **Minimum cost flow problem**)
pseudomonotone
[47J20, 49J40, 65K10, 90C26, 90C33]
(*see:* **Generalized monotone multivalued maps**; **Solution methods for multivalued variational inequalities**)
pseudomonotone bifunction
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
pseudomonotone bifunction (with respect to another)
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
pseudomonotone map *see:* strictly —
pseudomonotone mapping
[90C26]
(*see:* **Generalized monotone single valued maps**)
pseudomonotone mapping
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)
Pseudomonotone maps: properties and applications
(49J40; 49J53; 47H05; 47H04; 26B25)
(*refers to:* **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**)
pseudomonotone operator
[35A15, 46N10, 47J20, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: static problems**)
pseudomonotone operator *see:* strictly —
pseudopolynomial algorithm
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
pseudopolynomial time algorithm
[49-01, 49K45, 49N10, 90-01, 90C20, 90C27, 90C35, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Maximum flow problem**; **Minimum cost flow problem**)
pseudoquadratic constraint
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
pseudorandom
[90C05, 90C34]
(*see:* **Semi-infinite programming: methods for linear problems**)
pseudoshadow price
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
pseudoshadow prices *see:* best estimate using —
pseudosphere
[90C09, 90C10]
(*see:* **Oriented matroids**)
pSM
(*see:* **State of the art in modeling agricultural systems**)
PSPACE
[03D15, 68Q05, 68Q15, 90C60]
(*see:* **Complexity classes in optimization**; **Parallel computing: complexity classes**)
PSQP
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
psychology
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
PTAS
[90C60]
(*see:* **Complexity classes in optimization**)
PTAS *see:* Arora —; Mitchell —
PTSP
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
pull-in trip
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
pull objectives
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**)
pull-out trip
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
pumping facilities *see:* groundwater —; surface water —
purchase contract *see:* energy —
purchaser problem *see:* Traveling —
pure adaptive search
[65K05, 90C26, 90C30, 90C90]
(*see:* **Global optimization: hit and run methods**; **Random search methods**)
pure adaptive search
[90C26, 90C90]
(*see:* **Global optimization: hit and run methods**)
pure complementary gap function
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
pure exchange
[91B50]
(*see:* **Walrasian price equilibrium**)
pure exchange economic equilibrium model
[91B50]
(*see:* **Walrasian price equilibrium**)
pure exchange economy
[90C27, 90C60, 91A12, 91B50]

(*see:* **Combinatorial optimization games**; **Walrasian price equilibrium**)
pure exchange equilibrium
[91B50]
(*see:* **Walrasian price equilibrium**)
pure localization search
[65K05, 90C30]
(*see:* **Random search methods**)
pure Monte-Carlo
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**)
pure Monte-Carlo method
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: two-phase methods**)
pure network problem
[90C35]
(*see:* **Generalized networks**)
pure NP method
[90C11, 90C59]
(*see:* **Nested partitions optimization**)
pure random search
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30, 90C90]
(*see:* **Global optimization: hit and run methods**; **Random search methods**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
pure random search
[90C26, 90C90]
(*see:* **Global optimization: hit and run methods**)
pure strategy
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
pure trade
[91B50]
(*see:* **Walrasian price equilibrium**)
pure trade economic equilibrium model
[91B50]
(*see:* **Walrasian price equilibrium**)
pure trust region strategy
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
pure zero-one programming
[90C10, 90C11, 90C27, 90C57]
(*see:* **Set covering, packing and partitioning problems**)
purpose *see:* general —
purpose of models
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
purpose software library *see:* general- —
push
[90C35]
(*see:* **Maximum flow problem**)
push *see:* nonsaturating —; saturating —
push algorithm *see:* generic preflow- —; preflow- —
push objectives
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**)
PVM
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
PVM-based implementation
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
PVSPLIT
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)

Q

q *see:* CG-standard for minimizing —
q-coloring *see:* hypergraph —
Q-factor
[90C39]
(*see:* **Neuro-dynamic programming**)
Q-learning
[49L20, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**)
q-matrices
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
Q-matrix
[90C33]
(*see:* **Linear complementarity problem**)
Q-quadratic convergence
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
(Q,R) policy
[90B50]
(*see:* **Inventory management in supply chains**)
Q-splitting *see:* regular —
Q-superlinear
[65K05, 65K10, 90C06, 90C30, 90C34]
(*see:* **Feasible sequential quadratic programming**)
Q-superlinear convergence
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
qAP
[68Q25, 68R10, 68W40, 90C27, 90C59, 90C90]
(*see:* **Domination analysis in combinatorial optimization**; **Simulated annealing**)
QAP *see:* algebraic —; constant permutation —; general —; K-L type neighborhood structure for the —; Koopmans–Beckmann —
QBB global optimization method
(49M37, 65K10, 90C26, 90C30)
QC *see:* basic —
QD functions *see:* Quasidifferentiable optimization: algorithms for —
QD laws and systems of variational inequalities
[49J40, 49M05, 49S05, 74G99, 74H99, 74Pxx]
(*see:* **Quasidifferentiable optimization: variational formulations**)
QD-superpotentials *see:* convex variational inequality for an elastostatic problem involving —; elastostatic problem

involving —; variational equality for an elastostatic problem involving —
QP
[90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**)
QP
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
QP Kuhn–Tucker points *see:* multiple —
QP/NLP based branch and bound
[49M20, 90C11, 90C30]
(*see:* **Generalized outer approximation**)
QPP algorithm
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
QPwBC
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
QR algorithm *see:* implicit —
QR decomposition
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **QR factorization**)
QR factorization
(65F25, 15A23, 65F05, 65F20, 65F22)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Linear programming**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **Solving large scale and sparse semidefinite programs**; **Symmetric systems of linear equations**)
qR factorization
[15A23, 65F05, 65F20, 65F22, 65F25, 65Fxx, 90C30]
(*see:* **Generalized total least squares**; **Least squares problems**; **QR factorization**)
QR factorization
[15A23, 65F05, 65F20, 65F22, 65F25, 90C20, 90C30]
(*see:* **Orthogonal triangularization**; **Successive quadratic programming: decomposition methods**)
QR factorization *see:* rank revealing —
QR factorization with column-pivoting
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **Orthogonal triangularization**)
QR factorization using Householder transformations
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **QR factorization**)
QR method
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
(*QR*) policy *see:* Continuous review inventory models: —
QSAP
[90C08, 90C11, 90C27]
(*see:* **Quadratic semi-assignment problem**)
QSM model
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
quadratic
[49M20, 90C06, 90C10, 90C11, 90C27, 90C30, 90C57, 90C90]
(*see:* **Generalized outer approximation**; **Modeling difficult optimization problems**; **Simulated annealing**)
quadratic assignment
[05-XX, 62H30, 90C27]
(*see:* **Assignment methods in clustering**; **Frequency assignment problem**)
quadratic assignment
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
Quadratic assignment problem
(90C08, 90C11, 90C27, 90C57, 90C59)
(*referred to in:* **Assignment and matching**; **Assignment methods in clustering**; **Bi-objective assignment problem**; **Biquadratic assignment problem**; **Communication network assignment problem**; **Complexity theory: quadratic programming**; **Facilities layout problems**; **Feedback set problems**; **Frequency assignment problem**; **Graph coloring**; **Graph planarization**; **Greedy randomized adaptive search procedures**; **Linear ordering problem**; **Maximum partition matching**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Quadratic semi-assignment problem**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
(*refers to:* **Assignment and matching**; **Assignment methods in clustering**; **Bi-objective assignment problem**; **Communication network assignment problem**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **Concave programming**; **Extended cutting plane algorithm**; **Facilities layout problems**; **Feedback set problems**; **Frequency assignment problem**; **Generalized assignment problem**; **Graph coloring**; **Graph planarization**; **Greedy randomized adaptive search procedures**; **Heuristic search**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **Linear ordering problem**; **Linear programming: interior point methods**; **Maximum partition matching**; **Nondifferentiable optimization: subgradient optimization methods**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Quadratic semi-assignment problem**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
quadratic assignment problem
[05-04, 68Q25, 68R10, 68W40, 90B80, 90C05, 90C06, 90C08, 90C10, 90C11, 90C20, 90C27, 90C59]
(*see:* **Communication network assignment problem**; **Domination analysis in combinatorial optimization**; **Evolutionary algorithms in combinatorial optimization**; **Integer programming: branch and cut algorithms**; **Linear**

ordering problem; **Time-dependent traveling salesman problem**)
quadratic assignment problem
[90B80, 90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Facilities layout problems**; **Quadratic assignment problem**)
quadratic assignment problem *see:* algebraic —; bottleneck —; general —; Koopmans–Beckmann —
quadratic barrier-penalty function *see:* logarithmic- —
quadratic binary programming *see:* positive semi-definite —
quadratic (Brier) scoring rule
(*see:* **Bayesian networks**)
quadratic co-index
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
quadratic concave programming
[49M37, 90C26, 91A10]
(*see:* **Bilevel programming**)
quadratic constraint *see:* convex —; integral —
quadratic convergence
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
quadratic convergence *see:* Q- —
quadratic convergence theorem *see:* local —
quadratic fractional program
[90C32]
(*see:* **Fractional programming**)
Quadratic fractional programming: Dinkelbach method
(90C32)
(*referred to in:* **Complexity theory: quadratic programming**; **Fractional combinatorial optimization**; **Fractional programming**; **Linear ordering problem**; **Quadratic assignment problem**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
(*refers to:* **Bilevel fractional programming**; **Complexity theory: quadratic programming**; **Fractional combinatorial optimization**; **Fractional programming**; **Quadratic assignment problem**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
quadratic function
[90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**)
quadratic function *see:* convex —; piecewise linear —; positive definite —
quadratic Gaussian *see:* linear- —
quadratic generalized network problems
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
quadratic index
[49J52, 49Q10, 58E05, 74G60, 74H99, 74K99, 74Pxx, 90C30, 90C90]
(*see:* **Quasidifferentiable optimization: stability of dynamic systems**; **Topology of global optimization**)
Quadratic integer programming: complexity and equivalent forms
(65K05, 90C11, 90C20)
(*referred to in:* **Maximum cut problem, MAX-CUT**)
quadratic integer programming: models and applications *see:* Multi- —
Quadratic knapsack
(90C20, 90C60)
(*referred to in:* **Complexity theory: quadratic programming**; **Integer programming**; **Linear ordering problem**; **Multidimensional knapsack problems**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Reverse convex optimization**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
(*refers to:* **αBB algorithm**; **Complexity theory**; **Complexity theory: quadratic programming**; **Computational complexity theory**; **D.C. programming**; **Integer programming**; **Multidimensional knapsack problems**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Reverse convex optimization**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
quadratic knapsack problem *see:* convex —
quadratic Lagrangian
[90C25, 90C30]
(*see:* **Lagrangian multipliers methods for convex programming**)
quadratic maximum likelihood method *see:* iterative —
quadratic models *see:* positive definite —
quadratic nondegeneracy condition
[58E05, 90C30]
(*see:* **Topology of global optimization**)
quadratic optimization *see:* convex —
quadratic optimization problem *see:* standard —
quadratic optimization problems: algorithms *see:* Standard —
quadratic optimization problems: applications *see:* Standard —
quadratic optimization problems: theory *see:* Standard —
quadratic outer approximation
[49M20, 90C11, 90C30]
(*see:* **Generalized outer approximation**)
quadratic problem *see:* bound constrained —; linear- —
quadratic problems *see:* indefinite —
quadratic program
[65F10, 65F50, 65H10, 65K10, 90C31]
(*see:* **Globally convergent homotopy methods**; **Sensitivity and stability in NLP: approximation**)
quadratic program *see:* convex —; reduced —
quadratic programming
[05C60, 05C69, 37B25, 65K05, 65L99, 90C20, 90C25, 90C27, 90C30, 90C35, 90C59, 90C60, 91A22, 93-XX]

(*see:* **Complexity theory: quadratic programming**; **Optimization strategies for dynamic systems**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Replicator dynamics in combinatorial optimization**; **Successive quadratic programming: solution by active sets and interior point methods**)

quadratic programming
[05C60, 05C69, 37B25, 65K05, 90C20, 90C25, 90C27, 90C30, 90C31, 90C33, 90C35, 90C59, 90C60, 91A22, 91B28]
(*see:* **Complexity theory: quadratic programming**; **Frank–Wolfe algorithm**; **Linear complementarity problem**; **Operations research and financial markets**; **Portfolio selection: markowitz mean-variance model**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Replicator dynamics in combinatorial optimization**; **Sensitivity and stability in NLP: approximation**; **Successive quadratic programming: solution by active sets and interior point methods**)

quadratic programming *see:* Complexity theory: —; concave —; convex —; Feasible sequential —; full space successive —; indefinite —; matrix splitting methods in —; mixed-integer —; nonconvex —; sequential —; successive —

quadratic programming: applications in distillation systems *see:* Successive —

quadratic programming: applications in the process industry *see:* Successive —

Quadratic programming with bound constraints
(90C20, 65K05)
(*referred to in:* **Complexity theory: quadratic programming**; **Linear ordering problem**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming over an ellipsoid**; **Reverse convex optimization**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
(*refers to:* **Complexity theory: quadratic programming**; **D.C. programming**; **Linear complementarity problem**; **Linear programming: interior point methods**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming over an ellipsoid**; **Reverse convex optimization**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)

quadratic programming: decomposition methods *see:* Successive —

Quadratic programming over an ellipsoid
(90C20, 90C25)
(*referred to in:* **Complexity theory: quadratic programming**; **Linear ordering problem**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**; **Volume computation for polytopes: strategies and performances**)
(*refers to:* **Complexity theory: quadratic programming**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**; **Volume computation for polytopes: strategies and performances**)

quadratic programming: full space methods *see:* Successive —

quadratic programming: interior point methods for distributed optimal control problems *see:* Sequential —

quadratic programming method *see:* piecewise sequential —

quadratic programming methods *see:* active set —; sequential —

quadratic programming problem
[90C30, 90C90]
(*see:* **Design optimization in computational fluid dynamics**; **Successive quadratic programming: applications in distillation systems**)

quadratic programming problem *see:* extended —

quadratic programming problem in SQP
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)

quadratic programming: solution by active sets and interior point methods *see:* Successive —

quadratic programming sub-problems *see:* Kuhn–Tucker conditions for —

quadratic programming subproblem
[90C30, 90C90]
(*see:* **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**)

quadratic programming subproblem *see:* reduced —

quadratic programs *see:* indefinite —

quadratic proximal point algorithm
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)

Quadratic semi-assignment problem
(90C27, 90C11, 90C08)
(*referred to in:* **Feedback set problems**; **Graph coloring**; **Graph planarization**; **Greedy randomized adaptive search procedures**; **Linear ordering problem**; **Quadratic assignment problem**)
(*refers to:* **Feedback set problems**; **Generalized assignment problem**; **Graph coloring**; **Graph planarization**; **Greedy randomized adaptive search procedures**; **Quadratic assignment problem**)

quadratic semi-assignment problem
[90C08, 90C11, 90C27]
(*see:* **Quadratic semi-assignment problem**)

quadratic turning point
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)

quadratic zero-one problem
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)

quadrature *see:* Gaussian —

quadrature methods
[33C45, 65F20, 65F22, 65K10]
(*see:* **Least squares orthogonal polynomials**)
quadrature rule *see:* generalized Gauss —
qualification *see:* basic constraint —; constraint —; first order constraint —; generalized Slater constraint —; linear independence constraint —; linear independency constraint —; Mangasarian–Fromovitz constraint —; second order constraint —; Slater constraint —
qualification (LICQ) *see:* linear independence constraint —
qualification rule *see:* absolute —
qualifications *see:* constraint —; First order constraint —; input constraint —; Second order constraint —
qualitative class of a matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
qualitative forecasting methods
[90C26, 90C30]
(*see:* **Forecasting**)
quality *see:* establishing solution —
quality of both water environment *see:* minimizing the degradation in —
quality equalization
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
quantifier *see:* generalized —
quantifiers *see:* observational —
quantile function
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
quantitative continuity *see:* rates of —
quantitative forecasting methods
[90C26, 90C30]
(*see:* **Forecasting**)
quantity *see:* economic order —; relaxation —
quantity formulation
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
quantity model
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
quantity model
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
quantum group
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
quasi-assignment model
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
quasi-diagonal
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
quasi-Hessian
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
quasi-invex
[90C26]
(*see:* **Invexity and its applications**)
quasi-invex
[90C26]
(*see:* **Invexity and its applications**)
quasi-Newton
[90C30]
(*see:* **Cost approximation algorithms**)
quasi-Newton method
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
quasi-Newton method *see:* partitioned —; SR1 —; symmetric rank-one —
quasi-Newton method of Broyden class
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
quasi-Newton methods
[49M37, 65K05, 90C30, 90Cxx]
(*see:* **Broyden family of methods and the BFGS update**; **Cost approximation algorithms**; **Nonlinear least squares: Newton-type methods**; **Symmetric systems of linear equations**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
quasi-Newton methods
[49M37, 65K05, 65K10, 90C30]
(*see:* **ABS algorithms for optimization**; **Broyden family of methods and the BFGS update**; **Nonlinear least squares: Newton-type methods**)
quasi-Newton methods *see:* factorized —
quasi-Newton relation
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
quasi-Newton search engine
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms**)
quasi-Newton update
[15A15, 90C25, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**)
quasi-Newton update *see:* BFGS —; Broyden–Fletcher–Goldfarb–Shanno —
quasi-Newton updates
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
quasi-Newton updating *see:* inverse —
quasi-Newtonian descent direction
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
quasi-optimal solution
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
quasi-order
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
quasiconcave
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 90C15, 90C29, 91A05]
(*see:* **Generalized concavity in multi-objective optimization**; **Logconcave measures, logconvexity**; **Minimax theorems**)

quasiconcave function
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
quasiconcave function
[90C15]
(*see:* **Logconcave measures, logconvexity**)
quasiconcave function *see:* int U- —; Luc U- —; U- —
quasiconcave measure
[90C15]
(*see:* **Logconcave measures, logconvexity**)
quasiconcave probability distribution
[90C05, 90C15]
(*see:* **Probabilistic constrained linear programming: duality theory**)
quasiconcave probability measure
[90C15]
(*see:* **Logconcave measures, logconvexity**)
quasiconvex
[41A30, 46A22, 47A99, 49J35, 49J40, 54D05, 54H25, 55M20, 65K10, 90C26, 91A05]
(*see:* **Generalized monotone single valued maps**; **Invexity and its applications**; **Lipschitzian operators in best approximation by bounded or continuous functions**; **Minimax theorems**)
quasiconvex
[90C26]
(*see:* **Invexity and its applications**)
quasiconvex function
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
quasiconvex function
[90C26]
(*see:* **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**)
quasiconvex function *see:* semistrictly —; strictly —
quasiconvex medium regression
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
quasiconvex minorant *see:* greatest —
quasiconvex and umbrella regression
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
quasidifferentiability
[26B25, 26E25, 49J35, 49J40, 49J52, 49M05, 49Q10, 49S05, 52A27, 65K99, 70-08, 74A55, 74G60, 74G99, 74H99, 74K99, 74M10, 74M15, 74Pxx, 90C25, 90C26, 90C90, 90C99]
(*see:* **Quasidifferentiable optimization**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**)
quasidifferentiable
[26B25, 26E25, 49J35, 49J52, 65K99, 65Kxx, 70-08, 74A55, 74M10, 74M15, 90C25, 90C26, 90C90, 90C99, 90Cxx]
(*see:* **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**)
quasidifferentiable function
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
quasidifferentiable function
[65K05, 65Kxx, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Quasidifferentiable optimization: algorithms for QD functions**)
quasidifferentiable function *see:* Dini —; Hadamard —
quasidifferentiable functions
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
quasidifferentiable functions *see:* examples of —
Quasidifferentiable optimization
(49J52, 26B25, 90C99, 26E25)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**;

Quasidifferentiable optimization: applications; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Quasidifferentiable optimization: algorithms for hypodifferentiable functions
(49J52, 65K99)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Quasidifferentiable optimization: algorithms for QD functions
(90Cxx, 65Kxx)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational**

formulations; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Quasidifferentiable optimization: applications
(74A55, 74M10, 74M15, 65K99, 90C26, 49J35)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Quasidifferentiable optimization: applications to thermoelasticity
(74B99, 74D99, 74G99, 74H99, 47S40, 35R70)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Quasidifferentiable optimization: calculus of quasidifferentials
(49J52, 65K99, 90C90)

(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Quasidifferentiable optimization: codifferentiable functions
(65K99, 70-08, 49J52, 90C25)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Quasidifferentiable optimization: Dini derivatives, clarke derivatives
(49J52, 26E25, 52A27, 90C99)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to**

thermoelasticity; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Quasidifferentiable optimization: exact penalty methods (90Cxx)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Nondifferentiable optimization**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Quasidifferentiable optimization: optimality conditions (90Cxx)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities:**

F. E. approach; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Dini and Hadamard derivatives in optimization**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Quasidifferentiable optimization: stability of dynamic systems (74G60, 74H99, 49J52, 49Q10, 74K99, 74Pxx, 90C90)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Optimization strategies for dynamic systems**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Quasidifferentiable optimization: variational formulations (74G99, 74H99, 74Pxx, 49J40, 49M05, 49S05)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex**

energy functions: hemivariational inequalities; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
quasidifferentiable problems
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
quasidifferentiable programming problem
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
quasidifferentiable set
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
quasidifferentiable superpotential
[49J40, 49M05, 49S05, 74G99, 74H99, 74Pxx]
(*see:* **Quasidifferentiable optimization: variational formulations**)
quasidifferential
[26B25, 26E25, 49J52, 52A27, 90C99, 90Cxx]
(*see:* **Quasidifferentiable optimization**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: optimality conditions**)
quasidifferential
[90Cxx]
(*see:* **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**)
quasidifferential *see:* Dini —; Hadamard —
quasidifferential calculus
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
quasidifferential elastic boundary conditions
[35R70, 47S40, 74B99, 74D99, 74G99, 74H99]
(*see:* **Quasidifferentiable optimization: applications to thermoelasticity**)
quasidifferential functions
[90Cxx]
(*see:* **Quasidifferentiable optimization: exact penalty methods**)
quasidifferential laws *see:* variational formulation of —
quasidifferential thermal boundary conditions
[35R70, 47S40, 74B99, 74D99, 74G99, 74H99]
(*see:* **Quasidifferentiable optimization: applications to thermoelasticity**)
quasidifferential thermal boundary conditions *see:* variational formulation of —
quasidifferentials *see:* calculus of —; Quasidifferentiable optimization: calculus of —
quasigradient *see:* stochastic —; stochastic mollifier —
quasigradient method *see:* Two-stage stochastic programming: —
quasigradient methods *see:* stochastic —
quasigradient methods: applications *see:* Stochastic —
quasigradient methods in minimax problems *see:* Stochastic —
Quasigradient (SQG) methods *see:* stochastic —
quasigradients *see:* stochastic —
quasimonotone
[90C26]
(*see:* **Generalized monotone multivalued maps**)
quasimonotone bifunction
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
quasimonotone map
[90C26]
(*see:* **Generalized monotone single valued maps**)
quasimonotone map
[90C26]
(*see:* **Generalized monotone single valued maps**)
quasimonotone map *see:* semistrictly —; strictly —
quasimonotone operator
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
quasimonotone operator
[90C26]
(*see:* **Generalized monotone multivalued maps**)
quasimonotone operator *see:* properly —; semistrictly —; strictly —
quasimonotone pair
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
quasirandom
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: two-phase methods**)
Quasivariational inequalities
(49J40, 70-08, 49Q10, 74K99, 74Pxx)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization:**

codifferentiable functions; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Multilevel optimization in mechanics**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
quasivariational inequalities *see:* implicit variational inequalities and —
quasivariational inequality
[49Q10, 60G35, 65K05, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Differential equations and global optimization**; **Multilevel optimization in mechanics**)
queens problem *see:* n- —
quench *see:* simulated —
query point
[46N10, 90-00, 90C47]
(*see:* **Nondifferentiable optimization**)
question
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
question *see:* trade-off —
question-asking strategy
[90C09]
(*see:* **Inference of monotone boolean functions**)
question-asking strategy *see:* binary search-Hansel chains —; sequential Hansel chains —
queueing networks *see:* multiclass —
queueing shared-memory model
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
queuing networks
[90C15]
(*see:* **Stochastic quasigradient methods: applications**)
Quirk theorem *see:* Bassett–Maybee– —
quotient *see:* ball —; Rayleigh —; Temple —
quotient cuts
[90C35]
(*see:* **Feedback set problems**)
quotients *see:* difference —

R

r-Constraint Satisfaction Problem *see:* max- —
r-CSP *see:* max- —
R flat fuzzy number *see:* L- —
R fuzzy number *see:* L- —
r-linear convergence rate
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: subgradient optimization methods**)
R-opt heuristic
[90C27]
(*see:* **Time-dependent traveling salesman problem**)
R_{free} of unused partitions *see:* set —
$\mathbf{R}^n_+$*-upper semicontinuous function*
[90C29]
(*see:* **Vector optimization**)
R_{reac} of used partitions *see:* set —
RA algorithm
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
Rabinovich system *see:* Aizenberg– —
race *see:* Pareto —
radially continuous function
[90C26]
(*see:* **Generalized monotone multivalued maps**)
radiation exposure time
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
radiation therapy
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
radiation therapy *see:* Optimization based frameworkfor —
radio link frequency assignment problem
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
radiotherapy treatment design *see:* Beam selection in —
radius
[05A, 05C15, 05C62, 05C69, 05C85, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C, 90C27, 90C59]
(*see:* **Convex discrete optimization**; **Optimization problems in unit-disk graphs**)
radius *see:* spectral —

radius of information
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
radius of stability
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
Radon measure
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
Radon measures
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
RAF of CEP
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
Railroad
(*see:* **Railroad crew scheduling**; **Railroad locomotive scheduling**)
Railroad crew scheduling
Railroad locomotive scheduling
railroads *see:* engine routing and industrial in-plant —
RAM
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
Ramsdell method *see:* Newsam– —
Ramsey model
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
Ramsey rule of economic growth
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
Rand statistic
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
random access machine *see:* parallel —
random behavior
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
random choice *see:* rule of —
random construction
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
random interval arithmetic
[65G30, 65G40, 65K05, 90C30, 90C57]
(*see:* **Global optimization: interval analysis and balanced interval arithmetic**)
random interval arithmetic *see:* balanced —
random keys method
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
random multidimensional assignment problem *see:* Asymptotic properties of —
random numbers *see:* common —
random objective *see:* Stochastic programming models: —
random objective function
[90C15]
(*see:* **Stochastic programming models: random objective**)
random polling scheme
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
random sampling
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
random sampling
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
random sampling *see:* uniform —
random search *see:* pure —
random search algorithms
[90C26, 90C29]
(*see:* **Optimal design of composite structures**)
random search algorithms
[90C26, 90C29, 90C90]
(*see:* **Global optimization: hit and run methods**; **Optimal design of composite structures**)
random search method
[65K05, 90C30]
(*see:* **Random search methods**)
random search method *see:* adaptive —
Random search methods
(65K05, 90C30)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization: hit and run methods**; **Maximum cut problem, MAX-CUT**; **Monte-Carlo simulated annealing in protein folding**; **Optimal design of composite structures**; **Packet annealing**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
random vertex insertion (RVI)
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
random walk search
[92B05]
(*see:* **Genetic algorithms**)
random walk search
[92B05]
(*see:* **Genetic algorithms**)
randomization
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C10, 90C27, 94C10, 94C15]
(*see:* **Graph planarization**; **Maximum satisfiability problem**)
randomized adaptive search *see:* greedy —
randomized adaptive search procedure *see:* greedy —
randomized adaptive search procedures *see:* Greedy —

randomized algorithm
[60J65, 68Q25]
(*see:* **Adaptive global search**)
randomized algorithms
[05C85, 52A22, 60D05, 68Q25, 90C05]
(*see:* **Directed tree networks**; **Probabilistic analysis of simplex algorithms**)
randomized algorithms
[60J65, 68Q25]
(*see:* **Adaptive global search**)
randomized allocation scheme
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
randomized enumeration
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
randomized heuristics
[05C69, 05C85, 68W01, 90C09, 90C10, 90C59]
(*see:* **Heuristics for maximum clique and independent set**; **Optimization in boolean classification problems**)
randomized heuristics
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
randomized rounding
[05C85]
(*see:* **Directed tree networks**)
rANDOMIZED ROUNDING PROCEDURE
[49N15, 65Y20, 68W25, 90C22, 90C27]
(*see:* **Maximum likelihood detection via semidefinite programming**)
randomized setting
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
randomly generated
[92C40]
(*see:* **Monte-Carlo simulated annealing in protein folding**)
randomly with predefined probabilities
[65G30, 65G40, 65K05, 90C30, 90C57]
(*see:* **Global optimization: interval analysis and balanced interval arithmetic**)
randomly with the same probability
[65G30, 65G40, 65K05, 90C30, 90C57]
(*see:* **Global optimization: interval analysis and balanced interval arithmetic**)
Randomness
[90C60]
(*see:* **Kolmogorov complexity**)
randomness *see:* Algorithmic —
range-analysis
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
range and null space decomposition
[90C20, 90C30, 90C90]
(*see:* **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: decomposition methods**)
range and null space decomposition
[90C30, 90C90]
(*see:* **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**)
range planning *see:* long —
range space
[90C20, 90C30]
(*see:* **Successive quadratic programming: decomposition methods**)
range space
[90C20, 90C30]
(*see:* **Successive quadratic programming: decomposition methods**)
ranges *see:* Bounding derivative —
rank
[90C30]
(*see:* **Simplicial decomposition**)
rank *see:* Chvátal —; full row —; numerical —
rank completion problem *see:* maximum —; minimum —
rank determined graph
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
rank matrix completion
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
rank of a matroid
[90C09, 90C10]
(*see:* **Matroids**)
rank nonconvexity *see:* low- —
rank-one approach *see:* limited-memory symmetric —
rank one formula *see:* selfdual —
rank-one matrix
[90C25, 90C30]
(*see:* **Solving large scale and sparse semidefinite programs**)
rank-one quasi-Newton method *see:* symmetric —
rank-one update *see:* symmetric —
rank-one update formula *see:* Sherman-Morrison —
rank revealing factorization
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **Orthogonal triangularization**)
rank revealing factorization
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **Orthogonal triangularization**)
rank revealing QR factorization
[65Fxx]
(*see:* **Least squares problems**)
rank revealing URV factorization
[65Fxx]
(*see:* **Least squares problems**)
rank-two updates
[90C30]
(*see:* **Broyden family of methods and the BFGS update**)
ranking *see:* assignment —; extreme point —
ranking extreme points
[90C60]
(*see:* **Complexity of degeneracy**)
RANS code
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
Raoult law
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)

Raphson method *see:* Newton– —
RASP
[90C60]
(*see:* **Complexity classes in optimization**)
rate *see:* average nonredundancy —; average redundancy —; convergence —; geometric convergence —; local convergence —; r-linear convergence —; riskless interest —; spot interest —; superlinear convergent —; τ-estimate of the spot —
rate for bonds with constant maturities *see:* estimating the spot —
rate constraints *see:* upper and lower well oil —
rate of convergence
[90C30, 90C33]
(*see:* **Implicit lagrangian**)
rate estimation *see:* τ-programmed problem of spot —
rate of steepest ascent
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
rate of steepest descent
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
rate yield curve *see:* interest —
rateCT
(*see:* **Medium-term scheduling of batch processes**)
rates *see:* asymptotic convergence —; term structure of interest —
rates of convergence *see:* Stochastic integer programming: continuity, stability —
rates of quantitative continuity
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)
rates and stoichiometry *see:* estimation of reaction —
rating and optimization methods *see:* Credit —
ratio *see:* approximation —; bottleneck Steiner —; competitive —; consistency —; cost-to-time —; domination —; Euclidean Steiner —; k-Steiner —; method of optimal —; Sharpe —; Steiner —
ratio in Banach spaces *see:* Steiner —
ratio of biomolecular structures *see:* Steiner —
ratio cycle *see:* maximum profit-to-time —; minimum cost-to-time —
ratio disk graphs *see:* bounded —
ratio fractional (hyperbolic) 0-1 programming problem *see:* single- —
ratio fractional program *see:* single- —
ratio method *see:* likelihood —
ratio for portfolio management *see:* Competitive —
ratio principle *see:* local- —
ratio programs *see:* single- —
ratio spanning-tree *see:* minimum —
ratio test *see:* minimum —
rational choice
[90C30]
(*see:* **Global optimization based on statistical models**)
rational numbers
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
rational numbers *see:* finite —; infinitely near —
rational p-form
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
rational reaction set
[49-01, 49K10, 49M37, 90-01, 90C05, 90C27, 90C30, 90C90, 91B52]
(*see:* **Bilevel linear programming**; **Bilevel programming: global optimization**)
rational use of groundwater
[90C30, 90C35]
(*see:* **Optimization in water resources**)
rationality *see:* bounded —; individual —
rationality assumption *see:* human —
rationality factor *see:* human —
Ratios *see:* Maximization of the Smallest of Several —; maximizing a sum of —
ratios fractional program *see:* sum-of- —
Rawls objective function *see:* multifacility Weber– —
Rawls problem *see:* multiWeber– —; Weber– —
ray *see:* descent —; dual —; extremal —; primal —; termination on a secondary —
ray crystallography: Shake and bake approach *see:* Phase problem in X- —
ray diffraction data *see:* Optimization techniques for phase retrieval based on single-crystal X- —
Rayleigh quotient
[49R50, 65G20, 65G30, 65G40, 65L15, 65L60]
(*see:* **Eigenvalue enclosures for ordinary differential equations**)
Rayleigh–Ritz bound
[49R50, 65G20, 65G30, 65G40, 65L15, 65L60]
(*see:* **Eigenvalue enclosures for ordinary differential equations**)
Rayleigh–Ritz method
[49R50, 65G20, 65G30, 65G40, 65L15, 65L60]
(*see:* **Eigenvalue enclosures for ordinary differential equations**)
Rayleigh–Ritz method
[49R50, 65G20, 65G30, 65G40, 65L15, 65L60]
(*see:* **Eigenvalue enclosures for ordinary differential equations**)
rays functions on topological vector spaces *see:* Increasing and convex-along- —
razor *see:* Occam —
RCL
[90C35]
(*see:* **Feedback set problems**)
RD
[90C90]
(*see:* **MINLP: reactive distillation column synthesis**)
re-annealing
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
reachable *see:* directly left- —; directly right- —; left- —; right- —
reaction equation
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)

reaction equilibria
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
reaction equilibrium *see:* Global optimization in phase and chemical —
reaction flux estimation in lumped systems
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
reaction kinetics and transport *see:* Identification methods for —
reaction rates and stoichiometry *see:* estimation of —
reaction set *see:* rational —
reaction tangent-plane criterion
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
reaction tangent-plane criterion
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
reactive azeotropes
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
reactive distillation
[90C90]
(*see:* **MINLP: reactive distillation column synthesis**)
reactive distillation
[90C90]
(*see:* **MINLP: reactive distillation column synthesis**)
reactive distillation column synthesis *see:* MINLP: —
reactive GRASP
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Feedback set problems**; **Greedy randomized adaptive search procedures**)
Reactive scheduling of batch processes
reactive tabu search
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
reactive tabu search *see:* Hamming- —
reactive TS
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
reactor *see:* fed-batch —
real addition with order *see:* first order theory of —
real coefficients
[01A50, 01A55, 01A60]
(*see:* **Fundamental theorem of algebra**)
real interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
real interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
real number model
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26, 90C60]
(*see:* **Complexity theory**; **Information-based complexity and information-based optimization**)
real number model
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26, 90C60]
(*see:* **Complexity theory**; **Information-based complexity and information-based optimization**)
real numbers
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
real numbers *see:* infinitely small —; infinitely small negative —; infinitely small positive —
real symmetric interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
real symmetric interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
real-valued CNSO
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
real-valued CNSO *see:* extended —
real vectors space
[90C09, 90C10]
(*see:* **Oriented matroids**)
real world
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
real-world problem
[34-xx, 34Bxx, 34Lxx, 93E24]
(*see:* **Complexity and large-scale least squares problems**)
realisable problem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
realization
[51K05, 52C25, 68Q25, 68U05, 90C22, 90C35]
(*see:* Graph realization via semidefinite programming)
realization *see:* edge —
realization of an abstract group
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
realization of a matrix
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
Realization Problem *see:* graph —
realization via semidefinite programming *see:* Graph —
realizing with minimal social cost *see:* production —
reason *see:* laplace's principle of insufficient —
reasoning *see:* approximate —; interval logic system of approximate —; Laplace principle of insufficient —; point-based logic system of approximate —
rebalanced portfolio *see:* constant —
receiver-initiated
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)

receiver initiated mapping technique
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
recession functional
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)
recession functional
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)
recharge facilities *see:* controlled —
recognition *see:* Complementarity algorithms in pattern —; pattern —; statistical pattern —
recognition problem
[90C05, 90C10, 90C60]
(*see:* **Computational complexity theory**; **Simplicial pivoting algorithms for integer programming**)
recognition problem
[90C60]
(*see:* **Computational complexity theory**)
recognition problem *see:* language —
recognition problems *see:* language —
reconditioner
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
reconstruction
[92C05, 92C40]
(*see:* **Protein loop structure prediction methods**)
reconstruction *see:* Entropy optimization for image —; finite-dimensional models for entropy optimization for image —; image —; Maximum entropy principle: image —; parent node —; vector-space models for entropy optimization for image —
reconstruction from projection data *see:* feasibility approach to image —; image —; optimization approach to image —
reconstruction methods for nonconvex feasibility analysis *see:* Shape —
recourse
[90C15]
(*see:* **Stochastic programming: parallel factorization of structured matrices**)
recourse
[49M25, 90-08, 90C05, 90C06, 90C08, 90C15]
(*see:* **L-shaped method for two-stage stochastic programs with recourse**; **Simple recourse problem**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**)
recourse *see:* complete —; expected —; fixed —; full —; L-shaped method for two-stage stochastic programs with —; relatively complete —; simple —; simple integer —; stochastic integer program with —; stochastic linear program with —; stochastic program with —; Stochastic programming with simple integer —; two-stage stochastic program with —; two-stage stochastic programs with —; two-stage stochastic programs with simple integer —
recourse action
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
recourse actions
[90C10, 90C15]
(*see:* **Stochastic integer programs**)
recourse and arbitrary multivariate distributions *see:* Stochastic linear programs with —
recourse bound *see:* restricted- —
recourse decision
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**; **Stochastic programs with recourse: upper bounds**)
recourse function
[90C15]
(*see:* **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
recourse function *see:* approximating the —; expected —
recourse model
[90C15]
(*see:* **Static stochastic programming models**)
recourse models
[90C11, 90C15]
(*see:* **Stochastic programming with simple integer recourse**)
recourse models *see:* restricted —
recourse problem
[90B10, 90B15, 90C15, 90C35]
(*see:* **Preprocessing in stochastic programming**)
recourse problem *see:* dual method for the simple —; primal method for the simple —; Simple —
recourse problem: dual method *see:* Simple —
recourse problem: primal method *see:* Simple —
recourse: upper bounds *see:* Stochastic programs with —
rectangular partition
[90C27]
(*see:* **Steiner tree problems**)
rectangular partition problem
[90C27]
(*see:* **Steiner tree problems**)
rectilinear distance
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**)
rectilinear distance location *see:* Single facility location: multi-objective —
rectilinear distance location problem
[90B85]
(*see:* **Single facility location: multi-objective rectilinear distance location**)
rectilinear distances *see:* Optimizing facility location with euclidean and —
rectilinear Steiner arborescence tree
[90C27]
(*see:* **Steiner tree problems**)
rectilinear Steiner tree
[90C27]
(*see:* **Steiner tree problems**)
recurrence *see:* three-term- —
recurrence algorithm *see:* three-term- —
recurrence relation
[90C30]
(*see:* **Generalized total least squares**)
recurrent class of states
[49L99]
(*see:* **Dynamic programming: average cost per stage problems**)

recurrent neural network
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
recursion
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
recursion
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**; **Least-index anticycling rules**)
recursion *see:* adjoint —; dynamic programming —
recursions *see:* dynamic programming —
recursive algorithm
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
recursive dynamic programming equations
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
recursive dynamic programming equations
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
recursive least squares algorithm
[65Fxx]
(*see:* **Least squares problems**)
recursive Opt Matching (ROM)
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
recursive procedure
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
recursive state space search algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
reduce *see:* branch and —
reduced
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
reduced box
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
reduced cost
[68Q99, 90C35]
(*see:* **Branch and price: Integer programming with column generation**; **Generalized networks**)
reduced cost fixing
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**; **Integer programming: cutting plane algorithms**)
reduced gradient
[49M37, 65K05, 90C30]
(*see:* **Inequality-constrained nonlinear optimization**)
reduced gradient algorithm
[90C30]
(*see:* **Convex-simplex algorithm**)
reduced gradient algorithm
[90C30]
(*see:* **Convex-simplex algorithm**)
reduced gradient method *see:* Wolfe —
reduced Gröbner basis
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
reduced Hessian
[90Cxx]
(*see:* **Discontinuous optimization**)
reduced-Hessian *see:* affine- —; limited-memory affine —
reduced-Hessian algorithm *see:* affine- —
reduced-Hessian BFGS algorithm *see:* limited-memory —
reduced Hessian of a Lagrangian
[49M37, 65K05, 90C30]
(*see:* **Inequality-constrained nonlinear optimization**)
reduced Hessian SQP *see:* multiplier-free —
reduced Hessian SQP method
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
reduced master program
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
reduced model
[90C30]
(*see:* **Suboptimal control**)
reduced polyblock
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
reduced problem *see:* locally —
reduced problem in semi-infinite programming
[90C31, 90C34]
(*see:* **Semi-infinite programming: second order optimality conditions**)
reduced quadratic program
[90C20, 90C30]
(*see:* **Successive quadratic programming: decomposition methods**)
reduced quadratic programming subproblem
[90C20, 90C30]
(*see:* **Successive quadratic programming: decomposition methods**)
reduced RLT system
[90C26]
(*see:* **Reformulation-linearization technique for global optimization**)
reduced space SQP
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
Reduced VNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
reducibility
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
reducibility
[68Q25, 90C60]
(*see:* **Computational complexity theory**; **NP-complete problems and proof methodology**)
reducibility *see:* polynomial —; polynomial Turing —
reducibility of algorithms
[90C60]
(*see:* **Computational complexity theory**)

reducibility of problems
[90C60]
(*see:* **Computational complexity theory**)
reducible
[90C60]
(*see:* **Computational complexity theory**)
reducible graph *see:* cyclically —; Smith–Walford one- —
reducible problems
[90C60]
(*see:* **Computational complexity theory**)
reduction
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
reduction
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
reduction *see:* Burke–Poliquin —; coefficient —; complete —; feasible region —; Monte-Carlo sampling and variance —; polynomial time —; potential —; proper —; region —; simplex —; spherical —; successive affine —; variance —; weighting space —
reduction algorithm *see:* potential —; primal-dual potential —; primal potential —
reduction algorithms *see:* potential —
reduction Ansatz
[57R12, 90C25, 90C29, 90C30, 90C31, 90C34, 90C46]
(*see:* **Bilevel programming: optimality conditions and duality**; Generalized semi-infinite programming: optimality conditions; **Parametric global optimization: sensitivity**; **Smoothing methods for semi-infinite optimization**)
reduction Ansatz
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
reduction in auction algorithms *see:* graph —
reduction based method
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
reduction BFGS algorithm *see:* successive affine —
reduction of a constraint set
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
reduction cuts
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
reduction to finite costs
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
reduction lower bounds *see:* variance —
reduction methods for linear programming *see:* Potential —
reduction operations
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
reduction technique *see:* variance —
reductions
[05C85]
(*see:* **Directed tree networks**)
reductions *see:* ordinary NP-complete —
redundancy
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
redundancy
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
redundancy *see:* deterministic method for detecting —; probabilistic method for detecting —
Redundancy in nonlinear programs
(90C05, 90C20)
(*referred to in:* **Inequality-constrained nonlinear optimization**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Inequality-constrained nonlinear optimization**)
redundancy rate *see:* average —
redundancy test
[90C11, 90C31]
(*see:* **Multiparametric mixed integer linear programming**)
redundant constraint
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
redundant constraint *see:* relatively —
redundant restriction
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
Reeves algorithm *see:* Fletcher– —
Reeves formula *see:* Fletcher– —
Reeves method *see:* Fletcher– —
reference direction method
[90C29]
(*see:* **Multiple objective programming support**)
reference direction vector
[90C29]
(*see:* **Multiple objective programming support**)
reference point
[90C29]
(*see:* **Multiple objective programming support**)
refinement
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
refinement *see:* incremental strategy for model structure —; iterative model —; Murty least-index —
refining industry
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
reflection
[90C30]
(*see:* **Sequential simplex method**)
reflection arrangement of hyperplanes
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
reflection coefficient
[90C30]
(*see:* **Sequential simplex method**)
reflection operations
[90C30]
(*see:* **Sequential simplex method**)
reflexive closure of a relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)

reflexive relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
reflexive relation *see:* locally —
reflexivity
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
reformulation
[90C10, 90C11, 90C27, 90C57]
(*see:* **Set covering, packing and partitioning problems**)
reformulation *see:* linearized —; model —; parametric eigenvalue —; preprocessing and —; smoothing-nonsmooth —
reformulation descent
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
reformulation-Linearization/Convexification Technique
[90C26]
(*see:* **Reformulation-linearization technique for global optimization**)
reformulation-linearization/convexification techniques
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
reformulation-linearization technique
[90C09, 90C10, 90C11, 90C26, 90C27]
(*see:* **Disjunctive programming**; **Reformulation-linearization technique for global optimization**; **Time-dependent traveling salesman problem**)
reformulation-linearization technique
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
Reformulation-Linearization-Technique *see:* the —
Reformulation-linearization technique for global optimization
(90C26)
(*referred to in:* **αBB algorithm**; **Disjunctive programming**; **MINLP: branch and bound global optimization algorithm**; **MINLP: branch and bound methods**; **MINLP: global optimization with αBB**; **MINLP: logic-based methods**)
reformulation/spatial branch and bound
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
reformulation techniques
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
reformulations of discrete-continuous optimization problems *see:* Continuous —
regeneration networks
[93A30, 93B50]
(*see:* **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: mass and heat exchanger networks**)
regenerative processes
[60J05, 90C15]
(*see:* **Derivatives of markov processes and their simulation**)
regenerative set
[60J05, 90C15]
(*see:* **Derivatives of markov processes and their simulation**)
regenerative stopping times
[60J05, 90C15]
(*see:* **Derivatives of markov processes and their simulation**)
regime *see:* stationary —; transient —
region *see:* bundle trust —; complementary —; critical —; enlargement of a feasible —; exclusion —; feasible —; induced —; inducible —; large scale trust —; minimal representation of a feasible —; most promising —; prime representation of a feasible —; relaxation of a feasible —; relaxed constraint —; safe starting —; trust —; Voronoi —
region approach *see:* trust —
region of attraction
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
region of cooperation
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
region method *see:* achievable —; distrust —; trust —
region methodology *see:* trust —
region methods *see:* Nonlinear least squares: trust —; trust —
region model *see:* trust —
region network *see:* large —
region problem *see:* general case of the trust —; hard case of the trust —; large scale trust —; Newton step case of the trust —; trust —
region problems *see:* Large scale trust —
region reduction
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
region reduction *see:* feasible —
region strategy *see:* pure trust —
region technique *see:* trust —
regional demand
[90B85]
(*see:* **Single facility location: multi-objective rectilinear distance location**)
regional network
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
regions *see:* critical —; neighboring critical —; trust —
regions of stability
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
regression
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
regression *see:* convex and concave —; isotonic —; isotonic medium —; ordinal —; quasiconvex medium —; quasiconvex and umbrella —; simple order isotonic —
regression analysis
[90C26, 90C30]
(*see:* **Forecasting**)

regression analysis
[90C26, 90C30]
(*see:* **Forecasting**)
regression method
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
regression model *see:* classical linear —
regression problem *see:* convex —; isotonic —
regression problems *see:* algorithms for isotonic —; Isotonic —
Regression by special functions: algorithms and complexity
(90C26, 41A30, 62J02)
(*referred to in:* **Isotonic regression problems**)
(*refers to:* **Isotonic regression problems**)
regression trees *see:* classification and —
regret *see:* minimization of —
regret-fc and max-regret heuristics *see:* max- —
regret heuristics *see:* max-regret-fc and max- —
regret method
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
regular
[49M37, 65K05, 90C30]
(*see:* **Inequality-constrained nonlinear optimization**)
regular *see:* metrically —; p- —
regular constraints
[90C30]
(*see:* **Kuhn–Tucker optimality conditions**)
regular cost function
[90B15]
(*see:* **Dynamic traffic networks**)
regular critical direction *see:* high- —
regular family of probability measures
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
regular family of triangulations
[65M60]
(*see:* **Variational inequalities: F. E. approach**)
regular feasible point
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
regular link cost function
[90B15]
(*see:* **Dynamic traffic networks**)
regular local minimizer
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
regular matrix
[15A99, 65G20, 65G30, 65G40, 90C26]
(*see:* **Interval linear systems**)
regular matrix *see:* strongly —
regular matroid
[90C09, 90C10]
(*see:* **Matroids**)
regular measure *see:* inner —
regular operator *see:* p- —
regular point
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
regular polyhedron
[90C60]
(*see:* **Complexity of degeneracy**)
regular polyhedron
[90C60]
(*see:* **Complexity of degeneracy**)
regular problem *see:* JJT- —; KH- —; ρ- —
regular Q-splitting
[90C25, 90C33, 90C55]
(*see:* **Splitting method for linear complementarity problems**)
regular in the sense of Jongen–Jonker–Twilt *see:* problem —
regular in the sense of Kojima–Hirabayashi *see:* problem —
regular set
[90C33]
(*see:* **Order complementarity**)
regular set *see:* completely —; second order —
regular simplex
[90C30]
(*see:* **Sequential simplex method**)
regular solution
[65K10, 90C31, 90C33]
(*see:* **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**)
regular solution of the Wilson equation
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
regular stationary point
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
regular subdivision
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
regular triangulation
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
regular triangulations
[68Q20]
(*see:* **Optimal triangulations**)
regular value
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
regularity
[49M30, 49M37, 65G20, 65G30, 65G40, 65H20, 65K05, 65K10, 90C30, 93A13]
(*see:* **Interval fixed point theory**; **Multilevel methods for optimal design**; **Practical augmented Lagrangian methods**)
regularity
[49K27, 49K40, 90C30, 90C31, 90Cxx]
(*see:* **First order constraint qualifications**; **Quasidifferentiable optimization: exact penalty methods**)
regularity *see:* metric —; ρ- —
regularity assumptions
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)

regularity axiom
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
regularity condition
[90C05, 90C26, 90C30, 90C39, 90Cxx]
(*see:* **Quasidifferentiable optimization: exact penalty methods**; **Rosen's method, global convergence, and Powell's conjecture**; **Second order optimality conditions for nonlinear optimization**; **Smooth nonlinear nonconvex optimization**; **Theorems of the alternative and optimization**)
regularity condition
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
regularity condition for penalty methods
[90Cxx]
(*see:* **Quasidifferentiable optimization: exact penalty methods**)
regularity conditions
[49K27, 49K40, 90C05, 90C15, 90C26, 90C30, 90C31]
(*see:* **First order constraint qualifications**; **Global optimization using space filling**; **Image space approach to optimization**; **Probabilistic constrained linear programming: duality theory**)
regularity properties
[57R12, 90C31, 90C34]
(*see:* **Smoothing methods for semi-infinite optimization**)
regularity* *see:* metric —
regularization
[90C06, 90C15]
(*see:* **Stabilization of cutting plane algorithms for stochastic linear programming problems**)
regularization *see:* Tikhonov —; Tikhonov iterative —
regularization approach *see:* Tikhonov's —
regularization of deterministic cutting plane methods
[90C06, 90C15]
(*see:* **Stabilization of cutting plane algorithms for stochastic linear programming problems**)
regularization method *see:* iterative —
regularized direction finding problem
[90C30]
(*see:* **Frank–Wolfe algorithm**)
regularized Frank–Wolfe algorithm
[90C30]
(*see:* **Simplicial decomposition**)
regularized Frank–Wolfe algorithm
[90C30]
(*see:* **Simplicial decomposition**)
regularized Frank–Wolfe decomposition
[90C30]
(*see:* **Frank–Wolfe algorithm**)
regularized gap function
[90C30, 90C33]
(*see:* **Implicit lagrangian**)
regularized stochastic decomposition algorithm
[90C06, 90C15]
(*see:* **Stabilization of cutting plane algorithms for stochastic linear programming problems**)
regularized subproblem
[90C30]
(*see:* **Simplicial decomposition**)
regularizing state problem
[49J20, 49J52]
(*see:* **Shape optimization**)
regulation *see:* government —
Reichenbach implication
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
Reid vapor pressure
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
reinforcement learning
[49L99, 90C39]
(*see:* **Dynamic programming: average cost per stage problems**; **Neuro-dynamic programming**)
reinforcement learning
[90C39]
(*see:* **Neuro-dynamic programming**)
Reisner ideal *see:* Stanley– —
reject index for interval optimization *see:* Algorithmic improvements using a heuristic parameter —
rejection *see:* acceptance/ —
REL chart scores
[90B80]
(*see:* **Facilities layout problems**)
REL chart scores
[90B80]
(*see:* **Facilities layout problems**)
relabel
[90C35]
(*see:* **Maximum flow problem**)
related algorithm *see:* CG- —
related algorithms *see:* nonlinear CG- —
related descent *see:* gradient- —
related descent iterations *see:* Local attractors for gradient- —
related set function *see:* gradient- —
related techniques *see:* acceleration devices and —
relation *see:* α-cut of a fuzzy —; antisymmetric —; binary —; complementary —; converse —; covering —; crisp —; dominance —; equivalence —; equivalence closure of a —; functional —; fuzzy —; fuzzy outranking —; heterogeneous —; homogeneous —; inverse —; linear equality —; linear inequality —; local equivalence —; local equivalence closure of a —; local order —; local pre-order closure of a —; local tolerance —; local tolerance closure of a —; locally reflexive —; matrix representation of a —; n-ary —; negated —; onto —; order —; outranking —; partial order —; pre-order —; pre-order closure of a —; preference —; property-closure of a —; quasi-Newton —; recurrence —; reflexive —; reflexive closure of a —; secant —; separating —; state —; strictly antisymmetric —; symmetric —; symmetric interior of a —; tolerance closure of a —; trace of —; transitive —; transposed —; univalent —; valued —; weak duality —
relation of indiscernibility
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)

relation to Newton's method *see:* Gauss–Newton method: Least squares —
relational analysis
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
relational compositions
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
relational interval algebra
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
relational matrix notation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
relational model
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
relational operations *see:* matrix notation for —
relational product *see:* fuzzy —
relational properties *see:* local —; testing —
relations *see:* binary operations on —; BK-product of —; Boolean and fuzzy —; circle product of —; complementary slackness —; difference of —; foreset and afterset representation of —; fuzzy —; generalized morphisms of —; inclusion of —; k- —; legendre duality —; linear —; Morse —; outranking —; pseudo-associativity of products of —; round composition of —; self-inverse product of —; special properties of —; special properties of crisp —; special properties of fuzzy —; special properties of heterogeneous —; special properties of homogeneous —; square composition of —; square product of —; subproduct of —; superproduct of —; Tucker homogeneous systems of linear —; unary operations on —; universal properties of —
relations approach *see:* outranking —
relationship *see:* BFGS-CG —
relationships *see:* integral —
relative
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
relative complement
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
relative condition number *see:* normwise —
relative distance
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
relative duality gap
[90C25, 90C30]
(*see:* **Solving large scale and sparse semidefinite programs**)
relative entropy
[15A15, 90C25, 90C55, 90C90, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**; **Semidefinite programming and determinant maximization**)
relative measure *see:* combined —
relative minimum
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
relative minimum *see:* strict —; strong —
relative positioning algorithm
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
relative priorities
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
relative value iteration
[49L99]
(*see:* **Dynamic programming: average cost per stage problems**)
relatively complete recourse
[90B10, 90B15, 90C15, 90C35, 90C90, 91B28]
(*see:* **Preprocessing in stochastic programming**; **Robust optimization**)
relatively redundant constraint
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
relax-and-fix
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
relaxation
[49M29, 90C05, 90C06, 90C08, 90C10, 90C11, 90C30]
(*see:* **Generalized benders decomposition**; **Integer programming: branch and bound methods**; **Integer programming: lagrangian relaxation**; **Maximum constraint satisfaction: relaxations and upper bounds**)
relaxation
[90C10, 90C30]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**; **Relaxation in projection methods**)
relaxation *see:* continuous —; Decomposition techniques for MILP: lagrangian —; group —; Hull —; Integer programming: lagrangian —; Lagrange —; Lagrangian —; linear —; linear programming —; lP —; optimal —; outer approximation with equality —; surrogate —
relaxation algorithm
[90C30]
(*see:* **Cost approximation algorithms**)
relaxation and augmented penalty *see:* outer approximation with equality —
relaxation of a feasible region
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
relaxation in integer programming *see:* group —
relaxation labeling algorithm
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
relaxation labeling processes
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
relaxation method
[91B50]
(*see:* **Walrasian price equilibrium**)

relaxation method
[49J52, 90C30, 91B50]
(*see:* **Nondifferentiable optimization: relaxation methods**; **Walrasian price equilibrium**)
relaxation method *see:* Agmon–Motzkin–Fourier —
relaxation methods *see:* combined —; Nondifferentiable optimization: —
relaxation problem *see:* convex —
Relaxation in projection methods
(90C30)
(*referred to in:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Inequality-constrained nonlinear optimization**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Inequality-constrained nonlinear optimization**)
relaxation quantity
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
relaxation rule
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: subgradient optimization methods**)
relaxation step
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Cost approximation algorithms**; **Interval global optimization**)
relaxation strategy *see:* convexification/ —
relaxation with subgradient optimization *see:* Lagrangian —
Relaxation technique
(*see:* **Railroad crew scheduling**)
relaxations *see:* bounds based on semidefinite —; convex —; extended group —; Gomory —; linear programming —; tight —
relaxations and upper bounds *see:* Maximum constraint satisfaction: —
relaxed
[90B36]
(*see:* **Stochastic scheduling**)
relaxed constraint region
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)
relaxed control
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
relaxed control problem
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
relaxed dual algorithm *see:* generalized primal- —
relaxed dual approach *see:* Generalized primal- —; primal- —
relaxed master problem
[49M20, 90-08, 90C25]
(*see:* **Nondifferentiable optimization: cutting plane methods**)
relaxed multicommodity flow
[90C35]
(*see:* **Feedback set problems**)
relaxed nonlinear program
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
relaxed primal master problem
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
relaxed Problem
(*see:* **Railroad crew scheduling**)
reliability
[90C10, 90C30, 93-XX]
(*see:* **Dynamic programming: optimal control applications**; **Modeling languages in optimization: a new paradigm**)
relinking *see:* path —
rendez-vous communication
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
reorthogonalized Gram–Schmidt algorithm *see:* Daniel–Gragg–Kaufmann–Stewart —
repeated nearest neighbor (RNN)
[68Q25, 68R10, 68W40, 90C27, 90C59]
(*see:* **Domination analysis in combinatorial optimization**)
repertory grids
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
repetition
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
repetition *see:* machine —
repetitive *see:* strictly —
replacement
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
replicated network *see:* time —
replication heuristic *see:* annealed —
replicator dynamics
[90C20]
(*see:* **Standard quadratic optimization problems: algorithms**)
Replicator dynamics in combinatorial optimization
(90C27, 90C20, 90C35, 90C59, 91A22, 37B25, 05C69, 05C60)
(*referred to in:* **Combinatorial matrix analysis**; **Combinatorial optimization algorithms in resource allocation problems**; **Combinatorial optimization games**; **Evolutionary algorithms in combinatorial optimization**; **Fractional combinatorial optimization**; **Heuristics for maximum clique and independent set**; **Multi-objective combinatorial optimization**; **Neural networks for combinatorial optimization**; **Neuro-dynamic programming**; **Unconstrained optimization in neural network training**)
(*refers to:* **Combinatorial matrix analysis**; **Combinatorial optimization algorithms in resource allocation problems**; **Combinatorial optimization games**; **Evolutionary algorithms in combinatorial optimization**; **Fractional combinatorial optimization**; **Graph coloring**; **Greedy randomized adaptive search procedures**; **Heuristics for maximum clique and independent set**; **Multi-objective combinatorial optimization**; **Neural networks for**

combinatorial optimization; **Neuro-dynamic programming**; **Unconstrained optimization in neural network training**)
replicator equation
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
replicator equations
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
replicator models *see:* multipopulation —
repositioned
(*see:* **Railroad locomotive scheduling**)
representable
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
representable matroid
[90C09, 90C10]
(*see:* **Matroids**)
representation
[52B11, 52B45, 52B55, 90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**; **Volume computation for polytopes: strategies and performances**)
representation
[90C05]
(*see:* **Carathéodory theorem**)
representation *see:* compact —; declarative —; digraph —; disaggregated —; envelope —; Euclidean —; geometric (or disk) —; Global optimization: envelope —; $\mathcal{H}$- —; lookup table —; matroid —; minimal —; mixed Time —; orthonormal —; parametric —; problem —; process —; splitting variable —; unidimensional Euclidean —; V- —
representation of cutting plane coefficients *see:* statistical —
representation of a feasible region *see:* minimal —; prime —
representation of models
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
representation of a relation *see:* matrix —
representation of relations *see:* foreset and afterset —
representation theorem
[90C06, 90C25, 90C35]
(*see:* **Decomposition principle of linear programming**; **Simplicial decomposition algorithms**)
representation theorem
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
representation theorem *see:* topological —
representations *see:* compact —; necessary optimality condition without using (sub)gradients parametric —
representative set
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
reproductive
(*see:* **State of the art in modeling agricultural systems**)
repulsion *see:* Global optimization in Weber's problem with attraction and —; Weber problem with attraction and —
request *see:* global round robin —
required *see:* no pivoting —
required edge
[90B20]
(*see:* **General routing problem**)
required vertex
[90B20]
(*see:* **General routing problem**)
requirement *see:* Type I —
requirement exhaustive sequential coloring
[05-XX]
(*see:* **Frequency assignment problem**)
requirements planning
[90-02]
(*see:* **Operations research models for supply chain management and design**)
research *see:* european Journal of Operational —; GRASP in operations —; Operations —
research and financial markets *see:* Operations —
research models for supply chain management and design *see:* Operations —
reserved solution *see:* ε- —
reservoir *see:* hydro- —
residents *see:* permanent —
residents of special facilities
(*see:* **Emergency evacuation, optimization modeling**)
residual
[90C30]
(*see:* **Conjugate-gradient methods**)
residual *see:* large —; natural —; small —
residual algorithm *see:* conjugate —
residual capacity
[90C35]
(*see:* **Minimum cost flow problem**)
residual network
[90C35]
(*see:* **Maximum flow problem**; **Minimum cost flow problem**)
residual problem *see:* large —; nonzero —; zero —
residual vector
[65D10, 65K05]
(*see:* **Overdetermined systems of linear equations**)
residuals
[90C30]
(*see:* **Nonlinear least squares problems**)
residuals
[90C30]
(*see:* **Nonlinear least squares problems**)
residuation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
residues *see:* selected —
resolution *see:* single-lookahead-unit- —
resolution based theorem prover
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
resolvent equations
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
resolvent equations
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)

resolvent operator
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
resolving degeneracy
[90C60]
(*see:* **Complexity of degeneracy**)
resolving degeneracy
[90C60]
(*see:* **Complexity of degeneracy**)
resource allocation
[49-XX, 60Jxx, 65Lxx, 90B80, 90B85, 90Cxx, 91Axx, 91B32, 91Bxx, 92D30, 93-XX]
(*see:* **Facility location with externalities**; **Resource allocation for epidemic control**)
resource allocation
[49-XX, 60Jxx, 65Lxx, 90C09, 90C10, 91B32, 92D30, 93-XX]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**; **Resource allocation for epidemic control**)
Resource allocation for epidemic control
(49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**)
resource allocation problem
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
resource allocation problem *see:* discrete —
resource allocation problems *see:* Combinatorial optimization algorithms in —
resource-constrained minimum spanning tree problem
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
resource constrained project scheduling *see:* Static —
resource constrained: unified modeling frameworks *see:* Short-term scheduling —
resource constraint
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
resource-directive decomposition
[90C35]
(*see:* **Multicommodity flow problems**)
resource-payoff space
[90C30]
(*see:* **Lagrangian duality: BASICS**)
Resource planning
(*see:* **Railroad locomotive scheduling**)
resource planning *see:* water —
resource systems *see:* conjunctive use of water —
resource weighted assignment model *see:* the multi- —
resources *see:* Optimization in water —; Short-term scheduling of batch processes with —; stochastic approach to optimization in water —; surface and groundwater —
resources planning under uncertainty on hydrological exogenous inflow and demand *see:* water —
resources policies *see:* nonanticipative water —; nonanticipativity water —
respect to another) *see:* pseudomonotone bifunction (with —
respect to changes in cost coefficients *see:* sensitivity analysis with —
respect to a filtration *see:* stochastic process nonanticipative with —
respect to right-hand side changes *see:* sensitivity analysis with —
respect to a set *see:* invexity with —; pre-invexity with —; substationarity point with —
response *see:* derivatives of structural —
response mapping *see:* best —
response surface
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
response surface
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
response surface method
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
rest arcs
(*see:* **Railroad crew scheduling**)
restart
[90C06]
(*see:* **Large scale unconstrained optimization**)
restarted Lanczos method *see:* implicit —
restless bandit problem *see:* multi-armed —
restrict
[90C15]
(*see:* **Stochastic programs with recourse: upper bounds**)
restricted accessibility form of CEP
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
restricted Candidate List
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Broadcast scheduling problem**; **Feedback set problems**; **Greedy randomized adaptive search procedures**)
restricted gradient
[90Cxx]
(*see:* **Discontinuous optimization**)
restricted implicit Lagrangian
[90C30, 90C33]
(*see:* **Implicit lagrangian**)
restricted location problem
[90B85]
(*see:* **Multifacility and restricted location problems**)

restricted location problem
[90B85]
(*see:* **Multifacility and restricted location problems**)
restricted location problems *see:* Multifacility and —
restricted master problem
[90C06, 90C10, 90C11, 90C25, 90C30, 90C35, 90C57, 90C90]
(*see:* **Decomposition principle of linear programming**; **Modeling difficult optimization problems**; **Multicommodity flow problems**; **Simplicial decomposition algorithms**)
restricted master problem
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
restricted-recourse bound
[90C15]
(*see:* **Stochastic programs with recourse: upper bounds**)
restricted recourse models
[90C90, 91B28]
(*see:* **Robust optimization**)
restricted simplicial decomposition
[90C30]
(*see:* **Simplicial decomposition**)
restricted simplicial decomposition
[90C30]
(*see:* **Simplicial decomposition**)
restricted statistical inference *see:* order —
restriction
[90C30]
(*see:* **Simplicial decomposition**)
restriction *see:* inner linearization/ —; matroid —; order —; redundant —; taboo —
restriction of a matroid
[90C09, 90C10]
(*see:* **Matroids**)
restriction to the solution set in problem solving
[90C15]
(*see:* **Stochastic programs with recourse: upper bounds**)
restriction strategy *see:* projection- —
restrictions
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
restrictions *see:* control —; operational —
restrictive *see:* k- —
restrictive multilayer *see:* k- —
result *see:* problem —; strong duality —; weak duality —
result verification *see:* automatic —
resultant
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
results *see:* numerical —
results for RSM-distributions *see:* asymptotic —
retailer/model nodes
[90C35]
(*see:* **Minimum cost flow problem**)
retailer nodes
[90C35]
(*see:* **Minimum cost flow problem**)
retrieval *see:* Global optimization methods for harmonic —; harmonic —
retrieval based on single-crystal X-ray diffraction data *see:* Optimization techniques for phase —
return on investment *see:* maximization of —
return nodes
[90C30, 90C35]
(*see:* **Optimization in water resources**)
return and risk
[68Q25, 91B28]
(*see:* **Competitive ratio for portfolio management**)
return/risk *see:* maximization of —
revealing factorization *see:* rank —
revealing QR factorization *see:* rank —
revealing URV factorization *see:* rank —
revenue management
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
revenue management
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
reverse convex constraint *see:* linear program with an additional —
reverse convex inequality
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
Reverse convex optimization
(90C26, 90C30)
(*referred to in:* **αBB algorithm**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Standard quadratic optimization problems: theory**)
(*refers to:* **αBB algorithm**; **Concave programming**; **D.C. programming**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Standard quadratic optimization problems: theory**)
reverse convex programming
[90C05, 90C26, 90C30]
(*see:* **Continuous global optimization: models, algorithms and software**; **Reverse convex optimization**)
reverse convex programming
[90C26, 90C30]
(*see:* **Reverse convex optimization**)
reverse convex programs
[90C26, 90C30]
(*see:* **Reverse convex optimization**)
reverse convex set
[90C26, 90C30]
(*see:* **Reverse convex optimization**)
reverse differentiation
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**)
reverse mode
[65H99, 65K99]
(*see:* **Automatic differentiation: point and interval**)
reverse mode of AD
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
reverse mode of an AD algorithm
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)
reverse mode automatic differentiation
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)

reverse normal hull
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
reverse normal set
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
reverse polyblock
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
reverse polyblock algorithm
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
reverse polyblock (copolyblock) algorithm *see:* revised —
review *see:* Generalized variational inequalities: A brief —
review inventory models: (*QR*) policy *see:* Continuous —
review model *see:* continuous —; periodic —
revised geometric mean method
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
revised polyblock algorithm
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
revised reverse polyblock (copolyblock) algorithm
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
Reynolds-averaged Navier–Stokes code
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
RGM
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
RH-BFGS *see:* L- —
ρ-regular problem
[90Cxx]
(*see:* **Quasidifferentiable optimization: exact penalty methods**)
ρ-regularity
[90Cxx]
(*see:* **Quasidifferentiable optimization: exact penalty methods**)
Ribiére algorithm *see:* Polyak–Polak– —
Ribiére formula *see:* Polak– —
Ribiére method *see:* Polak– —
Riccati equation
[90C30]
(*see:* **Suboptimal control**)
Riccati equation *see:* continuous-time —
rich stream
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
Richardson iteration
[65H10, 65J15]
(*see:* **Contraction-mapping**)
ride *see:* dial-a- —; m-dial-a- —
ridge
[90Cxx]
(*see:* **Discontinuous optimization**)
ridge *see:* active —; fault —
Riemannian manifold
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
Riemannian metric
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
Riemannian metric *see:* C^k- —
Riesz theorem
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
right-chain justification
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
right-collection of a partition
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
right-hand side changes *see:* sensitivity analysis with respect to —
right-hand side perturbation model
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
right-hand side perturbation problem
[90C31]
(*see:* **Sensitivity and stability in NLP: continuity and differential stability**)
right-hand side problem
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
right-hand side simplex algorithm *see:* parametric —
right-hand-side uncertainty, duality and applications *see:* Robust linear programming with —
right-paired element
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
right-paired set
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
right-pairs
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
right-reachable
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
right-reachable *see:* directly —
right saddle point
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
right saddle-point theorem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
right-unpaired element
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
rigid templates *see:* De novo protein designUsing —
rigor *see:* with mathematical —
rigorous bound for solutions of nonlinear systems of equations
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)
rigorously
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval fixed point theory**)

ring networks
[05C85]
(*see:* **Directed tree networks**)
ring topology
[90-XX]
(*see:* **Survivable networks**)
risk *see:* business failure —; estimation —; maximization of return/ —; return and —
(risk averse, anticipative) decision *see:* ex-ante —
risk-free asset
[91B50]
(*see:* **Financial equilibrium**)
risk-free asset
[91B50]
(*see:* **Financial equilibrium**)
(risk prone, adaptive) decision *see:* ex-post —
riskless interest rate
[68Q25, 91B28]
(*see:* **Competitive ratio for portfolio management**)
risks *see:* agricultural —
Ritz bound *see:* Rayleigh– —
Ritz–Galerkin method
[65M60]
(*see:* **Variational inequalities: F. E. approach**)
Ritz–Galerkin method
[65M60]
(*see:* **Variational inequalities: F. E. approach**)
Ritz method *see:* Rayleigh– —
river hydropower nodes *see:* on-the- —
river plant *see:* run-of- —
river plants *see:* run-of- —
RLT system *see:* reduced —
RMP
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
rmsd threshold *see:* determination of —
rmsds by energy *see:* average —
(RNN) *see:* repeated nearest neighbor —
road traveling salesman problem
[90B20]
(*see:* **General routing problem**)
Robbins–Monro method
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
robin balancing scheme *see:* asynchronous round —
robin request *see:* global round —
Robinson *see:* anti- —
Robinson CQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
Robinson matrix *see:* anti- —
robust
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
robust *see:* model —; solution —
robust algorithms
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
robust conic optimization problems *see:* Approximations to —
Robust control
(93D09)
(*referred to in:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
Robust control: schur stability of polytopes of polynomials
(93D09, 93C55, 39A11)
(*referred to in:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Robust control**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Robust control**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
robust control synthesis
[93D09]
(*see:* **Robust control**)

robust control theory
[93D09]
(*see:* **Robust control**)
robust counterpart
(*see:* **Approximations to robust conic optimization problems**; **Design of robust model-based controllers via parametric programming**; **Price of robustness for linear optimization problems**)
Robust design of dynamic systems by constructive nonlinear dynamics
(37N40, 90C30, 90C34)
robust estimator
[65D10, 65K05]
(*see:* **Overdetermined systems of linear equations**)
Robust global optimization
(90C26, 90C31)
Robust linear programming with right-hand-side uncertainty, duality and applications
robust model-based controllers via parametric programming *see:* Design of —
robust obstacle-free shape design
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
robust obstacle-free truss design
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
Robust optimization
(90C90, 91B28)
(*referred to in:* **Competitive ratio for portfolio management**; **Financial applications of multicriteria analysis**; **Financial optimization**; **Portfolio selection and multicriteria analysis**; **Semi-infinite programming and applications in finance**)
(*refers to:* **Competitive ratio for portfolio management**; **Financial applications of multicriteria analysis**; **Financial optimization**; **Portfolio selection and multicriteria analysis**; **Semi-infinite programming and applications in finance**)
Robust optimization
[90C90, 91B28]
(*see:* **Robust optimization**)
Robust optimization: mixed-integer linear programs
(90C11, 90C30, 90B35)
robust parametric programs
[90C90, 91B28]
(*see:* **Robust optimization**)
robust programming problem
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
robust stability
[39A11, 93C55, 93D09]
(*see:* **Robust control: schur stability of polytopes of polynomials**)
robust stability analysis
[90C26, 90C90]
(*see:* **Global optimization in generalized geometric programming**)
robust stability analysis
[90C26, 90C90]
(*see:* **Global optimization in generalized geometric programming**)
robust stopping criteria *see:* Dykstra's algorithm and —
robustness
[90C25, 90C27, 90C90, 93D09]
(*see:* **Robust control**; **Semidefinite programming and structural optimization**)
robustness
[90C15, 90C29, 90C30, 90C99]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**; **SSC minimization algorithms**; **SSC minimization algorithms for nonsmooth and stochastic optimization**)
robustness analysis
[90C29, 90C70, 93D09]
(*see:* **Fuzzy multi-objective linear programming**; **Robust control**)
robustness for linear optimization problems *see:* Price of —
Rockafellar directional differential
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
Rockafellar duality *see:* Fenchel– —
Rockafellar duality theory *see:* Fenchel– —
Rockafellar generalized derivative *see:* Clarke– —
Rockafellar subdifferential *see:* Moreau– —
Rodríguez method
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
rolling horizon *see:* decision making with —
rollout method
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
(ROM) *see:* recursive Opt Matching —
root arc
[90C35]
(*see:* **Generalized networks**)
root-free Givens transformation *see:* square- —
root method *see:* square- —
root node
[34E05, 90C27]
(*see:* **Asymptotic properties of random multidimensional assignment problem**)
root problem
[65K05, 90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Automatic differentiation: root problem and branch problem**; **Traveling salesman problem**)
root problem
[65K05]
(*see:* **Automatic differentiation: root problem and branch problem**)
root problem and branch problem *see:* Automatic differentiation: —
root transformation *see:* logarithmic and square- —; modified square- —; square- —
root of a tree
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
rooted tree
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Generalized networks**; **Replicator dynamics in combinatorial optimization**)

rooted tree *see:* level of a vertex in a —
Rosen gradient projection method
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
Rosen gradient projection method
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
Rosen method
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
Rosen method *see:* global convergence problem for the —
Rosen's method, global convergence, and Powell's conjecture
(90C30)
(*referred to in:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **Frank–Wolfe algorithm**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**; **Successive quadratic programming: full space methods**)
Rosen mixed integer formulation *see:* LCP: Pardalos– —
Rosenbloom theorem
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
Rosenbrock hillclimbing procedure
[90C30]
(*see:* **Suboptimal control**)
Rosenbrock method
(90C30)
(*referred to in:* **Cyclic coordinate method**; **Powell method**; **Sequential simplex method**)
(*refers to:* **Cyclic coordinate method**; **Powell method**; **Sequential simplex method**)
Rosenbrock method
[90C30]
(*see:* **Rosenbrock method**)
rostering *see:* crew —
rotamer
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
rotamer
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
rotamer library
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
rotamer library
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
rotation *see:* diagnostic —; Givens —
rotation matrix
[46N10, 47N10, 49M37, 65K10, 90C26, 90C30]
(*see:* Global optimization: tight convex underestimators)
rotation in the solution of equations
[65K05, 90Cxx]
(*see:* **Symmetric systems of linear equations**)
rotation-symmetry model
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
rotations *see:* diagnostic —
roughness
[68Q20]
(*see:* **Optimal triangulations**)
roulette wheel procedure
[92B05]
(*see:* **Genetic algorithms**)
roulette wheel procedure
[92B05]
(*see:* **Genetic algorithms**)
round composition of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
round-off error
[90C60]
(*see:* **Complexity of degeneracy**)
round robin balancing scheme *see:* asynchronous —
round robin request *see:* global —
rounding *see:* consistent —; integer —; outward —; randomized —
rounding cut *see:* mixed integer —
rounding error estimation *see:* Automatic differentiation: introduction, history and —
rounding errors are under control
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
rounding function
[90C10, 90C46]
(*see:* **Integer programming duality**)
rounding heuristic
[90C35]
(*see:* **Multi-index transportation problems**)
rounding problem *see:* matrix —
ROUNDING PROCEDURE *see:* rANDOMIZED —
route choice adjustment process *see:* trip- —
routine *see:* individual software —; separation —
routines *see:* package of basic software —
routing
[90B80, 90B85, 90C35]
(*see:* **Multicommodity flow problems**; **Warehouse location problem**)
routing
[90B20]
(*see:* **General routing problem**)
routing *see:* aircraft —; arc —; inventory —; location- —; node —; period —; vehicle —; VLSI —
routing algorithm *see:* parallel —
routing and industrial in-plant railroads *see:* engine —
routing models *see:* location- —
routing pattern model *see:* fractional —; single path —

routing problem *see:* airline maintenance —; capacitated arc —; capacitated vehicle —; distance-constrained vehicle —; dynamic vehicle —; General —; inventory ship —; Location —; Metaheuristic algorithms for the vehicle —; stochastic vehicle —; vehicle —
routing problem with backhauls *see:* vehicle —
routing problem with simultaneous pickups and deliveries *see:* Vehicle —
routing problem with time windows *see:* vehicle —
routing problems *see:* approximate methods for solving vehicle —; cargo —; constructive methods for solving vehicle —; exact methods for solving vehicle —; Maritime inventory —; Stochastic vehicle —
routing and protection problems in optical networks *see:* Integer linear programs for —
routing of traffic in transmission network
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
row *see:* cost —; critical —
row-action algorithm
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
row-action method
[90C05, 90C25]
(*see:* **Young programming**)
row-action method
[90C05, 90C25]
(*see:* **Young programming**)
row bandwidth
[65Fxx]
(*see:* **Least squares problems**)
row conditional proximity data
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
row generation
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
row orthogonalization scheme *see:* sequential —
row rank *see:* full —
row sufficient
[90C33]
(*see:* **Linear complementarity problem**)
row sufficient matrix
[65K05, 90C20, 90C25, 90C33, 90C55]
(*see:* **Principal pivoting methods for linear complementarity problems**; **Splitting method for linear complementarity problems**)
RP
[90C30]
(*see:* **Lagrangian duality: BASICS**)
RPP
[90B20]
(*see:* **General routing problem**)
rRO
[74A40, 90C26]
(*see:* **Shape selective zeolite separation and catalysis: optimization methods**)
RSD
[90C30]
(*see:* **Simplicial decomposition**)
RSM-distribution with algebraically decreasing tail
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
RSM-distributions *see:* asymptotic results for —
rSQP
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
rSQP
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
RTPC
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
Rudolph method
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
rule *see:* absolute qualification —; adaptive subdivision —; Armijo —; Armijo steplength —; Bayes optimal —; Bayesian stopping —; best bound —; Bland —; Bland least index pivoting —; chain —; column dropping —; cyclic —; Dantzig —; Dantzig largest coefficient pivoting —; decision —; divergent series —; divergent series step-size —; divergent series steplength —; first-in last-out —; Fritz John —; fuzzy sum —; generalized Gauss quadrature —; generic pivoting —; geometric series —; global Lagrange multiplier —; Lagrange multiplier —; largest coefficient —; Levenberg–Marquardt —; lexicographic pivoting —; logarithmic scoring —; minimax decision —; momentum updating —; North–West corner —; parametric —; polyak II —; quadratic (Brier) scoring —; relaxation —; smallest index —; splitting —; stopping —; sum —; tie breaking —
rule-based system
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
rule for Bayesian networks *see:* chain —
rule of Bland
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
rule of economic growth *see:* Ramsey —
rule of greatest improvement
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
rule of justice
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
rule of random choice
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
rule of steepest ascent
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
rule of thumb
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)

rules *see:* anticycling —; chain —; Criss-cross pivoting —; Least-index anticycling —; Lexicographic pivoting —; look-ahead —; pivot —; pivoting —; plausible —; Stochastic global optimization: stopping —
rules of branch and bound *see:* basic —
rules of a Turing machine *see:* transition —
run *see:* artificial centering hit and —; hit and —; hyperspheres direction hit and —; improving hit and —
run algorithm *see:* hit and —
run algorithms *see:* hit and —
run generator *see:* hit and —
run methods *see:* Global optimization: hit and —; hit and —
run-of-river plant
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
run-of-river plant
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
run-of-river plants
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
run SQG *see:* single —
running list
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
running in $\mathcal{O}(n^c)$ time *see:* algorithm —
running time of a Turing machine
[90C60]
(*see:* **Complexity theory**)
runs *see:* minimal —; multiple —
rural postman problem
[90B20]
(*see:* **General routing problem**)
(RVI) *see:* random vertex insertion —
rVNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
RVP
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)

S

$S_{2\times2\times2}$ group *see:* symmetric —
S-DIAL
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
S-face
[90C20]
(*see:* **Standard quadratic optimization problems: applications**)
S-Fejérian
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
S-HEAP
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
S)policy *see:* (s —
S-procedure
[93D09]
(*see:* **Robust control**)
(s,S) optimal policies
[49L20]
(*see:* **Dynamic programming: inventory control**)
(s,S) policy
[49L20]
(*see:* **Dynamic programming: inventory control**)
(*s*, *S*)policy
[90B50]
(*see:* **Inventory management in supply chains**)
s-stress
[65K05, 90C27, 90C30, 90C57, 91C15]
(*see:* **Optimization-based visualization**)
s—t-cut
[90C35]
(*see:* **Maximum flow problem**)
s—t-cut *see:* flow across an —
S^*-matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
SA
(90C90, 90C27)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Broadcast scheduling problem**; **Discrete stochastic optimization**; **Genetic algorithms**; **Global optimization based on statistical models**; **Global optimization: hit and run methods**; **Global optimization in Lennard–Jones and morse clusters**; **Job-shop scheduling problem**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Optimization-based visualization**; **Optimization in medical imaging**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Random search methods**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Evolutionary algorithms in combinatorial optimization**; **Global optimization based on statistical models**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
SA
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
SA *see:* PA of —
SABB algorithm *see:* nonsmooth SSC- —; SSC- —

saccharomyces cerevisiae
[92B05]
(*see:* **Genetic algorithms**)
saddle duality theorem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
saddle function
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
saddle Lagrange duality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**; **Duality theory: monoduality in convex optimization**)
saddle Lagrangian
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
saddle-minimax point
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
saddle-minimax theorem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
saddle point
[49K35, 49M27, 65K10, 90C05, 90C06, 90C15, 90C25]
(*see:* **Convex max-functions**; **Probabilistic constrained linear programming: duality theory**; **Saddle point theory and optimality conditions**)
saddle point
[90C06]
(*see:* **Saddle point theory and optimality conditions**)
saddle point *see:* left —; right —
saddle-point formulation
[65M60]
(*see:* **Variational inequalities: F. E. approach**)
saddle-point inequalities
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
saddle-point problem
[49K35, 49M27, 65K10, 90C25]
(*see:* **Convex max-functions**)
saddle-point sufficient condition
[90C30]
(*see:* **Image space approach to optimization**)
saddle-point theorem *see:* right —
Saddle point theory and optimality conditions
(90C06)
(*referred to in:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Rosen's method, global convergence, and Powell's conjecture**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Rosen's method, global convergence, and Powell's conjecture**; **Second order constraint qualifications**; **Second order optimality conditions for nonlinear optimization**)
saddle points
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)
saddle value
[90C05, 90C15]
(*see:* **Probabilistic constrained linear programming: duality theory**)
safe starting region
[65G20, 65G30, 65G40]
(*see:* **Interval analysis: systems of nonlinear equations**)
safeguarded new trial steplength *see:* compute a —
Saito divergence *see:* Itakura– —
sales *see:* maximization of —
sales assumption *see:* lost —
salesman *see:* probabilistic traveling —
salesman problem *see:* classical traveling —; graphical traveling —; Heuristic and metaheuristic algorithms for the traveling —; prize collecting traveling —; road traveling —; Steiner graphical traveling —; Time-dependent traveling —; traveling —
Salesman Problem (ATSP) *see:* asymmetric Traveling —
salesman problems *see:* traveling —
salesperson problem *see:* traveling —
same face *see:* points on the —
same probability *see:* randomly with the —
sample *see:* validation —
sample and expectation functions
[90C15]
(*see:* **Stochastic quasigradient methods: applications**)
sample-path optimization
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
sample problem
[90C26, 90C29]
(*see:* **Optimal design of composite structures**)
sampler *see:* hidden Markov model and Gibbs —
samples *see:* training —
sampling
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
sampling *see:* exact —; importance —; Markov chain —; random —; uniform random —
sampling procedure *see:* interactive —
sampling and variance reduction *see:* Monte-Carlo —
sandwich condition
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
sandwich theorem
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
SAR
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
Sard theorem *see:* parametrized —
SAT
[05-04, 68Q25, 90C27, 90C60]

(*see:* **Evolutionary algorithms in combinatorial optimization**; **NP-complete problems and proof methodology**)
SAT
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
SAT *see:* 3- —; MAX- —; MAX-2- —
SAT-CNF problem
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
SAT-k-CNF
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
SAT problem *see:* weighted MAX- —
satellite orbit
[26A24, 65K99, 85-08]
(*see:* **Automatic differentiation: geometry of satellites and tracking stations**)
satellite problem
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
satellite systems
[26A24, 65K99, 85-08]
(*see:* **Automatic differentiation: geometry of satellites and tracking stations**)
satellites and tracking stations *see:* Automatic differentiation: geometry of —
satisfaction *see:* constraint —; maximum constraint —
Satisfaction Problem *see:* binary constraint —; constraint —; max-r-Constraint —; maximum constraint —; numerical constraint —
satisfaction problems *see:* constraint —; continuous constraint —
satisfaction: relaxations and upper bounds *see:* Maximum constraint —
satisfaction set
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
satisfaction techniques *see:* constraint —
satisfiability
[68Q25, 90C05, 90C10, 90C60]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**; **NP-complete problems and proof methodology**; **Simplicial pivoting algorithms for integer programming**)
satisfiability
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
satisfiability *see:* 3- —; Boolean —; maximum —
satisfiability of Boolean formulas
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
satisfiability problem
[03B50, 68T15, 68T30, 90C60]
(*see:* **Complexity theory**; **Finite complete systems of many-valued logic algebras**)
satisfiability problem
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
satisfiability problem *see:* Maximum —
satisfiable Boolean formula
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
satisficing method
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
saturating push
[90C35]
(*see:* **Maximum flow problem**)
Savings *see:* expected —
savings algorithm *see:* parallel —
savings heuristic
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
savings heuristic *see:* Whitney —
savings procedures
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
Savitch theorem
[90C60]
(*see:* **Complexity classes in optimization**)
sawtooth arc cost function
[90B10]
(*see:* **Piecewise linear network flow problems**)
SBB algorithm *see:* sSC- —
SBP
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
SC
[90B05, 90B06]
(*see:* **Global supply chain models**)
SC *see:* IM in —
scalar variational inequalities
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
scalarization
[90C29]
(*see:* **Vector optimization**)
scalarization
[90C29]
(*see:* **Vector optimization**)
scalarization property
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
scalarizing function
[90C29]
(*see:* **Multiple objective programming support**)
scalarizing function
[90C29]
(*see:* **Multiple objective programming support**)
scalarizing program *see:* achievement —
scalars
[14R10, 15A03, 51N20]
(*see:* **Linear space**)
scale
[90C29]

(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
scale
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
scale *see:* circular unidimensional —; economies of —; economy of —; exponential —; linear unidimensional —; unidimensional —
scale combinatorial optimization *see:* large- —
scale function
[90C26]
(*see:* **Invexity and its applications**)
scale function
[90C26]
(*see:* **Invexity and its applications**)
scale global optimization using terrain/funneling methods *see:* Multi- —
scale invariance criterion
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
scale-invariant
(*see:* **Global optimization: functional forms**)
scale least squares problems *see:* Complexity and large- —
scale linear systems *see:* large —
scale neighborhoods *see:* large- —
scale network *see:* intermediate —; macro —; micro —
scale nonlinear mixed integer programming problem *see:* large —
scale optimization *see:* large —
scale problem *see:* large —
scale and sparse semidefinite programs *see:* Solving large —
scale trust region *see:* large —
scale trust region problem *see:* large —
scale trust region problems *see:* Large —
scale unconstrained optimization *see:* Large —
scaled ABS class
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
scaled ABS class of algorithms
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
scaled subclass *see:* optimally —; orthogonally —
scales *see:* linear —; unidimensional —
scaling *see:* cost —; dual —; ϵ- —; multidimensional —; unidimensional —
scaling algorithm *see:* affine —; dual- —; primal- —; primal-dual —
scaling method *see:* distance —
scaling problem *see:* multidimensional —
scaling SQPIP methods *see:* affine —
scalings algorithm *see:* dual- —
scan
[90C35]
(*see:* **Multi-index transportation problems**)
Scan *see:* graham- —
Scarf formulation
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
Scarf formulation
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
scatter search
[05-04, 68T20, 68T99, 90C10, 90C11, 90C20, 90C27, 90C59]
(*see:* **Evolutionary algorithms in combinatorial optimization**; **Linear ordering problem**; **Metaheuristics**)
scenario
[90C15, 90C26, 90C90, 91B28]
(*see:* **Decomposition algorithms for the solution of multistage mean-variance optimization problems**; **Financial optimization**; **Global optimization in batch design under uncertainty**; **Stochastic programming: parallel factorization of structured matrices**; **Two-stage stochastic programs with recourse**)
scenario aggregation
[90C15]
(*see:* **L-shaped method for two-stage stochastic programs with recourse**)
scenario analysis
[90C15, 90C29, 90C30, 90C35]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**; **Optimization in water resources**; **Stochastic quasigradient methods in minimax problems**)
scenario analysis
[90C15, 90C29, 90C30, 90C35]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**; **Optimization in water resources**; **Stochastic quasigradient methods in minimax problems**)
scenario generation
[91B28]
(*see:* **Financial optimization**)
scenario set
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
scenario tree
[90C15, 90C30, 90C35, 90C90]
(*see:* **Decomposition algorithms for the solution of multistage mean-variance optimization problems**; **Multistage stochastic programming: barycentric approximation**; **Optimization in water resources**)
scenario trees *see:* barycentric —
scenarios *see:* what-if-when —
SCG *see:* algorithm- —
Schaible algorithm
[90C32]
(*see:* **Quadratic fractional programming: Dinkelbach method**)
Schauder degree *see:* Leray– —
Schauder fixed point theorem
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval fixed point theory**)
schedule *see:* annealing —; cooling —; flight —
schedule construction *see:* network design and —
schedule first-cluster second
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
schedule second strategy *see:* cluster first- —

scheduling
[90C26]
(*see:* **Global optimization in batch design under uncertainty**)
scheduling
[90B35, 90C11, 90C26, 90C90]
(*see:* **Job-shop scheduling problem**; **MINLP: design and scheduling of batch processes**; **MINLP: trim-loss problem**)
scheduling *see:* airline crew —; batch —; crew —; facility planning and —; Integrated planning and —; Locomotive —; Mixed integer optimization in well —; Optimal sensor —; Railroad crew —; Railroad locomotive —; Static resource constrained project —; Stochastic —; Vehicle —
scheduling of batch processes *see:* Medium-term —; MINLP: design and —; Reactive —
scheduling of batch processes with resources *see:* Short-term —
scheduling of continuous processes *see:* Short-term —
scheduling functions
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
scheduling: an MILP model *see:* Gasoline blending and distribution —
scheduling models *see:* single locomotive —
scheduling policy
[90B36, 90C26]
(*see:* **MINLP: design and scheduling of batch processes**; **Stochastic scheduling**)
scheduling problem
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
scheduling problem *see:* Broadcast —; crew- —; Flow shop —; Job-shop —; minimum Multiprocessor —; Multi-depot vehicle —; Single-depot vehicle —; vehicle —
scheduling problems *see:* Integrated vehicle and duty —; multi-depot vehicle —; Single-depot vehicle —
scheduling problems with a fixed number of vehicles *see:* Vehicle —
scheduling problems with multiple types of vehicles *see:* Vehicle —
scheduling problems with time constraints *see:* vehicle —
scheduling process *see:* duty —
scheduling, resource constrained: unified modeling frameworks *see:* Short-term —
scheduling (staff planning)
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
scheduling of switching engines
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
scheduling theory
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
scheduling with trip shifting *see:* Vehicle —
scheduling under uncertainty: sensitivity analysis *see:* Short-term —
schema
[92B05]
(*see:* **Genetic algorithms**)
schema
[92B05]
(*see:* **Genetic algorithms**)
Schema theorem
[92B05]
(*see:* **Genetic algorithms**)
Schema theorem
[92B05]
(*see:* **Genetic algorithms**)
Scheme
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
scheme *see:* asynchronous round robin balancing —; branch and bound —; chaotic iterative —; fully polynomial time approximation —; growth —; Hilbert —; iterative —; Kantorovich —; master-slave —; MS —; near-neighbor load balancing —; polynomial time approximation —; random polling —; randomized allocation —; sequential row orthogonalization —
schemes *see:* aggregation —
Schmidt algorithm *see:* Daniel–Gragg–Kaufmann–Stewart reorthogonalized Gram– —
Schmidt orthogonalization *see:* classical Gram– —; Gram– —; modified Gram– —
Schmidt type iteration *see:* Gram– —
Scholes model *see:* Black– —
Schroedinger equation
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
Schruben–Margolin method
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
Schur complement
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**)
Schur stability
[39A11, 93C55, 93D09]
(*see:* **Robust control: schur stability of polytopes of polynomials**)
schur stability of polytopes of polynomials *see:* Robust control: —
science *see:* cognitive —
scientific applications
[03B50, 03B52, 03E72, 47S40, 68T15, 68T27, 68T30, 68T35, 68Uxx, 90Bxx, 90C05, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Continuous global optimization: applications**; **Finite complete systems of many-valued logic algebras**)
Scientific Discovery *see:* logic of —
sCM
[68T20, 68T99, 90-02, 90C27, 90C59]
(*see:* **Metaheuristics**; **Operations research models for supply chain management and design**)
score
[90C39]
(*see:* **Neuro-dynamic programming**)
score function
[60J05, 90C15]

(*see:* **Derivatives of markov processes and their simulation**; **Derivatives of probability measures**)
score function martingale
[60J05, 90C15]
(*see:* **Derivatives of markov processes and their simulation**)
score function method
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
scores *see:* REL chart —
scoring function
[62H30, 68T10, 90C05, 90C11, 90C39]
(*see:* **Linear programming models for classification**; **Mixed integer classification problems**; **Neuro-dynamic programming**)
scoring rule *see:* logarithmic —; quadratic (Brier) —
SCOUT
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
SCP
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
(SCP) *see:* stacker Crane Problem —
screening *see:* mammography —
SD
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
Sd-classes
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
SD problem *see:* dual —; equivalent primal —; primal —; standard —
SDP
[90C25, 90C30]
(*see:* **Solving large scale and sparse semidefinite programs**)
SDP duality
[90C30]
(*see:* **Duality for semidefinite programming**)
SDSSS
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
SDVSP
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
SE
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms**)
search
[65G20, 65G30, 65G40, 68T20]
(*see:* **Interval constraints**)
search *see:* adaptive —; Adaptive global —; allowed neighbor in tabu —; aspiration —; best-first tree —; binary —; bisection —; chained local —; conformational —; curvilinear line —; cyclic coordinate —; depth-first —; depth-first tree —; direct —; domain of —; Fibonnaci section —; fixed tabs —; formulation space —; global —; Global equilibrium —; global optimum —; golden section —; graph —; greedy randomized adaptive —; grid —; Hamming-reactive tabu —; hesitant adaptive —; Heuristic —; history of a —; inexact line —; intelligent —; iterated local —; limited discrepancy —; line —; local —; localization —; Megiddo parametric —; move in a —; nonmonotone line —; nonoblivious local —; parallel aspiration —; Parallel Best-First Tree —; Parallel Depth-First Tree —; Parallel heuristic —; pattern —; prohibited neighbor in tabu —; pure adaptive —; pure localization —; pure random —; random walk —; reactive tabu —; scatter —; stochastic local —; systematic —; tabu —; topological —; tree —
search algorithm *see:* A* —; binary —; distributed game tree —; generalized game tree —; lexicographic —; optimal state space —; recursive state space —; state space —; synchronized distributed state space —
search algorithms
[90C30]
(*see:* **Frank–Wolfe algorithm**)
search algorithms *see:* local —; random —
search with backtracking *see:* depth-first —
search configuration
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
search device *see:* local —
search direction
[90C05, 90C22, 90C25, 90C30, 90C51]
(*see:* **Interior point methods for semidefinite programming**)
search direction *see:* compute the —
search directions *see:* orthogonal —
search engine
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms**)
search engine *see:* BB —; Newton —; quasi-Newton —
search with enhanced positioning *see:* Gene clustering: A novel decomposition-based clustering approach: global optimum —
search-Hansel chains question-asking strategy *see:* binary —
search heuristic
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
search heuristics *see:* advanced —; local —
search line *see:* inexact line —
search Luus—Jaakola optimization procedure *see:* Direct —
search method *see:* adaptive random —; binary —; local —; random —; stochastic —
search methodology *see:* tabu —
search methods *see:* line —; Random —; Variable neighborhood —
search optimization *see:* direct —
search overhead factor
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
search phase in GRASP *see:* local —
search problem *see:* line —
search problems *see:* polynomial time local —
search procedure *see:* greedy randomized adaptive —
search procedures *see:* Greedy randomized adaptive —
search technique *see:* inexact line —; line —
search trajectory
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)

searcher cooperation minimization algorithms *see:* supervisor and —
searches *see:* line —; pattern —
searching
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
searching *see:* line —; Minimax game tree —
searching state space graphs
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
secant equation
[49M07, 49M10, 65K, 90C06, 90C20]
(*see:* **Spectral projected gradient methods**)
secant relation
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
secant updating
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
second *see:* schedule first-cluster —
second generation modeling languages
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
second hypodifferential
[65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**)
second level problem
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
second order adjoints
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
second order approximation
[90C30]
(*see:* **Convex-simplex algorithm**)
second order codifferential
[49J52, 65K99, 65Kxx, 70-08, 90C25, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: codifferentiable functions**)
second order cone
[90C05]
(*see:* **Linear programming: interior point methods**)
second order constraint qualification
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
Second order constraint qualifications
(90C30, 49K27, 90C31, 49K40)
(*referred to in:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; Generalized semi-infinite programming: optimality conditions; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Nondifferentiable optimization: parametric programming**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order optimality conditions for nonlinear optimization**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order optimality conditions for nonlinear optimization**)
second order CQ
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
second order decomposition of a function
[90Cxx]
(*see:* **Discontinuous optimization**)
second order directional derivative *see:* generalized —
second order growth
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
second order hyperdifferential
[49J52, 65K99, 70-08, 90C25]
(*see:* **Quasidifferentiable optimization: codifferentiable functions**)
second order hypodifferential
[49J52, 65K99, 70-08, 90C25]
(*see:* **Quasidifferentiable optimization: codifferentiable functions**)
second order Lagrangian theory of CNSO problems
[46A20, 52A01, 90C30]
(*see:* **Composite nonsmooth optimization**)
second order necessary condition
[49M29, 65K10, 90C06, 90C31]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**; **Sensitivity and stability in NLP: approximation**)
second order necessary conditions
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
second order necessary and sufficient optimality conditions
[90C31, 90C34]
(*see:* **Semi-infinite programming: second order optimality conditions**)
second order optimality condition
[49M37, 65K05, 90C30]
(*see:* **Inequality-constrained nonlinear optimization**)
second order optimality conditions *see:* first order and —; Semi-infinite programming: —
Second order optimality conditions for nonlinear optimization
(90C39, 90C26)
(*referred to in:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order**

constraint qualifications; **Inequality-constrained nonlinear optimization**; **Kuhn–Tucker optimality conditions**; **Lagrangian duality: BASICS**; **Rosen's method, global convergence, and Powell's conjecture**; **Saddle point theory and optimality conditions**; **Second order constraint qualifications**)
second order procedures
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
second order regular set
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
second order sufficiency
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
second order sufficiency *see:* strong —
second order sufficient condition
[90C30, 90C31, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sensitivity and stability in NLP: continuity and differential stability**)
second order sufficient condition *see:* general —; general strong —; strong —
second order sufficient conditions
[90C26, 90C31, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**; **Sensitivity and stability in NLP**)
second order tangent set
[49K27, 49K40, 90C30, 90C31]
(*see:* **Second order constraint qualifications**)
second partial derivatives *see:* matrix of —
second principle *see:* Wardrop —
second slope lemma
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
second-stage
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
second-stage decision
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)
second-stage decisions
[90C10, 90C15]
(*see:* **Stochastic integer programs**; **Stochastic programming: parallel factorization of structured matrices**)
second-stage feasibility set
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)
second strategy *see:* cluster first-schedule —
secondary cone
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
secondary fan
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
secondary polytope
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
secondary ray *see:* termination on a —
secondary structure
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
secondary structure
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
section method *see:* golden —
section search *see:* Fibonnaci —; golden —
sectional shapes *see:* beam cross- —
sectioning
[90C30]
(*see:* **Nonlinear least squares problems**)
sector multi-instrument financial equilibrium model *see:* multi- —
sector stability criterion
[93D09]
(*see:* **Robust control**)
sectorization *see:* cell —
security market line
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
see *see:* wait-and- —
seed matrix
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
seed part
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
seed structure
[90C26, 90C90]
(*see:* **Global optimization in Lennard–Jones and morse clusters**)
seek algorithm *see:* hide-and- —
segment of a polyhedron
[90C60]
(*see:* **Complexity of degeneracy**)
segmentation
[90C90]
(*see:* **Optimization in medical imaging**)
segmentation *see:* feature —; spatial —
segmentation problem *see:* beam —
segments of polyhedra
[90C60]
(*see:* **Complexity of degeneracy**)
Seidel *see:* Gauss– —
Seidel algorithm *see:* Gauss– —
Seidel iteration *see:* Gauss– —
Seidel method *see:* Gauss– —
Seidel value iteration *see:* Gauss– —
seismic Vessel Problem (SVP)
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
selected residues
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
selection
[41A30, 47A99, 65K10, 92B05]
(*see:* **Genetic algorithms**; **Lipschitzian operators in best approximation by bounded or continuous functions**)

selection
[92B05]
(*see:* **Genetic algorithms**)
selection *see:* beam angle —; continuous —; controlled —; feature —; fundamental theorem of natural —; immediate —; lexicographic pivot —; portfolio —; priorities —; subinterval —; subset —
selection of architecture
[90C39]
(*see:* **Neuro-dynamic programming**)
selection equations
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
selection of functions *see:* continuous —
selection: markowitz mean-variance model *see:* Portfolio —
Selection of maximally informative genes
selection and multicriteria analysis *see:* Portfolio —
selection operator
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
selection operator *see:* continuous —; lipschitzian —; optimal Lipschitzian —
selection problem
[90C29]
(*see:* **Multiple objective programming support**)
selection problem *see:* portfolio —
selection in radiotherapy treatment design *see:* Beam —
selection step
[90C59]
(*see:* **Heuristic and metaheuristic algorithms for the traveling salesman problem**)
selection and wedge orientation optimization *see:* beam angle —
selective zeolite separation and catalysis: optimization methods *see:* Shape —
self-inverse product of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
self-organization
[68M20, 90B35]
(*see:* **Flow shop scheduling problem**)
selfadjoint operator *see:* closed —
selfdual
[90C05]
(*see:* **Homogeneous selfdual methods for linear programming**)
selfdual methods for linear programming *see:* Homogeneous —
selfdual model *see:* homogeneous and —
Selfdual parametric method for linear programs
(90C05, 90C06)
(*referred to in:* **Bounds and solution vector estimates for parametric NLPS**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**)
(*refers to:* **Bounds and solution vector estimates for parametric NLPS**; **Multiparametric linear programming**; **Multiparametric mixed integer linear programming**; **Nondifferentiable optimization: parametric programming**; **Parametric global optimization: sensitivity**; **Parametric linear programming: cost simplex algorithm**; **Parametric mixed integer nonlinear optimization**; **Parametric optimization: embeddings, path following and singularities**)
selfdual problem
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
selfdual rank one formula
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
selfdual system
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
selling *see:* short —
selling problem *see:* asset —
semantic analysis methodologies
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
semantics
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
semantics for fuzzy logics *see:* Checklist paradigm —
semantics of MVL connectives
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
semi-assignment constraints
[90-00]
(*see:* **Generalized assignment problem**)
semi-assignment problem *see:* Quadratic —
(semi) definite completion problem *see:* positive —
semi-definite quadratic binary programming *see:* positive —
semi-infinite
[03H10, 49J27, 57R12, 90C31, 90C34]
(*see:* **Semi-infinite programming and control problems**; **Smoothing methods for semi-infinite optimization**)
semi-infinite linear programming
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
semi-infinite optimization
[65K05, 65K10, 90C06, 90C30, 90C34]
(*see:* **Feasible sequential quadratic programming**)
semi-infinite optimization
[90C25, 90C26, 90C31, 90C34]
(*see:* **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**)
semi-infinite optimization *see:* Adaptive convexification in —; one-parametric —; Smoothing methods for —
semi-infinite optimization problem
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
semi-infinite optimization problems
[90C31, 90C34]
(*see:* **Semi-infinite programming: second order optimality conditions**)

semi-infinite problem *see:* generalized —
semi-infinite program
[90C34]
(*see:* **Semi-infinite programming: approximation methods**)
semi-infinite program *see:* dual —; primal (linear) —
semi-infinite programming
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
semi-infinite programming
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**; **Semi-infinite programming, semidefinite programming and perfect duality**)
semi-infinite programming *see:* linear —; perfect duality from the view of linear —; reduced problem in —
Semi-infinite programming and applications in finance
(90C34, 91B28)
(*referred to in:* **Competitive ratio for portfolio management**; **Financial applications of multicriteria analysis**; **Financial optimization**; **Portfolio selection and multicriteria analysis**; **Robust optimization**)
(*refers to:* **Competitive ratio for portfolio management**; **Financial applications of multicriteria analysis**; **Financial optimization**; **Portfolio selection and multicriteria analysis**; **Robust optimization**; **Semi-infinite programming: approximation methods**; **Semi-infinite programming and control problems**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**; **Semi-infinite programming, semidefinite programming and perfect duality**)
Semi-infinite programming: approximation methods
(90C34)
(*referred to in:* **Semi-infinite programming and applications in finance**; **Semi-infinite programming and control problems**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**; **Semi-infinite programming, semidefinite programming and perfect duality**)
(*refers to:* **Semi-infinite programming and control problems**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**; **Semi-infinite programming, semidefinite programming and perfect duality**)
Semi-infinite programming and control problems
(49J27, 90C34, 03H10)
(*referred to in:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and applications in finance**; **Semi-infinite programming: approximation methods**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
(*refers to:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming: approximation methods**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Suboptimal control**)
Semi-infinite programming: discretization methods
(90C34, 90C05, 90C25, 90C30)
(*referred to in:* **Semi-infinite programming and applications in finance**; **Semi-infinite programming: approximation methods**; **Semi-infinite programming and control problems**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Semi-infinite programming: approximation methods**; **Semi-infinite programming and control problems**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**; **Semi-infinite programming, semidefinite programming and perfect duality**)
Semi-infinite programming: methods for linear problems
(90C34, 90C05)
(*referred to in:* **Semi-infinite programming and applications in finance**; **Semi-infinite programming: approximation methods**; **Semi-infinite programming and control problems**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**; **Semi-infinite programming, semidefinite programming and perfect duality**)
(*refers to:* **Semi-infinite programming: approximation methods**; **Semi-infinite programming and control**

problems; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**; **Semi-infinite programming, semidefinite programming and perfect duality**)

Semi-infinite programming: numerical methods
(90C34, 90C26, 90C25, 90C34)
(*referred to in:* **Semi-infinite programming and applications in finance**; **Semi-infinite programming: approximation methods**; **Semi-infinite programming and control problems**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: second order optimality conditions**; **Semi-infinite programming, semidefinite programming and perfect duality**)
(*refers to:* **Semi-infinite programming: approximation methods**; **Semi-infinite programming and control problems**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: second order optimality conditions**; **Semi-infinite programming, semidefinite programming and perfect duality**)

semi-infinite programming: optimality conditions *see:* Generalized —

Semi-infinite programming: second order optimality conditions
(90C31, 90C34)
(*referred to in:* **Semi-infinite programming and applications in finance**; **Semi-infinite programming: approximation methods**; **Semi-infinite programming and control problems**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming, semidefinite programming and perfect duality**)
(*refers to:* **Semi-infinite programming: approximation methods**; **Semi-infinite programming and control problems**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming, semidefinite programming and perfect duality**)

Semi-infinite programming, semidefinite programming and perfect duality
(90C05, 90C25, 90C30, 90C34)
(*referred to in:* **Duality for semidefinite programming**; **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and structural optimization**; **Semi-infinite programming and applications in finance**; **Semi-infinite programming: approximation methods**; **Semi-infinite programming and control problems**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**; **Smoothing methods for semi-infinite optimization**; **Solving large scale and sparse semidefinite programs**)
(*refers to:* **Duality for semidefinite programming**; **Interior point methods for semidefinite programming**; **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and structural optimization**; **Semi-infinite programming: approximation methods**; **Semi-infinite programming and control problems**; **Semi-infinite programming: discretization methods**; **Semi-infinite programming: methods for linear problems**; **Semi-infinite programming: numerical methods**; **Semi-infinite programming: second order optimality conditions**; **Solving large scale and sparse semidefinite programs**)

semi-infinite programs
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)

semi-infinite programs *see:* computationally equivalent —; nonlinear —

semi-order
[90C29]
(*see:* **Preference modeling**)

semi-ordered spaces
[01A99]
(*see:* **Kantorovich, Leonid Vitalyevich**)

semicoercive function
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)

semicoercive hemivariational inequality
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)

semicoercive hemivariational inequality
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)

semicontinuity
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)

semicontinuous *see:* lower —

semicontinuous function *see:* lower —; $\mathbf{R}^{\eta}_{+}$-upper —; upper —

semidefinite
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)

semidefinite *see:* positive —

semidefinite matrices *see:* positive —

semidefinite matrix *see:* bisymmetric positive —; partial —; positive —

semidefinite matrix completion problem *see:* positive —

semidefinite program
[90C25, 90C27, 90C30, 90C90]
(*see:* **Duality for semidefinite programming**; **Semidefinite programming and structural optimization**; **Solving large scale and sparse semidefinite programs**)

semidefinite program *see:* dual —; linear —

semidefinite program as conic convex program
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)

semidefinite programming
[46A20, 52A01, 90C05, 90C22, 90C25, 90C30, 90C51, 93D09]
(*see:* **Copositive programming**; **Farkas lemma:**

generalizations; **Interior point methods for semidefinite programming**; **Robust control**)

semidefinite programming
[15A15, 46A20, 52A01, 90C22, 90C25, 90C27, 90C30, 90C31, 90C55, 90C90, 93D09]
(*see:* **Duality for semidefinite programming**; **Farkas lemma: generalizations**; **Large scale trust region problems**; **Robust control**; **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and structural optimization**; **Solving large scale and sparse semidefinite programs**)

semidefinite programming *see:* Duality for —; Graph realization via —; handbook on —; Interior point methods for —; Maximum likelihood detection via —

semidefinite programming approach
[90C30]
(*see:* **Large scale trust region problems**)

Semidefinite programming and determinant maximization
(90C25, 90C55, 90C25, 90C90, 15A15)
(*referred to in:* **αBB algorithm**; **Duality for semidefinite programming**; **Eigenvalue enclosures for ordinary differential equations**; **Hemivariational inequalities: eigenvalue problems**; **Interval analysis: eigenvalue bounds of interval matrices**; **Matrix completion problems**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and the sensor network localization problem, SNLP**; **Semidefinite programming and structural optimization**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Solving large scale and sparse semidefinite programs**)
(*refers to:* **αBB algorithm**; **Duality for semidefinite programming**; **Eigenvalue enclosures for ordinary differential equations**; **Hemivariational inequalities: eigenvalue problems**; **Interior point methods for semidefinite programming**; **Interval analysis: eigenvalue bounds of interval matrices**; **Matrix completion problems**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and structural optimization**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Solving large scale and sparse semidefinite programs**)

Semidefinite programming: optimality conditions and stability
(90C22, 90C25, 90C31)
(*referred to in:* **Duality for semidefinite programming**; **Maximum cut problem, MAX-CUT**; **Semidefinite programming and determinant maximization**; **Semidefinite programming and the sensor network localization problem, SNLP**; **Semidefinite programming and structural optimization**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Solving large scale and sparse semidefinite programs**; **Standard quadratic optimization problems: algorithms**)
(*refers to:* **Duality for semidefinite programming**; **Interior point methods for semidefinite programming**; **Semidefinite programming and determinant maximization**; **Semidefinite programming and structural optimization**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Solving large scale and sparse semidefinite programs**)

semidefinite programming and perfect duality *see:* Semi-infinite programming —

semidefinite programming problem
[15A15, 90C22, 90C25, 90C31, 90C55, 90C90]
(*see:* **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**)

semidefinite programming problem *see:* convex —; linear —

Semidefinite programming and the sensor network localization problem, SNLP
(*referred to in:* **Maximum cut problem, MAX-CUT**)
(*refers to:* Graph realization via semidefinite programming; **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and structural optimization**; **Solving large scale and sparse semidefinite programs**)

Semidefinite programming and structural optimization
(90C25, 90C27, 90C90)
(*referred to in:* **Duality for semidefinite programming**; **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and the sensor network localization problem, SNLP**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Solving large scale and sparse semidefinite programs**; **Topology of global optimization**; **Topology optimization**)
(*refers to:* **Duality for semidefinite programming**; **Interior point methods for semidefinite programming**; **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Solving large scale and sparse semidefinite programs**; **Structural optimization**; **Structural optimization: history**; **Topology of global optimization**; **Topology optimization**)

semidefinite programs *see:* Solving large scale and sparse —

semidefinite relaxations *see:* bounds based on —

semidefinite symmetric matrix *see:* positive —

semidefiniteness constraints *see:* positive —

semigreedy heuristic
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)

semilattice *see:* geometric —

semilinear set
[52B12, 68Q25]
(*see:* **Fourier–Motzkin elimination method**)

semilinear set
[52B12, 68Q25]
(*see:* **Fourier–Motzkin elimination method**)

seminormal equation *see:* corrected —

semipermeability
[35R70, 47S40, 74B99, 74D99, 74G99, 74H99]
(*see:* **Quasidifferentiable optimization: applications to thermoelasticity**)

semisets
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)

semisets
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
semismooth function
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
semismooth function *see:* strongly —; upper —
semismooth mapping
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
semismooth mapping
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
semismooth mapping *see:* strongly —
semismoothness
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
semismoothness *see:* strong —
semistrictly quasiconvex function
[90C26]
(*see:* **Generalized monotone single valued maps**)
semistrictly quasimonotone map
[90C26]
(*see:* **Generalized monotone single valued maps**)
semistrictly quasimonotone operator
[90C26]
(*see:* **Generalized monotone multivalued maps**)
semisublattice *see:* meet —
sender-initiated
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)
sender initiated mapping technique
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
sense of Jongen–Jonker–Twilt *see:* problem regular in the —
sense of Kojima–Hirabayashi *see:* problem regular in the —
sensitive heuristics *see:* history- —
sensitivity
[90B85, 90C30, 93-XX]
(*see:* **Boundary condition iteration BCI**; **Dynamic programming: optimal control applications**; **Single facility location: multi-objective euclidean distance location**; **Suboptimal control**)
sensitivity
[65L99, 90C05, 90C25, 90C29, 90C30, 90C31, 93-XX]
(*see:* **Nondifferentiable optimization: parametric programming**; **Optimization strategies for dynamic systems**)
sensitivity *see:* Parametric global optimization: —; variational —
sensitivity analysis
[13Cxx, 13Pxx, 14Qxx, 90C05, 90C10, 90C25, 90C29, 90C30, 90C31, 90C46, 90Cxx]
(*see:* **Integer programming: algebraic methods**; **Integer programming duality**; **Nondifferentiable optimization: parametric programming**; **Sensitivity and stability in NLP**)
sensitivity analysis
[13Cxx, 13Pxx, 14Qxx, 65K10, 90C22, 90C25, 90C31, 90C33, 90Cxx]
(*see:* **Bounds and solution vector estimates for parametric NLPS**; **Integer programming: algebraic methods**; **Semidefinite programming: optimality conditions and stability**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**)
sensitivity analysis *see:* applications of —; automated Fortran program for nonlocal —; nonlocal —; post-optimality —; shape —; Short-term scheduling under uncertainty: —
sensitivity analysis with automatic differentiation *see:* Nonlocal —
Sensitivity analysis of complementarity problems
(90C31, 90C33)
(*referred to in:* **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric global optimization: sensitivity**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**)
(*refers to:* **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric global optimization: sensitivity**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**)
sensitivity analysis with respect to changes in cost coefficients
[90C05, 90C31]
(*see:* **Parametric linear programming: cost simplex algorithm**)
sensitivity analysis with respect to right-hand side changes
[90C05, 90C31]
(*see:* **Multiparametric linear programming**)
Sensitivity analysis of variational inequality problems
(90C31, 65K10)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric global optimization: sensitivity**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity**

analysis of complementarity problems; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric global optimization: sensitivity**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of complementarity problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
sensitivity-based gradient
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
sensitivity derivatives
[90C26, 90C90]
(*see:* **Structural optimization: history**)
sensitivity equations
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
sensitivity in nonlinear programming
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
sensitivity of optimal flowsheets
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
sensitivity parameters
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
Sensitivity and stability in NLP
(90C31)
(*referred to in:* **Ill-posed variational problems**; **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric global optimization: sensitivity**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**)
(*refers to:* **Ill-posed variational problems**; **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric global optimization: sensitivity**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP: approximation**; **Sensitivity and stability in NLP: continuity and differential stability**)
Sensitivity and stability in NLP: approximation
(90C31)
(*referred to in:* **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric global optimization: sensitivity**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: continuity and differential stability**)
(*refers to:* **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric global optimization: sensitivity**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: continuity and differential stability**)
Sensitivity and stability in NLP: continuity and differential stability
(90C31)
(*referred to in:* **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric global optimization: sensitivity**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: approximation**)
(*refers to:* **Nonlocal sensitivity analysis with automatic differentiation**; **Parametric global optimization: sensitivity**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP**; **Sensitivity and stability in NLP: approximation**)
sensitivity theorem *see:* basic —
Sensor network localization problem
(*see:* **Semidefinite programming and the sensor network localization problem, SNLP**)
sensor network localization problem, SNLP *see:* Semidefinite programming and the —
sensor scheduling *see:* Optimal —
separability assumption
[49M29, 90C11]
(*see:* **Generalized benders decomposition**)
separable classes
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)

separable convex objective function
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
separable formulation
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
separable function *see:* partially —
separable objective function
[90C25]
(*see:* **Concave programming**)
separable optimization
[90C30]
(*see:* **Generalized total least squares**)
separable optimization problem
[90C30]
(*see:* **Generalized total least squares**)
separable problem
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
separable problem variables
[90C30]
(*see:* **Nonlinear least squares problems**)
separated *see:* 2- —
separated Newton method
[90C30]
(*see:* **Generalized total least squares**)
separated Newton method
[90C30]
(*see:* **Generalized total least squares**)
separated pair decomposition *see:* well- —
separating agents *see:* mass —
separating hyperplane
[62H30, 68T10, 90C05]
(*see:* **Linear programming models for classification**)
separating relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
separation
[49K27, 58C20, 58E30, 62H30, 68R10, 90B35, 90C11, 90C27, 90C30, 90C48]
(*see:* **Assignment methods in clustering**; **Branchwidth and branch decompositions**; **Nonsmooth analysis: Fréchet subdifferentials**; **Robust optimization: mixed-integer linear programs**)
separation
[90C30, 93A30, 93B50]
(*see:* **Image space approach to optimization**; **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: mass and heat exchanger networks**)
separation *see:* k- —; topological —
separation and catalysis: optimization methods *see:* Shape selective zeolite —
separation condition
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
separation functions
[90C30]
(*see:* **Image space approach to optimization**)
separation number
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
separation problem
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
separation procedure *see:* arc —
separation processes *see:* synthesis of —
separation routine
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
separation theorem *see:* Frank discrete —; L- —; M- —
separation theorems
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
sequence
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
sequence *see:* epiconvergent —; even —; Fejér monotone —; generalized finite —; generalized minimizing —; Levitin–Polyak minimizing —; minimizing —; minimum weight common mutated —; odd —
sequence alignment *see:* multiple —
sequence alignment via mixed-integer linear optimization *see:* Global pairwise protein —
sequence of arcs
[90C35]
(*see:* **Minimum cost flow problem**)
sequence comparison
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
sequence of greedy swaps *see:* monotone —
sequencing
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**; **Mixed integer programming/constraint programming hybrid methods**)
sequencing *see:* dNA —
sequencing problem
[05-04, 90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Evolutionary algorithms in combinatorial optimization**; **Traveling salesman problem**)
sequential
[68T20, 68T99, 9008, 90C26, 90C27, 90C59]
(*see:* **Metaheuristics**; **Variable neighborhood search methods**)
sequential *see:* direct- —
sequential approximate optimization
[90C26, 90C90]
(*see:* **Structural optimization: history**)
sequential approximate optimization
[90C26, 90C90]
(*see:* **Structural optimization: history**)
sequential CA algorithm
[90C30]
(*see:* **Cost approximation algorithms**)
sequential coloring *see:* frequency exhaustive —; requirement exhaustive —; uniform —
Sequential cutting plane algorithm
(90C11, 90C26)

sequential deterministic algorithm
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
sequential estimation procedure
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)
sequential estimation procedure
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)
sequential experimental design
[90C15, 90C27]
(*see:* **Discrete stochastic optimization**)
sequential greedy coloring heuristic
[90C35]
(*see:* **Graph coloring**)
sequential greedy heuristics
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
sequential Hansel chains question-asking strategy
[90C09]
(*see:* **Inference of monotone boolean functions**)
sequential Hansel chains question-asking strategy
[90C09]
(*see:* **Inference of monotone boolean functions**)
sequential heuristics
[05-XX]
(*see:* **Frequency assignment problem**)
sequential MEN synthesis method
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
sequential method
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
sequential minimax game tree algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
sequential normal compactness
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
sequential normal compactness *see:* partial —
sequential quadratic programming
[65K05, 65K10, 90C05, 90C06, 90C25, 90C30, 90C34]
(*see:* **Cost approximation algorithms**; **Feasible sequential quadratic programming**; **Semi-infinite programming: discretization methods**)
sequential quadratic programming
[49K20, 49M99, 65K05, 65K10, 90C06, 90C30, 90C34, 90C55]
(*see:* **Cost approximation algorithms**; **Feasible sequential quadratic programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
sequential quadratic programming *see:* Feasible —
Sequential quadratic programming: interior point methods for distributed optimal control problems
(49M99, 49K20, 90C55)
(*referred to in:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Entropy optimization: interior point methods**; **Feasible sequential quadratic programming**; **Hamilton–Jacobi–Bellman equation**; **Homogeneous selfdual methods for linear programming**; **Infinite horizon control and dynamic games**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Potential reduction methods for linear programming**; **Probabilistic analysis of simplex algorithms**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Standard quadratic optimization problems: theory**; **Suboptimal control**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Entropy optimization: interior point methods**; **Feasible sequential quadratic programming**; **Hamilton–Jacobi–Bellman equation**; **Homogeneous selfdual methods for linear programming**; **Infinite horizon control and dynamic games**; **Interior point methods for semidefinite programming**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Potential reduction methods for linear programming**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Suboptimal control**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
sequential quadratic programming method *see:* piecewise —
sequential quadratic programming methods
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
sequential row orthogonalization scheme
[65Fxx]
(*see:* **Least squares problems**)

Sequential simplex method
(90C30)
(*referred to in:* **Convex-simplex algorithm**; **Cyclic coordinate method**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Order complementarity**; **Parametric linear programming: cost simplex algorithm**; **Powell method**; **Principal pivoting methods for linear complementarity problems**; **Rosenbrock method**; **Topological methods in complementarity theory**)
(*refers to:* **Convex-simplex algorithm**; **Cyclic coordinate method**; **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Parametric linear programming: cost simplex algorithm**; **Powell method**; **Rosenbrock method**)
sequential simplex method
[90C30]
(*see:* **Sequential simplex method**)
sequential synthesis
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
sequentially convexifiable program
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
seriation
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
seriation
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
series *see:* Taylor —; time —
series analysis *see:* time —
series expansion *see:* first order Taylor —
series-parallel graph
[05C50, 15A48, 15A57, 90C25]
(*see:* **Matrix completion problems**)
series rule *see:* divergent —; geometric —
series step-size rule *see:* divergent —
series steplength rule *see:* divergent —
serious step *see:* long —; short —
service needs *see:* static/dynamic —
set *see:* active —; active index —; affine —; border nodes —; Borel —; bounded level —; branch of a feasible —; Cartesian product —; Chebyshev —; common dependency —; completely regular —; connected —; connected dominating —; constraint —; convex —; convex polyhedral —; cover the extremal —; d.c. —; decision —; dependent —; dominating —; dual feasible —; edge —; efficient —; efficient point —; ϵ-subdifferential —; essentially active index —; extension —; feasible —; feedback vertex —; finite dominating —; first order tangent —; fractal —; fuzzy —; fuzzy power —; generalized critical point —; geodesic convex —; ground —; Heuristics for maximum clique and independent —; hierarchy in a finite —; high-order feasible —; independent —; independent dominating —; index —; interior of a —; invariant —; invex —; invexity with respect to a —; L-convex —; left-paired —; level —; localization —; lower —; lower bound for a —; M-convex —; max-closed —; maximal independent —; maximum Independent —; maximum weighted independent —; middle —; minimal dependent —; minimum —; minimum feedback vertex —; motion of a —; nonconvex —; nondominated solution —; noninferior solution —; obstruction —; opposite of a signed —; ordered —; outcome —; p-order feasible —; Pareto optimal solution —; partition on a —; polyhedral —; pre-invex —; pre-invexity with respect to a —; primal feasible —; production —; proximal —; quasidifferentiable —; rational reaction —; reduction of a constraint —; regenerative —; regular —; representative —; reverse convex —; reverse normal —; right-paired —; satisfaction —; scenario —; second order regular —; second order tangent —; second-stage feasibility —; semilinear —; signed —; singular —; SIP index —; slope —; solution —; species index —; stability of a solution —; stable —; star-shaped —; strictly feasible —; subdifferential —; substationarity point with respect to a —; support —; test —; training —; tree nodes —; uncertainty —; upper —; upper bound for a —; vertex —
set algorithm *see:* active —; minimum lower —
set of alternatives
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
set of the alternatives *see:* finite —
set of bases of a matroid
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
set of connectives *see:* complete —
set constraints *see:* odd- —
set-contraction *see:* k- —; strict- —
Set covering, packing and partitioning problems
(90C10, 90C11, 90C27, 90C57)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Genetic algorithms**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer**

programming duality; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Neural networks for combinatorial optimization**; **Parametric mixed integer nonlinear optimization**; **Simplicial pivoting algorithms for integer programming**; **Simulated annealing methods in protein folding**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)

set covering problem
[90C05, 90C10, 90C11, 90C20, 90C27, 90C57]
(*see:* **Integer programming**; **Redundancy in nonlinear programs**; **Set covering, packing and partitioning problems**)

set D *see:* countable —

set of decision alternative
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)

set of decrease *see:* high-order —

set-definable classes
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)

set of discrete ε-global local maximizers
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)

set of edges of a digraph
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)

set of elementary functions
[90C26]
(*see:* **Global optimization: envelope representation**)

set of ε-global points
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)

set of ε-most active points
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)

set of faults
[90Cxx]
(*see:* **Discontinuous optimization**)

set of feasible points
[90C05, 90C25, 90C30, 90C33, 90C34]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**; **Semi-infinite programming: discretization methods**)

set of feasible solutions
[90C05, 90C34]
(*see:* **Semi-infinite programming: methods for linear problems**)

set of flowlines
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)

set of formation values
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)

set-formula
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)

set of a function *see:* effective —; gradient-related —; support —

set-inclusion operator *see:* fuzzy —

set L_{free} of unused partitions
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)

set L_{reac} of used partitions
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)

set of Lagrange multipliers *see:* extended —

set of loads
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)

set mapping *see:* closed point-to- —; point-to- —

set mappings *see:* point-to- —

set method *see:* active —

set methods *see:* active —

set packing problem
[90C10, 90C11, 90C27, 90C57]
(*see:* **Set covering, packing and partitioning problems**)

set partitioning
[68M20, 90B06, 90B10, 90B35, 90B80, 90C06, 90C08, 90C10, 90C27, 90C35, 90C90]
(*see:* **Airline optimization**; **Vehicle scheduling**)

set partitioning problem
[05-04, 90C10, 90C11, 90C27, 90C57]
(*see:* **Evolutionary algorithms in combinatorial optimization**; **Integer programming**; **Set covering, packing and partitioning problems**)

set of potential efficient solutions
[90C27, 90C29]
(*see:* **Multi-objective combinatorial optimization**)

set of prices *see:* almost at equilibrium of an assignment and a —; equilibrium of an assignment and a —

Set Problem *see:* feedback —; feedback arc —; feedback vertex —; maximum Independent —; minimum feedback arc —; minimum feedback vertex (arc) —; minimum weight feedback arc —; minimum weighted feedback vertex —; subset feedback vertex (arc) —; subset minimum feedback vertex (arc) —; unweighted feedback vertex —

set problem: branch & cut algorithms *see:* Stable —

set in problem solving *see:* restriction to the solution —

set problems *see:* Feedback —

set quadratic programming methods *see:* active —

set R_{free} of unused partitions
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)

set R_{reac} of used partitions
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)

set strategies *see:* active —

set strategy *see:* active —; Goldfarb–Idnani active —

set theory *see:* Alternative —; axioms of alternative —; Cantor —

set unification
[65K05, 90C26, 90C33, 90C34]

(*see:* **Adaptive convexification in semi-infinite optimization**)
set V *see:* vertex —
set-valued analysis
[49K27, 90C29, 90C48]
(*see:* **Set-valued optimization**)
set-valued analysis
[49K27, 90C29, 90C48]
(*see:* **Set-valued optimization**)
set-valued constraints
[49K27, 90C29, 90C48]
(*see:* **Set-valued optimization**)
set-valued objective function
[49K27, 90C29, 90C48]
(*see:* **Set-valued optimization**)
Set-valued optimization
(90C48, 90C29, 49K27)
(*referred to in:* **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**)
(*refers to:* **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**)
set-valued optimization
[49K27, 90C29, 90C48]
(*see:* **Set-valued optimization**)
set-valued optimization problem
[49K27, 90C29, 90C48]
(*see:* **Set-valued optimization**)
set of wells
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
sets
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
sets
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
sets *see:* connectedness of the efficient points —; differences of convex —; fuzzy —; high-order tangent —; independent —; joined —; Lagrange multiplier —; least squares problems with massive data —; max-closed —; maximum weight independent —; minimum lower —; orthogonal signed —; pseudoconnected family of —; test —
sets axioms *see:* existence of —
sets conjugation *see:* level —
sets and functions *see:* Affine —
sets in integer programming *see:* test —
sets and interior point methods *see:* Successive quadratic programming: solution by active —
setting *see:* average case —; randomized —
setting methods *see:* label —
settle-value
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
setup cost
[90C25]
(*see:* **Concave programming**)
Several Ratios *see:* Maximization of the Smallest of —
shadow price
[90C60]
(*see:* **Complexity of degeneracy**)
shadow prices
[90C05, 90C06, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**; **Saddle point theory and optimality conditions**; **Sensitivity and stability in NLP: continuity and differential stability**)
shadow of shadows
[01A99]
(*see:* **Gauss, Carl Friedrich**)
shadow-vertex
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
shadow-vertex algorithm
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
shadow-vertices *see:* expected number of —; variance of the number of —
shadows *see:* shadow of —
Shahshahani metric
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
shake and bake algorithm
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)
shake and bake algorithm
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)
Shake and bake approach *see:* Phase problem in X-ray crystallography: —
Shanno conjugate gradient method
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
Shanno method *see:* Broyden–Fletcher–Goldfarb– —
Shanno quasi-Newton update *see:* Broyden–Fletcher–Goldfarb– —
Shanno update *see:* Broyden–Fletcher–Goldfarb– —
Shannon
[94A17]
(*see:* **Jaynes' maximum entropy principle**)
Shannon entropy
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
Shannon function
[90C09]
(*see:* **Inference of monotone boolean functions**)
Shannon function
[90C09]
(*see:* **Inference of monotone boolean functions**)
shannon measure of entropy and its properties *see:* Entropy optimization: —
shannon zero-error capacity
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)

shape
[90C26, 90C90]
(*see:* **Structural optimization: history**)
shape design
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
shape design *see:* multiload —; optimal —; robust obstacle-free —
Shape optimization
(49J20, 49J52)
(*refers to:* **Structural optimization**; **Structural optimization: history**; **Topological derivative in shape optimization**)
shape optimization
[49J20, 49J52]
(*see:* **Shape optimization**)
shape optimization
[49J20, 49J52]
(*see:* **Shape optimization**)
shape optimization *see:* structural —; Topological derivative in —
Shape reconstruction methods for nonconvex feasibility analysis
(90-08, 90C26, 90C31)
Shape selective zeolite separation and catalysis: optimization methods
(74A40, 90C26)
shape sensitivity analysis
[49J20, 49J52]
(*see:* **Shape optimization**)
shape sensitivity analysis
[49J20, 49J52]
(*see:* **Shape optimization**)
shaped decomposition *see:* L- —
shaped method
[90C06, 90C15]
(*see:* **Stabilization of cutting plane algorithms for stochastic linear programming problems**)
shaped method *see:* integer L- —; l- —
shaped method for two-stage stochastic programs with recourse *see:* L- —
shaped set *see:* star- —
shapes *see:* beam cross-sectional —; design of optimal —
share
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**)
shared-memory model *see:* queueing —
shared memory parallel machines
[65K05, 65Y05]
(*see:* **Parallel computing: models**)
Sharpe ratio
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
Sharpe single index market model
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
sheet *see:* balance —; β- —
Sheffer function
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
Sherman–Morrison formula
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
Sherman-Morrison rank-one update formula
[90C30]
(*see:* **Numerical methods for unary optimization**)
Sherman–Morrison–Woodbury formula
[90C15]
(*see:* **Stochastic programming: parallel factorization of structured matrices**)
shift function *see:* cyclic —
shift-invariant
(*see:* **Global optimization: functional forms**)
shift matrix *see:* diagonal —
shift move
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
shift terms
[90C31, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions)
shifting *see:* Vehicle scheduling with trip —
Shindo method *see:* Kojima– —
ship routing problem *see:* inventory —
shipment delivery *see:* express —
shock
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
shock
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
shooting *see:* multiple —
shop *see:* flow- —; job- —
shop problem *see:* flow- —; job- —; open —
shop scheduling problem *see:* Flow —; Job- —
Shor, Naum Zuselevich
(01A70, 90-03)
short selling
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
short selling
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
short serious step
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
short-term memory
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
Short-term scheduling of batch processes with resources
Short-term scheduling of continuous processes
(90B35, 65K05, 90C90, 90C11)
Short-term scheduling, resource constrained: unified modeling frameworks
(90B35, 65K05, 90C90, 90C11)
Short-term scheduling under uncertainty: sensitivity analysis
shortest edge
[68Q20]
(*see:* **Optimal triangulations**)

shortest path
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
shortest path
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
shortest path *see:* stochastic —
shortest path algorithm *see:* successive —
shortest path algorithms *see:* generic —
shortest path problem
[49L20, 90C30, 90C40, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**; **Dynamic programming: stochastic shortest path problems**; **Simplicial decomposition**)
shortest path problem
[90B10, 90C27, 90C30]
(*see:* **Shortest path tree algorithms**; **Simplicial decomposition**)
shortest path problem *see:* deterministic —; stochastic —
shortest path problems *see:* Dynamic programming: stochastic —; stochastic —
shortest path procedure *see:* next —
Shortest path tree algorithms
(90C27, 90B10)
(*referred to in:* **Auction algorithms**; **Bottleneck steiner tree problems**; **Capacitated minimum spanning trees**; **Communication network assignment problem**; **Complexity theory**; **Dynamic traffic networks**; **Equilibrium networks**; **Generalized networks**; **Maximum flow problem**; **Minimax game tree searching**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
(*refers to:* **Auction algorithms**; **Bottleneck steiner tree problems**; **Capacitated minimum spanning trees**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Generalized networks**; **Maximum flow problem**; **Minimax game tree searching**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
shortest path tree problem *see:* single source —
shortest path tree problems
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
shortest paths *see:* problem of finding —
shortest program length
[90C60]
(*see:* **Kolmogorov complexity**)
Shortest program length
[90C60]
(*see:* **Kolmogorov complexity**)
Shubert algorithm *see:* Piyavskii– —
shutdown
(*see:* **Reactive scheduling of batch processes**)
sibles
[90C30, 90C35]
(*see:* **Optimization in water resources**)
side *see:* proof on the dual —
side changes *see:* sensitivity analysis with respect to right-hand —
side constraints
[90B06, 90C06, 90C08, 90C35, 90C90]
(*see:* **Airline optimization**)
side constraints
[90C30]
(*see:* **Simplicial decomposition**)
side payments *see:* game with —
side perturbation model *see:* right-hand —
side perturbation problem *see:* right-hand —
side problem *see:* right-hand —
side simplex algorithm *see:* parametric right-hand —
side uncertainty, duality and applications *see:* Robust linear programming with right-hand- —
sided differential *see:* one- —
Sierpinski theorem
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
σ-classes
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
sigma-field
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
sign of a circuit
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
sign function
[65D10, 65K05]
(*see:* **Overdetermined systems of linear equations**)
sign-invariance model
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
sign matrix
[15A39, 52A22, 60D05, 68Q25, 90C05]
(*see:* **Farkas lemma**; **Probabilistic analysis of simplex algorithms**)
sign-nonsingular matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
sign pattern of a matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
sign-solvable linear system
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
sign vector
[90C09, 90C10]
(*see:* **Oriented matroids**)
signal processing *see:* nonGaussian —; nonlinear —
Signal processing with higher order statistics
(90C26, 90C90)

(*referred to in:* **Global optimization methods for harmonic retrieval**)
(*refers to:* **Global optimization methods for harmonic retrieval**)
signature
[90C09, 90C10]
(*see:* **Oriented matroids**)
signed bigraph
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
signed circuits
[90C09, 90C10]
(*see:* **Oriented matroids**)
signed cocircuits
[90C09, 90C10]
(*see:* **Oriented matroids**)
signed decomposition
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
signed decomposition *see:* Lasserre —; Lawrence —
signed digraph
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
signed set
[90C09, 90C10]
(*see:* **Oriented matroids**)
signed set *see:* opposite of a —
signed sets *see:* orthogonal —
signed subset
[90C09, 90C10]
(*see:* **Oriented matroids**)
signomial programming
[90C26, 90C90]
(*see:* **Global optimization in generalized geometric programming**)
signomials
[90C26, 90C90]
(*see:* **Global optimization in generalized geometric programming**)
Signorini condition
[49J40, 49Q10, 70-08, 74K99, 74Pxx]
(*see:* **Quasivariational inequalities**)
Signorini-Coulomb unilateral frictional contact
[49J40, 49Q10, 70-08, 74K99, 74Pxx]
(*see:* **Quasivariational inequalities**)
SIM
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
similarity measure
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
similarity subtree isomorphism *see:* maximal —; maximum —
similarity of surrogates
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
simple arrangement
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements in optimization**)
simple Bayes
(*see:* **Bayesian networks**)
simple homogeneous process
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**)
simple integer recourse
[90C10, 90C15]
(*see:* **Stochastic integer programs**)
simple integer recourse
[90C11, 90C15]
(*see:* **Stochastic programming with simple integer recourse**)
simple integer recourse *see:* Stochastic programming with —; two-stage stochastic programs with —
simple linkage
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
simple order isotonic regression
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
simple plant location problem
[90B80, 90B85]
(*see:* **Warehouse location problem**)
simple plant location problem
[90B80, 90B85]
(*see:* **Warehouse location problem**)
simple polyhedron
[90C60]
(*see:* **Complexity of degeneracy**)
simple polyhedron
[90C60]
(*see:* **Complexity of degeneracy**)
simple principal pivot
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
simple recourse
[90C15, 90C30, 90C35]
(*see:* **Optimization in water resources**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
simple recourse
[90C11, 90C15]
(*see:* **Stochastic programming with simple integer recourse**)
Simple recourse problem
(90C06, 90C08, 90C15)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**)
simple recourse problem *see:* dual method for the —; primal method for the —
Simple recourse problem: dual method
(90-08, 90C05, 90C06, 90C08, 90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**;

Extremum problems with probability functions: kernel type solution methods; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

Simple recourse problem: primal method
(90-08, 90C05, 90C06, 90C08, 90C15, 49M25)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem**; **Simple recourse problem: dual method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem**; **Simple recourse problem: dual method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

simplex
[52B11, 52B45, 52B55, 65K05, 90C06, 90C25, 90C26, 90C30, 90C31, 90C35, 90C90]
(*see:* **Bisection global optimization methods**; **Global optimization in binary star astronomy**; **Sensitivity and stability in NLP: approximation**; **Sequential simplex method**; **Simplicial decomposition algorithms**; **Volume computation for polytopes: strategies and performances**)
simplex
[65K05, 90C06, 90C25, 90C30, 90C35]
(*see:* **Bisection global optimization methods**; **Simplicial decomposition algorithms**)
simplex *see:* centroid of a —; dual —; initial —; lexicographic —; p- —; regular —; standard —
simplex algorithm
[03H10, 49J27, 90C05, 90C34]
(*see:* **Linear programming: Klee–Minty examples**; **Semi-infinite programming and control problems**)
simplex algorithm
[90C05, 90C06, 90C10, 90C11, 90C30, 90C33, 90C57, 90C90]
(*see:* **Convex-simplex algorithm**; **Frank–Wolfe algorithm**; **Linear programming: Klee–Minty examples**; **Modeling difficult optimization problems**; **Pivoting algorithms for linear programming generating two paths**; **Simplicial decomposition**)
simplex algorithm *see:* Convex- —; dual —; network —; Parametric linear programming: cost —; parametric objective —; parametric right-hand side —; primal —; variant of the —
simplex algorithms *see:* primal and dual —; Probabilistic analysis of —
simplex method
[05B35, 65K05, 65K10, 68Q99, 90C05, 90C06, 90C20, 90C25, 90C33, 90C35]
(*see:* **ABS algorithms for optimization**; **Branch and price: Integer programming with column generation**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Linear programming: karmarkar projective algorithm**; **Simplicial decomposition algorithms**)
simplex method
[90C05]
(*see:* **Linear programming**)
simplex method *see:* downhill —; lexicographic —; lexicographic dual —; lexicographic primal —; Sequential —; steepest edge —
simplex program
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
simplex reduction
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
simplex tableau *see:* terminal —
simplex type algorithm
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
simplexes *see:* system of —
simplices
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
simplices *see:* bounds on —
simplicial *see:* nonlinear —
simplicial algorithms
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
simplicial constraints
[90C60]
(*see:* **Complexity theory: quadratic programming**)
simplicial constraints
[90C60]
(*see:* **Complexity theory: quadratic programming**)
Simplicial decomposition
(90C30)
(*referred to in:* **Decomposition principle of linear programming**; **Generalized benders decomposition**; **MINLP: generalized cross decomposition**; **MINLP: logic-based methods**; **Simplicial decomposition algorithms**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)
(*refers to:* **Decomposition principle of linear programming**; **Frank–Wolfe algorithm**; **Generalized benders decomposition**; **MINLP: generalized cross decomposition**; **MINLP: logic-based methods**; **Simplicial decomposition algorithms**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)
simplicial decomposition
[90C06, 90C25, 90C30, 90C35]
(*see:* **Frank–Wolfe algorithm**; **Simplicial decomposition**; **Simplicial decomposition algorithms**)
simplicial decomposition
[90C30]
(*see:* **Frank–Wolfe algorithm**; **Simplicial decomposition**)
simplicial decomposition *see:* disaggregate —; restricted —
Simplicial decomposition algorithms
(90C06, 90C25, 90C35)
(*referred to in:* **Decomposition principle of linear programming**; **Generalized benders decomposition**; **MINLP: generalized cross decomposition**; **MINLP: logic-based methods**; **Simplicial decomposition**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)
(*refers to:* **Decomposition principle of linear programming**; **Generalized benders decomposition**; **MINLP: generalized cross decomposition**; **MINLP: logic-based methods**; **Simplicial decomposition**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming: decomposition methods**)
simplicial partition
[90C20]
(*see:* **Standard quadratic optimization problems: applications**)
Simplicial pivoting algorithms for integer programming
(90C10, 90C05)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Criss-cross pivoting rules**;

Decomposition techniques for MILP: lagrangian relaxation; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Linear programming**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Pivoting algorithms for linear programming generating two paths**; **Set covering, packing and partitioning problems**; **Stable set problem: branch & cut algorithms**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Criss-cross pivoting rules**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Linear programming**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Pivoting algorithms for linear programming generating two paths**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**; **Set covering, packing and partitioning problems**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Time-dependent traveling salesman problem**)

simpliciality
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)

simpliciality *see:* near- —

simplicity
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)

simplicity *see:* near- —

Simulated annealing
(90C90, 90C27)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Broadcast scheduling problem**; **Discrete stochastic optimization**; **Genetic algorithms**; **Global optimization based on statistical models**; **Global optimization: hit and run methods**; **Global optimization in Lennard–Jones and morse clusters**; **Job-shop scheduling problem**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Optimization-based visualization**; **Optimization in medical imaging**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Random search methods**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Evolutionary algorithms in combinatorial optimization**; **Global optimization based on statistical models**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)

simulated annealing
[00-02, 01-02, 03-02, 05-04, 62C10, 65K05, 68Q25, 68T20, 68T99, 90B80, 90C05, 90C10, 90C15, 90C25, 90C26, 90C27, 90C30, 90C59, 90C90]
(*see:* **Bayesian global optimization**; **Bayesian networks**; **Capacitated minimum spanning trees**; **Communication network assignment problem**; **Design optimization in computational fluid dynamics**; **Evolutionary algorithms in combinatorial optimization**; **Global optimization in binary star astronomy**; **Global optimization: hit and run methods**; **Maximum constraint satisfaction: relaxations and upper bounds**; **Metaheuristics**; **Metropolis, Nicholas Constantine**; **MINLP: design and scheduling of batch processes**; **Random search methods**; **Simulated annealing methods in protein folding**; **Vehicle routing problem with simultaneous pickups and deliveries**)

simulated annealing
[90B80, 90C05, 90C25, 90C26, 90C29, 90C90, 92C40]
(*see:* **Facilities layout problems**; **Global optimization: hit and run methods**; **Metropolis, Nicholas Constantine**; **Monte-Carlo simulated annealing in protein folding**; **Optimal design of composite structures**)

simulated annealing *see:* adaptive —; stochastic —

simulated annealing and genetic algorithm
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)

simulated annealing and its application to protein folding *see:* Adaptive —

Simulated annealing methods in protein folding
(90C90)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Differential equations and global optimization**; **Facilities layout problems**; **Genetic algorithms**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization in Lennard–Jones and morse clusters**; **Graph coloring**; **Maximum satisfiability problem**; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet**

annealing; **Phase problem in X-ray crystallography: Shake and bake approach**; **Random search methods**; **Set covering, packing and partitioning problems**; **Simulated annealing**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Genetic algorithms**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization in Lennard–Jones and morse clusters**; Global optimization in protein folding; **Molecular structure determination: convex global underestimation**; **Monte-Carlo simulated annealing in protein folding**; **Multiple minima problem in protein folding: αBB global optimization approach**; **Packet annealing**; **Phase problem in X-ray crystallography: Shake and bake approach**; **Protein folding: generalized-ensemble algorithms**; **Random search methods**; **Simulated annealing**; **Stochastic global optimization: stopping rules**; **Stochastic global optimization: two-phase methods**)
simulated annealing in protein folding *see:* Monte-Carlo —
simulated quench
[92C05]
(*see:* **Adaptive simulated annealing and its application to protein folding**)
simulating annealing
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
simulation
[90C39]
(*see:* **Emergency evacuation, optimization modeling**; **Neuro-dynamic programming**)
simulation
[90C27, 90C39]
(*see:* **Neuro-dynamic programming**; **Operations research and financial markets**)
simulation *see:* Derivatives of markov processes and their —; dynamic —; monte-Carlo —; numerical —; optimization- —; process —
simulation algorithm *see:* Monte-Carlo —
simulation-based optimization
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
simulation-based optimization
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
simulation of derivatives
[90C15]
(*see:* **Derivatives of probability measures**)
simulation models
[90-02]
(*see:* **Operations research models for supply chain management and design**)
simulation models *see:* supply chain —
simulation problems
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
simulation procedure *see:* Monte-Carlo —
simulation programs *see:* process —
simulations for stochastic optimization *see:* Monte-Carlo —
simulator
(*see:* **State of the art in modeling agricultural systems**)
simultaneous
[34A55, 35R30, 62G05, 62G08, 62J02, 62K05, 62P10, 62P30, 76R50, 80A20, 80A23, 80A30]
(*see:* **Identification methods for reaction kinetics and transport**)
simultaneous adjustment
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
simultaneous approach
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
simultaneous Diophantine approximation problem
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
simultaneous displacements *see:* method of —
simultaneous equation models
[90C26, 90C30]
(*see:* **Forecasting**)
Simultaneous estimation and optimization of nonlinear problems
(93E20, 93E12, 49J15, 62J02, 62M10, 62M20, 91B28)
(*referred to in:* **Generalizations of interior point methods for the linear complementarity problem**; **Mathematical programming methods in supply chain management**)
(*refers to:* **Complementarity algorithms in pattern recognition**; **Generalizations of interior point methods for the linear complementarity problem**; **Mathematical programming methods in supply chain management**)
simultaneous pickups and deliveries *see:* Vehicle routing problem with —
simultaneous synthesis
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
single
(*see:* **Peptide identification via mixed-integer optimization**)
single assignment
[26A24, 65D25]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**)
single assignment algorithms
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
single assignment property
[90B80, 90B85]
(*see:* **Warehouse location problem**)
single-class software package
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
single cluster statistic *see:* generalized —
single color
[05C85]
(*see:* **Directed tree networks**)
single-commodity model in OR
[90B80, 90B85]
(*see:* **Warehouse location problem**)

single commodity network flow problem *see:* nonlinear —
single-commodity single-criterion uncapacitated static multifacility *see:* discrete —
single-criterion problem in OR
[90B80, 90B85]
(*see:* **Warehouse location problem**)
single-criterion uncapacitated static multifacility *see:* discrete single-commodity —
single-crystal X-ray diffraction data *see:* Optimization techniques for phase retrieval based on —
single depot/multiple depots
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
Single-depot vehicle scheduling problem
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
Single-depot vehicle scheduling problems
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
single-ended
(*see:* **Railroad crew scheduling**)
single-ended crew district
(*see:* **Railroad crew scheduling**)
single facility location
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**)
Single facility location: circle covering problem
(90B85, 90C27)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Carathéodory theorem**; **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
Single facility location: multi-objective euclidean distance location
(90B85)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: circle covering problem**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
Single facility location: multi-objective rectilinear distance location
(90B85)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**;

Resource allocation for epidemic control; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
single-facility problem in OR
[90B80, 90B85]
(*see:* **Warehouse location problem**)
single fixed cost with capacity constraints
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
single fixed cost with no capacity constraints
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
single hub heuristic
[90C35]
(*see:* **Multi-index transportation problems**)
single index market model *see:* Sharpe —
single-linkage *see:* multilevel —
single locomotive scheduling models
(*see:* **Railroad locomotive scheduling**)
single locomotive type
(*see:* **Railroad locomotive scheduling**)
single-lookahead-unit-resolution
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
single parametric mixed integer linear program
[90C11, 90C31]
(*see:* **Multiparametric mixed integer linear programming**)
single path routing pattern model
[68Q25, 90B80, 90C05, 90C27]
(*see:* **Communication network assignment problem**)
single-period model
[91B28]
(*see:* **Financial optimization**)
single-product campaign
[90C26]
(*see:* **Global optimization in batch design under uncertainty**; **MINLP: design and scheduling of batch processes**)
single-ratio fractional (hyperbolic) 0-1 programming problem
(*see:* **Fractional zero-one programming**)
single-ratio fractional program
[90C32]
(*see:* **Fractional programming**)
single-ratio programs
[90C32]
(*see:* **Fractional programming**)
single run SQG
[90C15]
(*see:* **Stochastic quasigradient methods: applications**)
single smooth function
[57R12, 90C31, 90C34]
(*see:* **Smoothing methods for semi-infinite optimization**)
single source shortest path tree problem
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
single-stage IM
[90B50]
(*see:* **Inventory management in supply chains**)
single stage inventory management models
[90B50]
(*see:* **Inventory management in supply chains**)
single underlying method
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
single-valued boundary laws and variational equalities
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
single valued maps *see:* Generalized monotone —
single versus Multiperiod Models
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
singleton
[90C35]
(*see:* **Feedback set problems**)
singular component
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
singular control problem
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
singular Fréchet subdifferential
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
singular limiting subdifferential
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
singular local attractors
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
singular matrix
[15A99, 65G20, 65G30, 65G40, 90C26]
(*see:* **Interval linear systems**)
singular set
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
singular value *see:* structured —
singular value decomposition solution *see:* truncated —
singularities
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
singularities *see:* generic —; Parametric optimization: embeddings, path following and —
singularity
[05B35, 20F36, 20F55, 52C35, 57N65, 90C31, 90C34]
(*see:* **Hyperplane arrangements**; **Parametric global optimization: sensitivity**)
singularity of an arrangement of hyperplanes
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
sink
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
sink node
[90C35]
(*see:* **Maximum flow problem**)

sinusoidal parameter estimation problem
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
sIP
[57R12, 65K05, 90C26, 90C31, 90C33, 90C34]
(*see:* **Adaptive convexification in semi-infinite optimization**; **Parametric global optimization: sensitivity**; **Smoothing methods for semi-infinite optimization**)
SIP *see:* convex —; linear —; nonlinear —; structural stability of —
SIP(f,h,g) *see:* global structual stability of —
SIP index set
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
SIP problem *see:* discretized —; duality of the linear —; nonlinear discretized —
SIP problems *see:* linear —
SIPs *see:* equivalence of —
site *see:* active —
size *see:* list —; machine —; population —; step —
size effects in microclusters
[90C26, 90C90]
(*see:* **Global optimization in Lennard–Jones and morse clusters**)
size of a graph
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
size of input data
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
size of the input of a Turing machine
[90C60]
(*see:* **Complexity theory**)
size of a problem instance
[90C60]
(*see:* **Computational complexity theory**)
size rule *see:* divergent series step- —
size threshold *see:* determination of clusters —
sizing *see:* lot —
sizing optimization
[49J20, 49J52]
(*see:* **Shape optimization**)
sizing problem *see:* capacitated lot- —; Economic lot- —
sizing variables
[90C26, 90C90]
(*see:* **Structural optimization: history**)
skeleton
[68Q20]
(*see:* **Optimal triangulations**)
skeleton *see:* lMT- —
skew-symmetric matrix
[15A39, 90C05]
(*see:* **Farkas lemma**; **Linear optimization: theorems of the alternative**)
skew-symmetric matrix
[15A39, 90C05]
(*see:* **Farkas lemma**)
skew-symmetric matrix M
[65K05, 90C20, 90C33]
(*see:* **Principal pivoting methods for linear complementarity problems**)
skew-symmetric proximity
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
Skewed VNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
skewness
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
skorokhod problem
[60G35, 65K05, 65K10, 90C90]
(*see:* **Differential equations and global optimization**; **Variational inequalities: projected dynamical system**)
Skorokhod problem
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)
slack cable
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
slack constraints
[90C60]
(*see:* **Complexity of degeneracy**)
slack constraints
[90C60]
(*see:* **Complexity of degeneracy**)
slackness *see:* complementarity —; complementary —; ϵ-complementary —; strict complementarity —; strict complementary —
slackness condition *see:* strict complementarity —
slackness conditions *see:* complementary —
slackness relations *see:* complementary —
slacks *see:* dual —
Slater's condition
[90C05, 90C15, 90C25, 90C26, 90C29, 90C30, 90C31, 90C34, 91A65, 91B28]
(*see:* **Bilevel programming: implicit function approach**; **Duality for semidefinite programming**; **Kuhn–Tucker optimality conditions**; **Nondifferentiable optimization: parametric programming**; **Probabilistic constrained linear programming: duality theory**; **Semi-infinite programming and applications in finance**; **Semi-infinite programming, semidefinite programming and perfect duality**)
Slater constraint
[90C26]
(*see:* **Invexity and its applications**)
Slater constraint qualification
[90C25, 90C29, 90C30, 90C33]
(*see:* **Lagrangian multipliers methods for convex programming**; **Multi-objective optimization: lagrange duality**; **Optimization with equilibrium constraints: A piecewise SQP approach**)
Slater constraint qualification *see:* generalized —
Slater CQ *see:* Strong —; Weak —
Slater dual *see:* extended Lagrange– —
Slater theorem
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
slave scheme *see:* master- —
slice *see:* time —

slope
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
slope arithmetic
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
slope lemma *see:* first —; second —; third —
slope lemmas
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
slope set
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)
slopes *see:* interval —
SLP
[90C15]
(*see:* **L-shaped method for two-stage stochastic programs with recourse**)
SLP *see:* decomposition of —
SLP algorithms
[90C30, 90C90]
(*see:* **MINLP: applications in blending and pooling problems**)
Smale function *see:* Chen–Harker–Kanzow– —
small *see:* sufficiently —
small gain
[93D09]
(*see:* **Robust control**)
small negative real numbers *see:* infinitely —
small positive real numbers *see:* infinitely —
small real numbers *see:* infinitely —
small residual
[90C30]
(*see:* **Nonlinear least squares problems**)
smaller *see:* lexicographically —
smallest enclosing circle
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
smallest enclosing-circle problem
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
smallest index rule
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
smallest K-majorant
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
Smallest of Several Ratios *see:* Maximization of the —
SMIN
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)
SMIN-αBB
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
SMIN-αBB algorithm
[90C26]
(*see:* **Bilevel optimization: feasibility test and flexibility index**)
Smith conjecture
[90C27]
(*see:* **Steiner tree problems**)
Smith–Walford-one algorithm
[90C35]
(*see:* **Feedback set problems**)
Smith–Walford one-reducible graph
[90C35]
(*see:* **Feedback set problems**)
Smoluchowski–Kramers equation
[60G35, 65K05]
(*see:* **Differential equations and global optimization**)
smooth function *see:* single —
Smooth nonlinear nonconvex optimization
(90C26)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: models, algorithms and software**; **Global optimization in batch design under uncertainty**; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: models, algorithms and software**; **Global optimization in batch design under uncertainty**; **Global optimization in generalized geometric programming**; **Global optimization methods for systems of nonlinear equations**; **Global optimization in phase and chemical reaction equilibrium**; **Interval global optimization**; **MINLP: branch and bound global optimization algorithm**; **MINLP: global optimization with αBB**)
smooth nonlinear optimization
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
smooth optimization *see:* Derivative-free methods for non- —
smooth potentials and stability in mechanics
[49J52, 49Q10, 74G60, 74H99, 74K99, 74Pxx, 90C90]
(*see:* **Quasidifferentiable optimization: stability of dynamic systems**)
smoothing
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
smoothing *see:* exponential —
smoothing algorithm *see:* potential —
smoothing algorithms
[90Cxx]
(*see:* **Discontinuous optimization**)
smoothing functions
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
smoothing methods
[90C26, 90C30, 90C33]
(*see:* **Forecasting**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
smoothing methods
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)

smoothing methods for nonlinear complementarity problems and variational inequalities *see:* Nonsmooth and —
Smoothing methods for semi-infinite optimization
(90C34, 90C31, 57R12)
(*refers to:* Generalized semi-infinite programming: optimality conditions; **Nonsmooth analysis: weak stationarity**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Semi-infinite programming, semidefinite programming and perfect duality**)
smoothing Newton method
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
smoothing NM
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
smoothing-nonsmooth reformulation
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
smoothing of the potential energy
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
smoothness *see:* lack of —
snakes
[90C90]
(*see:* **Optimization in medical imaging**)
SNDP
[90-XX]
(*see:* **Survivable networks**)
SNE *see:* GO for —
SNLP *see:* Semidefinite programming and the sensor network localization problem —
SO
[49J20, 49J52]
(*see:* **Shape optimization**)
soaring direction
[90Cxx]
(*see:* **Discontinuous optimization**)
Sobel edge filter
[90C90]
(*see:* **Optimization in medical imaging**)
Sobolev space
[49J52]
(*see:* **Hemivariational inequalities: eigenvalue problems**)
social cost *see:* production realizing with minimal —
social utility function
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
SOCQ *see:* ben-Tal —; directional —; McCormick —
software *see:* Continuous global optimization: models, algorithms and —; high-level —; low-level —; mathematical —; medium-level —; Optimization —
software development and evaluation
[90C05]
(*see:* **Continuous global optimization: models, algorithms and software**)
software for homotopy methods
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
software library *see:* general-purpose —
software package *see:* block truncated Newton —; multiple-class —; single-class —
software package for specific mathematical areas
[90C10, 90C26, 90C30]
(*see:* **Optimization software**)
software routine *see:* individual —
software routines *see:* package of basic —
soil
(*see:* **State of the art in modeling agricultural systems**)
Solanki method
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
Solomon algebra *see:* Orlik– —
solomonoff-Kolmogorov-Chaitin complexity
[90C60]
(*see:* **Kolmogorov complexity**)
Solomonoff–Kolmogorov–Chaitin complexity
[90C60]
(*see:* **Kolmogorov complexity**)
solution
[90C10, 90C29]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**; **Multi-objective optimization: pareto optimal solutions, properties**)
solution *see:* admissible —; basic —; basic feasible —; best-compromise —; (C_m^J)-efficient —; computable —; cooperative —; corner —; degenerate basic —; distinguished —; efficient —; enumerative —; ε-reserved —; essential optimal —; extreme feasible —; extreme point —; feasible —; feedback Stackelberg —; global —; global minimum —; global optimal —; incomplete —; incumbent —; infeasible —; initial —; integer —; interior —; Lipschitz stable —; local —; locally optimal —; M-Pareto optimal —; minimax —; minimum norm —; most preferred —; neighborhood of a —; noncooperative —; nondegenerate —; nondominated —; noninferior —; nonsupported efficient —; optimal —; optimum —; Pareto —; Pareto efficient —; Pareto optimal —; polynomial time —; practically feasible computational —; primal —; primal-dual —; problem —; properly efficient —; quasi-optimal —; regular —; spanning tree —; stable —; Stochastic network problems: massively parallel —; strictly complementary —; strongly polynomial —; strongly stable —; strongly stable optimal —; supported efficient —; truncated singular value decomposition —; unbounded —; unsupported nondominated —; value of stochastic —; weakly efficient —; weakly nondominated —; weakly noninferior —; weakly Pareto optimal —
solution by active sets and interior point methods *see:* Successive quadratic programming: —
solution algorithm *see:* incremental-iterative —
solution algorithms
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
solution of the alignment problem
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)

solution of bilevel programming problems
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)
solution block
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
solution of the convex moment problem
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
solution of equations *see:* rotation in the —
solution of the Euclidean distance location problem *see:* iterative —
solution of facility location problems with staircase costs
[90B80, 90C11]
(*see:* **Facility location with staircase costs**)
solution of the Hamilton–Jacobi–Bellman equation
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)
solution of inverse problems *see:* formulation and —
solution mapping *see:* optimal —
solution method *see:* support problems —
solution methodologies
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
solution methods
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
solution methods *see:* exact —; Extremum problems with probability functions: kernel type —
Solution methods for multivalued variational inequalities
(47J20, 49J40, 90C33, 65K10)
solution of multistage mean-variance optimization problems *see:* Decomposition algorithms for the —
solution point *see:* bounds on the distance of a feasible point to a —
solution-point bounds for NLP
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
solution of a problem
[90C60]
(*see:* **Computational complexity theory**)
solution procedures
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
solution of a program *see:* optimal —
solution quality *see:* establishing —
solution robust
[90C90, 91B28]
(*see:* **Robust optimization**)
solution set
[15A99, 65G20, 65G30, 65G40, 90C26, 90C31, 90C33]
(*see:* **Interval linear systems**; **Sensitivity and stability in NLP: continuity and differential stability**; **Topological methods in complementarity theory**)
solution set *see:* nondominated —; noninferior —; Pareto optimal —; stability of a —
solution set in problem solving *see:* restriction to the —
solution space
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
solution space *see:* convexity property of the —
solution strategies
[90C05]
(*see:* **Continuous global optimization: models, algorithms and software**)
solution of the system
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
solution of a system
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
solution vector estimates for parametric NLPS *see:* Bounds and —
solution of the Wilson equation *see:* regular —
solutions *see:* almost complementary —; basic —; comparison of parametric —; efficient —; enumerating extreme point —; equilibrium —; exploiting the interplay between primal and dual —; jumps of optimal —; least squares —; local —; nonsupported efficient —; Pareto optimal —; set of feasible —; set of potential efficient —; supported efficient —; verifying equilibrium —
solutions of equations *see:* test for the existence of —
solutions of nonlinear systems of equations *see:* error bound for approximate —; existence of —; rigorous bound for —; uniqueness of —
solutions, properties *see:* Multi-objective optimization: pareto optimal —
solvability *see:* polynomial —
solvability of equations
[01A99]
(*see:* **Lagrange, Joseph-Louis**)
solvability theorem
[90C26]
(*see:* **Global optimization: envelope representation**)
solvable linear system *see:* sign- —
solvation effects
[65K10, 92C40]
(*see:* **Multiple minima problem in protein folding: αBB global optimization approach**)
solvent design approaches *see:* Optimal —
solver
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
solver
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
solver *see:* flow —
solver formats *see:* independent of —
solvers
(*see:* **Planning in the process industry**)
solvers *see:* bottlenecks in NLP —
solving *see:* constraint —; iterative linear equation- —; restriction to the solution set in problem —
solving CAP on trees *see:* exact algorithm for —; heuristic approach to —
solving environment *see:* problem —
Solving hemivariational inequalities by nonsmooth optimization methods

(49J40, 49J52, 90C30, 65K05)
(*referred to in:* **Composite nonsmooth optimization**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Composite nonsmooth optimization**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
Solving large scale and sparse semidefinite programs
(90C25, 90C30)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Duality for semidefinite programming**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Maximum cut problem, MAX-CUT**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and the sensor network localization problem, SNLP**; **Semidefinite programming and structural optimization**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Symmetric systems of linear equations**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Duality for semidefinite programming**; **Interior point methods for semidefinite programming**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Linear programming**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Semidefinite programming and determinant maximization**; **Semidefinite programming: optimality conditions and stability**; **Semidefinite programming and structural optimization**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Symmetric systems of linear equations**)
solving a problem *see:* Turing machine —
solving a problem instance in time m *see:* algorithm —
solving vehicle routing problems *see:* approximate methods for —; constructive methods for —; exact methods for —
SONC
[90C31]
(*see:* **Sensitivity and stability in NLP: approximation**)
SOR method
[90C25, 90C33, 90C55]
(*see:* **Splitting method for linear complementarity problems**)
sorting
[90C29]
(*see:* **Multicriteria sorting methods**)
sorting
[90C29]
(*see:* **Multicriteria sorting methods**)
sorting method *see:* multicriteria —
sorting methods *see:* Multicriteria —
sorting problem
[90C29]
(*see:* **Preference disaggregation approach: basic features, examples from financial decision making**)
sorting problems
[90C26, 90C29, 91B28]
(*see:* **Portfolio selection and multicriteria analysis**; **Preference disaggregation approach: basic features, examples from financial decision making**)
sOS
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
source
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
source *see:* virtual —
source algorithm *see:* virtual —

source class
[90C05, 90C34]
(*see:* **Semi-infinite programming: methods for linear problems**)
source code transformation
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
source concept in auction algorithms *see:* virtual —
source node
[90C35]
(*see:* **Maximum flow problem**)
source shortest path tree problem *see:* single —
sources
[65D18, 90B85, 90C26]
(*see:* **Global optimization in location problems**)
Southwell method *see:* Gauss– —
SP *see:* split-variable formulation of —
space *see:* canonical function —; convexity property of the solution —; criterion —; design —; dual —; extended canonical function —; feature —; function —; Hilbert —; image —; kinetically admissible —; lineality —; Linear —; linear topological —; local properties of the configuration —; measure —; metric —; monotone operator on a Banach —; null —; optimization in a vector —; outcome —; phase —; probability measure —; range —; real vectors —; resource-payoff —; Sobolev —; solution —; symmetric element in a Hilbert —; triangulation of Euclidean —; trustworthy —; variable —; vector —
space approach to optimization *see:* Image —
space-bounded Turing machine *see:* exponentially —; polynomially —
space complexity of a deterministic Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
space complexity of a nondeterministic Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
space decomposition *see:* range and null —
space dilation
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: subgradient optimization methods**)
space filling *see:* Global optimization using —
space filling curve
[90C26]
(*see:* **Global optimization using space filling**)
space filling curve
[90C26]
(*see:* **Global optimization using space filling**)
space filling curves *see:* approximation of —
space graphs *see:* searching state —
space methods *see:* full —; Successive quadratic programming: full —
space model *see:* vector —
space models for entropy optimization for image reconstruction *see:* vector- —
space reduction *see:* weighting —
space search *see:* formulation —
space search algorithm *see:* optimal state —; recursive state —; state —; synchronized distributed state —
space SQP *see:* full —; reduced —
space SQP method *see:* full —
space successive quadratic programming *see:* full —
Space-time network
(*see:* **Railroad crew scheduling**; **Railroad locomotive scheduling**)
space-time network *see:* weekly —
space type methods *see:* Krylov —
space of x variables *see:* full —
spaces *see:* analyzing almost empty —; Best approximation in ordered normed linear —; Increasing and convex-along-rays functions on topological vector —; Increasing and positively homogeneous functions on topological vector —; measure —; ordered vector —; semi-ordered —; Steiner ratio in Banach —; subdifferentiability —
span of a T-coloring frequency assignment
[05-XX]
(*see:* **Frequency assignment problem**)
spanning acyclic tournament
[90C10, 90C11, 90C20]
(*see:* **Linear ordering problem**)
spanning arborescence problem *see:* capacitated minimum —
spanning network *see:* multiphase —
spanning tree
[90-XX]
(*see:* **Survivable networks**)
spanning tree
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
spanning-tree *see:* minimum —; minimum ratio —
spanning tree problem *see:* bounded degree minimum —; capacitated minimum —; minimum —; resource-constrained minimum —
spanning tree solution
[90C35]
(*see:* **Minimum cost flow problem**)
spanning tree structure
[90C35]
(*see:* **Minimum cost flow problem**)
spanning tree structure *see:* feasible —; optimal —
spanning trees *see:* Capacitated minimum —; minimum —
sparse doublet
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**)
sparse least squares problem
[65Fxx]
(*see:* **Least squares problems**)
sparse matrix
[90C25, 90C30]
(*see:* **Solving large scale and sparse semidefinite programs**; **Successive quadratic programming: full space methods**)
sparse matrix
[90C25, 90C30]
(*see:* **Solving large scale and sparse semidefinite programs**)
sparse semidefinite programs *see:* Solving large scale and —
sparse triplet
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**)
sparsity
[49-04, 65K05, 65Y05, 68N20, 90C30]

(*see:* **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: parallel computation**)
sparsity
[65K05, 90C25, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
spatial branch and bound *see:* reformulation/ —
spatial Characteristic
(*see:* **State of the art in modeling agricultural systems**)
spatial competition facility location model
[65K10, 90C31]
(*see:* **Sensitivity analysis of variational inequality problems**)
spatial Cournot–Nash equilibrium
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
spatial interaction
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)
spatial interaction
[90B80, 90C10]
(*see:* **Facility location problems with spatial interaction**)
spatial interaction *see:* Facility location problems with —
spatial-interaction model
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
spatial markets
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
spatial markets *see:* aspatial and —
spatial oligopoly model
[91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**)
Spatial price equilibrium
(91B50, 91B28)
(*referred to in:* **Equilibrium networks**; **Financial equilibrium**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Oligopolistic market equilibrium**; **Traffic network equilibrium**; **Walrasian price equilibrium**)
(*refers to:* **Equilibrium networks**; **Financial equilibrium**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Oligopolistic market equilibrium**; **Traffic network equilibrium**; **Walrasian price equilibrium**)
spatial price equilibrium
[65K10, 90C31, 90C35, 91B28, 91B50]
(*see:* **Multicommodity flow problems**; **Sensitivity analysis of variational inequality problems**; **Spatial price equilibrium**)
spatial price equilibrium
[90C30]
(*see:* **Equilibrium networks**)
spatial price equilibrium problem
[91B28, 91B50]
(*see:* **Spatial price equilibrium**)
spatial price equilibrium problem
[90C30]
(*see:* **Equilibrium networks**)
spatial price equilibrium problem *see:* network structure of the —
spatial segmentation
[90C90]
(*see:* **Optimization in medical imaging**)
SPE
[90C35]
(*see:* **Multicommodity flow problems**)
special facilities *see:* residents of —
special functions: algorithms and complexity *see:* Regression by —
special model features
(*see:* **Planning in the process industry**)
special properties of crisp relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
special properties of fuzzy relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
special properties of heterogeneous relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
special properties of homogeneous relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
special properties of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
special structure mixed integer αBB algorithm
[65K05, 90C11, 90C26]
(*see:* **MINLP: global optimization with αBB**)
species
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
species *see:* component —
species index set
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
specific *see:* unit- —
specific mathematical areas *see:* software package for —
spectral method *see:* Galerkin —
Spectral projected gradient methods
(49M07, 49M10, 65K, 90C06, 90C20)
spectral radius
[65K05, 90Cxx]
(*see:* **Symmetric systems of linear equations**)
spectroscopic visual binary star
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
spectrum *see:* higher-order —
Spedicato algorithms for linear equations and linear least squares *see:* Abaffi–Broyden– —
speedup *see:* linear —
sphere *see:* minimax location on a —; minimum —

sphere growing technique
[90C35]
(*see:* **Feedback set problems**)
sphere method *see:* largest inscribed —
sphere problem *see:* minimum —
spherical reduction
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
split *see:* phase —
split-variable formulation of SP
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
splitting
[90C30]
(*see:* **Cost approximation algorithms**)
splitting *see:* operator —; regular Q- —
splitting algorithm *see:* operator —; principal variation —; tree- —
splitting field
[01A50, 01A55, 01A60]
(*see:* **Fundamental theorem of algebra**)
splitting method
[90C25, 90C33, 90C55]
(*see:* **Splitting method for linear complementarity problems**)
Splitting method for linear complementarity problems
(90C25, 90C55, 90C25, 90C33)
(*referred to in:* **Linear complementarity problem**)
(*refers to:* **Lagrangian multipliers methods for convex programming**; **Linear complementarity problem**)
splitting methods in quadratic programming *see:* matrix —
splitting Newton method
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
splitting rule
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
splitting/unsplitting of load
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
splitting variable representation
[90C30, 90C35]
(*see:* **Optimization in water resources**)
splitting variables
[90C30, 90C35]
(*see:* **Optimization in water resources**)
SPLP
[90B80, 90B85]
(*see:* **Warehouse location problem**)
spot *see:* cold —; hot —
spot interest rate
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
spot rate *see:* τ-estimate of the —
spot rate for bonds with constant maturities *see:* estimating the —
spot rate estimation *see:* τ-programmed problem of —
spots *see:* cold —; hot —
SPP
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
SPR
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
spread
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
SPT
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
SQG
[90C15]
(*see:* **Derivatives of probability measures**)
SQG *see:* single run —
sQG methods
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
SQG methods
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
(SQG) methods *see:* stochastic Quasigradient —
sQG projection methods
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
SQG projection methods
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
sQP
[90C30, 90C90, 93-XX]
(*see:* **Dynamic programming: optimal control applications**; **Successive quadratic programming: applications in the process industry**)
SQP
[65L99, 90C30, 90C90, 93-XX]
(*see:* **Optimization strategies for dynamic systems**; **Successive quadratic programming: applications in the process industry**)
SQP *see:* convex —; full space —; multiplier-free reduced Hessian —; nonconvex —; quadratic programming problem in —; reduced space —
SQP approach *see:* Optimization with equilibrium constraints: A piecewise —
SQP method *see:* full space —; reduced Hessian —
SQP optimization in industrial problems
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
SQP type algorithm
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
SQPIP
[49K20, 49M99, 90C55]

(*see:* **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
SQPIP methods *see:* affine scaling —; primal-dual —
square *see:* equal circles in a —
square composition of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
square merit function *see:* list —
square product of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
square-root-free Givens transformation
[15A23, 65F05, 65F20, 65F22, 65F25]
(*see:* **QR factorization**)
square-root method
[65Fxx]
(*see:* **Least squares problems**)
square-root transformation
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
square-root transformation *see:* logarithmic and —; modified —
square statistic *see:* Pearson chi- —
squared error *see:* sum of —
squared Euclidean distance location problem
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
squares *see:* Abaffi–Broyden–Spedicato algorithms for linear equations and linear least —; ABS algorithms for linear equations and linear least —; generalized nonlinear least —; Generalized total least —; least —; linear least —; method of least —; nonlinear least —; sum of —; weighted least —
squares algorithm *see:* recursive least —
squares criterion *see:* least —
squares distance function *see:* least —
squares formal orthogonal polynomials *see:* least —
squares formulation *see:* least —
squares: Newton-type methods *see:* Nonlinear least —
squares orthogonal polynomials *see:* Least —
squares problem *see:* consistent least —; generalized least —; generalized nonlinear least —; least —; perturbed least —; sparse least —; total least —; unconstrained nonlinear least —; weighted least —
squares problems *see:* Complexity and large-scale least —; Least —; Nonlinear least —
squares problems with massive data sets *see:* least —
squares, relation to Newton's method *see:* Gauss–Newton method: Least —
squares solutions *see:* least —
squares: trust region methods *see:* Nonlinear least —
SR
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms**)
SR1 quasi-Newton method
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
SR1 update
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
SSC
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms**)
SSC minimization algorithms
(90C30, 90C15, 90C99)
(*referred to in:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Inequality-constrained nonlinear optimization**; **SSC minimization algorithms for nonsmooth and stochastic optimization**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Inequality-constrained nonlinear optimization**; **SSC minimization algorithms for nonsmooth and stochastic optimization**)
SSC minimization algorithms for nonsmooth and stochastic optimization
(90C30, 90C15, 90C99)
(*referred to in:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Inequality-constrained nonlinear optimization**; **SSC minimization algorithms**)
(*refers to:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Inequality-constrained nonlinear optimization**; **SSC minimization algorithms**)
SSC-SABB algorithm
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms for nonsmooth and stochastic optimization**)
SSC-SABB algorithm *see:* nonsmooth —
sSC-SBB algorithm
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms for nonsmooth and stochastic optimization**)
SSM
[90C30]
(*see:* **Sequential simplex method**)
SSS-2
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
SSS∗
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
SSS∗ -dual
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
stability
[05C15, 05C17, 05C35, 05C69, 90C05, 90C11, 90C15, 90C22, 90C25, 90C29, 90C30, 90C31, 90C35]
(*see:* **Lovász number**; **Nondifferentiable optimization: parametric programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Suboptimal control**)
stability
[49K27, 49K40, 90C05, 90C22, 90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and**

duality; **First order constraint qualifications**; **Nondifferentiable optimization: parametric programming**; **Second order constraint qualifications**; **Semidefinite programming: optimality conditions and stability**)
stability *see:* assumption —; asymptotic —; asymptotical system —; elastic —; global structural —; Lipschitz —; phase —; radius of —; regions of —; robust —; Schur —; Semidefinite programming: optimality conditions and —; Sensitivity and stability in NLP: continuity and differential —; structural —; system —; topological —
stability analysis
[90B15, 90C15, 90C31, 90C33]
(*see:* **Approximation of extremum problems with probability functionals**; **Dynamic traffic networks**; **Sensitivity analysis of complementarity problems**)
stability analysis
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)
stability analysis *see:* robust —
stability analysis of optimization problems
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)
stability criterion *see:* sector —
stability of dynamic systems *see:* Quasidifferentiable optimization: —
stability at an equilibrium
[90B15]
(*see:* **Dynamic traffic networks**)
stability at an equilibrium *see:* asymptotical —
Stability margin
[93D09]
(*see:* **Robust control**)
stability margin *see:* multivariable —
stability margin K *see:* multivariable —
stability in mechanics *see:* smooth potentials and —
stability in NLP *see:* Sensitivity and —
stability in NLP: approximation *see:* Sensitivity and —
stability in NLP: continuity and differential stability *see:* Sensitivity and —
stability number
[05C15, 05C62, 05C69, 05C85, 68W01, 90C27, 90C59]
(*see:* **Heuristics for maximum clique and independent set**; **Optimization problems in unit-disk graphs**)
stability number *see:* weighted —
stability of optimal trajectories *see:* Turnpike theory: —
stability on parametric programming
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
stability in parametric programming *see:* structural —; topological —
stability of polytopes of polynomials *see:* Robust control: schur —
stability problem
[90B36]
(*see:* **Stochastic scheduling**)
stability problem *see:* phase —
stability, rates of convergence *see:* Stochastic integer programming: continuity —
stability of SIP *see:* structural —
stability of SIP(f,h,g) *see:* global structual —
stability of a solution set
[90C33]
(*see:* **Topological methods in complementarity theory**)
stability of a stationary point *see:* strong —
stability of a structural analysis system
[49J52, 49Q10, 74G60, 74H99, 74K99, 74Pxx, 90C90]
(*see:* **Quasidifferentiable optimization: stability of dynamic systems**)
stability of a system
[90B15]
(*see:* **Dynamic traffic networks**)
stability of a system *see:* asymptotical —
stabilization
[90C06, 90C15]
(*see:* **Stabilization of cutting plane algorithms for stochastic linear programming problems**)
Stabilization of cutting plane algorithms for stochastic linear programming problems
(90C15, 90C06)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained**

problems: convexity theory; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

stable
[90B36]
(*see:* **Stochastic scheduling**)

stable *see:* asymptotically —

stable bilinear programming
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)

stable function
[90C06]
(*see:* **Saddle point theory and optimality conditions**)

stable marriage problem
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)

stable optimal solution *see:* strongly —

stable parametric programming
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)

stable primal problem *see:* J- —

stable problem
[90C26]
(*see:* **Cutting plane methods for global optimization**)

stable set
[05C15, 05C62, 05C69, 05C85, 68W01, 90C27, 90C35, 90C59]
(*see:* **Graph coloring**; **Heuristics for maximum clique and independent set**; **Optimization problems in unit-disk graphs**)

Stable set problem: branch & cut algorithms
(90C27, 90C35)
(*refers to:* **Heuristics for maximum clique and independent set**; **Integer programming**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Lovász number**; **Simplicial pivoting algorithms for integer programming**)

stable solution
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)

stable solution *see:* Lipschitz —; strongly —

stable stationary point *see:* asymptotically —

stable without pivoting *see:* guaranteed to be —

Stackelberg game
[49-01, 49K10, 49K45, 49M37, 49N10, 90-01, 90C05, 90C20, 90C26, 90C27, 90C31, 91A65, 91B52]
(*see:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel programming: implicit function approach**)

Stackelberg game
[49-01, 49K10, 49K45, 49M37, 49N10, 90-01, 90B30, 90B50, 90C05, 90C15, 90C20, 90C26, 90C27, 90C30, 90C31, 90C33, 90C90, 91A65, 91B32, 91B52, 91B74]
(*see:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming in management**; **Stochastic bilevel programs**)

Stackelberg game *see:* von —

Stackelberg game theory
[90C30, 90C90]
(*see:* **Bilevel programming: global optimization**)

Stackelberg games *see:* von —

Stackelberg–Nash–Cournot equilibrium
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)

Stackelberg–Nash equilibrium
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)

Stackelberg problems
[90C26, 90C30, 90C31]
(*see:* **Bilevel programming: introduction, history and overview**)

Stackelberg solution *see:* feedback —

stacker Crane Problem (SCP)
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)

(staff planning) *see:* scheduling —

stage *see:* average cost per —; conceptual design —; detailed design —; discounted problem with bounded cost per —; preliminary design —; second- —

stage decision *see:* first- —; second- —

stage decisions *see:* first- —; second- —

stage feasibility set *see:* second- —

stage IM *see:* single- —

stage inventory management models *see:* single —

stage-length *see:* variable —

stage model *see:* linear two- —

stage problem *see:* average cost per —

stage problems *see:* average cost per —; Dynamic programming: average cost per —

stage stochastic linear program *see:* two- —

stage stochastic program with recourse *see:* two- —

stage stochastic programming *see:* two- —

stage stochastic programming models *see:* two- —

stage stochastic programming problem *see:* dynamic two- —

stage stochastic programming: quasigradient method *see:* Two- —

stage stochastic programs *see:* multy- —

stage stochastic programs with recourse *see:* L-shaped method for two- —; two- —

stage stochastic programs with simple integer recourse *see:* two- —
staircase arc cost function
[90B10]
(*see:* **Piecewise linear network flow problems**)
staircase cost
[90B80, 90C11]
(*see:* **Facility location with staircase costs**)
staircase cost function
[90B80, 90C11]
(*see:* **Facility location with staircase costs**)
staircase costs *see:* convex piecewise linearization in facility location problems with —; Facility location with —; heuristics of facility location problems with —; linearization in facility location problems with —; solution of facility location problems with —
stalling
[90C60]
(*see:* **Complexity of degeneracy**)
stalling
[90C60]
(*see:* **Complexity of degeneracy**)
stamped models *see:* time- —
standard
[90C31, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions)
standard *see:* CG- —
standard auction algorithm *see:* modified —
standard determinant expansion of a matrix
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
standard form
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
standard form *see:* constraints in —; linear optimization problem in —; matrix in —
standard function
[90C26, 90C30]
(*see:* **Bounding derivative ranges**)
standard greedy form
[90B10, 90B80, 90C35]
(*see:* **Network location: covering problems**)
standard methods *see:* unbounded controls and non —
standard for minimizing q *see:* CG- —
standard mollifier
[57R12, 90C31, 90C34]
(*see:* **Smoothing methods for semi-infinite optimization**)
standard moment problem
[28-XX, 49-XX, 60-XX]
(*see:* **General moment optimization problems**)
standard monomials
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
standard pair decomposition of a monomial ideal
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
standard pair of a monomial ideal
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
standard part map
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
standard quadratic optimization problem
[90C20]
(*see:* **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
Standard quadratic optimization problems: algorithms
(90C20)
(*referred to in:* **Complexity theory: quadratic programming**; **Linear ordering problem**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
(*refers to:* **Complexity theory: quadratic programming**; **Interval analysis: eigenvalue bounds of interval matrices**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Semidefinite programming: optimality conditions and stability**; **Simplicial decomposition**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
Standard quadratic optimization problems: applications
(90C20)
(*referred to in:* **Complexity theory: quadratic programming**; **Linear ordering problem**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: theory**)
(*refers to:* **Complexity theory: quadratic programming**; **Portfolio selection and multicriteria analysis**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Simplicial decomposition**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: theory**)
Standard quadratic optimization problems: theory
(90C20)
(*referred to in:* **Complexity theory: quadratic programming**; **Linear ordering problem**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an ellipsoid**; **Reverse convex optimization**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**)
(*refers to:* **αBB algorithm**; **Complexity theory: quadratic programming**; **D.C. programming**; **Interior point methods for semidefinite programming**; **Quadratic assignment problem**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**; **Quadratic programming with bound constraints**; **Quadratic programming over an**

ellipsoid; **Reverse convex optimization**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**)
standard SD problem
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
standard simplex
[65K05, 90C20, 90C30]
(*see:* **Bisection global optimization methods**; **Standard quadratic optimization problems: algorithms**; **Standard quadratic optimization problems: applications**; **Standard quadratic optimization problems: theory**)
standard state
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
standard traffic equilibrium problem
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
Stanley–Reisner ideal
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
star
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
star
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
star *see:* binary —; double —; spectroscopic visual binary —; visual binary —
star astronomy *see:* Global optimization in binary —
star cluster
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
star network
[05A18, 05D15, 68M07, 68M10, 68Q25, 68R05]
(*see:* **Maximum partition matching**)
star-shaped set
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
start *see:* best —
start interior-point algorithm *see:* infeasible- —
start state of a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
start temperature
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
start-ups *see:* well-defined —
starting region *see:* safe —
state
[49K20, 49M99, 90C55]
(*see:* **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
state *see:* accessible —; equation of —; ground —; optimal steady —; standard —
State of the art in modeling agricultural systems
state constraint
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
state constraints
[49K05, 49K10, 49K15, 49K20, 93-XX]
(*see:* **Direct search Luus—Jaakola optimization procedure**; **Duality in optimal control with first order differential equations**)
state distribution density *see:* steady- —
state equations
[49K05, 49K10, 49K15, 49K20, 49M99, 90C55]
(*see:* **Duality in optimal control with first order differential equations**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
state equations *see:* generalized —
state feedback *see:* incomplete —
state inequality constraint
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
state of knowledge
[94A17]
(*see:* **Jaynes' maximum entropy principle**)
state Markov chain *see:* finite- —; stationary- —
state polyhedron
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
state polytope
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
state problem
[49J20, 49J52]
(*see:* **Shape optimization**)
state problem *see:* regularizing —
state relation
[49J20, 49J52]
(*see:* **Shape optimization**)
state space graphs *see:* searching —
state space search algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
state space search algorithm *see:* optimal —; recursive —; synchronized distributed —
state of a system
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
state-task-network
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
state of a Turing machine
[90C60]
(*see:* **Complexity theory**)
state of a Turing machine *see:* accepting —; control —; final —; start —
state vector
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
states *see:* recurrent class of —; transient class of —
states of plants *see:* tracing the —

static
[90B06]
(*see:* **Vehicle routing**)
static deterministic problem
[90B06]
(*see:* **Vehicle routing**)
static/dynamic service needs
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
static load balancing
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)
static model
[90B80, 90B85]
(*see:* **Warehouse location problem**)
static multifacility *see:* discrete single-commodity single-criterion uncapacitated —
static problems *see:* Hemivariational inequalities: —
Static resource constrained project scheduling
Static stochastic programming models
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
Static stochastic programming models: conditional expectations
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of**

discrete distributions; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

static TS
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)

station *see:* arr- —; arrival- —; dep- —; departure- —

stationarity
[49K27, 58C20, 58E30, 65K05, 90C26, 90C33, 90C34, 90C48]
(*see:* **Adaptive convexification in semi-infinite optimization**; **Nonsmooth analysis: Fréchet subdifferentials**)

stationarity *see:* Nonsmooth analysis: weak —; weak —

stationarity conditions *see:* KKT —

stationary
[05C60, 05C69, 37B25, 49L20, 49L99, 49M07, 49M10, 65K, 90C05, 90C06, 90C20, 90C22, 90C25, 90C27, 90C30, 90C31, 90C33, 90C34, 90C35, 90C40, 90C59, 91A22]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: undiscounted problems**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Replicator dynamics in combinatorial optimization**; **Semidefinite programming: optimality conditions and stability**; **Semi-infinite programming: discretization methods**; **Spectral projected gradient methods**)

stationary *see:* inf- —

stationary point
[49M29, 58C20, 58E30, 65K05, 65K10, 90C06, 90C26, 90C31, 90C34, 90C39, 90C46, 90C48, 90C90, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Local attractors for gradient-related descent iterations**; **Nonsmooth analysis: weak stationarity**; **Parametric global optimization: sensitivity**; **Second order optimality conditions for nonlinear optimization**; **Variational inequalities: projected dynamical system**)

stationary point
[65K10, 90C26, 90C39, 90C90]
(*see:* **Second order optimality conditions for nonlinear optimization**; **Variational inequalities: projected dynamical system**)

stationary point *see:* ∞- —; asymptotically stable —; Dini sup- —; ε- —; Hadamard ∞- —; Hadamard sup- —; inf- —; isolated —; regular —; strong stability of a —; sup- —

stationary points
[90C20]
(*see:* **Standard quadratic optimization problems: algorithms**)

stationary points *see:* inf- —

stationary policy
[49L20, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**)

stationary regime
[60J05, 90C15]
(*see:* **Derivatives of markov processes and their simulation**)

stationary-state Markov chain
[49L99]
(*see:* **Dynamic programming: average cost per stage problems**)

stations *see:* Automatic differentiation: geometry of satellites and tracking —; one-hop neighboring —; tracking —

statistic *see:* generalized single cluster —; Goodman–Kruskal τ_b —; Mann–Whitney —; Pearson chi-square —; Rand —

statistical analysis *see:* exploratory —

statistical classification
[62H30, 90C11]
(*see:* **Statistical classification: optimization approaches**)

statistical classification
[62H30, 90C11]
(*see:* **Statistical classification: optimization approaches**)

Statistical classification: optimization approaches
(62H30, 90C11)
(*referred to in:* **Linear programming models for classification**; **Mixed integer classification problems**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)
(*refers to:* **Linear programming models for classification**; **Mixed integer classification problems**; **Optimization in boolean classification problems**; **Optimization in classifying text documents**)

Statistical convergence and turnpike theory
(40A05, 49J24)
(*referred to in:* **Turnpike theory: stability of optimal trajectories**)
(*refers to:* **Turnpike theory: stability of optimal trajectories**)

statistical inference *see:* order restricted —

statistical method *see:* nonparametric —

statistical models
[90C30]
(*see:* **Global optimization based on statistical models**)

statistical models *see:* Global optimization based on —

statistical pattern classification
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)

statistical pattern recognition
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
statistical procedures
[90C26, 90C30]
(*see:* **Forecasting**)
statistical representation of cutting plane coefficients
[90C06, 90C15]
(*see:* **Stochastic linear programming: decomposition and cutting planes**)
statistics
[90C26, 90C30]
(*see:* **Forecasting**)
statistics *see:* algebraic —; higher-order —; Signal processing with higher order —
status of the wells *see:* operational —
steady state *see:* optimal —
steady-state distribution density
[60G35, 65K05]
(*see:* **Differential equations and global optimization**)
steamer problem *see:* tramp —
steep directions
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
steepest ascent *see:* rate of —; rule of —
steepest ascent direction
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
steepest ascent direction
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
steepest ascent direction *see:* Dini —; Hadamard —
steepest descent
[46N10, 68T20, 68T99, 90-00, 90C27, 90C47, 90C59]
(*see:* **Metaheuristics**; **Nondifferentiable optimization**)
steepest descent *see:* ε- —; method of —; rate of —
steepest descent algorithm
[60G35, 62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Differential equations and global optimization**; **Monte-Carlo simulations for stochastic optimization**)
steepest-descent direction
[90C05, 90C22, 90C25, 90C30, 90C51, 90Cxx]
(*see:* **Interior point methods for semidefinite programming**; **Quasidifferentiable optimization: optimality conditions**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
steepest descent direction
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
steepest descent direction *see:* Dini —; Hadamard —
steepest descent method
[49M29, 65K10, 90C06, 90C30]
(*see:* **Cost approximation algorithms**; **Large scale unconstrained optimization**; **Local attractors for gradient-related descent iterations**)
steepest descent vector
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
steepest edge simplex method
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
Stein estimators
[91B28]
(*see:* **Portfolio selection: markowitz mean-variance model**)
Stein estimators *see:* James– —
Steiner arborescence
[90C27]
(*see:* **Steiner tree problems**)
Steiner arborescence *see:* minimum —
Steiner arborescence tree *see:* rectilinear —
Steiner graphical traveling salesman problem
[90B20]
(*see:* **General routing problem**)
Steiner minimal tree
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
Steiner minimal tree
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
Steiner minimal tree problem
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
Steiner minimum tree
[90C27]
(*see:* **Steiner tree problems**)
Steiner network *see:* multiphase —
Steiner nodes
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
Steiner point
[90C27]
(*see:* **Steiner tree problems**)
Steiner points
[05C05, 05C40, 68Q20, 68R10, 90C27, 90C35]
(*see:* **Network design problems**; **Optimal triangulations**; **Steiner tree problems**)
Steiner points *see:* Steiner tree problem with minimum number of —
Steiner problem
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
Steiner problems *see:* multiphase —
Steiner ratio
[05C05, 05C40, 05C85, 68Q25, 68R10, 90B80, 90C35]
(*see:* **Bottleneck steiner tree problems**; **Network design problems**)
Steiner ratio
[90C27]
(*see:* **Steiner tree problems**)
Steiner ratio *see:* bottleneck —; Euclidean —; k- —
Steiner ratio in Banach spaces
[90C27]
(*see:* **Steiner tree problems**)
Steiner ratio of biomolecular structures
(90C27)

Steiner tree
[05-04, 05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Evolutionary algorithms in combinatorial optimization**; **Optimization problems in unit-disk graphs**)
Steiner tree
[90C27]
(*see:* **Steiner tree problems**)
Steiner tree *see:* full —; min-max —; rectilinear —
Steiner tree problem with minimum number of Steiner points
[90C27]
(*see:* **Steiner tree problems**)
Steiner tree problems
(90C27)
(*referred to in:* **Auction algorithms**; **Bottleneck steiner tree problems**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)

(*refers to:* **Auction algorithms**; **Bottleneck steiner tree problems**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Traffic network equilibrium**)
steiner tree problems *see:* Bottleneck —
Steiner trees *see:* bottleneck —; variations of —
Steiner triangulation *see:* minimum weight —
Steiner–Weber problem
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
Steinhauser function *see:* Kreisselmeier– —
step *see:* analysis —; bounding —; branching —; computational —; coordination —; decomposition —; descent —; escape —; fathoming —; insertion —; long serious —; Newton —; null —; preconditioning —; relaxation —; selection —; short serious —; time- —; trial —
step case of the trust region problem *see:* Newton —
step function
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
step Gauss–Newton method *see:* full- —
step size
[90C05, 90C22, 90C25, 90C30, 90C51]
(*see:* **Interior point methods for semidefinite programming**)
step-size rule *see:* divergent series —
step superlinear *see:* 2- —
steplength
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
steplength *see:* compute the —; compute a safeguarded new trial —
steplength rule *see:* Armijo —; divergent series —
steps *see:* acceleration —; Automatic differentiation: calculation of Newton —; average number of pivot —; expected number of pivot —
stepsize
[90C30]
(*see:* **Frank–Wolfe algorithm**)
Stewart reorthogonalized Gram–Schmidt algorithm *see:* Daniel–Gragg–Kaufmann– —
sTF
[90B35, 90C11, 90C30]
(*see:* **Robust optimization: mixed-integer linear programs**)
Stiefel algorithm *see:* Hestenes– —
Stiemke theorem
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
Stiemke transposition theorem
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
stiff problem
[65Fxx]
(*see:* **Least squares problems**)
stiffness matrix
[49M37, 65K05, 90C30]
(*see:* **Structural optimization**)
stiffness matrix
[49M37, 65K05, 90C30]
(*see:* **Structural optimization**)
sTO
[90B35, 90C11, 90C30]
(*see:* **Robust optimization: mixed-integer linear programs**)
STO problem *see:* nested —
stochastic approach
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
stochastic approach to optimization in water resources
[90C30, 90C35]
(*see:* **Optimization in water resources**)
stochastic approximation
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
stochastic approximation
[90C15]
(*see:* **Extremum problems with probability functions: kernel type solution methods**)
stochastic bilevel program
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
Stochastic bilevel programs
(90C15, 90C26, 90C33)
(*referred to in:* **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: global optimization**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel**

programming: optimality conditions and duality; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**)
(*refers to:* **Bilevel fractional programming**; **Bilevel linear programming**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Bilevel programming**; **Bilevel programming: applications**; **Bilevel programming: applications in engineering**; **Bilevel programming: implicit function approach**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**; **Bilevel programming: optimality conditions and duality**; **Multilevel methods for optimal design**; **Multilevel optimization in mechanics**)
stochastic bilevel programs *see:* algorithms for —
stochastic branch and bound
[90C15, 90C27]
(*see:* **Discrete stochastic optimization**)
stochastic combinatorial optimization
[90C15, 90C27]
(*see:* **Discrete stochastic optimization**)
stochastic counterpart method
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
stochastic decomposition
[62F12, 65C05, 65K05, 90C15, 90C26, 90C31, 90C33]
(*see:* **Monte-Carlo simulations for stochastic optimization**; **Stochastic bilevel programs**; **Two-stage stochastic programming: quasigradient method**)
stochastic decomposition
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
stochastic decomposition algorithm
[90C06, 90C15]
(*see:* **Stochastic linear programming: decomposition and cutting planes**)
stochastic decomposition algorithm *see:* regularized —
stochastic differential equation
[60G35, 65K05]
(*see:* **Differential equations and global optimization**)
stochastic discretization procedure
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming: discretization methods**)
stochastic dynamic optimization problem
[90C15]
(*see:* **Stochastic quasigradient methods: applications**)
stochastic dynamic programming
[90B05, 90B06]
(*see:* **Global supply chain models**)
stochastic dynamic programming
[90B05, 90B06]
(*see:* **Global supply chain models**)
stochastic dynamic systems
[90C15]
(*see:* **Stochastic quasigradient methods: applications**)
Stochastic facility location model
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)
stochastic flexibility
[90C90]
(*see:* **Chemical process planning**)
stochastic geometry
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
stochastic geometry
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
stochastic global optimization
[92B05]
(*see:* **Genetic algorithms**)
Stochastic global optimization: stopping rules
(65K05, 90C26, 90C30, 65Cxx, 65C30, 65C40, 65C50, 65C60)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization: hit and run methods**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: two-phase methods**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Concave programming**; **D.C. programming**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization: hit and run methods**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: two-phase methods**)
Stochastic global optimization: two-phase methods
(65K05, 90C26, 90C30, 65Cxx, 65C30, 65C40, 65C50, 65C60)
(*referred to in:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization: hit and run methods**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**)
(*refers to:* **Adaptive simulated annealing and its application to protein folding**; **Bayesian global optimization**; **Concave programming**; **D.C. programming**; **Genetic algorithms for protein structure prediction**; **Global optimization based on statistical models**; **Global optimization: hit and run methods**; **Monte-Carlo simulated annealing in protein folding**; **Packet annealing**; **Random search methods**; **Simulated annealing**; **Simulated annealing methods in protein folding**; **Stochastic global optimization: stopping rules**)
stochastic integer program with recourse
[90C10, 90C15]
(*see:* **Stochastic integer programs**)
stochastic integer programming
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)

Stochastic integer programming: continuity, stability, rates of convergence
(90C15, 90C11, 90C31)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Discrete stochastic optimization**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Multistage stochastic programming: barycentric approximation**; **Parametric mixed integer nonlinear optimization**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Set covering, packing and partitioning problems**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Simplicial pivoting algorithms for integer programming**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Time-dependent traveling salesman problem**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Set covering, packing and partitioning problems**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Simplicial pivoting algorithms for integer programming**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Time-dependent traveling salesman problem**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

Stochastic integer programs
(90C15, 90C10)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Multistage stochastic programming: barycentric approximation**; **Parametric mixed integer nonlinear optimization**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity**

theory; **Set covering, packing and partitioning problems**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Simplicial pivoting algorithms for integer programming**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Time-dependent traveling salesman problem**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **Fractional combinatorial optimization**; **General moment optimization problems**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Set covering, packing and partitioning problems**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Simplicial pivoting algorithms for integer programming**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Time-dependent traveling salesman problem**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
stochastic linear optimization problems
[90C15, 90C27]
(*see:* **Discrete stochastic optimization**)
stochastic linear program *see:* two-stage —
stochastic linear program with recourse
[90C10, 90C15]
(*see:* **L-shaped method for two-stage stochastic programs with recourse**; **Stochastic integer programs**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Two-stage stochastic programs with recourse**)
stochastic linear programming
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
Stochastic linear programming: decomposition and cutting planes
(90C15, 90C06)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Decomposition principle of linear programming**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **Generalized benders decomposition**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **MINLP: generalized cross decomposition**; **MINLP: logic-based methods**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Successive quadratic programming: decomposition methods**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Decomposition principle of linear programming**; **Discretely distributed stochastic programs:**

descent directions and efficient points; **Extremum problems with probability functions: kernel type solution methods**; **Generalized benders decomposition**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **MINLP: generalized cross decomposition**; **MINLP: logic-based methods**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Successive quadratic programming: decomposition methods**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

stochastic linear programming problems *see:* Stabilization of cutting plane algorithms for —

Stochastic linear programs with recourse and arbitrary multivariate distributions
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

stochastic local search
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)

stochastic matrix
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)

stochastic matrix *see:* doubly —

stochastic methods
[90C26, 90C90]
(*see:* **Global optimization: hit and run methods**)

stochastic (mixed-)integer programming
[90C11, 90C15]
(*see:* **Stochastic programming with simple integer recourse**)

stochastic model
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)

stochastic mollifier quasigradient
[90C15]
(*see:* **Stochastic quasigradient methods in minimax problems**)

stochastic network problem
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)

Stochastic network problems: massively parallel solution
(90B15, 68W10, 90C06, 90C30)
(*referred to in:* **Approximation of extremum problems with**

probability functionals; **Approximation of multivariate probability integrals**; **Asynchronous distributed optimization algorithms**; **Auction algorithms**; **Automatic differentiation: parallel computation**; **Communication network assignment problem**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Dynamic traffic networks**; **Equilibrium networks**; **Extremum problems with probability functions: kernel type solution methods**; **Generalized networks**; **General moment optimization problems**; **Heuristic search**; **Load balancing for parallel optimization techniques**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Multistage stochastic programming: barycentric approximation**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Parallel heuristic search**; **Piecewise linear network flow problems**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Shortest path tree algorithms**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Steiner tree problems**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Survivable networks**; **Traffic network equilibrium**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Asynchronous distributed optimization algorithms**; **Auction algorithms**; **Automatic differentiation: parallel computation**; **Communication network assignment problem**; **Directed tree networks**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Extremum problems with probability functions: kernel type solution methods**; **Generalized networks**; **General moment optimization problems**; **Heuristic search**; **Interval analysis: parallel methods for global optimization**; **Load balancing for parallel optimization techniques**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Maximum flow problem**; **Minimum cost flow problem**; **Multistage stochastic programming: barycentric approximation**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Parallel computing: complexity classes**; **Parallel computing: models**; **Parallel heuristic search**; **Piecewise linear network flow problems**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Shortest path tree algorithms**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Steiner tree problems**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Survivable networks**; **Traffic network equilibrium**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
stochastic neural network
[90C27, 90C30]
(*see:* **Neural networks for combinatorial optimization**)
Stochastic optimal stopping: numerical methods
Stochastic optimal stopping: problem formulations
stochastic optimization
[90C15]
(*see:* **Derivatives of probability and integral functions: general theory and examples**)
stochastic optimization
[60J05, 90C15, 90C27, 90C29, 90C30, 90C99]
(*see:* **Derivatives of markov processes and their simulation**; **Derivatives of probability measures**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Discrete stochastic optimization**; **SSC minimization algorithms for nonsmooth and stochastic optimization**)
stochastic optimization *see:* Discrete —; Monte-Carlo simulations for —; SSC minimization algorithms for nonsmooth and —
stochastic problems
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
stochastic process
[60G35, 65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Differential equations and global optimization**; **Unconstrained optimization in neural network training**)
stochastic process nonanticipative with respect to a filtration
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
stochastic program
[68W10, 90B15, 90C06, 90C15, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**; **Stochastic programming: nonanticipativity and lagrange multipliers**)

stochastic program
[90C15, 90C29]
(*see:* **Discretely distributed stochastic programs: descent directions and efficient points**)
stochastic program *see:* multiperiod —; multistage —
stochastic program with recourse
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
stochastic program with recourse *see:* two-stage —
stochastic programming
[01A99, 62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Approximation of extremum problems with probability functionals**; **History of optimization**; **Monte-Carlo simulations for stochastic optimization**; **Stochastic programming: parallel factorization of structured matrices**; **Two-stage stochastic programs with recourse**)
stochastic programming
[49M25, 62C20, 62F12, 65C05, 65K05, 68W10, 90-08, 90B10, 90B15, 90C05, 90C06, 90C08, 90C10, 90C15, 90C26, 90C27, 90C30, 90C31, 90C33, 90C35, 90C90, 91B28]
(*see:* **Chemical process planning**; **Financial optimization**; **L-shaped method for two-stage stochastic programs with recourse**; **Monte-Carlo simulations for stochastic optimization**; **Multistage stochastic programming: barycentric approximation**; **Operations research and financial markets**; **Preprocessing in stochastic programming**; **Simple recourse problem**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Stochastic bilevel programs**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming: parallel factorization of structured matrices**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
stochastic programming *see:* bibliography of —; multistage —; Preprocessing in —; probabilistic constrained —; two-stage —
stochastic programming: barycentric approximation *see:* Multistage —
Stochastic programming: minimax approach
(90C15, 62C20)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Minimax: directional differentiability**; **Minimax theorems**; **Multistage stochastic programming: barycentric approximation**; **Nondifferentiable optimization: minimax problems**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Minimax: directional differentiability**; **Minimax theorems**; **Multistage stochastic programming: barycentric approximation**; **Nondifferentiable optimization: minimax problems**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
stochastic programming models *see:* Static —; two-stage —

stochastic programming models: conditional expectations *see:* Static —
Stochastic programming models: random objective (90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
Stochastic programming: nonanticipativity and lagrange multipliers (90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models:**

random objective; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
Stochastic programming: parallel factorization of structured matrices
(90C15)
stochastic programming problem
[90C15]
(*see:* **Static stochastic programming models**)
stochastic programming problem *see:* dynamic two-stage —
stochastic programming: quasigradient method *see:* Two-stage —
Stochastic programming with simple integer recourse
(90C15, 90C11)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
stochastic programs *see:* discretely distributed —; multy-stage —
stochastic programs: descent directions and efficient points *see:* Discretely distributed —
stochastic programs with recourse *see:* L-shaped method for two-stage —; two-stage —
Stochastic programs with recourse: upper bounds
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
stochastic programs with simple integer recourse *see:* two-stage —
stochastic quasigradient
[62F12, 65C05, 65K05, 90C15, 90C31]
(*see:* **Monte-Carlo simulations for stochastic optimization**)
Stochastic quasigradient methods
(90C15)
(*referred to in:* **Derivatives of markov processes and their**

simulation; **Derivatives of probability measures**; **Stochastic quasigradient methods: applications**; **Stochastic quasigradient methods in minimax problems**; **Two-stage stochastic programming: quasigradient method**)
(*refers to:* **Stochastic programming: nonanticipativity and lagrange multipliers**)

stochastic quasigradient methods
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**; **Stochastic quasigradient methods: applications**)

Stochastic quasigradient methods: applications
(90C15)
(*referred to in:* **Stochastic quasigradient methods in minimax problems**; **Two-stage stochastic programming: quasigradient method**)
(*refers to:* **Stochastic quasigradient methods**)

Stochastic quasigradient methods in minimax problems
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Bilevel optimization: feasibility test and flexibility index**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Minimax: directional differentiability**; **Minimax theorems**; **Multistage stochastic programming: barycentric approximation**; **Nondifferentiable optimization: minimax problems**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)
(*refers to:* **Minimax theorems**; **Nondifferentiable optimization: minimax problems**; **Stochastic quasigradient methods**; **Stochastic quasigradient methods: applications**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**)

stochastic Quasigradient (SQG) methods
[90C15]
(*see:* **Stochastic quasigradient methods**)
stochastic quasigradients
[90C15]
(*see:* **Stochastic quasigradient methods**)
Stochastic scheduling
(90B36)
(*referred to in:* **Job-shop scheduling problem**; **MINLP: design and scheduling of batch processes**; **Vehicle scheduling**)
stochastic search method
[90C27, 90C90]
(*see:* **Simulated annealing**)
stochastic shortest path
[49L20, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**)
stochastic shortest path problem
[49L20, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**)
stochastic shortest path problems
[49L20, 49L99, 90C39, 90C40]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: infinite horizon problems, overview**)
stochastic shortest path problems *see:* Dynamic programming: —
stochastic simulated annealing
[90C15, 90C27]
(*see:* **Discrete stochastic optimization**)
stochastic solution *see:* value of —
stochastic transportation and location problem
[90B80, 90C11]
(*see:* **Stochastic transportation and location problems**)
Stochastic transportation and location problems
(90B80, 90C11)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **Minimum concave transportation problems**; **MINLP: application in facility location-allocation**; **Motzkin transposition theorem**; **Multifacility and restricted location problems**; **Multi-index transportation problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Frank–Wolfe algorithm**; **Global optimization in Weber's problem with attraction and repulsion**; **Minimum concave transportation problems**; **MINLP: application in facility location-allocation**; **Motzkin**

transposition theorem; **Multifacility and restricted location problems**; **Multi-index transportation problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Voronoi diagrams in facility location**; **Warehouse location problem**)
stochastic transportation problem
[90B80, 90C11]
(*see:* **Stochastic transportation and location problems**)
stochastic travel times
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
stochastic vehicle routing problem
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
Stochastic vehicle routing problems
(90C15, 90C10)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **General routing problem**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Two-stage stochastic programs with recourse**; **Vehicle routing**; **Vehicle scheduling**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **General routing problem**; **Integer programming: branch and cut algorithms**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programming with simple integer recourse**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Two-stage stochastic programming: quasigradient method**; **Two-stage stochastic programs with recourse**; **Vehicle routing**; **Vehicle scheduling**)
stochasticglobal optimization
[92B05]
(*see:* **Genetic algorithms**)
stochasticity
[90C30, 90C35]
(*see:* **Optimization in water resources**)
stock *see:* cutting —; echelon —
stock problem *see:* cutting- —
Stoica method
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
stoichiometric form of KT conditions
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
stoichiometry *see:* estimation of reaction rates and —
Stokes code *see:* Reynolds-averaged Navier– —
stopping *see:* optimal —
stopping criteria *see:* Dykstra's algorithm and robust —
stopping criterion
[90C30]
(*see:* **Frank–Wolfe algorithm**)
stopping: numerical methods *see:* Stochastic optimal —
stopping: problem formulations *see:* Stochastic optimal —
stopping rule
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
stopping rule *see:* Bayesian —
stopping rules *see:* Stochastic global optimization: —
stopping times *see:* regenerative —
storage *see:* unlimited intermediate —; variable- —
storage algorithm *see:* variable- —
storage capacity *see:* nodes with water —
storage entities *see:* multipurpose —
storage equation
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)

storage plant
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
storage plant
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
storage plants
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
STP
[90B80, 90C11]
(*see:* **Stochastic transportation and location problems**)
STP-MSP
[90C27]
(*see:* **Steiner tree problems**)
StQP
[90C20]
(*see:* **Standard quadratic optimization problems: applications**)
strain-displacement compatibility equations
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
strain tensor
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
stranded chain *see:* two- —
strategic design models
[90-02]
(*see:* **Operations research models for supply chain management and design**)
strategic design of a supply chain
[90-02]
(*see:* **Operations research models for supply chain management and design**)
strategic supply chain management
[90-02]
(*see:* **Operations research models for supply chain management and design**)
strategies
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
strategies *see:* active set —; default —; evolutionary —; solution —
strategies for dynamic systems *see:* Optimization —
strategies and performances *see:* Volume computation for polytopes: —
strategy
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
strategy *see:* active set —; binary search-Hansel chains question-asking —; black-box —; bold —; branch and bound —; cluster first-schedule second —; convexification/relaxation —; evolution —; game of —; Goldfarb–Idnani active set —; Markov —; projection-restriction —; pure —; pure trust region —; question-asking —; sequential Hansel chains question-asking —; TR —
strategy equilibrium *see:* memory —
strategy for interval-Newton method in deterministic global optimization *see:* LP —
strategy for model structure refinement *see:* incremental —
strategy Nash equilibrium *see:* memory —
strategy pattern
(*see:* **State of the art in modeling agricultural systems**)
stream *see:* lean —; rich —
stream arcs *see:* natural —
street planning
[90C35]
(*see:* **Multicommodity flow problems**)
strengthen triangle inequality
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
stress
[65K05, 90C27, 90C30, 90C57, 91C15]
(*see:* **Optimization-based visualization**)
stress *see:* equilibrium —; normalized —; s- —
stress equilibrium equations
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
stressed design *see:* fully —
strict
[58C20, 58E30, 90C46, 90C48]
(*see:* **Nonsmooth analysis: weak stationarity**)
strict complementarity
[49M37, 65K05, 90C22, 90C25, 90C30, 90C31]
(*see:* **Inequality-constrained nonlinear optimization**; **Semidefinite programming: optimality conditions and stability**)
strict complementarity
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
strict complementarity slackness
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
strict complementarity slackness condition
[65K10, 90C31]
(*see:* **Sensitivity analysis of variational inequality problems**)
strict complementary slackness
[90C31, 90C34]
(*see:* **Semi-infinite programming: second order optimality conditions**; **Sensitivity and stability in NLP: continuity and differential stability**)
strict efficiency
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
strict efficiency *see:* local —
strict feasibility condition
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
strict local maximizer
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)

strict local maximum point
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
strict local minimizer
[49M29, 65K05, 65K10, 90C06, 90C31, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Local attractors for gradient-related descent iterations**; **Sensitivity and stability in NLP: continuity and differential stability**)
strict local minimum
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
strict local minimum point
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
strict local optimizer
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
strict monotonicity
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
strict monotonicity
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
strict monotonicity *see:* local —
strict relative minimum
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
strict-set-contraction
[90C33]
(*see:* **Order complementarity**)
strict TS
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
strictly antisymmetric relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
strictly complementary conditions
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
strictly complementary solution
[15A39, 90C05]
(*see:* **Homogeneous selfdual methods for linear programming**; **Tucker homogeneous systems of linear relations**)
strictly convex function
[90C26]
(*see:* **Generalized monotone single valued maps**)
strictly copositive matrix
[65K05, 90C20]
(*see:* **Quadratic programming with bound constraints**)
strictly differentiable
[49K27, 58C20, 58E30, 90C46, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**; **Nonsmooth analysis: weak stationarity**)
strictly efficient point
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
strictly efficient point *see:* local —
strictly feasible set
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
strictly monotone
[47J20, 49J40, 65K10, 90C33]
(*see:* **Solution methods for multivalued variational inequalities**)
strictly monotone function
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
strictly monotone function *see:* locally —
strictly monotone map
[90C26]
(*see:* **Generalized monotone single valued maps**)
strictly monotone operator
[90C26]
(*see:* **Generalized monotone multivalued maps**)
strictly proper
(*see:* **Bayesian networks**)
strictly pseudoconvex
[90C26]
(*see:* **Generalized monotone multivalued maps**; **Generalized monotone single valued maps**)
strictly pseudoconvex function
[90C06, 90C25, 90C35]
(*see:* **Simplicial decomposition algorithms**)
strictly pseudomonotone map
[90C26]
(*see:* **Generalized monotone single valued maps**)
strictly pseudomonotone operator
[90C26]
(*see:* **Generalized monotone multivalued maps**)
strictly quasiconvex function
[90C26]
(*see:* **Generalized monotone single valued maps**)
strictly quasimonotone map
[90C26]
(*see:* **Generalized monotone single valued maps**)
strictly quasimonotone operator
[90C26]
(*see:* **Generalized monotone multivalued maps**)
strictly repetitive
(*see:* **Bayesian networks**)
string *see:* path- —
strong branching
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
strong cutting plane
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
strong dual
[90C10, 90C46]
(*see:* **Integer programming duality**)
strong duality
[90C06, 90C30]
(*see:* **Duality for semidefinite programming**; **Lagrangian**

duality: BASICS; **Saddle point theory and optimality conditions**)
strong duality result
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
strong duality theorem
[05B35, 49-XX, 65K05, 90-XX, 90C05, 90C20, 90C33, 93-XX]
(*see:* **Criss-cross pivoting rules**; **Duality theory: monoduality in convex optimization**)
strong homomorphism
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
strong homomorphism *see:* very —
strong local minimizer
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
strong local minimum
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
strong monotonicity
[65K10, 65M60, 91B06, 91B60]
(*see:* **Oligopolistic market equilibrium**; **Variational inequalities: geometric interpretation, existence and uniqueness**)
strong monotonicity
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
strong monotonicity *see:* local —
strong NP-complete completeness
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
strong NP-completeness
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
strong NP-completeness
[68Q25, 90C60]
(*see:* **NP-complete problems and proof methodology**)
strong operator topology
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
strong perfect graph theorem
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
strong product
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
strong relative minimum
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
strong second order sufficiency
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
strong second order sufficient condition
[90C31]
(*see:* **Sensitivity and stability in NLP: continuity and differential stability**)
strong second order sufficient condition *see:* general —
strong semismoothness
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
Strong Slater CQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
strong stability of a stationary point
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
strong and weak duality
[90C30]
(*see:* **Duality for semidefinite programming**)
strongly active constraints
[49M20, 90C11, 90C30]
(*see:* **Generalized outer approximation**)
strongly connected components of a digraph
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
strongly connected digraph
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
strongly connected network
[90C35]
(*see:* **Minimum cost flow problem**)
strongly convex
[90C30]
(*see:* **Frank–Wolfe algorithm**)
strongly convex
[90C30]
(*see:* **Frank–Wolfe algorithm**)
strongly determined variable
[65H20, 65K05, 90-01, 90B40, 90C10, 90C27, 90C35, 94C15]
(*see:* **Greedy randomized adaptive search procedures**)
strongly linearly monotonic over
[90C05]
(*see:* **Extension of the fundamental theorem of linear programming**)
strongly monotone
[47J20, 49J40, 65K10, 90C30, 90C33]
(*see:* **Cost approximation algorithms**; **Implicit lagrangian**; **Solution methods for multivalued variational inequalities**)
strongly monotone function
[65K10, 65M60]
(*see:* **Variational inequalities: geometric interpretation, existence and uniqueness**)
strongly monotone function *see:* locally —
strongly monotonic at *see:* locally —
strongly nonsingular matrix
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
strongly NP-hard
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
strongly polynomial algorithm
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)

strongly polynomial algorithm
[90C11, 90C15]
(*see:* **Stochastic programming with simple integer recourse**)
strongly polynomial solution
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
strongly polynomial time
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 65K05, 68Q, 68Q05, 68Q10, 68Q25, 68R, 68U, 68W, 90B, 90C, 90C05, 90C25, 90C26]
(*see:* **Convex discrete optimization**; **Information-based complexity and information-based optimization**)
strongly polynomial time
[90C60]
(*see:* **Complexity theory: quadratic programming**)
strongly polynomial time algorithm
[90C35]
(*see:* **Minimum cost flow problem**)
strongly polynomial time problem
[90C60]
(*see:* **Complexity theory: quadratic programming**)
strongly positive definite matrix
[65K10, 65M60]
(*see:* **Variational inequalities**)
strongly regular matrix
[15A99, 65G20, 65G30, 65G40, 90C26]
(*see:* **Interval linear systems**)
strongly semismooth function
[90C30, 90C33]
(*see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**)
strongly semismooth mapping
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
strongly stable optimal solution
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
strongly stable solution
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
structual stability of SIP(f,h,g) *see:* global —
structural analysis of cable structures
[49Q10, 74K99, 74Pxx, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**)
structural analysis system *see:* stability of a —
structural constraints
[90C90, 91B28]
(*see:* **Robust optimization**)
structural design
[90C25, 90C26, 90C27, 90C29, 90C90]
(*see:* **Optimal design of composite structures**; **Semidefinite programming and structural optimization**)
Structural optimization
(49M37, 65K05, 90C30)
(*referred to in:* **Semidefinite programming and structural optimization**; **Shape optimization**; **Structural optimization: history**; **Topology of global optimization**; **Topology optimization**)
structural optimization
[49M37, 65K05, 90C15, 90C26, 90C30, 90C33, 90C90]
(*see:* **Stochastic bilevel programs**; **Structural optimization**; **Structural optimization: history**)
structural optimization
[49M37, 65K05, 90C26, 90C29, 90C30, 90C90]
(*see:* **Optimal design of composite structures**; **Structural optimization**; **Structural optimization: history**)
structural optimization *see:* Semidefinite programming and —
Structural optimization: history
(90C90, 90C26)
(*referred to in:* **Design optimization in computational fluid dynamics**; **Interval analysis: application to chemical engineering design problems**; **Multidisciplinary design optimization**; **Multilevel methods for optimal design**; **Optimal design of composite structures**; **Optimal design in nonlinear optics**; **Semidefinite programming and structural optimization**; **Shape optimization**; **Topology of global optimization**; **Topology optimization**)
(*refers to:* **Bilevel programming: applications in engineering**; **Design optimization in computational fluid dynamics**; **Interval analysis: application to chemical engineering design problems**; **Multidisciplinary design optimization**; **Multilevel methods for optimal design**; **Optimal design of composite structures**; **Optimal design in nonlinear optics**; **Structural optimization**; **Topology optimization**)
structural response *see:* derivatives of —
structural shape optimization
[90C26, 90C90]
(*see:* **Structural optimization: history**)
structural stability
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
structural stability *see:* global —
structural stability in parametric programming
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
structural stability of SIP
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
structural topology optimization
[90C26, 90C90]
(*see:* **Structural optimization: history**)
structural variables
[90C90, 91B28]
(*see:* **Robust optimization**)
structure *see:* analysing declarative program —; block-angular —; coarse valuation —; cost —; dual block-angular —; feasible spanning tree —; fine valuation —; generalized upper bounding —; information —; neighborhood —; optimal spanning tree —; primary —; protein —; secondary —; seed —; spanning tree —; tertiary —
structure determination *see:* molecular —
structure determination: convex global underestimation *see:* Molecular —
structure factors *see:* normalized —
structure of interest rates *see:* term —
structure invariants
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)

structure invariants
[90C26]
(*see:* **Phase problem in X-ray crystallography: Shake and bake approach**)
structure of an irreducible matrix *see:* inductive —
structure mixed integer αBB algorithm *see:* general —; special —
structure prediction *see:* Genetic algorithms for protein —; tertiary —
structure prediction methods *see:* Protein loop —
structure for the QAP *see:* K-L type neighborhood —
structure refinement *see:* incremental strategy for model —
structure of the spatial price equilibrium problem *see:* network —
structured matrices *see:* Stochastic programming: parallel factorization of —
structured matrix
[90C25, 90C30]
(*see:* **Solving large scale and sparse semidefinite programs**)
structured matrix factorization
[90C15]
(*see:* **Stochastic programming: parallel factorization of structured matrices**)
structured singular value
[93D09]
(*see:* **Robust control**)
structures *see:* design of composite —; engineering —; Global optimization of planar multilayered dielectric —; maxdiag fine —; mindiag fine —; model —; Optimal design of composite —; prediction of crystal —; Steiner ratio of biomolecular —; structural analysis of cable —
struts
[51K05, 52C25, 68Q25, 68U05, 90C22, 90C35]
(*see:* Graph realization via semidefinite programming)
(STSP) *see:* symmetric TSP —
(sub)gradients parametric representations *see:* necessary optimality condition without using —
sub Lagrangian
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
sub-problems *see:* Kuhn–Tucker conditions for quadratic programming —
sub-tour elimination constraints
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
sub-tours
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
subclass *see:* conjugate direction —; optimally scaled —; orthogonally scaled —
subconjugate function
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
subcritical function
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
subcritical point
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
subdifferentiability spaces
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
subdifferentiable
[26B25, 26E25, 49J52, 65K99, 90C99]
(*see:* **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**)
subdifferentiable function
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
subdifferential
[26B25, 26E25, 46A20, 49-XX, 49J40, 49J52, 49K27, 49K35, 49M27, 49Q10, 52A01, 58C20, 58E30, 65G20, 65G30, 65G40, 65K05, 65K10, 65K99, 70-08, 70-XX, 74K99, 74Pxx, 80-XX, 90-XX, 90C06, 90C25, 90C26, 90C29, 90C30, 90C35, 90C46, 90C48, 90C99, 90Cxx, 93-XX]
(*see:* **Composite nonsmooth optimization**; **Convex max-functions**; **Duality theory: triduality in global optimization**; **Farkas lemma: generalizations**; **Generalized monotone multivalued maps**; **Global optimization: envelope representation**; **Image space approach to optimization**; **Interval global optimization**; **Minimax: directional differentiability**; **Nonconvex energy functions: hemivariational inequalities**; **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: subgradient optimization methods**; **Nonsmooth analysis: weak stationarity**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: optimality conditions**; **Set-valued optimization**; **Simplicial decomposition algorithms**)
subdifferential
[49J52, 65K05, 90C30]
(*see:* **Nondifferentiable optimization: minimax problems**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**)
subdifferential *see:* $\in$- —; B- —; C- —; Clarke —; Clarke generalized —; convex —; ϵ- —; Fenchel–Moreau —; Fréchet —; gâteaux —; generalized —; H- —; limiting Fréchet —; Moreau–Rockafellar —; singular Fréchet —; singular limiting —
subdifferential of F.H. Clarke *see:* generalized —
subdifferential laws *see:* variational formulation of —
subdifferential set
[46N10, 90-00, 90C47]
(*see:* **Nondifferentiable optimization**)
subdifferential set *see:* ϵ- —
subdifferential Variational Principle
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
subdifferentials
[46A20, 52A01, 65Kxx, 90C30, 90Cxx]
(*see:* **Farkas lemma: generalizations**; **Quasidifferentiable optimization: algorithms for QD functions**)
subdifferentials *see:* Continuous approximations to —; estimation of —; Fréchet —; limiting (Fréchet) —; Nonsmooth analysis: Fréchet —
subdigraph problem *see:* acyclic —

subdivision *see:* cell of a polyhedral —; face of a polyhedral —; guillotine —; polyhedral —; regular —
subdivision directions in interval branch and bound methods *see:* Interval analysis: —
subdivision rule *see:* adaptive —
subdivision via (w,i)
[90B80, 90C11]
(*see:* **Stochastic transportation and location problems**)
subdual function
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
subface
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements in optimization**)
subfamilies of n-valued PI-systems
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
subgame perfect
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
subgradient
[49J52, 49K27, 90C26, 90C29, 90C30, 90C35, 90C48]
(*see:* **Global optimization: envelope representation**; **Image space approach to optimization**; **Lagrangian duality: BASICS**; **Multicommodity flow problems**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**; **Set-valued optimization**)
subgradient
[49J52, 90C30]
(*see:* **Lagrangian duality: BASICS**; **Nondifferentiable optimization: relaxation methods**; **Nondifferentiable optimization: subgradient optimization methods**)
subgradient *see:* H- —
subgradient inequality
[46N10, 49M20, 90-00, 90-08, 90C25, 90C47]
(*see:* **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**)
subgradient locality measure
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
subgradient method *see:* conjugate —; ϵ- —
Subgradient Methods
[49J40, 49J52, 65K05, 90C30]
(*see:* **Nondifferentiable optimization: subgradient optimization methods**; **Solving hemivariational inequalities by nonsmooth optimization methods**)
subgradient optimization
[90C10, 90C11, 90C15, 90C26, 90C27, 90C30, 90C33, 90C57]
(*see:* **Cost approximation algorithms**; **Set covering, packing and partitioning problems**; **Stochastic bilevel programs**)
subgradient optimization
[90C30]
(*see:* **Cost approximation algorithms**)
subgradient optimization *see:* Lagrangian relaxation with —
subgradient optimization methods *see:* Nondifferentiable optimization: —
subgradient projection
[47H05, 65J15, 90C25, 90C55]
(*see:* **Fejér monotonicity in convex optimization**)
subgradient projection algorithm
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
subgradient techniques
[90C30, 90C90]
(*see:* **Decomposition techniques for MILP: lagrangian relaxation**)
subgradients
[46N10, 49K35, 49M27, 65K10, 90-00, 90C25, 90C30, 90C47]
(*see:* **Convex max-functions**; **Lagrangian duality: BASICS**; **Nondifferentiable optimization**)
subgraph
[90C35]
(*see:* **Feedback set problems**)
subgraph *see:* induced —; length of a —; maximal planar —; maximum bipartite —; maximum planar —; planar —
subgraph approach
[05C69, 05C85, 68Q20, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**; **Optimal triangulations**)
subgraph induced by a vertex subset
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
Subgraph Problem *see:* degree-constrained —; minMax Matching —
subinterval adaptation
[90C26, 90C30]
(*see:* **Bounding derivative ranges**)
subinterval selection
[65K05, 90C30]
(*see:* **Algorithmic improvements using a heuristic parameter, reject index for interval optimization**)
subject to moment conditions *see:* optimal integral bounds —
subjective curve fitting
[90C26, 90C30]
(*see:* **Forecasting**)
subjective curve fitting and extrapolation
[90C26, 90C30]
(*see:* **Forecasting**)
subjective interpretation
[94A17]
(*see:* **Jaynes' maximum entropy principle**)
subjet
[90C26]
(*see:* **Global optimization: envelope representation**)
sublattice
[90C35]
(*see:* **Multi-index transportation problems**)
sublinear *see:* difference —
sublinear function
[90C30]
(*see:* **Image space approach to optimization**)
sublinear function *see:* difference —
sublinear system
[46A20, 52A01, 90C30]
(*see:* **Farkas lemma: generalizations**)
submatrix *see:* principal —

submatroid
[90C09, 90C10]
(*see:* **Oriented matroids**)
submodular constraints
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
submodular function
[90C09, 90C10, 90C25, 90C27, 90C35]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**; **L-convex functions and M-convex functions**)
submodular function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
submodular polyhedron
[90C09, 90C10, 90C35]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**; **Multi-index transportation problems**)
submodular system
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
submodularity
[90C35]
(*see:* **Multi-index transportation problems**)
Suboptimal control
(90C30)
(*referred to in:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
(*refers to:* **Control vector iteration CVI**; **Duality in optimal control with first order differential equations**; **Dynamic programming: continuous-time optimal control**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Dynamic programming: optimal control applications**; **Hamilton–Jacobi–Bellman equation**; **Infinite horizon control and dynamic games**; **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**; **Optimal control of a flexible arm**; **Optimization strategies for dynamic systems**; **Robust control**; **Robust control: schur stability of polytopes of polynomials**; **Semi-infinite programming and control problems**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**)
suboptimal control
[90C30]
(*see:* **Suboptimal control**)
suboptimal trajectories and controls
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
suboptimality
[90C30]
(*see:* **Suboptimal control**)
suborder
[90C29]
(*see:* **Preference modeling**)
subproblem
[90C06, 90C30]
(*see:* **Decomposition principle of linear programming**; **Simplicial decomposition**)
subproblem
[90C30]
(*see:* **Simplicial decomposition**)
subproblem *see:* allocation —; column generation —; master —; NLP —; primal —; quadratic programming —; reduced quadratic programming —; regularized —
subproduct of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
subrelation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
subroutine library *see:* IMSL —
subroutines *see:* FORTRAN —
subset *see:* covering —; elemental —; good —; independent —; optimal —; signed —; subgraph induced by a vertex —
subset feedback vertex (arc) set problem
[90C35]
(*see:* **Feedback set problems**)
subset heterogeneity
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
subset interconnection designs
[90C27]
(*see:* **Steiner tree problems**)
subset minimum feedback vertex (arc) set problem
[90C35]
(*see:* **Feedback set problems**)
subset of participants
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
subset selection
[90C15, 90C27]
(*see:* **Discrete stochastic optimization**)
subset-sum problem
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C, 90C20, 90C60]
(*see:* **Convex discrete optimization**; **Quadratic knapsack**)
subsets *see:* transversal of a collection of —
subspace *see:* concurrent —; finite-dimensional —
subspace optimization *see:* concurrent —
substationarity
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]

(*see:* **Nonconvex energy functions: hemivariational inequalities**)
substationarity point of a functional
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
substationarity point with respect to a set
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
substationarity problems
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX, 90C90, 91A65]
(*see:* **Multilevel optimization in mechanics**; **Nonconvex energy functions: hemivariational inequalities**)
substationary point
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
substationary point
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
substitution
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
substitution *see:* forward —; successive —
substitution supernode
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
substructure property *see:* optimal —
substructuring
[65Fxx]
(*see:* **Least squares problems**)
subtour elimination constraints
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
subtours
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
subtract
[90C11]
(*see:* **Predictive method for interhelical contacts in alpha-helical proteins**)
subtree
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
subtree hypergraph
[90C27]
(*see:* **Steiner tree problems**)
subtree isomorphism
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
subtree isomorphism *see:* maximal —; maximal similarity —; maximum —; maximum similarity —
successive affine reduction
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
successive affine reduction BFGS algorithm
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
successive approximation
[49L20, 90C40]
(*see:* **Dynamic programming: undiscounted problems**)
successive displacements *see:* method of —
successive improvement of KKT points
[90C30]
(*see:* **Large scale trust region problems**)
successive overrelaxation
[90C33]
(*see:* **Linear complementarity problem**)
Successive quadratic programming
(90C30)
(*referred to in:* **Convex max-functions**; **Feasible sequential quadratic programming**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Feasible sequential quadratic programming**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
successive quadratic programming
[65L99, 90C20, 90C25, 90C30, 90C90, 93-XX]
(*see:* **Dynamic programming: optimal control applications**; **Optimization strategies for dynamic systems**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
successive quadratic programming
[65K05, 65K10, 90C06, 90C20, 90C30, 90C34, 90C90]
(*see:* **Feasible sequential quadratic programming**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: decomposition methods**)
successive quadratic programming *see:* full space —
Successive quadratic programming: applications in distillation systems
(90C30, 90C90)
(*referred to in:* **Feasible sequential quadratic programming**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sequential quadratic programming: interior**

point methods for distributed optimal control problems; **Successive quadratic programming**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Feasible sequential quadratic programming**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)

Successive quadratic programming: applications in the process industry
(90C30, 90C90)
(*referred to in:* **Feasible sequential quadratic programming**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Feasible sequential quadratic programming**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)

Successive quadratic programming: decomposition methods
(90C30, 90C20)
(*referred to in:* **Decomposition principle of linear programming**; **Feasible sequential quadratic programming**; **Generalized benders decomposition**; **MINLP: generalized cross decomposition**; **MINLP: logic-based methods**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Decomposition principle of linear programming**; **Feasible sequential quadratic programming**; **Generalized benders decomposition**; **MINLP: generalized cross decomposition**; **MINLP: logic-based methods**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Stochastic linear programming: decomposition and cutting planes**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**)

Successive quadratic programming: full space methods
(90C30, 90C25)
(*referred to in:* **Feasible sequential quadratic programming**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Rosen's method, global convergence, and Powell's conjecture**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: solution by active sets and interior point methods**)
(*refers to:* **Feasible sequential quadratic programming**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: solution by active sets and interior point methods**)

Successive quadratic programming: solution by active sets and interior point methods
(90C30, 90C25)
(*referred to in:* **Entropy optimization: interior point methods**; **Feasible sequential quadratic programming**; **Homogeneous selfdual methods for linear programming**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Potential reduction methods for linear programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**)
(*refers to:* **Entropy optimization: interior point methods**; **Feasible sequential quadratic programming**; **Homogeneous selfdual methods for linear programming**; **Interior point methods for semidefinite programming**; **Linear programming: interior point methods**; **Linear**

programming: karmarkar projective algorithm; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Potential reduction methods for linear programming**; **Sequential quadratic programming: interior point methods for distributed optimal control problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**)
successive shortest path algorithm
[90C35]
(*see:* **Minimum cost flow problem**)
successive shortest path algorithm
[90C35]
(*see:* **Minimum cost flow problem**)
successive substitution
[65H10, 65J15]
(*see:* **Contraction-mapping**)
sufficiency *see:* second order —; strong second order —
sufficiency theorem for the Hamilton–Jacobi–Bellman equation
[34H05, 49L20, 90C39]
(*see:* **Hamilton–Jacobi–Bellman equation**)
sufficient
[90C22, 90C25, 90C31, 90C33, 90C34, 90C46]
(*see:* Generalized semi-infinite programming: optimality conditions; **Linear complementarity problem**; **Semidefinite programming: optimality conditions and stability**)
sufficient *see:* column —; row —
sufficient condition
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
sufficient condition
[90C30]
(*see:* **Image space approach to optimization**)
sufficient condition *see:* general second order —; general strong second order —; saddle-point —; second order —; strong second order —
sufficient conditions
[90C26, 90C31, 90C34, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**; **Semi-infinite programming: second order optimality conditions**)
sufficient conditions *see:* necessary and —; second order —
sufficient decrease conditions
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
sufficient matrix
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**; **Principal pivoting methods for linear complementarity problems**)
sufficient matrix *see:* column —; row —
sufficient optimality condition
[90C26, 90C31, 91A65]
(*see:* **Bilevel programming: implicit function approach**)
sufficient optimality conditions
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
sufficient optimality conditions
[90C26, 90C31, 90C39, 91A65]
(*see:* **Bilevel programming: implicit function approach**; **Second order optimality conditions for nonlinear optimization**)
sufficient optimality conditions *see:* necessary and —; second order necessary and —
sufficiently high
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
sufficiently large
[90C11, 90C26]
(*see:* **Extended cutting plane algorithm**)
sufficiently small
[25A15, 34A05, 90C25, 90C26, 90C30, 90C31]
(*see:* **Convexifiable functions, characterization of**)
sum *see:* Minkowski —
sum of convex multiplicative functions
[90C26, 90C31]
(*see:* **Multiplicative programming**)
sum diagram *see:* cumulative —
sum game *see:* two-person zero- —
sum infinite horizon game *see:* nonzero- —
sum of integer infeasibilities
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
sum matrix
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
sum perfect-information game *see:* two-player zero- —
sum problem *see:* subset- —
sum of ratios *see:* maximizing a —
sum-of-ratios fractional program
[90C32]
(*see:* **Fractional programming**)
sum Rule
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
sum rule *see:* fuzzy —
sum of squared error
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
sum of squares
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
sums program *see:* weighted- —
sums programs with constraints *see:* weighted- —
sup norm *see:* weighter —
sup-norm contraction *see:* weighter —
sup-stationary point
[90Cxx]
(*see:* **Quasidifferentiable optimization: optimality conditions**)
sup-stationary point *see:* Dini —; Hadamard —
sup theorem *see:* James —
supconjugate function
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)

super-polynomial time
[90C27, 90C60, 91A12]
(*see:* **Combinatorial optimization games**)
superadditive dual
[90C10, 90C46]
(*see:* **Integer programming duality**)
superadditive duality
[90C10, 90C46]
(*see:* **Integer programming duality**)
superadditive function
[90C10, 90C46]
(*see:* **Integer programming duality**)
superbasic variables
[90C30]
(*see:* **Convex-simplex algorithm**)
superbasic variables
[90C30]
(*see:* **Convex-simplex algorithm**)
superconsistency
[90C05, 90C25, 90C30, 90C34]
(*see:* **Semi-infinite programming, semidefinite programming and perfect duality**)
supercritical function
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
supercritical point
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
supercritical point theorem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
superdifferentiable
[26B25, 26E25, 49J52, 90C99, 90Cxx]
(*see:* **Quasidifferentiable optimization**; **Quasidifferentiable optimization: optimality conditions**)
superdifferentiable function
[90C06]
(*see:* **Saddle point theory and optimality conditions**)
superdifferential
[26B25, 26E25, 46A20, 49-XX, 49J52, 49K27, 52A01, 58C20, 58E30, 65K05, 65K99, 70-08, 90-XX, 90C25, 90C30, 90C48, 90C99, 90Cxx, 93-XX]
(*see:* **Duality theory: triduality in global optimization**; **Farkas lemma: generalizations**; **Minimax: directional differentiability**; **Nonsmooth analysis: Fréchet subdifferentials**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: optimality conditions**)
superdifferential *see:* Fréchet —; limiting —
superfluous
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
supergraph *see:* three-layer —
superLagrangian
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
superLagrangian
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**; **Duality theory: triduality in global optimization**)
superLagrangian duality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
superLagrangian duality theorem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
superlinear *see:* 2-step —; Q- —
superlinear convergence
[49J52, 90C30]
(*see:* **Nondifferentiable optimization: Newton method**)
superlinear convergence *see:* Q- —
superlinear convergence condition
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
superlinear convergent rate
[90C30]
(*see:* **Simplicial decomposition**)
superlinear function
[90C30]
(*see:* **Image space approach to optimization**)
supermaximum point
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
superminimax point
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
superminimax theorem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
supermodular function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
supernode *see:* insertion —; substitution —
superpositions of functions
[01A60, 03B30, 54C70, 68Q17]
(*see:* **Hilbert's thirteenth problem**)
superpositions of functions
[01A60, 03B30, 54C70, 68Q17]
(*see:* **Hilbert's thirteenth problem**)
superpotential
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**)
superpotential *see:* nonconvex —; nonsmooth —; quasidifferentiable —
superpotentials *see:* convex variational inequality for an elastostatic problem involving QD- —; elastostatic problem involving QD- —; variational equality for an elastostatic problem involving QD- —
superproduct of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
superrelation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
superstep
[65K05, 65Y05]
(*see:* **Parallel computing: models**)

superstructure
[03H10, 49J27, 49M37, 90C11, 90C34]
(*see:* **MINLP: applications in the interaction of design and control**; **Semi-infinite programming and control problems**)
superstructure *see:* design —; distillation —; heat exchanger network —; MEN —
superstructure model
[90C90]
(*see:* **MINLP: heat exchanger network synthesis**)
supervised classification
[90C90]
(*see:* **Optimization in medical imaging**)
supervised learning
[65K05, 68T05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C52, 90C53, 90C55, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**; **Unconstrained optimization in neural network training**)
supervisor algorithm
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms**)
supervisor and searcher cooperation minimization algorithms
[90C15, 90C30, 90C99]
(*see:* **SSC minimization algorithms**)
supply *see:* net —
supply chain
[90-02, 90B05, 90B06]
(*see:* **Global supply chain models**; **Operations research models for supply chain management and design**)
supply chain
[90B50]
(*see:* **Inventory management in supply chains**)
supply chain *see:* global —; operational decisions in a —; strategic design of a —
supply chain design
[90-02]
(*see:* **Operations research models for supply chain management and design**)
Supply chain management
[90-02]
(*see:* **Operations research models for supply chain management and design**)
supply chain management
[90-02]
(*see:* **Operations research models for supply chain management and design**)
supply chain management *see:* Bilinear programming: applications in the —; Mathematical programming methods in —; operational —; strategic —
supply chain management and design *see:* Operations research models for —
supply chain models *see:* Global —
supply chain optimization
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
Supply chain performance measurement
(00-02, 01-02, 03-02)
supply chain simulation models
[90-02]
(*see:* **Operations research models for supply chain management and design**)
supply chains *see:* Inventory management in —
supply node
[90C35]
(*see:* **Minimum cost flow problem**)
support *see:* decision —; linear —; Multiple objective programming —; total —
support function
[26E25, 49J52, 52A27, 65K05, 90C26, 90C30, 90C99]
(*see:* **Global optimization: envelope representation**; **Minimax: directional differentiability**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**)
support function
[65K05, 90C30]
(*see:* **Minimax: directional differentiability**)
support hyperplane
[90C26]
(*see:* **Global optimization: envelope representation**)
support of an integral vector
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
support methodologies for auditing decisions *see:* Multicriteria decision —
support of a natural vector
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
support problems method *see:* extended —
support problems solution method
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
support set
[90C05, 90C10, 90C15]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**; **Probabilistic constrained linear programming: duality theory**)
support set
[90C26]
(*see:* **Global optimization: envelope representation**)
support set of a function
[90C26]
(*see:* **Global optimization: envelope representation**)
support system *see:* Asset liability management decision —; decision —; intelligent multicriteria decision —; multicriteria decision —; multicriteria group decision —
support systems *see:* decision —; intelligent multicriteria decision —; Multi-objective optimization and decision —; Optimization and decision —
support systems with multiple criteria *see:* Decision —
support vector machine *see:* generalized eigenvalue proximal —
support vector machine problem *see:* Generalized eigenvalue proximal —
support vector machines
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
supported efficient solution
[90C10, 90C29]
(*see:* **Multi-objective integer linear programming**)
supported efficient solutions
[90C10, 90C35]
(*see:* **Bi-objective assignment problem**)

supporting function *see:* linear —
supports problems method
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
supremal generator
[90C26]
(*see:* **Global optimization: envelope representation**)
supremal generator
[90C26]
(*see:* **Global optimization: envelope representation**)
supremum *see:* essential —
surface
(*see:* **State of the art in modeling agricultural systems**)
surface *see:* response —
surface formula *see:* integral over —
surface and groundwater resources
[90C30, 90C35]
(*see:* **Optimization in water resources**)
surface and groundwater systems
[90C30, 90C35]
(*see:* **Optimization in water resources**)
surface method *see:* response —
surface traction
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)
surface water pumping facilities
[90C30, 90C35]
(*see:* **Optimization in water resources**)
surplus wealth
[91B28]
(*see:* **Financial optimization**)
surrogate *see:* document —
surrogate constraint
[90C10, 90C46]
(*see:* **Integer programming duality**)
surrogate dual
[90C10, 90C30]
(*see:* **Integer programming: lagrangian relaxation**)
surrogate duality
[90C10, 90C46]
(*see:* **Integer programming duality**)
surrogate relaxation
[90C10, 90C46]
(*see:* **Integer programming duality**)
surrogates *see:* binary —; similarity of —
surveys *see:* integration of —
survivability
[90-XX]
(*see:* **Survivable networks**)
survivable network
[90C10, 90C27, 94C15]
(*see:* **Graph planarization**)
survivable network design problem
[90-XX]
(*see:* **Survivable networks**)
Survivable networks
(90-XX)
(*referred to in:* **Auction algorithms**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Traffic network equilibrium**)
(*refers to:* **Auction algorithms**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Traffic network equilibrium**)
survival of the fittest
(*see:* **Broadcast scheduling problem**)
sVNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
(SVP) *see:* seismic Vessel Problem —
SVRP
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
swap *see:* greedy —
swaps *see:* monotone sequence of greedy —
sweep algorithm
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
sweep method *see:* Aitken double —
switches *see:* maximum number of well —
switching
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
switching circuit *see:* combinatorial —
switching curve
[90C30]
(*see:* **Suboptimal control**)
switching engines *see:* scheduling of —
switching time
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
Sylvester problem
[90B85, 90C27]
(*see:* **Single facility location: circle covering problem**)
symbolic differentiation
[26A24, 65D25, 68W30]
(*see:* **Automatic differentiation: introduction, history and rounding error estimation**; **Complexity of gradients, Jacobians, and Hessians**)
symbolic manipulation
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
symbolic preprocessing
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
symbolic translation
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)

symbolically transforming declarative programs
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
symmetric
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
symmetric–Broyden method *see:* Powell- —
symmetric continuous form *see:* coercive bilinear —
symmetric element in a Hilbert space
[65M60]
(*see:* **Variational inequalities: F. E. approach**)
symmetric interior of a relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
symmetric interval matrix *see:* dimensional —; real —
symmetric matrix
[65K05, 90Cxx]
(*see:* **Symmetric systems of linear equations**)
symmetric matrix
[65K05, 90Cxx]
(*see:* **Symmetric systems of linear equations**)
symmetric matrix *see:* positive semidefinite —; skew- —
symmetric matrix M *see:* skew- —
symmetric multi-index transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)
symmetric network equilibrium
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
symmetric proximity
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
symmetric proximity *see:* skew- —
symmetric rank-one approach *see:* limited-memory —
symmetric rank-one quasi-Newton method
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
symmetric rank-one update
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
symmetric relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
symmetric $S_{2\times 2\times 2}$ group
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
Symmetric systems of linear equations
(65K05, 90Cxx)
(*referred to in:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Gauss, Carl Friedrich**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**)
(*refers to:* **ABS algorithms for linear equations and linear least squares**; **Cholesky factorization**; **Gauss, Carl Friedrich**; **Interval linear systems**; **Large scale trust region problems**; **Large scale unconstrained optimization**; **Linear programming**; **Orthogonal triangularization**; **Overdetermined systems of linear equations**; **QR factorization**; **Solving large scale and sparse semidefinite programs**)
symmetric TSP
[90C59]
(*see:* **Heuristic and metaheuristic algorithms for the traveling salesman problem**)
symmetric TSP (STSP)
[68Q25, 68R10, 68W40, 90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Domination analysis in combinatorial optimization**; **Traveling salesman problem**)
symmetry model *see:* rotation- —
synchronized distributed state space search algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
synchronized parallel CA algorithm
[90C30]
(*see:* **Cost approximation algorithms**)
synchronous implementation of the auction algorithm
[90C30, 90C35]
(*see:* **Auction algorithms**)
synchronous parallel *see:* bulk —
synchronous parallel computer *see:* bulk —
synchronous parallel model *see:* bulk —
syndrome *see:* NIMBY —
synthesis *see:* heat exchanger network —; HEN —; MINLP: heat exchanger network —; MINLP: reactive distillation column —; Mixed integer linear programming: heat exchanger network —; network —; problem —; process —; robust control —; sequential —; simultaneous —
synthesis and control *see:* interaction of design —; mu —
synthesis and design under uncertainty *see:* process —
synthesis method *see:* sequential MEN —
synthesis model *see:* multiperiod MINLP MEN —
synthesis problem
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
synthesis of separation processes
[90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**)
synthesis using MINLP *see:* HEN —
synthesis without decomposition *see:* heat exchanger network —
system
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
system *see:* Aizenberg–Rabinovich —; alternative linear —; ant —; Asset liability management decision support —; asymptotical stability of a —; bisubmodular —; center of an interval linear —; curvilinear coordinate —; decision support —; degenerate —; discrete event dynamic —; DSL —; dual —; dynamical —; electric power —; expert —; finite jump —; Fritz John —; generalized network optimization —; Hamiltonian —; high performance computing —; independence —; independent —;

infeasible —; initial —; intelligent multicriteria decision support —; interactive disaggregation —; interval linear —; iterative function —; lattice-type many-valued logic —; MAX-MIN ant —; moving coordinate —; multi-echelon arborescence —; multicriteria decision support —; multicriteria group decision support —; optimization —; perturbed —; Post —; preferential bidding —; projected dynamical —; propositional proof —; reduced RLT —; rule-based —; selfdual —; sign-solvable linear —; solution of a —; stability of a —; stability of a structural analysis —; state of a —; sublinear —; submodular —; time-delay —; tolerant —; totally dual integral —; variation of a —; Variational inequalities: projected dynamical —; variational inequality problem and a projected dynamical —
system of approximate reasoning *see:* interval logic —; point-based logic —
system cohomology *see:* local —
system conditions *see:* economic —
system design *see:* distribution —
system design problem *see:* Production-distribution —
system of equations
[65H10, 65K10, 65M60, 90C26, 90C30]
(*see:* **Global optimization methods for systems of nonlinear equations**; **Variational inequalities**)
system of equations
[65K10, 65M60]
(*see:* **Variational inequalities**)
system of equations *see:* nonlinear —; polynomial —
system of inequalities
[65H10, 90C26, 90C30]
(*see:* **Global optimization methods for systems of nonlinear equations**)
system of nonlinear equations *see:* overdetermined —; underdetermined —; well-determined —
system-optimization
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
system, optimization of *see:* Wastewater —
system-optimized transportation network
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
system-optimizing environment
[90B80, 90B85, 90Cxx, 91Axx, 91Bxx]
(*see:* **Facility location with externalities**)
system of simplexes
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
system stability
[90B15]
(*see:* **Dynamic traffic networks**)
system stability
[90B15]
(*see:* **Dynamic traffic networks**)
system stability *see:* asymptotical —
system of variational inequalities
[49J40, 49M05, 49S05, 74G99, 74H99, 74Pxx]
(*see:* **Quasidifferentiable optimization: variational formulations**)
systematic search
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
systems *see:* alternative —; batch production —; boundary flux estimation in distributed —; conjunctive use of water resource —; convex inequality —; convex-like —; decision support —; discrete dynamical —; discrete-time —; distribution —; estimating uncertainty in dynamical —; expert —; Global optimization in the analysis and management of environmental —; homogeneous —; homogeneous dual —; inequality —; intelligent multicriteria decision support —; Interval analysis for optimization of dynamical —; Interval linear —; inventory —; large scale linear —; linearly elastic —; Model based control for drug delivery —; Multi-objective optimization and decision support —; nondegenerate —; Optimization and decision support —; Optimization in operation of electric and energy power —; Optimization strategies for dynamic —; projected dynamical —; Quasidifferentiable optimization: stability of dynamic —; reaction flux estimation in lumped —; satellite —; State of the art in modeling agricultural —; stochastic dynamic —; subfamilies of n-valued PI- —; Successive quadratic programming: applications in distillation —; surface and groundwater —; time-delay —; uncertain —; water transportation —
systems by constructive nonlinear dynamics *see:* Robust design of dynamic —
systems of equations *see:* error bound for approximate solutions of nonlinear —; existence of solutions of nonlinear —; linear —; nonlinear —; rigorous bound for solutions of nonlinear —; uniqueness of solutions of nonlinear —
systems of equations: application to the enclosure of all azeotropes *see:* Nonlinear —
systems of linear equations *see:* Overdetermined —; Symmetric —
systems of linear relations *see:* Tucker homogeneous —
systems of many-valued logic algebras *see:* Finite complete —
systems modeling and management *see:* applications in environmental —
systems with multiple criteria *see:* Decision support —
systems of nonlinear equations
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)
systems of nonlinear equations
[65H10, 90C26, 90C30]
(*see:* **Global optimization methods for systems of nonlinear equations**)
systems of nonlinear equations *see:* Global optimization methods for —; Interval analysis: —
systems for oriented matroids *see:* axiom —
systems planning *see:* distribution —
systems theory and control
[49-XX, 60Jxx, 65Lxx, 91B32, 92D30, 93-XX]
(*see:* **Resource allocation for epidemic control**)
systems of variational inequalities *see:* QD laws and —

T

T *see:* prohibition parameter —

t-coloring
[05-XX]
(*see:* **Frequency assignment problem**)
T-coloring frequency assignment *see:* order of a —; span of a —
t-conorm
[03B50, 03B52, 03E72, 47S40, 68T15, 68T27, 68T30, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Finite complete systems of many-valued logic algebras**)
t-conorms
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
t-norm
[03B50, 03B52, 03E72, 47S40, 68T15, 68T27, 68T30, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**; **Finite complete systems of many-valued logic algebras**)
t-norms
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
TA
[90C05, 90C30]
(*see:* **Theorems of the alternative and optimization**)
table *see:* contingency —; k-way —
table with given marginals
[90C35]
(*see:* **Multi-index transportation problems**)
table representation *see:* lookup —
tableau *see:* distinguished —; initial —; lexico-positive basis —; terminal simplex —
tables *see:* triangulation problem for input-output —
taboo restriction
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
tabs search *see:* fixed —
tabu *see:* forbidden or —
tabu list
[05C69, 05C85, 68T99, 68W01, 90C27, 90C59]
(*see:* **Capacitated minimum spanning trees**; **Heuristics for maximum clique and independent set**)
tabu search
[03B05, 62C10, 65K05, 68P10, 68Q25, 68R05, 68T15, 68T20, 68T99, 90B80, 90C05, 90C08, 90C09, 90C10, 90C11, 90C15, 90C26, 90C27, 90C35, 90C57, 90C59, 94C10]
(*see:* **Bayesian global optimization**; **Capacitated minimum spanning trees**; **Communication network assignment problem**; **Maximum constraint satisfaction: relaxations and upper bounds**; **Maximum satisfiability problem**; **Metaheuristics**; **Multi-index transportation problems**; **Quadratic assignment problem**)
tabu search *see:* allowed neighbor in —; Hamming-reactive —; prohibited neighbor in —; reactive —
tabu search methodology
[90C10, 90C11, 90C20]
(*see:* **Linear ordering problem**)
tactics *see:* trading —
TAG
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
tail *see:* RSM-distribution with algebraically decreasing —
tail(l)
(*see:* **Railroad crew scheduling**)
tail of operation
[90B35]
(*see:* **Job-shop scheduling problem**)
tail problem *see:* head-body- —
tailing-off
[68Q99]
(*see:* **Branch and price: Integer programming with column generation**)
tailored optimization
[90C30, 90C90]
(*see:* **Successive quadratic programming: applications in the process industry**)
taker *see:* price —
Tal SOCQ *see:* ben- —
Tanabe–Todd–Ye potential function
[37A35, 90C05]
(*see:* **Potential reduction methods for linear programming**)
tangent
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
tangent *see:* bouligand —; weak —
tangent approximating vector *see:* high-order —
tangent cone
[65K05, 90C20, 90C22, 90C25, 90C31]
(*see:* **Quadratic programming with bound constraints**; **Semidefinite programming: optimality conditions and stability**)
tangent cone
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
tangent cone *see:* Bouligand —
tangent high-order approximating cone
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
tangent high-order approximating cones
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
tangent high-order approximating curve
[41A10, 46N10, 47N10, 49K27]
(*see:* **High-order necessary conditions for optimality for abnormal points**)
tangent hyperplane
[90C26]
(*see:* **Global optimization: envelope representation**)
tangent-plane criterion
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
tangent-plane criterion *see:* reaction —
tangent set *see:* first order —; second order —
tangent sets *see:* high-order —
tangents *see:* parallel —
tangents algorithm *see:* parallel- —
tangle basis
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)

tangle number
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)
tape
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
tape cell of a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
tape heads
[90C60]
(*see:* **Complexity classes in optimization**)
tape of a Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
target
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
target analysis
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
targets
(*see:* **Planning in the process industry**)
targets *see:* environmental —
Tarjan planarity-testing algorithm *see:* Hopcroft– —
task
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
task allocation
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
task-network *see:* state- —
tatonnement process
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)
τ*-estimate of the spot rate*
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
τ*-programmed problem of spot rate estimation*
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
τ_b statistic *see:* Goodman–Kruskal —
TAUT
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
TAUT-DNF
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
taxonomy
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
taxonomy of the PI-algebras of many-valued logics
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
taxonomy of Pi-logic algebras
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
Taylor approximation *see:* truncated —
Taylor coefficients
[90C26, 90C30]
(*see:* **Bounding derivative ranges**)
Taylor form
[90C26, 90C30]
(*see:* **Bounding derivative ranges**)
Taylor form test
[90C26, 90C30]
(*see:* **Bounding derivative ranges**)
Taylor Operator *see:* interval —; Point —
taylor operators *see:* Automatic differentiation: point and interval —
Taylor series
[65G20, 65G30, 65G40, 65L99, 90C30, 90C52, 90C53, 90C55]
(*see:* **Gauss–Newton method: Least squares, relation to Newton's method**; **Interval analysis: differential equations**; **Suboptimal control**)
Taylor series
[65K05, 90C26, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**; **Bounding derivative ranges**)
Taylor series expansion *see:* first order —
Taylor theorem
[65K05, 90C30]
(*see:* **Automatic differentiation: point and interval taylor operators**)
TBP
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
TCF
[90C60]
(*see:* **Computational complexity theory**)
TCF of an algorithm
[90C60]
(*see:* **Computational complexity theory**)
Tchebycheff metric *see:* w-weighted —
Tchebycheff program
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
TCVSP
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
TDI
[90C35]
(*see:* **Feedback set problems**)
TDTSP
[90C27]
(*see:* **Time-dependent traveling salesman problem**)
technique *see:* auction —; Bland —; boundary variation —; clipping —; dynamic load balancing —; greedy —; inexact line search —; line search —; penalty —; perturbation —; pictogram translation mapping —; receiver initiated mapping —; reformulation-linearization —; reformulation-Linearization/Convexification —; Relaxation —; sender initiated mapping —; sphere growing —; trust region —; uniform grid —; variance reduction —
technique for global optimization *see:* Reformulation-linearization —

techniques *see:* acceleration devices and related —; approximation —; branch and bound —; branch and bound enumerative —; computer aided —; constraint satisfaction —; decomposition —; dual —; enumeration —; Estimating data for multicriteria decision making problems: optimization —; Load balancing for parallel optimization —; measurement —; NLP —; presolving —; reformulation —; reformulation-linearization/convexification —; subgradient —
techniques for MILP: lagrangian relaxation *see:* Decomposition —
techniques for minimizing the energy function *see:* Optimization —
techniques for phase retrieval based on single-crystal X-ray diffraction data *see:* Optimization —
technological comparison
[90C26, 90C30]
(*see:* **Forecasting**)
telecommunication
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
telecommunication
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
telecommunications
[90C35]
(*see:* **Multicommodity flow problems**)
teletherapy
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
temperature
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
temperature *see:* initial —; initial annealing —; start —
temperature cascade
[90C90]
(*see:* **Mixed integer linear programming: heat exchanger network synthesis**)
temperature control
[35R70, 47S40, 74B99, 74D99, 74G99, 74H99]
(*see:* **Quasidifferentiable optimization: applications to thermoelasticity**)
temperature interval diagram
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
temperature parameter
[65K05, 90C30]
(*see:* **Random search methods**)
template *see:* checklist —; pool —
templates *see:* De novo protein design using flexible —; De novo protein designUsing rigid —; deformable —
Temple quotient
[49R50, 65G20, 65G30, 65G40, 65L15, 65L60]
(*see:* **Eigenvalue enclosures for ordinary differential equations**)
temporal difference
[49L20, 90C39, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**; **Neuro-dynamic programming**)
temporal links
(*see:* **Bayesian networks**)
tensegrity
[51K05, 52C25, 68Q25, 68U05, 90C22, 90C35]
(*see:* Graph realization via semidefinite programming)
tensegrity *see:* unyielding —
tensor *see:* strain —
tensor method
[90C06]
(*see:* **Large scale unconstrained optimization**)
tenth problem *see:* Hilbert —
Terlaky criss-cross method
[90C05]
(*see:* **Linear programming: Klee–Minty examples**)
term *see:* intermediate —
term to maturity
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
term memory *see:* short- —
term memory in GRASP *see:* long- —
term of a polynomial *see:* initial —
term-recurrence *see:* three- —
term-recurrence algorithm *see:* three- —
term scheduling of batch processes *see:* Medium- —
term scheduling of batch processes with resources *see:* Short- —
term scheduling of continuous processes *see:* Short- —
term scheduling, resource constrained: unified modeling frameworks *see:* Short- —
term scheduling under uncertainty: sensitivity analysis *see:* Short- —
term structure of interest rates
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
term structure of interest rates
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
terminal layout problem
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
terminal layout problem
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
terminal simplex tableau
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
terminals
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
terminals *see:* away —; home —
terminate
[90C26, 90C31]
(*see:* **Robust global optimization**)
termination
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)

termination criterion
[92C05, 92C40]
(*see:* **Protein loop structure prediction methods**)
termination on a secondary ray
[90C33]
(*see:* **Linear complementarity problem**)
terms *see:* bilinear —; cost —; indexing —; Interval analysis: intermediate —; linear fractional —; posynomial —; shift —; transportation —
ternary matroid
[90C09, 90C10]
(*see:* **Matroids**)
terrain/funneling methods *see:* Multi-scale global optimization using —
terrain methods *see:* Global —
tertiary structure
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
tertiary structure
[92B05]
(*see:* **Genetic algorithms for protein structure prediction**)
tertiary structure prediction
[92C40]
(*see:* **Monte-Carlo simulated annealing in protein folding**)
test *see:* feasibility —; Hessian —; infeasibility —; lower bound —; midpoint —; minimum ratio —; monotonicity —; monotonicity and nonconvexity —; Newton —; nonconvexity —; redundancy —; Taylor form —; upper-bound —; Wolfe —
test for the existence of solutions of equations
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
test and flexibility index *see:* Bilevel optimization: feasibility —
test nonmonotone Armijo-like criterion
[49M07, 49M10, 65K, 90C06, 90C20]
(*see:* **Spectral projected gradient methods**)
test of optimality
[90C05, 90C33]
(*see:* **Pivoting algorithms for linear programming generating two paths**)
test problems and problem generators *see:* Combinatorial —
test set
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
test sets
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
test sets in integer programming
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
testing *see:* planarity —
testing algorithm *see:* Hopcroft–Tarjan planarity- —
testing relational properties
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
tests *see:* feasibility convergence —; midpoint —; value convergence —
text classification
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
text classification
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
text documents *see:* classification of —; Optimization in classifying —
ThAlt
[15A39, 90C05]
(*see:* **Linear optimization: theorems of the alternative**)
than-truckload *see:* less- —
their simulation *see:* Derivatives of markov processes and —
theorem *see:* alternative —; asynchronous convergence —; Bandler–Kohout compatibility —; basic alternative —; basic sensitivity —; Bassett–Maybee–Quirk —; biduality —; birkhoff's —; block —; brouwer fixed point —; Broyden —; Carathéodory —; Clarke duality —; classical Lyusternik —; coincidence —; composition —; conic duality —; convergence —; cook's —; Cook–Levin —; Du–Hwang minimax —; duality —; Dubovitskii—Milyutin —; Duffin —; equivalence —; Finsler —; fixed point —; folks —; Frank discrete separation —; gap —; Gauss–Markoff —; Gauvin —; Geoffrion —; Gershgorin —; Gordan transposition —; Hahn–Banach —; Hahn–Banach linear extension —; Hansel —; Hardy–Littlewood–Pólya —; high-order generalization of Lyusternik —; implicit function —; integrality —; interlocking eigenvalue —; James sup —; Jaynes entropy concentration —; Kharitonov —; Krein–Milman —; L-separation —; Liouville —; local quadratic convergence —; Lyusternik —; M-separation —; majority —; max-flow min-cut —; Mazur–Orlicz —; Mazur–Orlicz version of the Hahn–Banach —; mean value —; metaminimax —; minimax —; Miranda fixed point —; mixed minimax —; monotone convergence —; Moreau —; Motzkin —; Motzkin transposition —; mountain pass —; parametrized Sard —; Parrott —; Perron–Frobenius —; representation —; Riesz —; right saddle-point —; Rosenbloom —; saddle duality —; saddle-minimax —; sandwich —; Savitch —; Schauder fixed point —; Schema —; Sierpinski —; Slater —; solvability —; Stiemke —; Stiemke transposition —; strong duality —; strong perfect graph —; supercritical point —; superLagrangian duality —; superminimax —; Taylor —; topological representation —; transposition —; triality —; triduality —; Tucker's —; Tucker transposition —; Tychonoff fixed point —; weak duality —; Weyl fundamental —; Yosida–Hewitt —; Zangwill —
theorem of algebra *see:* Fundamental —
theorem of the alternative
[15A39, 90C05, 90C30]
(*see:* **Farkas lemma**; **Motzkin transposition theorem**; **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**)
theorem of the alternative
[15A39, 90C05, 90C30]
(*see:* **Farkas lemma**; **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**)

theorem of the alternatives
[05B35, 90C05, 90C20, 90C33]
(*see:* **Least-index anticycling rules**)
theorem for the Hamilton–Jacobi–Bellman equation *see:* sufficiency —
theorem for linear optimization *see:* duality —
theorem of linear programming *see:* Extension of the fundamental —
theorem of natural selection *see:* fundamental —
theorem prover *see:* resolution based —
theorems *see:* Minimax —; separation —
theorems of the alternative *see:* Linear optimization: —
Theorems of the alternative and optimization
(90C05, 90C30)
(*referred to in:* **Farkas lemma**; **Image space approach to optimization**; **Linear optimization: theorems of the alternative**)
(*refers to:* **Farkas lemma**; **Farkas lemma: generalizations**; **Image space approach to optimization**; **Linear optimization: theorems of the alternative**)
theoretic framework *see:* Bayesian decision- —
theoretical
(*see:* **Global optimization: functional forms**)
theory *see:* Alternative set —; arbitrage pricing —; axioms of alternative set —; Cantor set —; complementary pivot —; Complexity —; Computational complexity —; control —; critical point —; decision —; degree —; Duality —; evolutionary game —; Fenchel–Rockafellar duality —; fixed point —; game —; geometric moment —; graph —; Interval fixed point —; Lagrangian —; location —; matroid —; Maximum entropy and game —; measure —; minimax —; moment —; Morse —; multi-attribute utility —; polyhedral —; portfolio —; Probabilistic constrained linear programming: duality —; Probabilistic constrained problems: convexity —; robust control —; scheduling —; Stackelberg game —; Standard quadratic optimization problems: —; Statistical convergence and turnpike —; Topological methods in complementarity —; triality —; triduality —; utility —
theory of algorithms *see:* complexity —
theory of automata
[01A99, 90C99]
(*see:* **Von Neumann, John**)
theory of automata
[01A99, 90C99]
(*see:* **Von Neumann, John**)
theory: biduality in nonconvex optimization *see:* Duality —
theory of CNSO problems *see:* second order Lagrangian —
theory and control *see:* systems —
theory for entropy optimization *see:* duality —
theory of envelopes
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
theory of envelops
[01A99]
(*see:* **Leibniz, gottfried wilhelm**)
theory and examples *see:* Derivatives of probability and integral functions: general —
theory of games
[01A99, 90C99]
(*see:* **Von Neumann, John**)
theory of generalized functions
[01A99]
(*see:* **Kantorovich, Leonid Vitalyevich**)
theory: monoduality in convex optimization *see:* Duality —
theory and optimality conditions *see:* Saddle point —
theory of PI-algebras *see:* complexity —
theory: quadratic programming *see:* Complexity —
theory of real addition with order *see:* first order —
theory: stability of optimal trajectories *see:* Turnpike —
theory: triduality in global optimization *see:* Duality —
therapy *see:* Optimization based frameworkfor radiation —; radiation —
thermal boundary conditions *see:* quasidifferential —; variational formulation of quasidifferential —
thermal equilibrium
[90C27, 90C90]
(*see:* **Simulated annealing**)
thermal fluctuations
[60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 92C40, 92E10]
(*see:* Global optimization in protein folding)
thermal plant
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
thermal plant
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
thermodynamic models
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
thermodynamic properties
[49K99, 65H20, 65K05, 80A10, 80A22, 90C90]
(*see:* **Global optimization: application to phase equilibrium problems**; **Optimality criteria for multiphase chemical equilibrium**)
thermodynamics
[01A99]
(*see:* **Carathéodory, Constantine**)
thermodynamics
[49K99, 65K05, 80A10, 90C30]
(*see:* **Nonlinear systems of equations: application to the enclosure of all azeotropes**; **Optimality criteria for multiphase chemical equilibrium**)
thermoelastic behavior of a generally nonhomogeneous and nonisotropic body *see:* linear —
thermoelastic model *see:* classical —
thermoelasticity
[35R70, 47S40, 74B99, 74D99, 74G99, 74H99]
(*see:* **Quasidifferentiable optimization: applications to thermoelasticity**)
thermoelasticity *see:* Quasidifferentiable optimization: applications to —
thesis *see:* parallel computation —
thickness
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
third slope lemma
[90C30]

(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
thirteenth problem *see:* Hilbert's —
thread
[90C35]
(*see:* **Generalized networks**)
three-argument function
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
three-dimensional transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)
three-dimensional transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)
three-index assignment problems
[90C35]
(*see:* **Multi-index transportation problems**)
three-index transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)
three-layer supergraph
[65K05, 90-00, 90-08, 90C11, 90C27, 90C35]
(*see:* **Algorithms for genomic analysis**)
three phase algorithm
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
three-term-recurrence
[90C30]
(*see:* **Conjugate-gradient methods**)
three-term-recurrence algorithm
[90C30]
(*see:* **Conjugate-gradient methods**)
threshold *see:* determination of clusters size —; determination of rmsd —; indifference —; preference —; unsatisfiability —; veto —
threshold accepting
[68T20, 68T99, 90C27, 90C59]
(*see:* **Metaheuristics**)
threshold accepting algorithms
[90C59]
(*see:* **Heuristic and metaheuristic algorithms for the traveling salesman problem**)
THS
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
thumb *see:* rule of —
TID
[93A30, 93B50]
(*see:* **Mixed integer linear programming: mass and heat exchanger networks**)
tie breaking rule
[90C60]
(*see:* **Complexity of degeneracy**)
tie-up time
(*see:* **Railroad crew scheduling**)
tie-up-time (l)
(*see:* **Railroad crew scheduling**)
tight constraints
[90C60]
(*see:* **Complexity of degeneracy**)
tight constraints
[90C60]
(*see:* **Complexity of degeneracy**)
tight convex underestimators *see:* Global optimization: —
tight relaxations
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)
tight relaxations
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
tightening *see:* node —
Tikhonov iterative regularization
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
Tikhonov regularization
[65Fxx]
(*see:* **Least squares problems**)
Tikhonov regularization
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
Tikhonov's regularization approach
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
tiling
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
time *see:* algorithm running in $\mathcal{O}(n^c)$ —; arr- —; completion —; cycle —; dep- —; idle —; minimization of cost/ —; on-duty —; one clause at a —; output-polynomial —; polylogarithmic —; polynomial —; radiation exposure —; strongly polynomial —; super-polynomial —; switching —; tie-up —
time algorithm *see:* efficient polynomially bounded polynomial —; exponential —; nondeterministic polynomial —; one clause at a —; polynomial —; pseudopolynomial —; strongly polynomial —; weakly polynomial —
time algorithms *see:* discrete- —; polynomial —
time analog of the dynamic programming equation *see:* continuous- —
time approach *see:* one clause at a —
time approximation scheme *see:* fully polynomial —; polynomial —
time-bounded Turing machine *see:* exponentially —; polynomially —
time coefficient generation *see:* one-at-a- —
time complexity of a deterministic Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
time complexity function
[90C60]
(*see:* **Computational complexity theory**)
time complexity function
[90C60]
(*see:* **Computational complexity theory**)
time complexity function of an algorithm
[90C60]
(*see:* **Computational complexity theory**)

time complexity of a nondeterministic Turing machine
[90C60]
(*see:* **Complexity classes in optimization**)
time computable function *see:* polynomial —
time constraints *see:* vehicle scheduling problems with —
time convergence *see:* polynomial —
time-delay system
[90C30]
(*see:* **Suboptimal control**)
time-delay systems
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
Time-dependent traveling salesman problem
(90C27)
(*referred to in:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **MINLP: trim-loss problem**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**)
(*refers to:* **Branch and price: Integer programming with column generation**; **Decomposition techniques for MILP: lagrangian relaxation**; **Graph coloring**; **Integer linear complementary problem**; **Integer programming**; **Integer programming: algebraic methods**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Integer programming: cutting plane algorithms**; **Integer programming duality**; **Integer programming: lagrangian relaxation**; **LCP: Pardalos–Rosen mixed integer formulation**; **Mixed integer classification problems**; **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**; **Multiparametric mixed integer linear programming**; **Parametric mixed integer nonlinear optimization**; **Set covering, packing and partitioning problems**; **Simplicial pivoting algorithms for integer programming**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**)
time-dependent traveling salesman problem
[90C27]
(*see:* **Time-dependent traveling salesman problem**)
time deterministic algorithm *see:* polynomial —
time discretization *see:* uniform —
time equivalent of the dynamic programming algorithm *see:* continuous- —
time formulation *see:* continuous- —
time formulations *see:* discrete- —
time horizon *see:* infinite —
time interior point methods *see:* polynomial —
time (l) *see:* tie-up- —
time local search problems *see:* polynomial —
time m *see:* algorithm solving a problem instance in —
Time Model *see:* continuous —; discrete —
time models *see:* continuous and discrete —; discrete- —
time network *see:* Space- —; weekly space- —
time optimal control
[90C30]
(*see:* **Suboptimal control**)
time optimal control *see:* continuous- —; Discrete- —; Dynamic programming: continuous- —
time optimal control problem
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
time parallelism
[49-04, 65Y05, 68N20]
(*see:* **Automatic differentiation: parallel computation**)
time problem *see:* strongly polynomial —
time ratio *see:* cost-to- —
time ratio cycle *see:* maximum profit-to- —; minimum cost-to- —
time reduction *see:* polynomial —
time replicated network
[90C30, 90C35]
(*see:* **Optimization in water resources**)
Time Representation *see:* mixed —
time Riccati equation *see:* continuous- —
time series
[90C26, 90C30]
(*see:* **Forecasting**)
time series analysis
[90C26, 90C30]
(*see:* **Forecasting**)
time slice
(*see:* **Bayesian networks**)
time solution *see:* polynomial —
time-stamped models
(*see:* **Bayesian networks**)
time-step
[90C10, 90C30, 90C35]
(*see:* **Optimization in operation of electric and energy power systems**)
time systems *see:* discrete- —
time of a Turing machine *see:* running —
time window constraints
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
time windows
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
time windows *see:* vehicle routing problem with —
times *see:* the New York —; regenerative stopping —; stochastic travel —
timetabling
[90C35]
(*see:* **Multi-index transportation problems**)
TM
[90C60]
(*see:* **Complexity theory**)

TNE
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
Todd–Ye potential function *see:* Tanabe– —
Toeplitz matrix
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
tolerance
[90C33]
(*see:* **Order complementarity**; **Peptide identification via mixed-integer optimization**)
tolerance closure of a relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
tolerance closure of a relation *see:* local —
tolerance relation *see:* local —
tolerances
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
tolerant system
[90C33]
(*see:* **Order complementarity**)
tollens *see:* checklist modus —; modus —
Tomlin penalty *see:* Driebeck– —
tomography *see:* computerized —
tON
[74A40, 90C26]
(*see:* **Shape selective zeolite separation and catalysis: optimization methods**)
tongue-and-groove constraint
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
tools *see:* parallel AD —
TOP and BOT types of logical connectives
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
topological degree
[65G20, 65G30, 65G40, 90C33]
(*see:* **Interval analysis: systems of nonlinear equations**; **Topological methods in complementarity theory**)
Topological derivative in shape optimization
(49Q10, 49Q12, 74P05, 35J85)
(*referred to in:* **Shape optimization**)
topological method
[90C33]
(*see:* **Topological methods in complementarity theory**)
topological methods
[90C33]
(*see:* **Topological methods in complementarity theory**)
Topological methods in complementarity theory
(90C33)
(*referred to in:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Linear complementarity problem**; **Order complementarity**; **Principal pivoting methods for linear complementarity problems**)
(*refers to:* **Convex-simplex algorithm**; **Equivalence between nonlinear complementarity problem and fixed point problem**; **Generalized nonlinear complementarity problem**; **Integer linear complementary problem**; **LCP: Pardalos–Rosen mixed integer formulation**; **Lemke method**; **Linear complementarity problem**; **Linear programming**; **Order complementarity**; **Parametric linear programming: cost simplex algorithm**; **Principal pivoting methods for linear complementarity problems**; **Sequential simplex method**)
topological representation theorem
[90C09, 90C10]
(*see:* **Oriented matroids**)
topological search
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: two-phase methods**)
topological separation
[93D09]
(*see:* **Robust control**)
topological space *see:* linear —
topological stability
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
topological stability in parametric programming
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
topological vector spaces *see:* Increasing and convex-along-rays functions on —; Increasing and positively homogeneous functions on —
topology
[05C05, 05C85, 68Q25, 90B80, 90C26, 90C90]
(*see:* **Bottleneck steiner tree problems**; **Structural optimization: history**)
topology
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
topology *see:* network —; ring —; strong operator —; tree —
Topology of global optimization
(90C30, 58E05)
(*referred to in:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Globally convergent homotopy methods**; **Global optimization based on statistical models**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Parametric optimization: embeddings, path following and singularities**; **Semidefinite programming and structural optimization**; **Topology optimization**)
(*refers to:* **αBB algorithm**; **Continuous global optimization: applications**; **Continuous global optimization: models, algorithms and software**; **Differential equations and global optimization**; **Direct global optimization algorithm**; **Globally convergent homotopy methods**; **Global optimization based on statistical models**; **Global optimization in binary star astronomy**; **Global optimization methods for systems of nonlinear equations**; **Global optimization using space filling**; **Parametric optimization: embeddings, path following and**

singularities; **Semidefinite programming and structural optimization**; **Structural optimization**; **Structural optimization: history**; **Topology optimization**)
Topology optimization
(90C90, 90C99)
(*referred to in:* **Semidefinite programming and structural optimization**; **Structural optimization: history**; **Topology of global optimization**)
(*refers to:* **Semidefinite programming and structural optimization**; **Structural optimization**; **Structural optimization: history**; **Topology of global optimization**)
topology optimization
[49J20, 49J52, 49M37, 65K05, 90C26, 90C30, 90C90]
(*see:* **Shape optimization**; **Structural optimization**; **Structural optimization: history**)
topology optimization
[49M37, 65K05, 90C30]
(*see:* **Structural optimization**)
topology optimization *see:* structural —
topology of transportation networks
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
toric ideal
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
toric ideal
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
toroidal butterfly
[90C35]
(*see:* **Feedback set problems**)
toroidal mesh
[90C35]
(*see:* **Feedback set problems**)
torus *see:* 2-dimensional —; d-dimensional —
total action
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: biduality in nonconvex optimization**)
total coloring
[90C35]
(*see:* **Graph coloring**)
total coloring problem
[90C35]
(*see:* **Graph coloring**)
total consistency
[90C29]
(*see:* **Estimating data for multicriteria decision making problems: optimization techniques**)
total cost function
[90C10, 90C25, 90C27, 90C35]
(*see:* **L-convex functions and M-convex functions**)
total cost infinite horizon problem
[49L20, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**)
total Gibbs free energy
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
total least squares *see:* Generalized —
total least squares problem
[65Fxx]
(*see:* **Least squares problems**)
total support
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
total variation
[90C11, 90C15]
(*see:* **Stochastic programming with simple integer recourse**)
totally acyclic oriented matroid
[90C09, 90C10]
(*see:* **Oriented matroids**)
totally asynchronous implementation of the auction algorithm
[90C30, 90C35]
(*see:* **Auction algorithms**)
totally asynchronous operation
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
totally dual integral system
[90C35]
(*see:* **Feedback set problems**)
totally unimodular
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C, 90C05, 90C09, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**; **Combinatorial optimization algorithms in resource allocation problems**; **Convex discrete optimization**)
totally unimodular matrix
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
tour
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: cutting plane algorithms**)
tour elimination constraints *see:* sub- —
tour partitioning algorithm *see:* K-iterated —
tournament *see:* bipartite —; spanning acyclic —
tours *see:* sub- —
toy problems
(*see:* **Planning in the process industry**)
Toyoda primal heuristic
[90C10, 90C27]
(*see:* **Multidimensional knapsack problems**)
TPBVP
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
TPC
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
TR
[90C30]
(*see:* **Large scale trust region problems**)
TR strategy
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
trace *see:* maximum weight —
trace of an alignment
[90C35]
(*see:* **Optimization in leveled graphs**)

trace polytope
[90C35]
(*see:* **Optimization in leveled graphs**)
trace of relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
tracing *see:* origin —
tracing infeasibilities *see:* diagnosing and —
tracing the states of plants
(*see:* **Planning in the process industry**)
track
[90C35]
(*see:* **Multi-index transportation problems**)
tracking
[34-XX, 49-XX, 65-XX, 68-XX, 90-XX]
(*see:* **Nonlocal sensitivity analysis with automatic differentiation**)
tracking *see:* multitarget —
tracking stations
[26A24, 65K99, 85-08]
(*see:* **Automatic differentiation: geometry of satellites and tracking stations**)
tracking stations *see:* Automatic differentiation: geometry of satellites and —
tractability *see:* analytical —; fixed parameter —
tractable algorithms *see:* fixed parameter —
traction *see:* surface —
trade *see:* pure —
trade economic equilibrium model *see:* pure —
trade-off
[90-XX]
(*see:* **Outranking methods**)
trade-off cutting plane
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
trade-off question
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
trade-offs
[49M37, 90C11, 90C29, 90C90]
(*see:* **MINLP: applications in the interaction of design and control**; **Multi-objective optimization: interaction of design and control**)
trading tactics
[90C27]
(*see:* **Operations research and financial markets**)
traffic assignment
[90B06, 90B20, 90C90, 91A65, 91B50, 91B99]
(*see:* **Bilevel programming: applications**; **Traffic network equilibrium**)
traffic assignment
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
traffic assignment *see:* dynamic —
traffic assignment problem
[90C30]
(*see:* **Simplicial decomposition**)
traffic assignment problem
[90C30]
(*see:* **Simplicial decomposition**)
traffic control *see:* ground delay problem in air —
traffic control and ground delay programs *see:* air —
traffic equilibrium problem *see:* standard —
Traffic network equilibrium
(90B06, 90B20, 91B50)
(*referred to in:* **Auction algorithms**; **Communication network assignment problem**; **Dynamic traffic networks**; **Equilibrium networks**; **Financial equilibrium**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Multicommodity flow problems**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Oligopolistic market equilibrium**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Spatial price equilibrium**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Walrasian price equilibrium**)
(*refers to:* **Auction algorithms**; **Communication network assignment problem**; **Directed tree networks**; **Dynamic traffic networks**; **Equilibrium networks**; **Evacuation networks**; **Financial equilibrium**; **Frank–Wolfe algorithm**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Generalized networks**; **Maximum flow problem**; **Minimum cost flow problem**; **Network design problems**; **Network location: covering problems**; **Nonconvex network flow problems**; **Oligopolistic market equilibrium**; **Piecewise linear network flow problems**; **Shortest path tree algorithms**; **Spatial price equilibrium**; **Steiner tree problems**; **Stochastic network problems: massively parallel solution**; **Survivable networks**; **Walrasian price equilibrium**)
traffic network equilibrium
[90B06, 90B15, 90B20, 91B50]
(*see:* **Dynamic traffic networks**; **Traffic network equilibrium**)
traffic network equilibrium
[90C30]
(*see:* **Equilibrium networks**)
traffic network equilibrium *see:* fixed demand —; multimodal —
traffic network equilibrium model *see:* multimodal —
traffic network equilibrium with travel disutility functions
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
traffic network model *see:* dynamic —
traffic network problems *see:* fixed demand —
traffic network problems with travel demand functions *see:* elastic demand —
traffic networks *see:* Dynamic —
traffic in transmission network *see:* routing of —
trails *see:* diverging —
train arc
(*see:* **Railroad crew scheduling**; **Railroad locomotive scheduling**)
train connection *see:* train-to- —
train connection arcs *see:* train- —

train-to-train connection
(*see:* **Railroad locomotive scheduling**)
train-train connection arcs
(*see:* **Railroad locomotive scheduling**)
training
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
training *see:* Unconstrained optimization in neural network —
training algorithms
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
training data
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90, 91B28 90C90 90C05 90C20 90C30]
(*see:* **Credit rating and optimization methods**; **Disease diagnosis: optimization-based methods**)
training a network
[90C39]
(*see:* **Neuro-dynamic programming**)
training samples
[62H30, 68T10, 90C05, 90C11]
(*see:* **Linear programming models for classification**; **Mixed integer classification problems**)
training set
[65K05, 90-08, 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90]
(*see:* **Disease diagnosis: optimization-based methods**)
trajectories *see:* Turnpike theory: stability of optimal —
trajectories and controls *see:* suboptimal —
trajectory *see:* central —; optimal —; optimization over a —; search —
trajectory and control functions *see:* asymptotically admissible pair of —
trajectory-control pair *see:* admissible —
trajectory function
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
trajectory-function and control-function *see:* admissible pair of —
tramp steamer problem
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
transcription element identification *see:* Mixed 0-1 linear programming approach for DNA —
transfer module *see:* mass/heat —
transfers *see:* principle of —
transform *see:* Cayley —; orthogonal —; principal pivotal —
transformable decision problem *see:* polynomially —
transformation *see:* canonical —; canonical dual —; code —; contradual —; convex —; coordinate —; dual —; exponential —; fast Givens —; Fenchel —; Given —; Householder —; identity —; integral Fenchel–Legendre —; inverted —; kernel —; Legendre —; linear —; logarithmic and square-root —; Markov —; modified square-root —; negation —; Piaget group of —; polynomial —; principal pivotal —; projective —; source code —; square-root —; square-root-free Givens —; unimodular max-closed form —
transformation method *see:* canonical dual —
transformation of problems
[90C60]
(*see:* **Computational complexity theory**)
transformations *see:* elementary —; elementary orthogonal —; QR factorization using Householder —; unimodular max-closed form —
transformed network
[90C35]
(*see:* **Maximum flow problem**)
transforming declarative programs *see:* symbolically —
transient class of states
[49L99]
(*see:* **Dynamic programming: average cost per stage problems**)
transient regime
[60J05, 90C15]
(*see:* **Derivatives of markov processes and their simulation**)
transients
(*see:* **Emergency evacuation, optimization modeling**)
transit planning *see:* extra-urban —
transition matrix
[93-XX]
(*see:* **Boundary condition iteration BCI**)
transition probability density
[60G35, 65K05]
(*see:* **Differential equations and global optimization**)
transition probability matrix
[49L20, 49L99, 90C39]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: discounted problems**)
transition rules of a Turing machine
[90C60]
(*see:* **Complexity theory**)
transitions
(*see:* **State of the art in modeling agricultural systems**)
transitions *see:* generic —
transitive relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
transitivity
[41A30, 47A99, 65K10]
(*see:* **Lipschitzian operators in best approximation by bounded or continuous functions**)
translation *see:* symbolic —
translation mapping technique *see:* pictogram —
transmission network *see:* routing of traffic in —
transparency
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
transport *see:* Identification methods for reaction kinetics and —
transportation
[90C15, 90C26, 90C30, 90C31, 90C33]
(*see:* **Bilevel programming: introduction, history and overview**; **Stochastic bilevel programs**)
transportation
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C11, 90C27, 90C35]
(*see:* **Multicommodity flow problems**; **Stochastic transportation and location problems**; **Vehicle scheduling**)

transportation cost
[90B80]
(*see:* **Facilities layout problems**)
transportation decisions *see:* inventory and —
transportation and location problem *see:* stochastic —
transportation and location problems *see:* Stochastic —
transportation models
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
transportation network *see:* system-optimized —; user-optimized —
transportation networks *see:* topology of —
transportation polytope *see:* k-way —
transportation problem
[90B80, 90C11, 90C35]
(*see:* **Minimum cost flow problem**; **Multi-index transportation problems**; **Stochastic transportation and location problems**)
transportation problem
[90C35]
(*see:* **Multi-index transportation problems**)
transportation problem *see:* 3D- —; axial multi-index —; capacitated —; convex integer —; fixed charge —; integer multi-index —; k-index —; minimum concave —; multi-index —; multidimensional —; planar multi-index —; stochastic —; symmetric multi-index —; three-dimensional —; three-index —
transportation problems *see:* Minimum concave —; Multi-index —
transportation systems *see:* water —
transportation terms
(*see:* **Planning in the process industry**)
transposed relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
transposition
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
transposition theorem
[15A39, 90C05, 90C30]
(*see:* **Farkas lemma**; **Linear optimization: theorems of the alternative**; **Theorems of the alternative and optimization**)
transposition theorem
[15A39, 90C05, 90C30]
(*see:* **Linear optimization: theorems of the alternative**; **Motzkin transposition theorem**; **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**)
transposition theorem *see:* Gordan —; Motzkin —; Stiemke —; Tucker —
transshipment
[90C30, 90C35]
(*see:* **Auction algorithms**)
transshipment model
[90C90, 93A30, 93B50]
(*see:* **MINLP: heat exchanger network synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**; **Mixed integer linear programming: mass and heat exchanger networks**)
transshipment model
[90C90]
(*see:* **Mixed integer linear programming: heat exchanger network synthesis**)
transshipment model *see:* expanded —
transshipment node
[90C35]
(*see:* **Minimum cost flow problem**)
transshipment problem
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
transshipment problem
[90C30, 90C35]
(*see:* **Auction algorithms**)
transshipment vertex
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
transversal of a collection of subsets
[90C09, 90C10]
(*see:* **Matroids**)
transversal intersection
[90C22, 90C25, 90C31]
(*see:* **Semidefinite programming: optimality conditions and stability**)
transversal matroid
[90C09, 90C10]
(*see:* **Matroids**)
transversal to zero
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
trap-door point
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
trap-door point
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
trapezoid graph
[90C35]
(*see:* **Feedback set problems**)
travel *see:* light —
travel behavior *see:* day-to-day dynamic —
travel demand *see:* elastic —; fixed —
travel demand functions *see:* elastic demand traffic network problems with —
travel disutility functions *see:* traffic network equilibrium with —
travel times *see:* stochastic —
Traveling purchaser problem
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
traveling salesman *see:* probabilistic —
Traveling salesman problem
(90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90)
(*referred to in:* **Domination analysis in combinatorial optimization**)
(*refers to:* **Domination analysis in combinatorial optimization**; **Evolutionary algorithms in combinatorial optimization**; **Heuristic and metaheuristic algorithms for the traveling salesman problem**)

traveling salesman problem
[05C05, 05C40, 68R10, 90C05, 90C06, 90C08, 90C10, 90C11, 90C27, 90C30, 90C35, 90C57]
(*see:* **Assignment and matching**; **Integer programming**; **Integer programming: cutting plane algorithms**; **Integer programming: lagrangian relaxation**; **Network design problems**)
traveling salesman problem *see:* classical —; graphical —; Heuristic and metaheuristic algorithms for the —; prize collecting —; road —; Steiner graphical —; Time-dependent —
Traveling Salesman Problem (ATSP) *see:* asymmetric —
traveling salesman problems
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and cut algorithms**)
traveling salesperson problem
[90C60]
(*see:* **Computational complexity theory**)
Traverso algorithm *see:* Conti– —
treatment design *see:* Beam selection in radiotherapy —
tree
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
tree *see:* additive —; attributed —; binary —; decision —; directed —; empty —; full Steiner —; game —; index —; k- —; level of a vertex in a rooted —; min-max Steiner —; minimax —; minimum ratio spanning- —; minimum spanning —; one- —; rectilinear Steiner —; rectilinear Steiner arborescence —; root of a —; rooted —; scenario —; spanning —; Steiner —; Steiner minimal —; Steiner minimum —
tree algorithm *see:* parallel minimax —; sequential minimax game —
tree algorithms *see:* Shortest path —
tree association graph
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
tree association graph *see:* weighted —
tree constraints
[90C09, 90C10]
(*see:* **Combinatorial optimization algorithms in resource allocation problems**)
tree decomposition
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)
tree dissection
[90C15]
(*see:* **Stochastic programming: parallel factorization of structured matrices**)
tree networks *see:* Directed —
tree node
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
tree nodes set
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
tree problem *see:* bounded degree minimum spanning —; capacitated minimum spanning —; minimum spanning —; resource-constrained minimum spanning —; single source shortest path —; Steiner minimal —
tree problem with minimum number of Steiner points *see:* Steiner —
tree problems *see:* Bottleneck steiner —; shortest path —; Steiner —
tree search
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
tree search
[68W10, 90C27]
(*see:* **Load balancing for parallel optimization techniques**)
tree search *see:* best-first —; depth-first —; Parallel Best-First —; Parallel Depth-First —
tree search algorithm *see:* distributed game —; generalized game —
tree searching *see:* Minimax game —
tree solution *see:* spanning —
tree-splitting algorithm
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
tree structure *see:* feasible spanning —; optimal spanning —; spanning —
tree topology
[05C85]
(*see:* **Directed tree networks**)
trees *see:* asymptotic behavior of CAP on —; barycentric scenario —; binary —; bottleneck Steiner —; CAP on —; Capacitated minimum spanning —; classification and regression —; exact algorithm for solving CAP on —; heuristic approach to solving CAP on —; minimum spanning —; parallelizing the exploration of minimax —; variations of Steiner —
treewidth
[68R10, 90C27]
(*see:* **Branchwidth and branch decompositions**)
trial point
[49M07, 49M10, 65K, 90C06, 90C20]
(*see:* **Spectral projected gradient methods**)
trial step
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
trial steplength *see:* compute a safeguarded new —
triality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
triality theorem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
triality theory
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
triangle *see:* Cholesky —
triangle inequality
[05C05, 05C40, 68R10, 90B06, 90B35, 90C06, 90C10, 90C27, 90C35, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Multi-index transportation problems**; **Network design problems**; **Traveling salesman problem**)
triangle inequality *see:* strengthen —
triangle product *see:* fuzzy —
triangularization *see:* Orthogonal —

triangulate
(*see:* **Semidefinite programming and the sensor network localization problem, SNLP**)
triangulation
[13Cxx, 13Pxx, 14Qxx, 68Q20, 90C05, 90C10, 90Cxx]
(*see:* **Integer programming: algebraic methods**; **Optimal triangulations**; **Simplicial pivoting algorithms for integer programming**)
triangulation *see:* boundary —; delaunay —; greedy —; Hickey–Cohen —; minimum weight —; minimum weight Steiner —; regular —
triangulation of Euclidean space
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
triangulation problem for input-output tables
[90C10, 90C11, 90C20]
(*see:* **Linear ordering problem**)
triangulations *see:* Optimal —; pseudo- —; regular —; regular family of —
tricanonical forms
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
tridiagonal matrix
[65K05, 90Cxx]
(*see:* **Symmetric systems of linear equations**)
triduality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
triduality
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
triduality in global optimization *see:* Duality theory: —
triduality theorem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
triduality theory
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: triduality in global optimization**)
trim loss
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
trim-loss problem
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
trim-loss problem
[90C11, 90C90]
(*see:* **MINLP: trim-loss problem**)
trim-loss problem *see:* MINLP: —; numerical example of a —
trinomial distribution
[90C15]
(*see:* **Logconcavity of discrete distributions**)
trip *see:* passenger —; pull-in —; pull-out —
trip-route choice adjustment process
[90B15]
(*see:* **Dynamic traffic networks**)
trip-route choice adjustment process
[90B15]
(*see:* **Dynamic traffic networks**)
trip shifting *see:* Vehicle scheduling with —
triplet
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**)
triplet
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of the Hessian**)
triplet *see:* sparse —
triplets
[05C85]
(*see:* **Directed tree networks**)
trisection
[65K05]
(*see:* **Direct global optimization algorithm**)
tRT
(*see:* **Integrated planning and scheduling**)
truckload *see:* less-than- —
true continuum
[90C30]
(*see:* **Conjugate-gradient methods**)
truncated Buchberger algorithm
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
truncated Newton
[65K05, 90C30]
(*see:* **Automatic differentiation: calculation of Newton steps**)
truncated Newton method
[90C06]
(*see:* **Large scale unconstrained optimization**)
truncated Newton method
[90C25, 90C30]
(*see:* **Successive quadratic programming: solution by active sets and interior point methods**)
truncated Newton method *see:* discrete —
truncated Newton software package *see:* block —
truncated singular value decomposition solution
[65Fxx]
(*see:* **Least squares problems**)
truncated Taylor approximation
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
truss
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
truss *see:* elastic bar of a —; node of a —
truss design
[90C25, 90C27, 90C90]
(*see:* **Semidefinite programming and structural optimization**)
truss design *see:* multiload —; robust obstacle-free —
trust region
[49M37, 90C30]
(*see:* **Large scale trust region problems**; **Nonlinear least squares problems**; **Nonlinear least squares: trust region methods**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
trust region
[90C30]
(*see:* **Nonlinear least squares problems**; **Successive quadratic programming**)

trust region *see:* bundle —; large scale —
trust region approach
[90C30]
(*see:* **Numerical methods for unary optimization**)
trust region method
[90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**)
trust region method
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
trust region methodology
[49M37, 65K05, 65K10, 90C30, 93A13]
(*see:* **Multilevel methods for optimal design**)
trust region methods
[49M37]
(*see:* **Nonlinear least squares: trust region methods**)
trust region methods *see:* Nonlinear least squares: —
trust region model
[65F10, 65F50, 65H10, 65K10]
(*see:* **Multidisciplinary design optimization**)
trust region model
[90C20, 90C25]
(*see:* **Quadratic programming over an ellipsoid**)
trust region problem
[90C60]
(*see:* **Complexity theory: quadratic programming**)
trust region problem
[90C60]
(*see:* **Complexity theory: quadratic programming**)
trust region problem *see:* general case of the —; hard case of the —; large scale —; Newton step case of the —
trust region problems *see:* Large scale —
trust region strategy *see:* pure —
trust region technique
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
trust regions
[90C30]
(*see:* **Broyden family of methods and the BFGS update**)
trustworthy space
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
truth assessment *see:* fuzzy —
truth-functional
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
TS
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
TS *see:* reactive —; static —; strict —
Tsallis probability distribution
[90C90]
(*see:* **Simulated annealing methods in protein folding**)
tSP
[05-04, 68Q25, 68R10, 68W40, 90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Domination analysis in combinatorial optimization**; **Evolutionary algorithms in combinatorial optimization**; **Traveling salesman problem**)
TSP *see:* asymmetric —; euclidean —; generalized —; m- —; max —; symmetric —
TSP (ATSP) *see:* asymmetric —
TSP (STSP) *see:* symmetric —
TSVSP
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
Tucker approach *see:* Kuhn– —
Tucker, A.W.
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
Tucker conditions *see:* generalized Karush–Kuhn– —; Karush–Kuhn– —; Kuhn– —
Tucker conditions for quadratic programming sub-problems *see:* Kuhn– —
Tucker CQ *see:* Kuhn– —
Tucker equations *see:* Kantorovich–Karush–Kuhn– —; Kuhn– —
Tucker homogeneous systems of linear relations
(15A39, 90C05)
(*referred to in:* **Farkas lemma**; **Linear optimization: theorems of the alternative**; **Linear programming**; **Motzkin transposition theorem**)
(*refers to:* **Farkas lemma**; **Linear optimization: theorems of the alternative**; **Linear programming**; **Motzkin transposition theorem**)
Tucker necessary optimality conditions *see:* Kuhn– —
Tucker optimality condition *see:* Kuhn– —
Tucker optimality conditions *see:* Karush–Kuhn– —; Kuhn– —
Tucker point *see:* Karush–Kuhn– —; Kuhn– —
Tucker points *see:* multiple Kuhn– —; multiple QP Kuhn– —
Tucker's theorem
[15A39, 90C05, 90C30]
(*see:* **Farkas lemma**; **Theorems of the alternative and optimization**)
Tucker transposition theorem
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
Tucker type condition *see:* Karush–Kuhn– —
tumors *see:* breast —
tuning
(*see:* **Bayesian networks**)
tuning parameter
[65K05]
(*see:* **Direct global optimization algorithm**)
tunneling method
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
turbine balancing problem
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
Turing machine
[90C20, 90C25, 90C60]
(*see:* **Complexity theory**; **Complexity theory: quadratic programming**; **Quadratic programming over an ellipsoid**)
Turing machine
[90C60]
(*see:* **Complexity theory**)
Turing machine *see:* accepting computation of a —; accepting state of a —; alternating —; control state of a —; deterministic —; execution of a —; exponentially space-bounded —; exponentially time-bounded —; final

state of a —; input alphabet of a —; language accepted by a —; length of a partial computation of a —; logspace —; move of a —; nonaccepting computation of a —; nondeterministic —; partial computation of a —; polynomially space-bounded —; polynomially time-bounded —; running time of a —; size of the input of a —; space complexity of a deterministic —; space complexity of a nondeterministic —; start state of a —; state of a —; tape of a —; tape cell of a —; time complexity of a deterministic —; time complexity of a nondeterministic —; transition rules of a —
Turing machine model
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
Turing machine solving a problem
[90C60]
(*see:* **Complexity theory**)
Turing machines *see:* complexity of —
Turing reducibility *see:* polynomial —
turning point
[90C31, 90C34]
(*see:* **Parametric global optimization: sensitivity**)
turning point *see:* quadratic —
turnpike theory *see:* Statistical convergence and —
Turnpike theory: stability of optimal trajectories
(49J24, 35B40, 37C70)
(*referred to in:* **Statistical convergence and turnpike theory**)
(*refers to:* **Statistical convergence and turnpike theory**)
twice codifferentiable
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
twice codifferentiable function
[49J52, 65K99, 70-08, 90C25]
(*see:* **Quasidifferentiable optimization: codifferentiable functions**)
twice continuously codifferentiable
[65Kxx, 90Cxx]
(*see:* **Quasidifferentiable optimization: algorithms for QD functions**)
twice continuously codifferentiable function
[49J52, 65K99, 70-08, 90C25]
(*see:* **Quasidifferentiable optimization: codifferentiable functions**)
twice-differentiable function *see:* piecewise —
twice-differentiable MINLPs
[65K05, 90C11, 90C26]
(*see:* **MINLP: global optimization with αBB**)
Twice-differentiable NLPs
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)
twice-differentiable part of a function
[90Cxx]
(*see:* **Discontinuous optimization**)
Twilt *see:* problem regular in the sense of Jongen–Jonker– —
twinplex
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
two affine functions *see:* program of minimizing a product of —
two cardinalities axiom
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
two-dimensional marginal probability distribution function
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
two-function minimax inequality
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
two-hop neighbors
(*see:* **Broadcast scheduling problem**)
two-layer feed-forward network
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
Two-level Optimization
(*see:* **Mixed integer nonlinear bilevel programming: deterministic global optimization**)
two-parameter CG family
[90C30]
(*see:* **Conjugate-gradient methods**)
two paths *see:* Pivoting algorithms for linear programming generating —
two-person game
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
two-person game *see:* cooperative case of a —
two-person zero-sum game
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 62C20, 90C15, 91A05]
(*see:* **Minimax theorems**; **Stochastic programming: minimax approach**)
two-phase
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C10, 90C26, 90C30, 90C35]
(*see:* **Bi-objective assignment problem**; **Stochastic global optimization: two-phase methods**)
two-phase method
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: two-phase methods**)
two-phase methods *see:* Stochastic global optimization: —
two-phase procedure
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
two-player zero-sum perfect-information game
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
two-point boundary value problem
[34A55, 65L99, 78A60, 90C30, 93-XX]
(*see:* **Optimal design in nonlinear optics**; **Optimization strategies for dynamic systems**)
two-point boundary value problem *see:* ODE —
two Polygons Arrangement
(*see:* **State of the art in modeling agricultural systems**)
two-stage model *see:* linear —
two-stage stochastic linear program
[90C15, 90C90]
(*see:* **Chemical process planning**; **Two-stage stochastic programs with recourse**)

two-stage stochastic program with recourse
[90C15]
(*see:* **Stochastic programming: parallel factorization of structured matrices**)
two-stage stochastic programming
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
two-stage stochastic programming models
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
Two-stage stochastic programming models
[90C15]
(*see:* **Two-stage stochastic programming: quasigradient method**)
two-stage stochastic programming problem *see:* dynamic —
Two-stage stochastic programming: quasigradient method
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programs with recourse**)
(*refers to:* **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic quasigradient methods**; **Stochastic quasigradient methods: applications**; **Stochastic quasigradient methods in minimax problems**; **Two-stage stochastic programs with recourse**)
Two-stage stochastic programs with recourse
(90C15)
(*referred to in:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**)
(*refers to:* **Approximation of extremum problems with probability functionals**; **Approximation of multivariate probability integrals**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Extremum problems with probability functions: kernel type solution methods**; **General moment optimization problems**; **Logconcave measures, logconvexity**; **Logconcavity of discrete distributions**; **L-shaped method for two-stage stochastic programs with recourse**; **Multistage stochastic programming: barycentric approximation**; **Preprocessing in stochastic programming**; **Probabilistic constrained linear programming: duality theory**; **Probabilistic constrained problems: convexity theory**; **Semi-infinite programming: discretization methods**; **Simple recourse problem: dual method**; **Simple recourse problem: primal method**; **Stabilization of cutting plane algorithms for stochastic linear programming problems**; **Static stochastic programming models**; **Static stochastic programming models: conditional expectations**; **Stochastic integer programming: continuity, stability, rates of convergence**; **Stochastic integer programs**; **Stochastic linear programming: decomposition and cutting planes**; **Stochastic linear programs with recourse and arbitrary multivariate distributions**; **Stochastic network problems: massively parallel solution**; **Stochastic programming: minimax approach**; **Stochastic programming models: random objective**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic**

programming with simple integer recourse; **Stochastic programs with recourse: upper bounds**; **Stochastic quasigradient methods in minimax problems**; **Stochastic vehicle routing problems**; **Two-stage stochastic programming: quasigradient method**)
two-stage stochastic programs with recourse
[90C06, 90C15]
(*see:* **Stochastic linear programming: decomposition and cutting planes**)
two-stage stochastic programs with recourse *see:* L-shaped method for —
two-stage stochastic programs with simple integer recourse
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)
two-stranded chain
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
two updates *see:* rank- —
Tychonoff fixed point theorem
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
type *see:* abstract variational inequality of elliptic —; homotopy —; single locomotive —
type a *see:* gas lift wells of —; naturally flowing wells of —
type A well
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
type A wells
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
type algorithm *see:* Craig conjugate gradient —; simplex —; SQP —
type approximation *see:* Padé- —
type b *see:* gas lift wells of —; naturally flowing wells of —
type B well
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
type B wells
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
type condition *see:* Fritz John —; Karush–Kuhn–Tucker —
type duality for M- and L-convex functions *see:* Fenchel- —
type function *see:* max- —; maximum- —; min- —
type functions *see:* difference of max- —; Lagrange- —
Type I requirement
[90C26]
(*see:* **Invexity and its applications**)
type iteration *see:* Gram–Schmidt —
type lower bounds *see:* Gilmore–Lawler —
type many-valued logic system *see:* lattice- —
type method *see:* Newton- —
type methods *see:* Krylov space —; Nonlinear least squares: Newton- —
type models *see:* multiple locomotive —
type neighborhood structure for the QAP *see:* K-L —
type of optimization
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
type solution methods *see:* Extremum problems with probability functions: kernel —
type variable
(*see:* **Bayesian networks**)
types *see:* crew —; model —
types of logical connectives *see:* TOP and BOT —
types of vehicles *see:* Vehicle scheduling problems with multiple —

U

U-concave function
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
U-continuous function
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
U-pseudoconcave function
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
U-quasiconcave function
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
U-quasiconcave function *see:* int —; Luc —
U^0-quasiconcave function
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
U-weakly pseudoconcave function
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
UAF of CEP
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
uBD
[49M37, 90C11]
(*see:* **Mixed-integer nonlinear optimization: A disjunctive cutting plane approach**)
UFVS
[90C35]
(*see:* **Feedback set problems**)
ultrametric
[62H30, 90C27, 90C39]
(*see:* **Assignment methods in clustering**; **Dynamic programming in clustering**)
ultrametric
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
umbrella function
[41A30, 62J02, 90C26]
(*see:* **Regression by special functions: algorithms and complexity**)
umbrella regression *see:* quasiconvex and —
unary length
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
unary operations on relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)

unary optimization
[90C30]
(*see:* **Numerical methods for unary optimization**)
unary optimization
[90C30]
(*see:* **Numerical methods for unary optimization**)
unary optimization *see:* Numerical methods for —
unary optimization problem
[90C30]
(*see:* **Numerical methods for unary optimization**)
unavoidable edges
[68Q20]
(*see:* **Optimal triangulations**)
unbounded
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
unbounded controls
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
unbounded controls and non standard methods
[03H10, 49J27, 90C34]
(*see:* **Semi-infinite programming and control problems**)
unbounded cost
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
unbounded domains *see:* global optimization over —
unbounded optimization
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**; **Random search methods**)
unbounded program
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
unbounded solution
[90C29]
(*see:* **Multicriteria sorting methods**)
uncapacitated center
[90C26]
(*see:* **MINLP: application in facility location-allocation**)
uncapacitated facility location problem
[90B10, 90B80, 90C10, 90C11, 90C27, 90C35, 90C57]
(*see:* **Integer programming**; **Network location: covering problems**)
uncapacitated facility location problem
[90B10, 90B80, 90C35]
(*see:* **Network location: covering problems**)
uncapacitated network flow problem
[90B10]
(*see:* **Piecewise linear network flow problems**)
uncapacitated plant location problem
[90B80, 90B85]
(*see:* **Warehouse location problem**)
uncapacitated static multifacility *see:* discrete single-commodity single-criterion —
uncertain
[37N40, 90C30, 90C34]
(*see:* **Robust design of dynamic systems by constructive nonlinear dynamics**)
uncertain information
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
uncertain systems
[93D09]
(*see:* **Robust control**)
uncertainties
[93D09]
(*see:* **Robust control**)
uncertainty
[68Q25, 90B85, 90C26, 90C29, 90C70, 91B28, 94A17]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Competitive ratio for portfolio management**; **Fuzzy multi-objective linear programming**; **Jaynes' maximum entropy principle**; **Single facility location: multi-objective rectilinear distance location**)
uncertainty
[90C15, 90C26, 90C29, 94A17]
(*see:* **Bilevel optimization: feasibility test and flexibility index**; **Discretely distributed stochastic programs: descent directions and efficient points**; **Global optimization in batch design under uncertainty**; **Jaynes' maximum entropy principle**; **Two-stage stochastic programs with recourse**)
uncertainty *see:* Bilevel programming framework for enterprise-wide process networks under —; budget of —; decision making under —; design under —; disaggregation under —; fictitious —; Global optimization in batch design under —; measure of —; multi-objective linear programming under —; nonstochastic —; probabilistic —; process synthesis and design under —; Production planning under —
uncertainty considerations
(*see:* **Selection of maximally informative genes**)
uncertainty, duality and applications *see:* Robust linear programming with right-hand-side —
uncertainty in dynamical systems *see:* estimating —
uncertainty embedded in a probability distribution
[90C25, 94A17]
(*see:* **Entropy optimization: shannon measure of entropy and its properties**)
uncertainty on hydrological exogenous inflow and demand *see:* water resources planning under —
uncertainty modeling
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
uncertainty with perturbations *see:* minimax observation problem under —
uncertainty: sensitivity analysis *see:* Short-term scheduling under —
uncertainty set
[93D09]
(*see:* **Robust control**)
unclassifiable examples
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
unclassifiable examples
[90C09, 90C10]
(*see:* **Optimization in boolean classification problems**)
unconstrained and constrained optimization *see:* Interval analysis: —
unconstrained dual in entropy optimization
[90C25, 90C51, 94A17]
(*see:* **Entropy optimization: interior point methods**)

unconstrained global optimization
[65T40, 90C26, 90C30, 90C90]
(*see:* **Global optimization methods for harmonic retrieval**)
unconstrained implicit Lagrangian
[90C30, 90C33]
(*see:* **Implicit lagrangian**)
unconstrained minimization
[90C26, 90C39]
(*see:* **Second order optimality conditions for nonlinear optimization**)
unconstrained minimization
[49M29, 65K10, 90C06, 90C30]
(*see:* **Conjugate-gradient methods**; **Local attractors for gradient-related descent iterations**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
unconstrained minimization *see:* algorithms for —
unconstrained nonlinear least squares problem
[49M37]
(*see:* **Nonlinear least squares: Newton-type methods**)
Unconstrained nonlinear optimization: Newton–Cauchy framework
(90C30)
(*referred to in:* **Automatic differentiation: calculation of Newton steps**; **Broyden family of methods and the BFGS update**; **Conjugate-gradient methods**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Interval Newton methods**; **Large scale unconstrained optimization**; **Nondifferentiable optimization: Newton method**; **Nonlinear least squares: Newton-type methods**; **Numerical methods for unary optimization**; **Unconstrained optimization in neural network training**)
(*refers to:* **Automatic differentiation: calculation of Newton steps**; **Broyden family of methods and the BFGS update**; **Dynamic programming and Newton's method in unconstrained optimal control**; **Interval Newton methods**; **Large scale unconstrained optimization**; **Nondifferentiable optimization: Newton method**; **Numerical methods for unary optimization**; **Unconstrained optimization in neural network training**)
unconstrained optimal control
[49M29, 65K10, 90C06]
(*see:* **Dynamic programming and Newton's method in unconstrained optimal control**)
unconstrained optimal control *see:* Dynamic programming and Newton's method in —
unconstrained optimization
[65F10, 65F50, 65H10, 65K05, 65K10, 90C30]
(*see:* **ABS algorithms for optimization**; **Broyden family of methods and the BFGS update**; **Globally convergent homotopy methods**)
unconstrained optimization
[65K05, 68T05, 90C06, 90C30, 90C52, 90C53, 90C55]
(*see:* **Broyden family of methods and the BFGS update**; **Large scale unconstrained optimization**; **Unconstrained optimization in neural network training**)
unconstrained optimization *see:* branch and bound for —; Large scale —; New hybrid conjugate gradient algorithms for —; Performance profiles of conjugate-gradient algorithms for —
unconstrained optimization algorithms
[65G20, 65G30, 65G40, 65K05, 90C30]
(*see:* **Interval global optimization**)
Unconstrained optimization in neural network training
(90C30, 90C30, 90C52, 90C53, 90C55, 65K05, 68T05)
(*referred to in:* **Broyden family of methods and the BFGS update**; **Conjugate-gradient methods**; **Large scale unconstrained optimization**; **Neural networks for combinatorial optimization**; **Neuro-dynamic programming**; **Numerical methods for unary optimization**; **Replicator dynamics in combinatorial optimization**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
(*refers to:* **Automatic differentiation: introduction, history and rounding error estimation**; **Broyden family of methods and the BFGS update**; **Conjugate-gradient methods**; **Large scale unconstrained optimization**; **Least squares problems**; **Neural networks for combinatorial optimization**; **Neuro-dynamic programming**; **Numerical methods for unary optimization**; **Replicator dynamics in combinatorial optimization**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
unconstrained optimization problem
[65K05, 90C26, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**; **Discontinuous optimization**; **Smooth nonlinear nonconvex optimization**)
unconstrained optimization problem *see:* global —
unconstrained optimum
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
unconstrained optimum
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
unconstrained optimum *see:* conditions for an —
unconstrained problem
[90C31]
(*see:* **Sensitivity and stability in NLP**)
undefined
[49M29, 65K10, 90C06]
(*see:* **Local attractors for gradient-related descent iterations**)
under control *see:* rounding errors are —
under extreme events *see:* decision making —
under network constraints *see:* optimization —
under principal pivoting *see:* matrix class invariant —
under probabilistic constraint *see:* programming —
under uncertainty *see:* Bilevel programming framework for enterprise-wide process networks —; decision making —; design —; disaggregation —; Global optimization in batch design —; multi-objective linear programming —; process synthesis and design —; Production planning —
under uncertainty on hydrological exogenous inflow and demand *see:* water resources planning —
under uncertainty with perturbations *see:* minimax observation problem —
under uncertainty: sensitivity analysis *see:* Short-term scheduling —
under weak assumptions
[57R12, 90C31, 90C34]
(*see:* **Smoothing methods for semi-infinite optimization**)

underdetermined system of nonlinear equations
[90C30]
(*see:* **Nonlinear least squares problems**)
underestimation *see:* convex global —; Molecular structure determination: convex global —
underestimation matrix *see:* diagonal —
underestimator
[90C26]
(*see:* **Global optimization in multiplicative programming**)
underestimator *see:* convex —; convex global —; global —; local —
underestimators *see:* feasible —; Global optimization: tight convex —
underlying deterministic problem
[90C05, 90C15]
(*see:* **Probabilistic constrained linear programming: duality theory**)
underlying deterministic problem
[90C15]
(*see:* **Static stochastic programming models**)
underlying matroid
[90C09, 90C10]
(*see:* **Oriented matroids**)
underlying method *see:* single —
underprojection
[90C30]
(*see:* **Relaxation in projection methods**)
undirected multicommodity network flow models
[90B10, 90C05, 90C06, 90C35]
(*see:* **Nonoriented multicommodity flow problems**)
undiscounted problem
[49L20, 90C39, 90C40]
(*see:* **Dynamic programming: infinite horizon problems, overview**)
undiscounted problems *see:* Dynamic programming: —
unfeasibility criterion *see:* minimum —
Unfolding *see:* maximum Variance —
UniCalc
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**)
unichain policy
[49L99]
(*see:* **Dynamic programming: average cost per stage problems**)
unicursal graph
[90B06]
(*see:* **Vehicle routing**)
unidimensional Euclidean representation
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
unidimensional scale
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
unidimensional scale *see:* circular —; linear —
unidimensional scales
[62H30, 90C27]
(*see:* **Assignment methods in clustering**)
unidimensional scaling
[62H30, 90C39]
(*see:* **Dynamic programming in clustering**)
UNIFAC equation
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
unification *see:* set —
unified algorithm
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
unified modeling frameworks *see:* Short-term scheduling, resource constrained: —
uniform
[49M37, 65K10, 90B36, 90C26, 90C30]
(*see:* **αBB algorithm**; **Maximum cut problem, MAX-CUT**; **Stochastic scheduling**)
uniform angle condition
[49J20, 49J52]
(*see:* **Shape optimization**)
uniform computations
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
uniform cone property
[49J20, 49J52]
(*see:* **Shape optimization**)
uniform distribution
[52A22, 60D05, 68Q25, 90C05, 90C27, 90C90]
(*see:* **Probabilistic analysis of simplex algorithms**; **Simulated annealing**)
uniform distribution
[90C11, 90C15]
(*see:* **Stochastic programming with simple integer recourse**)
uniform dose (eud) *see:* equivalent —
uniform extension
[49J20, 49J52]
(*see:* **Shape optimization**)
uniform fractional combinatorial optimization
[68Q25, 68R05, 90-08, 90C27, 90C32]
(*see:* **Fractional combinatorial optimization**)
uniform grid technique
[90C26]
(*see:* **Global optimization using space filling**)
uniform Hölder conditions
[90C26]
(*see:* **Global optimization using space filling**)
uniform matroid
[90C09, 90C10]
(*see:* **Matroids**)
uniform norm *see:* approximation in the —
uniform P-function
[90C30, 90C33]
(*see:* **Implicit lagrangian**)
uniform random sampling
[65C30, 65C40, 65C50, 65C60, 65Cxx, 65K05, 90C26, 90C30]
(*see:* **Stochastic global optimization: two-phase methods**)
uniform sequential coloring
[05-XX]
(*see:* **Frequency assignment problem**)
uniform time discretization
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)

uniformity, conformity
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
uniformly convex function
[65F10, 65F50, 65H10, 65K10]
(*see:* **Globally convergent homotopy methods**)
uniformly differentiable function *see:* Dini —
uniformly directionally differentiable function *see:* Dini —
unilateral boundary value problem
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
unilateral contact problem with friction *see:* coupled —
unilateral frictional contact *see:* Signorini-Coulomb —
unilateral growth condition
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)
unilateral growth condition
[35A15, 47J20, 49J40]
(*see:* **Hemivariational inequalities: static problems**)
unilateral mechanics
[49J40]
(*see:* **Nonconvex-nonsmooth calculus of variations**)
unilateral mechanics
[49J40, 49J52]
(*see:* **Hemivariational inequalities: eigenvalue problems**; **Nonconvex-nonsmooth calculus of variations**)
unimodular *see:* totally —
unimodular matrix
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)
unimodular matrix *see:* totally —
unimodular max-closed form transformation
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
unimodular max-closed form transformations
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
union
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
UNIQUAC equation
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
unique
[12D10, 12Y05, 13P10, 34-xx, 34Bxx, 34Lxx, 60J15, 60J60, 60J70, 60K35, 65C05, 65C10, 65C20, 68U20, 70-08, 82B21, 82B31, 82B41, 82B80, 90C05, 90C10, 90C22, 90C25, 90C31, 92C40, 92E10, 93E24]
(*see:* **Complexity and large-scale least squares problems**; Global optimization in protein folding; **Gröbner bases for polynomial equations**; **LP strategy for interval-Newton method in deterministic global optimization**; **Semidefinite programming: optimality conditions and stability**; **Simplicial pivoting algorithms for integer programming**)
unique path
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
uniqueness *see:* Variational inequalities: geometric interpretation, existence and —
uniqueness problem *see:* entry- —
uniqueness of solutions of nonlinear systems of equations
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)
unit-Disk Graphs
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
unit-disk graphs *see:* bisectored —; Optimization problems in —
unit-resolution *see:* single-lookahead- —
unit-specific
(*see:* **Medium-term scheduling of batch processes**)
unit weight CMST
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
united extension
[65H99, 65K99]
(*see:* **Automatic differentiation: point and interval**)
units *see:* building blocks for the process —
units problem *see:* minimum- —
unity demand
[90B80, 90B85]
(*see:* **Warehouse location problem**)
univalent relation
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
univariate discrete probability distribution *see:* logconcave —
univariate gradient free
[90C30]
(*see:* **Powell method**)
univariate interval Newton method
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)
univariate interval Newton operator
[65G20, 65G30, 65G40, 65H20, 65K99]
(*see:* **Interval Newton methods**)
univariate linear model *see:* general —
universal
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
universal class
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
universal cover
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements**)
universal Gröbner basis
[05A, 13Cxx, 13Pxx, 14Qxx, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C, 90Cxx]
(*see:* **Convex discrete optimization**; **Integer programming: algebraic methods**)
universal Gröbner basis
[13Cxx, 13Pxx, 14Qxx, 90Cxx]
(*see:* **Integer programming: algebraic methods**)

universal portfolio
[68Q25, 91B28]
(*see:* **Competitive ratio for portfolio management**)
universal properties of relations
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
universal wedge
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
universality
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
universally accessible form of CEP
[49K99, 65K05, 80A10]
(*see:* **Optimality criteria for multiphase chemical equilibrium**)
unknown information
[68Q25, 91B28]
(*see:* **Competitive ratio for portfolio management**)
unknown variables *see:* initializing —
unlimited intermediate storage
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
unnormalized fuzziness
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
unpaired element *see:* left- —; right- —
unpinned
[51K05, 52C25, 68Q25, 68U05, 90C22, 90C35]
(*see:* Graph realization via semidefinite programming)
unsatisfiability threshold
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
unsealed ABS class
[65K05, 65K10]
(*see:* **ABS algorithms for linear equations and linear least squares**)
unsplitting of load *see:* splitting/ —
unstructured optimization
[90C06]
(*see:* **Large scale unconstrained optimization**)
unsupervised
[90C26, 90C56, 90C90]
(*see:* **Nonsmooth optimization approach to clustering**)
unsupervised classification
[90C90]
(*see:* **Optimization in medical imaging**)
unsupported nondominated solution
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
unused partitions *see:* set L_{free} of —; set R_{free} of —
unweighted feedback vertex set problem
[90C35]
(*see:* **Feedback set problems**)
unweighted problem in OR
[90B80, 90B85]
(*see:* **Warehouse location problem**)
unyielding configuration
[51K05, 52C25, 68Q25, 68U05, 90C22, 90C35]
(*see:* Graph realization via semidefinite programming)
unyielding tensegrity
[51K05, 52C25, 68Q25, 68U05, 90C22, 90C35]
(*see:* Graph realization via semidefinite programming)
UONNT
[65K05, 68T05, 90C30, 90C52, 90C53, 90C55]
(*see:* **Unconstrained optimization in neural network training**)
up to first order changes
[90Cxx]
(*see:* **Discontinuous optimization**)
up penalty
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
update *see:* BFGS —; BFGS quasi-Newton —; Broyden family of methods and the BFGS —; Broyden–Fletcher–Goldfarb–Shanno —; Broyden–Fletcher–Goldfarb–Shanno quasi-Newton —; Davidon–Fletcher–Powell —; DFP —; quasi-Newton —; SR1 —; symmetric rank-one —
update formula *see:* Sherman-Morrison rank-one —
update Newton method *see:* partial- —
updates *see:* quasi-Newton —; rank-two —
updating *see:* fractional —; inverse quasi-Newton —; partial —; secant —
updating formula *see:* Moré —
updating input-output matrices
[90C35]
(*see:* **Multi-index transportation problems**)
updating rule *see:* momentum —
upper bandwidth
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
upper bound
[90B35, 90C10]
(*see:* **Job-shop scheduling problem**; **Maximum constraint satisfaction: relaxations and upper bounds**)
upper bound
[90C15]
(*see:* **Stochastic programs with recourse: upper bounds**)
upper bound *see:* Edmundson–Madansky —; Hunter–Worsley —; parametric —; piecewise linear —; polynomial —; valid —
upper-bound dichotomy *see:* generalized- —
upper bound on gas lift availability
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
upper bound for a set
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
upper-bound test
[49M37, 90C11]
(*see:* **Mixed integer nonlinear programming**)
upper boundary
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)

upper boundary point
[65K05, 90C26, 90C30]
(*see:* **Monotonic optimization**)
upper bounding
[49M37, 65K10, 90C26, 90C30]
(*see:* **αBB algorithm**)
upper bounding structure *see:* generalized —
upper bounds
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
upper bounds *see:* Maximum constraint satisfaction: relaxations and —; Stochastic programs with recourse: —
upper bounds constraints *see:* generalized —; lower and —
upper bounds for multivariate probability integrals
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
upper derivative
[65K05, 90Cxx]
(*see:* **Dini and Hadamard derivatives in optimization**)
upper derivative *see:* Dini —; Dini conditional —; Hadamard conditional —
upper directional derivative *see:* Dini —; Hadamard —
upper directional derivatives *see:* lower and —
upper envelope
[90C26]
(*see:* **Global optimization: envelope representation**)
upper hemicontinuous operator
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
upper level problem
[90C26, 90C31, 90C34, 91A65]
(*see:* **Bilevel programming: implicit function approach**; **Parametric global optimization: sensitivity**)
upper and lower bounds *see:* parametric —
upper and lower bounds to eigenvalues
[49R50, 65G20, 65G30, 65G40, 65L15, 65L60]
(*see:* **Eigenvalue enclosures for ordinary differential equations**)
upper and lower well oil rate constraints
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
upper problem
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
upper semicontinuous function
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
upper semicontinuous function *see:* $\mathbf{R}^n_+$- —
upper semismooth function
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
upper set
[62G07, 62G30, 65K05]
(*see:* **Isotonic regression problems**)
ups *see:* well-defined start- —
urban transit planning *see:* extra- —
URV factorization *see:* rank revealing —
use of groundwater *see:* rational —
use of PI-algebras
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
use of water resource systems *see:* conjunctive —
used partitions *see:* set L_{reac} of —; set R_{reac} of —
user interface *see:* graphical —
user-optimization
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
user-optimized transportation network
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
user-optimizing environment
[90B80, 90B85, 90Cxx, 91Axx, 91Bxx]
(*see:* **Facility location with externalities**)
using flexible templates *see:* De novo protein design —
using a heuristic parameter, reject index for interval optimization *see:* Algorithmic improvements —
using Householder transformations *see:* QR factorization —
using MINLP *see:* HEN synthesis —
using pseudocosts *see:* best estimate —
using pseudoshadow prices *see:* best estimate —
using space filling *see:* Global optimization —
using (sub)gradients parametric representations *see:* necessary optimality condition without —
using terrain/funneling methods *see:* Multi-scale global optimization —
UTA *see:* meta- —
UTA method
[90C29, 91A99]
(*see:* **Preference disaggregation**)
UTASTAR algorithm
[90C29, 91A99]
(*see:* **Preference disaggregation**)
utilité
[90C05, 90C90, 91B28]
(*see:* **Multicriteria methods for mergers and acquisitions**)
utilities *see:* consumption of —
utility
[90C29, 90C90, 91A65, 91B99]
(*see:* **Bilevel programming: applications**; **Multiple objective programming support**)
utility
[90C29]
(*see:* **Preference modeling**)
utility function
[90-01, 90B30, 90B50, 90C29, 91B32, 91B52, 91B74]
(*see:* **Bilevel programming in management**; **Preference disaggregation approach: basic features, examples from financial decision making**)
utility function *see:* coordinatewise increasing —; implicit —; social —
utility functions *see:* additive —; estimation of —
utility theory
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
utility theory
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
utility theory *see:* multi-attribute —

V

V *see:* vertex set —
V algebra
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
v at *see:* insertion of vertex —
(v)-differentiable family of measures *see:* weakly L_1 —
V-invex
[90C26]
(*see:* **Invexity and its applications**)
V-invex
[90C26]
(*see:* **Invexity and its applications**)
V-representation
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
valid cut
[90C26]
(*see:* **Cutting plane methods for global optimization**)
valid inequalities
[90C09, 90C10, 90C11]
(*see:* **Disjunctive programming**)
valid inequality *see:* nondominated —
valid lower bound
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)
valid upper bound
[90C10, 90C26]
(*see:* **MINLP: branch and bound global optimization algorithm**)
validation *see:* cross- —; model —
validation sample
[91B28 90C90 90C05 90C20 90C30]
(*see:* **Credit rating and optimization methods**)
validity conditions
[90C35]
(*see:* **Maximum flow problem**)
valuation
[03E70, 03H05, 91B16]
(*see:* **Alternative set theory**)
valuation of a checklist
[03B50, 03B52, 03C80, 62F30, 62Gxx, 68T27]
(*see:* **Checklist paradigm semantics for fuzzy logics**)
valuation structure *see:* coarse —; fine —
value *see:* best —; continuity property of the objective function —; convexity property of the objective function —; critical —; incumbent —; incumbent objective —; limiting —; marginal —; maximize net present —; mean —; minimal —; minimax —; positive marginal —; positive minimum —; regular —; saddle —; settle- —; structured singular —
value analysis
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
value analysis
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
value bounds *see:* computable optimal —
Value for Composite Convexifiable Function *see:* integral Mean- —
value conditions *see:* boundary —
value convergence tests
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
value cross decomposition *see:* mean —
value cuts
[49M27, 90C11, 90C30]
(*see:* **MINLP: generalized cross decomposition**)
value decomposition solution *see:* truncated singular —
value extension *see:* mean —
value of a flow
[90C35]
(*see:* **Maximum flow problem**)
value formula *see:* marginal —
value function
[90C10, 90C15, 90C29, 90C46]
(*see:* **Integer programming duality**; **Multi-objective optimization; Interactive methods for preference value functions**; **Multiple objective programming support**; **Multistage stochastic programming: barycentric approximation**)
value function
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**; **Multiple objective programming support**)
value function *see:* expected —; mean —; mixed integer —; optimal —; preference —
value function approach
[90-XX]
(*see:* **Outranking methods**)
value functions *see:* Multi-objective optimization; Interactive methods for preference —; optimal —
value iteration
[49L20, 49L99, 90C39, 90C40]
(*see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: discounted problems**; **Dynamic programming: infinite horizon problems, overview**; **Dynamic programming: stochastic shortest path problems**; **Dynamic programming: undiscounted problems**)
value iteration *see:* Gauss–Seidel —; relative —
value of a network flow
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
value of perfect information *see:* expected —
value problem *see:* bilateral boundary —; initial —; mean —; ODE two-point boundary —; two-point boundary —; unilateral boundary —
value of stochastic solution
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)
value theorem *see:* mean —
valued analysis *see:* set- —
valued approximate inference *see:* interval- —
valued boundary laws and variational equalities *see:* single- —

valued CNSO *see:* extended real- —; real- —
valued constraints *see:* set- —
valued families of the Pinkava logic algebras *see:* many- —
valued function *see:* Boolean 2- —
valued Lagrangian *see:* vector —
valued logic *see:* evaluation in multiple- —
valued logic algebra *see:* Boolean 2- —; many- —
valued logic algebras *see:* Finite complete systems of many- —
valued logic implication *see:* many- —
valued logic normal form *see:* complete many- —
valued logic system *see:* lattice-type many- —
valued logics *see:* classification of many- —; many- —; taxonomy of the PI-algebras of many- —
valued maps *see:* Generalized monotone single —
valued normal form *see:* many- —
valued normal forms *see:* PI-algebras and 2- —
valued objective function *see:* set- —
valued optimization *see:* Set- —
valued optimization problem *see:* set- —
valued PI-systems *see:* subfamilies of n- —
valued relation
[90C29]
(*see:* **Preference modeling**)
values *see:* negative marginal —; positive marginal —; set of formation —
VAM
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
VAM construction procedure *see:* mixed —
vapor phases
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
vapor pressure *see:* Reid —
variable
[90C10, 90C30]
(*see:* **Modeling languages in optimization: a new paradigm**)
variable *see:* auxiliary —; choice of the entering —; choice of the leaving —; consistent —; distinguished —; dual —; eligible nonbasic —; entering —; flow decision —; leaving —; most/least infeasible integer —; multiple branches for bounded integer —; strongly determined —; type —
variable assignment
[90C60]
(*see:* **Complexity theory**)
variable coefficients *see:* generalized linear programming with —
variable cost
[90C25]
(*see:* **Concave programming**)
variable dichotomy
[90C05, 90C06, 90C08, 90C10, 90C11]
(*see:* **Integer programming: branch and bound methods**)
variable factor programming
[49M29, 90C11]
(*see:* **Generalized benders decomposition**)
variable formulation of SP *see:* split- —
variable metric
[58E05, 90C30]
(*see:* **Topology of global optimization**)
variable metric bundle method
[49J40, 49J52, 65K05, 90C30]
(*see:* **Solving hemivariational inequalities by nonsmooth optimization methods**)
variable metric method
[90C30]
(*see:* **Unconstrained nonlinear optimization: Newton–Cauchy framework**)
variable metric methods
[90C26]
(*see:* **Smooth nonlinear nonconvex optimization**)
variable neighborhood descent
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
Variable neighborhood search methods
(9008, 90C59, 90C27, 90C26)
(*referred to in:* **Maximum cut problem, MAX-CUT**)
variable precision interval package
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: differential equations**)
variable representation *see:* splitting —
variable space
[90C29]
(*see:* **Multiple objective programming support**)
variable stage-length
[93-XX]
(*see:* **Dynamic programming: optimal control applications**)
variable-storage
[90C30]
(*see:* **Conjugate-gradient methods**)
variable-storage algorithm
[90C30]
(*see:* **Conjugate-gradient methods**)
variables
(*see:* **Planning in the process industry**; **Short-term scheduling under uncertainty: sensitivity analysis**)
variables
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
variables *see:* adjoint —; basic —; binary —; Boolean —; complementary pair of —; complicating —; constraints on —; control —; decision —; dependent —; design —; discrete —; discrete design —; dual —; eliminating blocks of —; full space of x —; Generalized geometric programming: mixed continuous and discrete free —; independent —; initializing unknown —; integer —; intermediate —; key —; nonbasic —; separable problem —; sizing —; splitting —; structural —; superbasic —
variables model *see:* errors-in- —
variables x *see:* decision —
variance
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
variance *see:* mean- —; one-way analysis of —
variance clustering *see:* minimal —
variance model *see:* Portfolio selection: markowitz mean- —
variance of the number of shadow-vertices
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
variance optimization problems *see:* Decomposition algorithms for the solution of multistage mean- —

variance portfolio analysis *see:* mean- —
variance reduction
[65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 90C15]
(*see:* **Approximation of multivariate probability integrals**)
variance reduction *see:* Monte-Carlo sampling and —
variance reduction lower bounds
[90C08, 90C11, 90C27, 90C57, 90C59]
(*see:* **Quadratic assignment problem**)
variance reduction technique
[62F12, 65C05, 65C30, 65C40, 65C50, 65C60, 65Cxx, 65D30, 65K05, 90C15, 90C31]
(*see:* **Approximation of multivariate probability integrals**; **Monte-Carlo simulations for stochastic optimization**)
Variance Unfolding *see:* maximum —
variant of the constraint-by-constraint method *see:* lexicographic —
variant of the simplex algorithm
[52A22, 60D05, 68Q25, 90C05]
(*see:* **Probabilistic analysis of simplex algorithms**)
variants of GBD
[49M29, 90C11]
(*see:* **Generalized benders decomposition**)
variates *see:* control —
variation *see:* total —
variation of an eigenvalue of an interval matrix *see:* interval of —
variation of the interval Newton method *see:* Krawczyk —
variation path *see:* principal —
variation splitting algorithm *see:* principal —
variation of a system
[65K05, 90C30]
(*see:* **Bisection global optimization methods**)
variation technique *see:* boundary —
variational equalities *see:* single-valued boundary laws and —
variational equality for an elastostatic problem involving QD-superpotentials
[49J40, 49M05, 49S05, 74G99, 74H99, 74Pxx]
(*see:* **Quasidifferentiable optimization: variational formulations**)
variational formulation *see:* mixed —
variational formulation of quasidifferential laws
[49J40, 49M05, 49S05, 74G99, 74H99, 74Pxx]
(*see:* **Quasidifferentiable optimization: variational formulations**)
variational formulation of quasidifferential thermal boundary conditions
[35R70, 47S40, 74B99, 74D99, 74G99, 74H99]
(*see:* **Quasidifferentiable optimization: applications to thermoelasticity**)
variational formulation of subdifferential laws
[49J40, 49M05, 49S05, 74G99, 74H99, 74Pxx]
(*see:* **Quasidifferentiable optimization: variational formulations**)
variational formulations *see:* Quasidifferentiable optimization: —
variational-hemivariational inequality
[49J40, 49J52, 49Q10, 70-XX, 74K99, 74Pxx, 80-XX]
(*see:* **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**)
variational-hemivariational inequality
[49J40]
(*see:* **Nonconvex-nonsmooth calculus of variations**)
variational inclusion
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
Variational inequalities
(65K10, 65M60)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Global optimization methods for systems of nonlinear equations**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Implicit lagrangian**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Optimization with equilibrium constraints: A piecewise SQP approach**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities: F. E. approach**;

Variational inequalities: geometric interpretation, existence and uniqueness; **Variational inequalities: projected dynamical system**; **Variational principles**)
variational inequalities
[26B25, 26E25, 46A22, 49J35, 49J40, 49J52, 49Q10, 49S05, 54D05, 54H25, 55M20, 65K99, 70-08, 74A55, 74G99, 74H99, 74K99, 74M10, 74M15, 74Pxx, 90C26, 90C30, 90C33, 90C90, 90C99, 91A05, 91A65]
(*see:* **Hemivariational inequalities: applications in mechanics**; **Minimax theorems**; **Multilevel optimization in mechanics**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: applications**; **Quasivariational inequalities**)
variational inequalities
[46N10, 49J40, 49M05, 49Q10, 49S05, 65K10, 70-08, 74G99, 74H99, 74K99, 74Pxx, 90C06, 90C25, 90C26, 90C30, 90C31, 90C33, 90C35]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Simplicial decomposition algorithms**)
variational inequalities *see:* approximation of —; multivalued monotone laws and —; Nonsmooth and smoothing methods for nonlinear complementarity problems and —; parametric —; QD laws and systems of —; scalar —; Solution methods for multivalued —; system of —; vector —
variational inequalities: A brief review *see:* Generalized —
variational inequalities and equilibrium problems *see:* Generalized monotonicity: applications to —
Variational inequalities: F. E. approach
(65M60)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**; **Variational principles**)
Variational inequalities: geometric interpretation, existence and uniqueness
(65K10, 65M60)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: projected dynamical system**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**;

Hemivariational inequalities: static problems; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: projected dynamical system**; **Variational principles**)

Variational inequalities: projected dynamical system
(65K10, 90C90)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational principles**)
(*refers to:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational principles**)

variational inequalities and quasivariational inequalities *see:* implicit —

variational inequality
[47H05, 47J20, 49J40, 49J52, 49M37, 49Q10, 65J15, 65K10, 70-XX, 74K99, 74Pxx, 80-XX, 90C25, 90C26, 90C33, 90C55, 91A10]
(*see:* **Bilevel programming**; **Fejér monotonicity in convex optimization**; **Nonconvex energy functions: hemivariational inequalities**; **Solution methods for multivalued variational inequalities**)

variational inequality *see:* generalized —; mixed —

variational inequality for an elastostatic problem involving QD-superpotentials *see:* convex —

variational inequality of elliptic type *see:* abstract —

variational inequality formulation
[90B06, 90B20, 90C30, 91B06, 91B28, 91B50, 91B60]
(*see:* **Equilibrium networks**; **Financial equilibrium**; **Oligopolistic market equilibrium**; **Spatial price equilibrium**; **Traffic network equilibrium**; **Walrasian price equilibrium**)

variational inequality formulation
[91B50]
(*see:* **Financial equilibrium**; **Walrasian price equilibrium**)

variational inequality formulation in link loads
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)

variational inequality formulation in path flows
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)

variational inequality formulations
[65K10, 65M60, 90B06, 90B20, 91B06, 91B28, 91B50, 91B60]
(*see:* **Oligopolistic market equilibrium**; **Spatial price equilibrium**; **Traffic network equilibrium**; **Variational inequalities**)

variational inequality problem
[49J52, 65K10, 65M60, 90C06, 90C15, 90C25, 90C26, 90C30, 90C33, 90C35, 90C90]
(*see:* **Cost approximation algorithms**; **Nondifferentiable optimization: Newton method**; **Simplicial decomposition algorithms**; **Stochastic bilevel programs**; **Variational inequalities**; **Variational inequalities: projected dynamical system**)

variational inequality problem
[49J52, 90C15, 90C26, 90C30, 90C33]

(*see:* **Cost approximation algorithms**; **Nondifferentiable optimization: Newton method**; **Stochastic bilevel programs**)
variational inequality problem *see:* dual —; finite-dimensional —; parametric —; vector —
variational inequality problem and a projected dynamical system
[65K10, 90C90]
(*see:* **Variational inequalities: projected dynamical system**)
variational inequality problems
[90C33]
(*see:* **Linear complementarity problem**)
variational inequality problems *see:* Sensitivity analysis of —
variational-like inequalities
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
variational-like inequalities
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
variational methods
[65L99, 93-XX]
(*see:* **Optimization strategies for dynamic systems**)
Variational Principle *see:* ekeland —; subdifferential —
Variational principles
(62H30, 65Cxx, 65C30, 65C40, 65C50, 65C60, 90C05, 49J40)
(*referred to in:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**)
(*refers to:* **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: algorithms for QD functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: optimality conditions**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Sensitivity analysis of variational inequality problems**; **Solving hemivariational inequalities by nonsmooth optimization methods**; **Variational inequalities**; **Variational inequalities: F. E. approach**; **Variational inequalities: geometric interpretation, existence and uniqueness**; **Variational inequalities: projected dynamical system**)
variational principles
[49J40, 49J52, 49K27, 49M05, 49S05, 58C20, 58E30, 74G99, 74H99, 74Pxx, 90C33, 90C48]
(*see:* **Hemivariational inequalities: applications in mechanics**; **Nonsmooth analysis: Fréchet subdifferentials**; **Quasidifferentiable optimization: variational formulations**)
variational principles
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
variational problem *see:* ill-posed —
variational problems
[03H10, 49J27, 49K05, 49K10, 49K15, 49K20, 90C34]
(*see:* **Duality in optimal control with first order differential equations**; **Semi-infinite programming and control problems**)
variational problems
[49J40, 49M05, 49S05, 74G99, 74H99, 74Pxx]
(*see:* **Quasidifferentiable optimization: variational formulations**)
variational problems *see:* Ill-posed —; implicit —
variational sensitivity
[90C90]
(*see:* **Design optimization in computational fluid dynamics**)
variations *see:* calculus of —; inverse problem of the calculus of —; Nonconvex-nonsmooth calculus of —
variations of Steiner trees
[90C27]
(*see:* **Steiner tree problems**)
variations of Steiner trees
[90C27]
(*see:* **Steiner tree problems**)
varying dimension pivoting algorithms
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
Vasicek model with impulse perturbations
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
VDSP
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
vector
[90C09, 90C10]
(*see:* **Oriented matroids**)

vector *see:* arc length —; augmenting —; characteristic —; charactertstic —; control —; descent —; differential cost —; dual —; feasible flow —; feasible high-order approximating —; feature —; generic cost —; geodesic gradient —; gradient —; high-order tangent approximating —; incidence —; Lagrange multiplier —; lexico-positive —; lexicographically positive —; position —; primitive integral —; proposal —; reference direction —; residual —; sign —; state —; steepest descent —; support of an integral —; support of a natural —; weight —; weighted characteristic —
vector configuration
[90C09, 90C10]
(*see:* **Oriented matroids**)
vector of decrease *see:* high-order approximating —
vector estimates for parametric NLPS *see:* Bounds and solution —
vector forward automatic differentiation
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
vector generalized concavity
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
vector geometry
[26A24, 65K99, 85-08]
(*see:* **Automatic differentiation: geometry of satellites and tracking stations**)
vector inequality
[90C29, 90C30]
(*see:* **Multi-objective optimization: lagrange duality**)
vector iteration *see:* control —
vector iteration CVI *see:* Control —
vector labeling
[90C05, 90C10]
(*see:* **Simplicial pivoting algorithms for integer programming**)
vector lattice
[90C33]
(*see:* **Equivalence between nonlinear complementarity problem and fixed point problem**; **Order complementarity**; **Topological methods in complementarity theory**)
vector lattice
[90C33]
(*see:* **Order complementarity**)
vector machine *see:* generalized eigenvalue proximal support —
vector machine problem *see:* Generalized eigenvalue proximal support —
vector machines *see:* support —
vector maximization method
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
vector minimization
(*see:* **Planning in the process industry**)
Vector optimization
(90C29)
(*referred to in:* **Composite nonsmooth optimization**; **Image space approach to optimization**)
(*refers to:* **Image space approach to optimization**)
vector optimization
[49K27, 65K05, 90B50, 90C05, 90C29, 90C48, 91B06]
(*see:* **Multi-objective optimization and decision support systems**; **Set-valued optimization**)
vector optimization
[46A20, 49K27, 52A01, 65K05, 90B50, 90C05, 90C29, 90C30, 90C48, 91B06]
(*see:* **Composite nonsmooth optimization**; **Multi-objective optimization and decision support systems**; **Set-valued optimization**)
vector of an oriented matroid
[90C09, 90C10]
(*see:* **Oriented matroids**)
vector space
[14R10, 15A03, 51N20]
(*see:* **Linear space**)
vector space *see:* optimization in a —
vector space model
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
vector space model
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
vector-space models for entropy optimization for image reconstruction
[94A08, 94A17]
(*see:* **Maximum entropy principle: image reconstruction**)
vector spaces *see:* Increasing and convex-along-rays functions on topological —; Increasing and positively homogeneous functions on topological —; ordered —
vector valued Lagrangian
[90C29, 90C30]
(*see:* **Multi-objective optimization: lagrange duality**)
Vector variational inequalities
(58E35, 90C29)
vector variational inequalities
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
vector variational inequality problem
[46N10, 49J40, 90C26]
(*see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**)
vectorial matroid
[90C09, 90C10]
(*see:* **Matroids**)
vectors
[14R10, 15A03, 51N20]
(*see:* **Linear space**)
vectors *see:* equivalent cost —; high-order approximating —; lexicographical ordering for n-dimensional —; ordering on binary —
vectors space *see:* real —
vegetative
(*see:* **State of the art in modeling agricultural systems**)
vehicle block
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
vehicle and duty scheduling problems *see:* Integrated —
Vehicle routing
(90B06)

(*referred to in:* **General routing problem**; **Stochastic vehicle routing problems**; **Vehicle scheduling**)
(*refers to:* **General routing problem**; **Stochastic vehicle routing problems**; **Vehicle scheduling**)
vehicle routing
[90-02, 90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**; **Operations research models for supply chain management and design**)
vehicle routing
[90C10, 90C15]
(*see:* **Stochastic vehicle routing problems**)
vehicle routing problem
[05-04, 90B06, 90C10, 90C11, 90C15, 90C27, 90C57]
(*see:* **Evolutionary algorithms in combinatorial optimization**; **Set covering, packing and partitioning problems**; **Stochastic vehicle routing problems**; **Vehicle routing**)
vehicle routing problem *see:* capacitated —; distance-constrained —; dynamic —; Metaheuristic algorithms for the —; stochastic —
vehicle routing problem with backhauls
[90B06]
(*see:* **Vehicle routing**)
Vehicle routing problem with simultaneous pickups and deliveries
(00-02, 01-02, 03-02)
vehicle routing problem with time windows
[90B06]
(*see:* **Vehicle routing**)
vehicle routing problems *see:* approximate methods for solving —; constructive methods for solving —; exact methods for solving —; Stochastic —
Vehicle scheduling
(90B06, 90B35, 68M20, 90C27, 90B80, 90B10, 90C10)
(*referred to in:* **Airline optimization**; **General routing problem**; **Integer programming**; **Job-shop scheduling problem**; **MINLP: design and scheduling of batch processes**; **Stochastic vehicle routing problems**; **Vehicle routing**)
(*refers to:* **Airline optimization**; **Complexity classes in optimization**; **Complexity theory**; **General routing problem**; **Integer programming**; **Job-shop scheduling problem**; **MINLP: design and scheduling of batch processes**; **Stochastic scheduling**; **Stochastic vehicle routing problems**; **Vehicle routing**)
vehicle scheduling problem
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
vehicle scheduling problem *see:* Multi-depot —; Single-depot —
vehicle scheduling problems *see:* multi-depot —; Single-depot —
Vehicle scheduling problems with a fixed number of vehicles
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
Vehicle scheduling problems with multiple types of vehicles
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
vehicle scheduling problems with time constraints
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
Vehicle scheduling with trip shifting
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
vehicles *see:* number of —; Vehicle scheduling problems with a fixed number of —; Vehicle scheduling problems with multiple types of —
vehicles' homogeneity/heterogeneity
[00-02, 01-02, 03-02]
(*see:* **Vehicle routing problem with simultaneous pickups and deliveries**)
venture capital investment
[91B06, 91B60]
(*see:* **Financial applications of multicriteria analysis**)
verification
[34-xx, 34Bxx, 34Lxx, 90C60, 93E24]
(*see:* **Complexity and large-scale least squares problems**; **Computational complexity theory**)
verification
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: intermediate terms**; **Interval analysis: nondifferentiable problems**)
verification *see:* automatic result —
verifying equilibrium solutions
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
verifying feasibility *see:* Interval analysis: —
version of the Hahn–Banach theorem *see:* Mazur–Orlicz —
versus Multiperiod Models *see:* single —
versus noninteractive methods *see:* interactive —
vertex
[05B35, 20F36, 20F55, 52C35, 57N65]
(*see:* **Hyperplane arrangements in optimization**)
vertex *see:* child of a —; co-optimal —; improper —; parent of a —; required —; shadow- —; transshipment —
vertex algorithm *see:* shadow- —
vertex (arc) deletion problem
[90C35]
(*see:* **Feedback set problems**)
vertex (arc) set problem *see:* minimum feedback —; subset feedback —; subset minimum feedback —
VERTEX COVER
[03B50, 05C15, 05C62, 05C69, 05C85, 68Q25, 68R10, 68T15, 68T30, 68W40, 90C27, 90C59, 90C60]
(*see:* **Complexity classes in optimization**; **Domination analysis in combinatorial optimization**; **Finite complete systems of many-valued logic algebras**; **Optimization problems in unit-disk graphs**)
vertex cover *see:* minimum weighted —
Vertex Cover Problem *see:* minimum —
vertex of a digraph
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
vertex of a graph
[05C05, 05C40, 68R10, 90C35]
(*see:* **Network design problems**)
vertex insertion algorithm *see:* generic —
vertex insertion (FVI) *see:* farthest —
vertex insertion (NVI) *see:* nearest —
vertex insertion (RVI) *see:* random —

vertex insertion (VI)
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
vertex matrix of an interval matrix
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: eigenvalue bounds of interval matrices**)
vertex packing
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
vertex in a rooted tree *see:* level of a —
vertex set
[49-01, 49K45, 49N10, 65K05, 90-01, 90C20, 90C26, 90C27, 90C30, 91B52]
(*see:* **Bilevel linear programming: complexity, equivalence to minmax, concave programs**; **Monotonic optimization**)
vertex set *see:* feedback —; minimum feedback —
vertex set problem *see:* feedback —; minimum weighted feedback —; unweighted feedback —
vertex set V
[90C35]
(*see:* **Graph coloring**)
vertex subset *see:* subgraph induced by a —
vertex v at *see:* insertion of —
vertex weights
[05C15, 05C17, 05C35, 05C69, 90C22, 90C35]
(*see:* **Lovász number**)
vertical linear complementarity problem
[90C33]
(*see:* **Linear complementarity problem**)
vertices
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
vertices *see:* expected number of shadow- —; intersatured —; variance of the number of shadow- —
vertices in a graph *see:* adjacent —
very strong homomorphism
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
Vessel Problem (SVP) *see:* seismic —
veto threshold
[90-XX]
(*see:* **Outranking methods**)
(VI) *see:* vertex insertion —
via linear optimization *see:* Distance dependent protein force field —
via mixed-integer linear optimization *see:* Global pairwise protein sequence alignment —
via mixed-integer optimization *see:* Multi-class data classification —; Peptide identification —
via negative fitness *see:* genetic engineering —
via parametric programming *see:* Design of robust model-based controllers —
via semidefinite programming *see:* Graph realization —; Maximum likelihood detection —
via (w,i) *see:* subdivision —
Vienna Fortran
[05-02, 05-04, 15A04, 15A06, 68U99]
(*see:* **Alignment problem**)
view *see:* beam's-eye- —
view of linear semi-infinite programming *see:* perfect duality from the —
violating point *see:* index of a constraint —
violation of constraints
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
violators algorithm *see:* pool adjacent —
VIP
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
virtual
[34-xx, 34Bxx, 34Lxx, 93E24]
(*see:* **Complexity and large-scale least squares problems**)
virtual depot
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
virtual displacements
[49J52, 49S05, 74G99, 74H99, 74Pxx, 90C33]
(*see:* **Hemivariational inequalities: applications in mechanics**)
virtual processor
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
virtual source
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
virtual source algorithm
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
virtual source concept in auction algorithms
[90B10, 90C27]
(*see:* **Shortest path tree algorithms**)
virtual work *see:* principle of —
visual binary star
[90C26, 90C90]
(*see:* **Global optimization in binary star astronomy**)
visual binary star *see:* spectroscopic —
visual inference
[90C26, 90C30]
(*see:* **Forecasting**)
visual interaction
[90C29, 90C70]
(*see:* **Fuzzy multi-objective linear programming**)
visual interactive method
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
visualization *see:* Optimization-based —
Vitalyevich *see:* Kantorovich, Leonid —
VLSI routing
[05C05, 05C85, 68Q25, 90B80]
(*see:* **Bottleneck steiner tree problems**)
vND
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)

VND
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
vNDS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
VNDS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
vNFSS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
vNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
VNS
[9008, 90C26, 90C27, 90C59]
(*see:* **Variable neighborhood search methods**)
VNS *see:* basic —; bI- —; fH- —; j- —; Parallel —; pD- —; Reduced —; Skewed —
vocabulary *see:* indexing —; optimal —; optimal indexing —
Vogel approximation method
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
Volterra filter
[90C26, 90C90]
(*see:* **Signal processing with higher order statistics**)
Volterra model of conflicting populations
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: differential equations**)
volume
[90C05]
(*see:* **Continuous global optimization: applications**)
volume
[52B11, 52B45, 52B55]
(*see:* **Volume computation for polytopes: strategies and performances**)
volume *see:* integral over a —; logarithmic —; normalized —
Volume computation for polytopes: strategies and performances
(52B11, 52B45, 52B55)
(*referred to in:* **Ellipsoid method**; **Quadratic programming over an ellipsoid**)
(*refers to:* **Quadratic programming over an ellipsoid**)
volume ellipsoid *see:* maximum- —; minimum- —
volume formula *see:* integral over —
volumetric method
[49M20, 90-08, 90C25]
(*see:* **Nondifferentiable optimization: cutting plane methods**)
von Neumann algebra
[01A99, 90C99]
(*see:* **Von Neumann, John**)
von Neumann architecture
[01A99, 90C99]
(*see:* **Von Neumann, John**)
Von Neumann, John
(01A99, 90C99)
(*referred to in:* **Duality theory: biduality in nonconvex optimization**; **Duality theory: triduality in global optimization**; **History of optimization**)
(*refers to:* **Duality theory: biduality in nonconvex optimization**; **Duality theory: monoduality in convex optimization**; **Duality theory: triduality in global optimization**; **History of optimization**)
von Stackelberg game
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
von Stackelberg games
[90C25, 90C29, 90C30, 90C31]
(*see:* **Bilevel programming: optimality conditions and duality**)
Voronoi diagram
[68Q20, 90B80, 90C27]
(*see:* **Optimal triangulations**; **Voronoi diagrams in facility location**)
Voronoi diagram
[90B80, 90B85, 90C27]
(*see:* **Multifacility and restricted location problems**; **Voronoi diagrams in facility location**)
Voronoi diagram *see:* farthest-point —
Voronoi diagrams
[90B85, 90C27]
(*see:* **Single facility location: circle covering problem**)
Voronoi diagrams in facility location
(90C27, 90B80)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Warehouse location problem**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Warehouse location problem**)
Voronoi edge
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)

Voronoi point
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
Voronoi region
[90B80, 90C27]
(*see:* **Voronoi diagrams in facility location**)
voting
[55R15, 55R35, 65K05, 90C11]
(*see:* **Deterministic and probabilistic optimization models for data classification**)
vous communication *see:* rendez- —
voxels
[90C90]
(*see:* **Optimization in medical imaging**)
VRP
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
VRPB
[90B06]
(*see:* **Vehicle routing**)
VRPTW
[05-04, 90C27]
(*see:* **Evolutionary algorithms in combinatorial optimization**)
VSM
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
VSM
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
VSP
[68M20, 90B06, 90B10, 90B35, 90B80, 90C10, 90C27]
(*see:* **Vehicle scheduling**)
VSP *see:* p- —

W

(w,i) *see:* subdivision via —
w-weighted Tchebycheff metric
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
W∗-algebra
[01A99, 90C99]
(*see:* **Von Neumann, John**)
wait *see:* zero- —
wait-and-see
[90C15]
(*see:* **Two-stage stochastic programs with recourse**)
waits *see:* younger brother —
Walford-one algorithm *see:* Smith– —
Walford one-reducible graph *see:* Smith– —
walk
[90C35]
(*see:* **Minimum cost flow problem**)
walk *see:* directed —
walk search *see:* random —
Walker method *see:* overdetermined Yule– —
Wallenius procedure *see:* Zionts– —
Walras law
[91B50]
(*see:* **Walrasian price equilibrium**)
Walras law
[91B50]
(*see:* **Walrasian price equilibrium**)
Walrasian price equilibrium
(91B50)
(*referred to in:* **Equilibrium networks**; **Financial equilibrium**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Oligopolistic market equilibrium**; **Spatial price equilibrium**; **Traffic network equilibrium**)
(*refers to:* **Equilibrium networks**; **Financial equilibrium**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Oligopolistic market equilibrium**; **Spatial price equilibrium**; **Traffic network equilibrium**)
Walrasian price equilibrium
[91B50]
(*see:* **Walrasian price equilibrium**)
Wang algorithm *see:* Goldfarb– —
Wardrop first principle
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
Wardrop second principle
[90B06, 90B20, 91B50]
(*see:* **Traffic network equilibrium**)
Warehouse location problem
(90B80, 90B85)
(*referred to in:* **Combinatorial optimization algorithms in resource allocation problems**; **Facilities layout problems**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**)
(*refers to:* **Combinatorial optimization algorithms in resource allocation problems**; **Competitive facility location**; **Facility location with externalities**; **Facility location problems with spatial interaction**; **Facility location with staircase costs**; **Global optimization in Weber's problem with attraction and repulsion**; **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**; **Network location: covering problems**; **Optimizing facility location with euclidean and rectilinear distances**; **Production-distribution system design problem**; **Resource allocation for epidemic control**; **Single facility location: circle covering problem**; **Single facility location: multi-objective euclidean distance location**; **Single facility location: multi-objective rectilinear distance location**; **Stochastic transportation and location problems**; **Voronoi diagrams in facility location**)

warehouse location problem
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
warmstart *see:* advanced —
Wastewater system, optimization of
(76D55)
water capacity constraints *see:* maximum oil, gas and —
water demand
[90C30, 90C35]
(*see:* **Optimization in water resources**)
water environment *see:* minimizing the degradation in quality of both —
water pumping facilities *see:* surface —
water resource planning
[90C30, 90C35]
(*see:* **Optimization in water resources**)
water resource systems *see:* conjunctive use of —
water resources *see:* Optimization in —; stochastic approach to optimization in —
water resources planning under uncertainty on hydrological exogenous inflow and demand
[90C30, 90C35]
(*see:* **Optimization in water resources**)
water resources policies *see:* nonanticipative —; nonanticipativity —
water storage capacity *see:* nodes with —
water transportation systems
[90C30, 90C35]
(*see:* **Optimization in water resources**)
wavelength-division multiplexing
[05C85]
(*see:* **Directed tree networks**)
wavelengths *see:* assignment of —
way analysis of variance *see:* one- —
way compatibility *see:* both- —
way graph partitioning problem *see:* k- —
way polytope *see:* k- —
way table *see:* k- —
way transportation polytope *see:* k- —
weak assumptions *see:* under —
weak compactness
[46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05]
(*see:* **Minimax theorems**)
weak convergence
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
weak convergence of probability measures
[90C11, 90C15, 90C31]
(*see:* **Stochastic integer programming: continuity, stability, rates of convergence**)
weak discrete convergence
[90C15]
(*see:* **Approximation of extremum problems with probability functionals**)
weak duality
[90C06, 90C30]
(*see:* **Duality for semidefinite programming**; **Lagrangian duality: BASICS**; **Saddle point theory and optimality conditions**)
weak duality *see:* strong and —
weak duality relation
[49K05, 49K10, 49K15, 49K20]
(*see:* **Duality in optimal control with first order differential equations**)
weak duality result
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
weak duality theorem
[49-XX, 90-XX, 93-XX]
(*see:* **Duality theory: monoduality in convex optimization**)
weak efficiency
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
weak efficiency *see:* local —
weak extremal
[41A10, 47N10, 49K15, 49K27]
(*see:* **High-order maximum principle for abnormal extremals**)
weak extremal *see:* abnormal —
weak homomorphism
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
weak minimizer
[49K27, 90C29, 90C48]
(*see:* **Set-valued optimization**)
weak order
[90-XX, 90C29]
(*see:* **Outranking methods**; **Preference modeling**)
weak prespecification
[03B52, 03E72, 47S40, 68T27, 68T35, 68Uxx, 90Bxx, 91Axx, 91B06, 92C60]
(*see:* **Boolean and fuzzy relations**)
weak principle of optimality
[90C31, 90C39]
(*see:* **Multiple objective dynamic programming**)
Weak Slater CQ
[49K27, 49K40, 90C30, 90C31]
(*see:* **First order constraint qualifications**)
weak stationarity
[58C20, 58E30, 90C46, 90C48]
(*see:* **Nonsmooth analysis: weak stationarity**)
weak stationarity *see:* Nonsmooth analysis: —
weak tangent
[49K27, 58C20, 58E30, 90C48]
(*see:* **Nonsmooth analysis: Fréchet subdifferentials**)
weakly efficient
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**; **Multi-objective optimization: pareto optimal solutions, properties**)
weakly efficient point
[90C29]
(*see:* **Generalized concavity in multi-objective optimization**)
weakly efficient point *see:* local —
weakly efficient solution
[90C29]
(*see:* **Multiple objective programming support**)
weakly efficient solution
[90C29]

(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
weakly informed
(*see:* **Beam selection in radiotherapy treatment design**)
weakly L_1 (v)-differentiable family of measures
[90C15]
(*see:* **Derivatives of probability measures**)
weakly necessary constraint
[90C05, 90C20]
(*see:* **Redundancy in nonlinear programs**)
weakly nondominated solution
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**; **Multiple objective programming support**)
weakly nondominated solution
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
weakly noninferior solution
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
weakly noninferior solution
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
weakly overtaking optimality
[49Jxx, 91Axx]
(*see:* **Infinite horizon control and dynamic games**)
weakly Pareto optimal solution
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
weakly Pareto optimal solution
[90C29]
(*see:* **Multi-objective optimization: pareto optimal solutions, properties**)
weakly polynomial time algorithm
[90C35]
(*see:* **Minimum cost flow problem**)
weakly pseudoconcave function *see:* U- —
wealth *see:* surplus —
weather
(*see:* **State of the art in modeling agricultural systems**)
Weber objective function *see:* multifacility —
Weber problem
[90B85, 90C26]
(*see:* **MINLP: application in facility location-allocation**; **Multifacility and restricted location problems**)
Weber problem
[90B85, 90C26, 90C90]
(*see:* **Global optimization in Weber's problem with attraction and repulsion**; **Multifacility and restricted location problems**)
Weber problem *see:* generalized —; multifacility —; Steiner– —
Weber problem with attraction and repulsion
[90C26, 90C90]
(*see:* **Global optimization in Weber's problem with attraction and repulsion**)
Weber's problem with attraction and repulsion *see:* Global optimization in —
Weber–Rawls objective function *see:* multifacility —
Weber–Rawls problem
[90B85]
(*see:* **Multifacility and restricted location problems**)
Weber–Rawls problem
[90B85]
(*see:* **Multifacility and restricted location problems**)
wedge *see:* universal —
wedge filters
[68W01, 90-00, 90C90, 92-08, 92C50]
(*see:* **Optimization based frameworkfor radiation therapy**)
wedge orientation optimization *see:* beam angle selection and —
weekly space-time network
(*see:* **Railroad locomotive scheduling**)
weight
[68Q20, 90-XX, 90B10, 90C26, 90C27]
(*see:* **Invexity and its applications**; **Optimal triangulations**; **Outranking methods**; **Shortest path tree algorithms**)
weight bounds *see:* lower —
weight clique *see:* maximum —
weight clique problem *see:* maximum —
weight CMST *see:* nonunit —; unit —
weight common mutated sequence *see:* minimum —
weight of a constraint
[90C10]
(*see:* **Maximum constraint satisfaction: relaxations and upper bounds**)
weight cost *see:* mean- —
weight of a customer
[90B80, 90B85]
(*see:* **Warehouse location problem**)
weight cut *see:* maximum mean- —
weight of evidence *see:* expected —
weight feedback arc set problem *see:* minimum —
weight function of a matroid
[90C09, 90C10]
(*see:* **Matroids**)
weight independent sets *see:* maximum —
weight optimization *see:* beam —
weight Steiner triangulation *see:* minimum —
weight trace *see:* maximum —
weight triangulation *see:* minimum —
weight vector
[05C60, 05C69, 05C85, 37B25, 68W01, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Heuristics for maximum clique and independent set**; **Replicator dynamics in combinatorial optimization**)
weighted
[90C09, 90C10]
(*see:* **Matroids**)
weighted assignment model *see:* the multi-resource —
weighted barycenter
[90C20]
(*see:* **Standard quadratic optimization problems: applications**)
weighted bipartite matching problem
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)

weighted characteristic vector
[05C60, 05C69, 05C85, 37B25, 68W01, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Heuristics for maximum clique and independent set**; **Replicator dynamics in combinatorial optimization**)
weighted clique number
[05C60, 05C69, 05C85, 37B25, 68W01, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Heuristics for maximum clique and independent set**; **Replicator dynamics in combinatorial optimization**)
weighted coloring
[90C35]
(*see:* **Graph coloring**)
weighted distance *see:* maximum —
weighted Euclidean norm *see:* A- —
weighted feedback vertex set problem *see:* minimum —
weighted graph
[90C35]
(*see:* **Feedback set problems**)
weighted graph bipartization problem *see:* minimum —
weighted graph coloring problem
[90C35]
(*see:* **Graph coloring**)
weighted graph planarization *see:* branch and bound algorithm for —
weighted independent set *see:* maximum —
weighted least squares
[90C30, 90C52, 90C53, 90C55]
(*see:* **Gauss–Newton method: Least squares, relation to Newton's method**)
weighted least squares problem
[65Fxx]
(*see:* **Least squares problems**)
weighted matching problem
[90C05, 90C10, 90C27, 90C35]
(*see:* **Assignment and matching**)
weighted matroid
[90C09, 90C10]
(*see:* **Matroids**)
weighted MAX-SAT problem
[03B05, 68P10, 68Q25, 68R05, 68T15, 68T20, 90C09, 90C27, 94C10]
(*see:* **Maximum satisfiability problem**)
weighted maximum norm
[90C30, 90C52, 90C53, 90C55]
(*see:* **Asynchronous distributed optimization algorithms**)
weighted planar graph *see:* maximum —
weighted problem in OR
[90B80, 90B85]
(*see:* **Warehouse location problem**)
weighted stability number
[05C69, 05C85, 68W01, 90C59]
(*see:* **Heuristics for maximum clique and independent set**)
weighted-sums program
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
weighted-sums programs with constraints
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
weighted Tchebycheff metric *see:* w- —
weighted tree association graph
[05C60, 05C69, 37B25, 90C20, 90C27, 90C35, 90C59, 91A22]
(*see:* **Replicator dynamics in combinatorial optimization**)
weighted vertex cover *see:* minimum —
weighter sup norm
[49L20, 90C40]
(*see:* **Dynamic programming: stochastic shortest path problems**)
weighter sup-norm contraction
[49L99]
(*see:* **Dynamic programming: average cost per stage problems**)
weighting space reduction
[90C29]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**)
weights
[65K05, 90C27, 90C29, 90C30, 90C57, 91C15]
(*see:* **Multi-objective optimization; Interactive methods for preference value functions**; **Optimization-based visualization**)
weights *see:* barycentric —; vertex —
Weir dual *see:* Mond– —
Weiszfeld procedure
[90B85]
(*see:* **Single facility location: multi-objective euclidean distance location**)
well *see:* type A —; type B —
well bore model
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
well-conditioned matrix
[15-XX, 65-XX, 90-XX]
(*see:* **Cholesky factorization**)
well-conditioned problem
[90C31]
(*see:* **Sensitivity and stability in NLP**)
well-defined start-ups
(*see:* **Planning in the process industry**)
well-determined system of nonlinear equations
[90C30]
(*see:* **Nonlinear least squares problems**)
well function *see:* double- —
well oil flowrate
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
well oil rate constraints *see:* upper and lower —
well-posed
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
well-posed problem
[90C05, 90C25, 90C29, 90C30, 90C31]
(*see:* **Nondifferentiable optimization: parametric programming**)
well-posed problem *see:* Levitin–Polyak —
well-posedness
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)
well-posedness
[49J40, 49M30, 65K05, 65M30, 65M32]
(*see:* **Ill-posed variational problems**)

well scheduling *see:* Mixed integer optimization in —
well-separated pair decomposition
[05C15, 05C62, 05C69, 05C85, 90C27, 90C59]
(*see:* **Optimization problems in unit-disk graphs**)
well switches *see:* maximum number of —
wells *see:* connection of —; operational status of the —; set of —; type A —; type B —
wells of type a *see:* gas lift —; naturally flowing —
wells of type b *see:* gas lift —; naturally flowing —
West corner rule *see:* North– —
Weyl fundamental theorem
[15A39, 90C05]
(*see:* **Tucker homogeneous systems of linear relations**)
what-if-when scenarios
[90C06, 90C10, 90C11, 90C30, 90C57, 90C90]
(*see:* **Modeling difficult optimization problems**)
wheel procedure *see:* roulette —
when scenarios *see:* what-if- —
Whitney savings heuristic
[68T99, 90C27]
(*see:* **Capacitated minimum spanning trees**)
Whitney statistic *see:* Mann– —
wide process networks under uncertainty *see:* Bilevel programming framework for enterprise- —
width *see:* excess —
Wiener–Hopf equations
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
Wiener–Hopf equations
[49J40, 62H30, 65C30, 65C40, 65C50, 65C60, 65Cxx, 90C05]
(*see:* **Variational principles**)
Wiener measure
[65K05, 68Q05, 68Q10, 68Q25, 90C05, 90C25, 90C26]
(*see:* **Information-based complexity and information-based optimization**)
Wiener model
[62C10, 65K05, 90C10, 90C15, 90C26]
(*see:* **Bayesian global optimization**)
Wiener probability measure
[60J65, 68Q25]
(*see:* **Adaptive global search**)
Wiener process
[60J65, 68Q25]
(*see:* **Adaptive global search**)
Wiener process
[60J65, 68Q25]
(*see:* **Adaptive global search**)
wilhelm *see:* Leibniz, gottfried —
Wilhelm Leibniz *see:* Gottfried —
Williams algorithm *see:* Esau– —
Wilson equation
[90C26, 90C90]
(*see:* **Global optimization in phase and chemical reaction equilibrium**)
Wilson equation *see:* regular solution of the —
window constraints *see:* time —
windows *see:* time —; vehicle routing problem with time —
wiring problem *see:* backboard —
with mathematical rigor
[65G20, 65G30, 65G40, 65H20]
(*see:* **Interval analysis: unconstrained and constrained optimization**)
(with respect to another) *see:* pseudomonotone bifunction —
without decomposition *see:* heat exchanger network synthesis —
without pivoting *see:* guaranteed to be stable —
without using (sub)gradients parametric representations *see:* necessary optimality condition —
Wolfe *see:* Frank– —
Wolfe algorithm *see:* Frank– —; regularized Frank– —
Wolfe decomposition *see:* Dantzig– —; nonlinear Dantzig– —; regularized Frank– —
Wolfe dual
[90C26]
(*see:* **Invexity and its applications**)
Wolfe dual
[90C26]
(*see:* **Invexity and its applications**)
Wolfe reduced gradient method
[65K05, 65K10]
(*see:* **ABS algorithms for optimization**)
Wolfe test
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
Wolfowitz method *see:* Keifer– —
Woodbury formula *see:* Sherman–Morrison– —
wOR
[76T30, 90C11, 90C90]
(*see:* **Mixed integer optimization in well scheduling**)
word patterns
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
word patterns
[90C09, 90C10]
(*see:* **Optimization in classifying text documents**)
words *see:* meaningful —
work
[03D15, 68Q05, 68Q15]
(*see:* **Parallel computing: complexity classes**)
work *see:* principle of virtual —
work first algorithm *see:* mandatory —
working basis
[49M25, 90-08, 90C05, 90C06, 90C08, 90C15]
(*see:* **Simple recourse problem: primal method**)
working basis
[49M25, 90-08, 90C05, 90C06, 90C08, 90C15]
(*see:* **Simple recourse problem: primal method**)
workloadBalanced
[65K05, 65Y05, 65Y10, 65Y20, 68W10]
(*see:* **Interval analysis: parallel methods for global optimization**)
world *see:* model —; real —
world problem *see:* real- —
Worsley bounds *see:* Hunter– —
Worsley upper bound *see:* Hunter– —
worst-case analysis
[60J65, 62C20, 68Q25, 90C15]
(*see:* **Adaptive global search**; **Stochastic programming: minimax approach**)

worst-case analysis
[62C20, 90C15]
(*see:* **Stochastic programming: minimax approach**)
worst-case approach
[90C15, 90C26, 90C33]
(*see:* **Stochastic bilevel programs**)
worst-case complexity
[90C35]
(*see:* **Minimum cost flow problem**)
worst-case optimality
[65D25, 68W30]
(*see:* **Complexity of gradients, Jacobians, and Hessians**)
worst-case performance guarantee
[05C85]
(*see:* **Directed tree networks**)
WPE
[91B50]
(*see:* **Walrasian price equilibrium**)
wrapping effect
[65G20, 65G30, 65G40, 65L99]
(*see:* **Interval analysis: differential equations**)

X

x *see:* decision variables —
X-ray crystallography: Shake and bake approach *see:* Phase problem in —
X-ray diffraction data *see:* Optimization techniques for phase retrieval based on single-crystal —
x variables *see:* full space of —

Y

Yadegar linearization *see:* Frieze– —
Ye potential function *see:* Tanabe–Todd– —
yield *see:* Ω-based —
yield curve *see:* interest rate —
yield to maturity
[90C34, 91B28]
(*see:* **Semi-infinite programming and applications in finance**)
yOP
(*see:* **Integrated planning and scheduling**)
York Times *see:* the New —
Yosida-Hewitt decomposition
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
Yosida–Hewitt theorem
[90C15]
(*see:* **Stochastic programming: nonanticipativity and lagrange multipliers**)
Young inequality
[90C05, 90C25]
(*see:* **Young programming**)
Young inequality *see:* Fenchel– —
Young programming
(90C25, 90C05)
(*refers to:* **Linear programming**)
Young programming
[90C05, 90C25]
(*see:* **Young programming**)
younger brother waits
[49J35, 49K35, 62C20, 91A05, 91A40]
(*see:* **Minimax game tree searching**)
Yuan algorithm *see:* Dai– —
Yule–Walker method *see:* overdetermined —

Z

z-critical cone
[90C30, 90C33]
(*see:* **Optimization with equilibrium constraints: A piecewise SQP approach**)
Zamolodchikov differential equation *see:* Knizhnik– —
Zangwill algorithm
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
Zangwill theorem
[90C30]
(*see:* **Rosen's method, global convergence, and Powell's conjecture**)
zemel measure
[90B06, 90B35, 90C06, 90C10, 90C27, 90C39, 90C57, 90C59, 90C60, 90C90]
(*see:* **Traveling salesman problem**)
Zeolite Association *see:* atlas of the International —
zeolite separation and catalysis: optimization methods *see:* Shape selective —
zero *see:* homogeneous of degree —; transversal to —
zero-epi mapping
[90C33]
(*see:* **Topological methods in complementarity theory**)
zero-epi mapping
[90C33]
(*see:* **Topological methods in complementarity theory**)
zero-error capacity *see:* shannon —
zero-Inventory Ordering
[55M05, 90B05, 90B10, 90C11, 90C39, 90C90]
(*see:* **Economic lot-sizing problem**)
zero-nonzero pattern
[90C09, 90C10]
(*see:* **Combinatorial matrix analysis**)
zero-one integer feasibility problem
[90C25, 90C33]
(*see:* **Integer linear complementary problem**)
zero-one integer problem *see:* linear —
zero-one integer program
[90C90, 91A65, 91B99]
(*see:* **Bilevel programming: applications**)
zero-one integer programming
[90C25, 90C33]
(*see:* **Integer linear complementary problem**)
zero-one knapsack problem
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
zero-one knapsack problem *see:* multidimensional —

zero-one optimization
[49M37, 90C26, 91A10]
(*see:* **Bilevel programming**)
zero-one problem *see:* quadratic —
zero-one programming *see:* Fractional —; pure —
zero-one programming problem
[90C10, 90C11, 90C27, 90C57]
(*see:* **Integer programming**)
zero pattern *see:* positive-negative- —
zero residual problem
[90C30]
(*see:* **Nonlinear least squares problems**)
zero-sum game *see:* two-person —
zero-sum perfect-information game *see:* two-player —
zero-wait
[90C26]
(*see:* **MINLP: design and scheduling of batch processes**)
zeros
[12D10, 12Y05, 13P10]
(*see:* **Gröbner bases for polynomial equations**)
Zhegalkin algebra
[03B50, 68T15, 68T30]
(*see:* **Finite complete systems of many-valued logic algebras**)
Ziont criss-cross method
[05B35, 65K05, 90C05, 90C20, 90C33]
(*see:* **Criss-cross pivoting rules**)
Zionts–Wallenius procedure
[90C11, 90C29]
(*see:* **Multi-objective mixed integer programming**)
zone consistency
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
zone of a function
[68W10, 90B15, 90C06, 90C30]
(*see:* **Stochastic network problems: massively parallel solution**)
zonotope
[05A, 15A, 51M, 52A, 52B, 52C, 62H, 68Q, 68R, 68U, 68W, 90B, 90C]
(*see:* **Convex discrete optimization**)
Zuselevich *see:* Shor, Naum —

Name Index

A

Aarts, E. *see:* **Heuristics for maximum clique and independent set**
Aasen, J. *see:* **Symmetric systems of linear equations**
Abaffy, J. *see:* **ABS algorithms for linear equations and linear least squares**
Abel, N.H. *see:* **Hilbert's thirteenth problem**; **Lagrange, Joseph-Louis**
Aberth, O. *see:* **Interval analysis: differential equations**
Aboudi, R. *see:* **Multidimensional knapsack problems**
Abounadi, J. *see:* **Resource allocation for epidemic control**
Ackerman, E. *see:* **Resource allocation for epidemic control**
Adams, W.P. *see:* **Disjunctive programming**; **Quadratic assignment problem**; **Time-dependent traveling salesman problem**
Adler, I. *see:* **Complexity theory: quadratic programming**; **Linear programming: interior point methods**; **Probabilistic analysis of simplex algorithms**
Afentakis, P. *see:* **Inventory management in supply chains**
Ahmed, S. *see:* **Chemical process planning**
Ahuja, R.K. *see:* **Greedy randomized adaptive search procedures**; **Maximum flow problem**
Aiyoshi, E. *see:* **Bilevel linear programming**; **Bilevel programming in management**
Akgul, M. *see:* **Complexity of degeneracy**
Akl, S.G. *see:* **Minimax game tree searching**
Aksoy, Y. *see:* **Multi-objective mixed integer programming**
Al-Baali, M. *see:* **Nonlinear least squares: Newton-type methods**
Al-Khayyal, F.A. *see:* **Global optimization in Weber's problem with attraction and repulsion**
Ali, A. *see:* **Multicommodity flow problems**
Almogy, Y. *see:* **Fractional programming**
Almquist, K. *see:* **Minimax game tree searching**
Althöfer, I. *see:* **Minimax game tree searching**
Altinkemer, K. *see:* **Capacitated minimum spanning trees**
Altiok, T. *see:* **Operations research models for supply chain management and design**
Alves, M.J. *see:* **Multi-objective mixed integer programming**
Alvim, A.C.F. *see:* **Greedy randomized adaptive search procedures**
Aly, S. *see:* **Design optimization in computational fluid dynamics**
Amberg, A. *see:* **Capacitated minimum spanning trees**
Anandalingam, G. *see:* **Bilevel linear programming**; **Bilevel programming in management**
Anderson, J.M. *see:* **Alignment problem**
Anderson, J.R. *see:* **Neural networks for combinatorial optimization**
Anderson, R.M. *see:* **Resource allocation for epidemic control**
Ando, K. *see:* **Combinatorial optimization algorithms in resource allocation problems**
Andricioaei, I. *see:* **Simulated annealing methods in protein folding**
Anstreicher, K.M. *see:* **Fractional programming**
Apprey, V. *see:* **Bilevel programming in management**
Arbel, A. *see:* **Multiple objective programming support**
Argand, J.R. *see:* **Gauss, Carl Friedrich**
Aristotle *see:* **Operations research**
Armijo, L. *see:* **Local attractors for gradient-related descent iterations**
Armstrong, R.D. *see:* **Overdetermined systems of linear equations**
Arnol'd, V. *see:* **Hilbert's thirteenth problem**
Arntzen, B.C. *see:* **Operations research models for supply chain management and design**
Arora, S. *see:* **Quadratic assignment problem**; **Steiner tree problems**
Arrow, K. *see:* **Financial equilibrium**; **Generalized concavity in multi-objective optimization**; **Inventory management in supply chains**; **Multilevel optimization in mechanics**
Asaithambi, N.S. *see:* **Interval global optimization**
Asano, T. *see:* **Maximum satisfiability problem**
Ashford, R.W. *see:* **Financial applications of multicriteria analysis**
Asmuth, R. *see:* **Spatial price equilibrium**
Assad, A.A. *see:* **Multicommodity flow problems**
Athans, M. *see:* **Robust control**
Auchmuty, G. *see:* **Quasidifferentiable optimization: variational formulations**
Avakov, E.R. *see:* **High-order maximum principle for abnormal extremals**
Averbakh, I. *see:* **Multidimensional knapsack problems**
Avis, D. *see:* **Complexity of degeneracy**; **Least-index anticycling rules**; **Linear programming: Klee–Minty examples**
Avriel, M. *see:* **MINLP: application in facility location-allocation**
Axsäter, S. *see:* **Inventory management in supply chains**
Ayer, M. *see:* **Isotonic regression problems**
Ayguagé, E. *see:* **Alignment problem**

B

Bäck, T. *see:* **Heuristics for maximum clique and independent set**
Bafna, V. *see:* **Feedback set problems**
Bagajewicz, M.J. *see:* **MINLP: mass and heat exchanger networks**
Baganha, M.P. *see:* **Operations research models for supply chain management and design**
Bajgier, S.M. *see:* **Multicriteria sorting methods**
Balabanov, V. *see:* **Multidisciplinary design optimization**
Balakrishnan, A. *see:* **Operations research models for supply chain management and design**
Balakrishnan, V. *see:* **Robust control**
Balas, E. *see:* **Disjunctive programming**; **Evolutionary algorithms in combinatorial optimization**; **Global optimization in Weber's problem with attraction and repulsion**; **Integer programming**; **Integer programming: branch and cut algorithms**
Ballard, B.W. *see:* **Minimax game tree searching**
Ballone, P. *see:* **Global optimization in Lennard–Jones and morse clusters**
Banach, S. *see:* **Theorems of the alternative and optimization**
Bandler, W. *see:* **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**
Bar-Yeruda, R. *see:* **Feedback set problems**
Barbosa-Póvoa, A.P.F.D. *see:* **MINLP: design and scheduling of batch processes**
Bard, J.F. *see:* **Bilevel programming in management**
Bard, Y. *see:* **Principal pivoting methods for linear complementarity problems**
Barmish, B.R. *see:* **Robust control**
Barnes, E. *see:* **Linear programming: interior point methods**
Barnett, D. *see:* **Multicommodity flow problems**
Barnhart, C. *see:* **Multicommodity flow problems**
Barrodale, I. *see:* **Overdetermined systems of linear equations**
Barros, A.I. *see:* **Fractional programming**
Barrow, H.G. *see:* **Replicator dynamics in combinatorial optimization**
Bartels, R. *see:* **Gauss, Carl Friedrich**
Bartholomew-Biggs, M.C. *see:* **Nonlinear least squares: Newton-type methods**
Barton, L.G. *see:* **Piecewise linear network flow problems**
Bassalygo, L. *see:* **Hilbert's thirteenth problem**
Battiti, R. *see:* **Heuristics for maximum clique and independent set**; **Multidimensional knapsack problems**
Baudet, G. *see:* **Minimax game tree searching**
Bauer, F.L. *see:* **Complexity of gradients, Jacobians, and Hessians**
Baur, W. *see:* **Automatic differentiation: introduction, history and rounding error estimation**
Beale, E.M.L. *see:* **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Stochastic linear programming: decomposition and cutting planes**
Beall, C.L. *see:* **Frequency assignment problem**
Beasley, J.E. *see:* **Evolutionary algorithms in combinatorial optimization**; **Multidimensional knapsack problems**; **Set covering, packing and partitioning problems**
Becker, A. *see:* **Feedback set problems**
Beckmann, M.J. *see:* **Dynamic traffic networks**; **Equilibrium networks**; **Quadratic assignment problem**; **Traffic network equilibrium**
Bélisle, C.J.P. *see:* **Random search methods**
Bellman, R. *see:* **Dynamic programming: optimal control applications**; **Image space approach to optimization**; **Multiple objective dynamic programming**; **Shortest path tree algorithms**
Ben-Tal, A. *see:* **Second order constraint qualifications**
Benders, J.F. *see:* **Stochastic linear programming: decomposition and cutting planes**
Benichou, M. *see:* **Integer programming: branch and bound methods**
Benoit, C. *see:* **Least squares problems**
Benton, W.C. *see:* **Operations research models for supply chain management and design**
Berger, A.J. *see:* **Stochastic programming: parallel factorization of structured matrices**
Bergeron, M. *see:* **Portfolio selection and multicriteria analysis**
Berghammer, R. *see:* **Boolean and fuzzy relations**
Berman, O. *see:* **Facility location with externalities**
Berman, P. *see:* **Feedback set problems**; **Steiner tree problems**
Bern, M.W. *see:* **Steiner tree problems**
Bernoulli, Jakob *see:* **Variational principles**
Bernoulli, Johann *see:* **Variational principles**
Bertocchi, M. *see:* **ABS algorithms for linear equations and linear least squares**
Bertsekas, D. *see:* **Local attractors for gradient-related descent iterations**; **Shortest path tree algorithms**
Bertsimas, D.J. *see:* **Survivable networks**
Besanko, I.E. *see:* **Interval global optimization**
Best, M.J. *see:* **Isotonic regression problems**; **Standard quadratic optimization problems: applications**
Bett, J.T. *see:* **Nonlinear least squares: Newton-type methods**
Beuthe, M. *see:* **Preference disaggregation**
Bhaskar, K. *see:* **Financial applications of multicriteria analysis**
Bialas, W.F. *see:* **Bilevel linear programming**; **Bilevel programming in management**
Bickel, T.C. *see:* **Chemical process planning**
Biegler, L.T. *see:* **MINLP: mass and heat exchanger networks**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**
Biggs, M.C. *see:* **Successive quadratic programming**
Billington, C. *see:* **Operations research models for supply chain management and design**
Binkley, J. *see:* **Multicommodity flow problems**
Birge, J.R. *see:* **History of optimization**; **Stochastic programming: parallel factorization of structured matrices**; **Stochastic programs with recourse: upper bounds**
Bischof, C. *see:* **Automatic differentiation: parallel computation**
Bixby, R. *see:* **Alignment problem**
Black, F. *see:* **Financial equilibrium**
Blackwell, D. *see:* **Dynamic programming: average cost per stage problems**; **Dynamic programming: discounted problems**; **Dynamic programming: undiscounted problems**
Bland, R.G. *see:* **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Oriented matroids**

Blau, G.J. *see:* **Global optimization in generalized geometric programming**
Blom, R. *see:* **Fractional programming**
Blum, E. *see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**
Board, J.L.G. *see:* **Portfolio selection: markowitz mean-variance model**
Bodon, E. *see:* **ABS algorithms for linear equations and linear least squares**
Boffey, T.B. *see:* **Communication network assignment problem**
Boggs, P.T. *see:* **Generalized total least squares**; **Nonlinear least squares problems**
Bogle, I.D.L. *see:* **Interval analysis: application to chemical engineering design problems**
Böhm, M. *see:* **Minimax game tree searching**
Bojkov, B. *see:* **Dynamic programming: optimal control applications**
Bomze, I. *see:* **Replicator dynamics in combinatorial optimization**
Boncompte, M. *see:* **Fractional programming**
Bonferroni, C.E. *see:* **Approximation of multivariate probability integrals**
Bookbinder, J.H. *see:* **Operations research models for supply chain management and design**
Boorstyn, R.R. *see:* **Capacitated minimum spanning trees**
Booth, K. *see:* **Graph planarization**
Borchers, A. *see:* **Steiner tree problems**
Borchers, B. *see:* **MINLP: branch and bound methods**
Border, K.C. *see:* **Walrasian price equilibrium**
Borgwardt, K.H. *see:* **Probabilistic analysis of simplex algorithms**
Born, M. *see:* **Carathéodory, Constantine**
Bornstein, C.T. *see:* **Piecewise linear network flow problems**
Borodin, A. *see:* **Parallel computing: complexity classes**
Borůvka, O. *see:* **Boolean and fuzzy relations**; **Network design problems**
Bouyssou, D. *see:* **Preference disaggregation**
Bowman, V.J. *see:* **Multi-objective integer linear programming**; **Multi-objective mixed integer programming**
Box, G.E. *see:* **Forecasting**
Boyd, S.P. *see:* **Robust control**
Boyle, P.P. *see:* **Operations research and financial markets**
Bracken, J. *see:* **Bilevel programming: introduction, history and overview**
Braid, R.M. *see:* **Facility location with externalities**
Bramel, J. *see:* **Set covering, packing and partitioning problems**
Bramley, R. *see:* **Relaxation in projection methods**
Brandeau, M.L. *see:* **Facility location with externalities**; **Resource allocation for epidemic control**
Brandstädt, A. *see:* **Feedback set problems**
Brealey, R.A. *see:* **Portfolio selection: markowitz mean-variance model**
Breeden, D.T. *see:* **Financial equilibrium**
Bregman, L.M. *see:* **Young programming**
Brelaz, D. *see:* **Frequency assignment problem**
Brennecke, J.F. *see:* **Global optimization: application to phase equilibrium problems**
Bresina, J.L. *see:* **Greedy randomized adaptive search procedures**
Brezis, H. *see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**
Brock, W.A. *see:* **Infinite horizon control and dynamic games**
Broeckx, F. *see:* **Quadratic assignment problem**
Brooks, C.L. *see:* **Simulated annealing methods in protein folding**
Brosilow, C.B. *see:* **Multilevel optimization in mechanics**
Browder, F. *see:* **Cost approximation algorithms**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**
Brown, K.M. *see:* **Nonlinear least squares: Newton-type methods**
Brown, R.G. *see:* **Forecasting**
Broyden, C.G. *see:* **ABS algorithms for linear equations and linear least squares**; **Broyden family of methods and the BFGS update**; **Farkas lemma**; **Nonlinear least squares: Newton-type methods**
Brucker, P. *see:* **Quadratic knapsack**
Brunk, H.D. *see:* **Isotonic regression problems**
Buehler, R.J. *see:* **Conjugate-gradient methods**
Bui, T.N. *see:* **Heuristics for maximum clique and independent set**
Bulger, D.W. *see:* **Random search methods**
Bulteau, J.P. *see:* **Nonlinear least squares: trust region methods**
Bunch, J.R. *see:* **Least squares problems**
Burch, S.F. *see:* **Maximum entropy principle: image reconstruction**
Burgee, S. *see:* **Multidisciplinary design optimization**
Burkard, R.E. *see:* **Biquadratic assignment problem**; **Communication network assignment problem**; **Quadratic assignment problem**
Burke, J.V. *see:* **Composite nonsmooth optimization**
Burks, A. *see:* **Von Neumann, John**
Burns, L.B. *see:* **Operations research models for supply chain management and design**
Burstall, R.M. *see:* **Replicator dynamics in combinatorial optimization**
Büttner, L. *see:* **Gauss, Carl Friedrich**
Bykadorov, I.A. *see:* **Fractional programming**
Byrd, R.H. *see:* **Generalized total least squares**; **Nonlinear least squares: trust region methods**; **Rosen's method, global convergence, and Powell's conjecture**; **Semidefinite programming and determinant maximization**; **Successive quadratic programming: applications in the process industry**
Byrne, R.P. *see:* **Interval analysis: application to chemical engineering design problems**

C

Cachon, G.P. *see:* **Operations research models for supply chain management and design**
Cai, J. *see:* **Graph planarization**
Cai, M. *see:* **Feedback set problems**
Calamai, P.H. *see:* **Global optimization in Weber's problem with attraction and repulsion**
Caley, A. *see:* **Symmetric systems of linear equations**
Calvin, J.M. *see:* **Information-based complexity and information-based optimization**
Camm, J.D. *see:* **Operations research models for supply chain management and design**
Campbell, M.S. *see:* **Minimax game tree searching**

Candler, W. *see:* **Bilevel linear programming**; **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**
Cantor, D.G. *see:* **Simplicial decomposition**
Cantor, G. *see:* **Vector optimization**
Carathéodory, C. *see:* **Image space approach to optimization**
Carle, A. *see:* **Automatic differentiation: parallel computation**
Carlier, J. *see:* **Job-shop scheduling problem**
Carpenter, T.J. *see:* **Financial optimization**
Cartan, E. *see:* **Variational principles**
Carter, B. *see:* **Heuristics for maximum clique and independent set**
Carvalho, L.A.V. *see:* **Feasible sequential quadratic programming**
Cauchy, A.-L. *see:* **Conjugate-gradient methods**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**
Cayley, A. *see:* **Matrix completion problems**
Cederbaum, I. *see:* **Graph planarization**
Çela, E. *see:* **Biquadratic assignment problem**; **Communication network assignment problem**
Černý, V. *see:* **Quadratic assignment problem**
Chaitin, G. *see:* **Kolmogorov complexity**
Chakravarti, N. *see:* **Isotonic regression problems**
Chakravarty, S. *see:* **Infinite horizon control and dynamic games**
Chamberlain, G. *see:* **Financial equilibrium**
Chan, L.M.A. *see:* **Operations research models for supply chain management and design**
Chandra, P. *see:* **Operations research models for supply chain management and design**
Chandrasekaran, R. *see:* **Complexity of degeneracy**; **Simplicial pivoting algorithms for integer programming**
Chandru, V. *see:* **Simplicial pivoting algorithms for integer programming**
Chang, K.-C. *see:* **Hemivariational inequalities: eigenvalue problems**; **Nonconvex-nonsmooth calculus of variations**
Chang, M.S. *see:* **Feedback set problems**
Chang, Y.Y. *see:* **Criss-cross pivoting rules**
Chapman, B.M. *see:* **Alignment problem**
Charnes, A. *see:* **Decision support systems with multiple criteria**; **Extremum problems with probability functions: kernel type solution methods**; **Fractional programming**; **Lexicographic pivoting rules**; **Multicriteria sorting methods**; **Preference disaggregation**; **Probabilistic constrained linear programming: duality theory**; **Semi-infinite programming, semidefinite programming and perfect duality**
Chebotarev, N. *see:* **Hilbert's thirteenth problem**
Chebyshev, P.L. *see:* **Nondifferentiable optimization**; **Nondifferentiable optimization: minimax problems**
Chen, D. *see:* **Steiner tree problems**
Chen, H.S. *see:* **Successive quadratic programming: decomposition methods**
Chen, J.J.J. *see:* **Global optimization of heat exchanger networks**
Chen, L.H. *see:* **Numerical methods for unary optimization**
Chen, M. *see:* **Alignment problem**
Chen, M.L. *see:* **Network location: covering problems**
Chen, P.-C. *see:* **Global optimization in Weber's problem with attraction and repulsion**
Chen, T.-S. *see:* **Alignment problem**
Chen, X. *see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**
Chenery, H.B. *see:* **Linear ordering problem**
Cheney, W. *see:* **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**
Cheriyan, J. *see:* **Maximum flow problem**
Chesser, L. *see:* **Portfolio selection and multicriteria analysis**
Chiba, T. *see:* **Graph planarization**
Chiu, S.S. *see:* **Facility location with externalities**
Cho, Y.B. *see:* **Graph planarization**
Cholesky, A.L. *see:* **Cholesky factorization**; **Least squares problems**
Choo, E.-U. *see:* **Multicriteria sorting methods**; **Multi-objective mixed integer programming**
Chopra, V.R. *see:* **Portfolio selection: markowitz mean-variance model**
Chou, W. *see:* **Capacitated minimum spanning trees**
Chow, S.N. *see:* **Globally convergent homotopy methods**
Chrystal, G. *see:* **Single facility location: circle covering problem**
Chu, A.T.W. *see:* **Estimating data for multicriteria decision making problems: optimization techniques**
Chu, P.C. *see:* **Evolutionary algorithms in combinatorial optimization**; **Multidimensional knapsack problems**
Chudak, F.A. *see:* **Feedback set problems**
Chung, F.R.K. *see:* **Steiner tree problems**
Chung, S.-J. *see:* **Complexity of degeneracy**
Church, A. *see:* **Modeling languages in optimization: a new paradigm**
Chvátal, V. *see:* **Integer programming**; **Integer programming: cutting plane algorithms**; **Least-index anticycling rules**
Ciric, A.R. *see:* **MINLP: heat exchanger network synthesis**; **MINLP: reactive distillation column synthesis**
Clark, A. *see:* **Inventory management in supply chains**
Clark, A.J. *see:* **Operations research models for supply chain management and design**
Clarke, F.H. *see:* **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Multilevel optimization in mechanics**; **Nonconvex energy functions: hemivariational inequalities**; **Nondifferentiable optimization**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**
Clarke, M.R.B. *see:* **Multidimensional knapsack problems**
Clausius, R. *see:* **Entropy optimization: shannon measure of entropy and its properties**
Cleary, J.G. *see:* **Interval constraints**
Cleveland, A.D. *see:* **Optimization in classifying text documents**
Cleveland, D. *see:* **Optimization in classifying text documents**
Clímaco, J. *see:* **Multi-objective mixed integer programming**
Cobham, A. *see:* **Complexity classes in optimization**
Cohen, J. *see:* **Volume computation for polytopes: strategies and performances**

Cohen, K.J. *see:* **Portfolio selection: markowitz mean-variance model**
Cohen, M.A. *see:* **Operations research models for supply chain management and design**
Cohon, J.L. *see:* **Multi-objective mixed integer programming**
Coleman, T.F. *see:* **Automatic differentiation: calculation of Newton steps**; **Discontinuous optimization**
Colson, G. *see:* **Portfolio selection and multicriteria analysis**
Conn, A. *see:* **Successive quadratic programming: applications in the process industry**
Conn, A.R. *see:* **Discontinuous optimization**; **Global optimization in Weber's problem with attraction and repulsion**
Connor, G. *see:* **Financial equilibrium**
Conti, P. *see:* **Integer programming: algebraic methods**
Cook, S.A. *see:* **Complexity classes in optimization**; **Complexity theory**; **Computational complexity theory**; **Finite complete systems of many-valued logic algebras**; **NP-complete problems and proof methodology**; **Parallel computing: complexity classes**
Cooke, K.L. *see:* **Multiple objective dynamic programming**
Cooper, W.W. *see:* **Decision support systems with multiple criteria**; **Extremum problems with probability functions: kernel type solution methods**; **Fractional programming**; **Multicriteria sorting methods**; **Preference disaggregation**; **Probabilistic constrained linear programming: duality theory**; **Semi-infinite programming, semidefinite programming and perfect duality**
Coorg, S.R. *see:* **Feedback set problems**
Corley, H.W. *see:* **Multiple objective dynamic programming**
Cottle, R.W. *see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Simplicial pivoting algorithms for integer programming**
Courant, R. *see:* **History of optimization**; **Interval global optimization**; **Steiner tree problems**
Cournot, A. *see:* **Farkas lemma**; **Oligopolistic market equilibrium**; **Second order optimality conditions for nonlinear optimization**; **Spatial price equilibrium**
Cover, T.M. *see:* **Competitive ratio for portfolio management**
Cox, J.C. *see:* **Financial equilibrium**
Cozzens, M.B. *see:* **Frequency assignment problem**
Crainic, T. *see:* **Multicommodity flow problems**
Crama, Y. *see:* **Multidimensional knapsack problems**
Craven, B.D. *see:* **Fractional programming**
Crooks, P. *see:* **Alignment problem**
Cross, H. *see:* **Multilevel optimization in mechanics**
Crouzeix, J.P. *see:* **Fractional programming**
Crowder, H. *see:* **Integer programming: branch and bound methods**
Crowder, H.P. *see:* **Integer programming: branch and cut algorithms**
Csiszar, I. *see:* **Young programming**
Cung, V.-D. *see:* **Minimax game tree searching**
Curtis, A.R. *see:* **Automatic differentiation: calculation of Newton steps**
Cyganski, D. *see:* **Quadratic assignment problem**
Czyzak, P. *see:* **Multi-objective combinatorial optimization**

D

Dadebo, S.A. *see:* **Dynamic programming: optimal control applications**
Daellenbach, H.G. *see:* **Multiple objective dynamic programming**
Dafermos, S.C. *see:* **Dynamic traffic networks**; **Equilibrium networks**; **Oligopolistic market equilibrium**; **Spatial price equilibrium**; **Traffic network equilibrium**; **Variational inequalities**; **Variational inequalities: projected dynamical system**; **Walrasian price equilibrium**
Dai, Z. *see:* **Steiner tree problems**
d'Alembert, J. *see:* **Leibniz, gottfried wilhelm**
Dammeyer, F. *see:* **Multidimensional knapsack problems**
Dantzig, G.B. *see:* **Complexity of degeneracy**; **Decomposition principle of linear programming**; **History of optimization**; **Integer programming: cutting plane algorithms**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Linear programming**; **Monte-Carlo simulations for stochastic optimization**; **Multicommodity flow problems**; **Probabilistic analysis of simplex algorithms**; **Semi-infinite programming, semidefinite programming and perfect duality**; **Simple recourse problem: primal method**; **Stochastic linear programming: decomposition and cutting planes**; **Tucker homogeneous systems of linear relations**
Darte, A. *see:* **Alignment problem**
Daskin, M.S. *see:* **Operations research models for supply chain management and design**; **Time-dependent traveling salesman problem**
Davidon, W.C. *see:* **Broyden family of methods and the BFGS update**; **Conjugate-gradient methods**; **Rosen's method, global convergence, and Powell's conjecture**
Davis, L. *see:* **Evolutionary algorithms in combinatorial optimization**
Davis, M. *see:* **Maximum satisfiability problem**
de Bruin, A. *see:* **Minimax game tree searching**
De Bruyn, C. *see:* **Portfolio selection and multicriteria analysis**
de Fermat, P. *see:* **History of optimization**
de la Vallée Poussin, Ch. *see:* **History of optimization**
de Leeuw, J. *see:* **Preference disaggregation**
Deák, I. *see:* **Approximation of multivariate probability integrals**
Debreu, G. *see:* **Financial equilibrium**
DeCani, J.S. *see:* **Linear ordering problem**
Dedekind, R. *see:* **Inference of monotone boolean functions**
DeKluyver, C.A. *see:* **Multiple objective dynamic programming**
del Ferro, S. *see:* **Hilbert's thirteenth problem**
Del Vecchio A. *see:* **Operations research models for supply chain management and design**
Delchambre, A. *see:* **Evolutionary algorithms in combinatorial optimization**
Demming, W.E. *see:* **Generalized total least squares**
DeMorgan, A. *see:* **Boolean and fuzzy relations**
Demyanov, V.F. *see:* **Convex max-functions**; **Nonconvex energy functions: hemivariational inequalities**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: calculus of quasidifferentials**; **Quasidifferentiable optimization:**

codifferentiable functions; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: exact penalty methods**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**
Deng, N.Y. *see:* **Numerical methods for unary optimization**
Deng, X. *see:* **Competitive ratio for portfolio management**; **Feedback set problems**
Dennis, J.E. *see:* **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**; **Optimal design in nonlinear optics**
Deshpande, A.S. *see:* **Optimization in boolean classification problems**
Despotis, D.K. *see:* **Portfolio selection and multicriteria analysis**; **Preference disaggregation**
Desrosiers, J. *see:* **Multicommodity flow problems**
DeTremblay, M. *see:* **Dynamic programming: optimal control applications**
Deuermeyer, B.L. *see:* **Inventory management in supply chains**
Devaud, J.M. *see:* **Preference disaggregation approach: basic features, examples from financial decision making**
Devine, M.D. *see:* **MINLP: application in facility location-allocation**
Di Battista, G. *see:* **Graph planarization**
Di Pillo, G. *see:* **Nonlinear least squares problems**
Dial, R.B. *see:* **Shortest path tree algorithms**
Diderich, C.G. *see:* **Alignment problem**; **Minimax game tree searching**
Dijkstra, E.W. *see:* **Network design problems**; **Shortest path tree algorithms**
Dikin, I.I. *see:* **Linear programming: interior point methods**
Dincer, M.C. *see:* **Operations research models for supply chain management and design**
Ding, B. *see:* **Standard quadratic optimization problems: applications**
Dinic, E.A. *see:* **Maximum flow problem**
Dinkelbach, W. *see:* **Fractional programming**; **Quadratic fractional programming: Dinkelbach method**
Dion, M. *see:* **Alignment problem**
direct search optimization *see:* **Dynamic programming: optimal control applications**
Dixon, L.C.W. *see:* **Automatic differentiation: calculation of Newton steps**; **Nonlinear least squares: Newton-type methods**; **Rosen's method, global convergence, and Powell's conjecture**
Doherty, M.F. *see:* **Global optimization: application to phase equilibrium problems**
Doig, A.G. *see:* **Integer programming**; **Integer programming: branch and bound methods**
Doignon, J.P. *see:* **Boolean and fuzzy relations**
Dominiak, C. *see:* **Portfolio selection and multicriteria analysis**
Dong, J. *see:* **Financial equilibrium**
Dorigo, M. *see:* **Evolutionary algorithms in combinatorial optimization**
Dosios, K. *see:* **Linear programming: Klee–Minty examples**
Doyle, J.C. *see:* **Robust control**
Drezner, Z. *see:* **Global optimization in Weber's problem with attraction and repulsion**; **Operations research models for supply chain management and design**
Du, D.-Z. *see:* **Rosen's method, global convergence, and Powell's conjecture**; **Steiner tree problems**
Dubovitskii, A. Ya. *see:* **High-order maximum principle for abnormal extremals**
Dubovitskii, A.Ya. *see:* **Stochastic programming: nonanticipativity and lagrange multipliers**
Dudàs T. *see:* **Communication network assignment problem**
Duff, J.S. *see:* **Automatic differentiation: calculation of Newton steps**
Duffin, R.J. *see:* **Theorems of the alternative and optimization**
Duke of Brunswick–Wolfenbüttel *see:* **Gauss, Carl Friedrich**
Dunn, R.F. *see:* **MINLP: mass and heat exchanger networks**
Dupuis, P. *see:* **Dynamic traffic networks**; **Variational inequalities: projected dynamical system**
Duran, M.A. *see:* **Generalized outer approximation**; **MINLP: outer approximation algorithm**
Durso, A. *see:* **Multi-objective mixed integer programming**
Dyer, M.E. *see:* **Quadratic assignment problem**
Dynkin, E.B. *see:* **Stochastic programming: nonanticipativity and lagrange multipliers**

E

Eades, P. *see:* **Optimization in leveled graphs**
Eaves, B.C. *see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Spatial price equilibrium**
Edelsbrunner, H. *see:* **Hyperplane arrangements in optimization**
Edirisinghe, N.C.P. *see:* **Stochastic programs with recourse: upper bounds**
Edmonds, J. *see:* **Assignment and matching**; **Combinatorial optimization games**; **Complexity classes in optimization**; **Complexity of degeneracy**; **Criss-cross pivoting rules**; **Integer programming: cutting plane algorithms**; **Matroids**; **Maximum flow problem**; **Oriented matroids**
Edmundson, H.P. *see:* **Multistage stochastic programming: barycentric approximation**; **Stochastic programs with recourse: upper bounds**
Ekeland, I. *see:* **Duality in optimal control with first order differential equations**; **Financial applications of multicriteria analysis**
El-Halwagi, M.M. *see:* **Interval analysis: application to chemical engineering design problems**; **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: mass and heat exchanger networks**
El-Yaniv, R. *see:* **Competitive ratio for portfolio management**
Elber, R. *see:* **Simulated annealing methods in protein folding**
ElGhaoui, L. *see:* **Robust control**
Elias, D. *see:* **Capacitated minimum spanning trees**
Elias, P. *see:* **Maximum flow problem**
Elton, E.J. *see:* **Portfolio selection: markowitz mean-variance model**
Elveback, L.R. *see:* **Resource allocation for epidemic control**
Ely, J.S. *see:* **Interval analysis: differential equations**
Elzinga, J. *see:* **Single facility location: circle covering problem**
Enke, S. *see:* **Equilibrium networks**; **Spatial price equilibrium**
Ensor, K.B. *see:* **Monte-Carlo simulations for stochastic optimization**

Enthoven, A.C. *see:* **Generalized concavity in multi-objective optimization**
Eppley, P.H. *see:* **Heuristics for maximum clique and independent set**
Erdös, P. *see:* **Feedback set problems**
Eremin, I.I. *see:* **Fejér monotonicity in convex optimization**
Ernst, R. *see:* **Operations research models for supply chain management and design**
Esbensen, H. *see:* **Evolutionary algorithms in combinatorial optimization**
Euler, L. *see:* **Lagrange, Joseph-Louis**; **Nonlinear least squares problems**; **Second order optimality conditions for nonlinear optimization**; **Variational principles**
Even, G. *see:* **Feedback set problems**
Even, S. *see:* **Graph planarization**
Everett, H. *see:* **Integer programming: lagrangian relaxation**
Evers, J.J.M. *see:* **Lemke method**
Evrard, Y. *see:* **Portfolio selection and multicriteria analysis**
Evtushenko, Y. *see:* **ABS algorithms for optimization**

F

Fábián, Cs.I. *see:* **Cutting-stock problem**
Facchinei, F. *see:* **Implicit lagrangian**
Fagnano, J.Fr. *see:* **History of optimization**
Fahringer, T. *see:* **Alignment problem**
Falk, J.E. *see:* **Fractional programming**; **Global optimization in generalized geometric programming**
Falkenauer, E. *see:* **Evolutionary algorithms in combinatorial optimization**
Fama, E. *see:* **Portfolio selection: markowitz mean-variance model**
Fan, K. *see:* **Minimax theorems**
Fan, M.K.H. *see:* **Robust control**
Farkas, J. *see:* **Farkas lemma**; **Farkas lemma: generalizations**; **Second order optimality conditions for nonlinear optimization**; **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**
Faustino, A. *see:* **Bilevel programming: global optimization**
Feautrier, P. *see:* **Alignment problem**
Federgruen, A. *see:* **Operations research models for supply chain management and design**
Fejér, L. *see:* **Fejér monotonicity in convex optimization**
Fekete, M. *see:* **Logconcavity of discrete distributions**; **Von Neumann, John**
Feldman, R. *see:* **Minimax game tree searching**
Fenchel, W. *see:* **Duality in optimal control with first order differential equations**
Feng, Q. *see:* **Steiner tree problems**
Fenstein, A. *see:* **Maximum flow problem**
Feo, T.A. *see:* **Feedback set problems**
Ferguson, M.J. *see:* **Capacitated minimum spanning trees**
Ferguson, O. *see:* **Preference disaggregation**
Ferland, J.A. *see:* **Evolutionary algorithms in combinatorial optimization**; **Fractional programming**; **Heuristics for maximum clique and independent set**; **Multicommodity flow problems**
Fermat, P. *see:* **Global optimization in Weber's problem with attraction and repulsion**; **Network design problems**; **Steiner tree problems**; **Variational principles**
Feron, E. *see:* **Robust control**
Ferreira, C. *see:* **Multi-objective mixed integer programming**
Feyerbend, P. *see:* **Multi-objective mixed integer programming**
Fiacco, A.V. *see:* **Linear programming: interior point methods**
Fichera, G. *see:* **Nonconvex energy functions: hemivariational inequalities**; **Quasivariational inequalities**
Filliman, P. *see:* **Volume computation for polytopes: strategies and performances**
Fincke, U. *see:* **Communication network assignment problem**; **Quadratic assignment problem**
Finkel, R.A. *see:* **Minimax game tree searching**
Fischer, A. *see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**
Fischer, H. *see:* **Automatic differentiation: point and interval**
Fishburn, J.P. *see:* **Minimax game tree searching**
Fishburn, P.C. *see:* **Decision support systems with multiple criteria**
Fisher, M.L. *see:* **Operations research models for supply chain management and design**
Fisher, R.A. *see:* **Multicriteria sorting methods**; **Statistical classification: optimization approaches**
Fisher, W.D. *see:* **Dynamic programming in clustering**
Fletcher, R. *see:* **Automatic differentiation: calculation of Newton steps**; **Broyden family of methods and the BFGS update**; **Conjugate-gradient methods**; **MINLP: branch and bound methods**; **MINLP: outer approximation algorithm**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares problems**; **Rosen's method, global convergence, and Powell's conjecture**; **Semidefinite programming and determinant maximization**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: solution by active sets and interior point methods**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**
Fleurent, C. *see:* **Evolutionary algorithms in combinatorial optimization**; **Greedy randomized adaptive search procedures**; **Heuristics for maximum clique and independent set**
Florian, M. *see:* **Spatial price equilibrium**
Floudas, C.A. *see:* **Bilevel optimization: feasibility test and flexibility index**; **Global optimization: application to phase equilibrium problems**; **Global optimization in generalized geometric programming**; **Global optimization in Lennard–Jones and morse clusters**; **MINLP: applications in blending and pooling problems**; **MINLP: heat exchanger network synthesis**; **MINLP: mass and heat exchanger networks**
Folkman, J. *see:* **Oriented matroids**
Fonseka, J. *see:* **Survivable networks**
Fontanari, J.F. *see:* **Multidimensional knapsack problems**
Ford Jr., L.R. *see:* **Network design problems**
Ford, L.R. *see:* **History of optimization**; **Maximum flow problem**; **Shortest path tree algorithms**
Fortuny-Amat, J. *see:* **Bilevel programming: global optimization**; **Bilevel programming in management**
Foster, J.A. *see:* **Heuristics for maximum clique and independent set**

Foulds, L.R. *see:* **Multicommodity flow problems**
Fourer, R. *see:* **Discontinuous optimization**
Fourier, J.B.J. *see:* **Farkas lemma**; **Fourier–Motzkin elimination method**; **History of optimization**; **Second order optimality conditions for nonlinear optimization**
Fox, Ch. *see:* **Optimization in classifying text documents**
Fox, G.E. *see:* **Multidimensional knapsack problems**
Fox, K. *see:* **Time-dependent traveling salesman problem**
Francis, J.C. *see:* **Financial equilibrium**
Francis, R.L. *see:* **Network location: covering problems**
Frank, H. *see:* **Capacitated minimum spanning trees**
Frank, M. *see:* **Frank–Wolfe algorithm**
Frankel, S. *see:* **Metropolis, Nicholas Constantine**
Frederick the Great *see:* **Lagrange, Joseph-Louis**
Freed, N. *see:* **Multicriteria sorting methods**; **Preference disaggregation**
Freedman, B.A. *see:* **Linear programming: interior point methods**
Freisleben, B. *see:* **Evolutionary algorithms in combinatorial optimization**
Freling, R. *see:* **Operations research models for supply chain management and design**
Frenk, J.B.G. *see:* **Fractional programming**
Frenk, J.C.B. *see:* **Quadratic assignment problem**
Freund, R.W. *see:* **Fractional programming**
Freville, A. *see:* **Multidimensional knapsack problems**
Friden, C. *see:* **Heuristics for maximum clique and independent set**
Fridman, B. *see:* **Hilbert's thirteenth problem**
Friedman, L. *see:* **Operations research**
Friedrich, C.M. *see:* **Resource allocation for epidemic control**
Friedrich, K. *see:* **Variational principles**
Friedrichs, K. *see:* **Duality in optimal control with first order differential equations**
Frieze, A.M. *see:* **Multidimensional knapsack problems**; **Quadratic assignment problem**
Frobenius, G. *see:* **Carathéodory, Constantine**
Frome, E.L. *see:* **Overdetermined systems of linear equations**
Fu, M. *see:* **Complexity theory: quadratic programming**
Fuchs, L. *see:* **Carathéodory, Constantine**
Fujii, Y. *see:* **Interval global optimization**
Fujishige, S. *see:* **Combinatorial optimization algorithms in resource allocation problems**
Fujito, T. *see:* **Feedback set problems**
Fukuda, K. *see:* **Complexity of degeneracy**; **Criss-cross pivoting rules**
Fukushima, M. *see:* **Implicit lagrangian**
Fulkerson, D.R. *see:* **History of optimization**; **Maximum flow problem**; **Network design problems**
Funke, M. *see:* **Feedback set problems**

G

Gabasov, R. *see:* **Semi-infinite programming and applications in finance**
Gabay, D. *see:* **Oligopolistic market equilibrium**
Gabow, H.N. *see:* **Maximum flow problem**
Gainanov, D.N. *see:* **Inference of monotone boolean functions**
Galantai, A. *see:* **ABS algorithms for linear equations and linear least squares**
Gale, D. *see:* **Preprocessing in stochastic programming**; **Tucker homogeneous systems of linear relations**
Galen of Pergamon *see:* **Boolean and fuzzy relations**
Galilei, G. *see:* **History of optimization**
Galli, M. *see:* **Dynamic programming: optimal control applications**
Galois, E. *see:* **Hilbert's thirteenth problem**; **Lagrange, Joseph-Louis**
Galperin, E.A. *see:* **Interval analysis: systems of nonlinear equations**
Gambardella, L.M. *see:* **Evolutionary algorithms in combinatorial optimization**
Ganeshan, R. *see:* **Inventory management in supply chains**; **Operations research models for supply chain management and design**
Gao, B. *see:* **Steiner tree problems**
Garcia, C.B. *see:* **Lemke method**
Garcia, J. *see:* **Alignment problem**
Gardner, L.M. *see:* **Survivable networks**
Gass, S.I. *see:* **Probabilistic analysis of simplex algorithms**
Gassmann, H. *see:* **Approximation of multivariate probability integrals**
Gaudioso, M. *see:* **Fractional programming**
Gauss, C.F. *see:* **Fundamental theorem of algebra**; **Gauss, Carl Friedrich**; **Gauss–Newton method: Least squares, relation to Newton's method**; **History of optimization**; **Lagrange, Joseph-Louis**; **Least squares problems**; **Network design problems**; **Nonlinear least squares problems**; **Second order optimality conditions for nonlinear optimization**; **Steiner tree problems**
Gavish, B. *see:* **Capacitated minimum spanning trees**; **Time-dependent traveling salesman problem**
Gay, D.M. *see:* **Nonlinear least squares: Newton-type methods**; **Quadratic programming over an ellipsoid**
Gehrlein, W.V. *see:* **Statistical classification: optimization approaches**
Geiger, D. *see:* **Feedback set problems**
Geitner, U. *see:* **Automatic differentiation: calculation of Newton steps**
Gelatt, C.D. *see:* **Heuristics for maximum clique and independent set**; **Quadratic assignment problem**; **Simulated annealing**
Gendreau, M. *see:* **Heuristics for maximum clique and independent set**
Gengler, M. *see:* **Alignment problem**; **Minimax game tree searching**
Genz, A. *see:* **Approximation of multivariate probability integrals**
Geoffrion, A.M. *see:* **MINLP: generalized cross decomposition**; **Multicommodity flow problems**; **Operations research models for supply chain management and design**; **Simplicial decomposition**
George, J.A. *see:* **Least squares problems**
Geraghty, M.A. *see:* **Minimax theorems**
Gerchak, Y. *see:* **Operations research models for supply chain management and design**
Gerla, M. *see:* **Simplicial decomposition**
Gersht, A. *see:* **Multicommodity flow problems**
Ghouila-Houri, M.A. *see:* **Minimax theorems**
Giannessi, F. *see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**

Gibbons, L.E. *see:* **Heuristics for maximum clique and independent set**
Gilbert, E.N. *see:* **Steiner tree problems**
Gill, P.E. *see:* **Nonlinear least squares: Newton-type methods**; **Successive quadratic programming: applications in the process industry**
Gilmore, P.C. *see:* **Cutting-stock problem**; **Fractional programming**; **Quadratic assignment problem**
Giunta, A.A. *see:* **Multidisciplinary design optimization**
Givens, W. *see:* **QR factorization**
Glover *see:* **Multidimensional knapsack problems**
Glover, F. *see:* **Disjunctive programming**; **Greedy randomized adaptive search procedures**; **Heuristics for maximum clique and independent set**; **Linear ordering problem**; **Maximum satisfiability problem**; **Multicriteria sorting methods**; **Multidimensional knapsack problems**; **Simplicial pivoting algorithms for integer programming**
Glover, G. *see:* **Preference disaggregation**
Glynn, P.W. *see:* **Monte-Carlo simulations for stochastic optimization**
Gochet, W. *see:* **Multicriteria sorting methods**
Goeleven, D. *see:* **Nonconvex-nonsmooth calculus of variations**
Goemans, M.X. *see:* **Feedback set problems**; **Survivable networks**
Goffin, J.-L. *see:* **Nondifferentiable optimization: subgradient optimization methods**
Goh, K.C. *see:* **Robust control**
Goldberg, A.V. *see:* **Maximum flow problem**
Goldfarb, D. *see:* **Broyden family of methods and the BFGS update**; **Linear programming: Klee–Minty examples**; **Numerical methods for unary optimization**; **Rosen's method, global convergence, and Powell's conjecture**; **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**
Goldfeld, S.M. *see:* **Nonlinear least squares: trust region methods**
Goldman, A.J. *see:* **Homogeneous selfdual methods for linear programming**; **Linear optimization: theorems of the alternative**; **Tucker homogeneous systems of linear relations**
Goldschmidt, O. *see:* **Graph planarization**
Goldstein, A. *see:* **Local attractors for gradient-related descent iterations**
Goldstein, A.A. *see:* **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**
Goldstine, H. *see:* **Von Neumann, John**
Golub, G.H. *see:* **Least squares problems**
Gomez, S. *see:* **Direct global optimization algorithm**
Gomory, R.E. *see:* **Cutting-stock problem**; **Fractional programming**; **History of optimization**; **Integer programming**; **Integer programming: cutting plane algorithms**
Gonzaga, C.C. *see:* **Linear programming: interior point methods**
Gonzalez, T. *see:* **Quadratic assignment problem**
González-Velarde, J.L. *see:* **Greedy randomized adaptive search procedures**
Gordan, P. *see:* **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**
Gorges-Schleuter, M. *see:* **Evolutionary algorithms in combinatorial optimization**
Gossen, H. *see:* **Financial equilibrium**
Gouveia, L. *see:* **Capacitated minimum spanning trees**; **Time-dependent traveling salesman problem**
Granot, D. *see:* **Combinatorial optimization games**
GRASP *see:* **Graph planarization**
Graves, G.W. *see:* **Multicommodity flow problems**; **Operations research models for supply chain management and design**
Graves, S. *see:* **Time-dependent traveling salesman problem**
Graves, S.C. *see:* **Inventory management in supply chains**; **Operations research models for supply chain management and design**
Green, K.A. *see:* **Global optimization: application to phase equilibrium problems**
Greenberg, D.E. *see:* **Quadratic semi-assignment problem**
Greenberg, H. *see:* **Linear optimization: theorems of the alternative**
Greenberg, H.J. *see:* **Preprocessing in stochastic programming**
Greenhalgh, D. *see:* **Resource allocation for epidemic control**
Griewank, A. *see:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: calculation of Newton steps**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: parallel computation**; **Automatic differentiation: point and interval**
Griggs, J.R. *see:* **Optimization in leveled graphs**
Grippo, L. *see:* **Nonlinear least squares problems**
Grossman, B. *see:* **Multidisciplinary design optimization**
Grossman, T. *see:* **Heuristics for maximum clique and independent set**
Grossmann, I.E. *see:* **Bilevel optimization: feasibility test and flexibility index**; **Chemical process planning**; **Generalized outer approximation**; **MINLP: heat exchanger network synthesis**; **MINLP: logic-based methods**; **MINLP: outer approximation algorithm**; **Mixed integer linear programming: heat exchanger network synthesis**; **Time-dependent traveling salesman problem**
Grötschel, M. *see:* **Assignment and matching**; **Survivable networks**
Grotzinger, S.J. *see:* **Isotonic regression problems**
Gruber, M.J. *see:* **Portfolio selection: markowitz mean-variance model**; **Preference disaggregation**
Gu Jun *see:* **Modeling languages in optimization: a new paradigm**
Guélat, J. *see:* **Frank–Wolfe algorithm**
Gugat, M. *see:* **Fractional programming**
Guignard, M. *see:* **Multidimensional knapsack problems**
Guisewite, G.M. *see:* **Nonconvex network flow problems**
Gumus, Z.H. *see:* **MINLP: reactive distillation column synthesis**
Gupta, M. *see:* **Alignment problem**
Gustafson, S.-Å. *see:* **Semi-infinite programming and applications in finance**; **Semi-infinite programming, semidefinite programming and perfect duality**

H

Ha, Ch.-W. *see:* **Farkas lemma: generalizations**

Habetler, G.J. *see:* **Generalized nonlinear complementarity problem**
Haftka, R.T. *see:* **Multidisciplinary design optimization**
Hahn, H. *see:* **Theorems of the alternative and optimization**
Haimes, Y.Y. *see:* **Dynamic programming: optimal control applications**
Haimovich, M. *see:* **Probabilistic analysis of simplex algorithms**
Hájek, P. *see:* **Checklist paradigm semantics for fuzzy logics**
Hakimi, S.B. *see:* **Network design problems**
Hale, W.K. *see:* **Frequency assignment problem**
Halemane, K.P. *see:* **Bilevel optimization: feasibility test and flexibility index**
Halkin, H. *see:* **Infinite horizon control and dynamic games**
Hall, L. *see:* **Capacitated minimum spanning trees**
Hall, L.H. *see:* **Complexity of degeneracy**
Hall, M.A. *see:* **Traffic network equilibrium**
Halley, E. *see:* **Lagrange, Joseph-Louis**
Halsey, E. *see:* **Multiple objective dynamic programming**
Hamel, G. *see:* **Second order optimality conditions for nonlinear optimization**
Hamilton, W. *see:* **Lagrange, Joseph-Louis**
Han, C.G. *see:* **Quadratic knapsack**
Han, J. *see:* **Rosen's method, global convergence, and Powell's conjecture**
Han, S.P. *see:* **Nonlinear least squares problems**; **Successive quadratic programming**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: full space methods**
Han, X. *see:* **Graph planarization**
Hanafi, S. *see:* **Multidimensional knapsack problems**
Hansel, G. *see:* **Inference of monotone boolean functions**
Hansen, E.R. *see:* **Global optimization methods for harmonic retrieval**; **Interval analysis: systems of nonlinear equations**; **Interval global optimization**; **Interval linear systems**
Hansen, P. *see:* **Heuristics for maximum clique and independent set**; **Maximum satisfiability problem**
Hansmann, U.H.E. *see:* **Simulated annealing methods in protein folding**
Hanson, M.A. *see:* **Invexity and its applications**
Harker, P.T. *see:* **Estimating data for multicriteria decision making problems: optimization techniques**
Harris, F. *see:* **Inventory management in supply chains**
Hart, J.P. *see:* **Greedy randomized adaptive search procedures**
Hartig, F. *see:* **Boundary condition iteration BCI**
Hartley, H.O. *see:* **Nonlinear least squares: Newton-type methods**
Hartley, R. *see:* **Multiple objective dynamic programming**; **Simulated annealing methods in protein folding**
Hartman, G.J. *see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**
Hartman, J.K. *see:* **Multicommodity flow problems**
Hartman, P. *see:* **Variational inequalities**
Hartmanis, J. *see:* **Complexity theory**
Hasham, R. *see:* **Portfolio selection and multicriteria analysis**
Hassouni, A. *see:* **Generalized monotone multivalued maps**
Haurie, H. *see:* **Infinite horizon control and dynamic games**
Hausdorff, F. *see:* **Vector optimization**
Haverly, C.A. *see:* **MINLP: applications in blending and pooling problems**
Havránek, T. *see:* **Checklist paradigm semantics for fuzzy logics**
He Jifeng *see:* **Boolean and fuzzy relations**
Healy, P. *see:* **Optimization in leveled graphs**
Hearn, D.W. *see:* **Heuristics for maximum clique and independent set**; **Simplicial decomposition**; **Simplicial decomposition algorithms**; **Single facility location: circle covering problem**
Heath, M.T. *see:* **Least squares problems**
Hebden, M.D. *see:* **Nonlinear least squares: trust region methods**
Hedayat, G.A. *see:* **Alignment problem**
Helgason, R.V. *see:* **Convex-simplex algorithm**; **Multicommodity flow problems**
Helly, E. *see:* **Theorems of the alternative and optimization**
Henig, M. *see:* **Operations research models for supply chain management and design**
Hensel, C.R. *see:* **Portfolio selection: markowitz mean-variance model**
Herman, G.T. *see:* **Maximum entropy principle: image reconstruction**
Hermite, Ch. *see:* **Symmetric systems of linear equations**
Herskovits, J.N. *see:* **Feasible sequential quadratic programming**
Hestenes, M.R. *see:* **Conjugate-gradient methods**; **Lagrangian multipliers methods for convex programming**; **Nonlinear least squares problems**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**
Hethcote, H.W. *see:* **Resource allocation for epidemic control**
Hettich, R. *see:* **Semi-infinite programming, semidefinite programming and perfect duality**
Hewett, R. *see:* **Minimax game tree searching**
Hext, G.R. *see:* **Sequential simplex method**
Hickey, T. *see:* **Volume computation for polytopes: strategies and performances**
Hicks, J.R. *see:* **Financial equilibrium**; **Portfolio selection: markowitz mean-variance model**
Hifi, M. *see:* **Heuristics for maximum clique and independent set**
Higle, J.L. *see:* **Monte-Carlo simulations for stochastic optimization**
Hilbert, D. *see:* **Hilbert's thirteenth problem**
Hildebrandt, S. *see:* **Duality in optimal control with first order differential equations**
Hildreth, C. *see:* **Splitting method for linear complementarity problems**
Hill, A.V. *see:* **Multicriteria sorting methods**
Hillestad, R.J. *see:* **Reverse convex optimization**
Hillstrom, K.E. *see:* **Automatic differentiation: introduction, history and rounding error estimation**
Himmelblau, D.M. *see:* **Chemical process planning**
Himsworth, F.R. *see:* **Sequential simplex method**
Hinkins, R.L. *see:* **Automatic differentiation: parallel computation**
Hisdal, E. *see:* **Checklist paradigm semantics for fuzzy logics**
Hitchcock, F.L. *see:* **History of optimization**
Hoare, C.A.R. *see:* **Boolean and fuzzy relations**
Hoare, M.R. *see:* **Global optimization in Lennard–Jones and morse clusters**
Hobson, A. *see:* **Entropy optimization: shannon measure of entropy and its properties**

Hochbaum, D. *see:* **Feedback set problems**
Hochbaum, D.S. *see:* **Combinatorial optimization algorithms in resource allocation problems**
Hodder, J.E. *see:* **Operations research models for supply chain management and design**
Hodges, S.D. *see:* **Portfolio selection: markowitz mean-variance model**
Hoffman, A.J. *see:* **Complexity of degeneracy**; **Least-index anticycling rules**; **Lexicographic pivoting rules**; **Preprocessing in stochastic programming**
Höfner, G. *see:* **Probabilistic analysis of simplex algorithms**
Hogarth, R.M. *see:* **Competitive ratio for portfolio management**
Hohl, D. *see:* **Global optimization in Lennard–Jones and morse clusters**
Holland, J.H. *see:* **Forecasting**; **Genetic algorithms**
Holland, O. *see:* **Assignment and matching**
Holloway, C.A. *see:* **Simplicial decomposition algorithms**
Holmberg, K. *see:* **Piecewise linear network flow problems**
Holmes, D.F. *see:* **Stochastic programming: parallel factorization of structured matrices**
Holt, C.C. *see:* **Operations research models for supply chain management and design**
Holyoak, K. *see:* **Forecasting**
Homer, S. *see:* **Heuristics for maximum clique and independent set**
Hong, S. *see:* **Combinatorial optimization algorithms in resource allocation problems**
Hooker, J.N. *see:* **Simplicial pivoting algorithms for integer programming**
Hopcroft, J. *see:* **Graph planarization**
Hopfield, J.J. *see:* **Heuristics for maximum clique and independent set**; **Neural networks for combinatorial optimization**
Horn, A. *see:* **Combinatorial matrix analysis**
Hoskins, J.A. *see:* **Fractional programming**
Householder, A.S. *see:* **QR factorization**
Hovland, P. *see:* **Automatic differentiation: parallel computation**
Hrymak, A.N. *see:* **Successive quadratic programming: applications in the process industry**
Hu, X.-D. *see:* **Rosen's method, global convergence, and Powell's conjecture**
Hua, J.Z. *see:* **Global optimization: application to phase equilibrium problems**
Huang, C.-H. *see:* **Alignment problem**
Huang, Z. *see:* **Information-based complexity and information-based optimization**
Huber, P.J. *see:* **Overdetermined systems of linear equations**
Hughes, M. *see:* **Financial equilibrium**
Huhn, P. *see:* **Probabilistic analysis of simplex algorithms**
Hung, P.-F. *see:* **Homogeneous selfdual methods for linear programming**
Hunter, D. *see:* **Approximation of multivariate probability integrals**
Hurson, Ch. *see:* **Portfolio selection and multicriteria analysis**
Hurwicz, L. *see:* **Multilevel optimization in mechanics**
Huygens, Chr. *see:* **Leibniz, gottfried wilhelm**
Hwang, F.K. *see:* **Steiner tree problems**

I

Ibaraki, T. *see:* **Fractional programming**; **Minimax game tree searching**; **Quadratic fractional programming: Dinkelbach method**
Ichida, K. *see:* **Interval global optimization**
Idnani, A. *see:* **Successive quadratic programming: applications in the process industry**
Iglesias, O.A. *see:* **Chemical process planning**
Ikey, M. *see:* **Alignment problem**
Imo, I.I. *see:* **Discontinuous optimization**
Infanger, G. *see:* **Monte-Carlo simulations for stochastic optimization**
Inori, M. *see:* **Fractional programming**
Ioffe, A.D. *see:* **Composite nonsmooth optimization**
Iosifescu, M. *see:* **Stochastic programming: minimax approach**
Iri, M. *see:* **Automatic differentiation: point and interval**
Irle, A. *see:* **Minimax theorems**
Isac, G. *see:* **Lemke method**
Isbell, J.R. *see:* **Quadratic fractional programming: Dinkelbach method**
Iserman, H. *see:* **Multi-objective optimization: lagrange duality**
Ishii, H. *see:* **Variational inequalities: projected dynamical system**

J

Jaakola, T.H.I. *see:* **Direct search Luus—Jaakola optimization procedure**; **Dynamic programming: optimal control applications**
Jablonskij, S.W. *see:* **Finite complete systems of many-valued logic algebras**
Jacobi, C.G. *see:* **QR factorization**; **Symmetric systems of linear equations**; **Variational principles**
Jacobsen, S.K. *see:* **Operations research models for supply chain management and design**
Jacquet-Lagrèze, E. *see:* **Preference disaggregation**
Jagota, A. *see:* **Heuristics for maximum clique and independent set**
Jahn, J. *see:* **Multi-objective optimization: lagrange duality**
Jaikumar, R. *see:* **Operations research models for supply chain management and design**
James, R.C. *see:* **Minimax theorems**
Jameson, A. *see:* **Design optimization in computational fluid dynamics**
Jarre, F. *see:* **Fractional programming**
Jaszkiewicz, A. *see:* **Multi-objective combinatorial optimization**
Jaumard, B. *see:* **Heuristics for maximum clique and independent set**; **Maximum satisfiability problem**
Jayaraman, V. *see:* **Operations research models for supply chain management and design**
Jaynes, E.T. *see:* **Entropy optimization: shannon measure of entropy and its properties**; **Jaynes' maximum entropy principle**
Jefferson, T.R. *see:* **Fractional programming**
Jenkins, G.M. *see:* **Forecasting**
Jensen, D. *see:* **Criss-cross pivoting rules**
Jensen, J.L. *see:* **Multistage stochastic programming: barycentric approximation**

Jernigan, R.L. *see:* **Molecular structure determination: convex global underestimation**
Jeroslow, R.G. *see:* **Bilevel linear programming**; **Disjunctive programming**; **Linear programming: Klee–Minty examples**
Jerrum, M. *see:* **Heuristics for maximum clique and independent set**
Jevons, W. *see:* **Financial equilibrium**
Jeyakumar, V. *see:* **Composite nonsmooth optimization**
Jiang, T. *see:* **Steiner tree problems**
Jimenez, A.G. *see:* **Chemical process planning**
Jin, H. *see:* **Multicommodity flow problems**
Joachimsthaler, E.A. *see:* **Multicriteria sorting methods**
John, F. *see:* **Parametric global optimization: sensitivity**; **Semi-infinite programming: second order optimality conditions**; **Theorems of the alternative and optimization**
John, Fritz *see:* **Nondifferentiable optimization: parametric programming**
Johnson, D.S. *see:* **Evolutionary algorithms in combinatorial optimization**; **Maximum satisfiability problem**; **Quadratic assignment problem**
Johnson, E.L. *see:* **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**
Johnson, T.A. *see:* **Quadratic assignment problem**
Jones, D.F. *see:* **Portfolio selection and multicriteria analysis**
Jones, K.L. *see:* **Multicommodity flow problems**
Jones, R.O. *see:* **Global optimization in Lennard–Jones and morse clusters**
Jongen, H.Th. *see:* **Parametric global optimization: sensitivity**; **Parametric optimization: embeddings, path following and singularities**
Jonker, P. *see:* **Parametric global optimization: sensitivity**; **Parametric optimization: embeddings, path following and singularities**
Joó, I. *see:* **Minimax theorems**
Jorion, P. *see:* **Portfolio selection: markowitz mean-variance model**
Jörnsten, K. *see:* **Multidimensional knapsack problems**
Judge, G.C. *see:* **Spatial price equilibrium**
Judice, J. *see:* **Bilevel programming: global optimization**
Juel, H. *see:* **MINLP: application in facility location-allocation**
Jünger, M. *see:* **Graph planarization**; **Optimization in leveled graphs**

K

Kabadi, S.N. *see:* **Complexity of degeneracy**; **Complexity theory: quadratic programming**; **Quadratic knapsack**; **Second order optimality conditions for nonlinear optimization**
Kagiwada, H. *see:* **Automatic differentiation: introduction, history and rounding error estimation**
Kahng, A. *see:* **Steiner tree problems**
Kaibel, V. *see:* **Quadratic assignment problem**
Kakutani, S. *see:* **Minimax theorems**
Kalaba, R.E. *see:* **Estimating data for multicriteria decision making problems: optimization techniques**
Kalantari, B. *see:* **Concave programming**; **Convex envelopes in optimization problems**
Kalitventzeff, B. *see:* **Successive quadratic programming: applications in the process industry**
Kall, P. *see:* **Approximation of multivariate probability integrals**; **Monte-Carlo simulations for stochastic optimization**
Kallberg, J.G. *see:* **Portfolio selection: markowitz mean-variance model**
Kalman, R.E. *see:* **Semi-infinite programming and applications in finance**
Kalogerakis, N. *see:* **Direct search Luus—Jaakola optimization procedure**
Kamaratou, I. *see:* **Portfolio selection and multicriteria analysis**
Kantorovich, L.V. *see:* **Complexity of gradients, Jacobians, and Hessians**; **History of optimization**; **Kantorovich, Leonid Vitalyevich**
Kany, G. *see:* **Multilevel optimization in mechanics**
Kanzow, C. *see:* **Implicit lagrangian**
Kaplan, E.H. *see:* **Resource allocation for epidemic control**
Kaplan, H. *see:* **Quadratic assignment problem**
Kapoor, S. *see:* **Complexity theory: quadratic programming**
Kapsiotis, G. *see:* **Operations research models for supply chain management and design**
Kapur, J.N. *see:* **Entropy optimization: shannon measure of entropy and its properties**
Karamardian, S. *see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Oligopolistic market equilibrium**
Karkazis, J. *see:* **Communication network assignment problem**
Karlin, S. *see:* **Inventory management in supply chains**
Karloff, H. *see:* **Maximum satisfiability problem**
Karmarkar, N.K. *see:* **Complexity theory: quadratic programming**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Nondifferentiable optimization: cutting plane methods**; **Pivoting algorithms for linear programming generating two paths**; **Rosen's method, global convergence, and Powell's conjecture**
Karmarkar, U.S. *see:* **Inventory management in supply chains**
Karnaugh, M. *see:* **Capacitated minimum spanning trees**
Karp, R.M. *see:* **Complexity classes in optimization**; **Complexity theory**; **Heuristics for maximum clique and independent set**; **Maximum flow problem**; **Minimax game tree searching**; **Network design problems**; **Probabilistic analysis of simplex algorithms**
Karpinski, M. *see:* **Steiner tree problems**
Karst, O.J. *see:* **Preference disaggregation**
Karush, W. *see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Second order optimality conditions for nonlinear optimization**
Karwan, M.H. *see:* **Bilevel linear programming**; **Bilevel programming in management**
Karzanov, V.V. *see:* **Combinatorial optimization algorithms in resource allocation problems**
Kass, M. *see:* **Optimization in medical imaging**
Kataoka, S. *see:* **Probabilistic constrained problems: convexity theory**; **Stochastic programming models: random objective**
Katzela, I. *see:* **Frequency assignment problem**
Kawasaki, H. *see:* **Semi-infinite programming: second order optimality conditions**
Kawatra, R. *see:* **Capacitated minimum spanning trees**
Kearfott, R.B. *see:* **Interval linear systems**
Kececioglu, J. *see:* **Optimization in leveled graphs**
Kedem, G. *see:* **Automatic differentiation: point and interval**

Keeney, R.L. *see:* **Preference disaggregation approach: basic features, examples from financial decision making**
Keil, F.J. *see:* **Boundary condition iteration BCI**
Keller, H.B. *see:* **Splitting method for linear complementarity problems**
Kelley, J.E. *see:* **Nondifferentiable optimization**; **Nondifferentiable optimization: cutting plane methods**; **Preference disaggregation**
Kelly, W. *see:* **Alignment problem**
Kemperman, J.H.B. *see:* **General moment optimization problems**
Kempthorne, O. *see:* **Conjugate-gradient methods**
Kennedy, K. *see:* **Alignment problem**
Kennington, J.L. *see:* **Convex-simplex algorithm**; **Generalized networks**; **Multicommodity flow problems**
Kepler, J. *see:* **Global optimization in binary star astronomy**
Keresztfalvi, T. *see:* **Fuzzy multi-objective linear programming**
Kernighan, B.W. *see:* **Quadratic assignment problem**; **Time-dependent traveling salesman problem**
Kershenbaum, A. *see:* **Capacitated minimum spanning trees**
Kesavan, H.K. *see:* **Entropy optimization: shannon measure of entropy and its properties**
Khachiyan, L.G. *see:* **Complexity theory: quadratic programming**; **Information-based complexity and information-based optimization**
Khoury, N.T. *see:* **Portfolio selection and multicriteria analysis**
Khuri, S. *see:* **Heuristics for maximum clique and independent set**
Kibzun, A.I. *see:* **Derivatives of probability and integral functions: general theory and examples**
Kim, D. *see:* **Multicommodity flow problems**; **Piecewise linear network flow problems**
Kinchin, A.I. *see:* **Jaynes' maximum entropy principle**
Kindler, J. *see:* **Minimax theorems**
Kirillova, F.M. *see:* **Semi-infinite programming and applications in finance**
Kirkpatrick, S. *see:* **Heuristics for maximum clique and independent set**; **Quadratic assignment problem**; **Simulated annealing**
Kiss, L.N. *see:* **Preference disaggregation**
Kiwiel, K.C. *see:* **Relaxation in projection methods**
Klaus, M. *see:* **Maximum entropy principle: image reconstruction**
Klee, V. *see:* **Criss-cross pivoting rules**; **Linear programming: Klee–Minty examples**
Klein, F. *see:* **Carathéodory, Constantine**
Kleindorfer, P.R. *see:* **Operations research models for supply chain management and design**
Kleitman, D. *see:* **Inference of monotone boolean functions**
Kleitman, D.J. *see:* **Survivable networks**
Klinz, B. *see:* **Biquadratic assignment problem**
Klötzler, R. *see:* **Duality in optimal control with first order differential equations**
Knight, F.H. *see:* **Traffic network equilibrium**
Knobe, K. *see:* **Alignment problem**
Knuth, D. *see:* **Minimax game tree searching**
Kochenberger, G.A. *see:* **Multidimensional knapsack problems**
Kocis, G.R. *see:* **MINLP: outer approximation algorithm**
Koepcke, R. *see:* **Suboptimal control**
Kohavi, Z. *see:* **Feedback set problems**
Kohlberg, E. *see:* **Facility location with externalities**
Kohout, L.J. *see:* **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**
Kojima, M. *see:* **Bilevel programming: implicit function approach**; **Linear programming: interior point methods**; **Nondifferentiable optimization: Newton method**
Kolen, A. *see:* **Network location: covering problems**
Kolmogorov, A.N. *see:* **Hilbert's thirteenth problem**; **Kolmogorov complexity**
Komiya, H. *see:* **Minimax theorems**
Kong, M.Y. *see:* **Nonlinear least squares: Newton-type methods**
König, H. *see:* **Minimax theorems**
Konno, H. *see:* **Fractional programming**; **Global optimization in multiplicative programming**; **Multiplicative programming**
Konnov, I.V. *see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**
Koopman, B.O. *see:* **Combinatorial optimization algorithms in resource allocation problems**
Koopmans, T. *see:* **Kantorovich, Leonid Vitalyevich**
Koopmans, T.C. *see:* **History of optimization**; **Quadratic assignment problem**
Korshunov, A.D. *see:* **Inference of monotone boolean functions**
Korst, J. *see:* **Heuristics for maximum clique and independent set**
Kortanek, K.O. *see:* **Semi-infinite programming, semidefinite programming and perfect duality**
Kostreva, M.M. *see:* **Generalized nonlinear complementarity problem**; **Lemke method**; **Multiple objective dynamic programming**
Kovoor, N. *see:* **Quadratic knapsack**
Kozen, D. *see:* **Replicator dynamics in combinatorial optimization**
Kozlov, M.K. *see:* **Complexity theory: quadratic programming**
Krämer, O. *see:* **Evolutionary algorithms in combinatorial optimization**
Kratsch, D. *see:* **Feedback set problems**
Kraus, A. *see:* **Portfolio selection: markowitz mean-variance model**
Krein, M. *see:* **Krein–Milman theorem**
Kremer, U. *see:* **Alignment problem**
Krishnamurthy, G. *see:* **Minimax game tree searching**
Kronrod, A. *see:* **Hilbert's thirteenth problem**
Krückeberg, F. *see:* **Interval analysis: differential equations**
Kruskal, J.B. *see:* **Matroids**; **Network design problems**
Kuenne, R.E. *see:* **MINLP: application in facility location-allocation**
Küfer, K. *see:* **Probabilistic analysis of simplex algorithms**
Kuhn, H.W. *see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **First order constraint qualifications**; **Global optimization in Weber's problem with attraction and repulsion**; **History of optimization**; **Second order optimality conditions for nonlinear optimization**; **Spatial price equilibrium**
Kullback, S. *see:* **Entropy optimization: parameter estimation**; **Young programming**
Kumar, A. *see:* **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: full space methods**
Kumar, S. *see:* **Facility location with externalities**
Kumar, V. *see:* **Load balancing for parallel optimization techniques**

Kumer, B. *see:* **Nondifferentiable optimization: Newton method**
Kunchithapadam, K. *see:* **Alignment problem**
Kung, D.S. *see:* **Overdetermined systems of linear equations**
Kuno, T. *see:* **Global optimization in multiplicative programming**; **Multiplicative programming**
Kunreuther, H. *see:* **Competitive ratio for portfolio management**
Kushner, H. *see:* **Adaptive global search**
Kuusik, A. *see:* **Optimization in leveled graphs**

L

Labarta, J. *see:* **Alignment problem**
Lagrange, J.L. *see:* **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Integer programming: lagrangian relaxation**; **Lagrange, Joseph-Louis**; **Second order optimality conditions for nonlinear optimization**; **Variational principles**
Laguna, M. *see:* **Greedy randomized adaptive search procedures**; **Linear ordering problem**
Lai, T.L. *see:* **Resource allocation for epidemic control**
Lakshmanan, A. *see:* **MINLP: mass and heat exchanger networks**
Lall, H. *see:* **Multicommodity flow problems**
Lam, M.S. *see:* **Alignment problem**
Lamar, B.W. *see:* **Piecewise linear network flow problems**
Lancaster, K. *see:* **Portfolio selection: markowitz mean-variance model**
Land, A.H. *see:* **Integer programming**; **Integer programming: branch and bound methods**
Lapidus, L. *see:* **Suboptimal control**
Laplace, P.S. *see:* **Lagrange, Joseph-Louis**; **Nonlinear least squares problems**
Laporte, G. *see:* **Operations research models for supply chain management and design**
Larson, R.C. *see:* **Facility location with externalities**
Larsson, T. *see:* **Simplicial decomposition**; **Simplicial decomposition algorithms**
Las Vergnas, M. *see:* **Oriented matroids**
Lasdon, L.C. *see:* **Multilevel optimization in mechanics**
Lasdon, L.S. *see:* **Multicommodity flow problems**
Lasserre, J.B. *see:* **Volume computation for polytopes: strategies and performances**
Lavoisier, A. *see:* **Lagrange, Joseph-Louis**
Lawler, E.L. *see:* **Network design problems**; **Quadratic assignment problem**
Lawphongpanich, S. *see:* **Simplicial decomposition**; **Simplicial decomposition algorithms**
Lawrence, J. *see:* **Oriented matroids**; **Volume computation for polytopes: strategies and performances**
Layn, K.M. *see:* **Successive quadratic programming: solution by active sets and interior point methods**
Leavens, D.H. *see:* **Portfolio selection: markowitz mean-variance model**
Lebourg, G. *see:* **Interval global optimization**
Lee, C.F. *see:* **Portfolio selection and multicriteria analysis**
Lee, D.T. *see:* **Steiner tree problems**
Lee, H.L. *see:* **Operations research models for supply chain management and design**; **Resource allocation for epidemic control**
Lee, J.S. *see:* **Multidimensional knapsack problems**
Lee, K. *see:* **Heuristics for maximum clique and independent set**
Lee, K.C. *see:* **Graph planarization**
Lee, P.L. *see:* **Spatial price equilibrium**
Lee, S. *see:* **MINLP: logic-based methods**
Lee, S.M. *see:* **Portfolio selection and multicriteria analysis**
Lee, Y. *see:* **Shortest path tree algorithms**
Leech, D.J. *see:* **Discontinuous optimization**
Legendre, A.M. *see:* **Least squares problems**; **Nonlinear least squares problems**; **Variational principles**
Lehmann, N.J. *see:* **Eigenvalue enclosures for ordinary differential equations**
Leibler, R.A. *see:* **Entropy optimization: parameter estimation**
Leibniz, G.W. *see:* **Alternative set theory**; **Leibniz, gottfried wilhelm**; **Symmetric systems of linear equations**; **Variational principles**
Leipert, S. *see:* **Graph planarization**
Lemaréchal, C. *see:* **Nondifferentiable optimization: cutting plane methods**; **Nondifferentiable optimization: relaxation methods**
Lemke, C.E. *see:* **Linear complementarity problem**; **Principal pivoting methods for linear complementarity problems**
Lempel, A. *see:* **Graph planarization**
Lesso, W.G. *see:* **MINLP: application in facility location-allocation**
Levenberg, K. *see:* **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**; **Quadratic programming over an ellipsoid**
Levin, O. *see:* **Fractional programming**
Levine, D. *see:* **Evolutionary algorithms in combinatorial optimization**
Levy, A. *see:* **Direct global optimization algorithm**
Lewis, R.M. *see:* **Optimal design in nonlinear optics**
Leyffer, S. *see:* **MINLP: branch and bound methods**; **MINLP: outer approximation algorithm**
L'Huiller, S. *see:* **History of optimization**
Li, D. *see:* **Dynamic programming: optimal control applications**
Li, G. *see:* **Optimal design in nonlinear optics**
Li, J. *see:* **Alignment problem**
Li, M. *see:* **Kolmogorov complexity**
Li, Y. *see:* **Quadratic assignment problem**
Liang, Y.D. *see:* **Feedback set problems**
Liebman, J.S. *see:* **Steiner tree problems**
Lin, B.-L. *see:* **Minimax theorems**
Lin, F. *see:* **Heuristics for maximum clique and independent set**
Lin, G.-L. *see:* **Steiner tree problems**
Lin, S. *see:* **Quadratic assignment problem**; **Time-dependent traveling salesman problem**
Lin, V. *see:* **Hilbert's thirteenth problem**
Linnainmaa, S. *see:* **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: point and interval**
Lintner, J. *see:* **Financial equilibrium**
Litzenberger, R.F. *see:* **Portfolio selection: markowitz mean-variance model**

Liu, M.L. *see:* **Chemical process planning**
Lloyd, E. *see:* **Feedback set problems**
Locatelli, M. *see:* **Random search methods**
Loewner, Ch. *see:* **Symmetric systems of linear equations**
Logemann, G. *see:* **Maximum satisfiability problem**
Lohner, R.J. *see:* **Interval analysis: differential equations**
Longini, I.M. *see:* **Resource allocation for epidemic control**
Lootsma, F. *see:* **Estimating data for multicriteria decision making problems: optimization techniques**
Lorenz, G. *see:* **Hilbert's thirteenth problem**
Los, M. *see:* **Spatial price equilibrium**
Louis XVI *see:* **Lagrange, Joseph-Louis**
Loulou, R. *see:* **Multidimensional knapsack problems**
Lovász, L. *see:* **Disjunctive programming**; **Integer programming**; **Standard quadratic optimization problems: applications**
Love, R.F. *see:* **MINLP: application in facility location-allocation**; **Network design problems**
Loveland, D. *see:* **Maximum satisfiability problem**
Lowe, T.J. *see:* **Network location: covering problems**
Lu, B. *see:* **Steiner tree problems**
Lu, C.L. *see:* **Feedback set problems**
Lu, M. *see:* **Bilevel linear programming**
Luc, D.T. *see:* **Generalized concavity in multi-objective optimization**; **Generalized monotone multivalued maps**
Lucena, A. *see:* **Time-dependent traveling salesman problem**
Lucia, A. *see:* **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**
Lueker, G. *see:* **Graph planarization**
Luenberger, D.G. *see:* **Cyclic coordinate method**
Luhn, H.P. *see:* **Optimization in classifying text documents**
Lukas, J.D. *see:* **Alignment problem**
Luks, K.D. *see:* **Global optimization: application to phase equilibrium problems**
Luo, X.-D. *see:* **Implicit lagrangian**
Luo, Z.-Q. *see:* **Complexity theory: quadratic programming**
Lustig, I. *see:* **Lexicographic pivoting rules**
Lustig, I.J. *see:* **Linear programming: interior point methods**
Luus, R. *see:* **Direct search Luus—Jaakola optimization procedure**; **Dynamic programming: optimal control applications**; **Suboptimal control**
Luyben, W.L. *see:* **Dynamic programming: optimal control applications**
Løkketangen, A. *see:* **Multidimensional knapsack problems**

M

Ma, X.F. *see:* **Nonlinear least squares: Newton-type methods**
Maany, Z. *see:* **Automatic differentiation: calculation of Newton steps**
Macchietto, S. *see:* **MINLP: design and scheduling of batch processes**; **Successive quadratic programming: applications in the process industry**
Macdonald, J.R. *see:* **Generalized total least squares**
Maciejowski, J.M. *see:* **Robust control**
MacKinnon, J.G. *see:* **Spatial price equilibrium**
Madansky, A. *see:* **Multistage stochastic programming: barycentric approximation**; **Stochastic programs with recourse: upper bounds**
Maddux, R.D. *see:* **Boolean and fuzzy relations**
Madsen, K. *see:* **Overdetermined systems of linear equations**
Madsen, O.B.G. *see:* **Operations research models for supply chain management and design**
Maehly, H.J. *see:* **Eigenvalue enclosures for ordinary differential equations**
Magazine, M.J. *see:* **Multidimensional knapsack problems**
Magirou, V.F. *see:* **Quadratic semi-assignment problem**
Magnanti, T.L. *see:* **Multicommodity flow problems**; **Time-dependent traveling salesman problem**
Maheshwari, S.N. *see:* **Maximum flow problem**
Mahmassani, H. *see:* **Dynamic traffic networks**
Makridakis, S. *see:* **Forecasting**
Malandraki, C. *see:* **Time-dependent traveling salesman problem**
Malik, K. *see:* **Capacitated minimum spanning trees**
Mallet-Paret, J. *see:* **Globally convergent homotopy methods**
Malone, M.F. *see:* **Global optimization: application to phase equilibrium problems**
Malozemov, V.N. *see:* **Convex max-functions**
Mandel, A. *see:* **Oriented matroids**
Mangasarian, O.L. *see:* **Generalized concavity in multi-objective optimization**; **Implicit lagrangian**; **Multicriteria sorting methods**
Manne, A.S. *see:* **Chemical process planning**
Manousiouthakis, V. *see:* **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: mass and heat exchanger networks**
Maranas, C.D. *see:* **Global optimization in generalized geometric programming**; **Global optimization in Lennard–Jones and morse clusters**; **Modeling languages in optimization: a new paradigm**
Marchiori, E. *see:* **Heuristics for maximum clique and independent set**
Marcotte, P. *see:* **Frank–Wolfe algorithm**; **Multilevel methods for optimal design**
Markowitz, H.M. *see:* **Competitive ratio for portfolio management**; **Financial applications of multicriteria analysis**; **Financial equilibrium**; **History of optimization**; **Operations research and financial markets**; **Portfolio selection: markowitz mean-variance model**; **Portfolio selection and multicriteria analysis**; **Stochastic programming models: random objective**
Markowski, C.A. *see:* **Multicriteria sorting methods**
Markowski, E.P. *see:* **Multicriteria sorting methods**
Markowsky, G. *see:* **Inference of monotone boolean functions**
Marlin, T.E. *see:* **Successive quadratic programming: applications in the process industry**
Marlow, W.H. *see:* **Quadratic fractional programming: Dinkelbach method**
Marquardt, D.W. *see:* **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**; **Quadratic programming over an ellipsoid**
Marroquin, G. *see:* **Dynamic programming: optimal control applications**
Marschak, J. *see:* **Portfolio selection: markowitz mean-variance model**
Marsland, T.A. *see:* **Minimax game tree searching**

Marsten, M.E. *see:* **Linear programming: interior point methods**
Martel, J.M. *see:* **Portfolio selection and multicriteria analysis**
Martello, S. *see:* **Multidimensional knapsack problems**
Marti, K. *see:* **Derivatives of probability and integral functions: general theory and examples**
Martí, R. *see:* **Greedy randomized adaptive search procedures**
Martin-Loef, P. *see:* **Kolmogorov complexity**
Martinez-Legaz, J.E. *see:* **Fractional programming**
Martins, P. *see:* **Capacitated minimum spanning trees**
Martos, B. *see:* **Fractional programming**
Mason, W.H. *see:* **Multidisciplinary design optimization**
Mata, C. *see:* **Steiner tree problems**
Matsatsinis, N.F. *see:* **Preference disaggregation**
Matstoms, P. *see:* **Least squares problems**
Matsui, T. *see:* **Criss-cross pivoting rules**
Matula, D.W. *see:* **Replicator dynamics in combinatorial optimization**
Mavridou, T. *see:* **Biquadratic assignment problem**
Maxwell, W.L. *see:* **Inventory management in supply chains**
May, R.M. *see:* **Resource allocation for epidemic control**
Mayer, J. *see:* **Approximation of multivariate probability integrals**
Maynard Smith, J. *see:* **Replicator dynamics in combinatorial optimization**
Mazumder, P. *see:* **Evolutionary algorithms in combinatorial optimization**
Mazzola, J.B. *see:* **Multidimensional knapsack problems**
McAllester, D.A. *see:* **Minimax game tree searching**
McAuley, K.B. *see:* **Dynamic programming: optimal control applications**
McCallum, C.J. *see:* **Multicommodity flow problems**
McCarl, B. *see:* **Bilevel programming: global optimization**; **Bilevel programming in management**; **Multicommodity flow problems**
McCarthy, I. *see:* **Minimax game tree searching**
McCormick, G.P. *see:* **Linear programming: interior point methods**; **Numerical methods for unary optimization**; **Rosen's method, global convergence, and Powell's conjecture**; **Sensitivity and stability in NLP: approximation**
McCormick, S.T. *see:* **Combinatorial optimization algorithms in resource allocation problems**
McCullough, W.S. *see:* **Neural networks for combinatorial optimization**
McDiarmid, C.J.H. *see:* **Quadratic assignment problem**
McDonald, C.M. *see:* **Global optimization: application to phase equilibrium problems**
McGeoch, L.A. *see:* **Evolutionary algorithms in combinatorial optimization**
McGill, J. *see:* **Bilevel programming: introduction, history and overview**
McGregor Smith, J. *see:* **Network design problems**
McGuire, C.B. *see:* **Dynamic traffic networks**; **Equilibrium networks**; **Traffic network equilibrium**
McKeown, P. *see:* **Piecewise linear network flow problems**
McNamee, P. *see:* **Financial applications of multicriteria analysis**
Mead, R. *see:* **Sequential simplex method**
Megiddo, N. *see:* **Combinatorial optimization games**; **Fractional programming**; **Information-based complexity and information-based optimization**; **Linear programming: interior point methods**; **Probabilistic analysis of simplex algorithms**; **Single facility location: circle covering problem**
Megretski, A. *see:* **Robust control**
Mehrotra, S. *see:* **Linear programming: interior point methods**
Mei, H.H.W. *see:* **Nonlinear least squares: trust region methods**
Mei-Ko, K. *see:* **Minimum cost flow problem**
Mekarapiruk, W. *see:* **Dynamic programming: optimal control applications**
Meketon, M.J. *see:* **Linear programming: interior point methods**
Melzak, Z.A. *see:* **Steiner tree problems**
Menger, K. *see:* **Matrix completion problems**
Merchant, D.K. *see:* **Dynamic traffic networks**
Mersenne, A. *see:* **History of optimization**
Mertins, U. *see:* **Eigenvalue enclosures for ordinary differential equations**
Merton, R.C. *see:* **Financial equilibrium**
Merz, P. *see:* **Evolutionary algorithms in combinatorial optimization**
Metzger, B.H. *see:* **Frequency assignment problem**
Meyer, A.R. *see:* **Complexity classes in optimization**
Meyer, R.R. *see:* **Multicommodity flow problems**
Meziani, R. *see:* **Preference disaggregation**
Michaelides, E. *see:* **Multidimensional knapsack problems**
Michelsen, M.L. *see:* **Global optimization: application to phase equilibrium problems**; **Global optimization in phase and chemical reaction equilibrium**
Mifflin, R. *see:* **Nondifferentiable optimization: Newton method**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**
Migdalas, A. *see:* **Frank–Wolfe algorithm**; **Simplicial decomposition**
Milani, P. *see:* **Global optimization in Lennard–Jones and morse clusters**
Milis, I.Z. *see:* **Quadratic semi-assignment problem**
Miller, B.L. *see:* **Probabilistic constrained linear programming: duality theory**
Miller, B.P. *see:* **Alignment problem**
Miller, L.B. *see:* **Static stochastic programming models**
Miller, M. *see:* **Financial equilibrium**
Miller, W. *see:* **Automatic differentiation: introduction, history and rounding error estimation**
Milman, D. *see:* **Krein–Milman theorem**
Milyutin, A.A. *see:* **Stochastic programming: nonanticipativity and lagrange multipliers**
Minerbo, G. *see:* **Maximum entropy principle: image reconstruction**
Minkowski, H. *see:* **Carathéodory, Constantine**; **Krein–Milman theorem**
Minty, G.J. *see:* **Criss-cross pivoting rules**; **Linear programming: Klee–Minty examples**
Mirnia, K. *see:* **ABS algorithms for linear equations and linear least squares**
Mitchell, J.E. *see:* **MINLP: branch and bound methods**
Mitchell, J.S.B. *see:* **Steiner tree problems**
Mitjuschin, L.G. *see:* **Generalized monotone multivalued maps**
Miyazawa, S. *see:* **Molecular structure determination: convex global underestimation**

Mizuno, S. *see:* **Homogeneous selfdual methods for linear programming**; **Linear programming: interior point methods**
Modigliani, F. *see:* **Financial equilibrium**; **Operations research models for supply chain management and design**
Mohamed, R.A. *see:* **Generalized networks**
Monaco, M.F. *see:* **Fractional programming**
Mongenet, C. *see:* **Alignment problem**
Monjardet, B. *see:* **Boolean and fuzzy relations**
Monma, C.L. *see:* **Survivable networks**
Monteiro, R.C. *see:* **Linear programming: interior point methods**
Moon, I.D. *see:* **Multiple objective dynamic programming**
Moon, S. *see:* **Operations research models for supply chain management and design**
Moore, E.H. *see:* **Least squares problems**
Moore, R. *see:* **Minimax game tree searching**
Moore, R.E. *see:* **Automatic differentiation: point and interval**; **Bounding derivative ranges**; **Interval analysis: differential equations**; **Interval analysis: systems of nonlinear equations**; **Interval constraints**; **Interval fixed point theory**; **Interval global optimization**; **Interval linear systems**; **Nonlocal sensitivity analysis with automatic differentiation**
Morari, M. *see:* **MINLP: applications in the interaction of design and control**
Moré, J.J. *see:* **Complexity theory: quadratic programming**; **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**; **Quadratic programming over an ellipsoid**
Moreau, J.-J. *see:* **Duality in optimal control with first order differential equations**; **Hemivariational inequalities: applications in mechanics**; **Lagrangian multipliers methods for convex programming**; **Nonconvex energy functions: hemivariational inequalities**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: variational formulations**
Morgenstern, O. *see:* **Decision support systems with multiple criteria**; **Tucker homogeneous systems of linear relations**; **Von Neumann, John**
Morrey, C.B. *see:* **Duality in optimal control with first order differential equations**
Morris, J.G. *see:* **MINLP: application in facility location-allocation**; **Network design problems**
Morton, D.P. *see:* **Stochastic programs with recourse: upper bounds**
Morton, W. *see:* **Successive quadratic programming: applications in the process industry**
Mossin, J. *see:* **Financial equilibrium**
Motreanu, D. *see:* **Nonconvex-nonsmooth calculus of variations**
Motzkin, T. *see:* **Multi-index transportation problems**
Motzkin, T.S. *see:* **Complexity theory: quadratic programming**; **Fejér monotonicity in convex optimization**; **Heuristics for maximum clique and independent set**; **Replicator dynamics in combinatorial optimization**; **Standard quadratic optimization problems: applications**; **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**
Moulin, H. *see:* **Oligopolistic market equilibrium**
Mousseau, V. *see:* **Preference disaggregation**
Mühlenbein, H. *see:* **Evolutionary algorithms in combinatorial optimization**; **Replicator dynamics in combinatorial optimization**
Mulvey, J.M. *see:* **Financial optimization**
Murchland, J.D. *see:* **Simplicial decomposition**
Murota, K. *see:* **Combinatorial optimization algorithms in resource allocation problems**
Murray, F.J. *see:* **Von Neumann, John**
Murray, W. *see:* **Nonlinear least squares: Newton-type methods**; **Successive quadratic programming**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**
Murtagh, B.A. *see:* **Convex-simplex algorithm**; **Successive quadratic programming: applications in the process industry**
Murtagh, B.H. *see:* **Rosen's method, global convergence, and Powell's conjecture**
Murthy, A.S. *see:* **Heuristics for maximum clique and independent set**
Murthy, K.A. *see:* **Quadratic assignment problem**
Murty, K.G. *see:* **Complexity of degeneracy**; **Complexity theory: quadratic programming**; **Criss-cross pivoting rules**; **Least-index anticycling rules**; **Lemke method**; **Linear programming: Klee–Minty examples**; **Piecewise linear network flow problems**; **Principal pivoting methods for linear complementarity problems**; **Quadratic knapsack**; **Second order optimality conditions for nonlinear optimization**
Mutzel, P. *see:* **Graph planarization**; **Optimization in leveled graphs**

N

Nabonna, N. *see:* **Multicommodity flow problems**
Naghshineh, M. *see:* **Frequency assignment problem**
Nagurney, A. *see:* **Dynamic traffic networks**; **Financial equilibrium**; **Oligopolistic market equilibrium**; **Spatial price equilibrium**; **Traffic network equilibrium**; **Variational inequalities: projected dynamical system**
Naitoh, T. *see:* **Combinatorial optimization algorithms in resource allocation problems**
Nakayama, H. *see:* **Portfolio selection and multicriteria analysis**
Naor, J. *see:* **Feedback set problems**
Narducci, R. *see:* **Design optimization in computational fluid dynamics**; **Multidisciplinary design optimization**
Nash, J.F. *see:* **Infinite horizon control and dynamic games**; **Oligopolistic market equilibrium**
Nelder, J.A. *see:* **Sequential simplex method**
Nemhauser, G.L. *see:* **Dynamic traffic networks**; **Integer programming: cutting plane algorithms**; **Nondifferentiable optimization: cutting plane methods**
Nemirovsky, A.S. *see:* **Complexity theory**; **Complexity theory: quadratic programming**; **Fractional programming**; **Information-based complexity and information-based optimization**; **Nondifferentiable optimization: relaxation methods**
Neogi, S. *see:* **Global optimization: hit and run methods**

Nesterov, Yu.E. *see:* **Conjugate-gradient methods**; **Fractional programming**; **Nondifferentiable optimization: relaxation methods**
Neumaier, A. *see:* **Interval linear systems**
Newton, H.N. *see:* **Multicommodity flow problems**
Newton, I. *see:* **Alternative set theory**; **Forecasting**; **Lagrange, Joseph-Louis**; **Leibniz, gottfried wilhelm**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**; **Variational principles**
Nguyen, S. *see:* **Convex-simplex algorithm**
Niculescu, S. *see:* **Robust control**
Niehaus, W. *see:* **Evolutionary algorithms in combinatorial optimization**
Nielsen, C.P. *see:* **Least squares problems**
Nielsen, H.B. *see:* **Overdetermined systems of linear equations**
Nieuwenhuis, J.W. *see:* **Multi-objective optimization: lagrange duality**
Nirenberg, L. *see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**
Nisbett, R. *see:* **Forecasting**
Nishioka, I. *see:* **Graph planarization**
Nobert, Y. *see:* **Operations research models for supply chain management and design**
Nocedal, J. *see:* **Conjugate-gradient methods**; **Rosen's method, global convergence, and Powell's conjecture**; **Semidefinite programming and determinant maximization**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**
Nold, A. *see:* **Resource allocation for epidemic control**
Noor, M.A. *see:* **Variational principles**
Northby, J.A. *see:* **Global optimization in Lennard–Jones and morse clusters**
Norton, R. *see:* **Bilevel programming: introduction, history and overview**; **Bilevel programming in management**
Nožička, F. *see:* **Nondifferentiable optimization: parametric programming**

O

O'Boyle, M. *see:* **Alignment problem**
Oettli, W. *see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Interval linear systems**
Oguz, O. *see:* **Multidimensional knapsack problems**
Okamoto, Y. *see:* **Simulated annealing methods in protein folding**
Okongwu, O.N. *see:* **Dynamic programming: optimal control applications**
O'Neill, M. *see:* **Generalized total least squares**
Orchard-Hays, W. *see:* **Nondifferentiable optimization: parametric programming**
Orden, A. *see:* **Lexicographic pivoting rules**
Orenstein, T. *see:* **Feedback set problems**
Orlin, J.B. *see:* **Greedy randomized adaptive search procedures**; **Maximum flow problem**; **Shortest path tree algorithms**
O'Rourke, J. *see:* **Hyperplane arrangements in optimization**
Osman, I.H. *see:* **Evolutionary algorithms in combinatorial optimization**
Ostresh, L.M. *see:* **Global optimization in Weber's problem with attraction and repulsion**
Ostrogradsky, M. *see:* **Farkas lemma**; **Second order optimality conditions for nonlinear optimization**
Ouveysi, I. *see:* **Survivable networks**
Overton, M.L. *see:* **Global optimization in Weber's problem with attraction and repulsion**; **Successive quadratic programming: decomposition methods**
Owen, G. *see:* **Combinatorial optimization games**
Owens, D.K. *see:* **Resource allocation for epidemic control**

P

Padberg, M.W. *see:* **Assignment and matching**; **Integer programming: branch and bound methods**; **Integer programming: branch and cut algorithms**; **Quadratic assignment problem**
Padmanabhan, V. *see:* **Operations research models for supply chain management and design**
Paixão, J. *see:* **Capacitated minimum spanning trees**
Pallaschke, D. *see:* **Global optimization: envelope representation**
Pallottino, S. *see:* **Shortest path tree algorithms**
Palubeckis, G.S. *see:* **Quadratic assignment problem**
Pamiagua, C.N. *see:* **Chemical process planning**
Panagiotopoulos, P.D. *see:* **Hemivariational inequalities: applications in mechanics**; **Hemivariational inequalities: eigenvalue problems**; **Hemivariational inequalities: static problems**; **Multilevel optimization in mechanics**; **Nonconvex energy functions: hemivariational inequalities**; **Nonconvex-nonsmooth calculus of variations**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: applications**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**; **Solving hemivariational inequalities by nonsmooth optimization methods**
Pang, J.S. *see:* **Nondifferentiable optimization: Newton method**; **Spatial price equilibrium**
Pantelides, C.C. *see:* **MINLP: design and scheduling of batch processes**
Papadimitriou, C.H. *see:* **Quadratic assignment problem**
Papalexandri, K.P. *see:* **MINLP: mass and heat exchanger networks**; **MINLP: reactive distillation column synthesis**
Paparrizos, K. *see:* **Linear programming: Klee–Minty examples**; **Pivoting algorithms for linear programming generating two paths**
Papoulias, S.A. *see:* **Mixed integer linear programming: heat exchanger network synthesis**
Pardalos, P.M. *see:* **Biquadratic assignment problem**; **Complexity theory: quadratic programming**; **Concave programming**; **Feedback set problems**; **Heuristics for maximum clique and independent set**; **Lemke method**; **Piecewise linear network flow problems**; **Preference disaggregation approach: basic features, examples from financial decision making**; **Quadratic fractional programming: Dinkelbach method**; **Quadratic knapsack**

Pareto, V. *see:* **Generalized concavity in multi-objective optimization**; **Infinite horizon control and dynamic games**; **Vector optimization**
Park, K. *see:* **Heuristics for maximum clique and independent set**
Parker, M. *see:* **Multicommodity flow problems**
Passy, U. *see:* **Global optimization in generalized geometric programming**
Paterson, W.R. *see:* **Global optimization of heat exchanger networks**
Patriksson, M. *see:* **Simplicial decomposition**; **Simplicial decomposition algorithms**
Peano, G. *see:* **Image space approach to optimization**
Pearl, J. *see:* **Minimax game tree searching**
Peinado, M. *see:* **Heuristics for maximum clique and independent set**
Peirce, B. *see:* **Single facility location: circle covering problem**
Pelillo, M. *see:* **Heuristics for maximum clique and independent set**; **Replicator dynamics in combinatorial optimization**
Pell, R. *see:* **Portfolio selection and multicriteria analysis**
Peng, J.-M. *see:* **Implicit lagrangian**
Penot, J.P. *see:* **Generalized monotone multivalued maps**
Penrose, R. *see:* **Least squares problems**
Perkins, J.D. *see:* **MINLP: applications in the interaction of design and control**
Perl, J. *see:* **Operations research models for supply chain management and design**
Perny, P. *see:* **Multicriteria sorting methods**
Perrott, R.H. *see:* **Alignment problem**
Peterson, C. *see:* **Neural networks for combinatorial optimization**
Peterson, E.L. *see:* **Spatial price equilibrium**
Peterssen, F. *see:* **MINLP: outer approximation algorithm**; **MINLP: trim-loss problem**
Petrovic, D. *see:* **Operations research models for supply chain management and design**
Pflug, G.Ch. *see:* **Derivatives of probability and integral functions: general theory and examples**
Phillips, A.T. *see:* **Heuristics for maximum clique and independent set**; **Quadratic fractional programming: Dinkelbach method**
Piaget, J. *see:* **Checklist paradigm semantics for fuzzy logics**
Picard, J.C. *see:* **Time-dependent traveling salesman problem**
Pierce, C.S. *see:* **Boolean and fuzzy relations**
Pierskalla, W.P. *see:* **Resource allocation for epidemic control**
Pietrzykowski, T. *see:* **Discontinuous optimization**
Pigou, A.C. *see:* **Traffic network equilibrium**
Pijls, W. *see:* **Minimax game tree searching**
Pincus, M. *see:* **Random search methods**
Pinkava, V. *see:* **Finite complete systems of many-valued logic algebras**
Pinson, E. *see:* **Job-shop scheduling problem**
Pippenger, N. *see:* **Parallel computing: complexity classes**
Pirkul, H. *see:* **Multidimensional knapsack problems**; **Operations research models for supply chain management and design**
Pistikopoulos, E.N. *see:* **Bilevel optimization: feasibility test and flexibility index**; **MINLP: mass and heat exchanger networks**; **MINLP: reactive distillation column synthesis**
Pitsoulis, L.S. *see:* **Biquadratic assignment problem**
Pitts, W. *see:* **Neural networks for combinatorial optimization**
Plambeck, E.L. *see:* **Monte-Carlo simulations for stochastic optimization**
Plastria, F. *see:* **Global optimization in Weber's problem with attraction and repulsion**
Plateau, G. *see:* **Multidimensional knapsack problems**
Platonoff, A. *see:* **Alignment problem**
Pnueli, A. *see:* **Simplicial pivoting algorithms for integer programming**
Pogue, G.A. *see:* **Financial equilibrium**
Pogue, J.A. *see:* **Portfolio selection: markowitz mean-variance model**
Polak, E. *see:* **Feasible sequential quadratic programming**; **Nondifferentiable optimization: minimax problems**; **Quasidifferentiable optimization: algorithms for QD functions**; **Rosen's method, global convergence, and Powell's conjecture**
Pollak, H.O. *see:* **Steiner tree problems**
Pollanski, D. *see:* **Global optimization in Weber's problem with attraction and repulsion**
Polterovich, W.M. *see:* **Generalized monotone multivalued maps**
Polyak, B.T. *see:* **Conjugate-gradient methods**; **Nondifferentiable optimization: relaxation methods**; **Quadratic programming with bound constraints**
Pomeranz, I. *see:* **Feedback set problems**
Popp, W. *see:* **Probabilistic constrained problems: convexity theory**
Posa, L. *see:* **Feedback set problems**
Post, E. *see:* **Finite complete systems of many-valued logic algebras**
Postlethwaite, I. *see:* **Robust control**
Potvin, J.-Y. *see:* **Evolutionary algorithms in combinatorial optimization**
Powell, D.R. *see:* **Generalized total least squares**
Powell, M.J.D. *see:* **Automatic differentiation: calculation of Newton steps**; **Broyden family of methods and the BFGS update**; **History of optimization**; **Lagrangian multipliers methods for convex programming**; **Nonlinear least squares: trust region methods**; **Powell method**; **Rosen's method, global convergence, and Powell's conjecture**; **Successive quadratic programming**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**; **Successive quadratic programming: full space methods**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**
Powell, W.B. *see:* **Frank–Wolfe algorithm**
Prager, W. *see:* **Interval linear systems**
Prais, M. *see:* **Feedback set problems**; **Greedy randomized adaptive search procedures**
Prékopa, A. *see:* **Approximation of multivariate probability integrals**; **Derivatives of probability and integral functions: general theory and examples**; **History of optimization**; **Probabilistic constrained linear programming: duality theory**; **Simple recourse problem: dual method**
Press, W.H. *see:* **Global optimization in binary star astronomy**
Prim, R.C. *see:* **Matroids**
Protasi, M. *see:* **Heuristics for maximum clique and independent set**
Provan, J.S. *see:* **Complexity of degeneracy**

Pryce, J.D. *see:* **Minimax theorems**
Pshenichnyi, B.N. *see:* **Farkas lemma: generalizations**
Pu, D. *see:* **Rosen's method, global convergence, and Powell's conjecture**
Pugh, W. *see:* **Alignment problem**
Putnam, H. *see:* **Maximum satisfiability problem**

Q

Qi, L. *see:* **Feasible sequential quadratic programming**; **Nondifferentiable optimization: Newton method**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Stochastic programming: parallel factorization of structured matrices**
Qian, T. *see:* **Feedback set problems**
Quandt, R.E. *see:* **Nonlinear least squares: trust region methods**
Quang, P.H. *see:* **Generalized monotone multivalued maps**
Quesada, I. *see:* **Generalized outer approximation**
Queyranne, M. *see:* **Quadratic assignment problem**; **Time-dependent traveling salesman problem**
Quinlan, J.R. *see:* **Optimization in boolean classification problems**

R

Rabinowitz, P. *see:* **Interval analysis: systems of nonlinear equations**
Rademacher, H. *see:* **Interval global optimization**
Radzik, T. *see:* **Fractional programming**
Raghavan, P. *see:* **Multicommodity flow problems**
Raiffa, H. *see:* **Preference disaggregation approach: basic features, examples from financial decision making**
Raik, E. *see:* **Approximation of extremum problems with probability functionals**; **Derivatives of probability and integral functions: general theory and examples**
Rall, L.B. *see:* **Automatic differentiation: calculation of the Hessian**; **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: point and interval**; **Bounding derivative ranges**
Ramaiyer, V. *see:* **Steiner tree problems**
Ramana, M. *see:* **Duality for semidefinite programming**
Ramana, M.V. *see:* **Semi-infinite programming, semidefinite programming and perfect duality**
Ramsey, F. *see:* **Infinite horizon control and dynamic games**
Rangan, C.P. *see:* **Feedback set problems**
Ranjan, R. *see:* **Operations research models for supply chain management and design**
Rantzer, A. *see:* **Robust control**
Rao, M.R. *see:* **Assignment and matching**; **Fractional programming**
Rao, S.K. *see:* **Steiner tree problems**
Rao, V.N. *see:* **Load balancing for parallel optimization techniques**
Ratschek, H. *see:* **Bounding derivative ranges**
Ravindran, A. *see:* **Lemke method**; **Multi-objective optimization; Interactive methods for preference value functions**
Rech, P. *see:* **Piecewise linear network flow problems**
Rechenberg, I. *see:* **Genetic algorithms**
Reece, K.E. *see:* **Operations research models for supply chain management and design**
Reese, S. *see:* **Automatic differentiation: calculation of Newton steps**
Reeves, C. *see:* **Conjugate-gradient methods**; **Evolutionary algorithms in combinatorial optimization**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**
Reid, J.R. *see:* **Automatic differentiation: calculation of Newton steps**
Reinelt, G. *see:* **Feedback set problems**
Reinert, K. *see:* **Optimization in leveled graphs**
Renaud, J.E. *see:* **Multidisciplinary design optimization**
Rendl, F. *see:* **Quadratic assignment problem**; **Quadratic programming over an ellipsoid**; **Replicator dynamics in combinatorial optimization**
Renegar, J. *see:* **Complexity theory: quadratic programming**; **Linear programming: interior point methods**
Rescher, N. *see:* **Boolean and fuzzy relations**
Resende, M.G.C. *see:* **Biquadratic assignment problem**; **Feedback set problems**; **Graph planarization**
ReVelle, C. *see:* **Resource allocation for epidemic control**
Rhee, W.T. *see:* **Quadratic assignment problem**
Rhodes, E. *see:* **Fractional programming**
Ribeiro, C.C. *see:* **Feedback set problems**; **Graph planarization**; **Greedy randomized adaptive search procedures**
Ricci, N. *see:* **Portfolio selection and multicriteria analysis**
Rice, J.R. *see:* **Optimization software**
Richter, A. *see:* **Resource allocation for epidemic control**
Riguet, J. *see:* **Boolean and fuzzy relations**
Rijal, M.P. *see:* **Quadratic assignment problem**
Rinnooy Kan, A.G. *see:* **Quadratic assignment problem**
Rios-Garcia, A. *see:* **Portfolio selection and multicriteria analysis**
Rios-Insua, S. *see:* **Portfolio selection and multicriteria analysis**
Ritter, K. *see:* **Adaptive global search**; **Information-based complexity and information-based optimization**; **Rosen's method, global convergence, and Powell's conjecture**
Rivest, R.L. *see:* **Minimax game tree searching**
Robbins, H. *see:* **History of optimization**; **Steiner tree problems**
Robert, Y. *see:* **Alignment problem**
Roberts, F.D.K. *see:* **Overdetermined systems of linear equations**
Roberts, F.S. *see:* **Frequency assignment problem**
Roberts, S.M. *see:* **Chemical process planning**
Robin, G. *see:* **Steiner tree problems**
Robinson, S.M. *see:* **Nondifferentiable optimization: Newton method**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Sensitivity analysis of complementarity problems**; **Sensitivity analysis of variational inequality problems**; **Sensitivity and stability in NLP: continuity and differential stability**
Rockafellar, R.T. *see:* **Composite nonsmooth optimization**; **Duality in optimal control with first order differential equations**; **Lagrangian multipliers methods for convex programming**; **Multilevel optimization in mechanics**; **Nonconvex energy functions: hemivariational inequalities**; **Quasidifferentiable optimization**; **Splitting method for linear complementarity problems**; **Stochastic programming: nonanticipativity and lagrange multipliers**
Rodríguez, J.F. *see:* **Multidisciplinary design optimization**

Rogers, P. *see:* **Portfolio selection and multicriteria analysis**
Rohn, J. *see:* **Interval linear systems**
Rokne, J. *see:* **Bounding derivative ranges**
Rolewicz, S. *see:* **Global optimization: envelope representation**
Romeijn, H.E. *see:* **Global optimization: hit and run methods**; **Operations research models for supply chain management and design**; **Random search methods**
Romero Morales, D. *see:* **Operations research models for supply chain management and design**
Rommelfanger, H. *see:* **Fuzzy multi-objective linear programming**
Roos, C. *see:* **Criss-cross pivoting rules**; **Linear programming: interior point methods**; **Linear programming: Klee–Minty examples**
Rosen, J.B. *see:* **Convex envelopes in optimization problems**; **Global optimization in Weber's problem with attraction and repulsion**; **History of optimization**; **LCP: Pardalos–Rosen mixed integer formulation**; **Lemke method**; **Multicommodity flow problems**; **Oligopolistic market equilibrium**; **Quadratic fractional programming: Dinkelbach method**; **Reverse convex optimization**; **Rosen's method, global convergence, and Powell's conjecture**
Rosenberg, I. *see:* **Finite complete systems of many-valued logic algebras**
Rosenbrock, H.H. *see:* **Direct search Luus—Jaakola optimization procedure**; **Robust control**; **Rosenbrock method**
Ross, S.A. *see:* **Financial equilibrium**
Rota, G.-C. *see:* **Metropolis, Nicholas Constantine**
Roth, R.M. *see:* **Feedback set problems**
Rothberg, E. *see:* **Financial optimization**
Roubens, M. *see:* **Boolean and fuzzy relations**
Roucairol, C. *see:* **Minimax game tree searching**
Roundy, R. *see:* **Inventory management in supply chains**
Rousseau, J.M. *see:* **Multicommodity flow problems**
Roy, A.D. *see:* **Portfolio selection: markowitz mean-variance model**
Roy, B. *see:* **Decision support systems with multiple criteria**; **Financial applications of multicriteria analysis**; **Multi-objective mixed integer programming**; **Multiple objective programming support**; **Outranking methods**; **Preference disaggregation**
Ruan, L. *see:* **Steiner tree problems**
Rubin, P.A. *see:* **Multicriteria sorting methods**
Rubinov, A.M. *see:* **Global optimization: envelope representation**; **Quasidifferentiable optimization**; **Quasidifferentiable optimization: algorithms for hypodifferentiable functions**; **Quasidifferentiable optimization: applications to thermoelasticity**; **Quasidifferentiable optimization: codifferentiable functions**; **Quasidifferentiable optimization: Dini derivatives, clarke derivatives**; **Quasidifferentiable optimization: stability of dynamic systems**; **Quasidifferentiable optimization: variational formulations**; **Quasivariational inequalities**
Rubinstein, M. *see:* **Financial equilibrium**
Rudd, D.F. *see:* **Chemical process planning**
Rudolph, H. *see:* **Semi-infinite programming and control problems**
Ruefli, T. *see:* **Bilevel programming in management**
Rump, S.M. *see:* **Interval linear systems**
Russell, B. *see:* **Boolean and fuzzy relations**
Rust, R. *see:* **Piecewise linear network flow problems**
Ruszczynski, A. *see:* **Financial optimization**
Rutenberg, D.P. *see:* **Convex-simplex algorithm**
Ryan, J. *see:* **Multicommodity flow problems**
Rydergren, C. *see:* **Simplicial decomposition**; **Simplicial decomposition algorithms**

S

Saaty, T.L. *see:* **Estimating data for multicriteria decision making problems: optimization techniques**; **Multiple objective programming support**; **Portfolio selection and multicriteria analysis**; **Probabilistic analysis of simplex algorithms**
Sadayappan, P. *see:* **Alignment problem**; **Steiner tree problems**
Safonov, M.G. *see:* **Robust control**
Sahinidis, N.V. *see:* **Chemical process planning**; **Time-dependent traveling salesman problem**
Sahni, S. *see:* **Complexity theory: quadratic programming**; **Quadratic assignment problem**; **Quadratic knapsack**
Salomaa, A. *see:* **Finite complete systems of many-valued logic algebras**
Salton, G. *see:* **Optimization in classifying text documents**
Sameh, A. *see:* **Relaxation in projection methods**
Samelson, H. *see:* **Linear complementarity problem**
Samuelson, P.A. *see:* **Equilibrium networks**; **Spatial price equilibrium**; **Traffic network equilibrium**
Sanchez, E. *see:* **Boolean and fuzzy relations**
Sandberg, I.W. *see:* **Robust control**
Sano, M. *see:* **Portfolio selection and multicriteria analysis**
Santiago, M. *see:* **Chemical process planning**
Santibez, J. *see:* **Chemical process planning**
Sargent, R.W.H. *see:* **Rosen's method, global convergence, and Powell's conjecture**; **Successive quadratic programming: applications in the process industry**
Saunder, M.A. *see:* **Convex-simplex algorithm**
Saviozzi, G. *see:* **Multicommodity flow problems**
Sawaragi, Y. *see:* **Multi-objective optimization: lagrange duality**
Scanella, G. *see:* **Preference disaggregation**
Scarf, H. *see:* **Operations research models for supply chain management and design**; **Walrasian price equilibrium**
Scarf, H.E. *see:* **Integer programming: algebraic methods**; **Inventory management in supply chains**
Schaible, S. *see:* **Fractional programming**
Schell, E. *see:* **Multi-index transportation problems**
Scheraga, H.A. *see:* **Simulated annealing methods in protein folding**
Schieber, B. *see:* **Feedback set problems**
Schilling, K.E. *see:* **Multidimensional knapsack problems**
Schittkowski, K. *see:* **Successive quadratic programming: applications in the process industry**
Schmid, C. *see:* **Successive quadratic programming: applications in distillation systems**
Schmidt, G. *see:* **Boolean and fuzzy relations**
Schmit, L. *see:* **Structural optimization: history**
Schnabel, R.B. *see:* **Generalized total least squares**; **Nonlinear least squares: trust region methods**

Schnepper, C.A. *see:* **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**
Schneur, R. *see:* **Multicommodity flow problems**
Schnitger, G. *see:* **Concave programming**
Schoeffler, J.D. *see:* **Multilevel optimization in mechanics**
Schoenberg, I.J. *see:* **Fejér monotonicity in convex optimization**; **Matrix completion problems**
Scholes, M. *see:* **Financial equilibrium**
Schräffer, A.A. *see:* **Quadratic assignment problem**
Schrijver, A. *see:* **Disjunctive programming**; **Integer programming**
Schröder, E. *see:* **Boolean and fuzzy relations**
Schultz, G.L. *see:* **Multicommodity flow problems**
Schumacher, H.C. *see:* **History of optimization**
Schwarz, H. *see:* **Carathéodory, Constantine**
Schwarz, L.B. *see:* **Inventory management in supply chains**
Schweitzer, B. *see:* **Checklist paradigm semantics for fuzzy logics**
Scott, A.J. *see:* **MINLP: application in facility location-allocation**
Scott, C.H. *see:* **Fractional programming**
Scudder, G.D. *see:* **Multidimensional knapsack problems**
Sculli, D. *see:* **Inventory management in supply chains**
Scutellá, M.G. *see:* **Shortest path tree algorithms**
Segall, R.S. *see:* **Multicommodity flow problems**
Seidel, R. *see:* **Hyperplane arrangements in optimization**
Seider, W.D. *see:* **Global optimization: application to phase equilibrium problems**
Selten, R. *see:* **Infinite horizon control and dynamic games**
Sen, S. *see:* **Disjunctive programming**; **Monte-Carlo simulations for stochastic optimization**
Sengupta, J.K. *see:* **Robust optimization**
Sengupta, P. *see:* **Lemke method**
Sengupta, S. *see:* **Interval global optimization**
Senju, S. *see:* **Multidimensional knapsack problems**
Sethi, S.P. *see:* **Resource allocation for epidemic control**
Seymour, P.D. *see:* **Feedback set problems**
Shah, B.V. *see:* **Conjugate-gradient methods**
Shahrokhi, F. *see:* **Optimization in leveled graphs**
Shallcross, D.F. *see:* **Survivable networks**
Shalloway, D. *see:* **Global optimization in Lennard–Jones and morse clusters**; **Simulated annealing methods in protein folding**
Shamir, A. *see:* **Feedback set problems**
Shamir, R. *see:* **Complexity theory: quadratic programming**; **Probabilistic analysis of simplex algorithms**
Shanno, D.F. *see:* **Broyden family of methods and the BFGS update**; **Linear programming: interior point methods**
Shannon, C.E. *see:* **Entropy optimization: parameter estimation**; **Entropy optimization: shannon measure of entropy and its properties**; **Jaynes' maximum entropy principle**; **Maximum flow problem**
Shanthikumar, J.G. *see:* **Combinatorial optimization algorithms in resource allocation problems**
Shapiro, A. *see:* **Operations research and financial markets**
Shapley, L.S. *see:* **Combinatorial optimization games**
Sharpe, W.F. *see:* **Financial equilibrium**; **Operations research and financial markets**
Shary, S.P. *see:* **Interval linear systems**
Shectman, J.P. *see:* **Chemical process planning**
Sheffi, Y. *see:* **Frank–Wolfe algorithm**
Shelton, J.P. *see:* **Operations research models for supply chain management and design**
Shen, Z. *see:* **Interval global optimization**
Sherali, H.D. *see:* **Bilevel programming in management**; **Disjunctive programming**; **MINLP: application in facility location-allocation**; **Quadratic assignment problem**; **Time-dependent traveling salesman problem**
Sherbrooke, C.C. *see:* **Inventory management in supply chains**
Shetty, C.M. *see:* **Disjunctive programming**; **MINLP: application in facility location-allocation**
Sheu, J.-P. *see:* **Alignment problem**
Shi, J. *see:* **Survivable networks**
Shi, W. *see:* **Steiner tree problems**
Shih, W. *see:* **Multidimensional knapsack problems**
Shimizu, K. *see:* **Bilevel linear programming**; **Bilevel programming in management**
Shimizu, Y. *see:* **Chemical process planning**
Shin, W.S. *see:* **Multi-objective optimization; Interactive methods for preference value functions**
Shindo, S. *see:* **Nondifferentiable optimization: Newton method**
Shirakawa, I. *see:* **Graph planarization**
Shmulevich, I. *see:* **Inference of monotone boolean functions**
Shogan, A.W. *see:* **Greedy randomized adaptive search procedures**
Shoker, A.D. *see:* **Preference disaggregation**
Shor, N.Z. *see:* **Nondifferentiable optimization**; **Nondifferentiable optimization: subgradient optimization methods**
Shor, P.W. *see:* **Steiner tree problems**
Shoven, J.B. *see:* **Walrasian price equilibrium**
Shubik, M. *see:* **Combinatorial optimization games**
Shukla, V. *see:* **Design optimization in computational fluid dynamics**
Shulman, A. *see:* **Multicommodity flow problems**
Shultz, G.A. *see:* **Nonlinear least squares: trust region methods**
Shuzhong Zhang *see:* **Semi-infinite programming, semidefinite programming and perfect duality**
Shvatal, V. *see:* **Linear programming: Klee–Minty examples**
Sikorski, K. *see:* **Information-based complexity and information-based optimization**
Simchi-Levi, D. *see:* **Operations research models for supply chain management and design**; **Set covering, packing and partitioning problems**
Simon, J.D. *see:* **Successive quadratic programming: applications in the process industry**
Simons, S. *see:* **Minimax theorems**
Sinclair, I.G. *see:* **Generalized total least squares**
Singer, I. *see:* **Global optimization: envelope representation**
Sinharoy, B. *see:* **Alignment problem**
Sion, M. *see:* **Minimax theorems**
Siskos, J. *see:* **Preference disaggregation**
Siskos, Y. *see:* **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**
Sit, W. *see:* **Linear programming: Klee–Minty examples**
Skelboe, S. *see:* **Interval global optimization**
Sklar, A. *see:* **Checklist paradigm semantics for fuzzy logics**
Skogestad, S. *see:* **Robust control**
Skrifvars, H. *see:* **MINLP: trim-loss problem**

Slater, M.L. *see:* **Theorems of the alternative and optimization**
Sloan, I. *see:* **Information-based complexity and information-based optimization**
Slowinski, R. *see:* **Fuzzy multi-objective linear programming**; **Preference disaggregation**
Slupecki, J. *see:* **Finite complete systems of many-valued logic algebras**
Smale, S. *see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Probabilistic analysis of simplex algorithms**; **Topology of global optimization**
Smith, C. *see:* **Multicriteria sorting methods**
Smith, C.A.B. *see:* **Statistical classification: optimization approaches**
Smith, G.W. *see:* **Feedback set problems**
Smith, J. *see:* **Generalized total least squares**
Smith, J.M. *see:* **Steiner tree problems**
Smith, M. *see:* **Variational inequalities: projected dynamical system**
Smith, M.J. *see:* **Dynamic traffic networks**; **Equilibrium networks**; **Traffic network equilibrium**; **Variational inequalities**
Smith, R. *see:* **Interval linear systems**
Smith, R.L. *see:* **Global optimization: hit and run methods**; **Random search methods**
Smith, R.R. *see:* **Interval global optimization**
Smith, R.T. *see:* **Maximum entropy principle: image reconstruction**
Sniedovich, M. *see:* **Fractional programming**
Sofer, A. *see:* **Numerical methods for unary optimization**
Soffa, M. *see:* **Feedback set problems**
Sokolov, N.A. *see:* **Inference of monotone boolean functions**
Soland, R.M. *see:* **Minimum concave transportation problems**; **MINLP: application in facility location-allocation**; **Multi-objective mixed integer programming**
Solanki, R. *see:* **Multi-objective mixed integer programming**
Solis, F.J. *see:* **Random search methods**
Solodov, M.V. *see:* **Implicit lagrangian**
Solomon, M. *see:* **Minimax game tree searching**
Solomonoff, R. *see:* **Kolmogorov complexity**
Solow, D. *see:* **Lemke method**
Sorensen, D.C. *see:* **Nonlinear least squares: trust region methods**
Sorenson, D.C. *see:* **Quadratic programming over an ellipsoid**
Soriano, P. *see:* **Heuristics for maximum clique and independent set**
Sosic Rok *see:* **Modeling languages in optimization: a new paradigm**
Soule, T. *see:* **Heuristics for maximum clique and independent set**
Southwell, W.H. *see:* **Generalized total least squares**
Sparrow, F.T. *see:* **Dynamic traffic networks**; **Traffic network equilibrium**
Speckenmeyer, E. *see:* **Feedback set problems**; **Minimax game tree searching**
Spedicato, E. *see:* **ABS algorithms for linear equations and linear least squares**
Speelpenning, B. *see:* **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: point and interval**
Spendley, W. *see:* **Sequential simplex method**
Spiliopoulos, P. *see:* **Preference disaggregation**
Spingarn, K. *see:* **Estimating data for multicriteria decision making problems: optimization techniques**
Sprecher, D. *see:* **Hilbert's thirteenth problem**
Sridhar, L.N. *see:* **Global optimization: application to phase equilibrium problems**
Srinivas, B.K. *see:* **MINLP: mass and heat exchanger networks**; **Mixed integer linear programming: mass and heat exchanger networks**
Srinivasan, V. *see:* **Preference disaggregation**
Srivastava, R. *see:* **Operations research models for supply chain management and design**
Staats, P.W. *see:* **Resource allocation for epidemic control**
Stachó, L.L. *see:* **Minimax theorems**
Stackelberg, H. *see:* **Bilevel programming: introduction, history and overview**; **Multilevel methods for optimal design**
Stadtherr, M.A. *see:* **Global optimization: application to phase equilibrium problems**; **Interval analysis: application to chemical engineering design problems**; **Successive quadratic programming: applications in the process industry**; **Successive quadratic programming: decomposition methods**
Stam, A. *see:* **Multicriteria sorting methods**; **Statistical classification: optimization approaches**
Stamm, H. *see:* **Feedback set problems**
Stampacchia, G. *see:* **Generalized monotonicity: applications to variational inequalities and equilibrium problems**; **Variational inequalities**
Stancu-Minasian, I.M. *see:* **Fractional programming**
Stear, E.B. *see:* **Entropy optimization: parameter estimation**
Stearns, R.E. *see:* **Complexity theory**
Steele Jr., G.L. *see:* **Alignment problem**
Steiglitz, K. *see:* **Survivable networks**
Steihaug, T. *see:* **Nonlinear least squares: trust region methods**
Stein, G. *see:* **Robust control**
Steinberg, I.R. *see:* **Minimax game tree searching**
Steinberg, R. *see:* **Traffic network equilibrium**
Steiner, J. *see:* **History of optimization**
Sterman, J.D. *see:* **Operations research models for supply chain management and design**
Steuer, R.E. *see:* **Multi-objective mixed integer programming**; **Multi-objective optimization and decision support systems**; **Multi-objective optimization; Interactive methods for preference value functions**
Stewart, G.W. *see:* **Least squares problems**
Stewart, T.J. *see:* **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**
Stiefel, E.L. *see:* **Conjugate-gradient methods**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**
Stiemke, E. *see:* **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**
Stockman, G.C. *see:* **Minimax game tree searching**
Stockmeyer, L.J. *see:* **Complexity classes in optimization**
Stoer, M. *see:* **Survivable networks**
Storey, C. *see:* **Dynamic programming: optimal control applications**
Stougie, L. *see:* **Stochastic integer programming: continuity, stability, rates of convergence**

Strassen, V. *see:* **Automatic differentiation: introduction, history and rounding error estimation**
Straub, J. *see:* **Simulated annealing methods in protein folding**
Straus, E.G. *see:* **Complexity theory: quadratic programming**; **Heuristics for maximum clique and independent set**; **Replicator dynamics in combinatorial optimization**; **Standard quadratic optimization problems: applications**
Strömberg, A.-B. *see:* **Simplicial decomposition algorithms**
Strum, J.F. *see:* **Semi-infinite programming, semidefinite programming and perfect duality**
Stuart, C.A. *see:* **Quasidifferentiable optimization: variational formulations**
Sturm, Jos.F. *see:* **Semi-infinite programming, semidefinite programming and perfect duality**
Su, C. *see:* **Steiner tree problems**
Sudborough, I.H. *see:* **Survivable networks**
Sun, A.C. *see:* **Global optimization: application to phase equilibrium problems**
Sun, D. *see:* **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**
Sun, J. *see:* **Nondifferentiable optimization: Newton method**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**
Sun, S. *see:* **Single facility location: circle covering problem**
Sun, T. *see:* **Evolutionary algorithms in combinatorial optimization**
Sung, Y. *see:* **Reverse convex optimization**
Sutcliffe, C.M.S. *see:* **Portfolio selection: markowitz mean-variance model**
Sutton, R.S. *see:* **Neuro-dynamic programming**
Swaney, R.E. *see:* **Interval analysis: application to chemical engineering design problems**
Swarup, K. *see:* **Global optimization in multiplicative programming**; **Multiplicative programming**
Sylvester, J.J. *see:* **Single facility location: circle covering problem**; **Symmetric systems of linear equations**
Szala, A. *see:* **Portfolio selection and multicriteria analysis**
Szanc, B.P. *see:* **Generalized nonlinear complementarity problem**
Szántai, T. *see:* **Approximation of multivariate probability integrals**
Szkatula, K. *see:* **Multidimensional knapsack problems**
Szpankowski, W. *see:* **Quadratic assignment problem**
Szu, H. *see:* **Simulated annealing methods in protein folding**
Szulkin, A. *see:* **Nonconvex-nonsmooth calculus of variations**
Szymanski, B.K. *see:* **Alignment problem**

T

Taji, K. *see:* **Implicit lagrangian**
Takács, L. *see:* **Approximation of multivariate probability integrals**
Takahashi, T. *see:* **Nonlinear least squares: Newton-type methods**
Takamatsu, T. *see:* **Chemical process planning**
Takane, Y. *see:* **Preference disaggregation**
Takayama, A. *see:* **History of optimization**
Takayama, T. *see:* **Spatial price equilibrium**
Takefuji, Y. *see:* **Graph planarization**
Takeguchi, T. *see:* **Portfolio selection and multicriteria analysis**
Takvorian, A. *see:* **Graph planarization**
Talman, A.J.J. *see:* **Walrasian price equilibrium**
Tamassia, R. *see:* **Graph planarization**
Tamir, A. *see:* **Combinatorial optimization games**; **Network location: covering problems**
Tamiz, M. *see:* **Portfolio selection and multicriteria analysis**
Tan, W.Y. *see:* **Resource allocation for epidemic control**
Tang, C.Y. *see:* **Feedback set problems**
Tangian, A. *see:* **Preference disaggregation**
Tanino, T. *see:* **Multi-objective optimization: lagrange duality**
Tank, D.W. *see:* **Heuristics for maximum clique and independent set**; **Neural networks for combinatorial optimization**
Tansel, B.C. *see:* **Network location: covering problems**
Tardos, É. *see:* **Complexity theory: quadratic programming**; **Information-based complexity and information-based optimization**
Tarjan, R.E. *see:* **Graph planarization**; **Steiner tree problems**
Tarjan, R.J. *see:* **Maximum flow problem**
Tarski, A. *see:* **Boolean and fuzzy relations**
Tassone, V. *see:* **Dynamic programming: optimal control applications**
Tecchiolli, G. *see:* **Multidimensional knapsack problems**
Tedenat, P. *see:* **History of optimization**
Tellier, L.-N. *see:* **Global optimization in Weber's problem with attraction and repulsion**
Temam, R. *see:* **Duality in optimal control with first order differential equations**
Terkelsen, F. *see:* **Minimax theorems**
Terlaky, T. *see:* **Criss-cross pivoting rules**
Ternet, D. *see:* **Successive quadratic programming: applications in the process industry**
Terzopoulos, D. *see:* **Optimization in medical imaging**
Tesauro, G. *see:* **Neuro-dynamic programming**
Tessier, S.R. *see:* **Global optimization: application to phase equilibrium problems**
Thach, P.T. *see:* **Piecewise linear network flow problems**
Thagard, P. *see:* **Forecasting**
Thangiah, S.R. *see:* **Evolutionary algorithms in combinatorial optimization**
Theodorescu, R. *see:* **Stochastic programming: minimax approach**
Thom, A. *see:* **Design optimization in computational fluid dynamics**
Thompson, C.D. *see:* **Multicommodity flow problems**
Thrall, R.M. *see:* **Linear complementarity problem**
Tikhonov, A.N. *see:* **Cost approximation algorithms**
Tits, A.L. *see:* **Robust control**
Tiwari, A. *see:* **Greedy randomized adaptive search procedures**
Todd, M.J. *see:* **Homogeneous selfdual methods for linear programming**; **Lexicographic pivoting rules**; **Linear programming: interior point methods**; **Principal pivoting methods for linear complementarity problems**; **Probabilistic analysis of simplex algorithms**; **Walrasian price equilibrium**
Toi, T.-H. *see:* **Alignment problem**
Toland, J.F. *see:* **Quasidifferentiable optimization: variational formulations**
Tolle, J.W. *see:* **Nonlinear least squares problems**
Tollis, I.G. *see:* **Survivable networks**
Tonti, E. *see:* **Variational principles**

Torczon, V. *see:* **Global optimization in binary star astronomy**
Toricelli, E. *see:* **Global optimization in Weber's problem with attraction and repulsion**
Torricelli, E. *see:* **History of optimization**
Toth, P. *see:* **Capacitated minimum spanning trees**; **Multidimensional knapsack problems**
Towill, D.R. *see:* **Operations research models for supply chain management and design**
Townsely, R. *see:* **Bilevel linear programming**
Traub, J.F. *see:* **Information-based complexity and information-based optimization**
Traverso, C. *see:* **Integer programming: algebraic methods**
Trefftz, E. *see:* **Variational principles**
Tretyakov, G.L. *see:* **Derivatives of probability and integral functions: general theory and examples**
Trevisan, L. *see:* **Maximum satisfiability problem**
Triantaphyllou, E. *see:* **Optimization in boolean classification problems**
Tribus, M. *see:* **Jaynes' maximum entropy principle**
Trost, R. *see:* **Minimax theorems**
Trotter, H.F. *see:* **Nonlinear least squares: trust region methods**
Tschirnhausen, W. *see:* **Hilbert's thirteenth problem**
Tse, E. *see:* **Complexity theory: quadratic programming**
Tseng, P. *see:* **Implicit lagrangian**
Tsiang, S. *see:* **Portfolio selection: markowitz mean-variance model**
Tsoo, C. *see:* **Simulated annealing methods in protein folding**
Tucker, A.W. *see:* **Criss-cross pivoting rules**; **Equality-constrained nonlinear programming: KKT necessary optimality conditions**; **Farkas lemma**; **First order constraint qualifications**; **History of optimization**; **Homogeneous selfdual methods for linear programming**; **Least-index anticycling rules**; **Principal pivoting methods for linear complementarity problems**; **Second order optimality conditions for nonlinear optimization**; **Theorems of the alternative and optimization**; **Tucker homogeneous systems of linear relations**
Tuncbilek, C.H. *see:* **MINLP: application in facility location-allocation**
Turing, A.M. *see:* **Complexity classes in optimization**
Turkay, M. *see:* **MINLP: logic-based methods**
Türksen, I.B. *see:* **Checklist paradigm semantics for fuzzy logics**
Turner, A.L. *see:* **Portfolio selection: markowitz mean-variance model**
Tutte, W. *see:* **Matroids**; **Oriented matroids**
Tuy, H. *see:* **Global optimization in Weber's problem with attraction and repulsion**; **Minimax theorems**; **Reverse convex optimization**
Twilt, F. *see:* **Parametric global optimization: sensitivity**; **Parametric optimization: embeddings, path following and singularities**
Tzafestas, S. *see:* **Operations research models for supply chain management and design**

U

Ueing, U. *see:* **Reverse convex optimization**
Ulam, S. *see:* **Metropolis, Nicholas Constantine**
Ulungu, E.L. *see:* **Multi-objective combinatorial optimization**
Umeda, T. *see:* **MINLP: heat exchanger network synthesis**; **Mixed integer linear programming: heat exchanger network synthesis**
Urich, T.J. *see:* **Portfolio selection: markowitz mean-variance model**
Uryasev, S. *see:* **Derivatives of probability and integral functions: general theory and examples**
Utke, J. *see:* **Automatic differentiation: calculation of Newton steps**
Uzawa, H. *see:* **Multilevel optimization in mechanics**

V

Vaidya, P.M. *see:* **Complexity theory: quadratic programming**; **Linear programming: interior point methods**; **Multicommodity flow problems**; **Nondifferentiable optimization: cutting plane methods**
Vaidyanathan, R. *see:* **Interval analysis: application to chemical engineering design problems**
Valiant, L.G. *see:* **Parallel computing: complexity classes**
van de Panne, C. *see:* **Probabilistic constrained problems: convexity theory**
van der Heijden, M.C. *see:* **Inventory management in supply chains**
van der Laan, G. *see:* **Walrasian price equilibrium**
van Houweninge, M. *see:* **Quadratic assignment problem**
Van Slyke, R.M. *see:* **Monte-Carlo simulations for stochastic optimization**; **Multicommodity flow problems**
Vance, P.H. *see:* **Multicommodity flow problems**
Vander Wiel, R.J. *see:* **Time-dependent traveling salesman problem**
Vanderbei, R.J. *see:* **Complexity of degeneracy**; **Financial optimization**; **Linear programming: interior point methods**
Vargas, L.G. *see:* **Estimating data for multicriteria decision making problems: optimization techniques**
Vasicek, O. *see:* **Semi-infinite programming and applications in finance**
Vavasis, S.A. *see:* **Complexity theory**; **Complexity theory: quadratic programming**; **Information-based complexity and information-based optimization**; **Quadratic knapsack**; **Quadratic programming over an ellipsoid**
Vaz, R.F. *see:* **Quadratic assignment problem**
Vecchi, M.P. *see:* **Heuristics for maximum clique and independent set**; **Quadratic assignment problem**; **Simulated annealing**
Veinott, A.F. *see:* **Inventory management in supply chains**; **Simplicial pivoting algorithms for integer programming**
Ventura, J.A. *see:* **Simplicial decomposition**
Vial, J.-Ph. *see:* **Linear programming: interior point methods**; **Nonlinear least squares: trust region methods**
Vigo, D. *see:* **Capacitated minimum spanning trees**
Villarreal, B. *see:* **Multi-objective mixed integer programming**
Ville, J. *see:* **Tucker homogeneous systems of linear relations**
Vincke, P. *see:* **Boolean and fuzzy relations**
Virball, V.G. *see:* **Quadratic assignment problem**
Viswanathan, J. *see:* **MINLP: outer approximation algorithm**
Visweswaran, V. *see:* **MINLP: applications in blending and pooling problems**
Vitanyi, P. *see:* **Kolmogorov complexity**
Vitushkin, A. *see:* **Hilbert's thirteenth problem**

Vladimirou, H. *see:* **Financial optimization**
Volgenant, A. *see:* **Multidimensional knapsack problems**
von Hohenbalken, B. *see:* **Simplicial decomposition**; **Simplicial decomposition algorithms**
von Neumann, J. *see:* **Decision support systems with multiple criteria**; **Entropy optimization: shannon measure of entropy and its properties**; **Fractional programming**; **Metropolis, Nicholas Constantine**; **Minimax theorems**; **Saddle point theory and optimality conditions**; **Tucker homogeneous systems of linear relations**
von Weizäcker, C.C. *see:* **Infinite horizon control and dynamic games**
Vopěnka, P. *see:* **Alternative set theory**
Voß, S. *see:* **Multidimensional knapsack problems**; **Time-dependent traveling salesman problem**

W

Wagner, H. *see:* **Static stochastic programming models**
Wagner, H.M. *see:* **Inventory management in supply chains**; **Preference disaggregation**; **Probabilistic constrained linear programming: duality theory**
Wald, A. *see:* **Financial equilibrium**; **Static stochastic programming models**
Walford, R.B. *see:* **Feedback set problems**
Wall, J. *see:* **Robust control**
Wallace, S.W. *see:* **Stochastic programs with recourse: upper bounds**
Walras, L. *see:* **Financial equilibrium**; **Walrasian price equilibrium**
Walras, M.-E.-L. *see:* **Vector optimization**
Walsh, S. *see:* **MINLP: applications in the interaction of design and control**
Walster, G.W. *see:* **Interval global optimization**
Wan, P.-J. *see:* **Steiner tree problems**
Wang, C. *see:* **Feedback set problems**
Wang, C.Y. *see:* **Global optimization in Weber's problem with attraction and repulsion**
Wang, D.I. *see:* **Frequency assignment problem**
Wang, S. *see:* **Numerical methods for unary optimization**
Wang, Zh. *see:* **Criss-cross pivoting rules**
Warburton, A.R. *see:* **Generalized concavity in multi-objective optimization**
Wardrop, J.G. *see:* **Dynamic traffic networks**; **Equilibrium networks**; **Traffic network equilibrium**
Wasilkowski, G.W. *see:* **Information-based complexity and information-based optimization**
Wasylkiewicz, S.K. *see:* **Global optimization: application to phase equilibrium problems**
Watanabe, T. *see:* **Linear ordering problem**
Waterman, M.S. *see:* **Optimization in leveled graphs**
Watkins, C.J.C.H. *see:* **Neuro-dynamic programming**
Watson, L.T. *see:* **Multidisciplinary design optimization**
Waugh, F.R. *see:* **Replicator dynamics in combinatorial optimization**
Webb, I.R. *see:* **Operations research models for supply chain management and design**
Wedley, W.C. *see:* **Multicriteria sorting methods**
Wei, Z. *see:* **Feasible sequential quadratic programming**
Weidemann, H.L. *see:* **Entropy optimization: parameter estimation**
Weierstrass, K. *see:* **Smooth nonlinear nonconvex optimization**; **Variational principles**
Wein, L.M. *see:* **Resource allocation for epidemic control**
Weiner, P. *see:* **Survivable networks**
Weiskircher, R. *see:* **Optimization in leveled graphs**
Weiszfeld, E. *see:* **Global optimization in Weber's problem with attraction and repulsion**
Welsch, R.E. *see:* **Nonlinear least squares: Newton-type methods**
Welsh, D.J.A. *see:* **Matroids**
Welzl, E. *see:* **Single facility location: circle covering problem**
Wengert, R. *see:* **Nonlocal sensitivity analysis with automatic differentiation**
Wengert, R.E. *see:* **Automatic differentiation: introduction, history and rounding error estimation**; **Automatic differentiation: point and interval**
Werbos, P.J. *see:* **Automatic differentiation: introduction, history and rounding error estimation**
Werschulz, A.G. *see:* **Information-based complexity and information-based optimization**
Wesler, O. *see:* **Linear complementarity problem**
Wesolowsky, G.O. *see:* **Global optimization in Weber's problem with attraction and repulsion**
Westerberg, A. *see:* **Successive quadratic programming: applications in the process industry**
Westerlund, T. *see:* **MINLP: outer approximation algorithm**; **MINLP: trim-loss problem**
Westervelt, R.M. *see:* **Replicator dynamics in combinatorial optimization**
Wets, R.J-B. *see:* **History of optimization**; **Monte-Carlo simulations for stochastic optimization**; **Random search methods**; **Simple recourse problem: primal method**; **Splitting method for linear complementarity problems**; **Stochastic programming: nonanticipativity and lagrange multipliers**; **Stochastic programs with recourse: upper bounds**
Weyl, H. *see:* **Tucker homogeneous systems of linear relations**
Whalley, J. *see:* **Walrasian price equilibrium**
Whang, S. *see:* **Operations research models for supply chain management and design**
Wheelwright, S.C. *see:* **Forecasting**
White, D.J. *see:* **Bilevel linear programming**; **Bi-objective assignment problem**
Whitin, T.M. *see:* **Inventory management in supply chains**
Whitney, H. *see:* **Matroids**; **Oriented matroids**
Wickwire, K. *see:* **Resource allocation for epidemic control**
Widhelm, W.B. *see:* **Nondifferentiable optimization: cutting plane methods**
Wiecek, M.M. *see:* **Lemke method**; **Multiple objective dynamic programming**
Wiener, N. *see:* **Semi-infinite programming and applications in finance**
Wierzbicki, A. *see:* **Multiple objective programming support**
Wikner, J. *see:* **Operations research models for supply chain management and design**
Wilde, D.J. *see:* **Global optimization in generalized geometric programming**
Wilhelm, C.E. *see:* **Interval analysis: application to chemical engineering design problems**

Wille, L.T. *see:* **Global optimization in Lennard–Jones and morse clusters**
Willems, S. *see:* **Operations research models for supply chain management and design**
Williams, H.C.L.W. *see:* **Single facility location: multi-objective rectilinear distance location**
Williams, H.P. *see:* **Disjunctive programming**; **Integer programming duality**
Williams, J.B. *see:* **Portfolio selection: markowitz mean-variance model**
Williamson, D.P. *see:* **Feedback set problems**
Williamson, K.A. *see:* **Optimal design in nonlinear optics**
Wilson, J.M. *see:* **Multicriteria sorting methods**
Wilson, R.B. *see:* **Successive quadratic programming**; **Successive quadratic programming: full space methods**
Wiman, A. *see:* **Hilbert's thirteenth problem**
Winsten, C.B. *see:* **Dynamic traffic networks**; **Equilibrium networks**; **Traffic network equilibrium**
Winston, W.L. *see:* **Interval global optimization**
Wirth, A. *see:* **Survivable networks**
Wisse, A. *see:* **Frequency assignment problem**
Witkin, A. *see:* **Optimization in medical imaging**
Witzgall, C. *see:* **Isotonic regression problems**
Woeginger, G.J. *see:* **Communication network assignment problem**
Wolfe, M. *see:* **Alignment problem**
Wolfe, P. *see:* **Convex-simplex algorithm**; **Decomposition principle of linear programming**; **Frank–Wolfe algorithm**; **History of optimization**; **Lagrangian duality: BASICS**; **Lexicographic pivoting rules**; **Nondifferentiable optimization**; **Rosen's method, global convergence, and Powell's conjecture**
Wolkowicz, H. *see:* **Quadratic programming over an ellipsoid**
Wolsey, L. *see:* **Integer programming: algebraic methods**; **Integer programming: cutting plane algorithms**
Wong, K.T. *see:* **Suboptimal control**
Wong, R.T. *see:* **Time-dependent traveling salesman problem**
Wood, G.R. *see:* **Random search methods**
Wood, R.K. *see:* **Stochastic programs with recourse: upper bounds**
Wormald, N. *see:* **Optimization in leveled graphs**
Worsley, K.J. *see:* **Approximation of multivariate probability integrals**
Woźniakowski, H. *see:* **Information-based complexity and information-based optimization**
Wrathall, C. *see:* **Automatic differentiation: introduction, history and rounding error estimation**
Wright, M.H. *see:* **Successive quadratic programming: decomposition methods**
Wu, C.H. *see:* **Simplicial decomposition**
Wu, T.S. *see:* **Survivable networks**
Wu, W.-T. *see:* **Minimax theorems**

X

Xia, Z. *see:* **ABS algorithms for optimization**
Xu, C.X. *see:* **Nonlinear least squares: Newton-type methods**; **Nonlinear least squares: trust region methods**
Xu, J. *see:* **Successive quadratic programming: applications in distillation systems**; **Successive quadratic programming: full space methods**; **Successive quadratic programming: solution by active sets and interior point methods**
Xu, X. *see:* **Homogeneous selfdual methods for linear programming**
Xue, G. *see:* **Single facility location: circle covering problem**
Xue, G.H. *see:* **Steiner tree problems**
Xue, G.L. *see:* **Global optimization in Lennard–Jones and morse clusters**; **Global optimization in Weber's problem with attraction and repulsion**

Y

Yabe, H. *see:* **Nonlinear least squares: Newton-type methods**
Yablonskii, S.V. *see:* **Finite complete systems of many-valued logic algebras**
Yadegar, J. *see:* **Quadratic assignment problem**
Yaged, B. *see:* **Nonconvex network flow problems**
Yakowitz, S. *see:* **Resource allocation for epidemic control**
Yakubovich, V.A. *see:* **Robust control**
Yamada, J. *see:* **Survivable networks**
Yamashita, H. *see:* **Successive quadratic programming: applications in the process industry**
Yamashita, N. *see:* **Implicit lagrangian**
Yang, X.Q. *see:* **Composite nonsmooth optimization**
Yannacopoulos, D. *see:* **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**
Yannakakis, M. *see:* **Steiner tree problems**
Yano, C.A. *see:* **Operations research models for supply chain management and design**
Yanovskaya, E.B. *see:* **Minimax theorems**
Yao, J.C. *see:* **Generalized monotone multivalued maps**; **Generalized monotonicity: applications to variational inequalities and equilibrium problems**
Yao, Y.C. *see:* **Steiner tree problems**
Ye, Y. *see:* **Complexity theory: quadratic programming**; **Homogeneous selfdual methods for linear programming**; **Linear programming: interior point methods**; **Linear programming: karmarkar projective algorithm**; **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities**; **Quadratic knapsack**
Yee, T.F. *see:* **MINLP: heat exchanger network synthesis**
Yorke, J.A. *see:* **Globally convergent homotopy methods**; **Resource allocation for epidemic control**
Yoshise, Y. *see:* **Linear programming: interior point methods**
Young, F.W. *see:* **Preference disaggregation**
Young, L.C. *see:* **Duality in optimal control with first order differential equations**; **Semi-infinite programming and control problems**
Young, W.H. *see:* **Young programming**
Yu, G. *see:* **Capacitated minimum spanning trees**
Yu, W. *see:* **Multicriteria sorting methods**; **Rosen's method, global convergence, and Powell's conjecture**
Yuan, Y. *see:* **Rosen's method, global convergence, and Powell's conjecture**
Yuan, Y.X. *see:* **Implicit lagrangian**
Yudin, D.B. *see:* **Complexity theory**; **Complexity theory: quadratic programming**; **Information-based complexity and information-based optimization**

Yue, M. *see:* **Rosen's method, global convergence, and Powell's conjecture**

Z

Zabinsky, Z.B. *see:* **Random search methods**
Žáčková, J. *see:* **Stochastic programming: minimax approach**
Zadeh, L.A. *see:* **Boolean and fuzzy relations**; **Checklist paradigm semantics for fuzzy logics**
Zames, G. *see:* **Robust control**
Zanakis, S.H. *see:* **Multidimensional knapsack problems**
Zang, I. *see:* **Discontinuous optimization**
Zang, W. *see:* **Feedback set problems**
Zangwill, W. *see:* **Traffic network equilibrium**
Zangwill, W.I. *see:* **Convex-simplex algorithm**; **Rosen's method, global convergence, and Powell's conjecture**
Zaremba, C. *see:* **Variational principles**
Zaric, G.S. *see:* **Resource allocation for epidemic control**
Zeleny, M. *see:* **Portfolio selection and multicriteria analysis**
Zelikovsky, A.Z. *see:* **Steiner tree problems**
Zha, G. *see:* **Design optimization in computational fluid dynamics**
Zhang, D. *see:* **Dynamic traffic networks**; **Variational inequalities: projected dynamical system**
Zhang, J.Z. *see:* **Nonlinear least squares: trust region methods**; **Numerical methods for unary optimization**
Zhang, L. *see:* **ABS algorithms for linear equations and linear least squares**
Zhang, S. *see:* **Criss-cross pivoting rules**; **Fractional programming**
Zhang, X.-S. *see:* **Rosen's method, global convergence, and Powell's conjecture**
Zhang, Y. *see:* **Linear programming: interior point methods**; **Minimax game tree searching**
Zhao, Y. *see:* **Operations research and financial markets**
Zhi-Quan Luo *see:* **Semi-infinite programming, semidefinite programming and perfect duality**
Zhou, S. *see:* **Global optimization: application to phase equilibrium problems**
Ziarati, K. *see:* **Multicommodity flow problems**
Ziemba, W.T. *see:* **Portfolio selection: markowitz mean-variance model**
Zierer, H. *see:* **Boolean and fuzzy relations**
Žilinskas, A. *see:* **Adaptive global search**
Zima, H.P. *see:* **Alignment problem**
Zimmermann, H.J. *see:* **Fuzzy multi-objective linear programming**
Zimmermann, S. *see:* **Eigenvalue enclosures for ordinary differential equations**
Ziont, S. *see:* **Criss-cross pivoting rules**
Zipf, H.P. *see:* **Optimization in classifying text documents**
Zipkin, P. *see:* **Operations research models for supply chain management and design**
Zippel, R. *see:* **Complexity theory: quadratic programming**; **Quadratic programming over an ellipsoid**
Zisswiller, R. *see:* **Portfolio selection and multicriteria analysis**
Zoellner, J.A. *see:* **Frequency assignment problem**
Zoon, J.A. *see:* **Multidimensional knapsack problems**
Zopounidis, C. *see:* **Portfolio selection and multicriteria analysis**; **Preference disaggregation**; **Preference disaggregation approach: basic features, examples from financial decision making**
Zosin, L. *see:* **Feedback set problems**
Zoutendijk, G. *see:* **History of optimization**; **Principal pivoting methods for linear complementarity problems**
Zwick, U. *see:* **Maximum satisfiability problem**